ML_HW3

April 11, 2019

```
In [1]: # Load the Drive helper and mount
        from google.colab import drive
        drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

```
In [2]: import os
        import numpy as np
        from os import listdir
        from os.path import isfile, join

        image_files = ['/content/drive/My Drive/yalefaces_ml/' + f for f in
        listdir('/content/drive/My Drive/yalefaces_ml')]
        total_no_images = len(image_files)
        print("Total number of images : ",total_no_images)
```

```
Total number of images :  165
```

```
In [0]: # Function to load the images
        from PIL import Image

        def load_image(infilename) :
            img = Image.open( infilename )
            img = img.resize((120,120))
            data = np.array(img, dtype ='float')
            data = data.flatten()
            return data
```

```
In [0]: # input
        X = np.empty((total_no_images,14400))
```

```
In [0]: #  normalize the data
        from sklearn.preprocessing import scale
        i = 0

        for image in image_files:
            data = load_image(image)
            X[i:] = scale(data)
            i = i + 1
```

```
In [6]: X.shape
```

```
Out[6]: (165, 14400)
```

```
In [7]: X
```

```
Out[7]: array([[0.49428811, 0.87158227, 0.92398424, ..., 0.78773913, 0.79821952,
         0.35804299],
        [0.74553227, 0.74553227, 0.74553227, ..., 0.74553227, 0.74553227,
         0.74553227],
        [0.68639664, 0.68639664, 0.68639664, ..., 0.68639664, 0.68639664,
         0.68639664],
        ...,
        [0.69592428, 0.69592428, 0.69592428, ..., 0.69592428, 0.69592428,
         0.53496968],
        [0.80523897, 0.80523897, 0.80523897, ..., 0.80523897, 0.80523897,
         0.80523897],
        [0.7173607 , 0.7173607 , 0.7173607 , ..., 0.7173607 , 0.7173607 ,
         0.7173607 ]])
```

```
In [8]: # Keep the mean to use later
        x0 = np.mean(X, axis=0)
        print(x0.shape)
```

```
(14400,)
```

```
In [0]: X_orig = X.copy()
        # Covariance matrix
        S = np.cov(X.T)
```

```
In [10]: print('Shape of the covariance matrix is ', S.shape)
```

```
Shape of the covariance matrix is  (14400, 14400)
```

```
In [11]: S
```

```
Out[11]: array([[ 0.02961649,  0.02324548,  0.02213133, ..., -0.04181545,
          -0.03895638, -0.03401127],
         [ 0.02324548,  0.02907016,  0.02652417, ..., -0.03144387,
          -0.03004873, -0.03203301],
         [ 0.02213133,  0.02652417,  0.02869489, ..., -0.02732674,
          -0.02621114, -0.02719972],
         ...,
         [-0.04181545, -0.03144387, -0.02732674, ...,  0.37548951,
           0.37690183,  0.37478871],
         [-0.03895638, -0.03004873, -0.02621114, ...,  0.37690183,
           0.41471389,  0.41248425],
         [-0.03401127, -0.03203301, -0.02719972, ...,  0.37478871,
           0.41248425,  0.43434604]])
```

```
In [0]: # Question 2.a Implement Principal Component Analysis (PCA)
        import time
        start = time.time()
        from numpy import linalg as LA
        values, vectors = LA.eig(S)
        end = time.time()
```

```
In [0]: pwd
```

In [0]:
```
#https://stackoverflow.com/questions/8092920/sort-eigenvalues-and-associated-
eigenvectors-after-using-numpy-linalg-eig-in-pyt
#np.save('/content/drive/My Drive/eigenvalues', values)
#np.save('/content/drive/My Drive/eigenvectors', vectors)

values = np.load('/content/drive/My Drive/eigenvalues.npy')

vectors = np.load('/content/drive/My Drive/eigenvectors.npy')
```

In [13]:
```
values.shape
```

Out[13]:
```
(14400,)
```

In [14]:
```
vectors.shape
```

Out[14]:
```
(14400, 14400)
```

In [0]:
```
sorted_idx = values.argsort()
```

In [16]:
```
sorted_idx
```

Out[16]:
```
array([156, 155, 163, ...,   2,   1,   0])
```

In [0]:
```
sorted_idx = sorted_idx[::-1]
```

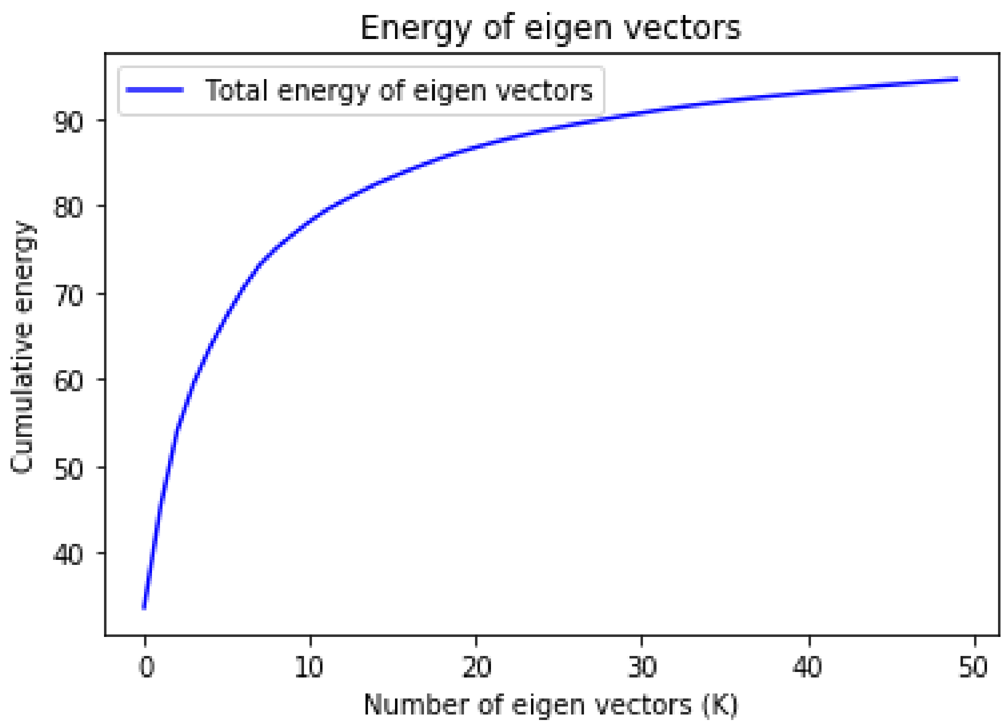
In [18]:
```
sorted_idx
```

Out[18]:
```
array([  0,   1,   2, ..., 163, 155, 156])
```

In [19]:
```
sum_all_values = np.sum(values)
print('sum of all the eigenvalues is : ', sum_all_values)
sum = 0
energies = []
for K in range(50):
    idx = sorted_idx[K]
    sum = sum + values[idx]
    energy = (sum*100)/sum_all_values
    energy = round(energy, 2)
    energies.append(energy)

print (energies)

import matplotlib.pyplot as plt

K_Values = range(50)
plt.plot(K_Values, energies, 'b', label='Total energy of eigen vectors')
plt.title('Energy of eigen vectors')
plt.xlabel('Number of eigen vectors (K)')
plt.ylabel('Cumulative energy')
plt.legend()

plt.show()
```

```
sum of all the eigenvalues is :  (6883.292283357105-2.938506871948269e-29j)
[(33.73+0j), (45.42+0j), (54.06+0j), (59.56+0j), (63.74+0j), (67.31+0j), (70.52+0j),
(73.18+0j), (75.08+0j), (76.64+0j), (78.12+0j), (79.43+0j), (80.48+0j), (81.47+0j),
(82.43+0j), (83.25+0j), (84.04+0j), (84.77+0j), (85.48+0j), (86.09+0j), (86.64+0j),
(87.17+0j), (87.66+0j), (88.11+0j), (88.54+0j), (88.93+0j), (89.3+0j), (89.64+0j),
(89.97+0j), (90.29+0j), (90.6+0j), (90.91+0j), (91.18+0j), (91.45+0j), (91.7+0j),
(91.94+0j), (92.16+0j), (92.37+0j), (92.57+0j), (92.76+0j), (92.96+0j), (93.14+0j),
(93.32+0j), (93.49+0j), (93.66+0j), (93.82+0j), (93.98+0j), (94.14+0j), (94.29+0j),
(94.43+0j)]
```

```
/usr/local/lib/python3.6/dist-packages/numpy/core/numeric.py:492: ComplexWarning:
Casting complex values to real discards the imaginary part
  return array(a, dtype, copy=False, order=order)
```

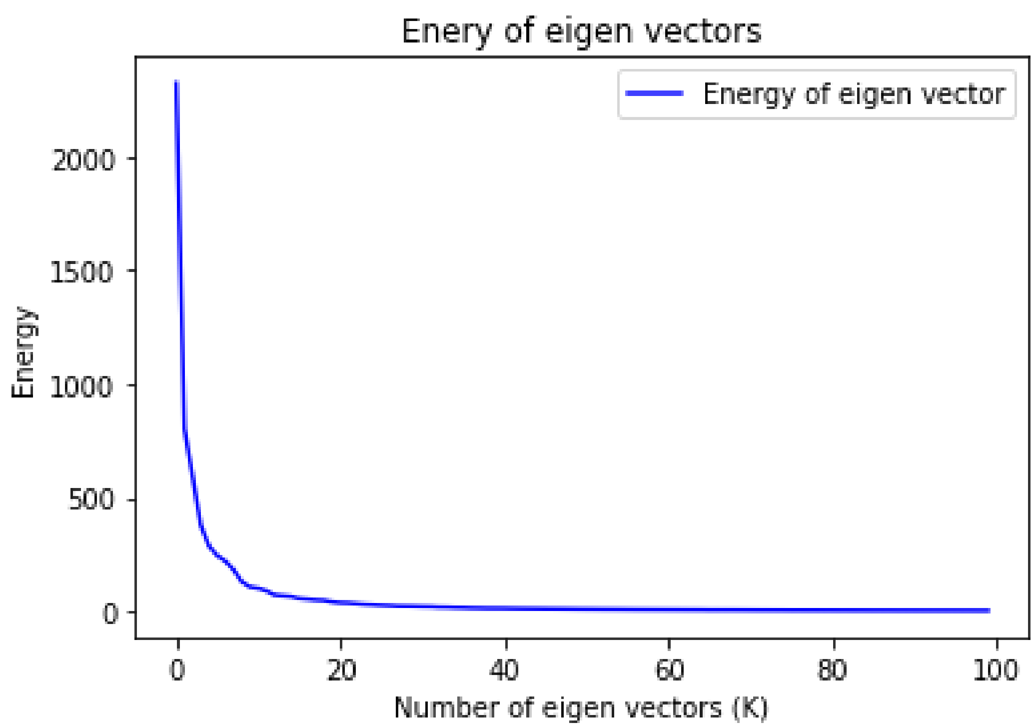



```
In [20]: absolute_energy = []

         for K in range(100):
             idx = sorted_idx[K]
             absolute_energy.append(values[idx])

         print (absolute_energy)

         K_Values = range(100)
         plt.plot(K_Values, absolute_energy, 'b', label='Energy of eigen vector')
         plt.title('Enery of eigen vectors')
         plt.xlabel('Number of eigen vectors (K)')
         plt.ylabel('Energy')
         plt.legend()

         plt.show()
```

```
[(2321.5613929688184+0j), (804.6496873800984+0j), (594.5740977550016+0j),
(378.9400434076328+0j), (287.3896127456431+0j), (245.71402601771157+0j),
(221.12068394839451+0j), (183.33478883686942+0j), (130.5264634307286+0j),
(107.40994285621112+0j), (101.75046566270586+0j), (90.48239896499516+0j),
(71.9894343963276+0j), (68.59072949923234+0j), (65.61131537462441+0j),
(56.87025945295701+0j), (54.04756717423103+0j), (50.401116066498204+0j),
(48.68669451687862+0j), (41.85343376700608+0j), (37.95778093541738+0j),
(36.881953454263254+0j), (33.65404085874752+0j), (30.822617678653362+0j),
(29.308873905177524+0j), (26.963711081171937+0j), (25.501483470489042+0j),
```

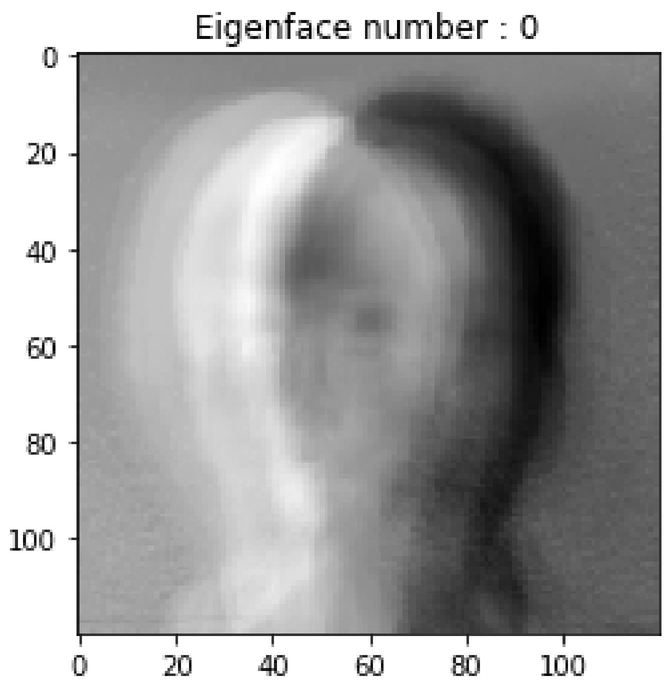
```
(23.338391978465097+0j), (22.843510078394093+0j), (22.32028190005816+0j),
(21.32571748917328+0j), (20.856114328563564+0j), (18.942744495797978+0j),
(18.531283531961176+0j), (17.21004287266184+0j), (16.331120266168345+0j),
(15.033340512344296+0j), (14.520048497091897+0j), (13.772824225758974+0j),
(13.644963928830666+0j), (13.366240014081106+0j), (12.674129457267213+0j),
(12.2276773361947l+0j), (11.904694593574193+0j), (11.647764945161127+0j),
(11.158922326319672+0j), (10.848056564148683+0j), (10.594887825912737+0j),
(10.270856787482538+0j), (9.89899656125189+0j), (9.658407646772472+0j),
(9.400227911371534+0j), (9.098313446195363+0j), (8.947390327943065+0j),
(8.772019720707853+0j), (8.446624782798358+0j), (8.141706038737269+0j),
(7.977107441870344+0j), (7.876263046003448+0j), (7.6623153653331135+0j),
(7.512239574954662+0j), (7.3953181829013825+0j), (7.174404333713891+0j),
(7.085868324474558+0j), (6.925613990114329+0j), (6.491562603998879+0j),
(6.429340693894445+0j), (6.410413057051207+0j), (6.356516628579197+0j),
(6.180326825454417+0j), (5.9909332348003+0j), (5.786829152745979+0j),
(5.668738154612552+0j), (5.589914409117323+0j), (5.463296862635242+0j),
(5.214917598457307+0j), (5.174578762276769+0j), (5.0393188133514615+0j),
(4.955654886691408+0j), (4.798025310032333+0j), (4.7652887641846995+0j),
(4.614123822548559+0j), (4.540468592697327+0j), (4.401237967499656+0j),
(4.3095025525815105+0j), (4.1514973222885825+0j), (4.116490410028322+0j),
(4.052558267709568+0j), (3.93578762058289+0j), (3.8953235536678807+0j),
(3.8262319344470503+0j), (3.7358809518742238+0j), (3.6872099306399226+0j),
(3.6426322408427723+0j), (3.581303115997521+0j), (3.5514868126812336+0j),
(3.4691760338717565+0j), (3.4120594298955473+0j), (3.284238631457151+0j),
(3.250320865938476+0j)]
```

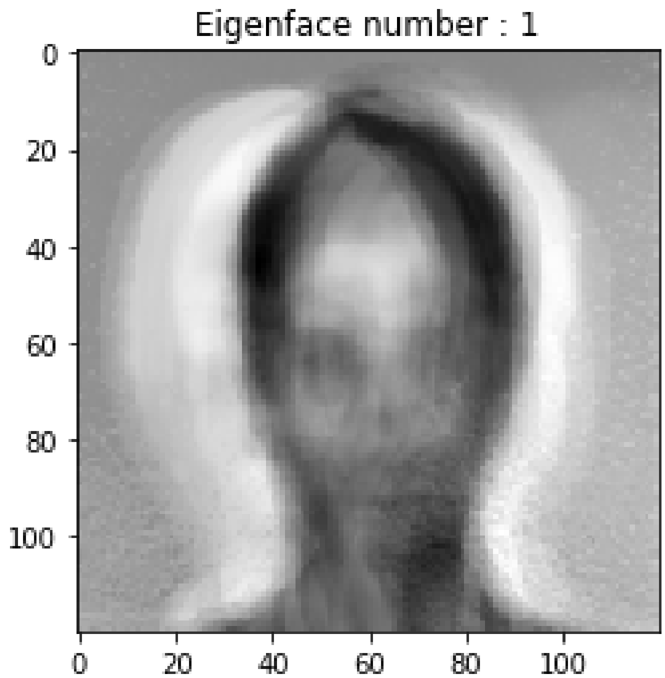
```
/usr/local/lib/python3.6/dist-packages/numpy/core/numeric.py:492: ComplexWarning:
Casting complex values to real discards the imaginary part
  return array(a, dtype, copy=False, order=order)
```

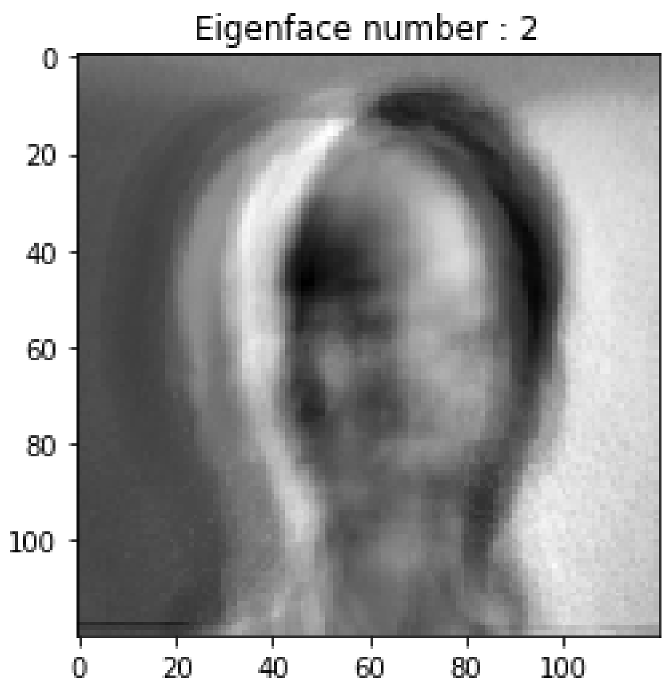


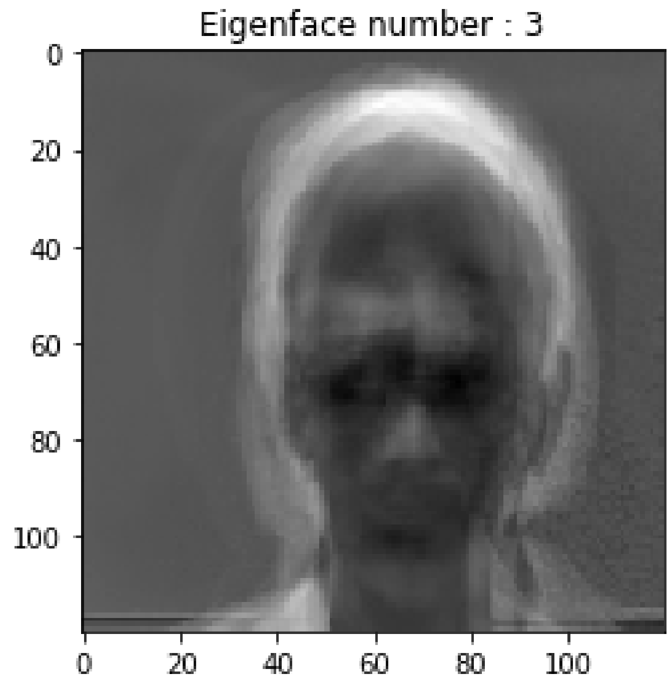
In [0]:
```
# Plot the top 10 eigenfaces, i.e. the eigenvectors uk, k = 1,..., 10 obtained by PCA.

