TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN CUỐI KÌ**

**NHẬP MÔN TRÍ TUỆ NHÂN TẠO**

**Project 25**

*Người thực hiện*: **TRẦN VŨ CÔNG KHANH – 51900804**

**TRẦN ĐỨC VĂN – 51900846**

Lớp**: 19050401**

**19050402**

Khoá**: 23**

*Người hướng dẫn*: **TS BÙI THANH HÙNG**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2022**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN CUỐI KÌ**

**NHẬP MÔN TRÍ TUỆ NHÂN TẠO**

**Project 25**

*Người thực hiện*: **TRẦN VŨ CÔNG KHANH – 51900804**

**TRẦN ĐỨC VĂN – 51900846**

Lớp**: 19050401**

**19050402**

Khoá**: 23**

*Người hướng dẫn*: **TS BÙI THANH HÙNG**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2022**

LỜI CẢM ƠN

Trước khi đi đến nội dung báo cáo. Với sự biết ơn sâu sắc và tình cảm chân thành nhất, chúng em xin được gửi lời cảm ơn đến tất cả các cá nhân, tổ chức của trường đại học Tôn Đức Thắng đã tạo điều kiện để chúng em có thể học tập và hoàn thành bài tiểu luận này.

Đặc biệt chúng em xin gửi lời cảm ơn chân thành đến thầy Bùi Thanh Hùng – người đã hỗ trợ chúng em học tập và nghiên cứu môn học Nhập môn trí tuệ nhân tạo trong học kỳ I năm học 2022-2023,

Nhờ có những buổi học bổ ích cùng với những hướng dẫn, dạy bảo của thầy mà ngày hôm nay chúng em mới có thể hoàn thành bài báo cáo này. Một lần nữa, chúng em xin chân thành cảm ơn thầy.

**ĐỒ ÁN ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là sản phẩm đồ án của riêng chúng tôi và được sự hướng dẫn của TS. Bùi Thanh Hùng. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình.** Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày 3 tháng 12 năm 2022*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Trần Vũ Công Khanh*

*Trần Đức Văn*

**PHẦN ĐÁNH GIÁ CỦA GIẢNG VIÊN**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

TÓM TẮT

**Bài 1: COMPUTER CHINESE CHESS GAME (Ứng dụng AI vào trò chơi cờ tướng)**

Đầu tiên nhóm tiến hành nghiên cứu, tham khảo và tiếp cận với bộ môn cờ tướng. Vì mặt hạn chế và độ phổ biến của bộ môn này ở tầm trung nên dẫn đến việc nghiên cứu gặp khá nhiều khó khăn.

Nhóm sau một thời gian tìm kiếm đã tìm thấy chương trình của ThanhLongDo và Huynhdo đến từ đại học Bách Khoa. Kết hợp với kiến thức và chỉ dẫn của các thuật toán trên mạng, nhóm đã áp dụng vào chương trình cờ tướng mà nhóm đã tham khảo.

Sau khi phân tích kỹ và xem xét chương trình nhóm đã tiếp thu thêm kiến thức mới về bộ môn cờ tướng, kèm đó là các thuật toán Minimax, Alpha-Beta, Negamax, … một cách sâu sắc hơn. Vì thời gian làm đồ án có hạn nên nhóm vẫn chưa thể thiết kế lập trình một chương trình mới, vì thế nhóm đã sử dụng bài giao diện này để làm mẫu cho chương trình.

**Bài 2: Tìm hiểu cuộc thi Rainforest Connection Species Detection (BirdCLEF: LIBROSA Audio Feature Extraction)**

Cuộc thi Rainforest Connection Species Detection là một bài toán đưa ra vấn đề: nhận biết âm thanh của các loài động vật trong rừng rậm nhiệt đới. Nhóm sau khi tìm hiểu và tham khảo các bài code trên Kaggle thì chưa thể kiếm được một bài giải có hướng đi đúng như mong muốn.

Vì thế nhóm đã tìm kiếm thêm thông qua cuộc thi khác nhưng cách đặt vấn đề của bài toán vẫn tương tự như để trước: BirdCLEF:LIBROSA Audio Feature Extraction - Phân tích đặc trưng giọng nói của các loài chim. Sau khi tìm hiểu thì nhóm cảm thấy vấn đề này có thể áp dụng vào bài toán trước.

Nhóm đã tìm ra một bài dùng để tham khảo và tiến hành phân tích, tìm hiểu. Vì mục tiêu bài giải là tìm giọng đặc trưng của các loài chim, nên nhóm có ý tưởng dùng chúng để áp dụng cho việc trích xuất đặc trưng của các loài trong rừng rậm nhiệt đới, từ đó có thể đưa vào mô hình thực tế để áp dụng. Trong đồ án này là bao gồm những lý thuyết mà nhóm tìm hiểu đi kèm đó là giới thiệu sơ lược về cách mà bài giải trên Kaggle hướng đến.

MỤC LỤC

[LỜI CẢM ƠN 1](#_Toc22238)

[TÓM TẮT 4](#_Toc29746)

[MỤC LỤC 6](#_Toc30745)

[DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT 8](#_Toc688)

[DANH MỤC CÁC HÌNH VẼ 9](#_Toc25146)

[DANH MỤC CÁC BẢNG 10](#_Toc16155)

[CHƯƠNG 1 11](#_Toc2469)

[1.1 Mô tả cấu trúc dữ liệu 11](#_Toc19315)

[1.1.1 Thuật toán Minmax (Minimax) 11](#_Toc4856)

[1.1.2 Alpha-Beta Pruning 12](#_Toc520)

[1.1.3 Negamax 13](#_Toc24237)

[1.1.4 Sequential search 14](#_Toc6965)

[1.1.5 Quiescence Search 14](#_Toc15977)

[1.1.6 Depth First Search 15](#_Toc6613)

[1.2 Sơ đồ giải thuật 15](#_Toc2601)

[1.2.1 Sơ đồ giải thuật tổng quát 15](#_Toc31835)

[1.2.2 Sơ đồ giải thuật alpha-beta minimax 16](#_Toc15210)

[1.2.3 Sơ đồ giải thuật negamax 16](#_Toc24506)

[1.3 Hiện thực 17](#_Toc9984)

[1.4 Kết quả và thảo luận 24](#_Toc27264)

[CHƯƠNG 2 25](#_Toc29241)

[2.1 Giới thiệu về bài toán 25](#_Toc18896)

[2.2 Phân tích yêu cầu của bài toán 26](#_Toc1659)

[2.2.1 Yêu cầu của bài toán 26](#_Toc5915)

[2.2.2. Các phương pháp giải quyết bài toán 26](#_Toc7815)

[2.2.2.1. Phương pháp Spectrograms 26](#_Toc29904)

[2.2.2.2 Phương pháp Mel-Frequency Cepstral Coefficients (MFCCs) 27](#_Toc8953)

[2.2.3 Phương pháp đề xuất giải quyết bài toán 27](#_Toc24923)

[2.3 Phương pháp giải quyết bài toán 28](#_Toc30670)

[2.3.1 Mô hình tổng quát 28](#_Toc31518)

[2.3.1.1 Mô hình tổng quát phương pháp Spectrograms 28](#_Toc10981)

[2.3.1.2 Mô hình tổng quát phương pháp Mel-Frequency Cepstral Coefficients 29](#_Toc1366)

[2.3.2 Đặc trưng của mô hình đề xuất 30](#_Toc13284)

[2.3.2.1 Đặc trưng mô hình Spectrograms 30](#_Toc15623)

[2.3.2.2 Đặc trưng mô hình Mel-Frequency Cepstral Coefficients (MFCC) 34](#_Toc14310)

[2.4 Thực nghiệm 38](#_Toc18019)

[2.4.1 Dữ liệu 38](#_Toc6097)

[2.4.2 Xử lý dữ liệu 39](#_Toc32327)

[2.5 Kết quả đạt được 40](#_Toc1621)

[2.6 Kết luận 44](#_Toc6340)

DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT

**CÁC KÝ HIỆU**

*dB Đơn vị cường độ âm thanh*

**CÁC CHỮ VIẾT TẮT**

DFS Depth-First Search

STFT Short-Time Fourier Transform

DFT Discreate Fourier Transform

FFT Fast Fourier Transform

MFCCs Mel-Frequency Cepstral Coefficients

IDFT Inverse Discrete Fourier Transform

DANH MỤC CÁC HÌNH VẼ

[Hình 1.1 Sơ đồ mô tả thuật toán Minmax 12](#_Toc11282)

[Hình 1.2 : Sơ đồ giải thuật tổng quát chương trình 15](#_Toc16945)

