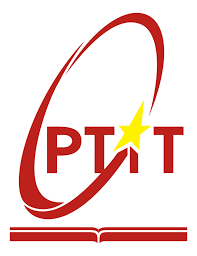
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN

**---------------------- 🙠🕮🙢 ----------------------**



BÁO CÁO MÔN HỌC

***Môn:* XỬ LÍ ẢNH**

**Nhóm báo cáo: DIPFL23PRJG30**

**Đề tài: Disease on plants**

**Thành viên nhóm:**

Vũ Hoàng Anh - B20DCCN079

Phạm Văn Hoàng - B20DCCN282

Hà Nội năm 2022

**Hà Nội, Tháng 12/2016**

Diseases classification on crops based on images of the plants

***Đặt vấn đề:***

Các bệnh trên cây trồng đặt ra những đe dọa đáng kể đối với năng suất nông nghiệp và an ninh thực phẩm trên toàn cầu. Việc xác định bệnh cây một cách kịp thời và chính xác là quan trọng để triển khai các chiến lược quản lý hiệu quả và giảm thiểu các mất mát tiềm ẩn. Trong những năm gần đây, sự tiến bộ trong các công nghệ thị giác máy tính và học sâu đã mở ra những cơ hội mới cho việc phân loại bệnh trên cây dựa trên hình ảnh của các cây trồng.

Nghiên cứu này tập trung vào việc phát triển một hệ thống tự động để phân loại bệnh trên cây sử dụng phân tích hình ảnh. Hệ thống được đề xuất tận dụng sức mạnh của mạng nơ-ron tích chập (CNN) để trích xuất các mẫu và đặc trưng phức tạp từ hình ảnh cây trồng, giúp nhận diện bệnh một cách chính xác. Các kiến trúc CNN hiện đại như ResNet, Inception, hoặc VGG được sử dụng và được điều chỉnh để đạt được hiệu suất phân loại bệnh tối ưu.

Bộ dữ liệu được sử dụng để huấn luyện và đánh giá bao gồm một bộ sưu tập đa dạng các hình ảnh cây trồng được gán nhãn, đại diện cho các điều kiện bệnh khác nhau. Hệ thống trải qua các thử nghiệm nghiêm ngặt để đảm bảo tính mạnh mẽ và độ chính xác của nó trong các tình huống thực tế. Kết quả của các thí nghiệm cho thấy hiệu suất của hệ thống trong việc nhận diện và phân loại chính xác các bệnh trên cây, từ đó hỗ trợ nông dân và chuyên gia nông nghiệp trong việc đưa ra quyết định có thông tin. Bằng cách cho phép phát hiện và can thiệp sớm, công nghệ này có khả năng giảm thiểu đáng kể mất mát cây trồng và nâng cao tổng năng suất nông nghiệp.

1. ***Giới thiệu***

Trong lĩnh vực nông nghiệp hiện đại, đảm bảo sức khỏe và năng suất của cây trồng là vô cùng quan trọng. Tuy nhiên, cây trồng trên toàn cầu đối mặt với những đe dọa liên tục từ các loại bệnh, sâu bệnh, và những tác nhân căng thẳng môi trường khác nhau. Việc phát hiện kịp thời và phân loại chính xác các bệnh là quan trọng để giảm thiểu ảnh hưởng của chúng đối với năng suất nông nghiệp và an ninh thực phẩm. Với sự tiến bộ nhanh chóng của các kỹ thuật thị giác máy tính và học sâu, phân loại bệnh dựa trên hình ảnh đã trở thành một giải pháp hứa hẹn để giải quyết thách thức nông nghiệp quan trọng này.

Một trong những kiến trúc học sâu mạnh mẽ và phổ biến nhất trong các nhiệm vụ nhận diện hình ảnh là ResNet (viết tắt của Residual Neural Network). Được phát triển bởi Kaiming He và đồng nghiệp vào năm 2015, ResNet đã làm thay đổi lĩnh vực thị giác máy tính bằng cách cải thiện đáng kể quá trình huấn luyện và cho phép xây dựng các mạng nơ-ron cực kỳ sâu. Khả năng giải quyết vấn đề giảm gradient và làm cho quá trình huấn luyện mạng nơ-ron sâu trở nên dễ dàng đã khiến cho ResNet trở thành lựa chọn ưa thích cho nhiều ứng dụng dựa trên hình ảnh.

