|  |  |  |  |
| --- | --- | --- | --- |
| **Logo T3H**  **PHÒNG LẬP TRÌNH & MẠNG**  **TRUNG TÂM TIN HỌC**  **ĐẠI HỌC KHOA HỌC TỰ NHIÊN**  **⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯**  **BÁO CÁO ĐỒ ÁN**  **Chuyên viên Machine Learning (Applied Machine Learning Certificate)**  **Đồ án:**  **BRAND DETECTION**  Giáo viên hướng dẫn: **NGUYỄN QUAN LIÊM**  Học viên thực hiện:  **ĐỒNG ĐỨC HUY– 0949064506**    ***TP. Hồ Chí Minh, ngày … tháng … năm …*** | | | |
| ***NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN***  ***⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯***  **………………………………………………………………………...**  **………………………………………………………………………...**  **………………………………………………………………………...**  **………………………………………………………………………...**  **………………………………………………………………………...**  **………………………………………………………………………...**  **………………………………………………………………………...**  **………………………………………………………………………...**  **………………………………………………………………………...**  **Ngày …. tháng …. năm …**  **Giáo viên hướng dẫn**  *(ký và ghi rõ họ tên)* | |
| ***NHẬN XÉT CỦA GIÁO VIÊN pHẢN BIỆN***  ⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯  **………………………………………………………………………..**  **………………………………………………………………………..**  **………………………………………………………………………..**  **………………………………………………………………………..**  **………………………………………………………………………..**  **………………………………………………………………………..**  **………………………………………………………………………..**  **………………………………………………………………………..**  **………………………………………………………………………..**  **Ngày …. tháng …. năm …**  **Giáo viên phản biện**  *(ký và ghi rõ họ tên)* | |

Mục lục

[1 Tổng Quan 6](#_Toc51260181)

[1.1 Giới thiệu 6](#_Toc51260182)

[1.2 thực trạng và giải pháp 6](#_Toc51260183)

[1.3 yêu cầu: 6](#_Toc51260184)

[2. CHUẨN BỊ VÀ LÀM SẠCH DỮ LIỆU 7](#_Toc51260185)

[2.1 Lựa chọn dữ liệu huấn luyện (Data Source) 7](#_Toc51260186)

[2.2 Tiền xử lí hình ảnh 7](#_Toc51260187)

[3. XÂY DỰNG MODEL MMDETECTION ĐỂ XÁC ĐỊNH VỊ TRÍ LOGO TRONG HÌNH 8](#_Toc51260188)

[3.1 giới thiệu về mmdetection 8](#_Toc51260189)

[3.2 chuyển file về dạng format phù hợp cho mmdetection 9](#_Toc51260190)

[3.3 Chọn lựa detection frameworks cho mmdetection 9](#_Toc51260191)

[3.4 tham số cho MMDETECTION 10](#_Toc51260192)

[3.5 Kết quả 11](#_Toc51260193)

[3.6 Ví dụ 11](#_Toc51260194)

[4 XÂY DỰNG MODEL RESNET\_50 ĐỂ XÁC ĐỊNH BRAND TRONG HÌNH 12](#_Toc51260195)

[4.1 Giới thiệu về resnet 50 12](#_Toc51260196)

[4.2 Model summary 13](#_Toc51260197)

[4.3 Cách đánh giá kết quả 14](#_Toc51260198)

[4.4 Kết quả 15](#_Toc51260199)

[4.5 triển khai mô hình 17](#_Toc51260200)

[5. TRIỂN KHAI FULL MÔ HÌNH MMDETECTION VÀ RESNET\_50 18](#_Toc51260201)

[5.1 scheme 18](#_Toc51260202)

[5.2 Kiểm tra full mô hình bằng custom dataset 18](#_Toc51260203)

[6. KẾT LUẬN 18](#_Toc51260204)

[7. XEM XÉT KHẢ NĂNG SỬ DỤNG TRÊN SMARTPHONE 19](#_Toc51260205)

[8. ĐỀ XUẤT CẢI TIẾN 19](#_Toc51260206)

[9. TRÍCH DẪN 19](#_Toc51260207)

***GIỚI THIỆU ĐỒ ÁN***

⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯

\_ Phát hiện thương hiệu/Logo là một phương pháp xác minh tự động có thể giải phóng một phần lực lượng con người khỏi quá trình xác minh thủ công. Bất cứ khi nào người bán đăng yêu cầu, các hình ảnh liên quan sẽ được chuyển tiếp đến hệ thống xác minh và sẽ có thể dự đoán thương hiệu.

\_ Vai trò chính của model là cung cấp thông tin sớm để từ chối các yêu cầu vi phạm hình ảnh có chứa logo thương hiệu không khớp với thương hiệu của sản phẩm.

# 1 Tổng Quan

## 1.1 Giới thiệu

*\*Giới thiệu đồ án:*

\_ Phát hiện thương hiệu/Logo là một phương pháp xác minh tự động có thể giải phóng một phần lực lượng con người khỏi quá trình xác minh thủ công. Bất cứ khi nào người bán đăng yêu cầu, các hình ảnh liên quan sẽ được chuyển tiếp đến hệ thống xác minh và sẽ có thể dự đoán thương hiệu.

\_ Với hàng chục nghìn cho tới hàng trăm nghìn hình ảnh mỗi ngày hiện nay thì việc phân loại các bình luận từ người dùng không phải là điều dễ dàng, và đòi hỏi rất nhiều nhân lực. Vì vậy ta cần một phướng án thích hợp đề có thể tự động hóa quá trình phân biệt các nhãn hành khác nhau

\_Cùng sự phát triển của AI và các phương pháp ML, hiện nay chúng ta có thể giải quyết bài toán này bằng các mô hình Deep Learning với độ chính xác có thể sánh ngang với khả năng đọc hiểu của con người.

*\* phạm vi đồ án:*

*\_*  Vai trò chính của model là cung cấp thông tin sớm để từ chối các yêu cầu vi phạm. Hình ảnh có chứa logo thương hiệu không khớp với thương hiệu của sản phẩm.

## 1.2 thực trạng và giải pháp

\* thực trạng:

Với hàng chục nghìn cho tới hàng trăm nghìn hình ảnh mỗi ngày hiện nay thì việc phân loại các bình luận từ người dùng không phải là điều dễ dàng, và đòi hỏi rất nhiều nhân lực.

\*giải pháp:

Ta sẽ dùng mãng mmdetection để xác định xem trong hình có logo hay là không. Nếu trong hình có logo. Mmdetection sẽ nhận diện vị trí của logo và cắt ra hình mới là hình logo. Sau đó ta sẽ sử dụng resnet để xác đinh xem hình logo thuộc vế hãng nào.

Cách đánh giá mô hình: Mô hình mmdetection được huấn luyện và đánh giá trên tập dữ liệu gồm 1000 hình ảnh gắn nhãn logo với độ đo mAP. Resnet được huấn luyện và đánh giá trên tập dữ liệu gồm 1000 hình ảnh với độ đo là mAP

## 1.3 yêu cầu:

- Yêu cầu 1: Thực hiện fine tune model mmdetection dựa trên gói datasets Flickr 27 gồm 27 nhãn hàng có sẵn để giúp model có thể xác định vị trí của logo trong sản phẩm

- Yêu cầu 2: Tiến hành crawl hình các sản phẩm từ tiki, shopee, lazada

- Yêu cầu 3: Lựa chọn các hình ảnh chứa logo

- Yêu cầu 4: Cho các hình ảnh chạy qua mmdetection model đã finetune để xác định logo và crop logo ra thành ảnh nhỏ

- Yêu cầu 5: Xây dựng model bằng cách finetune ResNet50 để dự đoán sản phẩm ứng với thương hiệu/Logo nào.

- Yêu cầu 6: Xây dựng UI để cho 1 hình ảnh sản phẩm chứa logo, model sẽ dự đoán xem sản phẩm này thuộc thương hiệu nào.

# 2. CHUẨN BỊ VÀ LÀM SẠCH DỮ LIỆU

## 2.1 Lựa chọn dữ liệu huấn luyện (Data Source)

*Dữ liệu huấn liệu cho mmdetection:*

\_Bộ dữ liệu flick27 được dùng cho việc huấn luyện model mmdetection (nguồn: <http://image.ntua.gr/iva/datasets/flickr_logos/>) .Dataset chia làm 3 set: Training set: 809 hình sản phẩm với logo từ 27 hãng dùng. Distractor set: 4207 hình logo từ trang web với logo design .Query set: 270 hình. 135 hình có logo từ 27 hãng và 135 hình không có logo.

\_Tổng cộng 27 hãng nhưng trong bài bào cáo này chỉ sử dụng 1 nhãn duy nhất là logo cho tất cả các hãng.

