ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG ĐIỆN - ĐIỆN TỬ

BÁO CÁO BÀI TẬP LỚN
TRÍ TUỆ NHÂN TẠO VÀ ỨNG DỤNG

Đề tài: Nhận diện hình khối

Giảng viên hướng dẫn: TS. Nguyễn Huy Hoàng

Nhóm sinh viên thực hiện:

- Vũ Lâm Huy 20224438
- Nguyễn An Khánh 20224439
- Bùi Ngọc Đạt 20224405

Hà Nội, 12 – 2024

Mục lục

Phân công công việc

Bảng 1.1 Dưới đây phân công công việc cho từng thành viên trong nhóm để triển khai bài tập lớn một cách hiệu quả.

Họ và tên	MSSV	Nội dung công việc
Vũ Lâm Huy	20224438	Viết code VXL, thu thập data, viết báo cáo
Nguyễn An Khánh	20224439	Tạo server xử lí AI, build model AI
Bùi Ngọc Đạt	20224405	Build model AI, thu thập data, viết báo cáo

Bảng 1.1 Phân công công việc

Lời nói đầu

Trí tuệ nhân tạo (AI) đang trở thành một trong những xu hướng công nghệ nổi bật nhất trong thời đại công nghiệp 4.0. AI không chỉ thay đổi cách con người tương tác với máy móc, mà còn đem lại những ứng dụng mang tính đổi mới và đột phá trong nhiều lĩnh vực khác nhau, từ y tế, giáo dục đến công nghiệp và giao thông. Trong đó, nhận diện hình khối là một được coi là nền tảng quan trọng cho nhiều ứng dụng AI.

Nhận diện hình khối không chỉ là việc nhận biết và phân loại các hình dáng khác nhau, mà còn giúp hỗ trợ trong việc phát triển các ứng dụng như robot, xe tự hành, hay kiểm tra chất lượng sản phẩm trong sản xuất. Với sự phát triển vũ bão của công nghệ xử lý hình ảnh và các thuật toán AI hiện đại, nhận diện hình khối đã trở thành một bước tiến lớn trong việc tự động hóa và tăng cường hiệu quả lao động.

Báo cáo này nhằm mục đích khám phá các nguyên lý cơ bản về nhận diện hình khối, đồng thời đề xuất một ứng dụng minh hoạ trong việc nhận diện các hình dáng khác nhau. Trong quá trình thực hiện, chúng em đã áp dụng nhiều kiến thức và công cụ AI để đạt được kết quả tối ưu. Chúng em hy vọng rằng.

Chương 1: Tổng quan về đề tài

1.1 Đặt vấn đề

Trong bối cảnh phát triển vượt bậc của trí tuệ nhân tạo (AI), việc ứng dụng công nghệ này để giải quyết các vấn đề trong thực tiễn ngày càng trở nên phổ biến. Một trong những lĩnh vực quan trọng là nhận diện và phân loại hình khối, một kỹ thuật không chỉ hữu ích trong công nghiệp mà còn trong các ứng dụng đời sống hằng ngày.

Hiện nay, nhu cầu kiểm tra, phân loại sản phẩm trong các dây chuyền sản xuất ngày càng tăng cao, đặc biệt đối với các ngành yêu cầu độ chính xác cao như điện tử, thực phẩm, và sản xuất vật liệu. Các phương pháp truyền thống dựa trên sức lao động con người hoặc thiết bị cơ khí đã không còn đáp ứng được yêu cầu về tốc độ và độ chính xác trong thời đại công nghiệp 4.0.

Trí tuệ nhân tạo, kết hợp với các hệ thống nhúng như ESP32-CAM, mang đến một giải pháp hiệu quả, nhanh chóng và kinh tế để giải quyết vấn đề này. Với khả năng xử lý hình ảnh và nhận diện đối tượng, các hệ thống này có thể tự động hóa quá trình nhận diện hình khối, góp phần nâng cao hiệu suất và giảm thiểu sai sót.

Dự án "Nhận diện hình khối" không chỉ nhằm hiện thực hóa một ứng dụng AI đơn giản mà còn thể hiện tiềm năng to lớn của công nghệ AI trong việc tối ưu hóa sản xuất và nâng cao chất lượng cuộc sống. Thông qua đề tài này, chúng ta sẽ tìm hiểu cách tích hợp AI vào thực tiễn, cụ thể là xây dựng một hệ thống nhận diện hình khối hoạt động hiệu quả.

1.2 Mục tiêu

Mục tiêu chính của đề tài "Nhận diện hình khối" là nghiên cứu và triển khai một hệ thống tích hợp trí tuệ nhân tạo (AI) để nhận diện và phân loại các hình khối dựa trên hình ảnh. Cụ thể, các mục tiêu bao gồm:

Nghiên cứu cơ sở lý thuyết: Tìm hiểu các khái niệm cơ bản về nhận diện hình ảnh, các thuật toán học sâu (Deep Learning), và các kỹ thuật xử lý ảnh để ứng dụng trong bài toán nhận diện hình khối.

Xây dựng hệ thống nhận diện:

Thiết kế một mô hình AI phù hợp để nhận diện và phân loại hình khối từ hình ảnh thu được.

Tích hợp mô hình AI với phần cứng ESP32-CAM để tạo ra một hệ thống nhúng hoạt động độc lập.

Kiểm tra và đánh giá: Đánh giá hiệu suất của hệ thống dựa trên các tiêu chí như độ chính xác, tốc độ xử lý và khả năng hoạt động trong môi trường thực tế.

Tối ưu hóa hệ thống: Cải thiện hệ thống để đảm bảo tính ổn định, dễ sử dụng và khả năng mở rộng trong các ứng dụng thực tiễn.

1.3 Nội dung nghiên cứu

Bài tập lớn được nghiên cứu theo phương pháp thu thập dữ liệu, nghiên cứu lý thuyết và thực nghiệm, theo thứ tự sau:

- Thu thập tài liệu nghiên cứu, lý thuyết và các mô hình thực nghiệm.

