NHẬN DẠNG CHỮ VIẾT TAY TIẾNG VIỆT

MÁY HỌC – MACHINE LEARNING

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

NỘI DUNG

I. ABSTRACT
II. MÔ TẢ BÀI TOÁN
III. MÔ TẢ DATASET
IV. DATASET PROCESSING
V. BUILDING MODEL
VI. FINAL RUN VÀ NHẬN XÉT

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).

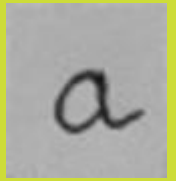
II.

MÔ TẢ BÀI TOÁN

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.

Input:



Output: **a**

III.

MÔ TẢ DATASET

Bảng chữ cái Việt Nam có 29 chữ cái, bao gồm:

- 22 chữ cái latin
- 7 chữ cái biến thể bằng cách thê dấu

Ngoài ra, 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 có 89 loại chữ cần thu thập.

Nhóm em sẽ thu nhập chữ viết từ người thân và bạn bè. Ngoài ra, nhóm em sẽ share dataset với nhóm của bạn **Nguyễn Dương Hải**.

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.

- Code cắt ảnh nhóm em được sự giúp đỡ từ nhóm của bạn Nguyễn Dương Hải, bao gồm các phần chính như sau:

  - **Sử dụng cv2 edge detection để cắt gọn những khoảng trắng dư thừa:**

```
def Cut(image):
  gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
  thresh = cv2.adaptiveThreshold(gray,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY_INV,57,5)
  contours = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)[0]
  max = -1
  L = []
  for cnt in contours:
      x, y, w, h = cv2.boundingRect(cnt)
      if cv2.contourArea(cnt) > max:
          x_max, y_max, w_max, h_max = x, y, w, h
          max = cv2.contourArea(cnt)
  table = image[y_max:y_max+h_max, x_max:x_max+w_max]
  return table
```

- Code cắt ảnh nhóm em được sự giúp đỡ từ nhóm của bạn Nguyễn Dương Hải, bao gồm các phần chính như sau:

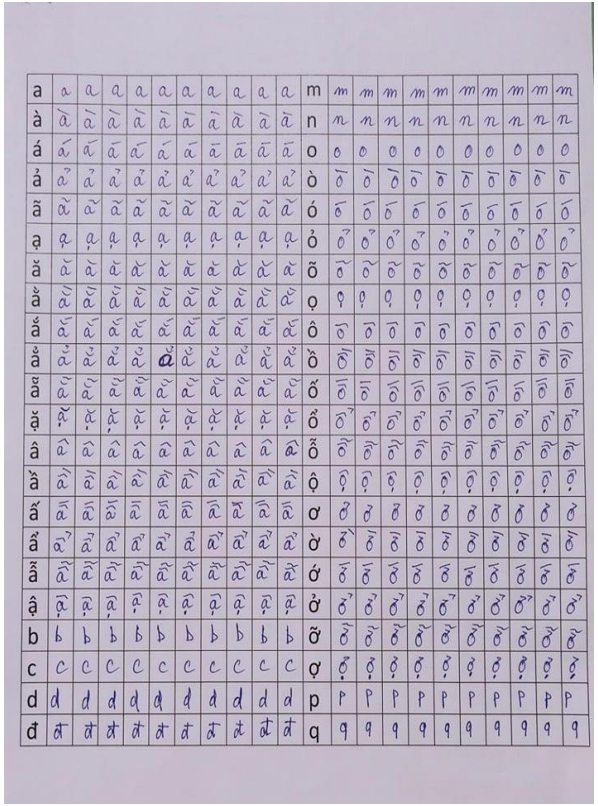
  - **Lọc ô chữ sau khi cắt gọn:**

```
cnts = cv2.findContours(thresh, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
cnts = cnts[0] if len(cnts) == 2 else cnts[1]
for c in cnts:
    area = cv2.contourArea(c)
    if area < 1000:
        cv2.drawContours(thresh, [c], -1, (0,0,0), -1)
# Fix horizontal and vertical lines
# Xoá các yếu tố gây nhiễu
vertical_kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (1,5))
thresh = cv2.morphologyEx(thresh, cv2.MORPH_CLOSE, vertical_kernel, iterations=9)
horizontal_kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (5,1))
thresh = cv2.morphologyEx(thresh, cv2.MORPH_CLOSE, horizontal_kernel, iterations=4)

# Sort by top to bottom and each row by left to right
invert = 255 - thresh
cnts = cv2.findContours(invert, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
cnts = cnts[0] if len(cnts) == 2 else cnts[1]
(cnts, _) = contours.sort_contours(cnts, method="top-to-bottom")

data_rows = []
row = []
for (i, c) in enumerate(cnts, 1):
    area = cv2.contourArea(c)
    if area < 50000:
        row.append(c)
        if i % 9 == 0:
            (cnts, _) = contours.sort_contours(row, method="left-to-right")
            data_rows.append(cnts)
            row = []
```