Trong ngữ cảnh phân loại bệnh cây trồng, việc sử dụng kiến trúc ResNet để phân tích hình ảnh cây trồng mang lại triển vọng lớn. Bằng cách huấn luyện mô hình trên một bộ dữ liệu đa dạng với hình ảnh của cây trồng khỏe mạnh và bị bệnh, mạng nơ-ron có thể học cách phân biệt các mẫu và đặc trưng tinh tế chỉ ra sự xuất hiện của các loại bệnh cụ thể. Mô hình ResNet đã được huấn luyện có thể được triển khai để phân loại bệnh một cách chính xác trong thời gian thực, mang lại cho nông dân và chuyên gia nông nghiệp cái nhìn quý báu về tình trạng sức khỏe của cây trồng của họ.

Trong bài nghiên cứu này,chúng tôi xem xét lĩnh vực phân loại bệnh cây trồng bằng cách sử dụng kiến trúc ResNet. Tôi khám phá quá trình xây dựng một bộ dữ liệu mạnh mẽ với hình ảnh của cây trồng, thảo luận về việc triển khai và điều chỉnh tinh chỉnh kiến trúc ResNet, và phân tích hiệu suất của mô hình trên các loại bệnh khác nhau. Hơn nữa,chúng tôi đánh giá tác động tiềm năng của công nghệ này đối với nông nghiệp hiện đại, từ việc hỗ trợ phát hiện bệnh sớm đến việc tạo điều kiện cho các chiến lược điều trị chính xác và có định hướng.

# Dataset

Trong lĩnh vực nghiên cứu nông nghiệp và quản lý bệnh trên cây trồng, sự có sẵn của các bộ dữ liệu lớn, đa dạng và được gán nhãn đúng đắn là quan trọng đối với việc phát triển các mô hình học máy mạnh mẽ. Trong những năm gần đây, lĩnh vực này đã chứng kiến sự bùng nổ đáng kể trong việc sử dụng các kỹ thuật học sâu để phát hiện bệnh dựa trên hình ảnh.

Tận dụng những phương pháp tiên tiến này, các nhà nghiên cứu đã có thể đạt được độ chính xác chưa từng có trong việc nhận diện và phân loại các bệnh trên cây, từ đó hỗ trợ nông dân trong việc bảo vệ mùa màng và nâng cao năng suất nông nghiệp.

Một khía cạnh quan trọng của sự tiến bộ này nằm trong việc tạo ra và mở rộng các bộ dữ liệu chất lượng cao. Bài này tôi giới thiệu một phương pháp mới trong việc chuẩn bị bộ dữ liệu, trong đó một bộ sưu tập mở rộng của hình ảnh RGB của lá cây trồng khỏe mạnh và bị bệnh được tạo ra và tăng cường thông qua các kỹ thuật offline. Bộ dữ liệu gốc, lấy từ một kho lưu trữ Github toàn diện, bao gồm khoảng 87,000 hình ảnh được phân loại vào 38 lớp khác nhau, mỗi lớp đại diện cho một loại bệnh cụ thể của cây trồng.

Để đánh giá hiệu suất của mô hình trên dữ liệu mới và thực tế, một bộ thử nghiệm riêng biệt với 33 hình ảnh được tạo ra. Những hình ảnh này được lựa chọn cẩn thận để đại diện cho các tình huống đa dạng và các trường hợp thách thức mà mô hình có thể gặp phải trong quá trình triển khai. Bộ thử nghiệm cung cấp một chỉ số quan trọng để đánh giá khả năng tổng quát và dự đoán chính xác của mô hình đối với các bệnh cây trồng trong các cài đặt thực tế nông nghiệp.



# Resnet Introduction

# Năm 2012, tại cuộc thi phân loại LSVRC2012, AlexNet đã giành giải nhất. Sau đó, ResNet đã trở thành điều thú vị nhất xảy ra trong thế giới thị giác máy tính và học sâu.

# Do khung công việc mà ResNets đưa ra, việc huấn luyện các mạng nơ-ron cực kỳ sâu trở nên khả thi, có nghĩa là một mạng có thể chứa hàng trăm hoặc hàng nghìn tầng và vẫn đạt được hiệu suất tuyệt vời.

# Ban đầu, ResNets được áp dụng cho nhiệm vụ nhận diện hình ảnh, nhưng như đã được đề cập trong bài báo, khung công việc này cũng có thể được sử dụng cho các nhiệm vụ không phải thị giác máy tính để đạt được độ chính xác cao hơn.

## Vấn đề

Như chúng ta biết, mạng nơ-ron tích chập sâu rất xuất sắc trong việc nhận diện các đặc trưng ở mức thấp, trung bình và cao từ các hình ảnh, và việc thêm nhiều tầng thông thường mang lại độ chính xác cao hơn. Vì vậy, một câu hỏi đặt ra: liệu việc cải thiện hiệu suất mô hình có dễ dàng như việc thêm nhiều tầng?

