**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**ĐỒ ÁN HỌC PHẦN**

TÊN HỌC PHẦN: **TRÍ TUỆ NHÂN TẠO (Artificial Intelligence: AI)**

MÃ SỐ LỚP HP: **ARIN330585 - Nhóm 03 (Sáng thứ 3)**

Tên đề tài:**PHÁT TRIỂN HỆ THỐNG TRỢ LÝ ẢO HỔ TRỢ PHÁT HIỆN VÀ NHẬN DIỆN KHUÔN MẶT NGƯỜI TRONG HÌNH**

Họ tên sinh viên: **NGUYỄN PHÚC AN**

**Mã số sinh viên: 19110321 [01]**

**Lớp: 191102A**

**Ngày nộp: 19/06/2021**

**Ký tên:**

**TP.HCM, ngày 10 tháng 06 năm 2021**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

**ĐỀ TÀI ĐỒ ÁN HỌC PHẦN**

**Giảng viên giảng dạy: VÕ XUÂN THỂ**

TÊN HỌC PHẦN: **TRÍ TUỆ NHÂN TẠO (Artificial Intelligence: AI)**

MÃ SỐ LỚP HP: **ARIN330585 - Nhóm 03 (Sáng thứ 3)**

Tên đề tài: **PHÁT TRIỂN TRỢ LÝ ẢO HỖ TRỢ PHÁT HIỆN VÀ NHẬN DIỆN KHUÔN MẶT NGƯỜI TRONG HÌNH**

Họ tên sinh viên: **Nguyễn Phúc An**

**Mã số sinh viên: 19110321 [01]**

**Lớp: 191102A**

**Tên sản phẩm đề tài: N3S3.01.NguyenPhucAn.19110321.DoaAnHP.AI.rar**

**Công cụ sử dụng : Visual Studio Code**

**Ngôn ngữ lập trình sử dụng: Python 3.9.1**

**Nhận xét của giảng viên:**

**………………………………………………………..**

**………………………………………………………..**

**Điểm đánh giá:………….(…………………)**

**Ngày……./……../2021**

**Giảng viên Ký tên**

**TP.HCM, ngày 10 tháng 06 năm 2021**

# LỜI CẢM ƠN

Em xin gửi lời cảm ơn chân thành nhất đến thầy Võ Xuân Thể đã giảng dạy và hướng dẫn chúng em trong suốt quá trình học môn Nhập môn Trí tuệ nhân tạo, thầy luôn giải đáp thắc mắc mà chúng em có cũng như luôn tạo điều kiện học tập thuận lợi nhất cho chúng em trong quá trình học tập. Thầy cũng đã hướng dẫn, góp ý cho chúng em trong suốt quá trình làm đề tài để cho chúng em không bị lạc hướng hay sai đề.

Vì khả năng còn hạn chế nên trong quá trình thực hiện báo không tránh khỏi sơ sài, sai sót, nên mong thầy thông cảm cũng như có những nhận xét, đánh giá để em có thể hoàn thành bài báo cáo này tốt hơn.

# DANH MỤC CÁC TỪ VIẾT TẮT

|  |  |
| --- | --- |
|  | |
| CSDL hoặc DB | Cơ sở dữ liệu: DataBase |
|  | |
| AI | Trí tuệ (Trí thông minh) nhân tạo: Artificial Intelligence |
| ANN | Mạng nơ ron nhân tạo: Artificial Neural Network |
| ES | Hệ thống chuyên gia: Expert Systems |
| ML | Máy học = Học máy: Machine Learning |
| NLP | Xử lý ngôn ngữ tự nhiện: Natural Language Processing |
| MIS | Hệ thống thông tin quản lý = Management Information System |
| KBMS | Các sản phẩm phần mềm mà trong đó có cung cấp 2 nhóm công cụ và phương tiện để phát triển, duy trì phát triển cơ sở tri thức. |
| CV | Nhận diện, xử lý hình ảnh = Computer vision |

# DANH MỤC CÁC THUẬT NGỮ

|  |  |
| --- | --- |
| Bài toán (vấn đề) AI:  AI Problem | Là các tình huống thực tế mà con người cần giải quyết thuộc một lĩnh vực nào đó, thông thường phải dựa vào “trí khôn” con người. |
| Giao diện người dùng  (User-Interface) | Là hệ thống các màn hình giao tiếp cho phép người sử dụng tương tác với các thành phần phần mềm, điều khiển phần mềm hoạt động theo yêu cầu của người dùng - tương ứng các chức năng hiện có của phần mềm. |
| Người dùng (User):  Tài khoản (Account) | Là một quyền làm việc trên hệ thống phần mềm được cấp phát cho một cá nhân thông qua tên tài khoản (username) và mật khẩu (password). |
|  |  |
| BigData | Dữ liệu lớn: là một tập hợp dữ liệu rất lớn và phức tạp, không thể xử lý dữ liệu bằng các phương pháp truyền thống. |
| Heuristics | Là các kỹ thuật dựa trên kinh nghiệm để giải quyết vấn đề, nhằm đưa ra một giải pháp mà không được đảm bảo là tối ưu |

# MỤC LỤC

[LỜI CẢM ƠN 3](#_Toc74738035)

[DANH MỤC CÁC TỪ VIẾT TẮT 4](#_Toc74738036)

[DANH MỤC CÁC THUẬT NGỮ 5](#_Toc74738037)

[MỤC LỤC 6](#_Toc74738038)

[PHIẾU GIAO NHIỆM VỤ ĐỀ TÀI ĐỒ ÁN HỌC PHẦN 9](#_Toc74738039)

[Chương 1: GIỚI THIỆU VỀ ĐỀ TÀI 10](#_Toc74738040)

[**1.1.** Tổng quan về đề tài 10](#_Toc74738041)

[**1.2.** Nội dung chuyên môn chính của đề tài 10](#_Toc74738042)

[**1.3.** Ngôn ngữ lập trình và công cụ sử dụng 10](#_Toc74738043)

[**1.4.** Sản phẩm của đề tài 11](#_Toc74738044)

[**1.5.** Bố cục của báo cáo 11](#_Toc74738045)

[Chương 2: CƠ SỞ LÝ THUYẾT VỀ TRÍ TUỆ NHÂN TẠO 13](#_Toc74738046)

[**2.1.** Tổng quan về Trí tuệ nhân tạo (AI) 13](#_Toc74738047)

[2.1.1. Các khái niệm 13](#_Toc74738048)

[2.1.2. Vai trò AI 15](#_Toc74738049)

[2.1.3. Nền tảng kỹ thuật của AI 15](#_Toc74738050)

[2.1.4. Các lĩnh vực nghiên cứu và ứng dụng cơ bản của AI 15](#_Toc74738051)

[**2.2.** Giới thiệu về các bài toán (vấn đề) AI 16](#_Toc74738052)

[2.2.1. Một số khái niệm 16](#_Toc74738053)

[2.2.2. Tìm kiếm lời giải (searching) cho AI Problem 17](#_Toc74738054)

[**2.3.** Biểu diễn một số bài toán AI 18](#_Toc74738055)

[2.3.1. Các bài toán Trò chơi 18](#_Toc74738056)

[2.3.2. Cây không gian trạng thái của Bài toán AI 18](#_Toc74738057)

[**2.4.** Các phương pháp biểu diễn tri thức 19](#_Toc74738058)

[2.4.1. Phân loại tri thức theo phương pháp biểu diễn 19](#_Toc74738059)

[2.4.2. Phân loại tri thức theo nền tảng hình thành Cơ sở tri thức 19](#_Toc74738060)

[2.4.3. Logic mệnh đề và Logic vị từ 20](#_Toc74738061)

[2.4.4. Biểu diễn tri thức bằng Frame (Cấu trúc = "Khung") 23](#_Toc74738062)

[2.4.5. Suy diễn tri thức bằng luật dẫn xuất 23](#_Toc74738063)

[2.4.6. Biểu diễn suy luận bằng đồ thị AND/OR 24](#_Toc74738064)

[2.4.7. Các biểu diễn tương đương trong Logic mệnh đề và Logic vị từ 24](#_Toc74738065)

[2.4.8. Xử lý ngôn ngữ tự nhiên (Natural Language Processing = NLP): Voice Assistant 27](#_Toc74738066)

[**2.5.** Các phương pháp tìm kiếm lời giải của bài toán AI 27](#_Toc74738067)

[2.5.1. Giới thiệu 27](#_Toc74738068)

[2.5.2. Các phương pháp tìm kiếm mù (Blind Search) = tìm kiếm không có thêm thông tin (uninformed search) 27](#_Toc74738069)

[2.5.3. Các phương pháp tìm kiếm theo kinh nghiệm (Heuristic Search) = tìm kiếm với thông tin bổ sung (Informed Search) = Cải tiến BFS 29](#_Toc74738070)

[2.5.4. Lập trình Logic (Logic Programming) phát triển cơ chế lập luận logic 31](#_Toc74738071)

[Chương 3: GIỚI THIỆU VỀ MÔ HÌNH BÀI TOÁN AI: PHÁT TRIỂN HỆ THỐNG TRỢ LÝ ẢO HỖ TRỢ PHÁT HIỆN VÀ NHẬN DIỆN KHUÔN MẶT NGƯỜI TRONG HÌNH 32](#_Toc74738072)

[3.1. Mô tả bài toán AI 32](#_Toc74738073)

[3.2. Các yếu tố xác định bài toán AI 32](#_Toc74738074)

[Chương 4: BIỂU DIỄN BÀI TOÁN AI: PHÁT TRIỂN HỆ THỐNG TRỢ LÝ ẢO HỖ TRỢ PHÁT HIỆN VÀ NHẬN DIỆN KHUÔN MẶT NGƯỜI TRONG HÌNH 33](#_Toc74738075)

[4.1. Giới thiệu 33](#_Toc74738076)

[4.2 Lập trình xử lý ngôn ngữ tự nhiên 33](#_Toc74738077)

[Chương 5: PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI CHO BÀI TOÁN AI: PHÁT TRIỂN HỆ THỐNG TRỢ LÝ ẢO HỖ TRỞ PHÁT HIỆN VÀ NHẬN DIỆN KHUÔN MẶT NGƯỜI TRONG HÌNH 36](#_Toc74738078)

[5.1. Giới thiệu 36](#_Toc74738079)

[5.2. Xác định phương pháp tìm kiếm lời giải cho bài toán AI: Phát triển hệ thống hỗ trợ phát hiện và nhận diện khuôn mặt người trong hình 36](#_Toc74738080)

[5.3. Mô tả phương pháp tìm kiếm lới giải cho bài toán AI:Phát triển hệ thống trợ lý ảo hỗ trợ phát hiện và nhận diện khuôn mặt người trong hình 36](#_Toc74738081)

[5.4. Cài đặt (lập trình) phương pháp tìm kiếm lới giải cho bài toán AI:Phát triển hệ thống hỗ trợ phát hiện và nhận diện khuôn mặt người trong hình 38](#_Toc74738082)

[Chương 6: GIỚI THIỆU SẢN PHẨM CỦA ĐỀ TÀI AI: PHÁT TRIỂN HỆ THỐNG TRỢ LÝ ẢO HỖ TRỢ PHÁT HIỆN VÀ NHẬN DIỆN KHUÔN MẶT NGƯỜI TRONG HÌNH 44](#_Toc74738083)

[6.1. Giới thiệu sản phẩm bài toán AI: Phát triển hệ thống trợ lý hỗ trợ ảo phát hiện và nhận diện khuôn mặt người trong hình 44](#_Toc74738084)

[6.2. Kết quả thực nghiệm sản phẩm bài toán AI: Phát triển hệ thống trợ lý ảo hỗ trợ phát hiện và nhận diện khuôn mặt người trong hình 53](#_Toc74738085)

[6.3. Nhận xét và đánh giá về sản phẩm bài toán AI: Phát triển hệ thống trợ lý ảo hỗ trợ phát hiện và nhận diện khuôn mặt người trong hình 56](#_Toc74738086)

[Chương 7: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN CỦA ĐỀ TÀI 57](#_Toc74738087)

[7.1. Kết luận 57](#_Toc74738088)

[7.1.1. Những kết quả đạt được 57](#_Toc74738089)

[7.1.2. Hạn chế 57](#_Toc74738090)

[7.2. Hướng phát triển 57](#_Toc74738091)

[7.2.1. Hướng khắc phục các hạn chế 57](#_Toc74738092)

[7.2.2. Hướng mở rộng đề tài 58](#_Toc74738093)

[DANH MỤC TÀI LIỆU THAM KHẢO 59](#_Toc74738094)

[CÁC PHỤ LỤC 60](#_Toc74738095)

