BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC CẦN THƠ**

**KHOA CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG**

**🙞 🕮 🙜**



**NIÊN LUẬN CƠ SỞ NGÀNH**

**NGÀNH KHOA HỌC MÁY TÍNH**

**Đề tài**

**XÂY DỰNG WEBSITE NHẬN DẠNG CHỮ VIẾT TAY   
TIẾNG VIỆT**

(Mã môn: CT201)

Sinh viên thực hiện: **Nguyễn Tấn Thành**

Mã số: **B1509894**

Khóa: **41**

**Học kì: I**

**Năm học (2019-2020)**

BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC CẦN THƠ**

**KHOA CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG**

**🙞 🕮 🙜**



**NIÊN LUẬN CƠ SỞ NGÀNH**

**NGÀNH KHOA HỌC MÁY TÍNH**

**Đề tài**

**XÂY DỰNG WEBSITE NHẬN DẠNG CHỮ VIẾT TAY   
TIẾNG VIỆT**

(Mã môn: CT201)

Giáo viên hướng dẫn: Sinh viên thực hiện:

**TS.Lưu Tiến Đạo Nguyễn Tấn Thành**

Mã số: **B1509894**

Khóa : **41**

**Học kì: I**

**Năm học (2019-2020)**

**NHẬN XÉT CỦA GIẢNG VIÊN**

**-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**LỜI CẢM ƠN**

Để có được bài niên luận này, em xin được bày tỏ lòng biết ơn chân thành và sâu sắc đến THẦY LƯU TIẾN ĐẠO – người đã trực tiếp tận tình hướng dẫn, giúp đỡ em.Trong suốt quá trình thực hiện niên luận, nhờ những sự chỉ bảo và hướng dẫn quý giá đó mà bài niên luận này được hoàn thành một cách tốt nhất.

Em cũng xin gửi lời cám ơn chân thành đến các Thầy Cô Giảng viên Đại học Cần Thơ, đặc biệt là các Thầy Cô ở Khoa CNTT & TT, những người đã truyền đạt những kiến thức quý báu trong thời gian qua.

Em cũng xin chân thành cảm ơn bạn bè cùng với gia đình đã luôn động viên, khích lệ và tạo điều kiện giúp đỡ trong suốt quá trình thực hiện để em có thể hoàn thành bài niên luận một cách tốt nhất.

Tuy có nhiều cố gắng trong quá trình thực hiện niên luận, nhưng không thể tránh khỏi những sai sót. Em rất mong nhận được sự đóng góp ý kiến quý báu của quý Thầy Cô và các bạn để bài niên luận hoàn thiện hơn.

Cần Thơ, ngày tháng 12 năm 2019

Người viết

Nguyễn Tấn Thành

**MỤC LỤC**

*Trang*

[CHƯƠNG 1: PHẦN GIỚI THIỆU 1](#_Toc26725655)

[1.1 Đặt vấn đề 1](#_Toc26725656)

[1.2 Lịch sử giải quyết vấn đề 1](#_Toc26725657)

[1.3 Mục tiêu đề tài 2](#_Toc26725658)

[1.4 Thu thập dữ liệu 2](#_Toc26725659)

[1.5 Đối tượng và phạm vi nghiên cứu 2](#_Toc26725660)

[1.5.1 Đối tượng 2](#_Toc26725661)

[1.5.2 Phương pháp nghiên cứu 2](#_Toc26725662)

[1.6 Kết quả đạt được 3](#_Toc26725663)

[CHƯƠNG 2: PHẦN NỘI DUNG 4](#_Toc26725664)

[2.1 Mô tả chi tiết bài toán 4](#_Toc26725665)

[2.2 Vấn đề và giải pháp liên quan đến bài toán 5](#_Toc26725666)

[2.2.1 Kiến trúc CNN – Convolutional Neural Network 5](#_Toc26725667)

[2.2.2 Tầng tích chập (Convolution) 6](#_Toc26725668)

[2.2.3 Một số khái niệm trong tích chập 6](#_Toc26725669)

[2.2.4 Tầng Pooling 7](#_Toc26725670)

[2.2.5 Tầng ReLU 8](#_Toc26725671)

[2.2.6 Tầng Fully Connected (FC) 8](#_Toc26725672)

[2.3 Tập dữ liệu 9](#_Toc26725673)

[2.4 Thiết kế và cài đặt 11](#_Toc26725674)

[2.4.1 Tiền xử lý dữ liệu 11](#_Toc26725675)

[2.4.2 Lựa chọn mô hình 12](#_Toc26725676)

[2.4.3 Định nghĩa mô hình 14](#_Toc26725677)

[2.4.4 Huấn luyện mô hình 15](#_Toc26725678)

[2.4.5 Kết quả 16](#_Toc26725679)

[2.4.6 Đánh giá mô hình 17](#_Toc26725680)

[CHƯƠNG 3: TRIỂN KHAI MÔ HÌNH LÊN TRÌNH DUYỆT WEB 18](#_Toc26725681)

[3.1 Chuyển đổi mô hình sang dạng Tensorflow JS 18](#_Toc26725682)

[3.2 Đưa mô hình lên web 19](#_Toc26725683)

[3.2.1 File Mapping Name 19](#_Toc26725684)

[3.2.2 Hàm dự đoán ký tự viết tay 20](#_Toc26725685)

[3.2.3 Giao diện chính 20](#_Toc26725686)

[3.3 Demo chương trình 20](#_Toc26725687)

[CHƯƠNG 4: KẾT LUẬN 21](#_Toc26725688)

[4.1 Kết quả đạt được 21](#_Toc26725689)

[4.2 Hướng phát triển 21](#_Toc26725690)

**DANH MỤC ẢNH**

*Trang*

[Hình 2.1: Quá trình dự đoán 4](file:///T:\NLCS\bao_cao\Thanh_B1509894_7_12_2019%20.docx#_Toc26697664)

[Hình 2.2: Quá trình đào tạo mô hình 4](file:///T:\NLCS\bao_cao\Thanh_B1509894_7_12_2019%20.docx#_Toc26697665)

[Hình 2.3: Một mô hình CNN 5](file:///T:\NLCS\bao_cao\Thanh_B1509894_7_12_2019%20.docx#_Toc26697666)

[Hình 2.4: Quá trình tích chập một pixel với kernel 3x3 6](file:///T:\NLCS\bao_cao\Thanh_B1509894_7_12_2019%20.docx#_Toc26697667)

[Hình 2.5: Max-Pooling với bộ lọc 2x2, stride = 2. 7](file:///T:\NLCS\bao_cao\Thanh_B1509894_7_12_2019%20.docx#_Toc26697668)

[Hình 2.6: Tầng ReLU trong CNN 8](file:///T:\NLCS\bao_cao\Thanh_B1509894_7_12_2019%20.docx#_Toc26697669)

