

DẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH



**ĐỒ ÁN TỔNG HỢP
HƯỚNG TRÍ TUỆ NHÂN TẠO
NHẬN DIỆN BIỂN SỐ XE VỚI OCR**

Nhóm: 4
Giảng viên: Trần Huy
Vũ Văn Tiến
Sinh viên thực hiện: Nguyễn Dắc Hoàng Phú 2010514
Phạm Hữu Phú 2010516
Trà Trung Tín 2010702
Phương Chung Tú 2010767

Hồ Chí Minh, 12/2022



Contents

1 Giới thiệu bài toán	2
1.1 Đặt vấn đề	2
1.2 Mục tiêu đề tài	2
2 Kiến thức tìm hiểu	3
2.1 Cơ sở lý thuyết	3
2.1.1 Neural network	3
2.1.2 Convolutional Neural Network (CNN)	4
2.1.3 Recurrent Neural Network (RNN)	7
2.2 Các nghiên cứu liên quan	8
2.3 Công nghệ sử dụng	8
3 Hướng tiếp cận	9
4 Hiện thực	11
4.1 Mô tả dữ liệu	11
4.1.1 Dữ liệu ký tự	11
4.1.2 Dữ liệu biển số xe	11
4.2 Tiền xử lý dữ liệu	11
4.2.1 Dữ liệu ký tự	11
4.2.2 Dữ liệu biển số xe	12
4.3 Xây dựng mô hình	13
4.3.1 Mô hình End2End	13
4.4 Hiện thực mô hình	15
4.4.1 Mô hình Classification	15
4.4.2 Mô hình End2End	18
4.5 Kết quả mô hình và đánh giá	20
5 Kết luận	25



1 Giới thiệu bài toán

1.1 Đặt vấn đề

Hiện nay với sự phát triển của kinh tế và xã hội, bên cạnh những mặt tích cực, nước ta vẫn còn phải đương đầu với các vấn đề phức tạp do sự phát triển gây ra. Một trong số những vấn đề đó là tình trạng giao thông quá tải và hỗn loạn thường xuyên xảy ra ở những trung tâm kinh tế lớn của cả nước. Thiệt hại từ các vấn đề giao thông gây ra là rất lớn cho nền kinh tế cũng như xã hội. Bên cạnh việc nâng cao ý thức chấp hành giao thông của người dân, việc giám sát quản lý giao thông cũng cần được đặt ưu tiên hàng đầu. Tuy nhiên, ở nước ta từ trước tới nay việc giám sát giao thông vẫn dùng phương pháp thủ công là chủ yếu. Việc giám sát thủ công nhìn chung rất khó khăn vì số lượng phương tiện giao thông lớn, đòi hỏi phải có cách tiếp cận một cách tự động để giảm nhẹ sức lao động của con người. Đây là hướng đi đã có từ lâu ở những nước phát triển. Hiện nay ở nước ta trước yêu cầu nắm bắt công nghệ trong thời đại cách mạng công nghiệp 4.0 để ứng dụng cho đời sống sản xuất, việc giải quyết bài toán giám sát giao thông tự động là vô cùng cấp thiết.

Bài toán giám sát giao thông tự động bao gồm nhiều bài toán con, một trong số đó là bài toán nhận diện biển số xe giúp quản lý phương tiện giao thông. Với các lý do nêu trên, đề tài hình thành với mục tiêu giải quyết tốt bài toán nhận diện biển số xe để góp phần giải quyết bài toán giao thông chung hiện nay. Vấn đề phát hiện và nhận diện biển số xe là một trong những hướng nghiên cứu đã có từ lâu trong lĩnh vực thị giác máy tính. Tuy nhiên, hiện nay với sự phát triển mạnh mẽ của phương pháp học sâu (deep learning) đã đem lại hướng tiếp cận mới cho vấn đề này. Có thể kể tới các mạng nơ-ron trong học sâu phổ biến hiện nay như Convolution Neural Networks (CNN), Recurrent Neural Networks (RNN) cùng các kiến trúc mạng như LeNet[7], ImageNet [8], Fast R-CNN [9], Đây là hướng nghiên cứu khá mới nhưng lại đạt được những kết quả hết sức ấn tượng qua các cuộc thi trong giới học thuật cũng như những ứng dụng thực tế. Bên cạnh những điểm tích cực nêu trên, vấn đề nhận diện biển số xe cũng có một số khó khăn thách thức bao gồm các yếu tố do môi trường, các yếu tố do quá trình thu nhận hình ảnh và các yếu tố liên quan tới quá trình huấn luyện mạng học sâu:

- Các yếu tố do môi trường có thể kể đến như độ phức tạp của khung cảnh có chứa biển số xe, gây khó khăn cho việc phân biệt biển số xe với các đối tượng khác. Ngoài ra vấn đề độ sáng của môi trường, độ nhạy sáng của các thiết bị cảm biến cũng ảnh hưởng rất lớn đến kết quả của quá trình phát hiện và nhận dạng.
- Các yếu tố do quá trình thu nhận hình ảnh gồm có độ mờ, chất lượng ảnh thấp và độ biến dạng của ảnh. Vấn đề độ mờ xuất phát chủ yếu từ việc cản chỉnh tiêu cự của máy quay và sự chuyển động của máy quay hoặc vật thể gây ra. Hình ảnh hay video trải qua quá trình nén và giải nén cũng sẽ dẫn đến giảm chất lượng hình ảnh. Cuối cùng với nhiều góc độ của máy quay, hình ảnh thu được sẽ bị biến dạng.
- Quá trình huấn luyện mạng deep learning cần nhiều dữ liệu để đảm bảo được tính tổng quát cao cũng như độ chính xác tốt. Tuy nhiên việc thu thập và gán nhãn số lượng lớn dữ liệu là công việc mất rất nhiều thời gian và công sức.
- Việc huấn luyện cũng tốn nhiều thời gian. Kích thước mạng càng lớn và phức tạp thì thời gian cũng tăng theo. Hơn nữa để tìm được bộ siêu tham số phù hợp với kiến trúc mạng, với mỗi một tham số truyền vào ta phải thực hiện việc huấn luyện lại mạng.

Với những khó khăn kể trên, việc cải tiến, kết hợp các phương pháp hiện tại để cho ra đời một phương pháp hiệu quả giúp giải quyết các vấn đề trên là hoàn toàn cấp thiết. Bên cạnh đó, đề tài cũng mong muốn có thể giải quyết được các vấn đề còn tồn tại của các đề tài đi trước.

1.2 Mục tiêu đề tài

Mục tiêu của đề tài là đề xuất một phương pháp phát hiện và nhận dạng biển số xe Việt Nam hiệu quả. Phương pháp chủ yếu được sử dụng là học sâu. Yêu cầu cần đạt được bao gồm độ chính xác cho việc phát hiện biển số là trên 90% và độ chính xác cho việc nhận diện chuỗi biển số là trên 80%. Ngoài ra đề tài cũng chú trọng đến thời gian thực thi của mô hình đề xuất.

Phương pháp mà đề tài đề xuất có thể được sử dụng trong các ứng dụng như giám sát giao thông tự động, bãi giữ xe thông minh, trạm thu phí tự động ... nhằm giảm bớt thời gian, công sức của con người, giảm tình trạng kẹt xe và tăng sự tiện lợi cho người tham gia giao thông. Xây dựng được tập dữ liệu chính xác, đa dạng và sát với điều kiện thực tế nước ta.

2 Kiến thức tìm hiểu

2.1 Cơ sở lý thuyết

2.1.1 Neural network



Figure 1: Neural

- **Neural network là gì?**

Neural Network đọc tiếng việt là Mạng nơ-ron nhân tạo, đây là một chuỗi những thuật toán được đưa ra để tìm kiếm các mối quan hệ cơ bản trong tập hợp các dữ liệu. Thông qua việc bắt bước cách thức hoạt động từ não bộ con người. Nói cách khác, mạng nơ-ron nhân tạo được xem là hệ thống của các tế bào thần kinh nhân tạo. Đây thường có thể là hữu cơ hoặc nhân tạo về bản chất.

- **Đặc điểm của Artificial Neural Network**

1. Trong lĩnh vực tài chính, mạng nơ-ron nhân tạo hỗ trợ cho quá trình phát triển các quy trình như: giao dịch thuật toán, dự báo chuỗi thời gian, phân loại chứng khoán, mô hình rủi ro tín dụng và xây dựng chỉ báo độc quyền và công cụ phát sinh giá cả. Mạng nơ-ron nhân tạo có thể hoạt động như mạng nơ-ron của con người. Mỗi một nơ-ron thần kinh trong nơ-ron nhân tạo là hàm toán học với chức năng thu thập và phân loại các thông tin dựa theo cấu trúc cụ thể.
2. Neural Network có sự tương đồng mạnh mẽ với những phương pháp thống kê như đồ thị đường cong và phân tích hồi quy. Neural Network có chứa những lớp bao hàm các nút được liên kết lại với nhau. Mỗi nút lại là một tri giác có cấu tạo tương tự với hàm hồi quy đa tuyến tính. Bên trong một lớp tri giác đa lớp, chúng sẽ được sắp xếp dựa theo các lớp liên kết với nhau. Lớp đầu vào sẽ thu thập các mẫu đầu vào và lớp đầu ra sẽ nhận các phân loại hoặc tín hiệu đầu ra mà các mẫu đầu vào có thể phản ánh lại.

