faceexpression2

June 13, 2019

In [1]:
```
import pandas as pd
import cv2
import matplotlib.pyplot as plt
import matplotlib.image as mig
import numpy as np
import cv2
import keras
from keras.utils import to_categorical
from keras.models import Input,Sequential, Model
from keras.layers import Dense ,Flatten,Dropout,Conv2D,MaxPooling2D
from keras.layers.normalization import BatchNormalization
from keras.layers.advanced_activations import LeakyReLU
```

```
Using TensorFlow backend.
```

In [2]:
```
data=pd.read_csv('fer20131.csv')
data.head()
```

Out[2]:

	emotion	pixels	Usage
0	0	70 80 82 72 58 58 60 63 54 58 60 48 89 115 121...	Training
1	0	151 150 147 155 148 133 111 140 170 174 182 15...	Training
2	2	231 212 156 164 174 138 161 173 182 200 106 38...	Training
3	4	24 32 36 30 32 23 19 20 30 41 21 22 32 34 21 1...	Training
4	6	4 0 0 0 0 0 0 0 0 0 0 0 3 15 23 28 48 50 58 84...	Training

In [3]:
```
data['Usage'].value_counts()
```

Out[3]:
```
Training       28709
PublicTest      3589
PrivateTest     3589
Name: Usage, dtype: int64
```

