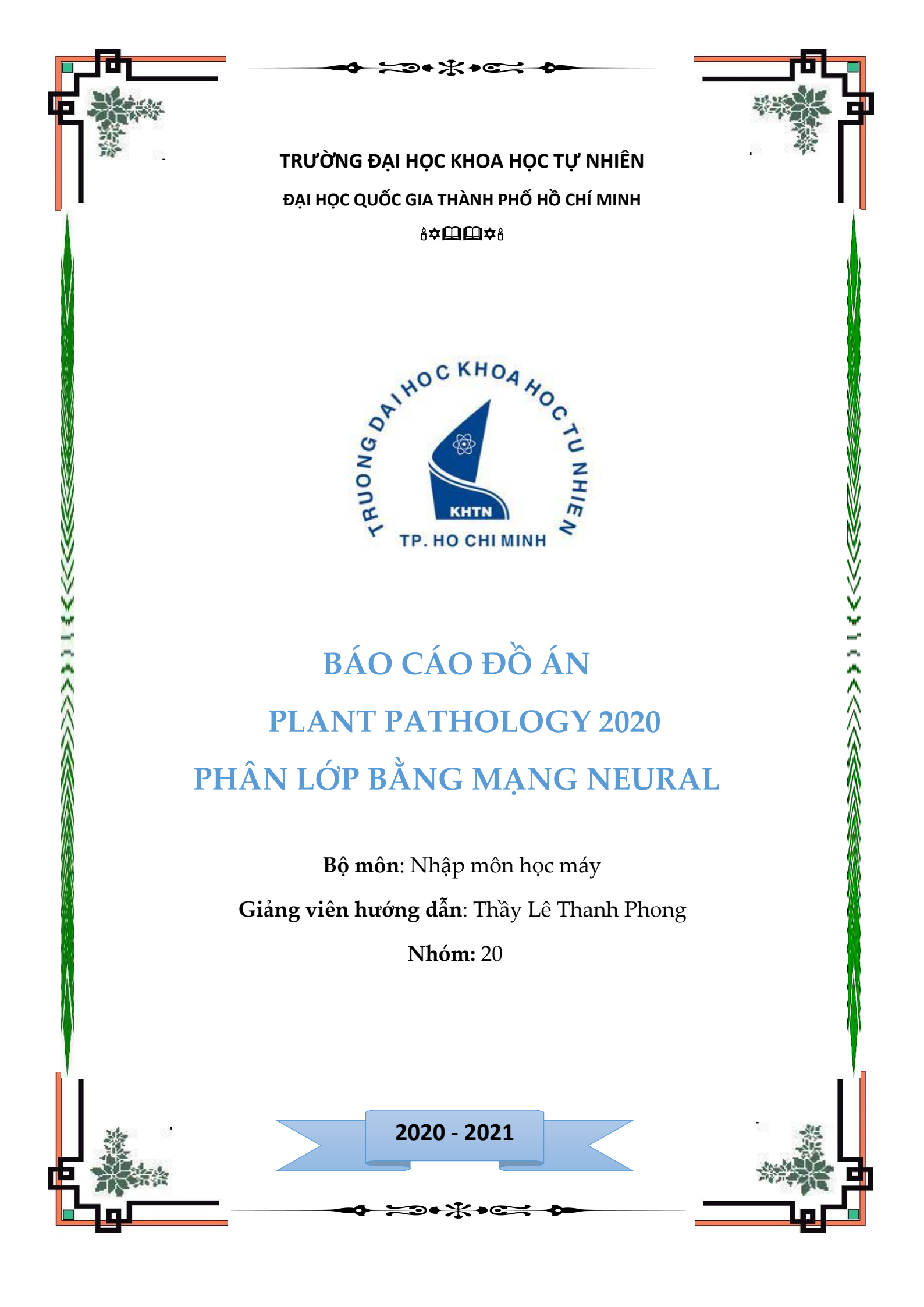
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH



# BÁO CÁO ĐỒ ÁN

# PLANT PATHOLOGY 2020

# PHÂN LỚP BẰNG MẠNG NEURAL

**Bộ môn**: Nhập môn học máy

**Giảng viên hướng dẫn**: Thầy Lê Thanh Phong

**Nhóm:** 20

**2020 - 2021**

# MỤC LỤC

## A. THÔNG TIN SINH VIÊN VÀ ĐÁNH GIÁ MỨC ĐỘ HOÀN THÀNH

| MSSV | Họ và tên | Tự đánh giá | Mức độ đóng góp |
|---|---|---|---|
| **18120492** | Du Chí Nhân | 100% | 100% |
| **18120496** | Lê Hoàng Phương Nhi | 100% | 100% |
| **18120529** | Phan Văn Võ Quyền | 100% | 100% |
| **18120530** | Lê Thị Như Quỳnh | 100% | 100% |
| **18120540** | Phạm Minh Sỹ | 100% | 100% |

## B. BÁO CÁO CHI TIẾT

### I. PHÂN TÍCH BÀI TOÁN VÀ CHỌN LỰA MÔ HÌNH NEURAL

#### 1. Phân tích bài toán

##### a. Vấn đề thực tiễn

- Các vườn cây luôn bị đe dọa bởi một số lượng lớn mầm bệnh và côn trùng. Việc triển khai quản lý bệnh phù hợp và kịp thời phụ thuộc vào việc phát hiện bệnh sớm. Việc chẩn đoán không chính xác và chậm trễ có thể dẫn đến việc sử dụng quá nhiều hoặc không đủ hóa chất, làm tăng chi phí sản xuất và tăng tác động đến môi trường và sức khỏe
- Ngành công nghiệp cây trồng, cây ăn quả hằng năm trị giá hàng tỷ đô la, chịu thiệt hại hàng triệu đô la do các áp lực sinh học và phi sinh học khác nhau, quản lý căng thẳng, liên tục và các tác động kéo dài nhiều năm từ việc rụng trái của những cây ăn quả. Qua mùa sinh trưởng, các vườn trái luôn bị đe dọa bởi một số lượng lớn côn trùng, cũng như nấm, vi khuẩn và vi rút gây bệnh. Tùy thuộc vào tỷ lệ và mức độ nghiêm trọng của sự lây nhiễm bệnh và côn trùng, các tác động từ ngoại hình, khả năng bán ra thị trường thấp và chất lượng trái cây kém, đến giảm năng suất hoặc rụng hoàn toàn trái hoặc cây, gây thiệt hại lớn về kinh tế. Việc phát hiện sớm sâu bệnh là rất quan trọng để triển khai các chương trình quản lý dịch bệnh và dịch hại phù hợp và kịp thời. Các mô hình dự báo rủi ro dịch bệnh, dịch hại và các chương trình quản lý được phát triển dựa trên tỷ lệ mắc bệnh, mức độ nghiêm trọng và thời gian lây nhiễm, có tính đến dữ liệu thời tiết hiện tại và dự báo. Việc chẩn đoán và điều trị không chính xác và / hoặc chậm trễ có thể dẫn đến sự lây lan nhanh chóng của dịch bệnh, và ngay cả những trường hợp côn trùng gây hại hay

bệnh tật nhỏ cũng có thể nhanh chóng trở thành một vấn đề lớn hơn và tốn kém hơn khi mầm bệnh nhân lên nhanh chóng, đặc biệt là trong điều kiện môi trường thuận lợi. Ngoài ra, việc chẩn đoán sai có thể dẫn đến việc sử dụng quá nhiều hoặc quá ít hóa chất, dẫn đến sự xuất hiện của các chủng mầm bệnh kháng thuốc, tăng chi phí sản xuất, tăng tác động đến môi trường và sức khỏe, hoặc có khả năng bùng phát dịch bệnh đáng kể. Các vườn trái cây mật độ cao hiện đại, thường được tạo thành từ một số giống cây trồng rất nhạy cảm, đặc biệt dễ bị lây lan mầm bệnh nhanh chóng, có khả năng giết chết toàn bộ vườn

