BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ ĐÔNG Á

BÀI TẬP LỚN

HỌC PHẦN: XỬ LÝ ẢNH VÀ THỊ GIÁC MÁY TÍNH

ĐỀ 06: XÂY DỰNG HỆ THỐNG NHẬN DIỆN CHỮ SỐ VIẾT TAY

Sinh viên thực hiện	Khóa	Lớp
Nguyễn Văn Huy	K12	DCCNTT12.10.9

Bắc Ninh, năm 2024

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ ĐÔNG Á

BÀI TẬP LỚN

HỌC PHẦN: XỬ LÝ ẢNH VÀ THỊ GIÁC MÁY TÍNH

Nhóm: 09

ĐỀ 06: XÂY DỰNG HỆ THỐNG NHẬN DIỆN CHỮ SỐ VIẾT TAY

STT	Sinh viên thực hiện	Mã sinh viên	Điểm bằng số	Điểm bằng chữ
1	Nguyễn Văn Huy	**20212454**		

CÁN BỘ CHẤM 1
(Ký và ghi rõ họ tên)

CÁN BỘ CHẤM 2
(Ký và ghi rõ họ tên)

MỤC LỤC

DANH MỤC ẢNH

LỜI MỞ ĐẦU

Trong thời đại số hóa và tự động hóa, việc áp dụng các công nghệ tiên tiến như trí tuệ nhân tạo (AI) và học máy (Machine Learning) đang thay đổi cách thức hoạt động của nhiều lĩnh vực trên toàn thế giới. Những tiến bộ trong công nghệ không chỉ tạo ra những phương pháp mới để xử lý dữ liệu mà còn giúp giải quyết các vấn đề phức tạp một cách hiệu quả hơn. Trong số những ứng dụng của AI và học máy, khả năng nhận dạng chữ viết tay, đặc biệt là các chữ số, đã trở thành một công cụ hữu ích và phổ biến, từ việc hỗ trợ xử lý tài liệu giấy, bài thi trắc nghiệm đến việc phân loại thông tin trong các ứng dụng di động và hệ thống quản lý văn bản.

Hệ thống nhận dạng chữ viết tay không chỉ đơn thuần là một công nghệ hữu ích mà còn đóng vai trò quan trọng trong việc cải thiện năng suất lao động. Với khả năng tự động nhận diện và xử lý dữ liệu, các hệ thống này có thể thay thế các phương pháp nhập liệu thủ công, từ đó giảm thiểu sai sót và công sức của con người. Điều này đặc biệt quan trọng trong các ngành như tài chính, giáo dục và hành chính công, nơi khối lượng tài liệu và dữ liệu cần xử lý là rất lớn.

Để hiện thực hóa một hệ thống như vậy, nhóm chúng em sẽ tập trung vào việc ứng dụng TensorFlow, một thư viện mã nguồn mở mạnh mẽ và phổ biến trong việc xây dựng các ứng dụng học sâu (Deep Learning). Hệ thống mà chúng em phát triển sẽ không chỉ giúp nhận dạng chính xác các chữ số viết tay mà còn minh chứng cho tiềm năng của công nghệ AI trong việc giải quyết những vấn đề thực tế và phức tạp.

Em sẽ tiến hành nghiên cứu và phát triển hệ thống này theo một quy trình gồm nhiều bước, bắt đầu từ việc thu thập dữ liệu, tiền xử lý dữ liệu, thiết kế mô hình học sâu, huấn luyện mô hình, đến giai đoạn đánh giá và tối ưu hóa hiệu năng. Trong quá trình này, em sẽ không chỉ mô tả các phương pháp và kỹ thuật được sử dụng mà còn phân tích những khó khăn gặp phải, đồng thời đề xuất các giải pháp để khắc phục.

Những kết quả đạt được sẽ không chỉ giúp cải thiện độ chính xác trong việc nhận dạng chữ số viết tay mà còn mở ra những tiềm năng mới cho việc áp dụng công nghệ học máy vào các bài toán tương tự. Việc xây dựng một hệ thống nhận dạng hiệu quả có thể tạo tiền đề cho các nghiên cứu và ứng dụng mở rộng hơn trong tương lai.

Chương 1: Cơ Sở lý thuyết

1.1. Giới thiệu về Python

1.1.1. Định nghĩa về Python

Python là một ngôn ngữ lập trình được sử dụng rộng rãi trong các ứng dụng web, phát triển phần mềm, khoa học dữ liệu và máy học (ML). Các nhà phát triển sử dụng Python vì nó hiệu quả, dễ học và có thể chạy trên nhiều nền tảng khác nhau.

Python là một ngôn ngữ lập trình thông dịch, dễ đọc và dễ hiểu. Nền tảng nổi tiếng với cú pháp đơn giản và được sử dụng rộng rãi trong nhiều lĩnh vực khác nhau. Điển hình như phát triển web, phân tích dữ liệu, trí tuệ nhân tạo và nhiều ứng dụng khác.

Python có cú pháp linh hoạt và cấu trúc dữ liệu mạnh mẽ, công nghệ được hỗ trợ bởi một cộng đồng lớn. Điều này đã mang đến các thư viện và framework phong phú mà người dùng có thể sử dụng để xây dựng các ứng dụng phức tạp. Python cũng là một trong những ngôn ngữ phổ biến cho người mới học lập trình nhờ vào tính linh hoạt của nó.

1.1.2. Lợi ích của python

Cú pháp đơn giản và dễ đọc, phù hợp cho người mới học lập trình và cũng dễ dàng cho những người có kinh nghiệm.

Python được sử dụng trong nhiều lĩnh vực như phát triển web, phân tích dữ liệu, trí tuệ nhân tạo, khoa học dữ liệu và nhiều ứng dụng khác. Điều này làm cho Python trở thành một ngôn ngữ lập trình linh hoạt và tiện lợi.

Cộng đồng Python khá lớn và luôn hoạt động tích cực nhằm cung cấp nhiều thư viện, framework hữu ích. Người dùng sẽ được nhận sự hỗ trợ từ cộng đồng thông qua tài liệu, diễn đàn và các nguồn thông tin trực tuyến.

Python cung cấp nhiều thư viện mạnh mẽ cho phân tích dữ liệu như Pandas, NumPy, và Matplotlib. Bộ ngôn ngữ giúp người dùng xử lý và thể hiện dữ liệu một cách hiệu quả.

Python có thể chạy trên nhiều hệ điều hành khác nhau và trở thành một trong những ngôn ngữ lập trình chéo phổ biến.

Ngôn ngữ Python là mã nguồn mở và miễn phí, cho phép người dùng tự do sử dụng, phân phối và thay đổi phiên bản của nó.

1.1.3. Phương thức hoạt động chính của Python

Phát triển trang web trên máy chủ:

Python có nhiều framework phát triển web phổ biến, ví dụ như Django, Flask, Pyramid và Fast API. Đây là các công cụ tối ưu giúp lập trình viên phát triển ứng dụng web từ phía máy chủ một cách dễ dàng và hiệu quả.

Các framework này cung cấp nhiều tính năng như xử lý URL, tương tác cơ sở dữ liệu, quản lý phiên và tạo giao diện người dùng. Django được xem là một trong những framework phát triển web phổ biến nhất trong cộng đồng Python. Nền tảng cung cấp sẵn các tính năng linh hoạt và nhiều công cụ hữu ích giúp cho việc xây dựng ứng dụng web phức tạp trở nên thuận lợi hơn.

Tập lệnh Python tự động hóa:

Để tự động hóa các tác vụ bằng Python, có thể sử dụng các tập lệnh và scripts Python để thực hiện các thao tác như quản lý tệp tin, tương tác với cơ sở dữ liệu, gửi email tự động và thậm chí là tự động hóa các tác vụ hệ thống.

Quản lý tệp tin và thư mục: Sử dụng thư viện os để thực hiện các thao tác như tạo, di chuyển, xóa tệp tin và thư mục.

Tương tác với cơ sở dữ liệu: Python có thư viện như SQLAlchemy cho việc tương tác với cơ sở dữ liệu SQL. Công nghệ có các thư viện khác cho các loại cơ sở dữ liệu khác như MongoDB, Redis và nhiều loại cơ sở dữ liệu khác.

Gửi email tự động: Thư viện smtplib có thể được sử dụng để Gửi email bằng Python.

Tự động hóa các tác vụ hệ thống: Bằng cách sử dụng thư viện như subprocess, bạn có thể gọi các lệnh hệ thống và thực hiện các tác vụ như sao chép tệp tin, nén và giải nén tệp tin và thực hiện các lệnh hệ thống khác.

Ứng dụng trong khoa học dữ liệu:

Ngôn ngữ Python được sử dụng phổ biến trong lĩnh vực khoa học dữ liệu và máy học. Cách sử dụng thư viện và framework mạnh mẽ trong Python giúp cho việc phân tích dữ liệu hoặc triển khai mô hình lập trình trở nên dễ dàng hơn. Một số thư viện và framework trong Python:

NumPy: Thư viện mạnh mẽ cho các phép toán trên mảng đa chiều và ma trận, cung cấp các hàm để thao tác dữ liệu số học một cách hiệu quả.

Pandas: Pandas cung cấp cấu trúc dữ liệu và công cụ phân tích dữ liệu mạnh mẽ, đặc biệt là trong việc làm việc với dữ liệu có cấu trúc như bảng và chuỗi thời gian.

Matplotlib và Seaborn: Đây là các thư viện hỗ trợ vẽ đồ thị và trực quan hóa dữ liệu một cách dễ dàng và mạnh mẽ.