[Hình 1.3 : Sơ đồ giải thuật alpha-beta minimax trong chương trình 16](#_Toc15436)

[Hình 1.4 : Sơ đồ giải thuật negamax của chương trình 17](#_Toc21746)

[Hình 1.5 : Khởi tạo bàn cờ 18](#_Toc1161)

[Hình 1.6 : Hàm kiểm tra chiếu tướng 19](#_Toc23343)

[Hình 1.7 : Khởi tạo hàm thuật toán negamax 21](#_Toc663)

[Hình 1.8 : Khởi tạo negamax vào giải thuật alpha-beta 21](#_Toc28616)

[Hình 1.9 : Khởi tạo hàm negamax alpha-beta move ordering 22](#_Toc19439)

[Hình 1.10 : Khỏi tạo hàm tìm root có quy tắc lặp ba lần 23](#_Toc5281)

[Hình 1.11 : Khởi tạo hàm tìm kiếm quiescence 23](#_Toc14139)

[Hình 2.12 : Mô hình tổng quát phương pháp Spectrogram 29](#_Toc18504)

[Hình 2.13 :Mô hình tổng quát phương pháp MFCC 30](#_Toc16149)

[Hình 2.14 : Giải thuật FFT Cooley-Tukey cho mảng 8 phần tử 32](#_Toc7896)

[Hình 2.15 : Hình mô tả quá trình biến đổi FFT 33](#_Toc530)

[Hình 2.16 : Biểu đồ ví dụ đầu ra Spectrogram của thuật toán FFT 34](#_Toc25117)

[Hình 2.17 : Ví dụ về quang phổ âm thanh của đầu ra DFT 36](#_Toc1023)

[Hình 2.18 : Ví dụ về mô hình Mel filterbank 37](#_Toc31992)

[Hình 2.19 : Biểu đồ output feature cúa một loài chim của nhóm 40](#_Toc5226)

[Hình 2.20 : Thực nghiệm về phương pháp Mel Spectrogram 41](#_Toc13829)

[Hình 2.21 : Thực nghiệm về phương pháp Spectral centroids 42](#_Toc32357)

[Hình 2.22 : Thực nghiệm phương pháp Log Frequency Spectrogram 43](#_Toc32124)

[Hình 2.23 : Đầu ra của chương trình qua phương pháp MFCCs 44](#_Toc25990)

DANH MỤC CÁC BẢNG

[Bảng 1 : Bảng tự đánh giá 50](#_Toc21077)

CHƯƠNG 1

**COMPUTER CHINESE CHESS GAME**

**(Ứng dụng AI vào trò chơi cờ tướng)**

* 1. Mô tả cấu trúc dữ liệu
     1. Thuật toán Minmax (Minimax)
* Là thuật toán đệ quy hoặc là quay lui (backtracking) được sử dụng trong việc ra hướng quyết định và các định luật của game. Thuật toán sẽ cung cấp những hướng quyết định tối ưu tùy theo cách người chơi
* Thuật toán sẽ sử dụng đệ quy để tìm hướng đi thông qua cây game (game-tree)
* Hầu hết thuật toán này được áp dụng trong việc chơi cùng AI (cờ vua, cờ tướng, caro,…)
* Ta hình dung thuật toán này như sau: sẽ có 2 người chơi đối đầu với nhau (được gọi là Max và Min), chỉ có người chơi này sẽ ra quyết định những nước đi mà càng có lợi cho mình càng tốt trong khi phải khiến cho đối thủ bất lợi. 2 bên sẽ đối đầu với nhau khi mà Max sẽ lựa chọn giá trị lớn và Min chỉ được chọn giá trị nhỏ
* Thuật toán minmax sẽ biểu diễn thông qua thuật toán tìm kiếm theo chiều sâu (depth-first search) để tìm ra cách hoàn thành được game tree. Thuật toán sẽ tìm những đường đi đến các node cuối (Terminal node) của cây và sau đó dùng thuật toán backtracking về node đầu của cây. Đây cũng là một thể hiện của thuật toán đệ quy



Hình 1.1 Sơ đồ mô tả thuật toán Minmax

1.1.2 Alpha-Beta Pruning

* Thuật toán Alpha-beta pruning là một kỹ thuật tối ưu của thuật toán minmax.
* Như em đã đề cập về thuật toán minmax ở trên, thuật toán sẽ tìm hướng đi theo cấp số nhân theo chiều sâu của game tree. Vì vậy nên ta sẽ cắt một nửa. Tuy kỹ thuật này sẽ bỏ việc kiểm tra từng node nhưng vẫn chạy được thuật toán minmax một cách chính xác, đó được gọi là kỹ thuật cắt tỉa (prunning). Việc cắt tỉa sẽ liên quan đến hai tham số là Alpha và Beta để ta có thể mở rộng cho hướng đi trong tương lai
* Thuật toán này được áp dụng với bất kỳ chiều sâu nào của cây, không chỉ cắt tỉa lá cây mà còn cắt tỉa toàn bộ cây phụ
* Hai tham số của thuật toán alpha-beta được định nghĩa như sau:

1. Alpha: hướng đi tốt nhất (giá trị cao nhất) tính từ đầu đến khi quyết định trên hướng đi của Maximizer. Giá trị ban đầu của tham số này là -inf
2. Beta: hướng đi tốt nhất (giá trị thập nhất) tính từ đầu đến khi quyết định trên hướng đi của Minimizer. Giá trị ban đầu của tham số này là inf

* Thuật toán alpha-beta pruning sẽ cho ra kết quả giống như thuật toán minmax nhưng lại loại bỏ các node không ảnh hưởng đến quyết định cuối cùng. Ưu điểm là sẽ cho ra cùng một quyết định nhưng tốc độ quyết định của alpha-beta sẽ nhanh hơn của minmax

1.1.3 Negamax

* Negamax là một dang biến thể của minimax dựa trên tính chất zero-sum của 2 người chơi. Đây là thuật toán tìm kiếm được sử dụng khá phổ biến trong lập trình các game (cờ vua, cờ tướng…)
* Thuật toán dựa trên công thức max(a,b) = -min(-a,-b) để giảm độ phức tạp của thuật toán minimax
* Nói một cách chính xác, nếu hướng đi a của người chơi A sẽ gây bất lợi cho người chơi B. người chơi sẽ tìm kiếm một nước đi để tối đa hóa sự phủ định (negation) của đối phương. Điều này có nghĩa là một hướng đi của người chơi sẽ đánh giá cả 2 vị trí. Đây là sự đơn giản hóa so với minimax, khi mà A chọn nước đi với giá trị maximum trong khi B chọn nước đi với giá trị minimum
* Negamax hoạt động trên cùng game tree có sử dụng thuật toán minmax. Mỗi node và root node là trạng thái của game (như trò chơi cờ) của 2 người chơi. Đi một nước cờ là đại diện của sự chuyển đổi của node con từ node đã cho
* Negamax trong alpha-beta prunning: thuật toán alpha-beta sẽ cắt tỉa làm giảm số node là thuật toán negamax đánh giá trong game tree tương tự như với thuật toán minmax. Khi negamax tìm kiếm game tree, bảng chuyển vị (transposition table) sẽ trả về các giá trị node đã tính toán để bỏ qua các node đã duyệt, từ đó giúp hiệu suất tính toán đường đi được cải thiện. Quá trình này sẽ thể hiện rõ đối với các game tree có hướng quyết định đến node chung nhất định

1.1.4 Sequential search

* Tìm kiếm tuyến tính (linear search) hay còn gọi là tìm kiếm tuần tự (sequential search) là một phương pháp tìm kiếm một phần tử trong một list. Thuật toán này sẽ tuần tự kiểm tra từng phần của list cho đến khi tìm thấy được giá trị phù hợp. Nếu không tìm được giá trị phù hợp cho đến khi cuối list, thuật toán sẽ dừng lại
* Trường hợp tệ nhất là theo thời gian tuyến tính n với n là độ dài của list. Nếu mỗi giá trị phần tử được tìm kiếm như nhau thì tìm kiếm tuyến tính có trường hợp trung bình là (n+1)/2 và thời gian tìm sẽ lâu hơn nếu các giá trị khác nhau
* Thực tế phương pháp này hiếm khi được áp dụng vào thực tế vì đã có các phương pháp thay thế tốt hơn (binary search, hash table) giảm thời gian tìm kiếm đáng kể

