MÔN: TRÍ TUỆ NHÂN TẠO

GVGD: PSG.TS NGUYỄN TRƯỜNG THỊNH

TÊN : TRẦN QUANG HUY

MSSV: 19146195

LỚP: CHIỀU THỨ 6, TIẾT 12-15

Bài tập tuần 12 : Artificial Neural Network

Bài 1: Cifar10

```
#  Gọi các thư viên cần thiết
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from keras.datasets import cifar10
import matplotlib.pyplot as plt
from tensorflow.keras.optimizers import RMSprop
from keras.callbacks import EarlyStopping
from keras.utils import np_utils
from keras.backend import dropout
from keras.models import Sequential
from keras.layers import Dense, Dropout

# Chia dữ liệu thành 2 phần:  phần huyến luyện v
à phần test
```

```
(x_train, y_train),(x_test,y_test) = cifar10.loa
d_data()

# Kích thước các tập dữ liệu
x_train.shape , x_test .shape,  y_train.shape,
y_test.shape
```

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

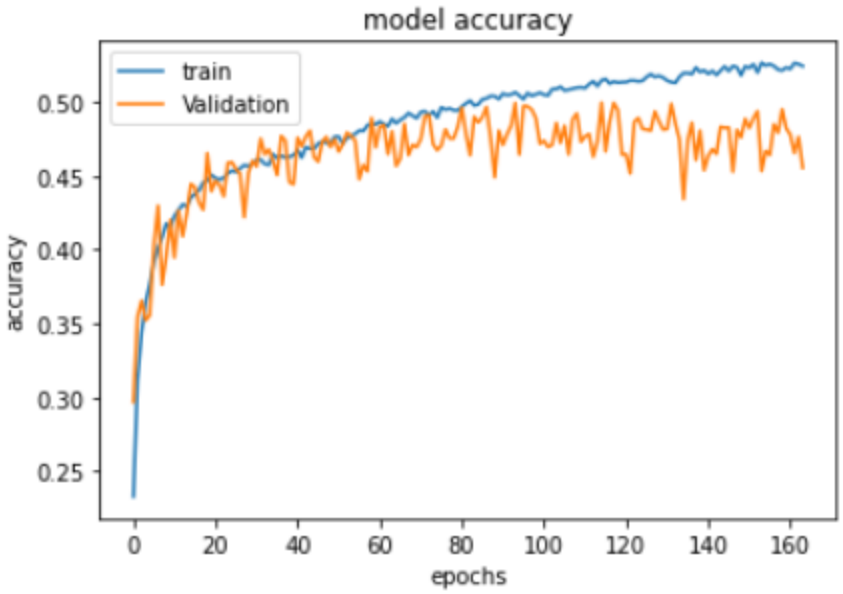
```
#  x_train , x_test  là mảng 4 chiều nên chuyển
sang mảng 2 chiều
x_train = x_train.reshape(50000 , 3072 )#32*32*3
x_test = x_test.reshape(10000 ,  3072 )#32*32*3

# Chuẩn hóa dữ liệu
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /=255
x_test /= 255

# Chuyển y thành 10 class do output là 10
y_train =np_utils.to_categorical(y_train,10)
y_test = np_utils.to_categorical(y_test,10)
```

```
# Tạo mạng neron nhân tạo

model = Sequential()
model.add(Dense(512,activation='relu',input_shap
e=(3072,)))
model.add(Dropout(0.2))
model.add(Dense(256,activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(512,activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(10,activation='softmax'))
model.summary()

# Huấn luyện mô hình
model.compile(loss='categorical_crossentropy',op
timizer=RMSprop(), metrics=['accuracy'])
history = model.fit(x_train,y_train, batch_size=
128, epochs=500 , verbose=1 , validation_split=0
.2 , callbacks=[EarlyStopping(monitor='val_loss'
,patience=70)])

# Lưu kết quả của model
from tensorflow.keras.models import load_model
model.save('huyCifar10.h5')
```

```
load_model('huyCifar10.h5')


# Đánh giá độ chính xác của mô hình
score = model.evaluate(x_test,y_test,verbose=0)
print('Sai số kiểm tra là: ',score[0])
print('Độ chính xác kiểm tra là: ',score[1])
```

```
Sai số kiểm tra là:  1.5730534791946411
Độ chính xác kiểm tra là:  0.4507000148296356
```

```
# Vẽ lại quá trình học
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epochs')
plt.legend(['train','Validation'])
plt.show()
```




```
# Kiểm tra kết quả của mô hình
from keras.preprocessing import image
from keras.applications.vgg16 import preprocess_
input
from tensorflow.keras.utils import load_img, img
_to_array
filename = 'Nai.png'
img = load_img(filename, target_size =(32,32))
img.show(filename)
img = img_to_array(img)
img = img.astype('float32')
img = img/255
img=img.reshape(1,32*32*3)
np.argmax (model.predict(img) , axis =-1)
```

Bài 2: Cifar100

```
#  Gọi các thư viên cần thiết
import numpy as np
import pandas as pd
from keras.datasets import cifar10 , cifar100
import matplotlib.pyplot as plt
from tensorflow.keras.optimizers import RMSprop
from keras.callbacks import EarlyStopping
from keras.utils import np_utils
from keras.backend import dropout
from keras.models import Sequential
from keras.layers import Dense, Dropout


# Chia dữ liệu thành 2 phần:  phần huyến luyện v
à phần test
(x_train, y_train),(x_test,y_test) = cifar100.lo
ad_data()

# Kích thước các tập dữ liệu
x_train.shape , x_test .shape,  y_train.shape,
y_test.shape
```