Với câu hỏi này nảy sinh vấn đề của độ gradien biến mất/đột ngột gia tăng. Những vấn đề này đã được giải quyết đa dạng và cho phép các mạng với hàng chục tầng có thể hội tụ. Tuy nhiên, khi mạng nơ-ron sâu bắt đầu hội tụ, chúng ta lại gặp một vấn đề khác là độ chính xác bắt đầu bão hòa và sau đó giảm nhanh chóng. Điều này không phải do quá mức quá mức hóa như người ta có thể nghĩ, và thêm nhiều tầng vào một mô hình sâu thích hợp chỉ làm tăng lỗi huấn luyện.

Vấn đề này đã được khắc phục bằng cách sử dụng một mô hình nông hơn và một mô hình sâu được xây dựng với các tầng từ mô hình nông hơn, và thêm các tầng định danh vào đó. Theo đúng cách này, mô hình sâu hơn không nên tạo ra lỗi huấn luyện cao hơn so với mô hình nông hơn, vì các tầng thêm vào chỉ là các tầng định danh.

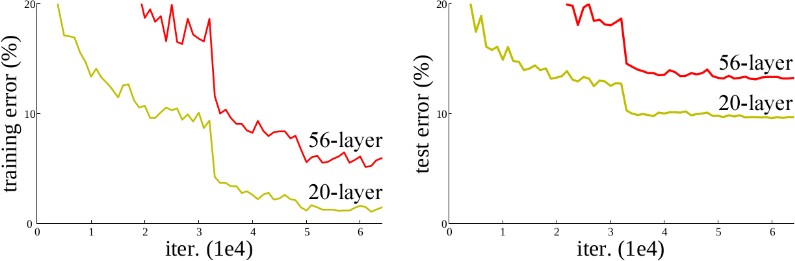


Figure 1

Trong Figure 1, chúng ta có thể thấy ở phía trái và phải rằng mô hình sâu luôn tạo ra nhiều lỗi hơn, trong khi thực tế nó không nên làm như vậy. Các tác giả đã giải quyết vấn đề này bằng cách giới thiệu khung công việc học sâu có dư thừa, trong đó họ giới thiệu các kết nối ngắn đơn giản chỉ thực hiện các ánh xạ đồng nhất.

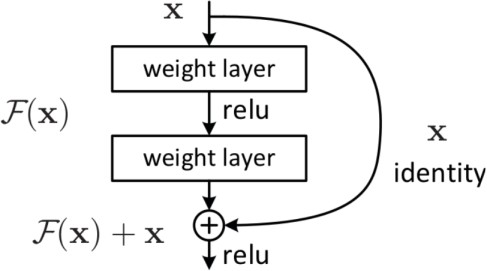


Figure 2

Họ một cách rõ ràng để các tầng vừa với một ánh xạ dư thừa và đặt tên cho nó là H(x), và họ để các tầng phi tuyến vừa với một ánh xạ khác F(x):=H(x)−x, do đó ánh xạ gốc trở thành H(x):=F(x)+x, như có thể thấy trong Figure 2

Lợi ích của việc sử dụng kết nối ngắn định danh này là không có thêm tham số được thêm vào mô hình và thời gian tính toán được kiểm soát.

