

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN I

—o0o—



ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC

**BAO LỖI XẤP XỈ VÀ ỨNG DỤNG TRONG VIỆC PHÁT
HIỆN ĐỊNH HƯỚNG VÀ ĐÓNG GÓI ĐỐI TƯỢNG**

Giảng Viên Hướng Dẫn:	TS. Nguyễn Kiều Linh
Sinh viên thực hiện:	Trần Xuân Độ
Mã sinh viên:	B19DCCN183
Lớp:	D19CNPM04
Niên khóa:	2019-2023
Hệ đào tạo:	Đại học chính quy

Hà Nội, 12/2023

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN I

—o0o—



ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC

**BAO LỖI XẤP XỈ VÀ ỨNG TRONG VIỆC PHÁT HIỆN
ĐỊNH HƯỚNG VÀ ĐÓNG GÓI ĐỐI TƯỢNG.**

Giảng Viên Hướng Dẫn:	TS. Nguyễn Kiều Linh
Sinh viên thực hiện:	Trần Xuân Độ
Mã sinh viên:	B19DCCN183
Lớp:	D19CNPM04
Niên khóa:	2019-2023
Hệ đào tạo:	Đại học chính quy

Hà Nội, 12/2023

Mục lục

Danh sách hình vẽ	iv
Danh sách bảng	vi
Danh sách các ký hiệu và chữ viết tắt	vii
Mở đầu	1
1 Giới thiệu bài toán phát hiện, định hướng và đóng gói đối tượng	3
1.1 Giới thiệu	3
1.2 Xây dựng tập bao lồi đầu tiên	4
1.3 Tích chập biến dạng	5
1.4 Thuật toán Convex Approximation	8
1.4.1 Thuật toán Outer Convex Approximation	8
1.4.2 Inner Convex Approximation	9
1.5 Định nghĩa công thức Convex Intersection over Union (CIoU) . .	9
1.6 Hàm mất mát	10
1.7 Thích ứng bao lồi	11
1.7.1 Xây dựng tập các bao lồi	11
1.7.2 Chiến lược phân đoạn tập các bao lồi	12
2 Thuật toán tính bao lồi xấp xỉ	15
2.1 Outer convex approximation	15
2.2 Inner convex approximation	22
3 Thực nghiệm và kết quả	28

3.1	Khởi tạo môi trường chạy	28
3.2	Tập dữ liệu DOTA	29
3.3	Cách thức thay thế thuật toán	31
4	Một số kết quả tính toán	36
4.1	Thay thế Outer Convex Approximation và huấn luyện sử dụng bộ dữ liệu đầy đủ	38
4.2	Thay thế Outer Convex Approximation và huấn luyện sử dụng dataset nhỏ	40
4.3	Thay thế Outer Convex Approximation vào hàm Jarvis() và huấn luyện sử dụng dataset nhỏ	41
4.4	Thay thế Inner Convex Approximation và huấn luyện sử dụng toàn bộ dữ liệu đầy đủ	42
4.5	Thay thế Inner Convex Approximation và huấn luyện sử dụng bộ dữ liệu nhỏ	42
	Kết luận	44

LỜI CẢM ƠN

Em xin chân thành cảm ơn cô giáo Nguyễn Kiều Linh, người đã đóng góp không ngừng sức mạnh và sự hiểu biết chuyên sâu, giúp em hoàn thành đề án một cách thành công. Những lời hướng dẫn và sự chia sẻ của cô không chỉ giúp em giải quyết các vấn đề khó khăn mà còn mở rộng tầm nhìn và kiến thức của em trong lĩnh vực công nghệ thông tin.

Em cũng muốn bày tỏ lòng biết ơn đặc biệt đến tác giả của các thư viện code Mmrotate và tác giả bài báo `BeyondBoundingBox` [5], người đã chia sẻ những lời khuyên quý báu qua email. Điều này thực sự là một nguồn động viên lớn, giúp em áp dụng những kỹ thuật và giải pháp hiệu quả vào dự án của mình.

Cảm ơn anh Linh, đồng nghiệp tại công ty ESC, vì sự hỗ trợ tận tâm và việc cho mượn máy làm việc. Sự thuận tiện này thực sự đã giúp em tiếp cận công nghệ và dữ liệu cần thiết một cách dễ dàng và hiệu quả.

Cuối cùng, xin gửi lời tri ân đến bạn bè, những người đã động viên và hỗ trợ em trong suốt quá trình thực hiện đề án. Cảm ơn mọi người vì tất cả!

Hà Nội, tháng 12 năm 2023

Sinh viên

Trần Xuân Độ

Danh sách hình vẽ

1.1	Minh hoạ vấn đề. (Bên trên) Sử dụng cách biểu diễn dạng hộp, so với cách biểu diễn bao lồi (Bên dưới), hiện tượng răng cưa đã được xử lý	4
1.2	Biểu đồ luồng quy trình thực hiện của bộ phát hiện CFA.	5
1.3	So sánh biểu diễn hộp có hướng (bên trên) so với biểu diễn bao lồi (ở dưới).	6
1.4	So sánh tích chập thông thường và tích chập biến dạng.	6
1.5	Các phép biến đổi tích chập biến dạng khác nhau	7
1.6	Quá trình thực hiện tích chập biến dạng.	7
1.7	Minh hoạ thuật toán Outer Convex Approximation.	8
1.8	Minh hoạ thuật toán Inner Convex Approximation với đầu vào gồm 100 điểm ngẫu nhiên.	9
1.9	Minh hoạ xây dựng tập các bao lồi để biểu diễn các đối tượng, đặc biệt với những đối tượng phân bố dày đặc.	11
1.10	Phân chia tập bao lồi theo hướng dẫn của nguyên tắc nhất quán độ dốc.	13
2.1	Đa giác n điểm ngẫu nhiên có khung 16 cạnh \mathcal{P}^\diamond	20
2.2	Bao lồi kết quả khi chạy với $\delta = 0$	20
2.3	Bao lồi kết quả khi chạy với $\delta = 5$	21
2.4	Bao lồi kết quả khi chạy với $\delta = 10$	21
2.5	Bao lồi kết quả khi chạy với $\delta = 30$	22
2.6	Inner Convex Approximation với $\delta = 0$	26
2.7	Inner Convex Approximation với $\delta = 10$	26

2.8	Inner Convex Approximation với $\delta = 20$	27
3.1	Minh hoạ kết quả hình ảnh output của thư viện Mmrotate.	29
3.2	Minh hoạ nhãn hộp bao có hướng của tập dữ liệu DOTA.	31
4.1	Kết quả của bộ phát hiện CFA thực hiện trên tập dữ liệu DOTA khi so sánh với các bộ phát hiện khác	37
4.2	So sánh hàm loss CIoU và hàm loss Smoothed-L1.	37
4.3	Nghiên cứu cắt bỏ của các module trong CFA. CGen là Convex hull Generation, FA-S là feature adaptation by set splitting, FA- A là Feature Adaptation với anti-aliasing. Mạng backbone được dùng là ResNet50.	38
4.4	Đánh giá tham số và các modules.	39
4.5	Quá trình phát triển của bao lỗi trong quá trình huấn luyện.	39
4.6	Biểu đồ nhiệt khi có thích nghi đặc trưng và không có thích nghi đặc trưng	40
4.7	Outer Convex Approximation sử dụng toàn bộ tập dữ liệu, sập tại epoch số 3.	40
4.8	Outer Convex Approximation chạy với dataset nhỏ, sập tại epoch 20.	41
4.9	Outer Convex Approximation train với dataset nhỏ, sập tại epoch 20.	41
4.10	Outer Convex Approximation chỉ thay code ở hàm Jarvis, đạt được 40 epoch.	42
4.11	Inner Convex Approximation train với toàn bộ dữ liệu, sập tại epoch số 15.	43
4.12	Inner Convex Approximation train với bộ dữ liệu nhỏ hơn, sập tại epoch số 20.	43

Danh sách bảng

2.1	Số cạnh trung bình của đa giác lồi xấp xỉ lồi trong $\mathcal{P}^{\text{outer}}$ trả về bởi thuật toán 1 và số lần thực hiện trung bình của bước 3 khi X gồm n điểm ngẫu nhiên trong đa giác có khung 16 cạnh (Hình 2.1).	19
2.2	Thời gian chạy trung bình $T_{\text{Alg.3}}$ thuật toán 2 khi X gồm n điểm ngẫu nhiên được trình bày trong bảng 2.3 phải được tạo ra trong đa giác có khung 16 cạnh \mathcal{P}^\diamond hiển thị trong hình 2.1.	19
2.3	Số cạnh trung bình của đa giác lồi xấp xỉ lồi trong $\mathcal{P}^{\text{inner}}$ trả về bởi thuật toán 2 và số lần thực hiện trung bình của bước 2 khi X gồm n điểm ngẫu nhiên trong đa giác có khung 16 cạnh \mathcal{P}^\diamond và được hiển thị trong hình 2.1.	25
2.4	Thời gian chạy trung bình $T_{\text{Alg.2}}$ thuật toán 2 khi X gồm n điểm ngẫu nhiên được trình bày trong bảng 2.4 phải được tạo ra trong đa giác có khung 16 cạnh \mathcal{P}^\diamond hiển thị trong hình 2.1.	25

Danh mục các ký hiệu và chữ viết tắt

CFA	Convex-hull Feature Adaptation
CIoU	Convex Intersection over Union
RoI	Region of Interest transformer
Conv	Convolution
DCN	Deformable Convolution Network
<i>loc</i>	localization
<i>cls</i>	classification
FL	Focal Loss
CUDNN	CUDA Deep Neural Network
CUDA	Compute Unified Device Architecture

Mở đầu

Trong nhiều thập kỷ, chúng ta đã chứng kiến quá trình phát triển đáng kể của bài toán nhận diện đối tượng. Đóng góp vào sự phát triển này chính là việc sử dụng mạng học sâu kết hợp với đa dạng các cách biểu diễn đặc trưng, các database có kích thước lớn hơn, và việc đào tạo trước các mô hình để tiết kiệm thời gian và chi phí. Tuy nhiên, phần lớn các bộ phát hiện đối tượng đều gặp phải vấn đề khi biểu diễn các đối tượng quan sát từ trên xuống, có hướng tùy ý, hoặc các đối tượng có bố cục khác nhau trong quá trình đào tạo. Vấn đề trở nên nghiêm trọng hơn khi các đối tượng phân bố dày đặc, gây ra hiện tượng răng cưa ở các vùng giao nhau của trường tiếp nhận.

Một trong những giải pháp để phát hiện đối tượng có hướng, đó là sử dụng phương pháp làm giàu đặc trưng/mở neo, cung cấp nhiều hơn các đặc trưng để đào tạo các bộ phát hiện. Giải pháp này tuy nhiên lại gây ra sự phức tạp trong tính toán, dễ gây ra sai sót. Một giải pháp khác là định nghĩa bộ biến đổi RoI, áp dụng phép biến đổi không gian RoIs trong khi tiếp tục học các tham số dưới sự giám sát của hộp bao có hướng. Các bộ biến đổi này được cho là linh hoạt, nhạy bén, hoạt động mượt mà, cho phép trường tiếp nhận thích nghi với các đối tượng có hướng. Tuy nhiên, vấn đề về cách điều chỉnh lưới đặc trưng cho đối tượng có bố cục bất kỳ vẫn chưa được giải quyết. Đây chính là nguyên nhân gây ra hiện tượng feature aliasing, xảy ra khi các đối tượng phân bố dày đặc trong khung hình.

Do đó, ta đề xuất một cách tiếp cận khác: sử dụng phương pháp điều chỉnh các đặc trưng bằng bao lồi (convex-hull) dành cho các đối tượng có hướng và phân bố dày đặc. Mục tiêu để điều chỉnh các đặc trưng nằm trong lưới tích chập thông thường với các đối tượng có bố cục phân bố không đều. Ta xây dựng bộ

cục đối tượng thành một bao lồi, có lợi thế hơn so với sử dụng bố cục hình chữ nhật, giúp bao phủ toàn bộ đối tượng nhưng giảm thiểu tối đa diện tích vùng nền của đối tượng. Mỗi bao lồi là tập hợp các điểm đặc trưng định nghĩa đường biên của đối tượng, biểu thị tỷ lệ Convex Intersection over Union (CIoU) để các định vị trí đối tượng. Bên trong bao lồi, các đặc trưng khác nhau biểu thị cho sự xuất hiện của các đối tượng được phân loại khác nhau.

Mục tiêu của đề án là tìm hiểu, nghiên cứu phương pháp phát hiện đối tượng dày đặc có hướng, cụ thể là nghiên cứu phương pháp BeyondBoundingBox dành cho vấn đề trên. Ngoài ra, đề án còn tìm hiểu thuật toán phát hiện bao lồi mới, thay thế vào phương pháp trên, kiểm nghiệm và đo lường độ hiệu quả của thuật toán mới này.

Trong đề án em sẽ tập trung trình bày một số nội dung chính như sau:

Chương 1: Giới thiệu bài toán phát hiện, định hướng và đóng gói đối tượng: Nội dung chương 1 khái quát các vấn đề và phương pháp nhận dạng đối tượng, trình bày về các phương pháp liên quan, nguyên lý và cách thức triển khai, giới thiệu sử dụng thuật toán bao lồi xấp xỉ để thực hiện bài toán.

Chương 2: Trình bày thuật toán bao lồi xấp xỉ: Nội dung của chương 2 giới thiệu thuật toán Outer Convex Approximation, Inner Convex Approximation, lý thuyết và một số kết quả của thuật toán.

Chương 3: Thực nghiệm và kết quả: Nội dung của chương 3 áp dụng bao lồi xấp xỉ cho bài toán phát hiện, định hướng và đóng gói đối tượng, cách thức triển khai bộ phát hiện, cách thức thay thế thuật toán.

Chương 4: Tổng kết: Nội dung chương 4 là tổng kết bài toán, tóm tắt những kết quả đã đạt được và còn chưa đạt được. Từ đó đề xuất mục tiêu hướng tới cũng như hướng nghiên cứu, phát triển tiếp theo.

Chương 1

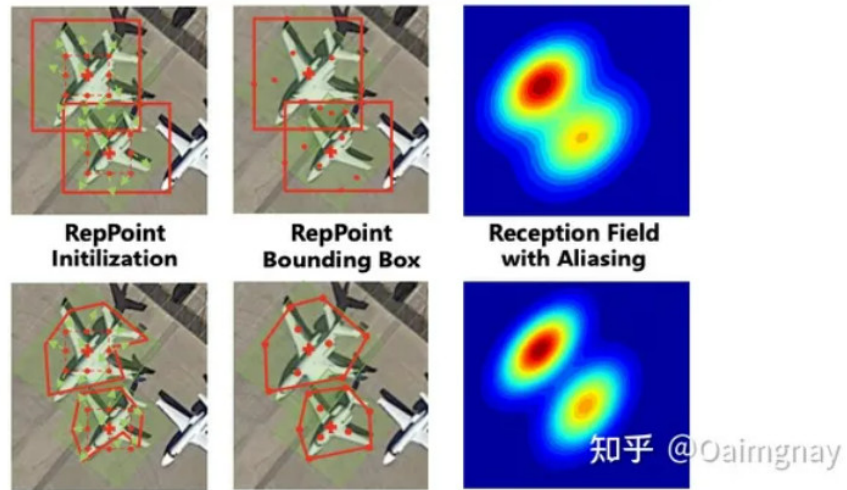
Giới thiệu bài toán phát hiện, định hướng và đóng gói đối tượng

Chương 1 đặt vấn đề về bài toán phát hiện đối tượng, trình bày lý thuyết về phương pháp BeyondBoundingBox.

1.1 Giới thiệu

Trong bài toán phát hiện đối tượng, việc định vị và phát hiện đối tượng dày đặc vẫn còn là một vấn đề thách thức do vấn đề đặc trưng răng cưa (feature aliasing), tức là các góc của bounding box sẽ bị chồng lên nhau. Đã có các giải pháp được sử dụng như: làm giàu các đặc trưng (enhance features), hay sử dụng anchors. Nhược điểm chung của các phương pháp này là: làm cho cấu trúc mạng trở nên phức tạp hơn, tăng thời gian huấn luyện (training) và suy luận (inference). Một phương pháp khác đó là sử dụng biến đổi RoI, tuy nhiên phương pháp này không thích ứng tốt với các đối tượng có hướng ngẫu nhiên, đặc biệt đối với các đối tượng xuất hiện dày đặc trong hình ảnh.

