baitapAI_ANN

May 13, 2022

1 INFORMATIONS

Nguyễn Nhật Tiến

mssv: 19146273

Lớp tối thứ sáu

LINK COLAB: https://bitly.com.vn/cvoc76

LINK GITHUB: https://bitly.com.vn/m9qf0m

LINK DATASETS FACE ID: https://bitly.com.vn/joaxlp

2 cifar10

[ ]:
```
from keras.datasets import cifar10
from matplotlib import pyplot as plt
import numpy as np
from tensorflow.keras.utils import to_categorical
from keras.models import Sequential
from keras.utils import np_utils
from keras.layers import Dense,Activation,Dropout,LSTM,BatchNormalization
from keras.layers import Flatten
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.utils import load_img,img_to_array
from tensorflow.keras.models import load_model
(x_train,y_train),(x_test,y_test)=cifar10.load_data()
```

[ ]:
```
classes=["airplane","automobile","bird","cat","deer","dog","frog","horse","ship","truck"]
```

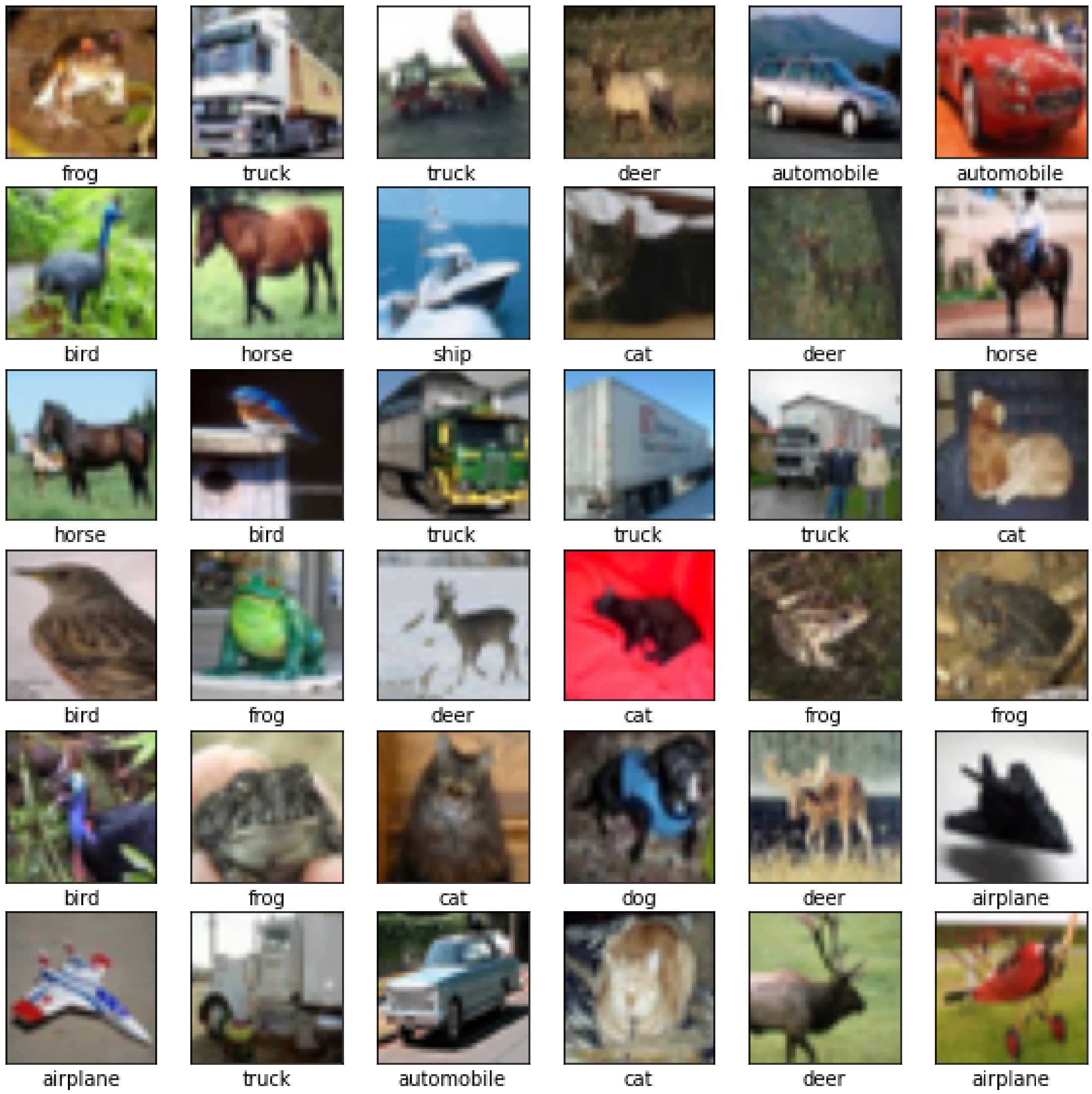
[ ]:
```
x_train=x_train.astype('float32')
x_test=x_test.astype('float32')
x_train/=255
x_test/=255
y_train=to_categorical(y_train,15)
y_test=to_categorical(y_test,15)
```

[ ]:
```python
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(50000, 32, 32, 3)
(10000, 32, 32, 3)
(50000, 15)
(10000, 15)
```

[ ]:
```python
plt.figure(figsize=(10,10))
for i in range(36):
  plt.subplot(6,6,i+1)
  plt.xticks([])
  plt.yticks([])
  plt.imshow(x_train[i])
  plt.xlabel(classes[np.argmax(y_train[i])])
plt.show
```

[ ]:
```
<function matplotlib.pyplot.show>
```



[ ]:
```python
model = Sequential()
model.add(Flatten(input_shape=(32,32,3)))
model.add(Dense(784,activation='relu'))
model.add(Dense(784,activation='relu'))
model.add(Dense(15,activation='Softmax'))
model.summary()
```

```
Model: "sequential_16"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_10 (Flatten)        (None, 3072)              0
```

```
 dense_48 (Dense)            (None, 784)              2409232

 dense_49 (Dense)            (None, 784)              615440

 dense_50 (Dense)            (None, 15)               11775

=================================================================
Total params: 3,036,447
Trainable params: 3,036,447
Non-trainable params: 0
_________________________________________________________________
```

