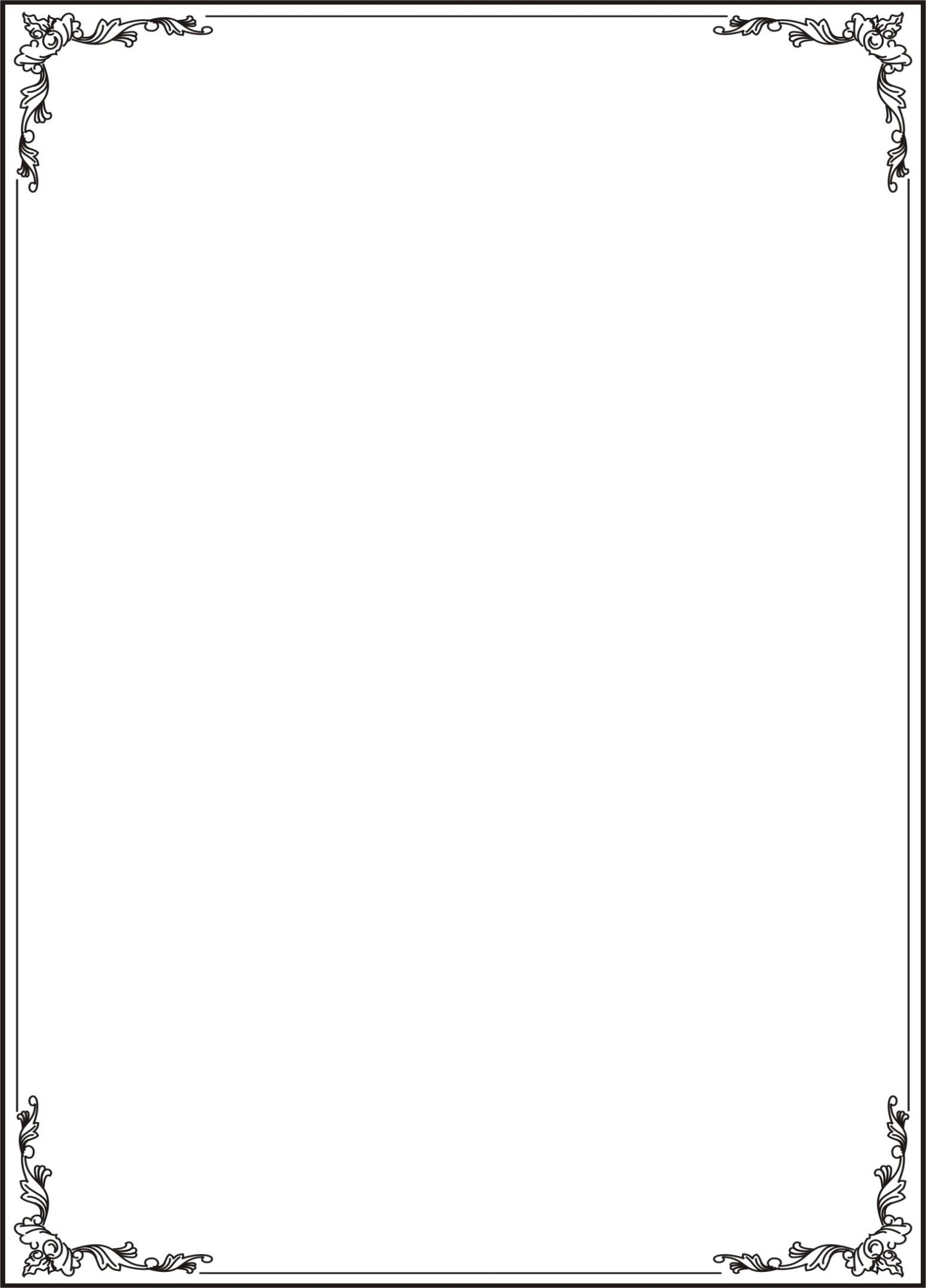
**TRƯỜNG ĐẠI HỌC LAO ĐỘNG – XÃ HỘI**



**KHÓA CÔNG NGHỆ THÔNG TIN**

**--------\*\*\*--------**

**BÁO CÁO BÀI TẬP LỚN**

**KHAI PHÁ DỮ LIỆU**

**ĐỀ TÀI** ỨNG DỤNG MẠNG NƠ-RON CNN TRONG NHẬN DẠNG BIỂN BÁO GIAO THÔNG BẰNG HÌNH ẢNH VÀ VIDEO

|  |  |
| --- | --- |
| Giảng viên hướng dẫn | NULL |
| Sinh viên thực hiện | |
| Nguyễn Xuân Hiệp  NULL | NULL  NULL |
| Lớp tín chỉ | D18CN02 |

Hà Nội, năm 2025

**MỤC LỤC**

[LỜI MỞ ĐẦU 3](#_Toc199354922)

[TỔNG QUAN ĐỀ TÀI 4](#_Toc199354923)

[1. Lý do lựa chọn đề tài. 4](#_Toc199354924)

[2. Phân tích bài toán. 5](#_Toc199354925)

[2.1. Sự đa dạng của biển báo giao thông. 5](#_Toc199354926)

[2.2. Điều kiện môi trường. 5](#_Toc199354927)

[2.3. Góc nhìn và khoảng cách. 5](#_Toc199354928)

[2.4. Tốc độ xử lý. 5](#_Toc199354929)

[2.5. Chất lượng dữ liệu. 5](#_Toc199354930)

[NHỮNG CÔNG NGHỆ ĐƯỢC SỬ DỤNG 7](#_Toc199354931)

[1. Python. 7](#_Toc199354932)

[2. Thư viện học sâu và xử lý ảnh. 7](#_Toc199354933)

[2.1. TensorFlow/Keras. 7](#_Toc199354934)

[2.2. OpenCV. 8](#_Toc199354935)

[2.3. PIL (Pillow). 8](#_Toc199354936)

[3. Thư viện xử lý dữ liệu. 8](#_Toc199354937)

[3.1. NumPy. 8](#_Toc199354938)

[3.2. Pandas. 9](#_Toc199354939)

[4. Thư viện trực quan hóa. 9](#_Toc199354940)

[5. Thư viện học máy. 10](#_Toc199354941)

[6. Tăng cường dữ liệu (Data Augmentation). 10](#_Toc199354942)

[7. Giao diện người dùng (GUI). 11](#_Toc199354943)

[8. Công nghệ mạng nơ-ron tích chập (CNN). 12](#_Toc199354944)

[9. Các kỹ thuật tối ưu hóa. 13](#_Toc199354945)

[9.1. Batch Normalization. 13](#_Toc199354946)

[9.2. Dropout. 13](#_Toc199354947)

[9.3. ReduceLROnPlateau. 13](#_Toc199354948)

[9.4. Early Stopping. 14](#_Toc199354949)

[10. Lưu và tải mô hình. 14](#_Toc199354950)

[PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG 16](#_Toc199354951)

[1. Biểu đồ Phân cấp Chức năng (Functional Decomposition Diagram - FDD). 16](#_Toc199354952)

[1.1. Tiền xử lý Dữ liệu. 16](#_Toc199354953)

[1.2. Huấn luyện Mô hình. 17](#_Toc199354954)

[1.3. Tăng cường Dữ liệu (Data Augmentation). 17](#_Toc199354955)

[1.4. Giao diện Người dùng (GUI). 18](#_Toc199354956)

[1.5. Lưu và Tải Mô hình. 19](#_Toc199354957)

[2. Biểu đồ Luồng Dữ liệu Mức Ngữ cảnh (Context-Level Data Flow Diagram - DFD). 20](#_Toc199354958)

[2.1. Người dùng → Hệ thống. 20](#_Toc199354959)

[2.2. Hệ thống → Mô hình CNN. 20](#_Toc199354960)

[2.3. Mô hình CNN → Hệ thống. 20](#_Toc199354961)

[2.4. Hệ thống → Người dùng. 20](#_Toc199354962)

[CÀI ĐẶT VÀ CHẠY PHẦN MỀM 22](#_Toc199354963)

[1. Cài đặt. 22](#_Toc199354964)

[1.1. Cài đặt Python. 22](#_Toc199354965)

[1.2. Cài Đặt Các Thư Viện Python Cần Thiết. 22](#_Toc199354966)

[1.3. Cài Đặt IDE (Môi Trường Phát Triển). 23](#_Toc199354967)

[1.4. Cài Đặt CUDA và cuDNN (Cho GPU - Tuỳ Chọn). 23](#_Toc199354968)

[2. Chạy phần mềm. 24](#_Toc199354969)

[2.1. Chạy file traffic\_signs.ipynb 24](#_Toc199354970)

[2.2. Chạy file gui.py 27](#_Toc199354971)

[KẾT LUẬN 29](#_Toc199354972)

[TÀI LIỆU THAM KHẢO 30](#_Toc199354973)

# LỜI MỞ ĐẦU

Trong bối cảnh xã hội hiện đại, giao thông đóng vai trò là một trong những yếu tố then chốt quyết định sự phát triển kinh tế và chất lượng cuộc sống của con người. Tuy nhiên, cùng với sự gia tăng mật độ phương tiện tham gia giao thông, các vấn đề liên quan đến an toàn giao thông ngày càng trở nên phức tạp và đáng lo ngại. Một trong những giải pháp quan trọng để giảm thiểu tai nạn giao thông và nâng cao hiệu quả quản lý giao thông là việc ứng dụng các công nghệ tiên tiến, đặc biệt là trí tuệ nhân tạo (AI) và học máy (Machine Learning), vào việc nhận dạng và phân loại các biển báo giao thông.

Biển báo giao thông là một phần không thể thiếu trong hệ thống giao thông, đóng vai trò hướng dẫn, cảnh báo và điều tiết lưu lượng phương tiện. Việc nhận dạng chính xác các biển báo này không chỉ giúp tài xế tuân thủ đúng luật giao thông mà còn góp phần quan trọng trong việc phát triển các hệ thống giao thông thông minh, chẳng hạn như xe tự hành. Tuy nhiên, nhận dạng biển báo giao thông là một bài toán đầy thách thức do sự đa dạng về hình dạng, màu sắc, kích thước, cũng như các yếu tố môi trường như ánh sáng, thời tiết, và góc nhìn.

