

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

----- ∞  ∞ -----



PROJECT III

Đề tài: Tìm hiểu và xây dựng mô hình
phát hiện và phân loại biến báo giao thông

Giảng viên: ThS. Bành Thị Quỳnh Mai

Sinh viên thực hiện: Hoàng Hà My 20207644

Hà Nội, tháng 3 năm 2024

MỤC LỤC

MỤC LỤC	2
MỞ ĐẦU	4
CHƯƠNG I. TỔNG QUAN ĐỀ TÀI	5
I.1. Thông tin tổng quan.....	5
I.1.1. Giới thiệu đề tài	5
I.1.2. Mục tiêu đề tài	5
I.1.3. Phương pháp thực hiện đề tài	6
I.1.4. Đối tượng nghiên cứu	6
CHƯƠNG II. LÝ THUYẾT TỔNG QUAN.....	7
II.1. Tổng quan về Mạng nơ-ron nhân tạo.....	7
II.1.1. Mạng nơ-ron nhân tạo	7
II.1.2. Một số hàm kích hoạt thông dụng	9
II.1.3. Backpropagation.....	12
II.2. Mạng nơ-ron tích chập (CNN)	12
II.2.1. Phép tính tích chập (Convolutional operation).....	12
II.2.2. Phép tích chập cho ảnh màu	16
II.2.3. Các loại lớp (layer) trong mạng nơ – ron tích chập.....	17
II.3. Mạng YOLO (You Only Look Once)	21
II.3.1. Tổng quan	21
II.3.2. Kiến trúc mạng YOLO	22
II.3.3. Cách hoạt động	22
II.3.4. Output của YOLO	23
II.3.5. Hàm tính IoU (Intersection Over Union)	24
II.3.6. Anchor box	24
II.3.7. Hàm mất mát	25
II.3.8. Cách thức dự đoán bounding box.....	26
II.3.9. Non – max suppression	27

CHƯƠNG III. XÂY DỰNG VÀ KIỂM THỬ	28
III.1. Thông tin tập dữ liệu huấn luyện	28
III.2. Xây dựng mô hình	29
III.2.1. Mô hình mạng CNN.....	29
III.2.2. Mô hình YOLO	31
III.3. Demo	32
KẾT LUẬN	35
TÀI LIỆU THAM KHẢO	36

MỞ ĐẦU

Xe ô tô tự lái (hay còn gọi là xe tự hành) là phương tiện có khả năng cảm nhận môi trường xung quanh và di chuyển an toàn mà không cần người lái hoặc cần sự can thiệp rất ít của con người. Loại phương này là sự kết hợp của rất nhiều công nghệ có khả năng cảm biến môi trường xung quanh như radar, lidar, GPS, AI (trí tuệ nhân tạo),... Theo hiệp hội kỹ sư ô tô Mỹ (SAE) hiện xác định 6 cấp độ khác nhau dành cho ô tô. Cấp độ 0 (no automation) sẽ cần con người điều khiển xe hoàn toàn theo phương pháp thủ công trong khi đó cấp độ 5 (full automation) sẽ cho phép xe vận hành hoàn toàn tự động.

Biển báo giao thông là những biển hiệu, chỉ dẫn trên đường thể hiện những thông tin về giao thông, mục đích cơ bản là giúp cho những người tham gia giao thông chấp hành luật giao thông một cách chính xác và an toàn nhất.

Như vậy, để nghiên cứu và xây dựng xe ô tô tự lái, rất cần thiết phải xây dựng mô hình giúp phương tiện phát hiện và nhận diện được biển báo giao thông, từ đó đưa ra những quyết định vận hành phù hợp.

CHƯƠNG I. TỔNG QUAN ĐỀ TÀI

I.1. Thông tin tổng quan

I.1.1. Giới thiệu đề tài

Ở đề tài này, chúng ta sẽ đi tìm hiểu và xây dựng mô hình xử lý bài toán phát hiện và phân loại biển báo giao thông với đầu vào (input) là video từ phương tiện tham gia giao thông (video quay sẵn hoặc video stream từ camera).

I.1.2. Mục tiêu đề tài

Mục tiêu cụ thể của đề tài được trình bày trong bảng dưới đây theo nguyên tắc SMART [6] :

Tính cụ thể (Specific)	Tìm hiểu về mô hình mạng nơ ron tích chập và mô hình YOLO. Xây dựng mô hình trên tập dữ liệu đã thu thập, đánh giá kết quả từ đó đưa ra kế hoạch cải tiến.
Tính đo lường (Measurable)	Thu thập hình ảnh và thông tin của biển báo giao thông để đưa vào quá trình huấn luyện, sau đó thực hiện tiền xử lý hình ảnh để kết quả dự đoán cao hơn 85%.
Tính khả thi (Achievable)	Ứng dụng được xây dựng và có thể triển khai như một phần của hệ thống tự lái.
Tính thực tế (Realistic)	Phạm vi nghiên cứu của đề tài phù hợp với trình độ của sinh viên thực hiện cũng như kết quả mà đề tài mang lại phù hợp với tình hình thực tế hiện tại.
Tính thời hạn (Timely)	Đề tài được hoàn thành đúng tiến độ theo thời gian của kì học mà người học thực hiện đề tài.

I.1.3. Phương pháp thực hiện đề tài

Đề tài này sử dụng phương pháp phân tích và tổng kết kinh nghiệm. Cụ thể là từ những công trình nghiên cứu liên quan đến đề tài và sự hỗ trợ từ các thư viện học máy để đề xuất một cách tiếp cận trong giải quyết vấn đề đặt ra.

Nội dung nghiên cứu về lý thuyết sẽ tập trung giới thiệu các mô hình sẽ sử dụng ở mức tổng quan. Sau đó, sinh viên thực hiện huấn luyện mô hình và kiểm thử hệ thống phát hiện và loại biển báo giao thông.

I.1.4. Đối tượng nghiên cứu

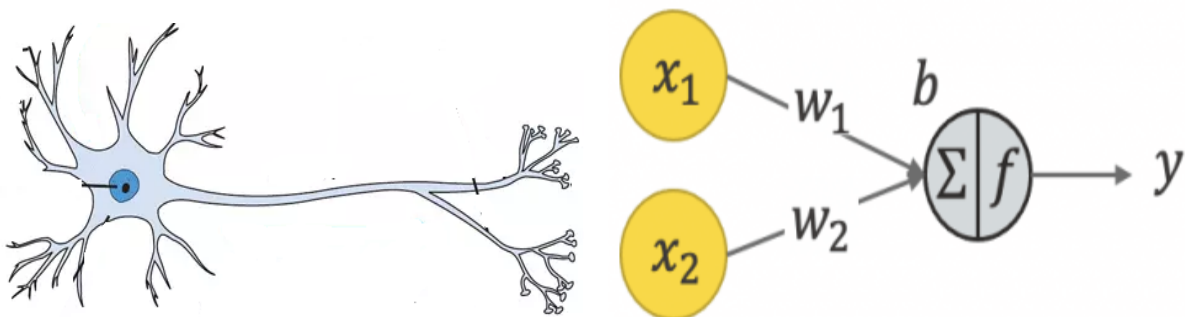
Đối tượng nghiên cứu của đề tài là phương pháp và ứng dụng phát hiện và phân loại biển báo giao thông. Do tính có thời hạn và năng lực của người thực hiện, đề tài sẽ được huấn luyện trên bộ dữ liệu biển báo giao thông German Traffic Sign (GTSRB và GTSDb) có sẵn. Mô hình xây dựng có thể sử dụng được với hệ thống biển báo giao thông tại Việt Nam khi có bộ dữ liệu về biển báo giao thông Việt Nam.

CHƯƠNG II. LÝ THUYẾT TỔNG QUAN

II.1. Tổng quan về Mạng nơ-ron nhân tạo

II.1.1. Mạng nơ-ron nhân tạo

Mạng nơ-ron nhân tạo (Artificial Neural Network) là cốt lõi của hệ thống học sâu. Mạng nơ-ron nhân tạo được xây dựng dựa trên các nguyên tắc về cấu tạo và hoạt động của các tế bào nơ-ron trong não bộ con người. Nơ-ron là đơn vị cơ bản cấu tạo hệ thống thần kinh và là phần quan trọng nhất của não. Não chúng ta gồm khoảng 10 triệu nơ-ron và mỗi nơ-ron liên kết với 10.000 nơ-ron khác.



H1. Nơ-ron nhân tạo mô phỏng nơ-ron sinh học

Mạng nơ-ron nhân tạo bao gồm nhiều lớp (layer) khác nhau, độ "sâu" của mạng được thể hiện ở số lượng lớp trong mạng đó. Trong mỗi lớp có các nút mạng (node, nơ-ron) và được liên kết với các lớp liền kề khác. Trong học máy, nơ-ron được định nghĩa là một hàm toán học nhận vào một hay nhiều giá trị đầu vào được nhân với các trọng số (weight). Trọng số là giá trị thể hiện của mỗi kết nối giữa hai nút mạng, trọng số này càng lớn thì kết nối này càng quan trọng đối với mạng.

Nơ-ron được định nghĩa với công thức sau :

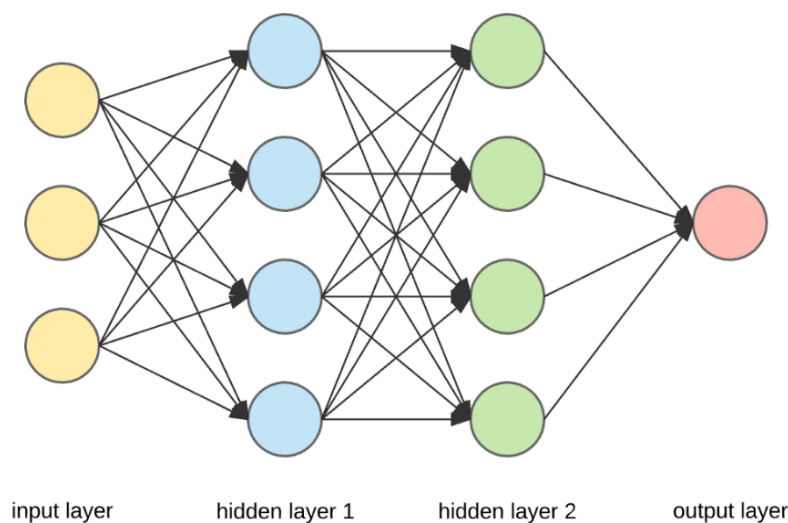
$$y = f(\sum x_i w_i + b)$$

Trong đó :

- x_i là giá trị đại diện cho dữ liệu đầu vào (input) của nơ-ron.

- w_i là trọng số (weight).
- b là một tham số bổ sung cho nơ-ron, dùng để điều chỉnh giá trị đầu ra của nơ-ron (bias).
- Hàm f được gọi là hàm kích hoạt (activation function).