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

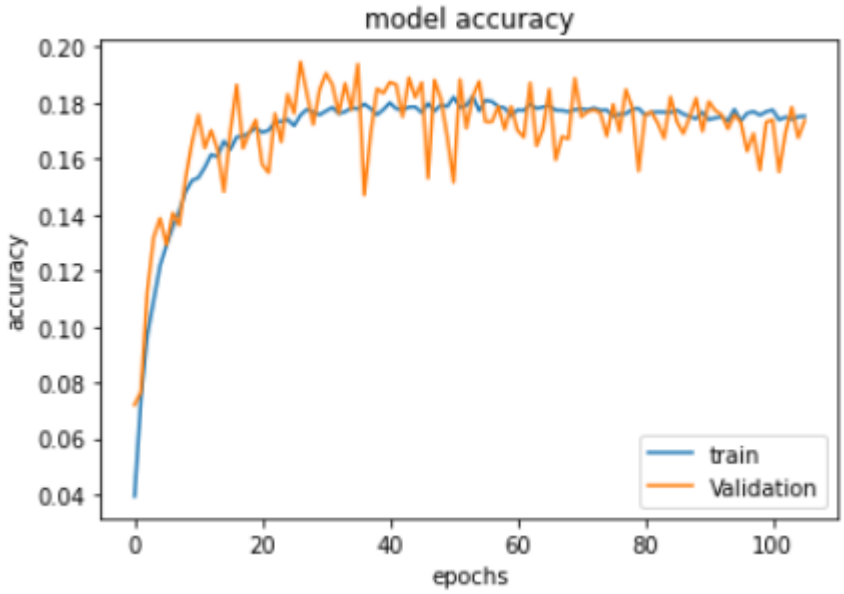
```
#  x_train , x_test  là mảng 4 chiều nên chuyển
sang mảng 2 chiều
x_train = x_train.reshape(50000 , 3072 )  #32*32
*3
x_test = x_test.reshape(10000 ,  3072 )   #32*32
*3

# Chuẩn hóa dữ liệu
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /=255
x_test /= 255

# Chuyển y thành 100 class do output là 100
y_train =np_utils.to_categorical(y_train,100)
y_test = np_utils.to_categorical(y_test,100)


# Tạo mạng neron nhân tạo

model = Sequential()
model.add(Dense(512,activation='relu',input_shap
e=(3072,)))
model.add(Dropout(0.2))
```

```
model.add(Dense(256,activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(512,activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(100,activation='softmax'))
model.summary()

# Huấn luyện mô hình
model.compile(loss='categorical_crossentropy',op
timizer=RMSprop(), metrics=['accuracy'])
history = model.fit(x_train,y_train, batch_size=
128, epochs=500 , verbose=1 , validation_split=0
.2 , callbacks=[EarlyStopping(monitor='val_loss'
,patience=70)])

# Lưu kết quả của model
from tensorflow.keras.models import load_model
model.save('huyCifar100.h5')
load_model('huyCifar100.h5')

# Đánh giá độ chính xác của mô hình
score = model.evaluate(x_test,y_test,verbose=0)
print('Sai số kiểm tra là: ',score[0])
```

```
print('Độ chính xác kiểm tra là: ',score[1])
```

```
Sai số kiểm tra là: 3.5452253818511963
Độ chính xác kiểm tra là: 0.17579999566078186
```

```
# Vẽ lại quá trình học
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epochs')
plt.legend(['train','Validation'])
plt.show()
```



```
# Kiểm tra kết quả của mô hình
from keras.preprocessing import image
from tensorflow.keras.utils import load_img, img
_to_array
filename = 'Nai.png'
img = load_img(filename, target_size =(32,32))
img.show(filename)
img = img_to_array(img)
img = img.astype('float32')
img = img/255
img=img.reshape(1,32*32*3)
np.argmax (model.predict(img) , axis =-1)
```

