In [1]:
```
import numpy as np
import pandas as pd

import os
#os.sys.path
import sys
sys.path.append('../src')

from pathlib import Path

import matplotlib.pyplot as plt
from PIL import Image
from numpy import asarray
import cv2

import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
```

In [2]:
```
# importing tensorflow model libraries
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, AveragePool
from tensorflow.keras.layers import Dense, Activation, Dropout, Flatt
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.metrics import categorical_accuracy
from tensorflow.keras.models import model_from_json,load_model
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.callbacks import ModelCheckpoint, CSVLogger, Ea
from tensorflow.keras.optimizers import *
import tensorflow.keras.backend as K
import json
import time
```

In [3]:
```
from sklearn.model_selection import train_test_split
```

IMGS

In [4]:
```
inp1 = Path.home()/'Iron'/'inp1'
```

In [5]:
```
train_imgs = inp1/'TRAIN'
train_imgs_haar = inp1/'TRAIN_haar'
val_imgs = inp1/'VALIDATION'
test_imgs = inp1/'TEST'
demo_imgs = Path.cwd().parent/'demo'
demo_imgs_faces = Path.cwd().parent/'demo_haar'
demo_imgs_haar = Path.cwd().parent/'demo_faces'
```

DATASETS to check

In [6]:
```
# df_fer_ok --> sin dummies
# df_fer_top --> con dummies!
```

In [6]:
```
df_fer = pd.read_csv(inp1/'Fer.csv',encoding = "ISO-8859-1")
df_train = pd.read_csv(inp1/'Training_Data.csv',encoding = "ISO-8859-
df_test = pd.read_csv(inp1/'Testing_Data.csv',encoding = "ISO-8859-1"
df_val = pd.read_csv(inp1/'Validation_Data.csv',encoding = "ISO-8859-
```

In [7]:
```
face_cascade = cv2.CascadeClassifier('../src/haarcascade_frontalface_
eye_cascade = cv2.CascadeClassifier('../src/haarcascade_eye.xml')
smile_cascade = cv2.CascadeClassifier('../src/haarcascade_smile.xml')
```

In [8]:
```
df = df_fer.copy()
print(df.shape, df.size, df.columns)
```

```
(35887, 3) 107661 Index(['emotion', 'pixels', 'Usage'], dtype='objec
t')
```

In [9]:
```
df.emotion.unique()
```

Out[9]:
```
array([0, 2, 4, 6, 3, 5, 1])
```

In [10]:
```
emos = {0:'Angry',1: 'Disgust',2:'Fear',3:'Happy',4:'Sad',5:'Surprise
#df['emos'] = df.emotion.map(emos)
df['emotion_names'] = df.emotion.map(emos)
```

In [11]:
```
emos2 = {0:'unhappy', 1:'unhappy',2:'unhappy',3:'happy',4:'unhappy',5
```

In [12]:
```
df['emo'] = df.emotion.map(emos2).to_numpy()
```

In [13]:
```
df.head()
```

Out[13]:

	emotion	pixels	Usage	emotion_names	emo

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

In [14]:
```
emo = pd.get_dummies(df['emo']).to_numpy()
```

In [15]:
```
df2 = pd.get_dummies(df['emotion']).to_numpy()
```

In [17]:
```
df
```

Out[17]:

	emotion	pixels	Usage	emotion_names	emo
0	0	70 80 82 72 58 58 60 63 54 58 60 48 89 115 121...	Training	Angry	unhappy
1	0	151 150 147 155 148 133 111 140 170 174 182 15...	Training	Angry	unhappy
2	2	231 212 156 164 174 138 161 173 182 200 106 38...	Training	Fear	unhappy
3	4	24 32 36 30 32 23 19 20 30 41 21 22 32 34 21 1...	Training	Sad	unhappy
4	6	4 0 0 0 0 0 0 0 0 0 0 0 3 15 23 28 48 50 58 84...	Training	Neutral	unhappy
...	...	...	...	...	...
35882	6	50 36 17 22 23 29 33 39 34 37 37 37 39 43 48 5...	PrivateTest	Neutral	unhappy
35883	3	178 174 172 173 181 188 191 194 196 199 200 20...	PrivateTest	Happy	happy
35884	0	17 17 16 23 28 22 19 17 25 26 20 24 31 19 27 9...	PrivateTest	Angry	unhappy
35885	3	30 28 28 29 31 30 42 68 79 81 77 67 67 71 63 6...	PrivateTest	Happy	happy
35886	2	19 13 14 12 13 16 21 33 50 57 71 84 97 108 122...	PrivateTest	Fear	unhappy

35887 rows × 5 columns

In [18]:
```
df.columns
```

Out[18]:
```
Index(['emotion', 'pixels', 'Usage', 'emotion_names', 'emo'], dtype
='object')
```

In [19]:
```
drpp = ['emotion', 'Usage']
df2 = df.drop(drpp, axis=1)
```

In [20]:
```
pth1 = Path.cwd().parent.parent
```

In [21]:
```
pth2 = Path.home()/'Iron'/'data_processed'
```

In [22]:
```
df2.to_csv(str(pth2/'df_fer_a.csv'))
```

In [23]:
```
df3 = df2.copy()
```

In [24]:
```
df3['pixar1'] = [[float(x) for x in each.split()] for each in df3['pi
df3['pixar2'] = df3['pixar1'].apply(lambda x: np.asarray(x).reshape(4
```

