

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
CƠ SỞ TẠI TP HỒ CHÍ MINH

BÁO CÁO TỔNG KẾT

ĐỀ TÀI NGHIÊN CỨU KHOA HỌC CỦA SINH VIÊN
NĂM 2024

**PHÂN LOẠI HÌNH ẢNH Ô TÔ
SỬ DỤNG MẠNG CNN**

38-SV-2024-TH2

Thuộc nhóm ngành khoa học: Công nghệ thông tin

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
CƠ SỞ TẠI TP HỒ CHÍ MINH

BÁO CÁO TỔNG KẾT

ĐỀ TÀI NGHIÊN CỨU KHOA HỌC CỦA SINH VIÊN
NĂM 2024

PHÂN LOẠI HÌNH ẢNH Ô TÔ
SỬ DỤNG MẠNG CNN

38-SV-2024-TH2

Thuộc nhóm ngành khoa học: Công nghệ thông tin

Sinh viên thực hiện: Đặng Thành Tân

Nam, Nữ: Nam

Dân tộc: Kinh

Lớp, khoa: D20CQCNP01-N, CNTT2

Năm thứ: 4.5/Số năm đào tạo: 4.5

Ngành học: Công nghệ phần mềm

Người hướng dẫn: ThS. Huỳnh Trung Trự

MỤC LỤC

MỞ ĐẦU.....	1
1. Lý do chọn đề tài.	1
2. Mục tiêu nghiên cứu.....	1
3. Đối tượng và phạm vi nghiên cứu.....	2
4. Phương pháp nghiên cứu.....	2
5. Ý nghĩa khoa học và thực tiễn của đề tài.	2
6. Bố cục của bài báo cáo.	3
CHƯƠNG 1: CƠ SỞ LÝ THUYẾT.....	4
1.1. Convolutional Neural Network.....	4
1.1.1. Lớp tích chập(Convolutional layer).....	5
1.1.2. Lớp gộp(Pooling layer).....	7
1.1.3. Lớp kết nối đầy đủ(Fully-connected layer).	7
1.1.4. Điều chỉnh siêu tham số.....	9
1.2. Đặc trưng chung của các mạng CNN.	11
1.3. Các mạng CNN tiêu biểu.	12
1.3.1. LeNet-5 (1998).....	12
1.3.2 AlexNet (2012).....	12
1.3.3. VGG-16 (2014).	13
1.3.4. GoogleNet - Inception-V1 (2014).....	14
1.3.5. GoogleNet - Inception-V3 (2015).....	15
1.3.6. ResNet-50 (2015).	16
1.4. EfficientNet.	16
1.4.1. Model scaling.....	16
1.4.2. EfficientNet.....	18
1.5. Một số kỹ thuật tăng cường dữ liệu(Data augmentation).	20
1.5.1 Kỹ thuật xoay ảnh.....	21
1.5.2 Kỹ thuật phóng to, thu nhỏ ảnh.	22
CHƯƠNG 2: XÂY DỰNG MÔ HÌNH.	23
2.1. Tập dữ liệu.	23
2.2. Tăng cường dữ liệu.	24
2.3. Mô hình Convolutional Neural Network.	26
2.4. Mô hình CNN với EfficientNetB3 làm base model.	30
2.5. Xây dựng chương trình demo.	34
CHƯƠNG 3: KẾT LUẬN.....	35

3.1. Kết quả thực nghiệm.....	35
3.2. Hướng phát triển.....	35
DANH MỤC TÀI LIỆU THAM KHẢO	36

DANH MỤC HÌNH ẢNH

Hình 1.1. Mô hình Convolutional Neural Network.

Hình 1.2. Các chiều của một bộ lọc.

Hình 1.3. Stride với độ trượt S.

Hình 1.4. Phép tích chập của bộ lọc và đầu vào.

Hình 1.5. Công thức hàm kích hoạt ReLU.

Hình 1.6. Biểu diễn hàm kích hoạt ReLU.

Hình 1.7. Biểu diễn lớp Fully-connected.

Hình 1.8. Công thức hàm kích hoạt sigmoid.

Hình 1.9. Biểu diễn của hàm sigmoid.

Hình 1.10. Công thức hàm kích hoạt Softmax.

Hình 1.11. Biểu diễn cách tính độ dài feature map.

Hình 1.12. Mạng Alexnet, một kiến trúc điển hình của CNN.

Hình 1.13. Kiến trúc LeNet.

Hình 1.14. Kiến trúc AlexNet.

Hình 1.15. Kiến trúc VGG-16.

Hình 1.16. Kiến trúc GoogleNet - Inception version 1.

Hình 1.17. Kiến trúc GoogleNet - Inception version 3.

Hình 1.18. Kiến trúc ResNet.

Hình 1.19. Tỷ lệ mô hình.

Hình 1.20. Mạng cơ sở EfficientNet-B0.

Hình 1.21. Hình ảnh gốc của mẫu.

Hình 1.22. Các mẫu được tạo ra từ cách xoay ảnh.

Hình 1.23. Các mẫu được tạo ra từ cách xoay ảnh.

Hình 2.1. Biểu đồ số lượng mẫu của tập dữ liệu huấn luyện (train).

Hình 2.2. Biểu đồ số lượng mẫu của tập dữ liệu kiểm tra (validation).

Hình 2.3. Biểu đồ số lượng mẫu của tập dữ liệu kiểm tra (test).

Hình 2.4. Ma trận thể hiện lỗi của mô hình.

Hình 2.5. Ma trận thể hiện lỗi của mô hình.

Hình 2.6. Giao diện demo model.

DANH MỤC TỪ VIẾT TẮT

CNN: Convolutional Neural Networks	Mạng neural tích chập
------------------------------------	-----------------------

MỞ ĐẦU.

1. Lý do chọn đề tài.

Trong bối cảnh trí tuệ nhân tạo và học sâu đang ngày càng phát triển và được áp dụng vào nhiều lĩnh vực, Convolutional Neural Networks (CNN) đã trở thành một công cụ mạnh mẽ và phổ biến trong việc xử lý và phân tích hình ảnh. Sự phát triển của CNN không chỉ mở ra những cánh cửa mới cho các ứng dụng thực tiễn mà còn thúc đẩy những bước tiến quan trọng trong nghiên cứu và phát triển công nghệ thị giác máy tính. Khi áp dụng Convolutional Neural Networks (CNN) vào phân loại hình ảnh ô tô mang lại nhiều lợi ích và giá trị, có thể kể đến như:

Tự động hóa và tối ưu hóa quy trình: Ứng dụng CNN giúp tự động hóa quá trình nhận diện và phân loại ô tô, giảm thiểu sự can thiệp của con người và tối ưu hóa các quy trình liên quan đến quản lý và giám sát phương tiện.

Ứng dụng trong thực tiễn: Công nghệ này có thể được áp dụng trong nhiều lĩnh vực như hệ thống giám sát giao thông, kiểm tra chất lượng sản phẩm trong công nghiệp ô tô.

Độ chính xác cao: mô hình CNN có khả năng đạt được độ chính xác cao trong việc nhận diện và phân loại hình ảnh, nhờ khả năng học và trích xuất các đặc trưng phức tạp từ dữ liệu hình ảnh.

Đóng góp cho lĩnh vực Khoa học và Công nghệ: Đề tài này không chỉ có ý nghĩa thực tiễn trong việc phát triển hệ thống thông minh và tự động hóa mà còn góp phần mở rộng kiến thức và ứng dụng của thị giác máy tính vào các hệ thống tự động hoá trong công nghiệp ô tô.

Được sự gợi ý của Thầy Huỳnh Trung Trụ hướng dẫn em đã chọn đề tài: “Phân loại hình ảnh ô tô sử dụng mạng CNN” làm bài nghiên cứu khoa học của mình. Mục tiêu chính của luận văn là tìm hiểu về mạng Convolutional Neural Networks, xây dựng mô hình Mạng thần kinh Convolution (CNN) trên bộ dữ liệu hình ảnh ô tô được sử dụng để dự đoán các ô tô thuộc bảy loại (Audi, Hyundai Creta, Mahindra Scorpio, Rolls Royce, Maruti Suzuki Swift, Tata Safari và Toyota Innova).

2. Mục tiêu nghiên cứu.

Mục tiêu tìm hiểu về mạng Convolutional Neural Networks, xây dựng mô hình Mạng thần kinh Convolution (CNN) trên bộ dữ liệu hình ảnh ô tô.

3. Đối tượng và phạm vi nghiên cứu.

Đối tượng nghiên cứu: mô hình CNN phân loại hình ảnh ô tô.

Phạm vi nghiên cứu:

- Tìm hiểu mạng Convolutional Neural Networks.
- Tìm hiểu dataset liên quan.
- Nghiên cứu mô hình phân loại hình ảnh ô tô.
- Xây dựng mô hình CNN phân loại hình ảnh ô tô.

4. Phương pháp nghiên cứu.

Phương pháp nghiên cứu lý thuyết: Nghiên cứu, tham khảo tài liệu về mạng Convolutional Neural Networks và thị giác máy tính thông qua sách báo, tạp chí, mạng Internet.

Phương pháp nghiên cứu thực nghiệm: Sau khi nghiên cứu lý thuyết, đề tài tập trung vào xây dựng mô hình CNN phân loại hình ảnh ô tô. Đánh giá kết quả sau khi thử nghiệm.

5. Ý nghĩa khoa học và thực tiễn của đề tài.

Ý nghĩa khoa học: Đề tài "Phân loại hình ảnh ô tô sử dụng mạng CNN" đóng góp vào phát triển lĩnh vực trí tuệ nhân tạo và học sâu, đặc biệt là trong ứng dụng của Convolutional Neural Networks (CNN). Nghiên cứu này không chỉ giúp cải thiện hiểu biết về các mô hình CNN mà còn thúc đẩy sự phát triển của các kỹ thuật mới trong xử lý ảnh và thị giác máy tính. Việc tối ưu hóa và huấn luyện các mô hình CNN để đạt được độ chính xác cao trong phân loại hình ảnh ô tô sẽ cung cấp những kiến thức quan trọng về cách thiết kế và triển khai các hệ thống học máy mạnh mẽ và hiệu quả. Những kết quả từ nghiên cứu này có thể được áp dụng trong các bài toán khác liên quan đến nhận diện và phân loại hình ảnh, góp phần vào sự tiến bộ của khoa học dữ liệu và trí tuệ nhân tạo.

Ý nghĩa thực tiễn: Về mặt thực tiễn, việc áp dụng CNN để phân loại hình ảnh ô tô mang lại nhiều lợi ích trong các lĩnh vực như quản lý giao thông, an ninh, và công nghiệp ô tô. Hệ thống nhận diện và phân loại ô tô tự động có thể được sử dụng trong các ứng dụng giám sát giao thông thông minh, giúp giảm thiểu tai nạn và cải thiện hiệu quả của hệ thống quản lý giao thông. Trong công nghiệp, các mô hình này có thể được sử dụng để kiểm tra chất lượng sản phẩm và quản lý dây chuyền sản xuất. Ngoài ra, các ứng dụng trong lĩnh vực an ninh như phát hiện xe bị đánh cắp, quản lý bãi đỗ xe thông minh, và nhận diện xe trong các khu vực cấm đỗ sẽ giúp nâng cao mức độ an toàn và hiệu quả trong quản lý phương tiện.

6. Bố cục của bài báo cáo.

Bố cục bài nghiên cứu khoa học được chia thành như sau:

Mở đầu:

- Mô tả chi tiết về mục tiêu, nhiệm vụ đề tài và hướng tiếp cận.

Chương 1: Cơ sở lý thuyết.

- Convolutional Neural Network.
- Đặc trưng chung của các mạng CNN.
- Các mạng CNN tiêu biểu.
- EfficientNet.
- Một số kỹ thuật tăng cường dữ liệu(Data augmentation).

Chương 2: Xây dựng mô hình.

- Tập dữ liệu.
- Tăng cường dữ liệu.
- Mô hình Convolutional Neural Network.
- Mô hình CNN với EfficientNetB3 làm base model.

Chương 3: Kết luận.

- Kết quả thực nghiệm.
- Hướng phát triển trong tương lai.

CHƯƠNG 1: CƠ SỞ LÝ THUYẾT.

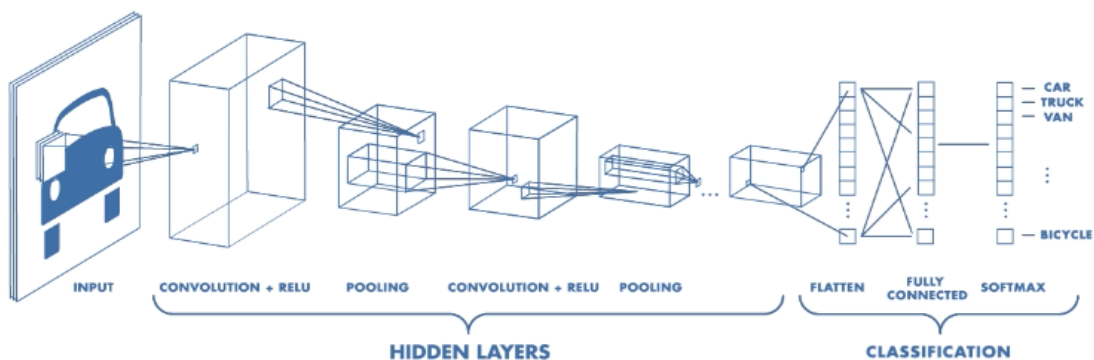
1.1. Convolutional Neural Network.