- Tìm hiểu, khảo sát các phương pháp hiệu quả.

- Tìm kiếm dữ liệu phù hợp với mô hình sẽ xây dựng.

- Tiến hành huấn luyện nhận diện biển báo giao thông bằng phương pháp đã lựa chọn.

- Đánh giá và hoàn chỉnh lại mô hình dựa trên mục tiêu đã đề ra.

1.4 Công nghệ sử dụng

Phần cứng:
- ESP32-CAM: Camera tích hợp Wi-Fi và Bluetooth.
- Nguồn cung cấp: 5V.

Phần mềm:
- Ngôn ngữ: Python.
- Công cụ phát triển: Arduino IDE, Python cho train model.
- Mô hình CNN: Được train với dữ liệu từ Kaggle

1.5 Kết luận chương 1

Trên đây chúng em đã giới thiệu sơ bộ về mục đích của AI về nhận diện hình khối, cũng như là phân công công việc trong nhóm để triển khai công việc. Hi vọng chương đầu tiên này đã đem lại những cái nhìn tổng quan và những hiểu biết cơ bản về AI mà nhóm em muốn xây dựng. Tiếp theo sau chương I, nhóm em sẽ trình bày chi tiết về những yêu cầu đặt ra và phương pháp thiết kế chi tiết cho bài của mình.

Chương 2 : Nguyên lý hoạt động

2.1 Chỉ tiêu thiết kế

Để xây dựng hệ thống nhận diện hình khối tích hợp AI trên ESP32-CAM, các chỉ tiêu thiết kế được xác định như sau:

- **Tính chính xác**

Hệ thống phải đạt độ chính xác tối thiểu 90% trong việc nhận diện và phân loại các hình khối cơ bản (ví dụ: hình vuông, hình tròn, hình tam giác).

- **Tốc độ xử lý**

Thời gian xử lý mỗi khung hình phải dưới 1 giây, đảm bảo hệ thống hoạt động nhanh và hiệu quả trong điều kiện thực tế.

- **Khả năng hoạt động trên phần cứng hạn chế**

Hệ thống phải được tối ưu hóa để chạy mượt mà trên ESP32-CAM, một thiết bị có tài nguyên hạn chế về bộ nhớ và sức mạnh xử lý.

- **Tính ổn định và độ tin cậy**

Hệ thống phải hoạt động ổn định trong các điều kiện môi trường khác nhau, bao gồm ánh sáng thay đổi và nền phức tạp.

Đảm bảo tỷ lệ lỗi (false positives/false negatives) không vượt quá 10%.

- **Khả năng triển khai thực tế**

Hệ thống phải dễ dàng triển khai và sử dụng, bao gồm khả năng tích hợp với các ứng dụng khác hoặc hệ thống tự động hóa.

Kích thước mã nguồn và mô hình phải nhỏ gọn để phù hợp với bộ nhớ hạn chế của ESP32-CAM.

- **Tính linh hoạt**

Hệ thống phải có khả năng mở rộng để nhận diện thêm các hình khối hoặc đối tượng khác trong tương lai mà không yêu cầu thay đổi lớn về kiến trúc.

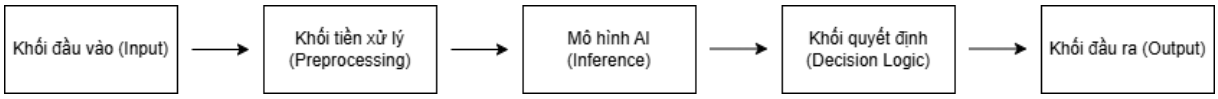
- **Hiệu quả kinh tế**

Phải đảm bảo chi phí thấp trong toàn bộ quá trình thiết kế, triển khai và vận hành, tận dụng tối đa phần cứng sẵn có.

- **Thân thiện với người dùng**

Giao diện hoặc cách vận hành của hệ thống phải đơn giản, dễ hiểu để người dùng không cần chuyên môn cao vẫn có thể sử dụng.

2.2 Sơ đồ chức năng



Hình 2.1 Sơ đồ khối chức năng

Chức năng từng khối:

- Khối đầu vào: Thu thập hình ảnh đầu vào từ môi trường thực tế.
- Khối tiền xử lý: Xử lý và chuẩn bị dữ liệu hình ảnh trước khi đưa vào AI.
- Mô hình AI: Nhận diện hình khối từ ảnh đầu vào.
- Khối quyết định: Phân tích và xử lý kết quả từ mô hình AI.
- Khối đầu ra: Hiển thị hoặc truyền thông tin nhận diện.

2.3 Tính toán và lựa chọn thiết bị phần cứng

Nhóm chúng em quyết định lựa chọn ESP32 – CAM làm phần cứng chính cho hệ thống nhận diện hình khối được dựa trên các lý do sau:

-Khả năng tích hợp cao: ESP32-CAM là một module nhỏ gọn, tích hợp cả camera và khả năng xử lý mạnh mẽ. Với vi xử lý ESP32 và khả năng hỗ trợ mạng Wi-Fi, module này phù hợp để triển khai các ứng dụng AI/IoT mà không cần thêm nhiều linh kiện bổ sung.

-Hỗ trợ xử lý AI: ESP32-CAM có khả năng chạy các mô hình AI nhỏ gọn, chẳng hạn như TensorFlow Lite hoặc các mô hình được tối ưu hóa. Điều này phù hợp với nhu cầu nhận diện hình khối mà không cần máy chủ bên ngoài.