In [25]:
```
df3['emo_arr1'] = df3.pixar2.copy()
```

In [26]:
```
df3['emo_arr'] = df3['emo_arr1'].apply(lambda x: np.array([[[c] for c
```

In [28]:
```
df5 = df3.copy()
```

In [29]:
```
df5.head(2)
```

Out[29]:

	pixels	emotion_names	emo	pixar1	pixar2	emo_arr1	emo_arr

	pixels	emotion_names	emo	pixar1	pixar2	emo_arr1	emo_arr
0	70 80 82 72 58 58 60 63 54 58 60 48 89 115 121...	Angry	unhappy	[70.0, 80.0, 82.0, 72.0, 58.0, 58.0, 60.0, 63....	[[70.0, 80.0, 82.0, 72.0, 58.0, 58.0, 60.0, 63...	[[70.0, 80.0, 82.0, 72.0, 58.0, 58.0, 60.0, 63...	[[[70.0], [80.0], [82.0], [72.0], [58.0], [58....
1	151 150 147 155 148 133 111 140 170 174 182 15...	Angry	unhappy	[151.0, 150.0, 147.0, 155.0, 148.0, 133.0, 111...	[[151.0, 150.0, 147.0, 155.0, 148.0, 133.0, 11...	[[151.0, 150.0, 147.0, 155.0, 148.0, 133.0, 11...	[[[151.0], [150.0], [147.0], [155.0], [148.0],...

In [31]:
```
df5.to_csv(pth2/'df_fer_ok.csv')
```

In [32]:
```
df5.head()
```

Out[32]:

	pixels	emotion_names	emo	pixar1	pixar2	emo_arr1	emo_arr
0	70 80 82 72 58 58 60 63 54 58 60 48 89 115 121...	Angry	unhappy	[70.0, 80.0, 82.0, 72.0, 58.0, 58.0, 60.0, 63....	[[70.0, 80.0, 82.0, 72.0, 58.0, 58.0, 60.0, 63...	[[70.0, 80.0, 82.0, 72.0, 58.0, 58.0, 60.0, 63...	[[[70.0], [80.0], [82.0], [72.0], [58.0], [58....
1	151 150 147 155 148 133 111 140 170 174 182 15...	Angry	unhappy	[151.0, 150.0, 147.0, 155.0, 148.0, 133.0, 111...	[[151.0, 150.0, 147.0, 155.0, 148.0, 133.0, 11...	[[151.0, 150.0, 147.0, 155.0, 148.0, 133.0, 11...	[[[151.0], [150.0], [147.0], [155.0], [148.0],...
2	231 212 156 164 174 138 161 173 182 200 106 38...	Fear	unhappy	[231.0, 212.0, 156.0, 164.0, 174.0, 138.0, 161...	[[231.0, 212.0, 156.0, 164.0, 174.0, 138.0, 16...	[[231.0, 212.0, 156.0, 164.0, 174.0, 138.0, 16...	[[[231.0], [212.0], [156.0], [164.0], [174.0],...

	pixels	emotion_names	emo	pixar1	pixar2	emo_arr1	emo_arr
3	24 32 36 30 32 23 19 20 30 41 21 22 32 34 21 1...	Sad	unhappy	[24.0, 32.0, 36.0, 30.0, 32.0, 23.0, 19.0, 20....	[[24.0, 32.0, 36.0, 30.0, 32.0, 23.0, 19.0, 20...	[[24.0, 32.0, 36.0, 30.0, 32.0, 23.0, 19.0, 20...	[[[24.0], [32.0], [36.0], [30.0], [32.0], [23....
4	4 0 0 0 0 0 0 0 0 0 0 0 3 15 23 28 48 50 58 84...	Neutral	unhappy	[4.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[[4.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...	[[4.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...	[[[4.0], [0.0], [0.0], [0.0], [0.0], [0.0], [0...

In [37]:
```
df6 = pd.get_dummies(df5, columns=['emo'])
```

In [38]:
```
df6.head()
```

Out[38]:

	pixels	emotion_names	pixar1	pixar2	emo_arr1	emo_arr	emo_happy	emo_u
0	70 80 82 72 58 58 60 63 54 58 60 48 89 115 121...	Angry	[70.0, 80.0, 82.0, 72.0, 58.0, 58.0, 60.0, 63....	[[70.0, 80.0, 82.0, 72.0, 58.0, 58.0, 60.0, 63...	[[70.0, 80.0, 82.0, 72.0, 58.0, 58.0, 60.0, 63...	[[[70.0], [80.0], [82.0], [72.0], [58.0], [58....	0	
1	151 150 147 155 148 133 111 140 170 174 182 15...	Angry	[151.0, 150.0, 147.0, 155.0, 148.0, 133.0, 111...	[[151.0, 150.0, 147.0, 155.0, 148.0, 133.0, 11...	[[151.0, 150.0, 147.0, 155.0, 148.0, 133.0, 11...	[[[151.0], [150.0], [147.0], [155.0], [148.0],...	0	