top_100 = np.empty((14400, 100))

for K in range(100):
    idx = sorted_idx[K]
    eigenVector = vectors[:,idx]
    top_100[:,K] = np.real(eigenVector)
```

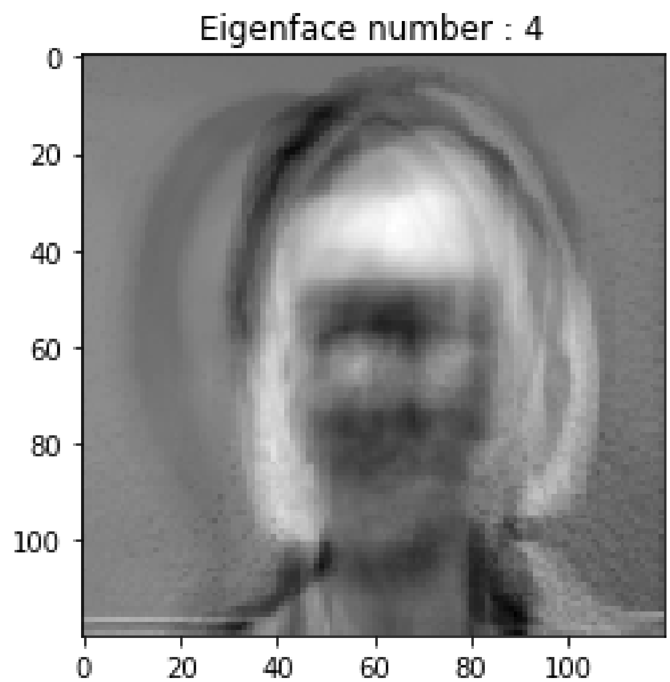
In [24]:
```
# https://matplotlib.org/users/image_tutorial.html
for vec_num in range(10):
    img = top_100[:,vec_num]
    imgplot = plt.imshow(img.reshape(120,120))
    imgplot.set_cmap('gray')
    plt.title('Eigenface number : %d'% vec_num)
    plt.show()
```

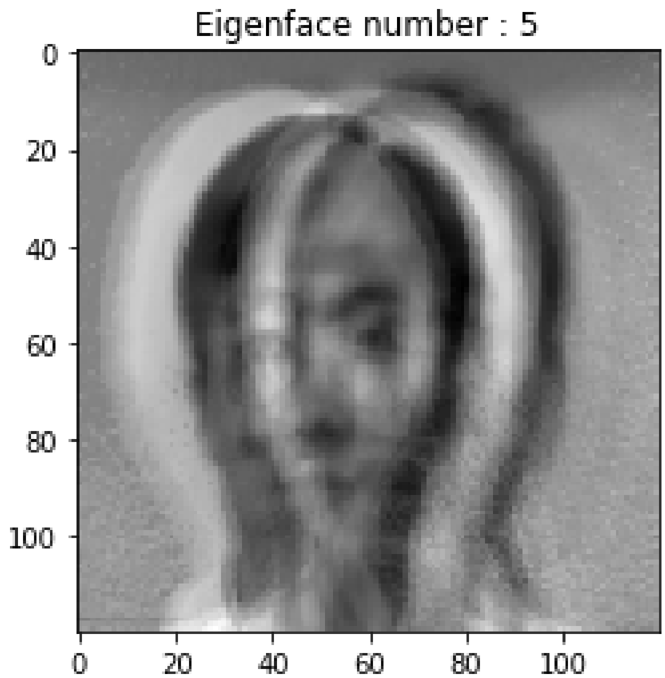


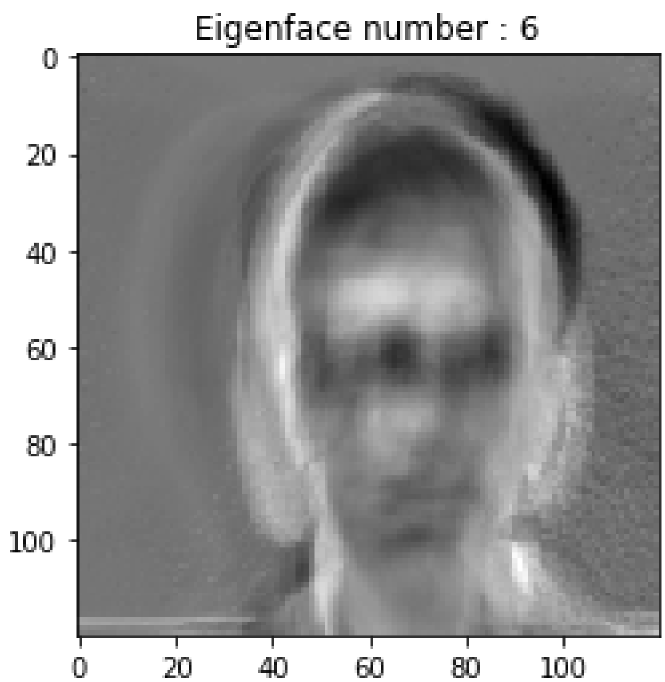

Eigenface number : 1
0
20
40
60
80
100
0
20
40
60
80
100

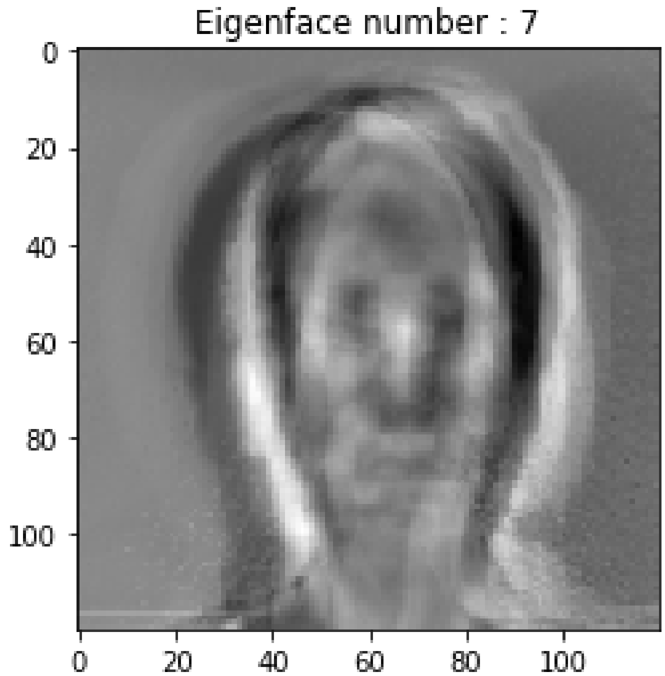

Eigenface number : 2
0
20
40
60
80
100
0
20
40
60
80
100

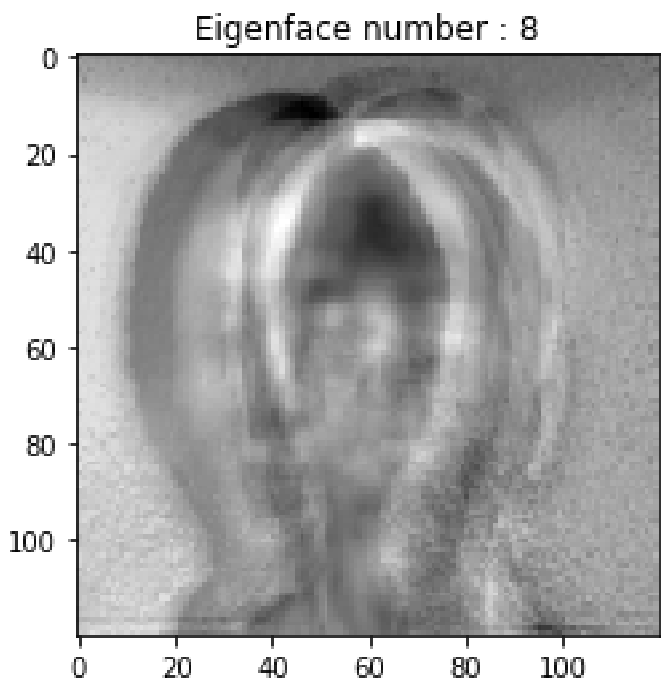

Eigenface number : 3
0
20
40
60
80
100
0
20
40
60
80
100

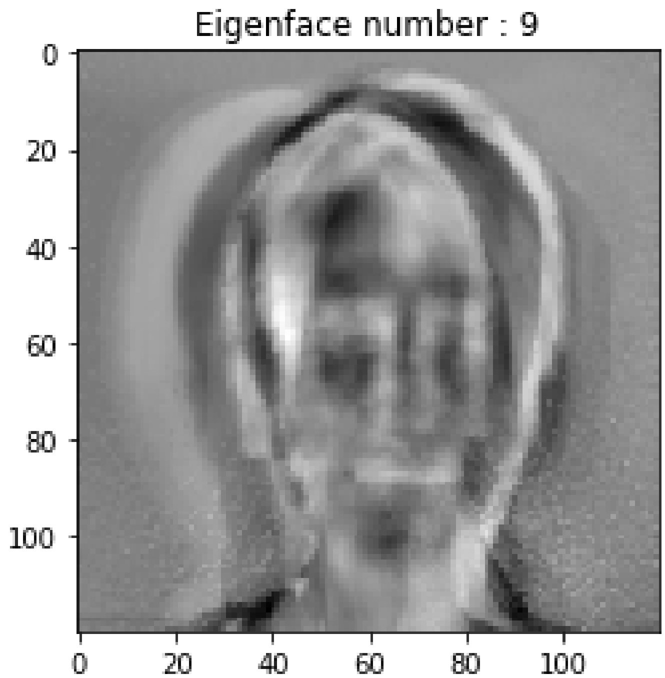

Eigenface number : 4
0
20
40
60
80
100
0
20
40
60
80
100


Eigenface number : 5
0
20
40
60
80
100
0
20
40
60
80
100

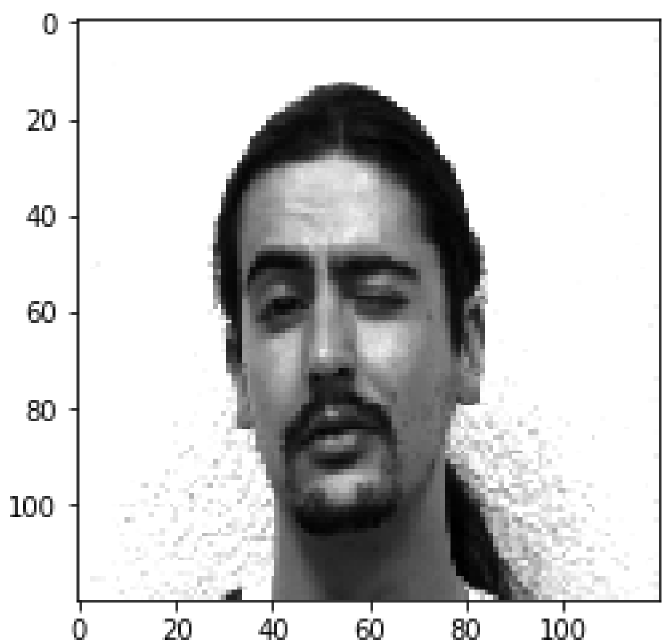

Eigenface number : 6
0
20
40
60
80
100
0
20
40
60
80
100

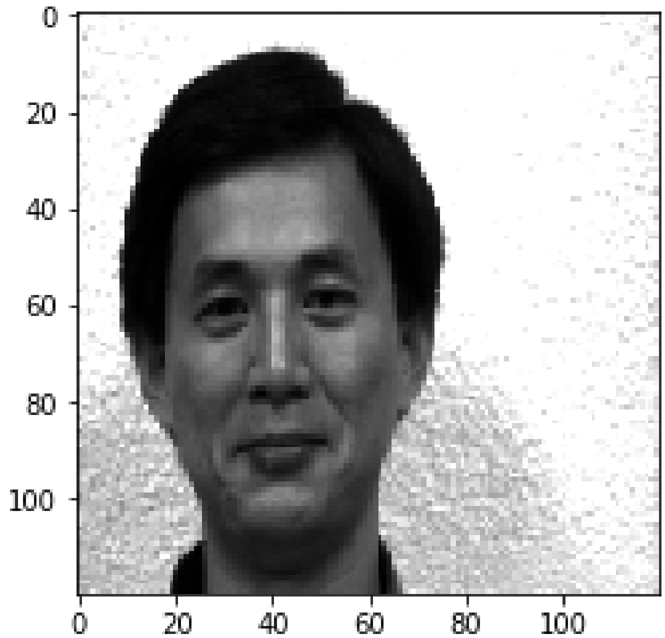

Eigenface number : 7
0
20
40
60
80
100
0
20
40
60
80
100


Eigenface number : 8
0
20
40
60
80
100
0
20
40
60
80
100

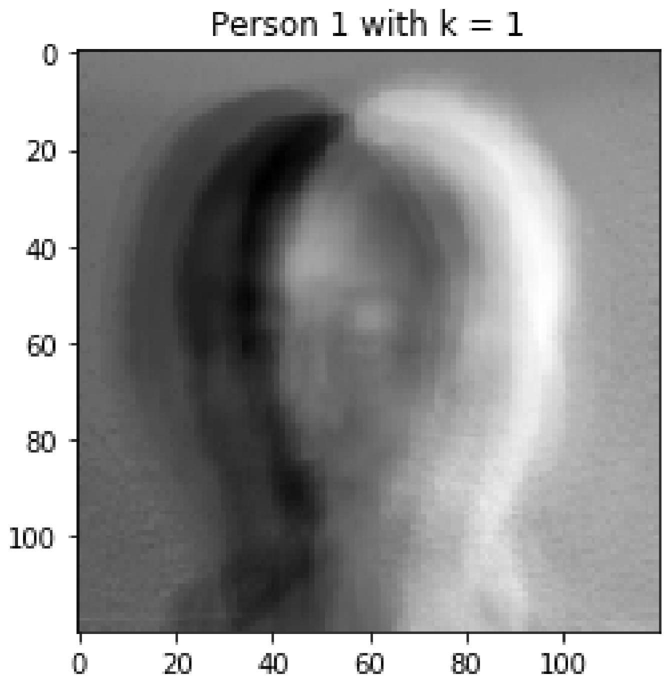


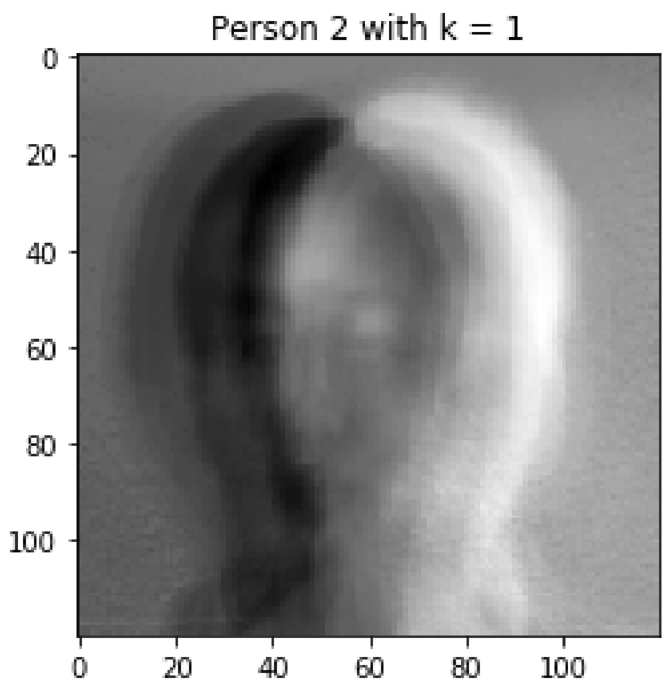