Scikit-learn: Scikit-learn là một trong những thư viện học máy phổ biến nhất trong Python, cung cấp nhiều thuật toán lập trình và công cụ cho tiền xử lý dữ liệu, đánh giá mô hình và tinh chỉnh tham số.

TensorFlow và PyTorch: Đây là hai framework phổ biến cho việc triển khai mô hình học sâu (deep learning). Cả hai đều cung cấp API mạnh mẽ để xây dựng và huấn luyện mạng nơ-ron.

1.2. Giới thiệu về Pycharm

1.2.1. Khái niệm về Pycharm

PyCharm là một phần mềm được phát triển bởi JetBrains, cung cấp các công cụ cần thiết giúp các lập trình viên Python tăng năng suất làm việc. Ngoài ra, PyCharm còn được tích hợp nhiều yếu tố mở rộng khác như: biên dịch mã, tô sáng cú pháp, điều hướng project nhanh chóng, công cụ cơ sở dữ liệu và trình soạn thảo văn bản có tích hợp lập trình nhằm mục đích thúc đẩy quá trình phát triển Website.

PyCharm là một trong những IDE được nhiều người dùng ưa chuộng và sử dụng rộng rãi trong giới lập trình. Nhờ sự tiện lợi và hiệu quả cao, PyCharm đã được nhiều doanh nghiệp có tầm ảnh hưởng lớn sử dụng làm IDE Python như: Symantec, Twitter, Pinterest,...

Một trong những điểm mạnh của việc sử dụng PyCharm là khả năng cung cấp API cho các nhà phát triển, đồng thời cho phép họ viết các plugin riêng để mở rộng tính năng. Hơn nữa, phần mềm này còn tương thích với đa dạng các hệ điều hành như Linux, Windows và macOS. Điều này, giúp các lập trình viên Python tiết kiệm thời gian đáng kể trong quá trình phát triển một ứng dụng hay website nào đó.

1.2.2. Đặc điểm nổi bật của Pycharm

Tự động hoàn thành mã: PyCharm cung cấp tính năng gợi ý mã thông minh, giúp hoàn thành mã nhanh chóng và chính xác, tiết kiệm thời gian cho lập trình viên.

Kiểm tra lỗi theo thời gian thực: IDE này có khả năng kiểm tra lỗi ngay trong quá trình viết code, giúp lập trình viên phát hiện và sửa lỗi nhanh chóng.

Hỗ trợ Debugging mạnh mẽ: PyCharm có công cụ gỡ lỗi mạnh mẽ, cung cấp các tính năng như điểm dừng (breakpoints), theo dõi giá trị của biến và bước qua từng dòng lệnh.

Hỗ trợ quản lý dự án và tích hợp hệ thống kiểm soát phiên bản (VCS): PyCharm tích hợp sẵn các công cụ như Git, SVN, và Mercurial, giúp quản lý mã nguồn hiệu quả.

Hỗ trợ cho Web Development: Với PyCharm Professional, bạn có thể phát triển ứng dụng web với các framework phổ biến như Django, Flask, hay Pyramid.

Tích hợp các công cụ phân tích mã nguồn: PyCharm có các công cụ để phân tích và tối ưu hóa mã nguồn, giúp cải thiện chất lượng mã và hiệu suất ứng dụng.

Khả năng mở rộng và tích hợp thư viện: PyCharm hỗ trợ tích hợp trực tiếp với các thư viện và công cụ phổ biến trong Python, chẳng hạn như NumPy, Pandas, TensorFlow, Jupyter Notebook, v.v.

1.3. Giới thiệu về thư viện

1.3.1. NumPy

Thư viện NumPy là một công cụ vô cùng mạnh mẽ và linh hoạt trong lĩnh vực xử lý dữ liệu, đặc biệt là trong các ứng dụng liên quan đến hình ảnh. Được xây dựng trên nền tảng ngôn ngữ C, NumPy cung cấp một bộ các hàm toán học tối ưu hóa, cho phép thực hiện các phép tính trên ma trận và vector với tốc độ cao. Nhờ khả năng thao tác trực tiếp trên các mảng đa chiều, NumPy trở thành công cụ không thể thiếu trong việc chuẩn hóa và xử lý dữ liệu hình ảnh từ các bộ dữ liệu tiêu chuẩn như MNIST. Ngoài ra, việc tích hợp dễ dàng với các thư viện khác như OpenCV và Scikit-learn giúp NumPy trở thành một nền tảng vững chắc cho việc xây dựng các mô hình học máy và xử lý ảnh phức tạp. Với NumPy, các nhà phát triển có thể tăng tốc độ xử lý dữ liệu, giảm thiểu lượng mã nguồn và linh hoạt ứng dụng thư viện này với nhiều loại dữ liệu khác nhau.

1.3.2. OpenCV

Thư viện OpenCV là một công cụ không thể thiếu trong lĩnh vực xử lý ảnh và thị giác máy tính. Được biết đến với kho thư viện khổng lồ và cộng đồng người dùng đông đảo, OpenCV cung cấp hàng ngàn hàm mạnh mẽ, hỗ trợ đa dạng các tác vụ xử lý ảnh từ cơ bản đến nâng cao. Với khả năng chuyển đổi ảnh màu sang thang độ xám, phân ngưỡng và tìm kiếm các vùng chứa chữ số một cách hiệu quả, OpenCV trở thành công cụ đắc lực trong các ứng dụng nhận dạng ký tự quang học (OCR). Đặc biệt, khả năng vẽ bounding box để trực quan hóa kết quả dự đoán giúp cho việc đánh giá và kiểm tra mô hình trở nên dễ dàng hơn.

Một trong những điểm mạnh của OpenCV là khả năng tích hợp tốt với NumPy, cho phép người dùng tận dụng tối đa sức mạnh của các phép toán ma trận trong việc xử lý ảnh. Bên cạnh đó, OpenCV còn hỗ trợ đa dạng các định dạng ảnh và video, giúp tăng tính linh hoạt trong các ứng dụng thực tế. Nhờ vào việc tối ưu hóa các thuật toán, OpenCV giúp tiết kiệm đáng kể thời gian xử lý ảnh, đồng thời hỗ trợ đa nền tảng, dễ dàng tích hợp vào các ứng dụng khác nhau.

Với những ưu điểm vượt trội, OpenCV không chỉ phù hợp cho các bài toán nhỏ như nhận dạng chữ số mà còn có thể ứng dụng trong các bài toán lớn hơn như nhận diện khuôn mặt, theo dõi vật thể và nhiều ứng dụng khác.

1.3.3. Scikit-learn

Scikit-learn là một thư viện học máy mạnh mẽ và linh hoạt, đặc biệt phù hợp cho những ai mới bắt đầu khám phá lĩnh vực này. Với giao diện trực quan và dễ sử dụng, Scikit-learn cung cấp một bộ công cụ đầy đủ để xây dựng và đánh giá các mô hình học máy, từ việc tiền xử lý dữ liệu đến việc triển khai mô hình vào thực tế. Trong bài toán nhận dạng chữ số, Scikit-learn cho phép chúng ta dễ dàng huấn luyện mô hình LinearSVC dựa trên các vector đặc trưng HOG (Histogram of Oriented Gradients) và đánh giá hiệu năng của mô hình thông qua các chỉ số như độ chính xác và ma trận nhầm lẫn.

Một trong những ưu điểm nổi bật của Scikit-learn là khả năng tích hợp nhiều thuật toán học máy cổ điển như SVM, KNN, Decision Tree, Random Forest, giúp người dùng có nhiều lựa chọn để xây dựng mô hình phù hợp với từng bài toán cụ thể. Bên cạnh đó, Scikit-learn

còn cung cấp các công cụ đánh giá hiệu năng mô hình một cách chi tiết, giúp chúng ta hiểu rõ hơn về điểm mạnh, điểm yếu của mô hình và từ đó đưa ra các cải tiến phù hợp.

Với Scikit-learn, việc xây dựng một hệ thống nhận dạng chữ số trở nên đơn giản hơn bao giờ hết. Người dùng chỉ cần tập trung vào việc chuẩn bị dữ liệu, lựa chọn thuật toán phù hợp và điều chỉnh các hyperparameter. Scikit-learn sẽ lo liệu phần còn lại, từ việc tính toán các vector đặc trưng, huấn luyện mô hình đến việc đánh giá kết quả.

1.3.4. Scikit-image

Scikit-image là một công cụ mạnh mẽ và linh hoạt trong lĩnh vực xử lý ảnh, đặc biệt hữu ích trong việc trích xuất các đặc trưng hình ảnh. Với Scikit-image, việc tính toán histogram hướng gradient (HOG) từ ảnh chữ số viết tay trở nên đơn giản và hiệu quả. HOG là một phương pháp đặc trưng rất mạnh, giúp mô tả hình dạng và kết cấu của đối tượng một cách chính xác, từ đó cải thiện đáng kể hiệu suất của các mô hình học máy.

Một trong những điểm mạnh của Scikit-image là khả năng tích hợp sâu với NumPy và Scikit-learn. Điều này giúp cho việc xây dựng các pipeline xử lý ảnh trở nên liền mạch và hiệu quả. Ngoài HOG, Scikit-image còn cung cấp một loạt các thuật toán xử lý ảnh khác như biến đổi Fourier, lọc ảnh, phát hiện biên, giúp người dùng thực hiện các tiền xử lý cần thiết trước khi trích xuất đặc trưng. Giao diện trực quan và dễ sử dụng của Scikit-image cũng là một lợi thế lớn, giúp các nhà khoa học dữ liệu và kỹ sư dễ dàng làm việc với thư viện này.