	pixels	emotion_names	pixar1	pixar2	emo_arr1	emo_arr	emo_happy	emo_u
2	231 212 156 164 174 138 161 173 182 200 106 38...	Fear	[231.0, 212.0, 156.0, 164.0, 174.0, 138.0, 161...	[[231.0, 212.0, 156.0, 164.0, 174.0, 138.0, 16...	[[231.0, 212.0, 156.0, 164.0, 174.0, 138.0, 16...	[[[231.0], [212.0], [156.0], [164.0], [174.0],...	0	
3	24 32 36 30 32 23 19 20 30 41 21 22 32 34 21 1...	Sad	[24.0, 32.0, 36.0, 30.0, 32.0, 23.0, 19.0, 20....	[[24.0, 32.0, 36.0, 30.0, 32.0, 23.0, 19.0, 20...	[[24.0, 32.0, 36.0, 30.0, 32.0, 23.0, 19.0, 20...	[[[24.0], [32.0], [36.0], [30.0], [32.0], [23....	0	
4	4 0 0 0 0 0 0 0 0 0 0 0 3 15 23 28 48 50 58 84...	Neutral	[4.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[[4.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...	[[4.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...	[[[4.0], [0.0], [0.0], [0.0], [0.0], [0.0], [0...	0	

In [39]:
```
df6['dums'] = df6[['emo_happy','emo_unhappy']].apply(lambda x: pd.Ser
```

Save

In [7]:
```
pth2
```

Save

```
---------------------------------------------------------------------
------
NameError                                 Traceback (most recent call
last)
<ipython-input-7-4ad6c53fb0a4> in <module>
----> 1 pth2

NameError: name 'pth2' is not defined
```

In [40]:
```
df6.to_csv(pth2/'df_fer_top.csv')
```

Save

In [43]:
```
df7 = pd.read_csv(pth2/'df_fer_top.csv')
```

Save

Out[43]: (35887, 9)

In [42]:
```
df6.head()
```

Save

Out[42]:

	pixels	emotion_names	pixar1	pixar2	emo_arr1	emo_arr	emo_happy	emo_u
0	70 80 82 72 58 58 60 63 54 58 60 48 89 115 121...	Angry	[70.0, 80.0, 82.0, 72.0, 58.0, 58.0, 60.0, 63....	[[70.0, 80.0, 82.0, 72.0, 58.0, 58.0, 60.0, 63...	[[70.0, 80.0, 82.0, 72.0, 58.0, 58.0, 60.0, 63...	[[[70.0], [80.0], [82.0], [72.0], [58.0], [58....	0	
1	151 150 147 155 148 133 111 140 170 174 182 15...	Angry	[151.0, 150.0, 147.0, 155.0, 148.0, 133.0, 111...	[[151.0, 150.0, 147.0, 155.0, 148.0, 133.0, 11...	[[151.0, 150.0, 147.0, 155.0, 148.0, 133.0, 11...	[[[151.0], [150.0], [147.0], [155.0], [148.0],...	0	
2	231 212 156 164 174 138 161 173 182 200 106 38...	Fear	[231.0, 212.0, 156.0, 164.0, 174.0, 138.0, 161...	[[231.0, 212.0, 156.0, 164.0, 174.0, 138.0, 16...	[[231.0, 212.0, 156.0, 164.0, 174.0, 138.0, 16...	[[[231.0], [212.0], [156.0], [164.0], [174.0],...	0	
3	24 32 36 30 32 23 19 20 30 41 21 22 32 34 21 1...	Sad	[24.0, 32.0, 36.0, 30.0, 32.0, 23.0, 19.0, 20....	[[24.0, 32.0, 36.0, 30.0, 32.0, 23.0, 19.0, 20...	[[24.0, 32.0, 36.0, 30.0, 32.0, 23.0, 19.0, 20...	[[[24.0], [32.0], [36.0], [30.0], [32.0], [23....	0	
4	4 0 0 0 0 0 0 0 0 0 0 0 3 15 23 28 48 50 58 84...	Neutral	[4.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[[4.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...	[[4.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...	[[[4.0], [0.0], [0.0], [0.0], [0.0], [0.0], [0...	0	

In [45]:
```
df6.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 35887 entries, 0 to 35886
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   pixels         35887 non-null  object
 1   emotion_names  35887 non-null  object
 2   pixar1         35887 non-null  object
 3   pixar2         35887 non-null  object
 4   emo_arr1       35887 non-null  object
 5   emo_arr        35887 non-null  object
 6   emo_happy      35887 non-null  uint8 
 7   emo_unhappy    35887 non-null  uint8 
 8   dums           35887 non-null  object
dtypes: object(7), uint8(2)
memory usage: 2.0+ MB
```

In [50]:
```
type(df6.pixar1[0])
```

Out[50]:
```
list
```

In [54]:
```
df6.emo_arr[0].shape
```

Out[54]:
```
(48, 48, 1)
```

EL wueno es emo_arr

In [56]:
```
X = (np.stack(df6['emo_arr'])) / 255.0
y = np.stack(df6.dums)
X.shape, y.shape
```

Out[56]:
```
((35887, 48, 48, 1), (35887, 2))
```

In [59]:
```
X_train, X_testval, y_train, y_testval = train_test_split(X,y,test_si
```