```
In [55]: # real image

         images_real = [load_image(image_files[7]), load_image(image_files[15])]
         for real_image in images_real:
             imgplot = plt.imshow(real_image.reshape(120,120))
             imgplot.set_cmap('gray')
             plt.show()
```

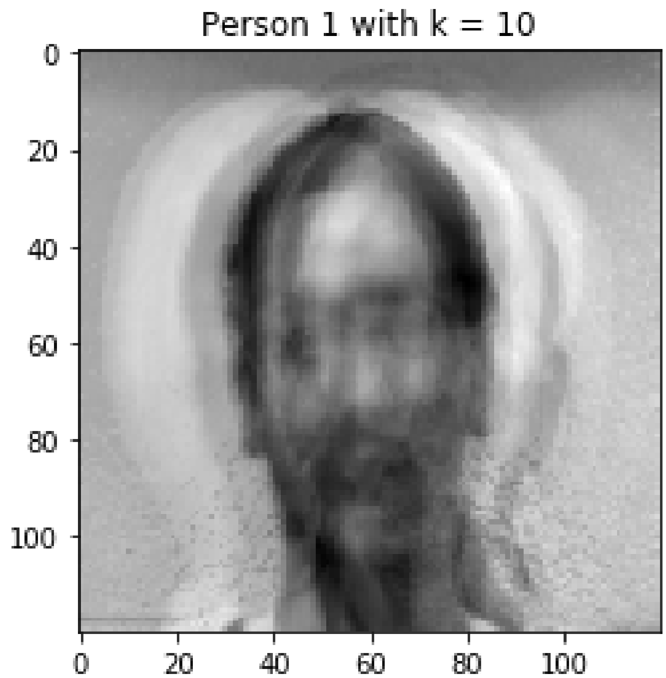


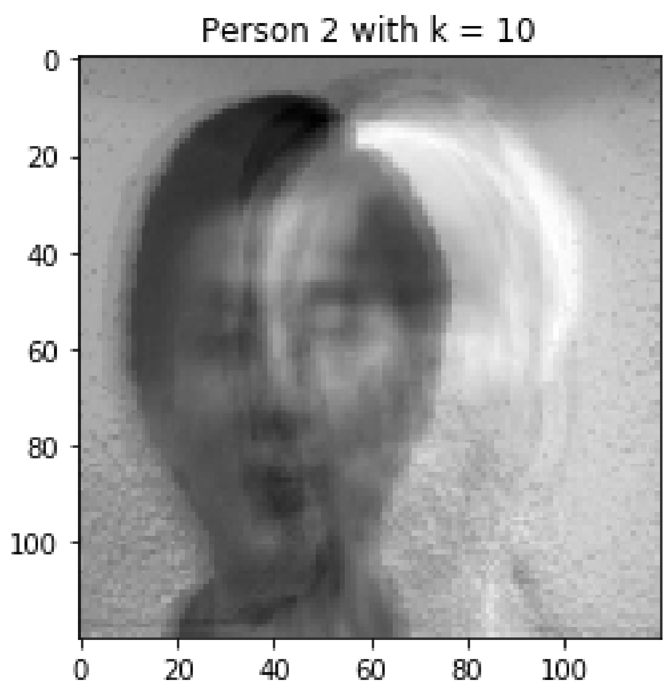


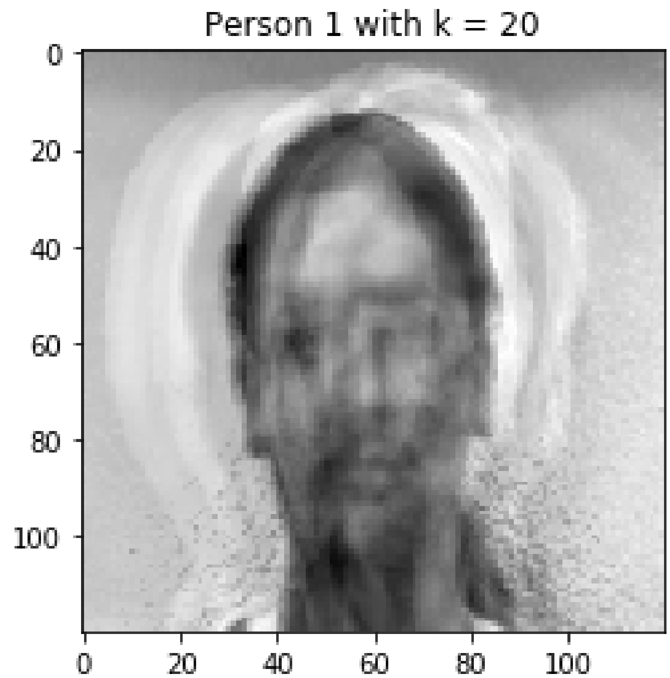
In [56]:
```
components = [1, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

for k in components:
    p = 1
    for real_image in images_real:
        add = np.zeros((14400,))
        for i in range(k):
            u = top_100[:,i]
            # real_image = scale(real_image)
            add = add + np.dot(u.T, real_image) * u

        image = x0 + add
        imgplot = plt.imshow(image.reshape(120,120))
        imgplot.set_cmap('gray')
        plt.title('Person %d with k = %d' % (p, k))
        plt.show()
        p = p + 1
```

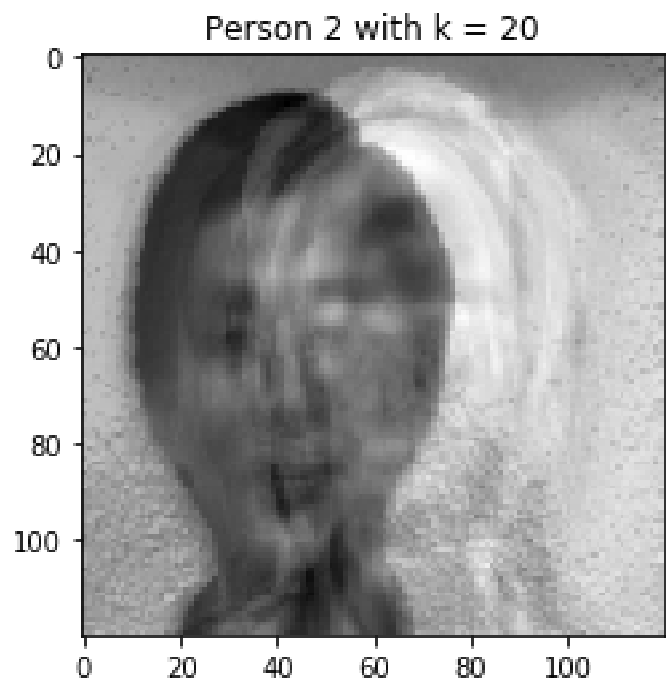

Person 1 with k = 1
0
20
40
60
80
100
0
20
40
60
80
100

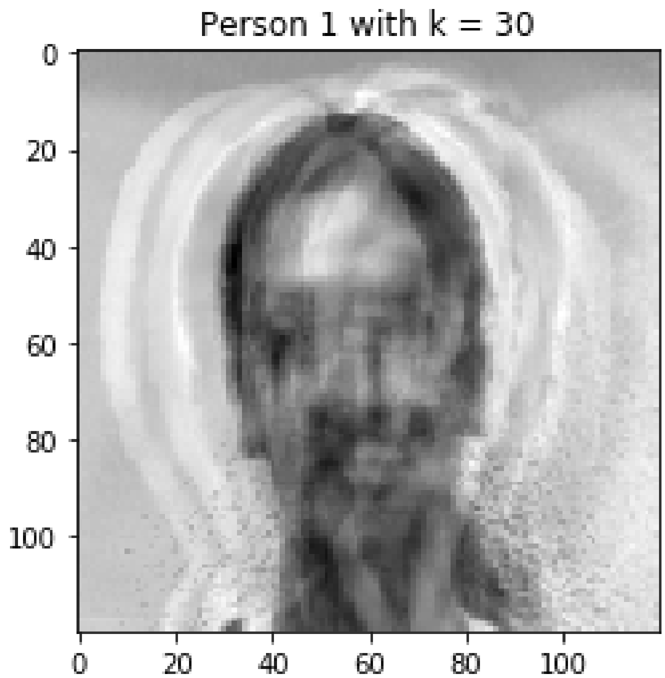

Person 2 with k = 1
0
20
40
60
80
100
0
20
40
60
80
100

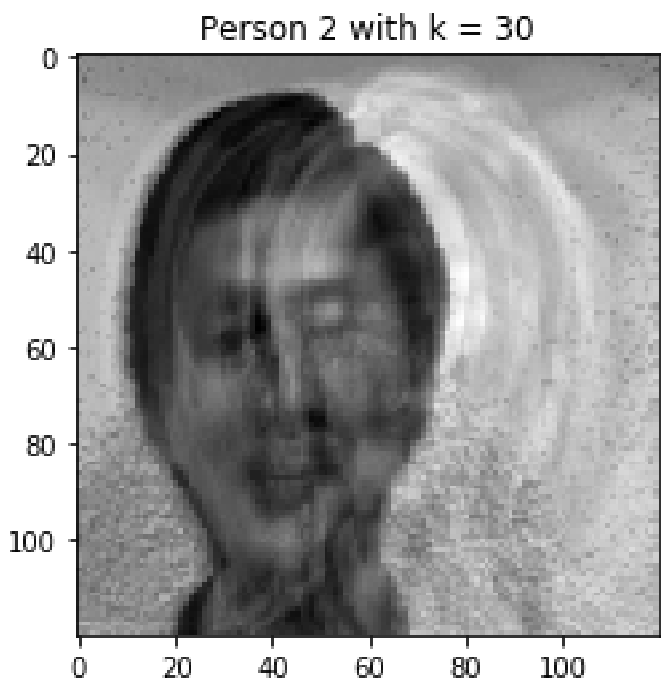

Person 1 with k = 10
0
20
40
60
80
100
0
20
40
60
80
100

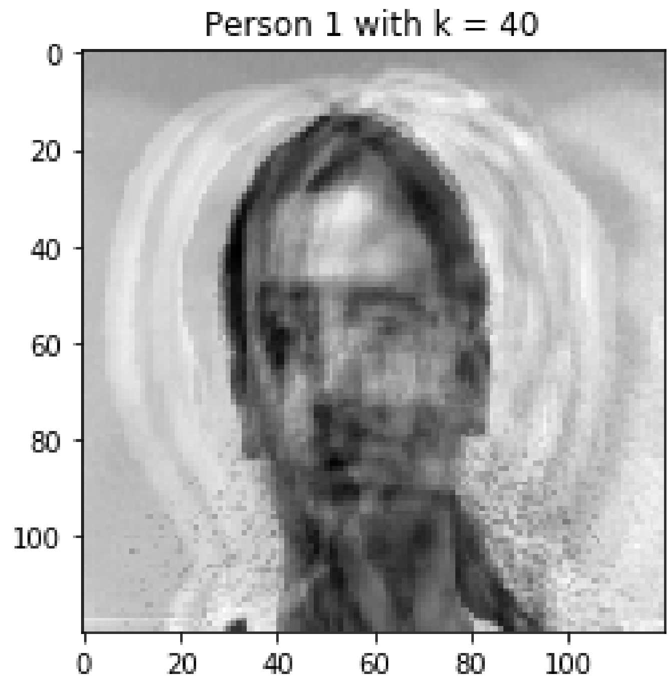

Person 2 with k = 10
0
20
40
60
80
100
0
20
40
60
80
100

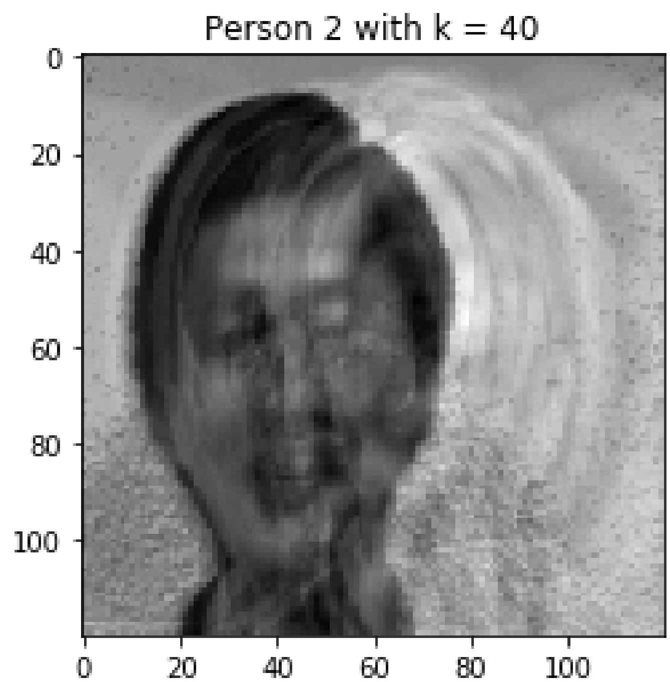

Person 1 with k = 20
0
20
40
60
80
100
0
20
40
60
80
100

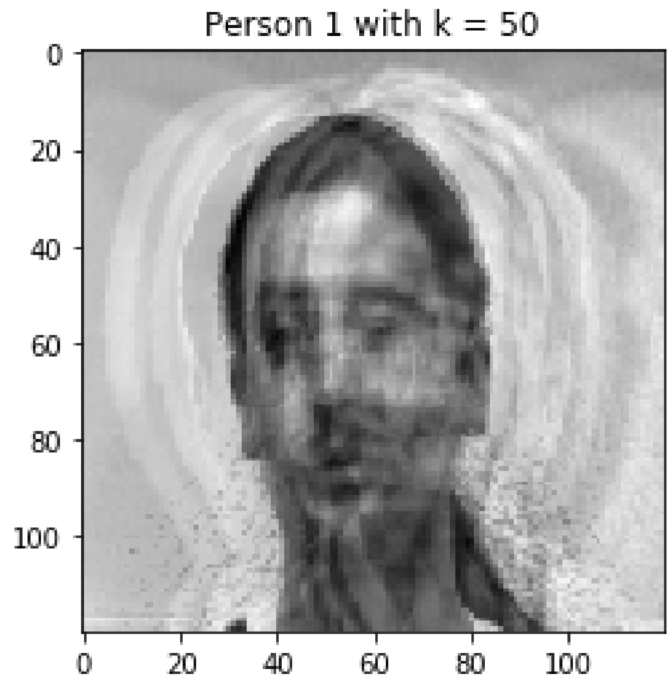

Person 2 with k = 20
0
20
40
60
80
100
0
20
40
60
80
100

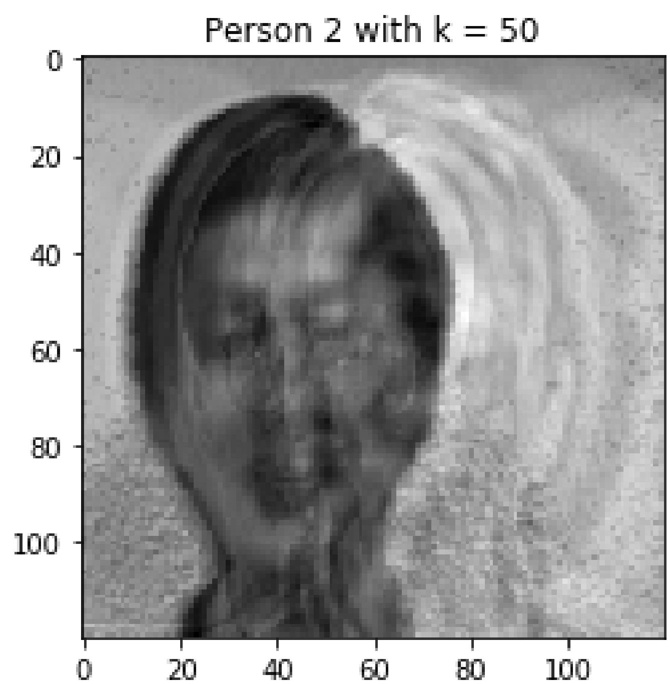

Person 1 with k = 30
0
20
40
60
80
100
0
20
40
60
80
100

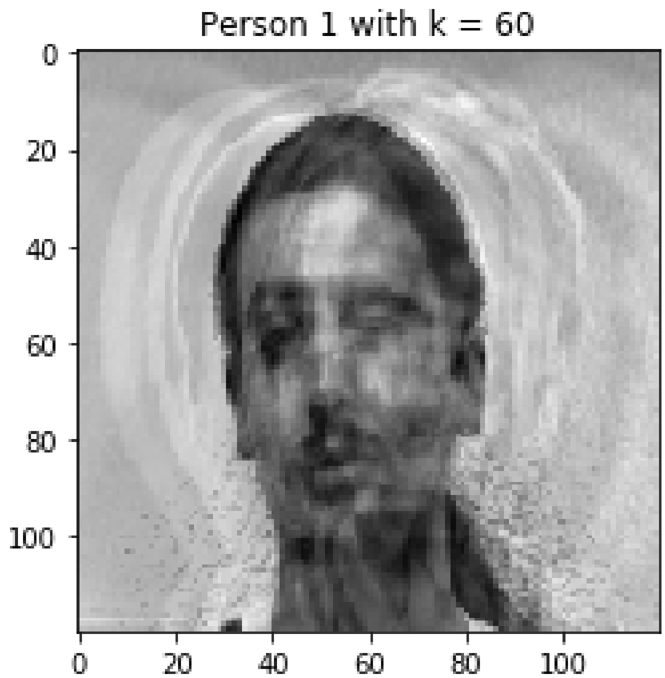

Person 2 with k = 30
0
20
40
60
80
100
0
20
40
60
80
100

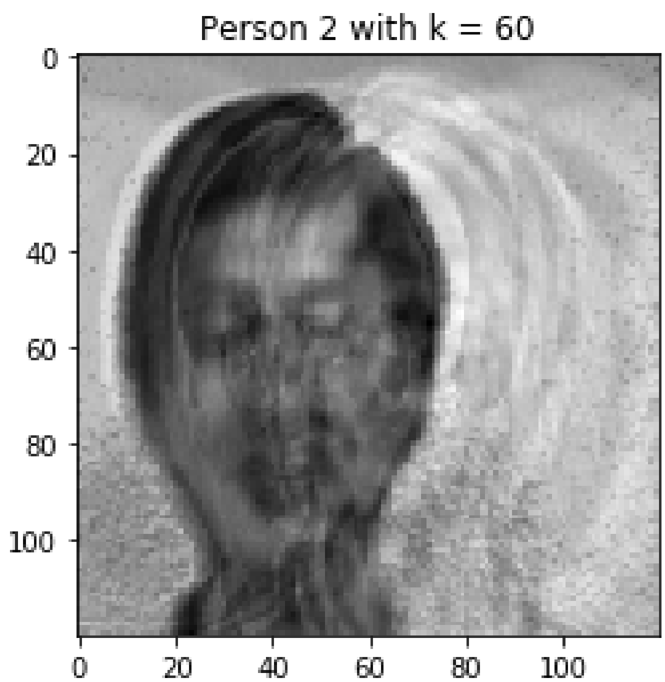

Person 1 with k = 40
0
20
40
60
80
100
0
20
40
60
80
100

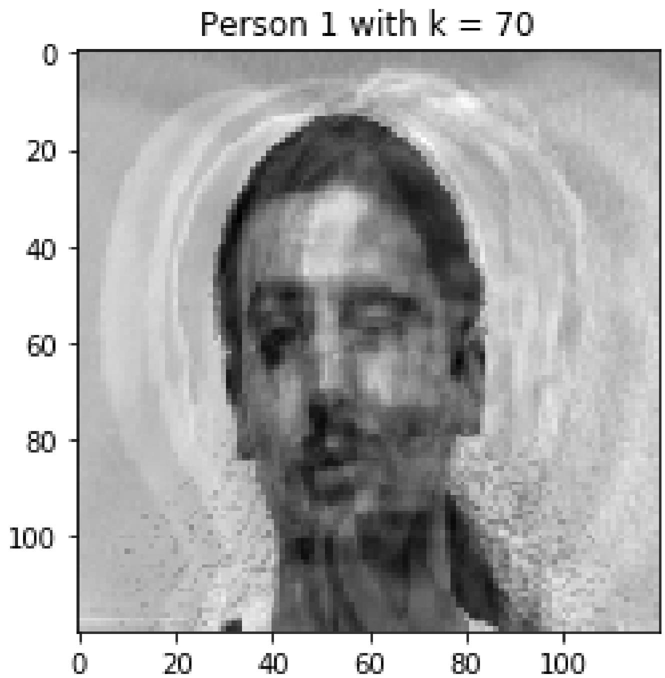

Person 2 with k = 40
0
20
40
60
80
100
0
20
40
60
80
100

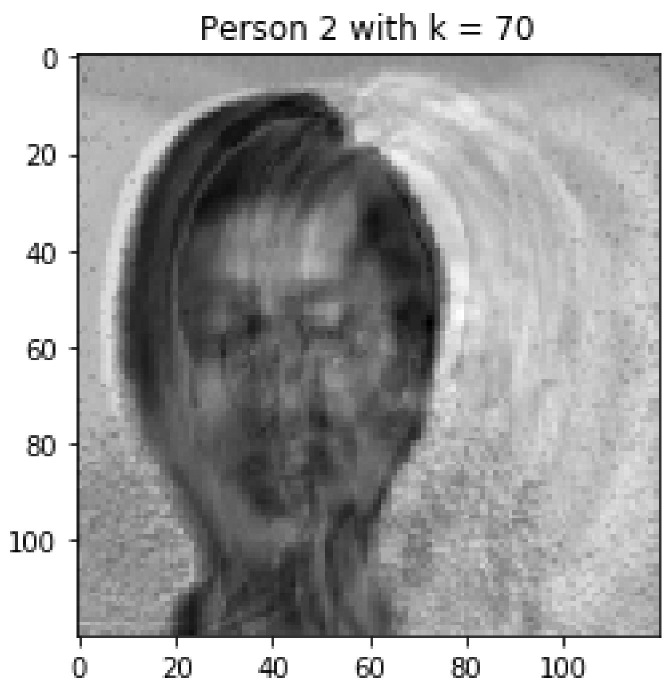

Person 1 with k = 50
0
20
40
60
80
100
0
20
40
60
80
100

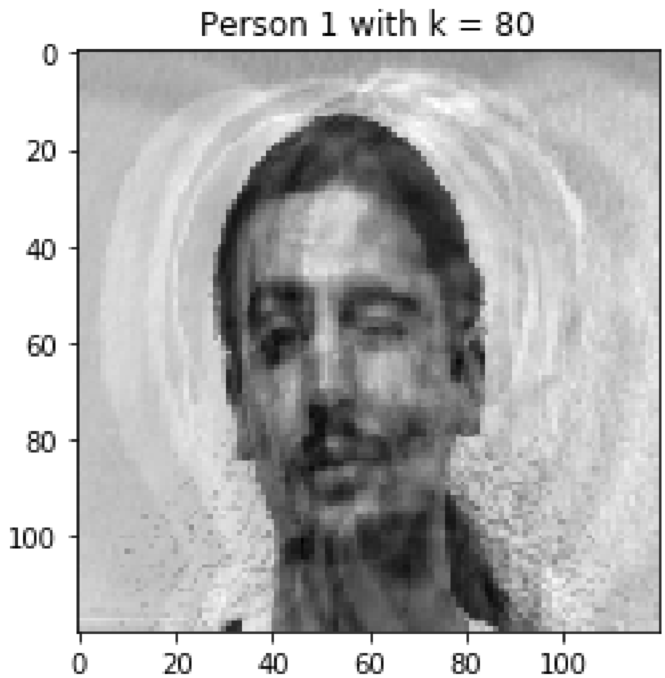

Person 2 with k = 50
0
20
40
60
80
100
0
20
40
60
80
100

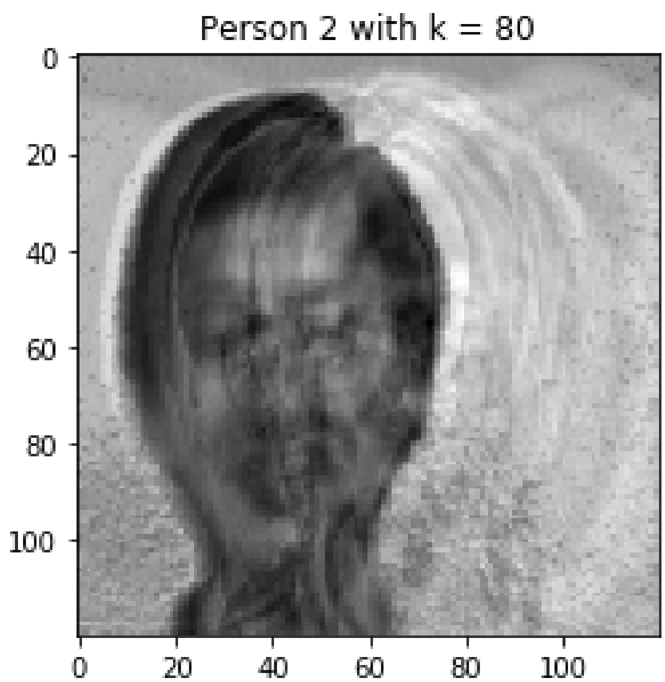

Person 1 with k = 60
0
20
40
60
80
100
0
20
40
60
80
100

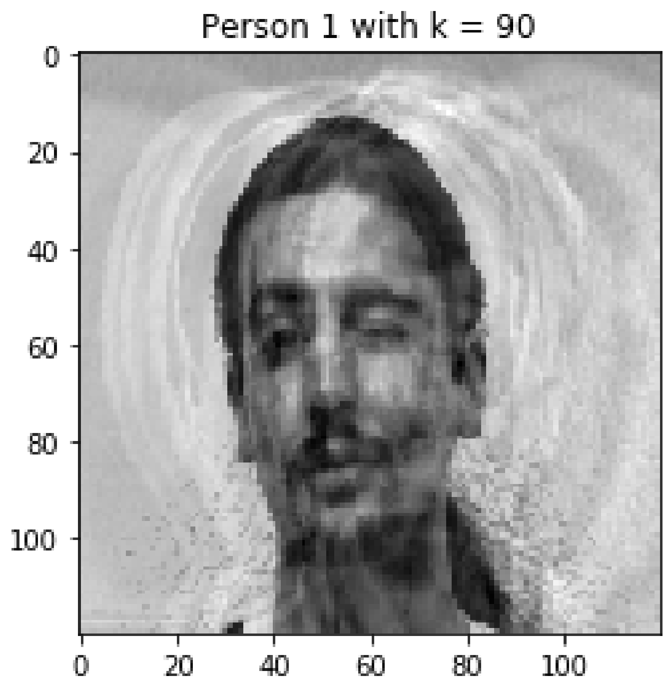

Person 2 with k = 60
0
20
40
60
80
100
0
20
40
60
80
100

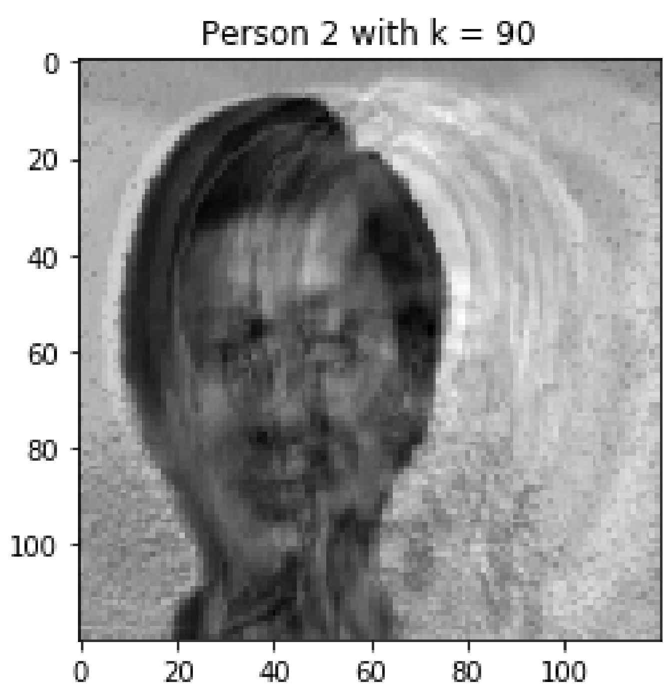

Person 1 with k = 70
0
20
40
60
80
100
0
20
40
60
80
100

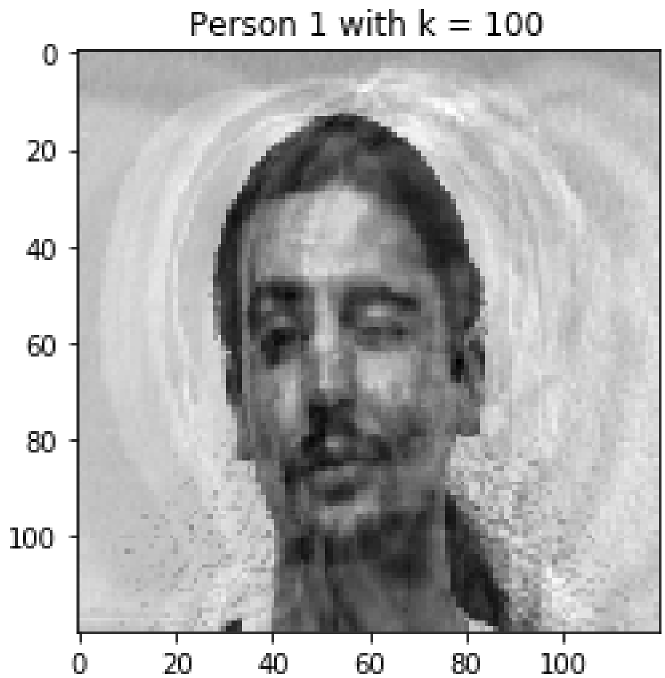

Person 2 with k = 70
0
20
40
60
80
100
0
20
40
60
80
100

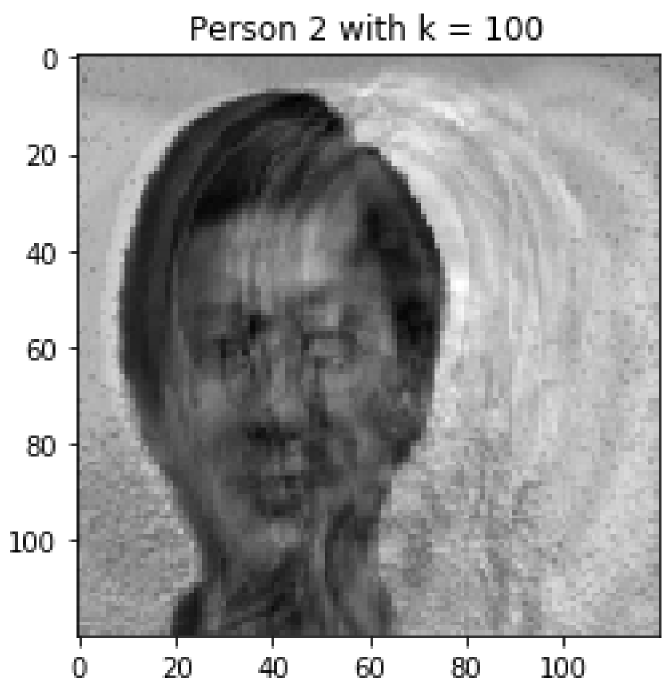

Person 1 with k = 80
0
20
40
60
80
100
0
20
40
60
80
100


Person 2 with k = 80
0
20
40
60
80
100
0
20
40
60
80
100


Person 1 with k = 90
0
20
40
60
80
100
0
20
40
60
80
100


Person 2 with k = 90
0
20
40
60
80
100
0
20
40
60
80
100





In [0]:
```
import re
label_list = []
```