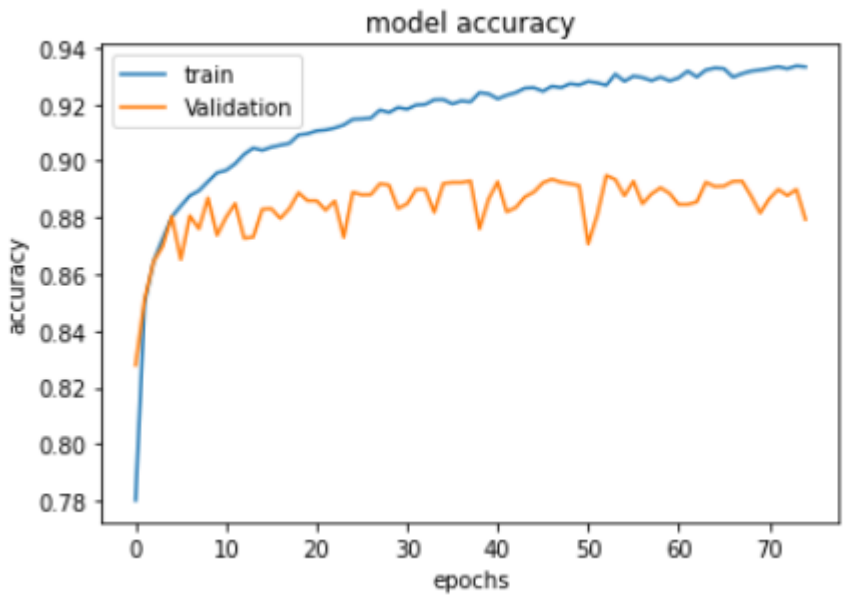
Bài 3: Fashion MNIST

```
#  Gọi các thư viên cần thiết
import numpy as np
import pandas as pd
from keras.datasets import fashion_mnist
import matplotlib.pyplot as plt
from tensorflow.keras.optimizers import RMSprop
from keras.callbacks import EarlyStopping
from keras.utils import np_utils
from keras.backend import dropout
from keras.models import Sequential
from keras.layers import Dense, Dropout

# Chia dữ liệu thành 2 phần:  phần huyến luyện v
à phần test
(x_train, y_train),(x_test,y_test) = fashion_mni
st.load_data()


# Kích thước các tập dữ liệu
x_train.shape , x_test .shape,  y_train.shape,
y_test.shape
```

```
#  x_train , x_test  là mảng 3 chiều nên chuyển
sang mảng 2 chiều
x_train = x_train.reshape(60000 , 784)  #28*28
x_test = x_test.reshape(10000 ,  784 )   #28*28

# Chuẩn hóa dữ liệu
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /=255
x_test /= 255

# Chuyển y thành 100 class do output là 100
y_train =np_utils.to_categorical(y_train,10)
y_test = np_utils.to_categorical(y_test,10)


# Tạo mạng neron nhân tạo
model = Sequential()
model.add(Dense(512, kernel_initializer= 'normal
', activation='relu',input_shape=(784,)))
model.add(Dense(256,activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(512,activation='relu'))
```

```
model.add(Dropout(0.2))

model.add(Dense(10,activation='softmax'))
model.summary()

# Huấn luyện mô hình
model.compile(loss='categorical_crossentropy',op
timizer=RMSprop(), metrics=['accuracy'])
history = model.fit(x_train,y_train, batch_size=
128, epochs=500 , verbose=1 , validation_split=0
.2 , callbacks=[EarlyStopping(monitor='val_loss'
,patience=70)])

# Lưu kết quả của model
from tensorflow.keras.models import load_model
model.save('huyFashion.h5')
load_model('huyFashion.h5')

# Đánh giá độ chính xác của mô hình
score = model.evaluate(x_test,y_test,verbose=0)
print('Sai số kiểm tra là: ',score[0])
print('Độ chính xác kiểm tra là: ',score[1])
```

```
# Vẽ lại quá trình học
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epochs')
plt.legend(['train','Validation'])
plt.show()
```




```
# Kiểm tra kết quả của mô hình
from tensorflow.keras.utils import load_img, img
_to_array
import numpy as np
from google.colab.patches import cv2_imshow
import cv2
# filename = "deplao.png"
```

```
filename = "bag.png"
# filename = "aokhoac.png"
img = cv2.imread(filename, cv2.IMREAD_UNCHANGED)
cv2_imshow(img)
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
cv2_imshow(gray)

dim = (28, 28)
# img_new = load_img(gray,target_size=(32,32))
resized = cv2.resize(gray, dim, interpolation =
cv2.INTER_AREA)

img = img_to_array(resized)
img = img.reshape(1,28*28)
img = img.astype('float32')
img = img/255

np.argmax(model.predict(img),axis=-1)
```

Bài 4: Face detect (Sử dụng CNN)

```
# Gọi các thư viện cần thiết
import numpy as np
import pandas as pd # Xu lý bảng
import seaborn as sns # Vẽ biểu đồ thị của dữ li
ệu
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
 # Xử lý chuẩn hóa dữ liệu
from sklearn.model_selection import train_test_s
plit # Chia dữ liệu ra làm 2 phần
from keras.layers import Dense, Activation, Drop
out, BatchNormalization, LSTM    # LSTM  biên dạ
ng ANN, BatchNormalization: cho nhỏ lại
from keras.models import Sequential
from tensorflow.keras.utils import to_categorica
l # Sử dung để làm nổi đối tượng cần phân loại
from keras import callbacks
from sklearn.metrics import precision_score, rec
all_score, confusion_matrix, classification_repo
rt, accuracy_score, f1_score # Để đo lường

from tensorflow.keras.preprocessing.image import
 ImageDataGenerator
```

```
from keras.utils import np_utils
from tensorflow.keras.preprocessing import image
from tensorflow.keras.optimizers import RMSprop
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.callbacks import EarlyStopping
import matplotlib.pyplot as plt
import tensorflow as tf
import numpy as np
import cv2
import os


image_generator = ImageDataGenerator(rescale=1/2
55, validation_split=0.2)

train_dataset = image_generator.flow_from_direct
ory(batch_size=32,directory='../input/qhuyver2/h
uyver1',shuffle=True,target_size=(32, 32), subse
t="training", class_mode='categorical')

validation_dataset = image_generator.flow_from_d
irectory(batch_size=32,directory='../input/qhuyv
```

```
er2/huyver1',shuffle=True,target_size=(32, 32),
subset="validation",class_mode='categorical')


import glob
Huy = glob.glob('../input/qhuyver2/huyver1/Huy/*
.*')
NoHuy = glob.glob('../input/qhuyver2/huyver1/NoH
uy/*.*')

data = []
labels = []

for i in Huy:
    image=tf.keras.preprocessing.image.load_img(
i, color_mode='rgb', target_size= (32,32))
    image=np.array(image)
    data.append(image)
    labels.append(0)
for j in NoHuy:
    image=tf.keras.preprocessing.image.load_img(
j, color_mode='rgb', target_size= (32,32))
    image=np.array(image)
    data.append(image)
```