In [4]:
```
data['emotion'].value_counts()
```

Out[4]:
```
3    8989
6    6198
4    6077
```

```
2    5121
0    4953
5    4002
1     547
Name: emotion, dtype: int64
```

```
In [5]: train=data[['emotion','pixels']][data["Usage"]=='Training']
        train['pixels']=train['pixels'].apply(lambda i :np.fromstring(i,sep=' '))
```

```
In [6]: train.head()
```

```
Out[6]:    emotion                                             pixels
        0        0  [70.0, 80.0, 82.0, 72.0, 58.0, 58.0, 60.0, 63...
        1        0  [151.0, 150.0, 147.0, 155.0, 148.0, 133.0, 111...
        2        2  [231.0, 212.0, 156.0, 164.0, 174.0, 138.0, 161...
        3        4  [24.0, 32.0, 36.0, 30.0, 32.0, 23.0, 19.0, 20...
        4        6  [4.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
```

```
In [11]: pbtest=data[['emotion','pixels']][data["Usage"]=='PublicTest']
         pbtest['pixels']=pbtest['pixels'].apply(lambda i : np.fromstring(i,sep=' '))
```

```
In [12]: prtest=data[['emotion','pixels']][data["Usage"]=='PrivateTest']
         prtest['pixels']=prtest['pixels'].apply(lambda i : np.fromstring(i,sep=' '))
```

```
In [14]: pbtest = pbtest.reset_index(drop=True)
         prtest = prtest.reset_index(drop=True)

         pbtest.head()
```

```
Out[14]:    emotion                                             pixels
         0        0  [254.0, 254.0, 254.0, 254.0, 254.0, 249.0, 255...
         1        1  [156.0, 184.0, 198.0, 202.0, 204.0, 207.0, 210...
         2        4  [69.0, 118.0, 61.0, 60.0, 96.0, 121.0, 103.0, ...
         3        6  [205.0, 203.0, 236.0, 157.0, 83.0, 158.0, 120...
         4        3  [87.0, 79.0, 74.0, 66.0, 74.0, 96.0, 77.0, 80...
```

```
In [15]: prtest.head()
```

```
Out[15]:    emotion                                             pixels
         0        0  [170.0, 118.0, 101.0, 88.0, 88.0, 75.0, 78.0, ...
         1        5  [7.0, 5.0, 8.0, 6.0, 7.0, 3.0, 2.0, 6.0, 5.0, ...
         2        6  [232.0, 240.0, 241.0, 239.0, 237.0, 235.0, 246...
         3        4  [200.0, 197.0, 149.0, 139.0, 156.0, 89.0, 111...
         4        2  [40.0, 28.0, 33.0, 56.0, 45.0, 33.0, 31.0, 78...
```

```
In [16]: pr_xtest=np.vstack(prtest['pixels'].values)
         pr_ytest=np.array(prtest['emotion'].values)
```

```
In [17]: pr_xtest
```

```
Out[17]: array([[170., 118., 101., ..., 159., 133., 131.],
                [  7.,   5.,   8., ...,  72.,  57.,  52.],
                [232., 240., 241., ...,   4.,   4.,   9.],
                ...,
                [ 17.,  17.,  16., ..., 154., 133., 113.],
                [ 30.,  28.,  28., ...,  35.,  30.,  28.],
                [ 19.,  13.,  14., ..., 189., 199., 201.]])

In [18]: pr_ytest

Out[18]: array([0, 5, 6, ..., 0, 3, 2], dtype=int64)

In [19]: pr_ytest=to_categorical(pr_ytest)
         pr_ytest

Out[19]: array([[1., 0., 0., ..., 0., 0., 0.],
                [0., 0., 0., ..., 0., 1., 0.],
                [0., 0., 0., ..., 0., 0., 1.],
                ...,
                [1., 0., 0., ..., 0., 0., 0.],
                [0., 0., 0., ..., 0., 0., 0.],
                [0., 0., 1., ..., 0., 0., 0.]], dtype=float32)

In [20]: Xtrain=np.vstack(train['pixels'].values)
         Ytrain=np.array(train['emotion'].values)
         Xtest=np.vstack(pb['pixels'].values)
         Ytest=np.array(pb['emotion'].values)

In [21]: Xtrain

Out[21]: array([[ 70.,  80.,  82., ..., 106., 109.,  82.],
                [151., 150., 147., ..., 193., 183., 184.],
                [231., 212., 156., ...,  88., 110., 152.],
                ...,
                [ 74.,  81.,  87., ..., 188., 187., 187.],
                [222., 227., 203., ..., 136., 136., 134.],
                [195., 199., 205., ...,   6.,  15.,  38.]])

In [22]: Xtest

Out[22]: array([[254., 254., 254., ...,  42., 129., 180.],
                [156., 184., 198., ..., 172., 167., 161.],
                [ 69., 118.,  61., ...,  88.,  87.,  90.],
                ...,
                [255., 255., 255., ...,  48.,  50.,  46.],
                [ 33.,  25.,  31., ...,   4.,   5.,   4.],
                [ 61.,  63.,  59., ..., 113., 165., 180.]])

In [23]: Ytrain
```

```
Out[23]: array([0, 0, 2, ..., 4, 0, 4], dtype=int64)

In [24]: Ytest

Out[24]: array([0, 1, 4, ..., 4, 4, 4], dtype=int64)

In [25]: Ytrain=to_categorical(Ytrain)
         Ytest=to_categorical(Ytest)

         Ytrain

Out[25]: array([[1., 0., 0., ..., 0., 0., 0.],
                [1., 0., 0., ..., 0., 0., 0.],
                [0., 0., 1., ..., 0., 0., 0.],
                ...,
                [0., 0., 0., ..., 1., 0., 0.],
                [1., 0., 0., ..., 0., 0., 0.],
                [0., 0., 0., ..., 1., 0., 0.]], dtype=float32)

In [26]: Ytest

Out[26]: array([[1., 0., 0., ..., 0., 0., 0.],
                [0., 1., 0., ..., 0., 0., 0.],
                [0., 0., 0., ..., 1., 0., 0.],
                ...,
                [0., 0., 0., ..., 1., 0., 0.],
                [0., 0., 0., ..., 1., 0., 0.],
                [0., 0., 0., ..., 1., 0., 0.]], dtype=float32)

In [28]: Xtrain.shape

Out[28]: (28709, 2304)

In [29]: Xtest.shape

Out[29]: (3589, 2304)

In [30]: Xtrain=Xtrain.reshape(-1,48,48,1)
         Xtest=Xtest.reshape(-1,48,48,1)

In [31]: Xtrain.shape

Out[31]: (28709, 48, 48, 1)

In [32]: Xtest.shape

Out[32]: (3589, 48, 48, 1)

In [33]: batch_size=128
         epochs=10
         number_of_classes=7
```

```
In [34]: model=Sequential()
         model.add(Conv2D(32,kernel_size=(3,3),activation='relu',padding='same',
                          input_shape=(48,48,1)))
         model.add(BatchNormalization())
         model.add(LeakyReLU(alpha=0.1))
         model.add(MaxPooling2D((2,2),padding='same'))

         model.add(Conv2D(64,kernel_size=(3,3),activation='relu',padding='same'))
         model.add(BatchNormalization())
         model.add(LeakyReLU(alpha=0.1))
         model.add(MaxPooling2D((2,2),padding='same'))

         model.add(Conv2D(128, (3, 3), activation='relu',padding='same'))
         model.add(BatchNormalization())
         model.add(LeakyReLU(alpha=0.1))
         model.add(MaxPooling2D(pool_size=(2, 2),padding='same'))

         model.add(Flatten())
         model.add(Dense(128,activation='relu'))
         model.add(LeakyReLU(alpha=0.1))
         model.add(Dense(number_of_classes,activation='softmax'))
         model.compile(loss=keras.losses.categorical_crossentropy,
                       optimizer=keras.optimizers.Adam(),metrics=['accuracy'])
         model.summary()

WARNING:tensorflow:From C:\Users\pheni\Anaconda3\envs\myfirstcondaenv\lib\site-packages\tensorfl
Instructions for updating:
Colocations handled automatically by placer.
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 48, 48, 32)        320
_________________________________________________________________
batch_normalization_1 (Batch (None, 48, 48, 32)        128
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 48, 48, 32)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 24, 24, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 24, 24, 64)        18496
_________________________________________________________________
batch_normalization_2 (Batch (None, 24, 24, 64)        256
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 24, 24, 64)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 12, 12, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 12, 12, 128)       73856
```

