ĐẠI HỌC ĐÀ NẴNG

**TRƯỜNG** **ĐẠI HỌC SƯ PHẠM KỸ THUẬT**

**BÁO CÁO TỔNG KẾT**

**ĐỀ TÀI NGHIÊN CỨU KHOA HỌC CỦA SINH VIÊN**

**ỨNG DỤNG THEO DÕI**

**VÀ PHÁT HIỆN BỆNH LÝ HÔ HẤP**

**SV2021-27**

**Sinh viên thực hiện: Đặng Thành Sơn**

**Lớp/ khoa: 20TDH2**

**Giáo viên hướng dẫn: Ths. Đỗ Hoàng Ngân Mi**

**Đà Nẵng, 6/2021**

ĐẠI HỌC ĐÀ NẴNG

**TRƯỜNG** **ĐẠI HỌC SƯ PHẠM KỸ THUẬT**

**BÁO CÁO TỔNG KẾT**

**ĐỀ TÀI NGHIÊN CỨU KHOA HỌC CỦA SINH VIÊN**

**ỨNG DỤNG THEO DÕI**

**VÀ PHÁT HIỆN BỆNH LÝ HÔ HẤP**

**SV2021-27**

**Xác nhận của cơ quan chủ trì Chủ nhiệm đề tài**

(ký, họ và tên, đóng dấu) (ký, họ và tên)

MỤC LỤC

[DANH MỤC CÁC HÌNH 4](#_Toc74341701)

[DANH MỤC CÁC BẢNG 5](#_Toc74341702)

[CHƯƠNG 1 8](#_Toc74341703)

[TỔNG QUAN VỀ ỨNG DỤNG THEO DÕI VÀ PHÁT HIỆN BỆNH LÝ HÔ HẤP 8](#_Toc74341704)

[1.1 Lý do chọn đề tài 8](#_Toc74341705)

[1.2 Mục tiêu đề tài 9](#_Toc74341706)

[1.3 Phương pháp nghiên cứu 9](#_Toc74341707)

[1.4 Đối tượng và phạm vi nghiên cứu 9](#_Toc74341708)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 10](#_Toc74341709)

[2.1 Tổng quan hệ thống. 10](#_Toc74341710)

[2.2.1 Mô hình DeepLearning và CNN 10](#_Toc74341711)

[2.2.2 Transfer Learning 12](#_Toc74341712)

[2.2.3 Tổng quan Tensorflow 12](#_Toc74341713)

[2.3.4 Tổng quan Module ESC-10 19](#_Toc74341714)

[2.3.4.1 Xây dựng hệ thống nhận dạng giọng nói 20](#_Toc74341715)

[2.3.4.2 Chuyển đổi tín hiệu âm thanh : 21](#_Toc74341716)

[2.3.4.3 Đặc trưng cho tín hiệu âm thanh: 23](#_Toc74341717)

[2.3.4.4 Tín hiệu âm thanh đơn điệu 25](#_Toc74341718)

[2.3.4.5 Tính năng trích xuất từ giọng nói 26](#_Toc74341719)

[2.3.4.6 Nhận biết các từ đã nói 28](#_Toc74341720)

[2.4 Mô hình Training 29](#_Toc74341721)

[2.5 Mô hình ứng dụng 31](#_Toc74341722)

[CHƯƠNG 3: 33](#_Toc74341723)

[THIẾT KẾ, TRIỂN KHAI, PHÂN TÍCH VÀ ĐÁNH GIÁ HỆ THỐNG 33](#_Toc74341724)

[3.1 Thiết kế hệ thống 33](#_Toc74341725)

[3.1.1 Cơ sở dữ liệu (CSDL) 33](#_Toc74341726)

[3.1.2 Trainning 35](#_Toc74341727)

[3.1.2.1 Tiền xử lý dữ liệu: 35](#_Toc74341728)

[3.1.2.2 Tăng cường dữ liệu và điều chỉnh tham số 35](#_Toc74341729)

[3.1.2.3 Quy trình thực hiện 36](#_Toc74341730)

[3.1.2.4 Kết quả 37](#_Toc74341731)

[3.2. Ứng dụng 39](#_Toc74341732)

[3.3. Kết quả 40](#_Toc74341733)

3.4. Đánh giá và phân tích phần mềm 41

[3.5. Kết luận](#_Toc74341734) 42

[TÀI LIỆU THAM KHẢO 4](#_Toc74341735)3

# 

# DANH MỤC CÁC HÌNH

Hình 2.1: Tổng quan hệ thống Ứng dụng theo dõi và phát hiện bệnh lý hô hấp 11

Hình 2.2: Mô hình CNN 12

Hình 2.3: Mô hình Tensorflow 15

Hình 2.4: Báo lỗi của Tensorflow khi khai báo một Tensor không hợp lệ. 18

Hình 2.5: Chuyển đổi sóng âm thanh 20

Hình 2.6: Đồ thị đầu ra 23

Hình 2.7: Biến đổi Fourier 25

Hình 2.8: Tín hiệu âm thanh 26

Hình 2.9: Kết quả a) MFCC và b) bộ lọc Filter bank. 28

Hình 2.10: Mô hình trainning. 30

Hình 2.11: Mô hình ứng dụng. 31

Hình 3.1: Cơ sở dữ liệu a) Trang quản lý 200000 mẫu dữ liệu, b)hình ảnh mẫu dữ liệu thu được, c) phân loại dữ liệu 34

Hình 3.2: Cơ sở dữ liệu a) bình thường, b) không xác định, c) có vấn đề. 34

Hình 3.3: Cơ sở dữ liệu ở khâu tiền xử lý. 35

Hình 3.4: Kiến trúc a) VGG-16, b) X-ception. 37

Hình 3.5:Kết quả training trên kiến trúc VGG-16. 38

Hình 3.6: Kết quả training trên kiến trúc X-ception 38

Hình 3.7: Hình ảnh về ứng dụng trên điện thoại. 40

Hình 3.8: Mã Q-code để truy cập. 41

Hình 3.9: phân loại đối tượng thử nghiệm a) theo tuổi b) theo giới tính 41

Hình 3.10: kiểm tra độ chính xác theo tuổi 42

Hình 3.11: kiểm tra độ chính xác theo giới tính 42

# DANH MỤC CÁC BẢNG

Bảng 1: Phân loại. 35

Bảng 2: Phân loại sử dụng phép biến hình lên ảnh. 36

Bảng 3: Phân loại dữ liệu sau khi sử dụng phép biến hình lên ảnh. 36

Bảng 4: Các thông số dùng để train model. 36

Bảng 5: 2 model sau 300 epoch 38

Bảng 6: Kết quả sau train. 39

ĐẠI HỌC ĐÀ NẴNG

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT**

**THÔNG TIN KẾT QUẢ NGHIÊN CỨU CỦA ĐỀ TÀI**

**1. Thông tin chung:**

**-** Tên đề tài: Ứng dụng theo dõi và phát hiện bệnh lí hô hấp

- Sinh viên thực hiện: Đặng Thành Sơn

- Lớp: 20TDH2 Khoa: Điện-Điện tử

- Người hướng dẫn: Ths. Đỗ Hoàng Ngân Mi

- Cơ quan chủ trì:

- Thời gian thực hiện: từ tháng 1 năm 2021 đến tháng 5 năm 2021

**2. Mục tiêu đề tài:**

Mục tiêu của đề tài là xây dựng ứng dụng trên nền tảng Android và Windows giúp theo dõi và phát hiện sớm các biểu hiện bất thường liên quan đến vấn đề hô hấp thông qua tiếng ho. Ứng dụng có thể hoạt động ngoại tuyến trên một thiết bị hoặc hoạt động trực tuyến đồng bộ trên nhiều thiết bị bằng tài khoản và được đăng tải miễn phí trên Google Play Store.

**3. Tính mới và sáng tạo:**

**-** Sử dụng phương pháp huấn luyện máy mới

- Cơ sở dữ liệu tự thu thập và xử lý

- Ứng dụng hướng đến tối giản, phù hợp với mọi người

**4. Tóm tắt kết quả nghiên cứu:**

Sau một thời gian dài tìm hiểu, nghiên cứu xây dựng ý tưởng và bắt tay vào thực hiện. Tác giả đã hoàn thành đề tài “Ứng dụng theo dõi và phát hiện bệnh lý hô hấp”. Nhìn chung, ứng dụng hoạt động ổn định, phát hiện nhanh và chuẩn đoán chính xác. Tuy nhiên ứng dụng vẫn chưa đáp ứng tốt yêu cầu về hiệu năng cũng như khả năng cập nhật lâu dài và cần được nghiên cứu phát triển thêm.

**5. Đóng góp về mặt kinh tế - xã hội,** **giáo dục và đào tạo, an ninh, quốc phòng và khả năng áp dụng của đề tài:**

Ngày tháng năm

|  |  |
| --- | --- |
| **Người hướng dẫn**  (ký, họ và tên) | **Chủ nhiệm đề tài**  (ký, họ và tên) |
|  |  |

**XÁC NHẬN CỦA TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT**

# CHƯƠNG 1

# TỔNG QUAN VỀ ỨNG DỤNG THEO DÕI VÀ PHÁT HIỆN BỆNH LÝ HÔ HẤP

## Lý do chọn đề tài

Theo thống kê của Bộ y tế [1], trung bình hằng năm nước ta ghi nhận hơn 800.000 trường hợp gặp vấn đề về sức khoẻ liên quan đến hô hấp, 70% trong số đó là đặc biệt nghiêm trọng hoặc tử vong. Đặc biệt trong năm 2020, bệnh viêm phổi Vũ Hán lây lan trên quy mô toàn cầu, gây ra cuộc khủng hoảng thiệt hại cả về người và kinh tế. Với đặc điểm truyền nhiễm qua không khí và nước bọt khi tiếp xúc gần, các bệnh hô hấp dễ bị lây lan và tốc độ lây lan nhanh. Dễ dàng hình thành ổ bệnh trong các gia đình, nhóm người, tập thể. Đặc biệt, hầu hết các bệnh hô hấp đều có những biểu hiện bệnh giai đoạn đầu tương đối giống nhau, gây khó khăn trong việc phát hiện sớm. Theo [1], 95 % bệnh nhân không phân biệt được dấu hiệu của các bệnh hô hấp nghiêm trọng với cảm cúm thông thường. 80% trong số đó tự sử dụng thuốc mà không có chỉ dẫn của bác sĩ và 67% phần trăm đến gặp bác sĩ khi bệnh đã ở giai đoạn nặng. Tiến sĩ Ngô Duy Mạnh giải thích về vấn đề trên: “Do người dân ở nước ta sống trong khí hậu nhiệt đới, đã quen với bệnh cảm cúm dẫn đến tâm lý chủ quan”.