Lớp đầu tiên của mạng nơ-ron nhân tạo (input layer) nhận thông tin đầu vào từ nguồn bên ngoài và chuyển nó đến lớp ẩn (hidden layer), một mạng có thể có một hay nhiều lớp ẩn. Mỗi nơ-ron ở lớp ẩn nhận thông tin từ những nơ-ron ở lớp ngay trước đó, tính toán tổng trọng số, sau đó chuyển tiếp cho các nơ-ron ở lớp tiếp theo. Mỗi nơ-ron sẽ có một hàm kích có nhiệm vụ chuẩn hóa đầu ra từ nơ-ron này. Cuối cùng, kết quả sẽ được trả về ở layer cuối cùng (output layer). Trong thực tế, nhiều bài toán yêu cầu đầu ra gồm nhiều hơn một giá trị. Do đó ta thường xây dựng những mạng nơ-ron với lớp đầu ra gồm nhiều nút mạng.

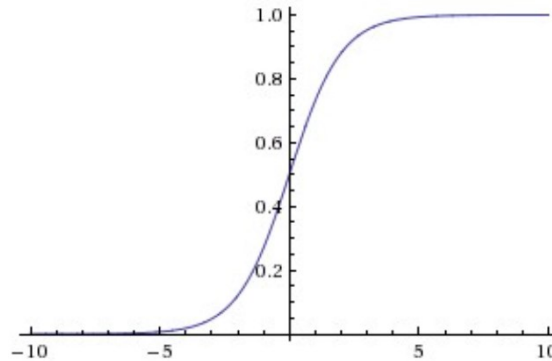


H2. Ví dụ cấu trúc của một ANN có 2 lớp ẩn với đầu ra có một giá trị

Quá trình trên được gọi là Feedforward (truyền thẳng). Dữ liệu từ tập mẫu huấn luyện được đưa vào mạng nơ-ron nhân tạo, sau đó kết quả đầu ra được so sánh với kết quả thực tế. Sự sai khác sẽ được sử dụng để điều chỉnh trọng số của các nút mạng. Việc thay đổi trọng số cho phù hợp sẽ được tính toán thông qua thuật toán Backpropagation (lan truyền ngược).

II.1.2. Một số hàm kích hoạt thông dụng

a. Hàm sigmoid



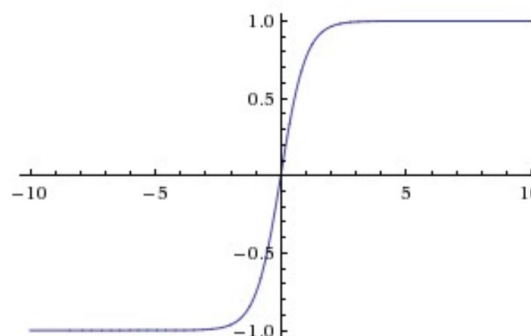
H3. Đồ thị hàm sigmoid (<https://cs231n.github.io/neural-networks-1/>)

Công thức toán học của hàm sigmoid:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Hàm số nhận đầu vào là một số thực sau đó "nén" lại thành đầu ra có giá trị trong khoảng $[0,1]$ và có thể được xem như xác suất trong một số bài toán. Tuy nhiên hiện nay hàm này rất ít được dùng vì xuất hiện một số nhược điểm. Ví dụ như hàm sigmoid làm bão hòa và triệt tiêu gradient: khi đầu vào có giá trị tuyệt đối rất lớn, gradient của hàm số này sẽ rất gần với 0. Điều này đồng nghĩa với việc các hệ số tương ứng với node đang xét sẽ gần như không được cập nhật.

b. Hàm tanh



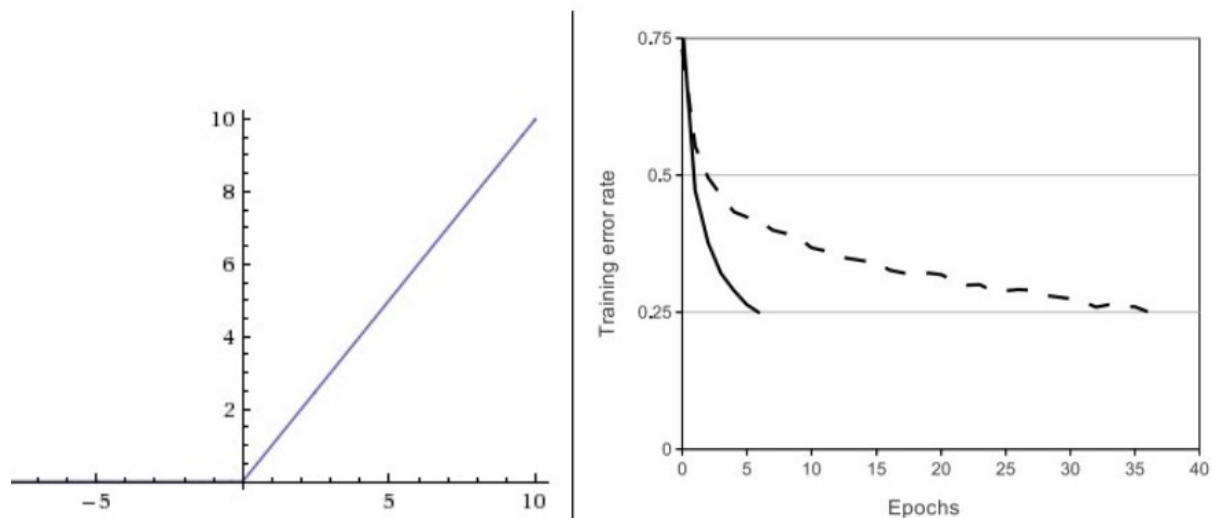
H4. Đồ thị hàm tanh (<https://cs231n.github.io/neural-networks-1/>)

Công thức toán học của hàm tanh:

$$\tanh(x) = 2\sigma(2x) - 1$$

Hàm tanh có nhược điểm tương tự như hàm sigmoid, xuất hiện vấn đề bão hòa và triệt tiêu đạo hàm khi đầu vào là rất lớn.

c. Hàm ReLU



H5. Đồ thị hàm ReLU và tốc độ hội tụ khi so sánh với hàm tanh.
(<https://cs231n.github.io/neural-networks-1/>)

Hàm ReLU trở nên phổ biến trong những năm gần đây, công thức toán học của hàm:

$$f(x) = \max(0, x)$$

Ưu điểm của hàm ReLU:

- Theo Krizhevsky et al., hàm ReLU được chứng minh có tốc độ tăng đáng kể cho việc huấn luyện các mạng học sâu. Điều này do dạng tuyến tính và không bão hòa của hàm.
- So với các hàm tanh/sigmoid liên quan đến các phép toán tốn kém (hàm mũ,...), hàm ReLU được triển khai bằng cách đơn giản.

Nhược điểm của hàm ReLU:

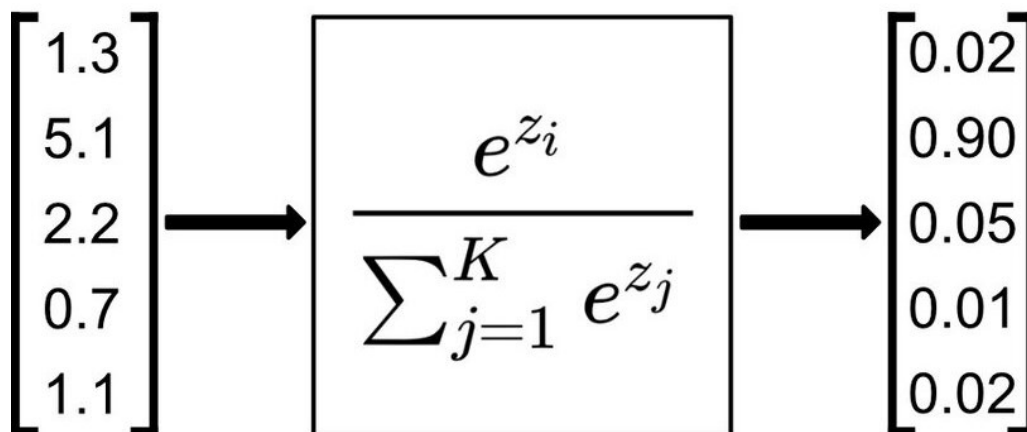
- Với các nút mạng có giá trị nhỏ hơn 0, sau khi áp dụng hàm ReLU sẽ thành 0, không còn ý nghĩa với lớp tiếp theo và các hệ số tương ứng từ nút đó cũng không được cập nhật với gradient descent. Hiện tượng này được gọi là "dying ReLU".

d. Hàm Softmax

Softmax hay là hàm trung bình mũ, tính toán xác suất xuất hiện của một sự kiện. Nói tổng quát, hàm trung bình mũ sẽ đưa ra khả năng hiện diện của một lớp trong tổng số toàn bộ các lớp của bài toán. Sau khi tính toán, tổng xác suất sẽ bằng 1, mỗi xác suất nằm trong khoảng (0;1].

Hàm softmax là một hàm kích hoạt thường được sử dụng trong các lớp đầu ra của mạng nơ-ron, đặc biệt là trong bài toán phân loại nhiều lớp. Hàm này chuyển đầu ra của một mạng nơ-ron thành một phân phối xác suất qua các lớp.

$$s(x_i) = \frac{e^{x_i}}{\sum_{j=1}^n e^{x_j}}$$



H6. Ví dụ áp dụng hàm softmax

(<https://botpenguin.com/glossary/softmax-function>)

II.1.3. Backpropagation

Backpropagation (lan truyền ngược) là một thuật toán quan trọng trong quá trình huấn luyện mạng nơ-ron, đặc biệt là trong các mô hình sâu. Quá trình này giúp mô hình "học" từ dữ liệu bằng cách điều chỉnh trọng số của các liên kết giữa các nơ-ron.

Lan truyền ngược bắt đầu sau khi đưa dữ liệu qua mạng nơ-ron để tạo ra dự đoán. Khi đó, giá trị dự đoán được so sánh với giá trị thực tế thông qua một hàm mất mát. Đạo hàm của hàm mất mát đối với từng trọng số trong mạng được tính toán sử dụng quy tắc chuỗi (chain rule). Đạo hàm của hàm mất mát được truyền ngược qua mạng, giúp đánh giá cách mỗi trọng số ảnh hưởng đến sự sai lệch giữa đầu ra dự đoán và giá trị thực tế.

Sau đó, thuật toán sử dụng các giá trị đạo hàm này để cập nhật trọng số của mạng, thường thông qua một phương pháp tối ưu hóa như gradient descent. Quá trình lặp này tiếp tục cho đến khi mạng đạt được sự hội tụ, tức là giảm thiểu mức độ sai lệch giữa dự đoán và thực tế đến mức tối ưu.

Backpropagation là một trong những cơ sở của học sâu và đã đóng góp đáng kể vào thành công của nhiều ứng dụng hiện đại như nhận diện hình ảnh, ngôn ngữ tự nhiên, và nhiều lĩnh vực khác.

