TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH

KHOA CƠ KHÍ CHẾ TẠO MÁY

BỘ MÔN CƠ ĐIỆN TỬ

HCMUTE

BÀI TẬP

TRÍ TUỆ NHÂN TẠO

Họ và Tên: Nguyễn Trọng Nhân

Mã số Sinh viên: 19146065

Thành phố Hồ Chí Minh, tháng 4 năm 2022

I. BÀI TẬP TIÊN ĐOÁN VỊ TRÍ CÁNH TAY ROBOT 2 BẬC TỰ DO

1. Code Tổng Quan

1.1 Tạo Data bằng cách sử dụng Matlab

Dựa vào phân tích động học Robot, tìm được động học thuận của Robot như sau:

P_x	$l_1 cos\theta_1 + l_2 cos(\theta_1 + \theta_2)$
P_y	$l_1 sin\theta_1 + l_2 sin(\theta_1 + \theta_2)$
φ	$\theta_1 + \theta_2$

Tạo hai vòng lặp for để tạo ra Data các trường hợp của vị trí trong các góc theta:

```
l1=50;l2=40

syms t1 t2

A=[]

for t1=0:10:180

    for t2=0:10:180

        Px=l1*cos(t1*(pi/180))+l2*cos((t1+t2)*(pi/180))

        Py=l1*sin(t1*(pi/180))+l2*sin((t1+t2)*(pi/180))

        phi=t1+t2;

        A=[A;t1 t2 Px Py phi];

        if(phi>360)

         phi=phi-floor(phi/360)*360

        end

    end

end
```

Lưu Data thành file *csv để upload lên Colab

Đặt tên file là "**Data_Arm_2_dofs.csv**"

1.2 Upload lên Colab

```
#import data from PC
from google.colab import files
uploaded=files.upload()
```

1.3 Import các thư viện cần thiết

```
import keras
from keras.datasets import boston_housing
from tensorflow.keras.optimizers import RMSprop  # tính sai số.
from keras.callbacks import EarlyStopping  # Dừng nhanh, khi đạt 1 giá trị nào đó thì dừng xử lý.
from sklearn import preprocessing
from sklearn.preprocessing import scale, StandardScaler
from keras.models import Sequential
from keras.layers import Dense, Activation
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import numpy as np
import pandas as pd
```

1.4 Lấy Data từ Colab

```
#Get data from colab
url ='Data_Arm_2_dofs.csv'
dataframe=pd.read_csv(url)
```

1.5 Chia Data thành các cột riêng biệt và ghép vào biến mảng

```
#Separate data into different column
theta=dataframe.drop(['px','py'], axis=1)
position=dataframe.drop(['theta1','theta2'], axis=1)
theta_train,theta_test,position_train,position_test=train_test_split(theta,position,test_size=0.2)
theta=theta.astype('float32')
```

