Mục lục

Báo cáo đồ án 1

1. Đặt vấn đề

1.1 Lý do chọn đề tài

Trong thời đại 4.0 ngày nay, công nghệ càng ngày càng trở nên dễ tiếp cận đối với mỗi cá nhân, kéo theo sự nảy sinh của rất nhiều loại dữ liệu trên internet. Chính bởi vậy, nhu cầu dành cho ngành trí tuệ nhân tạo với học máy càng ngày càng đi lên để có thể tận dụng được tối đa nguồn dữ liệu đó. Bởi vậy, em đã quyết định lựa chọn bài toán nhận diện chữ số viết tay để (một bài toán tương đối cơ bản trong lĩnh vực này) với mục đích để xây dựng nền tảng cơ bản về cả kiến thức với cả kĩ năng để sau này có thể quen dần với những bài toán nâng cao hơn.

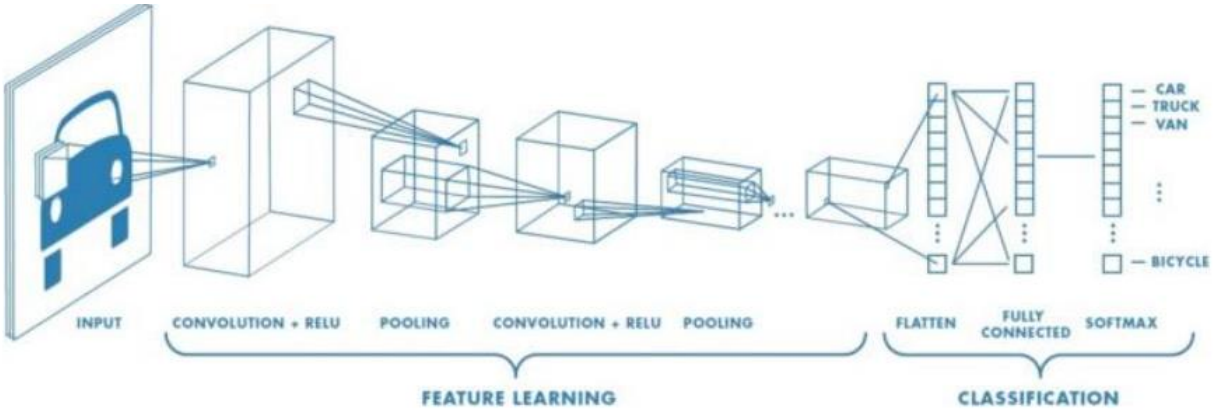
1.2 Mô tả bài toán

Từ tập dự liệu chữ số viết tay của MNIST, em sẽ xây dựng một mô hình học máy để có thể dự đoán được chữ số mà mình vẽ trên cửa sổ giao diện mình tạo ra.

2. Xây dựng dự án

2.1 Cơ sở lý thuyết

Sử dụng mạng tích chập (Convutional neuron network – CNN)



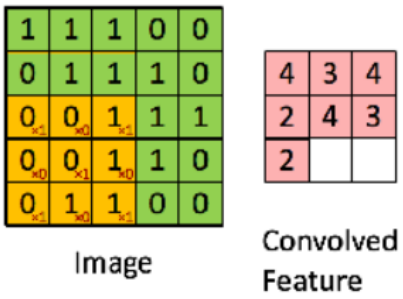
Ý tưởng cơ bản của CNN là việc mình chia nhỏ các khung ảnh ra, đưa nó vào trong những bộ lọc khác nhau để hi vọng máy sẽ tìm thấy những đặc điểm riêng biệt của các vùng ảnh để từ đó đưa ra dự đoán. Những khái niệm cần lưu ý khi đề cập tới mạng CNN là lớp tích chập, lớp gộp và lớp đầy đủ. Việc mô hình học thì sẽ sử dụng các kĩ thuật trong mạng nơ ron bình thường: tịnh tiến

(feed forward) và kiểm tra ngược (back propagation). Những kĩ thuật trên đều dựa trên thuật toán stochastic gradient descent (SGD).