Neural Network là một tập hợp con của Machine learning và chúng là cốt lõi của các thuật toán Deep learning. Neural Network bao gồm các lớp nút, chứa một input layer, một hoặc nhiều hidden layer và một output layer. Mỗi nút kết nối với một nút khác và có trọng số và ngưỡng liên quan. Nếu đầu ra của bất kỳ nút riêng lẻ nào cao hơn giá trị ngưỡng đã chỉ định, nút đó sẽ được kích hoạt, gửi dữ liệu đến lớp tiếp theo của mạng. Ngược lại, không có dữ liệu nào được truyền đến lớp tiếp theo của mạng.

Có nhiều loại Neural Network khác nhau được sử dụng trong nhiều trường hợp phù hợp với loại dữ liệu khác nhau. Ví dụ, Recurrent Neural Network(RNN) thường được sử dụng để xử lý ngôn ngữ tự nhiên và nhận dạng giọng nói trong khi Convolutional Neural Network(CNN) thường được sử dụng cho các tác vụ phân loại và thị giác máy tính. Trước khi CNN xuất hiện, cách để xác định các đối tượng trong hình ảnh cần sử dụng các phương pháp trích xuất đặc trưng thủ công tốn rất nhiều thời gian. Bây giờ, CNN cung cấp một phương pháp dễ dàng hơn cho các tác vụ phân loại hình ảnh và nhận dạng đối tượng, tận dụng các nguyên tắc từ đại số tuyến tính như là phép nhân ma trận để xác định các mẫu trong hình ảnh. Đi đôi với sự tiện lợi đó thì CNN đòi hỏi nhiều tính toán, yêu cầu các đơn vị xử lý đồ họa(GPU) để huấn luyện mô hình.

Convolutional Neural Network được phân biệt với các Neural Network khác nhờ hiệu suất vượt trội khi ứng dụng với các đầu vào tính hiệu hình ảnh, giọng nói hoặc âm thanh. CNN có ba lớp chính [1], đó là:

- Lớp tính chập(Convolutional layer).
- Lớp gộp(Pooling layer).
- Lớp kết nối đầy đủ(Fully-connected layer) .



Hình 1.1. Mô hình Convolutional Neural Network.

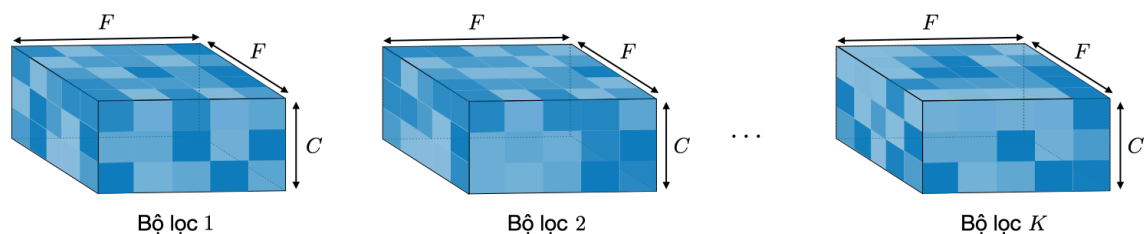
Lớp tích chập là lớp đầu tiên của Convolutional Neural Network. Trong khi các lớp tích chập có thể được theo sau bởi các lớp tích chập bổ sung hoặc các lớp gộp, thì lớp được kết nối đầy đủ là lớp cuối cùng. Với mỗi lớp, CNN tăng độ phức tạp của nó, xác định các phần lớn hơn của hình ảnh. Các lớp trước đó tập trung vào các tính năng đơn giản, chẳng hạn như màu sắc và các cạnh. Khi dữ liệu hình ảnh tiến triển qua các lớp của CNN, nó bắt đầu nhận dạng các phần tử hoặc hình dạng lớn hơn của đối tượng cho đến khi cuối cùng xác định được đối tượng mong muốn.

1.1.1. Lớp tích chập(Convolutional layer).

Lớp tích chập là một thành phần cốt lõi của Convolutional Neural Network - CNN, được thiết kế đặc biệt để xử lý dữ liệu dạng lưới như hình ảnh. Chức năng chính của lớp tích chập là phát hiện và học các đặc trưng (features) của dữ liệu đầu vào.

Lớp tích chập thực hiện phép toán tích chập giữa một bộ lọc (filter) và đầu vào để tạo ra một bản đồ đặc trưng (feature map). Bộ lọc là một ma trận nhỏ có các trọng số được học thông qua quá trình huấn luyện. Quá trình tích chập có thể được mô tả như sau:

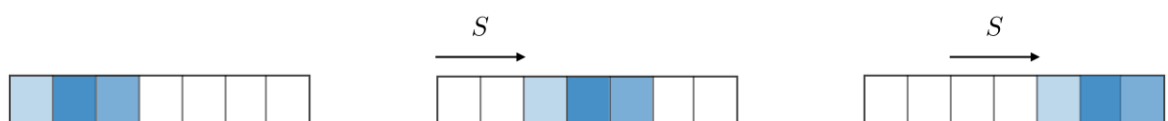
- Input: Một ma trận 2D (hoặc 3D với nhiều kênh màu).
- Filter (Kernel): Một ma trận nhỏ hơn đầu vào, thường có kích thước như 3x3, 5x5, Một bộ lọc kích thước $F \times F$ áp dụng lên đầu vào chứa C kênh (channels) thì có kích thước tổng thể là $F \times F \times C$ thực hiện phép tích chập trên đầu vào kích thước $I \times I \times C$ và cho ra một feature map (hay còn gọi là activation map) có kích thước $O \times O \times 1$.



Hình 1.2. Các chiều của một bộ lọc.

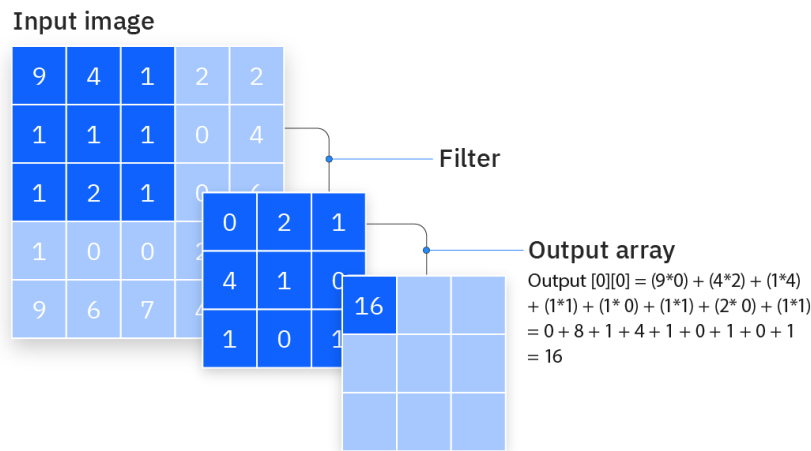
Việc áp dụng K bộ lọc có kích thước $F \times F$ cho ra một feature map có kích thước $O \times O \times K$.

- Stride: là khoảng cách hoặc số pixel mà bộ lọc di chuyển qua trên ma trận đầu vào. Giá trị của Stride thường là 1, giá trị bằng 2 hoặc lớn hơn ít khi được sử dụng, giá trị stride lớn hơn sẽ tạo ra đầu ra nhỏ hơn.



Hình 1.3. Stride với độ trượt S .

- Padding: Thêm các giá trị xung quanh biên của đầu vào để kiểm soát kích thước của bản đồ đặc trưng. Padding có thể là 'valid' (không thêm) hoặc 'same' (thêm để giữ nguyên kích thước đầu vào).

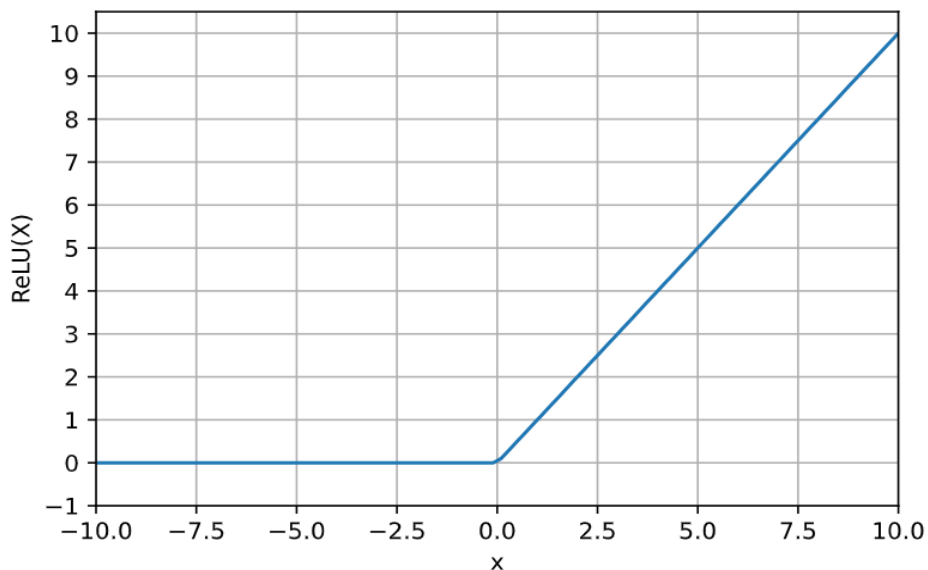


Hình 1.4. Phép tích chập của bộ lọc và đầu vào.

Sau mỗi phép toán tích chập, CNN sẽ áp dụng phép biến đổi Rectified Linear Unit (ReLU) để làm mượt kết quả đầu ra. Mục đích của nó là tăng tính phi tuyến tính cho mạng. Hàm này có chức năng đưa về kết quả dương. Cụ thể công thức như sau:

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

Hình 1.5. Công thức hàm kích hoạt ReLU.



Hình 1.6. Biểu diễn hàm kích hoạt ReLU.

1.1.2. Lớp gộp(Pooling layer).

Các lớp gộp, hay còn gọi là lớp downsampling, lớp này có chức năng chính là thực hiện giảm chiều, giảm số lượng tham số đầu vào. Tương tự như lớp tích chập, quá trình gộp được thực hiện bằng cách quét một bộ lọc trên toàn bộ đầu vào, nhưng điểm khác biệt là bộ lọc này không có bất kỳ trọng số nào. Thay vào đó, bộ lọc áp dụng một hàm tổng hợp cho các giá trị trong đầu vào tiếp nhận, ghi kết quả vào mảng đầu ra. Có hai loại gộp chính:

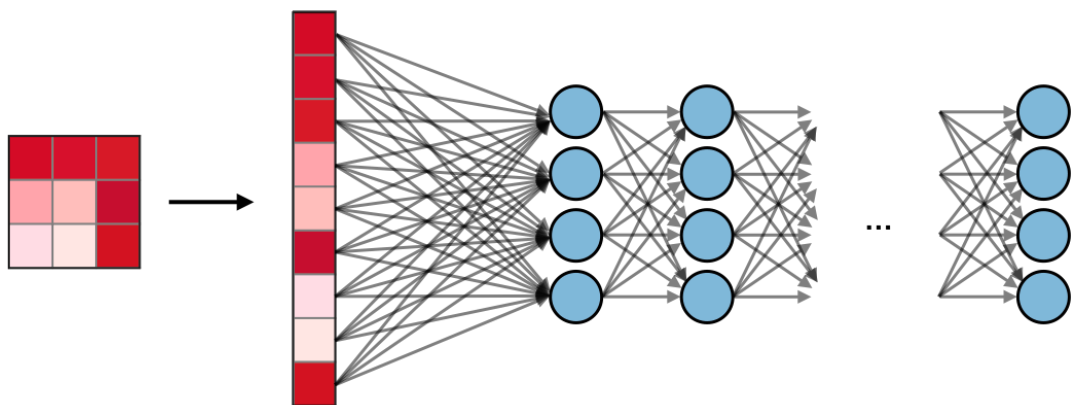
- Max pooling: Khi bộ lọc di chuyển qua đầu vào, nó sẽ chọn pixel có giá trị lớn nhất để gửi đến mảng đầu ra. Ngoài ra, cách tiếp cận này có xu hướng được sử dụng thường xuyên hơn so với average pooling.

- Average pooling: Khi bộ lọc di chuyển qua đầu vào, nó sẽ tính toán giá trị trung bình trong trường tiếp nhận để gửi đến mảng đầu ra.

Mặc dù nhiều thông tin bị mất trong lớp pooling, nhưng nó cũng có một số lợi ích cho CNN. Chúng giúp giảm độ phức tạp, cải thiện hiệu quả và hạn chế rủi ro quá khớp.

1.1.3. Lớp kết nối đầy đủ(Fully-connected layer).

Lớp này có cấu trúc như một Neural Network. Các giá trị pixel của hình ảnh đầu vào không được kết nối trực tiếp với lớp đầu ra trong các lớp mà được kết nối một phần. Tuy nhiên, trong lớp kết nối đầy đủ, mỗi nút trong lớp đầu ra kết nối trực tiếp với một nút trong lớp trước đó.



