**TRƯỜNG ĐẠI HỌC LÂM NGHIỆP**

KHOA CƠ ĐIỆN VÀ CÔNG TRÌNH

**===o0o===**

**BÁO CÁO ĐỒ ÁN CUỐI KỲ**

**MÔN: KHAI PHÁ DỮ LIỆU**

**ĐỀ TÀI: NHẬN DIỆN GIÀY DÉP SỬ DỤNG TENSORFLOW**



**Giảng viên hướng dẫn: Mai Hà An**

**Sinh viên thực hiện: Vì Minh Châu**

**Mã sinh viên: 2041110025**

**Lớp: K65 - HTTT**

**Hà Nội, 2024**

MỤC LỤC

LỜI CẢM ƠN

[PHẦN I: GIỚI THIỆU 3](#_Toc157887485)

[1.1. Tổng Quan 3](#_Toc157887486)

[1.1.1 Mở đầu 3](#_Toc157887487)

[1.1.2. Mục tiêu đề tài 3](#_Toc157887488)

[1.1.3 Đối tượng nghiên cứu 4](#_Toc157887489)

[1.1.4 Phạm vi nghiên cứu 4](#_Toc157887490)

[1.2. Phương pháp – kết quả 4](#_Toc157887491)

[1.2.1 Phương pháp 4](#_Toc157887492)

[1.2.2 Kết quả 4](#_Toc157887493)

[1.2.3 Cấu Trúc đồ án 5](#_Toc157887494)

[PHẦN II: CƠ SỞ LÝ THUYẾT 6](#_Toc157887495)

[2.1. Tổng quan về Deep Learning 6](#_Toc157887496)

[2.1.1 Deep Learning là gì? 6](#_Toc157887497)

[2.1.2 Một số ứng dụng của Deep Learning 6](#_Toc157887498)

[2.2. Giới thiệu Convolutional Neural Network 6](#_Toc157887499)

[2.2.1 Tổng quan 6](#_Toc157887500)

[2.2.2 Cấu trúc CNN 6](#_Toc157887501)

[2.3. Thư viện Tensorflow 8](#_Toc157887502)

[2.4. Thư viện Keras 9](#_Toc157887503)

[2.5. Giới thiệu Google colab 9](#_Toc157887504)

[PHẦN III: TRIỂN KHAI XÂY DỰNG 11](#_Toc157887505)

[3.1. Điều kiện tiên quyết 11](#_Toc157887506)

[3.2. Môi trường thực hiện 11](#_Toc157887507)

[3.3. Các bước thực hiện 11](#_Toc157887508)

[3.3.1 Tạo dữ liệu và kết nối Colab với Drive 11](#_Toc157887509)

[3.3.2 Tiền xử lý hình ảnh 12](#_Toc157887510)

[3.3.3 Huấn Luyện 14](#_Toc157887511)

[3.3.4 Test 16](#_Toc157887512)

[PHẦN IV: KẾT LUẬN 18](#_Toc157887513)

[4.1. Ưu điểm - nhược điểm 18](#_Toc157887514)

[4.2. Hướng phát triển 18](#_Toc157887515)

**LỜI CẢM ƠN**

Em xin được gửi lời cảm ơn trân thành đến các thầy cô trong ngành công nghệ thông tin Trường đại học lâm nghiệp và đặc biệt là thầy **Mai Hà An** là người đã trực tiếp giảng dạy và hướng dẫn, giúp đỡ về kiến thức và phương pháp nghiên cứu để em có thể hoàn thành đồ án kết thúc môn học Lập trình mạng với đề tài “***Nhận diện giày dép sử dụng Tensorflow***”.

Em xin chân thành cảm ơn!

*Hà Nội, ngày 30 tháng 1 năm 2024*

**Sinh viên thực hiện**

***Vì Minh Châu***

# PHẦN I: GIỚI THIỆU

## Tổng Quan

### 1.1.1 Mở đầu

Trong thời đại công nghệ ngày nay, trí tuệ nhân tạo (AI) đang trở thành một phần không thể thiếu của cuộc sống hàng ngày, từ ứng dụng trực tuyến đến các lĩnh vực công nghiệp. Trong lĩnh vực thị giác máy tính, AI đã đem lại những tiến bộ đáng kể, đặc biệt là trong việc nhận diện và phân loại đối tượng trong hình ảnh.

Một trong những ứng dụng thú vị của AI là việc nhận diện giày dép từ hình ảnh. Đối với ngành công nghiệp thời trang và bán lẻ, việc nhận diện và phân loại giày dép không chỉ là một công cụ tiện ích mà còn là một yếu tố quyết định trong trải nghiệm mua sắm của người tiêu dùng. Điều này thúc đẩy sự phát triển và ứng dụng của các công nghệ nhận diện đối tượng trong ngành thời trang.

