BỘ NÔNG NGHIỆP VÀ MÔI TRƯỜNG
PHÂN HIỆU TRƯỜNG ĐẠI HỌC THỦY LỢI

BÁO CÁO MÔN HỌC KHAI PHÁ DỮ LIỆU

Leaf Disease

Sinh viên thực hiện:

2251068211-Phạm Công Minh

2251068205 – Nguyễn Thành Lợi

Giáo viên hướng dẫn: Ths. Vũ Thị Hạnh

TP. Hồ Chí Minh,ngày 24 tháng 10 năm 2025

Mục Lục

Danh mục hình ảnh

I. GIỚI THIỆU VÀ TÓM TẮT ĐỀ TÀI

1. Giới thiệu đề tài

Trong bối cảnh nông nghiệp phát triển đôi với công nghệ hiện đại, việc phát hiện và phân loại bệnh hại trên cây trồng đóng vai trò quan trọng trong việc đảm bảo năng suất và chất lượng của nông sản. Theo thống kê năm 2025 mặc dù theo Tổ chức Lương thực và Nông nghiệp Liên hợp quốc (FAO) chưa có số liệu cụ thể nhưng FAO đã nhấn mạnh sự ảnh hưởng nghiêm trọng của các bệnh gây hại đến cây trồng như vàng lá,... Đến an ninh lương thực và thu nhập của người nông dân.

Phương pháp truyền thống trong việc chẩn đoán bệnh cây trồng chủ yếu dựa vào kinh nghiệm và quan sát trực tiếp của người nông dân hoặc chuyên gia nông nghiệp. Tuy nhiên, phương pháp này tồn tại nhiều hạn chế: Tốn thời gian, phụ thuộc vào trình độ chuyên môn, khó tiếp cận ở vùng sâu vùng xa, và thường phát hiện bệnh ở giai đoạn muộn khi đã gây thiệt hại đáng kể.

Với sự phát triển của công nghệ thông tin, đặc biệt là trí tuệ nhân tạo và học máy, việc ứng dụng các kỹ thuật khai phá dữ liệu (Data Mining) và học sâu (Deep Learning) vào các bài toán phân loại bệnh lá cây mở ra hướng tiếp cận mới, hiệu quả và khả thi. Các mô hình học máy có khả năng tự động nhận diện và phân loại bệnh thông qua hình ảnh lá cây với độ chính xác cao, giúp phát hiện sớm và đưa ra biện pháp xử lý kịp thời.

Đồ án "Phân loại Bệnh Lá Cây (Leaf Disease Classification)" được thực hiện với mục tiêu xây dựng một hệ thống tự động có khả năng phân loại các loại bệnh phổ biến trên lá cây thông qua phân tích hình ảnh.

Đề tài tập trung vào việc: Tìm hiểu, nghiên cứu và áp dụng các kỹ thuật tiền xử lý hình ảnh, trích xuất đặc trưng từ ảnh lá cây sử dụng các phương pháp xử lý ảnh và học sâu,

xây dựng và so sánh hiệu quả của các mô hình phân loại như CNN (Convolutional Neural Networks) và mô hình machine learning truyền thống SVM.

Kết quả của đồ có tiềm năng ứng dụng thực tiễn cao, góp phần vào việc phát triển nông nghiệp thông minh, giúp người nông dân có công cụ hỗ trợ đắc lực trong việc quản lý sức khỏe cây trồng, từ đó nâng cao năng suất và hiệu quả sản xuất.

2. Bộ dữ liệu

Leaf Disease Classification

Đường dẫn liên kết: https://www.kaggle.com/datasets/vipoooool/new-plant-diseases-dataset

Mô tả dữ liệu:

- Tên: New Plant Diseases Dataset
- Kích thước: Khoảng 87k ảnh, được chia thành 3 tập train(80%), valid(20%), test, về
- Bộ dữ liệu là các ảnh gồm 38 lớp các loại lá bệnh và không bệnh của nhiều cây
    - Apple: scab, black rot, cedar apple rust, healthy
    - Blueberry: healthy
    - Cherry(including sour): helthy, powdery mildew
    - Corn(maize): Cercospora leaf spot gray leaf spot, common rust, healthy, northern leaf blight
    - Grape: black rot, Esca(black measles), healthy, leaf blight(Isariopsis Leaf spot)
    - Orange: Haunglongbing (Citrus greeing)
    - Peach: bacterial spot, healthy
    - Pepper bell: bacterial spot, healthy
    - Potato: early blight, healthy, late blight,
    - Raspberry: healthy
    - Soybean: healthy
    - Squash: powdery mildew

- Strawbery: healthy, leaf scorch
- Tomato : bacterial spot, early blight, healthy, late blight, leaf mold, septoria leaf spot, spider mites two spotted spider mite, target spot, mosaic virus, yellow leaf curl virus

- Mục đích chính: phân loại lá bệnh và không bệnh cho các loại cây, và các loại bệnh có trong dataset
- Dữ liệu đã được tăng cường sẵn

3. Tóm tắt đề tài

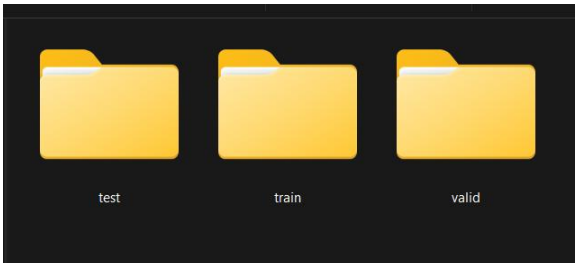
- Mục đích đề tài: Sử dụng các mô hình học sâu để phân loại lá bệnh hay không bệnh theo tường loại cây, nếu là bệnh thì đó là bệnh nào
- Mô hình chọn:
  - CNN (Convolutional Neural Network)
  - ResNet50V2
- Vài nét về mô hình đã chọn
  - **CNN**: là mô hình học sâu được thiết kế để xử lý dữ liệu dạng ảnh. Mô hình này sử dụng các lớp tích chập để tự động trích xuất đặc trưng, giúp đạt độ chính xác cao trong các bài toán như phân loại hình ảnh, nhận dạng đối tượng, và phát hiện bất thường.

  - **ResNet50V2**: mô hình CNN gồm 50 lớp, thuộc họ mạng Residual Network. Mô hình sử dụng cơ chế "skip connection" để khắc phục vấn đề suy giảm độ chính xác khi mạng sâu, giúp việc huấn luyện hiệu quả hơn. Nhờ đó, ResNet50V2 đạt độ chính xác cao trong các bài toán xử lý ảnh và thường được dùng cho học chuyển giao.

