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

KHOA CƠ KHÍ CHẾ TẠO MÁY

BỘ MÔN CƠ ĐIỆN TỬ

BÀI TẬP

TRÍ TUỆ NHÂN TẠO

Họ và Tên: Nguyễn Trọng Nhân

Mã số Sinh viên: 19146065

Lớp Trí Tuệ Nhân Tạo Sáng Thứ 7

Thành phố Hồ Chí Minh, tháng 5 năm 2022

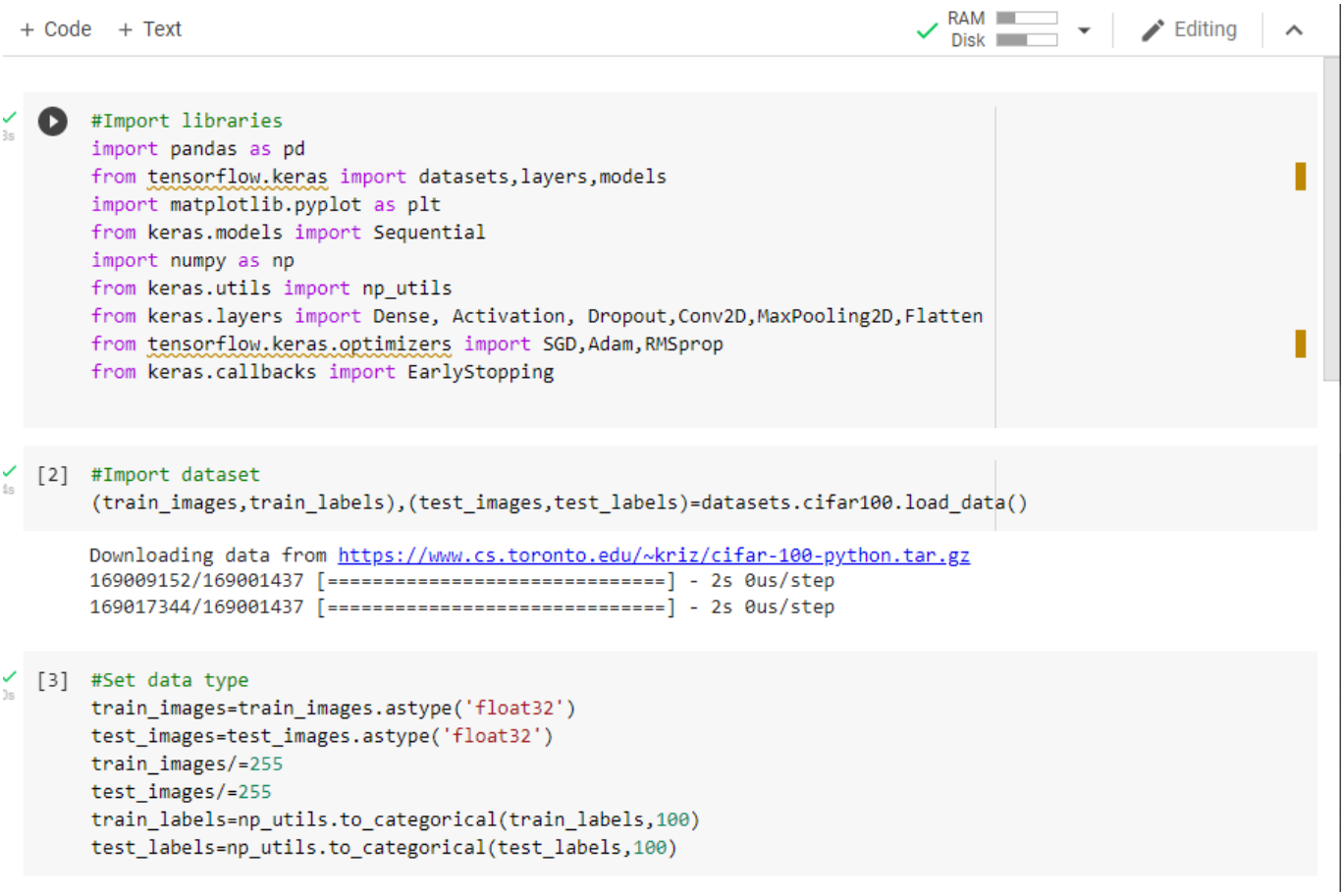
I. SỬ DỤNG CIFAR100 NHẬN DIỆN TỪ DATASET CÓ SẴN

1. Code Tổng Quan

```
#Import libraries
import pandas as pd
from tensorflow.keras import datasets,layers,models
import matplotlib.pyplot as plt
from keras.models import Sequential
import numpy as np
from keras.utils import np_utils
from keras.layers import Dense, Activation, Dropout,Conv2D,MaxPooling2D,Flat
ten
from tensorflow.keras.optimizers import SGD,Adam,RMSprop
from keras.callbacks import EarlyStopping
#Import dataset
(train_images,train_labels),(test_images,test_labels)=datasets.cifar100.load
_data()
#Set data type
train_images=train_images.astype('float32')
test_images=test_images.astype('float32')
train_images/=255
test_images/=255
train_labels=np_utils.to_categorical(train_labels,100)
test_labels=np_utils.to_categorical(test_labels,100)
#Create model for train
model = Sequential()
model.add(Conv2D(input_shape=(32, 32, 3),  kernel_initializer='he_uniform',k
ernel_size=(2,2),padding='same',strides=(2,2),filters=32))
model.add(MaxPooling2D(pool_size=(2,2),strides=(1,1),padding='same'))
model.add(Conv2D(kernel_size=(2,2),padding='same',strides=(2,2),filters=64))
model.add(MaxPooling2D(pool_size=(2, 2),strides=(1,1),padding='same'))
model.add(Flatten())
model.add(Dense(512,activation='relu'))
model.add(Dense(256, activation='relu'))
model.add(Dense(100, activation='softmax'))
model.summary()
#Compile and Training
model.compile(optimizer='Adam', loss='categorical_crossentropy', metrics=['a
ccuracy'])
history=model.fit(train_images,train_labels,batch_size=32,epochs=100,verbose
=1,validation_data=(test_images,test_labels),callbacks=[EarlyStopping(monito
r='val_loss',patience=10)])

#Draw plot and make evaluate
plt.plot(history.history['accuracy'])
plt.xlabel('epoch')
```

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

score = model.evaluate(test_images,test_labels, verbose=1)
print('Test error:',score[0])
print('Test accuracy: ',score[1])

#Make test and predict
n=int(input("Index ? "))
plt.imshow(test_images[n].reshape(32,32,3))
y_predict = model.predict(test_images[n].reshape(1,32,32,3))
print('Predicted value: ', np.argmax(y_predict))
print('Correct value: ',np.argmax(test_labels[n]))
```

2. Code trên Colab và kết quả của từng lệnh



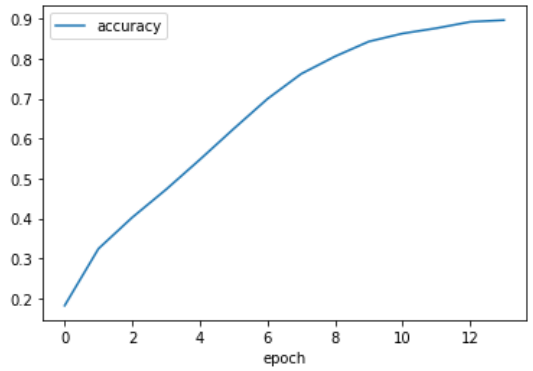
```python
#Create model for train
model = Sequential()
model.add(Conv2D(input_shape=(32, 32, 3), kernel_initializer='he_uniform',kernel_size=(2,2),padding='same',
model.add(MaxPooling2D(pool_size=(2,2),strides=(1,1),padding='same'))
model.add(Conv2D(kernel_size=(2,2),padding='same',strides=(2,2),filters=64))
model.add(MaxPooling2D(pool_size=(2, 2),strides=(1,1),padding='same'))
model.add(Flatten())
model.add(Dense(512,activation='relu'))
model.add(Dense(256, activation='relu'))
model.add(Dense(100, activation='softmax'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 16, 16, 32)        416

 max_pooling2d (MaxPooling2D  (None, 16, 16, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 8, 8, 64)          8256

 max_pooling2d_1 (MaxPooling  (None, 8, 8, 64)         0
 2D)

 flatten (Flatten)           (None, 4096)              0

 dense (Dense)               (None, 512)               2097664

 dense_1 (Dense)             (None, 256)               131328

 dense_2 (Dense)             (None, 100)               25700

=================================================================
Total params: 2,263,364
Trainable params: 2,263,364
Non-trainable params: 0
_________________________________________________________________
```