Tuy nhiên, trước những loại biến thể virus đặc biệt là covid-19, thực trạng này là hết sức đáng báo động. Xuất phát từ những thực trạng trên, tác giả đã tiến hành thu thập xấp xỉ 200000 mẫu từ những tình nguyện viên và xây dựng thuật toán nhận dạng các bệnh lý hô hấp dựa trên mẫu chuẩn do Anas elmasry cung cấp trên Kaggle để xây dựng “Ứng dụng theo dõi và phát hiện bệnh lý hô hấp”. Ứng dụng được đăng tải trên Google Play Store để người dùng thuận tiện tải về và sử dụng. Đây sẽ là phương pháp đơn giản, thuận tiện giúp phát hiện sớm các vấn đề bất thường liên quan đến hô hấp.

## 1.2 Mục tiêu đề tài

Xây dựng ứng dụng trên nền tảng Android và Windows giúp theo dõi và phát hiện sớm các biểu hiện bất thường liên quan đến vấn đề hô hấp thông qua tiếng ho. Ứng dụng có thể hoạt động ngoại tuyến trên một thiết bị hoặc hoạt động trực tuyến đồng bộ trên nhiều thiết bị bằng tài khoản và được đăng tải miễn phí trên Google Play Store.

Các yêu cầu kỹ thuật đề ra:

* Độ chính xác.
* Yêu cầu cấu hình thiết bị.
* Ứng dụng thân thiện với người dùng.
* Tạo ra 1 mô hình giúp phát hiện vùng chứa các dấu hiệu bị bệnh hô hấp tình âm thanh đầu vào có tính chính xác (accuracy) cao và hàm mất mát ( LOSS) thấp, đồng thời giá trị của Loss khi training và Loss khi validate model phải gần như song song với nhau và khoảng cách giữa chúng là tạm chấp nhận được ( tránh tình trạng Overfitting và Underfitting).

## Phương pháp nghiên cứu

Tổng hợp cơ sở dữ liệu từ các nguồn thu thập dự liệu.

Đánh giá dựa trên mô hình mẫu Kaggle, so sánh và hoàn thiện mẫu chuẩn.

Thiết kế kết hợp phương pháp CNN trong deeplearning truyền thống.

Mô phỏng sóng âm.

Thực nghiệm.

## 1.4 Đối tượng và phạm vi nghiên cứu

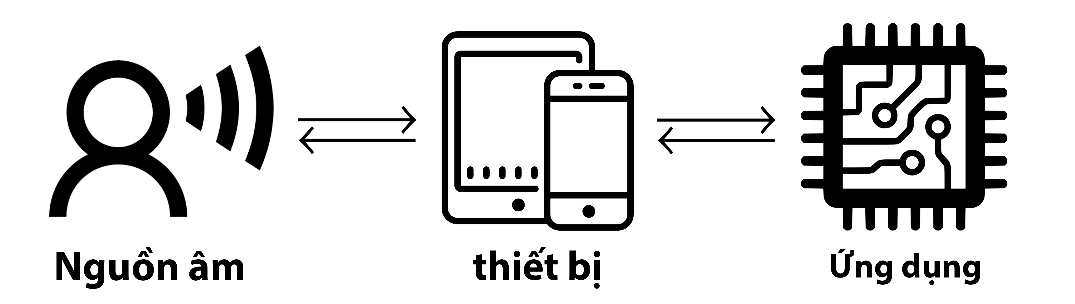
Đối tượng nghiên cứu: Người có biểu hiện bệnh lý hô hấp.

Phạm vi nghiên cứu: Trên mẫu dữ liệu do sinh viên Trường Đại học Sư phạm Kỹ thuật – Đại học Đà Nẵng cung cấp.

# CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

## Tổng quan hệ thống.

Tổng quan về ứng dụng theo dõi và phát hiện bệnh lý hô hấp. Trong đó, ứng dụng được xây dựng dựa trên thuật toán CNN trong Deaplearning. Ứng dụng có thể thu nhận âm thanh từ môi trường. Phát hiện, phân biệt tiếng ho, phân tích và đưa ra những dự đoán về sức khỏe dựa trên tiếng ho đồng thời gửi thông báo phản hồi đến người dùng.

****

Hình 2.1: Tổng quan hệ thống Ứng dụng theo dõi và phát hiện bệnh lý hô hấp.

* 1. **Thiết lập mô hình**

### 2.2.1 Mô hình DeepLearning và CNN

Mạng CNN là một tập hợp các lớp Convolution chồng lên nhau và sử dụng các hàm nonlinear activation như ReLU và tanh để kích hoạt các trọng số trong các node. Mỗi một lớp sau khi thông qua các hàm kích hoạt sẽ tạo ra các thông tin rộng hơn cho các lớp tiếp theo. Mỗi một lớp sau khi thông qua các hàm kích hoạt sẽ tạo ra các thông tin rộng hơn cho các lớp tiếp theo. Trong mô hình mạng truyền ngược (feedforward neural network) thì mỗi neural đầu vào (input node) cho mỗi neural đầu ra trong các lớp tiếp theo. Mô hình này gọi là mạng kết nối đầy đủ (fully connected layer) hay mạng toàn vẹn (affine layer). Còn trong mô hình CNNs thì ngược lại. Các layer liên kết được với nhau thông qua cơ chế convolution. Layer tiếp theo là kết quả convolution từ layer trước đó, nhờ vậy mà ta có được các kết nối cục bộ. Như vậy mỗi neuron ở lớp kế tiếp sinh ra từ kết quả của filter áp đặt lên một vùng ảnh cục bộ của neuron trước đó. Mỗi một lớp được sử dụng các filter khác nhau thông thường có rất nhiều filter như vậy và kết hợp chúng lại. Ngoài ra có một số layer khác như pooling/subsampling layer dùng để chắt lọc lại các thông tin hữu ích hơn (loại bỏ các thông tin nhiễu). Trong quá trình huấn luyện mạng (training) CNN tự động học các giá trị qua các lớp filter dựa vào cách thức mà bạn thực hiện. Ví dụ trong tác vụ phân lớp ảnh, CNNs sẽ cố gắng tìm ra thông số tối ưu cho các filter tương ứng theo thứ tự raw pixel > edges > shapes > facial > high-level features. Layer cuối cùng được dùng để phân lớp ảnh.



Hình 2.2: Mô hình CNN.

Trong mô hình CNN có 2 khía cạnh cần quan tâm là tính bất biến (Location Invariance) và tính kết hợp (Compositionality). Với cùng một đối tượng, nếu đối tượng này được chiếu theo các gốc độ khác nhau (translation, rotation, scaling) thì độ chính xác của thuật toán sẽ bị ảnh hưởng đáng kể. Pooling layer sẽ cho bạn tính bất biến đối với phép dịch chuyển (translation), phép quay (rotation) và phép co giãn (scaling). Tính kết hợp cục bộ cho ta các cấp độ biểu diễn thông tin từ mức độ thấp đến mức độ cao và trừu tượng hơn thông qua convolution từ các filter. Đó là lý do tại sao CNNs cho ra mô hình với độ chính xác rất cao. Cũng giống như cách con người nhận biết các vật thể trong tự nhiên. Mạng CNN sử dụng 3 yếu tố cơ bản:

- Trường tiếp nhận cục bộ (local receptive field). Đầu vào của mạng CNN là một ảnh. Kích thước ảnh tương đương kích thước ma trận và giá trị mỗi điểm ảnh là một ô trong ma trận. Trong mô hình mạng ANN truyền thống thì chúng ta sẽ kết nối các neuron đầu vào vào tầng ảnh. Tuy nhiên trong CNN chúng ta không làm như vậy mà chúng ta chỉ kết nối trong một vùng nhỏ của các neuron đầu vào như một filter. Mỗi một kết nối sẽ học một trọng số và mỗi neuron ẩn sẽ học một bias. Mỗi một vùng đấy gọi là một trường tiếp nhận cục bộ.

- Trọng số chia sẻ (shared weight and bias). Đầu tiên, các trọng số cho mỗi filter (kernel) phải giống nhau. Tất cả các nơ-ron trong lớp ẩn đầu sẽ phát hiện chính xác feature tương tự chỉ ở các vị trí khác nhau trong hình ảnh đầu vào. Chúng ta gọi việc map từ input layer sang hidden layer là một feature map. Một convolutional layer bao gồm các feature map khác nhau. Mỗi một feature map giúp detect một vài feature trong bức ảnh. Lợi ích lớn nhất của trọng số chia sẻ là giảm tối đa số lượng tham số trong mạng CNN.

- Lớp tổng hợp (pooling layer). Lớp pooling thường được sử dụng ngay sau lớp convulational để đơn giản hóa thông tin đầu ra, giảm bớt số lượng neuron. Thủ tục pooling phổ biến là max-pooling, thủ tục này chọn giá trị lớn nhất trong vùng đầu vào 2×2. Như vậy qua lớp Max Pooling thì số lượng neuron giảm đi phân nửa. Trong một mạng CNN có nhiều Feature Map nên mỗi Feature Map chúng ta sẽ cho mỗi Max Pooling khác nhau. Chúng ta có thể thấy rằng Max Pooling là cách hỏi xem trong các đặc trưng này thì đặc trưng nào là đặc trưng nhất. Ngoài Max Pooling còn có L2 Pooling. Cuối cùng ta đặt tất cả các lớp lại với nhau thành một CNN với đầu ra gồm các neuron với số lượng tùy chọn. 2 lớp cuối cùng của các kết nối trong mạng là một lớp đầy đủ kết nối (fully connected layer) . Lớp này nối mọi nơron từ lớp max pooled tới mọi nơron của tầng ra.

