Mnist-ANN

May 19, 2022

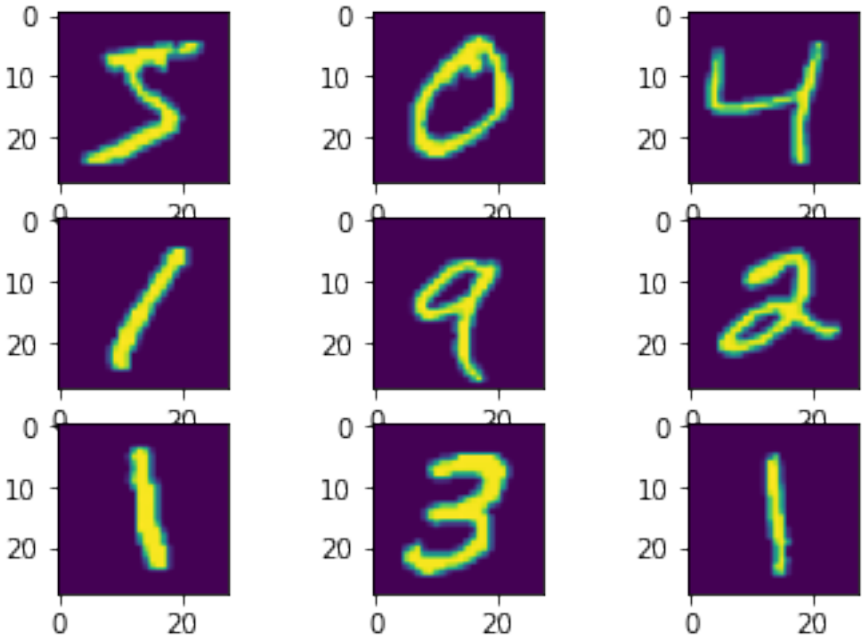
[ ]:
```
pip install keras
```

```
Requirement already satisfied: keras in /usr/local/lib/python3.7/dist-packages
(2.8.0)
```

[1]:
```
from keras.datasets import mnist
from tensorflow.keras.optimizers import RMSprop
(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
```

[2]:
```
import matplotlib.pyplot as plt
for i in range(9):
  plt.subplot(330+i+1)
  #330 la 3 hang 3 cot
  plt.imshow(x_train[i])
plt.show()
```



[3]:
```
x=x_test
```

[4]:
```
x_train=x_train.reshape(60000,784)
x_test=x_test.reshape(10000,784)
x_train=x_train.astype('float32')
x_test=x_test.astype('float32')
x_train/=255
x_test/=255
```

[5]:
```
from tensorflow.keras.utils import to_categorical
y_train=to_categorical(y_train,10)
y_test=to_categorical(y_test,10)
```

[6]:
```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Activation
from keras.layers import Dropout
```

[7]:
```
model = Sequential()
model.add(Dense(512,activation ='relu', input_shape = (784,))) # Layer 1
model.add(Dropout(0.2))

model.add(Dense(512,activation ='relu'))    #Layer 2(tín hiệu vào layer 2 bằng␣
 ↪tín hiệu vào layer 1)
model.add(Dropout(0.1))
```

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

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 dense (Dense)                (None, 512)               401920

 dropout (Dropout)            (None, 512)               0

 dense_1 (Dense)              (None, 512)               262656

 dropout_1 (Dropout)          (None, 512)               0

 dense_2 (Dense)              (None, 10)                5130

=================================================================
Total params: 669,706
Trainable params: 669,706
Non-trainable params: 0
_________________________________________________________________
```

[18]:
```
model.
 ↪compile(loss='categorical_crossentropy',optimizer=RMSprop(),metrics=['accuracy'])
history = model.
 ↪fit(x_train,y_train,batch_size=128,epochs=50,verbose=1,validation_data=(x_test,y_test))
```

```
Epoch 1/50
469/469 [==============================] - 10s 20ms/step - loss: 0.0032 -
accuracy: 0.9995 - val_loss: 0.4032 - val_accuracy: 0.9855
Epoch 2/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0034 -
accuracy: 0.9997 - val_loss: 0.3968 - val_accuracy: 0.9854
Epoch 3/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0031 -
accuracy: 0.9996 - val_loss: 0.3485 - val_accuracy: 0.9868
Epoch 4/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0043 -
accuracy: 0.9996 - val_loss: 0.3891 - val_accuracy: 0.9856
Epoch 5/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0032 -
accuracy: 0.9995 - val_loss: 0.3947 - val_accuracy: 0.9842
Epoch 6/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0036 -
accuracy: 0.9996 - val_loss: 0.3973 - val_accuracy: 0.9858
```