Trong bối cảnh đó, TensorFlow, một trong những thư viện hàng đầu trong lĩnh vực học sâu và máy học, đã trở thành công cụ quan trọng trong việc xây dựng các hệ thống nhận diện giày dép. TensorFlow cung cấp các công cụ mạnh mẽ và linh hoạt cho việc xây dựng, huấn luyện và triển khai mô hình học sâu.

Bài báo cáo này tập trung vào việc áp dụng TensorFlow để xây dựng một hệ thống nhận diện giày dép từ hình ảnh. Mục tiêu của dự án là phát triển một mô hình AI có khả năng nhận diện và phân loại các loại giày dép từ các hình ảnh được cung cấp. Điều này có thể áp dụng trong nhiều tình huống thực tế, từ quảng cáo trực tuyến đến quản lý kho hàng và trải nghiệm mua sắm trực tuyến.

Bài báo cáo này sẽ đề cập đến một quy trình toàn diện từ việc thu thập dữ liệu, tiền xử lý dữ liệu, xây dựng mô hình, huấn luyện mô hình, đánh giá hiệu suất, đến triển khai mô hình nhận diện giày dép sử dụng TensorFlow.

### 1.1.2. Mục tiêu đề tài

* Hiểu về TensorFlow Object Detection API và cách sử dụng nó để nhận diện giày dép trên các hình ảnh.
* Chuẩn bị dữ liệu huấn luyện và tạo tệp tin cấu hình cho mô hình nhận diện giày dép.
* Huấn luyện mô hình với dữ liệu và tinh chỉnh các tham số để đạt được hiệu suất tốt nhất.
* Đánh giá mô hình đã huấn luyện và kiểm tra khả năng nhận diện giày dép trên các ảnh mới.
* Đề xuất những cải tiến và ứng dụng tiềm năng của việc nhận diện giày dép sử dụng TensorFlow Object Detection API.

### 1.1.3 Đối tượng nghiên cứu

Đối tượng nghiên cứu chính trong dự án này là ứng dụng nhận diện giày dép. Nghiên cứu tập trung vào việc xây dựng một mô hình máy học nhằm nhận diện và phân loại các loại giày từ hình ảnh đầu vào. Mô hình được xây dựng bằng cách sử dụng các kỹ thuật Deep Learning và mạng neural tích chập (CNN) để học và trích xuất các đặc trưng từ hình ảnh giày.

### 1.1.4 Phạm vi nghiên cứu

* Xây dựng chương trình nhận diện giày dép sử dụng Tensorflow Object Detection trên Google Colab.
* Ứng dụng đề tài phục vụ cho việc nghiên cứu về nhận diện các loại giày dép phổ biến, bao gồm các loại sneaker, sandal, boot, và loafer.

## 1.2. Phương pháp – kết quả

### 1.2.1 Phương pháp

- Phương pháp nghiên cứu lý thuyết:

+ Tìm hiểu tổng quan về Deep Learning và bài toán nhận diện giày dép trong thị giác máy tính.

+ Tìm hiểu các thư viện Tensorflow, Keras, OpenCV và một số thư viện khác.

- Phương pháp nghiên cứu thực nghiệm:

+ Tiến hành phân tích và cài đặt trên Google Colab.

+ Tiến hành training data cho máy học.

### 1.2.2 Kết quả

* Kết quả của dự án là việc tạo ra một chương trình nhận diện giày dép trên hệ điều hành Windows. Chương trình này có khả năng nhận diện và phân loại các loại giày dép từ các hình ảnh đầu vào. Các kết quả đạt được được đánh giá dựa trên độ chính xác và độ tin cậy của việc nhận diện giày.

### 1.2.3 Cấu Trúc đồ án

*\*Nội dung chính được chia thành 4 phần như sau:*

* **Phần 1: Giới thiệu:** Phần này trình bày một cách tổng quan về mục tiêu, đối tượng và phạm vi nghiên cứu, các hướng tiếp cận để giải quyết bài toán nhận dạng và kết quả dự tính.
* **Phần 2: Cơ sở lý thuyết:** Tìm hiểu phương pháp máy học và tìm hiểu các thư viện, ứng dụng Deep Learning và bài toán nhận diện giày dép.
* **Phần 3: Ứng dụng thực nghiệm:** trên cơ sở thư viện mã nguồn mở xây dựng chương trình phát hiện giày dép qua hình ảnh, cùng với những phân tích chương trình.
* **Phần 4: Kết luận và hướng phát triển:** đánh giá kết quả và hướng phát triển nghiên cứu.

