TRƯỜNG ĐẠI HỌC BÁCH KHOA - ĐẠI HỌC ĐÀ NẴNG

KHOA ĐIỆN TỬ - VIỄN THÔNG

BÁO CÁO ĐỒ ÁN
PBL4: Trí tuệ nhân tạo

ĐỀ TÀI:

Nhận dạng biển số xe với Deep Neural Networks

Giảng viên hướng dẫn: **Trần Thị Minh Hạnh**

Sinh viên thực hiện:

Họ và tên	**MSSV**	**Mã học phần**
Lê Phạm Công	106200221	20.44A
Phan Công Danh	106200222	20.44A
Trần Đình Thi	106200246	20.44A

Đà Nẵng, ngày 13 tháng 05 năm 2024

Mục lục

Tóm tắt

Báo cáo này nghiên cứu việc áp dụng mô hình LPRNet, một kiến trúc mạng nơ-ron tích chập sâu (Deep Convolutional Neural Network - CNN), trong bài toán nhận dạng biển số xe. Mô hình này tận dụng lợi thế của CNN, một công nghệ đã được chứng minh là vượt trội trong nhiều tác vụ thị giác máy tính, từ phân loại ảnh đến phân đoạn ngữ nghĩa. Các nghiên cứu gần đây đã chỉ ra rằng CNN không chỉ giỏi trong việc phân loại ảnh mà còn trong các ứng dụng như phát hiện biến thể mã độc và nhận dạng đối tượng.

Kết quả độ chính xác 90% và khi huấn luyện với bộ dữ liệu 10000 ảnh biển số xe Trung Quốc và thực hiện đánh giá với 1000 ảnh (dữ liệu được tổng hợp từ bô dữ liệu CCPD2020 [1]). Với dữ liệu ảnh biển số xe Việt Nam tổng hợp từ nguồn ảnh chụp thực tế (12320 tấm ảnh huấn luyện và 1000 tấm ảnh đánh giá) mô hình đạt được độ chính xác 75%.

Mã nguồn của báo cáo có thể được tìm thấy tại: `https://github.com/lephamcong/PBL4_LPRNet.git`

1 Giới thiệu

Trong quản lý giao thông đô thị, giám sát video, nhận diện phương tiện và quản lý bãi đậu xe, nhận dạng biển số xe là một nhiệm vụ đầy thách thức và quan trọng. Độ rõ của hình ảnh, điều kiện ánh sáng, yếu tố thời tiết, sự biến dạng của hình ảnh và sự đa dạng của ký tự trên biển số làm cho vấn đề nhận dạng biển số xe trở nên phức tạp hơn. Một hệ thống nhận dạng biển số xe nên có khả năng đối phó với sự thay đổi của môi trường mà không mất đi độ chính xác.

Hệ thống Nhận dạng Biển số Xe tự động vững chắc cần phải đối phó với đa dạng môi trường trong khi duy trì mức độ chính xác cao, nói cách khác, hệ thống này nên hoạt động tốt trong điều kiện tự nhiên.

Báo cáo này thực hiện bài toán nhận dạng biển số xe với mô hình LPRNet [2] dựa trên Deep Neural Network, LPRNet được thiết kế để hoạt động mà không cần phân đoạn trước và nhận dạng các ký tự sau đó.

LPRNet dựa trên Deep Convolutional Neural Network (Mạng Neural tích chập sâu) . Các nghiên cứu gần đây đã chứng minh hiệu quả và sự vượt trội của Convolutional Neural Network (CNN) trong nhiều tác vụ Thị giác Máy tính như phân loại ảnh, phát hiện đối tượng và phân đoạn ngữ nghĩa.

Convolutional Neural Network có khả năng thích nghi cao và khả năng nhận dạng xuất sắc, đã được sử dụng rộng rãi trong các lĩnh vực như phân loại và nhận dạng cũng như phát hiện mục tiêu [3][4][5]. Ví dụ bao gồm phân loại rượu vang và

hoa iris của Ý [6], phát hiện biến thể mã độc [7]. Việc áp dụng các mạng neural tích chập (CNN) trong các thiết bị nhúng đã nhận được sự quan tâm đáng kể từ cộng đồng nghiên cứu và công nghiệp do khả năng của chúng trong việc xử lý và phân tích hình ảnh hiệu quả. Tuy nhiên, việc triển khai các mạng này trên thiết bị nhúng đặt ra những thách thức đặc biệt liên quan đến yêu cầu phần cứng, bao gồm cả về hiệu năng tính toán và bộ nhớ.

LPRNet không thực hiện phân đoạn ký tự trước, có thể được coi là một thuật toán nhận dạng biển số xe từ đầu đến cuối. Convolutional Neural Network đã chứng minh sự hiệu quả và ưu việt của mình trong các nhiệm vụ thị giác máy tính như phân loại hình ảnh, phát hiện đối tượng và phân đoạn ngữ nghĩa, nhiều thuật toán nhận dạng biển số xe LPR dựa trên mạng neural tích chập, và LPRNet cũng vậy. Khác với các thuật toán nhận dạng biển số xe khác, LPRNet nhắm đến các thiết bị nhúng.

2 Nghiên cứu liên quan

Các thuật toán nhận dạng biển số xe sớm được cấu thành từ hai giai đoạn: phân đoạn ký tự và phân loại ký tự.

- Phân đoạn ký tự thường sử dụng các thuật toán thủ công khác nhau, kết hợp chiếu sáng, kết nối và các thành phần dựa trên đường viền của hình ảnh. Do sử dụng hình ảnh nhị phân hoặc biểu diễn trung gian làm đầu vào, chất lượng phân đoạn ký tự bị ảnh hưởng nhiều bởi nhiễu trong hình ảnh đầu vào, độ phân giải thấp, mờ hoặc biến dạng.

- Giai đoạn phân loại ký tự sử dụng phương pháp OCR để nhận dạng ký tự trên biển số xe. Vì phân loại ký tự xảy ra sau phân đoạn ký tự, chất lượng nhận dạng phụ thuộc nhiều vào phương pháp phân đoạn. Để giải quyết vấn đề phân đoạn ký tự, tác giả của [2] đã áp dụng mạng neural tích chập từ đầu đến cuối, lấy toàn bộ hình ảnh biển số xe làm đầu vào và cuối cùng mạng sẽ xuất ra chuỗi ký tự.