1.3.5. Keras

Keras là một thư viện học sâu cấp cao được xây dựng trên nền tảng TensorFlow, nổi tiếng với giao diện trực quan và dễ sử dụng. Mặc dù Keras có nhiều tính năng mạnh mẽ để xây dựng các mô hình học sâu phức tạp, nhưng trong bài toán nhận dạng chữ số viết tay, chúng ta chỉ tập trung vào việc sử dụng Keras để tải bộ dữ liệu MNIST.

MNIST là một trong những bộ dữ liệu tiêu chuẩn được sử dụng rộng rãi trong lĩnh vực học máy và học sâu. Bộ dữ liệu này chứa hàng nghìn hình ảnh chữ số viết tay, được phân chia thành hai tập: tập huấn luyện và tập kiểm tra. Một trong những ưu điểm lớn của Keras là nó cung cấp sẵn bộ dữ liệu MNIST, giúp chúng ta tiết kiệm thời gian và công sức trong việc tải và chuẩn bị dữ liệu.

1.3.6. Tkinter

Tkinter là một thư viện xây dựng giao diện đồ họa (GUI) tích hợp sẵn trong Python, cung cấp một cách đơn giản và hiệu quả để tạo ra các ứng dụng có giao diện người dùng trực quan. Với Tkinter, bạn có thể dễ dàng xây dựng các cửa sổ, nút bấm, hộp nhập liệu, và các thành phần giao diện khác để tương tác với người dùng.

Trong lĩnh vực học máy và xử lý ảnh, Tkinter thường được sử dụng để xây dựng giao diện cho các ứng dụng nhận dạng hình ảnh. Ví dụ, bạn có thể sử dụng Tkinter để tạo một ứng dụng cho phép người dùng tải lên một bức ảnh, sau đó ứng dụng sẽ sử dụng mô hình học máy đã được huấn luyện để nhận dạng các đối tượng trong ảnh và hiển thị kết quả trực tiếp trên giao diện. Ngoài ra, Tkinter còn có thể được sử dụng để tạo các công cụ đánh giá hiệu năng của mô hình, cho phép người dùng trực quan hóa các kết quả dự đoán và so sánh với nhãn thực tế.

1.3.7. Pillow (PIL)

Pillow (PIL Fork) là một thư viện xử lý ảnh mạnh mẽ và dễ sử dụng trong Python. Nó cung cấp một bộ công cụ phong phú để thực hiện các thao thao tác cơ bản trên hình ảnh như mở, lưu, hiển thị, thay đổi kích thước, cắt xén, xoay, và chuyển đổi định dạng. Trong đó, hai ứng dụng phổ biến nhất của Pillow là chuyển đổi ảnh sang các định dạng cần thiết (như RGB, grayscale) và hiển thị ảnh trong giao diện Tkinter.

Khi làm việc với các dự án liên quan đến xử lý ảnh và xây dựng giao diện người dùng, Pillow đóng vai trò rất quan trọng. Nó giúp bạn dễ dàng chuyển đổi ảnh sang các định dạng phù hợp với yêu cầu của ứng dụng, đồng thời cung cấp các công cụ để hiển thị ảnh một cách trực quan trên giao diện Tkinter. Nhờ đó, bạn có thể tạo ra các ứng dụng xử lý ảnh với giao diện người dùng thân thiện và chuyên nghiệp.

1.4. Tổng quan về xử lý ảnh và thị giác máy tính

1.4.1. Khái niện về xử lý ảnh và thị giác máy tính

Computer Vision (Thị giác máy tính) là một lĩnh vực khoa học máy tính tập trung vào việc cho phép máy tính "nhìn" và "hiểu" nội dung của hình ảnh hoặc video. Nói cách khác, nó giúp máy tính có khả năng trích xuất thông tin hữu ích từ dữ liệu hình ảnh. (Nguồn tham khảo: https://200lab.io/blog/computer-vision-la-gi)

Xử lý ảnh kỹ thuật số (Digital Image Processing) là việc sử dụng máy tính để phân tích và cải thiện ảnh kỹ thuật số thông qua các thuật toán. Phân ngành này có ưu điểm hơn xử lý ảnh analog nhờ khả năng áp dụng nhiều thuật toán và giảm thiểu các vấn đề như nhiễu và méo hình. Được định nghĩa trên không gian hai hoặc nhiều chiều, xử lý ảnh kỹ thuật số được mô hình hóa như một hệ thống đa chiều. Lĩnh vực này phát triển nhờ ba yếu tố chính: tiến bộ máy tính, cải tiến toán học (đặc biệt trong lý thuyết toán rời rạc), và nhu cầu ứng dụng trong nhiều lĩnh vực như môi trường, y tế, và công nghiệp. (Nguồn tham khảo: https://vi.wikipedia.org/wiki/Xử_lý_ảnh)

1.4.2. Lĩnh vực ứng dụng, mục tiêu liên quan đến xử lý ảnh và thị giác máy tính

1.4.2.1. Nhận diện khuôn mặt (Face Recognition)

Ứng dụng: Nhận diện khuôn mặt được sử dụng trong nhiều lĩnh vực như mở khóa điện thoại, bảo mật hệ thống, và kiểm tra danh tính.

Ví dụ**:** Hệ thống mở khóa bằng khuôn mặt trên điện thoại thông minh (Face ID của iPhone), kiểm soát an ninh tại sân bay và cửa khẩu.

1.4.2.2. Mạng xã hội (Social Media)

Ứng dụng: Thị giác máy tính giúp tự động gắn thẻ bạn bè trong ảnh, cải thiện tính năng tìm kiếm bằng hình ảnh, và xử lý ảnh để tạo ra các bộ lọc, hiệu ứng thú vị.

Ví dụ: Facebook, Instagram, và Snapchat sử dụng công nghệ này để nhận diện khuôn mặt và áp dụng các bộ lọc thời gian thực (real-time filters).

1.4.2.3. Tìm kiếm hình ảnh (Image Search)

Ứng dụng: Các công cụ tìm kiếm hình ảnh cho phép người dùng tìm kiếm sản phẩm hoặc nội dung tương tự chỉ bằng cách tải lên một hình ảnh.

Ví dụ: Google Images và các nền tảng thương mại điện tử như Amazon, Shopee, cho phép tìm kiếm sản phẩm thông qua ảnh thay vì từ khóa.

1.4.2.4. Xe tự lái (Autonomous Vehicles)

Ứng dụng: Thị giác máy tính giúp xe tự hành "nhìn" và phân tích môi trường xung quanh như làn đường, biển báo, các phương tiện khác và người đi bộ để đưa ra quyết định lái xe an toàn.

Ví dụ: Các hãng xe như Tesla, Waymo sử dụng công nghệ này để phát triển xe tự lái.

1.4.2.5. Bảo mật và giám sát (Security and Surveillance)

Ứng dụng**:** Camera an ninh sử dụng thị giác máy tính để giám sát, phát hiện hành vi đáng ngờ và nhận diện khuôn mặt để cảnh báo sớm về các mối nguy hiểm.

Ví dụ: Hệ thống giám sát tại nhà thông minh như Ring, Nest Cam có khả năng phát hiện chuyển động và cảnh báo khi có người lạ tiếp cận.

1.4.2.6. Mua sắm thông minh (Smart Shopping)

Ứng dụng: Các hệ thống sử dụng thị giác máy tính để theo dõi hàng hóa trong siêu thị, tối ưu hóa quy trình thanh toán mà không cần quét mã vạch từng sản phẩm.

Ví dụ**:** Amazon Go sử dụng thị giác máy tính để khách hàng có thể mua sắm mà không cần phải qua quầy thanh toán (just walk out technology).

1.4.2.7. Y tế (Healthcare)

Ứng dụng**:** Trong y tế, thị giác máy tính được sử dụng để phân tích hình ảnh chụp từ các thiết bị y khoa như X-quang, MRI để hỗ trợ bác sĩ trong việc chẩn đoán bệnh.

Ví dụ: Phần mềm phân tích ảnh y tế có thể phát hiện ung thư sớm hoặc các bệnh lý từ ảnh chụp tế bào.

1.4.2.8. Thực tế ảo (Virtual Reality) và Thực tế tăng cường (Augmented Reality)

Ứng dụng: Các ứng dụng AR và VR sử dụng thị giác máy tính để tạo ra trải nghiệm tương tác trong không gian ảo hoặc đưa các đối tượng ảo vào thế giới thực.

Ví dụ: Game Pokémon Go sử dụng AR để đưa các nhân vật ảo vào môi trường thực, hay các ứng dụng nội thất như IKEA Place giúp người dùng ước lượng đồ đạc trong không gian nhà của họ.

1.4.2.9. Ứng dụng giao thông thông minh (Smart Traffic Applications)

Ứng dụng**:** Thị giác máy tính được áp dụng để giám sát giao thông, nhận diện biển số xe, phát hiện vi phạm giao thông, và quản lý tín hiệu giao thông tự động.

Ví dụ: Các thành phố thông minh sử dụng camera và công nghệ nhận diện biển số để xử lý phạt nguội hoặc điều tiết giao thông.

1.4.2.10. Dịch vụ khách hàng tự động (Automated Customer Service)

Ứng dụng**:** Thị giác máy tính có thể được tích hợp trong các hệ thống tự động để nhận diện người dùng và cung cấp dịch vụ một cách tự động, thông minh.