[Hình 2. 7: Kí tự hoa thường lẫn lộn 12](file:///T:\NLCS\bao_cao\Thanh_B1509894_7_12_2019%20.docx#_Toc26697670)

[Hình 2.8: Các kí tự khác nhau lẫn lộn 12](file:///T:\NLCS\bao_cao\Thanh_B1509894_7_12_2019%20.docx#_Toc26697671)

[Hình 2.9: Kiến trúc của mô hình MobileNet 13](file:///T:\NLCS\bao_cao\Thanh_B1509894_7_12_2019%20.docx#_Toc26697672)

[Hình 2.10: Độ chính xác trong 40 lần huấn luyện 16](file:///T:\NLCS\bao_cao\Thanh_B1509894_7_12_2019%20.docx#_Toc26697673)

[Hình 2.11: Tỉ lệ lỗi trong 40 lần huấn luyện 17](file:///T:\NLCS\bao_cao\Thanh_B1509894_7_12_2019%20.docx#_Toc26697674)

[Hinh 3. 1: Giao diện Demo trên Website 20](file:///T:\NLCS\bao_cao\Thanh_B1509894_8_12_2019.docx#_Toc26726975)

**DANH MỤC BẢNG**

*Trang*

[Bảng 2.1: Tổng quan về tập dữ liệu 10](#_Toc26346771)

**BẢNG DANH SÁCH CÁC TỪ VIẾT TẮT**

|  |  |
| --- | --- |
| **Từ viết tắt** | **Tên đầy đủ** |
| CNN | Convolutional Neural Network |
| DNN | Deep Neural Network |

1. PHẦN GIỚI THIỆU
   1. Đặt vấn đề

Ngày nay, trí tuệ nhân tạo đang là tâm điểm nghiên cứu của các nhà khoa học máy tính, bởi lợi ích mà trí tuệ nhân tạo mang lại.

Chúng được ứng dụng rộng rãi trong nhiều lĩnh vực của cuộc sống thường ngày như thương mại, các ứng dụng trợ lý ảo dành cho điện thoại thông minh, cho đến các vấn đề mang tính học thuật cao như hệ chuyên gia, người máy học (robotics).

Ngoài các lĩnh vực đã liệt kê, không thể không nhắc đến lĩnh vực thị giác máy tính mà ở đó, với trí thông minh nhân tạo, máy tính có thể phân lớp và dự đoán vật thể thông qua hình ảnh từ camera ghi lại, cùng với đó nhận dạng chữ là lĩnh vực được quan tâm nghiên cứu và ứng dụng từ nhiều năm nay.

Nhận dạng chữ gồm nhận dạng chữ in và nhận dạng chữ viết tay. Nhận dạng chữ in thường được ứng dụng phục vụ công việc tự động hóa đọc tài liệu, tăng tốc độ và hiệu quả nhập thông tin vào máy tính trực tiếp. Nhận dạng chữ viết tay thường phục vụ cho các ứng dụng đọc và xử lý chứng từ, hóa đơn, phiếu ghi, bản thảo viết tay.

Trong lĩnh vực này, thì các mô hình mạng nơ-ron của Học sâu (Deep learning) đang được áp dụng vào các sản phẩm tại các công ty công nghệ lớn như: Alex Assistant (Amazon), Alpha Go (Google),... Cho đến những ứng dụng đơn giản như nhận dạng và phân loại mà chúng ta vẫn tận dụng hàng ngày. Để bắt kịp với tốc độ phát triển đó, trong bài niên luận này ứng dụng mạng nơ-ron tích chập đa lớp (DNN) sẽ được sử dụng để giải quyết một bài toán cơ sở về nhận dạng chữ viết tay.

* 1. Lịch sử giải quyết vấn đề

Trong thực tế đã có nhiều công trình nghiên cứu thậm chí là các sản phẩm về nhận dạng chữ viết tay ra đời như:

* Nhận dạng chữ viết tay trong tiếng nhật.
* Ứng dụng nhận dạng chữ viết tay để chấm phiếu thi trắc nghiệm.
* Nhận dạng điểm viết tay trên bảng điểm[1].
  1. Mục tiêu đề tài
* Tìm hiểu về học sâu và mô hình DNN (Deep Neural Network), mô hình được sử dụng trong niên luận này là mô hình MobileNet [2] từ thư viện Keras.
* Phân tích tập dữ liệu, sử dụng mô hình MobileNet để huấn luyện và dự đoán.
* Nhận dạng chữ bằng cách đọc trực tiếp một file ảnh và vẽ trực tiếp trên trình duyệt web.
* Xây dựng ứng dụng Demo trên website.
  1. Thu thập dữ liệu

Dữ liệu được thu thập từ 2 nguồn là:

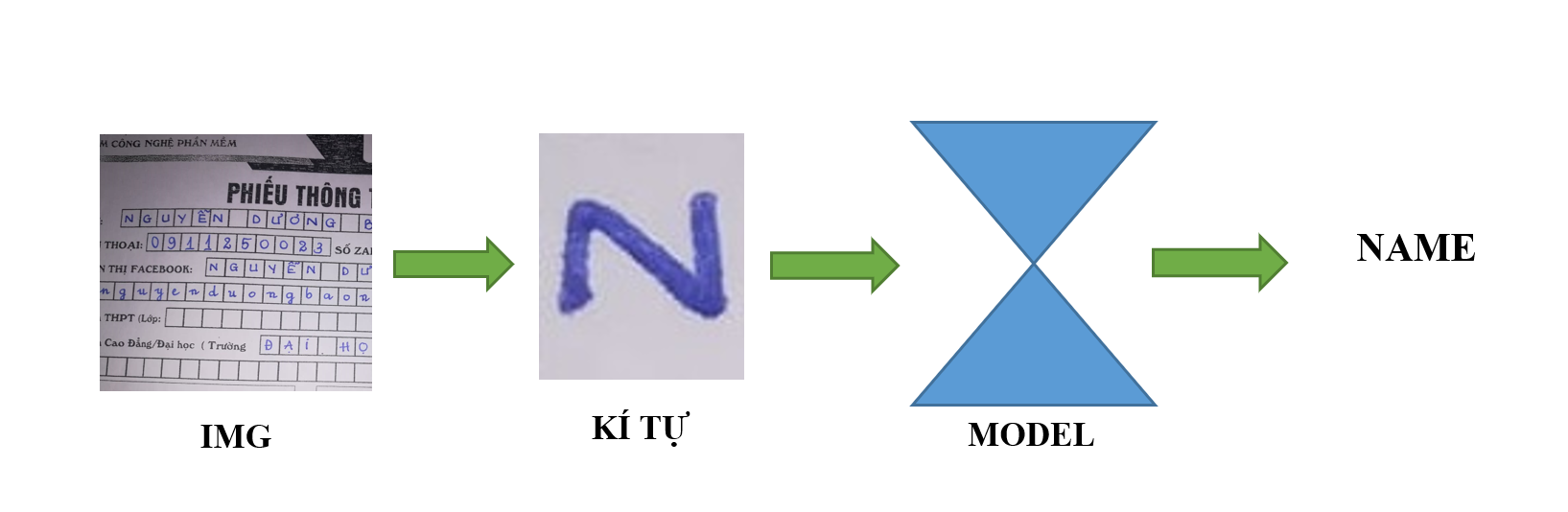
* A-Z Handwritten Alphabets với dung lượng là 108MB.
* NIST Special Database 19 với dung lượng là 3.64GB.
  1. Đối tượng và phạm vi nghiên cứu