[ ]:
```
opt=SGD(learning_rate=0.01,momentum=0.9)
model.
 ↪compile(loss='categorical_crossentropy',optimizer=opt,metrics=['accuracy'])
history=model.
 ↪fit(x_train,y_train,batch_size=128,epochs=50,verbose=1,validation_data=(x_test,y_test))
```

```
Epoch 1/50
391/391 [==============================] - 17s 43ms/step - loss: 1.8354 -
accuracy: 0.3444 - val_loss: 1.6491 - val_accuracy: 0.4053
Epoch 2/50
391/391 [==============================] - 17s 44ms/step - loss: 1.6344 -
accuracy: 0.4210 - val_loss: 1.6157 - val_accuracy: 0.4217
Epoch 3/50
391/391 [==============================] - 17s 43ms/step - loss: 1.5444 -
accuracy: 0.4529 - val_loss: 1.5256 - val_accuracy: 0.4520
Epoch 4/50
391/391 [==============================] - 17s 42ms/step - loss: 1.4919 -
accuracy: 0.4701 - val_loss: 1.5284 - val_accuracy: 0.4641
Epoch 5/50
391/391 [==============================] - 17s 45ms/step - loss: 1.4419 -
accuracy: 0.4875 - val_loss: 1.5335 - val_accuracy: 0.4612
Epoch 6/50
391/391 [==============================] - 16s 42ms/step - loss: 1.4023 -
accuracy: 0.5009 - val_loss: 1.4270 - val_accuracy: 0.4868
Epoch 7/50
391/391 [==============================] - 17s 45ms/step - loss: 1.3593 -
accuracy: 0.5180 - val_loss: 1.4328 - val_accuracy: 0.4847
Epoch 8/50
391/391 [==============================] - 17s 44ms/step - loss: 1.3219 -
accuracy: 0.5306 - val_loss: 1.3897 - val_accuracy: 0.5015
Epoch 9/50
391/391 [==============================] - 16s 42ms/step - loss: 1.2948 -
accuracy: 0.5428 - val_loss: 1.3533 - val_accuracy: 0.5193
Epoch 10/50
391/391 [==============================] - 16s 42ms/step - loss: 1.2637 -
accuracy: 0.5519 - val_loss: 1.3738 - val_accuracy: 0.5185
```

```
Epoch 11/50
391/391 [==============================] - 16s 42ms/step - loss: 1.2306 -
accuracy: 0.5654 - val_loss: 1.3422 - val_accuracy: 0.5310
Epoch 12/50
391/391 [==============================] - 16s 42ms/step - loss: 1.2022 -
accuracy: 0.5736 - val_loss: 1.4047 - val_accuracy: 0.5067
Epoch 13/50
391/391 [==============================] - 17s 44ms/step - loss: 1.1746 -
accuracy: 0.5838 - val_loss: 1.3329 - val_accuracy: 0.5312
Epoch 14/50
391/391 [==============================] - 17s 42ms/step - loss: 1.1420 -
accuracy: 0.5978 - val_loss: 1.3641 - val_accuracy: 0.5246
Epoch 15/50
391/391 [==============================] - 16s 42ms/step - loss: 1.1186 -
accuracy: 0.6033 - val_loss: 1.3532 - val_accuracy: 0.5222
Epoch 16/50
391/391 [==============================] - 16s 42ms/step - loss: 1.0875 -
accuracy: 0.6160 - val_loss: 1.3294 - val_accuracy: 0.5332
Epoch 17/50
391/391 [==============================] - 17s 42ms/step - loss: 1.0570 -
accuracy: 0.6258 - val_loss: 1.3235 - val_accuracy: 0.5436
Epoch 18/50
391/391 [==============================] - 17s 43ms/step - loss: 1.0335 -
accuracy: 0.6330 - val_loss: 1.3300 - val_accuracy: 0.5364
Epoch 19/50
391/391 [==============================] - 17s 43ms/step - loss: 1.0030 -
accuracy: 0.6446 - val_loss: 1.3618 - val_accuracy: 0.5353
Epoch 20/50
391/391 [==============================] - 17s 44ms/step - loss: 0.9743 -
accuracy: 0.6549 - val_loss: 1.3324 - val_accuracy: 0.5377
Epoch 21/50
391/391 [==============================] - 17s 43ms/step - loss: 0.9438 -
accuracy: 0.6628 - val_loss: 1.3401 - val_accuracy: 0.5441
Epoch 22/50
391/391 [==============================] - 17s 42ms/step - loss: 0.9182 -
accuracy: 0.6750 - val_loss: 1.3812 - val_accuracy: 0.5297
Epoch 23/50
391/391 [==============================] - 17s 42ms/step - loss: 0.8930 -
accuracy: 0.6860 - val_loss: 1.3461 - val_accuracy: 0.5445
Epoch 24/50
391/391 [==============================] - 16s 42ms/step - loss: 0.8649 -
accuracy: 0.6950 - val_loss: 1.3563 - val_accuracy: 0.5392
Epoch 25/50
391/391 [==============================] - 17s 43ms/step - loss: 0.8385 -
accuracy: 0.7013 - val_loss: 1.3778 - val_accuracy: 0.5512
Epoch 26/50
391/391 [==============================] - 16s 42ms/step - loss: 0.8115 -
accuracy: 0.7136 - val_loss: 1.3798 - val_accuracy: 0.5481
```

```
Epoch 27/50
391/391 [==============================] - 16s 42ms/step - loss: 0.7905 -
accuracy: 0.7206 - val_loss: 1.3953 - val_accuracy: 0.5461
Epoch 28/50
391/391 [==============================] - 17s 42ms/step - loss: 0.7563 -
accuracy: 0.7324 - val_loss: 1.4382 - val_accuracy: 0.5414
Epoch 29/50
391/391 [==============================] - 17s 42ms/step - loss: 0.7359 -
accuracy: 0.7400 - val_loss: 1.4538 - val_accuracy: 0.5405
Epoch 30/50
391/391 [==============================] - 17s 42ms/step - loss: 0.6995 -
accuracy: 0.7544 - val_loss: 1.5000 - val_accuracy: 0.5388
Epoch 31/50
391/391 [==============================] - 17s 42ms/step - loss: 0.6763 -
accuracy: 0.7633 - val_loss: 1.4272 - val_accuracy: 0.5538
Epoch 32/50
391/391 [==============================] - 16s 42ms/step - loss: 0.6474 -
accuracy: 0.7732 - val_loss: 1.4773 - val_accuracy: 0.5492
Epoch 33/50
391/391 [==============================] - 16s 42ms/step - loss: 0.6258 -
accuracy: 0.7800 - val_loss: 1.5519 - val_accuracy: 0.5437
Epoch 34/50
391/391 [==============================] - 16s 42ms/step - loss: 0.6107 -
accuracy: 0.7854 - val_loss: 1.5228 - val_accuracy: 0.5461
Epoch 35/50
391/391 [==============================] - 16s 42ms/step - loss: 0.5626 -
accuracy: 0.8016 - val_loss: 1.5576 - val_accuracy: 0.5515
Epoch 36/50
391/391 [==============================] - 17s 43ms/step - loss: 0.5445 -
accuracy: 0.8102 - val_loss: 1.5456 - val_accuracy: 0.5521
Epoch 37/50
391/391 [==============================] - 17s 42ms/step - loss: 0.5231 -
accuracy: 0.8175 - val_loss: 1.5968 - val_accuracy: 0.5490
Epoch 38/50
391/391 [==============================] - 16s 42ms/step - loss: 0.4887 -
accuracy: 0.8295 - val_loss: 1.6446 - val_accuracy: 0.5400
Epoch 39/50
391/391 [==============================] - 16s 42ms/step - loss: 0.4749 -
accuracy: 0.8340 - val_loss: 1.6660 - val_accuracy: 0.5485
Epoch 40/50
391/391 [==============================] - 16s 42ms/step - loss: 0.4550 -
accuracy: 0.8413 - val_loss: 1.7007 - val_accuracy: 0.5494
Epoch 41/50
391/391 [==============================] - 17s 43ms/step - loss: 0.4212 -
accuracy: 0.8536 - val_loss: 1.7064 - val_accuracy: 0.5464
Epoch 42/50
391/391 [==============================] - 16s 42ms/step - loss: 0.4005 -
accuracy: 0.8624 - val_loss: 1.7476 - val_accuracy: 0.5460
```

```
Epoch 43/50
391/391 [==============================] - 17s 43ms/step - loss: 0.3884 -
accuracy: 0.8638 - val_loss: 1.7794 - val_accuracy: 0.5461
Epoch 44/50
391/391 [==============================] - 17s 43ms/step - loss: 0.3649 -
accuracy: 0.8751 - val_loss: 1.8602 - val_accuracy: 0.5497
Epoch 45/50
391/391 [==============================] - 19s 49ms/step - loss: 0.3429 -
accuracy: 0.8818 - val_loss: 1.8667 - val_accuracy: 0.5487
Epoch 46/50
391/391 [==============================] - 16s 42ms/step - loss: 0.3342 -
accuracy: 0.8850 - val_loss: 1.8800 - val_accuracy: 0.5422
Epoch 47/50
391/391 [==============================] - 17s 43ms/step - loss: 0.3025 -
accuracy: 0.8963 - val_loss: 2.0181 - val_accuracy: 0.5402
Epoch 48/50
391/391 [==============================] - 17s 43ms/step - loss: 0.2803 -
accuracy: 0.9038 - val_loss: 1.9699 - val_accuracy: 0.5432
Epoch 49/50
391/391 [==============================] - 16s 42ms/step - loss: 0.2797 -
accuracy: 0.9041 - val_loss: 2.0353 - val_accuracy: 0.5410
Epoch 50/50
391/391 [==============================] - 16s 42ms/step - loss: 0.2551 -
accuracy: 0.9142 - val_loss: 2.0382 - val_accuracy: 0.5499
```

[ ]:
```
model.save('/content/drive/MyDrive/dulieuAI/cifar10/train_cifar10.h5')
```

[ ]:
```
#so 0 may bay , 1 xe hoi ,2 chim,3 meo ,4 nai,5 cho, 6 ech, 7 ngua, 8 tau thuy,␣
↪9 xe tai
img=load_img('meo1.png',target_size=(32,32))
img=img_to_array(img)
img=img.reshape(1,32,32,3)
img=img.astype('float32')
img=img/255
np.argmax(model.predict(img),axis=-1)
```

[ ]:
```
array([3])
```

[ ]:
```
score=model.evaluate(x_test,y_test,verbose=1)
print("do chinh xac = ",score[1])
```

```
313/313 [==============================] - 3s 9ms/step - loss: 2.0382 -
accuracy: 0.5499
do chinh xac =  0.5498999953269958
```

3 cifar100

[ ]:
```
from keras.datasets import cifar10
from matplotlib import pyplot as plt
import numpy as np
from tensorflow.keras.utils import to_categorical
from keras.models import Sequential
from keras.utils import np_utils
from keras.layers import Dense,Activation,Dropout,LSTM,BatchNormalization
from keras.layers import Flatten
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import load_img,img_to_array
from tensorflow.keras.models import load_model
from keras.datasets import cifar100
```

[ ]:
```
(x_train,y_train),(x_test,y_test)=cifar100.load_data()
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-100-python.tar.gz
169009152/169001437 [==============================] - 3s 0us/step
169017344/169001437 [==============================] - 3s 0us/step
```

