

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

BÁO CÁO ĐỒ ÁN

**MÔN HỌC: MÁY HỌC - MACHINE LEARNING**

**ĐỀ TÀI**

**NHẬN DẠNG CHỮ VIẾT TAY TIẾNG VIỆT**

**Giảng viên hướng dẫn:** Phạm Nguyễn Trường An

Lê Đình Duy

**Sinh viên thực hiện:** Tô Thanh Hiền - 19521490

Trần Vĩ Hào - 19521482 Trương Quốc Bình - 19521270

**Lớp:** CS114.L22.KHCL

Thành phố Hồ Chí Minh, ngày 19 tháng 7 năm 2021

MỤC LỤC

Chương 1. Tổng quan 4

1. Mô tả bài toán 4

2. Mô tả dữ liệu 4

Chương 2. Các nghiên cứu trước 6

Chương 3. Xây dựng bộ dữ liệu 7

1. Cách thức xây dựng bộ dữ liệu 7

2. Số lượng, độ đa dạng dữ liệu 8

3. Code cắt ảnh và zoom, resize ảnh sau khi cắt: 9

Chương 4. Training và test model 13

1. Data Preprocessing 13

1.1. Data Cleaning 13

1.1.1. Denoise 13

1.1.1.1. Lựa chọn phương pháp 13

1.1.1.2. Experiment 14

1.1.2. Segmentation & Morphological 17

1.1.2.1. Lựa chọn phương pháp 17

1.1.2.1.1. Erosion, dilation, opening & closing 17

1.1.2.1.2. Threshold 18

1.1.2.1.2.1. Simple threshold 18

1.1.2.1.2.2. Adaptive threshold 19

1.1.2.2. Experiment 19

1.1.2.3. Kết luận 22

1.2. Data augmentation 23

1.2.1 Addition of noise 23

1.2.2. Rotation 24

1.2.3. Translation 24

1.2.4. Shearing 25

1.2.5. Experiment 25

1.2.6. Kết luận 27

2. Building model 27

2.1. CNN (Convolutional Neural Network) 28

2.2. CRNN (Convolutional Recurrent Neural Network) 29

2.3. So sánh 2 model 30

2.4. Kết luận 30

3. Final run 31

3.1. Preprocessing 31

3.1.1. Data cleaning 31

3.1.2. Data Augmentation 31

3.2. Model sử dụng 33

3.2.1 ReLu activation function 34

3.2.2. Dropout 34

3.2.3. Optimizer 34

3.2.4. Loss funtion 34

4. Kết quả cuối cùng (test result) 34

5. Nhận xét 35

Chương 5. Ứng dụng và hướng phát triển 36

1. Ứng dụng 36

2. Hướng phát triển 36

Tài liệu tham khảo 37

# Chương 1. Tổng quan

## 1. Mô tả bài toán

Nhận diện chữ viết tay là ứng dụng cốt lõi của thị giác máy tính. Đây là ứng dụng tuy không hề mới mẻ nhưng vẫn còn đang phát triển và có nhiều tiềm năng ở Việt Nam. Trong bài báo cáo sẽ giới thiệu model nhận diện chữ cái viết tay tiếng Việt xây dựng bằng Convolutional Neural Network (CNN) và bằng Convolutional Recurrent Neural Network (CRNN).

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 thực sự là một thách thức đối với những nhà nghiên cứu.

Chữ viết tay xuất hiện ở hầu hết trong các công việc của các cơ quan, nhà máy, xí nghiệp, trường học. Việc nhập một văn bản viết tay trên giấy vào máy tính hầu như chỉ có một cách giải quyết đó là gõ lại từng ký tự, việc này luôn chiếm nhiều thời gian và đôi khi không đảm bảo tiến độ hoạt động của công việc. Việc nhận dạng chữ viết tay hầu như đã có nhiều nhà nghiên cứu thử sức, nhưng phần lớn họ chỉ nhận diện chữ viết tiếng Anh, vậy những ngôn ngữ của các quốc gia khác thì sao? Với những lý do trên nhóm em đã chọn nghiên cứu đề tài: **“Nhận dạng chữ viết tay tiếng Việt”**.

Có thể mô tả tổng quát bài toán như sau:

* Bài toán thuộc dạng bài phân loại.
* Input đầu vào sẽ là một tấm ảnh trong đó có chứa đúng một chữ cái tiếng Việt. Output (dạng text) sẽ là chữ cái tương ứng với tấm ảnh đó.

**VD:**

Input:



Output: **a**

## 2. Mô tả dữ liệu

Bảng chữ cái Việt Nam bao gồm 29 chữ cái. Trong đó 22 chữ cái là thuộc chữ cái latin. Có 7 chữ cái biến thể bằng cách thêm dấu. Không những thế chữ cái còn có thêm 6 ngữ âm là ngang, sắc, huyền , hỏi, ngã, nặng. Kết hợp với các nguyên âm sẽ tạo nên các biến thể khác. Tổng cộng chúng ta sẽ có 89 loại chữ cần thu thập.

Trong quá trình thu thập dữ liệu xuất hiện nhiều khó khăn ngoài ý muốn như sau:

* Do tình hình giãn cách xã hội nên không thể chỉ dẫn cụ thể, chỉ có thể điều khiển mọi việc từ xa dẫn đến việc dữ liệu thu thập được không đồng nhất. Người viết có xu hướng viết chữ lấn viền của ô thu thập data nên khi đưa vào code cắt ảnh, những tấm ảnh ấy đều bị xóa trắng không dùng được, trên lí thuyết mỗi bộ thu dataset sẽ thu được mỗi chữ cái 10 ảnh, nhưng thực tế số ảnh dùng được là rất ít, có khi không thu được ảnh nào.
* Ngoài ra còn có các vấn đề khác như: người cho dataset viết chữ theo mẫu của máy tính, đôi khi người cho không “nhiệt tình” nên chữ viết rất nguệch ngoạc, việc thu thập cũng như phân loại dữ liệu đều được làm thủ công bằng tay nên rất mất thời gian,…

Trước đây đã có người thu thập bộ dữ liệu trên nhưng nó hoàn toàn không public, nhóm em đã có tiếp cận xin bộ dataset trên nhưng không nhận được sự đồng ý từ họ. Chúng em đã tự thu thập dữ liệu và nhận thấy bộ dữ liệu của nhóm em cũng tạm ổn, nhờ vào sự “không nhiệt tình” của người cho chữ viết mà qua đó bộ dữ liệu của nhóm em lại càng thêm phong phú. Ngoài ra việc mô tả chi tiết bộ dữ liệu sẽ được đề cập cụ thể hơn ở Chương 3.

# Chương 2. Các nghiên cứu trước

Đối với lớp CS114 trước đã có nhóm [15] cùng làm đề tài này ở kỳ trước. Dataset chữ cái tiếng việt mà nhóm sử dụng là dataset nhóm bạn thu thập với các bạn trong lớp. Nhóm bạn đã sử dụng SVM, logistic regression và MLP tuy nhiên kết quả không tốt. Kết quả tốt nhất thì test accuracy chỉ đạt 26% trong khi đó val-acc đạt 50%. Dễ thấy model bị overfitting khi validation accuracy cao gấp đôi test accuracy. Nguyên nhân cho việc này theo nhóm em hiểu là SVM và logistic regression không bao giờ thích hợp cho việc xử lý ảnh. Còn MLP lại quá đơn giản chỉ với 1 layer. Một nguyên nhân khác có thể do chính dataset không đủ dữ liệu.

