**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC NÔNG LÂM TP HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

TIỂU LUẬN TỐT NGHIỆP

**NGHIÊN CỨU MÔ HÌNH HỌC MÁY NHẬN DẠNG LOẠI BỆNH TRÊN CÂY LÚA Ở GIAI ĐOẠN ĐẺ NHÁNH**

**Ngành : Công nghệ thông tin**

**Niên khoá : 2021-2025**

**Lớp : DH21DTC**

**Sinh viên thực hiện : Kiều Thị Ngọc Qúy**

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC NÔNG LÂM TP HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

TIỂU LUẬN TỐT NGHIỆP

**NGHIÊN CỨU MÔ HÌNH HỌC MÁY NHẬN DẠNG LOẠI BỆNH TRÊN CÂY LÚA Ở GIAI ĐOẠN ĐẺ NHÁNH**

**CÁN BỘ HƯỚNG DẪN SINH VIÊN THỰC HIỆN**

TS. Nguyễn Thị Phương Trâm Kiều Thị Ngọc Qúy (MSSV: 21130504)

TP. HỒ CHÍ MINH, tháng 4 năm 2024

# DANH SÁCH CHỮ VIẾT TẮT

# DANH MỤC HÌNH ẢNH

# DANH MỤC BẢNG

# TÓM TẮT

NÊU MỤC TIÊU NGHIÊN CỨU TRONG GIAI ĐOẠN. PHƯƠNG PHÁP SỬ DỤNG TRONG NGHIÊN CỨU. KẾT QUẢ CỦA NGHIÊN CỨU, ĐÓNG GÓP CỦA NGHIÊN CỨU (NẾU CÓ). THU HẸP PHẠM VI NGHIÊN CỨU CỦA ĐÊ TÀI.

**MỤC LỤC**

# MỞ ĐẦU

## LÝ DO CHỌN ĐỀ TÀI

Lúa là cây trồng chủ lực ở Việt Nam, đóng vai trò quan trọng trong an ninh lương thực và kinh tế. Theo thống kê của FAO (2023), Việt Nam sản xuất khoảng 43 triệu tấn lúa mỗi năm, là một trong những nước xuất khẩu gạo hàng đầu thế giới [1]. Tuy nhiên, các bệnh trên cây lúa, đặc biệt trong giai đoạn đẻ nhánh – thời kỳ cây phát triển chồi và lá non, có thể làm giảm năng suất từ 20-50% nếu không được phát hiện và xử lý kịp thời [2]. Các bệnh phổ biến như bạc lá, đạo ôn lá và khô vằn gây thiệt hại lớn, nhưng việc chẩn đoán chủ yếu dựa vào quan sát thủ công, tốn thời gian và kém chính xác.

Sự phát triển của học máy, đặc biệt là các mô hình học sâu như mạng nơ-ron tích chập (CNN), đã mở ra cơ hội để tự động hóa việc nhận dạng bệnh trên cây lúa, giúp cải thiện độ chính xác và tiết kiệm chi phí. Ở Việt Nam, nghiên cứu về nhận dạng bệnh lúa trong giai đoạn đẻ nhánh bằng học máy còn hạn chế, chủ yếu tập trung vào các phương pháp truyền thống. Vì vậy, đề tài này tập trung vào việc áp dụng các mô hình học sâu (DenseNet121, InceptionV3, MobileNetV2, ResNet50) để nhận dạng các bệnh trên lá lúa trong giai đoạn đẻ nhánh, đồng thời so sánh với các phương pháp học máy truyền thống như Random Forest, SVM và kNN, nhằm đề xuất giải pháp hiệu quả cho nông dân.

## MỤC TIÊU VÀ PHẠM VI NGHIÊN CỨU

Mục tiêu nghiên cứu:

* Áp dụng các mô hình học sâu (DenseNet121, InceptionV3, MobileNetV2, ResNet50) để phân loại các bệnh trên lá lúa trong giai đoạn đẻ nhánh, sử dụng tập dữ liệu Rice Leaf Diseases Detection.
* So sánh hiệu suất của các mô hình học sâu với các phương pháp truyền thống (Random Forest, SVM, kNN).
* Phân tích ưu nhược điểm của từng phương pháp trong việc nhận dạng bệnh lúa.

Phạm vi nghiên cứu:

* Tập dữ liệu: Sử dụng tập dữ liệu Rice Leaf Diseases Detection từ Kaggle, giới hạn vào 6 lớp (Bacterial Leaf Blight, Leaf Blast, Leaf Scald, Brown Spot, Sheath Blight, Healthy Rice Leaf), tương ứng với các bệnh phổ biến trong giai đoạn đẻ nhánh [4].
* Mô hình: Áp dụng các mô hình học sâu (DenseNet121, InceptionV3, MobileNetV2, ResNet50) và các mô hình truyền thống (Random Forest, SVM, kNN).
* Ứng dụng: Kết quả nghiên cứu tập trung vào nhận dạng bệnh trong giai đoạn đẻ nhánh, không bao gồm các giai đoạn sinh trưởng khác.

## Ý NGHĨA KHOA HỌC VÀ THỰC TIỄN CỦA ĐỀ TÀI

Ý nghĩa khoa học:

* Ứng dụng học sâu vào nhận dạng bệnh trên cây lúa, góp phần làm phong phú nghiên cứu về AI trong nông nghiệp tại Việt Nam.
* Đánh giá hiệu quả của các mô hình học sâu và truyền thống, cung cấp cơ sở cho các nghiên cứu tiếp theo về tối ưu hóa thuật toán.

Ý nghĩa thực tiễn:

* Hỗ trợ nông dân phát hiện sớm các bệnh trên cây lúa, giảm thiệt hại năng suất và chi phí quản lý.
* Đề xuất giải pháp tự động hóa chẩn đoán bệnh, dễ triển khai trên các thiết bị đơn giản, phù hợp với nông dân ở vùng nông thôn Việt Nam.