```python
    for file_name in image_files:
        x = re.split("subject", file_name)
        x = re.split("\.", x[1])
        label_list.append(x[0])

    #label_list = list(dict.fromkeys(label_list))
```

In [0]:
```python
y = np.array(label_list, dtype=int)
y = y - 1
```

In [59]:
```python
y.shape
```

Out[59]: (165,)

In [0]:
```python
X = X_orig.copy()
```

In [73]:
```python
from sklearn.model_selection import train_test_split
print('X shape is ', X.shape)
print('y shape is ', y.shape)
print(X[0])
#https://stackoverflow.com/questions/31521170/scikit-learn-train-test-split-with-indices
index_col = np.arange(X.shape[0])
X = np.insert(X, 0, index_col, axis=1)
print(X[0])
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y,
random_state=87)

print('X_train shape is ', X_train.shape)
print('y_train shape is ', y_train.shape)
print('X_test shape is ', X_test.shape)
print('y_test shape is ', y_test.shape)
print(X_train[0])
print(X_test[0])
print(y_train[0])
print(y_test[0])
```

```
X shape is  (165, 14400)
y shape is  (165,)
[0.49428811 0.87158227 0.92398424 ... 0.78773913 0.79821952 0.35804299]
[0.         0.49428811 0.87158227 ... 0.78773913 0.79821952 0.35804299]
X_train shape is  (132, 14401)
y_train shape is  (132,)
X_test shape is  (33, 14401)
y_test shape is  (33,)
[146.           0.51143885   0.8886216  ...   0.7808551    0.77007845
   0.31745915]
[39.          1.18635852  1.18635852 ...  0.72375777  0.85592941
  0.4971778 ]
5
6
```

In [74]:
```python
X_train_samples = X_train[:, 0]
X_test_samples = X_test[:, 0]
print(X_train_samples.shape)
print(X_test_samples.shape)
print(X_train_samples)
print(X_test_samples)
np.save('/content/drive/My Drive/X_train_samples.npy', X_train_samples)
np.save('/content/drive/My Drive/X_train_samples.npy', X_test_samples)

X = np.delete(X, 0, axis=1)
X_train = np.delete(X_train, 0, axis=1)
```

```
X_test = np.delete(X_test, 0, axis=1)
print('X shape is ', X.shape)
print('y shape is ', y.shape)
print('X_train shape is ', X_train.shape)
print('y_train shape is ', y_train.shape)
print('X_test shape is ', X_test.shape)
print('y_test shape is ', y_test.shape)
print(X_train[0])
print(X_test[0])
print(y_train[0])
print(y_test[0])
```

```
(132,)
(33,)
[146.  79.  24. 143. 153. 155. 157. 140.  98.  36.   8.  90.  28.   9.
  34.  69. 136. 101. 137.  40.  15.  91.  80.  70.  17. 149. 152.  29.
  46. 162.  94.   4. 119.  88. 128. 154.  37. 130. 160.   2.  78.  27.
  92.  12.  74. 110.  56.  82.  64. 116. 112. 159.  72.  53.  21.  14.
 118.  57.  77. 129.  96. 105.  87.  30. 134. 114.  83. 138.  52. 131.
 117.   6.  47.  38. 123.  19.  48.  89.  23. 121.  33. 145.  65. 107.
  60.  43.  41.  49.  35.  13.  32.  81.  86. 147.  93.  18. 124. 135.
   5. 126.  54. 103. 158. 164. 142.   1.  20. 139.  97.  75.  26.  22.
  25.  16. 156.  76. 144.  84. 125.  71. 163. 115. 120.  51.  58.  31.
  45. 100. 104.  67.   7.   3.]
[ 39. 150.  59. 108.  85.  62. 111. 161.  99. 113.  73. 106.  63.  68.
  11.  66. 151. 102. 141.  50.   0.  95.  55. 148. 109. 127.  10.  42.
  61. 133.  44. 132. 122.]
X shape is  (165, 14400)
y shape is  (165,)
X_train shape is  (132, 14400)
y_train shape is  (132,)
X_test shape is  (33, 14400)
y_test shape is  (33,)
[0.51143885 0.8886216  0.92095155 ... 0.7808551  0.77007845 0.31745915]
[1.18635852 1.18635852 1.18635852 ... 0.72375777 0.85592941 0.4971778 ]
5
6
```

In [75]:
```
print(X[0])
```

```
[0.49428811 0.87158227 0.92398424 ... 0.78773913 0.79821952 0.35804299]
```

In [76]:
```
np.unique(y_train)
```

Out[76]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14])

In [77]:
```
np.unique(y_test)
```

Out[77]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14])

In [78]:
```
"""
# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc_x = StandardScaler()
X_train = sc_x.fit_transform(X_train)
X_test = sc_x.transform(X_test)
"""
X_train[0]
```

Out[78]: array([0.51143885, 0.8886216 , 0.92095155, ..., 0.7808551 , 0.77007845,
                0.31745915])

In [79]:
```
from sklearn.metrics import accuracy_score
from sklearn.model_selection import StratifiedKFold
from sklearn import svm
from sklearn.preprocessing import StandardScaler

components = [1, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

accuracy = []


skf = StratifiedKFold(n_splits=5, random_state=12)

for k in components:
    all_scores = []
    for train_index, test_index in skf.split(X_train, y_train):
        train_data, test_data = X_train[train_index], X_train[test_index]
        train_target, test_target = y_train[train_index], y_train[test_index]
        U = top_100[:,0:k]
        train_data = np.matmul(train_data, U)
        test_data = np.matmul(test_data, U)

        # Standardization
        sc_x = StandardScaler()
        train_data = sc_x.fit_transform(train_data)
        test_data = sc_x.transform(test_data)

        svm_classifier = svm.SVC(kernel='linear', C=1.0, gamma='scale')
        svm_classifier.fit(train_data, train_target)
        y_pred = svm_classifier.predict(test_data)
        score = accuracy_score(test_target, y_pred) * 100
        all_scores.append(score)
    mean = np.mean(all_scores)
    accuracy.append(mean)
    print('For k = %d the CV accuracy is %f'%(k, mean))
```

```
For k = 1 the CV accuracy is 20.370370
For k = 10 the CV accuracy is 83.407407
For k = 20 the CV accuracy is 88.444444
For k = 30 the CV accuracy is 90.444444
For k = 40 the CV accuracy is 93.777778
For k = 50 the CV accuracy is 91.703704
For k = 60 the CV accuracy is 90.370370
For k = 70 the CV accuracy is 87.703704
For k = 80 the CV accuracy is 81.481481
For k = 90 the CV accuracy is 76.814815
For k = 100 the CV accuracy is 68.592593
```

In [80]:
```
import operator
index, value = max(enumerate(accuracy), key=operator.itemgetter(1))
print(value)
print(components[index])
print('Best K is {} | Accuracy is {} '.format(components[index], value))
```

```
93.77777777777777
40
Best K is 40 | Accuracy is 93.77777777777777
```

In [0]:
```
X_train_copy = X_train.copy()
X_test_copy = X_test.copy()
```

```
In [0]: U = top_100[:,0:40]
        X_train = np.matmul(X_train, U)
        X_test = np.matmul(X_test, U)

        # Standardization
        sc_x = StandardScaler()
        X_train = sc_x.fit_transform(X_train)
        X_test = sc_x.transform(X_test)

        svm_classifier = svm.SVC(kernel='linear', C=1.0, gamma='scale')
        svm_classifier.fit(X_train, y_train)
        y_pred = svm_classifier.predict(X_test)
        score = accuracy_score(y_test, y_pred) * 100
```

```
In [83]: print('Test accuracy is ', score)
```

```
Test accuracy is  96.96969696969697
```

```
In [0]: # Restore the values again

        X_train = X_train_copy.copy()
        X_test = X_test_copy.copy()
```

```
In [85]: # Using CNN to classify
         print(X_train[0])
         print(X_test[0])
         print(y_train[0])
         print(y_test[0])
```

```
[0.51143885 0.8886216  0.92095155 ... 0.7808551  0.77007845 0.31745915]
[1.18635852 1.18635852 1.18635852 ... 0.72375777 0.85592941 0.4971778 ]
5
6
```

```
In [86]: """
         image_files
         def load_image(infilename) :
             img = Image.open( infilename )
             img = img.resize((120,120))
             data = np.array(img, dtype ='float')
             data = data.flatten()
             return data
         """
         from skimage.io import imread

         print('Saved Train sample size ', X_train_samples.shape)
         print('Saved Test sample size ',X_test_samples.shape)

         X_train_CNN = np.empty((X_train_samples.shape[0], 243, 320))
         i = 0
         for train_file in X_train_samples:
             #img = Image.open(image_files[int(train_file)])
             X_train_CNN[i, :, :] = (np.array(imread(image_files[int(train_file)]))) * 1.0/255
             i = i + 1

         print('CNN training input shape ', X_train_CNN.shape)
         print('CNN training output shape ', y_train.shape)

