TRƯỜNG ĐẠI HỌC THỦY LỢI
PHÂN HIỆU

Họ và tên tác giả đồ án: Lê Thị Yến Nhi

ĐỀ TÀI: ỨNG DỤNG MẠNG HỌC SÂU/HỌC MÁY TRONG NHẬN DIỆN BỆNH PARKINSON QUA HÌNH ẢNH VÀ ÂM THANH

ĐỒ ÁN MÔN: CHUYÊN ĐỀ CÔNG NGHỆ THÔNG TIN

TP. HCM, ngày 23 tháng 10 năm 2025

Trang phụ bìa

TRƯỜNG ĐẠI HỌC THỦY LỢI

PHÂN HIỆU

Họ và tên tác giả đồ án: Lê Thị Yến Nhi

ĐỀ TÀI: ỨNG DỤNG MẠNG HỌC SÂU/HỌC MÁY TRONG NHẬN DIỆN BỆNH PARKINSON QUA HÌNH ẢNH VÀ ÂM THANH

ĐỒ ÁN MÔN: CHUYÊN ĐỀ CÔNG NGHỆ THÔNG TIN

GIÁO VIÊN HƯỚNG DẪN: TS. Hoàng Văn Quý

TP. HCM, ngày 23 tháng 10 năm 2025

MỤC LỤC

Danh mục các kí hiệu, chữ viết tắt

ResNet: Residual Network

VGG: Visual Geometry Group

STFT: Short-Time Fourier Transform

AI: Artificial Intelligence

DFT: Discrete Fourier Transform

FFT: Fast Fourier Transform

RNN: Recurrent Neural Network

CNN: Convolutional Neural Networks

FC: Fully Connected

Danh mục các hình vẽ, đồ thị

LỜI NÓI ĐẦU

Bệnh Parkinson là một căn bệnh phổ biến ở người già, là một bệnh thoái hóa thần kinh tiến triển, gây ra do tổn thương các tế bào thần kinh tiết dopamine, dẫn đến các triệu chứng vận động điển hình như run, cứng đờ, vận động chậm chạp và mất thăng bằng. Bệnh này ảnh hưởng đến khả năng vận động và nhiều chức năng khác của cơ thể có thể gây tàn phế nhưng không thể chữa khỏi hoàn toàn, tuy nhiên các phương pháp điều trị có thể giúp cải thiện triệu chứng.

Chính vì những lí do này, việc phát hiện bệnh khi có những dấu hiệu sớm có khả năng tiếp cận bệnh và chữa bệnh sớm nhất nhằm ngăn chặn bệnh tiến triển xấu hơn. Từ đó, em có ý tưởng ứng dụng AI mạng học sâu vào quá trình phát hiện bệnh thông qua hình ảnh vẽ tay của các bệnh nhân và âm thanh giọng nói của người bệnh, triển khai web app để người dùng có thể kiểm tra trực tiếp bằng việc tải ảnh và âm thanh một cách dễ dàng tại nhà, qua đó có thể phát hiện khả năng bị bệnh rồi điều trị ở thời điểm sớm nhất.

Chương 1: TỔNG QUAN (CƠ SỞ LÝ THUYẾT)

1.1. Bối cảnh thực tiễn

Trong bối cảnh thế giới nói chung và ngành y học nói riêng hiện nay, việc ứng dụng mô hình AI để hỗ trợ cho quá trình khám chữa bệnh đã không còn là điều viễn vông và quá xa lạ như trước. Tuy nhiên việc này cũng gặp phải khá nhiều thách thức trước tính chính xác và khả năng triển khai mô hình trong thực tiễn.

Đối với căn bệnh Parkinson không còn xa lạ với giới y học, dù được khám chưa bệnh nhưng hiện tại căn bệnh này vẫn chưa thể chữa trị được. Đứng trước thách thức đó, ý tưởng về việc ứng dụng học sâu trong nhận diện căn bệnh qua âm thanh và hình ảnh vẽ tay của các bệnh nhân là một ý tưởng đầy triển vọng.

1.2. Bộ dữ liệu

Việc sử dụng dữ liệu liên quan đến hình ảnh và âm thanh của bệnh nhân là hoàn toàn có cơ sở, dựa trên những đánh giá từ thực tiễn. Đối với căn bệnh Parkinson, việc chẩn đoán và phát hiện bệnh vẫn còn mất nhiều thời gian, chi phí khám chữa.

Theo một nghiên cứu về rối loạn giọng nói của bệnh nhân Parkinson đăng trên tờ báo Vietnam Journal of Physiology vào năm 2022 cho thấy: “Tỉ lệ rối loạn giọng nói ở bệnh nhân parkinson là 91%, trong đó các biểu hiện nhiều nhất là: Giảm độ to (88%), giọng đều đều (80%), hụt hơi khi nói (76%), giảm tốc độ nói (64%)”. Qua đó thấy được giọng nói là một trong những dấu hiệu nhận biết bệnh ở bệnh nhân Parkinson, dữ liệu âm thanh từ đó cũng có thể được thu thập dễ dàng.

Tổ chức WHO cũng đề cập đến việc não của bệnh nhân Parkinson bị ảnh hưởng, từ đó liên đới ảnh hưởng đến các hoạt động thể chất bên ngoài như bị run tay, chân. Qua đó việc xác định bệnh nhân thông qua tranh vẽ của bệnh nhân cũng có tính khả thi.

1.3. Các nghiên cứu

Hiện nay đã có những bài nghiên cứu cũng như, ứng dụng cho việc huấn luyện dữ liệu đối với phát hiện bệnh Parkinson như:

- Nghiên cứu ứng dụng machine learning để phát hiện bệnh Parkinson qua dữ liệu tần số âm thanh của bệnh nhân được đăng tải trên ScienceDirect.

- Nghiên cứu ứng dụng deep learning RNN phát hiện bệnh qua dữ liệu âm thanh được đăng tải trên ResearchGate.
- Một số bộ dữ liệu mẫu về hình ảnh và code được nhiều học sinh training trên Kaggle.

- Các nghiên cứu trên sử dụng nhiều phương pháp khác nhau để thực hiện dự đoán, tuy nhiên vẫn chưa có nghiên cứu nào ứng dụng trực tiếp trên web app kết hợp cả 2 vấn đề nhận diện hình ảnh và âm thanh. Từ đó em quyết định lên ý tưởng về việc triển khai web app thực hiện cả 2 vấn đề này để người dùng có thể lựa chọn 1 trong 2 phương án ngay lập tức.

CHƯƠNG 2: TỔNG QUAN MÔ HÌNH, THU THẬP VÀ TIỀN XỬ LÍ DỮ LIỆU

2.1. Giới thiệu tổng quan về thư viện và mô hình

2.1.1. Thư viện librosa

librosa: là một thư viện Python phổ biến dành cho việc xử lý tín hiệu âm thanh và âm nhạc.

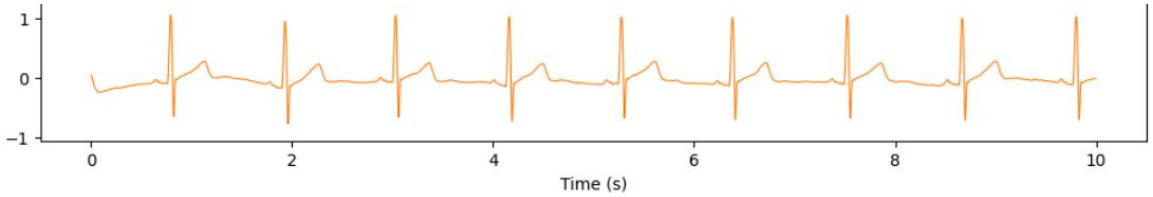
Cách hoạt động:

Thư viện librosa hoạt động theo một quy trình tuyến tính, từ biến đổi âm thanh dạng thô thành các biểu diễn toán học dưới dạng ma trận 2D mà máy tính có thể phân tích được hoặc hỗ trợ lọc nhiễu cho âm thanh, tải lên dữ liệu và biến đổi âm thanh. Âm thanh được trực quan qua các dạng sau:

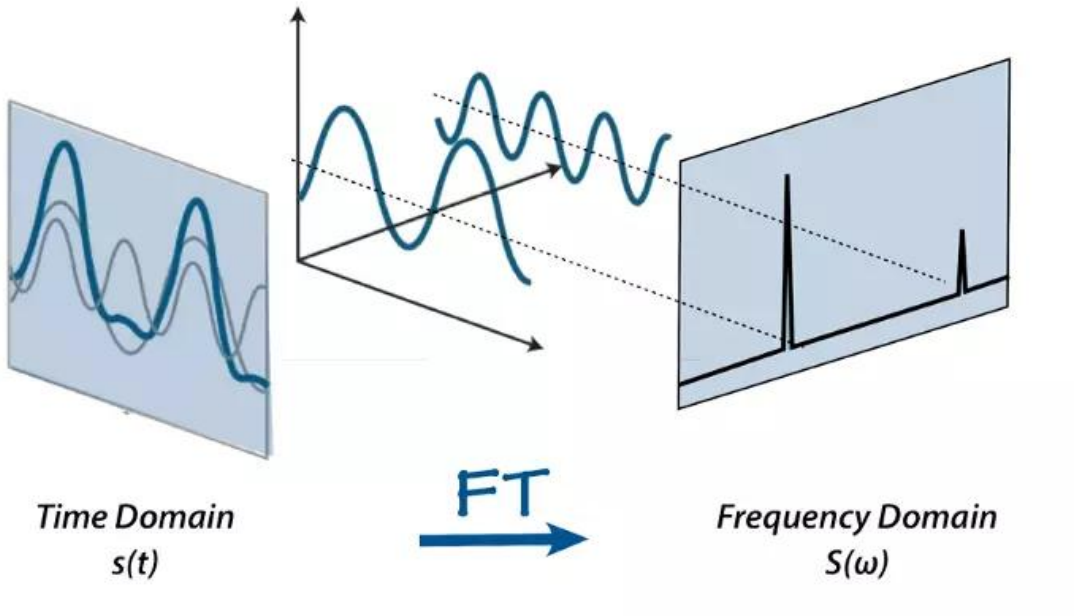
- **Âm thanh thô (Waveform - Time Series)**

Khi dữ liệu âm thanh được tải lên ở dạng wav hay mp3, tệp âm thanh tải lên sẽ được giải nén về dưới dạng mảng Numpy, ở đó mỗi phần tử trong mảng đại diện cho một biên độ sóng của âm thanh ở 1/sample_rate khoảng thời gian của giây. Ví dụ với một file âm thanh dài 5s với sample rate là 21000Hz thì số lượng samples của file này sẽ là 5*21000.