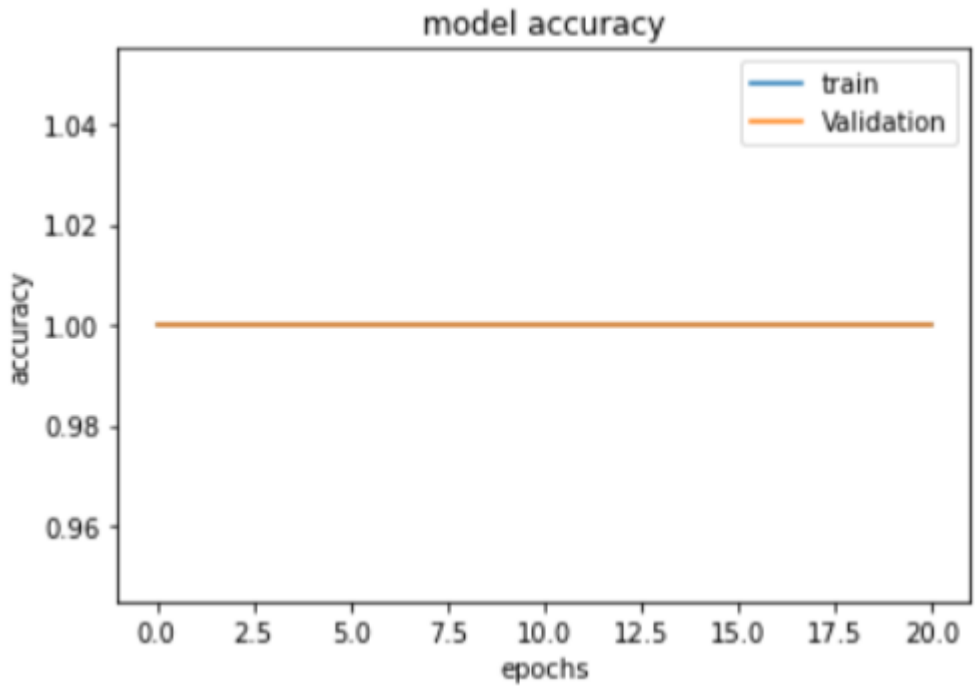
```
    labels.append(0)


data = np.array(data)
labels = np.array(labels)

from sklearn.model_selection import train_test_s
plit
X_train, X_test, y_train, y_test = train_test_sp
lit(data, labels, test_size=0.2, random_state=1)

# Kích thước các tập dữ liệu
X_train.shape, X_test.shape, y_train.shape, y_te
st.shape
```

```
((103, 32, 32, 3), (26, 32, 32, 3), (103,), (26,))
```

```
# Chuẩn hóa dữ liệu
x_train = X_train.reshape(103,3072)
x_test = X_test.reshape(26,3072)
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train/=255
x_test/=255
y_train = np_utils.to_categorical(y_train,2)
```

```
y_test = np_utils.to_categorical(y_test,2)

# Tạo mảng neron nhân tạo

model = Sequential()
model.add(Dense(512,activation = 'relu',input_sh
ape=(3072,)))
model.add(Dropout(0.25))
model.add(Dense(512,activation = 'relu'))
model.add(Dropout(0.25))
model.add(Dense(500,activation = 'relu'))
model.add(Dropout(0.25))
model.add(Dense(512,activation = 'relu'))
model.add(Dropout(0.25))
model.add(Dense(2))
#model.add(Dense(1))
model.summary()

# Huận luyện mô hình
model.compile(loss='categorical_crossentropy', o
ptimizer=RMSprop(), metrics = ['accuracy'])
history = model.fit(x_train,y_train,batch_size=6
4,epochs=100,verbose=1,validation_data=(x_test,y
```

```
_test),callbacks=[EarlyStopping(monitor='val_los
s',patience=20)])

# Lưu kết quả của model
from tensorflow.keras.models import load_model
model.save('huyface.h5')
model = load_model('huyface.h5')

# Đánh giá độ chính xác của mô hình
score = model.evaluate(x_test,y_test,verbose=0)
print('Sai số kiểm tra là: ',score[0])
print('Độ chính xác kiểm tra là: ',score[1])

# Vẽ lại quá trình học
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epochs')
plt.legend(['train','Validation'])
plt.show()
```




```
# Kiểm tra kết quả của mô hình
from tensorflow.keras.utils import load_img, img
_to_array
import numpy as np
filename = "../input/qhuyver2/huyver1/NoHuy/nohu
y (1).jpg"

predict = ["This is Huy","This isn't Huy"]
predict = np.array(predict)

img = load_img(filename,target_size=(32,32))
img = img_to_array(img)
```

```
img = img.reshape(1,32*32*3)
img = img.astype('float32')
img = img/255

result = np.argmax(model.predict(img),axis=-1)
predict[result]
```