II. DỮ LIỆU VÀ CÁC BƯỚC TIỀN XỬ LÝ

1. Dữ liệu

- Được chia ra làm 3 tập



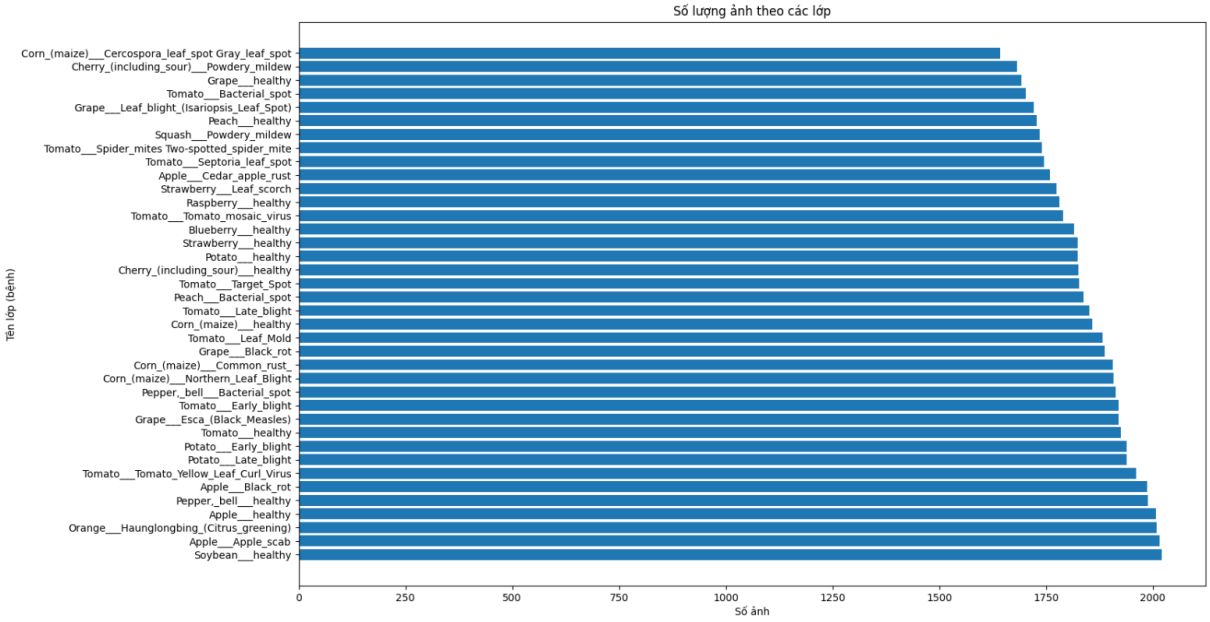
Hình 1: Phân loại 3 nhãn

- Tập train gồm 38 lớp với 70295 ảnh

```
Found 70295 files belonging to 38 classes.
Tổng số lớp:  38
01. Apple___Apple_scab
02. Apple___Black_rot
03. Apple___Cedar_apple_rust
04. Apple___healthy
05. Blueberry___healthy
06. Cherry_(including_sour)___Powdery_mildew
07. Cherry_(including_sour)___healthy
08. Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot
09. Corn_(maize)___Common_rust_
10. Corn_(maize)___Northern_Leaf_Blight
11. Corn_(maize)___healthy
12. Grape___Black_rot
13. Grape___Esca_(Black_Measles)
14. Grape___Leaf_blight_(Isariopsis_Leaf_Spot)
15. Grape___healthy
16. Orange___Haunglongbing_(Citrus_greening)
17. Peach___Bacterial_spot
18. Peach___healthy
19. Pepper,_bell___Bacterial_spot
20. Pepper,_bell___healthy
21. Potato___Early_blight
22. Potato___Late_blight
23. Potato___healthy
24. Raspberry___healthy
25. Soybean___healthy
26. Squash___Powdery_mildew
27. Strawberry___Leaf_scorch
28. Strawberry___healthy
29. Tomato___Bacterial_spot
30. Tomato___Early_blight
31. Tomato___Late_blight
32. Tomato___Leaf_Mold
33. Tomato___Septoria_leaf_spot
34. Tomato___Spider_mites Two-spotted_spider_mite
35. Tomato___Target_Spot
36. Tomato___Tomato_Yellow_Leaf_Curl_Virus
37. Tomato___Tomato_mosaic_virus
38. Tomato___healthy
```

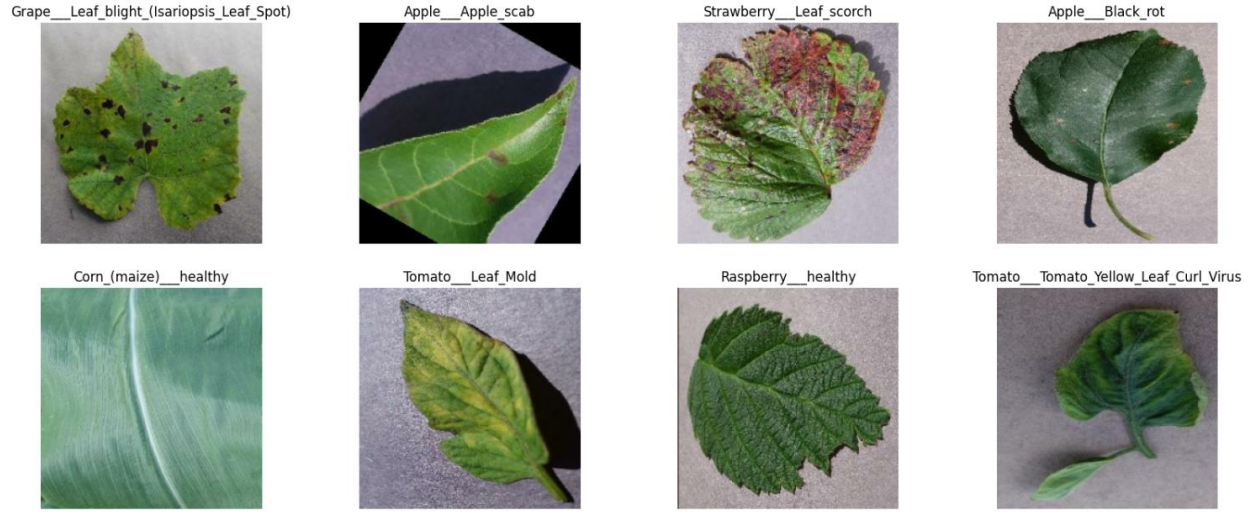
Hình 2: Các lớp

- Dữ liệu ở các lớp được phân bố đều



Hình 3:Phân bố dữ liệu

- Một sô mẫu trong tập dữ liệu



Hình 4:Dữ liệu mẫu

- Các ảnh đều đã cùng kích thướcc (256,256)

```
sample_path = os.path.join(train_dir, class_names[0])
img_path = os.path.join(sample_path, os.listdir(sample_path)[0])

img = PIL.Image.open(img_path)
print("Kích thước ảnh gốc:", img.size)
```

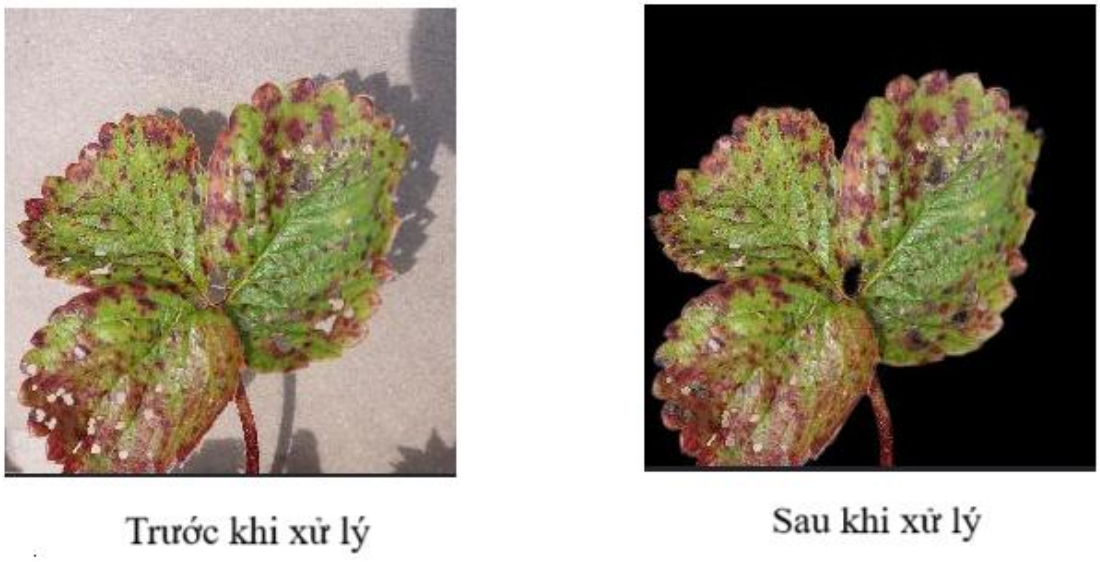
```
Kích thước ảnh gốc: (256, 256)
```

Hình 5:Kích thước ảnh

2. Tiền xử lý dữ liệu

2.1. Dùng hình U2Net để xóa nền

- Dữ liệu gốc là tập dữ liệu được chụp trên một nền xám với lá khỏe hoặc bệnh dưới ánh sáng dịu, nếu không xử lý sẽ làm cho mô hình học nhầm các đặc điểm là một chiếc lá bệnh được trải phẳng nằm trên nền xám, ảnh hưởng sau này mô hình dự đoán thực tế, mục tiêu của việc xóa nền là để cho mô hình bắt buộc phải tập trung vào các đặc điểm quan trong trên lá cây như màu sắc, hình dạng, vết bệnh để phân loại Sử dụng kỹ thuật tách nền hay còn gọi là phân vùng đối tượng, dùng thư viện rembg của python. Mô hình là U2-Net đây là mô hình deep learning tiên tiến cho việc phát hiện vật thể nổi bật, trong bài dùng u2netp là phiên bản nhỏ gọn, nhanh hơn, và phù hợp cho việc xử lý hàng loạt



Hình 6: Xóa nền ảnh

2.2.Xử lý mất cân bằng

- Mặc dù tỉ lệ ảnh giữa các lớp khá cân bằng, nhưng trong 38 lớp có 14 loại cây, có những loại cây chiếm đến 9 lớp nhưng cũng có loại cây chỉ có 1 hoặc 2 lớp làm cho dữ liệu dữ các loại cây bị chênh lệch lớn. Nếu huấn luyện trược tiếp trên mô hình này thì mô hình sẽ trở nên thiên vị cho các các cây có nhiều lớp, kết quả mô hình sẽ có độ chính xác cao trên các lớp đa số nhưng thất bại trên các lớp thiểu số.
- Trọng số càng cao cho thấy lớp đó càng ít dữ liệu

```
Trọng số cho từng loại cây:
Apple: 9.05
Blueberry: 38.71
Cherry_(including_sour): 20.03
Corn_(maize): 9.61
Grape: 9.73
Orange: 34.97
Peach: 19.71
Pepper,_bell: 18.02
Potato: 12.33
Raspberry: 39.47
Soybean: 34.77
Squash: 40.49
Strawberry: 19.54
Tomato: 3.83
```