1.6 Tạo Model để thực hiện Training

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

1.7 Complie, Training và kiểm tra

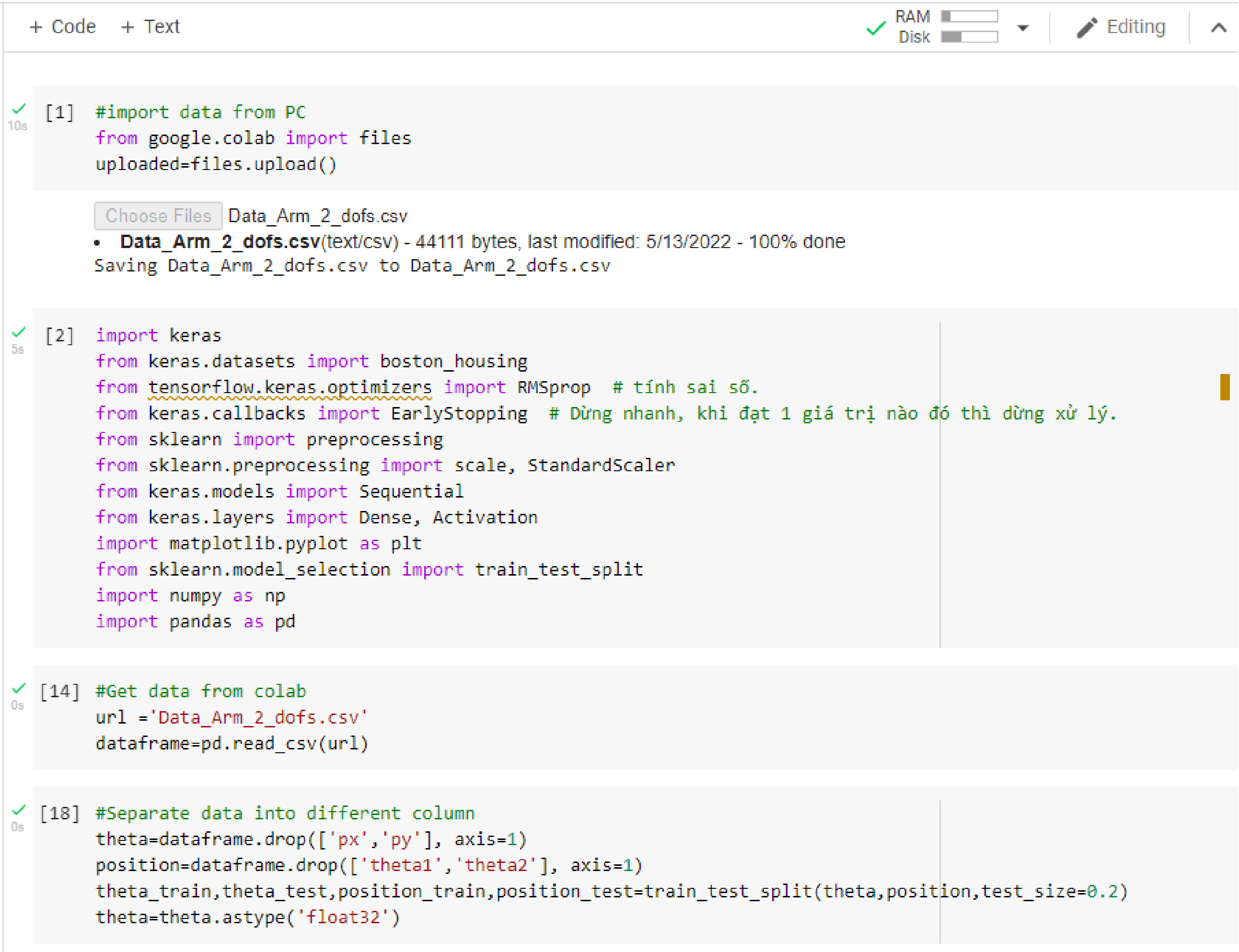
```
#Compile, Training and Checking
model.compile(loss='mae', optimizer=RMSprop(), metrics=['accuracy'])
history=model.fit(theta,position,batch_size=128, epochs=1000, verbose=1, validation_split=0.2, callbacks=[EarlyStopping(monitor='val_loss', patience=20)])
score = model.evaluate(theta,position, verbose=0)

print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
ylim=(0,1)
plt.plot(history.history['accuracy'])
plt.xlabel('epoch')

plt.legend(['accuracy'])
plt.show()
```

1.8 Tạo giá trị Tiên Đoán và So sánh

```
#Prediction and Result
theta_test=np.array(theta_test)
print(theta_test[720])
pos_predict = model.predict(theta_test[720].reshape(1,2))
print("Position Predicted: ",pos_predict)
position_test=np.array(position_test)
print("Real Position: ",position_test[720])
```