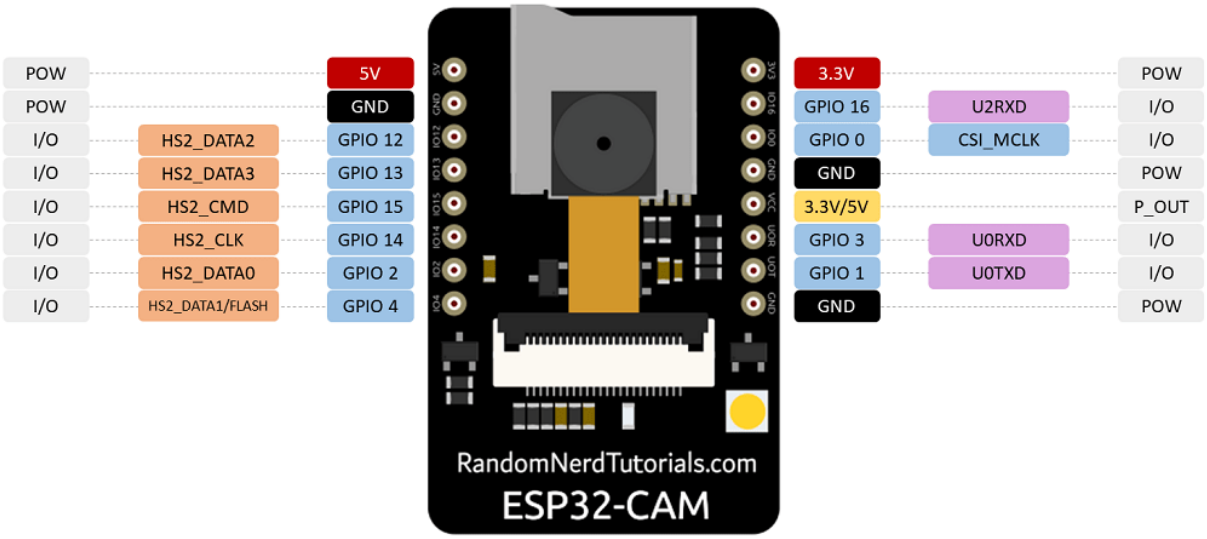
-Chi phí hợp lý: ESP32-CAM có giá thành thấp hơn so với nhiều giải pháp phần cứng khác. Điều này làm giảm chi phí tổng thể của dự án, đặc biệt quan trọng trong bối cảnh nghiên cứu hoặc ứng dụng thực tế quy mô nhỏ.

-Độ phổ biến và cộng đồng hỗ trợ lớn: ESP32-CAM có một cộng đồng người dùng và tài liệu phong phú, giúp dễ dàng tiếp cận tài nguyên, giải quyết lỗi, hoặc cải thiện hiệu năng hệ thống.

2.3.1. Thông số kỹ thuật

Thông số	Chi tiết
Vi xử lý	ESP32-D0WD, lõi kép, 32-bit, tốc độ lên đến 240 MHz
RAM	520 KB SRAM, 4 MB PSRAM
Bộ nhớ Flash	4 MB Flash
Wi-Fi	Chuẩn 802.11 b/g/n, hỗ trợ cả Access Point và Station
Bluetooth	Bluetooth 4.2 (BR/EDR và BLE)
Camera	OV2640 (2 MP), hỗ trợ độ phân giải tối đa 1600x1200
Giao thức camera	DVP
Cổng giao tiếp	UART, SPI, I2C, PWM, ADC, DAC, GPIO
Số chân GPIO	Lên đến 9 chân GPIO khả dụng
Điện áp hoạt động	3.3V hoặc 5V
Dòng điện tiêu thụ	160 mA ở chế độ hoạt động
Thẻ nhớ	Hỗ trợ khe cắm MicroSD (tối đa 4 GB)
Kích thước	27 x 40.5 mm
Cổng kết nối	Đầu nối UART (FTDI hoặc tương đương để nạp chương trình)
Tích hợp đèn Flash LED	LED Flash hỗ trợ camera (dùng làm đèn hoặc tín hiệu)
Tính năng khác	Chế độ Sleep tiết kiệm năng lượng

Bảng 2.1 Thông số kỹ thuật ESP32 CAM



Hình 2.2 Hình ảnh và sơ đồ chân ESP32 CAM

2.4 So sánh và lựa chọn mô hình

Tiêu chí	CNN	YOLOv4	YOLOv5	SSD
Độ phức tạp mô hình	Thấp	Cao	Cao	Trung bình
Tốc độ xử lý (FPS)	Thấp	Cao (Real-time)	Cao hơn YOLOv4	Trung bình
Độ chính xác (mAP)	Phụ thuộc vào thiết kế	Cao (trong bài toán tổng quát)	Cao hơn YOLOv4	Tốt nhưng kém hơn YOLO
Yêu cầu phần cứng	Thấp (ESP32-CAM khả thi)	Cao (GPU cần thiết)	Cao (nhẹ hơn YOLOv4)	Trung bình (cần GPU)
Tính dễ triển khai	Dễ triển khai, tùy chỉnh	Phức tạp, cần nhiều tinh chỉnh	Đơn giản hơn YOLOv4	Trung bình, ít linh hoạt
Khả năng mở rộng	Linh hoạt, dễ tùy chỉnh	Hạn chế	Hạn chế	Khá tốt nhưng cố định

Bảng 2.2 So sánh lựa chọn mô hình AI

Từ những yêu cầu trên cộng với việc nhìn nhận các ưu nhược điểm của mỗi phương pháp nhận thấy CNN là lựa chọn phù hợp nhất để xây dựng hệ thống nhận diện hình khối.