[ĐÁNH GIÁ KẾT QUẢ HỌC TẬP HỌC PHẦN 61](#_Toc74738096)

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

# PHIẾU GIAO NHIỆM VỤ ĐỀ TÀI ĐỒ ÁN HỌC PHẦN

|  |  |
| --- | --- |
| Giảng viên giảng dạy: **VÕ XUÂN THỂ** | |
| Tên học phần: **Trí tuệ nhân tạo (AI)**  Mã số lớp HP: **ARIN330585 – N3S3** | |
| Tên đề tài:**PHÁT TRIỂN TRỢ LÝ ẢO HỖ TRỢ PHÁT HIỆN VÀ NHẬN DIỆN KHUÔN MẶT NGƯỜI TRONG HÌNH (AI).** | |
| Sinh viên thực hiện: **Nguyễn Phúc An, 19110321** | |
| Thời gian thực hiện: **23/02/2021**  đến **10/06/2021** | |
| **Yêu cầu của đề tài**  **Lý thuyết:** Vận dụng AI vào thực tiễn và tiếp cận thành tựu tiên tiến trong lĩnh vực AI:  + Nền tảng và đặc trưng của các dạng bài toán AI và giải thuật tương ứng.  + Một số phương pháp biểu diễn không gian (trạng thái) giải quyết bài toán AI: biểu diễn tri thức và suy diễn logic.  + Một số phương pháp thông dụng trong tìm kiếm lời giải bài toán AI = giải quyết bài toán thông qua tìm kiếm lời giải (Solving Problems by Searching), đặc biệt là DFS và BFS,…  **Thực hành:** Phát triển trợ lý ảo phát hiện và nhận diện khuôn mặt dựa vào các giải thuật trí tuệ nhân tạo (AI):  + Thư viện chính: OpenCV, face\_detection (Yêu cầu: Cmake,dlib), gtts, speech\_recognition; Giải thuật Haar Cascade Classifier, phương pháp HOG. | |
| **GIẢNG VIÊN** | **Ngày 23 tháng 02 năm 2021**  **SV Thực hiện** |

# GIỚI THIỆU VỀ ĐỀ TÀI

## Tổng quan về đề tài

Mục tiêu của đề tài là làm một hệ thống trợ lý ảo với tính năng chính là hỗ trợ phát hiện khuôn mặt của người trong bức hình hoặc qua webcam và sau đó nhận diện người đó là ai, hiểu hơn về cách thuật toán sử dụng để phát hiện khuôn mặt thông qua việc thực hiện đề tài. Sử dụng thư viện OpenCV, face\_detection, os, speech\_recognition,gtts với ngôn ngữ lập trình Python để thực hiện đề tài.

## Nội dung chuyên môn chính của đề tài

+ Thư viện:

* gtts, speech\_recognition để xủ lý ngôn ngữ tự nhiên như nhận diện giọng nói, chuyển đổi giọng nói sang văn bản và văn bản sang giọng nói.
* OpenCV, dùng để nạp hình ảnh vào phần mềm, hoặc sử dụng webcam để làm sản phẩm thử nghiệm.
* face\_detection, dùng để nhận diện xem khuôn mặt là ai (Cần có Cmake và dlib để có thể cài thư viện face\_detection).
* os, dùng để đọc hình ảnh trong thư mục.

+ Giải thuật: Haar Cascade Classifier, HOG.

+ Sản phẩm đề tài: face\_detection.py.

+ Kết quả thực nghiệm:

Về khả năng của trợ lý ảo thì khả năng nghe giọng nói của phần mềm là tốt, tùy thuộc vào âm thanh môi trường có bị nhiễu hay không và chất lượng âm thanh đầu vào. Thực hiện tốt các chức năng đã được cài đặt.

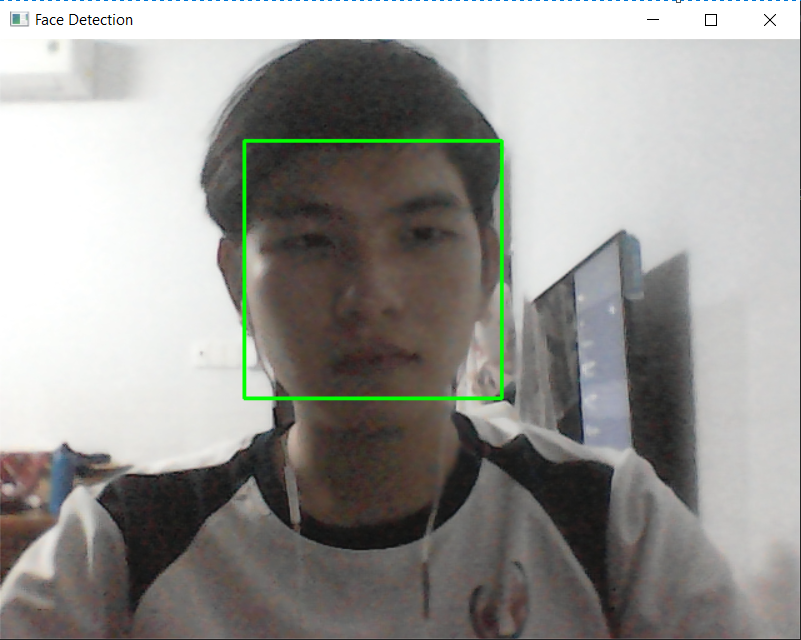
Về khả năng phát hiện và nhận diện phần mềm có độ chính xác tầm 85%, đôi khi không thể phát hiện ra nếu người trong ảnh hoặc trong webcam có đeo kính hoặc bịt mặt… Và nhận diện khá chính xác người trong ảnh là ai nếu có trước dữ liệu của người đó.

## Ngôn ngữ lập trình và công cụ sử dụng

+ Ngôn ngữ lập trình: Python 3.9.1

+ Công cụ sử dụng : Visual Studio Code

## Sản phẩm của đề tài



## Bố cục của báo cáo

Chương 1: Giới thiệu về đề tài

Chương 2: Cơ sở lý thuyết về trí tuệ nhân tạo

Chương 3: Giới thiệu về mô hình bài toán AI: Phát triển hệ thống trợ lý ảo hỗ trợ phát hiện và nhận diện khuôn mặt người trong hình

Chương 4: Biểu diễn bài toán AI: Phát triển hệ thống trợ lý ảo hỗ trợ phát hiện và nhận diện khuôn mặt người trong hình

Chương 5: Phương pháp tìm kiếm lời giải cho bài toán AI: Phát triển hệ thống trợ lý ảo hỗ trợ phát hiện và nhận diện khuôn mặt người trong hình

Chương 6: Giới thiệu sản phẩm của đề tài AI: Phát triển hệ thống trợ lý ảo hỗ trợ phát hiện và nhận diện khuôn mặt người trong hình

Chương 7: Kết luận và hướng phát triển của đề tài

# CƠ SỞ LÝ THUYẾT VỀ TRÍ TUỆ NHÂN TẠO

## Tổng quan về Trí tuệ nhân tạo (AI)

### Các khái niệm

* + - 1. Trí tuệ nhân tạo = AI (Artificial Intelligence)

Là ngành khoa học nghiên cứu và ứng dụng các nền tảng kỹ thuật làm cơ sở cho việc thiết kế và sản xuất ra các hệ thống, đặc biệt là các hệ thống CNTT, có khả năng thực hiện các hành vi công việc “khôn” như con người.

Có 2 dạng biểu hiện “khôn” của các hệ thống AI: “hành vi” & “nhận thức”/“suy luận”.

Các đặc trưng chính của hệ thống AI:

* Các kết quả xử lý tính toán, hành vi không nhất thiết chính xác 100%, tỷ lệ sai số tùy thuộc vào hệ thống AI cụ thể.
* Không nhất thiết phải chứng minh hoặc giải thích các kết quả. Lấy hiệu quả làm mục tiêu của các hệ thống AI.
* Có khả năng “học” và “tự học”.
* “Khôn” dần theo thời gian.
* Gồm “chuyên gia chuyên môn” và người lập trình xây dựng phần mềm AI.
* Ngôn ngữ và công cụ riêng.
  + - 1. **Phân loại hệ thống Trí tuệ nhân tạo**

+ Dựa vào “nền tảng trí khôn”: Heuristics và ES.

+ Dựa vào biểu hiện “khôn”: “hành vi”.

+ Dụa vào nền tảng kỹ thuật: ML, ANN, Mờ,…

* + - 1. **Turing Test kiểm tra khả năng khôn của các hệ thống AI**
      2. **Tác tử (Agent: Intelligent Agent) trong hệ thống AI**

Nhân tố xử lý bé nhất để thực hiên các chức năng trong các chương trình trí tuệ nhân tạo được gọi là tác tử.

Tác tử (Agent) có thể biểu diễn dạng hàm ánh xạ:

F: P\* 🡪 A

* + - 1. **Tri thức là gì (Knowledge)?**

Tri thức là hệ thống sự hiểu biết bằng lý thuyết hay thực tế về một chủ đề hay lĩnh vực. Có hai loại tri thức:

+ Tri thức tự nhiên là hệ thống trí thức có sẵn trong tự nhiên, tự biết.

+ Tri thức hệ thống có được bằng cách học hỏi, tiếp thu đúc kết mà có.

Các sản phẩm trí tuệ nhân tạo là được tạo ra từ trí tuệ hệ thống.

Phân biệt giữa phần phần mềm truyền thống và phần mềm trí tuệ nhân tạo:

+ Chương trình truyền thống:

= Cấu trúc dữ liệu + thuật toán (thuật giải)

+ Chương trình trí tuệ nhân tạo:

= Cơ sở tri thức (knowledge base) + qui tắc suy diễn (inference engine).

* + - 1. **Cơ sở tri thức (Knowledge Base: KB)**

Cơ sở tri thức là tập hợp các tri thức liên quan đến vấn đề nào đó, được hệ thống và các thiết bị số lưu trữ dưới dạng luật (rule) hoặc các mẫu (pattern).

+ Các luật hoặc mẫu nêu trên được hình thành thông qua các kho dữ liệu (Datastore) được tích lũy qua một thời gian lâu dài và khai phá dữ liệu (Datamining).

+ Tập hợp tất cả các luật (rule) hay mẫu (pattern) trong một hệ thống trong một lĩnh vực thì được gọi là cơ sở tri thức.

+ Một số luật (rule) được hình thành quá lâu và ít được sao trùng thì sẽ bị loại ra khỏi cơ sở tri thức.

Cơ sở tri thức chứa các kiến thức được sử dụng để giải quyết các vấn đề trong trí tuệ nhân tạo.

* + - 1. **Hệ quản trị cơ sở tri thức (KBMS)**

Hệ cơ sở tri thức các sản phẩm phần mềm mà trong đó có cung cấp 2 nhóm công cụ và phương tiện:

- Nhóm thứ nhất: cung cấp công cụ để hình thành, duy trì và phát triển các cơ sở tri thức mà cụ thể các luật (rule) hoặc các mẫu (pattern) nhờ vào việc khai phá dữ liệu.

- Nhóm thứ hai: nhóm các công cụ hỗ trợ việc khai thác sử dụng cơ sở tri thức vào thực tiễn thông qua các qui luật suy diễn.

Cơ chế suy diễn là phương pháp vận dụng tri thức trong cơ sở tri thức để giải quyết vấn đề.

Cơ chế suy diễn là các công cụ, phương thức thuộc nhóm khai thác sử dụng cơ sở tri thức trong KBMS.

Một số hệ quản trị cơ sở tri thức : IPLV, auto LISP, PLANNER, PROLOG,…

### Vai trò AI

Là nhân tố trung tâm và gần như không thể thiếu trong các hệ thống hiện nay và tương lai.

Trí tuệ nhân tạo là yếu tố nền tảng mang tính chất bắt buộc của các hệ thống trong thời đại Cách mạng công nghiệp 4.0.

### Nền tảng kỹ thuật của AI

[1] Cơ sở tri thức

[2] Cơ chế: suy diễn, suy luận (Inference) và lâp luận = (Reasoning).

[3] Xử lý ngôn ngữ tự nhiên

[4] Học máy (Machine learning)

Các cơ sở lý thuyết :

[1] Lý thuyết

[2] Lý thuyết tìm kiếm may rủi (Ngẫu nhiên)

[3] Lý thuyết biểu diễn tri thức và hệ chuyên gia

[4] Các ngôn ngữ công cụ về Trí tuệ nhân tạo

[5] Lý thuyết về nhận dạng và xử lý tiếng nói (CV).