Với mục tiêu đã đề ra thì đối tượng nghiên cứu của đề tài là công nghệ xử lý ảnh, thị giác máy tính được ứng dụng thông qua thư viện mã nguồn mở OpenCV, sử dụng mô hình MobileNet thông qua gói thư viện Keras trên nền tảng ngôn ngữ Python, triển khai mô hình lên website sử dụng thư viện Tensorflowjs. Sử dụng mô hình MobileNet cho tập dữ liệu.

* + 1. Đối tượng
* Mô hình được sử dụng thông qua gói thư viện Keras.
* Sử dụng gói thư viện OpenCV để xử lý hình ảnh đầu vào và tập dữ liệu sử dụng để huấn luyện.
* Sử dụng thư viện Tensoflowjs để chuyển đổi file mô hình sang các file TFJS.
  + 1. Phương pháp nghiên cứu
* Tìm kiếm, tổng hợp và phân tích tài liệu tham khảo.
* Nghiên cứu các công nghệ, thư viện mã nguồn mở có thể giúp ích cho việc giải quyết vấn đề.
* Tìm kiếm và phát triển giải pháp.
* Tìm hiểu về mô hình MobileNet.
* Thử nghiệm, đánh giá mô hình.
* Triển khai mô hình lên trên trình duyệt Web.
  1. Kết quả đạt được
* Kết quả là ứng dụng trên máy tính và ứng dụng Demo trên trình duyệt web.
* Xây dựng được Website nhận dạng chữ viết tay.

1. PHẦN NỘI DUNG
   1. Mô tả chi tiết bài toán

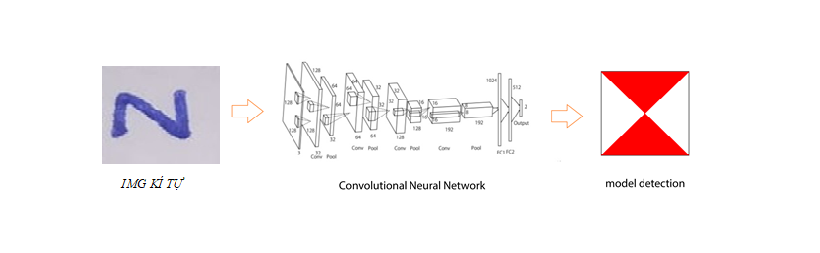
Ứng dụng nhận dạng chữ viết tay được sử dụng bằng cách đọc trực tiếp một file ảnh, mỗi file ảnh chứa duy nhất một kí tự in hoa hoặc số của một người nào đó viết. Bước tiếp theo sẽ tiến hành chuẩn hóa và sử dụng mô hình để dự đoán kí tự đó là kí tự nào trong bảng chữ cái in hoa từ A đến Z và số từ 0 đến 9.



Hình 2.1: Quá trình dự đoán

Trong mô hình trên, để dự đoán được ký tự trong ảnh, cần phải có một mô hình nhận diện (Model Detection). Mô hình nhận diện sẽ được thực hiện qua 2 bước sau:

* Thu thập bộ dữ liệu nhận dạng chữ viết tay.
* Sau khi có được tập dữ liệu ảnh của các kí tự, bước tiếp theo sẽ tiến hành chuẩn hóa ảnh từ tập dữ liệu.
* Tiến hành thiết kế xây dựng mô hình dự đoán bằng kiến trúc CNN và thực hiện quá trình học để cho ra mô hình dự đoán.



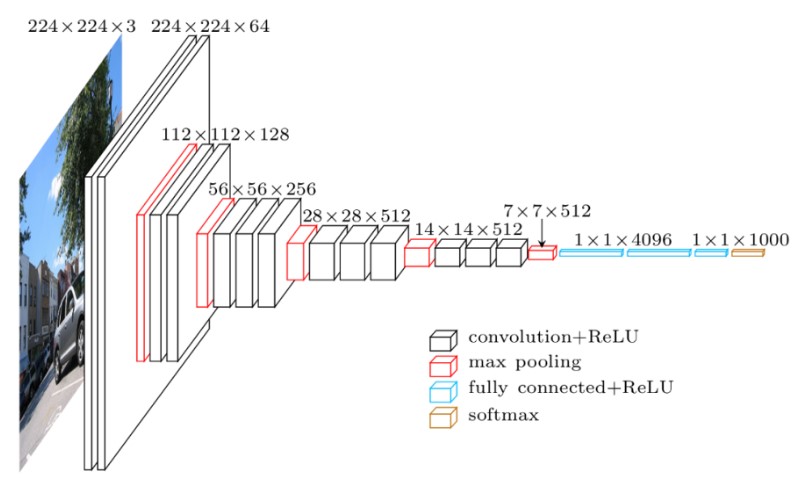
Hình 2.2: Quá trình đào tạo mô hình

* 1. Vấn đề và giải pháp liên quan đến bài toán
     1. Kiến trúc CNN – Convolutional Neural Network

Kiến trúc CNN[3] được xây dựng từ các lớp Convolution chồng lên nhau và sử dụng các hàm Nonlinear Activation như ReLU để kích hoạt các trọng số trong các mô hình. Mỗi một lớp sau khi thông qua các hàm kích hoạt sẽ tạo ra các thông tin trừu tượng hơn cho các lớp tiếp theo. Các lớp này liên kết được với nhau thông qua cơ chế Convolution. Lớp tiếp theo là kết quả Convolution từ lớp trước đó, nhờ vậy mà ta có được các kết nối cục bộ.

Như vậy, có thể nói mỗi nơ-ron ở lớp kế tiếp sinh ra từ kết quả của filter áp đặt lên một vùng ảnh cục bộ của nơ-ron trước đó. Mỗi một lớp được sử dụng các Filter khác nhau, thông thường có hàng trăm hàng nghìn filter và kết hợp kết quả của chúng lại. Ngoài ra có một số layer khác như Pooling/Subsampling layer dùng để chắt lọc lại các thông tin hữu ích hơn (loại bỏ các thông tin nhiễu).