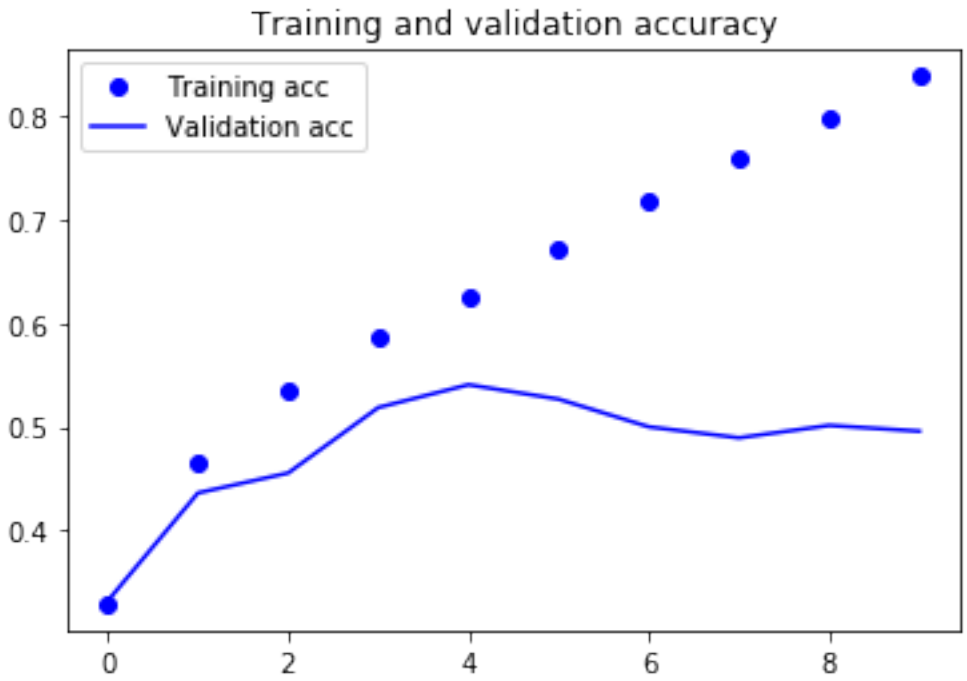
```
_________________________________________________________________
batch_normalization_3 (Batch (None, 12, 12, 128)       512
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 12, 12, 128)       0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 6, 6, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 4608)              0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               589952
_________________________________________________________________
leaky_re_lu_4 (LeakyReLU)    (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 7)                 903
=================================================================
Total params: 684,423
Trainable params: 683,975
Non-trainable params: 448
_________________________________________________________________
```

