

**BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH
KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**



HCMUTE

ĐỒ ÁN MÔN HỌC CUỐI KỲ



TRÍ TUỆ NHÂN TẠO

NUMBER SIGNAL INDICATION WEBSITE

Mã môn học: ARIN330585_22_1_03CLC

Sinh viên thực hiện:

Nguyễn Minh Tuấn 20110594

Bùi Văn Lâm 17110043

Tp. Hồ Chí Minh, tháng 12 năm 2022

MỤC LỤC

| | |
|---|----|
| <i>GIỚI THIỆU ĐỀ TÀI</i> | 3 |
| <i>I: TỔNG QUAN VỀ NHẬN DẠNG CHỮ SỐ VIẾT TAY</i> | 3 |
| 1. Giới thiệu chung | 3 |
| 2. Những khó khăn trong việc nhận dạng | 3 |
| <i>II: BỘ DỮ LIỆU TRAINING VÀ THƯ VIỆN SỬ DỤNG TRONG PYTHON</i> | 4 |
| 1. Bộ dữ liệu MNIST | 4 |
| 2. Thư viện Tensorflow | 5 |
| 2.1 Giới thiệu | 6 |
| 2.2 Cách hoạt động | 7 |
| <i>III: XÂY DỰNG ỨNG DỤNG</i> | 7 |
| <i>IV: KẾT LUẬN</i> | 11 |
| <i>V: TÀI LIỆU THAM KHẢO</i> | 11 |

GIỚI THIỆU ĐỀ TÀI

Nhận dạng là lĩnh vực được các nhà khoa học rất quan tâm để giải quyết các yêu cầu trong cuộc sống hiện nay, có nhiều lĩnh vực nhận dạng như nhận dạng tín hiệu, nhận dạng tiếng nói hay nhận dạng ảnh. Vấn đề nhận dạng chữ viết tay nói chung và nhận dạng chữ số viết tay nói riêng là một vấn đề thách thức đối với những nhà nghiên cứu. Chữ số viết tay xuất hiện ở hầu hết trong các công việc của các cơ quan, nhà máy, xí nghiệp, trường học. Trong các trường phổ thông hiện nay, có một bộ phận quản lý điểm thực hiện các khâu tiếp nhận và nhập vào máy tính bảng điểm viết tay của giáo viên bộ môn, công tác này luôn chiếm nhiều thời gian và đôi khi không đảm bảo tiến độ hoạt động của nhà trường.

Để nhận dạng chữ số viết tay, có nhiều phương pháp và kỹ thuật khác nhau như: logic mờ, giải thuật di truyền, mô hình xác suất thống kê, mô hình mạng nơ ron..

Với đề tài cuối kì này, chúng em chọn nhận dạng chữ số viết tay với thư viện Tensorflow.

I: TỔNG QUAN VỀ NHẬN DẠNG CHỮ SỐ VIẾT TAY

1. Giới thiệu chung

Hiện nay, vấn đề nhận dạng chữ số viết tay rất cần thiết, có nhiều ứng dụng rộng rãi trong đời sống xã hội như nhận dạng bảng điểm, nhận dạng bảng số xe, nhận dạng phiếu hàng hóa,... Vấn đề nhận dạng chữ viết tay nói chung và nhận dạng chữ số viết tay nói riêng là một thách thức lớn đối với các nhà nghiên cứu. Mỗi người có một cách viết chữ số khác nhau, chúng ta không thể xác định cách duy nhất để nhận dạng chữ số. Do vậy, xây dựng hệ thống nhận dạng chữ số viết tay một cách đáng tin cậy để có thể nhận dạng bất cứ ký tự số nào là điều không dễ dàng.