II.2. Mạng nơ-ron tích chập (CNN)

II.2.1. Phép tính tích chập (Convolutional operation)

a. Công thức

Tích chập là phép tính toán học áp dụng trên hai hàm, với kết quả đầu ra là hàm thứ ba thường là sự sửa đổi của một trong các hàm ban đầu. Trong xử lý ảnh, phép tính chập có thể được biểu diễn như sau:

$$(f * h)(x, y) = \sum_{i=-k}^k \sum_{j=-k}^k f(x - i, y - j) \times h(i, j)$$

trong đó f là ma trận ảnh đầu vào có kích thước $n \times n$, h là ma trận tích chập (mask/ filter) và k là bậc của ma trận h . Ma trận tích chập thường là ma trận vuông với bậc lẻ. Ma trận đầu ra sẽ có kích thước $(n - k + 1) \times (n - k + 1)$.

Ví dụ 1: ma trận f có kích thước 3×3 , ma trận h có kích thước 2×2 , ma trận đầu ra $f * h$ có kích thước 2×2 .

$$\begin{array}{|c|c|c|} \hline a & b & c \\ \hline d & e & f \\ \hline g & h & i \\ \hline \end{array} * \begin{array}{|c|c|} \hline u & v \\ \hline w & x \\ \hline \end{array} = \begin{array}{|c|c|} \hline \begin{array}{|c|c|} \hline au + bv + dw + ex & bu + cv + ew + fx \\ \hline \end{array} \\ \hline \begin{array}{|c|c|} \hline du + ev + gw + hx & eu + fv + hw + ix \\ \hline \end{array} \\ \hline \end{array}$$

f **h** **f * h**

Ví dụ 2: ma trận f có kích thước 6×6 , ma trận h có kích thước 3×3 , ma trận đầu ra có kích thước 4×4 .

$$\begin{array}{|c|c|c|c|c|c|} \hline 1 & 7 & 0 & 3 & 2 & 5 \\ \hline 4 & 4 & 1 & 9 & 0 & 2 \\ \hline 5 & 1 & 7 & 1 & 2 & 4 \\ \hline 8 & 6 & 1 & 4 & 3 & 2 \\ \hline 5 & 3 & 1 & 8 & 9 & 3 \\ \hline 6 & 7 & 1 & 4 & 7 & 1 \\ \hline \end{array} * \begin{array}{|c|c|c|} \hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline 1 & -1 & 4 & 2 \\ \hline 8 & -3 & 4 & 6 \\ \hline 9 & -3 & -5 & 4 \\ \hline 16 & 0 & -16 & 10 \\ \hline \end{array}$$

3x3 filter

b. Padding

Với công thức như trên, khi ta thực hiện phép tính tích chập thì đều thu được ma trận đầu ra có kích thước nhỏ hơn ma trận đầu vào. Muốn ma trận đầu ra có kích thước bằng với kích thước ma trận đầu vào, ta sẽ cần thêm phần đệm (padding), bằng cách thêm các giá trị 0 ở viền ngoài của ma trận đầu vào. Padding = p nghĩa là thêm p vector 0 vào mỗi phía của ma trận.

0	0	0	0	0	0	0	0
0	1	7	0	3	2	5	0
0	4	4	1	9	0	2	0
0	5	1	7	1	2	4	0
0	8	6	1	4	3	2	0
0	5	3	1	8	9	3	0
0	6	7	1	4	7	1	0
0	0	0	0	0	0	0	0

Ví dụ: padding = 1

Phần đệm có tác dụng chống mất thông tin của pixel ở viền ảnh khi thực hiện phép tính tích chập. Ma trận đầu ra có kích thước $(n - k + 1 + 2p) \times (n - k + 1 + 2p)$.

c. Strided

Trong phép tính tích chập thông thường, bước nhảy (stride) thường được thiết lập là 1, có nghĩa là mỗi lần áp dụng phép tính toán, ma trận trượt qua chỉ một bước. Ngược lại, khi chúng ta sử dụng strided convolution với một stride được đặt là s , điều này sẽ điều chỉnh cách ma trận trượt di chuyển, thực hiện mỗi lần tính toán với một bước nhảy là s . Điều này có nghĩa là, thay vì trượt qua mỗi pixel một cách liên tiếp, với stride là s , chúng ta sẽ bỏ qua $s-1$ pixel giữa các lần tính toán. Điều này có thể giảm kích thước của ma trận kết quả và giảm độ phức tạp của mô hình.

Stride trong phép tính tích chập là một yếu tố quan trọng có thể ảnh hưởng đến độ chính xác của mô hình và cũng có thể được điều chỉnh tùy thuộc vào yêu cầu cụ thể của bài toán.

1	7	0	3	2	5
4	4	1	9	0	2
5	1	7	1	2	4
8	6	1	4	3	2
5	3	1	8	9	3
6	7	1	4	7	1

*

1	0	-1
1	0	-1
1	0	-1

=

2	4		

3x3 filter

Ví dụ với strided $s = 2$

Sử dụng stride giúp tăng tốc tính toán do giảm số lượng phép tính cần thực hiện, bên cạnh đó phương pháp này giúp giảm chiều của dữ liệu mà vẫn giữ được các đặc trưng quan trọng. Ma trận đầu ra có kích thước $(\frac{n-k}{s} + 1) \times (\frac{n-k}{s} + 1)$.

d. Kết hợp padding và stride

Thực tế khi sử dụng phép tích chập để xây dựng mạng nơ – ron tích chập, ta cần kết hợp cả hai phương pháp padding và stride để tận dụng được hết các lợi ích mà nó mang lại.

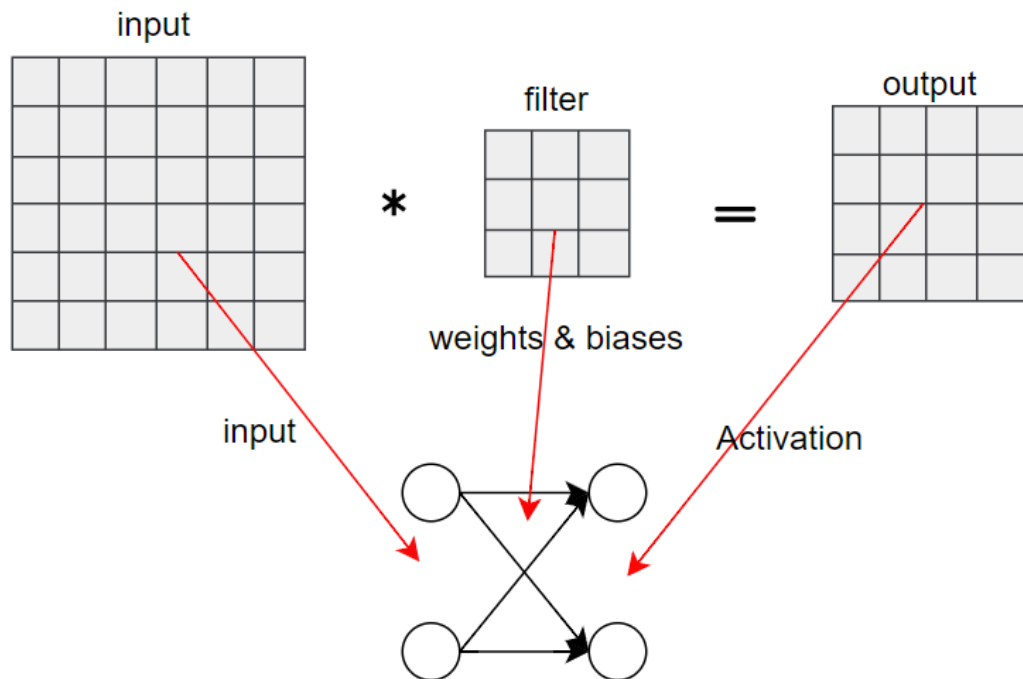
0	0	0	0	0	0	0	0
0	1	7	0	3	2	5	0
0	4	4	1	9	0	2	0
0	5	1	7	1	2	4	0
0	8	6	1	4	3	2	0
0	5	3	1	8	9	3	0
0	6	7	1	4	7	1	0
0	0	0	0	0	0	0	0

Ví dụ với $p = 1, s = 2$

Ma trận đầu ra sẽ có kích thước $(\frac{n-k+2p}{s} + 1) \times (\frac{n-k+2p}{s} + 1)$. Trong trường hợp $(n - k + 2p)$ không chia hết cho s , ta sẽ làm tròn xuống.

e. Sự tương đồng so với mạng nơ – ron nhân tạo

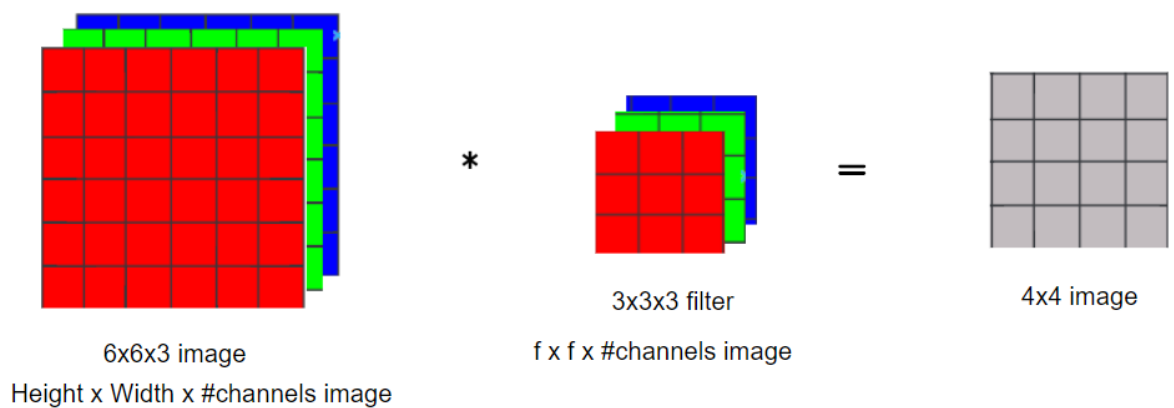
Sự tương đồng giữa các thành phần của phép tích chập so với các thành phần trong mạng nơ-ron nhân tạo được thể hiện trong sơ đồ dưới đây:



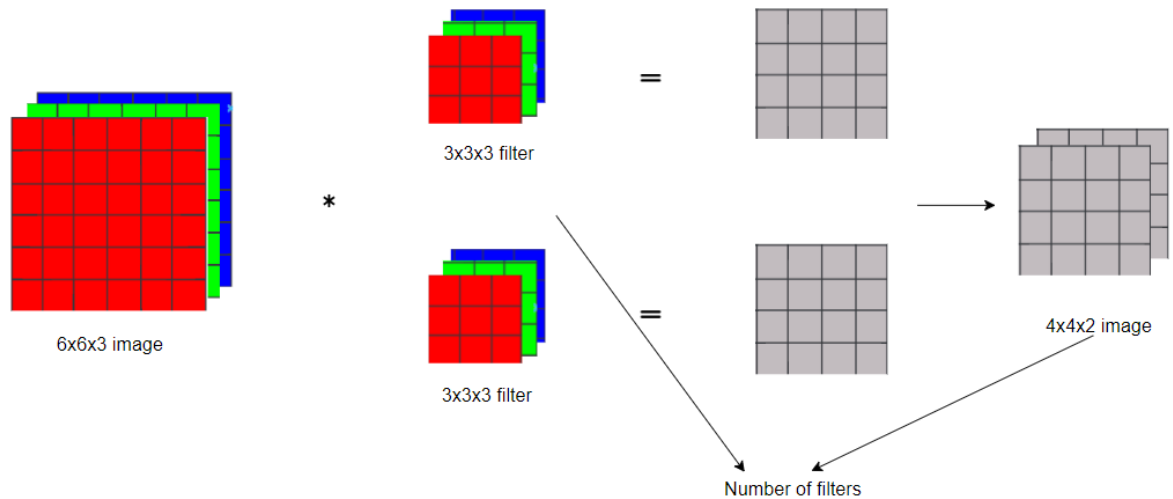
II.2.2. Phép tích chập cho ảnh màu

Ở những ví dụ trên, ta đang xét đầu vào là một ma trận tương ứng với một bức ảnh xám. Tuy nhiên, hiện nay ta cần làm việc nhiều hơn với các bức ảnh màu.

