**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ ĐÔNG Á**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI TẬP LỚN**

**HỌC PHẦN: XỬ LÝ ẢNH VÀ THỊ GIÁC MÁY TÍNH**

**Đề tài số 21: XÂY DỰNG HỆ THỐNG TRÍCH XUẤT**

**THÔNG TIN TỪ THẺ SINH VIÊN**

**Giảng viên hướng dẫn: Lương Thị Hồng Lan**

|  |  |  |  |
| --- | --- | --- | --- |
| **TT** | **Mã sinh viên** | **Sinh viên thực hiện** | **Lớp hành chính** |
| **1** | 20210069 | Phạm Văn Phúc | DCCNTT 12.10.1 |
| **2** | 20210148 | Nguyễn Bảo Lâm | DCCNTT 12.10.1 |

**Bắc Ninh, năm 2024**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ ĐÔNG Á**

**KHOA CÔNG NGHỆ THÔNG TIN**

**BÀI TẬP LỚN**

**HỌC PHẦN: XỬ LÝ ẢNH VÀ THỊ GIÁC MÁY TÍNH**

**Đề tài số 21: XÂY DỰNG HỆ THỐNG TRÍCH XUẤT**

**THÔNG TIN TỪ THẺ SINH VIÊN**

**Giảng viên hướng dẫn: Lương Thị Hồng Lan**

|  |  |  |  |
| --- | --- | --- | --- |
| **TT** | **Mã sinh viên** | **Sinh viên thực hiện** | **Lớp hành chính** |
| **1** | 20210069 | Phạm Văn Phúc | DCCNTT 12.10.1 |
| **2** | 20210148 | Nguyễn Bảo Lâm | DCCNTT 12.10.1 |

|  |  |
| --- | --- |
|  |  |

**Bắc Ninh, năm 2024**

|  |  |
| --- | --- |
| TRƯỜNG ĐẠI HỌC CÔNG NGHỆ ĐÔNG Á  **KHOA CÔNG NGHỆ THÔNG TIN** | **KỲ THI KẾT THÚC HỌC PHẦN**  **HỌC KỲ 1, NĂM HỌC 2024** – **2025** |

|  |  |
| --- | --- |
| **PHIẾU CHẤM THI BÀI TẬP LỚN KẾT THÚC HỌC PHẦN**  **Đề tài số 21 : Xây dựng hệ thống trích xuất thông tin từ thẻ sinh viên**  **Tên học phần: Xử lý ảnh và thị giác máy tính**  **Lớp Tín chỉ:**  **XATGMT.03.K12.01.LH.C04.1\_LT** | |
| **Cán bộ chấm thi 1**  *(Ký và ghi rõ họ tên)*  **Lương Thị Hồng Lan** | **Cán bộ chấm thi 2**  *(Ký và ghi rõ họ tên)* |

| **TT** | **TIÊU CHÍ** | **THANG ĐIỂM** | **Phạm Văn Phúc** | **Nguyễn Bảo Lâm** |
| --- | --- | --- | --- | --- |
| 20210069 | 20210148 |
| **1** | **Nội dung báo cáo trên Word đầy đủ** | **3.5** |  |  |
| 1.1 | Có bố cục rõ ràng (mục lục, phần mở đầu, nội dung chính, kết luận). | 0,5 |  |  |
| 1.2 | Nội dung phân tích rõ ràng, logic. | 0,5 |  |  |
| 1.3 | Có dẫn chứng, số liệu minh họa đầy đủ. | 0,5 |  |  |
| 1.4 | Ngôn ngữ và trình bày chuẩn, không lỗi chính tả. | 0,5 |  |  |
| 1.5 | Có trích dẫn tài liệu tham khảo đúng quy cách. | 0,5 |  |  |
| 1.6 | Được trình bày chuyên nghiệp (canh lề, font chữ, khoảng cách dòng hợp lý). | 0,5 |  |  |
| 1.7 | Tài liệu đầy đủ, bám sát yêu cầu của đề bài. | 0,5 |  |  |
| **2** | **Nội dung thuyết trình đầy đủ** | **1.0** |  |  |
| 2.1 | Trình bày tự tin, phát âm rõ ràng, mạch lạc. | 0,5 |  |  |
| 2.2 | Nội dung thuyết trình đúng trọng tâm, không lan man. | 0,5 |  |  |
| **3** | **Slides báo cáo đầy đủ nội dung + Hỏi đáp** | **3.0** |  |  |
| 3.1 | Slides có bố cục rõ ràng (mở đầu, nội dung, kết luận). | 0,5 |  |  |
| 3.2 | Thiết kế slides đẹp, chuyên nghiệp (màu sắc, hình ảnh minh họa). | 0,5 |  |  |
| 3.3 | Nội dung trên slides ngắn gọn, dễ hiểu, súc tích. | 0,5 |  |  |
| 3.4 | Nội dung slides phù hợp với nội dung báo cáo. | 0,5 |  |  |
| 3.5 | Trả lời câu hỏi đầy đủ, chính xác. | 0,5 |  |  |
| 3.6 | Trả lời câu hỏi tự tin, thuyết phục. | 0,5 |  |  |
| **4** | **Code đầy đủ** | **2.5** |  |  |
| 1.1 | Code được trình bày rõ ràng, có chú thích đầy đủ. | 0,5 |  |  |
| 1.2 | Code chạy đúng, không lỗi. | 0,5 |  |  |
| 1.3 | Code tối ưu, không dư thừa. | 0,5 |  |  |
| 1.4 | Đáp ứng đầy đủ các yêu cầu chức năng theo đề bài. | 0,5 |  |  |
| 1.5 | Có tính sáng tạo hoặc cải thiện so với yêu cầu. | 0,5 |  |  |
| **TỔNG ĐIỂM BẰNG SỐ:** | | **10** |  |  |
| **TỔNG ĐIỂM BẰNG CHỮ:** | | *Mười tròn* |  |  |

**Mục Lục**

[CHƯƠNG I. TỔNG QUAN LÝ THUYẾT 7](#_Toc184464437)

[1.1. Tổng quan về thị giác máy tính 7](#_Toc184464438)

[1.1.1. Thị giác máy tính là gì ? 7](#_Toc184464439)

[1.1.2. Cách thị giác máy tính hoạt động 8](#_Toc184464440)

[1.1.3. Lịch sử phát triển của thị giác máy tính 10](#_Toc184464441)

[1.2. Tổng quan về nhận dạng đối tượng 11](#_Toc184464442)

[1.2.1. Nhận dạng đối tượng là gì? 11](#_Toc184464443)

[1.2.2. Các phương pháp sử dụng để nhận dạng đối tượng 11](#_Toc184464444)

[1.3. Tổng quan về học máy 12](#_Toc184464445)

[1.3.1. Học máy là gì? 12](#_Toc184464446)

[1.3.2. Các phương pháp học máy 12](#_Toc184464447)

[1.4. Ngôn ngữ, thư viện và công cụ sử dụng trong bài 13](#_Toc184464448)

[1.4.1. Ngôn ngữ Python: 13](#_Toc184464449)

[1.4.2. Các thư viện sử dụng 14](#_Toc184464450)

[CHƯƠNG 2: PHÁT BIỂU VÀ PHÂN TÍCH BÀI TOÁN 18](#_Toc184464451)

[2.1. Phát biểu bài toán 18](#_Toc184464452)

[2.1.1. Đặt vấn đề 18](#_Toc184464453)

[2.1.2. Mô tả bài toán 19](#_Toc184464454)