Bài 5: Robot 2 bậc tự do

```
#  Gọi các thư viên cần thiết
import numpy as np
import pandas as pd
import math
import seaborn as ses
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_s
plit
from sklearn.metrics import precision_score,reca
ll_score,confusion_matrix,classification_report,
accuracy_score,f1_score

from keras.layers import Dense, Activation, Drop
out, BatchNormalization, LSTM,Conv2D, MaxPooling
2D, Flatten,LeakyReLU
from keras.models import Sequential
from tensorflow.keras.utils import to_categorica
l ,load_img, img_to_array
from tensorflow.keras.optimizers import RMSprop,
SGD
```

```
from tensorflow.keras.models import load_model
from keras import callbacks
from keras.callbacks import EarlyStopping

# Khởi tạo các giá trị của đầu công tác
data_robot2DoF =[]
L1 = 50
L2 = 40

for theta1 in range(-90,90):
  for theta2 in range(-90,90):
    Px = L1*np.cos(np.radians(theta1)) + L2*np.c
os(np.radians(theta1 + theta2))
    Py = L1*np.sin(np.radians(theta1)) + L2*np.s
in(np.radians(theta1 + theta2))
    data_robot2DoF.append([theta1, theta2, Px ,P
y])
data = pd.DataFrame(data_robot2DoF, columns = ['
theta1', 'theta2', 'Px','Py'])
data
```

	theta1	theta2	Px	Py
0	-90	-90	-40.000000	-50.000000
1	-90	-89	-39.993908	-50.698096
2	-90	-88	-39.975633	-51.395980
3	-90	-87	-39.945181	-52.093438
4	-90	-86	-39.902562	-52.790259
...	...	...	...	...
32395	89	85	-38.908255	54.173523
32396	89	86	-38.975168	53.478614
32397	89	87	-39.029942	52.782644
32398	89	88	-39.072561	52.085823
32399	89	89	-39.103013	51.388365

32400 rows × 4 columns

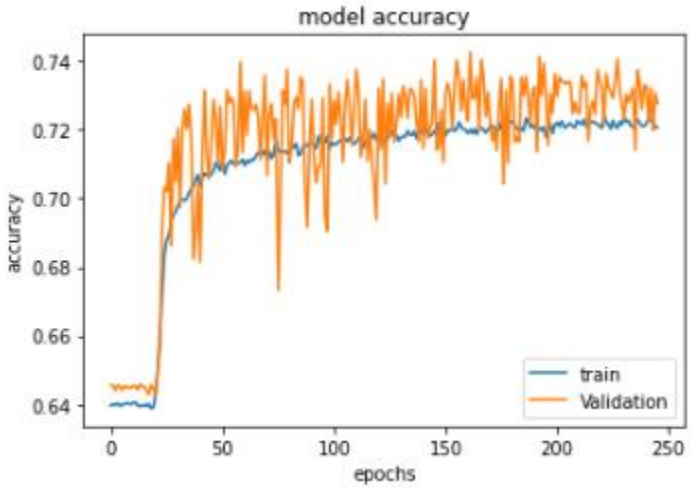
```
X_P=data.drop(['theta1','theta2'],axis =1)
Y_Theta=data.drop(['Px','Py'],axis =1)
print(X_P.shape,Y_Theta.shape)
```

```
(32400, 2)(32400, 2)
```

```
# Chia dữ liệu thành 2 phần:  phần huyến luyện v
à phần test
x_train, x_test, y_train, y_test = train_test_sp
lit(X_P, Y_Theta, test_size=0.2,random_state =7)


# Tạo mạng neron nhân tạo


model = Sequential()
```

```
model.add(Dense(64,kernel_initializer='normal',a
ctivation='relu',input_shape=(2,)))
model.add(Dense(64,activation ='relu'))
model.add(Dense(64,activation ='relu'))
model.add(Dense(2))
model.summary()

# Huấn luyện mô hình
model.compile(loss='mse',optimizer=RMSprop(), me
trics=['accuracy'])
history = model.fit(x_train,y_train, batch_size=
128, epochs=500 , verbose=1 , validation_split=0
.2 , callbacks=[EarlyStopping(monitor='val_loss'
,patience=70)])

# Lưu kết quả của model
from tensorflow.keras.models import load_model
model.save('huy2dof.h5')
load_model('huy2dof.h5')

# Đánh giá độ chính xác của mô hình
score = model.evaluate(x_test,y_test,verbose=0)
print('Sai số kiểm tra là: ',score[0])
```

```
print('Độ chính xác kiểm tra là: ',score[1])


# Vẽ lại quá trình học
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epochs')
plt.legend(['train','Validation'])
plt.show()
```



