**ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**

**TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

**---------------------------------------**

**ĐỒ ÁN TỐT NGHIỆP**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

NGHIÊN CỨU VÀ ỨNG DỤNG MÔ HÌNH MOBILENETV2 TRÊN NỀN TẢNG DI ĐỘNG TRONG BÀI TOÁN PHÂN LOẠI HOA QUẢ

**GVHD: TS. Nguyễn Văn Tỉnh**

**Sinh viên: Nguyễn Hưng Thịnh**

**Mã sinh viên: 2021603670**

**Lớp: KHMT1 Khóa: K16**

Hà Nội – 2025

# MỤC LỤC

[MỤC LỤC 2](#_Toc197681498)

[DANH MỤC HÌNH ẢNH 4](#_Toc197681499)

[LỜI CẢM ƠN 6](#_Toc197681500)

[LỜI NÓI ĐẦU 1](#_Toc197681501)

[CHƯƠNG 1. TỔNG QUAN VỀ LÝ THUYẾT 3](#_Toc197681502)

[1.1. Tổng quan về học máy 3](#_Toc197681503)

[1.1.1. Phân loại học máy 4](#_Toc197681504)

[1.1.2. Các bước huấn luyện mô hình học máy 7](#_Toc197681505)

[1.1.3. Ứng dụng của học máy 8](#_Toc197681506)

[1.2. Tổng quan về CNN 9](#_Toc197681507)

[1.3. Tổng quan về Dart và framework Flutter 13](#_Toc197681508)

[1.3.1. Tổng quan về Dart 13](#_Toc197681509)

[1.3.2. Tổng quan về flutter 14](#_Toc197681510)

[1.4. Giới thiệu về công cụ 15](#_Toc197681511)

[1.4.1. Android Studio 15](#_Toc197681512)

[1.4.2. Pycharm 17](#_Toc197681513)

[1.4.3. TensorFlow 18](#_Toc197681514)

[CHƯƠNG 2. THIẾT KẾ HỆ THỐNG VÀ MÔI TRƯỜNG 20](#_Toc197681515)

[2.1. Cấu trúc mô hình MobileNetV2 20](#_Toc197681516)

[2.1.1. Inverted Residuals và LinearBottleNecks 20](#_Toc197681517)

[2.1.2. Depthwise Separable Convolution 21](#_Toc197681518)

[2.2. Chuẩn bị và phân tích dữ liệu 24](#_Toc197681519)

[2.2.1. Chuẩn bị dữ liệu 24](#_Toc197681520)

[2.2.2. Phân tích dữ liệu 25](#_Toc197681521)

[2.3. Tiền xử lý dữ liệu 28](#_Toc197681522)

[CHƯƠNG 3: TRIỂN KHAI HUẤN LUYỆN MÔ HÌNH 30](#_Toc197681523)

[3.1. Huấn luyện mô hình 30](#_Toc197681524)

[3.1.1. Khái niệm Fine tune 30](#_Toc197681525)

[3.1.2. Các kỹ thuật Fine tune phổ biến 31](#_Toc197681526)

[3.1.3. Fine tune mô hình 33](#_Toc197681527)

[3.2. Phân tích mô hình 38](#_Toc197681528)

[CHƯƠNG 4: TRIỂN KHAI ỨNG DỤNG DI ĐỘNG 39](#_Toc197681529)

[4.1. Ứng dụng mô hình trong thiết bị di động 39](#_Toc197681530)

[4.1.1. Chuyển đổi mô hình sang định dạng Tensorflow Lite 39](#_Toc197681531)

[4.1.2. Ứng dụng mô hình Tensorflow Lite trong ứng dụng di động 39](#_Toc197681532)

[4.2. Thử nghiệm trên thiết bị thực tế 44](#_Toc197681533)

[KẾT LUẬN 45](#_Toc197681534)

[TÀI LIỆU THAM KHẢO 46](#_Toc197681535)

# DANH MỤC HÌNH ẢNH

[Hình 1.1. Mối quan hệ giữa học máy, học sâu và trí tuệ nhân tạo 3](#_Toc197681536)

[Hình 1.2. Sơ đồ hoạt động học có giám sát. 5](#_Toc197681537)

[Hình 1.3. Sơ đồ hoạt động học không giám sát. 6](#_Toc197681538)

[Hình 1.4. Thuật toán học tăng cường. 7](#_Toc197681539)

[Hình 1.5. Kiến trúc các lớp CNN 9](#_Toc197681540)

[Hình 1.6. Kiến trúc mạng VGG16 11](#_Toc197681541)

[Hình 1.7. Kiến trúc mạng InceptionV3 12](#_Toc197681542)

[Hình 1.8. Kiến trúc mạng ResNet50 12](#_Toc197681543)

[Hình 1.9. Biểu tượng của ngôn ngữ Dart 13](#_Toc197681544)

[Hình 1.10. Biểu tượng của flutter 14](#_Toc197681545)

[Hình 1.11. Biểu tượng android studio 16](#_Toc197681546)

[Hình 1.12. Biểu tượng pycharm 17](#_Toc197681547)

[Hình 2.1. Inverted Residuals và Linear bottle necks trong MobileNetV2 20](#_Toc197681548)

[Hình 2.2. Cấu trúc Depthwise Separable Convolution 22](#_Toc197681549)

[Hình 2.3. Các nhãn và hình đại diện 25](#_Toc197681550)

[Hình 2.4. Kích thước min, max và trung bình trong bộ dữ liệu 25](#_Toc197681551)

[Hình 2.5. Sự phân bổ dữ liệu của dataset 26](#_Toc197681552)

[Hình 2.6. Chuẩn hoá dữ liệu tập huấn luyện 29](#_Toc197681553)

[Hình 3.1. Cách thức hoạt động của Fine tune 31](#_Toc197681554)

[Hình 3.2. Full fine-tuning 32](#_Toc197681555)

[Hình 3.3. Parameter efficient fine-tuning 32](#_Toc197681556)

[Hình 3.4. Khởi tạo mô hình MobileNetV2 34](#_Toc197681557)

[Hình 3.5. Mở khoá một vài lớp để tinh chỉnh mô hình 35](#_Toc197681558)

[Hình 3.6. Thiết kế đầu ra mô hình 35](#_Toc197681559)

[Hình 3.7. Biên dịch mô hình 36](#_Toc197681560)

[Hình 3.8. Hai hàm callback để hỗ trợ huấn luyện 37](#_Toc197681561)

[Hình 3.9. Huấn luyện mô hình 37](#_Toc197681562)

[Hình 4.1. Chuyển đổi sang file .tflite 39](#_Toc197681563)

[Hình 4.2. Tải mô hình từ bộ nhớ điện thoại 41](#_Toc197681564)

[Hình 4.3. Tiền xử lý ảnh trước khi phân loại 42](#_Toc197681565)

[Hình 4.4. Phân loại hình ảnh 43](#_Toc197681566)

[Hình 4.5. Demo màn hình phân loại 43](#_Toc197681567)

# LỜI CẢM ƠN

Với tình cảm sâu sắc và chân thành nhất, em xin được phép bày tỏ lòng biết ơn đến tất cả các cá nhân và tổ chức đã tạo điều kiện hỗ trợ, giúp đỡ em trong suốt quá trình học tập và nghiên cứu đề tài “*Nghiên cứu và ứng dụng mô hình MobileNetV2 trong bài toán phân loại hoa quả*”. Trong suốt thời gian từ khi bắt đầu học tập tại trường đến nay, em đã nhận được rất nhiều sự quan tâm, giúp đỡ của quý thầy cô và bạn bè. Với lòng biết ơn sâu sắc nhất, em xin gửi đến quý thầy cô ở khoa công nghệ thông tin đã truyền đạt vốn kiến thức quý báu cho em trong suốt thời gian học tập tại trường. Nhờ có những lời hướng dẫn, dạy bảo của các thầy, các cô nên đề tài tốt nghiệp: “*Nghiên cứu và ứng dụng mô hình MobileNetV2 trong bài toán phân loại hoa quả*” của em mới có thể hoàn thiện. Một lần nữa, em xin chân thành cảm ơn thầy Nguyễn Văn Tỉnh – người đã trực tiếp giúp đỡ, quan tâm, hướng dẫn em hoàn thành tốt bài báo cáo này trong thời gian qua. Bài báo cáo đồ án của em thực hiện trong khoảng thời gian 9 tuần. Bước đầu đi vào thực tế của em còn hạn chế và còn nhiều bỡ ngỡ nên không tránh khỏi những thiếu sót, em rất mong nhận được những ý kiến đóng góp quý báu của quý thầy cô để kiến thức của em trong lĩnh vực này được hoàn thiện hơn đồng thời có điều kiện bổ sung, nâng cao kỹ năng nghề nghiệp của em.

Em xin chân thành cảm ơn!

Nguyễn Hưng Thịnh

# LỜI NÓI ĐẦU

Trong thời đại hiện nay, với sự bùng nổ của công nghệ trí tuệ nhân tạo (AI) và học sâu (Deep Learning), việc ứng dụng các mô hình học máy vào các thiết bị di động ngày càng trở nên phổ biến. Những tiến bộ này đã mở ra nhiều cơ hội mới trong việc xây dựng các hệ thống thông minh, hỗ trợ con người trong nhiều lĩnh vực như y tế, giáo dục, nông nghiệp, và đời sống thường ngày.

Xuất phát từ thực tế đó, đề tài **"Nghiên cứu và ứng dụng mô hình MobileNetV2 vào thiết bị di động trong bài toán phân loại rau quả"** được lựa chọn nhằm mục tiêu nghiên cứu, xây dựng và triển khai một mô hình học sâu nhẹ, tối ưu, có khả năng phân loại chính xác các loại rau quả ngay trên thiết bị di động. Trong đề tài này, nhóm tác giả tập trung vào việc tìm hiểu cấu trúc của MobileNetV2 — một trong những kiến trúc mạng nơ-ron tích chập (CNN) nổi bật nhờ hiệu suất cao và dung lượng nhỏ, rất phù hợp cho các ứng dụng trên thiết bị tài nguyên hạn chế.