# TỔNG QUAN ĐỀ TÀI

## Các công trình nghiên cứu liên quan

Trong những năm gần đây, trí tuệ nhân tạo (AI) và học máy đã được ứng dụng rộng rãi trong nông nghiệp, đặc biệt trong việc nhận dạng và quản lý bệnh trên cây trồng. Các nghiên cứu liên quan đến nhận dạng bệnh trên cây lúa đã đạt được nhiều kết quả đáng chú ý. Chẳng hạn, nghiên cứu của Lu et al. (2017) sử dụng mạng nơ-ron tích chập (CNN) để phân loại các bệnh trên lá lúa, đạt độ chính xác lên đến 95% trên tập dữ liệu tự xây dựng [5]. Tương tự, nghiên cứu của Zhang et al. (2019) áp dụng các mô hình học sâu như VGG16 và ResNet50 để nhận dạng bệnh đạo ôn và bạc lá, cho thấy hiệu quả vượt trội so với các phương pháp truyền thống như SVM và Random Forest [6]. Tuy nhiên, hầu hết các nghiên cứu này sử dụng tập dữ liệu tổng quát, không tập trung vào một giai đoạn sinh trưởng cụ thể của cây lúa, chẳng hạn giai đoạn đẻ nhánh, là thời kỳ quan trọng quyết định năng suất. Ở Việt Nam, mặc dù có nhiều nghiên cứu về bệnh lúa, nhưng các công trình tập trung vào nhận dạng bệnh trong giai đoạn đẻ nhánh bằng học máy còn hạn chế, chủ yếu dựa vào phương pháp quan sát thủ công hoặc phân tích phòng thí nghiệm [3]. Điều này tạo cơ hội cho việc áp dụng các mô hình học sâu để cải thiện hiệu quả chẩn đoán bệnh trong giai đoạn này.

## Những vấn đề còn tồn tại

Mặc dù đã có nhiều tiến bộ trong việc ứng dụng AI vào nhận dạng bệnh cây trồng, vẫn còn một số hạn chế. Thứ nhất, dữ liệu hình ảnh về bệnh trên cây lúa, đặc biệt trong giai đoạn đẻ nhánh, còn thiếu hụt và không đồng nhất. Các tập dữ liệu lớn như Rice Leaf Diseases Detection trên Kaggle cung cấp nhiều hình ảnh, nhưng không phải tất cả đều phù hợp với giai đoạn đẻ nhánh hoặc có đặc điểm hình thái rõ ràng [4]. Thứ hai, sự tương đồng về hình thái giữa một số bệnh (ví dụ: đốm nâu và đốm nâu hẹp) có thể gây nhầm lẫn cho mô hình học máy, làm giảm độ chính xác [7]. Thứ ba, việc xác thực tập dữ liệu đòi hỏi kiến thức chuyên môn cao về bệnh học cây trồng, trong khi các nghiên cứu ở Việt Nam còn thiếu các công cụ tự động hóa để hỗ trợ nông dân chẩn đoán bệnh sớm. Cuối cùng, các mô hình học sâu thường yêu cầu tài nguyên tính toán lớn, gây khó khăn khi triển khai trên các thiết bị di động phục vụ nông dân ở vùng nông thôn.

## Nội dung nghiên cứu

Nghiên cứu tập trung vào bài toán nhận dạng và phân loại các bệnh trên lá lúa trong giai đoạn đẻ nhánh, sử dụng các mô hình học sâu và so sánh với các phương pháp học máy truyền thống.

Tìm hiểu bài toán nhận dạng bệnh lúa trong giai đoạn đẻ nhánh, phân tích đặc điểm của các bệnh phổ biến như bạc lá, đạo ôn lá, cháy lá, đốm nâu và khô vằn.

Nghiên cứu các mô hình học sâu (DenseNet121, InceptionV3, MobileNetV2, ResNet50), phân tích ưu nhược điểm của chúng trong phân loại hình ảnh.

Xử lý và sử dụng tập dữ liệu Rice Leaf Diseases Detection, giới hạn vào 6 lớp (Bacterial Leaf Blight, Leaf Blast, Leaf Scald, Brown Spot, Sheath Blight, Healthy Rice Leaf).

Áp dụng kỹ thuật tiền xử lý dữ liệu (data augmentation), tinh chỉnh siêu tham số để tối ưu hóa mô hình.

Đánh giá và so sánh hiệu suất các mô hình dựa trên các chỉ số Accuracy, Precision, Recall, F1-score.

**Lý do lựa chọn 6 lớp dữ liệu**

Tập dữ liệu Rice Leaf Diseases Detection bao gồm 9 lớp, nhưng nghiên cứu này chọn 6 lớp (Bacterial Leaf Blight, Leaf Blast, Leaf Scald, Brown Spot, Sheath Blight, Healthy Rice Leaf) với 7.537 hình ảnh, dựa trên các tiêu chí sau:

* **Phổ biến trong giai đoạn đẻ nhánh**: Các bệnh như đạo ôn lá, khô vằn và bạc lá là những bệnh chính trong giai đoạn đẻ nhánh, có thể gây thiệt hại năng suất 20-50% [2]. Đốm nâu và cháy lá cũng thường xuất hiện trên lá non trong giai đoạn này [7].
* **Phù hợp với giai đoạn đẻ nhánh**: Các bệnh được chọn đều ảnh hưởng đến lá hoặc bẹ lá trong giai đoạn đẻ nhánh, trong khi lớp lá khỏe mạnh giúp mô hình nhận diện trạng thái không bệnh [2].
* **Giảm nhầm lẫn và độ phức tạp**: Các lớp bị loại (Narrow Brown Leaf Spot, Neck Blast, Rice Hispa) ít liên quan đến giai đoạn đẻ nhánh hoặc dễ gây nhầm lẫn. Ví dụ, Neck Blast chủ yếu xuất hiện ở giai đoạn trỗ, còn Rice Hispa là tổn thương do côn trùng [8].

# PHƯƠNG PHÁP VÀ NỘI DUNG NGHIÊN CỨU

## Cơ sở lý thuyết

### Tập dữ liệu Rice Leaf Diseases Detection [4]

Trong nghiên cứu này, tập dữ liệu Rice Leaf Diseases Detection được lấy từ Kaggle, bao gồm 9.790 hình ảnh lá lúa chụp bằng máy ảnh kỹ thuật số độ phân giải cao trong điều kiện ánh sáng tự nhiên [4]. Các hình ảnh được thu thập từ các cánh đồng lúa, đảm bảo tính chân thực và rõ nét, phù hợp để phân tích và nhận dạng bệnh bằng các mô hình học sâu. Trong nghiên cứu này, tập dữ liệu được giới hạn vào 6 lớp (Bacterial Leaf Blight, Leaf Blast, Leaf Scald, Brown Spot, Sheath Blight, Healthy Rice Leaf) với tổng cộng 7.537 hình ảnh, được chọn dựa trên tính phổ biến, đặc điểm hình thái rõ ràng và sự phù hợp với giai đoạn đẻ nhánh. Có 6 lớp được sử dụng để phân loại, cụ thể như sau:

