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

BÁO CÁO ĐỒ ÁN

MÔN HỌC: MÁY HỌC

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

I. ABSTRACT

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).

II. INTRODUCTION

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.

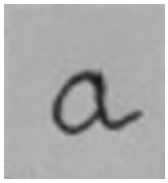
Chữ viết tay xuất hiện ở hầu hết trong các công việc cảu các cơ quan, nhà máy, xí nghiệp, trường học. Việc nhập một văn bản viết tay trên giấy vào máy tính hầu như chỉ có một cách giải quyết đó là gõ lại từng ký tự, việc này luôn chiếm nhiều thời gian và đôi khi không đảm bảo tiến độ hoạt động của công 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 sẽ là chữ cái tương ứng với tấm ảnh đó.

VD:

Input:

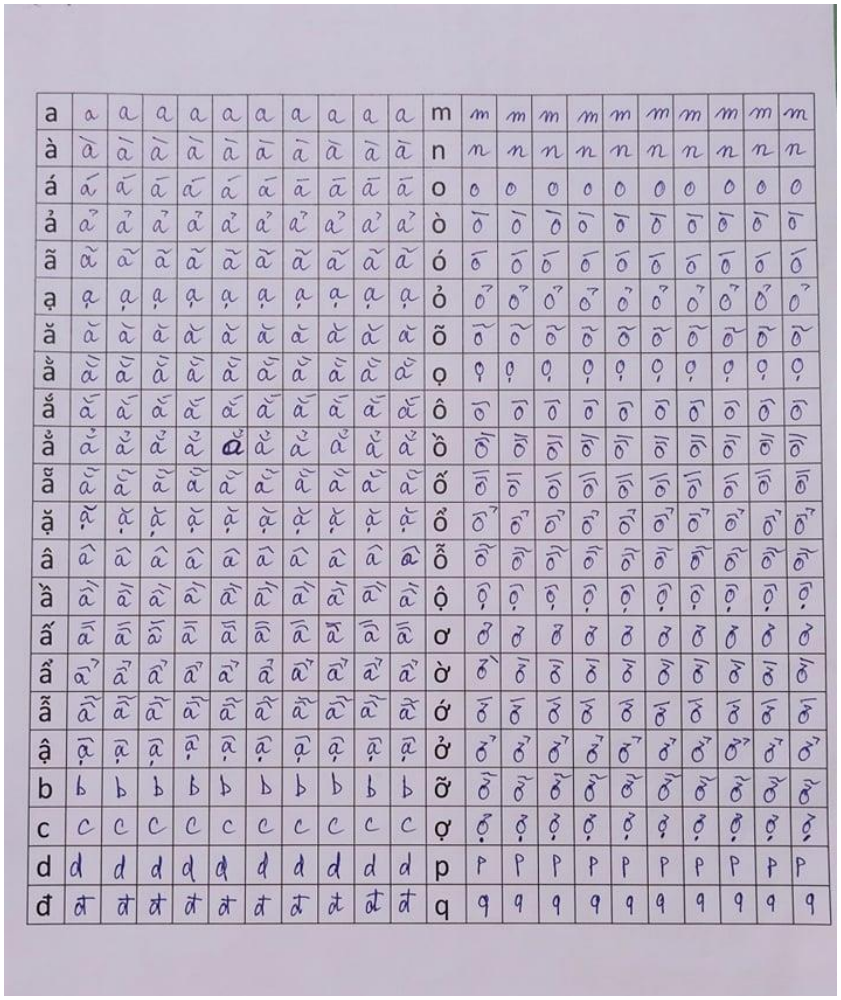


Output: **a**

III. DATASET

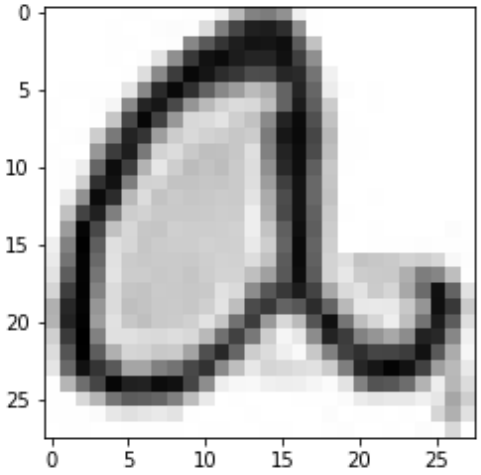
Bảng chữ cái Việt Nam bao gồm 29 chữ cái. Trong đó 22 chữ cái là thuộc chữ cái latin. Có 7 chữ cái biến thể bằng cách thêm dấu. Không những thế chữ cái còn có thêm 6 ngữ âm là ngang, sắc , huyền , hỏi, ngã, nặng. Kết hợp với các nguyên âm sẽ tạo nên các biến thể khác. Tổng cộng chúng ta sẽ có 89 loại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

Nhóm em đã làm các tờ giấy thu nhập chữ viết phân phát cho người thân và bạn bè. Ngoài ra, nhóm em sẽ share dataset với nhóm của bạn Nguyễn Dương Hải. Một bộ mẫu thu dataset 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. Đây là phương thức vô cùng mất thời gian nhưng đảm bảo tính đa dạng của dataset.



Hình 1. Một mẫu thu nhập chữ viết

Từng ảnh chứa chữ cái sẽ được zoom và resize để có cùng kiểu phân bố



Hình 2. Một mẫu ảnh chữ đã được zoom và resize

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

Dataset được chia làm ba set: 30068 ảnh cho Training set, 10023 ảnh cho Val set và 10023 ảnh cho Test set.

IV. DATA PREPROCESSING

Tiền xử lý ảnh là một bước vô cùng quan 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.