Trong tài liệu [8], mô hình không phân đoạn dựa trên giải mã chuỗi độ dài biến đổi với CTC Loss. Sử dụng các đặc trưng LBP dựa trên hình ảnh nhị phân làm đầu vào cho CNN để tạo ra xác suất các lớp ký tự. Bằng cách áp dụng phương pháp cửa sổ trượt trên hình ảnh đầu vào, chuỗi đầu vào cho LSTM hai chiều được tạo ra. Do độ dài đầu ra của bộ giải mã và chuỗi ký tự mục tiêu khác nhau, mất mát CTC được sử dụng cho việc đào tạo từ đầu đến cuối không cần phân đoạn.

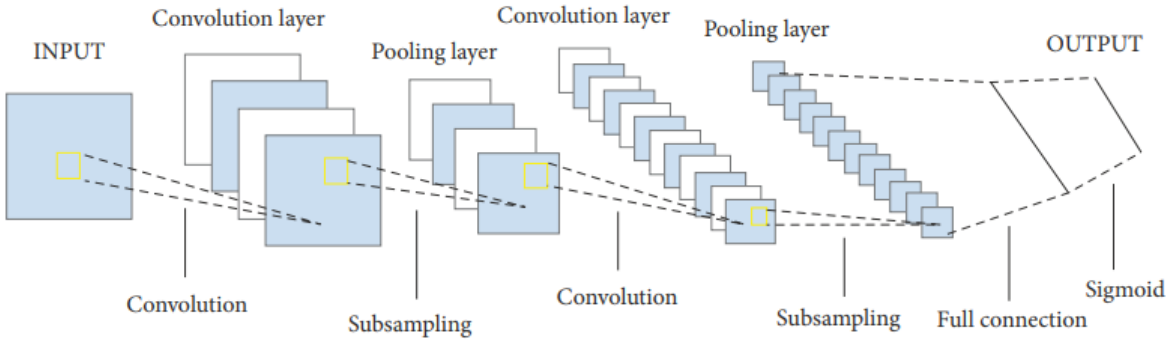
Mô hình trong tài liệu [9] chủ yếu tuân theo phương pháp được mô tả trong [8] ngoại trừ việc phương pháp cửa sổ trượt đã được thay thế bằng việc chia không

gian đầu ra của CNN thành chuỗi đầu vào RNN ("cửa sổ trượt" trên bản đồ đặc trưng thay vì đầu vào).

LPRNet không trích xuất các đặc trưng thủ công từ hình ảnh nhị phân mà sử dụng hình ảnh RGB gốc làm đầu vào cho CNN. LPRNet sử dụng mô hình toàn tích chập kết hợp với CTC Loss thay cho mô hình kết hợp LSTM và CTC, sử dụng CTC Loss để điều chỉnh mạng CNN trong giai đoạn đào tạo và sử dụng tìm kiếm tham lam (Greedy Search) và tìm kiếm tiền tố (Prefix Search) để hoàn thành suy diễn chuỗi trong giai đoạn suy diễn.

3 Phương pháp

3.1 Cấu trúc Convolutional Neural Networks (CNN)



Hình 1: Cấu trúc của Convolutional Neural Networks[10]

Cấu trúc cốt lõi của CNN bao gồm ba phần chính: tầng tích chập (convolutional layer), tầng gộp (pooling layer), và tầng kết nối đầy đủ (fully connected layer). Cấu trúc của mạng nơ-ron tích chập được thể hiện trong Hình 1, mỗi phần của mạng nơ-ron tích chập có thể là một lớp đơn hoặc nhiều lớp và có thể huấn luyện hình ảnh hoặc chuyển đổi dữ liệu dưới dạng hình ảnh. Lấy đầu vào dạng hình ảnh, chức năng chính của tầng tích chập là để trích xuất đặc trưng, chức năng của tầng gộp là để giảm lượng dữ liệu xử lý trong khi vẫn giữ thông tin hữu ích, và chức năng của tầng kết nối đầy đủ là để biến đổi bản đồ đặc trưng thành đầu ra cuối cùng mong muốn.

- *Tầng tích chập* Tầng tích chập thực hiện phép tích chập trên dữ liệu đầu vào để trích xuất bản đồ đặc trưng mới. Sau tầng tích chập, mạng nơ-ron không cần phải xử lý từng điểm dữ liệu mà là xử lý từng khối dữ liệu khu vực nhỏ. Qua sự di chuyển liên tục của kernel tích chập trên bản đồ đặc trưng đầu vào,

các bản đồ đặc trưng mới được thu được. Những bản đồ đặc trưng này liên tục được nén lại, nhưng độ sâu được tăng lên để có sự hiểu biết sâu sắc hơn về dữ liệu đầu vào. Trong xử lý ảnh, phép tích chập là tổng trọng số của mỗi khu vực bằng cách sử dụng kernel tích chập. Độ sâu biểu thị hình ảnh màu trong ảnh để thu được thông tin của ba màu RGB. Kernel tích chập di chuyển liên tục để thu được một bản đồ đặc trưng mới. Các kernel tích chập trong mạng nơ-ron tích chập là không xác định và có thể được điều chỉnh trong quá trình huấn luyện để thiết lập trọng số W. Phương pháp tính toán cho tầng tích chập được hiển thị theo công thức sau:

$$y'_j = \sum_{w_{ij} \in M_j} x_j^{l-1} \times w_{ij} + b_j \tag{1}$$

Trong đó, y'_j đại diện cho đầu ra của neural j trong lớp đầu tiên của mạng neural, M_j đại diện cho tập hợp tất cả các kernel tích chập, w đại diện cho kernel tích chập, x đại diện cho đầu vào, và b là sai lệch (bias).

- *Tầng gộp* Tầng gộp, còn được biết đến với tên gọi là tầng lấy mẫu xuống (downsampling layer), có thể thực hiện giảm kích thước không tuyến tính, mở rộng trường nhận thức, và trích xuất các đặc điểm thưa thớt của đầu vào. Công thức của tầng gộp được hiển thị theo công thức sau:

$$x_i = f(\beta_i \text{down}(x_i) + b_i). \tag{2}$$

Down đại diện cho các tham số lấy mẫu xuống, và các phương pháp gộp thường được sử dụng là hai cách để tối đa hóa việc gộp của khu vực địa phương của pooling kernel.

- *Tầng kết nối đầy đủ* Tầng kết nối đầy đủ (fully connected layer) còn được gọi là "bộ phân loại" trong toàn bộ mạng nơ-ron tích chập. Mỗi nơ-ron trong tầng kết nối đầy đủ được kết nối với các nơ-ron trong lớp trước đó. Tại tầng kết nối đầy đủ chỉ có phép nhân và không gian đặc trưng được biến đổi tuyến tính sang không gian đặc trưng khác để đạt được hiệu ứng phân loại.