1.1.5 Quiescence Search

* Tìm kiếm tĩnh (Quiescence search) là thuật toán sử dụng để mở rộng tìm kiếm ở các node không ổn định trong thuật toán minmax game tree. Đây là hàm để đánh giá một cách mở rộng để trì hoãn đánh giá cho đến khi node đó đủ ổn định
* Không xem xét lịch sử đường đi của vị trí đó hoặc tương lai khi rời khỏi vị trí đó. Việc này sẽ giảm thiểu tác động vấn đề đường chân trời (horizon) mà các AI gặp phải đối với các trò chơi như cờ vua, cờ tướng…
* Tìm kiếm này sẽ mô phỏng như cách ta thường hay lựa chọn một nước cờ tốt thay vì tìm một nước cờ xấu. Thuật toán này sẽ hướng dẫn AI xác định đâu là vị trí không ổn định gây bất lợi, đảm bảo không bị dính vào “bẫy” của đối phương để ước tính nước đi tốt hơn

1.1.6 Depth First Search

* Thuật toán tìm kiếm ưu tiên theo chiều sâu (DFS) là thuật toán duyệt hoặc tìm kiếm trên một cây hoặc là một đồ thị. Khởi đầu tại node gốc (root) và tìm kiếm với từng nhánh con cho tới khi tìm được node cần tìm hoặc tới node không có node con. Khi đó giải thuật quay lui (backtracking) sẽ tìm về node gốc và được lưu lại trong một Stack (không dùng đệ quy)
* Độ phức tạp của DFS thấp hơn của BFS nhưng thời gian tìm kiếm là như nhau O(|V| + |E|)

1.2 Sơ đồ giải thuật

1.2.1 Sơ đồ giải thuật tổng quát

Bắt đầu

Xác định ai đi trước

Người đi trước

Hiển thị nước đi trên bàn cờ

Máy xử lý, phân tích nước đi

Giải thuật minimax

Tìm nước đi tốt nhất

Thắng

Kết thúc

Lấy nước đi người chơi

Thắng

Sai

Đúng

Đúng

Sai

Đúng

Sai

Hình 1.2: Sơ đồ giải thuật tổng quát chương trình

1.2.2 Sơ đồ giải thuật alpha-beta minimax

Input

Duyệt theo chiều sâu (DFS)

Output

Duyệt từng node con

Mức Cực đại

Chọn giá trị MAX

Chọn giá trị MIN

Duyệt tất cả các node

getBestMove

Kiểm tra điều kiện parent node

Thay thế parent node

Xóa nhánh node

Sai

Đúng

Sai

Đúng

Đúng

Sai

Hình 1.3: Sơ đồ giải thuật alpha-beta minimax trong chương trình

1.2.3 Sơ đồ giải thuật negamax

Input

Duyệt theo chiều sâu (DFS)

Output

Duyệt từng node con

Mức Cực đại

Chọn giá trị MAX

Chọn giá trị MIN

Duyệt tất cả các node

getBestMove

Gắn node

max(a,b) = -min(-a,-b)