- Hiện tại, việc phát hiện bệnh và sâu bệnh trong các vườn trái cây thương mại chỉ dựa vào việc dò tìm thủ công của các chuyên gia, nhà tư vấn cây trồng và nhà cung cấp dịch vụ. Thật không may, rất ít người có sẵn kinh nghiệm, buộc họ phải bao quát nhiều vườn cây ăn trái lớn trong một khung thời gian hẹp. Các thực tập sinh đòi hỏi rất nhiều chuyên môn và đào tạo trước khi họ có thể chẩn đoán hiệu quả và chính xác một vườn cây ăn quả. Nói chung, họ được huấn luyện đầu tiên bằng cách sử dụng hình ảnh của các triệu chứng bệnh và thiệt hại do côn trùng gây ra, nhưng do sự hiện diện của rất nhiều biến số trong một vườn thực tế, họ cần thời gian đáng kể để làm quen với nhiều loại triệu chứng gây ra bởi cả tuổi và loại của các mô bị nhiễm bệnh hoặc giai đoạn của bệnh hoặc chu kỳ dịch hại, cũng như do thời tiết thay đổi, sự khác biệt về địa lý và sự khác biệt về văn hóa. Các chuyên gia quan sát có kinh nghiệm cũng thường thiết lập một mô hình lấy mẫu ngẫu nhiên để tránh đánh giá trực quan mọi cây, đặc biệt là trong các vườn cây ăn quả lớn, nơi các khu vực phải được quan sát một cách chiến lược để bao quát những cây quan trọng nhất. Các thực tập sinh sẽ tìm kiếm những giống cây trồng mẫn cảm cụ thể hoặc các vùng cụ thể của vườn cây ăn quả. Nhiều triệu chứng của bệnh, sâu bệnh và căng thẳng phi sinh học trong vườn cây đủ rõ ràng để phân biệt chỉ dựa trên các triệu chứng hình ảnh. Tuy nhiên, một số triệu chứng bệnh trông giống nhau đến mức khó xác định chính xác nguyên nhân của chúng. Đồng thời, các triệu chứng trực quan của một loại bệnh hoặc côn trùng cụ thể có thể khác nhau rất nhiều giữa các giống táo, do sự khác biệt về màu lá, hình thái và sinh lý. Nhiệt độ, độ ẩm cụ thể và giai đoạn phát triển sinh lý của cây cũng đóng một vai trò quan trọng trong việc lây nhiễm bệnh tật và sự phát triển của côn trùng. Hình dạng và hình thức của các triệu chứng cũng có thể thay đổi theo thời gian khi bệnh tiến triển và mô lá hoặc quả già đi. Ngoài thời gian ở vườn cây ăn quả, một chuyên gia quan sát dành một

lượng thời gian đáng kể cho từng khách hàng, nhập báo cáo quan sát, giải thích kết quả và đưa ra các khuyến nghị hành động (ví dụ: có cần phun thuốc hay không, dung dịch phun là gì, …). Nhìn chung, việc quan sát của con người thường tốn thời gian, tốn kém và trong một số trường hợp, dễ xảy ra sai sót.

- Các bệnh trên lá phổ biến nhất là bệnh vảy, bệnh phấn trắng, bệnh cháy lá, đốm lá Alternaria và đốm lá có vảy. Bệnh vảy, do nấm bệnh Venturia inaequalis gây ra, là một trong những bệnh nấm quan trọng nhất về kinh tế đối với táo ở các vùng ôn đới trên thế giới. Các triệu chứng điển hình của bệnh vảy nến là các cấu trúc nấm có thể nhìn thấy trên bề mặt lá và quả. Sự lây nhiễm ban đầu xuất hiện dưới dạng các vết bệnh màu đen hoặc màu nâu ôliu phồng lên trên bề mặt chính của lá; các giai đoạn sau biểu hiện các vết bệnh do nấm tạo diệp lục trên các lá bị nhiễm bệnh. Bệnh vảy nến không chỉ ảnh hưởng đến lá mà còn ảnh hưởng đến quả, và có thể gây rụng quả sớm, lá rụng và quả bị biến dạng, quả bị nhiễm bệnh có các vết bệnh màu sẫm, viền rõ, nâu và sần sùi. Bệnh gỉ sắt tuyết tùng do nấm Basidiomycotina Gymnosporangium yamadai miyabe gây ra. Các triệu chứng ban đầu của bệnh là những chấm nhỏ, màu vàng nhạt trên lá, sau đó sẽ nở ra và chuyển sang màu cam sáng. Các lá bị nhiễm bệnh sưng tấy, to ra, quăn mép; trong trường hợp nghiêm trọng, lá rụng sớm xảy ra. Sự bùng phát nghiêm trọng của mầm bệnh rỉ sắt kéo dài từ hai đến ba năm có thể làm bị thương nặng hoặc giết chết cây của các giống táo mẫn cảm. Các bệnh trên lá khác của táo, đặc biệt là bệnh đốm lá Alternaria và bệnh đốm lá mắt đỏ, có thể gây ra các triệu chứng xuất hiện tương tự khó phân biệt nếu không có chuyên gia và các chuyên gia kiểm tra cẩn thận, và nhiều nghiên cứu đã đưa ra các phương pháp để phân biệt chúng

**b. Phân tích về tập dữ liệu**

- Tập dữ liệu bao gồm 1821 hình ảnh RGB chân thực, chất lượng cao về nhiều triệu chứng bệnh trên lá trong mùa sinh trưởng. Các bức ảnh được chụp trong các điều kiện ánh sáng, góc chụp, bề mặt và độ nhiễu khác nhau. Sự phức tạp ở tập dữ liệu còn tăng lên khi bao gồm các hình ảnh không cân đối về các loại bệnh khác nhau, nền hình ảnh không đồng nhất, hình ảnh được chụp vào các thời điểm khác nhau trong ngày, hình ảnh từ các thời điểm khác nhau trong giai đoạn sinh trưởng của cây, hình ảnh hiển thị nhiều bệnh, hình ảnh được chụp bằng các cách lấy nét khác nhau. Các hình ảnh được chụp là ảnh các lá khỏe mạnh, hoặc bị bệnh vảy, hoặc bệnh gỉ sắt hay kết hợp nhiều loại bệnh

- Tập dữ liệu được chia ngẫu nhiên thành tập huấn luyện (80%) và tập thử nghiệm (20%), đảm bảo rằng tất cả bốn loại bệnh đều được đại diện trong cả hai tập dữ liệu. Trong đó tập dữ liệu train đã được các nhà nghiên cứu thực vật học chuyên nghiệp xác nhận các loại bệnh, cũng như chú thích thêm, đặc biệt ở những hình ảnh khó phân biệt về mặt triệu chứng hay nhiều triệu chứng phức tạp cũng xuất hiện trên một lá. Cụ thể ở tập train có 592 hình ảnh cho bệnh vảy, 622 hình ảnh cho bệnh gỉ sắt, 91 hình ảnh cho các lá mang nhiều bệnh và 516 hình ảnh cho các lá khỏe mạnh

**Ví dụ về hình ảnh cho lá khỏe mạnh (Train_2)**

Ví dụ về hình ảnh cho lá bị bệnh gỉ sắt (Train_0)

Ví dụ về hình ảnh cho lá bị nhiều triệu chứng bệnh khác nhau (Train_1)

Ví dụ về hình ảnh cho lá bị bệnh vảy (Train_6)