## 2.2.2 Transfer Learning

CNN yêu cầu 1 tập dữ liệu lớn để đạt được độ chính xác cao. Transfer learning thường được sử dụng khi mà tập dữ liệu không có đủ để đáp ứng nhu cầu của bài toán. Hơn nữa còn giúp giải quyết vấn đề tài nguyên sử dụng khi train model. Nguyên lý của Transfer learning sử dụng các “weight” đã có sẵn, tức là phần đã được huấn luyện để phân tích và trích xuất các đặc điểm riêng biệt của từng đối tượng mà chúng được huấn luyện trên ( cụ thể là nhận biết các đối tượng quen thuộc trong cuộc sống hàng ngày hoặc các bệnh trên lá… ) và áp dụng những chỉnh sửa (fine tuning) để tiến hành phân loại những đối tượng riêng theo mục đích của chúng ta. Những model được sử dụng để lấy những weight đó gọi là pre-trained model, thường là những model state-of-the-art như DenseNet, ImageNet…

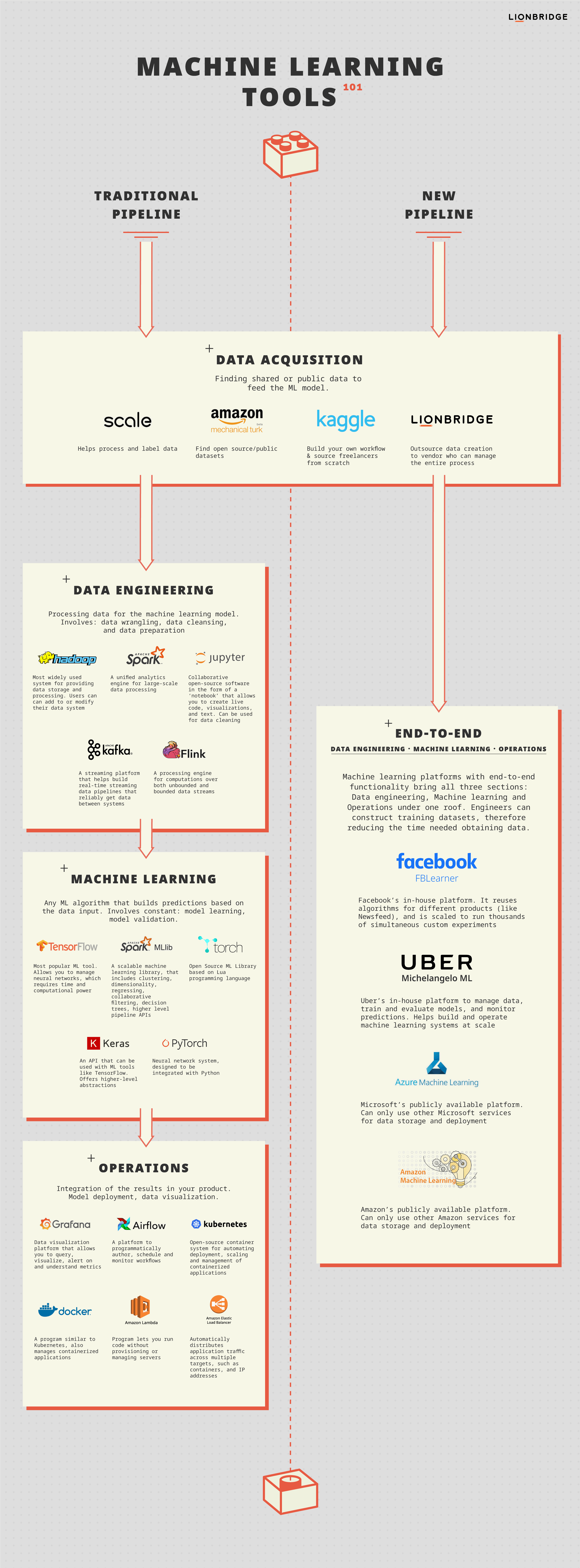
Thay thế lớp fully connected cuối cùng bằng 1 lớp xuất input duy nhất, tại lớp đó áp dụng hàm phi tuyến tính Sigmoid và xuất ra xác suất mà nội dung của ảnh bao hàm.

## 2.2.3 Tổng quan Tensorflow

TensorFlow là thư viện mã nguồn mở cho machine learning, được phát triển bởi Google. Các hàm được dựng sẵn trong thư viện cho từng bài toán cho phép TensorFlow xây dựng được nhiều neural network. Cho phép tính toán song song trên nhiều máy tính khác nhau, thậm chí trên nhiều CPU, GPU trong cùng 1 máy hay tạo ra các dataflow graph – đồ thị luồng dữ liệu để dựng nên các model. Được viết bằng C++ và thao tác interface bằng Python nên phần performance của TensorFlow cực kỳ tốt. TensorFlow cho phép tạo ra dataflow graph, cấu trúc mô tả làm thế nào dữ liệu có thể di chuyển qua 1 biểu đồ, hay 1 sê-ri các node đang xử lý. Mỗi node trong đồ thị đại diện 1 operation toán học và mỗi kết nối hay edge giữa các node là 1 mảng dữ liệu đa chiều, hay còn được gọi là ‘tensor’. Node và tensor trong TensorFlow là các đối tượng Python, và các ứng dụng TensorFlow cũng là các ứng dụng Python. Các operation toán học thực sự thì không được thi hành bằng Python. Các thư viện biến đổi có sẵn thông qua TensorFlow được viết bằng các binary C++ hiệu suất cao. Python chỉ điều hướng lưu lượng giữa các phần và cung cấp các high-level abstraction lập trình để nối chúng lại với nhau.

**Node**

Vì Tensorflow mô tả lại dòng chảy của dữ liệu thông qua graph nên mỗi một điểm giao cắt trong graph thì được gọi là Node. Tại sao điều này quan trọng thì là vì các Node chính là điểm đại diện cho việc thay đổi của dữ liệu nên việc lưu trữ lại tham chiếu của các Node này là rất quan trọng.



Hình 2.3: Mô hình [TensorFlow](https://www.tensorflow.org/).

**Tensor**

Như trong bài viết trước mình có đề cập, để giải được các bài toán Machine Learning, cần phải làm cho máy tính có thể hiểu được dữ liệu của tập nguồn và dữ liệu của tập đích. Tensorflow cung cấp một loại dữ liệu mới được gọi là Tensor. Trong thế giới của Tensorflow, mọi kiểu dữ liệu đều được quy về một mối được gọi là Tensor hay trong Tensorflow, tất cả các loại dữ liệu đều là Tensor. Vậy nên có thể hiểu được phần nào cái tên Tensorflow là một thư viện mô tả, điều chỉnh dòng chảy của các Tensor.

Tensor là một kiểu dữ liệu dạng mảng có nhiều chiều được mô tả dạng Tensor = [[[1,1,1],[178,62,74]],[[45,2,2],[19,0,17]],[[7,5,2],[0,11,4]],[[8,13,5],[1,6,7]]]. Mảng nhiều chiều này được đính kèm thêm một vài thuộc tính tham chiếu khác. Các thuộc tính của Tensor được mô tả trong tài liệu bao gồm:

* **device**: Tên của thiết bị mà Tensor hiện tại sẽ được xuất bản. Có thể None.
* **graph**: Đồ thị chứa Tensor hiện tại.
* **name**: Tên của Tensor hiện tại.
* **shape**: Trả về TensorShape mô tả lại Shape của Tensor hiện tại.
* **op**: Phép toán được sử dụng để xuất bản Tensor hiện tại.
* **dtype**: Kiểu của các phần tử trong Tensor hiện tại.

**Rank**

Rank là bậc hay độ sâu của một Tensor.

Ví dụ như Tensor = [1] sẽ có rank là 1, Tensor = [[[1,1,1],[178,62,74]]] sẽ có rank bằng 3, Tensor = [[1,1,1],[178,62,74]] sẽ có rank bằng 2. Cách nhanh nhất để xác định rank của một Tensor là đếm số lần mở ngoặc vuông cho đến giá trị khác ngoặc vuông đầu tiên. Việc phân rank này khá quan trọng vì nó đồng thời cũng giúp phân loại dữ liệu của Tensor. Khi ở cách rank đặc biệt cụ thể, Tensor có những tên gọi riêng như sau:

* **Scalar**: Khi Tensor có rank bằng 0, Tensor đại diện cho một số hoặc một chuỗi cụ thể. Ví dụ: scalar = 123.
* **Vector**: Vector là một Tensor rank 1. Trong python thì Vector là một mảng một chiều chứa các số. Ví dụ: list = [123,345].
* **Matrix**: Đây là một Tensor rank 2 hay mảng hai chiều theo khái niệm của Python. Ví dụ: matrix = [[1,2],[2,1]].
* **N-Tensor**: Khi rank của Tensor tăng lên lớn hơn 2, chúng được gọi chung là N-Tensor.

**Shape**

Shape là một tuple có số chiều bằng với rank của Tensor tương ứng dùng để mô tả lại cấu trúc của Tensor đó. Dưới đây là ví dụ về Shape.

* Tensor = 1 sẽ có Shape = ().
* Tensor = [1] sẽ có Shape = (1).
* Tensor = [[[1,1,1],[178,62,74]]] sẽ có Shape = (1,1,3).
* Tensor = [[1,1,1],[178,62,74]] sẽ có Shape = (1,3).

Dựa vào cấu trúc của Shape, ta dễ dàng thấy rằng ràng buộc cơ bản của Tensor là chiều của các elements trong Tensor tại mỗi bậc phải bằng nhau. Dưới đây là ví dụ báo lỗi của Tensorflow.

>>> test = tf.constant([[1], [1,2]])

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File"/usr/local/lib/python2.7/dist-packages/tensorflow/python/framework/constant\_op.py", line 102, in constant

tensor\_util.make\_tensor\_proto(value, dtype=dtype, shape=shape, verify\_shape=verify\_shape))

File"/usr/local/lib/python2.7/dist-packages/tensorflow/python/framework/tensor\_util.py", line 383, in make\_tensor\_proto

\_GetDenseDimensions(values)))

ValueError: Argument must be a dense tensor: [[1], [1, 2]] - got shape [2], but wanted [2, 1].