- **Kiến trúc Neural Network**

Mạng Neural Network là sự kết hợp của những tầng perceptron hay còn gọi là perceptron đa tầng. Và mỗi một mạng Neural Network thường bao gồm 3 kiểu tầng là:

1. **Tầng input layer (tầng vào):** Tầng này nằm bên trái cùng của mạng, thể hiện cho các đầu vào của mạng.
2. **Tầng output layer (tầng ra):** Là tầng bên phải cùng và nó thể hiện cho những đầu ra của mạng.
3. **Tầng hidden layer (tầng ẩn):** Tầng này nằm giữa tầng vào và tầng ra nó thể hiện cho quá trình suy luận logic của mạng. Lưu ý: Mỗi một Neural Network chỉ có duy nhất một tầng vào và 1 tầng ra nhưng lại có rất nhiều tầng ẩn.

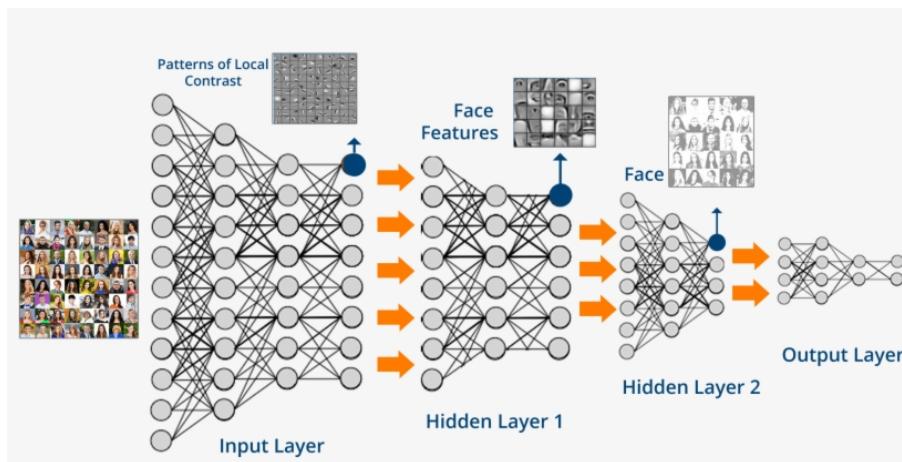


Figure 2: Kiến trúc Neural Network

Với mạng Neural Network thì mỗi nút mạng là một sigmoid neuron nhưng chúng lại có hàm kích hoạt khác nhau. Thực tế, người ta thường sử dụng có cùng loại với nhau để việc tính toán thuận lợi hơn. Tại mỗi tầng, số lượng nút mạng có thể khác nhau còn tùy vào bài toán hoặc cách giải quyết.

- **Lan truyền trong Neural Network**

Trong toàn bộ các nốt mạng neuron đều có thể kết hợp đôi một với nhau theo một chiều duy nhất từ tầng vào đến tầng ra. Có nghĩa là, mỗi nốt ở một tầng sẽ nhận đầu vào là tất cả các nốt ở tầng trước đó và ngược lại. Có nghĩa là, việc suy luận Neural Network là dạng suy luận tiến (feedforward).

- **Sử dụng Neural network**

Mạng neural nhân tạo có khả năng sử dụng được như một loại cơ chế xấp xỉ hàm tùy ý mà học được từ việc dữ liệu quan sát. Tuy nhiên, việc sử dụng chúng khá khó và cần phải có sự hiểu biết tương đối về những lý thuyết cơ bản về mạng neuron này.

1. **Lựa chọn mô hình:** Phụ thuộc vào cách trình bày dữ liệu và các ứng dụng của nó. Đây là mô hình khá phức tạp nên có thể dẫn đến nhiều thách thức cho quá trình học.
2. **Thuật toán học:** Thường sẽ có rất nhiều thỏa thuận giữa các thuật toán học. Và hầu hết, chúng sẽ làm việc tốt với những tham số đúng nhằm huấn luyện trên dữ liệu mà không nhìn thấy yêu cầu một số lượng đáng kể các thử nghiệm.
3. **Mạnh mẽ:** Nếu như các mô hình, thuật toán học và hàm chi phí được lựa chọn một cách thích hợp thì Neural Network có thể cho ra kết quả vô cùng hợp lý. Lưu ý: Mỗi một Neural Network chỉ có duy nhất một tầng vào và 1 tầng ra nhưng lại có rất nhiều tầng ẩn.

2.1.2 Convolutional Neural Network (CNN)

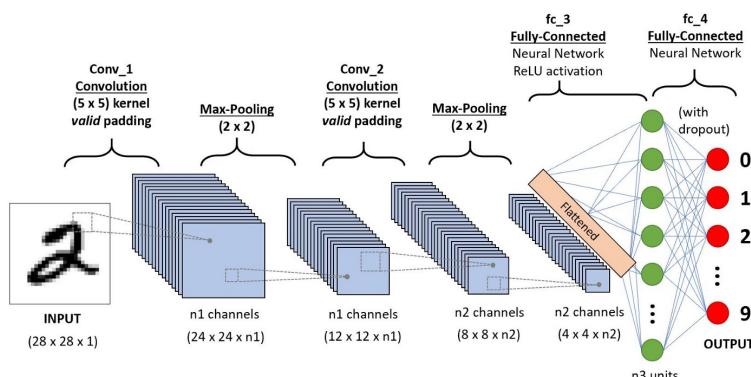


Figure 3: CNN

• Tổng quan

- Trong mạng neural, mô hình mạng neural tích chập (CNN) là 1 trong những mô hình để nhận dạng và phân loại hình ảnh. Trong đó, xác định đối tượng và nhận dạng khuôn mặt là 1 trong số những lĩnh vực mà CNN được sử dụng rộng rãi.
- CNN phân loại hình ảnh bằng cách lấy 1 hình ảnh đầu vào, xử lý và phân loại nó theo các hạng mục nhất định (Ví dụ: Chó, Mèo, Hổ, ...). Máy tính coi hình ảnh đầu vào là 1 mảng pixel và nó phụ thuộc vào độ phân giải của hình ảnh. Dựa trên độ phân giải hình ảnh, máy tính sẽ thấy $H \times W \times D$ (H : Chiều cao, W : Chiều rộng, D : Độ dày).

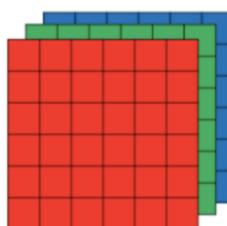


Figure 4: Size 6x6x3

- Về kỹ thuật, mô hình CNN để training và kiểm tra, mỗi hình ảnh đầu vào sẽ chuyển nó qua 1 loạt các lớp tích chập với các bộ lọc (Kernels), tổng hợp lại các lớp được kết nối đầy đủ (Full Connected) và áp dụng hàm Softmax để phân loại đối tượng có giá trị xác suất giữa 0 và 1. Hình dưới đây là toàn bộ luồng CNN để xử lý hình ảnh đầu vào và phân loại các đối tượng dựa trên giá trị.

• Lớp tích chập - Convolution Layer

- Tích chập là lớp đầu tiên để trích xuất các tính năng từ hình ảnh đầu vào. Tích chập duy trì mối quan hệ giữa các pixel bằng cách tìm hiểu các tính năng hình ảnh bằng cách sử dụng các ô vuông nhỏ của dữ liệu đầu vào. Nó là 1 phép toán có 2 đầu vào như ma trận hình ảnh và 1 bộ lọc hoặc hạt nhân.

