ỨNG DỤNG MẠNG NƠRON TÍCH CHẬP PHÁT HIỆN NGƯỜI ĐI BỘ

MỤC LỤC

DANH MỤC CÁC HÌNH VẼ

MỞ ĐẦU

Thuật toán khởi nguồn của Deep Learning là Neural Network đã ra đời từ năm 1943, tức khoảng hơn 70 năm trước và trải qua rất nhiều năm thăng trầm phát triển thì hiện nay Deep Learning đang là một trong những từ khóa hot nhất trong lĩnh vực công nghệ AI (Antificial Intelligence) hiện nay. Một trong những ứng dụng mạnh mẽ của Deep Learning là trong xử lý ảnh, nhận dạng và phát hiện đối tượng. Convolutional Neural Networks (CNNs) là một trong những mô hình được áp dụng rộng rãi nhất. Ứng dụng mạng CNNs cho bài toán phát hiện người đi bộ có thể giải quyết được nhiều vấn đề trong thực tiễn như xe tự lái, hệ thống ngăn cách cho người đi bộ, cảnh báo các phương tiện có người đi bộ qua đường.

Chương 1. TỔNG QUAN TÀI LIỆU

Một trong những phương pháp tốt nhất và nhanh nhất hiện nay là YOLO(You Only Look Once). Yolo là một mô hình mạng CNN cho việc phát hiện, nhận dạng, phân loại đối tượng. Tuy nhiên nó lại đang làm việc với quá nhiều đối tượng. Cụ thể YOLO có thể phát hiện đến 80 đối tượng. Vì vậy tốc độ xử lý cũng như độ chính xác vẫn chưa cao so với chỉ giải quyết một đối tượng. Trong nội dung của tiểu luận sẽ đề cập đến việc điều chỉnh các tham số của mạng YOLO để phù hợp với bài toán phát hiện người đi bộ. Mạng mới được huấn luyện dựa trên bộ dữ liệu được gán nhãn (đánh dấu tọa độ của người trong ảnh) và trọng số của mạng có sẵn.

Chương 2. CƠ SỞ LÝ THUYẾT VÀ MÔ HÌNH NGHIÊN CỨU

2.1. Convolution là gì?

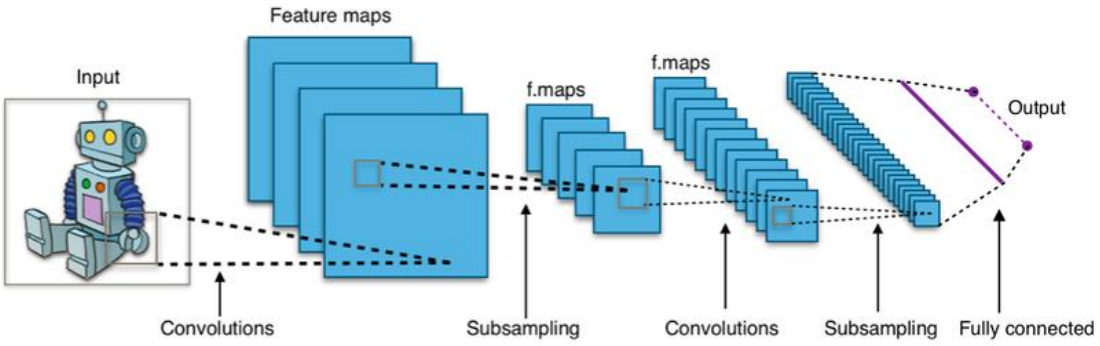
-Convolution (tích chập) là toán tử mà ta thực hiện xoay cửa sổ 180 độ (flip over, tức flip 2 lần lần lượt theo trục x và y) rồi sau đó áp dụng phép correlation (tương quan).

-Mỗi bức ảnh đều có các điểm ảnh nhất định được hiểu là các ma trận ảnh. Convolution layer được học để điều chỉnh lấy ra những thông tin chính xác mà không cần chọn ra các feature. Điều này được thực hiện bằng cách chia nhỏ bước ảnh thành các ma trận nhỏ hơn (Sliding Window) rồi nhân với một ma trận được gọi là Filter hoặc Feature Detect. Kết quả là một ma trận Convoled Feature.

2.2. Kiến trúc mạng CNNs

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. 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.

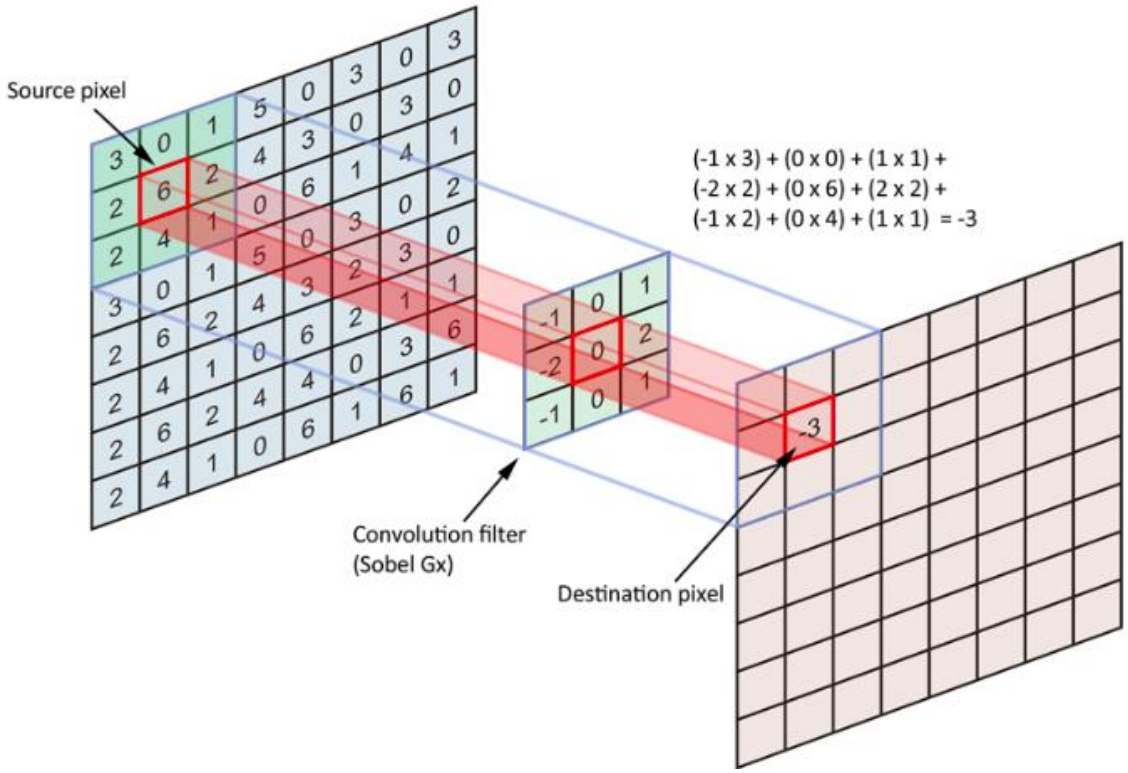
Mỗi một lớp được sử dụng các filter khác nhau thông thường có hàng trăm hàng nghìn filter như vậy và kết hợp kết quả của chúng lại. Ngoài ra có một số layer khác như pooling/subsampling layer dùng để chắt lọc lại các thông tin hữu ích hơn (loại bỏ các thông tin nhiễu). Trong quá trình huấn luyện mạng (traning) CNN tự động học các giá trị qua các lớp filter dựa vào cách thức mà bạn thực hiện.