[ ]:
```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

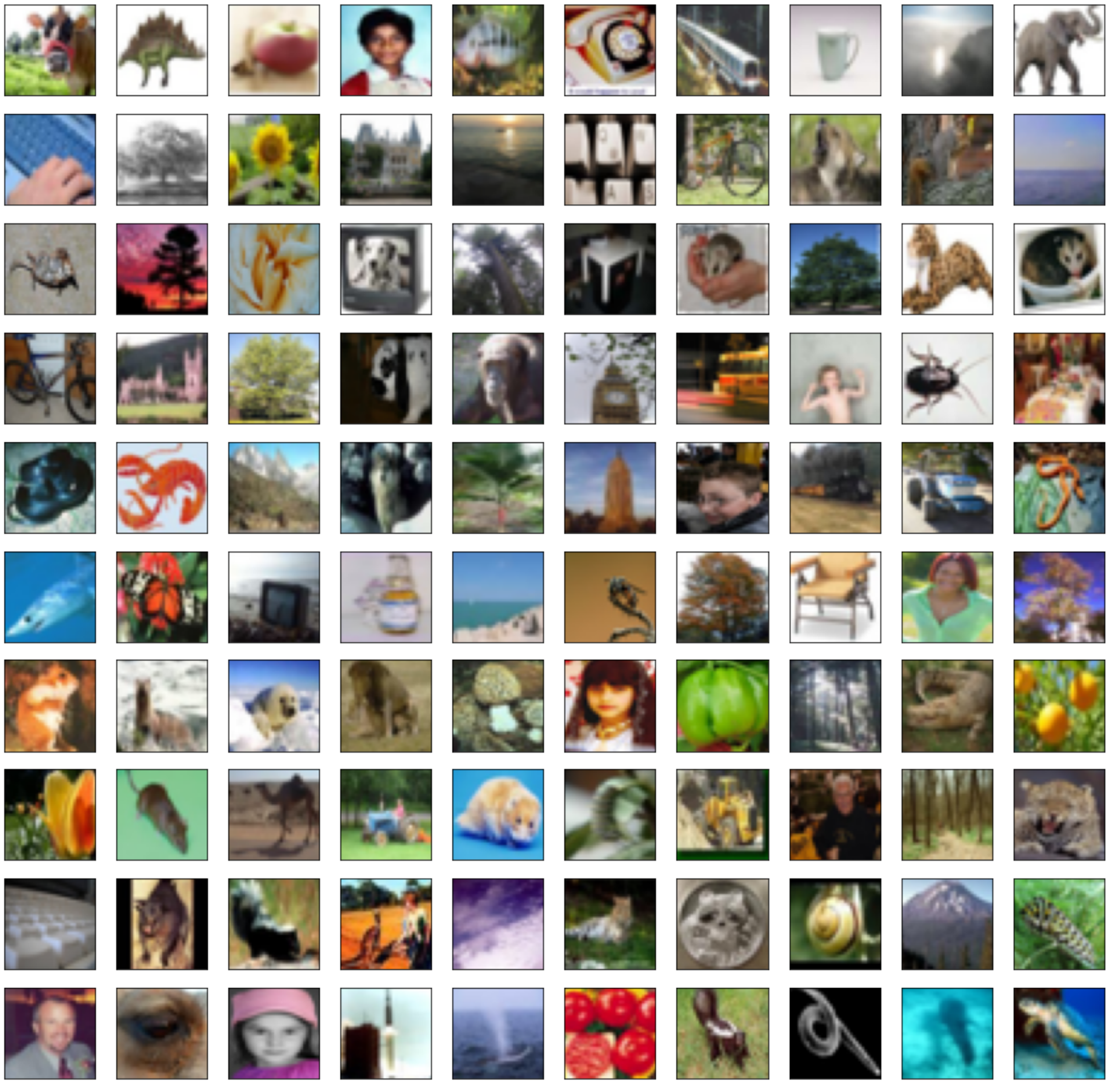
```
(50000, 32, 32, 3)
(10000, 32, 32, 3)
(50000, 1)
(10000, 1)
```

[ ]:
```
x_train=x_train.astype('float32')
x_test=x_test.astype('float32')
x_train/=255
x_test/=255
y_train=to_categorical(y_train,100)
y_test=to_categorical(y_test,100)
```

[ ]:
```
plt.figure(figsize=(15,15))
for i in range(100):
  plt.subplot(10,10,i+1)
  plt.xticks([])
  plt.yticks([])
  plt.imshow(x_train[i])

plt.show
```

[ ]: <function matplotlib.pyplot.show>



[ ]:
```
model = Sequential()
model.add(Flatten(input_shape=(32,32,3)))
model.add(Dense(784,activation='relu'))
model.add(Dense(512,activation='relu'))
model.add(Dense(100,activation='Softmax'))
model.summary()
```

```
Model: "sequential_13"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_12 (Flatten)        (None, 3072)              0
```

```
 dense_36 (Dense)            (None, 784)              2409232


 dense_37 (Dense)            (None, 512)              401920


 dense_38 (Dense)            (None, 100)              51300

=================================================================
Total params: 2,862,452
Trainable params: 2,862,452
Non-trainable params: 0
_________________________________________________________________
```

[ ]:
```
opt=SGD(learning_rate=0.005,momentum=0.9)
model.
 ↪compile(loss='categorical_crossentropy',optimizer=opt,metrics=['accuracy'])
```

[ ]:
```
history=model.
 ↪fit(x_train,y_train,batch_size=128,epochs=50,verbose=1,validation_data=(x_test,y_test))
```

```
Epoch 1/50
391/391 [==============================] - 22s 54ms/step - loss: 4.1782 -
accuracy: 0.0735 - val_loss: 3.9019 - val_accuracy: 0.1125
Epoch 2/50
391/391 [==============================] - 21s 53ms/step - loss: 3.7742 -
accuracy: 0.1335 - val_loss: 3.7150 - val_accuracy: 0.1417
Epoch 3/50
391/391 [==============================] - 21s 53ms/step - loss: 3.6173 -
accuracy: 0.1610 - val_loss: 3.5951 - val_accuracy: 0.1670
Epoch 4/50
391/391 [==============================] - 20s 52ms/step - loss: 3.5107 -
accuracy: 0.1793 - val_loss: 3.5321 - val_accuracy: 0.1806
Epoch 5/50
391/391 [==============================] - 25s 65ms/step - loss: 3.4177 -
accuracy: 0.1990 - val_loss: 3.5066 - val_accuracy: 0.1800
Epoch 6/50
391/391 [==============================] - 20s 52ms/step - loss: 3.3396 -
accuracy: 0.2095 - val_loss: 3.4192 - val_accuracy: 0.2026
Epoch 7/50
391/391 [==============================] - 20s 52ms/step - loss: 3.2656 -
accuracy: 0.2247 - val_loss: 3.3774 - val_accuracy: 0.2033
Epoch 8/50
391/391 [==============================] - 21s 54ms/step - loss: 3.2074 -
accuracy: 0.2338 - val_loss: 3.3809 - val_accuracy: 0.2078
Epoch 9/50
391/391 [==============================] - 21s 53ms/step - loss: 3.1452 -
accuracy: 0.2464 - val_loss: 3.3002 - val_accuracy: 0.2227
Epoch 10/50
```

```
391/391 [==============================] - 21s 53ms/step - loss: 3.0920 -
accuracy: 0.2570 - val_loss: 3.2857 - val_accuracy: 0.2244
Epoch 11/50
391/391 [==============================] - 21s 53ms/step - loss: 3.0399 -
accuracy: 0.2647 - val_loss: 3.2673 - val_accuracy: 0.2274
Epoch 12/50
391/391 [==============================] - 21s 53ms/step - loss: 2.9895 -
accuracy: 0.2746 - val_loss: 3.2265 - val_accuracy: 0.2379
Epoch 13/50
391/391 [==============================] - 21s 52ms/step - loss: 2.9392 -
accuracy: 0.2832 - val_loss: 3.2166 - val_accuracy: 0.2398
Epoch 14/50
391/391 [==============================] - 21s 53ms/step - loss: 2.8957 -
accuracy: 0.2919 - val_loss: 3.1777 - val_accuracy: 0.2447
Epoch 15/50
391/391 [==============================] - 21s 54ms/step - loss: 2.8489 -
accuracy: 0.3033 - val_loss: 3.1670 - val_accuracy: 0.2533
Epoch 16/50
391/391 [==============================] - 21s 55ms/step - loss: 2.8078 -
accuracy: 0.3093 - val_loss: 3.1586 - val_accuracy: 0.2569
Epoch 17/50
391/391 [==============================] - 21s 54ms/step - loss: 2.7652 -
accuracy: 0.3165 - val_loss: 3.1478 - val_accuracy: 0.2573
Epoch 18/50
391/391 [==============================] - 21s 55ms/step - loss: 2.7191 -
accuracy: 0.3269 - val_loss: 3.1539 - val_accuracy: 0.2564
Epoch 19/50
391/391 [==============================] - 21s 55ms/step - loss: 2.6738 -
accuracy: 0.3360 - val_loss: 3.1334 - val_accuracy: 0.2584
Epoch 20/50
391/391 [==============================] - 22s 57ms/step - loss: 2.6341 -
accuracy: 0.3448 - val_loss: 3.1182 - val_accuracy: 0.2618
Epoch 21/50
391/391 [==============================] - 21s 55ms/step - loss: 2.5891 -
accuracy: 0.3534 - val_loss: 3.1114 - val_accuracy: 0.2609
Epoch 22/50
391/391 [==============================] - 21s 54ms/step - loss: 2.5449 -
accuracy: 0.3623 - val_loss: 3.1095 - val_accuracy: 0.2686
Epoch 23/50
391/391 [==============================] - 22s 56ms/step - loss: 2.5122 -
accuracy: 0.3658 - val_loss: 3.1015 - val_accuracy: 0.2676
Epoch 24/50
391/391 [==============================] - 22s 57ms/step - loss: 2.4663 -
accuracy: 0.3742 - val_loss: 3.0990 - val_accuracy: 0.2719
Epoch 25/50
391/391 [==============================] - 22s 56ms/step - loss: 2.4229 -
accuracy: 0.3866 - val_loss: 3.0830 - val_accuracy: 0.2733
Epoch 26/50
```