CNN có đặc điểm là ít tham số mạng, trích xuất đặc trưng trừu tượng và receptive field. Bởi vì sự tồn tại của lớp tích chập, số lượng tham số của CNN được giảm đáng kể. Một đặc điểm khác của CNN là nó có một số lợi thế nhất định trong việc trích xuất các đặc trưng trừu tượng. Đối với các mạng nơ-ron thông thường, cần phải thiết kế các đặc trưng nhất định bằng cách thủ công, sau đó trích xuất các đặc trưng từ tập dữ liệu làm đầu vào của mạng. Bởi vì các đặc trưng được

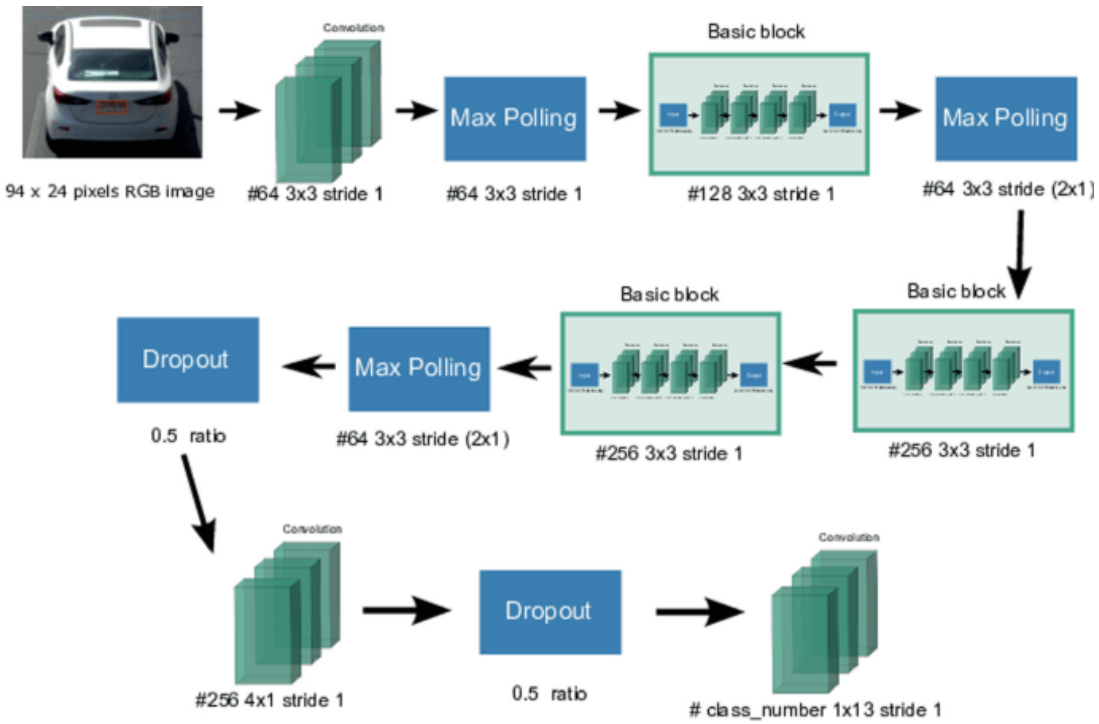
trích xuất có tính đại diện tốt hơn, hiệu suất của CNN cũng tốt hơn. Receptive field được định nghĩa là kích thước của vùng được ánh xạ bởi các pixel trên bản đồ đặc trưng được đầu ra bởi CNN tại mỗi lớp trên ảnh đầu vào, có thể phản ánh tốt mối tương quan giữa các đặc trưng cục bộ của bản đồ đặc trưng.

3.2 Kiến trúc LPRNet

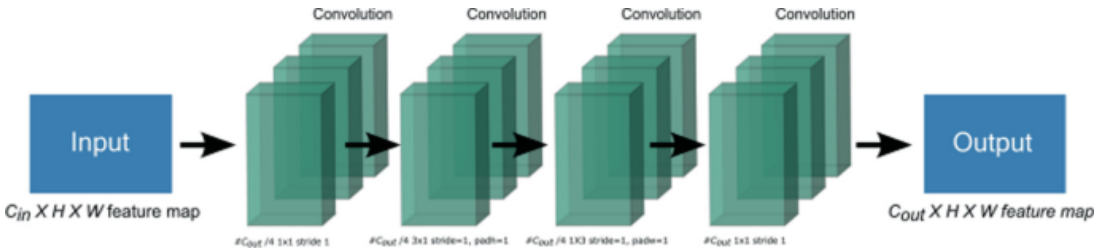
Trong các nghiên cứu gần đây, người ta có xu hướng sử dụng học chuyển giao để áp dụng các mạng phân loại mạnh mẽ như VGG, ResNet hoặc GoogLeNet làm mạng nền tảng cho các nhiệm vụ của họ. Tuy nhiên, để thiết kế một mạng có chi phí thấp, nhẹ nhàng, phương pháp này không phải là lựa chọn hàng đầu. Toàn bộ mạng của LPRNet có thể được tóm tắt như sau:

- Mạng định vị tùy Spatial Transformer Layer (tùy chọn)
- Mạng nền tảng CNN nhẹ (Backbone)
- Đầu phân loại ký tự cho mỗi vị trí
- Xác suất phân loại ký tự dùng cho giải mã chuỗi
- Quy trình lọc sau

Tác giả đầu tiên xử lý hình ảnh đầu vào bằng Spatial Transformer Nework (STN) được mô tả trong [11], quá trình xử lý là tùy chọn, trong phạm vi thực hiện của đề tài sẽ bỏ qua việc triển khai một Spatial Transformer Network mà tập trung vào LPRNet.



