ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

~~~~~~\*~~~~~~

Logo

Description automatically generated

**BÁO CÁO ĐỒ ÁN**

**ĐỀ TÀI: PHÁT HIỆN VÀ PHÂN LOẠI MỘT SỐ BIỂN BÁO GIAO THÔNG ĐƯỜNG BỘ VIỆT NAM**

**Lớp:** CS231.M21.KHTN

**Giảng viên:** TS. Mai Tiến Dũng

**Thành viên nhóm:**

|  |  |  |
| --- | --- | --- |
| **STT** | **MSSV** | **Họ và tên** |
| 1 | 20520051 | Nguyễn Quốc Huy Hoàng |
| 2 | 20520073 | Lê Nguyễn Khánh Nam |
| 3 | 20520094 | Lương Triều Vỹ |

TP. HỒ CHÍ MINH 06/2022

**LỜI CẢM ƠN**

Sau quá trình học tập và rèn luyện tại Trường Đại học Công Nghệ Thông Tin, chúng em đã được trang bị các kiến thức cơ bản, các kỹ năng thực tế để có thể hoàn thành đồ án môn học của mình.

Chúng em xin gửi lời cảm ơn chân thành đến thầy Mai Tiến Dũng – Giảng viên  
phụ trách lớp CS231.M21.KHTN – Môn Nhập môn thị giác máy tính đã tận tâm hướng dẫn, truyền đạt những kiến thức cũng như kinh nghiệm cho chúng em trong suốt thời gian học tập.

Trong quá trình làm đồ án môn học, mặc dù nhóm em đã cố gắng nhưng chắc chắn sẽ không tránh được những sai sót không đáng có. Qua một buổi trình bày trên lớp và một buổi bổ xung thì nhóm em nhận được một số sự góp ý của thầy, thì nhóm em sẽ có những sự sửa đổi cho hợp lý hơn trong bài báo cáo này, nhờ những lời góp đó mà nhóm em có thêm nhiều kinh nghiệm và kiến thức hơn. Qua môn học này ngoài kiến thức môn học thì em cũng đã củng cố được thêm một số điều khác như kĩ năng trình bày, kĩ năng tìm kiếm tài liệu. Em xin chân thành cảm ơn!

TP. Hồ Chí Minh, tháng 06 năm 2022.

**Mục lục**

[**I.** **Giới thiệu về bài toán** 4](#_Toc107324528)

[**1.** **Vấn đề mà bài toán giải quyết** 4](#_Toc107324529)

[**2.** **Tính ứng dụng của bài toán** 4](#_Toc107324530)

[**II.** **Giải quyết bài toán** 5](#_Toc107324531)

[**1.** **Hướng giải quyết** 5](#_Toc107324532)

[**2.** **Thông tin về tập dữ liệu (Dataset)** 5](#_Toc107324533)

[**3.** **Detection** 7](#_Toc107324534)

[**4.** **Recognition** 9](#_Toc107324535)

[**a.** **VGG-19** 9](#_Toc107324536)

[**b.** **MobileNet\_v1** 11](#_Toc107324537)

[**III.** **Thực nghiệm (demo)** 14](#_Toc107324538)

[**IV.** **Đánh giá tổng kết** 16](#_Toc107324539)

[**V.** **Tài liệu tham khảo** 17](#_Toc107324540)

1. **Giới thiệu về bài toán**
   1. **Vấn đề mà bài toán giải quyết**

- Đầu vào: Một bức ảnh chụp từ cabin của xe khách lúc lưu thông

- Đầu ra: Thông tin các biển báo giao thông xuất hiện trong bức ảnh đó

A road with a sign on it

Description automatically generated with low confidence

*Ảnh minh họa.*

Mô tả: Khi ta đưa vào một bức ảnh được chụp từ camera từ cabin của xe khách, thì bài toán sẽ giải quyết và cho ra kết quả là một bức ảnh có thêm thông tin về vị trí và loại biển báo của các biển báo giao thông xuất hiện trong bức ảnh đó.