* **Bacterial Leaf Blight** (Bạc lá) (1.197 hình ảnh): Bệnh bạc lá, do vi khuẩn Xanthomonas oryzae gây ra, là một trong những bệnh nghiêm trọng trên cây lúa, đặc biệt trong giai đoạn đẻ nhánh ở vùng nhiệt đới ẩm ướt. Bệnh biểu hiện qua các vết thấm nước trên mép lá, sau đó chuyển thành màu xám trắng hoặc vàng nhạt, với viền rõ ràng. Các vết này có thể lan rộng, gây héo lá và giảm khả năng quang hợp. Theo Ou (1985), bệnh bạc lá thường xuất hiện trên lá non, làm giảm số lượng chồi hữu hiệu nếu không được kiểm soát [7].
* **Leaf Blast** (Đạo ôn lá) (1.748 hình ảnh): Bệnh đạo ôn lá, do nấm Magnaporthe oryzae gây ra, là bệnh phá hoại nhất trên lúa, đặc biệt trong giai đoạn đẻ nhánh khi lá non dễ bị nhiễm. Ban đầu, bệnh xuất hiện dưới dạng các vết thấm nước nhỏ, sau đó mở rộng thành các vết đạo ôn lớn màu xám hoặc trắng, đôi khi có viền nâu. Các vết này có thể gây chết lá, ảnh hưởng nghiêm trọng đến sự phát triển của cây. IRRI (2020) ghi nhận rằng bệnh đạo ôn lá có thể làm giảm năng suất lên đến 50% trong điều kiện thuận lợi [2].
* **Leaf Scald** (Cháy lá) (1.332 hình ảnh): Bệnh cháy lá, do vi khuẩn Rhynchosporium oryzae gây ra, thường xuất hiện trên lá trong giai đoạn đẻ nhánh, đặc biệt ở vùng khí hậu ẩm. Bệnh được nhận diện bởi các vết thương kéo dài, màu vàng nhạt hoặc xám, thường xuất hiện ở mép lá. Các vết này có thể làm giảm diện tích lá quang hợp, ảnh hưởng đến sự phát triển chồi. Theo Mew (1993), bệnh cháy lá phổ biến ở các vùng nhiệt đới và có thể gây thiệt hại đáng kể nếu cây bị stress [9].
* **Brown Spot** (Đốm nâu) (1.546 hình ảnh): Bệnh đốm nâu, do nấm Cochliobolus miyabeanus gây ra, thường xuất hiện trong giai đoạn đẻ nhánh, đặc biệt khi cây lúa thiếu dinh dưỡng hoặc gặp điều kiện ẩm ướt. Bệnh đặc trưng bởi các vết đốm nhỏ, màu nâu sẫm với quầng vàng xung quanh, có thể lan rộng và gây tổn thương mô lá. Các vết đốm này làm giảm khả năng quang hợp và sức khỏe cây. Ou (1985) chỉ ra rằng bệnh đốm nâu là một trong những bệnh phổ biến ở các vùng trồng lúa châu Á [7].
* **Sheath Blight** (Khô vằn) (1.629 hình ảnh): Bệnh khô vằn, do nấm Rhizoctonia solani gây ra, ảnh hưởng chủ yếu đến bẹ lá và thân trong giai đoạn đẻ nhánh. Bệnh biểu hiện qua các vết xám trắng với viền nâu trên bẹ lá, có thể lan lên lá và gây héo. Trong trường hợp nghiêm trọng, bệnh dẫn đến đổ ngã, làm giảm năng suất. Theo IRRI (2020), khô vằn là một trong những bệnh quan trọng ở các vùng trồng lúa nhiệt đới, đặc biệt trong điều kiện ẩm cao [2].
* **Healthy Rice Leaf** (Lá lúa khỏe mạnh) (1.085 hình ảnh): Lớp lá lúa khỏe mạnh bao gồm các hình ảnh lá không bị ảnh hưởng bởi bệnh, có màu xanh đồng đều, không có vết đốm, vết thấm nước, hay dấu hiệu héo. Đây là lớp chuẩn so sánh, giúp mô hình nhận diện các đặc điểm của lá lúa trong trạng thái bình thường ở giai đoạn đẻ nhánh. Sự hiện diện của lớp này rất quan trọng để phân biệt giữa trạng thái khỏe mạnh và các trạng thái bệnh lý, đảm bảo độ chính xác của mô hình [10].

### Các công nghệ và thư viện sử dụng

Bài nghiên cứu sử dụng các công nghệ và thư viện sau để triển khai, huấn luyện và đánh giá các mô hình nhận dạng bệnh trên lá lúa trong giai đoạn đẻ nhánh. Dưới đây là định nghĩa và các tính năng chính của từng công nghệ/thư viện:

**Keras**: Là một API mã nguồn mở, cấp cao, được viết bằng Python, dùng để xây dựng và huấn luyện các mô hình học sâu. Nó cung cấp giao diện đơn giản, tích hợp với TensorFlow để hỗ trợ phát triển nhanh các mạng nơ-ron [11]. Cung cấp các mô hình học sâu như DenseNet121, InceptionV3, MobileNetV2, ResNet50 để phân loại hình ảnh. Hỗ trợ tiền xử lý hình ảnh, bao gồm tăng cường dữ liệu (data augmentation) như xoay, lật, hoặc thay đổi độ sáng. Cho phép tùy chỉnh kiến trúc mạng, huấn luyện mô hình trên GPU và lưu trữ trọng số.

**Scikit-learn**: Là một thư viện học máy mã nguồn mở, được viết bằng Python, tích hợp nhiều thuật toán học máy truyền thống và các công cụ xử lý dữ liệu [12]. Hỗ trợ các thuật toán như Random Forest, SVM, kNN để phân loại và đánh giá mô hình. Cung cấp các công cụ tiền xử lý (chuẩn hóa, chia tập dữ liệu), trích xuất đặc trưng và tinh chỉnh siêu tham số. Tích hợp các chỉ số đánh giá như Accuracy, Precision, Recall, F1-score.

**OpenCV** (Open Source Computer Vision Library): Là một thư viện mã nguồn mở, được viết bằng C++ và có giao diện Python, chuyên về xử lý hình ảnh và thị giác máy tính [13]. Hỗ trợ đọc, ghi và xử lý hình ảnh (thay đổi kích thước, chuyển đổi kênh màu, chuẩn hóa). Cung cấp các kỹ thuật tăng cường dữ liệu và trích xuất đặc trưng hình ảnh. Tối ưu hóa hiệu suất xử lý trên CPU và GPU.

**Google Drive**: Là một dịch vụ lưu trữ đám mây của Google, cho phép lưu trữ, chia sẻ và truy cập dữ liệu từ mọi thiết bị có kết nối internet [14]. Lưu trữ tập dữ liệu, mô hình đã huấn luyện và kết quả thí nghiệm một cách an toàn. Hỗ trợ chia sẻ dữ liệu dễ dàng và tích hợp trực tiếp với Google Colab. Đảm bảo truy cập nhanh vào dữ liệu từ môi trường đám mây.

