

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
KHOA CÔNG NGHỆ THÔNG TIN



TIỂU LUẬN CHUYÊN NGÀNH

Đề Tài:

**Nhận dạng hình ảnh thực phẩm dựa
trên Deep Learning bằng TensorFlow**

GVHD: ThS. LÊ THỊ MINH CHÂU

SVTH:

LƯU GIA BẢO 19133008

NGUYỄN QUỐC VIỆT 19133068

Tp.HCM, tháng 1/2023

PHIẾU NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

Họ và tên Sinh viên 1: Lưu Gia Bảo

MSSV: 19133008

Họ và tên Sinh viên 2: Nguyễn Quốc Việt

MSSV: 19133068

Ngành: Kỹ thuật dữ liệu

Tên đề tài: Nhận dạng hình ảnh thực phẩm dựa trên Deep Learning bằng Tensorflow

Họ và tên Giáo viên hướng dẫn: Lê Thị Minh Châu

NHẬN XÉT

Về nội dung đề tài khối lượng thực hiện:

.....

.....

.....

.....

1. Ưu điểm:

.....

.....

.....

2. Khuyết điểm

.....

.....

.....

3. Đề nghị cho bảo vệ hay không?

4. Đánh giá loại:

5. Điểm:

Tp. Hồ Chí Minh, ngày tháng năm 2022

Giáo viên hướng dẫn

(Ký & ghi rõ họ tên)

PHIẾU NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN

Họ và tên Sinh viên 1: Lưu Gia Bảo

MSSV: 19133008

Họ và tên Sinh viên 2: Nguyễn Quốc Việt

MSSV: 19133068

Ngành: Kỹ thuật dữ liệu

Tên đề tài: Nhận dạng hình ảnh thực phẩm dựa trên Deep Learning bằng Tensorflow

Họ và tên Giáo viên phản biện: Nguyễn Thiên Bảo

NHẬN XÉT

Về nội dung đề tài khối lượng thực hiện:

.....
.....
.....

1. Ưu điểm:

.....
.....
.....

2. Khuyết điểm

.....
.....
.....

3. Đề nghị cho bảo vệ hay không?

4. Đánh giá loại:

5. Điểm:

Tp. Hồ Chí Minh, ngày tháng năm 2022

Giáo viên phản biện

(Ký & ghi rõ họ tên)

PHÂN CÔNG

| Tên thành viên | Công việc |
|-----------------------------|---|
| Lưu Gia Bảo - 19133008 | Huấn luyện mô hình với toàn bộ dữ liệu |
| Nguyễn Quốc Việt - 19133068 | Huấn luyện mô hình cho 2 và 10 loại thực phẩm |

LỜI CẢM ƠN

Lời đầu tiên, nhóm em xin chân thành cảm ơn nhà trường, các thầy cô của khoa Công nghệ thông tin đã tạo điều kiện cũng như trang bị cho em các kiến thức nền tảng để thực hiện trọn vẹn bài tiểu luận này.

Nhóm em xin trân trọng cảm ơn cô Lê Thị Minh Châu – giảng viên hướng dẫn đã tận tình giúp đỡ và cung cấp những thông tin hữu ích trong quá trình thực hiện tiểu luận. Cảm ơn cô đã luôn giải đáp và đóng góp ý kiến cũng như những nhận xét để nhóm cải thiện và hoàn thành được bài tiểu luận chuyên ngành này.

Do kiến thức và thời gian còn hạn chế nên khó tránh khỏi những thiếu sót. Rất mong nhận được những ý kiến đóng góp từ cô để nhóm có thể có thể cải thiện hơn.

Nhóm thực hiện xin chân thành cảm ơn cô.

MỤC LỤC

| | |
|---|----|
| PHÂN CÔNG | 4 |
| LỜI CẢM ƠN..... | 5 |
| MỤC LỤC | 6 |
| DANH MỤC HÌNH ẢNH | 8 |
| MỞ ĐẦU | 10 |
| Lý do chọn đề tài: | 10 |
| Mục đích nghiên cứu: | 10 |
| Đối tượng và phạm vi nghiên cứu | 10 |
| CHƯƠNG 1: CƠ SỞ LÝ THUYẾT | 11 |
| I. Deep Learning..... | 11 |
| 1. Khái niệm | 11 |
| 2. Cách hoạt động của Deep Learning | 11 |
| 3. Ưu nhược điểm của Deep Learning | 12 |
| II. Neural Network..... | 12 |
| 1. Khái niệm về Neuron Network | 12 |
| 2. Cách thức hoạt động | 13 |
| 3. Mô hình Neural Network | 13 |
| 4. Ứng dụng của Neural Network | 15 |
| III. Convolutional neural network | 15 |
| 1. Khái niệm | 15 |
| 2. Các lớp của CNN..... | 16 |
| 3. Cách thức hoạt động CNN..... | 17 |

| | |
|--|----|
| CHƯƠNG 2: THIẾT KẾ HỆ THỐNG | 18 |
| I. Giới thiệu về bộ dữ liệu: | 18 |
| II. Đọc và tiền xử lý dữ liệu:..... | 18 |
| 1. Đọc dữ liệu:..... | 18 |
| 2. Tiền xử lý dữ liệu: | 20 |
| III. Huấn luyện các mô hình..... | 21 |
| 1. Huấn luyện với 2 loại thực phẩm:..... | 21 |
| 2. Huấn luyện với 10 loại thực phẩm:..... | 24 |
| 3. Huấn luyện với toàn bộ dữ liệu thực phẩm (101 loại):..... | 27 |
| CHƯƠNG 3: TRIỂN KHAI ỨNG DỤNG | 31 |
| I. Lấy thông tin thực phẩm bằng gọi API | 31 |
| II. Đưa mô hình dự đoán lên web bằng Streamlit | 34 |
| III. Kết quả chương trình..... | 38 |
| 1. Giao diện ứng dụng và cách chạy | 38 |
| 2. Thông tin chi tiết về model | 39 |
| 3. Nhận dạng thực phẩm bằng hình ảnh..... | 41 |
| ĐÁNH GIÁ VÀ KẾT LUẬN | 45 |
| Kết quả đạt được..... | 45 |
| Hạn chế..... | 45 |
| Hướng phát triển | 45 |
| TÀI LIỆU THAM KHẢO | 46 |

DANH MỤC HÌNH ẢNH

| | |
|---|----|
| Hình 1. Hoạt động Neuron network..... | 12 |
| Hình 2. Mô hình Neuron network..... | 14 |
| Hình 3. Ứng dụng Neuron network..... | 15 |
| Hình 4. Cấu trúc CNN | 16 |
| Hình 5. Cách hoạt động CNN..... | 17 |
| Hình 6. Food101 dataset | 18 |
| Hình 7. Cấu trúc thư mục data | 19 |
| Hình 8. Ảnh pizza..... | 19 |
| Hình 9. Ví dụ ảnh chuyển đổi sang numpy | 20 |
| Hình 10. Tiền xử lý dữ liệu..... | 21 |
| Hình 11. Cấu trúc mô hình với 2 loại..... | 22 |
| Hình 12. Fit model với 2 loại..... | 23 |
| Hình 13. Đánh giá mô hình với 2 loại | 23 |
| Hình 14. Tham số mô hình | 24 |
| Hình 15. Tensorflow workflow..... | 24 |
| Hình 16. EfficientNetB0..... | 25 |
| Hình 17. Cấu trúc mô hình với 10 loại..... | 26 |
| Hình 18. Fit model với 10 loại | 26 |
| Hình 19. Đánh giá mô hình với 10 loại..... | 27 |
| Hình 20. Biểu đồ đánh giá các mô hình | 28 |
| Hình 21. Cấu trúc mô hình toàn bộ dữ liệu | 29 |
| Hình 22. Fit model toàn bộ dữ liệu | 30 |
| Hình 23. Đánh giá model với toàn bộ dữ liệu | 30 |

| | |
|---|----|
| Hình 24. API National Agricultural Library | 31 |
| Hình 25. Kết quả trả về trên API | 32 |
| Hình 26. Lấy thông tin từ API | 32 |
| Hình 27. Lưu thông tin từ API..... | 33 |
| Hình 28. Đổi sang CSV | 33 |
| Hình 29. Kết quả file CSV..... | 34 |
| Hình 30. Mô hình dùng nhận dạng | 34 |
| Hình 31. Hàm dự đoán..... | 35 |
| Hình 32. Hàm vẽ đồ thị dự đoán..... | 36 |
| Hình 33. Hàm vẽ thông tin thực phẩm..... | 36 |
| Hình 34. Hàm vẽ lịch sử training mô hình | 37 |
| Hình 35. Hàm vẽ sai số mô hình..... | 38 |
| Hình 36. Chạy ứng dụng..... | 38 |
| Hình 37. Giao diện | 39 |
| Hình 38. Thông tin mô hình..... | 40 |
| Hình 39. Thông tin cấu trúc mô hình | 40 |
| Hình 40. Thông tin đánh giá mô hình | 41 |
| Hình 41. Chọn hình ảnh..... | 41 |
| Hình 42. Hình ảnh cần nhận dạng..... | 42 |
| Hình 43. Kết quả nhận dạng | 43 |
| Hình 44. Thông tin thực phẩm..... | 44 |
| Hình 45. Thông tin thực phẩm trên nutrition | 44 |

MỞ ĐẦU

Lý do chọn đề tài:

Thực phẩm là một nhu cầu thiết yếu và không thể thiếu trong đời sống của mỗi con người. Ngày nay, với sự phát triển của xã hội các loại thực phẩm cũng nhiều và đa dạng hơn kéo theo các ứng dụng về theo dõi sức khỏe, quản lý bữa ăn ngày càng phổ biến.

Vì thế nhóm em quyết định chọn đề tài này để xây dựng một hệ thống nhận dạng thực phẩm từ đó có thể tích hợp vào những hệ thống quản lý sức khỏe nhằm giúp cho người dùng dễ dàng quản lý và theo dõi chế độ dinh dưỡng hàng ngày một cách hiệu quả nhất.

Mục đích nghiên cứu:

Mục đích nghiên cứu đề tài là xây dựng 1 phần mềm nhận dạng thực phẩm bằng hình ảnh. Dữ liệu thu được giúp cho các ứng dụng quản lý liên quan tới sức khỏe và thực phẩm có thể nhận biết được các loại thực phẩm và đưa ra những phân tích có lợi cho người dùng. Việc nhận dạng thực phẩm dựa vào thuật toán CNN (Convolutional Neural Network).

Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu chính của nhóm là các thuật toán xử lý ảnh, nhằm cung cấp cho việc nhận dạng những dữ liệu tốt nhất. Cùng với đó là nghiên cứu mô hình EfficientNet của CNN nhằm so sánh tính hiệu quả của các mô hình trong việc nhận dạng thực phẩm từ các hình ảnh đã xử lý để đưa ra quyết định chọn mô hình nào sử dụng cho hệ thống và ứng dụng.