* 1. **Tính ứng dụng của bài toán**

Hiện nay việc đi lại bằng xe chở khách rất là phổ biến, cho nên là có rất nhiều xe khách lưu thông trên đường. Xe chở khách thường phải di chuyển một quãng đường khá là dài, và cũng hay hoạt động về ban đêm, chính vì thế mà tài xế lái xe khách thường không được tỉnh táo như là lái xe cá nhân hoặc là xe chở người khách đi quãng đường ngắn khác.

Vậy nên việc tạo ra một ứng dụng hỗ trợ cho tài xế xe khách lúc lái xe là việc cần thiết, giúp cho tài xế biết được một số biển báo quan trọng như là biển báo cấm, biển báo nguy hiểm, biển báo chỉ dẫn, để giúp cho tài xế lái xe an toàn hơn và tránh vi phạm luật giao thông.

Bài toán của chúng em làm sẽ phát hiện và nêu thông tin của một số biển báo xuất hiện trong hình ảnh, và từ bài toán này sau này có thể phát triển ra bài toán lớn hơn là phát hiện và chỉ ra các thông tin về các biển báo xuất hiện trong video, từ đó có thể phát triển ra ứng dụng chạy realtime.

1. **Giải quyết bài toán**
   1. **Hướng giải quyết**

A picture containing text, sky, scene, way

Description automatically generated

Để giải quyết bài toán này chúng em nhận thấy có 2 hướng giải quyết:

* Single-stage detector: Đầu vào sẽ là một bức ảnh, sau đó sẽ detect và đưa ra loại biển báo giao thông luôn. Ở phần này, chúng em sẽ sử dụng RetinaNet.
* Two-stage detector: Đầu vào sẽ là một bức ảnh, sau đó sẽ sử dụng một detector để detect ra biển báo giao thông nằm ở đâu, sau đó chúng em sẽ crop phần ảnh đó ra, và sử dùng ảnh đó cho bài toán Recognition để nhận diện đó là loại biển báo giao thông nào. Bài toán detection thì chúng em dùng luôn RetinaNet ở trên, còn phần Recognition thì chúng em sẽ dùng VGG-19 và MobileNet\_v1.
  1. **Thông tin về tập dữ liệu (Dataset)**

Đầu tiên ta cần biết sơ lược về biển báo giao thông ở Việt Nam trước. Ở buổi thuyết trình trên lớp thì nhóm em có đưa ra một số loại biển báo, nhưng chưa chứng minh được độ tin cậy, vậy nên từ góp ý đó nhóm chúng em đã tìm hiểu và đưa ra thông tin về biển báo giao thông ở Việt Nam như sau:

Khoản 4 Điều 10 Luật Giao thông đường bộ quy định biển báo giao thông đường bộ gồm có 5 nhóm:

a) Biển báo cấm để biểu thị các điều cấm;

b) Biển báo nguy hiểm để cảnh báo các tình huống nguy hiểm có thể xảy ra;

c) Biển hiệu lệnh để báo các hiệu lệnh phải thi hành;

d) Biển chỉ dẫn để chỉ dẫn hướng đi hoặc các điều cần biết;

đ) Biển phụ đề thuyết minh bổ sung các loại biển báo cấm, biển báo nguy hiểm, biển hiệu lệnh và biển chỉ dẫn.

Chúng em sẽ phân thành 7 tập có nhãn lần lượt là:

* Cấm đi ngược chiều
* Cấm rẽ
* Giới hạn tốc độ tối đa
* Biển cấm
* Biển báo nguy hiểm
* Biển hiệu lệnh và chỉ dẫn

Chúng em phân thành 7 tập như thế vì dựa vào tập dữ liệu mà mình thu thập được, và nhận thấy chia như thế cũng khá hợp lý, vì một số biển như cấm đi ngược chiều, cấm rẽ, giới hạn tốc độ tối đa gặp khá là nhiều, nên tách ra làm một tập riêng.

Dữ liệu mà chúng em dùng để detect là lấy từ zalo-traffic-sign (Nguồn: [Tại đây](https://www.kaggle.com/datasets/loitranv/zalo-traffic-sign))

Gồm có 4500 ảnh với kích thước 1622x626

Ở phần Recognition: Có 11000 ảnh biển báo crop từ 11000 bound box từ tập train thuộc 7 loại biển báo khác nhau

Trong đó có:

* 23 biển báo bị mất 1 phần.