Ví dụ: Máy ATM hiện đại có thể nhận diện khuôn mặt để tăng cường bảo mật, hoặc robot hỗ trợ khách hàng trong các cửa hàng bán lẻ.

1.4.2.11. Nông nghiệp thông minh (Smart Agriculture)

Ứng dụng: Trong nông nghiệp, thị giác máy tính được sử dụng để giám sát cây trồng, phát hiện dịch bệnh, và quản lý quá trình thu hoạch hiệu quả.

Ví dụ: Máy bay không người lái (drone) sử dụng thị giác máy tính để quét và phân tích sức khỏe cây trồng trên quy mô lớn.

Chương 2: Xây dựng chương trình

2.1. Giới thiệu bài toán

2.1.1. Phát biểu bài toán

Mô tả bài toán: Nhận dạng chữ số viết tay là một bài toán trong lĩnh vực học máy, với mục tiêu xây dựng một hệ thống có thể phân loại chính xác các chữ số viết tay từ 0 đến 9 dựa trên dữ liệu hình ảnh. Hệ thống sẽ sử dụng các phương pháp xử lý ảnh và mô hình học máy để trích xuất đặc trưng từ các hình ảnh và thực hiện dự đoán.

Yêu cầu bài toán:

Input: Một hình ảnh đầu vào chứa các chữ số viết tay.

Output: Dự đoán chính xác các chữ số trong hình ảnh đầu vào, hiển thị kết quả trực tiếp lên hình ảnh hoặc qua giao diện người dùng.

Mục tiêu:

- Xây dựng mô hình có độ chính xác cao trong việc nhận dạng chữ số viết tay.
- Hiển thị giao diện đơn giản, dễ sử dụng để thực hiện dự đoán từ hình ảnh.

Các bước chính để giải quyết bài toán:

Bước 1: Tiền xử lý dữ liệu. Đầu tiên, hình ảnh đầu vào sẽ được chuẩn hóa về kích thước và chuyển đổi sang ảnh xám để giảm thiểu nhiễu. Sau đó, chúng ta sẽ trích xuất các đặc trưng HOG (Histogram of Oriented Gradients) từ hình ảnh. HOG là một mô tả đặc trưng mạnh mẽ, giúp mô tả hình dạng và kết cấu của đối tượng một cách hiệu quả. Các đặc trưng HOG này sẽ được sử dụng làm đầu vào cho mô hình phân loại.

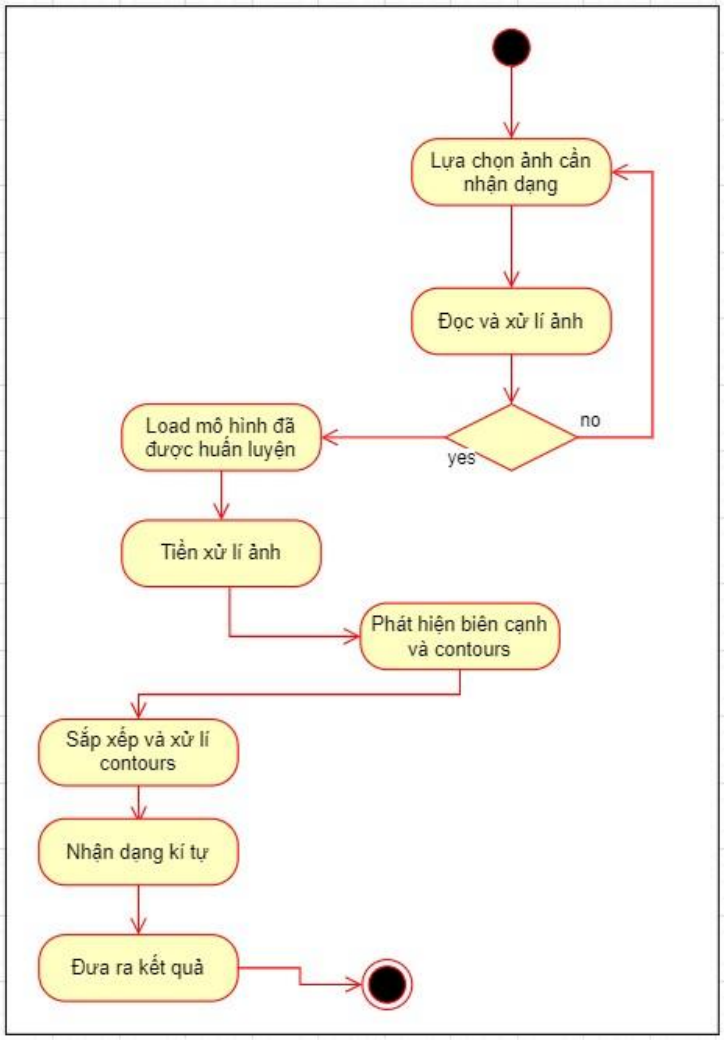
Bước 2: Xây dựng mô hình. Chúng ta sẽ sử dụng thuật toán SVM (Support Vector Machine) để xây dựng mô hình phân loại. SVM là một thuật toán học máy mạnh mẽ, thường được sử dụng để giải quyết các bài toán phân loại nhị phân và đa lớp. Mô hình SVM sẽ được huấn luyện trên bộ dữ liệu MNIST, với các đặc trưng HOG làm đầu vào và nhãn tương ứng là các chữ số từ 0 đến 9.

Bước 3: Hiển thị kết quả. Sau khi huấn luyện xong mô hình, chúng ta sẽ sử dụng OpenCV để thực hiện dự đoán trên hình ảnh đầu vào. Kết quả dự đoán sẽ được vẽ trực tiếp lên hình

ảnh ban đầu, giúp người dùng dễ dàng so sánh giữa hình ảnh gốc và kết quả dự đoán. Để tạo ra một giao diện thân thiện với người dùng, chúng ta sẽ sử dụng thư viện Tkinter để xây dựng một ứng dụng đơn giản. Ứng dụng này sẽ cho phép người dùng tải lên hình ảnh, thực hiện dự đoán và đánh giá hiệu suất của mô hình một cách trực quan.

2.1.2. Các bước tiến hành bài toán

Để giải quyết bài toán nhận dạng chữ số viết tay, chúng ta thực hiện qua các bước chính sau đây:

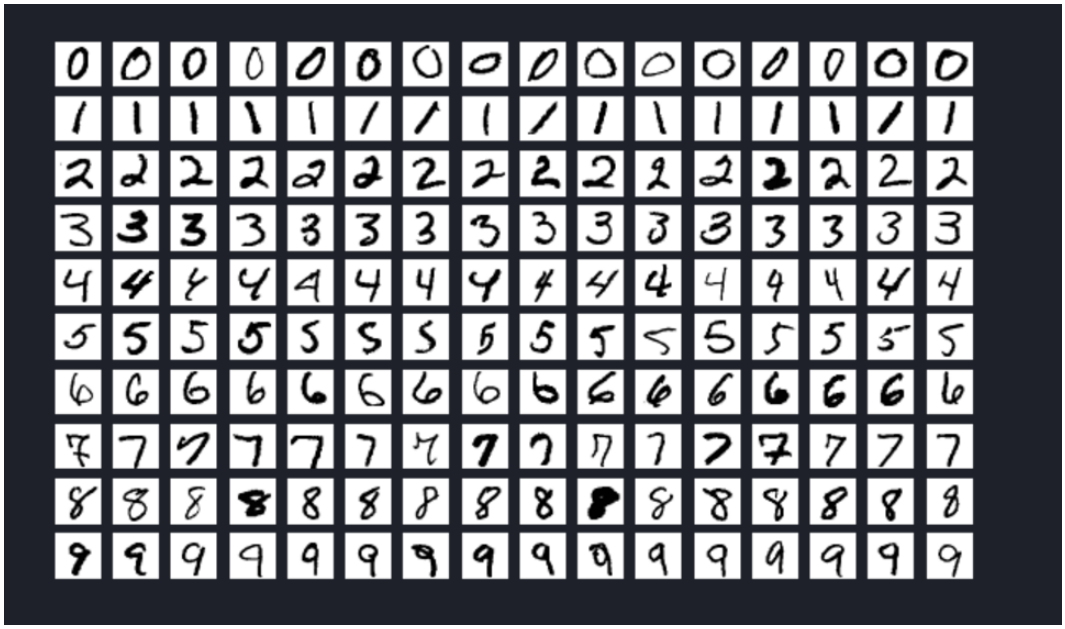


Hình 2.1 Sơ đồ hoạt động

Để giải quyết bài toán nhận dạng chữ số viết tay, chúng ta sẽ thực hiện các bước sau:

Chuẩn bị dữ liệu: Đầu tiên, chúng ta sẽ tải bộ dữ liệu MNIST, một trong những bộ dữ liệu tiêu biểu cho bài toán này. Bộ dữ liệu này chứa hàng nghìn hình ảnh chữ số viết tay đã

được phân loại. Sau đó, chúng ta sẽ chia dữ liệu thành hai tập: tập huấn luyện (sử dụng để huấn luyện mô hình) và tập kiểm tra (sử dụng để đánh giá hiệu suất của mô hình).



Hình 2.2 Bộ dữ liệu MNIST

Trích xuất đặc trưng: Tiếp theo, chúng ta sẽ sử dụng kỹ thuật HOG (Histogram of Oriented Gradients) để trích xuất các đặc trưng từ hình ảnh. HOG là một phương pháp hiệu quả giúp chúng ta mô tả hình dạng và kết cấu của các chữ số viết tay. Nhờ đó, mô hình học máy có thể dễ dàng phân biệt các chữ số khác nhau.