[2.2. Phân tích hệ thống 20](#_Toc184464455)

[2.2.1. Các kĩ thuật sử dụng 20](#_Toc184464456)

[2.2.2. Mô hình tổng quan hệ thống 24](#_Toc184464457)

[CHƯƠNG 3: KẾT QUẢ THỰC NGHIỆM 27](#_Toc184464458)

[3.1. Dữ liệu 27](#_Toc184464459)

[3.2. Kết quả thực nghiệm 27](#_Toc184464460)

[3.2.1. Thông tin thẻ sinh viên của “ Phạm Văn Phúc “ 27](#_Toc184464461)

[3.2.2. Thông tin thẻ sinh viên của “ Nguyễn Đức Minh Hiếu” 28](#_Toc184464462)

[3.3.3. Thông tin thẻ sinh viên của “ Hoàng Thái Gia ” 28](#_Toc184464463)

[KẾT LUẬN 29](#_Toc184464464)

[TÀI LIỆU THAM KHẢO 31](#_Toc184464465)

# **CHƯƠNG I. TỔNG QUAN LÝ THUYẾT**

## **1.1. Tổng quan về thị giác máy tính**

* + 1. **Thị giác máy tính là gì ?**

[1] Thị giác máy tính là một nhánh của trí tuệ nhân tạo (AI) ứng dụng machine learning và mạng nơ-ron để giúp máy tính và hệ thống hiểu và trích xuất thông tin ý nghĩa từ hình ảnh kỹ thuật số, video, và các dạng đầu vào trực quan khác. Nó cho phép máy móc đưa ra khuyến nghị hoặc thực hiện hành động khi phát hiện lỗi hoặc sự cố.

Mặc dù thị giác máy tính hoạt động theo cách tương tự như thị giác con người, nhưng con người có lợi thế nhờ bối cảnh phong phú mà họ đã tích lũy suốt đời. Điều này giúp con người phân biệt các đối tượng, đánh giá khoảng cách, nhận biết chuyển động, hoặc phát hiện các vấn đề bất thường một cách tự nhiên.

Thị giác máy tính dạy máy móc thực hiện các chức năng này trong khoảng thời gian ngắn hơn nhiều, dựa trên dữ liệu từ máy ảnh và các thuật toán, thay vì sử dụng võng mạc, dây thần kinh thị giác hay vỏ não thị giác. Nhờ khả năng phân tích hàng nghìn sản phẩm hoặc quy trình mỗi phút, phát hiện những lỗi nhỏ mà con người không thể nhìn thấy, hệ thống thị giác máy tính vượt trội hơn hẳn so với khả năng của con người trong nhiều ứng dụng cụ thể.

* + 1. **Cách thị giác máy tính hoạt động**

[1] Thị giác máy tính là một lĩnh vực cần lượng dữ liệu khổng lồ để hoạt động hiệu quả. Hệ thống thị giác máy tính thực hiện phân tích dữ liệu liên tục để nhận ra sự khác biệt và cuối cùng là nhận dạng được các hình ảnh hoặc đối tượng cụ thể. Ví dụ, để đào tạo máy tính nhận diện lốp xe ô tô, cần cung cấp cho nó một lượng lớn hình ảnh lốp xe ở các góc độ và tình trạng khác nhau, cùng với các hình ảnh liên quan để máy học cách phân biệt giữa lốp xe và các vật thể khác, đặc biệt là để nhận diện lốp xe đạt chuẩn và không có khuyết tật.

Để thực hiện quá trình này, thị giác máy tính dựa vào hai công nghệ quan trọng: deep learning (học sâu) và mạng nơ-ron tích chập (Convolutional Neural Networks - CNN).

* *Deep Learning và Machine Learning trong Thị giác Máy tính*

[2] Machine learning sử dụng các mô hình thuật toán để dạy máy tính cách tự học và hiểu bối cảnh của dữ liệu trực quan. Thay vì lập trình cụ thể từng quy tắc để máy nhận diện hình ảnh, các mô hình này cho phép máy tính tự học thông qua dữ liệu được cung cấp. Khi một tập dữ liệu đủ lớn được đưa vào, hệ thống sẽ "xem" và học cách phân biệt các đặc điểm giữa một hình ảnh với hình ảnh khác. Quá trình này không chỉ giúp máy tính nhận diện hình ảnh mà còn tăng khả năng đưa ra dự đoán chính xác về những gì nó nhìn thấy, dựa trên bối cảnh và đặc điểm đã học.

Điều này tương tự như cách con người học hỏi: khi nhìn thấy nhiều loại lốp xe khác nhau, chúng ta sẽ học cách phân biệt giữa lốp tốt và lốp hỏng thông qua quan sát các chi tiết như hình dạng, kết cấu, và các khuyết điểm nhỏ.

* *Mạng nơ-ron tích chập (CNN)*

[2] Mạng nơ-ron tích chập (CNN) đóng vai trò then chốt trong việc giúp hệ thống "nhìn thấy" và xử lý hình ảnh. CNN thực hiện điều này bằng cách chia nhỏ hình ảnh thành các pixel – các điểm dữ liệu cơ bản của hình ảnh kỹ thuật số. Mỗi pixel được gắn nhãn hoặc gắn thẻ, giúp CNN hiểu được đặc điểm của hình ảnh.

CNN sau đó áp dụng một phép toán gọi là phép tích chập trên các nhãn pixel. Phép tích chập là một phép toán toán học kết hợp hai hàm để tạo ra một hàm thứ ba, qua đó mạng CNN có thể xác định các đặc trưng như cạnh, góc, màu sắc, và kết cấu của hình ảnh. CNN sử dụng những đặc trưng này để đưa ra dự đoán về nội dung của hình ảnh mà nó đang xử lý.

Quá trình này diễn ra qua nhiều lần lặp (iterations), trong đó CNN cải thiện dần dần độ chính xác của dự đoán. Mỗi lần lặp, hệ thống kiểm tra lại dự đoán của mình so với thực tế và tự điều chỉnh thông qua một cơ chế gọi là lan truyền ngược (backpropagation). Sau nhiều vòng lặp, mạng CNN đạt được độ chính xác cao trong việc nhận diện và phân loại hình ảnh.

CNN hoạt động giống như cách con người nhận thức hình ảnh ở xa. Đầu tiên, chúng ta sẽ nhận biết các đặc trưng cơ bản như đường nét và hình dạng đơn giản. Sau đó, khi quan sát kỹ hơn, chúng ta điền thêm thông tin chi tiết để hoàn chỉnh nhận thức. CNN cũng hoạt động theo cách này: từ các đặc điểm đơn giản ban đầu, mạng dần học được các đặc trưng phức tạp hơn thông qua nhiều lớp tích chập, từ đó nhận diện toàn bộ đối tượng trong hình ảnh.

* *CNN và RNN trong Thị giác Máy tính*

[2] CNN đặc biệt hiệu quả trong việc hiểu và xử lý hình ảnh tĩnh, như một bức ảnh hoặc khung hình đơn lẻ. Tuy nhiên, khi làm việc với video – nơi có sự liên quan giữa các khung hình liên tiếp – các mạng nơ-ron hồi quy (Recurrent Neural Networks - RNN) được sử dụng.