**Google Colab**: Là một nền tảng tính toán trực tuyến miễn phí, cung cấp môi trường Jupyter Notebook để viết và chạy mã Python, tích hợp tài nguyên GPU/TPU [15]. Hỗ trợ chạy các thư viện như Keras, Scikit-learn và OpenCV để xây dựng mô hình học sâu. Tích hợp với Google Drive để truy cập dữ liệu và lưu kết quả. Cho phép thực hiện các thí nghiệm mà không cần máy tính cấu hình cao.

**Transfer Learning** (Học chuyển giao): Là kỹ thuật học máy, trong đó mô hình đã được huấn luyện trước trên tập dữ liệu lớn (như ImageNet) được tái sử dụng cho một bài toán mới với tập dữ liệu nhỏ hơn [16]. Tận dụng các trọng số đã huấn luyện để giảm thời gian và chi phí tính toán. Cải thiện hiệu suất trên tập dữ liệu nhỏ, như Rice Leaf Diseases Detection. Phù hợp với các mô hình phức tạp như DenseNet121, ResNet50. Trong nghiên cứu này, transfer learning được áp dụng để sử dụng các mô hình đã huấn luyện trước, sau đó tinh chỉnh cho bài toán nhận dạng bệnh lúa.

**Fine-tune** (Tinh chỉnh): Là quá trình điều chỉnh một mô hình đã được huấn luyện trước bằng cách huấn luyện lại các lớp cuối trên tập dữ liệu cụ thể, nhằm tối ưu hóa cho bài toán mới [17]. Tùy chỉnh mô hình để phù hợp với số lớp đầu ra (6 lớp bệnh lúa trong nghiên cứu này). Cải thiện độ chính xác bằng cách học các đặc trưng đặc thù của tập dữ liệu. Kết hợp giữ lại kiến thức chung và học thêm kiến thức mới. Trong nghiên cứu này, fine-tune được sử dụng để tinh chỉnh các mô hình học sâu trên tập dữ liệu Rice Leaf Diseases Detection, đảm bảo hiệu quả trong phân loại bệnh lúa.

## Mô hình CNN

### Định nghĩa CNN

Convolutional Neural Network (Mạng Nơ-ron Tích Chập) là mạng nơ-ron nhân tạo thường được sử dụng trong xử lý dữ liệu ảnh hoặc dữ liệu có cấu trúc lưới nhờ khả năng nhận diện và phân tích các đặc điểm quan trọng trong ảnh.

### Cấu trúc mạng CNN

Cấu trúc của mạng CNN bao gồm nhiều lớp được sắp xếp tuần tự để xử lý dữ liệu từ đầu vào đến đầu ra, thường được chia thành hai giai đoạn chính: trích xuất đặc trưng và phân loại. Các thành phần chính của CNN bao gồm lớp đầu vào, tích chập, kích hoạt, gộp, chuẩn hóa, dropout, kết nối đầy đủ và đầu ra; chúng được thiết kế để học các đặc trưng từ hình ảnh và đưa ra dự đoán phân loại ví dụ như nhận diện bệnh trên lá lúa trong bài nghiên cứu này.

*Hình 2.1: Cấu trúc cơ bản của CNN*

* **Input Layer** (Lớp đầu vào): Nhận hình ảnh với kích thước cố định, được biểu diễn dưới dạng (W, H, C), trong đó W là chiều rộng, H là chiều cao, C là số kênh màu (thường là 3 với ảnh RGB hoặc 1 với ảnh xám). Một số kích thước đầu vào phổ biến bao gồm 224x224x3, 299x299x3, 32x32x3, 28x28x1, tùy thuộc vào đặc trưng của bài toán và mô hình sử dụng. Lớp này đóng vai trò chuẩn hóa dữ liệu đầu vào để đảm bảo tính nhất quán trong quá trình xử lý.
* **Convolution Layer** (Lớp tích chập): Là thành phần cốt lõi của CNN, chịu trách nhiệm trích xuất đặc trưng từ hình ảnh đầu vào chẳng hạn như cạnh, góc, kết cấu. Lớp này sử dụng các bộ lọc kernel là ma trận nhỏ, thường có kích thước 3x3 hoặc 5x5 để trượt trên hình ảnh và thực hiện phép tích chập, tạo ra bản đồ đặc trưng (feature map). Quá trình trượt được điều chỉnh bởi hai siêu tham số:
  + **Stride** (bước nhảy): Là số pixel mà bộ lọc di chuyển sau mỗi bước, thường là 1 hoặc 2.
  + **Padding** (đệm): Được thêm vào viền ảnh để xử lý các cạnh, đặc biệt khi số pixel không chia hết cho kích thước bộ lọc, giúp giữ kích thước đầu ra không bị giảm quá nhiều.
* **Activation Layer** (Lớp kích hoạt): Được đặt sau lớp tích chập để tăng tính phi tuyến, giúp mạng học được các đặc trưng phức tạp hơn. Các hàm kích hoạt phổ biến bao gồm ReLU (Rectified Linear Unit), Sigmoid và Tanh. Trong đó, ReLU thường được sử dụng vì tính đơn giản và hiệu quả; chuyển các giá trị âm về 0 và giữ nguyên giá trị dương; tăng tốc độ hội tụ của mô hình. Lớp kích hoạt cũng có thể được áp dụng sau lớp kết nối đầy đủ (Fully Connected Layer - Dense Layer) để tối ưu hóa đầu ra.
* **Pooling Layer** (Lớp gộp): Được sử dụng để giảm kích thước không gian của bản đồ đặc trưng từ các lớp tích chập, giúp giảm số lượng tham số, tăng khả năng tổng quát hóa, và tránh hiện tượng quá khớp. Lớp này sử dụng một cửa sổ (window), thường có kích thước *k* x *k* (ví dụ: 2x2) để trượt trên bản đồ đặc trưng và thực hiện phép tính giảm chiều. Các loại gộp phổ biến bao gồm:
  + **Max Pooling**: Lấy giá trị lớn nhất trong mỗi vùng cửa sổ, giữ lại các đặc trưng nổi bật. Ví dụ, với cửa sổ 2x2 và stride 2, một bản đồ đặc trưng 224x224 sẽ giảm xuống 112x112.
  + **Average Pooling**: Tính giá trị trung bình trong mỗi vùng cửa sổ, giữ lại thông tin tổng quát.
  + **L2 Pooling**: Tính tổng bình phương các giá trị trong cửa sổ, sau đó lấy căn bậc hai, ít phổ biến hơn nhưng có thể được sử dụng để cân bằng giữa max và average pooling.
  + **Global Max Pooling**: Đặt kích thước cửa sổ bằng kích thước đầu vào, lấy giá trị lớn nhất toàn cục trên mỗi bản đồ đặc trưng. Ví dụ, với đầu vào 7x7x512, global max pooling tạo ra vector 1x1x512, giữ lại giá trị nổi bật nhất từ mỗi kênh.
  + **Global Average Pooling**: Tương tự Global Max Pooling, nhưng lấy giá trị trung bình toàn cục, tạo ra vector 1 chiều với thông tin tổng quát hơn.
* **Batch Normalization/ Layer Normalization** (Lớp chuẩn hóa): Được sử dụng để ổn định quá trình huấn luyện và tăng tốc độ hội tụ của mô hình. Có hai loại chuẩn hóa chính bao gồm:
  + **Batch Normalization:** Chuẩn hóa dữ liệu bằng cách điều chỉnh các giá trị trong mỗi lớp ẩn sao cho chúng có giá trị trung bình gần bằng 0 và độ lệch chuẩn gần bằng 1; làm giảm hiện tượng vanishing gradient (độ lớn của gradient giảm xuống quá thấp, làm chậm quá trình cập nhật trọng số [18]) hoặc exploding gradient (sự gia tăng đột ngột của gradient, dẫn đến việc cập nhật trọng số không ổn định [18]). Điều này giúp huấn luyện ổn định hơn, đặc biệt là với DNN.
  + **Layer Normalization:** Chuẩn hóa dữ liệu theo từng nơ-ron, không phụ thuộc vào kích thước batch, thường được sử dụng trong các mô hình như RNN, nhưng cũng có thể áp dụng trong CNN.
* **Dropout Layer** (Lớp Dropout): Được sử dụng để giảm “overfitting” (nguy cơ quá khớp) trong quá trình huấn luyện bằng cách ngắt ngẫu nhiên một số nơ-ron với dropout rate cho trước, thường từ 0.2 đến 0.5. Dropout buộc mô hình phải học các đặc trưng từ các tổ hợp nơ-ron khác nhau, làm tăng khả năng tổng quát hóa trên dữ liệu mới; giúp mô hình không bị phụ thuộc quá nhiều vào các đặc trưng cụ thể của tập huấn luyện và đảm bảo hiệu quả trên các dữ liệu thực tiễn như trong bài toán phân loại hình ảnh.
* **Fully Connected Layer** **- Dense Layer** (Lớp kết nối đầy đủ): Thường được đặt ở giai đoạn cuối của mạng, nó kết nối với tất cả các nơ-ron của các lớp trước đó để tổng hợp các đặc trưng đã trích xuất để đưa ra quyết định.
* **Output Layer** (Lớp đầu ra): Trả về kết quả cuối cùng của mô hình, tùy vào loại bài toán mà sẽ cho ra kết quả khác nhau, gồm:
  + **Phân loại nhị phân**: Đầu ra gồm 1 nơ-ron với hàm kích hoạt sigmoid, trả về giá trị trong khoảng [0, 1], biểu thị xác suất của một trong hai lớp.
  + **Phân loại đa lớp**: Đầu ra gồm *n* nơ-ron, tương ứng với *n* lớp cần phân loại, sử dụng hàm softmax để tạo ra xác suất cho mỗi lớp với tổng xác suất bằng 1.
  + **Hồi quy**: Đầu ra là một nơ-ron, trả về giá trị liên tục, không cần hàm kích hoạt.