Hình 2. 1 :Kiến trúc mạng Convolution Neural Networks

Một mạng CNNs thường có ba loại lớp: lớp tích chập (convolution layer), lớp tổng hợp (pooling layer) , lớp kết nối đầy đủ (fully connected layer)

a. Lớp tích chập (convolution layer)



Hình 2. 2.Mô hình hoạt động của lớp tích chập

- Lớp tích chập là khối xây dựng cốt lõi của CNN. Nó mang phần chính của tải tính toán của mạng.
- Lớp này thực hiện tích điểm giữa hai ma trận, trong đó một ma trận là tập hợp các tham số có thể học được còn được gọi là nhân và ma trận còn lại là phần bị hạn chế của trường tiếp nhận. Về mặt không gian kernel nhỏ hơn hình ảnh nhưng có chiều sâu hơn. Điều này có nghĩa là, nếu hình ảnh bao gồm ba kênh (RGB),

chiều cao và chiều rộng của hạt nhân sẽ nhỏ về mặt không gian, nhưng chiều sâu mở rộng đến cả ba kênh.

- Trong quá trình chuyển tiếp, hạt nhân trượt qua chiều cao và chiều rộng của hình ảnh, tạo ra hình ảnh biểu diễn của vùng tiếp nhận đó. Điều này tạo ra một biểu diễn hai chiều của hình ảnh được gọi là bản đồ kích hoạt cung cấp phản hồi của hạt nhân tại mỗi vị trí không gian của hình ảnh. Kích thước trượt của hạt nhân được gọi là sải chân.
- Nếu chúng ta có đầu vào có kích thước W x W x D và số hạt nhân D_{out} có kích thước không gian là F với sải chân S và lượng đệm P, thì kích thước của khối lượng đầu ra có thể được xác định theo công thức sau:

$$W_{out} = \frac{W - F + 2P}{S} + 1$$

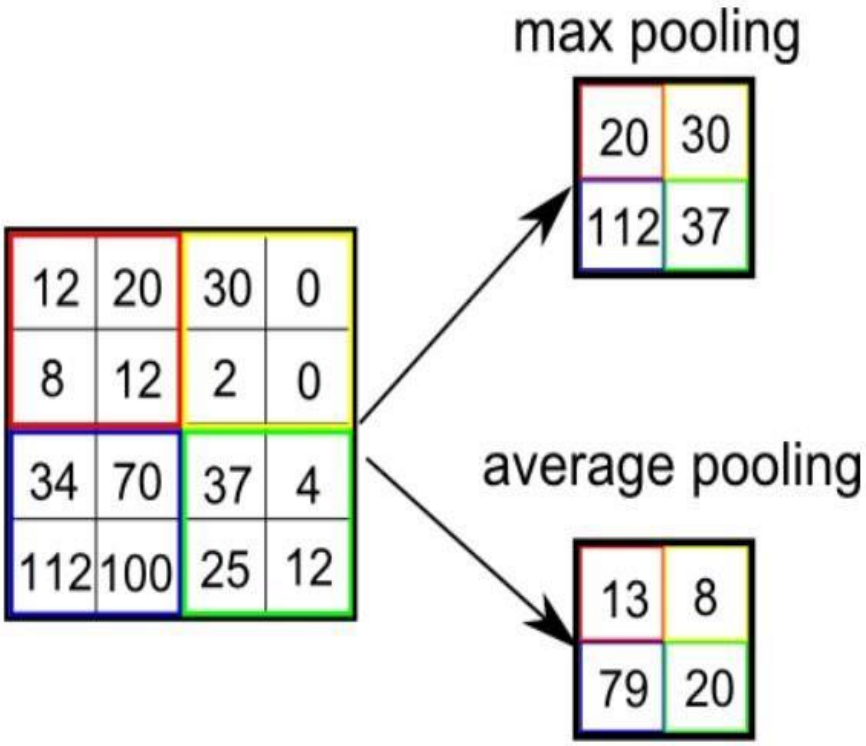
Convolution thúc đẩy ba ý tưởng quan trọng thúc đẩy các nhà nghiên cứu thị giác máy tính: tương tác thưa thớt, chia sẻ tham số và biểu diễn tương đương.

+Tương tác thưa thớt : một hình ảnh có thể có hàng triệu hoặc hàng nghìn pixel, nhưng trong khi xử lý nó bằng cách sử dụng hạt nhân, chúng ta có thể phát hiện thông tin có ý nghĩa có hàng chục hoặc hàng trăm pixel. Điều này có nghĩa là chúng ta cần lưu trữ ít tham số hơn, không chỉ làm giảm yêu cầu bộ nhớ của mô hình mà còn cải thiện hiệu quả thống kê của mô hình.

+Chia sẻ tham số: mạng tích chập có các tham số được chia sẻ, tức là để nhận được đầu ra, trọng số áp dụng cho một đầu vào giống với trọng số áp dụng ở nơi khác.

