ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO

HỌC PHẦN: NHẬP MÔN LẬP TRÌNH ĐIỀU KHIỂN THIẾT BỊ THÔNG MINH

Chủ đề: Hệ thống phát hiện người và vị trí tương đối trên ESP32-CAM AI THINKER

Giáo viên hướng dẫn:

Thầy Nguyễn Đức Hoàng Hạ

Sinh viên thực hiện:

Nguyễn Duy Quang - 20120360

01/2024

Mục lục

Danh sách hình vẽ

Danh sách mã nguồn

1 Giới thiệu

1.1 Bối cảnh

Lập trình thiết bị thông minh (Smart Device Programming, còn được gọi là Artificial Intelligence of Things - AIoT) là một trong những hiện tượng mới nổi gần đây. Là sự kết hợp giữa trí tuệ nhân tạo (AI) và lập trình kết nói vạn vật (IoT), là một trong những lĩnh vực có khả năng rất mạnh trong việc cải thiện chất lượng cuộc sống con người, đặc biệt là trong việc tối ưu hóa các thao tác, hoạt động thường thấy. Ở mức độ cao hơn, việc thiết kế ra các thiết bị thông minh không chỉ giúp giảm chi phí thao tác của tác vụ mà còn đảm bảo tính bảo mật của thiết bị.

Ngược lại, phát hiện con người đã là một bài toán cổ điển trong thị giác máy tính, đã có rất nhiều phương pháp từ truyền thống đến SOTA. Tuy nhiên, ứng dụng cụ thể ở mức độ vi mô, hay trong phạm vi bài này là các thiết bị nhúng, thì cũng chỉ mới ở mức độ sơ khai. Đặc biệt, với một vi điều khiển có giả phổ thông (200.000 VNĐ) là ESP32-CAM AI THINKER được sử dụng trong bài này thì càng khó nữa, bởi vì số lượng mô hình có thể áp dụng được là rất nhỏ với bộ nhớ hạn chế của chúng (520KB SRAM +4M PSRAM). Theo thông tin em tìm được hiện giờ, ESP32-CAM AI THINKER vẫn chưa thực hiện được các tác vụ trả về kết quả là bounding box.

Ngoài ra, phát triển từ đó, dự án sẽ cố gắng tiến thêm một bước nữa: Thay vì chỉ xác định là máy ảnh có phát hiện người hay không thì máy ảnh sẽ đồng thời xác định xem người đó đang ở vị trí tương đối như thế nào trong ảnh (trái, phải hoặc chưa xác định). Mục tiêu của em là nếu thiết bị này hoàn chỉnh, nó có thể được áp dụng trong các tác vụ quan sát con người như mở cửa tự động, đếm người ra vào...

1.2 Mục tiêu của dự án

Dựa trên ý tưởng từ mô hình phát hiện người từ ví dụ minh họa của Arduino TensorFlowLite ESP32, mục tiêu mà dự án hướng tới là:

- Thiết kế một hệ thống có thể thỏa mãn yêu cầu (phát hiện người cùng vị trí tương đối).
- Triển khai lên thiết bị nhúng.

2 Bộ dữ liệu

Bộ dữ liệu được dựa trên Visual Wake Words. Visual Wake Words là bộ dữ liệu được làm lại từ COCO, với mục tiêu là tạo ra bộ dữ liệu cho việc phát hiện rằng trong hình ảnh có con người hay không. Dựa trên điều đó, em xin được đề xuất một bộ dữ liệu cũng dựa từ COCO.

2.1 Quy ước

Đối tượng chính của bộ dữ liệu này là con người. Theo đó, một đối tượng trong một ảnh được gọi là con người nếu đối tượng đó được xem là con người trong chú thích của bộ dữ liệu COCO.

Ngoài ra, bởi vì ta đang hoạt động trên các thiết bị nhúng, với điều kiện và chất lượng thấp hơn bình thường, nên ta sẽ ràng buộc thêm một điều kiện nữa: Nếu diện tích của đối tượng lớn hơn 0.5% so với diện tích bức ảnh thì đối tượng sẽ được xem là con người.

Cùng với đó, bởi vì tập dữ liệu được trích từ một tập dữ liệu đa dạng như COCO, ta sẽ phải tập trung vào một nhóm đối tượng mà thôi. Vậy nên với việc chọn người để đánh giá vị trí, ta sẽ chọn người có cống hiến nhiều nhất với bức ảnh (người có diện tích chiếm >= 70% tổng diện tích người trong ảnh).

Cuối cùng, để phân biệt giữa các vị trí với nhau đồng thời phân biệt khi nào người đó nằm gọn trong ảnh, ta sẽ xem người đó là nằm ở bên trái ảnh nếu cả bounding box của người đó nằm gọn trong 1/3 bên trái ảnh ($most\_right(img) <= \frac{image\_width}{3.0}$). Tương tự, người là nằm bên phải nếu cả bounding box của người đó nằm gọn trong 1/3 bên phải ảnh ($most\_left(img) >= \frac{2*image\_width}{3.0}$).