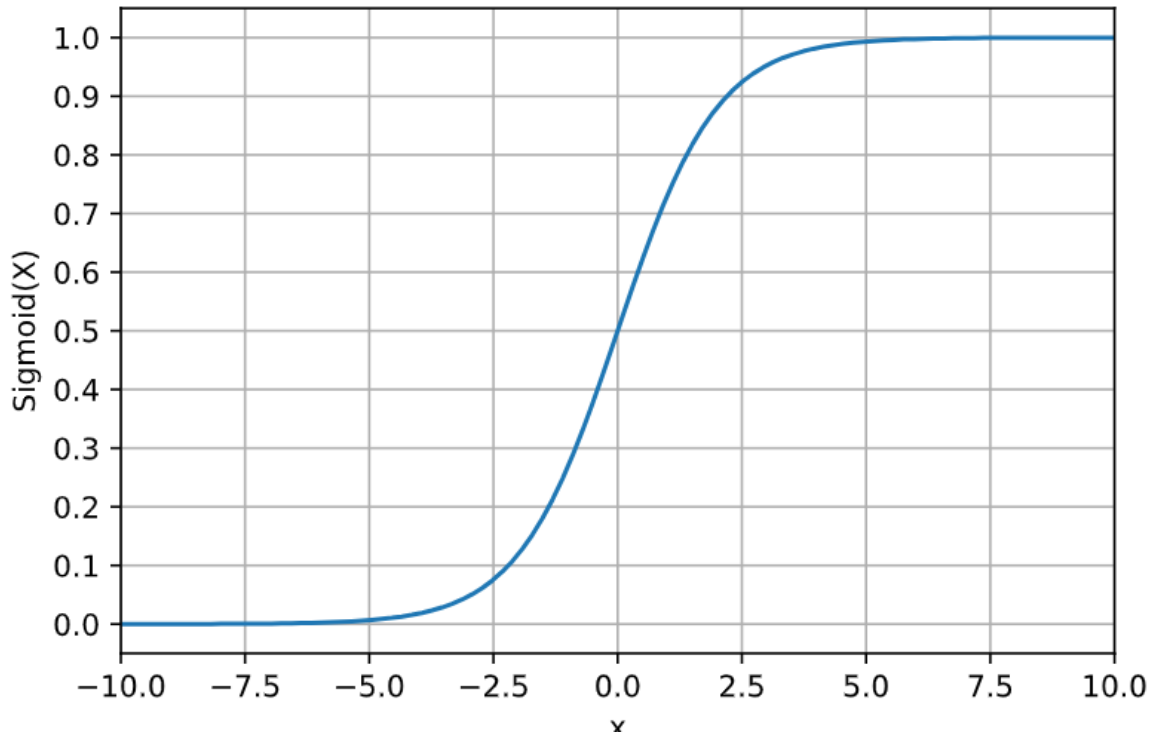
Hình 1.7. Biểu diễn lớp Fully-connected.

Lớp này thực hiện nhiệm vụ phân loại dựa trên các đặc điểm được trích xuất thông qua các lớp trước đó và các bộ lọc khác nhau của chúng. Trong khi các lớp tích chập và gộp có xu hướng sử dụng các hàm ReLu, các lớp FC thường tận dụng hàm kích hoạt sigmoid hoặc softmax để phân loại đầu vào một cách thích hợp, tạo ra xác suất từ 0 đến 1.

Hàm sigmoid:

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

Hình 1.8. Công thức hàm kích hoạt sigmoid.



Hình 1.9. Biểu diễn của hàm sigmoid.

Hàm sigmoid thường nằm ở lớp cuối của mô hình với bài toán phân loại nhị phân (2 nhãn). Lúc này lớp cuối thường chỉ có một unit. Kết quả luôn nằm trong khoảng 0 đến 1, thuộc số thực. Qua đó, có thể phân biệt giữa 2 nhãn bằng cách xem kết quả trả về là nhỏ hơn hoặc lớn hơn 0,5.

Hàm softmax:

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$

Hình 1.10. Công thức hàm kích hoạt Softmax.

Kết quả của hàm này là một con số từ 0 đến 1 thể hiện xác suất của dữ liệu vào với nhãn thuộc vị trí này. Hàm softmax thường dùng trong những bài toán phân loại nhiều nhãn.

1.1.4. Điều chỉnh siêu tham số.

- Tính tương thích của tham số trong tầng tích chập, ta có thể tính độ dài của feature map theo một chiều bằng công thức:

$$O = \frac{I - F + P_{\text{start}} + P_{\text{end}}}{S} + 1$$

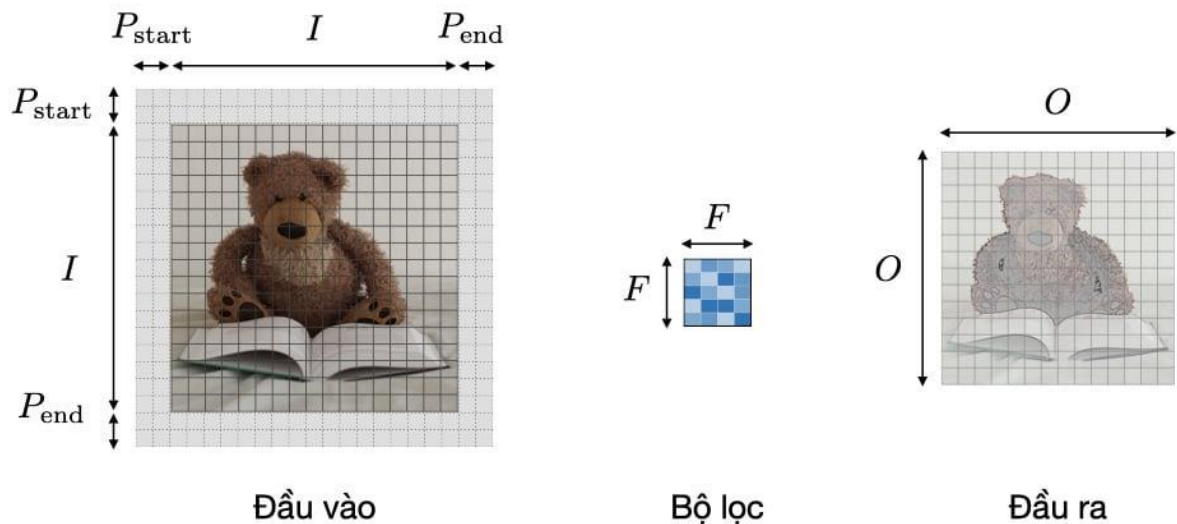
Trong đó: I là độ dài kích thước đầu vào,

F là độ dài của bộ lọc,

P là số lượng zero padding,

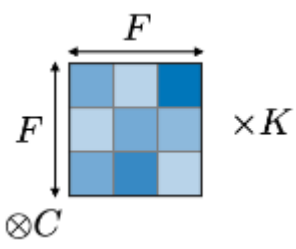
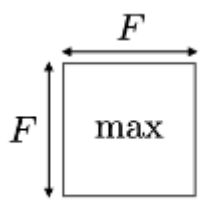
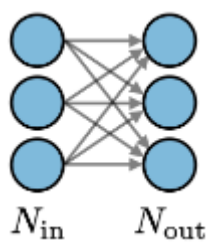
S là độ trượt,

O là độ dài của feature map.

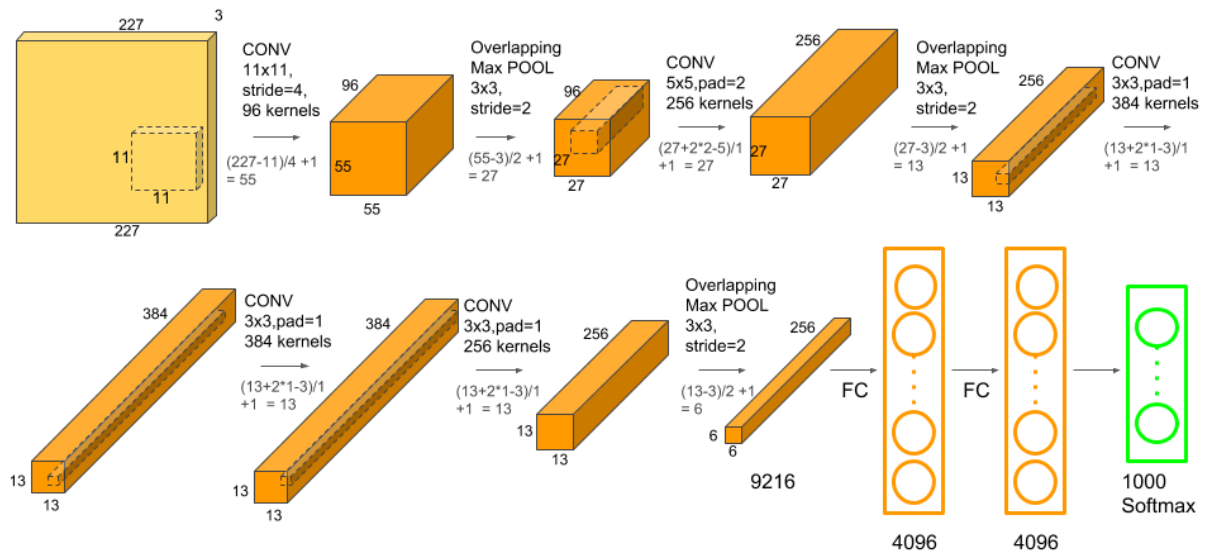


Hình 1.11. Biểu diễn cách tính độ dài feature map.

- Độ phức tạp của mô hình, để đánh giá độ phức tạp của một mô hình, cách hữu hiệu là xác định số tham số mà mô hình đó sẽ có. Trong một tầng của mạng neural tích chập, nó sẽ được tính toán như sau:

	CONV	POOL	FC
Minh hoạ			
Kích thước đầu vào	$I \times I \times C$	$I \times I \times C$	N_{in}
Kích thước đầu ra	$O \times O \times K$	$O \times O \times C$	N_{out}
Số lượng tham số	$(F \times F \times C + 1) \cdot K$	0	$(N_{in} + 1) \times N_{out}$
<p>Trong đó: I là độ dài kích thước đầu vào, F là độ dài của bộ lọc, C là số lượng kênh (channels), O là độ dài của feature map, N là số lượng neuron.</p>			

1.2. Đặc trưng chung của các mạng CNN.



Hình 1.12. Mạng Alexnet, một kiến trúc điển hình của CNN.

- Sử dụng tích chập: Tất cả các mạng CNN đều dựa trên nguyên lý tích chập để trích xuất đặc trưng từ hình ảnh.

- Kiến trúc phân tầng: Kiến trúc phân tầng giúp mạng CNN học được đặc trưng ở những cấp độ khác nhau, từ cấp độ low-level (bậc thấp) tới high-level (bậc cao). Theo đó, mức độ chi tiết của hình ảnh cũng tăng tiến dần từ các đặc trưng chung như các đường chéo, ngang, dọc rìa, cạnh tới những các đặc trưng chi tiết hơn giúp phân biệt vật thể như bánh xe, cánh cửa, mui xe (nếu vật thể là xe), tất cả các chi tiết đó được tổng hợp lại và ở layer tích chập cuối cùng ta thu được hình ảnh của một chiếc xe.

- Huấn luyện trên dữ liệu lớn: Khả năng học sâu của CNN được phát huy tốt nhất khi huấn luyện trên các bộ dữ liệu lớn.

- Giảm dần kích thước layer: Kiến trúc CNN thường giảm dần kích thước layer theo độ sâu để tối ưu hóa số lượng tham số.

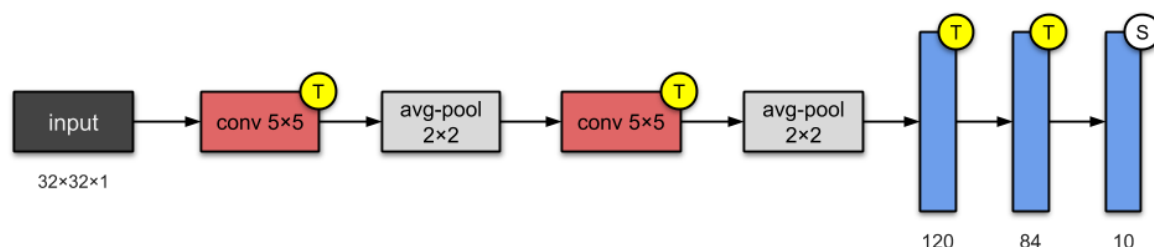
- Tăng độ sâu layer: Độ sâu của các layers tăng dần nhờ tăng số bộ lọc, thường là theo cấp số nhân. Độ sâu tăng sẽ giúp cho mạng CNN học được đa dạng các đặc trưng hơn. Ở những layer đầu tiên là những đặc trưng chung, chúng khá giống nhau về hình dạng, phương hướng, nên không cần quá nhiều bộ lọc để tạo ra chúng với số lượng lớn. Càng ở những layers sau đòi hỏi độ chi tiết cao hơn thì yêu cầu số lượng bộ lọc nhiều hơn để giúp phân biệt được nhiều chi tiết đặc trưng hơn.

- Sử dụng các fully connected layers để phân loại: Tích chập từ mạng CNN sẽ tạo ra những đặc trưng 2 chiều. Để sử dụng những đặc trưng này vào quá trình phân loại của mạng CNN thì chúng ta phải chuyển chúng thành đặc trưng 1 chiều bằng phương pháp

flatten và lan truyền thuận qua các fully connected layers. Sau mỗi một layer là một hàm kích hoạt phi tuyến nhằm gia tăng khả năng biểu diễn giúp cho kết quả phân loại tốt hơn.