2.2.1 Lớp tích chập (convutional layer)

Đầu vào của ta sẽ đi qua những đầu lọc khác nhau với những bước nhảy khác nhau để có thể mã hóa thông tin. Chúng ta làm như vậy với mục đích thứ nhất là thu nhỏ được chiều dữ liệu với cả bảo toàn vị trị của một điểm dữ liệu so với những điểm dữ liệu xung quanh nó (điều mà khi mình duỗi thẳng rồi cho vào lớp mạng đầy đủ không thể làm được).

Khi mà ta lọc thì ta có thể chọn cách mà bộ lọc lướt qua cái input như thế nào. Đó chinh là bước nhảy (stride) của bộ lọc. Ngoài ra, nhiều khi khi mình lọc thì có thể filter bị chèn ra ngoại khỏi đầu vào. Lúc này chúng ta có thể thêm padding cho input để tránh cho điều này xảy ra.

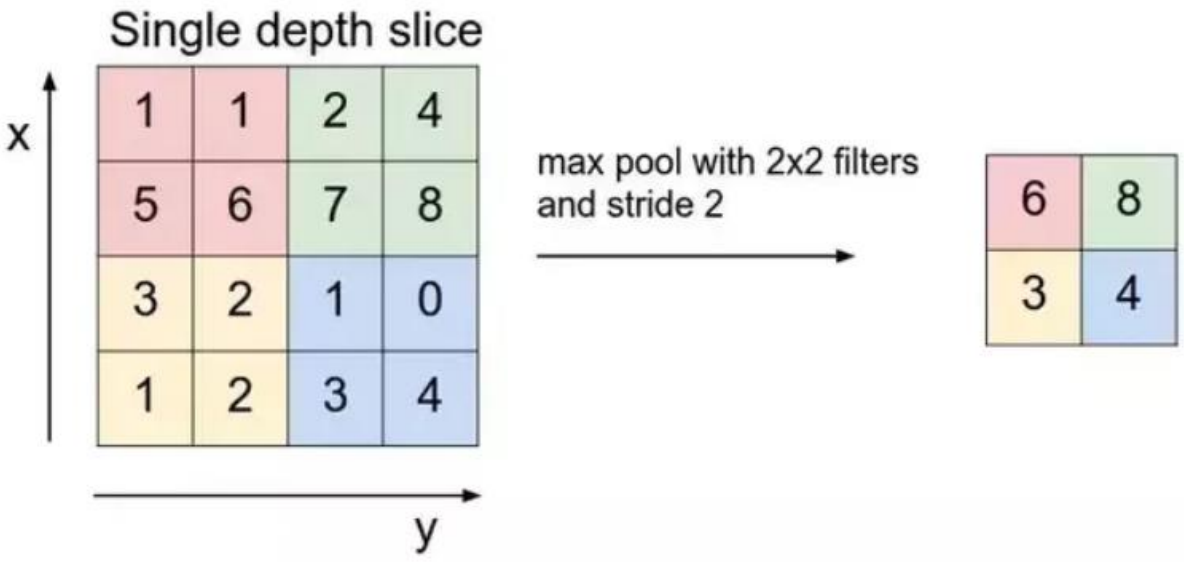


Bộ lọc bằng 3 với bước nhảy bằng 1
(Mảng màu xanh input, mảng màu vàng là filter, mảng màu hồng là output)

Sau khi đã có được đầu ra, ta sẽ cho nó chạy qua một hàm phi tuyến (activation function) để có thể trích xuất được ra những đặc điểm phi tuyến của dữ liệu. Các hàm kích hoạt thông dụng là: sigmoid, tanh,… Nhưng hàm đơn giản mà khá hiệu quả là hàm Relu(x) = max(0,x)

2.2.2 Lớp gộp (pooling layer)

Sau khi đã nhận được kết quả sau khi đưa lớp tích chập vào trong một hàm kích hoạt phi tuyến, ta sẽ có một hàm Pooling để chích ra đặc trưng mình cần. Điều này sẽ giúp ta giảm được chiều của dữ liệu mà hạn chế mất mát thông tin. Có khá nhiều cách để pooling lại như lấy trung bình, cộng tổng, lấy giá trị lớn nhất, trong các bài toán liên quan đến dữ liệu hình ảnh này thì lấy max là cách gộp thường được sử dụng nhất.



