

# THPT CHUYÊN TRẦN ĐẠI NGHĨA



## BÁO CÁO BÀI TẬP NHÓM

—o0o—

1. Vũ Gia Bảo
2. Trương Hoàng Long
3. Châu Tân Phát
4. Đào Tuấn Sơn
5. Nguyễn Thị Phương Thảo
6. Lê Minh Tiến
7. Lê Phú Trọng

# Mục lục

<b>1</b>	<b>Giới thiệu</b>	<b>1</b>
1.1	Đặt vấn đề . . . . .	1
1.2	Giải pháp hiện tại . . . . .	1
1.3	Giải pháp của nhóm . . . . .	1
<b>2</b>	<b>Thu thập dữ liệu</b>	<b>2</b>
2.1	Nguồn thu thập . . . . .	2
2.2	Quy trình thu thập . . . . .	2
2.3	Công cụ hỗ trợ . . . . .	3
2.4	Thời gian thu thập . . . . .	3
<b>3</b>	<b>Tiền xử lý dữ liệu</b>	<b>4</b>
3.1	Cấu trúc dữ liệu . . . . .	4
3.2	Ý tưởng . . . . .	4
3.3	Cài đặt môi trường . . . . .	4
3.4	Dán nhãn dữ liệu . . . . .	4
3.4.1	Thuật toán . . . . .	4
3.4.2	Định dạng dữ liệu . . . . .	9
<b>4</b>	<b>Xây dựng mô hình</b>	<b>12</b>
4.1	Xây dựng mô hình . . . . .	12
4.1.1	Chuẩn bị môi trường làm việc . . . . .	12
4.1.2	Đồng bộ với Google Drive để lưu trữ dữ liệu . . . . .	12
4.1.3	Huấn luyện mô hình với tập dữ liệu tùy chỉnh . . . . .	12
4.1.4	Kết quả của mô hình sau khi đã huấn luyện . . . . .	13
4.2	Dự đoán với mô hình đã huấn luyện . . . . .	14
4.3	Đánh giá mô hình . . . . .	16
4.4	Thử nghiệm mô hình . . . . .	16
<b>5</b>	<b>Tổng kết</b>	<b>20</b>
<b>6</b>	<b>Tài liệu tham khảo</b>	<b>20</b>



## 1 Giới thiệu

Manga, hay còn gọi là truyện tranh Nhật Bản, được xem là biểu tượng của văn hóa đại chúng Nhật Bản, được sáng tác cho nhiều lứa tuổi và sở thích với nhiều thể loại khác nhau. Manga không chỉ là một hình thức giải trí mà còn có vai trò quan trọng trong văn hóa và giáo dục. Nó truyền tải nhiều thông điệp ý nghĩa về cuộc sống, tình bạn, gia đình và đạo đức, giúp người đọc mở rộng tư duy và cảm nhận sâu sắc hơn về thế giới (Wikipedia, Manga).

Manga không chỉ phổ biến tại Nhật Bản mà còn có sức ảnh hưởng mạnh mẽ và ngày càng lan rộng đến nhiều quốc gia khác, trong đó bao gồm Việt Nam. Để xóa bỏ rào cản ngôn ngữ trong các tác phẩm manga và để mọi người có thể tiếp cận dễ dàng hơn thì nhu cầu dịch manga ngày càng gia tăng.

### 1.1 Đặt vấn đề

Quy trình dịch manga thường chia thành 2 phần chính: 1. xử lý trang truyện và xóa chữ trong khung hội thoại, 2. dịch truyện và điền bản dịch vào khung hội thoại. Trong đó, việc xóa chữ hội thoại gốc trong truyện thường được thực hiện bởi các dịch giả (translator) hoặc biên tập viên (editor) bằng các phần mềm chỉnh sửa ảnh như Photoshop, GIMP. Điều này đòi hỏi lượng nhân công và thời gian rất lớn, nhất là trong bối cảnh những bộ manga ngày càng được du nhập và xuất bản nhiều ở khắp nơi, với các tập mới được phát hành hàng tuần hay hàng tháng.

### 1.2 Giải pháp hiện tại

Hiện nay, các công cụ AI liên quan đến công nghệ OCR (Optical character recognition) thường được đưa vào ứng dụng để tối ưu quy trình xóa chữ trong khung hội thoại (Aeonss, 2022). Tuy phương pháp này mang lại hiệu quả cao, để huấn luyện được một mô hình như thế thường tốn rất nhiều tài nguyên và thời gian.

### 1.3 Giải pháp của nhóm

Dựa trên ý tưởng rằng không nhất thiết phải huấn luyện một chương trình AI nhận diện kí tự (sử dụng OCR) như các công cụ hiện nay để có thể tự động hóa thao tác xoá chữ trong khung hội thoại, mô hình AI của nhóm được huấn luyện theo hướng nhận dạng vị trí của khung hội thoại (Do Thuan, 2021), vì thế có thể đẩy nhanh tốc độ huấn luyện cũng như thực hiện thao tác so với phương pháp hiện có.



## 2 Thu thập dữ liệu

Để huấn luyện mô hình tự động xóa chữ hội thoại, trước hết mô hình cần xác định được đúng vị trí chữ cần xóa. Để giúp mô hình làm được điều này, ta cần huấn luyện mô hình bằng tập dữ liệu hình ảnh chứa chữ hội thoại cần xóa, sau đó dán nhãn vị trí vật thể cần xóa trong tập dữ liệu hình ảnh đó.

### 2.1 Nguồn thu thập

Trang web chính thức của nhà xuất bản: Ví dụ: [shueisha](#), [Kodansha \(Nhật Bản\)](#), [Lezhin \(Hàn Quốc\)](#), [Webtoon \(Naver, Hàn Quốc\)](#). Trang web phi chính thức (cộng đồng fan dịch hoặc scan): Ví dụ: [Rawkuma](#), [MangaDex](#), [KissManga](#), [cuutruyen](#).

### 2.2 Quy trình thu thập

Bước 1: Xác định danh sách truyện cần thu thập

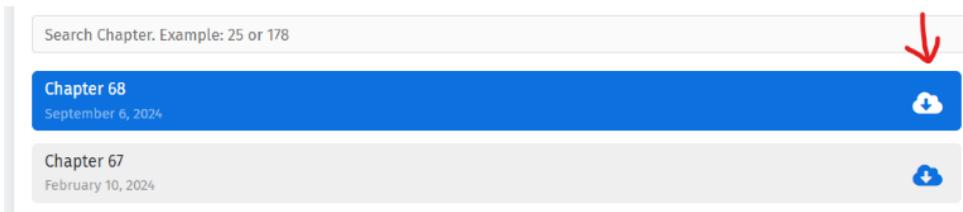
Ví dụ:

- Thể loại: Hành động, lãng mạn, hài hước.
- Định dạng: Truyện tranh đen trắng hoặc có màu.
- Độ dài: Từ 10 - 30 chương tùy theo mục tiêu nghiên cứu.

Bước 2: Truy cập các trang web đã chọn và tìm các truyện trong danh sách.

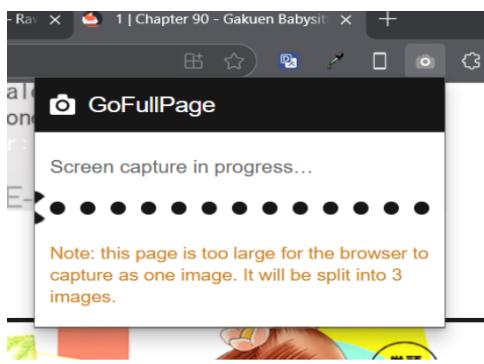
Bước 3: Tải dữ liệu hình ảnh từ từng trang/chương truyện

- Trường hợp trang web cho phép tải: Sử dụng chức năng tải xuống trực tiếp.