Theo phần II.1 đã xem xét ở trên, với một bức ảnh màu được biểu diễn theo hệ màu RGB ta sẽ cần 3 channel tương ứng với 3 ma trận. Biểu diễn một phép tích chập cơ bản như sau:



Filter cũng phải có chiều sâu bằng với chiều sâu của ảnh đầu vào. Ví dụ trên ta chỉ sử dụng một filter nên kết quả output vẫn là ảnh 2 chiều vì phép tích chập không được triển khai trên chiều sâu. Tuy nhiên như vậy sẽ dẫn đến đầu vào của phép tích chập tiếp theo chỉ là ảnh 2 chiều. Để bức ảnh đầu ra cuối cùng có chiều sâu, ta đi tính toán phép tích chập với nhiều filter. Một bức ảnh thực tế có nhiều góc cạnh đặc trưng cần phát hiện, do đó, ta sẽ sử dụng nhiều filter, mỗi filter có các giá trị khác nhau.



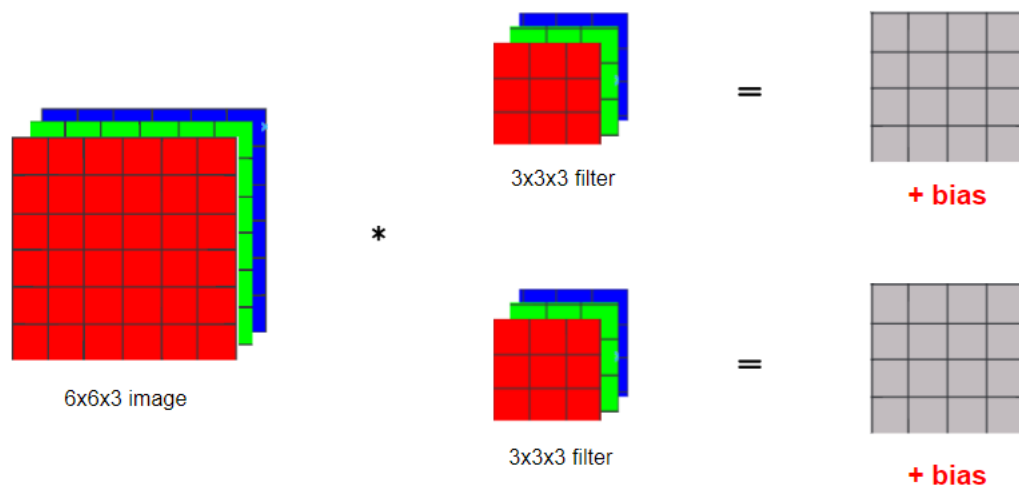
Ma trận đầu ra sẽ có kích thước $(n - f + 1) \times (n - f + 1) \times n'_c$. Trong đó n'_c là số lượng filter. Nếu sử dụng phương pháp padding hoặc strided thì kích thước ma trận có sự thay đổi tương ứng như đã xem xét ở những phần trên.

II.2.3. Các loại lớp (layer) trong mạng nơ – ron tích chập

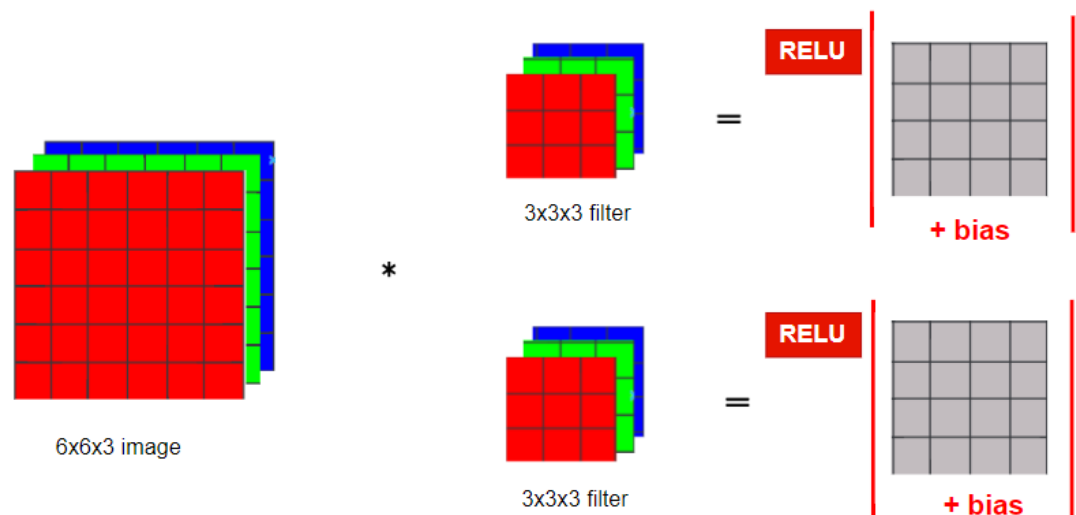
Trong mạng nơ – ron tích chập, đầu ra của lớp trước sẽ thành đầu vào của lớp tiếp theo. Ta đi tìm hiểu các loại lớp được sử dụng để xây dựng mạng.

a. Lớp tích chập cơ bản (Convolution layer)

Sau khi thực hiện phép tích chập với một ảnh đầu vào, để hoàn thiện một lớp tích chập ta cần thêm các bias và tính toán kết quả qua các hàm kích hoạt.



H7. Sau khi thực hiện tích chập sẽ cộng thêm bias



H8. Sau đó, áp dụng hàm kích hoạt (ví dụ với hàm ReLU)

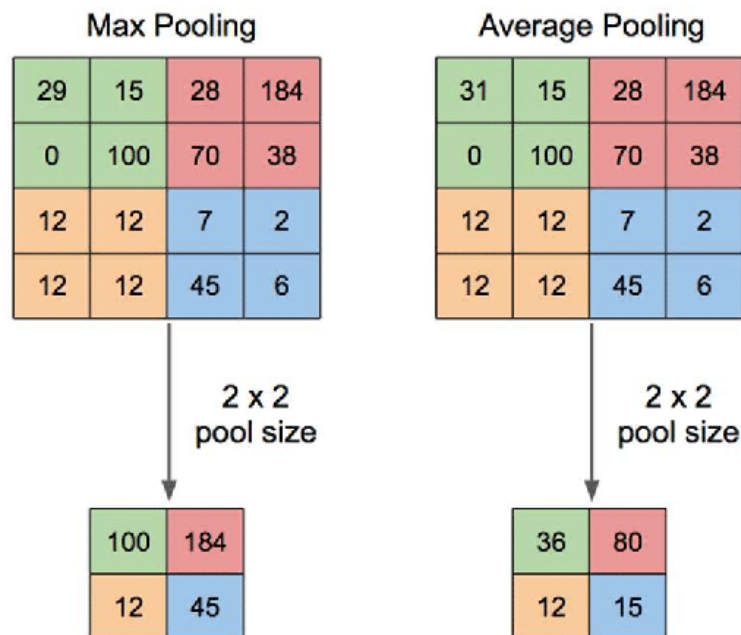
Để đưa các ma trận hoặc các tensor qua hàm kích hoạt, ta tính toán với từng giá trị của ô trong ma trận, tensor rồi ghi lại kết quả vào đúng ô đó.

b. Lớp tổng hợp (Pooling layer)

Lớp tổng hợp thường được dùng xen giữa các lớp tích chập nhằm giảm kích thước dữ liệu đầu vào nhưng vẫn giữ được các thuộc tính quan trọng. Kích thước dữ liệu giảm giúp ích rất nhiều trong việc tăng tốc độ tính toán của mô hình.

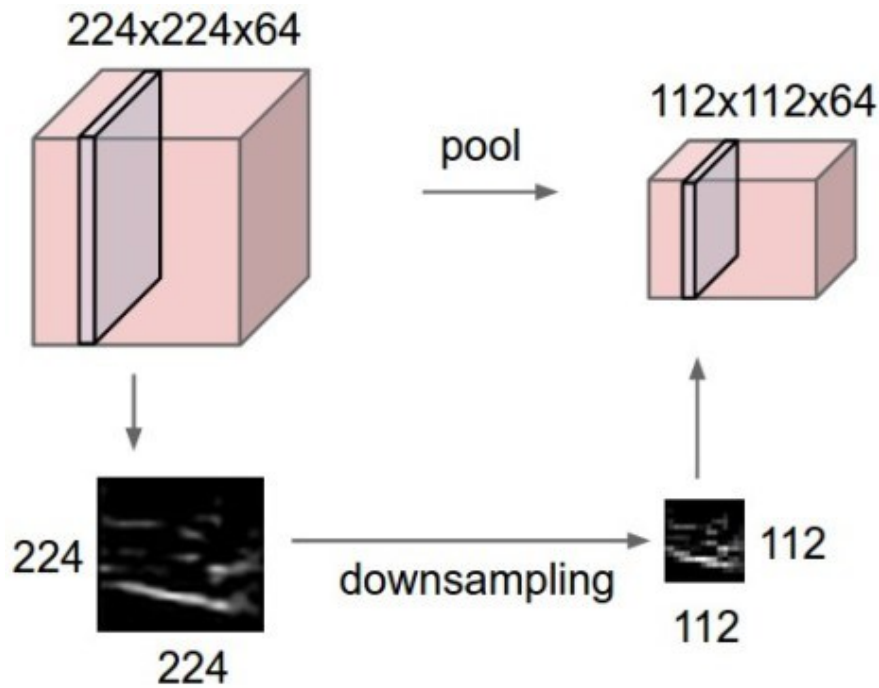
Hai loại lớp tổng hợp được sử dụng phổ biến nhất là Max pooling và Average pooling.

Giả sử ta có một pooling kích thước $k \times k$. Đầu vào của pooling layer có kích thước $n_H \times n_W \times n_C$ (chiều dài \times chiều rộng \times chiều sâu), ta tách tensor ra làm n_C ma trận kích thước $n_H \times n_W$. Với mỗi ma trận, trên vùng kích thước $k \times k$ ta tìm giá trị lớn nhất hoặc giá trị trung bình của dữ liệu (tương ứng với Max pooling và Average pooling). Quy tắc về padding và strided được áp dụng như phép tính tích chập trên ảnh.



