LỜI CẢM ƠN

Lời đầu tiên con xin cảm ơn bố mẹ đã sinh thành nuôi dưỡng, luôn tạo điều kiện cho con được học hành đầy đủ, luôn ở bên cạnh quan tâm hỗ trợ những lúc con gặp bế tắc trong cuộc sống.

Với tình cảm chân thành, em cũng xin bày tỏ lòng biết ơn sâu sắc tới giảng viên, PGS. TS Phạm Văn Cường, người đã tận tình hướng dẫn em trong quá trình làm đồ án này. Những lời của thầy trong thời gian qua đã giúp em có thêm kiến thức học thuật cũng như kinh nghiệm trong cuộc sống.

Cuối cùng em xin gửi lời cảm ơn tới toàn thể các thầy, cô trong Học viện Công nghệ Bưu chính Viễn thông, những người luôn tâm huyết với nghề đã dìu dắt, truyền đạt tri thức cho em trong suốt quãng thời gian học đại học. Đây sẽ là những hành trang giúp em bước vào tương lai.

Hà nội, tháng 12 năm 2019

Sinh viên

Nguyễn Tuấn Thành

MỤC LỤC

[LỜI CẢM ƠN i](#_Toc27502971)

[MỤC LỤC ii](#_Toc27502972)

[DANH SÁCH CÁC TỪ VIẾT TẮT iv](#_Toc27502973)

[DANH SÁCH BẢNG vi](#_Toc27502974)

[DANH MỤC HÌNH ẢNH vii](#_Toc27502975)

[MỞ ĐẦU 1](#_Toc27502976)

[CHƯƠNG 1. TỔNG QUAN 3](#_Toc27502977)

[1.1. Tổng quan về nghe âm phổi 3](#_Toc27502978)

[1.1.1 Cấu trúc nhịp thở 3](#_Toc27502979)

[1.1.2 Các loại âm phổi bình thường 4](#_Toc27502980)

[1.1.3 Các loại âm phổi bất bình thường 4](#_Toc27502981)

[1.2. Các nghiên cứu liên quan về phát hiện tiếng thở khò khè. 5](#_Toc27502982)

[1.2.1 Phân tích tự động tiếng phổi 5](#_Toc27502983)

[1.2.2 Hệ thống phát hiện hơi thở khò khè dựa trên Entropy 10](#_Toc27502984)

[1.2.3 Nghiên cứu so sánh các thuật toán học máy Svm và k-nn để chẩn đoán các bệnh lý hô hấp bằng cách sử dụng âm thanh phổi 16](#_Toc27502985)

[1.2.4 Phát hiện tiếng thở khò khè sử dụng wavelets và mạng noron 19](#_Toc27502986)

[1.3. Học sâu 20](#_Toc27502987)

[1.3.1 Convolutional Neural Network 24](#_Toc27502988)

[1.3.2 Recurrent neural network 30](#_Toc27502989)

[1.3.3 Mạng LSTM 33](#_Toc27502990)

[1.4. Tổng kết chương 1 37](#_Toc27502991)

[CHƯƠNG 2. MÔ HÌNH HỌC SÂU PHÁT HIỆN TIẾNG THỞ KHÒ KHÈ 38](#_Toc27502992)

[2.1. Tiền xử lý dữ liệu 38](#_Toc27502993)

[2.2. Trích xuất đặc trưng 40](#_Toc27502994)

[2.2.1 Spectrogram 40](#_Toc27502995)

[2.2.2 Mel-frequency cepstral coefficients 42](#_Toc27502996)

[2.2.3 Gammatone frequency cepstral coefficients (GFCC) 47](#_Toc27502997)

[2.3. Huấn luyện mô hình học sâu cho phát hiện tiếng thở khò khè 49](#_Toc27502998)

[2.3.1 Chuẩn hóa các đặc trưng 49](#_Toc27502999)

[2.3.2 Hàm mất mát và phương pháp tối ưu trên mạng học sâu 49](#_Toc27503000)

[2.3.3 Huấn luyện mạng CNN cơ bản (Convolution Neural Network) 51](#_Toc27503001)

[2.3.4 Huấn luyện mô hình học sâu LSTM tự động phát hiện tiếng thở khò khè 56](#_Toc27503002)

[2.4. Tích hợp vào thiết bị di động 58](#_Toc27503003)

[2.5. Tổng kết chương 2 60](#_Toc27503004)

[CHƯƠNG 3. THỰC NGHIỆM VÀ ĐÁNH GIÁ 62](#_Toc27503005)

[3.1. Thu thập và gán nhãn dữ liệu 62](#_Toc27503006)

[3.1.1 Thu thập dữ liệu 62](#_Toc27503007)

[3.1.2 Gán nhãn dữ liệu 63](#_Toc27503008)

[3.2. Xử lý dữ liệu 63](#_Toc27503009)

[3.3. Phương pháp và các độ đo đánh giá 66](#_Toc27503010)

[3.4. Kết quả và đánh giá 67](#_Toc27503011)

[3.4.1 Kết quả với mạng CNN 67](#_Toc27503012)

[3.4.2 Kết quả với mạng LSTM 69](#_Toc27503013)

[3.4.3 Đánh giá kết quả thực nghiệm 75](#_Toc27503014)

[3.5. Tổng kết chương 3 76](#_Toc27503015)

[CHƯƠNG 4. TỔNG KẾT 77](#_Toc27503016)

[TÀI LIỆU THAM KHẢO 79](#_Toc27503017)

DANH SÁCH CÁC TỪ VIẾT TẮT

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Từ viết tắt** | **Tiếng Anh** | **Tiếng Việt** |
| 1 | AI | Artificial Intelligence | Trí tuệ nhân tạo |
| 2 | AWGN | Additive white Gaussian noise | Thêm nhiễu gaussian trắng |
| 3 | BCE | Binary Cross Entropy | Entropy chéo nhị phân |
| 4 | CNN | Convolutional neural network | Mạng noron tích chập |
| 5 | COPD | Chronic obstructive pulmonary disease | Bệnh phổi tắc nghẽn mạn tính |
| 6 | DCT | Discrete Cosine Transform | Biến đổi Cosin rời rạc |
| 7 | DFT | Discrete Fourier transform | Biến đổi fourier rời rạc |
| 8 | DL | Deep learning | Học sâu |
| 9 | EBWD | Entropy bases wheeze detection | Phát hiện tiếng khò khè dựa trên Entropy |
| 10 | ERB | Equivalent Rectangular Bandwidth | Băng thông hình chữ nhật tương đương |
| 11 | EWA | Exponential Weighted Average | Trung bình lũy thừa |
| 12 | FC | Fully Connected | Kết nối đầy đủ |
| 13 | FFT | Fast Fourier Transform | Biến đổi Fourier nhanh |
| 14 | GFCC | Gammatone frequency cepstral coefficients | Hệ số cepstral tần số gammatone |
| 15 | GPU | Graphics processing unit | Bộ xử lý đồ họa |
| 16 | KNN | K-nearest neighbor | K láng giềng gần nhất |
| 17 | LSTM | Long short-term memory | Mạng bộ nhớ dài-ngắn |
| 18 | MFCC | Mel Frequency Cepstral Coefficient | Hệ số cepstral tần số Mel |
| 19 | ML | Machine learning | Học máy |
| 20 | MLP | Multi-layer Perceptron | Mạng Perceptron nhiều lớp |
| 21 | PDA | A personal digital assistant | Thiết bị kỹ thuật số hỗ trợ cá nhân |
| 22 | Relu | Rectified linear unit | Đơn vị tuyến tính chỉnh lưu |
| 23 | RNN | Recurrent Neural Network | Mạng noron hồi quy |
| 24 | SNR | Signal-to-noise ratio | Tỉ số tín hiệu cực đại trên nhiễu |
| 25 | STFT | Short-time Fourier transform | Biến đổi fourier thời gian ngắn |
| 26 | SVM | Support Vector Machine | Máy vector hỗ trợ |

DANH SÁCH BẢNG

[Bảng 1.1 So sánh hiệu năng các bộ phân loại trên crackles 10](#_Toc27497274)

[Bảng 1.2 Ma trận nhầm lẫn trong phân loại các file âm thanh đầy đủ [6] 10](#_Toc27497275)

[Bảng 1.3 Kết quả với SVM sử dụng RBF kernel [6] 10](#_Toc27497276)

[Bảng 3.1 Số lượng các loại âm phổi thu thập được 63](#_Toc27497277)

[Bảng 3.2 Số lượng các mẫu âm phổi mở rộng 66](#_Toc27497278)

[Bảng 3.3 Giá trị Accuracy MCC của Spectrogram huấn luyện trên mạng CNN 69](#_Toc27497279)

[Bảng 3.4 Giá trị Accuracy và MCC của đặc trưng MFCC huấn luyện trên mạng LSTM 71](#_Toc27497280)

[Bảng 3.5 Giá trị Accuracy và MCC với đặc trưng GFCC được huấn luyện trên mạng LSTM 73](#_Toc27497281)

[Bảng 3.6 Giá trị Accuracy và MCC với kết hợp đặc trưng MFCC GFCC được huấn luyện trên mạng LSTM 75](#_Toc27497282)

DANH MỤC HÌNH ẢNH

[Hình 1.1 Quy trình đánh dữ liệu của Tromso Study [6] 6](#_Toc27497287)

[Hình 1.2 Quy trình tạo ra bộ dữ liệu trong [6] 7](#_Toc27497288)

[Hình 1.3 Mỗi cửa sổ được đặc trưng bởi một vector 5 chiều [6] 8](#_Toc27497289)

[Hình 1.4 F1 scores trung bình cho từng đặc trưng, đường kẻ dọc đánh dấu F1 score cho việc đoán ngẫu nhiên [6]. 9](#_Toc27497290)

[Hình 1.5 Phần cứng của hệ thống phát hiện tiếng thở khò khè dựa trên entropy [8] 11](#_Toc27497291)

[Hình 1.6 Sơ đồ khối của phương pháp tự động phát hiện tiếng khò khè dựa trên entropy [8] 11](#_Toc27497292)

[Hình 1.7 Tín hiệu ở miền thời gian (a) Tín hiệu tiếng thở bình thường (b) Tín hiệu ở tiếng thở khò khè [8]. 12](#_Toc27497293)

[Hình 1.8 Tín hiệu ở miền tần số (a) Tín hiệu của tiếng thở bình thường (b) Tín hiệu của tiếng thở khò khè [8]. 12](#_Toc27497294)

[Hình 1.9 Tín hiệu sau khi thực hiện masking (a) Tín hiệu của tiếng thở bình thường (b) Tín hiệu của tiếng thở khò khè [8]. 13](#_Toc27497295)

[Hình 1.10 Smoothed Entropies(a) Smoothed Entropies của tiếng thở bình thường(b)Smoothed entropies của tiếng thở khò khè [8]. 14](#_Toc27497296)

[Hình 1.11 Kết quả phân loại của phương pháp EBWD [8]. 15](#_Toc27497297)

[Hình 1.12 Kết quả phân loại của phương pháp EBWD dưới các nhiễu khác nhau [8]. 15](#_Toc27497298)

[Hình 1.14 Tín hiệu miền thời gian cho các nhóm Bình thường (b) Tắc nghẽn đường thở (c) Bệnh lý nhu mô [9]. 16](#_Toc27497299)

[Hình 1.15 Lược đồ trích xuất đặc trưng MFCC [9]. 18](#_Toc27497300)

[Hình 1.16 Mối quan hệ giữa AI, học máy và deep learning [13] 21](#_Toc27497301)

[Hình 1.17 Hiệu năng của deep learning và lượng dữ liệu sử dụng để huấn luyện [14] 22](#_Toc27497302)

[Hình 1.18 Số lượng dữ liệu khổng lồ trong thời đại số [15] 22](#_Toc27497303)

[Hình 1.19 Hàm sigmoid và đạo hàm của hàm sigmoid [16] 23](#_Toc27497304)

[Hình 1.20 Sự khác nhau giữa Machine Learning và Deep Learning [17] 23](#_Toc27497305)

[Hình 1.20 Sự biến đổi dữ liệu khi đi qua mạng CNN [19] 25](#_Toc27497306)

[Hình 1.21 Minh hoạ phép tính tích chập trong mạng CNN [20] 26](#_Toc27497307)

[Hình 1.22 Đồ thị hàm ReLU [21] 27](#_Toc27497308)

[Hình 1.23 Minh hoạ dữ liệu trước (trái) và sau (phải) khi áp dụng Max Pooling [23] 28](#_Toc27497309)

[Hình 1.24 Kiến trúc mạng RNN [31] 31](#_Toc27497310)

[Hình 1.25 Diễn giải chi tiết kiến trúc mạng RNN [31] 31](#_Toc27497311)

[Hình 1.26 Minh hoạ sự phụ thuộc gần trong mạng RNN [31] 32](#_Toc27497312)

[Hình 1.27 Minh hoạ sự phụ thuộc xa trong mạng RNN [31] 33](#_Toc27497313)

[Hình 1.28 Chi tiết kiến trúc của mô-đun đơn giản trong mạng RNN [31] 33](#_Toc27497314)

[Hình 1.29 Chi tiết kiến trúc mô-đun trong mạng LSTM [31] 34](#_Toc27497315)

[Hình 1.30 Các kí hiệu sử dụng trong lược đồ LSTM [31] 34](#_Toc27497316)

[Hình 1.31 Trạng thái các cell trong mạng LSTM [31] 34](#_Toc27497317)

[Hình 1.32 Forgate gate trong LSTM [31] 35](#_Toc27497318)

[Hình 1.33 Input gate Layer và ứng cử viên sẽ được cập nhật vào cell Ct [31] 35](#_Toc27497319)

[Hình 1.34 Thực thi quá trình quên và cập nhật thông tin mới trong LSTM [31] 36](#_Toc27497320)

[Hình 1.35 Tính toán đầu ra cho mạng LSTM [23] 36](#_Toc27497321)

[Hình 2.1 Phân tích một chu kì thở [34] 39](#_Toc27497322)

[Hình 2.2 Biểu diễn một số mẫu âm thanh trên miền thời gian 39](#_Toc27497323)

[Hình 2.3 Spectrogram của một mẫu thở khò khè 41](#_Toc27497324)

[Hình 2.4 Lược đồ trích xuất đặc trưng MFCC trong đồ án 42](#_Toc27497325)

[Hình 2.5 (a) Mẫu thở khò khè nguyên bản (b) Mẫu thở khò khè sau khi được áp dụng bộ lọc pre-emphasis 43](#_Toc27497326)

[Hình 2.6 (a) Biểu diễn miền thời gian của 1 frame (b) Biểu diễn miền thời gian của cửa sổ Hamming (c) Biểu diễn miền thời gian của frame sau khi áp dụng của sổ Hamming 44](#_Toc27497327)

[Hình 2.7 Quá trình xây dựng ngân hàng bộ lọc Mel 44](#_Toc27497328)

[Hình 2.8 40 bộ lọc thang Mel được sử dụng 45](#_Toc27497329)

[Hình 2.9 Mel spectrogram của một mẫu thở khò khè 46](#_Toc27497330)

[Hình 2.10 39 hệ số MFCC của 1 frame 47](#_Toc27497331)

[Hình 2.11 Biểu diễn ma trận đặc trừng MFCC của một mẫu thở khò khè 47](#_Toc27497332)

[Hình 2.12 Lược đồ trích xuất đặc trưng GFCC 47](#_Toc27497333)

[Hình 2.13 40 hệ số GFCC của 1 frame 48](#_Toc27497334)

[Hình 2.14 Biểu diễn ma trận đặc trưng GFCC của 1 mẫu thở khò khè 48](#_Toc27497335)

[Hình 2.15 Đồ thị hàm sigmoid [43] 50](#_Toc27497336)

[Hình 2.16 Hình trạng mạng của mạng CNN cơ bản để phát hiện tiếng thở khò khè 52](#_Toc27497337)

[Hình 2.17 Mạng CNN được code bằng framework keras 53](#_Toc27497338)

[Hình 2.18 Chuẩn hóa spectrogram bằng zscore 54](#_Toc27497339)

[Hình 2.19 Callback LogResult được sử dụng trong keras 55](#_Toc27497340)

[Hình 2.20 Quá trình lan truyền ngược được thực thi trong framework keras 55](#_Toc27497341)

[Hình 2.21 Cấu trúc một khối LSTM 56](#_Toc27497342)

[Hình 2.22 Hình trạng mạng LSTM sử dụng để phát hiện tiếng thở khò khè 57](#_Toc27497343)

[Hình 2.23 Luồng làm việc khi tích hợp vào thiết bị di động 59](#_Toc27497344)

[Hình 2.24 (a) Tín hiệu gốc (b) Tín hiệu thông qua lọc nhiễu 60](#_Toc27497345)

[Hình 3.1 Đánh nhãn dữ liệu bằng công cụ Audacity 63](#_Toc27497346)

[Hình 3.2 (a) mẫu thở khò khè bình thường (b, c, d) một số mẫu thở khò khè được mở rộng 64](#_Toc27497347)

[Hình 3.3 Confusion Matrix thí nghiệm A với đặc trưng Spectrogram sử dụng mạng CNN 67](#_Toc27497348)

[Hình 3.4 Confusion Matrix thí nghiệm B1 với đặc trưng Spectrogram sử dụng mạng CNN 68](#_Toc27497349)

[Hình 3.5 Confusion Matrix thí nghiệm B2 với đặc trưng Spectrogram sử dụng mạng CNN 68](#_Toc27497350)

[Hình 3.6 Confusion Matrix thí nghiệm C với đặc trưng Spectrogram sử dụng mạng CNN 68](#_Toc27497351)

[Hình 3.7 Confusion Matrix thí nghiệm A với đặc trưng MFCC sử dụng mạng LSTM 69](#_Toc27497352)

[Hình 3.8 Confusion Matrix thí nghiệm B1 với đặc trưng MFCC sử dụng mạng LSTM 70](#_Toc27497353)

[Hình 3.9 Confusion Matrix thí nghiệm B2 với đặc trưng MFCC sử dụng mạng LSTM 70](#_Toc27497354)

[Hình 3.10 Confusion Matrix thí nghiệm C với đặc trưng MFCC sử dụng mạng LSTM 70](#_Toc27497355)

[Hình 3.11 Confusion Matrix thí nghiệm A với đặc trưng GFCC sử dụng mạng LSTM 71](#_Toc27497356)

[Hình 3.12 Confusion Matrix thí nghiệm B1 với đặc trưng GFCC sử dụng mạng LSTM 72](#_Toc27497357)

[Hình 3.13 Confusion Matrix thí nghiệm B2 với đặc trưng GFCC sử dụng mạng LSTM 72](#_Toc27497358)

[Hình 3.14 Confusion Matrix thí nghiệm C với đặc trưng GFCC sử dụng mạng LSTM 72](#_Toc27497359)

[Hình 3.15 Confusion Matrix thí nghiệm A với kết hợp đặc trưng MFCC GFCC sử dụng mạng LSTM 73](#_Toc27497360)

[Hình 3.16 Confusion Matrix thí nghiệm B1 với kết hợp đặc trưng MFCC GFCC sử dụng mạng LSTM 74](#_Toc27497361)

[Hình 3.17 Confusion Matrix thí nghiệm B2 với kết hợp đặc trưng MFCC GFCC sử dụng mạng LSTM 74](#_Toc27497362)

[Hình 3.18 Confusion Matrix thí nghiệm C với kết hợp đặc trưng MFCC GFCC sử dụng mạng LSTM 75](#_Toc27497363)

MỞ ĐẦU

Âm thanh của phổi đã được sử dụng như một công cụ chẩn đoán bệnh từ rất lâu. Ngoài việc tiết kiệm được thời gian và chi phí so với các phương pháp khác, phương pháp nghe âm phổi còn có ưu điểm khi tình trạng bệnh biến đổi thì các dấu hiệu về âm thanh thay đổi ngay lập tức trong khi các biện pháp tiên tiến khác (như ảnh x-ray) thì không. Ống nghe hiện nay vẫn được sử dụng như là một phương pháp sàng lọc có tiềm năng lớn cho việc theo dõi liên tục vì tính đơn giản và giá thành thấp.

Theo tổ chức y tế thế giới, bệnh phổi hiện là nguyên nhân phổ biến thứ ba dẫn đến tử vong chỉ sau bệnh tim và đột quỵ, kết quả thống kê ở nhiều quốc gia, khoảng 5% dân số ảnh hưởng bởi bệnh hen với các biểu hiện như ho, khó thở và dấu hiệu chính là xuất hiện âm phổi khò khè, một loại âm phổi bất thường có tính liên tục.

Âm thanh phổi tạo ra trong quá trình hô hấp là nguồn thông tin hữu ích cho việc đánh giá và điều trị các bệnh về phổi. Ví dụ như dấu hiệu xuất hiện của tiếng thở khò khè ở trẻ em được sử dụng như một tham số để đánh giá xu hướng của bệnh hen. Hiện nay một trong những nguyên nhân phổ biến nhất gây khó khăn trong điều trị bệnh phổi là chẩn đoán và phát hiện muộn. Với sự gia tăng số lượng bệnh nhân hen và các bệnh phổi khác dẫn đến việc không có đủ nhân viên y tế để chẩn đoán cho tất cả bệnh nhân kịp thời. Từ đó phát sinh nhu cầu lâu dài về các hệ thống phát hiện âm phổi khò khè để giám sát sức khỏe của phổi thường xuyên nhằm cung cấp kịp thời các thông tin quan trọng cho nhân viên y tế, nhằm nâng cao hiệu quả điều trị bệnh.

Học máy đặc biệt là các kỹ thuật học sâu hiện đang phát triển rất mạnh vì sự phát triển của sức mạnh tính toán và bùng nổ dữ liệu. Với các kỹ thuật học sâu, chương trình sẽ mềm dẻo hơn do có khả năng tự động điều chỉnh các tham số dựa vào dữ liệu được đưa vào để huấn luyện. Vì vậy học sâu đang được áp dụng trên rất nhiều lĩnh vực từ công nghệ thông tin, sinh trắc học, y tế,…

Trong phạm vi kiến thức, đồ án sẽ tập trung vào nghiên cứu và xây dựng mô hình phát hiện âm phổi khò khè dựa vào các kỹ thuật học sâu. Nội dung chi tiết của đồ án trình bày trong các chương sau, bao gồm:

* **Chương 1 - Tổng quan:**

Trong chương 1, đồ án tập trung trình bày về các khái niệm, đặc tính của một vài loại âm phổi, giới thiệu các nghiên cứu liên quan, đồng thời tìm hiểu về kiến trúc và các thuật toán cần thiết để xây dựng các mạng học sâu.

* **Chương 2 - Mô hình học sâu cho bài toán phát hiện tiếng thở khò khè:** Trong chương 2, đồ án trình bày định dạng dữ liệu của bài toán, các cách trích chọn đặc trưng và đề xuất các mô hình học sâu để học trên các đặc trưng đó.
* **Chương 3 - Thực nghiệm và đánh giá:**

Trong chương 3, đồ án trình bày về dữ liệu thu thập được, phương pháp thực nghiệm và đánh giá kết quả của những thực nghiệm đó.

* **Chương 4 –Tổng kết:**

Chương cuối cùng trong đồ án trình bày những kết quả đạt được trong quá trình thực hiện đồ án. Đồng thời đưa ra hướng phát triển trong tương lai của đề tài.