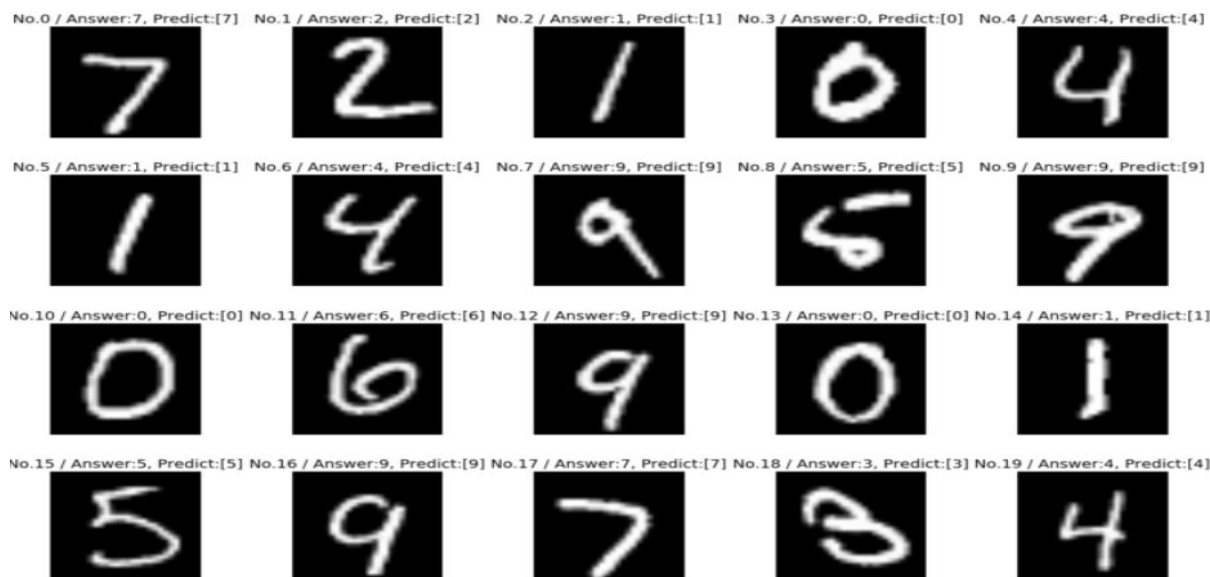
2. Những khó khăn trong việc nhận dạng

Chữ viết tay được viết bởi những người khác nhau, ở những trạng thái khác nhau nên đôi khi một số chữ viết bị nhòe hoặc mờ, bề mặt ký tự có thể bị mờ hoặc che khuất bởi một đối tượng khác, dẫn đến đọc ký tự sai.

Trong một số trường hợp, người viết có thể viết các chữ số dính liền nhau dẫn đến việc phân tích ký tự chưa được chính xác, dẫn đến đọc ký tự sai.

Một số chữ viết nằm giao nhau, cắt nhau hoặc chữ số này chứa trong chữ số kia gây khó khăn trong việc nhận dạng, dẫn đến đọc ký tự sai.

Trong phạm vi của đề tài , chúng em chỉ giới hạn nhận dạng mười ký tự số viết tay từ không đến chín.



II: BỘ DỮ LIỆU TRAINING VÀ THƯ VIỆN SỬ DỤNG TRONG PYTHON

1. Bộ dữ liệu MNIST

Bộ dữ liệu MNIST (tiếng Anh: **MNIST database**, viết tắt từ *Modified National Institute of Standards and Technology database*) là một cơ sở dữ liệu lớn chứa các chữ số viết tay thường được dùng trong việc huấn luyện các hệ thống xử lý hình ảnh khác nhau. Cơ sở dữ liệu này cũng được sử dụng rộng rãi để huấn luyện và kiểm thử trong lĩnh vực học máy. Cơ sở dữ liệu được tạo ra bằng cách "trộn lại" các mẫu từ bộ dữ liệu ban đầu của NIST.

Cơ sở dữ liệu MNIST của các chữ số viết tay, có một bộ đào tạo gồm 60.000 mẫu và một bộ thử nghiệm gồm 10.000 mẫu. Nó là một tập hợp con của một bộ lớn hơn có sẵn từ NIST. Các chữ số đã được bình thường hóa kích thước và tập trung vào một hình ảnh có kích thước cố định.

Đây là một cơ sở dữ liệu dành cho những người muốn thử các kỹ thuật học tập và phương pháp nhận dạng mẫu trên dữ liệu trong thế giới thực trong khi dành những nỗ lực tối thiểu cho việc tiền xử lý và định dạng.