2. Code trên Colab



```
[1] #import data from PC
    from google.colab import files
    uploaded=files.upload()
```

Choose Files Data_Arm_2_dofs.csv

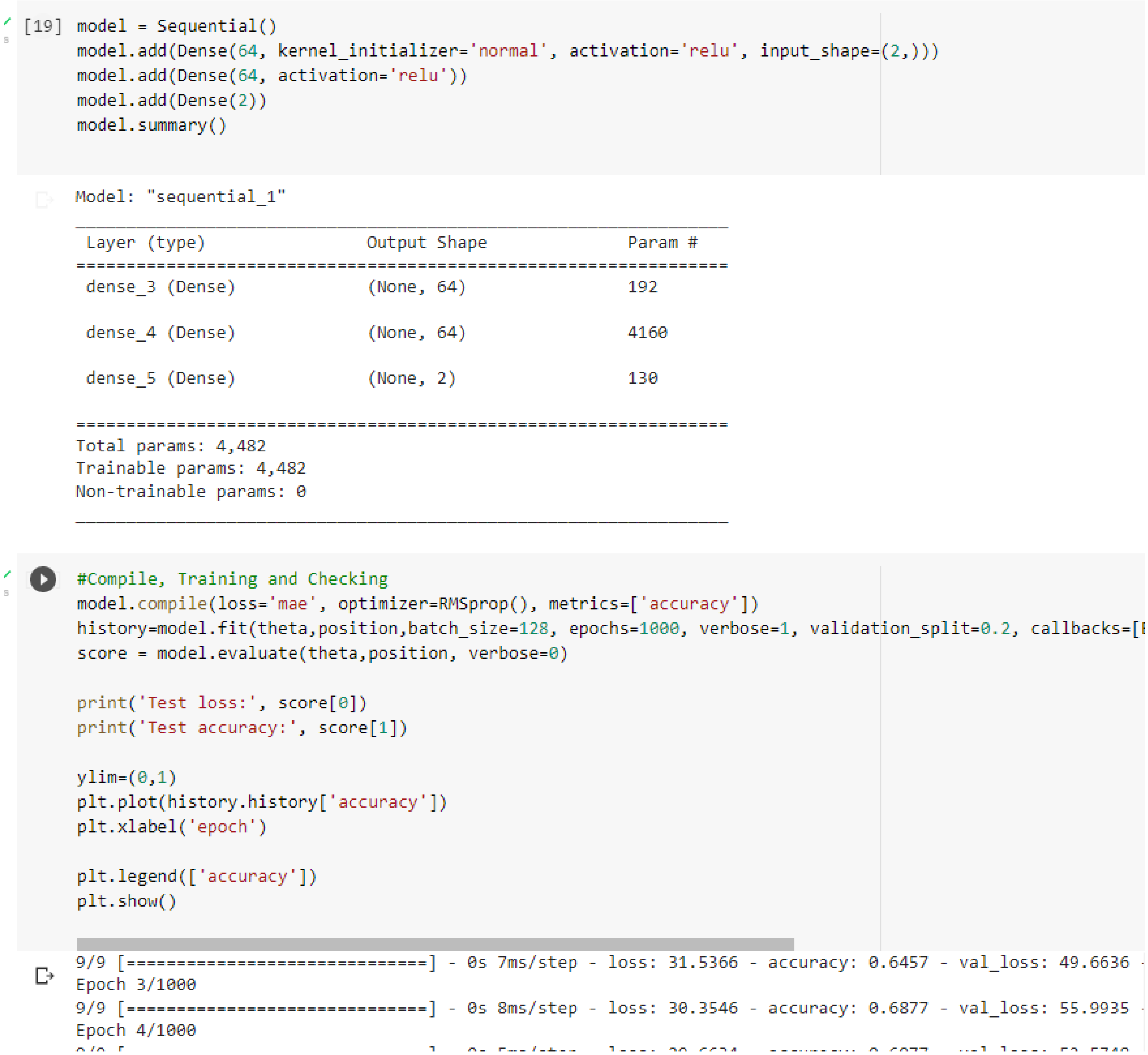
- **Data_Arm_2_dofs.csv**(text/csv) - 44111 bytes, last modified: 5/13/2022 - 100% done

```
Saving Data_Arm_2_dofs.csv to Data_Arm_2_dofs.csv
```

```
[2] import keras
    from keras.datasets import boston_housing
    from tensorflow.keras.optimizers import RMSprop  # tính sai số.
    from keras.callbacks import EarlyStopping  # Dừng nhanh, khi đạt 1 giá trị nào đó thì dừng xử lý.
    from sklearn import preprocessing
    from sklearn.preprocessing import scale, StandardScaler
    from keras.models import Sequential
    from keras.layers import Dense, Activation
    import matplotlib.pyplot as plt
    from sklearn.model_selection import train_test_split
    import numpy as np
    import pandas as pd
```

```
[14] #Get data from colab
     url ='Data_Arm_2_dofs.csv'
     dataframe=pd.read_csv(url)
```

```
[18] #Separate data into different column
     theta=dataframe.drop(['px','py'], axis=1)
     position=dataframe.drop(['theta1','theta2'], axis=1)
     theta_train,theta_test,position_train,position_test=train_test_split(theta,position,test_size=0.2)
     theta=theta.astype('float32')
```