1.3. Các mạng CNN tiêu biểu.

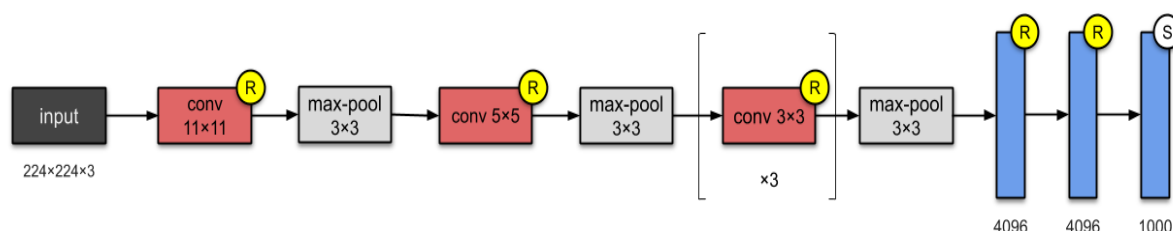
1.3.1. LeNet-5 (1998).



Hình 1.13. Kiến trúc LeNet [2].

LeNet-5, do Yann LeCun phát triển, là kiến trúc đầu tiên áp dụng mạng tích chập 2 chiều. Model ban đầu khá đơn giản và chỉ bao gồm 2 convolutional layers và 3 fully-connected layers. Mặc dù đơn giản nhưng nó có kết quả tốt hơn so với các thuật toán machine learning truyền thống khác trong phân loại chữ số viết tay như SVM, kNN.

1.3.2 AlexNet (2012).



Hình 1.14. Kiến trúc AlexNet [3].

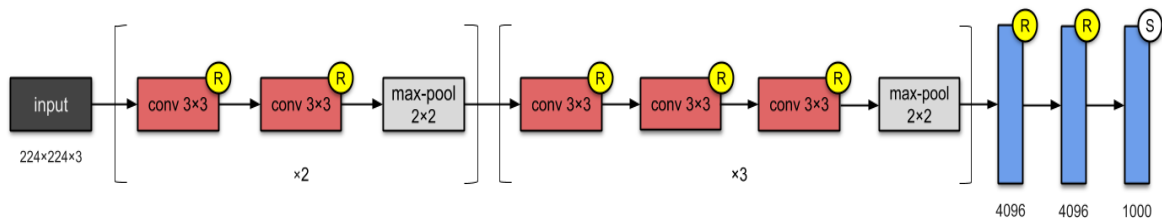
AlexNet là một mạng CNN nổi bật được giới thiệu vào năm 2012 bởi Alex Krizhevsky, giúp mạng này giành chiến thắng ấn tượng trong cuộc thi ImageNet. Mạng này dựa trên LeNet của Yann LeCun nhưng được cải tiến với các điểm chính sau:

- Tăng kích thước đầu vào và độ sâu của mạng: mở rộng kích thước của đầu vào và độ sâu của mạng để có thể xử lý các đặc trưng phức tạp hơn.
- Sử dụng các bộ lọc (kernel hoặc filter) với kích thước giảm dần: Điều này giúp mạng phù hợp với cả đặc trưng chung và đặc trưng riêng của dữ liệu.
- Local normalization: Giúp chuẩn hóa các layer, từ đó tăng tốc độ hội tụ trong quá trình huấn luyện.

- Activation function ReLU: Lần đầu tiên sử dụng hàm ReLU thay cho Sigmoid, giúp tăng tốc độ tính toán nhờ tính đơn giản của đạo hàm và vẫn đảm bảo tính phi tuyến.

- Dropout layer: Giảm số lượng liên kết neural, giúp kiểm soát overfitting và làm mô hình trở nên ít phức tạp hơn.

1.3.3. VGG-16 (2014).



Hình 1.15. Kiến trúc VGG-16 [4].

VGG-16 là một mạng CNN sâu, phát triển từ kiến trúc của AlexNet với những cải tiến quan trọng:

- Kiến trúc VGG-16 sâu hơn, có 13 layers tích chập 2 chiều (thay vì 5 như AlexNet) và 3 layers fully connected, làm tăng khả năng trích xuất đặc trưng của mạng.

- Khối tích chập (block): Lần đầu tiên xuất hiện trong VGG-16, các khối tích chập gồm các layers CNN giống nhau được lặp lại. Cách tổ chức này trở thành mẫu hình phổ biến cho các mạng CNN về sau.

- Giống như AlexNet, VGG-16 cũng sử dụng hàm kích hoạt ReLU, giúp tăng tốc độ tính toán và giữ nguyên tính phi tuyến.

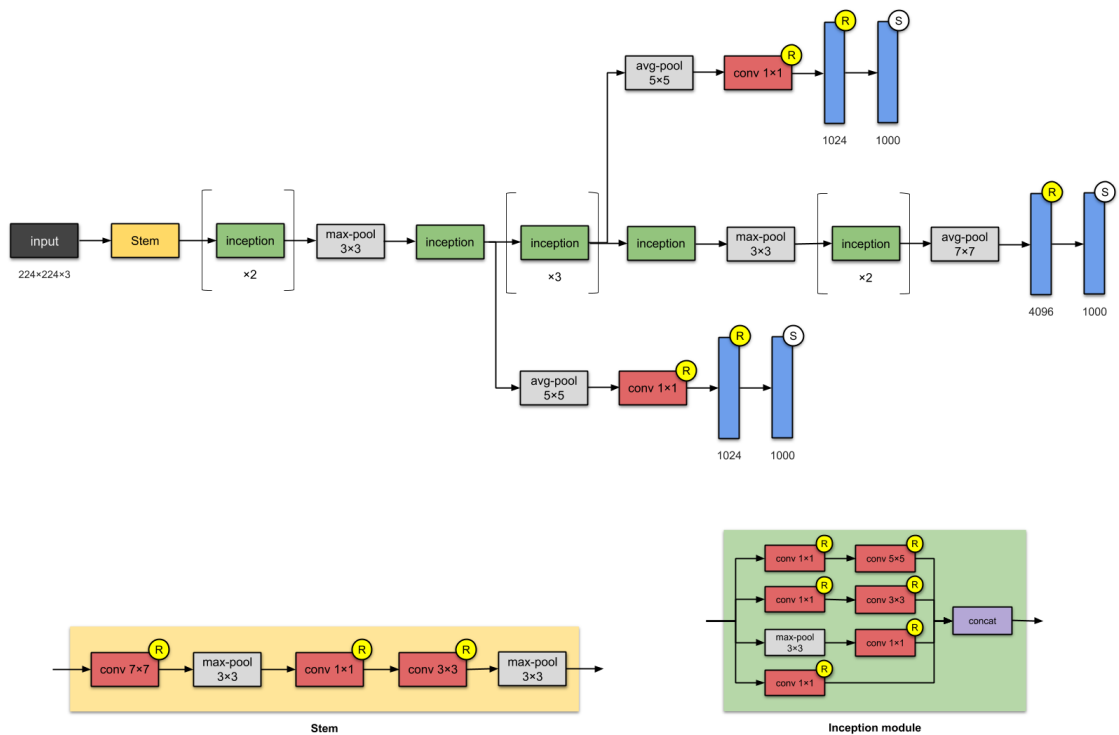
- Thay đổi thứ tự block: VGG-16 thay đổi cách xếp layers CNN và Max Pooling, sử dụng nhiều layers CNN liên tiếp trước khi áp dụng Max Pooling, giúp mạng trích lọc đặc trưng tốt hơn.

- Sử dụng bộ lọc nhỏ 3x3: VGG-16 chỉ sử dụng các bộ lọc kích thước nhỏ 3x3, thay vì nhiều kích thước bộ lọc như AlexNet, giúp giảm số lượng tham số và tăng hiệu quả tính toán.

Với 138 triệu tham số, VGG-16 là một trong những mạng có số lượng tham số lớn nhất.

Từ VGG-16, hình mẫu chung cho các mạng CNN trong các tác vụ học có giám sát về xử lý ảnh bắt đầu hình thành, với các mạng ngày càng sâu hơn và sử dụng các block [Conv2D*n + Max Pooling].

1.3.4. GoogleNet - Inception-V1 (2014).



Hình 1.16. Kiến trúc GoogleNet - Inception version 1 [5].

Inception-V1, do Christian Szegedy và đồng nghiệp phát triển, đã giành chiến thắng trong cuộc thi ImageNet năm 2015 với kiến trúc mạng giải quyết câu hỏi về kích thước kernel_size tối ưu trong mạng CNN. Các mạng trước đây sử dụng đa dạng kích thước bộ lọc, nhưng Inception-V1 kết hợp các bộ lọc với kích thước khác nhau (1x1, 3x3, 5x5) vào một block duy nhất.

Khối Inception:

- Khối Inception có 4 nhánh song song, mỗi nhánh áp dụng bộ lọc với kích thước khác nhau.

- Nhánh 1, 2, 4 sử dụng phép tích chập 1x1 để giảm số kênh và số lượng tham số. Với input 12x12x256, sau khi áp dụng bộ lọc 1x1 với 32 kênh, output có kích thước 12x12x32.

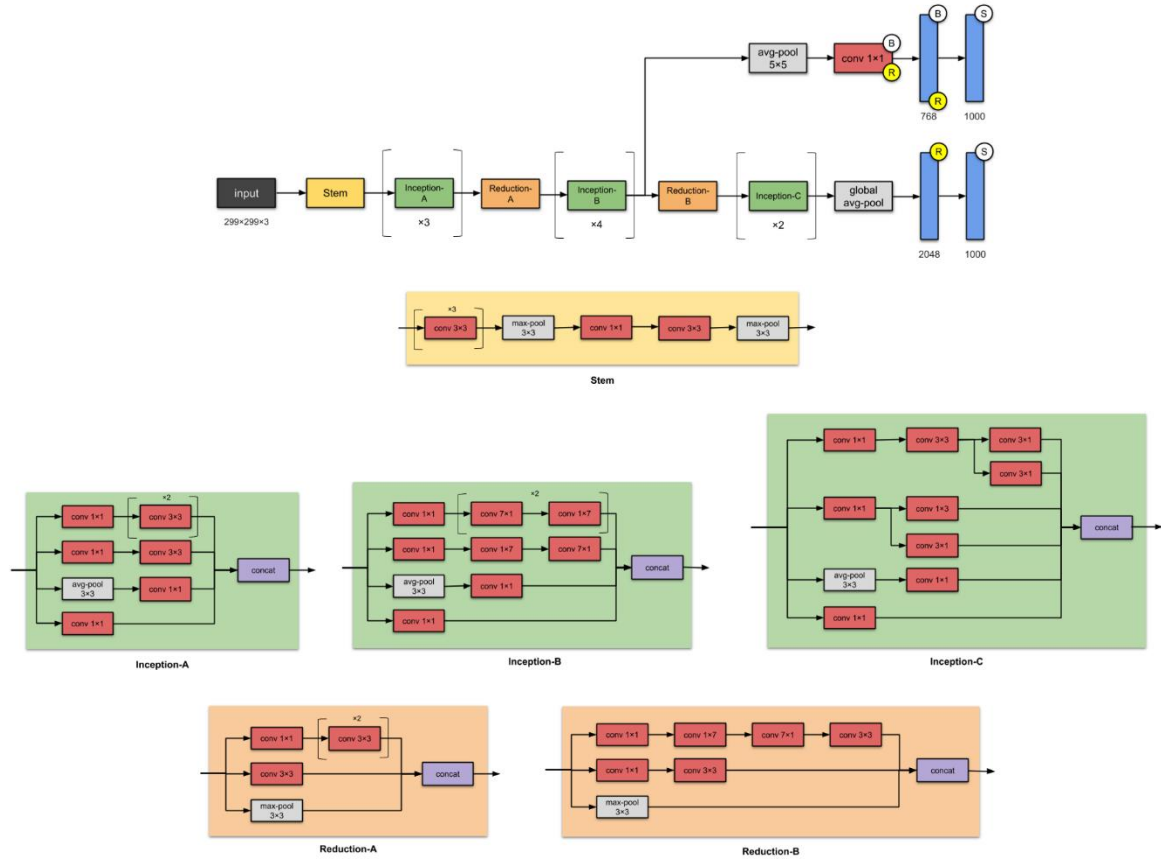
- Nhánh 3 sử dụng max-pooling 3x3 để giảm chiều dữ liệu, sau đó áp dụng bộ lọc 1x1 để thay đổi số kênh.

- Các nhánh được padding và stride sao cho output có cùng kích cỡ chiều dài và chiều rộng, sau đó concatenate kết quả theo kênh.

Khối Inception được lặp lại 7 lần trong Inception-V1. Toàn bộ mạng có 22 layers, nhiều hơn gần gấp đôi so với VGG-16. Inception-V1 có khoảng 5 triệu tham số, ít

hơn gần 27 lần so với VGG-16, nhờ sử dụng tích chập 1x1 giúp giảm đáng kể số lượng tham số cần thiết. Inception-V1 mang lại hiệu quả cao và thiết lập một bước tiến mới trong thiết kế kiến trúc mạng CNN.