Hình 7:Trọng số dữ liệu

- Phương pháp cân bằng trọng số lớp Class Weights. Để giải quyết mất cân bằng, chúng ta chọn phương pháp gắn trọng số cho các lớp. Đây là kỹ thuật cho mô hình biết rằng một số lớp quan trọng hơn những lớp khác
- Chọn Class_weight thay vì sử dụng oversampling hay undersampling vì tập dữ liệu này đã được tăng cường sẵn, có một số lớp chỉ có vài ảnh nhưng được tăng cường dữ liệu bằng cách xoay, lật, zoom,... trở nên rất nhiều ảnh, nên nếu dùng hai cách trên sẽ làm dữ liệu đồng nhất dễ overfiting
- Đầu tiên tính số lượng ảnh trên từng nhóm cây, tính trọng số của từng nhóm bằng tổng ảnh chia cho số lớp của nhóm cây, sau đó ánh xạ trọng số của nhóm và từng lớp

Trọng số cuối cùng theo từng lớp:

Chỉ số	Lớp bệnh	Loại cây	Trọng số
0	Apple___Apple_scab	Apple	9.05
1	Apple___Black_rot	Apple	9.05
2	Apple___Cedar_apple_rust	Apple	9.05
3	Apple___healthy	Apple	9.05
4	Blueberry___healthy	Blueberry	38.71
5	Cherry_(including_sour)___Powdery_mildew	Cherry_(including_sour)	20.03
6	Cherry_(including_sour)___healthy	Cherry_(including_sour)	20.03
7	Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot	Corn_(maize)	9.61
8	Corn_(maize)___Common_rust_	Corn_(maize)	9.61
9	Corn_(maize)___Northern_Leaf_Blight	Corn_(maize)	9.61
10	Corn_(maize)___healthy	Corn_(maize)	9.61
11	Grape___Black_rot	Grape	9.73
12	Grape___Esca_(Black_Measles)	Grape	9.73
13	Grape___Leaf_blight_(Isariopsis_Leaf_Spot)	Grape	9.73
14	Grape___healthy	Grape	9.73
15	Orange___Haunglongbing_(Citrus_greening)	Orange	34.97
16	Peach___Bacterial_spot	Peach	19.71
17	Peach___healthy	Peach	19.71
18	Pepper,_bell___Bacterial_spot	Pepper,_bell	18.02
19	Pepper,_bell___healthy	Pepper,_bell	18.02
20	Potato___Early_blight	Potato	12.33
21	Potato___Late_blight	Potato	12.33
22	Potato___healthy	Potato	12.33
23	Raspberry___healthy	Raspberry	39.47
24	Soybean___healthy	Soybean	34.77
25	Squash___Powdery_mildew	Squash	40.49
26	Strawberry___Leaf_scorch	Strawberry	19.54
27	Strawberry___healthy	Strawberry	19.54
28	Tomato___Bacterial_spot	Tomato	3.83
29	Tomato___Early_blight	Tomato	3.83
30	Tomato___Late_blight	Tomato	3.83
31	Tomato___Leaf_Mold	Tomato	3.83
32	Tomato___Septoria_leaf_spot	Tomato	3.83
33	Tomato___Spider_mites Two-spotted_spider_mite	Tomato	3.83
34	Tomato___Target_Spot	Tomato	3.83
35	Tomato___Tomato_Yellow_Leaf_Curl_Virus	Tomato	3.83
36	Tomato___Tomato_mosaic_virus	Tomato	3.83
37	Tomato___healthy	Tomato	3.83

Hình 8:Trọng số của từng lớp

2.3.Kỹ thuật EarlyStopping, ReduceLROnPlateau, ModelCheckpoint

- EarlyStopping: tự dừng quá trình huấn luyện mô hình khi hiệu suất của mô hình không được cải thiện. Chống Overfitting ngăn mô hình học vẹt khi val_loss(lỗi trên tập validation) bắt đầu tăng lên thì sẽ phát hiện và dừng mô hình

```
callbacks = [
    EarlyStopping(
        monitor='val_loss',
        patience=6, #6 epoch khong cai thien dung
        restore_best_weights=True
    ),
```

Hình 9:EarlyStopping

- ReduceLROnPlateau: tự động giảm tốc độ học(learning rate – LR), khi mới bắt đầu tốc độ học cao giúp mô hình học nhanh nhưng khi mô hình gần tiến đến điểm tối ưu nhất thì LR cao có thể làm cho mô hình quanh điểm tối ưu mà không hội tụ chính xác, giảm LR làm cho mô hình hội tụ chính xác để tiến vào sâu hơn

```
ReduceLROnPlateau(
    monitor='val_loss',
    factor=0.5,     # giảm LR 1/2 nếu val_loss không giảm
    patience=3,
    min_lr=1e-6
),
```

Hình 10:ReduceLROnPlateau

- ModelCheckpoint: tự động lưu lại mô hình tốt nhất trong quá trình huấn luyện, không phải lúc nào accuracy cũng tăng, có thể ở epochs 30 tăng nhưng sau đó giảm dần thì lúc đó ModelCheckpoint sẽ chọn cái tốt nhất.

```
    ModelCheckpoint(
        "best_model_CNN.h5",
        monitor='val_accuracy',
        save_best_only=True
    )
]
```