Vậy nếu từ mảng Numpy này với hàng chục hay có thể là trăm nghìn mẫu sẽ vẽ lên một biểu đồ đường gập ghềnh cũng là dạng biểu đồ biểu diễn âm thanh thô (Waveform). Từ ý tưởng này mà phép biến đổi Fourier ra đời giúp chuyển đổi tín hiệu từ miền thời gian sang miền tần số.



Hình 2. 1 Biểu diễn âm than dưới dạng sóng

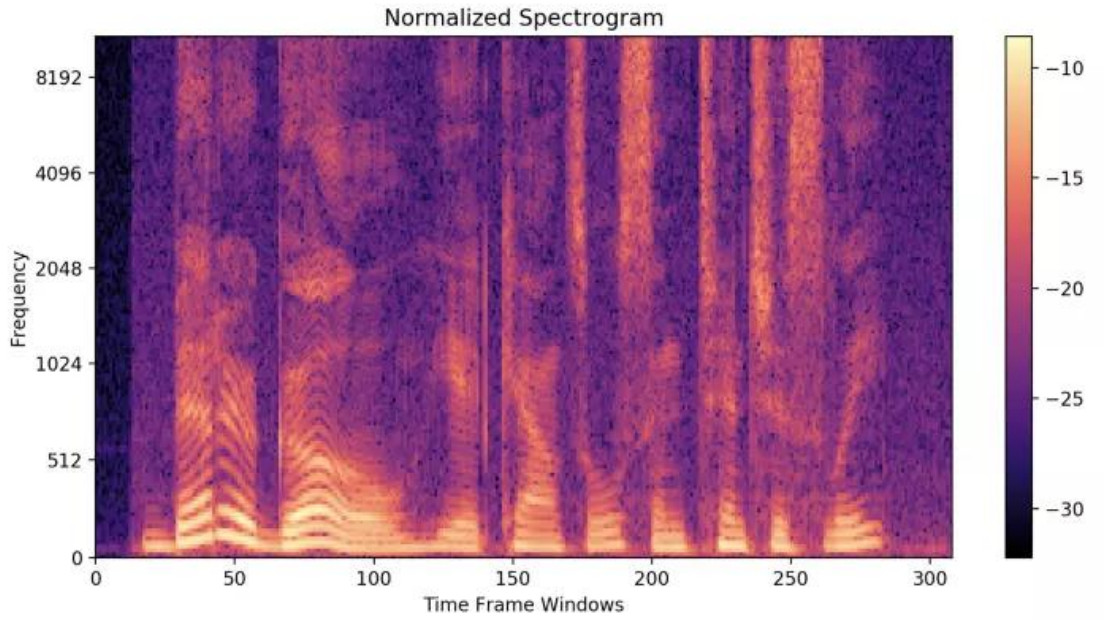


Hình 2. 2 Phép biến đổi Fourier

- **Âm thanh dưới dạng tần số và thời gian (Spectrogram)**

Biểu diễn trực quan các tần số của một tín hiệu nhất định với thời gian (tần số `f`, khung thời gian `t`) được gọi là Spectrogram. Biểu đồ biểu diễn Spectrogram - một trục biểu thị thời gian, trục thứ hai biểu thị tần số và màu sắc biểu thị độ lớn (biên độ) của tần số quan sát tại một thời điểm cụ thể. Màu sắc tươi sáng thể hiện tần số mạnh. Các tần số nhỏ hơn từ (0–1kHz) là mạnh (sáng). Các tần số mạnh chỉ nằm trong khoảng từ 0 đến 1kHz vì đoạn âm thanh này là lời nói của con người.

Ý tưởng chính là chia tín hiệu âm thanh thành các khung nhỏ hơn (cửa sổ) và tính toán DFT (hoặc FFT - Biến đổi Fourier nhanh) cho mỗi cửa sổ đó. Bằng cách này, ta sẽ nhận được tần số cho mỗi cửa sổ và số cửa sổ sẽ đại diện cho thời gian. Để không làm mất một vài tần số khi lấy các cửa sổ một cách liên tục, giữ cho các cửa sổ này chồng lên nhau (overlap).

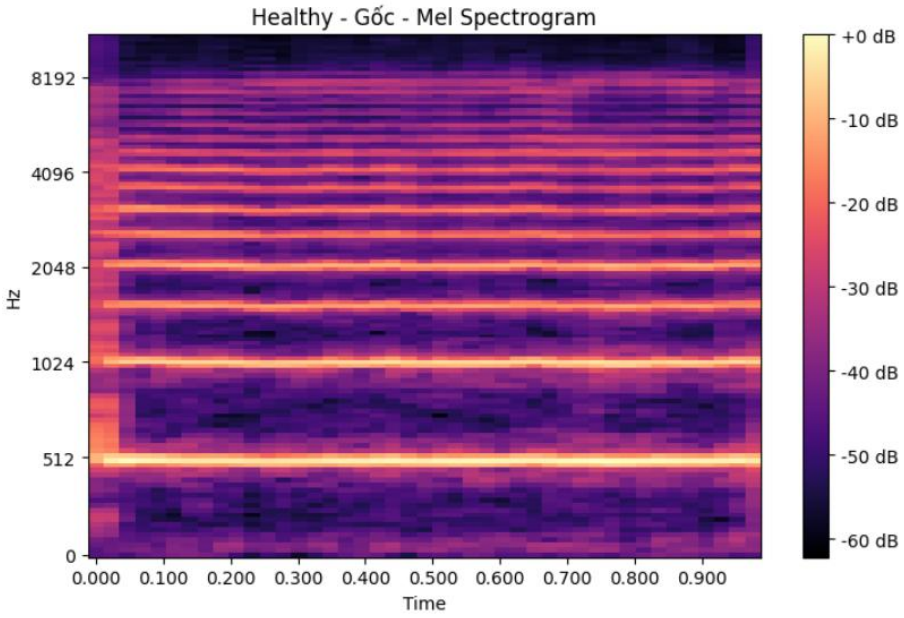


Hình 2. 3 Biểu diễn âm thanh dưới dạng Spectrogram

- **Âm thanh biểu diễn theo miền giá trị Mel (Mel - Spectrogram)**

Tuy spectrogram biểu diễn thông tin về tần số và biên độ âm thanh theo thời gian để ta có thể thấy âm thanh một cách trực quan hơn, tuy nhiên spectrogram chỉ hiển thị âm thanh ở tần số khá nhỏ trong khoảng nghe được của con người (0-1kHz). Vì thế mà chúng không thể biểu diễn hết được thông tin của âm thanh, chính vì thế Mel Spectrogram ra đời để biểu diễn rõ hơn các đặc trưng ấy.

Có thể nói Mel Spectrogram là một dạng biến đổi của spectrogram, nhưng được ánh xạ sang thang Mel - thang tần số cảm nhận của tai người. Quá trình biến đổi này đi từ tín hiệu time series, tính STFT, sau đó ánh xạ các tần số từ thang Hertz sang thang Mel bằng cách sử dụng một bộ lọc (filter bank).

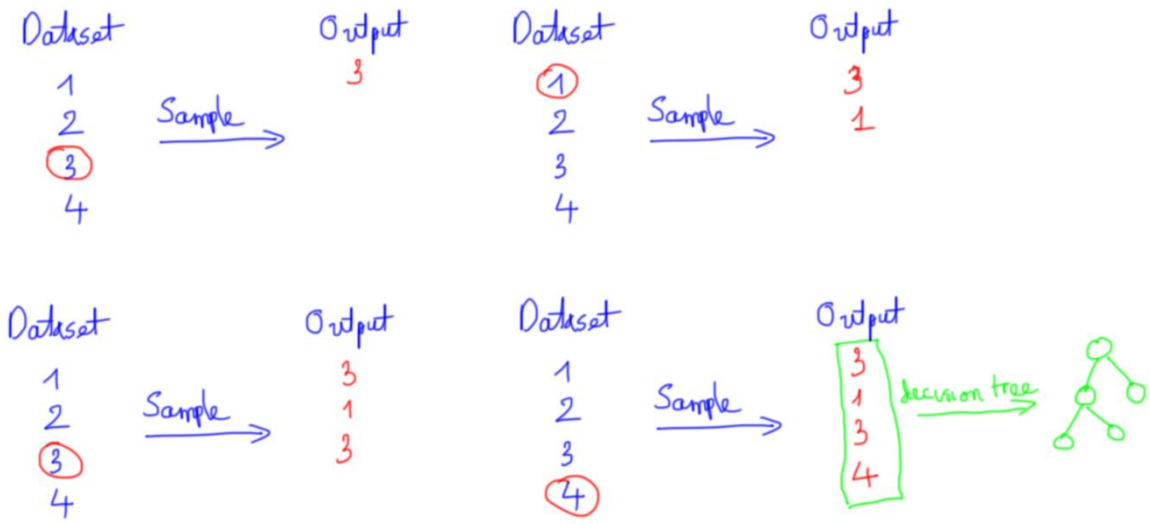


Hình 2. 4 Biểu diễn âm thanh theo dạng Mel Spectrogram

2.1.2. Thuật toán Random Forest

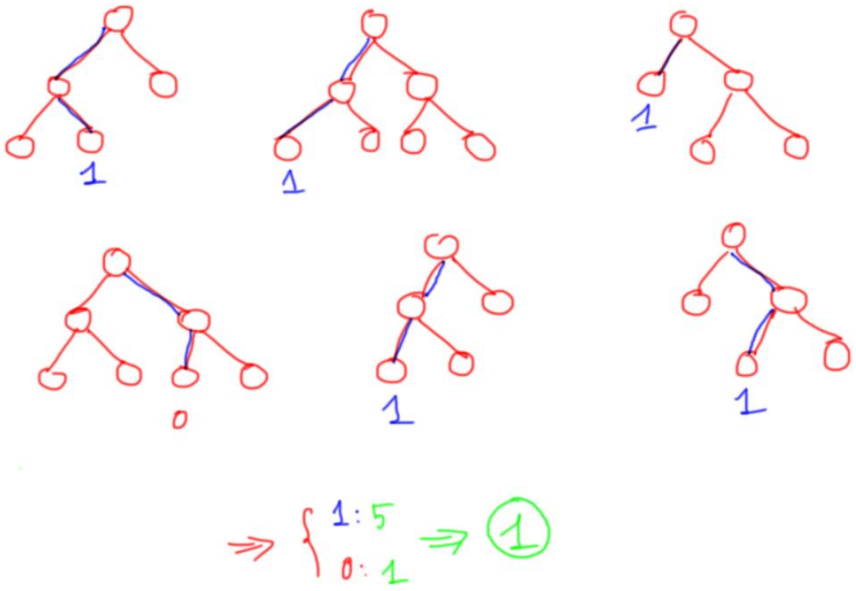
Thuật toán Random Forest hay còn gọi là rừng cây ngẫu nhiên được lấy ý tưởng từ thuật toán của Decision Tree - cây quyết định rồi mở rộng ra bởi các tác giả Leo Breiman và Adele Cutler, chính thức công nhận và đăng kí nhãn hiệu của thuật toán này vào năm 2006.