```python
#Compile and Training
model.compile(optimizer='Adam', loss='categorical_crossentropy', metrics=['accuracy'])
history=model.fit(train_images,train_labels,batch_size=32,epochs=100,verbose=1,validation_data=(test_images,
```

```
Epoch 1/100
1563/1563 [==============================] - 19s 5ms/step - loss: 3.4659 - accuracy: 0.1819 - val_loss: 2.981
Epoch 2/100
1563/1563 [==============================] - 7s 5ms/step - loss: 2.6691 - accuracy: 0.3242 - val_loss: 2.6056
Epoch 3/100
1563/1563 [==============================] - 7s 4ms/step - loss: 2.2848 - accuracy: 0.4025 - val_loss: 2.5361
Epoch 4/100
1563/1563 [==============================] - 7s 4ms/step - loss: 1.9549 - accuracy: 0.4724 - val_loss: 2.4883
Epoch 5/100
1563/1563 [==============================] - 7s 5ms/step - loss: 1.6301 - accuracy: 0.5473 - val_loss: 2.6236
Epoch 6/100
1563/1563 [==============================] - 7s 5ms/step - loss: 1.3161 - accuracy: 0.6243 - val_loss: 2.8521
Epoch 7/100
1563/1563 [==============================] - 7s 4ms/step - loss: 1.0212 - accuracy: 0.6992 - val_loss: 3.1297
Epoch 8/100
1563/1563 [==============================] - 7s 4ms/step - loss: 0.7883 - accuracy: 0.7619 - val_loss: 3.4431
Epoch 9/100
1563/1563 [==============================] - 7s 5ms/step - loss: 0.6256 - accuracy: 0.8055 - val_loss: 3.9524
Epoch 10/100
1563/1563 [==============================] - 7s 4ms/step - loss: 0.4997 - accuracy: 0.8428 - val_loss: 4.2755
Epoch 11/100
1563/1563 [==============================] - 7s 4ms/step - loss: 0.4384 - accuracy: 0.8629 - val_loss: 4.7422
Epoch 12/100
1563/1563 [==============================] - 7s 4ms/step - loss: 0.3974 - accuracy: 0.8761 - val_loss: 5.0034
Epoch 13/100
1563/1563 [==============================] - 7s 5ms/step - loss: 0.3429 - accuracy: 0.8920 - val_loss: 5.4976
Epoch 14/100
1563/1563 [==============================] - 7s 5ms/step - loss: 0.3291 - accuracy: 0.8963 - val_loss: 5.6835
```

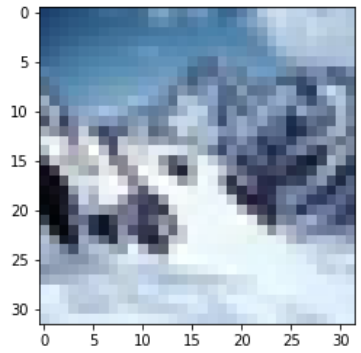
```python
#Draw plot and make evaluate
plt.plot(history.history['accuracy'])
plt.xlabel('epoch')
plt.legend(['accuracy'])
plt.show()

score = model.evaluate(test_images,test_labels, verbose=1)
print('Test error:',score[0])
print('Test accuracy: ',score[1])
```




```
313/313 [==============================] - 1s 5ms/step - loss: 5.6835 - accuracy: 0.3614
Test error: 5.683497905731201
Test accuracy:  0.3614000082015991
```

```
#Make test and predict
n=int(input("Index ? "))
plt.imshow(test_images[n].reshape(32,32,3))
y_predict = model.predict(test_images[n].reshape(1,32,32,3))
print('Predicted value: ', np.argmax(y_predict))
print('Correct value: ',np.argmax(test_labels[n]))
```

```
Index ? 21
Predicted value:  48
Correct value:  49
```



II. SỬ DỤNG MNIST NHẬN DIỆN TỪ DATASET CÓ SẴN

1. Code Tổng Quan

```
#Import libraries
from keras.datasets import mnist
from sklearn import utils
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout,Conv2D,MaxPooling2D,Flat
ten
from tensorflow.keras.optimizers import RMSprop,Adam
import numpy as np
from keras.utils import np_utils
import pandas as pd
from keras.callbacks import EarlyStopping
import matplotlib.pyplot as plt
#Import dataset
(x_train,y_train),(x_test,y_test)=mnist.load_data()

#Set data type
x_train=x_train.astype('float32')
x_test=x_test.astype('float32')

x_train=x_train.reshape(60000,28,28,1)
x_test=x_test.reshape(10000,28,28,1)

x_train/=255
x_test/=255
```

```
y_train=np_utils.to_categorical(y_train,10)
y_test=np_utils.to_categorical(y_test,10)

#Create model for train
model = Sequential()
model.add(Conv2D(64,(3,3),activation='relu',input_shape=(28,28,1)))
model.add(MaxPooling2D(2,2))
model.add(Conv2D(64,(3,3),input_shape=(28,28),activation='relu'))
model.add(MaxPooling2D(2,2))
model.add(Flatten())
model.add(Dense(128,activation='relu'))
model.add(Dense(10, activation='softmax'))
model.summary()
#Compile and Training
model.compile(optimizer='Adam', loss='categorical_crossentropy', metrics=['a
ccuracy'])
history=model.fit(x_train,y_train,batch_size=128,epochs=500,verbose=1,valida
tion_data=(x_test,y_test),callbacks=[EarlyStopping(monitor='val_loss',patien
ce=20)])
#Draw plot and make evaluate
plt.plot(history.history['accuracy'])
plt.xlabel('epoch')
plt.legend(['accuracy'])
plt.show()

score = model.evaluate(x_test, y_test, verbose=0)
print('Test error:',score[0])
print('Test accuracy: ',score[1])
```

2. Code trên Colab và kết quả của từng lệnh

```
[1] #Import libraries
    from keras.datasets import mnist
    from sklearn import utils
    from keras.models import Sequential
    from keras.layers import Dense, Activation, Dropout,Conv2D,MaxPooling2D,Flatten
    from tensorflow.keras.optimizers import RMSprop,Adam
    import numpy as np
    from keras.utils import np_utils
    import pandas as pd
    from keras.callbacks import EarlyStopping
    import matplotlib.pyplot as plt
```

```
[2] #Import dataset
    (x_train,y_train),(x_test,y_test)=mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
11501568/11490434 [==============================] - 0s 0us/step
```

```
[3] #Set data type
    x_train=x_train.astype('float32')
    x_test=x_test.astype('float32')

    x_train=x_train.reshape(60000,28,28,1)
    x_test=x_test.reshape(10000,28,28,1)

    x_train/=255
    x_test/=255

    y_train=np_utils.to_categorical(y_train,10)
    y_test=np_utils.to_categorical(y_test,10)
```