RNN được thiết kế để xử lý dữ liệu chuỗi, như video, bằng cách ghi nhớ mối quan hệ giữa các khung hình. Điều này giúp hệ thống không chỉ hiểu từng hình ảnh mà còn hiểu cách chúng liên kết với nhau theo thời gian. Ví dụ, trong một đoạn video quay cảnh giao thông, RNN có thể giúp hệ thống nhận biết không chỉ các phương tiện mà còn cả hướng di chuyển, tốc độ và mối quan hệ giữa chúng qua các khung hình liên tiếp.

* *Ưu điểm vượt trội của CNN và RNN*

[2] Nhờ khả năng xử lý hàng triệu dữ liệu một cách tự động, CNN và RNN cho phép hệ thống thị giác máy tính đạt hiệu quả vượt xa khả năng của con người. Một hệ thống được đào tạo tốt có thể phát hiện ra những lỗi hoặc đặc điểm nhỏ mà mắt người không thể nhận ra, đồng thời phân tích hàng nghìn hình ảnh hoặc video chỉ trong vài phút.

Sự kết hợp giữa deep learning, CNN, và RNN không chỉ giúp máy móc "nhìn thấy" mà còn có thể "hiểu" nội dung của dữ liệu trực quan một cách ngày càng giống con người.

* + 1. **Lịch sử phát triển của thị giác máy tính**

[3] Các nhà khoa học và kỹ sư đã cố gắng phát triển các phương pháp để máy móc có thể nhìn và hiểu dữ liệu hình ảnh trong khoảng 60 năm. Thí nghiệm bắt đầu vào năm 1959 khi các nhà sinh lý học thần kinh cho một con mèo xem một loạt hình ảnh, cố gắng liên hệ phản ứng trong não của nó. Họ phát hiện ra rằng nó phản ứng đầu tiên với các cạnh cứng hoặc đường thẳng và về mặt khoa học, điều này có nghĩa là quá trình xử lý hình ảnh bắt đầu bằng các hình dạng đơn giản như các cạnh thẳng.

Vào khoảng thời gian đó, công nghệ quét hình ảnh máy tính đầu tiên đã được phát triển, cho phép máy tính số hóa và thu thập hình ảnh. Một cột mốc khác đã đạt được vào năm 1963 khi máy tính có thể chuyển đổi hình ảnh hai chiều thành dạng ba chiều. Vào những năm 1960, AI nổi lên như một lĩnh vực nghiên cứu hàn lâm và nó cũng đánh dấu sự khởi đầu của nhiệm vụ AI nhằm giải quyết vấn đề thị giác của con người.

Năm 1974 chứng kiến ​​sự ra đời của công nghệ nhận dạng ký tự quang học (OCR), có thể nhận dạng văn bản được in bằng bất kỳ phông chữ hoặc kiểu chữ nào. Tương tự như vậy, nhận dạng ký tự thông minh (ICR) có thể giải mã văn bản viết tay bằng mạng nơ-ron. Kể từ đó, OCR và ICR đã tìm được đường vào xử lý tài liệu và hóa đơn, nhận dạng biển số xe, thanh toán di động, chuyển đổi máy và các ứng dụng phổ biến khác.

Năm 1982, nhà khoa học thần kinh David Marr đã xác định rằng thị giác hoạt động theo thứ bậc và giới thiệu các thuật toán để máy móc phát hiện các cạnh, góc, đường cong và các hình dạng cơ bản tương tự. Đồng thời, nhà khoa học máy tính Kunihiko Fukushima đã phát triển một mạng lưới các tế bào có thể nhận dạng các mẫu. Mạng lưới này, được gọi là Neocognitron, bao gồm các lớp tích chập trong mạng nơ-ron.

Đến năm 2000, trọng tâm nghiên cứu là nhận dạng đối tượng; và đến năm 2001, các ứng dụng nhận dạng khuôn mặt thời gian thực đầu tiên đã xuất hiện. Việc chuẩn hóa cách gắn thẻ và chú thích các tập dữ liệu trực quan đã xuất hiện trong những năm 2000. Năm 2010, tập dữ liệu ImageNet đã có sẵn. Nó chứa hàng triệu hình ảnh được gắn thẻ trên một nghìn lớp đối tượng và cung cấp nền tảng cho CNN và các mô hình học sâu được sử dụng ngày nay. Năm 2012, một nhóm từ Đại học Toronto đã đưa CNN vào cuộc thi nhận dạng hình ảnh. Mô hình có tên AlexNet đã giảm đáng kể tỷ lệ lỗi khi nhận dạng hình ảnh. Sau bước đột phá này, tỷ lệ lỗi đã giảm xuống chỉ còn vài phần trăm.

## **1.2. Tổng quan về nhận dạng đối tượng**

* + 1. **Nhận dạng đối tượng là gì?**

[4] Nhận dạng đối tượng (Object Recognition) là một lĩnh vực quan trọng trong thị giác máy tính, tập trung vào việc xác định, phân loại và xác định vị trí các đối tượng trong hình ảnh hoặc video. Công nghệ này đóng vai trò cốt lõi trong việc giúp máy móc "hiểu" và tương tác với thế giới thực thông qua dữ liệu trực quan, từ đó hỗ trợ nhiều ứng dụng thực tế trong đời sống và công nghiệp.

Nhờ sự phát triển nhanh chóng của trí tuệ nhân tạo và các mô hình học sâu (deep learning), khả năng nhận dạng đối tượng ngày càng chính xác và mạnh mẽ. Công nghệ này không chỉ cải thiện hiệu suất của các hệ thống tự động hóa mà còn mở ra những cơ hội lớn trong nhiều lĩnh vực như y tế, giao thông, công nghiệp và giải trí. Từ việc hỗ trợ chẩn đoán hình ảnh y khoa, phát hiện vật cản trong xe tự lái, đến nhận diện khuôn mặt trong các ứng dụng bảo mật, nhận dạng đối tượng đã trở thành một phần thiết yếu trong việc ứng dụng trí tuệ nhân tạo vào đời sống hiện đại.

* + 1. **Các phương pháp sử dụng để nhận dạng đối tượng**

[5] Nhận dạng đối tượng là một lĩnh vực quan trọng trong thị giác máy tính, với nhiều phương pháp được phát triển từ kỹ thuật truyền thống đến học sâu. Phương pháp truyền thống dựa trên đặc trưng thủ công như HOG, SIFT hoặc SURF, kết hợp với các thuật toán học máy như SVM hoặc KNN, phù hợp với dữ liệu nhỏ và các bài toán đơn giản. Tuy nhiên, học sâu đã mang lại bước đột phá nhờ khả năng tự động học đặc trưng từ dữ liệu. Các mạng nơ-ron tích chập (CNN) như AlexNet, VGG, ResNet và EfficientNet được sử dụng rộng rãi, trong khi các mô hình phát hiện đối tượng hiện đại như YOLO, SSD, và Faster R-CNN cho phép nhận dạng nhanh và chính xác.

Ngoài ra, học chuyển giao (Transfer Learning) giúp tận dụng các mô hình huấn luyện sẵn để áp dụng vào bài toán cụ thể, tiết kiệm thời gian và tài nguyên. Các phương pháp mới như DETR (Transformer-based) và GANs cũng đang dần thay đổi cách tiếp cận nhận dạng đối tượng nhờ khả năng tạo dữ liệu và chú ý vào chi tiết. Cuối cùng, việc kết hợp học sâu và học tăng cường đã mở ra tiềm năng mới cho các ứng dụng trong môi trường động. Việc lựa chọn phương pháp phụ thuộc vào yêu cầu bài toán, độ phức tạp của dữ liệu, và khả năng xử lý của hệ thống.

