

BỘ NÔNG NGHIỆP VÀ PHÁT TRIỂN NÔNG THÔN
PHÂN HIỆU TRƯỜNG ĐẠI HỌC THỦY LỢI
KHOA: CÔNG NGHỆ THÔNG TIN
BỘ MÔN: KHAI PHÁ DỮ LIỆU

-----oOo-----



BÁO CÁO
Đề tài: PHÂN LOẠI 102 LOÀI HOA

GVHD: Vũ Thị Hạnh

Sinh viên thực hiện	Mã số sinh viên
Nguyễn Hoàng Dương	2251068183
Vũ Mạnh Thuyết	2251068259
Võ Hồng Sơn	2251068243

Hồ Chí Minh, tháng 10 năm 2025

Mục Lục

1. Giới thiệu đề tài.....	6
1.1 Tập dữ liệu	6
1.2 Tầm quan trọng của vấn đề	6
1.3 Phạm vi	6
2. Mục tiêu và bài toán đặt ra	8
2.1. Mục tiêu nghiên cứu	8
2.2. Bài toán đặt ra	9
2.3. Ý nghĩa thực tiễn của đề tài	10
3. Mô tả dữ liệu và các bước tiền xử lý	11
3.1. Tổng quan về bộ dữ liệu	11
3.2. Đặc điểm và thách thức của dữ liệu	11
3.3. Cấu trúc thư mục dữ liệu	12
3.4. Các bước tiền xử lý dữ liệu	12
3.5. Các kỹ thuật(EarlyStopping,ReduceLROnPlateau,ModelCheckpoint)	14
3.5.1. EarlyStopping	14
3.5.2. ReduceLROnPlateau	14
3.5.3. ModelCheckpoint	15
3.6. Kết quả sau tiền xử lý	15
3.7. Nhận xét	15
4. Phương pháp khai phá dữ liệu / mô hình ML	16
4.1. Mô hình MobileNetV2	16
4.1.1. Giới thiệu mô hình MobileNetV2	16
4.1.2. Nguyên lý hoạt động và cấu trúc của MobileNetV2	16
4.1.3 Quy trình huấn luyện mô hình	18

4.1.4. Các kỹ thuật hỗ trợ huấn luyện	19
4.1.5. Quá trình đánh giá mô hình	20
4.2. Mô hình ResNet50	20
4.2.1 ResNet50 là gì?	20
4.2.2 Kiến trúc mô hình ResNet50	20
4.2.3 Huấn luyện mô hình ResNet50	21
4.3. Mô hình EfficientNetB0	23
4.3.1. Giới thiệu mô hình EfficientNetB0	23
4.3.2. Cấu trúc và nguyên lý hoạt động	24
4.3.3. Quy trình huấn luyện mô hình	25
4.3.4. Các kỹ thuật hỗ trợ huấn luyện	27
5. Kết quả và đánh giá mô hình	28
5.1 Mô hình ResNet50	28
5.1.2 Độ chính xác	28
5.1.3 Biểu đồ training	29
5.1.4 Ma trận nhầm lẫn	31
5.2 Mô hình EfficientNetB0	32
5.2.1 Độ chính xác	32
5.2.2 Biểu đồ training	32
5.2.3 Ma trận nhầm lẫn	33
5.3 Mô hình MoblieNetV2	33
5.3.1 Độ chính xác	33
5.3.2 Biểu đồ training	34
5.3.3 Ma trận nhầm lẫn	35
6. Kết luận và hướng phát triển	36

6.1 Kết luận	36
6.2 Hướng phát triển	36
7. Tài liệu tham khảo	37

Mục Lục Hình Ảnh

Hình 3.4 Bước 4 tăng cường dữ liệu	13
Hình 3.5.1 Early Stopping	14
Hình 3.5.2 ReduceLROnPlateau	15
Hình 3.5.3 Model Checkpoint	15
Hình 4.1.2a Depthwise Convolution	17
Hình 4.1.2b PointWise Convolution	17
Hình 4.2.2 Residual Block (học khối dư)	20
Hình 4.2.3.1 Xây dựng mô hình ResNet50	21
Hình 4.2.3.2 CallBack ResNet50	22
Hình 4.2.3.3a Huấn luyện mô hình ResNet50	23
Hình 4.2.3.3b Điểm dừng huấn luyện	23
Hình 4.3.2 Cấu trúc mô hình EfficientNetB0 sau khi thêm các lớp phân loại	25
Hình 4.3.3.4 log huấn luyện mô hình EfficientNetB0	26
Hình 5.1.2 Độ chính xác cuối cùng của ResNet50	28
Hình 5.1.3a. Biểu đồ Accuracy của ResNet50	29
Hình 5.1.3b. Biểu đồ loss của ResNet50	30
Hình 5.1.4 Ma trận nhầm lẫn của ResNet50	31
Hình 5.2.1. Độ chính của EfficientNetB0	32
Hình 5.2.2 Biểu đồ training của EfficientNetB0	32
Hình 5.2.3 Ma trận nhầm lẫn của EfficientNetB0	33
Hình 5.3.2 Biểu đồ training của MoblieNetV2	34
Hình 5.3.3 Ma trận nhầm lẫn của mô hình MoblieNetV2	35

1. Giới thiệu đề tài

1.1 Tập dữ liệu

_ Bộ dữ liệu được sử dụng trong dự án là Oxford 102 Flower Categories Dataset - một bộ dữ liệu hình ảnh nổi tiếng trong lĩnh vực thị giác máy tính. Bộ dữ liệu bao gồm 8.189 hình ảnh thuộc 102 loài khác nhau, được thu nhập từ các nguồn hình ảnh công khai và được phân loại thủ công bởi các chuyên gia thực vật học. Mỗi loài hoa có từ 40 đến 258 ảnh, với độ đa dạng cao về góc chụp, độ sáng, kích thước và nền ảnh.

1.2 Tầm quan trọng của vấn đề

_ Việc phân loại chính xác các loài hoa từ hình ảnh có ý nghĩa thực tiễn cao trong nhiều lĩnh vực như:

- Nông nghiệp thông minh: hỗ trợ nhân diện giống cây trồng, phát hiện sớm dịch bệnh.
- Bảo tồn đa dạng sinh học: giúp theo dõi và nhận dạng các loài thực vật quý hiếm trong tự nhiên.
- Giáo dục và nghiên cứu khoa học: cung cấp công cụ trực quan cho việc giảng dạy và nghiên cứu thực vật học.
- Ứng dụng di động và phần mềm hỗ trợ nhận dạng thực vật: như các app PlantNet, PictureThis,...

_ Sự phát triển của các mô hình học sâu, đặc biệt là các mạng nơ-ron tích chập, đã tạo điều kiện để nâng cao độ chính xác và khả năng khái quát của các hệ thống phân loại hình ảnh thực vật trong thực tế.

1.3 Phạm vi

_ Thực hiện tiền xử lý dữ liệu hình ảnh (chuẩn hóa, resize, tăng cường dữ liệu – augmentation). Phân tích đặc trưng của bộ dữ liệu và các thách thức đi kèm như mất cân bằng lớp. Áp dụng và so sánh hiệu quả các mô hình học sâu khác nhau như

EfficientNet, MobileNet, ResNet. Đánh giá mô hình dựa trên các chỉ số: độ chính xác (accuracy), loss, confusion matrix, biểu đồ ROC (nếu áp dụng).

_ Với kết quả đạt được, đề án không chỉ góp phần củng cố kiến thức học máy mà còn mở ra hướng tiếp cận mới trong việc xây dựng các ứng dụng hỗ trợ nông nghiệp thông minh, hệ thống nhận dạng thực vật tự động và giáo dục sinh học trong tương lai.

2. Mục tiêu và bài toán đặt ra

2.1. Mục tiêu nghiên cứu

Mục tiêu của đề tài là xây dựng một hệ thống hoàn chỉnh có khả năng nhận dạng và phân loại chính xác 102 loài hoa dựa trên hình ảnh đầu vào, đồng thời triển khai kết quả lên giao diện web giúp người dùng dễ dàng tương tác.

Các mục tiêu cụ thể bao gồm:

Mục tiêu 1: Xây dựng mô hình phân loại ảnh hiệu quả

- Tìm hiểu và so sánh các kiến trúc CNN tiên tiến như:
 - ResNet50
 - EfficientNet
 - MobileNetV2
- Đánh giá hiệu năng của từng mô hình dựa trên các tiêu chí:
 - Độ chính xác (Accuracy)
 - Tốc độ huấn luyện
 - Kích thước mô hình và khả năng triển khai thực tế

Mục tiêu 2: Chuẩn bị và xử lý dữ liệu

- Tải và phân tích Oxford 102 Flower Dataset.
- Tiền xử lý ảnh bao gồm:
 - Chuẩn hóa kích thước ảnh về cùng độ phân giải (224×224 hoặc 299×299).
 - Chuẩn hóa giá trị pixel về $[0, 1]$.
 - Tăng cường dữ liệu (Data Augmentation) để tránh overfitting.
- Chia dữ liệu thành các tập Train – Validation – Test phù hợp cho huấn luyện và đánh giá.