### Mô hình Resnet50 [19]

*Hình 2.2: Cấu trúc mô hình ResNet50*

ResNet50 là một kiến trúc mạng nơ-ron tích chập (CNN) thuộc họ ResNet (Residual Network), được phát triển để giải quyết vấn đề suy giảm hiệu suất trong các mạng nơ-ron sâu (DNN). Họ ResNet bao gồm nhiều biến thể với độ sâu khác nhau như ResNet18, ResNet34, ResNet101, trong đó ResNet50 là một biến thể có độ sâu trung bình.

Vấn đề chính mà ResNet nhắm đến là khi mạng nơ-ron trở nên quá sâu, độ chính xác đạt đến mức bão hòa và sau đó giảm nhanh chóng. Sự suy giảm này không phải do overfitting, mà do khó khăn trong việc tối ưu hóa quá trình huấn luyện, đặc biệt là vấn đề vanishing gradient.

ResNet50 bao gồm 50 tầng, bao gồm 48 lớp tích chập, 1 lớp gộp cực đại và 1 lớp gộp trung bình trước khi đi qua lớp kết nối đầy đủ. Với lớp đầu vào là hình ảnh kích thước 224x224x3 (3 kênh màu RGB). Tiếp theo là tiền xử lý với lớp tích chập 7x7 và lớp gộp cực đại để giảm kích thước và chiết xuất đặc trưng cơ bản, đồng thời giữ lại các thông tin quan trọng cho các khối tiếp theo. Khối dư (Residual Block) mỗi khối dư bao gồm tích chập 1x1 (giảm số kênh để giảm chi phí tính toán), tích chập 3x3 (học các đặc trưng không gian), tích chập 1x1 (khôi phục số kênh); khối này sử dụng Batch Normalization để chuẩn hóa đầu ra của các lớp tích chập để ổn định huấn luyện; sử dụng hàm kích hoạt ReLU để thêm tính phi tuyến tính, giúp mô hình học các đặc trưng phức tạp; với Residual Connection, trong đầu vào của khối được cộng trực tiếp với khối đầu ra của khối trước khi đi qua ReLU, theo công thức *y = F(x) + x* , giúp mạng tập trung học các đặc trưng mới mà không làm mất đặc trưng từ các lớp trước và đồng thời giúp làm giảm vấn đề biến mất gradient. Cuối cùng là lớp cuối, bao gồm lớp gộp trung bình toàn cục làm giảm kích thước xuống 1x1x2048, sau đó là lớp kết nối đầy đủ để dự đoán lớp.

Resnet50 có ưu điểm là có khả năng học tốt trên DNN và làm giảm vấn đề biến mất gradient, nhược điểm là nặng đồng thời là cần nhiều tài nguyên để huấn luyện, không phù hợp cho thiết bị di động.

### Mô hình MobileNetV2 [20]

*Hình 2.3: Cấu trúc mô hình MobileNetV2*

MobileNetV2 là một kiến trúc mạng nơ-ron tích chập (CNN) thuộc họ MobileNet, được phát triển để giải quyết vấn đề chi phí tính toán cao trong các mạng nơ-ron sâu (DNN) khi triển khai trên thiết bị di động. Họ MobileNet bao gồm nhiều biến thể như MobileNetV1, MobileNetV3, trong đó MobileNetV2 là một biến thể cải tiến với hiệu suất cao hơn.

Vấn đề chính mà MobileNetV2 nhắm đến là các mạng nơ-ron sâu thường nặng và tiêu tốn nhiều tài nguyên, không phù hợp cho các thiết bị di động có bộ nhớ và năng lượng hạn chế. MobileNetV2 sử dụng tích chập tách biệt chiều sâu và khối nghịch đảo dư để giảm chi phí tính toán.