Hình 2: Sơ đồ kiến trúc backbone của LPRNet



Hình 3: Sơ đồ cấu trúc khối Small Basic (Basic Block)

Kiến trúc backbone của LPRNet được trình bày trong Hình 2 và Bảng 2, trong đó cấu trúc small_block được trình bày trong Hình 3 và Bảng 1. Mạng backbone lấy hình ảnh RGB gốc làm đầu vào, sử dụng CNN để trích xuất đặc trưng hình ảnh. Sử dụng kernel lọc 1x13 kết hợp ngữ cảnh thay cho RNN dựa trên LSTM. Đầu ra của mạng backbone có thể được coi là một chuỗi đại diện cho xác suất của các ký tự tương ứng, chiều dài của nó liên quan đến chiều rộng của hình ảnh đầu vào. Do đầu ra của mạng không tương đương về độ dài với số ký tự trên biển số, tác giả sử dụng CTC Loss không phân đoạn để đào tạo từ đầu đến cuối. CTC Loss thường được sử dụng trong các trường hợp đầu vào và đầu ra không căn chỉnh và có độ dài biến đổi. Ngoài ra, CTC chuyển đổi xác suất của từng bước thời gian thành xác suất của chuỗi đầu ra. Chi tiết về CTC Loss được trình bày trong tài liệu [12].

Để cải thiện hiệu suất, tác giả đã thêm global context embedding vào bản đồ đặc trưng trung gian của bộ giải mã dự báo. Điều này được thực hiện bằng

cách tính toán trên đầu ra của mạng nền tảng thông qua một lớp kết nối đầy đủ (fully-connected layer) và điều chỉnh kích thước để kết nối với đầu ra chính. Để điều chỉnh độ sâu của bản đồ đặc trưng phù hợp với số lượng các lớp ký tự, một lớp lọc 1x1 bổ sung được áp dụng. Trong giai đoạn suy diễn, tác giả sử dụng hai phương pháp giải mã: tìm kiếm tham lam (greedy search) và tìm kiếm chùm (beam search). Tìm kiếm tham lam chọn giá trị xác suất lớn nhất ở mỗi vị trí, trong khi tìm kiếm chùm tối đa hóa tổng giá trị nhỏ nhất.

Đối với quá trình lọc sau (post-filtering), tác giả sử dụng mô hình ngôn ngữ hướng tới nhiệm vụ như một tập hợp các mẫu LP mục tiêu quốc gia. Lưu ý rằng quá trình lọc sau được áp dụng cùng với tìm kiếm chùm. Quá trình lọc sau thu được các chuỗi có khả năng cao nhất tìm thấy bởi tìm kiếm chùm và trả về chuỗi đầu tiên phù hợp với bộ mẫu được định nghĩa trước phụ thuộc vào quy tắc biển số xe của quốc gia.

Bảng 1: Small Basic Block

Layer Type	**Parameters/Dimensions**
Input	$C_{in} \times H \times W$ feature map
Convolution	$\#C_{out}/4 \times 1 \times 1$ stride 1
Convolution	$\#C_{out}/4 \times 3 \times 1$ stride=1, padh=1
Convolution	$\#C_{out}/4 \times 1 \times 3$ stride=1, padw=1
Convolution	$\#C_{out} \times 1 \times 1$ stride 1
Output	$C_{out} \times H \times W$ feature map

Bảng 2: Cấu trúc Backbone của LPRNet

Layer Type	**Parameters**
Input	94x24 pixels RGB image
Convolution	#64 3x3 stride 1
MaxPooling	#64 3x3 stride 1
Small basic block	#128 3x3 stride 1
MaxPooling	#64 3x3 stride (2, 1)
Small basic block	#256 3x3 stride 1
Small basic block	#256 3x3 stride 1
MaxPooling	#64 3x3 stride (2, 1)
Dropout	0.5 ratio
Convolution	#256 4x1 stride 1
Dropout	0.5 ratio
Convolution	# class_number 1x13 stride 1

4 Triển khai thực hiện

- Ngôn ngữ lập trình : Python 3.10
- Các thư viện được sử dụng:

  pytorch >= 1.0.0

  opencv-python 3.x

  python 3.x

  imutils

  Pillow

  numpy

4.1 Chuẩn bị dữ liệu

Báo cáo này thực hiện huấn luyện và đánh giá mô hình với 2 tập dữ liệu:

- Tập dữ liệu Biển số xe Trung Quốc: Dữ liệu biển số xe Trung quốc được chọn lọc từ bộ dữ liệu CCPD2020 được cung cấp tại [1]. Bộ CCPD2020 có khoảng 11776 tấm ảnh biển số xe Trung Quốc được thu thập trên đường phố. Trong phạm vi đề tài này, chỉ chọn lọc những dữ liệu tốt và áp dụng một số kỹ thuật tăng cường dữ liệu thu được khoảng 11000 tấm ảnh, chia dữ liệu theo tỷ lệ 10000 tấm ảnh làm dữ liệu huấn luyện và 1000 tấm ảnh làm dữ liệu đánh giá.
- Tập dữ liệu Biển số xe Việt Nam: Dữ liệu biển số xe Việt Nam được thu thập thông qua nguồn ảnh chụp thực tế sau đó chọn lọc dữ liệu tốt tiến hành xử lý và tăng cường dữ liệu thu được tập dữ liệu biển số Việt Nam là 13320 tấm, trong đó 12320 tấm làm dữ liệu huấn luyện, 1000 tấm làm dữ liệu đánh giá.

Dữ liệu khi thu thập là ảnh thô, cần tiến hành các bước xử lý để đảm bảo định dạng đầu vào như được mô tả ở Bảng 2. Có nhiều phương pháp khác nhau để phát hiện vùng chứa biển số với các kỹ thuật tiên tiến để tách vùng chứa biển số theo kích thước yêu cầu như sử dụng các mô hình CNN như YOLO (You Only Look Once), SSD (Single Shot MultiBox Detector), hoặc Faster-RCNN; các phương pháp học máy như Random Forest hoặc SVM (Support Vector Machine); sử dụng phương pháp so khớp mẫu (Template Matching). Trong phạm vi báo cáo này sử dụng kỹ thuật dựa trên xử lý ảnh truyền thống đó Phát hiện cạnh (Edge Detection)bằng cách sử dụng các bộ lọc như Sobel hoặc Canny để tìm các cạnh trong ảnh, giúp phân biệt biển số xe với nền và phân tách màu sắc (Color Segmentation) để xử lý ảnh thô tạo dữ liệu theo yêu cầu của mô hình.

