

**ĐẠI HỌC QUỐC GIA TP. HCM
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KHOA HỌC MÁY TÍNH**



**KHÓA LUẬN TỐT NGHIỆP
TẨN CÔNG HỘP ĐEN THUẨA CÁC MÔ HÌNH THỊ
GIÁC MÁY TÍNH BẰNG TÍNH TOÁN TIẾN HÓA**

CỬ NHÂN NGÀNH KHOA HỌC MÁY TÍNH

**LÊ CHÍ CƯỜNG
PHAN TRƯỜNG TRÍ**

TP. HỒ CHÍ MINH, 2025

ĐẠI HỌC QUỐC GIA TP. HCM
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KHOA HỌC MÁY TÍNH

LÊ CHÍ CƯỜNG - 21520012
PHAN TRƯỜNG TRÍ - 21520117

KHÓA LUẬN TỐT NGHIỆP
TẤN CÔNG HỘP ĐEN THUА CÁC MÔ HÌNH THỊ
GIÁC MÁY TÍNH BẰNG TÍNH TOÁN TIẾN HÓA

CỬ NHÂN NGÀNH KHOA HỌC MÁY TÍNH

GIẢNG VIÊN HƯỚNG DẪN:
TS. LƯƠNG NGỌC HOÀNG

TP. HỒ CHÍ MINH, 2025

DANH SÁCH HỘI ĐỒNG BẢO VỆ KHÓA LUẬN

Hội đồng chấm khóa luận tốt nghiệp, thành lập theo Quyết định Số 24/QĐ-ĐHCNTT, ngày 09 tháng 01 năm 2025 của Hiệu trưởng Trường Đại học Công nghệ Thông tin.

1. TS. Mai Tiến Dũng – Chủ tịch.
2. ThS. Phan Minh Quân – Thư ký.
3. TS. Võ Nguyễn Lê Duy – Ủy viên.

TP. HCM, ngày.....tháng.....năm.....

**NHẬN XÉT KHÓA LUẬN TỐT NGHIỆP
CÁN BỘ HƯỚNG DẪN**

Tên khóa luận:

**TẨN CÔNG HỘP ĐEN THUА CÁC MÔ HÌNH THỊ GIÁC MÁY TÍNH BẰNG
TÍNH TOÁN TIẾN HÓA**

Nhóm SV thực hiện:

Lê Chí Cường - 21520012

Phan Trường Trí - 21520117

Cán bộ hướng dẫn:

TS. Lương Ngọc Hoàng

Đánh giá Khóa luận

1. Về cuốn báo cáo:

Số trang _____
Số bảng số liệu _____
Số tài liệu tham khảo _____

Số chương _____
Số hình vẽ _____
Sản phẩm _____

Nhận xét về hình thức cuốn báo cáo:

.....
.....
.....
.....

2. Về nội dung nghiên cứu:

.....
.....
.....

3. Về chương trình ứng dụng:

.....
.....
.....

4. Về thái độ làm việc của sinh viên:

.....
.....
.....

Đánh giá chung:

.....
.....

Điểm từng sinh viên:

Lê Chí Cường **/10**

Phan Trường Trí **/10**

Người nhận xét
(Ký tên và ghi rõ họ tên)

TP. HCM, ngày.....tháng.....năm.....

**NHẬN XÉT KHÓA LUẬN TỐT NGHIỆP
CÁN BỘ PHẢN BIỆN**

Tên khóa luận:

**TẨN CÔNG HỘP ĐEN THUА CÁC MÔ HÌNH THỊ GIÁC MÁY TÍNH BẰNG
TÍNH TOÁN TIẾN HÓA**

Nhóm SV thực hiện:

Lê Chí Cường - 21520012

Phan Trường Trí - 21520117

Cán bộ phản biện:

TS. Võ Nguyễn Lê Duy

Đánh giá Khóa luận

1. Về cuốn báo cáo:

Số trang _____
Số bảng số liệu _____
Số tài liệu tham khảo _____

Số chương _____
Số hình vẽ _____
Sản phẩm _____

Nhận xét về hình thức cuốn báo cáo:

.....
.....
.....
.....

2. Về nội dung nghiên cứu:

.....
.....
.....

3. Về chương trình ứng dụng:

.....
.....
.....

4. Về thái độ làm việc của sinh viên:

.....
.....
.....

Đánh giá chung:

.....
.....

Điểm từng sinh viên:

Lê Chí Cường **/10**

Phan Trường Trí **/10**

Người nhận xét
(Ký tên và ghi rõ họ tên)

ĐỀ CƯƠNG CHI TIẾT

TÊN ĐỀ TÀI TIẾNG VIỆT: TÂN CÔNG HỘP ĐEN THUА CÁC MÔ HÌNH THỊ GIÁC MÁY TÍNH BẰNG TÍNH TOÁN TIẾN HÓA.

TÊN ĐỀ TÀI TIẾNG ANH: BLACK-BOX SPARSE ADVERSARIAL ATTACK ON COMPUTER VISION MODELS USING EVOLUTIONARY COMPUTATION.

Cán bộ hướng dẫn: TS. Lương Ngọc Hoàng, Khoa Khoa học Máy tính

Thời gian thực hiện: Từ ngày 02/09/2024.....đến ngày 28/12/2024.....

Sinh viên thực hiện:

Lê Chí Cường – 21520012

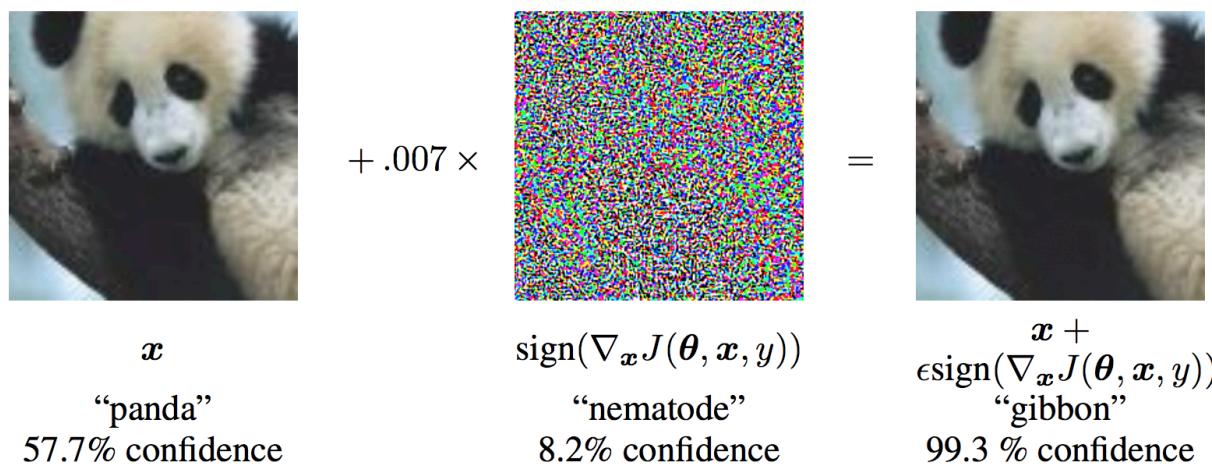
Phan Trường Trí – 21520117

Hệ đào tạo: Tài năng

Nội dung đề tài

Tổng quan đề tài :

Các mạng học sâu (Deep Neural Networks - DNNs) đã đạt được những thành tựu vượt bậc trong những năm qua và được ứng dụng ngày càng nhiều vào trong cuộc sống. Tuy nhiên, kết quả đầu ra (output) của DNNs có thể bị làm cho sai lệch với kết quả mong muốn chỉ bằng cách thêm một lượng nhiễu (**Adversarial Perturbation**) nhỏ (không khiến cho ngữ cảnh của dữ liệu đầu vào bị thay đổi). Việc thêm nhiễu vào input khiến cho mô hình DNNs cho ra kết quả sai lệch được gọi là **Adversarial Attack**. Mục tiêu của bài toán là cực tiểu hóa lượng nhiễu (theo Norm L_0, L_1, L_2 L_0, L_1, L_2 hoặc L_∞ L_∞) cần thêm vào dữ liệu gốc để mô hình dự đoán sai.



Hình 1: Ví dụ về adversarial attack. Bằng mắt thường ta vẫn nhận ra hình ảnh là "panda" nhưng mô hình lại dự đoán sai thành "gibbon".

Adversarial Attack lên các mô hình DNNs có thể được chia làm 2 loại chính:

- Tấn công hộp trắng (**white box attack**): người tấn công có quyền truy cập vào toàn bộ thông tin của mô hình (kiến trúc, đầu ra, gradient, trọng số, ...) và sử dụng các thông tin đó.
- Tấn công hộp đen (**black box attack**): người tấn công không được phép dùng thông tin của mô hình trong quá trình tấn công. Chỉ có thể nhận đầu ra tương ứng với từng dữ liệu đầu vào được truyền đến mô hình.

Tấn công hộp trắng thường cho ra hiệu suất cao hơn tấn công hộp đen, nhưng trong thực tế ngữ cảnh thường gặp hơn là trường hợp của tấn công hộp đen. Và trong ngữ cảnh này thì người ta thường sử dụng các giải thuật thuộc lớp tính toán tiến hóa.

Đối với các mô hình thị giác máy tính nói riêng, loại mô hình thường được tấn công trong các nghiên cứu gần đây là các mô hình thuộc bài toán phân lớp, còn các loại mô hình thuộc những bài toán khác thì chưa nhận được nhiều sự quan tâm.

Trong đề tài này chúng em tập trung vào bài toán tấn công hộp đen trên các mô hình thị giác máy tính thuộc nhiều loại bài toán khác nhau (phân lớp, phát hiện đối tượng, ...) với hình thức tấn công là tấn công thưa - thay đổi càng pixel càng tốt hay cực tiểu hóa L_0 L_0 .

Tài liệu tham khảo:

- [1] J. Peck, B. Goossens, and Y. Saeys, “An introduction to adversarially robust deep learning,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 4, pp. 2071–2090, 2024.
- [2] X. Dong, D. Chen, J. Bao, *et al.*, “Greedyfool: Distortion-aware sparse adversarial attack,” in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., 2020, pp. 11 226–11 236.
- [3] Y. Fan, B. Wu, T. Li, *et al.*, “Sparse adversarial attack via perturbation factorization,” in *European conference on computer vision*, 2020.
- [4] M. Zhu, T. Chen, and Z. Wang, *Sparse and imperceptible adversarial attack via a homotopy algorithm*, 2021.
- [5] C. Soussen, J. Idier, J. Duan, and D. Brie, “Homotopy based algorithms for ℓ_0 -regularized least-squares,” *IEEE Transactions on Signal Processing*, vol. 63, no. 13, pp. 3301–3316, 2015.
- [6] J. Su, D. V. Vargas, and K. Sakurai, “One pixel attack for fooling deep neural networks,” *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 828–841, 2019.
- [7] R. Storn and K. V. Price, “Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces,” *Journal of Global Optimization*, vol. 11, pp. 341–359, 1997.
- [8] F. Croce, M. Andriushchenko, N. D. Singh, N. Flammarion, and M. Hein, Sparse-rs: A versatile framework for query-efficient sparse black-box adversarial attacks, 2022.
- [9] P. N. Williams and K. Li, “Black-box sparse adversarial attack via multi-objective optimisation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 12 291–12 301.

[10] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: Nsga-ii,” IEEE Transactions on Evolutionary Computation, vol. 6, no. 2, pp. 182–197, 2002.

[11] A. Krizhevsky, “Learning multiple layers of features from tiny images,” 2009.

[12] F. Croce, M. Andriushchenko, V. Sehwag, et al., “Robustbench: A standardized adversarial robustness benchmark,” in Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track, 2021.

Mục tiêu của đề tài:

- Đề tài này hướng tới việc tìm ra phương pháp dựa trên thuật toán tiến hóa để thực hiện việc tấn công vào các mô hình thị giác máy tính.
- Một quyền báo cáo về đề tài và những kết quả của đạt được.

Phương pháp thực hiện:

- Đưa bài toán trở thành bài toán tìm kiếm.
- Sử dụng các chiến lược tính toán tiến hóa để tìm kiếm lời giải.
- So sánh phương pháp tìm được với các phương pháp hiện có.

Các nội dung chính và giới hạn của đề tài:

- Tìm hiểu các nghiên cứu có liên quan đến đề tài.
- Xây dựng các giải pháp và thực nghiệm các giải pháp trên các mô hình như mô hình phân lớp và mô hình phát hiện vật thể.
- Chạy thực nghiệm với các giải pháp hiện có để so sánh hiệu suất.
- Môi trường thực nghiệm (dự kiến): kaggle và colab

Kế hoạch thực hiện: (Mô tả kế hoạch làm việc và phân công công việc cho từng sinh viên tham gia)

Thời gian	Mô tả	Người thực hiện
Tháng 9	Tìm hiểu về các nghiên cứu có liên quan đến bài toán đang thực hiện	Trí, Cường
Tháng 10, 11	Xây dựng thuật toán và tiến hành chạy thực nghiệm	Trí, Cường
Tháng 12	Viết báo cáo.	Trí, Cường

Xác nhận của CBHD

(Ký tên và ghi rõ họ tên)

TP. HCM, ngày 6.tháng 9.năm 2024

Sinh viên

(Ký tên và ghi rõ họ tên)

LỜI CẢM ƠN

Đầu tiên, chúng tôi xin được gửi lời cảm ơn chân thành đến Thầy TS. Lương Ngọc Hoàng đã tận tình động viên, giúp đỡ và định hướng trong suốt thời gian nghiên cứu và thực hiện khóa luận này.

Tiếp theo, chúng tôi xin chân thành cảm ơn quý thầy cô trong khoa Khoa Học Máy Tính nói riêng và toàn thể thầy cô trong trường Đại Học Công Nghệ Thông Tin nói chung đã tận tình giảng dạy, trang bị cho tôi những kiến thức quý báu trong những năm ngồi trên ghế nhà trường.

Cuối cùng, chúng tôi bày tỏ lòng biết ơn sâu sắc đến gia đình. Gia đình là hậu phương vững chắc và nguồn động lực to lớn thúc đẩy chúng tôi hoàn thành tốt khóa luận này.

MỤC LỤC

LỜI CẢM ƠN	xii
TÓM TẮT	xix
1 TỔNG QUAN	1
1.1 Bài toán tấn công hộp đen thừa lên mô hình thị giác máy tính	1
1.1.1 Mô tả bài toán	1
1.1.2 Ngữ cảnh	3
Tấn công hộp trắng (White-box attack)	3
Tấn công hộp đen (Black-box attack)	3
1.1.3 Độ đo	4
1.2 Phạm vi và mục tiêu nghiên cứu	5
1.2.1 Phạm vi	5
1.2.2 Mục tiêu	5
1.3 Nội dung thực hiện	6
1.4 Đóng góp của khóa luận	6
1.5 Cấu trúc khóa luận	7
2 CÁC CÔNG TRÌNH LIÊN QUAN VÀ KIẾN THỨC NỀN TẢNG	8
2.1 Các công trình liên quan	8
2.1.1 Các mô hình phát hiện đối tượng tiêu biểu	8
2.1.2 Tấn công đối kháng trên các mô hình phát hiện đối tượng . .	9
2.1.3 Tấn công đối kháng hộp đen thừa	10
2.2 Các kiến thức nền tảng	11
2.2.1 Bài toán tấn công lên các mô hình thị giác máy tính	11
Tấn công lên các mô hình phân loại hình ảnh:	12
Tấn công lên các mô hình phát hiện đối tượng:	14

2.2.2	Thuật giải di truyền (Genetic Algorithm - GA):	18
3	CÁC PHƯƠNG PHÁP ĐỀ XUẤT	22
3.1	Không gian giá trị của nhiễu	23
3.2	Sử dụng GA để tìm kiếm nhiễu	23
3.2.1	Quần thể và cá thể	23
Cá thể trong tấn công mô hình Image Classification	25	
Cá thể trong tấn công mô hình Object Detection	26	
3.2.2	Khởi tạo quần thể	26
3.2.3	Lai ghép	27
3.2.4	Đột biến	28
3.2.5	Đánh giá	31
3.2.6	Chọn lọc	35
3.2.7	Sử dụng Multi-objective Elitist Archive	36
3.3	Hàm mất mát	37
3.3.1	Hàm mất mát cho tấn công mô hình image classification . . .	38
3.3.2	Hàm mất mát cho tấn công mô hình object detection	39
4	THỰC NGHIỆM	43
4.1	Thiết lập thực nghiệm	43
4.1.1	Các mô hình mục tiêu	43
Các mô hình image classification	44	
Các mô hình object detection	44	
4.1.2	Bộ dữ liệu được sử dụng	44
Tấn công lên các mô hình image classification	44	
Tấn công lên các mô hình object detection	45	
4.1.3	Độ đo đánh giá hiệu năng tấn công	45
Tỉ lệ tấn công thành công	46	
Giá trị l_p trung bình	46	
mAP@0.50	46	
Số lần gọi mô hình trung bình	49	
4.1.4	Thiết lập các phương pháp so sánh và các tham số	49
Các biến thể của phương pháp đề xuất	49	

Tấn công lên <i>image classification</i>	50
Tấn công lên <i>object detection</i>	52
4.2 Kết quả thực nghiệm	52
4.2.1 Tấn công các mô hình <i>image classification</i>	52
Các phương pháp hộp trắng	52
Các phương pháp hộp đen	53
4.2.2 Tấn công các mô hình <i>object detection</i>	55
5 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	62
5.1 Kết luận	62
5.2 Hướng phát triển	63
DANH MỤC CÔNG BỐ KHOA HỌC	64
PHỤ LỤC	65
TÀI LIỆU THAM KHẢO	67

DANH SÁCH HÌNH VẼ

2.1	Minh họa cho tấn công thành công lên mô hình image classification. Nhiều (Perturbation) làm cho mô hình phân loại sai lớp của hình ảnh (từ Stop Sign thành Yield Sign).	14
2.2	YOLOv8 dự đoán nhiều hơn một bounding box cho một đối tượng là con ngựa. (<i>Bức ảnh ví dụ được lấy từ bộ dữ liệu PascalVoc2007 [6]</i>)	17
2.3	Minh họa cho công thức tính IoU.	17
2.4	Ví dụ về tấn công thành công lên mô hình object detection. Mô hình vẫn phát hiện đúng vị trí các đối tượng so với bức ảnh gốc, nhưng lớp của một đối tượng bị dự đoán sai (cat thành bear).	18
2.5	Ví dụ về tấn công không thành công lên mô hình object detection. . .	19
2.6	Sơ đồ thuật giải di truyền	21
3.1	Minh họa quá trình lai ghép cho tấn công các mô hình image classification; $offspring_1$ và $offspring_2$ là các cá thể con được tạo ra sau quá trình lai ghép. Các vị trí nhiều được khoanh tròn màu đỏ là những vị trí bị thay thế, các vị trí nhiều được khoanh màu xanh lá là những vị trí được chọn để thay thế các vị trí khoanh đỏ , các vị trí nhiều được khoanh màu xanh lam là những vị trí không bị tác động trong quá trình lai ghép.	29
3.2	Minh họa quá trình đột biến một cá thể cho tấn công các mô hình image classification, $offpring$ là cá thể mới được tạo ra sau đột biến. Các vị trí nhiều được khoanh tròn màu đỏ là những vị trí bị thay thế, các vị trí nhiều được khoanh màu xanh lá là những vị trí mới thay thế cho những vị trí khoanh đỏ , các vị trí nhiều được khoanh màu xanh lam là những vị trí không bị tác động trong quá trình đột biến. . . .	32
4.1	Minh họa cho các trường hợp TP , FP , FN với ngưỡng IoU là $\delta = 0.5$. .	48

- 4.2 Kết quả dự đoán của mô hình DINO trên các bức ảnh chưa bị thêm nhiễu (cột trái) và trên các bức ảnh đã bị thêm nhiễu (cột phải). Chỉ có bức ảnh ở hàng đầu tiên (tính từ trên xuống) được xem là tấn công thành công. (*Tất cả các bức ảnh gốc được lấy từ tập kiểm tra của bộ dữ liệu PascalVoc2007 [6]*)

DANH SÁCH BẢNG

4.1	Tóm tắt đặc điểm của các mô hình được chọn làm mục tiêu tấn công	43
4.2	Các kết quả dự đoán cho đối tượng <i>con mèo</i> cùng với các giá trị <i>Precision</i> và <i>Recall</i> theo thứ tự.	48
4.3	ASR và giá trị l_p -norms ($p = 0, 2$) trung bình của các phương pháp tấn công lên mô hình AT1.	58
4.4	ASR và giá trị l_p -norms ($p = 0, 2$) trung bình của các phương pháp tấn công lên mô hình AT2.	58
4.5	ASR và giá trị l_p -norms ($p = 0, 2$) trung bình của các biến thể GA lên AT1. Với mục tiêu tối ưu là hàm mất mát và giá trị l_2 thay vì l_0	59
4.6	ASR và giá trị l_p -norms ($p = 0, 2$) trung bình của các biến thể GA lên AT2. Với mục tiêu tối ưu là hàm mất mát và giá trị l_2 thay vì l_0	59
4.7	Tỉ lệ tấn công thành công (ASR) và số lần gọi mô hình trung bình (Queries) của các phương pháp tấn công lên các mô hình <i>object detection</i> , với tấn công thành công được xét theo điều kiện 2.8.	60
4.8	Tỉ lệ tấn công thành công (ASR) của các phương pháp tấn công lên các mô hình <i>object detection</i> , với tấn công thành công được xét theo điều kiện 2.7.	60
4.9	Các kết quả dự đoán (mAP@0.50) của các mô hình nạn nhân lên những bức ảnh gốc và những bức ảnh được tạo ra bởi các phương pháp tấn công khác nhau.	60
1	ASR và giá trị l_p -norms ($p = 0, 2$) trung bình của các phương pháp tấn công lên mô hình AT1. (*_w/o_crossover là phương pháp * sau khi loại bỏ bước crossover; *_w/o_mutation là phương pháp * sau khi loại bỏ bước mutation)	66

2	ASR và giá trị l_p -norms ($p = 0, 2$) trung bình của các phương pháp tần công lên mô hình AT2. (*_w/o_crossover là phương pháp * sau khi loại bỏ bước crossover; *_w/o_mutation là phương pháp * sau khi loại bỏ bước mutation)	66
3	ASR và giá trị l_p -norms ($p = 0, 2$) trung bình của phương pháp tần công GA ($T = 2$) lên mô hình AT1 với các số lượng pixel tối đa được thay đổi khác nhau.	66
4	ASR và giá trị l_p -norms ($p = 0, 2$) trung bình của phương pháp tần công GA ($T = 2$) lên mô hình AT2 với các số lượng pixel tối đa được thay đổi khác nhau.	66

TÓM TẮT

Trong những năm qua, các mô hình thị giác máy tính vẫn luôn nhận được sự quan tâm đáng kể nhờ những ứng dụng nổi bật của chúng ở nhiều lĩnh vực trong đời sống. Tuy nhiên, sự gia tăng nhanh chóng các ứng dụng này, đặc biệt là trong những lĩnh vực cần độ an toàn, tin cậy và chính xác cao như xe tự hành, an ninh hay bảo mật, đòi hỏi các mô hình được sử dụng phải có khả năng chống lại được những cuộc tấn công có chủ đích nhắm vào chúng. Tấn công đối kháng (Adversarial Attack) là một phương thức tấn công lên các mô hình thị giác máy tính nhằm mục đích khiến cho các mô hình này đưa ra các dự đoán sai lệch. Loại tấn công này có thể được thực hiện tương đối dễ dàng bằng cách thêm những nhiễu rất nhỏ mà mắt thường khó có thể phân biệt được vào bức ảnh đầu vào. Và sự thật, rất nhiều nghiên cứu đã chứng minh rằng, phương thức tấn công tưởng chừng như đơn giản này thật sự có thể đánh lừa được các mô hình, tạo nên một mối đe dọa đến độ chính xác và tin cậy của chúng [4, 8, 15, 18, 22].

Dựa vào thông tin mà ta có được từ mô hình bị tấn công, các phương pháp tấn công đối kháng được chia làm hai loại chính là tấn công hộp trắng (white-box attack) và tấn công hộp đen (black-box attack). Với tấn công hộp trắng, ta giả định rằng toàn bộ thông tin của mô hình từ kiến trúc, tham số hay gradient đều có thể được biết và khai thác bởi kẻ tấn công. Nhờ đây, các phương pháp tấn công tận dụng thông tin về gradient có thể được sử dụng như Fast Gradient Sign Method (viết tắt là FGSM) [8] hay Projected Gradient Descent (viết tắt là PGD) [15] để nhanh chóng tạo ra những nhiễu với tỉ lệ tấn công thành công rất cao. Dù vậy, những phương pháp tấn công hộp trắng lại khó có thể thực hiện do những thông tin mà ta giả định trước đó thường sẽ không thể truy cập được trong thực tế. Mặt khác, các phương pháp tấn công hộp đen sẽ được triển khai trong điều kiện gần với thực tiễn hơn là kẻ tấn công chỉ biết được dữ liệu đầu vào và kết quả đầu ra mà mô hình trả về. Dĩ nhiên rằng, với lượng thông tin được biết bị hạn chế hơn, các phương pháp tấn công hộp đen sẽ khó đạt được tỉ lệ tấn công thành công cao như các phương pháp hộp trắng. Tuy nhiên, các nghiên cứu gần đây trên những phương pháp hộp đen đang dần rút ngắn khoảng cách này, và có hai hướng tiếp cận đang được chú ý là sử dụng các mô hình thay thế (surrogate model) [1, 17] và sử dụng các phương pháp tìm kiếm

không cần đến gradient (gradient-free), mà nổi bật trong đó là các thuật toán tiến hóa (evolutionary algorithms) [4, 18, 22].