## 2. Ánh xạ vấn đề thực tiễn qua bài toán cần giải quyết

### a. Vì sao nên lựa chọn mô hình máy học?

- Xuất phát từ việc quan sát mẫu lá bị bệnh của con người luôn tốn thời gian, công sức, chi phí, cũng như tồn đọng sai sót, trong những năm gần đây, hình ảnh kỹ thuật số và máy học đã phát triển vượt bậc, và cho thấy tiềm năng to lớn để tăng tốc độ chẩn đoán bệnh cây trồng. Cuộc cách mạng hình ảnh kỹ thuật số đã tạo ra những cơ hội to lớn trong nhiều lĩnh vực của đời sống xã hội và nghề nghiệp, và phần lớn thế giới đã sẵn sàng tiếp cận với điện thoại thông minh có tích hợp máy ảnh kỹ thuật số có thể được sử dụng để chụp những hình ảnh chất lượng cao về các triệu chứng bệnh. Các phương pháp thị giác máy tính đang được phát triển để sử dụng hình ảnh kỹ thuật số của các triệu chứng để phân loại bệnh. Các phương pháp này kết hợp chuyên môn của con người và các thuật toán học máy để tìm các mối quan hệ và các mẫu trực quan để phân nhóm và nhận dạng. Tập train được cung cấp thỏa mãn yếu tố, một là hình ảnh kĩ thuật số chất lượng cao, đảm bảo cho việc thực hiện các phương pháp thị giác máy tính và hai là các hình ảnh đã được các chuyên gia chú thích về mẫu bệnh trên từng lá, tạo điều kiện dễ dàng cho việc đưa vào các mô hình học máy để đào tạo. Sau khi các mô hình đã được đào tạo, các

hình ảnh không xác định có thể được tự động xác định bằng cách sử dụng các mô hình này.

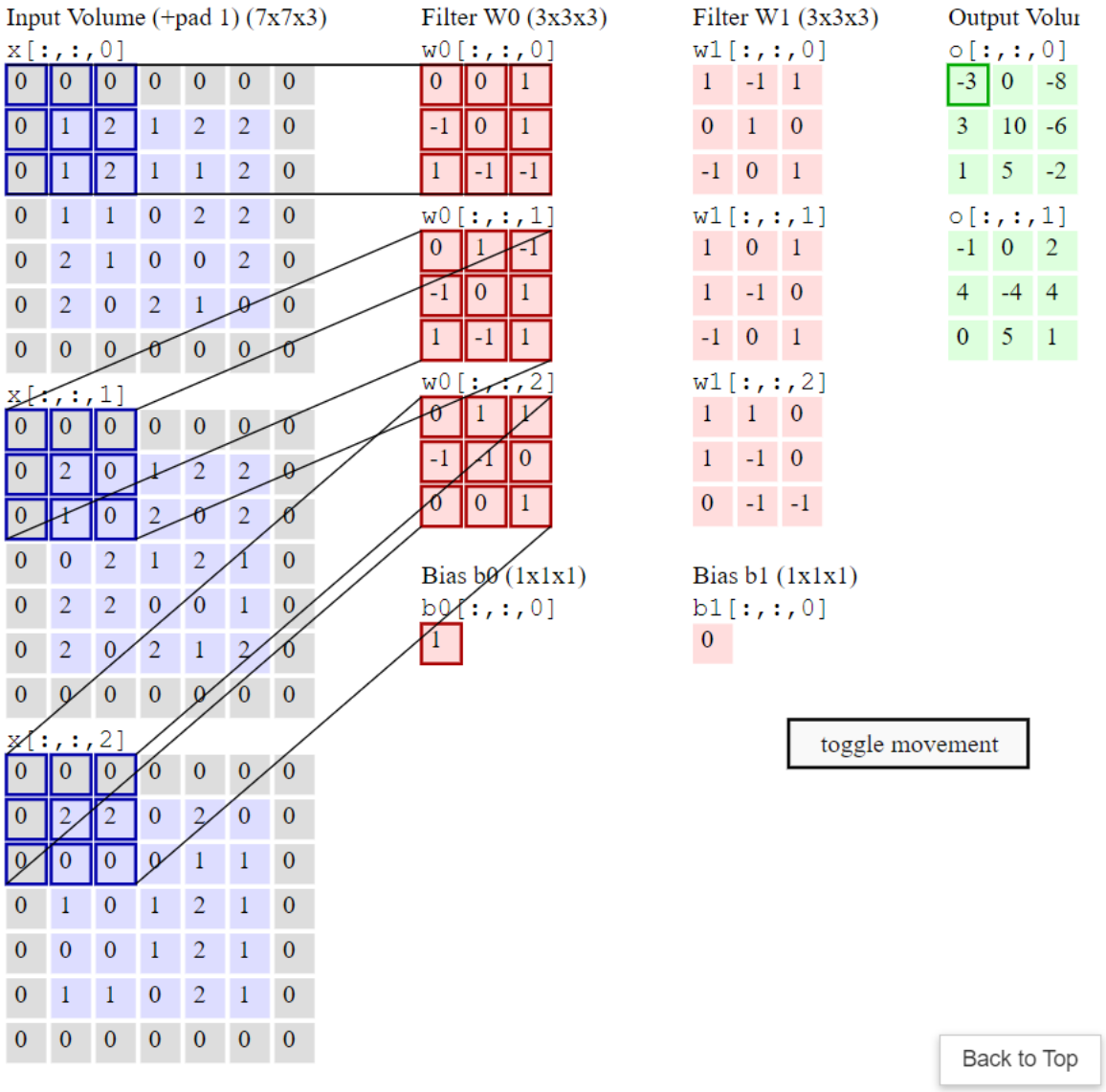
- Khi sử dụng thị giác máy tính như một công cụ để xác định bệnh chính xác, tất cả các biến số triệu chứng tiềm ẩn phải được tính toán trong cơ sở dữ liệu hình ảnh kỹ thuật số. Ví dụ, các điều kiện chụp ảnh cho một cái cây phải bao gồm nhiều vị trí và góc độ của mô bị nhiễm bệnh, nhiều mức độ ánh sáng xung quanh, nhiều loại cảm biến chụp khác nhau và các kết quả theo mùa và thời tiết khác nhau, ngoài việc minh họa ảnh hưởng của từng loại bệnh khác nhau. trên quả và lá ở nhiều độ tuổi khác nhau. Điều này khiến cho các mô hình thị giác máy tính dựa trên triệu chứng khó có thể tự động phân loại các triệu chứng bệnh với độ chính xác cao mà không cần sự giám sát của chuyên gia bệnh học thực vật. Một số lượng lớn hình ảnh thực chất lượng cao về các triệu chứng bệnh tật và côn trùng có thể được thu thập và được chuyên gia chú thích để đào tạo các mô hình thị giác máy tính với độ chính xác dự đoán cao, do đó mà việc lựa chọn các mô hình học máy mạng nơ-ron phức hợp tỏ ra khả quan hơn. Ở đây, nhóm quyết định lựa chọn mô hình học sâu CNN để tiến hành phân loại bệnh phẩm dựa trên hình ảnh về lá bị bệnh trong thời kì sinh trưởng của cây

**b. Khái quát về mô hình CNN**

- Convolutional Neural Network (CNNs – Mạng nơ-ron tích chập) là một trong những mô hình Deep Learning tiên tiến giúp chúng ta xây dựng được những hệ thống thông minh với độ chính xác cao. CNN được sử dụng nhiều trong các bài toán nhận dạng hình ảnh. Kiến trúc mạng CNN xuất hiện do các phương pháp xử lý dữ liệu ảnh thường sử dụng giá trị của từng pixel. Vì vậy nếu ta có một ảnh kích thước 100*100, khi sử dụng kênh RGB ta có tổng cộng $100*100*3 = 30\ 000$ nút ở lớp đầu vào. Điều đó kéo theo việc có một số lượng lớn weight và bias, dẫn đến mạng nơ-ron trở nên quá đồ sộ, gây khó khăn trong việc tính toán. Thêm vào đó, chúng ta nhận ra rằng thông tin của các pixel thường chỉ chịu tác động bởi các pixel ngay gần nó, vậy nên việc bỏ qua một số nút ở tầng đầu vào trong mỗi lần huấn luyện sẽ không làm giảm độ chính xác của mô hình. Vậy nên người ta sử dụng cửa sổ tích chập nhằm giải quyết vấn đề về số lượng tham số lớn mà vẫn trích xuất được đặc trưng của ảnh. Về mặt kĩ thuật, trong mô hình học sâu CNN, mô hình ảnh đầu vào sẽ chuyển nó qua một loạt các lớp tích chập với các bộ lọc, sau đó đến lớp Pooling,

rồi tiếp theo là các lớp FC – fully connected layers và cuối cùng áp dụng hàm softmax để phân loại đối tượng dựa trên giá trị xác suất trong khoảng (0;1)

