

## ĐỒ ÁN CHUYÊN NGÀNH

# NGHIÊN CỨU VỀ MẠNG NƠ-RON TÍCH CHẬP VÀ XÂY DỰNG WEBSITE TÌM KIẾM THÔNG TIN QUA PHÂN TÍCH HÌNH ẢNH

Ngành: **CÔNG NGHỆ THÔNG TIN**

Chuyên ngành: **CÔNG NGHỆ PHẦN MỀM**

Giảng viên hướng dẫn : TS.Nguyễn Thị Hải Bình

Sinh viên thực hiện :

Họ tên:

Nguyễn Phi Hùng

MSSV:

2080600875

Lớp:

20DTHE1

TP. Hồ Chí Minh, 2023

## LỜI CAM ĐOAN

Nhóm em xin cam đoan mọi kết quả của đồ án chuyên ngành này là công trình nghiên cứu độc lập và tự do của riêng nhóm dưới sự hướng dẫn của cô Nguyễn Thị Hải Bình, không có sự vay mượn của bất kì đồ án nào trước đây. Với số liệu và kết quả nghiên cứu trong bài báo cáo này là hoàn toàn trung thực và có sự tính toán chính xác. Tất cả hình ảnh được nêu trong đây đều liên quan đến chủ đề của đồ án. Mọi sự giúp đỡ giúp cho việc báo cáo này đã được cảm ơn với các thông tin trích dẫn và sử dụng đều có nguồn gốc rõ ràng.

TP. Hồ Chí Minh, Tháng 12 Năm 2023

Tác giả đồ án



Nguyễn Phi Hùng

# MỤC LỤC

DANH MỤC ẢNH .....	2
LỜI CẢM ƠN.....	4
LỜI MỞ ĐẦU .....	5
CHƯƠNG 1: MẠNG NƠ-RON .....	6
1.1. GIỚI THIỆU MẠNG NƠ-RON.....	6
1.3. CÁCH HOẠT ĐỘNG .....	9
1.4. CÁC LOẠI MẠNG NƠ-RON .....	11
CHƯƠNG 2: MẠNG NƠ-RON TÍCH CHẬP .....	12
2.1. GIỚI THIỆU TÍCH CHẬP .....	12
2.1.1. Tích chập một chiều.....	12
2.1.2. Thêm lè (Padding).....	13
2.1.3. Bước trượt (Stride).....	14
2.1.4. Tích chập 2 chiều .....	14
2.1.5. Thực hành áp dụng.....	15
2.2. MẠNG NƠ-RON TÍCH CHẬP (CONVOLUTIONAL NEURAL NETWORKS) .....	17
2.2.1. Tầng Convolution.....	17
2.2.2. Tầng kích hoạt (Activation) .....	20
2.2.3. Tầng Pooling .....	21
2.2.4. Tầng kết nối đầy đủ (Fully-Connected) .....	22
2.3. CÁC KIẾN TRÚC CNN HIỆN ĐẠI .....	24
2.3. EFFICIENTNET .....	26
2.3.1 Khả năng thu phóng .....	26
2.3.2. Các loại Model của EfficientNet.....	28
2.3.3. Độ hiệu quả của EfficientNet.....	30
CHƯƠNG 3: TRIỀN KHAI CHƯƠNG TRÌNH .....	33
3.1. CÁC MÔI TRƯỜNG VÀ CÔNG CỤ SỬ DỤNG .....	33
3.2. HUẤN LUYỆN MODEL (TRAINING MODEL) .....	34
3.2.1. Các bước thực hiện.....	34
3.2.2. Training model .....	37
3.3. DEMO TRIỀN KHAI MODEL LÊN WEBSITE .....	42
CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN .....	49
4.1. KẾT LUẬN .....	49
4.2. HƯỚNG PHÁT TRIỂN.....	49
TÀI LIỆU THAM KHẢO .....	50

## Danh Mục Ảnh

Hình 1.1: Cấu trúc mạng nơ-ron.....	10
Hình 2.1. Input và Output với kích thước mặc định .....	13
Hình 2.2. Input và Output với kích thước sau khi thêm lè.....	14
Hình 2.3. Không thêm lè, bước trượt mặc định ( $P = 0, S = 1$ ).....	15
Hình 2.4. Thêm lè, bước trượt mặc định ( $P = f - 1/2, S = 1$ ).....	15
Hình 2.5. Mẫu ảnh gốc.....	16
Hình 2.6. Mẫu ảnh sau khi convert.....	15
Hình 2.7. Mẫu ảnh khi dùng bộ lọc ngang.....	16
Hình 2.8. Mẫu ảnh khi dùng bộ lọc dọc.....	15
Hình 2.9. Minh họa về các lớp trong CNN .....	17
Hình 2.10. Minh họa bộ lọc, đầu vào và feature map .....	18
Hình 2.11. Ảnh RGB được tách ra.....	17
Hình 2.12. Kết hợp tích chập của các kênh.....	18
Hình 2.13. Mẫu ảnh và mẫu bộ lọc $7*7$ .....	18
Hình 2.14. Trích xuất đặc trưng của ô đầu tiên.....	19
Hình 2.15. Trích xuất đặc trưng của ô trên bên phải.....	19
Hình 2.16. Các biến thể khác của ReLU .....	21
Hình 2.17. Thực hiện Max-Pooling với bộ lọc $2*2$ .....	20
Hình 2.18. Thực hiện Average-Pooling với bộ lọc $2*2$ .....	21
Hình 2.19. Kết quả thu được sau khi thực hiện max-pooling .....	22
Hình 2.20. Làm phẳng tensor .....	23
Hình 2.21. Quá trình từ làm phẳng đến lúc ra kết quả .....	23
Hình 2.22. Timeline của kiến trúc CNN .....	24
Hình 2.23. Mạng CNN được thu phóng theo nhiều kiểu .....	26
Hình 2.24. Kết quả sau khi thực hiện các hoạt động thu phóng.....	27
Hình 2.25. Cấu trúc mạng EfficientNet-B0.....	29
Hình 2.26. Thông số mạng EfficientNet-B0 .....	29
Hình 2.27. So sánh với các cấu trúc mạng khác theo FLOPS, Parameter .....	30
Hình 2.28. So sánh với các Model phổ biến.....	31

Hình 2.29. So sánh Model với các Dataset khác nhau .....	32
Hình 2.30. Đạt top trên the ImageNet task .....	32
Hình 2.31. Các kết quả test trên ImageNet dataset .....	32
Hình 3.1. Trang chủ kaggle .....	35
Hình 3.2. Setting notebook .....	35
Hình 3.3. Chọn dataset trên kaggle .....	36
Hình 3.4. Dataset sau khi được chọn .....	36
Hình 3.5. Input và Output của notebook .....	37
Hình 3.6. Thư viện sử dụng để training .....	37
Hình 3.7. Lưu đường dẫn từ input vào để sử dụng .....	38
Hình 3.8. Mapping file .txt để lấy ra tên class .....	38
Hình 3.9. Generate ảnh để tiến hành training .....	38
Hình 3.10. Tạo model và tiến hành đặt các lớp .....	39
Hình 3.11. Trainning model .....	39
Hình 3.12. Thông tin của từng epoch .....	40
Hình 3.13. Hiển thị biểu đồ sau khi trainning .....	40
Hình 3.14. Biểu diễn sự thay đổi về accuracy trong lúc train .....	41
Hình 3.15. Biểu diễn sự thay đổi về loss trong lúc train .....	41
Hình 3.16. Kết quả model sau khi test .....	42
Hình 3.17. Giao diện trang chủ .....	43
Hình 3.18. Chọn ảnh từ thư mục .....	43
Hình 3.19. Dự đoán shower cap .....	44
Hình 3.20. Danh sách các lớp và tỉ lệ chính xác của nó .....	44
Hình 3.21. Tên của class đã được dự đoán .....	45
Hình 3.22. Tìm kiếm trên Google .....	45
Hình 3.23. Dự đoán chihuahua .....	46
Hình 3.24. Dự đoán sport car .....	46
Hình 3.25. Dự đoán tabby cat .....	47
Hình 3.26. Dự đoán warter bottle .....	48
Hình 3.27. Dự đoán hair drier .....	48

## LỜI CẢM ƠN

Em, Nguyễn Phi Hùng xin gửi một lời cảm ơn chân thành đến cô Nguyễn Thị Hải Bình đã nhiệt tình hỗ trợ, giúp đỡ tụi em trong suốt thời gian thực hiện đồ án chuyên ngành (25/09/2023 – 17/12/2023). Ngoài ra còn có sự giúp đỡ nhiệt tình của các bạn xung quanh đã làm nên một đồ án chuyên ngành đây đủ và chính xác như ngày hôm nay. Những kiến thức mà em nhận được trong quá trình làm đồ án sẽ là hành trang giúp chúng em vững bước trong tương lai. Tuy đồ án đã được hoàn thành nhưng vẫn còn khá nhiều thiếu sót, mong các bạn và cô tích cực đóng góp để sản phẩm của nhóm chúng em càng ngày càng hoàn thiện hơn. Xin cảm ơn rất nhiều!

Cuối cùng, với lòng quý trọng và biết ơn sâu sắc em xin kính chúc Cô dồi dào sức khỏe và thành công hơn nữa trong sự nghiệp giảng dạy của mình.

Sinh viên thực hiện: Nguyễn Phi Hùng.

## LỜI MỞ ĐẦU

Trong thời đại hiện đại, sự phát triển của công nghệ đã mở ra những cánh cửa mới đầy hứa hẹn, mà một trong những xu hướng đáng chú ý nhất chính là Machine Learning và Deep Learning. Đây không chỉ là những khái niệm kỹ thuật mà còn là những động lực mạnh mẽ đằng sau sự tiến bộ của nhiều lĩnh vực khác nhau trong cuộc sống hàng ngày.

Machine Learning, hay Học Máy, là một lĩnh vực của trí tuệ nhân tạo (AI) mà từng bước chuyển đổi cách chúng ta nghĩ về việc máy tính có khả năng học hỏi từ dữ liệu và tự động cải thiện hiệu suất mà không cần sự can thiệp trực tiếp từ con người. Được áp dụng rộng rãi từ dự đoán tài chính, xử lý ngôn ngữ tự nhiên đến tư vấn cá nhân, Machine Learning đang làm thay đổi cách chúng ta tương tác với công nghệ.

Trong khi đó, Deep Learning, hoặc Học Sâu, là một phần con của Machine Learning, nhấn mạnh vào việc sử dụng các mô hình mạng nơ-ron sâu để hiểu và biểu diễn dữ liệu phức tạp. Với khả năng học hỏi từ lượng lớn dữ liệu và đào tạo mô hình phức tạp, Deep Learning đã mang lại những thành tựu ấn tượng trong các lĩnh vực như thị giác máy tính, xử lý ngôn ngữ tự nhiên và tự động lái xe.

Là những công nghệ ngày càng trở nên không thể thiếu, Machine Learning và Deep Learning không chỉ giúp chúng ta hiểu rõ hơn về thế giới xung quanh, mà còn mở ra những triển vọng mới cho sự đổi mới và cải tiến. Chúng là những công cụ mạnh mẽ, tạo nên sự kết nối giữa con người và máy tính, đưa chúng ta vào một tương lai mà sự thông minh nhân tạo không còn là khả năng ẩn sau đám mây của tương lai, mà là hiện thực ngày nay.

Mạng nơ-ron nhân tạo chính là động lực chính để phát triển Deep Learning. Các mạng nơ-ron sâu (DNN) bao gồm nhiều lớp nơ-ron khác nhau, có khả năng thực hiện các tính toán có độ phức tạp rất cao. Deep Learning hiện đang phát triển rất nhanh và được xem là một trong những bước đột phá lớn nhất trong Machine Learning.

# **CHƯƠNG 1: MẠNG NƠ-RON**

## **1.1. Giới thiệu mạng Nơ-ron**

Mạng nơ-ron là một phương pháp trong lĩnh vực trí tuệ nhân tạo, mô phỏng cách bộ não con người xử lý thông tin. Đây là một phương tiện trong deep learning, trong đó các nút hoặc nơ-ron kết nối với nhau để tạo thành một cấu trúc tương tự như bộ não con người. Mô hình này tạo ra một hệ thống linh hoạt, cho phép máy tính học từ dữ liệu và ngày càng cải thiện hiệu suất của mình thông qua việc sửa lỗi.

Mạng nơ-ron nhân tạo nhằm giải quyết các vấn đề phức tạp, như tóm tắt tài liệu hoặc nhận diện khuôn mặt, với độ chính xác cao. Khả năng học hỏi từ kinh nghiệm và tự động điều chỉnh dựa trên dữ liệu mới giúp mô hình nhanh chóng thích ứng với môi trường và nhiệm vụ cụ thể. Điều này làm cho mạng nơ-ron trở thành công cụ mạnh mẽ trong nhiều ứng dụng thực tế, nơi đòi hỏi khả năng xử lý thông tin phức tạp và đưa ra quyết định chính xác.

## **1.2. Ứng dụng**

Mạng nơ-ron được sử dụng trong nhiều trường hợp trải dài khắp các lĩnh vực, chẳng hạn như:

- Chẩn đoán y tế bằng cách phân loại hình ảnh y khoa
- Tiếp thị nhằm mục tiêu bằng cách lọc mạng xã hội và phân tích dữ liệu hành vi
- Dự đoán tài chính bằng cách xử lý dữ liệu lịch sử của các công cụ tài chính
- Dự báo nhu cầu năng lượng và phụ tải điện
- Kiểm soát quy trình và chất lượng
- Nhận dạng hợp chất hóa học

Bên dưới là 4 ứng dụng quan trọng của mạng nơ-ron.

## **Thị giác máy tính**

Thị giác máy tính là khả năng trích xuất dữ liệu cũng như thông tin chuyên sâu từ hình ảnh và video của máy tính. Với mạng nơ-ron, máy tính có thể phân biệt và nhận diện hình ảnh tương tự như con người. Thị giác máy tính được ứng dụng trong nhiều trường hợp, chẳng hạn như:

- Hệ thống nhận diện hình ảnh trên ô tô tự lái để chúng có thể nhận ra các biển báo giao thông cũng như những người tham gia giao thông khác
- Kiểm duyệt nội dung để tự động loại bỏ nội dung không an toàn hoặc không phù hợp khỏi kho lưu trữ hình ảnh và video
- Nhận diện khuôn mặt để xác định khuôn mặt cũng như các đặc điểm như mồm, đeo kính và để râu
- Dán nhãn hình ảnh để xác định logo thương hiệu, quần áo, đồ bảo hộ và các chi tiết hình ảnh khác

## **Nhận dạng giọng nói**

Mạng nơ-ron có thể phân tích giọng nói con người, bất kể mẫu giọng, cao độ, tông, ngôn ngữ và giọng vùng miền khác nhau. Trợ lý ảo như Amazon Alexa và phần mềm phiên âm tự động sử dụng nhận dạng giọng nói để thực hiện các công việc như:

- Hỗ trợ các nhân viên trực tổng đài và tự động phân loại cuộc gọi
- Chuyển đổi các cuộc trò chuyện về y khoa thành văn bản trong thời gian thực
- Tạo phụ đề chính xác cho video và bản ghi âm cuộc họp để mở rộng phạm vi tiếp cận nội dung

## Kỹ thuật xử lý ngôn ngữ tự nhiên

Kỹ thuật xử lý ngôn ngữ tự nhiên (NLP) là khả năng xử lý văn bản tự nhiên do con người tạo ra. Mạng nơ-ron giúp máy tính thu thập thông tin chuyên sâu và ý nghĩa từ dữ liệu văn bản và tài liệu. NLP được sử dụng trong nhiều trường hợp, bao gồm trong những chức năng sau:

- Tổng đài viên ảo và chatbot tự động
- Tự động sắp xếp và phân loại dữ liệu được ghi
- Phân tích nghiệp vụ thông minh các tài liệu dài như email và biểu mẫu
- Lập chỉ mục các cụm từ quan trọng thể hiện cảm xúc, ví dụ như những bình luận tích cực và tiêu cực trên mạng xã hội
- Tóm tắt tài liệu và tạo bài viết về một chủ đề cho trước

## Công cụ để xuất

Mạng nơ-ron có thể theo dõi hoạt động của người dùng để đưa ra các đề xuất được cá nhân hóa. Chúng cũng có thể phân tích mọi hành vi của người dùng và tìm ra các sản phẩm hoặc dịch vụ mới mà người dùng cụ thể có thể quan tâm. Ví dụ: Curalate - một công ty khởi nghiệp có trụ sở tại Philadelphia - giúp các thương hiệu kiêm doanh số từ những bài đăng trên mạng xã hội. Các thương hiệu sử dụng dịch vụ gắn thẻ sản phẩm thông minh (IPT) của Curalate để tự động hóa việc thu thập và tuyển lựa nội dung do người dùng tạo trên mạng xã hội. IPT sử dụng mạng nơ-ron để tự động tìm và đề xuất các sản phẩm có liên quan đến hoạt động của người dùng trên mạng xã hội. Người tiêu dùng không còn phải săn lùng các danh mục trực tuyến để tìm một sản phẩm cụ thể từ hình ảnh trên mạng xã hội. Thay vào đó, họ có thể sử dụng dịch vụ tự động gắn thẻ sản phẩm của Curalate để mua hàng một cách dễ dàng.