# TỔNG QUAN

Trong chương 1, đầu tiên đồ án trình bày về tổng quan hô hấp và các đặc tính của một số loại âm phổi phổ biến, tiếp theo là những nghiên cứu trước đây về vấn đề phát hiện tiếng thở khò khè, từ đó phân tích và đề xuất phương pháp thích hợp dựa trên kỹ thuật học sâu. Cụ thể chương 1 sẽ gồm các nội dung sau đây:

* Tổng quan về các loại âm phổi
* Các nghiên cứu trước đây về tự động phát hiện tiếng thở khò khè
* Học sâu
* Mục tiêu đồ án

## Tổng quan về nghe âm phổi

### Cấu trúc nhịp thở

Hít thở là quá trình di chuyển không khí nhằm cung cấp oxi và thải carbon dioxide thông qua các cơ quan hô hấp như phổi hoặc mang. Hít thở cũng được coi là hệ thống thông gió và nó bao gồm cả hít vào và thở ra. Thở là một phần của hô hấp và sinh lý: nó là cần thiết để duy trì sự sống. Các sinh vật hiếu khí, chẳng hạn như các loài chim, động vật có vú và các loài bò sát cần có oxy để giải phóng năng lượng thông qua hô hấp đến từng tế bào, trong các hình thức chuyển hóa các phân tử giàu năng lượng như glucose. Hít thở là một quá trình cung cấp oxy đến nơi cần thiết trong cơ thể và loại bỏ carbon dioxide. Đây là một quá trình quan trọng liên quan đến sự chuyển động của máu trong hệ thống tuần hoàn. Sự trao đổi khí xảy ra ở các phế nang phổi bằng cách khuếch tán thụ động của khí giữa phế nang và máu trong các mao mạch phổi. Khi các khí hoà tan trong máu, tim co bóp đẩy máu chạy vòng quanh cơ thể (thông qua hệ thống tuần hoàn) [[1](#htt)].

Trong mỗi nhịp thở luôn tồn tại hai pha liên tục nối tiếp nhau:

* **Hít vào:** Việc hít vào là do sự co lại và làm phẳng của cơ hoành, một cơ vòm ngăn cách ngực và bụng. Nếu bụng dưới thư giãn, co cơ này làm cho bụng dưới nở ra, giãn dung lượng của cơ thể. Dung lượng tăng này làm giảm áp lực trong ngực và gây ra sự giãn nở của phổi. Khi cơ hoành thư giãn việc thở ra chủ yếu do độ đàn hồi của phổi. Cách thở thoải mái và thư giãn này cần ít năng lượng. Khi nhu cầu hít thở tăng lên, các cơ bụng chống lại sự giãn nở. Áp lực bụng tăng lên làm nghiêng cơ hoành và đẩy lồng ngực trở lên làm gia tăng lượng không khí hít vào.
* **Thở ra**: Pha thở ra nối tiếp sau việc thư giãn của cơ hoành và cơ bụng, nhưng có thể được ép tăng lên bằng hành động đi xuống của cơ bụng trên lồng ngực. Cơ chế ép thở ra này làm tăng áp lực trên các bộ phận của đường hô hấp và có thể dẫn đến sự thu hẹp đường thở và dẫn đến thở khò khè. Cơ liên sườn đóng vai trò phụ trợ, làm cứng và tạo khung lồng ngực. Hoạt động nói phụ thuộc vào sự cân bằng giữa hai hình thức hít vào và thở ra của hô hấp và con người thay đổi ý thức thường xuyên***.***

Âm thanh phổi thông thường được thu ở cả hai phía trước và sau ngực bằng ống nghe điện tử. Dựa vào đặc tính âm thanh của âm phổi, âm phổi có thể chia thành hai loại âm phổi bình thường và bất bình thường. Đối với âm phổi bất thường có thể chia thành âm phổi bất bình thường liên tục và âm phổi bất bình thường gián đoạn.

### Các loại âm phổi bình thường

Các âm phổi bình thường được được chia thành 3 loại:

* **Bronchial:** Thu được ở ngực trước và khí quản. Có tần số và âm thanh lớn. Pha hít vào ngắn hơn một chút so với pha thở ra.
* **Bronchovesicular:** Thu được ở vùng liên sườn gần xương ức đối với ngực trước, ở giữa các phần xương vai đối với phần sau ngực, phế quản. Âm phổi này có tần số trung bình, độ dài của pha hít vào và pha thở ra là như nhau.
* **Vesicular:** Thu được cả ở ngực trước, sau ngực và xung quanh phổi. Âm thanh mềm mại với tần số thấp, độ dài pha hít vào lớn hơn một chút đối với pha thở ra.

### Các loại âm phổi bất bình thường

Các loại âm phổi bất thường thường chỉ ra những vấn đề về phổi và đường hô hấp, các nguyên nhân chính gây ra các âm phổi bất bình thường bao gồm viêm phổi, suy tim, các bệnh phổi mãn tính (COPD), hen, viêm phế quản, dị vật trong phổi hoặc đường thở.

Các âm thanh bất thường bao gồm những loại sau đây:

* **Wheeze**: còn gọi là thở khò khè, là trạng thái thở bất thường, có âm thanh thở lạ diễn ra liên nghe như tiếng sáo. Đặc tính của âm thanh thở khò khè là hiện tượng xảy ra những bước nhảy về tần số đột ngột trong âm thanh (thường lớn hơn 100Hz) nhưng lại diễn ra định kỳ và đều đặn, thời lượng diễn ra của hiện tượng này khoảng 250ms [[2](#Pas97)].
* **Rhonchi (Low Pitch Wheeze)**: đây là hiện tượng thở cũng có tính chất tương tự như thở khò khè, tuy nhiên lại có âm vực thấp, nên thay vì nghe như tiếng sáo, âm thanh hơi thở này nghe giống tiếng ngáy. Thời lượng hiện tượng cũng vào khoảng 100ms, tần số bé hơn 300Hz [[3](#Sov00)].
* **Stridor**: hay còn gọi là thở rít. Âm thở này có tần số rất thấp, bắt nguồn từ thanh quản hoặc khí quản. Nó thường xuất hiện trong pha hít vào. Hiện tượng này xuất hiện khi bệnh nhân mắc bệnh ho gà, hẹp thanh quản hoặc khí quản [[4](#Bau89)].
* **Crackles**: Âm thanh không dạng nhạc, không liên tục, thời lượng ngắn được gọi là crackles. Crackles được phân 2 loại là loại âm sắc thấp và cường độ lớn gọi là coarse crackles (trước đây gọi là rales hoặc coarse rales) và loại âm sắc cao hơn, cường độ nhỏ hơn gọi là fine crackles (trước đây gọi là fine rales or crepitations). Âm thanh crackles phản ánh tình trạng bệnh lý ở mô phổi hay đường thở [[5](#Hoe90)].

## Các nghiên cứu liên quan về phát hiện tiếng thở khò khè.

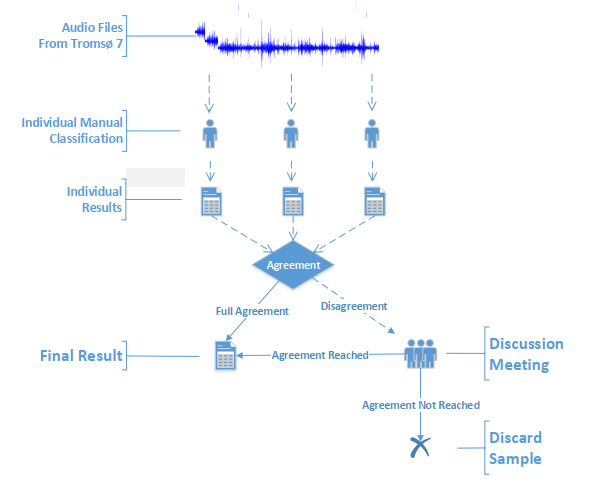
### Phân tích tự động tiếng phổi

Nghiên cứu [[6](#Grø16)] tập trung nghiên cứu về hai loại âm phổi bất thường hay gặp nhất là Crackles và Wheeze đồng thời tạo ra 1 cơ sở dữ liệu tiêu chuẩn với các âm thanh được phân loại bởi các chuyên gia y tế. 3 bộ phân loại được sử dụng trong [[6](#Grø16)] để đánh giá hiệu năng là Support Vector Machine (SVM), K-Nearest Neighbors, Decision Trees.

***Dữ liệu***

Dữ liệu trong Tromsø Study 7 [[7](#The)] thu thập tiếng phổi của 3000 người tham gia. Với mỗi cá nhân có 6 bản tiếng phổi được thu ở 6 vị trí khác nhau (hai vị trí ở thân trên trước, 2 ví trí trên của lưng, 2 vị trí dưới của lưng). Tất cả các file được lưu dưới định dạng wav với tần số lấy mẫu là 44100 Hz. Tổng cộng có 18000 bản ghi âm tiếng phổi.

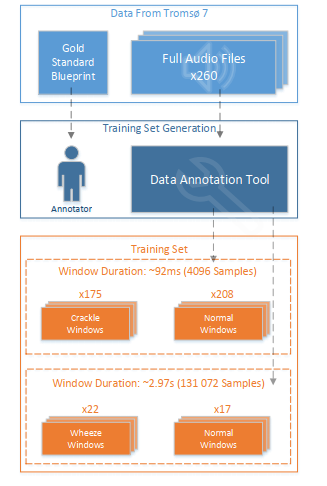
Quy trình gán nhãn được thể hiện trong Hình 1.1. Cụ thể dữ liệu được đánh nhãn bằng tay bởi 1 nhóm 3 bác sĩ phân loại từng bản ghi một cách độc lập. Nếu tất cả đồng ý, bản ghi đó coi như được gán nhãn thành công. Nếu có bất đồng giữa các cá nhân phân loại, bản ghi sẽ được lưu lại để thảo luận. Khi các bác sĩ gặp và thảo luận nhãn của bản ghi đó, nếu không đạt được sự thống nhất file sẽ bị loại bỏ khỏi bộ dữ liệu. Tại thời điểm viết [[6](#Grø16)] có 2500 file âm thanh được gán nhãn.



Hình 1.1 Quy trình đánh dữ liệu của Tromso Study [[6](#Grø16)]

Tuy nhiên chỉ một phần nhỏ của tập dữ liệu trên được sử dụng trong [[6](#Grø16)]. Có 3 nhãn được phân loại trong [[6](#Grø16)]: Normal, crackles, wheeze. Do một file âm thanh có độ dài trung bình 15s có thể lẫn một vài loại âm thở khác nhau. Nên một chuyên gia sẽ tiến hành gán nhãn thủ công cho các cửa sổ có kích thước nhỏ hơn với các file đã được phân loại ở bước trước.

Vì đặc tính khác nhau của crackles và wheeze, nên hai cửa sổ có độ dài khác nhau được áp dụng. Đối với crackles thì cửa sổ có độ dài 92ms (tương đương 4096 mẫu trên cửa sổ) còn với wheeze sử dụng cửa sổ độ dài 2.97s (tương đương 131072 mẫu trên cửa sổ). Các cửa sổ cạnh nhau sẽ chồng lấn lên nhau 50% để đảm bảo dữ liệu không bị mất mát. Đối với âm thở bình thường một cửa sổ có kích thước tương đương được sử dụng (92ms với bộ phân loại tiếng crackles, 2.97s với bộ phân loại tiếng thở khò khè) Hình 1.2.



Hình 1.2 Quy trình tạo ra bộ dữ liệu trong [[6](#Grø16)]

Như vậy với bộ phân loại tiếng crackles có 175 mẫu có nhãn crackles, 208 mẫu có nhãn normal. Với bộ phân loại tiếng thở khò khè có 22 mẫu có nhãn wheeze, 17 mẫu có nhãn là normal.

***Trích chọn đặc trưng***

Trong [[6](#Grø16)] các đặc trưng được trích chọn bao gồm:

* Variance.
* Range.
* Sum of Simple Moving Average (SMA).
* Spectrum Mean

Đặc trưng Range là giá trị tuyệt đối của giá trị trong file audio trừ đi giá trị nhỏ nhất của file audio. Được cho bởi công thức 1.1:

(1.1)

Đặc trưng Sum of Simple Moving Average (SMA) có hai phiên bản được sử dụng trong [[6](#Grø16)] và được tính theo công thức 1.2:

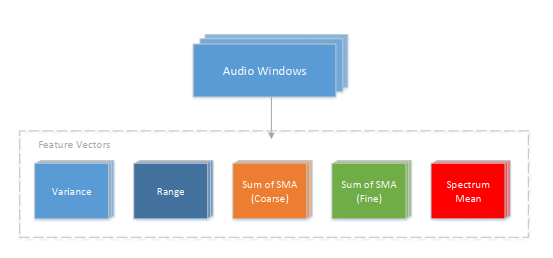
(1.2)

Hai phiên bản được sử dụng trong [[6](#Grø16)] là coarse version và fine version. Coarse version tính toàn bộ tín hiệu theo công thức trên. Fine version sử dụng một cửa sổ nhỏ hơn có kích thước là 800, mỗi lần trượt 100 mẫu và chỉ giữ lại cửa sổ có lượng thay đổi là lớn nhất.

(1.3)

Giá trị trung bình của spectrum cho chúng ta cái nhìn về xu hướng tập trung trên miền tần số. Crackles xuất hiện khi thở thường mang nhiều năng lượng ở các tần số lớn. Trung tâm của phân phối năng lượng sẽ mang giá trị lớn hơn cho bất kì cửa sổ chứa crackles.

Như vậy với mỗi cửa sổ được đặc trưng bởi 1 vector đặc trưng 5 chiều thể hiện trong Hình 1.3.



Hình 1.3 Mỗi cửa sổ được đặc trưng bởi một vector 5 chiều [[6](#Grø16)]

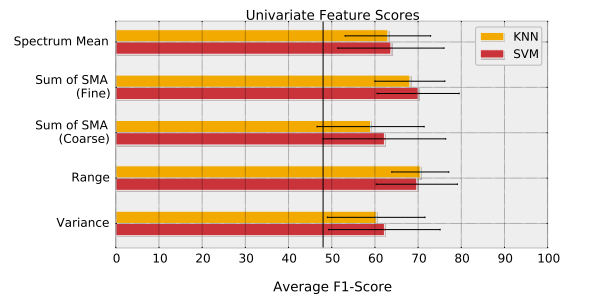
***Thực nghiệm và kết quả:***

Nghiên cứu [[6](#Grø16)] được thực hiện trên môi trường có cấu hình như sau:

* Windows 10 Pro 64-bit.
* Intel Core i5-4570s @ 2.90GHz (4 Physical Cores).
* 6Gb RAM Python 2.7.9.

Với bộ phân loại tiếng crackles, đầu tiên từng đặc trưng được sử dụng một cách độc lập để tìm ra đặc trưng nào tốt nhất cho việc phân loại giữa các lớp. Các đặc trưng được kiểm tra bằng cách chạy 1 chu kì train – validate 100 lần, sau đó tính trung bình F1-score qua tất cả các chu kì.

Cụ thể từng chu kì sẽ chia tập huấn luyện thành 70% để huấn luyện và 30% để kiểm chứng. Mỗi chu kì huấn luyện sẽ thực hiện grid search nhằm điều chỉnh các tham số. Việc điều chỉnh các tham số sẽ được đánh giá bằng kiểm chứng chéo 3 – fold trên tập huấn luyện để đánh giá và tìm ra tham số tốt nhất.



Hình 1.4 F1 scores trung bình cho từng đặc trưng, đường kẻ dọc đánh dấu F1 score cho việc đoán ngẫu nhiên [[6](#Grø16)].

Sau khi kiểm tra từng đặc trưng một cách riêng biệt, tất cả các đặc trưng sẽ được sử dụng. Các bộ phân loại được so sánh với một dummy classifier đưa ra kết quả theo một chiến lược cho trước (cụ thể kết quả sẽ tỉ lệ thuận với các mẫu trong từng lớp). Kết quả của bộ phân loại tiếng crackles được thể hiện trong Bảng 1.1:

Bảng 1.1 So sánh hiệu năng các bộ phân loại trên crackles

|  |  |  |  |
| --- | --- | --- | --- |
| **Classifier** | **Precision** | **Recall** | **F1-Score** |
| SVM (RBF) | 85.6 ± 6.1 | 83.6 ± 10.2 | 83.5 ± 3.6 |
| KNN | 84.4 ± 6.9 | 82.3 ± 11.3 | 82.5 ± 4.7 |
| AdaBoost (Decision Tree) | 82.7 ± 5.2 | 81.8 ± 8.3 | 81.9 ± 4.5 |
| Linear SVM | 88.7 ± 6.7 | 67.3 ± 8.6 | 76.0 ± 5.4 |
| Dummy Classifier (Stratified) | 49.7 ± 7.7 | 49.7 ± 7.4 | 49.5 ± 6.8 |

Bộ phân loại có hiệu năng tốt nhất được đánh giá lại trên 270 file audio có kích thước đầy đủ, 1 file được phân loại là có nhiều tiếng thở crackles khi số cửa sổ chứa crackles lớn hơn 30. Trong 270 file âm thanh có 23 file chứa crackles và 247 file chứa tiếng thở bình thường. Như đã thấy trong ma trận nhầm lẫn mặc dù sử dụng ngưỡng rất cao nhưng kết quả thấp hơn so với các cửa sổ riêng biệt.

Bảng 1.2 Ma trận nhầm lẫn trong phân loại các file âm thanh đầy đủ [[6](#Grø16)]

|  |  |  |
| --- | --- | --- |
|  | **Crackles** | **Normal** |
| **Crackles** | 14 | 9 |
| **Normal** | 61 | 186 |

Với bộ phân loại tiếng khò khè: Chỉ bộ phân loại SVM được sử dụng vì đây là bộ phân loại có hiệu năng tốt nhất cho crackles. Việc đánh giá được thực hiện tương tự như crackles. Kết quả cho thấy không tốt hơn so với việc đoán ngẫu nhiên. Tác giả của [[6](#Grø16)] giải thích vì đặc tính khác nhau của wheeze và crackles.

Bảng 1.3 Kết quả với SVM sử dụng RBF kernel [[6](#Grø16)]

|  |  |  |  |
| --- | --- | --- | --- |
| **Classifier** | Precision | Recall | F1-Score |
| SVM (RBF) | 64.8 ± 19.0 | 74.3 ± 20.8 | 63.7 ± 24.8 |

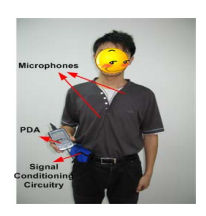
Trong khi crackles có độ dài ngắn và 1 sóng riêng biệt trong miền thời gian, wheezes lại chứa nhiều thông tin trong miền phổ, với 1 âm thanh giống tiếng huýt sáo. Ngoài ra wheezes có độ dài lớn hơn so với crackles vì vậy các tác giả tin rằng các đặc trưng trong miền phổ sẽ tốt hơn cho wheeze so với crackles chú ý vào các đặc trưng miền thời gian.

### Hệ thống phát hiện hơi thở khò khè dựa trên Entropy

Trong [[8](#Zha09)] trình bày một phương pháp mới phát hiện tiếng thở khò khè khai thác khái niệm entropy để mô tả các chuỗi của phổ tần số trong tín hiệu. Có thể tự động phát hiện tiếng thở khò khè sử dụng chỉ một hoặc hai đặc trưng dựa trên entropy. Vì thế phương pháp có độ phức tạp tính toán thấp, có khả năng triển khai trên các thiết bị nhỏ có thể đeo trên người để giám sát nhịp thở.

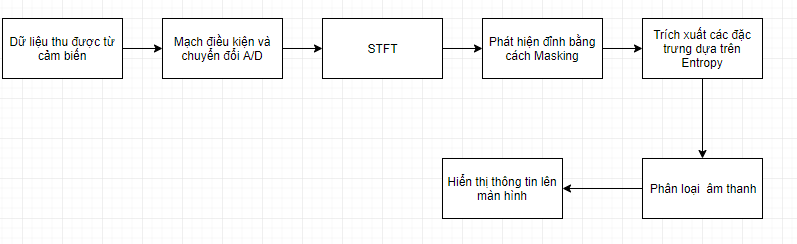
***Các phần cứng được sử dụng:***

Phương pháp đề xuất đã được triển khai trong một thiết bị âm thanh có thể đeo dựa trên hệ thống giám sát hơi thở. Như trong Hình 1.5, hệ thống bao gồm 2 module chính. Cảm biến và phần mạch điều kiện tín hiệu với nền tảng PDA. Microphones có số model Panasonic WM-64PN được sử dụng trong hệ thống này để thu tiếng thở thông thường. Một PDA với số hiệu model HP iPAQ hx2700 là nền tảng triển khai phương pháp đề xuất. Đầu ra của phát hiện/ nhận biết tín hiệu được chỉ thông qua các đèn thông báo trên màn hình của PDA.



Hình 1.5 Phần cứng của hệ thống phát hiện tiếng thở khò khè dựa trên entropy [[8](#Zha09)]

Chi tiết về phương pháp được thể hiện trong Hình 1.6.



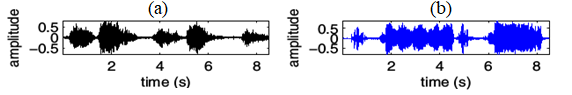
Hình 1.6 Sơ đồ khối của phương pháp tự động phát hiện tiếng khò khè dựa trên entropy [[8](#Zha09)]

***Tiền xử lý dữ liệu:***

Tín hiệu được thu bởi các cảm biến và gửi tới mạch điều kiện để khuếch đại, lọc và sau đó chuyển đổi thành tín hiệu số. Tín hiệu được số hóa tại đầu ra của bộ chuyển đổi A/D được viết dưới dạng:

*s(t) = w(t) + n(t)* (1.4)

Trong đó *w(t)* biểu diễn tín hiệu của âm thanh khò khè và n(t) tương ứng với tín hiệu nhiễu



Hình 1.7 Tín hiệu ở miền thời gian (a) Tín hiệu tiếng thở bình thường (b) Tín hiệu ở tiếng thở khò khè [[8](#Zha09)].

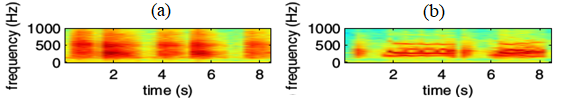
Tín hiệu số này có thể chuyển đổi sang miền tần số sử dụng STFT.

(1.5)

Trong đó:

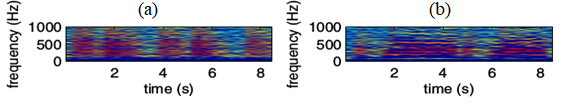
* *x[n]* là tín hiệu được chuyển đổi
* *w[n]* là cửa sổ trượt qua mỗi frame.

Cần lưu ý là trong xử lý tiếng nói, độ rộng của cửa sổ thường được cài là 10 - 30ms. Vì tín hiệu thở thay đổi chậm hơn tiếng nói rất nhiều nên cửa sổ Hamming rộng hơn (cụ thể là 60ms) được sử dụng cho việc phát hiện tiếng khò khè. Kết quả được thể hiện trong Hình 1.8:



Hình 1.8 Tín hiệu ở miền tần số (a) Tín hiệu của tiếng thở bình thường (b) Tín hiệu của tiếng thở khò khè [[8](#Zha09)].

Sau bước biến đổi STFT cho chúng ta các thành phần tần số của tín hiệu. Các thành phần tần số với các giá trị lớn được xác định bằng cách masking với các giá trị nhỏ hơn. Tại mỗi khoảng thời gian, các thành phần tần số với các giá trị lớn có thể chỉ ra bằng bộ lọc trung bình trên kết quả của STFT, các thành phần đỉnh sẽ được mask với các thành phần lân cận với có giá trị nhỏ hơn, kết quả của thủ tục này được minh họa trong ảnh dưới.