```
391/391 [==============================] - 21s 54ms/step - loss: 2.3760 -
accuracy: 0.3974 - val_loss: 3.1087 - val_accuracy: 0.2743
Epoch 27/50
391/391 [==============================] - 21s 55ms/step - loss: 2.3392 -
accuracy: 0.4051 - val_loss: 3.1246 - val_accuracy: 0.2717
Epoch 28/50
391/391 [==============================] - 22s 57ms/step - loss: 2.3002 -
accuracy: 0.4142 - val_loss: 3.1020 - val_accuracy: 0.2737
Epoch 29/50
391/391 [==============================] - 21s 54ms/step - loss: 2.2558 -
accuracy: 0.4221 - val_loss: 3.1539 - val_accuracy: 0.2678
Epoch 30/50
391/391 [==============================] - 21s 54ms/step - loss: 2.2142 -
accuracy: 0.4326 - val_loss: 3.1441 - val_accuracy: 0.2744
Epoch 31/50
391/391 [==============================] - 22s 56ms/step - loss: 2.1744 -
accuracy: 0.4413 - val_loss: 3.1290 - val_accuracy: 0.2788
Epoch 32/50
391/391 [==============================] - 21s 54ms/step - loss: 2.1331 -
accuracy: 0.4499 - val_loss: 3.1349 - val_accuracy: 0.2793
Epoch 33/50
391/391 [==============================] - 21s 54ms/step - loss: 2.0912 -
accuracy: 0.4593 - val_loss: 3.1548 - val_accuracy: 0.2781
Epoch 34/50
391/391 [==============================] - 26s 67ms/step - loss: 2.0522 -
accuracy: 0.4688 - val_loss: 3.1629 - val_accuracy: 0.2823
Epoch 35/50
391/391 [==============================] - 21s 54ms/step - loss: 1.9990 -
accuracy: 0.4807 - val_loss: 3.1677 - val_accuracy: 0.2842
Epoch 36/50
391/391 [==============================] - 22s 55ms/step - loss: 1.9609 -
accuracy: 0.4888 - val_loss: 3.2178 - val_accuracy: 0.2757
Epoch 37/50
391/391 [==============================] - 22s 56ms/step - loss: 1.9247 -
accuracy: 0.4966 - val_loss: 3.2412 - val_accuracy: 0.2747
Epoch 38/50
391/391 [==============================] - 21s 55ms/step - loss: 1.8779 -
accuracy: 0.5084 - val_loss: 3.2274 - val_accuracy: 0.2803
Epoch 39/50
391/391 [==============================] - 22s 57ms/step - loss: 1.8412 -
accuracy: 0.5170 - val_loss: 3.2567 - val_accuracy: 0.2757
Epoch 40/50
391/391 [==============================] - 22s 55ms/step - loss: 1.7893 -
accuracy: 0.5319 - val_loss: 3.2526 - val_accuracy: 0.2807
Epoch 41/50
391/391 [==============================] - 21s 54ms/step - loss: 1.7434 -
accuracy: 0.5409 - val_loss: 3.2937 - val_accuracy: 0.2775
Epoch 42/50
```

```
391/391 [==============================] - 22s 56ms/step - loss: 1.7044 -
accuracy: 0.5501 - val_loss: 3.2722 - val_accuracy: 0.2822
Epoch 43/50
391/391 [==============================] - 22s 56ms/step - loss: 1.6624 -
accuracy: 0.5618 - val_loss: 3.3398 - val_accuracy: 0.2749
Epoch 44/50
391/391 [==============================] - 21s 54ms/step - loss: 1.6337 -
accuracy: 0.5678 - val_loss: 3.3575 - val_accuracy: 0.2775
Epoch 45/50
391/391 [==============================] - 22s 56ms/step - loss: 1.5880 -
accuracy: 0.5797 - val_loss: 3.4482 - val_accuracy: 0.2706
Epoch 46/50
391/391 [==============================] - 22s 56ms/step - loss: 1.5362 -
accuracy: 0.5917 - val_loss: 3.4034 - val_accuracy: 0.2820
Epoch 47/50
391/391 [==============================] - 22s 56ms/step - loss: 1.4994 -
accuracy: 0.6010 - val_loss: 3.4470 - val_accuracy: 0.2746
Epoch 48/50
391/391 [==============================] - 25s 65ms/step - loss: 1.4616 -
accuracy: 0.6101 - val_loss: 3.4571 - val_accuracy: 0.2798
Epoch 49/50
391/391 [==============================] - 25s 65ms/step - loss: 1.4144 -
accuracy: 0.6209 - val_loss: 3.5053 - val_accuracy: 0.2765
Epoch 50/50
391/391 [==============================] - 22s 57ms/step - loss: 1.3792 -
accuracy: 0.6320 - val_loss: 3.6070 - val_accuracy: 0.2719
```

[ ]:
```
score=model.evaluate(x_test,y_test,verbose=1)
print("do chinh xac = ",score[1])
```

```
313/313 [==============================] - 3s 11ms/step - loss: 3.6070 -
accuracy: 0.2719
do chinh xac =  0.2718999981880188
```

[ ]:
```
model.save('/content/drive/MyDrive/dulieuAI/cifar100/train_cifar100.h5')
```

4 Fashion

[ ]:
```
from keras.datasets import fashion_mnist
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.layers import Dense , Activation, Dropout
from tensorflow.keras.utils import to_categorical
```

```python
from tensorflow.keras.models import load_model
from tensorflow.keras.utils import load_img,img_to_array
from keras.backend import categorical_crossentropy
from tensorflow.keras.optimizers import RMSprop
from keras.layers import Flatten
from tensorflow.keras.models import load_model
```

[ ]:
```python
(x_train,y_train),(x_test,y_test)= fashion_mnist.load_data()
x_test_bft=x_test
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/train-labels-idx1-ubyte.gz
32768/29515 [==================================] - 0s 0us/step
40960/29515 [==========================================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/train-images-idx3-ubyte.gz
26427392/26421880 [==============================] - 0s 0us/step
26435584/26421880 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/t10k-labels-idx1-ubyte.gz
16384/5148 [====================================================================
===========================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/t10k-images-idx3-ubyte.gz
4423680/4422102 [==============================] - 0s 0us/step
4431872/4422102 [==============================] - 0s 0us/step
```

[ ]:
```python
plt.figure(figsize=(10,10))
for i in range(25):
  plt.subplot(5,5,i+1)
  plt.xticks([])
  plt.yticks([])
  plt.imshow(x_train[i])
  plt.xlabel(y_train[i])
plt.show()
```

```
[ ]: x_train=x_train.reshape(60000,784)
     x_test=x_test.reshape(10000,784)#28x28
     x_train=x_train.astype('float32')
     x_test=x_test.astype('float32')
     x_train/=255
     x_test/=255
     y_train=to_categorical(y_train,10)
     y_test=to_categorical(y_test,10)
```

```
[ ]: model = Sequential()
     model.add(Dense(128,activation='relu',input_shape=(784,)))
     model.add(Dense(128,activation='relu'))
     model.add(Dense(10,activation='Softmax'))
```

```
model.summary()
model.compile(loss='categorical_crossentropy',optimizer=RMSprop(),␣
 ↪metrics=['accuracy'])
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_3 (Dense)             (None, 128)               100480