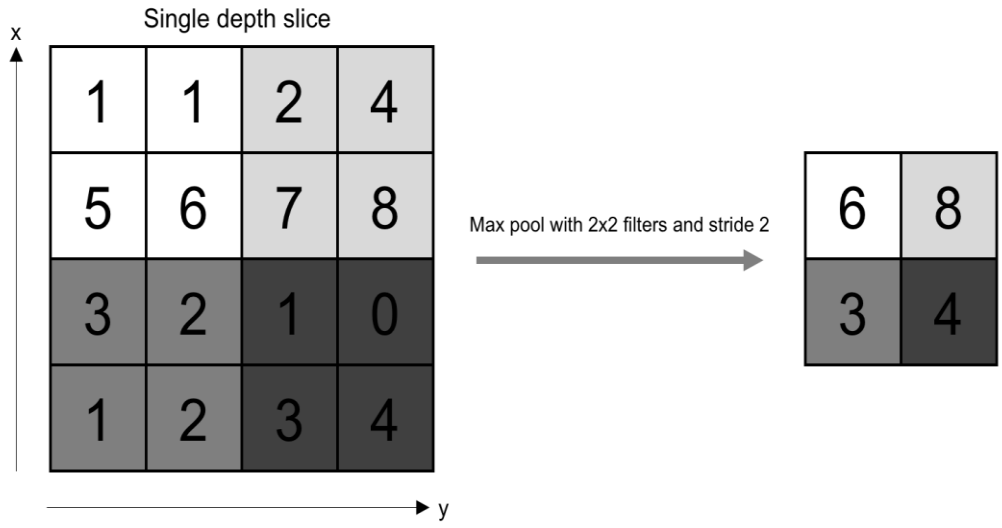
+Biểu diễn tương đương: Do chia sẻ tham số, các lớp của mạng nơ ron tích chập sẽ có thuộc tính tương đương với phép dịch. Nếu chúng ta thay đổi đầu vào theo một cách nào đó, thì đầu ra cũng sẽ thay đổi theo cách tương tự.

b. Lớp tổng hợp (pooling layers)



Hình 2. 3.Hai loại Pooling

- Lớp gộp thay thế đầu ra của mạng tại một số vị trí nhất định bằng cách lấy thống kê tóm tắt về các đầu ra lân cận. Điều này giúp giảm kích thước không gian của biểu diễn, làm giảm số lượng tính toán và trọng số cần thiết. Thao tác gộp được xử lý trên từng lát của biểu diễn riêng lẻ.
- Có một số hàm tổng hợp như giá trị trung bình của vùng lân cận hình chữ nhật, chuẩn L2 của vùng lân cận hình chữ nhật và giá trị trung bình có trọng số dựa trên khoảng cách từ pixel trung tâm. Tuy nhiên, quy trình phổ biến nhất là gộp tối đa, báo cáo kết quả đầu ra tối đa từ vùng lân cận.



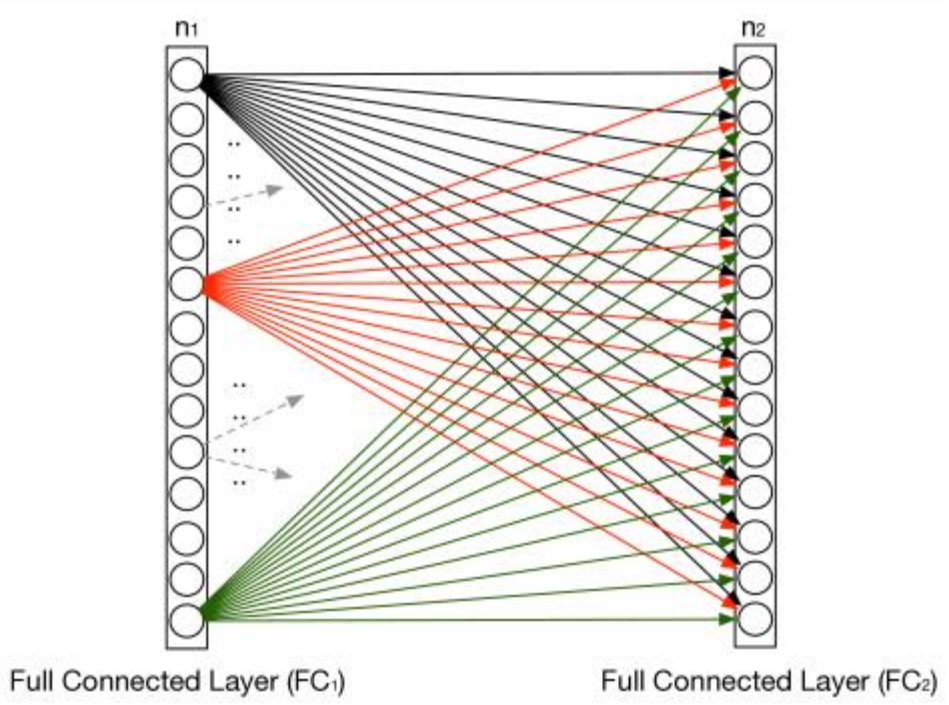
Hình 2. 4.Max-Pooling

Nếu chúng ta có một bản đồ kích hoạt có kích thước W x W x D, một nhân tổng hợp có kích thước không gian F và sải chân S, thì kích thước của khối lượng đầu ra có thể được xác định theo công thức sau:

$$W_{out} = \frac{W - F}{S} + 1$$

Điều này sẽ mang lại một khối lượng đầu ra có kích thước W_{out} x W_{out} x D . Trong mọi trường hợp, việc gộp chung cung cấp một số bất biến dịch chuyển có nghĩa là một đối tượng sẽ có thể nhận ra được bất kể nó xuất hiện ở đâu trên khung hình.

c. Lớp kết nối đầy đủ (fully connected layer)