Random Forest là một thuật toán sử dụng trong học máy thuộc phân khúc supervised learning tức là học có giám sát, hỗ trợ cho các bài toán hồi quy hay phân loại. Ở Random Forest, thuật toán sẽ xây dựng nên nhiều cây quyết định bằng việc lấy n dữ liệu từ bộ dữ liệu huấn luyện. Tức khi mình lấy mẫu được 1 dữ liệu thì mình không bỏ dữ liệu đấy ra mà vẫn giữ lại trong tập dữ liệu ban đầu, rồi tiếp tục lấy mẫu cho tới khi đủ n dữ liệu. Khi dùng kĩ thuật này thì tập n dữ liệu mới của mình có thể có những dữ liệu bị trùng nhau.



Hình 2. 5 Minh họa thuật toán rừng cây ngẫu nhiên

Do quá trình xây dựng mỗi cây quyết định đều có yếu tố ngẫu nhiên (random) nên kết quả là các cây quyết định trong thuật toán Random Forest có thể khác nhau. Thuật toán Random Forest sẽ bao gồm nhiều cây quyết định, mỗi cây được xây dựng dùng thuật toán Decision Tree trên tập dữ liệu khác nhau và dùng tập thuộc tính khác nhau. Sau đó kết quả dự đoán của thuật toán Random Forest sẽ được tổng hợp từ các cây quyết định, lấy kết quả phổ biến nhất thu được từ rừng cây.



Hình 2. 6 Minh họa cách duyệt cây trong rừng

Trong quá trình huấn luyện mỗi cây quyết định đều có những yếu tố ngẫu nhiên:

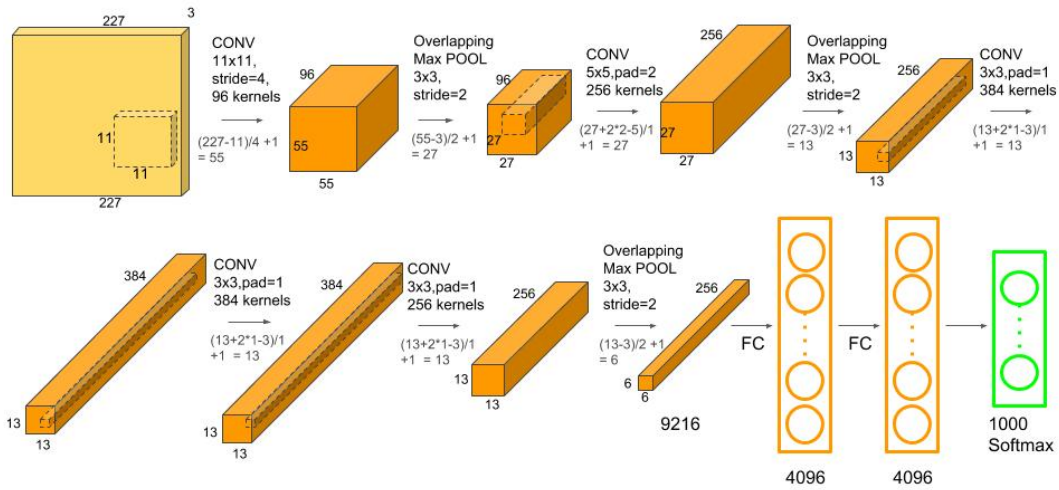
+ Lấy ngẫu nhiên dữ liệu để xây dựng cây quyết định.

+ Lấy ngẫu nhiên các thuộc tính để xây dựng cây quyết định.

Do mỗi cây quyết định trong thuật toán Random Forest không dùng tất cả dữ liệu huấn luyện, cũng như không dùng tất cả các thuộc tính của dữ liệu để xây dựng cây nên mỗi cây có thể sẽ dự đoán không tốt, khi đó mỗi mô hình cây quyết định không bị overfitting mà có thể bị underfitting, hay nói cách khác là mô hình có high bias. Tuy nhiên, kết quả cuối cùng của thuật toán Random Forest lại tổng hợp từ nhiều cây quyết định, thế nên thông tin từ các cây sẽ bổ sung thông tin cho nhau, dẫn đến mô hình có low bias và low variance, hay mô hình có kết quả dự đoán tốt.

2.1.3. Mạng VGG19

VGG là mạng convolutional neural network được đề xuất bởi K. Simonyan and A. Zisserman, University of Oxford. Nghiên cứu của mô hình ứng dụng dựa trên kiến trúc ConvNet cổ điển của AlexNet (Krizhevsky et al., 2012) và các mô hình kế thừa nó trong ILSVRC2013.



Hình 2. 7 Kiến trúc mạng ConvNet

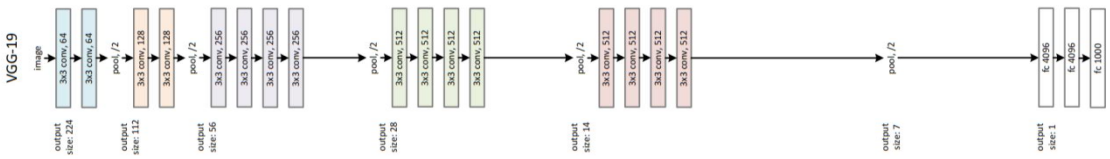
VGG vẫn giữ nguyên kiến trúc cơ bản "convolution layer + max pooling + fully connected layer" của AlexNet, hàm và các kỹ thuật huấn luyện cũ. Tuy nhiên, đột phá lớn nhất của mô hình là tăng độ sâu một cách có hệ thống hơn, thay vì chỉ có 8 tầng tích chập như AlexNet thì VGG tăng độ sâu lên 16-19 tầng. VGG cố định các

tham số khác của kiến trúc và tăng dần độ sâu của mạng bằng cách thêm nhiều tầng tích chập hơn, việc mà khả thi nhờ việc sử dụng các bộ lọc tích chập rất nhỏ (3×3) trong tất cả các tầng. Ở các mô hình trước như AlexNet sử dụng các bộ lọc lớn ở các tầng đầu (11x11), trong khi đó, VGG thay thế các bộ lọc lớn này bằng block chứa các tầng tích chập 3x3. Nghiên cứu cho thấy rằng cứ mỗi block 2 tầng 3x3 có vùng tiếp nhận tương đương 1 tầng 5x5, và block gộp 3 tầng 3x3 tương đương 1 tầng 7x7. Việc này giúp tăng tính phi tuyến (do có nhiều hàm ReLU hơn) và giảm đáng kể số lượng tham số so với việc sử dụng một tầng tích chập lớn.

Mô hình VGG16 tổng cộng có 5 block chính với đầu ra 3 tầng fully connected layer. Dữ liệu đầu vào là ảnh màu với size phù hợp huấn luyện với mô hình là 224x224x3, đi qua block 1 với 2 tầng tích chập 3x3/64 filters, block 2 cũng 2 tầng tích chập nhưng tăng số filters lên 128 , từ block 3, 4, 5 thì tăng thêm 1 tầng tích chập (3/block), số filters tăng lên từ 128 thành 256 và 512 để mô hình học sâu hơn vào các đặc điểm dữ liệu.

Hình 2. 8 Kiến trúc mạng VGG16

Mô hình VGG19 cải tiến hơn mô hình VGG16 với các block 3, 4, 5 đều tăng thêm 1 tầng tích chập 3x3 nữa, sau cùng đầu ra là 1000 nhãn với activation function softmax:



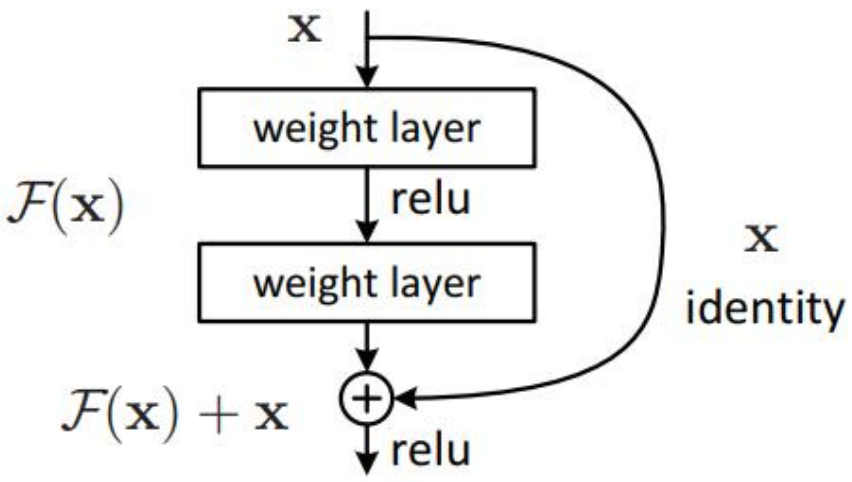
Hình 2. 9 Kiến trúc mạng VGG19

Sau cùng khi sử dụng 2 model VGG16 và VGG 19 để huấn luyện bộ dữ liệu hơn 14 triệu ảnh của ImageNet thì cho ra kết quả rất khả quan so với những mô hình cũ, độ chính xác của 2 mô hình cũng rất suýt xao khi top 5 error test của VGG16 và VGG19 lần lượt là 7,2 và 7,1%, từ đó ta tính được độ chính xác lần lượt 92,8 và 92,9%. Cũng từ kết quả này mà em quyết định lấy VGG19 để huấn luyện dữ liệu song song với ResNet, so sánh kết quả của 2 bên để có cái nhìn tổng quát hơn, đánh giá hiệu suất mô hình và lựa chọn được mô hình phù hợp nhất.