Các phương pháp tấn công hộp đen sử dụng thuật toán tiến hóa thu hút sự chú ý của chúng tôi không những bởi có khả năng áp dụng vào các cuộc tấn công trong thực tế cao, mà còn vì sự đơn giản nhưng hiệu quả mà các thuật toán tiến hóa mang lại trong tìm kiếm những nhiễu có thể đánh lừa được các mô hình. Trong khóa luận tốt nghiệp này, chúng tôi sẽ nghiên cứu việc áp dụng thuật toán tiến hóa trong bài toán tấn công hộp đen. Cụ thể hơn, chúng tôi tập trung vào một biến thể của tấn công hộp đen lên các mô hình thị giác máy tính là tấn công hộp đen thưa (sparse black-box adversarial attack), trong đó, mục tiêu là chỉ thay đổi một tỉ lệ nhỏ các điểm ảnh (pixel) trên bức hình gốc để tạo ra một bức hình mới khiến cho mô hình đưa ra các dự đoán sai lệch. Chúng tôi đề xuất một phương pháp tấn công sử dụng thuật giải di truyền (Genetic Algorithm - ký hiệu là GA) và hai hàm mất mát mới để tấn công lên hai loại mô hình thị giác máy tính là mô hình phân loại hình ảnh (image classification) và phát hiện đối tượng (object detection). Trong đó, hai hàm mất mát mà chúng tôi đề xuất sẽ hướng tới ngữ cảnh chỉ hoàn toàn sử dụng các thông tin được phép truy cập là dữ liệu đầu vào, và kết quả mà mô hình trả về.

Để kiểm chứng tính hiệu quả của phương pháp mà chúng tôi đề xuất, chúng tôi tiến hành thực nghiệm so sánh với các phương pháp tấn công đã được nghiên cứu trước đó, bao gồm cả các phương pháp tấn công hộp trắng và hộp đen. Các mục tiêu tấn công sẽ bao gồm hai mô hình phân loại hình ảnh được huấn luyện để chống lại các cuộc tấn công đối kháng [9, 10], và các mô hình phát hiện đối tượng được ứng dụng rộng rãi ngày nay là YOLO [12], DETR [2] và DINO[24].

Chương 1

TỔNG QUAN

Trong chương này, trước hết chúng tôi sẽ đưa ra cái nhìn tổng quan về bài toán Tấn công hộp đen thưa trên mô hình thị giác máy tính, mô tả nhiệm vụ cũng như mục tiêu của bài toán. Tiếp theo, chúng tôi sẽ làm rõ mục tiêu và phạm vi nghiên cứu trong khóa luận này. Cuối cùng, chương này sẽ tóm tắt các nội dung đã được thực hiện trong quá trình nghiên cứu, nêu ra những đóng góp chính của khóa luận và đồng thời bô cục của khóa luận cũng sẽ được trình bày chi tiết trong phần này để giúp người đọc nắm rõ cách tổ chức và cấu trúc nội dung.

1.1 Bài toán tấn công hộp đen thưa lên mô hình thị giác máy tính

1.1.1 Mô tả bài toán

Với sự phát triển của học sâu (Deep Learning), các mạng nơ-ron sâu (Deep Neural Networks - DNNs) ngày càng thể hiện độ hiệu quả của mình trong nhiều bài toán khác nhau, đặc biệt là trong lĩnh vực trí tuệ nhân tạo, với hai lĩnh vực đáng chú ý nhất là thị giác máy tính và xử lý ngôn ngữ tự nhiên. Tuy nhiên, nhiều nghiên cứu đã chỉ ra rằng các mô hình rất nhạy cảm với nhiễu: chỉ cần thay đổi một vài chi tiết nhỏ, không làm mất đi ý nghĩa của dữ liệu, cũng có thể khiến cho mô hình hoạt động sai.

Lấy ví dụ với các mô hình phân loại ảnh (image classification), chỉ cần thay đổi ảnh gốc ở một số pixels - thậm chí chỉ một pixel [18] - cũng có thể khiến các mô hình này đưa ra dự đoán khác với dự đoán trên ảnh gốc. Điều này đặt ra một mối lo ngại

nghiêm trọng về tính an toàn của các mô hình học sâu khi mà vai trò ngày càng lớn của chúng đã xuất hiện trong các tác vụ đòi hỏi mức độ chính xác cực cao như điều khiển xe tự hành hoặc khóa sinh trắc học. Vì vậy, việc khảo sát độ an toàn cũng như tăng cường khả năng chống chịu đối với các dữ liệu bị nhiễu trên các DNNs là cần thiết để đảm bảo chúng hoạt động tốt trong các nhiệm vụ được giao.

Một trong những cách để thực hiện việc này chính là tạo ra các dữ liệu nhiễu (dữ liệu đã bị chỉnh sửa nhưng không làm thay đổi ý nghĩa ban đầu) khiến cho mô hình đưa ra dự đoán sai. Các dữ liệu bị nhiễu này được gọi là các **adversarial examples**. Mục tiêu của bài toán tấn công (**adversarial attack**) hướng đến việc thiết kế các thuật toán hiệu quả phục vụ cho việc tạo ra những **adversarial examples** tốt (thay đổi ít nhưng lại làm cho mô hình đưa ra dự đoán có sự sai lệch cao so với dự đoán trên dữ liệu gốc). Bên cạnh đó, nghiên cứu về adversarial attack còn nhằm chỉ ra điểm yếu của các mô hình hiện nay trước các phương thức tấn công có chủ đích khác nhau, từ đó thúc đẩy các nghiên cứu mới để khắc phục những điểm yếu này.

Khi các mô hình được chọn để tấn công là các mô hình thị giác máy tính có đầu vào là hình ảnh thì bài toán có thể được mô hình hóa như sau:

- **Đầu vào:** Mô hình thị giác máy tính \mathcal{M} , ảnh gốc $\mathbf{X}^{h \times w \times 3}$, với:

- \mathcal{M} là mô hình thị giác máy tính nhận dữ liệu đầu vào là một hình ảnh có 3 kênh màu và đầu ra $\mathcal{M}(.)$ khác nhau tùy theo nhiệm vụ của mô hình.
- $\mathbf{X}^{h \times w \times 3} \in [0, 1]^{h \times w \times 3}$ là một hình ảnh có 3 kênh màu với chiều cao là h và chiều rộng là w và là một đầu vào hợp lệ của \mathcal{M} .

- **Đầu ra:** Hình ảnh $\hat{\mathbf{X}}$ thỏa mãn:

- $\hat{\mathbf{X}}^{h \times w \times 3} \in [0, 1]^{h \times w \times 3}$: $\hat{\mathbf{X}}$ là ảnh hợp lệ và có kích thước bằng ảnh gốc \mathbf{X} .
- $\text{SUCCESS}(\mathcal{M}(\mathbf{X}), \mathcal{M}(\hat{\mathbf{X}})) = \text{true}$: với $\text{SUCCESS}(A, B)$ là hàm kiểm tra nếu đầu ra **ứng với dữ liệu gốc** là A và đầu ra **ứng với dữ liệu nhiễu** là B thì **dữ liệu nhiễu** có tính là một adversarial example đối với **dữ liệu gốc** hay không.

1.1.2 Ngữ cảnh

Bài toán **tấn công đối kháng** sẽ được triển khai trên hai ngữ cảnh khác nhau dựa vào việc thuật toán có khả năng truy cập vào mô hình hay không:

Tấn công hộp trắng (White-box attack)

Thuật toán biết và có thể sử dụng toàn bộ thông tin của mô hình như thông tin về kiến trúc, tham số, gradient,... Khả năng truy cập vào các thông tin về mô hình giúp cho các thuật toán được triển khai trong ngữ cảnh hộp trắng tìm ra được các adversarial examples có chất lượng tốt hơn so với các phương pháp hộp đen. Hầu hết các phương pháp tấn công hộp trắng sẽ tận dụng thông tin về gradient để tối đa hóa khả năng gây lỗi với cùng một lượng thay đổi. Tuy nhiên, các phương pháp tấn công hộp trắng thường không thể áp dụng trong các tình huống thực tế vì trong những tình huống như vậy chúng ta gần như không thể truy cập vào các thông tin của mô hình.

Tấn công hộp đen (Black-box attack)

Thuật toán không khai thác và không được sử dụng bất cứ tri thức gì về mô hình (kiến trúc, trọng số, gradient,...) trừ không gian đầu vào (định dạng của đầu vào) và đôi khi là không gian đầu ra. Trong ngữ cảnh này, mô hình chỉ có thể được sử dụng để truy vấn đầu ra với mỗi đầu vào tương ứng. Vì không thể truy cập vào nhiều thông tin như ngữ cảnh hộp trắng, nên tấn công hộp đen thường khó có thể tạo ra các adversarial examples tốt như tấn công hộp trắng được. Tuy nhiên, các thuật toán này lại phù hợp với các tình huống thực tế khi mà ngày càng nhiều các dịch vụ API cho phép chúng ta gửi đầu vào và nhận lại đầu ra.

Các phương pháp tấn công hộp đen bao gồm nhưng không giới hạn:

- **Tấn công chuyển đổi (Transfer-based attack):** cố gắng xây dựng và huấn luyện một mô hình giả lập (surrogate model) càng giống mô hình gốc càng tốt bằng cách sử dụng nhiều mẫu đầu vào - đầu ra khác nhau trên mô hình được tấn công. Sau khi mô hình giả lập được tạo ra, việc tấn công sẽ được diễn ra trên mô hình giả lập này. Kết quả tấn công trên mô hình giả lập sau đó sẽ được dùng để tấn công ngược lại mô hình gốc.

- Tấn công không dựa trên gradient (Gradient-free attack): sử dụng các phương pháp tối ưu hóa không cần gradient, như thuật toán tiến hóa hoặc tìm kiếm ngẫu nhiên, để tìm ra các mẫu tấn công hiệu quả.

1.1.3 Độ đo

Bên cạnh mục tiêu chính là tạo ra được dữ liệu bị nhiễu khiến cho mô hình đưa ra dự đoán không chính xác, thì dữ liệu bị nhiễu còn cần đạt yêu cầu không được khác quá nhiều so với dữ liệu ban đầu. Cụ thể, sự khác nhau của ảnh gốc \mathbf{X} và adversarial example $\hat{\mathbf{X}}$ (tương ứng với X) sẽ được đo theo chuẩn l_p như sau:

$$\|\mathbf{x} - \hat{\mathbf{x}}\|_p = \left(\sum_{i \in \mathbb{Z}_{[0,h)} \times \mathbb{Z}_{[0,w)} \times \mathbb{Z}_{[0,3)}} (\mathbf{x}_i - \hat{\mathbf{x}}_i)^p \right)^{1/p}.$$

Với tham số p thường được chọn là:

- 1 hoặc 2: yêu cầu tổng độ thay đổi trên ảnh gốc càng nhỏ càng tốt.
- ∞ : yêu cầu độ thay đổi lớn nhất trên ảnh gốc càng nhỏ càng tốt.
- 0: yêu cầu số pixel bị thay đổi càng nhỏ càng tốt (còn được gọi là tấn công thưa - sparse attack).

Tùy theo độ đo khác nhau mà người ta sẽ thiết kế những chiến lược tấn công khác nhau để tạo ra được adversarial examples tốt theo độ đo đó. Một adversarial example A tốt hơn B theo độ đo l_p sẽ không có nghĩa là nó cũng sẽ tốt hơn B trên độ đo $l'_p \neq l_p$. Do đó việc so sánh độ tốt của các adversarial examples phải được thực hiện trên cùng một độ đo, và ngược lại, độ đo l_p cũng là một trong các tiêu chuẩn mà chúng ta dựa vào để có thể tạo ra được các adversarial examples tốt (tốt theo khía cạnh được đại diện bởi độ đo).

1.2 Phạm vi và mục tiêu nghiên cứu

1.2.1 Phạm vi

Trong khóa luận này, chúng tôi tập trung nghiên cứu bài toán tấn công đối kháng trong ngữ cảnh tấn công hộp đen, với các mô hình mục tiêu thuộc lĩnh vực thị giác máy tính. Cụ thể, các bài toán được lựa chọn bao gồm: phân loại hình ảnh (image classification) và phát hiện đối tượng (object detection).

Hình thức tấn công mà chúng tôi áp dụng là tấn công thưa, sử dụng độ đo l_0 để đánh giá mức độ khác biệt giữa ảnh gốc và ảnh đã bị thêm nhiễu. Nói cách khác, chúng tôi sẽ cố gắng tối thiểu hóa số lượng pixel bị thay đổi trong ảnh, trong khi vẫn đảm bảo hiệu quả tấn công.

Về phương pháp tiếp cận, chúng tôi chọn các phương pháp tấn công không dựa trên gradient. Cụ thể, chúng tôi sẽ sử dụng giải thuật di truyền như công cụ chính để tạo ra các adversarial examples hiệu quả, đáp ứng các yêu cầu của bài toán trong ngữ cảnh tấn công hộp đen.

1.2.2 Mục tiêu

Khóa luận đặt ra mục tiêu chính là thiết kế một thuật toán dựa trên giải thuật di truyền để giải quyết bài toán tấn công đối kháng trong các mô hình thị giác máy tính. Thuật toán được phát triển cần đạt được độ hiệu quả cao hay thỏa mãn các tiêu chí sau:

- Hiệu quả trong việc tạo ra các adversarial examples: trong hình thức tấn công thưa có nghĩa là tối thiểu hóa số lượng pixel bị thay đổi.
- Áp dụng được trên các mô hình mục tiêu thuộc hai bài toán: phân loại ảnh và phát hiện đối tượng.
- Đảm bảo khả năng thực hiện trong ngữ cảnh hộp đen: không yêu cầu truy cập vào thông tin gradient hoặc cấu trúc mô hình mà chỉ dựa vào các thông tin được biết là bức ảnh đầu vào và kết quả mô hình trả về.

Ngoài ra, chúng tôi hy vọng rằng khóa luận sẽ mang lại đóng góp trong việc nâng cao hiểu biết về bài toán tấn công đối kháng, đặc biệt là về các phương pháp

tấn công hộp đen thưa trên các mô hình thị giác máy tính, cũng như tạo tiền đề cho các nghiên cứu có liên quan được thực hiện trong tương lai.

1.3 Nội dung thực hiện

Để đạt được các mục tiêu nghiên cứu đã đề ra, chúng tôi đã triển khai và hoàn thành các nội dung sau đây:

- Tìm hiểu và tổng hợp các công trình nghiên cứu liên quan đến bài toán tấn công đối kháng nhằm xây dựng nền tảng lý thuyết và định hướng phát triển phương pháp mới.
- Đề xuất một phương pháp tấn công hiệu quả, dựa trên giải thuật di truyền, phù hợp với ngữ cảnh tấn công hộp đen và hình thức tấn công thưa đã được xác định trong phạm vi nghiên cứu.
- Tiến hành các thử nghiệm tấn công bằng phương pháp đã đề xuất trên nhiều mô hình thị giác máy tính khác nhau để đánh giá hiệu quả của phương pháp. Các mô hình được chọn bao gồm các mô hình phân loại hình ảnh (image classification) và phát hiện đối tượng (object detection).
- So sánh kết quả thực nghiệm của phương pháp đề xuất với các phương pháp tấn công hiện có. Phân tích kết quả so sánh để từ đó rút ra những điểm mạnh và hạn chế của phương pháp.

1.4 Đóng góp của khóa luận

Từ những kết quả đạt được trong quá trình thực hiện, thông qua khóa luận này, chúng tôi có những đóng góp sau đây:

- Phát triển một phương pháp tấn công hộp đen dựa trên giải thuật di truyền, đồng thời thiết kế hai hàm matsu để có thể tấn công vào các mô hình phân loại hình ảnh và phát hiện đối tượng sao cho thỏa mãn được mục tiêu đã đề ra.

- Cung cấp thông tin chi tiết về các bộ dữ liệu và các mô hình được sử dụng trong quá trình thực nghiệm, tạo điều kiện thuận lợi cho các nghiên cứu tiếp theo trong việc tái lập và so sánh kết quả.
- Chia sẻ mã nguồn thực nghiệm, giúp hỗ trợ trong việc kiểm chứng và mở rộng kết quả của khóa luận.¹

1.5 Cấu trúc khóa luận

Khóa luận của chúng tôi được chia thành 5 chương chính như sau:

- Chương 1: Tổng quan.
- Chương 2: Các công trình liên quan và kiến thức nền tảng.
- Chương 3: Các phương pháp đề xuất.
- Chương 4: Thực nghiệm.
- Chương 5: Kết luận và hướng phát triển.

¹Mã nguồn thực nghiệm: [github](#)

Chương 2

CÁC CÔNG TRÌNH LIÊN QUAN VÀ KIẾN THỨC NỀN TẢNG

Trong chương này, chúng tôi sẽ trình bày các công trình nghiên cứu liên quan đến các hướng tiếp cận hiện tại của bài toán tấn công đối kháng trên các mô hình thị giác máy tính và các kiến thức nền tảng được sử dụng xuyên suốt khóa luận này. Phần 2.1 trình bày các mô hình phát hiện đối tượng tiêu biểu hiện nay. Đồng thời, trình bày tổng quan các nghiên cứu đã có về bài toán tấn công đối kháng lên các mô hình phát hiện đối tượng, cũng như các nghiên cứu đã có về tấn công đối kháng hộp đen thưa. Tiếp đó, phần 2.2 sẽ trình bày về thuật giải di truyền - thuật toán sẽ được chúng tôi áp dụng trong bài toán.

2.1 Các công trình liên quan

2.1.1 Các mô hình phát hiện đối tượng tiêu biểu

Hiện nay, các mô hình phát hiện đối tượng được xây dựng chủ yếu dựa trên hai kiến trúc chính: mạng nơ-ron tích chập (Convolutional Neural Network - CNN) và Transformer cho thị giác máy tính (Vision Transformer - ViT).

Đối với các mô hình CNN, nhóm mô hình YOLO [12, 19, 20, 21] là những đại diện tiêu biểu, thường được sử dụng rộng rãi trong các bài toán nhận diện vật thể theo thời gian thực. Sự phổ biến của các mô hình này xuất phát từ khả năng cân bằng tốt giữa tốc độ xử lý và độ chính xác, giúp chúng trở thành lựa chọn tối ưu trong nhiều ứng dụng thực tiễn.

Song song với đó, các mô hình dựa trên ViT cũng thu hút sự chú ý mạnh mẽ từ cộng đồng nghiên cứu nhờ tiềm năng vượt trội. Một số mô hình đáng chú ý bao gồm DETR [2] và DINO [24], đã chứng minh được hiệu quả cao trong nhiều bài toán thị giác máy tính phức tạp.

Tuy nhiên, mặc dù đạt được hiệu năng cao và được sử dụng rộng rãi, cả hai nhóm mô hình này vẫn đối mặt với những thách thức lớn liên quan đến độ tin cậy và khả năng chống chịu trước các tấn công đối kháng. Đây là vấn đề quan trọng cần được nghiên cứu sâu hơn để đảm bảo tính an toàn và ổn định trong các ứng dụng thực tế, đặc biệt trong các lĩnh vực đòi hỏi độ tin cậy cao như an ninh và giám sát.

2.1.2 Tấn công đối kháng trên các mô hình phát hiện đối tượng

Bài toán phát hiện đối tượng vẫn luôn nhận được rất nhiều sự quan tâm từ cộng đồng nguyên cứu trong những năm qua. Hơn nữa gần đây, nhiều phương pháp tấn công đã được đề xuất nhằm đánh giá khả năng chống chịu của các mô hình phát hiện đối tượng trước tấn công đối kháng. Tuy nhiên, đa số các phương pháp này được thực hiện trong bối cảnh hộp trống, đòi hỏi thông tin chi tiết về cấu trúc và tham số của mô hình mục tiêu, điều này khó có thể áp dụng trong các ứng dụng thực tiễn [23].

Ngoài những phương pháp được triển khai trong bối cảnh hộp trống, một số phương pháp tấn công trong ngữ cảnh hộp đen cũng đã được phát triển để có thể đáp ứng với các tình huống thực tế. Hầu hết các phương pháp này cần sử dụng các mô hình thay thế (surrogate models) để giả lập hành vi của mô hình mục tiêu, từ đó thực hiện tấn công. Điều này dẫn đến gia tăng đáng kể chi phí tính toán cũng như thời gian thực hiện tấn công [1, 17].

Việc tối ưu hóa các phương pháp tấn công hộp đen để đạt hiệu quả cao hơn trong khi giảm thiểu chi phí tính toán vẫn là một hướng nghiên cứu được chú ý trong lĩnh vực tấn công đối kháng đối lên các mô hình phát hiện đối tượng nói riêng và các mô hình DNNs nói chung.

2.1.3 Tấn công đối kháng hộp đen thưa

Hiện nay đã có nhiều nghiên cứu về bài toán tấn công hộp đen thưa lên các mô hình phân loại hình ảnh và đạt được nhiều thành tựu. Ví dụ, nghiên cứu [18] đã chỉ ra rằng chỉ cần thay đổi một pixel cũng có thể khiến các mô hình bị tấn công đưa ra dự đoán sai lệch. Các cơ chế tìm kiếm cục bộ (local search) và leo đồi (hill-climbing) cũng được chứng minh là hiệu quả trong việc tìm ra các adversarial examples tốt (chuẩn l_0 trong giới hạn quy định) chỉ bằng một lượng ít các lần truy vấn mô hình [4].

Tuy nhiên, một phương pháp tấn công có khả năng thành công cao thường đi kèm với việc các adversarial examples có chất lượng không tốt, được thể hiện qua giá trị chuẩn l_0 lớn. Để đạt được đồng thời tỉ lệ tấn công thành công cao và chất lượng tốt của các adversarial examples, Li cùng các cộng sự [22] đã sử dụng Tối ưu hóa Đa Mục tiêu (Multi-Objective Optimization - MOO) để tối ưu hóa đồng thời hai mục tiêu: hàm mất mát (phục vụ cho việc tìm ra các adversarial examples) và độ lớn của chuẩn l_0 (đại diện cho độ lớn của sự thay đổi trên các adversarial examples).

Mặc dù được gọi là tấn công hộp đen, nhưng chúng tôi cho rằng phần lớn các phương pháp được đề cập ở trên nên được phân loại là tấn công hộp xám (gray-box) vì chúng được thực hiện với giả định rằng logits (xác suất của tất cả các lớp) là có thể truy cập được. Với giả định này, các phương pháp đó có thể không thực tế trong các tình huống thực tế, vì hầu hết các mô hình thường chỉ trả về xác suất cao nhất trong tất cả các lớp.

2.2 Các kiến thức nền tảng

2.2.1 Bài toán tấn công lên các mô hình thị giác máy tính

Bản chất của *tấn công đối kháng* (**adversarial attack**)¹ lên các mô hình *thị giác máy tính* (**computer vision**) là đi giải một bài toán tối ưu hóa:

$$\begin{aligned} \tilde{\mathbf{X}} = \operatorname{argmin}_{\hat{\mathbf{X}} \in \mathbb{X}} & \| \mathbf{X} - \hat{\mathbf{X}} \|_p \\ \text{s.t: } & \text{SUCCESS}_{\mathbb{Y}}(\mathcal{M}(\mathbf{Y}), \mathcal{M}(\hat{\mathbf{X}})) = \text{true} \end{aligned} \quad (2.1)$$

Trong đó:

- \mathbf{X} là dữ liệu (ảnh) gốc.
- \mathcal{M} là **mô hình bị tấn công**, trong phạm vi khóa luận này thì \mathcal{M} có thể là mô hình phân loại ảnh hoặc mô hình phát hiện đối tượng.
- $\mathcal{M}(\mathbf{X})$ là **đầu ra** của mô hình \mathcal{M} ứng với \mathbf{X} .
- \mathbb{X}, \mathbb{Y} là **không gian đầu vào** và **không gian đầu ra** của mô hình \mathcal{M} .
- $\text{SUCCESS}_{\mathbb{Y}}(A, B)$ là **hàm kiểm tra** mô hình có bị tấn công thành công hay không nếu đầu ra của mô hình bị thay đổi từ A thành B .
- p là tham số đại diện cho độ đo sự khác biệt giữa dữ liệu gốc (ảnh gốc) và dữ liệu sau khi bị thay đổi (ảnh bị thêm nhiễu), trong khóa luận này chúng tôi sử dụng $p = 0$.

Thay vì đi tìm trực tiếp ảnh bị thay đổi, ta có thể đi tìm độ thay đổi ϵ trên ảnh gốc. Tất nhiên số chiều của ϵ phải bằng với số chiều của \mathbf{X} và ϵ này được gọi là **nhiễu**.