## So sánh

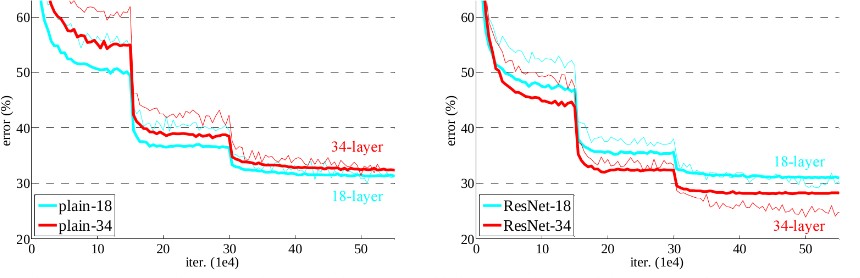


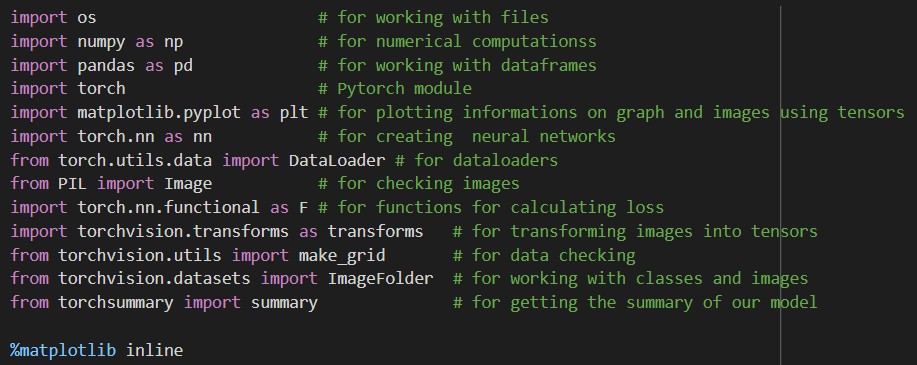
Figure 3

Để chứng minh sự ưu việt của ResNet, toi so sánh nó với một mô hình 34 tầng và một mô hình 18 tầng, cả hai đều có ánh xạ đơn và ánh xạ dư thừa. Kết quả không đem lại sự ấn tượng nhiều: mô hình 18 tầng với ánh xạ đơn vượt trội so với mô hình 34 tầng ánh xạ đơn và trong trường hợp của ResNet, mô hình ResNet 34 tầng vượt trội so với ResNet 18 tầng, như có thể thấy trong Figure 3.

## Bước Train models

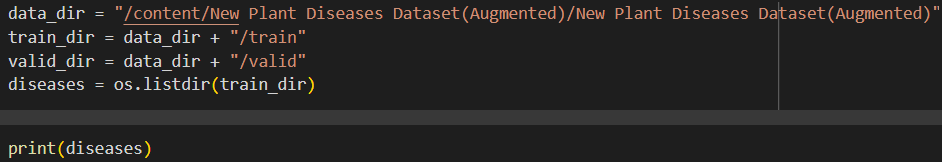
* 1. **Importing thư viện cần thiết**

Chúng ta cần torchsummary library để in model's summary ở dạng keras vị Pytourch không hỗ trỡ điều này



## Exploring the data

Tải dữ liệu vào



Số lượng tổng các loại bệnh là 38,:

'Tomato Late\_blight', 'Tomato healthy', 'Grape healthy',

'Orange Haunglongbing\_(Citrus\_greening)', 'Soybean healthy', 'Squash Powdery\_mildew', 'Potato healthy',

'Corn\_(maize) Northern\_Leaf\_Blight', 'Tomato Early\_blight', 'Tomato Septoria\_leaf\_spot', 'Corn\_(maize) Cercospora\_leaf\_spot Gray\_leaf\_spot', 'Strawberry Leaf\_scorch', 'Peach healthy',

'Apple Apple\_scab', 'Tomato Tomato\_Yellow\_Leaf\_Curl\_Virus', 'Tomato Bacterial\_spot', 'Apple Black\_rot', 'Blueberry healthy', 'Cherry\_(including\_sour) Powdery\_mildew', 'Peach Bacterial\_spot',

'Apple Cedar\_apple\_rust', 'Tomato Target\_Spot', 'Pepper,\_bell healthy', 'Grape Leaf\_blight\_(Isariopsis\_Leaf\_Spot)', 'Potato Late\_blight',

'Tomato Tomato\_mosaic\_virus', 'Strawberry healthy', 'Apple healthy', 'Grape Black\_rot', 'Potato Early\_blight', 'Cherry\_(including\_sour) healthy', 'Corn\_(maize) Common\_rust\_', 'Grape Esca\_(Black\_Measles)',

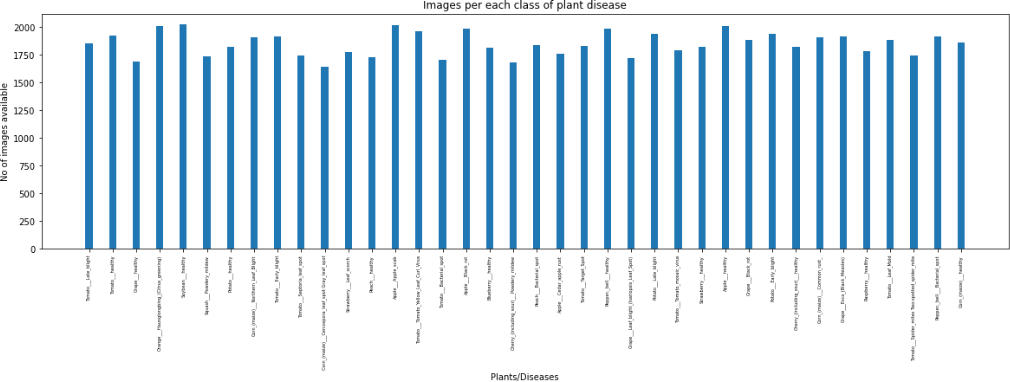
'Raspberry healthy', 'Tomato Leaf\_Mold', 'Tomato Spider\_mites Two- spotted\_spider\_mite', 'Pepper,\_bell Bacterial\_spot', 'Corn\_(maize) healthy'