         X_test_CNN = np.empty((X_test_samples.shape[0], 243, 320))
         i = 0
         for test_file in X_test_samples:
             #img = Image.open(image_files[int(train_file)])
             X_test_CNN[i, :, :] = (np.array(imread(image_files[int(test_file)]))) * 1.0/255
             i = i + 1
```

```
print('CNN test input shape ', X_test_CNN.shape)
print('CNN test output shape ', y_test.shape)
```

```
Saved Train sample size  (132,)
Saved Test sample size  (33,)
CNN training input shape  (132, 243, 320)
CNN training output shape  (132,)
CNN test input shape  (33, 243, 320)
CNN test output shape  (33,)
```

In [0]:
```
#y_train = y_train_orig.copy()
#y_test = y_test_orig.copy()
```

In [0]:
```
y_train_orig = y_train.copy()
y_test_orig = y_test.copy()
```

In [89]:
```
from keras.utils import to_categorical

y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
print('CNN training output shape ', y_train.shape)
print('CNN test output shape ', y_test.shape)
```

```
CNN training output shape  (132, 15)
CNN test output shape  (33, 15)
```

In [0]:
```
X_train_CNN = X_train_CNN.reshape(X_train_CNN.shape[0], 243, 320, 1)
X_test_CNN = X_test_CNN.reshape(X_test_CNN.shape[0], 243, 320, 1)
```

In [91]:
```
# Defining the Neural Network Architecture
from keras import layers
from keras import models
from keras import optimizers

model = models.Sequential()
model.add(layers.Conv2D(256, (3, 3), activation='relu',
                        input_shape=(243, 320, 1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Dropout(rate = 0.1))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Dropout(rate = 0.2))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Dropout(rate = 0.2))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(32, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dropout(rate = 0.1))
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dropout(rate = 0.2))
model.add(layers.Dense(256, activation='relu'))
model.add(layers.Dense(15, activation='softmax'))

model.summary()

# Compile the model, configure the optimizer
model.compile(optimizer=optimizers.RMSprop(lr=1e-4),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

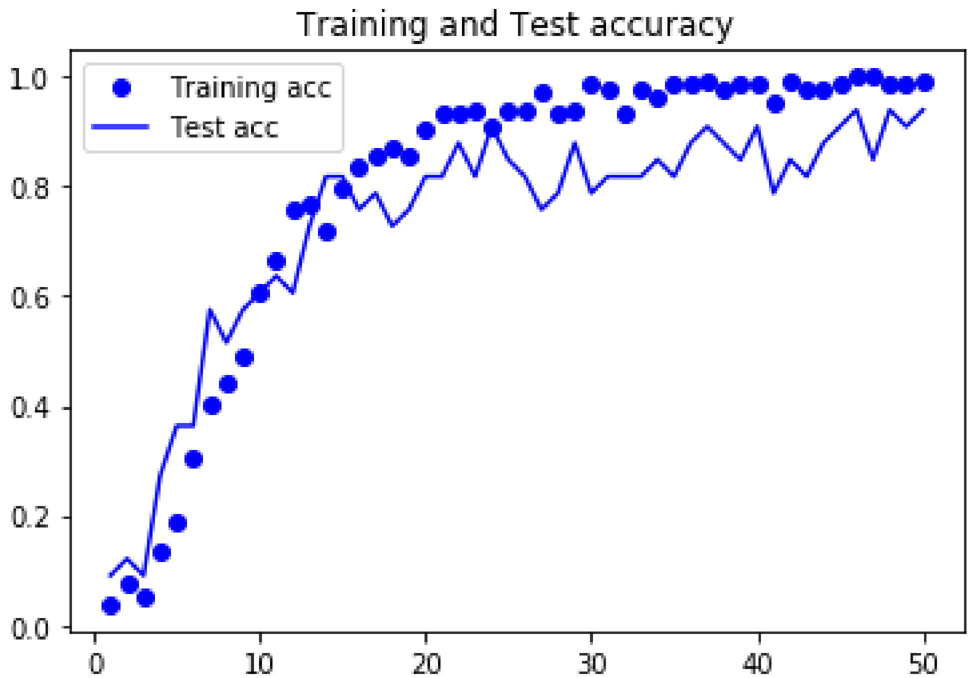
```python
# Training Phase. Record the accuracy and error/loss for tuning later on
history = model.fit(X_train_CNN, y_train, epochs=50, batch_size=8,
validation_data=(X_test_CNN, y_test))
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(acc) + 1)

plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Test acc')
plt.title('Training and Test accuracy')
plt.legend()

plt.figure()

plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Test loss')
plt.title('Training and Test loss')
plt.legend()

plt.show()
```

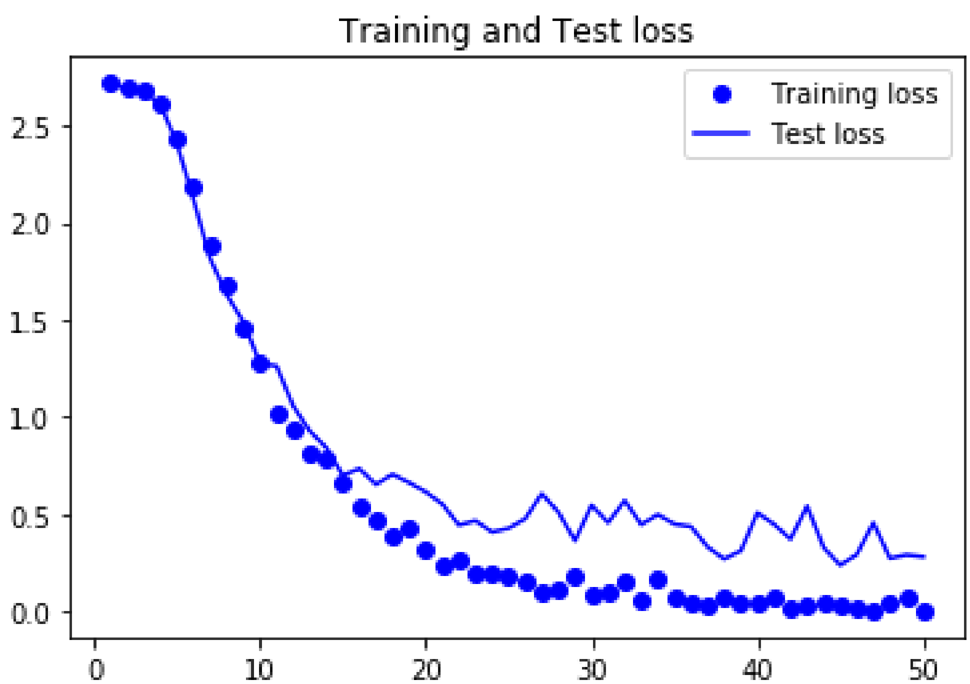
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_22 (Conv2D)           (None, 241, 318, 256)     2560
_________________________________________________________________
max_pooling2d_22 (MaxPooling (None, 120, 159, 256)     0
_________________________________________________________________
dropout_19 (Dropout)         (None, 120, 159, 256)     0
_________________________________________________________________
conv2d_23 (Conv2D)           (None, 118, 157, 128)     295040
_________________________________________________________________
max_pooling2d_23 (MaxPooling (None, 59, 78, 128)       0
_________________________________________________________________
dropout_20 (Dropout)         (None, 59, 78, 128)       0
_________________________________________________________________
conv2d_24 (Conv2D)           (None, 57, 76, 64)        73792
_________________________________________________________________
max_pooling2d_24 (MaxPooling (None, 28, 38, 64)        0
_________________________________________________________________
dropout_21 (Dropout)         (None, 28, 38, 64)        0
_________________________________________________________________
conv2d_25 (Conv2D)           (None, 26, 36, 64)        36928
_________________________________________________________________
max_pooling2d_25 (MaxPooling (None, 13, 18, 64)        0
_________________________________________________________________
conv2d_26 (Conv2D)           (None, 11, 16, 32)        18464
_________________________________________________________________
max_pooling2d_26 (MaxPooling (None, 5, 8, 32)          0
_________________________________________________________________
flatten_6 (Flatten)          (None, 1280)              0
_________________________________________________________________
dropout_22 (Dropout)         (None, 1280)              0
_________________________________________________________________
dense_16 (Dense)             (None, 512)               655872
_________________________________________________________________
dropout_23 (Dropout)         (None, 512)               0
_________________________________________________________________
```

```
dense_17 (Dense)             (None, 256)              131328
_________________________________________________________________
dense_18 (Dense)             (None, 15)               3855
=================================================================
Total params: 1,217,839
Trainable params: 1,217,839
Non-trainable params: 0
_________________________________________________________________
Train on 132 samples, validate on 33 samples
Epoch 1/50
132/132 [==============================] - 6s 44ms/step - loss: 2.7197 - acc: 0.0379 -
val_loss: 2.7048 - val_acc: 0.0909
Epoch 2/50
132/132 [==============================] - 5s 36ms/step - loss: 2.7000 - acc: 0.0758 -
val_loss: 2.6979 - val_acc: 0.1212
Epoch 3/50
132/132 [==============================] - 5s 36ms/step - loss: 2.6759 - acc: 0.0530 -
val_loss: 2.6722 - val_acc: 0.0909
Epoch 4/50
132/132 [==============================] - 5s 36ms/step - loss: 2.6146 - acc: 0.1364 -
val_loss: 2.6103 - val_acc: 0.2727
Epoch 5/50
132/132 [==============================] - 5s 36ms/step - loss: 2.4317 - acc: 0.1894 -
val_loss: 2.4074 - val_acc: 0.3636
Epoch 6/50
132/132 [==============================] - 5s 36ms/step - loss: 2.1825 - acc: 0.3030 -
val_loss: 2.1190 - val_acc: 0.3636
Epoch 7/50
132/132 [==============================] - 5s 36ms/step - loss: 1.8828 - acc: 0.4015 -
val_loss: 1.8166 - val_acc: 0.5758
Epoch 8/50
132/132 [==============================] - 5s 36ms/step - loss: 1.6804 - acc: 0.4394 -
val_loss: 1.6300 - val_acc: 0.5152
Epoch 9/50
132/132 [==============================] - 5s 36ms/step - loss: 1.4653 - acc: 0.4924 -
val_loss: 1.4919 - val_acc: 0.5758
Epoch 10/50
132/132 [==============================] - 5s 36ms/step - loss: 1.2817 - acc: 0.6061 -
val_loss: 1.2759 - val_acc: 0.6061
Epoch 11/50
132/132 [==============================] - 5s 36ms/step - loss: 1.0217 - acc: 0.6667 -
val_loss: 1.2643 - val_acc: 0.6364
Epoch 12/50
132/132 [==============================] - 5s 36ms/step - loss: 0.9407 - acc: 0.7576 -
val_loss: 1.0589 - val_acc: 0.6061
Epoch 13/50
132/132 [==============================] - 5s 36ms/step - loss: 0.8135 - acc: 0.7652 -
val_loss: 0.9294 - val_acc: 0.7273
Epoch 14/50
132/132 [==============================] - 5s 36ms/step - loss: 0.7816 - acc: 0.7197 -
val_loss: 0.8437 - val_acc: 0.8182
Epoch 15/50
132/132 [==============================] - 5s 36ms/step - loss: 0.6572 - acc: 0.7955 -
val_loss: 0.7012 - val_acc: 0.8182
Epoch 16/50
132/132 [==============================] - 5s 36ms/step - loss: 0.5393 - acc: 0.8333 -
val_loss: 0.7354 - val_acc: 0.7576
Epoch 17/50
132/132 [==============================] - 5s 36ms/step - loss: 0.4771 - acc: 0.8561 -
```

```
val_loss: 0.6539 - val_acc: 0.7879
Epoch 18/50
132/132 [==============================] - 5s 36ms/step - loss: 0.3853 - acc: 0.8712 -
val_loss: 0.7065 - val_acc: 0.7273
Epoch 19/50
132/132 [==============================] - 5s 36ms/step - loss: 0.4339 - acc: 0.8561 -
val_loss: 0.6633 - val_acc: 0.7576
Epoch 20/50
132/132 [==============================] - 5s 36ms/step - loss: 0.3240 - acc: 0.9015 -
val_loss: 0.6156 - val_acc: 0.8182
Epoch 21/50
132/132 [==============================] - 5s 36ms/step - loss: 0.2403 - acc: 0.9318 -
val_loss: 0.5524 - val_acc: 0.8182
Epoch 22/50
132/132 [==============================] - 5s 36ms/step - loss: 0.2585 - acc: 0.9318 -
val_loss: 0.4449 - val_acc: 0.8788
Epoch 23/50
132/132 [==============================] - 5s 36ms/step - loss: 0.1949 - acc: 0.9394 -
val_loss: 0.4673 - val_acc: 0.8182
Epoch 24/50
132/132 [==============================] - 5s 36ms/step - loss: 0.1949 - acc: 0.9091 -
val_loss: 0.4088 - val_acc: 0.9091
Epoch 25/50
132/132 [==============================] - 5s 36ms/step - loss: 0.1786 - acc: 0.9394 -
val_loss: 0.4282 - val_acc: 0.8485
Epoch 26/50
132/132 [==============================] - 5s 36ms/step - loss: 0.1526 - acc: 0.9394 -
val_loss: 0.4773 - val_acc: 0.8182
Epoch 27/50
132/132 [==============================] - 5s 36ms/step - loss: 0.1042 - acc: 0.9697 -
val_loss: 0.6065 - val_acc: 0.7576
Epoch 28/50
132/132 [==============================] - 5s 36ms/step - loss: 0.1196 - acc: 0.9318 -
val_loss: 0.5117 - val_acc: 0.7879
Epoch 29/50
132/132 [==============================] - 5s 36ms/step - loss: 0.1830 - acc: 0.9394 -
val_loss: 0.3651 - val_acc: 0.8788
Epoch 30/50
132/132 [==============================] - 5s 36ms/step - loss: 0.0818 - acc: 0.9848 -
val_loss: 0.5471 - val_acc: 0.7879
Epoch 31/50
132/132 [==============================] - 5s 36ms/step - loss: 0.0941 - acc: 0.9773 -
val_loss: 0.4576 - val_acc: 0.8182
Epoch 32/50
132/132 [==============================] - 5s 36ms/step - loss: 0.1557 - acc: 0.9318 -
val_loss: 0.5707 - val_acc: 0.8182
Epoch 33/50
132/132 [==============================] - 5s 36ms/step - loss: 0.0544 - acc: 0.9773 -
val_loss: 0.4474 - val_acc: 0.8182
Epoch 34/50
132/132 [==============================] - 5s 36ms/step - loss: 0.1649 - acc: 0.9621 -
val_loss: 0.4990 - val_acc: 0.8485
Epoch 35/50
132/132 [==============================] - 5s 36ms/step - loss: 0.0741 - acc: 0.9848 -
val_loss: 0.4498 - val_acc: 0.8182
Epoch 36/50
132/132 [==============================] - 5s 36ms/step - loss: 0.0475 - acc: 0.9848 -
val_loss: 0.4372 - val_acc: 0.8788
Epoch 37/50
```

```
132/132 [==============================] - 5s 36ms/step - loss: 0.0381 - acc: 0.9924 -
val_loss: 0.3324 - val_acc: 0.9091
Epoch 38/50
132/132 [==============================] - 5s 36ms/step - loss: 0.0777 - acc: 0.9773 -
val_loss: 0.2699 - val_acc: 0.8788
Epoch 39/50
132/132 [==============================] - 5s 36ms/step - loss: 0.0420 - acc: 0.9848 -
val_loss: 0.3122 - val_acc: 0.8485
Epoch 40/50
132/132 [==============================] - 5s 36ms/step - loss: 0.0461 - acc: 0.9848 -
val_loss: 0.5083 - val_acc: 0.9091
Epoch 41/50
132/132 [==============================] - 5s 36ms/step - loss: 0.0750 - acc: 0.9545 -
val_loss: 0.4490 - val_acc: 0.7879
Epoch 42/50
132/132 [==============================] - 5s 36ms/step - loss: 0.0114 - acc: 0.9924 -
val_loss: 0.3726 - val_acc: 0.8485
Epoch 43/50
132/132 [==============================] - 5s 36ms/step - loss: 0.0360 - acc: 0.9773 -
val_loss: 0.5422 - val_acc: 0.8182
Epoch 44/50
132/132 [==============================] - 5s 36ms/step - loss: 0.0502 - acc: 0.9773 -
val_loss: 0.3287 - val_acc: 0.8788
Epoch 45/50
132/132 [==============================] - 5s 36ms/step - loss: 0.0301 - acc: 0.9848 -
val_loss: 0.2384 - val_acc: 0.9091
Epoch 46/50
132/132 [==============================] - 5s 36ms/step - loss: 0.0147 - acc: 1.0000 -
val_loss: 0.2933 - val_acc: 0.9394
Epoch 47/50
132/132 [==============================] - 5s 36ms/step - loss: 0.0050 - acc: 1.0000 -
val_loss: 0.4567 - val_acc: 0.8485
Epoch 48/50
132/132 [==============================] - 5s 36ms/step - loss: 0.0437 - acc: 0.9848 -
val_loss: 0.2734 - val_acc: 0.9394
Epoch 49/50
132/132 [==============================] - 5s 36ms/step - loss: 0.0707 - acc: 0.9848 -
val_loss: 0.2916 - val_acc: 0.9091
Epoch 50/50
132/132 [==============================] - 5s 36ms/step - loss: 0.0106 - acc: 0.9924 -
val_loss: 0.2815 - val_acc: 0.9394
```






```
In [92]: from keras import regularizers
```

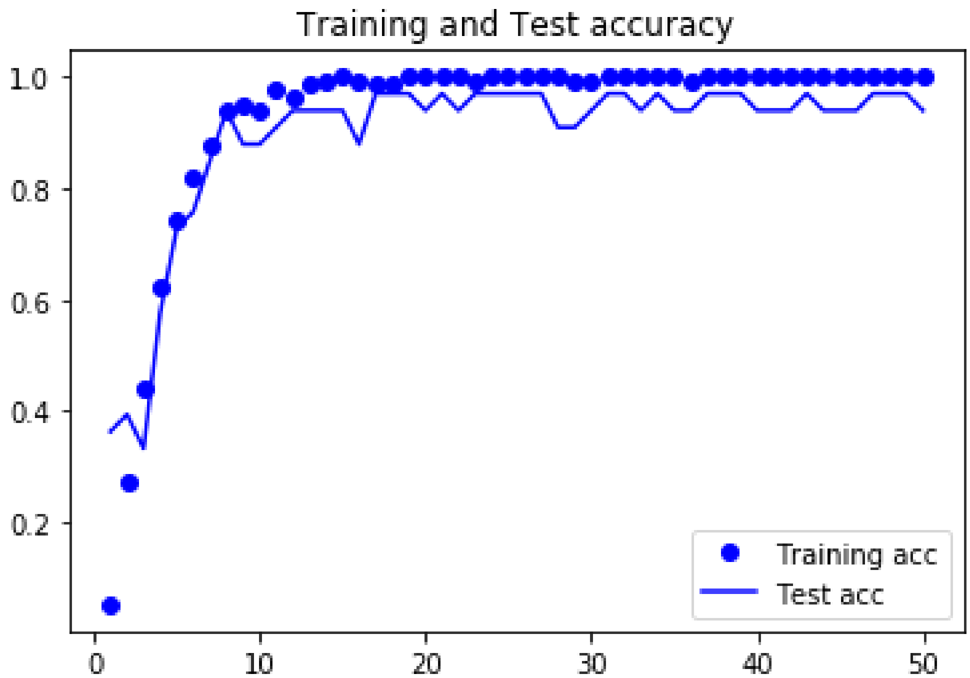
```python
model = models.Sequential()
model.add(layers.Conv2D(64, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=(243, 320, 1)))
model.add(layers.MaxPooling2D(pool_size=(2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D(pool_size=(2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D(pool_size=(2, 2)))
model.add(layers.Dropout(rate = 0.25))
model.add(layers.Conv2D(256, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D(pool_size=(2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dropout(rate = 0.2))
model.add(layers.Dense(256, activation='relu'))
model.add(layers.Dense(15, activation='softmax'))