## **1.3. Tổng quan về học máy**

* + 1. **Học máy là gì?**

[6] Học máy (machine learning) là một lĩnh vực của trí tuệ nhân tạo liên quan đến việc nghiên cứu và xây dựng các kĩ thuật cho phép các hệ thống “học” tự động từ dữ liệu để giải quyết những vấn đề cụ thể. Một cách tổng quát, trong cuốn sách Machine Learning của tác giả Tom Mitchell xuất bản năm 1997, học máy được định nghĩa như sau: “A computer program is said to learn to perform a task T from experience E, if its performance at task T, as measured by a performance metric P, improves with experience E over time” (một chương trình máy tính được cho là học để thực hiện một nhiệm vụ T từ kinh nghiệm E, nếu hiệu suất thực hiện công việc T của nó được đo bởi chỉ số hiệu suất P và được cải thiện bởi kinh nghiệm E theo thời gian).

Như vậy, học máy nói đến một chương trình giúp cho máy tính có thể giải một bài toán cụ thể bằng những “kinh nghiệm” mà nó đã được “học” từ dữ liệu. Chương trình này có sự khác biệt lớn so với các chương trình lập trình truyền thống như. Sự khác biệt này có thể được mô tả trực quan như trong hình 1. Ở các chương trình lập trình truyền thống, con người sẽ phải viết ra các qui tắc, điều kiện để máy tính thực hiện nhiệm vụ và đưa ra câu trả lời. Ngược lại, đối với Học máy, nhiệm vụ của máy tính là phải dựa vào dữ liệu quan sát để tìm ra được các qui tắc này.

Tương tự như con người, máy tính cũng tự cải thiện khả năng giải quyết vấn đề thông qua việc học. Điểm khác biệt là ở chỗ, con người học được từ những quan sát thực tế, những điều mà con người được nhìn, được nghe, còn máy tính học từ dữ liệu.

* + 1. **Các phương pháp học máy**

[1] Học có giám sát (Supervised Learning) là phương pháp huấn luyện mô hình bằng tập dữ liệu đã được gán nhãn, trong đó đầu vào và đầu ra đều rõ ràng. Mục tiêu là xây dựng mối quan hệ giữa dữ liệu đầu vào và kết quả để dự đoán cho các trường hợp mới. Phương pháp này thường được áp dụng trong các bài toán phân loại và hồi quy với các thuật toán phổ biến như hồi quy tuyến tính, cây quyết định, và mạng nơ-ron nhân tạo.

Học không giám sát (Unsupervised Learning) không yêu cầu dữ liệu gán nhãn mà tập trung vào việc tìm ra cấu trúc hoặc mẫu ẩn trong dữ liệu. Nó thường được áp dụng trong các bài toán phân cụm và giảm chiều dữ liệu, sử dụng các thuật toán như K-Means Clustering hoặc phân tích thành phần chính (PCA).

Học bán giám sát (Semi-Supervised Learning) kết hợp dữ liệu có nhãn và không nhãn, phù hợp trong các trường hợp khó thu thập dữ liệu gán nhãn đầy đủ. Trong khi đó, học tăng cường (Reinforcement Learning) cho phép mô hình học cách đưa ra quyết định thông qua tương tác với môi trường, tối đa hóa phần thưởng thông qua các thuật toán như Q-Learning hay Deep Q-Networks.

Một phương pháp đặc biệt khác là học truyền tải (Transfer Learning), nơi mô hình tận dụng kiến thức từ các bài toán trước để áp dụng cho các bài toán mới, thường được sử dụng trong thị giác máy tính và xử lý ngôn ngữ tự nhiên. Bên cạnh đó, học sâu (Deep Learning), sử dụng các mạng nơ-ron sâu, đã trở thành một phương pháp mạnh mẽ cho các bài toán phức tạp như nhận diện hình ảnh, xử lý ngôn ngữ và phân tích dữ liệu lớn.

## **1.4. Ngôn ngữ, thư viện và công cụ sử dụng trong bài**

* + 1. **Ngôn ngữ Python:**



Python là một trong những ngôn ngữ phổ biến nhất trong lĩnh vực **xử lý ảnh và thị giác máy tính**, nhờ vào sự hỗ trợ mạnh mẽ từ các thư viện chuyên dụng và cộng đồng phát triển rộng lớn. Các công cụ như OpenCV, PIL (Pillow), Tesseract OCR, và các framework học sâu như TensorFlow, PyTorch hay Keras đã giúp Python trở thành lựa chọn hàng đầu cho các ứng dụng nhận diện và xử lý ảnh.

* + 1. **Các thư viện sử dụng**

1. **Thư viện OpenCV**



OpenCV, viết tắt của Thư viện Thị giác Máy tính Mở, là một thư viện phần mềm dành cho máy học và thị giác máy tính. Ban đầu được phát triển bởi Intel, hiện tại nó do cộng đồng của OpenCV Foundation duy trì. OpenCV là một thư viện mã nguồn mở lớn phục vụ cho ứng dụng thị giác máy tính, học máy và xử lý hình ảnh. Hiện nay, thư viện này đóng vai trò quan trọng trong các hệ thống thời gian thực, có ý nghĩa tích cực trong nhiều ứng dụng hiện đại. Với OpenCV, người dùng có thể thực hiện việc phân tích hình ảnh và video để phát hiện các đối tượng, khuôn mặt hoặc chữ viết tay.

Khi kết hợp với nhiều thư viện khác như NumPy, Python có khả năng xử lý cấu trúc mảng của OpenCV để thực hiện các phép phân tích. Để nhận diện một mẫu hình ảnh và các đặc điểm khác của nó, ta sử dụng không gian vectơ và thực hiện các phép toán phù hợp.

OpenCV phiên bản đầu tiên là 1.0 và được phát hành dưới giấy phép BSD, do đó có thể sử dụng miễn phí cho cả mục đích học thuật và thương mại. Thư viện này hỗ trợ nhiều ngôn ngữ lập trình như C++, C, Python và Java, cùng với việc hoạt động trên các hệ điều hành như Windows, Linux, Mac OS, iOS và Android. Mục tiêu chính khi thiết kế OpenCV là để phục vụ cho các ứng dụng thời gian thực, với mã nguồn được tối ưu hóa cho khả năng xử lý đa lõi.

**Một số ứng dụng phổ biến của OpenCV bao gồm:**

• Nhận diện khuôn mặt

• Giám sát tự động và kiểm tra

• Đếm số lượng người (như lưu lượng đi bộ trong trung tâm mua sắm)

• Đếm xe trên đường và theo dõi tốc độ

• Tác phẩm nghệ thuật tương tác

• Phát hiện bất thường trong quy trình sản xuất (sản phẩm lỗi)

• Ghép hình ảnh từ chế độ xem phố

• Tìm kiếm và truy xuất video/hình ảnh

• Điều khiển và dẫn đường cho ô tô và robot tự lái

• Nhận diện đối tượng

• Phân tích hình ảnh y tế

• Tạo cấu trúc 3D từ chuyển động trong phim

• Nhận diện quảng cáo trên truyền hình

**Chức năng nổi bật của OpenCV bao gồm:**

• Nhập/xuất hình ảnh/video, xử lý và hiển thị

• Phát hiện đối tượng và đặc trưng

• Thị giác máy tính đơn kính hoặc stereo dựa trên hình học

• Học máy và phân cụm,

1. **Tesseract OCR**

là một thư viện mã nguồn mở mạnh mẽ, được phát triển bởi Google, dành cho việc nhận diện văn bản từ hình ảnh. Đây là công cụ phổ biến trong các ứng dụng xử lý ảnh, đặc biệt hữu ích trong các bài toán trích xuất thông tin từ tài liệu hoặc thẻ nhận diện.