Trích xuất số lượng cây trồng đặc biệt và bệnh đặc biệt

Cây trồng đặc biệt gồm:

['Tomato', 'Grape', 'Orange', 'Soybean', 'Squash', 'Potato', 'Corn\_(maize)', 'Strawberry', 'Peach', 'Apple', 'Blueberry', 'Cherry\_(including\_sour)', 'Pepper,\_bell', 'Raspberry']

Chúng ta có hình ảnh của lá cây của 14 loại cây, và khi loại bỏ lá khỏe mạnh, chúng tôi có 26 loại hình ảnh hiển thị một loại bệnh cụ thể trên một loại cây cụ thể..



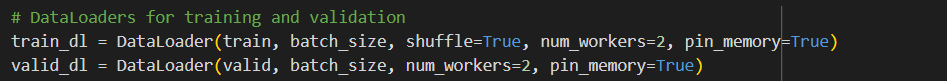
Có 70,295 hình ảnh để huấn luyện. Chúng ta có thể thấy rằng bộ dữ liệu gần như cân bằng cho tất cả các lớp, nên chúng ta có thể tiếp tục.

## Chuẩn bị dữ liệu

Tiếp theo, sau khi tải dữ liệu, chúng ta cần chuyển đổi giá trị pixel của mỗi hình ảnh (0-255) thành khoảng 0-1 vì mạng nơ-ron thường hoạt động tốt với dữ liệu được chuẩn hóa. Toàn bộ mảng giá trị pixel được chuyển đổi thành tensor torch và sau đó chia cho 255.

Chúng ta đặt giá trị 32 cho kích thước lô (batch size), đây là tổng số hình ảnh được đưa vào cùng một lần lan truyền thuận của mạng nơ-ron tích chập (CNN). Theo cách đơn giản, kích thước lô xác định số lượng mẫu sẽ được truyền qua mạng.

Ví dụ, giả sử bạn có 1050 mẫu huấn luyện và bạn muốn thiết lập kích thước lô bằng 100. Thuật toán sẽ lấy 100 mẫu đầu tiên (từ 1 đến 100) từ tập dữ liệu huấn luyện và huấn luyện mạng. Sau đó, nó lấy 100 mẫu tiếp theo (từ 101 đến 200) và huấn luyện mạng lại. Chúng ta có thể tiếp tục thủ tục này cho đến khi đã truyền tất cả các mẫu qua mạng.

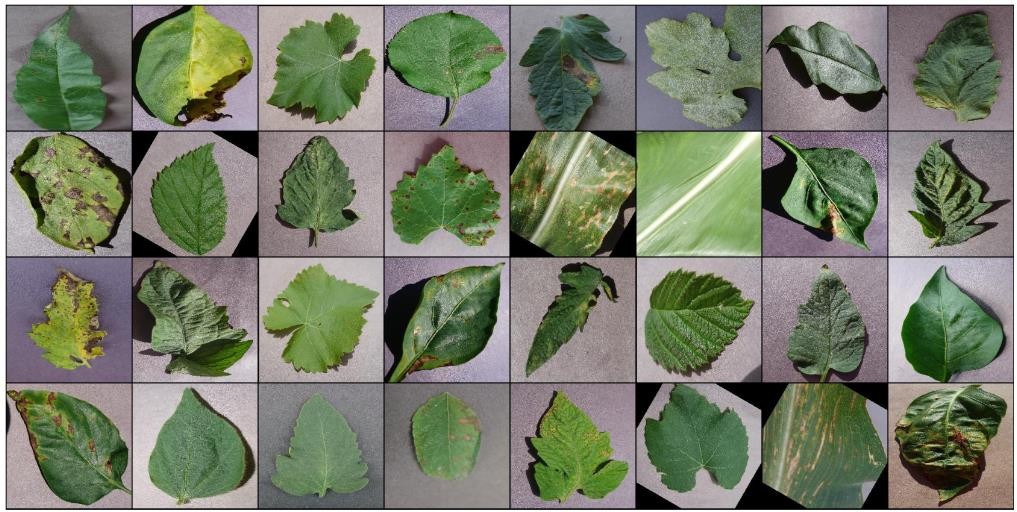


Chúng ta dùng DataLoader để train và đánh giá:

+)DataLoader là một lớp con xuất phát từ torch.utils.data. Nó hỗ trợ việc tải các bộ dữ liệu lớn và tốn bộ nhớ. Nó nhận vào tham số batch\_size, đại diện cho số lượng mẫu trong mỗi lô được tạo ra..