Hình 1.9 Tín hiệu sau khi thực hiện masking (a) Tín hiệu của tiếng thở bình thường (b) Tín hiệu của tiếng thở khò khè [[8](#Zha09)].

***Trích xuất đặc trưng dựa trên Entropy:***

So sánh a và b trong ảnh trên . Nhận thấy rằng các vùng bỏ trống bên cạnh các đỉnh trong Hình 1.9b lớn hơn rất nhiều trong Hình 1.9b. Điều này chỉ ra rằng tiếng khò khè là các tín hiệu với các đặc tính phân cụm về mặt phân phối năng lượng. Nói một các khác, phân phối năng lượng của tiếng thở bình thường khá là đồng đều qua phổ tần số , trong khi phân phối năng lượng của tiếng thở khò khè tập trung ở các tần số mà tiếng thở khò khè xuất hiện.

Đặc tính rải rác năng lượng có thể mô tả bởi khái niệm entropy. Mặc dù được xây dựng trong động lực học để đo độ ngẫu nhiên của phân tử trong một hệ thống, khái niệm entropy nhanh trong phù hợp với các lĩnh vực nghiên cứu khác bao gồm cơ học, vật lý học thiên thể và lí thuyết thông tin. Trong lý thuyết thông tin, entropy thường được đề cập tới là shannon entropy, một thước đo của sự không chắc chắn liên quan đến một biến ngẫu nhiên. Khái niệm entropy đã được mở rộng để mô tả tính nghiêm ngặt của các đặc tính phân cụm của phân phối năng lượng các tín hiệu.

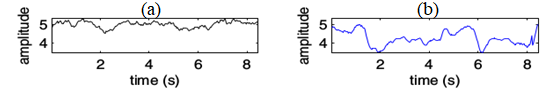
Giả sử có N thành phần phổ biến trong mỗi slot thời gian, những giá trị đỉnh này kí hiệu bởi C1, C2, …, CN. Trọng số của các thành phần tần số chủ yếu được đo bởi phần trăm khác nhau giữa năng lượng của từng thành phần năng lượng với tổng năng lượng của tất cả các thành phần.

*pn =* , *n = 1,2, ….N* (1.6)

Sử dụng những trọng số này, entropy có thể được tính tương tự như trong lý thuyết thông tin bằng công thức:

*E =*  (1.7)

Các entropy của các slot thời gian được cho bởi 1.7 có thể được lọc bằng bộ lọc trung bình theo trục thời gian. Các smoothed entropies cho hơi thở bình thường và hơi thở khò khè được trình bày trong Hình 1.10.



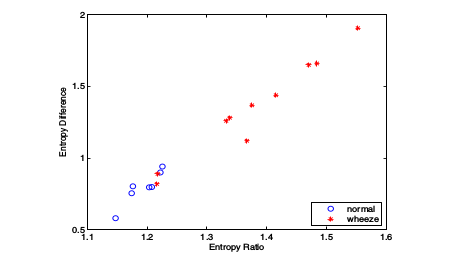
Hình 1.10 Smoothed Entropies(a) Smoothed Entropies của tiếng thở bình thường(b)Smoothed entropies của tiếng thở khò khè [[8](#Zha09)].

Sau đó các đặc trưng có thể được trích xuất bởi nhiều cách. Đặc trưng có thể là sự khác nhau giữa entropy lớn nhất và entropy nhỏ nhất hoặc tỉ lệ của entropy lớn nhất và entropy nhỏ nhất. Tín hiệu của tiếng thở khò khè có thể được phát hiện và bằng cách so sánh với ngưỡng của các đặc trưng. Ví dụ tín hiệu với các giá trị đặc trưng lớn hơn ngưỡng được phân loại như tiếng thở khò khè. Nếu không tín hiệu được phân loại là hơi thở thông thường.

***Kết quả:***

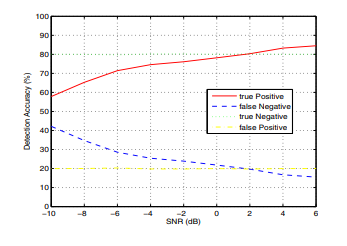
Để kiểm chứng phương pháp đề xuất EBWD [[8](#Zha09)] và đánh giá hiệu năng của thuật toán, phương pháp được để xuất được kiểm tra sử dụng các âm thanh phổi từ các bệnh nhân thật và các đối tượng thở bình thường. Tất cả các âm thanh được lấy mẫu lại tại tần số 2000Hz.

Kiểm chứng của phương pháp EBWD: Kiểm chứng của phương pháp được để xuất sử dụng các âm phổi được đề cập ở trên, kết quả được chỉ ra trong Hình 1.11. Rõ ràng là 8 / 10 bệnh nhân có thể phân biệt từ các đội tượng bình thường bởi sự khác nhau giữa entropy hoặc tỉ lệ giữa entropy hoặc cả hai. Phát hiện tiếng khò khè có thể hiện thực bằng chỉ một đặc trưng đơn.



Hình 1.11 Kết quả phân loại của phương pháp EBWD [[8](#Zha09)].

Hiệu năng của phương pháp EBWD dưới các nhiễu khác nhau: Để đánh giá sự mạnh mẽ của phương pháp EBWD chống lại nhiễu, hiệu năng của thuật toán được đề xuất được kiểm tra dưới các cấp độ khác nhau của SNR. Bài kiểm tra đã được tiếnhành trên các ghi âm âm phổi ở trên cộng với nhiễu AWGN nhân tạo. Hình 1.12 trình bày các kết quả của hiệu năng về mặt chính xác. Khi tỉ lệ phát hiện tiếng thở bình thường là cố định 80%, tỉ lệ phát hiện của tiếng thở khò khè cũng được tăng từ 58% tại 10dB lên 84.4% tại 6dB.



Hình 1.12 Kết quả phân loại của phương pháp EBWD dưới các nhiễu khác nhau [[8](#Zha09)].

### Nghiên cứu so sánh các thuật toán học máy Svm và k-nn để chẩn đoán các bệnh lý hô hấp bằng cách sử dụng âm thanh phổi

Trong [[9](#Pal14)] dữ liệu được sử dụng nằm trong cơ sở dữ liệu RALE [[10](#Ral)] là cơ sở dữ liệu âm thanh thương mại được sử dụng bởi rất nhiều nhà nghiên cứu, hệ thống đề xuất bao gồm 4 bước, tiền xử lý dữ liệu, trích xuất đặc trưng, phân loại và đánh giá. Trong bước trích xuất đặc trưng MFCC được trích xuất từ các tín hiệu hơi thở và được đưa vào bộ phân loại SVM và k-nn.

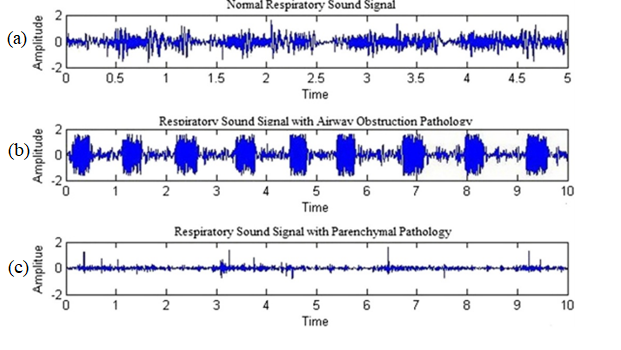
***Cơ sở dữ liệu âm thanh hơi thở***

Các bản âm thanh hơi thở được sử dụng trong nghiên cứu này lấy trong cơ sở dữ liệu RALE [[9](#Pal14)]. Bao gồm hơn 70 bản ghi từ các đối tượng khác nhau được ghi âm trên bề mặt của lồng ngực sử dụng gia tốc kế (EMT25C, Siemens). Những bản ghi âm này được gán nhãn bằng tay thành 3 nhóm khác nhau:

* Bệnh lý bình thường
* Bện tắc nghẽn đường thở,
* Bệnh lý nhu mô.

Tổng cộng có 68 bản ghi trong đó dữ liệu ứng với mỗi nhãn như sau:

* 17 bản ghi được gán nhãn bệnh lý bình thường
* 26 bản ghi được gán nhãn tắc nghẽn đường thở
* 25 bản ghi được gán nhãn bệnh lý nhu mô.



Hình 1.13 Tín hiệu miền thời gian cho các nhóm Bình thường (b) Tắc nghẽn đường thở (c) Bệnh lý nhu mô [[9](#Pal14)].

***Tiền xử lý dữ liệu:***

Các bản ghi âm hô hấp bị nhiễu như là các âm thanh của tim và một số nhiễu khác. Cơ sở dữ liệu rale [[10](#Ral)] bao gồm các bản ghi đã được lọc để loại bỏ những loại nhiễu đó. Cụ thể như sau:

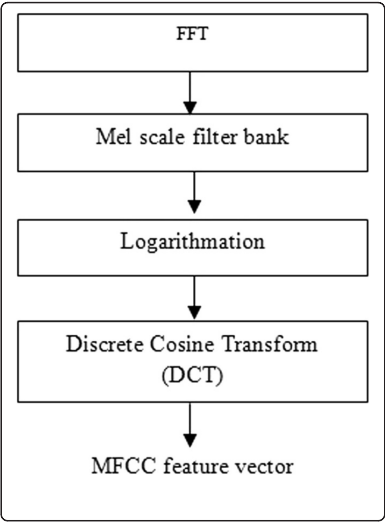
* Các tin hiệu được lọc qua bộ lọc high-pass tại tần số 7.5Hz để loại bỏ những DC offset sử dụng bộ lọc Butterworth đầu tiên.
* Bộ lọc low-pass tại tần số 2.5kHz được sử dụng để tránh aliasing sử dụng bộ lọc thứ 8 Butterworth.
* Tất cả các bản ghi âm được lấy mẫu tại tần số 10kHz

***Trích xuất đặc trưng:***

Đặc trưng được sử dụng trong bài báo này là MFCC. Trong phân tích MFCC tần số được đo bởi thang Mel. Tương tự với hệ thống phản hồi âm thanh của con người. Ngược lại với phân tích cepstral liên quan đến các dải tần cách nhau tuyến tính bằng cách sử dụng một cepstrum bình thường. Việc trích xuất MFCC được thực hiện như lược đồ Hình 1.14.

MFCC được tính toán từ các hệ số từ FFT, sau đó được lọc qua các ngân hàng bộ lọc thang mel. Tần số thông thường được ánh xạ vào tần số mel sử dụng công thức 1.8:

*Mel(f) = 2595log10(1 + )* (1.8)



Hình 1.14 Lược đồ trích xuất đặc trưng MFCC [[9](#Pal14)].

Thang logarithmic sau đó được chuyển đổi thành thời gian sử dụng biến đổi cosin rời rạc, đầu ra là một tập MFCCs. Các MFCCs thu được từ các âm thở được sử dụng như các đặc trưng trong bộ phân loại SVM và k-nn. Trong nghiên cứu này 13 hệ số MFCCs được trích xuất cho bộ phân loại.

***Thực nghiệm và kết quả:***

Đặc trưng MFCC được chuẩn hóa nhị phân và chuẩn hóa lưỡng cực. Việc chuẩn hóa đưa các biến độc lập về cùng một khoảng giúp cải thiện hiệu năng và hiệu quả của mô hình. Với bộ phân loại SVM nhân tuyến tính và nhân RBF được sử dụng để phân loại.

Trong [[9](#Pal14)] sử dụng hai cách kiểm chứng là kiểm chứng thông thường và kiểm chứng chéo 10 fold. Với kiểm chứng thông thường thì toàn bộ dữ liệu chia 60% để huấn luyện và 40% để kiểm tra.

Bộ phân loại SVM cho độ chính xác cao nhất là 92.19% với RBF kernel phương pháp kiểm chứng chéo 10 fold, dữ liệu được chuẩn hóa lưỡng cực. Bộ phân loại k-nn đạt độ chính xác cao nhất là 98.26% với k là 1, phương pháp kiểm chứng chéo 10 fold, dữ liệu chuẩn hóa lưỡng cực.

### Phát hiện tiếng thở khò khè sử dụng wavelets và mạng noron

Thở khò khè là một trong những loại thở bất thường mà cần được chú trọng theo dõi, nó là biểu hiện của bệnh hen suyễn, tắc nghẽn phổi (COPD) và bệnh viêm phế quản. Nghiên cứu [[11](#Has11)] trình bày phương pháp sử dụng mạng nơron, cụ thể là mạng Multi Layer Perceptron để thực hiện việc phân loại thở khò khè. Âm thanh thở khò khè sẽ được phân loại thành 2 nhóm: thở khò khè đơn âm và thở khò khè đa âm. Dựa vào thông tin phân loại này, các bác sĩ sẽ rút ra được chẩn đoán tốt hơn về bệnh mà bệnh nhân đang mắc phải.

***Dữ liệu:***

Dữ liệu âm thanh hô hấp sử dụng trong nghiên cứu được thu trực tiếp từ các bệnh nhân mắc bệnh hen suyễn, viêm phổi sử dụng ống nghe điện tử chuyên dụng tại Khoa Phổi ở Bệnh viện Loghman Hakim hospital, Tehran. Dữ liệu thu được sẽ được truyền trực tiếp tới máy tính để hiển thị và phân tích. Dữ liệu âm thanh thu được sẽ được lưu trữ với tần số mẫu là 8000Hz, tức là mỗi 1 giây âm thanh sẽ được lưu trữ bởi 8000 mẫu dữ liệu tương ứng.

Dữ liệu được thu thập từ 140 bệnh nhân khác nhau mắc các bệnh về hen suyễn, COPD và viêm phổi. Tất cả các bệnh nhân này đều mắc triệu chứng về hô hấp là thở khò khè. Trong đó có 77 bệnh nhân có âm thở khò khè thuộc nhóm đa âm, 63 bệnh nhân còn lại có âm thở khò khè thuộc nhóm đơn âm.

Thông tin về các chẩn đoán lâm sàng của bệnh nhân từ các xét nghiệm phổi, triệu chứng của bệnh nhân, nhịp tim, chụp X quang sẽ được thực hiện bởi các bác sĩ khoa hô hấp. Còn việc phân loại xem âm thanh thở khò khè thuộc nhóm nào sẽ được thực hiện bởi các chuyên gia y tế về hô hấp. Việc này đảm bảo dữ liệu thu thập được có độ chính xác và độ tin cậy cao.

***Tiền xử lý dữ liệu:***

Trước khi đưa vào áp dụng các phương pháp, thuật toán xử lý, toàn bộ dữ liệu sẽ được chuẩn hoá về biên độ, cường độ âm thanh. Sự thay đổi đột ngột về cường độ âm thanh có thể gây sai lệch thông tin trong việc phân tích và các âm thanh nhiễu xung quanh như tiếng ồn từ môi trường cũng ảnh hưởng rất nhiều trong quá trình phân tích và xử lý thông tin.

Kỹ thuật đường trung bình được áp dụng trong nghiên cứu [[11](#Has11)] để làm giảm đi những sự ảnh hưởng tiêu cực này. Dữ liệu sau khi xử lý sẽ mịn hơn, giảm đi sự biến thiên đột ngột về cường độ, đồng thời các âm thanh nhiễu nhỏ từ môi trường cũng sẽ được loại bỏ.

***Trích chọn đặc trưng:***

Âm thanh là dạng tín hiệu mà có sự biến đổi thông tin liên tục và một cách ngẫu nhiên theo thời gian mà không có quy luật nào cả. Chính vì vậy nếu áp dụng biến đổi Fourier trên cả đoạn âm thanh dài thì sẽ làm mất đi rất nhiều thông tin, không đủ để ta có thể thực hiện việc phân loại.

Kỹ thuật Short Time Fourier Transform (STFT), hay còn gọi là biến đổi Fourier đoạn ngắn đã khắc phục nhược điểm này. Một cửa sổ âm thanh nhỏ sẽ được trượt dọc theo tín hiệu âm thanh để thực hiện việc biến đổi trên từng đoạn đủ nhỏ, sao cho tín hiệu trên đoạn này có thể xem như không thay đổi. Kết quả của phép biến đổi ta được dữ liệu được biểu diễn dưới miền tần số và thời gian, hay còn gọi là Wavelet.

Trên mỗi đoạn dữ liệu sau khi được biến đổi, các đặc trưng sẽ được trích chọn, bao gồm:

* Giá trị cường độ trung bình.
* Giá trị năng lượng trung bình.
* Độ lệch chuẩn cường độ.
* Tỉ lệ trung bình cường độ giữa 2 đoạn liên tiếp.
* Đặc trưng Skewness.
* Đặc trưng Kurtosis.

***Phân loại và kết quả:***

Việc phân loại sẽ được thực hiện bằng phương pháp sử dụng mạng neural network, cụ thể là kiến trúc Multi Layer Perceptron. Cụ thể bao gồm 3 lớp: lớp đầu là lớp đầu vào, có số nơron bằng số đặc trưng, lớp cuối là lớp đầu ra, có 2 nơron tương ứng với 2 nhóm cần phân loại và lớp ở giữa là lớp ẩn, có số nơron tuỳ ý và số nơron được chọn là 45 trong nghiên cứu này. Kết quả của thực nghiệm khi áp dụng mạng MLP để phân loại âm thở khò khè đạt được là khá tốt, cụ thể có độ chính xác lên tới 89.28%.

## Học sâu

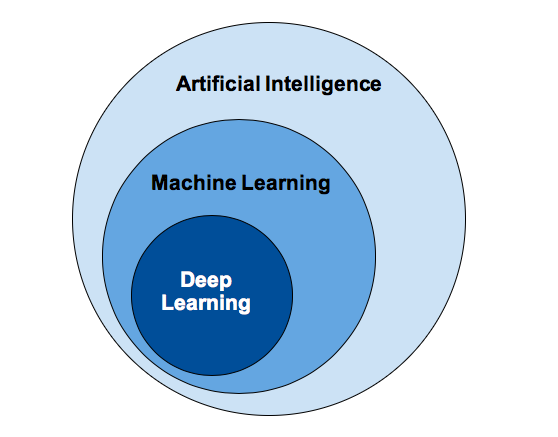
Trí tuệ nhân tạo (Artificial Intelligence – AI) là khả năng mà máy có thể làm những thứ mà não con người làm được. Trong lịch sử phát triển của AI, các nhà nghiên cứu phân thành các 4 hướng chính như sau :

* Hành động như người (acting humanly).
* Suy nghĩa như người (thinking humanly).
* Suy nghĩ hợp lý (thinking rationally).
* Hành động hợp lý (acting rationally).

Học máy (Machine Learning) chỉ là một thành phần trong hướng tiếp cận hành động như người và hướng tiếp cận này lại thuộc các hướng tiếp cận của AI. AI thể hiện một mục tiêu của con người, còn Machine Learning là một phương tiện được kỳ vọng sẽ giúp con người đạt được mục tiêu đó [[12](#Cop16)].

Để đạt được mục tiêu này, các nhà khoa học đã nghiên cứu ra nhiều giải thuật và các hướng giải quyết khác nhau của Machine Learning. Có nhiều cách để phân nhóm các thuật toán Machine learning, một trong những cách phổ biến đó là dựa trên phương thức học (learning style).

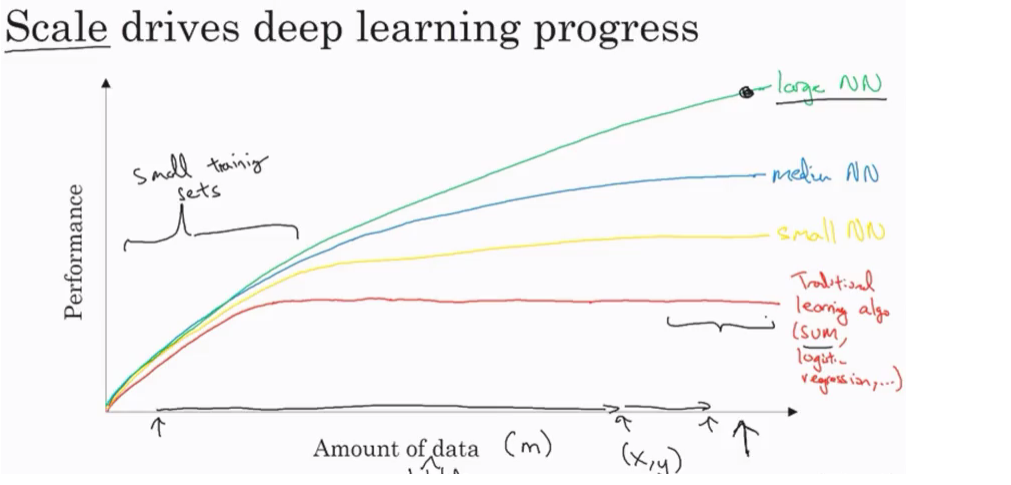
Nếu như AI là mục tiêu, Machine Learning là phương tiện, thì Học sâu (Deep Learning) là kỹ thuật giúp chúng ta thực hiện. Học sâu là một kỹ thuật trong Machine Learning, liên quan đến các thuật toán lấy cảm hứng từ cấu trúc và hoạt động của bộ não động vật gọi là mạng nornon nhân tao (Artificial Neural Networks) [[12](#Cop16)].



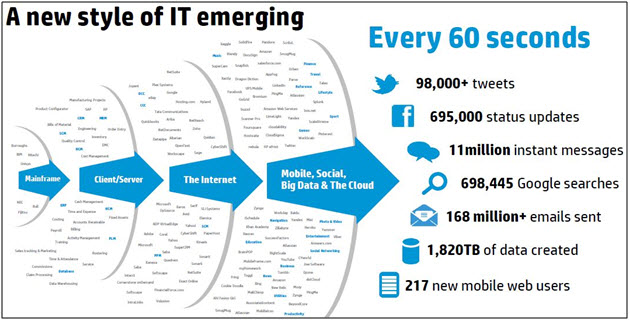
Hình 1.15 Mối quan hệ giữa AI, học máy và deep learning [[13](#Arece)]

Học sâu được đưa ra từ những năm 50 của thế kỷ trước. Mối quan hệ giữa AI, ML, DL được thể hiện trong Hình 1.15. Tuy nhiên trong thời gian gần đây thế giới mới thực sự thấy được hiệu quả của Deep Learning. Có thể giải thích bằng 3 lý do chính sau:

* **Dữ liệu:** Quan sát Hình 1.16chúng ta có thể thấy. Với bộ dữ liệu nhỏ mạng noron nhân tạo có hiệu năng ngang với hồi quy tuyến tính và SVM. Tuy nhiên với dữ liệu lớn hơn một mạng nornon nhân tạo nhỏ cho hiệu quả tốt hơn so với SVM. Với dữ liệu lớn thì mạng noron nhân tạo có kích thước càng lớn thì hiệu năng càng tốt. Hiện nay với sự phát triển của điện thoại, mạng xã hội, các thiết bị cảm biến cùng với các kỹ thuật của big data Hình 1.17. Dữ liệu chúng ta có càng nhiều, điều này tạo thuận lợi cho việc huấn luyện các mô hình học sâu.