Trên thực tế, thực phẩm cực kỳ đa dạng nên nếu muốn nhận dạng tất cả là rất khó và cần có hệ thống có vi xử lý mạnh, vì vậy, tiểu luận

của nhóm em sẽ chỉ nhận dạng khoảng 100 loại thực phẩm khác nhau có sẵn trong tập dữ liệu

CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

I. Deep Learning

1. Khái niệm

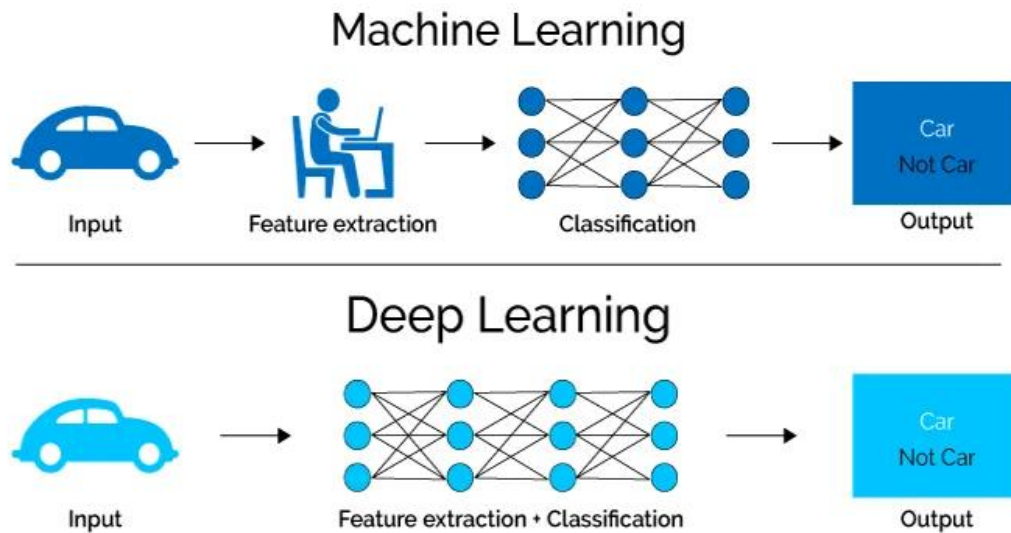
Deep Learning là một nhánh con của nghiên cứu Machine Learning, thông qua việc áp dụng những thuật toán để máy tính tự có thể học và phát triển chính nó. Deep learning được nghiên cứu dựa theo chiều sâu của vấn đề nên nó rất phức tạp.

Ví dụ 1: Để nhận biết một con vật thì đối với chúng ta thì vô cùng đơn giản (Ví dụ: tai thỏ thì dài, lông màu trắng ...), nhưng để diễn tả những điều này bằng dòng lệnh thì rất phức tạp. Bằng cách nghiên cứu và sử dụng Deep Learning thì việc này có thể đơn giản hoá.

2. Cách hoạt động của Deep Learning

Deep Learning sẽ tạo ra một sản phẩm trí tuệ nhân tạo có thể xác định được các kết quả cụ thể dựa vào một tập các dữ liệu đầu vào xác định. Hai kỹ thuật dùng để huấn luyện trong Deep Learning là học có giám sát và học không giám sát.

Machine Learning yêu cầu người dùng cần xác định các quy trình cụ thể và phải tự điều chỉnh các thông số đầu vào để máy tính có thể hoàn thành được mô hình cho kết quả chính xác. Trong khi đó Deep learning sẽ tự ghi nhận những kết quả sai và tự điều chỉnh thông số phù hợp để cho ra kết quả chính xác.



Hình 1. Hoạt động Neuron network

3. Ưu nhược điểm của Deep Learning

1.1. Ưu điểm:

- Không cần kỹ thuật Feature Engineering, một kỹ thuật quan trọng trong học máy để nâng cao độ chính xác
- Mang lại kết quả tốt với nhiều loại bài toán khác nhau
- Hỗ trợ chế độ song song và phân tán
- Có khả năng mở rộng cao

1.2. Nhược điểm:

- Tính toán trên ram nên yêu cầu dung lượng ram lớn để mô hình hoạt động tốt
- Yêu cầu dung lượng dữ liệu lớn
- Không có lý thuyết tiêu chuẩn nào để hướng dẫn lựa chọn các công cụ học sâu phù hợp

II. Neural Network

1. Khái niệm về Neuron Network

Mạng lưới thần kinh là một tập hợp bao gồm các thuật toán, được mô phỏng tương tự theo bộ não con người, thiết kế để nhận dạng các

mẫu. NN diễn giải dữ liệu cảm quan thông qua một loại nhận thức máy móc, gán nhãn hoặc phân cụm đầu vào thô. Các mẫu mà chúng nhận dạng là số, chứa trong các vectơ, trong đó các mẫu là tất cả dữ liệu trong thế giới thực, có thể là hình ảnh, âm thanh, văn bản hoặc chuỗi thời gian.

Mạng nơ-ron cũng có thể extract features để cung cấp cho các thuật toán khác phân cụm và phân loại; vì vậy, có thể coi mạng nơ-ron sâu như các thành phần của các ứng dụng máy học lớn liên quan đến reinforcement learning, classification and regression.

2. Cách thức hoạt động

Thông thường, một mạng nơ ron yêu cầu cung cấp một lượng lớn dữ liệu để đào tạo. Việc đào tạo bao gồm xác định đầu vào và cho mạng nơ ron biết đầu ra là gì. Ví dụ, để xây dựng một mạng để nhận dạng đồ ăn, quá trình đào tạo ban đầu có thể là một loạt ảnh về các đồ ăn, ảnh không phải đồ ăn. Mỗi đầu vào phải kèm theo thông tin nhận dạng phù hợp để xác định ảnh đó phải là đồ ăn hay không. Việc cung cấp thông tin giúp cho mô hình điều chỉnh các weight bên trong để thực hiện việc training tốt hơn.

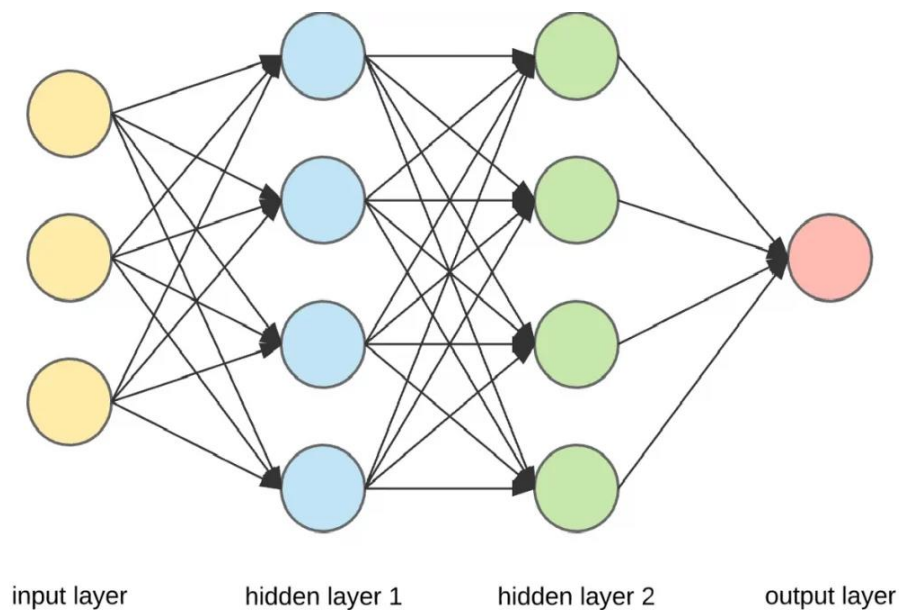
VD: Ví dụ node B, C nói với node A rằng hình ảnh đầu vào hiện tại là hình ảnh về pizza, nhưng node C nói là kem và mô hình đào tạo đã xác nhận đó là hình ảnh pizza, khi đó node C sẽ giảm weight (trọng số) gán vào đầu vào kem và tăng weight ở node B, C

3. Mô hình Neural Network

Mỗi mạng Neural Network bao gồm nhiều lớp với mỗi lớp có nhiều node. Mỗi node trong tầng trước đều sẽ nối với các node ở tầng tiếp theo tạo nên các lớp dày đặc. Các node là nơi diễn ra các quá trình tính toán. Đầu ra của lớp sẽ là đầu vào của lớp tiếp theo. Có 3 lớp cơ bản:

- Input Layer: Tầng đầu vào, là các node có màu vàng, lớp này sẽ chứa các dữ liệu đầu vào. Các node trong lớp này đều cố định nên cần xác định đầu vào tương ứng phù hợp
- Output Layer: Tầng đầu ra, là các node có màu đỏ chứa các thông tin để cho ra kết quả của bài toán, từ đó có thể chuyển đổi để người dùng có thể hiểu được
- Hidden Layer: Tầng ẩn, là tầng kết nối lớp đầu vào và đầu ra, nơi chứa các thông tin để tính toán, giải quyết các vấn đề bài toán đặt ra. Thông thường tầng này có thời gian training lâu nhất.

Một mô hình Neural Network chỉ có duy nhất một input layer và một output layer nhưng có thể có nhiều hidden layer



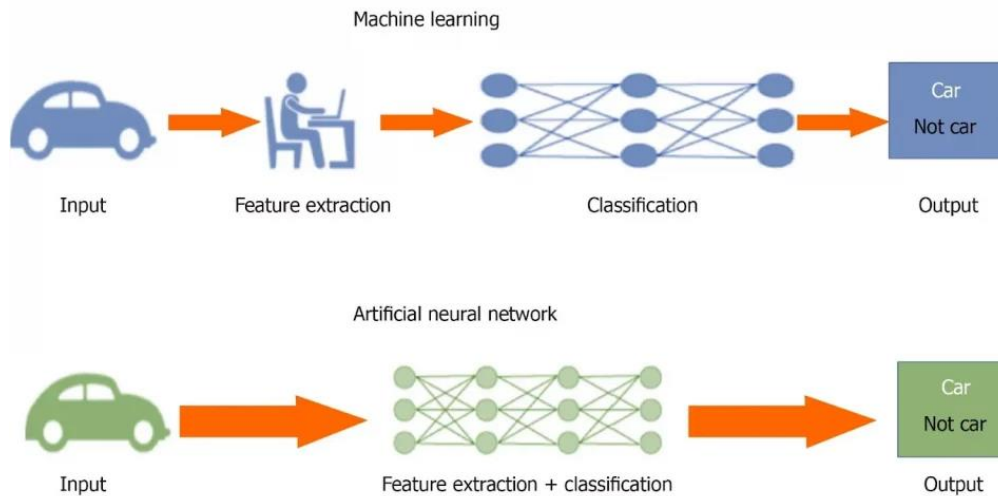
Hình 2. Mô hình Neuron network

Đối với mỗi node trong mô hình mạng neuron đều có các hàm activation, vì các phép toán trong mạng đều là các phép toán tuyến tính nên thông qua hàm activation thì mô hình có thể trở nên tuyến tính hoặc phi tuyến tùy vào yêu cầu bài toán. Một số hàm activation phổ biến (sigmode, relu, softmax, ...)