*Dữ liệu huấn liệu cho resnet:*

\_Bộ dữ liệu là hình ảnh được lấy từ các trang buôn bán online, google và shutter stock

\_12000 hình từ tiki được lấy bằng cách dùng package BeautifulSoup.

\_4000 hình từ goolge bằng add-on Download All Images trên google

\_4000 hình từ shutterstock.com bằng package scrappy

\_300 hình tải trực tiếp từ các trang web khác nhau bằng snipping tool

\_Bộ dữ liệu bao gồm 20000 hình thuộc về 30 hãng khác nhau.

## 2.2 Tiền xử lí hình ảnh

*Mmdetection:*

\_Bộ dataset được chuyển về 1 nhãn duy nhất là label

\_tổng cộng có 809 hình có logo, số lượng logo trong hình là 1260 hình.

\_Size hình phổ biến nhất là (500,375) với số lượng là 272, đứng thứ hai là (375,500) với số lượng là 121.

\_Chất lượng hình trung bình.

\_ Bao gồm 1 File txt chứa dữ liệu bounding box

*Resenet\_50:*

\_20500 hình ảnh các sản phẩm từ 30 hãng khác nhau đươc lấy từ các trang web shoopee, tiki, google và shutterstock image sẽ trải qua tiền xử lí hình ảnh. Các hình quá nhỏ, mờ, sẽ bị loại bỏ. Sau quá trình này còn lại 10000 hình.

\_mmdetection sẽ chạy qua cái hình ảnh được chấp nhận và cắt logo từ các hình đó ra. Size yêu cầu của logo là (32,32) và nếu logo nhỏ hơn thì sẽ bọ loại bỏ. Sau quá trình này còn lại 500 logo.

\_Logo được chấp nhận sẽ được resize về (32,32) sau đó sẽ trải qua Shifting position, rotate, resizing để tăng số lượng hình cho training.

\_Sau khi trải qua tiền xử lí, dataset gồm 10 class, size hình là (32,32), mỗi class chứa 7000 hình cho training và 700 cho testing:

1. Apple
2. BMW
3. Heineken
4. HP
5. Intel
6. Mini
7. Starbucks
8. Vodafone
9. Unknown
10. Ferrari

# 3. XÂY DỰNG MODEL MMDETECTION ĐỂ XÁC ĐỊNH VỊ TRÍ LOGO TRONG HÌNH

## 3.1 giới thiệu về mmdetection

MMDetection là một công cụ phát hiện đối tượng mã nguồn mở dựa trên PyTorch. Nó là một phần của dự án OpenMMLab do Phòng thí nghiệm Đa phương tiện, CUHK phát triển.  
Các tính năng chính  
• Thiết kế mô-đun  
MMDetection phân tách detection frameworks thành các thành phần khác nhau và người dùng có thể dễ dàng xây dựng detection frameworks tùy chỉnh bằng cách kết hợp các mô-đun khác nhau.  
• Hỗ trợ nhiều frameworks

Hộp công cụ hỗ trợ trực tiếp các detection frameworks phổ biến và hiện đại, ví dụ: Faster RCNN, Mask RCNN, RetinaNet, v.v.  
• Hiệu quả cao  
Tất cả các hoạt động bbox và mặt nạ cơ bản đều chạy trên GPU. Tốc độ đào tạo nhanh hơn hoặc có thể so sánh với các cơ sở mã khác, bao gồm Detectron2, maskrcnn-benchmark và SimpleDet.  
• Hiện đại nhất  
Hộp công cụ bắt nguồn từ cơ sở mã được phát triển bởi nhóm MMDet, người đã chiến thắng Thử thách phát hiện COCO vào năm 2018 và MMDetection tiếp tục thúc đẩy nó về phía trước.  
Nhánh chính hoạt động với PyTorch 1.3 đến 1.6. Nhánh v1.x cũ hoạt động với PyTorch 1.1 đến 1.4, nhưng v2.0 được khuyên dùng để có tốc độ nhanh hơn, hiệu suất cao hơn, thiết kế đẹp hơn và cách sử dụng thân thiện hơn.

## 3.2 chuyển file về dạng format phù hợp cho mmdetection

\_ Trong file dataset flick27 bao gồm file txt chứa thông tin vị trí các bounding box và tên nhãn của chúng. File được lưu dưới dạng custom nên cần được chuyển về Middle format như trong figure 1 để dủng cho mmdetection.

A screenshot of a cell phone

Description automatically generated

Figure 1. Middle format for mmdetection

## 3.3 Chọn lựa detection frameworks cho mmdetection

Chọn Faster R-CNN

A close up of a piece of paper

Description automatically generated

Figure 2 FASTER R CNN

## A screenshot of a cell phone Description automatically generated3.4 tham số cho MMDETECTION

Figure 3 HYPER PARAMETER FOR MMDETECTION

optimizer = dict(type='SGD', lr=0.0025, momentum=0.9, weight\_decay=0.0001)

optimizer\_config = dict(grad\_clip=None)

lr\_config = dict(

policy='step',

warmup=None,

warmup\_iters=500,

warmup\_ratio=0.001,

step=[8, 11])

total\_epochs = 12

## 3.5 Kết quả

acc: 99.3652, loss\_bbox: 0.0339, loss: 0.0588

![A screenshot of a cell phone

Description automatically generated]()

## 3.6 Ví dụ

Mmdetection phân tích hình ảnh đưa vào và trả về vị trí bounding với class là cartegory và xác suất của vị trí logo đó.

# A picture containing container, can, many, table Description automatically generated4 XÂY DỰNG MODEL RESNET\_50 ĐỂ XÁC ĐỊNH BRAND TRONG HÌNH

## 4.1 Giới thiệu về resnet 50

Quá trình đào tạo mô hình CNN sâu thường đòi hỏi chi phí tính toán cao và tăng dần khi mạng tăng độ sâu. Thông thường, vấn đề nảy sinh trong mạng neuron sâu là lỗi backpropagation nhận giá trị gradient nhỏ hơn khi tiếp cận các layer cấp thấp. Sự biến mất gradient phát sinh khi mạng giới thiệu quá nhiều lớp tính năng ánh xạ phi tuyến tính. Các layer xếp chồng lên nhau này làm giảm gradient cực kỳ nhỏ (gần bằng không) khi nó đạt đến các layer đầu tiên. Về lý thuyết, những vectơ gradient này vẫn có thể được nhận ra. Tuy nhiên, trong thực tế, khi các số thực được khai báo chủ yếu ở dạng 32bit hoặc 64bit (FLOAT32 hoặc FLOAT64), các dấu phẩy động vẫn có thể được xấp xỉ trở thành 0 khi nó quá nhỏ. Điều này dẫn đến tất cả các giá trị tiếp theo trở thành 0 hoặc biến mất gradient.

Mạng lưới ResNet đã khai thác ý tưởng về các đường tránh được sử dụng trong Highway Networks. Lớp nhận dạng hoặc "short-cut" giúp giải quyết sự biến mất của gradient bằng cách cho phép các giá trị gradient chảy trực tiếp đến các layer trên trong back-propagation. Ý tưởng này không mới; trên thực tế, đã có nhiều thiết kế mô hình như " shortcut connection", “highway " hoặc thêm các layer tuyến tính được kết nối với các bộ phân loại phụ trợ để giải quyết độ dốc biến mất / bùng nổ.

ResNet có năm phiên bản (ResNet-18, 34, 50, 101 và 152) tùy thuộc vào độ sâu của convolution layer. Trong nghiên cứu này, chúng tôi sử dụng kiến trúc ResNet 50 cho nhiệm vụ phân loại. Ví dụ minh họa về kiến trúc học sâu dựa trên ResNet50 được hiển thị trong Hình 3. ResNet50 có tổng cộng 50 lớp và cấu hình của nó như sau . Bộ này bao gồm một convolution layer, batch normalization và ReLU (activation function) với 49 lớp và một lớp cuối cùng fully connected layer. Convolution layer dầu tiên được đặt thành 7x7 kernel với stride 2 và padding 3, và kernel size của tất cả các lớp convolution layers tiếp theo là 3x3.

A screenshot of a computer

Description automatically generated

## 4.2 Model summary

Chúng tôi đã sử dụng TensorFlow (phiên bản 2.2.0, https://tensorflow.org/) làm thư viện học sâu để đào tạo, xác thực và kiểm tra CNN. Trong công việc này, chúng tôi triển khai mạng transfer learning, đã được đào tạo trước trên cơ sở dữ liệu ImageNet quy mô lớn đã được thiết lập tốt để phân loại ảnh. Tuy nhiên, để được điều chỉnh cho ứng dụng của chúng tôi, chúng tôi định cấu hình lại fully connected last layer.