* **Ưu điểm của Tesseract OCR:**

**Hỗ trợ nhiều ngôn ngữ**: Tesseract có khả năng nhận diện hơn 100 ngôn ngữ, bao gồm cả tiếng Việt.

**Mở rộng dễ dàng**: Người dùng có thể huấn luyện thêm mô hình nhận diện ký tự cho các ngôn ngữ hoặc định dạng không phổ biến.

**Tích hợp tốt với Python**: Với thư viện pytesseract, Tesseract OCR dễ dàng được sử dụng trong các ứng dụng Python.

* **Quy trình nhận diện văn bản bằng Tesseract OCR:**

**Nhập ảnh**: Đọc ảnh từ các định dạng phổ biến như PNG, JPEG hoặc TIFF.

**Tiền xử lý ảnh**: Ảnh được chuyển đổi sang dạng đen trắng (grayscale), làm mịn, hoặc tăng độ tương phản để tối ưu cho nhận diện.

**Nhận diện ký tự**: Tesseract trích xuất văn bản từ hình ảnh đầu vào và trả về kết quả dưới dạng chuỗi ký tự.

**Xử lý văn bản**: Chuỗi văn bản trích xuất có thể được xử lý thêm để chuẩn hóa hoặc phân loại.

* **Các bước cải thiện độ chính xác của Tesseract OCR:**

**Tiền xử lý ảnh chất lượng cao**: Sử dụng các kỹ thuật lọc nhiễu và phát hiện biên để làm nổi bật văn bản.

**Cấu hình tùy chỉnh**: Tesseract cho phép sử dụng các tùy chọn nhận diện cụ thể, như chế độ dò tìm vùng văn bản hoặc loại bỏ các phần không liên quan.

**Tối ưu hóa ngôn ngữ**: Đối với văn bản tiếng Việt, sử dụng tập dữ liệu ngôn ngữ huấn luyện vie.traineddata để nâng cao độ chính xác.

* **Ứng dụng của Tesseract OCR trong bài toán nhận diện thông tin từ thẻ sinh viên:**

**Phân đoạn thẻ sinh viên**: Xác định các vùng chứa họ tên, mã số sinh viên (MSSV), và các trường thông tin khác.

**Nhận diện chữ viết**: Tesseract OCR giúp trích xuất chính xác các ký tự và số từ vùng thông tin trên thẻ.

**Xử lý thông tin trích xuất**: Kết quả văn bản được chuẩn hóa để lưu trữ vào cơ sở dữ liệu hoặc tích hợp vào hệ thống quản lý.

* **Hạn chế:**

Độ chính xác của Tesseract phụ thuộc vào chất lượng ảnh: Ảnh mờ, nhiều nhiễu hoặc nền không đồng đều sẽ làm giảm hiệu quả.

Bố cục văn bản phức tạp: Các ảnh có bảng biểu hoặc cột dọc yêu cầu thêm các bước xử lý đặc thù để trích xuất chính xác.

Tesseract OCR là một giải pháp hữu ích và tiết kiệm chi phí trong việc tự động hóa trích xuất thông tin từ ảnh, đặc biệt phù hợp cho các bài toán xử lý văn bản trong thực tế.

# **CHƯƠNG 2: PHÁT BIỂU VÀ PHÂN TÍCH BÀI TOÁN**

## **2.1. Phát biểu bài toán**

### **2.1.1. Đặt vấn đề**

Trong các tổ chức giáo dục hiện nay, việc quản lý thông tin sinh viên là một trong những nhiệm vụ quan trọng và cần thiết. Tuy nhiên, quy trình nhập liệu thông tin sinh viên từ thẻ vào hệ thống thường được thực hiện thủ công, điều này dẫn đến nhiều vấn đề như mất thời gian, tốn kém nhân lực và dễ xảy ra sai sót trong quá trình nhập liệu. Những lỗi như nhập sai thông tin cá nhân, thiếu dữ liệu hoặc trùng lặp thông tin không chỉ ảnh hưởng đến chất lượng quản lý mà còn gây ra sự bất tiện cho sinh viên và cán bộ quản lý.

Thẻ sinh viên là một loại giấy tờ phổ biến trong các tổ chức giáo dục, chứa đựng các thông tin cơ bản của sinh viên như họ tên, ngày sinh, mã số sinh viên, khóa học và thời hạn sử dụng thẻ. Những thông tin này đóng vai trò quan trọng trong việc xác nhận danh tính, quản lý học vụ và cung cấp các dịch vụ liên quan đến sinh viên. Với sự phát triển của công nghệ, đặc biệt là trí tuệ nhân tạo và xử lý hình ảnh, bài toán tự động hóa quy trình trích xuất thông tin từ thẻ sinh viên thông qua hình ảnh chụp trở thành một giải pháp tiềm năng.

Mục tiêu chính của bài toán là phát triển một hệ thống thông minh, có khả năng nhận diện thông tin từ hình ảnh thẻ sinh viên được chụp từ camera. Hệ thống sẽ sử dụng các kỹ thuật xử lý ảnh và nhận diện ký tự quang học (OCR) để tự động trích xuất thông tin như họ tên, ngày sinh, mã số sinh viên, khóa học và giá trị thẻ. Sau khi trích xuất, thông tin sẽ được hiển thị trực quan trên giao diện người dùng và lưu trữ vào cơ sở dữ liệu để phục vụ cho việc quản lý.

Giải pháp này không chỉ cải thiện đáng kể tốc độ và độ chính xác trong việc nhập liệu thông tin mà còn giúp giảm tải công việc thủ công cho cán bộ quản lý. Ngoài ra, việc lưu trữ thông tin một cách có hệ thống cũng tạo điều kiện thuận lợi cho việc truy xuất và cập nhật dữ liệu khi cần thiết. Việc áp dụng công nghệ tự động hóa như vậy không chỉ nâng cao hiệu quả quản lý mà còn khẳng định sự chuyên nghiệp và hiện đại của các tổ chức giáo dục.

Câu hỏi đặt ra là: Làm thế nào để thiết kế và triển khai một hệ thống có thể xử lý hiệu quả bài toán này trong điều kiện thực tế, với những thách thức như chất lượng hình ảnh đầu vào, định dạng khác nhau của thẻ sinh viên và yêu cầu bảo mật dữ liệu? Đây chính là động lực để nghiên cứu và phát triển hệ thống tự động trích xuất thông tin từ thẻ sinh viên thông qua hình ảnh.

### **2.1.2. Mô tả bài toán**

Bài toán đặt ra là xây dựng một hệ thống thông minh để tự động trích xuất thông tin từ hình ảnh thẻ sinh viên. Hệ thống này nhằm giải quyết những bất cập trong quy trình nhập liệu thủ công và tăng hiệu quả quản lý thông tin sinh viên tại các tổ chức giáo dục.