Xây dựng và huấn luyện mô hình: Dựa trên các đặc trưng HOG đã trích xuất, chúng ta sẽ xây dựng một mô hình phân loại sử dụng thuật toán SVM (Support Vector Machine). SVM là một thuật toán mạnh mẽ, thường được sử dụng để giải quyết các bài toán phân loại. Mô hình SVM sẽ được huấn luyện trên tập dữ liệu huấn luyện để học cách phân biệt các chữ số.

Đánh giá mô hình: Sau khi huấn luyện, chúng ta sẽ sử dụng tập kiểm tra để đánh giá hiệu suất của mô hình. Các chỉ số đánh giá phổ biến bao gồm độ chính xác và ma trận nhầm lẫn. Độ chính xác cho biết tỷ lệ các dự đoán đúng của mô hình, trong khi ma trận nhầm lẫn cho ta biết mô hình thường nhầm lẫn các chữ số nào với nhau.

Phát triển giao diện người dùng: Cuối cùng, chúng ta sẽ xây dựng một giao diện đơn giản bằng thư viện Tkinter. Giao diện này sẽ cho phép người dùng tải lên hình ảnh chữ số

viết tay mà họ muốn nhận dạng. Sau khi người dùng tải ảnh lên, hệ thống sẽ tự động xử lý hình ảnh, thực hiện dự đoán và hiển thị kết quả trực tiếp trên màn hình. Ngoài ra, giao diện cũng sẽ hiển thị các thông số đánh giá của mô hình để người dùng có thể theo dõi hiệu suất của hệ thống.

2.2. Mô tả thuật toán

Bước 1: Trích xuất đặc trưng HOG từ hình ảnh

HOG (Histogram of Oriented Gradients) là một phương pháp hiệu quả để trích xuất các đặc trưng hình ảnh. Đặc trưng HOG tập trung vào việc mô tả hình dạng và cấu trúc của một đối tượng bằng cách phân tích hướng của các đường viền và gradient trong hình ảnh.

Quá trình trích xuất đặc trưng HOG bao gồm các bước sau:

Chia ảnh thành các ô: Hình ảnh ban đầu được chia thành các ô nhỏ. Mỗi ô này sẽ là đơn vị cơ bản để tính toán histogram.

Tính toán histogram: Trong mỗi ô, chúng ta tính toán histogram của các hướng gradient. Histogram này biểu diễn sự phân bố của các hướng gradient tại các điểm ảnh trong ô.

Chuẩn hóa histogram: Để tăng cường tính ổn định và giảm ảnh hưởng của các thay đổi về ánh sáng, chúng ta tiến hành chuẩn hóa các histogram. Quá trình chuẩn hóa thường được thực hiện trên các khối gồm nhiều ô.

Tạo vector đặc trưng: Cuối cùng, các histogram từ tất cả các ô được ghép lại thành một vector đặc trưng. Vector này sẽ đại diện cho toàn bộ hình ảnh và được sử dụng làm đầu vào cho các thuật toán học máy như SVM để thực hiện các nhiệm vụ như phân loại, phát hiện đối tượng.

Công thức toán học: Gradient của một pixel (x,y) được tính theo:

$$G_x = I(x+1, y) - I(x-1, y)$$

$$G_y = I(x, y+1) - I(x, y-1)$$

Với:

- G_x: Gradient theo trục x.
- G_y: Gradient theo trục y.
- $I(x, y)$: Giá trị cường độ pixel tại tọa độ (x, y).

Hình 2.3 Công thức toán Gradient

Bước 2: Huấn luyện mô hình SVM

Sau khi đã trích xuất được các đặc trưng HOG từ hình ảnh, chúng ta sẽ sử dụng thuật toán LinearSVC để huấn luyện một mô hình phân loại. LinearSVC được lựa chọn vì một số lý do sau:

Hiệu quả với dữ liệu có chiều cao: Khi chúng ta trích xuất đặc trưng HOG, mỗi hình ảnh sẽ được biểu diễn bởi một vector đặc trưng có chiều khá cao. LinearSVC tỏ ra hiệu quả trong việc xử lý các dữ liệu có chiều cao như vậy.

Tốc độ huấn luyện nhanh: So với các kernel khác của SVM, LinearSVC thường có tốc độ huấn luyện nhanh hơn, đặc biệt khi làm việc với các tập dữ liệu lớn.

Dễ hiểu và triển khai: LinearSVC có cơ chế hoạt động tương đối đơn giản, dễ dàng để hiểu và triển khai.

Quá trình huấn luyện LinearSVC:

Xây dựng mô hình: Chúng ta tạo một đối tượng LinearSVC từ thư viện scikit-learn.

Huấn luyện: Đưa dữ liệu huấn luyện (các vector đặc trưng HOG và nhãn tương ứng) vào mô hình để huấn luyện. Trong quá trình huấn luyện, mô hình sẽ tìm kiếm một siêu phẳng (hyperplane) tối ưu để phân chia dữ liệu thành các lớp khác nhau.

Tối ưu hóa hàm mất mát: Mục tiêu của quá trình huấn luyện là tìm ra siêu phẳng sao cho khoảng cách giữa siêu phẳng này và các điểm dữ liệu gần nó nhất (support vectors) là lớn nhất. Điều này tương đương với việc tối ưu hóa một hàm mất mát cụ thể.

Bước 3: Dự đoán trên ảnh thực tế

Sau khi đã huấn luyện mô hình SVM, chúng ta sẽ tiến hành dự đoán các chữ số trên một hình ảnh bất kỳ. Quá trình này bao gồm một số bước tiền xử lý và phân tích hình ảnh.

Tiền xử lý ảnh: Đầu tiên, hình ảnh đầu vào sẽ được chuyển đổi sang thang độ xám để đơn giản hóa quá trình xử lý. Tiếp theo, chúng ta sẽ áp dụng bộ lọc Gaussian Blur để làm mờ hình ảnh và giảm nhiễu. Cuối cùng, quá trình phân ngưỡng sẽ được sử dụng để tách các vùng chứa chữ số khỏi nền.

Phân đoạn ảnh: Để xác định chính xác vị trí của từng chữ số, chúng ta sẽ sử dụng kỹ thuật Contour Detection. Contour Detection sẽ giúp tìm các đường viền bao quanh các vùng có màu sắc đồng nhất trong ảnh, từ đó xác định các vùng chứa chữ số. Sau khi tìm được

các contour, chúng ta sẽ xác định bounding box (hình chữ nhật bao quanh) cho mỗi vùng chữ số.

Dự đoán từng chữ số: Đối với mỗi vùng chữ số đã được xác định, chúng ta sẽ trích xuất đặc trưng HOG tương tự như quá trình đã làm với dữ liệu huấn luyện. Sau đó, vector đặc trưng này sẽ được đưa vào mô hình SVM đã được huấn luyện để dự đoán chữ số tương ứng.

Hiển thị kết quả: Cuối cùng, kết quả dự đoán sẽ được hiển thị trực quan trên hình ảnh gốc. Chúng ta sẽ vẽ các hình chữ nhật bao quanh các vùng chữ số đã được xác định và ghi nhãn chữ số dự đoán vào bên trong mỗi hình chữ nhật. Điều này giúp người dùng dễ dàng so sánh giữa kết quả dự đoán và hình ảnh gốc.

2.3. Cài đặt chương trình

2.3.1. Chuẩn bị thư viện và môi trường

Trước khi bắt đầu, cần đảm bảo các thư viện cần thiết đã được cài đặt.

Thư viện sử dụng:

Tkinter: Tạo giao diện đồ họa (GUI).

Pillow (PIL): Xử lý hình ảnh cho giao diện.

OpenCV: Xử lý ảnh đầu vào (chuyển đổi thang độ xám, phân ngưỡng, tìm contour).

Scikit-learn: Huấn luyện mô hình SVM và tính toán độ chính xác, ma trận nhầm lẫn.

Keras: Tải dữ liệu chữ số viết tay MNIST.

NumPy và skimage: Hỗ trợ tính toán đặc trưng HOG.

2.3.2. Tải dữ liệu và huấn luyện mô hình

- Dữ liệu chữ số viết tay được tải từ MNIST, sau đó trích xuất đặc trưng HOG và huấn luyện mô hình SVM.

Mã nguồn:

```
import tkinter as tk
from tkinter import ttk, filedialog, messagebox
from PIL import Image, ImageTk
import numpy as np
from skimage.feature import hog
from sklearn.svm import LinearSVC
from sklearn.metrics import accuracy_score, confusion_matrix
import cv2
from keras.datasets import mnist
```

```
# Load dữ liệu MNIST và huấn luyện mô hình
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train_feature = [hog(x, orientations=9, pixels_per_cell=(14, 14),
cells_per_block=(1, 1), block_norm="L2") for x in x_train]
x_test_feature = [hog(x, orientations=9, pixels_per_cell=(14, 14),
cells_per_block=(1, 1), block_norm="L2") for x in x_test]

x_train_feature = np.array(x_train_feature, dtype=np.float32)
x_test_feature = np.array(x_test_feature, dtype=np.float32)

model = LinearSVC(C=10)
model.fit(x_train_feature, y_train)
y_pred = model.predict(x_test_feature)

accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
```

Giải thích:

1. **MNIST:** Bộ dữ liệu gồm 70,000 ảnh chữ số viết tay từ 0-9.
2. **HOG:** Trích xuất đặc trưng từ mỗi ảnh để mô tả hướng gradient.
3. **LinearSVC:** Huấn luyện mô hình phân loại chữ số với tham số C=10.

