Trần Ngọc Đoàn - 19146175

https://github.com/DoanAI/Face.git

```
import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.layers import Flatten, Dense, Dropout, Activation
from google.colab import drive
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/cor
```

```
drive.mount('/content/drive',force_remount=True)
from tensorflow.keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True)
training_set=train_datagen.flow_from_directory('/content/drive/MyDrive/CNN/Face/training_set',
                                               target_size=(256,256),
                                               batch_size=32,
                                               class_mode ='categorical')
test_set=train_datagen.flow_from_directory('/content/drive/MyDrive/CNN/Face/test_set',
                                               target_size=(256,256),
                                               batch_size=32,
                                               class_mode ='categorical')
```

```
Mounted at /content/drive
Found 80 images belonging to 5 classes.
Found 10 images belonging to 5 classes.
```

```
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/cor
```

```
model=Sequential()
model.add(Conv2D(128,(3,3),activation='relu',kernel_initializer='he_uniform',padding='same',input_sh
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(64,(3,3),activation='relu',kernel_initializer='he_uniform',padding='same'))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(32,(3,3),activation='relu',kernel_initializer='he_uniform',padding='same'))
model.add(MaxPooling2D((2,2)))
model.add(Flatten())
model.add(Dense(128,activation='relu',kernel_initializer = 'he_uniform'))
model.add(Dense(5,activation='Softmax'))
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.callbacks import EarlyStopping
model.compile(optimizer = 'adam', loss ='categorical_crossentropy',metrics = ['accuracy'])
callbacks=[EarlyStopping(monitor='val_loss',patience=100)]
```

```
history=model.fit(training_set,
                  steps_per_epoch=len(training_set),
                  batch_size = 64,
                  epochs=150,
                  validation_data=test_set,
                  validation_steps=len(test_set),
                  callbacks=callbacks,
                  verbose = 1)
```

```
Epoch 85/150
3/3 [==============================] - 7s 2s/step - loss: 0.6882 - accuracy: 0.6375 - val_lo
Epoch 86/150
3/3 [==============================] - 7s 2s/step - loss: 0.7859 - accuracy: 0.5750 - val_lo
Epoch 87/150
3/3 [==============================] - 7s 3s/step - loss: 0.7778 - accuracy: 0.6625 - val_lo
Epoch 88/150
3/3 [==============================] - 7s 2s/step - loss: 0.7451 - accuracy: 0.6625 - val_lo
Epoch 89/150
3/3 [==============================] - 7s 2s/step - loss: 0.7367 - accuracy: 0.6375 - val_lo
Epoch 90/150
3/3 [==============================] - 7s 2s/step - loss: 0.7614 - accuracy: 0.7000 - val_lo
Epoch 91/150
3/3 [==============================] - 7s 2s/step - loss: 0.6734 - accuracy: 0.6625 - val_lo
Epoch 92/150
3/3 [==============================] - 7s 2s/step - loss: 0.8129 - accuracy: 0.6250 - val_lo
Epoch 93/150
3/3 [==============================] - 7s 2s/step - loss: 0.6900 - accuracy: 0.7000 - val_lo
Epoch 94/150
3/3 [==============================] - 7s 2s/step - loss: 0.8081 - accuracy: 0.5875 - val_lo
Epoch 95/150
3/3 [==============================] - 7s 2s/step - loss: 0.7571 - accuracy: 0.6625 - val_lo
Epoch 96/150
3/3 [==============================] - 7s 2s/step - loss: 0.7115 - accuracy: 0.7000 - val_lo
Epoch 97/150
3/3 [==============================] - 7s 2s/step - loss: 0.6695 - accuracy: 0.6875 - val_lo
Epoch 98/150
3/3 [==============================] - 7s 3s/step - loss: 0.6475 - accuracy: 0.6875 - val_lo
Epoch 99/150
3/3 [==============================] - 7s 3s/step - loss: 0.5842 - accuracy: 0.7000 - val_lo
Epoch 100/150
3/3 [==============================] - 7s 2s/step - loss: 0.7123 - accuracy: 0.6250 - val_lo
Epoch 101/150
3/3 [==============================] - 7s 2s/step - loss: 0.6524 - accuracy: 0.7000 - val_lo
Epoch 102/150
3/3 [==============================] - 7s 2s/step - loss: 0.6445 - accuracy: 0.6625 - val_lo
Epoch 103/150
3/3 [==============================] - 7s 3s/step - loss: 0.7554 - accuracy: 0.6250 - val_lo
Epoch 104/150
3/3 [==============================] - 7s 2s/step - loss: 0.6163 - accuracy: 0.7250 - val_lo
Epoch 105/150
3/3 [==============================] - 7s 3s/step - loss: 0.6410 - accuracy: 0.7125 - val_lo
Epoch 106/150
3/3 [==============================] - 7s 2s/step - loss: 0.6780 - accuracy: 0.6750 - val_lo
Epoch 107/150
3/3 [==============================] - 7s 2s/step - loss: 0.5858 - accuracy: 0.7125 - val_lo
Epoch 108/150
3/3 [==============================] - 7s 2s/step - loss: 0.6869 - accuracy: 0.6000 - val_lo
Epoch 109/150
3/3 [==============================] - 7s 2s/step - loss: 0.6082 - accuracy: 0.6875 - val_lo
Epoch 110/150
3/3 [==============================] - 7s 2s/step - loss: 0.5752 - accuracy: 0.7500 - val_lo
Epoch 111/150
3/3 [==============================] - 7s 2s/step - loss: 0.5923 - accuracy: 0.6750 - val_lo
```