### **1.3. Cách hoạt động**

Bộ não con người chính là nguồn cảm hứng cho kiến trúc mạng nơ-ron. Các tế bào não của con người, còn được gọi là nơ-ron, tạo thành một mạng lưới phức tạp, có tính liên kết cao và gửi các tín hiệu điện đến nhau để giúp con người xử lý thông tin. Tương tự, một mạng nơ-ron nhân tạo được tạo ra từ các tế bào nơ-ron nhân tạo, cùng nhau phối hợp để giải quyết một vấn đề. Nơ-ron nhân tạo là các mô đun phần mềm, được gọi là nút và mạng nơ-ron nhân tạo là các chương trình phần mềm hoặc thuật toán mà về cơ bản, sử dụng hệ thống máy tính để giải quyết các phép toán.

#### **Kiến trúc mạng nơ-ron đơn giản**

Một mạng nơ-ron cơ bản bao gồm các nơ-ron nhân tạo liên kết theo 3 lớp:

- Lớp đầu vào

Thông tin từ thế giới bên ngoài đi vào mạng nơ-ron nhân tạo qua lớp đầu vào. Các nút đầu vào xử lý dữ liệu, phân tích hoặc phân loại và sau đó chuyển dữ liệu sang lớp tiếp theo.

- Lớp ẩn

Dữ liệu đi vào lớp ẩn đến từ lớp đầu vào hoặc các lớp ẩn khác. Mạng nơ-ron nhân tạo có thể có một số lượng lớn lớp ẩn. Mỗi lớp ẩn phân tích dữ liệu đầu ra từ lớp trước, xử lý dữ liệu đó sâu hơn và rồi chuyển dữ liệu sang lớp tiếp theo.

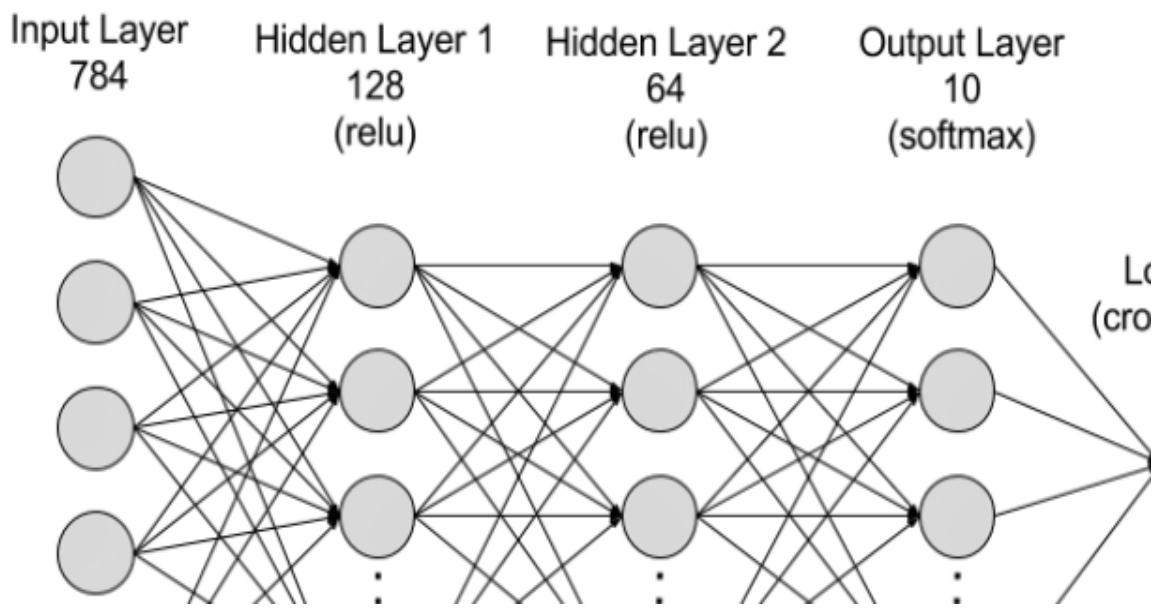
- Lớp đầu ra

Lớp đầu ra cho ra kết quả cuối cùng của tất cả dữ liệu được xử lý bởi mạng nơ-ron nhân tạo. Lớp này có thể có một hoặc nhiều nút. Ví dụ: giả sử chúng ta gấp phải một vấn đề phân loại nhị phân (có/không), lớp đầu ra sẽ có một nút đầu ra, nút này sẽ cho kết quả 1 hoặc 0. Tuy nhiên, nếu chúng ta gấp phải vấn đề phân loại nhiều lớp, lớp đầu ra sẽ có thể bao gồm nhiều hơn một nút đầu ra.

## Kiến trúc mạng nơ-ron chuyên sâu

Mạng nơ-ron chuyên sâu, hoặc mạng deep learning, có nhiều lớp ẩn với hàng triệu nơ-ron nhân tạo liên kết với nhau. Một con số, có tên gọi là trọng số, đại diện cho các kết nối giữa hai nút. Trọng số sẽ dương nếu một nút kích thích nút còn lại, hoặc âm nếu một nút ngăn cản nút còn lại. Các nút với trọng số cao hơn sẽ có ảnh hưởng lớn hơn lên các nút khác.

Về mặt lý thuyết, mạng nơ-ron chuyên sâu có thể ánh xạ bất kỳ loại dữ liệu đầu vào với bất kỳ loại dữ liệu đầu ra nào. Tuy nhiên, chúng cũng cần được đào tạo hơn rất nhiều so với các phương pháp máy học khác. Chúng cần hàng triệu ví dụ về dữ liệu đào tạo thay vì hàng trăm hoặc hàng nghìn ví dụ mà một mạng đơn giản hơn thường cần.



Hình 1.1: Cấu trúc mạng nơ-ron

## 1.4. Các loại mạng Nơ-ron

Mạng nơ-ron nhân tạo có thể được phân loại theo phương thức dữ liệu được truyền từ nút đầu vào đến nút đầu ra. Dưới đây là một số ví dụ:

### Mạng nơ-ron truyền thẳng

Mạng nơ-ron truyền thẳng xử lý dữ liệu theo một chiều, từ nút đầu vào đến nút đầu ra. Mỗi nút trong một lớp được kết nối với tất cả các nút trong lớp tiếp theo. Mạng truyền thẳng sử dụng một quy trình phản hồi để cải thiện dự đoán theo thời gian.

### Thuật toán truyền ngược

Mạng nơ-ron nhân tạo liên tục học hỏi bằng cách sử dụng vòng lặp phản hồi hiệu chỉnh để cải thiện phân tích dự đoán của chúng. Đơn giản mà nói, bạn có thể coi rằng dữ liệu truyền từ nút đầu vào đến nút đầu ra qua nhiều lối đi khác nhau trong mạng nơ-ron. Chỉ có duy nhất một lối đi chính xác, ánh xạ nút đầu vào đến nút đầu ra thích hợp. Để tìm ra lối đi này, mạng nơ-ron sử dụng một vòng lặp phản hồi với cách hoạt động như sau:

- Mỗi nút đưa ra một dự đoán về nút tiếp theo trên lối đi.
- Nút này sẽ kiểm tra tính chính xác của dự đoán. Các nút sẽ chỉ định giá trị trọng số cao hơn cho những lối đi tới nhiều dự đoán chính xác hơn và giá trị trọng số thấp hơn cho các lối đi tới dự đoán không chính xác.
- Đối với điểm dữ liệu tiếp theo, các nút đưa ra dự đoán mới bằng cách sử dụng các lối đi có trọng số cao hơn rồi lặp lại bước 1.

### Mạng nơ-ron tích chập

Những lớp ẩn trong mạng nơ-ron tích chập thực hiện các chức năng toán học cụ thể, như tóm tắt hoặc sàng lọc, được gọi là tích chập. Chúng rất hữu ích trong việc phân loại hình ảnh vì chúng có thể trích xuất các đặc điểm liên quan từ hình ảnh, điều này có lợi cho việc nhận dạng và phân loại hình ảnh.

## CHƯƠNG 2: MẠNG NỐ-RON TÍCH CHẬP

### 2.1. Giới thiệu tích chập

Tích chập là một khái niệm trong xử lý tín hiệu số nhằm biến đổi thông tin đầu vào thông qua một phép tích chập với bộ lọc để trả về đầu ra là một tín hiệu mới. Tín hiệu này sẽ làm giảm những đặc trưng mà bộ lọc không quan tâm và chỉ giữ những đặc trưng chính.

Tích chập đóng một vai trò quan trọng và xuất hiện từ sớm trong lịch sử xử lý tín hiệu số. Việc tìm ra các bộ lọc phù hợp cho mỗi loại tín hiệu và mỗi bài toán đã được nghiên cứu và giảng dạy rất nhiều trong các giáo trình kỹ thuật.

Cuối những năm 1980s, Yann Lecunn đề xuất một mô hình tích chập hai chiều cho dữ liệu ảnh và thu lại thành công lớn trong bài toán phân loại chữ số viết tay. Bằng việc sử dụng rất nhiều dữ liệu và thay các tầng nối kín (fully connected layer) trong mạng perceptron đa tầng bởi tích chập hai chiều, các bộ lọc phù hợp với bài toán và dữ liệu có thể được học để mang lại kết quả phân lớp tốt nhất.

#### 2.1.1. Tích chập một chiều

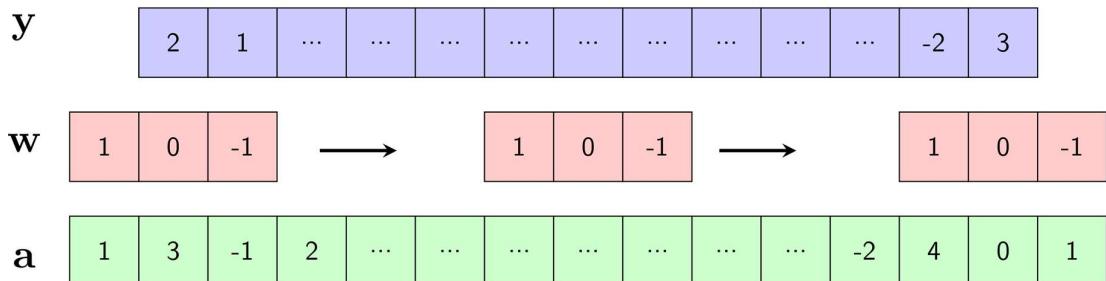
Xét tín hiệu một chiều  $a(n)$  và bộ lọc (filter)  $w(n)$ . Tích chập của tín hiệu và bộ lọc là một tín hiệu một chiều mới  $b(n)$  được xác định theo công thức:

$$b(n) = a(n) * w(n) = \sum_{k=-\infty}^{\infty} a(k)w(n-k) \quad (1)$$

Trong mạng neuron tích chập, tích chập được định nghĩa khác đi một chút. Cho tín hiệu đầu vào và bộ lọc lần lượt là các vector  $a \in R^N$  và  $w \in R^f$  ( $f$  có thể là một số tự nhiên bất kỳ nhưng thường là số lẻ). Khi đó đầu ra là một vector  $y$  với từng phần tử được tính bởi:

$$y_n = \sum_{i=0}^{f-1} a_{n+i}w_i \quad (2)$$

với  $n$  thoả mãn  $0 \leq n + i < N, \forall i = 0, 1, \dots, f - 1$ . Điều này tương đương với  $0 \leq n < N - f + 1$ . Vậy  $y \in R^{N-f+1}$ .



Hình 2.1. Input và Output với kích thước mặc định

Quá trình tính đầu ra  $y$  có thể được thực hiện như sau:

1. Đặt bộ lọc  $w$  vào vị trí tương ứng với  $f$  phần tử đầu tiên của  $a$ .
2. Nhân từng phần tử tương ứng của  $w$  và  $a$  rồi cộng các kết quả lại để được phần tử tương ứng của  $y$ .
3. Trượt bộ lọc  $w$  một bước sang bên phải. Nếu phần tử cuối cùng của bộ lọc không vượt ra ngoài phần tử cuối cùng của tín hiệu, quay lại Bước 2. Ngược lại, dừng các bước tính toán.

Ta có ví dụ như hình ảnh trên:

$$y_0 = a_0w_0 + a_1w_1 + a_2w_2 = 1 - (-1) = 2$$

$$y_1 = a_1w_0 + a_2w_1 + a_3w_2 = 3 - 2 = 1$$

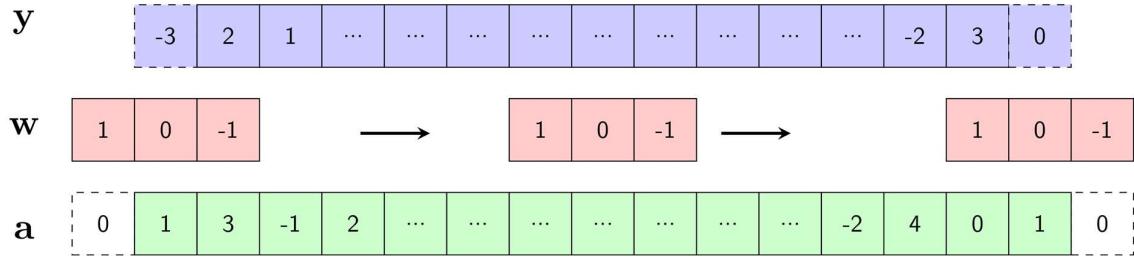
### 2.1.2. Thêm lè (Padding)

Với hình ảnh trên ta có thể thấy kích thước đầu ra (output)  $y$  đã nhỏ hơn kích thước đầu vào (input)  $x$  vì ta chỉ trượt bộ lọc  $w$  đến khi hết tín hiệu đầu vào. Trong trường hợp ta muốn kích thước của đầu ra bằng với kích thước của đầu vào, ta có thể thêm các giá trị bằng không ở 2 phía của tín hiệu đầu vào. Kỹ thuật này được gọi là thêm lè (padding).

Để có kích thước bằng nhau, tín hiệu đầu vào cần được thêm  $f - 1$  giá trị bằng không (với  $f$  là số phần tử đầu tiên của  $a$  khi bộ lọc  $w$  chạy vào). Khi  $f$  là một số lẻ, mỗi phía của tín hiệu đầu vào thường được thêm  $P = (f - 1)/2$  giá trị bằng không.

Giá trị  $P$  có thể là một số tự nhiên bất kỳ thuộc vào từng trường hợp,  $P$  không nhất thiết phải đảm bảo kích thước đầu ra và đầu vào của tín hiệu là như nhau. Khi không sử dụng thêm lè,  $P = 0$ . Việc thêm lè sẽ được minh họa bằng hình ảnh sau:

Khi  $f = 3$ , ta sẽ thêm  $P = 1$  ô vào 2 đầu của tín hiệu đầu vào (input) a. Kết quả của đầu ra (output) khi đó sẽ có kích thước bằng nhau.



Hình 2.2. Input và Output với kích thước sau khi thêm lè

### 2.1.3. Bước trượt (Stride)

Như ví dụ trên, bộ lọc w được dịch chuyển từng bước sang phải một khoảng cách đó là 1 ô. Trong một số trường hợp, số lượng ô dịch chuyển có thể khác 1. Số lượng ô dịch chuyển đó được gọi là bước trượt (Stride) được ký hiệu bằng chữ S. Lúc này, công thức tổng quát của chúng ta từ (2) chuyên thành:

$$y_n = \sum_{i=0}^{f-1} a_{nS+i}w_i \quad (3)$$

Trong trường hợp sử dụng thêm lè với độ rộng P, với kích thước của đầu vào (input) là N + 2P, giá trị của n để thoả mãn công thức trên là:

$$0 \leq nS + i < N + 2P, \forall i = 0, 1, \dots, f-1 \Leftrightarrow n \leq \frac{N + 2P - f}{S}$$

Thông thường P và S sẽ được chọn sao cho kết quả của  $\frac{N+2P-f}{S}$  là số nguyên.