* 72 biển báo bị che mất bởi cây.
* 15 biển báo bị che mất bởi các yếu tố khác.

Sau khi chạy thử thì chúng em có nhận xét là: Detection cho kết quả khá cao trên tập test. Tuy nhiên với các dữ liệu bên ngoài như biển báo bị cây che khuất thì model vẫn chưa dự đoán chuẩn.

* Giải pháp đề xuất: Thu thập thêm data về trường hợp khó sau đó tiếp tục huấn luyện mô hình rồi đánh giá kết quả

Thực hiện:

* Sử dụng tool sau đó crawl 1000 tấm ảnh biển báo giao thông bị che khuất.
* Lọc và chọn được 300 tấm ảnh phù hợp.
* Resize tất cả tấm ảnh về kích thước 1622x626
* Sử dụng website để gán nhãn dữ liệu
* Thêm vào dữ liệu để tiếp tục huấn luyện
  1. **Detection**
* Input: Ảnh chứa biển báo giao thông
* Output: Vùng chứa biển báo giao thông đó
* Phương pháp: nhóm đề xuất sử dụng mạng RetinaNet cho phần detection
* Lý do: RetinaNet có lợi thế cho việc detect các biển báo giao thông nhỏ, khó nhìn đối với mắt thường

Link paper: <https://arxiv.org/pdf/1708.02002.pdf>

Retinanet có cấu trúc như sau:

Diagram

Description automatically generated

Từ ảnh đầu vào sẽ đưa qua một mạng CNN đóng vai trò trích xuất đặc trưng. Tuy nhiên ở đây kết hợp thêm một mạng FPN (Feature Pyramid Network) để tạo ra một loạt các output dạng kim tự tháp như trên ảnh Mục đích của việc này là để nhận diện đối tượng ở các tỷ lệ scale khác nhau.

FPN sẽ gồm một nhánh đi lên (a) và nhánh đi xuống (b). Nhánh đi lên đóng vai trò trích xuất đặc trưng, sẽ giảm về kích thước nhưng tăng về ý nghĩa (giảm 2 lần sau mỗi lần downsampling) **. Một số mạng sau vài layer Conv thì dùng feature map (FM) cuối cùng để detection nên dễ bỏ qua các object nhỏ.**

Do đó, ở đây FPN bổ sung thêm nhánh bên phải đi xuống để up sampling từ (FM) nhỏ nhất. Mỗi lần upsampling kích thước sẽ tăng lên gấp đôi. Và lúc này tuy là layer được tạo ra vẫn giữ ý nghĩa nhưng vị trí của object đã bị mất qua các bước down sampling và up sampling. Tác giả khắc phục bằng cách nối với một layer ở nhánh đi lên (layer cùng level) bằng phép cộng element-wise additional với hi vọng giữ lại được thông tin đó.

Sau khi cộng vào thì sẽ sinh ra một loạt các FM mới và Retinanet dùng các FM đó để đưa vào một 2 mạng phụ (subnet ) là class subnet và box subnet để predict các thông tin chúng ta cần là:

* Class subnet thì predict probality của các class trên từng điểm ảnh trên ảnh. Output của mạng này như trên hình các bạn xem là WxHxKA. Trong đó W, H là width và height của ảnh. KA là độ sâu = K \* A, K là class cũ và A là số anchorbox .Ý nghĩa là với từng điểm ảnh, với anchorbox này, thì mỗi class có probality là bao nhiêu đó.
* Box subnet thì predict ra cái box của vật thể tại từng điểm ảnh và output có độ sâu là 4xA. Vì mỗi anchorbox ta sẽ dự đoán 4 giá trị là : (x, y, w, h) (tâm và rộng dài) của box đó.

Một số kết quả sau khi huấn luyện:

A picture containing text, tree, outdoor, track and field

Description automatically generated

A picture containing text, road, outdoor, tree

Description automatically generated

A road with trees on the side

Description automatically generated with medium confidence

* 1. **Recognition**
* Input: Ảnh biển báo giao thông
* Output: Tên/loại của biển báo giao thông đó
* Phương pháp: nhóm đề xuất sử dụng mạng VGG-19 và Mobilenet\_v1

Lý do:

* VGG-19 : là 1 trong những kiến trúc nổi tiếng trong object recognition, mạng dễ triển khai, huấn luyện và có sẵn trong nhiều thư viện.