Để giải quyết các vấn đề trên, một phương pháp đã được đưa ra đó là: phương pháp thích ứng bao lồi (convex-hull feature adaptation - CFA), hay *bộ phát hiện CFA*. CFA được thực hiện dựa trên nguyên tắc biểu diễn bao lồi, định nghĩa một tập các điểm đặc trưng, giới hạn phạm vi của đối tượng mục tiêu nhờ sử dụng chỉ số CIoU. Bộ phát hiện này đạt được sự phân bố đặc trưng tối ưu nhờ vào việc xây dựng tập bao lồi, cũng như phân chia linh hoạt các bao lồi thành các bao lồi âm và bao lồi dương. Ngoài ra, CFA xem xét sự chồng chéo nhau giữa



Hình 1.1: Minh hoạ vấn đề. (Bên trên) Sử dụng cách biểu diễn dạng hộp, so với cách biểu diễn bao lồi (Bên dưới), hiện tượng răng cưa đã được xử lý

bao lồi dự đoán và bao lồi thực tế, tiến hành phạt các bao lồi được chia sẻ bởi nhiều đối tượng, từ đó giúp giảm thiểu hiện tượng đặc trưng răng cưa (feature aliasing), đạt được sự thích ứng đặc trưng tối ưu. Bộ phát hiện đã đạt được kết quả tốt nhất khi thử nghiệm trên tập dữ liệu DOTA và SKUR110KR.

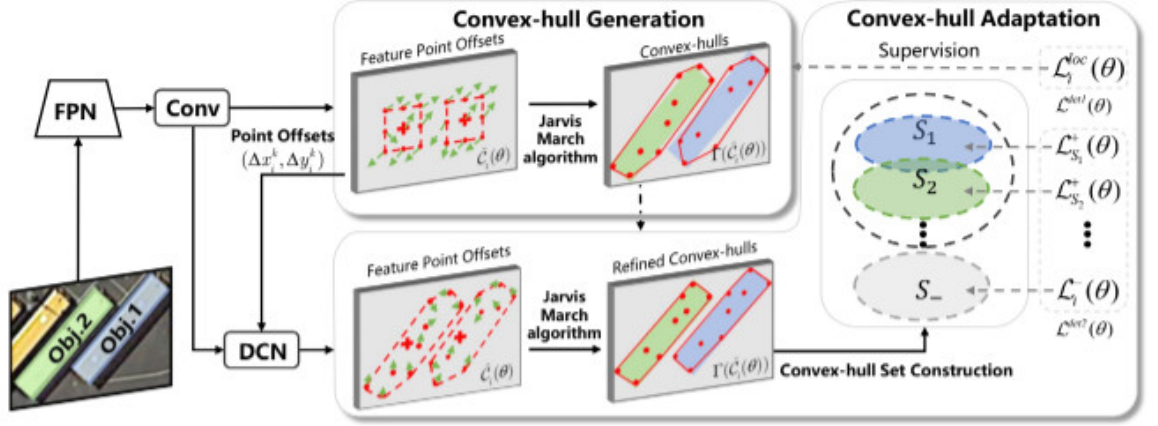
Phương pháp CFA được chia làm 2 giai đoạn thực hiện:

- Giai đoạn 1: tạo tập bao lồi, ước lượng sơ bộ bố cục của bao lồi.
- Giai đoạn 2: chỉnh sửa bao lồi để khớp với các đối tượng phân bố dày đặc.

1.2 Xây dựng tập bao lồi đầu tiên

Thông thường, các hộp bao (bounding-box) dùng cho việc phát hiện đối tượng sử dụng hình chữ nhật để biểu diễn. Điều này đôi khi làm giảm khả năng biểu diễn đối tượng, do không phải đối tượng nào cũng có dạng hình chữ nhật. Chính vì thế, phương pháp CFA đã đề xuất biểu diễn phạm vi của đối tượng bằng bao lồi, giúp cho việc phát hiện đối tượng được hiệu quả hơn.

Mỗi bao lồi (convex-hull) là một tập hợp các điểm thoả mãn công thức:



Hình 1.2: Biểu đồ luồng quy trình thực hiện của bộ phát hiện CFA.

$$C_i = \{(x_i^k, y_i^k)\}_{i=1 \dots K} \quad (1.1)$$

Trong đó:

C_i : bao lồi thứ i ,

(x_i^k, y_i^k) : điểm nằm trên bao lồi thứ k , với k là chỉ số của điểm đặc trưng,

$K = 9$: tương ứng với 9 điểm được khởi tạo từ đầu cho mỗi bao lồi.

Việc huấn luyện có thể xem như là quá trình dự đoán độ lệch (offset), trong khi hệ số CIoU cần được tối đa hoá để đạt được so khớp tối ưu nhất.

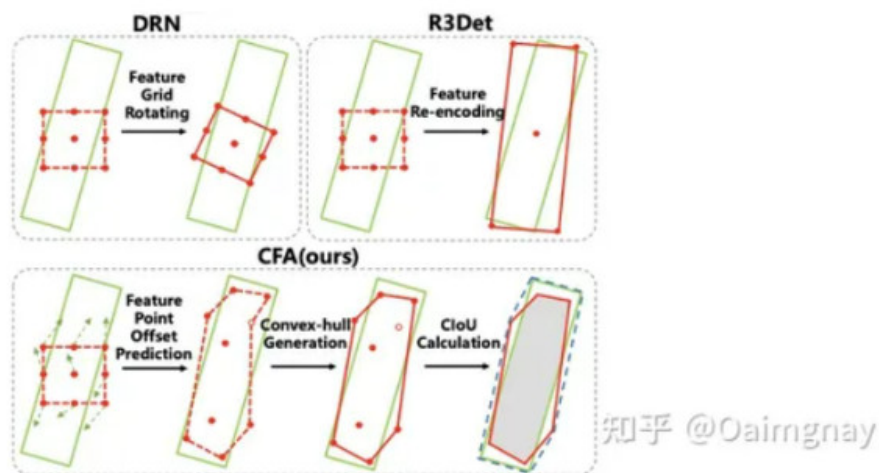
Dưới đây là phương pháp sử dụng phép toán tích chập để dự đoán độ lệch: $(\Delta x_i^k, \Delta y_i^k)$ với từng điểm đặc trưng, trả về một bản đồ phân bù các đặc trưng $O \in R^{H \times W \times W}$ (H, W, C lần lượt là chiều dài, chiều rộng và số kênh của bản đồ đặc trưng):

$$\hat{C}_i(\theta) \leftarrow \{(x_i^k + \Delta x_i^k(\theta), y_i^k + \Delta y_i^k(\theta))\}_{i=1 \dots K} \quad (1.2)$$

θ được ký hiệu là các tham số của mạng.

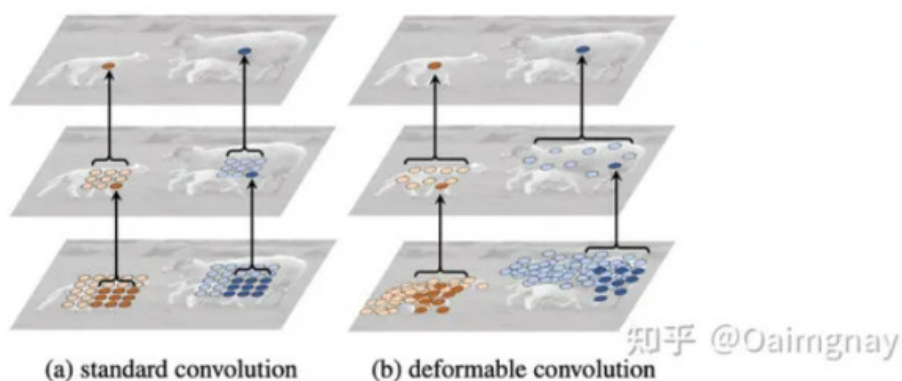
1.3 Tích chập biến dạng

Tích chập biến dạng (Deformable convolution - DCN) [3] là dạng tích chập mà vị trí thực hiện tích chập bị biến dạng, không giống tích chập truyền thống



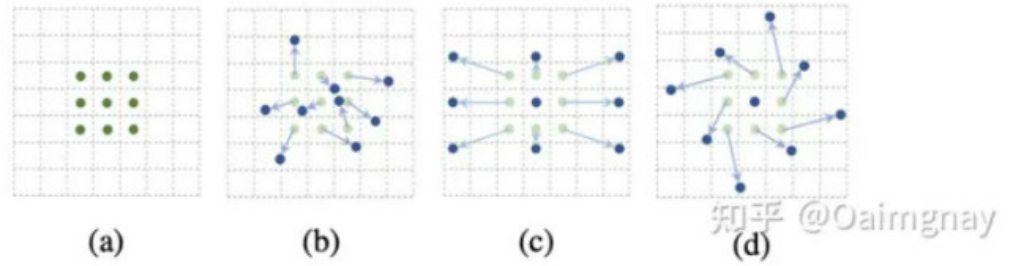
Hình 1.3: So sánh biểu diễn hộp có hướng (bên trên) so với biểu diễn bao lồi (ở dưới).

là dạng lưới $N \times N$. Ưu điểm của phương pháp này giúp trích xuất các đặc trưng mong muốn được chính xác hơn, lấy mẫu được ở những vị trí đa dạng hơn (Phép tích chập truyền thống chỉ có thể trích xuất các đặc trưng trên một khung hình chữ nhật). (Hình 1.4)

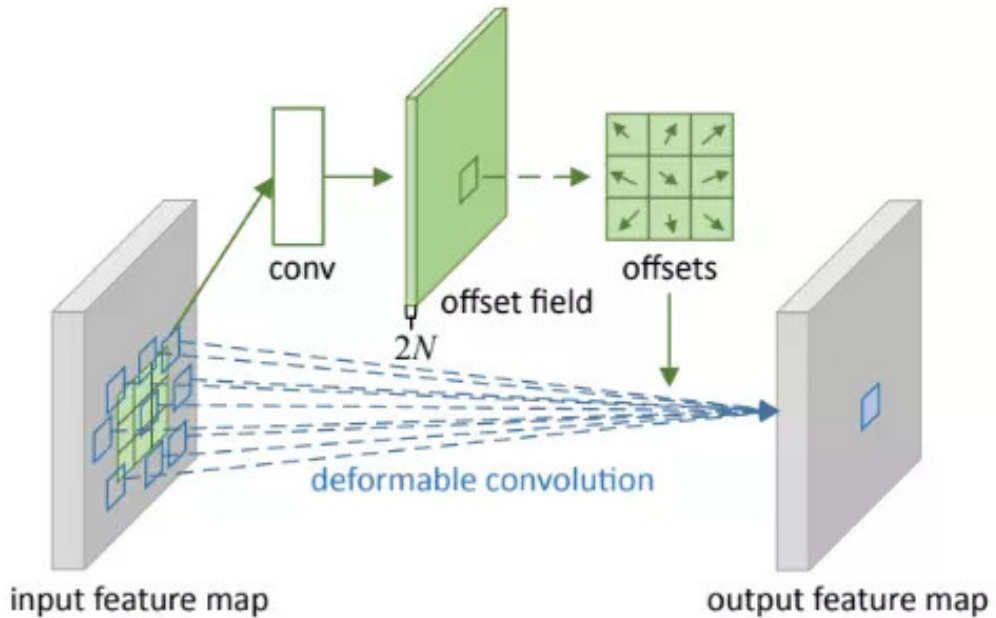


Hình 1.4: So sánh tích chập thông thường và tích chập biến dạng.

Phép tích chập biến dạng thực ra là thêm phần bù cho các điểm tích chập lấy mẫu. (Hình 1.5)



Hình 1.5: Các phép biến đổi tích chập biến dạng khác nhau



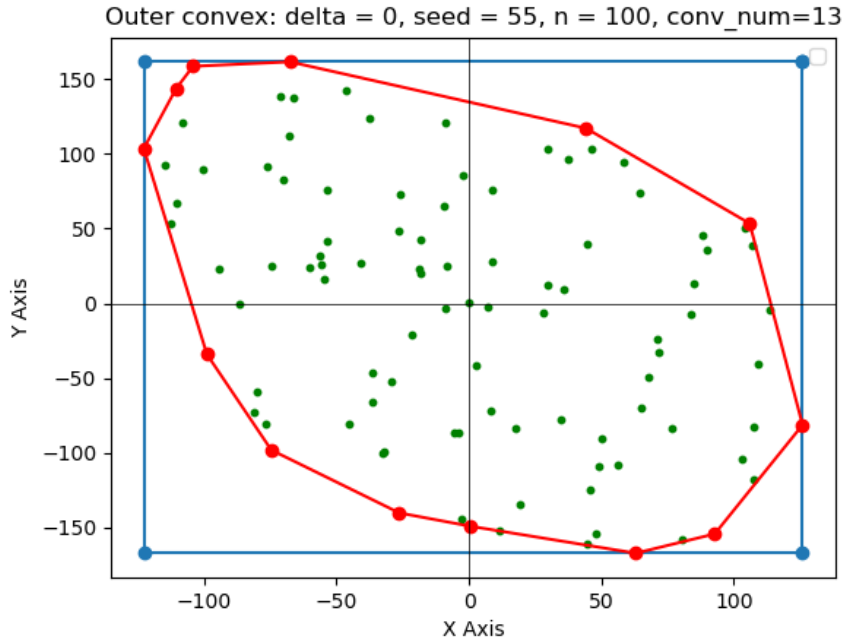
Hình 1.6: Quá trình thực hiện tích chập biến dạng.

Cho đầu vào là một bản đồ đặc trưng, giả sử phép tích chập là 3×3 . Để học được phần bù, định nghĩa một lớp tích chập 3×3 khác, chiều của đầu ra là kích thước của bản đồ đặc trưng ban đầu, số kênh $= 2N$. (Hình 1.6) Tiếp theo thực hiện tích chập biến dạng, dựa trên độ bù của các phần đã được tính trước đó, sau đó thực hiện phép tích chập như thông thường.

1.4 Thuật toán Convex Approximation

Sau khi học được phần bù, CFA cập nhật các điểm đặc trưng của bao lồi, được thực hiện bởi thuật toán Outer Convex Approximation, hoặc Inner Convex Approximation (cả hai thuật toán đều sử dụng $\delta = 0$, không lấy bao lồi xấp xỉ). Kết quả là bao lồi nhỏ nhất phù hợp điều kiện sẽ được chọn ra sau mỗi lần lặp.

1.4.1 Thuật toán Outer Convex Approximation

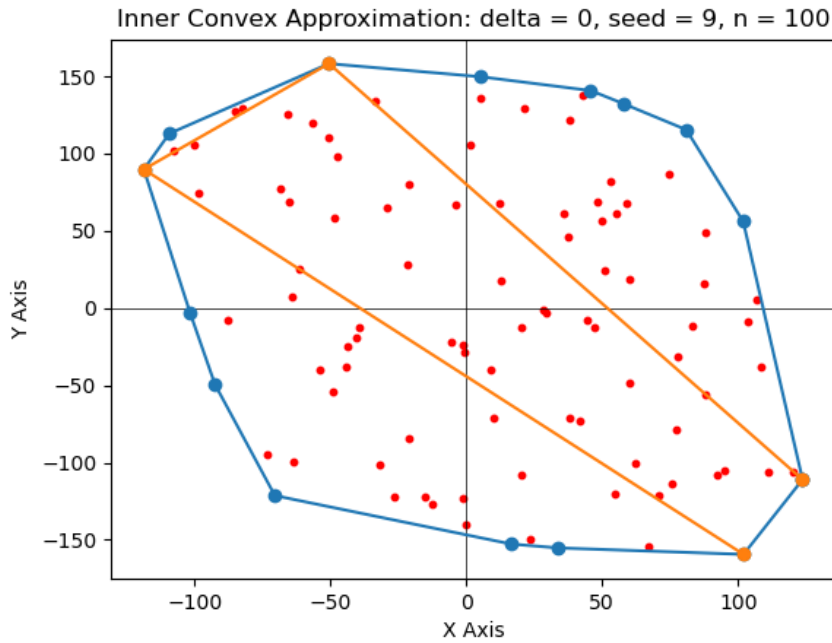


Hình 1.7: Minh họa thuật toán Outer Convex Approximation.

Tóm tắt thuật toán: Xuất phát từ một hình chữ nhật nhỏ nhất bao quanh tất cả các điểm (màu xanh dương), duyệt lần lượt các đỉnh được chọn, xác định hướng cực đại (direction) nhờ hai điểm liền trước (predecessor) và điểm liền sau (successor), tính toán theo công thức để quyết định việc lấy thêm đỉnh mới cho bao lồi hay không. Kết quả cuối cùng sẽ là hình chữ nhật được cải thiện dần thành bao lồi (màu đỏ) phù hợp với toàn bộ tập điểm (Hình 1.7).

1.4.2 Inner Convex Approximation

Tóm tắt thuật toán): Xuất phát từ hình tứ giác ban đầu (màu cam), xét mỗi cạnh của đa giác, chọn các điểm phù hợp trong tập điểm nằm phía trên cạnh, từ đó xem xét có nên thêm cạnh mới hay không, nhằm mở rộng tứ giác ban đầu. Sau cùng khi không còn điểm nào nằm phía trên các cạnh, thuật toán sẽ kết thúc và trả về bao lồi kết quả(màu xanh lam). (Hình 1.8).



Hình 1.8: Minh họa thuật toán Inner Convex Approximation với đầu vào gồm 100 điểm ngẫu nhiên.