Điều này dẫn đến công thức cuối cùng:

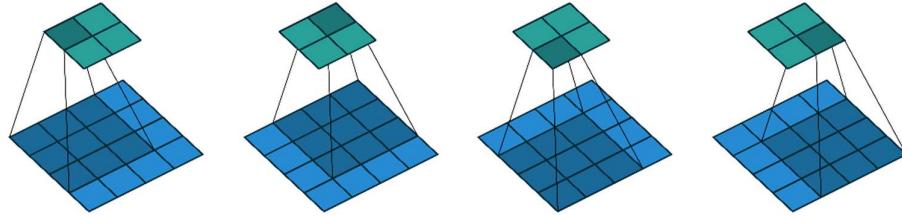
$$N_1 = \frac{N + 2P - f}{S} + 1 \quad (4)$$

### 2.1.4. Tích chập 2 chiều

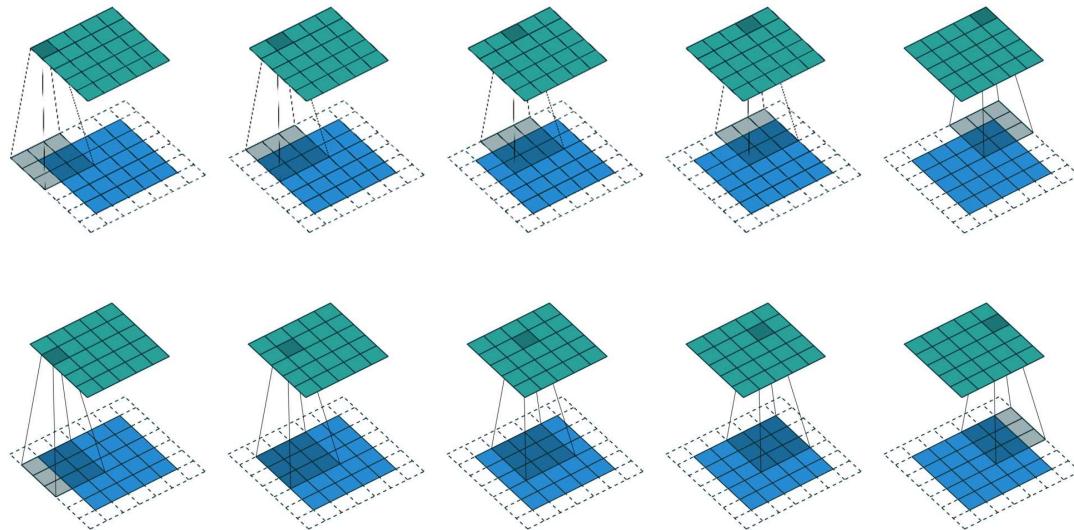
Tích chập hai chiều (2D convolution) là một phép toán cơ sở trong xử lý ảnh, được sử dụng để thực hiện các phép toán lọc hai chiều. Ngày nay, với sự ra đời của các mạng nơ-ron tích chập, phép toán tích chập hai chiều là một thành phần quan trọng của

các mạng nơ-ron tích chập, và đóng góp đến trên 90% khối lượng tính toán của các mô hình thực thi mạng nơ-ron tích chập.

Sau đây là một số ảnh minh họa về tích chập 2 chiều đơn kênh:



Hình 2.3. Không thêm lè, bước trượt mặc định ( $P = 0, S = 1$ )



Hình 2.4. Thêm lè, bước trượt mặc định ( $P = \frac{f-1}{2}, S = 1$ )

### 2.1.5. Thực hành áp dụng

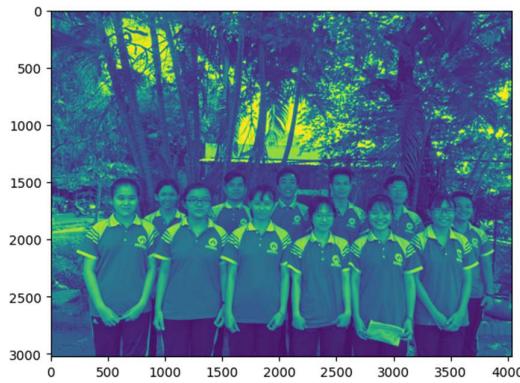
Ta sẽ thử thực hành sử dụng mạng tích chập hai chiều để trích xuất các đặc trưng chính của một bức ảnh. Các bộ lọc được sử dụng là:

$$\text{Bộ lọc ngang: } w = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

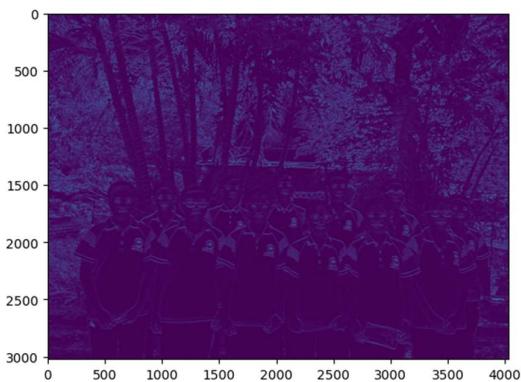
$$\text{Bộ lọc dọc: } w = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$



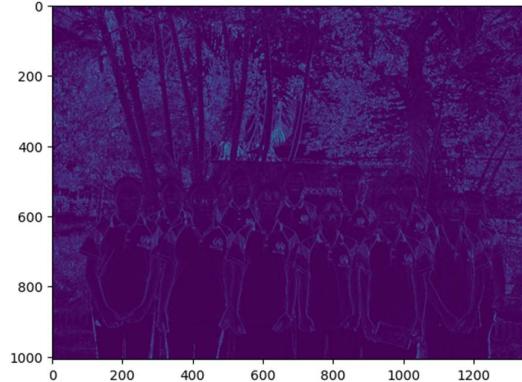
Hình 2.5. Mẫu ảnh gốc



Hình 2.6. Mẫu ảnh sau khi convert



Hình 2.7. Mẫu ảnh khi dùng bộ lọc ngang



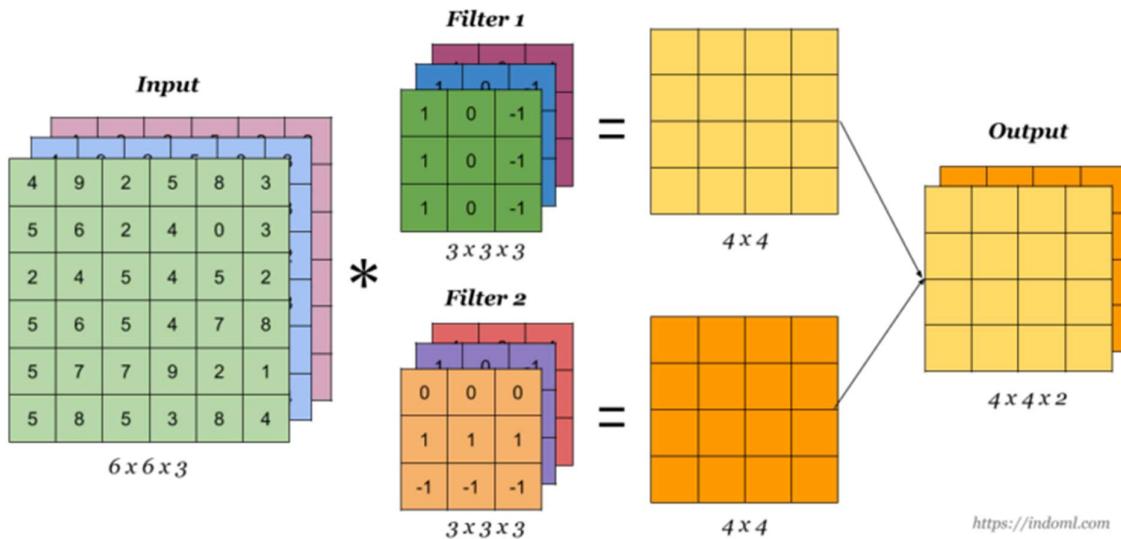
Hình 2.8. Mẫu ảnh khi dùng bộ lọc dọc

Sau khi dùng bộ lọc ngang để có được tấm ảnh 2.7, ta nhận thấy các đường nét được sáng lên sẽ là các vật thể theo chiều ngang như vai áo, tà áo, mắt kính, ... Do các điểm ảnh đầu ra bằng tổng giá trị các điểm phía trên trừ cho tổng giá trị các điểm phía dưới. Đối với các đường nét nằm ngang thì cường độ sáng ít thay đổi theo phương ngang nhưng xét theo chiều dọc thì chúng sẽ thay đổi. Do đó hiệu giữa 2 tổng phía trên và dưới càng lớn dẫn tới giá trị của tích chập càng lớn khi trượt theo các đường nét nằm ngang này. Khi hoàn thành thiện ma trận tích chập các đường nét nằm ngang sẽ có cường độ sáng lớn hơn nên nổi bật hơn.

Tương tự ta sẽ có bộ lọc dọc với đầu ra như ảnh 2.8. Bộ lọc đã cho thấy các đường nét sáng lên khi nay đã mất, thay vào đó là các đường nét dọc như cánh tay, óng quần, ... Như vậy chúng ta có thể thấy mỗi bộ lọc sẽ có 1 tác dụng trích xuất một đặc trưng khác nhau từ cùng 1 bức ảnh.

## 2.2. Mạng Nơ-ron tích chập (Convolutional Neural Networks)

Các kiến trúc dựa trên CNN hiện nay xuất hiện trong mọi ngóc ngách của lĩnh vực thị giác máy tính, và đã trở thành kiến trúc chủ đạo mà hiếm ai ngày nay phát triển các ứng dụng thương mại hay tham gia một cuộc thi nào đó liên quan tới nhận dạng ảnh, phát hiện đối tượng, hay phân vùng theo ngữ cảnh mà không xây nền móng dựa trên phương pháp này.



Hình 2.9. Minh họa về các lớp trong CNN

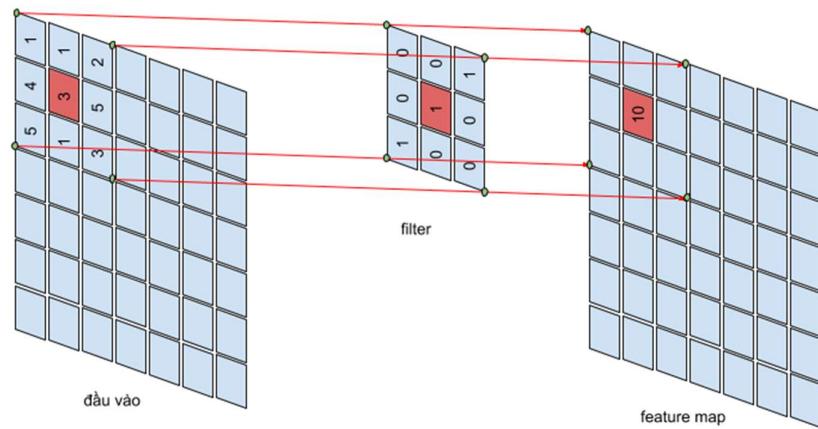
CNN được cấu tạo bởi các tầng convolution, non-linear activation, pooling và full connected. Các tầng sẽ có các chức năng và nhiệm vụ khác nhau. Tất cả làm nên một tổng thể đầy đủ về mạng CNN.

### 2.2.1. Tầng Convolution

Convolution layer thường là lớp đầu tiên trong mô hình CNN. Lớp này có chức năng phát hiện ra các đặc trưng về không gian một cách hiệu quả.

#### Filter – Bộ lọc

Filter là một trong những thành phần quan trọng trong CNN. Kích thước filter trong tầng convolutional phổ biến hiện nay là  $3 \times 3$ . Kích thước Filter thường là số lẻ, ví dụ như  $3 \times 3$ ,  $5 \times 5$ ,  $7 \times 7$ .

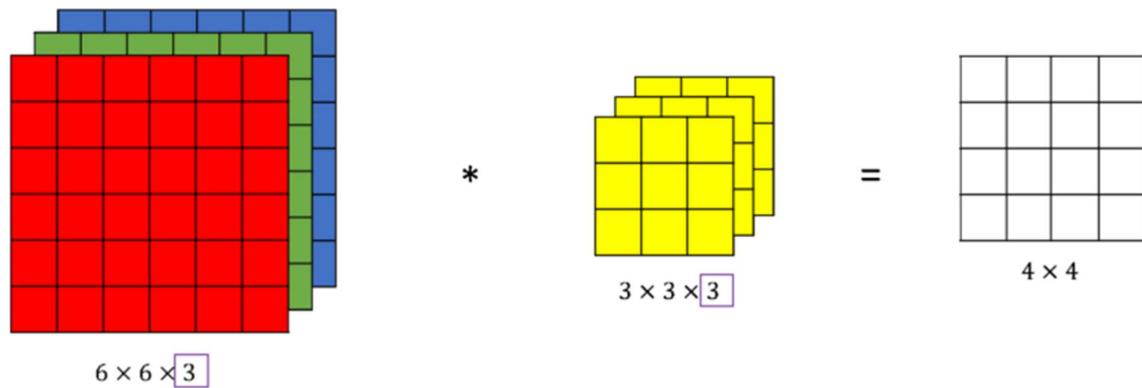


Hình 2.10. Minh họa bộ lọc, đầu vào và feature map

Kích thước của các Filter thường không quá lớn. Vì với kích thước nhỏ nó có thể trích xuất cục bộ chi tiết hơn, kích thước ảnh giảm chậm hơn; làm cho mạng sâu hơn và số lượng tham số phải học thấp hơn. Sử dụng 2 Filter kích thước  $3 \times 3$  (có 18 tham số) sẽ tối ưu hơn dùng 1 Filter  $5 \times 5$  (25 tham số).

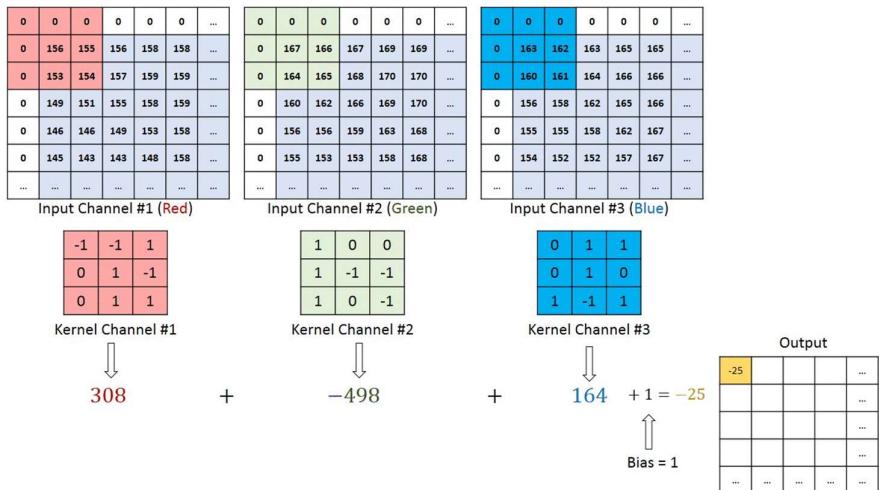
## 2D Convolution

Ta đã phân tích về tích chập 2 chiều ở mục 2.1.4 nên phần này ta sẽ tìm hiểu về cách tìm đặc trưng chính trong ảnh màu RGB (red, green, blue).



Hình 2.11. Ảnh RGB được tách ra

Đầu ra của ảnh RGB vẫn là một tensor 2 chiều bởi sự kết hợp các tích chập trên từng kênh màu như sau:



Hình 2.12. Kết hợp tích chập của các kênh

### Stride, Padding

Như phân tích tích chập 2 chiều ở trên, ta đã tìm hiểu và phân tích về stride và padding ở trên nên mục này ta sẽ đi sâu vào công thức (4) với ảnh đầu vào có kích thước  $W * H$ .

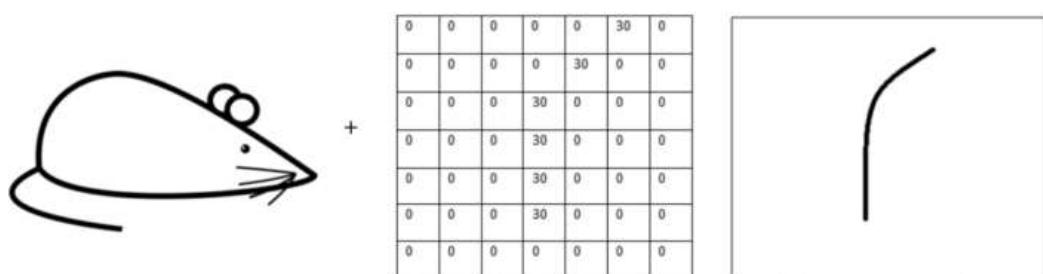
Với ảnh đầu vào có kích thước  $W * H$ , bộ lọc  $F * F$ , bước nhảy stride  $S$ , padding  $P$  thì feature map sẽ có kích thước  $W' * H'$  trong đó:

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

$$H' = \frac{H + 2P - F}{S} + 1$$

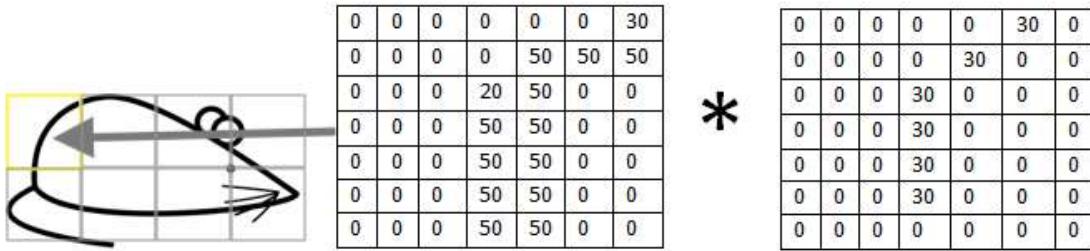
### Feature Detector (Phát hiện đặc trưng)

Để dễ dàng hình dung, ta sẽ ví dụ bằng bộ lọc  $7*7$  để tìm vị trí đường cong của con chuột ở bên trái giống với đường cong của bộ lọc.