 dense_4 (Dense)             (None, 128)               16512

 dense_5 (Dense)             (None, 10)                1290

=================================================================
Total params: 118,282
Trainable params: 118,282
Non-trainable params: 0
_________________________________________________________________
```

[ ]:
```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(60000, 784)
(10000, 784)
(60000, 10)
(10000, 10)
```

[ ]:
```
model.
 ↪fit(x_train,y_train,batch_size=128,epochs=1,verbose=10,validation_data=(x_test,y_test))
```

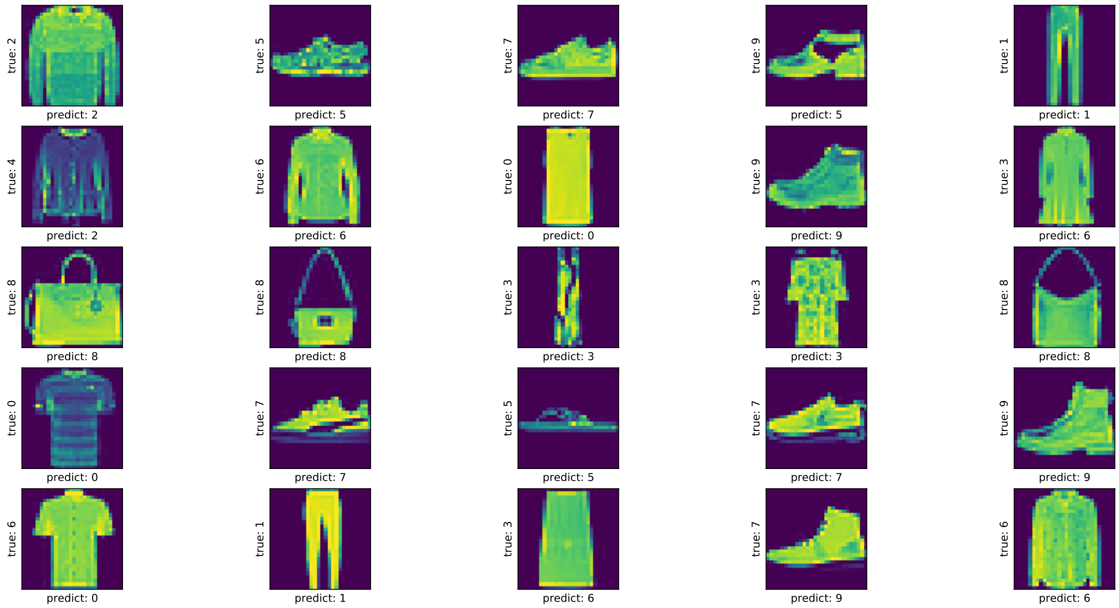
[ ]: <keras.callbacks.History at 0x7fd4aa6a6490>

[ ]:
```
score=model.evaluate(x_test,y_test,verbose=1)
print("do chinh xac = ",score[1])
```

```
313/313 [==============================] - 1s 2ms/step - loss: 0.4717 -
accuracy: 0.8261
do chinh xac =  0.8260999917984009
```

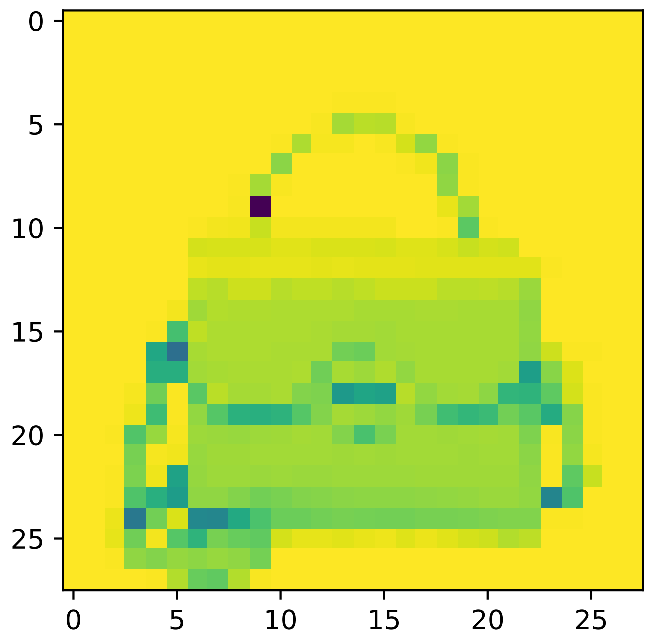
[ ]:
```
y_pred = model.predict(x_test)
plt.figure(figsize=(20,10))
for i in range(20,45,1):
  plt.subplot(5,5,i+1-20)
  plt.imshow(x_test_bft[i])
```

```
  plt.xticks([])
  plt.yticks([])
  plt.xlabel("predict: "+str(np.argmax(y_pred[i])))
  plt.ylabel("true: "+str(np.argmax(y_test[i])))
plt.show()
```



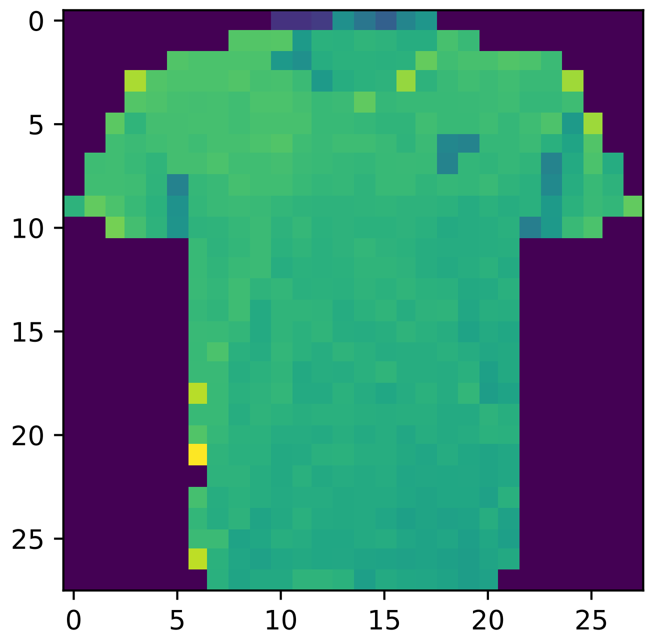
[ ]:
```
url='/content/tui xach.jfif'
img=load_img(url,target_size=(28,28),color_mode="grayscale")
plt.imshow(img)
img=img_to_array(img)
img=img.reshape(1,784)
img=img/255.0
print(np.argmax(model.predict(img)))
```

8



[ ]:
```
url='/content/t_shirt.png'
img=load_img(url,target_size=(28,28),color_mode="grayscale")
plt.imshow(img)
img=img_to_array(img)
img=img.reshape(1,784)
img=img/255.0
print(np.argmax(model.predict(img)))
```

```
0
```




```
[ ]: model.save('/content/drive/MyDrive/dulieuAI/fashion/train_fashion.h5')
```

```
[ ]: load_model('/content/drive/MyDrive/dulieuAI/fashion/train_fashion.h5')
```

```
[ ]: <keras.engine.sequential.Sequential at 0x7fd4ad158990>
```

5 FACEID_GROUP

```
[ ]: import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.utils import load_img,img_to_array
from tensorflow.keras.preprocessing import image
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.utils import np_utils
from keras.layers import Dense,Activation,Dropout,LSTM,BatchNormalization
from keras.layers import Flatten
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.utils import to_categorical
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
```

[ ]:
```
trainset='/content/drive/MyDrive/Colab Notebooks/GROUP/train'
```

[ ]:
```
train=ImageDataGenerator(rescale=1./255,validation_split=0.1)
```

[ ]:
```
train_data=train.flow_from_directory(
    trainset,

    target_size=(150,150),
    batch_size=10,
    class_mode='categorical',
    subset="training",
    shuffle=True,
   )
validation_set=train.flow_from_directory(
   trainset,
    target_size=(150,150),
    batch_size=10,
    class_mode='categorical',
    shuffle=True,
    subset="validation",
    )
```

```
Found 104 images belonging to 4 classes.
Found 10 images belonging to 4 classes.
```

[ ]:
```
print(train_data.class_indices)
print(validation_set.class_indices)
```

```
{'binh': 0, 'others': 1, 'tien': 2, 'trieu': 3}
{'binh': 0, 'others': 1, 'tien': 2, 'trieu': 3}
```

[ ]:
```
model=Sequential()
model.add(Conv2D(32,(3,3),activation='relu',input_shape=(150,150,3)))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(64,(3,3),activation='relu'))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(128,(3,3),activation='relu'))
model.add(Flatten())
model.add(Dense(128,activation='relu'))
model.add(Dense(4,activation='softmax'))
```

[ ]:
```
model.
 ↪compile(loss='categorical_crossentropy',optimizer='rmsprop',metrics=['accuracy'])
```

[ ]:
```
history=model.
 ↪fit(train_data,batch_size=10,epochs=5,verbose=1,validation_data=validation_set)
```

```
Epoch 1/5
```

```
11/11 [==============================] - 21s 2s/step - loss: 24.6193 - accuracy:
0.3750 - val_loss: 0.9128 - val_accuracy: 0.5000
Epoch 2/5
11/11 [==============================] - 19s 2s/step - loss: 0.6865 - accuracy:
0.8365 - val_loss: 2.8139 - val_accuracy: 0.5000
Epoch 3/5
11/11 [==============================] - 19s 2s/step - loss: 0.9478 - accuracy:
0.8558 - val_loss: 0.1874 - val_accuracy: 0.9000
Epoch 4/5
11/11 [==============================] - 19s 2s/step - loss: 0.0073 - accuracy:
1.0000 - val_loss: 0.3524 - val_accuracy: 0.9000
Epoch 5/5
11/11 [==============================] - 19s 2s/step - loss: 0.0018 - accuracy:
1.0000 - val_loss: 0.0612 - val_accuracy: 1.0000
```

[ ]:
```
model.save('/content/drive/MyDrive/dulieuAI/faceid_group/')
```

[ ]:
```
model_gr=load_model('/content/drive/MyDrive/dulieuAI/faceid_group')
```

[ ]:
```
<keras.engine.sequential.Sequential at 0x7fd4aa2dd5d0>
```

[ ]:
```
test_url= '/content/drive/MyDrive/Colab Notebooks/TEST_GROUP'
test=ImageDataGenerator(rescale=1./255)
```

[ ]:
```
score=model.evaluate(validation_set,verbose=1)
print("test accuracy = ",score[1])
```