[6] Người máy (ROBOT)

[7] Tâm lý học xử lý thông tin ( Bio-inspired computing)

### Các lĩnh vực nghiên cứu và ứng dụng cơ bản của AI

+ Games

+ Tối ưu

+ Dự đoán, dự báo

## Giới thiệu về các bài toán (vấn đề) AI

### Một số khái niệm

* + - 1. **AI Problem (Bài toán AI = Vấn đề AI)**

Là một tình huống thực tế của cong người cần giải quyết thuộc một lĩnh vực nào đó, thông thường giải quyết dựa vào “trí khôn” con người.

Ví dụ: trò chơi cờ tướng, bài toán tìm đường đi tốt nhất, tư vấn pháp luật, dự đoán hành vi khách hàng…

* + - 1. **Các yếu tố cơ bản xác định bài toán AI**

Với một bài toán AI được xác định bởi 4 yếu tố cơ bản:

+ Tập hợp các trạng thái của bài toán.

VD: Tập hợp các trường hợp có thể xảy ra trên bàn cờ.

+ Trạng thái ban đầu (có thể là 1 hoặc tập hợp).

VD: Khi bắt đầu một trận cờ vua có 20 cách để đi bước đầu tiên.

+ Có mực tiêu đặt ra (thông qua hàm mục tiêu) mà kỹ thuật AI phải giải quyết cho bài toán đó.

VD: Tìm đường đi ngắn nhất, nhanh nhất…

+ Có cơ chế chuyển trạng thái. (từ trạng thái này sang trạng thái khác)

VD: Cờ vua = nước đi của 1 quân cờ trên bàn cờ tương ứng với trạng thái ban đầu.

Chú ý: Cơ chế chuyển trạng thái dựa trên quy tắc || quy luật || quy định || rằng buộc.

* + - 1. **(Hàm) Mục tiêu của Bài toán AI**

Mục tiêu của bài toán AI là một cơ chế để xác định trạng thái mực tiêu, nếu một trạng thái nào đó trong không gian trạng thái của bài toán thỏa mãn các điều kiện của mục tiêu thì được xem là một giải pháp của bài toán.

Mục tiêu của bài toán AI rất đa dạng và tùy thuộc vào từng bài toán cụ thể. Mục tiêu có 2 trường hợp:

+ Mục tiêu “định tính”: Được xác định nhờ một tính chất nào đó của trạng thái bài toán.

+ Mục tiêu “định lượng”: Được xác định bằng các giá trị số cụ thể, đo lường được, tính toán được.

Thông thường mục tiêu các bài toán AI là dạng “định lượng” và được xác định bởi hàm mục tiêu.

VD: Ta có một bài toán tìm đường đi:

Hàm mực tiêu có thể được xác định dự vào các tham số (parameter) sau đây:

+ x là khoản cách từ A đến B (km).

+ y là thời gian đi từ A đến B (h).

+ z là chi phí đi từ A đến B (VNĐ).

Trên thực tế bài toán trên ngoài các tham số nêu trên còn có thể có nhiều tham số khác trong hàm mục tiêu nhưng tùy vào tính hiệu quả của bài toán mà ta có thể bỏ qua không xét tới.

Khi cài đặt hàm mục tiêu cho bài toán nêu trên thì cần phải xác định phương pháp, cách thức tính toán xác định giá trị của các tham số. Ví dụ như bài toán trên thì các tham số được xác định tại vi trí A và vì vậy giá trị khoản cách tường được xử lý bằng cách cộng đồ dài của các quảng đường trên bàn trồ đồ dựa vào độ dài các quảng đường trên đường đi từ A đến B. (không thể áp dụng xác định đồng hồ của ohương tiện vì như vậy đã đên các tham số khác). Các tham số khác nhau x,y,z thường được xác định thông qua quá trình tự họ như hệ thống AI.

Với mỗi một đối số nêu trên, tùy vào mỗi trường hợp cụ thể, người ta có thể xác định mức độ quan trọng khác nhau, và mức độ quan trọng này thường được xác định bởi 1 giá trị só mà ta thường gọi là trọng số. Do đó mực tiêu của bài toán nêu trên có thể biểu diễn như sau:

F = a1\*x + a2\*y +a3\*z với x,y,z nêu trên.

a1, ,a2, a3:

Ngoài trường hợp mục tiêu tuyến tính còn có hàm mục tiêu “phi tuyến”, đặc biệt là hàm bậc 2 parabol có hệ số a âm, trong trường hợp này đỉnh parabol và ví trí mục tiêu. Hàm mục tiêu là cơ sở quan trọng để xác định yếu tố thứ 3 của bài toán AI.

### Tìm kiếm lời giải (searching) cho AI Problem

* + - 1. **Một số dạng bài toán (vấn đề) AI**

+ Bài toán một trạng thái đơn giản.

+ Bài toán đa trạng thái.

+ Bài toán ngẫu nhiên.

+ Bài toán thăm dò.

* + - 1. **Các cấp độ tìm kiếm lời giải cho bài toán (vấn đề) AI**

Có 3 cấp độ tìm kiếm lời giải cho bài toàn AI:

+ Có tìm được lời giải không ?

+ Lời giải tìm được có là lời giải tốt nhất không ?

+ Chi phí tìm lời giải thấp nhất.

## Biểu diễn một số bài toán AI

Một bài toán AI có thể được biểu diễn ở dạng đồ thị, ma trận, hoặc các biến số…

Trên thực tế, mỗi bài toán AI khác nhau sẽ có cách biểu diễn bài toán khác nhau. Các phương pháp biểu diễn nêu trên chỉ mang tính giới thiệu.

### Các bài toán Trò chơi

### Cây không gian trạng thái của Bài toán AI

* + - 1. **Định nghĩa**

+ Cây không gian trạng thái của bài toán AI thông thường không có sẵn, khi “duyệt” các Agent đến đâu thì cây không gian trạng thái “mở” nút đến đó cho đến khi gặp “lá” || “bí đường” thì quay lại (backtracking).

+ Cây không gian trạng thái hình thành tùy thuộc vào phương pháp duyệt của Agent.

+ Cây không gian trạng thái của bài toán AI là vô cùng nhiều nên kết quả tìm lời giải của bài toán AI chỉ là các kết quả tìm thấy trong không gian trạng thái “đã mở”, khó có thể tìm được tất cả.

VD: DFS,BFS …

Chú ý: Mặc dù có 2 phương pháp tìm kiếm chính DFS và BFS tuy nhiên khi áp dụng vào các bài toán AI, cần phải hiệu chỉnh hợp lý. Như ở DFS thì qua vài lần tìm kiếm nhất định thì dừng lại và backtrack lại và tùy thuộc vào từng bài toán, mực tiêu, trường hợp bài toán mà số lần tìm kiếm có thể khác nhau. Nếu độ sâu backtrack mà ta cho quá cạn, thì giải thuật có thể kết thúc sớm nhưng lời giải là không triệt để. Ngược lại nếu độ sâu backtrack ta cho quá sâu, thì giải thuật sẽ mất thời gian hơn nhưng lời giải sẽ triệt để hơn.

* + - 1. **Phương pháp giải quyết bài toán AI**
         1. **Phương pháp gia tăng**
         2. **Phương pháp trạng thái**
      2. **Sự bùng nổ không gian trạng thái**

Sự bùng nổ không gian trạng thái xảy ra khi chúng ta cố gắng tìm kiếm lời giải cho bài toán hay khi chúng ta chuyển trạng thái cho bài toán.

Ví dụ như trong cờ vua, ban đầu chỉ có 20 nước đi có thể đi nhưng khi càng về sau số nước đi có thể đi ở trong một trạng thái tăng dần dần.

## Các phương pháp biểu diễn tri thức

### Phân loại tri thức theo phương pháp biểu diễn

* + - 1. **Tri thức sự kiện**

+ Tri thức sự kiện: Tri thức sự kiện là một khẳng định về một sự kiện, hiện tượng hay một khái niệm nào đó trong một hoàn cảnh không gian hoặc thời gian nhất định.

* + - 1. **Tri thức mô tả**

+ Tri thức mô tả: Tri thức mô tả đối tượng, hiện tượng, vấn đề trong tự nhiên.

* + - 1. **Tri thức thủ tục**

Tri thức thủ tục: Tri thức biểu diễn cách giải quyết một vấn đề, quy trình xử lý các công việc, lịch trình tiến hành các thao tác…

* + - 1. **Tri thức mêta**

+ Tri thức mêta: Là một dạng tri thức thứ cấp, tham chiếu đến các loại tri thức khác.

* + - 1. **Tri thức có cấu trúc**

+ Tri thức có cấu trúc: Trí thức về quan hệ giữa các khái niệm, quan hệ giữa các đối tượng / có quy tắc hình thành các yếu tố Agent (token) = bao gồm những yếu tố cấu thành giống nhau, giá trị khác nhau.

### Phân loại tri thức theo nền tảng hình thành Cơ sở tri thức

* + - 1. **Tri thức hệ Heuristic**

- Tri thức Heuristic: Tri thức dựa trên kinh nghiệm dân gian, các quy luật số đông, xác suất thống kê, có tỷ lệ sai. Phải đáp ứng 2 yêu cầu cơ bản là quy luật hay kinh nghiệm đó phải đủ phổ biến và đủ dàn trải.

+ Tri thức Heuristic thường được coi là một “mẹo” nhằm dẫn dắt tiến trình lập luận. Vì vậy có nhiều rủi ro, đơn giản nên thường có tỷ lệ sai do đó phải có các rằng buộc để đảm bảo suy luận của hệ thống AI có giá trị.

* + - 1. **Tri thức hệ chuyên gia (ES = Expert System)**

- Tri thức chuyên gia: Tri thức được xây dựng và hình thành dựa trên cơ sở dựa trên:

+ Người có đầy đủ kỹ năng, kiến thức sâu (cả về luật và các sự kiện) về một lĩnh vực nào đó.

+ Người có thể làm những việc mà người khác ít khả năng làm được.

* + - 1. **So sánh Tri thức hệ Heuristic với tri thức hệ chuyên gia (ES)**

### Logic mệnh đề và Logic vị từ

Là cơ sở để xây dựng các cơ chế suy luận logic, cơ sở tri thức biểu diễn.

* + - 1. **Logic Mệnh đề**

- Mệnh đề là một khẳng định có thể nhận giá trị ĐÚNG hoặc SAI (chân trị).

VD: Hà Nội là thủ đô của Việt Nam, Mặt trời mọc ở phương đông.

- Đa số các mệnh đề ĐÚNG hay SAI tùy thuộc vào nhiều yếu tố ngoại cảnh.

VD: Theo thời điểm, chủ đề liên quan

- Các bài toán Ai chỉ xem xét mệnh đề 1 cách tương đối (xét chân trị của mệnh đề) = VD theo số đông cho đúng.

* + - 1. **Logic vị từ**

- Là mở rộng của logic mệnh đề dựa vào khái niệm vị từ và các khái niệm lượng từ.

- Lượng từ với mọi, lượng từ tồn tại và tồn tại duy nhất.

- Vị từ hàm ĐÚNG/SAI tùy thuộc vào biến liên quan.

- Logic vị từ có thể biểu diễn gần như toàn bộ các khái niệm và nguyên lý của các khoa học cơ bản.

- Logic mệnh đề và logic vi từ là cơ sở để tạo ra các luật dẫn suất.

* + - * 1. **Vị từ và lượng từ**

- Lượng từ:

+ Phổ dụng (∀)

+ Tồn tại (∃) hoặc tồn tại duy nhất (∃!)

* Vị từ: là hàm (đúng/sai) tùy vào hàm và biến liên quan
  + - * 1. **Biểu diễn Logic vị từ và chân trị của biểu diễn tri thức vị từ**

##### Mở rộng Logic vị từ

##### Logic vị từ bậc||cấp cao

Ta có: P(x) 🡪 Q(P2(y))

Q(P2(y)): vị từ bậc/cấp 2.

P(x) và P2(y) : vị từ bậc/cấp 1.

* Vị từ cấp 1 (FOP: First Order Predicate)
* Vị từ cấp 2 (SOP: Second Order Predicate)

Chú ý: Với các tri thức dạng “luật” (Rule) có thể thể biểu diễn toàn bộ dưới dạng FOP (vị từ cấp 1 = lượng từ, vị từ, phép nối) hoặc cũng có thể biểu hiện dạng SOP (vị từ cấp 2) hoặc cấp cao hơn.

Với các tri thức dạng “luật” (Rule) có thể biểu diễn FOP với nhiều kết quả biểu diễn khác nhau.