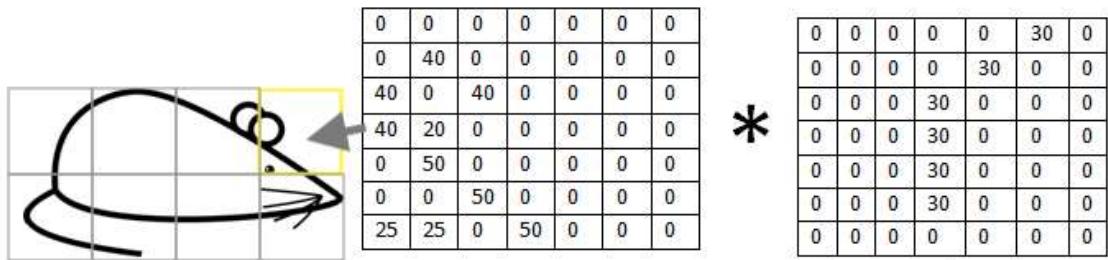
Hình 2.13. Mẫu ảnh và mẫu bộ lọc  $7*7$

Chia ảnh chuột ra thành các ô ma trận và bắt đầu xét ô ma trận đầu tiên.



Hình 2.14. Trích xuất đặc trưng của ô đầu tiên

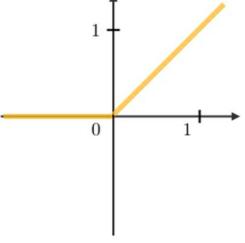
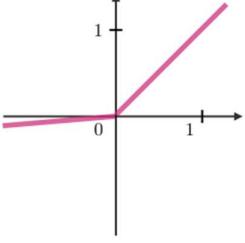
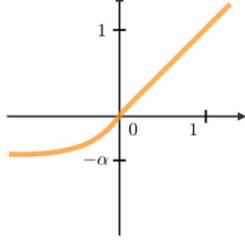
Thực hiện nhân ma trận đầu vào với bộ lọc  $7 \times 7$  ta được kết quả bằng 6000. Từ đó ta xác định được ô ma trận này có đường cong mà bộ lọc cần. Tương tự xét ô góc phải bên trên của hình chuột ta sẽ nhận được kết quả bằng 0 nên trường hợp này sẽ không phát hiện được đặc trưng của ảnh.



Hình 2.15. Trích xuất đặc trưng của ô trên bên phải

### 2.2.2. Tầng kích hoạt (Activation)

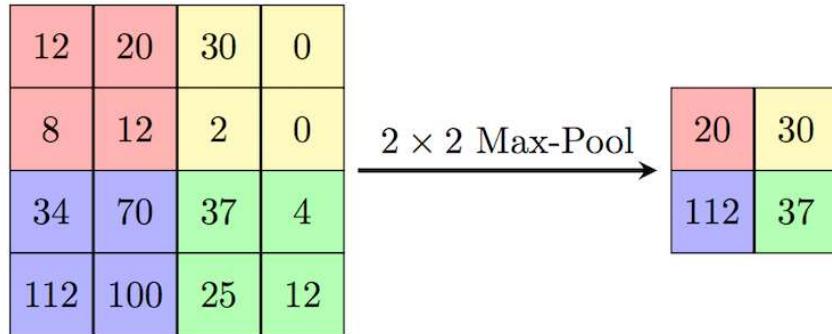
Trong cấu trúc CNN, tầng activation thường sẽ theo sau tầng convolution. Có nhiều loại hàm kích hoạt nhưng phổ biến được dùng là ReLU (Rectified Linear Unit). Một số biến thể khác của ReLU được thể hiện qua bảng sau:

ReLU	Leaky ReLU	ELU
$g(z) = \max(0, z)$	$g(z) = \max(\epsilon z, z)$ với $\epsilon \ll 1$	$g(z) = \max(\alpha(e^z - 1), z)$ với $\alpha \ll 1$
		
• Độ phức tạp phi tuyến tính có thể thông dịch được về mặt sinh học	• Gán vấn đề ReLU chết cho những giá trị âm	• Khả vi tại mọi nơi

Hình 2.16. Các biến thể khác của ReLU

### 2.2.3. Tầng Pooling

Tầng pooling (POOL) là một phép downsampling, thường được sử dụng sau tầng tích chập, giúp tăng tính bất biến không gian. Cụ thể, max pooling và average pooling là những dạng pooling đặc biệt, mà tương ứng là trong đó giá trị lớn nhất và giá trị trung bình được lấy ra.



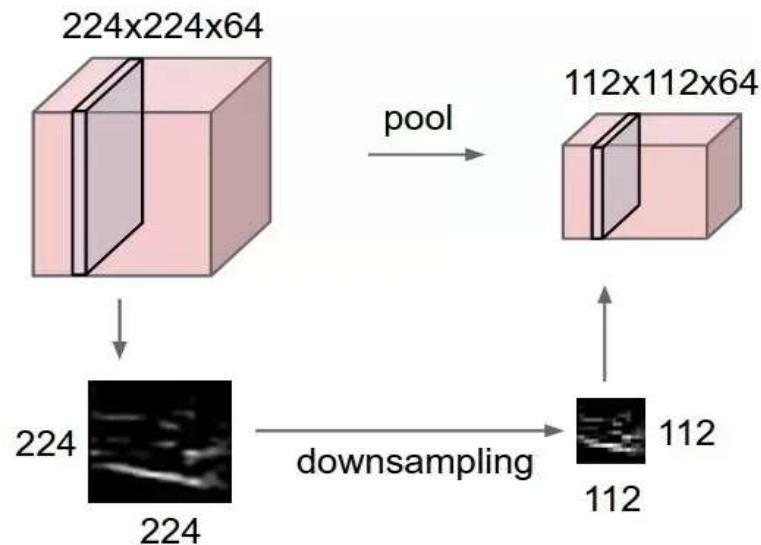
Hình 2.17. Thực hiện Max-Pooling với bộ lọc  $2 \times 2$

12	20	30	0
8	12	2	0
34	70	37	10
112	100	25	12

$2 \times 2$  Average-Pool

13	8
79	21

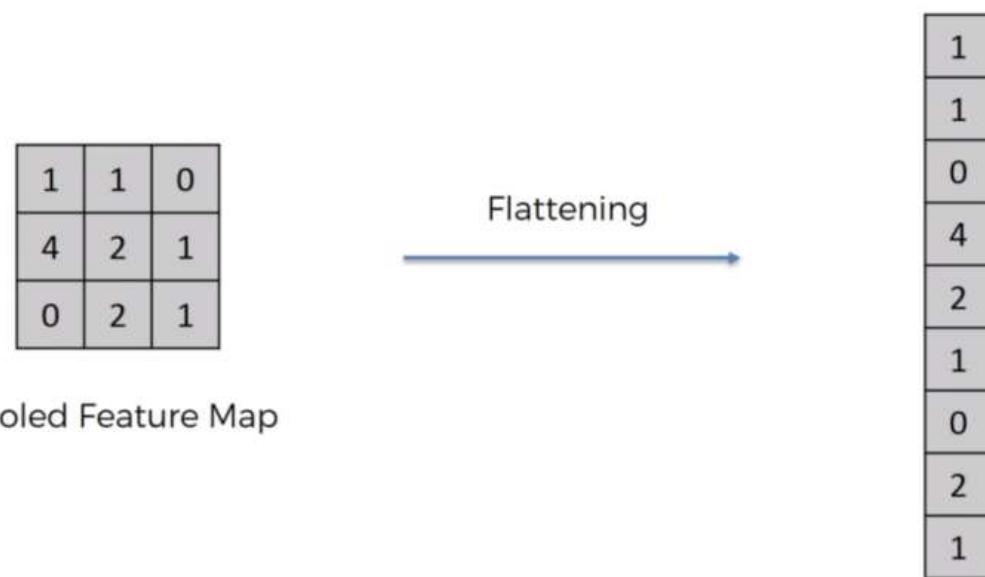
Hình 2.18. Thực hiện Average-Pooling với bộ lọc  $2*2$



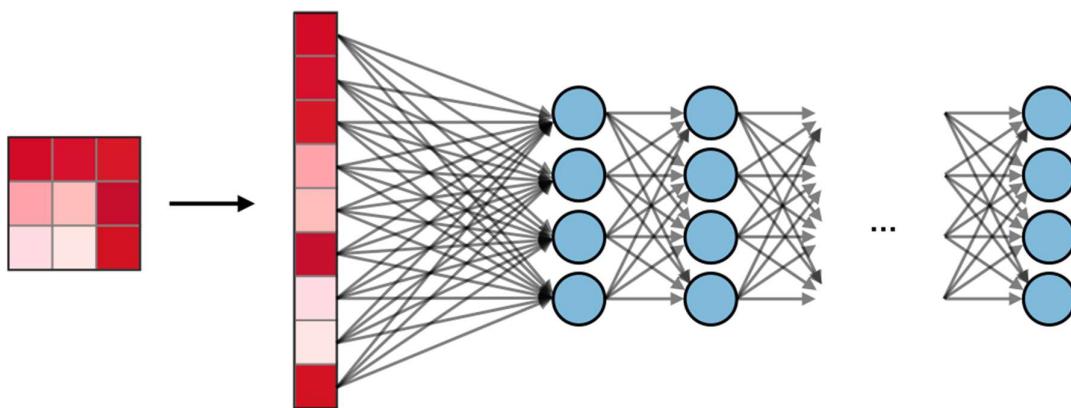
Hình 2.19. Kết quả thu được sau khi thực hiện max-pooling

#### 2.2.4. Tầng kết nối đầy đủ (Fully-Connected)

Tầng kết nối đầy đủ (FC) nhận đầu vào là các dữ liệu đã được làm phẳng, mà mỗi đầu vào đó được kết nối đến tất cả neuron. Trong mô hình mạng CNNs, các tầng kết nối đầy đủ thường được tìm thấy ở cuối mạng và được dùng để tối ưu hóa mục tiêu của mạng ví dụ như độ chính xác của lớp. Tensor trước khi đưa vào tầng Fully Connected sẽ được thực hiện bước Flattening (làm phẳng) để chuyển đổi từ dạng tensor 3D thành tensor 1D.



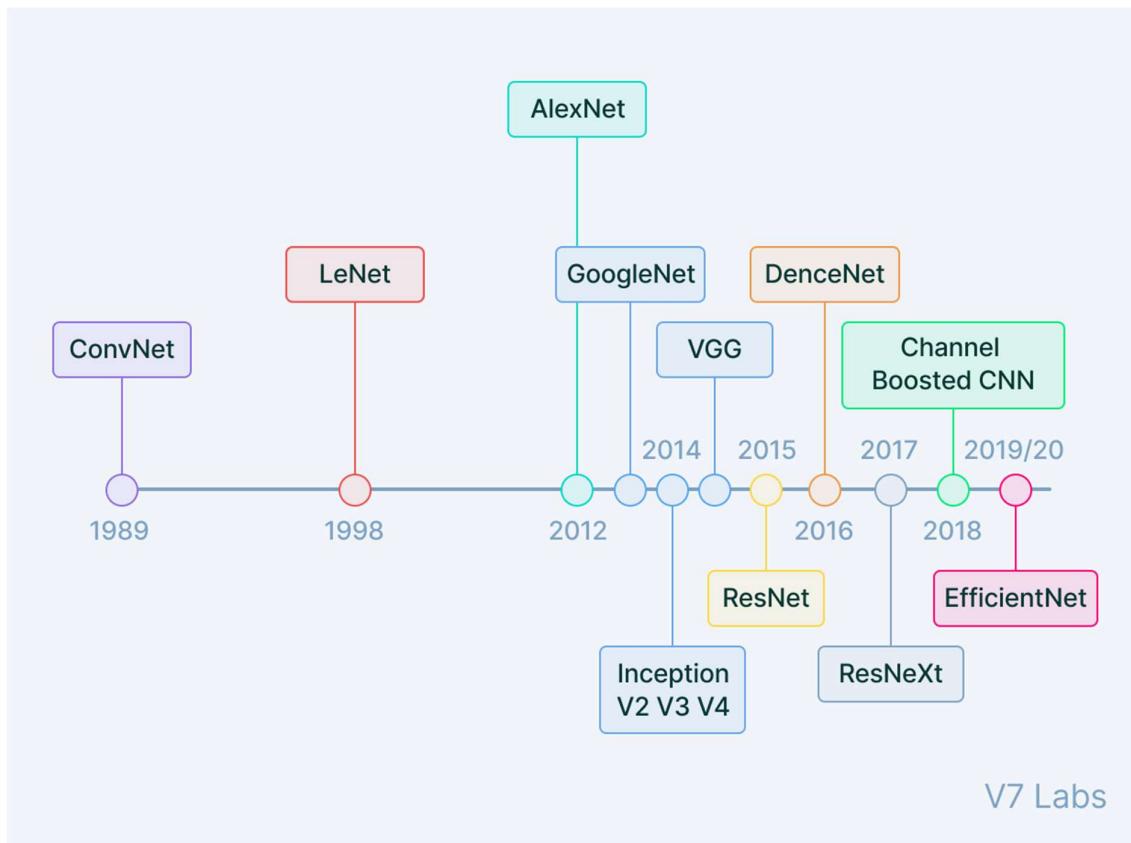
Hình 2.20. Làm phẳng tensor



Hình 2.21. Quá trình từ làm phẳng đến lúc ra kết quả

Sau khi ảnh được truyền qua nhiều convolutional layer và pooling layer thì model đã học được tương đối các đặc điểm của ảnh thì tensor của output của layer cuối cùng sẽ được là phẳng thành vector và đưa vào một lớp được kết nối như một mạng nơ-ron. Với FC layer được kết hợp với các tính năng lại với nhau để tạo ra một mô hình. Cuối cùng sử dụng softmax hoặc sigmoid để phân loại đầu ra.

### 2.3. Các kiến trúc CNN hiện đại



Hình 2.22. Timeline của kiến trúc CNN

Tiến trình phát triển của các kiến trúc CNN có thể được khái quát qua những dấu mốc quan trọng sau đây:

- 1998: Yan Lecun lần đầu tiên sử dụng mạng tích chập trong tác vụ phân loại chữ số viết tay và đạt hiệu quả cao. Tuy nhiên vào thời điểm đó do chưa có sự phát triển của dữ liệu và khả năng tính toán nên mạng CNN vẫn chưa có cơ hội bùng nổ. Các mô hình machine learning truyền thống như SVM, kNN vẫn được sử dụng phổ biến.
- 2009: Bộ dữ liệu ImageNet được giới thiệu vào năm 2009 là một trong những bộ dữ liệu tạo ra sự thay đổi trong cộng đồng computer vision. Đây là bộ dữ liệu lớn nhất so với các bộ dữ liệu từng có từ trước đến thời điểm đó. Với kích thước lên tới 1 triệu ảnh và phân bố đều trên 1000 nhãn. Các mô hình được huấn luyện trên ImageNet có thể chuyển giao tới rất nhiều những domain dữ liệu khác nhau. Kể

từ thời điểm 2010, ImageNet trở thành tiêu chuẩn đo đạc sự phát triển của các thuật toán học có giám sát trong thị giác máy tính.

- 2012: Mạng AlexNet sử dụng tích chập CNN lần đầu tiên vượt qua các phương pháp tạo đặc trưng thủ công truyền thống như HOG, SHIFT và đạt độ chính xác cách biệt trong cuộc thi ImageNet. Dấu mốc đó đã khởi đầu cho xu hướng ứng dụng CNN trong computer vision thay thế cho những thuật toán học máy truyền thống trước kia.