* Mobilenet\_v1 :đây là mô hình hữu ích cho việc chạy trên các ứng dụng di động và thiết bị nhúng.
  + 1. **VGG-19**
* Sơ lược về VGG-19:

VGG là viết tắt của Visual Geometry Group; nó là một kiến trúc Mạng nơ-ron tích chập (CNN) với nhiều lớp. Độ sâu ở đây đề cập đến số lượng các lớp với VGG-16 hoặc VGG-19 bao gồm 16 và 19 lớp phức hợp.

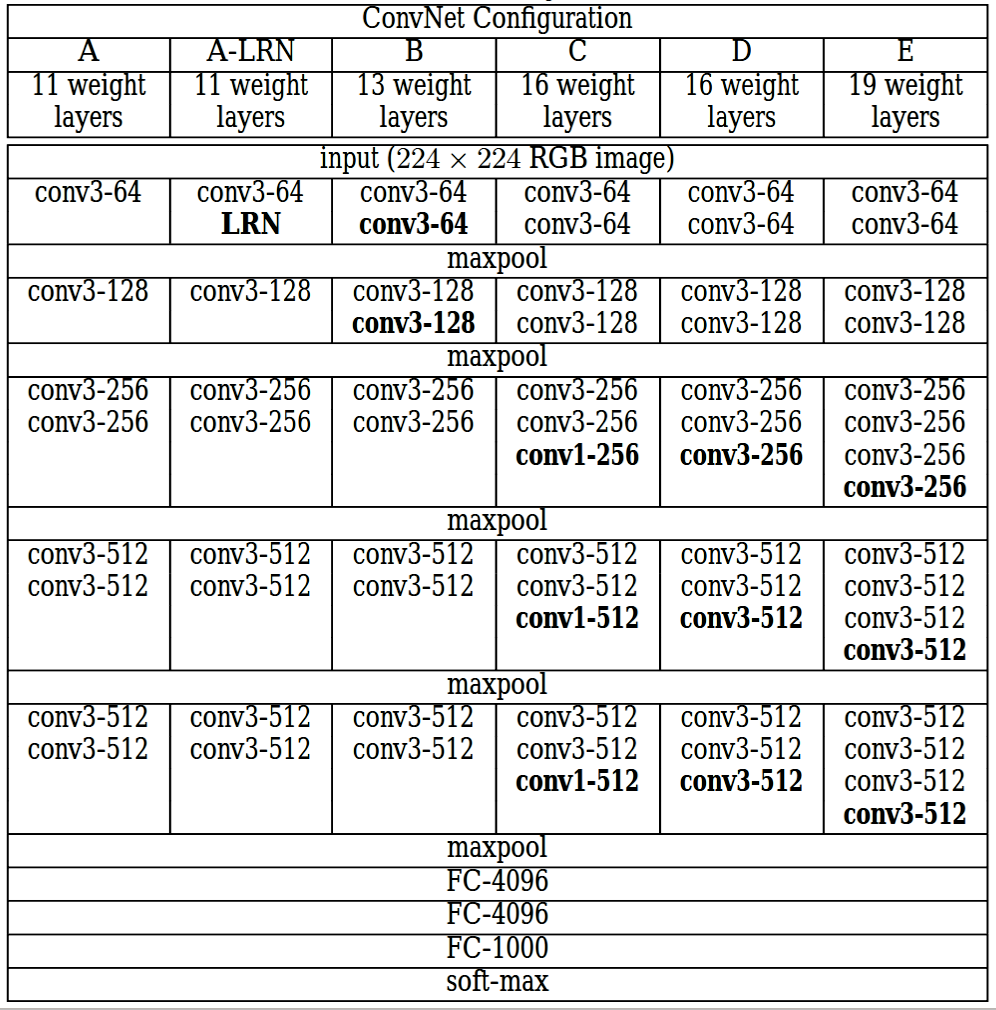
Kiến trúc VGG là cơ sở của các mô hình nhận dạng đối tượng mang tính đột phá. Được phát triển như một mạng nơ-ron sâu, VGGNet cũng vượt qua các đường cơ sở về nhiều tác vụ và bộ dữ liệu ngoài ImageNet. Hơn nữa, bây giờ nó vẫn là một trong những kiến trúc nhận dạng hình ảnh phổ biến nhất.