4. Ứng dụng của Neural Network

Xử lý ảnh là một trong những ứng dụng đầu tiên được mạng nơ ron áp dụng thành công để xử dụng trong nhiều lĩnh vực khác:

- Xử lý ngôn ngữ tự nhiên, dịch thuật.
- Chatbox.
- Xe lái tự động.
- Các ứng dụng nhận diện khuôn mặt, nhận dạng vật thể.
- Dự báo thời tiết, dự đoán giá cổ phiếu chứng khoán.



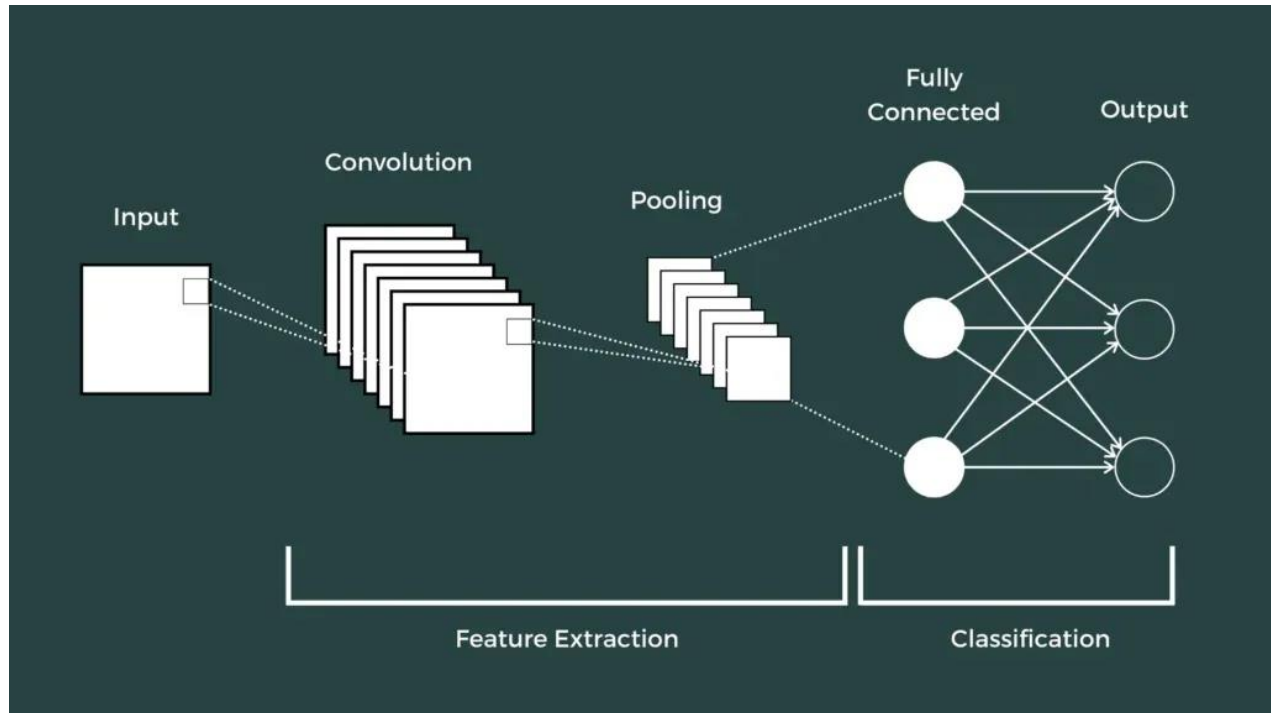
Hình 3. Ứng dụng Neuron network

III. Convolutional neural network

1. Khái niệm

Convolutional neural networks (CNN) là một kiểu của mạng nơ ron nhân tạo có thể khám thông tin trong chuỗi thời gian, âm thanh và hình ảnh. Vì vậy nó rất được phổ biến trong xử lý các tác vụ liên quan đến hình ảnh. CNN hoạt động bằng cách tận dụng các nguyên tắc của đại số tuyến tính như phép nhân hai ma trận

2. Các lớp của CNN



Hình 4. Cấu trúc CNN

a) Convolutional layer

Thông thường phần lớn các tính toán xảy ra trong lớp tích chập. Quá trình tích chập cho phép một ma trận, còn được gọi kernel hay filter, có thể di chuyển qua các trường tiếp nhận của hình ảnh, kiểm tra các feature có trong hình ảnh.

Qua nhiều lần lặp lại, filter quét qua toàn bộ hình ảnh. Sau mỗi lần lặp một tích vô hướng sẽ được tính toán giữa các pixel đầu vào và filter. Đầu ra cuối cùng sau hàng loạt lần lặp được gọi là feature map. Các filter này có tác dụng giúp mô hình có thể nhận biết được những chi tiết đơn giản trong hình ảnh như các cạnh ngang, dài, màu sắc đơn lẻ, từ đó có thể nhận dạng hình ảnh tốt hơn.

b) Pooling layer

Giống như lớp tích chập, Pooling layer cũng có các filter quét qua các hình ảnh đầu vào. Nhưng khác với lớp tích chập Pooling layer làm giảm số lượng tham số trong đầu vào, chỉ lấy ra những chi tiết quan trọng giúp tăng tốc độ xử lý và độ chính xác, bao gồm 2 loại là max pooling và average pooling.

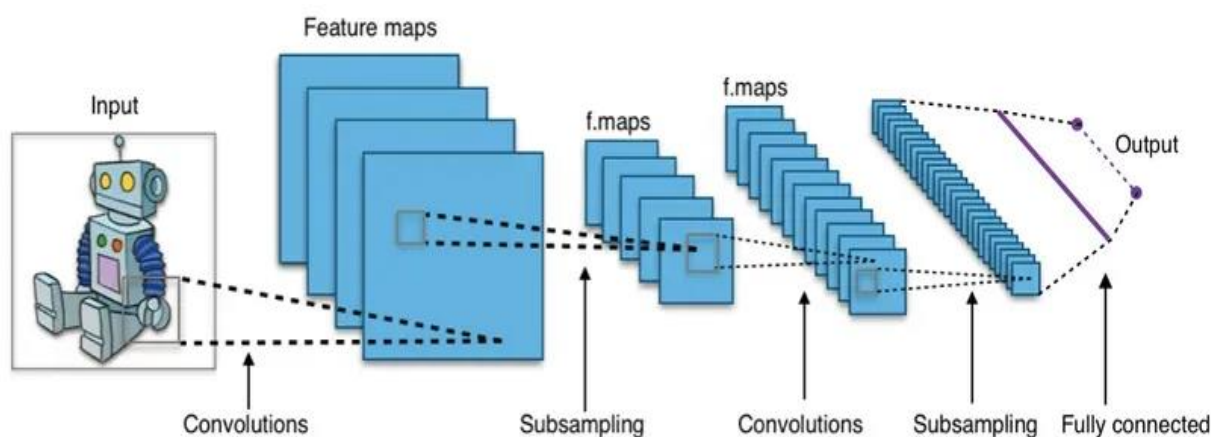
c) Fully connected layer

Đây là lớp diễn ra quá trình phân loại hình ảnh trong CNN dựa trên việc features extracted của lớp trước. Ví dụ: bài toán nhận dạng 10 chữ số viết tay từ 0 đến 9, lớp fully connected sẽ chuyển kết quả của ma trận đặc trưng thành vector có 10 chiều tương ứng với 10 số

3. Cách thức hoạt động CNN

Một mô hình CNN có thể có nhiều lớp, với mỗi lớp có cách thức hoạt động khác nhau. Một bộ lọc được áp dụng cho mỗi hình ảnh để tạo ra đầu ra ngày càng tốt và chi tiết hơn. Ở những lớp đầu tiên filter có thể nhận dạng được các feature đơn giản. Mỗi lớp tiếp theo các filter tăng độ phức tạp để xác định các feature duy nhất đại diện cho bức ảnh đầu vào.

Do đó mỗi hình ảnh đầu ra của các lớp tích chập sẽ là đầu vào của lớp tiếp theo. Ở lớp cuối cùng là lớp fully connected layer là lớp nhận dạng hình ảnh hoặc đối tượng mà nó đại diện



Hình 5. Cách hoạt động CNN

CHƯƠNG 2: THIẾT KẾ HỆ THỐNG

I. Giới thiệu về bộ dữ liệu:

Bộ dữ liệu Food101 bao gồm 101 phân loại thực phẩm, với 101.000 hình ảnh. Mỗi loại thực phẩm sẽ có 1000 hình ảnh và phân thành 750 hình cho tập train, 250 cho tập test. Các nhãn cho hình ảnh thử nghiệm đã được làm sạch theo cách thủ công, trong khi tập huấn luyện có chứa một số nhiễu có chủ ý.

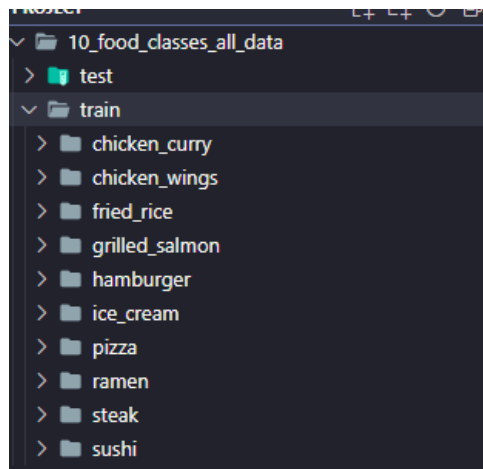


Hình 6. Food101 dataset

II. Đọc và tiền xử lý dữ liệu:

1. Đọc dữ liệu:

Vì đây là dữ liệu lớn (5gb) nên nhóm em tách thành 2 phần: Đầu tiên sẽ trích ra 10 loại thực phẩm (pizza, steak, ...) để dễ xây dựng và training mô hình. Sau đó sẽ đưa toàn bộ dữ liệu vào mô hình để huấn luyện



Hình 7. Cấu trúc thư mục data

Với 10 loại thực phẩm có khoảng 7500 ảnh trong tập train và 2500 ảnh trong tập test chia đều cho 10 nhãn

Hiển thị hình ảnh và số chiều (width, height, colour channels)



Hình 8. Ảnh pizza

Trong trường hợp trên width và height có thể khác nhau nhưng vì đây là ảnh màu nên colour channels luôn bằng 3, giá trị này là các giá trị khác nhau của [red, green và blue (RGB) pixel]

Chuyển đổi ảnh sang numpy array

```
1 # View the img (actually just a big array/tensor)
2 img

array([[185, 126, 48],
       [182, 123, 45],
       [182, 120, 43],
       ...,
       [226, 164, 61],
       [223, 161, 58],
       [227, 165, 64]],

       [[179, 120, 40],
       [183, 121, 44],
       [187, 125, 48],
       ...,
       [223, 161, 58],
       [220, 158, 57],
       [224, 162, 61]],

       [[187, 126, 46],
       [181, 120, 40],
       [182, 120, 43],
       ...,
       [220, 158, 57],
       [226, 164, 61]]])
```

Hình 9. Ví dụ ảnh chuyển đổi sang numpy

2. Tiền xử lý dữ liệu:

Thay đổi kích thước (chiều dài, chiều rộng) hình ảnh (về cùng kích thước mà mô hình đã được đào tạo)