Hai tập tin sau đây chứa dữ liệu để huấn luyện, gồm các ảnh và nhãn lớp gắn sẵn cho chúng:

train-images-idx3-ubyte.gz: Chứa dữ liệu ảnh của 60000 mẫu huấn luyện

train-labels-idx1-ubyte.gz: Chứa nhãn lớp của 60000 mẫu huấn luyện.

Kèm theo đó là hai tập tin chứa dữ liệu để kiểm tra, cũng gồm ảnh và nhãn lớp thật để phục vụ cho đánh giá độ chính xác khi nhận dạng:

t10k-images-idx3-ubyte.gz: Chứa dữ liệu ảnh của 10000 mẫu kiểm tra

t10k-labels-idx1-ubyte.gz: Chứa nhãn lớp của 10000 mẫu kiểm tra

Dữ liệu gồm 60.000 mẫu chữ số viết tay, mỗi mẫu là một ảnh grayscale kích thước 28x28 của các chữ số đơn viết tay trong khoảng từ 0 đến 9, cụ thể:

| Chữ số | Số mẫu |
|--------|--------|
| 0 | 6.903 |
| 1 | 7.877 |
| 2 | 6.990 |
| 3 | 7.141 |
| 4 | 6.824 |
| 5 | 6.313 |
| 6 | 6.876 |
| 7 | 7.293 |
| 8 | 6.825 |
| 9 | 6.958 |

2. Thư viện Tensorflow

2.1 Giới thiệu

Tensorflow chính là thư viện mã nguồn mở cho machine learning nổi tiếng nhất thế giới, được phát triển bởi các nhà nghiên cứu từ Google. Việc hỗ trợ mạnh mẽ các phép toán học để tính toán trong machine learning và deep learning đã giúp việc tiếp cận các bài toán trở nên đơn giản, nhanh chóng và tiện lợi hơn nhiều.

các hàm được dựng sẵn trong thư viện cho từng bài toán cho phép TensorFlow xây dựng được nhiều neural network. Nó còn cho phép bạn tính toán song song trên nhiều máy tính khác nhau, thậm chí trên nhiều CPU, GPU trong cùng 1 máy hay tạo ra các dataflow graph – đồ thị luồng dữ liệu để dựng nên các model.

TensorFlow 2.0, được ra mắt vào tháng 10 năm 2019, cải tiến framework theo nhiều cách dựa trên phản hồi của người dùng, để dễ dàng và hiệu quả hơn khi làm việc cùng nó (ví dụ: bằng cách sử dụng các Keras API liên quan đơn giản cho việc train model). Train phân tán dễ chạy hơn nhờ vào API mới và sự hỗ trợ cho TensorFlow Lite cho phép triển khai các mô hình trên khá nhiều nền tảng khác nhau.

Được viết bằng C++ và thao tác interface bằng Python nên phần performance của TensorFlow cực kỳ tốt. Đối tượng sử dụng nó cũng đa dạng không kém: từ các nhà nghiên cứu, nhà khoa học dữ liệu và dĩ nhiên không thể thiếu các lập trình viên.

** Kiến trúc TensorFlow*

- Tiền xử lý dữ liệu
- Dựng model
- Train và ước tính model

** Các thành phần của Tensorflow*

- **Tensor**

1 tensor là 1 vector hay ma trận của n-chiều không gian đại diện cho tất cả loại dữ liệu. Tất cả giá trị trong 1 tensor chứa đựng loại dữ liệu giống hệt nhau với 1 shape đã biết (hoặc đã biết 1 phần). Shape của dữ liệu chính là chiều của ma trận hay mảng.

- **Graph**

Biểu đồ tập hợp và mô tả tất cả các chuỗi tính toán được thực hiện trong quá trình training

- Nó được làm ra để chạy trên nhiều CPU hay GPU, ngay cả các hệ điều hành trên thiết bị điện thoại.
- Tính di động của biểu đồ cho phép bảo toàn các tính toán để bạn sử dụng ngay hay sau đó. Biểu đồ có thể được lưu lại để thực thi trong tương lai.
- Tất cả tính toán trong biểu đồ được thực hiện bằng cách kết nối các tensor lại với nhau. 1 tensor có 1 node và 1 edge. Node mang operation toán học và sản xuất các output ở đầu cuối. Các edge giải thích mối quan hệ input/output giữa các node.