VGG-19 là một kiến trúc mạng nơ-ron tích chập có độ sâu 19 lớp. Mục đích chính mà mạng VGG được thiết kế là để giành chiến thắng trong cuộc thi mạng hình ảnh ILSVRC.

Diagram

Description automatically generated

* Kiến trúc mạng VGG-19:

Kiến trúc VGG19 được thể hiện ở cột E ảnh sau:

* + Hình ảnh RGB có kích thước cố định (224 \* 224) được cung cấp làm đầu vào cho mạng này, có nghĩa là ma trận có hình dạng (224,224,3).
  + Quá trình tiền xử lý duy nhất đã được thực hiện là chúng đã trừ giá trị RGB trung bình từ mỗi pixel, được tính trên toàn bộ tập huấn luyện.
  + Đã sử dụng các kernel có kích thước (3 \* 3) với kích thước bước sóng là 1 pixel, điều này cho phép chúng bao quát toàn bộ khái niệm của hình ảnh
  + Spatial Padding được sử dụng để duy trì độ phân giải không gian của hình ảnh.
  + Max Pooling được thực hiện trên cửa sổ 2 \* 2 pixel với stride 2.
  + Theo sau đó là Rectified linear unit(ReLu) để có tính phi tuyến tính để làm cho mô hình phân loại tốt hơn và cải thiện thời gian tính toán vì các mô hình trước sử dụng hàm tanh hoặc hàm sigmoid, điều này tỏ ra tốt hơn nhiều so với các mô hình này.
  + Thực hiện 3 lớp fully connected mà từ đó hai lớp đầu tiên có kích thước 4096 và sau đó là lớp có 1000 kênh để phân loại ILSVRC 1000 chiều và lớp cuối cùng là hàm softmax.

Ở đồ án này thì tụi em cài đặt mạng VGG-19 như hình bên dưới:

Text

Description automatically generated

Bọn em chọn 32\*32 vì đây là kích thước nhỏ nhất của VGG-19 để thuận tiện cho việc tính toán nhanh, và dữ liệu của chúng em thì cũng thấy nhiều bức ảnh nhỏ.

Text

Description automatically generated

(Nguồn: <https://keras.io/api/applications/vgg/#vgg19-function>)

A screenshot of a computer

Description automatically generated with medium confidence

Và tụi em cũng thấy với kích thước 32x32 thì sẽ cho ra kết quả tốt hơn.

* + 1. **MobileNet\_v1**

Nhu cầu về lightweight model: Sự phát triển về mặt học thuật của thị giác máy tính đã tạo ra rất nhiều các models khác nhau với độ chính xác được đo lường trên bộ dữ liệu ImageNet ngày càng được cải thiện. Tuy nhiên không phải toàn bộ trong số chúng đều có thể sử dụng được trên các thiết bị gặp hạn chế về tài nguyên tính toán. Để phát triển được những ứng dụng AI trên các thiết bị như mobile, IoT thì chúng ta cần hiểu về tài nguyên của những thiết bị này để lựa chọn model phù hợp cho chúng. Những mô hình ưa chuộng được sử dụng thường là những model có số lượng tính toán ít và độ chính xác cao. MobileNet là một trong những lớp mô hình như vậy và chúng ta đi sâu về kiến trúc đặc biệt ẩn đằng sau họ các lớp mô hình MobileNet.

Các dạng tích chập:

* Tích chập 2 chiều thông thường:
* Như chúng ta đã biết tích chập 2 chiều thông thường sẽ được tính toán trên toàn bộ chiều sâu (channel). Do đó số lượng tham số của mô hình sẽ gia tăng đáng kể phụ thuộc vào độ sâu của layer trước đóChart

  Description automatically generatedĐể thống nhất, ký hiệu shape của các tensor3D theo thứ tự các chiều (height, width, channel).