+)Setting shuffle=True trộn bộ dữ liệu. Điều này hữu ích để các lô giữa các kỷ nguyên không giống nhau. Việc này cuối cùng sẽ làm cho mô hình của chúng ta trở nên mạnh mẽ hơn.

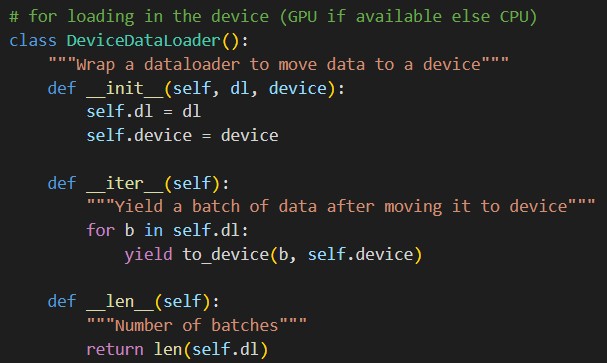
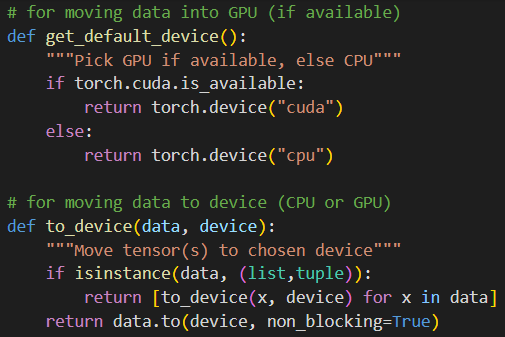
+)num\_workers, đại diện cho số tiến trình tạo ra các lô một cách song song. Nếu bạn có nhiều lõi trong CPU của mình, bạn có thể đặt nó bằng số lõi trong CPU của bạn. Vì Kaggle cung cấp một CPU 2 lõi, tôi đã đặt nó là 2.



* 1. Modelling

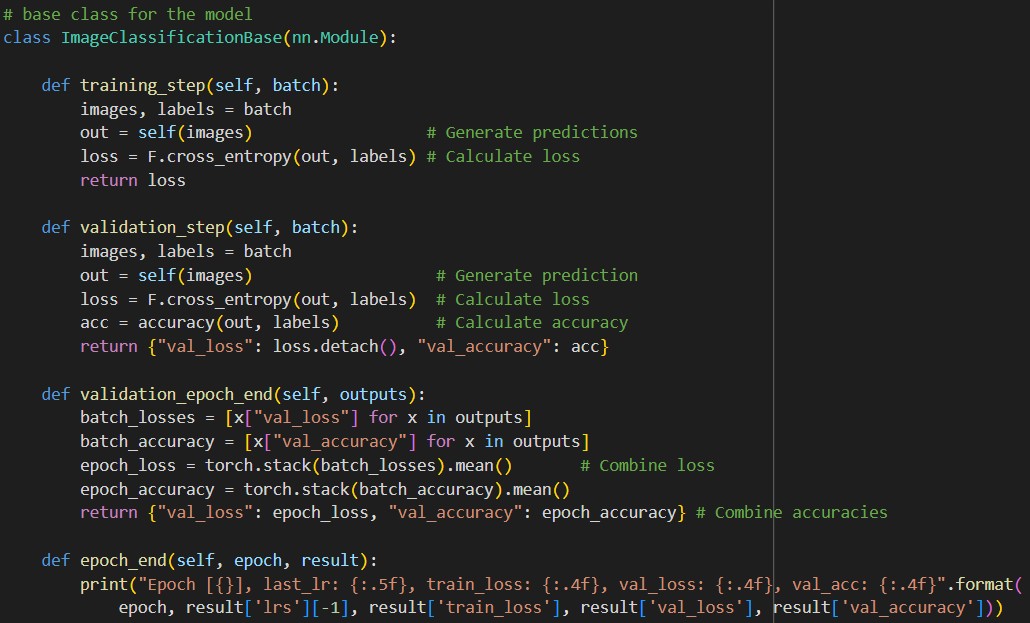
Nên sử dụng GPU thay vì CPU khi xử lý bộ dữ liệu hình ảnh, vì CPU được tổng quát hóa cho mục đích chung trong khi GPU được tối ưu hóa cho việc huấn luyện các mô hình học sâu do chúng có khả năng xử lý nhiều tính toán đồng thời.