2.1.4. Mạng ResNet

Trước khi ResNet được nghiên cứu và cho ra đời, cũng đã có những mô hình khác xuất hiện như: Plain Network, LeNet hay VGG. Tuy nhiên, những mô hình này xuất hiện vấn đề về Vanishing Gradient (Tiêu Biến Độ Dốc) hay Exploding Gradient (Bùng Nổ Độ Dốc), là hiện tượng khi mà mô hình được huấn luyện và học tập từ dữ liệu, khi tính toán hàm mất mát (loss function) trong quá trình lan truyền ngược (backpropagation) độ dốc (gradient) của hàm quá nhỏ hoặc quá lớn khiến cho các trọng số không được cập nhật hay trọng số quá lớn làm cho mô hình huấn luyện trở nên chậm đi hoặc không hội tụ được khiến hiệu suất của mô hình giảm đi rất nhiều, với đầu vào là dữ liệu mới có thể dẫn tới việc dự đoán không chính xác gây ra sai lệch lớn.

Mạng ResNet ra đời nhóm nghiên cứu đã đưa ra khái niệm "Khối dư" (residual block) và sử dụng kỹ thuật Skip Connection (bỏ qua kết nối) mục đích là để bỏ qua kết nối các hoạt động của một lớp với các lớp tiếp theo bằng việc bỏ qua một số lớp ở giữa, điều này tạo thành Khối Dư dẫn đến các mạng được tạo ra bằng cách xếp chồng các khối dư này lại tạo thành kiến trúc mạng nơ-ron dư thừa. Việc đưa ra khái niệm mới và kỹ thuật Skip Connection nhằm để giải quyết vấn đề về hiện tượng Tiêu Biến/Bùng Nổ độ dốc - vấn đề mà những mô hình trước đó chưa làm được.



Hình 2. 10 Kĩ thuật Skip Connection

Với:

- x là tập hợp các lớp tích chập
- F(x) là hàm phần dư (residual function)
- H(x) là hàm ánh xạ từ đầu vào đến đầu ra

Thay vì học một ánh xạ trực tiếp từ H(x) như những mạng khác, mạng ResNet sẽ học một ánh xạ phần dư F(x) = H(x) - x và đầu ra cuối cùng là F(x) + x. Điều này giúp cải thiện và hạn chế được những vấn đề về Tiêu Biến Độ Dốc và Bùng Nổ Độ Dốc.

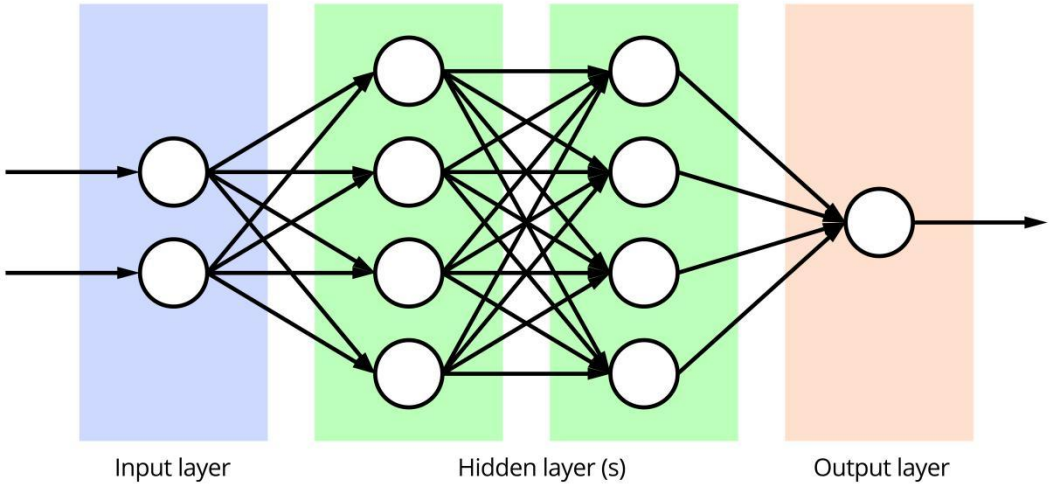
Tính năng

Tiến hành tìm hiểu sâu hơn, kiến trúc của mạng ResNet có các tính năng chính sau:

- Kết nối phần dư (Residual Connections): kết hợp các kết nối dư, cho phép huấn luyện mạng nơ-ron rất sâu và khắc phục vấn đề tiêu biến độ dốc.
- Ánh xạ đồng nhất (Identity Mapping): với hàm đầu ra cuối là F(x) + x giúp quá trình huấn luyện dễ dàng hơn thay vì ánh xạ thực tế trực tiếp thông qua H(x).
- Độ sâu (Depth): cho phép tạo ra các mạng nơ-ron rất sâu, từ đó cải thiện hiệu suất trong các tác vụ nhận diện hình ảnh.
- Giải quyết vấn đề về tham số: thông qua kiến trúc ResNet giúp mô hình làm việc hiệu quả hơn bằng việc giảm thiểu tham số.

Kiến trúc

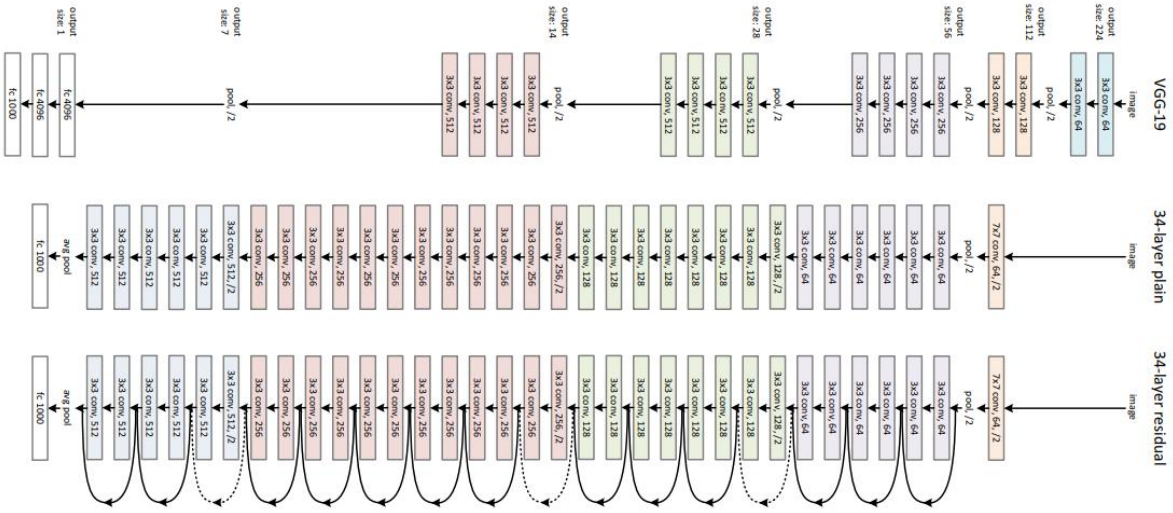
Mạng ResNet là một trong nhiều mô hình khác của mạng nơ-ron tích chập (CNN), có thể thấy về cấu trúc cơ bản mạng ResNet vẫn được cấu tạo bởi 3 phần chính: đầu vào (input), các lớp ẩn (hidden layers) và sau cùng là đầu ra (output) [5].



Hình 2. 11 Kiến trúc mạng học sâu

Với việc sử dụng mạng ResNet để nhận diện hình ảnh thì lớp đầu vào lúc này sẽ là ảnh, có thể là ảnh xám với số lượng màu đơn điệu hay ảnh màu RGB với số màu đa dạng hơn, được biểu diễn phức tạp hơn dưới dạng ma trận kích thước theo tùy kích thước của bức ảnh và chồng nhiều ma trận màu lên nhau. Sau đó, trải qua các lớp tích chập khác nhau để đến lớp tích chập đầy đủ cuối cùng trước khi đến lớp đầu ra.

Bằng việc sử dụng kiến trúc mạng Plain được lấy cảm hứng từ mô hình VGG-19, sau đó cải tiến để thêm Skip Connections vào mô hình từ đó kiến trúc mạng ResNet ra đời. Dưới đây là mô hình kiến trúc của VGG-19, Plain và ResNet:



Hình 2. 12 So sánh kiến trúc mạng ResNet với VGG19 và PlainNet

Mạng ResNet có các phiên bản: ResNet18, ResNet34, ResNet50, ResNet101, ResNet152,.. với các số tương ứng với số lớp tích chập được sử dụng của mô hình, càng về sau số lớp càng cao thì mô hình càng mạnh hơn cho hiệu suất nhận diện cao hơn nhưng cũng đồng nghĩa với việc tốn tài nguyên cũng như lượng lớn dữ liệu cần để huấn luyện mô hình. Dưới đây là bảng tổng hợp so sánh các lớp tích chập và thông số theo từng mô hình ResNet.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Hình 2. 13 Cấu trúc các tầng tích chập của ResNet

Có thể thấy với 5 mô hình thì mặc định luôn có 5 tầng tích chập: conv1, conv2_x, conv3_x, conv4_x, conv5_x. Dữ liệu sẽ được trải qua lớp trích xuất đặc trưng (max-pooling) cùng các tầng tích chập mà ở đó các lớp tích chập được tổng hợp và gói

gọn trong từng khối cơ bản (basic block) hay khối cổ chai (bottleneck block), tương ứng theo mỗi mô hình sẽ có số lượng khối cổ chai với các lớp tích chập khác nhau, sau cùng trải qua lớp kết nối cùng (fully-connected layer) với hàm kích hoạt (activation function) trước khi qua lớp đầu ra. Trong đó:

- **Khối cơ bản - Basic Block:** khối cơ bản, mỗi khối đều sử dụng bộ lọc (kernel) 3x3 để trích xuất đặc trưng nên đầu vào và đầu ra của khối đều có số lượng kênh (channels) như nhau. Sử dụng cho mô hình ResNet18/34 vì 2 mô hình này có độ sâu khá tương đối, việc sử dụng khối cơ bản cho mô hình này đã đủ để có thể trích xuất các đặc trưng cần thiết cho mô hình học và huấn luyện.