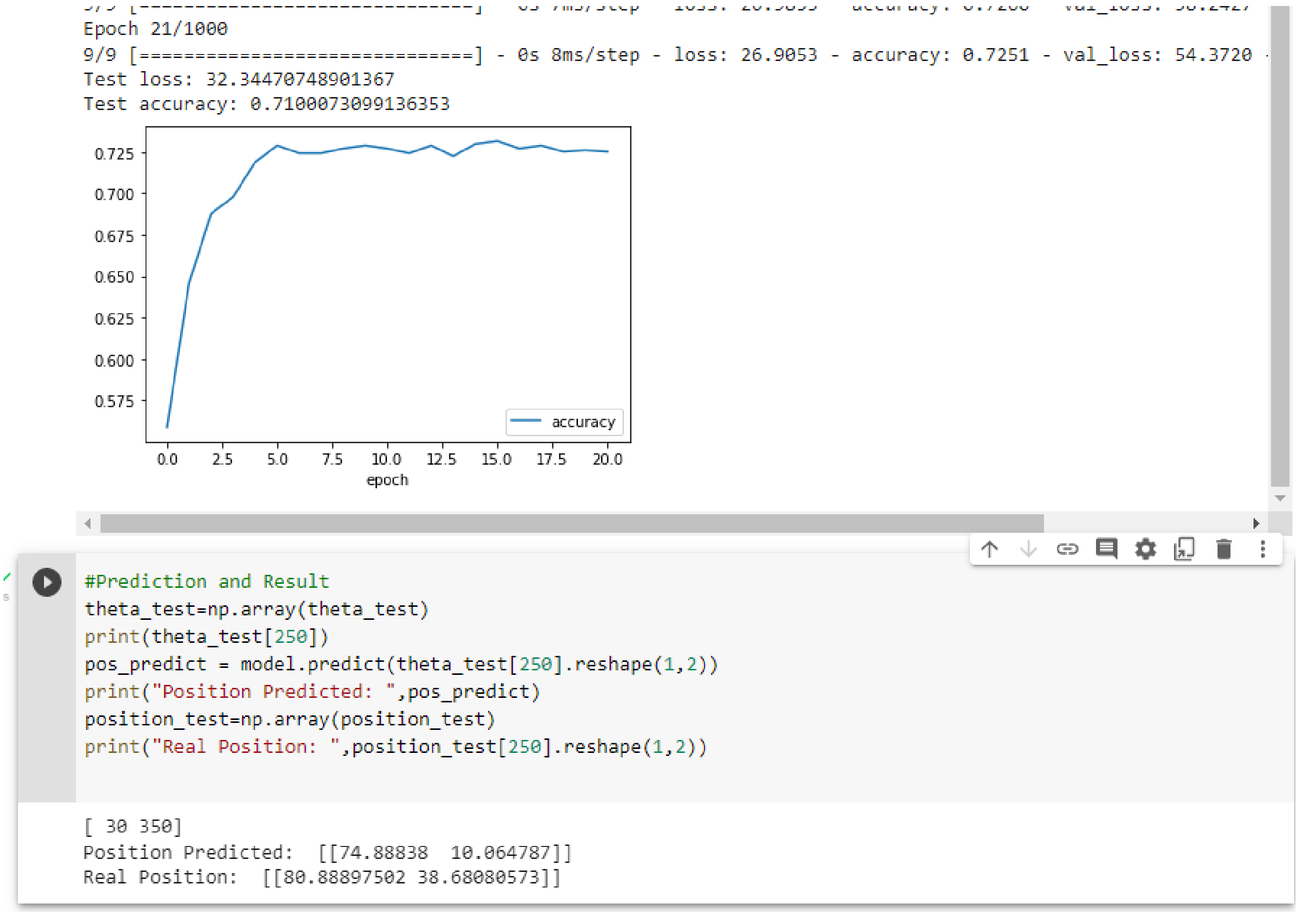


```
[19] model = Sequential()
     model.add(Dense(64, kernel_initializer='normal', activation='relu', input_shape=(2,)))
     model.add(Dense(64, activation='relu'))
     model.add(Dense(2))
     model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_3 (Dense)             (None, 64)                192

 dense_4 (Dense)             (None, 64)                4160

 dense_5 (Dense)             (None, 2)                 130

=================================================================
Total params: 4,482
Trainable params: 4,482
Non-trainable params: 0
_________________________________________________________________
```

```
#Compile, Training and Checking
model.compile(loss='mae', optimizer=RMSprop(), metrics=['accuracy'])
history=model.fit(theta,position,batch_size=128, epochs=1000, verbose=1, validation_split=0.2, callbacks=[E
score = model.evaluate(theta,position, verbose=0)

print('Test loss:', score[0])
print('Test accuracy:', score[1])

ylim=(0,1)
plt.plot(history.history['accuracy'])
plt.xlabel('epoch')

plt.legend(['accuracy'])
plt.show()
```

```
9/9 [==============================] - 0s 7ms/step - loss: 31.5366 - accuracy: 0.6457 - val_loss: 49.6636 -
Epoch 3/1000
9/9 [==============================] - 0s 8ms/step - loss: 30.3546 - accuracy: 0.6877 - val_loss: 55.9935 -
Epoch 4/1000
9/9 [==============================] - 0s 5ms/step - loss: 29.6624 - accuracy: 0.6877 - val_loss: 53.5748 -
```



```
9/9 [==============================] - 0s 7ms/step - loss: 26.9899 - accuracy: 0.7266 - val_loss: 58.2427
Epoch 21/1000
9/9 [==============================] - 0s 8ms/step - loss: 26.9053 - accuracy: 0.7251 - val_loss: 54.3720 -
Test loss: 32.34470748901367
Test accuracy: 0.7100073099136353
```

```
#Prediction and Result
theta_test=np.array(theta_test)
print(theta_test[250])
pos_predict = model.predict(theta_test[250].reshape(1,2))
print("Position Predicted: ",pos_predict)
position_test=np.array(position_test)
print("Real Position: ",position_test[250].reshape(1,2))
```

```
[ 30 350]
Position Predicted:  [[74.88838  10.064787]]
Real Position:  [[80.88897502 38.68080573]]
```

II. BÀI TẬP TIÊN ĐOÁN VỊ TRÍ CÁNH TAY ROBOT 3 BẬC TỰ DO

1. Code Tổng Quan

1.1 Tạo Data bằng cách sử dụng Matlab

Dựa vào phân tích động học Robot, tìm được động học thuận của Robot như sau:

P_x	$l_1 cos\theta_1 + l_2 cos(\theta_1 + \theta_2) + l_3 \cos((\theta_1 + \theta_2 + \theta_3)$
P_y	$l_1 sin\theta_1 + l_2 sin(\theta_1 + \theta_2) + l_3 \sin((\theta_1 + \theta_2 + \theta_3)$
φ	$\theta_1 + \theta_2 + \theta_3$

Tạo ba vòng lặp for để tạo ra Data các trường hợp của vị trí trong các góc theta:

```
l1=50;l2=40;l3=20
syms t1 t2 t3
A=[]
for t1=0:10:180
    for t2=0:10:180
            for t3=0:10:180
                    px=l1*cos(t1*(pi/180))+l2*cos((t1+t2)*(pi/180))+l3*cos((t1+t2+t3)*(pi/180))
                    py=l1*sin(t1*(pi/180))+l2*sin((t1+t2)*(pi/180))+l3*sin((t1+t2+t3)*(pi/180))
                     phi=t1+t2+t3;
                    A=[A;t1 t2 t3 px py phi];
                     if(phi>360)
                            phi=phi-floor(phi/360)*360
                            end
                    end
            end
    end
```

Lưu Data thành file *csv để upload lên Colab

Đặt tên file là “**Data_Arm_3_dofs.csv**”

1.2 Upload lên Colab

```
#import data from PC
from google.colab import files
uploaded=files.upload()
```

1.3 Import các thư viện cần thiết

```
import keras
from keras.datasets import boston_housing
from tensorflow.keras.optimizers import RMSprop  # tính sai số.
from keras.callbacks import EarlyStopping  # Dừng nhanh, khi đạt
1 giá trị nào đó thì dừng xử lý.
from sklearn import preprocessing
from sklearn.preprocessing import scale, StandardScaler
from keras.models import Sequential
from keras.layers import Dense, Activation
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import numpy as np
import pandas as pd
```

1.4 Lấy Data từ Colab

```
#Get data from colab
url ='Data_Arm_3_dofs.csv'
dataframe=pd.read_csv(url)
```

1.5 Chia Data thành các cột riêng biệt và ghép vào biến mảng

```
#Separate data into different column
theta=dataframe.drop(['px','py','phi'], axis=1)
position=dataframe.drop(['theta1','theta2','theta3'], axis=1)
theta_train,theta_test,position_train,position_test=train_test_sp
lit(theta,position,test_size=0.2)
theta=theta.astype('float32')
```

1.6 Tạo Model để thực hiện Training

```
#Create model, training
model = Sequential()
model.add(Dense(64, kernel_initializer='normal', activation='relu
', input_shape=(3,)))
model.add(Dense(64, activation='relu'))
model.add(Dense(3))
model.summary()
```