>>> test = tf.constant([[1,2], [1]])

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File"/usr/local/lib/python2.7/dist-packages/tensorflow/python/framework/constant\_op.py", line 102, in constant

tensor\_util.make\_tensor\_proto(value, dtype=dtype, shape=shape, verify\_shape=verify\_shape))

File"/usr/local/lib/python2.7/dist-packages/tensorflow/python/framework/tensor\_util.py", line 383, in make\_tensor\_proto

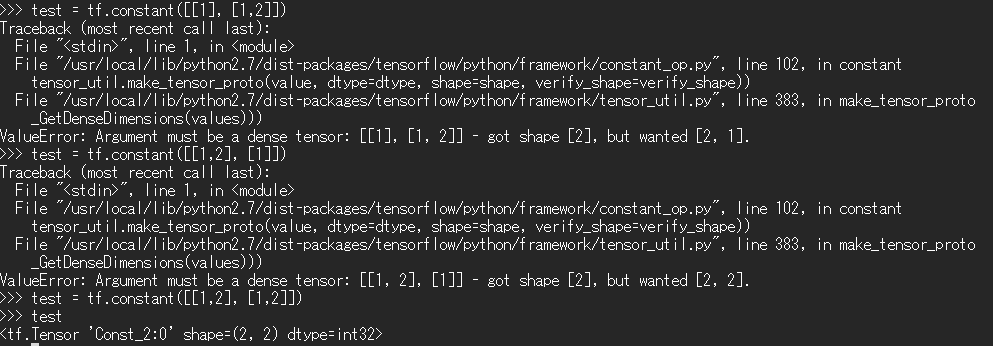
\_GetDenseDimensions(values)))

ValueError: Argument must be a dense tensor: [[1, 2], [1]] - got shape [2], but wanted [2, 2].

>>> test = tf.constant([[1,2], [1,2]])

>>> test

<tf.Tensor 'Const\_2:0' shape=(2, 2) dtype=int32>

  
 Hình 2.4: Báo lỗi của Tensorflow khi khai báo một Tensor không hợp lệ.

**Op**

Được viết tắt là op, khái niệm Operator là toán tử được dùng để thực thi Tensor tại node đó. Các toán tử này có thể là hằng số, biến số, phép cộng, phép nhân... Đôi khi mình cảm thấy việc dịch là toán tử cũng không hợp lí bởi lẽ các toán tử này đôi khi lại mô tả Node là Constant hay Variable. Có thể nói, khái niệm operator trong Tensorflow là khái niệm dùng để mô tả lại trạng thái của Node nói chung.

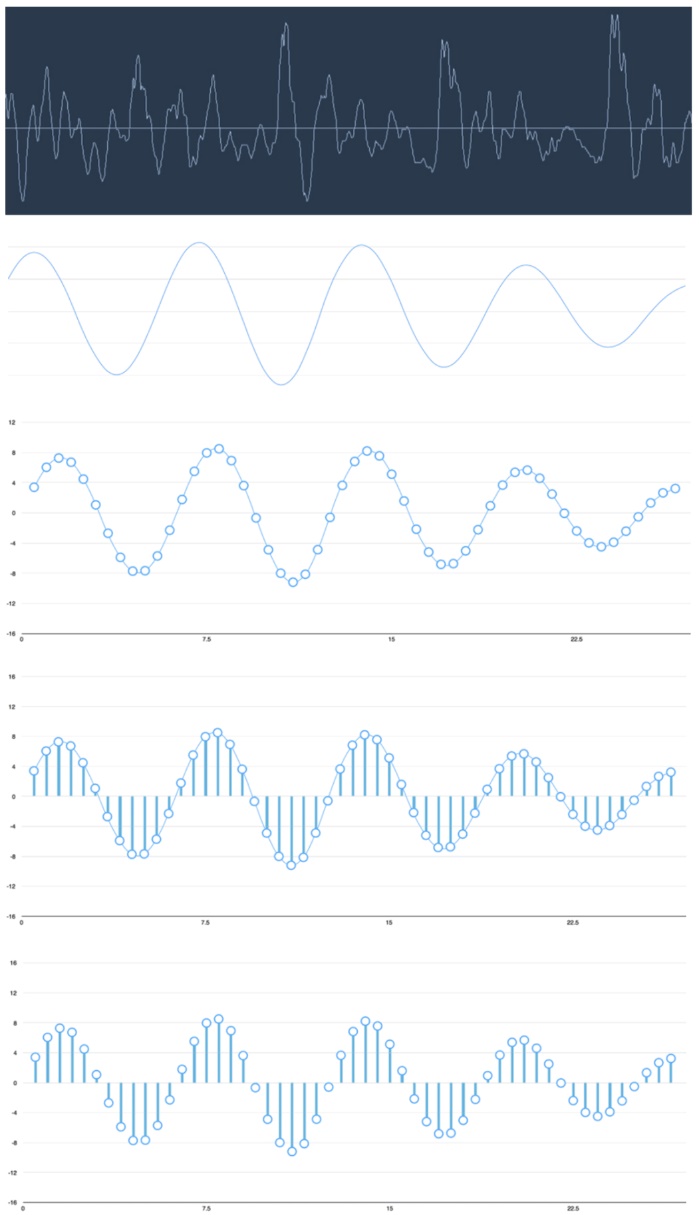
**DType**

Đây là kiểu dữ liệu của các elements trong Tensor. Vì một Tensor chỉ có duy nhất một thuộc tính DType nên từ đó cũng suy ra là chỉ có duy nhất một kiểu DType duy nhất cho toàn bộ các elements có trong Tensor hiện tại. Việc tạo ra hơn một DType khác nhau cho các elements của Tensor là không khả dụng. Hiện có hack hay trick nào chưa thì mình chưa rõ nhưng kể cả khi bạn thực hiện các phép toán thì cũng không thể làm điều đó được.

### 2.3.4 Tổng quan Module ESC-10

Nhận dạng giọng nói đề cập đến công nghệ chuyển đổi sóng âm thanh thành một mô tả cụ thể, thường bao gồm nhận dạng giọng nói, nhận dạng dấu giọng nói và nhận dạng cảnh âm thanh. Công nghệ nhận dạng giọng nói hiện tại chủ yếu là học máy, một số phương pháp trong học sâu như GMM, CNN, RNN, v.v.

Chuyển đổi âm thanh thành số, sóng âm có **một chiều** dữ liệu. Ở mỗi thời điểm, chúng có một giá trị cao độ. Để chuyển sóng âm thành số, cần ghi lại độ cao của sóng ở từng khoảng. Phương pháp này gọi là **sampling -** lấy mẫu. Chúng ta đọc mẫu mỗi 1/1000s và ghi lại con số đại diện chiều cao của sóng âm. Đây chính là file .wav khi không bị nén. Những âm thanh chất lượng tốt được ghi ở tần số 44.1khz (44,100 lần đọc mỗi giây). Nhưng với nhận diện giọng nói, tốc độ lấy mẫu là 16khz (16,000 mẫu mỗi giây). Bắt đầu nhóm mẫu âm thanh trong khoảng 20ms. Ghi lại những con số này trong đồ thị giúp ước lượng xấp xỉ về âm thanh gốc trong chu kỳ 20ms, kết quả cuối cùng là một bảng số thể hiện độ năng lượng của mỗi khoảng tần số, từ âm thấp tới âm cao. Mỗi số đại diện cho năng lượng dải 50hz trong 20ms. Truyền từng dải âm 20ms vào mạng nơron đa lớp. Với mỗi mảng cắt âm thanh chứa đặc điểm đại diện cho âm thanh phát ra.



Hình 2.5: Chuyển đổi sóng âm thanh.

### 2.3.4.1 Xây dựng hệ thống nhận dạng giọng nói

Những yêu cầu quan trọng

* **Kích thước của từ vựng**: ảnh hưởng đến sự dễ dàng của việc phát triển một ASR (kích thước từ vựng càng lớn thì việc nhận dạng càng khó).
* **Đặc điểm của channel** - Chất lượng channel cũng là một yếu tố quan trọng. Ví dụ, lời nói của con người có băng thông cao với dải tần đầy đủ, trong khi lời nói qua điện thoại bao gồm băng thông thấp với dải tần hạn chế. Lưu ý rằng nó khó hơn trong phần sau.
* **Chế độ nói**- Việc phát triển ASR dễ dàng cũng phụ thuộc vào chế độ nói, đó là liệu bài phát biểu có ở chế độ từ riêng biệt, hoặc chế độ từ được kết nối hay ở chế độ nói liên tục. Lưu ý rằng một bài phát biểu liên tục khó nhận ra hơn.
* **Phong cách nói**- Bài phát biểu được đọc có thể theo phong cách trang trọng, hoặc tự phát và đối thoại với phong cách bình thường. Cái sau khó nhận ra hơn.
* **Sự phụ thuộc người nó**i - Lời nói có thể phụ thuộc vào người nói, sự thích ứng của người nói hoặc độc lập với người nói. Rất khó nhất để xây dựng một diễn giả độc lập
* **Loại tiếng ồn** - Tiếng ồn là một yếu tố khác cần xem xét khi phát triển ASR. Tỷ lệ tín hiệu trên tiếng ồn có thể nằm trong nhiều phạm vi khác nhau, tùy thuộc vào môi trường âm thanh quan sát ít hơn so với nhiều tiếng ồn xung quanh   
  - Nếu tỷ lệ tín hiệu trên nhiễu lớn hơn 30dB, nó được coi là dải cao  
  - Nếu tỷ lệ tín hiệu trên nhiễu nằm trong khoảng từ 30dB đến 10db, nó được coi là SNR trung bình

- Nếu tỷ lệ tín hiệu trên nhiễu nhỏ hơn 10 dB, nó được coi là dải tần thấp

* **Đặc điểm của micrô** - Chất lượng của micrô có thể tốt, trung bình hoặc dưới trung bình. Ngoài ra, khoảng cách giữa miệng và micro phone có thể khác nhau. Các yếu tố này cũng cần được xem xét đối với hệ thống công nhận.

## 2.3.4.2 Chuyển đổi tín hiệu âm thanh :

**Recording :**

Đọc tín hiệu âm thanh từ một tệp, sau đó ghi lại bằng micrô.