Nhóm em có xem thêm các colab notebook trên dataset MNIST. MNIST dataset được xem như “hello world” dataset của thị giác máy tính. Đây là dataset về bảng chữ cái tiếng anh với chữ số. Đây là ví dụ nhóm em tham khảo [9]. Model được xây dựng trên kiến trúc CNN. Model đã đạt được training accuracy là 93,4% và validation accuracy là 98,7% chỉ sau 2 epoch nên model hoàn toàn có thể đạt được 99% val-acc nếu epoch là 30. Đây là 1 kết quả tuyệt vời. Validation accuracy cao hơn training accuracy chứng tỏ model không bị overfitting.

# Chương 3. Xây dựng bộ dữ liệu

## 1. Cách thức xây dựng bộ dữ liệu

Bộ dữ liệu thu thập: Thủ công.

Về cách thu thập bộ dữ liệu phải phù hợp với tiêu chí đặt ra trong quá trình nhóm làm việc.

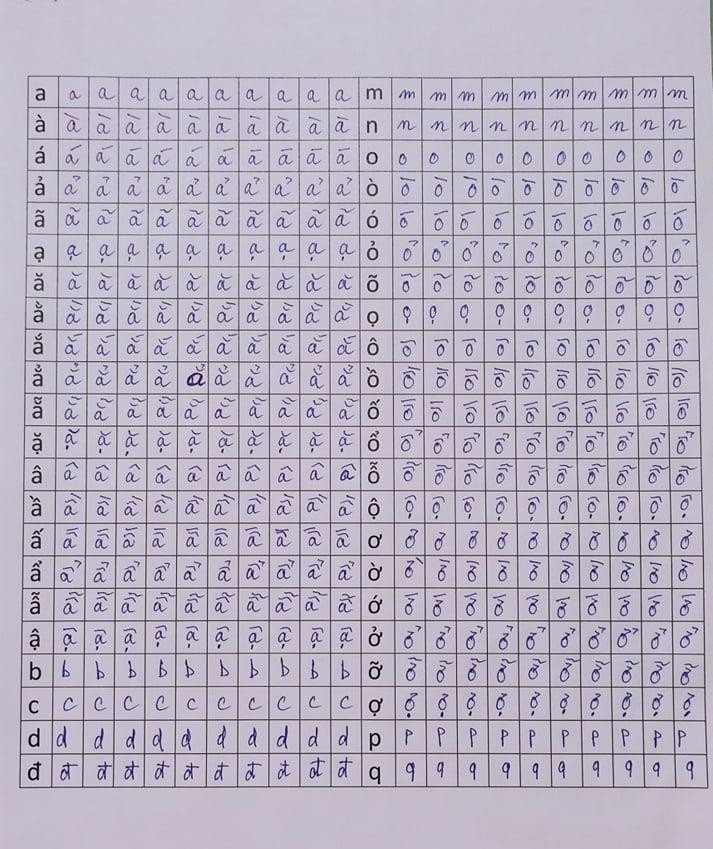
Quy trình thì nhóm tụi em đã làm các tờ giấy thu nhập chữ viết phân phát cho người thân và bạn bè. Nhóm em đã share dataset với nhóm của bạn Nguyễn Dương Hải. Một bộ thu dữ liệu sẽ có hai tờ giấy, bao gồm 89 chữ cái, mỗi chữ cái sẽ được điền 10 lần. Sau đó các tờ giấy sẽ được thu nhập và chụp hình lại. Hình sau đó sẽ được phân tách ra thành từng chữ cái và dán nhãn. Tuy là việc thu thập thủ công có hơi khó khăn vì ảnh hưởng dịch bệnh Covid và khá tốn thời gian nhưng nhờ vậy đã đảm bảo được tính đa dạng của bộ dữ liệu.

Ngoài ra nhóm em còn thống nhất với nhau các tiêu chí sau:

* Chữ viết không được dính với phần viền của ô chữ.
* Viết như chữ viết thường ngày mọi người hay dùng.
* Dùng bút bi đen hoặc bút chì để viết.
* Khi chụp ảnh mẫu thu dữ liệu sẽ chụp theo tỉ lệ 3x3, căn chỉnh để lấy đúng phiếu thu, không được để xuất hiện các yếu tố ngoại cảnh vào tấm ảnh.

Thời gian thu thập bộ dữ liệu rơi vào khoảng 3 tuần.

Kết quả thu được hơn 120 mẫu thu thập chữ viết và chất lượng các mẫu thu thập chữ viết chỉ dừng lại ở mức tạm ổn.



Hình 1. Một mẫu thu nhập chữ viết

## 2. Số lượng, độ đa dạng dữ liệu

Bộ dataset chữ cái viết tay tiếng Việt sau khi thu thập có tổng cộng 50114 hình ảnh đã được phân loại, bao gồm:

|  |  |  |
| --- | --- | --- |
| a - 1356 | ệ - 411 | r - 707 |
| à - 513 | g - 282 | s - 701 |
| ả - 543 | h - 430 | t - 747 |
| ã - 472 | i - 1127 | u - 908 |
| á - 551 | ì - 873 | ù - 753 |
| ạ - 615 | ỉ - 818 | ủ - 742 |
| ă - 667 | ĩ - 828 | ũ - 699 |
| ằ - 468 | í - 873 | ú - 702 |
| ẳ - 443 | ị - 834 | ụ - 711 |
| ẵ - 449 | k - 559 | ư - 776 |
| ắ - 470 | l - 560 | ừ - 661 |
| ặ - 474 | m - 584 | ử - 647 |
| â - 607 | n - 670 | ữ - 654 |
| ầ - 479 | o - 1242 | ứ - 599 |
| ẩ - 456 | ò - 461 | ự - 654 |
| ẫ - 410 | ỏ - 481 | v - 728 |
| ấ - 476 | õ - 433 | x - 754 |
| ậ - 485 | ó - 432 | y - 574 |
| b - 418 | ọ - 491 | ỳ - 658 |
| c - 806 | ô - 520 | ỷ - 338 |
| d - 438 | ồ - 423 | ỹ - 368 |
| đ - 372 | ổ - 387 | ý - 434 |
| e - 828 | ỗ - 398 | ỵ - 346 |
| è - 496 | ố - 425 |  |
| ẻ - 468 | ộ - 414 |  |
| ẽ - 467 | ơ - 462 |  |
| é - 471 | ờ - 362 |  |
| ẹ - 468 | ở - 334 |  |
| ê - 536 | ỡ - 300 |  |
| ề - 459 | ớ - 290 |  |
| ể - 430 | ợ - 306 |  |
| ễ - 368 | p - 496 |  |
| ế - 409 | q - 388 |  |