1.5 Định nghĩa công thức Convex Intersection over Union (CIoU)

Dựa vào mỗi bao lồi dự đoán, ta tính toán được hàm mất mát vị trí (localization) và phân lớp (classification) của một đối tượng. Công thức CIoU giữa bao lồi dự đoán thứ i : $C_i(\theta)$ và bao lồi thực tế \mathcal{B}_j của đối tượng thứ j được tính như sau:

$$\text{CIoU}(\mathcal{C}_i(\theta), \mathcal{B}_j) = \frac{|\mathcal{C}_i(\theta) \cap \mathcal{B}_j|}{|\mathcal{C}_i(\theta) \cup \mathcal{B}_j|} - \frac{|\mathcal{R}_j \setminus (\mathcal{C}_i(\theta) \cup \mathcal{B}_j)|}{|\mathcal{R}_j|} \quad (1.3)$$

Trong đó, \mathcal{R}_j là đa giác nhỏ nhất có thể bao quanh $\mathcal{C}_i(\theta)$ và \mathcal{B}_j .

1.6 Hàm mất mát

Theo công thức (1.3), hàm mất mát vị trí CIoU được định nghĩa là:

$$\mathcal{L}_i^{loc}(\theta) = 1 - \text{CIoU}(\mathcal{C}_i(\theta), \mathcal{B}_j) \quad (1.4)$$

Cho $f_i^k(\theta)$ là đặc trưng của điểm thứ k , bao lồi đặc trưng $f_i(\theta)$ được tính bởi tổng có trọng số của tất cả các điểm đặc trưng trên bao lồi dự đoán $\mathcal{C}_i(\theta)$, tức là bằng công thức: $f_i(\theta) = \sum_k m w_i^k \cdot f_i^k(\theta)$, trong đó, w_i^k biểu thị các trọng số đặc trưng có thể học được từ tích chập biến dạng (DCN). Dựa vào bao lồi đặc trưng, điểm dự đoán $\mathcal{S}_i(\theta)$ của bao lồi dự đoán $\mathcal{C}_i(\theta)$ được tính bởi phép tích chập, hàm mất mát phân loại của bao lồi dự đoán $\mathcal{C}_i(\theta)$ tương ứng với \mathcal{B}_j được định nghĩa là:

$$\mathcal{L}_i^{cls}(\theta) = \text{FL}(\mathcal{S}_i(\theta), Y_j) \quad (1.5)$$

Trong đó, Y_j biểu thị nhãn nhị phân thật sự (ground-truth) và $\text{FL}()$ là hàm mất mát Focal (Focal loss). Kết quả thu được là hàm mất mát của bao lồi dương:

$$\mathcal{L}_i^+(\theta) = \mathcal{L}_i^{cls}(\mathcal{S}_i(\theta), Y_j) + \lambda \mathcal{L}_i^{loc}(\mathcal{C}_i(\theta), \mathcal{B}_j) \quad (1.6)$$

Hàm mất mát (1.6) là tổng của hàm mất mát vị trí (1.4) và hàm mất mát phân loại (1.5). Tương tự ta có hàm mất mát dành cho bao lồi âm:

$$\mathcal{L}_i^-(\theta) = \mathcal{L}_i^{cls}(\mathcal{S}_i(\theta), Y_j) \quad (1.7)$$

Trong quá trình huấn luyện, vì các bao lồi được ban đầu chỉ được sinh ra bằng cách tối ưu CIoU, nên cần định nghĩa một hàm loss cho việc giám sát:

$$\mathcal{L}^{\text{det}1}(\theta) = \frac{1}{J} \sum_i \mathbb{I}_{(x_i, y_i)} \mathcal{L}_i^{loc}(\theta) \quad (1.8)$$

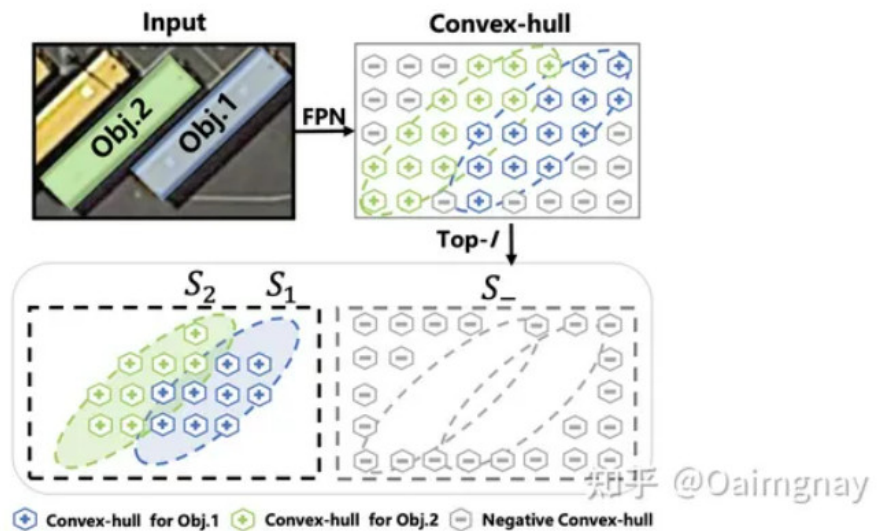
Nói tóm lại, trong giai đoạn đầu tiên của việc sinh bao lồi, hàm mất mát L^{det} bên trên là hàm cần được xem xét. Trong giai đoạn 2, hai hàm mất mát phân lớp (1.7) và (1.6) sẽ được sử dụng để phân loại bao lồi.

1.7 Thích ứng bao lồi

Phương pháp biểu diễn bằng bao lồi giúp định vị đối tượng ở bất kỳ hình dạng nào. Tuy nhiên vẫn xảy ra một vấn đề, đó là làm thế nào để định vị một cách chính xác các đối tượng dày đặc, đặc biệt là các đối tượng có đặc trưng rỗng cửa (feature aliasing). Chính vì thế, bộ phát hiện CFA đã đề xuất một phương pháp thích ứng mới, giúp tinh chỉnh các bao lồi trong ở giai đoạn 1 để thu được vị trí chính xác hơn và phân lớp hiệu quả hơn.

1.7.1 Xây dựng tập các bao lồi

CFA xây dựng một tập các bao lồi với mỗi đối tượng, để một đối tượng mục tiêu có thể khớp với nhiều bao lồi phù hợp, từ đó cùng nhau tối ưu các đặc trưng của các đối tượng dày đặc.



Hình 1.9: Minh họa xây dựng tập các bao lồi để biểu diễn các đối tượng, đặc biệt với những đối tượng phân bố dày đặc.

Với mỗi một đối tượng, việc xây dựng tập bao lỗi tương ứng là: xem xét hệ số CIoU giữa bao lỗi dự đoán và bao lỗi thực tế, chọn ra top- I bao lỗi làm bao lỗi dương để xây dựng tập các bao lỗi. Các bao lỗi còn lại không thuộc vào bất kỳ tập bao lỗi nào sẽ được gộp lại thành tập bao lỗi âm S .

$$\mathcal{L}_{S_j}^+(\theta) = \frac{1}{|S_j|} \sum_{i \in S_j} \omega_i \mathcal{L}_i^+(\theta) \quad (1.9)$$

Khi nhiều đối tượng trong ảnh nằm gần nhau, không phải tất cả các bao lỗi nằm trong tập bao lỗi đều phù hợp với đối tượng. Các bao lỗi có đặc trưng rằng cửa sẽ phải được phân loại thành tập các bao lỗi âm. Các bao lỗi được chia sẻ bởi nhiều đối tượng thì sẽ có độ tin cậy thấp hơn các bao lỗi khác.

1.7.2 Chiến lược phân đoạn tập các bao lỗi

Để giải quyết vấn đề đặc trưng rằng cửa ở các đối tượng dày đặc, bộ phát hiện CFA đề xuất chiến lược phân đoạn tập các bao lỗi để đánh giá động các mẫu bao lỗi âm và mẫu dương, chuyển đổi trọng số ω_i thành $f(L_i^+(\theta))$. Sau khi thay thế, được công thức sau:

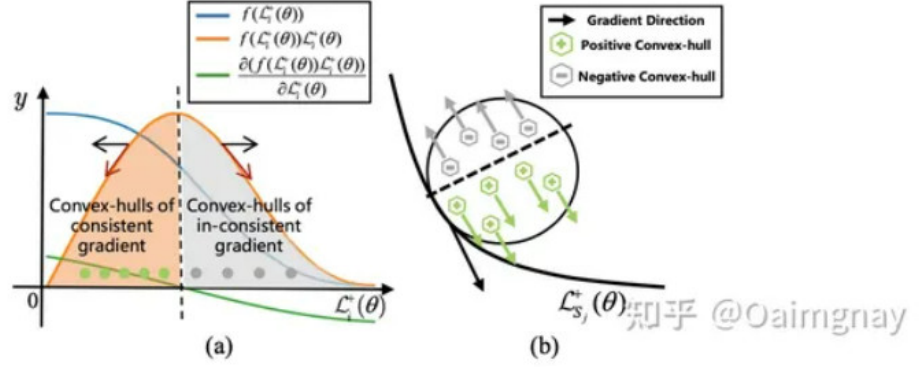
$$\mathcal{L}_{s_j}^+(\theta) = \frac{1}{|S_j|} \sum_{i \in S_j} f(L_i^+(\theta)) \mathcal{L}_i^+(\theta) \quad (1.10)$$

Trong đó: f là hàm lỗi đơn điệu giảm phân phối Gaussian: $f(x) = 1.0 - \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$, có nghĩa là giá trị mất càng nhỏ, độ tin cậy càng cao.

Nguyên tắc phân chia tập bao lỗi là nguyên tắc nhất quá độ dốc. Bằng cách lấy đạo hàm của công thức công thức 1.10, ta có được:

$$\frac{\partial \mathcal{L}_{s_j}^+(\theta)}{\partial \theta} = \frac{1}{|S_j|} \sum_{i \in S_j} \frac{\partial (f(L_i^+(\theta)) \mathcal{L}_i^+(\theta))}{\partial \mathcal{L}_i^+(\theta)} \frac{\partial \mathcal{L}_i^+(\theta)}{\partial \theta} \quad (1.11)$$

Đạo hàm của mỗi một bao lỗi dương $\frac{\partial \mathcal{L}_i^+(\theta)}{\partial \theta}$ yêu cầu đạo hàm của toàn bộ tập bao lỗi $\frac{\partial \mathcal{L}_{s_j}^+(\theta)}{\partial \theta}$ là nhất quán, nghĩa là: bao lỗi nào có độ dốc không nhất quán được xem là bao lỗi âm. Những bao lỗi này sẽ dẫn đến hiện tượng rằng cửa đặc trưng. Xét công thức (1.11), nếu $\frac{\partial (f(L_i^+(\theta)) \mathcal{L}_i^+(\theta))}{\partial \mathcal{L}_i^+(\theta)}$ mang giá trị dương, thì bao lỗi C_i được xếp là bao lỗi dương, hoặc sẽ là bao lỗi âm. Xem xét hình 1.10, khi sắp



Hình 1.10: Phân chia tập bao lỗi theo hướng dẫn của nguyên tắc nhất quán độ dốc.

xếp các giá trị mất mát $\frac{\partial L_i^+(\theta)}{\partial \theta}$ theo thứ tự tăng dần, $f(\frac{\partial L_i^+(\theta)}{\partial \theta}) L_i^+(\theta)$ (đường màu cam) là một hàm lỗi hướng lên với một cực trị duy nhất, trong khi đường $\frac{\partial(f(L_i^+(\theta))L_i^+(\theta))}{\partial L_i^+(\theta)}$ (màu xanh lục) chia các bao lỗi thành tập các bao lỗi dương S_j và tập bao lỗi âm S_- .

Cùng lúc đó, để xử lý đặc trưng răng cưa, CFA cũng giới thiệu hệ số chống đặc trưng răng cưa:

$$p_i = \gamma \frac{\text{CIoU}(\mathcal{C}_i, \mathcal{B}_j)}{\sum_{m=1}^M \text{CIoU}(\mathcal{C}_i, \mathcal{B}_j)} \quad (1.12)$$

Hệ số này thể hiện mức độ mà một đối tượng thuộc về một đối tượng duy nhất, khi nó chồng lên M đối tượng khác. γ là hệ số chống đặc trưng răng cưa. Hàm mất mát được cập nhật thành:

$$\mathcal{L}_{s_j}^+(\theta) = \frac{1}{|S_j|} \sum_{i \in S_j} p_i f(\mathcal{L}_i^+(\theta)) \mathcal{L}_i^+(\theta) \quad (1.13)$$

Nói tóm lại, giai đoạn 2 của quá trình tối ưu được điều khiển bởi hàm mất mát trên tập bao lỗi, được xác định bằng cách kết hợp hàm mất mát phân loại và hàm mất mát vị trí:

$$\begin{aligned} \mathcal{L}^{\text{det}2}(\theta) = & \frac{1}{J} \sum_{j=1}^J \frac{1}{|S_j|} \sum_{i \in S_j} p_i f(\mathcal{L}_i^+(\theta)) \mathcal{L}_i^+(\theta) \\ & + \frac{1}{|S_-|} \sum_{i \in S_-} \mathcal{L}_i^-(\theta) \end{aligned} \quad (1.14)$$

Hàm mất mát này xem xét sự tương ứng về đặc trưng của nhiều đối tượng, tiến hành phạt các bao lỗi được chia sẻ bởi nhiều đối tượng, và giảm thiểu đặc trưng riêng của để đạt được thích ứng đặc trưng tối ưu. Cuối cùng hàm mất mát của toàn bộ bộ phát hiện CFA là:

$$L^{\text{det } 1}(\theta) + L^{\text{det } 2}(\theta) \tag{1.15}$$

Chương 2

Thuật toán tính bao lồi xấp xỉ

Thuật toán tìm bao lồi xấp xỉ nhận đầu vào là một tập các điểm ngẫu nhiên, đầu ra trả về một tập điểm biểu diễn một bao lồi bao quanh tất cả các điểm còn lại. Với ngưỡng δ tùy chỉnh, sẽ cho ra được dạng bao lồi khác nhau.

2.1 Outer convex approximation

Cho tập X được định nghĩa như sau:

$$X := \{x_1, x_2, \dots, x_n\} \subset \mathbb{R}^2 \quad (2.1)$$

Giả sử không mất tính tổng quát rằng:

$$x_1, x_2, \dots, x_n \text{ không cùng nằm trên cùng một đường thẳng.} \quad (2.2)$$

Ta viết tất cả các vector thành dạng vector hàng, những vector này sẽ có chuyển vị của chúng được ký hiệu bởi chỉ số trên T , và sử dụng chỉ số trên để chỉ định các thành phần của chúng, ví dụ: $x = (x^1, x^2) \in \mathbb{R}^2$. Cho $x, x' \in \mathbb{R}^2$, chứng tỏ:

$$\begin{aligned} [x, x'] &:= \{(1 - \lambda)x + \lambda x' \mid \lambda \in [0, 1]\} \\ (x, x') &:= \{(1 - \lambda)x + \lambda x' \mid \lambda \in (0, 1)\} \end{aligned} \quad (2.3)$$

Cho X thỏa mãn (2.1) - (2.2) và $\delta \geq 0$, trong phần này ta muốn tìm một bao lồi xấp xỉ của X , ký hiệu:

$$\text{Đa giác lồi } \mathcal{P}^{outer} \text{ thỏa mãn bao lồi } X \subset \mathcal{P}^{outer} \quad (2.4)$$

sao cho:

$$\text{dist}_H(\text{conv } X, \mathcal{P}^{outer}) \leq \delta \quad (2.5)$$

\mathcal{P}^{outer} được xác định bởi:

$$\mathcal{P}^{outer} := \{x \in \mathbb{R}^2 \mid dx^T \leq \beta_d \text{ với tất cả } d \in D\} \quad (2.6)$$

Trong đó $D \subset \mathbb{R}^2$ biểu thị tập các hướng tối đa, $\beta_d \in \mathbb{R}$ biểu thị ngưỡng tương ứng với hướng $d \in D$. Với tập D cho trước, \mathcal{P}^{outer} là bao lồi xấp xỉ phù hợp nhất chứa X nếu như:

$$\beta_d := \max_{x \in X} dx^T \text{ với tất cả } d \in D \quad (2.7)$$

Cho P là tập các đỉnh của \mathcal{P}^{outer} .

