ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

NGUYỄN ĐỨC ANH

CÁC PHƯƠNG PHÁP ĐẢM BẢO TÍNH CHẮC CHẮN CHO MỘT SỐ MÔ HÌNH HỌC SÂU

LUẬN ÁN TIẾN SĨ KỸ THUẬT PHẦN MỀM

Hà Nội - 2023

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

NGUYỄN ĐỨC ANH

CÁC PHƯƠNG PHÁP ĐẢM BẢO TÍNH CHẮC CHẮN CHO MỘT SỐ MÔ HÌNH HỌC SÂU

Chuyên ngành: Kỹ thuật phần mềm

Mã số: 9480103.01

LUẬN ÁN TIẾN SĨ KỸ THUẬT PHẦN MỀM

NGƯỜI HƯỚNG DẪN KHOA HỌC:

PGS.TS. Phạm Ngọc Hùng

GS.TS. Nguyễn Lê Minh

Hà Nội - 2023

Mục lục

Danh sách hình vẽ

Danh sách bảng

Thuật ngữ và từ viết tắt

Từ viết tắt	**Từ tiếng Anh**	**Ý nghĩa/Tạm dịch**
ATN	Adversarial Transformation Networks	Mạng biến đổi đối kháng
AST	Abstract Syntax Tree	Cây cú pháp trừu tượng
API	Application Programming Interface	Giao diện lập trình ứng dụng
BIM	Basic Iterative Method	Phương pháp lặp lại cơ bản
CNN	Convolutional Neural Network	Mô hình tích chập
COI	Coefficient Input	Đầu vào hệ số
DNN	Deep Neural Network	Mô hình học sâu
EAD	Elastic-Net Attacks to Deep Neural Networks	Tấn công lưới đàn hồi cho mô hình học sâu
FGSM	Fast Gradient Sign Method	Phương pháp dấu đạo hàm nhanh
FFNN	Feed-Forward Neural Network	Mô hình nơ-ron truyền thẳng
HA4FNN	Heuristic-based Attack for Feed-forward Neural Network	Tấn công phỏng đoán cho mô hình nơ-ron truyền thẳng
JSMA	Jacobian-based Saliency Map Attack	Tấn công bản đồ nhô ra dựa theo Jacobian
L-BFGS	Limited-memory Broyden–Fletcher–Goldfarb–Shanno	Broyden–Fletcher–Goldfarb–Shanno có bộ nhớ hữu hạn
SCADefender	Stacked Convolutional Autoencoder-based Defender	Phòng thủ cho mô hình học sâu
MC/DC	Modified Condition/Decision Coverage	Độ phủ điều kiện con (độ phủ C3)

Từ viết tắt	**Từ tiếng Anh**	**Ý nghĩa/Tạm dịch**
MI-FGSM	Momentum Iterative Fast Gradient Sign Method	Phương pháp dấu đạo hàm nhanh lặp lại có động lượng
PatternAttack	Pattern-based Attack for Convolutional Neural Network	Tấn công dựa theo mẫu thêm nhiễu cho mô hình tích chập
PSNR	Peak Signal-to-Noise Ratio	Tỷ lệ tín hiệu trên tạp âm cực đại
QI4AE	Quality Improvement for Adversarial Examples	Cải thiện chất lượng cho ảnh đối kháng
SSIM	Structural Similarity Index Measure	Độ đo chỉ số tương đồng về cấu trúc
SMT	Satisfiability Modulo Theories	Lý thuyết Modulo thỏa mãn

Giải thích kí hiệu

Kí hiệu	Mô tả
$\mathbf{A}$	Mô hình mã hóa tự động
CE	Hàm cross-entropy
$\mathbf{M}$	Mô hình kiểm thử
$\mathbf{M}_i(\mathbf{x})$	Xác suất dự đoán của nhãn thứ i
MIN	Hàm lấy giá trị nhỏ nhất
MAX	Hàm lấy giá trị lớn nhất
SIGN	$\in \{-1, 0, 1\}$, là hàm trả về dấu
b_i	Độ chênh lệch của tầng L_i
c	Số nhãn của mô hình phân lớp
d	Số đặc trưng hay số điểm ảnh
f_i	Điểm ảnh trừu tượng thứ i
η	Tốc độ học
n_i^j	Nơ-ron thứ j của tầng thứ i
θ_i	Hàm kích hoạt của tầng L_i
L_i	Tầng thứ i của mô hình học sâu
h	Số tầng của mô hình học sâu
$\mathbf{x}'$	Ảnh đối kháng
x'_i	Điểm ảnh thứ i của ảnh đối kháng
$\mathbf{x}$	Ảnh đầu vào của mô hình kiểm thử
x_i	Điểm ảnh thứ i của ảnh dự đoán đúng
$\mathbf{x}_{out}$	Ảnh đầu ra của mô hình mã hóa tự động
$\mathbf{y}_{true}$	Véc-tơ xác suất đúng của ảnh
y_{true}	Nhãn đúng của ảnh
y^*	Nhãn đích (sử dụng trong tấn công đối kháng có định hướng)
$w_{i,j,k}$	Trọng số giữa nơ-ron n_i^j và nơ-ron n_{i+1}^k
$\boldsymbol{\zeta}$	Véc-tơ nhiễu
$\mathbb{1}$	Hàm chỉ thị