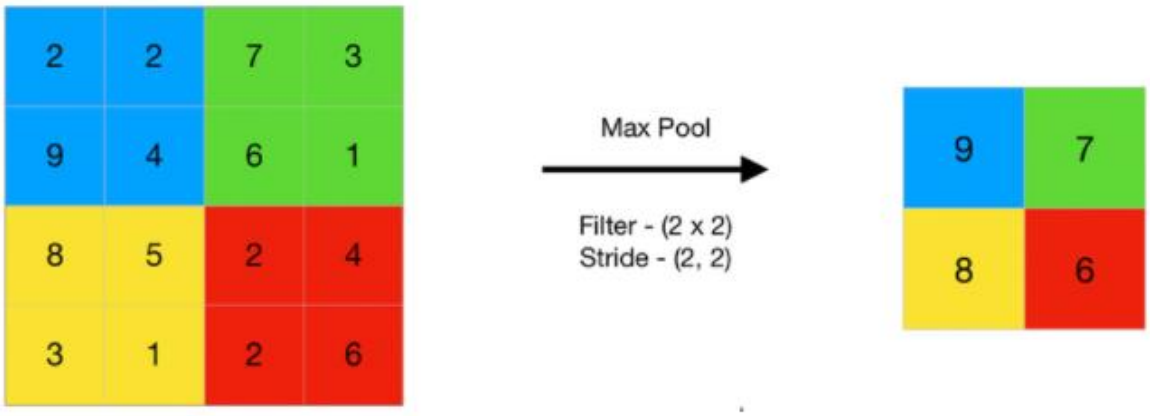
- Convolution Layer: là lớp đầu tiên trích xuất các đặc tính từ hình ảnh. Tham số lớp này bao gồm một tập hợp các bộ lọc có thể học được. Các bộ lọc đều nhỏ, thường có kích cỡ hai chiều đầu tiên khoảng 3*3 hoặc 5*5, ... và có độ sâu bằng với độ sâu của đầu vào. Bằng cách trượt dần bộ lọc theo chiều ngang và dọc của ảnh, chúng ta thu được một Feature Map chứa các đặc trưng được trích xuất từ hình ảnh đầu vào. Quá trình trượt các bộ lọc thường có các giá trị quy định bao gồm:
    - Padding: quy định bộ đệm của bộ lọc
    - Stride: quy định bước nhảy trong quá trình thực hiện




**Ví dụ minh họa về convolution layer (nguồn:** CS231n Convolutional Neural Networks for Visual Recognition)

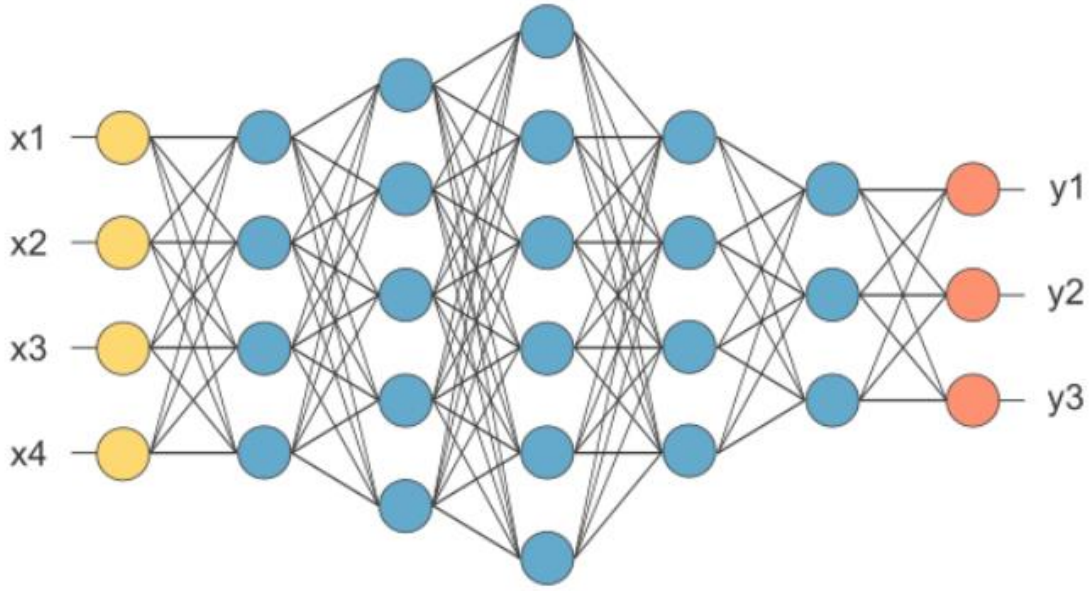
- Với mỗi kernel khác nhau, ta sẽ học được những đặc trưng khác nhau của ảnh, nên trong mỗi convolutional layer ta sẽ dùng nhiều kernel cho ra output là 1 matrix, nên k kernel sẽ cho ra k putput matrix. Ta kết hợp k output này lại thành một tensor 3 chiều, có chiều sâu k. Output này sẽ phải qua một hàm kích hoạt trước khi trở thành input của layer tiếp theo

- Pooling layer: thường được dùng giữa các convolutional layer, để giảm kích thước dữ liệu những vẫn giữ được các thuộc tính quan trọng. Kích thước dữ liệu giảm giúp giảm việc tính toán trong model. Trong quá trình này, quy tắc về stride và padding áp dụng như phép tính convolution trên ảnh




(Nguồn: CNN | Introduction to Pooling Layer - GeeksforGeeks)

- Fully connected layer: Sau khi được truyền qua nhiều convolutional layer và pooling layer thì model đã học được tương đối các đặc điểm của ảnh thì tensor của output của layer cuối cùng sẽ được là phẳng thành vector và đưa vào một lớp được kết nối như một mạng nơ-ron. Với FC layer được kết hợp các tính năng lại với nhau để tạo ra một mô hình. Cuối cùng sử dụng softmax hoặc sigmoid để phân loại đầu ra




(Nguồn: Tìm hiểu về Convolutional Neural Network và làm một ví dụ nhỏ về phân loại ảnh (viblo.asia))

- Trường tiếp nhận (Receptive field): trong mạng nơ-ron, mỗi nơ-ron nhận đầu vào từ một số vị trí trong lớp trước đó. Trong một lớp chập, mỗi nơ-ron chỉ nhận đầu vào từ một vùng hạn chế của lớp trước đó gọi là trường tiếp nhận của nơ-ron. Điển hình vùng này là một hình vuông kích thước 5*5. Trong khi trong một lớp FC, trường tiếp nhận là toàn bộ lớp phía trước đó. Do đó, trong một lớp tích chập, mỗi nơ-ron nhận đầu vào từ một vùng lớn hơn so với các lớp trước đó. Điều này là do áp dụng tích chập lặp đi lặp lại, có tính đến giá trị của một pixel, cũng như các pixel xung quanh nó
- Trọng số: mỗi nơ-ron trong mạng nơ-ron tính toán giá trị đầu vào bằng cách áp dụng một hàm cụ thể cho các giá trị nhận được từ các lớp trước đó. Hàm được áp dụng cho các giá trị đầu vào được xác định bằng vector trọng số và độ lệch. Việc học bao gồm việc điều chỉnh lặp đi lặp lại những bias và trọng số này. Vector trọng số và độ lệch bias được gọi là bộ lọc, và đại diện cho các tính năng cụ thể của đầu vào