Thay đổi tỷ lệ hình ảnh (các hình ảnh sẽ được scale về các giá trị từ 0 đến 1)



Hình 10. Tiền xử lý dữ liệu

III. Huấn luyện các mô hình

1. Huấn luyện với 2 loại thực phẩm:

Nhóm em lấy 2 loại thực phẩm (pizza, steak) trong tập dữ liệu để huấn luyện cho một mô hình CNN tích chập đơn giản

| | | |
|--------------------------------|----------------------|---------|
| conv2d (Conv2D) | (None, 222, 222, 10) | 280 |
| conv2d_1 (Conv2D) | (None, 220, 220, 10) | 910 |
| max_pooling2d (MaxPooling2D) | (None, 110, 110, 10) | 0 |
| conv2d_2 (Conv2D) | (None, 108, 108, 10) | 910 |
| conv2d_3 (Conv2D) | (None, 106, 106, 10) | 910 |
| max_pooling2d_1 (MaxPooling2D) | (None, 53, 53, 10) | 0 |
| flatten (Flatten) | (None, 28090) | 0 |
| dense (Dense) | (None, 100) | 2809100 |
| dense_1 (Dense) | (None, 100) | 10100 |
| dense_2 (Dense) | (None, 100) | 10100 |
| dense_3 (Dense) | (None, 1) | 101 |

Hình 11. Cấu trúc mô hình với 2 loại

Gồm 4 lớp tích chập (Conv2D) và 2 lớp Pooling (MaxPool2D) sau đó là tới lớp làm phẳng và 3 lớp Fullyconnected và 1 đầu ra.

```

Epoch 1/8
47/47 [=====] - 6s 102ms/step - loss: 0.6655 - accuracy: 0.6227 - val_loss: 0.6228 - val_accuracy: 0.6920
Epoch 2/8
47/47 [=====] - 5s 97ms/step - loss: 0.5816 - accuracy: 0.7013 - val_loss: 0.3706 - val_accuracy: 0.8180
Epoch 3/8
47/47 [=====] - 5s 98ms/step - loss: 0.4361 - accuracy: 0.8020 - val_loss: 0.4146 - val_accuracy: 0.7760
Epoch 4/8
47/47 [=====] - 5s 97ms/step - loss: 0.3591 - accuracy: 0.8460 - val_loss: 0.2941 - val_accuracy: 0.8680
Epoch 5/8
47/47 [=====] - 5s 96ms/step - loss: 0.3226 - accuracy: 0.8727 - val_loss: 0.3373 - val_accuracy: 0.8580
Epoch 6/8
47/47 [=====] - 5s 97ms/step - loss: 0.2204 - accuracy: 0.9180 - val_loss: 0.4389 - val_accuracy: 0.8300
Epoch 7/8
47/47 [=====] - 5s 97ms/step - loss: 0.1430 - accuracy: 0.9493 - val_loss: 0.3287 - val_accuracy: 0.8960
Epoch 8/8
47/47 [=====] - 5s 98ms/step - loss: 0.0844 - accuracy: 0.9700 - val_loss: 0.3531 - val_accuracy: 0.8760

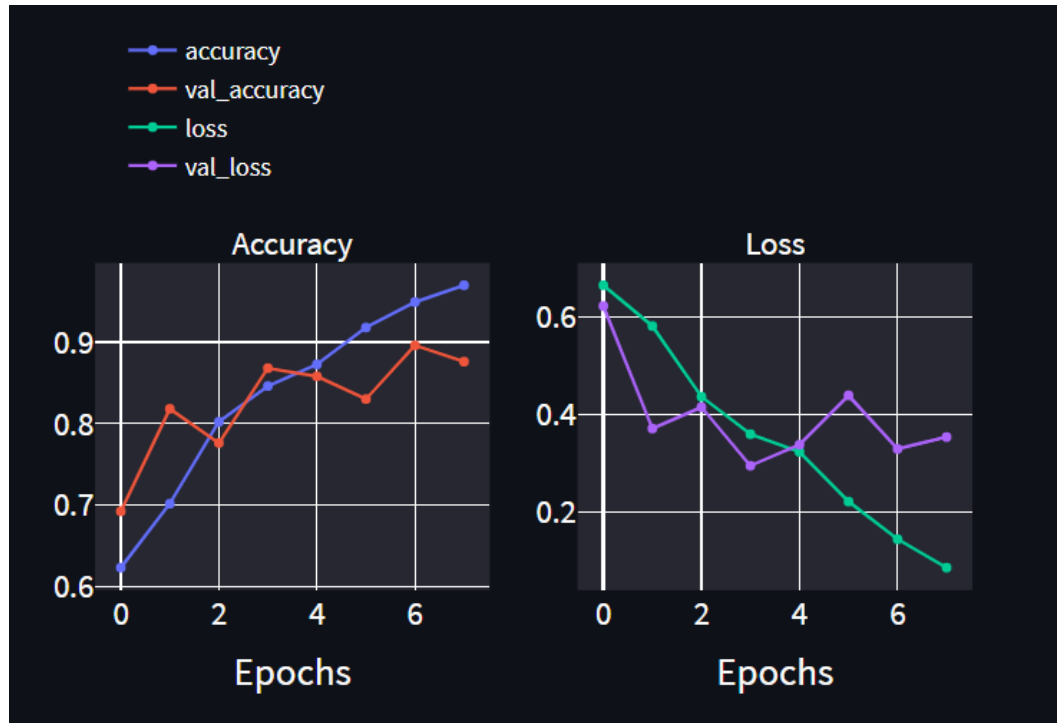
```

Hình 12. Fit model với 2 loại

Sau 8 lần huấn luyện mô hình:

+ Độ chính xác từ 62,27% tăng 97% trên tập train

+ Độ chính xác từ 69,2% tăng 87,6% trên tập test



Hình 13. Đánh giá mô hình với 2 loại

Tổng số tham số được dùng là 2 832 411

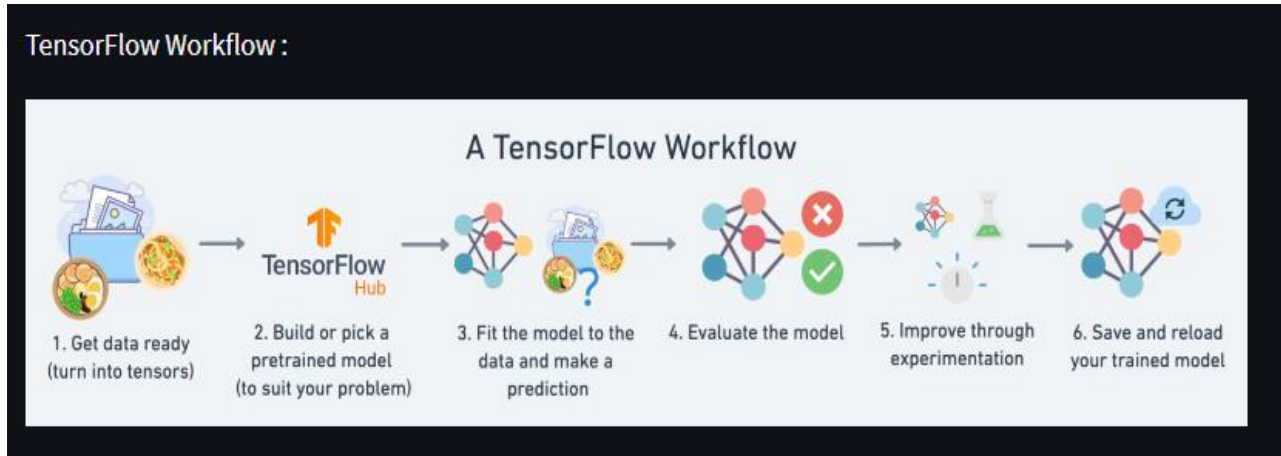
```
Total params: 2,832,411  
Trainable params: 2,832,411  
Non-trainable params: 0
```

Hình 14. Tham số mô hình

2. Huấn luyện với 10 loại thực phẩm:

Khi đưa mô hình huấn luyện 2 loại thực phẩm ở trên để huấn luyện cho 10 loại thực phẩm thì độ chính xác cực kỳ thấp. Vì vậy nhóm em đã tìm hiểu mô hình khác để tăng độ chính xác của việc huấn luyện.

Mô hình của nhóm em tìm hiểu có sẵn trên Tensorflow Hub, đây là nơi những mô hình được xây dựng và chia sẻ. Nhóm em đã lấy mô hình có sẵn trên Tensorflow hub và đưa vào huấn luyện dữ liệu.



Hình 15. Tensorflow workflow

Trong đề tài nhóm em sử dụng mô hình EfficientNet.

EfficientNet lấy cảm hứng từ nghiên cứu Compound Model Scaling (thu phóng mô hình), là phương pháp để tăng thêm độ hiệu quả và độ chính xác của mô hình dự đoán.

Table 1. EfficientNet-B0 baseline network – Each row describes a stage i with \hat{L}_i layers, with input resolution $\langle \hat{H}_i, \hat{W}_i \rangle$ and output channels \hat{C}_i . Notations are adopted from equation 2.

| 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 16. EfficientNetB0

Compound Model Scaling là sự cân bằng 3 tham số là chiều sâu, chiều rộng và độ phân giải của các loại thu phóng lần lượt là thu phóng theo chiều sâu (Depth Scaling), theo chiều rộng (Width Scaling), theo độ phân giải (Resolution Scaling)

Quay về mới mô hình EfficientNet để tránh bị over-fitting chúng ta sẽ thêm một lớp dropout

| Layer (type) | Output Shape | Param # |
|--|--------------------------|---------|
| input_layer (InputLayer) | [(None, 224, 224, 3)] | 0 |
| efficientnetb0 (Functional) | (None, None, None, 1280) | 4049571 |
| pooling_layer (GlobalAveragePooling2D) | (None, 1280) | 0 |
| dropout (Dropout) | (None, 1280) | 0 |
| dense (Dense) | (None, 10) | 12810 |
| activation (Activation) | (None, 10) | 0 |

Hình 17. Cấu trúc mô hình với 10 loại

Sau 5 lần huấn luyện, độ chính xác tăng từ 72,88% lên 87,84% (tập train) và từ 90,62% lên 92,33% (tập test)

| | |
|-----------|---|
| Epoch 1/5 | 235/235 [=====] - 228s 944ms/step - loss: 0.9157 - accuracy: 0.7288 - val_loss: 0.4093 - val_accuracy: 0.9062 |
| Epoch 2/5 | 235/235 [=====] - 239s 1s/step - loss: 0.5233 - accuracy: 0.8421 - val_loss: 0.3380 - val_accuracy: 0.8864 |
| Epoch 3/5 | 235/235 [=====] - 248s 1s/step - loss: 0.4524 - accuracy: 0.8575 - val_loss: 0.3172 - val_accuracy: 0.9176 |
| Epoch 4/5 | 235/235 [=====] - 240s 1s/step - loss: 0.4126 - accuracy: 0.8705 - val_loss: 0.2895 - val_accuracy: 0.9091 |
| Epoch 5/5 | 235/235 [=====] - 239s 1s/step - loss: 0.3812 - accuracy: 0.8784 - val_loss: 0.2887 - val_accuracy: 0.9233 |