¹Từ này trở về sau, "tấn công đối kháng" và "tấn công đối kháng hộp đen thưa" sẽ được viết gọn là "tấn công"

Khi đó bài toán trở thành:

$$\begin{aligned} \tilde{\epsilon} &= \operatorname{argmin}_{\epsilon} \|\epsilon\|_p && (2.2) \\ \text{s.t.: } & \text{SUCCESS}_{\mathbb{Y}}(\mathcal{M}(\mathbf{X}), \mathcal{M}(\mathbf{X} + \epsilon)) = \text{true} \\ & \mathbf{X} + \epsilon \in \mathbb{X} \end{aligned}$$

Nhìn chung, mục tiêu của việc tấn công vào một bức ảnh là để tạo ra một bức ảnh khác có cùng nội dung với bức ảnh gốc khi quan sát bằng mắt thường nhưng lại bị mô hình xem như hai bức ảnh có nội dung khác nhau. Hơn nữa, với bài toán 2.2 thì việc tìm ra không gian con chứa các ảnh tấn công thành công là rất khó, và với một ảnh không tấn công thành công ta cũng không thể biết được nó đã sắp tấn công thành công hay chưa. Do đó, thay vì tối ưu theo bài toán 2.2, ta sẽ đi tìm một lượng nhiễu ϵ có l_p không nhiều hơn θ (đảm bảo ảnh không bị thay đổi quá nhiều) và đánh giá lời giải theo một hàm mất mát $\mathcal{L}(\cdot)$ có quan hệ với việc đầu ra bị thay đổi (giá trị của $\mathcal{L}(\cdot)$ càng thấp thì càng có khả năng tấn công thành công), và điều chỉnh bài toán 2.2 trở thành:

$$\begin{aligned} \tilde{\epsilon} &= \operatorname{argmin}_{\epsilon} \mathcal{L}(\mathcal{M}(\mathbf{X} + \epsilon)) && (2.3) \\ \text{st: } & \|\epsilon\|_p \leq \theta \\ & \mathbf{X} + \epsilon \in \mathbb{X} \end{aligned}$$

Tấn công lên các mô hình phân loại hình ảnh:

Để phân biệt với các mô hình phát hiện đối tượng, chúng tôi sẽ kí hiệu mô hình phân loại ảnh là \mathcal{F} . Mô hình \mathcal{F} nhận đầu vào là một ảnh màu $\mathbf{X} \in [0, 1]^{h \times w \times 3}$ có 3 kênh màu với chiều cao là h và chiều rộng là w . Đầu ra của \mathcal{F} tương ứng với \mathbf{X} là vector $\mathcal{F}(\mathbf{X}) = (f_1, f_2, \dots, f_{|\mathcal{C}|})$ trong đó f_i là xác suất ảnh X thuộc lớp thứ i và \mathcal{C} là tập hợp các lớp mà mô hình được huấn luyện. Đồng thời, ta gọi $\mathcal{F}_i(\mathbf{X}) = f_i$ tức là xác suất ảnh \mathbf{X} thuộc vào lớp i . Nói cách khác, nếu gọi y là lớp mà mô hình sẽ dự đoán thì giá trị của y sẽ được tính bằng công thức:

$$y = \arg \max_i \mathcal{F}_i(\mathbf{X})$$

Để phù hợp với ngữ cảnh tấn công hộp đen, chúng tôi sẽ chỉ sử dụng thông tin về lớp của ảnh đầu vào được mô hình dự đoán y và xác suất f_y ảnh đó thuộc lớp y .

Ngoài ra, chúng tôi cũng sẽ định nghĩa một hàm măt măt \mathcal{L} sẽ được trình bày chi tiết ở Mục 3.3. Nhìn chung, hàm măt măt này có tính chất là nó sẽ nhận giá trị càng nhỏ nếu kết quả đầu ra của mô hình trên bức ảnh bị thêm nhiều là $\mathcal{F}(X + \epsilon)$ càng khác với kết quả đầu ra trên bức ảnh gốc. Việc định nghĩa một hàm măt măt như vậy giúp chúng tôi có thể gián tiếp giải quyết bài toán tấn công thông qua bài toán tối ưu hóa như đã đề cập ở 2.3:

$$\begin{aligned} \tilde{\epsilon} &= \underset{\epsilon}{\operatorname{argmin}} \mathcal{L}(\mathcal{F}(X + \epsilon)) \\ \text{st: } & \| \epsilon \|_0 \leq \theta \\ & X + \epsilon \in [0, 1]^{h \times w \times 3} \end{aligned} \tag{2.4}$$

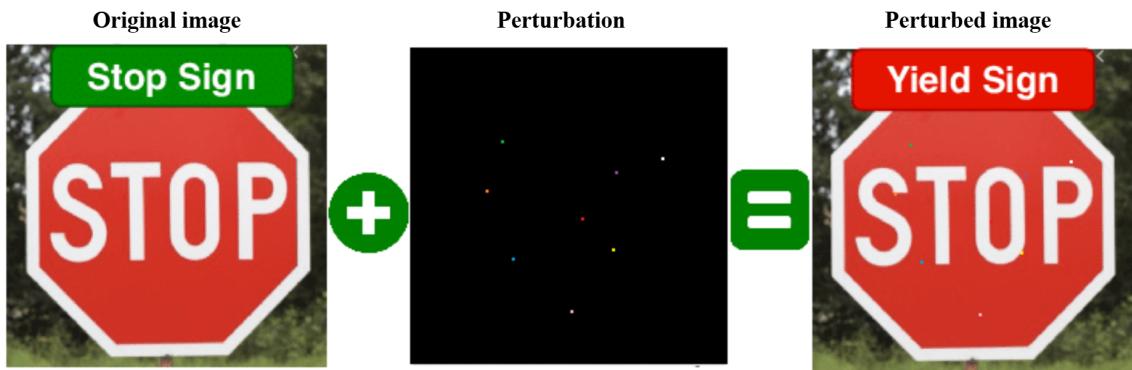
Vì tấn công được thực hiện trong ngữ cảnh tấn công thua nên chúng tôi đo lượng thay đổi bằng chuẩn l_0 , chính xác hơn là số điểm ảnh bị thay đổi. Có nghĩa là, nếu có một điểm ảnh nào đó bị thay đổi nhiều hơn một kênh màu thì ta cũng chỉ xem như điểm ảnh đó đóng góp 1 vào việc tính chuẩn l_0 (thay vì là 2).

Hàm măt măt \mathcal{L} không cho ta biết một nhiễu có tấn công thành công hay không, nó chỉ là một đại lượng có liên quan đến khả năng tấn công thành công của một nhiễu. Vì vậy ta vẫn cần định nghĩa lại như thế nào là tấn công thành công - nói cách khác là định nghĩa hàm SUCCESS. Một nhiễu ϵ sẽ được cho là tấn công thành công lên mô hình phân loại hình ảnh \mathcal{F} đối với bức ảnh X nếu bức ảnh sau khi bị thêm nhiễu có kết quả phân lớp khác với bức ảnh gốc X , cụ thể:

$$\text{SUCCESS}(\mathcal{F}(X), \mathcal{F}(X + \epsilon)) = \begin{cases} \text{true} & \text{nếu } \mathcal{F}'(X) \neq \mathcal{F}'(X + \epsilon) \\ \text{false} & \text{nếu } \mathcal{F}'(X) = \mathcal{F}'(X + \epsilon) \end{cases} \tag{2.5}$$

Với $\mathcal{F}'(X) = \operatorname{argmax}_{i \in \{1, 2, \dots, |\mathcal{C}| \}} \{f_1, f_2, \dots, f_{|\mathcal{C}|} | f_i \in \mathcal{F}(X)\}$

Hình 2.1 là một ví dụ về một trường hợp tấn công thành công. Ảnh gốc được mô hình dự đoán là **Stop Sign**, còn ảnh đã thêm nhiễu thì được dự đoán là **Yield Sign**. Lượng nhiễu được thêm vào cũng rất nhỏ, không làm thay đổi ý nghĩa của ảnh ban



HÌNH 2.1: Minh họa cho tấn công thành công lên mô hình image classification. Nhiều (Perturbation) làm cho mô hình phân loại sai lớp của hình ảnh (từ **Stop Sign** thành **Yield Sign**).

đầu khi nhìn bằng mắt thường.

Tấn công lên các mô hình phát hiện đối tượng:

Ta kí hiệu để \mathcal{D} là một mô hình phát hiện đối tượng để phân biệt với \mathcal{F} là một mô hình phân lớp ảnh. Giống như các mô hình phân lớp ảnh, \mathcal{D} cũng nhận vào một ảnh màu $\mathbf{X} \in [0, 1]^{h \times w \times 3}$. Tuy nhiên, trong khi các mô hình phân lớp trả về duy nhất một nhãn (và xác suất của nhãn đó) thì các mô hình phát hiện đối tượng sẽ trả về kết quả là một danh sách các bộ ba (B, c, f) đại diện cho các vật thể. Trong đó:

- B biểu diễn vị trí (bounding box) của vật thể, là tọa độ của hình chữ nhật bao quanh vật thể.
- c là nhãn (lớp) mà mô hình dự đoán cho vật thể nằm trong bounding box B .
- f là độ tin cậy của dự đoán mà mô hình dành cho đối tượng được phát hiện trong B . Cụ thể hơn, đây là xác suất mô hình dự đoán vật thể đó thuộc vào lớp c .

Ta gọi một bộ ba như vậy là một vật thể, và kí hiệu kết quả dự đoán của mô hình \mathcal{D} đối với bức ảnh đầu vào \mathbf{X} như sau

$$\mathcal{D}(\mathbf{X}) = \{O_i | O_i = (B_i, c_i, f_i), i = \overline{1..n}\}$$

Với n là số đối tượng mô hình phát hiện được.

Tương tự như trường hợp tấn công mô hình phân loại ảnh, ta cũng cần một hàm mất mát $\mathcal{L}(\cdot)$ để đưa bài toán tấn công trở thành bài toán tối ưu hóa như 2.3 (hàm mất mát này sẽ được đề cập chi tiết trong phần 3.3):

$$\begin{aligned}\tilde{\epsilon} &= \underset{\epsilon}{\operatorname{argmin}} \mathcal{L}(\mathcal{D}(\mathbf{X} + \epsilon)) \\ \text{st: } &\|\epsilon\|_0 \leq \theta \\ &\mathbf{X} + \epsilon \in [0, 1]^{h \times w \times 3}\end{aligned}\tag{2.6}$$

Đối với quá trình tấn công lên mô hình phát hiện đối tượng, chúng tôi chia ra làm hai trường hợp tấn công thành công.

Trường hợp tấn công thành công thứ nhất, một nhiễu ϵ được xem là tấn công thành công khi kết quả dự đoán của mô hình lên bức ảnh bị thêm nhiễu có ít nhất một đối tượng có kết quả dự đoán không đúng so với bức ảnh gốc. Dự đoán không đúng ở đây sẽ bao gồm dự đoán sai lớp (**misategorization**), hoặc không dự đoán được vị trí của đối tượng đó (**object vanishing**). Cụ thể, hàm **SUCCESS** trong trường hợp này được định nghĩa như sau:

$$\text{SUCCESS}(\mathcal{D}(\mathbf{X}), \mathcal{D}(\mathbf{X} + \epsilon)) = \begin{cases} \text{true} & \text{nếu } \exists(O \in \mathcal{D}(\mathbf{X})) : \\ & \nexists O' \in \mathcal{D}(\mathbf{X} + \epsilon) : (\text{IoU}(B_O, B_{O'}) \geq \delta) \wedge (c_O = c_{O'}) \\ \text{false} & \text{các trường hợp còn lại} \end{cases}\tag{2.7}$$

Trong đó:

- O và O' là hai đối tượng được phát hiện trong ảnh gốc và ảnh đã có nhiễu.
- $B_O, B_{O'}$ là vị trí của đối tượng O và O'
- $c_O, c_{O'}$ là nhãn của đối tượng O và O'
- $\text{IoU}(B_O, B_{O'})$ là giá trị *Intersection over Union* đối với hai vị trí (bounding box) B_O và $B_{O'}$.

Giá trị IoU dùng để đo độ chồng lấn giữa hai bounding boxes, được tính bằng tỉ số giữa phần diện tích chồng lấn (Area of Intersection) vào phần diện tích hợp nhất (Area of Union) của hai bounding boxes. (Hình 2.3 minh họa cho cách tính giá trị này)

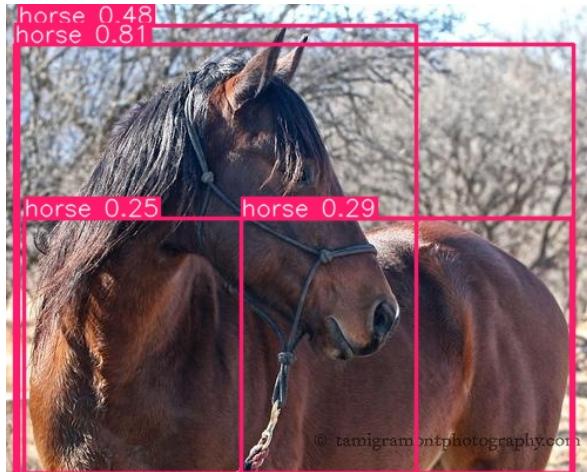
- $\delta > 0$ là ngưỡng giá trị IoU để hai bounding box được xem là chỉ vị trí của cùng một đối tượng.

Tiếp theo, chúng tôi xem xét đến một điều kiện tấn công thành công khó hơn. Một nhiễu ϵ được xem là tấn công thành công khi nó khiến cho kết quả dự đoán của mô hình trên bức ảnh bị thêm nhiễu có ít nhất một đối tượng được dự đoán có vị trí giữ nguyên so với bức ảnh gốc (được xác định bằng giá trị IoU), nhưng lớp của đối tượng đó bị thay đổi (*misclassification*). Nếu mô hình dự đoán nhiều bounding boxes cho cùng một đối tượng (ví dụ như trường hợp xảy ra ở Hình 2.2), thì tấn công thành công chỉ khi tất cả các lớp mà mô hình dự đoán cho các bounding boxes đó phải khác với nhãn đúng của đối tượng. Cụ thể, trong trường hợp này, hàm SUCCESS sẽ được định nghĩa như sau:

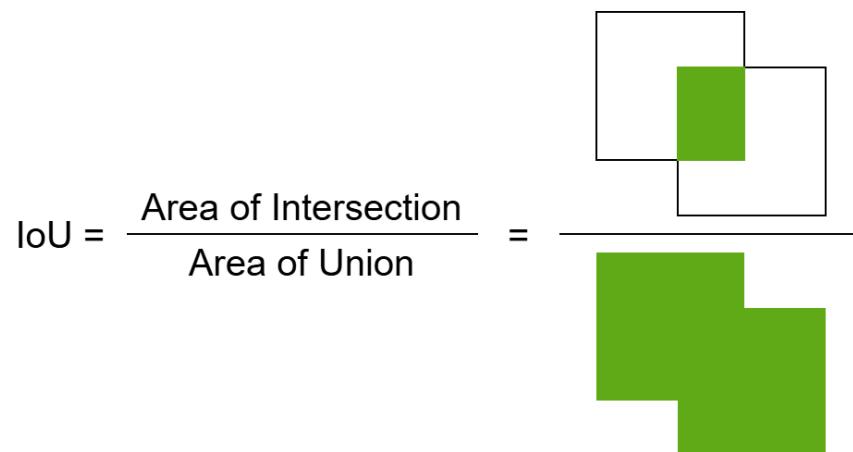
$$\text{SUCCESS}(\mathcal{D}(\mathbf{X}), \mathcal{D}(\mathbf{X} + \epsilon)) = \begin{cases} \text{true} & \text{nếu } \exists(O \in \mathcal{D}(\mathbf{X})) : \\ & (\exists O' \in \mathcal{D}(\mathbf{X} + \epsilon) : \text{IoU}(B_O, B_{O'}) \geq \delta) \wedge \\ & (\forall O' \in \mathcal{D}(\mathbf{X} + \epsilon), \text{IoU}(B_O, B_{O'}) \geq \delta : c_O \neq c_{O'}) \\ \text{false} & \text{các trường hợp còn lại} \end{cases} \quad (2.8)$$

Hình 2.4 là một ví dụ của tấn công thành công khi đối tượng **cat** đã bị dự đoán sai thành đối tượng **bear**, trong hình này, tấn công được xem là thành công đối với cả hai điều kiện 2.7 và 2.8. Trong khi đó, Hình 2.5 là một ví dụ về tấn công thành công với điều kiện 2.7 nhưng không thành công với điều kiện 2.8. Mặc dù khi thêm nhiễu vào bức ảnh sẽ khiến cho mô hình thất bại trong việc phát hiện được đối tượng và thỏa mãn 2.7, nhưng trong ví dụ này lại không có đối tượng nào có nhãn bị thay đổi so với bức ảnh gốc nên không thỏa mãn được điều kiện 2.8.

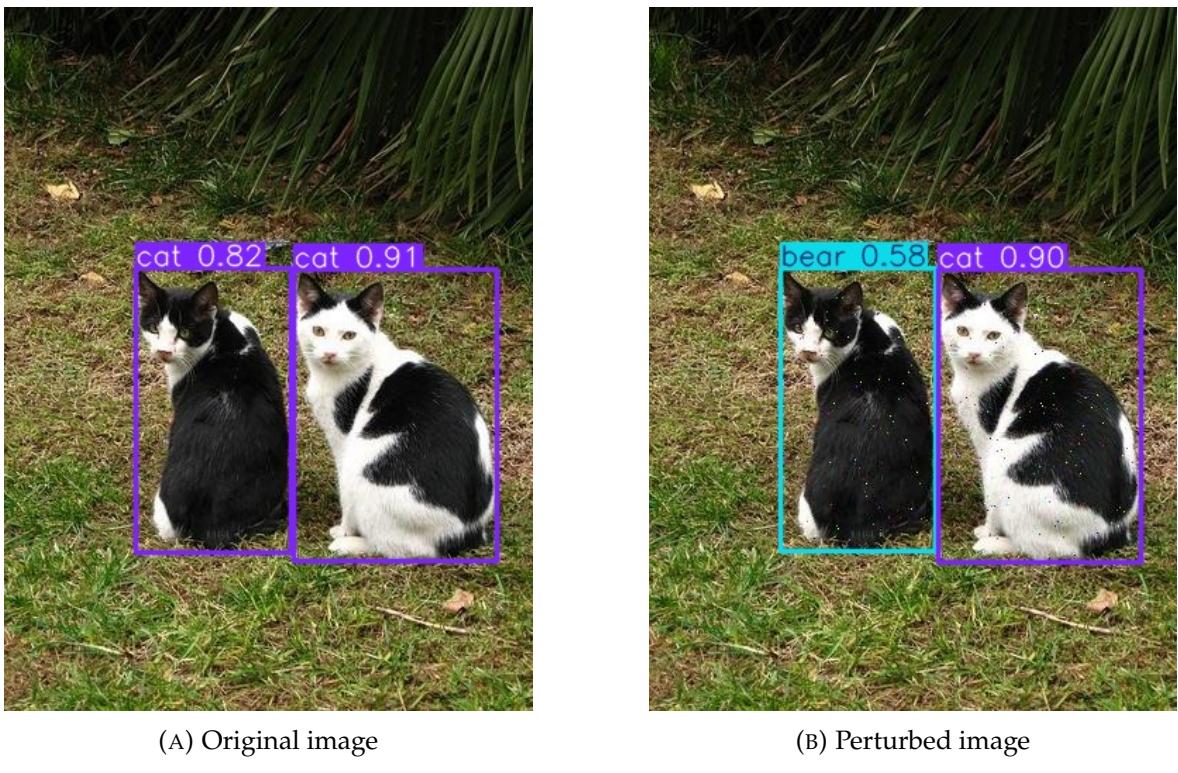
Một tấn công thành công với điều kiện 2.8 thì sẽ luôn thành công với điều kiện 2.7 nhưng điều ngược lại không đúng, vì vậy, chúng tôi sẽ dựa theo điều kiện khó hơn



HÌNH 2.2: YOLOv8 dự đoán nhiều hơn một bounding box cho một đối tượng là con ngựa. (Bức ảnh ví dụ được lấy từ bộ dữ liệu PascalVoc2007 [6])



HÌNH 2.3: Minh họa cho công thức tính IoU.



HÌNH 2.4: Ví dụ về tấn công thành công lên mô hình object detection. Mô hình vẫn phát hiện đúng vị trí các đối tượng so với bức ảnh gốc, nhưng lớp của một đối tượng bị dự đoán sai (**cat** thành **bear**).

là 2.8 để xây dựng hàm măt măt cho tấn công lên các mô hình *object detection* để có thể tìm ra các nhiễu tấn công hiệu quả mô hình theo bất kì điều kiện nào trong hai điều kiện trên (hàm măt sẽ được xây dựng chi tiết ở mục 3.3).

2.2.2 Thuật giải di truyền (Genetic Algorithm - GA):

Thuật giải di truyền [11] là một thuật toán tối ưu hóa và tìm kiếm dựa trên ý tưởng của quá trình chọn lọc và tiến hóa trong thế giới tự nhiên - những cá thể (individuals) tốt, biểu hiện qua độ thích nghi (fitness), sẽ có khả năng sinh tồn cao hơn và truyền lại những đặc tính tốt của mình cho các thế hệ sau.

Thuật giải di truyền có ba thành phần chính, đó là:

- **Quần thể:** gồm nhiều cá thể, là đối tượng mà thuật toán thực hiện quá trình tối ưu hóa hay tìm kiếm trên đó. Mỗi cá thể sẽ được đánh giá dựa trên một giá



HÌNH 2.5: Ví dụ về tấn công **không** thành công lên mô hình object detection.

tri là độ thích nghi (fitness value).

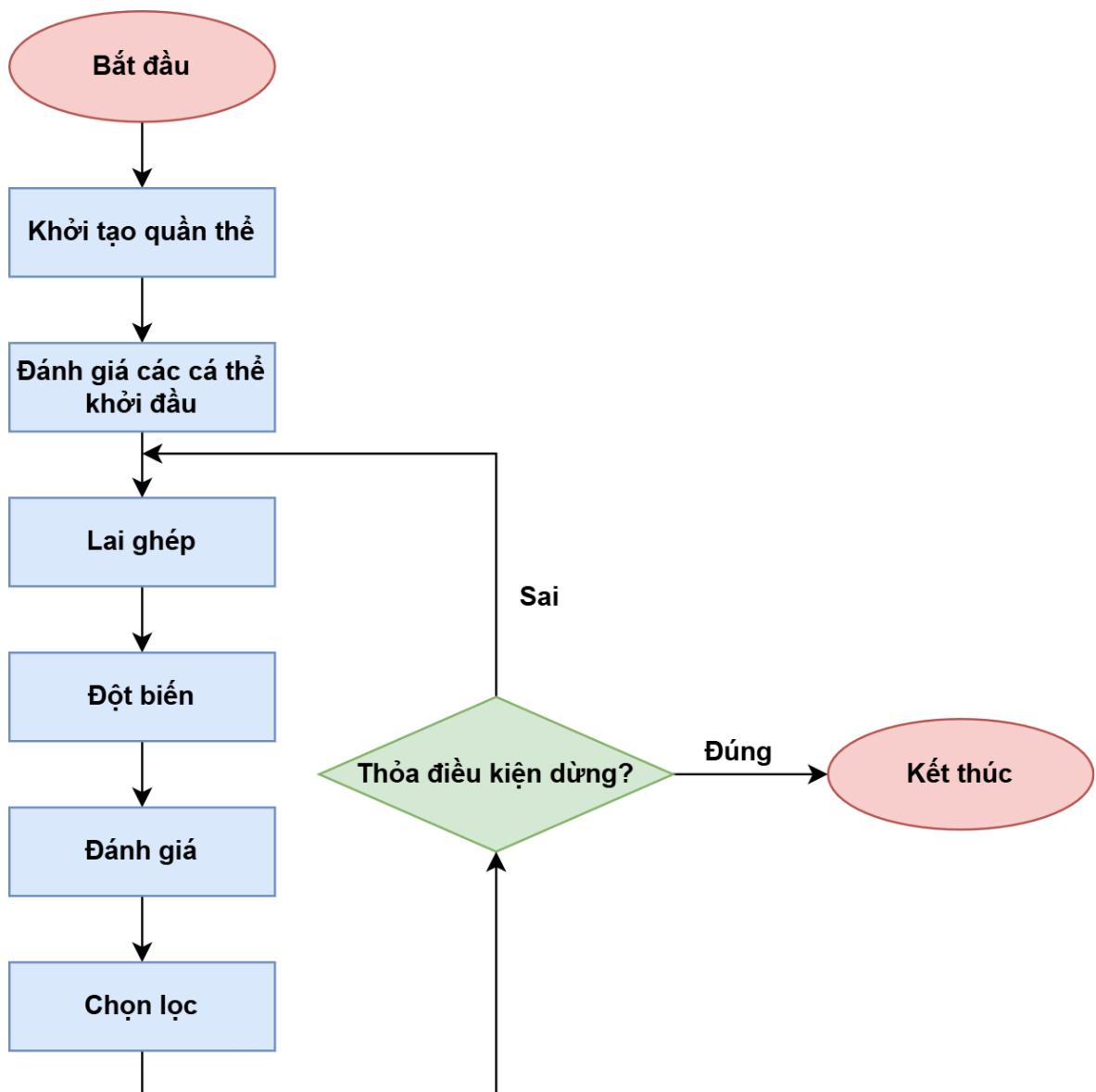
- **Các phép biến đổi:** dùng để tạo hay tìm ra các cá thể mới từ các cá thể đã có. Thuật giải di truyền thường sử dụng hai phép biến đổi là **lai ghép** (crossover) và **đột biến** (mutation). Trong đó, phép lai ghép kết hợp các đặc tính của hai cá thể cha mẹ để tạo ra các cá thể mới, còn phép đột biến sẽ tạo ra một cá thể mới bằng cách thay đổi một hoặc một số đặc điểm của một cá thể đã có.
 - **Phép chọn lọc:** Để lựa chọn và giữ lại các cá thể tốt (có độ thích nghi tốt) cho các thế hệ tiếp theo.

Hình 2.6 mô tả quá trình hoạt động của thuật giải di truyền. Thuật toán bắt đầu bằng việc khởi tạo quần thể ban đầu và đánh giá các cá thể trong quần thể ban đầu tiên này thông qua độ thích nghi. Sau đó tại mỗi thế hệ, hai phép biến đổi là lai ghép và đột biến sẽ lần lượt được thực hiện trên các cá thể đã có (gọi là các cá thể cha mẹ) để tạo ra các cá thể mới. Các cá thể mới này cũng sẽ được tính toán độ thích nghi và sau đó, chúng sẽ được so sánh với nhau, cũng như so sánh với các cá thể cha mẹ để chọn ra những cá thể tốt tiến vào thế hệ sau. Thuật toán sẽ được thực hiện cho đến khi điều kiện dừng được thỏa mãn. Điều kiện dừng ở đây có thể là:

- Sử dụng hết tài nguyên tính toán (như thời gian hay số lần đánh giá tối đa).
 - Cá thể tốt nhất đã thỏa mãn yêu cầu đặt ra.