Chúng có một số lõi lớn, cho phép tính toán tốt hơn cho nhiều quy trình song song. Ngoài ra, tính toán trong học sâu cần xử lý lượng lớn dữ liệu - điều này khiến băng thông bộ nhớ của GPU là sự lựa chọn phù hợp nhất. Để sử dụng GPU một cách mượt mà, nếu có sẵn, chúng ta định nghĩa một vài hàm hỗ trợ



(get\_default\_device và to\_device) và một lớp hỗ trợ DeviceDataLoader để chuyển mô hình và dữ liệu của chúng ta lên GPU khi cần thiết.

* 1. Building the model architecture :



Chúng tôi định nghĩa lớp ImageClassificationBase của chúng tôi với các chức năng sau:

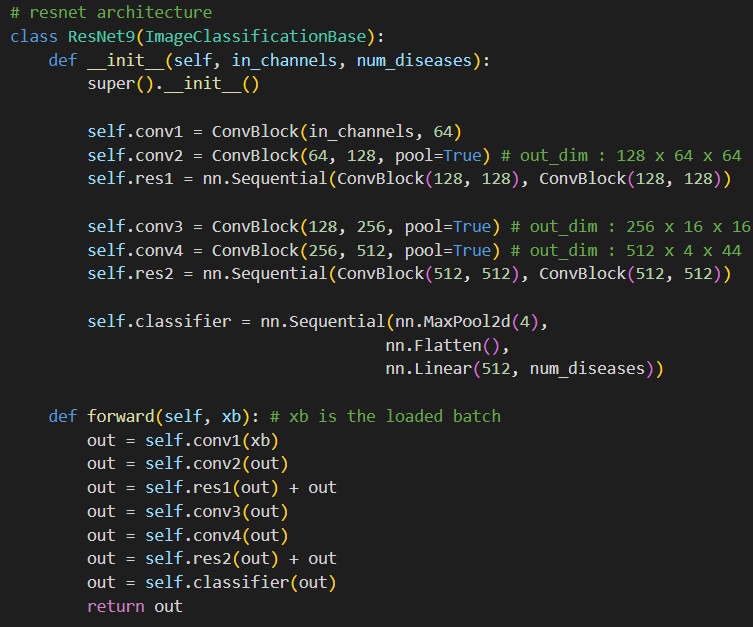
- **training\_step** - Để xác định mức "sai lầm" mà mô hình đang đi sau mỗi bước huấn luyện hoặc xác thực. Chúng tôi sử dụng chức năng này thay vì chỉ một độ chính xác có thể không khả thi (điều này có nghĩa là gradient không thể được xác định, điều cần thiết để mô hình cải thiện trong quá trình huấn luyện). Một cái nhìn nhanh vào tài liệu PyTorch cho ra hàm chi phí: cross\_entropy.

- **validation\_step** - Bởi vì một độ chính xác không thể được sử dụng trong quá trình huấn luyện mô hình không có nghĩa là nó không nên được thực hiện! Độ chính xác trong trường hợp này sẽ được đo lường bằng một ngưỡng và được đếm nếu sự khác biệt giữa dự đoán của mô hình và nhãn thực tế thấp hơn ngưỡng đó.

- **validation\_epoch\_end** - Chúng tôi muốn theo dõi các mất mát/chính xác xác thực và mất mát huấn luyện sau mỗi kỷ nguyên, và mỗi khi làm như vậy, chúng tôi phải đảm bảo rằng gradient không được theo dõi.

- **epoch\_end** - Chúng tôi cũng muốn in ra các mất mát/chính xác xác thực, mất mát huấn luyện và tỷ lệ học tốt vì chúng tôi đang sử dụng bộ lập lịch tỷ lệ học (sẽ thay đổi tỷ lệ học sau mỗi lô huấn luyện) sau mỗi kỷ nguyên.