Hình 2. 5.Lớp kết nối đầy đủ

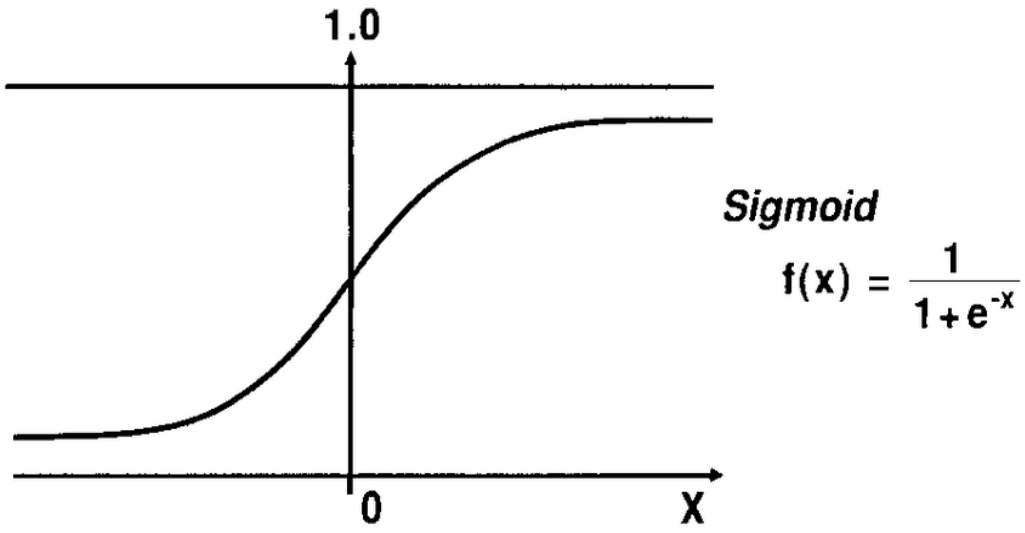
- Các tế bào thần kinh trong lớp này có khả năng kết nối đầy đủ với tất cả các tế bào thần kinh ở lớp trước và lớp kế tiếp như được thấy trong FCNN thông thường. Đây là lý do tại sao nó có thể được tính như bình thường bằng phép nhân ma trận theo sau là hiệu ứng thiên vị. Nó giúp lập bản đồ biểu diễn giữa đầu vào và đầu ra.

d. Lớp phi tuyến (non-linearity layers)

- Vì tích chập là một phép toán tuyến tính và hình ảnh ở xa tuyến tính, các lớp phi tuyến tính thường được đặt ngay sau lớp tích chập để đưa tính phi tuyến tính vào bản đồ kích hoạt.

Một số hàm phi tuyến phổ biến là :

+Signmoid : Tính phi tuyến tính sigmoid có dạng toán

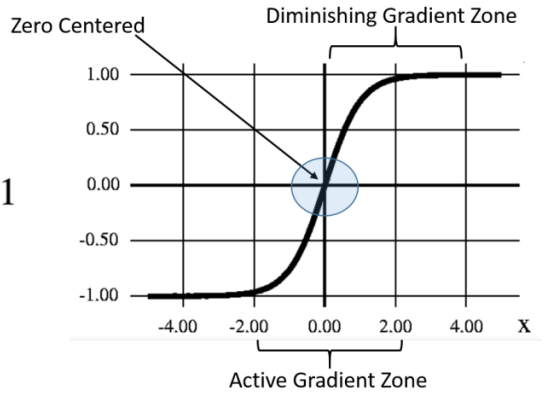


Hình 2. 6.Đồ thị hàm sigmoid

Nó nhận một số có giá trị thực và "sắp xếp" nó thành một phạm vi từ 0 đến 1. Tuy nhiên, một đặc tính không mong muốn của sigmoid là khi kích hoạt ở một trong hai đuôi, gradient gần như bằng không. Nếu gradient cục bộ trở nên rất nhỏ, thì trong quá trình lan truyền ngược, nó sẽ "giết" gradient một cách hiệu quả. Ngoài ra, nếu dữ liệu đi vào nơ-ron luôn dương, thì đầu ra của sigmoid sẽ là tất cả dương hoặc tất cả âm, dẫn đến zig-zag động cập nhật gradient cho trọng số.

+ Tanh : Tanh chọn một số có giá trị thực đến phạm vi [-1, 1]. Giống như sigmoid, kích hoạt bão hòa, nhưng - không giống như các nơron sigmoid - đầu ra của nó là 0 ở giữa.

$$f(x) = tanh(x) = \frac{2}{1+e^{-2x}} - 1$$

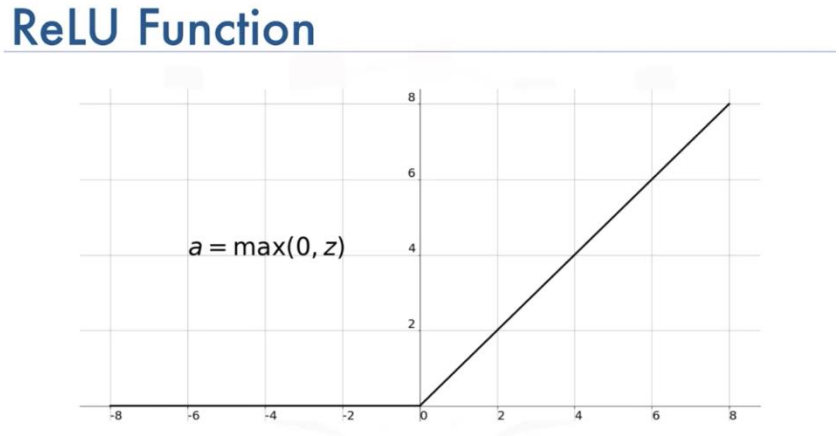


Hình 2. 7.Đồ thị hàm tanh

+ ReLU: Đơn vị tuyến tính chỉnh lưu (ReLU) đã trở nên rất phổ biến trong vài năm gần đây. Nó tính hàm $f(\kappa) = \max(0, \kappa)$. Nói cách khác, kích hoạt chỉ đơn giản là ngưỡng bằng không.

So với sigmoid và tanh, ReLU đáng tin cậy hơn và tăng tốc độ hội tụ gấp sáu lần.

Tuy nhiên, một sai lầm là ReLU có thể dễ vỡ trong quá trình đào tạo. Một gradient lớn chạy qua nó có thể cập nhật nó theo cách mà nơ-ron sẽ không bao giờ được cập nhật thêm. Tuy nhiên, chúng ta có thể g iải quyết vấn đề này bằng cách đặt một tỷ lệ học tập phù hợp.



Hình 2. 8.Đồ thị hàm ReLU

2.3.Kiến trúc mạng YOLO

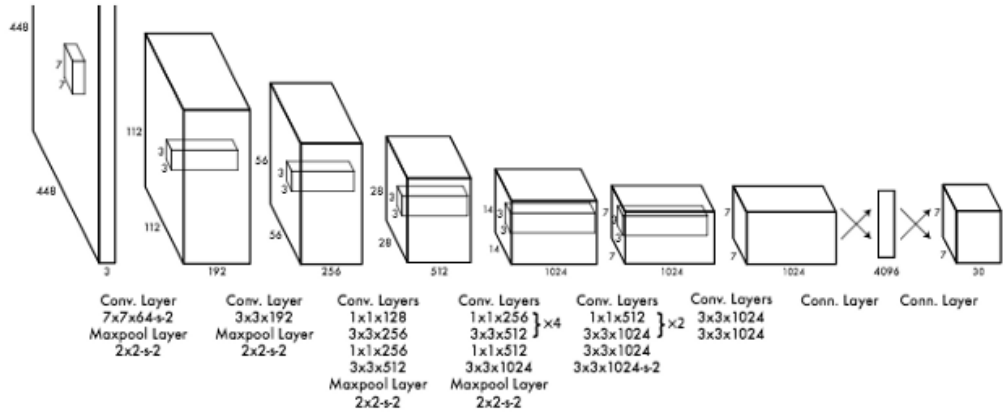
-Yolo xử lý vấn đề phát hiện hình ảnh như một vấn đề hồi quy chứ không phải là một vấn đề phân loại và hỗ trợ một mạng nơ-ron tích tụ duy nhất để thực hiện tất cả các tác vụ được đề cập ở trên. Việc hợp nhất tất cả các nhiệm vụ độc lập thành một mạng có những lợi ích:

+Tốc độ : YOLO có tốc độ cực nhanh so với các phiên bản tiền nhiệm vì nó sử dụng một mạng tích chập duy nhất để phát hiện các đối tượng. Phép chập được thực hiện trên toàn bộ hình ảnh đầu vào chỉ một lần để mang lại các dự đoán

+Ít lỗi nền : YOLO thực hiện tích chập trên toàn bộ hình ảnh chứ không phải các phần của nó do nó mã hóa thông tin theo ngữ cảnh về các lớp và sự xuất hiện của chúng. Nó ít mắc lỗi hơn trong việc dự đoán các bản vá nền dưới dạng các đối tượng vì nó xem toàn bộ hình ảnh và lý do trên toàn cầu thay vì cục bộ.

+Tổng quát cao : YOLO học các biểu diễn tổng quát của các đối tượng do đó nó có thể được áp dụng cho các miền mới và các đầu vào bất ngờ mà không bị hỏng.

-Yolo được triển khai dưới dạng mạng nơ-ron tích chập bao gồm tổng cộng 24 lớp chập, tiếp theo là 2 lớp được kết nối đầy đủ. Các lớp được phân tách theo chức năng của chúng .



Hình 2. 9.Kiến trúc mạng Yolo

Lớp cuối cùng sử dụng kích hoạt tuyến tính trong khi các lớp tích tụ khác sử dụng kích hoạt ReLU bị rò rỉ (f(x) = 0.01x nếu x<0).

Đầu vào là hình ảnh 448 x 448 và đầu ra là dự đoán lớp của đối tượng nằm trong hộp giới hạn.

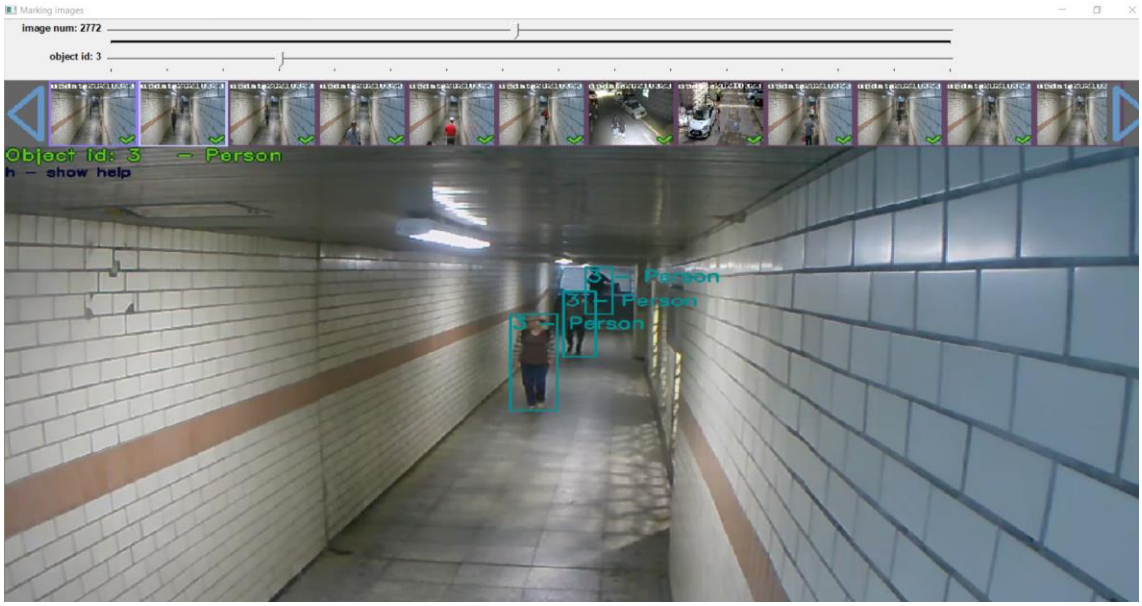
Chương 3. HUẤN LUYỆN YOLO TRÊN BỘ DỮ LIỆU RIÊNG

3.1.Chuẩn bị dữ liệu

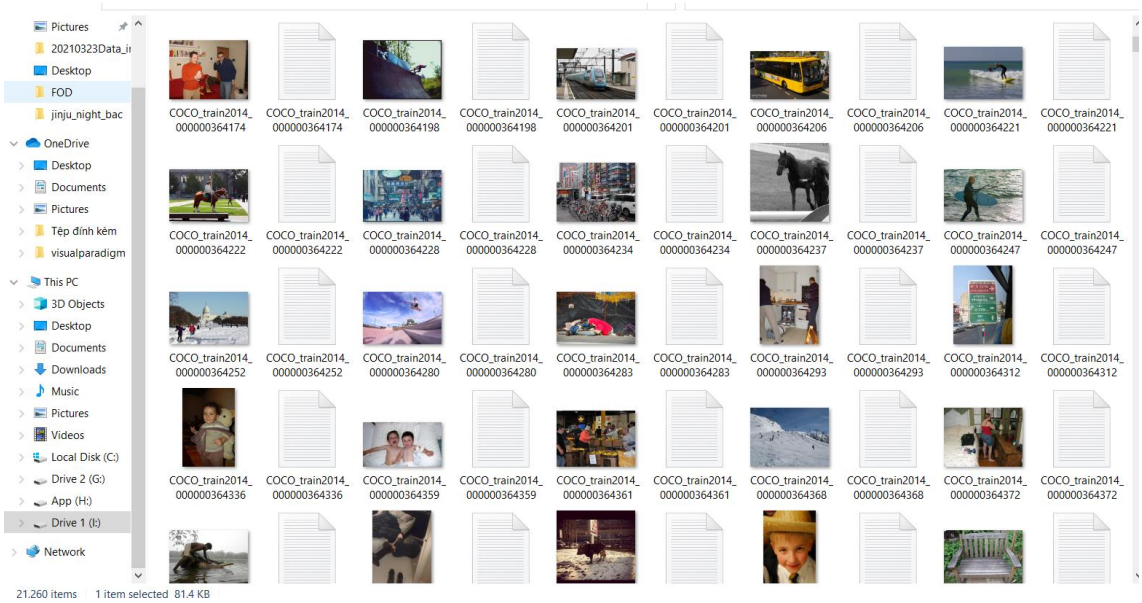
- Bộ dữ liệu sử dụng để huấn luyện là các ảnh được gán nhãn sẵn. Gán nhãn là việc xác định vị trí, tọa độ của hình chữ nhật bao quanh đối tượng trong ảnh (đối