Các tham số của các lớp tích chập trong mạng được huấn luyện trước được sử dụng làm giá trị ban đầu. Các trọng số trong mạng học sâu được đào tạo trước đã học tốt thông qua lượng lớn hình ảnh trong bộ dữ liệu ImageNet . Do đó, việc học chuyển giao bằng cách sử dụng các trọng lượng đã được đào tạo trước được kỳ vọng sẽ học nhanh hơn so với scratched networks và hứa hẹn sẽ có hiệu suất tốt hơn. Để tăng số lượng hình ảnh cho quá trình đào tạo và có đủ độ mạnh, chúng tôi áp dụng tăng cường dữ liệu trực tuyến mà mỗi hình ảnh đào tạo được thay đổi kích thước để tạo ra một hình ảnh đào tạo mới.

![A screenshot of a social media post

Description automatically generated]()

## 4.3 Cách đánh giá kết quả

Để đánh giá hiệu suất của bộ phân loại, chúng tôi báo cáo Accuracy (ACC), Precision (PRE) và Recall (REC). TP, TN, FP và FN lần lượt cho biết số lượng true positives, true negatives, false positives, và false negatives. Xét về những con số này, ACC, PRE và REC có thể được tính như sau:

ACC = (TP + TN) / (TP + TN + FP + FN) (1)  
PRE = TP / (TP + FP) (2)  
REC = TP / (TP + FN) (3)

Để giảm thiểu ảnh hưởng của việc khởi tạo trọng số ngẫu nhiên và tối đa hóa độ tin cậy của kết quả, chúng tôi thực hiện huấn luyện 24 lần. Từ kết quả, means và standard errors của kết quả phân loại được báo cáo.

## A close up of a map Description automatically generatedA screenshot of a social media post Description automatically generated4.4 Kết quả

![A screenshot of a social media post

Description automatically generated]()

![A screenshot of a social media post

Description automatically generated]()

## 4.5 triển khai mô hình

A screenshot of a cell phone

Description automatically generatedCho hình có logo thuộc 1 trong 9 hãng sẽ trả về tên hãng cho từng logo

Cho hình thuộc về hãng không nằm trong những hãng đã học

A picture containing cat, shirt

Description automatically generated

# 5. TRIỂN KHAI FULL MÔ HÌNH MMDETECTION VÀ RESNET\_50

## A picture containing screenshot Description automatically generated5.1 scheme

## 5.2 Kiểm tra full mô hình bằng custom dataset

Custom dataset được tạo ra bằng cách lấy hình ảnh sản phẩm bất kì từ google. Tổng cộng 50 hình, gồm nhiều size và nhiều solution khác nhau

Custom dataset:

1. Hình ảnh chất lượng thấp: 5 hình
2. Hình ảnh chứa logo của hãng được huấn luyện: 20 hình
3. Hình ảnh chứa logo từ các hãng không được huấn luyện: 15 hình
4. Hình ảnh chứa logo nhỏ: 5 hình

Kết quả: 70%

# 6. KẾT LUẬN

Tốc độ của full model: 10s

Độ nặng của model: mmdetection: 158 mb, resnet\_50: 883 mb

Độ chính xác của full model: 70%

# 7. XEM XÉT KHẢ NĂNG SỬ DỤNG TRÊN SMARTPHONE

Một trong những hướng phát triển khải thi là thay vì sử dụng model resnet\_50, ta có thể sử dụng MobileNetV1. Bằng cách này thay vì sử dụng model với kích thước >100 mb, ta có thể sử dụng một model với kích thước <20 mb. Vấn đề trong model này là model mmdetection có kích thước rất lớn > 100 mb khiến cho bộ full model luôn lớn hơn 100 mb không thích hợp cho smart phone. Trước khi có một object detection model thay thế được cho mmdetection thì không có khả năng đưa lên smart phone.

# 8. ĐỀ XUẤT CẢI TIẾN

Full mô hình cỏn đạt kết quả thấp, dataset cho training và testing khi đưa vào cần chất lượng cao hơn.

Có thể phát triển lên thành ứng dụng sử dụng trong thời gian thực.

# 9. TRÍCH DẪN

[1] [K. Albion, L. Briens, C. Briens, and F. Berruti, “Detection of the breakage of pharmaceutical tablets in pneumatic transport,” *Int. J. Pharm.*, vol. 322, no. 1–2, Sep. 2006.](http://paperpile.com/b/lQFcuE/PVYA)

[2] [A. H. Sabri, C. N. Hallam, N. A. Baker, D. S. Murphy, and I. P. Gabbott, “Understanding tablet defects in commercial manufacture and transfer,” *Journal of Drug Delivery Science and Technology*, vol. 46. pp. 1–6, 2018, doi:](http://paperpile.com/b/lQFcuE/I7Wp) [10.1016/j.jddst.2018.04.020](http://dx.doi.org/10.1016/j.jddst.2018.04.020)[.](http://paperpile.com/b/lQFcuE/I7Wp)

[3] [M. Možina, D. Tomaževič, F. Pernuš, and B. Likar, “Automated visual inspection of imprint quality of pharmaceutical tablets,” *Machine Vision and Applications*, vol. 24, no. 1. pp. 63–73, 2013, doi:](http://paperpile.com/b/lQFcuE/WwuH) [10.1007/s00138-011-0366-4](http://dx.doi.org/10.1007/s00138-011-0366-4)[.](http://paperpile.com/b/lQFcuE/WwuH)

# 10. Code

# -\*- coding: utf-8 -\*-

"""phase 1: image preprocessing.ipynb

Automatically generated by Colaboratory.

Original file is located at

    https://colab.research.google.com/drive/1KOzQvnGTILUD9pFmWH87C96dZQCBWt3F

# import library

"""

from google.colab import drive

drive.mount('/content/gdrive', force\_remount=True)

# Commented out IPython magic to ensure Python compatibility.

# %cd "/content/gdrive/My Drive/flick\_dataset\_preprocessing"

import pandas as pd

import numpy as np

import cv2

import os

import json

import glob

import shutil

import os

"""# Đọc dữ liệu từ  flickr\_logos\_27\_dataset\_training\_set\_annotation.txt"""

# đọc dữ liệu từ flie text

dataset = pd.read\_csv("flickr\_logos\_27\_dataset\_training\_set\_annotation.txt",header= None, sep=" ")

# đổi tên các cột sang tên tương ứng

flick = dataset.rename(columns={0:"file\_name",1:"categories\_names",2: "Training subset",3:"xmin",4:"ymin",5:"xmax",6:"ymax"})

# bỏ cột 7 do là cột trống

flick = flick.drop([7], axis=1)

flick

"""Giải thích ý nghĩa các cột:

\*   file\_name: tên hình

\*   categories\_names: tên nhãn của logo

\*   Training subset: Training subset

\*   xmin  ymin  xmax  ymax: Coordinates của 2 điểm để tạo bounding box

"""

# sort lại hình ảnh

flick = flick.sort\_values(by=['file\_name']).reset\_index(drop=True)

#thêm vào 2 cột width và height của hình ảnh

flick["width"] = 0

flick["height"] = 0

path = "flickr\_logos\_27\_dataset\_images"

# thêm vào width và height cho dataset

for i in range(len(flick)):

  im = cv2.imread(os.path.join(path,flick['file\_name'][i]))

  h, w, c = im.shape

  flick["width"][i] = w

  flick["height"][i] = h

# tạo một cột mới chứa width và height của hình ảnh

flick['image\_shape'] = list(zip(flick.width, flick.height))

# dataset sau khi trải qua xử lý

flick

#save flick

flick.to\_csv(r'/content/gdrive/My Drive/flick\_dataset\_preprocessing/flick.txt', index=None, sep='\t', mode='a')

"""# phân tích dataset sau preprocressing"""

print("số lượng logo: ", len(flick) )

print("số lượng hình: ", flick["file\_name"].nunique())

print("sô 1ượng brands: ",flick["categories\_names"].nunique())

print("số lượng hình thuộc từng brand")

counts = flick["file\_name"].value\_counts()

flick["categories\_names"].value\_counts()

flick["image\_shape"].value\_counts()

"""# tạo bộ dataset flick dựa trên thông tin từ flick dataframe"""

# vì bộ flickr\_logos\_27\_dataset\_images chứa những hình không được đánh dấu, ta cần tạo một bộ dataset mới chỉ có hình ảnh nằm trong flick

# đọc tất cả các tên đặt biệt từ flick

# chuyển tất cả các file ảnh từ dataset flickr\_logos\_27\_dataset\_images về một folder mới

src\_dir = "/content/gdrive/My Drive/flick\_dataset\_preprocessing/flickr\_logos\_27\_dataset\_images"

dst\_dir = "/content/gdrive/My Drive/mmdetection/flick/training/image\_2"

uquine\_name = flick["file\_name"].unique()

for  i in range(len(uquine\_name)):

    name = uquine\_name[i]

    jpgfile = os.path.join(src\_dir,name)

    shutil.copy(jpgfile, dst\_dir)

