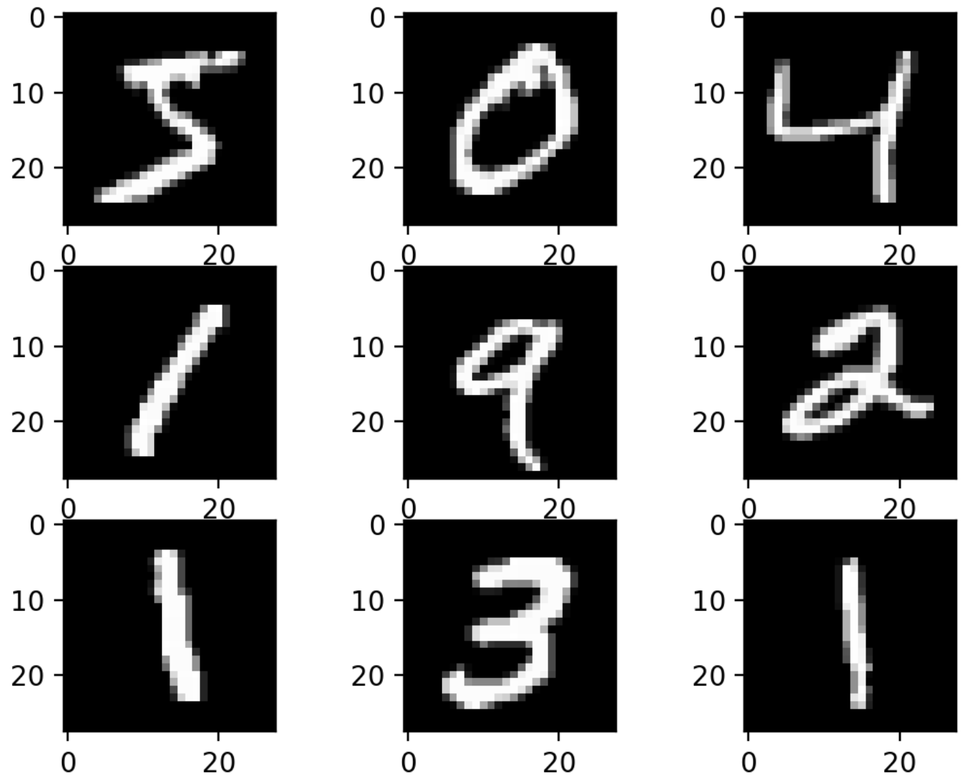
MNIST PREDCTION



MỤC LỤC

Tổng quan

Giới thiệu

- MNIST-là bộ dữ liệu chữ số viết tay các số từ 0-9 được tích hợp sẵn trong Keras. Tổng tập này có 70000 samples, là các ảnh có kích thước 28x28 pixel, với chiều kênh bằng 1(ảnh đen trắng)
- Mỗi ảnh đầu vào sẽ là một ma trận 28x28, với giá trị của các phần tử chạy từ 0-255(độ sáng của ảnh)

=> Ta sẽ xây dựng một Model để dự đoán xem input đưa vào là số mấy

Xây dựng model

Xác định bài toán

- Input: Ảnh đen trắng với kích thước mỗi ảnh là 28x28 pixel
- Output: Dãy 10 xác suất tương ứng với xác suất xuất hiện của 10 chứ số(0-9)
- Predict: Ta sẽ lấy max của list kể trên, từ đó đưa ra dự đoán số tương ứng

Xây dựng model

Các thư viện sử dụng

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.utils import np_utils
from keras.datasets import mnist
```

Xây dựng model

Tiền xử lí dữ liệu, chia các tập train, val, test

```
(X_train, y_train), (X_test, y_test) = mnist.load_data()
X_val, y_val = X_train[50000:60000,:], y_train[50000:60000]
X_train, y_train = X_train[:50000,:], y_train[:50000]
print(X_train.shape)
print(X_val.shape)
```

Hiệu chỉnh chiều dữ liệu đầu vào: Trong bài này là ảnh xám, kích thước 28x28 pixel

```
X_train = X_train.reshape(X_train.shape[0], 28, 28, 1)

X_val = X_val.reshape(X_val.shape[0], 28, 28, 1)
X_test = X_test.reshape(X_test.shape[0], 28, 28, 1)
```

Xây dựng model

Feature Scaling

```
X_train = X_train/255.
X_test = X_test/255.
X_val = X_val/255.
```

Ta sẽ chuẩn hóa các giá trị ảnh về khoảng từ 0-1

Xử lí output bằng one hot encoding