```
Epoch 7/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0046 -
accuracy: 0.9994 - val_loss: 0.3793 - val_accuracy: 0.9854
Epoch 8/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0035 -
accuracy: 0.9996 - val_loss: 0.3735 - val_accuracy: 0.9863
Epoch 9/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0017 -
accuracy: 0.9998 - val_loss: 0.3836 - val_accuracy: 0.9858
Epoch 10/50
469/469 [==============================] - 8s 16ms/step - loss: 0.0026 -
accuracy: 0.9997 - val_loss: 0.4053 - val_accuracy: 0.9863
Epoch 11/50
469/469 [==============================] - 8s 16ms/step - loss: 0.0029 -
accuracy: 0.9995 - val_loss: 0.4007 - val_accuracy: 0.9845
Epoch 12/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0037 -
accuracy: 0.9996 - val_loss: 0.4143 - val_accuracy: 0.9853
Epoch 13/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0030 -
accuracy: 0.9995 - val_loss: 0.3583 - val_accuracy: 0.9865
Epoch 14/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0017 -
accuracy: 0.9997 - val_loss: 0.3618 - val_accuracy: 0.9858
Epoch 15/50
469/469 [==============================] - 8s 18ms/step - loss: 0.0026 -
accuracy: 0.9995 - val_loss: 0.3599 - val_accuracy: 0.9861
Epoch 16/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0024 -
accuracy: 0.9997 - val_loss: 0.3599 - val_accuracy: 0.9855
Epoch 17/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0018 -
accuracy: 0.9997 - val_loss: 0.3791 - val_accuracy: 0.9854
Epoch 18/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0025 -
accuracy: 0.9997 - val_loss: 0.3637 - val_accuracy: 0.9855
Epoch 19/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0034 -
accuracy: 0.9997 - val_loss: 0.3836 - val_accuracy: 0.9850
Epoch 20/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0030 -
accuracy: 0.9996 - val_loss: 0.3874 - val_accuracy: 0.9849
Epoch 21/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0029 -
accuracy: 0.9998 - val_loss: 0.3885 - val_accuracy: 0.9855
Epoch 22/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0036 -
accuracy: 0.9996 - val_loss: 0.4053 - val_accuracy: 0.9861
```

```
Epoch 23/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0046 -
accuracy: 0.9995 - val_loss: 0.4144 - val_accuracy: 0.9862
Epoch 24/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0016 -
accuracy: 0.9997 - val_loss: 0.4092 - val_accuracy: 0.9870
Epoch 25/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0041 -
accuracy: 0.9997 - val_loss: 0.3779 - val_accuracy: 0.9862
Epoch 26/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0030 -
accuracy: 0.9995 - val_loss: 0.3930 - val_accuracy: 0.9852
Epoch 27/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0023 -
accuracy: 0.9996 - val_loss: 0.4153 - val_accuracy: 0.9852
Epoch 28/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0019 -
accuracy: 0.9997 - val_loss: 0.4198 - val_accuracy: 0.9850
Epoch 29/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0026 -
accuracy: 0.9997 - val_loss: 0.4010 - val_accuracy: 0.9852
Epoch 30/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0024 -
accuracy: 0.9996 - val_loss: 0.4166 - val_accuracy: 0.9852
Epoch 31/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0034 -
accuracy: 0.9995 - val_loss: 0.4222 - val_accuracy: 0.9852
Epoch 32/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0031 -
accuracy: 0.9996 - val_loss: 0.4201 - val_accuracy: 0.9850
Epoch 33/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0012 -
accuracy: 0.9998 - val_loss: 0.4202 - val_accuracy: 0.9858
Epoch 34/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0030 -
accuracy: 0.9996 - val_loss: 0.3954 - val_accuracy: 0.9870
Epoch 35/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0025 -
accuracy: 0.9997 - val_loss: 0.4453 - val_accuracy: 0.9852
Epoch 36/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0032 -
accuracy: 0.9998 - val_loss: 0.4263 - val_accuracy: 0.9847
Epoch 37/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0016 -
accuracy: 0.9997 - val_loss: 0.4304 - val_accuracy: 0.9855
Epoch 38/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0025 -
accuracy: 0.9997 - val_loss: 0.4086 - val_accuracy: 0.9849
```