# Compile the model, configure the optimizer
model.compile(optimizer=optimizers.RMSprop(lr=1e-4),
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.summary()
# Training Phase. Record the accuracy and error/loss for tuning later on

history = model.fit(X_train_CNN, y_train, epochs=50, batch_size=8,
validation_data=(X_test_CNN, y_test))
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(acc) + 1)

plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Test acc')
plt.title('Training and Test accuracy')
plt.legend()

plt.figure()

plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Test loss')
plt.title('Training and Test loss')
plt.legend()

plt.show()
```

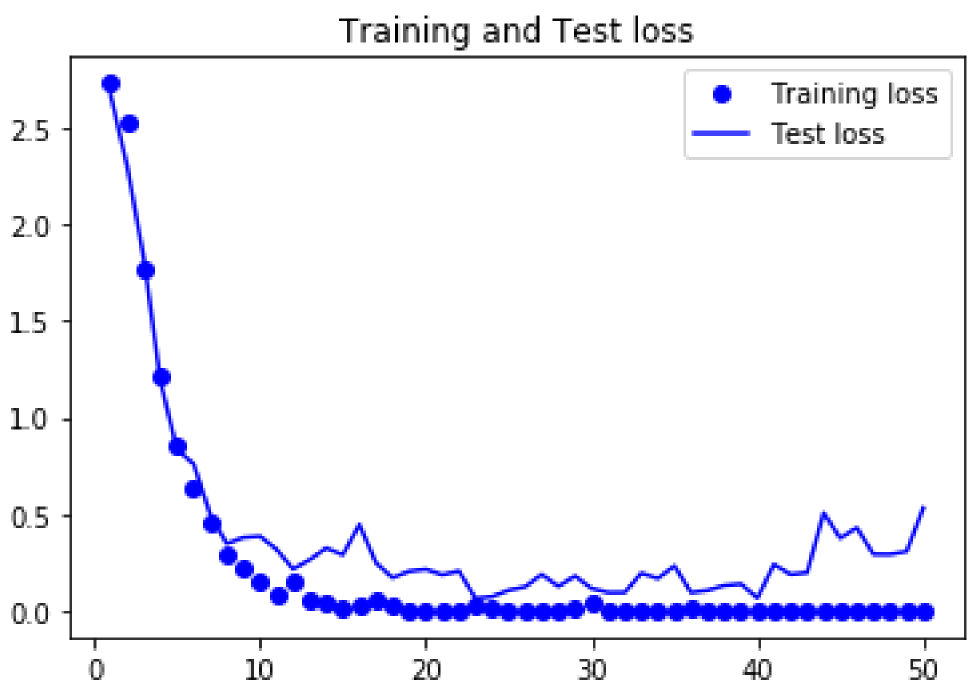
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_27 (Conv2D)           (None, 241, 318, 64)      640
_________________________________________________________________
max_pooling2d_27 (MaxPooling (None, 120, 159, 64)      0
_________________________________________________________________
conv2d_28 (Conv2D)           (None, 118, 157, 128)     73856
_________________________________________________________________
max_pooling2d_28 (MaxPooling (None, 59, 78, 128)       0
_________________________________________________________________
conv2d_29 (Conv2D)           (None, 57, 76, 128)       147584
_________________________________________________________________
max_pooling2d_29 (MaxPooling (None, 28, 38, 128)       0
_________________________________________________________________
dropout_24 (Dropout)         (None, 28, 38, 128)       0
```

```
_________________________________________________________________
conv2d_30 (Conv2D)           (None, 26, 36, 256)       295168
_________________________________________________________________
max_pooling2d_30 (MaxPooling (None, 13, 18, 256)       0
_________________________________________________________________
flatten_7 (Flatten)          (None, 59904)             0
_________________________________________________________________
dense_19 (Dense)             (None, 512)               30671360
_________________________________________________________________
dropout_25 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_20 (Dense)             (None, 256)               131328
_________________________________________________________________
dense_21 (Dense)             (None, 15)                3855
=================================================================
Total params: 31,323,791
Trainable params: 31,323,791
Non-trainable params: 0
_________________________________________________________________
Train on 132 samples, validate on 33 samples
Epoch 1/50
132/132 [==============================] - 5s 41ms/step - loss: 2.7323 - acc: 0.0530 -
val_loss: 2.6685 - val_acc: 0.3636
Epoch 2/50
132/132 [==============================] - 2s 18ms/step - loss: 2.5225 - acc: 0.2727 -
val_loss: 2.2988 - val_acc: 0.3939
Epoch 3/50
132/132 [==============================] - 2s 18ms/step - loss: 1.7721 - acc: 0.4394 -
val_loss: 1.8217 - val_acc: 0.3333
Epoch 4/50
132/132 [==============================] - 2s 18ms/step - loss: 1.2132 - acc: 0.6212 -
val_loss: 1.1878 - val_acc: 0.5758
Epoch 5/50
132/132 [==============================] - 2s 18ms/step - loss: 0.8533 - acc: 0.7424 -
val_loss: 0.8331 - val_acc: 0.7273
Epoch 6/50
132/132 [==============================] - 2s 18ms/step - loss: 0.6297 - acc: 0.8182 -
val_loss: 0.7599 - val_acc: 0.7576
Epoch 7/50
132/132 [==============================] - 2s 18ms/step - loss: 0.4521 - acc: 0.8788 -
val_loss: 0.4990 - val_acc: 0.8485
Epoch 8/50
132/132 [==============================] - 2s 18ms/step - loss: 0.2892 - acc: 0.9394 -
val_loss: 0.3457 - val_acc: 0.9394
Epoch 9/50
132/132 [==============================] - 2s 18ms/step - loss: 0.2180 - acc: 0.9470 -
val_loss: 0.3781 - val_acc: 0.8788
Epoch 10/50
132/132 [==============================] - 2s 18ms/step - loss: 0.1517 - acc: 0.9394 -
val_loss: 0.3847 - val_acc: 0.8788
Epoch 11/50
132/132 [==============================] - 2s 18ms/step - loss: 0.0802 - acc: 0.9773 -
val_loss: 0.3165 - val_acc: 0.9091
Epoch 12/50
132/132 [==============================] - 2s 18ms/step - loss: 0.1449 - acc: 0.9621 -
val_loss: 0.2146 - val_acc: 0.9394
Epoch 13/50
132/132 [==============================] - 2s 18ms/step - loss: 0.0529 - acc: 0.9848 -
val_loss: 0.2628 - val_acc: 0.9394
```

```
Epoch 14/50
132/132 [==============================] - 2s 18ms/step - loss: 0.0452 - acc: 0.9924 -
val_loss: 0.3237 - val_acc: 0.9394
Epoch 15/50
132/132 [==============================] - 2s 18ms/step - loss: 0.0179 - acc: 1.0000 -
val_loss: 0.2884 - val_acc: 0.9394
Epoch 16/50
132/132 [==============================] - 2s 18ms/step - loss: 0.0243 - acc: 0.9924 -
val_loss: 0.4453 - val_acc: 0.8788
Epoch 17/50
132/132 [==============================] - 2s 18ms/step - loss: 0.0595 - acc: 0.9848 -
val_loss: 0.2471 - val_acc: 0.9697
Epoch 18/50
132/132 [==============================] - 2s 18ms/step - loss: 0.0278 - acc: 0.9848 -
val_loss: 0.1693 - val_acc: 0.9697
Epoch 19/50
132/132 [==============================] - 2s 18ms/step - loss: 0.0016 - acc: 1.0000 -
val_loss: 0.2030 - val_acc: 0.9697
Epoch 20/50
132/132 [==============================] - 2s 18ms/step - loss: 0.0025 - acc: 1.0000 -
val_loss: 0.2135 - val_acc: 0.9394
Epoch 21/50
132/132 [==============================] - 2s 18ms/step - loss: 0.0018 - acc: 1.0000 -
val_loss: 0.1849 - val_acc: 0.9697
Epoch 22/50
132/132 [==============================] - 2s 18ms/step - loss: 2.5222e-04 - acc:
1.0000 - val_loss: 0.2041 - val_acc: 0.9394
Epoch 23/50
132/132 [==============================] - 2s 18ms/step - loss: 0.0234 - acc: 0.9924 -
val_loss: 0.0633 - val_acc: 0.9697
Epoch 24/50
132/132 [==============================] - 2s 18ms/step - loss: 0.0086 - acc: 1.0000 -
val_loss: 0.0725 - val_acc: 0.9697
Epoch 25/50
132/132 [==============================] - 2s 18ms/step - loss: 4.1951e-04 - acc:
1.0000 - val_loss: 0.1056 - val_acc: 0.9697
Epoch 26/50
132/132 [==============================] - 2s 18ms/step - loss: 7.8779e-04 - acc:
1.0000 - val_loss: 0.1233 - val_acc: 0.9697
Epoch 27/50
132/132 [==============================] - 2s 18ms/step - loss: 1.4414e-04 - acc:
1.0000 - val_loss: 0.1872 - val_acc: 0.9697
Epoch 28/50
132/132 [==============================] - 2s 18ms/step - loss: 0.0029 - acc: 1.0000 -
val_loss: 0.1255 - val_acc: 0.9091
Epoch 29/50
132/132 [==============================] - 2s 18ms/step - loss: 0.0101 - acc: 0.9924 -
val_loss: 0.1786 - val_acc: 0.9091
Epoch 30/50
132/132 [==============================] - 2s 18ms/step - loss: 0.0346 - acc: 0.9924 -
val_loss: 0.1147 - val_acc: 0.9394
Epoch 31/50
132/132 [==============================] - 2s 18ms/step - loss: 8.3173e-04 - acc:
1.0000 - val_loss: 0.0942 - val_acc: 0.9697
Epoch 32/50
132/132 [==============================] - 2s 18ms/step - loss: 6.9382e-05 - acc:
1.0000 - val_loss: 0.0945 - val_acc: 0.9697
Epoch 33/50
132/132 [==============================] - 2s 18ms/step - loss: 2.9704e-04 - acc:
```

```
1.0000 - val_loss: 0.1933 - val_acc: 0.9394
Epoch 34/50
132/132 [==============================] - 2s 18ms/step - loss: 5.3343e-04 - acc:
1.0000 - val_loss: 0.1660 - val_acc: 0.9697
Epoch 35/50
132/132 [==============================] - 2s 18ms/step - loss: 1.2926e-05 - acc:
1.0000 - val_loss: 0.2309 - val_acc: 0.9394
Epoch 36/50
132/132 [==============================] - 2s 18ms/step - loss: 0.0076 - acc: 0.9924 -
val_loss: 0.0940 - val_acc: 0.9394
Epoch 37/50
132/132 [==============================] - 2s 18ms/step - loss: 0.0015 - acc: 1.0000 -
val_loss: 0.1049 - val_acc: 0.9697
Epoch 38/50
132/132 [==============================] - 2s 18ms/step - loss: 1.3855e-04 - acc:
1.0000 - val_loss: 0.1300 - val_acc: 0.9697
Epoch 39/50
132/132 [==============================] - 2s 18ms/step - loss: 1.1994e-05 - acc:
1.0000 - val_loss: 0.1385 - val_acc: 0.9697
Epoch 40/50
132/132 [==============================] - 2s 18ms/step - loss: 0.0054 - acc: 1.0000 -
val_loss: 0.0654 - val_acc: 0.9394
Epoch 41/50
132/132 [==============================] - 2s 18ms/step - loss: 1.0165e-04 - acc:
1.0000 - val_loss: 0.2387 - val_acc: 0.9394
Epoch 42/50
132/132 [==============================] - 2s 18ms/step - loss: 1.2960e-04 - acc:
1.0000 - val_loss: 0.1868 - val_acc: 0.9394
Epoch 43/50
132/132 [==============================] - 2s 18ms/step - loss: 7.0367e-05 - acc:
1.0000 - val_loss: 0.1960 - val_acc: 0.9697
Epoch 44/50
132/132 [==============================] - 2s 18ms/step - loss: 8.6424e-05 - acc:
1.0000 - val_loss: 0.5060 - val_acc: 0.9394
Epoch 45/50
132/132 [==============================] - 2s 18ms/step - loss: 2.5447e-06 - acc:
1.0000 - val_loss: 0.3743 - val_acc: 0.9394
Epoch 46/50
132/132 [==============================] - 2s 18ms/step - loss: 4.1142e-06 - acc:
1.0000 - val_loss: 0.4293 - val_acc: 0.9394
Epoch 47/50
132/132 [==============================] - 2s 18ms/step - loss: 3.5587e-06 - acc:
1.0000 - val_loss: 0.2911 - val_acc: 0.9697
Epoch 48/50
132/132 [==============================] - 2s 18ms/step - loss: 2.4203e-07 - acc:
1.0000 - val_loss: 0.2907 - val_acc: 0.9697
Epoch 49/50
132/132 [==============================] - 2s 18ms/step - loss: 2.1313e-07 - acc:
1.0000 - val_loss: 0.3059 - val_acc: 0.9697
Epoch 50/50
132/132 [==============================] - 2s 18ms/step - loss: 1.8989e-06 - acc:
1.0000 - val_loss: 0.5309 - val_acc: 0.9394
```






```
In [93]: model = models.Sequential()
         model.add(layers.Conv2D(32, (3, 3), activation='relu',
```

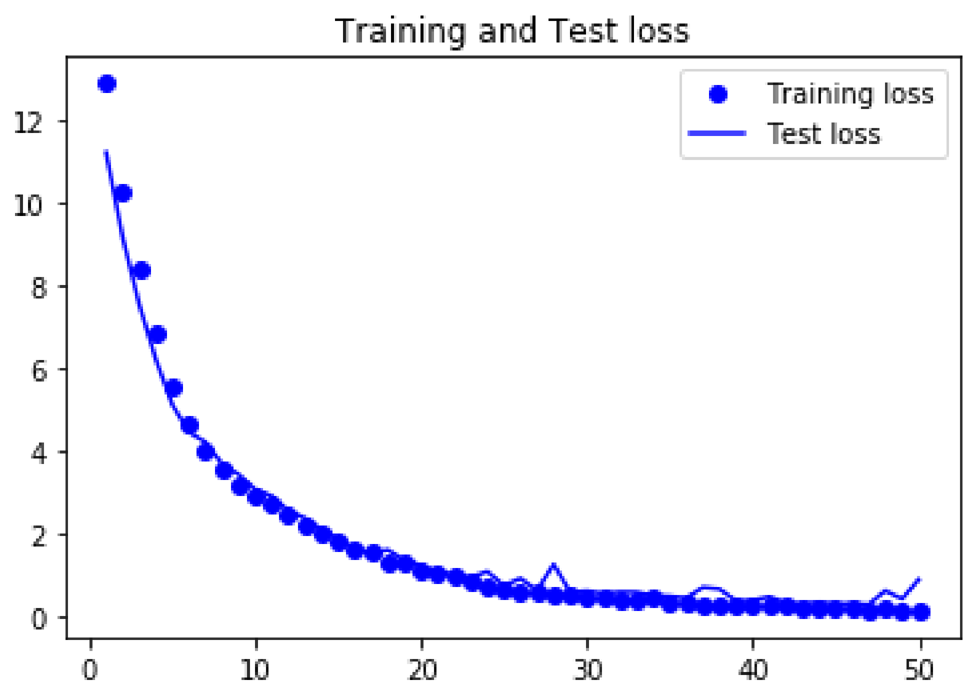
```python
                        input_shape=(243, 320, 1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu',
kernel_regularizer=regularizers.l2(0.01)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Dropout(0.1))
model.add(layers.Conv2D(128, (3, 3), activation='relu',
kernel_regularizer=regularizers.l2(0.01)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dropout(0.1))
model.add(layers.Dense(512, activation='relu',
kernel_regularizer=regularizers.l2(0.01)))
model.add(layers.Dropout(0.2))
model.add(layers.Dense(256, activation='relu'))
model.add(layers.Dense(15, activation='softmax'))

model.summary()
# Compile the model, configure the optimizer
model.compile(optimizer=optimizers.RMSprop(lr=1e-4),
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Training Phase. Record the accuracy and error/loss for tuning later on
history = model.fit(X_train_CNN, y_train, epochs=50, batch_size=8,
validation_data=(X_test_CNN, y_test))

acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(acc) + 1)

plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Test acc')
plt.title('Training and Test accuracy')
plt.legend()

plt.figure()

plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Test loss')
plt.title('Training and Test loss')
plt.legend()

plt.show()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_31 (Conv2D)           (None, 241, 318, 32)      320
_________________________________________________________________
max_pooling2d_31 (MaxPooling (None, 120, 159, 32)      0
_________________________________________________________________
conv2d_32 (Conv2D)           (None, 118, 157, 64)      18496
_________________________________________________________________
max_pooling2d_32 (MaxPooling (None, 59, 78, 64)        0
_________________________________________________________________
conv2d_33 (Conv2D)           (None, 57, 76, 128)       73856
_________________________________________________________________
max_pooling2d_33 (MaxPooling (None, 28, 38, 128)       0
_________________________________________________________________
dropout_26 (Dropout)         (None, 28, 38, 128)       0
```

```
_________________________________________________________________
conv2d_34 (Conv2D)           (None, 26, 36, 128)       147584
_________________________________________________________________
max_pooling2d_34 (MaxPooling (None, 13, 18, 128)       0
_________________________________________________________________
flatten_8 (Flatten)          (None, 29952)             0
_________________________________________________________________
dropout_27 (Dropout)         (None, 29952)             0
_________________________________________________________________
dense_22 (Dense)             (None, 512)               15335936
_________________________________________________________________
dropout_28 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_23 (Dense)             (None, 256)               131328
_________________________________________________________________
dense_24 (Dense)             (None, 15)                3855
=================================================================
Total params: 15,711,375
Trainable params: 15,711,375
Non-trainable params: 0
_________________________________________________________________
Train on 132 samples, validate on 33 samples
Epoch 1/50
132/132 [==============================] - 4s 28ms/step - loss: 12.8673 - acc: 0.0833
- val_loss: 11.1977 - val_acc: 0.2121
Epoch 2/50
132/132 [==============================] - 1s 10ms/step - loss: 10.2162 - acc: 0.1288
- val_loss: 9.1525 - val_acc: 0.1818
Epoch 3/50
132/132 [==============================] - 1s 10ms/step - loss: 8.4001 - acc: 0.2197 -
val_loss: 7.5341 - val_acc: 0.1515
Epoch 4/50
132/132 [==============================] - 1s 10ms/step - loss: 6.8313 - acc: 0.3636 -
val_loss: 6.2165 - val_acc: 0.3333
Epoch 5/50
132/132 [==============================] - 1s 10ms/step - loss: 5.5261 - acc: 0.5758 -
val_loss: 5.1205 - val_acc: 0.6061
Epoch 6/50
132/132 [==============================] - 1s 10ms/step - loss: 4.6530 - acc: 0.6970 -
val_loss: 4.4596 - val_acc: 0.6667
Epoch 7/50
132/132 [==============================] - 1s 10ms/step - loss: 4.0152 - acc: 0.8030 -
val_loss: 4.2233 - val_acc: 0.5758
Epoch 8/50
132/132 [==============================] - 1s 10ms/step - loss: 3.5919 - acc: 0.8561 -
val_loss: 3.6957 - val_acc: 0.7576
Epoch 9/50
132/132 [==============================] - 1s 10ms/step - loss: 3.1960 - acc: 0.9242 -
val_loss: 3.4408 - val_acc: 0.8182
Epoch 10/50
132/132 [==============================] - 1s 10ms/step - loss: 2.9324 - acc: 0.9470 -
val_loss: 3.0832 - val_acc: 0.8485
Epoch 11/50
132/132 [==============================] - 1s 10ms/step - loss: 2.7046 - acc: 0.9621 -
val_loss: 2.9220 - val_acc: 0.8182
Epoch 12/50
132/132 [==============================] - 1s 10ms/step - loss: 2.4531 - acc: 0.9848 -
val_loss: 2.5622 - val_acc: 0.9394
Epoch 13/50
```