```
#Create model for train
model = Sequential()
model.add(Conv2D(64,(3,3),activation='relu',input_shape=(28,28,1)))
model.add(MaxPooling2D(2,2))
model.add(Conv2D(64,(3,3),input_shape=(28,28),activation='relu'))
model.add(MaxPooling2D(2,2))
model.add(Flatten())
model.add(Dense(128,activation='relu'))
model.add(Dense(10, activation='softmax'))
model.summary()
```

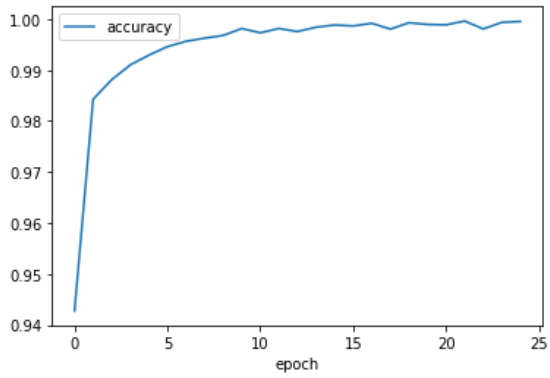
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 64)        640

 max_pooling2d (MaxPooling2D  (None, 13, 13, 64)       0
 )

 conv2d_1 (Conv2D)           (None, 11, 11, 64)        36928

 max_pooling2d_1 (MaxPooling  (None, 5, 5, 64)         0
 2D)

 flatten (Flatten)           (None, 1600)              0

 dense (Dense)               (None, 128)               204928

 dense_1 (Dense)             (None, 10)                1290

=================================================================
Total params: 243,786
Trainable params: 243,786
Non-trainable params: 0
_________________________________________________________________
```

```
#Compile and Training
model.compile(optimizer='Adam', loss='categorical_crossentropy', metrics=['accuracy'])
history=model.fit(x_train,y_train,batch_size=128,epochs=500,verbose=1,validation_data=(x_test,y_test),callba
```

```
Epoch 1/500
469/469 [==============================] - 85s 180ms/step - loss: 0.2003 - accuracy: 0.9428 - val_loss: 0.07
Epoch 2/500
469/469 [==============================] - 81s 172ms/step - loss: 0.0530 - accuracy: 0.9843 - val_loss: 0.05
Epoch 3/500
469/469 [==============================] - 88s 189ms/step - loss: 0.0373 - accuracy: 0.9881 - val_loss: 0.03
```

```
469/469 [==============================] - 81s 174ms/step - loss: 0.0039 - accur
Epoch 17/500
469/469 [==============================] - 81s 174ms/step - loss: 0.0028 - accuracy: 0.9992 - val_loss: 0.043
Epoch 18/500
469/469 [==============================] - 82s 176ms/step - loss: 0.0057 - accuracy: 0.9980 - val_loss: 0.036
Epoch 19/500
469/469 [==============================] - 80s 171ms/step - loss: 0.0026 - accuracy: 0.9993 - val_loss: 0.034
Epoch 20/500
469/469 [==============================] - 81s 172ms/step - loss: 0.0029 - accuracy: 0.9990 - val_loss: 0.044
Epoch 21/500
469/469 [==============================] - 81s 174ms/step - loss: 0.0032 - accuracy: 0.9989 - val_loss: 0.045
Epoch 22/500
469/469 [==============================] - 80s 171ms/step - loss: 0.0011 - accuracy: 0.9996 - val_loss: 0.038
Epoch 23/500
469/469 [==============================] - 81s 172ms/step - loss: 0.0057 - accuracy: 0.9981 - val_loss: 0.034
Epoch 24/500
469/469 [==============================] - 81s 172ms/step - loss: 0.0018 - accuracy: 0.9994 - val_loss: 0.036
Epoch 25/500
469/469 [==============================] - 81s 172ms/step - loss: 0.0018 - accuracy: 0.9995 - val_loss: 0.034
```

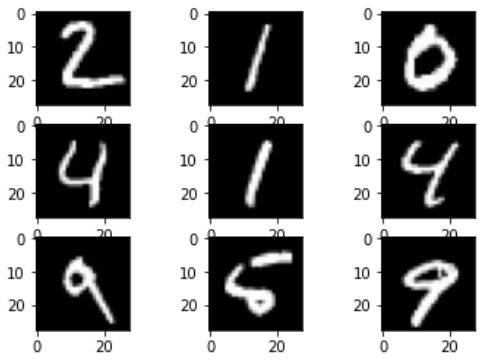
[6]
```
#Draw plot and make evaluate
plt.plot(history.history['accuracy'])
plt.xlabel('epoch')
plt.legend(['accuracy'])
plt.show()

score = model.evaluate(x_test, y_test, verbose=0)
print('Test error:',score[0])
print('Test accuracy: ',score[1])
```




```
Test error: 0.03459727764129639
Test accuracy:  0.9936000108718872
```

```
[7] #Make test and predict
    n=int(input("Index ? "))
    print(x_test.shape)

    y_predict = model.predict(x_test[n].reshape(1,28,28,1))
    print('Predicted value: ', np.argmax(y_predict))
    print('Correct value: ',np.argmax(y_test[n]))

    (x_tr,y_tr),(x_t,y_t)=mnist.load_data()
    for i in range(1,10):
      plt.subplot(330+i)
      plt.imshow(x_t[i],cmap='gray')
```

```
Index ? 35
(10000, 28, 28, 1)
Predicted value:  2
Correct value:  2
```



III. SỬ DỤNG MNIST NHẬN DIỆN FASHION TỪ DATASET CÓ SẴN

1. Code Tổng Quan

```
#Import libraries
from keras.datasets import fashion_mnist
from sklearn import utils
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout,Conv2D,MaxPooling2D,Flat
ten
from tensorflow.keras.optimizers import RMSprop,Adam
import numpy as np
from keras.utils import np_utils
import pandas as pd
from keras.callbacks import EarlyStopping
import matplotlib.pyplot as plt

#Import dataset
(x_train,y_train),(x_test,y_test)=fashion_mnist.load_data()

#Set data type
x_train=x_train.astype('float32')
```

```
x_test=x_test.astype('float32')

x_train=x_train.reshape(60000,28,28,1)
x_test=x_test.reshape(10000,28,28,1)

x_train/=255
x_test/=255

y_train=np_utils.to_categorical(y_train,10)
y_test=np_utils.to_categorical(y_test,10)

#Create model for train
model = Sequential()
model.add(Conv2D(64,(3,3),activation='relu',input_shape=(28,28,1)))
model.add(MaxPooling2D(2,2))
model.add(Conv2D(64,(3,3),input_shape=(28,28),activation='relu'))
model.add(MaxPooling2D(2,2))
model.add(Flatten())
model.add(Dense(128,activation='relu'))
model.add(Dense(10, activation='softmax'))
model.summary()
#Compile and Training
model.compile(optimizer='Adam', loss='categorical_crossentropy', metrics=['a
ccuracy'])
history=model.fit(x_train,y_train,batch_size=128,epochs=500,verbose=1,valida
tion_data=(x_test,y_test),callbacks=[EarlyStopping(monitor='val_loss',patien
ce=20)])
#Draw plot and make evaluate

plt.plot(history.history['accuracy'])
plt.xlabel('epoch')
plt.legend(['accuracy'])
plt.show()

score = model.evaluate(x_test, y_test, verbose=0)
print('Test error:',score[0])
print('Test accuracy: ',score[1])
#Make test and predict
n=int(input("Index ? "))
print(x_test.shape)

y_predict = model.predict(x_test[n].reshape(1,28,28,1))
print('Predicted value: ', np.argmax(y_predict))
print('Correct value: ',np.argmax(y_test[n]))

(x_tr,y_tr),(x_t,y_t)=fashion_mnist.load_data()
for i in range(1,10):
```

```
plt.subplot(330+i)
plt.imshow(x_t[i],cmap='gray')
```

2. Code trên Colab và kết quả của từng lệnh

+ Code + Text RAM Disk Editing

```
#Import libraries
from keras.datasets import fashion_mnist
from sklearn import utils
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout,Conv2D,MaxPooling2D,Flatten
from tensorflow.keras.optimizers import RMSprop,Adam
import numpy as np
from keras.utils import np_utils
import pandas as pd
from keras.callbacks import EarlyStopping
import matplotlib.pyplot as plt
```

```
[2] #Import dataset
(x_train,y_train),(x_test,y_test)=fashion_mnist.load_data()
```

```
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 [==============================================================================================]
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
```

```
[3] #Set data type
x_train=x_train.astype('float32')
x_test=x_test.astype('float32')

x_train=x_train.reshape(60000,28,28,1)
x_test=x_test.reshape(10000,28,28,1)

x_train/=255
x_test/=255

y_train=np_utils.to_categorical(y_train,10)
y_test=np_utils.to_categorical(y_test,10)
```

```
#Create model for train
model = Sequential()
model.add(Conv2D(64,(3,3),activation='relu',input_shape=(28,28,1)))
model.add(MaxPooling2D(2,2))
model.add(Conv2D(64,(3,3),input_shape=(28,28),activation='relu'))
model.add(MaxPooling2D(2,2))
model.add(Flatten())
model.add(Dense(128,activation='relu'))
model.add(Dense(10, activation='softmax'))
model.summary()
```