# PHẦN II: CƠ SỞ LÝ THUYẾT

## 2.1. Tổng quan về Deep Learning

### 2.1.1 Deep Learning là gì?

Deep Learning là một chủ đề Trí truệ nhân tạo (AI) và là một phạm trù nhỏ của máy học. Deep Learning tập trung giải quyết các vấn đề liên quan đến mạng nơron nhân tạo nhằm nâng cấp các công nghệ như nhận diện giọng nói, thị giác máy tính và xử lý ngôn ngữ tự nhiên.

Deep Learning đang trở thành một trong những lĩnh vực quan trọng trong khoa học máy tính. Chỉ trong vài năm, Deep Learning đã thúc đẩy tiến bộ trong đa dạng các lĩnh vực như nhận thức sự vật (object perception), dịch tự động (machine translation), nhận diện giọng nói - đó là những vấn đề từng rất khó khăn với các nhà nghiên cứu trí tuệ nhân tạo.

### 2.1.2 Một số ứng dụng của Deep Learning

* Hệ thống xử lý trên các nền tảng.
* Nhận diện hình ảnh.
* Xử lý ngôn ngữ tự nhiên.

## 2.2. Giới thiệu Convolutional Neural Network

### 2.2.1 Tổng quan

Convolutional Neural Network (CNN) bao gồm một hoặc nhiều lớp chập (thường với một bước lấy mẫu con) và sau đó theo sau bởi một hoặc nhiều hơn các lớp kết nối như trong một mạng nơron đa lớp chuẩn. Kiến trúc của một CNN được thiết kế để tận dụng lợi thế của cấu trúc 2 chiều của một hình ảnh đầu vào (hoặc đầu vào 2 chiều khác như một tín hiệu tiếng nói). Điều này đạt được với các kết nối cục bộ và trọng số ràng buộc theo một số hình thức tổng hợp mà kết quả là các đặc trưng không thay đổi. Một lợi ích khác của CNN là dễ dàng huấn luyện hơn và có ít thông số so với các mạng kết nối đầy đủ với cùng một số đơn vị ẩn.

### 2.2.2 Cấu trúc CNN

Mạng CNN là một tập hợp các lớp Convolution chồng lên nhau và sử dụng các hàm nonlinear activation như ReLU và tanh để kích hoạt các trọng số trong các node. Mỗi một lớp sau khi thông qua các hàm kích hoạt sẽ tạo ra các thông tin trừu tượng hơn cho các lớp tiếp theo.

Mỗi một lớp sau khi thông qua các hàm kích hoạt sẽ tạo ra các thông tin trừu tượng hơn cho các lớp tiếp theo. Trong mô hình mạng truyền ngược (feedforward neural network) thì mỗi neural đầu vào (input node) cho mỗi neural đầu ra trong các lớp tiếp theo.

Mô hình này gọi là mạng kết nối đầy đủ (fully connected layer) hay mạng toàn vẹn (affine layer). Còn trong mô hình CNNs thì ngược lại. Các layer liên kết được với nhau thông qua cơ chế convolution.

Layer tiếp theo là kết quả convolution từ layer trước đó, nhờ vậy mà ta có được các kết nối cục bộ. Như vậy mỗi neuron ở lớp kế tiếp sinh ra từ kết quả của filter áp đặt lên một vùng ảnh cục bộ của neuron trước đó.

Mỗi một lớp được sử dụng các filter khác nhau thông thường có hàng trăm hàng nghìn filter như vậy và kết hợp kết quả của chúng lại. Ngoài ra có một số layer khác như pooling/subsampling layer dùng để chắt lọc lại các thông tin hữu ích hơn (loại bỏ các thông tin nhiễu).

Trong quá trình huấn luyện mạng (traning) CNN tự động học các giá trị qua các lớp filter dựa vào cách thức mà bạn thực hiện. Ví dụ trong tác vụ phân lớp ảnh, CNNs sẽ cố gắng tìm ra thông số tối ưu cho các filter tương ứng theo thứ tự raw pixel > edges > shapes > facial > high-level features. Layer cuối cùng được dùng để phân lớp ảnh.



Trong mô hình CNN có 2 khía cạnh cần quan tâm là **tính bất biến** (Location Invariance) và **tính kết hợp** (Compositionality). Với cùng một đối tượng, nếu đối tượng này được chiếu theo các gốc độ khác nhau (translation, rotation, scaling) thì độ chính xác của thuật toán sẽ bị ảnh hưởng đáng kể.