**c. Vì sao nên lựa chọn mô hình CNN cho bài toán này?**

- Khi đưa ra một tập train, không giống với các kỹ thuật, mô hình học máy khác, CNN sẽ tối ưu hóa trọng số và các thông số lọc thông qua các lớp ẩn để tạo ra các tính năng phù hợp nhằm giải quyết vấn đề phân loại. Về nguyên tắc, các tham số trong

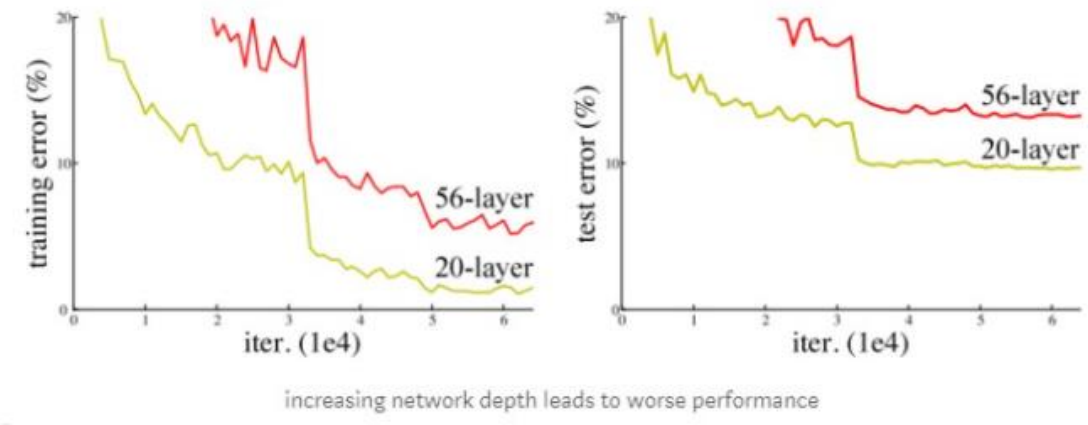
mạng được tối ưu hóa bằng cách tiếp cận lan truyền ngược và phương pháp gradient descent để cực tiểu hóa độ lỗi phân lớp

- Mô hình CNN lớn mạnh dần theo thời gian, sau phát minh của AlexNet (2012), cùng với tiến bộ trong phần cứng, kiến trúc CNN trở nên mạnh hơn bao giờ hết. Chẳng hạn VGG-19 bao gồm 19 lớp, GoogleNet (2014) hay ResNets (2015), Desnsenet (2016)

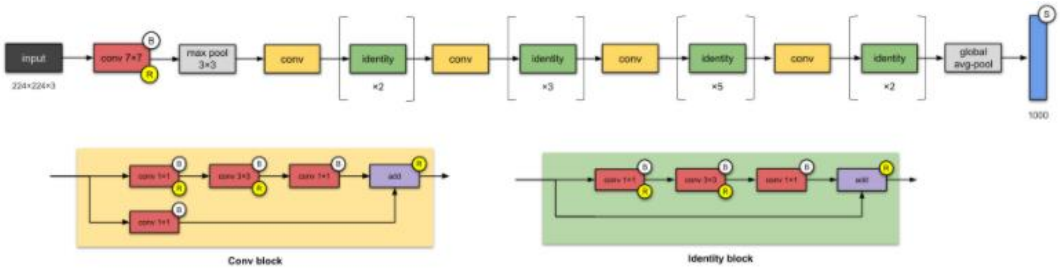
## 3. Các mô hình CNN dùng trong bài

### a. ResNet

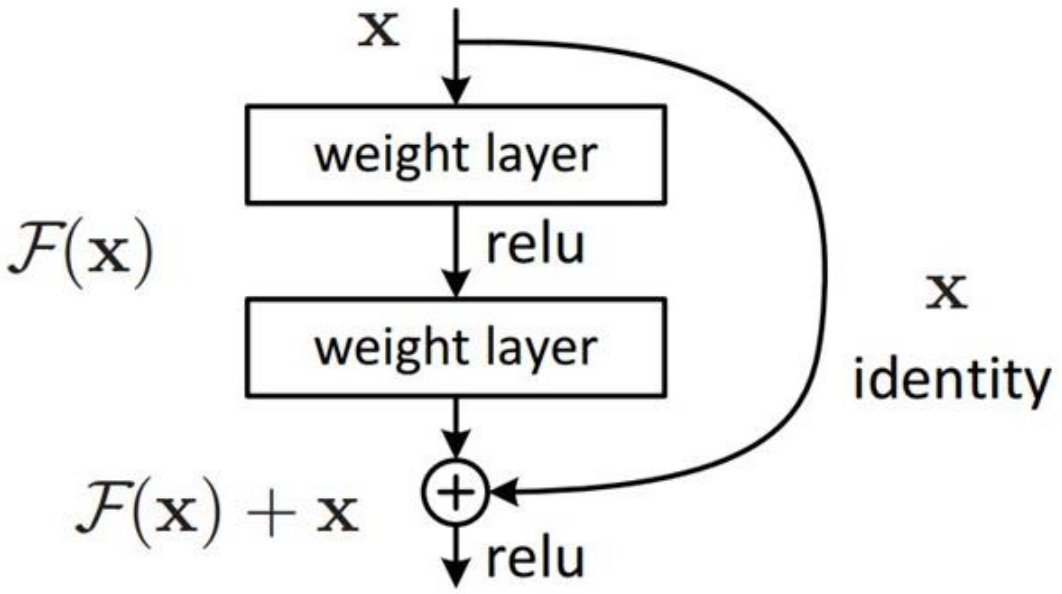
- ResNet được phát triển bởi Microsoft năm 2015 với paper "Deep residual learning for image recognition". ResNet chiến thắng trong cuộc thi ImageNet ILSVRC 2015 với error rate 3.57%, ResNet có cấu trúc gần giống VGG với nhiều stack layer làm cho model deeper hơn. Không giống VGG, ResNet có depth sâu hơn như 34, 55, 101 và 151. ResNet giải quyết được vấn đề của deep learning truyền thống, nó có thể dễ dàng training model với hàng trăm layer. Để hiểu ResNet chúng ta cần hiểu vấn đề khi stack nhiều layer khi training, vấn đề đầu tiên khi tăng model deeper hơn gradient sẽ bị vanishing/explodes. Vấn đề này có thể giải quyết bằng cách thêm Batch Normalization nó giúp normalize output giúp các hệ số trở nên cân bằng hơn không quá nhỏ hoặc quá lớn nên sẽ giúp model dễ hội tụ hơn. Vấn đề thứ 2 là degradation, khi model deeper accuracy bắt đầu bão hòa thậm chí là giảm. Như hình vẽ bên dưới khi stack nhiều layer hơn thì training error lại cao hơn ít layer như vậy vấn đề không phải là do overfitting. Vấn đề này là do model không dễ training khó học hơn, thử tưởng tượng muốn training một shallow model, sau đó chúng ta stack thêm nhiều layer, các layer sau khi thêm vào sẽ không học thêm được gì cả nên acccuracy sẽ tương tự như shallow model mà không tăng. ResNet được ra đời để giải quyết vấn đề degradation này.




- Kiến trúc ResNet bao gồm 2 khối đặc trưng là khối tích chập (Conv Block) và khối xác định (Indentity Block).