Đề tài không chỉ dừng lại ở việc nghiên cứu lý thuyết mà còn tiến hành huấn luyện mô hình với tập dữ liệu thực tế, tối ưu hóa hiệu suất thông qua các kỹ thuật như fine-tuning, chuyển mô hình sang định dạng TensorFlow Lite (TFLite), và xây dựng ứng dụng mẫu trên nền tảng di động để kiểm tra tính khả thi. Qua đó, đề tài góp phần chứng minh tiềm năng ứng dụng của các mô hình AI nhẹ trong việc hỗ trợ phân loại rau quả, đồng thời mở ra hướng phát triển cho các ứng dụng AI trong lĩnh vực nông nghiệp thông minh.

Trong quá trình thực hiện đề tài do giới hạn về thời gian và kinh nghiệm, chắc chắn sẽ không tránh khỏi những thiếu sót. Rất mong nhận được sự góp ý, đánh giá từ thầy cô và hội đồng chấm để đề tài được hoàn thiện tốt hơn.

Xin chân thành cảm ơn sự hướng dẫn tận tình của các thầy cô trong khoa Công nghệ Thông tin, đặc biệt là thầy hướng dẫn, đã hỗ trợ và tạo điều kiện hoàn thành đề tài này.

*Nội dung báo cáo gồm 4 chương:*

**Chương 1: Tổng quan về lý thuyết**

Từ những kiến thức thực tế, tổng hợp các lý thuyết về học máy và học sâu như: Các loại hình học máy, các bước xây dựng mô hình, tổng quan về CNN, các công nghệ sử dụng trong đồ án

**Chương 2: Thiết kế hệ thống và môi trường**

Thu thập dữ liệu phù hợp cho bài toán, phân tích và xử lý dữ liệu cho phù hợp với cấu trúc mô hình được sử dụng để huấn luyện mô hình

**Chương 3: Triển khai huấn luyện mô hình**

Từ những dữ liệu đã được tiền xử lý, cấu hình những thông số sao cho phù hợp để huấn luyên mô hình, phân tích những chỉ số như: Train loss, train accuracy, validation loss, validation accuracy, precision, recall, f1 score, …

**Chương 4: Triển khai ứng dụng di động**

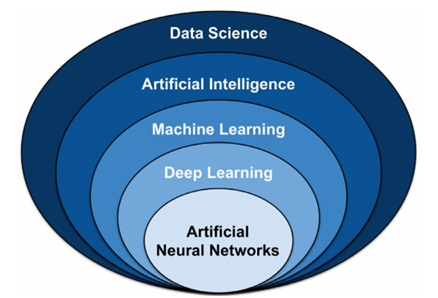
# CHƯƠNG 1. TỔNG QUAN VỀ LÝ THUYẾT

## Tổng quan về học máy

Học máy (Machine learning) là một lĩnh vực tập trung vào khía cạnh "học" của trí tuệ nhân tạo (Artificial intelligence) bằng cách phát triển các thuật toán có thể mô tả tốt nhất một tập dữ liệu.

Giả sử một công ty bất động sản muốn dự đoán giá của một căn nhà dựa trên các đặc điểm cụ thể của căn nhà đó. Trước tiên, công ty sẽ thu thập một tập dữ liệu bao gồm nhiều mẫu dữ liệu. Mỗi mẫu đại diện cho một quan sát duy nhất về một căn nhà cùng với các đặc điểm liên quan. Những đặc điểm này là các thuộc tính đã được ghi nhận của căn nhà, và có thể hữu ích để dự đoán giá (ví dụ: diện tích tổng thể, số tầng, có sân vườn hay không). **Target (mục tiêu)** là thuộc tính mà mô hình cần dự đoán — trong trường hợp này là **giá nhà**.

Thông thường, tập dữ liệu sẽ được chia thành ba phần: **training set (tập huấn luyện)**, **validation set (tập xác thực)** và **test set (tập kiểm tra)**, vì mô hình thường sẽ hoạt động tốt nhất trên dữ liệu mà nó đã được huấn luyện.

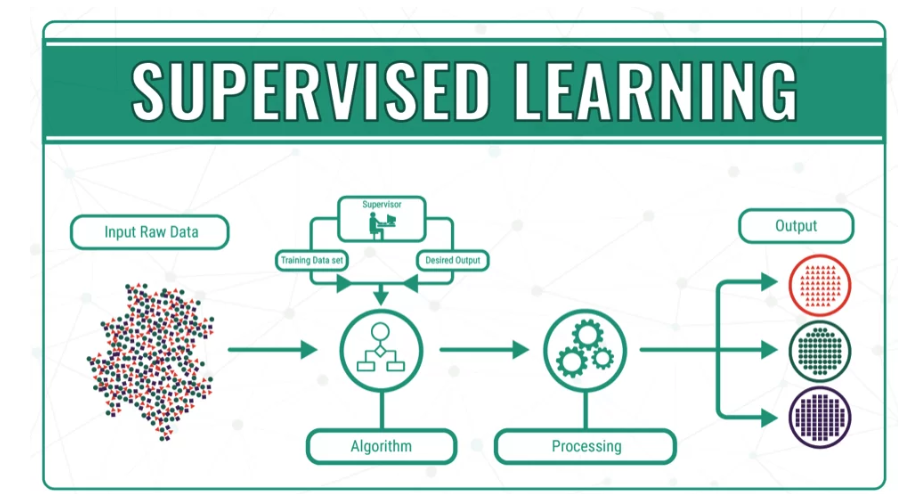


Hình 1.1. Mối quan hệ giữa học máy, học sâu và trí tuệ nhân tạo

### Phân loại học máy

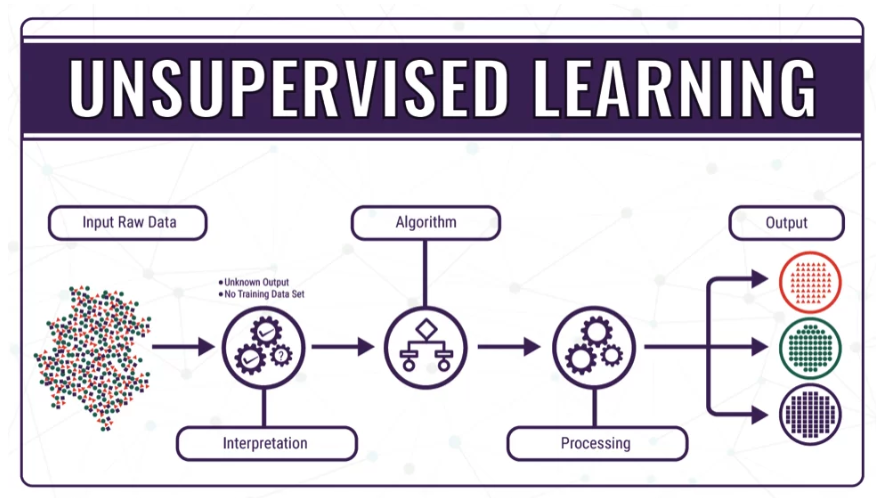
Học máy có thể được phân loại thành ba nhóm chính dựa trên cách thức dữ liệu được sử dụng trong quá trình huấn luyện và mục tiêu mà mô hình hướng đến. Ba nhóm chính này bao gồm **học giám sát** (supervised learning), **học không giám sát** (unsupervised learning), và **học tăng cường** (reinforcement learning). Mỗi loại có cách tiếp cận và ứng dụng khác nhau, tạo ra những điểm mạnh và hạn chế riêng biệt, tùy thuộc vào bài toán và dữ liệu cụ thể.

* Học có giám sát: Học giám sát là một trong những phương pháp phổ biến và cơ bản trong học máy. Trong phương pháp này, mô hình học từ dữ liệu có nhãn (labeled data), tức là mỗi ví dụ trong bộ dữ liệu huấn luyện có một nhãn đầu ra đã biết. Mục tiêu là huấn luyện một mô hình sao cho khi đưa vào dữ liệu mới, mô hình có thể dự đoán đúng nhãn đầu ra cho dữ liệu chưa được gắn nhãn
* **Quá trình huấn luyện học có giám sát**: Quá trình huấn luyện trong học giám sát thường bao gồm việc tối ưu hóa một hàm mất mát (loss function), trong đó mô hình cố gắng dự đoán nhãn đầu ra sao cho độ chệch giữa dự đoán và nhãn thực tế là nhỏ nhất.
* **Ứng dụng:** Học giám sát rất hữu ích trong các bài toán phân loại và hồi quy. Các bài toán phân loại yêu cầu phân nhóm dữ liệu vào các lớp khác nhau (ví dụ: nhận diện ảnh của các loài động vật). Các bài toán hồi quy yêu cầu dự đoán giá trị liên tục (ví dụ: dự đoán giá trị bất động sản, nhiệt độ, hay chỉ số chứng khoán).
* **Các thuật toán phổ biển:** Hồi quy tuyến tính, SVM (Supporrt vector machine), K lân cận (KNN),…



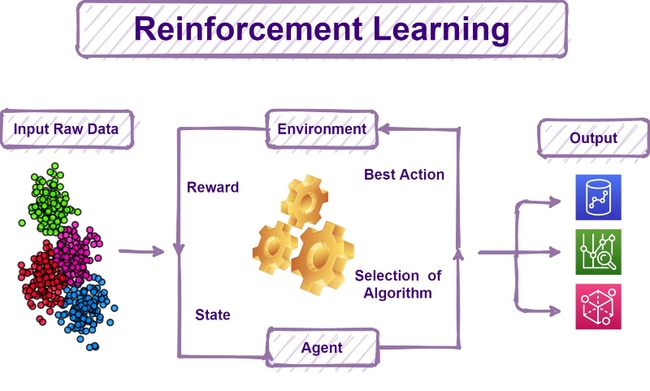
Hình 1.2. Sơ đồ hoạt động học có giám sát.