tượng người). Các thông tin này được lưu thành tệp riêng. Có nhiều công cụ để thực hiện gán nhãn. Ví dụ: Yolo mark

-Khi thực hiện gán nhãn đối tượng, cần vẽ khung hình chữ nhật bao quanh toàn bộ đối tượng.



Hình 3. 1.Gán nhãn đối tượng người bằng Yolo mark



Hình 3. 2.Bộ dữ liệu hoàn chỉnh

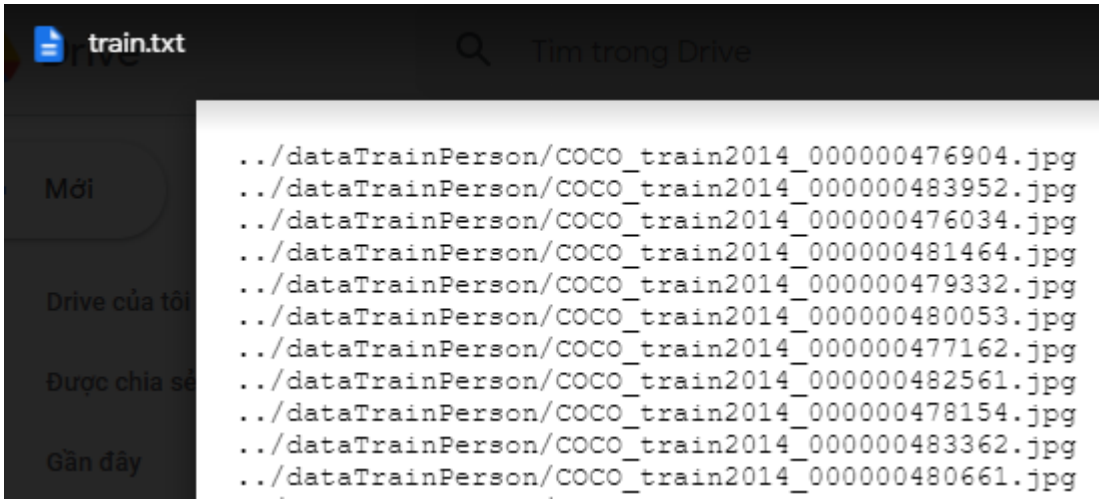
Thông tin tệp bao gồm : id lớp, tọa độ x (góc trái trên) , tọa độ y(góc trái trên), tọa độ x (góc phải dưới), tọa độ y(góc phải dưới).

- Chia bộ dữ liệu thành hai phần. Phần dành cho huấn luyện (training data) và phần dành cho kiểm tra (test data). Tỉ lệ phần huấn luyện nhiều hơn, khoảng 3:1.

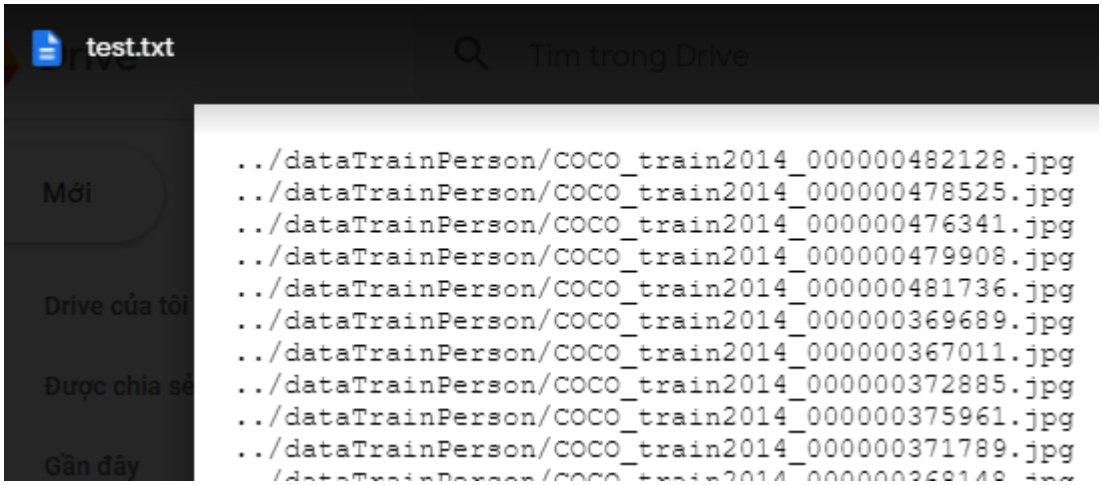
Cách chia dữ liệu:

+ Lấy ngẫu nhiên n ảnh và lưu tên vào tệp train.txt

+ Lưu tên số ảnh còn lại vào tệp test.txt



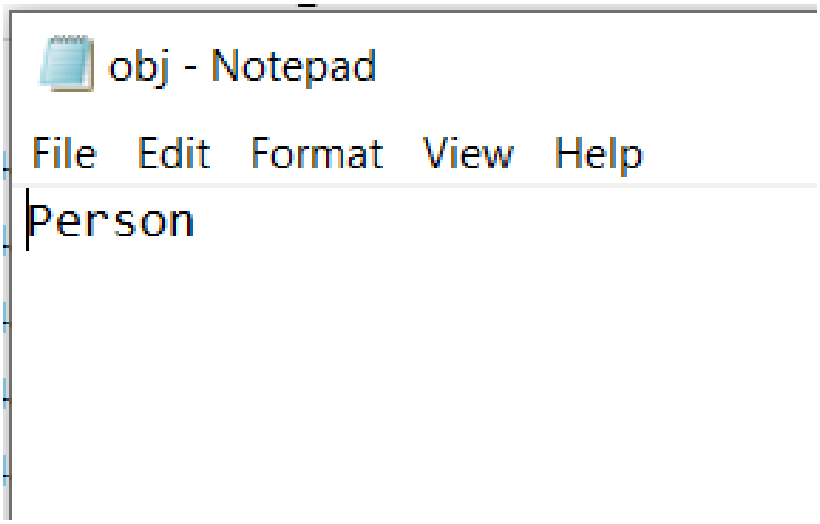
Hình 3. 3.Nội dung tệp train.txt



Hình 3. 4.Nội dung tệp test.txt

Tạo tệp obj.names chứa tên đối tượng cần phát hiện (Hình 3.5):

Ví dụ:



Hình 3. 5.Nội dung tệp obj.names

3.2. Thực hiện huấn luyện trên nền tảng Google Colab

- Do việc huấn luyện đòi hỏi cấu hình phần cứng khá cao nên Google Colab là một giải pháp dành cho việc học tập.