Lời cam đoan

Tôi xin cam đoan đây là công trình nghiên cứu do tôi thực hiện dưới sự hướng dẫn của PGS. TS. Phạm Ngọc Hùng tại Bộ môn Công nghệ Phần mềm, Khoa Công nghệ Thông tin, Trường Đại học Công nghệ, Đại học Quốc gia Hà Nội và GS. TS. Nguyễn Lê Minh tại Viện Khoa học và Công nghệ Tiên tiến Nhật Bản (JAIST). Các số liệu và kết quả trình bày trong luận án là trung thực, chưa được công bố bởi bất kỳ tác giả nào hay ở bất kỳ công trình nào khác.

Tác giả

Nguyễn Đức Anh

Lời cảm ơn

Trước tiên tôi xin gửi lời cảm ơn chân thành và sâu sắc đến thầy giáo, PGS. TS. Phạm Ngọc Hùng và GS. Nguyễn Lê Minh - người đã hướng dẫn, khuyến khích, truyền cảm hứng, chỉ bảo và tạo cho tôi những điều kiện tốt nhất từ khi bắt đầu làm nghiên cứu sinh đến khi hoàn thành luận án này.

Tôi xin chân thành cám ơn Quỹ Đổi mới sáng tạo Vingroup (VINIF) đã hỗ trợ tôi thông qua chương trình học bổng đào tạo thạc sĩ, tiến sĩ trong nước, mã số VINIF.2021.TS.105 và VINIF.2022.TS001.

Tôi xin chân thành cảm ơn các thầy cô giáo khoa Công nghệ thông tin, Trường Đại học Công nghệ, Đại học Quốc gia Hà Nội, đặc biệt là các Thầy Cô trong Bộ môn Công nghệ Phần mềm đã tận tình đào tạo, cung cấp cho tôi những kiến thức vô cùng quý giá, đã tạo điều kiện tốt nhất cho tôi về môi trường làm việc trong suốt quá trình học tập và nghiên cứu.

Tôi xin trân trọng cảm ơn Phòng Đào tạo và Ban Giám hiệu Trường Đại học Công nghệ đã tạo điều kiện thuận lợi cho tôi trong suốt quá trình thực hiện luận án.

Tôi xin gửi lời cảm ơn đến tất cả đến các thành viên trong nhóm nghiên cứu tại Phòng thí nghiệm đảm bảo chất lượng phần mềm, đặc biệt là em Đỗ Minh Khá và em Nguyễn Như Ngọc đã đồng hành cùng tôi trên chặng đường này.

Cuối cùng, tôi xin bày tỏ lòng biết ơn vô hạn đối với cha, mẹ, và em gái đã luôn ủng hộ và yêu thương tôi một cách vô điều kiện. Nếu không có sự ủng hộ của gia đình tôi không thể hoàn thành được luận án này.

Tóm tắt

Mô hình học sâu được sử dụng phổ biến trong bài toán phân loại ảnh. Để đảm bảo chất lượng của mô hình học sâu, nhiều độ đo đã được đề xuất như độ chuẩn xác, độ chính xác và điểm số F1. Tuy nhiên, dù mô hình được kiểm thử kĩ càng bởi các độ đo này, nhiều nghiên cứu gần đây cho thấy mô hình có thể dễ dàng bị tấn công đối kháng. Tính chắc chắn của mô hình học sâu là khả năng mô hình nhận diện được chính xác nhãn của ảnh đầu vào khi ảnh này được thêm nhiễu đối kháng. Kẻ tấn công có thể thêm nhiễu đối kháng vào ảnh dự đoán đúng để ảnh bị nhận diện sai, thậm chí chỉ cần thêm nhiễu đối kháng vào duy nhất một điểm ảnh. Do đó, cải thiện tính chắc chắn được coi là một trong những giải pháp quan trọng để nâng cao chất lượng của mô hình học sâu. Cụ thể, luận án đã đạt được bốn kết quả chính như sau.