* Học không giám sát: Khác với học giám sát, học không giám sát sử dụng dữ liệu không có nhãn. Mục tiêu của học không giám sát là tìm ra các cấu trúc, mẫu hoặc sự phân bố ẩn trong dữ liệu mà không cần sự giám sát từ người dùng. Điều này đặc biệt hữu ích khi không có sẵn dữ liệu đã được gắn nhãn hoặc khi chúng ta muốn khám phá các đặc trưng tiềm ẩn trong tập dữ liệu.
* **Quá trình huấn luyện học không giám sát**: Trong học không giám sát, mô hình học tự động nhận diện các mẫu trong dữ liệu mà không cần biết trước các nhóm hay nhãn. Mô hình sẽ học cách phân nhóm hoặc giảm chiều dữ liệu để tìm ra các mối quan hệ giữa các dữ liệu.
* **Ứng dụng**: Học không giám sát thường được sử dụng trong các bài toán phân cụm, giảm chiều dữ liệu, và phát hiện bất thường. Ví dụ, trong phân tích dữ liệu khách hàng, học không giám sát có thể giúp phát hiện các nhóm khách hàng có hành vi tương tự. Trong xử lý ảnh, các kỹ thuật này có thể được sử dụng để giảm số lượng đặc trưng mà vẫn giữ lại các thông tin quan trọng.
* **Các thuật toán phổ biến**: Phân cụm K-means (K-means clustering), Naïve Bayes,…



Hình 1.3. Sơ đồ hoạt động học không giám sát.

* Học tăng cường: Học tăng cường là một phương pháp học máy mà trong đó một tác nhân (agent) học cách đưa ra các quyết định thông qua tương tác với môi trường và nhận phản hồi dưới dạng thưởng (reward) hoặc phạt (punishment). Mục tiêu của tác nhân là tối đa hóa tổng hợp phần thưởng trong suốt quá trình tương tác, qua đó học được cách hành động tối ưu.
* **Quá trình huấn luyện**: Trong học tăng cường, tác nhân sẽ thực hiện các hành động và nhận phần thưởng từ môi trường. Thông qua thử nghiệm và sai sót, tác nhân dần dần tìm ra chiến lược tốt nhất để đạt được mục tiêu dài hạn. Phương pháp này yêu cầu sự kết hợp giữa việc khai thác (exploitation) các kiến thức hiện tại và khám phá (exploration) các chiến lược mới.
* **Ứng dụng**: Học tăng cường có ứng dụng nổi bật trong các trò chơi, robot học, xe tự lái, và các hệ thống tối ưu hóa. Một trong những ví dụ thành công nhất của học tăng cường là AlphaGo của Google DeepMind, nơi AI đã học cách chơi cờ vây và đánh bại các kỳ thủ giỏi nhất thế giới.
* **Các thuật toán phổ biến**: Deep Q-Networks (DQN), Policy Gradient, …



Hình 1.4. Thuật toán học tăng cường.

### Các bước huấn luyện mô hình học máy

**Bước 1:** Thu thập và xử lý dữ liệu

Dữ liệu là yếu tố quan trọng trong học máy. Thu thập và xử lý dữ liệu là bước đầu tiên và quan trọng nhất. Dữ liệu phải được thu thập từ các nguồn đáng tin cậy và được tiền xử lý để đảm bảo rằng chúng có chất lượng cao và sẵn sàng để huấn luyện mô hình. Tiền xử lý dữ liệu bao gồm:

* Làm sạch dữ liệu
* Chuyển đổi dữ liệu về định dạng phù hợp
* Chuẩn hoá dữ liệu để cải thiện hiệu suất mô hình

**Bước 2:** Lựa chọn mô hình phù hợp cho bài toán

Lựa chọn mô hình học máy phù hợp với bài toán là rất quan trọng. Các mô hình khác nhau sẽ có hiệu suất khác nhau tùy thuộc vào bản chất của dữ liệu và bài toán cần giải quyết. Ví dụ, nếu bài toán yêu cầu phân loại ảnh, các mô hình học sâu như mạng nơ-ron tích chập (Convolutional Neural Networks - CNNs) sẽ hiệu quả hơn.

**Bước 3:** Huấn luyện mô hình

Sau khi chuẩn bị dữ liệu và chọn mô hình, bước tiếp theo là huấn luyện mô hình. Quá trình huấn luyện liên quan đến việc tối ưu hóa các tham số của mô hình sao cho mô hình có thể dự đoán chính xác nhất trên dữ liệu huấn luyện.

**Bước 4:** Đánh giá mô hình

Sau khi huấn luyện, mô hình cần được đánh giá để kiểm tra độ chính xác và khả năng tổng quát của nó trên dữ liệu chưa thấy trước đó. Các chỉ số đánh giá phổ biến bao gồm:

* Độ chính xác (Accuracy)
* Độ chính xác trung bình
* Mất mát (Loss)
* Độ nhạy và độ đặc hiệu

### Ứng dụng của học máy

Học máy đang ngày càng trở thành một công cụ thiết yếu trong thời đại số, với sự ứng dụng rộng rãi trong hầu hết mọi lĩnh vực của đời sống hiện đại. Các thuật toán học máy giúp tự động hóa quá trình ra quyết định, phát hiện mẫu, và xử lý lượng dữ liệu khổng lồ mà con người không thể làm được một cách thủ công. Dưới đây là một số lĩnh vực tiêu biểu mà học máy đã và đang được ứng dụng mạnh mẽ:

* Thị giác máy tính: Nhận diện ảnh, phân loại ảnh, nhận dạng ký tự và biển số xe, chẩn đoán y tế từ ảnh, …
* Xử lý ngôn ngữ tự nhiên: Dịch máy, chatbot và trợ lý ảo, phân tích cảm xúc, tóm tắt văn bản tự động, …
* Tài chính ngân hàng: Phát hiện gian lận, dự báo tài chính, đánh giá tín dụng, tư vấn tài chính cá nhân,…
* Giao thông và phương tiện tự hành: Xe tự đoán, dự đoán lưu lượng giao thông, hệ thống giao thông thông minh, ..

## Tổng quan về CNN

Trong lĩnh vực thị giác máy tính, mạng nơ-ron tích chập (CNN) về cơ bản là một mạng perceptron đa lớp (MLP) được chuẩn hóa và đã trở thành một trong những đổi mới có ảnh hưởng nhất. Trong những năm gần đây, CNN đã thống trị lĩnh vực thị giác máy tính và xử lý hình ảnh cho các tác vụ nhận dạng, phân loại và phân đoạn hình ảnh quy mô lớn. Một CNN điển hình bắt đầu với một lớp đầu vào và kết thúc với một lớp đầu ra, giữa chúng là nhiều lớp ẩn. Các lớp tích chập, pooling, chuẩn hóa (ReLU), và lớp kết nối đầy đủ là một phần của lớp ẩn. Hình 1.1 dưới đây cho thấy các lớp CNN điển hình.

Diagram of a diagram of a diagram of a diagram of a diagram of a diagram of a diagram of a diagram of a diagram of a diagram of a diagram of a diagram of a diagram of a

Description automatically generated

Hình 1.5. Kiến trúc các lớp CNN

Lớp đầu vào nhận dữ liệu hình ảnh mục tiêu làm đầu vào. Hình ảnh sau đó được thay đổi kích thước tối ưu và chuyển tiếp đến lớp tiếp theo là lớp tích chập. Tại đây, tồn tại một số lượng hạt nhân hoặc bộ lọc thực hiện phép nhân theo từng phần tử để trích xuất đặc trưng. Thông qua hàm kích hoạt, giá trị âm của đầu vào sẽ được thay thế bằng không, nếu không nó sẽ được chuyển trực tiếp đến đầu ra. Hàm kích hoạt được sử dụng rộng rãi nhất là ReLU (Rectified Linear Unit) và đối với nhiều loại mạng nơ-ron, nó là hàm kích hoạt mặc định. Đây là một hàm phi tuyến và nhanh hơn các hàm kích hoạt khác như Sigmoid, SELU (Scaled Exponential Linear Unit), ELU (Exponential Linear Unit), GELU (Gaussian Error Linear Unit) v.v. Các đặc trưng được trích xuất từ lớp tích chập sau đó được gửi đến lớp pooling. Lớp này chỉ giữ lại các đặc trưng quan trọng từ một hình ảnh lớn bằng cách giảm số lượng tham số. Sau đó, lớp kết nối đầy đủ dịch các hình ảnh đã được lọc kỹ lưỡng này thành các danh mục. Và một hàm phi tuyến khác tên là softmax cuối cùng sẽ cung cấp các xác suất thập phân từ 0 đến 1 cho mỗi lớp. Trong nghiên cứu này, một mạng nơ-ron tích chập 6 lớp được đề xuất và được xây dựng hoàn toàn từ đầu. Kích thước hình ảnh đầu vào được chọn là 32×32 để giảm thời gian tính toán tổng thể, giúp tạo ra một mô hình tốt về mặt hiệu quả. Các kỹ thuật tăng cường dữ liệu như xoay, thay đổi tỷ lệ, cắt xén, phóng to và lật ngang cũng được áp dụng cho dữ liệu hình ảnh huấn luyện có kích thước 32×32 và 3 kênh. ReLU đã được sử dụng làm hàm kích hoạt cho mỗi lớp tích chập. Và để cải thiện lỗi tổng quát hóa, một tỷ lệ dropout là 0,25 được sử dụng để khắc phục các vấn đề overfitting. Cuối cùng, softmax được sử dụng để tìm các xác suất của mỗi lớp dưới dạng số thập phân. Hình 1.2 cho thấy kiến trúc mô hình CNN đã được phát triển.

Các mô hình CNN phổ biến

* **Mô hình VGG16:** VGG16 là một kiến trúc mạng nơ-ron liên kết (CNN) đơn giản và được sử dụng rộng rãi được sử dụng cho ImageNet, một dự án cơ sở dữ liệu trực quan lớn được sử dụng trong nghiên cứu phần mềm nhận dạng đối tượng trực quan. Kiến trúc VGG16 được phát triển và giới thiệu bởi Karen Simonyan và Andrew Zisserman từ Đại học Oxford, vào năm 2014, thông qua bài báo của họ “Các mạng kết hợp rất sâu để nhận dạng hình ảnh quy mô lớn”. 'VGG' là tên viết tắt của Visual Geometry Group, là một nhóm các nhà nghiên cứu tại Đại học Oxford, những người đã phát triển kiến ​​trúc này, và '16' ngụ ý rằng kiến ​​trúc này có 16 lớp

A diagram of a graph

Description automatically generated

Hình 1.6. Kiến trúc mạng VGG16

* **InceptionV3:** InceptionV3 là một phiên bản cải tiến của kiến trúc Inception, được phát triển bởi nhóm nghiên cứu của Google. Mô hình này được giới thiệu lần đầu tiên trong bài báo khoa học có tên "Rethinking the Inception Architecture for Computer Vision," do Christian Szegedy và các cộng sự công bố vào năm 2015. InceptionV3 là một trong những bước phát triển tiếp theo của dòng Inception, được xây dựng dựa trên các phiên bản trước đó như InceptionV1 (GoogLeNet) và InceptionV2. Mô hình này đã tích hợp nhiều kỹ thuật tiên tiến để cải thiện hiệu suất và giảm số lượng tham số, như việc sử dụng các lớp tích chập phân nhánh (factorized convolutions), kỹ thuật batch normalization, và việc áp dụng các module Inception nhiều lớp. Mô hình InceptionV3 đã đạt được thành công đáng kể trong cuộc thi ILSVRC (ImageNet Large Scale Visual Recognition Challenge), nơi nó được sử dụng để phân loại hình ảnh với độ chính xác cao. Nó đã trở thành một trong những kiến trúc phổ biến nhất cho các nhiệm vụ liên quan đến nhận dạng hình ảnh và được ứng dụng rộng rãi trong nhiều lĩnh vực, từ thị giác máy tính cho đến học sâu.

A diagram of a block diagram

Description automatically generated

Hình 1.7. Kiến trúc mạng InceptionV3

* **ResNet50:** ResNet là một cấu trúc mạng do He Kaiming, Sun Jian và những người khác của Microsoft Research Asia đề xuất vào năm 2015 và đã giành được vị trí đầu tiên trong nhiệm vụ phân loại ILSVRC-2015. Đồng thời, nó đã giành được vị trí đầu tiên trong các nhiệm vụ phát hiện ImageNet, bản địa hóa ImageNet, A diagram of a block diagram

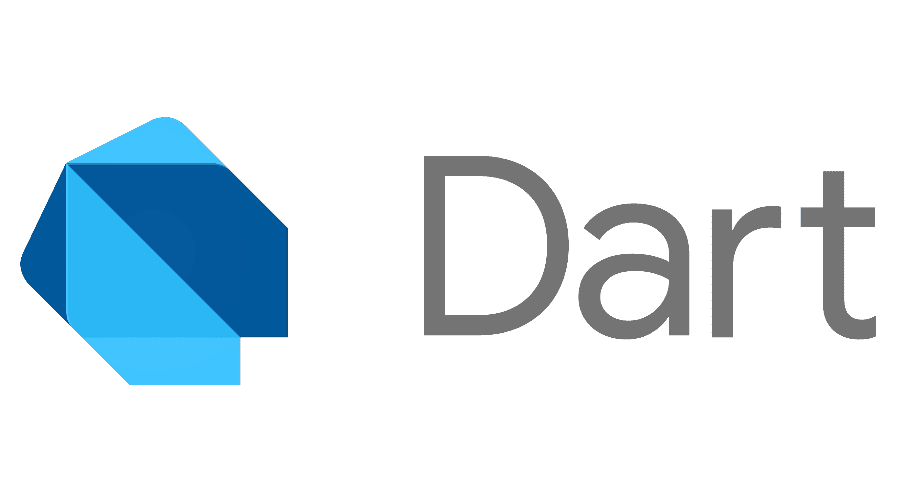
  Description automatically generatedphát hiện COCO và phân đoạn COCO.

Hình 1.8. Kiến trúc mạng ResNet50

* **MobileNetV2:** MobileNetV2 là một kiến trúc mạng nơ-ron tích chập (CNN) được phát triển bởi nhóm nghiên cứu của Google, cụ thể là nhóm Google Research – Brain Team. Kiến trúc này được công bố lần đầu tiên trong bài báo: "MobileNetV2: Inverted Residuals and Linear Bottlenecks". Tác giả: Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, Liang-Chieh Chen. Được công bố năm 2018, tại hội nghị CVPR (Conference on Computer Vision and Pattern Recognition).

## Tổng quan về Dart và framework Flutter

### Tổng quan về Dart



Hình 1.9. Biểu tượng của ngôn ngữ Dart

Dart, là một ngôn ngữ lập trình đa mục đích mã nguồn mở, đã được đặt nền móng bởi Google. Đây là một khía cạnh của ngôn ngữ lập trình hướng đối tượng, đặc trưng bởi cú pháp kiểu C. Tinh thần lập trình hướng đối tượng thể hiện qua việc hỗ trợ giao diện và lớp, mở ra khả năng sáng tạo không giới hạn khi đặt lên bàn cân với những ngôn ngữ khác. Sự đa dạng của Dart thể hiện thông qua khả năng phát triển ứng dụng web, di động, máy chủ và máy tính để bàn.

Trong lĩnh vực lập trình, Dart là đứa con tâm huyết từ Lars Bak và Kasper Lund, ra đời dưới sự quản lý của Google. Bắt đầu từ lần ra mắt tại hội nghị GOTO tại Đan Mạch vào ngày 10 tháng 12 tháng 10 năm 2011, Dart đã chứng tỏ tiềm năng vô hạn của mình. Một cột mốc quan trọng được ghi dấu bằng việc phát hành phiên bản Dart 1.0 vào ngày 14 tháng 11 năm 2013. Mặc dù ban đầu, Dart nhận được những ý kiến trái chiều, nhưng những phản hồi này không thể ngăn cản sự tiến bộ vượt trội của Dart.

Quãng thời gian này còn đánh dấu sự chuyển mình từ kế hoạch máy ảo Dart trong Chrome, sang việc biên dịch mã Dart thành JavaScript, điều quan trọng mà phiên bản Dart 2.0 đã mang lại vào tháng 8 năm 2018, cùng với những thay đổi ngôn ngữ kể cả hệ thống kiểu.

### Tổng quan về flutter



Hình 1.10. Biểu tượng của flutter

Flutter đã thu hút được sự chú ý của cộng đồng các nhà phát triển bằng cách giới thiệu các style cho phép việc xây dựng UI đẹp hơn và biểu cảm hơn vì thế mà việc code cũng trở nên thú vị hơn nhiều. Nó kết hợp một số khái niệm quen thuộc với những kinh nghiệm phát triển hiện đại như lập trình reactive và widget composition trong khi sử dụng nền tảng Dart làm cơ sở chính cho các hoạt động đó. Nhóm Flutter đã đánh giá nhiều ngôn ngữ khác nhau và cuối cùng họ chọn Dart vì nó phù hợp với cách mà họ xây dựng giao diện người dùng

Flutter có tính năng hot reload giúp bạn thử nghiệm, xây dựng UI, thêm tính năng và fix bug một cách nhanh chóng và dễ dàng hơn. Tính năng hot reload hoạt động bằng cách đưa những file source code đã được update vào máy ảo Virtual Machine (VM) của Dart. Sau khi VM cập nhập các class với phiên bản mới nhất của field và function, framework Flutter sẽ tự động tái xây dựng cây widget, cho phép bạn xem các hiệu ứng mà bạn đã thay đổi một cách nhanh chóng hơn.

Flutter cung cấp phân tích static cho phép bạn phát hiện ra những vấn đề ngay trước khi dòng code đó được thực thi. Nó thực sự là một công cụ mạnh mẽ có thể giúp các dev tránh việc phát sinh bug và đảm bảo được code tuân theo các quy tắc của style.

Ngôn ngữ Dart có trình biên dịch AOT (Ahead of Time) giúp nó biên dịch nhanh, đúng và native code. Điều này không chỉ đảm bảo giúp Flutter nhanh hơn mà còn đảm bảo rằng hầu như mọi thứ (bao gồm tất cả các widget) đều có thể được tùy chỉnh. Với lý do đó cho nên hầu hết các phần của Flutter đều được viết bằng ngôn ngữ này.

Với tất cả những cột mốc thành tựu, không thể phủ nhận vai trò và tiềm năng của Dart trong giới lập trình. Dù đã có những bước tiến, Dart vẫn tiếp tục hướng tới tương lai với sự đổi mới và sáng tạo không ngừng. Như một cây cầu nối giữa ý tưởng và hiện thực, Dart tiếp tục mở ra cơ hội cho sự sáng tạo và phát triển trong tương lai của lập trình.

## Giới thiệu về công cụ

### Android Studio



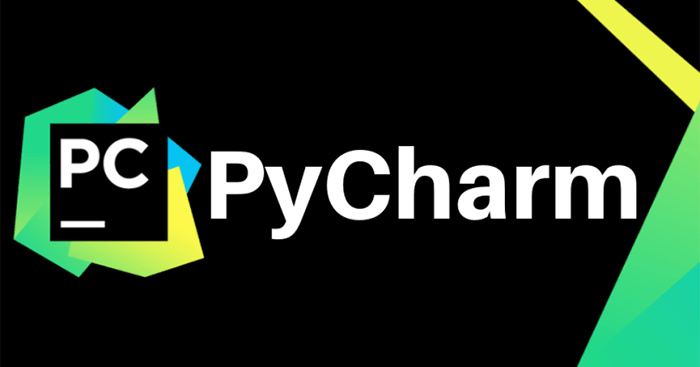
Hình 1.11. Biểu tượng android studio

Android Studio là môi trường phát triển tích hợp (IDE - Integrated Development Environment) được Google chính thức phát hành và hỗ trợ dành riêng cho việc phát triển ứng dụng Android. Với sự phát triển mạnh mẽ của Flutter – framework đa nền tảng của Google, Android Studio cũng cung cấp plugin để hỗ trợ lập trình Flutter hiệu quả.

**Tính năng nổi bật:**

* **Hỗ trợ đa nền tảng**: Dễ dàng phát triển ứng dụng cho Android, iOS (qua Flutter), máy tính để bàn và web.
* **Plugin Flutter & Dart**: Cung cấp giao diện thân thiện để tạo, chỉnh sửa và chạy ứng dụng Flutter trực tiếp trên trình giả lập hoặc thiết bị thật.
* **Code Editor mạnh mẽ**: Tích hợp các công cụ như tự động hoàn thành, kiểm tra lỗi cú pháp theo thời gian thực, và refactor code thông minh.
* **Android Emulator**: Cho phép mô phỏng thiết bị thật giúp dễ dàng kiểm thử ứng dụng mà không cần thiết bị vật lý.
* **Visual Layout Editor**: Với Flutter, có thể tận dụng các plugin hỗ trợ dựng UI bằng drag & drop hoặc theo mô hình Widget.
* **Tích hợp với Git và các hệ thống quản lý mã nguồn khác**, giúp việc làm việc nhóm trở nên dễ dàng và chuyên nghiệp hơn.

### Pycharm



Hình 1.12. Biểu tượng pycharm

**PyCharm** là một IDE nổi tiếng chuyên dụng cho ngôn ngữ lập trình Python, được phát triển bởi JetBrains – hãng phần mềm nổi tiếng với các công cụ phát triển như IntelliJ IDEA, WebStorm,... Với sự phát triển mạnh mẽ của Python trong lĩnh vực khoa học dữ liệu và trí tuệ nhân tạo, PyCharm trở thành công cụ không thể thiếu với các lập trình viên AI.

**Tính năng nổi bật:**

* **Hỗ trợ Python toàn diện**: Hệ thống gợi ý code, tự động sửa lỗi, định dạng lại mã nguồn và hỗ trợ debug nâng cao.
* **Quản lý môi trường ảo (virtual environment)**: Dễ dàng cô lập các thư viện và phiên bản Python để tránh xung đột giữa các dự án.
* **Tích hợp tốt với thư viện ML/AI**: Như TensorFlow, Keras, scikit-learn, NumPy, Pandas, giúp lập trình viên dễ dàng viết và huấn luyện mô hình.
* **Hỗ trợ Jupyter Notebook**: Cho phép tương tác với dữ liệu, trực quan hóa kết quả và thử nghiệm mô hình dễ dàng.
* **Tích hợp hệ thống version control**: Quản lý mã nguồn thông qua Git, GitHub hoặc GitLab.

Trong đồ án này, PyCharm được sử dụng để xây dựng và huấn luyện mô hình học sâu (Deep Learning) sử dụng thư viện TensorFlow. Mô hình CNN (Convolutional Neural Network) được huấn luyện để phân loại các loại hoa quả. Sau khi huấn luyện, mô hình sẽ được chuyển sang định dạng .tflite để triển khai trên thiết bị di động thông qua Flutter.

### TensorFlow

**TensorFlow** là một thư viện mã nguồn mở do Google phát triển và phát hành lần đầu tiên vào năm 2015. TensorFlow nhanh chóng trở thành một trong những nền tảng hàng đầu cho việc phát triển các ứng dụng học máy (Machine Learning - ML) và học sâu (Deep Learning - DL). TensorFlow hỗ trợ từ quá trình xây dựng mô hình, huấn luyện, đánh giá cho đến triển khai mô hình trên nhiều nền tảng khác nhau như web, máy chủ, thiết bị di động

**Tính năng nổi bật:**

* **Mô hình toán học linh hoạt**: Cho phép xây dựng các mô hình từ đơn giản đến phức tạp, như mạng nơ-ron tích chập (CNN), mạng tuần tự (RNN), attention model, transformer,...
* **Khả năng mở rộng cao**: Có thể huấn luyện mô hình trên CPU, GPU hoặc TPU, tăng tốc hiệu suất đáng kể.
* **TensorBoard**: Công cụ trực quan hóa giúp theo dõi quá trình huấn luyện như loss, accuracy, histogram, learning rate,...
* **Tích hợp dễ dàng với Keras**: Cho phép xây dựng mô hình với cú pháp thân thiện, dễ sử dụng.
* **Hỗ trợ TensorFlow Lite (TFLite)**: Giúp chuyển đổi mô hình sang định dạng nhẹ, tối ưu hóa để chạy trên các thiết bị di động hoặc nhúng.
* **Mô hình pretrained**: Có sẵn nhiều mô hình đã được huấn luyện trên các tập dữ liệu lớn, giúp tiết kiệm thời gian huấn luyện và cải thiện hiệu quả.

# CHƯƠNG 2. THIẾT KẾ HỆ THỐNG VÀ MÔI TRƯỜNG

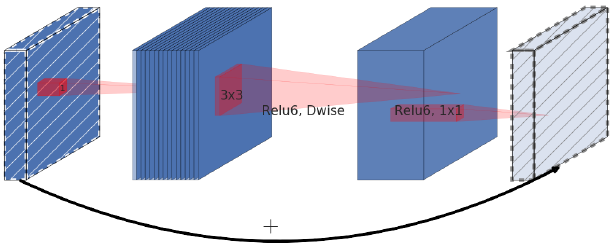
## Cấu trúc mô hình MobileNetV2

MobileNetV2 là phiên bản cải tiến từ mô hình MobileNetV1, với nhiều thay đổi sâu sắc trong cấu trúc kiến trúc để nâng cao hiệu quả tính toán, đặc biệt là trong các bài toán thị giác máy (computer vision) như nhận dạng hình ảnh, phân loại vật thể, phát hiện đối tượng, và thậm chí trong các mô hình kết hợp thời gian thực.

Không giống với những cấu trúc như VGG, InceptionNet (GoogleNet), ResNet,…vốn tập trung vào việc mở rộng chiều sâu và độ rộng của mạng để tăng khả năng trích xuất đặc trưng. MobileNet nói chung và MobileNetV2 nói riêng hướng tới tối ưu hóa hiệu suất trên thiết bị đầu cuối mà không đánh đổi đáng kể độ chính xác. Các thành phần nổi bật

### Inverted Residuals và LinearBottleNecks

Đây là yếu tố giúp mô hình vừa **nhẹ** vừa **giữ được độ chính xác cao**, đặc biệt phù hợp với các thiết bị di động hoặc hệ thống tính toán giới hạn tài nguyên.



Hình 2.1. Inverted Residuals và Linear bottle necks trong MobileNetV2

Khối residual là ý tưởng ban đầu đến từ ResNet, trong đó đầu vào được cộng với đầu ra của một chuỗi lớp (residual connection) giúp giảm hiện tượng mất gradient khi huấn luyện các mô hình sâu. Tuy nhiên, trong MobileNetV2, thay vì giữ nguyên chiều không gian và số kênh như ResNet, thì **Inverted Residual Block lại làm điều ngược lại**:

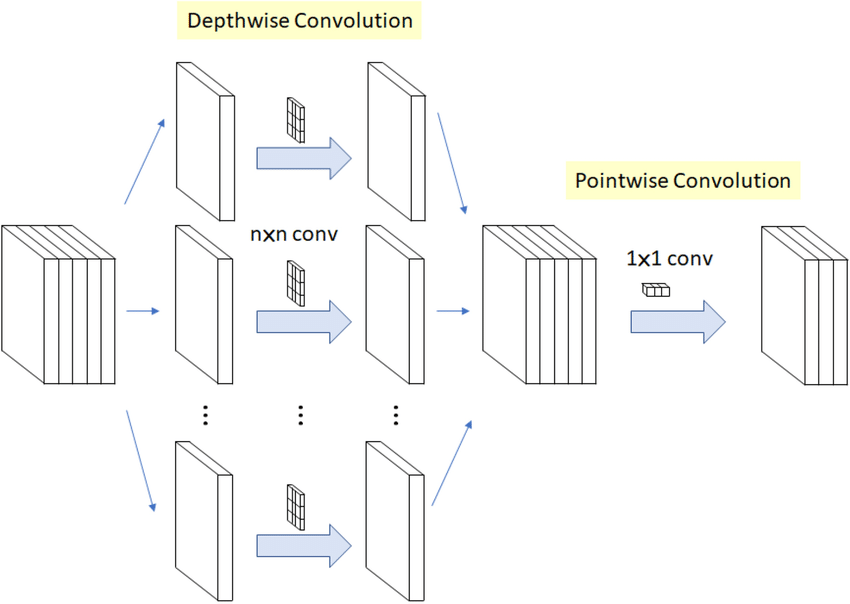
* Đầu vào có **số kênh nhỏ**, trước tiên được **mở rộng (expansion)** bằng một lớp **1x1 convolution** (thường tăng số kênh lên 6 lần).
* Sau đó, áp dụng lớp **depthwise convolution** để trích xuất đặc trưng không gian.
* Cuối cùng, sử dụng một lớp **projection 1x1** để **nén số kênh trở lại** giống đầu vào ban đầu.

Một đặc điểm quan trọng tiếp theo là **không dùng hàm kích hoạt phi tuyến (ReLU)** ở cuối mỗi khối residual. Thay vào đó, phần cuối là một lớp **tuyến tính (linear)**. Hàm ReLU, dù giúp mạng học tốt hơn và tránh gradient vanish, nhưng nếu áp dụng sau một tầng có ít chiều (bottleneck), nó sẽ **làm mất thông tin**, gây hỏng đặc trưng đầu ra. Vì vậy, nhóm nghiên cứu của Google đã chứng minh rằng **sử dụng tầng tuyến tính** thay cho ReLU sau lớp bottleneck giúp giữ nguyên thông tin đặc trưng và tăng độ chính xác của mô hình.

### Depthwise Separable Convolution

Kiến trúc MobileNetV2 sử dụng một kỹ thuật gọi là **Depthwise Separable Convolution** để giảm số lượng tham số và chi phí tính toán một cách đáng kể so với convolution truyền thống. Depthwise Separable Convolution gồm hai bước:

* **Depthwise Convolution**:  
  Thay vì áp dụng bộ lọc 3D (height × width × channels) trên toàn bộ đầu vào, depthwise conv chỉ áp dụng **mỗi bộ lọc 2D cho từng kênh đầu vào riêng biệt**. Nếu đầu vào có 32 kênh, thì sẽ có 32 bộ lọc độc lập, mỗi bộ lọc xử lý một kênh riêng.
* **Pointwise Convolution (1×1 Conv)**:  
  Sau bước lọc từng kênh, ta sử dụng một lớp **1×1 conv** để kết hợp thông tin giữa các kênh lại với nhau. Đây là bước tổng hợp các đặc trưng đã được lọc ở bước trước.



Hình 2.2. Cấu trúc Depthwise Separable Convolution

So sánh Depthwise Separable Convolution và Standard Convolution:

|  |  |  |
| --- | --- | --- |
| **Tiêu chí** | Standard Convolution | Depthwise Separable Convolution |
| Cơ chế hoạt động | Áp dụng bộ lọc 3D (filter size: *K×K×M*) cho toàn bộ đầu vào để tạo ra từng kênh đầu ra. | Gồm hai bước: Depthwise convolution và Pointwise convolution |
| Số lượng tham số (Parameters) | Cao. Số tham số = *K×K×M×N* | Thấp. Số tham số = *K×K×M* (depthwise) + *M×N* (pointwise) |
| Tốc độ xử lý | Chậm hơn. | Nhanh, phù hợp với các thiết bị có tài nguyên hạn chế |
| Mục tiêu | Hiệu suất cao, độ chính xác tốt. | Giảm chi phí tính toán, phục vụ triển khai trên mobile hoặc embedded systems. |
| Ứng dụng | Dùng trong các mạng lớn như VGG, ResNet, EfficientNet, UNet… | Rất phù hợp cho mạng nhẹ: MobileNet, ShuffleNet, EfficientNet-Lite, EdgeTPU, TFLite models… |

**Kết luận:** Hai đặc trưng chính là **Inverted Residuals với Linear Bottlenecks** và **Depthwise Separable Convolution** là cốt lõi giúp kiến trúc MobileNetV2 **nhẹ hơn, nhanh hơn**, nhưng vẫn **hiệu quả** trong nhiều bài toán nhận dạng ảnh, đặc biệt trên nền tảng hạn chế tài nguyên như điện thoại, nhúng, IoT.

## Chuẩn bị và phân tích dữ liệu

### Chuẩn bị dữ liệu

Bộ dữ liệu cho bài toán phân loại rau củ có nguồn gốc từ bài báo nghiên cứu “DCNN-Based Vegetable Image Classification Using Transfer Learning: A Comparative Study” của M. Israk Ahmed và các cộng sự, và được lưu trữ tại trang web kaggle.com.

Bộ dữ liệu gồm 21000 ảnh của 15 loại rau củ, mỗi loại có 1400 ảnh. Tương ứng là các nhãn: bean, bitter gourd, bottle gourd, brinjal, broccoli, cabbage, capsicum, carrot, cauliflower, cucumber, papaya, potato, pumpkin, radish và tomato. Bộ dữ liệu có định dạng là ảnh màu RGB, phần mở rộng \*.jpg, và kích cỡ 224×224.

Để phục vụ cho quá trình thực nghiệm, mỗi loại rau củ bộ dữ liệu sẽ được chia đều thành 15000 ảnh cho training, 3000 ảnh cho validation, và 3000 ảnh cho test. Quá trình huấn luyện sẽ sử dụng tập training và validation, quá trình đánh giá và nhận xét sẽ sử dụng tập test. Loại nhãn sẽ được chuyển thành dạng EndHotCode

### Phân tích dữ liệu

In ra một ảnh trong bộ dữ liệu của 15 loại rau củ để xem dữ liệu trông như thế nào và kiểm tra xem bộ dữ liệu có bị gán nhãn sai hay không:



Hình 2.3. Các nhãn và hình đại diện

Kiểm tra lại kích thước các ảnh xem có giống như trong mô tả không:

A screenshot of a computer screen

AI-generated content may be incorrect.

Hình 2.4. Kích thước min, max và trung bình trong bộ dữ liệu

Kiểm tra sự phân bổ dữ liệu, tránh tình trạng bộ dữ liệu bị mất cân bằng làm mô hình có thể bị Overfitting:

A blue and white graph

AI-generated content may be incorrect.

Hình 2.5. Sự phân bổ dữ liệu của dataset

Kết luận: Trước hết, xét về phương diện phân bố dữ liệu giữa các lớp, bộ dữ liệu bao gồm 16 lớp rau củ khác nhau, trong đó mỗi lớp đều có số lượng ảnh tương đương nhau, xấp xỉ 1000 ảnh. Điều này cho thấy bộ dữ liệu có tính **cân bằng cao**, nghĩa là không có lớp nào chiếm ưu thế về số lượng quá lớn hay quá nhỏ so với các lớp còn lại. Việc phân bố đồng đều này là một lợi thế rất lớn trong quá trình huấn luyện mô hình, bởi nó giúp mô hình học được các đặc trưng của từng lớp một cách công bằng, tránh hiện tượng mô hình bị thiên lệch (bias) về các lớp có nhiều dữ liệu hơn, vốn là một nguyên nhân phổ biến gây ra hiện tượng **overfitting** ở các lớp phổ biến và **underfitting** ở các lớp hiếm gặp trong các bộ dữ liệu mất cân bằng.

Bên cạnh yếu tố phân bố, kích thước ảnh trong bộ dữ liệu cũng được xử lý thống nhất ở mức 224x224 pixels và đều có 3 kênh màu RGB. Việc chuẩn hóa kích thước ảnh như vậy có ý nghĩa rất lớn, giúp mô hình học sâu hoạt động hiệu quả và giảm thiểu được thời gian tiền xử lý dữ liệu trước khi huấn luyện. Đây là kích thước tiêu chuẩn thường được sử dụng với nhiều kiến trúc CNN nổi tiếng như VGG, ResNet, hay đặc biệt là MobileNetV2 mà đồ án này đang áp dụng. Việc giữ nguyên độ phân giải ảnh ở mức vừa phải như vậy cũng giúp cân bằng giữa tốc độ huấn luyện và khả năng nhận dạng đặc trưng hình ảnh.

Ngoài ra, một yếu tố quan trọng khác trong chất lượng dữ liệu là **tính đa dạng và thực tế của hình ảnh**. Qua hình ảnh minh họa trực quan của từng lớp rau củ trong bộ dữ liệu, có thể thấy rằng ảnh được chụp từ nhiều góc độ khác nhau, trong các điều kiện ánh sáng và phông nền đa dạng. Một số ảnh có hậu cảnh đơn giản như nền trắng, trong khi nhiều ảnh khác lại được chụp trong môi trường thực tế như chợ, siêu thị hoặc ngoài trời. Điều này tạo ra sự phong phú về bối cảnh, giúp mô hình học được các đặc trưng không phụ thuộc vào điều kiện chụp, từ đó cải thiện **khả năng tổng quát hóa (generalization)** khi áp dụng mô hình cho các hình ảnh thực tế bên ngoài tập huấn luyện.

Tóm lại, bộ dữ liệu sử dụng trong bài toán phân loại rau củ được đánh giá là một tập dữ liệu có chất lượng tốt với các đặc điểm nổi bật như: phân bố lớp đồng đều, ảnh được chuẩn hóa kích thước, độ phân giải phù hợp, có tính đa dạng cao và phản ánh sát với hình ảnh thực tế. Những yếu tố này đóng vai trò quan trọng trong việc hỗ trợ mô hình học sâu đạt được hiệu quả cao, hạn chế các vấn đề như overfitting hoặc underfitting, đồng thời đảm bảo khả năng ứng dụng tốt khi triển khai trên thiết bị di động hoặc môi trường ngoài phòng thí nghiệm.

## Tiền xử lý dữ liệu

Trong quá trình xây dựng mô hình học sâu cho bài toán phân loại rau củ, bước tiền xử lý dữ liệu đóng vai trò vô cùng quan trọng, ảnh hưởng trực tiếp đến hiệu quả huấn luyện và khả năng tổng quát của mô hình. Dữ liệu hình ảnh đầu vào cần được chuẩn hóa và xử lý một cách hợp lý để đảm bảo phù hợp với kiến trúc mạng nơ-ron sử dụng (MobileNetV2) và tăng cường khả năng học của mô hình.

Các ảnh đầu vào có kích thước và tỉ lệ khác nhau, do đó cần được **resize về kích thước chuẩn 224x224 pixel** để tương thích với kiến trúc MobileNetV2. Bên cạnh đó, mô hình chỉ nhận ảnh màu, do đó tất cả ảnh đều được chuyển về định dạng RGB với 3 kênh màu.

Sau khi chuẩn hóa kích thước và kênh màu, ảnh được chuẩn hóa giá trị pixel về khoảng [0, 1] bằng cách chia mỗi giá trị cho 255. Điều này giúp mô hình học nhanh hơn và ổn định hơn vì tránh được hiện tượng gradient lớn nhỏ bất thường.

Đối với tập huấn luyện, ngoài việc chuẩn hóa, còn áp dụng các kỹ thuật **tăng cường dữ liệu (data augmentation)** nhằm giúp mô hình học được nhiều đặc trưng khác nhau của cùng một đối tượng, từ đó tăng khả năng tổng quát hóa. Các kỹ thuật tăng cường dữ liệu bao gồm:

* **Xoay ảnh (rotation\_range=20)**: mô phỏng việc đối tượng được chụp ở góc nghiêng.
* **Dịch chuyển chiều rộng và chiều cao (width\_shift\_range=0.2, height\_shift\_range=0.2)**: mô phỏng sự thay đổi vị trí vật thể trong ảnh.
* **Biến dạng hình học (shear\_range=0.2)** và **phóng to/thu nhỏ (zoom\_range=0.2)**: giúp mô hình thích ứng với các thay đổi hình học thực tế.
* **Lật ngang ảnh (horizontal\_flip=True)**: tạo thêm dữ liệu đối xứng nhằm tăng độ đa dạng.
* **Điền vùng trống (fill\_mode='nearest')**: sử dụng giá trị pixel gần nhất để điền vào các vùng bị khuyết sau khi biến đổi.

A screenshot of a computer code

AI-generated content may be incorrect.

Hình 2.6. Chuẩn hoá dữ liệu tập huấn luyện

Các kỹ thuật này không được áp dụng cho tập validation và test nhằm đảm bảo rằng dữ liệu kiểm tra phản ánh chính xác hiệu suất thật của mô hình mà không bị ảnh hưởng bởi biến đổi nhân tạo.

Với ImageDataGenerator, mỗi batch ảnh sẽ được sinh ra ngẫu nhiên theo đúng phân phối lớp, đồng thời đảm bảo áp dụng đầy đủ các kỹ thuật tăng cường (đối với tập huấn luyện). Tập validation và test được sinh ra mà không áp dụng augmentation, chỉ áp dụng chuẩn hóa.

# CHƯƠNG 3: TRIỂN KHAI HUẤN LUYỆN MÔ HÌNH

## Huấn luyện mô hình

### 3.1.1. Khái niệm Fine tune

Fine-tuning là quá trình điều chỉnh một mô hình đã được pre-trained trên một tập dữ liệu nhỏ hơn, có mục tiêu cụ thể nhằm tối ưu hóa hiệu suất cho các tác vụ hoặc trường hợp sử dụng cụ thể. Fine-tuning có thể được coi là một tập hợp con của transfer learning, cho phép tận dụng kiến thức mà mô hình hiện có đã học được làm điểm khởi đầu để tiếp thu kiến thức mới.

Về cơ bản, việc trau dồi khả năng của một base model đã được pre-trained sẽ dễ và rẻ hơn xây dựng và đào tạo một mô hình hoàn toàn mới từ đầu. Điều này đặc biệt hiệu quả với các mô hình deep learning phức tạp, có hàng triệu hoặc thậm chí hàng tỷ tham số, như các mô hình ngôn ngữ lớn (Large language models) trong lĩnh vực xử lý ngôn ngữ tự nhiên hoặc các Convolutional Neural Networks (CNN) và Vision Transformers (ViTs) được sử dụng cho các tác vụ thị giác máy tính.

Fine-tuning bắt đầu với một mô hình pretrained đã được đào tạo trên một bộ dữ liệu lớn, đa dạng, học được nhiều đặc điểm và mẫu khác nhau. Trong quá trình tiền huấn luyện, mô hình đã học cách tổng quát hóa bằng cách xác định các mẫu và đặc điểm cơ bản trong dữ liệu đào tạo, giúp nó có thể diễn giải chính xác các đầu vào mới.

Để bắt đầu fine-tuning, nhà phát triển mô hình xây dựng hoặc chọn một tập dữ liệu nhỏ hơn, chuyên biệt nhắm vào trường hợp sử dụng cụ thể. Các bộ dữ liệu này truyền đạt kiến thức miền cụ thể, phong cách hoặc tác vụ mà mô hình đang được fine-tuned. Ví dụ:

* Một LLM pre-trained cho ngôn ngữ chung có thể được fine-tuned cho lập trình với một bộ dữ liệu mới chứa các yêu cầu lập trình và đoạn mã tương ứng
* Một mô hình phân loại hình ảnh có thể học các loài chim mới thông qua các mẫu đào tạo có nhãn bổ sung

Sau khi thu thập dữ liệu, nhà phát triển tiếp tục đào tạo mô hình pretrained. Thông thường, các lớp đầu của neural network, nắm bắt các đặc điểm cơ bản như kết cấu đơn giản trong hình ảnh hoặc vector embeddings trong văn bản, được “đóng băng” (giữ nguyên). Các lớp sau được điều chỉnh hoặc thêm vào để nắm bắt dữ liệu mới và phù hợp hơn với tác vụ hiện tại.

A diagram of a model of a function

AI-generated content may be incorrect.

Hình 3.1. Cách thức hoạt động của Fine tune

## 3.1.2. Các kỹ thuật Fine tune phổ biến

* Full fine-tuning: Full fine-tuning là phương pháp đơn giản nhất về mặt khái niệm, đơn giản là cập nhật toàn bộ neural network. Để tránh những thay đổi mất ổn định, một số hyperparameter có thể được điều chỉnh so với thông số kỹ thuật trong quá trình pre-training. Ví dụ, sử dụng learning rate nhỏ hơn (làm giảm độ lớn của mỗi cập nhật đối với trọng số mô hình) ít có khả năng dẫn đến catastrophic forgetting.

A diagram of a training process

AI-generated content may be incorrect.

Hình 3.2. Full fine-tuning

* Parameter efficient fine-tuning: Parameter efficient fine-tuning (PEFT) bao gồm một loạt các phương pháp để giảm số lượng tham số cần được cập nhật để thích ứng hiệu quả một mô hình pre-trained lớn cho các ứng dụng downstream cụ thể. PEFT giảm đáng kể tài nguyên tính toán và bộ nhớ lưu trữ cần thiết. Các phương pháp PEFT thường ổn định hơn các phương pháp full fine-tuning, đặc biệt là cho các trường hợp sử dụng NLP.

A computer and monitor with text below

AI-generated content may be incorrect.

Hình 3.3. Parameter efficient fine-tuning

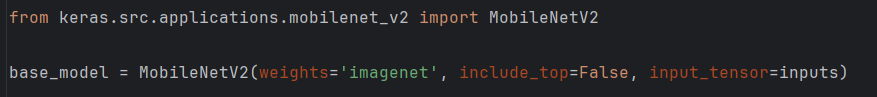
* Partial fine-tuning: Còn được gọi là selective fine-tuning, các phương pháp partial fine-tuning nhằm giảm yêu cầu tính toán bằng cách chỉ cập nhật tập con được chọn của các tham số đã pre-trained quan trọng nhất đối với hiệu suất mô hình. Các tham số còn lại được “đóng băng”, đảm bảo rằng chúng sẽ không bị thay đổi. Cách tiếp cận partial fine-tuning trực quan nhất là chỉ cập nhật các lớp ngoài của neural network. Trong hầu hết các kiến trúc mô hình, các lớp bên trong (gần với lớp đầu vào nhất) chỉ nắm bắt các đặc điểm rộng, chung chung. Ví dụ, trong một CNN được sử dụng để phân loại hình ảnh, các lớp đầu tiên thường phân biệt các cạnh và kết cấu; mỗi lớp tiếp theo phân biệt các đặc điểm ngày càng tinh tế hơn cho đến khi phân loại cuối cùng được dự đoán ở lớp ngoài cùng.
* Additive fine-tuning: Thay vì fine-tuning các tham số hiện có của một mô hình đã pre-trained, các phương pháp additive thêm tham số hoặc lớp bổ sung vào mô hình, đóng băng các trọng số pre-trained hiện có và chỉ đào tạo các thành phần mới đó. Cách tiếp cận này giúp duy trì sự ổn định của mô hình bằng cách đảm bảo rằng các trọng số pre-trained ban đầu vẫn không thay đổi. Mặc dù điều này có thể làm tăng thời gian đào tạo, nhưng nó làm giảm đáng kể yêu cầu bộ nhớ vì có ít gradients và trạng thái tối ưu hóa hơn để lưu trữ. Theo Lialin và cộng sự, việc đào tạo tất cả tham số của một mô hình đòi hỏi bộ nhớ GPU nhiều hơn 12–20 lần so với chỉ riêng trọng số mô hình. Tiết kiệm bộ nhớ hơn nữa có thể đạt được thông qua quantization của các trọng số mô hình đã đóng băng, tương tự như việc giảm bitrate của một file âm thanh.

### 3.1.3. Fine tune mô hình

Sau khi lựa chọn mô hình MobileNetV2 làm nền tảng cho bài toán phân loại rau củ, tiến hành quá trình huấn luyện và tinh chỉnh mô hình (fine-tuning) để mô hình có thể thích nghi và học tốt đặc trưng từ tập dữ liệu riêng. Việc huấn luyện mô hình không chỉ đơn thuần là đưa dữ liệu vào và tính toán, mà cần phải trải qua quá trình thiết lập mô hình hợp lý, đóng băng trọng số các lớp phù hợp, mở khóa các tầng cần thiết để tinh chỉnh, lựa chọn các thông số huấn luyện hiệu quả và theo dõi sát sao quá trình học để đạt được kết quả tối ưu.

MobileNetV2 là một trong những kiến trúc mạng nơ-ron tích chập hiệu quả và gọn nhẹ, được thiết kế tối ưu cho các thiết bị di động hoặc hệ thống có tài nguyên hạn chế. Thay vì huấn luyện lại toàn bộ mô hình từ đầu (tốn kém thời gian và tài nguyên), sử dụng mô hình MobileNetV2 đã được huấn luyện sẵn trên tập dữ liệu lớn ImageNet, với hàng triệu ảnh và 1.000 lớp. Việc tận dụng lại trọng số từ ImageNet giúp mô hình có khả năng trích xuất đặc trưng hình ảnh tốt ngay từ đầu.

Mô hình được khởi tạo bằng lệnh:



Hình 3.4. Khởi tạo mô hình MobileNetV2

Trong đó:

* Weights = ‘imagenet’: Sử dụng các trọng số từ tập dữ liệu ImageNet
* include\_top = False: loại bỏ các lớp fully-connected cuối cùng để có thể thêm vào phần phân loại riêng cho bài toán phân loại rau củ.

Với việc tinh chỉnh mô hình tận dụng lại các trọng số từ ImageNet, ban đầu sẽ đóng băng toàn bộ các lớp ở trong mô hình gốc để bảo toàn các đặc trưng đã học được. Việc đóng băng giúp mô hình tránh học lại từ đầu, đồng thời làm giảm số lượng tham số cần cập nhật, tiết kiệm thời gian huấn luyện và giảm nguy cơ overfitting

Sau đó, mở khóa một số lớp cuối cùng trong mô hình để thực hiện tinh chỉnh mô hình, tức là cho phép các lớp này học lại từ dữ liệu mới. Điều này giúp mô hình điều chỉnh các đặc trưng phù hợp hơn với tập dữ liệu rau củ. Trong đoạn mã, số lớp được mở khóa được xác định bởi tham số n\_layers\_to\_unfreeze và nhóm có thể thay đổi giá trị này để thử nghiệm các mức độ fine-tuning khác nhau.

A computer screen shot of text

AI-generated content may be incorrect.

Hình 3.5. Mở khoá một vài lớp để tinh chỉnh mô hình

Do mô hình gốc MobileNetV2 được thiết kế cho tập ImageNet (1.000 lớp), nên phần đầu ra của mô hình cần được thiết kế lại để phù hợp với số lớp mới (15 loại rau củ). Nhóm đã thêm vào các lớp phân loại (classification head) như sau:

A screen shot of a computer program

AI-generated content may be incorrect.

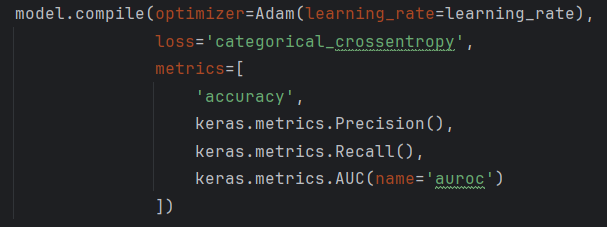
Hình 3.6. Thiết kế đầu ra mô hình

Trong đó:

* GlobalAveragePooling2D( ): Chuyển đầu ra của mô hình thành vector đặc trưng bằng cách tính trung bình theo không gian.
* Dense(64, activation='relu'): lớp fully-connected với 64 node và hàm kích hoạt ReLU, kết hợp với regularization L2 để giảm overfitting.
* BatchNormalization: giúp ổn định quá trình huấn luyện bằng cách chuẩn hóa đầu ra của lớp trước đó.
* Dropout(0.1): ngẫu nhiên bỏ qua 10% node trong quá trình huấn luyện nhằm tăng khả năng tổng quát.
* Dense(15, activation='softmax'): lớp đầu ra với số node tương ứng với số lớp cần phân loại, sử dụng hàm kích hoạt softmax để chuẩn hóa xác suất.

Sau đó cấu hình huấn luyện mô hình. Mô hình được biên dịch (compile) với các thông số như sau:

* **Optimizer**: Adam – một trong những thuật toán tối ưu hiệu quả nhất hiện nay, được sử dụng rộng rãi nhờ khả năng thích nghi tốt với các loại dữ liệu và tốc độ hội tụ nhanh.
* **Learning rate**: tốc độ học được truyền từ tham số đầu vào, cho phép điều chỉnh linh hoạt qua từng lần thử nghiệm.
* **Loss function**: categorical\_crossentropy – phù hợp với bài toán phân loại nhiều lớp với nhãn one-hot.
* **Metrics**: bao gồm các chỉ số để đánh giá hiệu năng mô hình: accuracy, precision, recall và AUC.



Hình 3.7. Biên dịch mô hình

Thêm một số hàm call back hỗ trợ cho việc huấn luyện mô hình:

A computer screen shot of a program

AI-generated content may be incorrect.

Hình 3.8. Hai hàm callback để hỗ trợ huấn luyện

* **EarlyStopping**: tự động dừng huấn luyện nếu val\_loss không giảm sau 5 epoch liên tiếp. Điều này giúp tránh việc mô hình học quá lâu dẫn đến overfitting.
* **ReduceLROnPlateau**: giảm learning rate nếu val\_loss không cải thiện trong 2 epoch, giúp mô hình tiếp tục học ở tốc độ chậm hơn và vượt qua các vùng cực tiểu cục bộ.

Sau đó, tiến hành huấn luyện mô hình:

A screen shot of a computer code

AI-generated content may be incorrect.

Hình 3.9. Huấn luyện mô hình

Kết quả huấn luyện được lưu lại trong biến history, bao gồm các chỉ số loss và accuracy qua từng epoch. Sau khi huấn luyện hoàn tất, mô hình được lưu lại dưới định dạng .keras để phục vụ cho việc triển khai trên nền tảng di động

## Phân tích mô hình

A graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

AI-generated content may be incorrect.

sssssssssssssssssssssssssssssssssssss

A graph with numbers and a bar chart

AI-generated content may be incorrect.

# CHƯƠNG 4: TRIỂN KHAI ỨNG DỤNG DI ĐỘNG

## Ứng dụng mô hình trong thiết bị di động

### 4.1.1. Chuyển đổi mô hình sang định dạng Tensorflow Lite

Sau khi huấn luyện và đánh giá mô hình phân loại rau củ trên tập dữ liệu huấn luyện, validation và kiểm tra, bước tiếp theo là triển khai mô hình này lên nền tảng di động. Việc triển khai mô hình học sâu lên thiết bị di động giúp người dùng cuối có thể sử dụng chức năng phân loại rau củ mọi lúc mọi nơi, không cần kết nối Internet và không phụ thuộc vào máy chủ từ xa.

Mô hình sau khi huấn luyện bằng Keras (model.keras) cần được chuyển đổi sang định dạng .tflite để sử dụng trên di động. TensorFlow Lite là phiên bản rút gọn và tối ưu hóa của TensorFlow, hỗ trợ inference nhanh, chiếm ít bộ nhớ và tương thích với nhiều nền tảng nhúng.

Quá trình chuyển đổi sử dụng công cụ TFLiteConverter như sau:

A computer screen with white and green text

AI-generated content may be incorrect.

Hình 4.1. Chuyển đổi sang file .tflite

Kết quả thu được là một file model.tflite có kích thước nhỏ hơn nhiều lần so với mô hình gốc, sẵn sàng để tích hợp vào ứng dụng Flutter.

### 4.1.2. Ứng dụng mô hình Tensorflow Lite trong ứng dụng di động

Có 3 thư viện chính để xử lý phân loại hình ảnh trên Pub.dev:

* Tflite\_flutter: Thư viện tflite\_flutter là cầu nối giữa ứng dụng Flutter và nền tảng TensorFlow Lite, cho phép tải và thực thi các mô hình học sâu đã được tối ưu hóa. Thư viện hỗ trợ thao tác với Interpreter, thực hiện suy luận (inference) trực tiếp trên thiết bị mà không cần kết nối mạng. Đây là thành phần cốt lõi để tích hợp mô hình .tflite vào ứng dụng di động.
* Image\_picker: Thư viện image\_picker cung cấp giao diện truy cập thư viện ảnh và camera của thiết bị di động. Thông qua thư viện này, người dùng có thể dễ dàng chụp ảnh trực tiếp hoặc chọn ảnh từ thư viện để đưa vào mô hình phân loại. Đây là công cụ quan trọng giúp kết nối giữa người dùng và chức năng nhận dạng hình ảnh trong ứng dụng.
* Image: Thư viện image là một công cụ xử lý ảnh thuần Dart, hỗ trợ các thao tác như resize, crop, convert định dạng ảnh, hoặc trích xuất dữ liệu điểm ảnh dưới dạng ma trận. Đây là thư viện thiết yếu để xử lý và chuẩn hóa ảnh đầu vào trước khi đưa vào mô hình, đảm bảo kích thước và định dạng phù hợp với yêu cầu của TensorFlow Lite.

Đầu tiên sử dụng thư viện tflite\_flutter để tải mô hình từ bộ nhớ trong (Local):

A computer screen shot of a program code

AI-generated content may be incorrect.

Hình 4.2. Tải mô hình từ bộ nhớ điện thoại

Mô hình .tflite được đặt trong thư mục assets của dự án Flutter. Khi ứng dụng khởi động, mô hình được tải vào bộ nhớ bằng Interpreter.fromAsset(). Đây là bước khởi tạo bộ suy luận.

Sau khi mô hình được tải thành công, ứng dụng có thể truy cập các thông tin như hình dạng của tensor đầu vào và đầu ra để đảm bảo quá trình chuẩn bị dữ liệu là chính xác.  
 Sau đó, tiền xử lý hình ảnh trước khi phân loại và phân loại hình ảnh:

Để mô hình hoạt động chính xác, ảnh đầu vào cần được xử lý sao cho phù hợp với yêu cầu của mô hình MobileNetV2, cụ thể là:

* Resize ảnh về kích thước cố định 224×224 pixels
* Chuyển đổi điểm ảnh từ định dạng RGB sang dạng mảng float32
* Chuẩn hóa từng giá trị điểm ảnh về khoảng [0, 1] bằng cách chia cho 255

A screen shot of a computer program

AI-generated content may be incorrect.

Hình 4.3. Tiền xử lý ảnh trước khi phân loại

Sau khi chuẩn bị xong ảnh đầu vào, mô hình thực hiện phân loại thông qua phương thức interpreter.run(). Đầu ra là một mảng có dạng [1, 15], tương ứng với xác suất thuộc về 15 lớp rau củ khác nhau. Nhãn có xác suất cao nhất được chọn làm kết quả cuối cùng. Người dùng sẽ nhận được tên loại rau củ được phân loại chính xác nhất từ ảnh họ cung cấp.

A screen shot of a computer code

AI-generated content may be incorrect.

Hình 4.4. Phân loại hình ảnh

Demo màn hình phân loại:

A screenshot of a phone

AI-generated content may be incorrect. A screenshot of a phone

AI-generated content may be incorrect.A screenshot of a phone

AI-generated content may be incorrect.

Hình 4.5. Demo màn hình phân loại

Người dùng tải ảnh từ thư viện lên hoặc chụp ảnh từ camera, sau đó ấn nút classify, app sẽ phân loại ảnh và hiển thị ra màn hình chi tiết của nhãn đó.

## Thử nghiệm trên thiết bị thực tế

# KẾT LUẬN

# TÀI LIỆU THAM KHẢO

1. Lập trình hướng đối tượng Java – Nguyễn Văn Thành chủ biên.
2. Phạm Hữu Khang - Lập trình cơ sở dữ liệu tập II: thiết kế cơ sở dữ liệu, tạo các thủ tục xử lý.
3. Tham khảo tài liệu qua mạng Internet: trang web laptrinhjavaweb.com, youtube.com