##### Logic tình huống

Logic vị từ có bổ sung thêm các tình huống

VD: *Thông tin (Input)*

*[Biến tình huống = biến thời gian = lượng hóa = biểu diễn định lượng]*

*“Tàu XYZ cập bến lúc 11g”*

*“Bốc dỡ hàng kết thúc lúc 18g”*

*“Anh A chỉ huy bốc dỡ hàng”*

*“Tàu XYZ rời cảng lúc 19g”*

*Hỏi (Goal): “17g A đang ở đâu ?”*

PC (phần mềm truyền thống) sẽ không thể trả lời cho câu hỏi này nhưng con người có thể trả lời tương tự như PC ( phần mềm AI: Logic vị từ) sẽ biết cách trả lời.

*Biến tình huống = biến thời gian = lượng hóa = biểu diễn định lượng*

*11h 🡪 18h 🡪 19h 11h 🡪 17h 🡪 18h*

**KL: Cài đặt bằng Logic vị từ “tình huống” với thông tin trên.**

##### Logic trạng thái (modal)

Liên quan đến Fuzzy logic (Logic “mờ”):

VD Wiki: “Anh Bảo ở phòng khách và phòng bếp” = “Mờ”

Với phát biểu trên có 3 trạng thái xảy ra:

Anh Bảo ở trong phòng khách (Không liên quan đến phòng bếp).

Anh Bảo ở trong phòng bếp (Không liên quan phòng khách).

Anh Bảo ở giữa cửa nối 2 phòng bếp và phòng khách = Logic trạng thái = “Mờ” nêu trên chấp nhận được nghĩa là có thể cài đặt được vào hệ thống AI dưới dạng Logic trạng thái. => Suy luận (lập luận) Logic cho bài toán AI.

##### Logic xác suất và Logic khả xuất

Liên quan đến Fuzzy logic (Logic “mờ”):

Tiếp tục sử dụng ví dụ của 2.3.2.1.3:

Anh Bảo ở giữa cửa nối 2 phòng bếp và phòng khách = Logic trạng thái.

Trên thực tế:

Anh Bảo sẽ có 30% cơ thể ở bên phía phòng khách.

Anh Bảo sẽ có 40% cơ thể ở bên phía phòng bếp.

Anh Bảo sẽ có phần còn lại (30%) cơ thể ở giữa cửa nối 2 phòng bếp và khách.

* Logic xác suất (ướm tương đối => thực tế không đo lường)

= Logic diễn đạt một sự kiện, có kèm theo tỷ suất xảy ra là ?%

Logic khả xuất là trường hợp cá biệt của logic xác suất. Thay vì “xác suất” thông thường thì xét đến “xác suất xảy ra” nghĩa là ?% sẽ xảy ra sự kiện đó. Thường biểu diễn :

VD: p 🡪 0.8 q (Độ tin cậy 0.8) : 80% xảy ra.

Tức là khi có sự kiến p thì 80% xảy ra sự kiện q.

VD: *“Trời nhiều mây thì 80% là chiều nay mưa”* là biểu diễn Logic vị từ (Logic khả xuất).

*“Mờ”* ở đây là không rõ như thế nào là *“nhiều mây”* => minh họa một cách giải quyết: Chụp ảnh bầu tời 🡪 Dùng phần mềm xử lý ảnh để xác định tỷ lệ % mây trên bầu trời => Biểu diễn dạng Logic vị từ (Logic xác suất) => Xác định “ngưỡng” (dưới = tối thiểu) = “Nhiều mây”.

### Biểu diễn tri thức bằng Frame (Cấu trúc = "Khung")

Frame là phương pháp biểu diễn tri thức có cấu trúc dữ liệu, chứa tất cả tri thức liên quan đến một đối tượng cụ thể nào đó = hình thức biểu diễn tri thức dựa trên cơ sở lập trình truyền thống (Dùng CTDL = định nghĩa Agent = Token).

Frame tương tự KDL “đối tượng” (Class) trong lập trình hướng đối tượng.

Frmae thường được dùng để biểu diễn những:

+ Tri thức “chuẩn”.

+ Những tri thức dựa trên kinh nghiệm.

+ Các tri thức đã được hiểu biết cặn kẽ.

### Suy diễn tri thức bằng luật dẫn xuất

#### Khái niệm

Suy diễn, lập luận dựa trên cơ sở tri thức ban đầu đã có/đã biết cộng với các luật dẫn xuất (Rules) => Xác định (suy diễn ra || lập luận để xác định ra) các sự kiện mới, các phát biểu mới nhầm bổ sung vào cơ sở tri thức.

Các sự kiện mới, các phát biểu mới là dạng Logic vị từ cho ta một chân trị gọi là tri thức dẫn suất.

Nền tảng luật (qui tắc) suy diễn:

+ Suy diễn tiến: Cho trước các tiền đề 🡪 Xác định ra các kết quả đạt được từ các tiền đề đó.

+ Suy diễn ngược: Cho trước các kết quả cần đạt được 🡪 Xác định tiền đề để có được kết quả có.

VD1: Ta có cơ sở tri thức “Pháp luật”

* Người dân yêu cầu một hoạt động nào đó 🡪 Dựa vào pháp luật xác định ra các điều kiện để người dân thực hiên các hoạt động đó 🡪 Suy diễn ngược.
* Nêu ra các điều kiện để người dân được phép thực hiện một hoạt động nào đó 🡪 Suy diễn tiến.

VD2: Ta có cơ sở tri thức “Từ điển Anh-Việt”

- Câu từ tiếng Anh 🡪 tiếng Việt.

- Câu từ tiếng Việt 🡪 tiếng Anh.

#### Luật dẫn xuất trong Cơ sở tri thức: Suy diễn tiến

Suy luận bằng đồ thị AND/OR dùng để “suy diễn lùi || ngược”.

Là dạng độ thị có hướng AND/OR trong đó:

+ Nút là một sự kiện.

+ Nút/cạnh: Suy luận 1 chiều từ sự kiện này 🡪 sự kiện khác (tiến).

+ Cung/cạnh từ 1 nút biểu diễn cho các quan hệ.

### Biểu diễn suy luận bằng đồ thị AND/OR

Suy luận bằng đồ thị AND/OR dùng để “suy diễn lùi || ngược”.

Là dạng độ thị có hướng AND/OR trong đó:

+ Nút là một sự kiện.

+ Nút/cạnh: Suy luận 1 chiều từ sự kiện này 🡪 sự kiện khác (tiến).

+ Cung/cạnh từ 1 nút biểu diễn cho các quan hệ.

### Các biểu diễn tương đương trong Logic mệnh đề và Logic vị từ

* Các tương đương được sử dụng thường xuyên trong quá trình biến đổi một biểu thức vị từ từ dạng này sang dạng khác.
* Khả năng biến đổi tương đương trên máy tính có thể được làm tự động nhờ vào các cơ chế suy diễn.
* Các tương đương :

Trong các tương đương sau A,B,C là các mệnh đề.

CHÚ Ý: Các phép toán

Phủ định Một ngôi (KHÔNG)

Hội Hai ngôi (VÀ)

Tuyển Hai ngôi (HOẶC)

=> Suy ra Hai ngôi

<=> Tương đương Hai ngôi

* Dạng phủ định kép

A <=> A

* Dạng suy ra

A => TRUE <=> TRUE

A => FALSE <=> A

TRUE => A <=> A

A => A <=> TRUE

FALSE => A <=> TRUE

* Dạng hấp thụ

A (A B) <=> A

A (A B) <=> A

A (A B) <=> A B

A (A B) <=> A B

* Dạng De Morgan

(A B) = A B

(A B) = A B

VD: **while** (chưa hết danh sách **và** chưa tìm thấy phần tử C) **do** tiếp tụ duyệt danh sách **repeat** tiếp tục danh sách **until** hết danh sách hoặc tìm thấy phần tử C trong danh sách.

* Dạng khác

A => B <=> A B

(A => B) <=> A B

(A => A) <=> A A => FALSE

* + Phép hội và tuyển có khả năng kết hợp.

A (B C) = (A B) C

* + Phép hội và tuyển có khả năng hoán vị

A B = B A

* + Phép hội có khả năng phân phối trên tuyển

A ( B C) <=> (A B) (A C)

* + Phép tuyển có năng phân phối trên hội

A ( B C) <=> (A B) (A C)

Luật suy diễn được áp dung để phát triển các ứng dụng có khả năng suy luận. Suy luận là hoạt động thường xuyên của con người để hiểu các lý lẽ, kiểm chứng, phán đoán các vấn đề.

Ứng dụng trong nhiều bài toán AI dạng:

+ “Tư vấn” như Tư vấn phát luật,…

+ “Phán đoán” như Thời tiết, chấn đoán y khoa, tư vi (game), game,…

+ “Chăm sóc khách hàng”:…

Một số luật được thường được sử dụng ngoài các luật cơ bản:

* Luật Modus Ponens (MP)

A, A => B <=> B

*“Nếu mệnh A đúng và A => B đúng thì B đúng”*

* Luật Modus Tollens (MT)

A => B, B <=> A

*“Nếu A => B đúng và B sai thì A sai”*

* Luật hội

A, B <=> A B

*“Nếu A đúng và B đúng thì A B đúng”*

* Luật đơn giản

A B <=> A,B

*“Nếu A B đúng thì A đúng, tất nhiên B cũng đúng”*

* Luật Cộng || Tuyển || Hợp

A <=> A B

*“Nếu A đúng thì A B đúng”*

* Luật tam đoạn luận tuyển (phản chứng)

A B, A <=> B

*“Nếu A B đúng và A sai thì B đúng”*

* Luật tam đoạn luận giả thiết (kéo theo)

A => B, B => C <=> A => C

*“Nếu A => B đúng và B => C cũng đúng thì A => C đúng”*

### Xử lý ngôn ngữ tự nhiên (Natural Language Processing = NLP): Voice Assistant

+ Thường dùng luật dẫn xuất nêu trên để biểu diễn các tri thức dạng “ký tự”, dạng “định tính” vào cơ sở tri thức để lưu trữ trên máy tính để phục vụ cho các phần mềm AI lập luận và giải quyết các vấn đề trong tự nhiên.

+ Khi phần mềm AI nhận (Input) các dữ liệu đầu vào thì phần mềm AI sẽ truy vấn tri thức theo phương pháp sao trùng mẫu (Pattern) 🡪 tìm ra các Pattern có độ trùng gần nhất (Tính sai số min) các tiền đề 🡪 nhằm đưa ra các đề xuất, kết luận tương ứng. Vì vậy đòi hỏi phải có các phương pháp biểu diễn các tri thức dạng “kí tự” 🡪 Sử dụng logic vị từ (mệnh đề) với các luật dẫn xuất nêu trên.

## Các phương pháp tìm kiếm lời giải của bài toán AI

### Giới thiệu

* + - 1. **Các tiêu chí đánh giá**

+ Tính hoàn thành: chiến lược có bảo đảm tìm thấy giải pháp cho bài toán ?

+ Độ phức tạp thời gian: chiến lược mất bao lâu để tìm ra một giải pháp ?

+ Độ phức tạp không gian: chiến lược đó cần bao nhiêu dung lượng bộ nhớ để thực hiện việc tìm kiếm.

+ Tính tối ưu: Chiến lược có tìm được giải pháp có chất lượng cao nhất khi có một số các giải pháp khác nhau.

* + - 1. **Phân loại các phương pháp (chiến lược) tìm kiếm lời giải**

Có nhiều phân loại các chiến lược tìm kiếm, phổ biến chia làm 2 loại:

+ Uninformed Search (Blind Search): tìm kiếm mù, tìm kiếm khi không có đủ thông tin.

+ Informed Search (Heuristic Search): tìm kiếm theo kinh nghiệm, tìm kiếm khi có đầy đủ thông tin.

### Các phương pháp tìm kiếm mù (Blind Search) = tìm kiếm không có thêm thông tin (uninformed search)

* + - 1. **Depth-First Search (DFS): Tìm kiếm theo chiều sâu**

Là giải thuật để duyệt cấu trúc cây [đồ thị] không gian trạng thái để tìm lời giải của bài toán AI, trong đó nó lần lượt tìm theo chiều sâu của từng nhánh cây [đồ thị] trước, cho đến khi KHÔNG tìm được lời giải [đáp án] trên nhánh đó [còn gọi là “bế tắc” || “bí” || hết đường || cùng đường] thì nó quay lui [back\_tracking] để tìm trên nhánh kế tiếp: cho đến khi:

+ Tìm được lời giải [đáp án] OR

+ Duyệt hết toàn bộ (OR giới hạn phạm vi) không gian cây [đồ thị] được xem xét mà vẫn không tim được lời giải [đáp án].