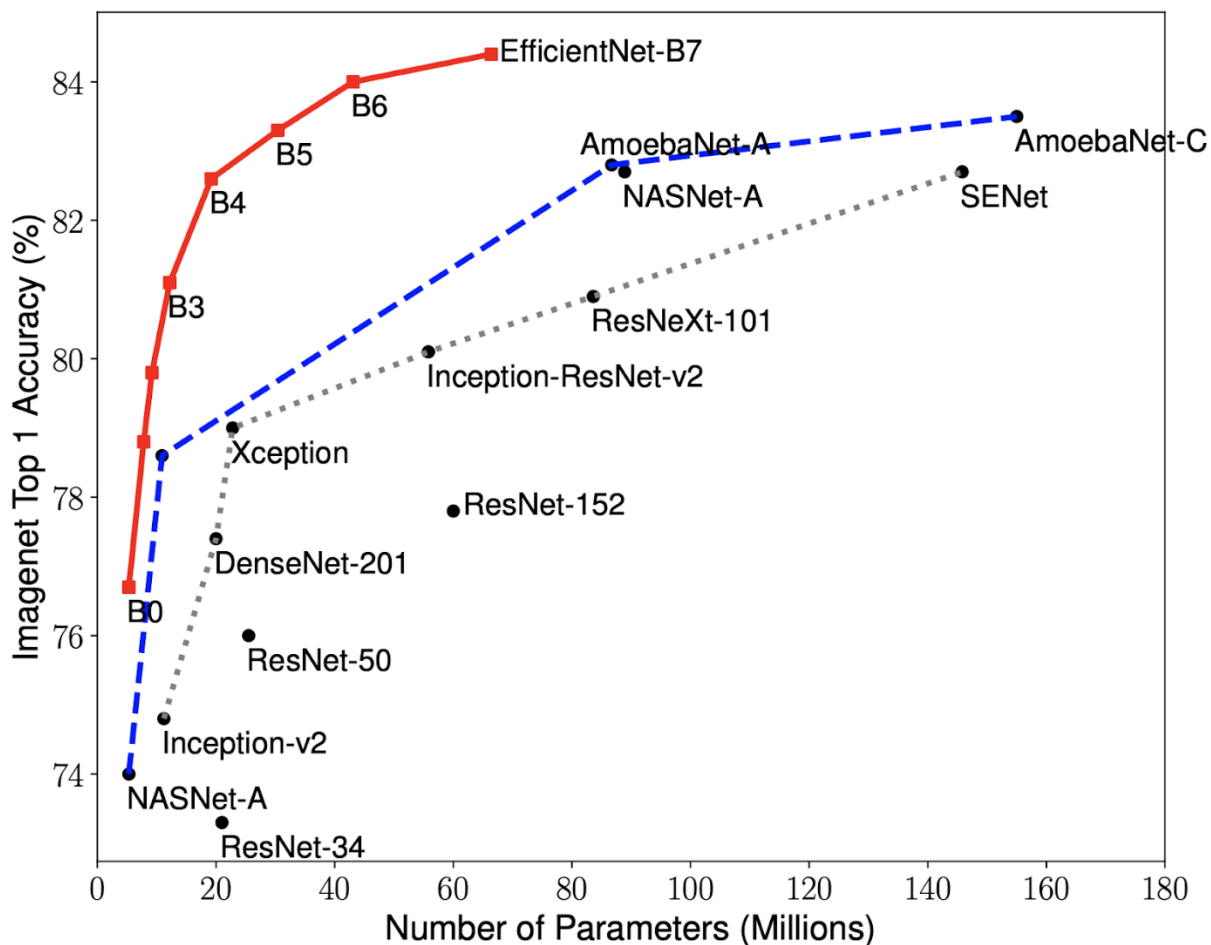
Hình 18. Fit model với 10 loại



Hình 19. Đánh giá mô hình với 10 loại

3. Huấn luyện với toàn bộ dữ liệu thực phẩm (101 loại):

EfficientNetB0 tuy khá hiệu quả để huấn luyện 10 loại thực phẩm, nhưng khi dùng cho 101 loại thì độ chính xác khá thấp. Tuy nhiên EfficientNet có nhiều loại phiên bản khác nhau để phù hợp với nhiều trường hợp khác nhau. Dưới đây là hình ảnh so sánh các mô hình theo bài báo của Mingxing Tan – kỹ sư phần mềm của google AI.



Hình 20. Biểu đồ đánh giá các mô hình

Trên đây là bảng so sánh các loại mô hình, mô hình EfficientNet luôn có đặt độ chính xác khá cao với tham số sử dụng thấp nhất.

Để phù hợp với toàn bộ dữ liệu (101 loại thực phẩm), nhóm em chọn EfficientNetB1 để huấn luyện và được độ chính xác đạt khá cao (85%)

| Layer (type) | Output Shape | Param # |
|--|--------------------------|---------|
| ===== | | |
| input_layer (InputLayer) | [(None, 224, 224, 3)] | 0 |
| efficientnetb1 (Functional) | (None, None, None, 1280) | 6575239 |
| pooling_layer (GlobalAveragePooling2D) | (None, 1280) | 0 |
| dropout (Dropout) | (None, 1280) | 0 |
| dense (Dense) | (None, 101) | 129381 |
| activation (Activation) | (None, 101) | 0 |
| ===== | | |
| Total params: 6,704,620 | | |
| Trainable params: 6,642,565 | | |
| Non-trainable params: 62,055 | | |

Hình 21. Cấu trúc mô hình toàn bộ dữ liệu

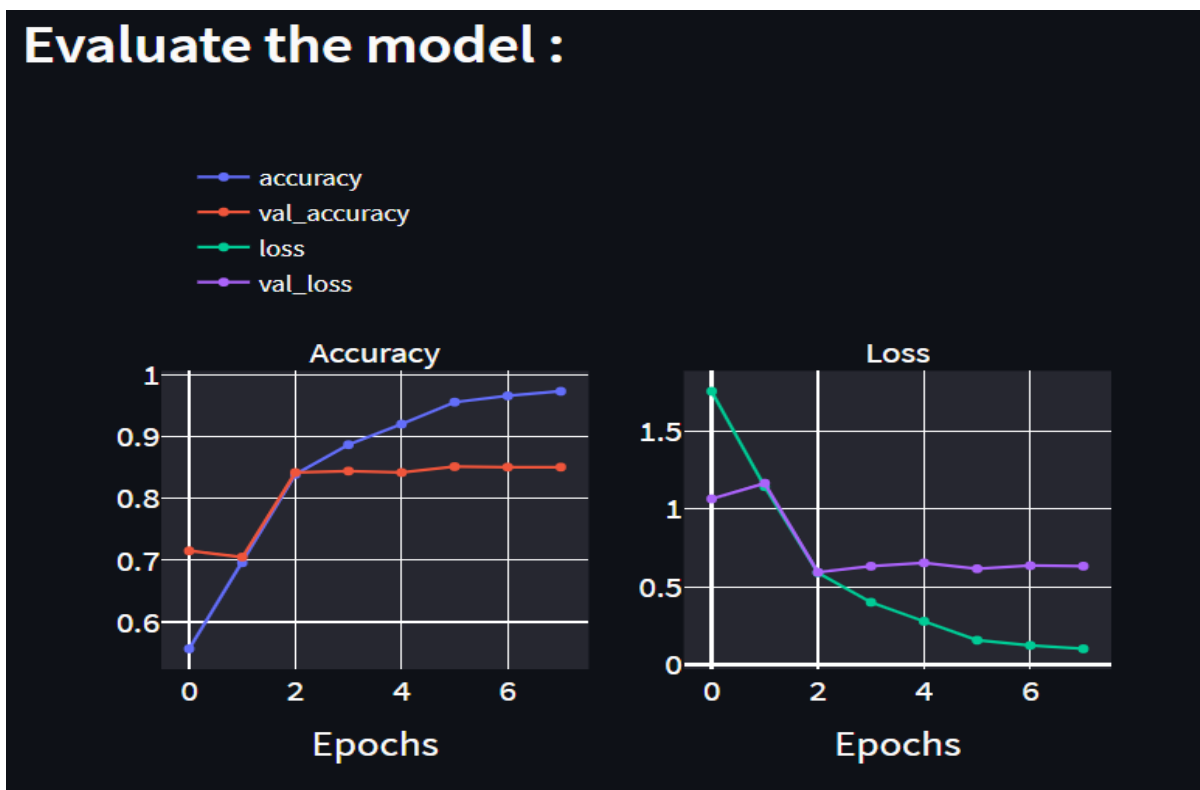
Cụ thể là sau 8 lần huấn luyện, độ chính xác tăng từ 55,67% lên 97,3% (tập train) và từ 71,5% lên 85,04% (tập test)

```

Saving TensorBoard log files to: training-logs/EfficientNetB1-/20221124-055450
Epoch 1/10
2368/2368 [=====] - 1071s 442ms/step - loss: 1.7534 - accuracy: 0.5567 - val_loss: 1.0633 - val_accuracy: 0.7150 - lr: 0.0010
Epoch 2/10
2368/2368 [=====] - ETA: 0s - loss: 1.1416 - accuracy: 0.6958
Epoch 2: ReduceLROnPlateau reducing learning rate to 0.00020000000949949026.
2368/2368 [=====] - 1047s 441ms/step - loss: 1.1416 - accuracy: 0.6958 - val_loss: 1.1640 - val_accuracy: 0.7050 - lr: 0.0010
Epoch 3/10
2368/2368 [=====] - 1048s 442ms/step - loss: 0.5901 - accuracy: 0.8387 - val_loss: 0.5921 - val_accuracy: 0.8416 - lr: 2.0000e-04
Epoch 4/10
2368/2368 [=====] - 1047s 441ms/step - loss: 0.4004 - accuracy: 0.8866 - val_loss: 0.6315 - val_accuracy: 0.8440 - lr: 2.0000e-04
Epoch 5/10
2368/2368 [=====] - ETA: 0s - loss: 0.2781 - accuracy: 0.9199
Epoch 5: ReduceLROnPlateau reducing learning rate to 4.0000001899898055e-05.
2368/2368 [=====] - 1047s 441ms/step - loss: 0.2781 - accuracy: 0.9199 - val_loss: 0.6529 - val_accuracy: 0.8416 - lr: 2.0000e-04
Epoch 6/10
2368/2368 [=====] - 1048s 442ms/step - loss: 0.1568 - accuracy: 0.9554 - val_loss: 0.6154 - val_accuracy: 0.8512 - lr: 4.0000e-05
Epoch 7/10
2368/2368 [=====] - ETA: 0s - loss: 0.1236 - accuracy: 0.9656
Epoch 7: ReduceLROnPlateau reducing learning rate to 8.000000525498762e-06.
2368/2368 [=====] - 1048s 442ms/step - loss: 0.1236 - accuracy: 0.9656 - val_loss: 0.6352 - val_accuracy: 0.8504 - lr: 4.0000e-05
Epoch 8/10
2368/2368 [=====] - ETA: 0s - loss: 0.1019 - accuracy: 0.9730Restoring model weights from the end of the best epoch: 6.
Epoch 8: ReduceLROnPlateau reducing learning rate to 1.6000001778593287e-06.
2368/2368 [=====] - 1047s 441ms/step - loss: 0.1019 - accuracy: 0.9730 - val_loss: 0.6318 - val_accuracy: 0.8504 - lr: 8.0000e-06
Epoch 8: early stopping