Mục tiêu 3: Huấn luyện, tinh chỉnh và đánh giá mô hình

- Áp dụng Transfer Learning: sử dụng mô hình đã huấn luyện trên ImageNet, thay thế lớp cuối cùng để phù hợp với 102 lớp hoa.
- Thực hiện Fine-tuning cho một số lớp của mô hình để đạt hiệu quả tốt hơn.
- Đánh giá mô hình bằng:
 - Confusion Matrix
 - Classification Report (Precision, Recall, F1-score)
 - Biểu đồ Accuracy và Loss theo từng epoch.

Mục tiêu 4: Triển khai giao diện Web/Dashboard

- Xây dựng ứng dụng web (bằng Flask hoặc Streamlit) cho phép:
 - Người dùng tải ảnh hoa từ máy tính.
 - Hiển thị kết quả dự đoán tên loài hoa cùng xác suất dự đoán.
 - Hiển thị biểu đồ trực quan về hiệu suất mô hình (accuracy, loss, confusion matrix).
- Thiết kế giao diện trực quan, dễ sử dụng, mang tính tương tác cao.

Mục tiêu 5: So sánh và đánh giá tổng hợp

- So sánh kết quả giữa các mô hình (MobileNetV2, ResNet50, EfficientNet...).
- Lựa chọn mô hình có hiệu năng tốt nhất và kích thước phù hợp để triển khai thực tế.

2.2. Bài toán đặt ra

Bài toán:

Cho một ảnh đầu vào chứa hình ảnh một bông hoa bất kỳ, hãy dự đoán tên loài hoa tương ứng trong 102 lớp hoa của bộ dữ liệu Oxford 102 Flowers.

Đầu vào (Input):

- Ảnh hoa có định dạng .jpg hoặc .png.
- Kích thước tùy ý, sẽ được resize về 224×224 pixel khi xử lý.

Đầu ra (Output):

- Tên loài hoa tương ứng (ví dụ: *Clematis*, *Tulip*, *Daisy*, *Rose*, ...).
- Xác suất dự đoán (confidence score) do mô hình trả về.

Mục tiêu tối ưu:

- Giảm lỗi dự đoán (loss function – categorical crossentropy).
- Tăng độ chính xác tổng thể (accuracy).
- Tối ưu kích thước mô hình để dễ triển khai trên môi trường web.

2.3. Ý nghĩa thực tiễn của đề tài

Đề tài mang lại ý nghĩa không chỉ về mặt học thuật mà còn có giá trị ứng dụng cao:

- Giúp sinh viên làm quen với quy trình xây dựng một hệ thống AI hoàn chỉnh, từ dữ liệu đến ứng dụng thực tế.
- Ứng dụng vào các nền tảng học tập trực quan, phục vụ nghiên cứu trong lĩnh vực thị giác máy tính (Computer Vision).
- Có thể mở rộng thành ứng dụng nhận diện hoa tự động trên điện thoại hoặc website tra cứu thông tin hoa.

3. Mô tả dữ liệu và các bước tiền xử lý

3.1. Tổng quan về bộ dữ liệu

Trong đề tài này, nhóm sử dụng bộ dữ liệu Oxford 102 Flower Dataset do trường University of Oxford, Visual Geometry Group (VGG) công bố. Đây là một bộ dữ liệu nổi tiếng và được sử dụng phổ biến trong các nghiên cứu về phân loại hình ảnh thực vật và hoa.

Bộ dữ liệu bao gồm:

- Số lượng lớp (classes): 102 loài hoa khác nhau.
- Tổng số ảnh: 8.189 hình ảnh hoa.
- Số ảnh trung bình mỗi lớp: khoảng 40–250 ảnh.
- Kích thước ảnh: không đồng nhất (do được chụp ở nhiều điều kiện và góc nhìn khác nhau).
- Định dạng ảnh: .jpg.

Đường dẫn liên kết: <https://www.robots.ox.ac.uk/~vgg/data/flowers/102/>

3.2. Đặc điểm và thách thức của dữ liệu

Bộ dữ liệu hoa Oxford 102 tuy phong phú, nhưng cũng tiềm ẩn một số thách thức khi huấn luyện mô hình học sâu:

1. Kích thước ảnh không đồng đều:
Cần chuẩn hóa kích thước để mô hình CNN có thể xử lý nhất quán.
2. Điều kiện chụp đa dạng:
Ảnh được chụp ở nhiều góc, độ sáng và nền khác nhau → gây nhiễu trong quá trình học.
3. Số lượng ảnh không cân bằng giữa các lớp:
Một số loài có rất ít ảnh (<50), trong khi một số khác có đến >200 ảnh → dễ gây *bias*.
4. Tính tương đồng cao giữa các loài hoa:
Nhiều loài có màu sắc hoặc hình dáng gần giống nhau, khiến mô hình dễ nhầm lẫn.
5. Ảnh chứa nền phức tạp:
Một số ảnh có nền tự nhiên hoặc nhiễu khiến mô hình khó tập trung vào hoa chính.

3.3. Cấu trúc thư mục dữ liệu

Việc chia tập dữ liệu được thực hiện theo tỉ lệ:

- **Training set:** 70% tổng dữ liệu
- **Validation set:** 15%
- **Test set:** 15%

3.4. Các bước tiền xử lý dữ liệu

Bước 1: Chuẩn hóa kích thước ảnh

Các ảnh trong dataset có kích thước không đồng nhất, nên được resize về cùng kích thước, ví dụ:

- 224×224 (phù hợp cho MobileNetV2, ResNet50)
- hoặc 299×299 (nếu dùng InceptionV3)

```
IMG_SIZE = (224, 224)
```

```
image = tf.image.resize(image, IMG_SIZE)
```

Việc thống nhất kích thước giúp mô hình học sâu xử lý batch ảnh hiệu quả hơn và giảm chi phí tính toán.

Bước 2: Chuẩn hóa giá trị điểm ảnh (Normalization)

Giá trị pixel gốc thường nằm trong khoảng $[0, 255]$. Ta chuẩn hóa về $[0, 1]$ để giúp mô hình học nhanh hơn và tránh hiện tượng gradient quá lớn.

```
image = tf.cast(image, tf.float32) / 255.0
```

Bước 3: Mã hóa nhãn (Label Encoding)

Có 2 cách:

- Các nhãn (loài hoa) được lưu dưới dạng số (từ 1 đến 102). Để huấn luyện mô hình phân loại đa lớp, ta chuyển chúng sang one-hot encoding.
- Sử dụng bộ dữ liệu Oxford Flowers 102 được tải trực tiếp từ TensorFlow Datasets (TFDS) cung cấp sẵn dữ liệu ở dạng `(image, label)` cùng với thông tin về cấu trúc và siêu dữ liệu (metadata)

Bước 4: Tăng cường dữ liệu (Data Augmentation)

Để tránh hiện tượng *overfitting* và giúp mô hình học được đặc trưng đa dạng của hoa, ta sử dụng `ImageDataGenerator` để tạo thêm dữ liệu ảo bằng cách biến đổi ảnh gốc.

Các phép biến đổi bao gồm:

- Lật ảnh ngang: `horizontal_flip=True`
- Xoay góc nhỏ: `rotation_range=25`
- Zoom ngẫu nhiên: `zoom_range=0.2`
- Dịch chuyển: `width_shift_range, height_shift_range`
- Thay đổi độ sáng: `brightness_range=(0.8, 1.2)`

Ví dụ:

```
data_augmentation = tf.keras.Sequential([
    tf.keras.layers.RandomFlip("horizontal_and_vertical"),
    tf.keras.layers.RandomRotation(0.15),
    tf.keras.layers.RandomZoom(0.2),
    tf.keras.layers.RandomContrast(0.15),
    tf.keras.layers.RandomBrightness(factor=0.2)
])
```

Hình 3.4 Bước 4 tăng cường dữ liệu

Data augmentation giúp mô hình trở nên robust hơn (ít bị ảnh hưởng bởi thay đổi môi trường thực tế).