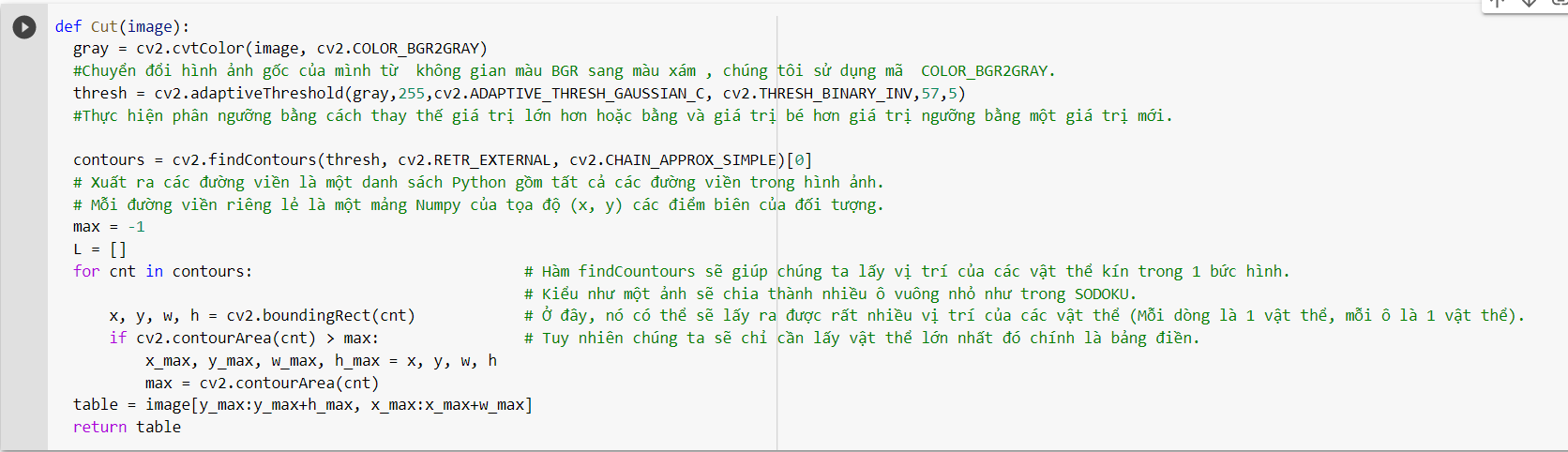
**\* SPLIT DATASET:**

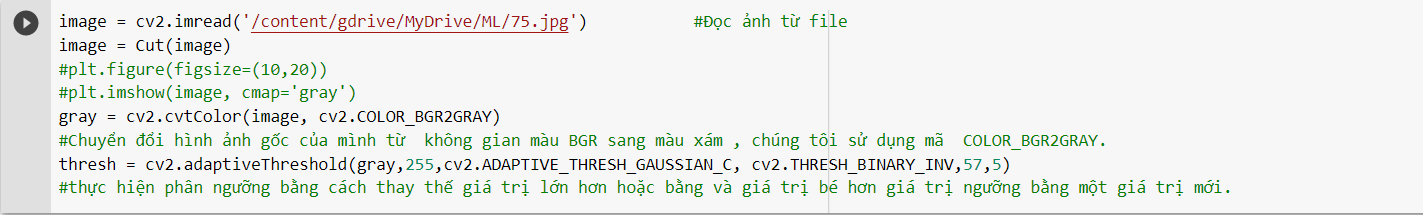
Dataset được chia làm ba set: 30068 ảnh cho Training set (60%), 10023 ảnh cho Validation set (20%) và 10023 ảnh cho Test set (20%).

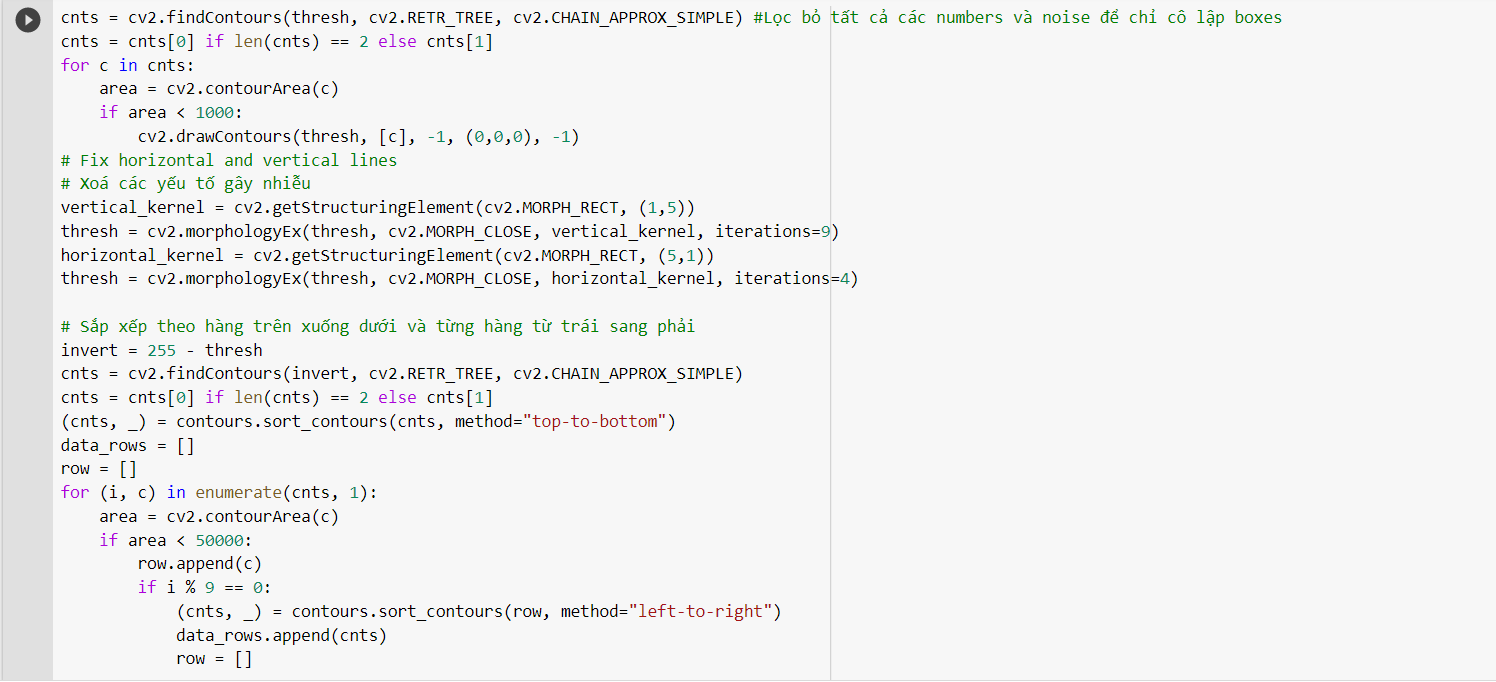
## 3. Code cắt ảnh và zoom, resize ảnh sau khi cắt:

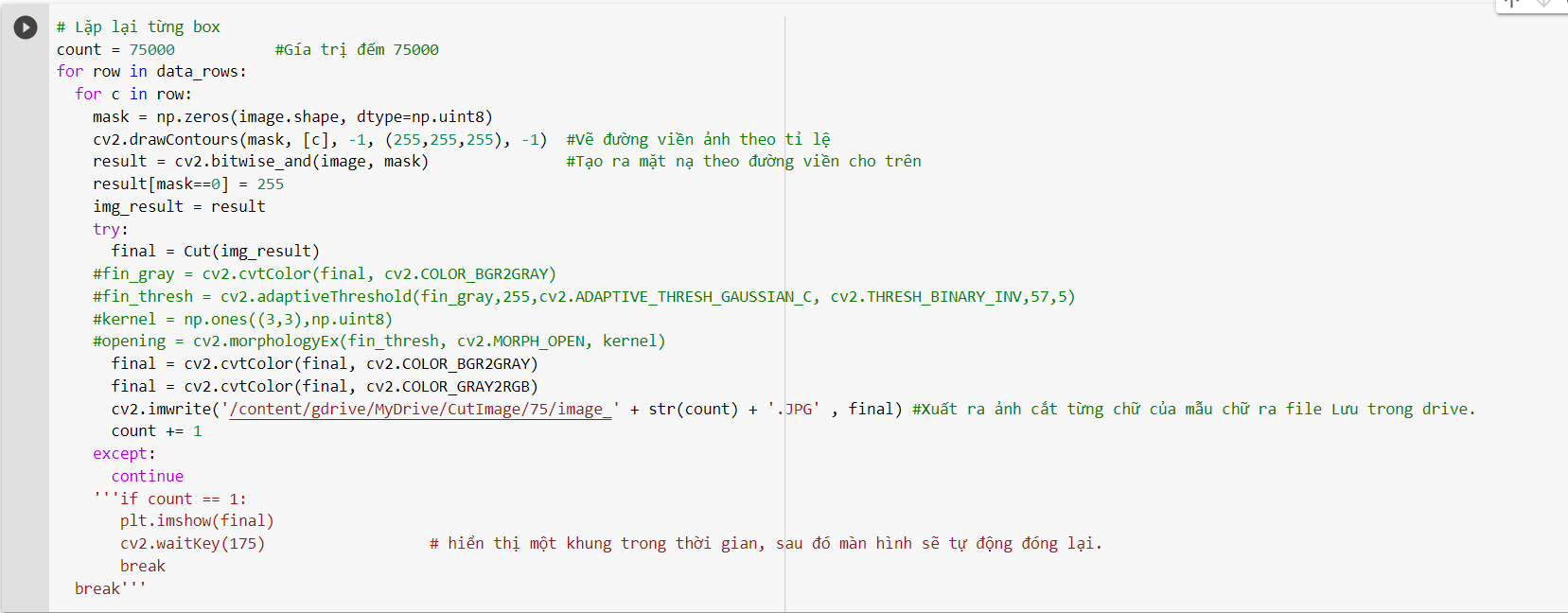
Code cắt ảnh nhóm em tham khảo từ nhóm của bạn Nguyễn Dương Hải, việc dùng chung code cắt ảnh và xử lí đơn giản trước trong quá trình cắt ảnh giúp bộ dữ liệu của cả hai nhóm được đồng nhất.

**CODE CẮT ẢNH TỪ ẢNH MẪU THU THẬP CHỮ VIẾT:**

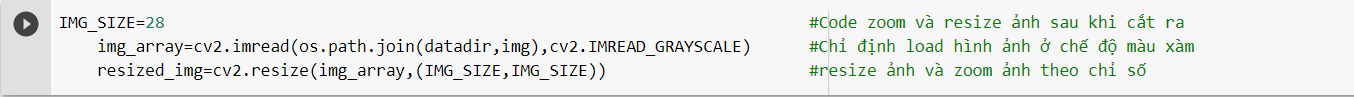








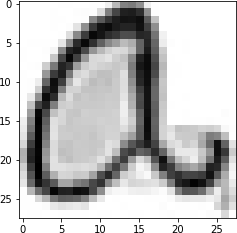
**CODE RESIZE VÀ ZOOM ẢNH:**



Một ảnh chữ ảnh sau khi cắt ra



Sau khi cắt ảnh và zoom resize ta sẽ được

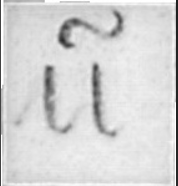


**Khó khăn khi cắt, zoom và resize ảnh:**

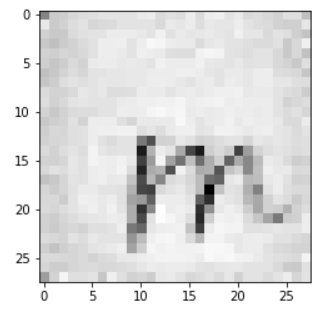
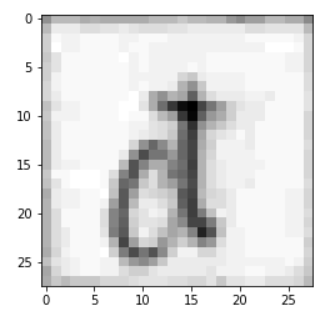
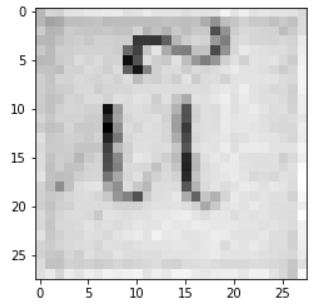
Trong quá trình cắt ảnh thì nhóm tụi em cũng gặp nhiều khó khăn vì rất nhiều ảnh cắt ra từ mẫu thu thập chữ viết, có ảnh bị mờ nét chữ, có ảnh không thấy rõ chữ, có ảnh bị xóa trắng nền… thế nên tụi em đã thủ công chọn lọc ra những ảnh không phù hợp.

VD:

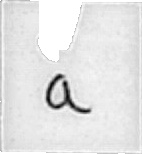
Ảnh trước khi zoom và resize cũng đã mờ và không rõ nét



Ảnh sau khi zoom và resize mờ, có khoảng trắng và nhiều nhiễu:



Trong trường hợp này, đây đáng lẽ là một chữ a có dấu ví dụ như á, ă, ẳ,v.v… nhưng sau khi cắt ảnh đã khiến cho dấu bị mất đi, nguyên nhân có thể do nét bút ấy đã dính với phần viền của ô từ đó tạo nên khoảng trắng, khoảng trắng này rất có thể sẽ gây khó khăn cho model trong việc học nét chữ.



# Chương 4. Training và Test model

## 1. Data Preprocessing

Xử lý tiền ảnh là bước vô cùng quang trọng để chuẩn bị dữ liệu cho các model. Có rất nhiều bước quan trọng trong xử lý tiền dữ liệu như data cleaning, data transformation và feature selection. Một tập dữ liệu sẽ chứa rất nhiều biến số, một biến số sẽ chứa rất nhiều thông tin. Vì vậy để đơn giản hóa các chiều không gian của model, chúng ta sẽ chỉ chọn những biến số độc đáo và chứa thông tin quan trọng.

### 1.1. Data Cleaning

#### 1.1.1. Denoise

##### **1.1.1.1. Lựa chọn phương pháp**

OpenCV cung cấp 1 số công cụ để loại bỏ nhiễu ra khỏi ảnh:

**Gaussian Blurring**

**Description: A picture containing logo

Description automatically generated**

Hình 2: Ảnh trước và sau Gaussian blur

**Average**

**Description: Graphical user interface

Description automatically generated with low confidence**

Hình 3: Ảnh trước và sau blur

**Median Blurring**

**Description: A picture containing text

Description automatically generated**

Hình 4: Ảnh trước và sau khi Median blur

**Bilateral Filtering**

**Description: A picture containing text, building material

Description automatically generated**

Hình 5: Ảnh trước và sau khi Bilateral blur

##### **1.1.1.2. Experiment**

Ở đây nhóm em sẽ kiểm thử từng phương pháp một. Dataset sẽ được xử lý qua từng phương pháp như Gaussian Bluring , Median Blurring,… Kết quả sẽ được đánh giá qua val-loss và val-acc.