Sai

Đúng

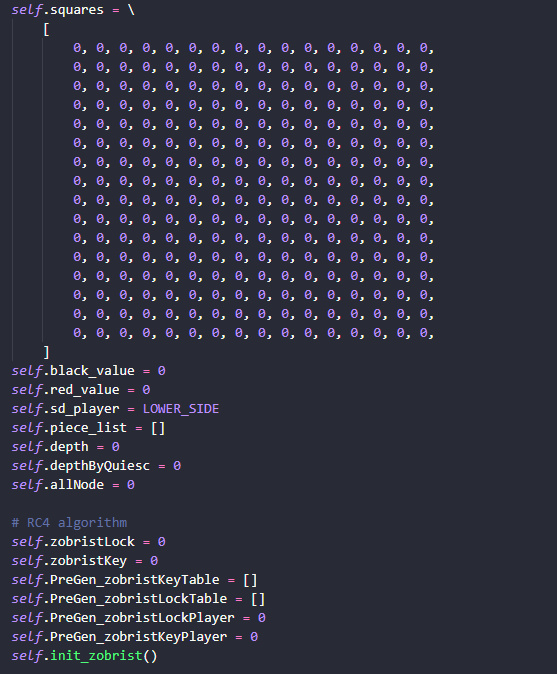
Đúng

Sai

Hình 1.4: Sơ đồ giải thuật negamax của chương trình

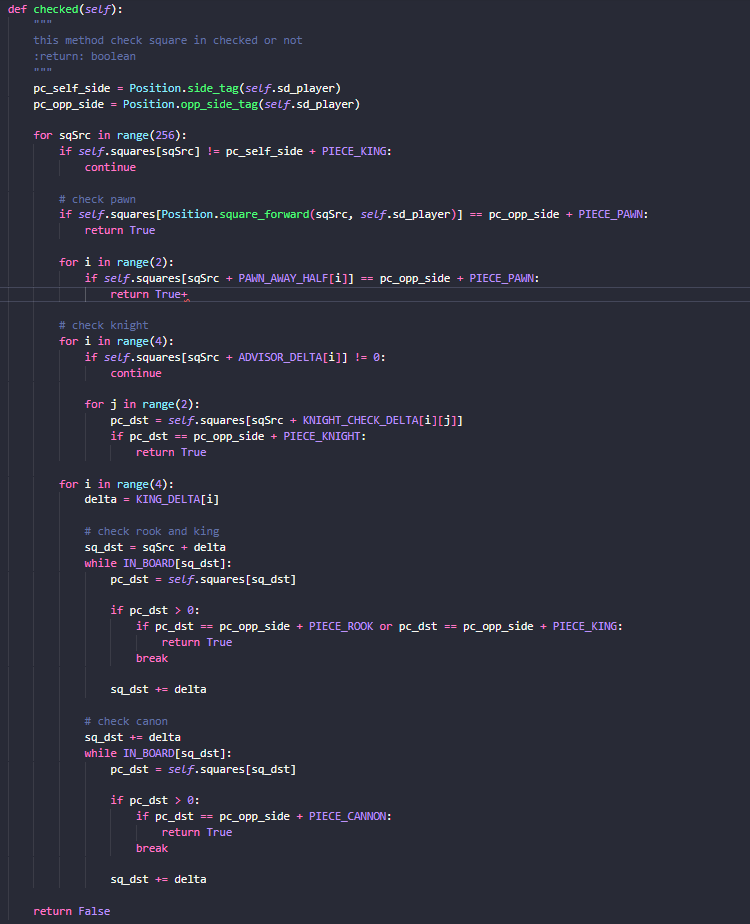
1.3 Hiện thực

* Khởi tạo bàn cờ



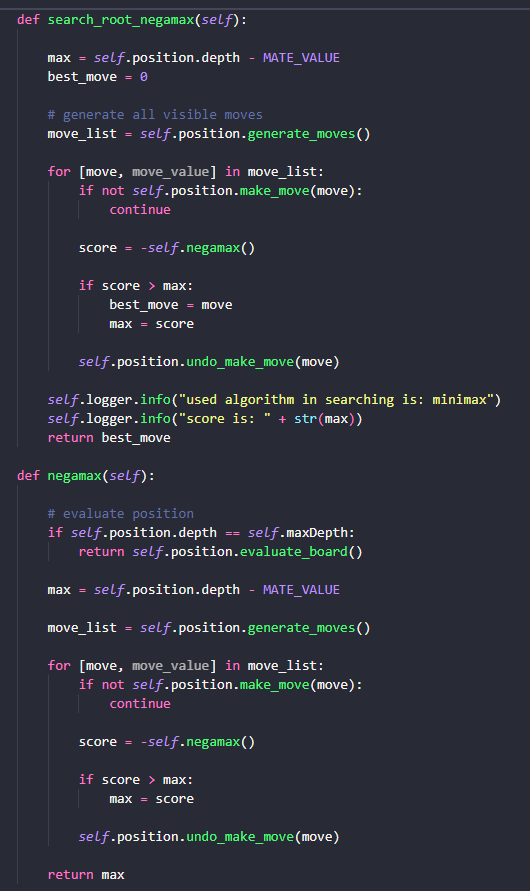
Hình 1.5: Khởi tạo bàn cờ

* Hàm kiểm tra chiếu tướng



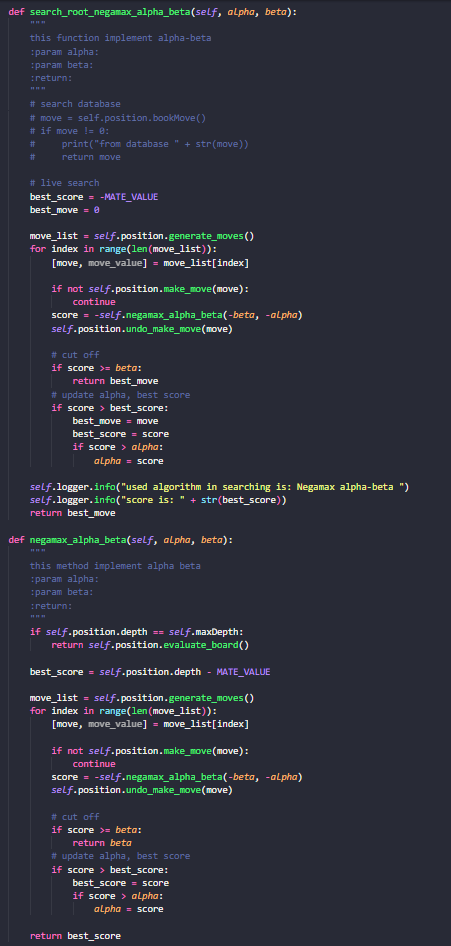
Hình 1.6: Hàm kiểm tra chiếu tướng

* Khởi tạo thuật toán negamax



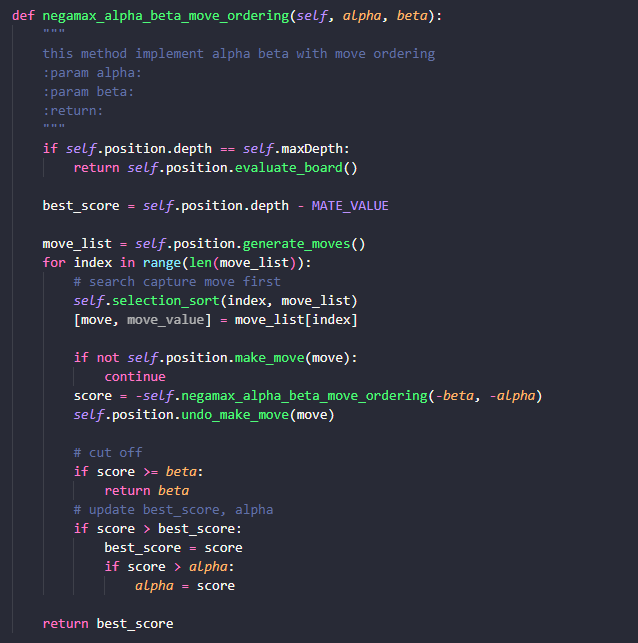
Hình 1.7: Khởi tạo hàm thuật toán negamax

* Áp dụng giải thuật negamax vào giải thuật alpha-beta



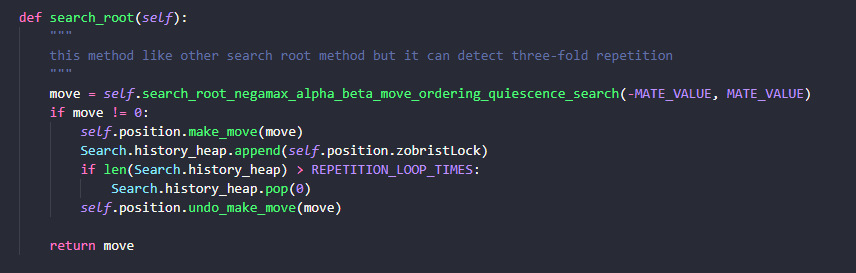
Hình 1.8: Khởi tạo negamax vào giải thuật alpha-beta

* Thuật toán negamax trên alpha-beta có sắp xếp



Hình 1.9: Khởi tạo hàm negamax alpha-beta move ordering

* Khởi tạo tìm node gốc bổ sung thêm tính năng quy tắc lặp lại ba lần



Hình 1.10: Khởi tạo hàm tìm root có quy tắc lặp ba lần

* Tìm kiếm quiescence này sẽ mô phỏng như cách ta thường hay lựa chọn một nước cờ tốt thay vì tìm một nước cờ xấu. Thuật toán này sẽ hướng dẫn AI xác định đâu là vị trí không ổn định gây bất lợi, đảm bảo không bị dính vào “bẫy” của đối phương để ước tính nước đi tốt hơn



Hình 1.11: Khởi tạo hàm tìm kiếm quiescence

1.4 Kết quả và thảo luận

Nhóm đã hiện thực được một chương trình chơi cờ tướng giữa người với máy (AI) qua bài tham khảo mà nhóm đã nghiên cứu trên mạng. Ngoài ra còn có chức năng đấu giữa người với người cùng nhiều tiện ích hỗ trợ khác.

Các giải thuật mà nhóm đã tìm hiểu, với giải thuật minimax nhóm đã bổ sung thêm alpha-beta để rút gọn cây node giúp chương trình hoạt động nhanh hơn. Thuật toán bổ sung Negamax rất phổ biến trong cờ vua và cờ tướng, khi phải phán đoán nước đi của đối thủ ngoài ra phải tìm kiếm nước đi tốt nhất. Cuối cùng là những thuật toán tìm kiếm khác (DFS, Sequential Search, Quiescence Search ) giúp phát triển, bổ sung cho giải thuật negamax

Thuật toán AI của chương trình tuy có độ phức tạp cao, nhưng lại khá đơn giản. Thông qua tìm hiểu thì nước đi của máy là do chương trình tự phân tích và tìm kiếm nước đi tốt nhất, không phải qua trainning. Vì thế giữa khoảng thời gian các nước đi có độ trễ khá cáo. Hướng phát triển của bài toán ta có thể mở rộng ra thành một mô hình AI có thể trainning và phát triển thêm nhiều hướng đi khác.

CHƯƠNG 2

**Rainforest Connection Species Detection**

**BirdCLEF:LIBROSA Audio Feature Extraction**

**(Trích xuất âm thanh)**

2.1 Giới thiệu về bài toán

Yêu cầu bài toán đưa ra là lập trình một chương trình tự động phát hiện âm thanh của các loài chim và ếch trong rừng nhiệt đới.

Ý nghĩa của bài toán:

Sự hiện diện của các loài sinh vật rừng nhiệt đới mang ý nghĩa tốt về tác động của biến đổi khí hậu và mất môi trường sống. Ta dễ dàng nghe thấy tiếng kêu của những động vật trong rừng hơn là nhìn thấy chúng, vì thế công nghệ này có thể hoạt động được trên quy mô toàn cầu. Dữ liệu thực tế rất cần thiết cho việc cung cấp thông qua các kỹ thuật học máy, từ dó có thể phát hiện sớm các tác động của con người đối với môi trường của Trái Đất. Việc này có thể thúc đẩy quá trình quản lý, bảo tồn và bảo vệ hiệu quả hơn.

Các phương pháp truyền thống để đánh giá, phát hiện các loài rất tốn kém và hạn chế về mặt không gian và thời gian. Trong khi đó, nhận dạng âm thanh tự động thông qua deep learning lại nổi trội hơn hẳn, tuy nhiên các mô hình lại yêu cầu một số lượng lớn các mẫu huấn luyện cho mỗi loài. Điều này hạn chế khả năng phát hiện đối với các loài quý hiếm hoặc đơn giản là loài chưa được tìm thấy, vốn là tâm điểm của các nỗ lực bảo tồn. Do đó, các phương pháp để tự động hóa việc phát hiện loài có độ chính xác cao trong môi trường âm thanh ồn ào với dữ liệu huấn luyện hạn chế là thứ chúng ta cần quan tâm.

2.2 Phân tích yêu cầu của bài toán

2.2.1 Yêu cầu của bài toán

Bài toán phát hiện âm thanh có yêu cầu như sau: Dữ liệu truyền vào là âm thanh thực từ một khu rừng nhiệt đới ở bất kì đâu trên Trái Đất. Nhiệm vụ được đưa ra là phải phát hiện được những loài nào đang phát ra âm thanh thông qua dữ liệu đó, nếu là động vật quý hiếm thì càng tốt. Việc phân tích và dự đoán này rất quan trọng trong việc bảo vệ những loài động vật quý hiếm.

Trích xuất đặc điểm (**Audio Feature Extraction**) là quá trình làm nổi bật các đặc điểm , tiếng kêu đặc trưng và tác động mạnh mẽ nhất trong tín hiệu. Phần này sẽ hướng vào trọng tâm là trích xuất những đặc điểm quan trọng trong xử lý âm thanh và từ đó có thể mở rộng nó sang nhiều loại tính năng khác phù hợp với những loại dữ liệu khác.