Nhãn của dữ liệu chính là dãy số ký tự trên biển số. Nhãn sẽ được gán thủ công. Định dạng của nhãn có dạng sau: *PlateNumber_ randomnumber.jpg* (đuôi mở rộng là tùy theo định dạng ảnh)

4.2 Triển khai mô hình LPRNet

Mô hình LPRNet được triển khai bằng với thư viện Pytorch. Cấu trúc khối Small Basic Block được trình bày ở Bảng 4

Bảng 3: Cấu trúc của small basic block triển khai bằng Pytorch

Thao tác	**Tham số**	**Mô tả**
conv2d	(ch_in, ch_out//4, kernel_size=1)	Chỉ thay đổi số lượng kênh đầu ra, H và W không thay đổi
relu		Kênh đầu ra, H và W không thay đổi
conv2d	(ch_out//4, ch_out//4, kernel_size=(3,1), padding=(1,0))	Kênh đầu ra, H và W không thay đổi, tích lũy thông tin pixel theo chiều cao tương ứng
relu		Kênh đầu ra, H và W không thay đổi
conv2d	(ch_out//4, ch_out//4, kernel_size=(1,3), padding=(0,1))	Kênh đầu ra, H và W không thay đổi, tích lũy thông tin pixel theo chiều rộng tương ứng
relu		Kênh đầu ra, H và W không thay đổi
conv2d	(ch_out//4, ch_out, kernel_size=1)	Chỉ thay đổi số lượng kênh đầu ra, H và W không thay đổi

Công thức tính toán kích thước của tích chập (convolution) là công thức giúp xác định kích thước đầu ra của một lớp tích chập dựa trên các tham số như kích thước đầu vào, kích thước bộ lọc, độ đệm (padding), và bước nhảy (stride) là:

$$\left\lfloor \frac{n + 2p - f}{s} \right\rfloor + 1 \tag{3}$$

Trong đó:

- Kích thước đầu vào (n): Đây là chiều cao hoặc chiều rộng của ảnh đầu vào hoặc lớp trước đó.
- Độ đệm (p): Độ đệm là số pixel thêm vào mỗi biên của ảnh đầu vào. Điều này có thể giúp bảo toàn kích thước không gian của đầu vào sau khi tích chập.

- Kích thước bộ lọc (f): Đây là kích thước của bộ lọc (thường là một giá trị như 3x3 hoặc 5x5).
- Bước nhảy (s): Bước nhảy là số lượng pixel mà bộ lọc dịch chuyển qua mỗi bước tích chập. Bước nhảy mặc định là 1, nghĩa là bộ lọc di chuyển 1 pixel mỗi lần.
- Hàm lấy phần nguyên ($\lfloor . \rfloor$): Hàm này lấy phần nguyên của giá trị bên trong, làm tròn xuống số nguyên gần nhất.

STT	Thao tác 1	Tham số	Đầu vào	Đầu ra	Thao tác 2	Tham số	Đầu vào	Đầu ra
0	Conv2d	(3, 64, kernel_size=(3,3), stride=(1,1))	[None, 3, 24, 94]	[None, 64, 22, 92]				
1	BatchNorm2d		[None, 64, 22, 92]	[None, 64, 22, 92]				
2	Relu		[None, 64, 22, 92]	[None, 64, 22, 92]	AvgPool2d	(kernel_size=5, stride=5)	[None, 64, 22, 92]	[None, 64, 4, 18]
3	MaxPool3d	(kernel_size=(1,3,3), stride=(1,1,1))	[None, 64, 22, 92]	[None, 64, 20, 90]				
4	small_basic_block		[None, 64, 20, 90]	[None, 128, 20, 90]				
5	BatchNorm2d		[None, 128, 20, 90]	[None, 128, 20, 90]				
6	Relu		[None, 128, 20, 90]	[None, 128, 20, 90]	AvgPool2d	(kernel_size=5, stride=5)	[None, 128, 20, 90]	[None, 128, 4, 18]
7	MaxPool3d	(kernel_size=(1,3,3), stride=(2,1,2))	[None, 128, 20, 90]	[None, 64, 18, 44]				
8	small_basic_block		[None, 64, 18, 44]	[None, 256, 18, 44]				
9	BatchNorm2d		[None, 256, 18, 44]	[None, 256, 18, 44]				
10	Relu		[None, 256, 18, 44]	[None, 256, 18, 44]				
11	small_basic_block		[None, 256, 18, 44]	[None, 256, 18, 44]				
12	BatchNorm2d		[None, 256, 18, 44]	[None, 256, 18, 44]				
13	Relu		[None, 256, 18, 44]	[None, 256, 18, 44]	AvgPool2d	(kernel_size=4,10), stride=(4,2))	[None, 256, 18, 44]	[None, 256, 4, 18]
14	MaxPool3d	(kernel_size=(1,3,3), stride=(4,1,2))	[None, 256, 18, 44]	[None, 64, 16, 21]				
15	DropOut	p=0.5	[None, 64, 16, 21]	[None, 64, 16, 21]				
16	Conv2d	(64, 256, kernel_size=(1,4), stride=(1,1))	[None, 64, 16, 21]	[None, 256, 16, 18]				
17	BatchNorm2d		[None, 256, 16, 18]	[None, 256, 16, 18]				
18	Relu		[None, 256, 16, 18]	[None, 256, 16, 18]				
19	DropOut	p=0.5	[None, 256, 16, 18]	[None, 256, 16, 18]				
20	Conv2d	(256, 68, kernel_size=(13,1), stride=(1,1))	[None, 256, 16, 18]	[None, 68, 4, 18]				
21	BatchNorm2d		[None, 68, 4, 18]	[None, 68, 4, 18]				
22	Relu		[None, 68, 4, 18]	[None, 68, 4, 18]				[None, 68,4,18]
					torch.cat	Thực hiện thao tác nối (concatenate) theo chiều thứ 2	[None, 68, 4, 18]	[None, 516, 4, 18]
					Conv2d	(516, 68, kernel_size=(1,1), stride=(1,1))	[None, 516, 4, 18]	[None, 68, 4, 18]
				Thao tác 3	torch.mean	Thực hiện thao tác tính trung bình (mean) theo chiều thứ 2.	[None, 68, 4, 18]	[None, 68, 18]

Hình 4: Cấu trúc LPRNet triển khai bằng Pytorch với bộ dữ liệu biển số xe Trung Quốc

Hình 4 mô tả cấu trúc mô hình triển khai bằng Pytorch, nguyên lý và thuật toán được trình bày trong [2], tác giả của [2] đã thực hiện nghiên cứu của mình với bộ dữ liệu biển số xe Trung Quốc. Trong phạm vi báo cáo này, việc huấn luyện và đánh giá mô hình sẽ được thực hiện với 2 bộ dữ liệu là biển số xe Trung Quốc và biển số xe Việt Nam được mô tả trong mục 4.1. Nguyên lý chung của mô hình như sau:

- **Thao tác 1:** Là cấu trúc xương sống (backbone) của mạng LPRNet. Xương sống của mạng bắt đầu bằng ảnh RGB, sử dụng CNN để trích xuất đặc trưng của ảnh. Vùng receptive field (1*13 kernel) tận dụng cấu trúc ký tự theo chiều dọc và ngang để thay thế mạng RNN dựa trên LSTM, bởi vì các ký tự trên biển số xe có trình tự. Xương sống của mạng có thể coi như là một chuỗi các xác suất ký tự tương ứng.
- **Thao tác 2:** Lấy đặc trưng từ các tầng 2, 6, 13, và sử dụng pooling để chuẩn hóa, sau đó tiến hành nối (concatenate) các đặc trưng theo chiều kênh. Thao

tác này nhằm cải thiện hiệu suất của mô hình, tăng cường thông tin không gian của các đặc trưng và sử dụng global context để nhúng. Chủ yếu lấy các đặc trưng từ các tầng 2, 6, 13, sau đó nối với các đặc trưng từ tầng xương sống theo chiều kênh.

- **Thao tác 3:** Sau khi lấy các đặc trưng từ tầng 2, sử dụng 'conv2d' để tính toán mean, lấy được dữ liệu cuối cùng với kích thước: [None, 68, 1, 18]. 68 là số lượng lớp ký tự trong bộ ký tự Trung Quốc còn với bộ ký tự Việt Nam là 37, 18 là chiều dài dự đoán. Lớp convolution này sử dụng kernel 1*1 để giữ nguyên chiều dữ liệu, chuyển đổi các đặc trưng thu được thành xác suất dự đoán. Đầu ra của model thiết kế để tính toán chuỗi ký tự dự đoán có độ dài bằng với chuỗi thực tế (với padding nếu cần). Cấu trúc này tuân theo nguyên lý CTC (Connectionist Temporal Classification) để đảm bảo mỗi ký tự thực tế trong chuỗi phải có ít nhất một vùng trống giữa các ký tự.

Đầu ra là ma trận xác suất, tương ứng với các tọa độ dọc là 68 phần tử, và các tọa độ ngang trong hình là 18 đoạn thời gian. Trong CTC, mỗi ký tự được thêm một khoảng trống phía trước và phía sau, do đó, độ dài của chuỗi đầu ra T nên được xem xét khi thiết kế mô hình nếu độ dài của chuỗi dài nhất được dự đoán (nhãn) là L thì $T \geq 2L + 1$. Sau đó, thông qua ma trận xác suất này và CTC loss, có thể huấn luyện chuỗi ký tự mà không cần sắp xếp trước

Nghĩa là ma trận xác suất là đầu ra cuối cùng của model sau khi qua các lớp convolution và các thao tác khác. Mỗi phần tử trong ma trận này đại diện cho xác suất của một ký tự tại một vị trí thời gian cụ thể.

- Tọa Độ Dọc (Vertical Axis): Đại diện cho 68 phần tử, mỗi phần tử tương ứng với một ký tự hoặc một nhãn trong bộ ký tự cần nhận dạng.

- Tọa Độ Ngang (Horizontal Axis): Đại diện cho 18 đoạn thời gian, mỗi đoạn thời gian tương ứng với một bước thời gian trong quá trình dự đoán.

CTC Loss

CTC Loss là một hàm mất mát (loss function) đặc biệt dùng cho các bài toán nhận dạng chuỗi, chẳng hạn như nhận dạng ký tự, nơi mà độ dài của chuỗi đầu ra không khớp với độ dài của chuỗi đầu vào.

- Khoảng Trống (Blanks): Trong CTC, mỗi ký tự được thêm một khoảng trống phía trước và phía sau để giúp model có thể xử lý các chuỗi có độ dài không đồng đều. Điều này có nghĩa là tổng số bước thời gian T phải lớn hơn hoặc bằng 2 lần số ký tự L cộng với 1, tức là $T \geq 2L + 1$.

Giá trị dự đoán đầu ra của mạng Lprnet được chỉnh sửa thành kích thước T×N×C. Sau đó, giá trị log_softmax (áp dụng softmax trước rồi lấy log) của giá trị này được sử dụng để tính toán CTC loss.

Trong đó:

- T: Số bước thời gian (time steps). Đây là số lượng đoạn thời gian mà mạng chia nhỏ để thực hiện dự đoán.
- N: Số mẫu trong batch (batch size). Đây là số lượng mẫu đầu vào được xử lý cùng một lúc trong một lần huấn luyện.
- C: Số lượng lớp ký tự (number of classes). Đây là số lượng các ký tự khác nhau mà mạng có thể dự đoán.

Log Softmax

- Softmax: Hàm softmax được áp dụng để chuyển đổi các giá trị đầu ra thành xác suất, sao cho tổng các xác suất này bằng 1.
- Log Softmax: Sau khi áp dụng softmax, log của giá trị này được lấy để dễ dàng tính toán loss. Log softmax thường được sử dụng trong CTC loss để tăng tính ổn định số học trong quá trình tính toán.

Thuật toán đánh giá thông qua Greedy Decode để tìm đường tối ưu

(1). Kích thước của mỗi chuỗi dữ liệu là 68×18. Trong đó, 68 là số lượng ký tự (các lớp ký tự) và 18 là số bước thời gian. Với dữ liệu biển số xe Việt Nam thì kích thước này là 37x18

(2). Với mỗi bước thời gian, tìm ký tự có xác suất cao nhất (argmax) trong số 68 ký tự. Kết quả là một chuỗi dự đoán gồm 18 ký tự.

(3). Quy tắc ánh xạ: hợp nhất các ký tự liên tiếp giống nhau và loại bỏ ký tự '-', tạo thành chuỗi ký tự cuối cùng. Ví dụ, từ chuỗi "a-aa" sẽ thành "a".

(4). So Sánh Kết Quả Dự Đoán Với Ký Tự Mục Tiêu:

TN_1: Tăng lên 1 nếu độ dài của chuỗi ký tự dự đoán không khớp với chuỗi ký tự mục tiêu.

TP: Tăng lên 1 nếu độ dài của chuỗi ký tự dự đoán khớp với chuỗi ký tự mục tiêu và nội dung cũng khớp.

TN_2: Tăng lên 1 nếu độ dài của chuỗi ký tự dự đoán khớp với chuỗi ký tự mục tiêu nhưng nội dung không khớp.

