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**

**6**

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

*Người thực hiện*: **HOÀNG KIẾN THIẾT – 51702187**

**VÕ NHẬT DUY – 51702092**

Lớp **: 17050202**

Khoá  **: 21**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2021**

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**

**6**

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

Người thực hiện: **HOÀNG KIẾN THIẾT**

**VÕ NHẬT DUY**

Lớp **: 17050202**

Khoá  **: 21**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2021**

LỜI CẢM ƠN

Chúng em xin gửi lời cảm ơn đến thầy Bùi Thanh Hùng, nhờ có sự tận tình giảng dạy của thầy mà chúng em mới có đủ kiến thức để hoàn thành môn học này. Cảm ơn Khoa Công nghệ thông tin trường Đạt học Tôn Đức Thắng đã tạo mọi điều kiện tốt nhất cho chúng em tiếp cận với những kiến thức hữu ích. Chúng em xin chân thành cám ơn.

**ĐỒ ÁN ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là sản phẩm đồ án của 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 tháng năm*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Hoàng Kiến Thiết*

*Võ Nhật Duy*

PHẦN ĐÁNH GIÁ CỦA GIẢNG VIÊN

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

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

TÓM TẮT

Bài 1 yêu cầu xây dưng một từ điển sử dụng Hash Tables chứa các cặp key-value tương ứng với từ tiếng Anh- nghĩa tiếng Việt. Để tiếp cận vấn đề, chúng em xây dựng Hash Tables dựa trên việc xây dựng một hàm Hash bằng Division method. Ngoài ra, trong quá trình xây dựn Hash Table xảy ra vấn đề đụng độ trong Hash Tables, từ đó chúng em giải quyết bằng Chianing method. Trong quá trình xây dựn từ điển, chúng em phát hiện cấu trúc dữ liệu Hash Tables rất hữu ích cho việc quản lý các cặp key-value để giải quyết các bài toán tương tự. Kết quả đạt được sau khi hoàn thành bài 1 là một ứng dụng từ điển xây dựng trên cấu trúc dữ liệu Hash Tables có giao diện và các chức năng cơ bản của một ứng dụng từ điển. Đi cùng với đó là các kiến thức cũng như kinh nghiệm có được khi nghiên cứu và thực hành trên cấu trúc dữ liệu Hash Tables.

Bài 2 là một cuộc thi gắn nhãn âm thanh trên trang Kaggle, cuộc thi có tên Freesound Audio Tagging 2019. Đây là một cuộc thi rất hấp dẫn, nó thử thách người chơi xây dựng một hệ thống gắn nhãn âm thanh tự động dựa trên tập dữ liệu FSDK2019 do Kaggle cung cấp. Cuộc thi đã kết thúc trong năm 2019 và đã nhận được sự tham gia, ủng hộ đông đảo của người chơi trên toàn thế giới. Trong số đó, chúng em chọn ra 3 phương pháp nổi bật để phân tích và chọn ra một phương pháp thích hợp nhất trong ba phương pháp trên để nghiên cứu cũng như hiện thực hóa bài toán này. Kết quả đạt được là một hệ thống gắn nhãn âm thanh đơn giản mô phỏng lại từ phương pháp mà em phân tích và các kiến thức cực kỳ hữu ích về mạng nơ-ron nhân tạo.

MỤC LỤC

[LỜI CẢM ƠN i](#_Toc71115367)

[PHẦN ĐÁNH GIÁ CỦA GIẢNG VIÊN iii](#_Toc71115368)

[TÓM TẮT iv](#_Toc71115369)

[MỤC LỤC 1](#_Toc71115370)

[DANH MỤC CÁC HÌNH VẼ 4](#_Toc71115371)

[CHƯƠNG 1 7](#_Toc71115372)

[1.1 Mô tả cấu trúc dữ liệu 7](#_Toc71115373)

[1.2 Sơ đồ giải thuật 11](#_Toc71115374)

[1.2.1 Sơ đồ giải thuật tổng quát 11](#_Toc71115375)

[1.2.2 Sơ đồ giải thuật chức năng Add/ Edit 12](#_Toc71115376)

[1.2.3 Sơ đồ giải thuật chức năng Uploads và Create từ điển 13](#_Toc71115377)

[1.2.4 Sơ đồ giải thuật chức năng Search 15](#_Toc71115378)

[1.2.4 Sơ đồ giải thuật chức năng Delete 16](#_Toc71115379)

[1.3 Hiện thực 17](#_Toc71115380)

[1.3.1 Giao diện và các chức năng của từ điển 17](#_Toc71115381)

[1.3.2 Hiện thực chức năng Upload Files 19](#_Toc71115382)

[1.3.3 Hiện thực chức năng Show 23](#_Toc71115383)

[1.3.4 Hiện thực chức năng Search 24](#_Toc71115384)

[1.3.5 Hiện thực chức năng Delete 25](#_Toc71115385)

[1.3.6 Hiện thực chức năng Add 26](#_Toc71115386)

[1.3.7 Hiện thực chức năng Edit 27](#_Toc71115387)

[1.3.8 Hiện thực chức năng Information Hash Table và Exit 28](#_Toc71115388)

[1.4 Kết quả và thảo luận 29](#_Toc71115389)

[1.4.1 Kết quả đạt được 29](#_Toc71115390)

[1.4.2 Đánh giá giải thuật 30](#_Toc71115391)

[1.4.3 Thảo luận về kết quả và giải thuật đã hiện thực 31](#_Toc71115392)

[CHƯƠNG 2 32](#_Toc71115393)

[2.1 Giới thiệu về bài toán 32](#_Toc71115394)

[2.2 Phân tích yêu cầu của bài toán 32](#_Toc71115395)

[2.2.1 Yêu cầu của bài toán 32](#_Toc71115396)

[2.2.2 Các phương pháp giải quyết bài toán 33](#_Toc71115397)

[2.2.2.1 Phương pháp “Audio Tagging with Noisy Labels and Minimal Supervision” 33](#_Toc71115398)

[2.2.2.2 Phương pháp “Convolutional Recurrent Neural Network and Data Augmentation for Audio Tagging with Noisy Labels and Minimal Supervision” 36](#_Toc71115399)

[2.2.2.3 Phương pháp “Audio Tagging using Linear Noise Modelling Layer” 38](#_Toc71115400)

[2.2.3 Phương pháp đề xuất giải quyết bài toán 40](#_Toc71115401)

[2.3 Phương pháp giải quyết bài toán 41](#_Toc71115402)

[2.3.1 Mô hình tổng quát 41](#_Toc71115403)

[2.3.2 Đặc trưng của mô hình đề xuất 43](#_Toc71115404)

[2.3.2.1 Data Augmentation 43](#_Toc71115405)

[2.2.3.2 Xây dựng mô hình mạng nơ-ron nhân tạo 47](#_Toc71115406)

[2.3.2.4 Phương pháp huấn luyện 49](#_Toc71115407)

[2.4 Thực nghiệm 50](#_Toc71115408)

[2.4.1 Dữ liệu 50](#_Toc71115409)

[2.4.2 Xử lý dữ liệu 50](#_Toc71115410)

[2.4.3 Công nghệ sử dụng 55](#_Toc71115411)

[2.4.4 Cách đánh giá 55](#_Toc71115412)

[2.5 Kết quả đạt được 56](#_Toc71115413)

[2.6 Kết luận 58](#_Toc71115414)

[TỰ ĐÁNH GIÁ 62](#_Toc71115415)

DANH MỤC CÁC HÌNH VẼ

[Hình 1. Khởi tạo Hash Tables 8](#_Toc70458968)

[Hình 2. Áp dụng Chaining Method 9](#_Toc70458969)

[Hình 3. Minh họa cho Chaining Method 9](#_Toc70458970)

[Hình 4. Áp dụng Division Method 10](#_Toc70458971)

[Hình 5. Sơ đồ giải thuật tổng quát 11](#_Toc70458972)

[Hình 6. Sơ đồ giải thuật chức năng Add/ Edit 12](#_Toc70458973)

[Hình 7. Sơ đồ giải thuật chức năng Uploads và Create từ điển 15](#_Toc70458974)

[Hình 8. Sơ đồ giải thuật chức năng Search 15](#_Toc70458975)

[Hình 9. Sơ đồ giải thuật chức năng Delete 16](#_Toc70458976)

[Hình 10. Giao diện chính của ứng dụng từ điển 17](#_Toc70458977)

[Hình 11. Trường hợp người dùng tìm kiếm một từ không có trong từ điển 18](#_Toc70458978)

[Hình 12. Giao diện cho phép người dùng chọn file 20](#_Toc70458979)

[Hình 13. Thông báo đến người dùng khi upload từ điển thành công. 20](#_Toc70458980)

[Hình 14. Sự kiện “Submit” 21](#_Toc70458981)

[Hình 15. Kết quả uploads hiển thị trên terminal 22](#_Toc70458982)

[Hình 16. Hàm add 23](#_Toc70458983)

[Hình 17. Hàm get\_all 23](#_Toc70458984)

[Hình 18. Sự kiện “Show” 24](#_Toc70458985)

[Hình 19. Sự kiện “Search” 24](#_Toc70458986)

[Hình 20. Hàm get 25](#_Toc70458987)

[Hình 21. Sự kiện “Delete” 25](#_Toc70458988)

[Hình 22. Hàm delete 26](#_Toc70458989)

[Hình 23. Sự kiện “Add” 26](#_Toc70458990)

[Hình 24. Sự kiện “Edit” 27](#_Toc70458991)

[Hình 25. Hàm insert 28](#_Toc70458992)

[Hình 26. Giao diện hiển thị thông tin Hash Tables 29](#_Toc70458993)

[Hình 27. Minh họa cho bài toán Freesound Audio Tagging 32](#_Toc70458994)

[Hình 28. Mô hình tổng quát của phương pháp 43](#_Toc70458995)

[Hình 29. Đồ thị biểu diễn cho kết quả của hàm cong vênh tuyến tính từng mảnh 45](#_Toc70458996)

[Hình 30. Hiệu suất khi thêm dần các phương pháp tăng dữ liệu. 46](#_Toc70458997)

[Hình 31. Convolutional Recurrent Neural Network cho Audio Tagging với các định dạng đầu ra của mỗi khối. Mỗi ConvXd sử dụng kích thước hạt nhân là 3 và một bước tiến và bao gồm BatchNorm và ReLU. 47](#_Toc70458998)

[Hình 32. Hàm split\_dataset 51](#_Toc70458999)

[Hình 33. Đối tượng AudioReader 52](#_Toc70459000)

[Hình 34. Chuyển dữ liệu audio thành STFT 53](#_Toc70459001)

[Hình 35. Normalize tập dữ liệu và kết thúc quá trình tiền xử lý dữ liệu: 54](#_Toc70459002)

[Hình 36. Đánh giá mô hình 56](#_Toc70459003)

[Hình 37. Kết quả cho các phương pháp xử lý nhiễu nhãn khác nhau 57](#_Toc70459004)

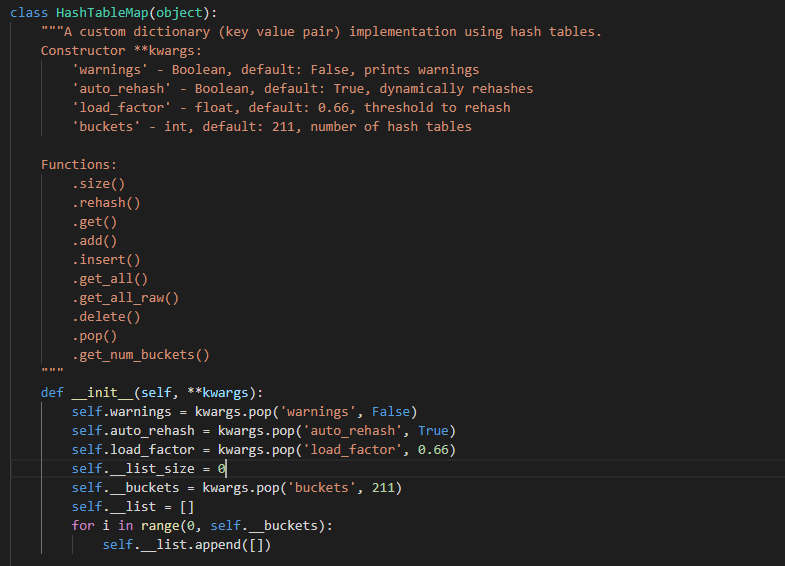
[Hình 38. Đánh giá các tập hợp của các mô hình khác nhau 58](#_Toc70459005)

CHƯƠNG 1

**TỪ ĐIỂN SỬ DỤNG HASH**

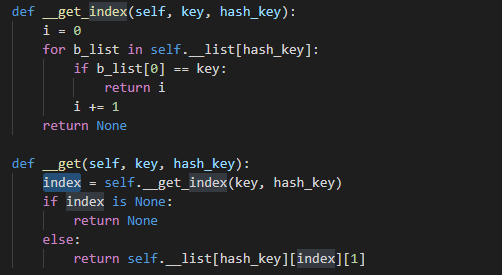
1.1 Mô tả cấu trúc dữ liệu

Từ điển được xây dựng dựa trên cấu trúc dữ liệu Hash Tables. Hash Tables là một cấu trúc dữ liệu ánh xạ các khóa (keys) đến các giá trị (values) tương ứng. Nó sử dụng một hàm băm ( hash function) để tính toán chỉ số cho khóa của dữ liệu và các khóa này được lưu trữ trong chỉ mục (index). Cụ thể, phần tạo Hash Tables trong chương trình như sau:



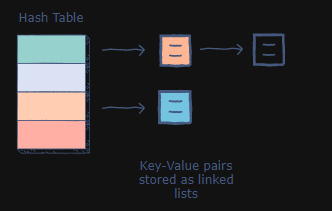
Hình . Khởi tạo Hash Tables

Trong quá trình xây dựng Hash Tables, việc xảy ra xung đột khi hai khóa được băm thành cùng một index trong bảng băm. Đây là một vấn đề cần phải giải quyết vì mọi vị trí trong bảng băm được cho là lưu trữ một phần tử duy nhất. Chúng em sử dụng Chaining Method để giải quyết đụng độ trên:



Hình . Áp dụng Chaining Method

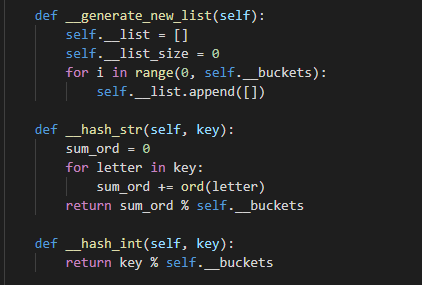
Trong cách tiếp cận của Chaining Method, bảng băm là một mảng các danh sách được liên kết, tức là mỗi chỉ mục có danh sách liên kết của riêng nó. Tất cả các cặp key-value sẽ ánh xạ tới cùng một chỉ mục và sẽ được lưu trữ trong danh sách liên kết của chỉ mục đó.