2.5 Chi tiết về phương pháp

2.5.1 Tổng quan về CNN (Convolutional Neural Network)

Convolutional Neural Network (CNN) là một loại mạng thần kinh nhân tạo chuyên dụng, được thiết kế đặc biệt để xử lý và phân tích dữ liệu có cấu trúc lưới, chẳng hạn như hình ảnh hoặc tín hiệu thời gian. CNN được sử dụng rộng rãi trong các bài toán về thị giác máy tính như phân loại hình ảnh, nhận diện đối tượng, xử lý video, và cả trong các lĩnh vực khác như xử lý văn bản.

a) Kiến trúc cơ bản của CNN

CNN bao gồm các lớp chính sau:

- **Lớp tích chập (Convolutional Layer)**:
    - Đây là lớp cốt lõi của CNN, sử dụng các bộ lọc (filters/kernels) để trích xuất các đặc trưng từ hình ảnh đầu vào, chẳng hạn như cạnh, góc, hoặc kết cấu.
    - Bộ lọc trượt trên hình ảnh đầu vào, thực hiện phép toán tích chập giữa bộ lọc và vùng ảnh tương ứng, tạo ra bản đồ đặc trưng (feature map).
- **Lớp kích hoạt (Activation Layer)**:
    - Thường sử dụng hàm kích hoạt phi tuyến như ReLU (Rectified Linear Unit), giúp mạng học được các mối quan hệ phi tuyến giữa đầu vào và đầu ra.
- **Lớp gộp (Pooling Layer)**:
    - Lớp này giảm kích thước không gian của bản đồ đặc trưng, làm giảm số lượng tham số và tính toán trong mạng. Phổ biến nhất là Max Pooling (lấy giá trị lớn nhất trong vùng) và Average Pooling (lấy giá trị trung bình).
    - Điều này giúp giảm thiểu hiện tượng overfitting.
- **Lớp kết nối đầy đủ (Fully Connected Layer)**:
    - Lớp này kết nối tất cả các neuron với nhau và thường được sử dụng ở cuối mạng để dự đoán kết quả.
- **Lớp chuẩn hóa (Normalization Layer)**:
    - Giúp chuẩn hóa dữ liệu giữa các lớp để tăng tốc độ huấn luyện và ổn định mạng.

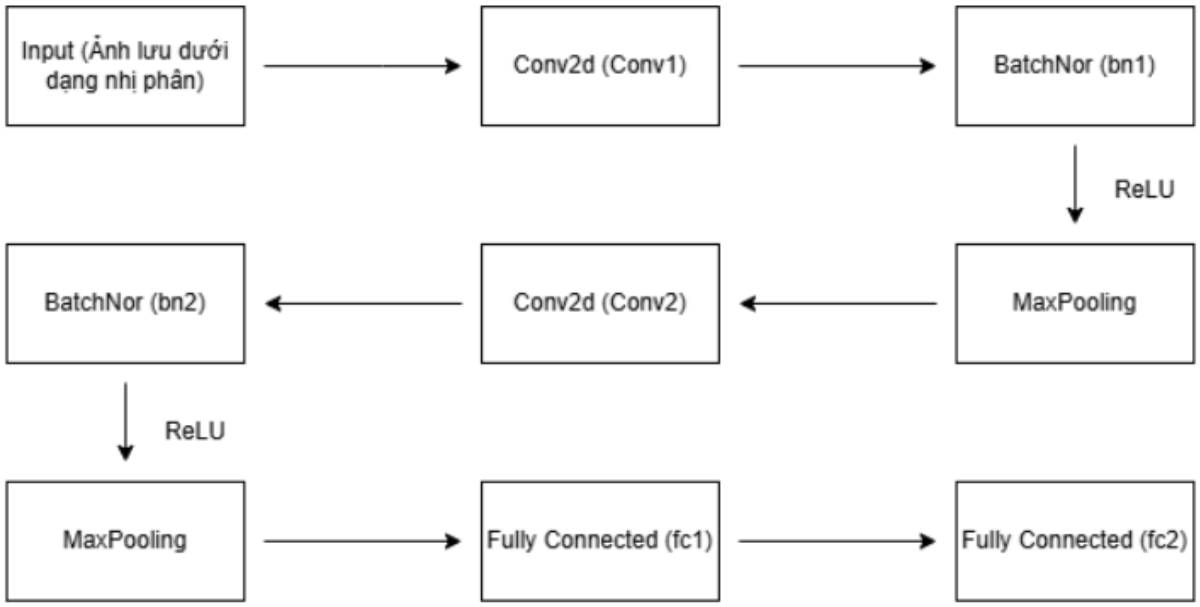
b) Đặc điểm nổi bật của CNN

- **Tự động trích xuất đặc trưng**:
    - Khác với các phương pháp truyền thống, CNN tự động học và trích xuất các đặc trưng quan trọng từ dữ liệu đầu vào mà không cần thiết kế thủ công.
- **Không gian tham số ít hơn**:
    - Bộ lọc được chia sẻ trọng số trên toàn bộ ảnh, làm giảm đáng kể số lượng tham số cần học.
- **Khả năng xử lý dữ liệu không gian**:

- CNN khai thác thông tin không gian (spatial) trong dữ liệu, như mối quan hệ giữa các điểm ảnh gần nhau trong hình ảnh.

2.5.2 Mô hình CNN mà nhóm sử dụng

Gồm các lớp Convolutional Layer , Activation Function, Pooling , Flattening và Fully Connected



Hình 2.3 Mô hình CNN của nhóm

Chương 3: Quy trình triển khai

3.1 Thu thập data cho huấn luyện mô hình AI

3.1.1 Sử dụng các data set có sẵn trên mạng

Kaggle

- **Lợi ích:**
    - Cung cấp hàng ngàn dataset về hình ảnh, văn bản, số liệu, v.v.
    - Có sẵn các cuộc thi và notebook hướng dẫn sử dụng dataset.

Google Dataset Search

- **Lợi ích:**
    - Công cụ tìm kiếm dataset của Google.
    - Cung cấp dataset từ nhiều nguồn khác nhau (khoa học, thương mại, giáo dục).

UCI Machine Learning Repository

- **Lợi ích:**
    - Tập trung vào các bài toán học máy truyền thống.
    - Phù hợp với các bài toán về dữ liệu dạng bảng (tabular data).

Open Data on Github

- **Lợi ích:**
    - Tập hợp nhiều dataset từ các nguồn khác nhau.