In [60]:
```
X_test, X_val, y_test, y_val = train_test_split(X_testval,y_testval,t
```

In [ ]:
```

```

In [61]:
```
X_train.shape
```

Out[61]: (28709, 48, 48, 1)

In [62]:
```
y_train.shape
```

Out[62]: (28709, 2)

In [63]:
```
X_test.shape
```

Out[63]: (3589, 48, 48, 1)

In [64]:
```
y_test.shape
```

Out[64]: (3589, 2)

In [65]:
```
X_val.shape
```

Out[65]: (3589, 48, 48, 1)

In [66]:
```
y_val.shape
```

Out[66]: (3589, 2)

In [67]:
```
X_test.shape, X_val.shape, y_test.shape, y_val.shape
```

Out[67]: ((3589, 48, 48, 1), (3589, 48, 48, 1), (3589, 2), (3589, 2))

In [ ]:

In [ ]:

In [ ]:
```
def transfImag(path, new_path):

    # ROCKET
    """
    recibe carpeta, en cada foto de esa carpeta:
    lectura
```

```
    to gray
    facecascade
    por xywh en cada cara:
        array
        reshape array a 2d
        stack array 3d
        normalizar
        expand 4d
    devuelve array x cada foto para pasarselo al modelo
    """
    counter_imgs = 0
    counter_faces = 0

    X_ = pd.Series([], dtype='float64')

    for file in sorted(path.iterdir()):

        counter_imgs += 1

        input_img1 = cv2.imread(str(file))
        input_img2 = cv2.cvtColor(input_img1, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(input_img2, 1.25, 6)

        for (x,y,w,h) in faces:

            counter_faces += 1
            img_data1 = input_img2 [y:y+h,x:x+w]
            img_data2 = cv2.resize (img_data1,(48,48))

            img_data3 = np.stack(img_data2)
            img_data4 = img_data2 / 255.0
            img_data5 = img_data3 / 255.0

            img_data6 = np.expand_dims(img_data5,axis=0).reshape(np.e

            print(img_data6.shape)
            print(img_data6)

            img_datashow = img_data3*255
            img_show = Image.fromarray(img_datashow)
            file_to_save = file.name.replace(".",f"_face{counter_face
            img_show.save(str(new_path/file_to_save))

            counter_faces = 0

            arr_for_model = img_data6


            return arr_for_model

    print('YAAA!')
    return counter_imgs
```

Save

In [78]:

```
def base_model():
    model = Sequential()
```

```
    input_shape = (48,48,1)
    #1st convolution layer
    model.add(Conv2D(64, (5, 5), input_shape=input_shape,activation='
    model.add(Conv2D(64, (5, 5), activation='relu', padding='same'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.5))

    #2nd convolution layer
    model.add(Conv2D(128, (5, 5),activation='relu',padding='same'))
    model.add(Conv2D(128, (5, 5),activation='relu',padding='same'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.5))

    #3rd convolution layer
    model.add(Conv2D(256, (3, 3),activation='relu',padding='same'))
    model.add(Conv2D(256, (3, 3),activation='relu',padding='same'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.5))

    model.add(Flatten())
    model.add(Dense(128))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(Dropout(0.2))
    model.add(Dense(2))
    model.add(Activation('softmax'))
    #model.add(Dense(1, activation='sigmoid')) (should be used for bi

    my_optimiser = tf.keras.optimizers.Adam(
    learning_rate = 0.001, beta_1=0.9, beta_2=0.999,
        epsilon=1e-07, amsgrad=False, name='Adam')

    model.compile(loss='categorical_crossentropy',
                  metrics=['accuracy'],
                  optimizer=my_optimiser)

    return model
```

Save

In [79]:
```
X_train.shape, y_train.shape
```

Save

Out[79]: ((28709, 48, 48, 1), (28709, 2))

In [80]:
```
X_val.shape, y_val.shape
```

Save

Out[80]: ((3589, 48, 48, 1), (3589, 2))

In [ ]:
```
model_2 = base_model()

model_2.fit(X_train, y_train,
```

```
            validation_data=(X_val, y_val),
            epochs=20,
            verbose=2,
            batch_size=50)

model_2.summary()
```

Save

In [ ]:
```
model_2.save("../src/model_v2_epoch30happyunhappy.hdf5")
```

Save

In [ ]:
```
model_2.summary()
```

Save

In [ ]:
```
scores = model_2.evaluate(x_test, y_test, verbose=2)
print("Accuracy: %.2f%%" % (scores[1]*100))
print('Test loss:', scores[0])
print('Test accuracy:', scores[1])
```

Save

In [ ]:
```
filepath2 = os.path.join("../src/model_v2_epoch30happyunhappy.hdf5")

checkpoint2 = ModelCheckpoint(
    filepath2,
    monitor='val_acc',
    verbose=1,
    save_best_only=True,
    mode='max')
```

Save

In [ ]:
```
src_dir = Path.cwd().parent/'src'
```

Save

In [83]:
```
#transfImag(demo_imgs, demo_imgs_faces)
```

Save

In [82]:
```
model_3 = base_model()

model_3.fit(X_train, y_train,
            validation_data=(X_val, y_val),
            epochs=12,
            verbose=1,
            batch_size=50)

model_3.summary()
model_3.save("../src/model_v3_epoch12happyunhappy.hdf5")
```