2.2.3 Lớp kết nối đầy đủ (fully-connected layer)

Sau khi đã có được kết quả từ những tầng trước đó, ta sẽ duỗi thẳng ma trận điểm ảnh ra thành một véc tơ và thực hiện dự đoán trên đấy. Trong bước này thì người ta có một kĩ thuật tránh overfitting là drop out: trong từng vòng lặp ngẫu nhiên loại bỏ một số điểm để tất cả các nút đều có thông tin về đặc trưng nào đó. Kết quả cuối cùng sẽ được đưa vào một hàm softmax để đưa ra xác suất xem điểm đó nằm trong nhãn nào.

2.2 Xử lý dữ liệu

Sử dụng bộ dữ liệu chữ số viết tay của tập MNIST bao gồm 6000 ảnh cho tập huấn luyện và 1000 ảnh cho tập test. Tất cả các ảnh đều đã được gán nhãn sẵn và có kích thước là 28 x 28. X_train là kích thước tập train, y_train là label của tập đấy. x_test và y_test cũng là tương tự của tập test.

```
# test set 10000 ảnh 0-9
# data set 60000 ảnh 0-9
(x_train,y_train),(x_test,y_test) = mnist.load_data()
```

✓ 0.3s

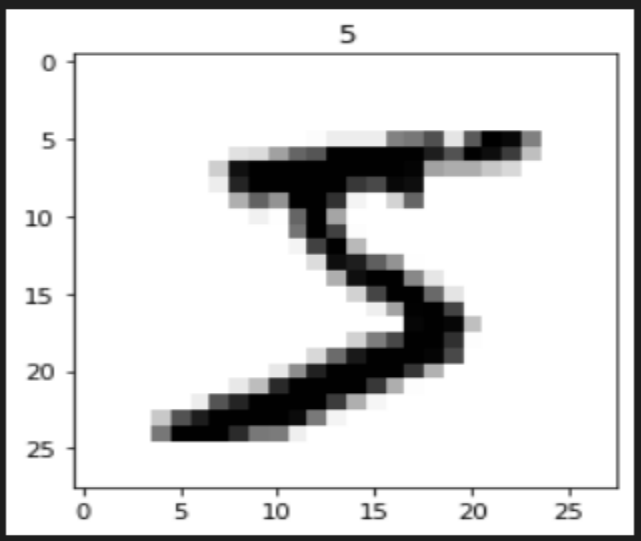
tiền xử lý dữ liệu

```
# kích thước ảnh 28 x 28
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(60000, 28, 28)
(60000,)
(10000, 28, 28)
(10000,)
```