H9. Ví dụ cơ bản về 2 loại pooling layer (sử dụng bước nhảy $s = 1$)

Max pooling được sử dụng thường xuyên hơn trong mạng nơ-ron tích chập, và thực tế thường lựa chọn pooling layer với kích thước 2×2 , bước nhảy $s = 2$, padding $p = 0$. Khi đó đầu ra sẽ có chiều dài và chiều rộng giảm đi một nửa, chiều sâu được giữ nguyên.



H10. Sử dụng pooling layer 2x2

(<http://cs231n.github.io/convolutional-networks/>)

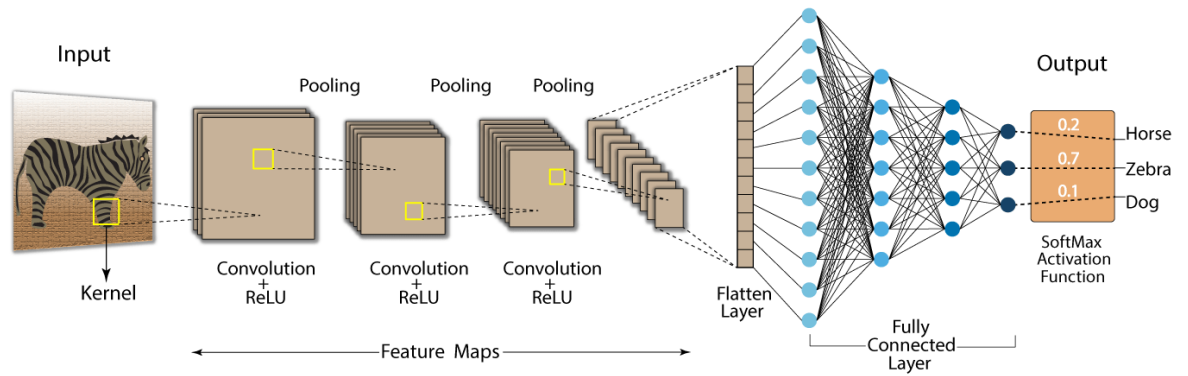
c. Làm phẳng (Flatten layer)

Sau khi thông tin được trích xuất từ các lớp tích chập, chúng ta cần làm phẳng (flatten) nó thành một vector một chiều. Flatten layer thực hiện công việc này. Việc sử dụng Flatten layer chính là cách để chuyển đổi từ mạng nơ – ron tích chập sang mạng nơ – ron nhân tạo và cho phép áp dụng các kiến trúc mạng nơ-ron tiêu biểu để thực hiện các nhiệm vụ phân loại hoặc dự đoán.

d. Lớp kết nối hoàn toàn (Fully connected layer)

Sau khi được làm phẳng, vector này sẽ được đưa vào các lớp fully connected để thực hiện quyết định cuối cùng. Các lớp Dense thường được sử dụng ở cuối mạng để thực hiện phân loại. Mỗi node trong lớp fully connected kết nối với tất cả các node trong lớp trước đó, do đó nó được gọi là fully connected layer.

Tensor đầu ra của layer cuối cùng, kích thước $n_H \times n_W \times n_C$ sẽ được chuyển về 1 vector kích thước bằng tích ($n_H \times n_W \times n_C$). Sau đó ta dùng các fully connected layer để kết hợp các đặc điểm của ảnh để có được đầu ra của mô hình



H11. Ví dụ về một mô hình mạng nơ – ron tích chập dự đoán hình ảnh động vật
(<https://www.analyticsvidhya.com/blog/2022/03/basics-of-cnn-in-deep-learning/>)

II.3. Mạng YOLO (You Only Look Once)

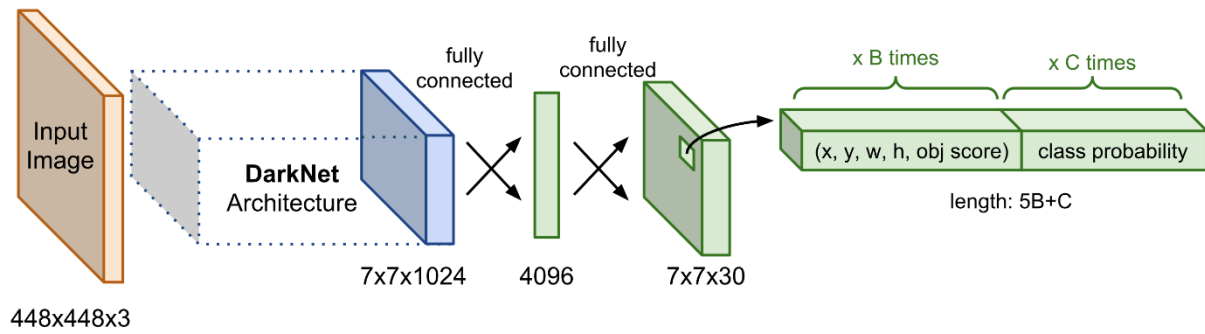
II.3.1. Tổng quan

YOLO (You Only Look Once) là một thuật toán phát hiện đối tượng thời gian thực sử dụng mạng nơ-ron tích chập (CNN). Trong đó các Convolutional layers sẽ trích xuất ra các đặc tính của ảnh, còn Fully-connected layers sẽ dự đoán ra xác suất đó và tọa độ của đối tượng.

Các khái niệm:

- Bounding box: là khung hình bao quanh vật thể.
- Anchor box: là những khung hình có kích thước xác định trước, có tác dụng dự đoán bounding box.
- Feature map: là một khối đầu ra mà ta sẽ chia nó thành một lưới ô vuông và áp dụng tìm kiếm và phát hiện vật thể trên từng ô.
- Non-max suppression: Phương pháp giúp giảm thiểu nhiều bounding box bị chồng lên nhau về 1 bounding box có xác suất lớn nhất.

II.3.2. Kiến trúc mạng YOLO

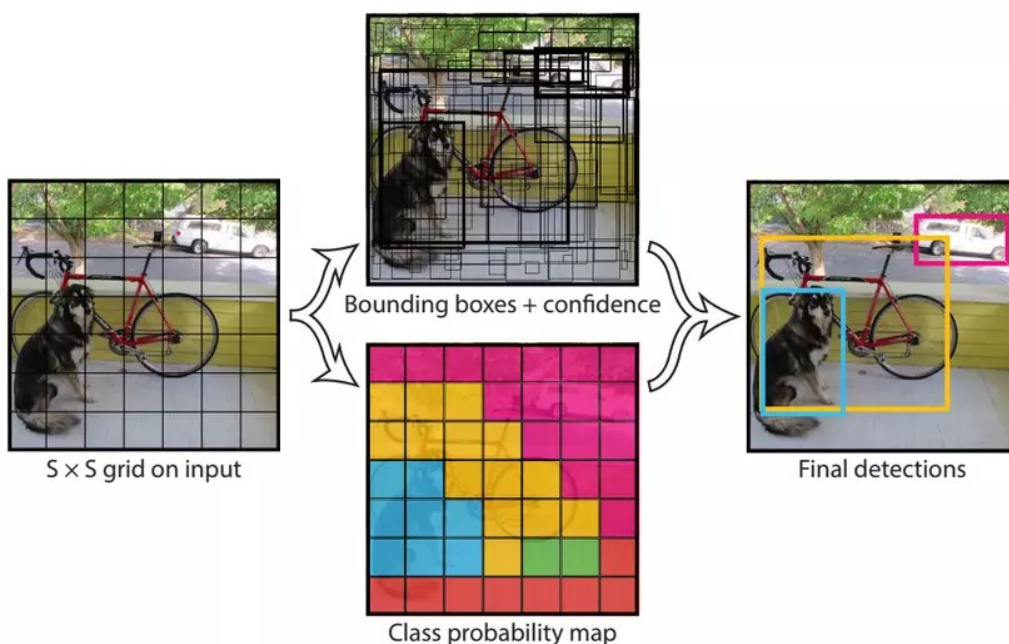


H12. Kiến trúc cơ bản mạng YOLO

Mạng bao gồm một thành phần DarkNet Architecture được gọi là base network có tác dụng trích xuất đặc trưng của ảnh. Đầu ra của base network là một feature map có kích thước $7 \times 7 \times 1024$ được sử dụng làm đầu vào cho 2 lớp Fully-connected có tác dụng dự đoán nhãn và tọa độ bounding box của vật thể. Thành phần DarkNet Architecture chủ yếu là các Convolutional layer và Fully-connected layer, mạng này được xây dựng khác nhau tùy theo các phiên bản YOLO.

II.3.3. Cách hoạt động

YOLO chia ảnh đầu vào thành một lưới $S \times S$ (ví dụ: 7×7). Mỗi ô lưới sẽ chịu trách nhiệm phát hiện các đối tượng mà trung tâm của chúng nằm trong ô đó.



Mỗi ô lưới sẽ dự đoán một số lượng cố định các bounding box (thường là B bounding box (ví dụ: 2). Mỗi bounding box bao gồm:

- Tọa độ trung tâm
- Chiều rộng và chiều h
- Giá trị confidence (độ tin cậy) của bounding box, thể hiện xác suất có đối tượng trong bounding box và độ chính xác của bounding box so với đối tượng thực.

Đầu ra là một véc tơ có kích thước $S \times S \times (B \times 5 + C)$.

Mỗi ô lưới cũng sẽ dự đoán các xác suất điều kiện (conditional class probabilities) cho các lớp đối tượng. Điều này thể hiện xác suất thuộc về mỗi lớp cho đối tượng nếu có một đối tượng trong ô lưới đó.

YOLO có thể dự đoán nhiều bounding box cho cùng một đối tượng. Để xử lý điều này, YOLO sử dụng thuật toán Non-Maximum Suppression để loại bỏ các bounding box trùng lặp và giữ lại bounding box có độ tin cậy cao nhất.

II.3.4. Output của YOLO

Output của mô hình YOLO là một véc tơ sẽ bao gồm các thành phần:


$$y^T = \left[p_0, \underbrace{\langle t_x, t_y, t_w, t_h \rangle}_{\text{bounding box}}, \underbrace{\langle p_1, p_2, \dots, p_c \rangle}_{\text{scores of c classes}} \right]$$

Trong đó:

- p_0 là xác suất dự đoán vật thể xuất hiện trong bounding box.
- t_x, t_y, t_w, t_h lần lượt là tọa độ tâm và kích thước rộng, dài của bounding box.
- p_1, p_2, p_3, p_4 là véc tơ phân phối xác suất dự báo của các lớp vật thể.