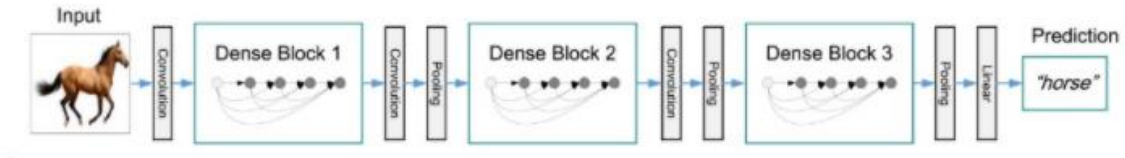

- ResNet là kiến trúc được sử dụng khá phổ biến. ResNet cũng là kiến trúc sớm nhất áp dụng Batch Normalization.
- Những kiến trúc trước đây thường cải tiến độ chính xác nhờ gia tăng chiều sâu của mạng CNN. Nhưng thực nghiệm cho thấy đến một ngưỡng độ sâu nào đó thì độ chính xác của mô hình sẽ bão hòa và thậm chí phản tác dụng và làm cho mô hình kém chính xác hơn. Khi đi qua quá nhiều tầng độ sâu có thể làm thông tin gốc bị mất đi thì các nhà nghiên cứu của Microsoft đã giải quyết vấn đề này trên ResNet bằng cách sử dụng kết nối tắt.
- Các kết nối tắt (skip connection) giúp giữ thông tin không bị mất bằng cách kết nối từ layer sớm trước đó tới layer phía sau và bỏ qua một vài layers trung gian.
- ResNet có khối tích chập sử dụng bộ lọc kích thước 3x3 giống với của InceptionNet. Khối tích chập bao gồm 2 nhánh tích chập trong đó một nhánh áp dụng tích chập 1x1 trước khi cộng trực tiếp vào nhánh còn lại.
- Khối xác định (Indentity block) thì không áp dụng tích chập 1x1 mà cộng trực tiếp giá trị của nhánh đó vào nhánh còn lại.




Giả sử có x là đầu vào của khối xác định. Ta cần ánh xạ đầu vào x thành hàm f(x). Để tìm ra ánh xạ chuẩn xác tương đương với hàm f(x) là một việc khá khó. Nhưng nếu cộng thêm ở đầu ra thành x + f(x) thì chúng ta sẽ qui về tham số hóa độ lệch, tức cần tham số hóa phần dư f(x). Tìm ánh xạ theo phần dư sẽ dễ hơn nhiều vì chỉ cần tìm giá trị f(x) sao cho nó gần bằng 0 là có thể thu được một ánh xạ chuẩn xác. Tại một khối xác định, chúng ta sẽ áp dụng một layer activation ReLU sau mỗi xen kẽ giữa những tầng trọng số.

**b. DenseNet**

- DenseNet (Dense connected convolutional network) là một trong những network mới nhất cho visual object recognition. Nó cũng gần giống ResNet nhưng có một vài điểm khác biệt. DenseNet có cấu trúc gồm các dense block và các transition layers. Được stack dense block – transition layers – dense block – transition layers như hình vẽ. Với CNN truyền thống nếu chúng ta có L layer thì sẽ có L connection, còn trong DenseNet sẽ có L(L+1)/2 connection.




- Ở Resnet người ta phân tách hàm số thành một hàm xác định và một hàm phi tuyến:

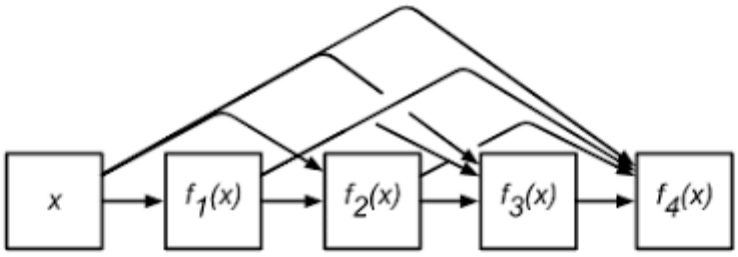
$$f(x) = x + g(x)$$

- Ta có công thức khai triển Taylor tại x = 0:

$$f(x) = f(0) + f'(x)x + \frac{f''(x)}{2!}x^2 + \cdots + \frac{f^{(n)}(x)}{n!}x^n + o(x^n)$$

Có thể thấy công thức của ResNet cũng gần tương tự như khai triển taylor tại đạo hàm bậc nhất, g(x) tương ứng với thành phần số dư. Khai triển Taylor sẽ càng chuẩn xác nếu ta phân rã được số dư thành nhiều đạo hàm bậc cao hơn.

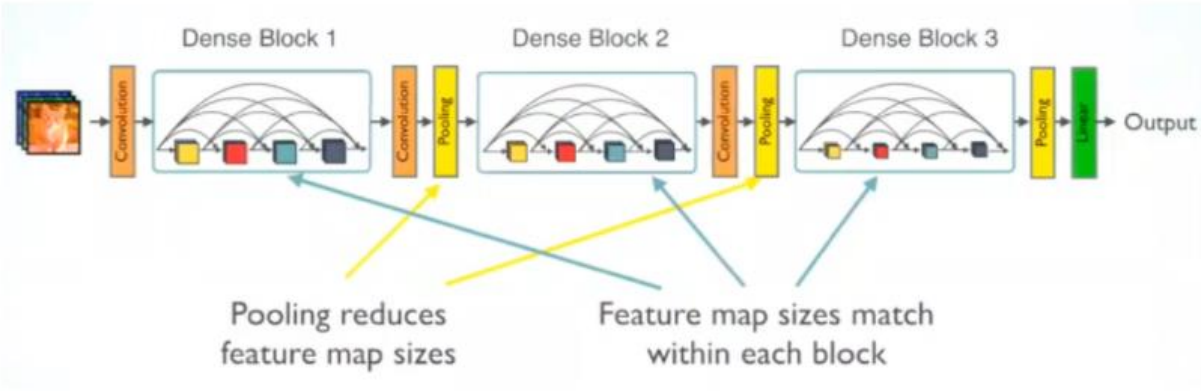
- Ý tưởng của DenseNet cũng như vậy, người ta sẽ sử dụng một mạng lưới các kết nối tắt dày đặc để liên kết các khối với nhau.




Từ đầu vào x sẽ áp dụng liên tiếp một chuỗi các ánh xạ liên tiếp với cấp độ phức tạp tăng dần:

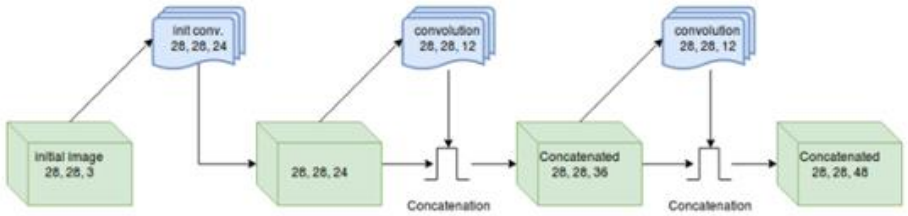
$$\mathbf{x} \to f_1(\mathbf{x})\ \mathbf{x} \to f_2(\mathbf{x}, f_1(\mathbf{x})) \ldots \mathbf{x} \to f_4(\mathbf{x}, f_3(\mathbf{x}, f_2(\mathbf{x}, f_1(\mathbf{x}))))$$

DenseNet sẽ không cộng trực tiếp x vào f(x) mà thay vào đó, các đầu ra của từng phép ánh xạ có cùng kích thước dài và rộng sẽ được nối lại với nhau thành một khối theo chiều sâu. Sau đó để giảm chiều dữ liệu sẽ áp dụng tầng chuyển tiếp (translation layer). Tầng này là kết hợp của một layer tích chập giúp giảm độ sâu và một max pooling giúp giảm kích thước dài và rộng:




- Hãy tưởng tượng ban đầu ta có 1 image size (28, 28, 3). Đầu tiên ta khởi tạo feature layer bằng Conv tạo ra 1 layer size (28, 28, 24). Sau mỗi layer tiếp theo (trong dense block) nó sẽ tạo thêm K = 12 feature giữ nguyên width và height. Khi đó

output tiếp theo sẽ là (28, 28, 24 + 12), (28, 28, 24 + 12 + 12). Ở mỗi dense block sẽ có normalization, nonlinearity và dropout. Để giảm size và depth của feature thì transition layer được đặt giữa các dense block, nó gồm Conv kernel size = 1, average pooling (2x2) với stride = 2 nó sẽ giảm output thành (14, 14, 48).