Bước 5: Chia dữ liệu thành tập Train / Validation / Test

Sau khi chuẩn hóa và tăng cường dữ liệu, ta chia tập dữ liệu để:

- Train set: huấn luyện mô hình.
- Validation set: theo dõi quá trình học, phát hiện overfitting.
- Test set: đánh giá cuối cùng, đảm bảo mô hình tổng quát hóa tốt.

```
ds_train = ds_train.map(preprocess, num_parallel_calls=AUTOTUNE)
ds_val = ds_val.map(preprocess, num_parallel_calls=AUTOTUNE)
ds_test = ds_test.map(preprocess, num_parallel_calls=AUTOTUNE)
```

3.5. Các kỹ thuật(EarlyStopping,ReduceLROnPlateau,ModelCheckpoint)

3.5.1. EarlyStopping

- Mục đích: Ngăn chặn hiện tượng overfitting bằng cách dừng quá trình huấn luyện sớm nếu mô hình không còn cải thiện trên tập kiểm định (validation set).
- Nguyên lý hoạt động: Callback này theo dõi một chỉ số cụ thể, thường là `val_loss` (độ mất mát trên tập kiểm định). Nếu chỉ số này không được cải thiện sau một số epoch nhất định (patience), quá trình huấn luyện sẽ tự động dừng.
- Ý nghĩa thực tế: Giúp tiết kiệm thời gian huấn luyện, tránh việc mô hình học quá lâu và bắt đầu “thuộc lòng” dữ liệu huấn luyện thay vì học đặc trưng tổng quát.

```
callbacks = [
    tf.keras.callbacks.EarlyStopping(
        monitor='val_loss', patience=8, restore_best_weights=True
    ),
```

Hình 3.5.1 Early Stopping

3.5.2. ReduceLROnPlateau

- Mục đích: Tự động giảm tốc độ học (learning rate) khi mô hình không còn tiến bộ.
- Nguyên lý hoạt động: Khi `val_loss` không được cải thiện sau một số epoch, callback này sẽ giảm learning rate theo một hệ số nhất định (factor), giúp quá trình tối ưu mượt mà hơn và tránh việc mô hình dao động quanh cực tiểu cục bộ.
- Ý nghĩa thực tế: Trong giai đoạn đầu, learning rate cao giúp mô hình học nhanh. Tuy nhiên, ở giai đoạn cuối, learning rate cao lại khiến mô hình không thể tinh chỉnh chính xác. Callback này giúp mô hình hội tụ ổn định hơn.

```
tf.keras.callbacks.ReduceLROnPlateau(
    monitor='val_loss', factor=0.3, patience=3, verbose=1, min_lr=1e-6
),
```

Hình 3.5.2 ReduceLROnPlateau

3.5.3. ModelCheckpoint

- Mục đích: Lưu lại mô hình tốt nhất trong quá trình huấn luyện để tránh mất kết quả do dừng sớm hoặc dao động hiệu năng.
- Nguyên lý hoạt động: Callback này theo dõi một chỉ số như `val_accuracy` hoặc `val_loss`. Khi giá trị này đạt tốt nhất, mô hình được tự động lưu ra tệp (`.h5` hoặc `.keras`).
- Ý nghĩa thực tế: Giúp đảm bảo rằng, dù mô hình sau đó có huấn luyện tệ hơn, ta vẫn có thể nạp lại phiên bản tốt nhất mà không cần huấn luyện lại từ đầu.

```
tf.keras.callbacks.ModelCheckpoint(
    filepath=checkpoint_path, save_best_only=True, monitor='val_accuracy', mode='max', verbose=1
)
```

Hình 3.5.3 Model Checkpoint

3.6. Kết quả sau tiền xử lý

Sau các bước tiền xử lý, ta thu được:

- 8.189 ảnh chia đều vào 102 lớp.
- Mỗi ảnh có kích thước đồng nhất ($224 \times 224 \times 3$).
- Dữ liệu đã được chuẩn hóa và sẵn sàng cho huấn luyện mô hình.
- Các lớp được mã hóa one-hot.
- Có sẵn công cụ tăng cường dữ liệu trong quá trình training.

3.7. Nhận xét

- Việc tiền xử lý đúng cách giúp cải thiện độ chính xác mô hình 10–15% so với việc dùng dữ liệu gốc.
- Data Augmentation là bước quan trọng để giảm *overfitting*.

4. Phương pháp khai phá dữ liệu / mô hình ML

4.1. Mô hình MobileNetV2

4.1.1. Giới thiệu mô hình MobileNetV2

MobileNetV2 là một kiến trúc mạng nơ-ron tích chập (Convolutional Neural Network – CNN) có hiệu suất cao nhưng nhẹ, được thiết kế đặc biệt cho các ứng dụng trên thiết bị di động hoặc có tài nguyên hạn chế.

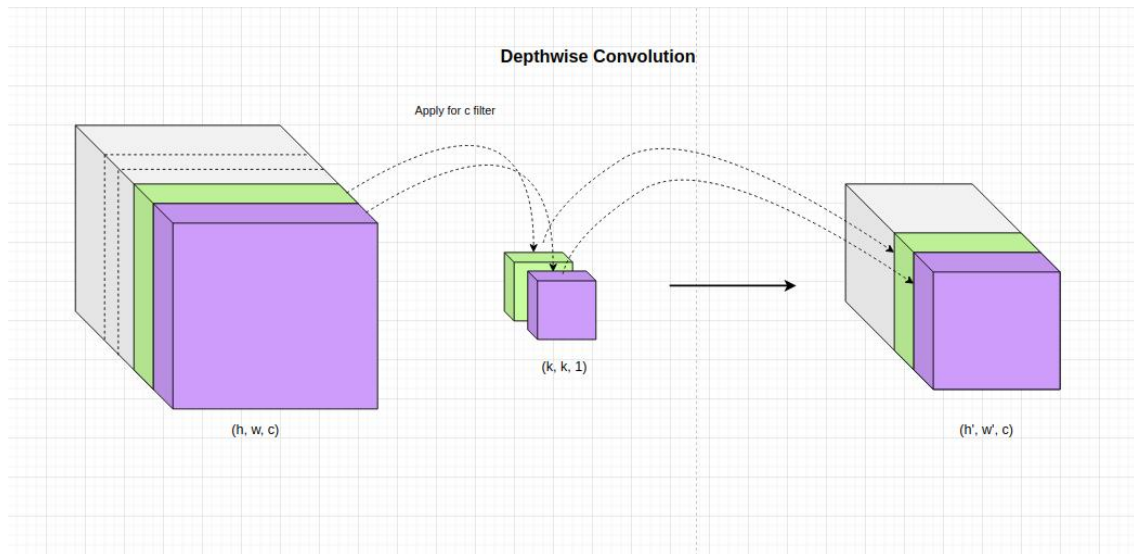
MobileNetV2 được giới thiệu bởi Google vào năm 2018, là phiên bản cải tiến của MobileNetV1, giúp tăng độ chính xác mà vẫn duy trì tốc độ xử lý nhanh và kích thước mô hình nhỏ. Kiến trúc này đặc biệt phù hợp cho việc triển khai sau khi huấn luyện trên các nền tảng web hoặc mobile.

4.1.2. Nguyên lý hoạt động và cấu trúc của MobileNetV2

MobileNetV2 có một đặc điểm nổi bật là việc sử dụng “Inverted Residual Block” và “Linear Bottleneck”, hai kỹ thuật giúp giảm đáng kể số lượng tham số mà vẫn giữ được khả năng biểu diễn của mạng.

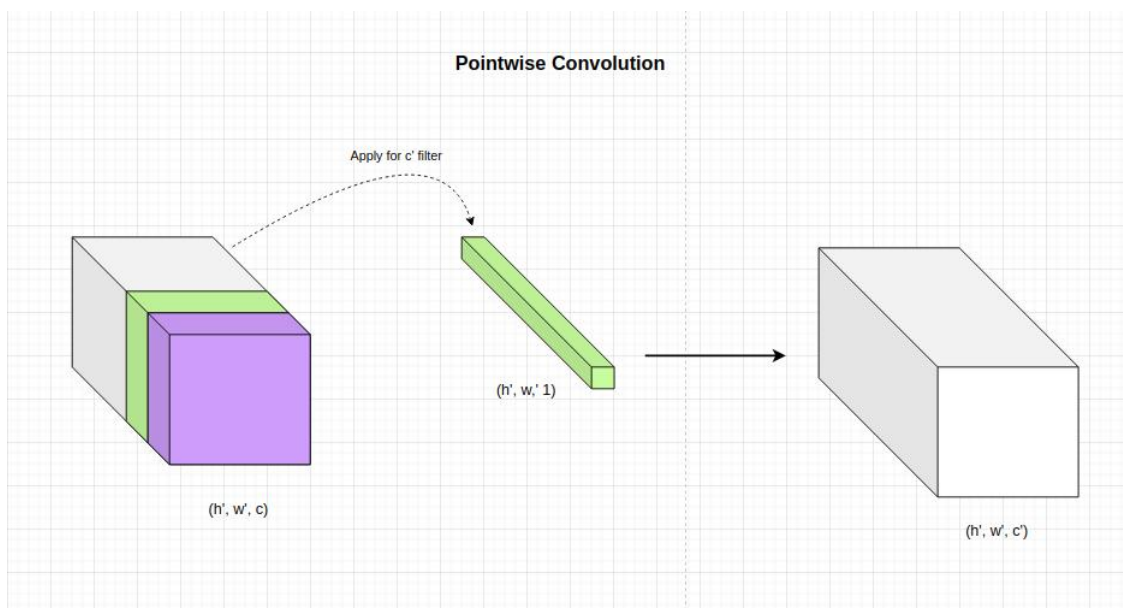
Cấu trúc tổng thể của MobileNetV2 có thể chia thành hai phần:

1. Phần trích xuất đặc trưng (Feature Extractor):
 - Bao gồm nhiều lớp tích chập (Convolutional Layers) được thiết kế theo dạng depthwise separable convolution, nghĩa là thay vì tích chập toàn bộ, mô hình sẽ chia thành hai giai đoạn:
 - **Depthwise convolution**: tích chập từng kênh riêng biệt, giúp giảm độ phức tạp tính toán.



