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

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 về 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 được 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.

```
[
    {
        'filename': 'a.jpg',
        'width': 1280,
        'height': 720,
        'ann': {
            'bboxes': <np.ndarray, float32> (n, 4),
            'labels': <np.ndarray, int64> (n, ),
            'bboxes_ignore': <np.ndarray, float32> (k, 4),
            'labels_ignore': <np.ndarray, int64> (k, ) (optional field)
        }
    },
    ...
]
```

FIGURE 1. MIDDLE FORMAT FOR MMDETECTION

3.3 Chọn lựa detection frameworks cho mmdetection

Chọn Faster R-CNN

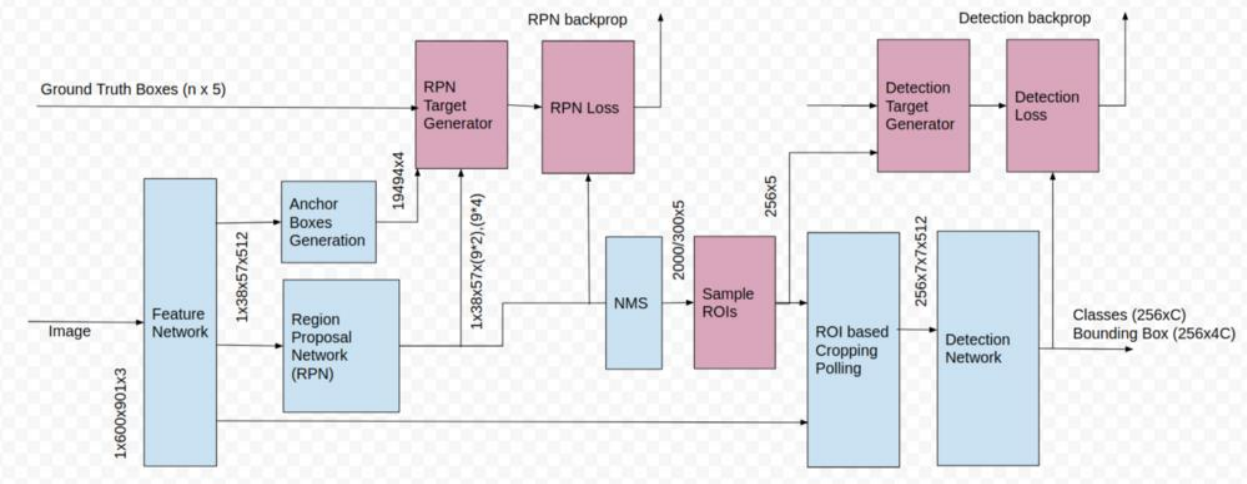


FIGURE 2 FASTER R CNN

3.4 tham số cho MMDETECTION

```
# 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}')
```

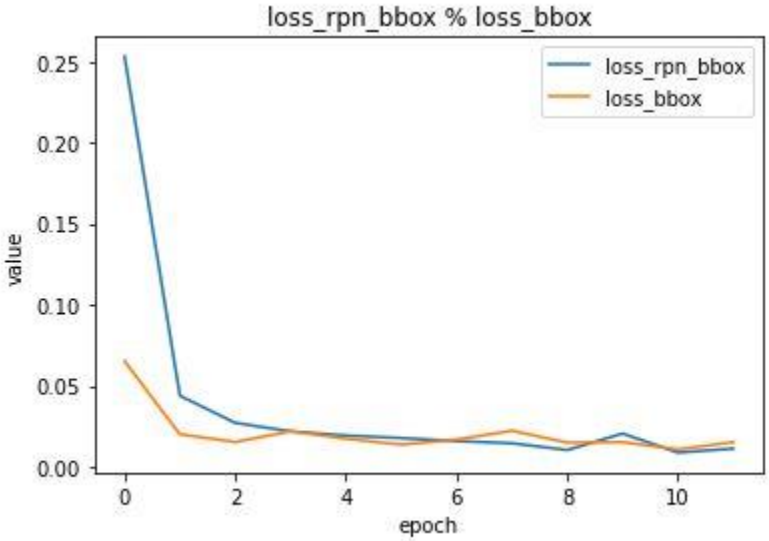
FIGURE 3 HYPER PARAMETER FOR MMDETECTION

```
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
```



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.