2.2 Cách hoạt động

TensorFlow cho phép các lập trình viên tạo ra dataflow graph, cấu trúc mô tả làm thế nào dữ liệu có thể di chuyển qua 1 biểu đồ, hay 1 seri các node đang xử lý. Mỗi node trong đồ thị đại diện 1 operation toán học, và mỗi kết nối hay edge giữa các node là 1 mảng dữ liệu đa chiều, hay còn được gọi là ‘tensor’.

TensorFlow cung cấp tất cả những điều này cho lập trình viên theo phương thức của ngôn ngữ Python. Vì Python khá dễ học và làm việc, ngoài ra còn cung cấp nhiều cách tiện lợi để ta hiểu được làm thế nào các high-level abstractions có thể kết hợp cùng nhau. Node và tensor trong TensorFlow là các đối tượng Python, và các ứng dụng TensorFlow bản thân chúng cũng là các ứng dụng Python.

Các operation toán học thực sự thì không được thi hành bằng Python. Các thư viện biến đổi có sẵn thông qua TensorFlow được viết bằng các binary C++ hiệu suất cao. Python chỉ điều hướng lưu lượng giữa các phần và cung cấp các high-level abstraction lập trình để nối chúng lại với nhau.

III: XÂY DỰNG ỨNG DỤNG

1. Bài toán:

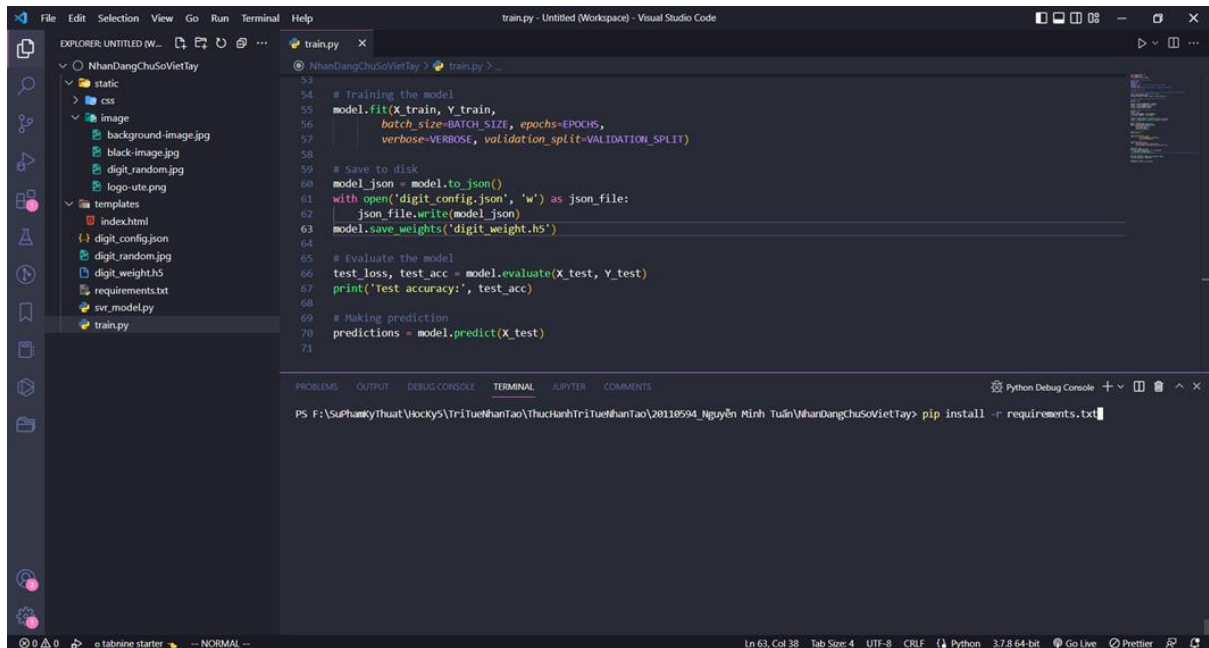
Tạo chữ số viết tay ngẫu nhiên cho vào hệ thống nhận dạng sau đó hệ thống sẽ sử dụng thuật toán, dự đoán chữ số viết tay ngẫu nhiên đó, các chữ số cần nhận dạng trong khoảng 0-9, quá trình nhận dạng có thể có sai sót.