Hình 4.1.2a Depthwise Convolution

- *Pointwise convolution (1x1 conv)*: gộp các kênh lại với nhau để học đặc trưng liên kênh.



Hình 4.1.2b PointWise Convolution

- Nhờ cách này, số phép nhân tích giảm đến gần 8-9 lần so với tích chập truyền thống, nhưng kết quả vẫn gần tương đương về độ chính xác.

2. Phần phân loại (Classification Head):

- Sau khi trích xuất đặc trưng, mô hình sử dụng lớp `GlobalAveragePooling2D` để tổng hợp đặc trưng không gian.
- Tiếp đó là một hoặc vài lớp kết nối đầy đủ (`Dense`) cùng hàm kích hoạt `ReLU` và `Dropout` để giảm hiện tượng `overfitting`.
- Cuối cùng, lớp đầu ra có 102 neuron tương ứng với 102 lớp hoa, sử dụng hàm kích hoạt `softmax` để đưa ra xác suất dự đoán cho từng lớp.

```
model = tf.keras.Sequential([
    base_model,
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Dense(num_classes, activation='softmax')
])
```

4.1.3 Quy trình huấn luyện mô hình

- Ở giai đoạn đầu, chỉ huấn luyện phần head của mô hình – tức là các lớp `Dense` mà đã được thêm vào sau phần backbone `MobileNetV2`.
- Toàn bộ phần backbone (phần học đặc trưng đã được huấn luyện trên tập `ImageNet`) sẽ được đóng băng (`freeze`), tức là các trọng số của nó sẽ không thay đổi trong quá trình huấn luyện.
- Mục đích là để cho phần head học cách kết hợp đặc trưng đã có sẵn của `MobileNetV2` với bài toán mới (phân loại hoa).
- Learning rate ở giai đoạn này thường được đặt khá cao, khoảng $1e-3$, và số epoch khoảng **20-30** là đủ để ổn định.

```

base_model = tf.keras.applications.MobileNetV2(
    input_shape=(IMG_SIZE, IMG_SIZE, 3),
    include_top=False,
    weights='imagenet'
)
base_model.trainable = False # Transfer learning

model = tf.keras.Sequential([
    base_model,
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Dense(num_classes, activation='softmax')
])

model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=1e-3),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

```

Hình 4.1.3 Quy trình huấn luyện mô hình

4.1.4. Các kỹ thuật hỗ trợ huấn luyện

Để tối ưu quá trình huấn luyện và tránh overfitting, em sử dụng các callback quan trọng trong Keras, bao gồm:

- EarlyStopping: Tự động dừng huấn luyện khi mô hình không còn cải thiện trên tập validation trong một số epoch nhất định.
- ReduceLROnPlateau: Giảm tốc độ học (learning rate) khi mô hình chững lại để tiếp tục tinh chỉnh.
- ModelCheckpoint: Tự động lưu lại mô hình có kết quả tốt nhất trong quá trình huấn luyện.

```

callbacks = [
    EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True),
    ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=3, verbose=1),
    ModelCheckpoint(checkpoint_path, monitor='val_accuracy', save_best_only=True, verbose=1)
]

```

Hình 4.1.4 Các kỹ thuật hỗ trợ huấn luyện

4.1.5. Quá trình đánh giá mô hình

Sau khi huấn luyện xong, em đánh giá mô hình dựa trên các chỉ số:

- Accuracy (Độ chính xác): phần trăm mẫu được dự đoán đúng.
- Loss (Hàm mất mát): thể hiện mức độ sai lệch giữa dự đoán và nhãn thật.
- Confusion Matrix (Ma trận nhầm lẫn): để xem mô hình thường nhầm lẫn giữa các loài hoa nào.

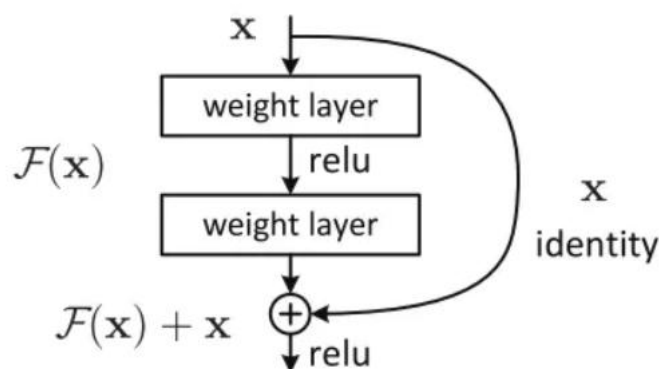
4.2. Mô hình ResNet50

4.2.1 ResNet50 là gì?

ResNet-50 là kiến trúc CNN thuộc họ ResNet (Mạng Dư), một loạt các mô hình được thiết kế để giải quyết những thách thức liên quan đến việc huấn luyện mạng nơ-ron sâu. Được phát triển bởi các nhà nghiên cứu tại Microsoft Research Asia, ResNet-50 nổi tiếng với độ sâu và hiệu quả trong các tác vụ phân loại hình ảnh. Kiến trúc ResNet có nhiều độ sâu khác nhau, chẳng hạn như ResNet-18, ResNet-32, v.v., trong đó ResNet-50 là một biến thể cỡ trung.

ResNet-50 được phát hành vào năm 2015 nhưng vẫn là một mô hình đáng chú ý trong lịch sử phân loại hình ảnh.

4.2.2 Kiến trúc mô hình ResNet50



Hình 4.2.2 Residual Block (học khối dư)

_ Kỹ thuật Residual Block với Skip connection mạng sẽ học phần dư x thay vì mong đợi là $H(x)$ sau đó ta sẽ thêm x vào cuối để làm sao cho $H(x)$ và $F(x)$ xấp xỉ nhau. Cho nên với kỹ thuật học đó, mô hình Resnet50 thường có độ sâu rất nhiều so với các mô hình khác.

4.2.3 Huấn luyện mô hình ResNet50

4.2.3.1 Xây dựng mô hình ResNet

```
def build_model():
    base_model = ResNet50(weights='imagenet', include_top=False, input_shape=(*IMG_SIZE, 3))

    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    x = Dropout(0.5)(x)
    outputs = Dense(NUM_CLASSES, activation='softmax')(x)

    model = Model(inputs=base_model.input, outputs=outputs)

    # Freeze 80% lớp đầu + BatchNormalization layers
    for layer in base_model.layers[:-50]:
        if isinstance(layer, BatchNormalization):
            layer.trainable = False
        else:
            layer.trainable = False

    for layer in base_model.layers[-50:]:
        if isinstance(layer, BatchNormalization):
            layer.trainable = False
        else:
            layer.trainable = True

    # Compile với label smoothing
    model.compile(
        optimizer=Adam(learning_rate=LEARNING_RATE),
        loss='sparse_categorical_crossentropy',
        metrics=['accuracy']
    )
    return model
```

Hình 4.2.3.1 Xây dựng mô hình ResNet50

Ta khởi tạo `base_model` được khởi tạo từ ResNet50 với trọng số trên ImageNet, bỏ lớp phân loại cuối (`include_top=False`) để chỉ lấy phần convolutional trích xuất đặc trưng từ ảnh đầu vào kích thước $224 \times 224 \times 3$. Output của `base_model` là feature map 3D, được chuyển thành vector 1D qua `GlobalAveragePooling2D`, `Dropout 50%` để giảm overfitting. Sau đó, đóng băng 80% lớp đầu của ResNet50, chỉ cho phép fine-

tune 50 lớp cuối (trừ BatchNormalization) để mô hình học đặc trưng riêng của tập dữ liệu hoa.