Hình 1: Chức năng tải truyện có sẵn của trang web rawkuma.com

- Trường hợp không có tính năng tải: Sử dụng công cụ hỗ trợ tải ảnh hàng loạt như:
  - Image Downloader (Chrome Extension): Hỗ trợ quét và tải tất cả hình ảnh từ một trang.
  - GoFullPage: Chụp toàn bộ trang web để lưu hình ảnh.



Hình 2: Giao diện của GoFullPage



## 2.3 Công cụ hỗ trợ

Phần cứng: Máy tính cấu hình tối thiểu Core i5, RAM 8GB để xử lý nhanh khối lượng dữ liệu lớn. Phần mềm:

- Trình duyệt Chrome: Image Downloader (Tải hàng loạt hình ảnh), GoFullPage(Chụp màn hình web).
- Phần mềm xử lý ảnh và xóa thoại: Photoshop, GIMP.

## 2.4 Thời gian thu thập

Ví dụ thực tế: Thu thập dữ liệu cho 12 bộ truyện, mỗi bộ 10 chương (trung bình mỗi chương có 20 trang). Thời gian:

- 1 ngày để tìm nguồn và thiết lập công cụ.
- 2 ngày để tải toàn bộ dữ liệu (tốc độ phụ thuộc vào mạng và kích thước tệp).
- 2-3 ngày để kiểm tra và xử lý dữ liệu ảnh.



Hình 3: Dữ liệu đã thu thập được (225 hình)



### 3 Tiết xử lý dữ liệu

#### 3.1 Cấu trúc dữ liệu

Để huấn luyện mô hình AI dự đoán khung chữ hội thoại, nhóm quyết định sử dụng **YOLOv5** vì tính ưu việt và tính dễ sử dụng của nó. YOLOv5 là thuật toán sử dụng kỹ thuật học máy có giám sát, vì thế ta cần tập dữ liệu được dán nhãn sẵn để huấn luyện mô hình (Ultralytics, YOLOv5, 2020). Với bài toán nhận diện vị trí vật thể, ta cần tập dữ liệu dùng để huấn luyện mô hình là các hình ảnh cần được dự đoán, đi kèm với nhãn gồm tất cả tọa độ của hình chữ nhật bao quanh vật thể cần được dự đoán, ở bài toán này, là các dòng chữ hội thoại nhân vật. Sử dụng dữ liệu đã được thu thập trước đó, chúng ta cần cài đặt chương trình giúp tạo nhãn cho từng hình ảnh trong tập dữ liệu để tiến hành huấn luyện mô hình.

#### 3.2 Ý tưởng

Sau khi đã thu thập dữ liệu cần thiết, ta cần dán nhãn cho tập dữ liệu vừa thu thập được. Nhãn cần thiết cho mỗi bức ảnh sẽ là tọa độ các hình chữ nhật chỉ vị trí của vật thể cần nhận dạng, ở bài toán này thì là chữ trong các khung hội thoại. Để làm việc này, ta có thể sử dụng các công cụ hoặc trang web giúp dán nhãn dữ liệu như [cvat.ai](#) hoặc [roboflow.com](#). Một cách khác là ta có thể xóa chữ trong ảnh một cách thủ công bằng các phần mềm chỉnh sửa hình ảnh. Khi đó, ta thu được tập dữ liệu gồm hình ảnh của các trang manga đã được xóa chữ trong các khung hội thoại. Tiếp theo, ta so sánh sự khác nhau giữa hình ảnh trước và sau khi xóa chữ và có được điểm khác biệt là chữ trong các khung hội thoại vì ta chỉ xóa chữ khỏi bản gốc mà không xóa thêm thứ gì khác. Bằng cách tìm kiếm và xử lý những điểm khác nhau này, ta có thể vẽ khung bao quanh các khu vực chứa chữ trong các khung hội thoại và tạo nhãn dán cho dữ liệu.

Để tiện xử lý, vì trang truyện là ngôn ngữ tiếng Nhật (ngôn ngữ gốc), ta sẽ chia khung hội thoại thành từng cột ký tự tiếng Nhật riêng (ví dụ có thể xem ở **Hình 7**), điều này sẽ giúp quá trình huấn luyện mô hình diễn ra hiệu quả hơn, đồng thời tránh tình trạng hộp giới hạn tràn khỏi ô hội thoại nếu ta để kích thước quá lớn.

#### 3.3 Cài đặt môi trường

Google Colab là một dịch vụ miễn phí từ Google cho phép bạn viết và chia sẻ mã Python. Google Colab cung cấp miễn phí sử dụng GPU và TPU, giúp tăng tốc độ huấn luyện mô hình máy học và thực thi tính toán lớn (FPT Shop, 2023). Không chỉ thế, Google Colab còn tải sẵn các thư viện nổi tiếng trong Python về lĩnh vực học máy như Pytorch, Tensorflow, OpenCV,... vì thế ta không cần tốn quá nhiều công sức để tải các thư viện cần dùng. Quá trình tiền xử lý dữ liệu còn có thể thực hiện trên máy tính cá nhân, ở đây, nhóm sẽ sử dụng Google Colab làm môi trường chính xử lý dữ liệu.

Để xử lý hình ảnh và dán nhãn dữ liệu, ta sẽ sử dụng ngôn ngữ lập trình Python và các thư viện hỗ trợ cho việc xử lý hình ảnh khác như: numpy, PIL. Để cài đặt thư viện còn thiếu, ta sử dụng lệnh `pip install tên_thư_viện`.

#### 3.4 Dán nhãn dữ liệu

##### 3.4.1 Thuật toán

Trong máy tính, người ta biểu diễn dữ liệu của mỗi bức ảnh là tập hợp các điểm ảnh (pixel). Mỗi ảnh sẽ bao gồm  $W \times H$  điểm ảnh, trong đó  $W$  là chiều rộng của bức ảnh,  $H$  là chiều cao của bức ảnh (Wikipedia). Ví dụ, 1 bức ảnh có kích thước  $640 \times 640$  sẽ có 409600 điểm ảnh. Trong



đó mỗi điểm ảnh sẽ chứa thông số của màu tại đó. Thông số của mỗi điểm ảnh có thể biểu diễn dưới nhiều cách khác nhau, gọi là, (ví dụ: RGB, HSV,...). Vì thế, một bức ảnh có thể biểu diễn dưới dạng 1 ma trận nhiều chiều. Vì độ phổ biến, thuật toán này sẽ xử lý hình ảnh dưới hệ màu RGB. Trong đó, mỗi điểm ảnh chứa 3 giá trị lần lượt là R, G, B, là mức sáng của 3 màu đỏ, xanh lá, xanh dương. Như vậy, ta sẽ xem như mỗi bức ảnh là một ma trận 3 chiều với kích thước  $W \times H \times 3$ , với  $W, H$  lần lượt là kích thước của bức ảnh, 3 ở đây là 3 thông số của màu RGB. Đầu tiên, ta mở 2 hình ảnh được đã chuyển đổi hệ cả 2 ảnh về hệ màu RGB. Ta sẽ cần một số hàm của thư viện PIL như sau:

Đầu tiên, ta mở 2 hình ảnh được đã chuyển đổi hệ cả 2 ảnh về hệ màu RGB. Ta sẽ cần một số hàm của thư viện PIL như sau:

```
raw_img_fname = '0_'+filename+'.jpg'  
edited_img_fname = '1_'+filename+'.jpg'  
  
# Init variable  
edited_img_file = Image.open(edited_img_fname).convert("RGB")  
raw_img_file = Image.open(raw_img_fname).convert("RGB")  
  
edited_img = edited_img_file.load()  
raw_img = raw_img_file.load()
```

**Hình 4:** Mã nguồn dùng để mở tập tin ảnh bằng các thư viện PIL

Khi này, dữ liệu của 2 biến `edited_img` hay `raw_img` sẽ là một ma trận có cấu trúc gồm 3 phần tử: 2 phần tử đầu tiên lần lượt là kích thước chiều rộng và chiều cao của bức ảnh, chiều thứ ba là một mảng gồm 3 phần tử, là thông số của điểm ảnh như đã đề cập. **Lưu ý:** 2 file ảnh phải được đảm bảo cùng kích thước và cùng định dạng tập tin.