**Sampling :**

Khi ghi âm bằng micrô, các tín hiệu được lưu trữ dưới dạng số hóa. Nhưng để hoạt động, máy cần chúng ở dạng số rời rạc. Do đó, chúng ta nên thực hiện lấy mẫu ở một tần số nhất định và chuyển đổi tín hiệu thành dạng số rời rạc. Việc chọn tần số cao để lấy mẫu ngụ ý rằng khi con người nghe tín hiệu, họ cảm thấy nó như một tín hiệu âm thanh liên tục.

Sử dụng Python, được lưu trữ trong một tệp. Tần số của tín hiệu âm thanh này là 44.100 HZ.

**import** numpy as np

**import** matplotlib.pyplot as plt

from scipy.io **import** wavfile

Đọc tệp âm thanh được lưu trữ. Nó sẽ trả về hai giá trị: tần số lấy mẫu và tín hiệu âm thanh. Cung cấp đường dẫn của tệp âm thanh nơi nó được lưu trữ

frequency\_sampling, audio\_signal = wavfile.read("/Users/admin/audio\_file.wav")

Hiển thị các thông số như tần số lấy mẫu của tín hiệu âm thanh, loại dữ liệu của tín hiệu và thời lượng của nó

print('\nSignal shape:', audio\_signal.shape)

print('Signal Datatype:', audio\_signal.dtype)

print('Signal duration:', round(audio\_signal.shape[0] /

float(frequency\_sampling), 2), 'seconds')

Chuẩn hóa tín hiệu

audio\_signal = audio\_signal / np.power(2, 15)

Trích xuất 100 giá trị đầu tiên từ tín hiệu này để hình dung.

audio\_signal = audio\_signal [:100]

time\_axis = 1000 \* np.arange(0, len(signal), 1) / float(frequency\_sampling)

Hình dung tín hiệu

**plt**.plot(time\_axis, signal, color='blue')

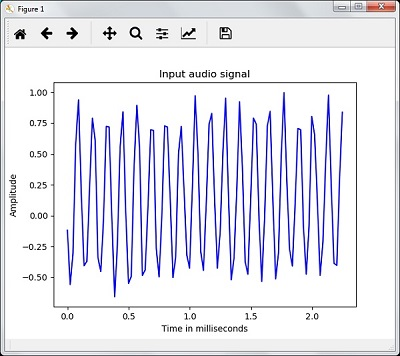
**plt**.xlabel('Time (milliseconds)')

**plt**.ylabel('Amplitude')

**plt**.title('Input audio signal')

**plt**.show()

Đồ thị đầu ra và dữ liệu được trích xuất cho tín hiệu âm thanh



Hình 2.6: Đồ thị đầu ra.

Signal shape: (132300,)

Signal Datatype: int16

Signal duration: 3.0 seconds

## 2.3.4.3 Đặc trưng cho tín hiệu âm thanh:

**Chuyển đổi sang dãy tần số**

Đặc trưng cho một tín hiệu âm thanh liên quan đến việc chuyển đổi tín hiệu miền thời gian thành miền tần số và hiểu các thành phần tần số của nó. Đây là bước quan trọng vì nó cho biết nhiều thông tin về tín hiệu. sử dụng một công cụ toán học như Fourier Transform để thực hiện phép biến đổi này.

Từng bước, cách mô tả đặc tính của tín hiệu, sử dụng Python, được lưu trữ trong một tệp. Lưu ý rằng ở đây mình đang sử dụng công cụ toán học Fourier Transform để chuyển đổi nó thành miền tần số.

**import** numpy as np

**import** matplotlib.pyplot as plt

from scipy.io **import** wavfile

Đọc tệp âm thanh được lưu trữ. Nó sẽ trả về hai giá trị: tần số lấy mẫu và tín hiệu âm thanh. Cung cấp đường dẫn của tệp âm thanh nơi nó được lưu trữ

frequency\_sampling, audio\_signal = wavfile.read("/Users/admin/sample.wav")

Hiển thị các thông số như tần số lấy mẫu của tín hiệu âm thanh, loại dữ liệu của tín hiệu và thời lượng của nó

print('\nSignal shape:', audio\_signal.shape)

print('Signal Datatype:', audio\_signal.dtype)

print('Signal duration:', round(audio\_signal.shape[0] /

float(frequency\_sampling), 2), 'seconds')

Chuẩn hóa tín hiệu

audio\_signal = audio\_signal / np.power(2, 15)

Trích xuất độ dài và nửa độ dài của tín hiệu :

length\_signal = len(audio\_signal)

half\_length = np.ceil((length\_signal + 1) / 2.0).astype(np.int)

Áp dụng các công cụ toán học để biến đổi thành miền tần số. Sử dụng Biến đổi Fourier.

signal\_frequency = np.fft.fft(audio\_signal)

Thực hiện chuẩn hóa tín hiệu miền tần số và bình phương nó

signal\_frequency = abs(signal\_frequency[0:half\_length]) / length\_signal

signal\_frequency \*\*= 2

Trích xuất độ dài và nửa độ dài của tín hiệu được biến đổi tần số :

len\_fts = len(signal\_frequency)

Lưu ý rằng tín hiệu biến đổi Fourier phải được điều chỉnh cho trường hợp chẵn cũng như lẻ.

if length\_signal % 2:

signal\_frequency[1:len\_fts] \*= 2

else:

signal\_frequency[1:len\_fts-1] \*= 2

Trích xuất công suất bằng decibel (dB) -

signal\_power = 10 \* np.log10(signal\_frequency)

Điều chỉnh tần số tính bằng kHz cho trục X -

x\_axis = np.arange(0, len\_half, 1) \* (frequency\_sampling / length\_signal) / 1000.0

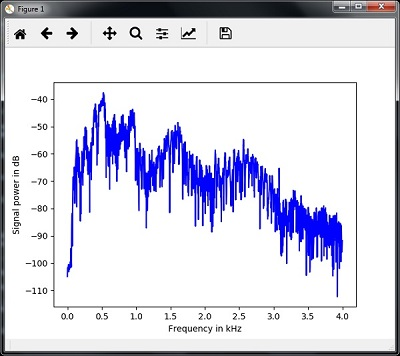
**plt**.figure()

**plt**.plot(x\_axis, signal\_power, color='black')

**plt**.xlabel('Frequency (kHz)')

**plt**.ylabel('Signal power (dB)')

**plt**.show()



Hình 2.7: Biến đổi Fourier.

## 2.3.4.4 Tín hiệu âm thanh đơn điệu

Tạo tín hiệu âm thanh với một số thông số được xác định trước. Lưu ý rằng bước này sẽ lưu tín hiệu âm thanh trong tệp đầu ra.

Tạo một tín hiệu đơn điệu, sử dụng Python, sẽ được lưu trữ trong một tệp

**import** numpy as np

**import** matplotlib.pyplot as plt

from scipy.io.wavfile **import** write

Cung cấp tệp nơi tệp đầu ra sẽ được lưu

output\_file = 'audio\_signal\_generated.wav'

Chỉ định các tham số

duration = 4 # in seconds

frequency\_sampling = 44100 # in Hz

frequency\_tone = 784

min\_val = -4 \* np.pi

max\_val = 4 \* np.pi

Tạo ra tín hiệu âm thanh

t = np.linspace(min\_val, max\_val, duration \* frequency\_sampling)

audio\_signal = np.sin(2 \* np.pi \* tone\_freq \* t)

Lưu tệp âm thanh trong tệp đầu ra :

**write**(output\_file, frequency\_sampling, signal\_scaled)

Trích xuất 100 giá trị đầu tiên cho biểu đồ

audio\_signal = audio\_signal[:100]

time\_axis = 1000 \* np.arange(0, len(signal), 1) / float(sampling\_freq)

Hình dung tín hiệu âm thanh được tạo ra

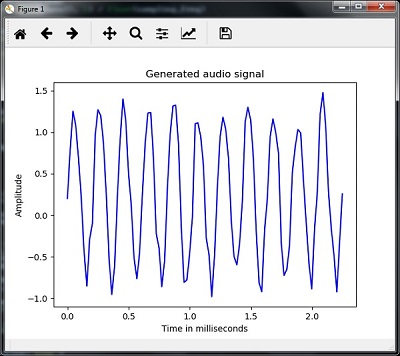
**plt**.plot(time\_axis, signal, color='blue')

**plt**.xlabel('Time in milliseconds')

**plt**.ylabel('Amplitude')

**plt**.title('Generated audio signal')

**plt**.show()



Hình 2.8: Tín hiệu âm thanh.

## 2.3.4.5 Tính năng trích xuất từ giọng nói

Đây là bước quan trọng nhất trong việc xây dựng bộ nhận dạng giọng nói vì sau khi chuyển đổi tín hiệu giọng nói sang miền tần số, chuyển nó thành dạng đặc trưng có thể sử dụng được. Có thể sử dụng các kỹ thuật trích xuất tính năng khác nhau như MFCC, PLP, PLP-RASTA, v.v. cho mục đích này.

Trích xuất các tính năng từ tín hiệu, từng bước, sử dụng Python, bằng cách sử dụng kỹ thuật MFCC.