II.3.5. Hàm tính IoU (Intersection Over Union)

IoU là hàm được sử dụng trong các bài toán phát hiện đối tượng để đánh giá mức độ chồng lấn giữa hai bounding box: một bounding box dự đoán và một bounding box thực tế. Chỉ số này được tính bằng cách lấy diện tích giao nhau của hai bounding box chia cho diện tích hợp nhất của chúng. Hàm tính IoU có thể được diễn tả như sau:

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$


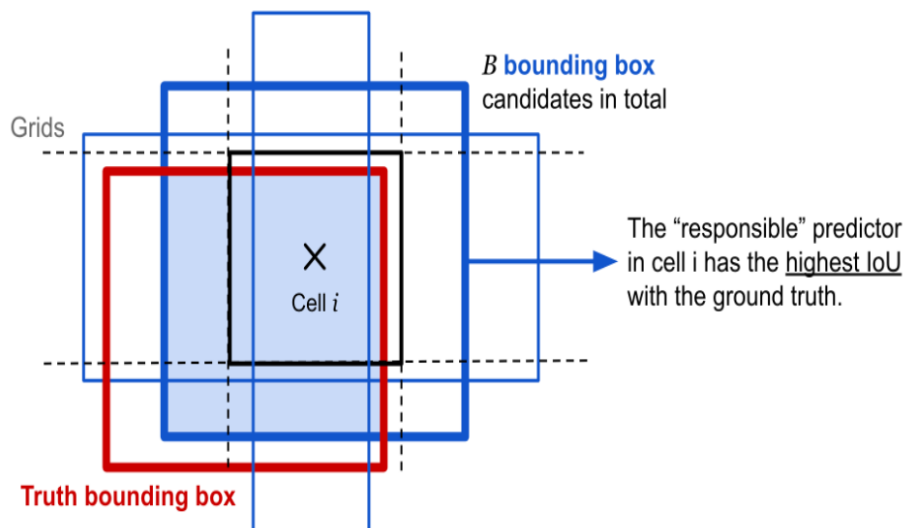
H13. Mô tả công thức tính IoU

II.3.6. Anchor box

Anchor box là các khung tham chiếu có kích thước và tỉ lệ cố định được sử dụng trong các mô hình phát hiện đối tượng YOLO. Chức năng của anchor box là giúp mô hình dự đoán các bounding box chính xác hơn bằng cách cung cấp các khung hình ban đầu hợp lý cho các đối tượng có kích thước và tỉ lệ khác nhau. Mô hình sẽ dựa vào các anchor box này để dự đoán sự dịch chuyển và tỷ lệ của các bounding box thực tế, từ đó tăng cường hiệu suất và độ chính xác của việc phát hiện đối tượng trong ảnh.

Mỗi một vật thể trong hình ảnh huấn luyện được phân bố về một anchor box. Trong trường hợp có từ 2 anchor boxes trở lên cùng bao quanh vật thể thì ta sẽ xác định anchor box mà có IoU với ground truth bounding box là cao nhất.

Như vậy khi xác định một vật thể ta sẽ cần xác định 2 thành phần là ô lưới chứa điểm trung tâm của nó và anchor box.



H14. Có thể có nhiều anchor boxes được xác định trên cùng một vật thể

Giả ví dụ trong hình trên, có một vật thể được xác định với điểm trung tâm ở ô $Cell\ i$, từ ô lưới này xác định được 3 anchor boxes phù hợp (3 khung màu xanh dương). Sau khi xét IoU của từng anchor box với truth bounding box ta sẽ lựa chọn anchor box màu xanh đậm hơn vì có giá trị IoU cao nhất.

II.3.7. Hàm mất mát

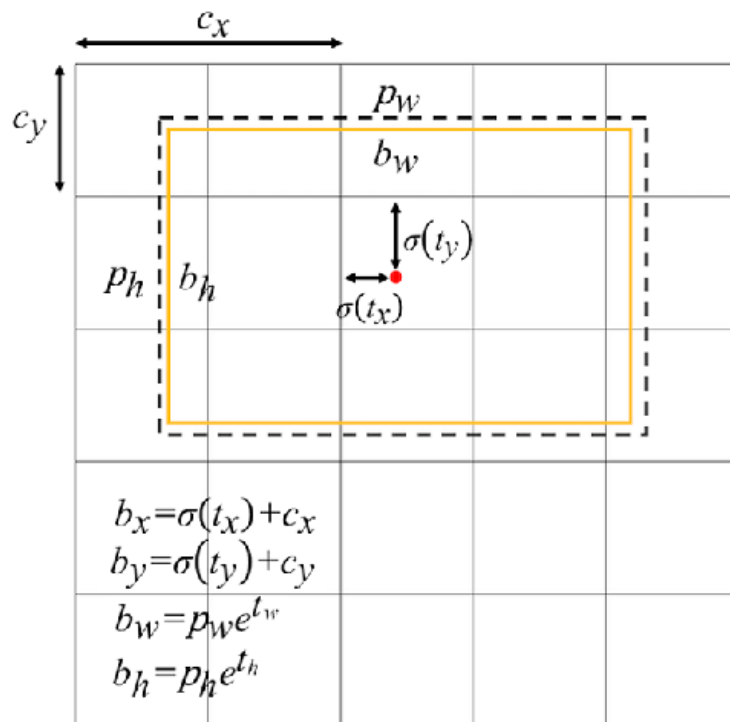
Hàm mất mát của YOLO chia thành 2 phần: L_{loc} (localization loss) đo lường sai số của bounding box so với thực tế và L_{cls} (confidence loss) đo lường sai số của phân phối xác suất các classes.

$$\begin{aligned}
 \mathcal{L}_{loc} &= \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] \\
 \mathcal{L}_{cls} &= \underbrace{\sum_{i=0}^{S^2} \sum_{j=0}^B (\mathbb{1}_{ij}^{obj} + \lambda_{noobj}(1 - \mathbb{1}_{ij}^{obj})) (C_{ij} - \hat{C}_{ij})^2}_{\text{cell contain object}} + \underbrace{\sum_{i=0}^{S^2} \sum_{c \in \mathcal{C}} \mathbb{1}_i^{obj} (p_i(c) - \hat{p}_i(c))^2}_{\text{probability distribution classes}} \\
 \mathcal{L} &= \mathcal{L}_{loc} + \mathcal{L}_{cls}
 \end{aligned}$$

- $\mathbb{1}_i^{obj}$: hàm indicator có giá trị 0, 1 nhằm xác định xem cell i có chứa vật thể hay không. Bằng 1 nếu chứa vật thể và 0 nếu không chứa.
- $\mathbb{1}_{ij}^{obj}$: Cho biết bounding box thứ j của cell i có phải là bounding box của vật thể được dự đoán hay không.

- C_{ij} : Điểm tin cậy của ô i , $P(\text{contain object}) * \text{IoU}(\text{predict box, ground truth box})$.
- \hat{C}_{ij} : Điểm tự tin dự đoán.
- C : Tập hợp tất cả các lớp.
- $p_i(c)$: Xác suất có điều kiện, có hay không ô i có chứa một đối tượng của lớp $c \in C$.
- $\hat{p}_i(c)$: Xác suất có điều kiện dự đoán.

II.3.8. Cách thức dự đoán bounding box



H15. Công thức dự đoán bounding box

Mô hình xác định các thông số của bounding box: (b_x, b_y) là tọa độ tâm, (b_w, b_h) lần lượt là kích thước chiều rộng, chiều dài của nó thông qua các tham số và hàm sigmoid, hàm exponential. Trong đó:

- c_x, c_y : khoảng cách từ góc trên bên trái của bức ảnh đến góc trên bên trái của ô lưới chứa tâm của đối tượng.

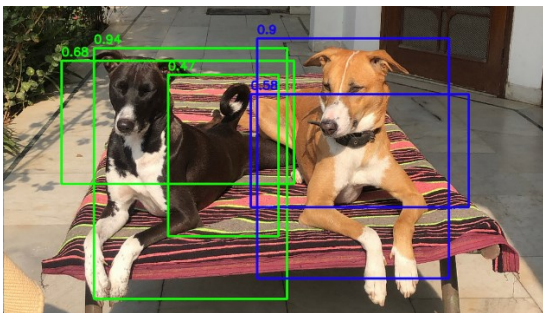
- t_x, t_y, t_w, t_h : 4 tham số mô hình dự đoán trong đó 2 tham số đầu là độ dịch chuyển của tâm bounding box so với tâm của ô lưới, 2 tham số sau là tỉ lệ thay đổi kích thước của bounding box so với anchor box.
- p_w, p_h : kích thước của anchor box.

II.3.9. Non – max suppression

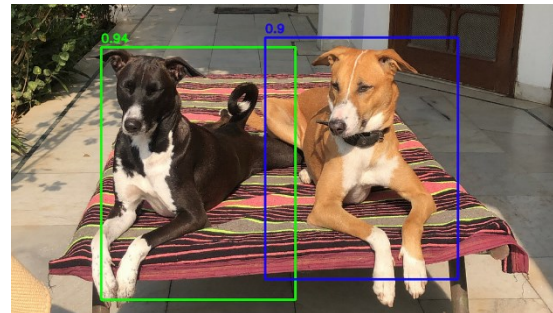
Non-Maximum Suppression (NMS) là một kỹ thuật được sử dụng trong các bài toán phát hiện đối tượng để loại bỏ các bounding box dư thừa cho cùng một đối tượng.

Đầu vào của thuật toán là một danh sách ban đầu các bounding box với các thông số tạo độ và độ tin cậy của nó. Quy trình NMS gồm các bước như sau:

- Chọn bounding box có xác suất cao nhất và thêm nó vào danh sách các bounding box cuối cùng.
- Tính toán IoU (Intersection over Union) giữa bounding box vừa chọn và các bounding box còn lại.
- Loại bỏ các bounding box có IoU lớn hơn một ngưỡng định trước (ví dụ 0.5) với bounding box vừa chọn.
- Lặp lại quá trình từ bước 1 cho đến khi không còn bounding box nào trong danh sách ban đầu.



H16. Nhiều bounding box trên cùng đối tượng



H17. Sau khi áp dụng NMS

CHƯƠNG III. XÂY DỰNG VÀ KIỂM THỬ

III.1. Thông tin tập dữ liệu huấn luyện

Để xây dựng một mô hình học máy, học sâu hoàn chỉnh, thành phần không thể thiếu đó là tập dữ liệu.

Về dữ liệu, em sử dụng bộ dữ liệu biển báo giao thông German Traffic Sign Detection Benchmark (GTSDB) để huấn luyện và kiểm thử mô hình YOLO, tương tự là German Traffic Sign Recognition Benchmark (GTSRB) cho mô hình CNN. Hai tập dữ liệu đều có sẵn trên Kaggle. Bộ dữ liệu được chia sẵn thành các thư mục sample, train, valid, test.