Như ý tưởng trước đó, ta sẽ duyệt qua mọi cặp điểm ảnh trong cả 2 file ảnh, và kiểm tra những ô tương ứng xem chúng có khác nhau hay không, nếu như “khác nhau”, tức là điểm ảnh này thuộc vùng chữ cần dán nhãn, ta đánh dấu điểm ảnh này và tiến hành gộp với các ô thuộc cùng vùng chữ khác để tạo khung.

```
for y in range(0, img_height):  
    for x in range(0, img_width):  
        if checkDiff(edited_img[x, y], raw_img[x, y]):  
            diff[x, y] = True  
        for xi in range(max(0, x-7), min(img_width, x+7)):  
            for yi in range(max(0, y-20), img_width):  
                if (diff[xi, yi] == True):  
                    addEdge(x * img_height + y, xi * img_height + yi)
```

**Hình 5:** Mã nguồn của phần xử lý dữ liệu

Giải thích mã nguồn:

- Các biến và hàm trong code:

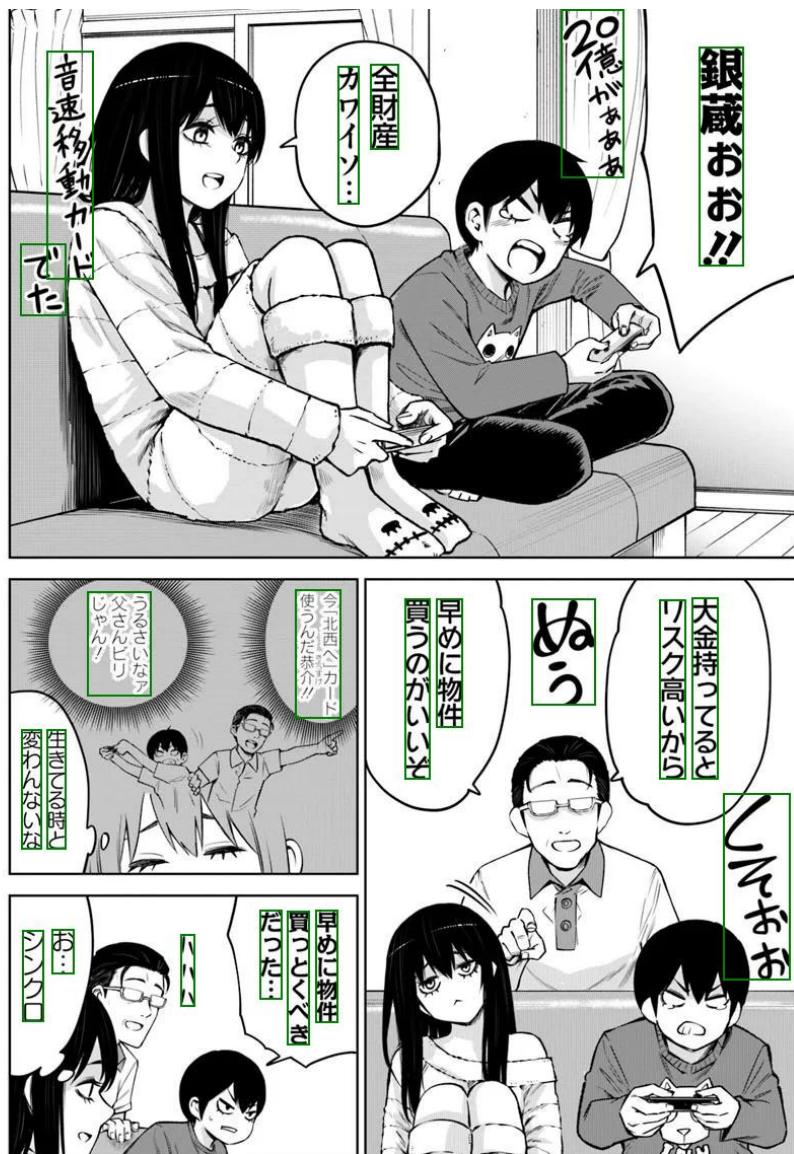


- `img_height, img_width`: độ cao và độ rộng của hình ảnh (đã được khai báo ở phần trên).
- `diff[x, y]`: ma trận 2 chiều lưu giá trị boolean là `true` nếu ô  $(x,y)$  trong ảnh gốc và ô  $(x,y)$  trong ảnh đã chỉnh sửa khác nhau, ngược lại là `false`;
- `addEdge(a, b)`: nối 2 điểm ảnh a và b.
- Vì mỗi điểm ảnh được xác định bởi 2 giá trị  $(x,y)$  của ma trận, ta có thể đánh số lại các ô trên ma trận thành 1 dãy số theo công thức:  $x \times y + H$ . Trong đó,  $(x,y)$  là tọa độ của điểm ảnh,  $H$  là chiều cao ma trận của tập tin ảnh. (Công thức này chỉ được áp dụng cho mảng 0-indexed).
- Khi đó ma trận ảnh có thể biểu diễn dưới dạng đồ thị  $H \times W$  đỉnh. Ta sẽ thực hiện việc nối đỉnh.
- Duyệt qua từng điểm ảnh  $(x,y)$  của hình, hàm `checkDiff` sẽ kiểm tra độ khác nhau giữa 2 điểm ảnh (sẽ được giải thích ở đoạn sau).
- Nếu ô  $(x,y)$  trong ảnh gốc khác ô  $(x,y)$  trong ảnh đã chỉnh sửa: lưu vào mảng `diff` và duyệt qua những ô xung quanh theo một kích thước xác định. Nếu ô  $(x_i, y_i)$  trong ảnh gốc cũng khác ô  $(x_i, y_i)$  trong ảnh đã chỉnh sửa, ta nối 2 đỉnh  $(x,y)$  và  $(x_i, y_i)$  với nhau.
- Hàm nối cạnh `addEdge(a, b)` sẽ tạo 1 cạnh trong đồ thị.

Khi này, những ô đã được nối sẽ tạo thành 1 thành phần liên thông, ta lấy hình chữ nhật ngoại tiếp của thành phần liên thông đó. Với mỗi một thành phần liên thông như vậy, ta sẽ lưu các tọa độ  $x, y$  nhỏ nhất và lớn nhất trong tất cả các điểm thuộc thành phần liên thông này, gọi lần lượt là  $x_{min}, y_{min}, x_{max}, y_{max}$ . Như vậy, khung chữ ta cần dán nhãn sẽ là hình chữ nhật có tọa độ trái trên là  $(x_{min}, y_{min})$  và tọa độ phải dưới là  $(x_{max}, y_{max})$ . Ta có thể cập nhật mỗi thành phần liên thông bằng cách duyệt qua ma trận 1 lần nữa bằng thuật toán **Depth First Search (DFS)**.