2.2.2. Các phương pháp giải quyết bài toán

**2.2.2.1. Phương pháp Spectrograms**

Phương pháp biểu đồ phổ (Spectrograms) là một mô tả trực quan về phổ tần số của tín hiệu âm thanh thay đổi theo thời gian. Do đó, phương pháp này bao gồm cả khía cạnh thời gian và tần số của âm thanh. Nó thu được bằng cách áp dụng **Short-Time Fourier Transform** (STFT) trên âm thanh. Nói một cách đơn giản nhất, STFT của tín hiệu được tính toán bằng cách áp dụng **Fast Fourier Transform** (FFT) cục bộ trên các phân đoạn thời gian nhỏ của âm thanh.

**Short-Time Fourier Transform** (STFT), là một phương pháp biến đổi liên quan đến Fourier được sử dụng để xác định tần số hình sin và pha của các phần cục bộ của tín hiệu khi nó thay đổi theo thời gian.

**Fast Fourier Transform** (FFT) là một phương pháp đo quan trọng trong việc đo đạc âm thanh. Nó chuyển đổi tín hiệu thành những quang phổ riêng lẻ, do đó cung cấp được những thông tin tần số về tín hiệu âm thanh. FFT thường được sử dụng để phân tích lỗi, kiểm soát chất lượng và giám sát tình trạng của máy móc hoặc hệ thống. Cụ thể ở đây ta áp dụng vào để giải quyết bài toán.

**2.2.2.2 Phương pháp Mel-Frequency Cepstral Coefficients (MFCCs)**

**Mel-Frequency Cepstral Coefficients (**MFCC) là một cách để trích xuất các đặc trưng (feature extraction) giọng nói (speech) thường được sử dụng trong các model nhận dạng giọng nói (Automatic Speech Recognition) hay phân loại giọng nói (Speech Classification). Như tên gọi của nó, MFCC sẽ cho ra kết quả là các hệ số (coefficients) của cepstral, từ Mel filter trên quang phổ ta lấy được từ các file âm thanh chứa giọng nói.

Cụ thể hơn là phương pháp này sẽ trích xuất những tín hiệu đầu vào, ví dụ input bài toán là những tập audio và tương ứng với chúng là mỗi một âm tiết. Ta sẽ tìm những feature đặc trưng để phân biệt từng âm tiết này. Và thông qua phương pháp MFCC sẽ giúp chúng ta tìm ra những feature kia, từ đó có thể giải quyết được vấn đề mà bài toán đặt ra.

2.2.3 Phương pháp đề xuất giải quyết bài toán

Với 2 phương pháp trên, ta sẽ kết hợp để giải quyết vấn đề của bài toán. Hiện tại yêu cầu bài toán là phân tích được những đặc điểm đặc trưng của từng loài thông qua dữ liệu truyền vào. Vì thế thông qua phân tích quang phổ tần số âm thanh (**Spectrograms**) và âm phổ (**Mel-Frequency Cepstral Coefficients**) sẽ giúp chúng ta phân tích chính xác từng đặc điểm, tần số tiếng nói của mỗi loài.

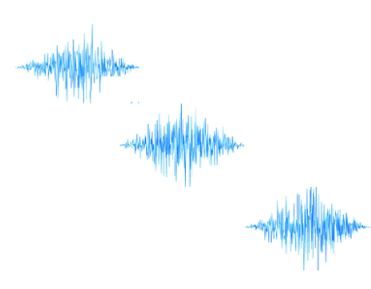
* **Spectrograms**: Phân tích quan phổ tần số âm thanh được sử dụng rộng rãi trong các lĩnh vực âm nhạc, ngôn ngữ học, sonar, radar, xử lý giọng nói, địa chấn học và các lĩnh vực khác. Ở đây chúng ta sử dụng để xác định ngữ âm và để phân loại các tiếng gọi khác nhau của động vật trong rừng nhiệt đới.
* **Mel-Frequency Cepstral Coefficients**: Phân tích âm phổ, phương pháp này sẽ dùng những hệ số phân tích được tạo thành tần số Mel(MFCC). Những dải tần số Mel khác đi một tí so với dải tần số âm thanh mà con người nghe được. Vì thế nó hoạt động rất tốt trong việc phân loại và phân tích âm thanh động vật.

Cuối cùng, thông qua 2 phương pháp trên ta sẽ phân tích được miền thời gian, miền tần số và sẽ biểu diễn toàn bộ giá trị trong 2 phương pháp trên lên biểu đồ thời gian-tần số thực. Do đó sẽ giúp cho việc phân tích rõ ràng và sẽ giải quyết được yêu cầu bài toán nhanh chóng hơn.

2.3 Phương pháp giải quyết bài toán

2.3.1 Mô hình tổng quát

**2.3.1.1 Mô hình tổng quát phương pháp Spectrograms**



**Tín hiệu âm thanh truyền vào**

**Window**

**Segmentation(phân khúc)/Windowing**

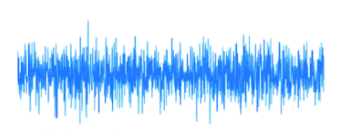
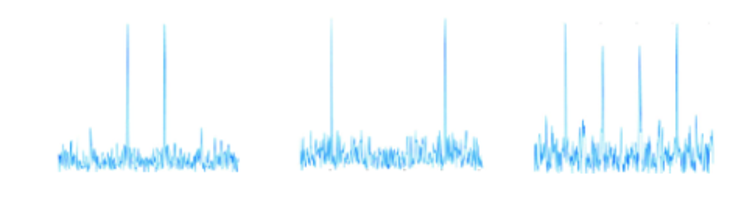
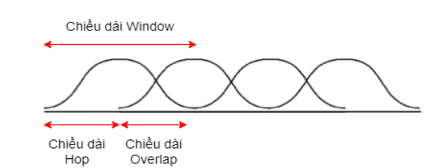
**FFT**

**(Fast Fourier Tranform)**

**FFT Ouput 1**

**FFT Ouput 2**

**FFT Ouput 3**



Hình 2.12: Mô hình tổng quát phương pháp Spectrogram

**2.3.1.2 Mô hình tổng quát phương pháp Mel-Frequency Cepstral Coefficients**

**Tín hiệu âm thanh truyền vào**

**Preemphasis**

**Window**

**DFT  
(Discrete Fourier Transform)**

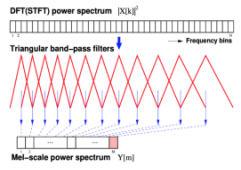
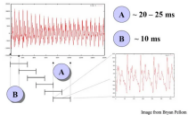
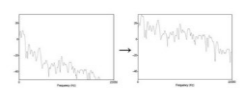
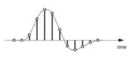
**Mel**

**Filterbank**

**IDFT  
(Inverse Discrete Fourier Transform)**

**MFCCs**

**(Kết quả)**



Hình 2.13:Mô hình tổng quát phương pháp MFCC

2.3.2 Đặc trưng của mô hình đề xuất

**2.3.2.1 Đặc trưng mô hình Spectrograms**

* **Tín hiệu truyền vào:**

Dữ liệu âm thanh truyền vào là một dạng tín hiệu liên tục, trộn lẫn phức tạp các tần số âm thanh khác nhau. Máy tính sẽ nhận dạng và làm việc với nó thành một dạng tập dữ liệu là những con số rời rạc.

* **Window:**

Chúng ta sẽ chia âm thanh truyền vào bằng các khung nhỏ (window) để tính toán FFT cho từng Window. Bằng cách chia từng khúc window như vậy ta sẽ nhận được tần số trong một khoảng thời gian xác định. Để tránh việc làm mất một vài tần số khi lấy window một cách liên tục, ta thường giữ các window chồng lên nhau (overlap). Để nhận dạng giọng nói hay là tiếng chim được áp dụng trong bài, chúng ta sẽ sử dụng window dài khoảng 20-30ms

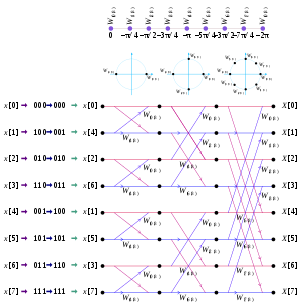
* **Segmentation:**

Với mỗi một window đã phân chia, ta sẽ tách từng tần số riêng biệt đã bị trộn lẫn vào nhau. Đây gọi là thủ thuật windowing hoặc segmentation

* **FFT (Fast Fourier Transform):**

Fast Fourier Transform là một thuật toán hiệu quả để tính toán DFT nhằm phân tích tín hiệu rời rạc theo thời gian trong miền tần số. FFT là cách để đạt được kết quả như DFT nhưng thời gian sẽ rút ngắn hơn nhiều với phép tính trong khi DFT phải mất tận phép tính