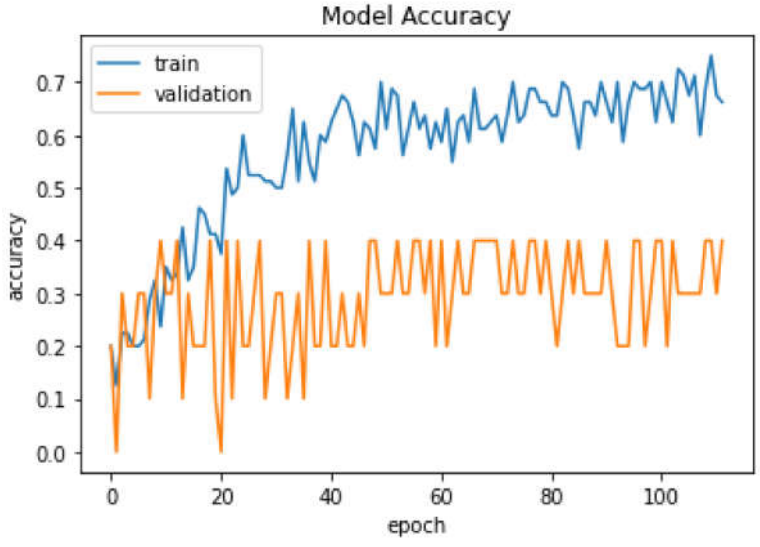
```
Epoch 112/150
3/3 [==============================] - 7s 3s/step - loss: 0.6784 - accuracy: 0.6625 - val_lo
```

```
score = model.evaluate(test_set,verbose=0)
print('Sai số kiểm tra là: ',score[0])
print('Độ chính xác kiểm tra là: ',score[1])
```

```
Sai số kiểm tra là:  3.5323104858398438
Độ chính xác kiểm tra là:  0.4000000059604645
```

```
import matplotlib.pyplot as plt
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train','validation'], loc='upper-left')
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: MatplotlibDeprecationWarning: U
        best
        upper right
        upper left
        lower left
        lower right
        right
        center left
        center right
        lower center
        upper center
        center
This will raise an exception in 3.3.
  import sys
<matplotlib.legend.Legend at 0x7fa1600e0050>
```




```
model.save('model_face.h5')
from tensorflow.keras.models import load_model
model=load_model('model_face.h5')
```

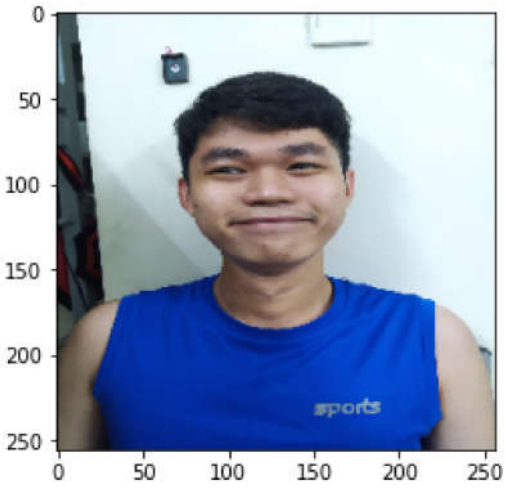
```
from tensorflow.keras.utils import load_img
from tensorflow.keras.utils import img_to_array
import numpy as np
```

```
img_0 = load_img('/content/drive/MyDrive/CNN/Face/prediction/Doan.jpg', target_size=(256,256))
img_1 = load_img('/content/drive/MyDrive/CNN/Face/prediction/Doann.jpg', target_size=(256,256))

plt.imshow(img_0)
imga = img_to_array(img_0)
imga = imga/255
imga = np.expand_dims(imga,axis=0)
result = model.predict(imga)

if round(result[0][0]+1)==1:
  prediction = "It's me"
elif round(result[0][1])==1:
  prediction = "It's me"
print(prediction)
```

```
It's me
```




```
img_0 = load_img('/content/drive/MyDrive/CNN/Face/prediction/Doan.jpg', target_size=(256,256))
img_1 = load_img('/content/drive/MyDrive/CNN/Face/prediction/Doann.jpg', target_size=(256,256))

plt.imshow(img_1)
imga = img_to_array(img_1)
imga = imga/255
imga = np.expand_dims(imga,axis=0)
result = model.predict(imga)

if round(result[0][0]+1)==1:
  prediction = "It's me"
elif round(result[0][1])==1:
  prediction = "It's me"
```

```
It's me
```