**Hình 6:** Bước 1, giả sử những điểm ảnh khác biệt giữa ảnh gốc và ảnh đã chỉnh sửa là những ô màu đen, điểm ảnh ta đang xét là ô màu xanh dương. Bước 2, ta duyệt hết những ô lân cận trong một vùng kích thước xác định, và nối những ô đó lại với nhau, tạo thành vùng màu đỏ. Bước 3, khi đã thực hiện hoàn tất bước nối các điểm ảnh, ta sẽ lấy hình chữ nhật ngoại tiếp với vùng đó (hình chữ nhật màu xanh lá)



Hình 7: Kết quả sau khi áp dụng thuật toán

Giải thích chi tiết về hàm checkDiff kiểm tra độ khác biệt: Vì bước tiền xử lý đôi khi có thể khiến bức ảnh bị thay đổi ở một vài pixel, dẫn đến hình ảnh khi xuất ra bị nhiễu nếu ta so sánh như thường. Để khắc phục, ta sẽ xem 2 điểm ảnh là "khác nhau" khi tổng của khoảng cách của mọi mức sáng màu  $> 100$ . Ví dụ pixel A có mã màu là  $(0, 0, 0)$ , pixel B là  $(177, 0, 13)$ , giá trị khác nhau của nó là  $(177 - 0) + (0 - 0) + (13 - 0) = 190 > 100$ , vì thế, ta nói "pixel A khác pixel B".



**Hình 8:** Một trường hợp khi dữ liệu bị nhiễu: vì một số pixel bị thay đổi nên thuật toán vô tình nhận dạng "sai" ô chữ



**Hình 9:** Hình ảnh sau khi chỉnh sửa thuật toán kiểm tra độ khác biệt

Thuật toán trên không đáp ứng yêu cầu về tốc độ xử lý. Vì thế, ta phải có một số bước tối ưu.

**Tối ưu 1:** Việc kết nối các ô lại với nhau, thay vì nối đỉnh và duyệt DFS từng thành phần liên thông, ta có thể sử dụng cấu trúc dữ liệu **Disjoint Set Union (DSU)**. Trong đó, mỗi thành phần của tập hợp sẽ chứa các giá trị:

- par: gốc của tập hợp.
- sz: kích thước của tập hợp.
- xl, yl, xr, yr: lần lượt là tọa độ trái, trên, phải, dưới của hình chữ nhật ngoại tiếp thành phần liên thông trong tập hợp.

```
for y in range(0, img_height):
    for x in range(0, img_width):
        if checkDiff(edited_img[x, y], raw_img[x, y]):
            diff[x, y] = True
            for xi in range(max(0, x-7), min(img_width, x+7)):
                for yi in range(max(0, y-20), min(img_width, y+20)):
                    if (diff[xi, yi] == True):
                        Union_set(x * img_height + y, xi * img_height + yi)
```

**Hình 10:** Tối ưu 1: Thay hàm addEdge thành hàm gộp tập hợp của DSU, ta giảm thiểu được bước DFS

**Tối ưu 2:** Ta cần chọn một kích thước hình chữ nhật tối ưu hơn, cụ thể, ta chỉ cần duyệt theo



3 hướng thay vì 4 hướng.



**Hình 11:** *Tối ưu 2. Trong hình bên trái, nếu ta chọn kích thước hình chữ nhật bao 4 hướng xung quanh, điểm màu đen ở trên sẽ nối với điểm màu đen ở dưới, và ngược lại. Điều này là không cần thiết, sẽ gây hao phí thời gian khi xử lý. Trong hình bên phải, ta chọn kích thước hình chữ nhật có một cạnh chứa đỉnh đang xét. Lúc này mỗi cặp đỉnh sẽ được nối với nhau duy nhất 1 lần.*

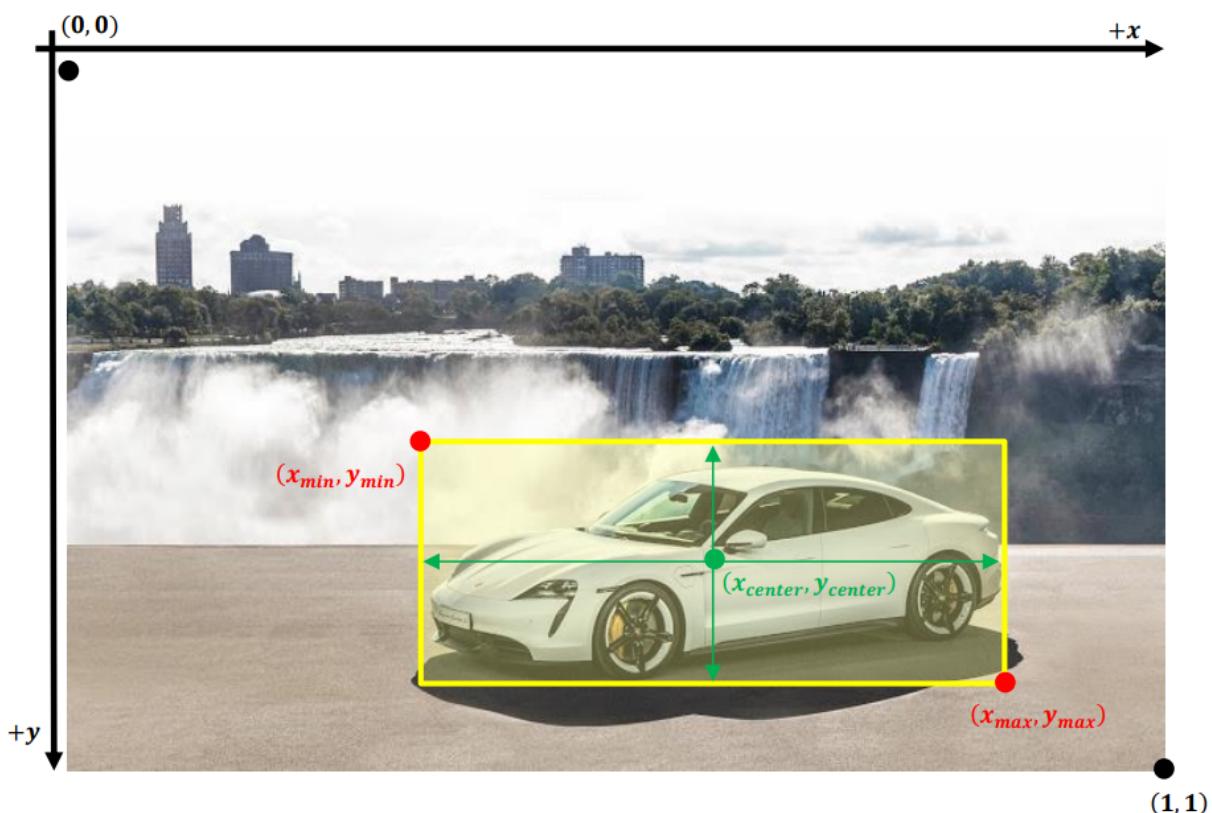
```
for y in range(0, img_height):
    for x in range(0, img_width):
        if checkDiff(edited_img[x, y], raw_img[x, y]):
            diff[x, y] = True
            for xi in range(max(0, x-7), min(img_width, x+7)):
                for yi in range(max(0, y-20), y):
                    if (diff[xi, yi] == True):
                        Union_set(x * img_height + y, xi * img_height + yi)
```

**Hình 12:** *Tối ưu 2: Ta chỉ duyệt theo 3 hướng thay vì 4 hướng*

### 3.4.2 Định dạng dữ liệu

YOLOv5 yêu cầu định dạng cũng nhãn được gán cho mỗi hình ảnh là một file văn bản định dạng .txt chứa tất cả tọa độ của hình chữ nhật chứa vật thể cần dự đoán và lớp (class) của chúng (roboflow, YOLOv5 PyTorch TXT, 2020). Mỗi dòng trong file này sẽ có dạng class\_id x\_center y\_center width height, trong đó:

- class\_id: Chỉ số của class cần dự đoán, được đánh dấu từ 0. Ở tập dữ liệu của chúng ta, vì ta chỉ cần mô hình nhận dạng được vị trí khung chữ, nên chỉ có 1 class duy nhất cần dự đoán (chữ hội thoại).
- x\_center y\_center: Tọa độ tâm hình chữ nhật.
- width height: Kích thước hình chữ nhật.