- **Khối cổ chai - Bottleneck Block:** khác với khối cơ bản, khối cổ chai giúp thu hẹp số lượng kênh sau đó mở rộng trở lại nhằm để giảm độ phức tạp tính toán và số lượng tham số trong khi vẫn duy trì khả năng học các đặc trưng phức tạp. Từ ResNet50 trở đi thì độ sâu của mô hình bắt đầu sâu hơn nên việc giảm và tăng chiều, trích xuất đặc trưng giúp giảm độ phức tạp tính toán, tăng hiệu suất mô hình hơn. Trong mỗi khối cổ chai, chia các lớp tích chập thành conv1: sử dụng bộ lọc 1x1, conv2: sử dụng bộ lọc 3x3, conv3: sử dụng bộ lọc 1x1 tương ứng với tính năng của chúng khi conv1 dùng để giảm số lượng kênh (channels) đầu vào, conv2 dùng để trích xuất đặc trưng từ dữ liệu sau khi được giảm chiều và cuối cùng là conv3 giúp tăng chiều khi dùng bộ lọc 1x1 để tăng số lượng kênh trở lại.

- **Bộ lọc - Kernel:** là một ma trận nhỏ và ma trận này luôn là ma trận lẻ, ví dụ 1x1 hay 3x3,… được sử dụng trong các lớp tích chập, ma trận này sẽ trượt qua ma trận dữ liệu - ở bài toán của chúng ta là dữ liệu ảnh màu thực hiện phép tính tích chập tạo ra bản đồ đặc trưng từ dữ liệu đầu vào, từ đó giúp giảm số lượng tham số và phát hiện được các đặc trưng quan trọng của ảnh.

- **Lớp trích xuất đặc trưng - Pooling Layer:** mục đích là để giảm kích thước dữ liệu nhưng vẫn giữ được các thuộc tính quan trọng.

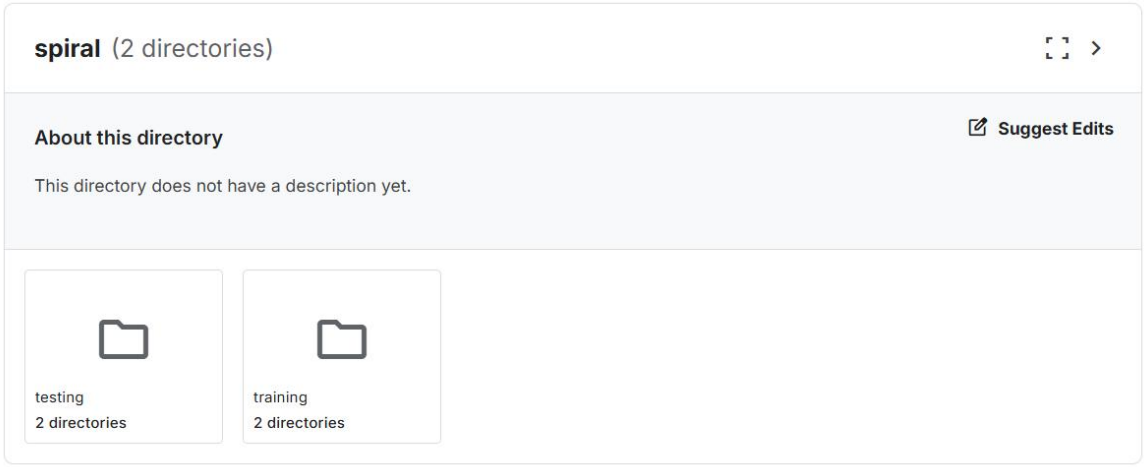
- **Lớp kết nối cuối cùng - Fully-Connected Layer:** là lớp cuối cùng của mô hình trước khi đưa ra đầu ra, sau khi ảnh được truyền qua nhiều lớp tích chập cũng như lớp trích xuất đặc trưng thì mô hình đã học được những đặc điểm khác nhau của hình ảnh, lúc này ảnh sẽ được làm phẳng tức là chuyển ma trận từ chiềucao*chiều rộng*chiều sâu (H*W*D) thành 1 vector, hay hiểu đơn giản hơn ví dụ từ ma trận 3x3, có 3 hàng thì những hàng ấy chuyển về dạng cột và nối đuôi nhau tạo thành một vector dài vector này đi qua lớp kết nối để kết hợp các đặc điểm của ảnh từ đó được đầu ra của mô hình.

- **Hàm kích hoạt - Activation Function:** được sử dụng sau mỗi lớp tích chập và lớp kết nối cuối cùng giúp mô hình học được các mối quan hệ phi tuyến phức tạp, cải thiện khả năng biểu diễn của mô hình, quyết định đầu ra dựa trên đầu vào. Có các loại hàm kích hoạt khác nhau, tùy theo mục đích sử dụng khác nhau chẳng hạn như hàm softmax dành cho các bài toán có đầu ra với dự đoán là tỉ lệ phần trăm, hàm tanh trả về giá trị từ [-1,1] cho các bài toán dự đoán cảm xúc,..

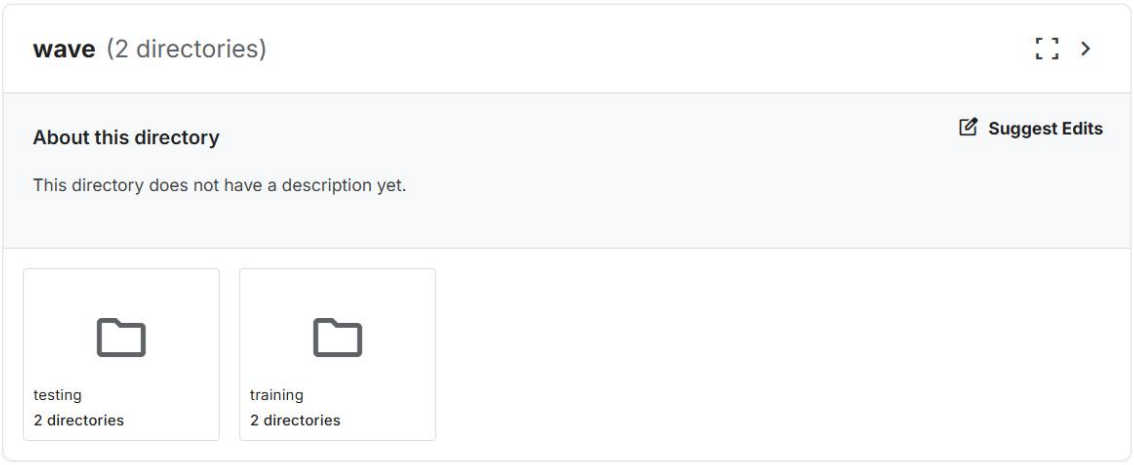
2.2. Thu thập dữ liệu

2.2.1. Thu thập dữ liệu hình ảnh

Do tính riêng tư của dữ liệu các bệnh nhân nên việc thu thập dữ liệu được em lựa chọn và xáo trộn với nhau từ những tập dữ liệu hình ảnh vẽ tay dạng sóng cong tròn và dạng sóng uống lượn của 2 nhãn bệnh: parkinson và healthy, có sẵn miễn phí trên trang web kaggle với lượng dữ liệu gốc là 204 hình ảnh với đuôi dạng png được chia sẵn dữ liệu ra 2 thư mục: spiral (dạng sóng cong tròn), wave (dạng sóng uốn lượn). Mỗi folder dạng sóng sẽ chia ra thêm 2 thư mục testing và training của 2 nhãn bệnh parkinson và healthy.



Hình 2. 14 Cấu trúc thư mục ảnh Spiral



Hình 2. 15 Cấu trúc thư mục ảnh Wave

Nguồn dữ liệu được lấy từ trang Kaggle:

https://www.kaggle.com/datasets/kmader/parkinsons-drawings

2.2.2. Thu thập dữ liệu âm thanh

Tương tự như dữ liệu hình ảnh, thì file âm thanh của các bệnh nhân cũng được bảo mật và rất hiếm để có thể thu thập được từ các bệnh viện, việc tìm kiếm dữ liệu cũng khá khó khăn khi không có nhiều tổ chức hỗ trợ công khai dữ liệu. Em quyết định lấy bộ dữ liệu âm thanh được đăng tải trên figshare, được tài trợ bởi National Science Foundation và National Institutes of Health.

Dữ liệu chia làm 2 thư mục lần lượt là file âm thanh dưới dạng .wav của bệnh nhân bị bệnh Parkinson và bệnh nhân khỏe mạnh: PD_AH và HC_AH, với tổng số bệnh nhân thu thập được là 40 với nhãn Parkinson và 41 người khỏe mạnh.

Hình 2. 16 Định dạng dữ liệu âm thanh bệnh nhân

Hình 2. 17 Định dạng dữ liệu âm thanh người khỏe mạnh

Nguồn dữ liệu được lấy từ figshare:
https://figshare.com/articles/dataset/Voice_Samples_for_Patients_with_Parkinson_s_Disease_and_Healthy_Controls/23849127?file=41836710

2.3. Xử lí dữ liệu

2.3.1. Đối với dữ liệu dạng hình ảnh

Khi huấn luyện dữ liệu với mô hình ResNet50 và VGG19, kích thước ảnh đầu vào phù hợp và cho ra kết quả tốt nhất với kích thước ảnh là 224x224 pixels. Nên cần điều chỉnh lại tất cả kích thước của ảnh (RandomResizedCrop). Ngoài ra, để cho mô hình có thể học được hình ảnh ở những góc độ khác nhau, ta lật, xoay hình ảnh ở nhiều góc (RandomRotation) và điều chỉnh ảnh ở chế độ ảnh màu (Normalize). Bên cạnh đó do sử dụng model ResNet50, VGG19 của Pytorch nên ta cũng đổi dạng ảnh từ PIL Image sang Pytorch Tensor (To Tensor).

```
transform = transforms.Compose([
    transforms.RandomHorizontalFlip(),
    transforms.RandomRotation(10),
    transforms.RandomResizedCrop(224, scale=(0.8, 1.0)), #224x224
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
])
```

2.3.2. Đối với dữ liệu dạng âm thanh

Khi huấn luyện dữ liệu dưới dạng âm thanh, ta có nhiều bước tiền xử lí khác nhau tùy thuộc theo hướng giải quyết vấn đề theo kiểu huấn luyện cho mô hình học máy hay mô hình học sâu. Với dạng dữ liệu âm thanh này, em giải quyết theo hướng sử dụng mô hình học máy, tiền xử lí với thư viện âm thanh librosa và trích xuất đặc trưng, từ đó huấn luyện dữ liệu với thuật toán Random Forest.