Save

```
Epoch 1/12
575/575 [==============================] - 1333s 2s/step - loss: 0.54
```

```
15 - accuracy: 0.7466 - val_loss: 0.4907 - val_accuracy: 0.7802
Epoch 2/12
575/575 [==============================] - 1451s 3s/step - loss: 0.39
63 - accuracy: 0.8266 - val_loss: 0.3725 - val_accuracy: 0.8370
Epoch 3/12
575/575 [==============================] - 1467s 3s/step - loss: 0.32
03 - accuracy: 0.8697 - val_loss: 0.3204 - val_accuracy: 0.8688
Epoch 4/12
575/575 [==============================] - 1375s 2s/step - loss: 0.28
39 - accuracy: 0.8862 - val_loss: 0.3127 - val_accuracy: 0.8716
Epoch 5/12
575/575 [==============================] - 1578s 3s/step - loss: 0.26
18 - accuracy: 0.8944 - val_loss: 0.3230 - val_accuracy: 0.8704
Epoch 6/12
575/575 [==============================] - 1520s 3s/step - loss: 0.24
73 - accuracy: 0.9019 - val_loss: 0.2718 - val_accuracy: 0.8927
Epoch 7/12
575/575 [==============================] - 1502s 3s/step - loss: 0.23
45 - accuracy: 0.9108 - val_loss: 0.3353 - val_accuracy: 0.8704
Epoch 8/12
575/575 [==============================] - 1561s 3s/step - loss: 0.22
84 - accuracy: 0.9106 - val_loss: 0.2586 - val_accuracy: 0.9005
Epoch 9/12
575/575 [==============================] - 1622s 3s/step - loss: 0.21
34 - accuracy: 0.9173 - val_loss: 0.3254 - val_accuracy: 0.8738
Epoch 10/12
575/575 [==============================] - 1497s 3s/step - loss: 0.20
66 - accuracy: 0.9200 - val_loss: 0.2577 - val_accuracy: 0.9042
Epoch 11/12
575/575 [==============================] - 1335s 2s/step - loss: 0.19
45 - accuracy: 0.9247 - val_loss: 0.2684 - val_accuracy: 0.9050
Epoch 12/12
575/575 [==============================] - 1325s 2s/step - loss: 0.18
78 - accuracy: 0.9295 - val_loss: 0.3093 - val_accuracy: 0.8919
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_30 (Conv2D)           (None, 48, 48, 64)        1664
_________________________________________________________________
conv2d_31 (Conv2D)           (None, 48, 48, 64)        102464
_________________________________________________________________
batch_normalization_20 (Batc (None, 48, 48, 64)        256
_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 24, 24, 64)        0
_________________________________________________________________
dropout_20 (Dropout)         (None, 24, 24, 64)        0
_________________________________________________________________
conv2d_32 (Conv2D)           (None, 24, 24, 128)       204928
_________________________________________________________________
conv2d_33 (Conv2D)           (None, 24, 24, 128)       409728
_________________________________________________________________
batch_normalization_21 (Batc (None, 24, 24, 128)       512
_________________________________________________________________
max_pooling2d_16 (MaxPooling (None, 12, 12, 128)       0
_________________________________________________________________
dropout_21 (Dropout)         (None, 12, 12, 128)       0
_________________________________________________________________
conv2d_34 (Conv2D)           (None, 12, 12, 256)       295168
_________________________________________________________________
conv2d_35 (Conv2D)           (None, 12, 12, 256)       590080
_________________________________________________________________
batch_normalization_22 (Batc (None, 12, 12, 256)       1024
```

```
max_pooling2d_17 (MaxPooling (None, 6, 6, 256)         0
_________________________________________________________________
dropout_22 (Dropout)         (None, 6, 6, 256)         0
_________________________________________________________________
flatten_5 (Flatten)          (None, 9216)              0
_________________________________________________________________
dense_11 (Dense)             (None, 128)               1179776
_________________________________________________________________
batch_normalization_23 (Batc (None, 128)               512
_________________________________________________________________
activation_10 (Activation)   (None, 128)               0
_________________________________________________________________
dropout_23 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_12 (Dense)             (None, 2)                 258
_________________________________________________________________
activation_11 (Activation)   (None, 2)                 0
=================================================================
Total params: 2,786,370
Trainable params: 2,785,218
Non-trainable params: 1,152
_________________________________________________________________
```

In [85]:
```
model_3.save("../src/model_v3.hdf5")
```

In [87]:
```
import json
model_json = model_3.to_json()
name_3 = 'model_v3.hdf5'
model_3.save_weights(name_3)
with open(name_3+'.json', "w") as json_file:
    json.dump(model_json, json_file)
```

In [88]:
```
model_3.load_weights("model_v3.hdf5")
```

In [91]:
```
scores = model_3.evaluate(X_test, y_test, verbose=2)
print("Accuracy: %.2f%%" % (scores[1]*100))
print('Test loss:', scores[0])
print('Test accuracy:', scores[1])
```

```
113/113 - 36s - loss: 0.2662 - accuracy: 0.9042
Accuracy: 90.42%
Test loss: 0.26619917154312134
Test accuracy: 0.9041515588760376
```