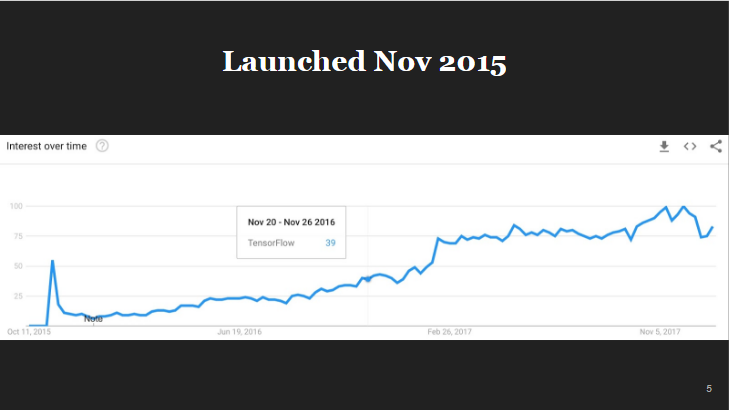
Pooling layer sẽ cho bạn tính bất biến đối với phép dịch chuyển (translation), phép quay (rotation) và phép co giãn (scaling). Tính kết hợp cục bộ cho ta các cấp độ biểu diễn thông tin từ mức độ thấp đến mức độ cao và trừu tượng hơn thông qua convolution từ các filter.

Đó là lý do tại sao CNNs cho ra mô hình với độ chính xác rất cao. Cũng giống như cách con người nhận biết các vật thể trong tự nhiên.

## 2.3. Thư viện Tensorflow

TensorFlow [9] là một thư viện phần mềm mở cho tính toán số, sử dụng biểu đồ luồng dữ liệu. Các nút trong đồ thị biểu diễn cho hoạt động toán học, trong khi các cạnh đồ thị biểu diễn cho các mảng dữ liệu đa chiều (tensors) trao đổi giữa chúng. Kiến trúc linh hoạt cho phép chúng ta triển khai tính toán trên một hoặc nhiều CPU hoặc GPU trong một máy tính để bàn, máy chủ, hoặc thiết bị di động với một API đơn. TensorFlow ban đầu được phát triển bởi các nhà nghiên cứu và kỹ sư làm việc trong nhóm Google Brain cho các nghiên cứu máy học và deep neural network.

TensorFlow có các API với một số ngôn ngữ lập trình cho cả xây dựng và thực thi một đồ thị TensorFlow. Python API là hiện tại hoàn thiện nhất và dễ sử dụng nhất, nhưng API C++ có một vài ưu điểm về hiệu năng trong việc thực thi đồ thị, và hỗ trợ triển khai các thiết bị nhỏ như Android.



**Mức độ phổ biến của Tensorflow tính từ lúc được Opensource.**

Một số project nổi tiếng sử dụng thư viện Tensorflow:

* Phân loại ung thư da – Dermatologist-level classification of skin cancer with deep neural networks (Esteva et al., Nature 2017)
* WaveNet: Text to speech – Wavenet: A generative model for raw audio (Oord et al., 2016)
* Vẽ hình – Draw Together with a Neural Network (Ha et al., 2017)
* Image Style Transfer Using Convolutional Neural Networks (Gatys et al., 2016) Tensorflow adaptation by Cameroon Smith (cysmith@github)

## 2.4. Thư viện Keras

**Keras** là một thư viện nơ-ron mã nguồn mở được viết bằng ngôn ngữ lập trình Python. Nó có khả năng chạy trên đầu trang của Tensorflow, Microsoft Cognitive Toolkit, R, Theano hoặc StripeML. Được thiết kế để cho phép thử nghiệm nhanh với các mạng thần kinh sâu, nó tập trung vào việc thân thiện với người dùng, mô-đun và mở rộng.

**Keras** được coi là một thư viện ‘high-level’ với phần ‘low-level’ (còn được gọi là backend) có thể là TensorFlow, CNTK, hoặc Theano (sắp tới Theano sẽ không được duy trì nâng cấp nữa). Keras có cú pháp đơn giản hơn TensorFlow rất nhiều. Với mục đích giới thiệu về các mô hình nhiều hơn là các sử dụng các thư viện deep learning, tôi sẽ chọn Keras với TensorFlow là ‘backend’.