- An image matrix (volume) of dimension $(h \times w \times d)$
- A filter $(f_h \times f_w \times d)$
- Outputs a volume dimension $(h - f_h + 1) \times (w - f_w + 1) \times 1$

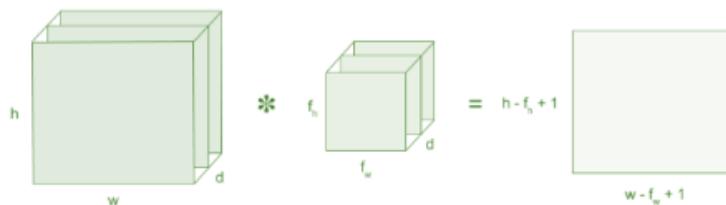


Figure 5: Matrix

- Xem xét 1 ma trận 5×5 có giá trị pixel là 0 và 1. Ma trận bộ lọc 3×3 như hình bên dưới.

$$5 \times 5 - \text{Image Matrix}$$
$$3 \times 3 - \text{Filter Matrix}$$

Figure 6: Feature map

3. Sau đó, lớp tích chập của ma trận hình ảnh 5×5 nhân với ma trận bộ lọc 3×3 gọi là 'Feature Map' như hình bên dưới.

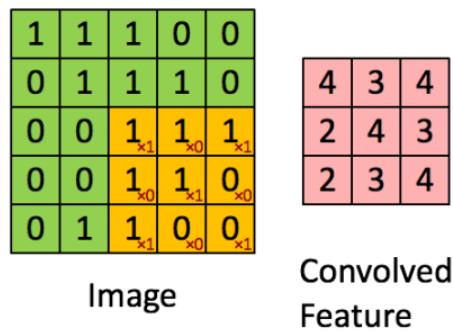


Figure 7: Feature map

4. Sự kết hợp của 1 hình ảnh với các bộ lọc khác nhau có thể thực hiện các hoạt động như phát hiện cạnh, làm mờ và làm sắc nét bằng cách áp dụng các bộ lọc. Ví dụ dưới đây cho thấy hình ảnh tích chập khác nhau sau khi áp dụng các Kernel khác nhau.

Operation	Kernel ω	Image result $g(x,y)$
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Box blur	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Gaussian blur 3×3	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	
Gaussian blur 5×5	$\frac{1}{256} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	

Figure 8: Example

- **Lớp gộp - Pooling Layer**

Lớp pooling sẽ giảm bớt số lượng tham số khi hình ảnh quá lớn. Không gian pooling còn được gọi là lấy mẫu con hoặc lấy mẫu xuống làm giảm kích thước của mỗi map nhưng vẫn giữ lại thông tin quan trọng. Các pooling có thể có nhiều loại khác nhau:

1. **Max Pooling:** lấy phần tử lớn nhất từ ma trận đối tượng
2. **Average Pooling:** lấy tổng trung bình phần tử từ ma trận đối tượng
3. **Sum Pooling :** Tổng tất cả các phần tử trong map

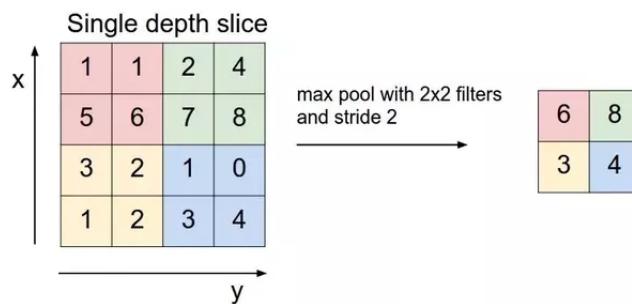


Figure 9: Max pooling

- **Tổng kết**

1. Đầu vào của lớp tích chập là hình ảnh
2. Chọn đối số, áp dụng các bộ lọc với các bước nhảy, padding nếu cần. Thực hiện tích chập cho hình ảnh và áp dụng hàm kích hoạt ReLU cho ma trận hình ảnh.
3. Thực hiện Pooling để giảm kích thước cho hình ảnh.
4. Thêm nhiều lớp tích chập sao cho phù hợp
5. Xây dựng đầu ra và dữ liệu đầu vào thành 1 lớp được kết nối đầy đủ (Full Connected)
6. Sử dụng hàm kích hoạt để tìm đối số phù hợp và phân loại hình ảnh.

2.1.3 Recurrent Neural Network (RNN)

1. Mạng nơ-ron hồi quy (RNN - Recurrent Neural Network) là một thuật toán được chú ý rất nhiều trong thời gian gần đây bởi các kết quả tốt thu được trong lĩnh vực xử lý ngôn ngữ tự nhiên.
2. Ý tưởng chính của RNN (Recurrent Neural Network) là sử dụng chuỗi các thông tin. Trong các mạng nơ-ron truyền thống tất cả các đầu vào và cả đầu ra là độc lập với nhau. Tức là chúng không liên kết thành chuỗi với nhau. Nhưng các mô hình này không phù hợp trong rất nhiều bài toán.
3. RNN được gọi là hồi quy (Recurrent) bởi lẽ chúng thực hiện cùng một tác vụ cho tất cả các phần tử của một chuỗi với đầu ra phụ thuộc vào cả các phép tính trước đó. Nói cách khác, RNN có khả năng nhớ các thông tin được tính toán trước đó. Trên lý thuyết, RNN có thể sử dụng được thông tin của một văn bản rất dài, tuy nhiên thực tế thì nó chỉ có thể nhớ được một vài bước trước đó mà thôi.

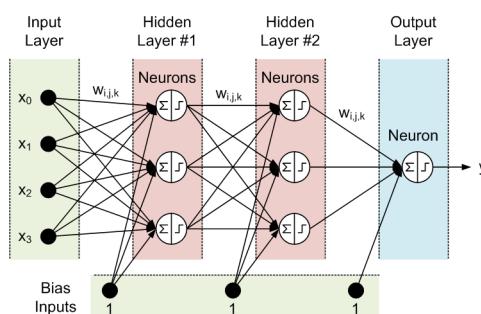


Figure 10: Mô hình mạng RNN



2.2 Các nghiên cứu liên quan

Nhóm thực hiện nghiên cứu trên một số thư viện nổi tiếng để tham khảo cũng như có cái nhìn tổng quan về độ tốt của model được nhóm hiện thực so với các thư viện có sẵn.

1. Tesseract OCR
2. VietOCR
3. EasyOCR
4. PadleOCR

Bên cạnh đó là các paper nghiên cứu nổi bật có liên quan đến đề tài mà nhóm đang hiện thực:

1. Adapting the Tesseract Open Source OCR Engine for Multilingual OCR (<https://storage.googleapis.com/pub-tools-public-publication-data/pdf/35248.pdf>)
2. ICDAR2019 Competition on Scanned Receipt OCR and Information Extraction (<https://arxiv.org/pdf/2103.10213v1.pdf>)
3. OCR-free Document Understanding Transformer (<https://arxiv.org/pdf/2111.15664v4.pdf>)
4. An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition (<https://arxiv.org/pdf/1507.05717v1.pdf>)

2.3 Công nghệ sử dụng

1. Python

- Một ngôn ngữ lập trình thông dịch (interpreted), hướng đối tượng (object-oriented), và là một ngôn ngữ bậc cao (high-level) ngữ nghĩa động (dynamic semantics). Python hỗ trợ các module và gói (packages), khuyến khích chương trình module hóa và tái sử dụng mã. Trình thông dịch Python và thư viện chuẩn mở rộng có sẵn dưới dạng mã nguồn hoặc dạng nhị phân miễn phí cho tất cả các nền tảng chính và có thể được phân phối tự do.
- Các nhà phát triển có thể dễ dàng đọc và hiểu một chương trình Python vì ngôn ngữ này có cú pháp cơ bản giống tiếng Anh. Python giúp cải thiện năng suất làm việc của các nhà phát triển vì so với những ngôn ngữ khác, họ có thể sử dụng ít dòng mã hơn để viết một chương trình Python.
- Phù hợp với việc phát triển các ứng dụng liên quan đến trí tuệ nhân tạo, xử lý ảnh,...

2. TensorFlow

- Là thư viện mã nguồn mở lớn và nổi tiếng nhất dành cho mảng Machine Learning (Học máy), và nó được xây dựng bởi những nhà phát triển từ Google. TensorFlow sẽ giúp giải quyết các bài toán nhanh chóng và đơn giản hơn thông qua việc tạo các mô hình tính toán trong Machine Learning trên máy tính.
- TensorFlow hỗ trợ rất tốt cho ngôn ngữ Python

3. OpenCV

- OpenCV là tên viết tắt của open source computer vision library – có thể được hiểu là một thư viện nguồn mở cho máy tính. Cụ thể hơn OpenCV là kho lưu trữ các mã nguồn mở được dùng để xử lý hình ảnh, phát triển các ứng dụng đồ họa trong thời gian thực.
- OpenCV cho phép cải thiện tốc độ của CPU khi thực hiện các hoạt động real time. Nó còn cung cấp một số lượng lớn các mã xử lý phục vụ cho quy trình của thị giác máy tính hay các learning machine khác.
- OpenCV sở hữu giao diện thiên thiện với mọi loại ngôn ngữ lập trình, ví dụ như C++, C, Python hay Java...