In [96]:
```
"%.2f%%" % (scores[1]*100)
```

Out[96]:
```
'90.42%'
```

In [8]:
```
scores = model_3.evaluate(X_test, y_test, verbose=2)
```

```
print("Accuracy: %.2f%%" % (scores[1]*100))
print('Test loss:', scores[0])
print('Test accuracy:', scores[1])
```

Save

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-8-d6e7b99e95fd> in <module>
----> 1 scores = model_3.evaluate(X_test, y_test, verbose=2)
      2 print("Accuracy: %.2f%%" % (scores[1]*100))
      3 print('Test loss:', scores[0])
      4 print('Test accuracy:', scores[1])

NameError: name 'model_3' is not defined
```

In [103…

```
accuracy = "%.2f%%" % (scores[1]*100)
test_loss = scores[0]
test_accuracy = scores[1]
```

Save

In [104…

```
print(accuracy,test_loss,test_accuracy)
```

Save

```
90.42% 0.26619917154312134 0.9041515588760376
```

In [101…

```
filepath='Checkpoint_{epoch:02d}_{val_accuracy:.2f}'
checkpoint = ModelCheckpoint(filepath, monitor='val_acc', verbose=1,
callbacks_list = [checkpoint]
```

Save

In [102…

```
filepath3 = os.path.join("../models/model_v2_.hdf5")

checkpoint3 = ModelCheckpoint(
    filepath3,
    monitor='val_acc',
    verbose=1,
    save_best_only=True,
    mode='max')
callbacks_list = [checkpoint3]
```

Save

In [ ]:

```
def transfImag2(path):
    print ('transforming image from {}'.format(path))

    input_img=cv2.imread(path)
    input_img=cv2.cvtColor(input_img, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(input_img, 1.25, 6)
    x,y,w,h = faces[0]
    img_data= input_img[y:y+h,x:x+w]
    img_data=cv2.resize(img_data,(48,48))

    img_data = np.stack(img_data)
```

```
    img_data = img_data / 255.0

    return img_data
```

Save

In [ ]:

```
PIC = transfImag2('foto.jpg') # transform pic
input_img=cv2.imread('foto.jpg') # get the array of the original pic

plt.subplot(121)
plt.imshow(input_img) # original pic
plt.subplot(122)
plt.imshow(Image.fromarray(PIC.squeeze()*255)) # transformed pic

PIC = np.expand_dims(PIC,axis=0).reshape(np.expand_dims(PIC,axis=0).s
print(PIC.shape)
pred2 = model_1.predict(PIC)[0]
print("Probs -> happy:{0:.5f} unhappy:{1:.5f}".format(pred2[0],pred2[

happy = pred2[0]
unhappy = pred2[1]
```

Save

In [ ]:

```
happy
```

Save

In [ ]:

```
if happy > 0.8:
    st.write("Llego mama")
    image = blavblabla
    st.image(image)
```

Save

In [106…

```
"""
Epoch 1/12
575/575 [==============================] - 1333s 2s/step - loss: 0.54
Epoch 2/12
575/575 [==============================] - 1451s 3s/step - loss: 0.39
Epoch 3/12
575/575 [==============================] - 1467s 3s/step - loss: 0.32
Epoch 4/12
575/575 [==============================] - 1375s 2s/step - loss: 0.28
Epoch 5/12
575/575 [==============================] - 1578s 3s/step - loss: 0.26
Epoch 6/12
575/575 [==============================] - 1520s 3s/step - loss: 0.24
Epoch 7/12
575/575 [==============================] - 1502s 3s/step - loss: 0.23
Epoch 8/12
575/575 [==============================] - 1561s 3s/step - loss: 0.22
Epoch 9/12
575/575 [==============================] - 1622s 3s/step - loss: 0.21
Epoch 10/12
575/575 [==============================] - 1497s 3s/step - loss: 0.20
Epoch 11/12
575/575 [==============================] - 1335s 2s/step - loss: 0.19
```

```
Epoch 12/12
575/575 [==============================] - 1325s 2s/step - loss: 0.18


Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_30 (Conv2D)           (None, 48, 48, 64)        1664
_________________________________________________________________
conv2d_31 (Conv2D)           (None, 48, 48, 64)        102464
_________________________________________________________________
batch_normalization_20 (Batc (None, 48, 48, 64)        256
_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 24, 24, 64)        0
_________________________________________________________________
dropout_20 (Dropout)         (None, 24, 24, 64)        0
_________________________________________________________________
conv2d_32 (Conv2D)           (None, 24, 24, 128)       204928
_________________________________________________________________
conv2d_33 (Conv2D)           (None, 24, 24, 128)       409728
_________________________________________________________________
batch_normalization_21 (Batc (None, 24, 24, 128)       512
_________________________________________________________________
max_pooling2d_16 (MaxPooling (None, 12, 12, 128)       0
_________________________________________________________________
dropout_21 (Dropout)         (None, 12, 12, 128)       0
_________________________________________________________________
conv2d_34 (Conv2D)           (None, 12, 12, 256)       295168
_________________________________________________________________
conv2d_35 (Conv2D)           (None, 12, 12, 256)       590080
_________________________________________________________________
batch_normalization_22 (Batc (None, 12, 12, 256)       1024
_________________________________________________________________
max_pooling2d_17 (MaxPooling (None, 6, 6, 256)         0
_________________________________________________________________
dropout_22 (Dropout)         (None, 6, 6, 256)         0
_________________________________________________________________
flatten_5 (Flatten)          (None, 9216)              0
_________________________________________________________________
dense_11 (Dense)             (None, 128)               1179776
_________________________________________________________________
batch_normalization_23 (Batc (None, 128)               512
_________________________________________________________________
activation_10 (Activation)   (None, 128)               0
_________________________________________________________________
dropout_23 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_12 (Dense)             (None, 2)                 258
_________________________________________________________________
activation_11 (Activation)   (None, 2)                 0
=================================================================