Do dữ liệu đầu vào là âm thanh giọng nói con người, ta đặt tần số lấy mẫu về chuẩn của thư viện librosa hỗ trợ là 22050Hz và cố định độ dài cho file âm thanh là 3 giây để đảm bảo tính đồng nhất:

```
class AudioPreprocessor:
    def __init__(self, sr=22050, duration=3):
        self.sr = sr
```

```
        self.duration = duration
```

Kế đến, tải và chuẩn bị file âm thanh bằng cách dùng librosa, tính chiều dài âm thanh bằng nhân tần số lấy mẫu với độ dài thời gian, nếu âm thanh ngắn hơn 3 giây thì thêm khoảng lặng và cắt bớt đi nếu âm thanh quá dài:

```
    def load_and_clean_audio(self, file_path):
        try:
            # Tải và chuẩn hóa độ dài âm thanh
            y, sr = librosa.load(file_path, sr=self.sr, duration=self.duration)
            target_length = self.sr * self.duration
            if len(y) < target_length:
                y = np.pad(y, (0, target_length - len(y)), mode='constant')  # Thêm khoảng lặng nếu âm thanh quá ngắn
            else:
                y = y[:target_length]  # Cắt bớt nếu âm thanh quá dài
```

Sau khi đã chuẩn hóa đồng nhất các file âm thanh về cùng tần số và độ dài âm thanh thì ta tiếp tục xử lý âm thanh, vì trong quá trình thu âm thanh sẽ có thể chứa lẫn lộn các tạp âm, tiếng thở, tiếng cử động, ngắt quãng điều đó sẽ ảnh hưởng đến quá trình huấn luyện và dự đoán.

Loại bỏ khoảng lặng:

```
    def remove_silence(self, y, top_db=25):
        try:
            intervals = librosa.effects.split(y, top_db=top_db)
            if len(intervals) > 0:
                return np.concatenate([y[start:end] for start, end in intervals])
        except:
            pass
        return y
```

top_db=25: Ngưỡng âm lượng (25 dB) để xác định khoảng lặng

librosa.effects.split(): Phát hiện các đoạn có âm thanh thực sự

np.concatenate(): Ghép các đoạn có âm thanh lại với nhau

Giảm nhiễu:

```
    def spectral_gating(self, y, n_fft=2048, hop_length=512, noise_frames=5, reduction=0.5):
        try:
            stft = librosa.stft(y, n_fft=n_fft, hop_length=hop_length)
            magnitude, phase = np.abs(stft), np.angle(stft)
            noise_profile = np.mean(magnitude[:, :noise_frames], axis=1, keepdims=True)
            magnitude_clean = np.maximum(magnitude - reduction * noise_profile, 0)
            y_clean = librosa.istft(magnitude_clean * np.exp(1j * phase), hop_length=hop_length)
            return y_clean
        except:
            return y
```

n_fft=2048: Kích thước cửa sổ FFT để phân tích tần số

noise_frames=5: Số khung đầu tiên được coi là nhiễu nền

magnitude - reduction * noise_profile: Trừ đi profile nhiễu từ tín hiệu

librosa.istft(): Chuyển đổi ngược lại sang dạng sóng thời gian

Chuẩn hóa biên độ:

```
    def normalize_audio(self, y):
        return y / (np.max(np.abs(y)) + 1e-8)
```

Chia tất cả giá trị cho biên độ lớn nhất để đưa về khoảng [-1, 1]

+ 1e-8: Tránh chia cho 0 trong trường hợp tín hiệu bằng 0

Tăng cường dữ liệu âm thanh để quá trình huấn luyện tốt hơn do số lượng dữ liệu quá ít:

```
    def augment_audio(self, y, sr):
```

```
        aug_choice = random.choice(['pitch', 'stretch', 'noise', 'none'])
        if aug_choice == 'pitch':
            # Dịch cao độ ±2 semitones
            n_steps = random.uniform(-2, 2)
            y = librosa.effects.pitch_shift(y, sr=sr, n_steps=n_steps)
        elif aug_choice == 'stretch':
            # Giãn hoặc co thời gian (0.9–1.1x)
            rate = random.uniform(0.9, 1.1)
            y = librosa.effects.time_stretch(y=y, rate=rate)
            # Cắt/pad lại cho đúng độ dài
            target_length = self.sr * self.duration
            if len(y) < target_length:
                y = np.pad(y, (0, target_length - len(y)), mode='constant')
            else:
                y = y[:target_length]
        elif aug_choice == 'noise':
            # Thêm nhiễu Gaussian nhẹ
            noise = np.random.normal(0, 0.005, y.shape)
            y = y + noise
```

pitch_shift(): Thay đổi cao độ ±2 nốt nhạc

time_stretch(): Thay đổi tốc độ phát 0.9-1.1 lần

random.normal(): Thêm nhiễu Gaussian nhẹ (độ lệch chuẩn 0.005)

Mỗi file chỉ áp dụng một kỹ thuật augmentation ngẫu nhiên để đảm bảo tính đa dạng của dữ liệu.

Trích xuất đặc trưng âm thanh cho quá trình huấn luyện với học máy:

```
    def extract_features(self, y):
        try:
            mfcc = librosa.feature.mfcc(y=y, sr=self.sr, n_mfcc=13)
            chroma = librosa.feature.chroma_stft(y=y, sr=self.sr)
```

```
        spec_centroid = librosa.feature.spectral_centroid(y=y, sr=self.sr)
        spec_bandwidth = librosa.feature.spectral_bandwidth(y=y, sr=self.sr)
        rolloff = librosa.feature.spectral_rolloff(y=y, sr=self.sr)
        zcr = librosa.feature.zero_crossing_rate(y)

        features = np.hstack([
            np.mean(mfcc, axis=1), np.std(mfcc, axis=1),
            np.mean(chroma, axis=1), np.std(chroma, axis=1),
            np.mean(spec_centroid), np.std(spec_centroid),
            np.mean(spec_bandwidth), np.std(spec_bandwidth),
            np.mean(rolloff), np.std(rolloff),
            np.mean(zcr), np.std(zcr)
        ])
        return features
```

Ta lựa chọn và trích xuất lọc ra 6 loại đặc trưng quan trọng trong âm thanh để tiến hành trích xuất dữ liệu:

MFCC (13 coefficients): Đặc trưng tần số quan trọng cho nhận dạng giọng nói

Chroma STFT: Đặc trưng âm nhạc, biểu diễn phân bố năng lượng theo 12 nốt

Spectral Centroid: Trung tâm phổ tần số, đại diện cho "độ sáng" của âm thanh

Spectral Bandwidth: Độ rộng băng thông phổ

Spectral Rolloff: Tần số dưới đó tập trung 85% năng lượng

Zero Crossing Rate: Tốc độ cắt trục hoành, đặc trưng cho âm nhiễu

Tính cả mean và std để nắm bắt cả giá trị trung bình và độ biến thiên. Sau khi tính toán và trích xuất ra được những giá trị khác nhau tùy theo loại đặc trưng trích xuất ta được 1 vector tương ứng với số đặc trưng đã được trích xuất. Từ đây dựa trên tổng tập dữ liệu huấn luyện sẽ ra được ma trận N dữ liệu nhân với số đặc trưng trích xuất làm đầu vào cho học máy.