H18. Một số hình ảnh trong bộ dữ liệu GTSRB

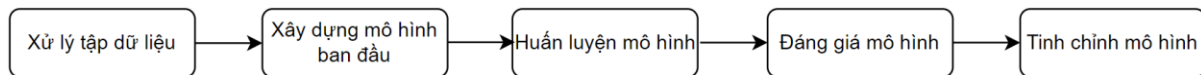


H19. Một số hình ảnh trong bộ dữ liệu GTSDB

III.2. Xây dựng mô hình

Sau khi tìm hiểu lý thuyết tổng quan như phần trên của báo cáo, em nhận thấy mô hình YOLO có thể giải quyết bài toán phát hiện biển báo giao thông theo thời gian thực và mô hình mạng nơ – ron tích chập có thể giải quyết bài toán phân loại biển báo giao thông. Do đó, ta sẽ kết hợp hai mô hình trên để giải quyết vấn đề đề tài đặt ra.

Để xây dựng mỗi mô hình, em đã làm theo quy trình sau:



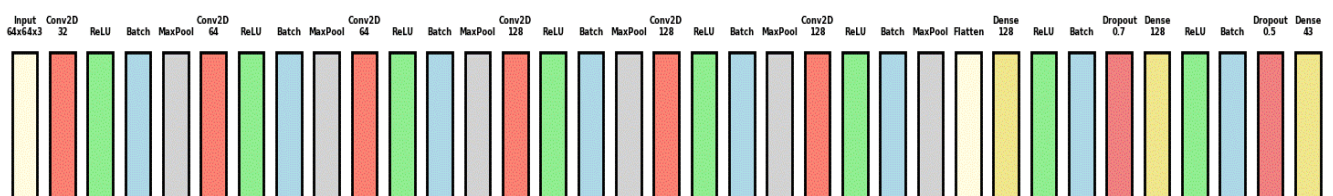
Trước tiên, em huấn luyện mô hình mạng CNN hoàn chỉnh bằng cách sử dụng tập dữ liệu GTSRB, lưu lại mô hình để thực hiện phân loại sau này. Sau đó, em sử dụng tập dữ liệu GTSDDB để đào tạo mô hình mạng YOLO với sự trợ giúp của thư viện Python YOLOv8 và lưu lại bộ tham số tốt nhất được lưu lại để sử dụng lại.

Kết hợp hai mô hình đã xây dựng trước để xây dựng phần mềm, với đầu vào là video thời gian thực từ camera hoặc video có sẵn trước đó. Mô hình YOLO đưa ra dự đoán về việc phát hiện biển báo giao thông trong mọi khung hình, các bounding box được vẽ sau đó chúng được cắt khỏi khung hình và dùng làm đầu vào cho mô hình CNN để phân loại biển báo giao thông. Kết quả hiển thị tên biển báo giao thông được dự đoán.

III.2.1. Mô hình mạng CNN

Mô hình CNN được đào tạo với bộ dữ liệu GTSRB bao gồm hơn 51.000 hình ảnh với 39.209 hình ảnh training và 12.630 hình ảnh valid. Bộ dữ liệu bao gồm 43 loại biển báo giao thông.

Qua thực nghiệm và kinh nghiệm cá nhân, em xây dựng mạng nơ – ron tích chập như sau:



Xây dựng mô hình bằng đoạn mã sử dụng thư viện Keras trong Python:

```
model = Sequential()

model.add(Conv2D(32, (3, 3), padding='same', input_shape=(64, 64, 3)))
model.add(ReLU())
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(64, (3, 3), padding='same'))
model.add(ReLU())
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(64, (3, 3), padding='same'))
model.add(ReLU())
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(128, (3, 3), padding='same'))
model.add(ReLU())
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))
```

```
model.add(Conv2D(128, (3, 3), padding='same'))
model.add(ReLU())
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(128, (3, 3), padding='same'))
model.add(ReLU())
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))

model.add(Flatten())

model.add(Dense(128))
model.add(ReLU())
model.add(BatchNormalization())
model.add(Dropout(0.7))

model.add(Dense(128))
model.add(ReLU())
model.add(BatchNormalization())
model.add(Dropout(0.5))

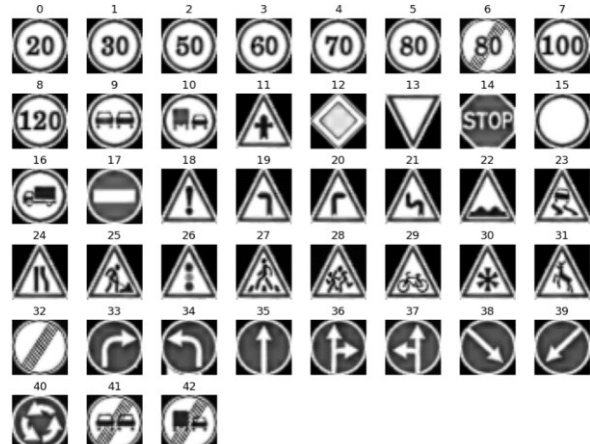
model.add(Dense(43, activation='softmax'))
```

Các hình ảnh được tải từ tập dữ liệu được tiền xử lý:

- Thay đổi về kích thước 32x32 pixel.
- Sử dụng kỹ thuật cân bằng histogram để chuẩn hóa độ sáng và giá trị của hình ảnh, và các giá trị hình ảnh được chuẩn hóa trong khoảng từ 0 đến 1.
- Tăng cường hình ảnh với các kỹ thuật: xoay ngẫu nhiên trong khoảng -10 đến +10 độ; phóng to và thu nhỏ 20%; dịch chuyển chiều cao và chiều rộng trong khoảng 10%; cắt góc biến dạng 10 độ.

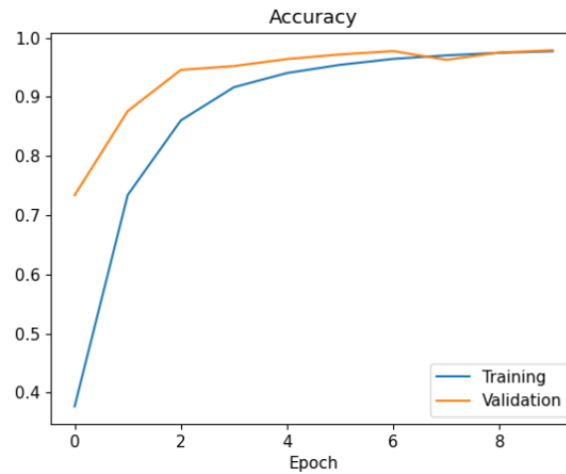


H20. sample 43 class ban đầu



H21. Sample 43 classes sau khi tiền xử lý

Kết quả: độ chính xác đạt được khi huấn luyện là 97.73% và độ chính xác khi kiểm thử đạt 97.90%.



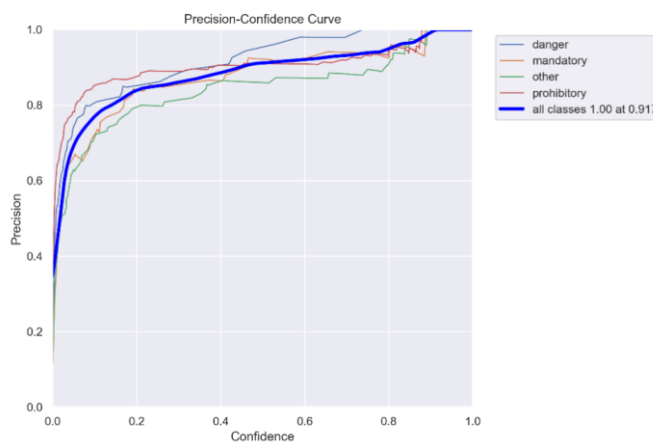
H22. Biểu đồ độ chính xác khi thay đổi số lượng epoch

Mô hình được lưu lại dưới tên 'traffic_sign_cnn_20207644.model'.

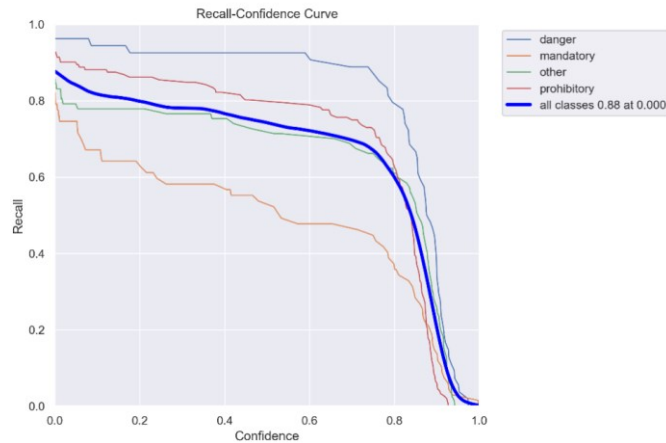
III.2.2. Mô hình YOLO

Ứng dụng thư viện Ultralytics' YOLOv8 bằng Python để huấn luyện mạng này. Dữ liệu đã được chú thích trước dựa trên một phần của tập GTSDDB được sử dụng để huấn luyện mô hình. Quá trình huấn luyện được thực hiện trong 100 epoch với kích thước hình ảnh 640x640 pixel và kích thước batch là 16.

Kết quả: Độ chính xác đạt được với 91.7% precision tại mức độ tin cậy 1.0, và 88% recall tại mức độ tin cậy 0.0. Thời gian xử lý trung bình ghi nhận là 45ms, cho phép xử lý ở hơn 20 khung hình mỗi giây (fps).



H23. Biểu đồ Precision



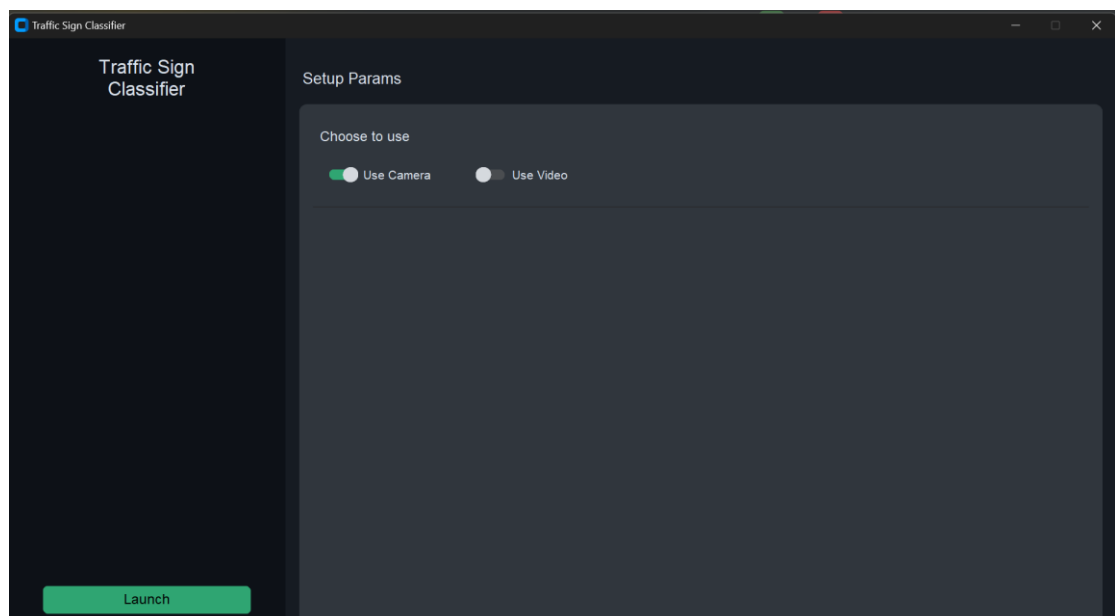
H24. Biểu đồ Recall

Sau khi huấn luyện, các trọng số tốt nhất được lưu lại để sử dụng sau.