```

Hình 22. Fit model toàn bộ dữ liệu

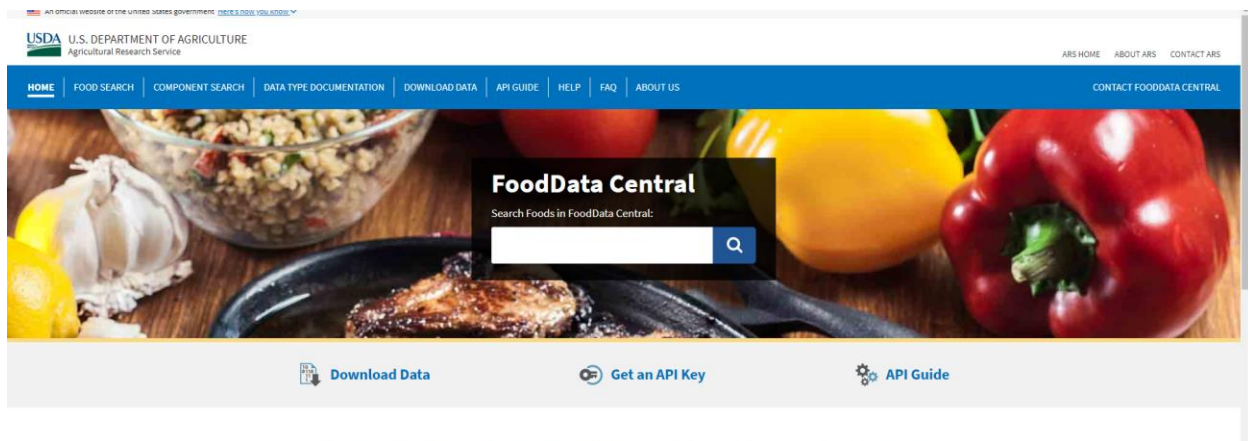


Hình 23. Đánh giá model với toàn bộ dữ liệu

CHƯƠNG 3: TRIỂN KHAI ỨNG DỤNG

I. Lấy thông tin thực phẩm bằng gọi API

Ngoài việc nhận dạng thực phẩm thì nhóm em còn muốn đưa thêm thông tin về loại thực phẩm đó như giá trị dinh dưỡng, ... Để làm được như vậy thì cần phải gọi api để lấy dữ liệu. Nhóm em chọn trang API National Agricultural Library, đây là trang web lưu trữ dữ liệu về thông tin thực phẩm phục vụ cho nông nghiệp



Hình 24. API National Agricultural Library

Để gọi api thì cần có api key, có thể lấy được trên trang FoodData Central ([usda.gov](https://www.usda.gov)). Sau khi có api key thì url có dạng:

`https://api.nal.usda.gov/fdc/v1/foods/search?api_key=d4D6dSOc81pTA
OY2gsNZ0YhjkMlhStLJRoII5SJJu&query=" + name`

Trong đó *name* sẽ là tên thực phẩm cần lấy thông tin, kết quả trả về với món sashimi:

```
https://api.nal.usda.gov/fdc/v1/foods/search?api_key=d4D6d5Oc81pTA0Y2gsNZ0YhjkMlhStLJR0II5SJu&query=sashimi

{"score": 1433.5157,
 "microbes": [
 ],
 "foodNutrients": [
 {
   "nutrientId": 1003,
   "nutrientName": "Protein",
   "nutrientNumber": "203",
   "unitName": "G",
   "derivationCode": "LCCS",
   "derivationDescription": "Calculated from value per serving size measure",
   "derivationId": 70,
   "value": 19.7,
   "foodNutrientSourceId": 9,
   "foodNutrientSourceCode": "12",
   "foodNutrientSourceDescription": "Manufacturer's analytical; partial documentation",
   "rank": 600,
   "indentLevel": 1,
   "foodNutrientId": 26969159
 },
 {
   "nutrientId": 1004,
   "nutrientName": "Total lipid (fat)",
   "nutrientNumber": "204",
   "unitName": "G",
   "derivationCode": "LCCS",
   "derivationDescription": "Calculated from value per serving size measure",
   "derivationId": 70,
   "value": 2.4
 }
```

Hình 25. Kết quả trả về trên API

Kết quả trên hiển thị các thông tin thực phẩm dưới dạng JSON. Nhóm em sẽ lấy 5 thành phần: 'protein', 'calcium', 'fat', 'carbohydrates', 'vitamins'

```
for name in food_name:
    url = "https://api.nal.usda.gov/fdc/v1/foods/search?api_key=d4D6d5Oc81pTA0Y2gsNZ0YhjkMlhStLJR0II5SJu&query=" + name
    response = requests.get(url)
    data = response.json()
    flatten_json = pd.json_normalize(data["foods"])
    first_food = flatten_json.iloc[0]
    first_food_nutrition_list = first_food.foodNutrients
    for item in first_food_nutrition_list:
        if item['nutrientNumber'] == "203":
            protein = item['value']
            continue
        if item['nutrientNumber'] == "301":
            calcium = item['value']
            continue
        if item['nutrientNumber'] == "204":
            fat = item['value']
            continue
        if item['nutrientNumber'] == "205":
            carbs = item['value']
            continue
        if item['nutrientNumber'] == "318":
            vitamin_a = item['value']
            continue
        if item['nutrientNumber'] == "401":
            vitamin_c = item['value']
            continue
```

Hình 26. Lấy thông tin từ API


```

        continue
    if item['nutrientNumber'] == "205":
        carbs = item['value']
        continue
    if item['nutrientNumber'] == "318":
        vitamin_a = item['value']
        continue
    if item['nutrientNumber'] == "401":
        vitamin_c = item['value']
        continue

    vitamins = float(vitamin_a) + float(vitamin_c)
    print(name)
    nutrition_data = nutrition_data.append({
        'name': name,
        'protein': protein,
        'calcium': calcium / 1000,
        'fat': fat,
        'carbohydrates': carbs,
        'vitamins': vitamins / 1000
    }, ignore_index=True)

return nutrition_data

```

Hình 27. Lưu thông tin từ API

```

    )
nutrition101 = nutrition101.reset_index(drop=True)
nutrition101.to_csv("nutrition101.csv")

```

Hình 28. Đổi sang CSV

Mỗi nutrientNumber tương ứng với mỗi thành phần và nhóm em gọi api cho 101 loại thực phẩm và lấy giá trị của mỗi thành phần đó lưu thành file csv.

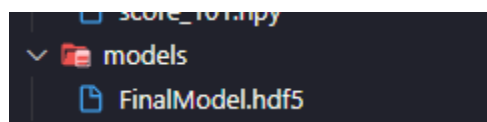
Kết quả:

```
nutrition101.csv X
nutrition101.csv
1 ,name,protein,calcium,fat,carbohydrates,vitamins
2 0,apple pie,3.25,0.0,8.13,46.3,0.0
3 1,baby back ribs,18.8,0.026,9.82,0.0,0.0
4 2,baklava,6.6,0.041,29.0,37.8,0.0013
5 3,beef carpaccio,18.5,0.02,6.0,4.2,0.002
6 4,beef tartare,17.4,0.028,15.6,0.34,0.0002
7 5,beet salad,1.32,0.018,5.29,11.0,0.4542
8 6,beignets,5.46,0.09,20.2,53.7,0.44210000000000005
9 7,bibimbap,5.51,0.025,2.92,8.18,0.4515
10 8,bread pudding,4.42,0.044,12.4,43.4,0.398
11 9,breakfast burrito,8.13,0.007,7.77,26.2,0.5342
12 10,bruschetta,4.29,0.05,7.61,21.3,0.5378999999999999
13 11,caesar salad,4.0,0.1,16.0,8.0,4.506
14 12,cannoli,6.32,0.021,14.7,42.1,0.0
15 13,caprese salad,5.76,0.079,10.5,9.42,0.0
16 14,carrot cake,2.5,0.025,26.2,50.0,0.0
17 15,ceviche,10.5,0.022,0.76,3.58,0.018600000000000002
18 16,cheese plate,6.46,0.146,11.5,15.6,0.2726
19 17,cheesecake,2.17,0.043,10.9,18.5,0.4415
20 18,chicken curry,6.47,0.02,6.45,6.47,0.44360000000000005
21 19,chicken quesadilla,10.6,0.133,5.31,15.0,0.2661
22 20,chicken wings,18.6,0.0,18.6,0.0,0.0
23 21,chocolate cake,3.03,0.03,22.7,57.6,0.0
24 22,chocolate mousse,1.19,0.036,23.8,19.0,0.0007
```

Hình 29. Kết quả file CSV

II. Đưa mô hình dự đoán lên web bằng Streamlit

Sau khi xây dựng mô hình nhận dạng thì lưu mô hình dưới dạng hdf5



Hình 30. Mô hình dùng nhận dạng

Sau đó load mô hình đã lưu bằng tensorflow và tệp nutrition101.csv được lấy từ gọi api phần trước. Sau khi có mô hình xây dựng hàm dự đoán kết quả:

```

def predicting(image, model):
    image = load_and_prep(image)
    image = tf.cast(tf.expand_dims(image, axis=0), tf.int16)
    preds = model.predict(image)
    pred_class = class_names[tf.argmax(preds[0])]
    pred_conf = tf.reduce_max(preds[0])
    top_5_i = sorted((preds.argsort())[0][-5:][::-1])
    values = preds[0][top_5_i] * 100
    labels = []
    for x in range(5):
        labels.append(class_names[top_5_i[x]])
    df = pd.DataFrame({"Top 5 Predictions": labels,
                      "Scores": values
                      })
    df = df.sort_values('Scores')
    return pred_class, pred_conf, df

```

Hình 31. Hàm dự đoán

Đầu vào có 2 tham số là mô hình và hình ảnh cần nhận dạng sau đó sử dụng phương thức predict của model để dự đoán, và chọn ra 5 loại thực phẩm có dự đoán cao nhất lưu vào dataframe, sử dụng kết quả cao nhất trong 5 loại thực phẩm đó để làm kết quả cuối cùng. Output bao gồm tên loại thực phẩm, phần trăm dự đoán thực phẩm đó và top 5 prediction.