2. Hướng dẫn sử dụng ứng dụng :

Chức năng chính của web:

+ Dự đoán chữ số viết tay ngẫu nhiên

+Ta cần cài các thư viện trong **requirements.txt** bằng lệnh **pip install -r requirements.txt**

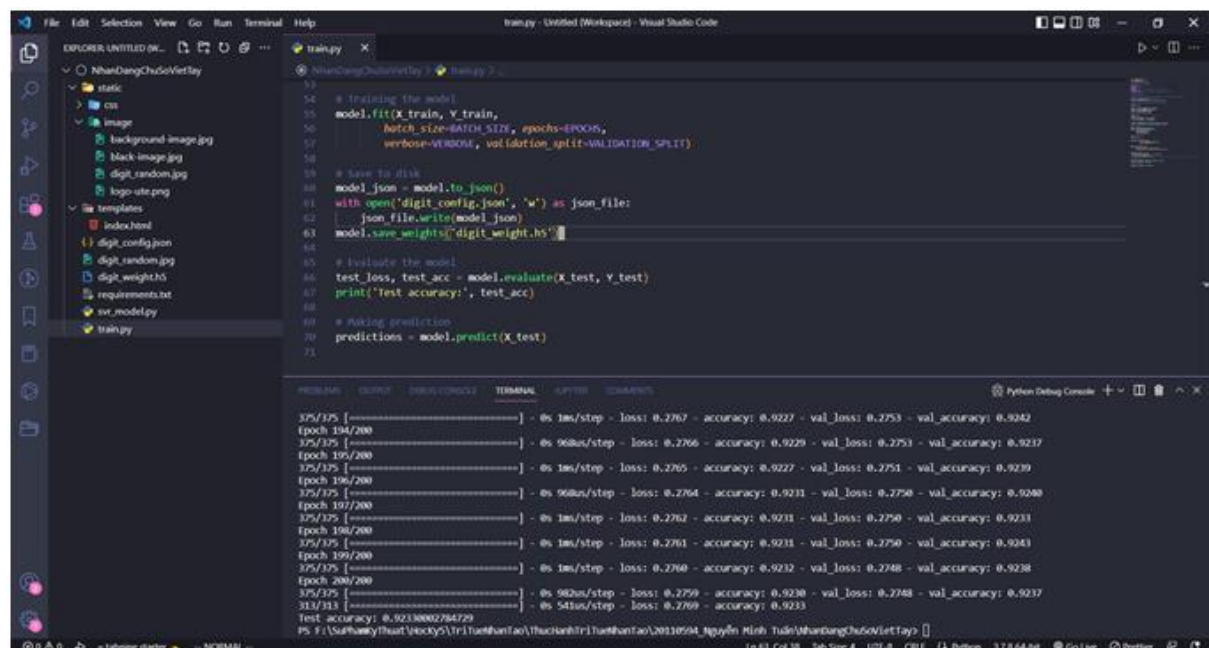


The screenshot shows the Visual Studio Code interface with a file explorer on the left displaying a project named 'NhanDangChuSoVietTay'. The file explorer lists various assets like images and JSON files, along with 'requirements.txt' and 'train.py'. The 'train.py' file is open in the editor, showing code for training a model, saving it to JSON, evaluating it, and making predictions. The terminal at the bottom shows the command 'pip install -r requirements.txt' being executed in a PowerShell window.

```
53
54 # Training the model
55 model.fit(X_train, Y_train,
56         batch_size=BATCH_SIZE, epochs=EPOCHS,
57         verbose=VERBOSE, validation_split=VALIDATION_SPLIT)
58
59 # Save to disk
60 model_json = model.to_json()
61 with open('digit_config.json', 'w') as json_file:
62     json_file.write(model_json)
63 model.save_weights('digit_weight.h5')
64
65 # Evaluate the model
66 test_loss, test_acc = model.evaluate(X_test, Y_test)
67 print('Test accuracy:', test_acc)
68
69 # Making prediction
70 predictions = model.predict(X_test)
71
```

```
PS F:\SuPhamkyThuat\Vecky5\TriTuanhanTao\ThuckanhTriTuanhanTao\20110594_Nguyen Minh Tuan\NhanDangChuSoVietTay> pip install -r requirements.txt
```