2.2 Bộ dữ liệu

Từ các quy ước trên, ta có tập dữ liệu mà trong đó:

- Nếu bức ảnh không có đối tượng người hoặc đối tượng người quá nhỏ (nhỏ hơn 0.5%) thì bức ảnh sẽ được xem là không có người.
- Nếu bức ảnh có người, có một người có đóng góp đáng kể trong ảnh, và người đó nằm ở bên trái ảnh thì bức ảnh sẽ được xem là có người bên trái.
- Nếu bức ảnh có người, có một người có đóng góp đáng kể trong ảnh, và người đó nằm ở bên phải ảnh thì bức ảnh sẽ được xem là có người bên phải.
- Các trường hợp có người còn lại sẽ được đánh nhãn là không rõ ràng (hoặc cả hai).

2.3 Tóm tắt bộ dữ liệu

Việc lọc giai đoạn ban đầu cho kết quả rằng bộ dữ liệu bị lệch rất nhiều: Bộ huấn luyện của bộ dữ liệu có các bức ảnh gắn nhãn trái và phải chỉ dưới 2000 bức ảnh với mỗi nhãn, trong đó không rõ ràng và không có người đều ở mức 3*1e5 đến 4*1e5 mỗi nhãn. Để bộ dữ liệu cân bằng hơn, ta sẽ chọn ngẫu nhiên 2000 bức từ mỗi tập.

Thực hiện tương tự với bộ kiểm định, ta được bộ dữ liệu như sau:

- Huấn luyện:
  - Trái: 1757.
  - Phải: 1779.
  - Không rõ ràng: 2000.
  - Không có người: 2000.
- Kiểm định:
  - Trái: 852.
  - Phải: 856.
  - Không rõ ràng: 1000.
  - Không có người: 1000.

3 Mô hình

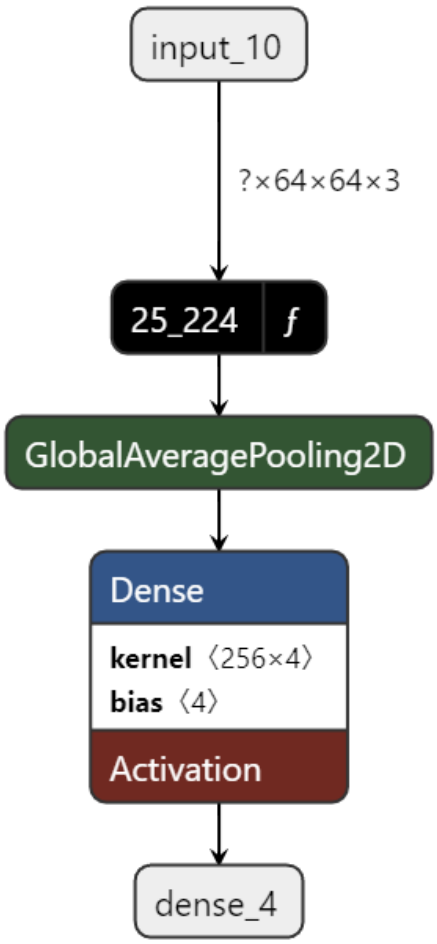
3.1 Mô hình

Để vừa đảm bảo mô hình đủ nhỏ để thêm vào vi điều khiển nhưng đồng thời đơn giản để thực hiện, ta sẽ thiết kế mô hình dựa trên mô hình MobileNet.

MobileNet là một mô hình nhẹ, được thiết kế bởi Google dành cho các thiết bị di động và nhúng, nhưng vẫn cho ra kết quả chấp nhận được. Nhờ khả năng tiêu hao ít, MobileNet là một trong những mô hình phổ biến cho các thiết bị nhỏ và nhúng. Nhìn chung, bài toán phát hiện đối tượng hay phát hiện con người của COCO thì mô hình MobileNet đã làm rất tốt nên ta chỉ cần tái chế lại, sử dụng mô hình pre-train của họ.

Tuy nhiên, để có thể sử dụng mô hình cho bài toán này, ta sẽ cần phải đưa output của mô hình thành output của bài toán. Để làm điều đó, ta thêm một lớp Pooling vào rồi một lớp Fully-connected để đưa về kết quả là 4 lớp.

Hình dưới là minh họa cho mô hình, với hàm f là mô hình MobileNet.



Hình 1: Mô hình

3.2 Tối ưu mô hình

Tại lúc này, mô hình của chúng ta có dung lượng xấp xỉ $1MB$, chưa thể đưa vào vi điều khiển ESP32-CAM AI THINKER. Để làm được điều đó, ta sẽ tối ưu mô hình bằng TFLite cùng với quantization. Quantization (tạm gọi lượng tử hóa) là kỹ thuật mapping vùng dữ liệu xử lý của mô hình thành vùng dữ liệu nhỏ hơn, từ đó làm giảm dung lượng, bộ nhớ... của mô hình.

Trong dự án này, mô hình sẽ được thực hiện tối ưu hóa bằng tflite cùng với quantization thành mô hình int8 thay vì float. Mô hình sau đó là:

- Mô hình gốc: 1030KB.
- Mô hình tối ưu bằng TFLite: 845KB.
- Mô hình tối ưu bằng TFLITE và quantization thành mô hình int8: 301KB.