* Chẳng hạn ở trên chúng ta có một đầu vào kích thước h×w×c, tích chập thông thường sẽ cần k×k×c tham số để thực hiện tích chập trên toàn bộ độ sâu của layers. Mỗi một bộ lọc sẽ tạo ra một ma trận output kích thước h′×w′×1. Áp dụng c′ bộ lọc khác nhau ta sẽ tạo ra đầu ra có kích thước h′×w′×c′ (các ma trận output khi áp dụng tích chập trên từng bộ lọc sẽ được concatenate theo chiều sâu). Khi đó số lượng tham số cần sử dụng cho một tích chập thông thường sẽ là: c′×k×k×c.
* Số lượng tham số sẽ không đáng kể nếu độ sâu của input channel c là nhỏ, thường là ở các layers ở vị trí đầu tiên. Tuy nhiên khi độ sâu tăng tiến dần về những layers cuối cùng của mạng CNN thì số lượng tham số của mô hình sẽ là một con số không hề nhỏ. Sự gia tăng tham số này tạo ra những mô hình cồng kềnh làm chiếm dụng bộ nhớ và ảnh hưởng tới tốc độ tính toán. Alexnet và VGGNet là những mô hình điển hình có số lượng tham số rất lớn do chỉ áp dụng những tích chập 2 chiều thông thường.
* Trên [ImageNet Leader Board](https://paperswithcode.com/sota/image-classification-on-imagenet) ta sẽ thấy rằng những mô hình thuộc họ MobileNet chỉ có vài triệu tham số nhưng độ chính xác tốt hơn AlexNet vài chục triệu tham số. Điểm mấu chốt tạo nên sự khác biệt này đó là MobileNet lần đầu tiên áp dụng kiến trúc tích chập tách biệt chiều sâu.
* Tích chập tách biệt chiều sâu (Depthwise Separable Convolution):

Chúng ta nhận định rằng độ sâu là một trong những nguyên nhân chính dẫn tới sự gia tăng số lượng tham số của mô hình. Tích chập tách biệt chiều sâu sẽ tìm cách loại bỏ sự phụ thuộc vào độ sâu khi tích chập mà vẫn tạo ra được một output shape có kích thước tương đương so với tích chập thông thường. Cụ thể quá trình sẽ được chia thành hai bước tuần tự:

* **Tích chập chiều sâu (Depthwise Convolution)**: Chúng ta sẽ chia khối input tensor3D thành những lát cắt ma trận theo độ sâu. Thực hiện tích chập trên từng lát cắt như hình minh họa bên dưới:Diagram

  Description automatically generated

Kết quả sau tích chập được concatenate lại theo độ sâu. Như vậy output thu được là một khối tensor3D có kích thước h′×w′×c.

* **Tích chập điểm (Pointwise Convolution)**: Có tác dụng thay đổi độ sâu của output bước trên từ c sang c′. Chúng ta sẽ áp dụng c′ bộ lọc kích thước 1×1×c. Như vậy kích thước width và height không thay đổi mà chỉ độ sâu thay đổi

A picture containing diagram

Description automatically generated

Kết quả sau cùng chúng ta thu được là một output có kích thước h′×w′×c′. Số lượng tham số cần áp dụng ở trường hợp này là c′×c.

So sánh tích chập tách biệt chiều sâu và tích chập thông thường:

* **Số lượng tham số:**

Như chúng ta đã biết, để tạo thành một shape có kích thước h′×w′×c′ thì số lượng tham số sử dụng ở tích chập thông thường là: c′×k×k×c Trong khi đó số lượng tham số sử dụng ở tích chập chiều tách biệt chiều sâu: k×k×c+c′×c Thông thường c′ sẽ lớn hơn c vì càng ở các layer sau thì độ sâu càng lớn. Do đó tỷ lệ Tỷ lệ này sẽ gần bằng k×k nếu c’ >> k × k. Đây là một mức giảm khá đáng kể về kích thước mô hình. Đó là lý do vì sao mà MobileNet lại có kích thước nhỏ gấp vài chục lần so với Alexnet.