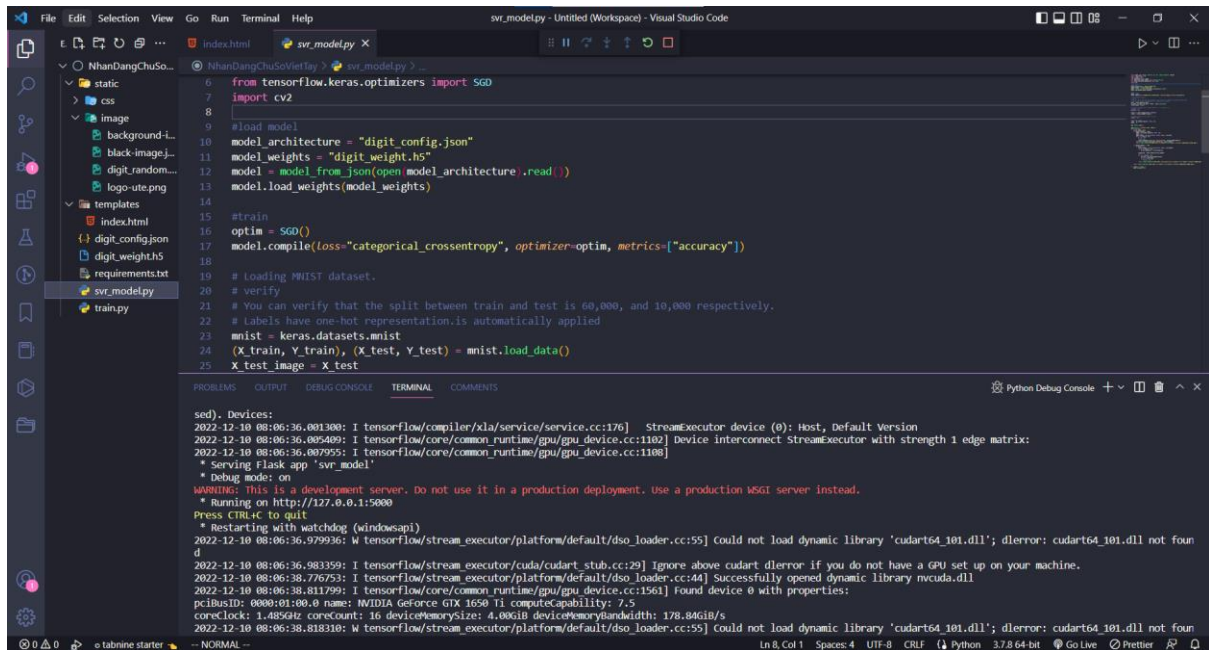
+ Training bộ dataset mnist bằng cách chạy file **train.py**



The screenshot shows the Visual Studio Code interface with the 'train.py' file open. The terminal at the bottom displays the output of the training process, showing progress bars and metrics for each epoch. The training is complete, and the final test accuracy is 0.9233.

```
375/375 [-----] - 6s 1ms/step - loss: 0.2767 - accuracy: 0.9227 - val_loss: 0.2753 - val_accuracy: 0.9242
Epoch 194/200
375/375 [-----] - 6s 968us/step - loss: 0.2766 - accuracy: 0.9229 - val_loss: 0.2753 - val_accuracy: 0.9237
Epoch 195/200
375/375 [-----] - 6s 1ms/step - loss: 0.2765 - accuracy: 0.9227 - val_loss: 0.2751 - val_accuracy: 0.9239
Epoch 196/200
375/375 [-----] - 6s 968us/step - loss: 0.2764 - accuracy: 0.9231 - val_loss: 0.2750 - val_accuracy: 0.9208
Epoch 197/200
375/375 [-----] - 6s 1ms/step - loss: 0.2762 - accuracy: 0.9231 - val_loss: 0.2750 - val_accuracy: 0.9231
Epoch 198/200
375/375 [-----] - 6s 1ms/step - loss: 0.2761 - accuracy: 0.9231 - val_loss: 0.2750 - val_accuracy: 0.9243
Epoch 199/200
375/375 [-----] - 6s 1ms/step - loss: 0.2760 - accuracy: 0.9232 - val_loss: 0.2748 - val_accuracy: 0.9238
Epoch 200/200
375/375 [-----] - 6s 982us/step - loss: 0.2759 - accuracy: 0.9230 - val_loss: 0.2748 - val_accuracy: 0.9237
313/313 [-----] - 5s 541us/step - loss: 0.2769 - accuracy: 0.9233
Test accuracy: 0.92330002784729
PS F:\SuPhamkyThuat\Vecky5\TriTuanhanTao\ThuckanhTriTuanhanTao\20110594_Nguyen Minh Tuan\NhanDangChuSoVietTay>
```