```
1/1 [==============================] - 1s 1s/step - loss: 0.0612 - accuracy:
1.0000
test accuracy =  1.0
```

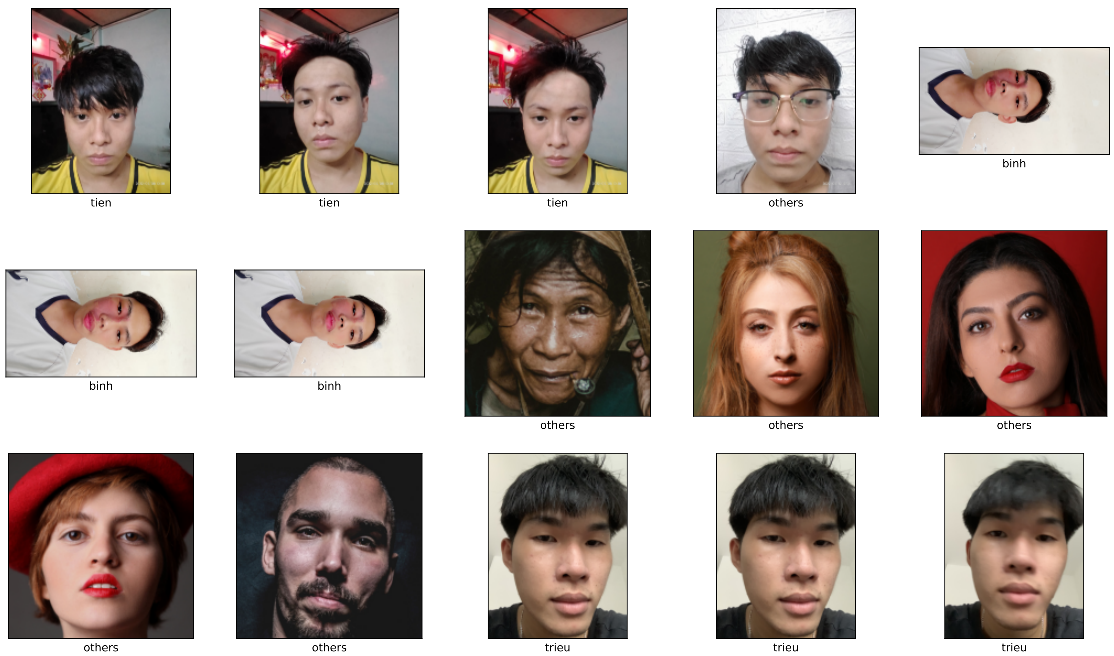
[ ]:
```
test_data=test.flow_from_directory(
    test_url,
    target_size=(150,150),
    batch_size=10,
    class_mode='categorical',
    shuffle=False,
   )
```

```
Found 15 images belonging to 1 classes.
```

[ ]:
```
results={ 0:'binh',1:'others',2:'tien',3:'trieu'}
pred = model.predict_generator(test_data)
plt.figure(figsize=(18,18))
for i in range(pred.shape[0]):
  plt.subplot(5,5,i+1)
  plt.xticks([])
  plt.yticks([])
```

```
    plt.imshow(load_img(test_data.filepaths[i]))
    plt.xlabel(results[np.argmax(pred[i])])
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: UserWarning:
`Model.predict_generator` is deprecated and will be removed in a future version.
Please use `Model.predict`, which supports generators.
```



6 2DOF

[96]:
```
import math as ma
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.utils import np_utils
from keras.layers import Dense,Activation,Dropout,LSTM,BatchNormalization
from keras.layers import Flatten
from tensorflow.keras.optimizers import SGD,Adam
l1=50
l2=40
data=pd.DataFrame()
data_test=pd.DataFrame()
```

[97]:
```
Px=[]
Py=[]
goctt1=[]
goctt2=[]
for tt1 in range(-150,150,1):
  for tt2 in range(-90,90,1):
    goctt1.append((tt1*ma.pi)/180)
    goctt2.append((tt2*ma.pi)/180)
    theta1=(tt1*ma.pi)/180
    theta2=(tt2*ma.pi)/180
    Px.append(l1*ma.cos(theta1)+l2*ma.cos(theta1+theta2))
    Py.append(l1*ma.sin(theta1)+l2*ma.sin(theta1+theta2))
```

[98]:
```
data['theta1']=goctt1
data['theta2']=goctt2
data['Px']=Px
data['Py']=Py
data
```

[98]:

	theta1	theta2	Px	Py
0	-2.617994	-1.570796	-63.301270	9.641016
1	-2.617994	-1.553343	-63.902793	9.286692
2	-2.617994	-1.535890	-64.498041	8.921924
3	-2.617994	-1.518436	-65.086832	8.546823
4	-2.617994	-1.500983	-65.668986	8.161503
…	…	…	…	…
53995	2.600541	1.483530	-66.369775	-6.608776
53996	2.600541	1.500983	-65.801422	-7.014178
53997	2.600541	1.518436	-65.226081	-7.409599
53998	2.600541	1.535890	-64.643926	-7.794919
53999	2.600541	1.553343	-64.055136	-8.170020

[54000 rows x 4 columns]

[99]:
```
x=data.drop(data.columns[0:2],axis=1)
y=data.drop(data.columns[2:4],axis=1)
```

[100]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.1)
```

[89]:
```
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(58320, 2)
(58320, 2)
(6480, 2)
```

```
(6480, 2)
```

[101]:
```python
model = Sequential()
model.add(Dense(784,activation='relu',input_shape=(2,)))
model.add(Dense(512,activation='relu'))
model.add(Dense(2))
```

[102]:
```python
model.compile(loss='mse',optimizer='Adam',metrics=['accuracy'])
history=model.
 ↪fit(x_train,y_train,batch_size=64,epochs=20,verbose=1,validation_data=(x_test,y_test))
```

```
Epoch 1/20
760/760 [==============================] - 3s 3ms/step - loss: 0.8922 -
accuracy: 0.7668 - val_loss: 0.5048 - val_accuracy: 0.7867
Epoch 2/20
760/760 [==============================] - 2s 3ms/step - loss: 0.5180 -
accuracy: 0.7772 - val_loss: 0.4837 - val_accuracy: 0.7869
Epoch 3/20
760/760 [==============================] - 2s 3ms/step - loss: 0.4958 -
accuracy: 0.7795 - val_loss: 0.4868 - val_accuracy: 0.7906
Epoch 4/20
760/760 [==============================] - 2s 3ms/step - loss: 0.4859 -
accuracy: 0.7801 - val_loss: 0.4846 - val_accuracy: 0.7833
Epoch 5/20
760/760 [==============================] - 3s 4ms/step - loss: 0.4779 -
accuracy: 0.7800 - val_loss: 0.4755 - val_accuracy: 0.7841
Epoch 6/20
760/760 [==============================] - 2s 3ms/step - loss: 0.4762 -
accuracy: 0.7797 - val_loss: 0.4679 - val_accuracy: 0.7859
Epoch 7/20
760/760 [==============================] - 2s 3ms/step - loss: 0.4762 -
accuracy: 0.7803 - val_loss: 0.4780 - val_accuracy: 0.7854
Epoch 8/20
760/760 [==============================] - 2s 3ms/step - loss: 0.4737 -
accuracy: 0.7799 - val_loss: 0.4663 - val_accuracy: 0.7865
Epoch 9/20
760/760 [==============================] - 2s 3ms/step - loss: 0.4736 -
accuracy: 0.7802 - val_loss: 0.4659 - val_accuracy: 0.7870
Epoch 10/20
760/760 [==============================] - 2s 3ms/step - loss: 0.4729 -
accuracy: 0.7801 - val_loss: 0.4638 - val_accuracy: 0.7889
Epoch 11/20
760/760 [==============================] - 2s 3ms/step - loss: 0.4696 -
accuracy: 0.7804 - val_loss: 0.4598 - val_accuracy: 0.7837
Epoch 12/20
760/760 [==============================] - 2s 3ms/step - loss: 0.4687 -
accuracy: 0.7798 - val_loss: 0.4765 - val_accuracy: 0.7835
Epoch 13/20
```

```
760/760 [==============================] - 2s 3ms/step - loss: 0.4609 -
accuracy: 0.7799 - val_loss: 0.4572 - val_accuracy: 0.7841
Epoch 14/20
760/760 [==============================] - 2s 3ms/step - loss: 0.4548 -
accuracy: 0.7793 - val_loss: 0.4385 - val_accuracy: 0.7837
Epoch 15/20
760/760 [==============================] - 2s 3ms/step - loss: 0.4492 -
accuracy: 0.7796 - val_loss: 0.4427 - val_accuracy: 0.7857
Epoch 16/20
760/760 [==============================] - 2s 3ms/step - loss: 0.4501 -
accuracy: 0.7787 - val_loss: 0.4404 - val_accuracy: 0.7848
Epoch 17/20
760/760 [==============================] - 2s 3ms/step - loss: 0.4485 -
accuracy: 0.7780 - val_loss: 0.4448 - val_accuracy: 0.7796
Epoch 18/20
760/760 [==============================] - 2s 3ms/step - loss: 0.4474 -
accuracy: 0.7772 - val_loss: 0.4436 - val_accuracy: 0.7826
Epoch 19/20
760/760 [==============================] - 2s 3ms/step - loss: 0.4468 -
accuracy: 0.7778 - val_loss: 0.4479 - val_accuracy: 0.7822
Epoch 20/20
760/760 [==============================] - 2s 3ms/step - loss: 0.4468 -
accuracy: 0.7768 - val_loss: 0.4370 - val_accuracy: 0.7802
```