MobileNetV2 bao gồm 53 tầng, với 17 khối nghịch đảo dư (inverted residual blocks), xen kẽ với các tầng tích chập và gộp. Với lớp đầu vào là hình ảnh kích thước 224x224x3 (3 kênh màu RGB). Tiếp theo là tiền xử lý với lớp tích chập 3x3 để giảm kích thước và chiết xuất đặc trưng cơ bản, đồng thời giữ lại các thông tin quan trọng cho các khối tiếp theo. Khối nghịch đảo dư (Inverted Residual Block) mỗi khối bao gồm tích chập 1x1 (mở rộng số kênh, thường gấp 6 lần), tích chập chiều sâu 3x3 (học đặc trưng không gian, áp dụng bộ lọc 3x3 riêng cho từng kênh), tích chập 1x1 (thu hẹp số kênh, không dùng ReLU để tránh mất thông tin); khối này không sử dụng Batch Normalization hay ReLU ở bước cuối để giữ thông tin; với kết nối tắt, trong đó đầu vào của khối được cộng trực tiếp với đầu ra nếu số kênh khớp, giúp tăng hiệu quả học. Cuối cùng là lớp cuối, bao gồm lớp tích chập 1x1 (tăng số kênh lên 1280), lớp gộp trung bình toàn cục làm giảm kích thước xuống 1x1x1280, sau đó là lớp kết nối đầy đủ để dự đoán lớp.

MobileNetV2 có ưu điểm là nhẹ và nhanh, phù hợp cho thiết bị di động, nhược điểm là hiệu suất thấp trên dữ liệu phức tạp và học đặc trưng yếu.

### Mô hình InceptionV3 [21]

*Hình 2.4: Cấu trúc mô hình InceptionV3*

InceptionV3 là một kiến trúc mạng nơ-ron tích chập (CNN) thuộc họ Inception (hay GoogLeNet), được phát triển để giải quyết vấn đề chi phí tính toán cao và hiệu suất trên dữ liệu phức tạp trong các mạng nơ-ron sâu (DNN). Họ Inception bao gồm nhiều biến thể như InceptionV1, InceptionV2, trong đó InceptionV3 là một biến thể cải tiến với hiệu quả cao hơn.

Vấn đề chính mà InceptionV3 nhắm đến là làm thế nào để chiết xuất đặc trưng ở nhiều tỷ lệ khác nhau mà không làm tăng quá nhiều chi phí tính toán, đồng thời cải thiện hiệu suất trên các tập dữ liệu lớn như ImageNet. InceptionV3 sử dụng các mô-đun Inception để xử lý nhiều kích thước bộ lọc cùng lúc.

InceptionV3 bao gồm 48 tầng, với các mô-đun Inception là thành phần chính, xen kẽ với các tầng gộp. Với lớp đầu vào là hình ảnh kích thước 299x299x3 (3 kênh màu RGB). Tiếp theo là tiền xử lý với các tầng tích chập 3x3 và gộp cực đại để giảm kích thước và chiết xuất đặc trưng cơ bản, đồng thời giữ lại các thông tin quan trọng cho các khối tiếp theo. Mô-đun Inception mỗi mô-đun gồm 4 nhánh xử lý song song: nhánh 1 với tích chập 1x1 (chiết xuất đặc trưng đơn giản), nhánh 2 với tích chập 1x1 và tích chập 3x3 (học đặc trưng lớn hơn), nhánh 3 với tích chập 1x1 và hai tích chập 3x3 (thay cho 5x5, học đặc trưng phức tạp), nhánh 4 với gộp cực đại 3x3 và tích chập 1x1 (giữ thông tin không gian); các nhánh này được nối (concatenate) trên chiều kênh để tạo đầu ra. Cuối cùng là lớp cuối, bao gồm lớp gộp trung bình toàn cục làm giảm kích thước xuống 1x1x2048, sau đó là lớp kết nối đầy đủ để dự đoán lớp.

InceptionV3 có ưu điểm là hiệu quả cao trên dữ liệu lớn và học đặc trưng đa dạng, nhược điểm là kiến trúc phức tạp và khó triển khai trên thiết bị yếu.

### Mô hình DenseNet121 [22]

*Hình 2.2: Cấu trúc mô hình DenseNet121*

DenseNet121 là một kiến trúc CNN thuộc họ DenseNet (Densely Connected Convolutional Networks), được thiết kế để cải thiện hiệu quả của mạng nơ-ron bằng cách tăng cường luồng thông tin giữa các tầng. Họ DenseNet có nhiều biến thể như DenseNet169, DenseNet201, trong đó DenseNet121 là phiên bản có độ sâu trung bình.

Vấn đề chính mà DenseNet giải quyết là sự mất mát thông tin trong các mạng sâu, cũng như số lượng tham số lớn. DenseNet sử dụng kết nối dày đặc (dense connectivity) để tái sử dụng đặc trưng và giảm số tham số.

DenseNet121 bao gồm 121 tầng, chia thành 4 khối dày đặc (dense blocks), xen kẽ với các tầng chuyển tiếp (transition layers). Với lớp đầu vào là hình ảnh kích thước 224x224x3 (3 kênh màu RGB). Tiếp theo là tiền xử lý với lớp tích chập 7x7 và lớp gộp cực đại để giảm kích thước và chiết xuất đặc trưng cơ bản, đồng thời giữ lại các thông tin quan trọng cho các khối tiếp theo. Khối dày đặc (Dense Block) mỗi tầng trong khối nhận đầu vào từ tất cả các tầng trước đó thông qua phép nối (concatenation); mỗi tầng bao gồm tích chập 1x1 (giảm số kênh để giảm chi phí tính toán), tích chập 3x3 (học các đặc trưng không gian); khối này sử dụng Batch Normalization để chuẩn hóa đầu ra của các lớp tích chập để ổn định huấn luyện; sử dụng hàm kích hoạt ReLU để thêm tính phi tuyến tính, giúp mô hình học các đặc trưng phức tạp; với kết nối dày đặc, giúp tái sử dụng đặc trưng từ các tầng trước, giảm số tham số và tăng hiệu quả học. Tầng chuyển tiếp (Transition Layer) bao gồm tích chập 1x1 (giảm số kênh) và gộp trung bình (2x2, stride 2) để giảm kích thước không gian. Cuối cùng là lớp cuối, bao gồm lớp gộp trung bình toàn cục làm giảm kích thước xuống 1x1x1024, sau đó là lớp kết nối đầy đủ để dự đoán lớp.