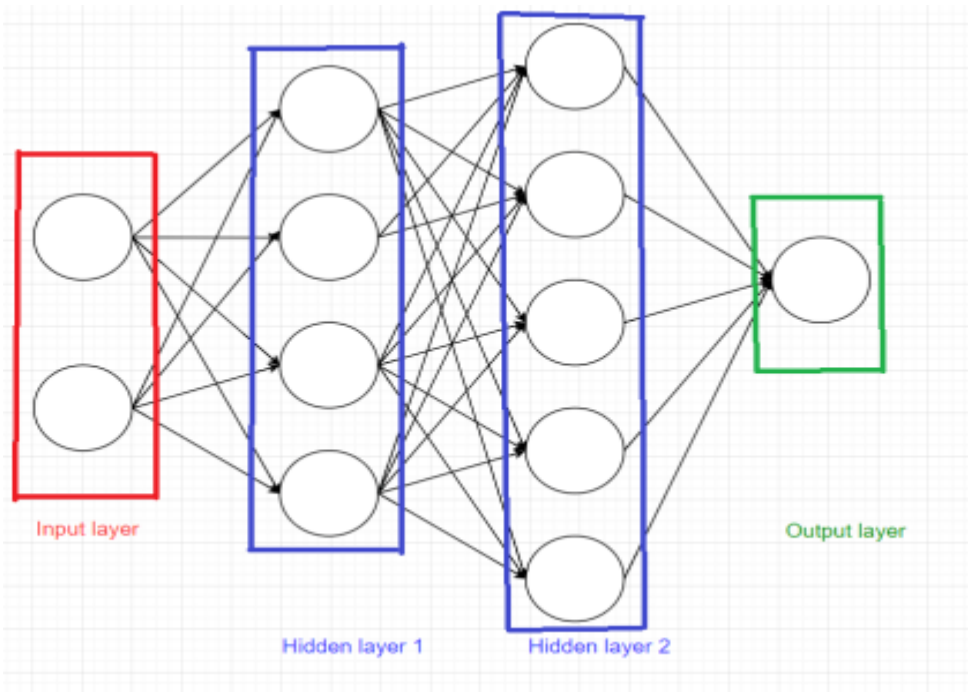
Hình 11:ModelCheckpoint

III. PHƯƠNG PHÁP KHAI PHÁ DỮ LIỆU VÀ MÔ HÌNH

1. Mô hình Convolutional Neural Network (CNN)

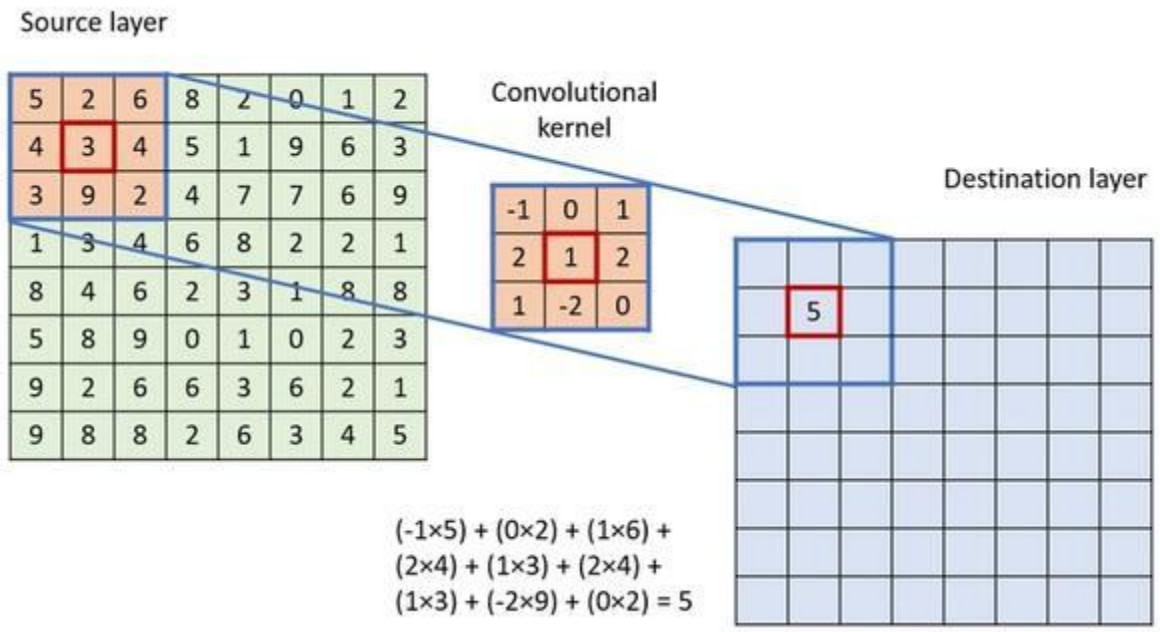
1.1. Cơ sở lý thuyết về CNN

- Kiến trúc mô hình CNN



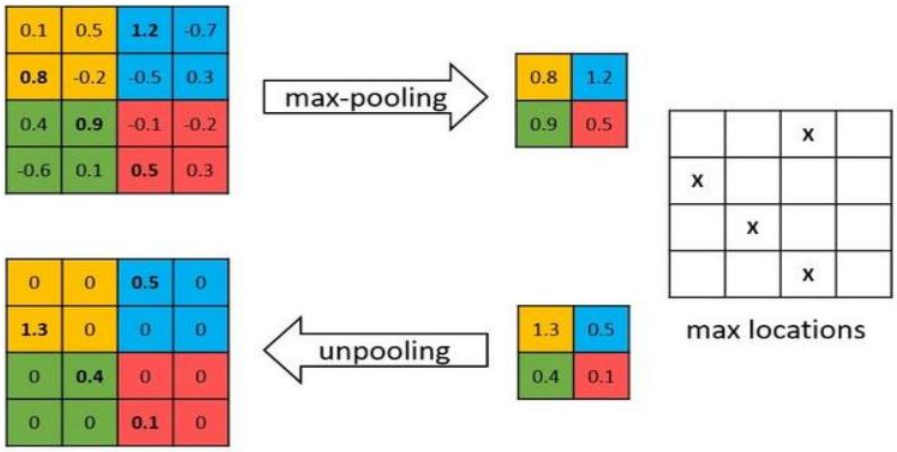
Hình 12:Kiến trúc mô hình CNN cơ bản 1

- CNN là mô hình deep learning được thiết kế để xử lý, phân tích dữ liệu có dạng lưới, phổ biến nhất là hình ảnh. CNN thay vì nhìn toàn bộ bức ảnh cùng lúc, nó hoạt động theo kiểu quét qua từng phần nhỏ của bức ảnh. Tìm các đặc điểm như hình dạng, đường cong, màu sắc. Ở các lớp sâu sẽ hội tụ kết hợp các đặc điểm để nhận diện
- Lớp tích chập (Convolutional Layer): Lớp này là phần quan trọng nhất của toàn mạng CNN, nó có nhiệm vụ thực thi các tính toán. Các yếu tố quan trọng trong lớp Convolutional là: padding, stride, feature map và filter map.
    - Sử dụng filter để áp dụng vào các vùng của ma trận hình ảnh. Các filter map là các ma trận 3 chiều, bên trong đó là những tham số và chúng được gọi là parameters.
    - Stride tức là bạn dịch chuyển filter map theo từng pixel dựa vào các giá trị từ trái qua phải.
    - Padding: Thường, giá trị viền xung quanh của ma trận hình ảnh sẽ được gán các giá trị 0 để có thể tiến hành nhân tích chập mà không làm giảm kích thước ma trận ảnh ban đầu.
    - Feature map: Biểu diễn kết quả sau mỗi lần feature map quét qua ma trận ảnh đầu vào. Sau mỗi lần quét thì lớp Convolutional sẽ tiến hành tính toán.



Hình 13:Kiến trúc mô hình CNN cơ bản 2

- Hàm kích hoạt (Relu Layer): thường đặt sau lớp tích chập, Thêm "tính phi tuyến" (non-linearity) vào mô hình. Nếu không có nó, toàn bộ mạng lưới (dù sâu bao nhiêu) cũng chỉ tương đương một phép toán tuyến tính đơn giản
- Lớp gộp (Pooling Layer): Khi ma trận ảnh đầu vào có kích thước quá lớn, các lớp Pooling layer sẽ được đặt vào giữa những lớp Convolutional để làm giảm những parameters. Hiện, hai loại lớp Pooling được sử dụng phổ biến là Max pooling và Average.



Hình 14:Kiến trúc mô hình CNN cơ bản 3

- Fully connected layer: Đây là lớp có nhiệm vụ đưa ra kết quả sau khi hai lớp Convolutional và Pooling đả nhận được ảnh truyền. Khi này, ta sẽ thu được một model đọc được thông tin của ảnh. Để có thể liên kế chúng cũng như cho nhiều đầu ra hơn ta sẽ sử dụng Fully connected layer. Ngoài ra, nếu lớp này có dữ liệu hình ảnh thì lớp sẽ chuyển chúng thành các much chưa được phân chia chất lượng để tìm ra ảnh có chất lượng cao nhất.

1.2. Chuẩn bị cho mô hình CNN

- Tăng cường dữ liệu: khi huấn luyện mô hình CNN từ đầu rất dễ bị overfitting nên tăng cường dữ liệu để mô hình học tổng quát hơn,
- Sử dụng ImageDataGenerator của keras, chuẩn hóa về khoảng [0,1]. Tăng cường chỉ với tập train, áp dụng các phép biến đổi hình học ngẫu nhiên như xoay, lật, zoom, dịch dọc, ngang.

```
BATCH_SIZE = 32
IMG_SIZE = (224, 224)

train_datagen = ImageDataGenerator(
    rescale=1./255,           # Chuẩn hóa pixel về [0,1]
    rotation_range=20,         # Xoay nhẹ ±20°
    width_shift_range=0.2,     # Dịch ngang
    height_shift_range=0.2,    # Dịch dọc
    zoom_range=0.2,            # Phóng to/thu nhỏ
    horizontal_flip=True       # Lật ngang
)

val_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

train_ds = train_datagen.flow_from_directory(
    train_dir,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    shuffle=True,
    seed=123
)

val_ds = val_datagen.flow_from_directory(
    valid_dir,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    shuffle=False
)
```

Hình 15:Xử lý đầu vào CNN

- Kiến trúc mô hình

```
model = Sequential([

        layers.Input(shape=(224, 224, 3)),

        layers.Conv2D(32, (3,3), padding='same'),
        layers.BatchNormalization(),
        layers.LeakyReLU(),
        layers.MaxPooling2D(2,2),


        layers.Conv2D(64, (3,3), padding='same'),
        layers.BatchNormalization(),
        layers.LeakyReLU(),
        layers.MaxPooling2D(2,2),
        layers.Dropout(0.2),


        layers.Conv2D(128, (3,3), padding='same'),
        layers.BatchNormalization(),
        layers.LeakyReLU(),
        layers.MaxPooling2D(2,2),
        layers.Dropout(0.25),


        layers.Conv2D(256, (3,3), padding='same'),
        layers.BatchNormalization(),
        layers.LeakyReLU(),
        layers.MaxPooling2D(2,2),
        layers.Dropout(0.3),

        layers.Flatten(),
        layers.Dense(256),
        layers.BatchNormalization(),
        layers.LeakyReLU(),
        layers.Dropout(0.4),
        layers.Dense(len(class_names), activation='softmax')
])
```

Hình 16: Khởi tạo các layer CNN

- Sử dụng Sequential xây dựng một mạng CNN truyền thẳng. Gồm một chuỗi các khối Conv-Pool với độ sâu tăng dần để học đặc trưng theo cấp độ
    - Khối 1: trích xuất 32 đặc trưng cơ bản
    - Khối 2: trích xuất 64 đặc trưng phức tạp hơn
    - Khối 3: trích xuất 128 đặc trưng cao
    - Trích xuất 256 đặc trương trừu tượng nhất
- Các thành phần chính mỗi khối
    - Con2D: lớp tích chập
    - BatchNormalization:ổn định và tăng tốc quá trình huấn luyện bằng cách chuẩn hóa đầu ra của lớp trước đó
    - LeakyReLU: hàm kích hoạt
    - MaxPooling2D: lớp gộp, giảm kích thước không gian đặc trưng
    - Dropout: loại bỏ 1 phần nơ – ron trong quá trình huấn luyện
- Tham số huấn luyện

```
Total params: 13,246,438
Trainable params: 13,244,966
Non-trainable params: 1,472
```