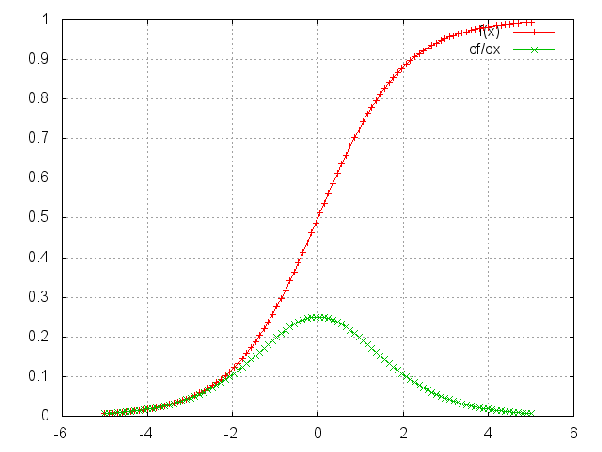


Hình 1.16 Hiệu năng của deep learning và lượng dữ liệu sử dụng để huấn luyện [[14](#And)]



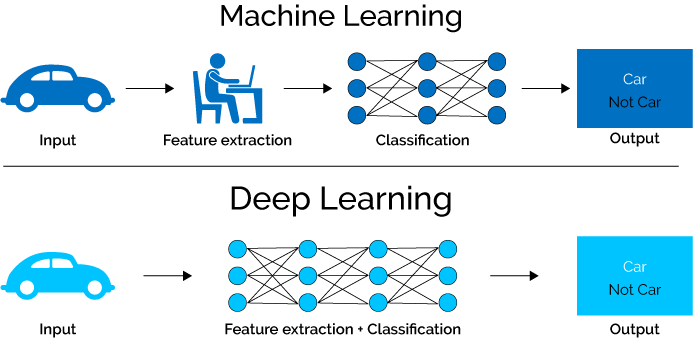
Hình 1.17 Số lượng dữ liệu khổng lồ trong thời đại số [[15](#Bigce)]

* **Sự phát triển năng lực tính toán:** Với sự phát triển và các tiện ích có sẵn của GPU khiến cho việc xử lý song song nhanh , rẻ và mạnh mẽ hơn. Điều này làm cho việc huấn luyện những mô hình học sâu với hàng triệu tham số là điều có thể thực hiện được.
* **Thuật toán:** Các thuật toán mới xuất hiện thay đổi cách mà mạng noron nhân tạo làm việc. Ví dụ sử dụng hàm RELU Hình 1.18 tốt hơn sử dụng hàm sigmoid trong một mạng noron nhân tạo vì nó giải quyết vấn đề vanishing gradient.



Hình 1.18 Hàm sigmoid và đạo hàm của hàm sigmoid [[16](#Howle)]

Deep Learning có thể coi là một dạng đặc biệt của Machine Learning.



Hình 1.19 Sự khác nhau giữa Machine Learning và Deep Learning [[17](#Wha86)]

Quy trình thực hiện của một mô hình machine learning cơ bản sẽ là :

* Trích xuất thủ công lượng đặc trưng (feature) từ ảnh.
* Hiệu chỉnh các tham số để kết hợp, biến đổi đặc trưng.
* Từ lượng feature thu được sẽ tiến hành phần loại đối tượng trong ảnh.

Đối với Deep Learning, tất các các bước trên của mô hình machine learning sẽ được thực thiện tự động, gói gọn trong một mô hình deep learning duy nhất . Đây là ưu điểm vượt trội của Deep Learning so với các phương pháp machine learning truyền thống.

### Convolutional Neural Network

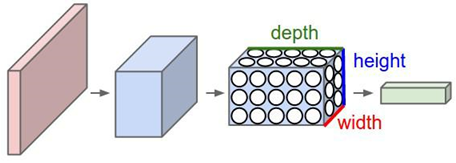
Convolutional Neural Networks (CNN - hay còn gọi là Mạng Nơron tích chập) có kiến trúc khá giống với mạng nơron truyền thống, chúng được cấu tạo từ rất nhiều nơron và có khả năng học dữ liệu nhờ thay đổi các trọng số và hệ số chênh lệch. Mỗi nơron nhận dữ liệu đầu vào từ nhiều nguồn, thực hiện các biến đổi nhân vô hướng.

Mạng CNN có đầu vào là một mảng các phần tử (có thể đại diện cho 1 ảnh), đầu ra của nó là xác suất tương ứng với mỗi lớp cần phân loại. Vì vậy mà cách huấn luyện trên mạng CNN không có nhiều thay đổi, vẫn sử dụng các hàm mất mát để đánh giá độ chính xác của mô hình tại lớp đầu ra cuối cùng.

Điểm đặc biệt của CNN là dữ liệu đầu vào của mạng có thể truyền dưới dạng ảnh, tức là dữ liệu dưới dạng ma trận. Các phép biến đổi trong mạng, ở đây là phép tích chập cũng được tích hợp dễ dàng và ít tham số hơn nhiều so với mạng nơron truyền thống.

Đối với mạng nơron truyền thống, lớp đầu vào nhận dữ liệu là một vector, tức mảng 1 chiều và biến đổi nó lần lượt qua mỗi lớp ẩn. Mỗi lớp ẩn được cấu tạo từ tập các nơron, mỗi nơron sẽ được kết nối toàn bộ từ các nơron của lớp trước đó. Mỗi nơron đều có tập các kết nối riêng biệt, không chia sẻ với các nơron khác, vì vậy mỗi noron sẽ có tập trọng số riêng của mình. Lớp cuối cùng được gọi là lớp đầu ra, trong bài toán phân loại, thì mỗi nơron trong lớp đầu ra sẽ biểu diễn xác suất là lớp tương ứng của dữ liệu đầu vào.

Mạng CNN được xây dựng và thiết kế để khắc phục nhược điểm tổng trọng số quá lớn của mạng nơron truyền thống. CNN có thể xử lý được dữ liệu ảnh và giữ nguyên cấu trúc của chúng, mà không phải biến đổi duỗi ma trận ảnh thành mảng một chiều. Cụ thể hơn, các nơron trong mạng CNN được sắp xếp trong không gian 3 chiều: chiều rộng, chiều cao và chiều sâu. Các nơron lớp ẩn không kết nối với toàn bộ nơron lớp trước mà chỉ cần kết nối tới một phần nơron nhỏ. Nhìn chung cứ qua mỗi lớp, dữ liệu giảm đi về kích thước chiều rộng, chiều cao và tăng về chiều sâu [[18](#Kri12)]. Sự biến đổi kích thước dữ liệu qua các lớp được biểu thị ở Hình 1.20.



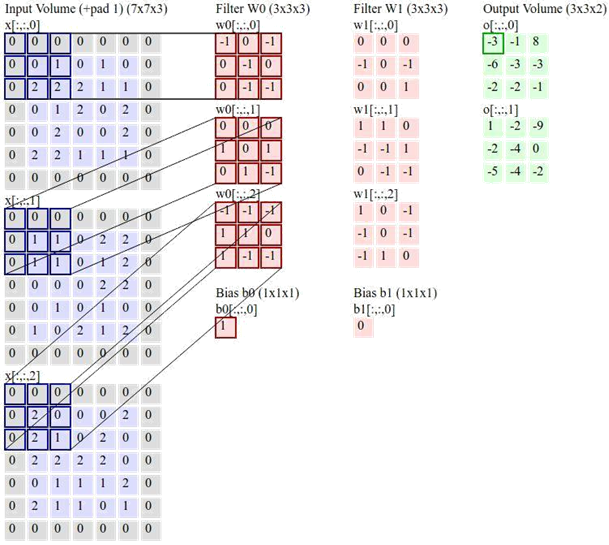
Hình 1.20 Sự biến đổi dữ liệu khi đi qua mạng CNN [[19](#Con12)]

Một mạng CNN đơn giản bao gồm chuỗi các lớp liên tiếp, mỗi lớp có hàm kích hoạt và cách biến đổi dữ liệu riêng của nó. Thường có 4 loại lớp chính để xây dựng một mạng CNN, bao gồm:

* Lớp tích chập (Convolutional layer)
* Lớp ReLU (ReLU layer)
* Lớp pooling (Pooling layer)
* Lớp kết nối toàn bộ (Fully-Connected layer)

***Lớp tích chập***

Như đã đề cập ở trên, điểm khác biệt giữa deep learning và machine learning, đó là mô hình mạng của deep learning có khả năng trích chọn đặc trưng tự động, mà không phải làm thủ công như ở các phương pháp, thuật toán ở machine learning. Đây cũng chính là nhiệm vụ, mục tiêu của lớp tích chập.



Hình 1.21 Minh hoạ phép tính tích chập trong mạng CNN [[20](#Der1B)]

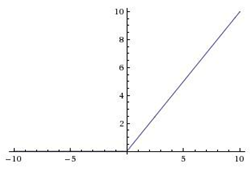
Hình 1.21 thể hiện các thành phần cũng như cách biến đổi, tính toán nằm trong lớp tích chập. Ảnh đầu vào được cho qua một bộ lọc chạy dọc bức ảnh. Bộ lọc có kích thước là (3x3 hoặc 5x5) và áp dụng phép tích vô hướng để tính toán, cho ra một giá trị duy nhất. Sau mỗi lần, bộ lọc lại dịch chuyển qua 1 vị trí khác lân cận, theo cả 2 chiều để áp dụng tiếp việc lọc. Đầu ra của phép tích chập là một tập các giá trị ảnh được gọi là mạng đặc trưng (features map).

Thực chất, ở các layer đầu tiên, phép tích chập đơn giản là phép tìm biên ảnh, sau khi cho qua bộ lọc nó sẽ làm hiện lên các đặc trưng của đối tượng trong ảnh như đường vẽ xung quanh đối tượng, các góc cạnh,... và các layer tiếp theo sẽ lại trích xuất tiếp các đặc trưng của các đối tượng đó, việc có nhiều layer như vậy cho phép chúng ta chia nhỏ đặc trưng của ảnh tới mức nhỏ nhất có thể [[18](#Kri12)]. Vì thế mới gọi là mạng đặc trưng.

Dưới đây là một số các khái niệm cơ bản của phần này:

* **Filter Kernel hay Feature Detector**: chúng đều là cách gọi của ma trận lọc. Thông thường, ở các lớp đầu tiên của Conv Layer sẽ có kích thước là [5x5x3]
* **Convolved Feature, Activation Map hay Feature Map**: là đầu ra của ảnh khi cho bộ lọc chạy hết bức ảnh với phép tích vô hướng.
* **Receptive field**: là vùng ảnh được chọn để tính tích chập, vùng này c ókích thước bằng đúng kích thước của bộ lọc.
* **Depth**: là số lượng bộ lọc. Lưu ý: ở đây là số lượng bộ lọc (filter) chứ không phảisố lượng kênh màu RGB trên ảnh.
* **Stride**: là khoảng cách dịch chuyển của bộ lọc sau mỗi lần tính. Ví dụ khi stride = 2. Tức sau khi tính xong tại 1 vùng ảnh, nó sẽ dịch sang phải 2 pixel. Tương tự với việc dịch xuống dưới.
* **Zero-Padding**: là việc thêm các giá trị 0 ở xung quanh biên ảnh, để đảm bảo phép tích chập được thực hiện đủ trên toàn ảnh.

***Lớp ReLU***



Hình 1.22 Đồ thị hàm ReLU [[21](#Bàilp)]

ReLU (Rectified Linear Unit) được sử dụng rộng rãi vì tính đơn giản. Đồ thị của hàm ReLU được minh họa trên Hình 1.22 và có công thức toán học:

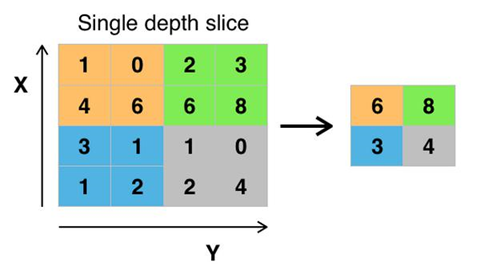
(1.9)

ReLU được chứng minh giúp cho việc training các Deep Networks nhanh hơn rất nhiều. Sự tăng tốc này được cho là vì ReLU được tính toán gần như tức thời và gradient của nó cũng được tính cực nhanh với gradient bằng 1 nếu đầu vào lớn hơn 0, bằng 0 nếu đầu vào nhỏ hơn 0 [[22](#Nai10)].

Mặc dù hàm ReLu không có đạo hàm tại s = 0, trong thực nghiệm, người ta vẫn thường định nghĩa ReLu(0) = 0 và khẳng định thêm rằng, xác suất để input của một unit bằng 0 là rất nhỏ.

Hàm ReLU có nhiều biến thể khác như Noisy ReLU, Leaky ReLu, ELUs, tuy nhiên đồ án sẽ không đề cập sâu đến các biến thể này.

***Lớp pooling***



Hình 1.23 Minh hoạ dữ liệu trước (trái) và sau (phải) khi áp dụng Max Pooling [[23](#CNNes)]

Lớp pooling thực hiện chức năng làm giảm kích thước đầu vào và giảm độ phức tạp tính toán của model ngoài ra lớp pooling còn giúp kiểm soát hiện tượng overffiting [[24](#Cir11)] Thông thường, pooling có nhiều hình thức khác nhau phù hợp cho nhiều bài toán, tuy nhiên có 2 phương pháp pooling được sử dụng rộng rãi và phổ biến, đó là:

**Max Pooling**: Thay thế vùng nhỏ trên dữ liệu bằng giá trị **lớn nhất** củavùng đó. Ý nghĩa của việc này là giữ lại những chi tiết có thông tin lớn nhất, quan trọng nhất để đại diện cho một vùng thông tin.

**Average Pooling**: Thay thế vùng nhỏ trên dữ liệu bằng giá trị **trung bình** của vùng đó. Ý nghĩa của việc này là lấy thông tin tổng thể để đại diện cho một vùng thông tin.

Ví dụ về max pooling với bộ lọc 2x2 và stride = 2. Bộ lọc sẽ chạy dọc ảnh vàvới mỗi vùng ảnh được chọn, sẽ chọn ra 1 giá trị lớn nhất và giữ lại. Hình 1.23 biểu diễn giá trị dữ liệu trước và sau khi áp dụng max pooling.

Lưu ý rằng, thông thường các lớp pooling có kích thước là 2 và stride = 2

Nếu lấy giá trị quá lớn, thay vì giảm tính toán nó lại làm phá vỡ cấu trúc ảnh và mất mát thông tin nghiêm trọng. Vì vậy mà một số chuyên gia không thích sử dụng layer này mà thay vào đó sử dụng thêm các lớp Conv Layer và tăng số stride lên mỗi lần.

***Lớp fully-conntected***

Tên tiếng Việt là Mạng liên kết đầy đủ. Tại lớp mạng này, mỗi một nơron của lớp này sẽ liên kết tới mọi nơron của lớp khác. Để đưa ảnh từ các lớp trước vào mạng này, buộc phải dàn phẳng bức ảnh ra thành mảng 1 chiều thay vì là mảng nhiều chiều như trước. Tại lớp cuối cùng sẽ sử dụng 1 hàm kinh điển trong học máy, đó là softmax để phân loại đối tượng dựa vào vector đặc trưng đã được tính toán của các lớp trước đó.

***Các mạng tiêu biểu***

Có rất nhiều kiến trúc mạng nổi tiếng thuộc nhóm mạng CNN mà được công bố và sử dụng rộng rãi, đồ án sẽ đề cập đến một số mạng thông dụng:

**LeNet** [[25](#LeC98)]: Đây là kiến trúc mạng đầu tiên áp dụng thành công mạng tích chập, được phát triển bởi Yann LeCun vào những năm 90 của thế kỷ trước. Ứng dụng được biết đến rộng rãi nhất của mạng LeNet đó là nhận diện, đọc các mã zip, ký tự số.

**AlexNet** [[18](#Kri12)]: Đây là công trình nghiên cứu tiên phong về việc áp dụng mạng tích chập trong lĩnh vực Thị giác máy tính (hay còn gọi là Computer Vision), mang tên AlexNet - được phát triển bởi Alex Krizhevsky, Ilya Sutskever và Geoff Hinton. Mạng AlexNet tham dự cuộc thi về thị giác máy tính hàng đầu thế giới - ImageNet ILSVRC vào năm 2012, kết quả mà AlexNet đạt được vô cùng ấn tượng, là một kỳ tích lúc bấy giờ khi mà sai số trên tập dữ liệu kiểm tra là 16%, bỏ xa đối thủ đứng thứ 2 với kết quả lên tới 26%. Kiến trúc mạng AlexNet khá tương đồng với mạng LeNet, tuy nhiên nó sâu hơn, to hơn và các lớp convolutional được nối tiếp nhau nhiều hơn (không như các kiến trúc cơ bản trước đây là sau mỗi lớp convolutional đều có một lớp pooling).

**ZF Net** [[26](#Zei14)]: Chức vô địch cuộc thi ImageNet ILSVRC năm 2013thuộc về mô hình mạng tích chập xây dựng bởi Matthew Zeiler và Rob Fergus. Nó dần nổi tiếng và được biết đến với cái tên ZFNet (viết tắt của Zeiler & Fergus Net). Kết quả của ZFNet đã được cải thiện đáng kể so với AlaxNet nhờ vào việc tinh chỉnh các cấu hình của kiến trúc mạng, cụ thể là mở rộng kích thước của các lớp convolutional ở giữa, đồng thời làm giảm kích thước bộ lọc và độ dịch của lớp convolutional đầu tiên, với mục đích làm giảm sự mất mát thông tin ở những lớp đầu.

**GoogLeNet** [[27](#Sze15)]: ImageNet ILSVRC tiếp tục là sân chơi hàng đầu để pháttriển các mạng nơron ngày càng chính xác hơn. Và chức vô địch năm 2014 vẫn là thuộc về một mạng tích chập, được xây dựng bởi Szegedy đến từ Google. Sự thay đổi và đóng góp chính của GoogLeNet đó là việc phát triển một biến thể trong mạng CNN, đó là Inception Module, với khả năng giảm đáng kể số lượng các trọng số của mô hình (xuống còn 4 triệu tham số, trong khi AlexNet có 60 triệu tham số). Hơn nữa ,công trình nghiên cứu này đặt một lớp Average Pooling tại đuôi của mạng CNN, thay vì một lớp fully-connected như trước, việc này giúp giảm đi đáng kể kích thước của dữ liệu, mà dường như không quá quan trọng và chứa quá nhiều thông tin. Có rất nhiều biến thể được phát triển theo mạng GoogLeNet, mà nổi bật trong số đó chính là mạng Inception-v4.

**VGGNet** [[28](#Sim14)]: Một mạng khác cũng giành vị trí chỉ ngay sau GoogLeNet tại cuộc thi ImageNet ILSVRC năm 2014 được phát triển bởi Karen Si-monyan và Andrew Zisserman, mà sau đó cũng khá phổ biến và được biết đến rộng rãi với cái tên VGGNet. Công trình nghiên cứu này đã chỉ ra rằng độ sâu của mô hình là yếu tố quan trọng tác động trực tiếp tới độ chính xác của mô hình, nếu mô hình càng sâu thì nó sẽ càng chính xác. Phiên bản tốt nhất của mạng VGGNet được phát triển bao gồm 16 cặp lớp CONV, FC nối tiếp nhau. Điều đặc biệt là họ chỉ sử dụng duy nhất một cấu hình là các lớp convolutional đều có bộ lọc kích thước 3x3 và các lớp pooling đều có bộ lọc kích thước 2x2. Tuy nhiên thì nhược điểm của mạng VGGNet là kích thước của nó quá lớn, chi phí tính toán khá đắt đỏ khi mà kích trước của nó lên tới 140 triệu trọng số. Và phần lớn trọng số này nằm ở các lớp FC phía đầu mạng. Thực nghiệm cũng chỉ ra rằng những lớp FC này có thể loại bỏ để giảm độ phức tạp của mô hình đi đáng kể, mà không giảm quá nhiều về độ chính xác của mạng.

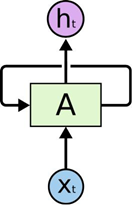
**ResNet** [[29](#HeK16)]: ResNet hay còn gọi là Residual Network được phát triển bởi Kaiming He, đã xuất sắc giành vị trí thứ nhất tại cuộc thi ImageNet ILSRC năm 2015. Ý tưởng của mạng Resnet đó là các kết nối không nhất thiết phải tuần tự giữa các lớp nối tiếp, ở đây, các kết nối có thể tồn tại giữa các lớp cách xa nhau. Mạng ResNet còn sử dụng thêm các lớp Batch Normalization, và các lớp FC ở cuối mạng cũng được bỏ đi. Hiện tại ResNet là một trong những mạng vô cùng phổ biến và được sử dụng rộng rãi trong các bài xử lý ảnh. ResNet được phát triển với nhiều cấu hình ResNet-18, ResNet-34, ResNet-50, ResNet-101 với các độ phức tạp khác nhau để người dùng có thể linh hoạt áp dụng vào bài toán của mình.

### Recurrent neural network

Con người không bắt đầu suy nghĩ của họ từ đầu tại tất cả các thời điểm. Khi đọc một vấn đề mới, để có thể hiểu được nó thì chúng ta phải vận dụng những kiến thức sẵn có trước đây của mình.

Tuy nhiên các mô hình mạng nơron truyền thống thì không thể làm được việc đó và đây có thể coi là một khuyết điểm chính của mạng nơron truyền thống. Ví dụ, ta muốn phân loại các bối cảnh xảy ra ở tất cả các thời điểm trong một bộ phim, thì rất khó để hiểu được tình huống hiện tại khi mà không có thông tin từ các tình huống trước.

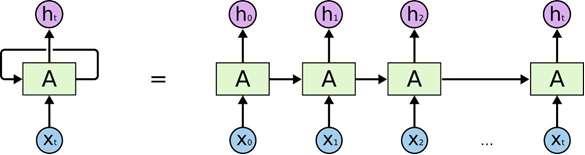
Mạng nơron hồi quy (RNN - Recurrent Neural Network) sinh ra để giải quyết vấn đề này. RNN này chứa các vòng lặp bên trong cho phép thông tin có thể lưu lại được [[30](#Kiế91)].



Hình 1.24 Kiến trúc mạng RNN [[31](#Und)]

Hình 1.24 mô tả một đoạn của mạng nơron hồi quy A với đầu vào là xt và đầu ra là ht. Một vòng lặp cho phép thông tin có thể được truyền từ bước này qua bước khác của mạng nơron.

Cấu trúc của RNN thực sự không khác nhiều so với các mạng nơron thuần. Một mạng nơron hồi quy có thể được coi là nhiều bản sao chép của cùng một mạng, trong đó mỗi đầu ra của mạng này là đầu vào của một mạng sao chép khác. Hình 1.25 mô tả rõ hơn về tính chất này.

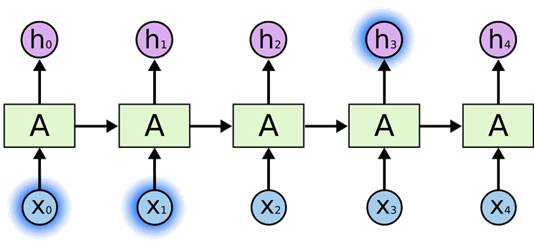


Hình 1.25 Diễn giải chi tiết kiến trúc mạng RNN [[31](#Und)]

***Hạn chế của RNN***

Một điểm nổi bật của RNN chính là ý tưởng kết nối các thông tin phía trước để dự đoán cho hiện tại. Việc này tương tự như ta xem một bộ phim, thực chất là một chuỗi các ảnh liên tiếp nhau, tuy nhiên thì để có thể hiểu toàn bộ nội dung của phim hay ý nghĩa của cảnh hiện tại, ta phải vận dụng trí nhớ và thông tin từ các cảnh phim trước đó mà đã xem qua.