Trong những năm gần đây, sự phát triển vượt bậc của các mạng nơ-ron tích chập (Convolutional Neural Networks - CNN) đã mở ra một kỷ nguyên mới trong lĩnh vực nhận dạng hình ảnh. CNN, với khả năng tự động học và trích xuất các đặc trưng từ dữ liệu hình ảnh, đã chứng minh được hiệu quả vượt trội so với các phương pháp truyền thống. Đặc biệt, trong lĩnh vực nhận dạng biển báo giao thông, CNN không chỉ mang lại độ chính xác cao mà còn có khả năng xử lý nhanh chóng, phù hợp với các ứng dụng thời gian thực.

Trong bài tiểu luận này, chúng tôi sẽ đi sâu vào quá trình xây dựng và huấn luyện một mô hình CNN để nhận dạng biển báo giao thông. Từ việc thu thập và tiền xử lý dữ liệu, đến việc thiết kế kiến trúc mô hình và tối ưu hóa hiệu suất, chúng tôi sẽ trình bày chi tiết từng bước thực hiện. Bên cạnh đó, chúng tôi cũng sẽ thảo luận về các kỹ thuật tăng cường dữ liệu (data augmentation) để cải thiện khả năng tổng quát hóa của mô hình, cũng như sử dụng các phương pháp đánh giá để đo lường độ chính xác và hiệu quả của mô hình trên cả tập dữ liệu huấn luyện và kiểm tra.

# TỔNG QUAN ĐỀ TÀI

## 1. Lý do lựa chọn đề tài.

Trong bối cảnh đô thị hóa nhanh chóng và sự gia tăng mật độ phương tiện giao thông, việc đảm bảo an toàn giao thông đã trở thành một vấn đề cấp bách trên toàn cầu. Theo thống kê của Tổ chức Y tế Thế giới (WHO), tai nạn giao thông là một trong những nguyên nhân hàng đầu gây tử vong và thương tích nghiêm trọng, đặc biệt là ở các quốc gia đang phát triển. Một trong những yếu tố quan trọng góp phần vào tai nạn giao thông là việc không tuân thủ hoặc không nhận biết kịp thời các biển báo giao thông. Do đó, việc phát triển các hệ thống hỗ trợ nhận dạng biển báo giao thông tự động không chỉ giúp nâng cao ý thức của người tham gia giao thông mà còn là nền tảng quan trọng cho các hệ thống giao thông thông minh và xe tự hành.

Sự phát triển của trí tuệ nhân tạo (AI) và học máy (Machine Learning), đặc biệt là các mạng nơ-ron tích chập (CNN), đã mở ra những cơ hội mới trong việc giải quyết bài toán nhận dạng biển báo giao thông. CNN có khả năng tự động học và trích xuất các đặc trưng từ dữ liệu hình ảnh, giúp cải thiện độ chính xác và tốc độ xử lý so với các phương pháp truyền thống. Điều này khiến CNN trở thành công cụ lý tưởng để xử lý các bài toán phức tạp như nhận dạng biển báo giao thông, nơi mà sự đa dạng về hình dạng, màu sắc, và điều kiện môi trường đặt ra nhiều thách thức.

Với những lý do trên, việc lựa chọn đề tài "Ứng dụng mạng nơ-ron tích chập (CNN) trong nhận dạng biển báo giao thông bằng hình ảnh và video" không chỉ mang tính thực tiễn cao mà còn có ý nghĩa khoa học, góp phần vào sự phát triển của các hệ thống giao thông thông minh và an toàn hơn trong tương lai.

## 2. Phân tích bài toán.

Bài toán nhận dạng biển báo giao thông là một bài toán phức tạp thuộc lĩnh vực thị giác máy tính (Computer Vision). Mục tiêu chính của bài toán là xây dựng một hệ thống có khả năng tự động nhận dạng và phân loại các biển báo giao thông từ dữ liệu hình ảnh hoặc video. Để đạt được mục tiêu này, bài toán cần giải quyết các thách thức sau:

### 2.1. Sự đa dạng của biển báo giao thông.

Biển báo giao thông có nhiều loại khác nhau, bao gồm biển chỉ dẫn, biển cảnh báo, biển cấm, và biển hiệu lệnh. Mỗi loại biển báo có hình dạng, màu sắc, và ký hiệu riêng biệt, đòi hỏi hệ thống phải có khả năng phân biệt chính xác.

### 2.2. Điều kiện môi trường.

Hình ảnh biển báo có thể bị ảnh hưởng bởi các yếu tố môi trường như ánh sáng yếu, thời tiết xấu (mưa, sương mù), hoặc bị che khuất một phần. Điều này làm tăng độ khó của bài toán, đòi hỏi hệ thống phải có khả năng xử lý các tình huống phức tạp.

### 2.3. Góc nhìn và khoảng cách.

Biển báo có thể được chụp từ nhiều góc độ khác nhau và ở các khoảng cách khác nhau, dẫn đến sự biến dạng về hình ảnh. Hệ thống cần có khả năng nhận dạng chính xác ngay cả khi biển báo không nằm ở vị trí lý tưởng.

### 2.4. Tốc độ xử lý.

Trong các ứng dụng thời gian thực, chẳng hạn như hệ thống hỗ trợ lái xe (ADAS) hoặc xe tự hành, tốc độ xử lý là yếu tố quan trọng. Hệ thống cần đưa ra kết quả nhận dạng nhanh chóng để đảm bảo an toàn và hiệu quả.

### 2.5. Chất lượng dữ liệu.

Dữ liệu huấn luyện đóng vai trò quan trọng trong việc xây dựng mô hình. Tuy nhiên, việc thu thập và gán nhãn dữ liệu biển báo giao thông có thể tốn kém và mất nhiều thời gian. Ngoài ra, dữ liệu cần phải đa dạng và đại diện cho các tình huống thực tế để đảm bảo mô hình có khả năng tổng quát hóa tốt.

Để giải quyết các thách thức trên, bài toán cần được tiếp cận bằng các phương pháp tiên tiến trong lĩnh vực thị giác máy tính và học sâu (Deep Learning). Cụ thể, mạng nơ-ron tích chập (CNN) được lựa chọn do khả năng tự động học các đặc trưng từ dữ liệu hình ảnh và đạt được độ chính xác cao trong các bài toán phân loại hình ảnh. Bên cạnh đó, các kỹ thuật tăng cường dữ liệu (data augmentation) và tối ưu hóa mô hình cũng được áp dụng để cải thiện hiệu suất và khả năng tổng quát hóa của hệ thống.

# NHỮNG CÔNG NGHỆ ĐƯỢC SỬ DỤNG

## 1. Python.

Mục đích: Python được lựa chọn làm ngôn ngữ lập trình chính trong dự án vì tính linh hoạt, cú pháp đơn giản, dễ sử dụng và có một hệ sinh thái phong phú với nhiều thư viện hỗ trợ mạnh mẽ. Ngoài ra, cộng đồng người dùng Python rất lớn, giúp việc tìm kiếm tài liệu, giải quyết vấn đề và cập nhật công nghệ trở nên thuận lợi hơn.

Ứng dụng: Python được sử dụng trong nhiều khía cạnh quan trọng của dự án. Cụ thể, nó được dùng để viết mã xử lý và tiền xử lý dữ liệu, bao gồm làm sạch, chuẩn hóa và trích xuất đặc trưng từ dữ liệu. Ngoài ra, Python hỗ trợ xây dựng mô hình học máy, từ việc thiết kế kiến trúc mô hình đến huấn luyện và tối ưu hóa để đạt hiệu suất cao. Quá trình đánh giá mô hình cũng được thực hiện bằng Python, sử dụng các kỹ thuật đo lường hiệu quả, trực quan hóa kết quả và so sánh giữa các phương pháp khác nhau. Bên cạnh đó, Python còn được ứng dụng trong việc phát triển giao diện người dùng, giúp tạo ra các công cụ trực quan và dễ sử dụng, hỗ trợ người dùng tương tác với hệ thống một cách thuận tiện.

## 2. Thư viện học sâu và xử lý ảnh.

### 2.1. TensorFlow/Keras.

Trong lĩnh vực học sâu, TensorFlow và Keras là hai thư viện quan trọng, cung cấp nền tảng mạnh mẽ để xây dựng và triển khai các mô hình trí tuệ nhân tạo. Với TensorFlow, người dùng có thể thực hiện các tính toán ma trận phức tạp và tối ưu hóa quá trình học của mô hình trên cả CPU lẫn GPU. Keras, với giao diện thân thiện và trực quan, giúp đơn giản hóa việc thiết kế mô hình mà không làm mất đi sự linh hoạt và khả năng tùy chỉnh.

Trong dự án này, TensorFlow và Keras được sử dụng để phát triển mô hình mạng nơ-ron tích chập (CNN), một kiến trúc phổ biến trong các bài toán nhận diện và phân loại hình ảnh. Quá trình xây dựng mô hình bắt đầu từ việc thiết kế các lớp Convolutional, Pooling và Fully Connected để trích xuất đặc trưng từ dữ liệu hình ảnh. Sau đó, mô hình được biên dịch với các hàm mất mát và thuật toán tối ưu hóa phù hợp, trước khi được huấn luyện trên tập dữ liệu lớn. Trong suốt quá trình này, các chỉ số đánh giá như độ chính xác, độ nhạy và F1-score được theo dõi để đảm bảo hiệu suất tối ưu. Sau khi hoàn tất, mô hình sẽ được lưu trữ để tái sử dụng hoặc triển khai vào hệ thống thực tế.

### 2.2. OpenCV.

Để xử lý hình ảnh trước khi đưa vào mô hình học sâu, dự án sử dụng OpenCV, một thư viện mạnh mẽ chuyên về xử lý ảnh và video. Với khả năng làm việc hiệu quả trên dữ liệu hình ảnh lớn, OpenCV giúp thực hiện các thao tác như thay đổi kích thước, chuẩn hóa và chuyển đổi định dạng ảnh, đảm bảo đầu vào nhất quán cho mô hình.