```
Y_train = np_utils.to_categorical(y_train, 10)
Y_val = np_utils.to_categorical(y_val, 10)
Y_test = np_utils.to_categorical(y_test, 10)
print('Dữ liệu y ban đầu ', y_train[0])
print('Dữ liệu y sau one-hot encoding ',Y_train[0])
```

```
Dữ liệu y ban đầu  5
Dữ liệu y sau one-hot encoding  [0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
```

Xây dựng model

Xây dựng model

```
# Định nghĩa model
model = Sequential()

# Thêm Convolutional layer với 32 kernel, kích thước kernel 3*3
# dùng hàm sigmoid làm activation và chỉ rõ input_shape cho layer đầu tiên
model.add(Conv2D(32, (3, 3), padding='same', activation='relu', input_shape=(28,28,1)))

# Thêm Convolutional layer
model.add(Conv2D(32, (3, 3), activation='relu'))

# Thêm Max pooling layer
model.add(MaxPooling2D())

# Flatten layer chuyển từ tensor sang vector
model.add(Flatten())

# Thêm Fully Connected layer với 128 nodes và dùng hàm sigmoid
model.add(Dense(128, activation='relu'))

# Output layer với 10 node và dùng softmax function để chuyển sang xác xuất.
model.add(Dense(10, activation='softmax'))

model.summary()
```

ở đây ra xây dựng một model với 6 lớp. Ở lớp đầu tiên ta sử dụng Convolutional layer với 32 kernel, kích thước kernel là 3*3, sử dụng hàm relu. Tiếp đến ta thêm một lớp Convolutional layer thứ 2 cũng sử dụng hàm relu và sau đó ta thêm 1 lớp pooling layer để gộp những đặc trưng và giảm chiều dữ liệu.

Sau đó, ta reshape lại đầu ra thành 1 vector, áp dụng Fully Connected layer sử dụng ReLu. Và cuối cùng ta sử dụng 1 lớp có 10 units, hàm softmax để cho ta kết quả là 10 xác suất tương ứng với khả năng xuất hiện của 10 số

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 32)        320

 conv2d_1 (Conv2D)           (None, 26, 26, 32)        9248

 max_pooling2d (MaxPooling2D  (None, 13, 13, 32)       0
 )

 flatten (Flatten)           (None, 5408)              0

 dense (Dense)               (None, 128)               692352

 dense_1 (Dense)             (None, 10)                1290

=================================================================
Total params: 703,210
Trainable params: 703,210
Non-trainable params: 0
_________________________________________________________________
```