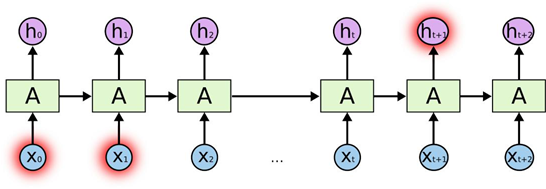
Đôi lúc ta chỉ cần xem lại thông tin vừa có thôi là đủ để biết được tình huống hiện tại. Một ví dụ trong mảng xử lý ngôn ngữ tự nhiên, ta có câu:“các đám mây trên bầu trời” thì chỉ cần đọc tới “các đám mây trên bầu” là đủ biết được chữ tiếp theo là “trời”. Trong tình huống này, khoảng cách tới thông tin có được cần để dự đoán là ngắn, nên RNN hoàn toàn có thể học được. Hình 1.26 thể hiện rằng nếu số lượng các lớp hồi quy ít, thì các lớp sau sẽ dễ dàng nhận được thông tin tường minh từ các lớp trước hơn, cụ thể lớp h3 sẽ có thông tin tường minh từ x0 và x1.



Hình 1.26 Minh hoạ sự phụ thuộc gần trong mạng RNN [[31](#Und)]

Như Hình 1.27, khi chuỗi các lớp đã quá dài, dữ liệu thông tin đầu ra tại *ht+1* sẽ có rất ít thông tin từ những lớp đầu như x0 và x1. Trong nhiều trường hợp như ví dụ ở trên đã trình bày, nếu không có những thông tin từ đầu này thì rất khó để có thể suy ra được kết quả đúng. Về mặt lý thuyết, rõ ràng là RNN có khả năng xử lý các phụ thuộc xa (long-term dependencies) bằng cách xem xét và cài đặt các tham số sao cho khéo léo. Tuy nhiên, trong thực tế RNN có vẻ không thể học được các tham số đó.

Vấn đề này đã được khám phá khá sâu bởi Hochreiter [[30](#Kiế91)] và Bengio, et al. [[32](#Ben94)].



Hình 1.27 Minh hoạ sự phụ thuộc xa trong mạng RNN [[31](#Und)]

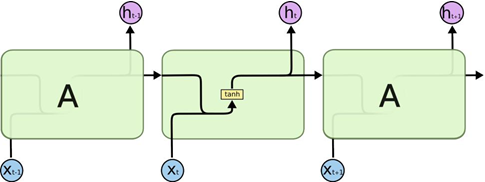
Vấn đề này tồn đọng trong mạng RNN mãi cho đến khi LSTM được phát triển. Ngay lập tức, LSTM đã chứng minh được sự hiệu quả của mình và được sử rộng rãi cho đến ngày nay. Phần sau đồ án sẽ nêu chi tiết hơn về mạng LSTM.

### Mạng LSTM

Mạng bộ nhớ dài-ngắn (Long Short Term Memory networks), thường được gọi là LSTM - một dạng đặc biệt của RNN, nó có khả năng học được các phụ thuộc xa. LSTM được giới thiệu bởi Hochreiter & Schmidhuber (1997) [[33](#Hoc97)], và sau đó đã được cải tiến và sử dụng bởi rất nhiều chuyên gia trong ngành AI. LSTM hoạt động cực kỳ hiệu quả trên nhiều bài toán khác nhau nên dần dần trở nên phổ biến.

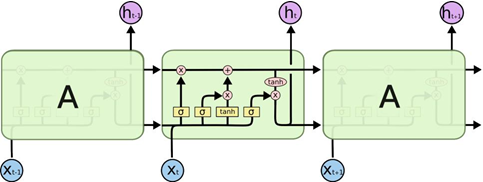
LSTM được thiết kế để tránh được vấn đề phụ thuộc xa (long-term depen-dency). Việc nhớ thông tin trong suốt thời gian dài là đặc tính mặc định, chứ không cần phải huấn luyện nó để có thể nhớ được. Tức là ngay nội tại của nó đã có thể ghi nhớ được mà không cần bất kỳ can thiệp nào.

Mọi mạng hồi quy đều có dạng là một chuỗi các mô-đun lặp đi lặp lại của mạng nơron. Với mạng RNN chuẩn, các mô-dun này có cấu trúc rất đơn giản, thường là một tầng tanh, thể hiện trong Hình 1.28.



Hình 1.28 Chi tiết kiến trúc của mô-đun đơn giản trong mạng RNN [[31](#Und)]

LSTM cũng có kiến trúc dạng chuỗi như vậy, tuy nhiên các mô-đun trong nó có cấu trúc khác, phức tạp hơn so với mạng RNN chuẩn. Thay vì chỉ có một tầng mạng nơron, chúng có tới 4 tầng tương tác với nhau một cách rất đặc biệt, thể hiện trong Hình 1.29.



Hình 1.29 Chi tiết kiến trúc mô-đun trong mạng LSTM [[31](#Und)]

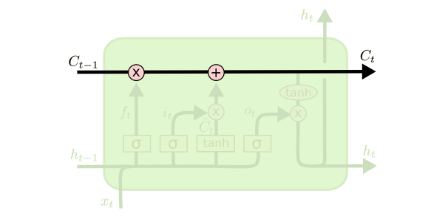
Các kí hiệu sử dụng trong lược đồ được thể hiện trong ảnh sau:



Hình 1.30 Các kí hiệu sử dụng trong lược đồ LSTM [[31](#Und)]

***Ý tưởng chính của LSTMs***

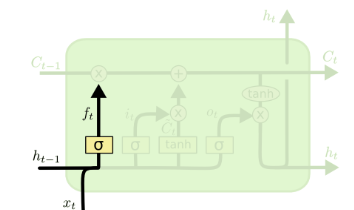
Chìa khóa của LSTMs là trạng thái của các cell, một đường ngang chạy qua đỉnh của lược đồ [[31](#Und)] thể hiện trong Hình 1.31.



Hình 1.31 Trạng thái các cell trong mạng LSTM [[31](#Und)]

Mạng LSTM có khả năng loại bỏ hoặc thêm các thông tin vào trạng thái các cell thông qua các cổng. LSTM có 3 loại cổng để bảo vệ và điều khiển trạng thái các cell. Chi tiết về hoạt động của mạng LSTM gồm 4 bước như sau:

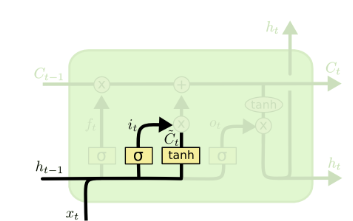
Bước đầu tiên là quyết định thông tin nào sẽ đi qua các cell state. Được đưa ra bởi một lớp sigmoid còn gọi là “forget gate layer” Hình 1.32. Cổng này xem xét *ht-1* và *xt* rồi đưa ra kết quả nằm trong khoảng 0 và 1 cho mỗi cell state *Ct-1* công thức 1.10. Với 1 có ý nghĩa là hoàn toàn giữ thông tin này trong khi 0 có nghĩa là hoàn toàn loại bỏ thông tin này.



Hình 1.32 Forgate gate trong LSTM [[31](#Und)]

*ft = σ(Wf ·[ht-1,xt] + bf )* (1.10)

Tiếp theo mạng sẽ quyết định thông tin mới nào cũng sẽ được lưu trong cell state. Bước này gồm 2 phần. 1 sigmoid layer gọi là “input gate layer” quyết định các giá trị nào sẽ được cập nhật theo công thức 1.11. Một tanh layer tạo 1 vector các giá trị ứng cử viên mới công thức 1.12 sẽ được thêm vào các trạng thái Hình 1.33.

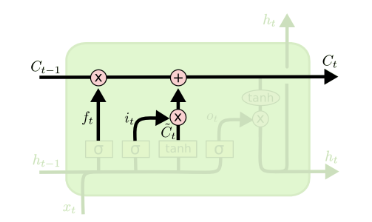


Hình 1.33 Input gate Layer và ứng cử viên sẽ được cập nhật vào cell Ct [[31](#Und)]

*it = (Wi · [ht-1, xt] + bi)* (1.11)

*(Wc·[ht-1, xt]+bC)* (1.12)

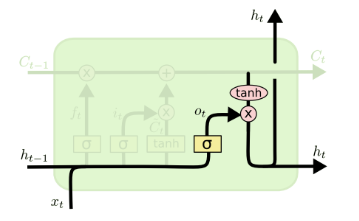
Bước thứ ba là lúc mạng thực sự quên những thông tin không cần thiết và cập nhật lại giá trị state cell Ct-1vào cell state mới Ct. Chúng ta nhân giá trị Ct-1  sau thêm it\* (1.13)



Hình 1.34 Thực thi quá trình quên và cập nhật thông tin mới trong LSTM [[31](#Und)]

*Ct = ft \* Ct-1 + it\** (1.13)

Cuối cùng đầu ra sẽ được dựa trên cell state nhưng sẽ là một phiên bản được lọc. Đầu tiên chúng ta có 1 sigmoid layer quyết định những phần nào của cell state sẽ được đưa ra đầu ra (công thức 1.14). Sau đó chúng ta đưa cell state thông qua tanh (để đưa các giá trị vào trong khoảng -1 và 1) và nhân nó bởi đầu ra của cổng sigmoid (công thức 1.15), để chỉ đưa ra đầu ra phần chúng ta đã quyết định Hình 1.35.



Hình 1.35 Tính toán đầu ra cho mạng LSTM [23]

(1.14)

(1.15)

## Tổng kết chương 1

Trong chương 1 đồ án đã trình bày về những đặc tính về các loại âm phổi, những phương pháp đã được sử dụng trước đây để phát hiện tiếng phổi khò khè. Sau đó đồ án tiếp tục trình bày những khái niệm, kỹ thuật cơ bản về học máy và học sâu.

Mục tiêu của đồ án là xây dựng một mô hình học máy có khả năng tự động phát hiện hơi thở khò khè với tiếng thở thông thường với độ chính xác cao, nhanh để có thể cung cấp thông tin bất thường về sức khỏe kịp thời nhất. Dữ liệu âm thanh là loại dữ liệu mang rất nhiều thông tin như vậy cần có các cách xử lý, trích chọn đặc trưng phù hợp để nén dữ liệu âm thanh, chỉ giữ lại những thông tin có ích cho bài toán. Các mô hình học sâu thông thường có số lượng tham số khổng lồ yêu cầu rất lớn tài nguyên tính toán và lưu trữ. Chính vì vậy đồ án cần xây dựng mô hình càng đơn giản càng tốt nhưng vẫn đảm bảo hiệu phân loại ở mức chấp nhận được.Chi tiết quá trình tiền xử lý âm thanh, trích chọn đặc trưng và đề xuất các mô hình học sâu được trình bày trong chương 2.

# MÔ HÌNH HỌC SÂU PHÁT HIỆN TIẾNG THỞ KHÒ KHÈ

Trong chương này, đồ án sẽ trình bày về các đặc trưng được trích chọn, hình trạng chi tiết về cách áp dụng học sâu vào việc tự động phát hiện tiếng thở khò khè qua các phần:

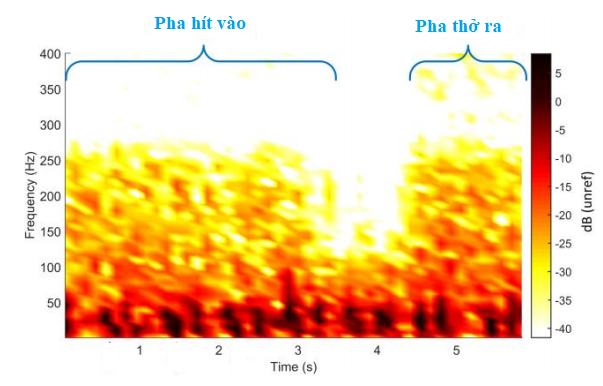
* Tiền xử lý dữ liệu
* Phân đoạn dữ liệu
* Trích chọn đặc trưng
* Đề xuất mô hình học sâu
* Tích hợp mô hình đã huấn luyện

## Tiền xử lý dữ liệu

Tất cả dữ liệu âm thanh đều được chuyển về cùng định dạng để dễ dàng cho các bước xử lý sau. Thông tin cụ thể như sau:

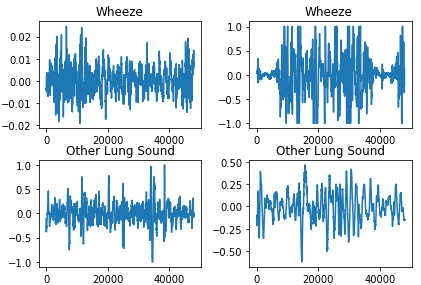
* **Định dạng (Format):** WAV – dạng dữ liệu thô nguyên bản, không có nén thông tin.
* **Mã hóa (Sample Encoding):** Mỗi mẫu dữ liệu được lưu bằng một số thực 32 bit được chuẩn hóa nằm trong khoảng -1 đến 1.
* **Tần số lấy mẫu (Sample Rate):** 16000Hz – cứ 1 giây sẽ có 16000 mẫu âm thanh được lưu trữ.

Trong 1 nhịp thở, thường thì pha hít vào sẽ có thời lượng trung bình là 3 giây pha thở ra là 2 giây ngoài ra ở giữa pha hít vào và thở ra sẽ có một khoảng trống giữa hai pha. Nên để hoàn thành 1 chu kì hít và thở sẽ tốn trung bình khoảng 6 giây.



Hình 2.1 Phân tích một chu kì thở [[34](#Mar17)]

Vì các âm thở bất bình thường có thể xuất hiện trong cả pha hít vào và thở ra. Nên dữ liệu âm thanh sẽ được cắt thành các đoạn nhỏ 3 giây để xử lý. Việc này đảm bảo kích thước các mẫu âm thanh là như nhau và trong mỗi mẫu đủ thông tin để có thể phân loại được, ngoài ra số lượng dữ liệu có được so với cắt thành các mẫu có độ dài tương ứng với 1 nhịp thở (trung bình 5.5 giây đến 6 giây) sẽ lớn hơn nhiều.Việc có thêm dữ liệu sẽ giúp các mô hình học máy học tốt hơn. Một số mẫu dữ liệu ở miền thời gian được thể hiện trong Hình 2.2



Hình 2.2 Biểu diễn một số mẫu âm thanh trên miền thời gian

Vì âm phổi được thu chủ yếu ở ngực trước và sau ngực, nên các bản ghi âm sẽ bị lẫn cả âm thanh của tim. Âm thở khò khè có tần số lớn hơn 100Hz trong khi tiếng tim có tần số từ 50 – 150Hz. Chính vì thế bộ lọc highpass Butterworth thứ tự số 9 với tần số cut off là 100Hz được sử dụng để giảm bớt ảnh hưởng của tiếng tim trong tín hiệu.

## Trích xuất đặc trưng

Mạng CNN ra đời với ý tưởng và mục đích để xử lý trong mảng thị giác máy tính, cho kết quả vượt trội trong các bài toán về xử lỳ ảnh. Đối với dữ liệu âm thanh ta có thể sử dụng các phương pháp trích xuất đặc trưng để có thể biểu diễn một mẫu âm thanh hơi thở dưới định dạng dữ liệu giống như một ảnh.

Mạng bộ nhớ dài ngắn (LSTM) đặc biệt hiệu quả trong các bài toán về xử lý các chuỗi thời gian. Khắc phục được điểm yếu của RNN như đã phân tích trong phần 1. Do có các cổng giúp mạng đánh giá được mức độ quan trọng của thông tin, từ đó đưa ra quyết định giữ lại hay bỏ đi thông tin đó. Nhờ cơ chế này các thông tin quan trọng có thể truyền đi rất xa và có ảnh hưởng đến tương lai. Để sử dụng mạng này chúng ta cần biến đổi âm thanh thành các đặc trưng để phù hợp với hình trạng của mạng.

Các dạng đặc trưng mà đồ án đề xuất áp dụng bao gồm:

* Spectrogram
* Mel-frequency cepstral coefficients (MFCC)
* Gammatone frequency cepstral coefficients (GFCC)

### Spectrogram

Spectrogram là biểu diễn của âm thanh dưới dạng tần số thời gian bằng không gian 3 chiều, với trục Oy thể hiện tần số, trục Ox biểu diễn thời gian, màu sắc biểu diễn âm độ của âm thanh[[35](#Spe)]. Spectrogram thu được bằng phép biến đổi fourier thời gian ngắn (STFT). Một đoạn âm thanh dài sẽ được chia thành các frame nhỏ, mỗi frame sau đó sẽ được áp dụng một hàm cửa sổ trước khi thực hiện phép biến đổi fourier rời rạc (DFT).Kết quả sẽ thu được biểu diễn tần số của frame đó và trở thành 1 cột trong kết quả của biến đổi STFT. Sau khi thực hiện biến đổi trên toàn bộ frame của đoạn âm thanh, ta sẽ thu được biểu diễn tần số thời gian của đoạn âm thanh [[36](#Sho)].

Biến đổi STFT được mô tả bằng công thức 2.1. Spectrogram thu được bằng bình phương trị tuyệt đối của kết quả phép biến đổi STFT:

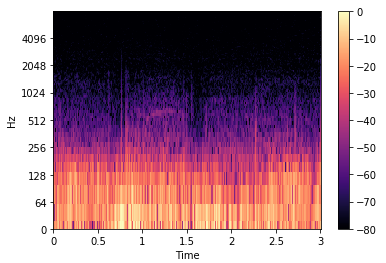
(2.1)

Trong đó:

* *x[n]* là tín hiệu được chuyển đổi
* *w[n]* là cửa sổ trượt qua mỗi frame.

Cụ thể trong đề tài mỗi frame sẽ được sẽ có độ dài 1024 mẫu , 2 frame cạnh nhau sẽ chồng lấn lên nhau 512 mẫu nhằm tránh mất mát dữ liệu nằm ở biên). Các frame sẽ được áp dụng cửa sổ Hamming trước khi thực hiện biến đổi DFT.

Với các thông số như trên 1 mẫu âm thanh 3 giây sẽ thu được ma trận có kích thước 513 x 94 là kết quả của biến đổi STFT. Kết quả này sẽ được lấy bình phương trị tuyệt đối để thu được Spectrogram. Spectrogram của một mẫu thở khò khè được minh họa trong Hình 2.3.



Hình 2.3 Spectrogram của một mẫu thở khò khè

Spectrogram là biểu diễn của tín hiệu âm thanh ở miền tần số thời gian. Lượng thông tin chứa trong spectrogram sấp xỉ với lượng tín hiệu thô ở miền thời gian. Sử dụng đặc trưng spectrogram có thể tận dụng được khả năng trích chọn đặc trưng tự động của mô hình học sâu.

Tuy nhiên sử dụng Spectrogram có một số nhược điểm như để mạng noron có thể hiểu và học spectrogram đòi hỏi một số lượng lớn dữ liệu huấn luyện và mạng noron đủ phức tạp để có thể thích nghi với dữ liệu. Như vậy rất tốn tài nguyên lưu trữ và tính toán để đạt được kết quả mong muốn.

Khi lượng dữ liệu ít hoặc không đủ tài nguyên để huấn luyện, trích chọn đặc trưng thủ công trước khi đưa vào mạng học sâu là cần thiết để tăng độ chính xác của mạng. Đồ án đề xuất trích xuất các đặc trưng MFCC và GFCC là các đặc trưng được sử dụng phổ biến và rộng rãi trong lĩnh vực xử lý tiếng nói.

### Mel-frequency cepstral coefficients

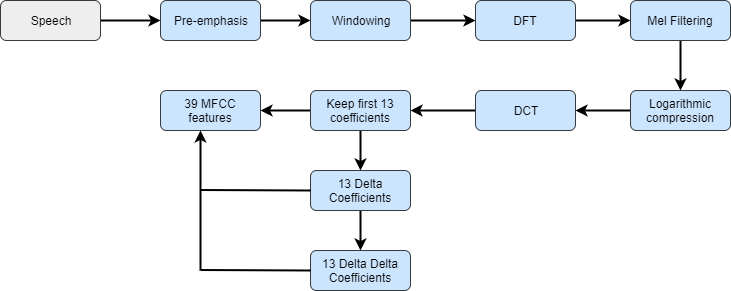
Thang Mel là miền tần số của âm thanh, đặc trưng cho sự cảm nhận âm thanh, âm lượng của 1 giai điệu. Tai người vô cùng nhạy cảm với những sự thay đổi nhỏ về tần số ở miền tần số thấp và độ nhạy sẽ giảm dần khi miền tần số tăng cao. Đây cũng chính là ý tưởng để chúng ta trích xuất đặc trưng MFCC, chỉ giữ lại những âm thanh nằm trong vùng tần số mà tai người nhạy cảm, để giảm độ lớn dữ liệu, qua đó giảm độ phức tạp tính toán.

Công thức 2.2 chuyển đổi từ tần số sang thang Mel và 2.3 dùng để chuyển đổi ngược lại.

(2.2)

(2.3)

Quá trình trích xuất đặc trưng từ dữ liệu âm thanh thành các hệ số Mel [[37](#MFCcs)] cụ thể được thể hiện trong Hình 2.4:

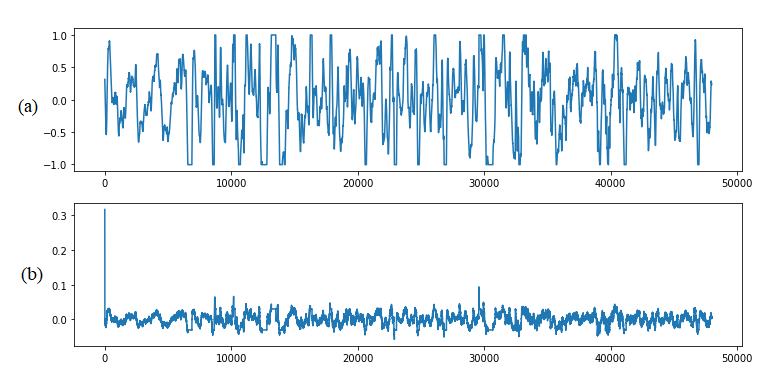


Hình 2.4 Lược đồ trích xuất đặc trưng MFCC trong đồ án

***Pre-emphasis***

Đầu tiên ta áp dụng 1 bộ lọc pre-emphasis để khuếch đại các tần số cao. Do âm thanh thu được có thể có nhiều âm thanh nhiễu từ môi trường. Bộ lọc pre-emphasis áp dụng vào tín hiệu x sử dụng công thức 2.4. Biểu diễn trên miền thời gian của tiếng thở khò khè trước và sau khi đi qua bộ lọc Pre-emphasis được thể hiện trong Hình 2.5:

(2.4)



Hình 2.5 (a) Mẫu thở khò khè nguyên bản (b) Mẫu thở khò khè sau khi được áp dụng bộ lọc pre-emphasis

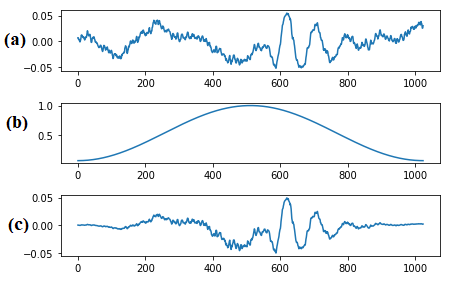
***Framing***

Do dữ liệu âm thanh thay đổi liên tục trong khoảng thời gian rất nhỏ, vậy nên lượng thông tin khác nhau trong mỗi bản ghi của chúng ta là vô cùng lớn.Vì vậy chúng ta sẽ giả sử rằng trên mỗi khoảng thời gian nhỏ, tín hiệu âm thanh không thay đổi quá nhiều và chúng ta có thể khái quát thông tin của nó bằng giá trị.Trong quá trình thực nghiệm dữ liệu âm thanh được chia thành các đoạn có độ dài 1024 mẫu. Các frame chồng lấn lên nhau 512 mẫu. Như vậy với một đoạn dữ liệu dài 3 giây. Mỗi mẫu sẽ được tách thành 93 frame.