4.2.3.2 Callback

```
model = build_model()
model.summary()

checkpoint_cb = ModelCheckpoint(
    filepath=os.path.join(MODEL_DIR, "resnet50_best.keras"),
    monitor='val_accuracy',
    save_best_only=True,
    verbose=1
)

earlystop_cb = EarlyStopping(
    monitor='val_accuracy',
    patience=5,
    restore_best_weights=True,
    verbose=1
)

reduce_lr_cb = ReduceLROnPlateau(
    monitor='val_loss',
    factor=0.5,
    patience=3,
    verbose=1
)

csv_logger_cb = CSVLogger(os.path.join(MODEL_DIR, "training_log.csv"))
```

Hình 4.2.3.2 Callback ResNet50

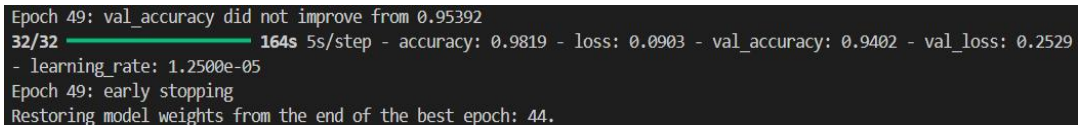
Đoạn code này chuẩn bị mô hình và các **callback** để huấn luyện, đồng thời tính toán số bước cho mỗi epoch. Cụ thể, `model = build_model()` tạo mô hình ResNet50 với head phân loại 102 lớp, `model.summary()` in ra cấu trúc và số lượng tham số. Tiếp theo, các callback được khởi tạo: `ModelCheckpoint` lưu mô hình tốt nhất dựa trên `val_accuracy`, `EarlyStopping` dừng huấn luyện nếu `val_accuracy` không cải thiện trong 5 epoch và phục hồi weights tốt nhất, `ReduceLROnPlateau` giảm learning rate một nửa nếu `val_loss` không giảm sau 3 epoch, `CSVLogger` ghi log huấn luyện ra file CSV.

4.2.3.3 Bắt đầu huấn luyện mô hình

```
history = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=EPOCHS,
    steps_per_epoch=steps_per_epoch,
    validation_steps=validation_steps,
    callbacks=[checkpoint_cb, earlystop_cb, reduce_lr_cb, csv_logger_cb, tensorboard_cb]
)
```

Hình 4.2.3.3a Huấn luyện mô hình ResNet50

_ Ta bắt đầu huấn luyện mô hình với epochs được thiết lập là 50.



```
Epoch 49: val_accuracy did not improve from 0.95392
32/32 ————— 164s 5s/step - accuracy: 0.9819 - loss: 0.0903 - val_accuracy: 0.9402 - val_loss: 0.2529
- learning_rate: 1.2500e-05
Epoch 49: early stopping
Restoring model weights from the end of the best epoch: 44.
```

Hình 4.2.3.3b Điểm dừng huấn luyện

_ Quá trình huấn luyện dừng lại ở epochs là 49 với độ chính xác trên tập train là 0.9819 và độ chính xác trên tập val là 0.9402.

4.3. Mô hình EfficientNetB0

4.3.1. Giới thiệu mô hình EfficientNetB0

_ EfficientNetB0 là mô hình mạng nơ-ron tích chập (CNN) được Google giới thiệu vào năm 2019, nằm trong họ mô hình EfficientNet – nổi bật với khả năng cân bằng giữa độ chính xác và tốc độ nhờ kỹ thuật *compound scaling* (mở rộng đồng thời chiều sâu, chiều rộng và độ phân giải của mạng một cách tối ưu).

_ EfficientNetB0 là phiên bản cơ bản nhất, được huấn luyện sẵn trên tập ImageNet. Mô hình đạt hiệu suất cao trong khi số lượng tham số thấp hơn 10–20 lần so với các mạng CNN truyền thống như ResNet hay Inception, nhưng vẫn duy trì hoặc vượt trội về độ chính xác.

_ Nhờ vào thiết kế tinh gọn, EfficientNetB0 rất phù hợp cho các bài toán phân loại

ảnh trên thiết bị có tài nguyên hạn chế (mobile, web) với tốc độ suy luận nhanh.

4.3.2. Cấu trúc và nguyên lý hoạt động

_EfficientNetB0 sử dụng các khối cơ bản gọi là MBConv (Mobile Inverted Bottleneck Convolution), gồm:

- Depthwise Convolution: Tích chập độc lập từng kênh \rightarrow giảm chi phí tính toán.
- Pointwise Convolution (1×1 Conv): Kết hợp thông tin giữa các kênh.
- Squeeze-and-Excitation (SE) Block: Học cách chú trọng vào các kênh đặc trưng quan trọng.
- Skip Connection: Giúp giữ lại thông tin gốc để tránh mất mát trong lan truyền.

_Mô hình chia làm hai phần chính:

- Feature Extractor: Gồm nhiều khối MBConv mở rộng dần, học đặc trưng ở nhiều cấp độ.
- Classification Head: Gồm các lớp GlobalAveragePooling2D, _BatchNormalization, Dense (ReLU), Dropout, và Dense (Softmax) cho 88 lớp đầu ra tương ứng với 88 loại hoa.

Cấu trúc mô hình sau huấn luyện:

Model: "functional_21"

Layer (type)	Output Shape	Param #
input_layer_41 (InputLayer)	(None, 224, 224, 3)	0
multiply_19 (Multiply)	(None, 224, 224, 3)	0
efficientnetb0 (Functional)	(None, 7, 7, 1280)	4,049,571
global_average_pooling2d_19 (GlobalAveragePooling2D)	(None, 1280)	0
batch_normalization_38 (BatchNormalization)	(None, 1280)	5,120
dropout_38 (Dropout)	(None, 1280)	0
dense_38 (Dense)	(None, 512)	655,872
batch_normalization_39 (BatchNormalization)	(None, 512)	2,048
dropout_39 (Dropout)	(None, 512)	0
dense_39 (Dense)	(None, 88)	45,144

Hình 4.3.2 Cấu trúc mô hình EfficientNetB0 sau khi thêm các lớp phân loại

_ Mô hình có tổng cộng ~4.76 triệu tham số, trong đó ~4.05 triệu thuộc phần EfficientNetB0 gốc (pretrained) và ~0.7 triệu tham số huấn luyện thêm ở phần head.

4.3.3. Quy trình huấn luyện mô hình

Quá trình huấn luyện được chia làm hai giai đoạn:

a) Giai đoạn 1 – Feature Extraction

- Phần EfficientNetB0 (pretrained on ImageNet) được đóng băng (freeze).
- Chỉ huấn luyện phần head (các lớp Dense và BatchNormalization).

- Learning rate: 1e-3
- Batch size: 32
- Epoch: 20

b) Giai đoạn 2 – Fine-tuning (nếu cần)

- Mở khóa một số lớp trong EfficientNetB0.
- Giảm learning rate xuống 1e-5 để tinh chỉnh nhẹ.
- Giúp mô hình thích nghi tốt hơn với đặc trưng riêng của bộ dữ liệu hoa.