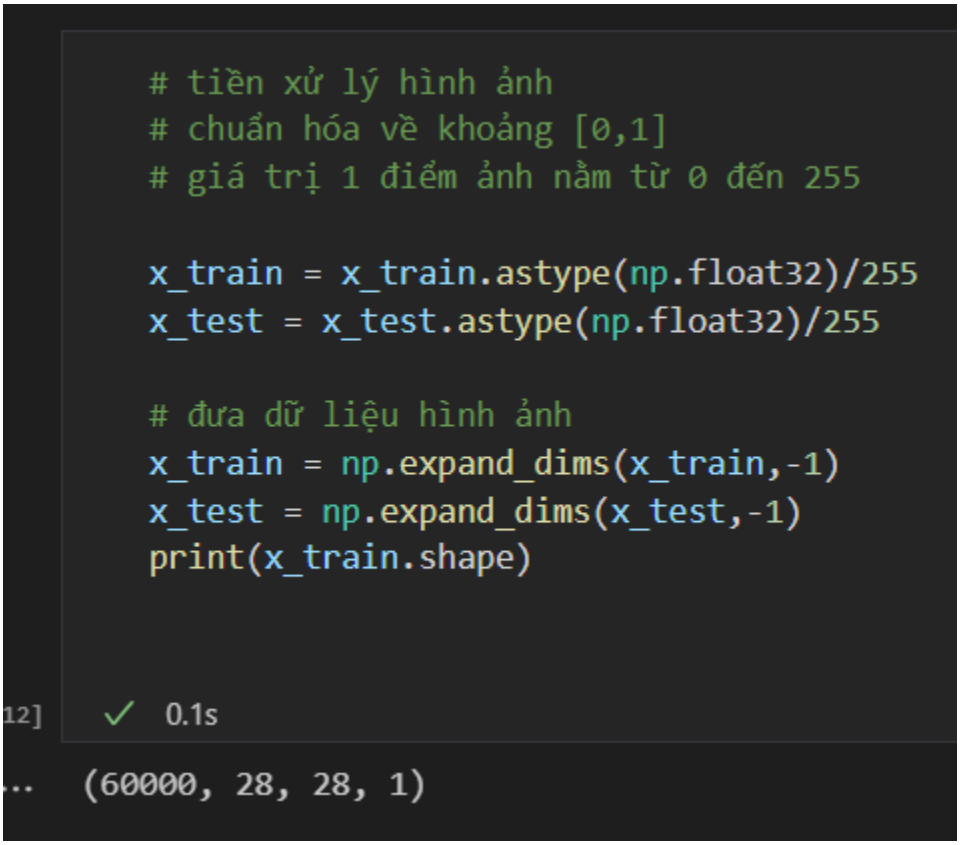
```
#plot train image
def plot_input_image(i):
    plt.imshow(x_train[i],cmap = 'binary')
    plt.title(y_train[i])
    plt.show()

for i in range(1):
    plot_input_image(i)
```

✓ 0.1s



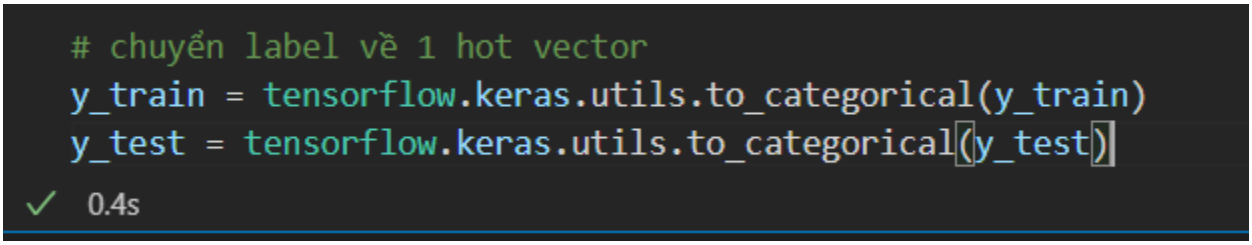
Hình ảnh của 1 điểm dữ liệu

Trước khi được đưa vào trong mạng CNN, các điểm ảnh trong các x đã được chuẩn hóa từ ban đầu nằm trong khoảng (0,255) xuống còn (0,1) để mục đích tăng tốc độ hội tụ khi mô hình học đồng thời giảm overfitting.



Ta để thêm 1 chiều nữa trong tập dữ liệu x để thể hiện chiều sâu của mỗi điểm dữ liệu để khớp với input của mạng CNN.

Cuối cùng ta chuyển toàn bộ các label về dưới dạng one hot vector. One hot vector là biểu diễn nhãn y dưới dạng vector có số chiều bằng số nhãn. Nếu y ban đầu có giá trị bằng bao nhiêu thì giá trị cột tương ứng với nhãn đấy bằng 1 và tất cả các trưởng còn lại bằng 0.



2.3 Huấn luyện mô hình

```
model = Sequential()
model.add(Conv2D(32,(3,3),input_shape = (28,28,1),activation='relu'))
model.add(MaxPool2D(2,2))

model.add(Conv2D(64,(3,3),activation='relu'))
model.add(MaxPool2D(2,2))

model.add(Flatten())
model.add(Dropout(0.25))
#10 class => 10
model.add(Dense(10,activation='softmax'))
```

Mô hình CNN mà em đã xây dựng là gồm 2 lớp tích chập.