Cấu hình của máy chủ miễn phí:

```
Wed Apr  7 04:54:55 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 460.67       Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla K80           Off  | 00000000:00:04.0 Off |                    0 |
| N/A   72C    P8    35W / 149W |      0MiB / 11441MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Hình 3. 6.Cấu hình máy chủ Google Colab miễn phí

Các bước thực hiện huấn luyện:

- Tải bộ dữ liệu lên Google Drive.

- Mở một trang Google Colab để thực hiện code.

- Kết nối với Google Drive bằng lệnh:

```
from google.colab import drive
drive.mount('/content/mydrive')
```

```
Mounted at /content/mydrive
```

- Tải về thư mục darknet từ trang chủ github của darknet.

```
!git clone https://github.com/pjreddie/darknet.git
```

Darknet là một khung mạng thần kinh mã nguồn mở được viết bằng C và CUDA. Nó nhanh chóng, dễ cài đặt và hỗ trợ tính toán CPU và GPU.

- Cấu hình của mạng được lưu trong file .cfg . Tiến hành chỉnh sửa file .cfg cho phù hợp với dữ liệu. Ví dụ : yolov3.cfg. Các tham số cần thay đổi :

+classes : Tại các lớp [yolo] . thay đổi *classes = 1 (* tương ứng với một đối tượng cần phát hiện*)*

+filters: Tại các lớp [convolutional] ngay phía trên lớp [yolo]. Thay đổi *filters = 18.* Tính theo công thức filters = (classes+5) *3.

+batch size = 64 : Số lượng ảnh sử dụng tại mỗi vòng lặp.

+subdivisons size = 12: Số phần được chia nhỏ (mini batch).

Ví dụ.

batch-size = 64 : Có 64 hình ảnh được sử dụng trong mỗi vòng lặp

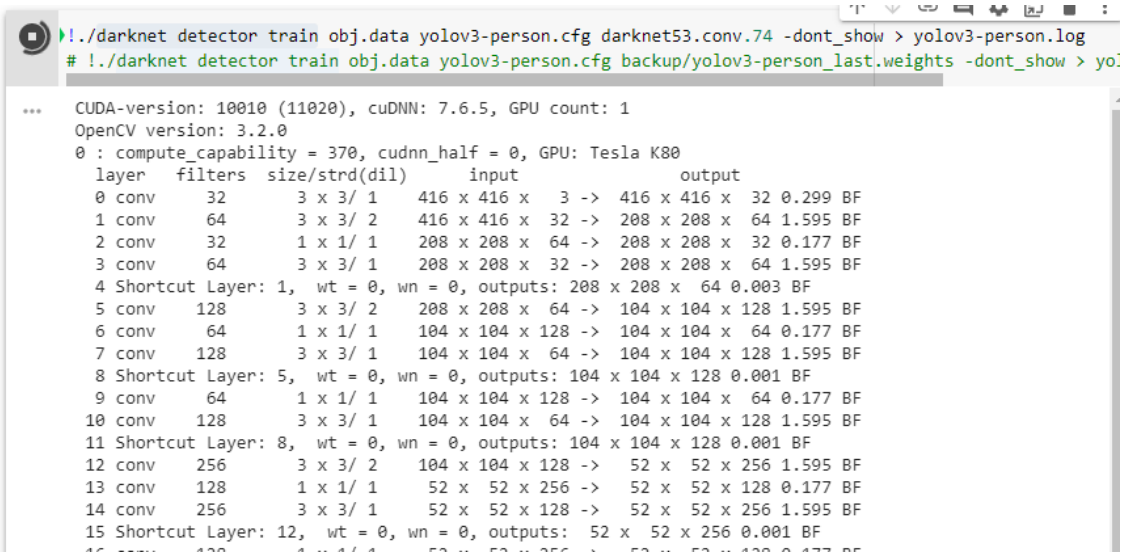
subdivisions = 16: 64 hình ảnh được chia thành 16 phần. Mỗi phần gồm 4 ảnh. Tương ứng mỗi vòng lặp sẽ có 16 lần xử lý. Mỗi lần GPU sẽ xử lý 4 ảnh.

+max-batches = 6000: Tính theo công thức *classes * 2000* nhưng không nhỏ hơn số lượng dữ liệu huấn luyện và không nhỏ hơn 6000.

- Tải file trọng số *darknet53.conv.74.weights*. Nó được dùng như một trình trích rút đặc trưng cho Yolo

- Tạo một folder ‘backup’ để lưu lại file trọng số trong quá trình huấn luyện. File trọng số sẽ tự động được lưu lại sau mỗi 100 vòng lặp. Có thể thay đổi tham số này trong file darknet/src/detector.c dòng 397.

-Tiến hành huấn luyện:

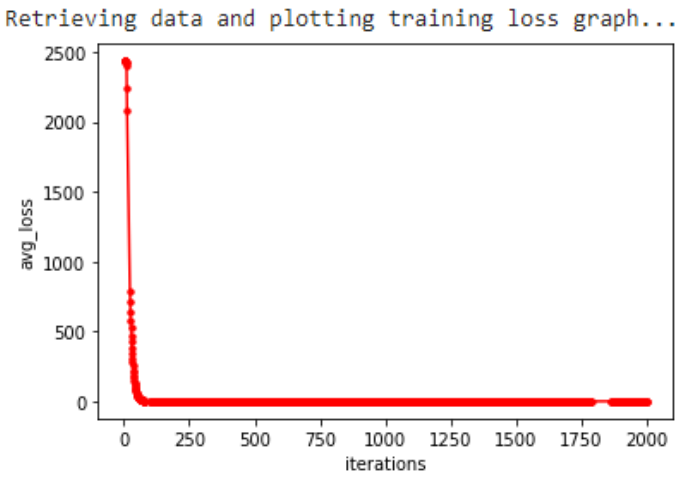


Hình 3. 7.. Bắt đầu quá trình huấn luyện