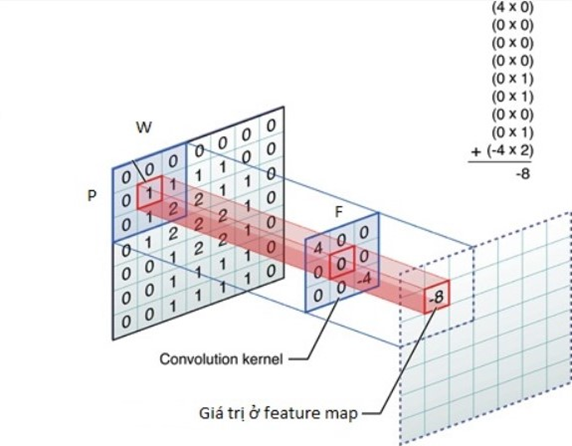
Trong quá trình huấn luyện mạng (Traning), CNN tự động học các giá trị qua các lớp filter dựa vào cách thức mà chúng ta thực hiện.



Hình 2.3: Một mô hình CNN

* + 1. Tầng tích chập (Convolution)

Tầng tích chập là tầng quan trọng nhất trong cấu trúc của CNN. Mục tiêu là thông qua việc lấy tích chập sẽ giúp trích xuất được những đặc trưng (thông tin quan trọng) từ dữ liệu hình ảnh.



Hình 2.4: Quá trình tích chập một pixel với kernel 3x3

Ảnh đầu là một ma trận số sẽ được bộ lọc (Kernel) chạy quét qua từng pixel của toàn bộ bức ảnh. Bộ lọc có kích thước là 3x3 (hoặc 5x5) và áp dụng phép tích vô hướng để tính toán, cho ra một giá trị duy nhất của từng pixel trong mảng đầu ra. Đầu ra của phép tích chập là một tập các giá trị ảnh được gọi là mạng- bản đồ đặc trưng (Features Map).

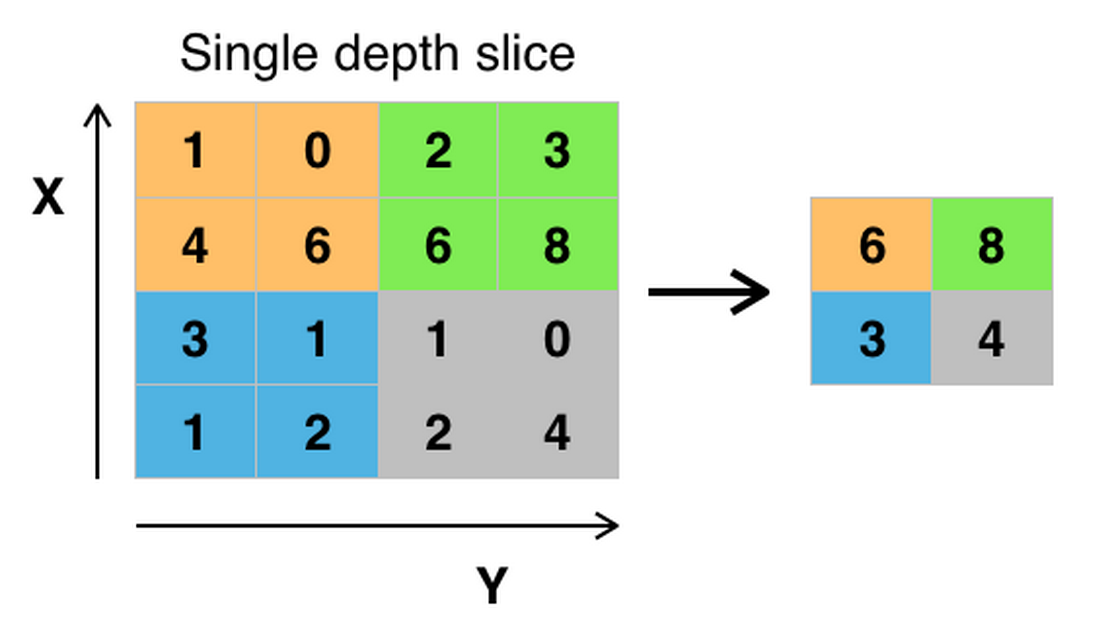
Công thức tích chập giữa hàm ảnh f(x, y) và bộ lọc k(x, y) có kích thước mxm.

* + 1. Một số khái niệm trong tích chập
* **Filter, Kernel** hay **Feature Detector** đều là cách gọi của ma trận lọc. Thông thường, ở các lớp đầu tiên của Convolution Kernel sẽ có kích thước là [5x5x3].
* **Convolved Feature, Activation Map hay Feature Map** là đầu ra của ảnh khi cho bộ lọc chạy hết bức ảnh với phép tích vô hướng.
* **Receptive** Field là vùng ảnh được chọn để tính tích chập, hay bằng đúng kích thước của bộ lọc.
* **Depth** là số lượng bộ lọc. Lưu ý: Depth không phải số lượng kênh màu RBG.
* **Stride** được hiểu là khoảng cách dịch chuyển của bộ lọc sau mỗi lần tính. Ví dụ khi Stride=2. Tức sau khi tính xong tại 1 vùng ảnh, nó sẽ dịch sang phải 2 pixel. Tương tự với việc dịch xuống dưới.
  + 1. Tầng Pooling

Tầng Pooling[4] (hay còn gọi Subsampling hoặc Downsample) là một trong những thành phần tính toán chính trong cấu trúc CNN. Xét về mặt toán học Pooling thực chất là quá trình tính toán trên ma trận, mục tiêu sau khi tính toán là giảm kích thước ma trận nhưng vẫn làm nổi bật lên được đặc trưng có trong ma trận đầu vào. Trong CNN toán tử Pooling được thực hiện độc lập trên mỗi kênh màu của ma trận ảnh đầu vào.

Có nhiều toán tử pooling như Sum-Pooling, Max-Pooling, L2-Pooling nhưng Max-Pooling thường được sử dụng. Max-Pooling sẽ giữ lại chi tiết quan trọng hay bằng cách giữ lại pixel có giá trị lớn nhất trong vùng thiết lập.

Ví dụ: Max-Pooling với bộ lọc 2x2 và stride = 2. Bộ lọc sẽ quét qua ảnh, với mỗi vùng ảnh được chọn, sẽ chọn ra 1 giá trị lớn nhất và giữ lại.