"""# chuyển dữ liệu từ flickr\_logos\_27\_dataset\_training\_set\_annotation.txt về dạng custom format cho mmdetection"""

#  đọc dữ liệu từ flick và chuyển về dạng format thích hợp cho mmdetection

dst\_dir2 = "/content/gdrive/My Drive/mmdetection/flick/training/label\_2"

for  i in range(len(uquine\_name)):

  name = uquine\_name[i]

  df3 = flick[flick.file\_name == name].reset\_index(drop = True)

  text\_name = name[:-4]+ ".txt"

  text\_name\_dir = os.path.join(dst\_dir2,text\_name)

  with open(text\_name\_dir, "w") as text\_file:

      for  j in range(len(df3)):

        label = "Car"

        x1 = str(df3["xmin"][j])

        y1 = str(df3["ymin"][j])

        x2 = str(df3["xmax"][j])

        y2 = str(df3["ymax"][j])

        strings = label + " " + x1 + " " + y1 + " " + x2 + " " + y2

        text\_file.write(strings)

        text\_file.write("\n")

"""# convert sang middle format cho mmdetection"""

# đọc dữ liệu từ flick dataframe và chuyển về dạng middle format để sử sụng cho mmdetection

df2 = flick

i=0

uquine\_name = flick["file\_name"].unique()

data\_infos = []

for i in range(len(uquine\_name)): # loop qua 809 hình trong training

  #tạo list rỗng đễ chứa tọa độ của bounding box và label

  bboxes = []

  labels = []

  bboxes\_ignore = []

  labels\_ignore = []

  #lấy ra tên từng hình

  name = uquine\_name[i]

  #dùng tên hình tạo dataframe mới chỉ chứa hình đó

  df3 = df2[df2.file\_name == name].reset\_index(drop = True)

  # lấy ra shape của hình

  w = df3["width"][0]

  h = df3["height"][0]

  #loop qua tất cả các boundind box và label của boundind box trong hình

  #bỏ vào trong 2 list rỗng bboxes và labels

  for j in range(len(df3)):

    x1 = df3["xmin"][j]

    y1 = df3["ymin"][j]

    x2 = df3["xmax"][j]

    y2 = df3["ymax"][j]

    bboxes.append([x1,y1,x2,y2])

    labels.append(1)

  # chuyển về np.array theo yêu cầu của format

  bboxes=np.array(bboxes).astype(np.float32)

  labels=np.array(labels).astype(np.int64)

  bboxes\_ignore=np.array(bboxes\_ignore).astype(np.float32)

  labels\_ignore=np.array(labels\_ignore).astype(np.int64)

  #tạo một list mới theo format

  data = [

    {

        'filename':name,

        'width': w,

        'height': h,

        'ann': {

            'bboxes': bboxes,

            'labels': labels,

            'bboxes\_ignore': bboxes\_ignore,

            'labels\_ignore': labels\_ignore

        }

    }

  ]

  #thêm lost đó vào list chung

  data\_infos.append(data)

"""# extract to json format"""

# tạo class MyEncoder để tránh lỗi

class MyEncoder(json.JSONEncoder):

    def default(self, obj):

        if isinstance(obj, np.integer):

            return int(obj)

        elif isinstance(obj, np.floating):

            return float(obj)

        elif isinstance(obj, np.ndarray):

            return obj.tolist()

        else:

            return super(MyEncoder, self).default(obj)

# extract ra json format và kiểm tra

jstr = json.dumps(data\_infos, indent=4,cls=MyEncoder)

print(jstr)

# save về một file mới để sử dụng mmdetection

with open('data\_middle\_format\_2.json', 'w') as outfile:

    json.dump(data\_infos, outfile,cls=MyEncoder)

# -\*- coding: utf-8 -\*-

"""phase 1: mmdetection.ipynb

Automatically generated by Colaboratory.

Original file is located at

    https://colab.research.google.com/drive/1iVWufV1YflfiiruPiStcyDl-XmBJ6Svn

"""

"""# import library"""

from google.colab import drive

drive.mount('/content/drive')

# Check nvcc version

!nvcc -V

# Check GCC version

!gcc --version

# Commented out IPython magic to ensure Python compatibility.

# %cd "/content/drive/My Drive"

# Commented out IPython magic to ensure Python compatibility.

# download thư viện mmcv-full cho mmdetection và donwload mmdetection

# install dependencies: (use cu101 because colab has CUDA 10.1)

!pip install -U torch==1.5.1+cu101 torchvision==0.6.1+cu101 -f https://download.pytorch.org/whl/torch\_stable.html

# install mmcv-full thus we could use CUDA operators

!pip install mmcv-full

# Install mmdetection

!rm -rf mmdetection

!git clone https://github.com/open-mmlab/mmdetection.git

# %cd mmdetection

!pip install -e .

# install Pillow 7.0.0 back in order to avoid bug in colab

!pip install Pillow==7.0.0

# kiểm tra xem việc thiết lập có ok hay chưa

# Check Pytorch installation

import torch, torchvision

print(torch.\_\_version\_\_, torch.cuda.is\_available())

# Check MMDetection installation

import mmdet

print(mmdet.\_\_version\_\_)

# Check mmcv installation

from mmcv.ops import get\_compiling\_cuda\_version, get\_compiler\_version

print(get\_compiling\_cuda\_version())

print(get\_compiler\_version())

import copy

import os.path as osp

import mmcv

import numpy as np

import pickle

from mmcv import Config

from mmdet.apis import set\_random\_seed

from mmdet.datasets.builder import DATASETS

from mmdet.datasets.custom import CustomDataset

from mmdet.apis import inference\_detector, init\_detector, show\_result\_pyplot

from mmdet.datasets import build\_dataset

from mmdet.models import build\_detector

from mmdet.apis import train\_detector

"""# config gốc và checkpoint\_file"""

config\_file = '/content/drive/My Drive/mmdetection/configs/faster\_rcnn/faster\_rcnn\_r50\_fpn\_1x\_coco.py'

checkpoint\_file = '/content/drive/My Drive/mmdetection/checkpoints/faster\_rcnn\_r50\_fpn\_1x\_coco\_20200130-047c8118.pth'

"""# tạo module để đọc file về middle format"""

DATASETS.register\_module()

class flick\_dataset5(CustomDataset):

    CLASSES = ('Car', )

    def load\_annotations(self, ann\_file):

        cat2label = {k: i for i, k in enumerate(self.CLASSES)}

        # load image list from file

        image\_list = mmcv.list\_from\_file(self.ann\_file)

        data\_infos = []

        # convert annotations to middle format

        for image\_id in image\_list:

            filename = f'{self.img\_prefix}/{image\_id}.jpg'

            image = mmcv.imread(filename)

            height, width = image.shape[:2]

            data\_info = dict(filename=f'{image\_id}.jpg', width=width, height=height)

            # load annotations

            label\_prefix = self.img\_prefix.replace('image\_2', 'label\_2')

            lines = mmcv.list\_from\_file(osp.join(label\_prefix, f'{image\_id}.txt'))

            content = [line.strip().split(' ') for line in lines]

            bbox\_names = [x[0] for x in content]

            bboxes = [[float(info) for info in x[1:5]] for x in content]

            gt\_bboxes = []

            gt\_labels = []

            gt\_bboxes\_ignore = []

            gt\_labels\_ignore = []

            # filter 'DontCare'

            for bbox\_name, bbox in zip(bbox\_names, bboxes):

                if bbox\_name in cat2label:

                    gt\_labels.append(cat2label[bbox\_name])

                    gt\_bboxes.append(bbox)

                else:

                    gt\_labels\_ignore.append(-1)

                    gt\_bboxes\_ignore.append(bbox)

            data\_anno = dict(

                bboxes=np.array(gt\_bboxes, dtype=np.float32).reshape(-1, 4),

                labels=np.array(gt\_labels, dtype=np.long),

                bboxes\_ignore=np.array(gt\_bboxes\_ignore,

                                       dtype=np.float32).reshape(-1, 4),

                labels\_ignore=np.array(gt\_labels\_ignore, dtype=np.long))

            data\_info.update(ann=data\_anno)

            data\_infos.append(data\_info)

        return data\_infos

"""# Modify the config"""

cfg = Config.fromfile('config\_file')

from mmdet.apis import set\_random\_seed

# Modify dataset type and path

cfg.dataset\_type = 'flick\_dataset5'

cfg.data\_root = '/content/drive/My Drive/mmdetection/flick/'

cfg.data.test.type = 'flick\_dataset5'

cfg.data.test.data\_root = '/content/drive/My Drive/mmdetection/flick/'

cfg.data.test.ann\_file = '/content/drive/My Drive/mmdetection/flick/train.txt'

cfg.data.test.img\_prefix = '/content/drive/My Drive/mmdetection/flick/training/image\_2'

cfg.data.train.type = 'flick\_dataset5'

cfg.data.train.data\_root = '/content/drive/My Drive/mmdetection/flick/'

cfg.data.train.ann\_file = '/content/drive/My Drive/mmdetection/flick/train.txt'

cfg.data.train.img\_prefix = '/content/drive/My Drive/mmdetection/flick/training/image\_2'

cfg.data.val.type = 'flick\_dataset5'

cfg.data.val.data\_root = '/content/drive/My Drive/mmdetection/flick/'

cfg.data.val.ann\_file = '/content/drive/My Drive/mmdetection/flick/val.txt'

cfg.data.val.img\_prefix = '/content/drive/My Drive/mmdetection/flick/training/image\_2'

# modify num classes of the model in box head

cfg.model.roi\_head.bbox\_head.num\_classes = 1

# We can still use the pre-trained Mask RCNN model though we do not need to

# use the mask branch

cfg.load\_from = checkpoint\_file

# Set up working dir to save files and logs.

cfg.work\_dir = '/content/drive/My Drive/mmdetection/tutorial\_exps'

# The original learning rate (LR) is set for 8-GPU training.

# We divide it by 8 since we only use one GPU.

cfg.optimizer.lr = 0.02 / 8

cfg.lr\_config.warmup = None

cfg.log\_config.interval = 10

# Change the evaluation metric since we use customized dataset.

cfg.evaluation.metric = 'mAP'

# We can set the evaluation interval to reduce the evaluation times

cfg.evaluation.interval = 12

# We can set the checkpoint saving interval to reduce the storage cost

cfg.checkpoint\_config.interval = 12

# Set seed thus the results are more reproducible

cfg.seed = 0

set\_random\_seed(0, deterministic=False)

cfg.gpu\_ids = range(1)

# We can initialize the logger for training and have a look

# at the final config used for training

print(f'Config:\n{cfg.pretty\_text}')

"""# train"""

from mmdet.datasets import build\_dataset

from mmdet.models import build\_detector

from mmdet.apis import train\_detector

# Build dataset

datasets = [build\_dataset(cfg.data.train)]

# Build the detector

model = build\_detector(

    cfg.model, train\_cfg=cfg.train\_cfg, test\_cfg=cfg.test\_cfg)

# Add an attribute for visualization convenience

model.CLASSES = datasets[0].CLASSES

# Create work\_dir

mmcv.mkdir\_or\_exist(osp.abspath(cfg.work\_dir))

train\_detector(model, datasets, cfg, distributed=False, validate=True)

"""# testing: quên chỉnh tên class về logo"""

img = mmcv.imread('/content/drive/My Drive/mmdetection/flick/training/image\_2/2534155497.jpg')

model.cfg = cfg

result = inference\_detector(model, img)

show\_result\_pyplot(model, img, result)

"""## pickle"""

import pickle

pkl\_filename = "/content/drive/My Drive/flick\_dataset\_preprocessing/model/model\_1.pkl"

with open(pkl\_filename, 'wb') as file:

    pickle.dump(model, file)

"""## torch.save"""

PATH = '/content/drive/My Drive/flick\_dataset\_preprocessing/model/model\_1.pth'

torch.save(model, PATH)

# -\*- coding: utf-8 -\*-

"""phase 2: crawl image.ipynb

Automatically generated by Colaboratory.

Original file is located at

    https://colab.research.google.com/drive/1wM1gU1qEtpFBONOi3pC8FXT9y-TEfWSE

"""

import re

import requests

from bs4 import BeautifulSoup

# lảm mỗi lần một hãng để tránh trục trặc

urls = ['https://tiki.vn/search?q=lock%20and%20lock&ref=searchBar']

base\_url = 'https://tiki.vn'

#lấy tên các trang muốn download hình ảnh về

i = True

while(i):

  with requests.get(urls[-1]) as r:

    soup = BeautifulSoup(r.text)

    if soup.find('a', {'class': 'next'}):

      elm = soup.find('a', {'class': 'next'})

      next\_page\_link = base\_url + elm['href']

      urls.append(next\_page\_link)

      print(next\_page\_link)

    else:

      i = False

      break

#tạo dataframe chưa tên web

import pandas as pd

df = pd.DataFrame(data={"url": urls})

#len(df)

# download hình từ dataframe

for i in range(len(df)):

  print(df.url[i])

  site = df.url[i]

  response = requests.get(site)

  soup = BeautifulSoup(response.text, 'html.parser')

  img\_tags = soup.find\_all('img')

  #img\_tags

  img\_tags[0]['src']

  urls = []

  for i in range(len(img\_tags)):

    if img\_tags[i].has\_attr('src'):

      urls.append(img\_tags[i]['src'])

  for url in urls:

    filename = re.search(r'/([\w\_-]+[.](jpg|gif|png))$', url)

    if filename:

      with open("/content/drive/My Drive/mmdetection/images\_tiki\_false/lockandlock"+filename.group(0), 'wb') as f:

        if 'http' not in url:

          url = '{}{}'.format(site, url)

        response = requests.get(url)

        f.write(response.content)

print("done")

# -\*- coding: utf-8 -\*-

"""phase 2: image preprocessing.ipynb

Automatically generated by Colaboratory.

Original file is located at

    https://colab.research.google.com/drive/14nsXrE6vhMW8XO0G-k85Amf1Qz9rUjKz

# import library

"""

import matplotlib.pyplot

import numpy as np

import os

from PIL import Image

from collections import defaultdict

from itertools import product

from sklearn.model\_selection import train\_test\_split

import shutil

import re

import glob

from scipy import ndimage

import pickle

from six.moves import cPickle as pickle

from six.moves import range

#from \_\_future\_\_ import division, print\_function, absolute\_import

import tflearn

from tflearn.data\_utils import shuffle

from tflearn.layers.core import input\_data, dropout, fully\_connected

from tflearn.layers.conv import conv\_2d, max\_pool\_2d

from tflearn.layers.estimator import regression

from tflearn.data\_preprocessing import ImagePreprocessing

from tflearn.data\_augmentation import ImageAugmentation

import tensorflow as tf

# Commented out IPython magic to ensure Python compatibility.

# %cd "D:\\Ananconda\\login detect\\dataset"

"""# Load Data"""

# khai báo width height, rotation, shift, scale parameter

width = 32

height = 32

posshiftshift\_min = -5

posshiftshift\_max = 5

scales = [0.9, 1.1]

rot\_min = -15

rot\_max = 15

dir = 'dataset'

imgdir = os.path.join(dir, 'main\_dataset')

pp\_dir = os.path.join(

    dir, 'dataset8')

annot = 'result.txt'

annot\_train = np.loadtxt(os.path.join(dir, annot), dtype='a')

print('train\_annotation: %d, %d ' % (annot\_train.shape))

"""# Crop và augmente data

## augmente data

"""

# lấy bounding box từ text file

def parse\_annot(annot):

    fn = annot[0].decode('utf-8')

    class\_name = annot[1].decode('utf-8')

    train\_subset\_class = annot[2].decode('utf-8')

    return fn, class\_name, train\_subset\_class

def get\_rect(annot):

    rect = defaultdict(int)

    x1, y1, x2, y2 = rect\_coord(annot[3:])

    cx, cy, wid, hgt = center\_wid\_hgt(x1, y1, x2, y2)

    rect['x1'] = x1

    rect['y1'] = y1

    rect['x2'] = x2

    rect['y2'] = y2

    rect['cx'] = cx

    rect['cy'] = cy

    rect['wid'] = wid

    rect['hgt'] = hgt

    return rect

# shift vị trí bounding box trong hình

def posshift(annot, im):

    posshift\_ims = []

    posshift\_suffixes = []

    rect = get\_rect(annot)

    for sx, sy in product(

            range(posshiftshift\_min, posshiftshift\_max),

            range(posshiftshift\_min, posshiftshift\_max)):

        cx = rect['cx'] + sx

        cy = rect['cy'] + sy

        cropped\_im = im.crop((cx - rect['wid'] // 2, cy - rect['hgt'] // 2,

                              cx + rect['wid'] // 2, cy + rect['hgt'] // 2))

        resized\_im = cropped\_im.resize((width, height))

        posshift\_ims.append(resized\_im)

        posshift\_suffixes.append('p' + str(sx) + str(sy))

        cropped\_im.close()

    return posshift\_ims, posshift\_suffixes

# sclae bounding box theo parameter

def scale(annot, im):

    scale\_ims = []

    scale\_suffixes = []

    rect = get\_rect(annot)

    for s in scales:

        w = int(rect['wid'] \* s)

        h = int(rect['hgt'] \* s)

        cropped\_im = im.crop((rect['cx'] - w // 2, rect['cy'] - h // 2,

                              rect['cx'] + w // 2, rect['cy'] + h // 2))

        resized\_im = cropped\_im.resize((width, height))

        scale\_ims.append(resized\_im)

        scale\_suffixes.append('s' + str(s))

        cropped\_im.close()

    return scale\_ims, scale\_suffixes

# rotate bounding box theo parameter

def rotate(annot, im):

    rotate\_ims = []

    rotate\_suffixes = []

    rect = get\_rect(annot)

    for r in range(rot\_min, rot\_max):

        rotated\_im = im.rotate(r)

        cropped\_im = rotated\_im.crop(

            (rect['cx'] - rect['wid'] // 2, rect['cy'] - rect['hgt'] // 2,

             rect['cx'] + rect['wid'] // 2, rect['cy'] + rect['hgt'] // 2))

        resized\_im = cropped\_im.resize((width, height))

        rotate\_ims.append(resized\_im)

        rotate\_suffixes.append('r' + str(r))

        rotated\_im.close()

        cropped\_im.close()

    return rotate\_ims, rotate\_suffixes

"""## crop hình"""

#Cropping the logo

def crop(annot, im):

    x1, y1, x2, y2 = rect\_coord(annot[3:])

    cropped\_im = im.crop((x1, y1, x2, y2))

    cropped\_im = cropped\_im.resize((width, height))

    cropped\_suffix = 'p00'

    return [cropped\_im], [cropped\_suffix]

def rect\_coord(annot\_part):

    return list(map(int, annot\_part))

def center\_wid\_hgt(x1, y1, x2, y2):

    cx = x1 + (x2 - x1) // 2

    cy = y1 + (y2 - y1) // 2

    wid = (x2 - x1)

    hgt = (y2 - y1)

    return cx, cy, wid, hgt

# kiểm tra điều kiện bỏ logo, save hình và đóng file

def is\_skip(annot\_part):

    x1, y1, x2, y2 = rect\_coord(annot\_part)

    \_, \_, wid, hgt = center\_wid\_hgt(x1, y1, x2, y2)

    if wid <= 0 or hgt <= 0:

        return True

    else:

        return False

def save\_im(annot, cnt, \*args):

    fn, class\_name, train\_subset\_class = parse\_annot(annot)

    dst\_dir = os.path.join(pp\_dir, class\_name)

    if not os.path.exists(dst\_dir):

        os.makedirs(dst\_dir)

    for i, arg in enumerate(args):

        for im, suffix in zip(arg[0], arg[1]):

            save\_fn = '\_'.join([

                fn.split('.')[0], class\_name, train\_subset\_class, str(cnt),

                suffix

            ]) + os.path.splitext(fn)[1]

            im.save(os.path.join(dst\_dir, save\_fn))

def close\_im(\*args):

    for ims in args:

        for im in ims:

            im.close()

# parent function để gọi tất cà các sub functions

def crop\_and\_aug(annot\_train):

    cnt\_per\_file = defaultdict(int)

    for annot in annot\_train:

        # for generating a file name

        fn, \_, \_ = parse\_annot(annot)

        cnt\_per\_file[fn] += 1

        # skip if width or height equal zero

        if is\_skip(annot[3:]):

            print('Skip: ', fn)

            continue

        # open an image

        im = Image.open(os.path.join(imgdir, fn))

        # normal cropping

        cropped\_ims, cropped\_suffixes = crop(annot, im)

        # augment by shifting a center

        shifted\_ims, shifted\_suffixes = posshift(annot, im)

        # augment by scaling

        scaled\_ims, scaled\_suffixes = scale(annot, im)

        # augment by rotation

        rotated\_ims, rotated\_suffixes = rotate(annot, im)

        # save images

        save\_im(annot, cnt\_per\_file[fn], [cropped\_ims, cropped\_suffixes],

                [shifted\_ims, shifted\_suffixes], [scaled\_ims, scaled\_suffixes],

                [rotated\_ims, rotated\_suffixes])

        # close image file

        close\_im([im], cropped\_ims, shifted\_ims, scaled\_ims, rotated\_ims)

"""## tạo dataset"""

# gọi tất cà các function và tạo dataset mới

def crop\_and\_aug\_with\_none(annot\_train, with\_none=False):

    # root directory to save processed images

    if not os.path.exists(pp\_dir):

        os.makedirs(pp\_dir)

    # crop images and apply augmentation

    crop\_and\_aug(annot\_train)

    # print results

    org\_imgs = [img for img in os.listdir(imgdir)]

    crop\_and\_aug\_imgs = [

        fname

        for root, dirs, files in os.walk(pp\_dir)

        for fname in glob.glob(os.path.join(root, '\*.jpg'))  # look for the file with .jpg extension.

    ]

    print('original: %d' % (len(org\_imgs)))

    print('cropped: %d' % (len(crop\_and\_aug\_imgs)))

"""# splitting data"""

# tạo train và test set

def do\_train\_test\_split():

    class\_names = [cls for cls in os.listdir(pp\_dir)]

   # create directories under a particular class name.

    for class\_name in class\_names:

        if os.path.exists(

                os.path.join(pp\_dir, class\_name, 'train')):

            continue

        if os.path.exists(

                os.path.join(pp\_dir, class\_name, 'test')):

            continue

        imgs = [

            img

            for img in os.listdir(

                os.path.join(pp\_dir, class\_name))

        ]

        # train=0.75, test=0.25

        train\_imgs, test\_imgs = train\_test\_split(imgs)

        # move images to train or test directory

        # create directories

        os.makedirs(os.path.join(pp\_dir, class\_name, 'train'))

        os.makedirs(os.path.join(pp\_dir, class\_name, 'test'))

        for img in train\_imgs:

            dst = os.path.join(pp\_dir, class\_name, 'train')

            src = os.path.join(pp\_dir, class\_name, img)

            # moving image into that directory

            shutil.move(src, dst)

        for img in test\_imgs:

            dst = os.path.join(pp\_dir, class\_name, 'test')

            src = os.path.join(pp\_dir, class\_name, img)

            shutil.move(src, dst)

"""# gọi function"""

crop\_and\_aug\_with\_none(annot\_train)

do\_train\_test\_split()

"""# Tạo Pickle File chứa dataset

## tạo thông số cho pickle file

"""

#Parameters của hình

width = 32

height = 32

channel = 3

pix\_val = 255.0

dir = 'dataset'

# Directory where processed images are stored

pp\_dir = os.path.join(dir, 'dataset8')

# tên pickle file

pickle\_file = 'logo\_dataset.pickle'

# sớ hình sử dụng cho training và validate

train\_size = 70000

val\_size = 5000

# sớ hình sử dụng cho test

test\_size = 7000

# tạo một array chứa height, width và channel các hình trong dataset

def array(nb\_rows, image\_width, image\_height, image\_ch=1):

    if nb\_rows:

        dataset = np.ndarray(                               #  stores its height, width and channel into an array

            (nb\_rows, image\_height, image\_width, image\_ch), dtype=np.float32)

        labels = np.ndarray(nb\_rows, dtype=np.int32)        #  stores its labels

    else:

        dataset, labels = None, None

    return dataset, labels

"""## các function cho việc tạo pickle"""

# dụng nhập pickle files của 10 classes vào chung pickle file.

def combine(pickle\_files, train\_size, val\_size=0):

    num\_classes = len(pickle\_files)

    valid\_dataset, valid\_labels = array(val\_size, width,

                                              height, channel)

    train\_dataset, train\_labels = array(train\_size, width,

                                              height, channel)

    vsize\_per\_class = val\_size // num\_classes

    tsize\_per\_class = train\_size // num\_classes

    start\_v, start\_t = 0, 0

    end\_v, end\_t = vsize\_per\_class, tsize\_per\_class

    end\_l = vsize\_per\_class + tsize\_per\_class

    for label, pickle\_file in enumerate(pickle\_files):

        try:

            with open(pickle\_file, 'rb') as f:

                logo\_set = pickle.load(f)

                np.random.shuffle(logo\_set)

                if valid\_dataset is not None:

                    valid\_logo = logo\_set[:vsize\_per\_class, :, :, :]

                    valid\_dataset[start\_v:end\_v, :, :, :] = valid\_logo

                    valid\_labels[start\_v:end\_v] = label

                    start\_v += vsize\_per\_class

                    end\_v += vsize\_per\_class

                train\_logo = logo\_set[vsize\_per\_class:end\_l, :, :, :]

                train\_dataset[start\_t:end\_t, :, :, :] = train\_logo

                train\_labels[start\_t:end\_t] = label

                start\_t += tsize\_per\_class

                end\_t += tsize\_per\_class

        except Exception as e:

            print('Unable to process data from', pickle\_file, ':', e)

            raise

    return valid\_dataset, valid\_labels, train\_dataset, train\_labels

# tạo pickle files cho một class

def makepickle(train\_dataset, train\_labels, valid\_dataset, valid\_labels,

                test\_dataset, test\_labels):

    try:

        f = open(pickle\_file, 'wb')

        save = {

            'train\_dataset': train\_dataset,

            'train\_labels': train\_labels,

            'valid\_dataset': valid\_dataset,

            'valid\_labels': valid\_labels,

            'test\_dataset': test\_dataset,

            'test\_labels': test\_labels,

        }

        pickle.dump(save, f, pickle.HIGHEST\_PROTOCOL)      # Saving data of the images into a pickle file

        f.close()

    except Exception as e:

        print('Unable to save data to', pickle\_file, ':', e)

        raise

# load hình logo từ thông số

def load\_logo(data\_dir):

    image\_files = os.listdir(data\_dir)

    dataset = np.ndarray(

        shape=(len(image\_files), height, width, channel),

        dtype=np.float32)

    print(data\_dir)

    num\_images = 0

    for image in image\_files:

        image\_file = os.path.join(data\_dir, image)

        try:

            image\_data = (matplotlib.pyplot.imread(image\_file).astype(float) -

                          pix\_val / 2) / pix\_val

            if image\_data.shape != (height, width, channel):

                raise Exception('Unexpected image shape: %s' %

                                str(image\_data.shape))

            dataset[num\_images, :, :] = image\_data

            num\_images = num\_images + 1

        except IOError as e:

            print('Could not read:', image\_file, ':', e,

                  '-it\'s ok, skipping.')

    dataset = dataset[0:num\_images, :, :]

    print('Full dataset tensor:', dataset.shape)       # Tell processed number of images for a particular class

    print('Mean:', np.mean(dataset))                   # Calculate mean over that entire class

    print('Standard deviation:', np.std(dataset))      # Calculate standard deviation over that entire class

    return dataset

# tạo  parent function Pickle file từ các sub functions

def pickling(data\_dirs, force=False):

    dataset\_names = []

    for dir in data\_dirs:

        set\_filename = dir + '.pickle'

        dataset\_names.append(set\_filename)

        if os.path.exists(set\_filename) and force:

            print('%s already present - Skipping pickling. ' % set\_filename)

        else:

            print('Pickling %s.' % set\_filename)

            dataset = load\_logo(dir)

            try:

                with open(set\_filename, 'wb') as f:

                    pickle.dump(dataset, f, pickle.HIGHEST\_PROTOCOL)

            except Exception as e:

                print('Unable to save data to', set\_filename, ':', e)

    return dataset\_names

def randomize(dataset, labels):

    permutation = np.random.permutation(labels.shape[0])

    shuffled\_dataset = dataset[permutation, :, :]

    shuffled\_labels = labels[permutation]

    return shuffled\_dataset, shuffled\_labels

"""# tạo pickle file"""

CLASS\_NAME = [

   'Apple', 'BMW','Heineken','HP','Intel','Mini','Starbucks','Vodafone', 'unknown', 'Ferrari'

]

dirs = [

        os.path.join(pp\_dir, class\_name, 'train')      # Look into all the train folder of the class

        for class\_name in CLASS\_NAME

    ]

test\_dirs = [

        os.path.join(pp\_dir, class\_name, 'test')        # Look into all the test folder of the class

        for class\_name in CLASS\_NAME

    ]

train\_datasets = pickling(dirs)

test\_datasets = pickling(test\_dirs)

valid\_dataset, valid\_labels, train\_dataset, train\_labels = combine(train\_datasets, train\_size, val\_size)# function called for merging

a,b,test\_dataset, test\_labels= combine(test\_datasets, test\_size, val\_size=0)

train\_dataset, train\_labels = randomize(train\_dataset, train\_labels)   # function called for randomizing

valid\_dataset, valid\_labels = randomize(valid\_dataset, valid\_labels)

test\_dataset, test\_labels = randomize(test\_dataset, test\_labels)

makepickle(train\_dataset, train\_labels, valid\_dataset, valid\_labels,test\_dataset, test\_labels)# function called for making a pickle file.

statinfo = os.stat(pickle\_file)                         # Shows size of the file

print('Compressed pickle size:', statinfo.st\_size)

"""# test the dataset"""

def read\_data():

    with open("logo\_dataset.pickle", 'rb') as f:

        save = pickle.load(f)

        X = save['train\_dataset']       # assign X as train dataset

        Y = save['train\_labels']        # assign Y as train labels

        X\_test = save['test\_dataset']   # assign X\_test as test dataset

        Y\_test = save['test\_labels']    #assign Y\_test as test labels

        del save

    return [X, X\_test], [Y, Y\_test]

def reformat(dataset, labels):

    dataset = dataset.reshape((-1, 32, 32,3)).astype(np.float32)    # Reformatting shape array to give a scalar value for dataset.

    labels = (np.arange(10) == labels[:, None]).astype(np.float32)

    return dataset, labels

dataset, labels = read\_data()

X,Y = reformat(dataset[0], labels[0])

X\_test, Y\_test = reformat(dataset[1], labels[1])

print('Training set', X.shape, Y.shape)

print('Test set', X\_test.shape, Y\_test.shape)

# Shuffle the data

X, Y = shuffle(X, Y)    # Imported from TFLearn.

# -\*- coding: utf-8 -\*-

"""Phase 2: resnet\_50.ipynb

Automatically generated by Colaboratory.

Original file is located at

    https://colab.research.google.com/drive/1O3-ZVU3ieFZAkBz76W4P3qY1eVd\_09KX

#  import library and pre\_train model

"""

from google.colab import drive

drive.mount('/content/gdrive', force\_remount=True)

# Commented out IPython magic to ensure Python compatibility.

# %cd "/content/gdrive/My Drive/flick\_dataset\_preprocessing"

import pandas as pd

import numpy as np

import cv2

import os

import json

import glob

import shutil

import os

from tensorflow.keras import applications

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras import optimizers

from tensorflow.keras import Sequential, Model

from tensorflow.keras.layers import Dropout, Flatten, Dense, GlobalAveragePooling2D

from tensorflow.keras import backend as k

from tensorflow.keras.callbacks import ModelCheckpoint, LearningRateScheduler, TensorBoard, EarlyStopping

from keras.applications.resnet50 import ResNet50

from keras.models import Model

import keras

"""#create model"""

img\_width, img\_height = 32,32

model\_50 = ResNet50(include\_top=False, weights='imagenet', input\_shape=(img\_height,img\_width,3))

for layer in model.layers:

    layer.trainable = False

x = model.output

x = Flatten()(x)

x = Dense(1024, activation="relu")(x)

x = Dropout(0.5)(x)

x = Dense(1024, activation="relu")(x)

output = Dense(10, activation="softmax")(x)

model\_resnet = Model(inputs = model.input, outputs = output)

model\_resnet.compile(loss = "categorical\_crossentropy",

                    optimizer = "adam",

                    metrics=["accuracy"],

                     )

model.summary()

"""# read data from piclke file"""

def read\_data():

    with open("logo\_dataset.pickle", 'rb') as f:

        save = pickle.load(f)

        X = save['train\_dataset']       # assign X as train dataset

        Y = save['train\_labels']        # assign Y as train labels

        X\_test = save['test\_dataset']   # assign X\_test as test dataset

        Y\_test = save['test\_labels']    #assign Y\_test as test labels

        del save

    return [X, X\_test], [Y, Y\_test]

def reformat(dataset, labels):

    dataset = dataset.reshape((-1, 32, 32,3)).astype(np.float32)    # Reformatting shape array to give a scalar value for dataset.

    labels = (np.arange(10) == labels[:, None]).astype(np.float32)

    return dataset, labels

dataset, labels = read\_data()

X,Y = reformat(dataset[0], labels[0])

X\_test, Y\_test = reformat(dataset[1], labels[1])

print('Training set', X.shape, Y.shape)

print('Test set', X\_test.shape, Y\_test.shape)

# Shuffle the data

X, Y = shuffle(X, Y)    # Imported from TFLearn.

"""# train model"""

from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint

callback = [EarlyStopping(monitor = "val\_loss",patience=5),

            ModelCheckpoint("check\_point.h5",save\_best\_only= True)]

history = model.fit(X,Y,

                epochs=12,

               validation\_data=(X\_test, Y\_test),

                batch\_size=128

               # callbacks = [callback]

               )

import matplotlib.pyplot as plt

plt.figure()

plt.plot(history.history["acc"])

plt.plot(history.history["val\_acc"])

plt.xlabel("epoch")

plt.ylabel("value")

plt.title("acc % val\_acc")

plt.legend(["acc","val\_acc"])

plt.show()

"""# tạo confusion matrix"""

def plot\_confusion\_matrix(cm, classes,

                          normalize=False,

                          title='Confusion matrix',

                          cmap=plt.cm.Blues):

    """

    This function prints and plots the confusion matrix.