Ta bắt đầu quá trình xác định bao lồi xấp xỉ ngoài với hình chữ nhật nhỏ nhất chứa X , có chứa cạnh song song với trục tung và trục hoành. Theo công thức (2.5) - (2.6), hình chữ nhật \mathcal{P}^{outer} được xác định như sau:

$$D := \{(1, 0), (0, 1), (-1, 0), (0, -1)\} \quad (2.8)$$

và:

$$\begin{aligned} \beta_{(1,0)} &:= \max \{x^1 \mid (x^1, x^2) \in X\}, \\ \beta_{(0,1)} &:= \max \{x^2 \mid (x^1, x^2) \in X\}, \\ \beta_{(-1,0)} &:= \max \{-x^1 \mid (x^1, x^2) \in X\}, \\ \beta_{(0,-1)} &:= \max \{-x^2 \mid (x^1, x^2) \in X\}. \end{aligned} \quad (2.9)$$

Theo công thức (2.2) ta có:

$$\beta_{(-1,0)} < \beta_{(1,0)} \quad \text{và} \quad \beta_{(0,-1)} < \beta_{(0,1)}$$

Vì vậy, \mathcal{P}^{outer} là một hình chữ nhật phù hợp với 4 đỉnh phân biệt, có tập đỉnh là:

$$P := \{r_1, r_2, r_3, r_4\} \quad (2.10)$$

Trong đó:

$$\begin{aligned} r_1 &:= (\beta_{(1,0)}, \beta_{(0,1)}), \\ r_2 &:= (\beta_{(-1,0)}, \beta_{(0,1)}), \\ r_3 &:= (\beta_{(-1,0)}, \beta_{(0,-1)}), \\ r_4 &:= (\beta_{(1,0)}, \beta_{(0,-1)}). \end{aligned} \quad (2.11)$$

Trong các bước xấp xỉ tiếp theo, xây dựng đa giác \mathcal{P}^{outer} được cải thiện dần dần như sau:

Với mỗi $p \in P$, lấy $p^- \in P$ và $p^+ \in P$ lần lượt là

điểm liền trước ngược chiều kim đồng hồ và điểm liền sau của p . (2.12)

Xét công thức sau:

$$\begin{aligned} d_p^T &:= \|p^+ - p^-\|^{-1} R(p^+ - p^-)^T, \\ \beta_{d_p} &:= \max\{d_p x^T \mid x \in X\}, \end{aligned} \quad (2.13)$$

Trong đó:

$$R := \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \quad (2.14)$$

là ma trận xoay ngược chiều theo hướng kim đồng hồ với góc xoay $\pi/2$. Vì R là ma trận xoay, ta có:

$$\|d_p\| = \|p^+ - p^-\|^{-1} \|(p^+ - p^-)R^T\| = \|p^+ - p^-\|^{-1} \|p^+ - p^-\| = 1. \quad (2.15)$$

Có hai trường hợp xảy ra khi ta thêm các ràng buộc tuyến tính sau vào định nghĩa của \mathcal{P}^{outer} trong công thức (2.6):

$$d_p x^T \leq \beta_{d_p}. \quad (2.16)$$

Trường hợp 1, nếu:

$$\beta_{d_p} = d_p p^+ \quad (2.17)$$

thì công thức (2.15) sẽ không tạo thêm đỉnh mới mà thêm cạnh mới $[p^-, p^+]$ vào \mathcal{P}^{outer} . Xét $d_{[p^-, p]}$ và $d_{[p, p^+]}$ là 2 hướng cực đại trong D , định nghĩa hai cạnh $[p^-, p]$ và $[p, p^+]$ của \mathcal{P}^{outer} . Sau đó $d_{[p^-, p]}$ and $d_{[p, p^+]}$ sẽ trở nên vô dụng. Vậy nên khi thêm d_p vào tập D cần phải loại bỏ $d_{[p^-, p]}$ và đỉnh $d_{[p, p^+]}$ và p trong P :

$$\begin{aligned} D &:= (D \cup \{d_p\}) \setminus \{d_{[p^-, p]}, d_{[p, p^+]}\}, \\ P &:= P \setminus \{p\}. \end{aligned} \quad (2.18)$$

Trường hợp 2, nếu

$$\beta_{d_p} > d_p p^+ \quad (2.19)$$

và:

$$d_p p^T - \beta_{d_p} > \delta \quad (2.20)$$

thì ràng buộc (2.16) tạo ra hai đỉnh mới của $\mathcal{P}^{\text{outer}}$ có tên \hat{p}^- và \hat{p}^+ , được tính như sau:

$$\begin{aligned}\lambda_p &:= (\beta_{d_p} - d_p p^{-T}) / (d_p p^T - d_p p^{-T}) \in (0, 1), \\ \hat{p}^- &:= (1 - \lambda_p) p^{-T} + \lambda_p p^T, \\ \hat{p}^+ &:= (1 - \lambda_p) p^{+T} + \lambda_p p^T.\end{aligned}\tag{2.21}$$

Ta thêm d_p vào D và thay $p \in P$ bởi \hat{p}^- và \hat{p}^+ :

$$\begin{aligned}D &:= D \cup \{d_p\}, \\ P &:= (P \setminus \{p\}) \cup \{\hat{p}^-, \hat{p}^+\}.\end{aligned}\tag{2.22}$$

Quy trình thực hiện được mô tả bên dưới, trong đó P_{doubt} biểu thị tập hợp các đỉnh vẫn cần được kiểm tra.

Thuật toán 1

Input: Tập hữu hạn $X \subset \mathbb{R}^2$ và tham số xấp xỉ $\delta \geq 0$.

Output: Đa giác xấp xỉ lồi $\mathcal{P}^{\text{outer}}$ được xác định ở công thức (2.6) bởi D và β_d cho $d \in D$ và tập đỉnh P .

1. Xác định D , β_d với $d \in D$, và P theo (2.8)–(2.11).
2. Đặt $P_{\text{doubt}} := P$.
3. Chọn một đỉnh bất kỳ $p \in P_{\text{doubt}}$, tính toán d_p , β_{d_p} theo (2.12)–(2.14).
Nếu (2.17) là đúng thì thay đổi D , P như (2.18), thay đổi P_{doubt} thành:

$$P_{\text{doubt}} := P_{\text{doubt}} \setminus \{p, p^-, p^+\},\tag{2.23}$$

sau đó chuyển sang bước 4.

Nếu (2.20) là đúng, thay đổi D , P như (2.22), cập nhật:

$$P_{\text{doubt}} := (P_{\text{doubt}} \setminus \{p\}) \cup \{\hat{p}^-, \hat{p}^+\},\tag{2.24}$$

sau đó chuyển sang bước 4.

Nếu hai trường hợp trên không đúng thì,

$$P_{\text{doubt}} := P_{\text{doubt}} \setminus \{p\}.\tag{2.25}$$

4. Nếu $P_{\text{doubt}} \neq \emptyset$ thì quay lại bước 3.
 5. Kết quả trả về tập hợp các hướng cực đại D và β_d với $d \in D$, tập đỉnh P của $\mathcal{P}^{\text{outer}}$, kết thúc thuật toán.
-

Bảng 2.2 cho thấy một số kết quả thử nghiệm, trong đó:

- $\#\text{Vertices@Alg. 1}$ là số cạnh trung bình của đa giác xấp xỉ lồi bao ngoài $\mathcal{P}^{\text{outer}}$ được trả về bởi thuật toán 1,

- $\#_{\text{Step III@Alg. 1}}$ là số lần thực hiện trung bình của bước III của thuật toán 1.

$\#X = n$		50	500	1000	2500	5000	7500
$\delta = 70$	$\#_{\text{Edges@Alg. 1}}$	5	5	5	5	5	5
	$\#_{\text{Step III@Alg. 1}}$	6	6	6	6	6	6
$\delta = 10$	$\#_{\text{Edges@Alg. 1}}$	11	11	14	13	11	11
	$\#_{\text{Step III@Alg. 1}}$	18	18	24	22	18	18
$\delta = 1$	$\#_{\text{Edges@Alg. 1}}$	15	33	36	33	32	33
	$\#_{\text{Step III@Alg. 1}}$	38	70	70	62	60	64
$\delta = 0$	$\#_{\text{Edges@Alg. 1}}$	12	23	30	38	43	43
	$\#_{\text{Step II@Alg. 1}}$	44	88	116	148	168	168

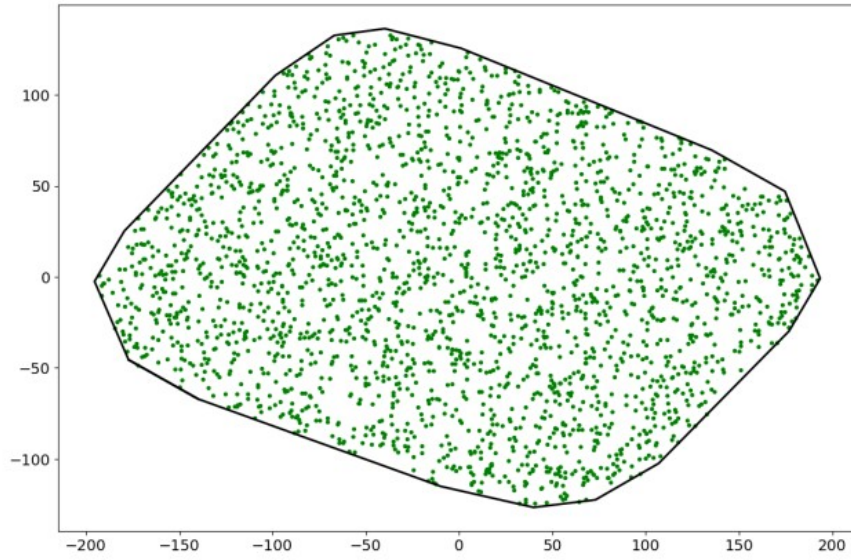
Bảng 2.1: Số cạnh trung bình của đa giác lồi xấp xỉ lồi trong $\mathcal{P}^{\text{outer}}$ trả về bởi thuật toán 1 và số lần thực hiện trung bình của bước 3 khi X gồm n điểm ngẫu nhiên trong đa giác có khung 16 cạnh (Hình 2.1).

Bảng 2.2 thể hiện một số kết quả thực nghiệm về thời gian chạy của thuật toán Outer Convex Approximation khi X gồm n điểm ngẫu nhiên được trình bày trong bảng 2.2 phải được tạo ra trong đa giác có khung 16 cạnh \mathcal{P}^\diamond hiển thị trong hình 2.1.

$\#X = n$		50	500	1000	2500	5000	7500
$\delta = 70$	$T_{\text{Alg. 1}}$	0.001531	0.001005	0.0	0.001025	0.000975	0.001025
	$T_{\text{Alg. 1}}/n$	3.06273e-05	2.0099e-06	0.0	4.102e-07	1.949e-07	1.367e-07
$\delta = 10$	$T_{\text{Alg. 3}}$	0.003005	0.003535	0.008002	0.006525	0.003	0.006528
	$T_{\text{Alg. 3}}/n$	6.01006e-05	7.0691e-06	8.0023e-06	2.61e-06	6.001e-07	8.705e-07
$\delta = 1$	$T_{\text{Alg. 3}}$	0.012033	0.016051	0.014219	0.01155	0.013153	0.013655
	$T_{\text{Alg. 3}}/n$	0.0002406549	3.21026e-05	1.42186e-05	4.62e-06	2.6307e-06	1.8206e-06
$\delta = 0$	$T_{\text{Alg. 3}}$	0.009627	0.019119	0.027318	0.036492	0.060799	0.054085
	$T_{\text{Alg. 3}}/n$	0.0001925468	3.82371e-05	2.7318e-05	1.45967e-05	1.21599e-05	7.2113e-06

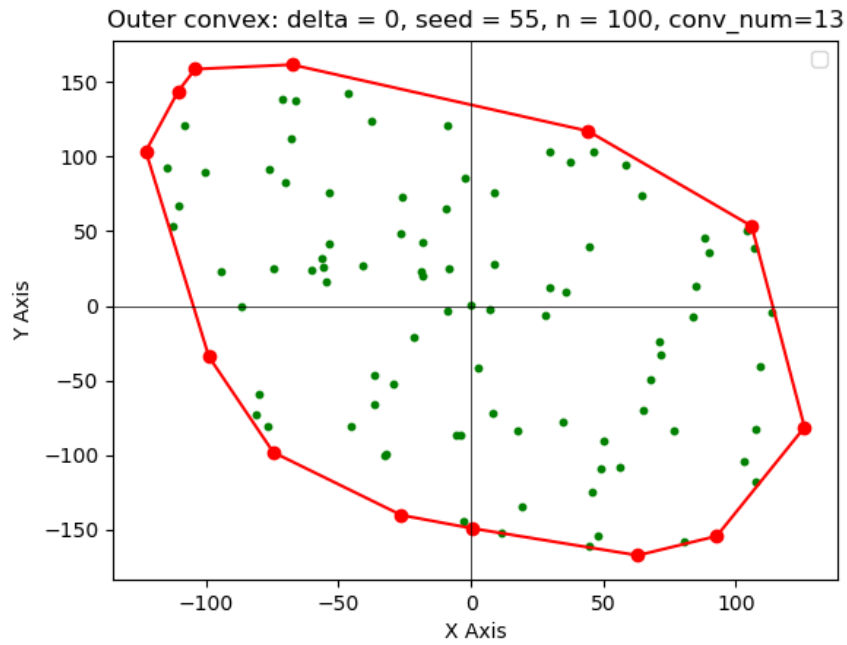
Bảng 2.2: Thời gian chạy trung bình $T_{\text{Alg. 3}}$ thuật toán 2 khi X gồm n điểm ngẫu nhiên được trình bày trong bảng 2.3 phải được tạo ra trong đa giác có khung 16 cạnh \mathcal{P}^\diamond hiển thị trong hình 2.1.

Một vài hình ảnh kết quả chạy thuật toán Outer Convex Approximation với số điểm đầu vào bằng 100 được sinh ngẫu nhiên, và được đa giác 16 cạnh cắt

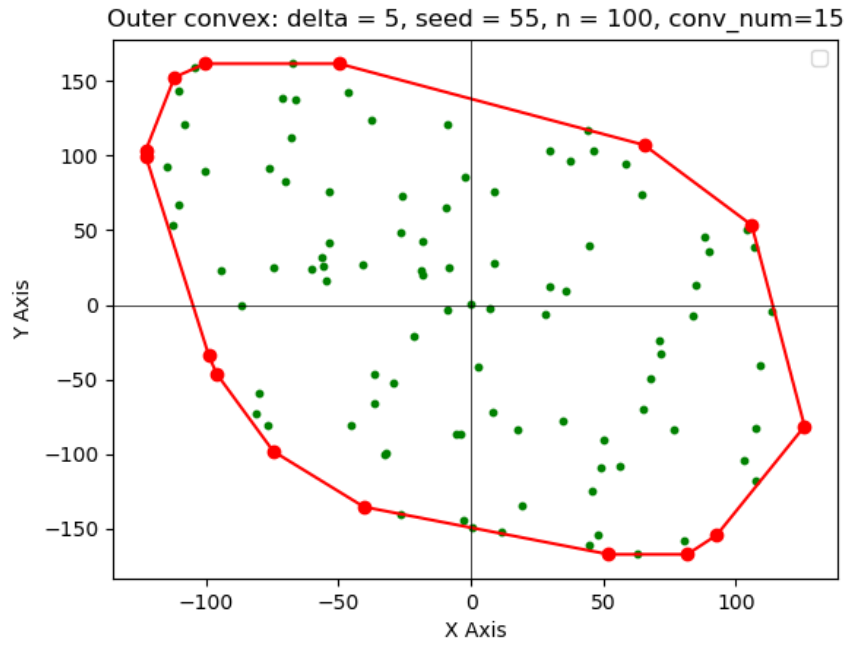


Hình 2.1: Đa giác n điểm ngẫu nhiên có khung 16 cạnh \mathcal{P}^\diamond .