**Dataset**: ta sẽ dùng dataset đã thu thập được

**Proposed model architecture**

**Description: Table

Description automatically generated**

Hình 6: Model sử dụng

**Result:**

Training set: 3333, val set: 1667 sau 30 epoch

|  |  |  |
| --- | --- | --- |
|  | VAL-ACC | Val-LOSS |
| raw | 0.8666 | 0.5545 |
| Average | 0.8430 | 0.6659 |
| Gaussian | 0.8755 | 0.5658 |
| MEDIAN | 0.8343 | 0.6783 |
| Bilateral | 0.8335 | 0.8335 |

Training set: 37586, val set: 12528 sau 20 epoch

|  |  |  |
| --- | --- | --- |
|  | Val-acc | val-loss |
| Raw | 0,8902 | 0,3456 |
| average | 0,8076 | 0,5825 |
| gaussian | 0,9073 | 0,2997 |
| MEDIAN | 0,8579 | 0,4559 |
| bilateral | 0,8902 | 0,3481 |

**Kết luận:**

Với cả dataset nhỏ và dataset lớn, Gaussian blur hoặc data raw cho ra kết quả tốt nhất.

**Nguyên nhân:**

Các phương pháp xử lý ảnh không chỉ xóa bớt nhiễu số nhưng đồng thời cũng lỡ làm mờ thêm các edge (cạnh) làm mất đi 1 số feature quan trọng.

Description: A picture containing text, clock

Description automatically generated

Hình 7: ảnh trước và sau median blur

Dễ thấy các phương pháp Denoise không chỉ xóa nhiễu mà còn làm mờ luôn các cạnh của chữ vốn đã mờ do chất lượng ảnh không tốt. Có thể hiểu chính chất lượng ảnh không tốt đã dẫn tới việc mất 1 phần chữ sau xử lý loại nhiễu.

#### 1.1.2. Segmentation & Morphological

##### **1.1.2.1. Lựa chọn phương pháp**

###### **1.1.2.1.1. Erosion, dilation, opening & closing**

**Erosion**

**Description: Graphical user interface, application

Description automatically generated**

Hình 8: Ảnh trước và sau Erode

Erosion giống như tên gọi của nó. Nó làm xói mòn (erode) loại bỏ lớp viền (cạnh) giúp cho vật nhỏ hơn.

**Dilation**

**Description: A picture containing text, clock, screenshot

Description automatically generated**

Hình 9: Ảnh trước và sau dilation

Hoàn toàn ngược lại với erosion, nó làm tăng kích thước của vật.

**Opening**

**Description: Icon

Description automatically generated**

Hình 10: Ảnh trước và sau opening

**Closing**

**Description: Icon

Description automatically generated**

Hình 11: Ảnh trước và sau closing

Qua thử nghiệm nhóm em quyết định sẽ không sử dụng những phương pháp này. Chất lượng của dataset là rất kém, không phân biệt được giữa chữ (đối tượng) và background. Có nhiều ảnh sau khi qua xử lý đã mất dạng hoàn toàn chỉ còn hình background.

###### **1.1.2.1.2. Threshold**

**1.1.2.1.2.1. Simple threshold**

Đơn giản mà nói trong phương pháp Thresholding nếu pixel có giá trị nhỏ hơn giá trị threshold nó sẽ trở thành 0, ngược lại sẽ thành 255.

**Binary**

**Description: Chart

Description automatically generated**

Hình 12: Ảnh trước và sau threshold

**Trunc**

**Description: Graphical user interface, application

Description automatically generated**

Hình 13: Ảnh trước và sau threshold

**Tozero**

**Description: Chart

Description automatically generated**

Hình 14: Ảnh trước và sau threshold

**1.1.2.1.2.2. Adaptive threshold**

Khác với Simple threshold, khi Threshold là con số cố định. Nếu bức ảnh có điều kiện chiếu sáng khác nhau ở từng khu vực, Adaptive có thuật toán xác định Threshold của pixel dựa trên khu vực nhỏ xung quanh nó.

**Adaptive Mean threshold**

**Description: Chart

Description automatically generated with medium confidence**

Hình 15: Ảnh trước và sau threshold

**Adaptive Gaussian threshold**

**Description: A picture containing qr code

Description automatically generated**

Hình 16: Ảnh trước và sau threshold

##### **1.1.2.2. Experiment**

**Dataset :** ta sẽ sử dụng dataset thu nhập được

Ở đây nhóm em sẽ kiểm thử từng phương pháp Segmentation và Morphological. Nhưng chúng ta sẽ xử lý dataset qua Gaussian blur trước, sau đó sẽ so sánh với các kết quả của dataset raw hay chỉ được xử lý qua phương pháp Segmentation và Morphological mà không qua Gaussian blur. Kết quả sẽ được đánh giá qua val-acc và val-loss.

**Proposed model architecture**

**Description: Table

Description automatically generated**

Hình 17: Model sử dụng

**Training set: 3333, Val set: 1667 sau 30 epochs**

Có Gaussian blur, Simple threshold:

|  |  |  |
| --- | --- | --- |
|  | Val-ACC | VAL-LOSS |
| Binary | 0.801 | 0.7961 |
| TOZERO | 0.8636 | 0.5711 |
| TRUNC | 0.6558 | 1.3378 |

Có Gaussian blur, Adaptive threshold:

|  |  |  |
| --- | --- | --- |
|  | VAL-ACC | VAL-LOSS |
| thresh-mean | 0,78 | 0,7438 |
| thresh-gaussian | 0,7922 | 0,7917 |

Không Gaussian blur, Simple threshold:

|  |  |  |
| --- | --- | --- |
|  | VAL-ACC | VAL-LOSS |
| Binary | 0,8065 | 0,8482 |
| ToZERO | 0,892 | 0,4661 |
| TRUNC | 0,6558 | 1,397 |

Không Gaussian blur, Adaptive threshold:

|  |  |  |
| --- | --- | --- |
|  | VAL-ACC | VAL-LOSS |
| Thresh-mean | 0,6955 | 1,1861 |
| thresh-GAUSSIAN | 0,6225 | 1,5902 |

So với Raw dataset:

|  |  |
| --- | --- |
| Val-ACC | val-loss |
| 0,8493 | 0,5898 |

**Training set: 37586 , Val set:12528 sau 10 epochs**

Có Gaussian blur, Simple threshold:

|  |  |  |
| --- | --- | --- |
|  | Val-ACC | VAL-LOSS |
| Binary | 0,6604 | 1,234 |
| TOZERO | 0,8645 | 0,4610 |
| TRUNC | 0,0268 | 4,435 |

Có Gaussian blur, Adaptive threshold:

|  |  |  |
| --- | --- | --- |
|  | val-acc | val-loss |
| THresh-mean | 0,8373 | 0,5317 |
| thresh-gaussian | 0,8506 | 0,4814 |

Không Gaussian blur, Simple threshold:

|  |  |  |
| --- | --- | --- |
|  | val-acc | val-loss |
| binary | 0,7238 | 0,9818 |
| trunc | 0,0268 | 4,4356 |
| tozero | 0,8716 | 0,4417 |

Không Gaussian blur, Adaptive threshold:

|  |  |  |
| --- | --- | --- |
|  | val-acc | val-loss |
| Thresh-MEAN | 0,6623 | 1,4653 |
| THRESH-GAUSSIAN | 0,6854 | 1,3452 |

So với Raw dataset:

|  |  |
| --- | --- |
| Val-ACC | val-loss |
| 0,8911 | 0,3123 |

##### **1.1.2.3. Kết luận**

**Không** sử dụng các phương pháp segmentation & morphological là tốt nhất.

**Nguyên nhân:**

Mỗi bức ảnh là từ nhiều nguồn, khác nhau do cách chụp ảnh trước khi cắt nên chất lượng sau khi cắt cũng khác nhau nên threshold giữa background và chữ cũng rất khác nhau.