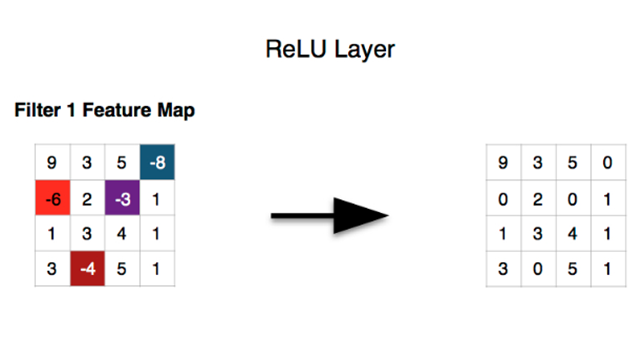
Hình 2.5: Max-Pooling với bộ lọc 2x2, stride = 2.

Hình 2.6: Tầng ReLU trong CNNHình 2.7: Max-Pooling với bộ lọc 2x2, stride = 2.

* + 1. Tầng ReLU

ReLU[4] layer áp dụng các kích hoạt (Activation function) max(0,x) lên đầu ra của tầng tích chập, có tác dụng đưa các giá trị âm về thành 0. Tầng này không thay đổi kích thước của ảnh và không có thêm bất kì tham số nào.

Mục đích của lớp ReLu là đưa ảnh một mức ngưỡng, ở đây là 0. Để loại bỏ các giá trị âm không cần thiết mà có thể sẽ ảnh hưởng cho việc tính toán ở các layer sau đó.



Hình 2.8: Tầng ReLU trong CNN

* + 1. Tầng Fully Connected (FC)

Sau các lớp Convolution và Pooling là lớp Fully Connected[4], để kết nối tất cả các feature đã tìm được ở các lớp trước đó. Tại tầng này, mỗi một nơ-ron của tầng này sẽ liên kết tới mọi nơ-ron của tầng khác.

Để đưa ảnh từ các tầng trước vào mạng này, buộc phải dàn phẳng bức ảnh ra thành 1 vector thay vì là mảng nhiều chiều như trước. Tại tầng cuối cùng sẽ sử dụng 1 hàm trong học máy đó là Softmax để phân loại đối tượng dựa vào vector đặc trưng đã được tính toán của các tầng trước đó.

* 1. Tập dữ liệu

Thông tin về tập dữ liệu, tập dữ liệu được sử dụng trong bài niên luận này là tập dữ liệu bao gồm hình ảnh của 26 chữ cái trong tiếng anh từ A đến Z và 10 số từ 0 đến 9, tổng là 36 ký tự. Tập dữ liệu trên là sự kết hợp giữa dữ liệu nhận dạng ký tự in hoa và bộ dữ liệu ký tự viết tay của NIST**.**

* Dung lượng của tập dữ liệu là: 2.94 GB.
* Mỗi tấm ảnh có kích thước là 128x128.
* Các giá trị nhãn từ 0 đến 36 tương ứng với 10 số từ 0 đến 9 và 26 chữ cái từ A đến Z.

Số lượng phần tử của mỗi lớp là:

|  |  |
| --- | --- |
| **Ký tự** | **Số lượng** |
| A | 20.835 |
| B | 12.756 |
| C | 34.703 |
| D | 15.073 |
| E | 16.856 |
| F | 18.869 |
| G | 36.375 |
| H | 9.381 |
| I | 6.858 |
| J | 11.762 |
| K | 7.989 |
| L | 14.160 |
| M | 24.992 |
| N | 24.247 |
| O | 55.825 |
| P | 28.614 |
| Q | 8.376 |
| R | 17.001 |
| S | 72.196 |
| T | 33.401 |
| U | 43.135 |
| V | 9.133 |
| W | 15.806 |
| X | 9.000 |
| Y | 15.1791 |
| Z | 11.497 |
| 0 | 20.499 |
| 1 | 20.014 |
| 2 | 20.069 |
| 3 | 20.211 |
| 4 | 20.259 |
| 5 | 21.199 |
| 6 | 20.982 |
| 7 | 20.529 |
| 8 | 20.358 |
| 9 | 21.456 |

Bảng 2.1: Tổng quan về tập dữ liệu

* Tập dữ liệu được chia thành 2 phần theo ti lệ 9:1.
* Tập train: gồm 709.595 file ảnh.
* Tập validation: gồm 72.000 file ảnh.
  1. Thiết kế và cài đặt
     1. Tiền xử lý dữ liệu

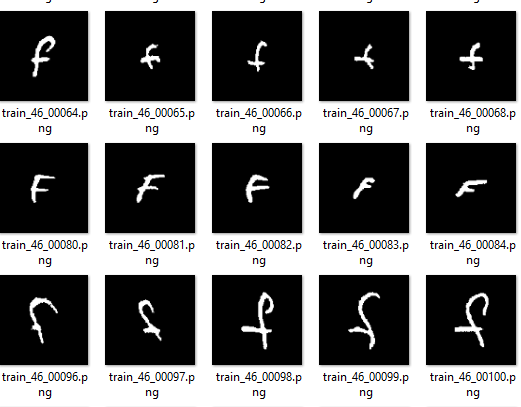
Sử dụng ImageDataGenerator của thư viện Keras để làm giàu dữ liệu trước khi đưa vào mô hình, được áp dụng cho cả phần train và validation, bản chất của hàm sinh dữ liệu là tạo ra các mẫu dữ liệu mới dựa trên các mẫu dữ liệu đã có sẵn trước đó.

Ta quan tâm đến một số tham số sau:

* **Rotation\_range**: chúng ta sẽ xoay ảnh ngẫu nhiên trong khoảng 'rotation\_range' độ.
* **Width\_shift\_range**: cắt ngẫu nhiên một phần chiều rộng theo tỷ lệ.
* **Height\_shift\_range**: cắt ngẫu nhiên một phần chiều cao theo tỷ lệ.
* **Rescale**: chuẩn hóa hay còn gọi là Normalize dữ liệu, giúp đưa giá trị của dữ liệu về khoảng 0..1 giúp quá trình training được nhanh hơn.
* **Shear\_range:** cắt ngẫu nhiên một góc theo chiều kim đồng hồ.
* **Horizontal\_flip**: lật ảnh theo chiều ngang, ở đây chúng ta set bằng False vì dữ liệu chúng ta là chữ, lật ngược ảnh sẽ làm ảnh hưởng đến độ chính xác dữ liệu.
* **Vertical\_flip**: lật ảnh theo chiều dọc, ở đây chúng ta set bằng False vì dữ liệu chúng ta là chữ, lật ngược ảnh sẽ làm ảnh hưởng đến độ chính xác dữ liệu.

Xử lý thủ công, lọc và loại bỏ các ảnh bị nhiễu ảnh hưởng đến độ chính xác của mô hình như:

* Ảnh với các pixel bị mờ.
* Ảnh không không thuộc các ký tự nhận dạng.
* Các ký tự bị lẫn với nhau.