III.3. Demo

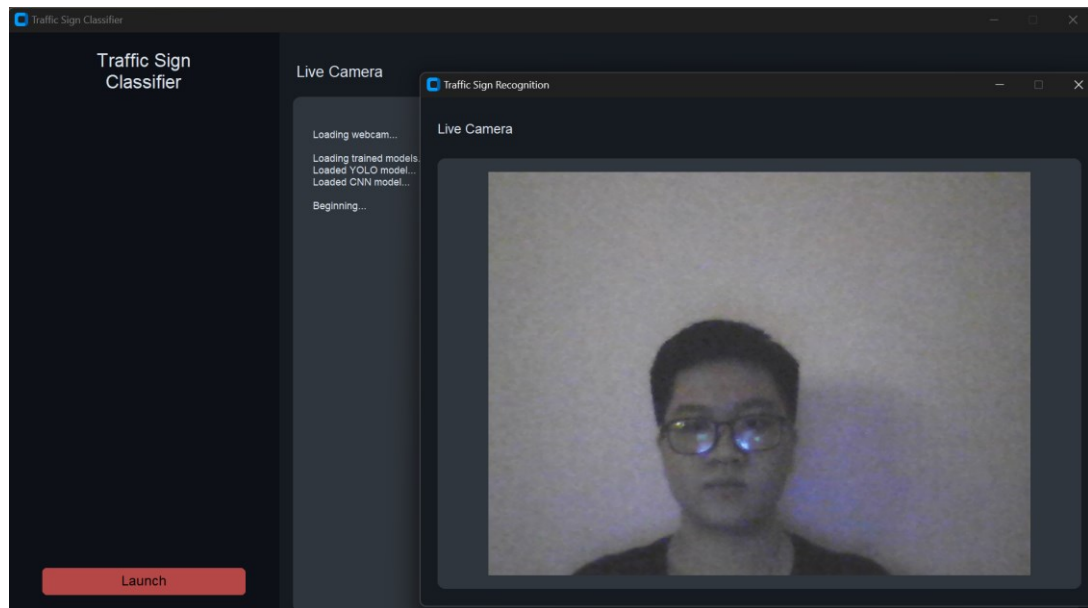
Em sẽ xây dựng một ứng dụng với giao diện đơn giản để tiện cho việc demo kết hợp hai mô hình trên. Đầu tiên, sử dụng mô hình đã huấn luyện và lưu lại ở trên.

Màn hình ban đầu, có thể chọn sử dụng live camera hoặc sử dụng video từ nguồn có sẵn.

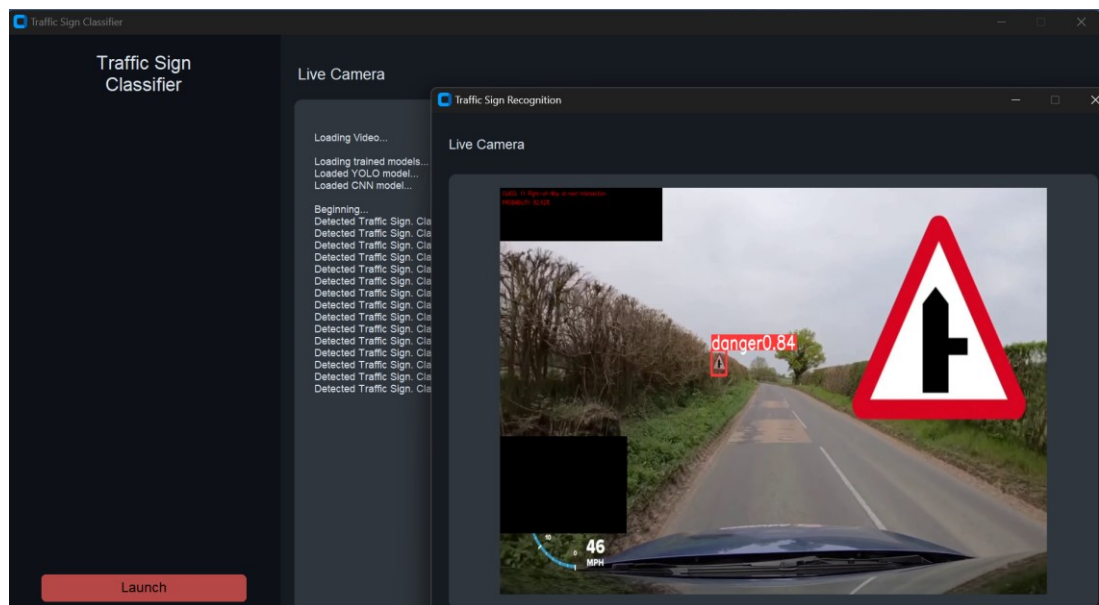


Khung chọn cài đặt sẽ được chuyển đổi thành nơi hiển thị thông tin từ console.

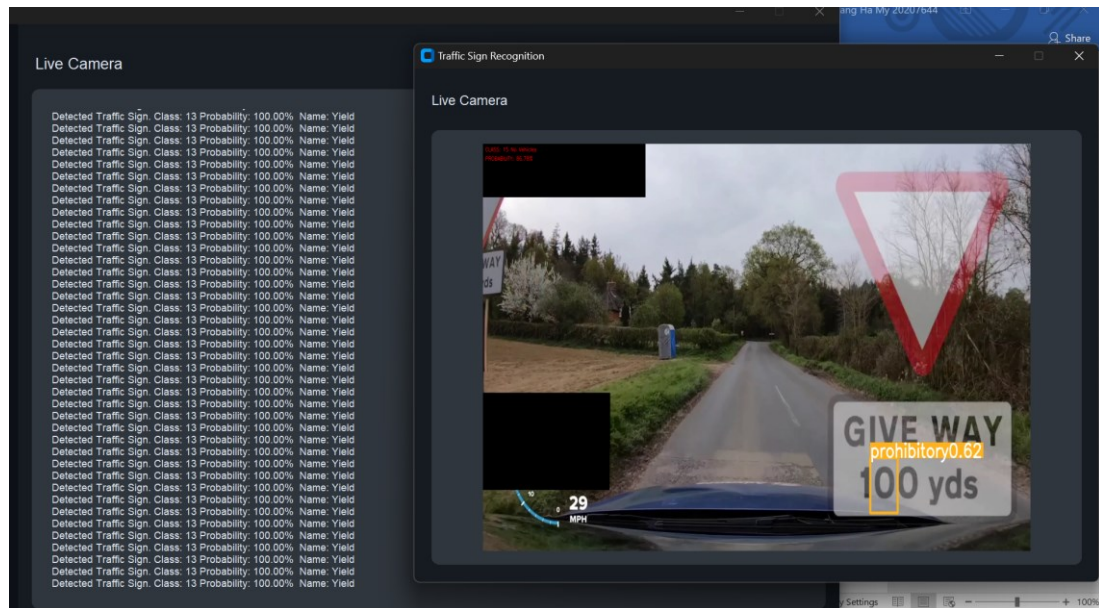
Nếu chọn Use Camera, cửa sổ mới hiện ra kích hoạt camera laptop:



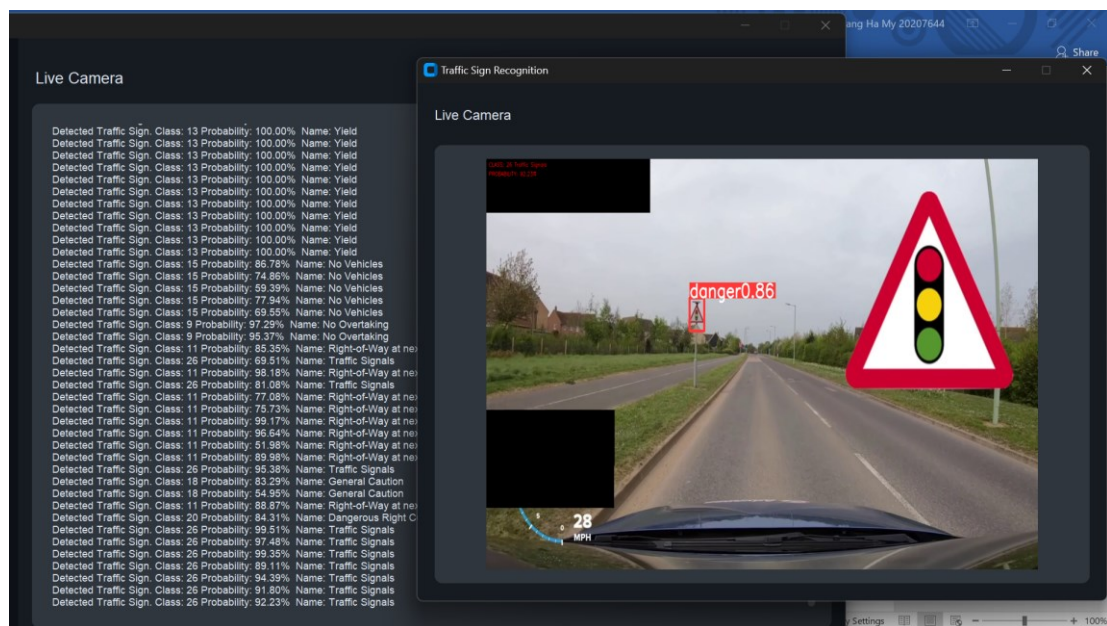
Nếu chọn Use Video, chương trình sẽ sử dụng video có sẵn.



Ví dụ một đoạn kết quả demo:



Dự đoán đúng biển báo "Yield".



Dự đoán đúng biển báo "Traffic Sign".

KẾT LUẬN

Qua thời gian nghiên cứu thực hiện, em đã nắm bắt được lý thuyết tổng quan về học máy, học sâu, các khái niệm về mạng nơ – ron nhân tạo, mạng nơ – ron tích chập, mạng YOLO và xây dựng được mô hình phát hiện và phân loại biển báo giao thông với độ chính xác khá tốt.

Tuy nhiên, do thời gian và hiểu biết của em còn hạn chế nên hiện nay chương trình chỉ dừng lại ở mức thử nghiệm được trên máy tính cá nhân, hướng phát triển tiếp theo của đề tài là áp dụng vào các hệ thống xe tự lái thực tế. Đồng thời, đồ án chắc chắn cũng không tránh khỏi những thiếu sót, nên em rất mong nhận được ý kiến đóng góp chỉ dẫn từ cô.

Em xin chân thành cảm ơn cô!

TÀI LIỆU THAM KHẢO

[1] Wikipedia

[2] CS231n: Deep Learning for Computer Vision (Stanford - Spring 2023)

[3] datalya.com

[4] slideshare.net

[5] analyticsvidhya.com

...