Block of convolution layers with results concatenated

- Các tham số chi tiết:

| Layers | Output Size | DenseNet-121 | DenseNet-169 | DenseNet-201 | DenseNet-264 |
|---|---|---|---|---|---|
| Convolution | 112 × 112 | 7 × 7 conv, stride 2 | | | |
| Pooling | 56 × 56 | 3 × 3 max pool, stride 2 | | | |
| Dense Block (1) | 56 × 56 | [1 × 1 conv, 3 × 3 conv] × 6 | [1 × 1 conv, 3 × 3 conv] × 6 | [1 × 1 conv, 3 × 3 conv] × 6 | [1 × 1 conv, 3 × 3 conv] × 6 |
| Transition Layer (1) | 56 × 56 | 1 × 1 conv | | | |
| | 28 × 28 | 2 × 2 average pool, stride 2 | | | |
| Dense Block (2) | 28 × 28 | [1 × 1 conv, 3 × 3 conv] × 12 | [1 × 1 conv, 3 × 3 conv] × 12 | [1 × 1 conv, 3 × 3 conv] × 12 | [1 × 1 conv, 3 × 3 conv] × 12 |
| Transition Layer (2) | 28 × 28 | 1 × 1 conv | | | |
| | 14 × 14 | 2 × 2 average pool, stride 2 | | | |
| Dense Block (3) | 14 × 14 | [1 × 1 conv, 3 × 3 conv] × 24 | [1 × 1 conv, 3 × 3 conv] × 32 | [1 × 1 conv, 3 × 3 conv] × 48 | [1 × 1 conv, 3 × 3 conv] × 64 |
| Transition Layer (3) | 14 × 14 | 1 × 1 conv | | | |
| | 7 × 7 | 2 × 2 average pool, stride 2 | | | |
| Dense Block (4) | 7 × 7 | [1 × 1 conv, 3 × 3 conv] × 16 | [1 × 1 conv, 3 × 3 conv] × 32 | [1 × 1 conv, 3 × 3 conv] × 32 | [1 × 1 conv, 3 × 3 conv] × 48 |
| Classification Layer | 1 × 1 | 7 × 7 global average pool | | | |
| | | 1000D fully-connected, softmax | | | |

- Một số ưu điểm của DenseNet:
  - ✓ Accuracy: DenseNet training tham số ít hơn 1 nửa so với ResNet nhưng có same accuracy so trên ImageNet classification dataset.
  - ✓ Overfitting: DenseNet resistance overfitting rất hiệu quả.
  - ✓ DenseNet áp dụng BatchNormalization trước khi thực hiện tích chập ở các tầng chuyển tiếp nên giảm được vashing gradient.
  - ✓ Sử dụng lại feature hiệu quả hơn.

**c. EfficientNet**

- Kiến trúc efficientNet dựa trên việc tìm kiếm tối ưu trên không gian các tham số Depth, width, channel nhằm mục đích tối ứu hóa cả độ chính xác và FLOPS. Để đạt được độ chính xác và hiệu quả tốt cho mô hình, điều quan trọng là phải cân bằng

tất cả các tham số của Depth, width, channel trong quá trình scalling quy mô convNet.

- Sử dụng hệ số kép φ để scaling đồng nhất depth, width, channel của mạng theo cách có nguyên tắc:

$$d = \alpha^{\varphi}$$

$$w = \beta^{\varphi}$$

$$r = \gamma^{\varphi}$$

$$st\ \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$$

$$\alpha \geq 1, \beta \geq 1, \gamma \geq$$

Trong đó :

- ✓ d, w, r lần lượt là depth, width, channel của mạng
- ✓ α, β, γ là các hằng số có thể được xác định bằng small grid search.
- ✓ φ là hệ số do người dùng chỉ định để kiểm soát số lượng tài nguyên khác có sẵn để scaling mô hình, trong khi α, β, γ chỉ định cách gán các tài nguyên bổ sung này cho độ rộng, độ sâu và độ phân giải của mạng tương ứng. Quan trọng, nếu dept tăng gấp đôi thì FLOPS tăng gấp đôi, width hoặc channel tăng gấp đôi thì FLOPS tăng gấp 4 lần.

## 4. Cài đặt mạng Nơron

### a. ResNet

- Sử dụng model ResNet50 với các cài đặt:
  - ✓ Pretrain model với bộ dữ liệu 'imagenet' để có được kết quả tốt hơn.
  - ✓ Thêm GlobalAveragePooling2D để chuyển output từ dạng 4D sang 2D.
  - ✓ Thêm các Dense layer:

```
x = Dense(128, activation="relu")(x)
x = Dense(64, activation="relu")(x)
```

Để làm dày đặc, sâu thêm model

  - ✓ Thêm Dense layer cho output:

```
predictions = Dense(4, activation="softmax")(x)
```

  - ✓ Compile model với optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'].
- Fit model vào training dataset với các tham số:
  - ✓ epochs = 40.
  - ✓ step_per_epoch = 11.
  - ✓ callbacks với 'ReduceLROnPlateau' để giảm learning rate khi val_loss không được cải thiện, 'ModelCheckpoint' để lưu lại mô hình và trọng số mô

hình cho tỉ lệ val_loss là thấp nhất và 'EarlyStopping' để dừng training khi val_loss không được cải thiện, giúp tránh overfitting.

**b. DenseNet**

- Sử dụng model DenseNet121 với các cài đặt:
  - ✓ Pretrain model với bộ dữ liệu 'imagenet' để có được kết quả tốt hơn.
  - ✓ Thêm GlobalAveragePooling2D để chuyển output từ dạng 4D sang 2D.
  - ✓ Thêm Dense layer cho output:

```
L.Dense(4, activation='softmax')])
```

  - ✓ Compile model với optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'].
- Fit model vào training dataset với các tham số:
  - ✓ epochs = 40.
  - ✓ step_per_epoch = 11.
  - ✓ callbacks với 'ModelCheckpoint' để lưu lại mô hình và trọng số mô hình cho tỉ lệ val_loss là thấp nhất và 'EarlyStopping' để dừng training khi val_loss không được cải thiện, giúp tránh overfitting.