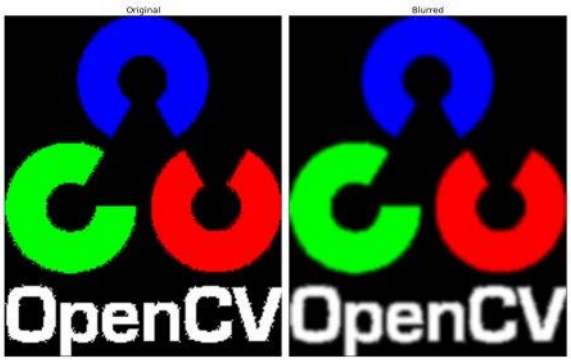
1. Data Cleaning

1.1. Denoise

1.1.1. Lựa chọn phương pháp

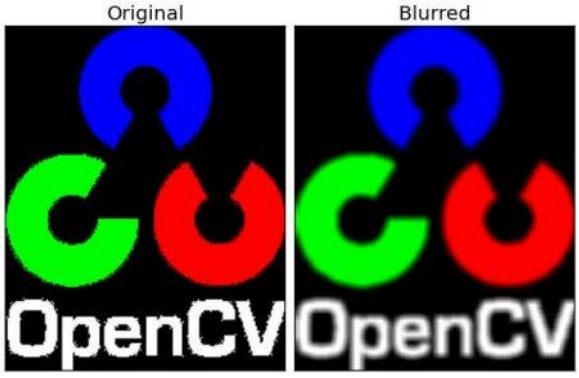
OpenCV cung cấp một số công cụ để loại bỏ nhiễu ra khỏi ảnh.

1.1.1.1. Gaussian Blurring



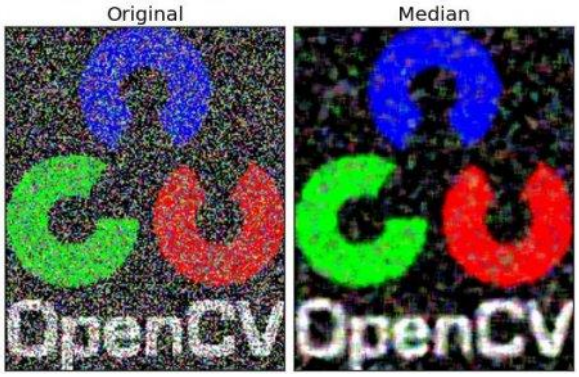
Hình 3: Ảnh trước và sau Gaussian blur

1.1.1.2. Average



Hình 4: Ảnh trước và sau blur

1.1.1.3. Median Blurring



Hình 5: Ảnh trước và sau khi Median blur

1.1.1.4. Bilateral Filtering

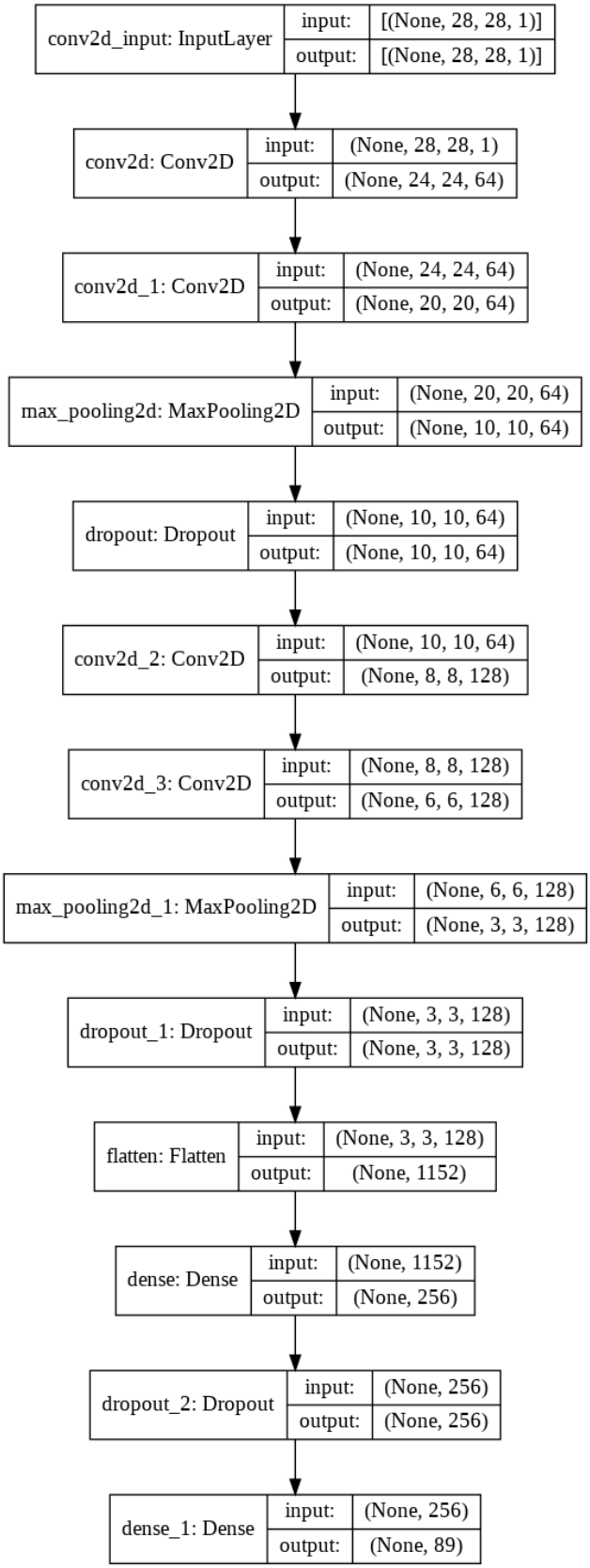
Hình 6: Ảnh trước và sau khi Bilateral blur

1.1.2. Experiment

Ở đây chúng ta sẽ kiểm thử từng phương pháp. 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



Hình 7: 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:

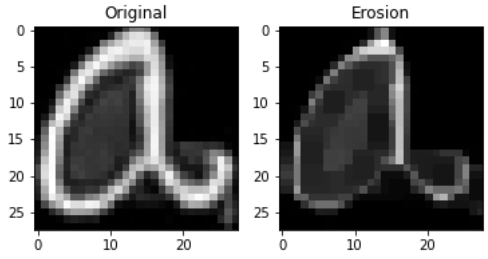
Với cả dataset nhỏ và dataset lớn, Gaussian blur hoặc không xử lý gì hết cho ra kết quả tốt nhất.