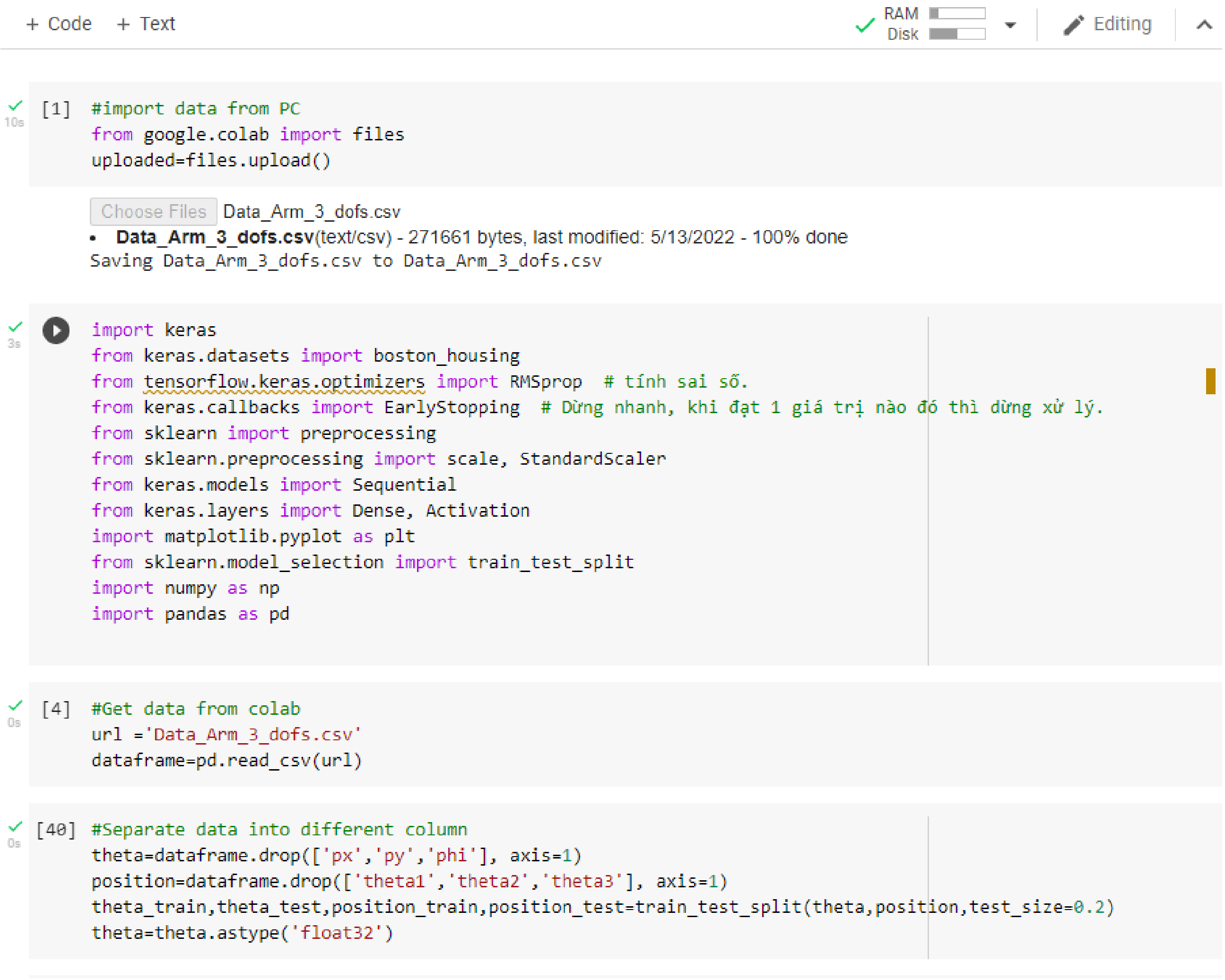
1.7 Complie, Training và kiểm tra

```
#Compile, Training and Checking
model.compile(loss='mae', optimizer=RMSprop(), metrics=['accuracy
'])
history=model.fit(theta,position,batch_size=256, epochs=1000, ver
bose=1, validation_split=0.2, callbacks=[EarlyStopping(monitor='v
al_loss', patience=20)])
score = model.evaluate(theta,position, verbose=0)

print('Test loss:', score[0])
print('Test accuracy:', score[1])

ylim=(0,1)
plt.plot(history.history['accuracy'])
plt.xlabel('epoch')

plt.legend(['accuracy'])
plt.show()
```

1.8 Tạo giá trị Tiên Đoán và So sánh

```
#Prediction and Result
theta_test=np.array(theta_test)
print(theta_test[720])
pos_predict = model.predict(theta_test[720].reshape(1,3))
print("Position Predicted: ",pos_predict)
position_test=np.array(position_test)
print("Real Position: ",position_test[720])
```

2. Code trên Colab



[1]
```
#import data from PC
from google.colab import files
uploaded=files.upload()
```