Thuật toán FFT phổ biến nhất có lẽ là thuật toán FFT Cooley-Tukey. Là một thuật toán chia để trị bằng việc dùng đệ quy để chia bài toán tính DFT có kích thước thành nhiều bài toán DFT nhỏ hơn như cùng với phép nhân với căn của đơn vị (thừa số xoay)



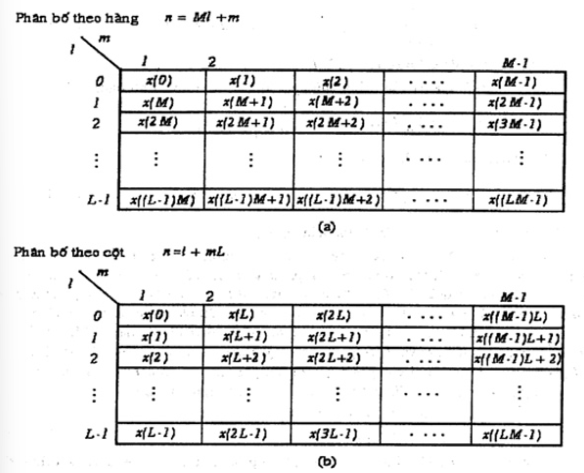
Hình 2.14: Giải thuật FFT Cooley-Tukey cho mảng 8 phần tử

Để so sánh và đánh giá hiệu quả của FFT, ta xem xét số lượng số lượng các phép toán cần được thực hiện trong trường hợp DFT tính trực tiếp. Từ công thức ban đầu của DFT ta có thể viết lại như sau:

(2.1)

(2.2)

Với việc tính toán trực tiếp này đòi hỏi phải có phép tính xác định giá trị hàm lượng giác, phép nhân số thực, phép cộng số thực… Hầu hết các thuật toán FFT sẽ tối ưu hóa quá trình tính toán của DFT theo nhiều cách. Phổ biến là cách phương pháp chia nhỏ tích của hai thừa số nguyên khác nhau . Với cách phân tách trên các phần từ của sẽ chứa trong một mảng chữ nhật theo hai cách (theo hàng và theo cột):



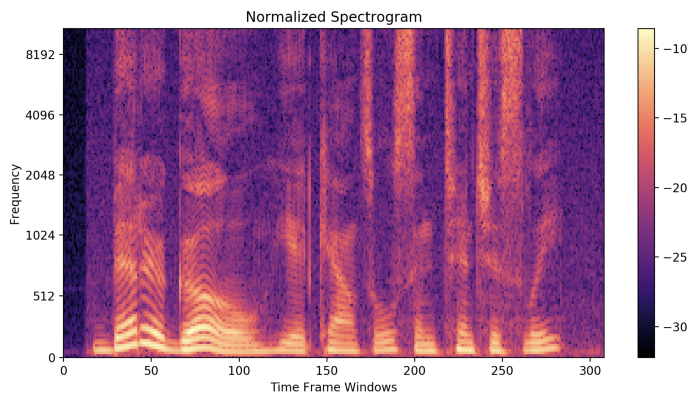
Hình 2.15: Hình mô tả quá trình biến đổi FFT

Từ đó ta có thể thấy kết quả khác của việc tính DFT:

 (2.3)

Ta thấy rằng thực hiện tính DFT bằng cách phân tách như trên bao gồm việc tính các DFT theo L và M. Tuy cảm giác sẽ phức tạp nhưng lại hiệu quả của việc chia nhỏ bằng cách xác định cụ thể số phép toán chủ yếu cần thực hiện. Cụ thể, số lượng phép nhân đã giảm xuống từ thành ). Vậy số lượng phép toán thực hiện đã giảm đi khoảng hai lần, với phép cộng thì cũng giảm theo số lần tương tự. Kết luận rằng ta chỉ cần phép nhân phức và phép cộng phức qua việc thực hiện thuật toán FFT.

Đầu ra của thuật toán FFT là 1 mảng các số đại diện cho các biên độ của các tần số khác nhau trong window. Với ma trận 2D thu được chính là biểu đồ Spectrogram.



Hình 2.16: Biểu đồ ví dụ đầu ra Spectrogram của thuật toán FFT

**2.3.2.2 Đặc trưng mô hình Mel-Frequency Cepstral Coefficients (MFCC)**

* **Tín hiệu truyền vào:**

Dữ liệu âm thanh truyền vào là một dạng tín hiệu liên tục, máy tính sẽ nhận dạng và làm việc với nó thành một dạng tập dữ liệu là những con số rời rạc.

* **Preemphasis:**

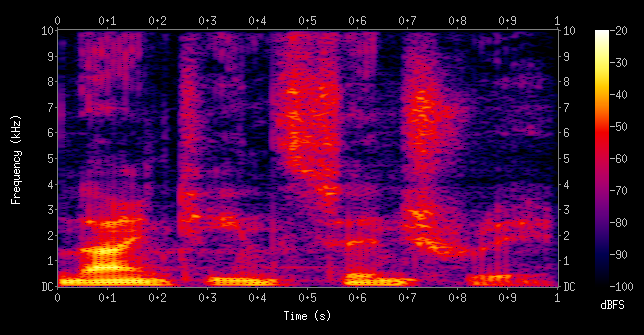
Bước này là dùng để nâng mức năng lượng của âm thanh ở tần số cao truyền vào, lí do là vì theo cấu trúc thanh quản và âm thanh thì những âm thanh tần số thấp thì phát ra năng lượng cao và âm thanh ở tần số cao thì ngược lại. Những âm thanh tần số cao chứa những thông tin quan trọng trong âm vị để phân tích vì thế ta cần Preemphasis ở bước này.

* **Windows:**

Ở bước này ta dùng phương pháp window trượt dọc theo tín hiệu để tạo ra những **Frame** sau đó áp dụng DFT (Discrete Fourier Transform). Ví dụ cụ thể: tốc độ nói trung bình của con người là 3 - 4 từ mỗi giây (tiếng anh), mỗi từ khoảng 3 - 4 âm tiết, mỗi âm tiết lại chia ra 3 - 4 phần. Do đó trong 1 giây âm thanh chia ra thành 36 - 40 phần (Frame) ta chọn độ rộng cho mỗi frame khoảng 20 - 25 ms sẽ đủ để quan sát được một phần tín hiệu âm thanh truyền vào, từ đó áp dụng với bài toán đưa ra là các loài động vật trong rừng nhiệt đới.

* **DFT:**

**Discrete Fourier Transform** ở đầy sau khi áp dụng phương pháp DFT kết quả thu được của chúng ta sẽ là một biểu đồ quang phổ âm thanh (Spectrogram).