Description: A picture containing graphical user interface

Description automatically generated

Hình 18: Minh họa cho khác nhau giữa chất lương ảnh

Vì vậy khó mà đặt được một Threshold cho tất cả các ảnh. Nên có ảnh sẽ mất đi các feature quan trọng. Adaptive threshold cũng không khá khẩm hơn. Khi trong dataset này ảnh sau xử lý Adaptive threshold lại xuất hiện các lõm trắng và đen lẫn lộn.

Description: A picture containing qr code

Description automatically generated

Hình 19: Ảnh minh họa cho lõm trắng của adaptive threshold

### 1.2. Data augmentation

Data augmentation là phương thức tạo ra dataset mới từ dataset đã có. Ví dụ một dataset mới đã được tạo ra bằng cách quay dataset cũ một góc 10 độ cùng chiều kim đồng hồ. Kết hợp cả hai dataset ta có một dataset mới.

#### 1.2.1 Addition of noise

Description: Graphical user interface, chart, application

Description automatically generated

Hình 20: Ảnh trước và sau adding noise

#### 1.2.2. Rotation

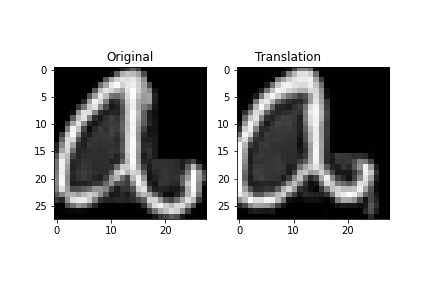
**Description: Graphical user interface, chart

Description automatically generated**

Hình 21: Ảnh trước và sau rotation

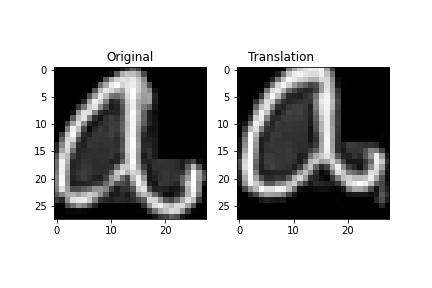
#### 1.2.3. Translation

**Horizontal**

****

Hình 22: Ảnh trước và sau khi Horizontal translation

**Vertical**

****

Hình 23: Ảnh trước và sau Vertical translation

#### 1.2.4. Shearing

**Description: Graphical user interface, application

Description automatically generated**

Hình 24: Ảnh trước và sau shearing

#### 1.2.5. Experiment

**Set up:**

Sẽ có 5000 bức ảnh được chọn trong dataset thu nhập được. 5000 ảnh sẽ chia thành traing set 3334 val set 1666. Training set sẽ được thử qua từng phương pháp Rotation, Translation, Scaling, Shearing và Noise addition.

**Proposed model architecture**

Description: Table

Description automatically generated

Hình 25: Model được sử dụng

**Result:**

|  |  |  |
| --- | --- | --- |
|  | val-acc | val-loss |
| Raw | 0,8452 | 0,5964 |
| Rotation | 0,9090 | 0,4272 |
| HORIZONTAL Translation | 0,9090 | 0,4000 |
| VERTICAL TRANSLATION | 0,8985 | 0,3981 |
| Gaussian noise | 0,9151 | 0,3728 |
| SHEARING | 0,8549 | 0,5461 |

#### 1.2.6. Kết luận

Ngoại trừ Shearing với chênh lệch val-acc giữa Raw và Shearing là dưới 1% thì tất cả các phương pháp Data augmentation đều cho kết quả tốt nằm vào khoảng 89% - 91%. Tốt nhất là Gaussian noise với val-acc là 0,9151 và val-loss là 0,3728.

**Nguyên nhân:**

Các phương pháp đều làm gia tăng dataset vì thế làm tăng accuracy. Ngoại trừ Shearing vì khi Shearing làm cho augmented dataset biến dạng nhiều so với ảnh gốc tạo ra các false feature cho máy để học.

## 2. Building model

Ở đây nhóm em sẽ xây dựng 2 model dựa trên cấu trúc CNN và CRNN.