Cụ thể, bài toán bao gồm các yêu cầu chính như sau:

1. Đầu vào:
   * Hình ảnh thẻ sinh viên được chụp từ camera hoặc các thiết bị tương tự.
   * Hình ảnh đầu vào có thể có các vấn đề như: nghiêng, mờ, nhiễu, hoặc ánh sáng không đồng đều.
2. Xử lý hình ảnh:
   * Thực hiện tiền xử lý hình ảnh để cải thiện chất lượng, bao gồm:
     + Loại bỏ nhiễu.
     + Chỉnh góc độ nếu hình ảnh bị nghiêng.
     + Tăng độ tương phản để làm rõ thông tin trên thẻ.
   * Phát hiện và định vị các vùng chứa thông tin trên thẻ (ví dụ: họ tên, ngày sinh, mã sinh viên, khóa học, giá trị thẻ).
3. Nhận diện và trích xuất thông tin:
   * Áp dụng thuật toán nhận diện ký tự quang học (OCR) để trích xuất các thông tin từ vùng được định vị.
   * Các thông tin cần trích xuất bao gồm:
     + Họ tên: Dữ liệu dạng chuỗi văn bản.
     + Ngày sinh: Dữ liệu dạng ngày tháng.
     + Mã sinh viên: Dữ liệu dạng chuỗi hoặc số.
     + Khóa học: Thông tin về khóa học hoặc chuyên ngành.
     + Giá trị thẻ: Thời hạn sử dụng thẻ.
4. Xử lý và hiển thị kết quả:
   * Kết quả sau khi trích xuất sẽ được:
     + Hiển thị trên giao diện để người dùng kiểm tra và xác nhận.
     + Tự động lưu trữ vào cơ sở dữ liệu để phục vụ quản lý.
5. Yêu cầu hệ thống:
   * Đảm bảo độ chính xác cao trong việc trích xuất thông tin, đặc biệt là đối với các thông tin quan trọng như mã số sinh viên.
   * Xử lý nhanh chóng để đáp ứng nhu cầu thực tế trong các trường hợp nhập liệu khối lượng lớn.
   * Tương thích với nhiều định dạng thẻ sinh viên khác nhau.
6. Bảo mật và lưu trữ:
   * Đảm bảo an toàn dữ liệu cá nhân của sinh viên trong quá trình xử lý và lưu trữ.
   * Hệ thống cần mã hóa dữ liệu và tuân thủ các quy định về bảo mật thông tin.

Kỳ vọng: Hệ thống này sẽ giúp tự động hóa quy trình nhập liệu, giảm thiểu thời gian và công sức của nhân viên quản lý, đồng thời cải thiện độ chính xác và hiệu quả trong việc lưu trữ thông tin sinh viên. Hơn nữa, giải pháp sẽ tạo tiền đề cho việc áp dụng công nghệ hiện đại vào các quy trình quản lý giáo dục khác.

## **2.2. Phân tích hệ thống**

### **2.2.1. Các kĩ thuật sử dụng**

**1. Xử lý ảnh với OpenCV**

**a. Chụp ảnh từ camera**

* **Nguyên lý:** OpenCV sử dụng đối tượng cv2.VideoCapture() để kết nối với thiết bị camera của hệ thống. Camera cung cấp các khung hình video dưới dạng ma trận ảnh.
* **Diễn giải:**

cap = cv2.VideoCapture(0)

* + 0: Chỉ định camera đầu tiên được kết nối (mặc định là webcam tích hợp).
  + cap.read() trả về hai giá trị:
    - **ret**: Trả về True nếu camera hoạt động.
    - **frame**: Mảng NumPy chứa dữ liệu ảnh của khung hình.

**b. Chuyển đổi không gian màu**

* **Nguyên lý:** Ảnh từ camera thường ở không gian màu BGR (Blue, Green, Red). Chuyển đổi sang ảnh xám (grayscale) giúp giảm kích thước dữ liệu và tập trung vào cường độ sáng.
* **Diễn giải:**

gray\_frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

* + **cv2.COLOR\_BGR2GRAY**: Ánh xạ từng pixel từ không gian màu BGR sang cường độ ánh sáng đơn sắc.
  + **Lý do sử dụng:** Ảnh xám dễ xử lý và tăng tốc độ phân tích (như phân ngưỡng, phát hiện cạnh).

**c. Phân ngưỡng (Thresholding)**

* **Nguyên lý:** Phân ngưỡng là kỹ thuật biến ảnh xám thành ảnh nhị phân (chỉ có hai mức màu: đen và trắng). Pixel nào có giá trị cường độ lớn hơn ngưỡng sẽ trở thành **trắng**, và ngược lại là **đen**.
* **Diễn giải:**

ret, thresh = cv2.threshold(cutTEN, 100, 255, cv2.THRESH\_BINARY\_INV)

* + cutTEN: Ảnh đầu vào.
  + 100: Ngưỡng cường độ ánh sáng.
  + 255: Giá trị pixel sau phân ngưỡng (trắng).
  + **cv2.THRESH\_BINARY\_INV**: Đảo ngược màu (nền trắng, ký tự đen).
  + **Lý do sử dụng:** Tăng độ tương phản giữa nền và ký tự, giúp nhận dạng văn bản chính xác hơn.

**d. Cắt vùng quan tâm (ROI)**

* **Nguyên lý:** Ảnh là một mảng hai chiều, nên có thể dùng chỉ số hàng (y) và cột (x) để trích xuất một phần ảnh.
* **Diễn giải:**

TEN = frame[215:240, 305:480]

* + **[215:240, 305:480]**:
    - y=215 đến y=240: Giới hạn hàng.
    - x=305 đến x=480: Giới hạn cột.
  + **Lý do sử dụng:** Tập trung vào phần ảnh chứa thông tin cần trích xuất (tên, ngày sinh...).

**2. Nhận dạng ký tự quang học (OCR) với Tesseract**

**a. Tiền xử lý ảnh đầu vào**

* **Nguyên lý:** Tesseract nhận một ảnh đầu vào và trước tiên sẽ thực hiện một số thao tác tiền xử lý để chuẩn bị ảnh cho việc nhận dạng ký tự. Các bước bao gồm:
  + Chuyển ảnh thành grayscale (nếu chưa có).
  + Phân ngưỡng ảnh để tạo ra ảnh nhị phân (ảnh chỉ có hai màu đen và trắng).
  + Loại bỏ nhiễu và làm sạch nền.
* **Ví dụ:** Một ảnh nhị phân có nền trắng và ký tự đen giúp Tesseract dễ dàng phân biệt giữa các vùng chứa ký tự và vùng nền.

**b. Xác định vùng chứa văn bản**

* **Nguyên lý:** Tesseract chia ảnh thành các khối (blocks), sau đó phân đoạn thành:
  + **Dòng văn bản:** Tìm các hàng ngang chứa văn bản.
  + **Từ:** Tách các từ trong dòng văn bản.
  + **Ký tự:** Nhận dạng từng ký tự riêng lẻ.
* **Kỹ thuật sử dụng:**
  + **Connected Component Analysis (CCA):** Tìm kiếm các vùng pixel kết nối liền nhau để phát hiện khối văn bản.
  + **Hough Transform hoặc Projection Profiling:** Phân tích dựa trên mật độ pixel để xác định dòng và ký tự.

**c. Trích xuất đặc trưng ký tự**

* **Nguyên lý:** Tesseract phân tích từng ký tự và trích xuất các đặc trưng (features) để so sánh với cơ sở dữ liệu ký tự đã được huấn luyện.
* **Các đặc trưng trích xuất:**
  + Hình dạng (contour): Đường viền của ký tự.
  + Vùng tối (black pixel regions): Phân bố pixel đen bên trong ký tự.
  + Gốc và đỉnh (strokes): Các đường nét chính của ký tự.
* **Ví dụ:**
  + Chữ "A" có cấu trúc là một đỉnh nhọn và hai nét xiên song song.
  + Chữ "O" là một hình tròn khép kín.