3 Hướng tiếp cận

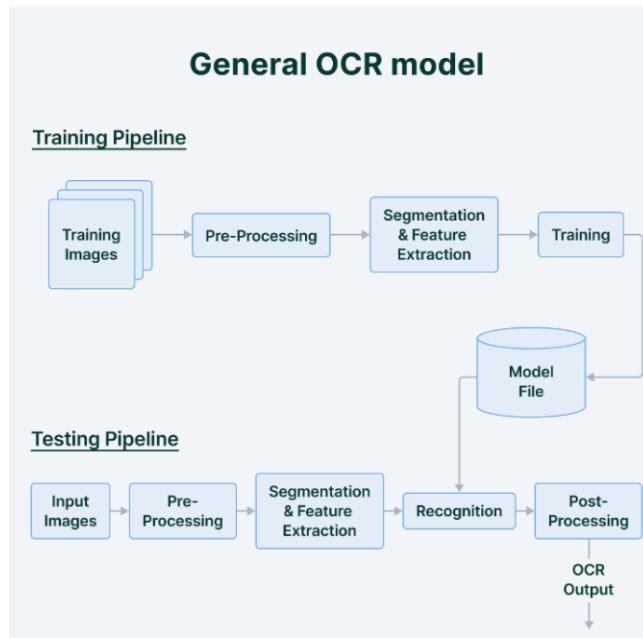


Figure 11: General OCR Model

Với việc nghiên cứu một mô hình OCR tổng quan như trên thì nhóm đã đề xuất một sơ đồ hướng đi cụ thể cho bài toán nhận diện biển số xe như hình bên dưới:

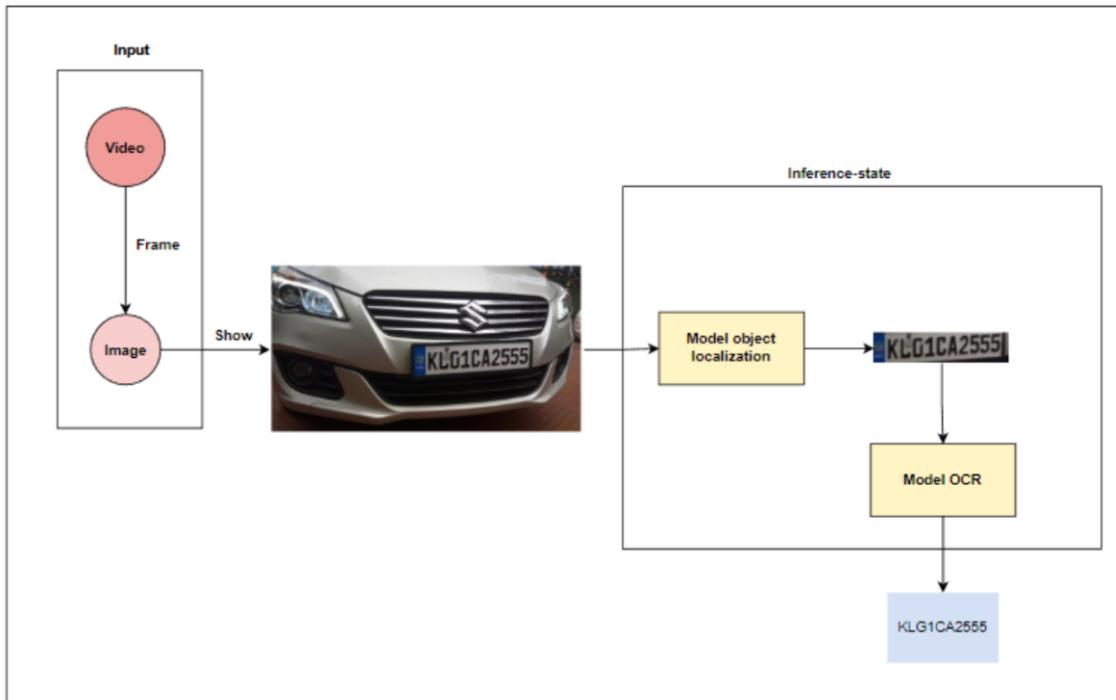


Figure 12: Hướng đi cho bài toán

Mục tiêu của nhóm trước hết là hoàn thiện model OCR theo 2 phương pháp : Classification, End-to-end. Sau đó tiếp tục nghiên cứu và hoàn thành mô hình đã được đề với việc thành công thử nghiệm trên video (nếu còn thời gian), kết hợp với Yolo để phát hiện vật thể - cụ thể là đối tượng biển số xe.

Bên cạnh đó, nhóm cũng thiết kế một hướng đi khác - Model Localization với các hướng tiếp cận cụ thể như sau:

- Xây dựng các mô hình SSD, Fast RCNN, RCNN (cần một tập lớn dữ liệu bounding box)
- Config lại Yolo model chỉ quan tâm đến các vật thể bảng số xe.

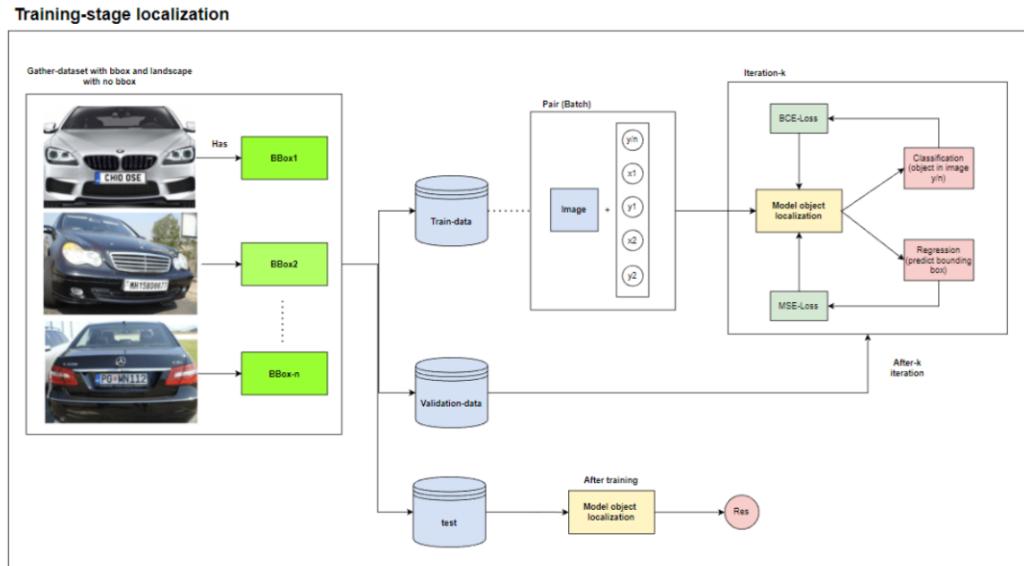


Figure 13: Hướng đi model localization

4 Hiện thực

4.1 Mô tả dữ liệu

4.1.1 Dữ liệu ký tự

Data gồm 350000 sample ảnh kích thước (128,128,3) nhưng đã qua các bước xử lý augmentation và chuyển qua grayscale, mỗi hình ảnh là một chữ riêng biệt và nhãn sẽ là chữ đó.

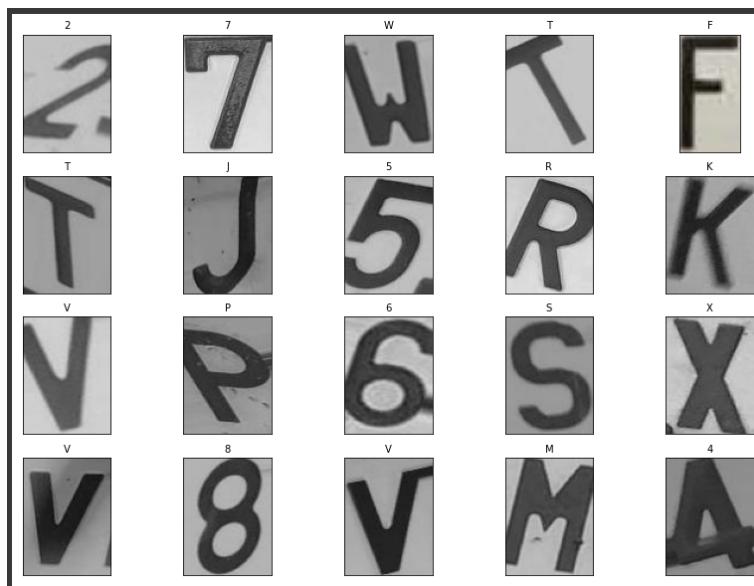


Figure 14: Mẫu dữ liệu cho mô hình Classification

4.1.2 Dữ liệu biển số xe

Data set được chọn bao gồm 100000 sample kích thước (1025,128,3). Mỗi sample gồm một biển số xe đầy đủ, mỗi sample đã được đánh nhãn.