- Code cắt ảnh nhóm em được sự giúp đỡ từ nhóm của bạn Nguyễn Dương Hải, bao gồm các phần chính như sau:

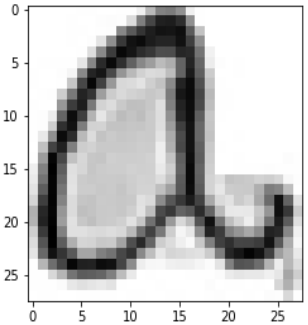
  - **Duyệt từng ô chữ và lưu ảnh đã cắt vào Drive:**

```
# Iterate through each box
count = 75000
for row in data_rows:
  for c in row:
    mask = np.zeros(image.shape, dtype=np.uint8)
    cv2.drawContours(mask, [c], -1, (255,255,255), -1)
    result = cv2.bitwise_and(image, mask)
    result[mask==0] = 255
    img_result = result
    try:
      final = Cut(img_result)
    #fin_gray = cv2.cvtColor(final, cv2.COLOR_BGR2GRAY)
    #fin_thresh = cv2.adaptiveThreshold(fin_gray,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY_INV,57,5)
    #kernel = np.ones((3,3),np.uint8)
    #opening = cv2.morphologyEx(fin_thresh, cv2.MORPH_OPEN, kernel)
      final = cv2.cvtColor(final, cv2.COLOR_BGR2GRAY)
      final = cv2.cvtColor(final, cv2.COLOR_GRAY2RGB)
      cv2.imwrite('/content/gdrive/MyDrive/CutImage/75/image_' + str(count) + '.JPG' , final)
      count += 1
    except:
      continue
    '''if count == 1:
       plt.imshow(final)
       cv2.waitKey(175)
       break
  break'''
```



Mẫu dataset trước khi cắt

Mẫu dataset sau khi cắt

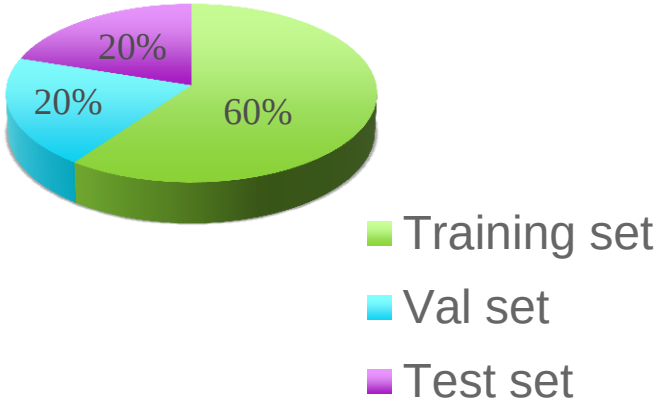


Mẫu dataset đã được zoom và resize

Bộ dataset chữ cái viết tay tiếng Việt sau khi thu thập có tổng cộng 50114 hình ảnh đã được phân loại theo từng nhãn riêng biệt. Bao gồm:

Dataset sử dụng cho bước Final run sẽ được chia làm ba set theo tỉ lệ xấp xỉ 3:1:1 như sau:

- 30068 ảnh cho Training set
- 10023 ảnh cho Val set
- 10023 ảnh cho Test set



a - 1356
à - 513
ả - 543
ã - 472
á - 551
ạ - 615
ă - 667
ằ - 468
ắ - 443
ẵ - 449
ẳ - 470
ặ - 474
â - 607
ầ - 479
ấ - 456
ẫ - 410
ẩ - 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
ế - 430
ễ - 368
ể - 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
ố - 387
ỗ - 398
ổ - 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

IV.