Hình 17: Tham số mô hình CNN

    - Tổng kiến trúc có 13tr tham số, 99% là tham số có thể huấn luyện vì đây là mô hìn được xây dựng từ đầu

1.3. Huấn luyện mô hình CNN

```
model.compile(
    optimizer = tf.keras.optimizers.Adam(learning_rate=1e-4),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
history = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=50,
    callbacks=callbacks,
    workers = 6,
    use_multiprocessing = False
)
```

Hình 18: Huấn luyện mô hình CNN

- Bắt đầu huấn luyện mô hình với epochs được thiết lập là 50
- Quá trình huấn luyện được dừng ở epochs 36/50

```
Epoch 30/50
2197/2197 [==============================] - 291s 132ms/step - loss: 0.1953 - accuracy: 0.9395 - val_loss: 1.2253 - val_accuracy: 0.7707 - lr: 6.2500e-06
```

Hình 19: Epoch cuối của CNN

- Mô hình tốt nhất được lưu tại epochs 30

```
best_epoch = np.argmin(history.history['val_loss']) + 1
best_val_acc = history.history['val_accuracy'][best_epoch - 1]
print(f"Best model at epoch {best_epoch} with val_accuracy = {best_val_acc:.2%}")
```

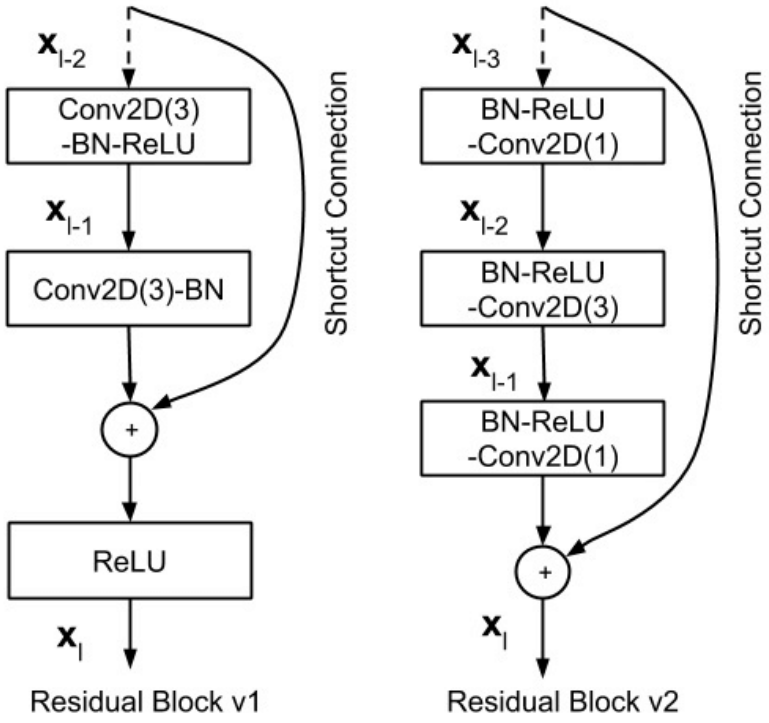
```
Best model at epoch 24 with val_accuracy = 77.11%
```

Hình 20: Độ chính xác mô hình CNN

2. Transfer learning mô hình ReNet50V2

2.1. Cơ sở lý thuyết về ReNet50V2

- Residual Network, 50 layers, version 2
- Kiến trúc mô hình



Hình 21: Kiến trúc mô hình ResNet50V2

- Thay vì xây dựng mô hình từ đầu, chúng ta sử dụng mô hình chuyển giao cụ thể là ResNet50V2, đã học được một hệ thống phân cấp các đặc trưng (features) vô cùng phong phú từ các cạnh, góc, màu sắc ở lớp thấp, đến các kết cấu và hình dạng phức tạp ở lớp sâu
- ResNet (Residual Network) là một kiến trúc CNN mang tính cách mạng, được thiết kế để giải quyết vấn đề "mất mát độ dốc" (vanishing gradients) khi huấn luyện các mạng rất sâu. Ý tưởng đột phá của ResNet là sử dụng các "Khối Thặng dư" (Residual Blocks), hay còn gọi là "Kết nối tắt" (Skip Connections). Thay vì bắt các lớp phải học một phép biến đổi trực tiếp H(x), ResNet cho phép chúng học "phần còn lại" (thặng dư) F(x), sau đó cộng với đầu vào x ban đầu
- ResNet50V2, là phiên bản cải tiến (Version 2) với 50 lớp. Nó tinh chỉnh kiến trúc V1 bằng cách sử dụng "kích hoạt trước" (pre-activation), tức là di chuyển các lớp BatchNormalization và ReLU lên trước lớp Convolution. Cải tiến này giúp đường truyền tín hiệu qua kết nối tắt trở nên thông suốt hoàn toàn, giúp mô hình hội tụ nhanh hơn và ổn định hơn.

2.2. Chuẩn bị cho ResNet50V2

```
# Augmentation cho train
train_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,  #hàm xử lý của ResNetV2 [-1,1]
    rotation_range=30, #xoay
    width_shift_range=0.2,      #dịch trái, phải
    height_shift_range=0.2,     #dịch dọc
    horizontal_flip=True,       #lật ảnh
    brightness_range=[0.7, 1.3], # Thêm thay đổi độ sáng
    zoom_range=0.2,
    fill_mode='constant', cval=0
)

val_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input
)

train_gen = train_datagen.flow_from_directory(
    train_dir,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical'
)

val_gen = val_datagen.flow_from_directory(
    valid_dir,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical'
)
```

Hình 22: Đầu vào ResNet50V2

- Tăng cường dữ liệu và tiền xử lý cho mô hình ResNet50V2, thay vì rescale về khoảng [0,1] như mô hình trước mà dùng preprocessing_function=preprocess_input, một hàm xử lý dành riêng cho các mô hình như ResNetV2, bởi vì đầu vào của mô hình được chuẩn hóa theo một cách khác thường là về khoảng [-1,1], vì thế dùng ham xử lý để mô hình có thể hoạt động hiểu quả

2.3. Huấn luyện mô hình RestNet50V2

```
base_model = ResNet50V2(
    include_top=False,
    weights='imagenet',
    input_shape=(IMG_SIZE[0], IMG_SIZE[1], 3)
)

#Đóng băng các lớp của mô hình gốc
base_model.trainable = False


#Xây dựng mô hình
inputs = base_model.input
x = GlobalAveragePooling2D()(base_model.output)
x = Dropout(0.5)(x) # Thêm Dropout để giảm overfitting
outputs = Dense(NUM_CLASSES, activation='softmax')(x)

model = Model(inputs=inputs, outputs=outputs)

model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

model.summary()
```

Hình 23: giai đoạn 1 mô hình ResNet50V2

- Tải mô hình ResNet50V2 đã được huấn luyện sẵn từ bộ dữ liệu Imagenet.
- Sau đó đóng băng để tái sử dụng kiến thức của ResNet không thay đổi nó, bằng cách trainable = False giữ nguyên toàn bộ trọng số của ResNet50V2, khi huấn luyện không thay đổi bất kỳ trọng số nào. Mục đích dùng để trích xuất đặc trưng, mô hình lúc này hoạt động như một bộ trích xuất đặc trưng cố định, và chúng ta chỉ huấn luyện các lớp mới mà chúng ta thêm vào
- Tham số huấn luyện

```
Total params: 23,642,662
Trainable params: 77,862
Non-trainable params: 23,564,800
```

Hình 24: tham số huấn luyện giai đoạn 1 mô hình ResNet50V2

- Mô hình có tổng cộng 23,6 triệu tham số và đã đóng băng 23,5 triệu tham số, chỉ để mô hình học 77,862 tham số chúng ta đã thêm vào
- Bắt đàu huấn luyện với epochs 50

```
history = model.fit(
    train_gen,
    validation_data=val_gen,
    epochs=50,
    workers = 6,
    use_multiprocessing = False,
    class_weight=final_class_weights,
    callbacks=callbacks
)
```

Hình 25: huấn luyện ResNet50V2 giai đoạn 1

- Fit vào mô hình các tham số như train_gen đã tăng cường và tiền xử lý, valdation_data không tăng cường để đánh giá hiệu suất mô hình, tham số class_weight là để mô hình biết trọng số của các lớp không cân bằng, bắt buộc mô hình học các lớp thiểu số, để cải thiện độ chính xác tổng thể , callbacks dùng để dừng hiệu quả tránh overfitting
- Quá trình huấn luyện dừng lại ở epochs 50

```
Epoch 50/50
2197/2197 [==============================] - 385s 175ms/step - loss: 2.0743 - accuracy: 0.9155 - val_loss: 0.1212 - val_accuracy: 0.9607 - lr: 1.5625e-05
```