Một số tính năng của Keras:

* Keras ưu tiên trải nghiệm của người lập trình
* Keras đã được sử dụng rộng rãi trong doanh nghiệp và cộng đồng nghiên cứu
* Keras giúp dễ dàng biến các thiết kế thành sản phẩm
* Keras hỗ trợ huấn luyện trên nhiều GPU phân tán
* Keras hỗ trợ đa backend engines và không giới hạn bạn vào một hệ sinh thái

## 2.5. Giới thiệu Google colab

Google Colab là một phiên bản lưu trữ trên đám mây giống với Jupyter Notebook do Google Research phát triển. Bằng cách sử dụng Google Colab, người dùng không cần cài đặt và nâng cấp phần cứng máy tính cá nhân mà vẫn có thể xử lý các công việc nặng về CPU/GPU trong Python. Colab cung cấp miễn phí quyền truy cập vào các hạ tầng điện toán đám mây như bộ lưu trữ, bộ nhớ, GPU (đơn vị xử lý đồ hoạ), CPU (khả năng xử lý) và TPU (đơn vị xử lý tensor).

Google đã tạo ra công cụ mã hóa Python dựa trên nền tảng đám mây này để đáp ứng nhu cầu của lập trình viên máy học, nhà khoa học dữ liệu, nhà phân tích dữ liệu, nhà nghiên cứu trí tuệ nhân tạo và người học ngôn ngữ lập trình Python.

Một điểm đáng chú ý là Google Colab cung cấp một sổ ghi chú code hoàn chỉnh, cho phép trình bày một dự án khoa học dữ liệu hoặc máy học đầy đủ cho các người giám sát hoặc nhà tài trợ. Sổ ghi chú Colab có thể chứa mã thực thi, mã Python trực tiếp, HTML, LaTeX, văn bản định dạng, hình ảnh, biểu đồ, bảng,…

# PHẦN III: TRIỂN KHAI XÂY DỰNG

## 3.1. Điều kiện tiên quyết

Yêu cầu với cấu hình mặc định:

* Phiên bản của GPU Nvidia: Card đồ họa NVIDIA có ít nhất 1.6 GB (lệnh nvidia-smi kiểm tra bộ nhớ GPU sẵn có trong Ubuntu). Ít nhất 2GB RAM bộ nhớ miễn phí.
* Khuyến khích sử dụng cuDNN.
* CPU: 8GB RAM bộ nhớ.
* Sẽ rất tốt nếu CPU có nhiều hơn 8 lõi.

## 3.2. Môi trường thực hiện

* Google Colab
* Ngôn ngữ Python 3
* Thư viện Tensorflow 2.12.0
* CPU Intel(R) Core(TM) i5-6300U CPU @ 2.40GHz, 2496 Mhz, 2 Core(s), 4 Logical Processor(s)
* Hệ điều hành: Window 10 Pro

## 3.3. Các bước thực hiện

### 3.3.1 Tạo dữ liệu và kết nối Colab với Drive

Bước đầu tiên ta cần chuẩn bị dữ liệu ảnh để train, sau đó gán nhãn dữ liệu bằng LblImage. Khi gán nhãn xong ta Upload dữ liệu lên Google Drive, chia dữ liệu vào images và labels ở thư mục raw\_data(thư mục con trong thư mục data), thư mục images chứa dữ liệu ảnh còn labels chứa nhãn của từng bức ảnh.

Bước thứ hai ta chọn loại thời gian chạy là T4 GPU ( nếu chưa mua thêm đơn vị điện toán) vì sử dụng GPU tối ưu hơn CPU, TPU và GPU xử lý dữ liệu nhanh hơn, hoặc ta cũng có thể chọn CPU,TPU tuỳ theo nhu cầu và mục đích xử dụng.

Tiếp theo ta kết nối với Drive bằng đoạn code:

# 0. Link Colab với Drive

from google.colab import drive

drive.mount('/content/gdrive')

%cd /content/gdrive/MyDrive

Thông thường nếu không có lỗi sẽ xuất ra kết quả:



Còn nếu có lỗi thì ta kiểm tra lại đoạn code, tài khoản Drive xem đã đúng chưa.

### 3.3.2 Tiền xử lý hình ảnh

# 1. Tạo thư mục data/split\_data và chạy lệnh chia train test

import os

import glob

import random

import shutil

raw\_data\_path = "/content/gdrive/MyDrive/data/raw\_data/images"

train\_data\_path = "/content/gdrive/MyDrive/data/split\_data/train"

test\_data\_path = "/content/gdrive/MyDrive/data/split\_data/test"

if not os.path.exists("/content/gdrive/MyDrive/data/split\_data/"):

  os.mkdir("/content/gdrive/MyDrive/data/split\_data/")

try:

  shutil.rmtree(train\_data\_path)

  shutil.rmtree(test\_data\_path)

except:

  pass

os.mkdir(train\_data\_path)

os.mkdir(test\_data\_path)

total\_files = glob.glob(raw\_data\_path + "/\*.jpg")

print("Sample file = ", total\_files[0])

print("Total file = ", len(total\_files))

indices = list(range(len(total\_files)))

train\_indices = random.sample(indices, k = int(len(total\_files)\*0.9))

print("Total train file = ", len(train\_indices))

for i in indices:

    if (i in train\_indices):