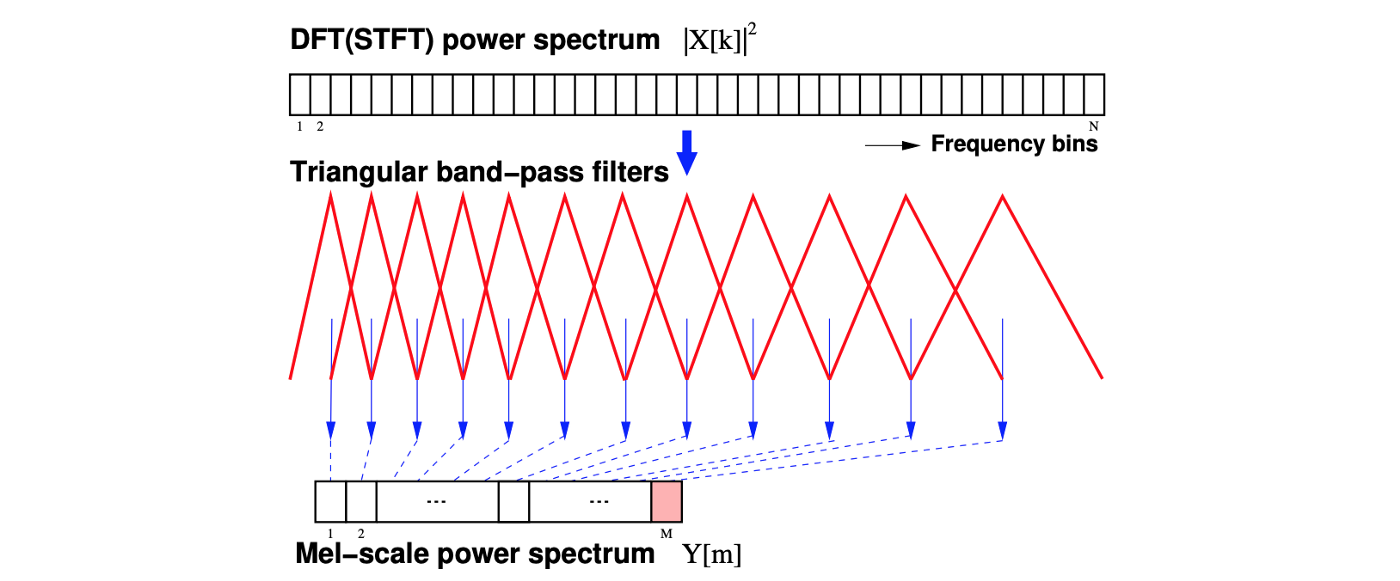


Hình 2.17: Ví dụ về quang phổ âm thanh của đầu ra DFT

Từ quang phổ trên, ta dựa vào những tần số âm thanh đặc trưng gọi là **Formant** (một thuật ngữ dùng trong chuyên ngành xử lý tiếng nói) được chia ra nhiều tần số bao gồm F1, F2, F3, … Ta dựa vào vị trí, thời gian và sự thay đổi của nó để xem xét nó là âm thanh của âm vị nào. Nhưng trong nhiều trường hợp thì không thực sự đúng toàn bộ vì thế ta phải trải qua thêm nhiều bước xử lý nữa để thu được đầu ra **MFCC**, tốt hơn, phổ biến và chính xác hơn.

* **Mel filterbank**:

Theo như cảm âm của tai chúng ta thì nó là phi tuyến tính do đó sẽ khác so với những thiết bị đo. Tai chúng ta cảm nhận rất tốt ở tần số thấp và ngược lại với tần số cao, áp dụng một cơ chế mapping như vậy cho bài toán chúng ta.



Hình 2.18: Ví dụ về mô hình Mel filterbank

Ta sẽ bình phương những giá trị trong spectrogram ở bước trên ta sẽ thu được **DFT power spectrum** (phổ công suất). Sau đó, sẽ dùng tập bộ lọc dải **Mel-scale filter** lên trên từng khoảng tần số (mỗi dải tần xác định sẽ được áp dụng mỗi filter riêng). Đầu ra của mỗi filter là năng lượng dải tần mà nó bao phủ. Thu được **Mel-scale power spectrum** (phổ năng lượng công suất âm thanh).

* **IDFT (Inverse Discrete Fourier Transform):**

Như đã mô tả ở những bước trước, trong giọng nói âm thanh sẽ có những tần số như: F0 - tần số cơ bản và **formant** (tần số đặc trưng) F1, F2, F3 ,... Thông tin về tần số F0 không giúp ích trong việc nhận dạng âm thanh đặc trưng vì thế ở bước này ta phải loại bỏ những dữ liệu hay thông tin về F0 đi, để mô hình chúng ta dễ dàng phân tích hơn và không bị ánh hưởng bởi F0.

Phép biến đổi IDFT sẽ cho đầu ra là những Cepstral features sau khi đã loại bỏ những thông tin liên quan tới F0.

* **MFCCs:**

Ở bước cuối cùng sau khi đã thu được Cepstral features ta sẽ extract feature cuối cùng của frame làm năng lượng của chính nó, cách tính theo công thức bên dưới. Feature này giúp chúng ta phân biệt các âm tiết.

 (2.4)

Trong xử lý tiếng nói, môi trường và thời gian thay đổi rất quan trọng vì thế sự thay đổi ở thời điểm bắt đầu và kết thúc rất rõ rệt tùy thuộc vào phụ âm. Vì thế ở feature cuối cùng sẽ là đạo hàm của toàn bộ của những feature trước theo công thức từ frame thứ t đến frame t+1:

 (2.5)

Và giá trị toàn bộ những giá trị Feature MFCC là sự thay đổi d(t) theo thời gian đạo hàm của d(t) cũng chính là đạo hàm bậc 2 của c(t) theo công thức:

 (2.6)

Cuối cùng qua toàn bộ quá trình trên ta sẽ tạo ra được MFCC (feature transform).

2.4 Thực nghiệm

2.4.1 Dữ liệu

Dữ liệu thực nghiệm được lấy từ cuộc thi trên Kaggle **Bird CLEF 2021 - Bird call Identification,** mô tả cuộc thi: **Identify bird calls in soundscape recordings.**

Dữ liệu truyền vào của bài toán trên tương tự như bài toán mà nhóm đang giải quyết (**Rainforest Connection Species Audio Detection**), nhưng ở đây tín hiệu âm thanh truyền vào là dùng để phân tích các loài chim. Ta có thể áp dụng thực tế vào bài toán rừng rậm nhiệt đới dùng để phân tích thêm các loài khác.

Data csv gồm có, tên loài chim, đặc điểm cụ thể cách chúng phát ra âm thanh, địa điểm thu tín hiệu (kinh độ và vĩ độ), tác giả dữ liệu, thời gian thu âm, tên file, giấy phép,.…

Link Data public có sẵn từ bài toán:

[https://www.kaggle.com/code/shreyasajal/birdclef-librosa-audio-feature-extraction/data](https://www.kaggle.com/code/shreyasajal/birdclef-librosa-audio-feature-extraction/data" \o "https://www.kaggle.com/code/shreyasajal/birdclef-librosa-audio-feature-extraction/data" \t "https://discord.com/channels/1043147365361659904/_blank)

2.4.2 Xử lý dữ liệu

Vì dữ liệu truyền vào là một tín hiệu âm thanh, nên bắt buộc phải tiền xử lý dữ liệu trước sau đó chúng ta mới áp dụng trong mô hình. Dữ liệu âm thanh là một chuỗi số rời rạc, việc xử lý dữ liệu là một phần bắt buộc trước khi áp dụng mô hình tránh việc nó không nhận dạng được. Mặt khác, xử lý tiền dữ liệu sẽ giúp cho việc training nhanh hơn so với bình thường.

Ta dùng phương pháp **Spectrogram** để tiền xử lý dữ liệu, cụ thể là phân tích tín hiệu đầu vào thành dạng biểu diễn biểu đồ quang phổ trực quan thay đổi theo thời gian. Biểu đồ sẽ thể hiện những giá trị mức năng lượng (dB) dưới dạng sóng cụ thể.

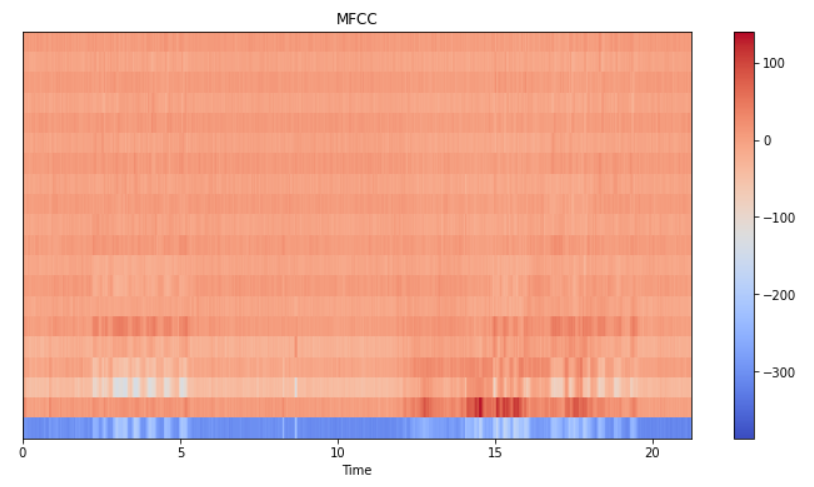
**2.4.3 Công nghệ sử dụng**

Ngôn ngữ lập trình được sử dụng là Python một ngôn ngữ lập trình khá phổ biến trong ngành AI và học máy.

Những thư viện được sử dụng trong bài toán bao gồm: Os, pandas, torch, torchaudio, numpy, seaborn, matplotlib, plotly, librosa, IPython, sklearn, warnings.

nhóm sử dụng Google colab và Kaggle để lập trình với mục đích là lấy data trực tiếp từ cuộc thi dễ dàng hơn tiện lợi trong việc xử lý dữ liệu.

**2.4.4 Cách đánh giá**



Hình 2.19: Biểu đồ output feature cúa một loài chim của nhóm

Mục tiêu của nhóm là tạo ra được một feature đặc trưng cho loài từ đó mới bắt đầu đưa vào thực tế là nhận dạng âm thanh từ môi trường. Thông qua chương trình trên nhóm đã có thể dự đoán được feature đặc trưng của loài chim đó.