Hình 26: Epoch cuối giai đoạn 1 ResNet50V2

- Sau giai đoạn đóng băng mô hình đã huấn luyện xong phần đầu phân loại, nên giờ mở bộ mô hình từ layer 155 và huấn luyện lại trên tập dữ liệu của chúng ta. Điều này cho phép ResNet50V2 vốn học các đặc trưng tổng quát được điều chỉnh để học chuyên biệt với tập dữ liệu của chúng ta. Huấn luyện với learning rate nhỏ

```python
model.trainable = True

fine_tune_at = 155
# Đóng băng tất cả các lớp TRƯỚC lớp 155
for layer in model.layers[:fine_tune_at]:
    layer.trainable = False


#Re-compile với Learning Rate nhỏ
model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=1e-5),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

Hình 27: Giai đoạn 2: fine tunig

- Tham số của mô hình sau khi đã mở từ layer 155

```
Total params: 23,642,662
Trainable params: 15,046,694
Non-trainable params: 8,595,968
```

Hình 28: tham số huấn luyện giai đoan fine tuning ResNet50V2

- Mô hình dừng lại ở epochs 50

```
Epoch 50/50
2197/2197 [==============================] - 383s 174ms/step - loss: 0.0586 - accuracy: 0.9966 - val_loss: 0.0134 - val_accuracy: 0.9969 - lr: 1.0000e-07
```

Hình 29: Epoch cuối của fine tuning ResNet50V2

- Lưu lại mô hình tốt nhất sau fine Tuning

```
val_acc = history_finetune.history['val_accuracy']

best_epoch = val_acc.index(max(val_acc)) + 1

print("Epoch tốt nhất:", best_epoch)
print("Validation Accuracy cao nhất:", max(val_acc))
```

```
Epoch tốt nhất: 47
Validation Accuracy cao nhất: 0.9969838261604309
```

Hình 30: Độ chính xác giai đoạn fine tuning ResNet50V2

IV. KẾT QUẢ VÀ ĐÁNH GIÁ MÔ HÌNH

1. Mô hình CNN cơ bản

1.1. Độ chính xác

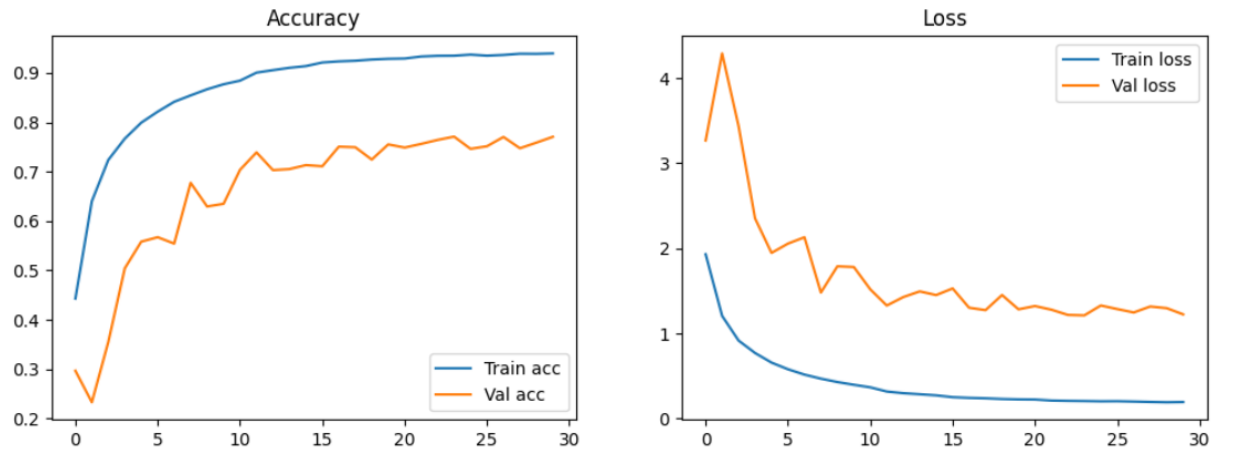
- Độ chính xác: với mô hình CNN cơ bản mang lại độ chính xác cao nhất là 77.11%

```
best_epoch = np.argmin(history.history['val_loss']) + 1
best_val_acc = history.history['val_accuracy'][best_epoch - 1]
print(f"Best model at epoch {best_epoch} with val_accuracy = {best_val_acc:.2%}")
```

```
Best model at epoch 24 with val_accuracy = 77.11%
```

1.2. Biểu đồ Histogram

- Với mô hình CNN cơ bản thì mô hình chạy qua 30/50 epoch, cho ra biểu đồ



Hình 31: Biểu đồ Histogram CNN

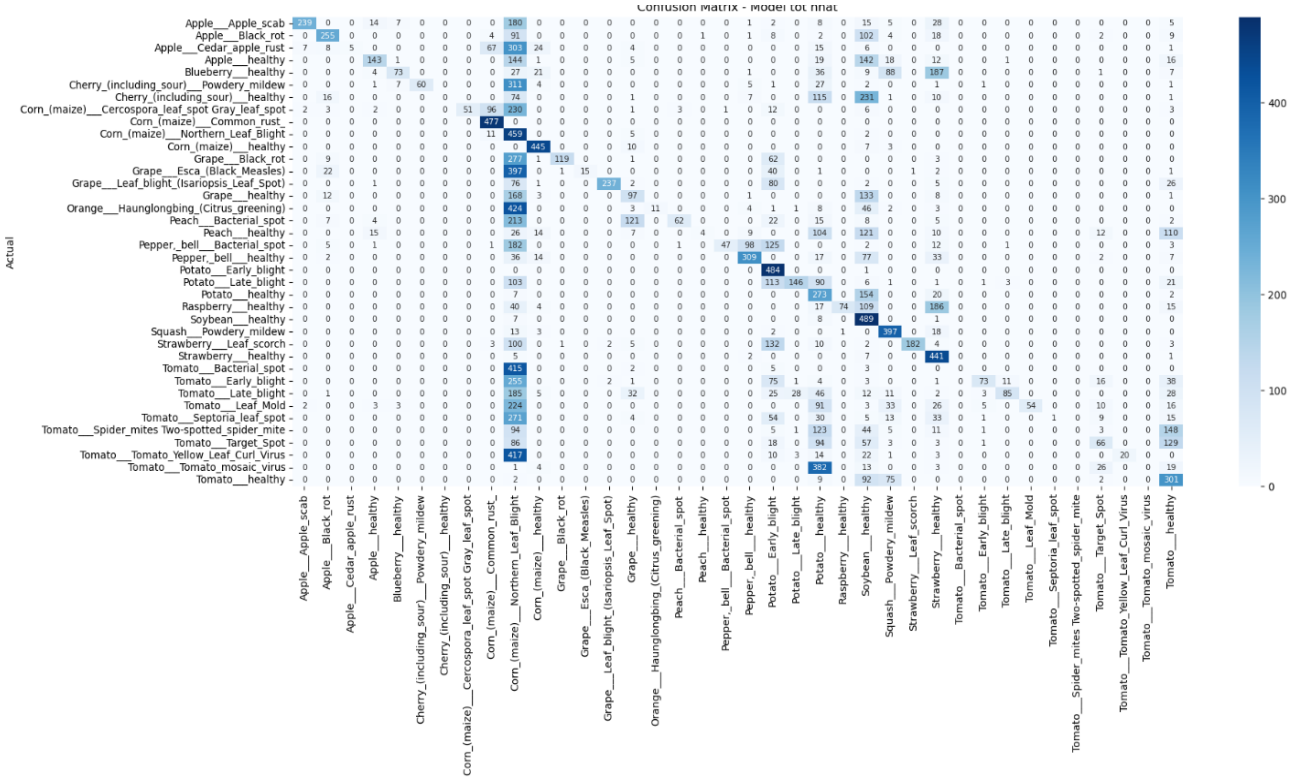
- Với biểu đồ của Accuracy:

- Đường train acc(màu xanh) đường này tăng rất nhanh và đều, mức chính xác khoảng 93% ở epoch 30, cho thấy mô hình học tốt và nhớ gần như toàn bộ dữ liệu của tập huấn luyện
- Đường Val acc(màu cam): đường này tăng khá chậm so với train acc, không ổn định, dao động lên xuống khá nhiều và đạt đỉnh chỉ ở khoảng 75%
- Khoảng cách giữa đường train acc và val acc ngày càng rộng, vào giai đoạn cuối chệch lệch lớn

- Với biểu đồ Loss
  - Đường train loss: giảm nhanh và đều, xuống mức rất thấp gần như 0.2 cho thấy mô hình rất tự tin trên tập huấn luyện
  - Đường val loss: giảm mạnh trong vài epoch đầu tiên mặc dù có một dao động nhẹ ở epoch 2, sau đó ngưng giảm và đi ngang đôi lúc có tăng nhẹ
  - Khoảng cách giữa đường train loss và val loss tương tự như biểu đồ của Accuracy đường train loss giảm đều trong khi val loss đã ngưng giảm

- Mô hình có dấu hiệu overfitting. Với mô hình 13 triệu tham số đã đủ khả năng học thuộc toàn bộ tập huấn luyện, thể hiện qua Train acc rất cao và Train loss rất thấp và thất bại cho việc tổng quát hóa kiến thức đó cho tập dữ liệu mới(tập validation). Tuy đã sử dụng các kỹ thuật tăng cường và Dropout thì mô hình nãy vẫn không phù hợp cho việc huấn luyện từ đầu trên tập dữ liệu có

1.3. Ma trận nhầm lẫn



Hình 32: Ma trận nhầm lẫn CNN

- Ma trận nhầm lẫn là bảng cho chúng ta thấy hiệu xuất 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 mô hình dự đoán nhầm khá nhiều vào lớp Corn Northem Leaf blight, ngoài ra các trường hợp nhầm lẫn xảy ra trên cùng một loài cây cho thấy mô hình học tốt trong việc phân biệt loại cây nhưng khá tệ trong việc các bệnh có chịu chứng tương đồng

2. Mô hình ResNet50V2

2.1. Độ chính xác

- Với Resnet50V2 mang lại độ chính xác là 99.6%

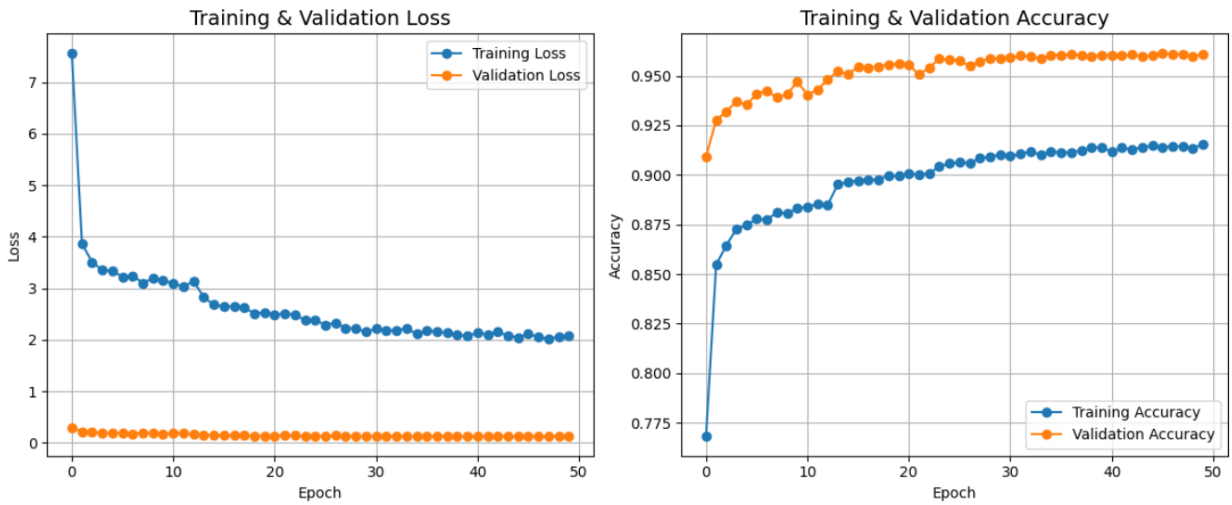
```
val_acc = history_finetune.history['val_accuracy']