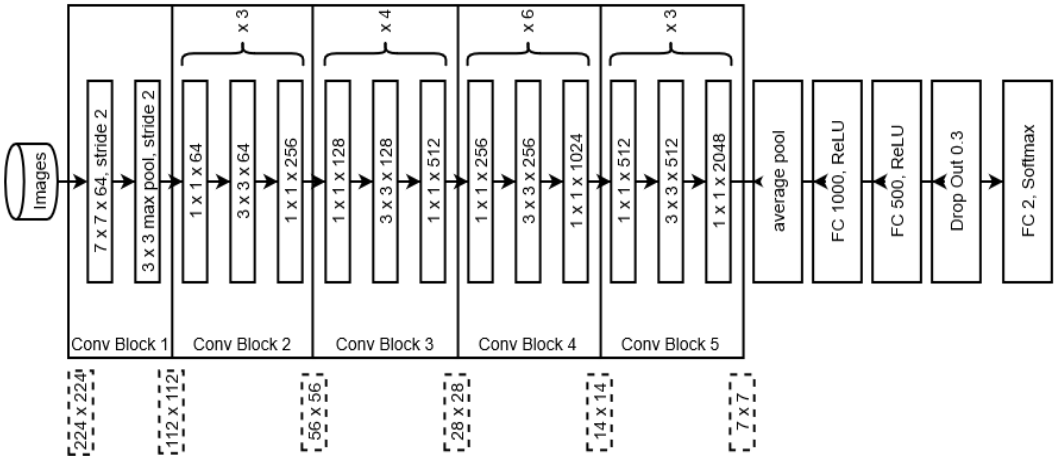
4 XÂY DỰNG MODEL RESNET_50 ĐỂ XÁC ĐỊNH BRAND TRONG HÌNH

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.



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.

```
1 img_width, img_height = 32,32
```

```
[ ]  1 model_50 = ResNet50(include_top=False, weights='imagenet', input_shape=(img_height,img_width,3))
     2
     3 for layer in model.layers:
     4     layer.trainable = False
```

```
[ ]  1 x = model.output
     2 x = Flatten()(x)
     3 x = Dense(1024, activation="relu")(x)
     4 x = Dropout(0.5)(x)
     5 x = Dense(1024, activation="relu")(x)
     6 output = Dense(10, activation="softmax")(x)
```

```
[ ]  1 model_resnet = Model(inputs = model.input, outputs = output)
```

```
[ ]  1 model_resnet.compile(loss = "categorical_crossentropy",
     2                     optimizer = "adam",
     3                     metrics=["accuracy"],
     4                      )
```

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:

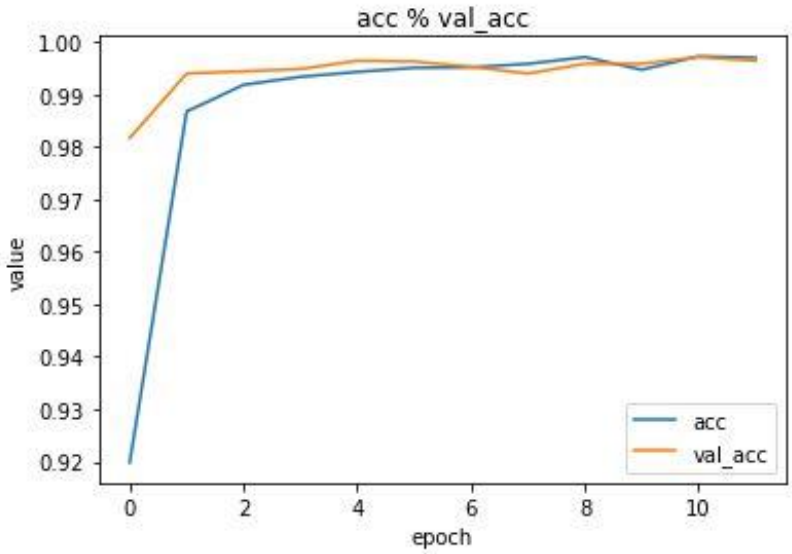
$$ACC = (TP + TN) / (TP + TN + FP + FN) \quad (1)$$