1.2. Segmentation & Morphological

1.2.1. Lựa chọn phương pháp

1.2.1.1. Erosion, dilation, opening & closing

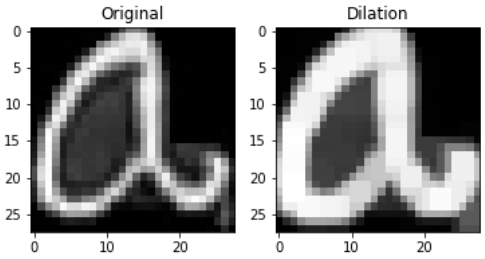
Erosion



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



Hình 9: Ảnh trước và sau dilation

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

Opening

Hình 10: Ảnh trước và sau opening

Closing

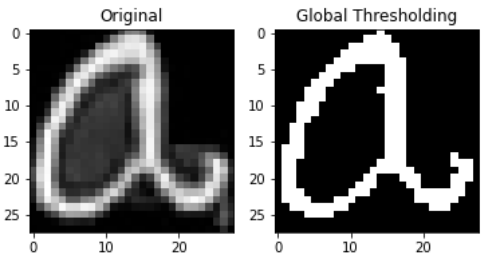
Hình 11: Ảnh trước và sau closing

Chất lượng của dataset là không đồng đều. Nên có nhiều ảnh sau khi opening đã mất dạng hoàn toàn.

1.2.1.2. Threshold

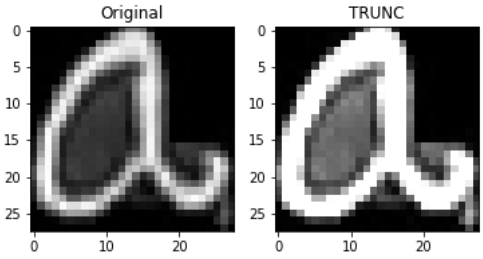
1.2.1.2.1. Simple threshold

Binary



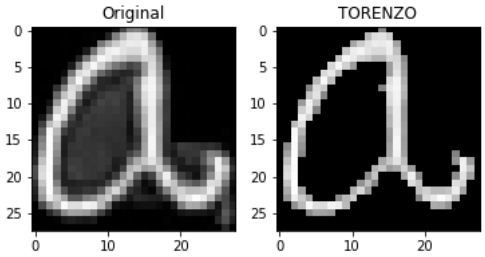
Hình 12: Ảnh trước và sau threshold

Trunc



Hình 13: Ảnh trước và sau threshold

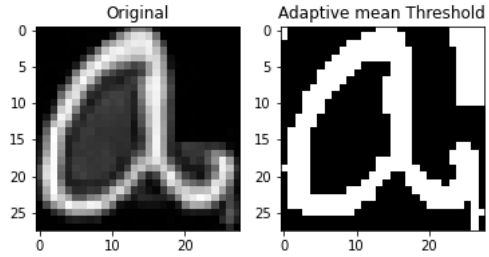
Torenzo



Hình 14: Ảnh trước và sau threshold

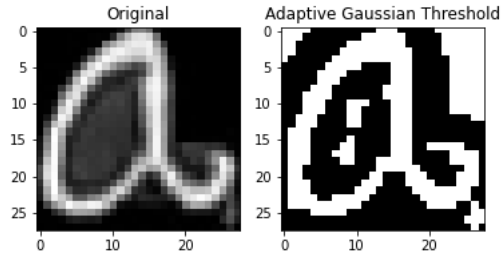
1.2.1.2.2. Adaptive threshold

Adaptive Mean threshold



Hình 15: Ảnh trước và sau threshold

Adaptive Gaussian threshold



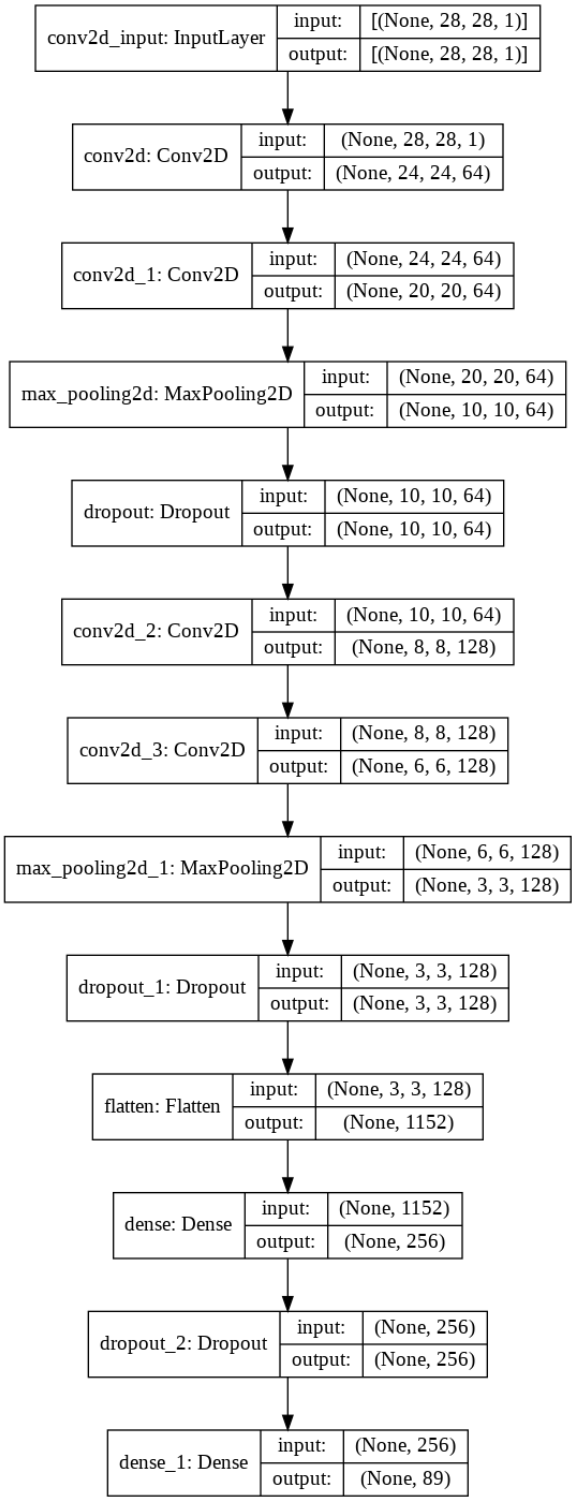
Hình 16: Ảnh trước và sau threshold

1.2.2. Experiment

Dataset : ta sẽ sử dụng dataset thu nhập được.

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

Proposed model architecture



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

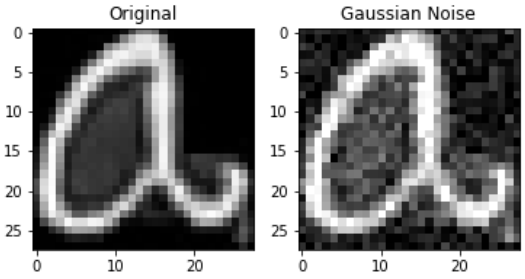
1.2.3. Kết luận:

Không sử dụng các phương pháp segmentation & morphological là tốt nhất. Với các dataset nhỏ khoảng 5000 ảnh xử lý segmentation & morphological làm tăng độ chính xác. Nhưng với các dataset lớn hơn thì ngược lại.