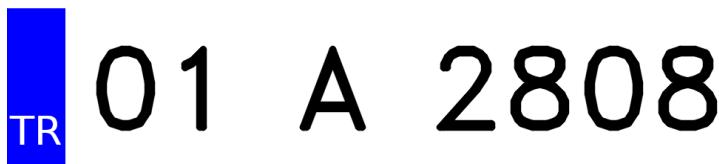


Figure 15: Mẫu dữ liệu cho mô hình End2End

4.2 Tiền xử lý dữ liệu

4.2.1 Dữ liệu ký tự

Với việc dataset ban đầu đã ở định dạng grayscale, được augmentation cũng như gắn nhãn một cách đầy đủ, nên dataset chỉ được tiền xử lý một cách khá đơn giản

- Resize ảnh thành (128,128) (Xử lý với ImageDataGenerator)
- Normalize bằng cách chia các pixel cho 255



```
dataGen = tf.keras.preprocessing.image.ImageDataGenerator(  
    rescale=1./255,  
)  
  
Train_gen = dataGen.flow_from_dataframe(  
    Train_df,  
    x_col = 'Path',  
    y_col = 'Labels',  
    target_size = (128, 128),  
    batch_size = 512,  
    class_mode = 'categorical',  
    color_mode='rgb'  
)  
  
Test_gen = dataGen.flow_from_dataframe(  
    Test_df,  
    x_col = 'Path',  
    y_col = 'Labels',  
    target_size = (128, 128),  
    batch_size = 512,  
    class_mode = 'categorical',  
    color_mode='rgb'  
)  
  
Valid_gen = dataGen.flow_from_dataframe(  
    Valid_df,  
    x_col = 'Path',  
    y_col = 'Labels',  
    target_size = (128, 128),  
    batch_size = 512,  
    class_mode = 'categorical',  
    color_mode='rgb'  
)
```

Figure 16: Tiền xử lý ảnh

4.2.2 Dữ liệu biến số xe

- Đọc hình ảnh đầu vào và chuyển sáng gray-scale
- Resize mỗi ảnh thành (128,32)
- Mở rộng chiều của ảnh thành (128,32,1) để phù hợp với đầu vào
- Normalize bằng cách chia các pixel cho 255
- Đối với nhãn chúng ta mã hóa ký tự bằng một hàm. Ví dụ: 'abcd' sẽ mã hóa thành '0123'.

```
1     char_list = string.ascii_letters+string.digits  
2  
3     def encode_to_labels(txt):  
4         # encoding each output word into digits  
5         dig_lst = []  
6         for index, char in enumerate(txt):  
7             try:  
8                 dig_lst.append(char_list.index(char))  
9             except:  
10                 print(char)  
11     return dig_lst
```

4.3 Xây dựng mô hình

Dể giải quyết bài toán trên nhóm quyết định sử dụng hai hướng mô hình: Classification và End2End.

Mô hình bài toán Classification

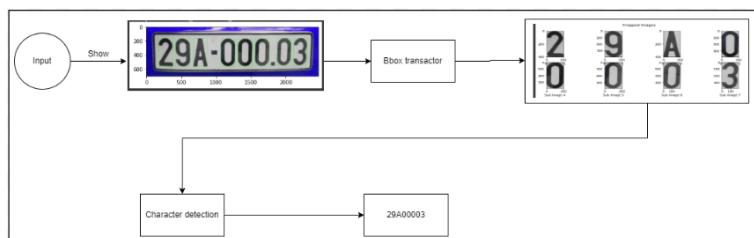


Figure 17: Mô hình bài toán classification

Trước khi được đưa vào model (model được hiện thực ở bước "Character detection" như trong hình), ảnh đầu vào sẽ được đưa qua bbox extactor cũng như giải thuật ad-hoc mà nhóm đã hiện thực sẵn.

Model được nhóm hiện thực dựa trên sơ đồ về mạng CNN các thành phần cụ thể như sau:

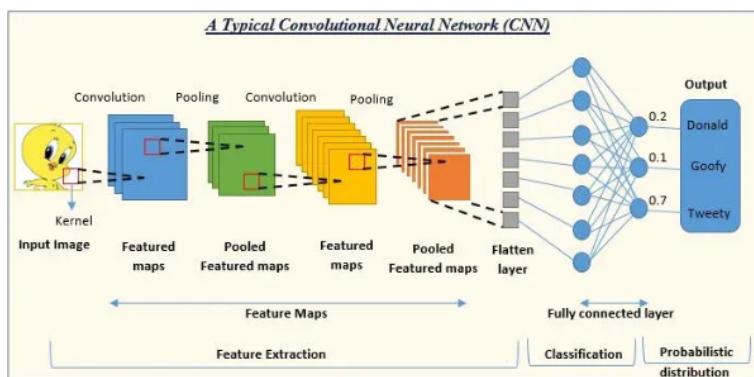


Figure 18: Sơ đồ minh họa cho CNN

1. **Convolution layer** sử dụng các bộ lọc trên input đầu vào
2. Xen kẽ giữa các Convolution layer là các **MAX Pooling layer** có chức năng giảm kích thước dữ liệu nhưng vẫn giữ được các thuộc tính quan trọng. Việc giảm kích thước dữ liệu giúp giảm các phép tính toán trong model.
3. Sau khi ảnh được truyền qua nhiều convolutional layer và pooling layer thì model đã học được tương đối các đặc điểm của ảnh. Khi đó tensor của output của layer cuối cùng sẽ được là phẳng thành vector (**lớp Flatten**) và đưa vào một lớp được kết nối như một mạng nơ-ron với các Fully connected layer (**lớp Dense**)

4.3.1 Mô hình End2End

Đối với bài toán nhận diện chuỗi trinh tự chúng ta thường dùng mạng Recurrent Neural Networks (RNN), còn để nhận dạng hình ảnh chúng ta thường dùng Convolution Neural Networks(CNN). Vì vậy để giải quyết bài toán trên nhóm thử kết hợp cả hai giải pháp trên.

1. Convolutional Layer: Đầu tiên input sẽ đi qua lớp CNN, đầu ra của lớp CNN sẽ là một feature maps.
2. Recurrent Layers: Đầu vào của lớp này sẽ là các chuỗi đặc tính từ lớp Convolutional. Recurrent Layers bao gồm bidirectional LSTM (Long short term memory). Đầu ra từ lớp RNN sẽ bao gồm các giá trị xác suất cho mỗi nhãn tương ứng với mỗi đặc điểm đầu vào (input feature). Giả sử đầu vào cho recurrent layer có kích thước là (batch_size, 250) và tổng số labels là 53 thì đầu ra của recurrent layer sẽ có kích thước (batch_size, 250, 53).

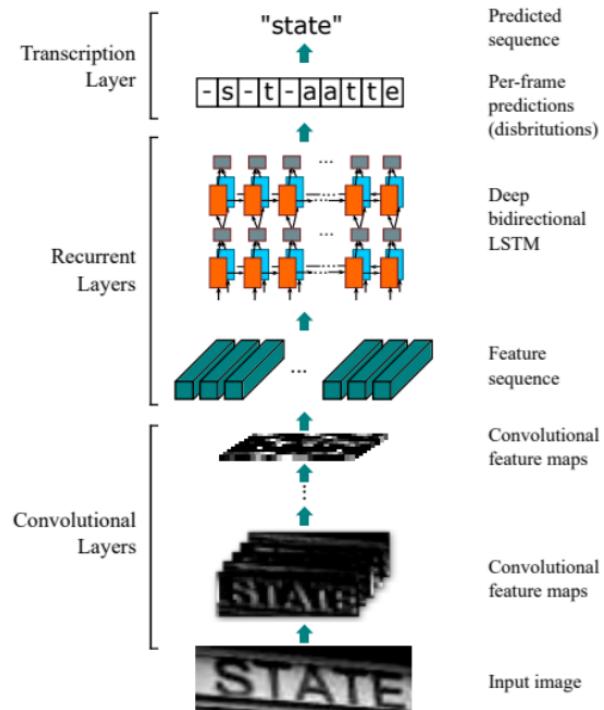


Figure 19: Mô hình tổng quan của RCNN

3. **Transcription Layer:** Thành phần cuối cùng là Transcription Layers. Nó sử dụng Connectionist temporal classification(CTC) cho từng frame. Ví dụ ta có đầu vào là ảnh với text là "hello", đầu ra của RNN có thể là ['h', 'h', 'e', 'e', 'l', 'l', 'l', 'o', 'o', 'o']. Nếu chúng ra gộp các kí tự giống lại với nhau thì kết quả sẽ là 'hel0' sai với text đầu vào. CTC giải quyết bằng cách thêm khoảng trắng vào các kí tự giống nhau liên tục và khi decode sẽ cho ra output chính xác với dữ liệu đầu vào

CTC Loss

- Để training mạng nơ-ron chúng ta cần một hàm loss function. Ở đây chúng ta sài CTC Loss function, nó khác với các hàm tính loss ở các model deep learning khác. Để tính CTC Loss chúng ta thực hiện theo các bước sau:



4.4 Hiện thực mô hình

4.4.1 Mô hình Classification

```
model = tf.keras.models.Sequential([
    # This is the first convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu',
                         input_shape=(128, 128, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    # The second convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The third convolution
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The fourth convolution
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    # Flatten
    tf.keras.layers.Flatten(),

    # Fully connected + Dropout => prevent overfitting
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.5),

    # Output
    tf.keras.layers.Dense(len(list_labels), activation='softmax')
])
```