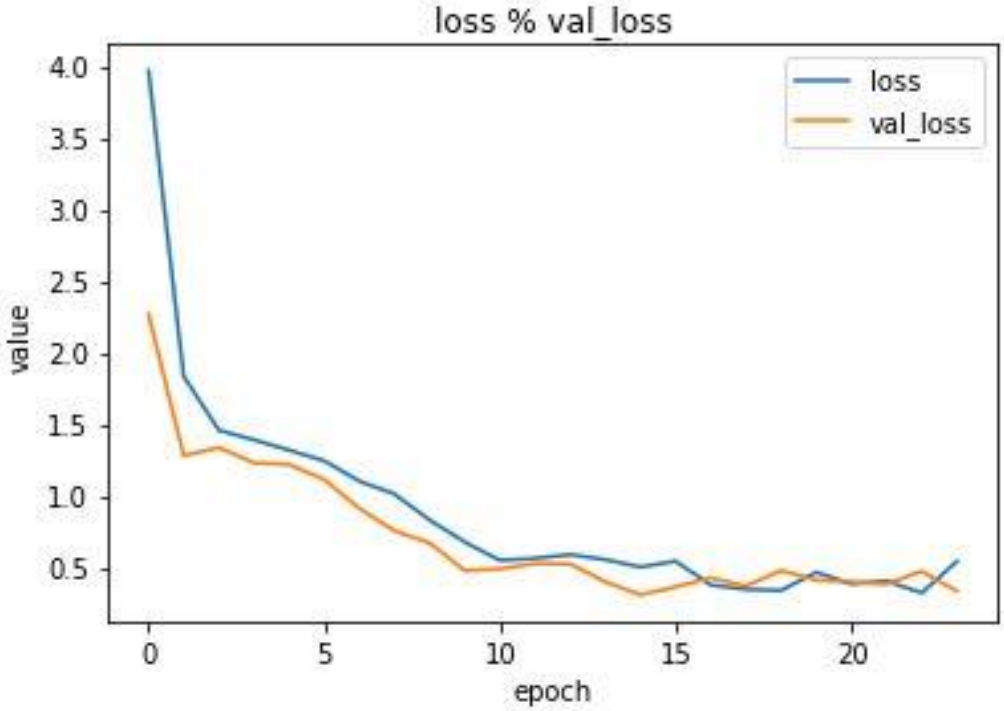
$$PRE = TP / (TP + FP) \quad (2)$$

$$REC = TP / (TP + FN) \quad (3)$$

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.