Liên tiếp vào những năm sau đó, ngày càng xuất hiện nhiều các kiến trúc CNN mới. Chúng được hình thành, phát triển và cải tiến về độ sâu, cách thiết kế block, cách kết nối giữa các block. Lần lượt từ VGG Net, GoogleNet, ResNet, DenseNet,... được ra đời dựa trên sự kế thừa những ý tưởng cũ và phát triển những ý tưởng mới mẽ. Quá trình phát triển của các kiến trúc mạng song hành cùng với sự phát triển phần cứng máy tính như các GPU có tốc độ nhanh hơn. Kỹ thuật huấn luyện phân tán và song song trên nhiều GPU cho phép một model huấn luyện chỉ trong vòng một vài tiếng so với việc huấn luyện kéo dài qua nhiều ngày và tốn kém như trước đây. Các framework hỗ trợ deep learning cũng xuất hiện nhiều hơn, được cải tiến và trở thành công cụ đáp ứng mọi nhu cầu cần thiết cho quá trình huấn luyện deep learning. Phổ biến nhất có thể kể tới ba frameworks pytorch (facebook), tensorflow (google), mxnet (intel) được phát triển và hậu thuẫn từ những công ty công nghệ hàng đầu thế giới.

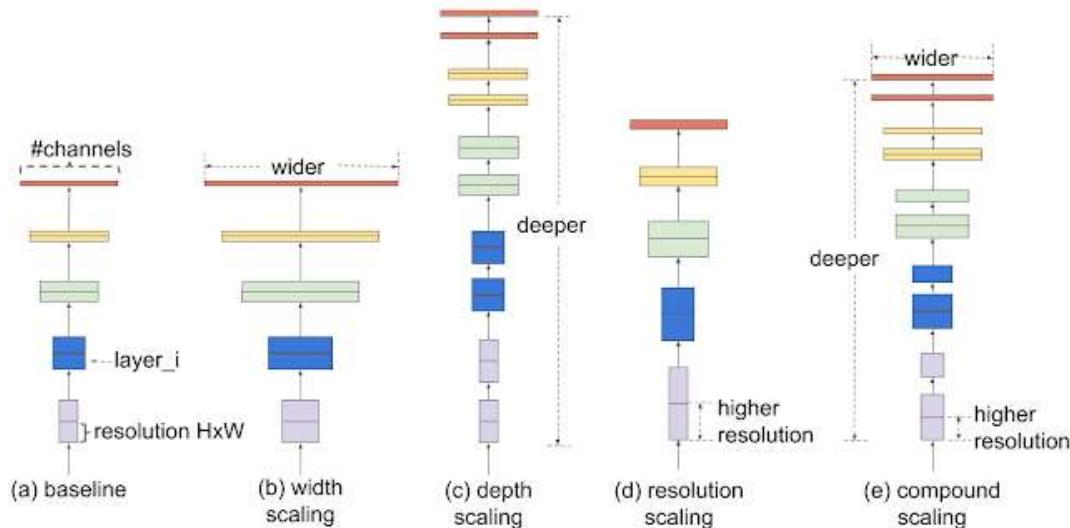
Nhờ sự mở rộng của những nền tảng huấn luyện free như google colab, kaggle mà mọi người đều có thể tiếp cận được với AI. Chiến lược phát triển toàn cầu về AI ở các tập đoàn, quốc gia trên thế giới dẫn tới sự hình thành những viện nghiên cứu về AI qui tụ được nhiều nhà khoa học xuất sắc và có những nghiên cứu đột phá.

Gần đây thì kiến trúc EfficientNet dựa trên việc tìm kiếm tối ưu trên không gian các tham số Depth, Width và Channel đã được Google phát triển và tạo ra kết quả SOTA trên bộ dữ liệu ImageNet. Vậy nên nhóm đã sử dụng kiến trúc EfficientNet cho Website phân tích ảnh của mình.

## 2.3. EfficientNet

EfficientNet, được giới thiệu lần đầu tiên ở Arxiv.org vào 5/2019 đến 9/2020 bởi Mingxing Tan và Quoc V. Le, được công bố mã nguồn mở bởi Google Brain, nằm trong những model có hiệu quả nhất (và yêu cầu số lượng FLOPS ít nhất) có tỉ lệ chính xác đạt chuẩn State-of-the-Art ở cả ImageNet và các dataset khác (CIFAR-100, Flower, Food-101). EfficientNet được cho là một kiến trúc mạng CNN hiệu quả bởi có sự kết hợp tối ưu giữa các cách cải thiện độ chính xác là: thu phóng chiều sâu, thu phóng chiều rộng và tăng giảm độ phân giải.

### 2.3.1 Khả năng thu phóng



Hình 2.23. Mạng CNN được thu phóng theo nhiều kiểu

#### Thu phóng theo chiều sâu (Depth Scaling):

Thu phóng theo chiều sâu là một cách thông dụng nhất được sử dụng để thu phóng một mô hình CNN. Độ sâu có thể được thu phóng cũng như thu nhỏ bằng cách thêm hoặc bớt các lớp tương ứng. Ví dụ: ResNets có thể được mở rộng từ ResNet-50 đến ResNet-200 cũng như chúng có thể được thu nhỏ từ ResNet-50 thành ResNet-18.

Việc sử dụng một mô hình quá phức tạp với lượng dữ liệu không tương xứng, như ta đã biết, có thể gây ra hiện tượng Overfitting. Thêm nữa, các mạng sâu hơn có xu hướng bị vanishing gradients và trở nên khó đào tạo. Vậy nên không phải lúc nào, thu phóng theo chiều sâu cũng là sự lựa chọn thích hợp để cải thiện mô hình CNN.

## Thu phóng theo chiều rộng (Width Scaling):

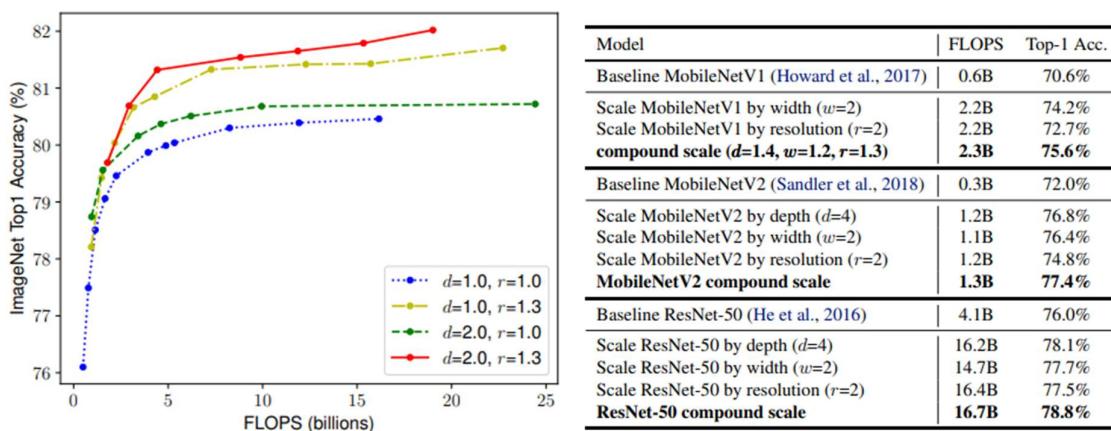
Việc thu phóng theo chiều rộng của mạng (theo như trong hình minh họa ta có thể hiểu là thêm dữ liệu đầu vào) cho phép các lớp tìm hiểu các tính năng chi tiết hơn. Khái niệm này đã được sử dụng rộng rãi trong nhiều công trình như Wide ResNet và Mobile Net. Tuy nhiên, cũng như trường hợp tăng chiều sâu, tăng chiều rộng ngăn cản mạng học các tính năng phức tạp, dẫn đến giảm độ chính xác.

## Thu phóng theo độ phân giải (Resolution Scaling)

Độ phân giải đầu vào cao hơn cung cấp hình ảnh chi tiết hơn và do đó nâng cao khả năng suy luận của mô hình về các đối tượng nhỏ hơn và trích xuất các mẫu mịn hơn. Tuy nhiên cũng như các cách thu phóng trên, việc chỉ thu phóng theo độ phân giải không hề luôn luôn hiệu quả trong mọi trường hợp mà thậm chí nó còn có thể giảm độ chính xác của mô hình đi một cách nhanh chóng.

## Kết hợp thu phóng theo chuẩn (Compound Scaling)

Các nhà nghiên cứu của EfficientNet đã nhận ra việc thu phóng để tăng độ chính xác chỉ hoạt động từ 1 phía nên theo trực giác của bản thân, họ đã thử phối hợp và cân bằng kích thước của việc thu phóng theo các tỉ lệ khác nhau. Từ đó họ đã cho ra một vài thử nghiệm về việc thu phóng theo chiều rộng và theo các độ sâu cũng như độ phân giải mạng ở các mức độ khác nhau.



Hình 2.24. Kết quả sau khi thực hiện các hoạt động thu phóng

Từ kết quả trên, họ đã đi đến kết luận: Để đạt được độ chính xác và hiệu quả tốt hơn, điều quan trọng là phải cân bằng tất cả các kích thước của chiều rộng, chiều sâu và độ phân giải mạng trong quá trình thu phóng quy mô của mạng CNN. Đồng thời, họ cũng đưa ra được công thức để tối ưu việc compound scaling này.

$$\begin{aligned} d &= \alpha^\varphi & w &= \beta^\varphi & r &= \gamma^\varphi \\ \alpha \cdot \beta^2 \cdot \gamma^2 &\approx 2 & \alpha \geq 1, \beta \geq 1, \gamma \geq 1 \end{aligned}$$

Trong đó :

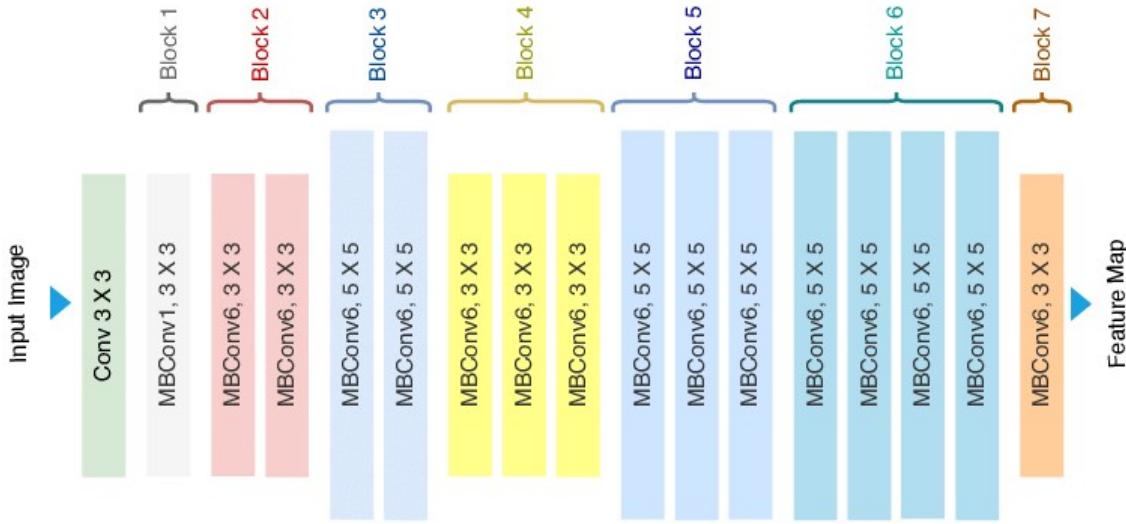
- $d, w, r$  lần lượt là độ rộng, độ sâu và độ phân giải của mạng
- $\alpha, \beta, \gamma$  là các hằng số có thể được xác định bằng small grid search.

Theo công thức trên,  $\varphi$  là hệ số do người dùng chỉ định để kiểm soát số lượng tài nguyên khác có sẵn để thu phóng mô hình, trong khi  $\alpha, \beta, \gamma$  chỉ định cách gán các tài nguyên bổ sung này cho độ rộng, độ sâu và độ phân giải của mạng tương ứng. Do công thức sẽ làm tăng FLOPS (Floating-point Operations Per Second) là một thước đo hiệu suất máy tính, các tính toán thập phân trong một giây) đặc biệt khi tăng gấp đôi  $w$  hoặc  $r$  thì FLOPS sẽ tăng gấp bốn lần. Vì các hoạt động tích phân thường chiếm ưu thế trong chi phí tính toán trong ConvNets, nên với phương trình 3 sẽ làm tăng tổng FLOPS khoảng  $(\alpha \cdot \beta^2 \cdot \gamma^2)^\varphi$ . Vì vậy các tác giả đã ràng buộc  $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$  sao cho với bất kỳ  $\varphi$  nào, tổng FLOPS cũng sẽ tăng gấp  $2^\varphi$ .

### 2.3.2. Các loại Model của EfficientNet

Mô hình cơ bản nhỏ nhất của EfficientNet tương đồng với MnasNet, một mô hình đạt đến gần SOTA (State-of-the-Art) với kích thước mô hình đáng kể nhỏ hơn. Bằng cách giới thiệu một cách hợp lý để mở rộng kích thước của mô hình, EfficientNet cung cấp một họ mô hình (từ B0 đến B7) phù hợp nhất cho sự kết hợp tốt giữa hiệu suất và độ chính xác trên nhiều tỷ lệ khác nhau.

Bắt đầu với EfficientNet-B0, các tác giả đã thiết kế và dựa vào đó tăng  $\varphi$  để tạo ra các model phù hợp với kích thước lớn hơn.



Hình 2.25. Cấu trúc mạng EfficientNet-B0

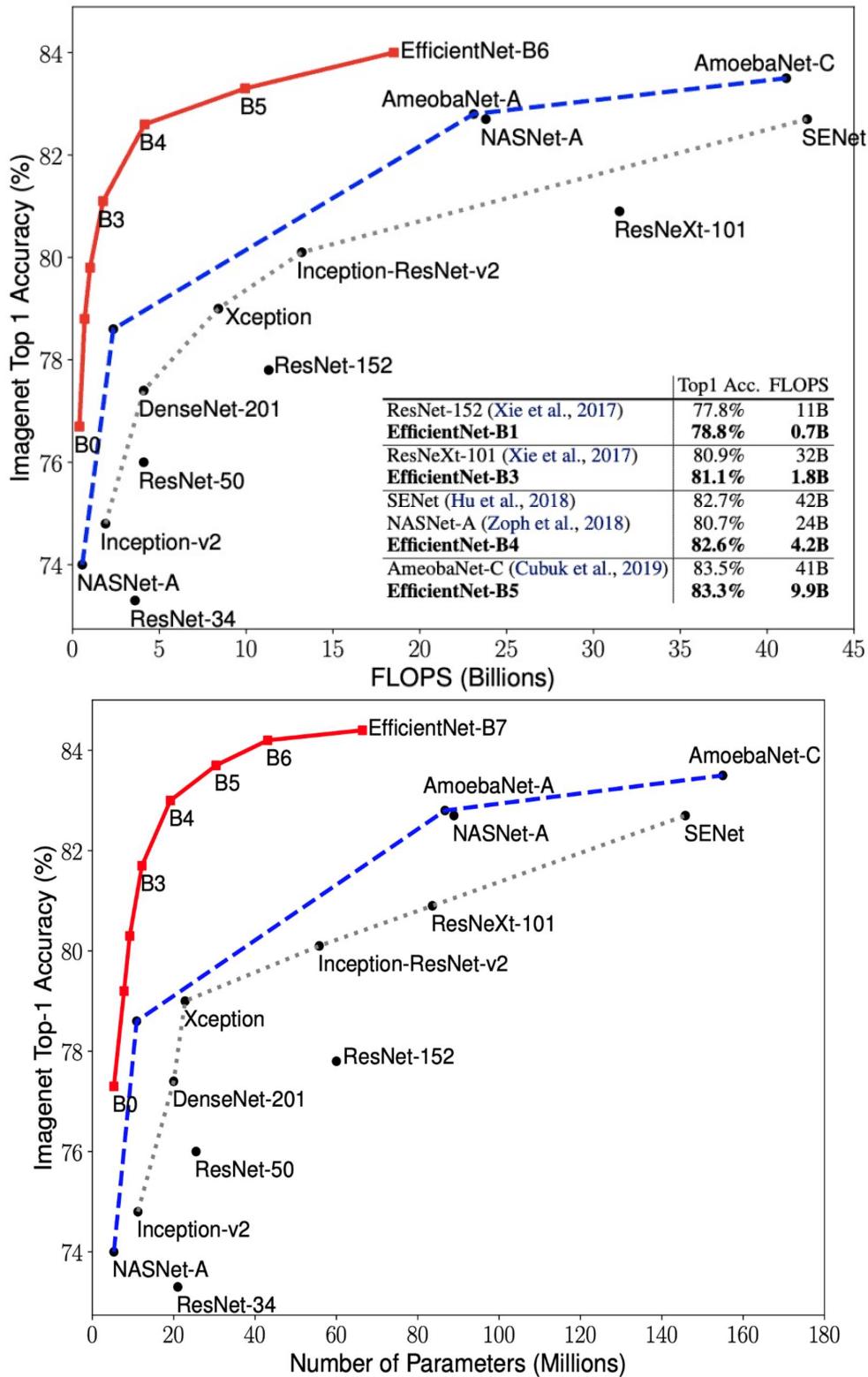
Stage $i$	Operator $\hat{F}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	$224 \times 224$	32	1
2	MBCovn1, k3x3	$112 \times 112$	16	1
3	MBCovn6, k3x3	$112 \times 112$	24	2
4	MBCovn6, k5x5	$56 \times 56$	40	2
5	MBCovn6, k3x3	$28 \times 28$	80	3
6	MBCovn6, k5x5	$14 \times 14$	112	3
7	MBCovn6, k5x5	$14 \times 14$	192	4
8	MBCovn6, k3x3	$7 \times 7$	320	1
9	Conv1x1 & Pooling & FC	$7 \times 7$	1280	1

Hình 2.26. Thông số mạng EfficientNet-B0

Bắt đầu với B0, họ đã áp dụng công thức đã được thực hiện từ trước:

- Đặt  $\varphi = 1$ , họ thu được bộ giá trị tối ưu  $\alpha = 1,2$ ;  $\beta = 1,1$ ;  $\gamma = 1,15$ , theo ràng buộc  $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$
- Cố định  $\alpha$ ;  $\beta$ ;  $\gamma$  dưới dạng các hằng số và thu phóng bằng cách thay đổi  $\varphi$  từ đó thu được các mạng từ EfficientNet-B1 đến Efficient-B7.