2. Data augmentation

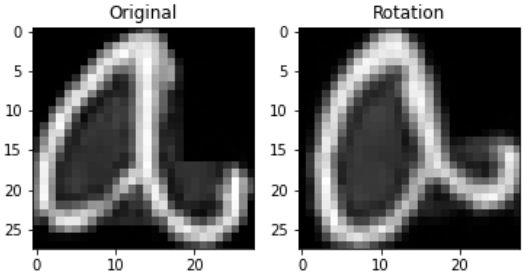
Data augmentation là phương thức tạo ra dataset mới từ dataset đã có. Ví dụ một dataset mới đã được tạo ra bằng cách quay dataset cũ một góc 10 độ cùng chiều kim đồng hồ. Kết hợp cả 2 dataset ta có 1 dataset mới.

2.1. Addition of noise



Hình 18: Ảnh trước và sau adding noise

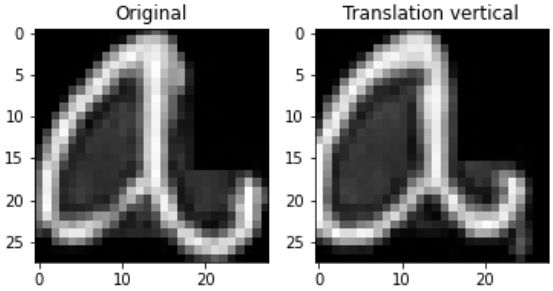
2.2. Rotation



Hình 19: Ảnh trước và sau rotation

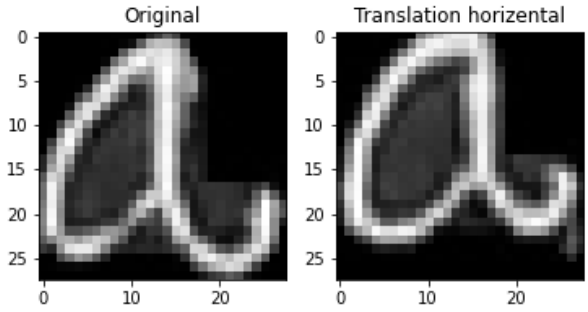
2.3. Translation

Vertical



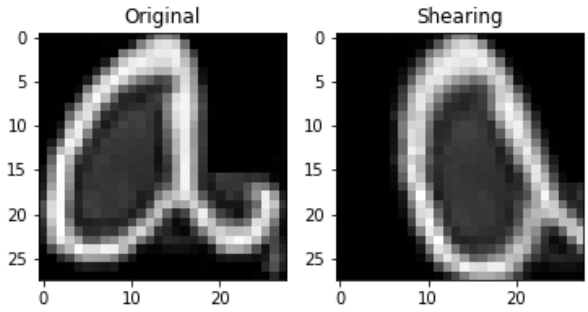
Hình 20: Ảnh trước và sau khi vertical translation

Horizontal



Hình 21: Ảnh trước và sau Horizontal translation

2.4. Shearing



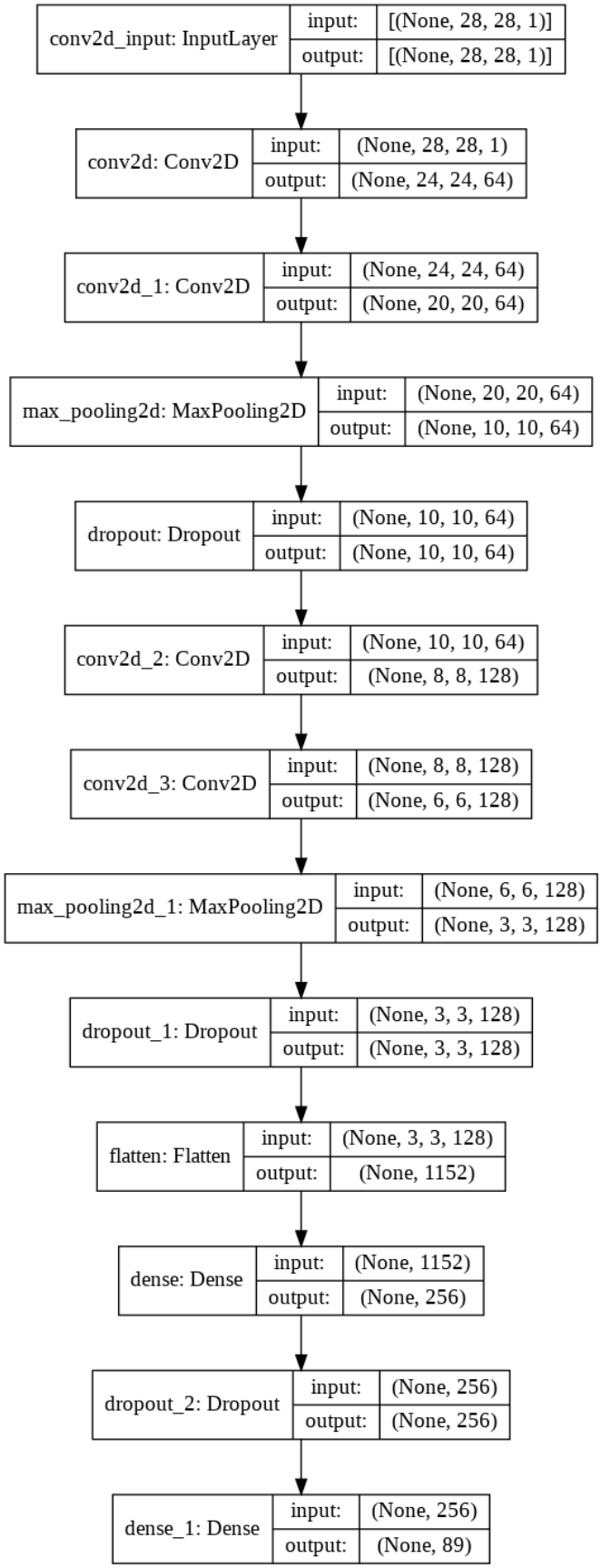
Hình 22: Ảnh trước và sau shearing

2.5. Experiment

Set up:

Sẽ có 5000 bức ảnh được chọn trong bộ dataset thu nhập được, 5000 ảnh sẽ chia thành Traing set: 3334 và Val set: 1666. Traing set sẽ được thử qua từng phương pháp rotation, translation, scaling, shearing và noise addition.