Trong thực tế, hình ảnh thu thập từ các nguồn khác nhau có thể có chất lượng không đồng đều, kích thước khác nhau hoặc chứa nhiều nhiễu. Vì vậy, OpenCV được sử dụng để làm sạch dữ liệu, cân bằng độ sáng và áp dụng các kỹ thuật nâng cao chất lượng ảnh. Bên cạnh đó, thư viện này còn hỗ trợ phát hiện và trích xuất đặc trưng từ ảnh, giúp tối ưu hóa quá trình nhận diện và phân loại.

### 2.3. PIL (Pillow).

Bên cạnh OpenCV, dự án cũng sử dụng PIL (Pillow) – một thư viện nhẹ nhưng hiệu quả cho các tác vụ xử lý ảnh cơ bản. PIL cho phép đọc và ghi các định dạng ảnh phổ biến như JPEG, PNG, BMP, đồng thời cung cấp công cụ để thay đổi kích thước và hiển thị hình ảnh trực tiếp trên giao diện người dùng.

Trong quá trình tiền xử lý dữ liệu, Pillow hỗ trợ chuyển đổi hình ảnh sang định dạng phù hợp, cắt xén hoặc thay đổi độ phân giải để tối ưu hóa đầu vào mô hình. Hơn nữa, thư viện này còn giúp hiển thị trực quan dữ liệu hình ảnh, tạo điều kiện thuận lợi cho việc kiểm tra kết quả xử lý trước khi đưa vào huấn luyện.

Nhìn chung, sự kết hợp giữa TensorFlow/Keras, OpenCV và PIL giúp dự án đạt được hiệu quả cao trong việc xây dựng, huấn luyện và triển khai mô hình học sâu, đồng thời đảm bảo dữ liệu đầu vào được xử lý một cách tối ưu nhất.

## 3. Thư viện xử lý dữ liệu.

### 3.1. NumPy.

Trong lập trình khoa học dữ liệu và trí tuệ nhân tạo, NumPy là một thư viện không thể thiếu nhờ khả năng xử lý dữ liệu số học hiệu quả. Với NumPy, người dùng có thể làm việc với các mảng nhiều chiều (ndarray) và thực hiện các phép toán ma trận nhanh chóng, tối ưu hóa hiệu suất so với danh sách thông thường trong Python.

Trong dự án này, NumPy đóng vai trò quan trọng trong xử lý dữ liệu hình ảnh. Khi hình ảnh được đưa vào hệ thống, chúng thường được chuyển đổi thành mảng số học để mô hình có thể tiếp nhận và xử lý. Các phép toán như chuẩn hóa dữ liệu, thay đổi hình dạng mảng (reshape), hoặc thực hiện các phép tính tuyến tính đều được hỗ trợ mạnh mẽ bởi NumPy. Ngoài ra, thư viện này còn giúp xử lý dữ liệu nhãn, hỗ trợ các thao tác như chuyển đổi nhãn sang dạng one-hot encoding để phù hợp với các thuật toán học máy.

### 3.2. Pandas.

Pandas là thư viện chuyên dụng để xử lý dữ liệu dạng bảng, cung cấp các công cụ mạnh mẽ để quản lý, phân tích và trực quan hóa dữ liệu. Được xây dựng trên NumPy, Pandas giúp thao tác với dữ liệu dễ dàng hơn nhờ các cấu trúc như DataFrame và Series, giúp xử lý dữ liệu giống như các bảng trong cơ sở dữ liệu hoặc bảng tính Excel.

Trong dự án này, nếu có dữ liệu dạng bảng (chẳng hạn như danh sách hình ảnh, nhãn tương ứng, hoặc các thuộc tính mô tả dữ liệu), Pandas sẽ hỗ trợ việc đọc, làm sạch và sắp xếp dữ liệu. Các thao tác phổ biến bao gồm lọc dữ liệu, nhóm dữ liệu theo các tiêu chí cụ thể, hoặc tính toán các thống kê mô tả. Pandas giúp đảm bảo rằng dữ liệu đầu vào được tổ chức một cách hợp lý trước khi đưa vào các mô hình học máy.

Sự kết hợp giữa NumPy và Pandas tạo ra một nền tảng vững chắc để xử lý dữ liệu hiệu quả, giúp tối ưu hóa quá trình huấn luyện và phân tích mô hình trong dự án.

## 4. Thư viện trực quan hóa.

Trong lĩnh vực khoa học dữ liệu và học máy, việc trực quan hóa dữ liệu đóng vai trò quan trọng giúp phân tích và đánh giá hiệu suất của mô hình. Matplotlib là một thư viện mạnh mẽ, hỗ trợ vẽ biểu đồ và trực quan hóa dữ liệu dưới nhiều hình thức khác nhau. Với khả năng tùy chỉnh cao, Matplotlib giúp người dùng dễ dàng tạo ra các biểu đồ trực quan, hỗ trợ quá trình phân tích dữ liệu một cách trực quan và dễ hiểu hơn.

Trong dự án này, Matplotlib được sử dụng để vẽ biểu đồ theo dõi quá trình huấn luyện mô hình. Cụ thể, nó giúp hiển thị:

Biểu đồ độ chính xác (accuracy): Giúp theo dõi sự cải thiện của mô hình qua từng epoch, đánh giá khả năng phân loại chính xác dữ liệu.

Biểu đồ hàm mất mát (loss function): Cho thấy mức độ lỗi của mô hình trong quá trình huấn luyện, giúp xác định xem mô hình có bị overfitting hay underfitting không.

Ngoài ra, Matplotlib còn có thể được sử dụng để trực quan hóa hình ảnh đầu vào, hiển thị kết quả dự đoán của mô hình, hoặc so sánh các phương pháp khác nhau. Việc sử dụng Matplotlib giúp quá trình đánh giá mô hình trở nên trực quan hơn, hỗ trợ người dùng đưa ra các điều chỉnh phù hợp nhằm cải thiện hiệu suất.

## 5. Thư viện học máy.

Scikit-learn là một trong những thư viện học máy phổ biến nhất trong Python, cung cấp nhiều công cụ hỗ trợ cho việc xử lý dữ liệu, huấn luyện mô hình và đánh giá hiệu suất. Với giao diện đơn giản và dễ sử dụng, Scikit-learn giúp triển khai các thuật toán học máy một cách nhanh chóng, đồng thời hỗ trợ nhiều phương pháp phân tích dữ liệu và đánh giá mô hình.

Trong dự án này, Scikit-learn được sử dụng chủ yếu trong hai khía cạnh quan trọng:

Phân chia dữ liệu thành tập huấn luyện và tập kiểm tra:

Trước khi huấn luyện mô hình, dữ liệu cần được chia thành hai phần:

Tập huấn luyện (training set): Dữ liệu mà mô hình sử dụng để học các đặc trưng.

Tập kiểm tra (test set): Dữ liệu được giữ lại để kiểm tra xem mô hình có hoạt động tốt hay không.

Scikit-learn cung cấp hàm train\_test\_split giúp chia dữ liệu một cách ngẫu nhiên, đảm bảo rằng tập dữ liệu kiểm tra không bị trùng lặp với tập huấn luyện, tránh tình trạng mô hình ghi nhớ dữ liệu thay vì học được quy luật chung.

Tính toán độ chính xác của mô hình:

Sau khi huấn luyện, hiệu suất của mô hình cần được đánh giá bằng cách so sánh dự đoán của mô hình với nhãn thực tế. Scikit-learn cung cấp hàm accuracy\_score, giúp tính toán tỷ lệ phần trăm dự đoán đúng, từ đó đánh giá mức độ hiệu quả của mô hình.

## 6. Tăng cường dữ liệu (Data Augmentation).

ImageDataGenerator (Keras) – Tăng cường dữ liệu để cải thiện mô hình học sâu

Trong lĩnh vực học sâu, đặc biệt là khi làm việc với dữ liệu hình ảnh, một trong những thách thức lớn nhất là thiếu dữ liệu đa dạng. Mô hình có thể học quá mức từ tập dữ liệu huấn luyện (overfitting) và không thể tổng quát hóa tốt trên dữ liệu thực tế. Để khắc phục vấn đề này, Keras cung cấp công cụ ImageDataGenerator, giúp tạo ra nhiều biến thể khác nhau của hình ảnh từ cùng một nguồn dữ liệu gốc.

Mục đích

ImageDataGenerator được sử dụng để tăng cường dữ liệu (data augmentation), giúp mô hình có thể học từ nhiều phiên bản khác nhau của hình ảnh mà không cần thu thập thêm dữ liệu mới. Điều này giúp mô hình hoạt động tốt hơn trên dữ liệu chưa từng thấy, cải thiện độ chính xác và khả năng tổng quát hóa.

Ứng dụng trong thực tế

Khi áp dụng vào quá trình huấn luyện, ImageDataGenerator có thể thực hiện nhiều phép biến đổi hình ảnh, giúp mô hình làm quen với các tình huống thực tế như:

Xoay ảnh (*rotation\_range*): Hình ảnh có thể xuất hiện ở các góc độ khác nhau, do đó cần huấn luyện mô hình để nhận diện ngay cả khi hình bị xoay nhẹ.

Dịch chuyển ảnh (*width\_shift\_range, height\_shift\_range*): Trong thực tế, một đối tượng trong ảnh có thể không nằm ở vị trí cố định, do đó mô hình cần học cách nhận diện ngay cả khi vật thể bị dịch chuyển.

Thu phóng ảnh (*zoom\_range*): Một vật thể có thể xuất hiện gần hoặc xa camera, nên cần huấn luyện mô hình để nhận diện chúng ở nhiều kích thước khác nhau.

Lật ảnh (*horizontal\_flip, vertical\_flip*): Một số hình ảnh có thể xuất hiện ở dạng phản chiếu hoặc bị đảo ngược, do đó cần lật ảnh để làm đa dạng tập dữ liệu huấn luyện.

Biến dạng ảnh (*shear\_range*): Giúp mô hình nhận diện ngay cả khi hình ảnh bị biến dạng nhẹ do góc nhìn thay đổi.