3.1.2 Chụp ảnh trực tiếp từ máy

Thiết bị phần cứng: Thiết bị chụp ảnh: ESP32-CAM

ESP32-CAM là một module camera tích hợp vi điều khiển ESP32, được thiết kế để chụp ảnh và truyền dữ liệu thông qua kết nối Wi-Fi hoặc Bluetooth. Đây là lựa chọn phổ biến nhờ kích thước nhỏ gọn, chi phí thấp, và tính linh hoạt cao.

- **Thông số kỹ thuật chính:**
    - Độ phân giải: Lên đến 1600x1200 (UXGA).
    - Định dạng ảnh: JPEG, BMP, hoặc grayscale.
    - Kết nối: Wi-Fi, Bluetooth.
    - Tích hợp thẻ nhớ microSD: Hỗ trợ lưu trữ dữ liệu cục bộ.
- **Ưu điểm:**
    - Gọn nhẹ, dễ triển khai.
    - Phù hợp cho các ứng dụng IoT và AI.
    -

Hệ thống hỗ trợ: PC

PC đóng vai trò điều khiển, lập trình, và xử lý dữ liệu được gửi từ ESP32-CAM. Nó cung cấp:

- Môi trường lập trình (Arduino IDE) để cấu hình và điều khiển ESP32-CAM.
- Bộ nhớ và năng lực xử lý mạnh mẽ để lưu trữ, xử lý dữ liệu ảnh và triển khai mô hình AI.

Phần mềm điều khiển camera: Arduino IDE

Arduino IDE là môi trường lập trình phổ biến để lập trình và điều khiển ESP32-CAM. Với giao diện thân thiện và thư viện hỗ trợ sẵn, Arduino IDE giúp dễ dàng cấu hình và thực hiện các tác vụ như:

- Khởi tạo camera.
- Chụp ảnh và lưu trữ cục bộ hoặc truyền qua mạng.
- Thiết lập các thông số ảnh như độ phân giải, định dạng, và tốc độ chụp.

Tự động hóa việc chụp: Cài đặt chế độ chụp tự động theo thời gian

ESP32-CAM có thể được lập trình để tự động chụp ảnh sau một khoảng thời gian nhất định và gửi ảnh về PC hoặc lưu trữ cục bộ.

3.2 Tiền xử lí

3.2.1 Xử lí ảnh sang dạng grayscale

```
image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
image = cv2.resize(image, (64, 64))
image = image / 255.0
image = image.reshape(1, 64, 64, 1)
image.tofile(output_path)
```

1. **Thư viện được sử dụng**

- **os:** Thư viện xử lý hệ thống tệp và đường dẫn.
- **cv2:** OpenCV, dùng để đọc, xử lý ảnh.
- **numpy (np):** Dùng để thao tác trên dữ liệu ảnh dưới dạng mảng số (numpy array).

---

2. Chức năng của hàm

2.1. Hàm save_image_to_binary

1. **Đọc ảnh dưới dạng grayscale:**
    - Ảnh được chuyển sang dạng grayscale (ảnh đen trắng) thay vì RGB.

- Điều này giúp giảm kích thước dữ liệu và đơn giản hóa bài toán.

2. **Resize ảnh về kích thước chuẩn (64x64):**

   Kích thước ảnh được chuẩn hóa để đảm bảo tất cả ảnh đầu vào có cùng kích thước, phù hợp với đầu vào của mô hình CNN.

3. **Chuẩn hóa dữ liệu ảnh:**
   - Pixel của ảnh grayscale nằm trong khoảng [0, 255].
   - Việc chia cho 255.0 đưa các giá trị pixel về khoảng [0, 1], giúp mạng CNN học hiệu quả hơn.
4. **Thay đổi hình dạng mảng (reshape):**
   - Mảng được thay đổi thành dạng 4D: (batch_size, height, width, channels).
     - batch_size = 1: Một ảnh duy nhất.
     - height = 64, width = 64: Kích thước ảnh.
     - channels = 1: Vì ảnh grayscale chỉ có một kênh màu.
   - Định dạng này chuẩn bị ảnh để đưa vào mô hình CNN.
5. **Lưu dữ liệu vào file nhị phân:**
   - Mảng ảnh được lưu dưới dạng file nhị phân (binary file), giúp giảm kích thước lưu trữ so với định dạng ảnh thông thường (như PNG, JPEG).

3. Ý nghĩa và ứng dụng

3.1. Ý nghĩa:

- **Chuẩn bị dữ liệu:** Tiền xử lý ảnh để phù hợp với đầu vào của mô hình CNN.
- **Lưu trữ hiệu quả:** Lưu ảnh dưới dạng nhị phân giúp giảm kích thước file và tăng tốc độ truy cập dữ liệu trong quá trình training.
- **Tăng hiệu suất mô hình:** Các bước như resize, chuẩn hóa pixel giúp mô hình AI học nhanh và chính xác hơn.

3.2. Ứng dụng:

- **Huấn luyện mô hình AI:** File nhị phân được dùng làm dữ liệu đầu vào cho mô hình CNN.
- **Xử lý hàng loạt:** Có thể áp dụng hàm này để chuẩn bị hàng loạt ảnh, lưu thành file nhị phân và tối ưu hiệu suất khi xử lý dữ liệu lớn.

3.3 Quy trình huấn luyện mô hình

3.3.1 Chuẩn bị dữ liệu

1.1 Dữ liệu từ Kaggle:

- Tập dữ liệu từ Kaggle được tải về, chứa các hình khối đã được gắn nhãn (hình vuông, tròn, tam giác, ngôi sao).
- Ảnh được kiểm tra, tiền xử lý như:
    - Resize tất cả ảnh về kích thước chuẩn (ví dụ: 64x64 pixel).
    - Chuẩn hóa giá trị pixel (đưa về khoảng [0, 1]).
    - Nếu cần, thực hiện **data augmentation** để tăng cường dữ liệu bằng cách xoay, lật, thay đổi độ sáng.