Proposed model architecture



Hình 23 : Model được sử dụng

Result:

	VAL-ACC	VAL-LOSS
RAW	0,8452	0,5964
ROTATION	0,8632	0,5213
VERTICAL TRANSLATION	0,8711	0,4865
HORIZONTAL TRANSLATION	0,8588	0,5557
GAUSSIAN NOISE	0,8490	0,6123
SHEARING	0,8523	0,5877

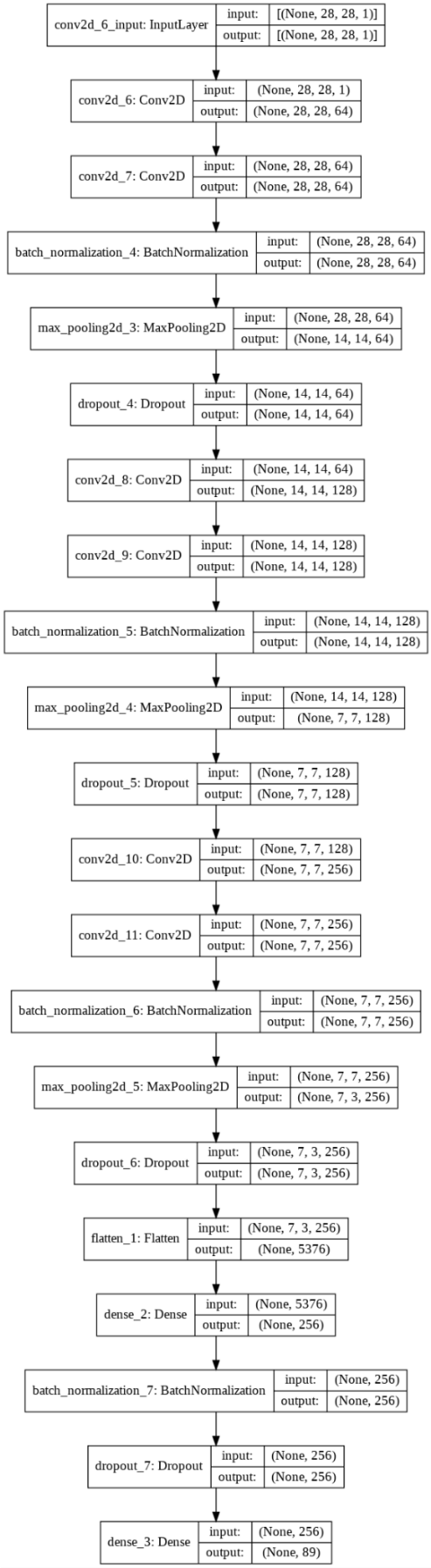
Kết luận:

Vertical Translation và rotation thể hiện tốt nhất.

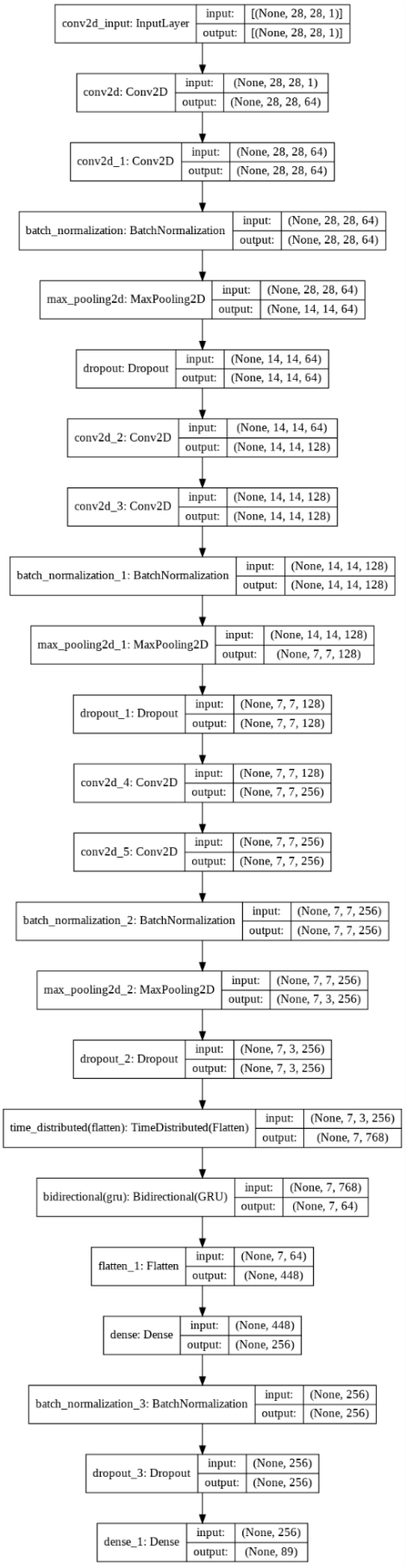
V. BUILDING MODEL

Ở đây chúng ta sẽ xây dựng 2 model dựa trên cấu trúc CNN và CRNN.

1. CNN


conv2d_6_input: InputLayer
input: [(None, 28, 28, 1)]
output: [(None, 28, 28, 1)]
conv2d_6: Conv2D
input: (None, 28, 28, 1)
output: (None, 28, 28, 64)
conv2d_7: Conv2D
input: (None, 28, 28, 64)
output: (None, 28, 28, 64)
batch_normalization_4: BatchNormalization
input: (None, 28, 28, 64)
output: (None, 28, 28, 64)
max_pooling2d_3: MaxPooling2D
input: (None, 28, 28, 64)
output: (None, 14, 14, 64)
dropout_4: Dropout
input: (None, 14, 14, 64)
output: (None, 14, 14, 64)
conv2d_8: Conv2D
input: (None, 14, 14, 64)
output: (None, 14, 14, 128)
conv2d_9: Conv2D
input: (None, 14, 14, 128)
output: (None, 14, 14, 128)
batch_normalization_5: BatchNormalization
input: (None, 14, 14, 128)
output: (None, 14, 14, 128)
max_pooling2d_4: MaxPooling2D
input: (None, 14, 14, 128)
output: (None, 7, 7, 128)
dropout_5: Dropout
input: (None, 7, 7, 128)
output: (None, 7, 7, 128)
conv2d_10: Conv2D
input: (None, 7, 7, 128)
output: (None, 7, 7, 256)
conv2d_11: Conv2D
input: (None, 7, 7, 256)
output: (None, 7, 7, 256)
batch_normalization_6: BatchNormalization
input: (None, 7, 7, 256)
output: (None, 7, 7, 256)
max_pooling2d_5: MaxPooling2D
input: (None, 7, 7, 256)
output: (None, 7, 3, 256)
dropout_6: Dropout
input: (None, 7, 3, 256)
output: (None, 7, 3, 256)
flatten_1: Flatten
input: (None, 7, 3, 256)
output: (None, 5376)
dense_2: Dense
input: (None, 5376)
output: (None, 256)
batch_normalization_7: BatchNormalization
input: (None, 256)
output: (None, 256)
dropout_7: Dropout
input: (None, 256)
output: (None, 256)
dense_3: Dense
input: (None, 256)
output: (None, 89)