Trong phạm vi khóa luận này, chúng tôi sử dụng thuật giải di truyền để thiết kế một phương pháp tân công hộp đen thưa lên các mô hình thị giác máy tính vì những lí do sau:

- Đây là một phương pháp tối ưu hóa **hiệu quả trên không gian tìm kiếm rời rạc**, mà mục tiêu bài toán của chúng tôi là tìm kiếm các vị trí và giá trị điểm ảnh bị thay đổi trên bức ảnh gốc, là các giá trị rời rạc.
- Đây là một thuật toán tối ưu hóa **không cần sử dụng gradient**, thích hợp để áp dụng vào tân công hộp đen, nơi mà ta không có được bất kỳ thông tin nào về mô hình nạn nhân ngoại trừ bức ảnh đầu vào và giá trị dự đoán mô hình trả về.
- Đây là một thuật toán có thể dễ cài đặt và sửa đổi, hơn hết, nó có khả năng kết hợp tốt với các kỹ thuật khác để nâng cao hiệu suất.



HÌNH 2.6: Sơ đồ thuật giải di truyền

Chương 3

CÁC PHƯƠNG PHÁP ĐỀ XUẤT

Trong chương này, chúng tôi trình bày chi tiết phương pháp để tìm kiếm các nhiễu thưa (*sparse perturbations*)¹ có thể đánh lừa các mô hình *image classification*² và *object detection*³. Cụ thể, chúng tôi sử dụng giải thuật di truyền (*Genetic Algorithm - GA*)⁴ từng bước tìm kiếm các giá trị tối ưu cho nhiễu bao gồm vị trí các điểm ảnh (*pixels*)⁵ bị thay đổi và giá trị của các nhiễu. Chi tiết của phương pháp được trình bày ở phần 3.2.

Cũng trong chương này, chúng tôi trình bày các hàm mất mát (*Loss functions*) được thiết kế sao cho phù hợp với từng đối tượng bị tấn công là các mô hình *image classification* hay các mô hình *object detection*. Các hàm mất mát này đóng vai trò vô cùng quan trọng làm cho phương pháp của chúng tôi có thể thực hiện tấn công trong điều kiện *black-box* hoàn toàn, khi mà các mô hình chỉ trả về duy nhất điểm tin cậy (*confidence score*) cao nhất trong các lớp. Chi tiết về các hàm mất mát này được trình bày ở phần 3.3

¹Từ này trở về sau của khóa luận, "nhiễu thưa" sẽ được gọi đơn giản là "nhiễu" hay "perturbation(s)"

²Từ này trở về sau của khóa luận, để thuận tiện, mô hình "phân loại hình ảnh" sẽ được thay thế bằng "image classification"

³Từ này trở về sau của khóa luận, để thuận tiện, mô hình "phát hiện đối tượng" sẽ được thay thế bằng "object detection"

⁴Từ này trở về sau của khóa luận, "giải thuật di truyền" sẽ được gọi tắt là "GA"

⁵Từ này trở về sau của khóa luận, "điểm ảnh" sẽ được thay thế bằng "pixel(s)"

3.1 Không gian giá trị của nhiễu

Như đã đề cập ở phần trước, nhiệm vụ của chúng ta là đi tìm một nhiễu ϵ có cùng kích thước với đầu vào. Đối với cách tiếp cận tấn công thưa, ϵ cần được thiết kế sao cho khi thêm vào bức ảnh gốc thì chỉ làm nó thay đổi ở một số lượng nhỏ điểm ảnh. Đồng thời, không gian tìm kiếm lớn cũng là một vấn đề khó khăn. Vì vậy, để giới hạn không gian tìm kiếm, chúng tôi sẽ giới hạn phạm vi giá trị của ϵ bởi một tham số σ để mỗi phần tử chỉ nhận một trong ba giá trị $\{-\sigma, 0, \sigma\}$:

$$\epsilon \in \{-\sigma, 0, \sigma\}^{h \times w \times 3}$$

với $\sigma > 0$ dùng để điều chỉnh mức độ bị thay đổi của các điểm ảnh, giá trị σ càng lớn thì màu sắc mà các điểm ảnh bị thay đổi sẽ càng rõ.

Ngoài ra, chúng tôi cũng sẽ đặt thêm giới hạn cho ϵ :

$$\|\epsilon\|_0 < \theta$$

với tham số $\theta > 0$ và như đã đề cập, chúng tôi đồng nhất việc tính giá trị chuẩn l_0 cho ϵ là tính số lượng pixel bị thay đổi:

$$\|\epsilon\|_0 = \sum_{i \in \mathbb{Z}_{[0,h)} \times \mathbb{Z}_{[0,w)}} (\epsilon_i \neq (0,0,0))$$

3.2 Sử dụng GA để tìm kiếm nhiễu

Thuật giải 1 mô tả cách chúng tôi sử dụng GA để tìm kiếm nhiễu. Chi tiết về từng thành phần trong thuật toán sẽ được chúng tôi trình bày trong những mục tiếp theo.

3.2.1 Quần thể và cá thể

Trong giải thuật di truyền, mỗi lời giải - mà trong bài toán này là các nhiễu ϵ - được gọi là một cá thể⁶(Individual), và tập hợp nhiều cá thể I như vậy sẽ tạo thành một

⁶Các cụm từ "cá thể" (*individual*), "giải pháp" (*solution*) đều chỉ cá thể trong quần thể

Thuật giải 1 GA

Đầu vào: Tập các bounding boxes \mathcal{B} (chỉ sử dụng cho tấn công lên *object detection*) và lớp đúng (*ground truth*) tương ứng \mathbf{c} ; kích thước quần thể s ; tỉ lệ pixels được hoán đổi ở lai ghép p_c và đột biến p_m ; xác suất tạo ra giá trị 0 p_0 ; tỉ lệ pixel bị thay đổi tối đa trên toàn bộ bức ảnh (đối với tấn công *image classification*) hay tỉ lệ pixel bị thay đổi tối đa cho mỗi đối tượng (đối với tấn công lên *object detection*) k ; số lượng thế hệ tối đa max_gen ; kích thước tối đa cho archive $max_archive$; kích thước giải đầu T .

Đầu ra: Nhiều tốt nhất trong quần thể.

```

1:  $P \leftarrow \text{Init}(\mathcal{B}, p_0, k)$       # Khởi tạo quần thể
2:  $archive \leftarrow \text{Copy}(P)$ 
3: for  $I \in P$  do
4:   Evaluation( $I, \mathcal{B}, \mathbf{c}$ )      # Đánh giá các cá thể
5:    $g \leftarrow 0$ 
6:   while  $g < max\_gen$  do
7:      $Offspring \leftarrow \text{Crossover}(P, p_c)$       # Lai ghép, tùy vào đối tượng tấn công sẽ là
                                                 Thuật giải 2 hoặc 3
8:     for  $o \in Offspring$  do
9:        $o \leftarrow \text{Mutation}(o, archive, p_m, p_0)$       # Đột biến, tùy vào đối tượng tấn
                                                 công sẽ là Thuật giải 4 hoặc 5
10:      Evaluation( $o, \mathcal{B}, \mathbf{c}$ )
11:       $PO \leftarrow P \cup Offspring$ 
12:       $P \leftarrow \text{TournamentSelection}(PO, T, s)$       # Chọn lọc giải đầu
13:       $archive \leftarrow \text{UpdateArchive}(Offspring)$       # Cập nhật archive với các cá thể từ
                                                 Offspring
14:       $g \leftarrow g + 1$ 
15:  $result \leftarrow \text{FindBest}(P)$       # Chọn ra cá thể tốt nhất trong quần thể
16: return  $result$ 

```

quần thể P (**Population**). Các quần thể sẽ bị thay đổi qua từng thế hệ nhưng sẽ luôn có số lượng cá thể là s .

Các cá thể sẽ được biểu diễn theo một cách thích hợp để:

- Mỗi cá thể biểu diễn được đầy đủ thông tin của một nhiễu mà nó đại diện cho.
- Cách biểu diễn phải thuận tiện cho các phép biến đổi lai ghép và đột biến

Vì vậy, tùy theo loại mô hình bị tấn công, chúng tôi sẽ có những cách biểu diễn khác nhau cho các cá thể. Định nghĩa cụ thể của cá thể cho từng bài toán sẽ được đề cập ở tiếp theo đây.

Cá thể trong tấn công mô hình Image Classification

Khi tấn công mô hình *image classification*, mục tiêu của ta là tạo ra nhiều thêm vào bức ảnh đầu vào để khiến cho mô hình dự đoán sai lầm của bức ảnh đó. Để làm được điều này, chúng tôi định nghĩa một cá thể trong quần thể sẽ bao gồm những thông tin sau:

- Tập các vị trí của các pixels bị thay đổi, kí hiệu là Pos . Mỗi giá trị $u \in Pos$ là một số nguyên trong đoạn $[0, h \times w - 1]$ với h và w lần lượt là chiều cao và chiều rộng của bức ảnh đầu vào và $i \times w + j$ đại diện cho (i, j) (hàng i , cột j) trên ảnh. Nếu pixel ở (i, j) trên ảnh bị thay đổi thì giá trị $i \times w + j$ sẽ nằm trong tập Pos .

Ví dụ: Bức ảnh đầu vào có kích thước là 32×32 , quần thể chứa một cá thể có đúng hai pixels bị thay đổi ở vị trí lần lượt là hàng 1, cột 2 và hàng 30, cột 31. Khi đó tập Pos của cá thể này là $Pos = \{1 \times 32 + 2, 30 \times 32 + 31\}$ hay $Pos = \{34, 991\}$.

- Tập các giá trị nhiễu tương ứng với mỗi vị trí pixel bị thay đổi, kí hiệu là Δ , với ràng buộc là $|Pos| = |\Delta|$. Mỗi $v \in \Delta$ là một bộ ba giá trị (v_R, v_G, v_B) tương ứng với ba kênh màu trong bức ảnh đầu vào. Mỗi giá trị v_R, v_G, v_B nhận một trong ba giá trị thuộc tập $\{-\sigma, 0, \sigma\}$. Cuối cùng, ta đánh chỉ số, mỗi $v_i \in \Delta$ sẽ tương ứng với mỗi $u_i \in Pos$ theo thứ tự.

Ví dụ: với cá thể có đúng hai pixel bị thay đổi thì Δ có thể như sau $\Delta = \{(0, 0, 0), (\sigma, \sigma, \sigma)\}$ hoặc $\Delta = \{(0, \sigma, -\sigma), (\sigma, 0, 0)\}$. Nếu $Pos = \{34, 991\}$ (kích

thước ảnh là 32×32) và $\Delta = \{(0, \sigma, -\sigma), (\sigma, 0, 0)\}$ thì thay đổi tại pixel $(1, 2)$ sẽ là $(0, \sigma, -\sigma)$ và tại $(30, 31)$ là $(\sigma, 0, 0)$

Cá thể trong tấn công mô hình Object Detection

Mục tiêu khi tấn công các mô hình *object detection* là khiến cho mô hình dự đoán sai lốp của các đối tượng được phát hiện. Vì vậy, tấn công mô hình *object detection* có thể được đưa về nhiều bài toán tương tự tấn công lên mô hình *image classification* cho mỗi đối tượng trong bức ảnh đầu vào. Từ đây, chúng tôi định nghĩa mỗi cá thể trong quần thể bao gồm các thành phần sau:

- Danh sách các vị trí ứng với mỗi đối tượng trong bức ảnh bị tấn công

$$\mathcal{P} = \{Pos_{O_1}, Pos_{O_2}, \dots, Pos_{O_n}\}.$$

- Danh sách các giá trị của nhiễu tương ứng $\Theta = \{\Delta_{O_1}, \Delta_{O_2}, \dots, \Delta_{O_n}\}$

Chi tiết hơn, mỗi Pos_{O_i} và Δ_{O_i} là tập các vị trí pixels bị thay đổi và tập các giá trị nhiễu tương ứng với đối tượng O_i và sẽ được định nghĩa tương tự Pos và Δ trong tấn công mô hình *image classification*. Cụ thể,

- $|Pos_{O_i}| = |\Delta_{O_i}|, \forall O_i$.
- Pos_{O_i} là tập các vị trí của các pixels bị thay đổi ở đối tượng O_i . Mỗi vị trí $u \in Pos_{O_i}$ là một số nguyên trong đoạn $[0, h_{O_i} \times w_{O_i} - 1]$ với h_{O_i} và w_{O_i} lần lượt là chiều cao và chiều rộng của bounding box chứa đối tượng O_i .
- Δ_{O_i} là tập các giá trị nhiễu tương ứng với mỗi vị trí pixel bị thay đổi ở đối tượng O_i . Mỗi $v \in \Delta_{O_i}$ là một bộ ba giá trị (v_R, v_G, v_B) tương ứng với ba kênh màu. Mỗi giá trị v_R, v_G, v_B nhận một trong ba giá trị thuộc tập $\{-\sigma, 0, \sigma\}$. Cuối cùng, ta cũng đánh chỉ số để $v_j \in \Delta_{O_i}$ sẽ tương ứng là nhiễu tại vị trí $u_j \in Pos_{O_i}$ theo đúng thứ tự.

3.2.2 Khởi tạo quần thể

Bước đầu tiên của quá trình tìm kiếm nhiễu sử dụng GA là khởi tạo quần thể. Tùy vào đối tượng bị tấn công là mô hình *image classification* hay mô hình *object detection*, quần thể sẽ được khởi tạo theo các cách khác nhau.

- **Mô hình bị tấn công là *image classification*:**

Ta định nghĩa trước số nguyên dương s là kích thước của quần thể và số thực dương $k < 1$ là tỉ lệ số lượng pixels bị thay đổi tối đa. Sau đó, s cá thể trong quần thể sẽ được khởi tạo. Với mỗi cá thể I trong quần thể, tập vị trí Pos được khởi tạo bằng cách lấy $H \cdot W \cdot k$ mẫu phân biệt từ tập $\{0, 1, \dots, H \times W - 1\}$ với xác suất lấy mẫu đồng đều; trong khi, tập các giá trị nhiễu Δ được khởi tạo như sau: Với mỗi $v \in \Delta$, mỗi giá trị tương ứng cho từng kênh màu v_R, v_G, v_B sẽ lấy một giá trị ngẫu nhiên từ tập $\{-\sigma, 0, \sigma\}$ với xác suất nhận giá trị 0 là p_0 và xác suất nhận hai giá trị còn lại là bằng nhau.

- **Mô hình bị tấn công là *object detection*:**

Trước tiên, ta cũng định nghĩa trước số nguyên dương s là kích thước quần thể, tuy nhiên, khác với trường hợp *image classification*, ta định nghĩa $k < 1$ là tỉ lệ số lượng pixels bị thay đổi tối đa **cho từng đối tượng trong bức ảnh đầu vào**. Sau đó, s cá thể trong quần thể sẽ được khởi tạo, với mỗi cá thể I được khởi tạo như sau:

- Mỗi tập Pos_{O_i} được khởi tạo bằng cách lấy $h_{O_i} \cdot w_{O_i} \cdot k$ mẫu phân biệt từ tập hợp $\{0, 1, \dots, h_{O_i} \times w_{O_i} - 1\}$ với xác suất lấy mẫu đồng đều.
- Đối với các tập Δ_{O_i} , với mỗi $v \in \Delta_{O_i}$ mỗi giá trị tương ứng cho ba kênh màu là v_R, v_G, v_B cũng sẽ nhận một giá trị ngẫu nhiên từ tập $\{-\sigma, 0, \sigma\}$ với xác suất tương tự như ở trường hợp mô hình bị tấn công là *image classification*.

3.2.3 Lai ghép

Phép biến đổi đầu tiên mà chúng tôi sử dụng trong thuật toán GA là lai ghép - **crossover** (còn có thể gọi là tái tổ hợp - **recombination**). Đây là phép biến đổi kết hợp thông tin từ hai cá thể đã có từ thế hệ trước (được xem là hai cá thể cha mẹ) để tạo ra các cá thể mới. Nhờ việc kết hợp này, phép biến đổi cho phép giữ lại những đặc tính tốt của các cá thể cha mẹ và truyền lại cho cá thể con ở thế hệ kế tiếp, nhờ đó mà tăng khả năng tạo ra các cá thể tốt hơn. Trong ngữ cảnh bài toán của chúng tôi, các đặc tính tốt được giữ và truyền lại sẽ là các vị trí pixels quan trọng và các giá

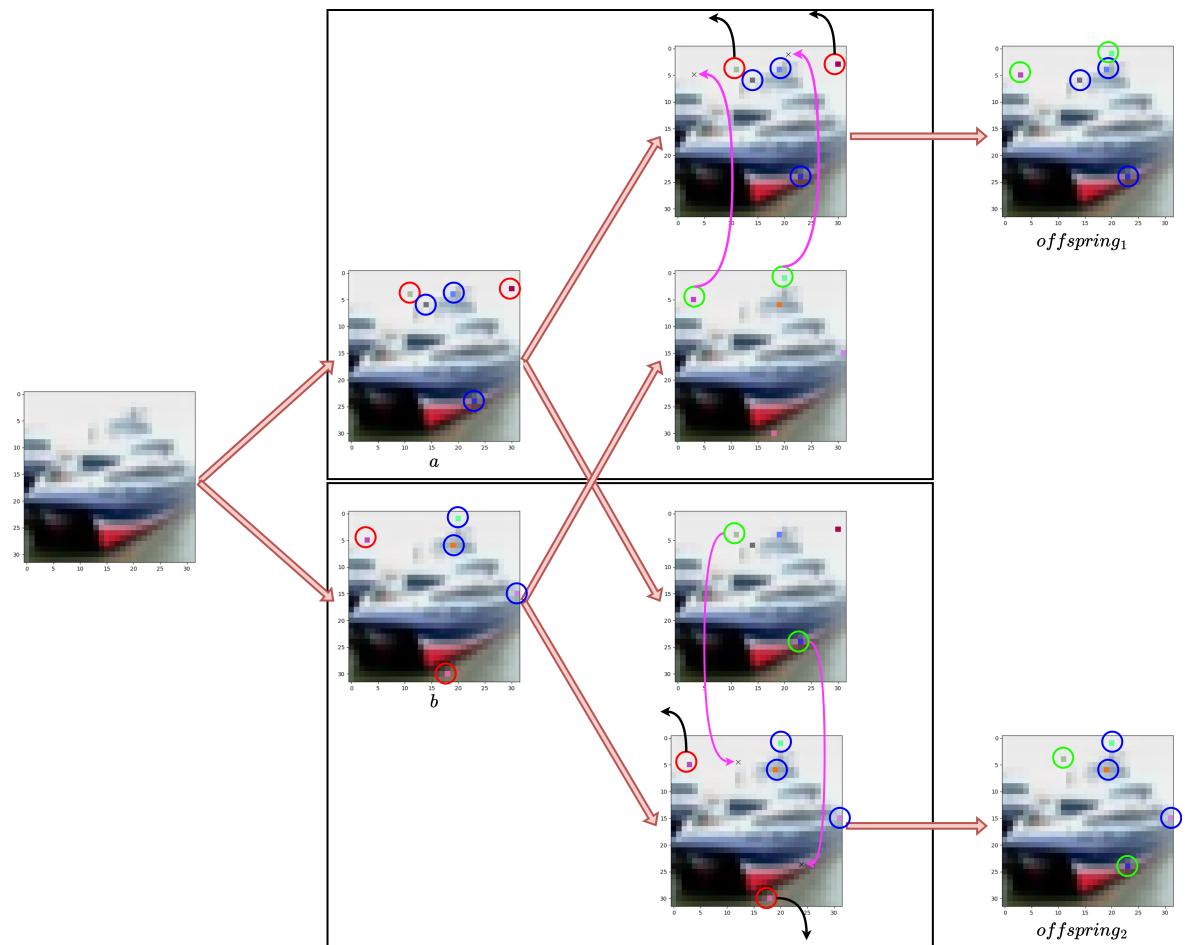
trị nhiễu tương ứng, để khi thay đổi pixels các ở vị trí này với những giá trị nhiễu như vậy trên bức ảnh đầu vào thì sẽ có khả năng cao làm cho mô hình đưa ra dự đoán không chính xác. Xét về phương diện một bài toán tối ưu hóa, mục tiêu của phép lai ghép định hướng quá trình tìm kiếm về phía những vùng tiềm năng trong không gian tìm kiếm, nơi mà những giải pháp tốt hơn, hay trong bài toán của chúng tôi là những nhiễu có khả năng đánh lừa mô hình, có thể được tìm thấy.

Đối với tấn công lên các mô hình *image classification*, phép lai ghép của chúng tôi được lấy ý tưởng từ nghiên cứu [22], quá trình lai ghép được mô tả trong Thuật giải 2. Đầu tiên, quần thể ở thế hệ trước được chia thành các nhóm 2 cá thể một cách ngẫu nhiên, sau đó, phép lai ghép sẽ được thực hiện với từng cặp cá thể. Với mỗi cặp cá thể bất kì, ta gọi từng cá thể lần lượt là a và b . Ta tạo tập hợp S gồm các vị trí pixels bị thay đổi có trong Pos_b nhưng không có trong Pos_a (dòng 3 trong Thuật giải 2) và định nghĩa số lượng pixels được thay thế là $\tau = \min\{p_c \cdot (h \cdot w \cdot k), |S|\}$ (dòng 4 trong Thuật giải 2), với p_c là tỉ lệ pixels được thay thế ở phép lai ghép. Sau đó, ta chọn ngẫu nhiên τ phần tử từ tập vị trí Pos_a (gán vào biến A) và ngẫu nhiên τ vị trí từ tập S (gán vào biến B) rồi thay thế hoàn toàn các vị trí A trong Pos_a thành B cùng với các giá trị nhiễu tương ứng của chúng (các dòng từ 5 đến 8 trong Thuật giải 2) để tạo nên một cá thể mới. Quá trình trên cũng được thực hiện tương tự theo hướng ngược lại (vòng **for** trong Thuật giải 2) để từ **một cặp** cá thể cha mẹ ban đầu, ta tạo ra **hai** cá thể con mới. Từ đây, ta thấy rằng tập các cá thể con được tạo ra (*Offsprings*) có kích thước đúng bằng với tập cha mẹ và bằng s là kích thước quần thể. Hình 3.1 minh họa rõ hơn cho quá trình lai ghép giữa hai cá thể a và b trong quần thể.

Đối với tấn công lên các mô hình *object detection*, sau khi chia quần thể cha mẹ thành các cặp cá thể, với mỗi cặp cá thể, Thuật giải 2 sẽ được áp dụng cho mỗi đối tượng O_i trong bức ảnh đầu vào (Xem Thuật giải 3).

3.2.4 Đột biến

Phép biến đổi thứ hai được chúng tôi sử dụng là đột biến - **mutation**. Phép đột biến giúp tạo ra **một** cá thể mới bằng cách thêm vào những thay đổi ngẫu nhiên rất nhỏ ở **một** cá thể đã có, cụ thể trong bài toán của chúng tôi, phép đột biến sẽ thay đổi một số ít vị trí pixels hay giá trị nhiễu của một cá thể. Dưới góc nhìn của bài toán



HÌNH 3.1: Minh họa quá trình lai ghép cho tấn công các mô hình image classification; $offspring_1$ và $offspring_2$ là các cá thể con được tạo ra sau quá trình lai ghép. Các vị trí nhiều được khoanh tròn màu **đỏ** là những vị trí bị thay thế, các vị trí nhiều được khoanh màu **xanh lá** là những vị trí được chọn để thay thế các vị trí khoanh **đỏ**, các vị trí nhiều được khoanh màu **xanh lam** là những vị trí không bị tác động trong quá trình lai ghép.

Thuật giải 2 Crossover1 (lai ghép trên một đối tượng)

Đầu vào: Vị trí các pixels bị thay đổi lần lượt là Pos_a, Pos_b và giá trị các nhiễu tương ứng lần lượt là Δ_a, Δ_b ; tỉ lệ pixels được thay thế p_c ; tỉ lệ pixel bị thay đổi tối đa k ; chiều cao và chiều rộng của bức ảnh (cho trường hợp tấn công lên *image classification*) hoặc của bounding box chứa đối tượng (cho trường hợp tấn công lên *object detection*) lần lượt là h và w .

```

1:  $\mathcal{O} = \{\}$ 
2: for  $(r, e) \in \{(a, b), (b, a)\}$  do
3:    $\mathcal{S} \leftarrow Pos_e \setminus (Pos_r \cap Pos_e)$ 
4:    $\tau \leftarrow \min\{p_c \cdot (h \cdot w \cdot k), |\mathcal{S}|\}$ 
    //  $\mathcal{U}(X)^y$  là phép chọn ngẫu nhiên các phần tử phân biệt từ tập  $X$  với xác suất
    // chọn bằng nhau, kết quả của phép chọn là một tập hợp có kích thước là  $y$ .
5:    $A \leftarrow \mathcal{U}(Pos_r)^\tau$ 
6:    $B \leftarrow \mathcal{U}(\mathcal{S})^\tau$ 
7:    $Pos'_r \leftarrow (Pos_r \setminus A) \cup B$ 
8:    $\Delta'_r \leftarrow (\Delta_r \setminus \Delta_{r_A}) \cup \Delta_{e_B}$ 
9:    $\mathcal{O} \leftarrow \mathcal{O} \cup \{\{Pos'_r, \Delta'_r\}\}$ 
10: return  $\mathcal{O}$ 

```

Thuật giải 3 CrossoverObjectDetection (lai ghép cho tấn công object detection)

Đầu vào: Hai cá thể a, b ; tập các bounding box cho các đối tượng trong bức ảnh đầu vào \mathcal{B} ; tỉ lệ pixels được thay thế p_c ; tỉ lệ pixel bị thay đổi tối đa cho mỗi đối tượng k .

```

1:  $\mathcal{P}_{\mathcal{X}} \leftarrow \{\}, \Theta_{\mathcal{X}} \leftarrow \{\}$ 
2:  $\mathcal{P}_{\mathcal{Y}} \leftarrow \{\}, \Theta_{\mathcal{Y}} \leftarrow \{\}$ 
3: for  $i \in [0, |\mathcal{B}|]$  do
    //  $Pos_{x;O_i}$  và  $\Delta_{x;O_i}$  lần lượt là tập vị trí pixels bị thay đổi và tập các giá trị nhiễu
    // của cá thể  $x$  ứng với đối tượng  $O_i$  trong bức ảnh đầu vào;  $h_{O_i}, w_{O_i}$  lần lượt là
    // chiều cao và chiều rộng của bounding box của đối tượng  $O_i$ , có thể được suy ra
    // từ tập  $\mathcal{B}$ .
4:    $\mathcal{X}_{O_i}, \mathcal{Y}_{O_i} \leftarrow \text{Crossover1}(Pos_{a;O_i}, Pos_{b;O_i}, \Delta_{a;O_i}, \Delta_{b;O_i}, p_c, k, h_{O_i}, w_{O_i})$ 
5:    $\mathcal{P}_{\mathcal{X}} \leftarrow \mathcal{P}_{\mathcal{X}} \cup (Pos \in \mathcal{X}_{O_i}), \Theta_{\mathcal{X}} \leftarrow \Theta_{\mathcal{X}} \cup (\Delta \in \mathcal{X}_{O_i})$ 
6:    $\mathcal{P}_{\mathcal{Y}} \leftarrow \mathcal{P}_{\mathcal{Y}} \cup (Pos \in \mathcal{Y}_{O_i}), \Theta_{\mathcal{Y}} \leftarrow \Theta_{\mathcal{Y}} \cup (\Delta \in \mathcal{Y}_{O_i})$ 
7:  $\mathcal{O}_{\mathcal{X}} \leftarrow \{\mathcal{P}_{\mathcal{X}}, \Theta_{\mathcal{X}}\}$ 
8:  $\mathcal{O}_{\mathcal{Y}} \leftarrow \{\mathcal{P}_{\mathcal{Y}}, \Theta_{\mathcal{Y}}\}$ 
9: return  $\{\mathcal{O}_{\mathcal{X}}, \mathcal{O}_{\mathcal{Y}}\}$ 

```

tối ưu hóa, phép biến đổi này hướng tới việc khám phá những vùng chưa biết trong không gian tìm kiếm, giúp tăng khả năng tìm được các giải pháp mới có tiềm năng, đồng thời ngăn chặn hiện tượng hội tụ sớm của quá trình tìm kiếm. Cũng như lai ghép, việc thực hiện phép đột biến sẽ phụ thuộc vào mô hình bị tấn công là *image classification* hay *object detection*.