### 2.1. CNN (Convolutional Neural Network)

**Description: Diagram

Description automatically generated**

### 2.2. CRNN (Convolutional Recurrent Neural Network)

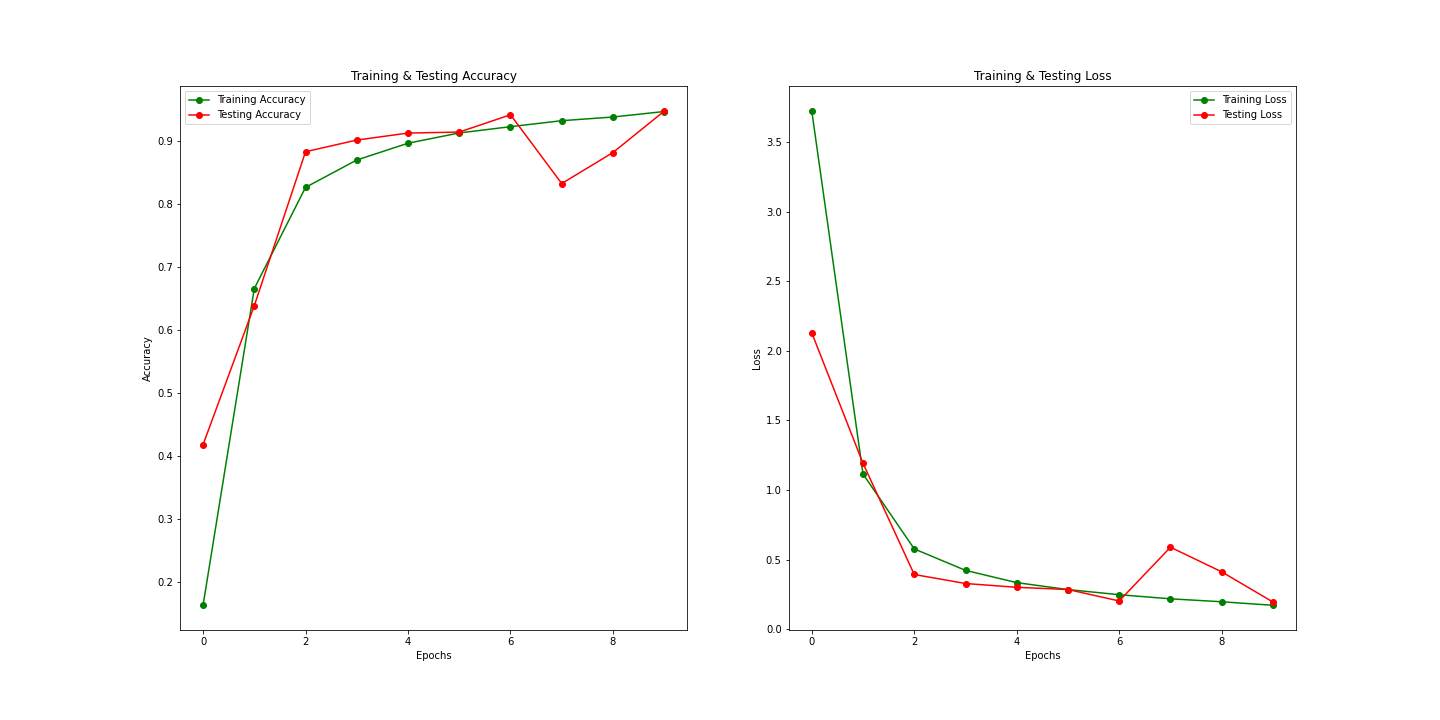
**Description: Diagram

Description automatically generated**

### 2.3. So sánh 2 model

Dataset sẽ chia thành training set: 37585 và val set: 12528.

**CNN**

****

**CRNN**

**Description: Chart, line chart

Description automatically generated**

### 2.4. Kết luận

Cả 2 model đều fit ổn, không có sự chênh lệch lớn của accuracy và loss giữa Training set và Val set. Tuy nhiên CRNN fit ổn định hơn. Khó có thể kết luận cái nào tốt hơn.

## 3. Final run

### 3.1. Preprocessing

#### 3.1.1. Data cleaning

Ảnh sẽ được chuyển từ ảnh nền trắng chữ đen sang ảnh nền đen chữ trắng bằng OpenCV bit\_wise not sau đó sẽ được Gaussian Blur, cuối cùng resize về kích thước (28, 28).

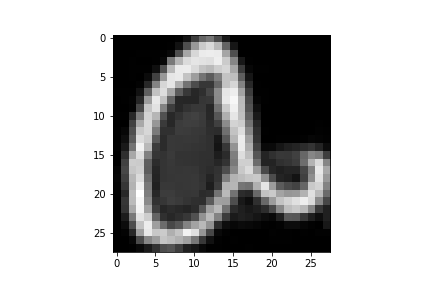
#### 3.1.2. Data Augmentation

Dataset sẽ được tăng kích thước dataset lên 4 lần bằng các phương pháp Data augmentation. Các phương pháp sử dụng là counter clockwise ten degree rotation, Horizontal translation về phía trái hoặc kết hợp horizontal translation về phía trái sau đó counter clockwise ten degree rotation theo khuyến nghị của [5] Data augumentation of handwritten character dataset for Convolutional Neural Network and integration into a Bayensian Linear Framework.

Description: Graphical user interface, application

Description automatically generated with medium confidence

**Original**



**Counter clockwise rotation**

Description: Application

Description automatically generated with low confidence

**Horizontal translation**

Description: Graphical user interface, application

Description automatically generated with medium confidence

**Horizontal translation and counter clockwise rotation**

### 3.2. Model sử dụng

**Description: Diagram

Description automatically generated**

#### 3.2.1 ReLu activation function

ReLU (Rectified Linear Unit) function là 1 non-linear function. ReLU là 1 function phổ biến.

#### 3.2.2. Dropout

Dropout giúp cho model không bị overfitting. Nó giúp model trong lúc train tránh bị hiện tượng overfitting.

#### 3.2.3. Optimizer

Nhóm em sẽ sử dụng Adam.

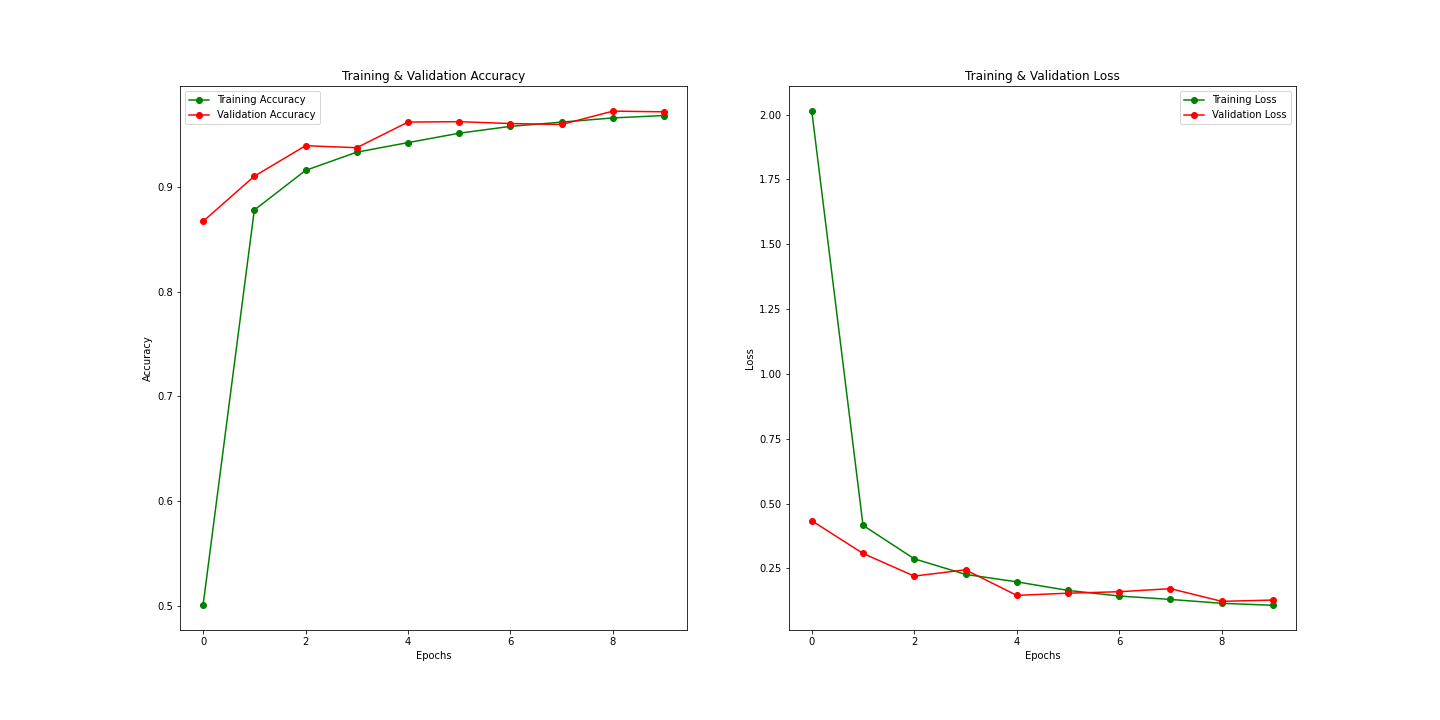
#### 3.2.4. Loss funtion

Loss function là categorical\_crossentropy cho multi-label classification.