```
Epoch 39/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0031 -
accuracy: 0.9996 - val_loss: 0.3765 - val_accuracy: 0.9862
Epoch 40/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0023 -
accuracy: 0.9997 - val_loss: 0.3963 - val_accuracy: 0.9847
Epoch 41/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0022 -
accuracy: 0.9997 - val_loss: 0.4200 - val_accuracy: 0.9852
Epoch 42/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0033 -
accuracy: 0.9995 - val_loss: 0.4029 - val_accuracy: 0.9853
Epoch 43/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0023 -
accuracy: 0.9997 - val_loss: 0.3653 - val_accuracy: 0.9864
Epoch 44/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0039 -
accuracy: 0.9996 - val_loss: 0.3908 - val_accuracy: 0.9857
Epoch 45/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0016 -
accuracy: 0.9997 - val_loss: 0.3934 - val_accuracy: 0.9864
Epoch 46/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0041 -
accuracy: 0.9995 - val_loss: 0.4195 - val_accuracy: 0.9859
Epoch 47/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0025 -
accuracy: 0.9997 - val_loss: 0.3674 - val_accuracy: 0.9868
Epoch 48/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0040 -
accuracy: 0.9996 - val_loss: 0.3898 - val_accuracy: 0.9866
Epoch 49/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0018 -
accuracy: 0.9997 - val_loss: 0.3710 - val_accuracy: 0.9853
Epoch 50/50
469/469 [==============================] - 8s 17ms/step - loss: 0.0021 -
accuracy: 0.9998 - val_loss: 0.3879 - val_accuracy: 0.9850
```

[19]:
```
score = model.evaluate(x_test,y_test,verbose=1)
```

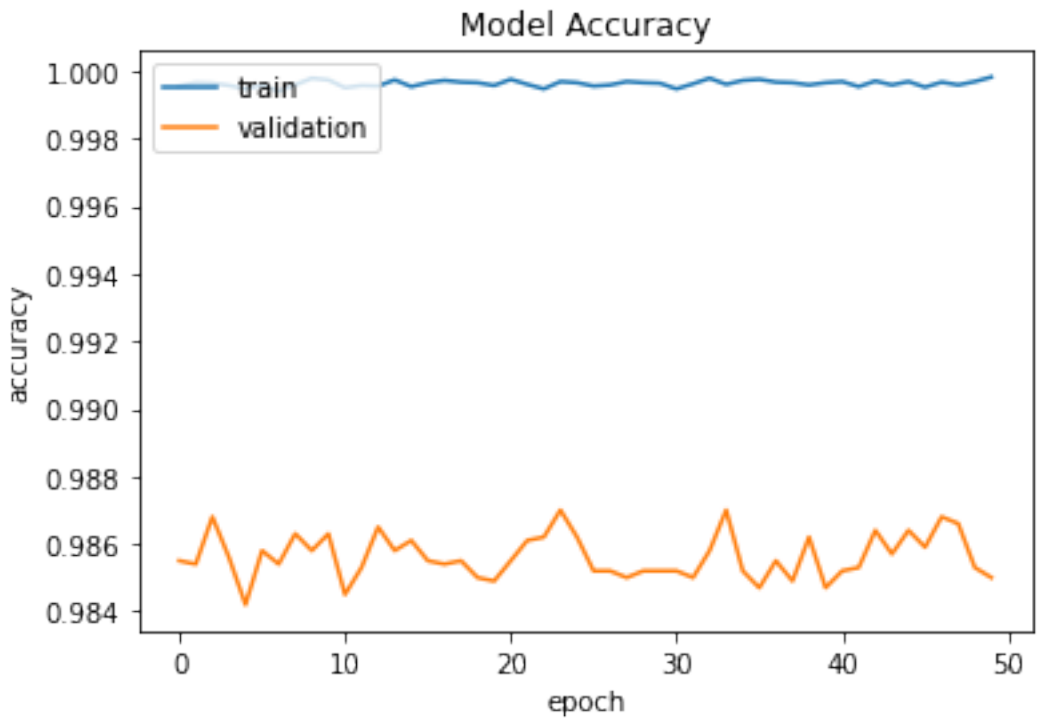
```
313/313 [==============================] - 1s 3ms/step - loss: 0.3879 -
accuracy: 0.9850
```

[20]:
```
print('Test loss=',score[0])
print('Test accuracy=',score[1])
```

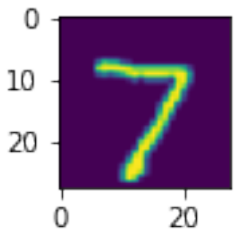
```
Test loss= 0.38790398836135864
Test accuracy= 0.9850000143051147
```

[22]:
```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train','validation'],loc='upper left')
```

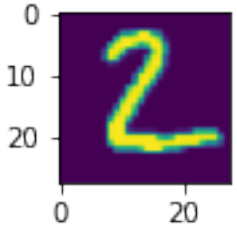
[22]:
```
<matplotlib.legend.Legend at 0x7fd787877510>
```



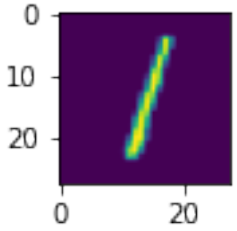
[24]:
```
import numpy as np
y_pred=model.predict(x_test)
for i in range(9):
  plt.subplot(330+i+1) #330: 3 hang 3 cot, stt di tu trai sang phai tren␣
 ↪duoi,i=0 thi 331 phan tu tuong tac la so 1
  plt.imshow(x[i])
  plt.show()
  print(np.round(y_pred[i]))
```



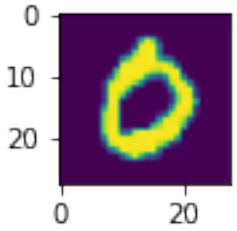