Sau đó xây dựng các hàm vẽ đồ thị để trực quan hoá kết quả dự đoán cũng như thông tin giá trị dinh dưỡng của loại thực phẩm đó. Để vẽ đồ thị thì sử dụng thư viện plotly:

Vẽ đồ thị dự đoán:

```

0
1 def plotPrediction(labels ,values ):
2     y_saving = values
3     x = labels
4
5     fig = go.Figure()
6     fig.add_trace(go.Bar(
7         x=values,
8         y=labels,
9         marker=dict(
10             color='rgba(50, 171, 96, 0.6)',
11             line=dict(
12                 color='rgba(50, 171, 96, 1.0)',
13                 width=1),
14             ),
15         orientation='h',
16     ))
17     fig.update_layout(
18         title='<b>Top 5 Predictions</b>',
19         yaxis=dict(
20             showgrid=False,
21             showline=False,
22             showticklabels=True,
23             # domain=[0, 0.85],
24             tickfont = dict(size=15),
25         ),
26         xaxis_title="Percentage of prediction",
27         paper_bgcolor='rgb(248, 248, 255)',
28         plot_bgcolor='rgb(248, 248, 255)',
29         margin_pad=10,
30     )
31

```

Hình 32. Hàm vẽ đồ thị dự đoán

Vẽ đồ thị thông tin chi tiết món ăn:

```

def plotFoodInfo(namefood):
    foodinfo = nutritiondf[nutritiondf['name']==namefood].loc[:,['protein','calcium','fat','carbohydrates','vitamins']]
    labels = foodinfo.columns.to_numpy()
    values = foodinfo.values[0]
    # Create subplots: use 'domain' type for Pie subplot
    fig = make_subplots(rows=1, cols=2, specs=[[{'type':'domain'}], [{'type':'domain'}]])
    fig.add_trace(go.Pie(labels=labels, values=values))

    fig.update_layout(
        title_text="Nutrition Facts of "+namefood+" (100g)",
        margin =dict(l=3,r=3),
        font=dict(
            # family="Courier New, monospace",
            size=20
            # color="RebeccaPurple"
        ),
        xaxis=go.layout.XAxis(
            title=go.layout.xaxis.Title(
                text="Fruits<br><sup>Fruit sales in the month of January</sup>"
            )
        )
    )
    st.write(fig)
    urlname = namefood.replace(" ", "-")
    st.write(f"You can check out this [link](https://www.nutritionix.com/food/{urlname})")

```

Hình 33. Hàm vẽ thông tin thực phẩm

Ngoài ra còn xây dựng hàm vẽ đồ thị thể hiện thông tin của mô hình nhận diện cũng như độ sai số của mô hình để người dùng có cái nhìn trực quan hơn về mô hình.

Hàm vẽ lịch sử training mô hình:

```
def plotHistory(history_Load):  
    loss = history_Load['loss']  
    val_loss = history_Load['val_loss']  
  
    accuracy = history_Load['accuracy']  
    val_accuracy = history_Load['val_accuracy']  
  
    epochs = np.arange(0, len(history_Load['loss']))  
  
    fig = make_subplots(rows=1, cols=2, subplot_titles = ('Accuracy', 'Loss'))  
    fig.add_trace(  
        go.Scatter(x=epochs, y=accuracy, name='accuracy'),  
        row=1, col=1  
    )  
  
    fig.add_trace(  
        go.Scatter(x=epochs, y=val_accuracy, name='val_accuracy'),  
        row=1, col=1  
    )  
  
    fig.add_trace(  
        go.Scatter(x=epochs, y=loss, name='loss'),  
        row=1, col=2  
    )  
  
    fig.add_trace(  
        go.Scatter(x=epochs, y=val_loss, name='val_loss'),  
        row=1, col=2  
    )
```

Hình 34. Hàm vẽ lịch sử training mô hình

Hàm vẽ sai số của mô hình:

```
def plotF1(class_f1_scores):
    # class_f1_scores=np.load('./history/score_101.npy',allow_pickle='TRUE').item()
    report_df = pd.DataFrame(class_f1_scores, index = ['f1-scores']).T
    report_df = report_df.sort_values("f1-scores", ascending=False)
    class_names = report_df.index.values.tolist()
    fig, ax = plt.subplots(figsize=(12, 25))
    scores = ax.barh(range(len(report_df)), report_df["f1-scores"].values)
    ax.set_yticks(range(len(report_df)))
    plt.axvline(x=0.80, linestyle='--', color='r')
    ax.set_yticklabels(class_names)
    ax.set_xlabel("f1-score")
    ax.set_title("F1-Scores for 101 Different Classes")
    ax.invert_yaxis(); # reverse the order
    return fig
```

Hình 35. Hàm vẽ sai số mô hình

III. Kết quả chương trình

1. Giao diện ứng dụng và cách chạy

Để chạy ứng dụng dùng lệnh

streamlit run.\food-vision\app.py

```
PS C:\Users\baobo\Desktop\TLCN\food\food-vision>
PS C:\Users\baobo\Desktop\TLCN\food\food-vision> streamlit run .\food-vision\app.py

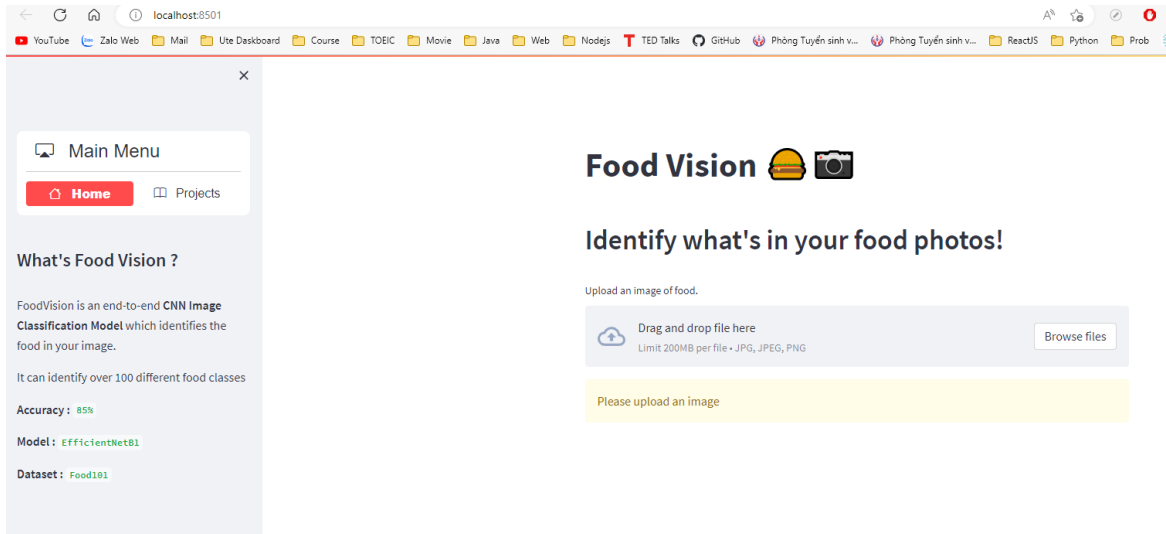
You can now view your Streamlit app in your browser.

Local URL: http://localhost:8501
Network URL: http://192.168.1.10:8501

□
```

Hình 36. Chạy ứng dụng

Sau đó truy cập vào <http://localhost:8501>



Hình 37. Giao diện

Giao diện của trang web giới thiệu sơ lược về ứng dụng và thông tin về model gồm tên model: EfficientNetB1 có độ chính xác: 85%. Main Menu gồm 2 phần:

- Home: Nhận dạng thực phẩm bằng hình ảnh
- Project: Thông tin chi tiết về model

2. Thông tin chi tiết về model

Để xem thông tin chi tiết về model ta chọn tab project của ứng dụng:



Multiclass Classification using TensorFlow 2.0 on Food-101 Dataset

The Food-101 Data Set :

- This dataset consists of 101 food categories, with 101'000 images
- Each type of food has 750 training samples and 250 test samples
- Size : 5 gb

TensorFlow Workflow :

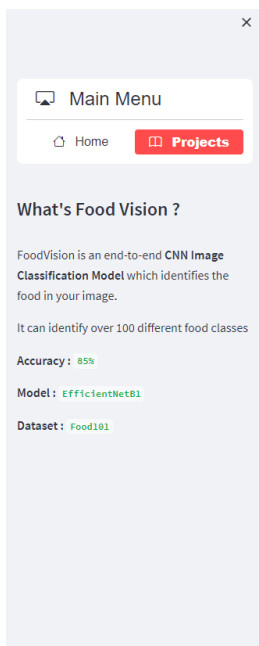


Preprocessing data :

- Load the image and convert into numpy array
- Resize the image (to the same size our model was trained on)
- Rescale the image (get all values between 0 and 1)

Hình 38. Thông tin mô hình

Trang web giới thiệu về tập dữ liệu dùng để training. Các bước để xây dựng model. Ngoài ra còn có thể xem tất cả các model mà nhóm xây dựng để nhận dạng thực phẩm:



Select Your Model:

2 Classes

2 Classes

10 Classes

101 Classes (final model)

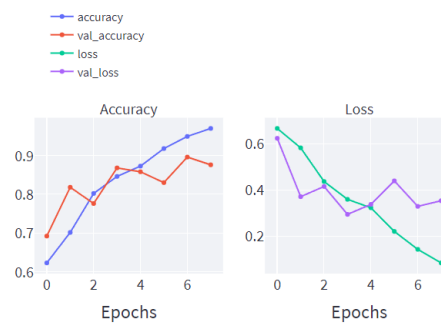
| Layer (type) | Output Shape | Param # |
|--------------------------------|----------------------|---------|
| conv2d (Conv2D) | (None, 222, 222, 10) | 280 |
| conv2d_1 (Conv2D) | (None, 220, 220, 10) | 910 |
| max_pooling2d (MaxPooling2D) | (None, 110, 110, 10) | 0 |
| conv2d_2 (Conv2D) | (None, 108, 108, 10) | 910 |
| conv2d_3 (Conv2D) | (None, 106, 106, 10) | 910 |
| max_pooling2d_1 (MaxPooling2D) | (None, 53, 53, 10) | 0 |
| flatten (Flatten) | (None, 28090) | 0 |

Hình 39. Thông tin cấu trúc mô hình



| | |
|-----------|---|
| epoch 3/8 | 6s 180ms/step - loss: 0.6655 - accuracy: 0.6227 - val_loss: 0.6228 - val_accuracy: 0.6528 |
| epoch 2/8 | 6s 180ms/step - loss: 0.6655 - accuracy: 0.6227 - val_loss: 0.6228 - val_accuracy: 0.6528 |
| epoch 1/8 | 6s 180ms/step - loss: 0.6655 - accuracy: 0.6227 - val_loss: 0.6228 - val_accuracy: 0.6528 |
| epoch 4/8 | 6s 180ms/step - loss: 0.6655 - accuracy: 0.6227 - val_loss: 0.6228 - val_accuracy: 0.6528 |
| epoch 5/8 | 6s 180ms/step - loss: 0.6655 - accuracy: 0.6227 - val_loss: 0.6228 - val_accuracy: 0.6528 |
| epoch 6/8 | 6s 180ms/step - loss: 0.6655 - accuracy: 0.6227 - val_loss: 0.6228 - val_accuracy: 0.6528 |
| epoch 7/8 | 6s 180ms/step - loss: 0.6655 - accuracy: 0.6227 - val_loss: 0.6228 - val_accuracy: 0.6528 |
| epoch 8/8 | 6s 180ms/step - loss: 0.6655 - accuracy: 0.6227 - val_loss: 0.6228 - val_accuracy: 0.6528 |

Evaluate the model :

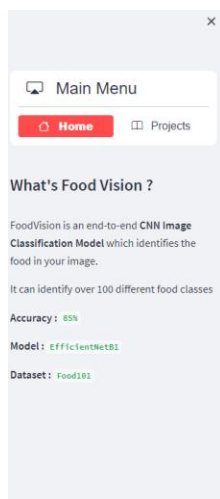


Hình 40. Thông tin đánh giá mô hình

Trang web sẽ hiện thị chi tiết cấu trúc của model, thời gian chạy từng bước cũng như đánh giá độ chính xác của mô hình.