CHƯƠNG 3: THỰC NGHIỆM VÀ KẾT QUẢ ĐẠT ĐƯỢC

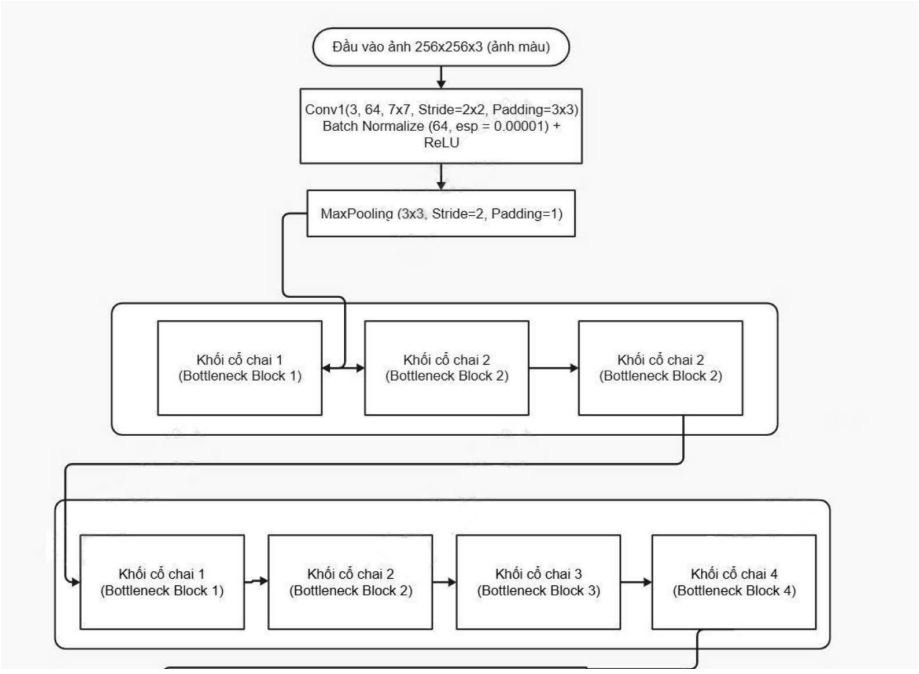
3.1. Xây dựng mô hình huấn luyện

3.1.1. Xây dựng mô hình cho huấn luyện hình ảnh

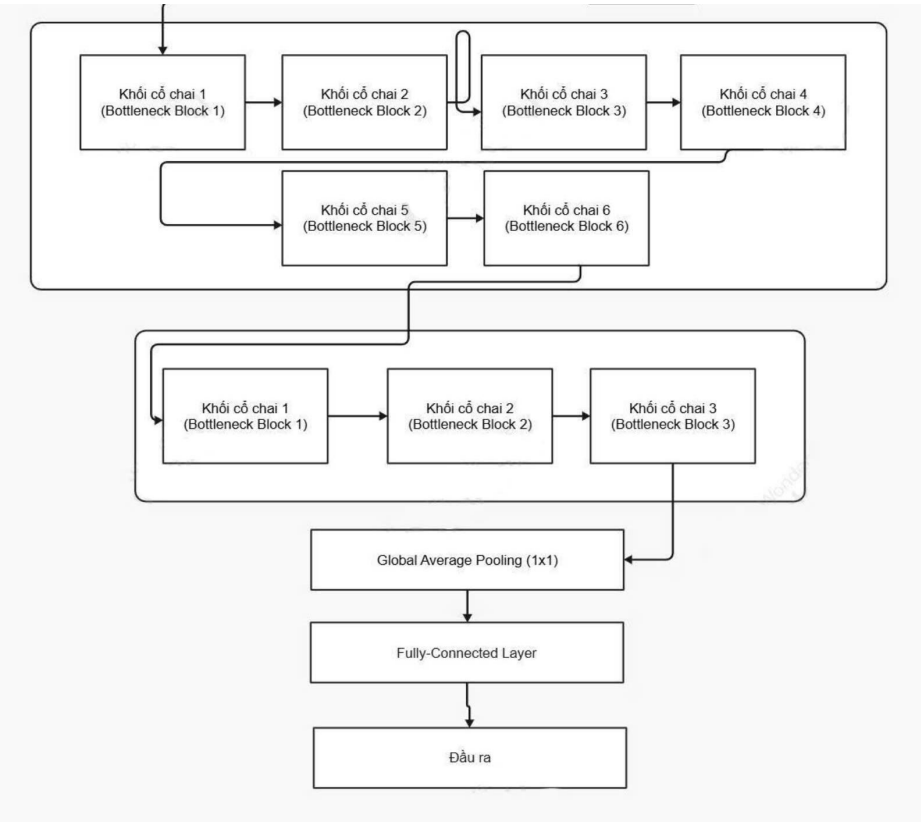
3.1.1.1. Xây dựng mô hình ResNet50

Với ResNet, em sử dụng mô hình ResNet50 với số tầng sâu hơn phù hợp để học chi tiết các đặc trưng của dữ liệu, với kĩ thuật Skip Connection cũng như các khối Residual Blocks giúp giảm mất mát thông tin đáng kể. Mô hình có khả năng phát hiện Deepfake tốt hơn nhưng đồng thời cũng yêu cầu tài nguyên lớn.

ResNet50 sẽ có 4 tầng tích chập chính mà mỗi tầng sẽ có số khối cổ chai khác nhau, mỗi khối cổ chai sẽ có 3 lớp tích chập, lớp tích chập đầu tiên mang bộ lọc 1x1 mang nhiệm vụ giảm số lượng kênh, tiếp tục đến lớp tích chập số 2 sẽ là 3x3 để trích xuất đặc trưng và sau cùng là lớp có bộ lọc 1x1 lần nữa mà lần này sẽ mang nhiệm vụ tăng số lượng kênh trở lại để đến khối cổ chai tiếp theo.

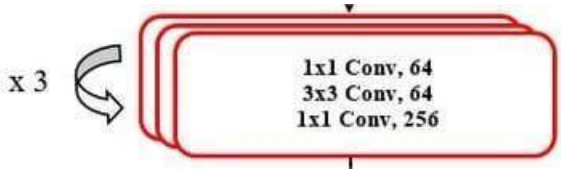


Hình 3. 1 Kiến trúc ResNet 50



Hình 3. 2 Kiến trúc ResNet50

Đây là ảnh minh họa cho tầng tích chập đầu tiên với 3 khối cổ chai:

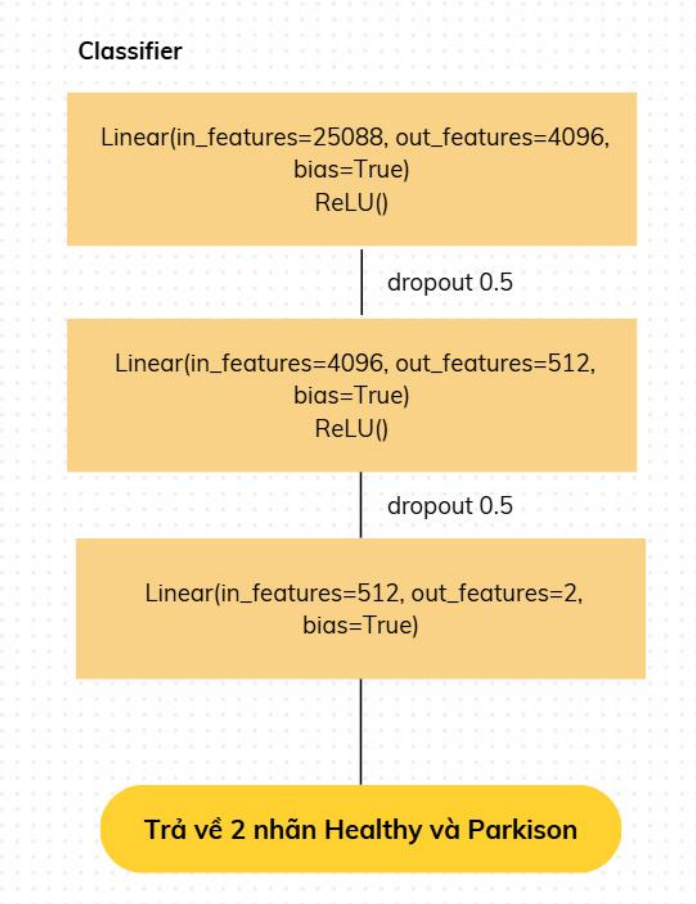


Hình 3. 3 Minh họa tầng tích chập đầu tiên của ResNet50

3.1.1.2. Xây dựng mô hình VGG19

Với VGG19 có kiến trúc đồng nhất và sâu, phù hợp để học các đặc trưng hình ảnh ở nhiều mức độ trừu tượng khác nhau thông qua việc sử dụng liên tiếp các bộ lọc 3x3 nhỏ giúp tăng độ phi tuyến mà không làm tăng đáng kể tham số. Tuy nhiên, do độ sâu của mạng nên VGG19 yêu cầu tài nguyên tính toán lớn.

VGG19 sẽ có 5 khối tích chập chính mà mỗi khối sẽ có từ 2-4 lớp tích chập liên tiếp với số bộ lọc tăng dần 64-128-256-512-512, mỗi lớp tích chập đều sử dụng bộ lọc 3x3 với padding=1 để giữ nguyên kích thước không gian, sau mỗi khối là một lớp MaxPooling 2x2 để giảm kích thước. Cuối cùng là 3 lớp Fully Connected với Dropout 0.5 để phân loại thành 2 lớp Healthy và Parkinson.



Hình 3. 4 Minh họa 3 lớp FC cuối cùng của VGG19

3.1.2. Xây dựng mô hình cho huấn luyện âm thanh

Đối với dữ liệu âm thanh, ở phần tiền xử lý ta đã làm các bước từ xử lí dữ liệu đến lọc nhiễu, đồng nhất tần số lấy mẫu và độ dài âm thanh, kế đến ta chia dữ liệu thành X, y lần lượt chứa vector đặc trưng và nhãn dự đoán, chia train/test với 80% dữ liệu huấn luyện và 20% dữ liệu kiểm thử:

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=42
)
```

Tạo huấn luyện cho Random Forest:

```
clf = RandomForestClassifier(
    n_estimators=300,
    max_depth=10,
    min_samples_split=3,
    min_samples_leaf=2,
    max_features='sqrt',
    class_weight='balanced',
    random_state=42
)
clf.fit(X_train, y_train)
```

Tối ưu hóa tham số:

n_estimators=300: Đặt số cây trong rừng là 300

max_depth=10: Ngăn overfitting bằng giới hạn độ sâu

class_weight='balanced': Tự động cân bằng tập mất mát cho lớp thiểu số

max_features='sqrt': Giảm tương quan giữa các cây, cải thiện độ chính xác

3.2. Kết quả đạt được

3.2.1. Kết quả huấn luyện cho dữ liệu hình ảnh

Đối với mô hình ResNet50:

Classification Report cho sóng cong tròn (Spiral):

```
Classification Report:
              precision    recall  f1-score   support

     healthy       0.93      0.93      0.93        15
   parkinson       0.93      0.93      0.93        15

    accuracy                           0.93        30
   macro avg       0.93      0.93      0.93        30
weighted avg       0.93      0.93      0.93        30
```

Hình 3. 5 Báo cáo của sóng Spiral trên ResNet50

Classification Report cho sóng lượn (Wave):

```
Classification Report:
              precision    recall  f1-score   support

     healthy       0.82      0.93      0.88        15
   parkinson       0.92      0.80      0.86        15

    accuracy                           0.87        30
   macro avg       0.87      0.87      0.87        30
weighted avg       0.87      0.87      0.87        30
```

Hình 3. 6 Báo cáo của sóng Wave trên ResNet50

Đối với mô hình VGG19:

Classification Report cho sóng cong tròn (Spiral):

```
Classification Report:
              precision    recall  f1-score   support

     healthy       0.71      0.80      0.75        15
   parkinson       0.77      0.67      0.71        15