        # Copy to train

        shutil.copy(total\_files[i], train\_data\_path + total\_files[i].replace(raw\_data\_path,""))

        shutil.copy(total\_files[i].replace("images","labels").replace(".jpg",".xml"),

                    train\_data\_path + total\_files[i].replace(raw\_data\_path, "").replace(".jpg",".xml"))

    else:

        # Copy to test

        shutil.copy(total\_files[i], test\_data\_path + total\_files[i].replace(raw\_data\_path, ""))

        shutil.copy(total\_files[i].replace("images", "labels").replace(".jpg", ".xml"),

                    test\_data\_path + total\_files[i].replace(raw\_data\_path, "").replace(".jpg",".xml"))

# 2. Chuyển XML label sang CSV. File CSV lưu trong thư mục split\_data

import os

import glob

import pandas as pd

import xml.etree.ElementTree as ET

def xml\_to\_csv(path):

    xml\_list = []

    for xml\_file in glob.glob(path + '/\*.xml'):

        tree = ET.parse(xml\_file)

        root = tree.getroot()

        for member in root.findall('object'):

            value = (root.find('filename').text,

                     int(root.find('size')[0].text),

                     int(root.find('size')[1].text),

                     member[0].text,

                     int(member[4][0].text),

                     int(member[4][1].text),

                     int(member[4][2].text),

                     int(member[4][3].text)

                     )

            xml\_list.append(value)

    column\_name = ['filename', 'width', 'height', 'class', 'xmin', 'ymin', 'xmax', 'ymax']

    xml\_df = pd.DataFrame(xml\_list, columns=column\_name)

    return xml\_df

for directory in ['train','test']:

        image\_path = os.path.join('/content/gdrive/MyDrive/data/split\_data/{}'.format(directory))

        xml\_df = xml\_to\_csv(image\_path)

        xml\_df.to\_csv('/content/gdrive/MyDrive/data/split\_data/{}\_labels.csv'.format(directory), index=None)

        print('Successfully converted xml to csv.')

# 3. Tải TF Object Detection APi vào thư mục models

%cd /content/gdrive/MyDrive

!git clone https://github.com/tensorflow/models.git

# 4. Biên dịch các protos (Bộ đệm giao thức protobuf là định dạng dữ liệu đa nền tảng nguồn mở và miễn phí được sử dụng để tuần tự hoá dữ liệu)

%cd /content/gdrive/MyDrive/models/research

!protoc object\_detection/protos/\*.proto --python\_out=.

# 5. Cài dặt API

%cd /content/gdrive/MyDrive/models/research

!cp object\_detection/packages/tf2/setup.py .

!python -m pip install .

# 6. Chuyển từ CSV thành TFRecord.

%cd /content/gdrive/MyDrive/models/research

!git clone https://github.com/ViMinhChau/Tensorflow\_tfrecord

!cp Tensorflow\_tfrecord/generate\_tfrecord.py .

!cp Tensorflow\_tfrecord/label\_map.txt /content/gdrive/MyDrive/data

if not os.path.exists("/content/gdrive/MyDrive/data/tfrecord\_data/"):

  os.mkdir("/content/gdrive/MyDrive/data/tfrecord\_data/")

!python generate\_tfrecord.py --image\_dir=/content/gdrive/MyDrive/data/split\_data/train --csv\_input=/content/gdrive/MyDrive/data/split\_data/train\_labels.csv --output\_path=/content/gdrive/MyDrive/data/tfrecord\_data/train.record

!python generate\_tfrecord.py --image\_dir=/content/gdrive/MyDrive/data/split\_data/test --csv\_input=/content/gdrive/MyDrive/data/split\_data/test\_labels.csv --output\_path=/content/gdrive/MyDrive/data/tfrecord\_data/test.record

# 7. Tải xuống pretrain và config

if not os.path.exists("/content/gdrive/MyDrive/pretrained"):

  os.mkdir("/content/gdrive/MyDrive/pretrained")

%cd /content/gdrive/MyDrive/pretrained

!wget http://download.tensorflow.org/models/object\_detection/tf2/20200711/ssd\_mobilenet\_v2\_fpnlite\_320x320\_coco17\_tpu-8.tar.gz

# Unzip

!tar -xzvf ssd\_mobilenet\_v2\_fpnlite\_320x320\_coco17\_tpu-8.tar.gz

# Cấu hình pipeline.config, label\_map.txt nếu cần thiết

### 3.3.3 Huấn Luyện

# 8. Train

%cd /content/gdrive/MyDrive/models/research

%cp /content/gdrive/MyDrive/models/research/object\_detection/model\_main\_tf2.py .