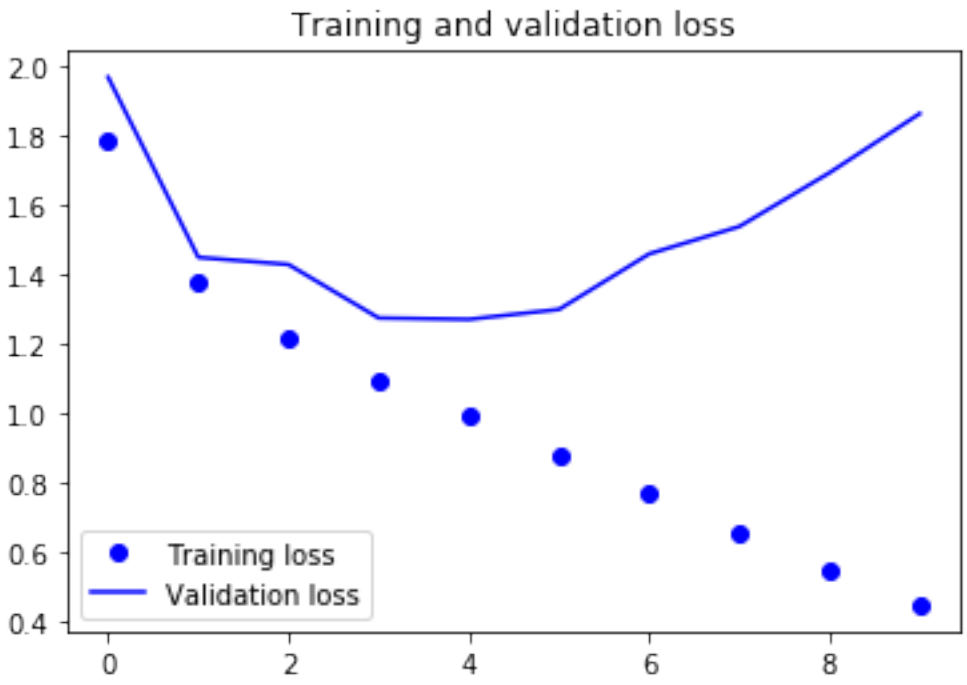
```
In [35]: trained_model=model.fit(Xtrain, Ytrain, batch_size=batch_size,
                                 epochs=epochs,verbose=1,validation_data=(Xtest, Ytest))
```

```
WARNING:tensorflow:From C:\Users\pheni\Anaconda3\envs\myfirstcondaenv\lib\site-packages\tensorfl
Instructions for updating:
Use tf.cast instead.
Train on 28709 samples, validate on 3589 samples
Epoch 1/10
28709/28709 [==============================] - 228s 8ms/step - loss: 1.7832 - acc: 0.3291 - val_
Epoch 2/10
28709/28709 [==============================] - 223s 8ms/step - loss: 1.3777 - acc: 0.4668 - val_
Epoch 3/10
28709/28709 [==============================] - 224s 8ms/step - loss: 1.2143 - acc: 0.5354 - val_
Epoch 4/10
28709/28709 [==============================] - 223s 8ms/step - loss: 1.0940 - acc: 0.5857 - val_
Epoch 5/10
28709/28709 [==============================] - 223s 8ms/step - loss: 0.9912 - acc: 0.6267 - val_
Epoch 6/10
28709/28709 [==============================] - 227s 8ms/step - loss: 0.8783 - acc: 0.6712 - val_
Epoch 7/10
28709/28709 [==============================] - 224s 8ms/step - loss: 0.7662 - acc: 0.7179 - val_
Epoch 8/10
28709/28709 [==============================] - 224s 8ms/step - loss: 0.6558 - acc: 0.7598 - val_
Epoch 9/10
28709/28709 [==============================] - 225s 8ms/step - loss: 0.5450 - acc: 0.7992 - val_
Epoch 10/10
28709/28709 [==============================] - 225s 8ms/step - loss: 0.4464 - acc: 0.8388 - val_
```

In [36]:
```
model.save('facial_1')
```

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

epochs = range(len(acc))

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

plt.figure()

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

plt.show()
```






```
In [37]: pr_xtest=pr_xtest.reshape(-1,48,48,1)
         score = model.evaluate(pr_xtest, pr_ytest, verbose=0)
         print ("model %s: %.2f%%" % (model.metrics_names[1], score[1]*100))
```