**import** numpy as np

**import** matplotlib.pyplot as plt

from scipy.io **import** wavfile

from python\_speech\_features **import** mfcc, logfbank

Đọc tệp âm thanh được lưu trữ. Nó sẽ trả về hai giá trị - tần số lấy mẫu và tín hiệu âm thanh. Cung cấp đường dẫn của tệp âm thanh nơi nó được lưu trữ.

frequency\_sampling, audio\_signal = wavfile.read("/Users/admin/audio\_file.wav")

Lưu ý rằng ở đây đang lấy 15000 mẫu đầu tiên để phân tích.

audio\_signal = audio\_signal[:15000]

Sử dụng các kỹ thuật của MFCC và thực hiện lệnh sau để trích xuất các tính năng của MFCC

features\_mfcc = mfcc(audio\_signal, frequency\_sampling)

In các tham số MFCC như sau :

print('\nMFCC:\nNumber of windows =', features\_mfcc.shape[0])

print('Length of each feature =', features\_mfcc.shape[1])

Vẽ features MFCC như sau :

**features\_mfcc** = features\_mfcc.T

plt.matshow(features\_mfcc)

plt.title('MFCC')

Làm việc với filter bank features , đầu tiên cần phải extract filtter bank features :

filterbank\_features = logfbank(audio\_signal, frequency\_sampling)

In các đối số filter bank

print('\nFilter bank:\nNumber of windows =', filterbank\_features.shape[0])

print('Length of each feature =', filterbank\_features.shape[1])

Vẽ và visualize filter bank features :

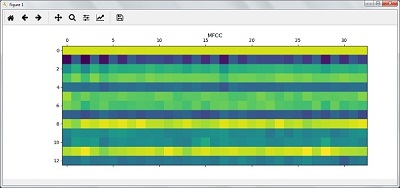
**filterbank\_features** = filterbank\_features.T

plt.matshow(filterbank\_features)

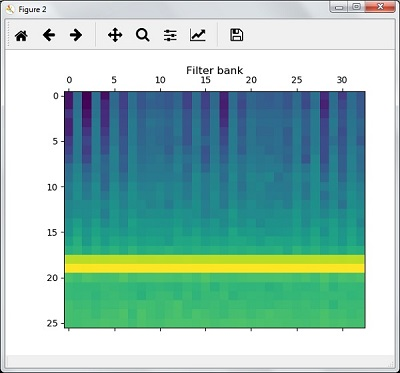
plt.title('Filter bank')

plt.show()

Theo kết quả của các bước trên, quan sát các kết quả đầu ra sau:

F

a)



b)

Hình 2.9: Tín hiệu qua a) MFCC và b) bộ lọc Filter bank.

## 2.3.4.6 Nhận biết các từ đã nói

Sử dụng Google Speech API bằng Python

* **Pyaudio :** pip install Pyaudio
* **SpeechRecognition :**pip install SpeechRecognition.
* **Google-Speech-API :** pip install google-api-python-client

**import** speech\_recognition as sr

Tạo object

recording = sr.Recognizer()

mô-đun Micrô () sẽ lấy giọng nói làm đầu vào :

**with** sr.Microphone() **as** **source**: recording.adjust\_for\_ambient\_noise(**source**)

print("Please Say something:")

audio = recording.listen(**source**)

Google API sẽ nhận dạng giọng nói và đưa ra đầu ra.

**try**:

print("You said: \n" + recording.recognize\_google(audio))

**except** Exception **as** e:

print(e)

Output;

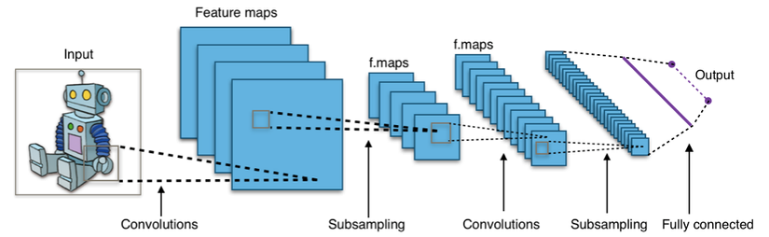
Please Say Something:

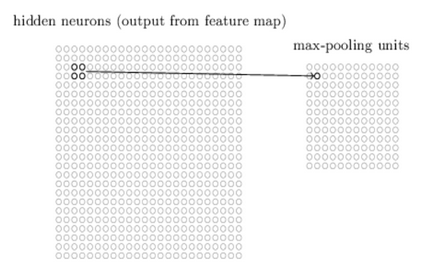
You said:

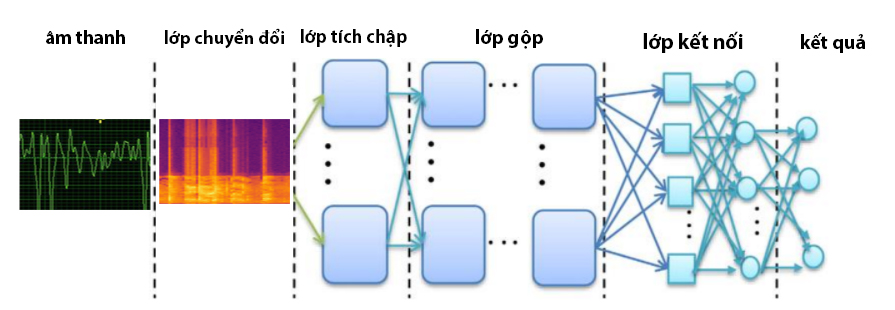
## 2.4 Mô hình Training

Như đã đề cập ở trên, CNN đang dần trở nên phổ biến để cải thiện hiệu suất và độ chính xác cho bài toán phân loại nhị phân cho ảnh. Các lớp chập (convolution) cùng với các bộ lọc (filter) giúp trích xuất ra các đặc điểm đặc trưng bên trong hình ảnh. Các layer có nhiệm lưu trữ các thông tin về đặc điểm đặc trưng – weight giúp cho việc tái sử dụng các “kinh nghiệm” đã học từ các model khác nhằm giảm thiểu thời gian cũng như yêu cầu về độ lớn của dataset.

CNNs là 1 cải tiến của Artificial Neural Networks(ANNs) với 2 ràng buộc: các neuron trong cùng 1 layer có được thông qua việc tính tích chập bằng các kỹ thuật slide với các padding riêng biệt được kết nối với nhau và output của layer này sử dụng làm input cho layer tiếp theo để giảm thiểu số tham số của model và giữ kích thước không gian ( thông số độ dài, độ rộng, độ sâu). 1 mạng CNN gồm 3 phần chính: i)Các lớp tích chập (convolutional) để xác định feature thông qua việc kích hoạt các neuron. ii) Lớp pooling thường được đặt sau các lớp tích chập để đơn giản hóa thông tin đầu ra và giảm bớt số lượng neuron thông qua việc thiết lập các stride có kích thước giống nhau và lấy ra tham số lớn nhất và iii) Fully-connected layer và hàm softmax để tính toán và trả ra xác suất nội dung mà hình ảnh input đại diện.





****

Hình 2.10: Mô hình trainning.

- Lớp chuyển đổi: chuyển đổi âm thành đầu vào thành hình ảnh dựa trên cao độ, sử dụng phổ nhiệt độ màu để biểu diễn, tốc độ lấy mẫu 16khz.

- Lớp tích chập: 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

có kích thước 8x8 px và có độ sâu bằng với độ sâu của đầu vào. Bằng cách trượt

dần bộ lọc theo chiều ngang và dọc trên ảnh, thu được một Feature Map chứa các

đặc trưng được trích xuất từ trên hình ảnh đầu vào.

- Lớp gộp: được dùng giữa các convolutional layer để giảm kích thước dữ liệu

nhưng vẫn giữ được các thuộc tính quan trọng bằng cách chỉ chọn ra thuộc tính

nổi bật nhất ở mỗi block. 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.

- Lớp kết nối đầy đủ: sau khi ảnh được truyền qua nhiều convolutional layer và

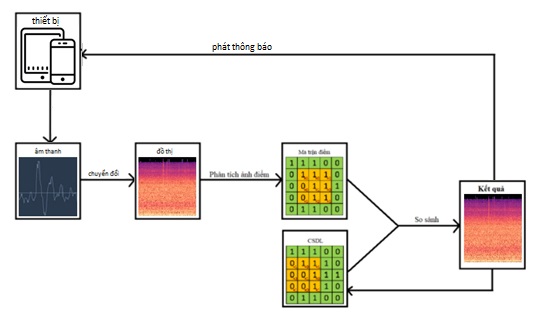
pooling layer thì model đã học được các đặc điểm của ảnh. Tensor output của layer

cuối cùng sẽ được chuyển thành vector và đưa vào một layer được kết nối như một

mạng nơ-ron. Với nhiều vector kết hợp tại layer tạo thành một mô hình. Cuối cùng

sử dụng softmax hoặc sigmoid để phân loại đầu ra.

## 2.5 Mô hình ứng dụng

****

Hình 2.11: Mô hình ứng dụng.

Ứng dụng xây dựng trên nền tảng java, hỗ trợ android 5.0 trở lên. Ứng dụng sẽ phát thông báo đến người dùng khi phát hiện dấu hiệu bất thường trong tiếng ho. Nhằm hướng đến sự tối giản trong sử dụng, ứng dụng không có giao diện tương tác, ứng dụng sẽ hoạt động ngầm ngay sau khi cài đặt và được cấp quyền. Ứng dụng yêu cầu thiết bị có chức năng thu âm và có thể hoạt động ngay cả trong các chế độ hạn chế (chế độ máy bay, chế độ tiết kiệm pin). Ứng dụng được hướng đến phát triển thành 2 phiên bản gồm ứng dụng cơ bản (chỉ bao gồm 1 số ma trận điểm mẫu cố định) và ứng dụng mở rộng (bao gồm chức năng tự học và bổ sung ).

# CHƯƠNG 3:

# THIẾT KẾ, TRIỂN KHAI, PHÂN TÍCH VÀ ĐÁNH GIÁ HỆ THỐNG

## 3.1 Thiết kế hệ thống

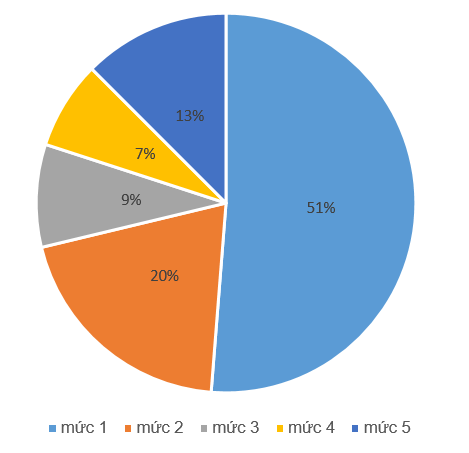
## 3.1.1 Cơ sở dữ liệu (CSDL)



a)

****

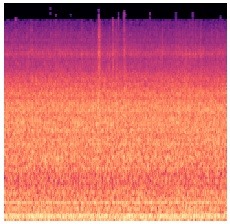
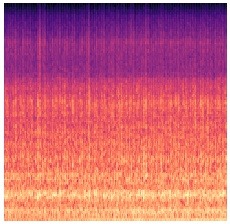
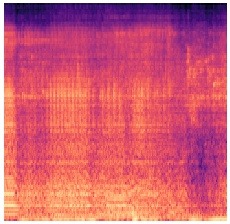
b)



c)

Hình 3.1: Cơ sở dữ liệu a) Trang quản lý 20000 mẫu dữ liệu, b)hình ảnh mẫu dữ liệu thu được, c) phân loại dữ liệu.