Bài 6: Robot 3 bậc tự do

```
#  Gọi các thư viện cần thiết
import numpy as np
import pandas as pd
import math
import seaborn as ses
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_score,recall_score,confusion_matrix,
classification_report,accuracy_score,f1_score

from keras.layers import Dense, Activation, Dropout, BatchNormalization, L
STM,Conv2D, MaxPooling2D, Flatten,LeakyReLU
from keras.models import Sequential
from tensorflow.keras.utils import to_categorical ,load_img, img_to_array
from tensorflow.keras.optimizers import RMSprop,SGD
from tensorflow.keras.models import load_model
from keras import callbacks
from keras.callbacks import EarlyStopping


# Khởi tạo các giá trị của đầu công tác
data_robot2DoF =[]
L1 = 50
L2 = 40
L3 = 20
for theta1 in range(-90,90):
  for theta2 in range(-60,60):
    for theta3 in range(-45,45):
      Px = L1*np.cos(np.radians(theta1)) + L2*np.cos(np.radians(theta1 + t
heta2)) +L3*np.cos(np.radians(theta1 + theta2 +theta3))
      Py = L1*np.sin(np.radians(theta1)) + L2*np.sin(np.radians(theta1 + t
heta2)) +L3*np.sin(np.radians(theta1 + theta2+theta3))
      ci = theta1 + theta2 +theta3
      data_robot2DoF.append([theta1, theta2,theta3, Px ,Py,ci])
data = pd.DataFrame(data_robot2DoF, columns = ['theta1', 'theta2','theta3'
, 'Px','Py','ci'])
```

data

	theta1	theta2	theta3	Px	Py	ci
0	-90	-60	-45	-53.959533	-64.823619	-195
1	-90	-60	-44	-54.046931	-65.161562	-194
2	-90	-60	-43	-54.128417	-65.500979	-193
3	-90	-60	-42	-54.203968	-65.841766	-192
4	-90	-60	-41	-54.273560	-66.183820	-191
...	...	...	...	...	...	...
1943995	89	59	40	-52.854665	68.405693	188
1943996	89	59	41	-52.803070	68.060466	189
1943997	89	59	42	-52.745459	67.716192	190
1943998	89	59	43	-52.681847	67.372975	191
1943999	89	59	44	-52.612256	67.030922	192

1944000 rows × 6 columns

```
X_P=data.drop(['theta1','theta2','theta3'],axis =1)
Y_Theta=data.drop(['Px','Py','ci'],axis =1)
print(X_P.shape,Y_Theta.shape)
```

```
# Chia dữ liệu thành 2 phần:  phần huyến luyện và phần test

x_train, x_test, y_train, y_test = train_test_split(X_P, Y_Theta, test_siz
e=0.2,random_state =7)

# Kích thước các tập dữ liệu
x_train.shape , x_test .shape,  y_train.shape,  y_test.shape
```

((1555200, 3), (388800, 3), (1555200, 3), (388800, 3))

```
# Tạo mạng neron nhân tạo

model = Sequential()
model.add(Dense(64,kernel_initializer='normal',activation='relu',input_sha
pe=(3,)))
model.add(Dense(64,activation ='relu'))
model.add(Dense(64,activation ='relu'))
model.add(Dense(3))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 64)                256

dense_1 (Dense)              (None, 64)                4160

dense_2 (Dense)              (None, 64)                4160

dense_3 (Dense)              (None, 3)                 195

=================================================================
Total params: 8,771
Trainable params: 8,771
Non-trainable params: 0
```

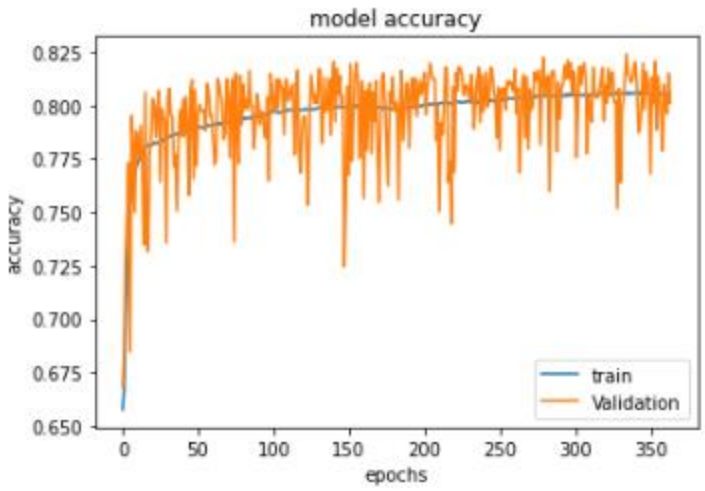
```
# Huấn luyện mô hình
model.compile(loss='mse',optimizer=RMSprop(), metrics=['accuracy'])
history = model.fit(x_train,y_train, batch_size=128, epochs=500 , verbose=
1 , validation_split=0.2 , callbacks=[EarlyStopping(monitor='val_loss',pat
ience=70)])
```

```
# Lưu kết quả của model
from tensorflow.keras.models import load_model
model.save('huy3dof.h5')
load_model('huy3dof.h5')
```

```
# Đánh giá độ chính xác của mô hình
score = model.evaluate(x_test,y_test,verbose=0)
print('Sai số kiểm tra là: ',score[0])
print('Độ chính xác kiểm tra là: ',score[1])
```

```
Sai số kiểm tra là:  173.39004516601562
Độ chính xác kiểm tra là:  0.7991254925727844
```

```
# Vẽ lại quá trình học
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epochs')
plt.legend(['train','Validation'])
plt.show()
```


model accuracy
accuracy
epochs
train
Validation
0.825
0.800
0.775
0.750
0.725
0.700
0.675
0.650
0
50
100
150
200
250
300
350