```
model acc: 51.27%
```

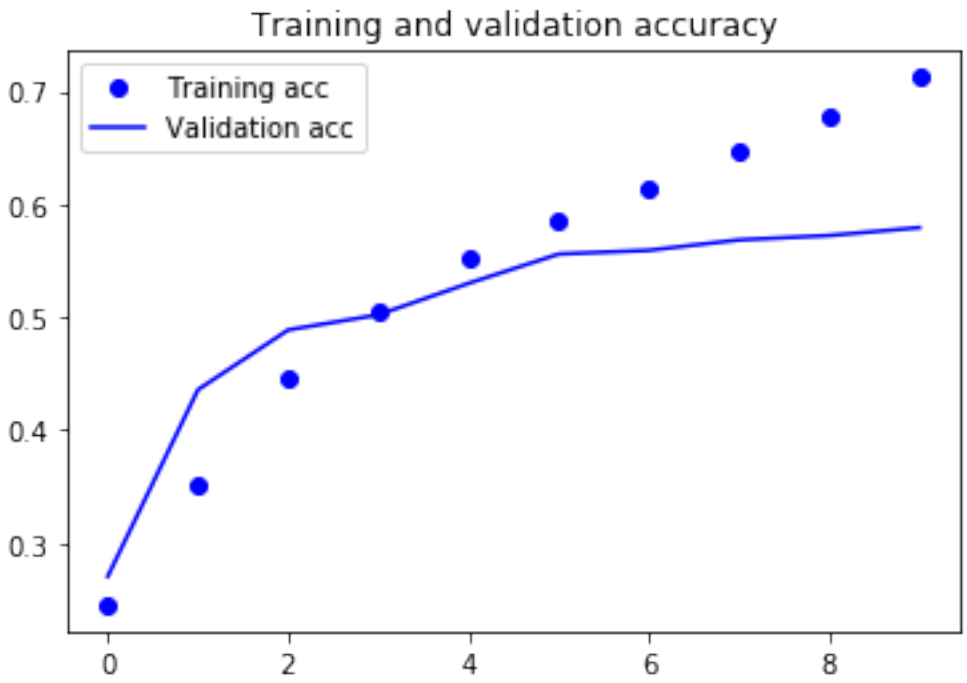
```
In [40]: model2 = Sequential()
         model2.add(Conv2D(32, (3, 3), padding='same', activation='relu',
                                 input_shape=(48, 48, 1)))
         model2.add(Conv2D(32, (3, 3), padding='same', activation='relu'))
         model2.add(Conv2D(32, (3, 3), padding='same', activation='relu'))
         model2.add(MaxPooling2D(pool_size=(2, 2)))

         model2.add(Conv2D(64, (3, 3), padding='same', activation='relu'))
         model2.add(Conv2D(64, (3, 3), padding='same', activation='relu'))
         model2.add(Conv2D(64, (3, 3), padding='same', activation='relu'))
         model2.add(MaxPooling2D(pool_size=(2, 2)))

         model2.add(Conv2D(128, (3, 3), padding='same', activation='relu'))
         model2.add(Conv2D(128, (3, 3), padding='same', activation='relu'))
         model2.add(Conv2D(128, (3, 3), padding='same', activation='relu'))
         model2.add(MaxPooling2D(pool_size=(2, 2)))

         model2.add(Flatten())  # this converts our 3D feature maps to 1D feature vectors
```

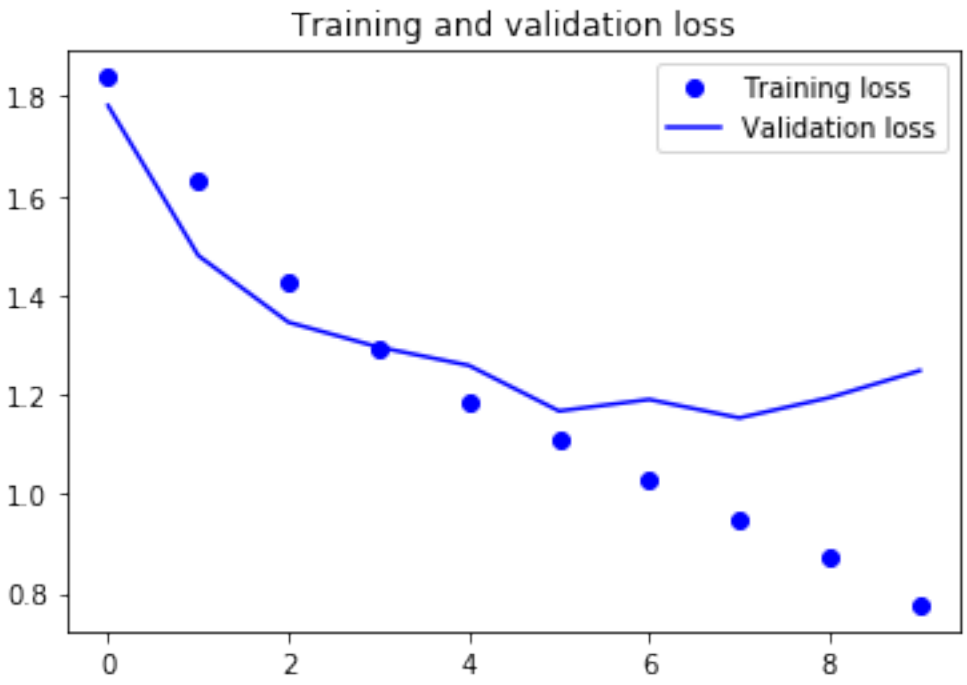
```
model2.add(Dense(64, activation='relu'))
model2.add(Dense(32, activation='relu'))
model2.add(Dense(number_of_classes, activation='softmax'))