1.2. Ảnh tự chụp từ ESP32-CAM:

- Sử dụng ESP32-CAM để chụp ảnh thực tế các hình khối.
- Viết script trên ESP32 để tự động chụp ảnh theo thời gian và lưu trữ ảnh dưới dạng .png.
- Ảnh chụp được đưa vào tập dữ liệu, sau đó thực hiện tiền xử lý tương tự như dữ liệu Kaggle.

1.3. Chia tập dữ liệu:

- Dữ liệu được chia thành:
    - **Train set:** 70% dữ liệu.
    - **Validation set:** 20% dữ liệu, dùng để theo dõi hiệu suất mô hình trong quá trình train.
    - **Test set:** 10% dữ liệu, dùng để đánh giá mô hình sau khi train.

2. Xây dựng mô hình

Kiến trúc CNN:

Mô hình CNN bao gồm:

- **3 lớp convolutional:** Lớp đầu học các đặc trưng cơ bản (như cạnh, góc), các lớp tiếp theo học đặc trưng phức tạp hơn.
- **2 lớp fully connected (FC):** Tổng hợp các đặc trưng và đưa ra dự đoán cuối cùng.

3. Huấn luyện mô hình

3.1. Cấu hình mô hình:

- **Loss function:** categorical_crossentropy vì đây là bài toán phân loại nhiều lớp.
- **Metrics:** accuracy để đánh giá độ chính xác.

3.2. Huấn luyện (Training):

- Sử dụng model để huấn luyện mô hình trên tập train, đồng thời theo dõi hiệu suất trên tập validation.
- Huấn luyện qua nhiều epochs để đạt được kết quả tốt nhất.

4. Đánh giá mô hình

Sau khi train xong, mô hình được đánh giá trên tập test để kiểm tra hiệu suất thực tế:

- **Accuracy:** Độ chính xác tổng thể.
- **Precision, Recall, F1-score:** Đánh giá chi tiết hiệu suất từng lớp.

Kết quả:

- **Accuracy:** 97.56849315068493%
- **Precision:** 97.65881601921423%
- **Recall:** 97.56849315068493%
- **F1-score:** 97.56768538617257%
- **F1-score:** 97.56768538617257%

5. Lưu và triển khai mô hình

Lưu mô hình:

Sau khi huấn luyện, các trọng số sẽ được in ra và sau đó hardcore vào code để có thể sử dụng lại trong các lần tiếp theo

Triển khai với ESP32-CAM:

- ESP32-CAM chụp ảnh và gửi dữ liệu ảnh tới server Python qua Wi-Fi.
- Server load mô hình và chạy dự đoán:

6. Tóm tắt

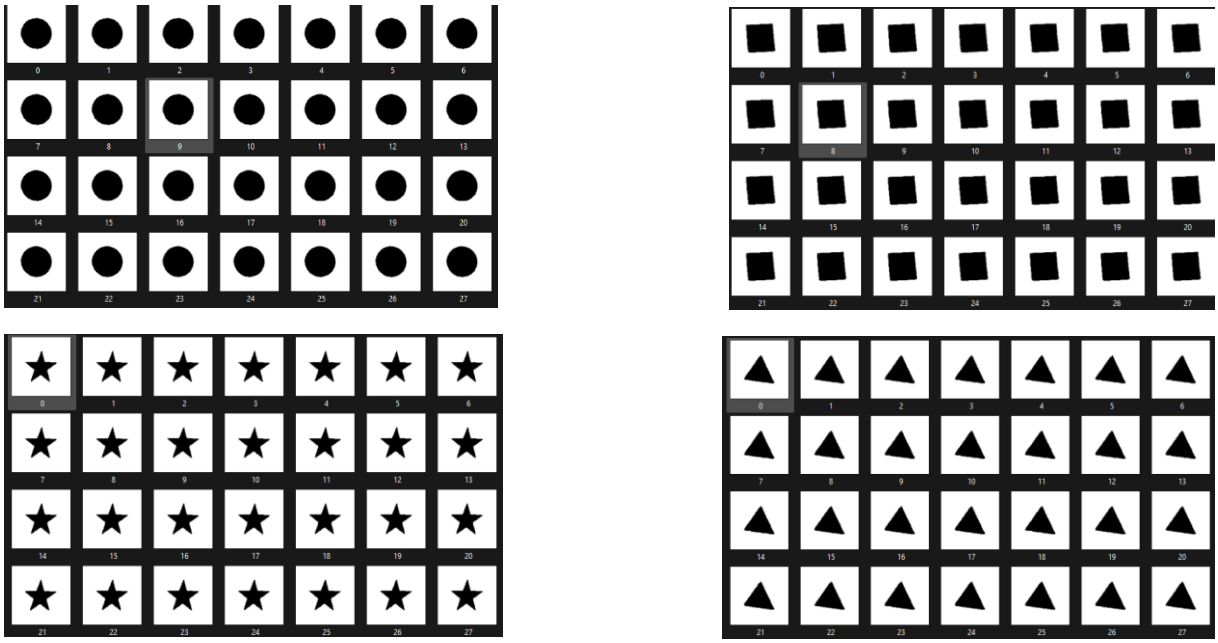
- **Quy trình train model:**
  1. Chuẩn bị dữ liệu từ Kaggle và ảnh ESP32-CAM.
  2. Tiền xử lý dữ liệu (resize, chuẩn hóa).
  3. Xây dựng mô hình CNN gồm 3 lớp convolutional và 2 lớp fully connected.
  4. Huấn luyện mô hình với dữ liệu đã chia (train, validation, test).
  5. Đánh giá mô hình qua các chỉ số: Accuracy, Precision, Recall, F1-score.
- **Kết quả:** Mô hình đạt độ chính xác cao (97.63%) và được triển khai sử dụng với ESP32-CAM để nhận diện các hình khối.