DATA PREPROCESSING

- DATA CLEANING
  - Denoise
    - Gaussian Blurring
    - Average
    - Median Blurring
    - Bilateral Filtering
  - Segmentation & Morphological
    - Erosion, dilation, opening & closing
    - Threshold (Simple & Adaptive)
- DATA AUGMENTATION
  - Addition of noise
  - Rotation
  - Translation
  - Shearing

DATA CLEANING

❖ **Denoise**

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

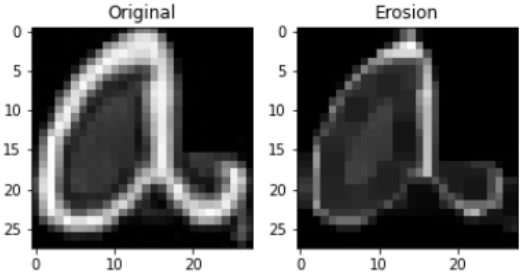
	VAL-ACC	VAL-LOSS
RAW	0,8902	0,3456
AVERAGE	0,8076	0,5825
GAUSSIAN	0,9073	0,2997
MEDIAN	0,8579	0,4559
BILATERAL	0,8902	0,3481

Kết luận:

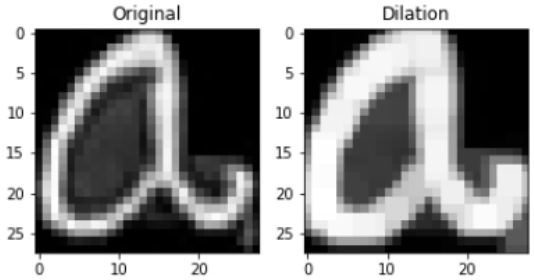
Với cả dataset nhỏ và dataset lớn, Gaussian blur hoặc không xử lý gì hết cho ra kết quả tốt nhất.

DATA CLEANING

- **Segmentation & Morphological**



Ảnh trước và sau Erode



Ảnh trước và sau Dilattion

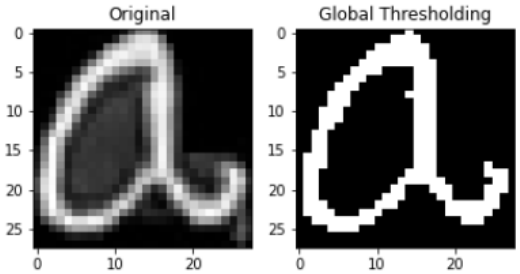
Ảnh trước và sau Opening

Ảnh trước và sau Closing

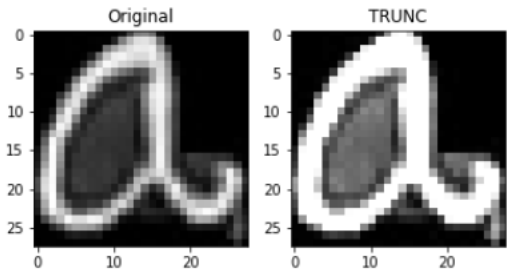
DATA CLEANING

Simple Threshold

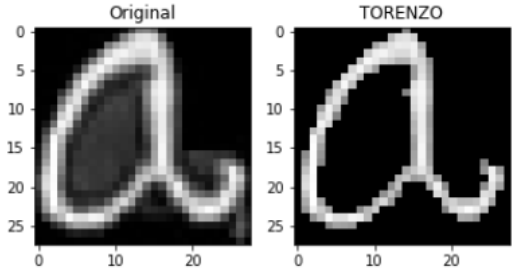
❖ **Segmentation & Morphological**



Ảnh trước và sau Binary



Ảnh trước và sau Trunc

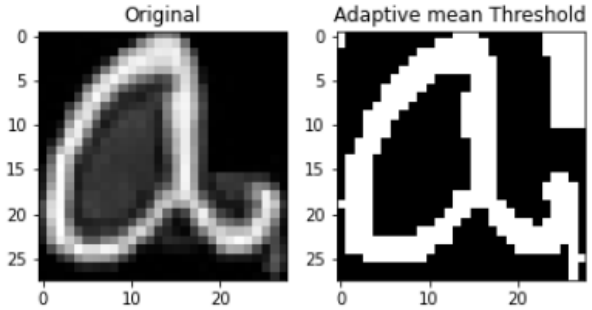


Ảnh trước và sau Torenzo

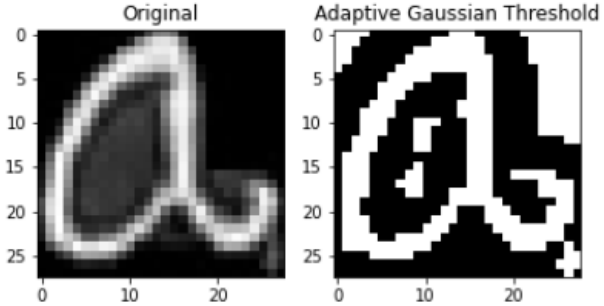
DATA CLEANING

Adaptive Threshold

- **Segmentation & Morphological**



Adaptive Mean threshold



Adaptive Gaussian threshold

DATA CLEANING

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

Không Gaussian blur, simple threshold:

	VAL-ACC	VAL-LOSS
BINARY	0,7238	0,9818
TRUNC	0,0268	4,4356
TOZERO	0,8716	0,4417

Có Gaussian blur, adaptive threshold:

	VAL-ACC	VAL-LOSS
THRESH-MEAN	0,8373	0,5317
THRESH-GAUSSIAN	0,8506	0,4814

Không Gaussian blur, adaptive threshold:

	VAL-ACC	VAL-LOSS
THRESH-MEAN	0,6623	1,4653
THRESH-GAUSSIAN	0,6854	1,3452

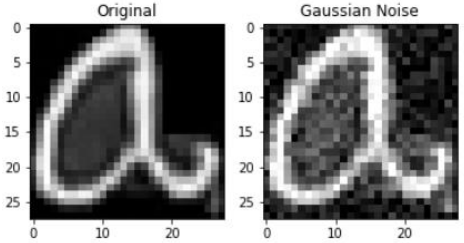
So với raw dataset:

VAL-ACC	VAL-LOSS
0,8911	0,3123

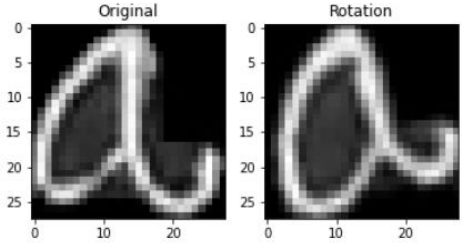
Với các dataset nhỏ khoảng 5000 ảnh xử lý segmentation & morphological làm tăng độ chính xác. Nhưng với các dataset lớn hơn thì ngược lại. => **Không** sử dụng các phương pháp segmentation & morphological.

DATA AUGMENTATION

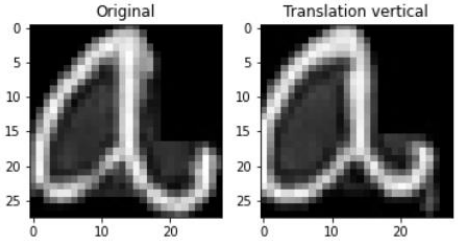
Data augmentation là phương thức tạo ra dataset mới từ dataset đã có. Ví dụ một dataset mới đã được tạo ra bằng cách quay dataset cũ một góc 10 độ cùng chiều kim đồng hồ. Kết hợp cả 2 dataset ta có 1 dataset mới.



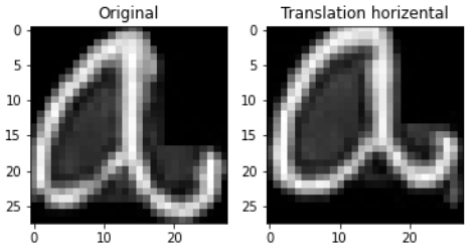
Addition of noise



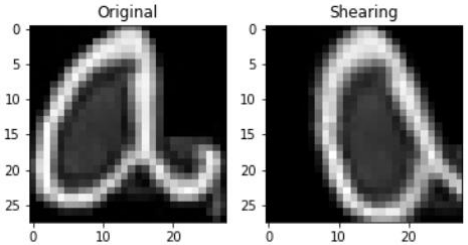
Rotation



Vertical translation



Horizontal translation



Shearing

DATA AUGMENTATION

	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.

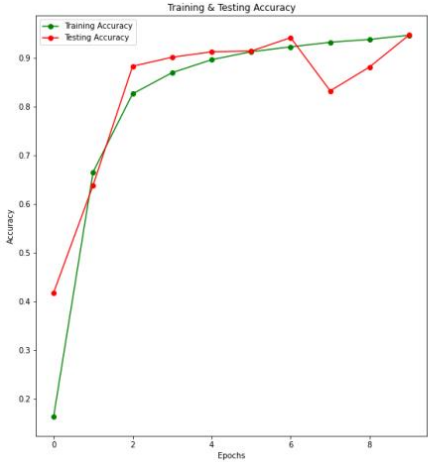
BUILDING MODEL

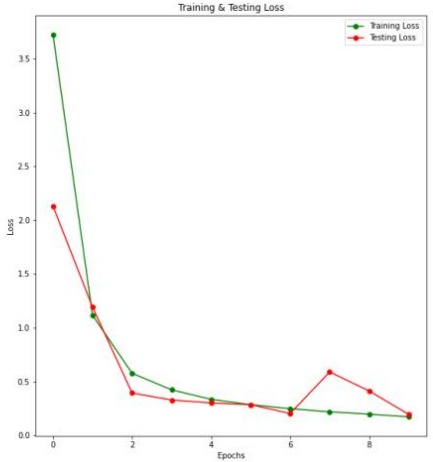
Ở đây chúng ta sẽ xây dựng 2 model dựa trên cấu trúc CNN và CRNN.