Hình . Minh họa cho Chaining Method

Thông qua Chaining, việc chèn trong bảng băm luôn xảy ra trong O (1) vì danh sách được liên kết cho phép chèn trong thời gian không đổi. Về mặt lý thuyết, một Hash Tables với Chaining có thể phát triển vô hạn miễn là có đủ không gian. Ngoài ra, một Hash Table sử dụng Chaining Method sẽ không bao giờ cần phải thay đổi kích thước.

Có nhiều cách để tạo một Hash Function, trong ứng dụng từ điển này, chúng em sẽ sử dụng phương pháp Division Method bởi đây là phương pháp đơn giản và phổ biến nhất để tạo một Hash Function. Dưới đây là đoạn code mô tả việc áp dụng Division Method để xây dựng Hash Function:



Hình . Áp dụng Division Method

Bản chất của Division Method nằm ở công thức h(k) = k mod(n). Ở đây, h(k) là giá trị băm thu được bằng cách chia giá trị khóa k cho kích thước của bảng băm n và lấy phần dư. Trong một số tình huống nhất định, ta cần cân nhắc trong việc chọn một giá trị thích hợp cho n. Đa phần ta sẽ có xu hướng chọn n là một số lẻ để thuận tiện cho việc chia, điều này có nghĩa là h (x) là chẵn nếu x là chẵn và h(x) là số lẻ nếu x là số lẻ. Nếu tất cả các khóa có thể có là tương đương nhau thì đây không phải là vấn đề. Tuy nhiên, giả sử nếu các khóa chẵn có nhiều khả năng xuất hiện hơn các khóa lẻ thì Hash Function sẽ không trải đều các giá trị băm của các khóa đó. Trường hợp tốt nhất thì n là số nguyên tố để đảm bảo các khóa được phân phối đồng đều.

Một ví dụ về Division Method như sau:

k = 1276

n = 10

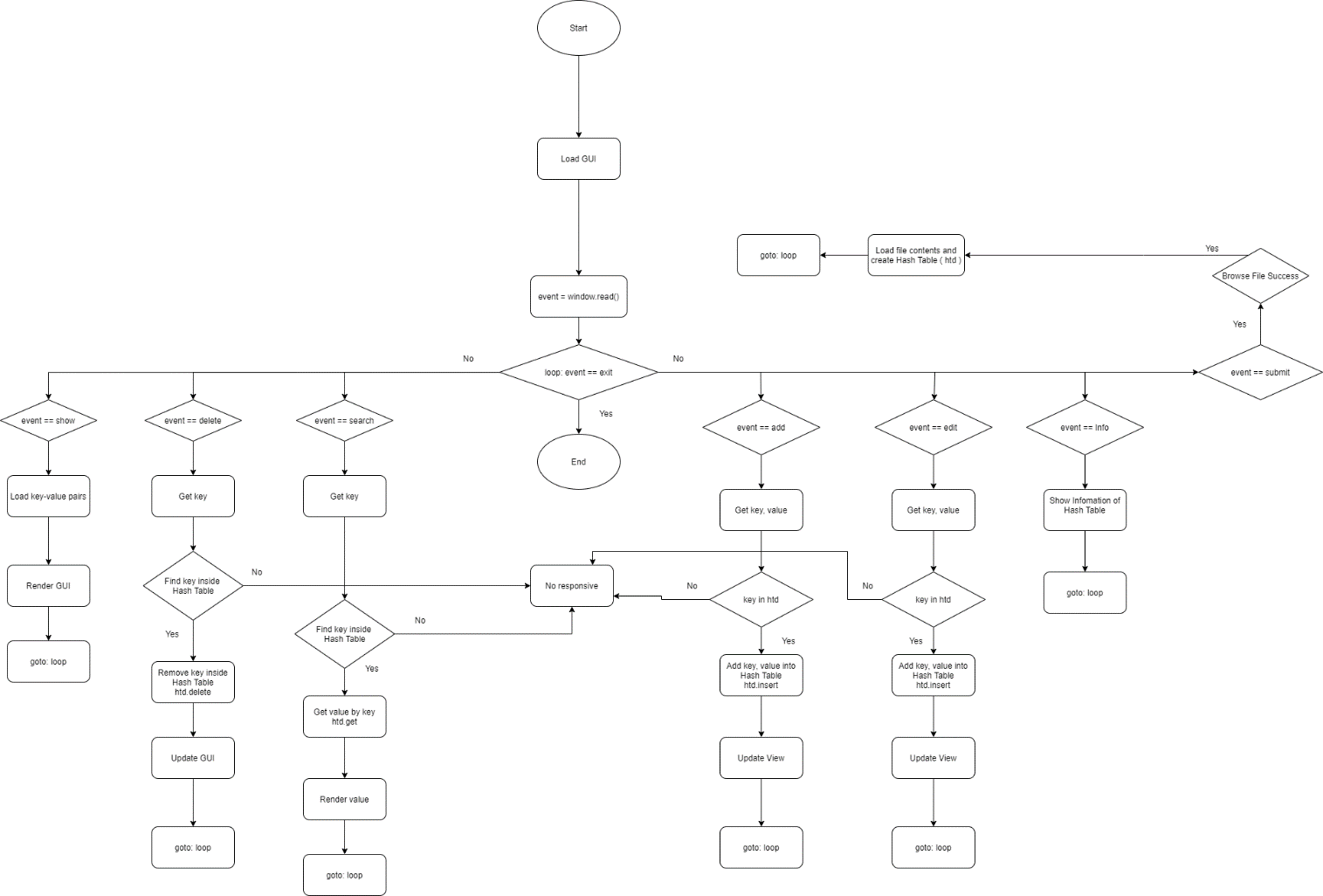
h (1276) = 1276 mod 10 = 6

Giá trị băm thu được là 6

Một nhược điểm của Division Method là các khóa liên tiếp ánh xạ tới các giá trị băm liên tiếp trong Hash Tables và điều này sẽ dẫn đến một hiệu suất kém ccho bài toán

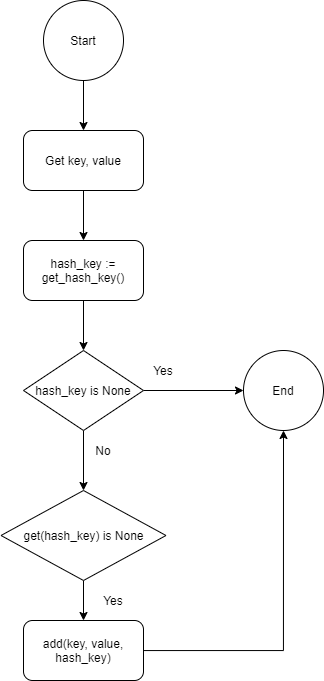
1.2 Sơ đồ giải thuật

1.2.1 Sơ đồ giải thuật tổng quát



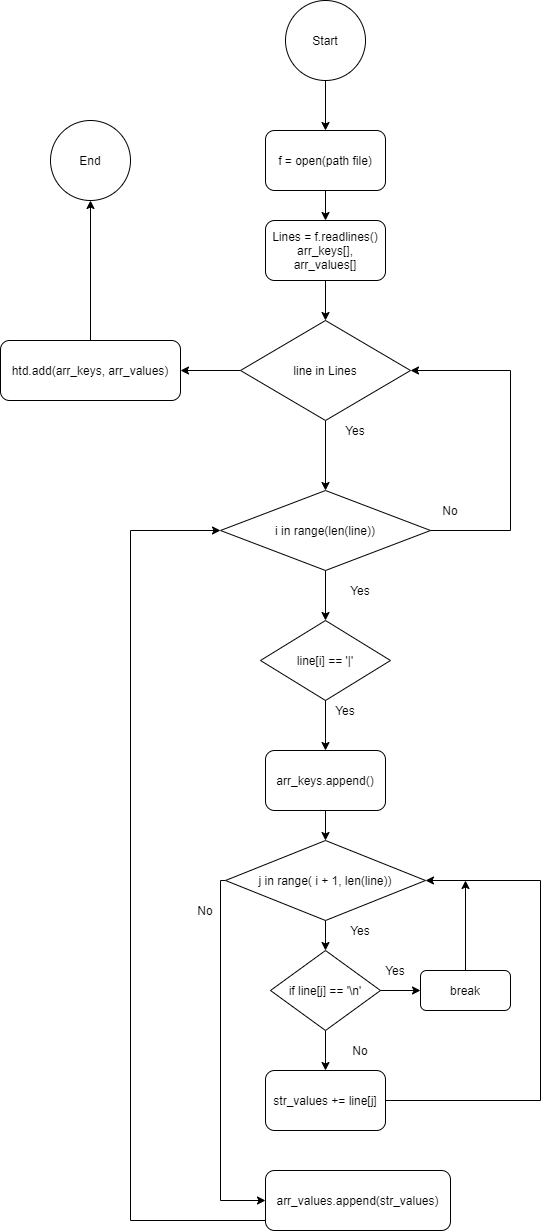
Hình . Sơ đồ giải thuật tổng quát

1.2.2 Sơ đồ giải thuật chức năng Add/ Edit



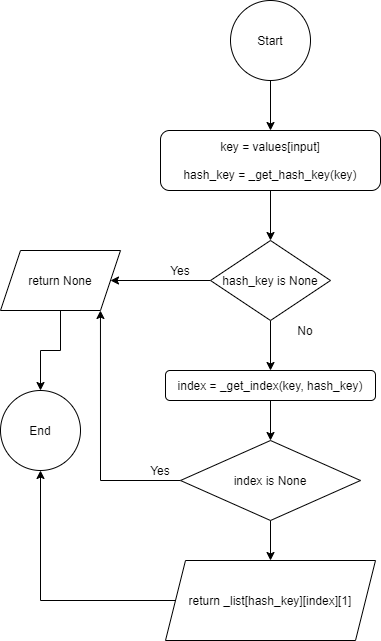
Hình . Sơ đồ giải thuật chức năng Add/ Edit

1.2.3 Sơ đồ giải thuật chức năng Uploads và Create từ điển



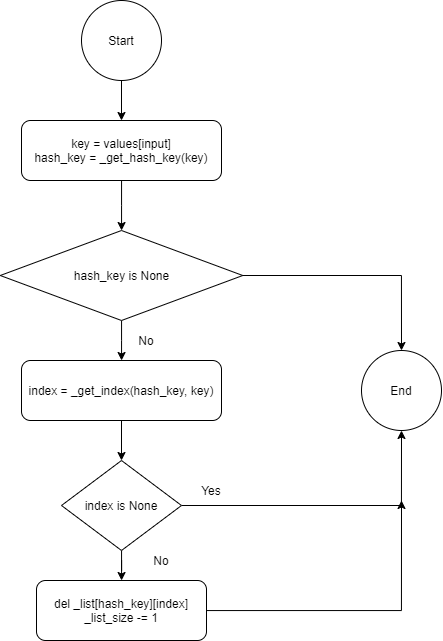
Hình . Sơ đồ giải thuật chức năng Uploads và Create từ điển

1.2.4 Sơ đồ giải thuật chức năng Search



Hình . Sơ đồ giải thuật chức năng Search

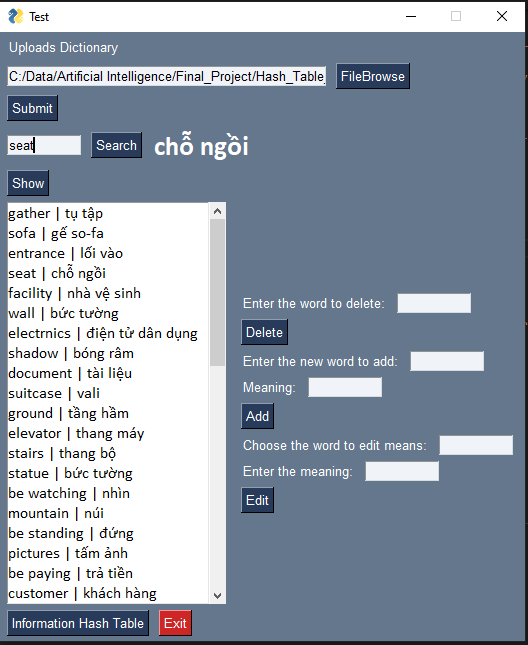
1.2.4 Sơ đồ giải thuật chức năng Delete



Hình . Sơ đồ giải thuật chức năng Delete

1.3 Hiện thực

1.3.1 Giao diện và các chức năng của từ điển



Hình . Giao diện chính của ứng dụng từ điển

Trên đây là giao diện và các chức năng của chương trình. Giao diện này được xây dựng dựa trên thư viện PySimpleGUI, một thư viện đơn giản dùng để xử lý các giao diện người dùng chạy trên Python3. Chức năng đầu tiên là uploads file từ điển theo cấu trúc file chứa nhiều dòng (tối thiểu 50 dòng) và mỗi dùng có cấu trúc “tiếng Anh | nghĩa tiếng Việt”. Chức năng này tương ứng với nút FileBrowse để chọn file cần upload, sau đó nhấn Submit để thực hiện việc uploads file vừa chọn.

Sau khi đã upload file từ điển, người dùng có thể nhấn nút Show để hiển thị nội dung toàn bộ file từ điển đã được upload. Chức năng Show này có thể cập nhật nội dung từ điển sau các tác vụ như thêm, xóa, sửa của người dùng và hiển thị lại chính xác nội dung từ điển khi mới cập nhật. Ngoài ra, khi người dùng uploads lần lượt nhiều file, từ điển sẽ hiển thị lần lượt tất cả nội dung của các file từ điển đã upload.