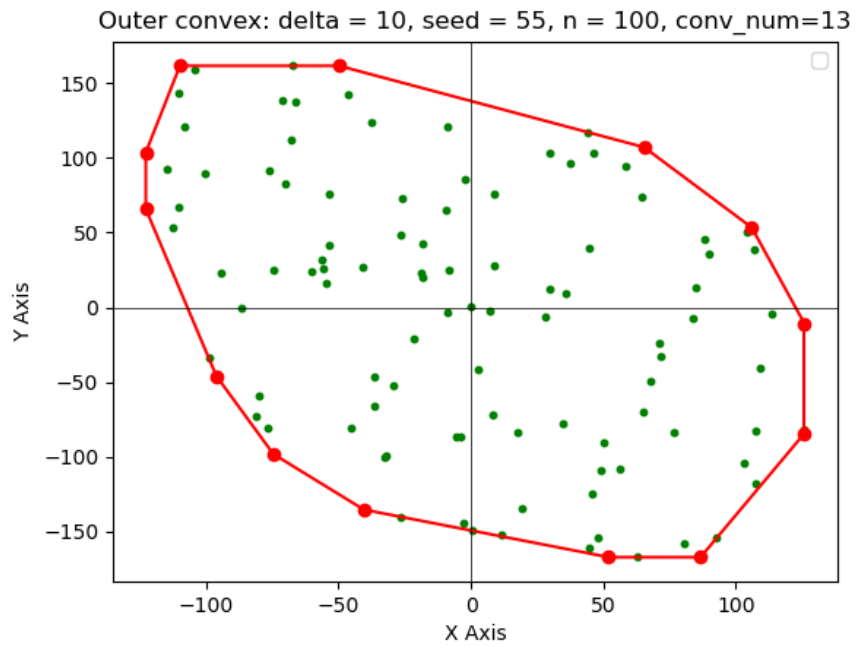
thành tập các điểm nằm trong giới hạn kích thước nhỏ bao quanh mốc 150, với δ khác nhau (Hình 2.2, 2.3, 2.4, 2.5):



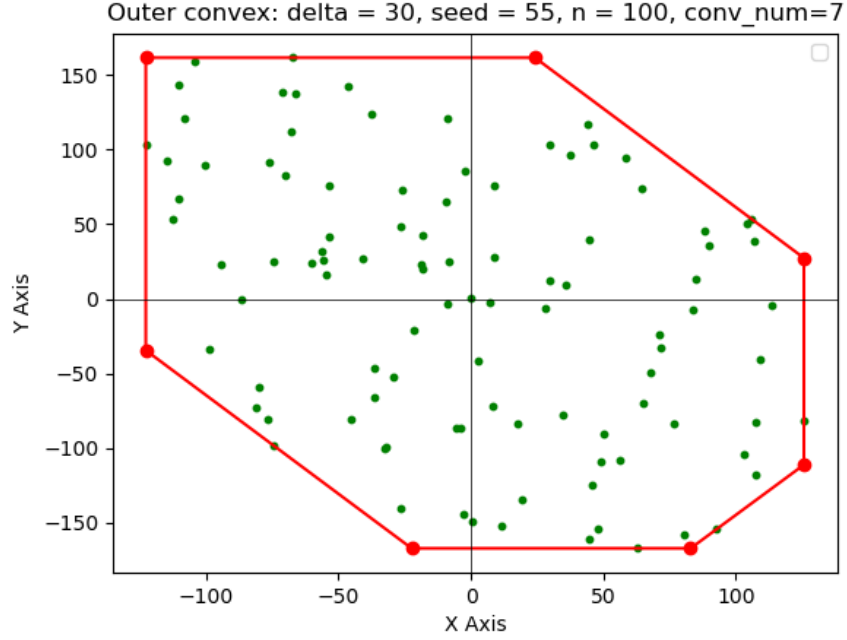
Hình 2.2: Bao lồi kết quả khi chạy với $\delta = 0$.



Hình 2.3: Bao lồi kết quả khi chạy với $\delta = 5$.



Hình 2.4: Bao lồi kết quả khi chạy với $\delta = 10$.



Hình 2.5: Bao lồi kết quả khi chạy với $\delta = 30$.

2.2 Inner convex approximation

Cho X thỏa mãn (2.1)–(2.2) và $\delta \geq 0$. Ta sẽ tìm một bao lồi xấp xỉ trong $\mathcal{P}^{\text{inner}}$ của X , tức là:

$$\mathcal{P}^{\text{inner}} := \text{conv}X', \quad \text{với } X' \subset X, \quad (2.26)$$

Sao cho:

$$\text{dist}_H(\text{conv}X, \mathcal{P}^{\text{inner}}) \leq \delta. \quad (2.27)$$

Ta mô tả $\mathcal{P}^{\text{inner}}$ bởi tập E của các cạnh có hướng $[p, p^+]$, trong đó p^+ là đỉnh kế tiếp ngược chiều kim đồng hồ của p , tức là:

$$E := \{[p, p^+] \mid p, p^+ \in X', [p, p^+] \text{ là một cạnh của } \mathcal{P}^{\text{inner}}\}. \quad (2.28)$$

Ta bắt đầu quá trình tìm bao lồi xấp xỉ trong với tứ giác $\bar{q}_1\bar{q}_2\bar{q}_3\bar{q}_4$. Xét hai công thức sau:

$$\begin{aligned} X' &:= \{\bar{q}_1, \bar{q}_2, \bar{q}_3, \bar{q}_4\}, \\ E &:= \{[\bar{q}_1, \bar{q}_2], [\bar{q}_2, \bar{q}_3], [\bar{q}_3, \bar{q}_4], [\bar{q}_4, \bar{q}_1]\}, \end{aligned} \quad (2.29)$$

Trong đó $\bar{q}_1, \bar{q}_2, \bar{q}_3$, và \bar{q}_4 là độc nhất và được xác định bởi:

$$\begin{aligned} x_{\min}^1 &:= \min\{x^1 \mid (x^1, x^2) \in X\}, \\ x_{\max}^1 &:= \max\{x^1 \mid (x^1, x^2) \in X\}, \\ x_{\min}^2 &:= \min\{x^2 \mid (x^1, x^2) \in X\}, \\ x_{\max}^2 &:= \max\{x^2 \mid (x^1, x^2) \in X\} \end{aligned} \quad (2.30)$$

$$\begin{aligned} X_{\min}^1 &:= \{(x^1, x^2) \in X \mid x^1 = x_{\min}^1\}, \\ X_{\max}^1 &:= \{(x^1, x^2) \in X \mid x^1 = x_{\max}^1\}, \\ X_{\min}^2 &:= \{(x^1, x^2) \in X \mid x^2 = x_{\min}^2\}, \\ X_{\max}^2 &:= \{(x^1, x^2) \in X \mid x^2 = x_{\max}^2\} \end{aligned} \quad (2.31)$$

Và

$$\begin{aligned} \bar{q}_1 &= (\bar{q}_1^1, \bar{q}_1^2) \in X_{\max}^1 \text{ thỏa mãn } \bar{q}_1^2 = \max\{x^2 \mid (x^1, x^2) \in X_{\max}^1\}, \\ \bar{q}_2 &= (\bar{q}_2^1, \bar{q}_2^2) \in X_{\max}^2 \text{ thỏa mãn } \bar{q}_2^1 = \min\{x^1 \mid (x^1, x^2) \in X_{\max}^2\}, \\ \bar{q}_3 &= (\bar{q}_3^1, \bar{q}_3^2) \in X_{\min}^1 \text{ thỏa mãn } \bar{q}_3^2 = \min\{x^2 \mid (x^1, x^2) \in X_{\min}^1\}, \\ \bar{q}_4 &= (\bar{q}_4^1, \bar{q}_4^2) \in X_{\min}^2 \text{ thỏa mãn } \bar{q}_4^1 = \max\{x^1 \mid (x^1, x^2) \in X_{\min}^2\}. \end{aligned} \quad (2.32)$$

Lưu ý rằng có thể hai trong số bốn điểm $\bar{q}_1, \bar{q}_2, \bar{q}_3$, và \bar{q}_4 trùng nhau, nhưng (2.2) ngụ ý rằng phải ít nhất ba trong số chúng khác nhau.

Trong các bước tiếp theo, đa giác $\mathcal{P}^{\text{inner}}$ được xây dựng và cải thiện như sau. Xét cạnh bất kỳ $[p, p^+] \in E$ ($p \neq p^+$), xác định:

$$\begin{aligned} \bar{d}_{[p, p^+]}^T &:= \|p^+ - p\|^{-1} R(p^+ - p)^T, \\ X_{[p, p^+]} &:= \{x \in X \mid \bar{d}_{[p, p^+]} x^T > \bar{d}_{[p, p^+]} p^T\}, \end{aligned} \quad (2.33)$$

Với

$$R := \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}.$$

Nếu $X_{[p, p^+]} \neq \emptyset$ thì xác định:

$$\begin{aligned} \beta_{[p, p^+]} &:= \max\{\bar{d}_{[p, p^+]} x^T \mid x \in X_{[p, p^+]}\}, \\ B_{[p, p^+]} &:= \{x \in X_{[p, p^+]} \mid \bar{d}_{[p, p^+]} x^T = \beta_{[p, p^+]}\}. \end{aligned} \quad (2.34)$$

Nếu:

$$\beta_{[p, p^+]} - \bar{d}_{[p, p^+]} p^T \leq \delta \quad (2.35)$$

Thì không cần phải mở rộng $\mathcal{P}^{\text{inner}}$ theo hướng $\bar{d}_{[p, p^+]}$ nữa.

Ngược lại, nếu:

$$\beta_{[p, p^+]} - \bar{d}_{[p, p^+]} p^T > \delta \quad (2.36)$$

thì xác định điểm:

$$\hat{p} \in B_{[p, p^+]} \text{ thỏa mãn } \|\hat{p} - p\| = \max\{\|x - p\| \mid x \in B_{[p, p^+]}\}, \quad (2.37)$$

Và cập nhật X' và B bởi

$$\begin{aligned} X' &:= X' \cup \{\hat{p}\}, \\ E &:= E \cup \{[p, \hat{p}], [\hat{p}, p^+]\}, \end{aligned} \quad (2.38)$$

Và xác định:

$$\begin{aligned} X_{[p, \hat{p}]} &:= \{x \in X_{[p, p^+]} \mid \bar{d}_{[p, \hat{p}]} x^T > \bar{d}_{[p, \hat{p}]} p^T\}, \\ X_{[\hat{p}, p^+]} &:= \{x \in X_{[p, p^+]} \mid \bar{d}_{[\hat{p}, p^+]} x^T > \bar{d}_{[\hat{p}, p^+]} \hat{p}^T\}. \end{aligned} \quad (2.39)$$

Lưu ý rằng ta chỉ xem xét $x \in X_{[p, p^+]}$ trong định nghĩa này của $X_{[p, \hat{p}]}$ và $X_{[\hat{p}, p^+]}$, nhưng tất cả $x \in X$ đều được định nghĩa (2.33) của $X_{[p, p^+]}$.

Quy trình tìm bao lồi được trình bày trong thuật toán sau, trong đó E_{doubt} biểu thị tập hợp các cạnh cần được kiểm tra.

Thuật toán 2

Input: Tập hữu hạn $X \subset \mathbb{R}^2$ và tham số xấp xỉ $\delta \geq 0$.

Output: Đa giác lồi xấp xỉ trong $\mathcal{P}^{\text{inner}}$ được mô tả bởi X' và E .

1. Tìm X' và E theo (2.29)–(2.32).
 Với mọi $[p, p^+] \in E$ xác định $d_{[p, p^+]}$ và $X_{[p, p^+]}$ theo (2.33).
 Cho $E_{\text{doubt}} := E$.
2. Chọn một cạnh $[p, p^+] \in E_{\text{doubt}}$.
 Nếu $X_{[p, p^+]} = \emptyset$, đặt $E_{\text{doubt}} := E_{\text{doubt}} \setminus \{[p, p^+]\}$ và đi tới bước 3.
 Tìm $\beta_{[p, p^+]}$ và $B_{[p, p^+]}$ theo (2.34).
 Nếu (2.35) đúng, đặt $E_{\text{doubt}} := E_{\text{doubt}} \setminus \{[p, p^+]\}$ và đi tới bước 3.
 Trường hợp còn lại, lấy \hat{p} định nghĩa theo (2.37), cập nhật X' và E theo (2.38), và xác định $X_{[p, \hat{p}]}$ và $X_{[\hat{p}, p^+]}$ theo (2.39), đặt

$$E_{\text{doubt}} := (E_{\text{doubt}} \setminus \{[p, p^+]\}) \cup \{[p, \hat{p}], [\hat{p}, p^+]\}.$$

3. Nếu $E_{\text{doubt}} \neq \emptyset$ thì quay lại bước 2.
 4. Trả về X' , E và kết thúc thuật toán.
-

Bảng 2.3 cho thấy một số kết quả thử nghiệm, trong đó:

- $\#_{\text{Edges@Alg. 2}}$ là số cạnh trung bình của đa giác xấp xỉ lồi bao ngoài $\mathcal{P}^{\text{inner}}$ được trả về bởi thuật toán 2,
- $\#_{\text{Step II@Alg. 2}}$ là số lần thực hiện trung bình của bước II của thuật toán 2.

#X = n		50	500	1000	2500	5000	7500
$\delta = 70$	#Edges@Alg. 3	6	6	6	6	6	6
	#Step II@Alg. 3	8	8	8	8	8	8
$\delta = 10$	#Edges@Alg. 3	9	10	10	11	12	11
	#Step II@Alg. 3	14	16	16	18	20	18
$\delta = 1$	#Edges@Alg. 3	12	19	22	23	26	22
	#Step II@Alg. 3	20	34	40	42	48	40
$\delta = 0$	#Edges@Alg. 3	12	23	30	38	43	43
	#Step II@Alg. 3	20	42	56	72	82	82

Bảng 2.3: Số cạnh trung bình của đa giác lồi xấp xỉ lồi trong $\mathcal{P}^{\text{inner}}$ trả về bởi thuật toán 2 và số lần thực hiện trung bình của bước 2 khi X gồm n điểm ngẫu nhiên trong đa giác có khung 16 cạnh \mathcal{P}^\diamond và được hiển thị trong hình 2.1.

LƯU Ý (Liên quan đến bảng 2.3): n điểm ngẫu nhiên được trình bày trong bảng 2.3 phải được tạo ra trong đa giác có khung 16 cạnh \mathcal{P}^\diamond và hiển thị trong hình 2.1.

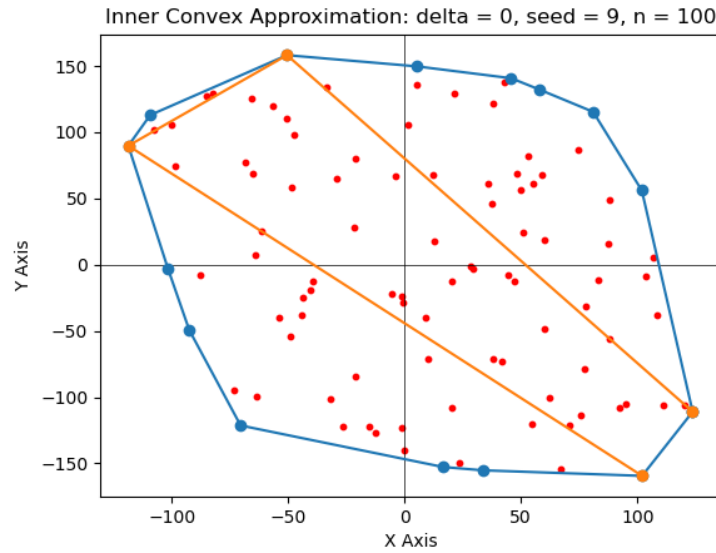
Bảng 2.4 thể hiện một số kết quả thực nghiệm về thời gian chạy của thuật toán Inner Convex Approximation khi X gồm n điểm ngẫu nhiên được trình bày trong bảng 2.3 phải được tạo ra trong đa giác có khung 16 cạnh \mathcal{P}^\diamond hiển thị trong hình 2.1, trong đó

$T_{\text{Alg. 2}}$ là thời gian chạy trung bình của thuật toán 2.

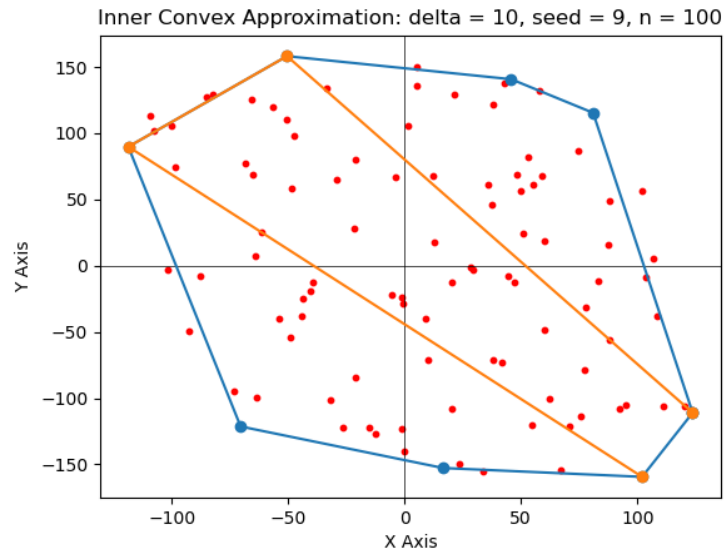
#X = n		50	500	1000	2500	5000	7500
$\delta = 70$	$T_{\text{Alg. 2}}$	0.002035	0.01511	0.03	0.065514	0.145027	0.197571
	$T_{\text{Alg. 2}}/n$	4.06981e-05	3.02205e-05	3e-05	2.62054e-05	2.90054e-05	2.63428e-05
$\delta = 10$	$T_{\text{Alg. 2}}$	0.005001	0.026086	0.047984	0.13192	0.283488	0.402505
	$T_{\text{Alg. 2}}/n$	0.0001000214	5.21712e-05	4.79844e-05	5.27678e-05	5.66977e-05	5.36673e-05
$\delta = 1$	$T_{\text{Alg. 2}}$	0.005164	0.052596	0.127814	0.291402	0.645425	0.814138
	$T_{\text{Alg. 2}}/n$	0.0001032829	0.0001051922	0.0001278136	0.0001165608	0.000129085	0.0001085517
$\delta = 0$	$T_{\text{Alg. 2}}$	0.006178	0.063702	0.158502	0.486671	1.094417	1.650608
	$T_{\text{Alg. 2}}/n$	0.0001235676	0.0001274037	0.0001585019	0.0001946683	0.0002188833	0.000220081

Bảng 2.4: Thời gian chạy trung bình $T_{\text{Alg. 2}}$ thuật toán 2 khi X gồm n điểm ngẫu nhiên được trình bày trong bảng 2.4 phải được tạo ra trong đa giác có khung 16 cạnh \mathcal{P}^\diamond hiển thị trong hình 2.1.