Figure 20: Kiến trúc của mô hình Classification

Mô hình được xây dựng dựa trên mô tả đã được trình bày trong phần **Xây dựng mô hình**

- Đầu vào của mô hình là ảnh có shape (128,128,3)
- Ảnh đầu vào sẽ lần lượt được đưa qua 4 lớp Convolutional. Xen kẽ giữa các Conv2D (kích thước kernel là 3x3, các tham số stride và padding được truyền mặc định với giá trị bằng 1) là các Max Pooling layer kích thước 2x2.
- Trước khi được đưa qua các lớp Dense (fully connected layer) thì đầu ra của ảnh khi đi qua các lớp tích chập được duỗi phẳng bằng lớp Flatten như hình trên
- Xen kẽ giữa các lớp Dense, nhóm sử dụng các lớp Dropout. Lý do là khi chúng ta sử dụng fully connected layer, các neural sẽ phụ thuộc “mạnh” lẫn nhau trong suốt quá trình huấn luyện, điều này làm giảm sức mạnh cho mỗi neural và dẫn đến bị over-fitting tập train. Nên Dropout là một giải pháp khá hữu dụng để đảm bảo hiệu quả của model
- Cuối cùng là sử dụng softmax để phân loại đầu ra cho model.
- Model sử dụng hàm kích hoạt (activation function) reLu. ReLU viết tắt của Rectified Linear Unit, là 1 hàm phi tuyến. Với đầu ra là: $f(x) = \max(0, x)$. Sở dĩ reLu thực sự quan trọng là vì dữ liệu trong thế giới mà chúng ta tìm hiểu (image) là các giá trị tuyến tính không âm.

```
model.compile(loss = 'categorical_crossentropy', optimizer='rmsprop',
               metrics=['accuracy'])
```

Figure 21: Loss function

Cross-entropy được sử dụng mặc định cho các bài toán phân đa lớp. Trong bài toán phân đa lớp, mục đích của mô hình là dự đoán xác suất của một điểm dữ liệu rơi vào class (lớp) nào trong số các class {0, 1, 2, ..., a,b,c,...}, mỗi class tương ứng với ký tự có thể sử dụng trên biển số xe tương ứng như dữ liệu được nhóm sử dụng



Training model

- Tỉ lệ tập Train/Validation/Test là 72/8/20

```
[ ] Train_df, Test_df = train_test_split(df, test_size=0.2, random_state=101)
    Train_df, Valid_df = train_test_split(Train_df, test_size=0.1, random_state=101)
```

Figure 22: Sử dụng train_test_split để chia tập dữ liệu theo nhu cầu

- Batch size là 512, được cài đặt trong quá trình generator từng tập Train, Validation và Test đã được chia trước đó.

```
dataGen = tf.keras.preprocessing.image.ImageDataGenerator(
    rescale=1./255,
)

Train_gen = dataGen.flow_from_dataframe(
    Train_df,
    x_col = 'Path',
    y_col = 'Labels',
    target_size = (128, 128),
    batch_size = 512,
    class_mode = 'categorical',
    color_mode='rgb'
)

Test_gen = dataGen.flow_from_dataframe(
    Test_df,
    x_col = 'Path',
    y_col = 'Labels',
    target_size = (128, 128),
    batch_size = 512,
    class_mode = 'categorical',
    color_mode='rgb'
)

Valid_gen = dataGen.flow_from_dataframe(
    Valid_df,
    x_col = 'Path',
    y_col = 'Labels',
    target_size = (128, 128),
    batch_size = 512,
    class_mode = 'categorical',
    color_mode='rgb'
)
```

Figure 23: ImageDataGenerator



- Kết quả training (epochs=10): Thời gian train model là 5 phút , có bật GPU trong quá trình train

```
▶ model.fit(Train_gen, epochs = 10, validation_data = Valid_gen, verbose = 1)

↳ Epoch 1/10
51/51 [=====] - 35s 456ms/step - loss: 3.0692 - accuracy: 0.1532 - val_loss: 0.8658 - val_accuracy: 0.8244
Epoch 2/10
51/51 [=====] - 21s 400ms/step - loss: 1.0747 - accuracy: 0.6757 - val_loss: 0.1836 - val_accuracy: 0.9574
Epoch 3/10
51/51 [=====] - 21s 407ms/step - loss: 0.4844 - accuracy: 0.8557 - val_loss: 0.0898 - val_accuracy: 0.9780
Epoch 4/10
51/51 [=====] - 21s 408ms/step - loss: 0.2747 - accuracy: 0.9166 - val_loss: 0.0389 - val_accuracy: 0.9888
Epoch 5/10
51/51 [=====] - 21s 407ms/step - loss: 0.1764 - accuracy: 0.9470 - val_loss: 0.0294 - val_accuracy: 0.9913
Epoch 6/10
51/51 [=====] - 22s 421ms/step - loss: 0.1348 - accuracy: 0.9590 - val_loss: 0.0137 - val_accuracy: 0.9962
Epoch 7/10
51/51 [=====] - 21s 403ms/step - loss: 0.1017 - accuracy: 0.9692 - val_loss: 0.0090 - val_accuracy: 0.9965
Epoch 8/10
51/51 [=====] - 21s 405ms/step - loss: 0.0842 - accuracy: 0.9749 - val_loss: 0.0114 - val_accuracy: 0.9979
Epoch 9/10
51/51 [=====] - 21s 405ms/step - loss: 0.0729 - accuracy: 0.9794 - val_loss: 0.0080 - val_accuracy: 0.9969
Epoch 10/10
51/51 [=====] - 21s 404ms/step - loss: 0.0581 - accuracy: 0.9830 - val_loss: 0.0039 - val_accuracy: 0.9990
<keras.callbacks.History at 0x7f9d50386250>
```

Figure 24: Training

4.4.2 Mô hình End2End

- Như đã trình bày ở trên mô hình gồm 3 phần:
 - Convolutuinal Layers: để trích xuất feature của input
 - Recurrent Layers: Để dự đoán đầu ra cho từng step
 - CTC Loss function

Model: "model_1"			
Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[None, 32, 128, 1]	0	[]
conv2d (Conv2D)	(None, 32, 128, 64)	640	['input_1[0][0]']
max_pooling2d (MaxPooling2D)	(None, 16, 64, 64)	0	['conv2d[0][0]']
conv2d_1 (Conv2D)	(None, 16, 64, 128)	73856	['max_pooling2d[0][0]']
max_pooling2d_1 (MaxPooling2D)	(None, 8, 32, 128)	0	['conv2d_1[0][0]']
conv2d_2 (Conv2D)	(None, 8, 32, 256)	295168	['max_pooling2d_1[0][0]']
conv2d_3 (Conv2D)	(None, 8, 32, 256)	590080	['conv2d_2[0][0]']
max_pooling2d_2 (MaxPooling2D)	(None, 4, 32, 256)	0	['conv2d_3[0][0]']
conv2d_4 (Conv2D)	(None, 4, 32, 512)	1180160	['max_pooling2d_2[0][0]']
batch_normalization (BatchNorm alization)	(None, 4, 32, 512)	2048	['conv2d_4[0][0]']
conv2d_5 (Conv2D)	(None, 4, 32, 512)	2359008	['batch_normalization[0][0]']
batch_normalization_1 (BatchNo rmalization)	(None, 4, 32, 512)	2048	['conv2d_5[0][0]']
max_pooling2d_3 (MaxPooling2D)	(None, 2, 32, 512)	0	['batch_normalization_1[0][0]']
conv2d_6 (Conv2D)	(None, 1, 31, 512)	1049088	['max_pooling2d_3[0][0]']
lambda (Lambda)	(None, 31, 512)	0	['conv2d_6[0][0]']
bidirectional (Bidirectional)	(None, 31, 256)	656384	['lambda[0][0]']
bidirectional_1 (Bidirectional)	(None, 31, 256)	394240	['bidirectional[0][0]']
dense (Dense)	(None, 31, 63)	16191	['bidirectional_1[0][0]']
the_labels (InputLayer)	[(None, 7)]	0	[]
input_length (InputLayer)	[(None, 1)]	0	[]
label_length (InputLayer)	[(None, 1)]	0	[]
ctc (Lambda)	(None, 1)	0	['dense[0][0]', 'the_labels[0][0]', 'input_length[0][0]', 'label_length[0][0]']

Figure 25: Kiến trúc của mô hình

- Dầu vào là ảnh có kích thước (128,32).
- Có tất cả 7 Con2d với 6 cái có kernel (3,3) và 1 cái có kernel là (2,2).
- Hai max-pooling layers với kích thước (2,2) và hai max-pooling layers với kích thước (2,1).
- Chúng ta có sử dụng thêm batch normalization layers sau convolutional thứ 5 và 6 để tăng tốc độ training.
- Sau đó sử dụng 1 hàm lambda để ép dầu ra từ conv tương thích với dầu vào LSTM.
- Sử dụng 2 Bidirectional LSTM layers với mỗi layer là 128 units. RNN layer có đầu ra với kích thước (batch_size, 31, 63). Số 63 ở đây là tổng các lớp đầu ra bao gồm cả kí tự trống