Data_Arm_3_dofs.csv

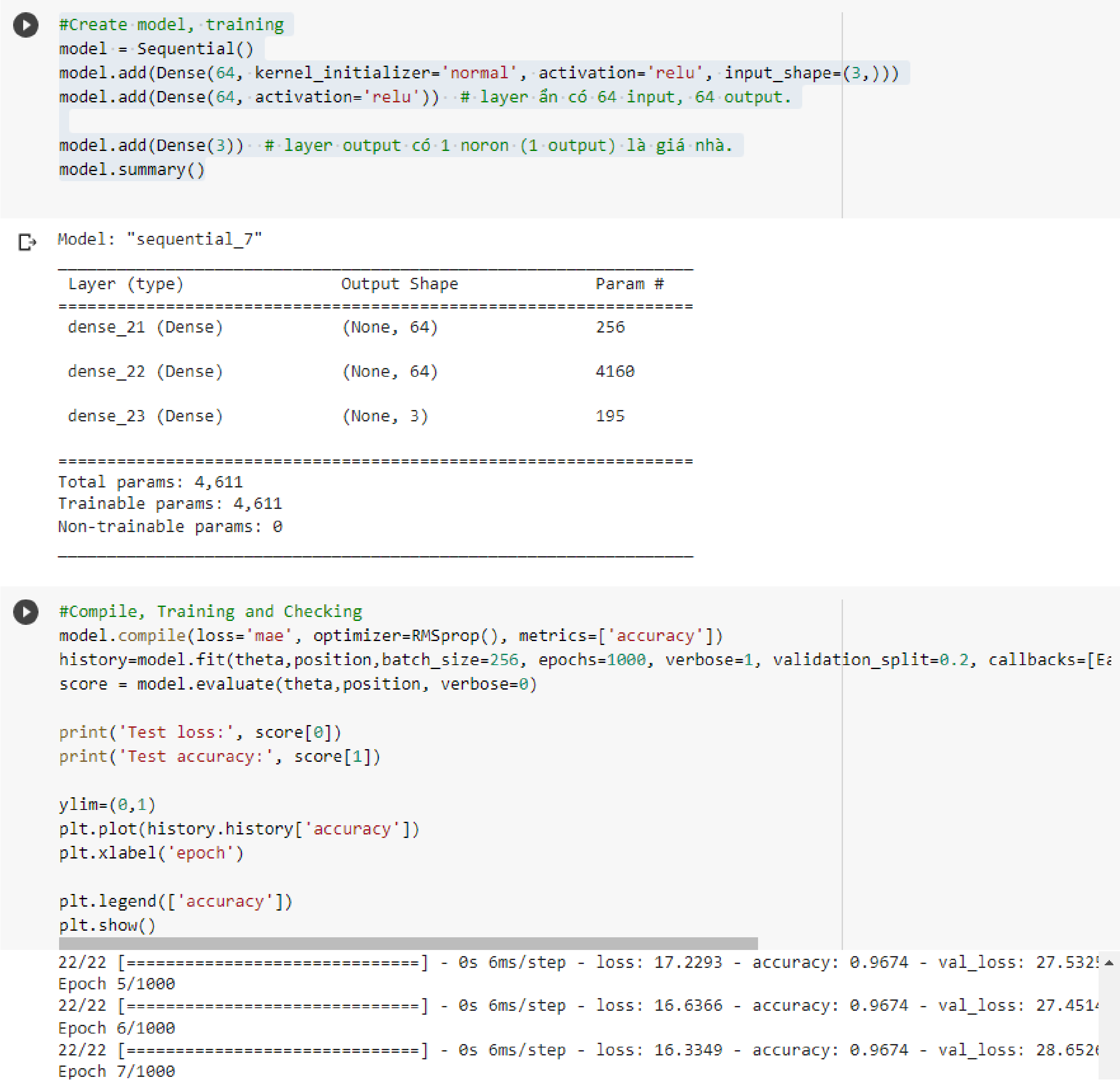
- **Data_Arm_3_dofs.csv**(text/csv) - 271661 bytes, last modified: 5/13/2022 - 100% done

```
Saving Data_Arm_3_dofs.csv to Data_Arm_3_dofs.csv
```

```
import keras
from keras.datasets import boston_housing
from tensorflow.keras.optimizers import RMSprop  # tính sai số.
from keras.callbacks import EarlyStopping  # Dừng nhanh, khi đạt 1 giá trị nào đó thì dừng xử lý.
from sklearn import preprocessing
from sklearn.preprocessing import scale, StandardScaler
from keras.models import Sequential
from keras.layers import Dense, Activation
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import numpy as np
import pandas as pd
```

[4]
```
#Get data from colab
url ='Data_Arm_3_dofs.csv'
dataframe=pd.read_csv(url)
```

[40]
```
#Separate data into different column
theta=dataframe.drop(['px','py','phi'], axis=1)
position=dataframe.drop(['theta1','theta2','theta3'], axis=1)
theta_train,theta_test,position_train,position_test=train_test_split(theta,position,test_size=0.2)
theta=theta.astype('float32')
```