Hình 13: Tọa độ hộp giới hạn theo định dạng của YOLO

Sau bước xử lý trên, với mỗi thành phần liên thông (tập các điểm ảnh cùng nằm trong một khung chữ hội thoại), ta sẽ vẽ được một hình chữ nhật bao quanh thành phần liên thông đó. Với mỗi thành phần liên thông, ta đã lưu được  $(x_{min}, y_{min})$  và  $(x_{max}, y_{max})$  là tọa độ của hình chữ nhật cần tìm. Tuy nhiên, tọa độ hình chữ nhật trong định dạng chuẩn đầu vào của YOLO, như đã nói ở trên, sẽ có giá trị trong đoạn  $[0; 1]$ , theo tỉ lệ của hình, vì thế ta cần phải chuẩn hóa các tọa độ mà ta đang có theo định dạng của YOLO, bằng công thức sau:

$$x_{center} = \frac{x_{min} + x_{max}}{2}$$

$$y_{center} = \frac{y_{min} + y_{max}}{2}$$

`width`, `height` là kích thước của hình chữ nhật, có thể tính như sau:

$$width = x_{max} - x_{min}$$

$$height = y_{max} - y_{min}$$

Tiếp theo, ta chỉnh lại các tọa độ của hộp giới hạn theo tỉ lệ khung hình, các tọa độ trên sẽ được chia cho chiều rộng hoặc chiều cao của hình để giá trị nằm trong đoạn  $[0; 1]$ , ta sẽ xây dựng hàm convert như sau:

```
def convert(box):
    dw = 1./img_width
    dh = 1./img_height
    x, y, w, h = (box[0] + box[1])/2.0, (box[2] + box[3])/2.0, box[1] - box[0], box[3] - box[2]
    x, y, w, h = x*dw, y*dh, w*dw, h*dh
    return (x, y, w, h)
```

Hình 14: Hàm chuyển đổi tọa độ thường sang tọa độ chuẩn của YOLO



## Trường THPT chuyên Trần Đại Nghĩa

### Lớp 12CTin

Trong đó, box [0] và box [1] lần lượt là  $x_{min}$  và  $y_{min}$  của hình chữ nhật, tương tự với box [2] và box [3] lần lượt là  $x_{max}$  và  $y_{max}$ . Hàm sẽ trả về tọa độ của hình sau khi được chuẩn hóa.

Sau khi chuẩn hóa các tọa độ, ta sẽ duyệt lại các thành phần liên thông nói trên, và in tất cả tọa độ hộp giới hạn ra file .txt chứa nhãn của hình ảnh.

Name	Date modified	Type	Size
0_001.txt	1/17/2025 5:56 PM	Text Document	1 KB
0_002.txt	1/17/2025 5:56 PM	Text Document	1 KB
0_003.txt	1/17/2025 5:56 PM	Text Document	1 KB
0_004.txt	1/17/2025 5:56 PM	Text Document	1 KB
0_005.txt	1/17/2025 5:56 PM	Text Document	1 KB
0_006.txt	1/17/2025 5:56 PM	Text Document	2 KB
0_007.txt	1/17/2025 5:56 PM	Text Document	3 KB
0_008.txt	1/17/2025 5:56 PM	Text Document	3 KB
0_009.txt	1/17/2025 5:56 PM	Text Document	4 KB
0_010.txt	1/17/2025 5:57 PM	Text Document	4 KB
0_011.txt	1/17/2025 5:57 PM	Text Document	2 KB
0_012.txt	1/17/2025 5:57 PM	Text Document	2 KB
0_013.txt	1/17/2025 5:57 PM	Text Document	1 KB
0_014.txt	1/17/2025 5:57 PM	Text Document	2 KB
0_015.txt	1/17/2025 5:57 PM	Text Document	2 KB
0_016.txt	1/17/2025 5:57 PM	Text Document	2 KB
0_017.txt	1/17/2025 5:57 PM	Text Document	1 KB
0_018.txt	1/17/2025 5:57 PM	Text Document	1 KB
0_019.txt	1/17/2025 5:57 PM	Text Document	1 KB
0_020.txt	1/17/2025 5:57 PM	Text Document	1 KB
0_021.txt	1/17/2025 5:57 PM	Text Document	1 KB
0_022.txt	1/17/2025 5:57 PM	Text Document	1 KB
0_023.txt	1/17/2025 5:57 PM	Text Document	1 KB
0_024.txt	1/17/2025 5:57 PM	Text Document	1 KB
0_025.txt	1/17/2025 5:57 PM	Text Document	1 KB
0_026.txt	1/17/2025 5:57 PM	Text Document	2 KB
0_027.txt	1/17/2025 5:57 PM	Text Document	2 KB
0_028.txt	1/17/2025 5:57 PM	Text Document	1 KB
0_029.txt	1/17/2025 5:57 PM	Text Document	1 KB
0_030.txt	1/17/2025 5:57 PM	Text Document	1 KB

Hình 15: Nhãn của dữ liệu sau khi đã tiền xử lý



## 4 Xây dựng mô hình

### 4.1 Xây dựng mô hình

Để tiến hành xây dựng mô hình, đầu tiên ta sẽ chuẩn bị môi trường huấn luyện, ở đây ta sẽ sử dụng Google Colab. Quy trình xây dựng mô hình bao gồm các bước sau:

#### 4.1.1 Chuẩn bị môi trường làm việc

Đầu tiên, ta chạy lệnh sau để cài đặt các thư viện cần thiết cho yolov5. Sau khi cài đặt xong, ta import các thư viện cần dùng và in ra “Set up completed!” sau khi đã thực hiện xong bước trên.

```
!git clone https://github.com/ultralytics/yolov5 # clone
%cd yolov5
%pip install -qr requirements.txt comet_ml # install

import torch
import utils
from IPython.display import clear_output

clear_output()
display = utils.notebook_init() # checks
print("Setup completed!")
```

Hình 16: Cài đặt những thư viện cần thiết

#### 4.1.2 Đồng bộ với Google Drive để lưu trữ dữ liệu

Vì môi trường sử dụng là Google Colab, ta sẽ tải bộ dữ liệu dùng để huấn luyện mô hình lên bộ lưu trữ đám mây Google Drive. Sau khi tải các dữ liệu cần thiết lên Google Drive, ta chạy lệnh sau để thư mục lưu trữ tập tin hiện tại của notebook có thể đồng bộ với bộ nhớ của Google Drive, từ đó ta có thể truy cập và xử lý trực tiếp dữ liệu từ Google Colab.

```
from google.colab import drive
drive.mount('/content/gdrive')
```

Hình 17: Đồng bộ với Google Drive

#### 4.1.3 Huấn luyện mô hình với tập dữ liệu tùy chỉnh

Để tiến hành huấn luyện mô hình, ta sẽ thực hiện lệnh sau để chạy tập tin train.py và bắt đầu huấn luyện mô hình:

```
!python train.py --img 640 --batch 16 --epochs 100 --data data.yaml --weights yolov5s.pt
```