* + - 1. **Breadth-First Search (BFS): Tìm kiếm theo chiều rộng || ngang**

Là giải thuật để duyệt cấu trúc cây [đồ thị] không gian trạng thái để tìm lời giải của bài toán AI, trong đó nó lần lượt tìm theo chiều ngang [rộng] của từng mức [cấp] cây [đồ thị] trước, cho đến khi KHÔNG tìm được lời giải [đáp án] trên mức [cấp] đó [còn gọi là “bế tắc” || “bí” || hết đường || cùng đường] thì nó quay lui [back\_tracking] để tìm trên mức [cấp] kế tiếp: cho đến khi:

+ Tìm được lời giải [đáp án] OR

+ Duyệt hết toàn bộ (OR giới hạn phạm vi) không gian cây [đồ thị] được xem xét mà vẫn không tim được lời giải [đáp án].

* + - 1. **Uniform Cost Search (UCS): Tìm kiếm với chi phí ĐỀU (không thay đổi) = chi phí cực tiểu [Áp dụng thuật toán Dijkstra]**

Là một cách duyệt cây dùng cho việc duyệt hay tìm kiếm trên cây có trọng lượng chi phí. Thuật toán sẽ phát triển các nút chưa xét có chi phí thấp nhất – các nút được xét theo thứ tự chi phí tăng dần. Khi đồ thị có chi phí ở mỗi bước là như nhau thì thuật toán trở thành phương pháp tìm kiếm theo chiều rồng.

* + - 1. **Interative Deepening Search (IDS) = Tìm kiếm sâu dần = cải tiến DFS**

Thuật toán tìm kiếm kiếm sâu dần là 1 thuật toán duyệt hoặc tìm kiếm trên cây hoặc đồ thị. Thuật toán được đưa ra để khắc phục điểm yêu của thuật toán tìm kiếm giới hạn độ sâu[DLS](https://bloglaptrinh2016.wordpress.com/2016/06/09/dls-thuat-toan-tim-kiem-gioi-han-do-sau/). Đó là khi mà tất cả các lời giải nằm ở độ sâu lớn hơn giới hạn độ sâu thì giải thuật [DLS](https://bloglaptrinh2016.wordpress.com/2016/06/09/dls-thuat-toan-tim-kiem-gioi-han-do-sau/) sẽ thất bại.

* + - 1. **Depth-Limited Search (DLS) = Tìm kiếm giới hạn độ sâu = cải tiến DFS**

Thuật toán tìm kiếm có giới hạn độ sâu (DLS) hay depth-limited search algorithm là một thuật toán phát triển các nút chưa xét các theo chiều sâu nhưng có giới hạn mức để tránh đi vào những con đường không mang lại kết quả tốt như trong thuật toán tìm kiếm sâu dần.

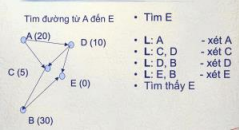
### Các phương pháp tìm kiếm theo kinh nghiệm (Heuristic Search) = tìm kiếm với thông tin bổ sung (Informed Search) = Cải tiến BFS

* + - 1. **Best first: BestBFS**

Tìm kiếm theo lựa chọn tốt nhất là một [thuật toán tìm kiếm](https://vi.wikipedia.org/wiki/Gi%E1%BA%A3i_thu%E1%BA%ADt_t%C3%ACm_ki%E1%BA%BFm) tối ưu hóa [tìm kiếm theo chiều rộng](https://vi.wikipedia.org/wiki/T%C3%ACm_ki%E1%BA%BFm_theo_chi%E1%BB%81u_r%E1%BB%99ng) bằng cách mở rộng nút hứa hẹn nhất được chọn theo một quy tắc nào đó.

Để có được hiệu quả về thời gian chạy cho việc chọn ra ứng cử viên tốt nhất cho việc mở rộng, người ta thường dùng một [hàng đợi ưu tiên](https://vi.wikipedia.org/w/index.php?title=H%C3%A0ng_%C4%91%E1%BB%A3i_%C6%B0u_ti%C3%AAn&action=edit&redlink=1) để cài đặt cấu trúc dữ liệu lưu trữ các lựa chọn hiện hành.

Ví dụ 1: tại mỗi nút có hàm Heuristic[hf] = tính chi phí để từ đó đến nút đích (E)  Vd: hf(A) = 20 : chi phí từ A đến E ước tính (phỏng đoán) là 20  hf(E) = 0 : đương nhiên, vì E là nút đích

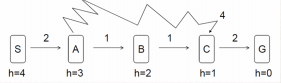


. Chọn node kế tiếp được đánh giá là tốt nhất

. Giá trị của hàm đánh giá tại 1 điểm được ghi bên cạnh: A(20), C(5) . Nghĩa là nó đánh giá dựa vào 1 tính chất nào đó mà

 AE = 20, CE = 5, EE = 0 (tất nhiên)

. Kết quả: G-BFS: A -> C -> D -> E

Ví dụ 2: G-BFS hiệu quả nhưng chưa chắc thật sự tối ưu, xét vd sau . Hình dưới, g(n) là chi phí thực tế, h(n) là ước lượng theo hàm đánh giá. 

. Hàm Heuristic ước tính: h

 h(S->G) = 4, h(A->G) = 3, h(B->G) = 2, h(C->G) =1

. Hàm cho phí thực tế đi qua các nút: g

 g(S->A) = 2, g(A->C) = 4, g(A->B) = 1, g(B->C) = 1, g(C->G) = 2.  . Nếu chỉ dựa theo đánh giá h(n): G-BFS: S->A->C->G (cost 8) . Trong khi đi thẳng : S->A->B->C->G (cost 6)

* + - 1. **A\* và SMA\* (A\* với bộ nhớ giới hạn): tìm kiếm trên đồi thị (Graph Search)**

- Thuật toán A\* tìm một đường đi từ một [nút](https://vi.wikipedia.org/w/index.php?title=N%C3%BAt_(l%C3%BD_thuy%E1%BA%BFt_%C4%91%E1%BB%93_th%E1%BB%8B)&action=edit&redlink=1) khởi đầu tới một nút đích cho trước (hoặc tới một nút thỏa mãn một điều kiện đích). Thuật toán này sử dụng một "đánh giá heuristic" để xếp loại từng nút theo ước lượng về tuyến đường tốt nhất đi qua nút đó. Thuật toán này duyệt các nút theo thứ tự của đánh giá heuristic này.

+ Kết hợp tính hiệu quả (nhanh) của G-BFS, và tính tối ưu (luôn tìm ra lời giải đúng) của UCS (Uniform Cost Search) + Hàm đánh giá:

f(n)= g(n) + h(n).

\*g(n) – Chi phí từ node hiện tại tới node n (tính hàm g)

\*h(n) – ước lượng khoảng cách min từ node n -> đích

+ UCS chỉ dùng g(n) nên chậm, G-BFS chỉ dựa h(n) nên không tối ưu  -> f(n) = tối ưu + hiệu quả

+ h(n), g(n) dùng độ đo Euclidean distance (Đường thẳng):

 d(A[xa, ya], B[xb, yb]) = **Sqrt[(xa-xb)2 + (ya-yb)2]**

- SMA\* là thuật toán tìm kiếm đường đi ngắn nhất dựa trên thuật toán A\*. SMA\* có lợi thế hơn A\* là giới hạn bộ nhớ.

* + - 1. **Hill-Climbing Search (Tìm kiếm tối ưu cục bộ || leo đồi|| lep núi) = HCS**

Giải thuật Hill Climbing là một kỹ thuật tối ưu toán học thuộc họ tìm kiếm cục bộ. Nó thực hiện tìm một trạng thái tốt hơn trạng thái hiện tại để mở rộng. Để biết trạng thái tiếp theo nào là lớn hơn, nó dùng một hàm H để xác định trạng thái nào là tốt nhất. Đi từ node hiện tại (chưa tốt = thấp hơn) – “trèo lên” nút tốt hơn (“cao hơn”) lân cận.

* + - 1. **Chiến lựơc tìm kiếm đối kháng (Board Game) với Giải thuật MiniMax||MinMax**

Giải thuật Minimax là một thuật toán đệ quy lựa chọn bước đi kế tiếp trong một trò chơi có hai người bằng cách định giá trị cho các Node trên cây trò chơi sau đó tìm Node có giá trị phù hợp để đi bước tiếp theo.

Một phiên bản của giải thuật áp dụng cho các trò chơi như [tic-tac-toe](https://vi.wikipedia.org/wiki/Tic-tac-toe), khi mà mỗi người chơi có thể thắng, thua, hoặc hòa. Nếu người chơi A *có thể* thắng trong 1 nước đi, thì "nước đi tốt nhất" chính là nước đi để dẫn đến kết quả thắng đó. Nếu người B biết rằng có một nước đi mà dẫn đến tình huống người A có thể thắng ngay ở nước đi tiếp theo, trong khi nước đi khác thì sẽ dẫn đến tình huống mà người chơi A chỉ có thể, tốt nhất, là hòa thì nước đi tốt nhất của người B chính là nước đi sau.

Ta sẽ nắm rõ, thế nào là một nước đi "tốt nhất". Giải thuật Minimax giúp tìm ra nước đi tốt nhất, bằng cách đi ngược từ cuối trò chơi trở về đầu. Tại mỗi bước, nó sẽ ước định rằng người A đang cố gắng tối đa hóa cơ hội thắng của A khi đến phiên anh ta, còn ở nước đi kế tiếp thì người chơi B cố gắng để tối thiểu hóa cơ hội thắng của người A (nghĩa là tối đa hóa cơ hội thắng của B).

### Lập trình Logic (Logic Programming) phát triển cơ chế lập luận logic

Để lập trình Logic (Logic Programming) với python chúng ta cần các thư viện sau:

+ Thư viẹn kanren: Cho phép chúng ta hiện logic như các quy tắc và sự kiện và đơn giản hóa việc tạo mã logic, cho phép chúng ta thêm các tính chất như giao hoán, kết hợp vào quy tắc.

+ Thư viện SymPy: Là một thư viện python cho toán học tượng trưng. Nó gần như là một hệ thống đại số máy tính đầy đủ tính năng.

# GIỚI THIỆU VỀ MÔ HÌNH BÀI TOÁN AI: PHÁT TRIỂN HỆ THỐNG TRỢ LÝ ẢO HỖ TRỢ PHÁT HIỆN VÀ NHẬN DIỆN KHUÔN MẶT NGƯỜI TRONG HÌNH

## Mô tả bài toán AI

Trợ lý ảo là một đề tài rất thú vị trong AI, một trợ lý ảo có thể rất đơn giản chỉ dựa những câu lệnh điều kiện hay có thể trở nên rất phước tạp với nhiều công nghệ được tích hợp như trợ lý ảo của Apple là Siri với khả năng lập luận logic, có khả năng thực hiện các nhiệm vụ cơ bản như báo giờ, báo ngày, hẹn giờ, trả lời như một chatbot. Chính vì thế em quyết định chọn đề tài làm đề tài cuối kỳ của em.

Cùng với trợ lý ảo thì nhận diện và phát hiện luôn là một đề tài được quan tâm trong lập trình trí tuệ nhân tạo như nhận diện hay phát hiện xe, máy bay, con vật, cây… Được quan tâm nhiều nhất chính là nhận diện khuôn mặt con người.

## Các yếu tố xác định bài toán AI

Trợ lý ảo của đề tài xoay quanh vấn đề phát hiện và nhận diện thế nên bài toán của đề tài cũng xoay quanh nó:

- Trạng thái của bài toán: Toàn bộ khung hình được chia ra từ bức hình đưa vào có thể hiểu là chúng ta đang rời rạc hóa bức hình.

- Mục tiêu đặt ra: Từ bức hình được đưa vào, hãy phát hiện xem trong bức hình ấy có khuôn mặt của con người hay không và nhận diện xem người đó là ai.

- Trạng thái ban đầu của bài toán: Khung hình đầu tiên được chia ra từ bức hình đưa vào.

- Cơ chế chuyển trạng thái: Khung hình được duyệt dần qua bức hình và tăng kích thước dần cho tới khi đạt kích thước tối đa và dừng.

# BIỂU DIỄN BÀI TOÁN AI: PHÁT TRIỂN HỆ THỐNG TRỢ LÝ ẢO HỖ TRỢ PHÁT HIỆN VÀ NHẬN DIỆN KHUÔN MẶT NGƯỜI TRONG HÌNH

## Giới thiệu

Một trợ lý ảo sẽ nhận được yêu cầu dưới dạng văn bản hay giọng nói vì vậy chúng ta cần những hàm, chức năng để nhận văn bản, nhận giọng nói và xử lý chúng sao cho trợ lý có thể hiểu và đưa ra lời giải, vì thế chúng ta cũng sẽ cần những hàm, thư viện để có thể xử lý những yêu cầu mà người dùng đưa vào đồng thời kết hợp với xử lý ảnh để phát hiện và nhận diện.

Để nhận diện và phát hiên sẽ được giải quyết bằng thuật toán Haar Cascade Classifier và phương pháp HOG.

+ Thuật toán Haar Cascade Classifier được đánh giá là có tốc độ cao trong việc phát hiện khuôn mặt nhưng sẽ hay có những phát hiện sai lầm.

+ Phương pháp HOG, HOG là viết tắt của Histogram of Oriented Gradient – một loại “feature descriptor”. Mục đích của “feature descriptor” là để trừu tượng hóa đối tượng bằng cách trích xuất những đặc trưng của đối tượng (ở đây là khuôn mặt) và bỏ đi những thông tin không hữu ích.

## Lập trình xử lý ngôn ngữ tự nhiên

r = sr.Recognizer()

def speak(text):

tts = gTTS(text=text, lang='vi')

filename = 'voice.mp3'

tts.save(filename)

playsound.playsound(filename)

os.remove(filename)

while True:

with sr.Microphone() as source:

print("Tinh chỉnh âm thanh nền trong 3 giây")

r.adjust\_for\_ambient\_noise(source,3)

print("Thu âm trong 3 giây")

ThuAm = r.record(source,3)

try:

text = r.recognize\_google(ThuAm,language='vi')

print(text)

except:

text =""

text = text.lower()

if text =="":

AI\_Assitant = "Xin lỗi tôi nghe không rõ"

speak(AI\_Assitant)

elif "xin chào" in text:

AI\_Assitant = "Xin chào"

speak(AI\_Assitant)

elif "hôm nay là ngày mấy" in text:

time = datetime.now()

AI\_Assitant = time.strftime("Hôm nay là ngày %d tháng %m năm %y")

speak(AI\_Assitant)

elif "mấy giờ" in text:

time = datetime.now()

AI\_Assitant = time.strftime("%H:%M:%S")

speak(AI\_Assitant)

elif "bật webcam" in text:

speak("Bật webcam, muốn tắt vui lòng bấm ESC")

FacialRecognitionWithWebcam()

elif "mở thư mục" in text:

speak("Mở thư mục và chọn file hình ảnh muốn nhận diện")

path = filedialog.askopenfilenames()[0]

resultName = FacialRecognitionWithImage(path)

if resultName == "":

speak("Không nhận diện được người trong ảnh hoặc người trong ảnh không có trong tập dữ liệu")

else:

result = "Tên người trong ảnh là "+ resultName

speak(result)

elif "nhận diện khuôn mặt qua hình ảnh" in text:

AI\_Assitant = "Hãy nhập tên hình ảnh và đảm bảo hình ảnh có trong thư mục"

speak(AI\_Assitant)

text = input("Tên hình ảnh: ") #ex: RDJ.jpg

resultName = FacialRecognitionWithImage(text)

if resultName == "":

speak("Không nhận diện được người trong ảnh hoặc người trong ảnh không có trong tập dữ liệu")

else:

result = "Tên người trong ảnh là "+ resultName

speak(result)

else:

AI\_Assitant ="Bạn vui lòng nói lại"

speak(AI\_Assitant)

# PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI CHO BÀI TOÁN AI: PHÁT TRIỂN HỆ THỐNG TRỢ LÝ ẢO HỖ TRỞ PHÁT HIỆN VÀ NHẬN DIỆN KHUÔN MẶT NGƯỜI TRONG HÌNH

## Giới thiệu

Việc trợ lý ảo có thể phát hiện và nhận diện thì chúng ta cần sử dụng những thư viện đã có sẳn để xử lý giọng nói nhận vào là speech\_recognition và gtts.

Phát hiện và nhận diện thì ngày nay đã có rất nhiều phương pháp phát hiện và nhận diện khuôn mặt khác nhau với đặc tính khác nhau, các phương pháp nhận diện có thể đựa chia dựa trên 4 nhóm:

+ Dựa vào đặc trưng (Feature-based):

+ Dựa vào biểu hiện (Appearance-based):

+ Dựa vào tri thức (Knowledge-based):

+ Dựa vào so sánh khớp (Templatematching):

## Xác định phương pháp tìm kiếm lời giải cho bài toán AI: Phát triển hệ thống hỗ trợ phát hiện và nhận diện khuôn mặt người trong hình

Đề tài của em sẽ sử dụng 2 thư viện chính là speech\_recognition và gtts làm nền tảng để phát triển hệ thống trợ lý ảo và thuật toán Haar Cascade Classifier và phương pháp HOG để hỗ trợ việc nhận diện và phát hiện:

* speech\_recognition sẽ cung cấp cho em các công cụ, hàm cần thiết để nhận diện và xử lý giọng nói, chuyện giọng nói sang dạng văn bản.
* gtts sẽ là công cụ chính trong việc chuyển từ dạng văn bảng sang giọng nói với nhiều âm ngữ khác nhau.
* Thuật toán Haar Cascade Classifier sẽ là thuật toán được sử dụng chính trong việc phát hiện còn phương pháp HOG sẽ sử dụng trong việc nhận diện.

## Mô tả phương pháp tìm kiếm lới giải cho bài toán AI:Phát triển hệ thống trợ lý ảo hỗ trợ phát hiện và nhận diện khuôn mặt người trong hình

Để lập trình một trợ lý ảo trước tiên ta phải cho nó khả năng nghe giọng nói:

-> Để xử lý vấn đề này ta sẽ sử dụng thư viện speech\_recognition để cho trợ lý ảo khả năng nghe:

Bước 1: Cài đặt thư viện speech\_recognition với lệnh pip install speech\_recognition hay conda install speech\_recognition.

Bước 2: Sử dụng microphone làm input với hàm speech\_recognition.microphone().

Bước 3: Tinh chỉnh âm thanh nền để lọc tạp âm như gió với hàm adjust\_for\_ambient\_noise().

Bước 4: Chuyển âm thanh nghe được thành văn bản.

Bước 5: Lập trình các câu lệnh để trợ lý ảo thực thi khi nhận được lệnh.

-> Để xử lý khả năng tạo ra giọng nói ta sẽ dùng thư viện gtts (Google Translate) để cho trợ lý ảo khả năng nói:

Bước 1: Cài đặt thư viện gtts với lệnh pip install gtts hay conda install gtts

Bước 2: lưu văn bản cần trợ lý ảo đọc và chọn chất giọng là Việt Nam.

Bước 3: Tạo một file mp3 dựa vào bước 2.

Bước 4: Chạy file mp3 vừa được tạo.

-> Để xử lý khả năng phát hiện và nhận diện cần thuật toán Haar Cascade Classifier và phương pháp HOG.

**1. Thuật toán Haar Cascade Classifier**

Thuật toán Haar Cascade Classifier được thực hiện qua 4 giai đoạn:

Giai đoạn 1: Thu thập các đặc tính Haar.

Giai đoạn 2: Tạo các Integral Images

Giai đoạn 3: Adaboost Training

Giai đoạn 4: Áp dụng Cascading Classifier

**2. Phương pháp HOG**

Phương pháp HOG cũng có thể dùng để phát hiện khuôn mặt như thuật toán Haar Cascade Classifier nhưng trong đề tài của em chỉ quan tâm đến khả năng nhận diện của nó, quan tâm đến khả năng so sánh và nhận diện xem khuôn mặt đó thuộc về ai mà thuật toán Haar Cascade Classifier không có khả năng làm.

Bước 1: Tính toán hướng dốc (Gradiant).

Bước 2: Chỉnh hướng khuôn mặt.

Bước 3: Mã hóa khuôn mặt.

Qua các bước trên chúng ta đã có thể so sánh khuôn mặt được đưa vào với những khuôn mặt có sẵn trong cơ sở dữ liệu của chúng ta và từ đó đưa ra kết quả người trong hình là ai.

## Cài đặt (lập trình) phương pháp tìm kiếm lới giải cho bài toán AI:Phát triển hệ thống hỗ trợ phát hiện và nhận diện khuôn mặt người trong hình

def FacialRecognitionWithWebcam():

    ## duyệt từng hình ảnh có trong thư mục và encode các đặc tính của nó sau đó đưa vào một list để lưu trữ

    path = 'HinhAnh'

    listofImg = []

    listofName = []

    listofImgInDir = os.listdir(path)

    for img in listofImgInDir: #duyệt từng hình ảnh trong thư mục và lưu lại vào list hình ảnh và tên

        currentImg =  cv2.imread(f'{path}/{img}')

        listofImg.append(currentImg)

        listofName.append(os.path.splitext(img)[0])

    def Encoding(listofImg): #duyệt từng hình ảnh trong list và encode nó

        listEncode = []

        for img in listofImg:

            imgRGB = cv2.cvtColor(img,cv2.COLOR\_BGR2RGB)

            encode = face\_recognition.face\_encodings(imgRGB)[0]

            listEncode.append(encode)

        return listEncode

    listKnowEncode = Encoding(listofImg)

    ##

    #Nạp dữ liệu đã được train sẵn để nhận diện khuôn mặt trực diện (haar cascade algorithm)

    face\_data = cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml')

    #Classifier là một detector, để nhận diện khuôn mặt

    #lấy hình ảnh từ webcam

    webcam = cv2.VideoCapture(0)

    while True:

        #đọc khung ảnh hiện tại từ webcam

        place\_holder,frame = webcam.read()

        #tạo một bản sao nhỏ của khung hình đang đọc

        frameS = cv2.resize(frame,(0,0),None,0.25,0.25)

        #chuyển tấm hình sang dạng trắng đen

        grayscaled\_img = cv2.cvtColor(frame,cv2.COLOR\_BGR2GRAY)

        #chuyển bản sao nhỏ sang dạng rgb

        rgb\_img = cv2.cvtColor(frameS,cv2.COLOR\_BGR2RGB)

        #phát hiện khuôn mặt và mắt

        face\_cordianate = face\_data.detectMultiScale(grayscaled\_img)

        #encode các đặc tính ở khuôn mặt được đưa vào

        frameSCurrentFrame = face\_recognition.face\_locations(rgb\_img)

        encodingCurrentFrame = face\_recognition.face\_encodings(rgb\_img,frameSCurrentFrame)

        #vẽ hình chữ nhật xung quanh khuôn mặt

        for (x,y,w,h) in face\_cordianate:

            cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,0),2)

        #so sánh khuôn mặt trong webcam với các khuôn mặt có sẵn

        for encodingCurrent, faceCorCurrent in zip(encodingCurrentFrame,frameSCurrentFrame):

            matches = face\_recognition.compare\_faces(listKnowEncode,encodingCurrent) #so sánh

            faceDis = face\_recognition.face\_distance(listKnowEncode,encodingCurrent) #tính toán sự khác biệt giữa bức ảnh đc đưa vào và ảnh trong thư mục

            matchIndex = np.argmin(faceDis)  #phải lấy khuôn mặt có độ lệch nhỏ nhất (gần giống với khuôn mặt trong thư mục nhất)

            if matches[matchIndex]:

                x,y,w,h =  faceCorCurrent

                x,y,w,h = x\*4,y\*4,w\*4,h\*4

                Name = listofName[matchIndex].upper()

                cv2.putText(frame,Name,(h+6,w-6),cv2.FONT\_ITALIC,1,(0,255,0),2)

                if(Name != ""):

                    result = "Tên người trong ảnh là "+ Name

                    threading.Thread(speak(result)).start()

        #xem hình ảnh

        cv2.imshow('Face Detection',frame)

        key = cv2.waitKey(1) #tạm dừng chương trình

        if key == 27: #nhấn ESC để dừng chương trình

            break

    cv2.destroyAllWindows()

def FacialRecognitionWithImage(text):

    Name = ''

    ## duyệt từng hình ảnh có trong thư mục và encode các đặc tính của nó sau đó đưa vào một list để lưu trữ

    path = 'HinhAnh'

    listofImg = []

    listofName = []

    listofImgInDir = os.listdir(path)

    for img in listofImgInDir: #duyệt từng hình ảnh trong thư mục và lưu lại vào list hình ảnh và tên

        currentImg =  cv2.imread(f'{path}/{img}')

        listofImg.append(currentImg)

        listofName.append(os.path.splitext(img)[0])

    def Encoding(listofImg): #duyệt từng hình ảnh trong list và encode nó

        listEncode = []

        for img in listofImg:

            imgRGB = cv2.cvtColor(img,cv2.COLOR\_BGR2RGB)

            encode = face\_recognition.face\_encodings(imgRGB)[0]

            listEncode.append(encode)

        return listEncode

    listKnowEncode = Encoding(listofImg)

    ##

    #Nạp dữ liệu đã được train sẵn để nhận diện khuôn mặt trực diện (haar cascade algorithm)

    face\_data = cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml')

    #Classifier là một detector, để nhận diện khuôn mặt

    #lấy hình ảnh từ webcam

    img = cv2.imread(f"{text}")

    #chuyển tấm hình sang dạng trắng đen

    grayscaled\_img = cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

    #chuyển bản sao nhỏ sang dạng rgb

    rgb\_img = cv2.cvtColor(img,cv2.COLOR\_BGR2RGB)

    #phát hiện khuôn mặt và mắt

    face\_cordianate = face\_data.detectMultiScale(grayscaled\_img)

    #encode các đặc tính ở khuôn mặt được đưa vào

    frameSCurrentFrame = face\_recognition.face\_locations(rgb\_img)

    encodingCurrentFrame = face\_recognition.face\_encodings(rgb\_img,frameSCurrentFrame)

    #vẽ hình chữ nhật xung quanh khuôn mặt

    for (x,y,w,h) in face\_cordianate:

        cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)

        #so sánh khuôn mặt trong webcam với các khuôn mặt có sẵn

    for encodingCurrent, faceCorCurrent in zip(encodingCurrentFrame,frameSCurrentFrame):

        matches = face\_recognition.compare\_faces(listKnowEncode,encodingCurrent) #so sánh

        faceDis = face\_recognition.face\_distance(listKnowEncode,encodingCurrent) #tính toán sự khác biệt giữa bức ảnh đc đưa vào và ảnh trong thư mục

        matchIndex = np.argmin(faceDis)  #phải lấy khuôn mặt có độ lệch nhỏ nhất (gần giống với khuôn mặt trong thư mục nhất)

        if matches[matchIndex]:

            x,y,w,h =  faceCorCurrent

            x,y,w,h = x\*4,y\*4,w\*4,h\*4

            Name = listofName[matchIndex].upper()

            cv2.putText(img,Name,(h+6,w-6),cv2.FONT\_ITALIC,1,(0,255,0),2)

    return Name

# GIỚI THIỆU SẢN PHẨM CỦA ĐỀ TÀI AI: PHÁT TRIỂN HỆ THỐNG TRỢ LÝ ẢO HỖ TRỢ PHÁT HIỆN VÀ NHẬN DIỆN KHUÔN MẶT NGƯỜI TRONG HÌNH

## Giới thiệu sản phẩm bài toán AI: Phát triển hệ thống trợ lý hỗ trợ ảo phát hiện và nhận diện khuôn mặt người trong hình

Thông tin sản phẩm:

+ Tên gọi: AI\_Assitance.py

+ Công cụ: Visual Studio Code

+ Ngôn ngữ: Python 3.9.1

+ Thư viện: speech\_recognition, gtts, OpenCV, os, face\_recognition (Cần Cmake và dlib), , numpy.

+ Thuật toán sử dụng Haar Cascade Classifier và phương pháp HOG

Code của sản phẩm:

import speech\_recognition as sr

from gtts import gTTS

import os

import playsound

import cv2

import face\_recognition

import os

import numpy as np

import threading

import tkinter as tk

from tkinter import filedialog

root = tk.Tk()

root.withdraw()

##

def FacialRecognitionWithWebcam():

    ## duyệt từng hình ảnh có trong thư mục và encode các đặc tính của nó sau đó đưa vào một list để lưu trữ

    path = 'HinhAnh'

    listofImg = []

    listofName = []

    listofImgInDir = os.listdir(path)

    for img in listofImgInDir: #duyệt từng hình ảnh trong thư mục và lưu lại vào list hình ảnh và tên

        currentImg =  cv2.imread(f'{path}/{img}')

        listofImg.append(currentImg)

        listofName.append(os.path.splitext(img)[0])

    def Encoding(listofImg): #duyệt từng hình ảnh trong list và encode nó

        listEncode = []

        for img in listofImg:

            imgRGB = cv2.cvtColor(img,cv2.COLOR\_BGR2RGB)

            encode = face\_recognition.face\_encodings(imgRGB)[0]

            listEncode.append(encode)

        return listEncode

    listKnowEncode = Encoding(listofImg)

    ##

    #Nạp dữ liệu đã được train sẵn để nhận diện khuôn mặt trực diện (haar cascade algorithm)

    face\_data = cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml')

    #Classifier là một detector, để nhận diện khuôn mặt

    #lấy hình ảnh từ webcam

    webcam = cv2.VideoCapture(0)

    while True:

        #đọc khung ảnh hiện tại từ webcam

        place\_holder,frame = webcam.read()

        #tạo một bản sao nhỏ của khung hình đang đọc

        frameS = cv2.resize(frame,(0,0),None,0.25,0.25)

        #chuyển tấm hình sang dạng trắng đen

        grayscaled\_img = cv2.cvtColor(frame,cv2.COLOR\_BGR2GRAY)

        #chuyển bản sao nhỏ sang dạng rgb

        rgb\_img = cv2.cvtColor(frameS,cv2.COLOR\_BGR2RGB)

        #phát hiện khuôn mặt và mắt

        face\_cordianate = face\_data.detectMultiScale(grayscaled\_img)

        #encode các đặc tính ở khuôn mặt được đưa vào

        frameSCurrentFrame = face\_recognition.face\_locations(rgb\_img)

        encodingCurrentFrame = face\_recognition.face\_encodings(rgb\_img,frameSCurrentFrame)

        #vẽ hình chữ nhật xung quanh khuôn mặt

        for (x,y,w,h) in face\_cordianate:

            cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,0),2)

        #so sánh khuôn mặt trong webcam với các khuôn mặt có sẵn

        for encodingCurrent, faceCorCurrent in zip(encodingCurrentFrame,frameSCurrentFrame):

            matches = face\_recognition.compare\_faces(listKnowEncode,encodingCurrent) #so sánh

            faceDis = face\_recognition.face\_distance(listKnowEncode,encodingCurrent) #tính toán sự khác biệt giữa bức ảnh đc đưa vào và ảnh trong thư mục

            matchIndex = np.argmin(faceDis)  #phải lấy khuôn mặt có độ lệch nhỏ nhất (gần giống với khuôn mặt trong thư mục nhất)

            if matches[matchIndex]:

                x,y,w,h =  faceCorCurrent

                x,y,w,h = x\*4,y\*4,w\*4,h\*4

                Name = listofName[matchIndex].upper()

                cv2.putText(frame,Name,(h+6,w-6),cv2.FONT\_ITALIC,1,(0,255,0),2)

                if(Name != ""):

                    result = "Tên người trong ảnh là "+ Name

                    threading.Thread(speak(result)).start()

        #xem hình ảnh

        cv2.imshow('Face Detection',frame)

        key = cv2.waitKey(1) #tạm dừng chương trình

        if key == 27: #nhấn ESC để dừng chương trình

            break

    cv2.destroyAllWindows()

def FacialRecognitionWithImage(text):

    Name = ''

    ## duyệt từng hình ảnh có trong thư mục và encode các đặc tính của nó sau đó đưa vào một list để lưu trữ

    path = 'HinhAnh'

    listofImg = []

    listofName = []

    listofImgInDir = os.listdir(path)

    for img in listofImgInDir: #duyệt từng hình ảnh trong thư mục và lưu lại vào list hình ảnh và tên

        currentImg =  cv2.imread(f'{path}/{img}')

        listofImg.append(currentImg)

        listofName.append(os.path.splitext(img)[0])

    def Encoding(listofImg): #duyệt từng hình ảnh trong list và encode nó

        listEncode = []

        for img in listofImg:

            imgRGB = cv2.cvtColor(img,cv2.COLOR\_BGR2RGB)

            encode = face\_recognition.face\_encodings(imgRGB)[0]

            listEncode.append(encode)

        return listEncode

    listKnowEncode = Encoding(listofImg)

    ##

    #Nạp dữ liệu đã được train sẵn để nhận diện khuôn mặt trực diện (haar cascade algorithm)

    face\_data = cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml')

    #Classifier là một detector, để nhận diện khuôn mặt

    #lấy hình ảnh từ webcam

    img = cv2.imread(f"{text}")

    #chuyển tấm hình sang dạng trắng đen

    grayscaled\_img = cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

    #chuyển bản sao nhỏ sang dạng rgb

    rgb\_img = cv2.cvtColor(img,cv2.COLOR\_BGR2RGB)

    #phát hiện khuôn mặt và mắt

    face\_cordianate = face\_data.detectMultiScale(grayscaled\_img)

    #encode các đặc tính ở khuôn mặt được đưa vào

    frameSCurrentFrame = face\_recognition.face\_locations(rgb\_img)

    encodingCurrentFrame = face\_recognition.face\_encodings(rgb\_img,frameSCurrentFrame)

    #vẽ hình chữ nhật xung quanh khuôn mặt

    for (x,y,w,h) in face\_cordianate:

        cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)

        #so sánh khuôn mặt trong webcam với các khuôn mặt có sẵn

    for encodingCurrent, faceCorCurrent in zip(encodingCurrentFrame,frameSCurrentFrame):

        matches = face\_recognition.compare\_faces(listKnowEncode,encodingCurrent) #so sánh

        faceDis = face\_recognition.face\_distance(listKnowEncode,encodingCurrent) #tính toán sự khác biệt giữa bức ảnh đc đưa vào và ảnh trong thư mục

        matchIndex = np.argmin(faceDis)  #phải lấy khuôn mặt có độ lệch nhỏ nhất (gần giống với khuôn mặt trong thư mục nhất)

        if matches[matchIndex]:

            x,y,w,h =  faceCorCurrent

            x,y,w,h = x\*4,y\*4,w\*4,h\*4

            Name = listofName[matchIndex].upper()

            cv2.putText(img,Name,(h+6,w-6),cv2.FONT\_ITALIC,1,(0,255,0),2)

    return Name

        #xem hình ảnh

def ShowWebcam():

    webcam = cv2.VideoCapture(0,cv2.CAP\_DSHOW)

    while True:

        place\_holder,frame = webcam.read()

        cv2.imshow('webcam',frame)

        key = cv2.waitKey(1) #tạm dừng chương trình

        if key == 27: #nhấn ESC để dừng chương trình

            break

    cv2.destroyAllWindows()

##

r = sr.Recognizer()

def speak(text):

tts = gTTS(text=text, lang='vi')

filename = 'voice.mp3'

tts.save(filename)

playsound.playsound(filename)

os.remove(filename)

while True:

with sr.Microphone() as source:

print("Tinh chỉnh âm thanh nền trong 3 giây")

r.adjust\_for\_ambient\_noise(source,3)

print("Thu âm trong 3 giây")

ThuAm = r.record(source,3)

try:

text = r.recognize\_google(ThuAm,language='vi')

print(text)

except:

text =""

text = text.lower()

if text =="":

AI\_Assitant = "Xin lỗi tôi nghe không rõ"

speak(AI\_Assitant)

elif "xin chào" in text:

AI\_Assitant = "Xin chào"

speak(AI\_Assitant)

elif "hôm nay là ngày mấy" in text:

time = datetime.now()

AI\_Assitant = time.strftime("Hôm nay là ngày %d tháng %m năm %y")

speak(AI\_Assitant)

elif "mấy giờ" in text:

time = datetime.now()

AI\_Assitant = time.strftime("%H:%M:%S")

speak(AI\_Assitant)

elif "bật webcam" in text:

speak("Bật webcam, muốn tắt vui lòng bấm ESC")

FacialRecognitionWithWebcam()

elif "mở thư mục" in text:

speak("Mở thư mục và chọn file hình ảnh muốn nhận diện")

path = filedialog.askopenfilenames()[0]

resultName = FacialRecognitionWithImage(path)

if resultName == "":

speak("Không nhận diện được người trong ảnh hoặc người trong ảnh không có trong tập dữ liệu")

else:

result = "Tên người trong ảnh là "+ resultName

speak(result)

elif "nhận diện khuôn mặt qua hình ảnh" in text:

AI\_Assitant = "Hãy nhập tên hình ảnh và đảm bảo hình ảnh có trong thư mục"

speak(AI\_Assitant)

text = input("Tên hình ảnh: ") #ex: RDJ.jpg

resultName = FacialRecognitionWithImage(text)

if resultName == "":

speak("Không nhận diện được người trong ảnh hoặc người trong ảnh không có trong tập dữ liệu")

else:

result = "Tên người trong ảnh là "+ resultName

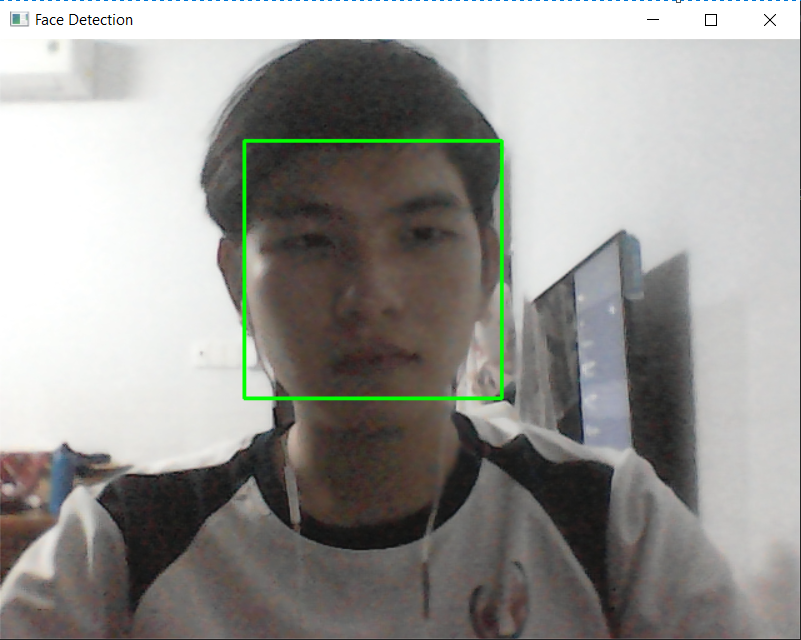
speak(result)

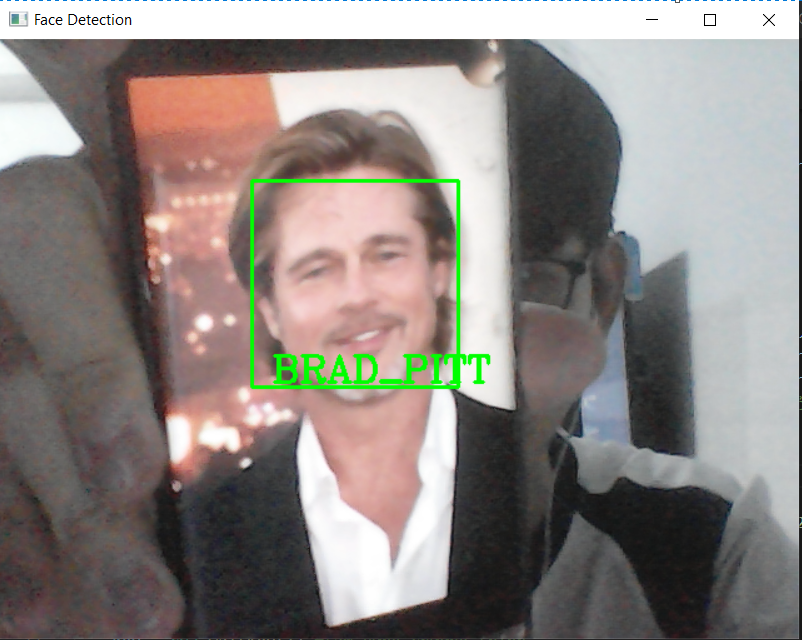
else:

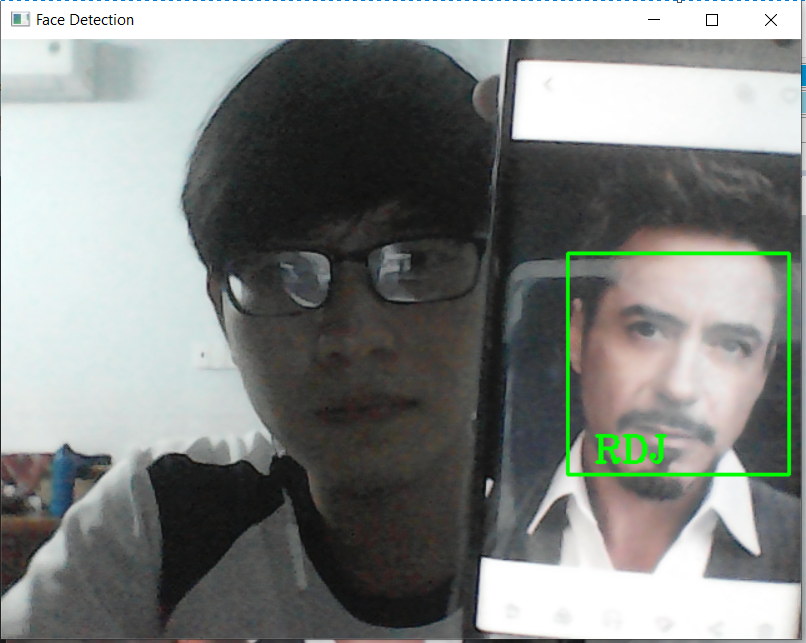
AI\_Assitant ="Bạn vui lòng nói lại"

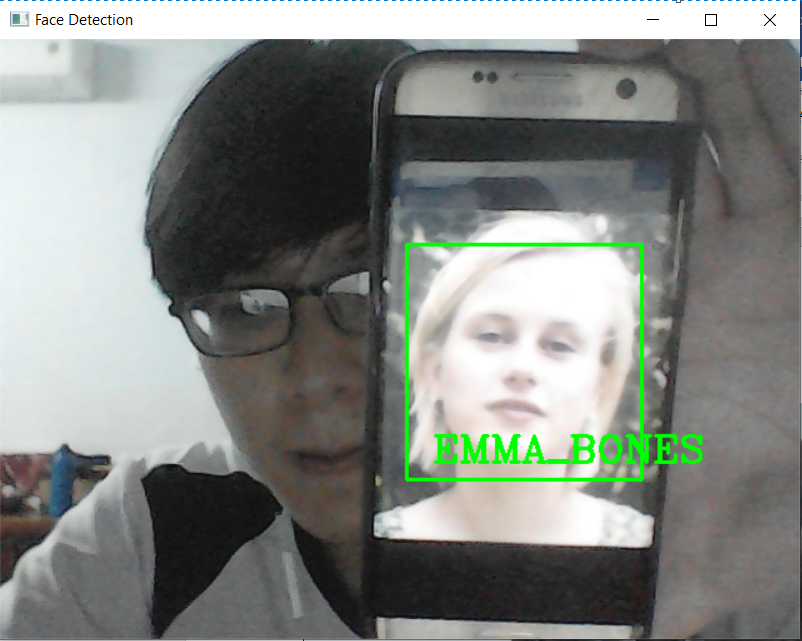
speak(AI\_Assitant)

## Kết quả thực nghiệm sản phẩm bài toán AI: Phát triển hệ thống trợ lý ảo hỗ trợ phát hiện và nhận diện khuôn mặt người trong hình









## Nhận xét và đánh giá về sản phẩm bài toán AI: Phát triển hệ thống trợ lý ảo hỗ trợ phát hiện và nhận diện khuôn mặt người trong hình

Do đề tài sử dụng những thư viện, hàm có sẵn nên code khá ngắn, đơn giản và dễ hiểu. Tuy nhiên, trong quá trình cài đặt cũng gặp một vài khó khăn trong quá trình cài đặt thư viện face\_recognition.

Cũng chính vì sử dụng những thư viện và hàm có sẵn nên sản phẩm không mấy chuyên nghiệp và cũng như không có khả năng can thiệp vào thuật toán của thư viện đã viết sẵn, hạn chế khả năng mở rộng cũng như cải tiến sản phẩm.

Sản phầm đã đáp ứng được những yêu cầu cơ bản đã được đề ra trong bài toán tuy nhiên vẫn còn rất nhiều chỗ để cải thiện cũng như cải tiến thêm.

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN CỦA ĐỀ TÀI

## Kết luận

### Những kết quả đạt được

Thông qua việc thực hiện đề tài em đã:

+ Biết cách tạo một trợ lý ảo sử dụng xử lý ngôn ngữ tự nhiên, từ giọng nói sang văn bản và từ văn bản sang giọng nói.

+ Biết cách viết một chương trình Python cơ bản phát hiện và nhận diện khuôn mặt.

+ Biết và hiểu thêm về thuật toán Haar Cascade Classifier và phương pháp HOG.

+ Biết thêm về lý thuyết cơ bản của chủ đề phát hiện và nhận diện khuôn mặt.

### Hạn chế

Chức năng của trợ lý ảo còn khá hạn chế, chỉ có thể thực hiện những yêu cầu đơn giản như mở webcam lên để nhận diện, mở thư mục chọn hình ảnh để nhận diện, gõ tên ảnh từ chat.

Sản phẩm vẫn phát hiện sai đâu là khuôn mặt con người, nguyên nhân đầu tiên là do hạn chế và tính chất của thuật toán Haar Cascade Classifier cùng với chất lượng webcam kém của máy tính.

Về khả năng nhận diện thì sản phẩm chỉ có thể nhận diện dựa trên nguồn dữ liệu cho trước, không thể tự động cập nhật dữ liệu mới nếu có.

## Hướng phát triển

### Hướng khắc phục các hạn chế

Lập trình thêm nhiều chức năng cho trợ lý ảo, định nghĩa thêm nhiều câu lệnh để trợ lý ảo có thể hiểu và thực hiện nhiều câu lệnh hơn.

Để cải thiện khả năng phát hiện của thuật toán Haar Cascade Classifier thì chúng ta có thể tiếp tục nạp thêm hình ảnh khuông mặt của con người để tiếp tục rèn luyện cho thuật toán.

Có thể lập trình một hàm cơ bản cho đề tài mỗi khi nhận diện được một khuôn mặt mới và không có trong dữ liệu có sẵn thì mã hóa và đưa vào trong dữ liệu, việc đặt tên có thể theo một quy định nào đấy nếu chưa biết tên người trong ảnh.

### Hướng mở rộng đề tài

- Kết hợp lập trình Logic, tăng thêm nhiều chức năng với khả năng hiểu được nhiều câu lệnh hơn, kết hợp thêm nhiều ngôn ngữ vào trợ lý ảo.

- Dụa vào khả năng nhận diện có thể thêm vào khả năng dự đoán cảm xúc, hành vi dựa vào biểu cảm trên khuôn mặt.

- Có thể tích hợp phần mềm phát hiện và nhận diện vào các hệ thống chấm công, quản lý của công ty.

# DANH MỤC TÀI LIỆU THAM KHẢO

1. Võ Xuân Thể (2021), *Bài giảng học phần Trí tuệ nhân tạo*, TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM (lưu hành nội bộ).
2. [Aditya Mittal](https://adityamittal307.medium.com/?source=post_page-----38210e57970d--------------------------------) (2020), *Haar Cascades, Explained*: <https://medium.com/analytics-vidhya/haar-cascades-explained-38210e57970d>
3. Adam Geitgey (2016), *Machine Learning is Fun! Part 4: Modern Face Recognition with Deep Learning*: <https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78>
4. Divyansh Dwivedi (2018), *Face Detection For Beginners*: https://towardsdatascience.com/face-detection-for-beginners-e58e8f21aad9
5. Trần Xuân Trang (2018), *Lập trình logic Python với ví dụ*: <https://helpex.vn/article/lap-trinh-logic-python-voi-vi-du-5c6648d9ae03f60128765bb0>

# CÁC PHỤ LỤC

# ĐÁNH GIÁ KẾT QUẢ HỌC TẬP HỌC PHẦN

ĐỒ ÁN HP THAY CHO BÀI THI KẾT THÚC HỌC PHẦN

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Hình thức KT** | **Nội dung** | **Chuẩn đầu ra đánh giá** | **Trình độ năng lực** | **Phương pháp đánh giá** | **Công cụ đánh giá** | **Tỉ lệ (%)** |
| **BÀI TẬP LẬP TRÌNH** | | | | | |  |
| BL#1 | Bài tập lập trình theo từng Buổi thực hành = tính bình quân  Kiểm tra viết | G1.1 | 3 | Bài tập  Kiểm tra | Bài tập  Kiểm tra | 50 |
| G2.1 | 3 |
| G3.1  G3.2 | 4  3 |
| G4.1 | 5 |
| **ĐỒ ÁN HỌC PHẦN (Project)** | | | | | |  |
| ĐA#1 | Phát triển một hệ thống AI đơn giản:  + Đề tài cá nhân  + Xây dựng và thực nghiệm giải thuật AI theo bài toán cá nhân  => báo cáo vào ngày thi. | G1.1 | 3 | Báo cáo, Demo và thuyết trình | Rubric | 50 |
| G2.1 | 3 |
| G3.1 | 4 |
| G3.2  G4.1 | 3  5 |
| G2.1 | 4 |
| G3.1 | 4 |
| G3.2  G4.1 | 3  5 |