## 7. Giao diện người dùng (GUI).

Tkinter – Xây dựng giao diện đồ họa cho ứng dụng nhận dạng biển báo giao thông

Khi phát triển một ứng dụng học sâu, không chỉ cần có một mô hình mạnh mẽ mà còn phải đảm bảo rằng người dùng có thể dễ dàng tương tác với hệ thống. Đây là lý do Tkinter, thư viện GUI (giao diện người dùng đồ họa) mặc định của Python, trở thành một công cụ hữu ích trong việc xây dựng giao diện cho ứng dụng nhận dạng biển báo giao thông.

Tkinter giúp tạo ra một giao diện đơn giản, dễ sử dụng, cho phép người dùng dễ dàng tải lên hình ảnh và xem kết quả nhận dạng biển báo giao thông. Đây là bước quan trọng để đưa mô hình học sâu từ lý thuyết ra thực tế, giúp người dùng có thể tương tác trực tiếp với công nghệ mà không cần phải có kiến thức về lập trình.

Trong ứng dụng nhận dạng biển báo giao thông, Tkinter đóng vai trò như một cầu nối giữa người dùng và mô hình học sâu. Người dùng có thể chọn hình ảnh từ máy tính, sau đó giao diện sẽ hiển thị bức ảnh đó để họ xác nhận. Tiếp theo, họ chỉ cần nhấn nút "Nhận dạng" và mô hình sẽ tiến hành phân tích, xác định loại biển báo giao thông có trong ảnh. Kết quả nhận dạng sẽ được hiển thị ngay trên giao diện, giúp người dùng dễ dàng hiểu được biển báo là gì.

Điều quan trọng là Tkinter không chỉ đơn thuần giúp xây dựng giao diện, mà còn tạo nên một trải nghiệm người dùng mượt mà và thuận tiện. Người dùng có thể tải lại hình ảnh, lưu kết quả, hoặc thậm chí điều chỉnh mô hình trực tiếp từ giao diện mà không gặp phải bất kỳ khó khăn nào. Với những tính năng này, Tkinter giúp làm cho ứng dụng trở nên dễ dàng tiếp cận với tất cả mọi người, kể cả những người không có chuyên môn về công nghệ.

## 8. Công nghệ mạng nơ-ron tích chập (CNN).

CNN – Kiến trúc mạng nơ-ron chuyên dụng cho nhận dạng biển báo giao thông

Convolutional Neural Network (CNN) là một trong những kiến trúc mạng nơ-ron mạnh mẽ nhất hiện nay, đặc biệt được ứng dụng rộng rãi trong việc xử lý hình ảnh. Mục tiêu của việc sử dụng CNN trong các bài toán nhận dạng hình ảnh là giúp máy tính có thể tự động học các đặc trưng quan trọng từ dữ liệu, mà không cần sự can thiệp trực tiếp từ con người. Đây chính là lý do CNN được lựa chọn để giải quyết bài toán nhận dạng và phân loại biển báo giao thông từ hình ảnh.

CNN hoạt động dựa trên nguyên lý chuyển đổi các vùng nhỏ trong hình ảnh đầu vào thành các đặc trưng quan trọng, sau đó dùng những đặc trưng này để nhận diện và phân loại các đối tượng trong ảnh. Đặc biệt, trong bài toán nhận dạng biển báo giao thông, CNN có thể học và nhận diện những đặc điểm đặc trưng của biển báo, chẳng hạn như hình dạng, màu sắc hay các biểu tượng, từ đó phân loại chính xác các loại biển báo.

Quá trình xây dựng mô hình CNN bao gồm việc sử dụng các lớp như Conv2D, MaxPool2D, Dropout, Flatten, và Dense. Cụ thể:

Conv2D (Lớp tích chập) là lớp đầu tiên trong CNN, giúp trích xuất các đặc trưng cơ bản từ hình ảnh, chẳng hạn như đường viền và kết cấu.

MaxPool2D (Lớp chập tối đa) giúp giảm kích thước dữ liệu sau khi trích xuất đặc trưng, đồng thời giữ lại các thông tin quan trọng nhất, giúp giảm độ phức tạp và khả năng overfitting.

Dropout là một kỹ thuật để ngăn ngừa mô hình học quá mức từ dữ liệu huấn luyện, giúp tăng khả năng tổng quát hóa của mô hình.

Flatten chuyển đổi các đặc trưng sau khi qua các lớp Conv2D và MaxPool2D thành một dãy số để có thể đưa vào lớp Dense, lớp kết nối đầy đủ giúp mô hình đưa ra quyết định phân loại cuối cùng.

Khi mô hình đã được xây dựng, việc huấn luyện mô hình với tập dữ liệu biển báo giao thông là bước quan trọng tiếp theo. Mô hình sẽ học từ những hình ảnh biển báo và các nhãn tương ứng, tối ưu hóa các tham số để có thể phân loại đúng các biển báo trong dữ liệu mới. Sau khi huấn luyện, mô hình có thể nhận dạng và phân loại các biển báo giao thông mà nó chưa từng thấy trong quá trình huấn luyện, từ đó hỗ trợ việc nhận diện biển báo trong các tình huống giao thông thực tế.

## 9. Các kỹ thuật tối ưu hóa.

### 9.1. Batch Normalization.

Mục đích: Batch Normalization là một kỹ thuật giúp chuẩn hóa đầu vào của mỗi lớp trong mạng nơ-ron, từ đó giúp tăng tốc quá trình huấn luyện và cải thiện hiệu suất của mô hình.

Ứng dụng: Khi các đầu vào của mạng nơ-ron được chuẩn hóa, mô hình không phải chịu ảnh hưởng bởi các độ lệch của dữ liệu đầu vào. Kỹ thuật này giúp giảm thiểu vấn đề vanishing gradients, giúp quá trình huấn luyện ổn định và nhanh hơn. Đặc biệt, trong các mô hình CNN phức tạp, Batch Normalization giúp mô hình đạt được hiệu quả học cao hơn và dễ dàng hội tụ hơn.

### 9.2. Dropout.

Mục đích: Dropout là một kỹ thuật ngẫu nhiên loại bỏ một phần các đơn vị (neurons) trong quá trình huấn luyện, giúp giảm hiện tượng overfitting và tăng khả năng tổng quát hóa của mô hình.

Ứng dụng: Trong quá trình huấn luyện, Dropout sẽ ngẫu nhiên "tắt" một số lượng neurons, khiến mô hình không phụ thuộc quá nhiều vào bất kỳ đơn vị nào. Điều này giúp mô hình học được nhiều đặc trưng khác nhau từ dữ liệu, thay vì chỉ tập trung vào một số đặc trưng cụ thể mà có thể dẫn đến overfitting. Kết quả là mô hình có thể tổng quát tốt hơn khi gặp phải dữ liệu chưa thấy.

### 9.3. ReduceLROnPlateau.

Mục đích: ReduceLROnPlateau là một chiến lược giảm tốc độ học (learning rate) khi mô hình không cải thiện hiệu suất sau một số epoch nhất định.  
Ứng dụng: Khi mô hình đã học gần như hết mức, nhưng độ chính xác vẫn không cải thiện, giảm tốc độ học có thể giúp mô hình học chi tiết hơn trong những giai đoạn cuối. Kỹ thuật này giúp mô hình tránh rơi vào tình trạng "bão hòa" và tiếp tục cải thiện hiệu suất trong khi tránh việc tốc độ học quá lớn làm mất đi các cải thiện nhỏ. Điều này rất hữu ích trong việc huấn luyện các mô hình phức tạp như CNN cho nhận dạng biển báo giao thông.

### 9.4. Early Stopping.

Mục đích: Early Stopping giúp dừng quá trình huấn luyện sớm nếu mô hình không có sự cải thiện đáng kể trên tập kiểm tra.

Ứng dụng: Trong huấn luyện mô hình học sâu, nếu mô hình tiếp tục huấn luyện quá lâu mà không có sự cải thiện trên tập kiểm tra, có thể dẫn đến hiện tượng overfitting. Early Stopping theo dõi sự thay đổi của độ chính xác trên tập kiểm tra và dừng huấn luyện khi hiệu suất không cải thiện sau một số epoch nhất định. Điều này không chỉ giúp tiết kiệm thời gian huấn luyện mà còn giúp mô hình duy trì tính tổng quát cao, tránh việc học quá mức trên dữ liệu huấn luyện.

## 10. Lưu và tải mô hình.

HDF5 (.h5) – Định dạng tệp lưu trữ mô hình học sâu

Mục đích: HDF5 (.h5) là một định dạng tệp lưu trữ dữ liệu hiệu quả, được sử dụng rộng rãi trong các ứng dụng học sâu để lưu và tải mô hình đã huấn luyện. Định dạng này hỗ trợ lưu trữ các cấu trúc dữ liệu phức tạp và kích thước lớn, làm cho nó trở thành lựa chọn lý tưởng để lưu trữ các mô hình học sâu có nhiều lớp và tham số.

Ứng dụng trong huấn luyện mô hình

Lưu mô hình sau khi huấn luyện:

Khi huấn luyện một mô hình học sâu, quá trình huấn luyện có thể kéo dài và yêu cầu sử dụng nhiều tài nguyên tính toán. Sau khi mô hình đạt được hiệu suất tốt nhất trên dữ liệu huấn luyện và kiểm tra, việc lưu mô hình vào tệp HDF5 giúp bảo vệ công sức huấn luyện và giữ lại các tham số (weights) cùng cấu trúc của mô hình. Điều này giúp tránh phải huấn luyện lại mô hình từ đầu khi cần sử dụng lại hoặc triển khai trong các ứng dụng thực tế.

Tải lại mô hình để sử dụng trong giao diện người dùng:

Sau khi lưu mô hình vào định dạng HDF5, bạn có thể tải lại mô hình trong các ứng dụng hoặc giao diện người dùng mà không cần huấn luyện lại mô hình từ đầu. Điều này giúp tiết kiệm thời gian và tài nguyên, đồng thời tạo ra trải nghiệm người dùng mượt mà hơn. Ví dụ, trong một hệ thống nhận dạng biển báo giao thông, mô hình đã huấn luyện có thể được tải lại để nhận dạng biển báo từ các hình ảnh đầu vào trong thời gian thực, mang lại khả năng phản hồi nhanh chóng.

# PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

## 1. Biểu đồ Phân cấp Chức năng (Functional Decomposition Diagram - FDD).

Biểu đồ FDD phân chia hệ thống thành các chức năng con, giúp xác định các nhiệm vụ chính và phụ của hệ thống. Dưới đây là mô tả các chức năng chính của hệ thống nhận dạng biển báo giao thông:

Hệ thống Nhận dạng Biển báo Giao thông

### 1.1. Tiền xử lý Dữ liệu.

Tiền xử lý dữ liệu là một bước quan trọng trong hệ thống nhận dạng biển báo giao thông, nhằm chuẩn bị các hình ảnh đầu vào để mô hình học sâu (CNN) có thể xử lý và phân loại chính xác. Giai đoạn này bao gồm một số bước thiết yếu để làm sạch và chuẩn hóa dữ liệu.

Đầu tiên, hệ thống sẽ đọc hình ảnh từ thư mục mà người dùng đã tải lên hoặc từ thiết bị đầu cuối. Đây là bước cơ bản giúp hệ thống tiếp nhận dữ liệu đầu vào. Sau khi hình ảnh được tải về, chúng sẽ được thay đổi kích thước sao cho đồng nhất với yêu cầu của mô hình. Việc thay đổi kích thước này là cần thiết vì các mô hình học sâu, đặc biệt là mạng nơ-ron tích chập (CNN), yêu cầu các hình ảnh đầu vào có kích thước cố định để xử lý hiệu quả.

Tiếp theo, để giúp tăng tốc quá trình huấn luyện và cải thiện độ chính xác, hình ảnh sẽ được chuẩn hóa dữ liệu. Cụ thể, giá trị pixel của mỗi hình ảnh sẽ được chia cho 255, giúp đưa tất cả các giá trị pixel về phạm vi từ 0 đến 1. Điều này làm cho quá trình huấn luyện trở nên mượt mà hơn và giúp mô hình học nhanh hơn.

Cuối cùng, nhãn của mỗi hình ảnh, tức là loại biển báo giao thông trong hình, sẽ được chuyển thành dạng one-hot encoding. Trong dạng này, mỗi nhãn sẽ được biểu diễn dưới dạng một vector, trong đó mỗi phần tử tương ứng với một loại biển báo, với giá trị 1 nếu nhãn đó đúng và 0 nếu không phải. Việc chuyển nhãn thành one-hot encoding giúp mô hình dễ dàng phân loại các biển báo giao thông chính xác hơn.

Nhờ vào các bước tiền xử lý này, dữ liệu sẽ được chuẩn bị đầy đủ và phù hợp, giúp mô hình học sâu có thể hoạt động hiệu quả, nhận dạng và phân loại các biển báo giao thông một cách chính xác.

### 1.2. Huấn luyện Mô hình.

**Huấn luyện Mô hình** là một giai đoạn quan trọng trong việc xây dựng hệ thống nhận dạng biển báo giao thông. Chức năng này bao gồm nhiều bước khác nhau để xây dựng và tối ưu hóa mô hình học sâu (CNN) nhằm đảm bảo khả năng phân loại chính xác các biển báo giao thông từ hình ảnh. Các nhiệm vụ chính trong quá trình huấn luyện mô hình bao gồm việc xây dựng kiến trúc mô hình CNN, biên dịch mô hình, huấn luyện mô hình trên dữ liệu và cuối cùng là đánh giá mô hình.

**Xây dựng kiến trúc mô hình CNN** là bước đầu tiên và rất quan trọng. Mô hình CNN sẽ được xây dựng với các lớp khác nhau như **Conv2D**, **MaxPool2D** và **Dense**. Các lớp này có vai trò trong việc xử lý và trích xuất các đặc trưng từ hình ảnh biển báo giao thông. Các lớp Conv2D sẽ học được các đặc trưng không gian trong hình ảnh, trong khi lớp MaxPool2D giúp giảm độ phức tạp của dữ liệu đầu vào, và lớp Dense đóng vai trò trong việc phân loại biển báo giao thông sau khi các đặc trưng đã được trích xuất.

Sau khi kiến trúc mô hình được xây dựng, bước tiếp theo là **biên dịch mô hình**. Trong giai đoạn này, mô hình sẽ được biên dịch bằng cách chọn các tham số quan trọng như **optimizer**, **loss function** và các tham số khác. **Optimizer**, chẳng hạn như Adam, sẽ giúp điều chỉnh trọng số của các lớp trong quá trình huấn luyện để tối ưu hóa việc học. **Loss function** được chọn sẽ phản ánh mức độ sai lệch giữa dự đoán của mô hình và kết quả thực tế, giúp mô hình cải thiện độ chính xác trong suốt quá trình huấn luyện.

Tiếp theo, **huấn luyện mô hình trên tập dữ liệu** là bước quan trọng để mô hình học và cải thiện độ chính xác. Trong bước này, mô hình sẽ được huấn luyện trên một tập dữ liệu lớn, với mỗi hình ảnh biển báo giao thông được đưa vào mô hình. Mô hình sẽ học các đặc trưng và đặc điểm của từng loại biển báo giao thông, nhờ đó có thể phân biệt và phân loại biển báo một cách chính xác.

Cuối cùng, sau khi huấn luyện, mô hình sẽ được **đánh giá trên tập kiểm tra**. Tập kiểm tra chứa các dữ liệu mà mô hình chưa được huấn luyện trước đó, giúp đánh giá hiệu suất và độ chính xác của mô hình trên dữ liệu thực tế. Việc đánh giá này giúp đảm bảo rằng mô hình không chỉ học tốt từ dữ liệu huấn luyện mà còn có khả năng tổng quát hóa và hoạt động tốt trên các dữ liệu mới, không nhìn thấy trước đó.

### 1.3. Tăng cường Dữ liệu (Data Augmentation).

**Tăng cường Dữ liệu (Data Augmentation)** là một kỹ thuật quan trọng giúp cải thiện khả năng tổng quát hóa của mô hình học máy, đồng thời giảm thiểu hiện tượng overfitting (quá khớp) bằng cách tạo ra các biến thể mới từ dữ liệu huấn luyện ban đầu. Các phương pháp tăng cường dữ liệu trong hệ thống nhận dạng biển báo giao thông bao gồm nhiều kỹ thuật xử lý hình ảnh khác nhau, giúp làm phong phú thêm bộ dữ liệu và giúp mô hình học được nhiều đặc trưng hơn từ các hình ảnh.

Một trong những kỹ thuật tăng cường dữ liệu phổ biến là **xoay hình ảnh**. Hình ảnh sẽ được xoay một góc ngẫu nhiên trong phạm vi nhất định, tạo ra những biến thể khác nhau của hình ảnh ban đầu. Điều này giúp mô hình học cách nhận diện biển báo giao thông từ các góc nhìn khác nhau, tăng tính đa dạng của dữ liệu và làm cho mô hình trở nên mạnh mẽ hơn khi gặp phải các biến thể trong thực tế.

Kỹ thuật tiếp theo là **dịch chuyển hình ảnh**. Hình ảnh có thể được dịch chuyển lên, xuống, trái hoặc phải, nhằm tạo ra những phiên bản mới của hình ảnh mà mô hình chưa được huấn luyện. Việc dịch chuyển hình ảnh giúp mô hình học được cách nhận diện biển báo giao thông trong các tình huống khi biển báo không ở vị trí trung tâm của hình ảnh.

**Thu phóng hình ảnh** cũng là một kỹ thuật hữu ích trong tăng cường dữ liệu. Phương pháp này thay đổi tỷ lệ của các đối tượng trong ảnh, giúp mô hình học cách nhận diện biển báo giao thông dù kích thước của biển báo có thay đổi. Điều này đặc biệt quan trọng vì trong thực tế, biển báo có thể xuất hiện gần hoặc xa camera, làm thay đổi kích thước của chúng.

Cuối cùng, **cắt xén hình ảnh** là một phương pháp tăng cường dữ liệu hiệu quả khác. Bằng cách cắt xén một phần của hình ảnh gốc, hệ thống tạo ra các dữ liệu huấn luyện mới từ các phần khác nhau của hình ảnh, giúp mô hình học được cách nhận diện biển báo giao thông dù chỉ có một phần của biển báo xuất hiện trong ảnh.

### 1.4. Giao diện Người dùng (GUI).

**Giao diện Người dùng (GUI)** là phần giao tiếp giữa hệ thống và người dùng, cho phép người dùng dễ dàng tương tác với hệ thống nhận dạng biển báo giao thông. Giao diện này được thiết kế để đơn giản, dễ sử dụng và giúp người dùng thực hiện các thao tác từ tải lên hình ảnh cho đến nhận kết quả phân loại biển báo.

Một trong những chức năng quan trọng trong GUI là **tải lên hình ảnh từ người dùng**. Người dùng có thể dễ dàng tải lên hình ảnh biển báo giao thông từ thiết bị của mình thông qua giao diện, giúp hệ thống nhận dạng biển báo trong hình ảnh đó. Khi người dùng chọn hình ảnh, hệ thống sẽ tự động xử lý và chuẩn bị dữ liệu cho bước tiếp theo.

Sau khi hình ảnh được tải lên, hệ thống sẽ **hiển thị hình ảnh đã tải lên** trên giao diện người dùng. Việc này cho phép người dùng xem lại hình ảnh trước khi thực hiện phân loại, giúp người dùng kiểm tra lại hình ảnh và chắc chắn rằng đó là hình ảnh biển báo giao thông mà họ muốn nhận dạng.

Khi hình ảnh đã được hiển thị và xác nhận, **mô hình CNN sẽ tiến hành phân loại hình ảnh**. Dựa trên các đặc trưng mà mô hình đã học trong quá trình huấn luyện, hệ thống sẽ phân tích hình ảnh và xác định loại biển báo giao thông có trong đó, như biển báo tốc độ, biển báo cấm, hay biển báo chỉ dẫn.

Cuối cùng, **kết quả phân loại sẽ được hiển thị** trên giao diện người dùng. Hệ thống không chỉ trả về loại biển báo mà còn cung cấp độ chính xác của mô hình đối với dự đoán đó. Điều này giúp người dùng dễ dàng xác định và kiểm tra kết quả nhận dạng, đồng thời tăng cường độ tin cậy vào khả năng của hệ thống trong việc phân loại biển báo giao thông.