    Normalization can be applied by setting `normalize=True`.

    """

    if normalize:

        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

        print("Normalized confusion matrix")

    else:

        print('Confusion matrix, without normalization')

    plt.imshow(cm, interpolation='nearest', cmap=cmap)

    plt.title(title)

    plt.colorbar()

    tick\_marks = np.arange(len(classes))

    plt.xticks(tick\_marks, classes, rotation=45)

    plt.yticks(tick\_marks, classes)

    fmt = '.2f' if normalize else 'd'

    thresh = cm.max() / 2.

    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):

        plt.text(j, i, format(cm[i, j], fmt),

                 horizontalalignment="center",

                 color="white" if cm[i, j] > thresh else "black")

    plt.tight\_layout()

    plt.ylabel('True label')

    plt.xlabel('Predicted label')

from sklearn.metrics import confusion\_matrix

preds = np.argmax(model.predict(X\_test), axis = 1)

y\_orig = np.argmax(Y\_test, axis = 1)

cm = confusion\_matrix(preds, y\_orig)

from collections import OrderedDict

import itertools

genres = {'Apple': 0, 'BMW': 1, 'Heineken': 2, 'HP': 3, 'Intel': 4,

          'Mini': 5, 'Starbucks': 6, 'Vodafone': 7, 'unknown': 8, 'Ferrari': 9}

keys = OrderedDict(sorted(genres.items(), key=lambda t: t[1])).keys()

plt.figure(figsize=(8,8))

plot\_confusion\_matrix(cm, keys, normalize=True)

"""# save model"""

model\_resnet.save("model\_resnet\_dataset8.h5")

# -\*- coding: utf-8 -\*-

"""phase 3: pipe line cho hai model.ipynb

Automatically generated by Colaboratory.

Original file is located at

    https://colab.research.google.com/drive/12kBE6Joa48n3VoZApIgSykbOKlMuWwBA

"""

"""# import library"""

from google.colab import drive

drive.mount('/content/drive')

# Commented out IPython magic to ensure Python compatibility.

# %cd "/content/drive/My Drive"

# Commented out IPython magic to ensure Python compatibility.

# install dependencies: (use cu101 because colab has CUDA 10.1)

!pip install -U torch==1.5.1+cu101 torchvision==0.6.1+cu101 -f https://download.pytorch.org/whl/torch\_stable.html

# install mmcv-full thus we could use CUDA operators

!pip install mmcv-full

# Install mmdetection

!rm -rf mmdetection

!git clone https://github.com/open-mmlab/mmdetection.git

# %cd mmdetection

!pip install -e .

# install Pillow 7.0.0 back in order to avoid bug in colab

!pip install Pillow==7.0.0

from mmdet.apis import inference\_detector

import mmcv

import glob

import shutil

import os

import torch

import tensorflow as tf

import matplotlib.pyplot as plt

from PIL import Image

from keras.preprocessing import image

import numpy as np

import cv2

# load 2 model

mmdetection\_path = '/content/drive/My Drive/logo detection: phase 2/model\_1.pth'

resnet\_path= '/content/drive/My Drive/logo detection: phase 2/model\_resnet\_dataset8.h5'

mmdetection\_model = torch.load(mmdetection\_path)

resnet\_model = tf.keras.models.load\_model(resnet\_path)

# tạo tên class

class\_list =[

   'Apple', 'BMW','Heineken','HP','Intel','Mini','Starbucks','Vodafone', 'Unknown', 'Ferrari'

]

"""#pipe line"""

# dùng mmdetection để lấy vị trí logo trong hình và crop ra logo

def detect\_logo\_location(image\_path):

  list\_img =[]

  img = mmcv.imread(image\_path)

  j=0

  dst\_dir = "/content/drive/My Drive/logo detection: phase 2/dataset for testing/crop"

  result = inference\_detector(mmdetection\_model, img) # lấy bouding box từ hình sử dụng model mmdetection

  if result: # nếu tập không rỗng( có tồn tại logo)

    img = Image.open(image\_path)

    for i in range(len(result[0])): # loop qua từng bounding box trong hình

      if result[0][i][4] >= 0.3: # nếu probality > 0.5 lấy bouding và crop hình

        left = result[0][i][0]

        top = result[0][i][1]

        right = result[0][i][2]

        bottom = result[0][i][3]

        im\_crop = img.crop((left, top, right, bottom))

        if im\_crop.mode == "RGBA":

          im\_crop = im\_crop.convert('RGB')

        name = os.path.join(dst\_dir, "crop\_" + str(j) + ".jpg")

        im\_crop.save(name)

        j=j+1

        list\_img.append(name)

  return list\_img

# kiểm tra chất lượng logo

def check\_image\_quality(list\_img):

  list\_img2 =[]

  if list\_img:

    list\_img2 =[]

    for i in range(len(list\_img)):

      image\_path = list\_img[i]

      im = cv2.imread(image\_path)

      h, w, c = im.shape

      if h < 32 or w < 32:

        print("logo quá nhỏ")

      else:

        list\_img2.append(list\_img[i])

  return list\_img2

# dủng resnet để lấy ta xác suất từng class, class cao nhất lấy về 1

def detect\_brand\_probality(list\_img):

  classes\_probality = []

  for i in range(len(list\_img)):

    image\_path = list\_img[i]

    img\_width, img\_height =32,32

    img = image.load\_img(image\_path, target\_size=(img\_width, img\_height))

    x = image.img\_to\_array(img)

    x = np.expand\_dims(x, axis=0)

    images = np.vstack([x])

    classes = resnet\_model.predict(images)

    classes\_probality.append(list(classes[0]))

  return classes\_probality

# trả về tên nhãn hiệu từ xác suất

def class\_name(classes\_probality):

  Class\_name = []

  for i in range(len(classes\_probality)):

    list\_class = classes\_probality[i]

    location = list\_class.index(max(list\_class))

    Class\_name.append(class\_list[location])

  return Class\_name

src\_dir = "/content/drive/My Drive/logo detection: phase 2/dataset for testing/intel"

for jpgfile in glob.iglob(os.path.join(src\_dir, "\*.jpg")):

  print(jpgfile)

  image\_path = jpgfile

  list\_img = detect\_logo\_location(image\_path)

  list\_img2 = check\_image\_quality(list\_img)

  if list\_img2:

    classes\_probality = detect\_brand\_probality(list\_img2)

    print(class\_name(classes\_probality))

  else:

    print("logo không tồn tại")

"""phase 4: demo.ipynb

Automatically generated by Colaboratory.

Original file is located at

    https://colab.research.google.com/drive/12kBE6Joa48n3VoZApIgSykbOKlMuWwBA

"""

from PIL import Image, ImageOps

import streamlit as st

import tensorflow as tf

from tensorflow.keras.preprocessing import image

import numpy as np

from androguard.core.bytecodes.apk import APK

st.title("Logo detection with mmdetection + resnet\_50")

st.header("mmdetection chỉ có thể sử dụng trên linux và macOS")

st.header("Logo detection Example")

st.text("Upload an Image for Logo detection")

class\_list =[

   'Apple', 'BMW','Heineken','HP','Intel','Mini','Starbucks','Vodafone', 'Unknown', 'Ferrari'

]

resnet\_model = tf.keras.models.load\_model('D:\Ananconda\streamlit\model\_resnet\_dataset8.h5')

def detect\_brand\_probality(img):

    # Create the array of the right shape to feed into the keras model

    data = np.ndarray(shape=(1, 32, 32, 3), dtype=np.float32)

    image = img

    #image sizing

    size = (32, 32)

    image = ImageOps.fit(image, size, Image.ANTIALIAS)

    #turn the image into a numpy array

    image\_array = np.asarray(image)

    # Normalize the image

    normalized\_image\_array = (image\_array.astype(np.float32) / 127.0) - 1

    # Load the image into the array

    data[0] = normalized\_image\_array

    classes = resnet\_model.predict(data)

    # run the inference

    classes = list(classes[0])

    location = classes.index(max(classes))

    return class\_list[location]

st.set\_option('deprecation.showfileUploaderEncoding', False)

uploaded\_file = st.file\_uploader("Choose an image ...", type="jpg")

if uploaded\_file is not None:

    image = Image.open(uploaded\_file)

    st.image(image, caption='Uploaded logo', use\_column\_width=True)

    st.write("")

    st.write("Classifying...")

    label = detect\_brand\_probality(image)

    st.write(label)