4.3 Huấn luyện mô hình

Quá trình huấn luyện mô hình được thực hiện trên môi trường Google Colab với cấu hình như tài nguyên như sau:

CPU: Intel(R) Xeon(R) CPU @ 2.00GHz

RAM: 12GB

GPU: Tesla T4 12GB VRAM

Bảng 4: Thiết lập thông số ban đầu để huấn luyện mô hình

Tham số	**Dữ liệu biển số xe Trung Quốc**	**Dữ liệu biển số xe Việt Nam**
max epoch	150	150
img size	[94, 24]	[94, 24]
dropout rate	0.5	0.5
learning rate	0.001	0.001
lpr max len	8	9
train batch size	32	32
test batch size	20	20
momentum	0.9	0.9
weight decay	2e-5	2e-5

4.4 Chỉ số đánh giá

- **Accuracy**

$$\text{Accuracy} = \frac{\text{TP}}{\text{TP} + \text{TN}_1 + \text{TN}_2} \tag{4}$$

  True Positive (TP) là số lần phân loại đúng biển số xe, True Negative (TN_1) đại diện cho việc phân loại sai khi độ dài của chuỗi giữa nhãn và dự đoán khác nhau, và (TN_2) đại diện cho việc phân loại sai khi độ dài chuỗi bằng nhau. Chỉ số này chỉ xem xét tỷ lệ biển số xe được phân loại hoàn toàn đúng trên tổng số biển số xe được phân loại đúng và sai. Tuy nhiên, có thể một biển số xe bị phân loại sai chứa nhiều hơn một ký tự sai, do đó sẽ sử dụng thêm chỉ số Levenshtein distance.

- **Levenshtein distance**: Levenshtein distance là một khái niệm trong khoa học máy tính và lý thuyết thông tin để đo lường sự khác biệt giữa hai chuỗi ký tự. Khoảng cách này được định nghĩa là số lượng thao tác chỉnh sửa nhỏ nhất cần thiết để biến đổi chuỗi này thành chuỗi kia. Các phép biến đổi này bao gồm:

Thay thế một ký tự (substitution)

Chèn thêm một ký tự (insertion)

Xóa một ký tự (deletion)

- **Thời gian xử lý**: Kiểm tra thời gian xử lý trung bình của 1 mẫu dữ liệu để kiểm tra hiệu suất hoạt động của mô hình

5 Kết quả thực hiện và đánh giá

5.1 Kết quả

Bảng 5 thể hiện kết của của các chỉ số đánh giá khi đánh giá mô hình trên từng bộ dữ liệu và kết quả khi thực hiện với Tesseract (mô hình train trước được cung cấp). Quá trình đánh giá mô hình được thực hiện trên môi trường Google Colab với cấu hình như tài nguyên tương tự quá trình huấn luyện như sau:

CPU: Intel(R) Xeon(R) CPU @ 2.00GHz

RAM: 12GB

GPU: Tesla T4 12GB VRAM

Bảng 5: Kết quả đánh giá mô hình với tập dữ liệu biển số xe Trung Quốc và tập dữ liệu biển số xe Việt Nam

Mô hình	**Tập dữ liệu**	**Mean Accuracy**	**Mean Levenshtein distance**	**Thời gian xử lý 1 mẫu**
LPRNet	Biển số xe Trung Quốc với 10000 tấm huấn luyện và 1000 tấm ảnh đánh giá	90%	0.045	9ms
LPRNet	Biển số xe Việt Nam với 12320 tấm ảnh huấn luyện và 1000 tấm ảnh đánh giá	75%	0.155	10ms
Tesseract	1000 tấm ảnh Biển số xe Việt Nam	84%	0.22	129.8ms

Hình 5, 6, 7 là kết quả dự đoán của 3 mẫu dữ liệu trong tập đánh giá của Bộ dữ liệu biển số xe Trung Quốc

Hình 5: Kết quả dự đoán mẫu 1

Hình 6: Kết quả dự đoán mẫu 2

Hình 7: Kết quả dự đoán mẫu 3

5.2 Đánh giá

Trong nghiên cứu này đã phát triển và đánh giá thành công mô hình LPRNet để nhận dạng biển số xe, thực hiện thử nghiệm trên hai bộ dữ liệu từ Trung Quốc và Việt Nam, và so sánh hiệu suất với mô hình nhận dạng biển số xe sử dụng Tesseract. Kết quả cho thấy sự khác biệt đáng kể về độ chính xác và hiệu quả xử lý giữa hai mô hình cũng như giữa hai bộ dữ liệu

5.2.1 Hiệu suất của LPRNet

Bộ dữ liệu Trung Quốc: Mô hình LPRNet đạt được độ chính xác ấn tượng là 90%, với khoảng cách Levenshtein trung bình là 0.045 và thời gian xử lý trung bình chỉ 9ms cho mỗi tấm ảnh. Điều này chỉ ra rằng mô hình đã được tối ưu hóa tốt cho bộ dữ liệu này.

Bộ dữ liệu Việt Nam: Độ chính xác giảm xuống còn 75% với khoảng cách Levenshtein là 0.155 và thời gian xử lý là 10ms mỗi ảnh. Sự sụt giảm này có thể là do các khác biệt về đặc điểm biển số xe và có thể bộ dữ liệu huấn luyện chưa đủ đa dạng, tối ưu, chất lượng dữ liệu chưa đủ tốt.

5.2.2 So sánh với mô hình Tesseract

Tesseract, khi được thử nghiệm trên cùng tập dữ liệu Việt Nam với 1000 tấm ảnh, cho thấy độ chính xác là 84% với khoảng cách Levenshtein là 0.22. Tuy nhiên, thời gian xử lý trung bình của Tesseract là 129.8ms mỗi ảnh, đáng kể lâu hơn so với LPRNet. Điều này cho thấy LPRNet tối ưu hơn đáng kể về mặt thời gian xử lý, mặc dù có độ chính xác thấp hơn một chút so với Tesseract trong trường hợp này. Với ảnh biển số xe Trung Quốc, Tesseract gặp khó khăn vì ký tự hán tự trong biển số do đó không thể kiểm tra so sánh LPRNet với Tesseract dựa trên đầu vào là biển số xe Trung Quốc.