1.3.5. GoogleNet - Inception-V3 (2015).

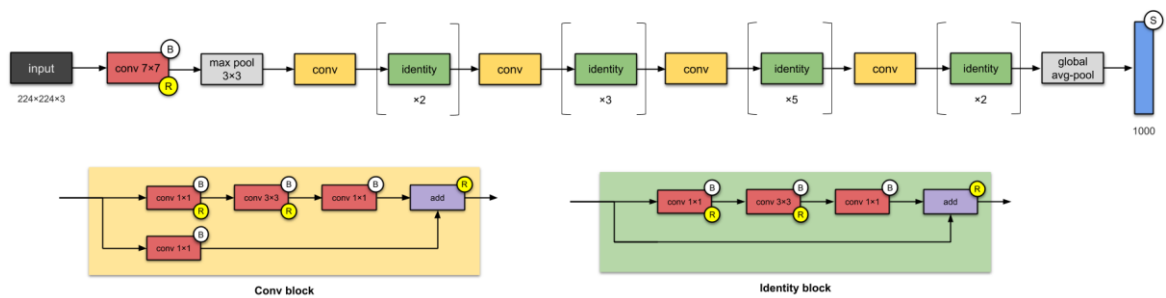


Hình 1.17. Kiến trúc GoogleNet - Inception version 3 [6].

Inception-V3 là kế thừa của Inception-V1 bao gồm 24 triệu tham số. Toàn bộ các layer tích chập của Inception-V3 được theo sau bởi một layer batch normalization và một ReLU activation. Batch normalization là kỹ thuật chuẩn hóa đầu vào theo từng minibatch tại mỗi layer theo phân phối chuẩn hóa $N(0,1)$, giúp cho quá trình huấn luyện thuật toán nhanh hơn.

Inception-V3 giải quyết được vấn đề thắt cổ chai (representational bottlenecks). Tức là kích thước của các layers không bị giảm một cách đột ngột. Đồng thời Inception-V3 có một cách tính toán hiệu quả hơn nhờ sử dụng phương pháp nhân tố (factorisation methods).

1.3.6. ResNet-50 (2015).



Hình 1.18. Kiến trúc ResNet [7].

ResNet là kiến trúc được sử dụng phổ biến nhất ở thời điểm hiện tại. ResNet cũng là kiến trúc sớm nhất áp dụng batch normalization. Mặc dù là một mạng rất sâu khi có số lượng layer lên tới 152 nhưng nhờ áp dụng những kỹ thuật đặc biệt mà ta sẽ tìm hiểu bên dưới nên kích thước của ResNet50 chỉ khoảng 26 triệu tham số. Kiến trúc với ít tham số nhưng hiệu quả của ResNet đã mang lại chiến thắng trong cuộc thi ImageNet năm 2015.

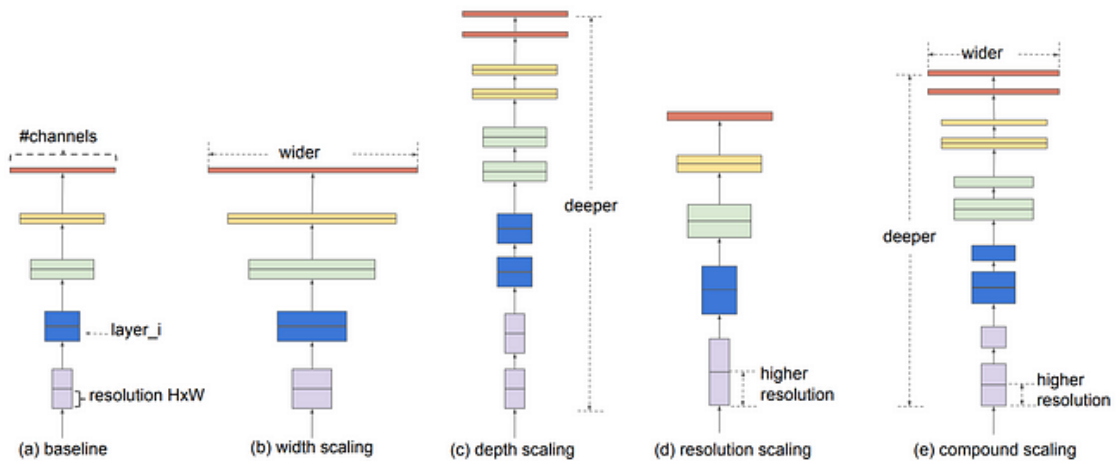
ResNet có khối tích chập (Convolutional Block, chính là Conv block trong hình) sử dụng bộ lọc kích thước 3 x 3 giống với của InceptionNet. Khối tích chập bao gồm 2 nhánh tích chập trong đó một nhánh áp dụng tích chập 1 x 1 trước khi cộng trực tiếp vào nhánh còn lại.

Khối xác định (Identity block) thì không áp dụng tích chập 1 x 1 mà cộng trực tiếp giá trị của nhánh đó vào nhánh còn lại.

1.4. EfficientNet.

1.4.1. Model scaling.

Model Scaling là quá trình mở rộng kích thước và khả năng của một mô hình học máy để tăng cường hiệu suất mà không làm tăng đáng kể chi phí tính toán. Có ba kích thước tỷ lệ của CNN: chiều rộng (width), chiều sâu (depth), và độ phân giải (resolution) của mô hình.



Hình 1.19. Tỷ lệ mô hình.

(a) là một ví dụ về mạng đường cơ sở; (b) - (d) là tỷ lệ thông thường chỉ tăng một chiều của chiều rộng, chiều sâu hoặc độ phân giải của mạng. (e) Phương pháp chia tỷ lệ hợp chất được đề xuất để chia tỷ lệ đồng nhất cả ba chiều với tỷ lệ cố định.

- Width scaling (Mở rộng theo chiều rộng): là việc tăng số lượng kênh trong mỗi lớp của mạng. Khi mở rộng chiều rộng, số lượng bộ lọc (filters) trong các lớp convolutional được tăng lên. Qua đó, mô hình học được nhiều đặc trưng hơn trong mỗi lớp nhưng có thể dẫn đến mô hình trở nên quá lớn và dễ bị overfitting.

- Depth scaling (Mở rộng chiều sâu): là việc tăng số lượng các lớp trong mạng, tức là tăng số lượng lớp convolutional hoặc fully connected layers. Điều này giúp mô hình có thể nắm bắt được nhiều thông tin hơn và học hỏi được các đặc trưng phức tạp, trừu tượng. Nhưng tăng chiều sâu quá mức có thể làm cho mô hình trở nên khó huấn luyện hơn do vấn đề vanishing gradient.

- Resolution scaling (Mở rộng độ phân giải): là việc tăng kích thước của đầu vào ảnh. Mở rộng độ phân giải là việc tăng kích thước ảnh đầu vào để mô hình có thể học được nhiều chi tiết hơn. Nhược điểm là cần lượng tài nguyên lớn để thực hiện tính toán và lưu trữ.

- Compound scaling: là phương pháp kết hợp việc mở rộng cả chiều sâu, chiều rộng và độ phân giải của mô hình một cách đồng đều dựa trên hệ số kép ϕ và theo nguyên tắc sau:

$$\begin{aligned}
&\text{depth: } d = \alpha^\phi \\
&\text{width: } w = \beta^\phi \\
&\text{resolution: } r = \gamma^\phi \\
&\text{s.t. } \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \\
&\alpha \geq 1, \beta \geq 1, \gamma \geq 1
\end{aligned}$$

Trong đó:

- + d là chiều sâu của mô hình (Depth).
- + w là chiều rộng của mô hình (Width).
- + r là độ phân giải của ảnh đầu vào (Resolution).
- + ϕ là hệ số do người dùng chỉ định kiểm soát số lượng tài nguyên có sẵn.
- + α, β, γ là các hệ số chỉ định cách gán các tài nguyên này cho độ sâu, chiều rộng và độ phân giải của mạng.

1.4.2. EfficientNet.

EfficientNet [8] là một họ mô hình mạng nơ-ron tích chập (CNN) được thiết kế để đạt hiệu suất cao trong việc phân loại hình ảnh. EfficientNet sử dụng phương pháp compound scaling để mở rộng mô hình một cách cân bằng giữa chiều rộng, chiều sâu và độ phân giải

EfficientNet-B0 là phiên bản cơ bản của họ mô hình EfficientNet, được tối ưu hóa bằng cách sử dụng kỹ thuật Neural Architecture Search (NAS) để tìm ra cấu trúc tối ưu nhất với các tham số α, β , và γ . Từ EfficientNet-B0, các phiên bản khác (EfficientNet-B1 đến EfficientNet-B7) được mở rộng dựa trên công thức Compound scaling.

Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels \hat{C}_i	#Layers \hat{L}_i
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1



Hình 1.20. Mạng cơ sở EfficientNet-B0.

Từ mạng cơ sở EfficientNet-B0 áp dụng phương pháp compound scaling để tiến hành mở rộng mô hình trong hai bước:

- Bước 1: Cố định $\phi = 1$, giả sử rằng có gấp đôi tài nguyên hơn và thực hiện tìm kiếm trên lưới nhỏ cho α , β và γ . Đối với mạng cơ sở B0, các giá trị tối ưu là $\alpha = 1,2$, $\beta = 1,1$ và $\gamma = 1,15$ sao cho $\alpha * \beta^2 * \gamma^2 \approx 2$

- Bước 2: Gán cố định α , β và γ làm các hằng số (với các giá trị được tìm thấy ở bước trên) và thử nghiệm với các giá trị ϕ khác nhau để thu được EfficientNet-B1 đến B7.

1.5. Một số kỹ thuật tăng cường dữ liệu(Data augmentation).

Data Augmentation (tăng cường dữ liệu) là một kỹ thuật trong học sâu nhằm làm tăng số lượng và sự đa dạng của các mẫu dữ liệu bằng cách tạo ra các mẫu dữ liệu khác nhau bằng các kỹ thuật như xoay, dịch chuyển, lật, phóng to hoặc thu nhỏ dữ liệu hiện có. Kỹ thuật này giúp tập dữ liệu trở nên đa dạng hơn cải thiện việc huấn luyện mô hình Neural Network. Sử dụng Data Augmentation mang đến một số ưu điểm như:

- Tăng độ chính xác của mô hình: Bằng cách tạo ra nhiều biến thể của dữ liệu huấn luyện, mô hình có thể học được nhiều đặc trưng hơn và đạt được độ chính xác tốt khi dự đoán.

- Giảm hiện tượng overfitting: Overfitting xảy ra khi mô hình học quá tốt các đặc trưng của tập dữ liệu huấn luyện và đạt độ chính xác thấp trên tập dữ liệu kiểm tra. Data augmentation giúp giảm hiện tượng này bằng cách cung cấp nhiều biến thể của dữ liệu, giúp mô hình không chỉ học thuộc các mẫu cụ thể mà học cách nhận diện các đặc trưng chung.

- Tăng cường dữ liệu khi dữ liệu hạn chế: Trong nhiều trường hợp, việc thu thập dữ liệu huấn luyện có thể tốn kém và khó khăn. Data augmentation giúp tạo ra nhiều mẫu dữ liệu hơn từ một tập dữ liệu nhỏ.

Ví dụ về các kỹ thuật Data Augmentation:

Code đọc ảnh gốc từ file vào:

```
import random
import numpy as np
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img, img_to_array, array_to_img
import matplotlib.pyplot as plt

image_path = '/content/drive/MyDrive/CarsDataset/train/Toyota Innova/990.jpg'
image = load_img(image_path)

fig, ax = plt.subplots()
ax.imshow(image)
plt.show()
```



Hình 1.21. Hình ảnh gốc của mẫu.

1.5.1 Kỹ thuật xoay ảnh.

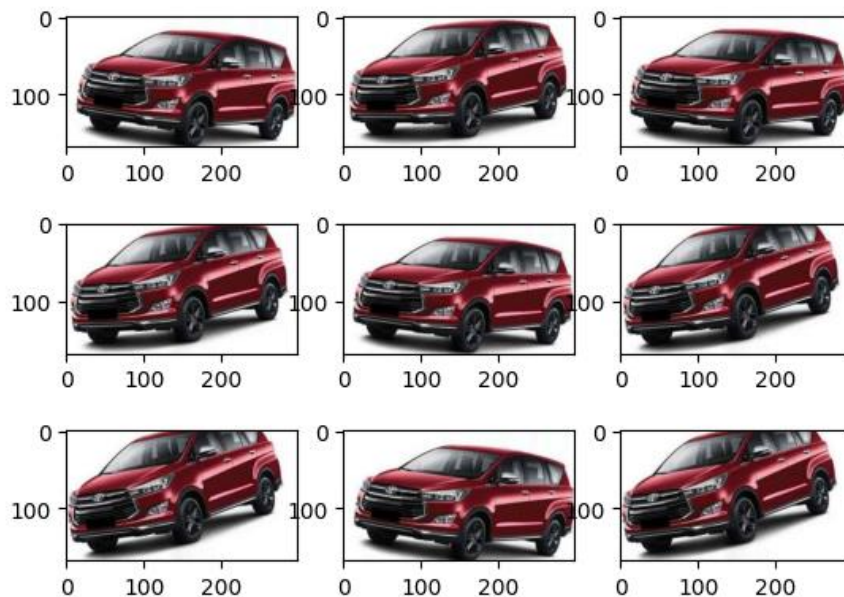
Code thực hiện xoay ảnh:

```
image_array = img_to_array(image)
datagen = ImageDataGenerator(rotation_range=10)
plt_rows=3
plt_cols=3
fig, axes = plt.subplots(plt_rows, plt_cols)
for i in range(plt_rows):
    for j in range(plt_cols):
        s_img = datagen.random_transform(image_array, random.randrange(0, 33333333, 4)).astype('uint8')
        s_img = array_to_img(s_img)
        axes[i, j].imshow( s_img )

plt.show()
```

Sử dụng ImageDataGenerator(rotation_range=10) để thực hiện tạo mẫu xoay ảnh ngẫu nhiên từ khoảng -10 độ đến 10 độ.