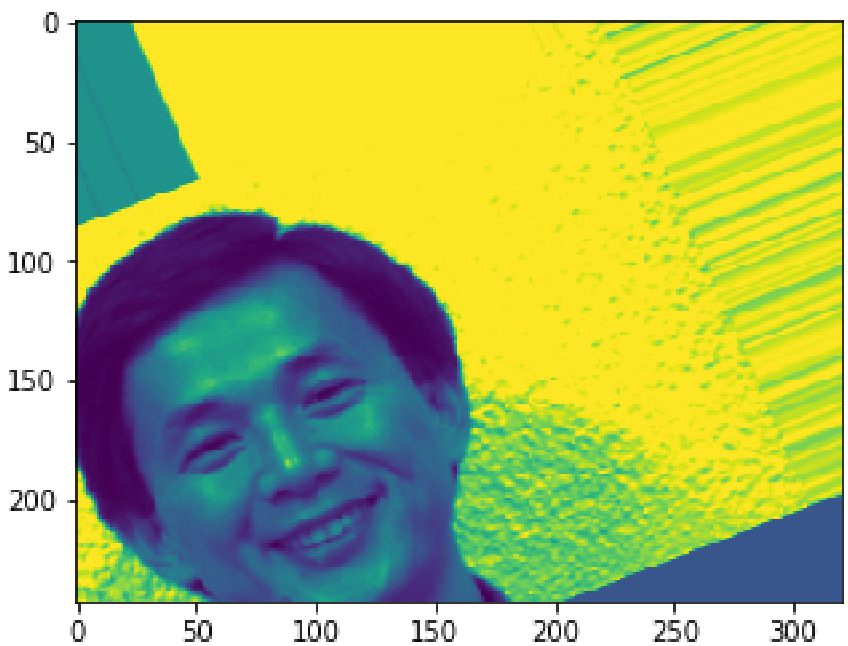
```
132/132 [==============================] - 1s 10ms/step - loss: 2.2330 - acc: 0.9848 -
val_loss: 2.3896 - val_acc: 0.8788
Epoch 14/50
132/132 [==============================] - 1s 10ms/step - loss: 2.0422 - acc: 0.9848 -
val_loss: 2.1232 - val_acc: 0.9394
Epoch 15/50
132/132 [==============================] - 1s 10ms/step - loss: 1.8193 - acc: 1.0000 -
val_loss: 1.9010 - val_acc: 0.9394
Epoch 16/50
132/132 [==============================] - 1s 10ms/step - loss: 1.6395 - acc: 0.9924 -
val_loss: 1.6570 - val_acc: 0.9697
Epoch 17/50
132/132 [==============================] - 1s 10ms/step - loss: 1.5922 - acc: 0.9621 -
val_loss: 1.5399 - val_acc: 0.9697
Epoch 18/50
132/132 [==============================] - 1s 10ms/step - loss: 1.3410 - acc: 1.0000 -
val_loss: 1.6258 - val_acc: 0.9091
Epoch 19/50
132/132 [==============================] - 1s 10ms/step - loss: 1.3082 - acc: 0.9773 -
val_loss: 1.3715 - val_acc: 0.9394
Epoch 20/50
132/132 [==============================] - 1s 10ms/step - loss: 1.1541 - acc: 1.0000 -
val_loss: 1.2507 - val_acc: 0.9697
Epoch 21/50
132/132 [==============================] - 1s 10ms/step - loss: 1.0279 - acc: 1.0000 -
val_loss: 1.1085 - val_acc: 0.9394
Epoch 22/50
132/132 [==============================] - 1s 10ms/step - loss: 0.9824 - acc: 0.9924 -
val_loss: 1.0306 - val_acc: 0.9394
Epoch 23/50
132/132 [==============================] - 1s 10ms/step - loss: 0.8468 - acc: 1.0000 -
val_loss: 0.9688 - val_acc: 0.9697
Epoch 24/50
132/132 [==============================] - 1s 10ms/step - loss: 0.7613 - acc: 1.0000 -
val_loss: 1.1006 - val_acc: 0.9091
Epoch 25/50
132/132 [==============================] - 1s 10ms/step - loss: 0.7012 - acc: 1.0000 -
val_loss: 0.7671 - val_acc: 0.9697
Epoch 26/50
132/132 [==============================] - 1s 10ms/step - loss: 0.6333 - acc: 1.0000 -
val_loss: 0.9361 - val_acc: 0.8788
Epoch 27/50
132/132 [==============================] - 1s 10ms/step - loss: 0.5779 - acc: 1.0000 -
val_loss: 0.6501 - val_acc: 0.9697
Epoch 28/50
132/132 [==============================] - 1s 10ms/step - loss: 0.5471 - acc: 0.9848 -
val_loss: 1.2762 - val_acc: 0.8485
Epoch 29/50
132/132 [==============================] - 1s 10ms/step - loss: 0.5533 - acc: 0.9773 -
val_loss: 0.6074 - val_acc: 0.9697
Epoch 30/50
132/132 [==============================] - 1s 10ms/step - loss: 0.4824 - acc: 0.9924 -
val_loss: 0.6402 - val_acc: 0.9394
Epoch 31/50
132/132 [==============================] - 1s 10ms/step - loss: 0.4499 - acc: 0.9924 -
val_loss: 0.5791 - val_acc: 0.9394
Epoch 32/50
132/132 [==============================] - 1s 10ms/step - loss: 0.4100 - acc: 1.0000 -
val_loss: 0.6132 - val_acc: 0.9394
```

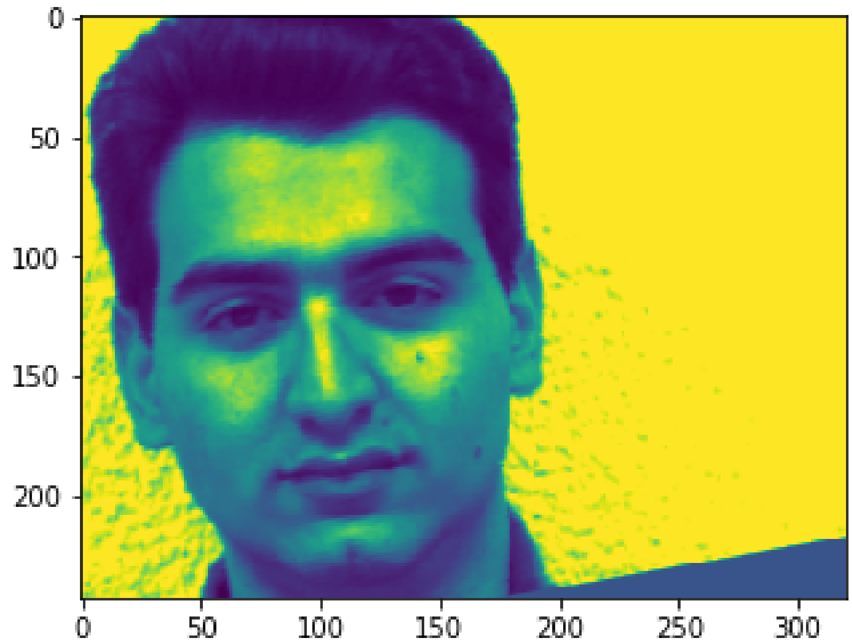
```
Epoch 33/50
132/132 [==============================] - 1s 10ms/step - loss: 0.4169 - acc: 0.9924 -
val_loss: 0.6082 - val_acc: 0.9091
Epoch 34/50
132/132 [==============================] - 1s 10ms/step - loss: 0.4593 - acc: 0.9924 -
val_loss: 0.5242 - val_acc: 0.9394
Epoch 35/50
132/132 [==============================] - 1s 10ms/step - loss: 0.3575 - acc: 1.0000 -
val_loss: 0.5117 - val_acc: 0.9697
Epoch 36/50
132/132 [==============================] - 1s 10ms/step - loss: 0.3312 - acc: 1.0000 -
val_loss: 0.4687 - val_acc: 0.9697
Epoch 37/50
132/132 [==============================] - 1s 10ms/step - loss: 0.3044 - acc: 1.0000 -
val_loss: 0.7256 - val_acc: 0.8182
Epoch 38/50
132/132 [==============================] - 1s 10ms/step - loss: 0.3132 - acc: 0.9848 -
val_loss: 0.6849 - val_acc: 0.8485
Epoch 39/50
132/132 [==============================] - 1s 10ms/step - loss: 0.2845 - acc: 0.9924 -
val_loss: 0.4006 - val_acc: 0.9394
Epoch 40/50
132/132 [==============================] - 1s 10ms/step - loss: 0.2560 - acc: 1.0000 -
val_loss: 0.4146 - val_acc: 0.9394
Epoch 41/50
132/132 [==============================] - 1s 10ms/step - loss: 0.3082 - acc: 0.9848 -
val_loss: 0.4984 - val_acc: 0.8788
Epoch 42/50
132/132 [==============================] - 1s 10ms/step - loss: 0.2699 - acc: 0.9924 -
val_loss: 0.3577 - val_acc: 0.9394
Epoch 43/50
132/132 [==============================] - 1s 10ms/step - loss: 0.2254 - acc: 1.0000 -
val_loss: 0.3288 - val_acc: 0.9697
Epoch 44/50
132/132 [==============================] - 1s 10ms/step - loss: 0.2439 - acc: 0.9848 -
val_loss: 0.3697 - val_acc: 0.9394
Epoch 45/50
132/132 [==============================] - 1s 10ms/step - loss: 0.2122 - acc: 1.0000 -
val_loss: 0.3270 - val_acc: 0.9697
Epoch 46/50
132/132 [==============================] - 1s 10ms/step - loss: 0.2031 - acc: 1.0000 -
val_loss: 0.3381 - val_acc: 0.9091
Epoch 47/50
132/132 [==============================] - 1s 10ms/step - loss: 0.1879 - acc: 1.0000 -
val_loss: 0.3124 - val_acc: 0.9697
Epoch 48/50
132/132 [==============================] - 1s 10ms/step - loss: 0.2032 - acc: 0.9773 -
val_loss: 0.6447 - val_acc: 0.8788
Epoch 49/50
132/132 [==============================] - 1s 10ms/step - loss: 0.1859 - acc: 0.9924 -
val_loss: 0.4472 - val_acc: 0.9091
Epoch 50/50
132/132 [==============================] - 1s 10ms/step - loss: 0.1634 - acc: 1.0000 -
val_loss: 0.9058 - val_acc: 0.7879
```

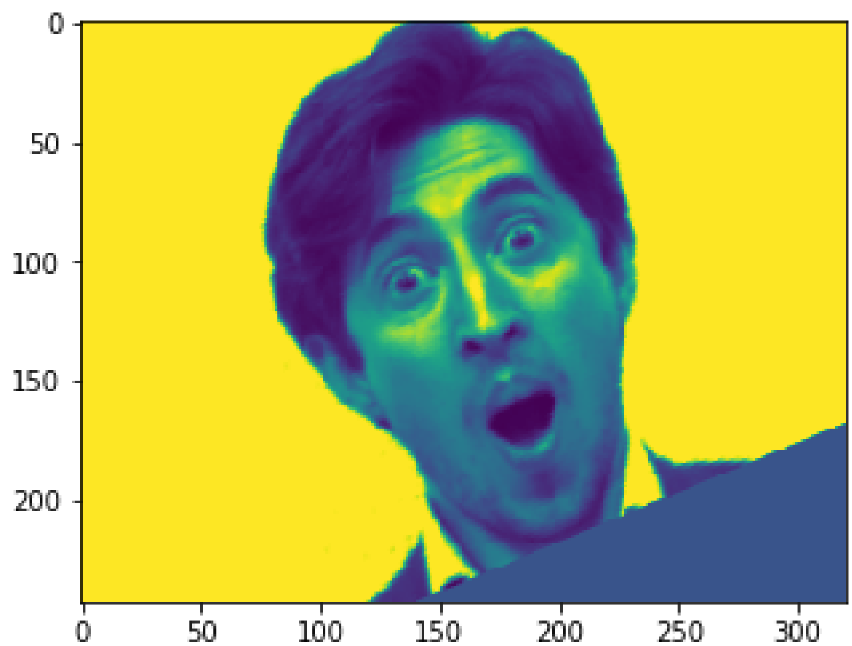


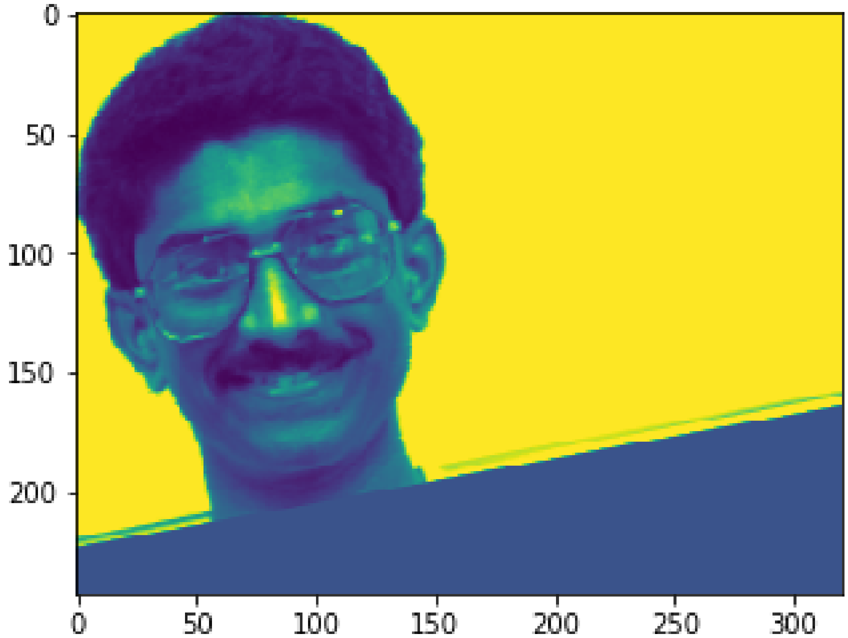

```
In [94]: # Question 2.g
         from keras.preprocessing.image import ImageDataGenerator
```

```python
datagen = ImageDataGenerator(
    rotation_range=30,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)

# compute quantities required for featurewise normalization
# (std, mean, and principal components if ZCA whitening is applied)
datagen.fit(X_train_CNN)

# Displaying some randomly augmented training images
from keras.preprocessing import image
i = 0
for batch in datagen.flow(X_train_CNN, batch_size=1):
    plt.figure(i)
    imgplot = plt.imshow(image.array_to_img(batch[0]))
    i += 1
    if i % 4 == 0:
        break

plt.show()
```




0
50
100
150
200
0
50
100
150
200
250
300


0
50
100
150
200
0
50
100
150
200
250
300



```
In [95]: model = models.Sequential()
         model.add(layers.Conv2D(32, (3, 3), activation='relu',
                                 input_shape=(243, 320, 1)))
         model.add(layers.MaxPooling2D((2, 2)))
         model.add(layers.Conv2D(64, (3, 3), activation='relu'))
         model.add(layers.MaxPooling2D((2, 2)))
         model.add(layers.Conv2D(128, (3, 3), activation='relu'))
         model.add(layers.MaxPooling2D((2, 2)))
         model.add(layers.Dropout(0.1))
         model.add(layers.Conv2D(128, (3, 3), activation='relu'))
         model.add(layers.MaxPooling2D((2, 2)))
         model.add(layers.Flatten())
         model.add(layers.Dropout(0.1))
         model.add(layers.Dense(512, activation='relu',
         kernel_regularizer=regularizers.l2(0.01)))
         model.add(layers.Dropout(0.2))
         model.add(layers.Dense(256, activation='relu'))
         model.add(layers.Dense(15, activation='softmax'))

         model.summary()
         # Compile the model, configure the optimizer
         model.compile(optimizer=optimizers.RMSprop(lr=1e-4),
                       loss='categorical_crossentropy',
                       metrics=['accuracy'])

         # Training Phase. Record the accuracy and error/loss for tuning later on
         history = model.fit_generator(datagen.flow(X_train_CNN, y_train, batch_size=8),
         steps_per_epoch = 132, epochs=80, validation_data=(X_test_CNN, y_test))

         acc = history.history['acc']
         val_acc = history.history['val_acc']
         loss = history.history['loss']
         val_loss = history.history['val_loss']

         epochs = range(1, len(acc) + 1)
```

```python
plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Test acc')
plt.title('Training and Test accuracy')
plt.legend()

plt.figure()

plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Test loss')
plt.title('Training and Test loss')
plt.legend()

plt.show()
```