Lớp tích chập đầu tiên có đầu vào là 28x28x1, đi qua 32 filter có kích thước 3x3, có thể viết dưới dạng: 3x3x32. Nó sẽ sinh ra output có kích thước 26x26x32. Ta cho nó đi qua hàm kích hoạt relu, rồi cho qua lớp gộp được output có dạng 16x16x32.

Lớp tích chập đầu tiên có đầu vào là 16x16x32, đi qua 64 filter có kích thước 3x3, có thể viết dưới dạng: 3x3x64. Nó sẽ sinh ra output có kích thước 11x11x64. Ta cho nó đi qua hàm kích hoạt relu, rồi cho qua lớp gộp được output có dạng 5x5x64.

Lớp cuối cùng ta sẽ dải thẳng ma trận kia thành một vector có số chiều là 5x5x64 = 1600. Rồi đưa vào trong lớp kết nối đầy đủ có tỷ lệ dropout là 0.25.

```
model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_6 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 11, 11, 64)        18496
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
flatten_3 (Flatten)          (None, 1600)              0
_________________________________________________________________
dropout_3 (Dropout)          (None, 1600)              0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                16010
=================================================================
Total params: 34,826
Trainable params: 34,826
Non-trainable params: 0
```

Hàm mất mát của ta sẽ là hàm cross entropy, sử dụng Stochaic Gradient Descend với cách chọn learning rate theo adam để học tham số.

```
model.compile(optimizer='adam',
              loss = keras.losses.categorical_crossentropy,
              metrics='accuracy')
```

Ngoài ra để tránh overfiting, ta còn áp dụng kĩ thuật early stopping (Nếu sau 4 lần liên tiếp độ chinh xác không vượt quá min_deta thì dừng). Model check point để lưu lại mô hình có kết quả tốt nhất

```
# callback
from keras.callbacks import EarlyStopping, ModelCheckpoint

# earlystopping
# monitor : thang đánh giá
# min_delta: độ tăng tối thiểu để đc coi là chất lượng tăng
# patience: sau 4 lần epoch mà chất lượng models ko tăng thì dừng

es = EarlyStopping(monitor = 'val_accuracy',min_delta = 0.01, patience = 4,verbose = 1)

# model checkpoint
mc = ModelCheckpoint('./mymodel.h5',monitor='val_accuracy',verbose = 1,save_best_only = True)

cb = [es,mc]
# cb =[mc]
```

Ta huấn luyện mô hình với 20 lần lặp, chia 30% tập huấn luyện thành tập đánh giá để học các tham số.

```
hi = model.fit(x_train,y_train,epochs = 20,validation_split=0.3,callbacks = cb)
```

```
Epoch 1/20
1313/1313 [==============================] - 19s 14ms/step - loss: 0.0094 - accuracy: 0.9966 - val_loss: 0.0524 - val_accuracy: 0.9892

Epoch 00001: val_accuracy improved from -inf to 0.98917, saving model to .\mymodel.h5
Epoch 2/20
1313/1313 [==============================] - 19s 14ms/step - loss: 0.0093 - accuracy: 0.9967 - val_loss: 0.0451 - val_accuracy: 0.9898

Epoch 00002: val_accuracy improved from 0.98917 to 0.98978, saving model to .\mymodel.h5
Epoch 3/20
1313/1313 [==============================] - 18s 14ms/step - loss: 0.0077 - accuracy: 0.9973 - val_loss: 0.0504 - val_accuracy: 0.9902

Epoch 00003: val_accuracy improved from 0.98978 to 0.99017, saving model to .\mymodel.h5
Epoch 4/20
1313/1313 [==============================] - 19s 14ms/step - loss: 0.0080 - accuracy: 0.9971 - val_loss: 0.0471 - val_accuracy: 0.9898

Epoch 00004: val_accuracy did not improve from 0.99017
Epoch 5/20
1313/1313 [==============================] - 17s 13ms/step - loss: 0.0094 - accuracy: 0.9965 - val_loss: 0.0437 - val_accuracy: 0.9904
```