### 1.5. Lưu và Tải Mô hình.

Chức năng **Lưu và Tải Mô hình** trong hệ thống nhận dạng biển báo giao thông đóng vai trò quan trọng trong việc bảo quản mô hình đã huấn luyện và tạo điều kiện cho việc sử dụng lại mô hình mà không cần phải huấn luyện lại từ đầu. Chức năng này bao gồm hai bước chính: lưu mô hình và tải mô hình.

Sau khi quá trình huấn luyện mô hình kết thúc và mô hình đã được đánh giá với kết quả tốt, mô hình sẽ được **lưu trữ dưới dạng tệp**, chẳng hạn như định dạng HDF5, giúp giữ lại toàn bộ các tham số, trọng số và cấu trúc của mô hình. Việc lưu mô hình này rất quan trọng, vì nó cho phép hệ thống sử dụng lại mô hình đã huấn luyện mà không cần phải trải qua quá trình huấn luyện tốn thời gian và tài nguyên máy tính mỗi lần cần sử dụng.

Khi người dùng muốn sử dụng hệ thống nhận dạng biển báo giao thông, mô hình đã được huấn luyện và lưu trữ sẽ được **tải lại vào hệ thống**. Mô hình này sẽ được sử dụng để thực hiện việc nhận dạng và phân loại các biển báo giao thông từ các hình ảnh mà người dùng tải lên qua giao diện người dùng (GUI). Việc tải lại mô hình giúp hệ thống hoạt động nhanh chóng và hiệu quả, đồng thời giảm bớt thời gian xử lý, vì không cần phải huấn luyện lại mô hình từ đầu mỗi lần sử dụng.

## 2. Biểu đồ Luồng Dữ liệu Mức Ngữ cảnh (Context-Level Data Flow Diagram - DFD).

### 2.1. Người dùng → Hệ thống.

Quá trình bắt đầu khi người dùng tương tác với giao diện người dùng (GUI) của hệ thống. Người dùng tải lên một hình ảnh biển báo giao thông mà họ muốn phân loại. Giao diện người dùng đóng vai trò là điểm đầu vào của hệ thống, nơi hình ảnh được cung cấp dưới dạng dữ liệu đầu vào. Người dùng có thể dễ dàng lựa chọn và tải lên tệp hình ảnh từ thiết bị của họ thông qua các chức năng của GUI. Sau khi tải lên, hình ảnh này sẽ được gửi tới hệ thống để tiếp tục quá trình xử lý. Hệ thống → Mô hình CNN: Sau khi nhận được hình ảnh từ người dùng, hệ thống tiến hành tiền xử lý hình ảnh. Quá trình này bao gồm các bước như thay đổi kích thước và chuẩn hóa hình ảnh để đảm bảo chúng phù hợp với yêu cầu đầu vào của mô hình CNN. Sau khi tiền xử lý, hình ảnh được đưa vào mô hình CNN để tiến hành phân loại biển báo giao thông.

### 2.2. Hệ thống → Mô hình CNN.

Sau khi hình ảnh được người dùng tải lên, hệ thống thực hiện các bước tiền xử lý để chuẩn bị dữ liệu cho mô hình học sâu (CNN). Quá trình tiền xử lý có thể bao gồm các bước như thay đổi kích thước hình ảnh để phù hợp với yêu cầu đầu vào của mô hình, chuẩn hóa dữ liệu để các giá trị pixel nằm trong một phạm vi xác định, và có thể là việc chuyển đổi định dạng hình ảnh nếu cần thiết. Các hình ảnh sau khi đã được chuẩn bị sẽ được đưa vào mô hình CNN. Mô hình CNN (Convolutional Neural Network) này được sử dụng để nhận dạng và phân loại biển báo giao thông từ hình ảnh đầu vào.

### 2.3. Mô hình CNN → Hệ thống.

Mô hình CNN sau khi tiếp nhận hình ảnh đã qua tiền xử lý sẽ thực hiện quá trình phân loại. Mô hình này sẽ phân tích các đặc trưng trong hình ảnh và đưa ra kết quả phân loại, ví dụ như "biển báo dừng", "biển báo tốc độ", hay "biển báo cấm", tùy thuộc vào nội dung hình ảnh biển báo. Đồng thời, mô hình cũng trả về một giá trị độ chính xác, thể hiện mức độ tự tin của mô hình đối với kết quả phân loại mà nó đưa ra. Giá trị này là một chỉ số quan trọng, giúp người dùng đánh giá độ tin cậy của kết quả.

### 2.4. Hệ thống → Người dùng.

Cuối cùng, sau khi nhận được kết quả phân loại và độ chính xác từ mô hình CNN, hệ thống sẽ xử lý và hiển thị thông tin này lên giao diện người dùng. Người dùng sẽ thấy kết quả phân loại biển báo giao thông (loại biển báo và nội dung của nó) và mức độ chính xác của mô hình trong việc nhận dạng hình ảnh. Mức độ chính xác này sẽ giúp người dùng hiểu được khả năng của hệ thống trong việc phân loại đúng biển báo giao thông và đánh giá hiệu quả của quá trình nhận dạng.

# CÀI ĐẶT VÀ CHẠY PHẦN MỀM

## 1. Cài đặt.

Để triển khai và chạy một dự án nhận dạng biển báo giao thông sử dụng **Convolutional Neural Network (CNN)**, bạn cần cài đặt một số phần mềm và thư viện. Dưới đây là hướng dẫn chi tiết về cách cài đặt môi trường và các công cụ cần thiết.

### 1.1. Cài đặt Python.

**Tải và cài đặt Python**.

Truy cập trang web chính thức của Python: <https://www.python.org/downloads/>

Tải xuống phiên bản Python mới nhất (khuyến khích cài đặt Python 3.x).

Trong quá trình cài đặt, đảm bảo tích chọn tùy chọn **Add Python to PATH**.

### 1.2. Cài Đặt Các Thư Viện Python Cần Thiết.

**Cài đặt TensorFlow và Keras**: TensorFlow là thư viện chủ yếu để xây dựng và huấn luyện các mô hình học sâu, còn Keras là API cao cấp của TensorFlow giúp dễ dàng xây dựng mô hình. Để cài đặt TensorFlow và Keras, bạn sử dụng pip:

pip install tensorflow

**Cài đặt OpenCV**: OpenCV là thư viện xử lý hình ảnh, được sử dụng để xử lý ảnh và video trong dự án nhận dạng biển báo giao thông.

pip install opencv-python

**Cài đặt PIL (Pillow)**: Pillow là thư viện xử lý hình ảnh trong Python, giúp bạn đọc, hiển thị và thay đổi kích thước ảnh.

pip install pillow

**Cài đặt NumPy**: NumPy là thư viện tính toán số học giúp xử lý mảng dữ liệu.

pip install numpy

**Cài đặt Matplotlib**: Matplotlib là thư viện vẽ biểu đồ và trực quan hóa dữ liệu.

pip install matplotlib

**Cài đặt Scikit-learn**: Scikit-learn hỗ trợ các thuật toán học máy, chẳng hạn như phân chia dữ liệu và tính toán độ chính xác của mô hình.

pip install scikit-learn

**Cài đặt Tkinter** (nếu chưa cài): Tkinter là thư viện xây dựng giao diện người dùng (GUI) trong Python. Tkinter thường đã được cài sẵn với Python, nhưng nếu chưa có, bạn có thể cài đặt nó bằng cách:

pip install tk

**Cài đặt HDF5**: Để lưu và tải mô hình đã huấn luyện, bạn cần thư viện h5py để làm việc với định dạng HDF5.

pip install h5py

### 1.3. Cài Đặt IDE (Môi Trường Phát Triển).

Để viết và chạy mã Python, bạn có thể sử dụng một trong các IDE sau:

**Visual Studio Code**: Cài đặt từ <https://code.visualstudio.com/>.

Cài đặt extension Python từ marketplace.

**PyCharm**: Cài đặt từ <https://www.jetbrains.com/pycharm/>.

**Jupyter Notebook**: Cài đặt từ <https://jupyter.org/>, rất phù hợp cho việc phát triển mã học sâu và phân tích dữ liệu.

Để cài đặt Jupyter Notebook:

pip install notebook

### 1.4. Cài Đặt CUDA và cuDNN (Cho GPU - Tuỳ Chọn).

Nếu bạn muốn tận dụng GPU để tăng tốc quá trình huấn luyện mô hình, bạn cần cài đặt **CUDA** và **cuDNN**:

**CUDA**: Tải từ [NVIDIA CUDA Toolkit](https://developer.nvidia.com/cuda-toolkit).

**cuDNN**: Tải từ [NVIDIA cuDNN](https://developer.nvidia.com/cudnn).

## 2. Chạy phần mềm.

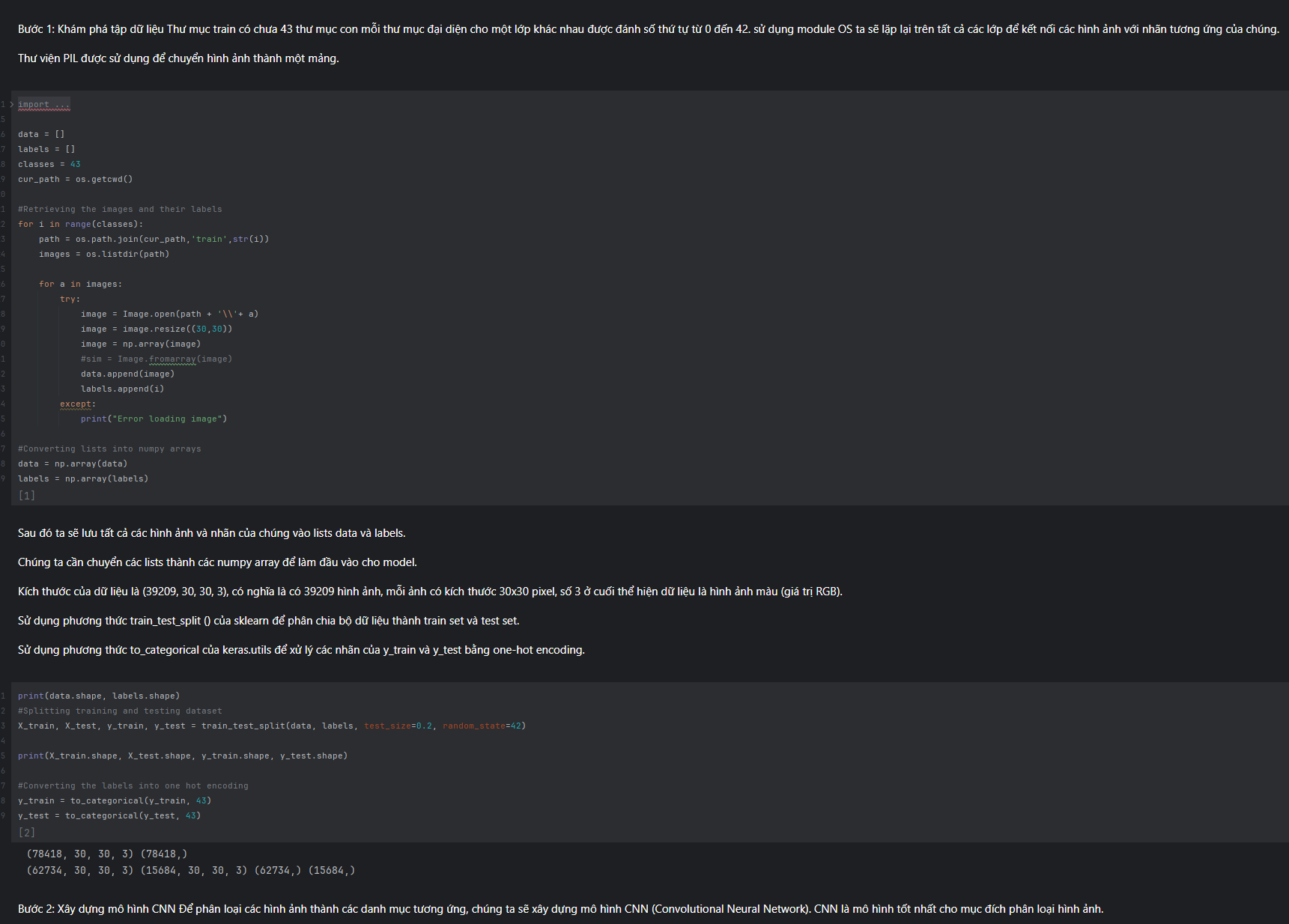
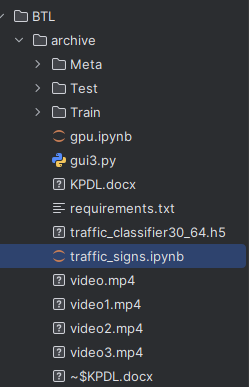
### 2.1. Chạy file traffic\_signs.ipynb

Bước 1: Khám phá tập dữ liệu.

dữ liệu hình ảnh được tải từ nguồn.

<https://www.kaggle.com/api/v1/datasets/download/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign>

Sau đó sắp xếp lại cây thư mục giống như hình bên dưới.

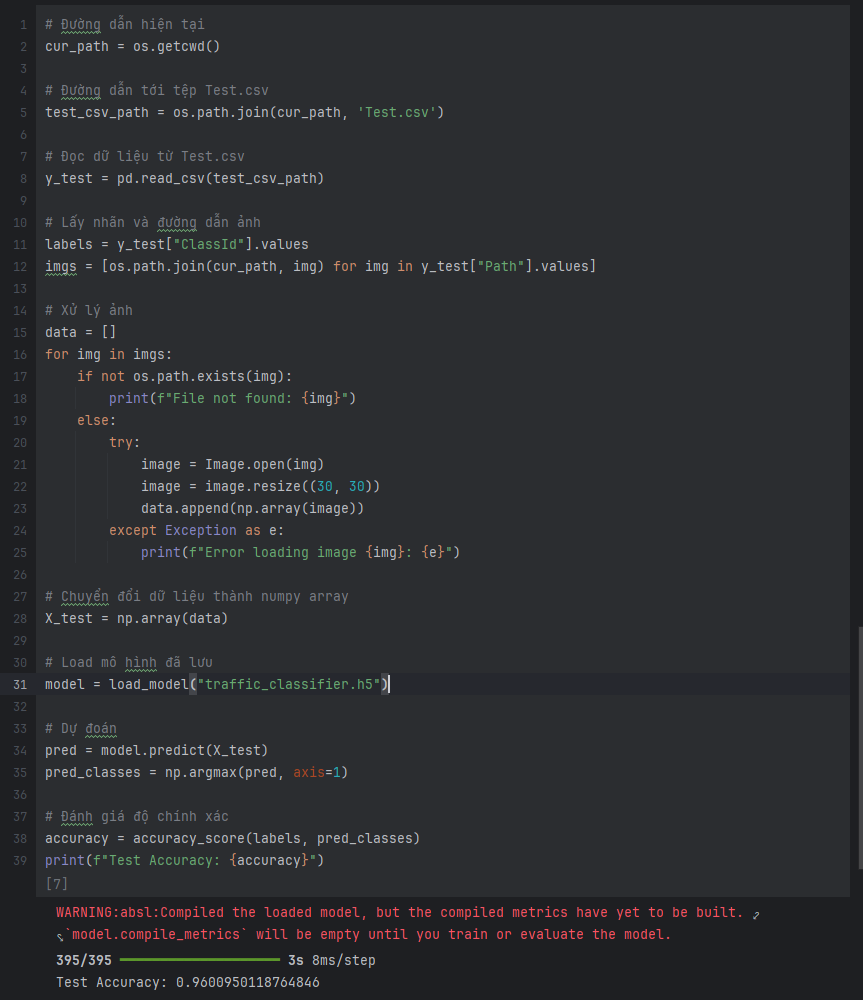


Bước 2: Xây dựng mô hình CNN.



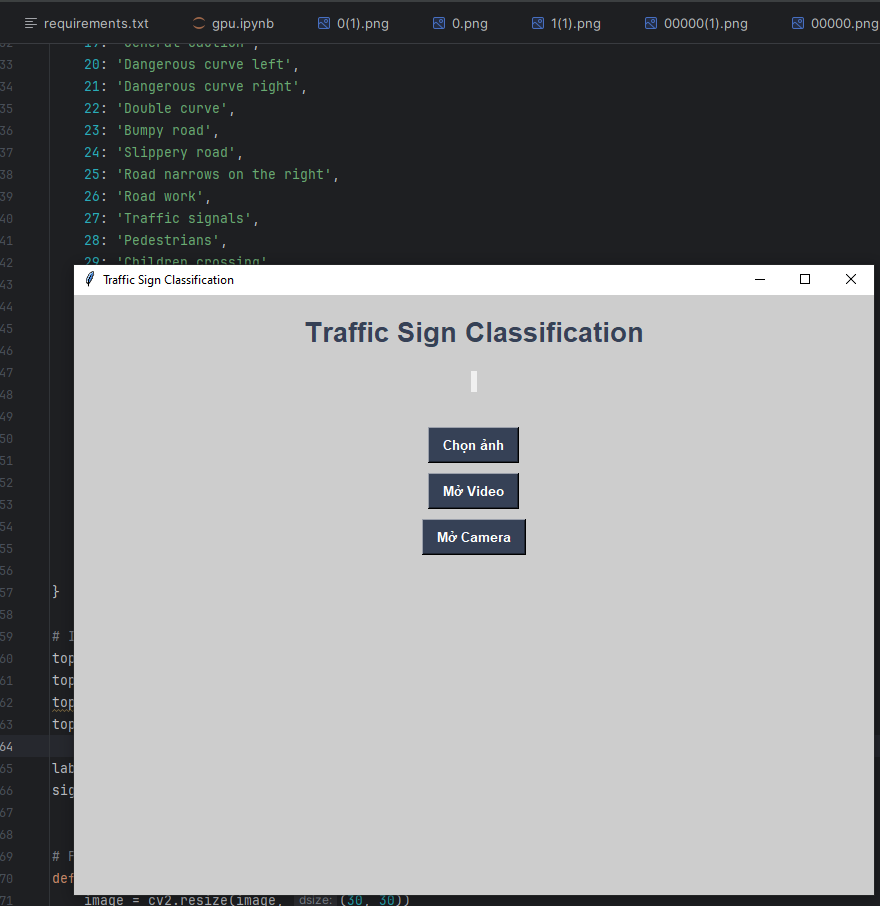
Sau khi huấn luyện xong sẽ lưu mô hình với tên là  
model.save("traffic\_classifier.h5")

Bước 3: Đánh giá độ chính xác.