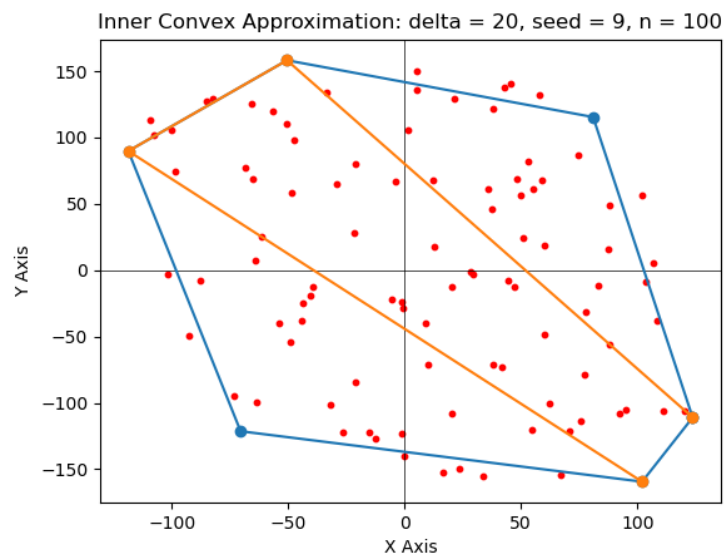
Dưới đây là một vài hình ảnh kết quả chạy của Inner Convex Approximation với nhiều δ khác nhau:



Hình 2.6: Inner Convex Approximation với $\delta = 0$.



Hình 2.7: Inner Convex Approximation với $\delta = 10$.



Hình 2.8: Inner Convex Approximation với $\delta = 20$.

Chương 3

Thực nghiệm và kết quả

Chương này trình bày cách thức triển khai của đề án: khởi tạo môi trường chạy, lấy tập dữ liệu DOTA, chuyển đổi thuật toán từ mã Python sang mã c++, thay thế thuật toán mới cho thuật toán cũ, cấu hình file config để thực hiện huấn luyện, một vài kết quả số.

3.1 Khởi tạo môi trường chạy

Môi trường được sử dụng trong đề án: Linux, bản phân phối Ubuntu 20.04.06 LTS, sử dụng 2 máy có thông số khác nhau để chạy: máy laptop GTX 1650 4GB RAM, máy pc RTX 3060 12GB RAM. CUDA Toolkit 11.6 đã cài đặt CUDNN.

Do code trong paper Beyond Bounding Box đã không còn tương thích với phiên bản Mmcv [2] hiện đại, em đã chuyển qua sử dụng thư viện Mmrotate [4], một thư viện mới được xây dựng vài tháng gần đây, có triển khai lại paper trên như là một bộ phát hiện mới, có tên là CFA.

Thực hiện tải về các thư viện Mmcv==1.7.1, Mmdetection==2.28.2 và Mmrotate==0.3.4. Riêng 2 thư viện Mmcv và Mmrotate thực hiện clone repo về và xây dựng thư viện từ nguồn (build from source) theo như tài liệu của OpenMmlab.

OpenMmlab là một dự án mã nguồn mở phục vụ cho nghiên cứu học thuật và ứng dụng công nghiệp, bao gồm nhiều chủ đề nghiên cứu trong lĩnh vực thị giác máy tính như: phân loại hình ảnh, phát hiện mục tiêu, phân đoạn mục tiêu,

tạo hình ảnh siêu phân giải, và nhiều hơn nữa.

OpenMmlab đã phát hành hơn 30 thư viện thị giác, đã triển khai hơn 300 thuật toán, và chứa hơn 2000 mô hình được huấn luyện trước.

Thư viện Mmcv đóng vai trò cung cấp các phép toán hỗ trợ ở mức thấp (sử dụng code CUDA c++ để lập trình tính toán cho card đồ hoạ). Thư viện Mmdetection [1] là một bộ thư viện chủ đạo của phòng nghiên cứu OpenMmlab, cung cấp nền tảng để xây dựng tiếp một nhánh khác đó chính là thư viện Mmrotate.

Mmdetection là một hộp công cụ phát hiện đối tượng chứa tập hợp các phương pháp phát hiện đối tượng phong phú, phân đoạn các thể hiện và phân đoạn toàn cảnh, cũng như các thành phần liên quan và các mô đun.

Mmrotate là một nhánh mới được xây dựng từ Mmdetection, là một hộp công cụ phát hiện đối tượng xoay dựa trên Pytorch. Nó cũng là một phần của các dự án OpenMMLab.



Hình 3.1: Minh hoạ kết quả hình ảnh output của thư viện Mmrotate.

3.2 Tập dữ liệu DOTA

Tập dữ liệu được sử dụng trong đề án chính là tập dữ liệu DOTA v1.0. Tập dữ liệu DOTA được thu thập từ nền tảng Google Earth, GF-2 và vệ tinh JL-1

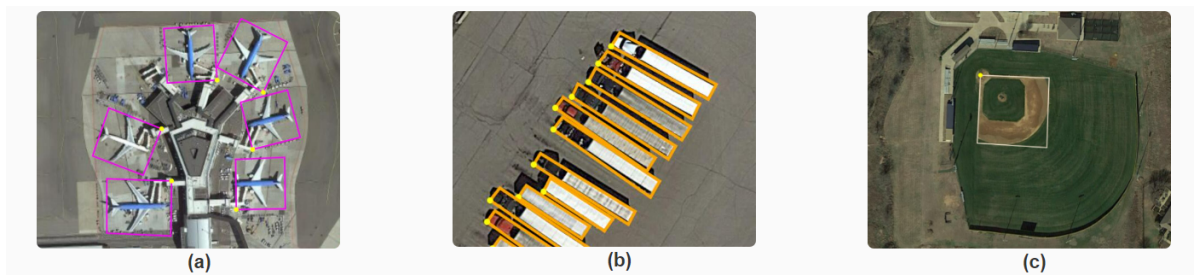
được cung cấp bởi Trung tâm Trung quốc nghiên cứu dữ liệu vệ tinh và ứng dụng, và các hình ảnh trên không được cung cấp bởi CycloMedia B.V. DOTA bao gồm các ảnh RGB và các ảnh mức xám. Tất cả các ảnh đều được lưu trữ dưới định dạng png. Tập dữ liệu DOTA có 3 phiên bản: v1.0, v1.5 và v2.0. Riêng tập dữ liệu v1.0 đã được chia sẵn các tập train, test, val.

Danh mục các class trong tập DOTA v1.0 bao gồm: plane, ship, storage tank, baseball diamond, tennis court, basketball court, ground track field, harbor, bridge, large vehicle, small vehicle, helicopter, roundabout, soccer ball field and swimming pool. (Hình 3.2)

Mỗi đối tượng được gán nhãn bởi một hộp bao có hướng (oriented bounding box - OBB), được ký hiệu là: $(x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4)$. Trong đó (x_i, y_i) biểu thị đỉnh thứ i của OBB. Các đỉnh được sắp xếp theo chiều kim đồng hồ. Ảnh 3.2 minh họa trực quan các nhãn. Điểm màu vàng đại diện cho điểm bắt đầu, có nghĩa là: a) góc trái của máy bay, b) góc trái trên cùng của một phương tiện to, c) trung tâm của một sân bóng chày.

Mỗi một thể hiện trong file nhãn có thêm một danh mục (category) và độ khó (difficult) biểu thị liệu thể hiện này có khó nhận diện hay không (1 là khó, 0 là không khó). Các nhãn của một ảnh được lưu trong một file text với cùng tên file. Mỗi dòng trong file đại diện cho một thể hiện. Dưới đây là ví dụ 1 đoạn nội dung trong 1 file annotations:

```
717.0 76.0 726.0 78.0 722.0 95.0 714.0 90.0 small-vehicle 0
737.0 82.0 744.0 84.0 739.0 101.0 731.0 98.0 small-vehicle 0
658.0 242.0 648.0 237.0 657.0 222.0 667.0 225.0 small-vehicle 1
735.0 122.0 754.0 129.0 750.0 136.0 733.0 128.0 small-vehicle 0
773.0 137.0 788.0 144.0 784.0 151.0 770.0 144.0 small-vehicle 0
809.0 153.0 827.0 161.0 823.0 168.0 806.0 160.0 small-vehicle 0
696.0 122.0 705.0 124.0 697.0 141.0 691.0 137.0 small-vehicle 0
707.0 126.0 714.0 130.0 706.0 145.0 700.0 141.0 small-vehicle 0
711.0 140.0 718.0 141.0 712.0 157.0 706.0 154.0 small-vehicle 1
924.0 113.0 926.0 106.0 945.0 115.0 942.0 122.0 small-vehicle 0
935.0 95.0 939.0 88.0 955.0 97.0 950.0 104.0 small-vehicle 1
```



Hình 3.2: Minh hoạ nhãn hộp bao có hướng của tập dữ liệu DOTA.

3.3 Cách thức thay thế thuật toán

Sau khi download được tập dữ liệu DOTA về máy, thực hiện tiền xử lý dữ liệu. Ta cần để cấu trúc dữ liệu dưới dạng cây như sau:

```

data
├── DOTA
│   ├── train
│   │   ├── images
│   │   │   ├── P001.png
│   │   │   ├── P003.png
│   │   │   └── ...
│   │   ├── labelTxt
│   │   │   ├── P001.txt
│   │   │   ├── P003.txt
│   │   │   └── ...
│   │   ├── val
│   │   │   ├── images
│   │   │   │   ├── P000.png
│   │   │   │   ├── P002.png
│   │   │   │   └── ...
│   │   │   ├── labelTxt
│   │   │   │   ├── P000.txt
│   │   │   │   ├── P002.txt
│   │   │   │   └── ...
│   │   └── test
│   │       ├── images
│   │       │   ├── P004.png
│   │       │   ├── P005.png
│   │       │   └── ...
│   │       └── test_info.json

```

Mở terminal rồi điều hướng vào thư mục gốc, thực hiện các lệnh sau:

Listing 3.1: Câu lệnh để split ảnh trong tập dữ liệu DOTA

```
$ python tools/data/dota/split/img_split.py --base-json
tools/data/dota/split/split_configs/ss_trainval.json
$ python tools/data/dota/split/img_split.py --base-json
tools/data/dota/split/split_configs/ss_test.json
```

Sử dụng lệnh để chia ảnh ra với 8 process, kích thước ảnh 1024 pixels, overlaps 200. Sau khi chia ta có cây thư mục của tập dữ liệu như sau:

```
data
├── DOTA
├── split_ss_dota
│   └── trainval
│       ├── images
│       │   ├── P0000__1024__0__0.png
│       │   ├── P0000__1024__0__824.png
│       │   ├── P0000__1024__0__1648.png
│       │   └── ...
│       └── annfiles
│           ├── P0000__1024__0__0.txt
│           ├── P0000__1024__0__824.txt
│           ├── P0000__1024__0__1648.txt
│           └── ...
```

Tiếp theo thực hiện cài đặt miniconda, một phiên bản thu gọn của Anaconda. Sau đó chạy các câu lệnh sau:

Listing 3.2: Câu lệnh để split ảnh trong tập dữ liệu DOTA

```
$ conda create -n mmrotate_v26_beta1 --no-default-packages \
python==3.9 -y
$ conda activate mmrotate_v26_beta1
$ conda config --add channels conda-forge
$ conda install pytorch==1.12.1 torchvision==0.13.1 \
torchaudio==0.12.1 \
cudatoolkit=11.6 -c pytorch -c conda-forge
$ cd mmcv
$ git checkout v1.7.1
$ pip install -r requirements/optional.txt
$ Mmcv_WITH_OPS=1 pip install -e . -v
$ pip install mmdet==2.28.2
$ cd ../mmrotate
$ git checkout v0.3.4
$ pip install -v -e .
$ conda install packaging
$ conda install scipy
$ conda install python-dateutil
```

```
$ pip install future tensorboard
```

Ta cần phải thay thế code thuật toán mới vào thuật toán Jarvis. Để thay thế được ta cần phải thay thế vào thư viện `mmcv`, để thay thế 2 hàm Jarvis và `Jarvis_and_index`. Hai hàm này có đầu vào là một mảng các Point tên là `in_poly` đại diện cho số điểm cần tìm bao lồi, `n_poly` đại diện cho số lượng phần tử trong mảng `in_poly`. Đầu tiên thay thế bằng thuật toán Outer Convex Approximation thứ nhất. Thuật toán này có code Python như sau:

Một hàm khác cần thay thế là `Jarvis_and_index()`. Hàm này tương tự hàm Jarvis, nhưng bảo toàn thứ tự của các điểm trong bao lồi khi chúng còn là một tập hợp các điểm. Ta thêm vào đoạn code này phần đầu của hàm `Jarvis_and_index()`:

Tiếp theo ta thêm đoạn code tương tự đoạn code Jarvis bên trên. Cuối cùng ta thêm đoạn code sau đây để lấy được chỉ số của các điểm nằm trên bao lồi trong tập điểm gốc:

Listing 3.3: Câu lệnh để split ảnh trong tập dữ liệu DOTA

```
from mmcv import Config
from mmrotate.datasets.builder import ROTATED_DATASETS
from mmrotate.datasets.dota import DOTADataset

@ROTATED_DATASETS.register_module()
class CustomDotaDataset(DOTADataset):
    """SAR ship dataset for detection."""
    CLASSES = ('plane', 'baseball-diamond',
               'bridge', 'ground-track-field',
               'small-vehicle', 'large-vehicle',
               'ship', 'tennis-court',
               'basketball-court', 'storage-tank',
               'soccer-ball-field',
               'roundabout', 'harbor',
               'swimming-pool', 'helicopter')
    cfg = Config.fromfile('./configs/cfa/cfa_r50_fpn_40e_dota_oc.py')
    from mmdet.apis import set_random_seed

    cfg.dataset_type = 'CustomDotaDataset'
    cfg.data_root = 'data/split_ss_dota/'

    cfg.data.test.type = 'CustomDotaDataset'
    cfg.data.test.data_root = 'data/split_ss_dota/'
    cfg.data.test.ann_file = 'test/annfiles/'
    cfg.data.test.img_prefix = 'test/images/'
```

```

cfg.data.train.type = 'CustomDotaDataset'
cfg.data.train.data_root = 'data/split_ss_dota/'
cfg.data.train.ann_file = 'trainval/annfiles/'
cfg.data.train.img_prefix = 'trainval/images/'

cfg.data.val.type = 'CustomDotaDataset'
cfg.data.val.data_root = 'data/split_ss_dota/'
cfg.data.val.ann_file = 'trainval/annfiles/'
cfg.data.val.img_prefix = 'trainval/images/'

cfg.work_dir = './results'
cfg.log_config.interval = 50

cfg.evaluation.interval = 5
cfg.checkpoint_config.interval = 1

# Set seed thus the results are more reproducible
cfg.seed = 0
set_random_seed(0, deterministic=False)
cfg.gpu_ids = range(1)
cfg.device='cuda'

import os.path as osp
import os
import mmcv
from mmdet.datasets import build_dataset
from mmdet.models import build_detector
from mmdet.apis import train_detector

import torch
torch.cuda.empty_cache()

# Build dataset
datasets = [build_dataset(cfg.data.train)]

# Build the detector
model = build_detector(
    cfg.model, train_cfg=cfg.get('train_cfg'),
    test_cfg=cfg.get('test_cfg'))
# Add an attribute for visualization convenience
model.CLASSES = datasets[0].CLASSES
mmcv.mkdir_or_exist(osp.abspath(cfg.work_dir))
train_detector(model, datasets,
    cfg, distributed=False, validate=True)

```

Ta cần phải triển khai code của thuật toán Outer Convex Approximation và Inner Convex Approximation dưới dạng mã CUDA C++. CUDA (Compute Unified Device Architecture) C++ là một phần của môi trường lập trình CUDA,

được phát triển bởi NVIDIA để hỗ trợ lập trình đa luồng trên GPU (Graphics Processing Unit). Mã CUDA C++ cung cấp một cách hiệu quả để triển khai và thực hiện các tác vụ tính toán song song trên GPU, giúp tận dụng sức mạnh tính toán lớn của các card đồ họa hiện đại.

Mã CUDA C++ đều được viết trong một chương trình C++ tiêu chuẩn, nhưng có thêm các mở rộng và đặc điểm cụ thể của CUDA để hỗ trợ lập trình song song trên GPU. Điều này giúp lập trình viên hiệu quả hóa ứng dụng của mình để sử dụng tối đa sức mạnh tính toán của GPU.