Total params: 2,786,370
Trainable params: 2,785,218
Non-trainable params: 1,152

"""
```

Save

Out[106…

```
'\nEpoch 1/12\n575/575 [==============================] - 1333s 2s/st
ep - loss: 0.5415 - accuracy: 0.7466 - val_loss: 0.4907 - val_accurac
y: 0.7802\nEpoch 2/12\n575/575 [==============================] - 145
1s 3s/step - loss: 0.3963 - accuracy: 0.8266 - val_loss: 0.3725 - val
_accuracy: 0.8370\nEpoch 3/12\n575/575 [==============================
=] - 1467s 3s/step - loss: 0.3203 - accuracy: 0.8697 - val_loss: 0.32
04 - val_accuracy: 0.8688\nEpoch 4/12\n575/575 [======================
=========] - 1375s 2s/step - loss: 0.2839 - accuracy: 0.8862 - val_lo
ss: 0.3127 - val_accuracy: 0.8716\nEpoch 5/12\n575/575 [=============
================] - 1578s 3s/step - loss: 0.2618 - accuracy: 0.8944
- val_loss: 0.3230 - val_accuracy: 0.8704\nEpoch 6/12\n575/575 [=====
=========================] - 1520s 3s/step - loss: 0.2473 - accuracy:
0.9019 - val_loss: 0.2718 - val_accuracy: 0.8927\nEpoch 7/12\n575/575
[==============================] - 1502s 3s/step - loss: 0.2345 - acc
uracy: 0.9108 - val_loss: 0.3353 - val_accuracy: 0.8704\nEpoch 8/12\n
575/575 [==============================] - 1561s 3s/step - loss: 0.22
84 - accuracy: 0.9106 - val_loss: 0.2586 - val_accuracy: 0.9005\nEpoc
h 9/12\n575/575 [==============================] - 1622s 3s/step - lo
ss: 0.2134 - accuracy: 0.9173 - val_loss: 0.3254 - val_accuracy: 0.87
38\nEpoch 10/12\n575/575 [==============================] - 1497s 3s/
step - loss: 0.2066 - accuracy: 0.9200 - val_loss: 0.2577 - val_accur
acy: 0.9042\nEpoch 11/12\n575/575 [==============================] -
1335s 2s/step - loss: 0.1945 - accuracy: 0.9247 - val_loss: 0.2684 -
val_accuracy: 0.9050\nEpoch 12/12\n575/575 [==========================
=====] - 1325s 2s/step - loss: 0.1878 - accuracy: 0.9295 - val_loss:
0.3093 - val_accuracy: 0.8919\n\n\nModel: "sequential_5"\n_____________
_________________________________________________\nLayer (type)
Output Shape              Param #   \n=================================
=================================\nconv2d_30 (Conv2D)           (Non
e, 48, 48, 64)        1664      \n_________________________________
_________________\nconv2d_31 (Conv2D)           (None, 4
8, 48, 64)        102464    \n_________________________________
_________________\nbatch_normalization_20 (Batc (None, 48, 4
8, 64)        256       \n_________________________________
_________________\nmax_pooling2d_15 (MaxPooling (None, 24, 24, 6
4)        0         \n_________________________________
_________________\ndropout_20 (Dropout)         (None, 24, 24, 64)
0         \n_________________________________
________\nconv2d_32 (Conv2D)           (None, 24, 24, 128)       2049
28    \n_________________________________
____\nconv2d_33 (Conv2D)           (None, 24, 24, 128)       409728
\n_____________________________________________________\n
batch_normalization_21 (Batc (None, 24, 24, 128)       512       \n__
_____________________________________________\nmax_
pooling2d_16 (MaxPooling (None, 12, 12, 128)       0         \n_____
_____________________________________\ndropout_
21 (Dropout)         (None, 12, 12, 128)       0         \n_________
_____________________________\nconv2d_34 (C
onv2D)           (None, 12, 12, 256)       295168    \n_____________
_____________\nconv2d_35 (Conv2
D)           (None, 12, 12, 256)       590080    \n_____________
_____\nbatch_normalization_
22 (Batc (None, 12, 12, 256)       1024      \n_____________
_____\nmax_pooling2d_17 (MaxPoo
ling (None, 6, 6, 256)         0         \n_____________
_____\ndropout_22 (Dropout)
(None, 6, 6, 256)         0         \n_____________
_____\nflatten_5 (Flatten)          (Non
e, 9216)              0         \n_____________
_____\ndense_11 (Dense)             (None, 1
```