* **Số lượng phép toán cần thực hiện:**

Để cùng tạo ra một output shape có kích thước h′×w′×c′ thì tích chập thông thường cần thực hiện: (h′×w′×c′)×(k×k×c) Trong đó h′×w′×c′ là số lượng pixels cần tính và k×k×c là số phép nhân để tạo ra một pixel.

Tích chập tách biệt chiều sâu chỉ phải thực hiện lần lượt trên:

* Tích chập chiều sâu: (h′×w′×c)×(k×k) phép nhân.
* Tích chập điểm: (h′×w′×c)×(h′×w′) phép nhân. Tỷ lệ các phép tính giữa tích chập thông thường và tích chập chiều sâu:

Đây là một tỷ lệ khá lớn cho thấy tích chập chiều sâu tách biệt có chi phí tính toán thấp hơn rất nhiều so với tích chập thông thường. Do đó nó phù hợp để áp dụng trên các thiết bị cấu hình yếu.

Ở đồ án này thì tụi em cài đặt mạng MobileNet\_v1 như hình bên dưới

Text

Description automatically generated

Lý do chọn kích thước 32x32 thì cũng tương tự với VGG-19

1. **Thực nghiệm (demo)**

Toàn bộ dữ liệu, chương trình (Dữ liệu để detection, dữ liệu để recognition, crawler, chương trình demo...) mà nhóm chúng em làm việc được chúng em upload lên đây:<https://drive.google.com/drive/folders/1RarkridVopVlFFtMsQdeL3niYkQjjBPT?usp=sharing>

Code demo: <https://drive.google.com/file/d/1QtxrelszbL8mjaw61N-wFvb4iT2cmBrB/view?usp=sharing>

Một số hình ảnh demo:

Graphical user interface, application

Description automatically generated

*Ảnh 1*



*Ảnh 2*

Graphical user interface, application

Description automatically generated

*Ảnh 3*

Nhận xét: Ở ảnh 1 và 2 thì hoạt động tốt, nhận diện đúng và đủ các biển báo giao thông phức tạp. Nhưng ở ảnh 3 thì detect sai nhiều vật, các biển detect đúng thì recognition vẫn chính xác.

1. **Đánh giá tổng kết**

Qua quá trình làm và thực nghiệm thì chúng em có rút ra được một số ưu điểm và hạn chế như sau:

* Ưu điểm
* Detection và Recognition hoạt động khá tốt, một số trường hợp xấu như bị cây che một phần, bị biển báo phía trước che mất hoặc bị vật gì đó chắn ngang thì vẫn thực hiện được.
* Tốc độ Recognition khá nhanh, có thể khả thi về mặt tốc độ nếu triển khai thực tế
* Hạn chế
* Chưa được đầy đủ các loại biển báo
* Dữ liệu thu thập còn thiếu các trường hợp đặc biệt như là bị cây che, bị vật chắn nhang, bị mất một phần, bị nghiêng, … nên nhiều trường hợp detection không ra hoặc recognition
* Chưa phân loại chi tiết được các loại biển báo
* Detection hoạt động chậm (nhóm chúng em chưa tìm ra được nguyên nhân)
* Vì là mục đích làm trợ lý ảo nên cần cải thiện thêm độ chính xác khi recognition để tránh bị nhầm biển báo gây mất lòng tin

1. **Tài liệu tham khảo**

Chúng em có tham khảo một số các paper và trang web sau:

* Luật giao thông đường bộ: <https://luatvietnam.vn/giao-thong/luat-giao-thong-duong-bo-2008-39051-d1.html>
* Dataset: <https://www.kaggle.com/datasets/loitranv/zalo-traffic-sign>
* RetinaNet: <https://arxiv.org/pdf/1708.02002.pdf>
* VGG-19: <https://iq.opengenus.org/vgg19-architecture/>
* MobileNet: <https://phamdinhkhanh.github.io/2020/09/19/MobileNet.html>
* Và một số nguồn tài liệu khác:

<https://arxiv.org/pdf/1409.1556v6.pdf>

<https://arxiv.org/abs/1704.04861>

<https://viso.ai/deep-learning/vgg-very-deep-convolutional-networks/>

<https://keras.io/api/applications/vgg/#vgg19-function>