Chức năng Search giúp người dùng tìm kiếm nghĩa của các từ có trong từ điển. Khi người dùng nhập một từ tiếng Anh và nhấn nút Search, nghĩa tiếng Việt của từ sẽ hiển thị ra như ở hình trên. Trong trường hợp người dùng tìm kiếm một từ không có trong từ điển thì kết quả sẽ hiển thị như sau:



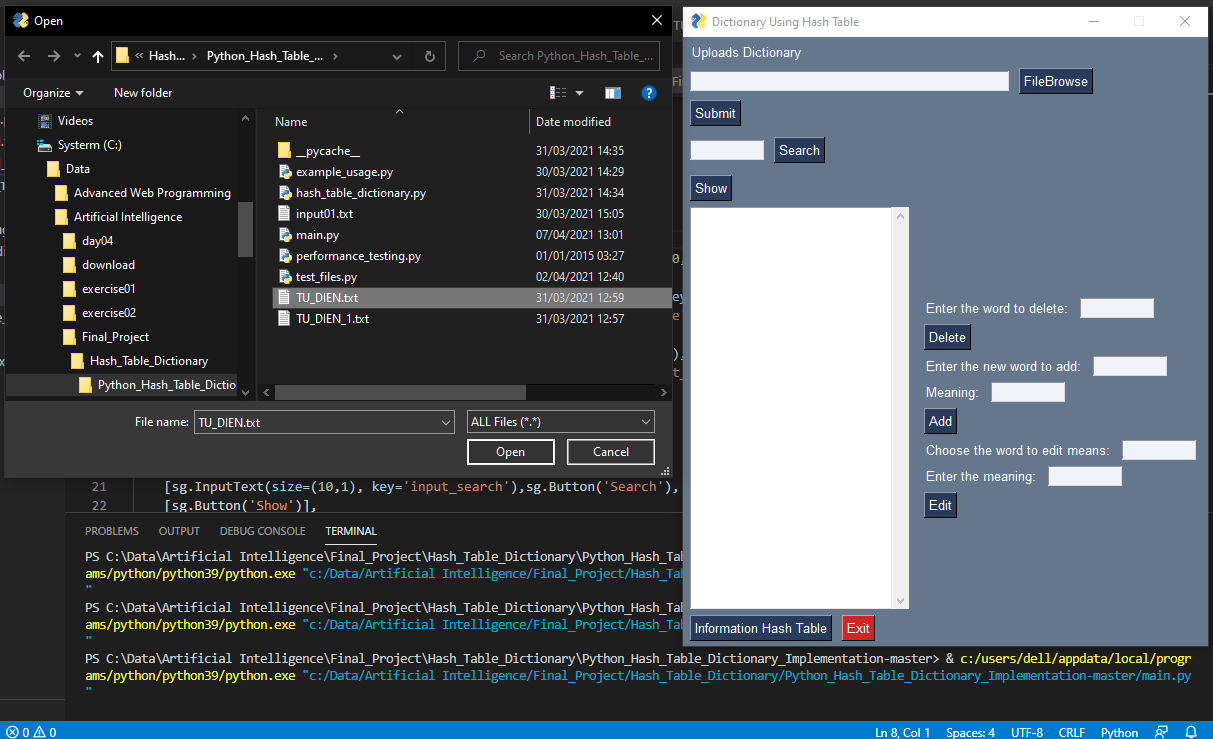
Hình . Trường hợp người dùng tìm kiếm một từ không có trong từ điển

Từ điển có thể thực hiện nhóm chức năng CRUD (thêm, xóa, sửa) tương ứng với các nút Add, Delete và Edit. Khi người dùng cần thêm một từ mới vào từ điển, họ chỉ cần nhập từ mới đó kèm với nghĩa tiếng Việt và nhấn nút Add, khi đó từ điển sẽ tự động thêm từ mới và hiển thị ra bên cạnh. Từ mới sau khi được thêm vào từ điển sẽ có thể thực hiện bất kỳ chức năng hiện có của từ điển mà không cần phải khởi động lại chương trình. Cũng tương tự như chức năng Add, khi người dùng muốn chỉnh sửa thông tin của bất kỳ từ nào trong từ điển, họ sẽ nhập từ đó và nghĩa của từ mà họ muốn thay đổi và nhấn Edit. Đối với chức năng Delete, người dùng cần nhập từ mà họ cần xóa trước khi nhấn nút Delete, sau đó từ đó sẽ được xóa ra khỏi từ điển và người dùng sẽ không thể tìm kiếm từ vừa xóa đó nữa.

Ngoài ra, chương trình còn có nút Information Hash Tables dùng để hiển thị thông tin của Hash Tables như kích thước của Hash Table, phần tử dài nhất trong Hash Tables và nút Exit màu đỏ để thoát chương trình. Sau khi người dùng thoát chương trình, khi họ khởi động lại thì toàn bộ nội dung từ điển đã được uploads trước đó sẽ mất đi, người dùng sẽ cần phải uploads lại các file từ điển cần thiết cho mỗi lần sử dụng chương trình.

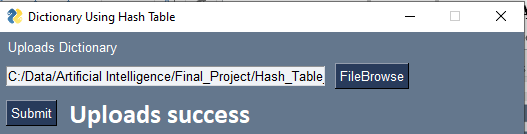
1.3.2 Hiện thực chức năng Upload Files

Khi người dùng nhấn chọn nút FileBrowse, giao diện cho phép người dùng chọn file như sau:



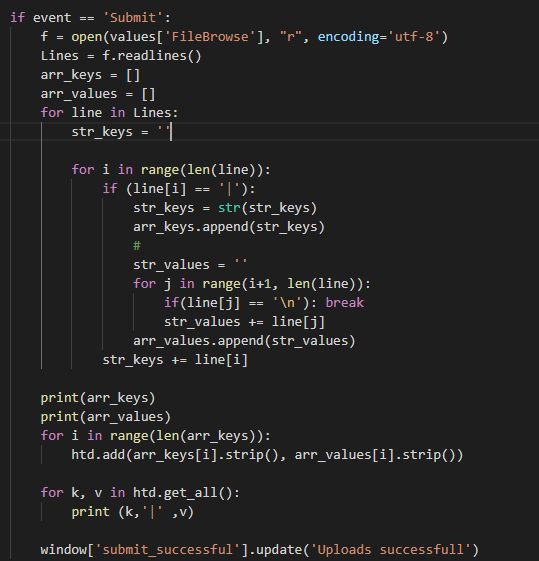
Hình . Giao diện cho phép người dùng chọn file

Sau khi chọn file có đuôi .txt và nhấn nút open, một đường dẫn (path) trực tiếp của file .txt đó được hiển thị ở ô trống kế bên nút FileBrowse. Để upload file từ điển, người dùng cần nhấn chọn nút Submit và chương trình sẽ hiển thị một thông báo đến người dùng khi upload từ điển thành công.



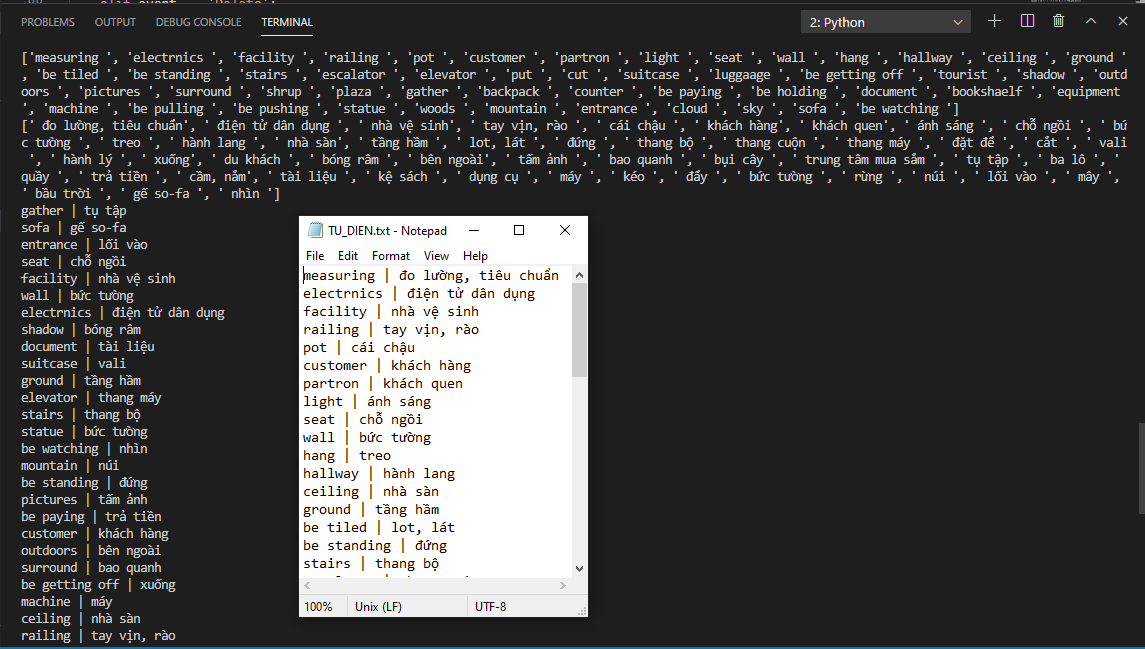
Hình . Thông báo đến người dùng khi upload từ điển thành công.

Đầu tiên, em khởi tạo một object FileBrowse() trong một layout của PySimpleGUI dùng để upload file từ điển, sau đó em bắt sự kiện người dùng nhấn nút ‘Submit’ và xử lý như sau:



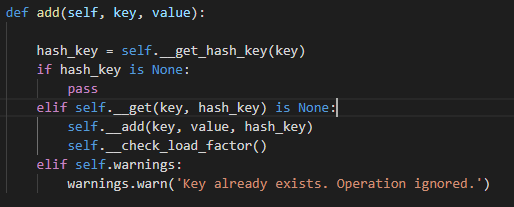
Hình . Sự kiện “Submit”

Khi người dùng nhấn nút Submit, em tiến hành đọc file từ đường dẫn trực tiếp đến file từ điển (values[FileBrowse]) và đọc từng dòng trong file đó. Do nội dung file từ điển tối thiểu 50 dòng, mỗi dòng đều có dạng “tiếng Anh | nghĩa tiếng Việt”, tức là từ tiếng anh sẽ cách nghĩa tiếng Việt bởi ký tự “|”. Hai vòng lặp for ở trên dùng để xét từng dòng trong file từ điển, nếu gặt ký từ “|” thì các ký tự trước nó sẽ là từ tiếng Anh và thêm từ này vào mảng arr\_keys. Khi gặp ký tự “|”, vòng lặp tiếp theo sẽ bắt đầu duyệt tiếp chuỗi đang duyệt từ vị trí sau ký tự “|” để nhận được nghĩa tiếng Việt và thêm nó vàng mảng arr\_values. Kết thúc hai vòng lặp for ở trên, kết quả là ta đã tách được mỗi dòng trong file từ điển được upload thành từ tiếng Anh chứa trong mảng arr\_keys, nghĩa tiếng Việt chứa trong mảng arr\_values. Để có thể quan sát, em hiển thị kết quả vừa tách được dưới terminal như sau:



Hình . Kết quả uploads hiển thị trên terminal

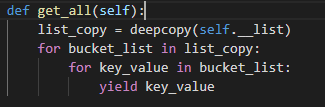
Sau khi đã có hai mảng arr\_keys tương ứng với keys trong Hash Table và mảng arr\_values tương ứng với values trong Hash Table, em sẽ add nó vào Hash Table ở vòng lặp tiếp theo hai vòng lặp lồng nhau ở trên. Đoạn code sau đây thực hiện việc add các keys và values vào trong Hash Table:



Hình . Hàm add

Đầu tiên em sẽ lấy hash key của Hash Table và kiểm tra xem hash key này có phải None không, nếu không thì kiểm tra xem key có tồn tại chưa, nếu đã tồn tại thì hiển thị một cảnh báo đến người dùng và không add cặp key-value đó vào Hash Table, nếu không thì add cặp key-value đó vào Hash Table thành công.

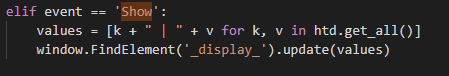
Sau khi đã add tất cả các cặp key-value vào Hash Table, em có tạo một hàm get\_all() trả về các giá trị key, value sau đó lần lượt in ra các giá trị key và value có trong Hash Table:



Hình . Hàm get\_all

1.3.3 Hiện thực chức năng Show

Sau khi đã lấy nội dung trong file từ điển và thêm vào Hash Table các cặp key-value, em hiện thực chức năng Show toàn bộ nội dung từ điển như sau:

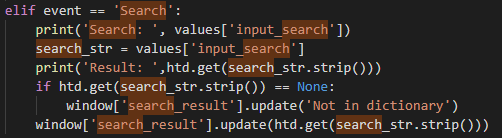


Hình . Sự kiện “Show”

Sự kiện “Show” này dùng một object ListBox() có trong PySimpleGUI để hiển thị từng dòng có trong Hash Table. Sau mỗi sự kiện như thêm, xóa, sửa hoặc sự kiện show, nội dung của từ điển sẽ được cập nhật dựa trên các cặp key-value có trong Hash Table và dùng hàm get\_all() để lấy tất cả giá trị của từng key, value trong Hash Table và hiển thị ra trong ListBox().

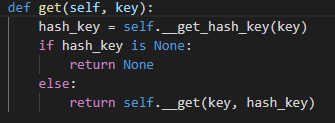
1.3.4 Hiện thực chức năng Search

Chức năng Search này cho phép người dùng tra nghĩa tiếng Việt của một từ tiếng Anh trong từ điển. Dưới đây là đoạn code xử lý sự kiện “Search”:



Hình . Sự kiện “Search”

Khi người dùng nhập một từ tiếng Anh ( tức key trong Hash Table) và nhấn chọn nút Search, chương trình sẽ thực hiện hàm get() trong Hash Table, cụ thể hàm get() như sau:

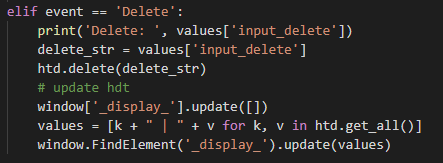


Hình . Hàm get

Hàm này nhận vào chuỗi từ tiếng Anh mà người dùng nhập và so sánh với từng key trong Hash Table, nếu có thì trả về một value của key trùng với từ tiếng Anh và hiển thị giá trị values đó ra cho người dùng, ngược lạ sẽ hiển thị thông báo “Not in dictionary”.

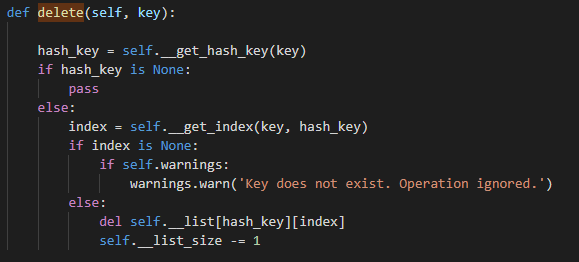
1.3.5 Hiện thực chức năng Delete

Chức năng Delete này cho phép người dùng xóa một từ có trong từ điển. Cụ thể người dùng sẽ nhập một từ tiếng Anh và nhấn nút Delete, chương trình sẽ lập tức xóa từ đó khỏi từ điển và cập nhật lại trong ListBox().