```
28)                1179776   \n_____
_____\nbatch_normalization_23 (Batc (None, 128)
512       \n_____
________\nactivation_10 (Activation)   (None, 128)                0
\n_____\n
dropout_23 (Dropout)         (None, 128)                0         \n__
_____\ndens
e_12 (Dense)              (None, 2)                 258       \n_____
_____\nactivati
on_11 (Activation)   (None, 2)                 0         \n==========
==========\n\nTotal para
ms: 2,786,370\nTrainable params: 2,785,218\nNon-trainable params: 1,1
52\n\n'
```

In [ ]:

```
# HAAR CASCADE CLASSIFIER
def detect_face_eyes_smile(pth,new_pth):
    """
    Extracts all .jpg files from local path,
    calls on haar cascade classifiers (frontalface, eyes and smile)
    and draws detection rectangles on each .jpg

    Takes: local path of directory with .jpg images

    Returns: individual windows for .jpg files with detection rectang
    """

    counter_imgs = 0
    counter_faces = 0
    counter_smiles = 0
    counter_eyes = 0
    face_cascade = cv2.CascadeClassifier('../src/haarcascade_frontalf
    eye_cascade = cv2.CascadeClassifier('../src/haarcascade_eye.xml')
    smile_cascade = cv2.CascadeClassifier('../src/haarcascade_smile.x

    for file in sorted(pth.iterdir()):
        if file.suffix != '.jpg':
            pass
        else:
            counter_imgs += 1
            print(file.name)

            img = cv2.imread(str(file))
            img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

            #plt.imshow(img)

            # FRONTAL FACE

            faces = face_cascade.detectMultiScale(
                img_gray,
                scaleFactor=1.06,
                minNeighbors=7,
                minSize=(30, 30),
                flags=cv2.CASCADE_SCALE_IMAGE)
            if faces is None:
                print("No Face Found")

            for (fx,fy,fw,fh) in faces:

                counter_faces += 1
```

```
roi_gray = img_gray [fy:fy+fh, fx:fx+fw] # region of
roi_gray2 = cv2.resize (roi_gray, (48,48))
roi_gray3 = np.stack(roi_gray2)
roi_gray4 = roi_gray2 / 255

roi_color = img[fy:fy+fh, fx:fx+fw] # region of inter
roi_color2 = cv2.resize (roi_color, (48,48))
roi_color3 = np.stack(roi_color2)
roi_color4 = roi_color2 / 255

cv2.rectangle(
    img,
    (fx,fy),
    (fx+fw,fy+fh),
    #(127,0,255),
    (0,255,0),
    2)

# SMILES

smiles = smile_cascade.detectMultiScale(
    roi_gray,
    scaleFactor = 1.35,
    minNeighbors = 8)

for (sx, sy, sw, sh) in smiles:
    counter_smiles += 1
    cv2.rectangle(
        roi_color,
        (sx, sy),
        (sx + sw, sy + sh),
        #(255, 0, 130),
        #(0,220,80),
        (127,0,255),
        1)

# EYES

eyes = eye_cascade.detectMultiScale(
    roi_gray,
    scaleFactor=1.05,
    minNeighbors = 6)

for (ex,ey,ew,eh) in eyes:
    counter_eyes += 1
    cv2.rectangle(
        roi_color,
        (ex , ey),
        (ex + ew, ey + eh),
        (0,255,255),
        1)

# save images with detected regions
file_to_save = file.name.replace(".",f"_face{counter_

#cv2.imwrite(str(pth.parent/'demo_faces'/file_to_save
cv2.imwrite(str(new_pth/file_to_save),roi_color)
counter_imgs = 0
counter_faces = 0
```

```
            # show the output frame
            cv2.imshow(f"img{file_to_save}", img)
            key = cv2.waitKey(100) & 0xFF

    cv2.destroyAllWindows(f"img{file_to_save}")




    """
        # if the `q` key was pressed, break from the loop
            if key == ord("q"):
                # do a bit of cleanup
                cv2.destroyAllWindows()
                break

    # do a bit of cleanup
    cv2.destroyAllWindows()
cv2.destroyAllWindows()


        """
```

Save

In [ ]:
```
demo_imgs = Path.cwd().parent/'demo'
demo_imgs_faces = Path.cwd().parent/demo2
demo1 = demo_imgs/'A_0.jpg'
demo2 = demo_imgs/'demooo_01.jpg'
```

Save

In [ ]:
```
face_cascade = cv2.CascadeClassifier('../src/haarcascade_frontalface_
eye_cascade = cv2.CascadeClassifier('../src/haarcascade_eye.xml')
smile_cascade = cv2.CascadeClassifier('../src/haarcascade_smile.xml')
```

Save

In [89]:
```
filepath='Checkpoint_{epoch:02d}_{val_accuracy:.2f}'
```

Save

In [90]:
```
%.2f%%
```

Save

```
UsageError: Line magic function `%.2f%%` not found.
```

In [ ]:

Save

In [ ]:
```
cv2.imwrite(str(pth.parent/*'_haar'/file.name),roi_color)
```

Save

In [ ]:
```
str(pth.parent/'*_haar'/file.name)
```

Save

In [ ]:
```

```

Save