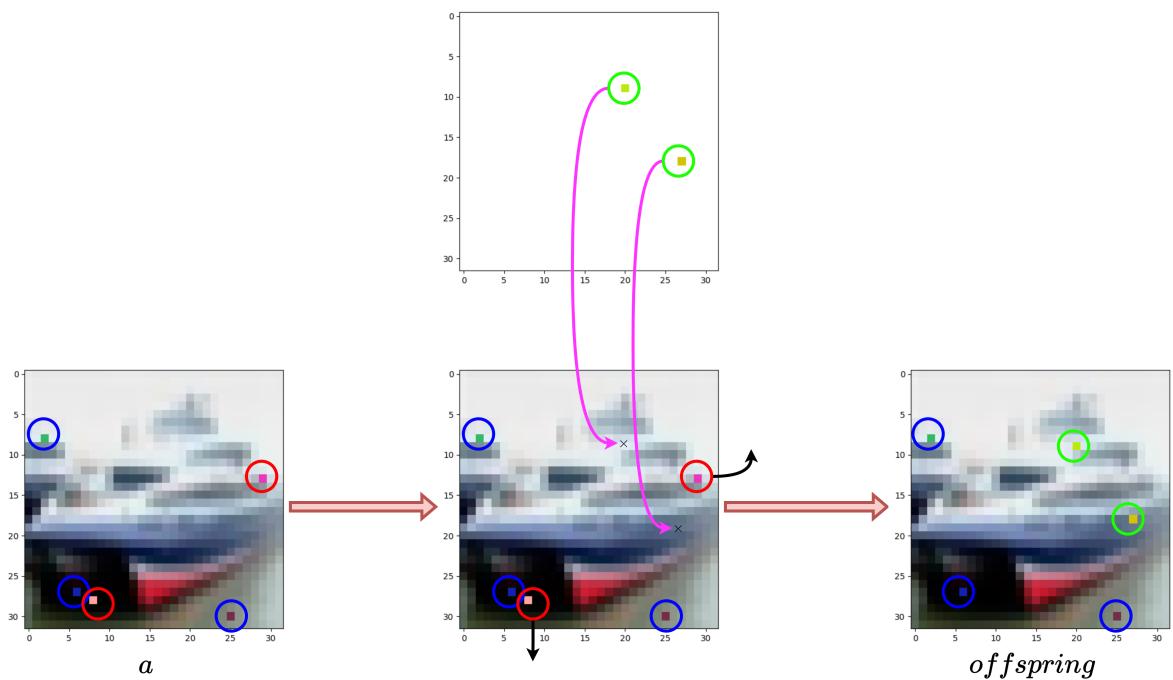
Với mô hình bị tấn công là *image classification*, phép đột biến diễn ra trên mỗi cá thể a trong quần thể, được điều chỉnh từ phép đột biến trong nghiên cứu [22], như sau. Trước tiên, ta tạo tập V gồm những vị trí pixels trên bức ảnh đầu vào không bị a thay đổi, hay nói cách khác là không thuộc Pos_a (dòng 1, 2 trong Thuật giải 4). Sau đó, ta định nghĩa số lượng pixels được thay thế là $\tau = \min\{p_m \cdot (h \cdot w \cdot k), 1\}$, với p_m là tỉ lệ pixels được thay thế ở phép đột biến. Ta chọn ngẫu nhiên τ vị trí từ Pos_a và loại bỏ chúng cùng với các giá trị nhiễu tương ứng khỏi Pos_a và Δ_a . Đồng thời, thay thế bằng τ vị trí ngẫu nhiên mới được chọn từ V , với các vị trí mới này, ta tạo ra các giá trị nhiễu tương ứng bằng cách chọn ngẫu nhiên các giá trị từ tập $\{-\sigma, 0, \sigma\}$ với xác suất chọn giá trị 0 là p_0 (các dòng từ 4 đến 7 trong Thuật giải 4). Hình 3.2 minh họa rõ hơn cho quá trình đột biến một cá thể a trong quần thể.

Khi chuyển qua tấn công lên các mô hình *object detection*, Thuật giải 4 vẫn sẽ được sử dụng, tuy nhiên, phạm vi áp dụng của nó sẽ trở thành trên từng đối tượng của bức ảnh đầu vào thay vì toàn bộ bức ảnh. Thuật giải 5 thể hiện sự thay đổi này.

3.2.5 Đánh giá

Để đánh giá độ tốt của một cá thể trong quần thể, trước tiên ta cần tạo ra nhiều từ cá thể đó để áp dụng lên bức ảnh đầu vào. Nhiều được tạo ra bằng cách cộng các giá trị nhiễu (từ tập Δ) lên các vị trí pixels tương ứng (từ tập Pos) trên bức ảnh đầu vào. Sau đó, nếu có bất kỳ giá trị của pixels nào sau khi được cộng nhiễu vượt ra khỏi ngưỡng giá trị hợp lệ thì các giá trị vượt ngưỡng đó sẽ được gán lại bằng giá trị hợp lệ gần nhất với nó. Lấy ví dụ, nếu ngưỡng giá trị hợp lệ của các pixels là từ 0 đến 255, nhưng sau khi được cộng nhiễu thì một pixel trong bức ảnh đầu vào có giá trị là $(-1, 100, 270)$, khi đó giá trị của pixel này sẽ được gán lại thành $(0, 100, 255)$.

Sau khi đã có được bức ảnh bị làm nhiễu, bước tiếp theo ta sẽ đưa bức ảnh này vào mô hình bị tấn công, kết quả dự đoán của mô hình trên bức ảnh này sẽ được đánh giá bằng một **hàm mất mát**, ký hiệu là \mathcal{L} (chi tiết về hàm mất mát được trình



HÌNH 3.2: Minh họa quá trình đột biến một cá thể cho tấn công các mô hình image classification, *offspring* là cá thể mới được tạo ra sau đột biến. Các vị trí nhiễu được khoanh tròn màu **đỏ** là những vị trí bị thay thế, các vị trí nhiễu được khoanh màu **xanh lá** là những vị trí mới thay thế cho những vị trí khoanh **đỏ**, các vị trí nhiễu được khoanh màu **xanh lam** là những vị trí không bị tác động trong quá trình đột biến.

Thuật giải 4 Mutation1 (đột biến trên một đối tượng)

Đầu vào: Vị trí các pixels bị thay đổi Pos_a ; giá trị các nhiễu tương ứng Δ_a , tỉ lệ pixels được thay thế p_m ; tỉ lệ pixel bị thay đổi tối đa k ; xác suất tạo ra giá trị 0 là p_0 ; chiều cao và chiều rộng của bức ảnh (cho trường hợp tấn công lên *image classification*) hoặc của bounding box chứa đối tượng (cho trường hợp tấn công lên *object detection*) lần lượt là h và w .

- 1: $\mathcal{S} \leftarrow \{0, 1, \dots, h \cdot w - 1\}$
 - 2: $V \leftarrow \mathcal{S} \setminus Pos_a$
 - 3: $\tau \leftarrow \min\{p_m \cdot (h \cdot w \cdot k), 1\}$
 - 4: $A \leftarrow \mathcal{U}(Pos_a)^\tau$
 - 5: $B \leftarrow \mathcal{U}(V)^\tau$
 - 6: $Pos'_a \leftarrow (Pos_a \setminus A) \cup B$
// $\mathcal{N}(\{-\sigma, 0, \sigma\})^y$ là phép chọn ngẫu nhiên các giá trị từ tập $\{-\sigma, 0, \sigma\}$ với xác suất chọn giá trị 0 là p_0 . Kết quả của phép chọn này là một tập với kích thước y .
 - 7: $\Delta'_a \leftarrow (\Delta_a \setminus \Delta_{a_A}) \cup \mathcal{N}(\{-\sigma, 0, \sigma\})^{\tau \times 3}$
 - 8: **return** $\{Pos'_a, \Delta'_a\}$
-

Thuật giải 5 MutationObjectDetection (đột biến cho tấn công object detection)

Đầu vào: Một cá thể a ; tập các bounding box cho các đối tượng trong bức ảnh đầu vào \mathcal{B} ; tỉ lệ pixels được thay thế p_m ; tỉ lệ pixel bị thay đổi tối đa cho mỗi đối tượng k .

- 1: $\mathcal{P}_{\mathcal{Z}} \leftarrow \{\}, \Theta_{\mathcal{Z}} \leftarrow \{\}$
 - 2: **for** $i \in [0, |\mathcal{B}|]$ **do**
 - 3: $\mathcal{Z} \leftarrow \text{Mutation1}(Pos_{a;O_i}, \Delta_{a;O_i}, p_m, k, h_{O_i}, w_{O_i})$
 - 4: $\mathcal{P}_{\mathcal{Z}} \leftarrow \mathcal{P}_{\mathcal{Z}} \cup (Pos \in \mathcal{Z}_{O_i}), \Theta_{\mathcal{Z}} \leftarrow \Theta_{\mathcal{Z}} \cup (\Delta \in \mathcal{Z}_{O_i})$
 - 5: $\mathcal{O}_{\mathcal{Z}} \leftarrow \{\mathcal{P}_{\mathcal{Z}}, \Theta_{\mathcal{Z}}\}$
 - 6: **return** $\mathcal{O}_{\mathcal{Z}}$
-

bày ở 3.3). Giá trị của hàm măt măt trên kết quả dự đoán của mô hình sẽ được dùng để đánh giá độ tốt của một cá thể trong quần thể. Tuy nhiên, rõ ràng là nếu có hai hoặc nhiều nhiễu đều khiến cho mô hình đưa ra dự đoán không chính xác thì ta nên ưu tiên chọn nhiễu nó nhiều giá trị bằng $(0, 0, 0)$ hơn như đã được đề cập ở phần 3.1, nói cách khác nếu có nhiều ảnh cùng khiến mô hình đưa ra dự đoán sai thì ảnh tốt hơn là ảnh khác với ảnh gốc ở pixel hơn.

Vì vậy, trước hết chúng tôi sẽ ưu tiên việc tạo ra nhiều tần công thành công với số pixel thay đổi không vượt quá ngưỡng cho phép bằng cách tối ưu hàm măt măt \mathcal{L} . Sau đó, chúng tôi sẽ cố gắng giảm số pixel cần thay đổi nhưng vẫn giữ được tính thành công của mô hình. Do đó, sự vượt trội của một cá thể so với cá thể khác sẽ được định nghĩa như sau:

Định nghĩa 1 (Cá thể tốt hơn). Cho hai cá thể A và B tạo nên hai nhiễu lần lượt là ϵ_A và ϵ_B . A được xem là tốt hơn B nếu thỏa mãn một trong những điều kiện sau đây:

- ϵ_A là một nhiễu thành công (khiến mô hình đưa ra dự đoán sai lệch) nhưng ϵ_B thì không:

$$\begin{cases} \text{SUCCESS}(\mathcal{M}(\mathbf{X}), \mathcal{M}(\mathbf{X} + \epsilon_A)) = \text{true} \\ \text{SUCCESS}(\mathcal{M}(\mathbf{X}), \mathcal{M}(\mathbf{X} + \epsilon_B)) = \text{false} \end{cases}$$

- Cả ϵ_A và ϵ_B đều là nhiễu **không thành công**, và giá trị hàm măt măt trên ϵ_A nhỏ hơn trên ϵ_B :

$$\begin{cases} \text{SUCCESS}(\mathcal{M}(\mathbf{X}), \mathcal{M}(\mathbf{X} + \epsilon_A)) = \text{false} \\ \text{SUCCESS}(\mathcal{M}(\mathbf{X}), \mathcal{M}(\mathbf{X} + \epsilon_B)) = \text{false} \\ \mathcal{L}[\mathcal{M}(\mathbf{X} + \epsilon_A)] < \mathcal{L}[\mathcal{M}(\mathbf{X} + \epsilon_B)] \end{cases}$$

- Cả ϵ_A và ϵ_B đều là **nhiễu thành công**, và độ đo chuẩn l_0 trên ϵ_A thấp hơn trên ϵ_B :

$$\begin{cases} \text{SUCCESS}(\mathcal{M}(\mathbf{X}), \mathcal{M}(\mathbf{X} + \epsilon_A)) = \text{true} \\ \text{SUCCESS}(\mathcal{M}(\mathbf{X}), \mathcal{M}(\mathbf{X} + \epsilon_B)) = \text{true} \\ \|\epsilon_A\|_0 < \|\epsilon_B\|_0 \end{cases}$$

Nếu A không tốt hơn B và B cũng không tốt hơn A thì khi đó hai cá thể A và B được xem là tốt như nhau.

3.2.6 Chọn lọc

Để tăng độ thích nghi của quần thể hay tăng độ thích nghi (ở đây là độ tốt của nhiều) trung bình của các cá thể trong quần thể, ta cần chọn ra từ thế hệ trước những cá thể tốt để làm nền tảng cho quá trình lai ghép được diễn ra ở thế hệ tiếp theo. Quá trình này được gọi là chọn lọc.

Có nhiều phép chọn lọc khác nhau, và trong phương pháp đề xuất, chúng tôi sẽ sử dụng phép chọn lọc giải đấu (**tournament selection**)⁷, chi tiết về tournament selection được mô tả trong Thuật giải 6. Với quần thể cần được chọn lọc P , quá trình chọn lọc bắt đầu bằng việc sao chép quần thể P vào một quần thể tạm thời $Temp$, và khởi tạo quần thể kết quả (gồm các cá thể được giữ lại) S là một tập rỗng (dòng 1, 2 trong Thuật giải 6). Tại mỗi vòng lặp, một tập con ngẫu nhiên của $Temp$ là t sẽ được chọn ra, với $|t| = T$ (T được gọi là kích thước giải đấu - **tournament size**), hoặc nếu $|Temp| < T$ thì toàn bộ tập $Temp$ sẽ được chọn làm tập t (dòng 6 đến dòng 9). Từ t , cá thể tốt nhất sẽ được chọn ra theo Định nghĩa 1 và thêm vào tập kết quả S , đồng thời loại các phần tử trong t ra khỏi $Temp$ (các dòng từ 10 đến 12 trong Thuật giải 6). Các vòng lặp sẽ được diễn ra cho đến khi kích thước của tập S đúng bằng N (dòng 3 trong Thuật giải 6), với N là kích thước mong muốn của tập được chọn, trong phương pháp của chúng tôi N sẽ luôn được gán bằng s là kích thước quần thể được khởi tạo ban đầu. Bên cạnh đó, tại bất kỳ vòng lặp nào trong quá trình chọn lọc, nếu tập $Temp$ trở thành tập rỗng (và kích thước tập S vẫn chưa bằng N), tập $Temp$ sẽ được gán lại thành một bản sao P (dòng 4, 5 trong Thuật giải 6).

Với tournament selection, ta thấy rằng cá thể tốt nhất trong quần thể sẽ luôn được chọn để đóng góp vào việc tạo ra thế hệ tiếp theo. Hơn nữa, các cá thể còn lại, trừ cá thể kém nhất cũng sẽ có cơ hội được chọn làm cha mẹ. Điều này đảm bảo rằng những tính chất tốt nhất (từ cá thể tốt nhất) sẽ luôn được truyền lại cho các thế hệ tiếp theo, đồng thời vẫn giữ được sự đa dạng của các cá thể do phép chọn lọc này không chỉ tập trung khai thác những cá thể tốt. Việc duy trì sự đa dạng của các cá

⁷Từ này về sau, "chọn lọc giải đấu" sẽ được thay thế bằng "tournament selection"; "kích thước giải đấu" sẽ được thay thế bằng "tournament size"

thể là cần thiết cho thuật toán GA bởi nó là một trong các yếu tố quan trọng (bên cạnh phép đột biến diễn ra với tỉ lệ nhỏ) giúp cho thuật toán có thể khám phá những giải pháp tiềm năng mới và tránh việc hội tụ sớm.

Thuật giải 6 TournamentSelection (chọn lọc giải đấu)

Input: Quần thể đã được đánh giá P , kích thước giải đấu $T < |P|$, kích thước mong muốn của tập được chọn N .

Output: Tập được chọn S .

```

1: Temp  $\leftarrow P$ 
2:  $S \leftarrow \emptyset$ 
3: while  $|S| < N$  do
4:   if  $|Temp| = 0$  then
5:     Temp  $\leftarrow P$ 
6:   if  $|Temp| < T$  then
7:      $t \leftarrow Temp$ 
8:   else
9:      $t \leftarrow RANDOMSELECT(T, Temp)$ 
10:     $t_{best} \leftarrow \max(t_i \in t)$ 
11:     $S \leftarrow S \cup \{t_{best}\}$ 
12:    Temp  $\leftarrow Temp \setminus t$ 
13: return  $S$ 
```

3.2.7 Sử dụng Multi-objective Elitist Archive

Chúng tôi sử dụng một kỹ thuật trong tối ưu hóa đa mục tiêu (**Multi-objective optimization - MOO**) gọi là **elitist archive**⁸ [14] để tăng thêm khả năng cân bằng giữa việc khám phá (**exploration**) những giải pháp chưa biết và khai thác (**exploitation**) những cá thể đã được tạo ra. Elitist archive là một cơ chế lưu trữ dùng để theo dõi những cá thể tốt tìm được xuyên suốt quá trình tìm kiếm. Nó có thể đóng vai trò như một quần thể thứ hai hoặc là một chỉ dẫn cho quá trình tìm kiếm.

Chúng tôi sử dụng archive trong phương pháp đề xuất với vai trò thứ hai là một chỉ dẫn. Trước tiên, archive sẽ được khởi tạo là một bản sao của quần thể ban đầu (quần thể ở thế hệ đầu tiên), sau đó, archive sẽ được cập nhật liên tục. Tại mỗi thế hệ, các cá thể ở thế hệ hiện tại sẽ được so sánh với các cá thể đã có trong archive,

⁸Từ này về sau, "elitist archive" sẽ được gọi tắt là "archive"

và một cá thể sẽ được thêm vào archive nếu nó không bị thống trị (**non-dominated**) bởi bất kì cá thể nào có trong archive cả, đồng thời, các cá thể trong archive bị thống trị bởi bất kì cá thể bên ngoài nào cũng sẽ bị loại bỏ. Đối với ngữ cảnh là bài toán này, định nghĩa về sự thống trị giữa cá thể này và cá thể khác sẽ như sau:

Định nghĩa 2 (Sự thống trị). Cho hai cá thể A và B tạo nên hai nhiều lần lượt là ϵ_A và ϵ_B . A được xem là bị B thống trị khi và chỉ khi:

$$\begin{cases} \mathcal{L}[\mathcal{M}(\mathbf{X} + \epsilon_A)] > \mathcal{L}[\mathcal{M}(\mathbf{X} + \epsilon_B)] \\ \|\epsilon_A\|_0 > \|\epsilon_B\|_0 \end{cases} \quad (3.1)$$

Điểm đặc biệt của sự thống trị trong Định nghĩa 2 là coi hai giá trị hàm măt măt \mathcal{L} và l_0 -norm có tầm quan trọng như nhau, chỉ khi một cá thể có hai giá trị này vượt trội hơn hai giá trị tương ứng của cá thể khác thì mới được xem là cá thể thống trị. Nhờ có tính chất này, chúng tôi sử dụng archive để những cá thể dù không tạo nên nhiều thành công, nhưng lại có giá trị l_0 -norm nhỏ, hay nói cách khác là tạo ra cách nhiều **không nhận thấy được**, vẫn có cơ hội đóng góp vào việc hướng dẫn quá trình tìm kiếm phát hiện ra những cá thể mới có khả năng tạo ra nhiều vừa thành công, vừa khó nhận biết được. Để làm được điều này, tại mỗi thế hệ, ở giai đoạn đột biến các cá thể, thay vì thực hiện phép đột biến bình thường như mục 3.2.4, sẽ có 50% khả năng thuật toán của chúng tôi chọn ngẫu nhiên một cá thể từ tập 50% những cá thể trong archive có giá trị hàm măt măt \mathcal{L} thấp nhất, để thực hiện thay thế các đặc trưng Pos và Δ với cá thể đang được đột biến. Quá trình thay thế đặc trưng sẽ được thực hiện tương tự Thuật giải 2 (đối với tấn công *image classification*) hoặc Thuật giải 3 (đối với tấn công *object detection*), nhưng với tỉ lệ pixels được thay thế là p_m , và chỉ thực hiện một chiều là thay thế vào cá thể đang được đột biến để tạo ra duy nhất một cá thể mới.

3.3 Hàm măt măt

Kết quả dự đoán của mô hình lên một bức ảnh bị thêm nhiễu sẽ được đánh giá bằng hàm măt măt. Giá trị hàm măt măt cùng với giá trị l_0 -norm sẽ được dùng để đánh

giá nhiễu do một cá thể tạo ra có tốt hay không. Tùy thuộc vào loại mô hình bị tấn công, chúng tôi sẽ thiết kế các công thức hàm mất mát khác nhau.

3.3.1 Hàm mất mát cho tấn công mô hình image classification

Như đã đề cập ở 2.2.1, đầu ra của các mô hình *image classification* là xác suất bức ảnh (ở đầu vào) thuộc vào từng lớp. Mục tiêu của tấn công lên *image classification*, như đã trình bày ở biểu thức 2.5 là khiến cho mô hình đưa ra kết quả dự đoán khác với nhãn đúng của bức ảnh đầu vào. Để làm được điều này, hàm mất mát cần được thiết kế sao cho sẽ nhận giá trị càng nhỏ khi xác suất bức ảnh thuộc vào lớp đúng càng thấp và thuộc vào các còn lại càng cao. Cụ thể, trong các nghiên cứu [4, 22], hàm mất mát được thiết kế như sau.

$$\mathcal{L}(\mathcal{F}(\mathbf{X} + \epsilon)) = f_y - f_r \quad (3.2)$$

Với \mathcal{F} là một mô hình *image classification*, f_i là xác suất mô hình dự đoán bức ảnh thuộc vào lớp i , y là nhãn đúng của ảnh chưa bị thêm nhiễu \mathbf{X} , $r = \operatorname{argmax}_{i \neq y} f_i(\mathbf{X} + \epsilon)$ là lớp có xác suất bức ảnh thuộc vào cao nhất khác nhãn đúng của \mathbf{X} .

Tuy nhiên, hàm mất mát 3.2 chỉ có thể sử dụng được khi ta có thể biết được xác suất của tất cả cá lớp f_i , hoặc ít nhất là biết được xác suất của những lớp mà bức ảnh có khả năng thuộc vào nhất. Điều này sẽ khó có thể thực hiện được trong thực tế do để thân thiện với người dùng và tăng tính bảo mật, các mô hình trong thực tế thường sẽ chỉ trả về một lớp mà bức ảnh có khả năng thuộc về cao nhất cùng với xác suất tương ứng của lớp đó. Xem xét đến trường hợp này, chúng tôi đã thiết kế một hàm mất mát mới chỉ sử dụng lớp mà mô hình dự đoán và xác suất của lớp đó. Cụ thể, giả sử ϵ được tạo ra trong thế hệ g của giải thuật GA, khi đó giá trị hàm mất mát

$\mathcal{L}(\mathcal{F}(\mathbf{X} + \epsilon))$ sẽ được tính như sau:

$$\begin{aligned}\mathcal{L}(\mathcal{F}(\mathbf{X} + \epsilon)) &= F(\mathbf{X} + \epsilon) + \bar{F}(\mathbf{X} + \epsilon) \\ \text{với } F(\mathbf{X} + \epsilon) &= (-1)^{\mathbb{I}(y' \neq y)} \cdot f_{y'}(\mathbf{X} + \epsilon) \\ \bar{F}(\mathbf{X} + \epsilon) &= F(\mathbf{X} + \epsilon) - \hat{F}^{(g-1)} \\ \hat{F}^{(g)} &= \min(F(\mathbf{X} + \epsilon_I)) \quad \forall I \in P^{(g)} \\ \hat{F}^{(0)} &= 0\end{aligned}\tag{3.3}$$

Trong đó:

- y' và y là lớp mà mô hình dự đoán cho ảnh đã thêm nhiễu $\mathbf{X} + \epsilon$ và ảnh gốc \mathbf{X} .
- $P^{(g)}$ là tập hợp các cá thể được tạo ra và được tính giá trị hàm mất mát trong thế hệ g .
- ϵ_I là nhiễu được tạo ra bởi cá thể I .
- $\mathbb{I}(y' \neq y)$ là hàm nhận giá trị 1 nếu $y' \neq y$ và 0 trong trường hợp ngược lại.