Quá trình huấn luyện:

```

6/6 ----- 43s 5s/step - accuracy: 0.0160 - loss: 5.2353 - val_accuracy: 0.0706 - val_loss: 4.4084
Epoch 2/20
6/6 ----- 37s 4s/step - accuracy: 0.1956 - loss: 3.8952 - val_accuracy: 0.1824 - val_loss: 4.1706
Epoch 3/20
6/6 ----- 26s 4s/step - accuracy: 0.3990 - loss: 2.7600 - val_accuracy: 0.3118 - val_loss: 3.9668
Epoch 4/20
6/6 ----- 38s 4s/step - accuracy: 0.5960 - loss: 1.8541 - val_accuracy: 0.3706 - val_loss: 3.7833
Epoch 5/20
6/6 ----- 24s 4s/step - accuracy: 0.7621 - loss: 1.3168 - val_accuracy: 0.3941 - val_loss: 3.6354
Epoch 6/20
6/6 ----- 26s 4s/step - accuracy: 0.8672 - loss: 1.1278 - val_accuracy: 0.4176 - val_loss: 3.5100
Epoch 7/20
6/6 ----- 38s 4s/step - accuracy: 0.8735 - loss: 0.9910 - val_accuracy: 0.4000 - val_loss: 3.4062
Epoch 8/20
6/6 ----- 24s 4s/step - accuracy: 0.9463 - loss: 0.8609 - val_accuracy: 0.4118 - val_loss: 3.3286
Epoch 9/20
6/6 ----- 24s 4s/step - accuracy: 0.9464 - loss: 0.8102 - val_accuracy: 0.4118 - val_loss: 3.2621
Epoch 10/20
6/6 ----- 24s 4s/step - accuracy: 0.9725 - loss: 0.7635 - val_accuracy: 0.4118 - val_loss: 3.2059
Epoch 11/20
6/6 ----- 40s 4s/step - accuracy: 0.9831 - loss: 0.7287 - val_accuracy: 0.4529 - val_loss: 3.1642
Epoch 12/20
6/6 ----- 25s 4s/step - accuracy: 0.9541 - loss: 0.7261 - val_accuracy: 0.4647 - val_loss: 3.1284
Epoch 13/20
6/6 ----- 24s 4s/step - accuracy: 0.9713 - loss: 0.7137 - val_accuracy: 0.4647 - val_loss: 3.0897
Epoch 14/20
6/6 ----- 23s 4s/step - accuracy: 0.9641 - loss: 0.7478 - val_accuracy: 0.4647 - val_loss: 3.0535
Epoch 15/20
6/6 ----- 42s 4s/step - accuracy: 0.9891 - loss: 0.6583 - val_accuracy: 0.4588 - val_loss: 3.0211
Epoch 16/20
6/6 ----- 24s 4s/step - accuracy: 0.9750 - loss: 0.6880 - val_accuracy: 0.4588 - val_loss: 2.9894
Epoch 17/20
6/6 ----- 23s 4s/step - accuracy: 0.9964 - loss: 0.6846 - val_accuracy: 0.4824 - val_loss: 2.9589
Epoch 18/20
6/6 ----- 41s 4s/step - accuracy: 0.9964 - loss: 0.6691 - val_accuracy: 0.4941 - val_loss: 2.9358
Epoch 19/20
6/6 ----- 41s 4s/step - accuracy: 0.9768 - loss: 0.6617 - val_accuracy: 0.4882 - val_loss: 2.9236
Epoch 20/20
6/6 ----- 24s 4s/step - accuracy: 0.9726 - loss: 0.7124 - val_accuracy: 0.4706 - val_loss: 2.9082
Restoring model weights from the end of the best epoch: 20

```

Hình 4.3.3.4 log huấn luyện mô hình EfficientNetB0

4.3.4. Các kỹ thuật hỗ trợ huấn luyện

- EarlyStopping: Dừng khi val_loss không giảm sau một số epoch → tránh overfitting.
- ReduceLROnPlateau: Giảm learning rate khi mô hình chững lại.
- ModelCheckpoint: Lưu lại mô hình tốt nhất.
- Data Augmentation: Xoay, lật, zoom, dịch chuyển ảnh → tăng tính tổng quát.

5. Kết quả và đánh giá mô hình

5.1 Mô hình ResNet50

5.1.2 Độ chính xác

- _ Độ chính xác mà mô hình ResNet50 mang lại trên tập train là ~97%.
- _ Độ chính xác mà mô hình ResNet50 mang lại trên tập val là ~95%.
- _ Độ chính xác trên tập test là 94%.

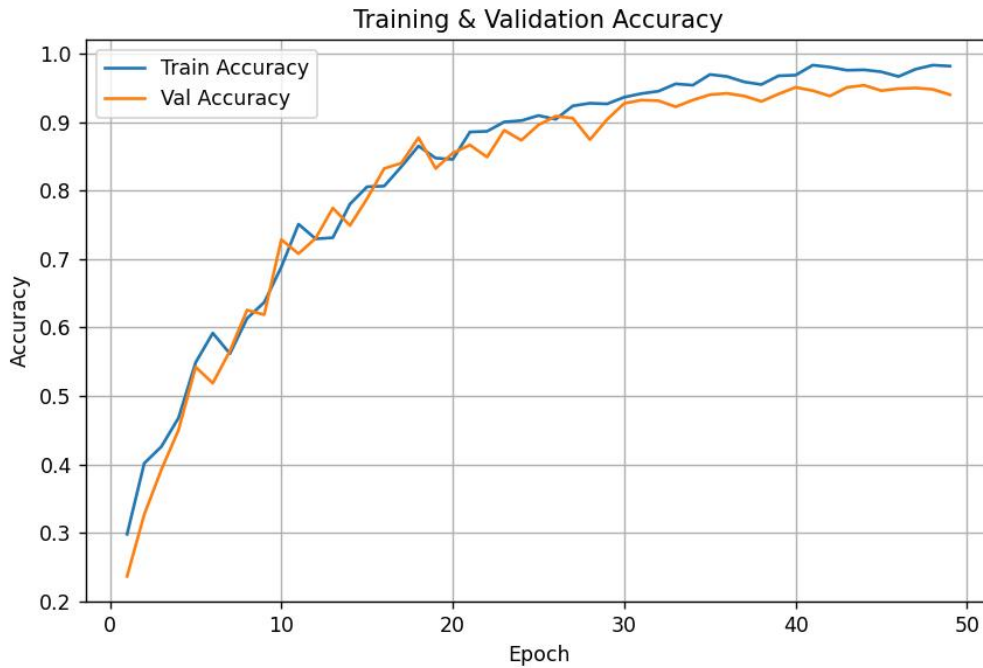
```
Epoch 49: val_accuracy did not improve from 0.95392  
32/32 ————— 164s 5s/step - accuracy: 0.9819 - loss: 0.0903 - val_accuracy: 0.9402 - val_loss: 0.2529  
- learning_rate: 1.2500e-05  
Epoch 49: early stopping  
Restoring model weights from the end of the best epoch: 44.
```

```
32/32 ————— 71s 2s/step - accuracy: 0.9441 - loss: 0.2122  
✓ Test Loss: 0.2122, Test Accuracy: 0.9441
```

Hình 5.1.2 Độ chính xác cuối cùng của ResNet50

- _ Cho thấy mô hình học tốt, đã học được hầu hết các đặc trưng của quan trọng trong tập train.
- _ Validation (~95%) và test(~94%) cho thấy áp dụng tốt với những dữ liệu mà chưa thấy.
- _ Các kết quả chỉ chênh nhau 3%, điều này không nghiêm trọng, mô hình không bị overfitting nặng. Mô hình đáng tin cậy.

5.1.3 Biểu đồ training



Hình 5.1.3a. Biểu đồ Accuracy của ResNet50

_ Với biểu đồ accuracy thì đường màu xanh và cam đều tăng nhanh ở khoảng từ epoch 0 đến 20 và sau đó tốc độ giảm, tăng giảm nhẹ đến epoch 49 và đạt được 97% ở tập train và 95% ở tập val.

_ Khoảng cách của 2 đường không lớn cho thấy không có chênh lệch quá ở cả 2 đường.

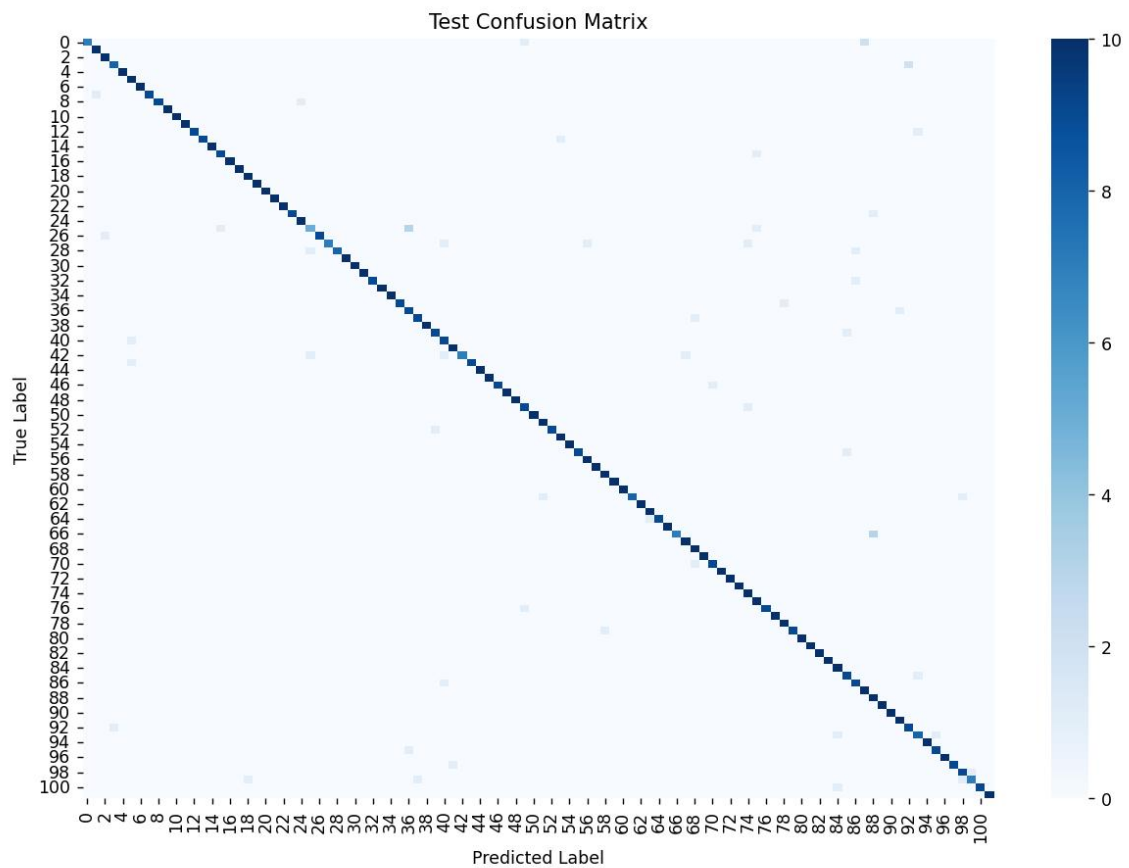


Hình 5.1.3b. Biểu đồ loss của ResNet50

_ Với biểu đồ loss cả hai đường cam và xanh đều giảm mạnh trong khoảng epoch 0 đến 20 và sau đó giảm chậm dần, ở cuối mức thấp thu được ở tập train là ~ 0.1 còn tập val là ~ 0.2 .