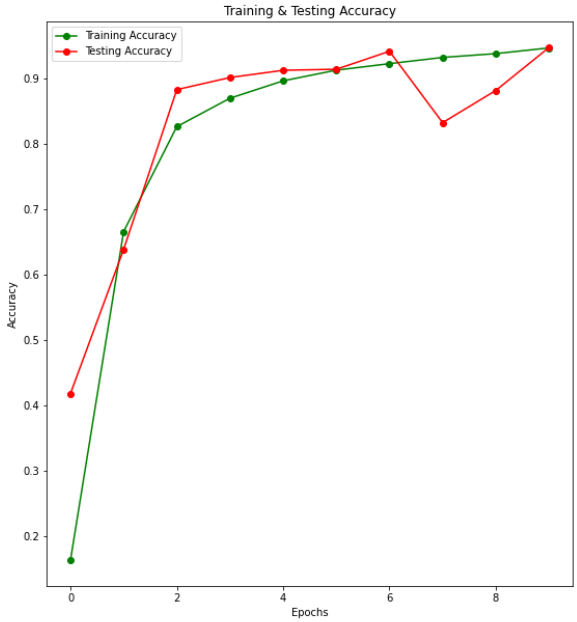
2. CRNN

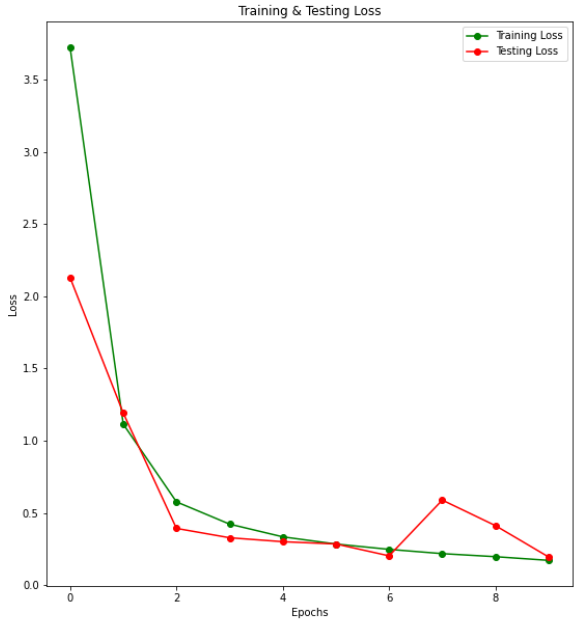

conv2d_input: InputLayer | input: [(None, 28, 28, 1)] | output: [(None, 28, 28, 1)]
conv2d: Conv2D | input: (None, 28, 28, 1) | output: (None, 28, 28, 64)
conv2d_1: Conv2D | input: (None, 28, 28, 64) | output: (None, 28, 28, 64)
batch_normalization: BatchNormalization | input: (None, 28, 28, 64) | output: (None, 28, 28, 64)
max_pooling2d: MaxPooling2D | input: (None, 28, 28, 64) | output: (None, 14, 14, 64)
dropout: Dropout | input: (None, 14, 14, 64) | output: (None, 14, 14, 64)
conv2d_2: Conv2D | input: (None, 14, 14, 64) | output: (None, 14, 14, 128)
conv2d_3: Conv2D | input: (None, 14, 14, 128) | output: (None, 14, 14, 128)
batch_normalization_1: BatchNormalization | input: (None, 14, 14, 128) | output: (None, 14, 14, 128)
max_pooling2d_1: MaxPooling2D | input: (None, 14, 14, 128) | output: (None, 7, 7, 128)
dropout_1: Dropout | input: (None, 7, 7, 128) | output: (None, 7, 7, 128)
conv2d_4: Conv2D | input: (None, 7, 7, 128) | output: (None, 7, 7, 256)
conv2d_5: Conv2D | input: (None, 7, 7, 256) | output: (None, 7, 7, 256)
batch_normalization_2: BatchNormalization | input: (None, 7, 7, 256) | output: (None, 7, 7, 256)
max_pooling2d_2: MaxPooling2D | input: (None, 7, 7, 256) | output: (None, 7, 3, 256)
dropout_2: Dropout | input: (None, 7, 3, 256) | output: (None, 7, 3, 256)
time_distributed(flatten): TimeDistributed(Flatten) | input: (None, 7, 3, 256) | output: (None, 7, 768)
bidirectional(gru): Bidirectional(GRU) | input: (None, 7, 768) | output: (None, 7, 64)
flatten_1: Flatten | input: (None, 7, 64) | output: (None, 448)
dense: Dense | input: (None, 448) | output: (None, 256)
batch_normalization_3: BatchNormalization | input: (None, 256) | output: (None, 256)
dropout_3: Dropout | input: (None, 256) | output: (None, 256)
dense_1: Dense | input: (None, 256) | output: (None, 89)

3. So sánh 2 model

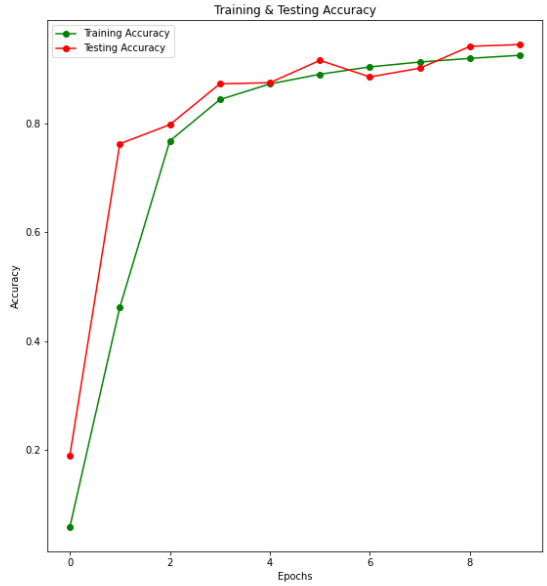
Dataset sẽ chia thành Training set : 37585 và Val set: 12528.