Quá trình huấn luyện có thể mất khoảng 5 giờ đồng hồ tùy thuộc vào các tham số và cấu hình phần cứng . Mọi thông số huấn luyện được xuất ra file .log để tiện theo dõi.

3.3. Kết quả

- Hiển thị giá trị avg_loss trong quá trình huấn luyện.



Hình 3. 8.Hiển thị giá trị avg_loss trong quá trình huấn luyện

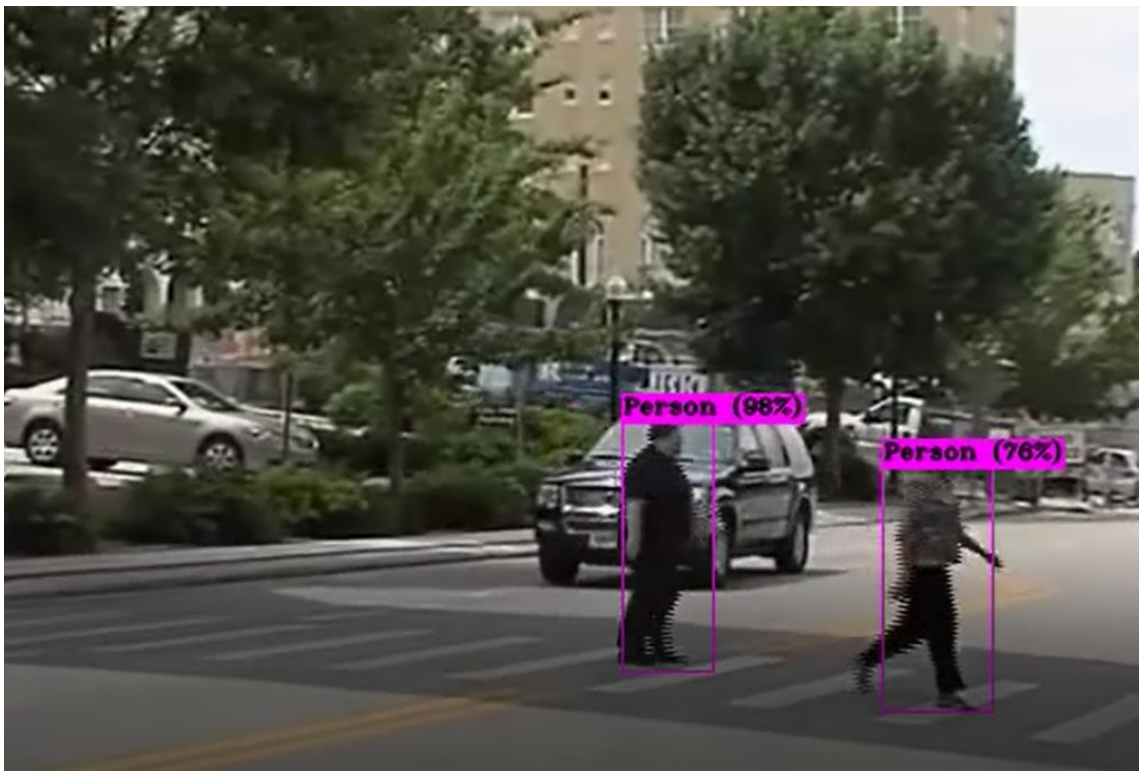
Giá trị avg_loss giảm dần trong khoảng 200 vòng lặp đầu tiên. Có thể dừng huấn luyện ngay khi thấy giá trị này không giảm thêm được.

Kết quả kiểm tra phát hiện người trên video:

Hình 3. 9.Kết quả kiểm tra trên video test1.mp4

Hình 3. 10.Kết quả kiểm tra trên video test1.mp4

Hình 3. 11.Kết quả kiểm tra trên video test2.mp4



Hình 3. 12.Kết quả kiểm tra trên video test2.mp4

3.4. Thảo luận

- Mô hình đã phát hiện được người đi bộ trong video với tỉ lệ dự đoán lên đến 98%. Tuy nhiên tại sao lại có khác biệt giữa các lần dự đoán?

- Trên thực tế, con người có rất nhiều tư thế, hình dáng khác nhau. Thêm vào đó, tùy vào bối cảnh, việc thay đổi độ sáng của môi trường cũng ảnh hưởng đến tỉ lệ dự đoán.

- Một lý do quan trọng và là vấn đề của tất cả các hệ thống phát hiện đối tượng, đó là dữ liệu. Khi dữ liệu đủ lớn và quá trình gán nhãn càng chính xác thì kết quả đạt tỉ lệ càng cao.

- Cũng phải nói đến yếu tố con người. Chính xác hơn đó là kinh nghiệm của người xây dựng hệ thống. Với khả năng điều chỉnh các tham số của mô hình một cách phù hợp với dữ liệu và đối tượng cần phát hiện thì vẫn sẽ đạt được kết quả tốt.

KẾT LUẬN VÀ HƯỚNG NGHIÊN CỨU TIẾP THEO

- Bài toán phát hiện người đi bộ chỉ là bước đầu tiên của một hệ thống ứng dụng deep learning. Bước này chủ yếu trả lời các câu hỏi “Có đối tượng A trong ảnh hay không?” , “Tọa độ của đối tượng là gì?”. Các câu trả lời sẽ được sử dụng để tạo ra các ứng dụng có ích trong cuộc sống.

-Hướng đi tiếp theo sau bài toán phát hiện người đi bộ có thể là:

+Hệ thống cảnh báo có người qua đường.

+Hệ thống thống kê số người đi bộ qua cửa hàng.

TÀI LIỆU THAM KHẢO

1. Mayank Mishra (27/8/2020). Convolutional Neural Networks, Explained - https://towardsdatascience.com/
2. Ankushsharma (25/6/2020) . YOLO V1 Architecture - https://medium.com/@ankushsharma2805
3. Mauricio Menegaz (17/3/2018). Understanding YOLO - https://hackernoon.com/understanding-yolo-f5a74bbc7967
4. Topdev.vn Thuật toán CNN – Convolutional Neural Network https://topdev.vn/blog/thuat-toan-cnn-convolutional-neural-network/