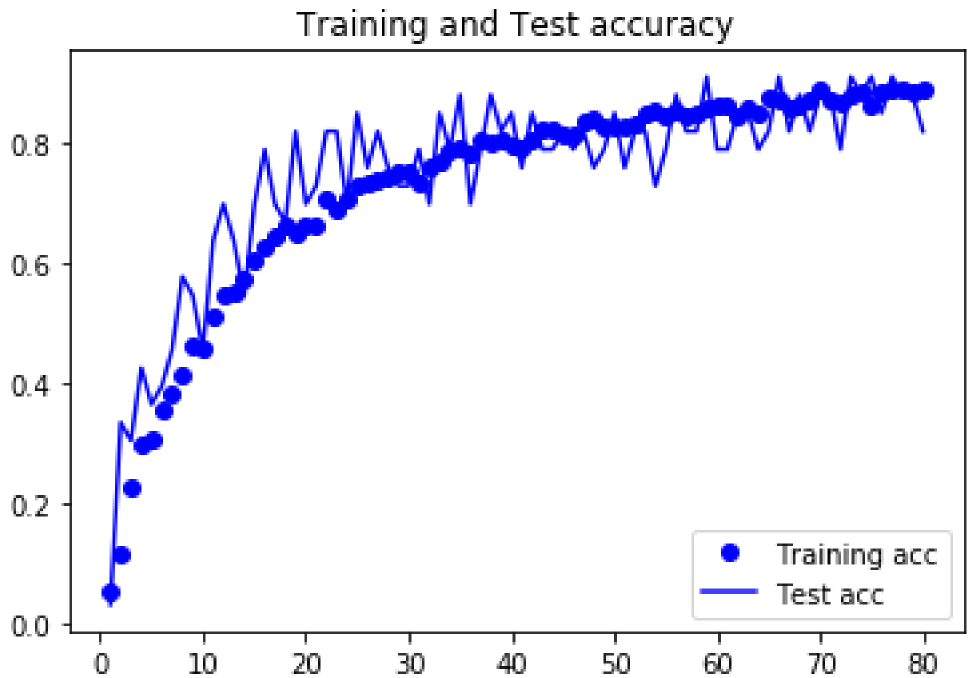
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_35 (Conv2D)           (None, 241, 318, 32)      320
_________________________________________________________________
max_pooling2d_35 (MaxPooling (None, 120, 159, 32)      0
_________________________________________________________________
conv2d_36 (Conv2D)           (None, 118, 157, 64)      18496
_________________________________________________________________
max_pooling2d_36 (MaxPooling (None, 59, 78, 64)        0
_________________________________________________________________
conv2d_37 (Conv2D)           (None, 57, 76, 128)       73856
_________________________________________________________________
max_pooling2d_37 (MaxPooling (None, 28, 38, 128)       0
_________________________________________________________________
dropout_29 (Dropout)         (None, 28, 38, 128)       0
_________________________________________________________________
conv2d_38 (Conv2D)           (None, 26, 36, 128)       147584
_________________________________________________________________
max_pooling2d_38 (MaxPooling (None, 13, 18, 128)       0
_________________________________________________________________
flatten_9 (Flatten)          (None, 29952)             0
_________________________________________________________________
dropout_30 (Dropout)         (None, 29952)             0
_________________________________________________________________
dense_25 (Dense)             (None, 512)               15335936
_________________________________________________________________
dropout_31 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_26 (Dense)             (None, 256)               131328
_________________________________________________________________
dense_27 (Dense)             (None, 15)                3855
=================================================================
Total params: 15,711,375
Trainable params: 15,711,375
Non-trainable params: 0
_________________________________________________________________
Epoch 1/80
132/132 [==============================] - 12s 94ms/step - loss: 5.8023 - acc: 0.0521
- val_loss: 2.9258 - val_acc: 0.0303
Epoch 2/80
132/132 [==============================] - 10s 74ms/step - loss: 2.6994 - acc: 0.1146
- val_loss: 2.2831 - val_acc: 0.3333
Epoch 3/80
132/132 [==============================] - 10s 76ms/step - loss: 2.3394 - acc: 0.2169
- val_loss: 2.1243 - val_acc: 0.3030
Epoch 4/80
```

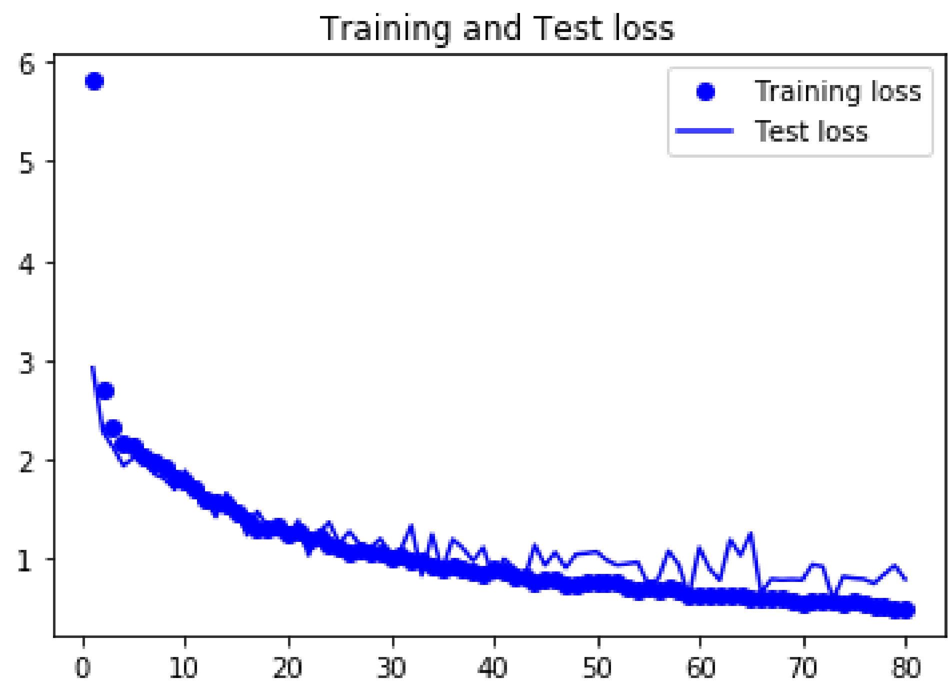
```
132/132 [==============================] - 10s 75ms/step - loss: 2.1681 - acc: 0.2926
- val_loss: 1.9397 - val_acc: 0.4242
Epoch 5/80
132/132 [==============================] - 10s 76ms/step - loss: 2.1285 - acc: 0.3059
- val_loss: 2.0229 - val_acc: 0.3636
Epoch 6/80
132/132 [==============================] - 10s 76ms/step - loss: 2.0299 - acc: 0.3560
- val_loss: 1.9740 - val_acc: 0.3939
Epoch 7/80
132/132 [==============================] - 10s 75ms/step - loss: 1.9672 - acc: 0.3826
- val_loss: 1.8593 - val_acc: 0.4545
Epoch 8/80
132/132 [==============================] - 10s 76ms/step - loss: 1.9379 - acc: 0.4110
- val_loss: 1.8090 - val_acc: 0.5758
Epoch 9/80
132/132 [==============================] - 10s 76ms/step - loss: 1.8245 - acc: 0.4583
- val_loss: 1.7018 - val_acc: 0.5455
Epoch 10/80
132/132 [==============================] - 10s 76ms/step - loss: 1.7838 - acc: 0.4583
- val_loss: 1.8918 - val_acc: 0.4545
Epoch 11/80
132/132 [==============================] - 10s 76ms/step - loss: 1.7076 - acc: 0.5076
- val_loss: 1.6919 - val_acc: 0.6364
Epoch 12/80
132/132 [==============================] - 10s 76ms/step - loss: 1.6049 - acc: 0.5464
- val_loss: 1.5951 - val_acc: 0.6970
Epoch 13/80
132/132 [==============================] - 10s 76ms/step - loss: 1.5796 - acc: 0.5502
- val_loss: 1.4263 - val_acc: 0.6364
Epoch 14/80
132/132 [==============================] - 10s 76ms/step - loss: 1.5340 - acc: 0.5729
- val_loss: 1.6652 - val_acc: 0.5455
Epoch 15/80
132/132 [==============================] - 10s 76ms/step - loss: 1.4526 - acc: 0.6032
- val_loss: 1.5272 - val_acc: 0.6970
Epoch 16/80
132/132 [==============================] - 10s 76ms/step - loss: 1.3818 - acc: 0.6307
- val_loss: 1.2410 - val_acc: 0.7879
Epoch 17/80
132/132 [==============================] - 10s 76ms/step - loss: 1.3126 - acc: 0.6458
- val_loss: 1.4817 - val_acc: 0.6970
Epoch 18/80
132/132 [==============================] - 10s 76ms/step - loss: 1.3179 - acc: 0.6629
- val_loss: 1.3312 - val_acc: 0.6667
Epoch 19/80
132/132 [==============================] - 10s 76ms/step - loss: 1.3101 - acc: 0.6496
- val_loss: 1.2363 - val_acc: 0.8182
Epoch 20/80
132/132 [==============================] - 10s 76ms/step - loss: 1.2793 - acc: 0.6563
- val_loss: 1.2395 - val_acc: 0.6970
Epoch 21/80
132/132 [==============================] - 10s 76ms/step - loss: 1.2678 - acc: 0.6657
- val_loss: 1.3884 - val_acc: 0.7273
Epoch 22/80
132/132 [==============================] - 10s 76ms/step - loss: 1.1888 - acc: 0.7027
- val_loss: 1.0411 - val_acc: 0.8182
Epoch 23/80
132/132 [==============================] - 10s 76ms/step - loss: 1.2119 - acc: 0.6884
- val_loss: 1.2647 - val_acc: 0.8182
```

```
Epoch 24/80
132/132 [==============================] - 10s 76ms/step - loss: 1.1473 - acc: 0.7045
- val_loss: 1.3725 - val_acc: 0.6970
Epoch 25/80
132/132 [==============================] - 10s 76ms/step - loss: 1.1221 - acc: 0.7310
- val_loss: 1.1626 - val_acc: 0.8485
Epoch 26/80
132/132 [==============================] - 10s 76ms/step - loss: 1.0621 - acc: 0.7320
- val_loss: 1.2785 - val_acc: 0.7576
Epoch 27/80
132/132 [==============================] - 10s 76ms/step - loss: 1.1055 - acc: 0.7311
- val_loss: 1.1459 - val_acc: 0.8182
Epoch 28/80
132/132 [==============================] - 10s 75ms/step - loss: 1.0705 - acc: 0.7452
- val_loss: 1.1145 - val_acc: 0.7576
Epoch 29/80
132/132 [==============================] - 10s 76ms/step - loss: 1.0722 - acc: 0.7509
- val_loss: 1.2046 - val_acc: 0.7273
Epoch 30/80
132/132 [==============================] - 10s 76ms/step - loss: 1.0134 - acc: 0.7519
- val_loss: 1.0483 - val_acc: 0.7273
Epoch 31/80
132/132 [==============================] - 10s 76ms/step - loss: 1.0575 - acc: 0.7339
- val_loss: 1.1000 - val_acc: 0.7879
Epoch 32/80
132/132 [==============================] - 10s 75ms/step - loss: 0.9880 - acc: 0.7623
- val_loss: 1.3395 - val_acc: 0.6970
Epoch 33/80
132/132 [==============================] - 10s 76ms/step - loss: 0.9820 - acc: 0.7652
- val_loss: 0.8361 - val_acc: 0.8485
Epoch 34/80
132/132 [==============================] - 10s 76ms/step - loss: 0.9240 - acc: 0.7860
- val_loss: 1.2504 - val_acc: 0.7879
Epoch 35/80
132/132 [==============================] - 10s 76ms/step - loss: 0.9192 - acc: 0.7870
- val_loss: 0.8927 - val_acc: 0.8788
Epoch 36/80
132/132 [==============================] - 10s 76ms/step - loss: 0.9312 - acc: 0.7869
- val_loss: 1.2016 - val_acc: 0.6970
Epoch 37/80
132/132 [==============================] - 10s 76ms/step - loss: 0.9036 - acc: 0.8077
- val_loss: 1.1132 - val_acc: 0.7879
Epoch 38/80
132/132 [==============================] - 10s 74ms/step - loss: 0.8859 - acc: 0.7927
- val_loss: 0.9837 - val_acc: 0.8788
Epoch 39/80
132/132 [==============================] - 10s 76ms/step - loss: 0.8545 - acc: 0.8040
- val_loss: 1.1185 - val_acc: 0.8182
Epoch 40/80
132/132 [==============================] - 10s 75ms/step - loss: 0.8943 - acc: 0.7992
- val_loss: 0.8035 - val_acc: 0.8485
Epoch 41/80
132/132 [==============================] - 10s 75ms/step - loss: 0.8697 - acc: 0.7935
- val_loss: 1.0030 - val_acc: 0.7576
Epoch 42/80
132/132 [==============================] - 10s 75ms/step - loss: 0.8244 - acc: 0.8049
- val_loss: 0.9162 - val_acc: 0.8485
Epoch 43/80
132/132 [==============================] - 10s 76ms/step - loss: 0.8165 - acc: 0.8229
```

```
- val_loss: 0.8067 - val_acc: 0.7879
Epoch 44/80
132/132 [==============================] - 10s 75ms/step - loss: 0.7617 - acc: 0.8219
- val_loss: 1.1432 - val_acc: 0.7879
Epoch 45/80
132/132 [==============================] - 10s 76ms/step - loss: 0.8056 - acc: 0.8125
- val_loss: 0.9402 - val_acc: 0.8182
Epoch 46/80
132/132 [==============================] - 10s 76ms/step - loss: 0.8099 - acc: 0.8097
- val_loss: 1.0697 - val_acc: 0.7879
Epoch 47/80
132/132 [==============================] - 10s 76ms/step - loss: 0.7407 - acc: 0.8381
- val_loss: 0.9147 - val_acc: 0.8182
Epoch 48/80
132/132 [==============================] - 10s 75ms/step - loss: 0.7479 - acc: 0.8390
- val_loss: 1.0492 - val_acc: 0.7576
Epoch 49/80
132/132 [==============================] - 10s 75ms/step - loss: 0.7605 - acc: 0.8257
- val_loss: 1.0605 - val_acc: 0.7879
Epoch 50/80
132/132 [==============================] - 10s 76ms/step - loss: 0.7676 - acc: 0.8267
- val_loss: 1.0765 - val_acc: 0.8485
Epoch 51/80
132/132 [==============================] - 10s 76ms/step - loss: 0.7689 - acc: 0.8305
- val_loss: 0.9909 - val_acc: 0.7576
Epoch 52/80
132/132 [==============================] - 10s 77ms/step - loss: 0.7727 - acc: 0.8305
- val_loss: 0.9406 - val_acc: 0.8182
Epoch 53/80
132/132 [==============================] - 10s 76ms/step - loss: 0.7212 - acc: 0.8466
- val_loss: 0.9554 - val_acc: 0.8182
Epoch 54/80
132/132 [==============================] - 10s 76ms/step - loss: 0.6931 - acc: 0.8541
- val_loss: 0.9701 - val_acc: 0.7273
Epoch 55/80
132/132 [==============================] - 10s 76ms/step - loss: 0.7133 - acc: 0.8456
- val_loss: 0.7548 - val_acc: 0.7879
Epoch 56/80
132/132 [==============================] - 10s 76ms/step - loss: 0.6960 - acc: 0.8513
- val_loss: 0.7998 - val_acc: 0.8788
Epoch 57/80
132/132 [==============================] - 10s 76ms/step - loss: 0.7111 - acc: 0.8419
- val_loss: 1.0838 - val_acc: 0.8182
Epoch 58/80
132/132 [==============================] - 10s 75ms/step - loss: 0.6761 - acc: 0.8485
- val_loss: 0.9463 - val_acc: 0.8182
Epoch 59/80
132/132 [==============================] - 10s 76ms/step - loss: 0.6461 - acc: 0.8579
- val_loss: 0.6283 - val_acc: 0.9091
Epoch 60/80
132/132 [==============================] - 10s 76ms/step - loss: 0.6436 - acc: 0.8627
- val_loss: 1.1186 - val_acc: 0.7879
Epoch 61/80
132/132 [==============================] - 10s 76ms/step - loss: 0.6197 - acc: 0.8636
- val_loss: 0.9047 - val_acc: 0.7879
Epoch 62/80
132/132 [==============================] - 10s 76ms/step - loss: 0.6359 - acc: 0.8437
- val_loss: 0.7896 - val_acc: 0.8485
Epoch 63/80
```

```
132/132 [==============================] - 10s 76ms/step - loss: 0.6421 - acc: 0.8551
- val_loss: 1.1957 - val_acc: 0.8485
Epoch 64/80
132/132 [==============================] - 10s 76ms/step - loss: 0.6490 - acc: 0.8466
- val_loss: 1.0389 - val_acc: 0.7879
Epoch 65/80
132/132 [==============================] - 10s 76ms/step - loss: 0.5996 - acc: 0.8759
- val_loss: 1.2614 - val_acc: 0.8182
Epoch 66/80
132/132 [==============================] - 10s 76ms/step - loss: 0.6131 - acc: 0.8674
- val_loss: 0.6667 - val_acc: 0.9091
Epoch 67/80
132/132 [==============================] - 10s 76ms/step - loss: 0.6077 - acc: 0.8598
- val_loss: 0.8001 - val_acc: 0.8182
Epoch 68/80
132/132 [==============================] - 10s 76ms/step - loss: 0.6082 - acc: 0.8636
- val_loss: 0.7906 - val_acc: 0.8788
Epoch 69/80
132/132 [==============================] - 10s 74ms/step - loss: 0.5968 - acc: 0.8693
- val_loss: 0.7960 - val_acc: 0.8182
Epoch 70/80
132/132 [==============================] - 10s 75ms/step - loss: 0.5568 - acc: 0.8873
- val_loss: 0.7906 - val_acc: 0.8788
Epoch 71/80
132/132 [==============================] - 10s 76ms/step - loss: 0.5643 - acc: 0.8740
- val_loss: 0.9474 - val_acc: 0.8788
Epoch 72/80
132/132 [==============================] - 10s 76ms/step - loss: 0.5904 - acc: 0.8646
- val_loss: 0.9266 - val_acc: 0.7879
Epoch 73/80
132/132 [==============================] - 10s 76ms/step - loss: 0.5919 - acc: 0.8788
- val_loss: 0.6006 - val_acc: 0.9091
Epoch 74/80
132/132 [==============================] - 10s 77ms/step - loss: 0.5759 - acc: 0.8769
- val_loss: 0.8271 - val_acc: 0.8788
Epoch 75/80
132/132 [==============================] - 10s 76ms/step - loss: 0.5952 - acc: 0.8608
- val_loss: 0.8094 - val_acc: 0.9091
Epoch 76/80
132/132 [==============================] - 10s 76ms/step - loss: 0.5600 - acc: 0.8816
- val_loss: 0.8003 - val_acc: 0.8485
Epoch 77/80
132/132 [==============================] - 10s 77ms/step - loss: 0.5361 - acc: 0.8817
- val_loss: 0.7583 - val_acc: 0.9091
Epoch 78/80
132/132 [==============================] - 10s 76ms/step - loss: 0.5244 - acc: 0.8854
- val_loss: 0.8500 - val_acc: 0.8788
Epoch 79/80
132/132 [==============================] - 10s 76ms/step - loss: 0.5153 - acc: 0.8826
- val_loss: 0.9373 - val_acc: 0.8788
Epoch 80/80
132/132 [==============================] - 10s 76ms/step - loss: 0.5040 - acc: 0.8835
- val_loss: 0.7997 - val_acc: 0.8182
```


Training and Test accuracy
0.0
0.2
0.4
0.6
0.8
0
10
20
30
40
50
60
70
80
Training acc
Test acc


Training and Test loss
1
2
3
4
5
6
0
10
20
30
40
50
60
70
80
Training loss
Test loss