3. Nhận dạng thực phẩm bằng hình ảnh

Để nhận dạng thực phẩm ta chọn tab home của ứng dụng sau đó chọn hình cần nhận dạng và upload lên web



Food Vision 🍔 📷

Identify what's in your food photos!

Upload an image of food.

Drag and drop file here

Limit 200MB per file • JPG, JPEG, PNG

Browse files

Please upload an image

Hình 41. Chọn hình ảnh

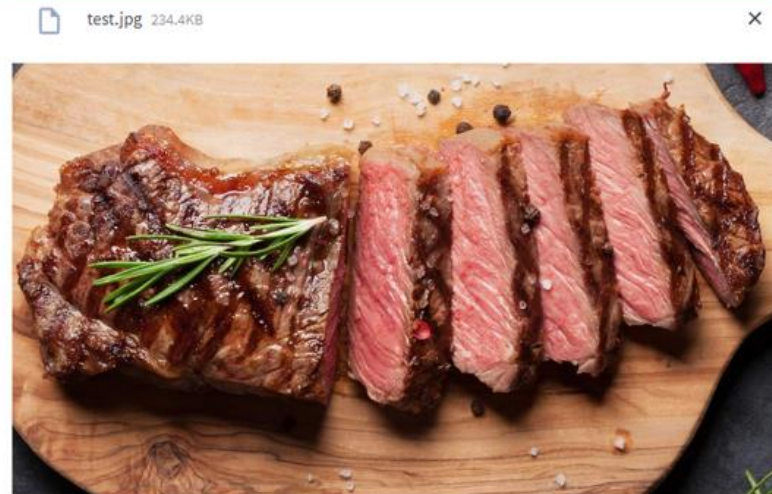
Sau đó nhấn vào nút predict để dự đoán

Identify what's in your food photos!

Upload an image of food.

 Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG

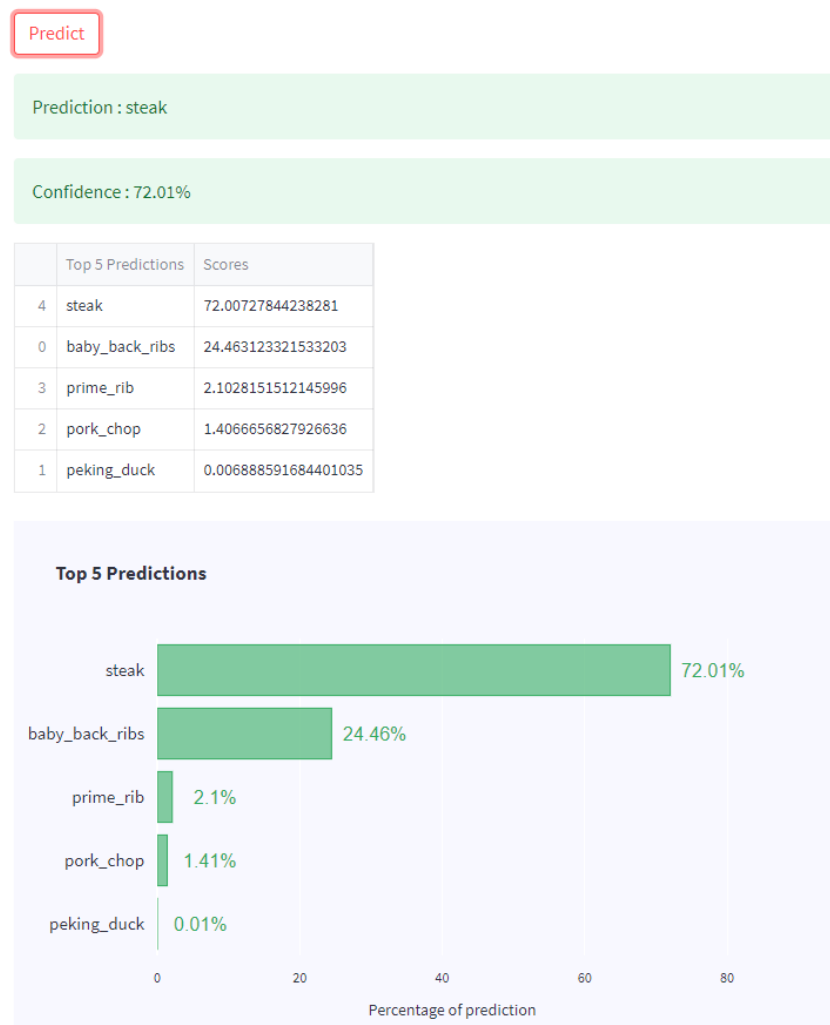
Browse files



Predict

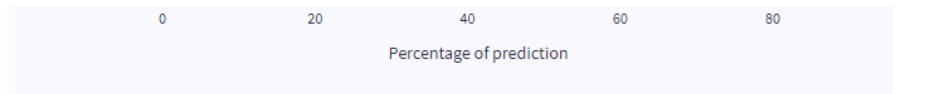
Hình 42. Hình ảnh cần nhận dạng

Kết quả:

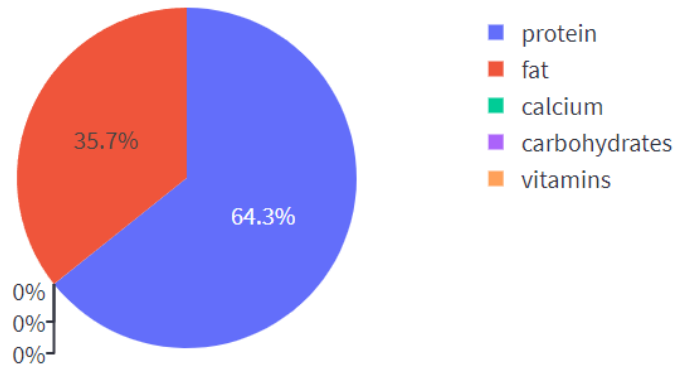


Hình 43. Kết quả nhận dạng

Ta thấy kết quả nhận dạng là steak với confidence là 72% và top 5 prediction. Sau khi dự đoán thì sẽ thì hiện thêm giá trị dinh dưỡng của thức ăn đó dựa vào trang <https://www.nutritionix.com/> và đưa lên web



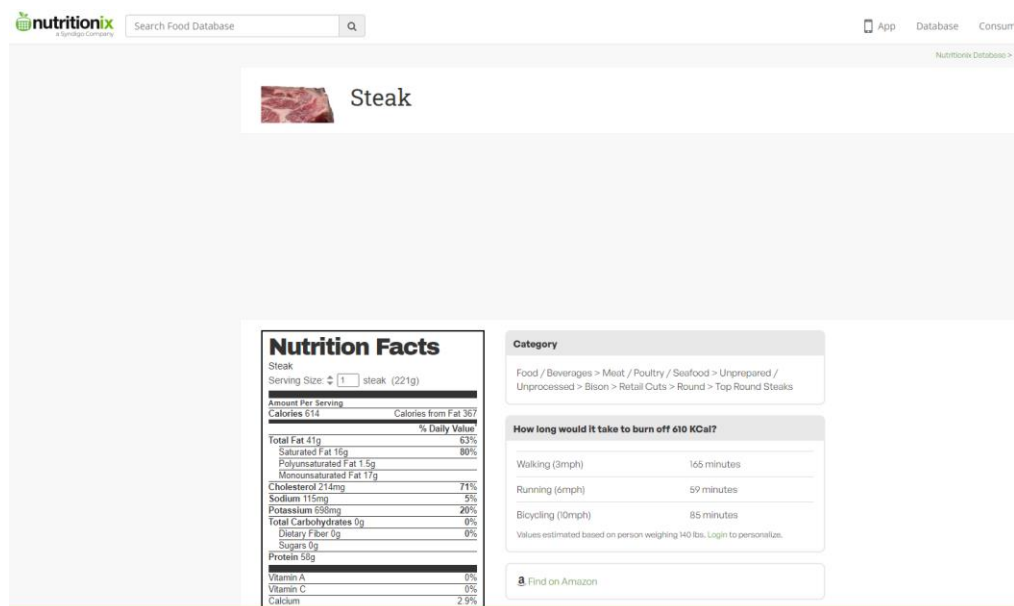
Nutrition Facts of steak (100g)



You can check out this [link](#)

Hình 44. Thông tin thực phẩm

Có thể xem thêm thông tin thực phẩm bằng cách nhấn vào link để xem chi tiết:



Hình 45. Thông tin thực phẩm trên nutrition

ĐÁNH GIÁ VÀ KẾT LUẬN

Kết quả đạt được

- Tìm hiểu được lý thuyết Deep learning, thuật toán CNN và mô hình EfficientNet
- Xây dựng được mô hình nhận dạng thức ăn với thuật toán EfficientNet-B1.
- Thiết kế giao diện web cơ bản bằng Streamlit để có thể sử dụng mô hình trên web và triển khai lên server.

Hạn chế

- Hạn chế về kỹ thuật nên thời gian huấn luyện khá lâu, chỉ train được đến EfficientNet-B1.
- Xử lý theo kiểu tuần tự.
- Đối với tập dữ liệu lớn hơn sẽ độ chính xác giảm.

Hướng phát triển

- Chuyển đổi sang xử lý song song trên hệ thống phân tán, huấn luyện với tập dữ liệu lớn hơn.
- Tích hợp thêm vào các phần mềm quản lý sức khỏe
- Nhận dạng bằng video, sử dụng trên điện thoại

TÀI LIỆU THAM KHẢO

mhadhbi, N. (2021, December 21). Python tutorial: Streamlit. DataCamp. Retrieved November 27, 2022, from <https://www.datacamp.com/tutorial/streamlit>

Python plotly tutorial. GeeksforGeeks. (2022, October 19). Retrieved November 27, 2022, from <https://www.geeksforgeeks.org/python-plotly-tutorial/>

Tutorials: Tensorflow Core. TensorFlow. (n.d.). Retrieved November 27, 2022, from <https://www.tensorflow.org/tutorials>

Viettelidc.com.vn. (n.d.). Retrieved November 27, 2022, from <https://viettelidc.com.vn/tin-tuc/cam-nang-ai-artificial-neural-network-la-gi-cau-truc-cach-hoat-dong-va-ung-dung-cua-mo-hinh-nay>