Bài tập nhóm: Face Member Regconition

```
# Import Libraries

import tensorflow as tf

import matplotlib.pyplot as plt

import cv2 as cv

import os

import numpy as np

from tensorflow.keras.preprocessing.image import
 ImageDataGenerator

from tensorflow.keras.preprocessing import image

from tensorflow.keras.optimizers import RMSprop

from tensorflow.keras import layers

# Gọi các thư viện cần thiết

import numpy as np

import pandas as pd # Xu lý bảng

import seaborn as sns # Vẽ biểu đồ thị của dữ li
ệu

import matplotlib.pyplot as plt
```

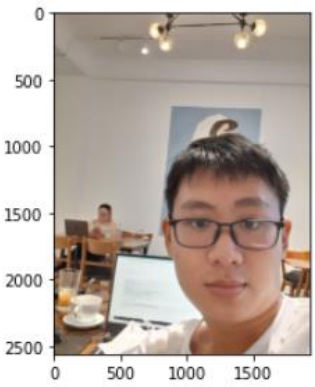
```
from sklearn.preprocessing import StandardScaler
 # Xử lý chuẩn hóa dữ liệu

from sklearn.model_selection import train_test_s
plit # Chia dữ liệu ra làm 2 phần

from keras.layers import Dense, Activation, Drop
out, BatchNormalization, LSTM    # LSTM  biên dạ
ng ANN, BatchNormalization: cho nhỏ lại

from keras.models import Sequential

from tensorflow.keras.utils import to_categorica
l # Sử dung để làm nổi đối tượng cần phân loại

from keras import callbacks

from sklearn.metrics import precision_score, rec
all_score, confusion_matrix, classification_repo
rt, accuracy_score, f1_score # Để đo lường


from tensorflow.keras.preprocessing.image import
 ImageDataGenerator

from keras.utils import np_utils

from tensorflow.keras.preprocessing import image

from tensorflow.keras.optimizers import RMSprop
```

```
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.callbacks import EarlyStopping
import matplotlib.pyplot as plt
import tensorflow as tf
import numpy as np
import cv2
import os

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_s
plit
```

```
from keras.layers import Dense, Activation, Drop
out, BatchNormalization, LSTM

from keras.models import Sequential

from tensorflow.keras.utils import to_categorica
l

from keras import callbacks

from sklearn.metrics import precision_score, rec
all_score, confusion_matrix, classification_repo
rt, accuracy_score, f1_score

import keras

from keras.models import Sequential

from keras.layers import Dense # fully connected

from keras.datasets import boston_housing

from tensorflow.keras.optimizers import RMSprop
# toi uu

from keras.callbacks import EarlyStopping # dung
 lai ngay lap tuc

from sklearn.preprocessing import scale # xu li
du lieu
```

```
from sklearn.preprocessing import StandardScaler
 # xu li du lieu


# Load 1 image
img = image.load_img("../input/traintth1/Tai/1.png")
plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7
```




```
image_generator = ImageDataGenerator(rescale=1/255, validation_split=0.2)


train_dataset = image_generator.flow_from_directory(batch_size=32,
```

```
 directory="../input/traintth1/",

 shuffle=True,

 target_size=(150, 150),

 subset="training",

 class_mode='categorical')

validation_dataset = image_generator.flow_from_d
irectory(batch_size=32,

 directory="../input/traintth1/",

 shuffle=True,

 target_size=(150, 150),

 subset="validation",
```

```
 class_mode='categorical')
```

```
Found 143 images belonging to 3 classes.
Found 35 images belonging to 3 classes.
```

```
train_dataset.class_indices
```

```
{'Huy': 0, 'Tai': 1, 'Tuan': 2}
```

```
# Create model
from keras.layers import Conv2D, MaxPooling2D
model = Sequential()
model.add(Conv2D(32,(3,3), activation='relu',ker
nel_initializer='he_uniform',padding='same',inpu
t_shape=(150,150,3)))
model.add(Conv2D(32,(3,3), activation='relu',ker
nel_initializer='he_uniform',padding='same'))
model.add(MaxPooling2D(2,2))

model.add(Conv2D(64,(3,3), activation='relu',ker
nel_initializer='he_uniform',padding='same')) #
64 lan tich chap
```

```
model.add(Conv2D(64,(3,3), activation='relu',ker
nel_initializer='he_uniform',padding='same'))
model.add(MaxPooling2D(2,2))


model.add(Conv2D(128,(3,3), activation='relu',ke
rnel_initializer='he_uniform',padding='same')) #
 128 lan tich chap
model.add(Conv2D(128,(3,3), activation='relu',ke
rnel_initializer='he_uniform',padding='same'))
model.add(MaxPooling2D(2,2))


# model.add(Conv2D(256,(3,3), activation='relu',
kernel_initializer='he_uniform',padding='same'))
 # 128 lan tich chap
# model.add(Conv2D(256,(3,3), activation='relu',
kernel_initializer='he_uniform',padding='same'))
# model.add(MaxPooling2D(2,2))
from keras.layers import Dense, Activation, Flat
ten
model.add(Flatten())
```

```
model.add(Dense(128, activation = 'relu', kernel
_initializer='he_uniform'))

model.add(Dense(3))

model.summary()
```