Với hàm mất mát 3.3, thuật toán tìm kiếm cũng sẽ cố gắng làm giảm xác suất bức ảnh thuộc vào lớp đúng và tăng xác suất thuộc vào các lớp còn lại, thể hiện ở giá trị $F(\mathbf{X} + \epsilon)$. Ngoài ra, thành phần $\bar{F}(\mathbf{X} + \epsilon)$ được thêm vào để thể hiện sự phát triển của cá thể so với thế hệ trước đó, giúp khuyến khích thuật toán tìm ra được các nhiễu mới tốt hơn so với ở thế hệ cũ.

So sánh hàm mất mát 3.3 với 3.2, hàm 3.2 sẽ hiệu quả hơn bởi nó giúp cho thuật toán tối ưu hóa mục tiêu (giảm xác suất bức ảnh thuộc vào lớp đúng và tăng xác suất thuộc vào lớp sai) một cách trực tiếp, trong khi hàm 3.3 chỉ sử dụng xác suất cao nhất được trả về không thể làm được điều này. Tuy nhiên, hàm 3.3 được thiết kế vẫn có khả năng giúp thuật toán đạt được mục tiêu đặt ra, hơn nữa, lại còn có khả năng áp dụng vào tấn công thực tế cao hơn so với 3.2.

3.3.2 Hàm mất mát cho tấn công mô hình object detection

Đầu ra của các mô hình *object detection* phức tạp hơn so với *image classification*, nó có thể được xem như là một sự kết hợp của **kết quả dự đoán các bounding boxes** và

các **kết quả phân lớp** cho từng đối tượng trong các bounding boxes dự đoán được. Từ đây, chúng tôi có thể viết lại đầu ra của một mô hình *object detection* bất kì dưới dạng một tập hợp các bộ ba (B_i, c_i, f_i) ; với B_i, c_i, f_i lần lượt chỉ vị trí (bounding box), nhãn và độ tin cậy của dự đoán cho đối tượng thứ i trên bức ảnh.

Mục tiêu của tấn công lên các mô hình *object detection*, như đã trình bày ở biểu thức 2.8, là khiến cho ít nhất một đối tượng O trên bức ảnh bị thêm nhiễu có kết quả nhãn dự đoán khác với nhãn của chính đối tượng đó trên kết quả dự đoán ở ảnh gốc, nói rõ hơn, mô hình vẫn dự đoán đúng vị trí (bounding box - B_O) của đối tượng nhưng lại dự đoán sai nhãn c_O của đối tượng đó. Từ mục tiêu tấn công như trên, chúng tôi thiết kế hàm mất mát trong trường hợp này như sau.

Định nghĩa 3 (Hàm mất mát cho tấn công mô hình *object detection*). Cho kết quả dự đoán của một mô hình *object detection* đối với bức ảnh chưa bị thêm nhiễu là $\mathcal{D}(\mathbf{X}) = \{O_i = (B_i, c_i, f_i), i = \overline{1..n}\}$ và đối với bức ảnh đã bị thêm nhiễu là $\mathcal{D}(\mathbf{X} + \epsilon)$ cũng là một tập hợp các objects. Giá trị của hàm mất mát đối với kết quả dự đoán của mô hình cho một bức ảnh bị thêm nhiễu $\mathcal{L}(\mathcal{D}(\mathbf{X} + \epsilon))$, trong đó nhiễu ϵ là do một cá thể được tạo ra tại thế hệ thứ g được tính như sau:

$$\mathcal{L}(\mathcal{D}(\mathbf{X} + \epsilon)) = \sum_{i=1}^n (F_i(\mathbf{X} + \epsilon) + \bar{F}_i(\mathbf{X} + \epsilon) + \mathcal{I}_i(\mathbf{X} + \epsilon) + \gamma_i \|\epsilon_{B_i}\|_0); \quad (3.4)$$

$$\text{Với: } F_i(\mathbf{X} + \epsilon) = \begin{cases} \frac{1}{|A_i(\mathbf{X} + \epsilon)|} \sum_{O \in A_i(\mathbf{X} + \epsilon)} (-1)^{\mathbb{I}(c_i \neq c_O)} \cdot f_O(\mathbf{X} + \epsilon), & \text{nếu } |A_i| \neq 0 \\ 0, & \text{nếu } |A_i| = 0 \end{cases}$$

$$A_i(\mathbf{X} + \epsilon) = \{O \in \mathcal{D}(\mathbf{X} + \epsilon) \mid \text{IoU}(B_i, B_O) > \delta\} \quad (3.5)$$

$$\bar{F}_i(\mathbf{X} + \epsilon) = F_i(\mathbf{X} + \epsilon) - \hat{F}_i^{(g-1)} \quad (3.6)$$

$$\hat{F}_i^{(g)} = \min(F_i(\mathbf{X} + \epsilon_I)) \quad \forall I \in P^{(g)}$$

$$\mathcal{I}_i(\mathbf{X} + \epsilon) = \frac{1}{|A_i(\mathbf{X} + \epsilon)|} \sum_{O \in A_i(\mathbf{X} + \epsilon)} \log(1 - \text{IoU}(B_i, B_O) + \zeta) \quad (3.7)$$

Trong đó:

- B_O, c_O, f_O lần lượt là các giá trị (B, c, f) đối với đối tượng O .
- ϵ_{B_i} là tập con của ϵ , chỉ lấy các giá trị tại các vị trí thuộc bounding box B_i

- γ_i là tham số phụ thuộc vào độ lớn của bounding box B_i
- δ, ζ là các hằng số
- $\mathbb{I}(c_i \neq c_O)$ là hàm nhận giá trị 1 nếu $c_i \neq c_O$ và 0 trong trường hợp ngược lại.
- $P^{(g)}$ là tập hợp các cá thể được tạo ra và được tính giá trị hàm mất mát trong thế hệ g .
- ϵ_I là nhiễu được tạo ra bởi cá thể I .

Tập A_i của ảnh đã thêm nhiễu $\mathbf{X} + \epsilon$ ở công thức 3.5 là tập hợp gồm các đối tượng được dự đoán trong ảnh bị thêm nhiễu được cho là có vị trí trùng khớp với đối tượng O_i trong kết quả dự đoán ở bức ảnh gốc. Hai đối tượng sẽ được xem là trùng nhau nếu giá trị IoU giữa hai bounding boxes lớn hơn ngưỡng δ . Và đối với một đối tượng O_i ở bức ảnh gốc, chúng tôi sẽ xét tất cả các đối tượng trong tập $A_i(\mathbf{X} + \epsilon)$ của bức ảnh bị thêm nhiễu để tính toán giá trị hàm mất mát.

Khi thêm nhiễu, sẽ có khả năng cao khiến cho mô hình không phát hiện được các đối tượng trong bức hình sau khi bị thêm nhiễu, hoặc dự đoán sai vị trí của các đối tượng (ví dụ như ở Hình 2.5). Tuy nhiên, những nhiễu gây nên hiện tượng này sẽ không được chúng tôi xem là một nhiễu thành công theo biểu thức 2.8, những nhiễu được cộng nhận là thành công là những nhiễu làm cho mô hình vẫn dự đoán đúng vị trí của đối tượng, nhưng lại sai nhẫn.

Để theo dõi và đánh giá độ tốt của nhiễu dựa trên hiện tượng này, chúng tôi thêm thành phần \mathcal{I}_i được mô tả ở Công thức 3.7. Thành phần này có mục đích khuyến khích thuật toán tìm kiếm các nhiễu không làm cho mô hình dự đoán sai vị trí hoặc không phát hiện được đối tượng trên bức ảnh sau khi bị tấn công; bởi vì khi có càng nhiều đối tượng ở bức ảnh bị tấn công vẫn được dự đoán đúng vị trí so với bức ảnh gốc, các giá trị của \mathcal{I}_i sẽ càng nhỏ. Hằng số ζ trong công thức của \mathcal{I}_i giúp giữ cho giá trị bên trong log không quá gần giá trị 0.

Thành phần chủ chốt trong hàm mất mát là F_i . Giá trị này sẽ nhỏ (trung bình cộng của nhiều giá trị âm) nếu nhiều thành công trên đối tượng O_i (khiến cho kết quả dự đoán của mô hình đối với đối tượng O_i bị sai nhẫn) và ngược lại. Đồng thời, nếu có hai nhiễu cùng thành công trên O_i thì nhiễu nào có độ tin cậy dự đoán trên nhẫn sai cao hơn sẽ được xem là tốt hơn, thể

hiện ở $(-1) \cdot f_O(\mathbf{X} + \epsilon)$ (do $c_i \neq c_O$). Hoặc, nếu có hai nhiễu không thành công trên O_i thì nhiễu có độ tin cậy dự đoán trên nhãn đúng thấp hơn sẽ được xem là tốt hơn, thể hiện ở $1 \cdot f_O(\mathbf{X} + \epsilon)$ (do $c_i = c_O$).

Hơn nữa, để đánh giá mức độ phát triển của quần thể hiện tại so với quần thể trước đó, chúng tôi thêm thành phần \bar{F}_i là hiệu số của giá trị F_i ở thế hệ này so với thế hệ trước. Tuy nhiên, ở mỗi thế hệ, ta sẽ không chỉ đánh giá một mà là nhiều cá thể, số lượng cá thể được tạo ra ở mỗi thế hệ sẽ tương ứng với kích thước quần thể. Để tăng cường khả năng tìm thấy nhiều cá thể tốt hơn so với thế hệ trước, chúng tôi lấy $\hat{F}_i^{(g-1)} = \min(F_i(\mathbf{X} + \epsilon_I))$, $\forall I \in P^{(g-1)}$ là giá trị nhỏ nhất (tốt nhất) trong số các cá thể ở thế hệ trước để so với các kết quả hiện tại.

Bên cạnh đó, để khuyến khích thuật toán tìm kiếm các nhiễu **không nhận thấy được**, chúng tôi cũng thêm thành phần l_0 -norm của giá trị nhiễu, giá trị này càng nhỏ thì nhiễu được tạo thành sẽ càng khó để nhận biết. Tuy nhiên, chúng tôi nhận thấy rằng độ lớn của giá trị $\|\epsilon_{B_i}\|_0$ sẽ phụ thuộc vào kích thước của đối tượng đang được xét tới, do đó, chúng tôi thêm thành phần chuẩn hóa $\gamma_i = \frac{1}{h_{O_i} \cdot w_{O_i} \cdot k}$ (h_{O_i}, w_{O_i}, k lần lượt là chiều cao của đối tượng, chiều rộng của đối tượng và tỉ lệ số pixel tối đa được thay đổi trên mỗi đối tượng) để tránh sự khác biệt quá lớn giữa các đối tượng có kích thước lớn và kích thước nhỏ, bởi lẽ $\gamma_i \|\epsilon_{B_i}\|_0$ sẽ luôn có giá trị thuộc vào $(0, 1]$, do giá trị lớn nhất của $\|\epsilon_{B_i}\|_0$ là $h_{O_i} \cdot w_{O_i} \cdot k$.

Với hàm mất mát được thiết kế như trên, một cá thể sẽ càng tốt nếu nhiều được nó tạo ra:

- Là nhiều thành công, có độ tin cậy trên nhãn sai lớn; hoặc là nhiều không thành công nhưng có độ tin cậy trên nhãn đúng nhỏ (giá trị F_i nhỏ).
- Có sự phát triển so với thế hệ trước đó (giá trị \bar{F}_i nhỏ).
- Vẫn khiến cho mô hình phát hiện đúng vị trí các đối tượng như ở bức hình chưa được thêm nhiễu (giá trị I_i nhỏ).
- Không dễ nhận thấy (giá trị $\gamma_i \|\epsilon_{B_i}\|_0$ nhỏ).

Sẽ rất khó để một cá thể có thể thỏa mãn tất cả các yếu tố nêu trên, ví dụ nếu nhiều thành công (F_i nhỏ) thì thường sẽ đi kèm với dễ nhận thấy ($\gamma_i \|\epsilon_{B_i}\|_0$ lớn). Vì vậy nên chúng tôi sử dụng phép cộng các thành phần để cân bằng các yếu tố này.

Chương 4

THỰC NGHIỆM

4.1 Thiết lập thực nghiệm

4.1.1 Các mô hình mục tiêu

Để kiểm tra tính hiệu quả của phương pháp đề xuất, chúng tôi tiến hành thực nghiệm tấn công lên các mô hình *image classification* và *object detection* khác nhau, và so sánh hiệu suất của phương pháp đề xuất với các phương pháp tấn công đã được nghiên cứu trước đó. Bảng 4.1 trình bày tóm tắt các đặc điểm của những mô hình được chúng tôi chọn làm mục tiêu tấn công.

BẢNG 4.1: Tóm tắt đặc điểm của các mô hình được chọn làm mục tiêu tấn công

Mô hình	Số tham số	Bộ dữ liệu huấn luyện/tấn công	Độ đo đánh giá hiệu năng tấn công
AT1	36.5M	CIFAR-10/	Attack Success Rate (ASR);
AT2	36.5M	CIFAR-10	l_p trung bình ($p = 0, 2$)
YOLOv8	3.5M		
YOLOv9	2.0M		
YOLOv10	2.3M		
YOLOv11	2.6M		
DETR	41.3M	COCO/ PascalVoc2007	Attack Success Rate (ASR); mean Average Precision (mAP@0.50); số lần gọi mô hình trung bình (Queries)
DINO	46.7M		

Các mô hình image classification

Đối với tấn công lên *image classification*, theo thiết lập thực nghiệm từ nghiên cứu [22], chúng tôi chọn hai mô hình mục tiêu là hai mô hình sử dụng kiến trúc WideResNet-70-16 được huấn luyện đối kháng (**Adversarial Training**) trên bộ dữ liệu CIFAR-10 lần lượt là AT1 [10] và AT2 [9]. Các tham số đã được huấn luyện trước của hai mô hình được cung cấp thông qua thư viện RobustBench [3]. Độ chính xác (**accuracy**) của hai mô hình đã được huấn luyện lần lượt là 89.48% và 87.50% trên tập đánh giá của bộ dữ liệu CIFAR-10.

Các mô hình object detection

Đối với tấn công lên các mô hình *object detection*, chúng tôi chọn các mô hình mục tiêu là bốn phiên bản mới nhất của YOLO trong hai năm gần đây là YOLOv8 [19], YOLOv9 [21], YOLOv10 [20] và YOLOv11 [12]. Việc chọn YOLOs là các mô hình mục tiêu không chỉ để đánh giá hiệu quả của phương pháp tấn công mà còn để kiểm tra độ tin cậy và an toàn của những mô hình này. Chúng tôi cho rằng việc kiểm tra khả năng chống chịu này là cần thiết vì các mô hình YOLOs hiện nay có rất nhiều ứng dụng trong thực tế nhờ vào tốc độ xử lý nhanh và độ chính xác cao mà nó mang lại.

Bên cạnh đó, các mô hình được thiết kế dựa trên kiến trúc transformer cũng đã được nhiều nghiên cứu chỉ ra tiềm năng trong việc áp dụng vào bài toán phát hiện đối tượng [5]. Vì vậy, chúng tôi cũng chọn hai mô hình giải quyết bài toán *object detection* sử dụng kiến trúc transformer là DETR [2] và DINO [24] và thực hiện các thực nghiệm tấn công lên chúng.

4.1.2 Bộ dữ liệu được sử dụng

Tấn công lên các mô hình image classification

CIFAR-10 [13] là bộ dữ liệu được chúng tôi sử dụng trong các thực nghiệm tấn công lên mô hình *image classification*. Bộ dữ liệu CIFAR-10 chứa 60,000 ảnh màu RGB có kích thước 32×32 , được chia thành 10 lớp khác nhau, mỗi lớp có 6,000 ảnh. Bộ dữ liệu được chia thành tập huấn luyện và tập đánh giá theo tỉ lệ là 5 : 1.

Trong các thực nghiệm, chúng tôi sẽ chọn ngẫu nhiên 100 bức ảnh mà mỗi mô hình AT1 và AT2 dự đoán đúng từ tập đánh giá và thực hiện tìm kiếm nhiều trên các bức ảnh này để tấn công mô hình tương ứng.

Tấn công lên các mô hình object detection

Chúng tôi sử dụng các bức ảnh từ bộ dữ liệu PascalVoc2007 [6] để thực hiện các thực nghiệm tấn công lên các mô hình *object detection*. PascalVoc2007 gồm các hình ảnh từ thực tế với nhiều đối tượng, mỗi đối tượng được xác định bằng vị trí (**bounding boxes**) và nhãn (hay lớp - **label/class**). Bộ dữ liệu có tổng cộng 9963 bức ảnh với tổng số đối tượng được gán nhãn là khoảng hơn 24,000. Các đối tượng trong ảnh được phân chia vào 20 lớp với 4 nhóm chính là *động vật, phương tiện giao thông, vật thể trong nhà và con người*. Bộ dữ liệu được phân chia thành tập huấn luyện/kiểm định (train/validation) và tập kiểm tra (test) theo tỉ lệ 1 : 1.

Trong các thực nghiệm, chúng tôi chọn ra 500 ảnh ngẫu nhiên từ tập kiểm tra của bộ dữ liệu và thực hiện tìm kiếm nhiều trên những bức ảnh này để tấn công các mô hình.

4.1.3 Độ đo đánh giá hiệu năng tấn công

Đối với tấn công lên các mô hình *image classification*, chúng tôi sử dụng hai độ đo để đánh giá hiệu năng của các phương pháp tấn công là **tỉ lệ tấn công thành công** (attack success rate - ký hiệu là ASR) và **giá trị l_p trung bình**.

Đối với tấn công lên các mô hình *object detection*, chúng tôi sử dụng ba độ đo để đánh giá hiệu năng tấn công là **ASR**, **giá trị mean Average Precision (mAP@0.50)** và **số lần gọi mô hình trung bình** (ký hiệu là Queries). Bên cạnh đó, chúng tôi sẽ xét đến hai điều kiện tấn công thành công là theo điều kiện 2.7, chỉ cần khiến mô hình không phát hiện đúng đối tượng và theo điều kiện 2.8, phải khiến mô hình dự đoán sai nhãn của đối tượng; chúng tôi sẽ ghi lại các kết quả **ASR** theo hai trường hợp này độc lập với nhau.

Chi tiết về từng độ đo sẽ được trình bày ở các phần bên dưới.

Tỉ lệ tấn công thành công

Giá trị ASR được tính bằng tỉ lệ giữa số ảnh bị tấn công thành công trên tổng số ảnh bị tấn công.

$$\text{ASR} = \frac{\#\text{Ảnh bị tấn công thành công}}{\#\text{Ảnh bị tấn công}} \quad (4.1)$$

Một bức ảnh bị tấn công thành công khi phương pháp tấn công có thể tìm được nhiều thành công, theo biểu thức 2.5 đối với mô hình *image classification* hoặc theo biểu thức 2.8 đối với mô hình *object detection*, trên bức ảnh đó.

Ngoài ra, đối với tấn công lên các mô hình *object detection*, các bức ảnh bị tấn công là những bức ảnh mà trên đó mô hình phát hiện được có ít nhất một đối tượng. Những bức ảnh mà kết quả dự đoán của mô hình đối với chúng không có bất kì đối tượng nào sẽ không bị tấn công.

Giá trị ASR **càng cao** thì phương pháp tấn công sẽ được coi là càng hiệu quả.

Giá trị l_p trung bình

Các giá trị l_0 và l_2 sẽ chỉ được tính trên các bức hình bị tấn công thành công. l_0 của một bức ảnh bị tấn công theo các phương pháp tấn công thua sẽ được tính bằng số lượng pixel bị thay đổi của bức ảnh đó. Trong khi đó, l_2 -norm sẽ được tính theo công thức tính l_2 -norm thông thường giữa bức ảnh gốc ban đầu và bức ảnh sau khi bị thay đổi (các giá trị kênh màu được chuẩn hóa về đoạn $[0, 1]$), hay nói cách khác, với l_2 , ta sẽ tính $\|X' - X\|_2$ với X là bức ảnh gốc và X' là bức ảnh sau khi bị thay đổi. Cuối cùng, ta lấy trung bình các giá trị l_0 và l_2 trên tất cả các ảnh bị tấn công thành công.

Các giá trị l_0 và l_2 **càng thấp** đồng nghĩa với phương pháp tấn công có khả năng tạo ra những nhiễu không nhận thấy được càng tốt. Trong ngữ cảnh của khóa luận này, giá trị l_0 vẫn được ưu tiên nhiều hơn l_2 .

mAP@0.50

Mean Average Precision (mAP) là một độ đo phổ biến dùng để đánh giá độ chính xác của các mô hình *object detection*, mAP được tính bằng trung bình cộng của các giá trị

Average Precision (AP) đối với mỗi lớp trong bộ dữ liệu.

$$\text{mAP} = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \text{AP}_c \quad (4.2)$$

với \mathcal{C} là tập các lớp trong bộ dữ liệu.

Trước khi đi vào chi tiết về cách tính AP, chúng tôi sẽ làm rõ một số khái niệm cần thiết để có thể tính được giá trị này bao gồm giá trị *Intersection over Union* (ký hiệu là IoU, đã được trình bày rõ ở mục 2.2.1), giá trị *Precision* và giá trị *Recall*.

- Các giá trị *Precision* và *Recall* được tính như sau:

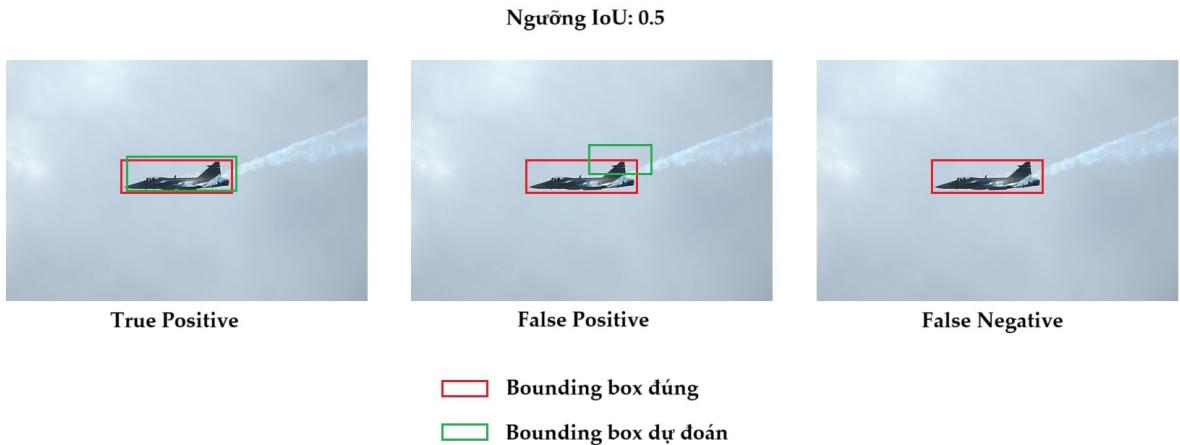
$$\text{Precision} = \frac{TP}{TP + FP} \quad (4.3)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (4.4)$$

trong đó: TP là *True Positive*, FP là *False Positive* và FN là *False Negative*.

Để xác định một kết quả dự đoán bounding box của đối tượng là TP , FP hay FN , trước tiên kết quả dự đoán đó phải có cùng lớp đúng với đối tượng đang xét, sau đó ta sẽ tính giá trị IoU giữa bounding box được dự đoán với bounding box đúng của đối tượng. Nếu giá trị IoU lớn hơn giá trị ngưỡng δ được định nghĩa từ trước (gọi là ngưỡng IoU) thì kết quả dự đoán sẽ là *True Positive*, ngược lại nếu giá trị IoU nhỏ hơn ngưỡng thì kết quả dự đoán là *False Positive*. Trong trường hợp mô hình không thể xác định được bounding box cho đối tượng, kết quả dự đoán sẽ được xem là *False Negative*. Hình 4.1 minh họa cho các trường hợp trên.

Tiếp theo, chúng tôi sẽ trình bày quá trình tính giá trị *Average Precision* đối với mỗi lớp trong bộ dữ liệu. Đầu tiên, ta sắp xếp các kết quả dự đoán của lớp đó trên tất cả bức ảnh được xét theo thứ tự độ tin cậy dự đoán (*confidence score*) giảm dần. Sau đó, chúng ta sẽ đi theo chiều giảm *confidence score* để tính dần các giá trị *Precision* và *Recall* theo thứ tự, ta ký hiệu các giá trị *Precision* và *Recall* tính được lần lượt là $Prec(i)$, $Rec(i)$ với i chạy từ 1 đến N là số lượng giá trị *Recall* phân biệt và $Rec(0) = 0$. Sau đó, giá trị AP có thể được tính theo các phương pháp **11-point Interpolation**, **All-point Interpolation** hay **101-point Interpolation** theo [16].



HÌNH 4.1: Minh họa cho các trường hợp TP, FP, FN với ngưỡng IoU là $\delta = 0.5$.

BẢNG 4.2: Các kết quả dự đoán cho đồi tượng *con mèo* cùng với các giá trị *Precision* và *Recall* theo thứ tự.