+ Chạy file `srv_model.py` khởi chạy trang web



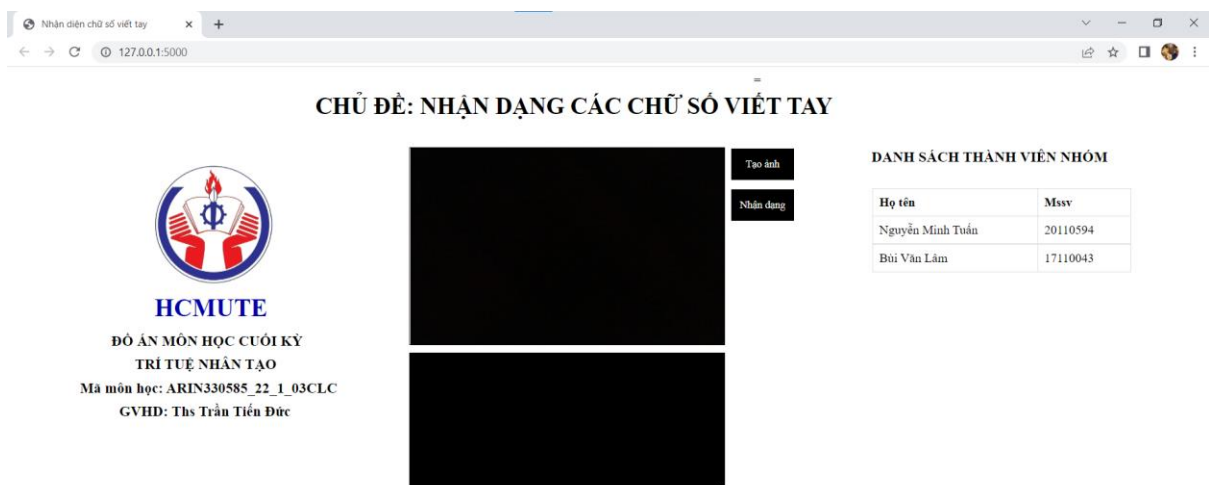
```
6 from tensorflow.keras.optimizers import SGD
7 import cv2
8
9 #load model
10 model_architecture = "digit_config.json"
11 model_weights = "digit_weight.h5"
12 model = model_from_json(open(model_architecture).read())
13 model.load_weights(model_weights)
14
15 #train
16 optim = SGD()
17 model.compile(loss="categorical_crossentropy", optimizer=optim, metrics=["accuracy"])
18
19 # Loading MNIST dataset.
20 # verify
21 # You can verify that the split between train and test is 60,000, and 10,000 respectively.
22 # Labels have one-hot representation, is automatically applied
23 mnist = keras.datasets.mnist
24 (X_train, Y_train), (X_test, Y_test) = mnist.load_data()
25 X_test_image = X_test
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL COMMENTS

Python Debug Console