**d. Nhận dạng ký tự bằng so khớp mẫu (Pattern Matching)**

* **Nguyên lý:** Tesseract so sánh đặc trưng của ký tự đã trích xuất với cơ sở dữ liệu ký tự trong mô hình đã huấn luyện.
* **Phương pháp:**
  + **Template Matching:** So sánh trực tiếp với các mẫu ký tự (đã được huấn luyện).
  + **Machine Learning (LSTM):** Tesseract (phiên bản từ 4.0 trở lên) sử dụng mạng nơ-ron hồi tiếp (LSTM - Long Short-Term Memory) để nhận dạng ký tự.
    - LSTM học cách phân biệt ký tự dựa trên ngữ cảnh và hình dạng.

**e. Hậu xử lý văn bản**

* **Nguyên lý:** Sau khi nhận dạng từng ký tự, Tesseract thực hiện các bước để cải thiện độ chính xác của kết quả:
  + **Kiểm tra từ điển:** Đối chiếu các từ nhận dạng với một từ điển ngôn ngữ (nếu có). Ví dụ: Khi sử dụng tiếng Việt, nó sẽ ưu tiên các từ có trong từ điển tiếng Việt.
  + **Chỉnh sửa lỗi:** Sửa lỗi chính tả hoặc gợi ý các từ thay thế dựa trên ngữ cảnh.

**f. Trả về văn bản đầu ra**

* **Nguyên lý:** Toàn bộ văn bản được kết hợp từ các ký tự hoặc từ đã nhận dạng. Sau đó, Tesseract trả về kết quả dưới dạng chuỗi văn bản (string).

### **2.2.2. Mô hình tổng quan hệ thống**

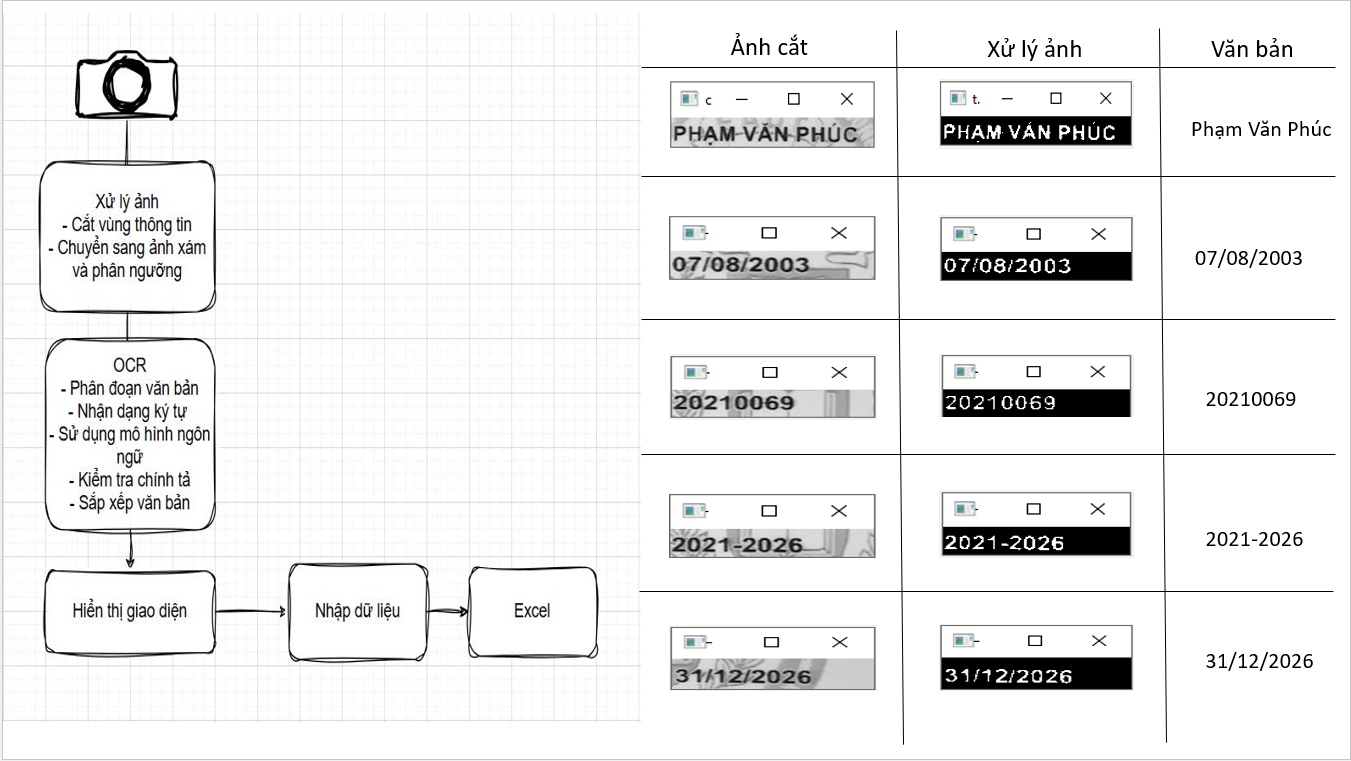
**1. Khởi tạo**

* **Input**:   
  Hình ảnh thẻ sinh viên được chụp từ camera. Ảnh được đưa vào hệ thống để bắt đầu quá trình xử lý.
* **Chức năng**:
  + Hệ thống tiếp nhận hình ảnh từ camera và hiển thị trên giao diện người dùng để người dùng có thể quan sát.
  + Hệ thống chuẩn bị các công cụ cần thiết để xử lý ảnh, như Tesseract OCR cho nhận dạng văn bản.

**2. Xử lý ảnh (Image Processing)**

* **Bước 1: Tiền xử lý ảnh (Pre-processing)**
  1. **Chuyển đổi ảnh sang ảnh xám (Grayscale)**:  
     Ảnh màu sẽ được chuyển thành ảnh xám để đơn giản hóa việc nhận dạng ký tự.
  2. **Phân ngưỡng (Thresholding)**:  
     Phân ngưỡng để chuyển ảnh xám thành ảnh nhị phân (đen trắng), giúp tăng cường độ tương phản giữa văn bản và nền.
  3. **Cắt vùng chứa thông tin**:  
     Hệ thống cắt các vùng chứa thông tin quan trọng trên thẻ sinh viên (Họ tên, Ngày sinh, Mã sinh viên, Khóa học, Giá trị thẻ).
  4. **Lọc nhiễu (Noise Removal)**:  
     Loại bỏ các điểm nhiễu và các phần không liên quan như nền ảnh để cải thiện độ chính xác nhận dạng.
* **Bước 2: Nhận dạng văn bản (Text Recognition using OCR)**
  1. **Phân đoạn văn bản (Segmentation)**:  
     Hệ thống chia ảnh thành các phần như dòng, từ, ký tự để dễ dàng nhận diện.
  2. **Nhận dạng ký tự (Character Recognition)**:  
     Tesseract OCR sẽ nhận dạng các ký tự trong ảnh và chuyển chúng thành văn bản.
  3. **Sử dụng mô hình ngôn ngữ**:  
     Mô hình ngữ cảnh sẽ được sử dụng để cải thiện độ chính xác nhận dạng các từ và sửa lỗi nhận dạng khi cần thiết.
* **Bước 3: Hậu xử lý (Post-processing)**
  1. **Kiểm tra chính tả (Spell-check)**:  
     Hệ thống kiểm tra chính tả của văn bản nhận dạng và sửa các lỗi chính tả nếu có.
  2. **Sắp xếp và định dạng văn bản**:  
     Các thông tin nhận dạng được sắp xếp theo thứ tự và cấu trúc chuẩn như Họ tên, Ngày sinh, Mã sinh viên, Khóa học.