CTC Loss

CTC loss yêu cầu 4 tham số để tính toán đó là: predicted outputs, ground truth labels, input sequence length to LSTM and ground truth label length. Sau khi chúng ta custom được hàm loss thì sẽ pass nó vào trong mô hình trên.

```

1 labels = Input(name='the_labels', shape=[max_label_len], dtype='float32')
2 input_length = Input(name='input_length', shape=[1], dtype='int64')
3 label_length = Input(name='label_length', shape=[1], dtype='int64')
4

```



```
5 def ctc_lambda_func(args):
6     y_pred, labels, input_length, label_length = args
7     return K.ctc_batch_cost(labels, y_pred, input_length, label_length)
8
9 loss_out = Lambda(ctc_lambda_func, output_shape=(1,), name='ctc')([outputs, labels,
10               input_length, label_length])
11 #model to be used at training time
12 crnn_model = Model(inputs=[inputs, labels, input_length, label_length], outputs=loss_out)
```

- Training Model

Tỉ lệ tập Train/Valid/Test là 60/20/20

```
1 crnn_model.compile(loss={'ctc': lambda y_true, y_pred: y_pred}, optimizer = 'adam')
2 filepath="best_model.hdf5"
3 checkpoint = ModelCheckpoint(filepath=filepath, monitor='val_loss', verbose=1,
4                               save_best_only=True, mode='auto')
4 callbacks_list = [checkpoint]
```

```
1 training_img = np.array(training_img)
2 train_input_length = np.array(train_input_length)
3 train_label_length = np.array(train_label_length)
4
5 valid_img = np.array(valid_img)
6 valid_input_length = np.array(valid_input_length)
7 valid_label_length = np.array(valid_label_length)
```

```
1 batch_size = 32
2 epochs = 10
3 crnn_model.fit(x=[training_img, train_padded_txt, train_input_length, train_label_length
        ], y=np.zeros(len(training_img)), batch_size=batch_size, epochs = epochs,
        validation_data = ([valid_img, valid_padded_txt, valid_input_length,
        valid_label_length], [np.zeros(len(valid_img))]), verbose = 1, callbacks =
        callbacks_list)
```

Với epochs = 10, thời gian training là 20 phút có bật GPU

- Kết quả dự đoán trên tập test

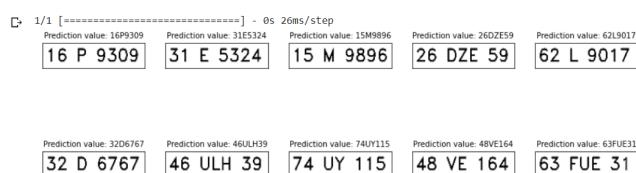


Figure 26: Kết quả dự đoán trên tập test

4.5 Kết quả mô hình và đánh giá

Nhóm tập trung thể hiện kết quả của mô hình Classification sau quá trình train được hiện thực trên Colab.

- Độ chính xác trên tập Test tương đối cao (99.82%)

```
▶ test_loss, test_acc = model.evaluate(Test_gen, verbose=2)
[ 14/14 - 5s - loss: 0.0095 - accuracy: 0.9982 - 5s/epoch - 387ms/step ]
```

Figure 27: Test accuracy

- Một vài sample được dự đoán trên tập Test (thể hiện kết quả dự đoán khá chính xác)

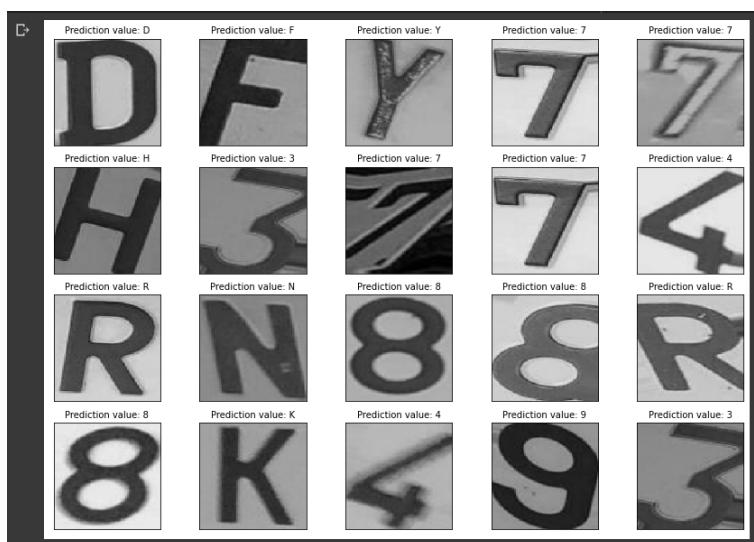


Figure 28: Predict in Test dataset

- Một vài sample khác ở bên ngoài

- Ảnh bên ngoài sẽ được preprocess lại thành ảnh grayscale với kích thước (128,128,3) phù hợp với model.
- Model adapt khá tốt với nhiều loại data khác, model chỉ sai với dữ liệu có dạng chữ quá nhỏ trong hình nhưng bbox extractor khi trích xuất ra các chữ bự chiếm đa số khung hình nên ta không lo ngại điều này trên tập data biển số xe.

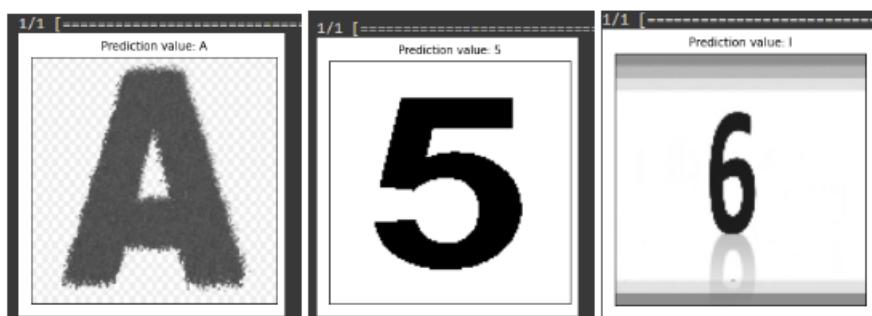


Figure 29: Predict in another sample

- Kết hợp với Bbox Extractor

- **Bbox extractor** được nhóm viết là một script tự động để trích xuất các chữ trong hình ra dựa vào các thuật toán xử lý ảnh cũng như là thuật toán ad-hoc của nhóm.
- **Có hai kiểu biển số xe chính:** Các chữ nằm trong một dòng và chia ra 2 dòng. Cả hai trường hợp này với tập ảnh kiểm được, đều trích xuất được hết đúng chữ cũng như là đúng thứ tự