Xây dựng model

Compile và fit model

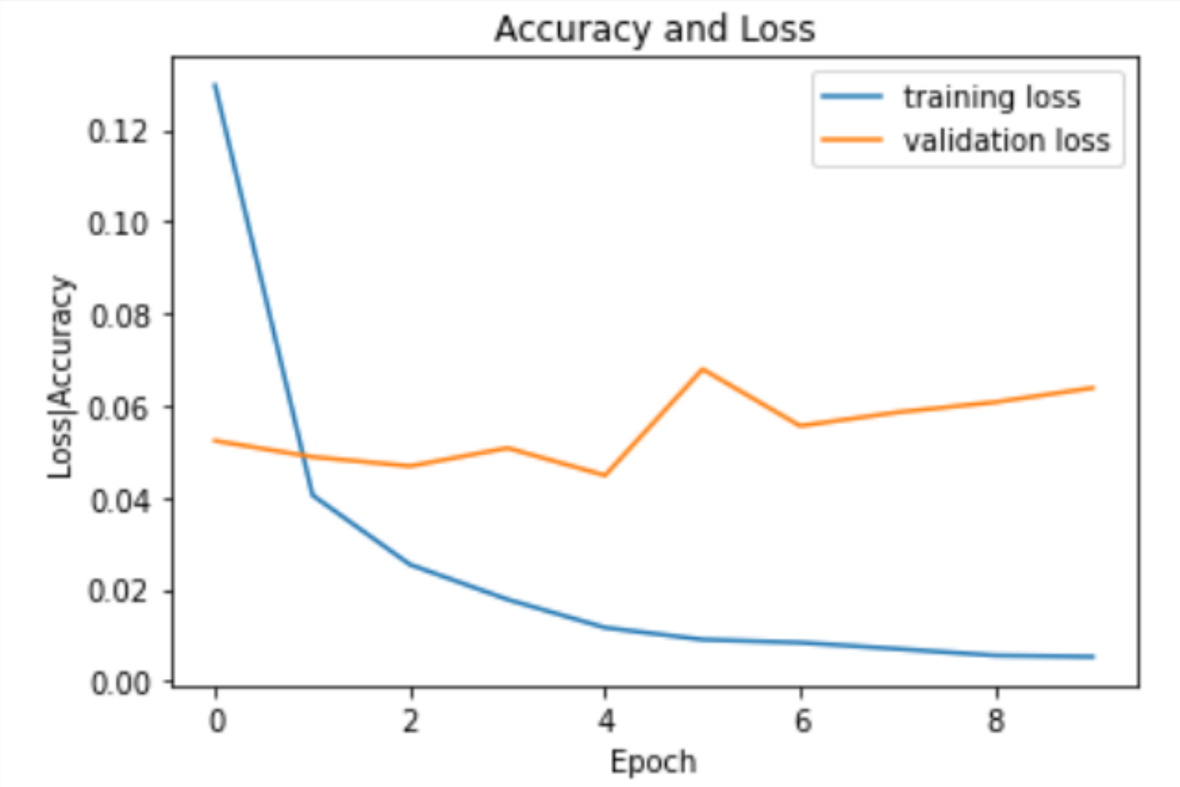
```
[7]  model.compile(loss='categorical_crossentropy',
                   optimizer='adam',
                   metrics=['accuracy'])
```

```
H = model.fit(X_train, Y_train, validation_data=(X_val, Y_val),
              batch_size=32, epochs=10, verbose=1)
```

```
Epoch 1/10
1563/1563 [==============================] - 7s 4ms/step - loss: 0.1297 - accuracy: 0.9605 - val_loss: 0.0523 - val_accuracy: 0.9856
Epoch 2/10
1563/1563 [==============================] - 7s 4ms/step - loss: 0.0405 - accuracy: 0.9873 - val_loss: 0.0488 - val_accuracy: 0.9852
Epoch 3/10
1563/1563 [==============================] - 6s 4ms/step - loss: 0.0254 - accuracy: 0.9915 - val_loss: 0.0468 - val_accuracy: 0.9871
Epoch 4/10
1563/1563 [==============================] - 6s 4ms/step - loss: 0.0178 - accuracy: 0.9943 - val_loss: 0.0508 - val_accuracy: 0.9854
Epoch 5/10
1563/1563 [==============================] - 6s 4ms/step - loss: 0.0117 - accuracy: 0.9961 - val_loss: 0.0448 - val_accuracy: 0.9893
Epoch 6/10
1563/1563 [==============================] - 7s 4ms/step - loss: 0.0091 - accuracy: 0.9968 - val_loss: 0.0679 - val_accuracy: 0.9851
Epoch 7/10
1563/1563 [==============================] - 7s 5ms/step - loss: 0.0084 - accuracy: 0.9969 - val_loss: 0.0556 - val_accuracy: 0.9889
Epoch 8/10
1563/1563 [==============================] - 7s 4ms/step - loss: 0.0071 - accuracy: 0.9978 - val_loss: 0.0585 - val_accuracy: 0.9892
Epoch 9/10
1563/1563 [==============================] - 7s 4ms/step - loss: 0.0056 - accuracy: 0.9982 - val_loss: 0.0607 - val_accuracy: 0.9876
Epoch 10/10
1563/1563 [==============================] - 7s 4ms/step - loss: 0.0054 - accuracy: 0.9982 - val_loss: 0.0638 - val_accuracy: 0.9881
```

Đánh giá model

Train Loss và Val Loss



Đánh giá model

```
score = model.evaluate(X_test, Y_test, verbose=1)
print(score)
```

```
313/313 [==============================] - 1s 3ms/step - loss: 0.0536 - accuracy: 0.9890
[0.053627606481313705, 0.9890000224113464]
```

Kiểm thử model

```
plt.imshow(X_test[0].reshape(28,28), cmap='gray')
print(X_test[0].shape)
y_predict = model.predict(X_test[0].reshape(1,28,28,1))
print(y_predict)
print('Giá trị dự đoán: ', np.argmax(y_predict))
```

```
(28, 28, 1)
[[1.2373029e-20 2.3665148e-14 2.1154419e-15 3.1412714e-13 7.5220186e-21
  1.3151884e-21 2.7783005e-22 1.0000000e+00 1.4435562e-18 5.7055741e-15]]
Giá trị dự đoán:  7
```