Ta sẽ so sánh hai model trên bằng cách chạy thử chúng trên tập dataset được chia như sau:

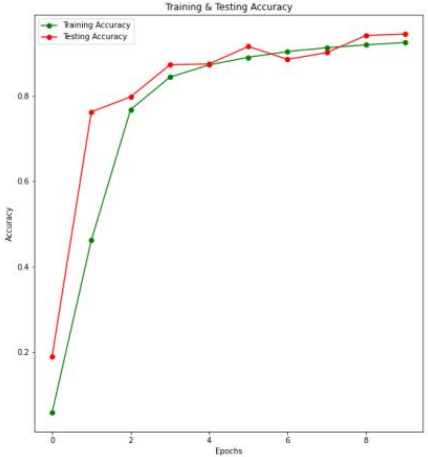
- Training set: 37585 ảnh
- Val set: 12528 ảnh

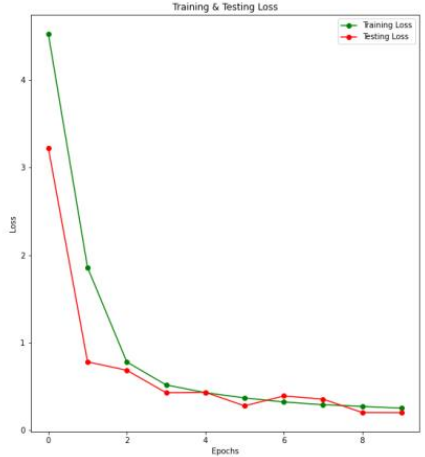
CNN





CRNN





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 nên nhóm em sẽ chọn CRNN cho bước Final run.

VI.

FINAL RUN VÀ NHẬN XÉT

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).

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 hoặc kết hợp vertical translation về phía trái sau đó counter clockwise ten degree rotation.

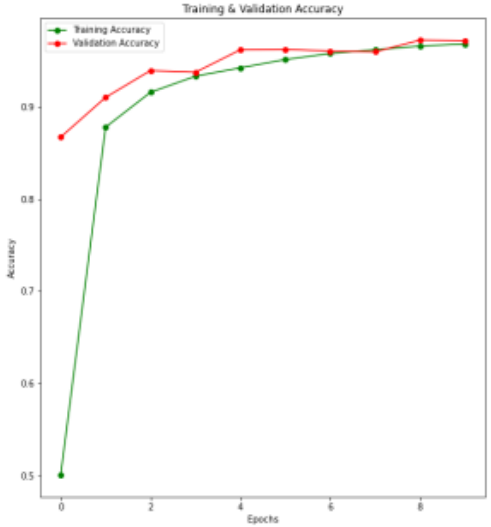
Ngoài ra:

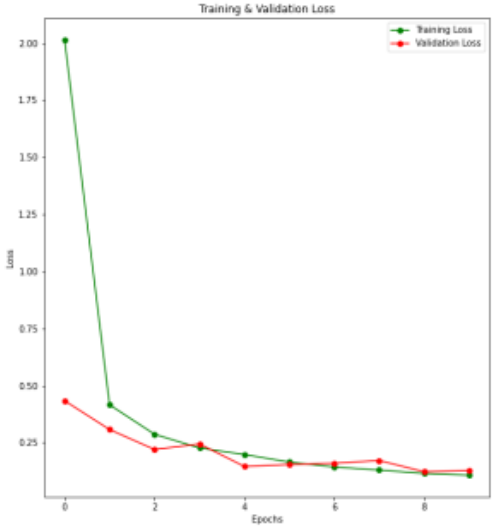
- ReLU (Rectified Linear Unit) function là 1 non-linear function. ReLU là 1 function phổ biến.
- Dropout giúp cho model không bị overfitting. Nó giúp model trong lúc train tránh bị hiện tượng overfitting.
- Nhóm em sẽ sử dụng adam. Adam thể hiện tốt hơn Stochastic gradient descent.
- Loss function là categorical-crossentropy cho multi-label classification.

TEST RESULT

Training process:

Network sẽ train trong 10 epoch, 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

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: 0.972

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 'ề' 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.

THANKS!

Any questions?

You can find us at
https://github.com/noeffortnomoney/CS114.L22.KHCL