Hình 2. 9: Kí tự hoa thường lẫn lộn

Hình 2.10: Các kí tự khác nhau lẫn lộn

Hình 2. 11: Kí tự hoa thường lẫn lộnHình 2.12: Các kí tự khác nhau lẫn lộn



* + 1. Lựa chọn mô hình

Mô hình được sử dụng để huấn luyện trong bài này là sử dụng mô hình MobileNet từ thư viện Keras. Keras đã hỗ trợ cho ta nhiều mô hình Delearning nổi tiếng như:

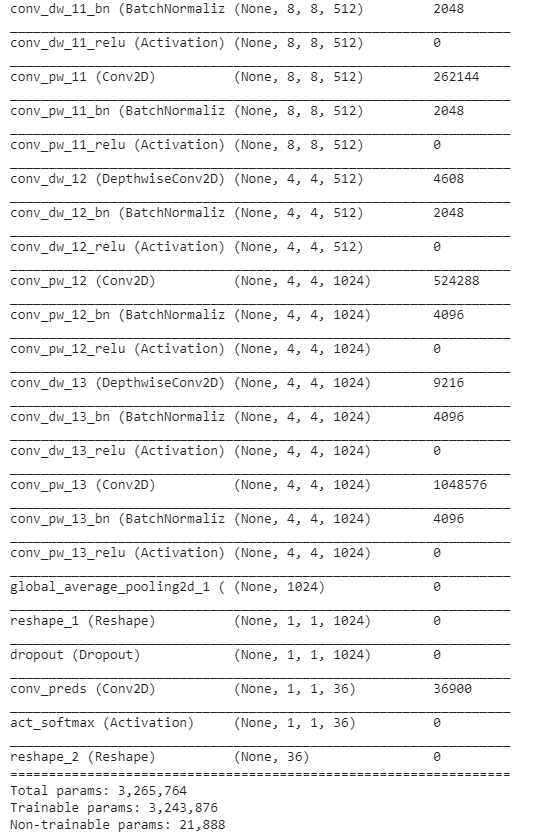
* VGG16.
* VGG19.
* MobileNet.
* MobileNetV2.
* ...

Vì sao chúng ta sử dụng mô hình MobileNet. Vì mô hình MobileNet cho hiệu năng tốt, chạy nhanh vì thế nó thích hợp để triển khi mô hình lên trình duyệt web.

Từ thư viện keras.applications.mobilenet import MobileNet, ở đây do ta sử dụng mô hình MobileNet nên ta sẽ sử dụng lại toàn bộ kiến trúc mạng của MobileNet:

* **Weights** = **None**: nếu **weights** = **True** thì mặc định MobileNet sẽ sử dụng weights đã được huấn luyện từ tập dữ liệu ImageNet.
* **Input\_shape**: có kích thước ảnh đầu vào và hệ màu của ảnh, trong bài này chúng ta sử dụng kích thức 128x128 và kênh màu là 3 (ảnh RGB).
* **Classes**: số lớp của ta là 36 lớp.
* **Include\_top**: bằng True nếu không có weights.

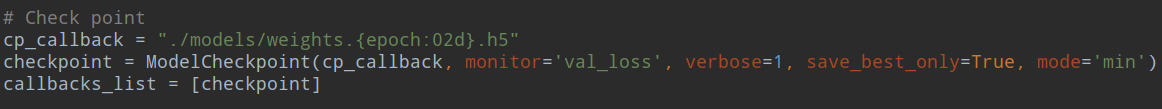
Kiến trúc của mô hình như sau:



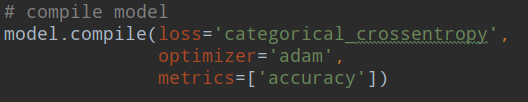
Hình 2.13: Kiến trúc của mô hình MobileNet

Có khoảng hơn 3 triệu tham số phải tối ưu trong quá trình huấn luyện.

* + 1. Định nghĩa mô hình
       1. Định nghĩa Callbacks

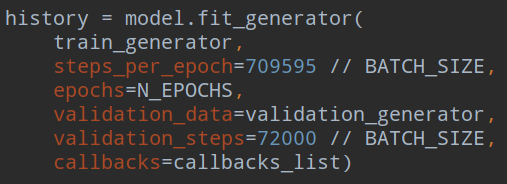
Định nghĩa Callbacks của mô hình giúp ta theo dõi quá trình huấn luyện của mô hình và lưu lại những mô hình đạt kết quả tốt nhất. Trong bài này ta sử dụng ModelCheckpoint để lưu lại mô hình tốt nhất.

Từ thư viện keras.callbacks import ModelCheckpoint:

* Cp\_callback: nơi để lưu lại mô hình tốt nhất, và ở đây ta sẽ lưu mô hình trong thư mục model.
* Ở đây chúng ta sử dụng val\_loss để biết được khi nào mô hình đạt kết quả tốt nhất. Ta sẽ lấy val\_loss thấp nhất.
  + - 1. Compile mô hình
* Sử dụng hàm mất mát là Categorical Cross-Entropy, do dữ liệu của ta được đọc theo dạng thư mục mỗi thư mục tương ứng một nhãn.
* Hàm tối ưu của ta là Adam.
  + 1. Huấn luyện mô hình

Đây là bước quan trọng và mất rất nhiều thời gian. Thời gian huấn luyện trên CPU sẽ chậm hơn rất nhiều so với việc chúng ta huấn luyện mô hình trên GPU. Nếu ta không có GPU mạnh và máy của chúng ta có cấu hình trung bình hoặc thấp thì ta có thể sử dụng Google\_Colab để huấn luyện.

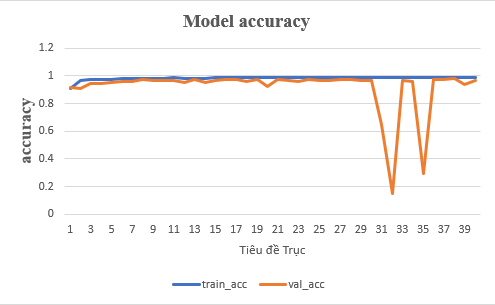
Google Colab là một dịch vụ đám mây miễn phí, hiện nay có hỗ trợ GPU (Tesla K80) và TPU (TPUv2).

Google Colab cài đặt sẵn cho chúng ta những thư viện rất phổ biến trong nghiên cứu Deep Learning như PyTorch, TensorFlow, Keras và OpenCV.

Ta tiến hành huấn luyện với số lần huấn luyện mô hình là 40 lần, mỗi lần là 1 giờ 30 phút. Lưu ý rằng Colab chỉ cho ta mượn 8 giờ 1 lần kết nối sau 8 giờ thì Colab sẽ tự động xóa các file dữ liệu của ta, nên chúng ta phải chia ra nhiều đợt huấn luyện và xong sau mỗi đợt huấn luyện chúng ta lấy mô hình về và huấn luyện tiếp thì ta tải mô hình lên lại cứ thế đủ 40 lần. Tổng số giờ huấn luyện mô hình trong bài này là 52 giờ.