### 2.3.3. Độ hiệu quả của EfficientNet



Hình 2.27. So sánh với các cấu trúc mạng khác theo FLOPS, Parameter

Ta có thể dễ dàng nhận ra, với cùng số lượng FLOPS hoặc Parameter, EfficientNet luôn đạt được độ chính xác cao hơn các cấu trúc mạng tiền nhiệm. Những lợi ích này đến từ cả kiến trúc tốt hơn, thu phóng quy mô tốt hơn và cài đặt đào tạo tốt hơn được tùy chỉnh cho EfficientNet.

Model	Top-1 Acc.	Top-5 Acc.	#Params	Ratio-to-EfficientNet	#FLOPs	Ratio-to-EfficientNet
<b>EfficientNet-B0</b>	<b>77.1%</b>	<b>93.3%</b>	<b>5.3M</b>	<b>1x</b>	<b>0.39B</b>	<b>1x</b>
ResNet-50 (He et al., 2016)	76.0%	93.0%	26M	4.9x	4.1B	11x
DenseNet-169 (Huang et al., 2017)	76.2%	93.2%	14M	2.6x	3.5B	8.9x
<b>EfficientNet-B1</b>	<b>79.1%</b>	<b>94.4%</b>	<b>7.8M</b>	<b>1x</b>	<b>0.70B</b>	<b>1x</b>
ResNet-152 (He et al., 2016)	77.8%	93.8%	60M	7.6x	11B	16x
DenseNet-264 (Huang et al., 2017)	77.9%	93.9%	34M	4.3x	6.0B	8.6x
Inception-v3 (Szegedy et al., 2016)	78.8%	94.4%	24M	3.0x	5.7B	8.1x
Xception (Chollet, 2017)	79.0%	94.5%	23M	3.0x	8.4B	12x
<b>EfficientNet-B2</b>	<b>80.1%</b>	<b>94.9%</b>	<b>9.2M</b>	<b>1x</b>	<b>1.0B</b>	<b>1x</b>
Inception-v4 (Szegedy et al., 2017)	80.0%	95.0%	48M	5.2x	13B	13x
Inception-resnet-v2 (Szegedy et al., 2017)	80.1%	95.1%	56M	6.1x	13B	13x
<b>EfficientNet-B3</b>	<b>81.6%</b>	<b>95.7%</b>	<b>12M</b>	<b>1x</b>	<b>1.8B</b>	<b>1x</b>
ResNeXt-101 (Xie et al., 2017)	80.9%	95.6%	84M	7.0x	32B	18x
PolyNet (Zhang et al., 2017)	81.3%	95.8%	92M	7.7x	35B	19x
<b>EfficientNet-B4</b>	<b>82.9%</b>	<b>96.4%</b>	<b>19M</b>	<b>1x</b>	<b>4.2B</b>	<b>1x</b>
SENet (Hu et al., 2018)	82.7%	96.2%	146M	7.7x	42B	10x
NASNet-A (Zoph et al., 2018)	82.7%	96.2%	89M	4.7x	24B	5.7x
AmoebaNet-A (Real et al., 2019)	82.8%	96.1%	87M	4.6x	23B	5.5x
PNASNet (Liu et al., 2018)	82.9%	96.2%	86M	4.5x	23B	6.0x
<b>EfficientNet-B5</b>	<b>83.6%</b>	<b>96.7%</b>	<b>30M</b>	<b>1x</b>	<b>9.9B</b>	<b>1x</b>
AmoebaNet-C (Cubuk et al., 2019)	83.5%	96.5%	155M	5.2x	41B	4.1x
<b>EfficientNet-B6</b>	<b>84.0%</b>	<b>96.8%</b>	<b>43M</b>	<b>1x</b>	<b>19B</b>	<b>1x</b>
<b>EfficientNet-B7</b>	<b>84.3%</b>	<b>97.0%</b>	<b>66M</b>	<b>1x</b>	<b>37B</b>	<b>1x</b>
GPipe (Huang et al., 2018)	84.3%	97.0%	557M	8.4x	-	-

Hình 2.28. So sánh với các Model phổ biến

Model EfficientNet-B7 Top-1 Acc là 84.3% với số parameter 66 M và FLOPS 37 B, nhỏ hơn 8.4 lần về Parameter và nhanh hơn 6.1 lần so với model GPipe-AmoebaNet-B Top-1 Acc là 84.3%.

Model EfficientNet-B1 nhỏ hơn 7.6 lần về số Parameter, nhanh hơn 5.7 lần và độ chính xác gần như tương đương 78.8% (Top1 Acc) so với model ResNet-152 Top1 Acc 77.8%.

So với model ResNet-50 được nhiều người sử dụng, EfficientNet-B4 có độ chính xác top-1 82.9% cao hơn hẳn so với ResNet-50 là 76% với FLOPS gần như tương đương.

	Model	Comparison to best public-available results							Model	Comparison to best reported results					
		Acc.	#Param	Our Model	Acc.	#Param	(ratio)			Acc.	#Param	Our Model	Acc.	#Param	(ratio)
CIFAR-10	NASNet-A	98.0%	85M	EfficientNet-B0	98.1%	4M (21x)		<sup>†</sup> Gpipe	<b>99.0%</b>	556M	EfficientNet-B7	98.9%	64M (8.7x)		
CIFAR-100	NASNet-A	87.5%	85M	EfficientNet-B0	88.1%	4M (21x)		Gpipe	91.3%	556M	EfficientNet-B7	<b>91.7%</b>	64M (8.7x)		
Birdsnap	Inception-v4	81.8%	41M	EfficientNet-B5	82.0%	28M (1.5x)		GPipe	83.6%	556M	EfficientNet-B7	<b>84.3%</b>	64M (8.7x)		
Stanford Cars	Inception-v4	93.4%	41M	EfficientNet-B3	93.6%	10M (4.1x)		<sup>†</sup> DAT	<b>94.8%</b>	-	EfficientNet-B7	94.7%	-		
Flowers	Inception-v4	98.5%	41M	EfficientNet-B5	98.5%	28M (1.5x)		DAT	97.7%	-	EfficientNet-B7	<b>98.8%</b>	-		
FGVC Aircraft	Inception-v4	90.9%	41M	EfficientNet-B3	90.7%	10M (4.1x)		DAT	92.9%	-	EfficientNet-B7	<b>92.9%</b>	-		
Oxford-IIIT Pets	ResNet-152	94.5%	58M	EfficientNet-B4	94.8%	17M (5.6x)		GPipe	<b>95.9%</b>	556M	EfficientNet-B6	95.4%	41M (14x)		
Food-101	Inception-v4	90.8%	41M	EfficientNet-B4	91.5%	17M (2.4x)		GPipe	93.0%	556M	EfficientNet-B7	<b>93.0%</b>	64M (8.7x)		
Geo-Mean							(4.7x)								(9.6x)

Hình 2.29. So sánh Model với các Dataset khác nhau

Model EfficientNet đạt được thành tựu *new-state-of-the-art* ở 5 trên 8 datasets (CIFAR-100 (91.7%), Flowers (98.8%),...,), với số lượng tham số (Parameters) ít hơn 9.6 lần so với các model khác

View		Top 1 Accuracy	All models							Edit		
RANK	MODEL	TOP 1 ACCURACY	TOP 5 ACCURACY	NUMBER OF PARAMS	EXTRA TRAINING DATA	PAPER	CODE	RESULT	YEAR			
1	FixEfficientNet-L2	88.5%	98.7%	480M	✓	Fixing the train-test resolution discrepancy: FixEfficientNet	<a href="#">Code</a>	<a href="#">Result</a>	2020			
2	NoisyStudent (EfficientNet-L2)	88.4%	98.7%	480M	✓	Self-training with Noisy Student improves ImageNet classification	<a href="#">Code</a>	<a href="#">Result</a>	2020			
3	BiT-L (ResNet)	87.54%	98.46%		✓	Big Transfer (BiT): General Visual Representation Learning	<a href="#">Code</a>	<a href="#">Result</a>	2019			
4	FixEfficientNet-B7	87.1%	98.2%	66M	✓	Fixing the train-test resolution discrepancy: FixEfficientNet	<a href="#">Code</a>	<a href="#">Result</a>	2020			
5	NoisyStudent (EfficientNet-B7)	86.9%	98.1%	66M	✓	Self-training with Noisy Student improves ImageNet classification	<a href="#">Code</a>	<a href="#">Result</a>	2019			

Hình 2.30. Đạt top trên the ImageNet task

	B0	B1	B2	B3	B4	B5	B6	B7
Val top1	77.11	79.13	80.07	81.59	82.89	83.60	83.95	84.26
Test top1	77.23	79.17	80.16	81.72	82.94	83.69	84.04	84.33
Val top5	93.35	94.47	94.90	95.67	96.37	96.71	96.76	96.97
Test top5	93.45	94.43	94.98	95.70	96.27	96.64	96.86	96.94

Hình 2.31. Các kết quả test trên ImageNet dataset

Kết quả test trên dataset ImageNet, là bộ dataset hình ảnh phổ biến được sử dụng trong nghiên cứu phát hiện vật thể trong ảnh (object detection). Dataset có hơn 14 triệu ảnh được gắn nhãn, khoảng 1 triệu trong số đó có khoanh vùng vật thể (bounding boxes). ImageNet có hơn 20,000 nhãn, mỗi nhãn có ít nhất vài trăm ảnh. Và đó cũng là dataset nhóm dùng để thực hiện website phân tích ảnh của mình.

## CHƯƠNG 3: TRIỂN KHAI CHƯƠNG TRÌNH

### 3.1. Các môi trường và công cụ sử dụng

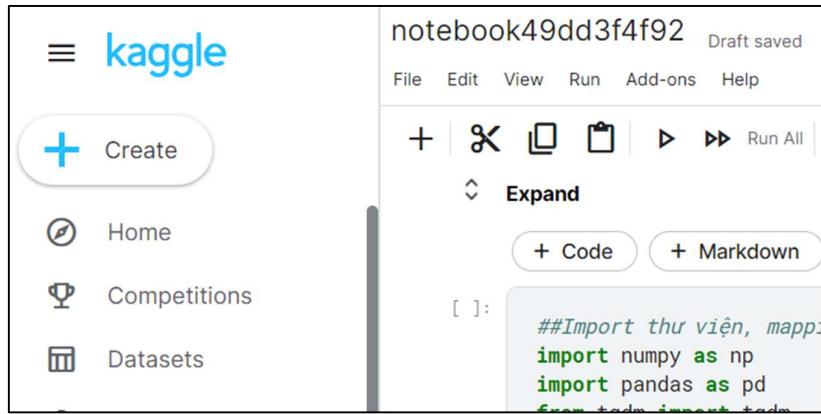
- Việc huấn luyện (training) mạng CNN đòi hỏi tài nguyên tính toán lớn và nhiều dữ liệu. Hiện nay có rất nhiều dịch vụ cung cấp môi trường để người dùng có nhiều lựa chọn tùy theo mục đích, quy mô và nhu cầu của người dùng. Đây là một số lựa chọn cho người dùng:
  - Google Colab:
    - Ưu điểm: Miễn phí sử dụng với một số tài nguyên hạn chế. Dễ sử dụng, chạy trực tiếp trên trình duyệt web. Có sẵn các thư viện hỗ trợ. Cung cấp GPU miễn phí để tăng tốc quá trình huấn luyện.
    - Nhược điểm: Hạn chế thời gian sử dụng tài nguyên GPU miễn phí mỗi lần. Không phù hợp cho các dự án yêu cầu tài nguyên lớn hoặc thời gian lâu dài.
  - Kaggle:
    - Ưu điểm: Cung cấp môi trường notebook và tài nguyên GPU mạnh mẽ. Hỗ trợ tải lên và lưu trữ dữ liệu trực tuyến. Cộng đồng lớn, có thể chia sẻ và thảo luận dự án với cộng đồng Kaggle.
    - Nhược điểm: Có thể giới hạn về thời gian sử dụng tài nguyên GPU. Đôi khi giới hạn về dung lượng lưu trữ.
  - AWS (Amazon Web Services):
    - Ưu điểm: Cung cấp nhiều tùy chọn về cấu hình máy ảo và GPU. Linh hoạt về tài nguyên và thời gian sử dụng. Dùng được cho các dự án lớn và lâu dài.
    - Nhược điểm: Yêu cầu kiến thức về quản lý tài nguyên trên AWS. Phải trả phí dựa trên tài nguyên sử dụng.
  - Microsoft Azure:
    - Ưu điểm: Cung cấp các tài nguyên mạnh mẽ và linh hoạt. Hỗ trợ nhiều ngôn ngữ và framework. Có các dịch vụ hỗ trợ machine learning và deep learning.

- Nhược điểm: Cần kiến thức về quản lý tài nguyên trên Azure. Yêu cầu thanh toán dựa trên sử dụng.
- Nếu cấu hình máy tính cá nhân của bạn đủ mạnh thì có thể huấn luyện mô hình CNN trực tiếp trên máy cá nhân của mình.
- Trong đề tài này nhóm sử dụng dịch vụ do Kaggle cung cấp để training model, Kaggle notebooks sử dụng môi trường Jupyter để thực thi mã Python. Người dùng có thể viết và chạy mã trong các ô mã (code cells). Đôi với đề tài này thì lượng dataset lớn cho nên dùng Kaggle có thể tiết kiệm được tài nguyên lưu trữ vì có sẵn các data có thể tải lên và lưu trữ dữ liệu trực tuyến. Kaggle cũng hỗ trợ các cấu hình đủ mạnh (GPU T4 x2, GPU P100, TPU VM v3-8 ...) và miễn phí trong thời gian giới hạn nên nhóm không cần phải trả phí để mua hay thuê các server.
- Visual Studio Code (VS Code) để triển khai website. VS Code là một trình soạn thảo mã nguồn mở và miễn phí được phát triển bởi Microsoft. Được phát hành lần đầu tiên vào năm 2015, VS Code nhanh chóng trở thành một trong những trình soạn thảo mã phổ biến nhất trong cộng đồng phát triển phần mềm. VS Code được thiết kế để hỗ trợ nhiều ngôn ngữ lập trình và kịch bản khác nhau, với sự tích hợp mạnh mẽ cho các công cụ như TypeScript, JavaScript, HTML, CSS, Python, Java, và nhiều ngôn ngữ khác. Điều này giúp cho các nhà phát triển làm việc trên các dự án đa ngôn ngữ một cách dễ dàng.