Ta có kết quả như sau:

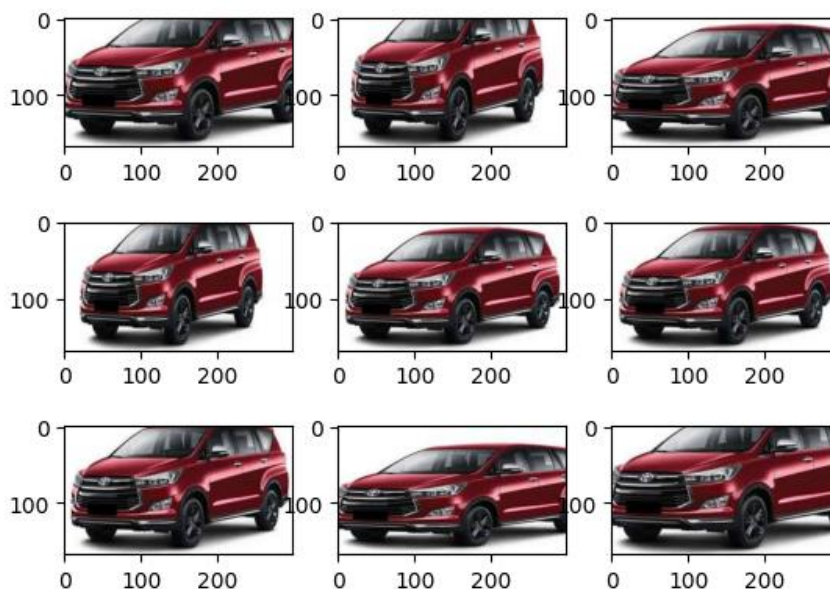


Hình 1.22. Các mẫu được tạo ra từ cách xoay ảnh.

1.5.2 Kỹ thuật phóng to, thu nhỏ ảnh.

Sử dụng `ImageDataGenerator(zoom_range=0.2)` để phóng to hoặc thu nhỏ ngẫu nhiên hình ảnh trong khoảng từ 80% đến 120% của kích thước gốc của ảnh.

Kết quả có được:

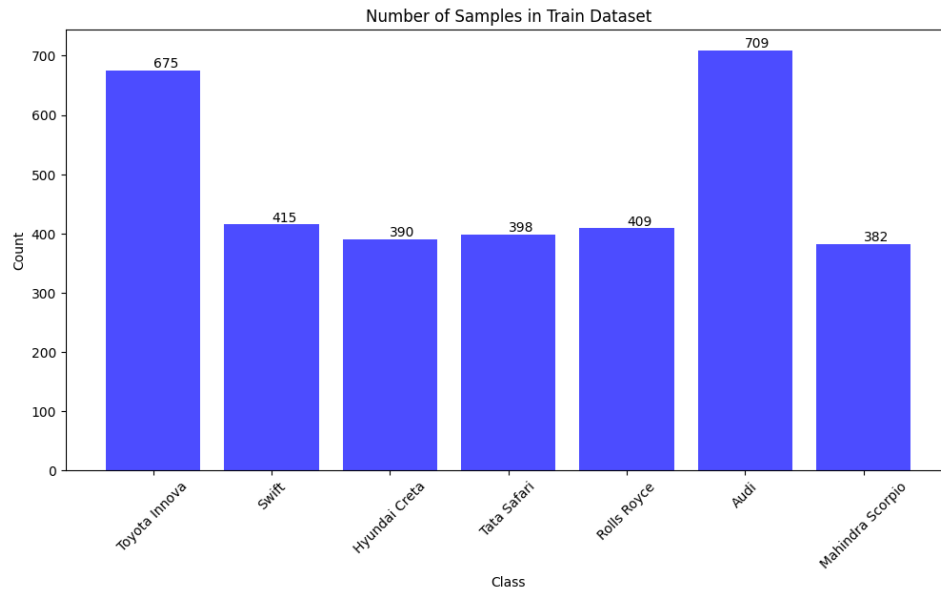


Hình 1.23. Các mẫu được tạo ra từ cách xoay ảnh.

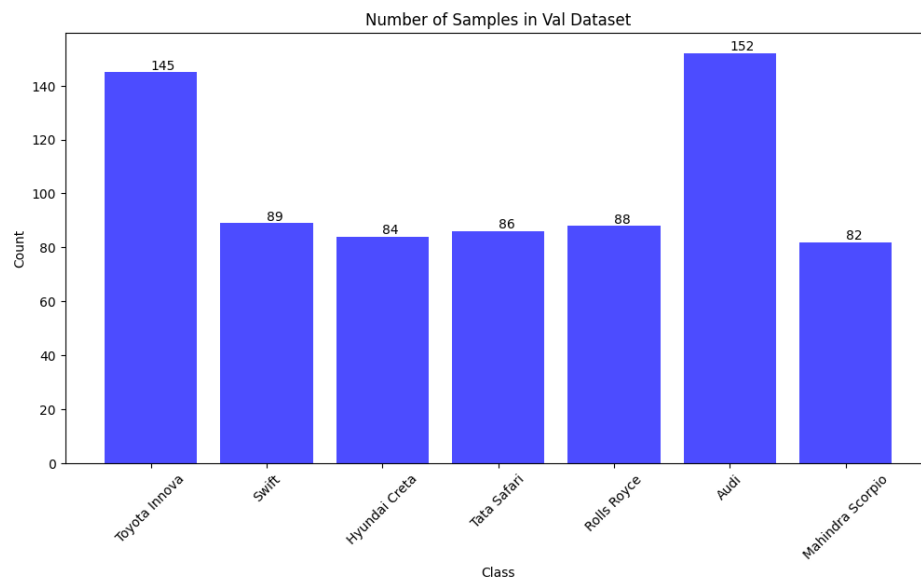
CHƯƠNG 2: XÂY DỰNG MÔ HÌNH.

2.1. Tập dữ liệu.

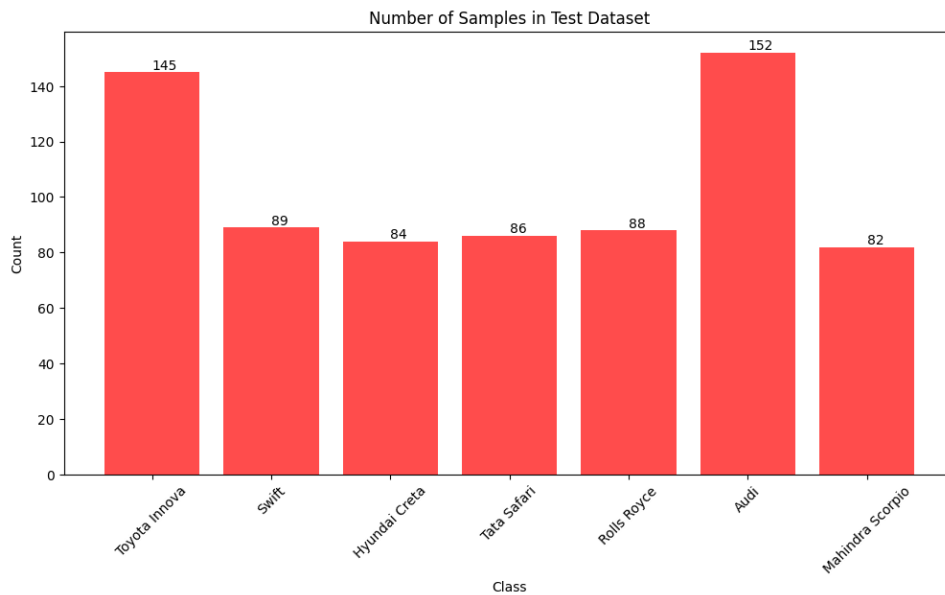
- Car Images Dataset [9] : bao gồm 4830 hình ảnh của nhiều loại ô tô khác nhau, được chia làm ba thư mục train, validation, test theo tỉ lệ 70%, 15%, 15% chứa hình ảnh ô tô của bảy hãng: Tata Safari, Swift, Toyota Innova, Rolls Royce, Audi, Mahindra Scorpio, Hyundai Creta.



Hình 2.1. Biểu đồ số lượng mẫu của tập dữ liệu huấn luyện (train).



Hình 2.2. Biểu đồ số lượng mẫu của tập dữ liệu kiểm tra (validation).



Hình 2.3. Biểu đồ số lượng mẫu của tập dữ liệu kiểm tra (test).

2.2. Tăng cường dữ liệu.

```
IMAGE_SIZE = 224

train_datagen = ImageDataGenerator(
    rescale=1./255,
    horizontal_flip = True,
    rotation_range=20,
    width_shift_range=0.1,
    height_shift_range=0.1,
    fill_mode='nearest',
)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(IMAGE_SIZE,IMAGE_SIZE),
    class_mode="categorical",
    color_mode= 'rgb',
    shuffle=True,
    batch_size=16
)

datagen = ImageDataGenerator(
    rescale=1./255,
    horizontal_flip = False)

val_generator = datagen.flow_from_directory(
    val_dir,
    target_size=(IMAGE_SIZE,IMAGE_SIZE),
    class_mode="categorical",
    color_mode= 'rgb',
    shuffle = False,
)

test_generator = datagen.flow_from_directory(
    test_dir,
    target_size=(IMAGE_SIZE,IMAGE_SIZE),
    class_mode="categorical",
    color_mode= 'rgb',
    shuffle = False,
)
```

Các tham số được truyền vào ImageDataGenerator dùng để thực hiện các phép biến đổi dữ liệu:

- `rescale=1./255`: Chia tất cả các giá trị pixel của hình ảnh cho 255 để chuẩn hóa chúng vào khoảng $[0, 1]$ giúp mô hình học tốt hơn vì các giá trị đầu vào được chuẩn hóa.
- `horizontal_flip=True`: Lật ngẫu nhiên các hình ảnh theo chiều ngang (trái-phải).
- `rotation_range=20`: Xoay ngẫu nhiên các hình ảnh trong khoảng từ -20 đến 20 độ.
- `width_shift_range=0.1` và `height_shift_range=0.1`: Dịch chuyển ngẫu nhiên hình ảnh theo chiều ngang và chiều dọc trong khoảng $\pm 10\%$ kích thước chiều rộng và chiều cao của hình ảnh.
- `fill_mode='nearest'`: Khi hình ảnh được dịch chuyển hoặc xoay, các pixel ngoài biên có thể không có dữ liệu, `fill_mode='nearest'` sẽ sử dụng các giá trị pixel gần nhất để điền vào các vùng trống.
- `flow_from_directory` là một phương pháp của `ImageDataGenerator` để tải dữ liệu hình ảnh từ một thư mục chứa các thư mục con, mỗi thư mục con chứa các hình ảnh thuộc cùng một lớp (class).
- `target_size=(IMAGE_SIZE, IMAGE_SIZE)`: Thay đổi kích thước tất cả các hình ảnh trong tập dữ liệu thành kích thước 224×224 pixels.
- `class_mode="categorical"`: Chỉ định rằng các nhãn là dạng phân loại nhiều lớp (multi-class), tức là mô hình sẽ sử dụng hàm softmax để phân loại các hình ảnh vào một trong nhiều lớp.
- `color_mode='rgb'`: Hình ảnh được tải lên dưới dạng ảnh màu RGB (3 kênh màu: Đỏ, Lục, Lam).
- `shuffle=True`: Xáo trộn ngẫu nhiên các hình ảnh sau mỗi epoch để mô hình không học được sự tuần tự nào trong dữ liệu.
- `batch_size=16`: Số lượng hình ảnh trong mỗi batch sẽ là 16. Mỗi lần generator sinh ra một batch, nó sẽ chứa 16 hình ảnh đã được biến đổi và chuẩn hóa.

Tập dữ liệu validation và test sử dụng `horizontal_flip=False` và `shuffle=False` vì hai tập dữ liệu này sử dụng để kiểm tra không cần thực hiện xoay hình và trộn ngẫu nhiên.

2.3. Mô hình Convolutional Neural Network.

Khởi tạo mô hình:

```
def create_cnn(input_shape, num_classes):
    model = Sequential()

    model.add(Convolution2D(32, (5, 5), input_shape=input_shape, activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Convolution2D(64, (5, 5), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Convolution2D(128, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Convolution2D(256, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Flatten())

    model.add(Dense(units=128, activation='relu'))
    model.add(Dropout(0.40))
    model.add(Dense(units=32, activation='relu'))
    model.add(Dense(units=num_classes, activation='softmax'))

    return model
```

- Convolution2D(filters, kernel_size, input_shape, activation):

+ filters: Số lượng bộ lọc (filters) trong lớp tích chập. Mỗi bộ lọc sẽ học cách phát hiện các đặc trưng khác nhau trong hình ảnh.

+ kernel_size: Kích thước của cửa sổ bộ lọc (ví dụ: (5, 5) hoặc (3, 3)).

+ input_shape: Kích thước đầu vào của hình ảnh, thường là (chiều rộng, chiều cao, số kênh màu). Được xác định ở lớp tích chập đầu tiên.

+ activation='relu': Hàm kích hoạt ReLU (Rectified Linear Unit) được sử dụng để thêm tính phi tuyến cho mô hình, giúp mô hình học được các đặc trưng phức tạp hơn.

- MaxPooling2D(pool_size=(2, 2)):

+ Lớp gộp cực đại (Max Pooling) giúp giảm kích thước của dữ liệu đầu ra từ lớp tích chập, đồng thời giữ lại các đặc trưng quan trọng. pool_size=(2, 2) nghĩa là sẽ lấy giá trị lớn nhất từ mỗi khối 2x2.