# optimizer:
model2.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model2.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 48, 48, 32)        320
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 48, 48, 32)        9248
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 48, 48, 32)        9248
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 24, 24, 32)        0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 24, 24, 64)        18496
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 24, 24, 64)        36928
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 24, 24, 64)        36928
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 12, 12, 64)        0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 12, 12, 128)       73856
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 12, 12, 128)       147584
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 12, 12, 128)       147584
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 6, 6, 128)         0
_________________________________________________________________
flatten_2 (Flatten)          (None, 4608)              0
_________________________________________________________________
dense_3 (Dense)              (None, 64)                294976
_________________________________________________________________
dense_4 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_5 (Dense)              (None, 7)                 231
=================================================================
Total params: 777,479
Trainable params: 777,479
Non-trainable params: 0
_________________________________________________________________
```

```
In [41]: print ('Training....')
         #fit
         nb_epoch = 10
         batch_size = 128

         trained_model_2 = model2.fit(Xtrain, Ytrain, epochs=nb_epoch,
                                      batch_size=batch_size, validation_data=(Xtest, Ytest), ver

Training...
Train on 28709 samples, validate on 3589 samples
Epoch 1/10
28709/28709 [==============================] - 221s 8ms/step - loss: 1.8388 - acc: 0.2452 - val_
Epoch 2/10
28709/28709 [==============================] - 221s 8ms/step - loss: 1.6318 - acc: 0.3503 - val_
Epoch 3/10
28709/28709 [==============================] - 221s 8ms/step - loss: 1.4266 - acc: 0.4470 - val_
Epoch 4/10
28709/28709 [==============================] - 231s 8ms/step - loss: 1.2943 - acc: 0.5055 - val_
Epoch 5/10
28709/28709 [==============================] - 226s 8ms/step - loss: 1.1857 - acc: 0.5521 - val_
Epoch 6/10
28709/28709 [==============================] - 222s 8ms/step - loss: 1.1068 - acc: 0.5861 - val_
Epoch 7/10
28709/28709 [==============================] - 223s 8ms/step - loss: 1.0314 - acc: 0.6141 - val_
Epoch 8/10
28709/28709 [==============================] - 718s 25ms/step - loss: 0.9474 - acc: 0.6477 - val
Epoch 9/10
28709/28709 [==============================] - 503s 18ms/step - loss: 0.8733 - acc: 0.6775 - val
Epoch 10/10
28709/28709 [==============================] - 507s 18ms/step - loss: 0.7768 - acc: 0.7125 - val


In [42]: model2.save('facial_2')

         acc = trained_model_2.history['acc']
         val_acc = trained_model_2.history['val_acc']
         loss = trained_model_2.history['loss']
         val_loss = trained_model_2.history['val_loss']

         epochs = range(len(acc))

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

         plt.figure()
```

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

plt.show()
```






```
In [43]: score = model2.evaluate(pr_xtest, pr_ytest, verbose=0)
         print ("model 2 %s: %.2f%%" % (model2.metrics_names[1], score[1]*100))
```

```
model 2 acc: 57.68%
```

```
In [ ]:
```