[103]:
```
score=model.evaluate(x_test,y_test,verbose=1)
print("do chinh xac = ",score[1])
```

```
169/169 [==============================] - 0s 2ms/step - loss: 0.4370 -
accuracy: 0.7802
do chinh xac =  0.7801851630210876
```

[104]:
```
model.predict(x_test)
```

[104]:
```
array([[-2.4381435 , -0.25730884],
       [-1.507339  ,  0.04999128],
       [-1.3871214 ,  0.04511946],
       …,
       [ 2.4265735 ,  0.4184114 ],
       [-1.9268329 , -0.8616683 ],
       [-0.82285476,  0.02225251]], dtype=float32)
```

[105]:
```
y_test
```

[105]:

	theta1	theta2
1877	-2.443461	-0.226893
13930	-1.274090	-0.349066
16252	-1.047198	-0.663225
52137	2.426008	0.471239

```
46482  1.884956 -0.837758
…         …        …
30620  0.349066 -1.221730
45694  1.797689  1.117011
53014  2.513274  0.069813
8303  -1.815142 -1.169371
22549 -0.436332 -0.715585

[5400 rows x 2 columns]
```

[106]:
```
model.save('/content/drive/MyDrive/dulieuAI/2dof/train_2dof.h5')
```

7 FACEID_PERSONAL

[ ]:
```
import numpy as np
from tensorflow import keras
from tensorflow.keras.models import load_model
from tensorflow.keras.utils import load_img,img_to_array
from tensorflow.keras.preprocessing import image
from tensorflow.keras.optimizers import SGD,Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.utils import np_utils
from keras.layers import Dense,Activation,Dropout,LSTM,BatchNormalization
from keras.layers import Flatten
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.utils import to_categorical
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
```

[ ]:
```
trainset='/content/drive/MyDrive/Colab Notebooks/PERSONAL/train'
```

[ ]:
```
data_generator = ImageDataGenerator(rescale=1./255,validation_split=0.2)
```

[ ]:
```
train_dataset=data_generator.flow_from_directory(trainset,
                                     target_size=(150,150),
                                     batch_size=10,
                                     class_mode='categorical',
                                     subset="training",
                                     shuffle=True,)
validation_set=data_generator.flow_from_directory(trainset,
                                              target_size=(150,150),
                                              batch_size=10,
                                              class_mode='categorical',
```

```
                                                 subset="validation",
                                                 shuffle=True)
```

```
Found 143 images belonging to 2 classes.
Found 35 images belonging to 2 classes.
```

[ ]:
```
validation_set.class_indices
```

[ ]:
```
{'kptien': 0, 'tien': 1}
```

[ ]:
```
model = Sequential()
model.add(Flatten(input_shape=(150,150,3)))
model.add(Dense(784,activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(512,activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(2,activation='softmax'))
```

[ ]:
```
model.
 ↪compile(loss='categorical_crossentropy',optimizer='Adam',metrics=['accuracy'])
```

[ ]:
```
history=model.
 ↪fit(train_dataset,batch_size=10,epochs=10,verbose=1,validation_data=validation_set)
```

```
Epoch 1/10
15/15 [==============================] - 17s 1s/step - loss: 25.9957 - accuracy:
0.6084 - val_loss: 36.7309 - val_accuracy: 0.6857
Epoch 2/10
15/15 [==============================] - 15s 992ms/step - loss: 5.5170 -
accuracy: 0.7832 - val_loss: 4.7542 - val_accuracy: 0.6571
Epoch 3/10
15/15 [==============================] - 15s 999ms/step - loss: 3.4718 -
accuracy: 0.8671 - val_loss: 13.3014 - val_accuracy: 0.6857
Epoch 4/10
15/15 [==============================] - 15s 1s/step - loss: 2.3816 - accuracy:
0.8881 - val_loss: 13.5660 - val_accuracy: 0.6857
Epoch 5/10
15/15 [==============================] - 15s 1s/step - loss: 1.8198 - accuracy:
0.9301 - val_loss: 7.9115 - val_accuracy: 0.6571
Epoch 6/10
15/15 [==============================] - 20s 1s/step - loss: 4.8556 - accuracy:
0.8671 - val_loss: 2.4773 - val_accuracy: 0.8286
Epoch 7/10
15/15 [==============================] - 15s 1s/step - loss: 0.8752 - accuracy:
0.9650 - val_loss: 7.0022 - val_accuracy: 0.7143
Epoch 8/10
15/15 [==============================] - 15s 1s/step - loss: 0.2826 - accuracy:
```

```
0.9790 - val_loss: 7.2781 - val_accuracy: 0.7143
Epoch 9/10
15/15 [==============================] - 15s 1s/step - loss: 0.9629 - accuracy:
0.9580 - val_loss: 1.7935 - val_accuracy: 0.8571
Epoch 10/10
15/15 [==============================] - 15s 1s/step - loss: 1.4478 - accuracy:
0.9441 - val_loss: 11.5498 - val_accuracy: 0.7143
```

[ ]:
```
score=model.evaluate(validation_set,verbose=1)
print("test accuracy = ",score[1])
```

```
4/4 [==============================] - 7s 1s/step - loss: 11.5498 - accuracy:
0.7143
test accuracy =  0.7142857313156128
```

[ ]:
```
model.save('/content/drive/MyDrive/dulieuAI/faceid_personal')
```

```
INFO:tensorflow:Assets written to:
/content/drive/MyDrive/dulieuAI/faceid_personal/assets
```

[ ]:
```
mode_per=load_model('/content/drive/MyDrive/dulieuAI/faceid_personal')
```

[ ]:
```
<keras.engine.sequential.Sequential at 0x7fd4a7649590>
```

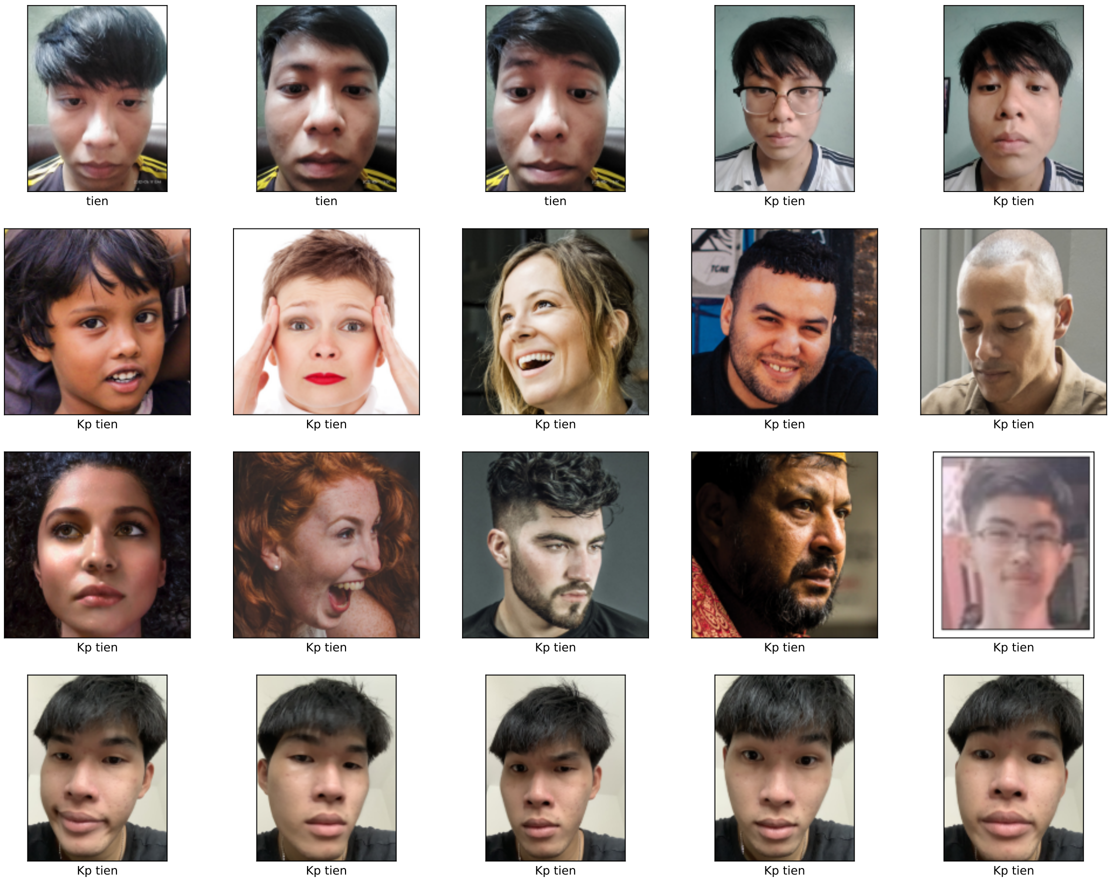
[ ]:
```
test_url= '/content/drive/MyDrive/Colab Notebooks/TEST_PERSONAL'
test=ImageDataGenerator(rescale=1./255)
```