Chúng tôi cũng định nghĩa một hàm độ chính xác tính toán độ chính xác tổng thể của mô hình trên toàn bộ lô dữ liệu đầu ra, để chúng tôi có thể sử dụng nó làm chỉ số trong fit\_one\_cycle**.**



* 1. Training the model

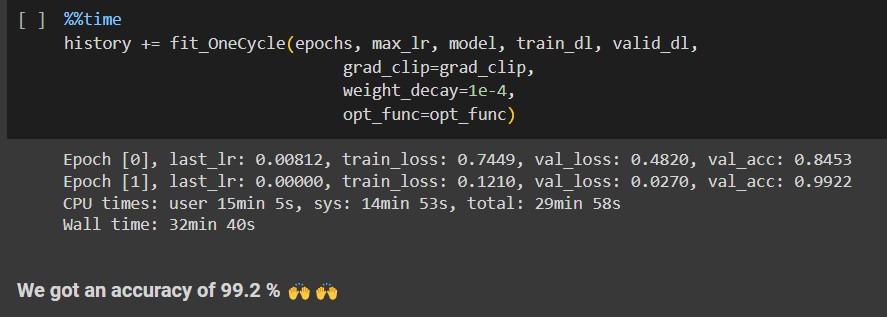
Trước khi huấn luyện mô hình, hãy định nghĩa một hàm tiện ích là hàm evaluate, sẽ thực hiện giai đoạn xác thực, và một hàm fit\_one\_cycle sẽ thực hiện toàn bộ quá trình huấn luyện. Trong fit\_one\_cycle, chúng ta sử dụng một số kỹ thuật:

- **(Learning Rate Scheduling)**: Thay vì sử dụng một tỷ lệ học cố định, chúng ta sẽ sử dụng một bộ lập lịch tỷ lệ học, sẽ thay đổi tỷ lệ học sau mỗi lô huấn luyện. Có nhiều chiến lược để biến đổi tỷ lệ học trong quá trình huấn luyện, và chiến lược chúng ta sử dụng được gọi là "Chính sách Tỉ lệ Học One Cycle", bắt đầu với một tỷ lệ học thấp, từ từ tăng lên từng lô để đạt đến một tỷ lệ học cao khoảng 30% của số kỷ nguyên, sau đó từ từ giảm nó xuống một giá trị rất thấp cho số kỷ nguyên còn lại.

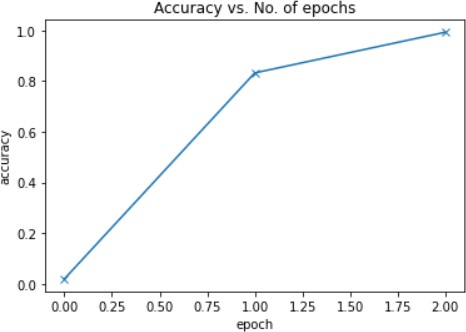
- **(Weight Decay):** Chúng ta cũng sử dụng giảm trọng lượng, một kỹ thuật chính quy hóa ngăn chặn trọng số trở nên quá lớn bằng cách thêm một thuật ngữ bổ sung vào hàm mất mát.

- **(Gradient Clipping)**: Ngoài trọng số và đầu ra của lớp, việc giới hạn giá trị độ dốc vào một phạm vi nhỏ cũng giúp ngăn chặn sự thay đổi không mong muốn trong các tham số do giá trị độ dốc lớn. Kỹ thuật đơn giản nhưng hiệu quả này được gọi là cắt đoạn độ dốc.

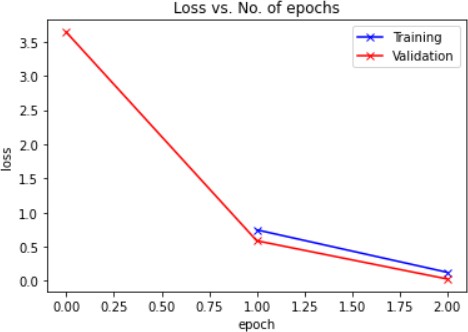
Chúng ta cũng sẽ ghi lại tỷ lệ học được sử dụng cho mỗi lô..



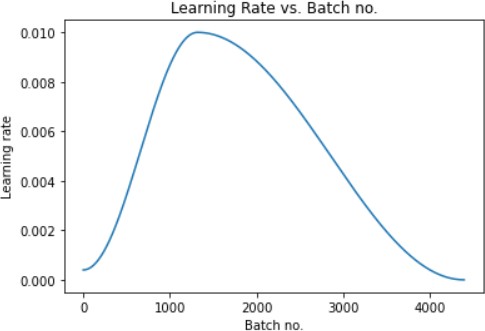
* 1. Plotting



Validation Accuracy



Validation Loss



Learning Rate overtime

* 1. Testing model on test data

Chúng ta chỉ có 33 hình ảnh trong dữ liệu kiểm thử, vì vậy hãy kiểm tra mô hình trên tất cả các hình ảnh. Viết một hàm dự đoán.