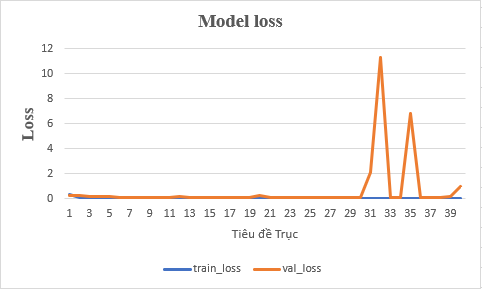
* + 1. Kết quả

Sau khi huấn luyện 40 lần, ta có được biểu đồ sau:



Hình 2.14: Độ chính xác trong 40 lần huấn luyện

Qua biểu đồ trên cho thấy mô hình MobileNet đã sử dụng phù hợp với tập dữ liệu, vì sau 5 lần huấn luyện thì Accuracy trên 94%. Và từ lần huấn luyện thứ 8 đến 30, độ chính xác của tập train(train\_acc) và tập test(val\_acc) không còn thay đổi nhiều và tương đối cao trên 96%. Từ 31 đảo đi thì mô hình không còn ổn định đã có sự biến động đột ngột của tập test(val\_acc).



Hình 2.15: Tỉ lệ lỗi trong 40 lần huấn luyện

Tương tự với Accuracy, thông qua giá trị Loss cho thấy lỗi của mô hình tương đối thắp từ lần huấn luyện thứ 6 đến 30. Như vậy, có thể thấy rằng với các trọng số từ lần lặp thứ 6 đến 30, ta có thể sự dụng cho quá trình dự đoán với tỷ lệ chính xác cao nhất.

* + 1. Đánh giá mô hình
       1. Ưu điểm

Qua các lần chạy thử nghiệm, mô hình chạy tương đối tốt, cải thiện được tình trạng học thuộc lòng của mô hình. Nhận dạng được các ký tự số và chữ in hoa với độ chính xác tương đối.

* + - 1. Khuyết điểm

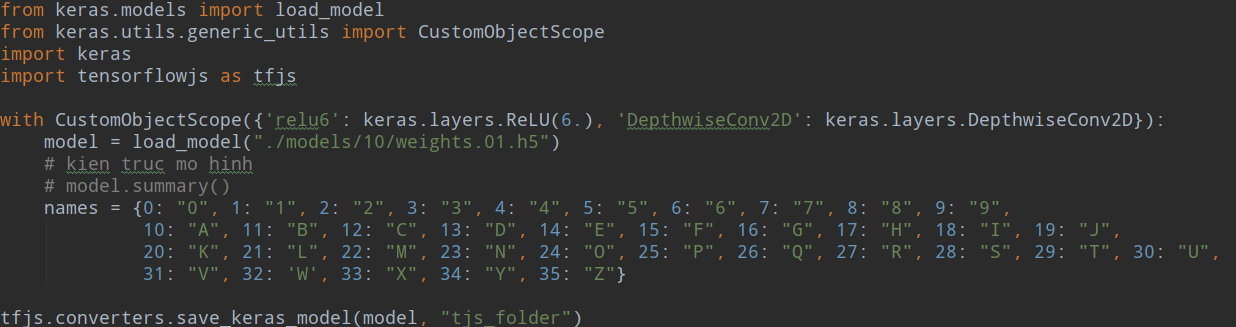
Những hạn chế của mô hình đã xây dựng, mô hình chỉ nhận dạng được từng ký tự, chưa nhận dạng được nhiều ký tự cùng lúc. Khi chạy thực tế vẫn có ký tự nhận dạng sai.

Thời gian tải mô hình lên và dự đoán chậm, có thể cải thiện bằng cách nâng thêm RAM và sử dụng ổ đĩa SSD để tăng tốc độ xử lý.

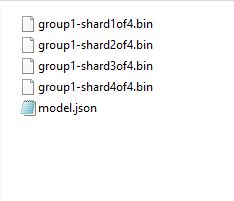
Mô hình chỉ mới nhận dạng được các chữ số và các chữ in hoa, chưa nhận dạng được tiếng việt có dấu, chữ thường và ký tự đặc biệt.

1. TRIỂN KHAI MÔ HÌNH LÊN TRÌNH DUYỆT WEB
   1. Chuyển đổi mô hình sang dạng Tensorflow JS

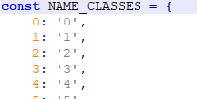
Tensorflow JS là một nền tảng mới của Tensorflow sử dụng để chạy các mô hình Deep Learning trên trình duyệt web. Nhưng trước tiên muốn thực hiện chúng ta cần phải cài đặt thư viện để chuyển đổi mô hình của chúng ta sang định dạng này.

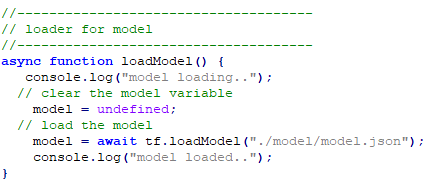
Câu lệnh cài đặt: pip install tensorflowjs

Câu lệnh trên sẽ thực hiện, tải mô hình lên và sử dụng thư viện Tensorflowjs để chuyển đổi mô hình thành các file TFJS. Các file được chuyển đổi sẽ nằm trong thư mục tjs\_folder.

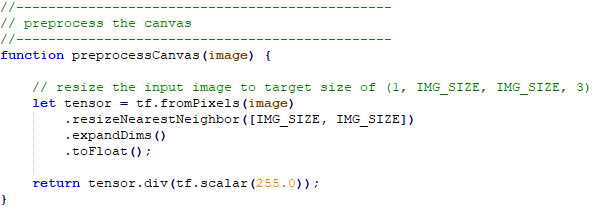
****Sau khi chuyển đổi ta được các file như sau:

* 1. Đưa mô hình lên web
     1. File Mapping Name

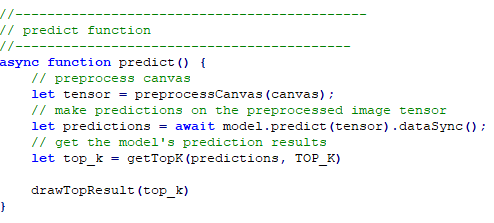
****Chứa thứ tự của các nhãn dự đoán, được tạo bằng list:

1. **Hàm loadModel**

Khi tải mô hình lên ta sẽ mất khoảng thời gian là 40s để mô hình tải xong. Có thể nhấn F12 để xem tiến trình tải mô hình, khi mô hình tải xong sẽ in ra thông báo ‘model loaded..’.