Kết quả tốt nhất sẽ được lưu vào file mymodel.h5

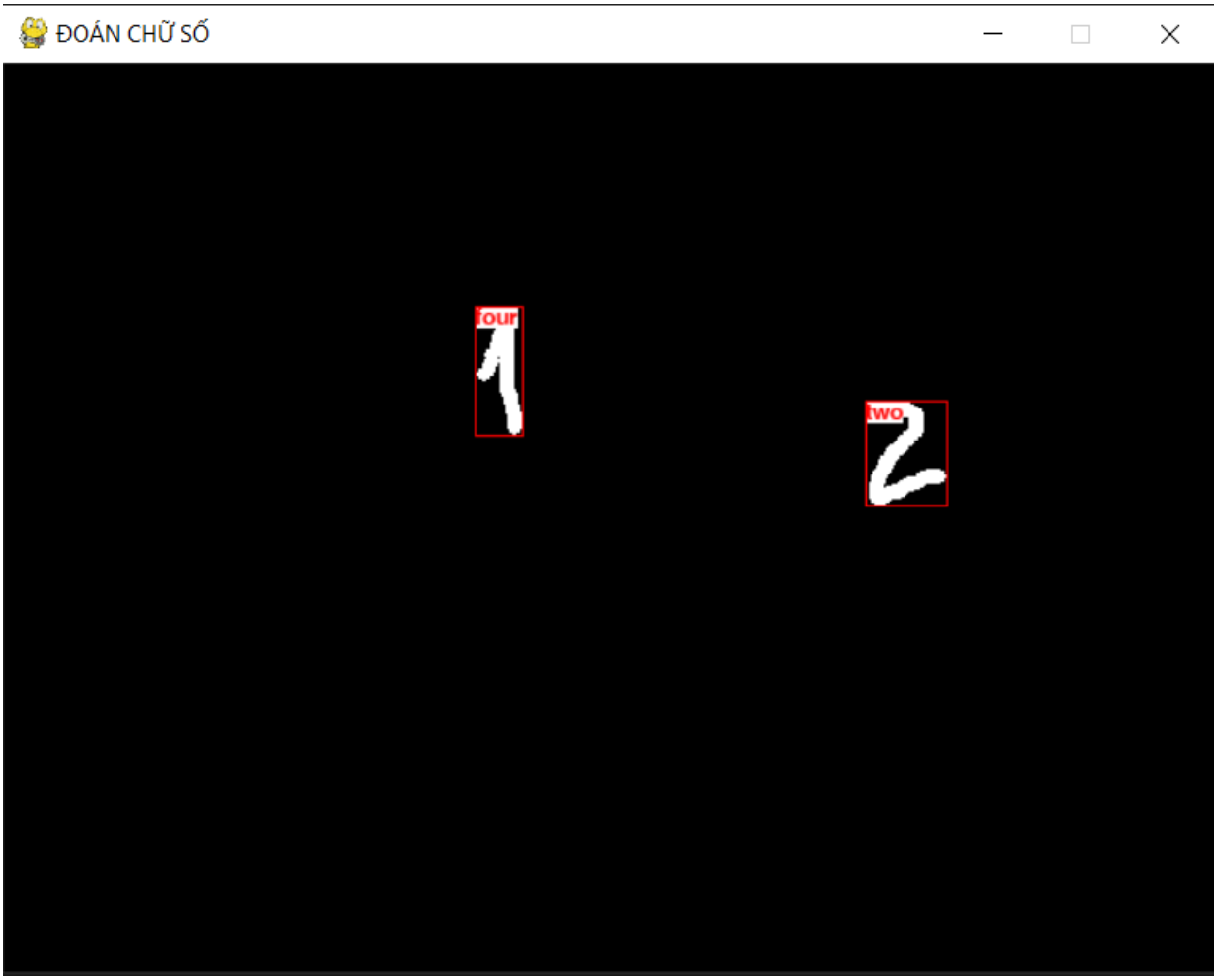
```
model_S = keras.models.load_model('D:\project1\mymodel.h5')
```