Hình 18: Bắt đầu huấn luyện mô hình

Các tham số trong câu lệnh có ý nghĩa như sau (Do Thuan, 2021):



- -img 640: Kích thước ảnh đầu vào. Ban đầu, kích thước các ảnh lớn, và để quá trình huấn luyện mô hình sẽ trả nén nhanh hơn thì ta sẽ nén các ảnh thành kích thước nhỏ hơn. Qua nhiều thử nghiệm, nhóm đã thống nhất rằng kích thước 640 x 640 là kích thước lý tưởng nhất để nén ảnh vì vừa tối ưu thời gian huấn luyện mà cũng không làm mất quá nhiều chi tiết trong ảnh.
- -batch 16: Số lượng ảnh xử lý trong mỗi batch. Dữ liệu sẽ được chia thành các batch nhỏ, trong đó mỗi batch chứa một lượng nhỏ ảnh, rồi xử lý từng batch một. Giả sử, ta có 160 ảnh, và kích thước của batch là 16, nên số lượng batch sẽ là  $160/16 = 10$ . Dữ liệu học được từ batch sẽ được lưu trữ trong RAM nên càng nhiều batch, càng tốn dung lượng RAM. Tuy nhiên, xử lý quá nhiều bức ảnh cùng một lúc có thể làm tăng lượng dữ liệu mà mô hình cần học trong một khoảng thời gian tăng nhanh. Chung quy lại, số lượng batch quá lớn thì dung lượng RAM sẽ bị tiêu tốn nhiều, nhưng quá nhỏ thì không đủ hiệu quả để giảm thời gian. Ta sẽ cân bằng cả hai yếu tố bằng để kích thước batch là 16.
- -epochs 100: Số lần huấn luyện. Vì toàn bộ các hình ảnh được chia thành các batch, nên trong mỗi lần huấn luyện thì mô hình sẽ học tất cả các batch và cập nhật các trọng số (trong neural network) để làm việc chính xác hơn. Ta sẽ cho mô hình huấn luyện 100 lần.
- -data data.yaml: Đường dẫn đến file tổng quan về bộ dữ liệu. Quá trình đánh giá mô hình sẽ được xử lý ngay sau mỗi epoch, nên mô hình có thể truy cập vào file tổng hợp cấu trúc dữ liệu để sử dụng thông tin trong file cho việc đánh giá. Tập tin này sẽ chứa các đường dẫn đến các thư mục chứa dữ liệu dùng để huấn luyện, đánh giá mô hình, đồng thời cho biết thông tin tất cả class có trong bộ dữ liệu. Nhóm sẽ trích 80% dữ liệu sẵn có để huấn luyện mô hình, 20% dữ liệu còn lại dùng để đánh giá mô hình qua từng lần chạy (epoch).

```
1 path: ../data # dataset root dir
2 train: images # train images (relative to 'path') 128 images
3 val: images # val images (relative to 'path') 128 images
4 test: # test images (optional)
5
6 # Classes
7 names:
8 | 0: text
```

Hình 19: Nội dung trong tập tin data.yaml

- -weights yolov5s.pt: Đường dẫn đến mô hình có sẵn. Ở đây ta sẽ bắt đầu huấn luyện mô hình dựa trên mô hình có sẵn của YOLOv5 - yolov5s.pt.

#### 4.1.4 Kết quả của mô hình sau khi đã huấn luyện

Sau khi quá trình huấn luyện mô hình hoàn tất, ta sẽ nhận được các thông báo qua terminal như sau:



# Trường THPT chuyên Trần Đại Nghĩa

## Lớp 12CTin

```
99/99      3.97G   0.04539   0.05836      0       423       640: 92% 11/12 [00:02<00:00, 4.05it/s]/content/gd
with torch.cuda.amp.autocast(amp):
  99/99      3.97G   0.04537   0.05879      0       118       640: 100% 12/12 [00:03<00:00, 3.95it/s]
    Class     Images Instances      P       R   mAP50   mAP50-95: 100% 2/2 [00:00<00:00, 3.67it/s]
      all      45      581      0.957      0.837      0.868      0.664

100 epochs completed in 0.140 hours.
Optimizer stripped from runs/train/exp6/weights/last.pt, 14.4MB
Optimizer stripped from runs/train/exp6/weights/best.pt, 14.4MB

Validating runs/train/exp6/weights/best.pt...
Fusing layers...
Model summary: 157 layers, 7012822 parameters, 0 gradients, 15.8 GFLOPs
    Class     Images Instances      P       R   mAP50   mAP50-95: 100% 2/2 [00:00<00:00, 2.43it/s]
      all      45      581      0.962      0.832      0.873      0.667
Results saved to runs/train/exp6
```

**Hình 20:** Một đoạn trích từ hộp thoại terminal sau khi huấn luyện hoàn tất

Từ hình 20, ta có thể thấy tổng quan kết quả của mô hình sau quá trình huấn luyện kéo dài 0.14 giờ (xấp xỉ 8 phút). Theo ví dụ trên, thông tin và kết quả của mô hình sẽ được lưu ở đường dẫn `runs/train/exp6` (nằm ở thư mục ta đã truy cập lúc đầu), với cấu trúc và trọng số của mô hình được lưu bên trong thư mục `weights` nằm ở cùng đường dẫn.

## 4.2 Dự đoán với mô hình đã huấn luyện

Trong quá trình huấn luyện, mô hình của chúng ta sẽ được chạy thử trên các dữ liệu không được dùng trong lúc huấn luyện, là những dữ liệu mà ta đã trích xuất 20% ở bước trước, những dữ liệu này sẽ được dùng để đánh giá sơ bộ về mô hình qua mỗi lần huấn luyện.



**Hình 21:** Mô hình được thử nghiệm trên bộ dữ liệu hình ảnh dùng để đánh giá

Sau khi quá trình huấn luyện mô hình kết thúc, YOLOv5 cung cấp cho chúng ta các biểu đồ để đánh giá mô hình một cách trực quan. Kết quả đánh giá của mô hình được lưu trong thư mục chứa thông tin của lần huấn luyện.



Để hiểu được những thông tin được YOLOv5 cung cấp, ta cần biết về các thông số sau:

- Precision: Là giá trị biểu thị tỉ lệ của những giá trị dự đoán chính xác trên tổng giá trị dự đoán, nói cách khác là số lần đoán đúng trên số lần đoán. Precision được tính bằng công thức:

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

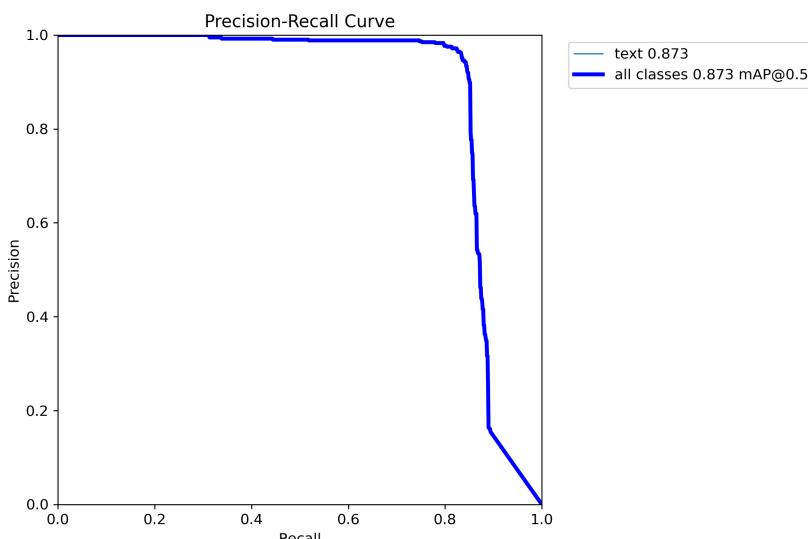
- Recall: Là giá trị biểu thị tỉ lệ của số lần đoán đúng trên số kết quả đúng. Recall được tính bằng công thức:

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

- F1-Score: Là sự kết hợp giữa Precision và Recall. F1-Score được tính bằng công thức::

$$F1-score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

- mAP: medium Average Precision hay trung bình độ chính xác, là giá trị phổ biến để đo lường độ chính xác của bài toán object detection. Do ở đây chỉ có 1 class dữ liệu là class text, mAP cũng chính là AP của class này. Để tìm được AP thì ta cần biết đến Precision-Recall Curve:

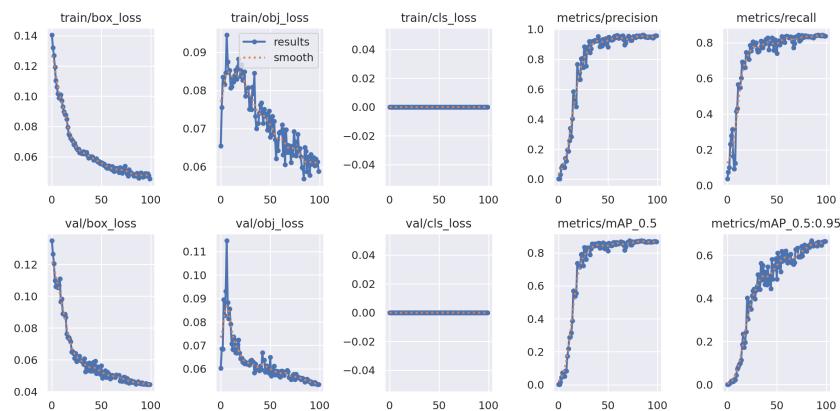


Hình 22: Biểu đồ PR\_Curve

Precision-Recall Curve có thể hiểu là sự thay đổi Precision và Recall khi ta thay đổi confidence score, AP sẽ là phần diện tích được tạo bởi đường cong này và 2 trục. Với công thức:

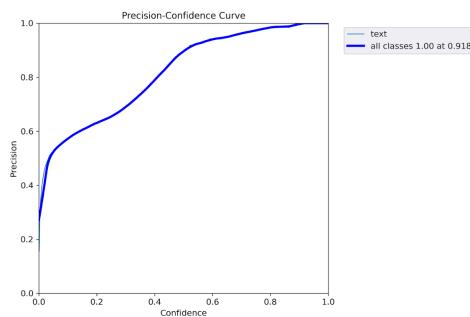
$$AP = \int_0^1 P(r) dr$$

(P(r) là phương trình biểu thị sự phụ thuộc của Precision vào Recall) Đây là một vài đồ thị cho thấy sự thay đổi của các số liệu trên của mô hình sau khi được huấn luyện theo các tham số trong phần 4.1.3:

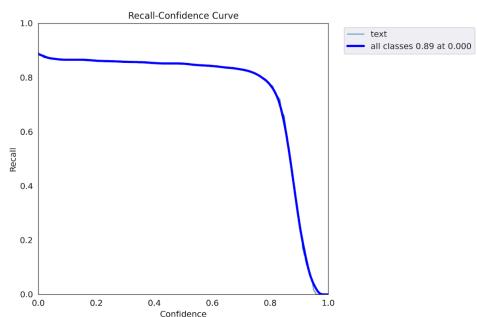


Ý nghĩa của 6 giá trị và đồ thị tương ứng liên quan đến hàm mất mát (loss function) như sau:

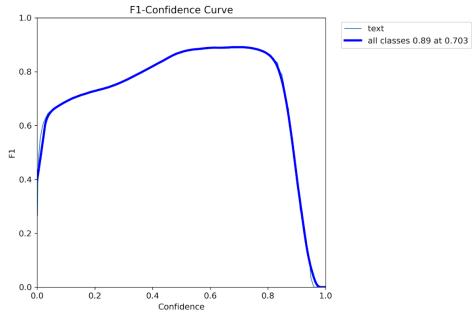
### 4.3 Đánh giá mô hình



Hình 23: Biểu đồ P\_Curve



Hình 24: Biểu đồ R\_Curve



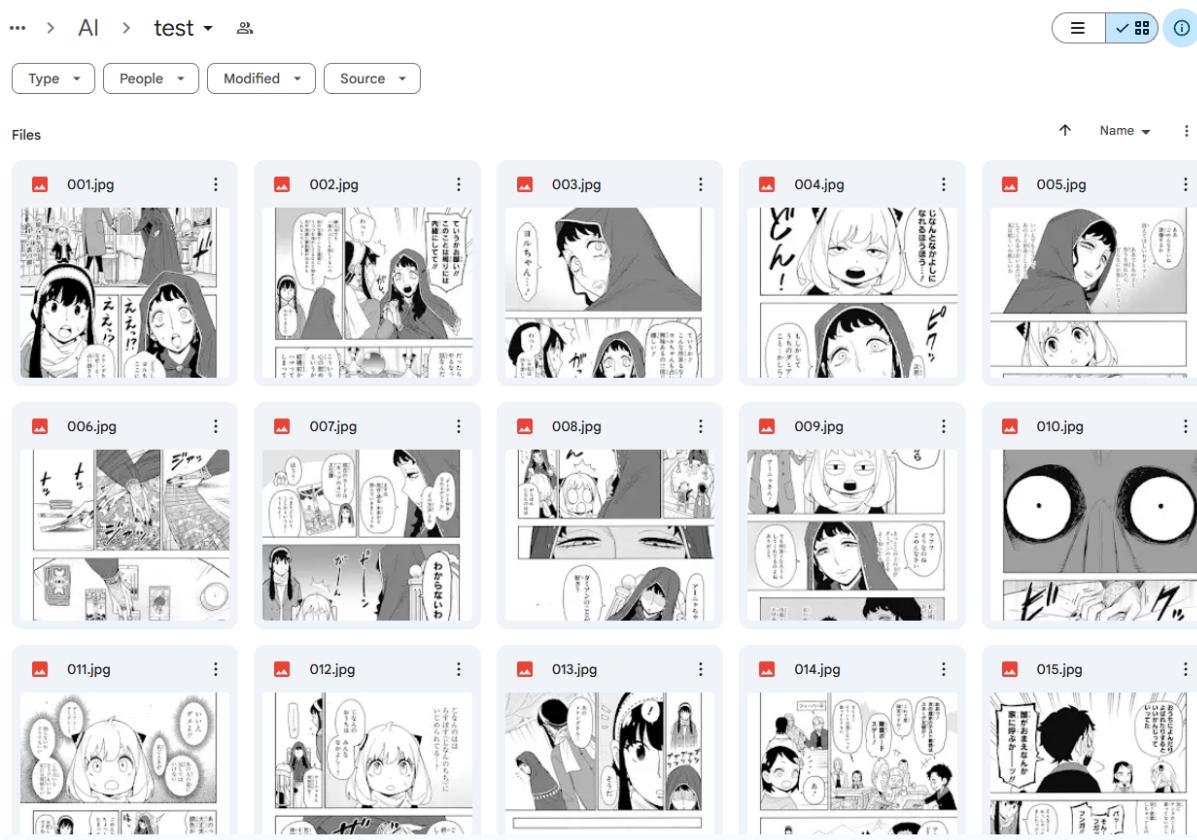
Hình 25: Biểu đồ F1-Score

Đây là 3 biểu đồ biểu thị sự phụ thuộc của Recall, Precision và F1-score của model vào confidence score. Vì ứng dụng model của nhóm cần cả Recall và Precision, biểu đồ của F1-score cho ta thấy được rằng ở mức confidence score xấp xỉ 0.8, model sẽ chạy hiệu quả.