Thứ nhất, luận án đề xuất phương pháp HA4FNN để cải thiện tỉ lệ thành công và hiệu năng thấp của DeepCheck khi kiểm thử mô hình nơ-ron truyền thẳng. Phương pháp HA4FNN sử dụng bộ giải phỏng đoán thay vì bộ giải SMT và loại bỏ việc duy trì trạng thái kích hoạt nơ-ron. Từ mô hình kiểm thử, HA4FNN chuyển mô hình này sang mã nguồn C, sau đó biên dịch và thực thi mã nguồn này với đầu vào là ảnh dự đoán đúng để lấy đường thi hành. Sau đó, thực thi tượng trưng chuyển đường thi hành thành hệ ràng buộc và dùng bộ giải phỏng đoán để tìm nghiệm. Nghiệm này tương ứng với ảnh đối kháng và có thể có trạng thái kích hoạt nơ-ron khác với ảnh dự đoán đúng. Thực nghiệm trên MNIST, Fashion-MNIST và bộ chữ cái viết tay cho thấy phương pháp HA4FNN có hiệu năng và tỉ lệ thành công vượt trội so với DeepCheck. Một công cụ đã được cài đặt để chứng minh hiệu quả của phương pháp HA4FNN.

Thứ hai, luận án đề xuất phương pháp PatternAttack để cải thiện tính đa dạng và chất lượng ảnh đối kháng sinh bởi ATN. Tư tưởng chính của Patter-

nAttack là xây dựng ATN khái quát có kiến trúc mô hình mã hóa tự động để thêm nhiễu đối kháng vào ảnh đầu vào theo các mẫu thêm nhiễu khác nhau, từ đó làm tăng tính đa dạng của ảnh đối kháng. Từ ảnh đối kháng sinh ra, PatternAttack sử dụng thuật toán tham lam để loại bỏ nhiễu dư thừa, từ đó tăng chất lượng ảnh đối kháng. Thực nghiệm trên MNIST và CIFAR-10 cho thấy ATN khái quát có thể tấn công mô hình học sâu với tỉ lệ thành công cao và thuật toán tham lam có khả năng cải thiện chất lượng ảnh đối kháng với tỉ lệ giảm nhiễu tốt. Một công cụ đã được cài đặt để chứng minh hiệu quả của PatternAttack.

Thứ ba, luận án đề xuất phương pháp QI4AE để nâng cao chất lượng ảnh đối kháng sinh bởi các phương pháp tấn công đối kháng. Độ đo chất lượng ảnh đối kháng là L_0 và L_2. Phương pháp QI4AE được cải tiến từ thuật toán tham lam đề xuất trong PatternAttack. Ý tưởng chính của QI4AE là kết hợp thuật toán tham lam với mô hình mã hóa tự động. Ảnh đối kháng được đẩy qua mô hình mã hóa tự động để lấy ảnh đối kháng cải thiện mức thô, rồi đẩy tiếp qua thuật toán tham lam để lấy ảnh đối kháng cải thiện mức tinh chế. Thực nghiệm trên MNIST và CIFAR-10 cho thấy phương pháp QI4AE có thể cải thiện chất lượng ảnh đối kháng đáng kể với chi phí tính toán thấp. Một công cụ đã được cài đặt để chứng minh hiệu quả của phương pháp QI4AE.

Cuối cùng, để nâng cao tính chắc chắn của mô hình học sâu, luận án đề xuất phương pháp SCADefender để loại bỏ nhiễu đối kháng khỏi ảnh đối kháng. Một phần dữ liệu học của SCADefender là tập ảnh đối kháng sinh bởi nhiều phương pháp tấn công đối kháng khác nhau. Kết quả của quá trình học là một mô hình mã hóa tự động phòng thủ có khả năng loại bỏ nhiễu đối kháng khỏi ảnh đối kháng. Thực nghiệm trên MNIST, CIFAR-10 và Fashion-MNIST cho thấy SCADefender có thể loại bỏ nhiễu đối kháng khỏi ảnh đối kháng khá tốt. Một công cụ đã được cài đặt để chứng minh hiệu quả của phương pháp SCADefender.

Các nghiên cứu được trình bày trong luận án không những có ý nghĩa về mặt lý thuyết mà còn góp phần làm phương pháp kiểm thử tính chắc chắn cho mô hình học sâu dễ dàng được áp dụng hơn trong thực tiễn. Điều này đặc biệt có ý nghĩa với những mô hình học sâu có yêu cầu cao về khả năng chống lại tấn công từ bên ngoài, trong đó có tấn công đối kháng. Ngoài ra, các công cụ của luận án đã được triển khai sử dụng tại TSDV và nhận được phản hồi tích cực.

Chương 1

Giới thiệu

1.1. Đặt vấn đề

Sự ra đời của học sâu được coi là một cuộc cách mạng trong thế kỉ 21 [1]. Tư tưởng cho sự ra đời học sâu là mô phỏng quá trình học của não người từ dữ liệu. Kết quả mô phỏng này được biểu diễn bởi mô hình học sâu. Mô hình học sâu có nhiều tầng gồm tầng đầu vào, các tầng ẩn và tầng đầu ra [1, 2, 3, 4]. Nhiều mô hình học sâu đã được đề xuất như mô hình nơ-ron truyền thẳng, mô hình tích chập, v.v. Nhiều nghiên cứu đã cho thấy học sâu đạt được kết quả tương đương hoặc tốt hơn con người trong nhiều bài toán như phân loại ảnh [5], nhận dạng đối tượng [6], nhận dạng khuôn mặt [7], xử lý ngôn ngữ tự nhiên [8], xe tự lái [9, 10], phát hiện mã độc [11, 12] và chăm sóc sức khỏe [13, 14], v.v.

Trong bài toán phân loại ảnh, với đầu vào là tập học gồm các ảnh và nhãn tương ứng, lập trình viên sẽ định nghĩa kiến trúc mô hình học sâu, rồi chọn các siêu tham số phù hợp như tốc độ học, số lần lặp, v.v. để xây dựng mô hình. Để đánh giá chất lượng mô hình học sâu, các độ đo được sử dụng phổ biến gồm độ chính xác, độ chuẩn xác và điểm số F1 [15]. Tuy nhiên, dù mô hình học sâu phân loại ảnh đạt được kết quả tốt với các độ đo nêu trên, mô hình học sâu vẫn có thể có tính chắc chắn chưa đủ tốt [16, 17, 18, 19, 20, 21]. Tính chắc chắn của mô hình học sâu là khả năng mô hình nhận diện được chính xác nhãn của ảnh đầu vào khi ảnh này được thêm nhiễu đối kháng. Khái niệm tính chắc chắn được mô tả trong nhiều nghiên cứu và trình bày trong Định nghĩa 1.

Định nghĩa 1. [Tính chắc chắn] Mô hình học sâu $\mathbf{M}$ có tính chắc chắn với ảnh dự đoán đúng $\mathbf{x}$ và ngưỡng khoảng cách L_p kí hiệu là δ khi và chỉ khi với mọi ảnh đối kháng $\mathbf{x}'$ mà $L_p(\mathbf{x}, \mathbf{x}') \leq \delta$, $\arg\max(\mathbf{M}(\mathbf{x})) = \arg\max(\mathbf{M}(\mathbf{x}'))$ [22, 23].

Quá trình kẻ tấn công cố tình thêm nhiễu đối kháng vào ảnh đã dự đoán đúng để đánh lừa mô hình gọi là tấn công đối kháng. Ảnh trước khi thêm nhiễu đối kháng và được dự đoán đúng bởi mô hình học sâu gọi là ảnh dự đoán đúng. Ảnh sau khi thêm nhiễu đối kháng gọi là ảnh đối kháng. Trong đó, nhiễu đối kháng được tính dựa trên những điểm ảnh khác nhau giữa ảnh dự đoán đúng và ảnh đối kháng. Khái niệm nhiễu đối kháng được mô tả ở Định nghĩa 2.

Định nghĩa 2. [Nhiễu đối kháng] Cho ảnh dự đoán đúng $\mathbf{x}$ và mô hình kiểm thử $\mathbf{M}$, véc tơ nhiễu $\boldsymbol{\zeta} = [\zeta_0, \zeta_1, ..., \zeta_{d-1}]^T \in [0, 1]^d$ được gọi là nhiễu đối kháng khi và chỉ khi $\mathbf{x} + \boldsymbol{\zeta}$ được dự đoán sai bởi $\mathbf{M}$ [16, 17].

Để đánh giá được tính chắc chắn của mô hình học sâu, có hai hướng nghiên cứu chính gồm chứng minh tính chắc chắn của mô hình học sâu và sinh ảnh đối kháng. Đối với hướng chứng minh tính chắc chắn, ba hướng nghiên cứu con sử dụng phổ biến gồm sử dụng bộ giải SMT-Solver như [24, 25], sử dụng kĩ thuật làm mịn mức trừu tượng như [26, 27, 28] và kĩ thuật giải thích mức trừu tượng như [29, 30, 31]. Nhược điểm ba hướng này là không hỗ trợ tốt cho mô hình học sâu phức tạp [22]. Đối với hướng sinh ảnh đối kháng, các phương pháp theo hướng này sinh các ảnh đối kháng và coi đó là bằng chứng thể hiện tính chắc chắn của mô hình học sâu. Ưu điểm của hướng này là dễ dàng áp dụng cho các mô hình học sâu phức tạp nên được sử dụng phổ biến. Các nghiên cứu tiêu biểu theo hướng này có thể kể đến CW [17], ATN [18], L-BFGS [32] và DeepFool [33], BIS [19], MI-FGSM [34], PGD [35], v.v.

Theo hướng sinh ảnh đối kháng, hai tiêu chí phổ biến để đánh giá chất lượng phương pháp tấn công đối kháng gồm chất lượng ảnh đối kháng và tỉ lệ thành công [36]. Công thức đánh giá chất lượng ảnh đối kháng có hai đầu vào chính gồm ảnh dự đoán đúng và ảnh đối kháng tương ứng. Các công thức phổ biến là sử dụng độ đo khoảng cách L_p, độ đo cấu trúc như SSIM [37] và các độ đo khác như PSNR [38]. Đối với tỉ lệ thành công, tiêu chí này thể hiện tỉ lệ ảnh dự đoán đúng được thêm nhiễu đối kháng thành công để sinh ảnh đối kháng. Nếu tỉ lệ thành công là 100% thì tất cả ảnh dự đoán đúng đều được thêm nhiễu đối

kháng thành công để mô hình kiểm thử nhận diện sai. Một trong những mục tiêu chính của các phương pháp tấn công đối kháng theo hướng này là sinh ảnh đối kháng với tỉ lệ thành công cao nhất có thể.

Hai hướng chính để sinh ảnh đối kháng là kiểm thử hộp đen và kiểm thử hộp trắng [2, 39, 40]. Trong kiểm thử hộp đen, kiểm thử viên giả định rằng họ không biết được kiến trúc và trọng số của mô hình kiểm thử. Kẻ tấn công chỉ có thể truy vấn mô hình kiểm thử thông qua API để lấy kết quả trả về. Kết quả trả về có thể là nhãn dự đoán hoặc véc tơ xác suất của từng nhãn. Trong kiểm thử hộp trắng, kiểm thử viên có thể truy cập kiến trúc và trọng số của mô hình kiểm thử. Chi phí của kiểm thử hộp trắng thường cao hơn hộp đen do thường phải tính toán đạo hàm hàm mục tiêu của mô hình kiểm thử. Do kiểm thử viên biết được kiến trúc mô hình nên tỉ lệ thành công của kiểm thử hộp trắng thường cao hơn so với kiểm thử hộp đen.

Trong hướng kiểm thử hộp trắng, tấn công đối kháng có hai hướng chính gồm tấn công đối kháng có định hướng và tấn công đối kháng không định hướng [5, 40]. Điểm chung của hai hướng này là thực hiện thêm nhiễu đối kháng vào ảnh dự đoán đúng để sinh ảnh đối kháng có nhãn khác nhãn của ảnh dự đoán đúng. Điểm khác biệt chính giữa hai hướng này là nhãn của ảnh đối kháng. Trong tấn công đối kháng có định hướng, nhãn của ảnh đối kháng cần giống nhãn đích, trong đó nhãn đích được định nghĩa trước khi tấn công. Ví dụ, xét ảnh số chín trong bộ dữ liệu MNIST [41], mô hình kiểm thử nhận diện chính xác nhãn của ảnh này. Kiểm thử viên chọn một nhãn bất kì khác nhãn số chín trong tập nhãn này, ví dụ nhãn số một. Sau đó, tấn công đối kháng có định hướng sẽ tìm cách thêm nhiễu đối kháng vào ảnh dự đoán đúng để sinh một ảnh đối kháng. Trong đó, mô hình kiểm thử nhận diện ảnh đối kháng này có nhãn số một. Trong tấn công đối kháng không định hướng, nhãn của ảnh đối kháng có thể là bất kì nhãn nào ngoại trừ nhãn của ảnh dự đoán đúng.

Hướng tấn công đối kháng không định hướng cho mô hình nơ-ron truyền thẳng sử dụng thực thi tượng trưng được đề xuất lần đầu tiên trong DeepCheck [42]. Tuy nhiên, thực nghiệm cho thấy phương pháp này có tỉ lệ thành công và hiệu năng chưa đủ tốt. Tư tưởng chính của DeepCheck là biến đổi mô hình nơ-ron truyền thẳng thành mã nguồn C. Sau đó, ảnh dự đoán đúng được chuyển thành đầu vào để thực thi trên mã nguồn này. Kết quả thực thi ảnh dự đoán đúng này

là một đường thi hành. Kế tiếp, kĩ thuật thực thi tượng trưng được áp dụng trên đường thi hành này để sinh hệ ràng buộc và sử dụng bộ giải SMT để giải hệ ràng buộc. Nghiệm của hệ ràng buộc tương ứng với ảnh đối kháng và phải có cùng trạng thái kích hoạt nơ-ron với ảnh dự đoán đúng. Nguyên nhân của tỉ lệ thành công và hiệu năng thấp là do DeepCheck sử dụng bộ giải SMT và yêu cầu trạng thái kích hoạt nơ-ron của ảnh đối kháng phải giống ảnh dự đoán đúng. Đối với hệ ràng buộc phức tạp, bộ giải SMT có thể tốn chi phí tính toán khá lớn để tìm nghiệm. Yêu cầu trạng thái kích hoạt nơ-ron của ảnh đối kháng phải giống ảnh dự đoán đúng sẽ làm giảm vùng không gian thêm nhiễu đối kháng vào ảnh dự đoán đúng để sinh ảnh đối kháng. Trong thực tế, một ảnh dự đoán đúng có nhiều cách thêm nhiễu đối kháng để đạt được mục đích tấn công, mà ảnh đối kháng tương ứng có thể không cùng trạng thái kích hoạt nơ-ron với ảnh dự đoán đúng.

Trong hướng tấn công đối kháng có định hướng cho mô hình học sâu, nhiều phương pháp đã đề xuất thiếu tính khái quát hóa. Tính khái quát hóa là khả năng một phương pháp có thể học được cách thêm nhiễu đối kháng vào ảnh dự đoán đúng để sinh ảnh đối kháng và áp dụng tri thức này để thêm nhiễu đối kháng vào ảnh đầu vào mới trong tương lai [18]. Các phương pháp tiêu biểu thiếu tính khái quát hóa có thể kể đến FGSM [16], CW [17], BIM [19], L-BFGS [32], DeepFool [33], MI-FGSM [34], DeepExplore [43], v.v. Cụ thể, các phương pháp này sẽ thêm nhiễu đối kháng vào ảnh dự đoán đúng một cách độc lập để sinh ảnh đối kháng. Để giải quyết vấn đề thiếu tính khái quát hóa của các phương pháp này, ATN [18] đã được đề xuất để thêm nhiễu đối kháng vào ảnh dự đoán đúng theo độ đo khoảng cách L_2. Tư tưởng của ATN là xây dựng mô hình mã hóa tự động để chuyển ảnh dự đoán đúng thành ảnh đối kháng. Sau khi xây dựng xong mô hình mã hóa tự động, ATN có thể thêm nhiễu đối kháng vào ảnh dự đoán đúng để sinh ảnh đối kháng với chi phí thấp. Tuy nhiên, ảnh đối kháng sinh bởi ATN gặp hai vấn đề gồm chất lượng ảnh đối kháng và tính đa dạng của ảnh đối kháng. Đối với vấn đề chất lượng, ảnh đối kháng sinh bởi ATN thường có nhiều nhiễu dư thừa. Nếu loại bỏ những nhiễu này khỏi ảnh đối kháng, chất lượng ảnh đối kháng theo độ đo L_2 tăng lên. Đối với vấn đề tính đa dạng của ảnh đối kháng, ATN thường thêm nhiễu đối kháng vào mọi điểm ảnh. Nếu kiểm thử viên muốn đánh giá tính chắc chắn bằng cách thêm nhiễu vào các điểm ảnh thuộc vùng biên đối tượng hoặc vùng nền thì ATN không hỗ trợ.

Sau khi đã hiểu được bản chất của các phương pháp tấn công đối kháng, nhiệm vụ kế tiếp là chống lại các cuộc tấn công như vậy. Đây là bài toán cải thiện tính chắc chắn. Các hướng cải thiện tính chắc chắn phổ biến gồm *(i)* xây dựng lại mô hình kiểm thử [16, 35, 44, 45], *(ii)* xây dựng một mô hình phân lớp để nhận diện ảnh dự đoán đúng và ảnh đối kháng [46, 47, 48] và *(iii)* loại bỏ nhiễu đối kháng khỏi ảnh đầu vào [46, 49, 50, 51]. Trong cách tiếp cận *(iii)*, ảnh đầu vào được đi qua một mô hình loại bỏ nhiễu đối kháng, ví dụ như mô hình mã hóa tự động. Ảnh sau khi loại bỏ nhiễu đối kháng sẽ được đẩy vào mô hình kiểm thử để lấy kết quả. Ưu điểm của cách tiếp cận này là mô hình mã hóa tự động loại bỏ nhiễu đối kháng có thể được xây dựng từ trước. Khi có ảnh đầu vào mới, mô hình này có thể được sử dụng để loại bỏ nhiễu đối kháng với chi phí thấp. Theo hướng này, các phương pháp kinh điển có thể kể đến MagNet [46], PuVAE [49] và Defense-VAE [50]. Tuy nhiên, ba phương pháp này chưa loại bỏ nhiễu đối kháng đủ tốt đối với ảnh đối kháng có nhiễu đối kháng đa dạng. Ví dụ, MagNet xây dựng bộ khôi phục với tập học là ảnh đối kháng có phân phối Gaussian. Vai trò của bộ khôi phục là khử nhiễu đối kháng trong ảnh đầu vào nếu có. Tuy nhiên, nếu kẻ tấn công thêm nhiễu có phân phối không phải Gaussian như dùng phương pháp CW [17], FGSM [16] hoặc ATN [18] thì bộ khôi phục này hoạt động có thể không hiệu quả.

Từ các phân tích trên, luận án hướng tới giải quyết các vấn đề sau. Vấn đề thứ nhất là nghiên cứu phương pháp cải thiện tỉ lệ thành công và hiệu năng của DeepCheck. Vấn đề thứ hai là đề xuất phương pháp cải thiện ATN để sinh ảnh đối kháng có nhiễu đối kháng đa dạng. Vấn đề thứ ba là nghiên cứu phương pháp loại bỏ nhiễu dư thừa khỏi ảnh đối kháng, hay nói cách khác khoảng cách L_0 hoặc L_2 giữ ảnh dự đoán đúng và ảnh đối kháng càng nhỏ càng tốt. Vấn đề thứ bốn là kết hợp các kết quả nghiên cứu về phương pháp tấn công đối kháng trước đó để xây dựng phương pháp cải thiện tính chắc chắn.

Đề tài nghiên cứu này có ứng dụng quan trọng trong thực tiễn. Thứ nhất, việc đánh giá tính chắc chắn của mô hình học sâu giúp người kiểm thử có thêm bằng chứng về chất lượng của mô hình khi hoạt động trong môi trường bất thường. Nguyên nhân là do kẻ tấn công cố tình khiến mô hình nhận diện sai ảnh đầu vào bằng cách thêm nhiễu cố ý. Ví dụ, đối với biển báo giao thông, kẻ tấn công có thể dán những ô vuông màu đen lên biển ở những vị trí đặc biệt [52, 53]. Mô

hình sẽ nhận diện sai biển báo bị chỉnh sửa này. Kết quả là hệ thống sử dụng mô hình có thể đưa ra phán đoán sai. Thứ hai, hiểu được bản chất các phương pháp tấn công đối kháng sẽ giúp ngăn chặn được các cuộc tấn công tương tự như vậy trong tương lai [3]. Hệ thống sử dụng mô hình nhận diện ảnh sẽ đưa ra phán đoán chính xác hơn khi kẻ tấn công cố tình sửa ảnh đầu vào.

1.2. Phạm vi nghiên cứu

Thứ nhất, luận án tập trung vào đánh giá chất lượng các mô hình học sâu phân loại ảnh có kích thước nhỏ như ảnh $28 \times 28 \times 1$ trong bộ dữ liệu MNIST [41] hoặc ảnh $28 \times 28 \times 3$ trong bộ dữ liệu CIFAR-10 [54]. Trong đó, hai loại ảnh được nghiên cứu gồm ảnh xám và ảnh màu. Ảnh xám chứa các điểm ảnh có giá trị số thực từ 0 đến 1 hoặc số nguyên từ 0 đến 255. Thứ hai, luận án tập trung vào đề xuất phương pháp sinh các ảnh đối kháng để đánh giá tính chắc chắn của mô hình học sâu. Thứ ba, luận án áp dụng phương pháp kiểm thử hộp trắng để sinh ảnh đối kháng.

1.3. Các đóng góp chính của luận án

Để giải quyết được bốn vấn đề đã trình bày, nghiên cứu được tiến hành qua hai giai đoạn chính. Trong giai đoạn một, luận án nghiên cứu về các phương pháp tấn công đối kháng mô hình học sâu để sinh ảnh đối kháng có tỉ lệ thành công cao, hiệu năng tốt và chất lượng tốt theo các độ đo phổ biến như L_0 và L_2. Kết quả giai đoạn này được trình bày trong Chương 3, Chương 4 và Chương 5. Trong giai đoạn hai, luận án nghiên cứu về phương pháp cải thiện tính chắc chắn. Kết quả giai đoạn này được trình bày trong Chương 6. Tóm tắt lại, luận án có bốn đóng góp chính.

Thứ nhất, luận án cải thiện tỉ lệ thành công và hiệu năng thấp của phương pháp DeepCheck. Luận án đề xuất phương pháp HA4FNN. Tư tưởng của phương pháp HA4FNN là sử dụng bộ giải phỏng đoán và loại bỏ việc duy trì trạng thái kích hoạt nơ-ron để sinh ảnh đối kháng. Mô hình kiểm thử là mô hình nơ-ron

truyền thẳng. Thực nghiệm trên MNIST, Fashion-MNIST và bộ chữ cái viết tay cho thấy phương pháp HA4FNN có hiệu năng và tỉ lệ thành công vượt trội so với phương pháp DeepCheck.

Thứ hai, luận án cải thiện phương pháp ATN để sinh ảnh đối kháng có nhiễu đối kháng đa dạng cho mô hình học sâu bằng cách sử dụng mẫu thêm nhiễu. Ngoài ra, luận án đề xuất thuật toán tham lam để cải thiện chất lượng ảnh đối kháng theo độ đo L_0 và L_2. Hai kết quả này được cài đặt trong phương pháp PatternAttack. Thực nghiệm trên MNIST và CIFAR-10 cho thấy phương pháp PatternAttack có thể tấn công mô hình học sâu với tỉ lệ thành công cao và cải thiện chất lượng ảnh đối kháng với tỉ lệ giảm nhiễu tốt.