**c. EfficientNet**

- Sử dụng model EfficientNetB7 với các cài đặt:
  - ✓ Pretrain model với bộ dữ liệu 'noisystudent' để có được kết quả tốt hơn.
  - ✓ Thêm Dense layer cho output:

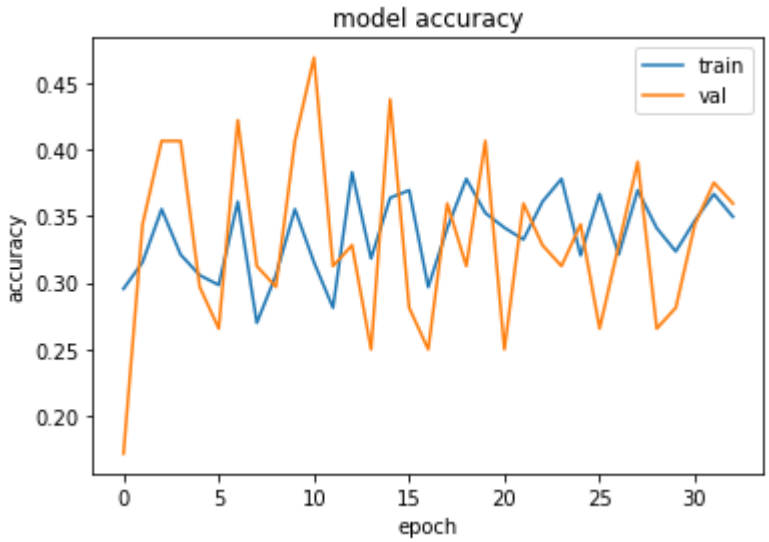
```
model.add(L.Dense(4, activation='softmax'))
```

  - ✓ Compile model với optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'] .
- Fit model vào training dataset với các tham số:
  - ✓ epochs = 40.
  - ✓ step_per_epoch = 11.
  - ✓ callbacks với 'ModelCheckpoint' để lưu lại mô hình và trọng số mô hình cho tỉ lệ val_loss là thấp nhất và 'EarlyStopping' để dừng training khi val_loss không được cải thiện, giúp tránh overfitting.

## 5. Kết quả đạt được

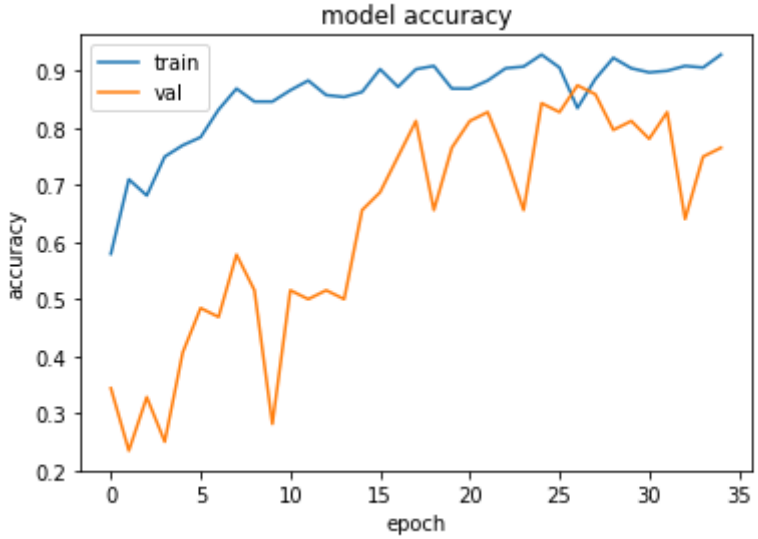
**a. Trên tập Train và Validation**

- ResNet:



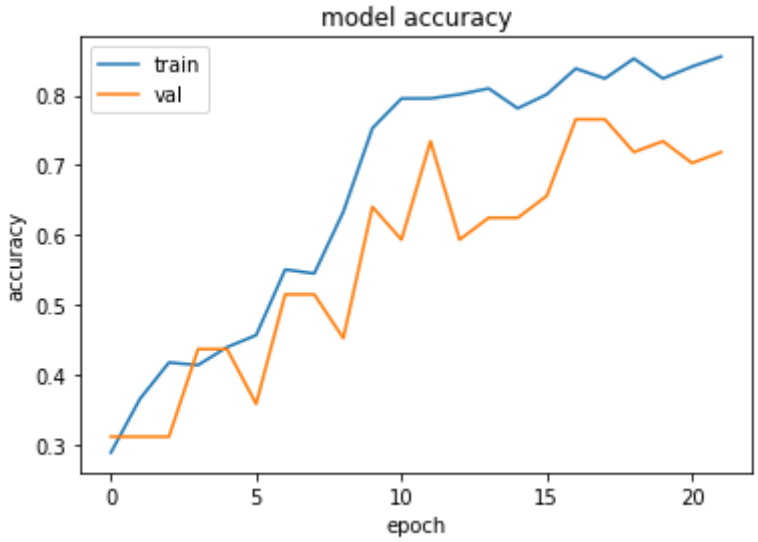

Có thể thấy accuracy trên cả tập Train và Validation đều rất hỗn loạn và rất thấp, chủ yếu ở mức 0.3 – 0.35. Và theo kết quả chạy được thì val_loss tốt nhất nhận được là 1.14739.

- DenseNet:




Có thể thấy accuracy ở cả tập Train và Validation đều được cải thiện so với mô hình ResNet. Và có vẻ ổn định trong khoảng epoch từ 25 đến 30 với accuracy ở mức 0.8 – 0.9. Theo kết quả chạy được thì val_loss tốt nhất nhận được là 0.35555.

- EfficientNet:




Có thể thấy accuracy ở cả tập Train và Validation đều được cải thiện so với mô hình ResNet nhưng thấp hơn so với DenseNet. Và có vẻ ổn định trong khoảng epoch từ 10 đến 15 với accuracy ở mức 0.7 – 0.8. Theo kết quả chạy được thì val_loss tốt nhất nhận được là 0.76227.

**b. Trên tập Test**

Từ kết quả trên tập Train và Validation thì có thể thấy DenseNet cho kết quả tốt nhất vì vậy sử dụng mô hình này để dự đoán tập Test.

Load lại trọng số model cho val_loss thấp nhất:

```
densenet.load_weights('densenet.h5')
```

Được kết quả trên tập Test:

```
test_df
```

| | image_id | healthy | multiple_diseases | rust | scab |
|---|---|---|---|---|---|
| **0** | Test_0.jpg | 2.727472e-03 | 0.766061 | 0.058882 | 1.723296e-01 |
| **1** | Test_1.jpg | 8.792978e-10 | 0.001608 | 0.998392 | 1.045499e-09 |
| **2** | Test_2.jpg | 5.238519e-05 | 0.305216 | 0.690926 | 3.805607e-03 |
| **3** | Test_3.jpg | 1.018201e-03 | 0.063575 | 0.000527 | 9.348801e-01 |
| **4** | Test_4.jpg | 5.942808e-02 | 0.009689 | 0.930551 | 3.319252e-04 |
| **...** | ... | ... | ... | ... | ... |
| **1816** | Test_1816.jpg | 6.876115e-01 | 0.134234 | 0.027363 | 1.507924e-01 |
| **1817** | Test_1817.jpg | 3.336060e-03 | 0.074003 | 0.001342 | 9.213194e-01 |
| **1818** | Test_1818.jpg | 3.312915e-04 | 0.226591 | 0.770005 | 3.072104e-03 |
| **1819** | Test_1819.jpg | 1.263853e-06 | 0.228837 | 0.000108 | 7.710544e-01 |
| **1820** | Test_1820.jpg | 5.361857e-10 | 0.000543 | 0.999456 | 2.786364e-09 |

1821 rows × 5 columns

## C. TÀI LIỆU THAM KHẢO


https://www.kaggle.com/c/plant-pathology-2020-fgvc7

Quá trình phát triển của CNN từ LeNet đến DenseNet. (dlapplications.github.io)

CS231n Convolutional Neural Networks for Visual Recognition

What is the relation between Logistic Regression and Neural Networks and when to use which? (sebastianraschka.com)

Tìm hiểu về Convolutional Neural Network và làm một ví dụ nhỏ về phân loại ảnh (viblo.asia)

Understanding Convolutional Neural Networks for NLP – WildML

Sử dụng CNN trong bài toán nhận dạng mặt người ( Phần 1 ) (viblo.asia)

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7526434/

https://spj.sciencemag.org/journals/plantphenomics/2019/9237136/

https://en.wikipedia.org/wiki/Convolutional_neural_network

https://www.kaggle.com/tarunpaparaju/plant-pathology-2020-eda-models#Modeling-

https://phamdinhkhanh.github.io/2020/05/31/CNNHistory.html

https://phamdinhkhanh.github.io/2020/04/15/TransferLearning.html