Hình . Sự kiện “Delete”

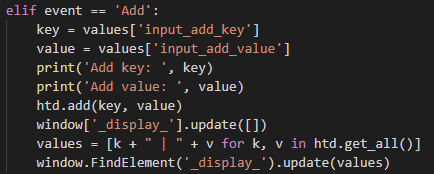
Sau khi bắt sự kiện “Delete”, em sẽ gọi hàm delete để thực hiện việc xóa key-value ra khỏi từ điển:



Hình . Hàm delete

1.3.6 Hiện thực chức năng Add

Người dùng cần nhập vào từ tiếng Anh, sau đó là nghĩa tiếng Việt tương ứng trước khi nhấn chọn nút Add. Sau đó chương trình sẽ thêm cặp key-value tướng ướng với từ tiếng Anh – nghĩa tiếng Việt vào Hash Table, cập nhật lại trong ListBox() để hiển thị ra cho người dùng. Đoạn code bên dưới thực hiện sự kiện “Add”:

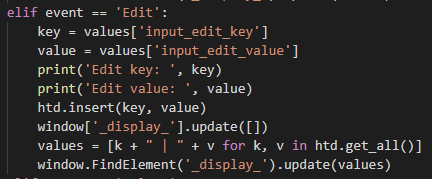


Hình . Sự kiện “Add”

Chức năng Add này về bản chất tương tự với đoạn thêm từ điển đã upload vào Hash Table, cụ thể em dùng key và value là hai biến chứa nội dung người dùng nhập vào tương ứng với từ tiếng Anh, nghĩa tiếng Việt sau đó truyền vào hàm add cặp key-value và update lại nội dung của Hash Table và giao diện ở ListBox().

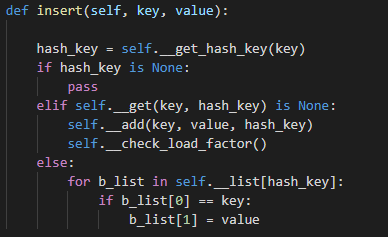
1.3.7 Hiện thực chức năng Edit

Chức năng này cho phép người dùng chỉnh sửa nghĩa tiếng Việt của một từ tiếng Anh có trong từ điển. Cụ thể người dùng sẽ phải nhập vào từ mà người dùng muốn chỉnh sửa, sau đó nhập nghĩa tiếng Việt cần chỉnh sửa và nhấn chọn nút Edit. Đoạn code dưới đây xử lý sự kiện “Edit”:



Hình . Sự kiện “Edit”

Sau khi lấy được từ tiếng Anh, nghĩa tiếng Việt tương ứng với cặp key-value em sẽ truyền vào hàm insert để thực hiện việc chỉnh sửa value của key truyền vào và ghi đè value mới lên value cũ. Hàm insert như sau:

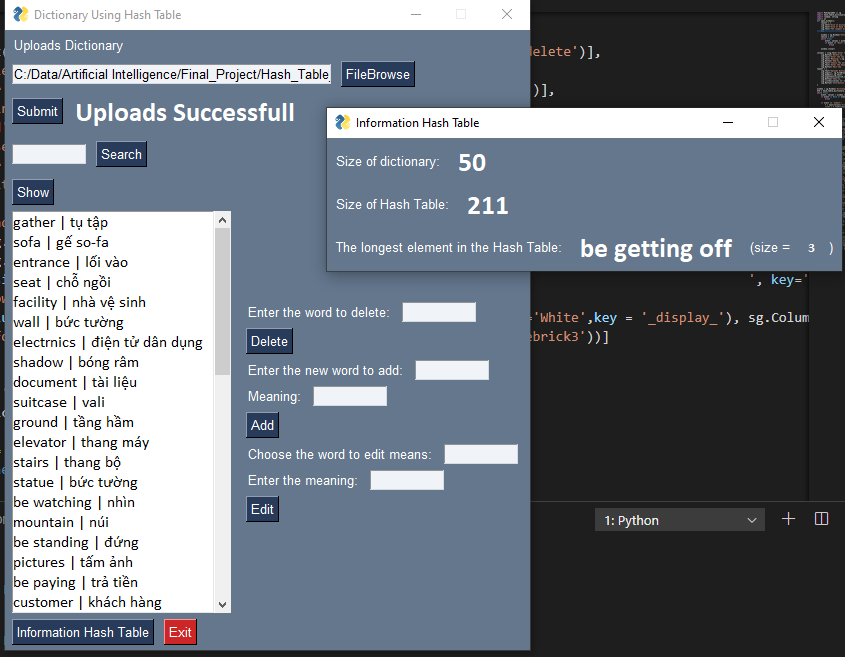


Hình . Hàm insert

Hàm insert nhận vào một cặp key-value, kiểm tra xem nếu key đó tồn tại thì ghi đè value mới vào value cũ.

1.3.8 Hiện thực chức năng Information Hash Table và Exit

Chức năng “Information Hash Table” dựa trên yêu cầu của đề bài, cụ thể yêu cầu đề bài in ra màn hình kích thước của Hash Table cùng với phần tử dài nhất trong Hash Table. Ngoài ra em cũng in thêm kích thước của từ điển trong chức năng này để dễ dàng đến được từ điển đang có bao nhiêu phần tử. Dưới đây là giao diện của cửa sổ “Information Hash Table”:



Hình . Giao diện hiển thị thông tin Hash Tables

Ngoài ra, chức năng “Exit” là chức năng cơ bản trong một ứng dụng, nó giúp người dùng thoát khỏi từ điển khi kết thúc quá trình sử dụng.

1.4 Kết quả và thảo luận

1.4.1 Kết quả đạt được

Từ điển em xây dựng đã đáp ứng được các yêu cầu của đề bài đặt ra. Cụ thể từ điển này sử dụng Hash với Chaining Method để xử lý các đụng độ xảy ra giữa các key trong các cặp key-value và Hash Function được xây dựng dựa trên Division Method.

Từ điển đã xây dựng được các chức năng upload từ điển từ file theo định dạng cho trước, chức năng thêm từ mới cũng như xóa từ đã có trong từ điển. Từ điển còn có chức năng tra cứu nghĩa của một từ có trong từ điển theo yêu cầu của đề bài. Dựa trên yêu cầu đề bài, từ điển còn có thêm chức năng hiển thị thông tin về Hash Table.

Ngoài các chức năng trên, em còn xây dựng thêm chức năng hiển thị toàn bộ nội dung từ điển và chức năng chỉnh sửa một từ cùng với giao diện từ điển dễ tiếp cận với người dùng.

Từ quá trình nghiên cứu và thực hành xây dựng từ điển này, em đã có thêm các kiến thức rất hữu ích về Hash Table, đặc biệt là sử dụng Chaining Method để xử lý các đụng độ trong quá trình xây dựng Hash Table và Division Method để tạo một Hash Function đơn giản. Trong quá trình xây dựng các chức năng của từ điển theo yêu cầu đề bài, em đã có thêm nhiều kiến thức và kinh nghiệm khi sử dụng cấu trúc dữ liệu Hash Table và cách thao tác, truy cập đến các cặp key-value trong Hash Table.

1.4.2 Đánh giá giải thuật

Đối với một số cách sử dụng Hash Table, ta không thể tạo trước chúng có kích thước "đúng", vì không biết có bao nhiêu phần tử sẽ cần phải được giữ đồng thời trong suốt vòng đời của bảng. Trong từ điển mà chúng em xây dựng, em muốn giữ quyền truy cập nhanh do đó cần thay đổi kích thước bảng theo thời gian khi số lượng phần tử tăng lên. Việc thay đổi kích thước này mất thời gian tỷ lệ thuận với số lượng phần tử đã có trong bảng và thường được thực hiện khi chèn, khi các phần tử số vượt qua ngưỡng.

Các thao tác thay đổi kích thước này có thể được thực hiện đủ khi chi phí chèn khấu hao vẫn không đổi (bằng cách làm theo tiến trình hình học cho kích thước của bảng, ví dụ như nhân đôi kích thước mỗi lần thay đổi kích thước). Nhưng một lần chèn theo thời gian sẽ mất O(n) thời gian vì nó kích hoạt thay đổi kích thước. Hay nói cách khác, chèn một giá trị vào Hash Table mất O(1) time trong trường hợp trung bình. Hàm băm được tính toán bởi hàm được chọn từ bảng băm và sau đó là mục được chèn. Trong trường hợp xấu nhất, tất cả các phần tử sẽ được băm đến cùng một giá trị, có nghĩa là toàn bộ danh sách buckets phải đi qua hoặc trong trường hợp open addresss, toàn bộ bảng phải được thăm dò cho đến khi tìm thấy một điểm trống. Do đó, trong trường hợp xấu nhất, việc chèn sẽ mất O(n) time.

1.4.3 Thảo luận về kết quả và giải thuật đã hiện thực

Từ điển mà chúng em xây dựng áp dụng cấu trúc dữ liệu Hash Table, và đã áp dụng thành công Chaining Method để giải quyết các đụng độ cùng với Division Method để xây dựng Hash function theo như yêu cầu của đề bài. Đối với giải thuật Chaining Method, đây là một phương pháp khá đơn giản như cực kỳ mạnh mẽ và hữu ích trong việc giải quyết các đụng độ trong quá trình xây dựng Hash Table. Trong khi đó, việc xây dựng Hash Table sẽ cần một Hash function, và Division Method là một phương pháp đơn giản trong việc xây dựng một Hash function để xây dựng nên cấu trúc dữ liệu Hash Table sử dụng trong chương trình từ điển, từ đó dễ dàng truy xuất đến các cặp key-value có trong từ điển.

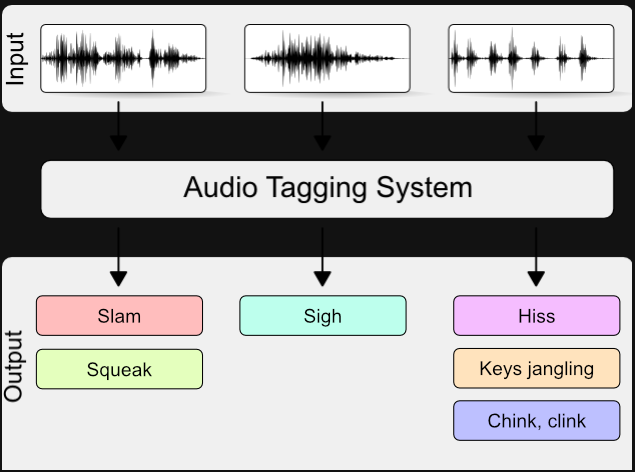
Ứng dụng từ điển mà chúng em thực hiện này có các chức năng như uploads file, thêm từ mới, xóa từ cũ, tìm kiếm từ có trong từ điển và in ra được thông tin về Hash Table. Ngoài các chức năng trên, chúng em đã hiện thực được thêm các chức năng khác như edit một từ mới, hiển thị tất cả các từ đang có trong từ điển và cập nhật nội dung từ điển sau mỗi thao tác. Kết quả thu được sau quá trình xây dựng ứng dụng từ điển này là kiến thức cũng như khả năng sử dụng Hash Table trong các ứng dụng thực tế dựa trên các phương pháp đã áp dụng như Division Method, Chaining Method và các thao tác tạo giao diện với thư viện PySimpleGUI. Ứng dụng này mang tính thực tế, có thể ứng dụng trong cuộc sống như triển khai một từ điển nhỏ cho học sinh, sinh viên. Phương hướng phát triển tiếp theo là hoàn thiện hơn các chức năng CRUD, xử lý tất cả các trường hợp phát sinh ngoài mong đợi và hiển thị thông báo cho người dùng. Ngoài ra, chương trình từ điển có thể sẽ được Optimize nếu làm việc với một dữ từ điển lớn.

CHƯƠNG 2

**FREESOUND AUDIO TAGGING 2019**

2.1 Giới thiệu về bài toán

Freesound và Google’s Machine Perception đã tổ chức một cuộc thi gắn thẻ âm thanh thách thức các Kagglers xây dựng một hệ thống gắn thẻ tự động có mục đích chung. Trong cuộc thi Freesound Audio Tagging này, họ đã đưa thử thách lên cấp độ tiếp theo với tính năng gắn thẻ âm thanh cho nhiều nhãn, số lượng danh mục âm thanh tăng gấp đôi và bộ đào tạo nhiễu nhiều hơn. Ý nghĩa của cuộc thi này giúp cho người chơi nâng cao khả năng lập trình và kiến thức cũng như ứng dụng trong các mô hình Học máy. Dưới đây là hình minh họa cho bài toán Freesound Audio Tagging:



Hình . Minh họa cho bài toán Freesound Audio Tagging

2.2 Phân tích yêu cầu của bài toán

2.2.1 Yêu cầu của bài toán

Một số thông tin cơ bản của bài toán bao gồm: Một số âm thanh khác biệt và có thể nhận ra ngay lập tức, chẳng hạn như tiếng cười của trẻ nhỏ hoặc tiếng gảy đàn guitar. Các âm thanh khác rất khó xác định. Ví dụ nếu nhắm mắt lại, rất khó để có thể phân biệt được đâu là tiếng cưa máy và tiếng máy xay.

Vấn đề đặt ra ở đây là có quá nhiều âm thanh khó phân biệt như thế nhưng lại không có hệ thống gắn thẻ âm thanh với mục đích chung tự động phân biệt các âm thanh đó với độ tin cậy cao tồn tại. Để làm được việc này, con người cần một lượng lớn nỗ lực thủ công dành cho các công việc như chú thích bộ sưu tập âm thanh và cung cấp phụ đề cho các sự kiện không phải lời nói trong nội dung nghe nhìn.

Để giải quyết vấn đề này, Freesound (một sáng kiến ​​của MTG-UPF duy trì cơ sở dữ liệu cộng tác với hơn 400.000 âm thanh được Creative Commons cấp phép) và Nhóm Google Research’s Machine Perception xủa Google Research (những người tạo ra AudioSet, một tập dữ liệu quy mô lớn về các sự kiện âm thanh được chú thích thủ công với hơn 500 các lớp) đã hợp tác để phát triển tập dữ liệu cho cuộc thi mới này.

Để giành chiến thắng trong cuộc thi này, Kagglers sẽ phát triển một thuật toán để gắn thẻ dữ liệu âm thanh tự động bằng cách sử dụng từ vựng đa dạng gồm 80 danh mục. Nếu thành công, hệ thống của bạn có thể được sử dụng cho một số ứng dụng, từ việc gắn nhãn tự động cho các bộ sưu tập âm thanh đến việc phát triển các hệ thống tự động gắn thẻ nội dung video hoặc nhận dạng các sự kiện âm thanh xảy ra trong thời gian thực.

2.2.2 Các phương pháp giải quyết bài toán

2.2.2.1 Phương pháp “Audio Tagging with Noisy Labels and Minimal Supervision”

Phương pháp phổ biến nhất để tạo chương trình gắn thẻ cho các sự kiện của âm thanh là dựa vào phương pháp Supervised Learning (việc học có giám sát) thông qua tập dữ liệu âm thanh được gắn nhãn. Một trong các phương pháp áp dụng Supervised Learning trong cuộc thi Freesound Audio Tagging 2019 là phương pháp được trình bày trong bài báo “AUDIO TAGGING WITH NOISY LABELS AND MINIMAL SUPERVISION” của nhóm tác giả Eduardo Fonseca1, Manoj Plakal, Frederic Font , Daniel P. W. Ellis , Xavier Serra và được xuất bản tháng 10 năm 2019.

Cách mà phương pháp này tiếp cận bài toán cũng rất thú vị, cụ thể họ chỉ ra rằng các tập dữ liệu mới được phát hành có xu hướng ngày càng tăng kích thước để cho phép khai thác các phương pháp tiếp cận theo hướng dữ liệu như Deep Learning. Tuy nhiên, việc ghi nhãn thủ công cho các bộ dữ liệu lớn là tốn kém và người ta sẽ hạn chế thực hiện các công việc thủ công này. Do đó, người sáng tạo thường buộc phải quy định trước kích thước tập dữ liệu và chất lượng nhãn. Hầu hết các bộ dữ liệu gần đây có kích thước lớn nhưng việc ghi nhãn của chúng lại kém chính xác hơn so với các bộ dữ liệu nhỏ và được dán nhãn đầy đủ thông thường. Do đó, chúng ta đang đứng trước ​​sự chuyển đổi sang các bộ dữ liệu lớn hơn mà chắc chắn sẽ bao gồm một số mức độ nhiễu của các nhãn. Tương tự như vậy, xu hướng hiện nay đang hướng tới các thiết bị nhận dạng sự kiện âm thanh có mục đích chung, có thể nhận ra nhiều loại âm thanh hàng ngày. Phổ biến hiện nay có thể kể đến AudioSet Ontology, một cây phân cấp (hierarchical tree) với 627 lớp bao gồm những âm thanh phổ biến nhất hàng ngày.

Phương pháp mà em trình bày ở đây không chỉ quan tâm đến việc nhận ra các nguồn âm thanh điển hình (ví dụ: Vỏ cây) mà còn cả các lớp âm thanh ít thông thường Hơn nữa, những lớp này không chỉ ít thường xuyên hơn mà còn một số lớp có thể tương tự về mặt ngữ nghĩa hoặc âm học với những lớp khác Việc chú thích thủ công các danh mục mơ hồ hơn này trở nên khó khăn hơn, điều này làm cho chúng dễ mắc lỗi ghi nhãn hơn. Một giải pháp thay thế cho việc thu thập dữ liệu để đào tạo các trình gắn thẻ âm thanh có mục đích chung là truy xuất âm thanh và siêu dữ liệu từ các trang web như Freesound hoặc Flickr. Các nhãn có thể được suy ra tự động bằng cách sử dụng phương pháp phỏng đoán tự động được áp dụng cho siêu dữ liệu hoặc áp dụng các bộ phân loại được đào tạo trước trên tài liệu âm thanh. Cách tiếp cận này hỗ trợ thu thập nhanh chóng lượng lớn dữ liệu, nhưng với chi phí là một lượng nhiễu nhãn đáng kể. Trong bối cảnh này, nhiễu nhãn phát sinh như một thách thức trong việc nhận dạng sự kiện âm thanh đa năng, bao gồm các tác động bất lợi như giảm hiệu suất hoặc tăng độ phức tạp của các mô hình, và cũng cản trở việc học đúng các mạng sâu. Do đó, đối phó với tiếng ồn nhãn có thể mở ra cánh cửa cho các bộ phân loại sự kiện âm thanh tốt hơn và có thể cho phép khai thác lượng lớn âm thanh web để đào tạo, đồng thời giảm nhu cầu chú thích thủ công. Chủ đề học tập với nhãn nhiễu là một lĩnh vực nghiên cứu hợp nhất trong thị giác máy tính. Tuy nhiên, trong nhận dạng âm thanh, nó đã ít được chú ý, có thể là do mô hình thông thường là học từ các tập dữ liệu nhỏ và sạch, chỉ có một số công trình đề cập trực tiếp đến việc phân tích và giảm thiểu nhiễu nhãn.

Trong phương pháp này đề xuất một tập dữ liệu và một hệ thống cơ sở để thúc đẩy nghiên cứu nhiễu nhãn trong gắn thẻ sự kiện âm thanh có mục đích chung. Phương pháp này cũng đề xuất điều tra trường hợp có sẵn một tập hợp nhỏ dữ liệu được gắn nhãn thủ công, cùng với một tập hợp dữ liệu được gắn nhãn nhiễu lớn hơn, trong cài đặt gắn thẻ âm thanh nhiều nhãn và Sử dụng vốn từ vựng gồm 80 lớp âm thanh hàng ngày.

Các đề xuất trên của phương pháp giải quyết hai vấn đề nghiên cứu chính. Vấn đề đầu tiên là làm thế nào để khai thác một lượng lớn các nhãn nhiễu, nhiều trong số đó không chính xác và / hoặc không đầy đủ, và làm thế nào để bổ sung nó với sự giám sát được cung cấp bởi một lượng nhỏ hơn nhiều dữ liệu được dán nhãn thủ công đáng tin cậy (giám sát tối thiểu). Vấn đề thứ hai được đưa ra bởi sự không phù hợp về âm thanh giữa bộ tàu ồn ào và bộ thử nghiệm. Sự thay đổi phân phối giữa dữ liệu đã được chứng minh là gây ra sự sụt giảm hiệu suất đáng kể trong học máy, cho cả thị giác và âm thanh. Trong trường hợp tập hợp tiếng ồn đến từ một nguồn âm thanh web khác với tập hợp thử nghiệm, đôi khi là một hạn chế trong thế giới thực.

Kết quả đạt được của phương pháp này là đã mô tả thiết lập tác vụ, tập dữ liệu và hệ thống cơ sở của DCASE2019 trong cuộc thi Freesound Audio Tagging 2019. Bộ dữ liệu FSDKaggle2019 và hệ thống cơ sở được đề xuất có sẵn miễn phí và không giới hạn để sử dụng trong cuộc thi. Phương pháp này được lưu trữ trên nền tảng Kaggle với tên gọi “Freesound Audio Tagging 2019”. Cùng với kết quả đó, phương pháp này đã có thể khai thác một cách thỏa đáng một tập hợp lớn dữ liệu được gắn nhãn nhiễu và một lượng nhỏ dữ liệu được gắn nhãn thủ công, trong cài đặt gắn thẻ âm thanh đa nhãn với vốn từ vựng gồm 80 lớp âm thanh hàng ngày. Ngoài ra, bộ dữ liệu đặt ra vấn đề không khớp về âm thanh giữa bộ tiếng ồn ào và bộ thử nghiệm do chúng đến từ các nguồn âm thanh của các trang web khác nhau.

Bản thân em nhận xét rằng phương pháp này có thể áp dụng vào thực tế do giải quyết được các khó khăn trong việc thu thập một lượng lớn dữ liệu được dán nhãn thủ công. Ngoài ra, em cũng đề xuất sử dụng thước đo độ chính xác trung bình xếp hạng nhãn theo trọng số nhãn (viết tắt là Lωlrap) để làm thước đo đánh giá độ chính xác cho mỗi nhãn trong tập kiểm tra. Các thực nghiệm cơ bản chỉ ra rằng việc tận dụng dữ liệu được gắn nhãn nhiễu với sự thay đổi phân phối để gắn thẻ sự kiện âm thanh có thể là một thách thức lớn.

2.2.2.2 Phương pháp “Convolutional Recurrent Neural Network and Data Augmentation for Audio Tagging with Noisy Labels and Minimal Supervision”

Phương pháp tiếp theo em đề cập sau đây được trình bày trong bài báo “CONVOLUTIONAL RECURRENT NEURAL NETWORK AND DATA AUGMENTATION FOR AUDIO TAGGING WITH NOISY LABELS AND MINIMAL SUPERVISION” của tác giả Janek Ebbers và Reinhold Haeb-Umbach ra mắt trong năm 2019.

Để tiếp cận với bài toán, họ nhận định rằng trong nhận dạng âm thanh ở môi trường thực tế, các nhiệm vụ của Phát hiện sự kiện âm thanh (SED), Gắn thẻ âm thanh và Phân loại cảnh âm thanh (ASC) có thể được phân biệt, khác nhau về mức độ chi tiết thu được về môi trường âm thanh. Trong khi SED đưa ra dự đoán ở cấp khung hình, Gắn thẻ âm thanh và ASC đưa ra dự đoán ở cấp trình tự. Vì các chú thích cấp khung (được gọi là nhãn mạnh) rất khó lấy và tốn nhiều thời gian, các tập dữ liệu quy mô lớn hiện tại, chẳng hạn như AudioSet của Google, chỉ cung cấp các nhãn cấp trình tự (nhãn yếu). Hơn nữa, đối với nhiều ứng dụng, dự đoán mức khung không được yêu cầu, cùng với sự sẵn có của bộ dữ liệu quy mô lớn được gắn nhãn yếu, dẫn đến việc Gắn thẻ âm thanh ngày càng phổ biến. Mặc dù chú thích yếu dễ lấy hơn chú thích mạnh, nhưng chúng vẫn yêu cầu con người chú thích.

Để tránh sự phức tạp trong việc chú thích các nhãn âm thanh này, một dòng nghiên cứu được dành để phát triển các phương pháp tiếp cận bán giám sát trực tiếp lấy dữ liệu từ tập dữ liệu không được gắn nhãn vốn đã rất dồi dào. Tuy nhiên, cũng có một lượng lớn dữ liệu đi kèm với các phương thức và siêu dữ liệu khác, có thể được khai thác để lấy nhãn tự động. Tác giả của bài báo này cũng đã khẳng định trong tập dữ liệu được cung cấp bởi cuộc thi, các nhãn đã được tạo từ dữ liệu web bằng cách sử dụng dự đoán cấp video. Họ lưu ý rằng các nhãn được tạo tự động đó chắc chắn có lỗi, đó là lý do tại sao chúng được gọi là nhãn nhiễu.

Trong phương pháp này, mục tiêu nghiên cứu chính là trả lời câu hỏi: “Làm thế nào để có thể đào tạo một hệ thống Gắn thẻ âm thanh hiệu suất cao cung cấp tương đối ít dữ liệu với các nhãn đáng tin cậy nhưng lượng dữ liệu không khớp lớn hơn và bị nhiễu Nhãn mác?”. Tác giả của phương pháp chủ yếu giải quyết vấn đề này bằng cách khám phá các phương pháp khác nhau để tăng dữ liệu và dành ít thời gian hơn cho việc điều chỉnh kiến ​​trúc mạng thần kinh. Đầu tiên họ phác thảo kiến ​​trúc mạng nơ-ron nhân tạo, sau đó là áp dụng các phương pháp tăng dữ liệu và quy trình đào tạo, thử nghiệm (train,test) trên tập dữ liệu của đã được xử lý. Họ cho rằng sự chênh lệch tần số và thời gian và mặt nạ tần số để tăng dữ liệu cải thiện đáng kể hiệu suất.

Các kết quả thực nghiệm của phương pháp này có xu hướng cho thấy rằng mô hình của họ mạnh mẽ chống lại nhiễu nhãn, vì nó đạt được hiệu suất hiện đại mà không cần bất kỳ xử lý cụ thể nào về nhiễu nhãn. Mặc dù các phương pháp này được nghiên cứu để xử lý nhiễu nhãn riêng lẻ không dẫn đến cải tiến bộ phân loại, nhưng chúng vẫn làm tăng tính đa dạng của các mô hình được đào tạo, dẫn đến cải thiện hiệu suất thông qua kết hợp hệ thống. Hệ thống của họ được cung cấp công khai trên github với đường dẫn: <https://github.com/fgnt/upb_audio_tagging_2019>.

Các kết quả thực nghiệm của phương pháp này được thực hiện trên dữ liệu “FSDKaggle2019”, từ đó làm nổi bật tầm quan trọng của các kỹ thuật xử lý dữ liệu để đạt được hiệu suất phân loại cao. Tuy điểm mạnh là như thế, điểm yếu của phương pháp này nằm ở việc xét cụ thể về từng tiếng ồn nhãn không tỏ ra hiệu quả. Hơn nữa, hiệu suất của mô hình mạng nơ-ron của họ sẽ được tăng cường nhờ sự kết hợp hệ thống nâng hiệu suất từ ​​0,724 lwlrap của hệ thống mô hình đơn tốt nhất của họ lên 0,765 bằng cách kết hợp tổng số 27 mô hình, thiết lập một trạng thái mới cho mạng nơ-ron nhân tạo này.

2.2.2.3 Phương pháp “Audio Tagging using Linear Noise Modelling Layer”

Một phương pháp khác đơn giản hơn hai phương pháp trên nhưng cũng rất thú vị là phương pháp được trình bài trong bài báo “AUDIO TAGGING USING A LINEAR NOISE MODELLING LAYER” của nhóm tác giả Shubhr Singh, Arjun Pankajakshan và Emmanouil Benetos trình bày trong năm 2019.

Cách tiếp cận và giải quyết bài toán của phương pháp là dựa trên đào tạo có giám sát của mạng nơ-ron sâu (deep neural network - DNN) trên các nhãn được liên kết với mỗi bản ghi âm. Họ giả định cơ bản đi kèm với các cách tiếp cận này là nhãn được cung cấp cùng với bản ghi âm là chính xác, tức là sự hiện diện của sự kiện âm thanh được liên kết với nhãn tương ứng với bản ghi âm đã được xác minh theo cách thủ công. Tuy nhiên giả định này không phải lúc nào cũng đúng vì xác minh thủ công nhãn dữ liệu là một việc tốn kém, hạn chế hiệu quả kích thước của tập dữ liệu. Việc họ tích lũy một lượng lớn dữ liệu âm thanh sẽ phải trả giá bằng việc các nhãn không chính xác, đặc biệt là trong các trường hợp khi các nhãn được suy ra dựa trên siêu dữ liệu do người dùng cung cấp, cụ thể là dữ liệu được cung cấp từ. Khi các nhãn bị nhiễu, có khả năng cao thông tin gây hiểu nhầm, điều này sẽ phá hủy quá trình đào tạo DNN. Các nghiên cứu gần đây đã chỉ ra rằng khả năng tổng quát hóa của một DNN giảm trên các tập dữ liệu có nhãn nhiễu, tức là mô hình trang bị quá nhiều dữ liệu huấn luyện.

Trong bài báo trình bày về phương pháp này, họ khai phá tập dữ liệu FSDnoisy18k cho âm thanh. Các phân loại sự kiện, chứa một tập hợp con nhỏ (10%) dữ liệu được gắn nhãn chính xác và tập hợp con lớn (90%) bao gồm các mẫu dữ liệu có nhãn nhiễu. Họ sử dụng phương pháp tiếp cận lớp mô hình phân bố nhiễu tuyến tính và so sánh hiệu suất của nó trên tập thử nghiệm với hiệu suất của mô hình cơ sở và cả với phương pháp tiếp cận hàm mất mát khởi động mềm. Cả hai cách tiếp cận đã được thực hiện cho vấn đề nhãn nhiễu trong thị giác máy tính, tuy nhiên theo hiểu biết của các tác giả, các thử nghiệm về dữ liệu âm thanh với lớp mô hình nhiễu tuyến tính vẫn chưa được khám phá. Tác giả áp dụng kiến ​​trúc MobilenetV2 làm mô hình cơ sở mà không có bất kỳ trọng số nào được đào tạo trước và đào tạo mạng trên bộ đào tạo của FSDnoisy18k. Các trọng số mô hình có tổn thất cross entropy phân loại (CCE) ít nhất trên tập xác nhận được chọn và một lớp dày đặc có kích hoạt softmax được đặt trên đầu mạng và được đào tạo lại trên tập huấn luyện. Mục đích của việc đào tạo lại là để tìm hiểu các thông số trọng lượng của lớp mô hình nhiễu. Lớp mô hình nhiễu được loại bỏ trong quá trình dự đoán trên tập thử nghiệm.

Trong bài báo cáo cũng chỉ ra rằng phương pháp tiếp cận lớp mô hình nhiễu cải thiện độ chính xác của mạng đường cơ sở trên bộ thử nghiệm lên khoảng 2% và phương pháp tiếp cận hàm mất mát khởi động mềm cải thiện điểm số chính xác của mạng đường cơ sở lên khoảng 1,5%. Mặc dù độ chính xác của hệ thống thấp hơn so với mục tiêu ban đầu, tác giả của phương pháp này đã đạt được các bước đầu tiên trong việc khám phá mô hình phân bố nhiễu nhãn và em hy vọng rằng họ sẽ đạt được kết quả tốt hơn trong tương lai. Câu hỏi đặt ra ở đây là “Tại sao lớp mô hình nhiễu chỉ có thể khắc phục một số loại nhiễu nhãn và không làm được như vậy đối với các loại nhiễu khác?”. Em hy vọng họ sẽ có những bài đăng về việc triển khai phương pháp tiếp cận trên tập dữ liệu âm thanh nhiễu đa nhãn và đánh giá hiệu suất của mô hình từ các chỉ số đánh giá khác nhau ngoài độ chính xác để hiểu em rõ hơn về các khái niệm mà họ sử dụng.

2.2.3 Phương pháp đề xuất giải quyết bài toán

Qua các phương pháp em vừa trình bày ở trên, ta có thể cơ bản giải quyết bài toán thông qua các bước sau:

- Bước 1, Xử lý tập dữ liệu: Tập dữ đã liệu được cung cấp sẵn, ta cần xử lý nó trước khi đưa vào mạng nơ ron nhân tạo để giảm thiểu độ nhiễu của dữ liệu, loại bỏ các dữ liệu không cần thiết, tránh over-fitting từ đó tạo đặc trưng của tập dữ liệu. Phương pháp em đề xuất có thể áp dụng ở bước này là Data Augmentation sử dụng trong phương pháp thứ 2 em đã trình bày ở trên bởi phương pháp này cực kỳ mạnh mẽ để xử lý tập dữ liệu.

- Bước 2, xây dựng mô hình CNN: Sau khi đã có đặc trưng của tập dữ liệu, ta xây dựng mô hình CNN. Phương pháp xây dựng mô hình CNN trong bài báo thứ nhất là một đề xuất của em để giải quyết vấn đề trong bước này.

- Bước 3, Deep Learning mô hình CNN trên tập dữ liệu đã được xử lý: Sau khi có mô hình CNN và tập dữ liệu đã xử lý, ta sử dụng các phương pháp Supervised Learning để học sâu tập dữ liệu này trên mô hình CNN, kết quả là một model.

- Bước 4, train/test và prediction cho mô hình: Sau khi đã có model ở bước 3, ta tiến hành huấn luyện và kiểm thử với một tập dữ liệu nhỏ hơn tập dữ liệu chính và cho model đó dự đoán và gắn nhãn cho các âm thanh của tập dữ liệu test.

- Bước 5, đánh giá và cải thiện mô hình: Sau khi đã có kết quả gắn nhãn cho các âm thanh ở tập dữ liệu test, ta sẽ tiến hành đánh giá mô hình dự trên các độ đo như accuracy, Fscore, presicon, recall hoặc có thể vẽ confusion matrix để đánh giá. Dựa trên các đánh giá này, ta có thể cải thiện cho mô hình ngày càng tốt hơn.

Trong ba phương pháp em đã trình bày ở trên, có thể thấy phương pháp thứ nhất chú trọng việc xây dựng mô hình mạng nơ-ron nhân tạo và phương pháp thứ ba dùng các Deep Learning method để cải thiện khả năng học của mô hình. Trong khi đó phương pháp thứ hai lại mạnh mẽ trong việc xử lý tập dữ liệu cùng với đó là sự vận dụng, kết hợp giữa mô hình mạng nơ-ron nhân tạo 1D và 2D. Do tính chất tìm hiểu về mô hình mạng nơ-ron nhân tạo của đồ án này và hơn nữa các kết quả cũng chỉ ra rằng phương pháp thứ hai thực sự hiệu quả để giải bài toán Free sound audio tagging 2019 trên Kaggle nên trong phần tiếp theo của bài báo cáo này, chúng em sẽ phân tích cụ thể hơn về phương pháp giải quyết bài toán “Convolutional Recurrent Neural Network and Data Augmentation for Audio Tagging with Noisy Labels and Minimal Supervision”.

2.3 Phương pháp giải quyết bài toán

2.3.1 Mô hình tổng quát

Mô hình tổng quát giải quyết bài toán Free sound audio tagging 2019 trên Kaggle được trình bày theo sơ đồ dưới đây. Trong mô hình này có gồm có 4 phần chính như sau:

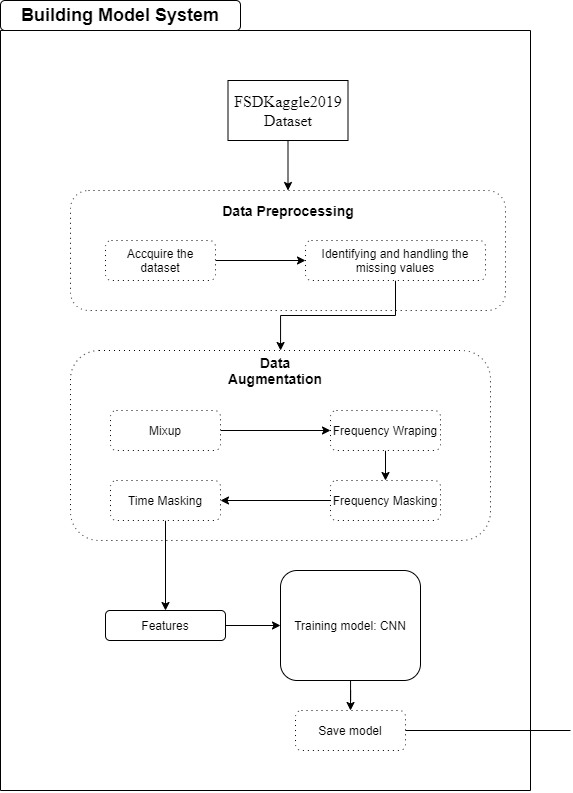
Phần 1: Download FSDKaggle2019 Dataset

Phần 2: Data Preprocessing

Phần 3: Data Augmentation

Phần 4: Features dataset

Phần 5: Training model: CNN



Hình . Mô hình tổng quát của phương pháp

Dataset của bài toán đã được cung cấp sẵn trên trang Kaggle với tên gọi FSDKaggle2019 Dataset, tác giả sẽ thực hiện giai đoạn tiền xử lý dữ liệu trên tập dữ liệu này. Cụ thể, họ nhận tập dữ liệu và xác định cũng như xử lý missing values. Sau khi đã tiền xử lý dữ liệu, họ tiến hành bước tăng dữ liệu và đây là điểm đặc trưng trong cách làm của họ. Sau đó họ trích xuất đặc trưng cho tập dữ liệu và đưa vào mô hình CNN để học. Kết quả đạt được là một mô hình mạng nơ-ron nhân tạo có khả năng gắn nhãn cho các tệp âm thanh.

2.3.2 Đặc trưng của mô hình đề xuất

2.3.2.1 Data Augmentation

Bởi vì chỉ có ít dữ liệu có sẵn nên việc tăng dữ liệu một cách hiệu quả là rất quan trọng để ngăn chặn việc xảy ra hiện tượng over-fitting và cải thiện khả năng tổng quát hóa của hệ thống. Trong phần sau, tác giả phác thảo các phương pháp tăng dữ liệu kết hợp trong quá trình đào tạo mô hình. Tất cả các phương pháp nâng cao được thực hiện nhanh chóng trong quá trình đào tạo, mang lại một số lượng cực kỳ lớn các mẫu đào tạo.

Phương pháp đầu tiên là Mixup, đây là một kỹ thuật tăng dữ liệu bắt nguồn từ các nhiệm vụ phân loại trong đó một mẫu đào tạo mới được tạo ra dưới dạng trung bình có trọng số của hai mẫu từ tập dữ liệu:

= λ + (1 − λ) , λ ∈ [0, 1]

Tương tự như vậy, các mục tiêu được one-hot encoded và được kết hợp với một vectơ mục tiêu:

= λ + (1 − λ) , λ ∈ [0, 1]

Đối với phương pháp Mixup này, việc kết hợp các nhiệm vụ phân loại dẫn đến các mẫu không rõ ràng (có thể cũng không có ý nghĩa gì đối với con người), nhưng dù vậy nó đã cho thấy được sự cải thiện tính tổng quát cùng với tính mạnh mẽ của mạng phân loại được đào tạo. Mixup đã được sử dụng thành công để gắn thẻ âm thanh cho mục đích chung. Đối với gắn thẻ âm thanh (trái ngược với bài toán phân loại), âm thanh đầu vào có thể đã là sự chồng chất của nhiều nguồn. Do đó, trộn hai hoặc nhiều tín hiệu âm thanh với nhau sẽ tạo ra tín hiệu âm thanh hợp lệ mới với số lượng sự kiện hoạt động tăng lên. Do đó, thay vì xây dựng giá trị trung bình có trọng số, chúng ta sử dụng hai dạng sóng như sau:

= (t) + (t- r) ,

với λ ∈ [0, 1],

γ ∼ B(2/3),

t ∼ U(max(− , − 30 s), min( , 30 s − )),

≤ 1.1 \*

= ; ∼ U(1, 2); bm ∼ B(1/2); m∈{0, 1}.

Trong đó B biểu thị phân phối Bernoulli. Để hiể hơn về bản chất của phương trình trên, em sẽ trình bày bằng lời văn như sau:

• Chỉ thực hiện trộn lẫn với xác suất 2/3

• Chỉ các tín hiệu trộn ngắn hơn 1,1 lần tín hiệu cơ sở

• Cho phép trộn lẫn kéo dài tín hiệu miễn là nó không vượt quá độ dài tối đa 30s • Chuẩn hóa tín hiệu đến giá trị lớn nhất của tín hiệu cơ sở

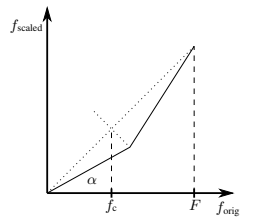
​​• Làm suy giảm hoặc khuếch đại mỗi tín hiệu chuẩn hóa bằng một yếu tố ngẫu nhiên.

Ở đây ta cũng không cần xây dựng mức trung bình có trọng số của các mục tiêu riêng lẻ, mà chỉ cần kết hợp tất cả các thẻ thành một mục tiêu được mã hóa n-hot duy nhất dựa trên .

Phương pháp Data Augmentation tiếp theo là Frequency Warping. Gần đây, SpecAugment đã được giới thiệu như một phương pháp gia tăng đơn giản nhưng hiệu quả của quang phổ log-mel để nhận dạng giọng nói tự động. Nó sử dụng ba biến dạng khác nhau là cong vênh thời gian, che tần số và che dấu thời gian. Tuy nhiên, trong các thử nghiệm của phương pháp “Convolutional Recurrent Neural Network and Data Augmentation for Audio Tagging with Noisy Labels and Minimal Supervisio”, tác giả nhận thấy rằng đối với việc gắn thẻ âm thanh, việc làm cong hình quang phổ trên trục tần số mang lại hiệu suất tốt hơn so với việc làm cong thời gian. Do đó, tác giả thay đổi độ lệch thời gian bằng độ lệch tần số trong phiên bản SpecAugment của họ và được giải thích như phân tích dưới đây.

Đầu tiên, họ coi quang phổ log-mel là một hình ảnh ở đây có chiều rộng T và chiều cao F. Sự cong trục dọc (tần số) của hình ảnh được điều khiển bởi tần số cắt: ∼ (0.5·F). Trong đó biểu thị phân phối hàm mũ được tham số hóa bởi thang đo β = (0,5·F) và bởi hệ số cong vênh ; u ~ (0.07), s ~ β(0.5).