Một Page được tạo để phục vụ việc thu thập dữ liệu do các tình nguyện viên cung cấp, sau hơn 2 tháng hoạt động, page đã nhận được hơn 4000 giờ voidchat. Sau khi sử dụng công cụ mẫu chuẩn do Anas elmasry cung cấp trên Kaggle đã lọc ra được hơn 1200 giờ âm thanh là tiếng ho có thể sử dụng. Tiếp tục sử dụng công cụ trên để chia 1200 giờ âm thanh thành 200000 đoạn âm thanh có độ dài khoảng 20 giây, sau đó những đoạn âm thanh này được mã hóa và phân thành 2 nhóm bình thường và không bình thường.



1. b) c)

Hình 3.2: Cơ sở dữ liệu a) bình thường, b) không xác định, c) có vấn đề.

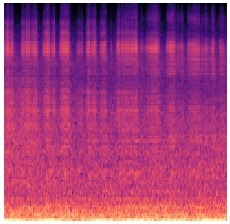
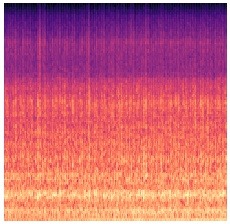
## 3.1.2 Trainning

## 3.1.2.1 Tiền xử lý dữ liệu:

Các pretrained model luôn được train trước từ các hình ảnh vuông có độ dài không gian bằng nhau (chiều dài = chiều cao). Và mỗi kiến trúc mạng khác nhau (ResNet, DenseNet, VGG) lại có kích thước không gian đầu vào khác nhau.

Dữ liệu trong dataset được trải dài trong khoảng 2916 x 2583 pixels (lớn nhất) đến 384 x 127 pixels (nhỏ nhất).

Vì vậy, để giữ nguyên thông tin thông tin thu được có từ bức ảnh và thỏa mãn yêu cầu về dữ liệu đầu vào của mỗi loại kiến trúc mạng, dữ liệu đầu vào phải được thay đổi kích thước dựa trên kiến trúc mạng được chọn. (Xception: 299x299x3, VGG16: 224x224x3).



Hình 3.3: Cơ sở dữ liệu ở khâu tiền xử lý.

## 3.1.2.2 Tăng cường dữ liệu và điều chỉnh tham số

Loại bỏ những bức ảnh được gán nhãn “Không bình thường” nhưng không mắc vấn đề về hô hấp”. Sau khi xử lý cho ra kết quả như sau:

Bảng 1: Phân loại.

|  |  |  |
| --- | --- | --- |
| Nhãn | Target | Số lượng |
| Có vấn đề | 1 | 180555 |
| Bình thường | 0 | 20851 |

Để tránh việc model bị overfitting (hệ thống có xu hướng phán đoán lệch hẳn về 1 lớp nội dung) do có sự chênh lệch quá lớn về số lượng dữ liệu của mỗi lớp nội dung, vậy nên việc tăng cường số lượng dữ liệu là điều cần thiết. Theo như chúng ta thấy thì tỉ lệ hình ảnh mang nhãn “không bình thường”/ hình ảnh mang nhãn “Bình thường” là xấp xỉ 1.079

Tuy nhiên, để tăng độ chính xác của model, tác giả đã thực hiện tăng cường để tạo thêm dữ liệu bằng các phương pháp sử dụng các phép biến hình lên ảnh:

Bảng 2: Phân loạisử dụng phép biến hình lên ảnh.

|  |  |  |
| --- | --- | --- |
| Phương pháp | Thông số | Mô tả |
| Xoay ảnh | 90 | Xoay trục Oy của bức ảnh theo các thông số ngẫu nhiên từ 1-90o với độ lệch ± 0.1o. |
| Lật ngang |  | Lật ngược bức ảnh 180o |
| Phóng ảnh |  | Phóng to thu nhỏ bức ảnh ngẫu nhiên trong khoảng 10% nhưng vẫn giữ nguyên kích thước ban đầu. |

Sau khi áp dụng các phương pháp tăng cường dữ liệu, bộ dữ liệu thu được như sau:

Bảng 3: Phân loại dữ liệu sau khi sử dụng phép biến hình lên ảnh.

|  |  |  |
| --- | --- | --- |
| Nhãn | Target | Số lượng |
| Có vấn đề | 1 | 300356 |
| Bình thường | 0 | 90473 |

## 3.1.2.3 Quy trình thực hiện

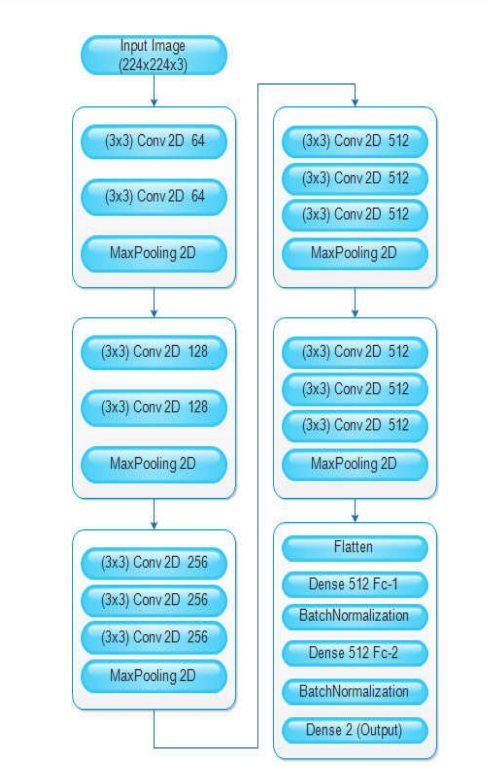
Chia dữ liệu theo tỉ lệ 80-20 cho việc training/testing model.

Sau đó tiến hành sử dụng kĩ thuật transfer learning lên 2 kiến trúc mô hình, được thể hiện qua 2 sơ đồ dưới đây.

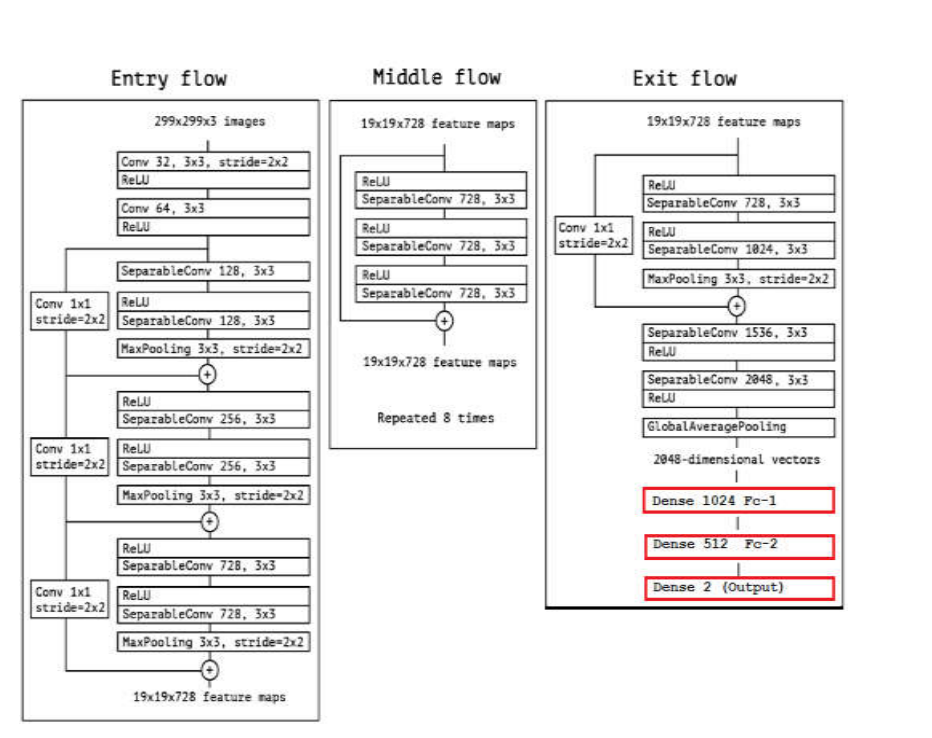
Bảng 4:Các thông số dùng để train model.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Cost function | Learning rate | Optimizer | Số epoch | Kích thước Batch | Các công cụ dùng thêm |
| Cross Entroy (Đã được đề cập ở trên). | 1x10-3 | Adam  Beta1=0,9  Beta2=0,999 | 300 | 32 | Decay |