***Windowing***

Tiếp theo chúng ta sẽ tiến hành làm mịn vùng giá trị cho từng frame bằng cách nhân toàn bộ giá trị dữ liệu của frame với cửa sổ Hamming được cho bởi công thức 2.5. Kết quả của bước này được mô tả thể hiện trong Hình 2.6.

(2.5)



Hình 2.6 (a) Biểu diễn miền thời gian của 1 frame (b) Biểu diễn miền thời gian của cửa sổ Hamming (c) Biểu diễn miền thời gian của frame sau khi áp dụng của sổ Hamming

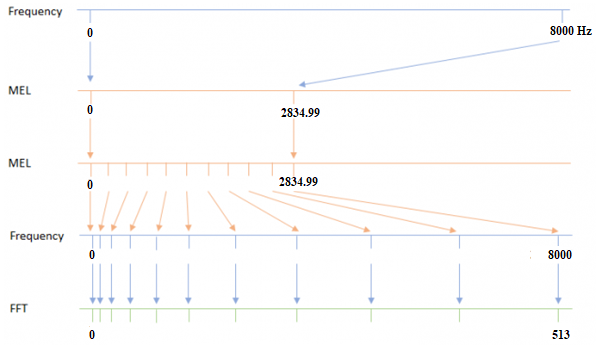
***Tính năng lượng phổ***

Sau đó ta áp dụng biến đổi DFT 2.6 lên kết quả của bước trước để thu được biểu diễn ở miền tần số của tín hiệu. Tiếp theo ước tính năng lượng phổ được thu được bằng cách bình phương trị tuyệt đối kết quả của biến đổi DFT.

(2.6)

***Mel Filtering***

Phổ tín hiệu tần số thu được từ bước trước sẽ được đưa qua bộ lọc thang Mel.

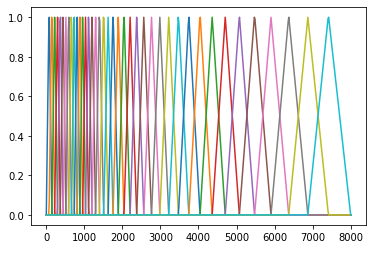


Hình 2.7 Quá trình xây dựng ngân hàng bộ lọc Mel

Cách xây dựng bộ lọc được sử dụng trong đề tài được thể hiện trong Hình 2.7 cụ thể như sau. Do tần số lấy mẫu của các file âm thanh là 16000Hz. Theo tần số Nyquist [[38](#Nyq)] tần số lớn nhất có thể có trong tín hiệu sẽ là 8000Hz. Như vậy ta sẽ xây dựng các bộ lọc trong khoảng tần số từ 0Hz đến 8000Hz.

Đầu tiên chúng ta chuyển tần số này sang thang Mel dựa theo công thức 2.2: 0Hz ứng với giá trị 0 trên thang Mel, 8000 ứng với giá trị 2834.99. Số lượng bộ lọc được sử dụng trong đề tài là 40 bộ lọc. Nên chúng ta cần thêm 40 điểm có khoảng cách tuyến tính nằm trong khoảng 0 – 2834.99. Vậy tổng cộng ta có 42 điểm được đo ở thang Mel là: 0, 69.14, 138.28, 207.42, …, 2834.99.Cuối cùng chúng ta chuyển đổi ngược lại giá giá trị Hz gọi là h(i). Rồi sẽ chuyển đổi xem tần số nào thuộc bin nào của DFT theo công thức 2.7. 40 bộ lọc Mel được sử dụng trong đề tài được biểu diễn trong Hình 2.8.

(2.7)

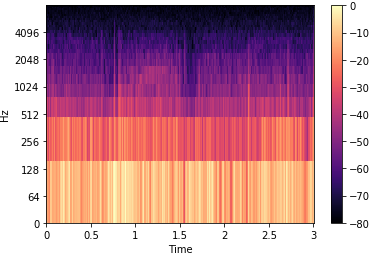


Hình 2.8 40 bộ lọc thang Mel được sử dụng

Do ta thực hiện biến đổi DFT trên 1024 mẫu nên ta sẽ thu được 513 frequency bin. Bộ lọc thang mel thu được sử dụng công thức 2.8.

*Hm(k)* (2.8)

Sau khi đưa tín hiệu qua ngân hàng bộ lọc thang Mel, chúng ta thu được Mel spectrogram Hình 2.9.

****

Hình 2.9 Mel spectrogram của một mẫu thở khò khè

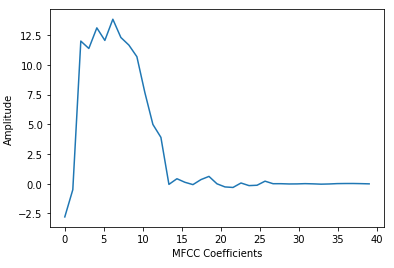
***Logarithmic compression***

Dữ liệu dạng vector thu được là những miền giá trị nằm trong bộ lọc được lấy logarithm với mục đích mô phỏng lại sự cảm nhận độ to của âm thanh tai người. Đồng thời việc lấy logarithm cũng là làm giảm độ lớn dữ liệu đi đáng kể, giảm độ phức tạp tính toán về sau.

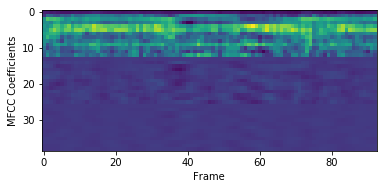
***DCT***

DCT, viết tắt của Discrete Cosine Transformation, còn được gọi là biến đổi cosine rời rạc Đây là công đoạn cuối cùng trong quá trình trích xuất đặng trưng MFCC. Biến đổi cosine cho phép tính toán, rút ra các hệ số tương ứng với các bậc khác nhau từ dãy dữ liệu ở công đoạn trước.

Đồ án đề xuất lấy 13 hệ số đặc trưng MFCC và 13 hệ số delta, 13 hệ số delta delta lần lượt là đạo hàm của 13 hệ số MFCC và hệ số delta với số lượng các frame xem xét tính đạo hàm là 10. Như vậy 1 mẫu âm thanh sẽ được đặc trưng bởi 1 ma trận đặc trưng GFCC kích thước 93 x 39.Hình 2.10, Hình 2.11 lần lượt là biểu diễn 39 hệ số MFCC của 1 frame và minh họa ma trận đặc trưng MFCC của 1 tập tin thở khò khè bằng matplotlib [[39](#Matrg)].



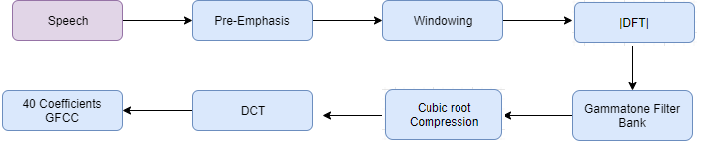
Hình 2.10 39 hệ số MFCC của 1 frame



Hình 2.11 Biểu diễn ma trận đặc trừng MFCC của một mẫu thở khò khè

### Gammatone frequency cepstral coefficients (GFCC)

Một trong những vấn đề lớn của của nhận dạng âm thanh là nhiễu. Độ nhạy với nhiễu là nhược điểm chính của MFCC. GFCC là đặc trưng âm thanh dựa trên 1 tập các gammatone filter bank[[40](#Gamer)]. Gammatone filter chính là mô phỏng sự phân tích tần số của màng đáy để biến sự dao động của âm thanh thành tín hiệu noron. Lược đồ trích xuất đặc trưng GFCC được thể hiện trong Hình 2.12.



Hình 2.12 Lược đồ trích xuất đặc trưng GFCC

Về cơ bản việc trích xuất đặc trưng GFCC tương tự như trích xuất MFCC tuy nhiên thay vì đi qua ngân hàng bộ lọc Mel thì ngân hàng bộ lọc Gammatone được sử dụng [[41](#HXu12)] và nén dữ liệu sử dụng hàm lấy căn thay vì hàm log như trong MFCC.

Bộ lọc Gammatone có đáp ứng xung được mô tả bằng công thức sau:

(2.9)

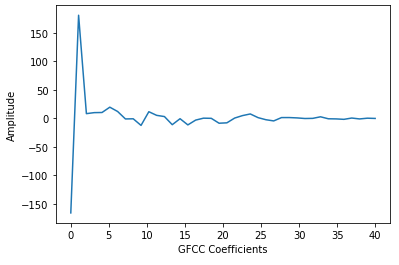
Trong đó t là thời gian, giá trị α = 4 là thứ tự của bộ lọc, b là băng thông của bộ lọc chữ nhật với tần số trung tâm f.

Các tần số trung tâm phân phối đều trên thang ERB [[42](#Equth)] liên hệ với tham số b trong công thức 2.9 như sau:

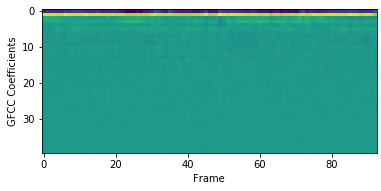
(2.10)

(2.11)

Số lượng bộ lọc được sử dụng trong đề tài là 40 bộ lọc và 40 hệ số được giữ lại để huấn luyện mô hình. Như vậy 1 mẫu âm thanh sẽ được biểu diễn bằng 1 ma trận đặc trưng GFCC có kích thước 93x40. Biểu diễn 40 hệ số GFCC và ma trận đặc trưng GFCC của 1 âm thở khò khè và crackles lần lượt được minh họa bằng matplotlib [[39](#Matrg)] trong Hình 2.13, Hình 2.14.



Hình 2.13 40 hệ số GFCC của 1 frame



Hình 2.14 Biểu diễn ma trận đặc trưng GFCC của 1 mẫu thở khò khè

## Huấn luyện mô hình học sâu cho phát hiện tiếng thở khò khè

Ở phần trước, đồ án đã trình bày các đặc trưng trích chọn cho một mẫu âm thanh. Trong phần này, đồ án sẽ trình bày kiến trúc các mô hình học sâu được sử dụng để phát hiện tiếng thở khò khè dựa trên những đặc trưng được trích chọn.

Hiện nay việc áp dụng các mạng noron tích chập để học trên spectrogram là vô cùng phổ biến trong xử lý âm thanh. Vì vậy đồ án đề xuất sử dụng mạng CNN trích chọn đặc trưng tự động trên spectrogram là biểu diễn của âm thanh ở miền tần số thời gian.

Đối với đặc trưng MFCC, GFCC, kết hợp MFCC và GFCC đồ án đề xuất sử dụng mạng LSTM để học.

### Chuẩn hóa các đặc trưng

Sau khi thu được các đặc trưng ở bước trước để trở thành đầu vào cho các mô hình học sâu. Ta nên chuẩn hóa các đặc trưng để nó có tính chất của phân phối chuẩn với µ = 0 và σ = 1. Trong đó σ là độ lệch chuẩn và µ là giá trị trung bình. Cụ thể ta sẽ chuẩn hóa sử dụng z score với công thức 2.12:

(2.12)

Sau khi chuẩn hóa các đặc trưng sẽ nằm xung quanh 0 với độ lệch chuẩn là 1. Việc này sẽ đưa các đặc trưng về cùng một khoảng như vậy mạng học sâu sẽ hội tụ nhanh hơn.

Vì các đặc trưng x đóng vai trò trong việc cập nhật trọng số ở các thuật toán tối ưu, nếu các đặc trưng ở các khoảng khác nhau sẽ gây ra việc có những trọng số sẽ được cập nhật nhanh hơn các trọng số khác.

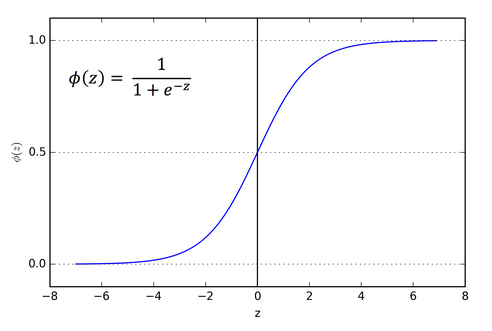
Sau khi chuẩn hóa dữ liệu sử dụng z score ta cần lưu lại giá trị µ, σ của tập dữ liệu để chuẩn hóa dữ liệu kiểm tra vào thời điểm sử dụng mô hình để suy luận.

### Hàm mất mát và phương pháp tối ưu trên mạng học sâu

Do mô hình thực hiện phân loại trên hai nhãn, nên output layer sẽ gồm 1 noron với ý nghĩa là xác suất nó có phải thở khò khè hay không. Do xác suất sẽ nằm trong khoảng [0, 1] nên hàm sigmoid (2.13) được sử dụng để làm hàm kích hoạt cho lớp đầu ra.

(2.13)

Hàm sigmoid Hình 2.15 sẽ chuẩn hóa giá trị của lớp đầu ra nằm trong khoảng [0, 1].



Hình 2.15 Đồ thị hàm sigmoid [[43](#Logon)]

Song song với đó hàm mất mát được sử dụng là hàm Binary Cross Entropy được mô tả trong công thức sau:

(2.14)

Trong đó , lần lượt là kết quả của lớp đâu ra và nhãn của bản ghi thứ i.

Quá trình huấn luyện mạng học sâu là quá trình điều chỉnh các trọng số dựa trên thuật toán tối ưu và dữ liệu đưa vào để học. Thuật toán tối ưu được sử dụng trong đồ án là thuật toán tối ưu adam. Khai thác exponential weighted average (EWA) là kỹ thuật đánh giá xu hướng di chuyển của một tham số. EWA được điều khiển bởi tham số β, giá trị EWA tại thời điểm t có giá trị xấp xỉ giá trị trung bình của 1 / (1 – β) giá trị ngay trước t. Tại thời điểm t EWA được tính theo công thức sau:

(2.15)

Thuật toán tối ưu adam sử dụng 2 moment:

(2.16)

(2.17)

Các tham số được cập nhật theo công thức 2.18. Mẫu số của 2.18 được cộng với giá trị để tránh tình trạng chia cho 0:

(2.18)

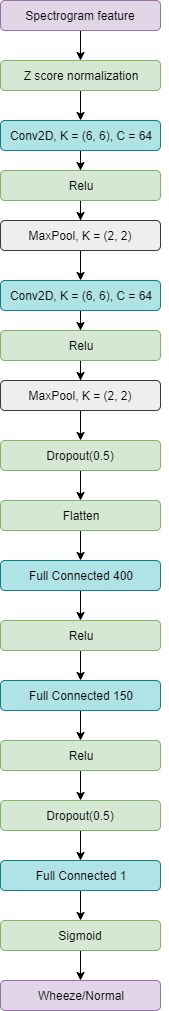
Với α là tốc độ học, quyết định mạng sẽ học nhanh hay chậm.

Sử dụng thuật toán tối ưu adam sẽ giảm sự dao động khi cập nhật trọng số, giúp cho việc hội tụ điểm tối ưu nhanh hơn so với thuật toán hạ gradient thông thường. Trong đồ án tham số lần lượt có giá trị là 0.9 và 0.999.

Ngoài ra trong quá trình huấn luyện, đồ án còn sử dụng thêm learning rate decay sau một vài epoch nếu giá trị hàm chi phí không được cải thiện, tốc độ học sẽ tự động giảm xuống để có thể đạt được đến kết quả tối ưu hơn. Cụ thể trong vòng 5 epoch nếu hàm chi phí không được cải thiện tốc độ học sẽ giảm xuống còn 90% tốc độ học hiện tại đối với mạng CNN và 80% đối với mạng LSTM. Chi tiết về các mạng được CNN và LSTM được trình bày ngay dưới đây.

### Huấn luyện mạng CNN cơ bản (Convolution Neural Network)

Hình trạng của mạng CNN sử dụng trong đề tài được thể hiện ở Hình 2.16:



Hình 2.16 Hình trạng mạng của mạng CNN cơ bản để phát hiện tiếng thở khò khè

Đầu vào của mạng là spectrogram được chuẩn hóa sử dụng z-score. Mạng CNN có cấu trúc tuần tự như sau:

* Ban đầu dữ liệu đi qua lớp tích chập với 64 bộ lọc kích thước 6 \* 6 hàm kích hoạt là hàm Relu rồi đến lớp Max Pooling kích thước 2 \* 2. Qúa trình này diễn ra hai lần.
* Kết quả bước trên lần lượt đi qua lớp Dropout với xác suất Dropout là 0.5 để giảm tình trạng quá vừa dữ liệuvà phẳng hóa để đưa vào các lớp kết nối đầy đủ.
* Lớp kết nối đầy đủ bao gồm 2 lớp ẩn lần lượt là 400 và 150 noron sử dụng hàm kích hoạt là hàm Relu. Kết quả các bước trước đi qua lớp Dropout với xác suất Dropout là 0.5 rồi đi qua lớp đầu ra 1 noron với hàm kích hoạt là hàm sigmoid để đưa ra kết quả phân loại.

Mạng được tạo thông qua framework keras [[44](#Keras)] được thể hiện trong Hình 2.17.



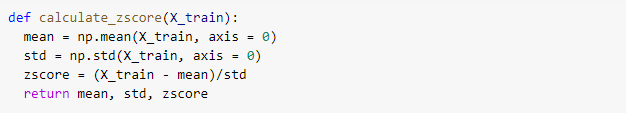
Hình 2.17 Mạng CNN được code bằng framework keras

Mạng tối ưu dựa trên thuật toán adam với kích thước batch là 64, tốc độ học khởi điểm là 0.0005. Số lượng tham số 60M. Quá trình học trên mạng CNN diễn ra như sau:

Dữ liệu spectrogram thu được coi như một ảnh xám (ảnh có số chiều là 1 thay cho ảnh có số chiều là 3 như ảnh RGB) được đưa vào một ma trận X\_train. Với mỗi hàng trên ma trận là một đặc trưng spectrogram.

Song song với đó nhãn được mã hóa dưới dạng one hot vector, có chỉ số hàng tương ứng với chỉ số hàng của spectrogram feature trong ma trận đặc trưng X\_train. Do đây là bài toán phân loại hai nhãn nên nhãn có giá trị 0 tương ứng với thở khò khè và 1 tương ứng với thở bình thường.

Tiếp theo đó dữ liệu được chuẩn hóa sử dụng zscore theo công thức 2.12 theo từng pixel trên ảnh spectrogram. Ta sẽ sử dụng thư viện numpy [[45](#NumPy)] để hỗ trợ tính giá trị trung bình và độ lệch chuẩn. Giá trị trung bình và độ lệch chuẩn cũng cần được lưu lại để chuẩn hóa dữ liệu kiểm chứng trước khi đưa ra dự đoán.



Hình 2.18 Chuẩn hóa spectrogram bằng zscore

Sau khi xử lý xong về mặt dữ liệu, dữ liệu được chia thành các batch có kích thước là 64, mạng sẽ cập nhật trọng số sau khi hoàn thành 1 batch.

Mỗi một batch đưa vào huấn luyện sẽ được thông qua hai quá trình là quá trình lan truyền xuôi và lan truyền ngược.

Ban đầu các tham số của mạng được khởi tạo bằng cách lấy ngẫu nhiên từ phân phối đều trong khoảng giới hạn [-limit, limit] với limit được tính theo công thức:

(2.19)

Quá trình lan truyền xuôi là quá trình dữ liệu được đưa đưa thông qua các lớp của mạng. Với kích thước mỗi batch là 64 như trên mỗi một lần một tensor kích thước (64, 513, 94, 1) được đưa vào mạng. Sau đó tensor đầu vào được đưa qua lớp tích chập kích thước (6 x 6 x 1) đầu ra sẽ được cộng với một hằng số b trước khi đi qua hàm kích hoạt relu sau bước này kích thước của tensor giảm xuống còn (64, 508, 89, 64).

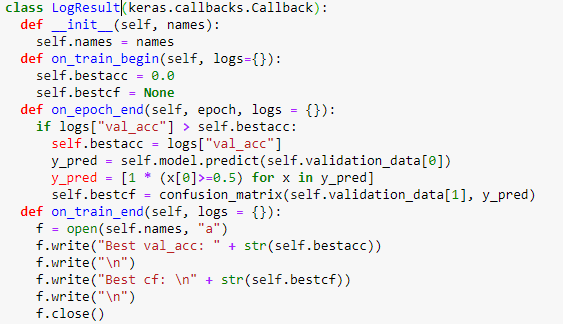
MaxPooling tiếp tục được sử dụng để giảm kích thước tensor xuống còn (64, 254, 44, 64). Tương tự như bước trước tensor được đi qua trước 64 bộ lọc kích thước (6 x 6 x 64) và lớp maxpooling kích thước (2 x 2 x 64). Trước khi đi qua lớp Flatten một lớp Dropout với xác suất dropout là 0.5 (tức là một noron có xác suất 0.5 bị vô hiệu hóa trong quá trình lan truyền xuôi và lan truyền ngược trong batch đó).

Cuối cùng dữ liệu được truyền qua 2 lớp kết nối đầy đủ và 1 lớp Dropout trước khi đi qua hàm sigmoid để đưa ra kết quả dự đoán cho batch này. Trung bình lỗi của từng batch sẽ được sử dụng để thực hiện quá trình lan truyền ngược nhằm cập nhật trọng số trong mạng.

(2.20)

Quá trình lan truyền ngược là quá trình các trọng số w ở từng lớp được tính đạo hàm từng phần với hàm chi phí, kí hiệu , sử dụng quy tắc chuỗi khi tính đạo hàm. Mạng CNN sử dụng trong đồ án sử dụng phương pháp tối ưu adam nên một số thông tin khác như EWA cũng được tính kết hợp với đạo hàm từng phần để tối ưu mạng. Cụ thể đã được trình bày trong phần 2.3.2.

Trong quá trình huấn luyện kết quả tốt nhất khi huấn luyện mạng sẽ được ghi lại bằng cách tạo mới một callback trong keras. Learning rate decay sử dụng sẵn callback ReduceLROnPlateau định nghĩa trước trong keras.



Hình 2.19 Callback LogResult được sử dụng trong keras

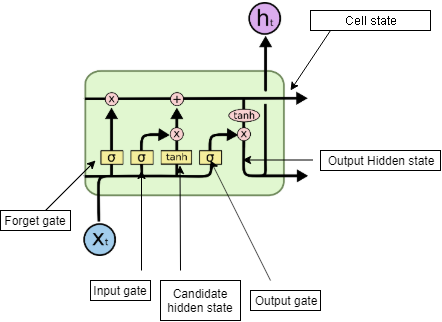
Quá trình lan truyền ngược được framework keras tính sẵn dựa trên cấu hình mạng được khai báo bằng cách sử dụng phương thức fit của mô hình.



Hình 2.20 Quá trình lan truyền ngược được thực thi trong framework keras

### Huấn luyện mô hình học sâu LSTM tự động phát hiện tiếng thở khò khè

Trong Hình 2.21 thể hiện cấu trúc của một đơn vị LSTM được sử dụng trong đồ án. Như đã trình bày ở phần Mạng LSTM, cấu trúc của một khối LSTM gồm 4 mạng noron tương ứng với input gate, forget gate, output gate và new candidate.