#train

!python model\_main\_tf2.py \

--pipeline\_config\_path=/content/gdrive/MyDrive/pretrained/ssd\_mobilenet\_v2\_fpnlite\_320x320\_coco17\_tpu-8/pipeline.config\

--model\_dir=/content/gdrive/MyDrive/output\_model --alsologtostderr --num\_train\_steps=2000

# 9. Export model

%cd /content/gdrive/MyDrive/models/research

%cp /content/gdrive/MyDrive/models/research/object\_detection/exporter\_main\_v2.py .

!python exporter\_main\_v2.py \

--trained\_checkpoint\_dir=/content/gdrive/MyDrive/output\_model \

--pipeline\_config\_path=/content/gdrive/MyDrive/pretrained/ssd\_mobilenet\_v2\_fpnlite\_320x320\_coco17\_tpu-8/pipeline.config \

--output\_directory=/content/gdrive/MyDrive/export\_model

import io

import os

import scipy.misc

import numpy as np

import six

import time

import glob

from IPython.display import display

from six import BytesIO

import matplotlib

import matplotlib.pyplot as plt

from PIL import Image, ImageDraw, ImageFont

import tensorflow as tf

from object\_detection.utils import ops as utils\_ops

from object\_detection.utils import label\_map\_util

from object\_detection.utils import visualization\_utils as vis\_util

#Load model

tf.keras.backend.clear\_session()

model = tf.saved\_model.load("/content/gdrive/MyDrive/export\_model/saved\_model")

# Các hàm inference

import cv2

def run\_inference\_for\_single\_image(model, image):

  image = np.asarray(image)

  input\_tensor = tf.convert\_to\_tensor(image)

  input\_tensor = input\_tensor[tf.newaxis,...]

  model\_fn = model.signatures['serving\_default']

  output\_dict = model\_fn(input\_tensor)

  num\_detections = int(output\_dict.pop('num\_detections'))

  output\_dict = {key:value[0, :num\_detections].numpy()

                 for key,value in output\_dict.items()}

  output\_dict['num\_detections'] = num\_detections

  output\_dict['detection\_classes'] = output\_dict['detection\_classes'].astype(np.int64)

  if 'detection\_masks' in output\_dict:

    detection\_masks\_reframed = utils\_ops.reframe\_box\_masks\_to\_image\_masks(

              output\_dict['detection\_masks'], output\_dict['detection\_boxes'],

               image.shape[0], image.shape[1])

    detection\_masks\_reframed = tf.cast(detection\_masks\_reframed > 0.5,

                                       tf.uint8)

    output\_dict['detection\_masks\_reframed'] = detection\_masks\_reframed.numpy()

  return output\_dict

def load\_image\_into\_numpy\_array(path):

  img\_data = tf.io.gfile.GFile(path, 'rb').read()

  image = Image.open(BytesIO(img\_data))

  (im\_width, im\_height) = image.size

  return np.array(image.getdata()).reshape(

      (im\_height, im\_width, 3)).astype(np.uint8)

### 3.3.4 Test

%cd /content/gdrive/MyDrive/models/research

category\_index = label\_map\_util.create\_category\_index\_from\_labelmap("/content/gdrive/MyDrive/data/label\_map.txt", use\_display\_name=True)

image\_path = '/content/gdrive/MyDrive/data/split\_data/train/250.jpg'

image\_np = load\_image\_into\_numpy\_array(image\_path)

print("Done load image ")

image\_np = cv2.resize(image\_np, dsize=None, fx=0.2, fy=0.2)

output\_dict = run\_inference\_for\_single\_image(model, image\_np)

print("Done inference")

vis\_util.visualize\_boxes\_and\_labels\_on\_image\_array(

    image\_np,

    output\_dict['detection\_boxes'],

    output\_dict['detection\_classes'],

    output\_dict['detection\_scores'],

    category\_index,

    instance\_masks=output\_dict.get('detection\_masks\_reframed', None),

    use\_normalized\_coordinates=True,

    line\_thickness=8)

print("Done draw on image ")

display(Image.fromarray(image\_np))