Layer (type)	Output Shape	Param #
conv2d_115 (Conv2D)	(None, 150, 150, 32)	896
conv2d_116 (Conv2D)	(None, 150, 150, 32)	9248
max_pooling2d_57 (MaxPooling	(None, 75, 75, 32)	0
conv2d_117 (Conv2D)	(None, 75, 75, 64)	18496
conv2d_118 (Conv2D)	(None, 75, 75, 64)	36928
max_pooling2d_58 (MaxPooling	(None, 37, 37, 64)	0
conv2d_119 (Conv2D)	(None, 37, 37, 128)	73856
conv2d_120 (Conv2D)	(None, 37, 37, 128)	147584
max_pooling2d_59 (MaxPooling	(None, 18, 18, 128)	0
flatten_17 (Flatten)	(None, 41472)	0
dense_34 (Dense)	(None, 128)	5308544
dense_35 (Dense)	(None, 3)	387

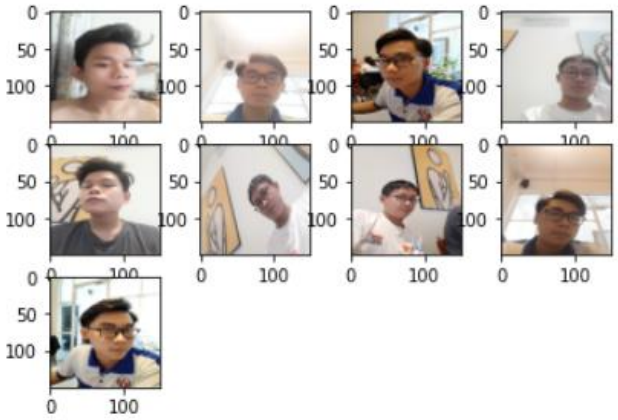
Total params: 5,595,939
Trainable params: 5,595,939
Non-trainable params: 0

```
# Compile

model.compile(loss='mse',optimizer=RMSprop(),met
rics=['accuracy'])

# Train model

history=model.fit(train_dataset,batch_size=100,e
pochs=10,validation_data=validation_dataset)
```

```
Epoch 1/10
5/5 [==============================] - 17s 3s/step - loss: 2634.1877 - accuracy: 0.3217 - val_loss: 0.2621 - val_accuracy: 0.3429
Epoch 2/10
5/5 [==============================] - 14s 3s/step - loss: 0.2744 - accuracy: 0.3916 - val_loss: 0.3605 - val_accuracy: 0.3143
Epoch 3/10
5/5 [==============================] - 14s 3s/step - loss: 0.2313 - accuracy: 0.4965 - val_loss: 0.1880 - val_accuracy: 0.3429
Epoch 4/10
5/5 [==============================] - 14s 3s/step - loss: 0.1276 - accuracy: 0.7343 - val_loss: 0.1122 - val_accuracy: 0.8857
Epoch 5/10
5/5 [==============================] - 14s 3s/step - loss: 0.0924 - accuracy: 0.9161 - val_loss: 0.1753 - val_accuracy: 0.6571
Epoch 6/10
5/5 [==============================] - 14s 3s/step - loss: 0.1725 - accuracy: 0.7203 - val_loss: 0.0853 - val_accuracy: 0.8571
Epoch 7/10
5/5 [==============================] - 14s 3s/step - loss: 0.0846 - accuracy: 0.9790 - val_loss: 0.0649 - val_accuracy: 0.9143
Epoch 8/10
5/5 [==============================] - 14s 3s/step - loss: 0.0343 - accuracy: 0.9860 - val_loss: 0.0600 - val_accuracy: 0.9429
Epoch 9/10
5/5 [==============================] - 14s 3s/step - loss: 0.0725 - accuracy: 0.9930 - val_loss: 0.0685 - val_accuracy: 0.9143
Epoch 10/10
5/5 [==============================] - 14s 3s/step - loss: 0.0577 - accuracy: 0.9650 - val_loss: 0.0552 - val_accuracy: 0.9143
```

```
for i in range(10,19):
  plt.subplot(330+i+1)
  plt.imshow(X_train[i])
```

```
plt.show()
```



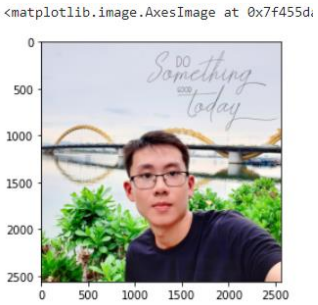
```
# Save model
from tensorflow.keras.models import load_model
model.save('Final.h5')
model_ANN = load_model('Final.h5')

# Check accuracy
from tensorflow.keras.utils import load_img, img
_to_array
import numpy as np
filename = "../input/taistest/tai1.png"
```

```
predict = ['QuangHuy-19146195','ĐứcTài-
19146255','PMinhTuan-19146297']

predict = np.array(predict)


img = load_img(filename,target_size=(150,150))

img = img_to_array(img)

img = img.reshape(1,150,150,3)

img = img.astype('float32')

img = img/255


result = np.argmax(model_ANN.predict(img),axis=-
1)

predict[result]
```

```
array(['ĐứcTài-19146255'], dtype='<U18')
```

```
# See input

from tensorflow.keras.utils import load_img, img
_to_array
```

```
img = image.load_img("../input/taistest/tai1.png
")

plt.imshow(img)
```




```
# Draw plot

plt.plot(history.history['accuracy'])

plt.plot(history.history['val_accuracy'])

plt.title('model accuracy')

plt.ylabel('accuracy')

plt.xlabel('epochs')

plt.legend(['train','Validation'])

plt.show()
```