DenseNet121 có ưu điểm là ít tham số hơn ResNet và học hiệu quả nhờ tái sử dụng đặc trưng, nhược điểm là cần nhiều bộ nhớ và tốc độ chậm trên phần cứng yếu.

## Random Forest [23]

### Định nghĩa Random Forest

### Cách hoạt động của thuật toán

### Ưu và nhược điểm

## K-Nearest Neighbors [24]

### Định nghĩa của K-Nearest Neighbors (kNN)

### Cách hoạt động của thuật toán

### Ưu và nhược điểm

## Support Vector Machine [25]

### Định nghĩa Support Vector Machine (SVM)

### Cách hoạt động của thuật toán

### Ưu và nhược điểm

## Đánh giá mô hình

Mô hình được đánh giá dựa trên các chỉ số được tính từ confusion matrix (ma trận nhầm lẫn).

Confusion matrix là một bảng được dùng để đánh giá hiệu suất của mô hình phân loại nhị phân hoặc đa lớp.



Hình 1: Confusion Matrix

Với:

* **TP**: số lượng mẫu dương được dự đoán là dương.
* **TN**: số lượng mẫu âm được dự đoán là âm.
* **FP**: số lượng mẫu âm được dự đoán là dương.
* **FN**: số lượng mẫu dương được dự đoán là âm.

Mô hình được đánh giá dựa trên:

* **Accuracy**: đo tỷ lệ dự đoán đúng trên toàn bộ mẫu dữ liệu.
* **Precision**: đo tỷ lệ dự đoán dương đúng trên tổng số dự đoán dương.
* **Recall**: đo tỷ lệ dự đoán dương đúng trên tổng số thực tế dương.
* **F1-Score**: còn gọi là thước đo F, cho biết sự cân bằng giữa precision và recall.
* **Đường cong ROC**: là biểu đồ được vẽ bằng cách tính tỷ lệ dương thực tế (recall) và tỷ lệ dương sai ().
* **AUC**: là diện tích dưới đường cong ROC. AUC đo lường khả năng phân biệt giữa các lớp của mô hình. AUC nằm trong khoảng từ 0 đến 1, mô hình có thể được xem là tốt khi AUC từ 0.8 trở lên.

Vẽ biểu đồ huấn luyện: Sử dụng Matplotlib để vẽ biểu đồ theo dõi độ chính xác (accuracy) và độ mất mát (loss) trên tập train và tập validation. Biểu đồ trực quan hóa quá trình huấn luyện của mô hình, giúp phát hiện các vấn đề trong quá trình huấn luyện như overfitting.

# GIẢI PHÁP CHO BÀI TOÁN

## Phát biểu bài toán

## Giải pháp cụ thể để nhận dạng bệnh

### Giải pháp tiền xử lý dữ liệu

### Xây dựng mô hình

### Huấn luyện và tối ưu mô hình

## Hiện thực giải pháp

### Môi trường triển khai thực nghiệm

Ngôn ngữ lập trình: Python.

Môi trường chạy: Google Colab với RAM, tích hợp với Google Drive để lưu trữ dữ liệu và kết quả mô hình.

Thông tin tập dữ liệu

Bảng 3.1: Thông tin tập dữ liệu

|  |  |
| --- | --- |
| **Thông tin** | **Mô tả** |
| Tên dataset |  |
| Số lượng ảnh |  |
| Định dạng ảnh |  |
| Số lớp |  |
| Số lượng ảnh trong tập huấn luyện |  |
| Số lượng ảnh trong tập xác thực |  |
| Số lượng ảnh trong tập kiểm tra |  |
| Tên các lớp: số lượng ảnh |  |

### Kết quá thực nghiệm [[1]](#footnote-1)

Hyperparameter của từng mô hình fine-tune:

Bảng 3.2: Hyperparameter của từng mô hình học sâu

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Hyperparameter** | **ResNet50** | **InceptionV3** | **DenseNet121** | **MobileNetV2** |
| **Số lớp trainable** |  |  |  |  |
| **Learning Rate** | 0.001 (trước khi fine-tune), 0.0001 (khi fine-tune) | | | 0.001 (trước khi fine-tune), 0.0005 (khi fine-tune) |
| **Batch Size** | 32 | 32 | 32 | 16 |
| **Số Epochs** | 20 | 24 | 25 | 20 |
| **Optimizer** | Adam | | | |
| **Dropout** | 0.5 | 0.5 | 0.5 | 0.6 |
| **Accuracy (Train)** |  |  |  |  |
| **Accuracy (Val)** |  |  |  |  |
| **Thời gian huấn luyện (Epochs)** |  |  |  |  |

Với:

* **Số lớp trainable**: Số lớp được mở khi fine-tune.
* **Learning Rate**: Tốc độ học của mô hình.
* **Batch** **Size**: Số lượng hình ảnh trong mỗi batch.
* **Số** **Epochs**: Số vòng huấn luyện mô hình.
* **Optimizer**: Thuật toán được dùng để tối ưu mô hình.
* **Dropout**: Tỷ lệ ngắt ngẫu nhiên các nơ-ron.
* **Accuracy (Train):** Độ chính xác trên tập huấn luyện.
* **Accuracy (Val):** Độ chính xác trên tập kiểm tra.
* **Thời gian huấn luyện (Epoch):** Thời gian huấn luyện trung bình trên mỗi epochs.

Kết quả thực nghiệm với phương pháp fine-tune các mô hình CNN gồm ResNet50, InceptionV3, DenseNet121, MobileNetV2 và các phương pháp truyền thống như SVM, kNN, Random Forest.

**ResNet50**

*Hình 3.5: Biểu đồ theo dõi hiệu suất mô hình ResNet50*

*Hình 3.6: Confusion Matrix của mô hình ResNet50*

*Hình 3.7: Biểu đồ ROC và chỉ số AUC các class của mô hình InceptionV3*

**InceptionV3**

*Hình 3.8: Biểu đồ theo dõi hiệu suất mô hình InceptionV3*

*Hình 3.9: Confusion Matrix của mô hình InceptionV3*

*Hình 3.10: Biểu đồ ROC và chỉ số AUC từng lớp của mô hình ResNet50*

**DenseNet121**

*Hình 3.11: Biểu đồ theo dõi hiệu suất mô hình DenseNet121*

*Hình 3.12: Confusion Matrix của mô hình DenseNet121*

*Hình 3.13: Biểu đồ ROC và chỉ số AUC từng lớp của mô hình InceptionV3*

**MobileNetV2**

*Hình 3.14: Biểu đồ theo dõi hiệu suất mô hình MobileNetV2*

*Hình 3.15: Confusion Matrix của mô hình MobileNetV2*

*Hình 3.16: Biểu đồ ROC và chỉ số AUC từng lớp của mô hình MobileNetV2* **Random Forest**

*Hình 3.17: Confusion Matrix của Random Forest*

*Hình 3.18: Biểu đồ ROC và chỉ số AUC từng lớp của Random Forest*

**SVM**

*Hình 3.19: Confusion Matrix của SVM*

*Hình 3.20: Biểu đồ ROC và chỉ số AUC từng lớp của SVM*

**kNN**

*Hình 3.21: Confusion Matrix của kNN*

*Hình 3.22: Biểu đồ ROC và chỉ số AUC từng lớp của kNN*

Sử dụng các chỉ số Accuracy, Precision, Recall, F1 – Score để đánh giá mô hình.