Giá trị mAP khoảng 0.87, theo nhóm đánh giá thì model có thể chạy ổn định nhưng chưa đáng tin cậy.

### 4.4 Thử nghiệm mô hình

Ta sẽ thử nghiệm mô hình với bộ dữ liệu 25 hình ảnh truyền tranh được lấy từ truyện **Spy x Family, Chapter 108**. Các hình ảnh dùng để thử nghiệm này sẽ được đặt trong thư mục test.



Hình 26: Dữ liệu đầu vào, Spy x Family, Chapter 108

Để tiến hành dự đoán thử vị trí của class text trong hình ảnh bằng mô hình đã huấn luyện, ta sử dụng lệnh sau:

```
!python detect.py --weights runs/train/exp6/weights/best.pt --img 640 --source ./test
```

Hình 27: Sử dụng mô hình đã huấn luyện để thực hiện dự đoán

Trong đó:

- weights: Đường dẫn đến tập tin chứa mô hình mà ta sử dụng để dự đoán.
- img: Kích thước ảnh đầu vào, ý nghĩa tương tự giống với lúc ta huấn luyện mô hình
- source: Đường dẫn đến nguồn các dữ liệu cần được dự đoán. Ở ví dụ này, đầu vào của chúng ta sẽ là đường dẫn đến thư mục chứa tập các hình ảnh cần dự đoán, thư mục này có tên là test.

```
python detect.py --weights yolov5s.pt --source 0
image.jpg
video.mp4
screen
path/
list.txt
# webcam
# image
# video
# screenshot
# directory
# list of images
```

Hình 28: Một số đầu vào dữ liệu phổ biến mà mô hình hỗ trợ dự đoán (Ultralytics)

Sau khi hoàn tất dự đoán, ta nhận được hộp thoại terminal như sau:



Fusing layers...

```
Model summary: 157 layers, 7012822 parameters, 0 gradients, 15.8 GFLOPs
image 1/25 /content/gdrive/MyDrive/DuAn/AI/test/001.jpg: 640x416 12 texts, 30.0ms
image 2/25 /content/gdrive/MyDrive/DuAn/AI/test/002.jpg: 640x416 34 texts, 8.2ms
image 3/25 /content/gdrive/MyDrive/DuAn/AI/test/003.jpg: 640x416 21 texts, 10.7ms
image 4/25 /content/gdrive/MyDrive/DuAn/AI/test/004.jpg: 640x416 18 texts, 8.2ms
image 5/25 /content/gdrive/MyDrive/DuAn/AI/test/005.jpg: 640x416 8 texts, 8.2ms
image 6/25 /content/gdrive/MyDrive/DuAn/AI/test/006.jpg: 640x416 11 texts, 12.4ms
image 7/25 /content/gdrive/MyDrive/DuAn/AI/test/007.jpg: 640x416 29 texts, 8.2ms
image 8/25 /content/gdrive/MyDrive/DuAn/AI/test/008.jpg: 640x416 15 texts, 8.2ms
image 9/25 /content/gdrive/MyDrive/DuAn/AI/test/009.jpg: 640x416 12 texts, 8.2ms
image 10/25 /content/gdrive/MyDrive/DuAn/AI/test/010.jpg: 640x416 (no detections), 8.2ms
image 11/25 /content/gdrive/MyDrive/DuAn/AI/test/011.jpg: 640x416 26 texts, 8.2ms
image 12/25 /content/gdrive/MyDrive/DuAn/AI/test/012.jpg: 640x416 22 texts, 8.2ms
image 13/25 /content/gdrive/MyDrive/DuAn/AI/test/013.jpg: 640x416 6 texts, 8.2ms
image 14/25 /content/gdrive/MyDrive/DuAn/AI/test/014.jpg: 640x416 30 texts, 8.2ms
image 15/25 /content/gdrive/MyDrive/DuAn/AI/test/015.jpg: 640x416 33 texts, 8.3ms
image 16/25 /content/gdrive/MyDrive/DuAn/AI/test/016.jpg: 640x416 22 texts, 8.8ms
image 17/25 /content/gdrive/MyDrive/DuAn/AI/test/017.jpg: 640x416 17 texts, 8.2ms
image 18/25 /content/gdrive/MyDrive/DuAn/AI/test/018.jpg: 640x416 12 texts, 8.2ms
image 19/25 /content/gdrive/MyDrive/DuAn/AI/test/019.jpg: 640x416 25 texts, 8.2ms
image 20/25 /content/gdrive/MyDrive/DuAn/AI/test/020.jpg: 640x416 3 texts, 8.2ms
image 21/25 /content/gdrive/MyDrive/DuAn/AI/test/021.jpg: 640x416 6 texts, 8.2ms
image 22/25 /content/gdrive/MyDrive/DuAn/AI/test/022.jpg: 640x416 19 texts, 8.2ms
image 23/25 /content/gdrive/MyDrive/DuAn/AI/test/023.jpg: 640x416 8 texts, 8.2ms
image 24/25 /content/gdrive/MyDrive/DuAn/AI/test/024.jpg: 640x416 28 texts, 8.2ms
image 25/25 /content/gdrive/MyDrive/DuAn/AI/test/025.jpg: 640x416 16 texts, 8.2ms
Speed: 0.5ms pre-process, 9.4ms inference, 13.0ms NMS per image at shape (1, 3, 640, 640)
Results saved to runs/detect/exp14
```

**Hình 29:** Hộp thoại terminal sau khi mô hình hoàn thành quá trình dự đoán

Qua hộp thoại này, ta có thể thấy sơ bộ về kết quả sau khi dự đoán. Kết quả tập dữ liệu đã được mô hình dữ đoán sẽ được lưu ở thư mục runs/detect/exp14.



**Hình 30:** Một hình ảnh từ tập dữ liệu đã được dự đoán vị trí chữ hội thoại

Qua đánh giá sơ bộ kết quả bằng mắt thường, tuy vẫn còn nhiều hộp thoại được nhận dạng không đúng, nhưng có thể thấy mô hình dự đoán tương đối ổn định và chính xác trong hầu hết trường hợp.



## 5 Tổng kết

Như vậy, nhóm đã thu thập đầy đủ dữ liệu cần thiết để huấn luyện mô hình nhận dạng lời thoại trong manga, đồng thời sử dụng Google Colab và các thư viện xử lý hình ảnh để dán nhãn cho tập dữ liệu sao cho phù hợp với định dạng yêu cầu của YOLOv5.

## 6 Tài liệu tham khảo

- [1] Wikipedia, Manga, <https://vi.wikipedia.org/wiki/Manga>.
- [2] Aeonss (2022), Bubble Blaster, <https://github.com/Aeonss/BubbleBlaster>.
- [3] Do Thuan (2021), Evolution of YOLO algorithm and YOLOv5: The state-of-the-art object detection algorithm, <https://www.theseus.fi/bitstream/handle/10024/452552/DoThuan.pdf>.
- [4] Ultralytics (2020), YOLOv5, <https://github.com/ultralytics/yolov5>.
- [5] FPT Shop (2023), Google Colab là gì?, <https://fptshop.com.vn/tin-tuc/danh-gia/google-colab-167087>.
- [6] Wikipedia, Pixel, <https://vi.wikipedia.org/wiki/Pixel>.
- [7] roboflow (2020), YOLOv5 PyTorch TXT, <https://roboflow.com/formats/yolov5-pytorch-txt>.