**3. Kết quả (Output)**

* **Kết quả văn bản nhận dạng**:  
  Sau khi hoàn thành quá trình xử lý và nhận dạng văn bản, hệ thống sẽ xuất ra một chuỗi văn bản chứa các thông tin đã được trích xuất từ thẻ sinh viên, bao gồm:
  + **Họ tên**
  + **Ngày sinh**
  + **Mã sinh viên**
  + **Khóa học**
  + **Giá trị thẻ**
* **Lưu trữ kết quả**:
  + Khi người dùng nhấn nút "Nhập dữ liệu", hệ thống sẽ lưu thông tin vào file Excel để quản lý sau này.

Hình 1: Mô hình hoạt động của hệ thống

# **CHƯƠNG 3: KẾT QUẢ THỰC NGHIỆM**

## **3.1. Dữ liệu**

* Dữ liệu đầu vào sẽ được đọc từ camera theo thời gian thực, sau khi thẻ sinh viên được đưa vào trong khay và camera hiển thị ngay lên trên giao diện, sau đó hệ thống sẽ cắt dữ liệu theo từng khung cô định dữ liệu được đặt sẵn từ đầu.
* Sau khi cắt được thông tin hệ thống sẽ sử lý ảnh trước khi đọc dữ liệu trên ảnh bằng cách sử dụng Opencv để chuyển sang ảnh xám, phân ngưỡng để nổi bật kí tự văn bản.
* Ảnh khi đã được xử lý xong thì sẽ được Tesseract nhận diện và chuyển sang dạng văn bản để hiển thị lên màn hình.

## **3.2. Kết quả thực nghiệm**

### **3.2.1. Thông tin thẻ sinh viên của “ Phạm Văn Phúc “**



Hình 2: Thông tin thẻ sinh viên của “ Phạm Văn Phúc “

**3.2.2. Thông tin thẻ sinh viên của “ Nguyễn Đức Minh Hiếu”**

****

Hình 3.2.2: Thông tin thẻ sinh viên của “ Nguyễn Đức Minh Hiếu”

**3.3.3. Thông tin thẻ sinh viên của “ Hoàng Thái Gia ”**

Hình 3: Thông tin thẻ sinh viên của “ Hoàng Thái Gia ”

**KẾT LUẬN**

Trong khuôn khổ bài tập lớn cuối kỳ, chúng em đã hoàn thành dự án "Trích xuất thông tin từ thẻ sinh viên" bằng cách sử dụng các công nghệ hiện đại như OpenCV, Tesseract OCR, và Python. Dự án này không chỉ giúp chúng em nắm vững các kiến thức về xử lý ảnh, thị giác máy tính, mà còn rèn luyện kỹ năng lập trình thực tiễn trong việc phát triển ứng dụng phần mềm.

**Các phần đã làm được:**

* Xây dựng giao diện người dùng với Tkinter để hiển thị thông tin và tương tác với người dùng.
* Sử dụng OpenCV để lấy ảnh từ camera và xử lý ảnh theo các bước cần thiết.
* Áp dụng Tesseract OCR để nhận diện và trích xuất các thông tin từ ảnh thẻ sinh viên (như họ tên, ngày sinh, mã sinh viên, khóa học, và giá trị thẻ).
* Lưu trữ và quản lý dữ liệu sinh viên vào file Excel bằng OpenPyXL, giúp hệ thống có thể tra cứu và lưu trữ thông tin một cách dễ dàng.
* Giao diện thân thiện và dễ sử dụng, hỗ trợ việc kiểm tra và nhập liệu nhanh chóng cho người dùng.

**Các phần chưa làm được:**

* Cải thiện độ chính xác của OCR: Mặc dù Tesseract OCR có thể trích xuất thông tin từ thẻ sinh viên, nhưng độ chính xác trong một số trường hợp vẫn chưa cao, đặc biệt khi thông tin trên thẻ bị mờ hoặc có chất lượng ảnh không tốt.
* Tích hợp nhận diện khuôn mặt: Việc tích hợp nhận diện khuôn mặt vào hệ thống để xác thực danh tính của sinh viên vẫn chưa được thực hiện do thiếu thời gian và tài nguyên.
* Tự động phân loại thông tin: Mặc dù hệ thống có thể trích xuất thông tin từ các vùng xác định trước trên thẻ, nhưng việc tự động phân loại thông tin từ các thẻ sinh viên có chất lượng ảnh khác nhau vẫn chưa được tối ưu.

**Hướng phát triển:**

* Cải thiện và tối ưu hóa OCR: Tăng cường độ chính xác của Tesseract OCR thông qua việc điều chỉnh các tham số, kết hợp với các thuật toán tiền xử lý ảnh nâng cao như làm sắc nét ảnh, loại bỏ nhiễu, và cải thiện phân ngưỡng.
* Tích hợp nhận diện khuôn mặt: Phát triển tính năng nhận diện khuôn mặt của sinh viên để tăng tính bảo mật và xác thực thông tin chính xác hơn.
* Hỗ trợ nhiều loại thẻ sinh viên: Phát triển hệ thống có khả năng nhận diện thông tin từ các loại thẻ sinh viên khác nhau, bao gồm các thẻ có định dạng khác nhau và thông tin được trình bày theo cách khác nhau.
* Tự động phân loại và kiểm tra thông tin: Sử dụng học máy để phát triển tính năng tự động phân loại và kiểm tra thông tin nhận dạng, từ đó nâng cao tính chính xác và hiệu quả của hệ thống.

Kết quả đạt được từ dự án này không chỉ đáp ứng được yêu cầu bài tập mà còn tạo ra một sản phẩm có thể ứng dụng trong thực tế, giúp tự động hóa quá trình trích xuất thông tin từ thẻ sinh viên, giảm thiểu sai sót do con người, đồng thời tiết kiệm thời gian và công sức trong việc quản lý dữ liệu sinh viên.

Dự án này mở ra nhiều cơ hội cho chúng em trong việc phát triển thêm các tính năng mới như nhận dạng chữ viết tay, tăng cường độ chính xác của hệ thống OCR, và tích hợp thêm các công nghệ khác như nhận diện khuôn mặt để phục vụ cho các ứng dụng trong lĩnh vực quản lý thông tin.

Chúng em xin cảm ơn thầy cô đã tạo cơ hội để thực hiện bài tập này và sẽ tiếp tục nghiên cứu, cải tiến hệ thống để đáp ứng tốt hơn các yêu cầu thực tế trong tương lai.

**TÀI LIỆU THAM KHẢO**

**[1]** Trí tuệ nhân tạo và học sâu trong thị giác máy tính, Ian Goodfellow, Yoshua Bengio, và Aaron Courville - "Deep Learning" (2016)

**[2]** Backpropagation theory, Rumelhart, Hinton, and Williams (1986)

**[3]** The History of Computer Vision, Tsai and Lee (2008)

**[4]** Object Detection & Recognition: Theories and Implementations, Sinno Jialin Pan (2015)

**[5]** Deep Learning for Computer Vision, Adrian Rosebrock (2020)

**[6]** Machine Learning, Tom M. Mitchell (1997)