## 4. Kết quả cuối cùng (Test result)

**Training process:**

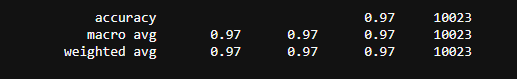
Network sẽ train trong 10 epoch, mỗi epoch có 1879 step với batch size là 64. Thời gian train mỗi epoch là khoảng 2000 giây.



Kết quả tập Training set: 0.9681

Kết quả tập Val set: 0.9716

**TEST RESULT:**

****

**Kết quả tập test set: 0.972**

## 5. Nhận xét

Model có độ chính xác cao. Accuracy đối với Test set là 0.97. Model fit tốt cụ thể là 0,9681 đối với Training set; 0,9716 với Val set và 0,972 với Test set.

Tuy vậy, vẫn có sự lẫn lộn giữa những class có nét tương đồng cao như ‘ê’ với ‘ể’ hay ‘ề’. Ngoài ra còn lẫn lộn giữa ‘o’ và ‘ỏ’ hay ‘ơ’ và ‘ọ’ v.v…

**Nguyên nhân:**

* Dataset chưa đủ nhiều. Dataset không cân bằng giữa các class. Có class lại có trên 1500 ảnh nhưng có class chỉ dưới 300 ảnh. Và không ngoài dự đoán, những class dự đoán thiếu chính xác nhất chính là những class có ít ảnh nhất.
* Dataset có chất lượng không tốt. Có ảnh nhìn rõ, có ảnh thì mờ.
* Xử lý ảnh vẫn chưa tốt. Xử lý nhiễu sẽ loại bỏ một phần các edge (cạnh) của ảnh làm mờ chữ. Vẫn chưa tìm ra phương pháp Segmentation & Morphological thích hợp. Các phương pháp đều không hiệu quả khi có ảnh mất trắng.

**Hướng cải thiện:**

* Tăng số lượng dataset. Nếu đảm bảo ít nhất mỗi class có thể đạt tối thiểu trên 1000 ảnh thì model có khả năng sẽ đạt độ chính xác trên 99%
* Học cách xử lý mô hình tốt hơn
* Tìm kiếm các phương pháp xử lý ảnh tốt hơn.

# Chương 5. Ứng dụng và hướng phát triển

## 1. Ứng dụng

Bài toán này ra đời trong bối cảnh trên thị trường các ứng dụng nhận dạng chữ viết dùng để nhập liệu vào máy còn kém trong việc nhận dạng tiếng Việt.

Ta lấy ví dụ một trường hợp như sau: ta đang có một văn bản giấy viết tay cần nhập liệu vào máy tính để in ra văn bản hoàn chỉnh, theo lẽ tự nhiên thì chúng ta sẽ chọn cách ngồi gõ lại, nhập liệu tay tài liệu ấy. Việc làm trên sẽ rất mất thời gian nếu ta gặp phải một khối lượng tài liệu lớn. Vì vậy việc có một ứng dụng có thể chụp ảnh lại tài liệu ấy và xuất ra text trên máy tính hoặc điện thoại là thực sự hữu ích trong trường hợp này. Nếu bài toán này thành công và thực sự được phát triển thành một ứng dụng cụ thể thì sẽ giúp ích rất ích nhiều cho các đối tượng thường xuyên tiếp xúc nhiều với các loại tài liệu văn bản như nhân viên văn phòng, học sinh, sinh viên,…

Thực tế, trên thế giới cũng đã có những ứng dụng tương tự ví dụ như Google Lens, qua một thời gian sử dụng thì nhóm em thấy được nhược điểm của ứng dụng trên như sau:

* Độ chính xác cao với tiếng Anh, tiếng Việt vẫn không cao lắm.
* Scan được những chữ viết được in trên sách báo, văn bản được in từ máy tính,…. nhưng độ chính xác lại thấp với chữ viết tay.

Từ những lẽ trên mà nhóm em đã quyết định chọn giải quyết bài toán nhận dạng chữ viết tay tiếng Việt này, những sự hữu ích mà bài toán này mang lại là không phải bàn cãi.

## 2. Hướng phát triển

Model trên có độ chính xác vào khoảng 97%, là một con số ấn tượng, tuy nhiên trong tương lai nhóm em sẽ cải thiện thêm nữa, ví dụ như:

* Do model này được thực hiện trên Google Colab, mà thời gian giới hạn của một phiên làm việc chỉ là 12 tiếng, vì vậy model chỉ học được tối đa 10 epoch, nếu có một server đủ lớn mạnh và có thể treo máy trong khoảng thời gian dài thì nhóm em sẽ để model học nhiều hơn, nhóm em tin rằng độ chính xác hoàn toàn có thể lên đến con số 99%.
* Bài toán trên chỉ mới dừng lại ở việc dự đoán riêng lẻ từng chữ cái và tổ hợp của các chữ cái với các thanh âm, vì vậy để phục vụ cho các mục đích mà nhóm em hướng tới trong phần ứng dụng thì nhóm em phải phát triển bài toán thành nhận dạng từ, câu, đoạn văn bản… cụ thể là một tổ hợp gồm nhiều kí tự.
* Việc phát triển ứng dụng dựa trên một bài toán giàu tiềm năng như thế này thật sự rất cần thiết, nếu ứng dụng được phát triển thành công thì lại có nhiều hướng mới mở ra, ngoài việc giúp ích trong việc nhập liệu, chúng ta còn có thể kết hợp với các công cụ tìm kiếm để tìm thông tin dựa vào văn bản mình nhận dạng được.

# Tài liệu tham khảo

[1] <https://docs.opencv.org/4.5.2/d7/d4d/tutorial_py_thresholding.html>

[2] <https://www.geeksforgeeks.org/erosion-dilation-images-using-opencv-python/>

[3] <https://docs.opencv.org/3.4/db/df6/tutorial_erosion_dilatation.html>

[4] <https://docs.opencv.org/4.5.2/d4/d13/tutorial_py_filtering.html>

[5] <https://theses.ubn.ru.nl/bitstream/handle/123456789/2620/Klep%2C_D.M.J._1.pdf?sequence=1>

[6]f<https://towardsdatascience.com/improving-accuracy-on-mnist-using-data-augmentation-b5c38eb5a903>

[7] <https://towardsdatascience.com/image-pre-processing-c1aec0be3edf>

[8] <https://towardsdatascience.com/get-started-with-using-cnn-lstm-for-forecasting-6f0f4dde5826>

[9] <https://towardsdatascience.com/data-preprocessing-and-network-building-in-cnn-15624ef3a28b>

[10]<https://www.researchgate.net/publication/341798621_Vietnamese_handwritten_character_recognition_using_convolutional_neural_network>

[11] <https://www.kaggle.com/yassineghouzam/introduction-to-cnn-keras-0-997-top-6>

[12] <https://www.kaggle.com/residentmario/automated-feature-selection-with-sklearn>

[13] <https://www.kaggle.com/samfc10/handwriting-recognition-using-crnn-in-keras>

[14] <https://www.tensorflow.org/tutorials/images/data_augmentation>

[15] <https://github.com/thanhnhan311201/CS114.L11.KHCL>