[ ]:
```
test_data=test.flow_from_directory(
    test_url,
    target_size=(150,150),
    batch_size=15,
    class_mode='categorical',
    shuffle=False,
   )
```

```
Found 20 images belonging to 1 classes.
```

[ ]:
```
results={ 0:'Kp tien',1:'tien'}
pred = model.predict_generator(test_data)
plt.figure(figsize=(17,17))
for i in range(pred.shape[0]):
  plt.subplot(5,5,i+1)
  plt.xticks([])
  plt.yticks([])
  plt.imshow(load_img(test_data.filepaths[i]))
  plt.xlabel(results[np.argmax(pred[i])])
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: UserWarning:
`Model.predict_generator` is deprecated and will be removed in a future version.
Please use `Model.predict`, which supports generators.
```



8 3D0F

[107]:
```
import math as ma
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.utils import np_utils
from keras.layers import Dense,Activation,Dropout,LSTM,BatchNormalization
from keras.layers import Flatten
from tensorflow.keras.optimizers import SGD,Adam
from keras.callbacks import EarlyStopping
l1=50
```

```
l2=40
l3=20
data=pd.DataFrame()
```

[108]:
```
Px=[]
Py=[]
goctt1=[]
goctt2=[]
goctt3=[]
gocphi=[]
for phi in range(0,90,45):
  for tt1 in range(-150,150,1):
    for tt2 in range(-90,90,1):
     theta1=(tt1*ma.pi)/180
     theta2=(tt2*ma.pi)/180
     a=((phi*ma.pi)/180)-theta1-theta2 # phi = theta1 + theta2 + theta3 ====>␣
↪theta3=a=phi(rad) - theta1 - theta2
     Px.append(l1*ma.cos(theta1)+l3*ma.cos(theta1+theta2+a)+l2*ma.
↪cos(theta1+theta2))
     Py.append(l1*ma.sin(theta1)+l3*ma.sin(theta1+theta2+a)+l2*ma.
↪sin(theta1+theta2))
     goctt1.append(theta1)
     goctt2.append(theta2)
     gocphi.append((phi*ma.pi)/180)
     goctt3.append(a)
```

[109]:
```
data['theta1']=goctt1
data['theta2']=goctt2
data['theta3']=goctt3
data['phi']=gocphi
data['Px']=Px
data['Py']=Py
data
```

[109]:

	theta1	theta2	theta3	phi	Px	Py
0	-2.617994	-1.570796	4.188790	0.000000	-43.301270	9.641016
1	-2.617994	-1.553343	4.171337	0.000000	-43.902793	9.286692
2	-2.617994	-1.535890	4.153884	0.000000	-44.498041	8.921924
3	-2.617994	-1.518436	4.136430	0.000000	-45.086832	8.546823
4	-2.617994	-1.500983	4.118977	0.000000	-45.668986	8.161503
…	…	…	…	…	…	…
107995	2.600541	1.483530	-3.298672	0.785398	-52.227640	7.533360
107996	2.600541	1.500983	-3.316126	0.785398	-51.659287	7.127958
107997	2.600541	1.518436	-3.333579	0.785398	-51.083946	6.732536
107998	2.600541	1.535890	-3.351032	0.785398	-50.501791	6.347217
107999	2.600541	1.553343	-3.368485	0.785398	-49.913000	5.972116

```
[108000 rows x 6 columns]
```

```
[110]: y = data.drop(['Px','Py','phi'],axis=1)
x = data.drop(['theta1','theta2','theta3'],axis=1)
print(y)
print(x)
```

```
          theta1    theta2    theta3
0      -2.617994 -1.570796  4.188790
1      -2.617994 -1.553343  4.171337
2      -2.617994 -1.535890  4.153884
3      -2.617994 -1.518436  4.136430
4      -2.617994 -1.500983  4.118977
…            …         …         …
107995  2.600541  1.483530 -3.298672
107996  2.600541  1.500983 -3.316126
107997  2.600541  1.518436 -3.333579
107998  2.600541  1.535890 -3.351032
107999  2.600541  1.553343 -3.368485

[108000 rows x 3 columns]
             phi         Px        Py
0       0.000000 -43.301270  9.641016
1       0.000000 -43.902793  9.286692
2       0.000000 -44.498041  8.921924
3       0.000000 -45.086832  8.546823
4       0.000000 -45.668986  8.161503
…            …          …         …
107995  0.785398 -52.227640  7.533360
107996  0.785398 -51.659287  7.127958
107997  0.785398 -51.083946  6.732536
107998  0.785398 -50.501791  6.347217
107999  0.785398 -49.913000  5.972116

[108000 rows x 3 columns]
```

```
[113]: x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.1)

model = Sequential()
model.add(Dense(784,activation='relu',input_shape=(3,)))
model.add(Dropout(0.2))
model.add(Dense(512,activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(3))

model.compile(loss='mse',optimizer='Adam',metrics=['accuracy'])
```

```
history=model.
↪fit(x_train,y_train,batch_size=128,epochs=10,verbose=1,validation_data=(x_test,y_test))
```

```
Epoch 1/10
760/760 [==============================] - 3s 3ms/step - loss: 1.0297 -
accuracy: 0.7599 - val_loss: 0.5779 - val_accuracy: 0.7844
Epoch 2/10
760/760 [==============================] - 2s 3ms/step - loss: 0.5526 -
accuracy: 0.7906 - val_loss: 0.4765 - val_accuracy: 0.8036
Epoch 3/10
760/760 [==============================] - 2s 3ms/step - loss: 0.5132 -
accuracy: 0.7978 - val_loss: 0.4768 - val_accuracy: 0.7948
Epoch 4/10
760/760 [==============================] - 2s 3ms/step - loss: 0.5020 -
accuracy: 0.7994 - val_loss: 0.4631 - val_accuracy: 0.8081
Epoch 5/10
760/760 [==============================] - 2s 3ms/step - loss: 0.4919 -
accuracy: 0.8000 - val_loss: 0.4562 - val_accuracy: 0.8031
Epoch 6/10
760/760 [==============================] - 2s 3ms/step - loss: 0.4837 -
accuracy: 0.8006 - val_loss: 0.4555 - val_accuracy: 0.8006
Epoch 7/10
760/760 [==============================] - 2s 3ms/step - loss: 0.4760 -
accuracy: 0.8011 - val_loss: 0.4401 - val_accuracy: 0.8074
Epoch 8/10
760/760 [==============================] - 2s 3ms/step - loss: 0.4706 -
accuracy: 0.8013 - val_loss: 0.4516 - val_accuracy: 0.8056
Epoch 9/10
760/760 [==============================] - 2s 3ms/step - loss: 0.4663 -
accuracy: 0.8014 - val_loss: 0.4335 - val_accuracy: 0.8061
Epoch 10/10
760/760 [==============================] - 2s 3ms/step - loss: 0.4606 -
accuracy: 0.8011 - val_loss: 0.4337 - val_accuracy: 0.8073
```

[114]:
```
model.save('/content/drive/MyDrive/dulieuAI/3dof/train_3dof.h5')
```

[115]:
```
model.predict(x_test)
```

[115]:
```
array([[ 1.3164849 , -0.03779971, -0.71266174],
       [ 1.478925  , -0.00734305, -1.5263509 ],
       [-2.030914  , -0.18038042,  2.4091587 ],
       …,
       [ 1.0460303 , -0.03325614, -0.4448613 ],
       [-2.099925  ,  0.00428504,  2.8532715 ],
       [-1.4076934 ,  0.03683862,  2.1860106 ]], dtype=float32)
```

[116]:
```
y_test
```

[116]:
```
            theta1    theta2    theta3
100647  1.902409 -1.099557 -0.017453
37604   1.012291  1.291544 -2.303835
6513   -1.989675 -0.994838  2.984513
52366   2.443461  1.326450 -3.769911
66787  -1.378810 -1.448623  3.612832
…            …         …         …
98629   1.692969  1.378810 -2.286381
97404   1.588250 -1.151917  0.349066
98303   1.675516 -1.169371  0.279253
58235  -2.216568  0.087266  2.914700
71487  -0.925025 -1.099557  2.809980

[10800 rows x 3 columns]
```

9 convert pdf

[117]:
```python
from IPython.display import set_matplotlib_formats
set_matplotlib_formats('pdf', 'svg')
```

[118]:
```python
%%capture
!wget -nc https://raw.githubusercontent.com/brpy/colab-pdf/master/colab_pdf.py
from colab_pdf import colab_pdf
colab_pdf('baitapAI_ANN.ipynb')# ten file colab #luu do drive
```