2.5 Kết quả đạt được

**Graphical user interface, text

Description automatically generated**

Hình 2.20: Thực nghiệm về phương pháp Mel Spectrogram

* Sau khi áp dụng phương pháp Mel Spectrogram, nhóm đã thống kê được biểu đồ quang phổ biểu diễn tần số theo thời gian

**Text

Description automatically generated**

Hình 2.21: Thực nghiệm về phương pháp Spectral centroids

* Áp dụng thuật toán Spectral centroids, nhóm em đã thống kê được các trọng tâm (centroids) của các vector lấy từ dữ liệu đầu vào trong từng frame theo thời gian, đồng thời thể hiện dưới dạng đồ thị

Text

Description automatically generated

Hình 2.22: Thực nghiệm phương pháp Log Frequency Spectrogram

* Với phương pháp Log Frequency Spectrogram, nhóm đã thể hiện được đồ thị quang phổ từ tiếng chim của loài Amecro

****

Hình 2.23: Đầu ra của chương trình qua phương pháp MFCCs

* Phương pháp MFCCs nhóm em đã thực hiện sẽ lọc các đặc trưng của giọng nói, thể hiện qua các hệ số cesepstral từ Mel filter để thu thập dữ liệu tiếng chim của amecro

2.6 Kết luận

**Kết quả đạt được:** nhóm đã trích xuất được feature đặc trưng cho tín hiệu âm thanh truyền vào, cụ thể ở đây là các loài chim. Mục tiêu chính của bài nhóm em là tìm feature vì thế, sau khi trích xuất được những đặc trưng của từng loài thì ta có thể áp dụng vào những mô hình sau này để phát hiện dễ dàng.

**Hạn chế:** Chính vì phụ thuộc vào dữ liệu truyền vào để tiến hành trích xuất rồi mới training, điều này dẫn đến những hạn chế sau:

* Nếu đó là loài động vật quý hiếm hoặc chưa từng xuất hiện, việc lấy dữ liệu âm thanh để xử lý gần như là không thể.
* Những loài động vật sống tự nhiên để thu được âm thanh thì dữ liệu truyền vào sẽ rất khác so với việc thu trực tiếp từ việc nuôi nhốt. Ngoài ra trước khi giải quyết bài toán phân biệt âm thanh, ta bắt buộc phải trích xuất được âm thanh của loài đó là gì trước rồi mới phân tích và áp dụng.
* Cuối cùng là việc áp dụng công nghệ vào thực tế rất khó, vì động vật không ở tại một vị trí nhất định. Ngoài ra vị trí địa lý và môi trường sẽ khiến tiếng kêu khác đi rất nhiều, dẫn đến tỷ lệ dự đoán càng thấp.

**Hướng phát triển:** Với cách giải quyết bài toán của nhóm như trên là tìm tiếng kêu đặc trưng của mỗi loài chim với dữ liệu train truyền vào. Thì ta có thể áp dụng nó cho bài nhận biết tiếng kêu của từng loài trong rừng rậm nhiệt đới, hiện tại có rất nhiều data để chúng ta train mô hình trước khi áp dụng vào thực tế, giúp cho việc dự đoán tốt hơn nhanh hơn.

Ngoài việc áp dụng trong việc nhận biết đặc trưng của các loài động vật trong rừng nhiệt đới, ta có thể mở rộng ra phân tích âm thanh con người hoặc dưới lòng đại dương (nơi con người chúng ta vẫn chưa khám phá hết) và còn nhiều cách phát triển khác mà chúng ta có thể hướng tới,…

**TÀI LIỆU THAM KHẢO**

1. *Mini-max-algorithm-in-ai Tutorials - Javatpoint*. (n.d.). www.javatpoint.com. Retrieved December 5, 2022, from [https://www.javatpoint.com/mini-max-algorithm-in-ai](https://www.javatpoint.com/mini-max-algorithm-in-ai)](https://www.javatpoint.com/mini-max-algorithm-in-ai](https:/www.javatpoint.com/mini-max-algorithm-in-ai))
2. *Ai-alpha-beta-pruning 5D Tutorials - Javatpoint*. (n.d.). www.javatpoint.com. Retrieved December 5, 2022, from [https://www.javatpoint.com/ai-alpha-beta-pruning](https://www.javatpoint.com/ai-alpha-beta-pruning)](https://www.javatpoint.com/ai-alpha-beta-pruning](https:/www.javatpoint.com/ai-alpha-beta-pruning))
3. George T. Heineman; Gary Pollice & Stanley Selkow (2008). "Chapter 7:Path Finding in AI". Algorithms in a Nutshell. [Oreilly Media] (<https://en.wikipedia.org/wiki/Oreilly_Media>)
4. John P. Fishburn (1984). "Appendix A: Some Optimizations of α-β Search". Analysis of Speedup in Distributed Algorithms (revision of 1981 PhD thesis). [UMI Research Press] (<https://en.wikipedia.org/wiki/UMI_Research_Press>)
5. Breuker, Dennis M. [Memory versus Search in Games] (<http://breukerd.home.xs4all.nl/thesis/> ), Maastricht University, October 16, 1998.
6. [Knuth 1998] (<https://en.wikipedia.org/wiki/Linear_search#CITEREFKnuth1998>), §6.1 ("Sequential search").
7. Beal, Don (April 1990). ["A generalised quiescence search algorithm"] (<https://www.sciencedirect.com/science/article/abs/pii/0004370290900728> ). Artificial Intelligence.
8. [Thomas H. Cormen](<https://en.wikipedia.org/wiki/Thomas_H._Cormen>), [Charles E. Leiserson] [Introduction to Algorithms], Second Edition. MIT Press and McGraw-Hill, 2001. [ISBN] [0-262-03293-7] (<https://en.wikipedia.org/wiki/Special:BookSources/0-262-03293-7>). Section 22.3: Depth-first search
9. Cooley, James W.; [Tukey, John W.] (1965). [“An algorithm for the machine calculation of complex Fourier series”](<https://archive.org/details/sim_mathematics-of-computation_1965-04_19_90/page/297>)
10. Chaudhary, K. (2021, December 13). *Understanding Audio data, Fourier Transform, FFT and Spectrogram features for a Speech Recognition System*. Medium. Retrieved December 5, 2022, from <https://towardsdatascience.com/understanding-audio-data-fourier-transform-fft-spectrogram-and-speech-recognition-a4072d228520>
11. Jeon, Hohyub & Jung, Yongchul & Lee, Seongjoo & Jung, Yunho. (2020). Area-Efficient Short-Time Fourier Transform Processor for Time–Frequency Analysis of Non-Stationary Signals. Applied Sciences. 10. 7208. 10.3390/app10207208.
12. *FFT*. (n.d.). Retrieved December 5, 2022, from <https://www.nti-audio.com/en/support/know-how/fast-fourier-transform-fft>
13. Huy, N. L. (2022, December 5). *Sơ lược về Mel Frequency Cepstral Coefficients (MFCCs)*. Viblo. Retrieved December 5, 2022, from <https://viblo.asia/p/so-luoc-ve-mel-frequency-cepstral-coefficients-mfccs-1VgZv1m2KAw>
14. S, S. (2022, February 23). *BirdCLEF:LIBROSA Audio Feature Extraction*. Kaggle. Retrieved December 5, 2022, from <https://www.kaggle.com/code/shreyasajal/birdclef-librosa-audio-feature-extraction/data>

**TỰ ĐÁNH GIÁ**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Câu | Nội dung | Điểm chuẩn | Tự chấm | Ghi chú |
| 1  (4.0) | **1.1 Mô tả cấu trúc dữ liệu** | 0.5 | 0.5 |  |
| **1.2 Sơ đồ giải thuật** | 1 | 0.5 |  |
| **1.3 Hiện thực** | 2 | 1 |  |
| **1.4 Kết quả và thảo luận** | 0.5 | 0.5 |  |
| 2  (4.5) | **2.1 Giới thiệu về bài toán** | 0.25 | 0.25 |  |
| **2.2 Phân tích yêu cầu của bài toán** | 1 | 1 |  |
| **2.3 Phương pháp giải quyết bài toán** | 1.25 | 0.75 |  |
| **2.4 Thực nghiệm** | 1 | 0.5 |  |
| **2.5 Kết quả đạt được** | 0.75 | 0.5 |  |
| **2.6 Kết luận** | 0.25 | 0.25 |  |
| 3 | **Điểm nhóm** | 0.5đ | 0.5 |  |
| 4 | **Báo cáo (**chú ý các chú ý 2,3,4,6 ở trang trước, nếu sai sẽ bị trừ điểm nặng**)** | 1đ | 0.75 |  |
| **Tổng điểm** | | | 7 |  |

Bảng 1: Bảng tự đánh giá