    accuracy                           0.73        30
   macro avg       0.74      0.73      0.73        30
weighted avg       0.74      0.73      0.73        30
```

Hình 3. 7 Báo cáo của sóng Spiral trên VGG19

Classification Report cho sóng lượn (Wave):

Classification Report:

	precision	recall	f1-score	support
healthy	0.89	0.53	0.67	15
parkinson	0.67	0.93	0.78	15
accuracy			0.73	30
macro avg	0.78	0.73	0.72	30
weighted avg	0.78	0.73	0.72	30

Hình 3. 8 Báo cáo của sóng Wave trên VGG19

Kết quả từ các bản báo cáo cho thấy độ vượt trội khi nhận diện của mô hình ResNet50 so với VGG19 ở cả 2 dạng sóng. Tiến tới việc sử dụng mô hình ResNet50 ứng dụng cho WebApp.

3.2.2. Kết quả huấn luyện cho dữ liệu âm thanh

Classification Report:

Classification Report:

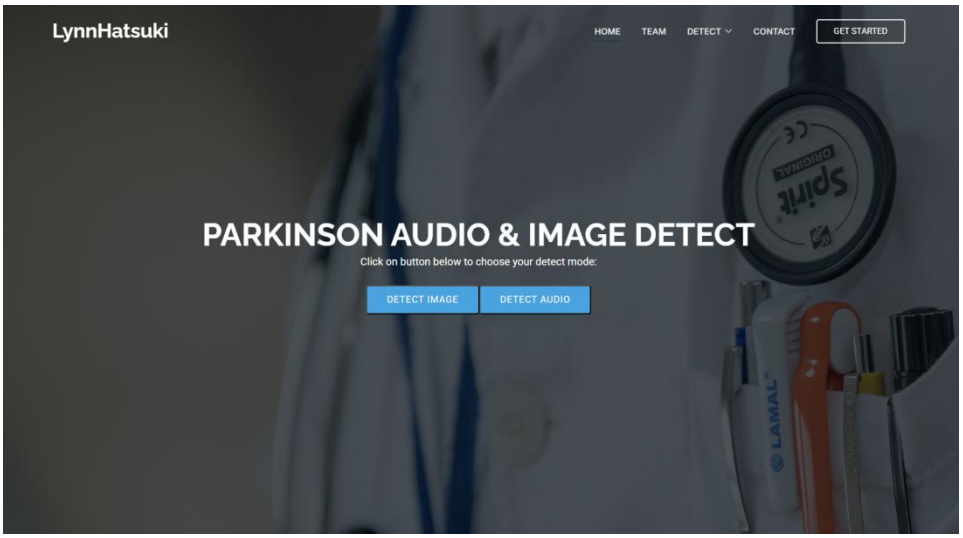
	precision	recall	f1-score	support
0	0.80	0.71	0.75	17
1	0.72	0.81	0.76	16
accuracy			0.76	33
macro avg	0.76	0.76	0.76	33
weighted avg	0.76	0.76	0.76	33

Hình 3. 9 Báo cáo huấn luyện âm thanh với Random Forest

Từ kết quả kiểm thử huấn luyện cho thấy còn nhầm bệnh nhân khỏe mạnh thành bệnh nhân bị bệnh Parkinson. Tuy nhiên tỉ lệ recall của nhãn 1 cao cho thấy mô hình tập trung nhận diện các nhãn bệnh tốt hơn. Nhìn chung, mô hình vẫn chưa thực sự tốt do chỉ số accuracy vẫn còn khá thấp do vấn đề về lượng dữ liệu, khiến mô hình không học được đủ sâu, từ đó độ nhận diện trên tập test cũng không quá cao.

3.3. Thực nghiệm trên Web App

Sau huấn luyện và chọn lựa được mô hình tốt nhất để lưu về máy thì ta tiến hành triển khai Web App để dự đoán. Giao diện được chia làm 2 hướng là một sẽ nhận diện hình ảnh riêng và một sẽ là nhận diện âm thanh. Ở trang chủ khi bắt đầu sẽ hiện ra giao diện 2 nút để người dùng chọn hướng xử dụng, khi nhấn vào nút Detect Image thì web sẽ chuyển hướng người dùng đến với giao diện nhận diện hình ảnh và ngược lại với nhận diện dạng âm thanh:



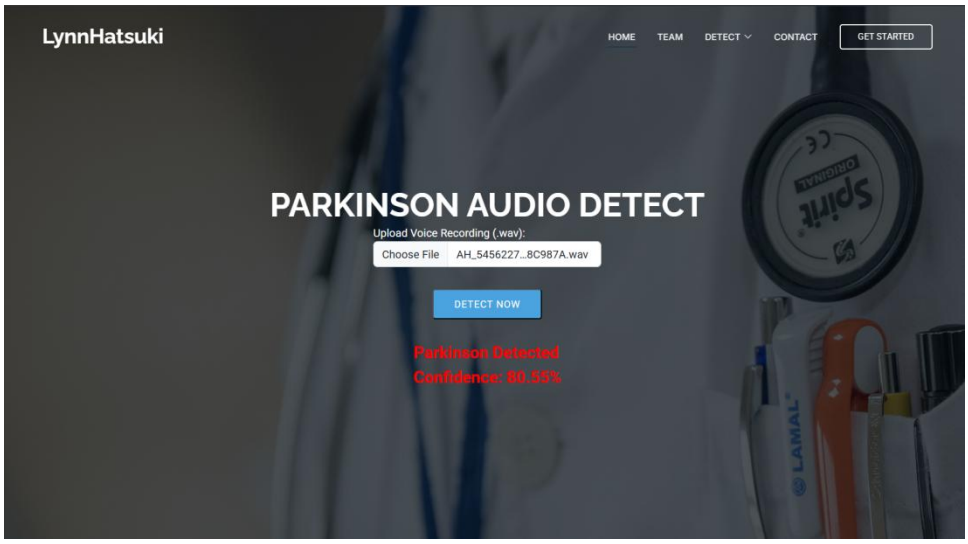
Hình 3. 10 Giao diện trang chủ Web App

Sau khi nhấn chọn vào giao diện nhận diện hình ảnh, người dùng chọn ảnh vẽ dạng sóng hoặc dạng cong tròn lưu dưới định dạng .png hoặc .jpg để tải lên, khi nhấn vào nút Detect Now thì giao diện trả về ảnh Preview của hình vừa được tải và cho kết quả nhận diện bệnh Parkinson:

Hình 3. 11 Giao diện thực nghiệm dự đoán trên ảnh

Hình 3. 12 Giao diện thực nghiệm dự đoán trên ảnh

Với giao diện nhận diện âm thanh thì người dùng cũng tải lên file âm thanh tương tự như bên nhận diện ảnh, file âm thanh được chấp nhận dưới định dạng lưu là .wav:



Hình 3. 13 Giao diện thực nghiệm dự đoán trên âm thanh

NHẬN XÉT VÀ KẾT LUẬN

Trong quá trình làm và phát triển đề tài, em nhận thấy có những ưu và nhược điểm khi làm về nhận diện hình ảnh lẫn âm thanh.

Về ưu điểm:

+ Đối với mô hình nhận diện hình ảnh, kết quả nhận diện trả về phân biệt khá tốt giữa nhãn bệnh và nhãn không bị bệnh, huấn luyện cả 2 mô hình cho thấy sự khác biệt giữa mô hình ResNet50 với kết quả vượt trội trên cả 2 dạng sóng so với VGG19. Ứng dụng khả quan khi thực nghiệm trên Web App.

+ Đối với mô hình nhận diện âm thanh, khi xử lí âm thanh và quá trình huấn luyện dữ liệu với mô hình học máy cho thấy mức độ học của mô hình ở mức khá ổn định, nhận diện ở mức khá khi thực nghiệm trên Web App.

Về nhược điểm:

+ Cả 2 mô hình cho âm thanh và hình ảnh vẫn chưa tối ưu được vấn đề nhầm lẫn từ nhãn người khỏe mạnh sang người bệnh.

+ Vẫn chưa triển khai được trực quan dữ liệu lên Web App để người sử dụng xem rõ được đặc điểm nhận diện bệnh của mình qua âm thanh, giao diện chỉ dừng ở mức nhận diện file âm thanh để đưa ra kết quả dự đoán trên mặt chữ chứ chưa trực quan lên được dữ liệu. Ngoài ra cũng chưa tích hợp được việc ghi âm giọng nói trực tiếp dạng realtime để người sử dụng có thể ghi âm luôn trên nền tảng mà không phải ghi âm sẵn có trên máy rồi đẩy lên để nhận diện.

HƯỚNG PHÁT TRIỂN

Với kết quả thu được từ cả hai dạng mô hình, có thể từ đây phát triển đề tài tốt hơn bằng việc thu thập nhiều dữ liệu có tính chọn lọc cao, tăng cao số lượng dữ liệu để mô hình có thể học được đa dạng dữ liệu hơn, từ đó cải thiện vấn đề về độ chính xác, phù hợp với thực tế ở cả dạng dữ liệu âm thanh và hình ảnh.

Chặng đường tiếp theo hướng đến phát hiện bệnh Parkinson bằng việc thiết kế các thiết bị đo độ run ở tay từ đó có thể nhận diện bệnh tốt hơn thay vì sử dụng hình ảnh vẽ từ những bệnh nhân. Đối với dữ liệu dạng âm thanh thì hướng tới việc cho phép người dùng ghi âm giọng nói trực tiếp trên nền tảng, từ đó có thể nhận diện, sàng lọc âm thanh một cách trực tiếp mà không phải sử dụng dữ liệu có sẵn. Để làm được điều đó cần phải tìm hiểu kỹ lưỡng về mô hình học máy, học sâu để tăng hiệu suất tốc độ dự đoán và độ chính xác, cũng như giải quyết triệt để vấn đề về sàng lọc, lọc nhiễu âm thanh khi ghi âm ở thời gian thực, đây cũng là một trong những thách thức lớn phải đối mặt để tiến tới mô hình tối ưu hơn.

TÀI LIỆU THAM KHẢO

Tiếng Việt:

[1] Nhân, N., Châu, C., Bình, N., & Tắt, T. (2022). RỐI LOẠN GIỌNG NÓI CỦA BỆNH NHÂN PARKINSON TẠI BỆNH VIỆN LÃO KHOA TRUNG ƯƠNG BẰNG THANG ĐIỂM VOICE HANDICAP INDEX. *Vietnam Journal of Physiology*, *26*(2). https://scholar.dlu.edu.vn/thuvienso/bitstream/DLU123456789/190556/1/CVv205V26S22022024.pdf

Tiếng Anh:

[2] Govindu, A., & Palwe, S. (2023). Early detection of Parkinson's disease using machine learning. *Procedia Computer Science*, *218*(1), 249–261. https://doi.org/10.1016/j.procs.2023.01.007

[3] MATLAB. (2017). *Denoise Speech Using Deep Learning Networks - MATLAB & Simulink*. Mathworks.com. https://in.mathworks.com/help/deeplearning/ug/denoise-speech-using-deep-learning-networks.html

[4] Mcfee, B., Raffel, C., Liang, D., Ellis, D., Mcvicar, M., Battenberg, E., & Nieto, O. (2015). librosa: Audio and Music Signal Analysis in Python. In *PROC. OF THE 14th PYTHON IN SCIENCE CONF*. https://proceedings.scipy.org/articles/Majora-7b98e3ed-003.pdf

[5] Prior, F. (2023). Voice Samples for Patients with Parkinson's Disease and Healthy Controls. *Figshare*. https://doi.org/10.6084/u002Fm9.figshare.23849127.v1

[6] Simonyan, K., & Zisserman, A. (2015). *VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION*. https://arxiv.org/pdf/1409.1556

[7] souro12. (2020, July 16). *Parkinsons Syndrome Prediction w/ Resnet CNN*. Kaggle.com; Kaggle. https://www.kaggle.com/code/souro12/parkinsons-syndrome-prediction-w-resnet-cnn