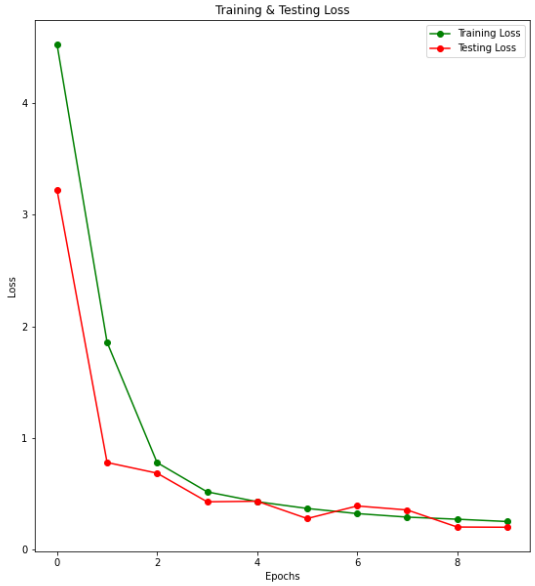
CNN





CRNN





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.

VI. Final run

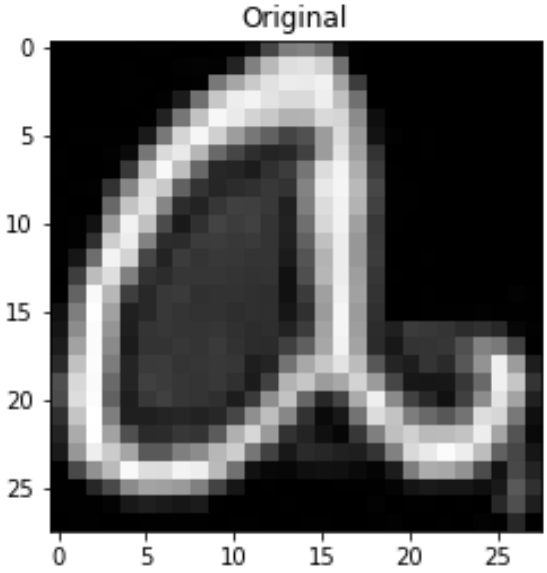
1. Preprocessing

1.1. Data cleaning

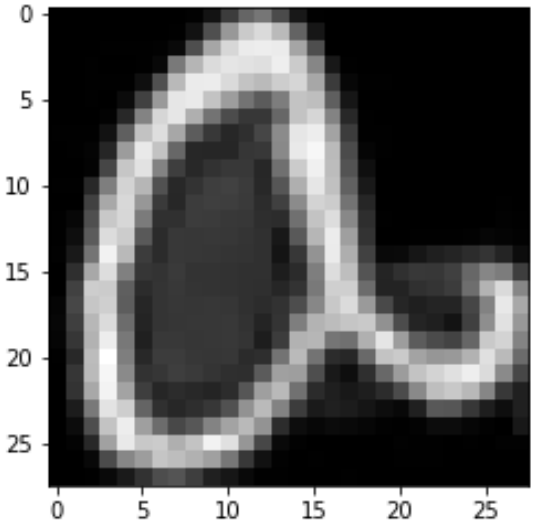
Ảnh sẽ được chuyển từ ảnh ***nền trắng chữ đen*** sang ảnh ***nền đen chữ trắng*** bằng OpenCV bit_wise not sau đó sẽ được Gaussian Blur, cuối cùng resize về kích thước (28,28).

1.2. Data Augmentation

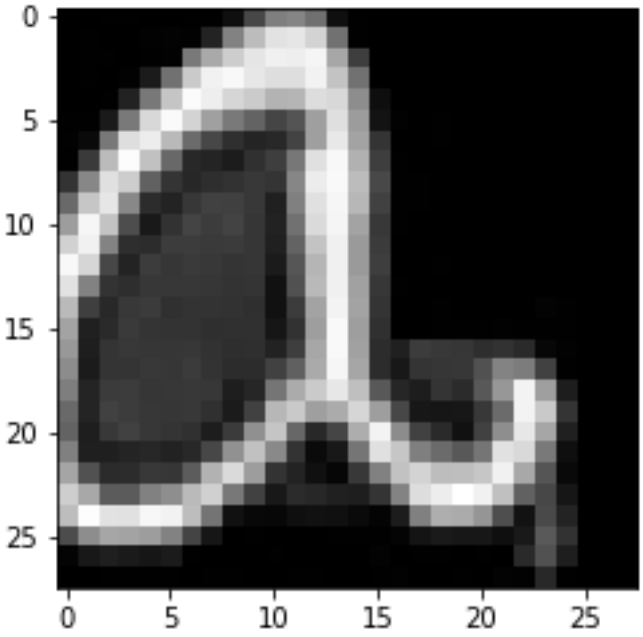
Dataset sẽ thể hiện tốt nhất nếu tăng kích thước dataset lên 4 lần bằng các phương pháp data augmentation. Các phương pháp sử dụng là counter clockwise ten degree rotation, vertical translation về phía trái và vertical translation về phía trái sau đó counter clockwise ten degree rotation.