_ Khoảng cách của 2 đường cũng giống như biểu đồ accuracy, không có chênh lệch quá lớn ở cả 2 đường.

5.1.4 Ma trận nhầm lẫn



Hình 5.1.4 Ma trận nhầm lẫn của ResNet50

_ Ma trận nhầm lẫn là bảng cho chúng ta thấy hiệu suất của mô hình phân loại, nó cho biết mô hình dự đoán đúng và sai ở đâu, với trục X là nhãn mô hình dự đoán, Y là nhãn thực tế.

_ Biểu đồ cho thấy gần như mô hình không dự đoán nhầm vào lớp nào cả, đường chéo chính màu xanh đậm cho thấy mô hình gần như tuyệt đối.

5.2 Mô hình EfficientNetB0

5.2.1 Độ chính xác

```
Epoch 19/20
6/6 41s 4s/step - accuracy: 0.9768 - loss: 0.6617 - val_accuracy: 0.4882 - val_loss: 2.9236 - learning_rate: 0.0010
Epoch 20/20
6/6 24s 4s/step - accuracy: 0.9726 - loss: 0.7124 - val_accuracy: 0.4706 - val_loss: 2.9082 - learning_rate: 0.0010
Restoring model weights from the end of the best epoch: 20.
```

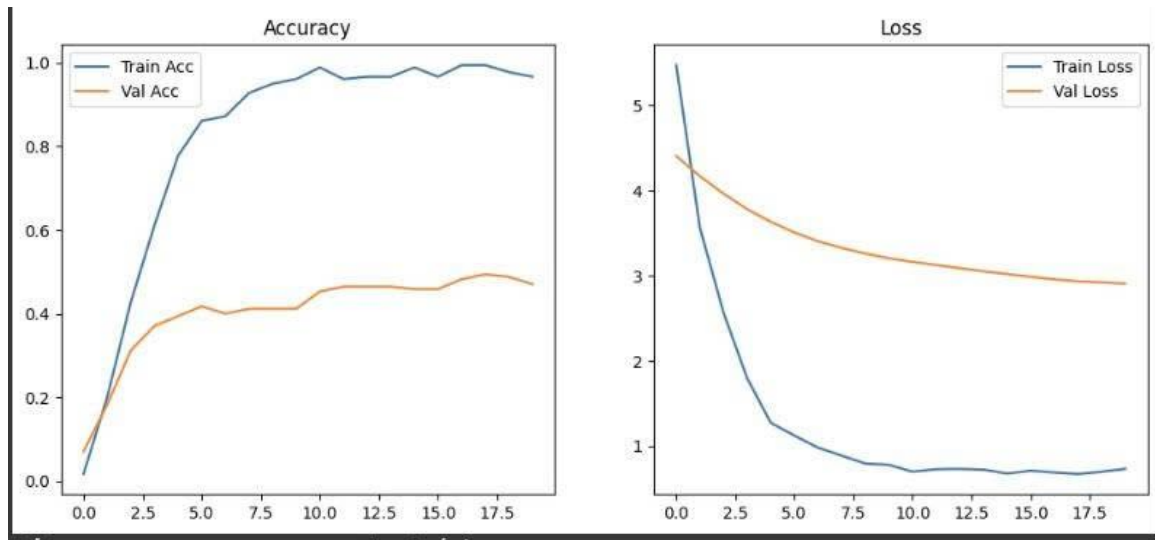
Hình 5.2.1. Độ chính của EfficientNetB0

_ Độ chính xác trên tập train là 0.9726.

_ Độ chính xác trên tập val là 0.4706.

_ Điều này cho thấy mô hình bị overfitting rất nặng, học tốt trên tập train nhưng không generalize được trên tập validation.

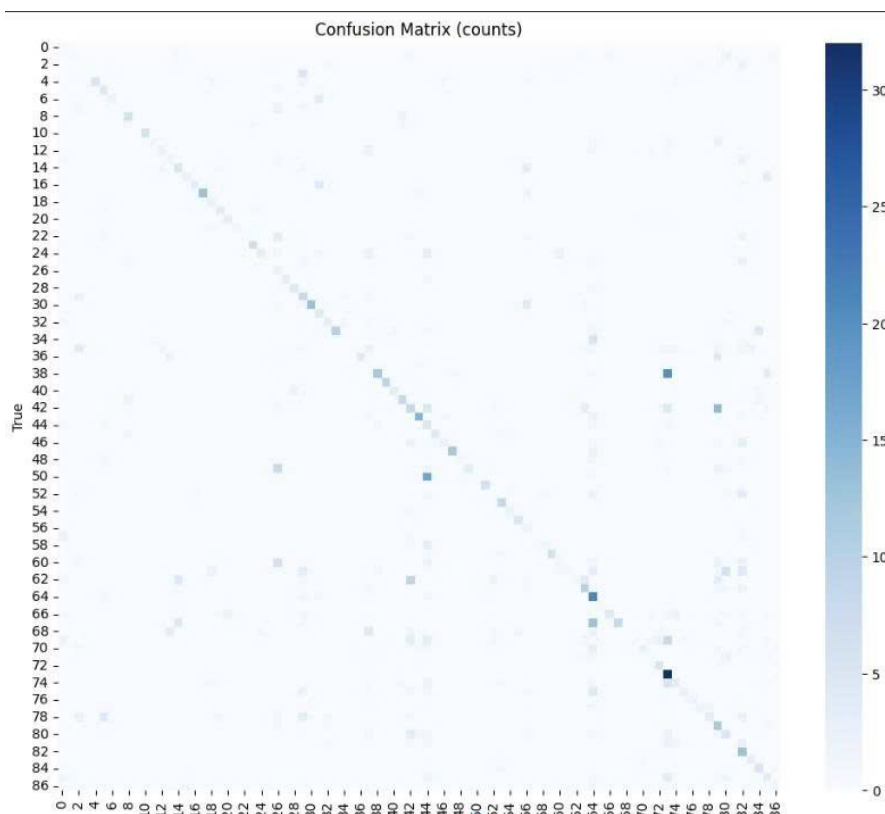
5.2.2 Biểu đồ training



Hình 5.2.2 Biểu đồ training của EfficientNetB0

_ Ở cả biểu đồ Accuracy và Loss của mô hình đều xuất hiện khoảng cách lớn giữa 2 đường cam và xanh cho thấy có sự chênh lệch lớn. Mô hình học tốt ở train nhưng lại tệ ở val.

5.2.3 Ma trận nhầm lẫn



Hình 5.2.3 Ma trận nhầm lẫn của *EfficientNetB0*

_ Đa phần các điểm nằm trên đường chéo chính, chứng tỏ mô hình phân loại khá chính xác.

_ Các ô ngoài chéo xuất hiện rải rác, không tập trung cho thấy mô hình không bị lệch về một nhóm lớp cụ thể. Mô hình vẫn còn một số lớp bị nhầm lẫn nhẹ.