-Flatten(): Lớp này chuyển đổi dữ liệu từ dạng 2D (hoặc 3D) thành một vector 1D để có thể đưa vào các lớp hoàn toàn kết nối (fully connected).

- Dense(units, activation):

+ units: Số lượng nơ-ron trong lớp.

+ activation: Hàm kích hoạt được sử dụng trong mỗi lớp. Với lớp cuối cùng, softmax được sử dụng để chuyển đổi đầu ra thành xác suất cho mỗi lớp trong bài toán phân loại.

- Dropout(0.40): Lớp Dropout với tỷ lệ bỏ nơ-ron là 40%. Dropout giúp ngăn chặn overfitting bằng cách ngẫu nhiên loại bỏ một số nơ-ron trong quá trình huấn luyện.

Cấu hình và huấn luyện mô hình:

```
model.compile(Adamax(learning_rate= 0.001), loss= 'categorical_crossentropy', metrics= ['accuracy'])
callbacks = keras.callbacks.EarlyStopping(monitor="val_accuracy",patience= 10,verbose=1,min_delta = 0.01,restore_best_weights=True)
history = model.fit(
    train_generator,
    validation_data=val_generator,
    epochs=epochs,
    callbacks=[callbacks]
)
```

- Adamax: là một biến thể của thuật toán Adam, Adamax được thiết kế để ổn định hơn và có thể làm việc tốt với các tập dữ liệu lớn hoặc gradient lớn.

- learning_rate=0.001: Tốc độ học (learning rate) được đặt là 0.001.

- categorical_crossentropy: là hàm mất mát được sử dụng phổ biến trong các bài toán phân loại nhiều lớp (multi-class classification).

- accuracy: là phép đo được sử dụng để đánh giá hiệu suất của mô hình. Độ chính xác (accuracy) là tỷ lệ phần trăm các dự đoán đúng trên tổng số dự đoán.

- EarlyStopping: là một kỹ thuật dùng để dừng quá trình huấn luyện sớm nếu mô hình không còn cải thiện. Điều này giúp tránh overfitting (mô hình quá khớp với dữ liệu huấn luyện) và tiết kiệm thời gian huấn luyện.

- monitor="val_accuracy": Xác định giá trị cần theo dõi trong quá trình huấn luyện. Ở đây, callback sẽ theo dõi độ chính xác trên tập dữ liệu xác thực (val_accuracy).

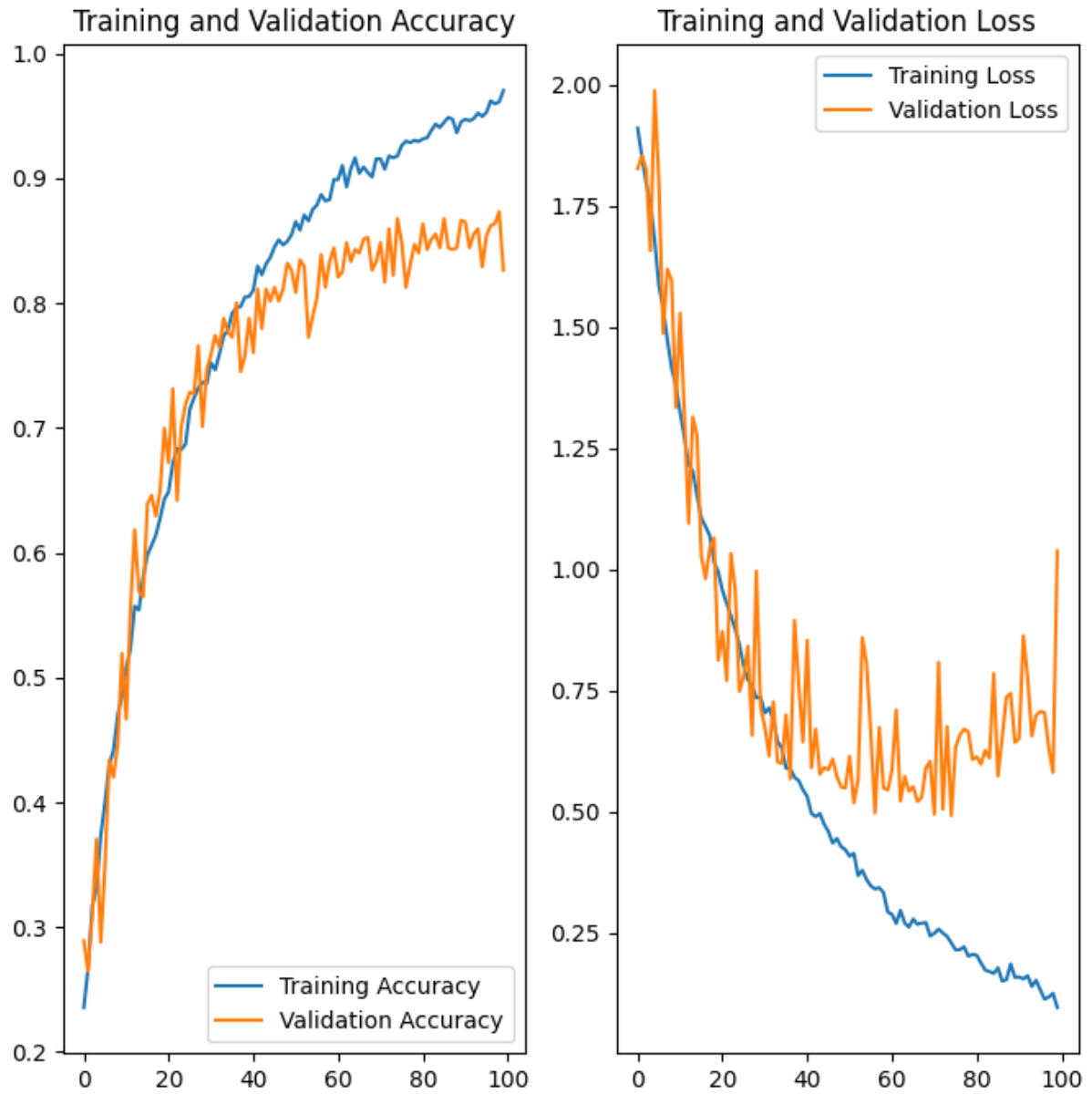
- patience=10: Số epoch mô hình sẽ tiếp tục huấn luyện nếu giá trị theo dõi không cải thiện. Nếu sau 10 epoch liên tiếp mà val_accuracy không cải thiện (với mức tối thiểu min_delta), quá trình huấn luyện sẽ dừng lại.

- verbose=1: In thông tin chi tiết về quá trình dừng sớm. Mức verbose=1 sẽ hiển thị thông tin mỗi khi huấn luyện dừng lại sớm.

- min_delta=0.01: xác định mức cải thiện tối thiểu cần đạt được để được coi là một cải thiện. Nếu val_accuracy không cải thiện ít nhất 0.01 trong patience số epoch, quá trình huấn luyện sẽ dừng.

- restore_best_weights=True: Khi dừng huấn luyện sớm, mô hình sẽ khôi phục các trọng số từ epoch có độ chính xác tốt nhất trên tập dữ liệu xác thực. Điều này đảm bảo rằng mô hình tốt nhất được lưu lại, ngay cả khi nó đã bị overfitting trong các epoch cuối.

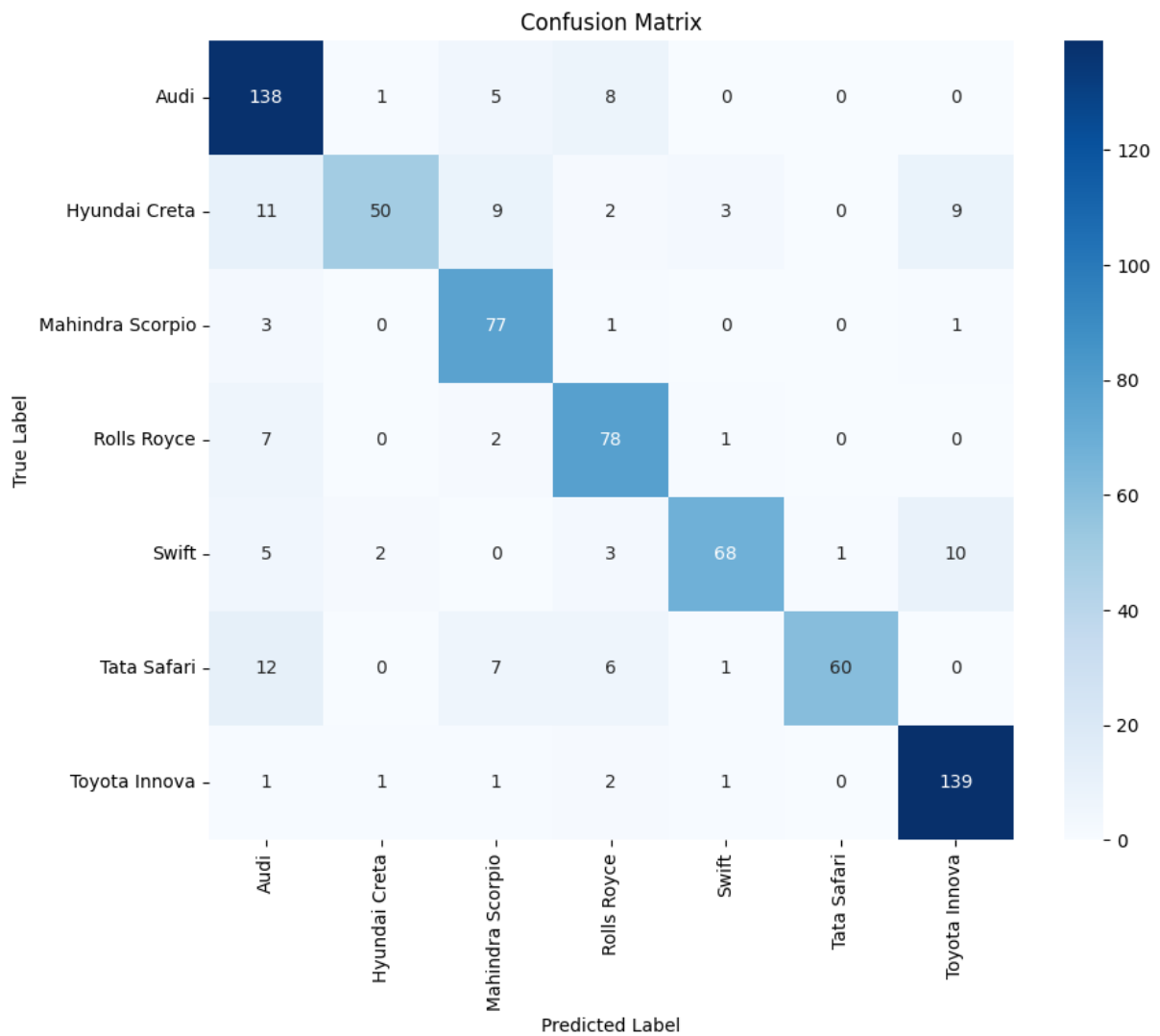
- Biểu đồ thể hiện độ chính xác và độ lệch sau 100 lần huấn luyện.



Thực hiện kiểm thử mô hình trên tập dữ liệu test đạt độ chính xác 84%

```
scores = model.evaluate(test_generator)
print(f"Test Accuracy: {scores[1]*100}")
```

23/23 ————— 216s 10s/step - accuracy: 0.8480 - loss: 0.8232
Test Accuracy: 84.02203917503357



Hình 2.4. Ma trận thể hiện lỗi của mô hình.

2.4. Mô hình CNN với EfficientNetB3 làm base model.

Khởi tạo mô hình:

```
def create_cnn(input_shape, num_classes):  
    base_model = tf.keras.applications.EfficientNetB3(include_top=False, weights="imagenet", input_shape=input_shape, pooling="max")  
    base_model.trainable = True  
    x = base_model.output  
    x = BatchNormalization()(x)  
    x = Dense(512, activation="relu")(x)  
    x = Dropout(0.3)(x)  
    outputs = Dense(num_classes, activation="softmax")(x)  
    model = Model(inputs=base_model.input, outputs=outputs)  
    return model
```

- Sử dụng EfficientNetB3 làm base model:

+ include_top=False: Loại bỏ phần đầu ra gốc của EfficientNetB3, chỉ giữ lại phần thân của mô hình (phần chứa các lớp convolutional để trích xuất đặc trưng).

+ weights="imagenet": Sử dụng các trọng số đã được huấn luyện trước trên tập dữ liệu ImageNet. Điều này giúp mô hình bắt đầu từ một trạng thái đã học tốt về các đặc trưng chung của hình ảnh.

+ input_shape=input_shape: Xác định kích thước đầu vào của hình ảnh (theo tham số input_shape được truyền vào hàm).

+ pooling="max": Áp dụng Max Pooling lên các đặc trưng đầu ra của EfficientNetB3, giúp giảm kích thước đầu ra xuống một vector có kích thước bằng số kênh của tensor cuối cùng trong base model.

- base_model.trainable = True: Cho phép các lớp trong EfficientNetB3 có thể được tinh chỉnh (fine-tuned) trong quá trình huấn luyện. Qua đó các trọng số của EfficientNetB3 sẽ được cập nhật khi huấn luyện, giúp mô hình học tốt hơn cho tác vụ cụ thể.

- x = base_model.output: Lấy đầu ra từ base model.

- BatchNormalization(): Thêm một lớp Batch Normalization để chuẩn hóa đầu ra, giúp tăng tốc độ huấn luyện và ổn định hơn.

- Dense(512, activation="relu"): Thêm một lớp Dense (fully connected) với 512 đơn vị (units) và hàm kích hoạt ReLU. Lớp này thêm khả năng phi tuyến tính và học thêm các đặc trưng từ đầu ra của EfficientNetB3.

- Dropout(0.3): Thêm một lớp Dropout với tỷ lệ 0.3 để ngẫu nhiên loại bỏ 30% các đơn vị kết nối trong lớp trước đó trong mỗi bước huấn luyện, giúp giảm thiểu hiện tượng overfitting.

- outputs = Dense(num_classes, activation="softmax"): Lớp Dense cuối cùng với số đơn vị bằng số lớp (num_classes), sử dụng hàm kích hoạt Softmax để dự đoán xác suất của từng lớp.

Cấu hình và huấn luyện mô hình:

```
model.compile(Adamax(learning_rate= 0.001), loss= 'categorical_crossentropy', metrics= ['accuracy'])
callbacks = keras.callbacks.EarlyStopping(monitor="val_accuracy",patience = 5,verbose=1,min_delta = 0.01,restore_best_weights=True)
history = model.fit(
    train_generator,
    validation_data=val_generator,
    epochs=epochs,
    callbacks=[callbacks]
)
```

- Adamax: là một biến thể của thuật toán Adam, Adamax được thiết kế để ổn định hơn và có thể làm việc tốt với các tập dữ liệu lớn hoặc gradient lớn.
- learning_rate=0.001: Tốc độ học (learning rate) được đặt là 0.001.
- categorical_crossentropy: là hàm mất mát được sử dụng phổ biến trong các bài toán phân loại nhiều lớp (multi-class classification).
- accuracy: là phép đo được sử dụng để đánh giá hiệu suất của mô hình. Độ chính xác (accuracy) là tỷ lệ phần trăm các dự đoán đúng trên tổng số dự đoán.
- EarlyStopping: là một kỹ thuật dùng để dừng quá trình huấn luyện sớm nếu mô hình không còn cải thiện. Điều này giúp tránh overfitting (mô hình quá khớp với dữ liệu huấn luyện) và tiết kiệm thời gian huấn luyện.
- monitor="val_accuracy": Xác định giá trị cần theo dõi trong quá trình huấn luyện. Ở đây, callback sẽ theo dõi độ chính xác trên tập dữ liệu xác thực (val_accuracy).
- patience=5: Số epoch mô hình sẽ tiếp tục huấn luyện nếu giá trị theo dõi không cải thiện. Nếu sau 5 epoch liên tiếp mà val_accuracy không cải thiện (với mức tối thiểu min_delta), quá trình huấn luyện sẽ dừng lại.
- verbose=1: In thông tin chi tiết về quá trình dừng sớm. Mức verbose=1 sẽ hiển thị thông tin mỗi khi huấn luyện dừng lại sớm.
- min_delta=0.01: xác định mức cải thiện tối thiểu cần đạt được để được coi là một cải thiện. Nếu val_accuracy không cải thiện ít nhất 0.01 trong patience số epoch, quá trình huấn luyện sẽ dừng.
- restore_best_weights=True: Khi dừng huấn luyện sớm, mô hình sẽ khôi phục các trọng số từ epoch có độ chính xác tốt nhất trên tập dữ liệu xác thực. Điều này đảm bảo rằng mô hình tốt nhất được lưu lại, ngay cả khi nó đã bị overfitting trong các epoch cuối.

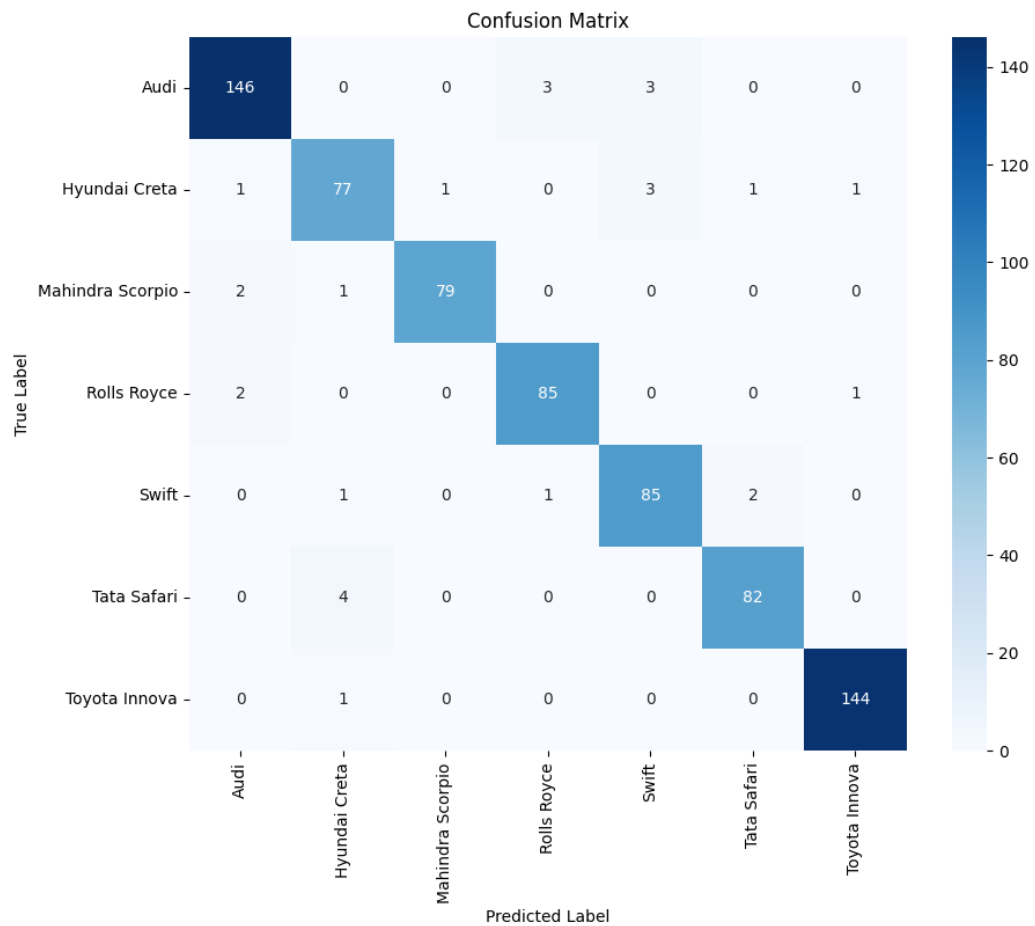
- Biểu đồ thể hiện độ chính xác và độ lệch sau 14 lần huấn luyện.



Thực hiện kiểm thử mô hình trên tập dữ liệu test đạt độ chính xác 96%

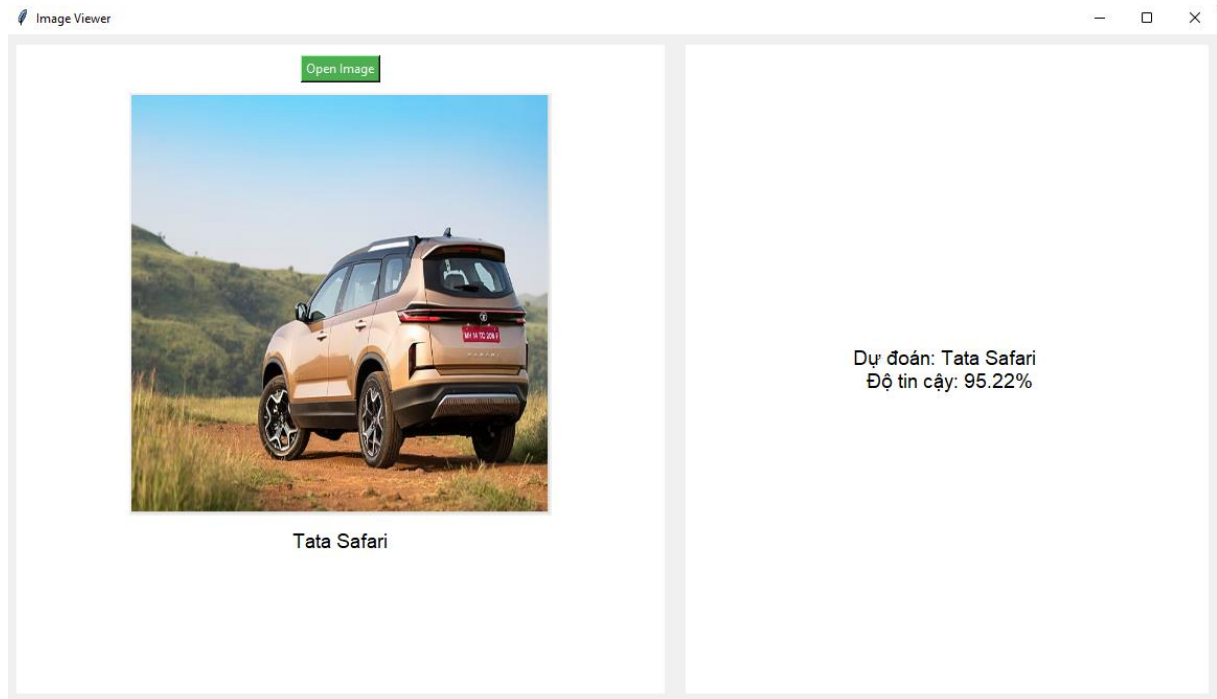
```
scores = model.evaluate(test_generator)
print(f"Test Accuracy: {scores[1]*100}")
```

23/23 ————— 346s 16s/step - accuracy: 0.9600 - loss: 0.1463
Test Accuracy: 96.14325165748596



Hình 2.5. Ma trận thể hiện lỗi của mô hình.

2.5. Xây dựng chương trình demo.



Hình 2.6. Giao diện demo model.

- Chương trình cho phép tải hình ảnh vào và thực hiện dự đoán ô tô thuộc bảy loại (Audi, Hyundai Creta, Mahindra Scorpio, Rolls Royce, Maruti Suzuki Swift, Tata Safari và Toyota Innova), sau đó hiển thị loại ô tô gốc của hình ảnh, kết quả dự đoán từ model và độ tin cậy của kết quả dự đoán.

CHƯƠNG 3: KẾT LUẬN.

3.1. Kết quả thực nghiệm.

- Tìm hiểu cách CNN hoạt động và các mạng CNN tiêu biểu.
- Huấn luyện được mô hình CNN và mô hình CNN với EfficientNetB3 làm base model có hiệu suất phân loại cao.
- Sử dụng mô hình để dự đoán các ô tô thuộc bảy loại (Audi, Hyundai Creta, Mahindra Scorpio, Rolls Royce, Maruti Suzuki Swift, Tata Safari và Toyota Innova) đạt kết quả chính xác với độ tin cậy cao.
- Tập dữ liệu chưa đủ lớn và đa dạng dẫn tới mô hình hoạt động kém trong các tình huống thực tế.

3.2. Hướng phát triển.

- Thử nghiệm mô hình trên dữ liệu thực tế qua camera.
- Đánh giá mô hình với bộ dữ liệu khác cùng loại nhãn.
- Xây dựng ứng dụng để áp dụng mô hình vào việc sử dụng.

DANH MỤC TÀI LIỆU THAM KHẢO

- [1] H. Nguyễn, "VIETNIX," CÔNG TY CỔ PHẦN GIẢI PHÁP VÀ CÔNG NGHỆ VIETNIX, 18 May 2024. [Online]. Available: <https://vietnix.vn/cnn-la-gi/>. [Accessed 20 May 2024].
- [2] Y. Bengio, "A NN/HMM Hybrid for on-line HandWriting Recognition.," *Neural Computation*, p. 5, September 1995.
- [3] A. Krizhevsky, "ImageNet Classification with Deep Convolutional," in *Proceedings of the 25th International Conference on Neural Information Processing Systems (NeurIPS)*, Lake Tahoe, 2012.
- [4] A. Z. Karen Simonyan, "arXiv," Cornell University, 04 September 2014. [Online]. Available: <https://arxiv.org/pdf/1409.1556>. [Accessed 20 June 2024].
- [5] W. L. Y. J. P. S. S. R. D. A. D. E. V. V. A. R. G. U. o. M. U. o. N. C. Christian Szegedy, "arXiv," 17 September 2014. [Online]. Available: <https://arxiv.org/pdf/1409.4842>. [Accessed 25 June 2024].
- [6] V. V. S. I. J. S. Z. W. Christian Szegedy, "arXiv," 2 December 2015. [Online]. Available: <https://arxiv.org/pdf/1512.00567>. [Accessed 26 June 2024].
- [7] X. Z. S. R. J. S. Kaiming He, "arXiv," 10 December 2015. [Online]. Available: <https://arxiv.org/pdf/1512.03385>. [Accessed 27 June 2024].
- [8] S. S. E. a. Q. V. L. P. S. Mingxing Tan, "Google Research," Google, 29 May 2019. [Online]. Available: <https://research.google/blog/efficientnet-improving-accuracy-and-efficiency-through-automl-and-model-scaling/>. [Accessed 9 August 2024].
- [9] K. KUMAR, "Kaggle," Kaggle, 05 April 2022. [Online]. Available: <https://www.kaggle.com/datasets/kshitij192/cars-image-dataset/data>. [Accessed 03 May 2024].