Hai thuật toán mới đưa vào phải tuân theo đầu vào và đầu ra tương tự thuật toán Jarvis. Các tham số của thuật toán Jarvis là:

`Jarvis(Point* in_poly, int n_poly)`

trong đó `in_poly` là một mảng gồm 9 hoặc 18 phần tử `Point(x, y)` đại diện cho các điểm đặc trưng. Thuật toán Jarvis sẽ thay đổi phần tử trong mảng `in_poly`, để trả về chỉ còn các phần tử `Point` của bao lồi, và thay đổi `n_poly` thành số lượng các phần tử trên bao lồi. Ngoài ra còn một hàm `Jarvis_and_index` có các tham số đầu vào như sau:

`Jarvis_and_index(Point* in_poly, int n_poly, int* point_to_convex_index).`

Hàm này tính toán tương tự như hàm `Jarvis()`, tuy nhiên trả về thêm một mảng chỉ số của các phần tử gốc trước khi mảng `in_poly` bị thay đổi. Mảng này được khởi tạo toàn bộ giá trị bằng -1. Vậy hai hàm của từng thuật toán mới sẽ được thay vào hai hàm CUDA trên và hoạt động tương tự thuật toán Jarvis.

Tóm lại trong chương 3 này trình bày cách thức để tạo môi trường huấn luyện, cách thức xử lý data trước khi đem vào huấn luyện, cuối cùng là cách thức để thay thế thuật toán Jarvis bằng thuật toán mới sao cho quá trình huấn luyện có thể được thực thi.

Chương 4

Một số kết quả tính toán

Bộ phát hiện CFA được so sánh với các bộ phát hiện tốt nhất trên tập dữ liệu DOTA với nhiệm vụ xác định hộp bao có hướng. Vì là một bộ phát hiện không dùng anchor, CFA vượt trội hơn bộ phát hiện DRN mới nhất khoảng 5.97% (76.67% vs 70.70%), một khoảng cách khá lớn. Cụ thể, trong các lớp SV, LV, RA, HA và SP, CFA vượt trội hơn 7.75% (81.23% vs. 73.48%), 10.27% (80.96% vs. 70.69%), 11.53% (69.94% vs. 58.41%), 7.90% (75.52% vs. 67.62%), and 12.16% (80.76% vs. 68.60%). Nguyên nhân ở đây có thể là do danh mục các đối tượng có hình dạng bất thường, và CFA sử dụng biểu diễn bao lồi có thể thích nghi tốt hơn với những đối tượng có bố cục và hướng bất thường. Vì là bộ phát hiện anchor-free, CFA có thể so sánh nếu không muốn nói là tốt hơn các bộ phát hiện base-detector như RoI-Transformer, SCRDet, Gliding Vertex và CSL.

Hình 4.5 cho thấy quá trình phát triển của bao lồi. Có thể thấy sau khi khởi tạo, bao lồi dần dần tiếp cận tới hộp bao thực tế (ground-truth). Phần lớn các điểm đặc trưng (feature point), đều nằm bên trong phần mở rộng của đối tượng (phần được bao bởi ground-truth).

Trong hình 4.6 so sánh bộ phát hiện khi có và không có chức năng anti-feature aliasing. Với việc thích ứng đặc trưng, biểu đồ nhiệt tại vị trí các đối tượng nằm dày đặc được chia tách rõ ràng. Điều này xác nhận sự ảnh hưởng của tính năng feature anti-aliasing của bộ phát hiện CFA.

Thực hiện một vài nghiên cứu về các siêu tham số và các module khác, CFA cũng thu được các kết quả như sau:

Method	Backbone	PL	BD	BR	GTF	SV	LV	SH	TC	BC	ST	SBF	RA	HA	SP	HC	mAP
one-stage method																	
SSD [6]	-	44.74	11.21	6.22	6.91	2.00	10.24	11.34	15.59	12.56	17.94	14.73	4.55	4.55	0.53	1.01	10.94
YOLOv2 [25]	-	76.90	33.87	22.73	34.88	38.73	32.02	52.37	61.65	48.54	33.91	29.27	36.83	36.44	38.26	11.61	39.20
FR-O [31]	ResNet101	79.09	69.12	17.17	63.49	34.20	37.16	36.20	89.19	69.60	58.96	49.40	52.52	46.69	44.80	46.30	52.93
R3Det [33]	ResNet152	89.49	81.17	50.53	66.10	70.92	78.66	78.21	90.81	85.26	84.23	61.81	63.77	68.16	69.83	67.17	73.74
two-stage method																	
R-DFPN [34]	ResNet101	80.92	65.82	33.77	58.94	55.77	50.94	54.78	90.33	66.34	68.66	48.73	51.76	55.10	51.32	35.88	57.94
R2CNN [12]	ResNet101	80.94	65.67	35.34	67.44	59.92	50.91	55.81	90.67	66.92	72.39	55.06	52.23	55.14	53.35	48.22	60.67
ICN [1]	ResNet101	81.40	74.30	47.70	70.30	64.90	67.80	70.00	90.80	79.10	78.20	53.60	62.90	67.00	64.20	50.20	68.20
RoI-Transformer [4]	ResNet101	88.64	78.52	43.44	75.92	68.81	73.68	83.59	90.74	77.27	81.46	58.39	53.54	62.83	58.93	47.67	69.56
SCRDet [37]	ResNet101	89.41	78.83	50.02	65.59	69.96	57.63	72.26	90.73	81.41	84.39	52.76	63.62	62.01	67.62	61.16	69.83
SCRDet* [37]	ResNet101	89.98	80.65	52.09	68.36	68.36	60.32	72.41	90.85	87.94	86.86	65.02	66.68	66.25	68.24	65.21	72.61
CSL† [35]	ResNet152	90.14	83.97	54.25	67.84	70.44	73.51	77.62	90.71	85.90	86.45	63.30	65.78	73.83	70.24	68.93	74.86
Gliding Vertex* [32]	ResNet101	89.64	85.00	52.26	77.34	73.01	73.14	86.82	90.74	79.02	86.81	59.55	70.91	72.94	70.86	57.32	75.02
CSL* [35]	ResNet152	90.25	85.53	54.64	75.31	70.44	73.51	77.62	90.84	86.15	86.69	69.60	68.04	73.83	71.10	68.93	76.17
anchor-free method																	
IE-Net [18]	ResNet101	80.20	64.54	39.82	32.07	49.71	65.01	52.58	81.45	44.66	78.51	46.54	56.73	64.40	64.24	36.75	57.14
CenterNet [42]	Hourglass104	89.02	69.71	37.62	63.42	65.23	63.74	77.28	90.51	79.24	77.93	44.83	54.64	55.93	61.11	45.71	65.04
DRN [24]	Hourglass104	88.91	80.22	43.52	63.35	73.48	70.69	84.94	90.14	83.85	84.11	50.12	58.41	67.62	68.60	52.50	70.70
DRN* [24]	Hourglass104	89.45	83.16	48.98	62.24	70.63	74.25	83.99	90.73	84.60	85.35	55.76	60.79	71.56	68.82	63.92	72.95
CFA(ours)	ResNet101	89.26	81.72	51.81	67.17	79.99	78.25	84.46	90.77	83.40	85.54	54.86	67.75	73.04	70.24	64.96	75.05
CFA(ours)	ResNet152	89.08	83.20	54.37	66.87	81.23	80.96	87.17	90.21	84.32	86.09	52.34	69.94	75.52	80.76	67.96	76.67

Hình 4.1: Kết quả của bộ phát hiện CFA thực hiện trên tập dữ liệu DOTA khi so sánh với các bộ phát hiện khác

Method CIoU Smoothed-L1		
mAP	66.30	65.35

Hình 4.2: So sánh hàm loss CIoU và hàm loss Smoothed-L1.

CIoU: CIoU phản ánh mức độ ảnh hưởng của mức độ chồng lấp của hộp dự đoán và hộp bao thực tế. Tối thiểu hóa hàm loss CIoU sẽ cải thiện được hộp bao dự đoán. Bảng 4.2 thực hiện so sánh hàm loss CIoU và hàm loss Smoothed-L1, CIoU đã cải thiện chỉ số mAP lên 0.95% (66.30% và 65.35%).

Convex-Hull Generation: Bằng cách mô hình hóa đối tượng thành dạng bao lồi, giúp giảm được hiện tượng feature aliasing từ cả phần nền và các đối tượng lân cận. Trong quá trình huấn luyện, bao lồi được sinh ra và thích ứng với phần mở rộng của đối tượng. Trong bảng 4.3 và hình 4.4b, bằng cách sử dụng phương pháp phân chia tập bao lồi, CFA cải thiện được hiệu suất bộ phát hiện lên 1.52% (69.70% và 68.18%), xác nhận nguyên tắc đạo hàm nhất quán với feature aliasing. Ở hình 4.4d, xác nhận rằng khởi tạo 6 bao lồi với mỗi mức feature pyramid ($I=6$) với mỗi tập hợp bao lồi sẽ đạt được hiệu suất tốt nhất.

Anti-aliasing Coefficient: Bằng cách sử dụng hệ số khử đặc trưng răng cưa (FA-A), việc thích ứng bao lồi với nhiều đối tượng được triển khai và hiệu suất được cải thiện bởi 0.43% (70.13% và 69.70%) với hệ số $\gamma=0.75$ ở hình 4.4d.

CIoU	CGen	FA-S	FA-A	mAP	Δ	$\sum \Delta$
				65.35		
✓				66.30	+0.95	0.95
✓	✓			68.18	+1.88	2.84
✓	✓	✓		69.70	+1.52	4.36
✓	✓	✓	✓	70.13	+0.43	4.79

Hình 4.3: Nghiên cứu cắt bỏ của các module trong CFA. CGen là Convex hull Generation, FA-S là feature adaptation by set splitting, FA-A là Feature Adaptation với anti-aliasing. Mạng backbone được dùng là ResNet50.

Nói chung, bộ phát hiện CFA đã cải thiện bộ phát hiện gốc lên 4.79% mAP.

Trong quá trình thay mới thuật toán Convex Approximation, ta cần quan tâm đến các yếu tố sau:

Về vị trí cần thay hàm: Có hai hàm cần thay là Jarvis() và Jarvis_and_index().

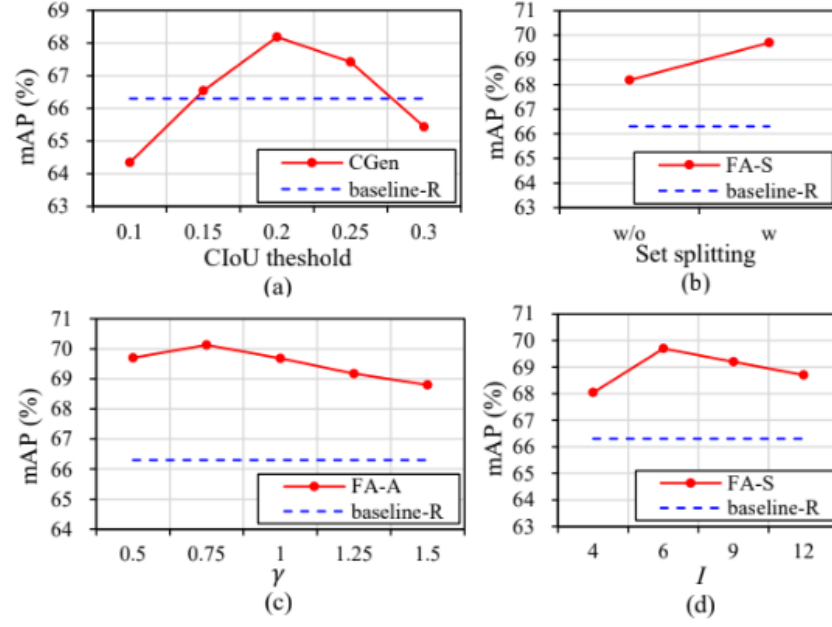
Về bộ dataset sử dụng: Có hai bộ đó là bộ hoàn chỉnh và bộ có 1000 ảnh trainval, 500 ảnh test. Yêu cầu của lần huấn luyện là phải đi qua được 40 epoch.

Với mỗi yếu tố trên ta sẽ thử với nhiều lần, trong đó có các lần thử tiêu biểu như sau:

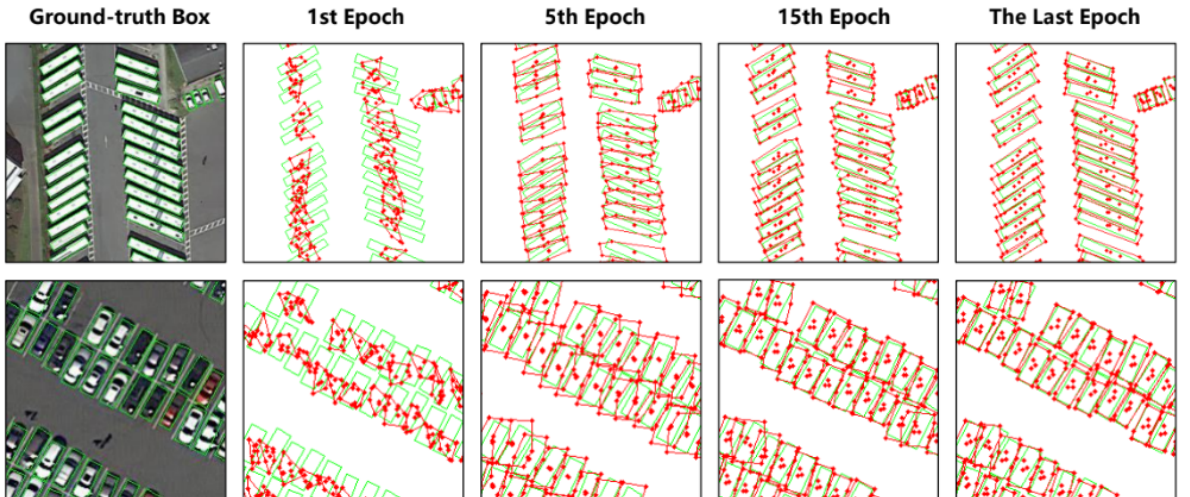
4.1 Thay thế Outer Convex Approximation và huấn luyện sử dụng bộ dữ liệu đầy đủ

Sử dụng thuật toán Outer Convex Approximation để thay thế vào hai hàm trên, sau đó cho thuật toán huấn luyện với bộ dữ liệu DOTA đã được chia cắt tỷ lệ ảnh.

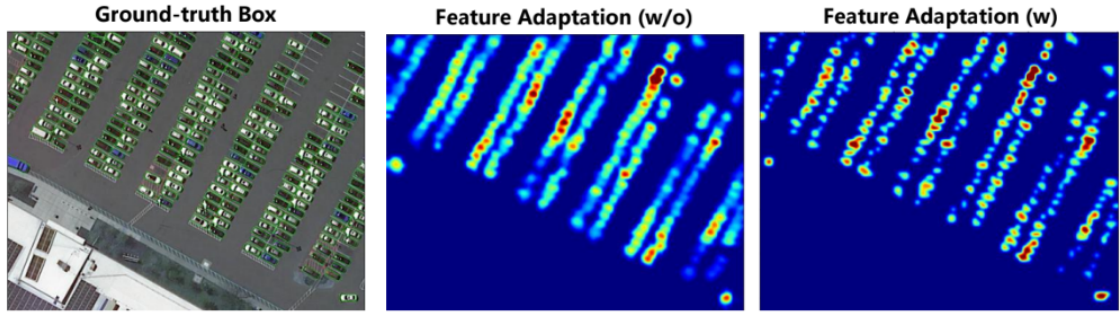
Kết quả huấn luyện: Máy chạy tốt cho đến epoch thứ 3 báo lỗi illegal memory (Hình 4.7). Dự đoán nguyên nhân do tính toán bị sai hoặc máy hết dung lượng. Thực hiện restart máy và cho chạy lại từ file checkpoint mới nhất, tuy nhiên kết quả vẫn tương tự và không chạy thêm được epoch nào.



Hình 4.4: Đánh giá tham số và các modules.



Hình 4.5: Quá trình phát triển của bao lỗi trong quá trình huấn luyện.