```
sed). Devices:
2022-12-10 08:06:36.001300: I tensorflow/compiler/xla/service/service.cc:176] StreamExecutor device (0): Host, Default Version
2022-12-10 08:06:36.005409: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1102] Device interconnect StreamExecutor with strength 1 edge matrix:
2022-12-10 08:06:36.007955: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1106]
* Serving Flask app 'srv_model'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with watchdog (windowsapi)
2022-12-10 08:06:36.979936: W tensorflow/stream_executor/platform/default/dso_loader.cc:55] Could not load dynamic library 'cudart64_101.dll'; dlerror: cudart64_101.dll not found
2022-12-10 08:06:36.983359: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
2022-12-10 08:06:38.776753: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library mcuda.dll
2022-12-10 08:06:38.811799: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1561] Found device 0 with properties:
pciBusID: 0000:01:00:0 name: NVIDIA GeForce GTX 1650 Ti computeCapability: 7.5
coreClock: 1.4850GHz coreCount: 16 deviceMemorySize: 4.00GiB deviceMemoryBandwidth: 178.84GiB/s
2022-12-10 08:06:38.818310: W tensorflow/stream_executor/platform/default/dso_loader.cc:55] Could not load dynamic library 'cudart64_101.dll'; dlerror: cudart64_101.dll not found
```

+ **Copy** hoặc **ctrl + click url**: <http://127.0.0.1:5000> và chạy trên trình duyệt, giao diện chính của web



+ Ta nhấn vào nút “**Tạo ảnh**” để tạo ra hình ảnh của những số viết tay ngẫu nhiên cần nhận diện, kết quả sẽ hiện ra

CHỦ ĐỀ: NHẬN DẠNG CÁC CHỮ SỐ VIẾT TAY

HCMUTE
ĐỒ ÁN MÔN HỌC CUỐI KỲ
TRÍ TUỆ NHÂN TẠO
Mã môn học: ARIN330585_22_1_03CLC
GVHD: Ths Trần Tiến Đức

Tạo ảnh
Nhận dạng

DANH SÁCH THÀNH VIÊN NHÓM

| Họ tên | Mssv |
|------------------|----------|
| Nguyễn Minh Tuấn | 20110594 |
| Bùi Văn Lâm | 17110043 |

+ Ta nhấn vào nút “**Nhận dạng**” nhận diện những chữ số viết tay ngẫu nhiên, kết quả sẽ hiện ra

CHỦ ĐỀ: NHẬN DẠNG CÁC CHỮ SỐ VIẾT TAY

HCMUTE
ĐỒ ÁN MÔN HỌC CUỐI KỲ
TRÍ TUỆ NHÂN TẠO
Mã môn học: ARIN330585_22_1_03CLC
GVHD: Ths Trần Tiến Đức

Tạo ảnh
Nhận dạng

DANH SÁCH THÀNH VIÊN NHÓM

| Họ tên | Mssv |
|------------------|----------|
| Nguyễn Minh Tuấn | 20110594 |
| Bùi Văn Lâm | 17110043 |

Nhận diện thành công!

Các chữ số sẽ được nhận diện sẽ hiển thị bên dưới hình ảnh các chữ số

IV: KẾT LUẬN

Qua tìm hiểu bài toán nhận diện chữ viết số chúng em đã thấy được sự quan trọng sự phát triển và tầm quan trọng của bài toán nhận diện chữ viết số trong đời sống.

Đề tài đã giới thiệu tuần tự về nhận dạng chữ viết số tay. Xây dựng được website thực hiện nhận dạng các chữ số viết

Đề tài và bài báo cáo được chúng em thực hiện trong khoảng thời gian ngắn, với những kiến thức còn hạn chế cùng nhiều hạn chế khác về mặt kỹ thuật và kinh nghiệm trong việc thực hiện một dự án phần mềm. Do đó, trong quá trình làm nên đề tài có những thiếu sót là điều không thể tránh khỏi nên chúng em rất mong nhận được những ý kiến đóng góp quý báu của thầy để kiến thức của chúng em được hoàn thiện hơn và chúng em có thể làm tốt hơn nữa trong những lần sau. Chúng em xin chân thành cảm ơn.

V: TÀI LIỆU THAM KHẢO

[1]<https://wiki.tino.org/tensorflow-la-gi/>

[2]<http://yann.lecun.com/exdb/mnist/>

[3]https://vi.wikipedia.org/wiki/C%C6%A1_s%E1%BB%9F_d%E1%BB%AF_li%E1%BB%87u_MNIST

[4]<https://123docz.net/document/2581176-tim-hieu-va-nang-cao-hieu-qua-nhan-dang-chu-viet-tay-roi-rac-dua-tren-cac-ky-thuat-lay-dac-trung-va-phat-trien-ung-dung.htm>