HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

CƠ SỞ TẠI THÀNH PHỐ HỒ CHÍ MINH

BÁO CÁO MÔN HỌC

XÂY DỰNG CÁC HỆ THỐNG NHÚNG

ĐỀ TÀI: NHẬN DIỆN BIỂN BÁO GIAO THÔNG

NHÓM 12

Thành viên nhóm:

N19DCCN179 - Nguyễn Quốc Tuấn

N20DCCN023 - Phan Xuân Huynh

N20DCCN029 – Đinh Hồng Kông

N20DCCN037 - Phan Văn Lục

N20DCCN065 – Nguyễn Trần Trọng Tín

N20DCCN067 – Đặng Khắc Toản

Hồ Chí Minh – Tháng 5, năm 2024

Lời cảm ơn

Đầu tiên, chúng em xin gửi lời cảm ơn chân thành đến "Học viện Công nghệ Bưu chính viễn thông" đã đưa môn học "Xây dựng các hệ thống nhúng" vào chương trình giảng dạy.

Đặc biệt, chúng em xin gửi lời cảm ơn sâu sắc đến giáo viên bộ môn - thầy giáo Nguyễn Trọng Kiên đã dạy dỗ, truyền đạt những kiến thức quý báu cho chúng em trong suốt thời gian học tập vừa qua. Trong thời gian tham gia lớp học, chúng em đã có thêm cho mình nhiều kiến thức bổ ích, tinh thần học tập hiệu quả, nghiêm túc. Đây chắc chắn sẽ là những kiến thức quý báu, là hành trang để chúng em có thể vững bước sau này.

Bộ môn "Xây dựng các hệ thống nhúng" là môn học thú vị, vô cùng bổ ích và có tính thực tế cao. Đảm bảo cung cấp đủ kiến thức, gắn liền với nhu cầu thực tiễn của sinh viên. Tuy nhiên, do vốn kiến thức còn nhiều hạn chế và khả năng tiếp thu thực tế còn nhiều bỡ ngỡ. Mặc dù chúng em đã cố gắng hết sức nhưng chắc chắn bài báo cáo khó có thể tránh khỏi những thiếu sót và nhiều chỗ còn chưa chính xác, kính mong thầy xem xét và góp ý để bài báo cáo của nhóm 12 chúng em được hoàn thiện hơn.

Chúng em xin chân thành cảm ơn!

MỤC LỤC

1. GIỚI THIỆU ĐỀ TÀI

1.1.Lí do chọn đề tài

Trong bối cảnh cuộc sống ngày nay, sự phát triển nhanh chóng của công nghệ đặt ra nhiều thách thức và cơ hội mới. Trong lĩnh vực giao thông, việc áp dụng công nghệ để tăng cường an toàn và hiệu suất đang trở thành một ưu tiên hàng đầu. Một trong những thách thức quan trọng là nhận diện biển báo giao thông, đặc biệt là khi chúng ta chuyển từ lái xe thủ công sang xe tự lái và hệ thống hỗ trợ lái xe thông minh. Điều này đòi hỏi sự phát triển mạnh mẽ trong lĩnh vực xử lý ảnh và học máy, đặc biệt là sử dụng các mô hình như Convolutional Neural Network (CNN) để nhận diện và hiểu biển báo giao thông.

1.2.Mục tiêu

Chúng em đã chọn đề tài này với mục tiêu chính là nghiên cứu và phát triển một hệ thống nhận diện biển báo giao thông sử dụng mô hình CNN. Cụ thể, chúng em đặt ra các mục tiêu sau:

- Nắm vững kiến thức cơ bản về lập trình nhúng và học máy.
- Phát triển một mô hình CNN có khả năng nhận diện chính xác và nhanh chóng các biển báo giao thông trong các điều kiện khác nhau.
- Thông báo cho người dùng về biển báo.

Chúng em tin rằng nếu đạt được những mục tiêu này, đề tài của chúng tôi sẽ đóng góp một phần quan trọng vào việc phát triển các hệ thống an toàn giao thông thông minh trong tương lai.

1.3.Các chức năng

- Lấy hình ảnh từ camera
- Sử dụng mô hình học máy để phát triển và nhận dạng biển báo giao thông từ hình ảnh
- Thông báo kết quả đến ứng dụng di động

1.4.Công nghệ sử dụng

- Ngôn ngữ: python, c/c++, java
- Môi trường phát triển: pycharm, arduino, android studio
- Mô hình học máy: Convolutional Neural Networks (CNN)

1.5. Linh kiện sử dụng:

- Esp32Cam

2. CƠ SỞ LÝ THUYẾT

2.1. GIỚI THIỆU VỀ FREERTOS

2.1.1. FreeRTOS là gì ?

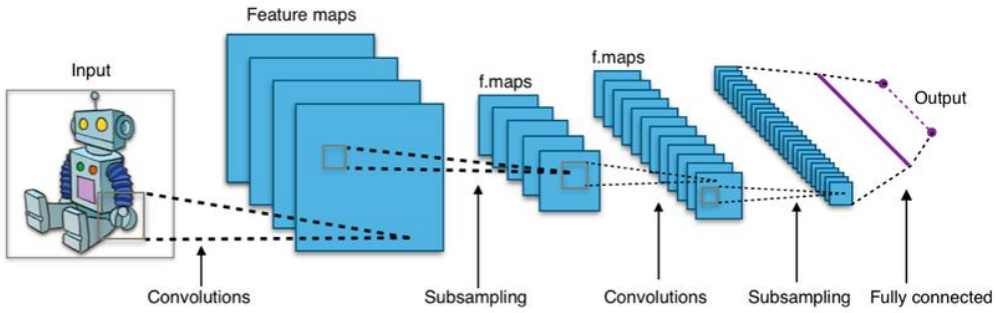
FreeRTOS là một hệ điều hành nhúng thời gian thực (Real Time Operating System) mã nguồn mở được phát triển bởi Real Time Engineers Ltd, sáng lập và sở hữu bởi Richard Barry. FreeRTOS được thiết kế phù hợp cho nhiều hệ nhúng nhỏ gọn vì nó chỉ triển khai rất ít các chức năng như: cơ chế quản lý bộ nhớ và tác vụ cơ bản, các hàm API quan trọng cho cơ chế đồng bộ. Nó không cung cấp sẵn các giao tiếp mạng, drivers, hay hệ thống quản lý tệp (file system) như những hệ điều hành nhúng cao cấp khác. Tuy vậy, FreeRTOS có nhiều ưu điểm, hỗ trợ nhiều kiến trúc vi điều khiển khác nhau, kích thước nhỏ gọn (4.3 Kbytes sau khi biên dịch trên Arduino), được viết bằng ngôn ngữ C và có thể sử dụng, phát triển với nhiều trình biên dịch C khác nhau (GCC, OpenWatcom, Keil, IAR, Eclipse, …), cho phép không giới hạn các tác vụ chạy đồng thời, không hạn chế quyền ưu tiên thực thi, khả năng khai thác phần cứng. Ngoài ra, nó cũng cho phép triển khai các cơ chế điều độ giữa các tiến trình như: queues, counting semaphore, mutexes [1].

2.1.2. Ưu điểm của FreeRTOS

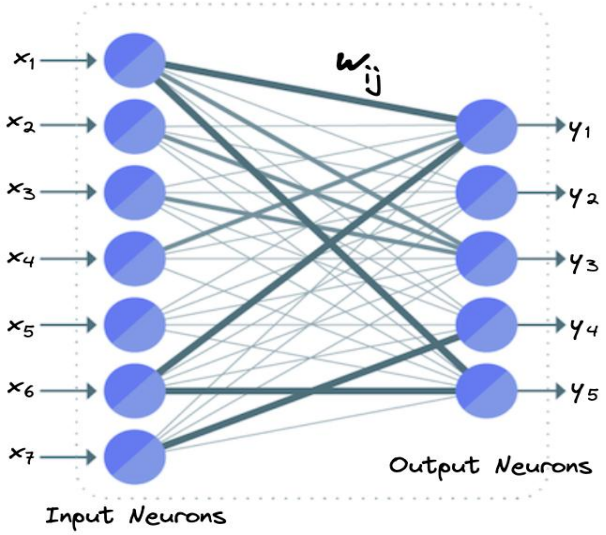
Ưu điểm lớn của RTOS là xử lý nhanh chóng vì thế nó sẽ dành cho các thiết bị đòi hỏi khả năng xử lý có độ trễ thấp nhất có thể. Lợi ích nó đem lại bao gồm đa nhiệm tốt, ưu tiên các nhiệm vụ và quản lý chia sẻ các tài nguyên. Ngoài ra nó cũng không đòi hỏi nhiều về tài nguyên hay bộ nhớ RAM quá lớn.

2.2. GIỚI THIỆU MÔ HÌNH CNN (CONVOLUTIONAL NEURAL NETWORKS)

CNN là một trong những mô hình học sâu tiên tiến giúp cho chúng ta xây dựng được những hệ thống xử lý thông minh, cho kết quả có độ chính xác cao. Mô hình CNN như hình 1 có các layer liên kết được với nhau thông qua cơ chế tích chập (convolution). Layer tiếp theo là kết quả tích chập từ layer trước đó. Nhờ vậy, ta có được các kết nối cục bộ. Tính kết hợp cục bộ cho ta các cấp độ biểu diễn thông tin từ mức độ thấp đến mức độ cao và trừu tượng hơn thông qua tích chập (convolution) từ các bộ lọc [2].



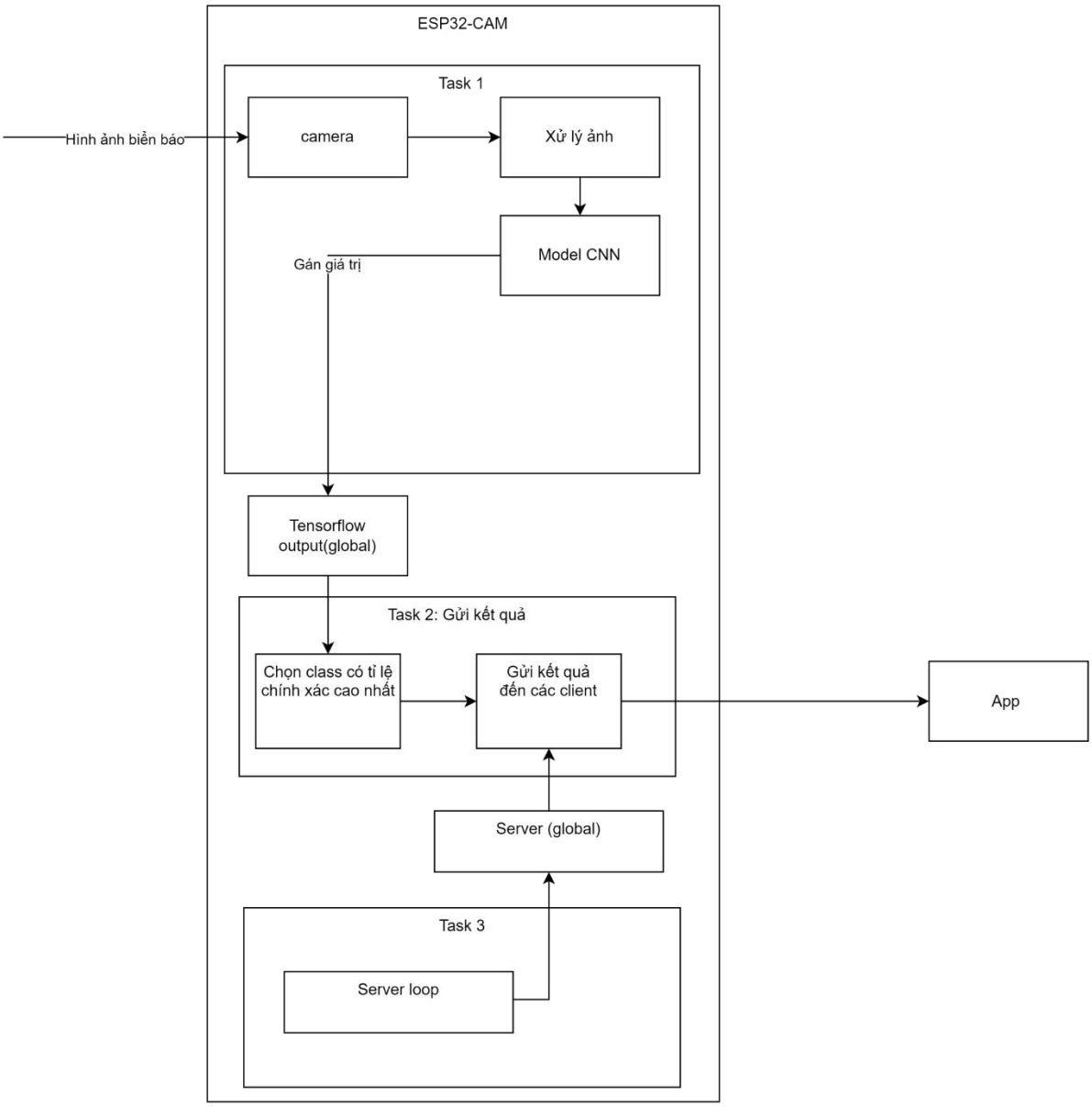
- Tầng Convolution sử dụng phép tích chập để xử lý dữ liệu bằng cách trượt cửa sổ trượt (slide windows) có kích thước cố định (còn gọi là kernel) trên ma trận dữ liệu đầu vào để thu được kết quả đã được tinh chỉnh. Trong khi đó, tầng Pooling tổng hợp các vector kết quả của tầng Convolution và giữ lại những vector quan trọng nhất.
- Đầu ra cuối cùng của Lớp Pooling cuối cùng đóng vai trò là đầu vào của Lớp Fully Connected trong CNN. Một hoặc nhiều lớp có thể có trong này. Kết nối đầy đủ cho thấy, mọi nút trong lớp ban đầu được kết nối với mọi nút của lớp tiếp theo như minh họa dưới đây



-Một số kiến trúc mô hình CNN nổi tiếng: Lenet, AlexNet, VGG,…

3. TRIỂN KHAI

3.1.SƠ ĐỒ HOẠT ĐỘNG



Mô tả hoạt động:

- Hình ảnh được đưa vào trước camera của esp32cam.
- Trong esp32cam sử dụng FreeRTOS cài đặt sẵn 3 task:
    - Task 1 có nhiệm vụ khởi động lại camera, tiền xử lý hình ảnh, và đưa vào model CNN sau đó gán giá trị dự đoán ra biến output. được thực hiện lặp đi lặp lại.

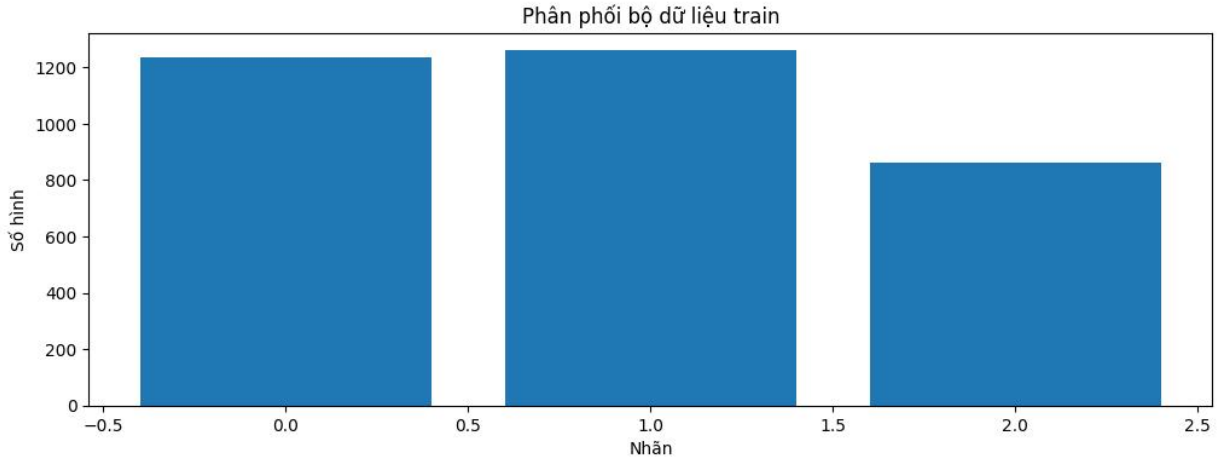
- Task 2: có nhiệm vụ chọn ra class có tỉ lệ chính xác cao nhất nếu kết quả lớn hơn giá trị mặc định thì sẽ gửi kết quả về các ứng dụng thông qua biến server. Quá trình này diễn ra liên tục
- Task 3: có nhiệm vụ chạy server để hỗ trợ dịch vụ gửi kết quả ở task 2. Quá trình này diễn ra liên tục.

- Ứng dụng di động khi nhận kết quả sẽ hiển thị ảnh biển báo và phát thông báo ra loa điện thoại.

3.2. TẠO MODEL CNN

3.2.1. Chuẩn bị dữ liệu

- Số nhãn sử dụng: 3
- Biểu đồ hiển thị phân phối số nhãn trong từng ảnh:



- Chia tập dữ liệu: train 80%, test 20%, sau đó từ tập train lại chia : train 80%, validation 20%

3.2.2. Tiền xử lí dữ liệu:

```
def grayscale(img):
    img = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    return img
def equalize(img):
    img =cv2.equalizeHist(img)
    return img
def preprocessing(img):
    img = grayscale(img)
    img = equalize(img)
    img = img/255
    return img
```

- Hàm **grayscale** : Sử dụng hàm "**cv2.cvtColor**" của OpenCV để chuyển đổi ảnh từ không gian màu BGR (Blue, Green, Red) sang không gian màu xám.
- Hàm **equalize:** Sử dụng hàm cv2.equalizeHist của OpenCV để cân bằng histogram của ảnh. Cân bằng histogram giúp cải thiện độ tương phản của ảnh, làm cho các chi tiết trong ảnh rõ ràng hơn
- Hàm **preprocessing:** gọi hàm grayscale và equalize, sau đó sẽ chuẩn hóa ảnh bằng cách chia giá trị pixel cho 255, chuyển đổi giá trị pixel từ khoảng 0-255 thành khoảng 0-1. Điều này giúp tăng hiệu quả của mô hình học sâu khi làm việc với dữ liệu ảnh.

```
dataGen= ImageDataGenerator(width_shift_range=0.1,
                            height_shift_range=0.1,
                            zoom_range=0.2,
                            shear_range=0.1,
                            rotation_range=10)
```

Tạo các phép biến đổi để áp dụng ngẫu nhiên cho mỗi ảnh trong quá trình huấn luyện, tạo ra các biến thể khác nhau của ảnh gốc và giúp mô hình học sâu trở nên linh hoạt hơn, có khả năng chống lại hiện tượng overfitting và cải thiện khả năng tổng quát hóa của mô hình trên dữ liệu thực tế.

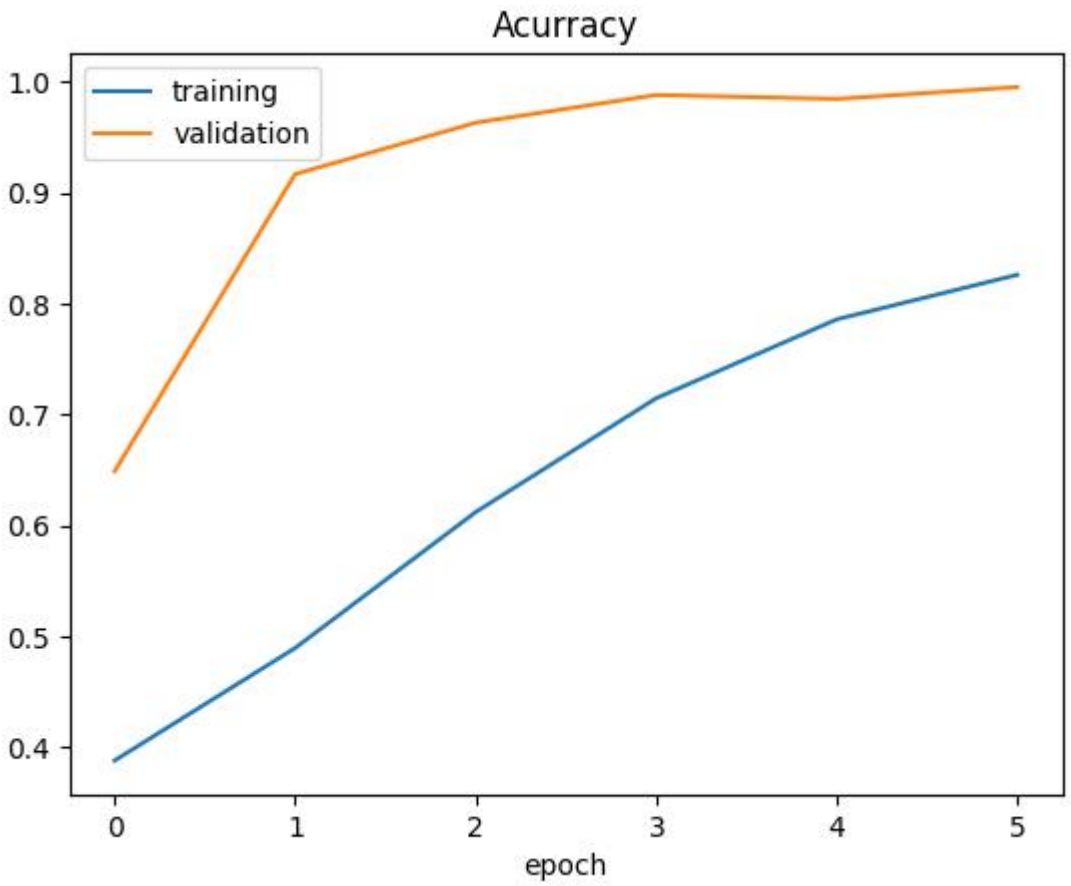
3.2.3. Huấn luyện mô hình CNN

```
model = tf.keras.Sequential()
model.add((Conv2D(6,(3,3),input_shape=(imageDimesions[0],imageDimesions[1],1),activation='relu')))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add((Conv2D(6, (3,3), activation='relu')))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.5))

model.add(Flatten())
model.add(Dense(15,activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(noOfClasses,activation='softmax'))
model.compile(Adam(lr=0.001),loss='categorical_crossentropy',metrics=['accuracy'])
```

Mô hình CNN bao gồm 9 lớp:

- 2 lớp Convolutional để trích xuất đặc trưng từ ảnh.
- 2 lớp MaxPooling để giảm kích thước ảnh nhưng vẫn giữ các đặc trưng quan trọng.
- 2 lớp Dropout để ngăn ngừa overfitting bằng cách lọc số nơ-ron.
- 1 lớp Flatten để chuyển đổi đầu ra của lớp tích chập thành 1 vector phẳng để có thể cho các lớp Fully connected.
- 2 lớp Fully connected với một lớp đầu ra sử dụng softmax để phân loại ảnh.

Biểu đồ hiển thị độ chính xác của mô hình qua các lần huấn luyện:



3.3. TÍCH HỢP MODEL CNN VÀO MẠCH ESP32CAM

Model CNN sau khi được huấn luyện sẽ được chuyển đổi sang định dạng TensorFlow Lite, giúp mô hình có thể chạy trên các thiết bị nhúng như ESP32

```
# Convert Keras model to a tflite model
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()
```

Code chuyển đổi định dạng model

Sau đó sẽ chuyển đổi model từ định dạng TensorFlow Lite thành một mảng C. Đây là cách để nhúng dữ liệu mô hình TensorFlow Lite vào mã C.

```
def hex_to_c_array(hex_data, var_name):
  c_str = ''
  c_str += '#ifndef ' + var_name.upper() + '_H\n'
  c_str += '#define ' + var_name.upper() + '_H\n\n'
  c_str += '\nunsigned int ' + var_name + '_len = ' + str(len(hex_data)) + ';\n'
  c_str += 'unsigned char ' + var_name + '[] = {'
  hex_array = []
  for i, val in enumerate(hex_data) :
    hex_str = format(val, '#04x')
    if (i + 1) < len(hex_data):
      hex_str += ','
    if (i + 1) % 12 == 0:
      hex_str += '\n '
    hex_array.append(hex_str)
  c_str += '\n ' + format(' '.join(hex_array)) + '\n};\n\n'
  c_str += '#endif //' + var_name.upper() + '_H'
  return c_str
```

Code chuyển đổi định dạng TensorFlow Lite thành một mảng C

Nạp mô hình TensorFlow Lite vào ESP32-CAM:

```
void initTensorFlowLite() {
  static tflite::MicroErrorReporter micro_error_reporter;
  error_reporter = &micro_error_reporter;
  model = tflite::GetModel(bbgt_model);
  if (model->version() != TFLITE_SCHEMA_VERSION) {
    TF_LITE_REPORT_ERROR(error_reporter,
                         "Model provided is schema version %d not equal "
                         "to supported version %d.",
                         model->version(), TFLITE_SCHEMA_VERSION);
    return;
  }
  static tflite::AllOpsResolver resolver;
  static tflite::MicroInterpreter static_interpreter(
      model, resolver, tensor_arena, kTensorArenaSize, error_reporter);
  interpreter = &static_interpreter;

  TfLiteStatus allocate_status = interpreter->AllocateTensors();
  if (allocate_status != kTfLiteOk) {
    TF_LITE_REPORT_ERROR(error_reporter, "AllocateTensors() failed");
    return;
  }
  model_input = interpreter->input(0);
  model_output = interpreter->output(0);
}
```

3.4.XỬ LÍ TÍN HIỆU HÌNH ẢNH

```
void resize_image_to_32x32(uint8_t* input, uint8_t* output, int inputWidth, int inputHeight) {
  float scaleWidth = inputWidth / (float)kImageWidth;
  float scaleHeight = inputHeight / (float)kImageHeight;

  for (int y = 0; y < kImageHeight; y++) {
    for (int x = 0; x < kImageWidth; x++) {
      int srcX = (int)(x * scaleWidth);
      int srcY = (int)(y * scaleHeight);
      srcX = min(srcX, inputWidth - 1);
      srcY = min(srcY, inputHeight - 1);
      int inputIndex = (srcY * inputWidth) + srcX;
      int outputIndex = (y * kImageWidth) + x;
      output[outputIndex] = input[inputIndex];
    }
  }
}
```

Hàm chuyển đổi kích thước ảnh mà ESP32Cam nhận được ,sử dụng phương pháp lấy mẫu gần nhất để thay đổi kích thước ảnh đầu vào thành kích thước 32x32. Bằng cách tính toán lại tọa độ pixel từ ảnh đầu ra về ảnh đầu vào, hàm này đảm bảo ảnh đầu ra có kích thước như mong muốn mà không làm biến dạng dữ liệu ảnh.

```
void preprocess_image(uint8_t* input, float* output, int width, int height) {
    uint8_t grayscale_img[width * height];
    uint8_t equalized_img[width * height];

    // Convert image to grayscale
    for (int i = 0; i < width * height; i++) {
        grayscale_img[i] = input[i];
    }

    // Equalize histogram
    int hist[256] = {0};
    for (int i = 0; i < width * height; i++) {
        hist[grayscale_img[i]]++;
    }
    int cumulative = 0;
    int minCumulative = width * height;
    for (int i = 0; i < 256; i++) {
        cumulative += hist[i];
        if (cumulative > 0 && cumulative < minCumulative) {
            minCumulative = cumulative;
        }
    }
    float scale = 255.0 / (width * height - minCumulative);
    int sum = 0;
    for (int i = 0; i < 256; i++) {
        sum += hist[i];
        hist[i] = round((sum - minCumulative) * scale);
    }
    for (int i = 0; i < width * height; i++) {
        equalized_img[i] = hist[grayscale_img[i]];
    }

    // Normalize the image and copy to output
    for (int i = 0; i < width * height; i++) {
        output[i] = equalized_img[i] / 255.0f;
    }
}
```

Hàm tiền xử lí ảnh nhận được từ ESP32Cam để đưa vào model CNN sẽ thực hiện ba bước : chuyển ảnh sang dạng xám, cân bằng histogram và chuẩn hóa giá trị pixel. Quá trình này giúp chuẩn hóa ảnh đầu vào, cải thiện hiệu quả của mô hình bằng cách đảm bảo rằng dữ liệu đầu vào nằm trong một phạm vi giá trị nhất định và tăng cường các đặc điểm của ảnh.

3.5. THÔNG BÁO KẾT QUẢ ĐẾN ỨNG DỤNG DI ĐỘNG

Hệ thống này sử dụng ứng dụng Android kết nối với ESP32 Cam qua socket để nhận kết quả dự đoán biển báo.

ESP32 Cam sẽ đóng vai trò là máy chủ và phát wifi. Ứng dụng Android sẽ đóng vai trò là máy khách, lắng nghe dữ liệu từ ESP32 Cam gửi đến và hiển thị lên màn hình dựa trên dữ liệu nhận được.

Cấu trúc hệ thống:

- ESP32 Cam: Phát Wi-Fi. Lập trình ESP32 Cam tạo socket server, gửi dữ liệu tới các máy khách đã kết nối.
- Ứng dụng Android: Tạo kết nối socket client để nhận dữ liệu từ ESP32 Cam. Hiển thị thông báo kết quả lên giao diện người dùng.

Kết quả nhận diện được hiển thị trên ứng dụng di động

4. KẾT LUẬN VÀ KIẾN NGHỊ

4.1.Kết quả đạt được

Qua quá trình nghiên cứu và thực hiện đồ án, chúng em đã đạt được những kết quả quan trọng trong lĩnh vực nhận diện biển báo giao thông bằng mô hình CNN và esp32cam. Mô hình không chỉ đạt được độ chính xác cao mà còn có tốc độ xử lý khá ổn định trong nhiều điều kiện khác nhau. Chúng em đã kiểm tra mô hình trên một bộ dữ liệu đa dạng với nhiều biển báo giao thông và thu được kết quả khả quan.

4.2.Hạn chế đề tài

- ESP32-CAM có bộ vi xử lý và bộ nhớ hạn chế, không đủ mạnh để thực hiện các mô hình deep learning phức tạp.
- Với tốc độ xung nhịp tối đa khoảng 240 MHz, ESP32-CAM không đủ khả năng xử lý các tác vụ tính toán nặng một cách nhanh chóng. Điều này dẫn đến thời gian phản hồi chậm khi nhận diện biển báo.
- ESP32-CAM có bộ nhớ flash tối đa 4MB (SPI Flash), không đủ để lưu trữ lượng dữ liệu đào tạo lớn, dẫn đến nhận diện với độ chính xác thấp.
- Camera của ESP32-CAM có độ phân giải và chất lượng hình ảnh hạn chế , ảnh hưởng đến chất lượng đầu vào cho các mô hình , từ đó ảnh hưởng đến độ chính xác của kết quả phân tích.
- Các thư viện deep learning phổ biến như TensorFlow Lite, PyTorch không được hỗ trợ đầy đủ cho ESP32-CAM. Điều này làm cho việc triển khai mô hình học máy trở nên phức tạp và khó khăn.

4.3.Hướng phát triển và kết luận

Mặc dù đồ án đã đạt được những kết quả tích cực, nhưng vẫn còn nhiều cơ hội để cải thiện và mở rộng nghiên cứu trong tương lai. Dưới đây là một số hướng phát triển tiếp theo:

- Mở rộng bộ dữ liệu: Nghiên cứu có thể được mở rộng bằng cách thu thập thêm dữ liệu từ nhiều nguồn và điều kiện đa dạng, giúp mô hình trở nên mạnh mẽ hơn và phát triển khả năng tổng quát hóa.
- Tối ưu hoá mô hình: Tiếp tục tối ưu hóa cấu trúc mô hình và thử nghiệm với các phương pháp tăng cường dữ liệu để tăng cường khả năng nhận diện và giảm overfitting.
- Thí nghiệm trên thực tế: Mở rộng phạm vi thí nghiệm bằng cách triển khai mô hình trên môi trường thực tế, đặt ra những thách thức mới như biến đổi ánh sáng, thời tiết, và góc chụp.
- Kết hợp với công nghệ khác: Nghiên cứu tích hợp mô hình nhận diện vào các hệ thống hỗ trợ lái xe tự động và các ứng dụng thực tế khác, tạo ra các giải pháp toàn diện hơn.

Kết luận, đồ án không chỉ mang lại những hiểu biết về lĩnh vực xử lý ảnh và học máy mà còn làm nền tảng cho các nghiên cứu và ứng dụng tiếp theo trong việc cải thiện an toàn giao thông và phát triển xe tự lái. Chúng em hy vọng rằng kết quả từ đề tài này sẽ làm đóng góp tích cực vào sự phát triển của ngành công nghiệp ô tô thông minh.

TÀI LIỆU THAM KHẢO

[1] Đ. H. Toàn, "Chạy đa nhiệm trên Arduino với FreeRTOS," [Online]. Available: http://arduino.vn/bai-viet/1673-chay-da-nhiem-tren-arduino-voi-freertos.

[2] P. T. Tài, "https://topdev.vn/blog/thuat-toan-cnn-convolutional-neural-network/," [Online]. Available: https://topdev.vn/blog/thuat-toan-cnn-convolutional-neural-network/.