# PHẦN IV: KẾT LUẬN

## 4.1. Ưu điểm - nhược điểm

* *Ưu điểm*
* Phương pháp nhận diện giày dép sử dụng TensorFlow đã đạt được kết quả tốt. Mô hình nhận diện giày đạt được độ chính xác cao và có khả năng phân loại chính xác giữa các loại giày phổ biến.
* Sử dụng mạng neural tích chập (CNN) đã cho phép mô hình học và rút trích các đặc trưng đáng chú ý từ hình ảnh giày dép. Điều này giúp cải thiện khả năng nhận diện và phân loại của mô hình.
* Việc sử dụng các kỹ thuật tiền xử lý dữ liệu như chuẩn hóa kích thước hình ảnh đã giúp đảm bảo tính nhất quán và hiệu suất của quá trình huấn luyện.
* Các cải tiến và tối ưu hóa mô hình đã được thực hiện để cải thiện hiệu suất và tốc độ xử lý. Việc tinh chỉnh siêu tham số, áp dụng transfer learning và sử dụng các kiến trúc mạng khác nhau đã cải thiện đáng kể hiệu suất của mô hình.
* *Nhược điểm*
* Mặc dù đã đạt được kết quả tốt, mô hình nhận diện giày dép vẫn có thể gặp khó khăn khi đối mặt với các hình ảnh chất lượng thấp, góc chụp không lý tưởng hoặc nhiễu. Điều này có thể làm giảm độ chính xác và tin cậy của mô hình trong một số trường hợp.
* Quá trình huấn luyện mô hình có thể tốn nhiều thời gian và tài nguyên tính toán, đặc biệt khi sử dụng các kiến trúc mạng phức tạp và tập dữ liệu lớn. Điều này có thể làm hạn chế khả năng triển khai và sử dụng mô hình trên các thiết bị có tài nguyên hạn chế.
* Mô hình nhận diện giày dép hiện tại chỉ tập trung vào một số loại giày phổ biến. Việc mở rộng và đa dạng hóa danh mục loại giày có thể đòi hỏi việc thu thập và gắn nhãn thêm nhiều dữ liệu huấn luyện, cũng như tinh chỉnh lại mô hình để phù hợp với các loại giày mới.

## 4.2. Hướng phát triển

Mở rộng khả năng phân loại: hiện tại, đồ án tập trung vào việc nhận diện các loại giày dép phổ biến như sneaker, sandal, high heels, flip-flops, soccer shoes, boot và loafer. Một hướng phát triển tiếp theo có thể là mở rộng hệ thống để phân loại các loại giày khác như chunky loafer, combat boot, oxford shoes, hay slippers. Điều này đòi hỏi việc thu thập thêm dữ liệu và huấn luyện mô hình trên các loại giày mới.

Tăng cường dữ liệu (Data augmentation): Data augmentation là một kỹ thuật được sử dụng để mở rộng tập dữ liệu huấn luyện bằng cách áp dụng các biến đổi nhỏ như xoay, phóng đại, cắt tỉa, hay lật ngang và lật dọc cho các hình ảnh. Việc áp dụng kỹ thuật này có thể giúp cải thiện khả năng tổng quát hóa của mô hình khi đối mặt với các hình ảnh mới.

Sử dụng các mô hình deep learning tiên tiến khác: Mặc dù mô hình CNN đã được sử dụng trong đồ án này, có nhiều mô hình deep learning khác có thể được khám phá. Ví dụ, mạng neural gia đình EfficientNet đã chứng minh hiệu suất cao trên nhiều bài toán nhận diện hình ảnh. Thử nghiệm và so sánh hiệu suất của các mô hình khác nhau có thể là một hướng phát triển thú vị.

Tối ưu hóa hiệu suất mô hình: Có nhiều cách để tối ưu hóa hiệu suất của mô hình. Bằng cách thay đổi kiến trúc mạng, thay đổi siêu tham số, hay sử dụng kỹ thuật như transfer learning, fine-tuning, hay ensemble learning, người dùng có thể cải thiện hiệu suất và độ chính xác của mô hình nhận diện giày dép.

Tích hợp vào ứng dụng thực tế: Sau khi xây dựng mô hình nhận diện giày dép, một hướng phát triển tiếp theo là tích hợp mô hình này vào một ứng dụng thực tế. Ví dụ: phát triển một ứng dụng di động cho phép người dùng chụp ảnh giày dép và mô hình sẽ nhận diện và cung cấp thông tin về loại giày đó.

Nâng cao hiệu suất và tốc độ xử lý: Đối với các ứng dụng có yêu cầu thời gian thực hoặc xử lý hàng loạt hình ảnh lớn, việc tối ưu hóa hiệu suất và tốc độ xử lý của mô hình là rất quan trọng. Người dùng có thể xem xét sử dụng các kỹ thuật như quantization, pruning, hay sử dụng phần cứng đặcialized như GPU hay TPU để đạt được hiệu suất và tốc độ tốt hơn.