## 3.2. Huấn luyện Model (Training Model)

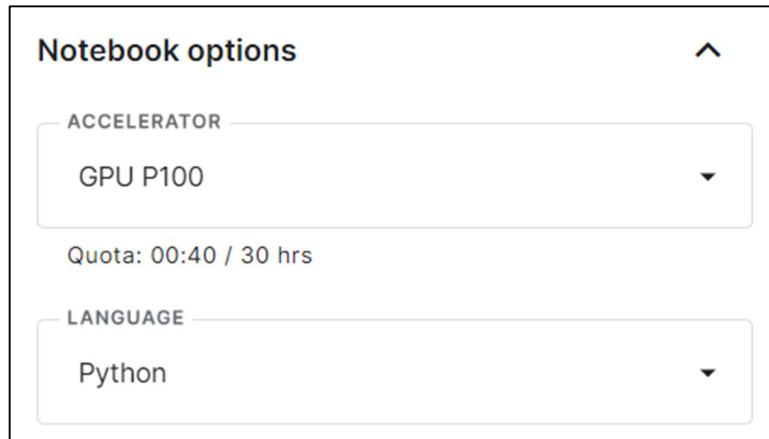
### 3.2.1. Các bước thực hiện

- Truy cập vào “<https://www.kaggle.com/>” tạo mới một notebook trong kaggle, nhấp vào nút create dưới logo Kaggle



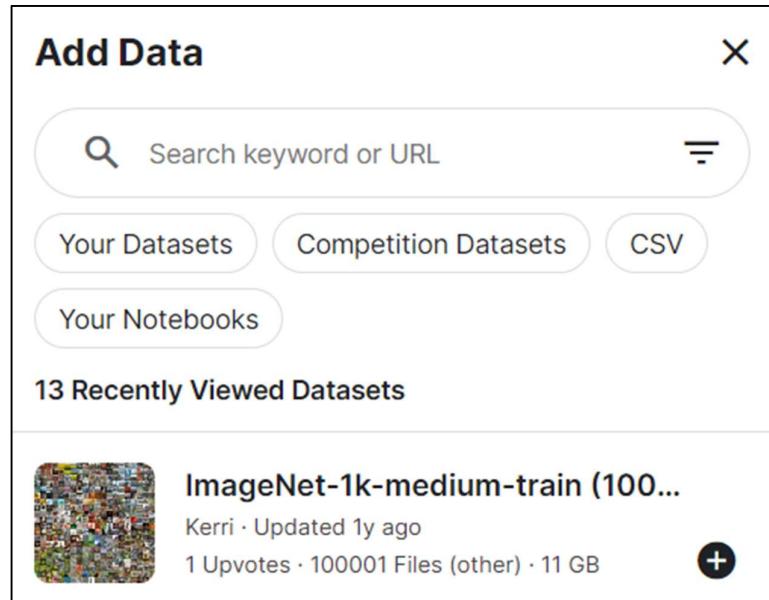
Hình 3.1. Trang chủ kaggle

- Khởi động một session và lựa chọn cấu hình để training. Tài nguyên này bị giới hạn trong vòng 30 tiếng và sẽ tái sử dụng được sau 1 tuần.



Hình 3.2. Setting notebook

- Kaggle có hỗ trợ tải lên và lưu trữ dữ liệu trực tuyến hoặc dùng dataset có sẵn trên nền tảng. Tìm kiếm một dataset phù hợp với để tải và add data vào để triển khai training.



Hình 3.3. Chọn dataset trên kaggle

- Dataset này gồm 1000 class ứng với 100000 hình ảnh của các loài vật và đồ vật. Tổng khối lượng của dataset này khoảng 11GB.

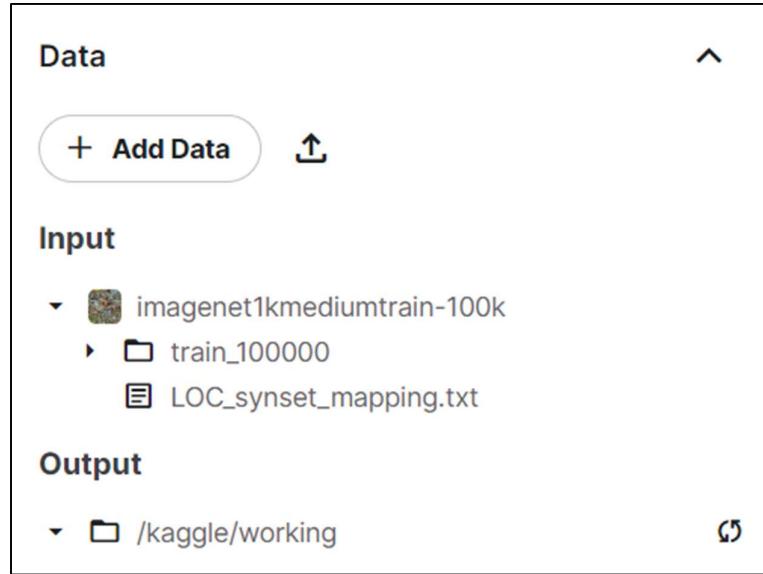
File	Size
LOC_synset_mapping.txt	31.68 kB

```

n01440764 tench, Tinca tinca
n01443537 goldfish, Carassius auratus
n01484858 great white shark, white shark, man-eater, man-eating shark, Carcharodon carcharias
n01491361 tiger shark, Galeocerdo cuvieri
n01494475 hammerhead, hammerhead shark
n01496331 electric ray, crampfish, numbfish, torpedo
n01498041 stingray
n01514668 cock
n01514859 hen
n01518878 ostrich, Struthio camelus
n01530575 brambling, Fringilla montifringilla
n01531178 goldfinch, Carduelis carduelis
n01532829 house finch, linnet, Carpodacus mexicanus
n01534433 junco, snowbird
n01537544 indigo bunting, indigo finch, indigo bird, Passerina cyanea
n01558993 robin, American robin, Turdus migratorius
n01560419 bulbul
n01580077 jay
n01582220 magpie
n01592084 chickadee

```

Hình 3.4. Dataset sau khi được chọn



Hình 3.5. Input và Output của notebook

### 3.2.2. Training model

- Viết code vào trong ô mã (code cells). Nhập vào (Import) các thư viện cần sử dụng.

```
[ ]: ##Import thư viện, mappingpath
import numpy as np
import pandas as pd
from tqdm import tqdm
import os
from matplotlib import pyplot as plt
import random
import cv2
from glob import glob
import tensorflow as tf
import keras
from keras.models import *
from keras.layers import *
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.optimizers import Adamax
from keras.applications.efficientnet import preprocess_input
from keras.preprocessing.image import ImageDataGenerator, load_img, img_to_array, array_to_img
from keras.applications.efficientnet import EfficientNetB7
from IPython.display import display
from PIL import Image
```

Hình 3.6. Thư viện sử dụng để training

- Khai báo các đường dẫn và batchsize.

```
[ ]:
#PathFile

mapping_path = '/kaggle/input/imagenet1kmediumtrain-100k/LOC_synset_mapping.txt'
src_path_train = "/kaggle/input/imagenet1kmediumtrain-100k/train_100000/train_100000"
batch_size = 128
```

*Hình 3.7. Lưu đường dẫn từ input vào để sử dụng*

- Trích xuất dữ liệu từ file .txt để lấy ra mã và tên của các class.

```
[ ]:
#ReadTXTfile
class_mapping_dict = {}
i = 0
for line in open(mapping_path):
    if 0 <= i < 1000:
        label = line[9:].strip()
        class_mapping_dict[i] = label
    i += 1
```

*Hình 3.8. Mapping file .txt để lấy ra tên class*

- Xử lý ảnh và tạo ra hai luồng dữ liệu hình ảnh, một để huấn luyện mô hình (train\_generator) và một để kiểm tra mô hình (test\_generator) từ dữ liệu trong thư mục src\_path\_train. Các hình ảnh sẽ được chia thành hai phần theo tỉ lệ 80-20 cho huấn luyện và kiểm tra.

```
► #Image

image_gen = ImageDataGenerator(
    horizontal_flip=True,
    preprocessing_function = preprocess_input,
    validation_split=0.20)

train_generator = image_gen.flow_from_directory(
    src_path_train,
    target_size=(300,300),
    batch_size=batch_size,
    subset="training",
)

test_generator = image_gen.flow_from_directory(
    src_path_train,
    target_size=(300,300),
    batch_size=batch_size,
    subset="validation",
)
```

*Hình 3.9. Generate ảnh để tiến hành training*

- Sử dụng thư viện Keras để xây dựng một mô hình mạng nơ-ron sử dụng kiến trúc EfficientNetB7 và thêm một số lớp trên đỉnh để tạo ra một mô hình đầy đủ và cài đặt các thông số cần thiết cho quá trình huấn luyện.

```
#Tạo model
model_efficientnet = EfficientNetB7(
    weights='imagenet',
    include_top=False,
    input_shape=(300, 300, 3),
    pooling="max"
)

# model_efficientnet.trainable = True
for layer in model_efficientnet.layers:
    layer.trainable = False

x = model_efficientnet.output
x = BatchNormalization()(x)
x = Dense(1024, activation="relu")(x)
x = Dropout(0.3)(x)
x = Dense(512, activation="relu")(x)
x = Dropout(0.3)(x)
x = Dense(128, activation="relu")(x)
x = Dropout(0.3)(x)
outputs = Dense(len(class_mapping_dict), activation="softmax")(x)

model = Model(inputs=model_efficientnet.input, outputs=outputs)
lr = 0.001

model.compile(optimizer=Adamax(learning_rate=lr), loss="categorical_crossentropy", metrics=["accuracy"])
```

*Hình 3.10. Tạo model và tiến hành đặt các lớp*

- Sử dụng mô hình được định nghĩa trước đó và dữ liệu từ “train\_generator” để huấn luyện mô hình. Sử dụng dữ liệu từ test\_generator để kiểm tra hiệu suất của mô hình sau mỗi epoch. Sử dụng callback “ModelCheckpoint” để lưu lại trọng số của mô hình tại mỗi epoch nếu thấy độ chính xác tăng trên tập validation (val\_accuracy). Biến “hist” được sử dụng để lưu lại thông tin về quá trình huấn luyện của mô hình.

```
[ ]: # Train model

from tensorflow.keras.callbacks import ModelCheckpoint
ckpoint = ModelCheckpoint("/kaggle/working/model_ImagenetB7.h5", monitor="val_accuracy", save_best_only=True, mode="auto")

n_epochs = 12
hist = model.fit(
    train_generator,
    epochs=n_epochs,
    validation_data=test_generator,
    steps_per_epoch=len(train_generator),
    validation_steps=len(test_generator),
    callbacks=[ckpoint]
)
```

*Hình 3.11. Training model*

```

625/625 [=====] - 1361s 2s/step - loss: 3.8503 - accuracy: 0.3437 - val_loss: 0.7808 - val_accuracy: 0.8470
Epoch 2/12
625/625 [=====] - 1272s 2s/step - loss: 1.5766 - accuracy: 0.6782 - val_loss: 0.6072 - val_accuracy: 0.8672
Epoch 3/12
625/625 [=====] - 1272s 2s/step - loss: 1.2185 - accuracy: 0.7434 - val_loss: 0.5721 - val_accuracy: 0.8733
Epoch 4/12
625/625 [=====] - 1273s 2s/step - loss: 1.0489 - accuracy: 0.7728 - val_loss: 0.5574 - val_accuracy: 0.8752
Epoch 5/12
625/625 [=====] - 1272s 2s/step - loss: 0.9390 - accuracy: 0.7901 - val_loss: 0.5401 - val_accuracy: 0.8797
Epoch 6/12
625/625 [=====] - 1270s 2s/step - loss: 0.8594 - accuracy: 0.8049 - val_loss: 0.5374 - val_accuracy: 0.8784
Epoch 7/12
625/625 [=====] - 1272s 2s/step - loss: 0.7969 - accuracy: 0.8171 - val_loss: 0.5255 - val_accuracy: 0.8832
Epoch 8/12
625/625 [=====] - 1272s 2s/step - loss: 0.7402 - accuracy: 0.8265 - val_loss: 0.5247 - val_accuracy: 0.8842
Epoch 9/12
625/625 [=====] - 1270s 2s/step - loss: 0.6909 - accuracy: 0.8346 - val_loss: 0.5254 - val_accuracy: 0.8816
Epoch 10/12
625/625 [=====] - 1272s 2s/step - loss: 0.6535 - accuracy: 0.8418 - val_loss: 0.5216 - val_accuracy: 0.8856
Epoch 11/12
625/625 [=====] - 1270s 2s/step - loss: 0.6239 - accuracy: 0.8473 - val_loss: 0.5222 - val_accuracy: 0.8849
Epoch 12/12
625/625 [=====] - 1270s 2s/step - loss: 0.5870 - accuracy: 0.8525 - val_loss: 0.5291 - val_accuracy: 0.8844

```

Hình 3.12. Thông tin của từng epoch

- Sử dụng thông tin được lưu trong biến “hist” (history) và thư viện plt (plot) để vẽ đồ thị biểu diễn sự phát triển của độ chính xác và mất mát trên cả tập huấn luyện và tập kiểm tra qua các epochs.

```

> accuracy = hist.history['accuracy']
val_accuracy = hist.history['val_accuracy']

loss = hist.history['loss']
val_loss = hist.history['val_loss']

epochs = range(len(accuracy))
plt.plot(epochs, accuracy, 'b', label='Training accuracy')
plt.plot(epochs, val_accuracy, 'r', label='Validation accuracy')

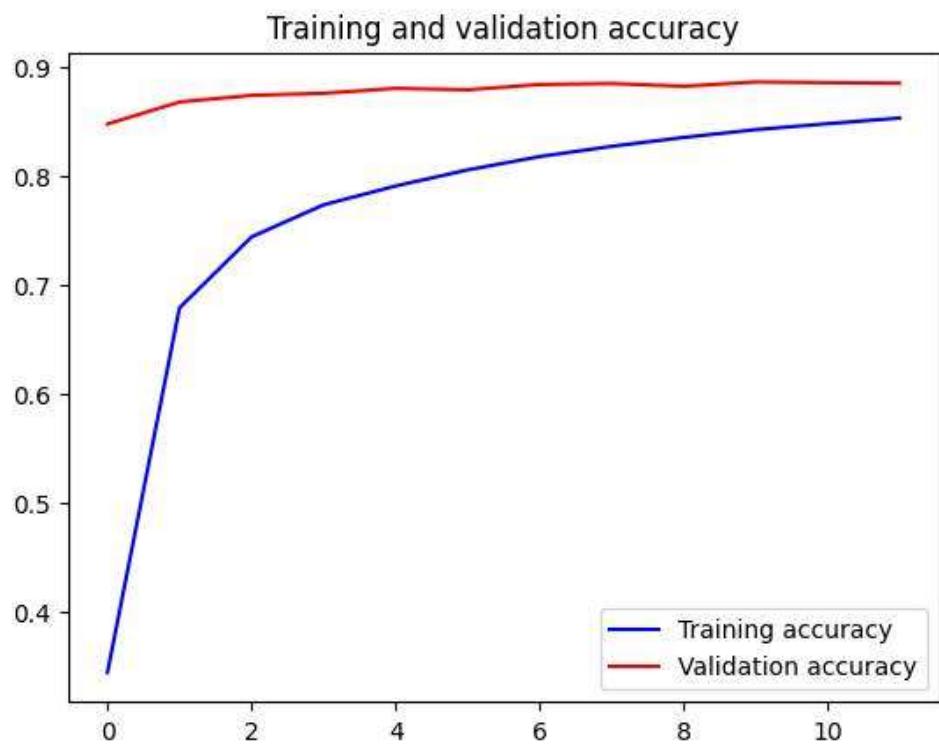
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()

plt.plot(epochs, loss, 'b', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')

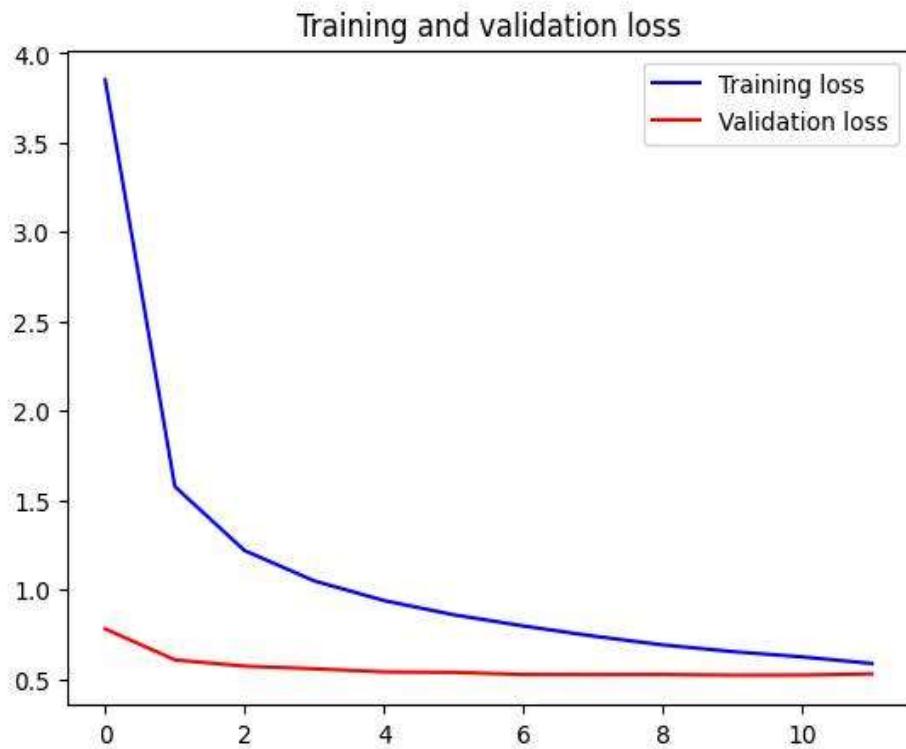
plt.title('Training and validation loss')
plt.legend()
plt.show()

```

Hình 3.13. Hiển thị biểu đồ sau khi training



Hình 3.14. Biểu diễn sự thay đổi về accuracy trong lúc train



Hình 3.15. Biểu diễn sự thay đổi về loss trong lúc train

- Dựa trên biểu đồ training model và kết quả số liệu cụ thể khi training cho thấy sự tiến triển tích cực trong quá trình huấn luyện. Dưới đây là một số đánh giá chung:

- Loss Function: Loss trên tập huấn luyện giảm đều dần qua các epoch, từ 3.8503 giảm xuống còn 0.5870. Loss trên tập validation cũng giảm, chỉ có một số biến động nhỏ.
- Accuracy (Độ chính xác): Accuracy trên tập huấn luyện tăng dần, từ 0.3437 lên đến 0.8525. Accuracy trên tập validation cũng tăng, từ 0.8470 lên đến 0.8844.
- Overfitting: Không có dấu hiệu rõ ràng của overfitting do accuracy trên tập validation tiếp tục tăng.