```
[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
```



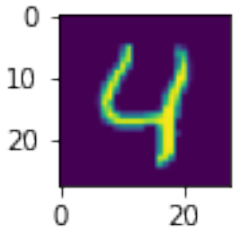

```
[0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
```



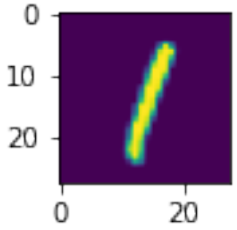

```
[0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
```



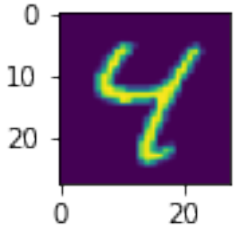

```
[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```



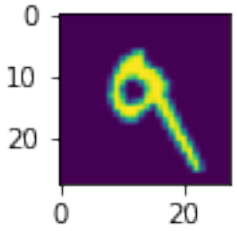

```
[0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
```



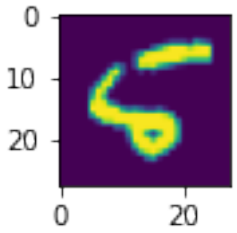

```
[0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
```




```
[0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
```




```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
```




```
[0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
```

[ ]:
```
from google.colab import drive
drive.mount('/content/drive')
!wget -nc https://raw.githubusercontent.com/brpy/colab-pdf/master/colab_pdf.py
from colab_pdf import colab_pdf
colab_pdf('Mnist-ANN.ipynb')
```

```
Mounted at /content/drive
--2022-05-19 06:01:29--  https://raw.githubusercontent.com/brpy/colab-
pdf/master/colab_pdf.py
Resolving raw.githubusercontent.com (raw.githubusercontent.com)…
185.199.111.133, 185.199.109.133, 185.199.108.133, …
Connecting to raw.githubusercontent.com
(raw.githubusercontent.com)|185.199.111.133|:443… connected.
HTTP request sent, awaiting response… 200 OK
Length: 1864 (1.8K) [text/plain]
Saving to: ‘colab_pdf.py’

colab_pdf.py        100%[===================>]   1.82K  --.-KB/s    in 0s

2022-05-19 06:01:29 (23.8 MB/s) - ‘colab_pdf.py’ saved [1864/1864]


WARNING: apt does not have a stable CLI interface. Use with caution in scripts.


WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Extracting templates from packages: 100%
```