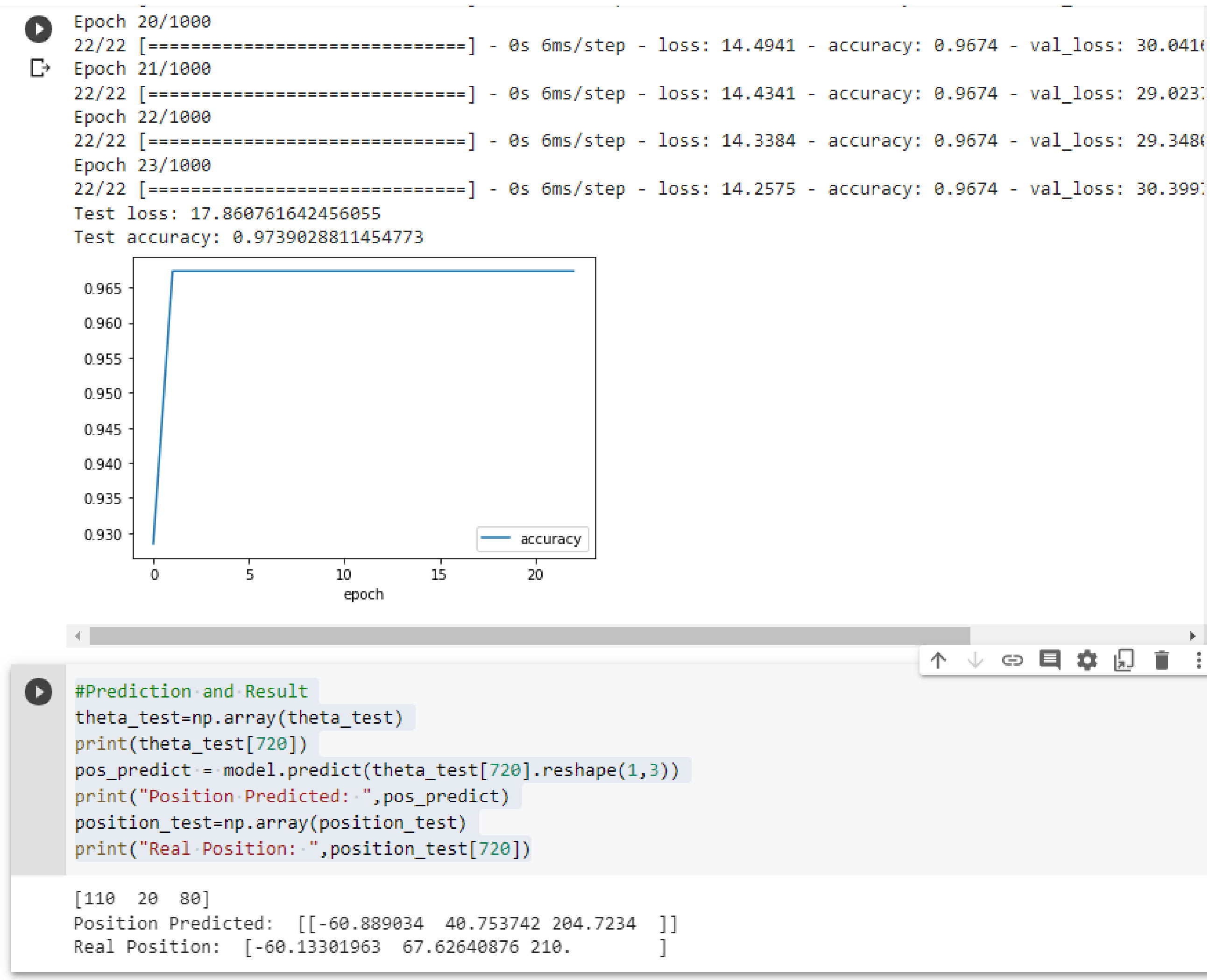


```python
#Create model, training
model = Sequential()
model.add(Dense(64, kernel_initializer='normal', activation='relu', input_shape=(3,)))
model.add(Dense(64, activation='relu'))  # layer ẩn có 64 input, 64 output.

model.add(Dense(3))  # layer output có 1 noron (1 output) là giá nhà.
model.summary()
```

```
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense_21 (Dense)            (None, 64)                256       
                                                                 
 dense_22 (Dense)            (None, 64)                4160      
                                                                 
 dense_23 (Dense)            (None, 3)                 195       
                                                                 
=================================================================
Total params: 4,611
Trainable params: 4,611
Non-trainable params: 0
_________________________________________________________________
```

```python
#Compile, Training and Checking
model.compile(loss='mae', optimizer=RMSprop(), metrics=['accuracy'])
history=model.fit(theta,position,batch_size=256, epochs=1000, verbose=1, validation_split=0.2, callbacks=[Ea
score = model.evaluate(theta,position, verbose=0)

print('Test loss:', score[0])
print('Test accuracy:', score[1])

ylim=(0,1)
plt.plot(history.history['accuracy'])
plt.xlabel('epoch')

plt.legend(['accuracy'])
plt.show()
```

```
22/22 [==============================] - 0s 6ms/step - loss: 17.2293 - accuracy: 0.9674 - val_loss: 27.532
Epoch 5/1000
22/22 [==============================] - 0s 6ms/step - loss: 16.6366 - accuracy: 0.9674 - val_loss: 27.451
Epoch 6/1000
22/22 [==============================] - 0s 6ms/step - loss: 16.3349 - accuracy: 0.9674 - val_loss: 28.652
Epoch 7/1000
```



```
Epoch 20/1000
22/22 [==============================] - 0s 6ms/step - loss: 14.4941 - accuracy: 0.9674 - val_loss: 30.041
Epoch 21/1000
22/22 [==============================] - 0s 6ms/step - loss: 14.4341 - accuracy: 0.9674 - val_loss: 29.023
Epoch 22/1000
22/22 [==============================] - 0s 6ms/step - loss: 14.3384 - accuracy: 0.9674 - val_loss: 29.348
Epoch 23/1000
22/22 [==============================] - 0s 6ms/step - loss: 14.2575 - accuracy: 0.9674 - val_loss: 30.399
Test loss: 17.860761642456055
Test accuracy: 0.9739028811454773
```

```
#Prediction and Result
theta_test=np.array(theta_test)
print(theta_test[720])
pos_predict = model.predict(theta_test[720].reshape(1,3))
print("Position Predicted: ",pos_predict)
position_test=np.array(position_test)
print("Real Position: ",position_test[720])
```

```
[110  20  80]
Position Predicted:  [[-60.889034  40.753742 204.7234  ]]
Real Position:  [-60.13301963  67.62640876 210.        ]
```

III. GITHUB UPLOAD

https://github.com/nhanguyene/HOMEWORK_ARTIFICIAL_INTELLIGIENT