4 Cài đặt

4.1 Mạch vi điều khiển

4.2 Mã nguồn bộ dữ liệu

Bộ dữ liệu được lấy từ https://www.kaggle.com/datasets/jeffaudi/coco-2014-dataset-for-yolov3. Code xử lý dữ liệu được tham khảo từ https://gist.github.com/AyishaR/ccad108c6c5c4838833766ab63ad8bf7.

Ứng dụng sử dụng các thư viện sau để chạy nên hãy đảm bảo ứng dụng đã được cài đặt các thư viện trước khi chạy.

Mã nguồn 1: Chú thích con người

```
def check_human_annotation(img_name, threshold=0.005):
    anns = get_annotations(img_name)
    width, height = get_size(img_name)
    human_anns = anns[(anns['category_id'] == 1) & (anns['area'] > threshold *
        width * height)] # Human and bigger than threshold
    if len(human_anns) == 0:
        return False, 'none'

    main_human = human_anns['area'].argmax()

    if len(human_anns) > 1 and human_anns['area'].values[main_human] < 0.7 *
        human_anns['area'].sum():
        return True, 'none' # Check the main human

    bbox = human_anns['bbox'].values[0]
    left_x, right_x = bbox[0], bbox[0] + bbox[2]

    if right_x < 0.33 * width: # Position
        return True, 'left'
    elif left_x > 0.66 * width:
        return True, 'right'
    else:
        return True, 'none'
```

Và hàm chọn ngẫu nhiên ảnh từ thư mục ảnh sau khi phân loại.

Mã nguồn 2: Chọn ngẫu nhiên ảnh

```
import random
```

```
def select_1k(folder):
    os.mkdir(folder + '_1k')
    files = os.listdir(folder)
    random.shuffle(files)
    files = files[:1000]
    for file in files:
        shutil.move(os.path.join(folder, file), os.path.join(folder + '_1k',
            file))

select_1k('vwwd/both')
select_1k('vwwd/no')
```

Toàn bộ mã nguồn có thể kiểm tra tại thư mục repo.

4.3 Mã nguồn mô hình

Ở khâu thiết kế bộ dữ liệu, ta có thể chọn kích cỡ ảnh khi đưa vào mô hình.

Mã nguồn 3: Thiết kế bộ dữ liệu

```
batch_size = 32

print("Training set")
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
  train_dir,
  image_size=(64, 64),
  validation_split=0.2,
  subset="training",
  seed=123,
  batch_size=batch_size)

print("Validation set")
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
  val_dir,
  image_size=(64, 64),
  validation_split=0.2,
  subset="validation",
  seed=123,
  batch_size=batch_size)
```

Việc thiết kế mô hình bao gồm lấy mô hình MobileNet đã được pretrain, đóng băng nó lại và thêm các lớp cần thiết. Bởi vì ta chỉ cần huấn luyện một lượng nhỏ tham số, ta thêm hàm callback để ngưng huấn luyện khi mô hình có cảm giác không thể cải thiện được nữa, đồng thời lưu lại mô hình tốt nhất.

Mã nguồn 4: Thiết kế mô hình

```
base_model = tf.keras.applications.MobileNet(input_shape=input_shape, weights='
    imagenet', alpha=0.25, include_top=False)

base_model.trainable = False

inputs = tf.keras.Input(shape=input_shape)

x = base_model(inputs, training=False)
x = tf.keras.layers.GlobalAveragePooling2D()(x)
outputs = tf.keras.layers.Dense(len(class_names), activation = 'softmax')(x)
    # Add own classififcation layer

model = tf.keras.Model(inputs, outputs)

cb = [callbacks.EarlyStopping(monitor = 'val_loss', patience = 5,
    restore_best_weights = True)]
model.summary()
```

4.4 Mã nguồn dự án

Mã nguồn dự án được tái chế lại từ mã nguồn ví dự của phát hiện người, trong đó:

- Xóa thư mục static_images để giảm dung lượng.
- Thay đổi các hàm kết quả và hiển thị (2 nhãn thành 4 nhãn).
- Thay đổi model sử dụng và các thông số sử dụng.
- ...

Chi tiết thay đổi đều nằm trong repo.

4.5 Thử nghiệm

5 Tổng kết

Nhìn chung, ứng dụng chưa hoàn thành được mục tiêu mà nó đề ra với tỉ lệ chính xác thấp (Chỉ khoản 40%).

Tài liệu

[1] Tflite micro - training a model. https://github.com/tensorflow/tflite-micro/blob/main/tensorflow/lite/micro/examples/person_detection/training_a_model.md.

[2] Coco 2014 dataset (for yolov3) - kaggle. https://www.kaggle.com/datasets/jeffaudi/coco-2014-dataset-for-yolov3.

[3] Visual wake word detection — on cainvas. https://medium.com/ai-techsystems/visual-wake-word-detection-on-cainvas-6ec3424b497e.