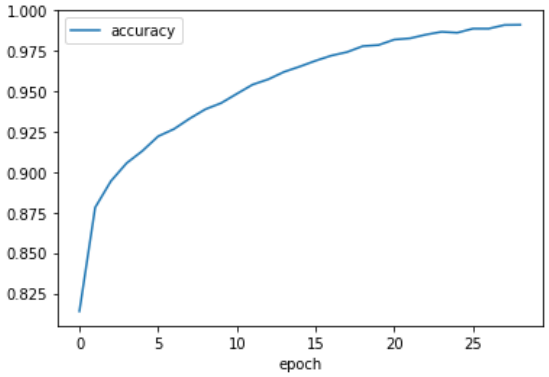
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 64)        640

 max_pooling2d (MaxPooling2D  (None, 13, 13, 64)       0
 )

 conv2d_1 (Conv2D)           (None, 11, 11, 64)        36928

 max_pooling2d_1 (MaxPooling  (None, 5, 5, 64)         0
 2D)

 flatten (Flatten)           (None, 1600)              0

 dense (Dense)               (None, 128)               204928

 dense_1 (Dense)             (None, 10)                1290

=================================================================
Total params: 243,786
Trainable params: 243,786
Non-trainable params: 0
_________________________________________________________________
```

```python
#Compile and Training
model.compile(optimizer='Adam', loss='categorical_crossentropy', metrics=['accuracy'])
history=model.fit(x_train,y_train,batch_size=128,epochs=500,verbose=1,validation_data=(x_test,y_test),call
```

```
Epoch 1/500
469/469 [==============================] - 90s 189ms/step - loss: 0.5222 - accuracy: 0.8140 - val_loss: 0.
Epoch 2/500
469/469 [==============================] - 82s 176ms/step - loss: 0.3391 - accuracy: 0.8782 - val_loss: 0.
Epoch 3/500
469/469 [==============================] - 82s 175ms/step - loss: 0.2921 - accuracy: 0.8945 - val_loss: 0.
Epoch 4/500
469/469 [==============================] - 83s 178ms/step - loss: 0.2597 - accuracy: 0.9057 - val_loss: 0.
Epoch 5/500
469/469 [==============================] - 83s 176ms/step - loss: 0.2385 - accuracy: 0.9132 - val_loss: 0.
Epoch 6/500
469/469 [==============================] - 83s 177ms/step - loss: 0.2153 - accuracy: 0.9222 - val_loss: 0.
Epoch 7/500
469/469 [==============================] - 82s 174ms/step - loss: 0.1982 - accuracy: 0.9267 - val_loss: 0.
Epoch 8/500
469/469 [==============================] - 81s 172ms/step - loss: 0.1811 - accuracy: 0.9332 - val_loss: 0.
Epoch 9/500
469/469 [==============================] - 81s 173ms/step - loss: 0.1661 - accuracy: 0.9390 - val_loss: 0.
Epoch 10/500
469/469 [==============================] - 81s 172ms/step - loss: 0.1529 - accuracy: 0.9428 - val_loss: 0.
Epoch 11/500
469/469 [==============================] - 80s 171ms/step - loss: 0.1390 - accuracy: 0.9485 - val_loss: 0.
Epoch 12/500
469/469 [==============================] - 81s 173ms/step - loss: 0.1252 - accuracy: 0.9542 - val_loss: 0.
Epoch 13/500
469/469 [==============================] - 81s 173ms/step - loss: 0.1149 - accuracy: 0.9575 - val_loss: 0.
Epoch 14/500
469/469 [==============================] - 80s 171ms/step - loss: 0.1039 - accuracy: 0.9621 - val_loss: 0.
Epoch 15/500
469/469 [==============================] - 79s 169ms/step - loss: 0.0928 - accuracy: 0.9654 - val_loss: 0.
```

```
[5]
Epoch 27/500
469/469 [==============================] - 80s 171ms/step - loss: 0.0298 - accuracy: 0.9888 - val_loss: 0.
Epoch 28/500
469/469 [==============================] - 80s 170ms/step - loss: 0.0256 - accuracy: 0.9911 - val_loss: 0.
Epoch 29/500
469/469 [==============================] - 80s 170ms/step - loss: 0.0245 - accuracy: 0.9912 - val_loss: 0.
```

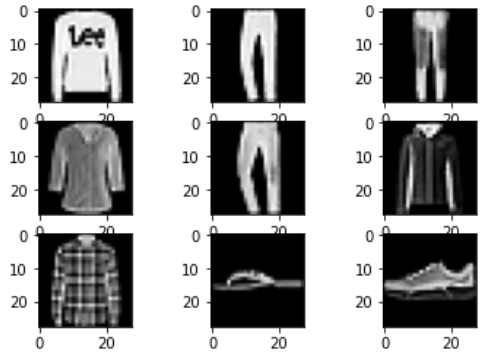
```
#Draw plot and make evaluate

plt.plot(history.history['accuracy'])
plt.xlabel('epoch')
plt.legend(['accuracy'])
plt.show()

score = model.evaluate(x_test, y_test, verbose=0)
print('Test error:',score[0])
print('Test accuracy: ',score[1])
```




```
Test error: 0.4804755747318268
Test accuracy:  0.9136999845504761
```

```
#Make test and predict
n=int(input("Index ? "))
print(x_test.shape)

y_predict = model.predict(x_test[n].reshape(1,28,28,1))
print('Predicted value: ', np.argmax(y_predict))
print('Correct value: ',np.argmax(y_test[n]))

(x_tr,y_tr),(x_t,y_t)=fashion_mnist.load_data()
for i in range(1,10):
  plt.subplot(330+i)
  plt.imshow(x_t[i],cmap='gray')
```

```
Index ? 52
(10000, 28, 28, 1)
Predicted value:  5
Correct value:  5
```



IV. NHẬN DIỆN TRÁI CÂY DÙNG CNN

1. Code Tổng Quan

```
#Import libraries
import glob
import cv2
import pandas as pd
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
from random import randint
from keras.utils import np_utils
from keras.models import Sequential, load_model
from keras.layers import Dense, Flatten, Dropout, Conv2D, MaxPooling2D
from tensorflow.keras.optimizers import RMSprop, SGD, Adam
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras.preprocessing.image import ImageDataGenerator
#Connect with my drive
from google.colab import drive
drive.mount('/content/drive')
#Get dataset from drive
```

```
train=ImageDataGenerator(rescale=1/255)
validation=ImageDataGenerator(rescale=1/255)
training_set=train.flow_from_directory('/content/drive/MyDrive/AI/FRUIT/trai
n/train',target_size=(150,150), batch_size=40, class_mode='categorical')
validation_set=validation.flow_from_directory('/content/drive/MyDrive/AI/FRU
IT/train/val',target_size=(150,150), batch_size=40, class_mode='categorical'
)
#Check label of dataset
training_set.class_indices
#Create Model for train
model = Sequential()
model.add(Conv2D(16,(3,3),padding='same',kernel_initializer='he_normal',inpu
t_shape=(150,150,3)))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(32,(3,3),padding='same',kernel_initializer='he_normal'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(64,(3,3),padding='same',kernel_initializer='he_normal'))
model.add(MaxPooling2D(pool_size=(2,2)))


model.add(Flatten())
model.add(Dense(512,activation='relu',kernel_initializer='he_normal'))
model.add(Dense(10,activation='softmax'))
model.summary()
#Training
opt = SGD(lr=0.01, momentum=0.9)
model.compile(optimizer=RMSprop(),loss='categorical_crossentropy',  metrics=
['accuracy'])
from keras.callbacks import EarlyStopping
history = model.fit(training_set, epochs = 10, validation_data = validation_
set, verbose=1, callbacks=[EarlyStopping(monitor='val_loss', patience=15)])
#Save a file after train
model.save('CNN_FRUIT.h5')
#Draw plot and evaluate
score = model.evaluate(validation_set,verbose=0)
print('Test error: ',score[0])
print('Test accuracy: ',score[1])
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.ylabel('accuracy')
plt.xlabel('epochs')
plt.legend(['train','Validation'])
plt.show()
#Load trained file
CNN_FRUIT =load_model('CNN_FRUIT.h5')
```

```
#Check detect
img_path = '/content/drive/MyDrive/AI/FRUIT/test/52.jpg'
img=load_img(img_path,target_size=(150,150))
plt.imshow(img)
img=img_to_array(img)
img=img.reshape(1,150,150,3)
img=img.astype('float32')
img=img/255
Fruit=np.argmax(CNN_FRUIT.predict(img),axis=1)
pred = model.predict(img)
classes = ['passionfruit','peaches','pears','pineapples','plums','pomegranat
es','raspberries','strawberries','tomatoes','watermelons']
print(np.argmax(pred))
if Fruit==0: print(classes[0])
elif Fruit==1: print(classes[1])
elif Fruit==2: print(classes[2])
elif Fruit==3: print(classes[3])
elif Fruit==4: print(classes[4])
elif Fruit==5: print(classes[5])
elif Fruit==6: print(classes[6])
elif Fruit==7: print(classes[7])
elif Fruit==8: print(classes[8])
elif Fruit==9: print(classes[9])
```

2. Code trên Colab và kết quả của từng lệnh

+ Code + Text Reconnect Editing

```
#Import libraries
import glob
import cv2
import pandas as pd
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
from random import randint
from keras.utils import np_utils
from keras.models import Sequential, load_model
from keras.layers import Dense, Flatten, Dropout, Conv2D, MaxPooling2D
from tensorflow.keras.optimizers import RMSprop, SGD, Adam
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
[ ] #Connect with my drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
[ ]
drive/MyDrive/AI/FRUIT/train/train',target_size=(150,150), batch_size=40, class_mode='categorical')
ontent/drive/MyDrive/AI/FRUIT/train/val',target_size=(150,150), batch_size=40, class_mode='categorical')
```

```
Found 338 images belonging to 10 classes.
Found 338 images belonging to 10 classes.
```

```
[ ] #Check label of dataset
    training_set.class_indices
```

```
{'passionfruit': 0,
 'peaches': 1,
 'pears': 2,
 'pineapples': 3,
 'plums': 4,
 'pomegranates': 5,
 'raspberries': 6,
 'strawberries': 7,
 'tomatoes': 8,
 'watermelons': 9}
```

```
[ ] #Create Model for train
    model = Sequential()
    model.add(Conv2D(16,(3,3),padding='same',kernel_initializer='he_normal',input_shape=(150,150,3)))
    model.add(MaxPooling2D(pool_size=(2,2)))

    model.add(Conv2D(32,(3,3),padding='same',kernel_initializer='he_normal'))
    model.add(MaxPooling2D(pool_size=(2,2)))

    model.add(Conv2D(64,(3,3),padding='same',kernel_initializer='he_normal'))
    model.add(MaxPooling2D(pool_size=(2,2)))


    model.add(Flatten())
    model.add(Dense(512,activation='relu',kernel_initializer='he_normal'))
    model.add(Dense(10,activation='softmax'))
    model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 150, 150, 16)      448

 max_pooling2d (MaxPooling2D  (None, 75, 75, 16)       0
 )

 conv2d_1 (Conv2D)           (None, 75, 75, 32)        4640

 max_pooling2d_1 (MaxPooling  (None, 37, 37, 32)       0
 2D)
```

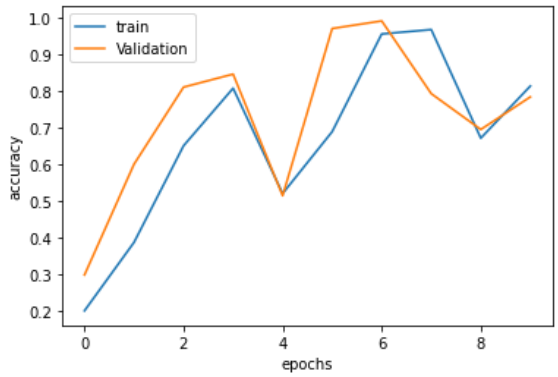
```
#Training
opt = SGD(lr=0.01, momentum=0.9)
model.compile(optimizer=RMSprop(),loss='categorical_crossentropy',  metrics=['accuracy'])
from keras.callbacks import EarlyStopping
history = model.fit(training_set, epochs = 10, validation_data = validation_set, verbose=1, callbacks=[Earl
```

```
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descent.py:102: UserWarning: The `lr` argu
  super(SGD, self).__init__(name, **kwargs)
Epoch 1/10
9/9 [==============================] - 262s 29s/step - loss: 57.5984 - accuracy: 0.2012 - val_loss: 21.3964 -
Epoch 2/10
9/9 [==============================] - 21s 2s/step - loss: 12.2867 - accuracy: 0.3876 - val_loss: 2.3318 - va
Epoch 3/10
9/9 [==============================] - 21s 2s/step - loss: 1.8595 - accuracy: 0.6509 - val_loss: 0.6878 - val
Epoch 4/10
9/9 [==============================] - 21s 2s/step - loss: 0.8379 - accuracy: 0.8077 - val_loss: 0.4958 - val
Epoch 5/10
9/9 [==============================] - 21s 2s/step - loss: 6.3797 - accuracy: 0.5207 - val_loss: 3.3969 - val
Epoch 6/10
9/9 [==============================] - 21s 2s/step - loss: 2.2402 - accuracy: 0.6893 - val_loss: 0.1163 - val
Epoch 7/10
9/9 [==============================] - 21s 2s/step - loss: 0.1391 - accuracy: 0.9556 - val_loss: 0.0451 - val
Epoch 8/10
9/9 [==============================] - 21s 2s/step - loss: 0.0961 - accuracy: 0.9675 - val_loss: 1.0558 - val
Epoch 9/10
9/9 [==============================] - 21s 2s/step - loss: 2.1547 - accuracy: 0.6716 - val_loss: 2.4518 - val
Epoch 10/10
9/9 [==============================] - 20s 2s/step - loss: 1.0251 - accuracy: 0.8136 - val_loss: 0.8940 - val
```

```
[ ] #Save a file after train
    model.save('CNN_FRUIT.h5')
```

```
#Draw plot and evaluate
score = model.evaluate(validation_set,verbose=0)
print('Test error: ',score[0])
print('Test accuracy: ',score[1])
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.ylabel('accuracy')
plt.xlabel('epochs')
plt.legend(['train','Validation'])
plt.show()
```

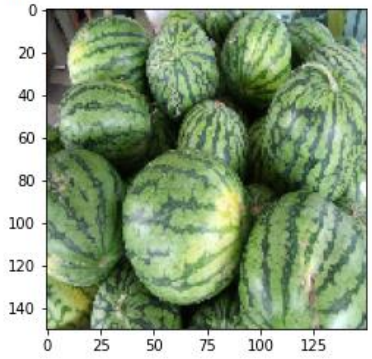
```
Test error:  0.8940480351448059
Test accuracy:  0.784023642539978
```




```
[ ] #Load trained file
    CNN_FRUIT =load_model('CNN_FRUIT.h5')
```

```
#Check detect
img_path = '/content/drive/MyDrive/AI/FRUIT/test/52.jpg'
img=load_img(img_path,target_size=(150,150))
plt.imshow(img)
img=img_to_array(img)
img=img.reshape(1,150,150,3)
img=img.astype('float32')
img=img/255
Fruit=np.argmax(CNN_FRUIT.predict(img),axis=1)
pred = model.predict(img)
classes = ['passionfruit','peaches','pears','pineapples','plums','pomegranates','raspberries','strawberries'
print(np.argmax(pred))
if Fruit==0: print(classes[0])
elif Fruit==1: print(classes[1])
elif Fruit==2: print(classes[2])
elif Fruit==3: print(classes[3])
elif Fruit==4: print(classes[4])
elif Fruit==5: print(classes[5])
elif Fruit==6: print(classes[6])
elif Fruit==7: print(classes[7])
elif Fruit==8: print(classes[8])
elif Fruit==9: print(classes[9])
```

```
9
watermelons
```



V. NHẬN DIỆN TIỀN DÙNG CNN

1. Code Tổng Quan

```
#Import libraries
import glob
import cv2
import pandas as pd
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
from random import randint
from keras.utils import np_utils
from keras.models import Sequential, load_model
from keras.layers import Dense, Flatten, Dropout, Conv2D, MaxPooling2D
from tensorflow.keras.optimizers import RMSprop, SGD, Adam
from tensorflow.keras.preprocessing.image import load_img, img_to_array
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
#Connect with my drive
from google.colab import drive
drive.mount('/content/drive')
#Get dataset from drive
train=ImageDataGenerator(rescale=1/255)
validation=ImageDataGenerator(rescale=1/255)
training_set=train.flow_from_directory('/content/drive/MyDrive/AI/Tien/train
/train',target_size=(150,150), batch_size=40, class_mode='categorical')
validation_set=validation.flow_from_directory('/content/drive/MyDrive/AI/Tie
n/train/validation',target_size=(150,150), batch_size=40, class_mode='catego
rical')
#Check label of dataset
training_set.class_indices
#Create Model for train
model = Sequential()
model.add(Conv2D(16,(3,3),padding='same',kernel_initializer='he_normal',inpu
t_shape=(150,150,3)))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(32,(3,3),padding='same',kernel_initializer='he_normal'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(64,(3,3),padding='same',kernel_initializer='he_normal'))
model.add(MaxPooling2D(pool_size=(2,2)))


model.add(Flatten())
model.add(Dense(512,activation='relu',kernel_initializer='he_normal'))
model.add(Dense(6,activation='softmax'))
model.summary()
#Training
opt = SGD(lr=0.01, momentum=0.9)
model.compile(optimizer=RMSprop(),loss='categorical_crossentropy',  metrics=
['accuracy'])
from keras.callbacks import EarlyStopping
history = model.fit(training_set, epochs = 10, validation_data = validation_
set, verbose=1, callbacks=[EarlyStopping(monitor='val_loss', patience=15)])
#Save a file after train
model.save('CNN_MONEY.h5')
#Draw plot and evaluate
score = model.evaluate(validation_set,verbose=0)
print('Test error: ',score[0])
print('Test accuracy: ',score[1])
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.ylabel('accuracy')
```

```
plt.xlabel('epochs')
plt.legend(['train','Validation'])
plt.show()
#Load trained file
CNN_money =load_model('CNN_MONEY.h5')
#Check detect
img_path = '/content/drive/MyDrive/AI/Tien/test/20k/20k (9).jpg'
img=load_img(img_path,target_size=(150,150))
plt.imshow(img)
img=img_to_array(img)
img=img.reshape(1,150,150,3)
img=img.astype('float32')
img=img/255
money=np.argmax(CNN_money.predict(img),axis=1)
pred = model.predict(img)
classes = ['01k','02k','05k','10k','20k','50k']

print(np.argmax(pred))
if money==0: print(classes[0])
elif money==1: print(classes[1])
elif money==2: print(classes[2])
elif money==3: print(classes[3])
elif money==4: print(classes[4])
```

2. Code trên Colab và kết quả của từng lệnh

+ Code + Text

RAM Disk

Editing

```
#Import libraries
import glob
import cv2
import pandas as pd
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
from random import randint
from keras.utils import np_utils
from keras.models import Sequential, load_model
from keras.layers import Dense, Flatten, Dropout, Conv2D, MaxPooling2D
from tensorflow.keras.optimizers import RMSprop, SGD, Adam
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

[2]

```
#Connect with my drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[3]

```
yDrive/AI/Tien/train/train',target_size=(150,150), batch_size=40, class_mode='categorical')
/drive/MyDrive/AI/Tien/train/validation',target_size=(150,150), batch_size=40, class_mode='categorical')
```

```
Found 120 images belonging to 6 classes.
Found 120 images belonging to 6 classes.
```

[4]

```
#Check label of dataset
training_set.class_indices
```

```
{'01k': 0, '02k': 1, '05k': 2, '10k': 3, '20k': 4, '50k': 5}
```

Double-click (or enter) to edit

+ Code + Text

RAM Disk

Editing

```
#Create Model for train
model = Sequential()
model.add(Conv2D(16,(3,3),padding='same',kernel_initializer='he_normal',input_shape=(150,150,3)))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(32,(3,3),padding='same',kernel_initializer='he_normal'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(64,(3,3),padding='same',kernel_initializer='he_normal'))
model.add(MaxPooling2D(pool_size=(2,2)))


model.add(Flatten())
model.add(Dense(512,activation='relu',kernel_initializer='he_normal'))
model.add(Dense(6,activation='softmax'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 150, 150, 16)      448

 max_pooling2d (MaxPooling2D  (None, 75, 75, 16)       0
 )

 conv2d_1 (Conv2D)           (None, 75, 75, 32)        4640

 max_pooling2d_1 (MaxPooling  (None, 37, 37, 32)       0
 2D)

 conv2d_2 (Conv2D)           (None, 37, 37, 64)        18496

 max_pooling2d_2 (MaxPooling  (None, 18, 18, 64)       0
 2D)

 flatten (Flatten)           (None, 20736)             0

 dense (Dense)               (None, 512)               10617344

 dense_1 (Dense)             (None, 6)                 3078

=================================================================
Total params: 10,644,006
Trainable params: 10,644,006
Non-trainable params: 0
_________________________________________________________________
```

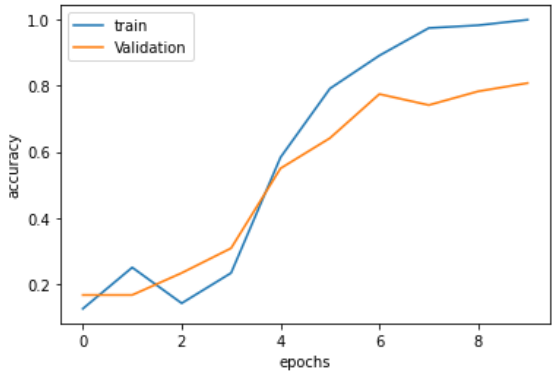
```
#Training
opt = SGD(lr=0.01, momentum=0.9)
model.compile(optimizer=RMSprop(),loss='categorical_crossentropy',  metrics=['accuracy'])
from keras.callbacks import EarlyStopping
history = model.fit(training_set, epochs = 10, validation_data = validation_set, verbose=1, callbacks=[Earl
```

```
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descent.py:102: UserWarning: The `lr` argu
  super(SGD, self).__init__(name, **kwargs)
Epoch 1/10
3/3 [==============================] - 63s 24s/step - loss: 87.0198 - accuracy: 0.1250 - val_loss: 225.6956 -
Epoch 2/10
3/3 [==============================] - 5s 2s/step - loss: 139.7541 - accuracy: 0.2500 - val_loss: 80.0438 - v
Epoch 3/10
3/3 [==============================] - 5s 2s/step - loss: 55.6856 - accuracy: 0.1417 - val_loss: 19.5561 - va
Epoch 4/10
3/3 [==============================] - 5s 2s/step - loss: 17.7833 - accuracy: 0.2333 - val_loss: 4.8543 - val
Epoch 5/10
3/3 [==============================] - 5s 2s/step - loss: 2.5973 - accuracy: 0.5833 - val_loss: 2.1795 - val_
Epoch 6/10
3/3 [==============================] - 5s 2s/step - loss: 0.7957 - accuracy: 0.7917 - val_loss: 1.2054 - val_
Epoch 7/10
3/3 [==============================] - 5s 2s/step - loss: 0.3386 - accuracy: 0.8917 - val_loss: 0.9113 - val_
Epoch 8/10
3/3 [==============================] - 5s 2s/step - loss: 0.1427 - accuracy: 0.9750 - val_loss: 0.9072 - val_
Epoch 9/10
3/3 [==============================] - 5s 2s/step - loss: 0.1201 - accuracy: 0.9833 - val_loss: 0.7979 - val_
Epoch 10/10
3/3 [==============================] - 5s 2s/step - loss: 0.0503 - accuracy: 1.0000 - val_loss: 0.7753 - val_
```

```
[7] #Save a file after train
    model.save('CNN_MONEY.h5')
```

```
#Draw plot and evaluate
score = model.evaluate(validation_set,verbose=0)
print('Test error: ',score[0])
print('Test accuracy: ',score[1])
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.ylabel('accuracy')
plt.xlabel('epochs')
plt.legend(['train','Validation'])
plt.show()
```

```
Test error:  0.77530837059021
Test accuracy:  0.8083333373069763
```




```
[9] #Load trained file
CNN_money =load_model('CNN_MONEY.h5')
```

```
#Check detect
img_path = '/content/drive/MyDrive/AI/Tien/test/20k/20k (8).jpg'
img=load_img(img_path,target_size=(150,150))
plt.imshow(img)
img=img_to_array(img)
img=img.reshape(1,150,150,3)
img=img.astype('float32')
img=img/255
money=np.argmax(CNN_money.predict(img),axis=1)
pred = model.predict(img)
classes = ['01k','02k','05k','10k','20k','50k']

print(np.argmax(pred))
if money==0: print(classes[0])
elif money==1: print(classes[1])
elif money==2: print(classes[2])
elif money==3: print(classes[3])
elif money==4: print(classes[4])
```

```
4
20k
```

VI. NHẬN DIỆN THỨC ĂN DÙNG CNN

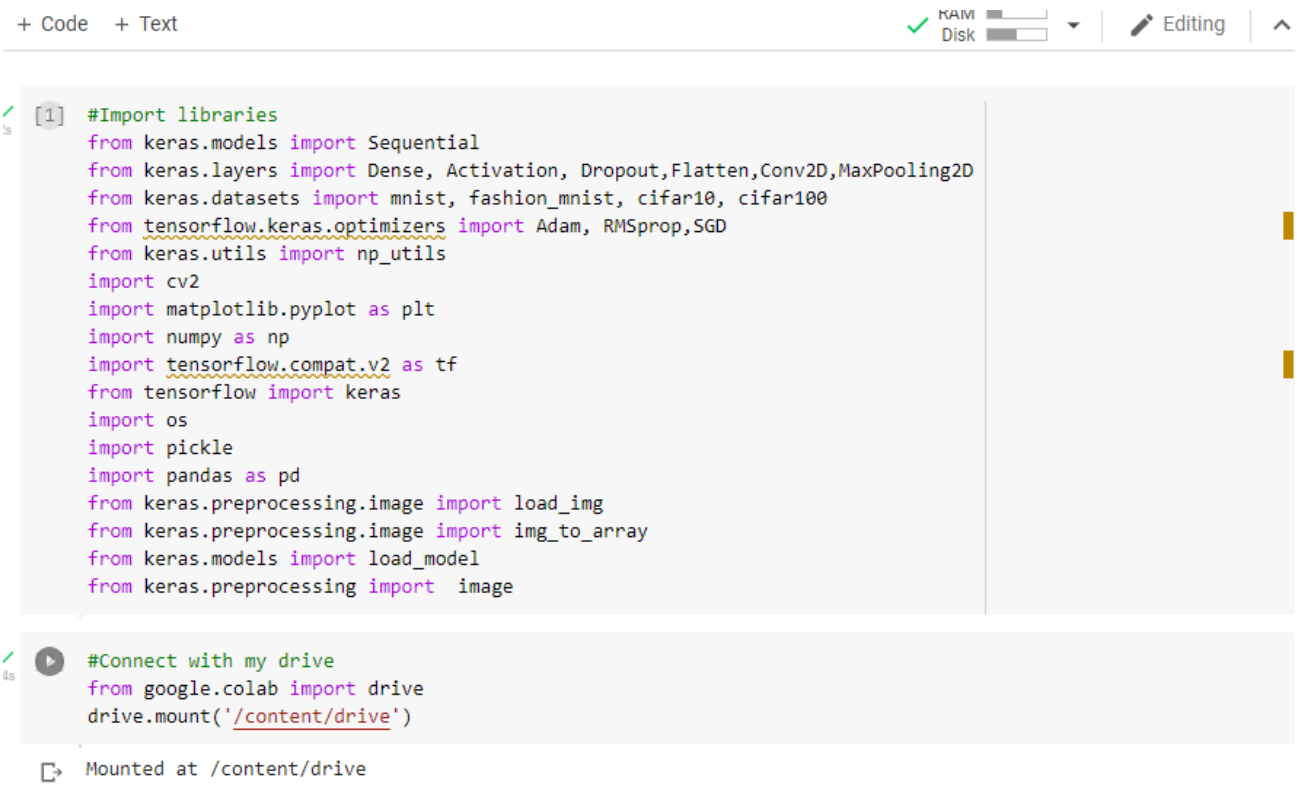
1. Code Tổng Quan

```
#Import libraries
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout,Flatten,Conv2D,MaxPoolin
g2D
from keras.datasets import mnist, fashion_mnist, cifar10, cifar100
from tensorflow.keras.optimizers import Adam, RMSprop,SGD
from keras.utils import np_utils
import cv2
import matplotlib.pyplot as plt
import numpy as np
import tensorflow.compat.v2 as tf
from tensorflow import keras
import os
import pickle
```

```
import pandas as pd
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.models import load_model
from keras.preprocessing import  image
#Connect with my drive
from google.colab import drive
drive.mount('/content/drive')
#Get dataset from drive
train_path = '/content/drive/MyDrive/AI/MonAn'
path_img = []
labels = ['BanhMi','Banhcuon','Banhtrangnuong','BunBo','Bunthitnuong','Comta
m','Goicuon','HuTieu','Miquang','Pho']
x_train = []
y_train = []
x_test = []
y_test =[]
# Create dataset
for i in labels:
  path = os.path.join(train_path,i)
  index_label = labels.index(i)
  for j in os.listdir(path):
      img_path = os.path.join(path, j)
      img = image.load_img(img_path, target_size=(300,300)) # độ phân giải t
ùy ae chọn nhé!
      img = img_to_array(img)
      img = img.reshape(300,300,3) #hàm reshape phải có cùng độ phân giải vớ
i target_size của nhé
      img = img.astype('float32')
      img = img/255
      x_train.append(img)
      y_train.append(index_label)

# Process data
x_train = np.array(x_train)
y_train = np.array(y_train)
y_train = np_utils.to_categorical(y_train)
#Create Model for train
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_unifo
rm', padding='same', input_shape=(300, 300, 3)))
model.add(MaxPooling2D(2,2))
model.add(Flatten())
model.add(Dense(256,activation='relu',kernel_initializer='he_uniform'))
model.add(Dense(10,activation='softmax'))
model.summary()
#Compile and Training
```

```
opt=SGD(learning_rate=0.01,momentum=0.9)
model.compile(loss='binary_crossentropy',optimizer=opt, metrics=['accuracy']
)
history = model.fit(x_train,y_train,epochs = 20)
#Draw plot and make evaluate
plt.subplot(2,1,1)
plt.plot(history.history['loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['train','validation'])
plt.show()

plt.subplot(2,1,2)
plt.plot(history.history['accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['train','validation'])
plt.show()
#Check detect
img = image.load_img('/content/Banhtrangnuong/Banhtrangnuong_8.jpg', target_
size=(300,300))
plt.imshow(img)
img = img_to_array(img)
img = img.reshape(1,300,300,3)
img = img.astype('float32')
img = img/255
print('This is: '+ labels[np.argmax(model.predict(img))])

img1 = image.load_img('/content/HuTieu/HuTieu_7.jpg', target_size=(300,300))

plt.imshow(img1)
img1 = img_to_array(img1)
img1 = img1.reshape(1,300,300,3)
img1 = img1.astype('float32')
img1 = img1/255
print('Day la '+ labels[np.argmax(model.predict(img1))])

img2 = image.load_img('/content/BunBo/BunBo_9.jpg', target_size=(300,300))
plt.imshow(img2)
img2 = img_to_array(img2)
img2 = img2.reshape(1,300,300,3)
img2 = img2.astype('float32')
img2 = img2/255
print('Day la '+ labels[np.argmax(model.predict(img2))])
```

2. Code trên Colab và kết quả của từng lệnh



+ Code + Text

RAM
Disk

Editing

```
[1] #Import libraries
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout,Flatten,Conv2D,MaxPooling2D
from keras.datasets import mnist, fashion_mnist, cifar10, cifar100
from tensorflow.keras.optimizers import Adam, RMSprop,SGD
from keras.utils import np_utils
import cv2
import matplotlib.pyplot as plt
import numpy as np
import tensorflow.compat.v2 as tf
from tensorflow import keras
import os
import pickle
import pandas as pd
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.models import load_model
from keras.preprocessing import  image
```

```
#Connect with my drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[3]
```
#Get dataset from drive
train_path = '/content/drive/MyDrive/AI/MonAn'
path_img = []
labels = ['BanhMi','Banhcuon','Banhtrangnuong','BunBo','Bunthitnuong','Comtam','Goicuon','HuTieu','Miquang',
x_train = []
y_train = []
x_test = []
y_test =[]
# Create dataset
for i in labels:
  path = os.path.join(train_path,i)
  index_label = labels.index(i)
  for j in os.listdir(path):
      img_path = os.path.join(path, j)
      img = image.load_img(img_path, target_size=(300,300)) # độ phân giải tùy ae chọn nhé!
      img = img_to_array(img)
      img = img.reshape(300,300,3) #hàm reshape phải có cùng độ phân giải với target_size của nhé
      img = img.astype('float32')
      img = img/255
      x_train.append(img)
      y_train.append(index_label)

# Process data
x_train = np.array(x_train)
y_train = np.array(y_train)
y_train = np_utils.to_categorical(y_train)
```

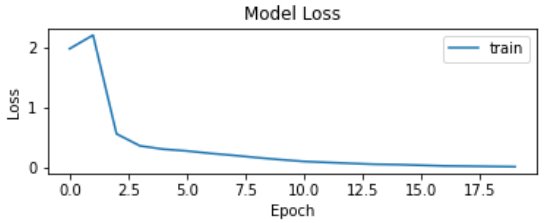
[4]
```
#Create Model for train
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', input_shape
model.add(MaxPooling2D(2,2))
model.add(Flatten())
model.add(Dense(256,activation='relu',kernel_initializer='he_uniform'))
model.add(Dense(10,activation='softmax'))
model.summary()
```

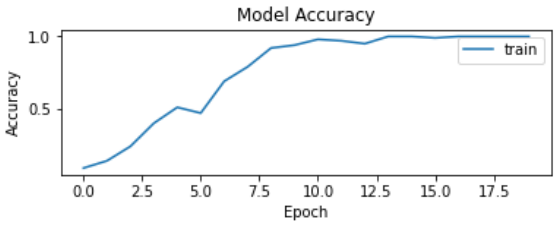
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 300, 300, 32)      896

 max_pooling2d (MaxPooling2D  (None, 150, 150, 32)     0
```

```python
#Compile and Training
opt=SGD(learning_rate=0.01,momentum=0.9)
model.compile(loss='binary_crossentropy',optimizer=opt, metrics=['accuracy'])
history = model.fit(x_train,y_train,epochs = 20)
```

```
Epoch 1/20
4/4 [==============================] - 12s 89ms/step - loss: 1.9712 - accuracy: 0.0900
Epoch 2/20
4/4 [==============================] - 0s 67ms/step - loss: 2.1966 - accuracy: 0.1400
Epoch 3/20
4/4 [==============================] - 0s 65ms/step - loss: 0.5580 - accuracy: 0.2400
Epoch 4/20
4/4 [==============================] - 0s 66ms/step - loss: 0.3583 - accuracy: 0.4000
Epoch 5/20
4/4 [==============================] - 0s 67ms/step - loss: 0.3034 - accuracy: 0.5100
Epoch 6/20
4/4 [==============================] - 0s 66ms/step - loss: 0.2739 - accuracy: 0.4700
Epoch 7/20
4/4 [==============================] - 0s 66ms/step - loss: 0.2339 - accuracy: 0.6900
Epoch 8/20
4/4 [==============================] - 0s 67ms/step - loss: 0.2004 - accuracy: 0.7900
Epoch 9/20
4/4 [==============================] - 0s 66ms/step - loss: 0.1629 - accuracy: 0.9200
Epoch 10/20
4/4 [==============================] - 0s 67ms/step - loss: 0.1293 - accuracy: 0.9400
Epoch 11/20
4/4 [==============================] - 0s 66ms/step - loss: 0.0997 - accuracy: 0.9800
Epoch 12/20
4/4 [==============================] - 0s 66ms/step - loss: 0.0832 - accuracy: 0.9700
Epoch 13/20
4/4 [==============================] - 0s 67ms/step - loss: 0.0672 - accuracy: 0.9500
Epoch 14/20
4/4 [==============================] - 0s 66ms/step - loss: 0.0530 - accuracy: 1.0000
Epoch 15/20
4/4 [==============================] - 0s 68ms/step - loss: 0.0474 - accuracy: 1.0000
Epoch 16/20
4/4 [==============================] - 0s 66ms/step - loss: 0.0364 - accuracy: 0.9900
Epoch 17/20
4/4 [==============================] - 0s 68ms/step - loss: 0.0251 - accuracy: 1.0000
Epoch 18/20
4/4 [==============================] - 0s 69ms/step - loss: 0.0213 - accuracy: 1.0000
Epoch 19/20
4/4 [==============================] - 0s 66ms/step - loss: 0.0164 - accuracy: 1.0000
Epoch 20/20
4/4 [==============================] - 0s 64ms/step - loss: 0.0121 - accuracy: 1.0000
```

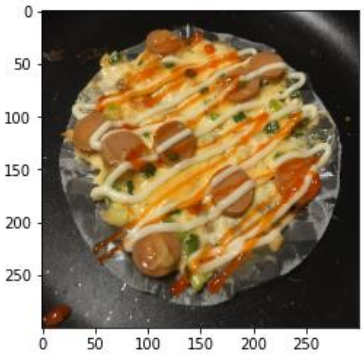
```
#Draw plot and make evaluate
plt.subplot(2,1,1)
plt.plot(history.history['loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['train','validation'])
plt.show()

plt.subplot(2,1,2)
plt.plot(history.history['accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['train','validation'])
plt.show()
```





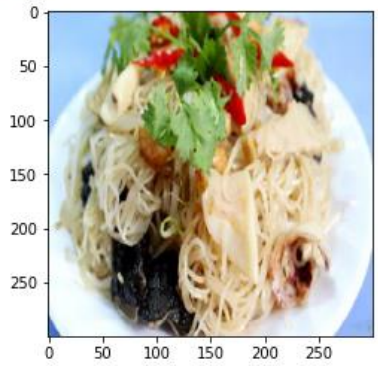

```
#Check detect
img = image.load_img('/content/drive/MyDrive/AI/MonAn/Banhtrangnuong/Banhtrangnuong_8.jpg', target_size=(300
plt.imshow(img)
img = img_to_array(img)
img = img.reshape(1,300,300,3)
img = img.astype('float32')
img = img/255
print('This is: '+ labels[np.argmax(model.predict(img))])
```

```
This is: Banhtrangnuong
```



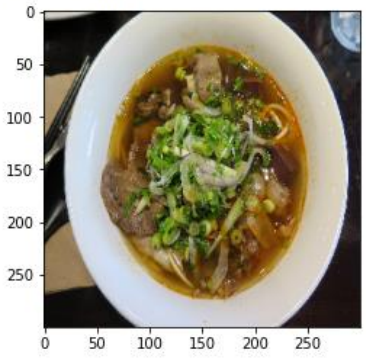
```
img1 = image.load_img('/content/drive/MyDrive/AI/MonAn/HuTieu/HuTieu_7.jpg', target_size=(300,300))
plt.imshow(img1)
img1 = img_to_array(img1)
img1 = img1.reshape(1,300,300,3)
img1 = img1.astype('float32')
img1 = img1/255
print('Day la '+ labels[np.argmax(model.predict(img1))])
```

```
Day la HuTieu
```




```
img2 = image.load_img('/content/drive/MyDrive/AI/MonAn/BunBo/BunBo_9.jpg', target_size=(300,300))
plt.imshow(img2)
img2 = img_to_array(img2)
img2 = img2.reshape(1,300,300,3)
img2 = img2.astype('float32')
img2 = img2/255
print('Day la '+ labels[np.argmax(model.predict(img2))])
```

```
Day la BunBo
```



VII. GITHUB UPLOAD

https://github.com/nhanguyene/HOMEWORK_AI_21_05_22