- Đánh giá hiệu suất của mô hình trên luồng để kiểm tra là “test\_generator” và in ra kết quả mất mát (loss) và độ chính xác (accuracy)

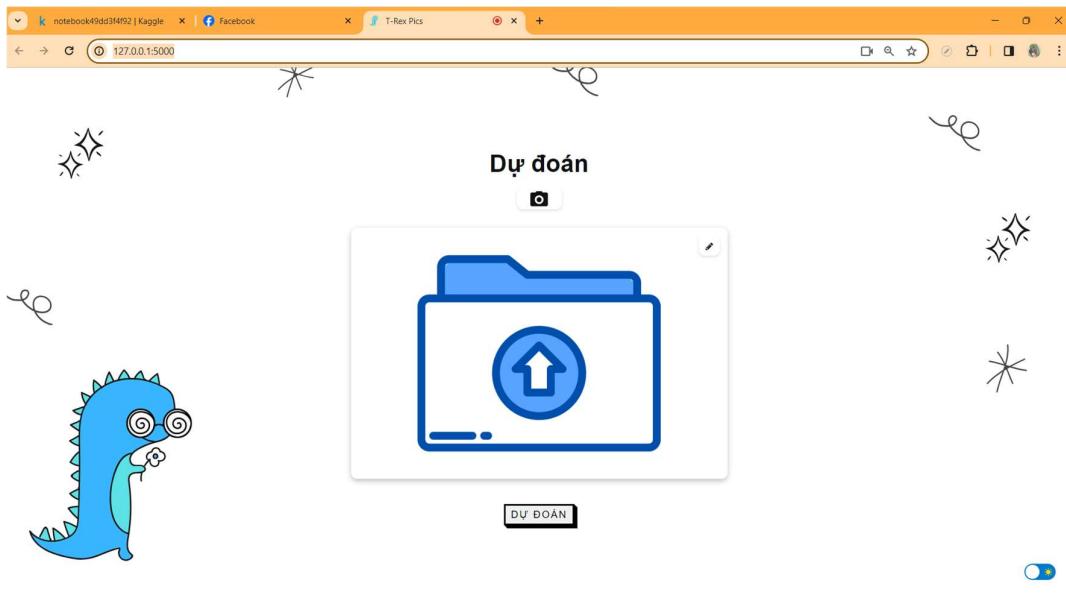
```
evaluation_dico = model.evaluate(test_generator, batch_size = 128, verbose = 0, return_dict=True)
print('model_ImageNetB7 :')
print('Loss : ' + str(evaluation_dico['loss']))
print('Accuracy : ' + str(evaluation_dico['accuracy']))
```

```
model_ImageNetB7 :
Loss : 0.5186105370521545
Accuracy : 0.8850499987602234
```

Hình 3.16. Kết quả model sau khi test

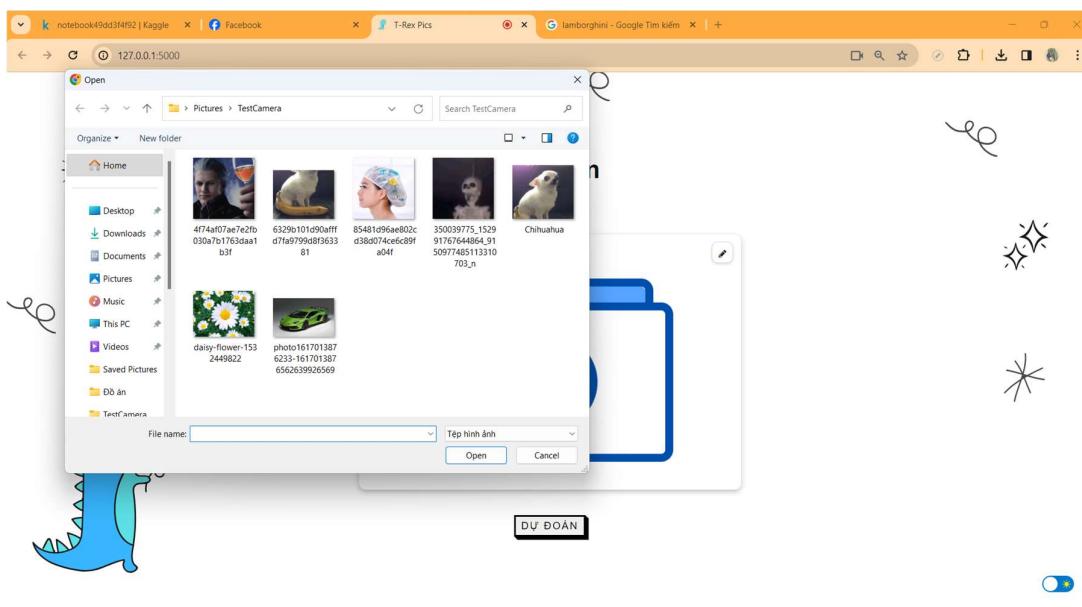
### 3.3. Demo triển khai model lên website

- Sau khi đã có model, nhóm thiết kế một trang web để đẩy model lên và dự đoán kết quả dựa trên hình ảnh truyền vào.
- Đây là giao diện của website, model đã được sẵn lên chỉ cần truyền hình ảnh vào là có thể dự đoán.



Hình 3.17. Giao diện trang chủ

- Nhấn vào biểu tượng chỉnh sửa trên góc để chọn đường dẫn hình ảnh trong máy.



Hình 3.18. Chọn ảnh từ thư mục

- Đây là một số bức hình tải xuống trên mạng và không có trong dữ liệu huấn luyện mô hình. Chọn một tấm và nhấn vào dự đoán để xem kết quả.



DỰ ĐOÁN

**SHOWER CAP**

Tìm hiểu thêm

Hình 3.19. Dự đoán shower cap

- Kết quả được trả ra là lớp có xác suất cao nhất và dựa theo file chứa mã và tên để lấy ra tên của lớp đó.

Xác suất của lớp 788: 1.010841674627769e-17  
Xác suất của lớp 789: 2.5174232782704083e-17  
Xác suất của lớp 790: 5.022710472786168e-19  
Xác suất của lớp 791: 1.8456166918382526e-20  
Xác suất của lớp 792: 2.566263661230234e-16  
Xác suất của lớp 793: 1.0  
Xác suất của lớp 794: 5.766505437217184e-14  
Xác suất của lớp 795: 2.945288736686117e-17  
Xác suất của lớp 796: 3.1424530590396437e-16  
Xác suất của lớp 797: 3.779556528860084e-12  
Xác suất của lớp 798: 2.5635619239135135e-17  
Xác suất của lớp 799: 1.4413946428536628e-17

Hình 3.20. Danh sách các lớp và tỉ lệ chính xác của nó

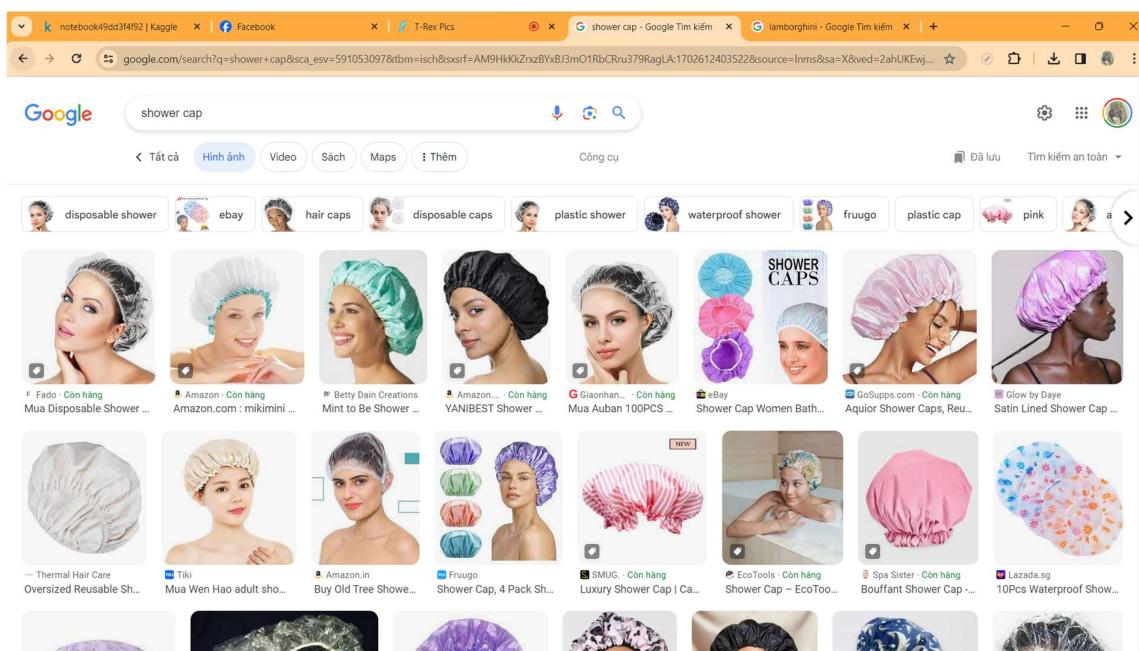
```

789 n04200800 shoe shop, shoe-shop, shoe store
790 n04201297 shoji
791 n04204238 shopping basket
792 n04204347 shopping cart
793 n04208210 shovel
794 n04209133 shower cap
795 n04209239 shower curtain
796 n04228054 ski
797 n04229816 ski mask
798 n04235860 sleeping bag
799 n04229762 slide rule, clinometer

```

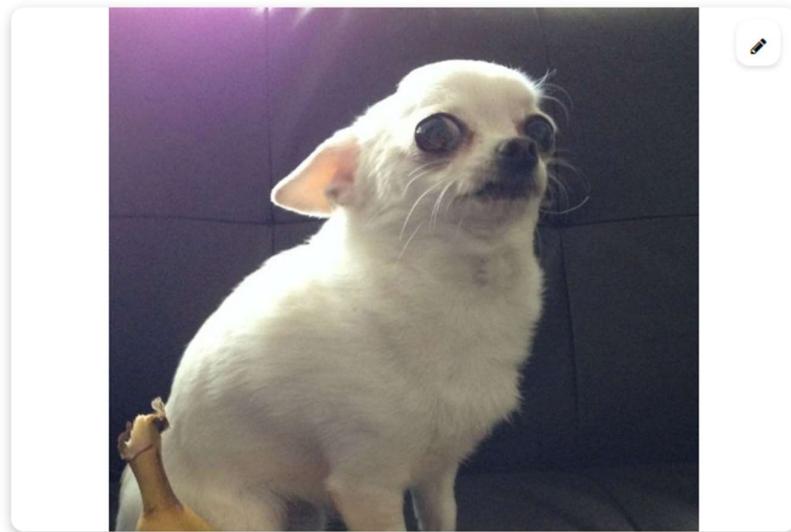
Hình 3.21. Tên của class đã được dự đoán

- Nhấn vào nút “Tìm hiểu thêm” để thấy các sản phẩm với tên tương tự trên trang tìm kiếm của Google.



Hình 3.22. Tìm kiếm trên Google

- Dưới đây là một vài kết quả dự đoán khác :



DỰ ĐOÁN

CHIHUAHUA

Tìm hiểu thêm

Hình 3.23. Dự đoán chihuahua



DỰ ĐOÁN

SPORTS CAR, SPORT CAR

Tìm hiểu thêm

Hình 3.24. Dự đoán sport car



DỰ ĐOÁN

TABBY, TABBY CAT

Tìm hiểu thêm

*Hình 3.25. Dự đoán tabby cat*

- Có thể lấy hình ảnh truyền thẳng vào từ camera của bạn để dự đoán. Sau khi nhập vào biểu tượng máy ảnh ở phía trên và cấp quyền cho phép camera hoạt động chúng ta có thể chụp được hình ảnh từ camera và dự đoán.



DỰ ĐOÁN

WATER BOTTLE

Tìm hiểu thêm

Hình 3.26. Dự đoán warter bottle



DỰ ĐOÁN

HAND BLOWER, BLOW DRYER, BLOW DRIER, HAIR DRYER, HAIR DRIER

Tìm hiểu thêm

Hình 3.27. Dự đoán hair drier

## **CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

### **4.1. Kết luận**

- Bài báo cáo và demo đã trình bày được phương pháp huấn luyện mô hình và phân tích dự đoán được hình ảnh dựa trên kiến trúc mạng nơ-ron EfficientNet. Trong kết quả huấn luyện của mô hình EfficientNet, chúng ta có thể thấy rõ sự hiệu quả và khả năng tổng quát hóa của kiến trúc này. Tuy nhiên vẫn còn một số hạn chế nhất định như: Khó nhận diện được các vật thể không rõ ràng (nhiều vật thể, hình không đúng trọng tâm, các hình ảnh bị biến dạng, ...), dữ liệu để huấn luyện chưa được đầy đủ, cũ hoặc sai dữ liệu ảnh hưởng đến tỉ lệ dự đoán.
- Thông qua việc huấn luyện mô hình chúng ta cũng đã thấy sự quan trọng của việc theo dõi và đánh giá hiệu suất của mô hình trong quá trình huấn luyện để có thể điều chỉnh chiến lược huấn luyện một cách linh hoạt. Điều này giúp chúng ta phát hiện và giải quyết các vấn đề nhanh chóng, đồng thời cải thiện khả năng học của mô hình.

### **4.2. Hướng phát triển**

- Nâng cao hiệu suất và tỉ lệ dự đoán của mô hình.
- Mở rộng dữ liệu huấn luyện để có khả năng dự đoán tốt hơn.
- Tạo thêm các chức năng mới cho website như: Đoán hình từ ảnh vẽ, Giải đố, ... các chức năng có tích hợp Dự đoán.
- Cải thiện và tối ưu thời gian chạy của website.
- Mở rộng khả năng dự đoán trên nền tảng di động.
- Nghiên cứu thêm các cấu trúc mạng tối ưu hơn theo từng mục đích sử dụng.
- Tăng cường bảo mật và lưu trữ dữ liệu cho người dùng.

## Tài liệu tham khảo

- [1] <https://www.image-net.org>
- [2] <https://www.freecodecamp.org/news/learn-to-build-a-convolutional-neural-network-on-the-web/>
- [3] <https://www.tensorflow.org/tutorials/keras/classification>
- [4] <https://www.analyticsvidhya.com/blog/2020/02/mathematics-behind-convolutional-neural-network>
- [5] <https://blog.research.google/2019/05/efficientnet-improving-accuracy-and.html>
- [6] <https://github.com/huggingface/pytorch-image-models/blob/main/docs/results.md>
- [7] [https://keras.io/examples/vision/image\\_classification\\_efficientnet\\_fine\\_tuning/](https://keras.io/examples/vision/image_classification_efficientnet_fine_tuning/)
- [8] [https://www.image-net.org/static\\_files/files/supervision.pdf](https://www.image-net.org/static_files/files/supervision.pdf)
- [9] <https://www.kaggle.com/code/uysimty/get-start-image-classification/notebook>
- [10] <https://vinbigdata.com/kham-pha/04-mo-hinh-pre-trained-cnn-giup-ban-giai-quyetcac-baitoan-thi-giac-may-tinh-voi-transfer-learning.html>
- [11] <https://arxiv.org/pdf/1905.11946.pdf>
- [12] EfficientNet - một cách nghĩ mới trong scale model với accuracy cao - Mì AI - YouTube
- [13] <https://tanca.io/blog/deep-learning-la-gi-hieu-tong-quan-ve-deep-learning-va-ung-dung>
- [14] <https://anonystick.com/blog-developer/su-dung-javascript-truy-cap-camera-sau-truoc-va-chup-man-minh-thiet-bi-di-dong>
- [15] <https://flask.palletsprojects.com/en/3.0.x/quickstart/>
- [16] <https://viblo.asia/p/efficientnet>