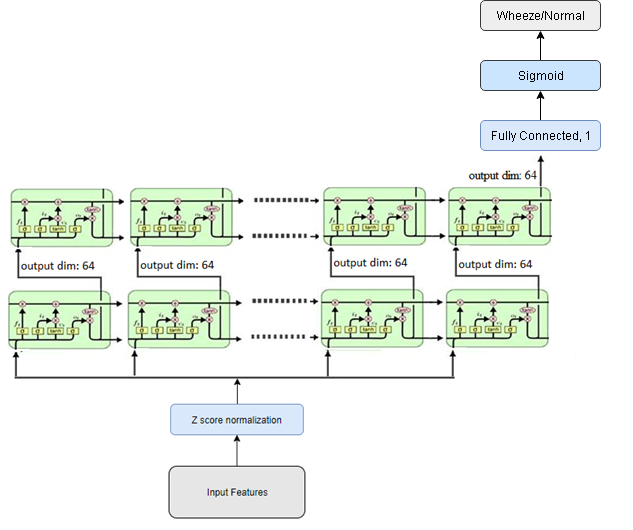


Hình 2.21 Cấu trúc một khối LSTM

Hàm kích hoạt cho input gate, forget gate, output gate là hàm hard sigmoid. Một xấp xỉ của hàm sigmoid được cho bởi công thức 2.21. Việc tính hàm hard sigmoid nhanh hơn nhiều so với hàm sigmoid do không phải thực hiện tính . Hàm kích hoạt cho Output Hidden State và Candidate hidden state là hàm Tanh.

(2.21)

Kích thước của output hidden state là 64. Sẽ là một phần đầu vào cho các khối LSTM phía sau. Hình trạng mạng đầy đủ của mạng LSTM thể hiện trong Hình 2.22



Hình 2.22 Hình trạng mạng LSTM sử dụng để phát hiện tiếng thở khò khè

Tương tự như trong mạng CNN, dữ liệu cũng được đặt vào các hàng của một ma trận X\_train rồi tiến hành chuẩn hóa sử dụng công thức 2.12 và one hot vector các nhãn.

Như đã phân tích ở trên mỗi một ma trận đặc trưng sẽ bao gồm 93 hàng tương ứng với số frame của bản ghi âm thanh đó, số cột tương ứng với số đặc trưng mô tả frame đó (40 đối với đặc trưng GFCC, 39 đối với đặc trưng MFCC, 79 với sự kết hợp của cả 2 đặc trưng MFCC và GFCC), đây cũng là kích thước đầu vào tại mỗi bước thời gian cho mạng LSTM.

Việc học trong mạng LSTM thông qua hai quá trình là lan truyền xuôi theo thời gian và lan truyền ngược theo thời gian. Do mạng phải đi lần lượt từ bước thời gian đầu tiên để tính output hidden state và cell state phục vụ quá trình tính toán cho các bước thời gian sau.

Cụ thể với kích thước mỗi batch được sử dụng trong đồ án là 64, để tiết kiệm tính toán một ma trận có kích thước (64, đặc trưng mô tả 1 frame) tương ứng với một batch được đưa vào bước thời gian đầu tiên của mạng để tính toán ra ngay kết quả của output gate và cell state của 64 mẫu thuộc batch này. Kết quả output gate sẽ và cell state có kích thước (64, 64) sẽ được chuyển vào làm một phần đầu vào cho bước thời gian thứ hai, ngoài ra kết quả của output gate cũng được đưa thành đầu vào của lớp LSTM ở phía trên có kết nối một một với lớp LSTM ở dưới.

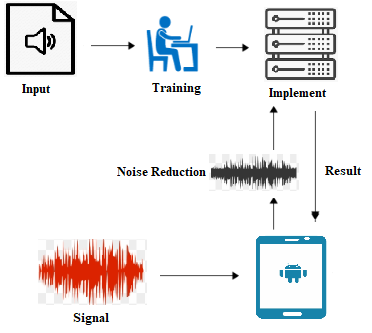
Tương tự như lớp LSTM ở dưới lớp LSTM ở trên cũng cho đầu ra là ma trận kích thước (64, 64). Quá trình lan truyền xuôi sẽ được thực hiện lần lượt theo thời gian. Tuy nhiên chỉ đầu ra bước thời gian cuối cùng (bước thời gian thứ 93 do dữ liệu của bài toán) của lớp trên được kết nối với 1 noron cùng hàm kích hoạt là hàm sigmoid để đưa ra xác suất tương ứng với 64 mẫu trong batch đó. Quá trình lan truyền xuôi theo thời gian của batch dừng ở đây.

Kết quả của quá trình lan truyền xuôi sẽ là đầu vào cho quá trình lan truyền ngược theo thời gian, trung bình lỗi của batch được tính theo công thức 2.20. Việc lan truyền ngược sẽ đi lần lượt từ bước thời gian 93 về bước thời gian 1, từ lớp LSTM trên xuống lớp LSTM ở dưới. Đạo hàm từng phần của cổng trong các khối LSTM với hàm mất mát sẽ được tính để phục vụ cho quá trình tối ưu của thuật toán adam.

Quá trình lan truyền ngược được framework keras tính sẵn dựa trên cấu hình mạng được khai báo bằng cách sử dụng phương thức fit của model. Giống với mạng CNN learning rate decay và callback LogResult được sử dụng trong phương thức fit của keras.

## Tích hợp vào thiết bị di động

Do hạn chế của đồ án nên mô hình sau khi huấn luyện thay vì được tích hợp vào ống nghe điện tử sẽ tạm thời được triển khai trên điện thoại Android. Luồng làm việc cụ thể được thể hiện trong Hình 2.23



Hình 2.23 Luồng làm việc khi tích hợp vào thiết bị di động

Trong thực tế rất khó có thể thu được tín hiệu âm thanh ở môi trường không có nhiễu, vậy nên trước khi sử dụng mô hình để suy luận ta cần xử lý nhiễu tín hiệu. Trong đồ án sử dụng phương pháp Spectral Subtraction [[46](#SVV96)]. Cụ thể phương pháp này coi tín hiệu không nhiễu ở miền thời gian là với nhiễu là . Tín hiệu thu được là:

(2.21)

Tín hiệu sẽ được chia thành từng frame nhỏ, mỗi frame sẽ chồng lấn lên nhau để tránh mất thông tin ở biên, mỗi frame sẽ được biến đổi sang miền tần số bằng biến đổi fourier rời rạc, thông tin về pha sẽ được lưu lại để sử dụng về sau. Theo giả thuyết trước, phổ của frame thứ i sẽ tương đương với:

(2.22)

Trong đó là công suất phổ của tín hiệu thu được, tín hiệu không nhiễu và tín hiệu nhiễu. Nếu tín hiệu nhiễu không thay đổi hoặc thay đổi chậm, chúng ta có thể sấp xỉ được công suất phổ của tín hiệu nhiễu bằng cách đánh giá trên các đoạn hoàn toàn chỉ có nhiễu.

Như vậy ta sẽ thu được phổ của tín hiệu không nhiễu bằng công thức:

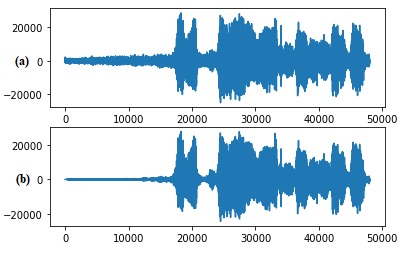
(2.23)

Trong đó M là các frame chỉ chứa tín hiệu nhiễu. Nếu kết quả của 2.22 nhỏ hơn 0 thì sẽ được gán bằng 0 [[47](#Spe1)]. Tuy nhiên theo [[48](#MBe79)] việc gán giá trị 0 khi hiệu của phổ nhỏ hơn 0 sẽ tạo ra nhiễu mới cho tín hiệu, vậy nên một biến thể khác với hiệu quả tốt hơn sẽ được điều khiển bởi hai tham số α ≥ 1, 0 ≤ β << 1 cụ thể theo công thức 2.24 và 2.25:

(2.24)

(2.25)

Sau khi trừ đi phổ công suất trung bình của nhiễu, tín hiệu sẽ được biến được biến đổi ngược lại miền tần số bằng biến đổi ngược fourier rời rạc và thông tin về pha của phổ được lưu ở trước đó. Kết quả việc lọc nhiễu được thể hiện trong Hình 2.24:



Hình 2.24 (a) Tín hiệu gốc (b) Tín hiệu thông qua lọc nhiễu

Mô hình phù hợp nhất sau khi huấn luyện (đảm bảo về mặt độ chính xác và tốc độ) sẽ được triển khai trên framework Flask [[49](#Fla)]. Âm thanh được đánh giá qua các khoảng 3 giây. Rồi được tiến hành lọc nhiễu. Dựa trên các tham số đã được học trong quá trình huấn luyện và các đặc trưng của đoạn âm thanh để đưa ra kết quả và hiển thị trên điện thoại android.

## Tổng kết chương 2

Trong chương 2 đồ án đã trình bày về cách tiền xử lý dữ liệu âm thanh thô để chuẩn hóa về chung một định dạng nhằm xử lý dễ dàng hơn cho các bước sau. Ngoài ra đồ án đã trình bày chi tiết về tính chất, cách trích chọn các đặc trưng thường được sử dụng trong các bài toán xử lý âm thanh. Từ các đặc trưng được trích chọn ở bước trước, đồ án trình bày cấu hình các mạng học sâu để học trên các đặc trưng đó. Cuối chương 2 đồ án trình bày luồng làm việc để tích hợp mô hình huấn luyện vào thiết bị di động.

Áp dụng nền tảng lý thuyết, các đặc trưng và mô hình học sâu được trình bày trong chương 2, chúng ta cần tiến hành thực nghiệm trên các đặc trưng và mô hình, để có thể đưa ra kết luận nhằm cải thiện, nâng cao hiệu năng của các mô hình. Chi tiết quá trình thực nghiệm được trình bày trong chương tiếp theo.

# THỰC NGHIỆM VÀ ĐÁNH GIÁ

Trong chương này, đồ án sẽ trình bày về quá trình tiến hành để thu được kết quả thực nghiệm trên bộ dữ liệu thu thập. Cụ thể bao gồm các phần sau:

- Thu thập và gán nhãn dữ liệu

- Phương pháp và các độ đo đánh giá

- Kết quả thực nghiệm và đánh giá

## Thu thập và gán nhãn dữ liệu

Đồ án đề xuất xây dựng mô hình phát hiện và phân loại tiếng thở khò khè và các âm thanh khác thu được trong quá trình nghe bệnh bằng ống nghe của bác sĩ. Tuy nhiên đồ án đang gặp hạn chế về mặt dữ liệu vì âm thanh phổi (được đo bằng ống nghe điện tử) là một loại thông tin y tế thường rất ít được công bố trên internet và khó thu thập trực tiếp vì yêu cầu phải thu thông tin trực tiếp từ người bệnh, ngoài ra còn yêu cầu các thiết bị y tế và bác sĩ để tiến hành đo thông tin. Vậy nên trong phạm vi nghiên cứu đồ án sẽ cố gắng thu thập thông tin về âm phổi có sẵn trên internet và thực nghiệm các mô hình học sâu trên bộ dữ liệu thu thập được.

### Thu thập dữ liệu

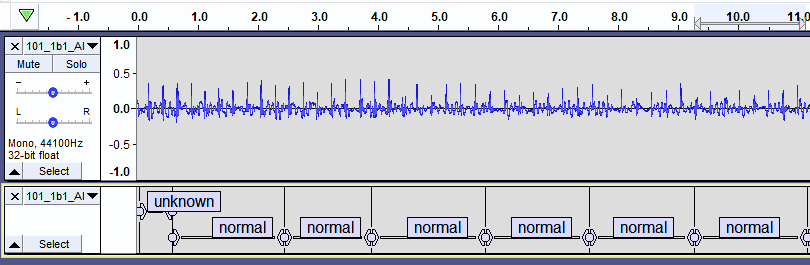
Trước khi tiến hành thu thập dữ liệu, đồ án đề xuất phân loại âm phổi thành 2 loại như sau:

* **Thở khò khè:** Là một loại thở bất thường. Thở khò khè là âm thanh nghe như tiếng huýt sáo liên tục, thô ráp và có âm vang cao tạo ra trong đường hô hấp, nghe được rõ nhất trong quá trình thở ra. Thở khò khè là dấu hiệu rõ ràng về các vấn đề hô hấp, chủ yếu xảy ra do hẹp hoặc tắc nghẹn đường hô hấp. Nguyên nhân phổ biến nhất là do hen và viêm phế quản.
* **Thở bình thường:** Là nhịp thở dài, cường độ nhẹ, xảy ra trong lúc đối tượng không thực hiện việc gì như nằm, ngồi và được ghi âm bằng ống nghe điện tử.

Dữ liệu được thu thập ở nhiều nguồn trên mạng như [[10](#Ral)], [[50](#Thids)], [[51](#Resse)]. Trong đó lớn nhất là [50], đây là một cơ sở dữ liệu được tạo bởi hai nhóm nghiên cứu ở Portugal và Greece, trong [50] chứa 920 các bản ghi âm được gán nhãn với độ dài từ 10s – 90s thu được từ 126 bệnh nhân, các vị trí thu là khí quản, ngực trước trái, ngực trước phải, sau ngực trái, sau ngực phải, lateral left, lateral right. Tổng cộng là 5.5 giờ với 6898 chu kì thở được thu trong cả môi trường yên tĩnh và có ồn. Ngoài ra dữ liệu còn được thu thập bằng cắt từ các video trên youtube.Tín hiệu sẽ được lọc qua bộ lọc butterworth highpass filter với tần số cut off là 100Hz để giảm bớt sự ảnh hưởng của tiếng tim trong tín hiệu.

### Gán nhãn dữ liệu

Dữ liệu được gán nhãn bằng công cụ Audacity [[52](#rgAu)]. Đây là phần mềm có giao diện, dễ sử dụng. Sử dụng audacity ta có thể gán nhãn cho từng đoạn của các file âm thanh rồi xuất ra tập tin văn bản để có thể tiện cho quá trình xử lý sau này.



Hình 3.1 Đánh nhãn dữ liệu bằng công cụ Audacity

Với một số tập tin trên mạng đã được gán nhãn sẵn ta sẽ sử dụng luôn nhãn này để tiếp tục xử lý các bước về sau.

## Xử lý dữ liệu

Dữ liệu thu thập được là các đoạn âm thanh có độ dài khác nhau. Như đã đề cập ở phần trước. Các mẫu âm thanh được đưa vào xử lý là các đoạn âm thanh có độ dài 3 giây. Vì thế tất cả các file âm thanh sẽ được cắt thành các đoạn 3 giây, dựa trên thông tin về nhãn đã được tạo ra ở bước trước, các mẫu liên tiếp sẽ chồng lấn lên nhau 1 giây để tránh việc mất và gián đoạn thông tin ở biên.

Để đảm bảo sự chính xác của nhãn dữ liệu, các mẫu sẽ được kiểm tra thủ công một lần. Những mẫu không phải âm thở khò khè hoặc bình thường, được gán nhãn sai sẽ bị loại ra khỏi tập dữ liệu.

Sau khi hoàn tất, số lượng các mẫu dữ liệu được thể hiện trong Bảng 3.1

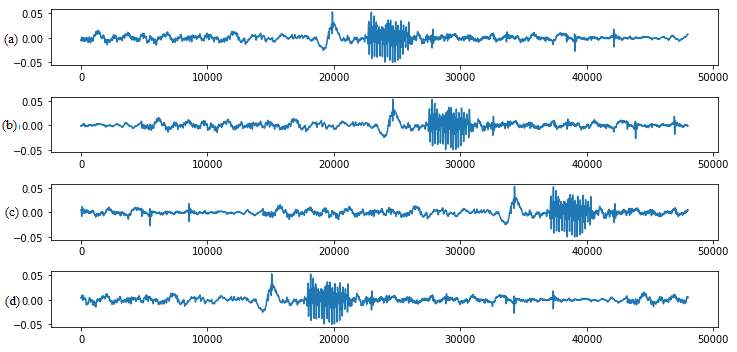
Bảng 3.1 Số lượng các loại âm phổi thu thập được

|  |  |
| --- | --- |
| **Nhãn dữ liệu** | **Số lượng** |
| Wheeze | 1250 |
| Normal | 4242 |

***Làm giàu dữ liệu***

Nhìn vào Bảng 3.1, ta thấy rằng đây là một bộ dữ liệu không cân bằng. Việc mất cân bằng hoặc thiếu hụt dữ liệu là một vấn đề phổ biến trong lĩnh vực học máy, dẫn đến mô hình huấn luyện không học được tốt với các nhãn có dữ liệu ít. Ngoài ra thì một mô hình học sâu sẽ học tốt hơn nếu có nhiều dữ liệu. Điều này đồng nghĩa với việc ta phải thu thêm dữ liệu với các nhãn có dữ liệu ít để cho các lớp dữ liệu ngang bằng nhau. Điều này là bất khả thi do gặp nhiều rào cản để được phép tiếp xúc với tín hiệu y tế.

Để khắc phục vấn đề này, đồ án đề xuất cách thức làm giàu dữ liệu bằng cách với mỗi bản ghi, chúng ta sẽ tiến hành xoay dữ liệu trên miền thời gian đi một số khoảng thời gian sao cho đảm bảo không có hai mẫu nào giống nhau được sinh ra trong quá trình làm giàu dữ liệu.Việc mở rộng một tập tin dữ liệu được trình bày ở Hình 3.2.



Hình 3.2 (a) mẫu thở khò khè bình thường (b, c, d) một số mẫu thở khò khè được mở rộng

Việc mở rộng dữ liệu theo cách này sẽ tạo ra rất nhiều biến thể của âm phổi các đoạn thở khò khè có thể xuất hiện ở bất kì vị trí nào trong mẫu, điều này cũng sẽ góp phần phù hợp với điều kiện thực tế vì khi sử dụng ống nghe điện tử, tùy thuộc vào lúc ta ghi âm vị trí của các đoạn thở khò khè có thể khác nhau. Dữ liệu mở rộng sẽ được tách riêng để thực hiện các đánh giá. Cụ thể với dữ liệu âm phổi khò khè được mở rộng tăng lên 14 lần, âm phổi bình thường được mở rộng thêm 5 lần. Số lượng dữ liệu mở rộng được thể hiện trong

Bảng 3.2

Bảng 3.2 Số lượng các mẫu âm phổi mở rộng

|  |  |
| --- | --- |
| **Nhãn dữ liệu** | **Số lượng** |
| Wheeze | 17500 |
| Normal | 21210 |

Việc xây dựng mô hình học sâu, đồ án sử dụng framework học sâu keras. Quá trình huấn luyện được thực hiện trên Google Colab cung cấp dịch vụ dựa trên Jupyter Notebook và hộ trợ GPU. Cụ thẻ cấu hình máy tính dành cho việc huấn luyện là:

* GPU Tesla K80
* Bộ nhớ 300GB
* Chip Intel Core i7 3700 4 Core vật lý.
* Ram 25GB

## Phương pháp và các độ đo đánh giá

Như đã trình bày ở trên dữ liệu mở rộng và dữ liệu nguyên bản sẽ được tách rời khỏi nhau và được thực nghiệm thông qua 4 thí nghiệm. Mỗi thí nghiệm gắn với một tiêu chí đánh giá một khía cạnh riêng. Cụ thể 4 thí nghiệm như sau:

* A: Tiến hành trên dữ liệu chưa mở rộng để so sánh hiệu năng phân loại giữa các mô hình và các đặc trưng.
* B1: Chỉ sử dụng dữ liệu mở rộng để huấn luyện và kiểm chứng
* B2: Sử dụng tập dữ liệu mở rộng để huấn luyện và kiểm tra kết quả trên tập dữ liệu gốc. Thí nghiệm B1 và B2 nhằm đánh giá sự mạnh mẽ khi âm thanh bị chuyển dịch.
* C: Sử dụng tập dữ liệu gốc để huấn luyện và đánh giá trên tập dữ liệu mở rộng. Nhằm đánh giá khả năng tổng quát hóa dữ liệu đã bị dịch chuyển khi được huấn luyện trên tập dữ liệu gốc.

Ngoài thí nghiêm B1, C sử dụng tập huấn luyện và tập kiểm tra riêng biệt. Với các thí nghiệm sẽ được đánh giá thông qua kiểm chứng chéo 5 folds. Cụ thể toàn bộ tập dữ liệu có sẽ được chia thành 5 fold có số lượng sấp xỉ bằng nhau. Quá trình kiểm chứng sẽ diễn ra 5 lần, với mỗi một lần 4 fold sẽ được sử dụng để huấn luyện, 1 fold sử dụng để kiểm chứng. Sau cho bất kì fold nào cũng được kiểm chứng đúng 1 lần và được dùng để huấn luyện đúng 4 lần. Kết quả sẽ là trung bình của 5 lần kiểm thử.

Thông thường thước đo để đánh giá độ tốt của việc phát hiện và phân loại thường là tham số Accuracy (Độ chính xác).Ngoài ra chúng ta sử dụng thêm Confusion Matrix (Ma trận nhầm lẫn), MCC để đánh giá độ tốt của mô hình.

Ma trận nhầm lẫn (Confusion Matrix) chỉ ra cụ thể mỗi loại được phân loại như thế nào, lớp nào được phân loại đúng nhiều nhất và dữ liệu thuộc lớp nào thường bị phân loại nhầm vào lớp khác. Về cơ bản, confusion matrix là một hình vuông mà mỗi nhãn tương ứng với một hàng và một cột. Ví dụ một ô bất kỳ thuộc hàngA, cột B thì có ý nghĩa là số mẫu thuộc nhãn A nhưng được dự đoán thành nhãn B.

MCC: do bộ dữ liệu được đánh giá là bộ dữ liệu không cân bằng nên MCC cũng được sử dụng để đánh giá độ tốt của mô hình. MCC nhận giá trị từ -1 đến +1 với -1 là dự đoán ngược hoàn toàn với nhãn và +1 là dự đoán hoàn toàn đúng. MCC được tính dựa trên kết quả của Confusion Matrix với công thức sau:

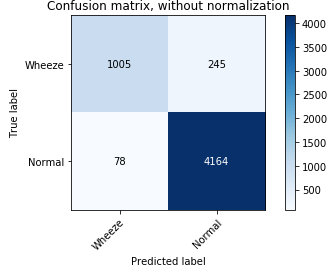
(3.1)

Accuracy là độ chính xác dự đoán của mẫu, được tính toán bằng cách lấy số mẫu dự liệu mà mô hình dự đoán đúng chia cho tổng số mẫu phải dự đoán. Ý nghĩa của tham số accuracy cho chúng ta biết tỉ lệ dự đoán chính xác của mô hình là bao nhiêu.

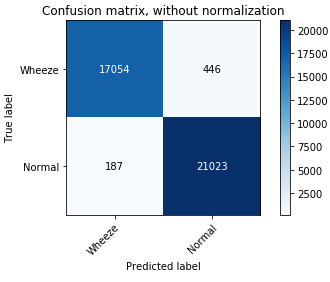
## Kết quả và đánh giá

### Kết quả với mạng CNN

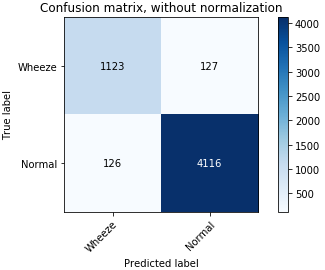
Ma trận nhầm lẫn của mạng CNN thông qua 4 thí nghiệm A, B1, B2, C lần lượt được thể hiện trong Hình 3.3, Hình 3.4, Hình 3.5, Hình 3.6.