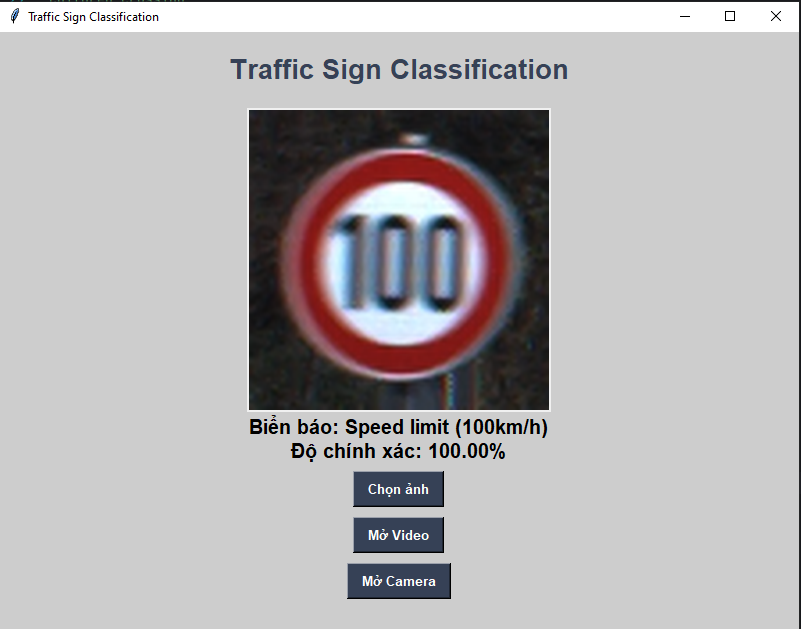
Kiểm tra độ chính xác của mô hình

### 2.2. Chạy file gui.py

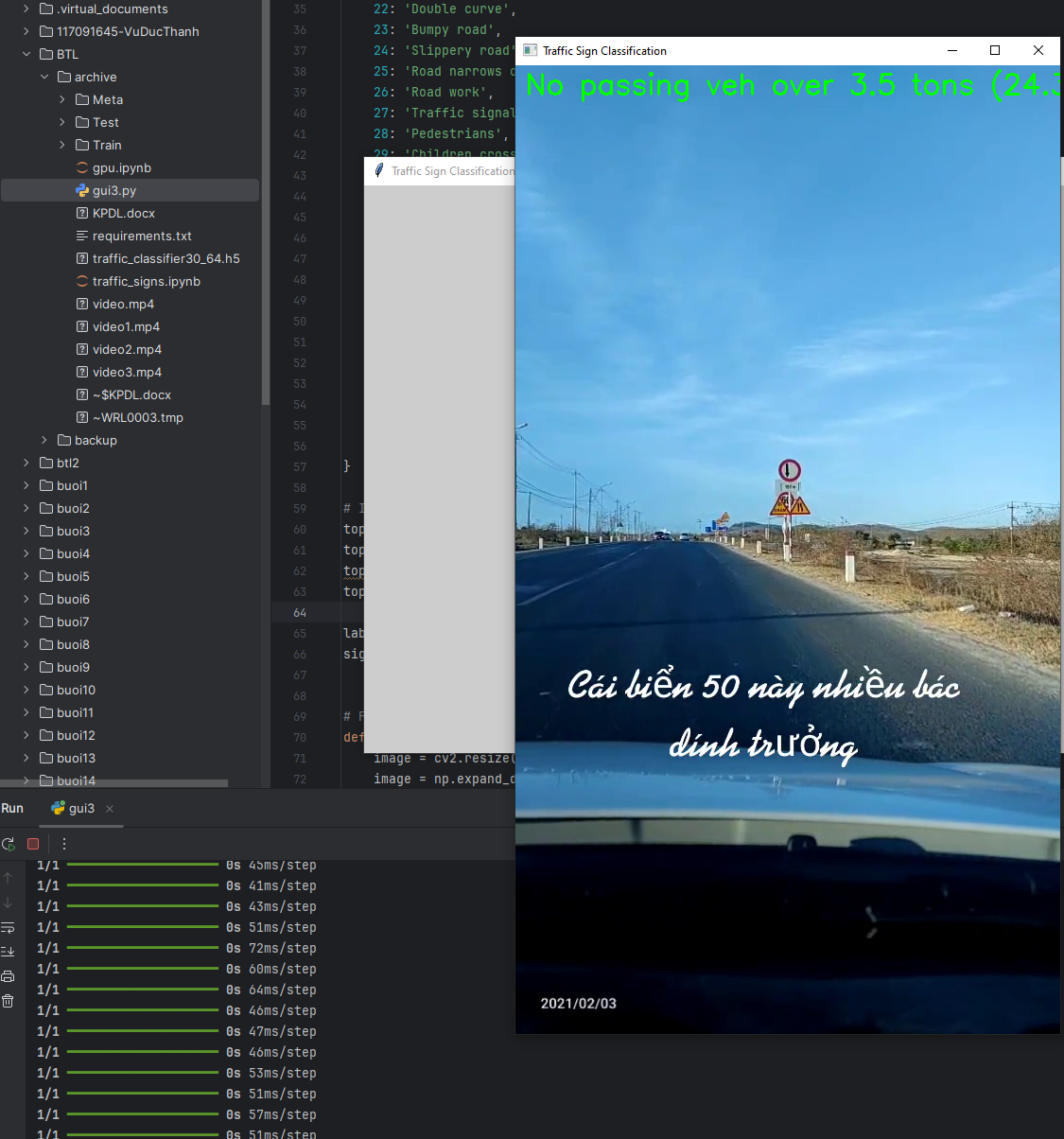


Giao diện của gui.py

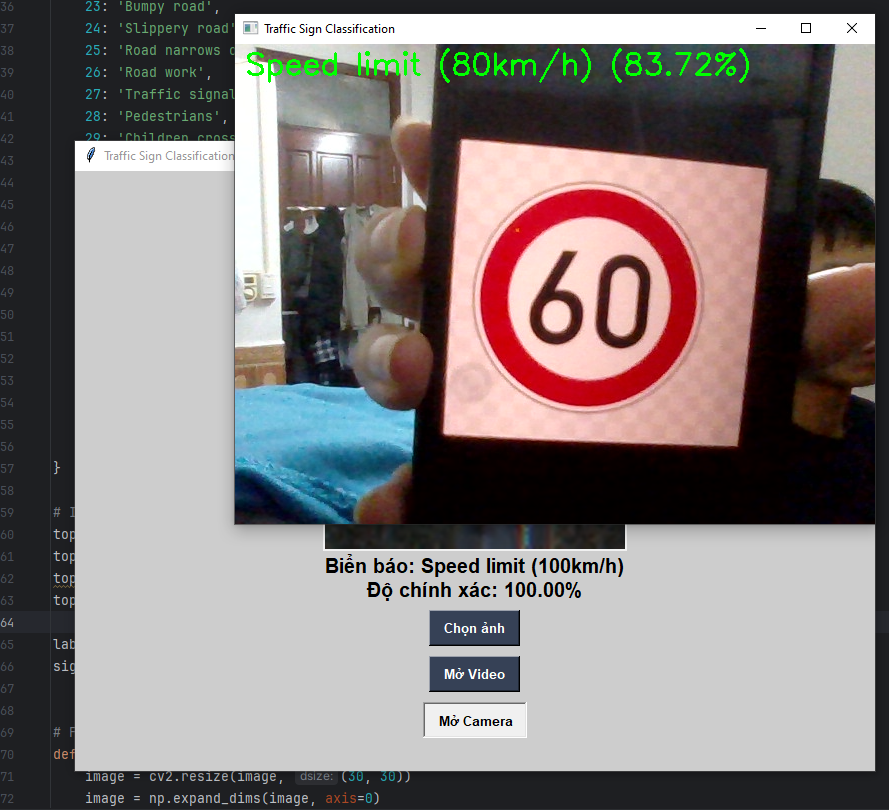
Sẽ cho phép chọn ảnh, chọn video và chọn camera



Khi sử dụng chức năng chọn ảnh



Khi sử dụng chức năng chọn video



Khi sử dụng chức năng mở camera

# KẾT LUẬN

Trong quá trình triển khai một dự án học sâu sử dụng Convolutional Neural Networks (CNN) cho nhận dạng biển báo giao thông, việc cài đặt đúng phần mềm và thư viện là yếu tố then chốt để đảm bảo mô hình có thể hoạt động hiệu quả và ổn định. Một môi trường phát triển hoàn chỉnh không chỉ giúp bạn huấn luyện mô hình nhanh chóng mà còn đảm bảo tính linh hoạt và dễ dàng mở rộng của dự án.

Các công cụ và thư viện như TensorFlow và Keras là nền tảng cơ bản giúp xây dựng, huấn luyện và đánh giá các mô hình học sâu. TensorFlow, với khả năng tối ưu hóa và tính toán song song, kết hợp với Keras, một API dễ sử dụng, tạo ra môi trường phát triển lý tưởng cho việc thiết kế và triển khai các mô hình CNN. Ngoài ra, OpenCV và PIL (Pillow) là hai thư viện hỗ trợ mạnh mẽ trong việc xử lý ảnh đầu vào, từ việc chuẩn hóa hình ảnh đến việc thay đổi kích thước và hiển thị kết quả. NumPy là thư viện không thể thiếu trong việc xử lý dữ liệu số học, giúp thực hiện các phép toán trên mảng và dữ liệu hình ảnh.

Khi xây dựng các mô hình học sâu, việc sử dụng HDF5 (.h5) để lưu trữ và tải lại mô hình là một phần quan trọng. Định dạng này không chỉ giúp bạn bảo vệ mô hình đã huấn luyện mà còn tối ưu hóa thời gian sử dụng lại mô hình mà không phải huấn luyện lại từ đầu. Điều này đặc biệt quan trọng khi làm việc với các mô hình phức tạp, đòi hỏi nhiều tài nguyên tính toán.

Để tạo ra một hệ thống thân thiện với người dùng, bạn có thể sử dụng Tkinter, một thư viện GUI mạnh mẽ trong Python, để xây dựng giao diện người dùng đơn giản. Giao diện này giúp người dùng dễ dàng tải lên hình ảnh và nhận kết quả nhận dạng biển báo giao thông một cách trực quan và nhanh chóng.

Ngoài ra, việc cài đặt CUDA và cuDNN nếu sử dụng GPU có thể giúp bạn tối ưu hóa quá trình huấn luyện mô hình, làm giảm thời gian huấn luyện đáng kể, nhất là khi làm việc với lượng dữ liệu lớn. Các công cụ này giúp tận dụng tối đa sức mạnh của phần cứng, đặc biệt là khi mô hình có nhiều lớp và yêu cầu tài nguyên tính toán lớn.

Cuối cùng, việc cài đặt môi trường phát triển đúng cách và sử dụng các công cụ phù hợp là rất quan trọng để dự án của bạn có thể hoạt động một cách trơn tru và hiệu quả.

# TÀI LIỆU THAM KHẢO

**Olaf Ronneberger, Philipp Fischer, and Thomas Brox (2015).**

*U-Net: Convolutional Networks for Biomedical Image Segmentation*.

**TensorFlow Documentation**

*https://www.tensorflow.org/learn*  
Tài liệu chính thức của TensorFlow cung cấp rất nhiều hướng dẫn về cách xây dựng và huấn luyện các mô hình CNN, từ cơ bản đến nâng cao.

**Keras Documentation**

[*https://keras.io*](https://keras.io)

<https://www.kaggle.com/api/v1/datasets/download/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign>