FACE ID GROUP

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing import image
from tensorflow.keras.optimizers import SGD, RMSprop
from tensorflow.keras.utils import to_categorical
import matplotlib.pyplot as plt
from tensorflow.keras.models import load_model
import tensorflow as  tf
import numpy as np
import cv2
import os
from keras.utils import np_utils
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPlateau
from keras.models import Sequential
from keras.layers import Dense,Flatten, Dropout
from tensorflow.keras.utils import load_img
from tensorflow.keras.utils import img_to_array
from keras.layers import Conv2D, MaxPooling2D
```

```
train = ImageDataGenerator(rescale=1/255)
validation = ImageDataGenerator(rescale=1/255)
```

```
train_set=train.flow_from_directory('/content/drive/MyDrive/AI(FG)/train',target_size = (150,
test_set=validation.flow_from_directory('/content/drive/MyDrive/AI(FG)/validation',target_siz
```

```
     Found 125 images belonging to 3 classes.
     Found 25 images belonging to 3 classes.
```

```
train_set.class_indices
```

```
     {'Luong': 0, 'Nam': 1, 'Trong': 2}
```

```
model = tf.keras.models.Sequential(
    [ tf.keras.layers.Conv2D(16,(3,3),activation = 'relu',padding='same',input_shape =(150,15
      tf.keras.layers.Conv2D(16,(3,3),activation = 'relu',padding='same'),
      tf.keras.layers.MaxPool2D(2,2),

      tf.keras.layers.Conv2D(32,(3,3),activation = 'relu',padding='same'),
      tf.keras.layers.Conv2D(32,(3,3),activation = 'relu',padding='same'),
      tf.keras.layers.MaxPool2D(2,2),

      tf.keras.layers.Conv2D(64,(3,3),activation = 'relu',padding='same'),
      tf.keras.layers.Conv2D(64,(3,3),activation = 'relu',padding='same'),
      tf.keras.layers.MaxPool2D(2,2),
      tf.keras.layers.Conv2D(128,(3,3),activation = 'relu',padding='same'),
      tf.keras.layers.Conv2D(128,(3,3),activation = 'relu',padding='same'),
      tf.keras.layers.MaxPool2D(2,2),

      tf.keras.layers.Flatten(),
      tf.keras.layers.Dense(256,activation = 'relu'),
      tf.keras.layers.Dense(3,activation='softmax')])


opt = SGD(lr=0.001, momentum=0.9)
model.compile(optimizer=opt, loss='categorical_crossentropy', metrics = ['accuracy'])
```

```
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descent.py:102: UserW
  super(SGD, self).__init__(name, **kwargs)
```

```
history = model.fit(train_set,batch_size=12,epochs=20,verbose=1,validation_data=test_set)
```

```
Epoch 1/20
11/11 [==============================] - 7s 621ms/step - loss: 1.0923 - accuracy: 0.568
Epoch 2/20
11/11 [==============================] - 6s 534ms/step - loss: 1.0653 - accuracy: 0.800
Epoch 3/20
11/11 [==============================] - 6s 551ms/step - loss: 1.0183 - accuracy: 0.800
Epoch 4/20
11/11 [==============================] - 6s 579ms/step - loss: 0.8875 - accuracy: 0.800
Epoch 5/20
11/11 [==============================] - 6s 546ms/step - loss: 0.5537 - accuracy: 0.800
Epoch 6/20
11/11 [==============================] - 6s 567ms/step - loss: 0.5189 - accuracy: 0.784
Epoch 7/20
11/11 [==============================] - 6s 526ms/step - loss: 0.3205 - accuracy: 0.864
Epoch 8/20
11/11 [==============================] - 6s 544ms/step - loss: 0.2529 - accuracy: 0.904
Epoch 9/20
11/11 [==============================] - 6s 545ms/step - loss: 0.1358 - accuracy: 0.968
Epoch 10/20
11/11 [==============================] - 6s 541ms/step - loss: 0.0673 - accuracy: 0.992
Epoch 11/20
11/11 [==============================] - 6s 555ms/step - loss: 0.1840 - accuracy: 0.936
Epoch 12/20
11/11 [==============================] - 6s 549ms/step - loss: 0.1523 - accuracy: 0.936
Epoch 13/20
11/11 [==============================] - 7s 609ms/step - loss: 0.2477 - accuracy: 0.896
Epoch 14/20
11/11 [==============================] - 6s 555ms/step - loss: 0.1330 - accuracy: 0.968
Epoch 15/20
11/11 [==============================] - 6s 542ms/step - loss: 0.0770 - accuracy: 0.976
Epoch 16/20
11/11 [==============================] - 6s 562ms/step - loss: 0.0434 - accuracy: 0.976
Epoch 17/20
11/11 [==============================] - 6s 541ms/step - loss: 0.0257 - accuracy: 0.992
```

```
Epoch 18/20
11/11 [==============================] - 6s 528ms/step - loss: 0.0232 - accuracy: 0.984
Epoch 19/20
11/11 [==============================] - 6s 524ms/step - loss: 0.0342 - accuracy: 0.992
Epoch 20/20
11/11 [==============================] - 6s 552ms/step - loss: 0.1595 - accuracy: 0.920
```

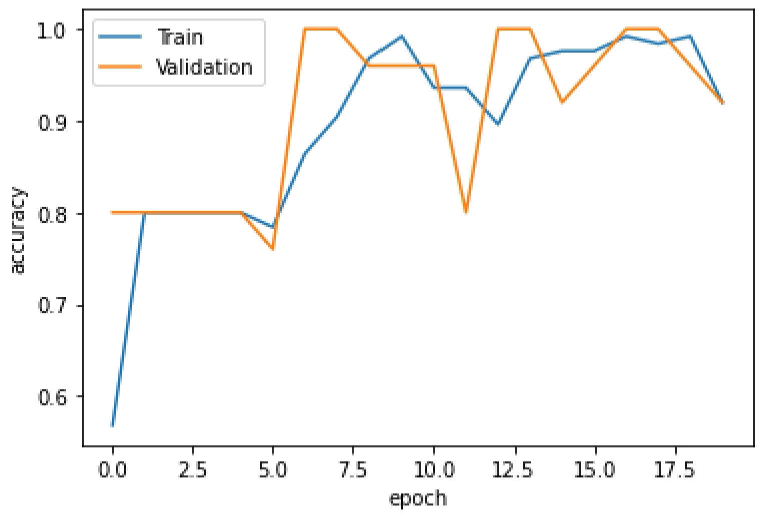
```
model.save('./content/drive/MyDrive/AI_EXAM/FaceGr.h5')
```

```
model=load_model('./content/drive/MyDrive/AI_EXAM/FaceGr.h5')
```

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

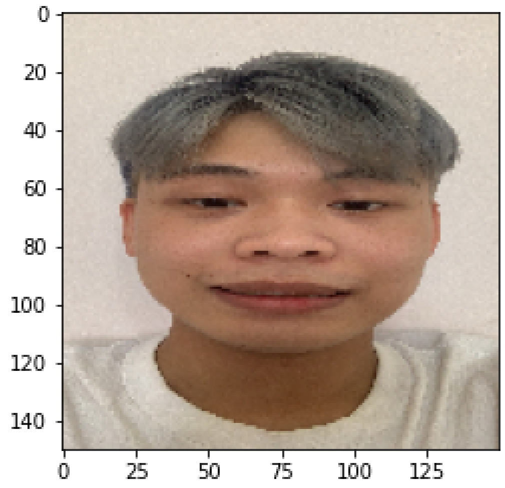
```
3/3 [==============================] - 1s 162ms/step - loss: 0.2363 - accuracy: 0.9200
Sai số:  0.2363031506538391
Độ chính xác:  0.9200000166893005
```

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['Train','Validation'],loc='upper left')
plt.show()
```



```
test_img=load_img('/content/drive/MyDrive/AI(FG)/Luong/NTNA8278.JPG',target_size=(150,150))
plt.imshow(test_img)
test_img= img_to_array(test_img)
test_img=test_img/255
test_img=np.expand_dims(test_img,axis=0)
result=model.predict(test_img)
if round(result[0][0])==1:
    prediction="LUONG"
elif round(result[0][1])==1:
    prediction="NAM"
elif round(result[0][2])==1:
    prediction="TRONG"
print(prediction)
```

```
  LUONG
```



```
test_img=load_img('/content/drive/MyDrive/AI(FG)/Nam/11.jpg',target_size=(150,150))
plt.imshow(test_img)
test_img= img_to_array(test_img)
test_img=test_img/255
test_img=np.expand_dims(test_img,axis=0)
result=model.predict(test_img)
if round(result[0][0])==1:
    prediction="LUONG"
elif round(result[0][1])==1:
    prediction="NAM"
elif round(result[0][2])==1:
    prediction="TRONG"
print(prediction)
```

```
     NAM
```

```
test_img=load_img('/content/drive/MyDrive/AI(FG)/Trong/0.jpg',target_size=(150,150))
plt.imshow(test_img)
test_img= img_to_array(test_img)
test_img=test_img/255
test_img=np.expand_dims(test_img,axis=0)
result=model.predict(test_img)
if round(result[0][0])==1:
   prediction="LUONG"
elif round(result[0][1])==1:
   prediction="NAM"
elif round(result[0][2])==1:
   prediction="TRONG"
print(prediction)
```

```
     TRONG
```