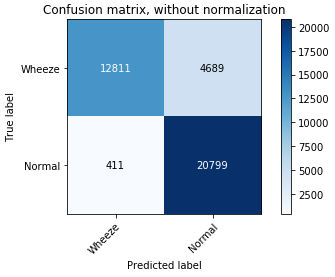
Hình 3.3 Confusion Matrix thí nghiệm A với đặc trưng Spectrogram sử dụng mạng CNN



Hình 3.4 Confusion Matrix thí nghiệm B1 với đặc trưng Spectrogram sử dụng mạng CNN



Hình 3.5 Confusion Matrix thí nghiệm B2 với đặc trưng Spectrogram sử dụng mạng CNN



Hình 3.6 Confusion Matrix thí nghiệm C với đặc trưng Spectrogram sử dụng mạng CNN

Giá trị tham số Accuracy và MCC của đặc trưng Spectrogram huấn luyện trên mạng CNN được thể hiện trong Bảng 3.3.

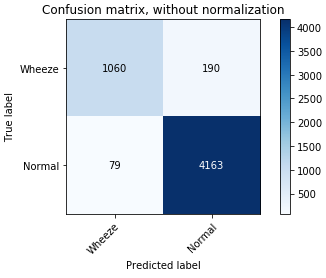
Bảng 3.3 Giá trị Accuracy MCC của Spectrogram huấn luyện trên mạng CNN

|  |  |  |
| --- | --- | --- |
| **Thí nghiệm** | **Accuracy** | **MCC** |
| **A** | 94.12% | 0.82 |
| **B1** | **98.36%** | **0.96** |
| **B2** | 95.39% | 0.86 |
| **C** | 86.83% | 0.74 |

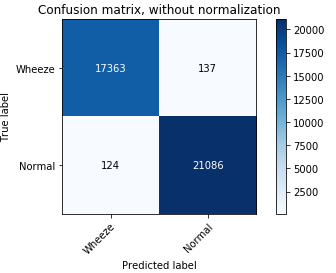
### Kết quả với mạng LSTM

***Kết quả với đặc trưng MFCC:***

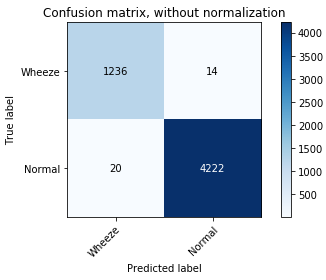
Ma trận nhầm lẫn của đặc trưng MFCC được huấn luyện và kiểm chứng trên mạng LSTM tương ứng với thí nghiệm A, B1, B2, C được thể hiện lần lượt trong Hình 3.7, Hình 3.8, Hình 3.9, Hình 3.10.

****

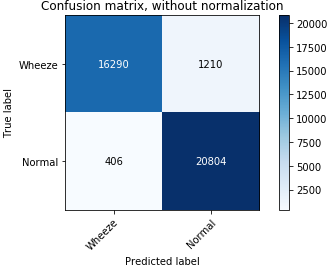
Hình 3.7 Confusion Matrix thí nghiệm A với đặc trưng MFCC sử dụng mạng LSTM

****

Hình 3.8 Confusion Matrix thí nghiệm B1 với đặc trưng MFCC sử dụng mạng LSTM

****

Hình 3.9 Confusion Matrix thí nghiệm B2 với đặc trưng MFCC sử dụng mạng LSTM

****

Hình 3.10 Confusion Matrix thí nghiệm C với đặc trưng MFCC sử dụng mạng LSTM

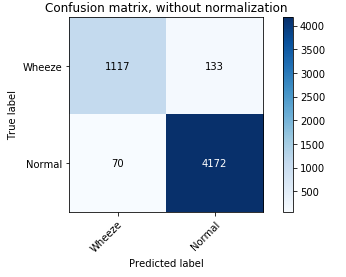
Giá trị tham số Accuracy và MCC của đặc trưng MFCC huấn luyện trên mạng LSTM được thể hiện trong Bảng 3.4

Bảng 3.4 Giá trị Accuracy và MCC của đặc trưng MFCC huấn luyện trên mạng LSTM

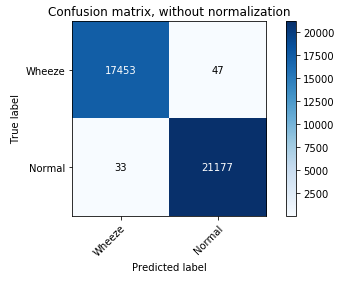
|  |  |  |
| --- | --- | --- |
| **Thí nghiệm** | **Accuracy** | **MCC** |
| **A** | 95.10% | 0.85 |
| **B1** | 99.33% | 0.98 |
| **B2** | **99.38%** | **0.98** |
| **C** | 95.83% | 0.91 |

***Kết quả với đặc trưng GFCC:***

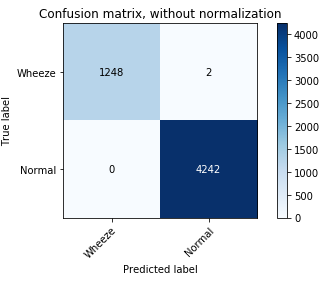
Ma trận nhầm lẫn của đặc trưng GFCC huấn luyện và đánh giá trên mạng LSTM tương ứng với thí nghiệm A, B1, B2, C lần lượt được thể hiện trong Hình 3.11, Hình 3.12, Hình 3.13, Hình 3.14.

****

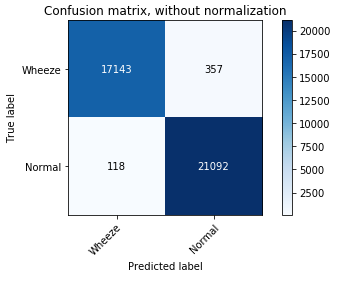
Hình 3.11 Confusion Matrix thí nghiệm A với đặc trưng GFCC sử dụng mạng LSTM

****

Hình 3.12 Confusion Matrix thí nghiệm B1 với đặc trưng GFCC sử dụng mạng LSTM

****

Hình 3.13 Confusion Matrix thí nghiệm B2 với đặc trưng GFCC sử dụng mạng LSTM

****

Hình 3.14 Confusion Matrix thí nghiệm C với đặc trưng GFCC sử dụng mạng LSTM

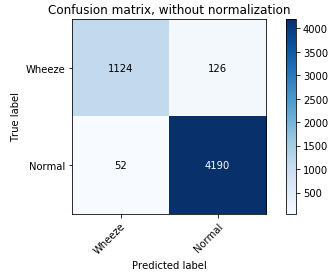
Giá trị Accuracy và MCC với đặc trưng GFCC được huấn luyện trên mạng LSTM được thể hiện trong Bảng 3.5

Bảng 3.5 Giá trị Accuracy và MCC với đặc trưng GFCC được huấn luyện trên mạng LSTM

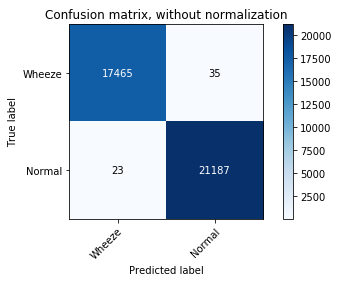
|  |  |  |
| --- | --- | --- |
| **Thí nghiệm** | **Accuracy** | **MCC** |
| **A** | 96.30% | 0.89 |
| **B1** | 99.79% | 0.99 |
| **B2** | **99.96%** | **0.99** |
| **C** | 98.77% | 0.97 |

***Kết quả với kết hợp đặc trưng MFCC và GFCC:***

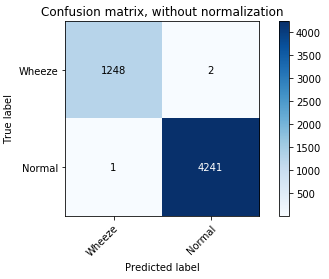
Ma trận nhầm lẫn của kết hợp đặc trưng MFCC và GFCC được huấn luyện và kiểm chứng trên mạng LSTM tương ứng với thí nghiệm A, B1, B2, C được thể hiện lần lượt trong Hình 3.15, Hình 3.16, Hình 3.17, Hình 3.18.

****

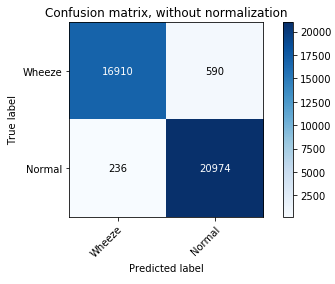
Hình 3.15 Confusion Matrix thí nghiệm A với kết hợp đặc trưng MFCC GFCC sử dụng mạng LSTM

****

Hình 3.16 Confusion Matrix thí nghiệm B1 với kết hợp đặc trưng MFCC GFCC sử dụng mạng LSTM

****

Hình 3.17 Confusion Matrix thí nghiệm B2 với kết hợp đặc trưng MFCC GFCC sử dụng mạng LSTM

****

Hình 3.18 Confusion Matrix thí nghiệm C với kết hợp đặc trưng MFCC GFCC sử dụng mạng LSTM

Giá trị Accuracy và MCC với kết hợp đặc trưng MFCC GFCC được huấn luyện trên mạng LSTM được thể hiện trong Bảng 3.6

Bảng 3.6 Giá trị Accuracy và MCC với kết hợp đặc trưng MFCC GFCC được huấn luyện trên mạng LSTM

|  |  |  |
| --- | --- | --- |
| **Thí nghiệm** | **Accuracy** | **MCC** |
| **A** | 97.76% | 0.90 |
| **B1** | 99.85% | 0.99 |
| **B2** | **99.95%** | **0.99** |
| **C** | 97.87% | 0.95 |

### Đánh giá kết quả thực nghiệm

Kết quả thực nghiệm cho thấy khi với tất cả các mạng và đặc trưng, nhóm thí nghiệm B (B1, B2) luôn đạt kết quả cao nhất. Điều này chứng tỏ khi lượng dữ liệu tăng lên các mạng noron trong đề tài có khả năng tổng quát hóa dữ liệu tốt hơn.

Kết quả của trích chọn đặc trưng tự động sử dụng mạng CNN nhìn chung cho kết quả kém hơn so với việc trích chọn các đặc trưng thủ công. Khi lượng dữ liệu tăng lên mạng CNN đạt kết quả tốt nhất ở thí nghiệm B2 với độ chính xác là 98.36% và MCC là 0.96 như vậy để mạng CNN có tiềm năng trong trích chọn đặc trưng tự động. Tuy nhiên để đạt được kết quả mong muốn cần lượng dữ liệu đủ lớn để tiến hành huấn luyện.

Với trích chọn đặc trưng thủ công và học trên mạng LSTM, cho kết quả rất tốt. Đặc trưng MFCC cho độ chính xác 99.38%, GFCC là 99.96%, kết hợp cả 2 đặc trưng 99.95%. Kết quả thực nghiệm cho thấy đặc trưng GFCC cho kết quả tốt hơn so với MFCC điều này được lý giải do MFCC rất nhạy cảm với nhiễu, đây cũng là nhược điểm chính của MFCC.

Khi kết hợp cả 2 đặc trưng cho kết quả tốt hơn với sử dụng riêng lẻ đặc trưng trong thí nghiệm A, chứng tỏ khi lượng dữ liệu nhỏ việc kết hợp đặc trưng cho kết quả tốt hơn, tuy nhiên khi lượng dữ liệu tăng lên thì khoảng cách này trở nên không đáng kể, nhưng nhìn chung vẫn làm tăng độ hiệu năng của mô hình.

Trong thí nghiệm C đặc trưng GFCC cho kết quả tốt hơn so với kết hợp 2 đặc trưng. Như vậy MFCC có khả năng tổng quát trên dữ liệu bị chuyển dịch kém hơn so với GFCC, làm kết quả bị kém hơn một chút so với sử dụng riêng lẻ đặc trưng GFCC.

Với số lượng tham số nhỏ hơn 0.06M tham số so với 60M tham số của mạng CNN. Và còn đạt kết quả tốt hơn, như vậy mô hình sử dụng mạng LSTM với đặc trưng GFCC được sử dụng để tích hợp vào ứng dụng di động nhằm phát hiện tiếng thở khò khè.

## Tổng kết chương 3

Trong chương 3, đồ án đã trình bày quá trình thực nghiệm trong đồ án, lần lượt từ việc thu thập dữ liệu, gán nhãn và xử lý dữ liệu.Sau đó sử dụng các phương pháp trích chọn đặc trưng và mô hình được đề xuất để đưa ra kết quả thực nghiêm. Dựa trên các thước đo đánh giá thực nghiệm (độ chính xác, hệ số tương quan Matthews, ma trận nhầm lẫn) kết hợp với tính chất của các đặc trưng và các mô hình học sâu để đưa ra nhận xét, đánh giá về kết quả của thực nghiệm.

# TỔNG KẾT

Trong quá trình tìm hiểu và phân tích những nghiên cứu về các hệ thống, thuật toán tự động phát hiện tiếng khò khè. Đồ án đã đạt được một số kết quả sau:

* Giới thiệu khái quát về các nhịp thở, các đặc trưng riêng biệt của các âm phổi phổ biến về cường độ, các đặc trưng trong miền tần số.
* Trình bày một số nghiên cứu có liên quan về việc tự đông phát hiện tiếng thở khò khè cụ thể bao như phân tích âm phổi tự động, phương pháp phát hiện âm thở khò khè dựa trên Entropy, phương pháp phát hiện âm thở khò khè sử dụng mạng noron, so sánh các thuật toán học máy SVM và KNN để chẩn đoán các bệnh lý hô hấp bằng cách sử dụng âm thanh phổi.
* Trình bày khái niệm, tính chất đặc điểm của trí tuệ nhân tạo, học máy, học sâu. Mô tả chi tiết về cấu trúc của mạng học sâu phổ biến như mạng noron tích chập, mạng noron hồi quy, mạng noron bộ nhớ dài ngắn.
* Trình bày quá trình tiền xử lý dữ liệu âm thanh, biến đổi từ dữ liệu thô thành các mẫu dữ liệu có thời lượng như nhau để xử lý. Trình bày cách trích chọn đặc trưng phù hợp cho bài toán. Đề xuất các kiến trúc mạng sử dụng các đặc trưng đã được trích chọn để áp dụng vào quá trình thực nghiệm. Cụ thể đồ án đề xuất 2 loại mạng phổ biến hiện nay là mạng noron tích chập và mạng noron bộ nhớ dài ngắn cùng cấu hình phù hợp với bài toán.
* Thu thập dữ liệu và gán nhãn sau đó tiến hành xử lý dữ liệu, thực nghiệm với các mô hình học sâu và các đặc trưng được đề xuất. Trình bày phương pháp cùng các thang đo để đánh giá. Đưa ra nhận xét về kết quả của quá trình thực nghiệm. Các kết quả thực nghiệm cho thấy đặc trưng GFCC với huấn luyện trên mạng LSTM cho kết quả tốt nhất. Với độ chính xác đạt 99.96%, MCC đạt giá trị 0.99. Mạng CNN với trích chọn đặc trưng tự động với điều kiện tốt nhất đạt 98.36% về độ chính xác, cho thấy mạng CNN có tiềm năng lớn trong việc trích chọn đặc trưng tự động, tuy nhiên để phát huy được sức mạnh thì cần thêm nhiều dữ liệu để tiến hành huấn luyện.

***Hướng phát triển trong tương lai***

Ngoài các trạng thái thở được sử dụng trong bài toán, còn rất nhiều triệu chứng thở khác mà rất quan trọng liên quan đến các bệnh về phổi cũng như đường hô hấp, như vậy hướng phát triển đầu tiên là sẽ thu thập thêm dữ liệu về các triệu chứng thở khác.

Tiếp theo việc xử lý nhiễu khi phân loại vẫn còn là vấn đề rất lớn trong thực tế khi ghi âm có thể có rất nhiều nhiễu ảnh hưởng tới tín hiệu, các nhiều này không tuân theo một phân phối nhất định nào cả nên hướng phát triển tiếp theo chính là tìm hiểu về các kỹ thuật xử lý tín hiệu hiệu quả sao cho có thể hoạt động tốt trong thực tế.

Trong tương lai sẽ cố gắng tích hợp mô hình vào ống nghe điện tử để có thể ứng dụng trong thực tế.

TÀI LIỆU THAM KHẢO

[1] Hít thở, https://vi.wikipedia.org/wiki/H%C3%ADt\_th%E1%BB%9F.

[2] Pasterkamp, Hans, Steve S. Kraman, and George R. Wodicka, "Respiratory sounds: advances beyond the stethoscope.," in American journal of respiratory and critical care medicine, 1997, pp. 974-987.

[3] A. R. A., et al Sovijarvi, "Definition of terms for applications of respiratory sounds," in European Respiratory Review, 2000, pp. 597-610.

[4] Robert P., and Robert G. Loudon Baughman, "Stridor: Differentiation from Asthma or Upper Airway Noise1-3," in Asthma or Upper Airway Noise1-3, 1989, pp. 1407-1409.

[5] Jeannette, and Robert G. Loudon Hoevers, "Measuring Cracklesa.," in Chest, 1990, pp. 1240-1243.

[6] Morten Grønnesby, "Automated Lung Sound Analysis," UiT The Arctic University of Norway, Master thesis 2016.

[7] The Tromsø Study, https://en.uit.no/om/enhet/artikkel?p\_document\_id=80172&p\_dimension\_id=88111.

[8] Zhang, Jianmin & Ser, Wee & yu, Jufeng & Zhang, T.T., "A Novel Wheeze Detection Method for Wearable Monitoring Systems," in International Symposium on Intelligent Ubiquitous Computing and Education, 2009, pp. 331-334.

[9] Palaniappan, Rajkumar & Sundaraj, Kenneth & Sundaraj, Sebastian., "A comparative study of the SVM and K-nn machine learning algorithms for the diagnosis of respiratory pathologies using pulmonary acoustic signals," in BMC bioinformatics., 2014.

[10] Rale Repository, http://rale.ca/.

[11] Hashemi, Amjad & Arabalibeik, Hossein & Agin, Khosrow., "Classification of Wheeze Sounds Using Wavelets and Neural Networks," in nternational Conference on Biomedical Engineering and Technology, 2011, p. 11.

[12] Copeland, Michael, "What’s the difference between artificial intelligence, machine learning, and deep learning," Nvidia, https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learningdeep-learning-ai/, 2016.

[13] Are neural networks and deep learning a part of machine learning and artificial intelligence?, https://www.quora.com/Are-neural-networks-and-deep-learning-a-part-of-machine-learning-and-artificial-intelligence.

[14] Andrew Ng, "Deep Learning," http://deeplearning.ai,.

[15] Big Data Analysis and Storage - Scientific Figure on ResearchGate, https://www.researchgate.net/figure/Examples-of-big-data-velocity\_fig3\_313400371/actions#reference.

[16] How to Compute the Derivative of a Sigmoid Function, https://kawahara.ca/how-to-compute-the-derivative-of-a-sigmoid-function-fully-worked-example/.

[17] What is the difference between deep learning and usual machine learning?, https://www.quora.com/What-is-the-difference-between-deep-learning-and-usual-machine-learning?redirected\_qid=2417286.

[18] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in neural information processing systems, 2012.

[19] Convolutional Neural Networks for Autonomous Cars, https://electronicsforu.com/market-verticals/automotive/convolutional-neural-networks-autonomous-cars-part-1-2.

[20] Derivation of Convolutional Neural Network Back Propagation Theory, http://www.programmersought.com/article/5864521999/;jsessionid=24454B8676F1ECF94689009C74282C1B

[21] Multi-layer Perceptron và Backpropagation, https://machinelearningcoban.com/2017/02/24/mlp/.

[22] Nair, Vinod, and Geoffrey E. Hinton, "Rectified linear units improve restricted boltzmann machines," in Rectified linear units improve restricted boltzmann machines, 2010, pp. (ICML-10).

[23] A QUICK GUIDE FOR NEWBIES CNNS, https://www.esantus.com/blog/2019/1/31/convolutional-neural-networks-a-quick-guide-for-newbies.

[24] Ciresan, Dan C., et al., "Flexible, high performance convolutional neural networks for image classification," in Flexible, high performance convolutional neural networks for image classification, 2011, p. Vol. 22. No. 1.

[25] Yann, et al LeCun, "Gradient-based learning applied to document recognition.," in Proceedings of the IEEE, 86.11, 1998, pp. 2278-232.

[26] Zeiler, Matthew D., and Rob Fergus., "Visualizing and understanding convolutional networks.," in Visualizing and understanding convolutional networks., Springer, Cham, 2014.

[27] Szegedy, Christian, et al., "Going deeper with convolutions.," in Proceedings of the IEEE conference on computer vision and pattern recognition., 2015.

[28] Szegedy, Christian, et al., "Going deeper with convolutions.," in Proceedings of the IEEE conference on computer vision and pattern recognition., 2015.

[29] He, Kaiming, et al., "Deep residual learning for image recognition.," in Proceedings of the IEEE conference on computer vision and pattern recognition., 2016.

[30] Hochreiter, Sepp., "Untersuchungen zu dynamischen neuronalen Netzen.," in Diploma, Technische Universitat M ¨ unchen ¨, 1991, p. 91.1.

[31] Understanding LSTM Networks., https://colah.github.io/posts/2015-08- Understanding- LSTMs

[32] Yoshua, Patrice Simard, and Paolo Frasconi. Bengio, "Learning long-term dependencies with gradient descent is difficult.," in IEEE transactions on neural networks, 1994, pp. 1735 – 1780

[33] Hochreiter, Sepp, and Jurgen Schmidhuber., "Long short-term memory.," in Neural computation, 1997, pp. 1735 - 1780.

[34] Martin, Alexis & Voix, Jeremie, "In-Ear Audio Wearable: Measurement of Heart and Breathing Rates for Health and Safety Monitoring.," in IEEE Transactions on Biomedical Engineering, 2017, pp. 1-1. 10.1109/TBME.2017.2720463.

[35] Spectrogram, <https://en.wikipedia.org/wiki/Spectrogram>.

[36] Short-time Fourier transform, https://en.wikipedia.org/wiki/Short-time\_Fourier\_transform.

[37] MFCC Tutorial, http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/.

[38] Nyquist frequency, https://en.wikipedia.org/wiki/Nyquist\_frequency.

[39] Matplotlib, https://matplotlib.org/.

[40] Gammatone filter, https://en.wikipedia.org/wiki/Gammatone\_filter.

[41] H. Xu, L. Lin, X. Sun and H. Jin, "A New Algorithm for Auditory Feature Extraction ," in 2012 International Conference on Communication Systems and Network Technologies, Rajkot, 2012, pp. 229-232.

[42] Equivalent rectangular bandwidth, https://en.wikipedia.org/wiki/Equivalent\_rectangular\_bandwidth.

[43] Logistic Regression, http://www.jasonhhao.com/machine-learning-blogs/logistic-regression/.

[44] Keras, https://en.wikipedia.org/wiki/Keras.

[45] Numpy, https://en.wikipedia.org/wiki/NumPy.

[46] Vaseghi S.V., "Spectral Subtraction," in Advanced Signal Processing and Digital Noise Reduction., 1996.

[47] Speech Enhancement tutorial: Spectral Subraction, "http://practicalcryptography.com/miscellaneous/machine-learning/tutorial-spectral-subraction/,".

[48] M. Berouti, R. Schwartz and J. Makhoul, "Enhancement of speech corrupted by acoustic noise," in IEEE International Conference on Acoustics, Speech, and Signal Processing, Washington DC, 1979, pp. 208-211.

[49] Flask Web FrameWork, https://en.wikipedia.org/wiki/Flask\_(web\_framework).

[50] ThinkLabs Lung sound, https://www.thinklabs.com/copy-of-lung-sounds.

[51] Respiratory Sound Database, https://www.kaggle.com/vbookshelf/respiratory-sound-database.

[52] Audacity, https://www.audacityteam.org/.