Bảng 3.3: Bảng thống kê kết quả của từng mô hình

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Mô hình** | **Accuracy** | **Precision** | **Recall** | **F1-Score** |
| **RestNet50** | 0.9891 | 0.9892 | 0.9891 | 0.9891 |
| **InceptionV3** | 0.9859 | 0.9861 | 0.9859 | 0.986 |
| **DenseNet121** | 0.9789 | 0.9792 | 0.9789 | 0.979 |
| **MobileNetV2** | **0.9914** | **0.9915** | **0.9914** | **0.9914** |
| **SVM** | 0.9266 | 0.93 | 0.93 | 0.93 |
| **kNN** | 0.9274 | 0.93 | 0.93 | 0.93 |
| **Random Forest** | 0.8259 | 0.83 | 0.83 | 0.83 |

### Phân tích, đánh giá và nhận xét kết quả

Dựa vào chỉ số Accuracy, Precision, Recall, F1-Score của từng mô hình rút ra được kết luận sau:

# KẾT QUẢ, KẾT LUẬN VÀ KIẾN NGHỊ

## Kết quả

## Kết luận

## Kiến nghị

# TÀI LIỆU THAM KHẢO

[1] Food and Agriculture Organization of the United Nations, “World Food and Agriculture Statistical Yearbook 2023,” 2023. Accessed: May 1, 2025. [Online]. Available: https://www.fao.org/documents/card/en/c/cc8166en

[2] International Rice Research Institute (IRRI), “Rice Diseases: Identification and Management,” 2020. Accessed: May 1, 2025. [Online]. Available: https://www.irri.org/resources/rice-diseases

[3] T. T. Tran, H. V. Nguyen, and Q. L. Pham, “Quản lý bệnh trên cây lúa tại Việt Nam,” Tạp chí Khoa học Nông nghiệp Việt Nam, vol. 18, no. 3, pp. 205–214, 2020.

[4] Kaggle, “Rice Leaf Diseases Detection Dataset,” 2023. Accessed: May 1, 2025. [Online]. Available: https://www.kaggle.com/datasets/loki4514/rice-leaf-diseases-detection/data

[5] Y. Lu, S. Yi, N. Zeng, Y. Liu, and Y. Zhang, “Deep Learning for Rice Disease Detection,” Computers and Electronics in Agriculture, vol. 140, pp. 321–329, Aug. 2017.

[6] X. Zhang, Y. Qiao, F. Meng, C. Fan, and M. Zhang, “Convolutional Neural Networks for Rice Disease Classification,” Precision Agriculture, vol. 20, no. 5, pp. 1047–1060, Oct. 2019.

[7] S. H. Ou, Rice Diseases, 2nd ed. Kew, Surrey: Commonwealth Mycological Institute, 1985.

[8] E. A. Heinrichs, Biology and Management of Rice Insects. Los Baños, Philippines: International Rice Research Institute, 1994.

[9] T. W. Mew, “Bacterial Blight of Rice,” in Plant Pathology in Asia, H. S. Chaube, Ed. New Delhi: Oxford & IBH Publishing Co., 1993, pp. 123–140.

[10] Z. Liu, L. Zhang, and Y. Yang, “Deep Learning for Plant Disease Detection,” Computers and Electronics in Agriculture, vol. 175, pp. 105–115, Aug. 2020.

[11] Keras Team, “Keras: The Python Deep Learning API,” 2023. Accessed: May 1, 2025. [Online]. Available: https://keras.io/

[12] F. Pedregosa et al., “Scikit-learn: Machine Learning in Python,” Journal of Machine Learning Research, vol. 12, pp. 2825–2830, Oct. 2011.

[13] G. Bradski and A. Kaehler, “OpenCV,” Dr. Dobb’s Journal of Software Tools, vol. 25, no. 11, pp. 120–126, Nov. 2000.

[14] Google, “Google Drive: Cloud Storage for Work and Home,” 2023. Accessed: May 1, 2025. [Online]. Available: https://www.google.com/drive/

[15] Google, “Google Colaboratory: Write and Execute Python Code in Your Browser,” 2023. Accessed: May 1, 2025. [Online]. Available: https://colab.research.google.com/

[16] S. J. Pan and Q. Yang, “A Survey on Transfer Learning,” IEEE Transactions on Knowledge and Data Engineering, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[17] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How Transferable Are Features in Deep Neural Networks?” in Advances in Neural Information Processing Systems (NIPS), vol. 27, pp. 3320–3328, 2014.

[18] vMixGPT, “Vanishing Gradient và Exploding Gradient là gì? Làm sao để ngăn chặn chúng,” 2025. Accessed: May 30, 2025. [Online]. Available: https://vmixgpt.com/vanishing-gradient-va-exploding-gradient-la-gi-lam-sao-e-ngan-chan-chung/

[19] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778, 2016. [Online]. Available: https://www.cv-foundation.org/openaccess/content\_cvpr\_2016/papers/He\_Deep\_Residual\_Learning\_CVPR\_2016\_paper.pdf

[20] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted Residuals and Linear Bottlenecks,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4510–4520, 2018. [Online].

[21] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the Inception Architecture for Computer Vision,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2818–2826, 2016. [Online].

[22] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, “Densely Connected Convolutional Networks,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2261–2269, 2017. [Online].

[23] L. Breiman, “Random Forests,” Machine Learning, vol. 45, no. 1, pp. 5–32, 2001. [Online]. Available: https://link.springer.com/article/10.1023/A:1010933404324

[24] T. M. Cover and P. E. Hart, “Nearest Neighbor Pattern Classification,” IEEE Transactions on Information Theory, vol. 13, no. 1, pp. 21–27, 1967. [Online]. Available: https://ieeexplore.ieee.org/document/1053964

[25] C. Cortes and V. Vapnik, “Support-Vector Networks,” Machine Learning, vol. 20, no. 3, pp. 273–297, 1995. [Online]. Available: https://link.springer.com/article/10.1007/BF00994018

# PHỤ LỤC

1. GitHub Repository: Notebook Colab cho tiểu luận. Truy cập tại: https://github.com/Kiwi-0102/tieuLuan [↑](#footnote-ref-1)