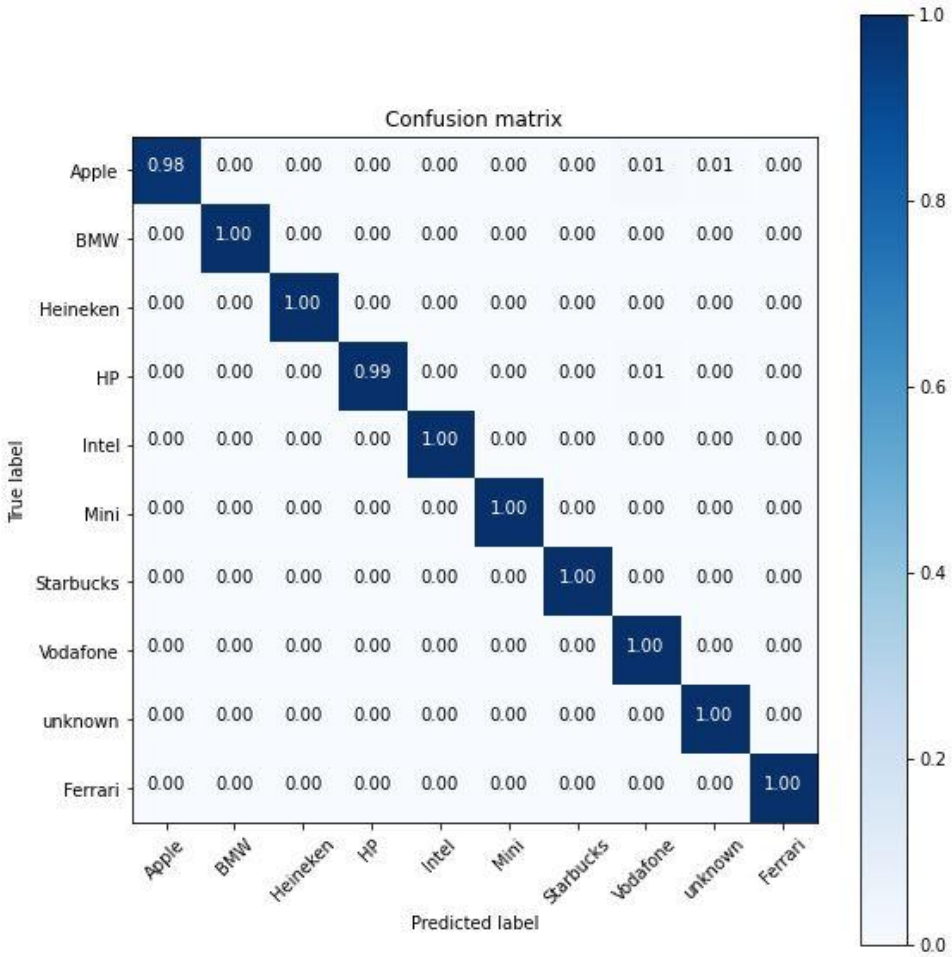
4.4 Kết quả


acc % val_acc
1.00
0.99
0.98
0.97
0.96
0.95
0.94
0.93
0.92
value
0
2
4
6
8
10
epoch
acc
val_acc


loss % val_loss
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
value
0
5
10
15
20
epoch
loss
val_loss

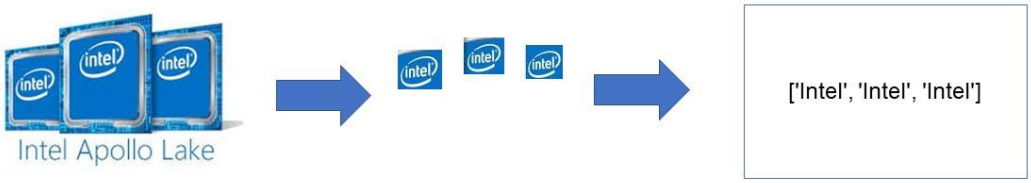
```
score=model.evaluate(X_test, Y_test)
print(score)
```

```
7000/7000 [==============================] - 4s 615us/sample - loss: 0.0154 - acc: 0.9959
[0.015377571084094337, 0.9958571]
```