best_epoch = val_acc.index(max(val_acc)) + 1

print("Epoch tốt nhất:", best_epoch)
print("Validation Accuracy cao nhất:", max(val_acc))
```

```
Epoch tốt nhất: 47
Validation Accuracy cao nhất: 0.9969838261604309
```

Hình 33: Độ chính xác mô hình ResNet50V2

2.2. Biểu đồ Histogram

Giai đoạn trích xuất đặc trưng:



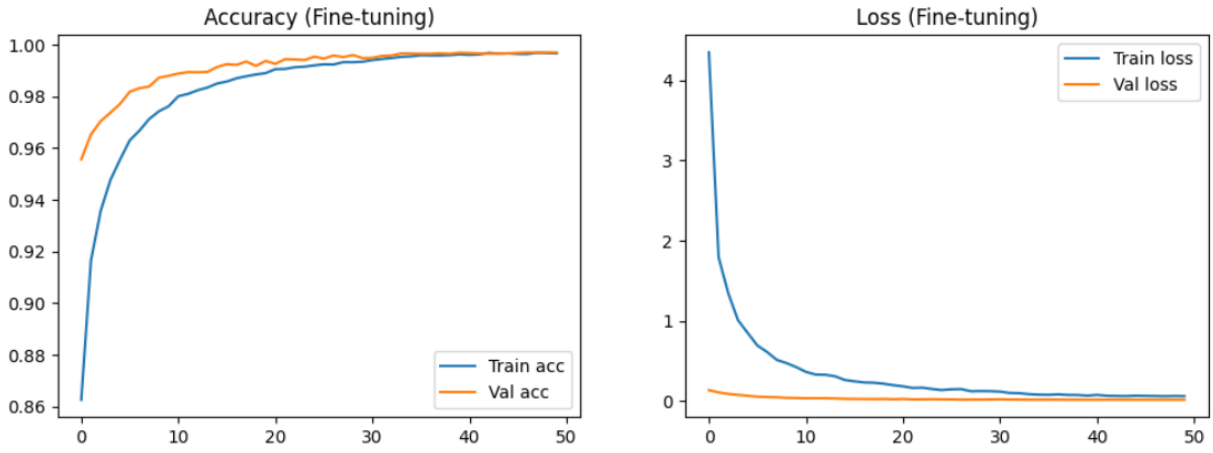
Hình 34: biểu đồ Histogram giai đoạn 1 ResNet50V2

- Biểu đồ Loss
    - Train loss bắt đầu ở khoảng rất cao sau đó giảm mạnh trong 10 epoch đầu. Sau đó tiếp tục giảm đều nhưng chậm hơn, kết thúc ở khoảng 2.0
    - Val loss: bắt đầu ở khoảng rất thấp và duy trì đều ở mức đó cho đến heestt 50 epoch
- Biểu đồ Accuracy
    - Train acc: Bắt đầu ở rất thấp và sau đó tăng đều kết thúc ở khoảng 91%

- Val acc: bắt đầu ở khoảng rất cao khoảng 91% và sau đó tăng dần lên khoảng 96%

- Val acc từ đầu cao hơn train acc và val loss thấp hơn train loss: cho thấy giai đoạn trích xuất đặc trưng cho thấy mô hình hoạt động tốt hội tụ nhanh train loss giảm mạnh và train acc tăng đều, mô hình đã học phân loại hiệu quả. Điểm chú ý là val acc cao hơn đáng kể so với train acc điều này do dữ liệu đưa vào tập train đã được tăng cường điều này làm cho bài toán trên tập train trở nên khó hơn và sự hoạt động của lớp Dropout(0.5) chỉ hoạt động trên tập train làm giảm hiệu suất của train acc

Giai đoạn 2 Fine tuning

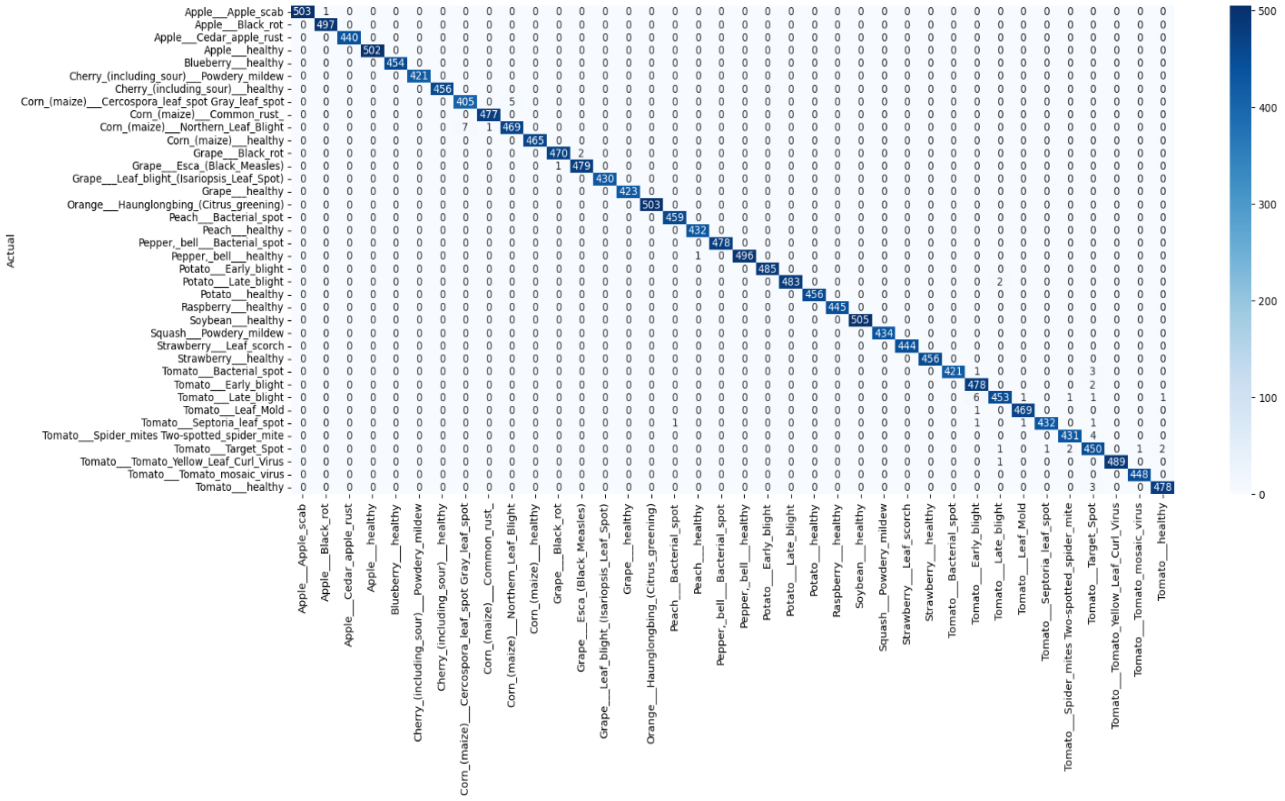


Hình 35: biểu đồ Histogram giai đoạn fine tuning ResNet50V2

- Biểu đồ Accurancy
  - Val acc: bắt đầu ở mức rất cao khoảng 95% đây chính là kết quả tốt nhất từ giai đoạn 1, sau đó tiếp tục tăng ổn định và đạt đỉnh ở mức 99%
  - Train acc: Bắt đầu ở mức thấp hơn khoảng 86% và sau đó tăng rất nhanh, cuối cùng hội tụ gần như song song với val acc
- Biểu đồ Loss
  - Val loss: tiếp tục giảm tử giai đoạn 1 và ổn định ở mức cực kỳ thấp
  - Train loss: Bắt đầu mức cao do chúng ta mở các layer cuối của mô hình mặc dù với learnig rate nhỏ nhưng điều này cũng tạo ra loss khá lớn trên tập train, sau đó train loss giảm đều và về sau hội tự gần như ngang với Val loss

- Giai đoạn fine tuning khá thành công: giai đoạn này đã cải thiện hiệu suất mô hình lên cao, và không có hiện tượng overfitting khi 2 đường train và val của cả accuracy và loss đều hội tụ rất gần nhau.

2.3. Ma trận nhầm lẫn



Hình 36: Ma trận nhầm lẫn ResNet50V2

- Đường chéo chính có màu xanh đậm cho thấy mô hình gần như tuyệt đối.

V. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. Kết luận

- Nghiên cứu đã thực hiện được nhiệm vụ phân loại 38 lớp bệnh trên lá cây từ bộ dữ liệu hình ảnh New Plant Diseases Dataset. Được thực hiện với 2 phương pháp đánh giá
    - Mô hình CNN tùy chỉnh: một kiến trúc CNN với 13,2 triệu tham số được xây dựng từ đầu. Kết quả cho thấy mô hình có hiện tượng overfitting với độ chính xác trên tập huấn luyện là 93% nhưng tập kiểm tra chỉ có 77%
    - Mô hình học chuyển giao ResNet50V2: phương pháp này tận dụng kiến thức từ mô hình ResNet50V2 đã được huấn luyện trước trên ImageNet. Sau đó chúng ta đã áp dụng quy trình huấn luyện 2 giai đoạn đóng băng các layer đầu để trích

xuất đặc trưng và fine tuning các layer cuối để mô hình học sâu vào tập dữ liệu của chúng ta

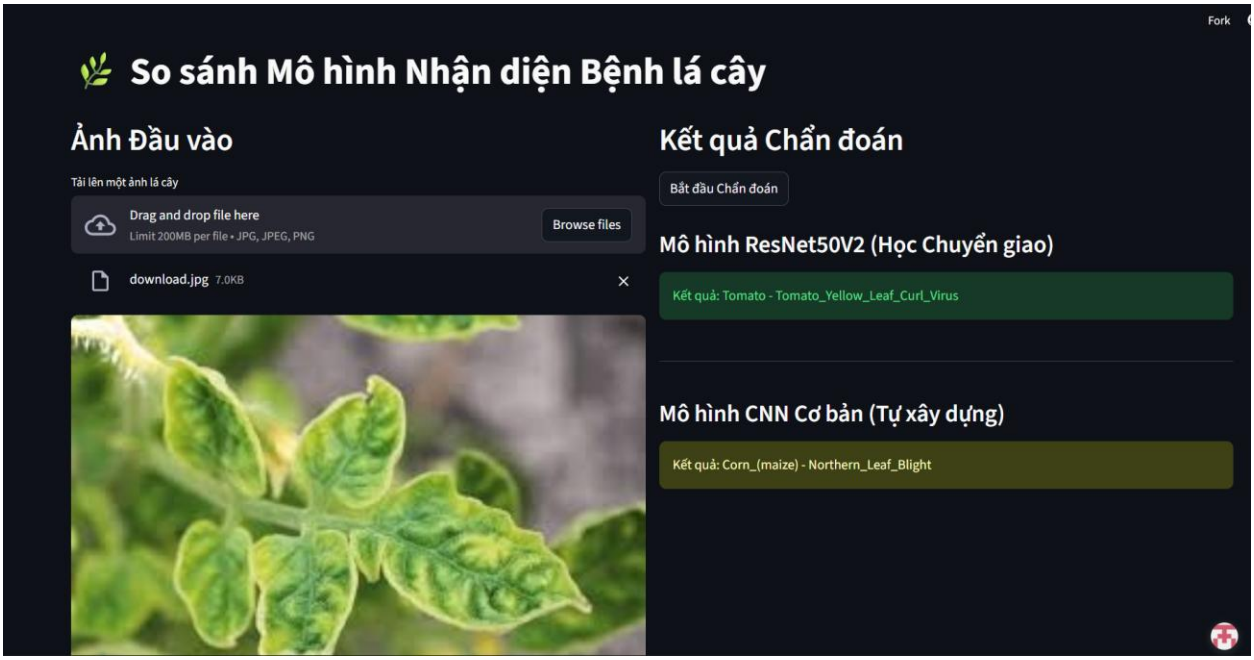
- Kết quả cho thấy mô hình RestNet tốt hơn và đạt độ chính xác trên tập kiểm tra là 99,6%. Ma trận nhầm lẫn xác định mô hình có khả năng phân loại 38 lớp bệnh của cây hoàn hảo các trường hợp sai dường như không có

2. Hướng phát triển

- Triển khai ứng dụng: đưa ứng dụng lên web. Mọi người có thể đưa ảnh lá cây và chẩn đoán ngay
- Mặc dù mô hình có độ chính xác cao nhưng khả năng dự đoán trên thực tế môi trường thấp hơn, do tập dữ liệu huấn luyện và kiểm tra đều được chụp khá rõ nét và có mức tương đồng rất cao, làm cho mô hình có thể bối rối khi đưa các dữ liệu thực tế chẳng hạn lá chụp không đầy đủ chi tiết, hoặc là lá không phải cùng loại trong tập huấn luyện
- Hướng phát triển: thu thập thêm các dự liệu có ảnh chụp từ môi trường thực tế, để có khả năng tổng quát hóa trong môi trường đời thực. Tối ưu hóa mô hình để có thể chạy mượt trên các thiết bị
- Mở rộng bài toán: có thể mở rộng mô hình không chỉ nhận diện bệnh mà là ước lượng mức độ nghiêm trọng của bệnh ví dụ như 10% lá hoặc 50% lá bị ảnh hưởng, đề xuất các phương án điều trị

VI. KẾT QUẢ ĐẠT ĐƯỢC

- Link dự án: https://leaf-disease-min.streamlit.app
- Link model: https://huggingface.co/min190503/RN5/tree/main



Hình 37: Giao diện web

VII. TÀI LIỆU THAM KHẢO

1. https://docs.streamlit.io/deploy/streamlit-community-cloud
2. https://www.tensorflow.org/api_docs/python/tf/keras/applications/ResNet50V2
3. https://www.tensorflow.org/tutorials/images/cnn
4. https://www.tensorflow.org/api_docs/python/tf/keras/applications/resnet_v2/preprocess_input
5. https://www.tensorflow.org/api_docs/python/tf/keras/Model#fit