Thứ tự	True Positive?	Precision	Recall
1	Đúng	1.00	0.25
2	Sai	0.50	0.25
3	Đúng	0.67	0.50
4	Sai	0.50	0.50
5	Đúng	0.60	0.75
6	Đúng	0.67	1.00

Lấy ví dụ, có 4 đồi tượng *con mèo* trong bộ dữ liệu và 6 dự đoán cho đồi tượng đó. Các kết quả dự đoán được sắp xếp theo thứ tự giảm dần *confidence score* cùng các giá trị *Precision* và *Recall* được tính theo thứ tự được thể hiện ở Bảng 4.2. Khi này, nếu sử dụng phương pháp **All-point Interpolation** thì giá trị AP sẽ được tính như sau:

$$\begin{aligned}
 AP_{mèo} &= \sum_{i=1}^N [Rec(i) - Rec(i-1)] \cdot Prec(j^*), \quad j^* = \operatorname{argmax}_{j \geq i} Prec(j) \quad (4.5) \\
 &= (0.25 - 0) \cdot 1.0 + (0.5 - 0.25) \cdot 0.67 \\
 &\quad + (0.75 - 0.25) \cdot 0.67 + (1.0 - 0.75) \cdot 0.67 \\
 &= 0.7525
 \end{aligned}$$

Trong các thực nghiệm của chúng tôi, giá trị AP sẽ được tính theo phương pháp **101-point Interpolation** [16]. Bên cạnh đó, độ đo mAP của chúng tôi là mAP@0.50, nghĩa là các giá trị *Precision* và *Recall* sẽ được tính với ngưỡng IoU là 0.50.

Vì tấn công sẽ đánh lừa mô hình, khiến cho mô hình đưa ra các dự đoán sai lệch, nên các phương pháp tấn công càng tốt sẽ làm cho giá trị mAP trên những bức ảnh bị thêm nhiễu **càng thấp**.

Số lần gọi mô hình trung bình

Với mỗi nhiễu mới tìm được trong quá trình chạy thuật toán tấn công, ta phải tính lại giá trị hàm mất mát \mathcal{L} (được định nghĩa ở công thức 3.4) đối với nhiều đó để đánh giá độ tốt của nhiễu, và mỗi lần như vậy ta phải gọi mô hình một lần để lấy kết quả dự đoán của mô hình trên bức ảnh bị thêm nhiễu. Để tính giá trị Queries, trước hết, với mỗi bức ảnh bị tấn công thành công, ta đếm số lần gọi mô hình cần thiết để thuật toán tìm được nhiễu thành công đầu tiên trên bức ảnh đó, sau đó ta lấy trung bình các giá trị này trên tất cả những bức ảnh bị tấn công thành công.

Giá trị Queries **càng nhỏ** thì phương pháp tấn công sẽ càng hiệu quả trong điều kiện thực tế hơn, bởi vì khác với khi thực nghiệm, khi tấn công trong thực tế, ta thường sẽ bị hạn chế số lần gọi mô hình.

4.1.4 Thiết lập các phương pháp so sánh và các tham số

Các biến thể của phương pháp đề xuất

Chúng tôi tiến hành thực nghiệm trên 4 biến thể của phương pháp đề xuất là các tổ hợp của hai kích thước giải đấu khác nhau là $T = 2$ và $T = 4$ với việc có sử dụng *archive* hay không. Cụ thể các biến thể được kí hiệu như sau:

- Biến thể với $T = 4$ và sử dụng *archive*, kí hiệu: **GAA** ($T = 4$).
- Biến thể với $T = 2$ và sử dụng *archive*, kí hiệu: **GAA** ($T = 2$).
- Biến thể với $T = 4$ và **không** sử dụng *archive*, kí hiệu: **GA** ($T = 4$).
- Biến thể với $T = 2$ và **không** sử dụng *archive*, kí hiệu: **GA** ($T = 2$).

Nguyên nhân chúng tôi tiến hành thực nghiệm trên bốn biến thể này là để kiểm tra vai trò của kích thước giải đấu, cũng như *archive* trong việc giúp cho thuật toán GA cân bằng giữa khám phá (exploration) các giải pháp chưa biết và khai thác (exploitation) các giải pháp đã có để tìm ra một giải pháp mới.

Cần lưu ý rằng, ở Thuật giải 1, ta tiến hành chọn lọc giải đấu trên tập PO là hợp của tập các cá thể cha mẹ (ở thế hệ trước) và tập các cá thể con được tạo ra ở thế hệ này, có kích thước là $2s$. Vì vậy, với $T = 2$, thuật toán chọn lọc giải đấu (Thuật giải 6) sẽ chỉ thực hiện một vòng lặp duy nhất để chọn ra s cá thể tiến vào thế hệ tiếp theo. Trong đó, cá thể tốt nhất trong tập PO sẽ chắc chắn được chọn, và chỉ được chọn một lần duy nhất, trong khi các cá thể còn lại ngoại trừ cá thể tệ nhất trong quần thể đều cũng sẽ có cơ hội được chọn để làm cha mẹ cho thế hệ sau. Đối với $T = 4$, quá trình chọn lọc giải đấu sẽ trải qua hai vòng lặp, bởi vì sau mỗi vòng lặp ta chỉ chọn được $2s/4 = s/2$ cá thể; và do ở mỗi vòng lặp, cá thể tốt nhất trong tập PO , ta kí hiệu là t^* , đều chắc chắn được chọn đúng một lần nên sau khi kết thúc chọn lọc giải đấu, tập được chọn sẽ chứa hai bản sao của cá thể t^* này. Điểm đáng chú ý ở đây là, nếu ở các thế hệ tiếp theo thuật toán vẫn chưa tìm được một cá thể mới tốt hơn t^* thì quá trình chọn lọc như trên sẽ làm tăng số cá thể t^* trong quần thể sau mỗi thế hệ. Và trong trường hợp lí tưởng, chúng sẽ tăng gấp đôi số cá thể t^* trong tập được chọn lên sau mỗi thế hệ cho đến khi một cá thể tốt hơn được tìm thấy.

So sánh chọn lọc giải đấu với $T = 2$ và $T = 4$, ta thấy rằng trên lý thuyết, thuật toán với $T = 2$ sẽ ưu tiên khám phá (exploration) do các cá thể ở thế hệ trước, dù không tốt vẫn có cơ hội được chọn để tạo ra thế hệ tiếp theo; ngoài ra, cài đặt này vẫn có khả năng khai thác do luôn giữ lại một bản sao của cá thể tốt nhất. Trong khi với $T = 4$, thuật toán sẽ ưu tiên vào khai thác (exploitation) triệt để cá thể tốt nhất trong quần thể là t^* bằng việc liên tục gấp đôi bản sao của cá thể này cho đến khi tìm được một cá thể tốt hơn.

Tấn công lên *image classification*

Chúng tôi sẽ so sánh hiệu năng của phương pháp tấn công đề xuất với hai phương pháp tấn công thừa hộp trống tốt nhất hiện nay (**state-of-the-art**) là **SAPP** [7] và **Homotopy** [25]. Đồng thời, chúng tôi cũng sẽ so sánh với ba phương pháp tấn công

thưa hộp đen cũng sử dụng ý tưởng tính toán tiền hóa là OnePixel [18], Sparse-RS [4] và SA-MOO [22]. Trong đó, theo hiểu biết của chúng tôi, Sparse-RS và SA-MOO đang là hai phương pháp tấn công thưa hộp đen **state-of-the-art**.

Với các phương pháp tấn công hộp trắng, chúng tôi sử dụng các thiết lập tham số được sử dụng trong nghiên cứu của các tác giả. Đối với các phương pháp tấn công hộp đen sử dụng tính toán tiền hóa, trong đó có phương pháp của chúng tôi, với mỗi bức ảnh bị tấn công, chúng tôi đặt số lượng truy vấn mô hình (cũng là số lần đánh giá các giải pháp được tạo thành) là 2000 và cho các thuật toán tấn công sử dụng hết hoàn toàn số lượng truy vấn này để tìm kiếm nhiễu. Đồng thời, chúng tôi đặt số lượng pixels bị thay đổi tối đa là 24, tương đương với tỉ lệ số pixel bị thay đổi là $k = 2.34\%$. Với phương pháp SA-MOO, chúng tôi thiết lập kích thước quần thể theo nghiên cứu của tác giả là $s = 2$, và số lượng thế hệ tối đa là $max_gen = 1000$. Với phương pháp OnePixel sử dụng thuật toán *differential evolution* (DE) và phương pháp chúng tôi đề xuất, chúng tôi thiết lập $s = 16$ và $max_gen = 125$. Các tham số p_c , p_m và p_0 chúng tôi sẽ sử dụng thiết lập theo nghiên cứu [22] là $p_c = 0.4$, $p_m = 0.1$, $p_0 = 0.3$ vì chúng đã được chứng minh là đem lại sự cân bằng tốt giữa việc khám phá các giải pháp mới và khai thác các giải pháp hiện có theo nghiên cứu đó. Ngoài ra, để hạn chế tính ngẫu nhiên của các phương pháp tấn công hộp đen, chúng tôi sẽ chạy mỗi phương pháp này 10 lần với 10 random seeds khác nhau và ghi lại trung bình cộng các giá trị độ đo đánh giá trên tất cả các seeds đó.

Đối với tham số quyết định mức độ thay đổi của các pixels là σ , chúng tôi thiết lập $\sigma = 1$. Cân lưu ý là trước khi thực hiện tấn công, chúng tôi sẽ chuẩn hóa các giá trị pixels trên bức ảnh vào đoạn $[0, 1]$ nên với tham số σ được thiết lập như trên, bất kỳ kênh màu của pixels nào nếu bị thay đổi sẽ luôn nhận giá trị 0 hoặc 1 (vì sau khi thêm nhiễu, kênh màu đó sẽ có giá trị vượt ngoài phạm vi hợp lệ và sẽ được gán lại). Chúng tôi biết rằng thiết lập như vậy sẽ tăng nguy cơ khiến cho nhiễu có thể bị nhận thấy, nhưng nguy cơ này sẽ tăng lên không nhiều do số lượng pixels được phép thay đổi là rất nhỏ. Mặt khác, chúng tôi nhận ra cách làm này cũng sẽ góp phần giúp tăng tỉ lệ tấn công thành công. Vì vậy nên sự đánh đổi này chúng tôi cho là có thể chấp nhận được.

Chúng tôi sẽ không giới hạn kích thước tối đa của elitist archive, chỉ cần một cá thể không bị thống trị bởi những cá thể khác trong archive thì nó sẽ được thêm vào đó.

Tấn công lên *object detection*

Do có rất ít các nghiên cứu hiện nay về việc tấn công hộp đen thưa lên các mô hình *object detection*, chúng tôi chỉnh sửa phương pháp **OnePixel** để có thể tấn công lên loại mô hình này và so sánh hiệu suất tấn công với phương pháp đề xuất của chúng tôi.

Trong tất cả các thực nghiệm của phương pháp đề xuất cũng như của phương pháp **OnePixel** đã được chỉnh sửa, chúng tôi thiết lập số thế hệ tối đa là $max_gen = 50$, và kích thước quần thể là $s = 16$, tương đương với số lần truy vấn mô hình tối đa là 800. Do hạn chế về tài nguyên tính toán, cũng như để phù hợp với ngữ cảnh tấn công thực tế, khi mà ta có thể bị giới hạn về số lượng truy vấn mô hình, đối với tấn công lên các mô hình *object detection*, chúng tôi sẽ cho các thuật toán dừng lại ngay sau khi nhiều thành công đầu tiên được tìm thấy. Với tỉ lệ pixel bị thay đổi tối đa cho mỗi đối tượng, chúng tôi đặt $k = 0.01$. Các tham số p_c , p_m , p_0 và σ chúng tôi thiết lập tương tự cho trường hợp tấn công lên *image classification*.

4.2 Kết quả thực nghiệm

4.2.1 Tấn công các mô hình *image classification*

Các kết quả thực nghiệm tấn công lên hai mô hình AT1 và AT2 lần lượt được thể hiện ở Bảng 4.3 và Bảng 4.4.

Các phương pháp hộp trắng

Đối với các phương pháp tấn công hộp trắng, ta thấy rằng, dù là tấn công thưa, nhưng các phương pháp này lại hướng đến tối ưu hóa giá trị l_2 thay vì l_0 -norm, được thể hiện thông qua giá trị trung bình l_2 tính được của các nhiều thành công do hai phương pháp SAPF và Homotopy tạo thành là thấp hơn hẳn so với do các phương pháp hộp đen. Tuy nhiên giá trị tối ưu chính là l_0 hay số pixel bị thay đổi lại rất lớn. Nguyên nhân là do các thuật toán tấn công hộp trắng đều dựa trên thông tin gradient để thực hiện tìm kiếm nhiều, mà thông tin này lại chỉ thích hợp để tối ưu hóa các giá trị liên tục như l_2 -norm.

Kết quả đo bằng ASR của phương pháp Homotopy trên cả hai mô hình AT1 và AT2 đều thấp hơn nhiều khi so sánh với kết quả của phương pháp SAPF (0.360 so với 0.630 trên AT1 và 0.480 so với 0.990 trên mô hình AT2). Xét về thông số l_0 và l_2 , mặc dù Homotopy cho ra kết quả rất tốt trên l_2 nhưng số lượng pixel bị thay đổi, tức là l_0 lại rất cao. Mặc dù là các phương pháp tấn công hộp trắng nhưng kết quả của các phương pháp này - trên cả 3 độ đo - không hẳn lúc nào cũng vượt trội hơn kết quả của các phương pháp hộp đen, đặc biệt khi xem xét độ đo l_0 thì các phương pháp này luôn có l_0 cao hơn rất nhiều so với các phương pháp hộp đen.

Các chỉ số ASR đo được còn cho thấy rằng việc huấn luyện các mô hình để phòng thủ trước các cuộc tấn công có mang lại hiệu quả, tuy nhiên hiệu quả này chỉ được thể hiện ở một số phương pháp nhất định chứ chưa thật sự chung được nhiều phương pháp tấn công khác nhau. Cụ thể, mô hình AT1 hiệu quả trong việc chống các cuộc tấn công hộp trắng (ASR của các phương pháp hộp trắng thấp hơn mặc dù l_0 cao hơn so với phương pháp hộp đen) và điều ngược lại có thể quan sát được khi xét ASR và l_0 của các phương pháp tấn công trên mô hình AT2.

Các phương pháp hộp đen

Đối với các phương pháp tấn công hộp đen, ngoại trừ phương pháp OnePixel, các phương pháp Sparse-RS, SA-MOO và các biến thể sử dụng GA do chúng tôi đề xuất đều có tỉ lệ tấn công thành công từ khá cao đến rất cao trên cả hai mô hình, thậm chí đối với mô hình AT1, các phương pháp tấn công hộp đen này lại có ASR cao hơn hẳn so với các phương pháp hộp trắng. Trong đó phương pháp Sparse-RS cho ra tỉ lệ tấn công thành công cao nhất, nhưng lại không có khả năng tối ưu hóa các giá trị l_0 hay l_2 , thể hiện ở kết quả trung bình của những giá trị này rất cao, đặc biệt ở giá trị l_0 gần như không suy giảm gì so với số lượng pixel được phép thay đổi tối đa mà chúng tôi đặt ban đầu là 24. Trong khi đó, phương pháp SA-MOO tuy có một sự giảm nhẹ về ASR so với Sparse-RS nhưng đã cho thấy khả năng tối ưu đáng kể ở cả hai giá trị l_0 và l_2 nhờ kĩ thuật tối ưu hóa đồng thời cả hàm mất mát và hai giá trị này thông qua tối ưu hóa đa mục tiêu.

Về các biến thể sử dụng GA do chúng tôi đề xuất, mặc dù kết quả ASR chưa thể so sánh được với hai phương pháp tấn công thưa hộp đen thưa state-of-the-art, nhưng cũng đã đạt được con số tương đối cao (từ 71% đến 75% tỉ lệ thành công

trên mô hình AT1 và từ 61% đến 69% tỉ lệ thành công trên mô hình AT2, thấp hơn khoảng 10% so với các phương pháp hộp đen state-of-the-art). Chúng tôi cho rằng những kết quả này là rất đáng ghi nhận, khi mà các biến thể GA của chúng tôi sử dụng hàm măt măt 3.3, hạn chế hơn nữa lượng thông tin có thể sử dụng để tìm kiếm nhiều (chỉ sử dụng xác suất cao nhất trong các lớp được trả về, thay vì cần đến xác suất của tất cả các lớp như hàm măt măt 3.2 mà các phương pháp Sparse-RS và SA-MOO sử dụng), việc hạn chế hơn về thông tin được biết này là để góp phần giúp cho phương pháp tấn công tiến gần hơn đến ngữ cảnh "hộp đen" trong thực tế, làm tăng khả năng phương pháp có thể đưa vào tấn công vào trong thực tiễn hơn. Dù kết quả ASR không quá ấn tượng như hai phương pháp Sparse-RS hay SA-MOO, các phương pháp GA của chúng tôi lại rất hiệu quả trong việc tạo ra những nhiễu khó phân biệt được, biểu hiện ở giá trị l_0 trung bình trên các nhiễu thành công được tối ưu cực kì mạnh mẽ, thấp hơn nhiều so với các phương pháp tấn công hộp đen thừa còn lại (giảm từ 3 đến 9 pixel bị thay đổi so với các phương pháp khác). Dù vậy, vì tập trung vào làm giảm số lượng pixel bị thay đổi, nên các giá trị màu của pixel bị thay đổi cũng phải tăng lên để đảm bảo được tỉ lệ tấn công thành công, do đó, giá trị l_2 ở các nhiễu thành công do các biến thể GA tạo ra lại tương đối cao, nhưng vẫn thấp hơn so với Sparse-RS. Chúng tôi đã thực hiện thêm các thực nghiệm bổ sung trên các biến thể GA, trong đó ở giai đoạn đánh giá (được định nghĩa ở 1), thay vì so sánh hàm măt măt và l_0 của nhiễu, chúng tôi so sánh hàm măt măt và giá trị l_2 để quyết định cá thể tốt hơn; các kết quả thực nghiệm bổ sung này được ghi lại ở bảng 4.5 và 4.6. Kết quả thực nghiệm cho thấy rằng, khi thay thế như vậy sẽ không ảnh hưởng quá nhiều đến ASR, tuy nhiên lại có sự thay đổi trong giá trị l_0 và l_2 trung bình của các nhiễu thành công. Nay giờ, giá trị l_2 của nhiễu thành công được tối ưu hơn, thậm chí tốt hơn cả phương pháp SA-MOO, nhưng đồng thời, giá trị l_0 hay số pixel bị thay đổi cũng vẫn được giảm và cũng gần ngang bằng với kết quả do SA-MOO tạo thành.

Về sự ảnh hưởng của các kích thước giải đấu khác nhau, chúng tôi quan sát được rằng, đối với tấn công lên các mô hình *image classification* thì biến thể sử dụng kích thước giải đấu lớn ($T = 4$) sẽ có ASR cao hơn. Điều này có thể giải thích là đối với tấn công lên các mô hình *image classification*, không gian tìm kiếm của các nhiễu là tương đối nhỏ (chỉ làm thay đổi tối đa 24 pixel), nên việc tập trung khai thác các cá thể tốt nhất mà sử dụng $T = 4$ mang lại sẽ giúp thuật toán nhanh chóng xác định

và khai thác các vùng tiềm năng trên không gian tìm kiếm, nơi có tỉ lệ xuất hiện các nhiễu thành công cao.

Về sự ảnh hưởng của sử dụng archive, chúng tôi nhận thấy rằng khi tấn công lên các mô hình *image classification* thì sử dụng archive sẽ gây ra sự suy giảm nhẹ về ASR so với không sử dụng. Điều này cũng có thể là do kích thước nhỏ của không gian tìm kiếm, khiến cho việc sử dụng archive làm tăng khả năng các nhiễu không thành công đóng góp vào các thế hệ tiếp theo sẽ dễ gây sự nhiễu loạn trong quá trình xác định các vùng tốt có giải pháp thành công, hoặc thậm chí còn định hướng quá trình tìm kiếm ra xa khỏi các vùng tiềm năng này.

4.2.2 Tấn công các mô hình object detection

Bảng 4.7 ghi lại các kết quả ASR và Queries của các phương pháp tấn công lên các mô hình *object detection* khác nhau, với điều kiện tấn công thành công được xét theo 2.8 là phải thay đổi nhãn của các đối tượng đã được dự đoán; bảng 4.8 ghi lại các kết quả ASR của các phương pháp tấn công với điều kiện thành công được xét theo 2.7 là chỉ cần khiến cho mô hình thất bại trong việc phát hiện đúng đối tượng (bao gồm cả thay đổi nhãn và khiến cho mô hình không phát hiện được đối tượng); bên cạnh đó, Bảng 4.9 cho thấy giá trị mAP@0.50 bị suy giảm đối với những dự đoán của các mô hình trên các bức hình bị thêm nhiễu.

Chúng tôi quan sát được rằng, phương pháp OnePixel được điều chỉnh và các biến thể GA do chúng tôi đề xuất đều có khả năng tạo ra các nhiễu có thể đánh lừa các mô hình *object detection*, trong đó các biến thể GA đều có tỉ lệ tấn công thành công cao hơn hẳn so với phương pháp OnePixel (cao hơn khoảng 10% ở các thực nghiệm với điều kiện phải thay đổi nhãn dự đoán, và cao hơn từ 20% đến 30% với điều kiện khiến cho mô hình thất bại trong phát hiện đúng đối tượng). Khi xét đến lượng truy vấn mô hình cần thiết để tìm được nhiều tấn công thành công (Queries), phương pháp OnePixel cần sử dụng ít số lần truy vấn hơn, dù vậy số lần truy vấn mô hình cần thiết cho các biến thể GA cũng không quá nhiều (chưa dùng đến 20% số lượng truy vấn tối đa được phép sử dụng).

Về sự ảnh hưởng của các kích thước giải đấu khác nhau cũng như việc sử dụng archive, chúng tôi quan sát được các hiện tượng trái ngược hoàn toàn so với khi tấn công lên các mô hình *image classification*. Đối với tấn công lên các mô hình *object*

detection, sử dụng kích thước giải đấu nhỏ ($T = 2$), cùng với sử dụng archive lại có kết quả ASR tốt hơn so với kích thước giải đấu lớn ($T = 4$) và không sử dụng archive, trong đó biến thể **GAA** ($T = 2$) kết hợp sử dụng $T = 2$ cùng với archive cho ra kết quả ASR cao nhất trên phần lớn các thực nghiệm. Nguyên nhân cho hiện tượng này cũng là do tính chất trái ngược ở kích thước không gian tìm kiếm của hai bài toán, khi mà ở bài toán tấn công lên mô hình *object detection*, số lượng pixel tối đa được thay đổi phụ thuộc vào kích thước bức ảnh đầu vào, số lượng đối tượng có trong bức ảnh đó và kích thước của các đối tượng trong bức ảnh, những yếu tố này khiến cho kích thước không gian tìm kiếm tăng lên gấp nhiều lần so với tấn công lên các mô hình *image classification*. Khi kích thước không gian tìm kiếm tăng cao và trở nên phức tạp hơn, việc tập trung khai thác các cá thể tốt khi sử dụng $T = 4$ sẽ dễ khiến cho thuật toán tìm kiếm mắc kẹt ở những vùng cực trị địa phương mà bỏ sót nhiều vùng tiềm năng khác trên không gian tìm kiếm, từ đó làm giảm khả năng tìm thấy các nhiều thành công. Ngược lại, sử dụng $T = 2$ làm tăng khả năng khám phá của GA, giúp thuật toán tìm được nhiều vùng tiềm năng, dẫn đến tăng tỉ lệ tìm thấy được các cá thể tấn công thành công. Điều tương tự cũng xảy ra khi sử dụng archive, việc tăng khả năng các cá thể không tốt được đóng góp vào thế hệ tiếp theo sẽ làm tăng tính đa dạng của quần thể, từ đó nâng cao khả năng khám phá ra những cá thể tốt hơn thông qua các phép lai ghép và đột biến. Tuy nhiên, việc nâng cao khả năng khám phá của GA như trên có thể sẽ khiến độ đa dạng của quần thể trở nên quá cao, dẫn đến thuật toán sẽ cần nhiều lần đánh giá hơn để chọn ra được một cá thể tốt nhất, điều này đã được chứng minh ở các kết quả thực nghiệm, các biến thể sử dụng $T = 2$ và archive thường cần nhiều Queries hơn để tìm được nhiều thành công.

Trong các mô hình bị tấn công, chúng tôi nhận thấy các mô hình YOLO, đặc biệt là YOLOv10 có khả năng chống lại các cuộc tấn công đối kháng làm thay đổi nhãn của đối tượng khá tốt, khi mà kết quả ASR của các phương pháp tấn công lên YOLOv10 đều dưới 45%. Bên cạnh đó, xét về khả năng dự đoán ổn định, chúng tôi cho rằng mô hình YOLOv11 giữ khả năng phát hiện đối tượng ổn định nhất dù bị tấn công bởi các phương pháp tấn công thua, biểu hiện ở sự suy giảm giá trị mAP@0.50 không quá lớn trên các bức ảnh bị thêm nhiễu so với các bức ảnh gốc. Đối với hai mô hình sử dụng kiến trúc transformer, qua các kết quả thực nghiệm, chúng tôi thấy rằng khả năng chống lại các cuộc tấn công đối kháng của chúng chưa

được tốt. Tuy tỉ lệ tấn công thay đổi nhãn thành công lên mô hình DINO rất thấp ở tất cả các phương pháp (từ 40% trở xuống), nhưng các phương pháp tấn công lại dễ dàng khiến cho mô hình này không phát hiện được các đối tượng (ví dụ có thể thấy ở các cặp Hình 4.2c- 4.2d và 4.2e- 4.2f), biểu hiện ở tỉ lệ thành công với điều kiện 2.7 là rất cao và sự suy giảm giá trị mAP@0.50 ở các kết quả dự đoán của DINO đối với những bức hình bị thêm nhiễu so với bức hình gốc là cực kì nghiêm trọng (từ 0.714 giảm xuống 0.516 khi tấn công bằng OnePixel và giảm xuống từ 0.270 đến 0.280 khi tấn công bằng các biến thể GA). Trong khi đó, với mô hình DETR, các phương pháp tấn công, đặc biệt là các phương pháp do chúng tôi đề xuất đều đạt được tỉ lệ thành công cực kỳ cao cho dù là với điều kiện khó hơn là phải thay đổi nhãn của đối tượng, hay với điều kiện dễ hơn là chỉ cần khiến mô hình không phát hiện đúng đối tượng.