Đồ thị dưới đây biểu diễn cho kết quả của hàm cong vênh tuyến tính từng mảnh. Lưu ý rằng có thể lớn hơn F, điều này dẫn đến việc kéo giãn / nén toàn bộ hình phổ. Điều đáng chú ý là độ vênh tần số được thực hiện ở đây rất giống với Vocal Tract Length Pertubation ( độ dài đoạn giọng nói):



Hình . Đồ thị biểu diễn cho kết quả của hàm cong vênh tuyến tính từng mảnh

Hai phương pháp Data Augmentation tiếp theo là Frequency Masking và Time Masking. Đối với phương pháp Frequency Masking, tác giả randomly mask ngẫu nhiên tần số mel liên tiếp của H trong khoảng [, + H], trong đó H và f0 được lấy từ các phân bố đồng đều:

H ∼ U(0, ),

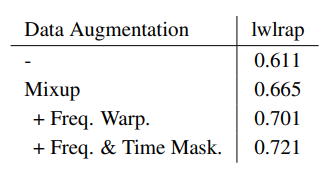
∼ U(0, F − H) với 16.

Trong khi đó, phương pháp Time Masking randomly mask ngẫu nhiên khung thời gian W liên tiếp trong phạm vi [, + W], trong đó W và được rút ra từ các phân phối đồng nhất:

W ∼ U(0, min(, p·N))

∼ U(0, T − W) với = 70 và p = 0.2.

Để đánh giá các phương pháp Data Augmentation khác nhau đã trình bày ở trên, tác giả đã train các hệ thống mô hình đơn lẻ bằng cách sử dụng dữ liệu có nhãn được sắp xếp và nhiễu bằng cách sử dụng các nhãn được cung cấp. Trong phần này, tất cả các mô hình đã được huấn luyện với tỷ lệ nhãn nhiễu là R = 0,56. Hình bên dưới cho thấy hiệu suất tăng do thêm các phương pháp tăng dữ liệu được đề xuất. Có thể thấy rằng mỗi phương pháp tăng đều làm tăng đáng kể hiệu suất của mô hình về mặt lwlrap trên tập thử nghiệm riêng lẻ.

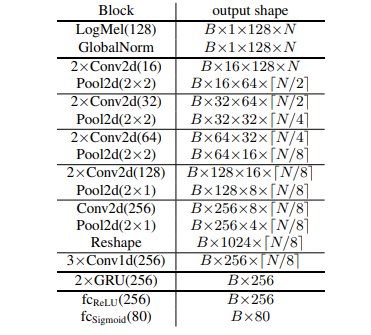


Hình . Hiệu suất khi thêm dần các phương pháp tăng dữ liệu.

2.2.3.2 Xây dựng mô hình mạng nơ-ron nhân tạo

Như đã đề cập trước đó, đầu tiên ta sẽ thực hiện chuyển đổi Audio sang định dạng STFT (Short-time Fourier transform) với độ dài khung hình là 40 ms (1764 mẫu) và kích thước bước nhảy là 20 ms (882 mẫu) trên tín hiệu âm thanh 44,1 kHz được cung cấp mà không cần lấy mẫu lại. Sau đó, đối với mỗi khung, ta trích xuất 128 đặc điểm năng lượng băng tần log-mel với fmin = 50 Hz và fmax = 16 kHz. Cuối cùng, ta lấy giá trị trung bình chung của mỗi đối tượng và sau đó chia cho độ lệch chuẩn chung trên tất cả các đối tượng.

Bước tiếp theo là xây dựng kiến trúc của mạng nơ-ron nhân tạo. Kiến trúc này dựa trên học sâu và được mô tả trong bảng dưới đây:



Hình . Convolutional Recurrent Neural Network cho Audio Tagging với các định dạng đầu ra của mỗi khối. Mỗi ConvXd sử dụng kích thước hạt nhân là 3 và một bước tiến và bao gồm BatchNorm và ReLU.

Mạng nơ-ron bao gồm một mô-đun tích lũy sau mỗi lần lặp lại và một mô-đun được kết nối đầy đủ. Đầu vào bốn chiều cho mô hình có dạng B × C × F × Nm với B, C, F, Nm là kích thước mini-batch, số kênh, số tính năng và số khung hình trong đầu vào thứ m với tín hiệu tương ứng, trong đó C = 1 và F = 128 là cố định. Trong phần sau đây chỉ số tín hiệu m bị bỏ qua.

Mô hình này là sư kết hợp giữa CNN 2D và CNN 1D. CNN 2D bao gồm năm lượt chuyển đổi khối, với mỗi khối bao gồm một hoặc hai chuyển đổi các lớp và một lớp tổng hợp tối đa. Trong khi bốn khối đầu tiên có hai lượt chuyển đổi. Các lớp khối cuối cùng chỉ có một khối duy nhất. Trong mỗi khối, số lượng kênh được tăng gấp đôi bắt đầu từ 16 trong khi số lượng tính năng giảm đi một nửa bằng cách tổng hợp tối đa. Số bước thời gian cũng được giảm một nửa trong ba khối đầu tiên trong khi không thay đổi trong hai khối cuối cùng. Điều này dẫn đến kết quả đầu ra là hình dạng B × C 0 × F 0 × N 0 với C 0 = 256, F 0 = 4 và N 0 = N 8. Mỗi lượt chuyển đổi 2D Lớp sử dụng kích thước hạt nhân là 3 × 3 và tiếp theo là chuẩn hóa hàng loạt và kích hoạt ReLU.

CNN 2D được dùng để trích xuất các bản đồ tính năng cấp cao từ quang phổ năng lượng băng tần log-mel, trong khi đó CNN 1D cung cấp các biểu diễn tổng thể bằng cách cùng xử lý tất cả các tần số và kênh của các khung liền kề. Do đó, nó lấy đầu ra được định hình lại (B × C 0 · F 0 × N 0) của CNN 2D làm đầu vào và áp dụng ba lượt chuyển đổi 1D của mỗi lớp với 256 kênh ẩn. Mỗi lần chuyển đổi 1D, lớp sử dụng kích thước hạt nhân là 3 và tiếp theo là chuẩn hóa hàng loạt và kích hoạt ReLU.

Đầu ra của CNN sau đó được đưa vào bộ mã hóa tuần tự lặp lại. Tác giả sử dụng hai lớp Gated Recurrent Units (GRU) với 256 đơn vị mỗi lớp. Chỉ vectơ đầu ra cuối cùng của mỗi chuỗi trong một unit được chuyển tiếp đến mạng phân loại. Các nhóm mạng phân loại được kết nối đầy đủ của một lớp ẩn với 256 đơn vị ẩn và chức năng kích hoạt ReLU và lớp phân loại cuối cùng với điểm xuất kích hoạt Sigmoid từ 0 đến 1 cho mỗi trong số 80 lớp sự kiện mục tiêu.

2.3.2.4 Phương pháp huấn luyện

Để huấn luyện mô hình mạng nơ-ron nhân tạo đã trình bày ở trên, ta sử dụng cả dữ liệu có nhãn được sắp xếp cũng như dữ liệu có nhãn nhiễu.

Vì phần train được cung cấp có nhãn nhiễu chủ yếu chứa các tín hiệu âm thanh có độ dài 15 giây và mô hình của chúng tôi xử lý toàn bộ các chuỗi, nên có sự thiên vị đối với các chuỗi có độ dài 15 giây. Để khắc phục sự sai lệch này, trước khi bắt đầu đào tạo ta cần khởi tạo các đoạn trích âm thanh mới có độ dài khác nhau bằng cách tách ngẫu nhiên từng tín hiệu âm thanh có nhãn nhiễu thành hai ở độ dài l = ( ·) với là độ dài của tín hiệu thứ m và ∼ U (0,1, 0,9). Điều này dẫn đến độ dài đoạn trích âm thanh được phân bổ gần như đồng đều trong khoảng từ 1 giây đến 13,5 giây. Vì việc tăng cường dữ liệu trộn có thể kéo dài tín hiệu âm thanh, việc trộn lẫn các đoạn trích nhiễu tạo ra tín hiệu được phân phối trong khoảng từ 1 giây đến 27 giây. Mỗi đoạn trích âm thanh sao chép các thẻ sự kiện của âm thanh gốc, điều này dẫn đến một số tạp âm bổ sung cho nhãn. Việc tách ngẫu nhiên được thực hiện ba lần dẫn đến ba bộ dữ liệu khác nhau và tác giả gọi là phép tách 0-2.

Để tối ưu hơn phương pháp huấn luyện, tác giả đưa ra tiêu chí huấn luyện là cross-entropy nhị phân giữa các dự đoán của mô hình và vectơ mục tiêu n-hot theo phương trình sau:

= -

Với k = 80 biểu thị cho số lớp sự kiện mục tiêu. Ta lấy mẫu ngẫu nhiên các lô nhỏ có kích thước 16 từ dữ liệu đào tạo sao cho:

* Không có tín hiệu nào trong lô nhỏ được đệm hơn 20%,
* Không có ví dụ nào trong lô mini bao gồm các sự kiện tương tự như một ví dụ khác trong cùng một lô nhỏ
* Và tính toán độ dốc của tổn thất trung bình trong lô nhỏ. Ta cắt gradient ở ngưỡng 15 để tối ưu hóa. Huấn luyện được thực hiện cho 200K lần lặp.

Trong phương pháp này, tác giả thực hiện trung bình trọng số Stochastic (Stochastic Weight Averaging - SWA) cho 50 nghìn lần lặp cuối cùng với tần suất 1 nghìn lần lặp. Vào cuối quá trình đào tạo, họ thay trọng lượng mô hình cho trọng lượng trung bình. Ở bước cuối cùng là việc cập nhật số liệu thống kê của tất cả các lớp chuẩn hóa hàng loạt bằng cách chuyển tiếp dữ liệu đào tạo của họ bằng cách sử dụng mô hình SWA. SWA cho thấy sự cải thiện khả năng tổng quát hóa và không cần dữ liệu dự trữ để xác định điểm kiểm tra hoạt động tốt nhất.

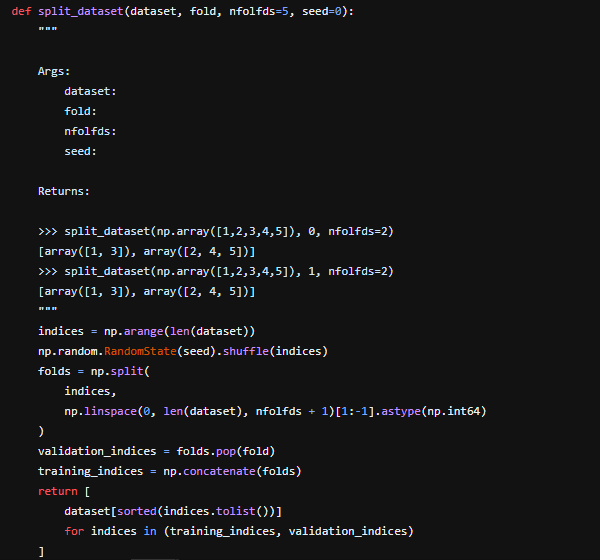
2.4 Thực nghiệm

2.4.1 Dữ liệu

Tập dữ liệu được cung cấp sẵn trên trang chủ của cuộc thi với tên gọi “FSDKaggle2019” bao gồm hai tập con: một tập hợp nhỏ gồm 4970 đoạn âm thanh được gắn nhãn thủ công và một tập hợp ồn ào gồm 19815 đoạn âm thanh trong đó các nhãn được tự động lấy từ các dự đoán cấp video từ nhiều mô hình âm thanh được đào tạo trước. Một dictionary gồm 80 sự kiện âm thanh được sử dụng, trong đó nhiều sự kiện có thể hoạt động cùng một lúc dẫn đến vấn đề phân loại đa nhãn.

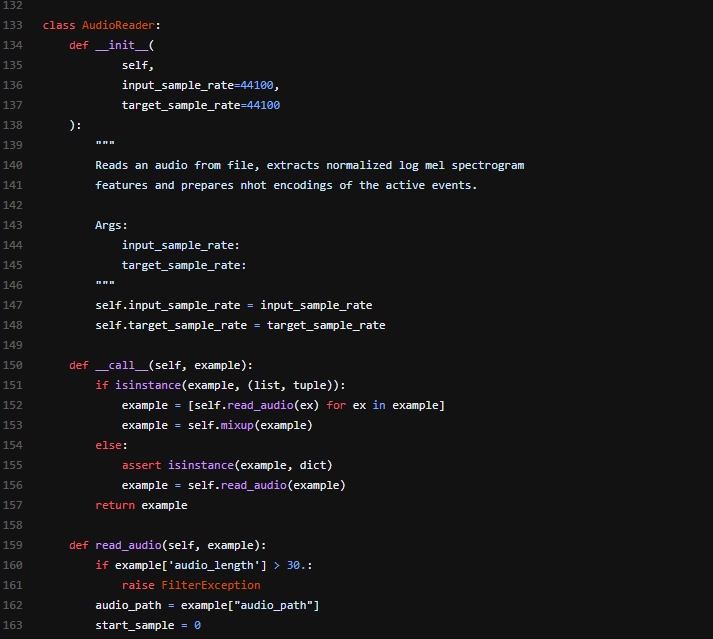
2.4.2 Xử lý dữ liệu

Do tập dữ liệu FDSKaggle2019 nhiều nên họ cần họ chia nhỏ dataset để dễ xử lý thông qua hàm split\_dataset() như sau:



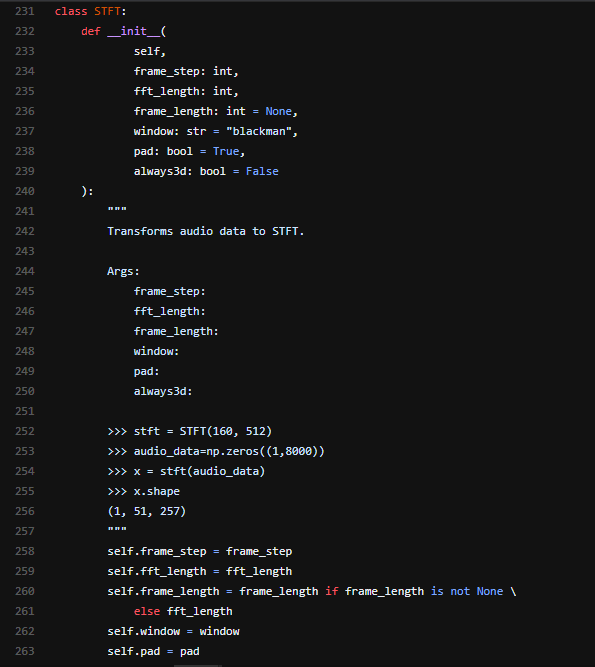
Hình . Hàm split\_dataset

Hàm này trả về một danh sách nhiều các sub-dataset. Sau đó họ khởi tạo một đối tượng AudioReader dùng để đọc âm thanh từ tệp, trích xuất biểu đồ quang phổ log mel được chuẩn hóa các tính năng và chuẩn bị mã hóa nhỏ của các sự kiện đang hoạt động.



Hình . Đối tượng AudioReader

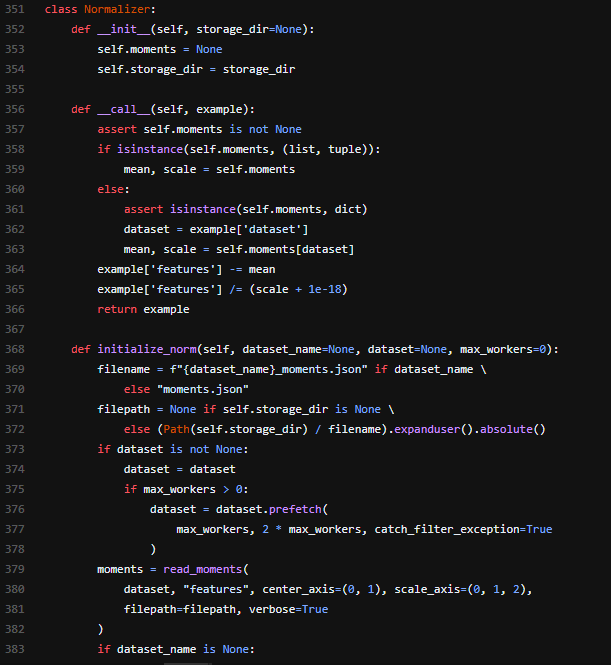
Bước tiếp theo, họ chuyển dữ liệu audio đó thành dạng STFT như sau:



Hình . Chuyển dữ liệu audio thành STFT

Cụ thể, tác giả thực hiện STFT với độ dài khung hình là 40 ms (1764 mẫu) và kích thước bước nhảy là 20 ms (882 mẫu) trên tín hiệu âm thanh 44,1 kHz được cung cấp mà không cần lấy mẫu lại. Sau đó, đối với mỗi khung, chúng tôi trích xuất 128 đặc điểm năng lượng băng tần log-mel với fmin = 50 Hz và fmax = 16 kHz. Cuối cùng, họ lấy giá trị trung bình chung của mỗi đối tượng và sau đó chia cho độ lệch chuẩn chung trên tất cả các đối tượng.

Bước cuối cùng trong phần này là Normalize tập dữ liệu và kết thúc quá trình tiền xử lý dữ liệu:



Hình . Normalize tập dữ liệu và kết thúc quá trình tiền xử lý dữ liệu:

2.4.3 Công nghệ sử dụng

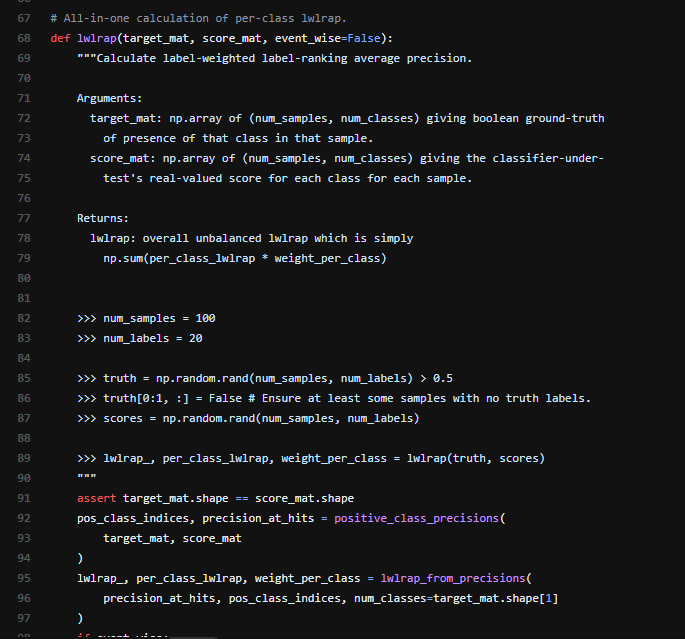
* Ngôn ngữ lập trình: Python
* cached\_property phiên bản 1.5.2 trở lên
* sacred phiên bản 0.8.2
* scikit-image phiên bản 0.19.0
* Pytorch phiên bản 1.8.0 gồm các module nhỏ như 'torch', 'torchcontrib', 'torchvision', 'tensorboardX'

2.4.4 Cách đánh giá

Trong cuộc thi của Kaggle Freesound Audio Tagging 2019 này, các phương pháp hầu hết đều được đánh giá bởi Lwlrap score (Label ranking average precision) bởi vì nó đã được chỉ mục của nó thông qua như một bản ghi nhớ của nó để biết một lần nữa. Đây là một trong những thước đo về độ chính xác, một trong những chỉ số đánh giá độ chính xác của mô hình được sử dụng làm độ chính xác của bài toán phân loại nhiều nhãn.

Các nhãn xếp hạng chính xác thay vì sử dụng Mathcing Rate hoặc Recall Score với đầu ra giá trị tương đối bởi hàm Sigmoid hoặc hàm Softmax, những nhãn được xếp hạng theo giá trị thứ tự lớn hơn được sử dụng.

Độ đo Lwlrap mà phương pháp này sử dụng được tính dựa trên phát biểu của một Kaggler tên là PankajMahan cũng là người tham gia cuộc thi này trên Kaggler với đường dẫn: <https://www.kaggle.com/pkmahan/understanding-lwlrap>. Và toàn bộ source code đánh giá được chứa trong file lwlrap.py của chương trình:



Hình . Đánh giá mô hình

2.5 Kết quả đạt được

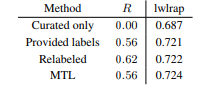
Trong phần này em sẽ trình bày quá trình đánh giá các phương pháp khác nhau để xử lý nhiễu nhãn khi sử dụng tất cả các kỹ thuật nâng cao dữ liệu được đề xuất.

Đối với mỗi phần trong số ba phần tách, phương pháp này đã huấn luyện các mô hình trên năm phần gấp khác nhau bằng cách sử dụng tỷ lệ nhãn nhiễu là R = 0,5. Điều này dẫn đến tổng cộng 15 mô hình khác nhau, tất cả được kết hợp thành một nhóm để đưa ra dự đoán cho các đoạn trích nhiễu từ cả ba phần tách. Ở đây, điểm số sự kiện của các mô hình riêng lẻ được tính trung bình để thu được điểm số đầu ra tổng thể. Sau đó, đối với mỗi sự kiện, tác giả xác định ngưỡng quyết định mang lại tỷ lệ lỗi tốt nhất được đánh giá chung trên tập hợp tất cả các đoạn trích nhiễu từ tất cả các phần tách. Các ngưỡng quyết định này được sử dụng để gắn nhãn lại các đoạn trích bị nhiễu, trong đó các đoạn trích không có bất kỳ sự kiện hoạt động nào sẽ bị loại bỏ. Sau đó, họ sử dụng toàn bộ dữ liệu được gắn nhãn lại của phần tách để đào tạo một mô hình mới vì chúng tôi không cần dữ liệu dự trữ để xác định điểm kiểm tra tốt nhất do SWA.

Đội chiến thắng trong Task 2 của cuộc thi DCASE 2019 đã đề xuất phương pháp Học đa tác vụ (Multi Task Learning- MTL) để đối phó với các nhãn nhiễu. Ở đây, em sẽ so sánh các kết quả được được của phương pháp này với phương pháp của đội chiến thắng trên, từ đó rút ra được các ưu điểm, hạn chế của phương pháp và đề xuất phương hướng phát triển.

Cụ thể, các lớp phân loại khác nhau được sử dụng trong quá trình đào tạo để dự đoán các nhãn được sắp xếp và tạp âm tương ứng. Sau khi thử thách kết thúc, chúng tôi đã áp dụng cách tiếp cận này cho mô hình của mình, tức là chúng tôi đào tạo các lớp được kết nối đầy đủ khác nhau cho các nhãn được sắp xếp và nhiễu. Tại thời điểm thử nghiệm, chỉ các lớp phân loại được đào tạo trên các nhãn được sắp xếp mới được sử dụng để đưa ra dự đoán.

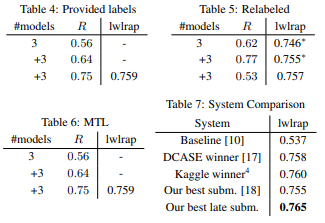
Kết quả cho các phương pháp xử lý nhiễu nhãn khác nhau được trình bày trong hình bên dưới:



Hình . Kết quả cho các phương pháp xử lý nhiễu nhãn khác nhau

Có thể thấy rằng việc sử dụng các nhãn nhiễu dẫn đến hiệu suất tăng đáng kể. Tuy nhiên, việc sử dụng gắn nhãn lại và MTL chỉ mang lại cải thiện nhỏ cho lwlrap.

Trong phần tiếp theo, các phương pháp này sẽ được đánh giá thêm khi thực hiện tập hợp. Cụ thể tác giả đánh giá các tập hợp của các mô hình khác nhau. Đặc biệt, họ kết hợp các mô hình được đào tạo trên các phân tách khác nhau và với tỷ lệ nhãn nhiễu khác nhau. Sự kết hợp hệ thống đạt được bằng cách lấy trung bình điểm đầu ra của các hệ thống đóng góp.



Hình . Đánh giá các tập hợp của các mô hình khác nhau

Mỗi hàng trong Bảng 4-6 thêm ba mô hình đã được huấn luyện trên ba phần khác nhau. Các kết quả được đánh dấu bằng dấu hoa thị trong Bảng 5 thể hiện sự đệ trình của chúng tôi đối với thử thách. Có thể thấy rằng việc gắn nhãn lại hoặc học đa tác vụ riêng lẻ đều không cải thiện hiệu suất so với các nhãn được cung cấp. Tuy nhiên, kết hợp tất cả các mô hình từ Bảng 4-6 thành một nhóm lớn gồm 27 mô hình, tăng hiệu suất lên 76,5% lwlrap, thiết lập một trạng thái mới nhất cho nhiệm vụ này như được hiển thị trong Bảng 7. Lưu ý rằng nhóm này có tổng thời gian ngừng hoạt động <2 giờ khi sử dụng CPU, đáp ứng các hạn chế về thử thách.

2.6 Kết luận

Các kết quả thực nghiệm của phương pháp này được thực hiện trên dữ liệu “FSDKaggle2019”, từ đó làm nổi bật tầm quan trọng của các kỹ thuật xử lý dữ liệu để đạt được hiệu suất phân loại cao. Tuy điểm mạnh là như thế, điểm yếu của phương pháp này nằm ở việc xét cụ thể về từng tiếng ồn nhãn không tỏ ra hiệu quả. Hơn nữa, hiệu suất của mô hình mạng nơ-ron của họ sẽ được tăng cường nhờ sự kết hợp hệ thống nâng hiệu suất từ ​​0,724 lwlrap của hệ thống mô hình đơn tốt nhất của họ lên 0,765 bằng cách kết hợp tổng số 27 mô hình, thiết lập một trạng thái mới cho mạng nơ-ron nhân tạo này.

**TÀI LIỆU THAM KHẢO**

1. Michael Mandel, Justin Salamon, and Daniel P. W. Ellis*, Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)*. New York University, NY, USA October 2019.
2. Janek Ebbers, Reinhold Haeb-Umbach*, Convolutional Recurrent Neural Network and Data Augmentation for Audio Tagging with Noisy Labels and Minimal Supervision.* Paderborn University, Department of Communications Engineering, Paderborn, Germany.
3. Eduardo Fonseca, Manoj Plakal, Frederic Font, Daniel P. W. Ellis, Xavier Serra, *Audio Tagging with Noisy Labels and Minimal Supervision.* Music Technology Group, Universitat Pompeu Fabra, Barcelona.
4. Shubhr Singh, Arjun Pankajakshan and Emmanouil Benetos, *Audio Tagging using Linear Noise Modelling Layer.* School of Electronic Engineering and Computer Science, Queen Mary University of London.
5. M. D. P. T. Virtanen and D. Ellis, *Computational Analysis of Sound Scenes and Events.*
6. S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, and K. W. Wilson, *“CNN architectures for large-scale audio classification,” in International Conference on Acoustics, Speech and Signal Processing* (ICASSP).
7. Eduardo Fonseca*,* Shawn Hershey, Manoj Plakal, Daniel P.W. Ellis, *Addressing Missing Labels in Large-scale Sound Event Recognition using a Teacher-student Framework with Loss Masking.* Creative Commons Attribution 4.0 International, University Pompeu Fabra.
8. *“DCASE 2019 Task2.”* [Online]. Available: https://dcase. community/challenge2019/task-audio-tagging
9. “*FSDKaggle2019*”. [Online]. Available: <https://zenodo.org/record/3612637>
10. osciiart. *“make\_final\_submission1”*. [Online]. Available: https://www.kaggle.com/osciiart/make-final-submission1/data
11. romul0212, *“freesound-predict-dataset”*. [Online]. Available: <https://www.kaggle.com/romul0212/freesound-predict-dataset/code?select=submission.csv>
12. sudarsanvs. “*free-sound-audio-tagging”.* [Online]. Available: [*https://www.kaggle.com/sudarsanvs/free-sound-audio-tagging*](https://www.kaggle.com/sudarsanvs/free-sound-audio-tagging)*.*
13. pkmahan. “*understanding-lwlrap*” [Online]. Available: [*https://www.kaggle.com/pkmahan/understanding-lwlrap*](https://www.kaggle.com/pkmahan/understanding-lwlrap)
14. fgnt*, UPB system for the Kaggle competition "Freesound Audio Tagging 2019"* [Online]. Available: [*https://github.com/fgnt/upb\_audio\_tagging\_2019*](https://github.com/fgnt/upb_audio_tagging_2019)*.*

TỰ ĐÁNH GIÁ

(Với nhóm có 2 thành viên)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Câu | Nội dung | Điểm chuẩn | Tự chấm | Ghi chú |
| 1  (4.5) | **1.1 Mô tả cấu trúc dữ liệu** | 1 | 1 |  |
| **1.2 Sơ đồ giải thuật** | 1 | 1 |  |
| **1.3 Hiện thực** | 2 | 2 |  |
| 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 | 1.25 |  |
| 2.4 Thực nghiệm | 1 | 1 |  |
| **2.5 Kết quả đạt được** | 0.75 | 0.75 |  |
| 2.6 Kết luận | 0.25 | 0.25 |  |
| 3 | Điểm nhóm | 1đ | 1 |  |
| Tổng điểm | | | 10 |  |