6 Kết luận

Bài báo này đã trình bày kết quả nghiên cứu về hiệu suất của mô hình LPRNet trong việc nhận dạng biển số xe với 2 bộ dữ liệu Trung Quốc và Việt Nam, đồng thời so sánh với hiệu suất của mô hình nhận dạng biển số sử dụng Tesseract. Từ các kết quả thu được, chúng tôi nhận thấy rằng mô hình LPRNet có nhiều ưu điểm nổi bật như tốc độ xử lý nhanh, độ chính xác cao với dữ liệu Trung Quốc, và khả năng thích ứng tốt với các thay đổi nhỏ trong dữ liệu đầu vào. Tuy nhiên, mô hình

cũng bộc lộ một số hạn chế khi áp dụng với dữ liệu Việt Nam, điển hình là độ chính xác giảm sút.

6.1 Ưu điểm

Tốc độ xử lý nhanh: LPRNet cho thấy khả năng xử lý dữ liệu cực kỳ nhanh, chỉ khoảng 9ms đến 10ms cho mỗi ảnh, điều này là rất phù hợp cho các ứng dụng thời gian thực.

Độ chính xác cao với dữ liệu được huấn luyện khá tốt: Mô hình đạt được độ chính xác lên đến 90% với dữ liệu Trung Quốc, chứng tỏ khả năng hiệu quả khi có đủ dữ liệu huấn luyện phù hợp

6.2 Nhược điểm

Biến động độ chính xác giữa các bộ dữ liệu: Độ chính xác giảm xuống 75% đối với dữ liệu Việt Nam cho thấy mô hình còn thiếu sự ổn định và cần được tinh chỉnh khi áp dụng cho các bộ dữ liệu khác nhau.

Yêu cầu cao về dữ liệu huấn luyện: Hiệu suất của mô hình phụ thuộc nhiều vào chất lượng và số lượng dữ liệu huấn luyện, yêu cầu cao về việc chuẩn bị dữ liệu.

6.3 Hướng phát triển

Cải thiện độ chính xác trên dữ liệu đa dạng: Tăng cường số lượng và đa dạng hóa bộ dữ liệu huấn luyện để cải thiện độ chính xác của mô hình trên nhiều loại biển số xe khác nhau

Ứng dụng các kỹ thuật tăng cường dữ liệu: Áp dụng các phương pháp tăng cường dữ liệu như biến dạng hình ảnh, thay đổi độ sáng, và thêm nhiễu, để mô hình có thể học cách nhận dạng chính xác hơn trong các điều kiện khác nhau.

Nghiên cứu và phát triển các mô hình mới: Khám phá thêm các kiến trúc mạng nơ-ron sâu mới hoặc cải tiến LPRNet hiện tại để tối ưu hóa cả về mặt tốc độ lẫn độ chính xác. Kết hợp các phương pháp phát hiện biển số để hoàn thiện quy trình xử lý từ ảnh thực tế đến kết quả đầu ra.

Như vậy, mặc dù LPRNet đã thể hiện được tiềm năng ứng dụng cao, việc tiếp tục nghiên cứu và phát triển mô hình sẽ là chìa khóa để nâng cao hiệu quả sử dụng trong thực tế, đặc biệt là trong các ứng dụng đòi hỏi độ chính xác và tốc độ phản hồi nhanh.

7 Phụ lục

- **Toàn bộ dữ liệu và mã nguồn của báo cáo:** `https://github.com/lephamcong/PBL4_LPRNet.git`
- **Bộ dữ liệu biển số xe Trung Quốc:** `https://github.com/lephamcong/PBL4_LPRNet/tree/main/Dataset/BiensoxeTrungQuoc`
- **Bộ dữ liệu biển số xe Việt Nam:** `https://github.com/lephamcong/PBL4_LPRNet/tree/main/Dataset/BiensoxeVietNam`
- **Mã nguồn huấn luyện và đánh giá với bộ dữ liệu biển số xe Trung Quốc:** `https://github.com/lephamcong/PBL4_LPRNet/tree/main/LPRNet_Pytorch_China`
- **Mã nguồn huấn luyện và đánh giá với bộ dữ liệu biển số xe Việt Nam:** `https://github.com/lephamcong/PBL4_LPRNet/tree/main/LPRNet_Pytorch_VietNam`
- **Mã nguồn đánh giá so sánh với Tesseract:** `https://github.com/lephamcong/PBL4_LPRNet/tree/main/Tesseract_VietNam`

Tài liệu tham khảo

[1] Xu, Zhenbo, et al.“Towards end-to-end license plate detection and recognition: A large dataset and baseline." *Proceedings of the European conference on computer vision (ECCV).* 2018.

[2] Zherzdev, Sergey, and Alexey Gruzdev.“Lprnet: License plate recognition via deep neural networks.” *arXiv preprint arXiv:1806.10447* (2018).

[3] M. W. Lung, C. W. Chun, C. T. Wang, and Y. H. Lin, “Recycling waste classification using optimized convolutional neural network,” *Resources, Conservation and Recycling,* vol. 164, Article ID 105132, 2021

[4] S. T. Sara, M. M. Hasan, A. Ahmad, and S. Shatabda, “Convolutional neural networks with image representation of amino acid sequences for protein function prediction,” *Computational Biology and Chemistry,* vol. 92, Article ID 107494, 2021.

[5] N. Nahla, E. Mohammed, and V. Serestina, “Face expression recognition using convolution neural network (CNN) models” *International Journal of Grid Computing & Applications,* vol. 11, no. 4, pp. 1–11, 2020.

[6] Y. Sun, J. Xu, G. Lin, and N. Sun, "Adaptive neural network control for maglev vehicle systems with time-varying mass and external disturbance" *Neural Computing and Applications,* pp. 1–12, 2021

[7] Z. Cui, F. Xue, and X. Cai, "Detection of malicious code variants based on deep learning" *IEEE Transactions on Industrial Informatics,* vol. 14, no. 7, pp. 3187–3196, 2018.

[8] H. Li and C. Shen, "Reading Car License Plates Using Deep Convolutional Neural Networks and LSTMs," arXiv:1601.05610 [cs], Jan. 2016.

[9] T. K. Cheang, Y. S. Chong, and Y. H. Tay,"Segmentationfree Vehicle License Plate Recognition using ConvNetRNN," arXiv:1701.06439 [cs], Jan. 2017.

[10] Wang, Zhichao, et al. "Research and implementation of fast-LPRNet algorithm for license plate recognition." *Journal of Electrical and Computer Engineering 2021 (2021):* 1-11.

[11] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial Transformer Networks," *arXiv:1506.02025 [cs],* Jun. 2015, arXiv: 1506.02025.

[12] A. Hannun, "Sequence modeling with ctc," *Distill, 2017,* https://distill.pub/2017/ctc