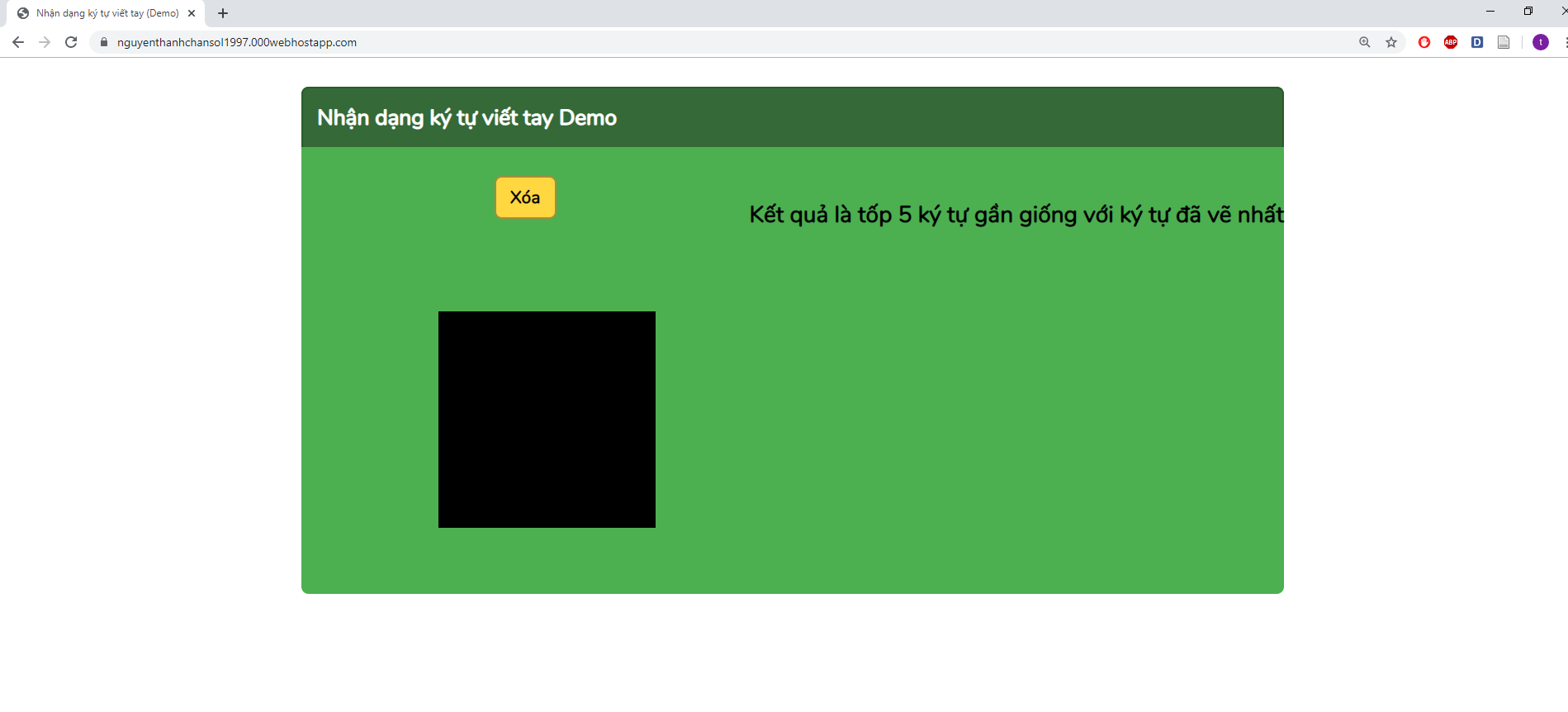
1. **Hàm xử lý dữ liệu đầu vào**

Hàm này chúng ta sẽ đưa ảnh về đúng kích thước đầu vào của mô hình là 128x128.

* + 1. Hàm dự đoán ký tự viết tay

Hàm này sẽ lấy dữ liệu từ canvas đi qua hàm xử lý dữ liệu rồi dự đoán bằng mô hình đã được tải lên. Kết quả được dự đoán sẽ được xử lý trong hàm getTopK tức là lấy ra top 5 ký tự dự đoán chính xác nhất. Độ chính xác giảm dần từ 1 đến 5.

* + 1. Giao diện chính



Hinh 3. 1: Giao diện Demo trên Website

Vùng được tô đen là nơi ta vẽ ký tự cần nhận dạng vào, kết quả sẽ được in phía dưới dòng chữ, kết quả là tốp 5 ký tự gần giống với ký tự đã vẽ nhất.

* 1. Demo chương trình

Link: <https://nguyenthanhchansol1997.000webhostapp.com/>

1. KẾT LUẬN
   1. Kết quả đạt được

Xây dựng được Website nhận dạng chữ viết tay tiếng Việt chạy với thời gian thực, độ chính xác tương đối.

Có được bộ dữ liệu nhận dạng chữ viết tay tiếng việt.

* 1. Hướng phát triển

Chúng ta có thể phát triển mô hình lên thành một ứng dụng nhận dạng các biểu mẫu viết tay của các tổ chức, công ty và cá nhân... Để nhập liệu và lưu trữ vào máy tính, hình thức nhận dạng chữ viết tay sẽ rút ngắn được đáng kể thời gian nhập liệu từ biểu mẫu vào máy tính khi muốn lưu trữ dữ liệu, giảm thiểu được chi phí thuê nhân công nhập liệu.

Xây dựng một API để nhận dạng các biểu mẫu với dữ liệu đầu vào là các kí tự đã được cắt sẵn, dự đoán và trả về kết quả là đoạn text các ký tự dự đoán.

Sử dụng Transfer Learning để cải thiện tốc độ học cũng như độ chính xác cho mô hình.

**TÀI LIỆU THAM KHẢO**

1. Lê Thanh Trúc, Phạm Nguyên Khang. "NHẬN DẠNG ĐIỂM VIẾT TAY TRÊN BẢNG ĐIỂM VỚI BIẾN ĐỔI HOUGH VÀ ĐẶC TRƯNG GIST." Tạp chí Khoa học Trường Đại học Cần Thơ (2015): 79-87.
2. Sustika, Rika, et al. "Evaluation of Deep Convolutional Neural Network Architectures for Strawberry Quality Inspection." International Journal of Engineering & Technology 7.4.40 (2018): 75-80.
3. Hijazi, Samer, Rishi Kumar, and Chris Rowen. "Using convolutional neural networks for image recognition." Cadence Design Systems Inc.: San Jose, CA, USA (2015).
4. Zhang, Aston, et al. "Dive into Deep Learning." May 19 (2019): 2019.