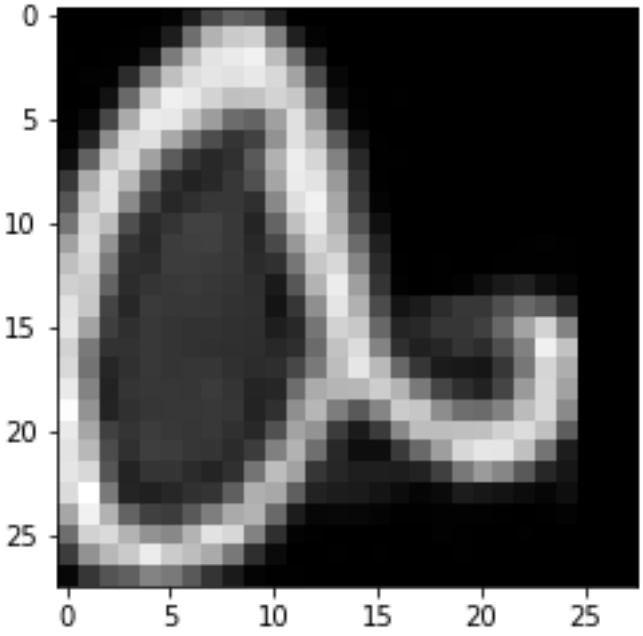
Original



Counter clockwise rotation

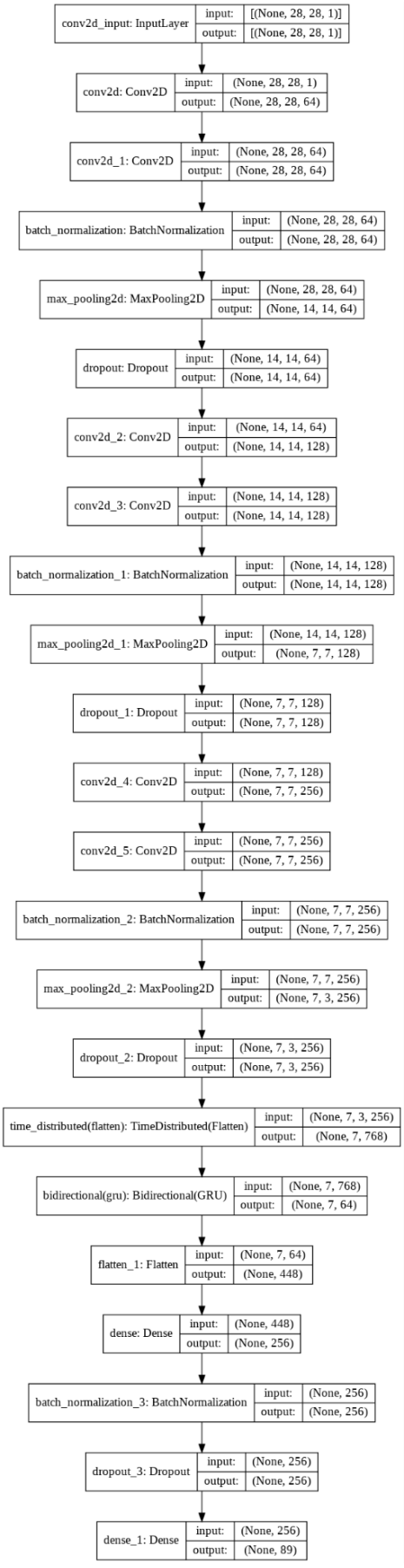


Vertical translation



Vertical translation and counter clockwise rotation.

3. Model sử dụng



3.1 ReLu activation function

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

3.2. Dropout

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

3.3. Optimizer

Nhóm em sẽ sử dụng adam. Adam thể hiện tốt hơn Stochastic gradient descent.

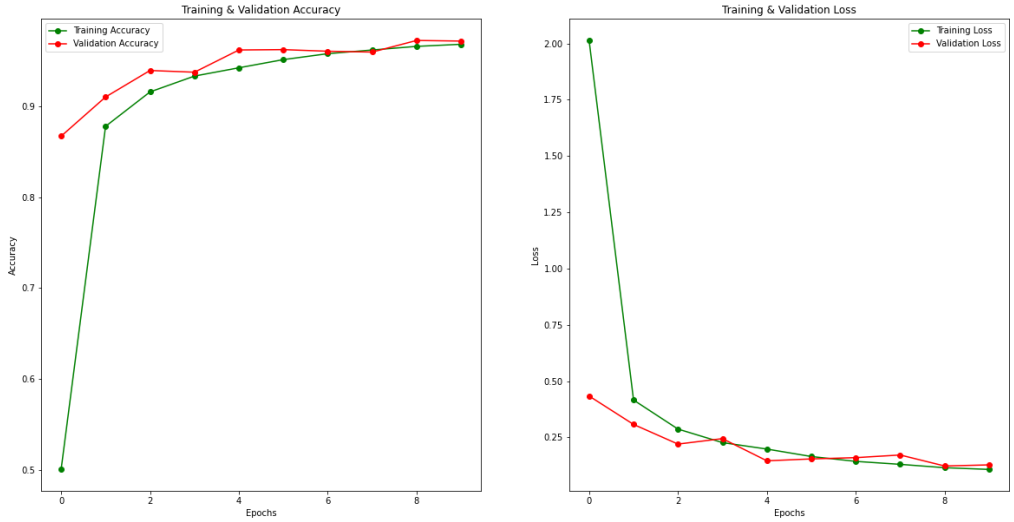
3.4. Loss funtion

Loss function là categorical-crossentropy cho multi-label classification.

4. Kết quả cuối cùng (test result):

Training process:

Network sẽ train trong epoch 10, mỗi epoch có 1879 step với batch size là 64.



Kết quả tập Training set: 0.9681

Kết quả tập Val set: 0.9716

TEST RESULT:

accuracy			0.97	10023
macro avg	0.97	0.97	0.97	10023
weighted avg	0.97	0.97	0.97	10023

Kết quả tập test set: 0.9720

5. Nhận xét

Model có độ chính xác cao. Accuracy đối với test set là 0,97. Model fit tốt với 0,9681 đối với Training set; 0,9716 với Val set và 0,9720 với Test set.

Tuy vậy vẫn có sự lẫn lộn giữa những class có độ tương đồng cao như ‘ê’ với ‘ể’ hay ‘ề’; ‘o’ và ‘ỏ’ hay ‘ơ’ v.v…

Nguyên nhân:

- Dataset chưa thật sự nhiều và không cân bằng giữa các class. Có class trên 1500 ảnh nhưng có class chỉ dưới 400 ảnh. Và những class có tỷ lệ dự đoán thiếu chính xác nhất rơi vào chính những class có ít ảnh ấy.
- Dataset có chất lượng không cân bằng. Có ảnh nhìn rõ, có ảnh thì mờ.
- Xử lý ảnh vẫn chưa tốt. Nếu quá cố loại bỏ nhiễu ra khỏi ảnh thì nhiều ảnh thì sẽ mất hoàn toàn, còn không thì một số ảnh sẽ có rất nhiều nhiễu số.

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 thể 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.

VII. 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] 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