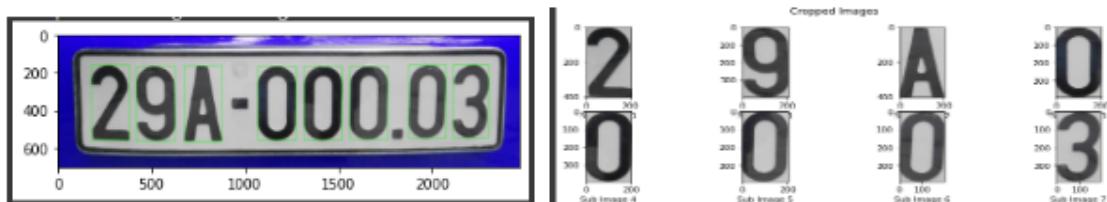


Figure 30: Bbox extractor

- **Điểm mạnh:** Thời gian cho ra kết quả nhanh và model khá nhẹ

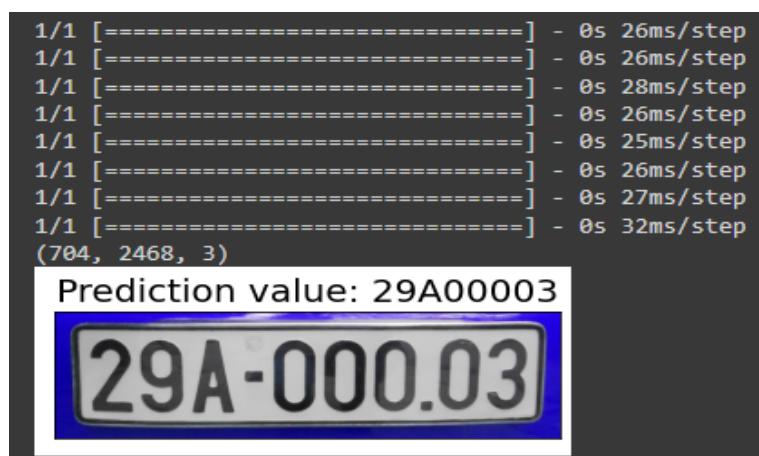


Figure 31: Predict with Bbox extractor

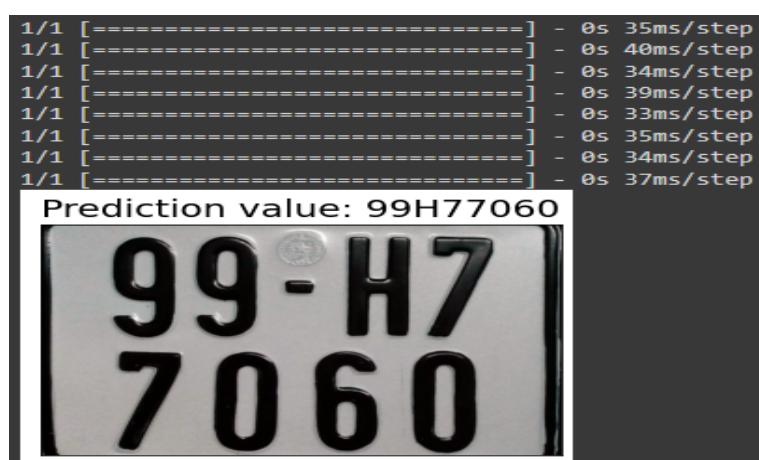


Figure 32: Predict with Bbox extractor

- **Điểm yếu:** do script tự động trích xuất nên vẫn còn một số case sai trích xuất sai mà nhóm chưa phát hiện được.



- Thử nghiệm training với input đầu vào có shape là (128,128) cũng đưa ra kết quả tương đối tốt (tương đương với shape (128,128,3) được trình bày ở phía trên)

```
[60] model.fit(Train_gen, epochs = 10, validation_data = Valid_gen, verbose = 1)

Epoch 1/10
50/50 [=====] - 24s 475ms/step - loss: 1.4065 - accuracy: 0.5778 - val_loss: 0.3199 - val_accuracy: 0.9282
Epoch 2/10
50/50 [=====] - 21s 416ms/step - loss: 0.5577 - accuracy: 0.8310 - val_loss: 0.0914 - val_accuracy: 0.9796
Epoch 3/10
50/50 [=====] - 19s 379ms/step - loss: 0.2996 - accuracy: 0.9097 - val_loss: 0.0678 - val_accuracy: 0.9838
Epoch 4/10
50/50 [=====] - 19s 380ms/step - loss: 0.1884 - accuracy: 0.9451 - val_loss: 0.0240 - val_accuracy: 0.9947
Epoch 5/10
50/50 [=====] - 19s 379ms/step - loss: 0.1332 - accuracy: 0.9613 - val_loss: 0.0217 - val_accuracy: 0.9947
Epoch 6/10
50/50 [=====] - 19s 377ms/step - loss: 0.0989 - accuracy: 0.9700 - val_loss: 0.0103 - val_accuracy: 0.9979
Epoch 7/10
50/50 [=====] - 19s 378ms/step - loss: 0.0842 - accuracy: 0.9736 - val_loss: 0.0094 - val_accuracy: 0.9982
Epoch 8/10
50/50 [=====] - 19s 378ms/step - loss: 0.0696 - accuracy: 0.9791 - val_loss: 0.0056 - val_accuracy: 0.9993
Epoch 9/10
50/50 [=====] - 19s 378ms/step - loss: 0.0570 - accuracy: 0.9835 - val_loss: 0.0076 - val_accuracy: 0.9979
Epoch 10/10
50/50 [=====] - 19s 377ms/step - loss: 0.0554 - accuracy: 0.9833 - val_loss: 0.0054 - val_accuracy: 0.9993
<keras.callbacks.History at 0x7f187d6ee250>
```

Figure 33: Input shape (128,128)

Ngoài ra nhóm cũng đã kết hợp với Yolo (đã được config để hoạt động trên đối tượng là biển số xe) để đánh giá về khả năng của model với những đầu vào dữ liệu mới.

- Kết quả detect biển số xe khi sử dụng Yolo



Figure 34: Detect Liscent Plate with Yolo



- Kết hợp với model mà nhóm đã hiện thực

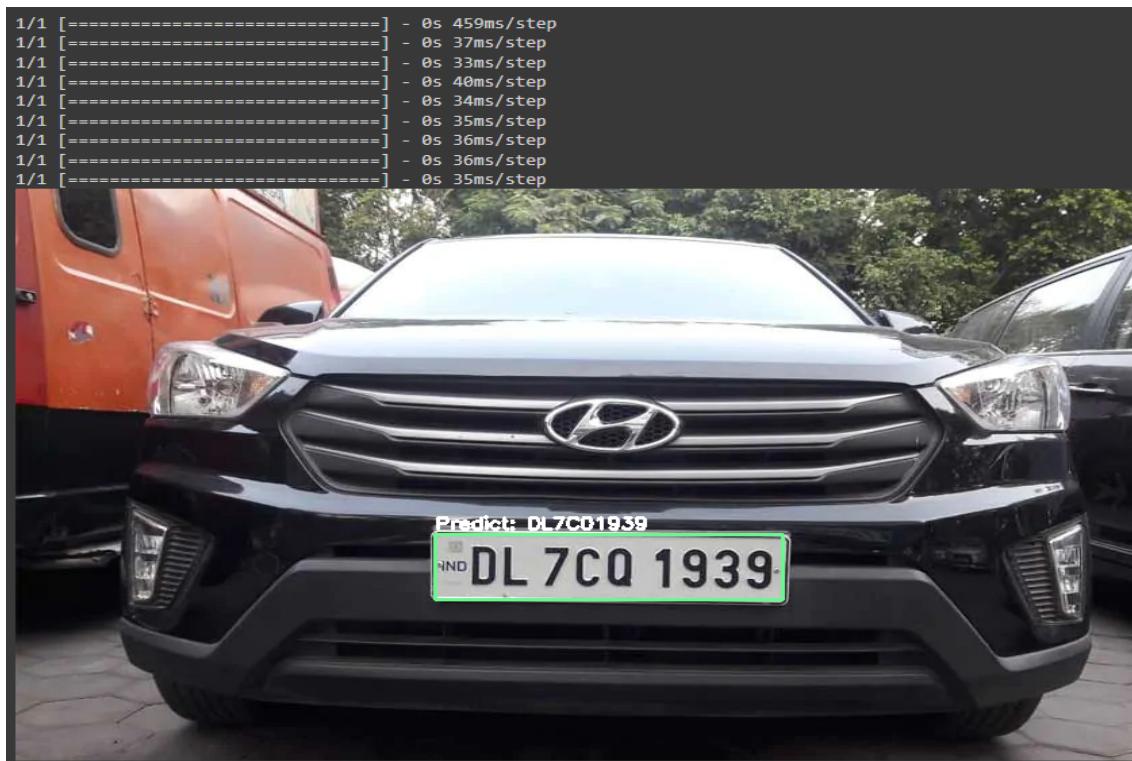


Figure 35: Predict Liscent Plate with Yolo



Figure 36: Predict Liscent Plate with Yolo

- Dánh giá:

- Mô hình thể hiện khá tốt trên những biển số được detect chính xác, không bị mờ, lệch hướng.
- Cần cải thiện, train thêm với Yolo để có cơ hội đánh giá nhiều hơn với model của nhóm.

- Một số thử nghiệm liên quan:

1. Tesseract OCR



Figure 37: Thử nghiệm trên Tesseract OCR

2. Easy OCR



Figure 38: Thử nghiệm trên Easy OCR



5 Kết luận

Tóm lại là trong suốt quá trình học tập và làm việc, nhóm chúng em tổng kết lại các công việc đã hoàn thành được như sau:

- Hiện thực thành công BBox Extractor module dùng để đọc input là các biển số xe và phân tách thành các hình ảnh riêng biệt. Mỗi hình ảnh là một ký tự và có nhãn là ký tự đó.
- Hiện thực thành công Training module dùng để huấn luyện AI dựa trên tập dữ liệu có sẵn và tập dữ liệu được phân tích từ BBox Extractor module.
- Hiện thực thành công Classification module dự đoán biển số xe sử dụng các tập dữ liệu các ký tự đã được huấn luyện cho AI dựa trên mô hình RNN.
- Kết hợp các module hiện thực tạo nên module End2End dựa trên YOLO, một phương pháp phát hiện và nhận diện biển số xe Việt Nam do nhóm chúng em tạo ra.

Module End2End của nhóm chúng em có tỉ lệ nhận diện chính xác khá cao đối với các biển số xe không bị mờ, khuất góc, lệch hướng,... Trong tương lai, nhóm chúng em sẽ tiếp tục phát triển, nghiên cứu thêm để có thể nhận diện các biển số xe bị mờ, khuất, lệch hướng tốt hơn. Từ đó, phương pháp của tụi em có thể giúp ích cho các ứng dụng ở nước Việt Nam, xây dựng được tập dữ liệu chính xác, đa dạng và đúng với điều kiện thực tế của nước ta.