BẢNG 4.3: ASR và giá trị l_p -norms ($p = 0, 2$) trung bình của các phương pháp tấn công lên mô hình AT1.

Phương pháp	Mô hình AT1		
	ASR	l_0	l_2
SAPF	0.630	61.444	2.443
Homotopy	0.360	285.666	0.724
OnePixel	0.467 (± 0.022)	23.563 (± 0.087)	3.604 (± 0.050)
Sparse-RS	0.880 (± 0.005)	23.992 (± 0.009)	4.929 (± 0.021)
SA-MOO	0.852 (± 0.016)	19.496 (± 0.239)	2.744 (± 0.042)
GAA ($T = 4$)	0.723 (± 0.017)	15.587 (± 0.245)	3.612 (± 0.051)
GAA ($T = 2$)	0.737 (± 0.017)	16.545 (± 0.226)	3.717 (± 0.026)
GA ($T = 4$)	0.751 (± 0.017)	13.851 (± 0.341)	3.390 (± 0.055)
GA ($T = 2$)	0.712 (± 0.022)	15.741 (± 0.269)	3.578 (± 0.055)

BẢNG 4.4: ASR và giá trị l_p -norms ($p = 0, 2$) trung bình của các phương pháp tấn công lên mô hình AT2.

Phương pháp	Mô hình AT2		
	ASR	l_0	l_2
SAPF	0.990	35.586	2.693
Homotopy	0.480	402.500	0.967
OnePixel	0.404 (± 0.016)	23.512 (± 0.109)	3.616 (± 0.042)
Sparse-RS	0.819 (± 0.019)	23.994 (± 0.006)	5.009 (± 0.016)
SA-MOO	0.767 (± 0.015)	19.773 (± 0.113)	2.885 (± 0.037)
GAA ($T = 4$)	0.633 (± 0.023)	15.907 (± 0.302)	3.706 (± 0.057)
GAA ($T = 2$)	0.621 (± 0.019)	16.711 (± 0.185)	3.787 (± 0.050)
GA ($T = 4$)	0.694 (± 0.019)	14.587 (± 0.200)	3.561 (± 0.050)
GA ($T = 2$)	0.615 (± 0.022)	16.032 (± 0.178)	3.662 (± 0.039)

BẢNG 4.5: ASR và giá trị l_p -norms ($p = 0, 2$) trung bình của các biến thể GA lên AT1. Với mục tiêu tối ưu là hàm mất mát và giá trị l_2 thay vì l_0 .

Phương pháp	Mô hình AT1		
	ASR	l_0	l_2
GAA ($T = 4$)	0.749 (± 0.016)	20.431 (± 0.144)	2.613 (± 0.044)
GAA ($T = 2$)	0.726 (± 0.017)	21.480 (± 0.127)	2.874 (± 0.037)
GA ($T = 4$)	0.756 (± 0.016)	20.417 (± 0.165)	2.702 (± 0.053)
GA ($T = 2$)	0.715 (± 0.013)	21.549 (± 0.185)	2.955 (± 0.025)

BẢNG 4.6: ASR và giá trị l_p -norms ($p = 0, 2$) trung bình của các biến thể GA lên AT2. Với mục tiêu tối ưu là hàm mất mát và giá trị l_2 thay vì l_0 .

Phương pháp	Mô hình AT2		
	ASR	l_0	l_2
GAA ($T = 4$)	0.675 (± 0.017)	20.670 (± 0.157)	2.764 (± 0.037)
GAA ($T = 2$)	0.597 (± 0.010)	21.474 (± 0.188)	2.910 (± 0.037)
GA ($T = 4$)	0.694 (± 0.028)	20.624 (± 0.196)	2.872 (± 0.079)
GA ($T = 2$)	0.615 (± 0.024)	21.679 (± 0.258)	3.059 (± 0.075)

BẢNG 4.7: Tỉ lệ tấn công thành công (ASR) và số lần gọi mô hình trung bình (Queries) của các phương pháp tấn công lên các mô hình *object detection*, với **tấn công thành công được xét theo điều kiện 2.8.**

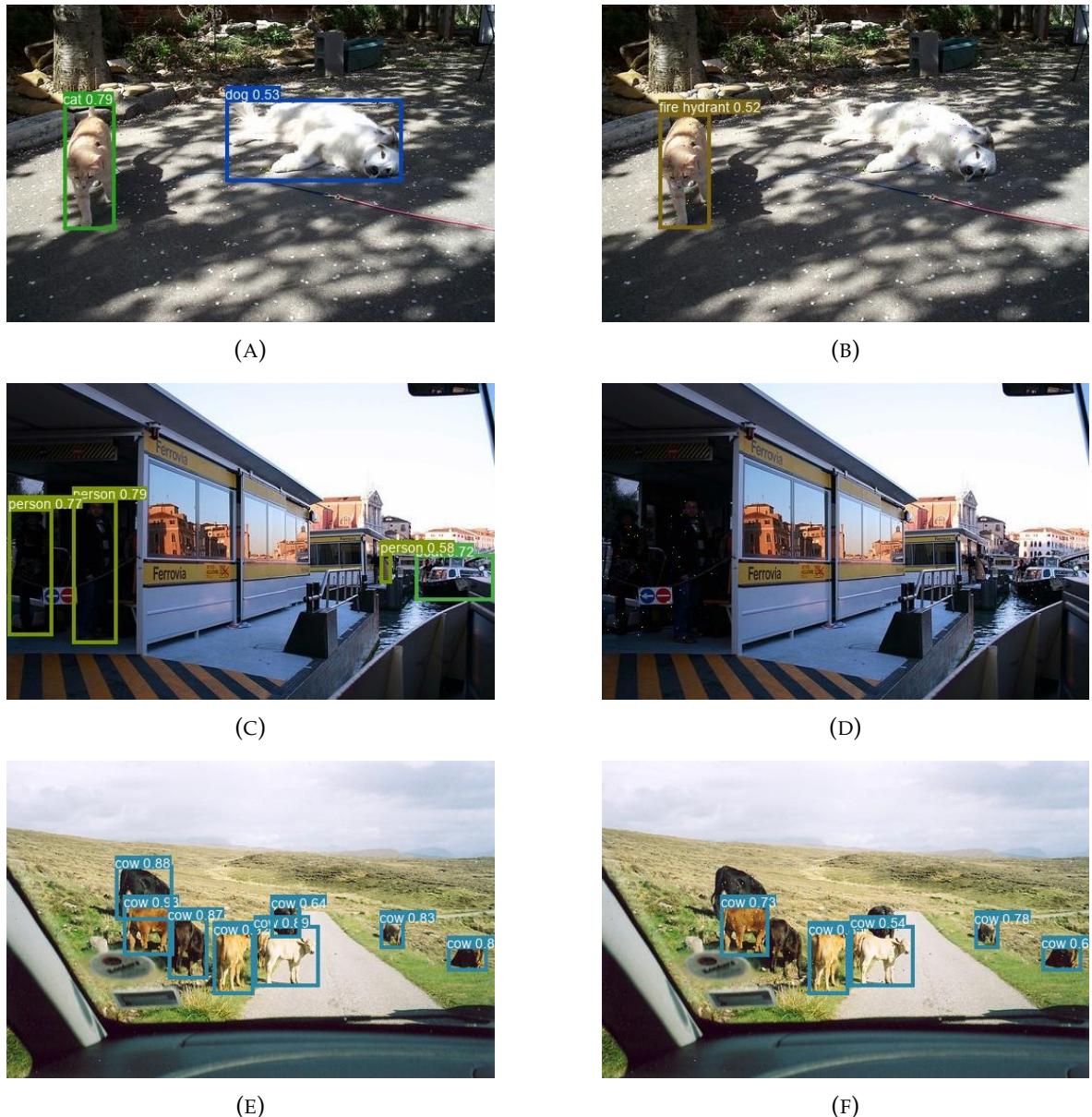
Mô hình	OnePixel		GAA ($T = 4$)		GAA ($T = 2$)		GA ($T = 4$)		GA ($T = 2$)	
	ASR	Queries	ASR	Queries	ASR	Queries	ASR	Queries	ASR	Queries
YOLOv8	49.18%	70.93	59.02%	89.0	59.02%	86.2	57.99%	87.2	58.51%	101.7
YOLOv9	47.39%	56.8	58.15%	97.3	57.34%	114.2	54.12%	77.9	55.93%	101.5
YOLOv10	33.74%	85.0	42.27%	92.6	44.74%	134.8	42.06%	103.1	43.09%	121.4
YOLOv11	45.49%	65.8	54.23%	95.8	56.05%	112.3	54.03%	80.2	53.23%	97.3
DINO	28.78%	93.7	37.27%	96.7	38.92%	130.6	37.68%	108.4	40.99%	123.9
DETR	71.14%	77.8	77.15%	65.6	78.76%	73.0	76.95%	59.0	78.36%	67.2

BẢNG 4.8: Tỉ lệ tấn công thành công (ASR) của các phương pháp tấn công lên các mô hình *object detection*, với **tấn công thành công được xét theo điều kiện 2.7.**

Mô hình	OnePixel	GAA ($T = 4$)	GAA ($T = 2$)	GA ($T = 4$)	GA ($T = 2$)
YOLOv8	57.82%	79.85%	79.87%	79.19%	79.53%
YOLOv9	52.53%	80.12%	80.37%	78.51%	79.71%
YOLOv10	40.59%	71.74%	72.79%	73.17%	72.69%
YOLOv11	49.09%	74.23%	74.80%	75.26%	72.45%
DINO	43.87%	82.90%	86.39%	84.61%	85.53%
DETR	86.72%	93.71%	93.78%	93.04%	93.14%

BẢNG 4.9: Các kết quả dự đoán (mAP@0.50) của các mô hình nàn nhau lên những bức ảnh gốc và những bức ảnh được tạo ra bởi các phương pháp tấn công khác nhau.

Mô hình	Ảnh gốc	OnePixel	GAA ($T = 4$)	GAA ($T = 2$)	GA ($T = 4$)	GA ($T = 2$)
YOLOv8	0.726	0.576	0.458	0.468	0.451	0.487
YOLOv9	0.731	0.562	0.461	0.459	0.467	0.468
YOLOv10	0.716	0.585	0.432	0.433	0.441	0.450
YOLOv11	0.759	0.610	0.517	0.509	0.512	0.522
DINO	0.714	0.516	0.276	0.271	0.278	0.270
DETR	0.798	0.408	0.272	0.275	0.280	0.282



HÌNH 4.2: Kết quả dự đoán của mô hình DINO trên các bức ảnh chưa bị thêm nhiễu (cột trái) và trên các bức ảnh đã bị thêm nhiễu (cột phải). Chỉ có bức ảnh ở hàng đầu tiên (tính từ trên xuống) được xem là tấn công thành công. (Tất cả các bức ảnh gốc được lấy từ tập kiểm tra của bộ dữ liệu PascalVoc2007 [6])

Chương 5

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 Kết luận

Trong khóa luận này, chúng tôi đã đề xuất một phương pháp tấn công thưa hộp đen sử dụng thuật giải di truyền và hai hàm măt măt cho hai mục tiêu tấn công là các mô hình *image classification* và *object detection*. Các hàm măt măt được thiết kế để đảm bảo hoàn toàn tính chất "black-box" của hình thức tấn công, chỉ sử dụng thông tin về đầu vào và kết quả trả về của mô hình. Dù phương pháp chúng tôi đề xuất vẫn chưa hoàn toàn đạt đến hiệu suất ấn tượng như các công trình state-of-the-art đã được nghiên cứu trước đó, nhưng cũng đã đạt được kết quả ở mức tốt và cho thấy tiềm năng của ứng dụng thuật giải di truyền nói riêng và các thuật toán tiến hóa nói chung trong việc áp dụng vào bài toán này. Hơn nữa, hai hàm măt măt mà chúng tôi thiết kế đã có thể đưa phương pháp tấn công tiến gần hơn đến khả năng gây ra một mối đe dọa thật sự trong thực tế. Điều này cho thấy rằng, cho dù các mô hình thị giác máy tính vẫn không ngừng được nghiên cứu phát triển hơn từng ngày, vẫn tiềm ẩn nguy cơ suy giảm về độ an toàn và tin cậy của những dự đoán. Vì vậy, nghiên cứu để sớm tìm ra một phương pháp bảo vệ hiệu quả các mô hình thị giác máy tính nói riêng và các mô hình mạng học sâu nói chung là một công việc đáng để lưu tâm, đặc biệt là trong bối cảnh ngày nay, khi mà các mô hình học sâu đang ngày càng đóng vai trò quan trọng trong cuộc sống của chúng ta.

5.2 Hướng phát triển

Sau khi xem xét và phân tích các kết quả thực nghiệm, chúng tôi nhận thấy những hướng phát triển tiềm năng như sau:

- Nghiên cứu cách áp dụng tối ưu hóa đa mục tiêu vào quá trình tấn công, để có thể tìm được nhiều vừa thành công vừa thay đổi ít pixel nhất có thể một cách đồng thời.
- Nghiên cứu cách sử dụng một mô hình thay thế (surrogate model), kết hợp sử dụng mô hình thay thế và thuật toán tiến hóa để tận dụng điểm mạnh của cả hai phương pháp (sự đơn giản, cần ít tài nguyên máy của thuật toán tiến hóa và khả năng giảm thiểu đáng kể các lượt truy vấn mô hình của phương pháp sử dụng mô hình thay thế)

Bên cạnh đó, nghiên cứu các cách thức bảo vệ mô hình trước những phương pháp tấn công đối kháng khác nhau cũng là một hướng nghiên cứu cần được chú ý.

DANH MỤC CÔNG BỐ KHOA HỌC

Hội nghị

Chi Cuong Le, Tri Phan and Ngoc Hoang Luong. "Gradient-Free Sparse Adversarial Attack on Object Detection Models". (in preparation)

PHỤ LỤC

1. Nghiên cứu phân tích thành phần

Chúng tôi tiến hành nghiên cứu phân tích tính quan trọng của hai thành phần là **lai ghép** (crossover) và **đột biến** (mutation) của phương pháp đề xuất bằng cách loại bỏ từng thành phần này trên biến thể **GA** ($T = 2$). Sau đó, chúng tôi ghi lại kết quả tấn công của biến thể này lên hai mô hình image classification AT1 và AT2 sau khi đã bỏ đi mỗi thành phần.

Kết quả tấn công được ghi lại ở hai Bảng 1 và 2. Chúng tôi thấy rằng, hiệu suất tấn công của mô hình bị giảm đáng kể nếu bước mutation bị loại bỏ, trong khi nếu không có bước crossover, hiệu suất tấn công dường như không bị ảnh hưởng. Điều này cho thấy rằng việc thêm một vài nhiễu loạn nhỏ vào các cá thể đã có trong quần thể (được thực hiện trong bước mutation) có vai trò chủ chốt giúp thuật toán khám phá không gian tìm kiếm một cách hiệu quả và tìm được các nhiễu tấn công thành công. Trong khi đó, việc truyền lại các đặc tính tốt của cha mẹ cho cá thể con (được thực hiện trong bước crossover) không làm tăng tính đa dạng của quần thể nên không giúp thuật toán mở rộng không gian tìm kiếm và không ảnh hưởng lớn đến hiệu suất của mô hình.

2. Sự ảnh hưởng của số lượng pixel bị thay đổi

Bảng 3 và 4 ghi lại kết quả tấn công của biến thể **GA** ($T = 2$) lên hai mô hình AT1 và AT2 với các số lượng pixel tối đa được phép thay đổi khác nhau. Kết quả cho thấy rằng hiệu suất tấn công được tăng lên khi tăng số lượng pixel tối đa được phép thay đổi. Ngoài ra, chúng tôi còn quan sát được rằng, phương pháp đề xuất có khả năng tối ưu số lượng pixel bị thay đổi rất hiệu quả, thể hiện ở giá trị l_0 trung bình được giảm đi đáng kể so với giới hạn được phép thay đổi.

BẢNG 1: ASR và giá trị l_p -norms ($p = 0, 2$) trung bình của các phương pháp tấn công lên mô hình AT1. (*_w/o_crossover là phương pháp * sau khi loại bỏ bước crossover; *_w/o_mutation là phương pháp * sau khi loại bỏ bước mutation)

Phương pháp	Mô hình AT1		
	ASR	l_0	l_2
GA ($T = 2$)	0.712 (± 0.022)	15.741 (± 0.269)	3.578 (± 0.055)
GA ($T = 2$)_w/o_crossover	0.716 (± 0.017)	15.927 (± 0.222)	3.576 (± 0.043)
GA ($T = 2$)_w/o_mutation	0.472 (± 0.026)	21.357 (± 0.181)	3.999 (± 0.037)

BẢNG 2: ASR và giá trị l_p -norms ($p = 0, 2$) trung bình của các phương pháp tấn công lên mô hình AT2. (*_w/o_crossover là phương pháp * sau khi loại bỏ bước crossover; *_w/o_mutation là phương pháp * sau khi loại bỏ bước mutation)

Phương pháp	Mô hình AT2		
	ASR	l_0	l_2
GA ($T = 2$)	0.615 (± 0.022)	16.032 (± 0.178)	3.662 (± 0.039)
GA ($T = 2$)_w/o_crossover	0.619 (± 0.026)	16.168 (± 0.320)	3.665 (± 0.053)
GA ($T = 2$)_w/o_mutation	0.382 (± 0.0128)	21.436 (± 0.262)	4.023 (± 0.056)

BẢNG 3: ASR và giá trị l_p -norms ($p = 0, 2$) trung bình của phương pháp tấn công **GA** ($T = 2$) lên mô hình AT1 với các số lượng pixel tối đa được thay đổi khác nhau.

Số lượng pixel tối đa được thay đổi	Mô hình AT1		
	ASR	l_0	l_2
12	0.572 (± 0.017)	6.685 (± 0.201)	2.404 (± 0.033)
24	0.712 (± 0.022)	15.741 (± 0.269)	3.578 (± 0.055)
36	0.796 (± 0.017)	25.431 (± 0.276)	4.459 (± 0.042)
48	0.849 (± 0.016)	35.337 (± 0.295)	5.179 (± 0.035)

BẢNG 4: ASR và giá trị l_p -norms ($p = 0, 2$) trung bình của phương pháp tấn công **GA** ($T = 2$) lên mô hình AT2 với các số lượng pixel tối đa được thay đổi khác nhau.

Số lượng pixel tối đa được thay đổi	Mô hình AT2		
	ASR	l_0	l_2
12	0.493 (± 0.017)	7.002 (± 0.193)	2.545 (± 0.060)
24	0.615 (± 0.022)	16.032 (± 0.178)	3.662 (± 0.039)
36	0.704 (± 0.014)	26.267 (± 0.282)	4.573 (± 0.029)
48	0.751 (± 0.016)	36.332 (± 0.301)	5.302 (± 0.036)

TÀI LIỆU THAM KHẢO

- [1] Zikui Cai, Yaoteng Tan, and M. Salman Asif. "Ensemble-based Blackbox Attacks on Dense Prediction". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*. IEEE, 2023, pp. 4045–4055. DOI: 10.1109/CVPR52729.2023.00394. URL: <https://doi.org/10.1109/CVPR52729.2023.00394>.
- [2] Nicolas Carion et al. "End-to-End Object Detection with Transformers". In: *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part I*. Ed. by Andrea Vedaldi et al. Vol. 12346. Lecture Notes in Computer Science. Springer, 2020, pp. 213–229. DOI: 10.1007/978-3-030-58452-8_13. URL: https://doi.org/10.1007/978-3-030-58452-8_13.
- [3] Francesco Croce et al. "RobustBench: a standardized adversarial robustness benchmark". In: *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*. Ed. by Joaquin Vanschoren and Sai-Kit Yeung. 2021. URL: <https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/a3c65c2974270fd093ee8a9bf8ae7d0b-Abstract-round2.html>.
- [4] Francesco Croce et al. "Sparse-RS: A Versatile Framework for Query-Efficient Sparse Black-Box Adversarial Attacks". In: *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelfth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*. AAAI Press, 2022, pp. 6437–6445. DOI: 10.1609/aaai.v36i6.20595. URL: <https://doi.org/10.1609/aaai.v36i6.20595>.

- [5] Alexey Dosovitskiy et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL: <https://openreview.net/forum?id=YicbFdNTTy>.
- [6] Mark Everingham et al. "The Pascal Visual Object Classes (VOC) Challenge". In: *Int. J. Comput. Vis.* 88.2 (2010), pp. 303–338. DOI: 10.1007/S11263-009-0275-4. URL: <https://doi.org/10.1007/s11263-009-0275-4>.
- [7] Yanbo Fan et al. "Sparse Adversarial Attack via Perturbation Factorization". In: *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXII*. Ed. by Andrea Vedaldi et al. Vol. 12367. Lecture Notes in Computer Science. Springer, 2020, pp. 35–50. DOI: 10.1007/978-3-030-58542-6_3. URL: https://doi.org/10.1007/978-3-030-58542-6_3.
- [8] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and Harnessing Adversarial Examples". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: <http://arxiv.org/abs/1412.6572>.
- [9] Sven Gowal et al. "Improving Robustness using Generated Data". In: *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*. Ed. by Marc'Aurelio Ranzato et al. 2021, pp. 4218–4233. URL: <https://proceedings.neurips.cc/paper/2021/hash/21ca6d0cf2f25c4dbb35d8dc0b679c3f-Abstract.html>.
- [10] Sven Gowal et al. "Uncovering the Limits of Adversarial Training against Norm-Bounded Adversarial Examples". In: *CoRR abs/2010.03593* (2020). arXiv: 2010.03593. URL: <https://arxiv.org/abs/2010.03593>.
- [11] John H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.
- [12] Glenn Jocher and Jing Qiu. *Ultralytics YOLO11*. Version 11.0.0. 2024. URL: <https://github.com/ultralytics/ultralytics>.

- [13] Alex Krizhevsky. "Learning Multiple Layers of Features from Tiny Images". In: 2009. URL: <https://api.semanticscholar.org/CorpusID:18268744>.
- [14] Hoang N. Luong and Peter A. N. Bosman. "Elitist Archiving for Multi-Objective Evolutionary Algorithms: To Adapt or Not to Adapt". In: *Parallel Problem Solving from Nature - PPSN XII - 12th International Conference, Taormina, Italy, September 1-5, 2012, Proceedings, Part II*. Ed. by Carlos A. Coello Coello et al. Vol. 7492. Lecture Notes in Computer Science. Springer, 2012, pp. 72–81. DOI: 10.1007/978-3-642-32964-7_8. URL: https://doi.org/10.1007/978-3-642-32964-7_8.
- [15] Aleksander Madry et al. "Towards Deep Learning Models Resistant to Adversarial Attacks". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL: <https://openreview.net/forum?id=rJzIBfZAb>.
- [16] Satya Mallick. *Mean Average Precision (mAP) – Object Detection Model Evaluation Metric*. <https://learnopencv.com/mean-average-precision-map-object-detection-model-evaluation-metric/>. Accessed: 2024-11-30. 2021.
- [17] Yunxiao Qin et al. "Training Meta-Surrogate Model for Transferable Adversarial Attack". In: *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*. Ed. by Brian Williams, Yiling Chen, and Jennifer Neville. AAAI Press, 2023, pp. 9516–9524. DOI: 10.1609/AAAI.V37I8.26139. URL: <https://doi.org/10.1609/aaai.v37i8.26139>.
- [18] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. "One Pixel Attack for Fooling Deep Neural Networks". In: *IEEE Trans. Evol. Comput.* 23.5 (2019), pp. 828–841. DOI: 10.1109/TEVC.2019.2890858. URL: <https://doi.org/10.1109/TEVC.2019.2890858>.

- [19] Rejin Varghese and Sambath M. "YOLOv8: A Novel Object Detection Algorithm with Enhanced Performance and Robustness". In: *2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS)*. 2024, pp. 1–6. DOI: 10.1109/ADICS58448.2024.10533619.
- [20] Ao Wang et al. "YOLOv10: Real-Time End-to-End Object Detection". In: *CoRR* abs/2405.14458 (2024). DOI: 10.48550/ARXIV.2405.14458. arXiv: 2405.14458. URL: <https://doi.org/10.48550/arXiv.2405.14458>.
- [21] Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. "YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information". In: *CoRR* abs/2402.13616 (2024). DOI: 10.48550/ARXIV.2402.13616. arXiv: 2402.13616. URL: <https://doi.org/10.48550/arXiv.2402.13616>.
- [22] Phoenix Neale Williams and Ke Li. "Black-Box Sparse Adversarial Attack via Multi-Objective Optimisation CVPR Proceedings". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*. IEEE, 2023, pp. 12291–12301. DOI: 10.1109/CVPR52729.2023.01183. URL: <https://doi.org/10.1109/CVPR52729.2023.01183>.
- [23] Cihang Xie et al. "Adversarial Examples for Semantic Segmentation and Object Detection". In: *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. IEEE Computer Society, 2017, pp. 1378–1387. DOI: 10.1109/ICCV.2017.153. URL: <https://doi.org/10.1109/ICCV.2017.153>.
- [24] Hao Zhang et al. "DINO: DETR with Improved DeNoising Anchor Boxes for End-to-End Object Detection". In: *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL: <https://openreview.net/forum?id=3mRwyG5one>.
- [25] Mingkang Zhu, Tianlong Chen, and Zhangyang Wang. "Sparse and Imperceptible Adversarial Attack via a Homotopy Algorithm". In: *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 12868–12877. URL: <http://proceedings.mlr.press/v139/zhu21a.html>.