Tiến hành thử nghiệm trên 2 kiến trúc là X-ception và VGG-16 theo 2 lược đồ



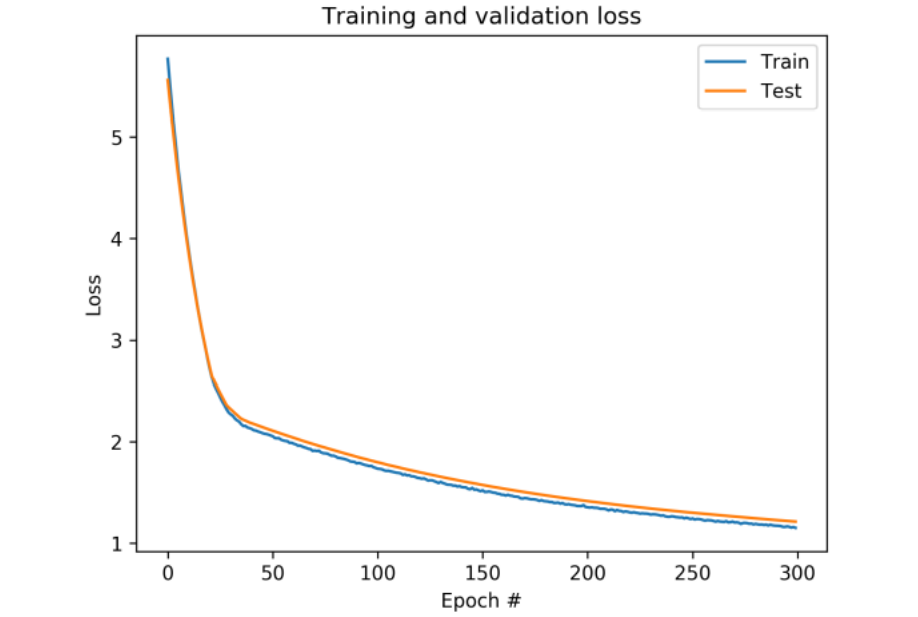
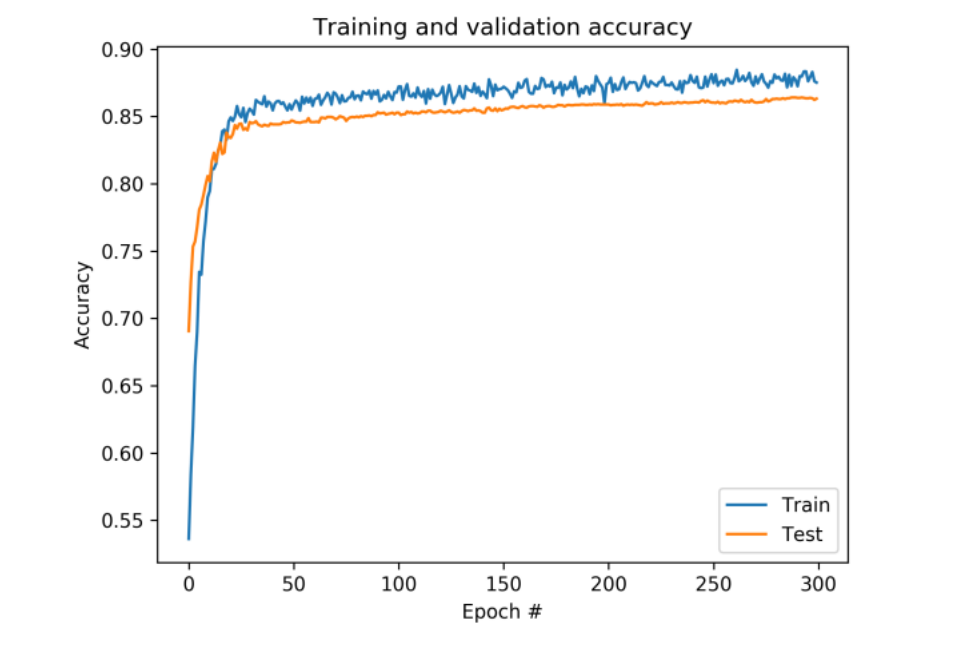
a)



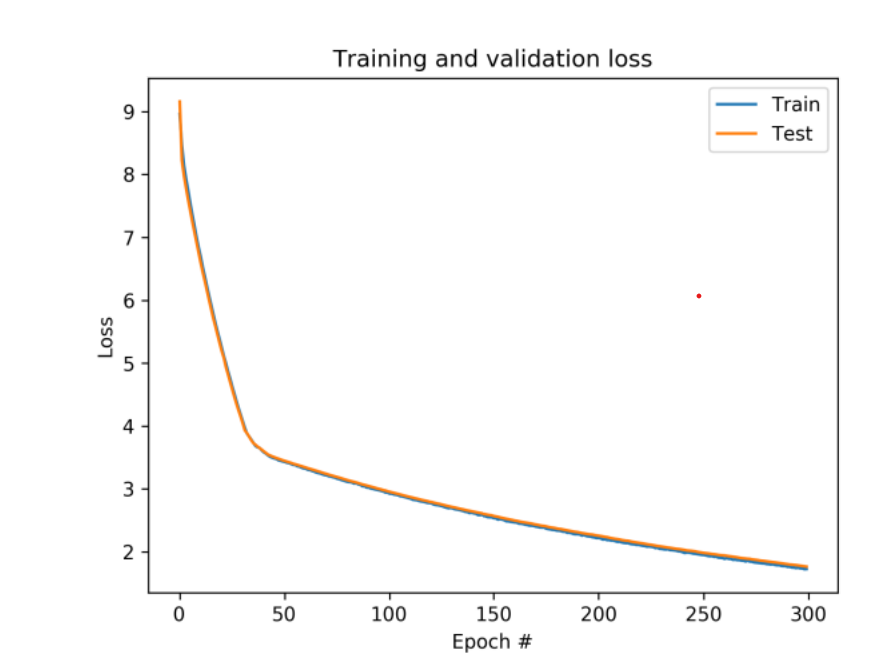
b)

Hình 3.4: Kiến trúc a) VGG-16, b) X-ception.

## 3.1.2.4 Kết quả



Hình 3.5:Kết quả training trên kiến trúc VGG-16.



Hình 3.6: Kết quả training trên kiến trúc X-ception

Bắt đầu từ những epoch model, cả 2 model đều có kết quả dự đoán với độ chính xác khá thấp (1 điều thường thấy ở những model sử dụng transfer learning) do sự khác biệt về đối tượng đích (Model gốc sử dụng hình ảnh tự nhiên còn đối tượng của chúng ta là những hình ảnh x-quang có sự khác biệt rõ ràng về mặt quang học và hình thái học).

Độ chính xác thu được ở 2 model sau 300 epoch, thử nghiệm ở tập dữ liệu test:

Bảng 5: 2 model sau 300 epoch

|  |  |  |
| --- | --- | --- |
| Model | Accuracy | LOSS |
| VGG-16 | 87,73% | 0,034 |
| X-ception | 95,93% | 0,027 |

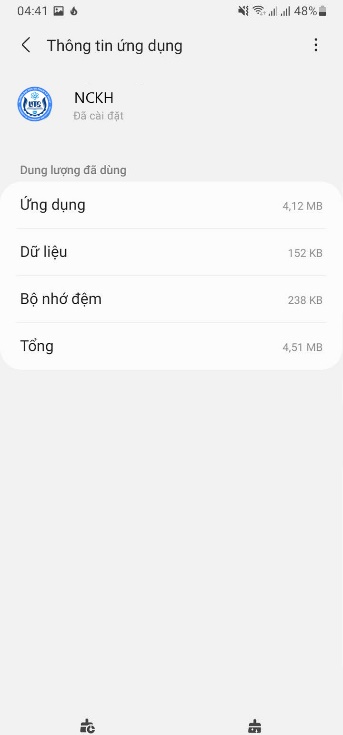
Kết quả: Sau khi train thu được tệp kết quả là một mô hình mẫu có kích thước 8x5539754, độ chính xác 87%.

Bảng 6: Kết quả sau train.

|  |  |
| --- | --- |
| **Seed** | 42 |
| **Figsize** | 8,8 |
| **Imgsize** | 1280 |
| **Batchsize** | 8 |
| **Trainepochs** | 50 |
| **Sizeepochs** | 128 |
| **Nch** | 3 |
| **Densae** | 15 |

## 3.2. Ứng dụng

** **

** **

Hình 3.7: Hình ảnh về ứng dụng trên điện thoại.

Ứng dụng hoạt động trên android, chỉ nặng hơn 4mb và chiếm khoảng 110mb ram

****

Hình 3.8: Mã Q-code để truy cập.

## 3.3. Kết quả

Sau một thời gian tìm hiểu, nghiên cứu xây dựng ý tưởng và bắt tay vào thực hiện. Tác giả đã hoàn thành đề tài “Ứng dụng theo dõi và phát hiện bệnh lý hô hấp”. Nhìn chung, ứng dụng hoạt động ổn định, phát hiện nhanh và chuẩn đoán chính xác. Tuy nhiên ứng dụng vẫn chưa đáp ứng tốt yêu cầu về hiệu năng cũng như khả năng cập nhật lâu dài và cần được nghiên cứu phát triển thêm.

**3.4. Đánh giá và phân tích phần mềm**

1. b)

Hình 3.9: Phân loại đối tượng thử nghiệm a) theo tuổi b) theo giới tính

Hình 3.10: kiểm tra độ chính xác theo tuổi

Hình 3.11: kiểm tra độ chính xác theo giới tính

So sánh độ chính xác của công cụ Kaggle và ứng dụng do tác giả phát triển bằng cách phân nhóm và kiểm tra trên 20000 mẫu dữ liệu, ứng dụng có độ chính xác trung bình cao hơn ( 87% - 76% ). Ứng dụng có độ chính xác ổn định trên các nhóm đối tượng khác nhau (~87%)

## 3.5. Kết luận

Đề tài “Ứng dụng theo dỏi và phát hiện bệnh lý hô hấp” đã cơ bản đạt được nhưng mục tiêu đề ra ban đầu. Ứng dụng hoạt động với độ tin cậy cao, sử dụng ít tài nguyên của thiết bị và thân thiện với người dùng. Tuy nhiên, do hoạt động dựa trên những gói dịch vụ của google, ứng dụng cần liên tục được cập nhật. Đây cũng là hạn chế lớn nhất của đề tài và sẽ được khắc phục.

# TÀI LIỆU THAM KHẢO

1. Thống kê của google về tình trạng và thiệt hại do Covid19, dữ liệu từ các nguồn như Wikipedia, bộ y tế các nước, New York Times và những nguồn đáng tin cậy khác, như được ghi tên nguồn. Cập nhật lần cuối 15/12/2020.
2. Tổng quan về xét nghiệm Covid19, đăng bởi vietnamese.cdc.gov, trích dẫn nội dung từ trung tâm quốc gia về chủng ngừa và bệnh hô hấp (ncird), phân ban bệnh do virus ngày 7/12/2002.
3. Tổng quan mạng tích chập CNN, Phạm Văn Chung, đăng trên báo ViBlo.Asia ngày 12/10/2020.
4. Paul Mooney. Developer Advocate at Kaggle. Boulder, Colorado, United States.
5. Bachir. Chief Kitchen Officer at Couscous Inc. El Oued, El Oued Province, Algeria.
6. Tổng quan về Convolutional Neural Network, Trần Đức Trung, đăng trên báo Viblo.Asis ngày 18/11/2020.
7. Tổng quan PythonFlask và MySQL, Jayrai, đăng trên envatotuts ngày 16/5/2015