Chương 4: Kết quả đạt được

4.1 Mô tả về tập dữ liệu

Dữ liệu được sử dụng là bộ dữ liệu các hình khối được chia ra thành các tệp theo từng dạng hình. Đây là dữ liệu có sẵn trên internet với khoảng 14950 ảnh đã được gắn nhãn.

Hình 4.1 Chia tập dữ liệu



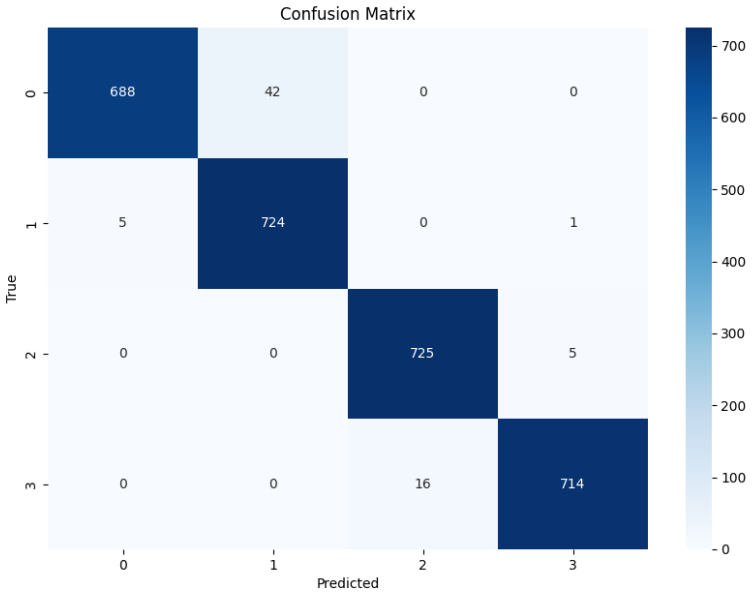
Hình 4.2 Dữ liệu trong các folder

Nhận thấy dữ liệu thu thập được có số lượng khá lớn và tỉ lệ các class khá đồng đều nên nhóm em quyết định không tăng cường dữ liệu.

4.2 Phương pháp đánh giá

- Thay vì dùng các phương pháp khác như accuracy chỉ cho ta biết phần trăm lượng dữ liệu được phân loại đúng mà không chỉ ra được cụ thể mỗi loại được phân loại như thế nào, lớp nào được phân loại đúng nhiều nhất, và dữ liệu thuộc lớp nào thường bị phân loại nhầm vào lớp khác. Để có thể đánh giá được các giá trị này, chúng ta sử dụng một ma trận được gọi là confusion matrix.

- Cấu trúc của Confusion matrix:



Hình 4.3 Confusion matrix

Confusion matrix như trên biểu đồ thể hiện khả năng phân loại của mô hình. Đây là đánh giá chi tiết từng lớp:

1. **Lớp 0 (True label: 0):**

- Dự đoán đúng: 688 mẫu.
- Dự đoán sai:
    - 42 mẫu bị nhầm sang lớp 1.
- Nhận xét: Lớp 0 có hiệu suất nhận diện tốt, nhưng việc nhầm sang lớp 1 chiếm tỷ lệ đáng kể, cần cải thiện.

2. **Lớp 1 (True label: 1):**

- Dự đoán đúng: 724 mẫu.
- Dự đoán sai:
    - 5 mẫu nhầm sang lớp 0.
    - 1 mẫu nhầm sang lớp 3.

- Nhận xét: Mô hình nhận diện lớp 1 tốt, chỉ có một số lỗi nhầm nhỏ.

3. **Lớp 2 (True label: 2):**

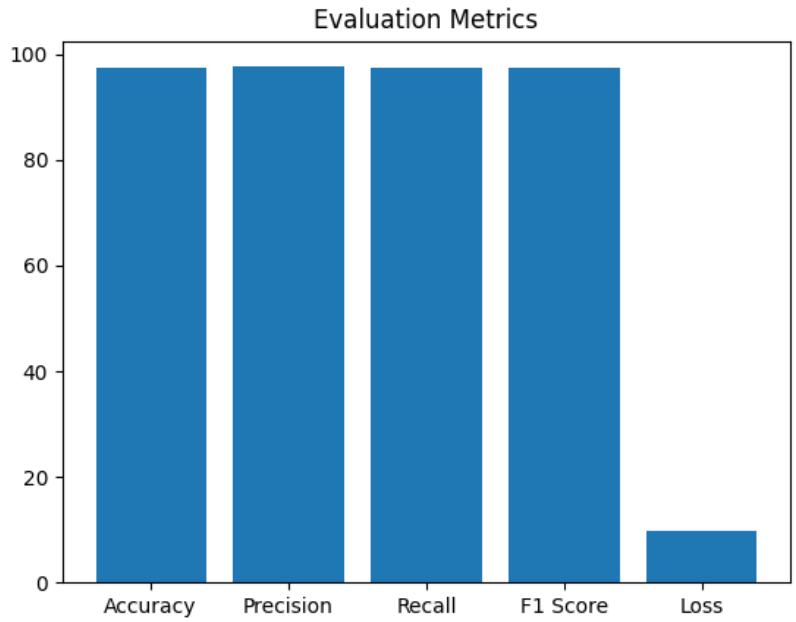
- Dự đoán đúng: 725 mẫu.
- Dự đoán sai:
    - 5 mẫu nhầm sang lớp 3.
- Nhận xét: Hiệu suất lớp 2 rất cao, chỉ có một vài nhầm lẫn với lớp 3.