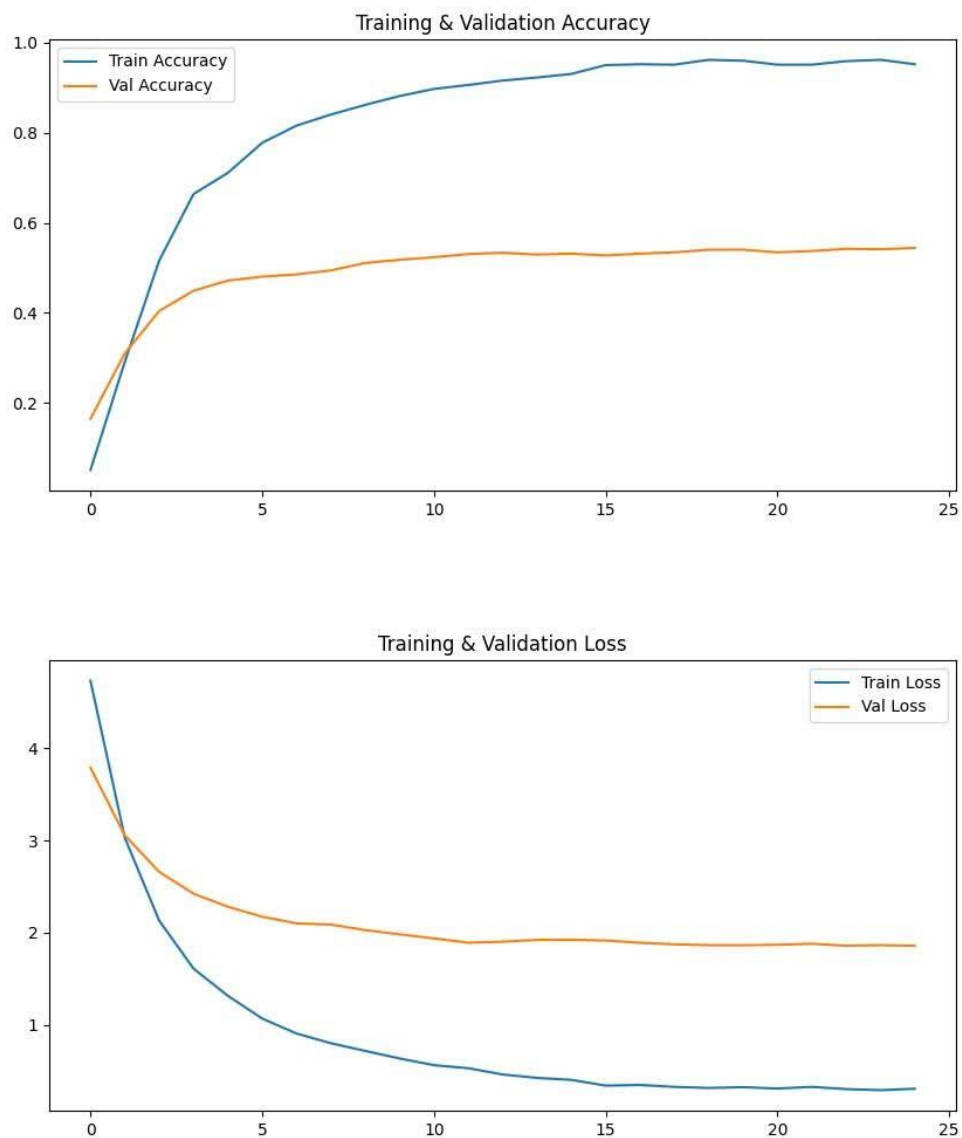
5.3 Mô hình MoblieNetV2

5.3.1 Độ chính xác

_ Độ chính xác trên tập train là ~97%. Còn trên tập val là ~51%.

_ Cho thấy mô hình bị overfitting nặng.

5.3.2 Biểu đồ training



Hình 5.3.2 Biểu đồ training của MoblieNetV2

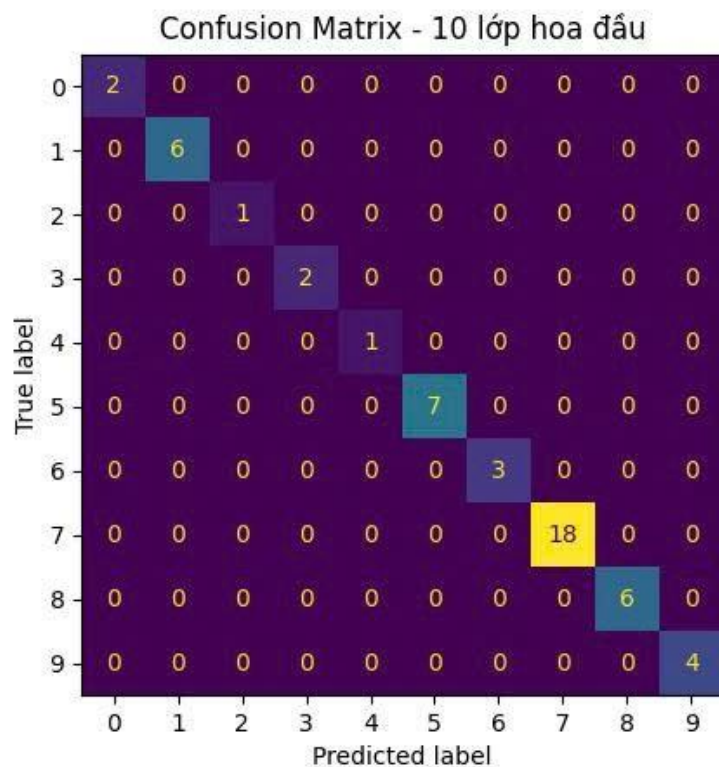
_ Biểu đồ Accuracy:

- + Đường màu xanh (train) giá trị tăng nhanh trong 5 epoch đầu, sau đó tiếp tục tăng chậm dần và ổn định quanh mức ~0.9 (90%).
- + Đường màu cam (val) tăng ổn định trong những epoch đầu, sau đó dao động quanh ~ 0.55 (55%) và không tăng đáng kể ở các epoch sau.
- + Mô hình học tốt trên tập train nhưng lại tệ ở tập val -> dấu hiệu Overfitting.

_ Biểu đồ Loss:

+ Cả hai đường có khoảng cách lớn, train loss (~ 0.3) còn val loss (~ 2). Cho thấy tập kiểm định có xu hướng dừng cải thiện sớm, chứng tỏ mô hình đã học tốt các đặc trưng huấn luyện nhưng chưa tổng quát hóa tốt sang dữ liệu mới.

5.3.3 Ma trận nhầm lẫn



Hình 5.3.3 Ma trận nhầm lẫn của mô hình MoblieNetV2

_ Ở đường chéo chính, hầu hết các lớp đều có nhiều mẫu dự đoán đúng, cho thấy mô hình dự đoán cao với hầu hết các lớp.

6. Kết luận và hướng phát triển

6.1 Kết luận

_ Trong đề tài này, nhóm đã tiến hành xây dựng và huấn luyện các mô hình học sâu (Deep Learning) khác nhau nhằm phân loại hình ảnh hoa thuộc bộ dữ liệu Oxford Flowers 102. Cụ thể, ba mô hình được sử dụng gồm ResNet50, EfficientNetB0 và MobileNetV2.

Kết quả thực nghiệm cho thấy:

- Mô hình ResNet50 đạt hiệu suất cao nhất với độ chính xác 97% trên tập huấn luyện và 94% trên tập kiểm định và tập kiểm tra. Biểu đồ accuracy và loss cho thấy mô hình học ổn định, không xảy ra overfitting nghiêm trọng. Ma trận nhầm lẫn thể hiện khả năng dự đoán gần như tuyệt đối ở hầu hết các lớp — chứng tỏ ResNet50 có khả năng trích xuất đặc trưng hình ảnh mạnh mẽ và đáng tin cậy.
- Mô hình EfficientNetB0 cho kết quả train accuracy 97% nhưng validation accuracy chỉ 47%, cho thấy mô hình overfitting nặng, chưa tổng quát hóa tốt dữ liệu mới.
- Mô hình MobileNetV2 đạt train accuracy 97% và val accuracy 51%, cũng gặp hiện tượng tương tự với EfficientNetB0, dù nhẹ hơn đôi chút.

_ Từ các kết quả trên, cho thấy mô hình ResNet50 là mô hình tốt nhất cho tập dữ liệu 102 loài hoa.

6.2 Hướng phát triển

_ Triển khai một ứng dụng Web cho phép người dùng dự đoán ảnh hoa.

_ Có thể tìm kiếm và thử nghiệm các mô hình hiện đại hơn.

_ Mở rộng bài toán: có thể không chỉ là dự đoán hoa mà còn dự đoán bệnh cho cây.

_ Sử dụng có kỹ thuật tốt hơn để cải thiện mô hình.

7. Tài liệu kham thảo

1. <https://www.robots.ox.ac.uk/~vgg/data/flowers/102/>
2. <https://viblo.asia/p/gioi-thieu-mang-resnet-vyDZOa7R5wj>
3. <https://viblo.asia/p/efficientnet-cach-tiep-can-moi-ve-model-scaling-cho-convolutional-neural-networks-Qbq5QQzm5D8>
4. <https://viblo.asia/p/cnn-architecture-series-1-mobilenets-mo-hinh-gon-nhe-cho-mobile-applications-1VgZvJV1ZAw>