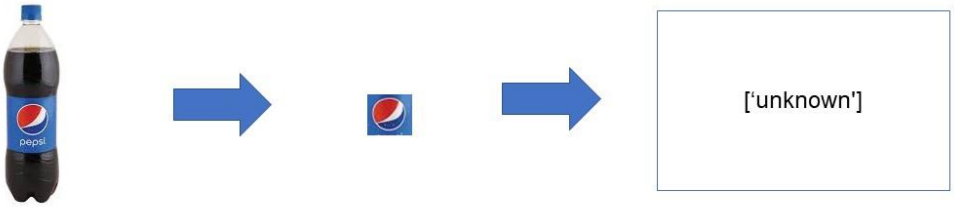


4.5 triển khai mô hình

Cho hình có logo thuộc 1 trong 9 hãng sẽ trả về tên hãng cho từng logo

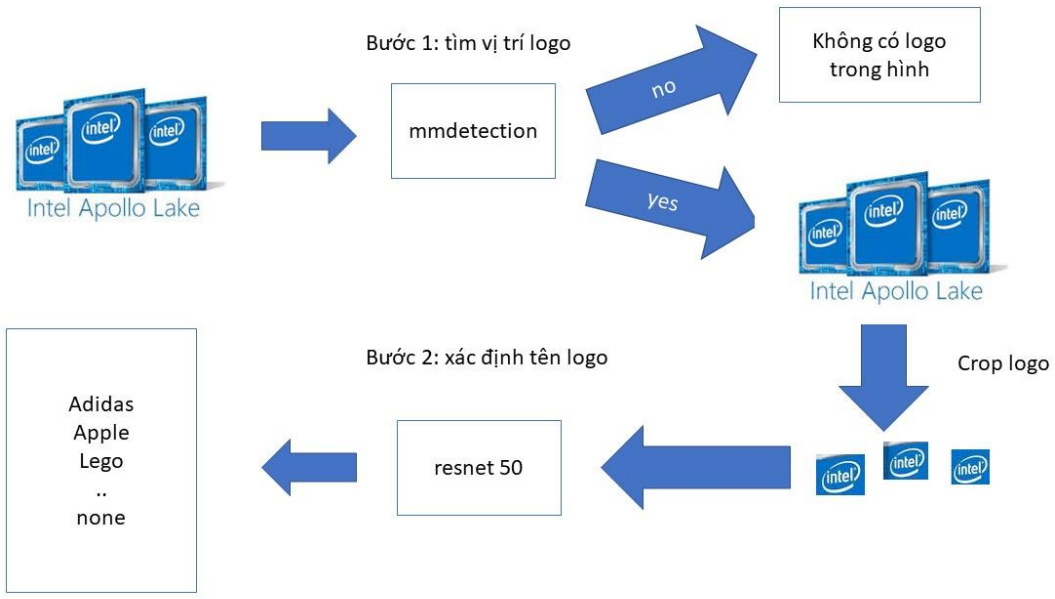


Cho hình thuộc về hãng không nằm trong những hãng đã học



5. TRIỂN KHAI FULL MÔ HÌNH MMDETECTION VÀ RESNET_50

5.1 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.

[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: 10.1016/j.jddst.2018.04.020.

[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: 10.1007/s00138-011-0366-4.

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",heade
r= None, sep=" ")
# đổi tên các cột sang tên tương ứng
flick = dataset.rename(columns={0:"file_name",1:"categories_names",2: "Training s
ubset",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', i
ndex=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 cầ
n 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 fol
der mới

src_dir = "/content/gdrive/My Drive/flick_dataset_preprocessing/flickr_logos_27_d
ataset_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 dạn
g 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 m
mdetection

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 để 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_rcn
n_r50_fpn_1x_coco.py'
checkpoint_file = '/content/drive/My Drive/mmdetection/checkpoints/faster_rcnn_r5
0_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=heig
ht)

            # 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/im
age_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/i
mage_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/ima
ge_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_2/253
4155497.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/lockandloc
k"+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 i
mages 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 devia
tion 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 v
alue 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, GlobalAveragePooling
2D
from tensorflow.keras import backend as k
```

```
from tensorflow.keras.callbacks import ModelCheckpoint, LearningRateScheduler, Te
nsorBoard, 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_heigh
t,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 v
alue 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_datase
t8.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 s
ử 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/in
tel"
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_da
taset8.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)
```