4. **Lớp 3 (True label: 3):**

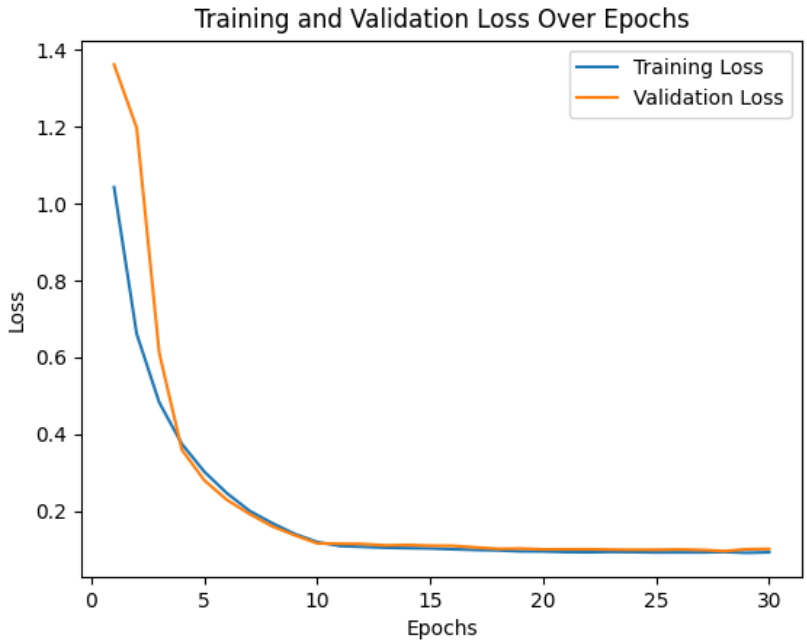
- Dự đoán đúng: 714 mẫu.
- Dự đoán sai:
    - 16 mẫu nhầm sang lớp 2.
- Nhận xét: Lớp 3 có nhầm lẫn đáng kể với lớp 2, cần phân tích nguyên nhân.

Đánh giá tổng quan:

- **Ưu điểm:**
    - Mô hình hoạt động rất tốt trên các lớp, đặc biệt là lớp 2 và lớp 3.
    - Tỷ lệ nhầm lẫn giữa các lớp thấp.
    - Confusion matrix cho thấy rằng các dự đoán chính xác chiếm đa số.
- **Hạn chế:**
    - Một số lỗi nhỏ xảy ra giữa lớp 0 và lớp 1, hoặc lớp 3 và lớp 2. Điều này có thể do đặc điểm dữ liệu chưa đủ phân biệt.



Hình 4.4 Evaluation Metrics



Hình 4.5 Biểu đồ Loss Over Epochs

- Train Box Loss (đường cong màu xanh): Điều này thể hiện tổn thất được tính toán trên tập dữ liệu huấn luyện. Đó là lỗi dự đoán các khung giới hạn của đối tượng trong quá trình huấn luyện.

- Val Box Loss (đường cong màu cam): Điều này thể hiện tổn thất được tính toán trên tập dữ liệu xác thực, được sử dụng để đánh giá hiệu suất của mô hình trên dữ liệu không nhìn thấy.

- Xu hướng giảm: Cả đường cong train val đều cho thấy xu hướng giảm dần, đây là một dấu hiệu tích cực cho thấy mô hình đang học hỏi và cải thiện dự đoán của nó theo thời gian.

- Tổn thất cao ban đầu: Các giá trị loss cao hơn khi bắt đầu (epoch 0), điều này điển hình do mô hình bắt đầu với các trọng số ngẫu nhiên và dần dần học cách đưa ra dự đoán tốt hơn.

Chương 5: Kết luận và hướng phát triển

5.1 Kết luận

1. **Hiệu quả của mô hình:**
    - Mô hình AI nhận diện hình khối đã được triển khai thành công trên ESP32-CAM, cho thấy khả năng tích hợp AI với các vi xử lý nhỏ gọn.
    - Kết quả nhận diện đạt độ chính xác cao với tốc độ xử lý phù hợp trong các ứng dụng thời gian thực.
2. **Ưu điểm:**
    - Chi phí thấp, dễ triển khai và tiếp cận, phù hợp cho các hệ thống nhúng.
    - Tính linh hoạt trong ứng dụng, từ giáo dục đến công nghiệp.
3. **Hạn chế:**
    - Hiệu suất hạn chế khi làm việc với mô hình phức tạp hoặc yêu cầu xử lý lượng lớn dữ liệu.
    - Cần tối ưu hóa thêm để giảm tải bộ nhớ và cải thiện tốc độ xử lý.

5.2 Hướng phát triển

1. **Tối ưu hóa mô hình AI:**
    - Sử dụng các thuật toán nén (quantization, pruning) để giảm kích thước mô hình mà không làm giảm nhiều độ chính xác.
    - Triển khai các mô hình nhẹ hơn, chẳng hạn MobileNet hoặc TinyML.
2. **Tăng khả năng mở rộng:**
    - Mở rộng hệ thống để nhận diện đa dạng các hình khối hoặc đối tượng hơn.
    - Kết hợp thêm cảm biến khác (ví dụ: cảm biến khoảng cách, ánh sáng) để cải thiện độ chính xác.
3. **Cải tiến phần cứng:**
    - Sử dụng vi xử lý mạnh hơn như ESP32-S3, STM32 hoặc các dòng hỗ trợ tăng tốc phần cứng cho AI.
    - Kết hợp với các module xử lý AI như Google Edge TPU hoặc NVIDIA Jetson Nano nếu cần hiệu năng cao.
4. **Ứng dụng thực tế:**
    - Tích hợp vào các hệ thống tự động hóa như robot phân loại, dây chuyền sản xuất.
    - Áp dụng vào giáo dục STEM, hỗ trợ học sinh tìm hiểu về lập trình nhúng và AI.

5. **Cải thiện phần mềm:**
    - Sử dụng các thư viện chuyên dụng như TensorFlow Lite Micro hoặc Edge Impulse để hỗ trợ triển khai nhanh và tối ưu hơn.
    - Tăng tính thân thiện với người dùng qua giao diện hoặc khả năng kết nối từ xa.

Kết hợp những hướng phát triển này sẽ giúp hệ thống AI trên vi xử lý đạt hiệu quả cao hơn và có tính ứng dụng rộng rãi trong tương lai.