Hình 4.6: Biểu đồ nhiệt khi có thích nghi đặc trưng và không có thích nghi đặc trưng

```
{
  "mode": "train",
  "epoch": 3,
  "iter": 350,
  "lr": 0.008,
  "memory": 3390,
  "data_time": 0.01005,
  "loss_cls": 1.28748,
  "loss_pts_init": 0.9525,
  "loss_pts_refine": 1.0,
  "loss": 3.23998,
  "grad_norm": 2.08665,
  "time": 0.85798
}
{
  "mode": "train",
  "epoch": 3,
  "iter": 400,
  "lr": 0.008,
  "memory": 3390,
  "data_time": 0.00998,
  "loss_cls": 1.28083,
  "loss_pts_init": 0.9825,
  "loss_pts_refine": 1.0,
  "loss": 3.26333,
  "grad_norm": 2.07903,
  "time": 1.01928
}
{
  "epoch": 2,
  "iter": 13548,
  "mmcv_version": "1.7.1",
  "time": "Thu Dec 7 02:11:44 2023",
  "CLASSES": [
    "plane",
    "baseball-diamond",
    "bridge",
    "ground-track-field",
    "small-vehicle",
    "large-vehicle",
    "ship",
    "tennis-court",
    "basketball-court",
    "storage-tank",
    "soccer-ball-field",
    "roundabout",
    "harbor",
    "swimming-pool",
    "helicopter"
  ]
}
{
  "mode": "train",
  "epoch": 3,
  "iter": 50,
  "lr": 0.008,
  "memory": 3390,
  "data_time": 0.05035,
  "loss_cls": 1.28085,
  "loss_pts_init": 1.005,
  "loss_pts_refine": 1.0,
  "loss": 3.28585,
  "grad_norm": 2.15354,
  "time": 1.30152
}
{
  "mode": "train",
  "epoch": 3,
  "iter": 100,
  "lr": 0.008,
  "memory": 3390,
  "data_time": 0.00857,
  "loss_cls": 1.28274,
  "loss_pts_init": 0.9825,
  "loss_pts_refine": 1.0,
  "loss": 3.26524,
  "grad_norm": 2.34896,
  "time": 0.89406
}
{
  "mode": "train",
  "epoch": 3,
  "iter": 150,
  "lr": 0.008,
  "memory": 3390,
  "data_time": 0.01009,
  "loss_cls": 1.27691,
  "loss_pts_init": 0.99,
  "loss_pts_refine": 1.0,
  "loss": 3.26691,
  "grad_norm": 2.38729,
  "time": 1.05814
}
```

Hình 4.7: Outer Convex Approximation sử dụng toàn bộ tập dữ liệu, sập tại epoch số 3.


4.2 Thay thế Outer Convex Approximation và huấn luyện sử dụng dataset nhỏ

Do cần có kết quả sớm để xem thuật toán có thực hiện được hay không, ta sẽ sử dụng với bộ dataset nhỏ hơn. Bộ dataset này được lấy từ bộ dataset đầy đủ với 1000 ảnh và nhãn dành cho tập trainval, 500 ảnh dành cho tập test. Kết quả là thuật toán chạy được đến epoch số 20 là bị sập, báo lỗi illegal memory (Hình 4.8). Ngoài ra quá trình huấn luyện thu được file checkpoint và file log (Hình 4.9).

```

4:24:03, time: 0.961, data_time: 0.044, memory: 1904, loss_cls: 0.6836, loss_pts_init:
0.6811, loss_pts_refine: 1.0000, loss: 2.3647, grad_norm: 13.4189
2023-12-03 15:05:48,082 - mmdet - INFO - Epoch [20][100/878]   lr: 1.000e-03, eta:
4:23:23, time: 0.908, data_time: 0.003, memory: 1904, loss_cls: 0.7780, loss_pts_init:
0.6967, loss_pts_refine: 1.0000, loss: 2.4746, grad_norm: 14.6331
2023-12-03 15:06:33,002 - mmdet - INFO - Epoch [20][150/878]   lr: 1.000e-03, eta:
4:22:41, time: 0.898, data_time: 0.003, memory: 1904, loss_cls: 0.7606, loss_pts_init:
0.6726, loss_pts_refine: 1.0000, loss: 2.4332, grad_norm: 14.9846
2023-12-03 15:07:14,181 - mmdet - INFO - Epoch [20][200/878]   lr: 1.000e-03, eta:
4:21:56, time: 0.824, data_time: 0.003, memory: 1904, loss_cls: 0.7875, loss_pts_init:
0.7322, loss_pts_refine: 1.0000, loss: 2.5197, grad_norm: 17.5209
2023-12-03 15:08:01,415 - mmdet - INFO - Epoch [20][250/878]   lr: 1.000e-03, eta:
4:21:18, time: 0.945, data_time: 0.003, memory: 1904, loss_cls: 0.7396, loss_pts_init:
0.8009, loss_pts_refine: 1.0000, loss: 2.5404, grad_norm: 17.0405

```

 **Traceback...**

```

RuntimeError: CUDA error: an illegal memory access was encountered
CUDA kernel errors might be asynchronously reported at some other API call, so the
stacktrace below might be incorrect.
For debugging consider passing CUDA_LAUNCH_BLOCKING=1.

```

Hình 4.8: Outer Convex Approximation chạy với dataset nhỏ, sập tại epoch 20.

```

{"mode": "val", "epoch": 19, "iter": 880, "lr": 0.001, "mAP": 0.0}
{"mode": "train", "epoch": 20, "iter": 50, "lr": 0.001, "memory": 1904, "data_time": 0.04438,
"loss_cls": 0.68364, "loss_pts_init": 0.6811, "loss_pts_refine": 1.0, "loss": 2.36474,
"grad_norm": 13.41888, "time": 0.96126}
{"mode": "train", "epoch": 20, "iter": 100, "lr": 0.001, "memory": 1904, "data_time":
0.00296, "loss_cls": 0.77797, "loss_pts_init": 0.69665, "loss_pts_refine": 1.0, "loss":
2.47462, "grad_norm": 14.63312, "time": 0.90799}
{"mode": "train", "epoch": 20, "iter": 150, "lr": 0.001, "memory": 1904, "data_time":
0.00299, "loss_cls": 0.76063, "loss_pts_init": 0.6726, "loss_pts_refine": 1.0, "loss":
2.43324, "grad_norm": 14.9846, "time": 0.8984}
{"mode": "train", "epoch": 20, "iter": 200, "lr": 0.001, "memory": 1904, "data_time":
0.00297, "loss_cls": 0.78748, "loss_pts_init": 0.73223, "loss_pts_refine": 1.0, "loss":
2.5197, "grad_norm": 17.52086, "time": 0.82358}
{"mode": "train", "epoch": 20, "iter": 250, "lr": 0.001, "memory": 1904, "data_time":
0.00294, "loss_cls": 0.73958, "loss_pts_init": 0.80086, "loss_pts_refine": 1.0, "loss":
2.54044, "grad_norm": 17.04048, "time": 0.94466}

```

Hình 4.9: Outer Convex Approximation train với dataset nhỏ, sập tại epoch 20.

4.3 Thay thế Outer Convex Approximation vào hàm Jarvis() và huấn luyện sử dụng dataset nhỏ

Do nghi ngờ kết quả lỗi xuất hiện ở hàm Jarvis_and_index(), em chỉ thay thuật toán vào hàm Jarvis, tức là trong quá trình huấn luyện sẽ có sử dụng cả

thuật toán Jarvis March và thuật toán Outer Convex Approximation. Kết quả quá trình huấn luyện diễn ra được thuận lợi, thuật toán chạy hết được 40 epochs mà không gặp lỗi gì. Tuy nhiên do bộ dataset quá nhỏ, cùng với giới hạn các class trong dataset bị mất cân bằng nên không sử dụng được kết quả này, chỉ có thể chứng minh được là thuật toán đã hoạt động được và cho kết quả chạy chính xác. Tuy nhiên kết luận này vẫn cần phải kiểm tra và xác minh lại tính chính xác khách quan hơn.

```
{
  "mode": "train", "epoch": 40, "iter": 650, "lr": 0.0, "memory": 1904, "data_time": 0.00303, "loss_cls": 0.74902, "loss_pts_init": 0.66048, "loss_pts_refine": 1.0, "loss": 2.4095, "grad_norm": 41.90588, "time": 0.80684}
{
  "mode": "train", "epoch": 40, "iter": 700, "lr": 0.0, "memory": 1904, "data_time": 0.00298, "loss_cls": 0.70652, "loss_pts_init": 0.78217, "loss_pts_refine": 1.0, "loss": 2.48869, "grad_norm": 145.42664, "time": 0.78938}
{
  "mode": "train", "epoch": 40, "iter": 750, "lr": 0.0, "memory": 1904, "data_time": 0.00297, "loss_cls": 0.67511, "loss_pts_init": 0.73762, "loss_pts_refine": 1.0, "loss": 2.41273, "grad_norm": 50.39252, "time": 0.77488}
{
  "mode": "train", "epoch": 40, "iter": 800, "lr": 0.0, "memory": 1904, "data_time": 0.00302, "loss_cls": 0.76324, "loss_pts_init": 0.72348, "loss_pts_refine": 1.0, "loss": 2.48672, "grad_norm": 34.10888, "time": 0.78428}
{
  "mode": "train", "epoch": 40, "iter": 850, "lr": 0.0, "memory": 1904, "data_time": 0.00301, "loss_cls": 0.70641, "loss_pts_init": 0.77421, "loss_pts_refine": 1.0, "loss": 2.48062, "grad_norm": 60.11066, "time": 0.80681}
{
  "mode": "val", "epoch": 40, "iter": 880, "lr": 0.0, "mAP": 0.0}
}
```

Hình 4.10: Outer Convex Approximation chỉ thay code ở hàm Jarvis, đạt được 40 epoch.

4.4 Thay thế Inner Convex Approximation và huấn luyện sử dụng toàn bộ dữ liệu đầy đủ

Kết quả: Quá trình huấn luyện chạy thuận lợi cho đến epoch số 15, máy báo lỗi Illegal Memory, tức là lỗi truy cập vùng nhớ không hợp lệ.

4.5 Thay thế Inner Convex Approximation và huấn luyện sử dụng bộ dữ liệu nhỏ

Kết quả: Máy chạy đến epoch số 20 báo lỗi Illegal Memory.


```
{
  "mode": "train",
  "epoch": 15,
  "iter": 6150,
  "lr": 0.008,
  "memory": 3391,
  "data_time": 0.00797,
  "loss_cls": 0.23237,
  "loss_pts_init": 0.24966,
  "loss_pts_refine": 0.31388,
  "loss": 0.79591,
  "grad_norm": 1.88259,
  "time": 0.87933
}
{"mode": "train", "epoch": 15, "iter": 6200, "lr": 0.008, "memory": 3391, "data_time": 0.00803, "loss_cls": 0.2216, "loss_pts_init": 0.24466, "loss_pts_refine": 0.30215, "loss": 0.7684, "grad_norm": 1.93116, "time": 1.26589}
{"mode": "train", "epoch": 15, "iter": 6250, "lr": 0.008, "memory": 3391, "data_time": 0.00795, "loss_cls": 0.25732, "loss_pts_init": 0.2629, "loss_pts_refine": 0.32325, "loss": 0.84348, "grad_norm": 2.07284, "time": 0.92475}
{"mode": "train", "epoch": 15, "iter": 6300, "lr": 0.008, "memory": 3391, "data_time": 0.00856, "loss_cls": 0.25533, "loss_pts_init": 0.28362, "loss_pts_refine": 0.32792, "loss": 0.86687, "grad_norm": 2.05459, "time": 1.24971}
{"mode": "train", "epoch": 15, "iter": 6350, "lr": 0.008, "memory": 3391, "data_time": 0.01066, "loss_cls": 0.27237, "loss_pts_init": 0.27552, "loss_pts_refine": 0.32163, "loss": 0.86951, "grad_norm": 2.41547, "time": 1.14398}
{"mode": "train", "epoch": 15, "iter": 6400, "lr": 0.008, "memory": 3391, "data_time": 0.01058, "loss_cls": 0.28917, "loss_pts_init": 0.28606, "loss_pts_refine": 0.33767, "loss": 0.9129, "grad_norm": 2.20943, "time": 1.03897}
```

Hình 4.11: Inner Convex Approximation train với toàn bộ dữ liệu, sập tại epoch số 15.

```
{
  "mode": "train",
  "epoch": 20,
  "iter": 150,
  "lr": 0.008,
  "memory": 3391,
  "data_time": 0.01056,
  "loss_cls": 0.2656,
  "loss_pts_init": 0.3696,
  "loss_pts_refine": 0.41727,
  "loss": 1.05247,
  "grad_norm": 2.25209,
  "time": 0.99265
}
{"mode": "train", "epoch": 20, "iter": 200, "lr": 0.008, "memory": 3391, "data_time": 0.01056, "loss_cls": 0.29144, "loss_pts_init": 0.40818, "loss_pts_refine": 0.43142, "loss": 1.13104, "grad_norm": 2.59973, "time": 0.96622}
{"mode": "train", "epoch": 20, "iter": 250, "lr": 0.008, "memory": 3391, "data_time": 0.01049, "loss_cls": 0.29265, "loss_pts_init": 0.38577, "loss_pts_refine": 0.41371, "loss": 1.09213, "grad_norm": 2.65041, "time": 0.84849}
{"mode": "train", "epoch": 20, "iter": 300, "lr": 0.008, "memory": 3391, "data_time": 0.01046, "loss_cls": 0.27335, "loss_pts_init": 0.35161, "loss_pts_refine": 0.38105, "loss": 1.00601, "grad_norm": 2.54941, "time": 0.8957}
{"mode": "train", "epoch": 20, "iter": 350, "lr": 0.008, "memory": 3391, "data_time": 0.00718, "loss_cls": 0.31545, "loss_pts_init": 0.36797, "loss_pts_refine": 0.42471, "loss": 1.10812, "grad_norm": 2.41123, "time": 0.8399}
{"mode": "train", "epoch": 20, "iter": 400, "lr": 0.008, "memory": 3391, "data_time": 0.00969, "loss_cls": 0.27778, "loss_pts_init": 0.33397, "loss_pts_refine": 0.38832, "loss": 1.00008, "grad_norm": 2.57561, "time": 0.86401}
```

Hình 4.12: Inner Convex Approximation train với bộ dữ liệu nhỏ hơn, sập tại epoch số 20.

Kết luận

Trong tất cả các lần huấn luyện trên, chỉ có lần thử với bộ dataset nhỏ cùng với việc thay thế 1 nửa thuật toán là đáp ứng được quá trình training 40 epoch. Vì số lượng bộ dữ liệu không đảm bảo cho kết quả huấn luyện đúng, nên chỉ có thể kết luận rằng thuật toán mới có thể thay thế được vào bộ phát hiện, nhưng chưa thể sử dụng được kết quả. Có nhiều nguyên nhân khác nhau dẫn đến lỗi, nhưng nguyên nhân chính vẫn là chưa tìm hiểu đủ kiến thức cũng như kinh nghiệm trong công việc này.

Qua các lần training không thành công, em xin đề xuất hướng xử lý tiếp theo của đề án:

- Cần kiểm tra lại file config cấu hình training. Nếu file này cấu hình chưa đúng dẫn đến hiện tượng training giữa chừng thì bị lỗi.
- Xem xét lỗi Illegal Memory. Lỗi này là lỗi chính bắt gặp khi thực hiện train. Cần phải tìm hiểu kỹ hơn nữa về nguyên nhân của lỗi là do phần cứng hay phần mềm sinh ra.
- Cần tìm hiểu thêm về cách thư viện Mmrotate hoạt động. Đây là một thư viện lớn, có nhiều thuật ngữ và thuật toán khó hiểu.
- Tìm hiểu thêm nhiều tài liệu về bài toán phát hiện đối tượng nói riêng, cũng như trong lĩnh vực trí tuệ nhân tạo nói chung.

Trong quá trình làm đề án tốt nghiệp, em đã có cơ hội được trải nghiệm chuyên ngành AI cũng như được thử sức qua các công nghệ nhận diện ảnh mới nhất. Em đã tích lũy được những kinh nghiệm về kiến thức trong công việc cũng như các kỹ năng mềm trong xử lý các công việc liên quan đến hệ điều

hành linux, cách triển khai một dự án trí tuệ nhân tạo, cách tìm hiểu thông tin từ các tài liệu chính thống.

Em được rèn luyện kỹ năng giải quyết các công việc con, cách tương tác với người khác để trao đổi kiến thức, từ đó rút ngắn công sức và thời gian. Đồng thời em cũng được áp dụng lại các kiến thức đã được học từ các môn trên trường vào đề án thực tế này.

Tài liệu tham khảo

- [1] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.
- [2] MMCV Contributors. MMCV: OpenMMLab computer vision foundation. <https://github.com/open-mmlab/mmcv>, 2018.
- [3] Yuwen Xiong Yi Li Guodong Zhang Han Hu Yichen Wei Jifeng Dai, Haozhi Qi. Deformable convolutional networks. *arXiv preprint arXiv:1703.06211*, 2017.
- [4] Yue Zhou, Xue Yang, Gefan Zhang, Jiabao Wang, Yanyi Liu, Liping Hou, Xue Jiang, Xingzhao Liu, Junchi Yan, Chengqi Lyu, Wenwei Zhang, and Kai Chen. Mmrotate: A rotated object detection benchmark using pytorch. In *Proceedings of the 30th ACM International Conference on Multimedia*, 2022.
- [5] Xiaosong Zhang Jianbin Jiao Xiangyang Ji Zonghao Guo, Chang Liu and Qixiang Ye. Beyond bounding-box: Convex-hull feature adaptation for oriented and densely packed object detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021.