2.3.3. Dự đoán chữ số từ ảnh

- Ảnh đầu vào được tiền xử lý (chuyển thang độ xám, phân ngưỡng) và dự đoán từng chữ số.

Mã nguồn:

```
# Hàm xử lý và dự đoán chữ số từ ảnh
def process_and_predict(image_path):
    image = cv2.imread(image_path)
    im_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    im_blur = cv2.GaussianBlur(im_gray, (5, 5), 0)
    _, thre = cv2.threshold(im_blur, 90, 255, cv2.THRESH_BINARY_INV)

    # Tìm các vùng chứa chữ số
    contours, _ = cv2.findContours(thre, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
    rects = [cv2.boundingRect(cnt) for cnt in contours]

    for (x, y, w, h) in rects:
        cv2.rectangle(image, (x, y), (x + w, y + h), (255, 255, 0), 2)
        roi = thre[y:y + h, x:x + w]
        roi = np.pad(roi, ((20, 20), (20, 20)), 'constant', constant_values=0)
        roi = cv2.resize(roi, (28, 28), interpolation=cv2.INTER_AREA)

        # Tính toán đặc trưng HOG
        roi_hog_fd = hog(roi, orientations=9, pixels_per_cell=(14, 14),
cells_per_block=(1, 1), block_norm="L2")
        nbr = model.predict(np.array([roi_hog_fd], np.float32))
```

```
        # Hiển thị kết quả dự đoán trên ảnh
        cv2.putText(image, str(int(nbr[0])), (x, y - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.9, (255, 255, 0), 2)

    cv2.imshow("ket qua du doan", image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

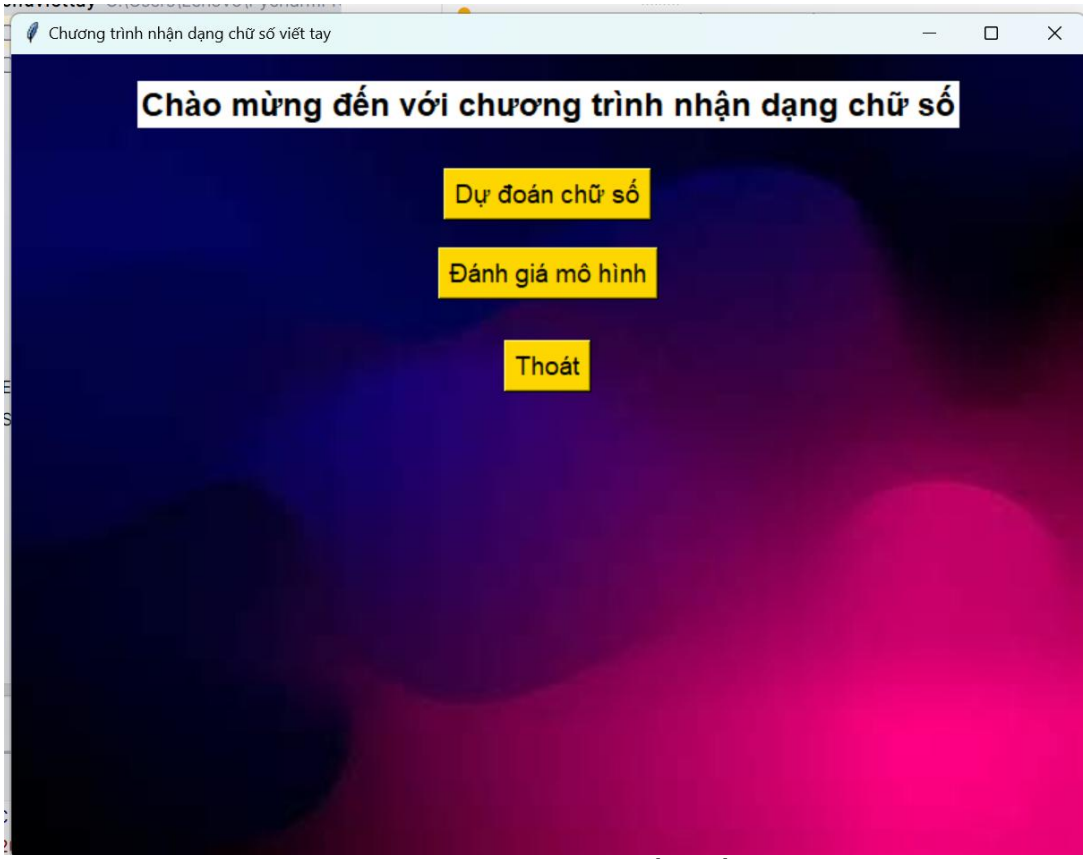
2.3.4. Giao diện chính

- Giao diện bao gồm các nút:
    1. "Dự đoán chữ số": Tải ảnh và thực hiện dự đoán.
    2. "Đánh giá mô hình": Hiển thị độ chính xác và ma trận nhầm lẫn.
    3. "Thoát": Thoát khỏi chương trình.

Mã nguồn:

```
# Hàm mở file ảnh
def open_file():
    file_path = filedialog.askopenfilename(filetypes=[("Image files",
"*.jpg;*.png;*.bmp")])
    if file_path:
        process_and_predict(file_path)

# Hàm tạo giao diện đánh giá
def create_evaluation_window():
    eval_window = tk.Toplevel(root)
    eval_window.title("Đánh giá mô hình")
    eval_window.geometry("800x600")

    bg_image = Image.open("background.png")
    bg_image = bg_image.resize((800, 600), Image.LANCZOS)
    bg_photo = ImageTk.PhotoImage(bg_image)
    bg_label = tk.Label(eval_window, image=bg_photo)
    bg_label.image = bg_photo
    bg_label.place(x=0, y=0, relwidth=1, relheight=1)

    title = tk.Label(eval_window, text="Đánh giá mô hình SVM", font=("Arial",
18, "bold"), bg="#FFFFFF")
    title.pack(pady=10)

    acc_label = tk.Label(eval_window, text=f"Độ chính xác: {accuracy:.2%}",
font=("Arial", 14), bg="#FFFFFF")
    acc_label.pack(pady=5)

    cm_title = tk.Label(eval_window, text="Ma trận nhầm lẫn:", font=("Arial",
14), bg="#FFFFFF")
    cm_title.pack(pady=5)

    cm_frame = tk.Frame(eval_window, bg="#FFFFFF")
    cm_frame.pack(pady=10)

    tree = ttk.Treeview(cm_frame, columns=[str(i) for i in range(10)],
```

```
show="headings", height=10)
    for i in range(10):
        tree.heading(str(i), text=str(i))
        tree.column(str(i), width=50, anchor="center")

    for i, row in enumerate(conf_matrix):
        tree.insert("", "end", values=[str(x) for x in row])

    tree.pack()

    close_button = tk.Button(eval_window, text="Đóng",
command=eval_window.destroy, font=("Arial", 12))
    close_button.pack(pady=10)

# Giao diện chính
root = tk.Tk()
root.title("Chương trình nhận dạng chữ số viết tay")
root.geometry("800x600")

bg_image = Image.open("backgr3.png")
bg_image = bg_image.resize((800, 600), Image.LANCZOS)
bg_photo = ImageTk.PhotoImage(bg_image)
bg_label = tk.Label(root, image=bg_photo)
bg_label.image = bg_photo
bg_label.place(x=0, y=0, relwidth=1, relheight=1)

title = tk.Label(root, text="Chào mừng đến với chương trình nhận dạng chữ số",
font=("Arial", 18, "bold"), bg="#FFFFFF")
title.pack(pady=20)

pred_button = tk.Button(root, text="Dự đoán chữ số", command=open_file,
font=("Arial", 14), bg="#FFD700")
pred_button.pack(pady=10)

eval_button = tk.Button(root, text="Đánh giá mô hình",
command=create_evaluation_window, font=("Arial", 14), bg="#FFD700")
eval_button.pack(pady=10)

exit_button = tk.Button(root, text="Thoát", command=root.quit, font=("Arial",
14), bg="#FFD700")
exit_button.pack(pady=20)