Ta có thể thấy được là độ chính xác của mô hình lên tới 0,9915 và hàm mất mát chỉ có giá trị là 0.036. Kết quả được đánh giá là sử dụng tập test để kiểm tra.

```
score = model_S.evaluate(x_test,y_test)
print(score)
```

```
313/313 [==============================] - 1s 3ms/step - loss: 0.0306 - accuracy: 0.9915
[0.030562246218323708, 0.9915000200271606]
```

2.4 Giao diện cơ bản



3. Công nghệ đã sử dụng

Em đã sử dụng ngôn ngữ lập trình Python để thực hiện project này

3.1 Mô hình học máy:

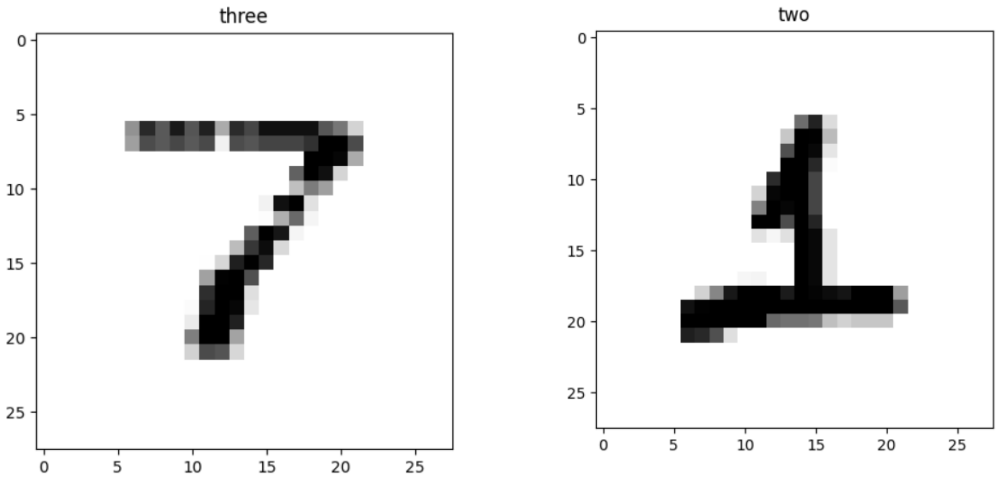
- Thư viện xử lý số học: numpy
- Thư viện để vẽ biểu đồ: matplotib
- Thư viện xử lý hình ảnh: OpenCV
- Thử viện xử lý xây dựng mô hình học sâu: tensorflow, keras

3.2 Giao diện:

- Sử dụng thư viện đồ họa trong thư viện pygame

4. Kết quả

Mô hình dự đoán khá tốt với những chữ số: 0,2,3,4,5,6,8,9
Mô hình dự đoán rất kém khi dự đoán số 7 với số 1.



Có thể thấy ở hình trên là tuy rằng các nét vẽ tương đối là khá rõ ràng, mô hình vẫn gán nhãn saic ho dù số 3 có rất ít nét tương đồng với số 7 và số 2 có rất ít nét tương đồng với số 1. Nguyên nhân của lỗi này hiện tại vẫn chưa được làm rõ.

5. Hướng phát triển của đề tài và kết luận

5.1 Hướng phát triển của đề tài

Cách xây dựng models trên hoàn toàn có thể áp dụng được cho việt phân biệt chữ cái viết tay chứ không chỉ dừng lại ở việc chỉ dự đoán các chữ số viết tay. Vấn đề cản trở duy nhất chỉ là tìm tập dữ liệu chữ viết tay đủ lớn và chinh xác để huấn luyện cho mô hình học sâu.

Project này hoàn toàn có thể phát triển thành bài toán nhận diện chữ viết tay. Ngoài ra, nhóm có thể phát triển project này thành nhận dạng chữ số hoặc chữ cái trên hình ảnh bất kì cho trước.

5.2 Kết luận

Qua project này, em đã hiểu thêm về deep learning (nhất là mạng CNN) và được làm quen với các công cụ hỗ trợ viết lên phần mềm học sâu. Đây sẽ là cơ sở để có thể làm những dự án liên quan đến mảng này trong tương lai.