root.mainloop()
```



Hình 2.4 Giao diện phần mềm

Bảng 2.1 Các phần tử trong giao diện chương trình

Tên phần tử	Loại	Mô tả
Title	Text	
Dự đoán chứ số	Button	Ấn vào để chọn ảnh từ thư viện để nhận diện chữ số từ ảnh
Đánh giá mô hình	Button	Ấn vào để đánh giá độ chính xác và đưa ra ma trận nhầm lẫn
Thoát	Buttom	Ấn vào để thoát khỏi chương trình

Chương 3: Kết quả thực nghiệm chương trình

3.1. Kết quả huấn luyện và kiểm tra

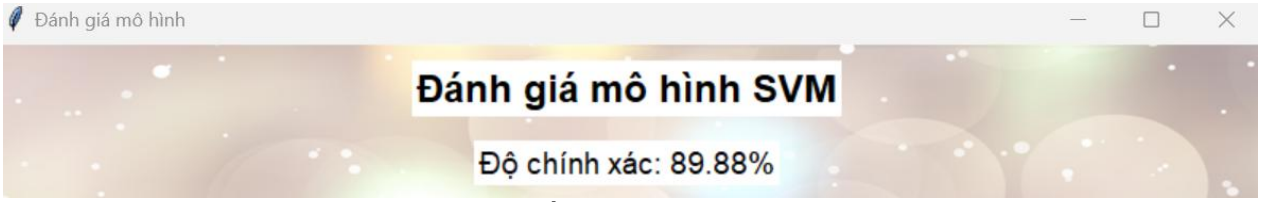
Ở giao diện chính chọn buttom đánh giá mô hình → sẽ hiện ra giao diện có 2 kiểu đánh giá: đánh giá độ chính xác và đánh giá ma trận nhầm lẫn

Độ chính xác của mô hình: Sau khi huấn luyện mô hình **LinearSVC** với dữ liệu **MNIST**, mô hình đạt được **độ chính xác (accuracy)** như sau:

Độ chính xác trên tập kiểm tra :

accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
Cách tính độ chính xác:
Trong Python, khi sử dụng hàm accuracy_score từ thư viện sklearn.metrics, công thức để tính **độ chính xác (accuracy)** là:

Accuracy= $\frac{Số\ lượng\ dự\ đoán}{Tổng\ số\ dự\ đoán}$



Hình 3.1 Kết quả đánh giá mô hình

Nhận xét:

- Đây là một kết quả khả quan đối với một mô hình dựa trên trích xuất đặc trưng HOG và Linear SVM.
- Tuy nhiên, hiệu suất này có thể thấp hơn các mô hình Deep Learning (như CNN), vốn có khả năng tự động học đặc trưng từ dữ liệu.

Ma trận nhầm lẫn (Confusion Matrix): Ma trận nhầm lẫn cho biết số lần dự đoán đúng và sai giữa các lớp chữ số (0–9).

conf_matrix = confusion_matrix(y_test, y_pred)

0	1	2	3	4	5	6	7	8	9
946	3	1	2	4	5	2	3	4	10
0	1110	4	0	11	0	5	1	4	0
18	2	916	59	11	0	3	10	11	2
8	0	54	892	7	11	1	13	11	13
9	24	12	2	857	4	36	1	15	22
3	0	0	32	3	800	4	2	32	16
11	4	3	1	18	12	892	0	14	3
9	7	45	6	11	2	0	887	15	46
9	4	11	9	13	7	36	11	861	13
27	14	2	28	20	23	1	30	37	827

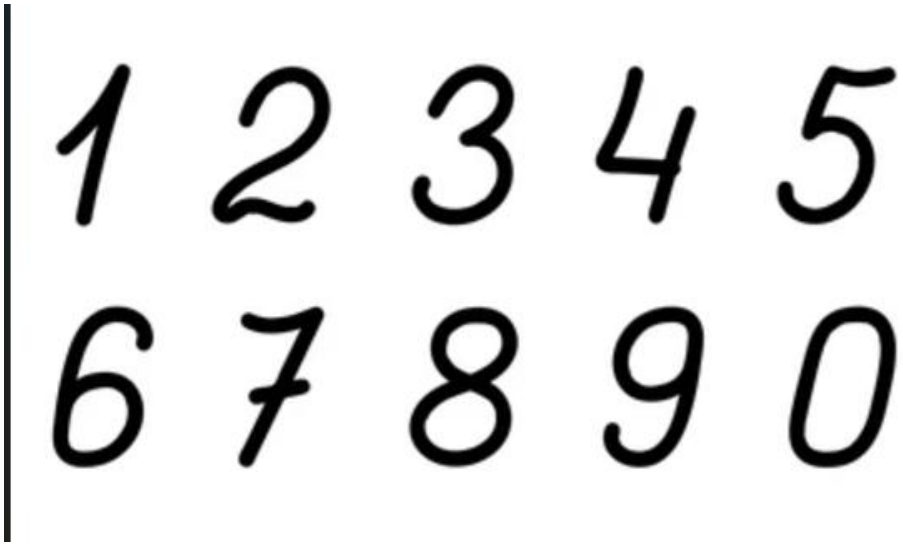
Hình 3.2 Ma trận nhầm lẫn

Nhận xét:

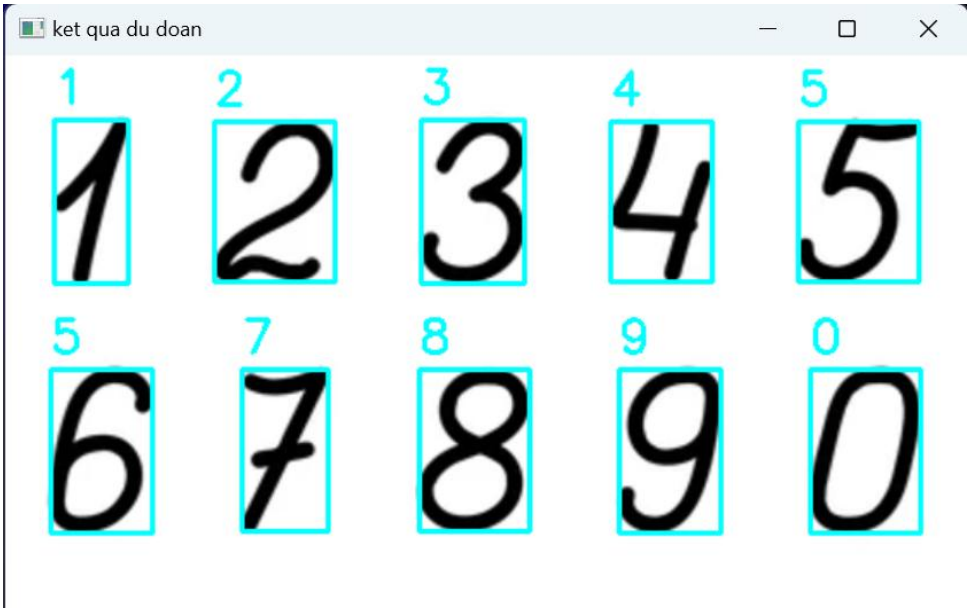
- Các giá trị nằm trên đường chéo chính (highlighted) là số lần dự đoán đúng.
- Số liệu ngoài đường chéo biểu thị các lỗi nhầm lẫn giữa các chữ số.

3.2. Kết quả dự đoán trên ảnh thực tế

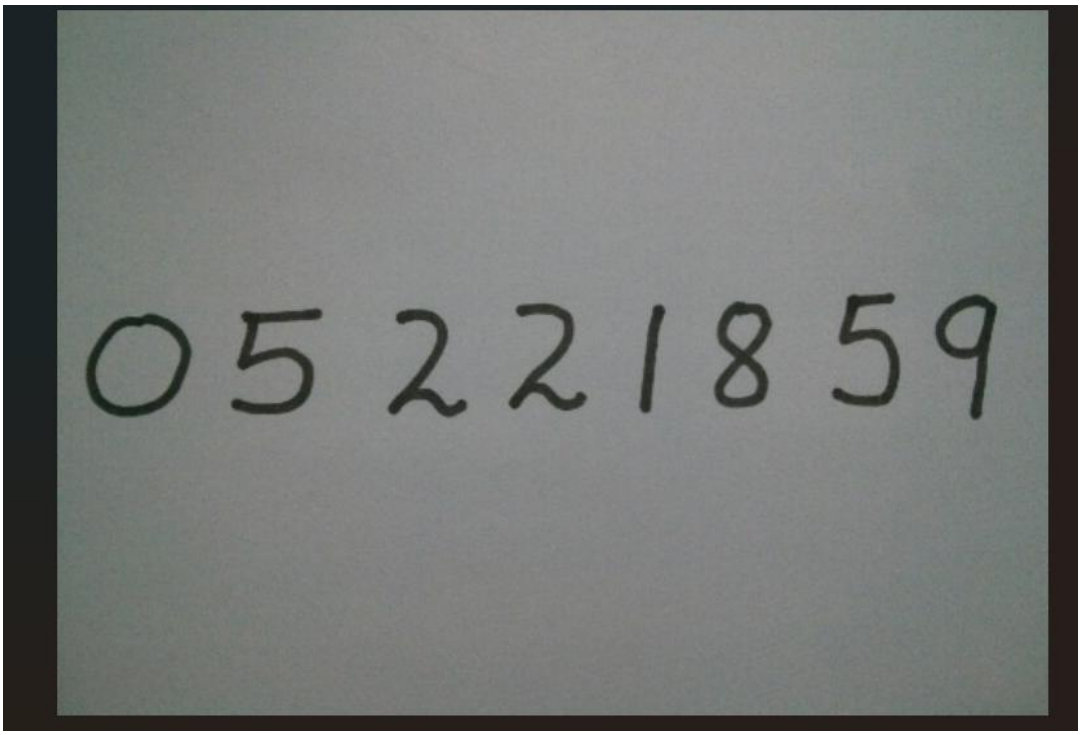
Dự đoán trên ảnh người dùng tải lên: Chương trình cho phép tải ảnh chứa các chữ số viết tay, dự đoán từng chữ số và hiển thị kết quả trực tiếp trên ảnh.



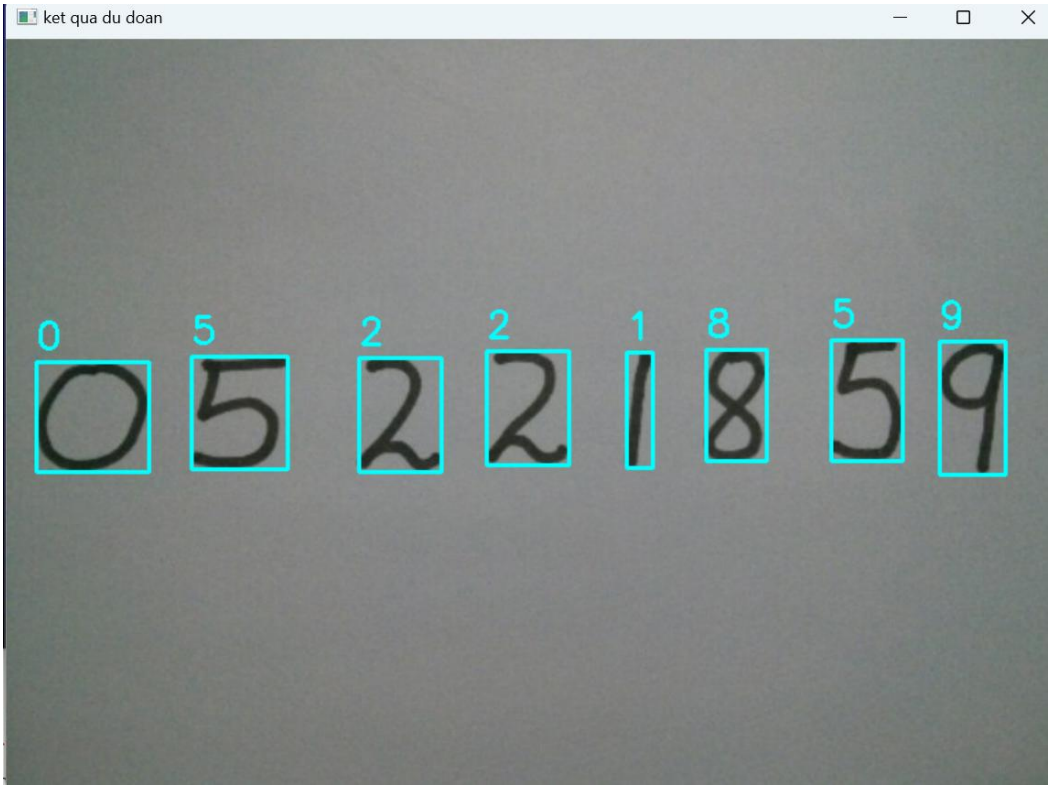
Hình 3.3 Mẫu thử thứ nhất



Hình 3.4 Kết quả dự đoán mẫu thử thứ nhất



Hình 3.5 Mẫu thử thứ hai



Hình 3.6 Kết quả dự đoán mẫu thử thứ hai

3.3. Đánh giá và thảo luận

Ưu điểm:

Hiệu quả về thời gian: Việc sử dụng HOG và LinearSVC giúp quá trình trích xuất đặc trưng và huấn luyện mô hình diễn ra nhanh chóng, phù hợp với các ứng dụng thực tế đòi hỏi thời gian xử lý nhanh.

Độ chính xác khá tốt: Với độ chính xác đạt gần 90% trên tập dữ liệu MNIST, mô hình đã chứng minh được khả năng nhận dạng chữ số viết tay một cách đáng tin cậy.

Giao diện thân thiện: Việc xây dựng giao diện người dùng bằng Tkinter giúp cho người dùng dễ dàng tương tác với hệ thống và nhận được kết quả dự đoán một cách trực quan.

Nhược điểm:

Hiệu suất hạn chế: So với các mô hình học sâu (Deep Learning) như CNN, mô hình dựa trên HOG và LinearSVC có thể cho kết quả kém hơn khi xử lý các dữ liệu phức tạp hơn, đặc biệt là các dữ liệu có nhiều biến thể và nhiễu.

Khó khăn với dữ liệu phức tạp: HOG là một phương pháp trích xuất đặc trưng hiệu quả nhưng có thể gặp khó khăn khi đối mặt với các dữ liệu chữ số viết tay có nhiều biến thể, kiểu chữ khác nhau hoặc bị nhiễu nặng.

Nhận xét chung:

Mô hình mà chúng ta đã xây dựng là một điểm khởi đầu tốt để hiểu về các kỹ thuật xử lý ảnh và học máy. Mô hình này có thể hoạt động tốt trong các điều kiện lý tưởng, khi dữ liệu đầu vào tương đối sạch và đơn giản.

Tuy nhiên, để đạt được hiệu suất cao hơn và khả năng ứng dụng rộng rãi hơn, chúng ta cần phải xem xét một số cải tiến sau:

Sử dụng mô hình Deep Learning: Các mô hình như CNN có khả năng tự động học các đặc trưng phức tạp từ dữ liệu, giúp cải thiện đáng kể độ chính xác trên các dữ liệu phức tạp.

Tiền xử lý dữ liệu kỹ lưỡng hơn: Áp dụng các kỹ thuật tiền xử lý như tăng cường dữ liệu (data augmentation), loại bỏ nhiễu, chuẩn hóa dữ liệu để cải thiện chất lượng dữ liệu đầu vào.

Kết hợp nhiều phương pháp: Kết hợp HOG với các phương pháp trích xuất đặc trưng khác như SIFT, SURF để tạo ra một bộ đặc trưng đa dạng hơn. Đánh giá mô hình toàn diện: Sử dụng các chỉ số đánh giá khác ngoài độ chính xác như độ nhạy, độ đặc hiệu, F1-score để đánh giá một cách toàn diện hơn về hiệu suất của mô hình.

Tổng kết:

Mô hình nhận dạng chữ số viết tay dựa trên HOG và LinearSVC là một ví dụ điển hình về cách áp dụng các kỹ thuật xử lý ảnh và học máy để giải quyết một bài toán thực tế. Mặc dù có những hạn chế nhất định, mô hình này vẫn cung cấp một nền tảng vững chắc để chúng ta xây dựng các hệ thống nhận dạng phức tạp hơn trong tương lai.

Chương 4: Kết luận

4.1. Kết luận

Bài toán nhận dạng chữ số viết tay đã được giải quyết bằng cách kết hợp các phương pháp xử lý ảnh, trích xuất đặc trưng HOG và mô hình phân loại LinearSVC. Qua quá trình thực hiện, chương trình đã đạt được một số kết quả đáng chú ý:

Hiệu quả của mô hình: Độ chính xác trên tập dữ liệu MNIST là ~89.88%, cho thấy mô hình hoạt động tốt trong môi trường chuẩn hóa.

Ma trận nhầm lẫn thể hiện rõ ràng các lỗi nhầm lẫn giữa các chữ số, chủ yếu xảy ra ở các chữ số có hình dạng tương tự như 3-5, 7-9.

Giao diện thân thiện: Giao diện người dùng đơn giản, cho phép tải ảnh trực tiếp để dự đoán hoặc đánh giá mô hình một cách dễ dàng.

Khả năng ứng dụng: Hệ thống có thể được sử dụng trong các ứng dụng cơ bản như nhận dạng chữ số trên tài liệu, xử lý ảnh tự động trong hệ thống kiểm tra số liệu.

4.2. Hạn chế

Hiệu suất: Độ chính xác vẫn còn thấp so với các mô hình hiện đại như CNN (Convolutional Neural Networks), vốn có khả năng học đặc trưng tốt hơn từ dữ liệt. Không tối ưu khi làm việc với dữ liệu viết tay phức tạp, ngoài tập chuẩn hóa MNIST.

Khả năng mở rộng: Mô hình chưa hỗ trợ nhận dạng ký tự (chữ cái) hoặc các ký tự viết tay khác.

Tốc độ xử lý: LinearSVC cần thời gian huấn luyện đáng kể khi làm việc với lượng lớn dữ liệu hoặc đặc trưng phức tạp.

4.3. Hướng phát triển

Để nâng cao hiệu quả và khả năng ứng dụng của chương trình, một số hướng phát triển có thể được đề xuất:

Sử dụng các mô hình Deep Learning:

Chuyển sang các mô hình CNN như LeNet, AlexNet hoặc các mô hình tiên tiến hơn như ResNet để cải thiện độ chính xác.

Sử dụng framework như TensorFlow hoặc PyTorch để xây dựng và huấn luyện mô hình.

Mở rộng dữ liệu:Kết hợp thêm các bộ dữ liệu viết tay khác như EMNIST (Extended MNIST) để huấn luyện mô hình trên các dạng chữ cái và ký tự viết tay.

Tối ưu giao diện người dùng: Tích hợp thêm tính năng kéo-thả ảnh vào giao diện. Hỗ trợ dự đoán trực tiếp từ camera hoặc video.

Ứng dụng thực tế: Triển khai chương trình trên thiết bị di động hoặc hệ thống nhúng (embedded systems) để nhận dạng chữ số trong thời gian thực.

Khử nhiễu và tiền xử lý nâng cao: Cải thiện thuật toán tiền xử lý để hoạt động tốt hơn với ảnh có độ phân giải thấp hoặc chứa nhiều nhiễu.

Kết luận tổng thể: Bài toán nhận dạng chữ số viết tay đã được giải quyết ở mức cơ bản với các phương pháp truyền thống. Mặc dù hiệu quả đã đạt mức khá tốt trên dữ liệu MNIST, vẫn còn nhiều tiềm năng để cải thiện và mở rộng hệ thống trong tương lai.

Tài liệu tham khảo

[1] Trang web: https://docs.ultralytics.com/vi/datasets/classify/mnist/#usage

[tham khảo ngày 20 tháng 11 năm 2024]

[2] Trang web : https://stackoverflow.com/questions/45384185/what-is-the-difference-between-linearsvc-and-svckernel-linear

[tham khảo ngày 21 tháng 11 năm 2024]