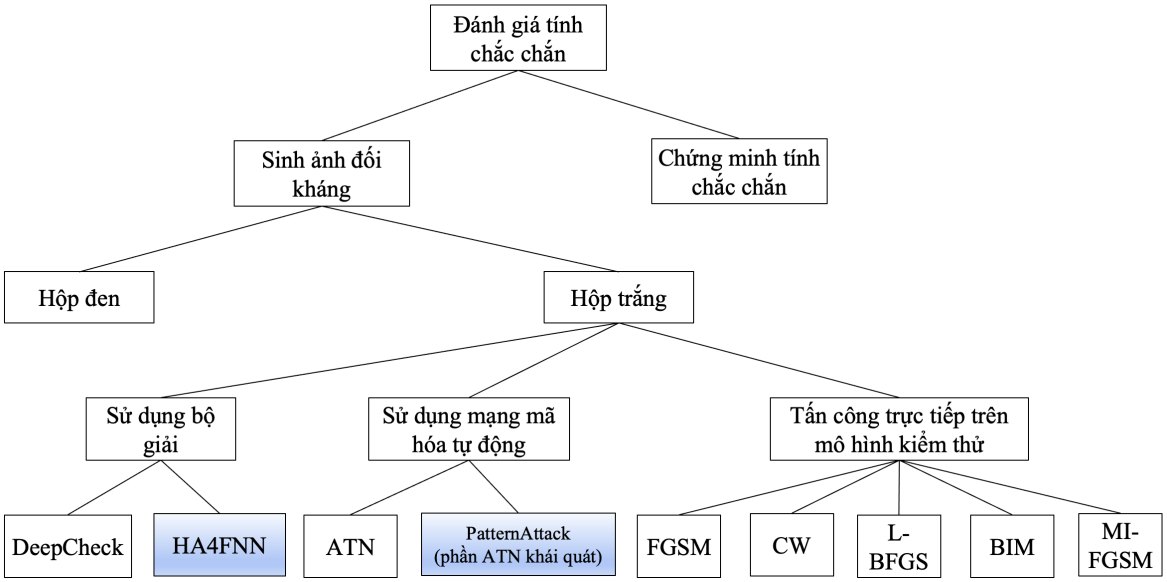
Thứ ba, luận án kết hợp thuật toán tham lam và sử dụng mô hình mã hóa tự động để nâng cao hiệu năng của quá trình cải thiện chất lượng ảnh đối kháng, gọi là QI4AE. Đề xuất này là cải tiến của thuật toán tham lam trình bày trong phương pháp PatternAttack. Thực nghiệm trên MNIST và CIFAR-10 cho thấy phương pháp QI4AE có hiệu năng tốt hơn thuật toán tham lam.

Thứ bốn, luận án đề xuất phương pháp cải thiện tính chắc chắn, gọi là SCADefender, để loại bỏ nhiễu đối kháng khỏi ảnh đầu vào. Trong khi ba đề xuất trên liên quan đến tấn công đối kháng, phương pháp SCADefender hướng đến chống lại các phương pháp tấn công đối kháng. Thực nghiệm trên MNIST, CIFAR-10 và Fashion-MNIST cho thấy phương pháp SCADefender có thể loại bỏ nhiễu đối kháng khỏi ảnh đầu vào khá tốt.

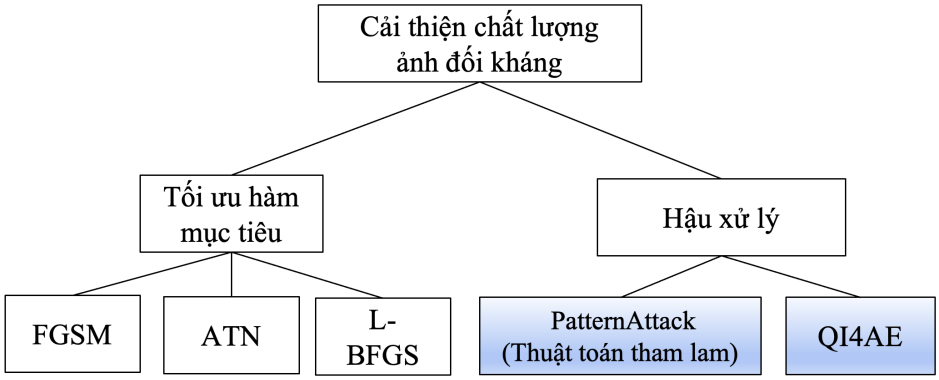
1.4. Cây nghiên cứu

Để có một cái nhìn rõ hơn về mối tương quan giữa phương pháp đề xuất và các phương pháp so sánh, phần này trình bày cây nghiên cứu liên quan. Để giảm độ phức tạp, những phương pháp thuộc các hướng khác và các hướng nghiên cứu khác sẽ bị lược bỏ. Các thông tin này sẽ được trình bày chi tiết hơn trong các nghiên cứu liên quan ở các chương đề xuất tương ứng. Hình 1.1 trình bày cây nghiên cứu của các phương pháp tấn công đối kháng. Hai phương pháp đề xuất là HA4FNN và PatternAttack (phần ATN khái quát). Hình 1.2 trình bày

cây nghiên cứu của các phương pháp cải thiện chất lượng ảnh đối kháng. Hai phương pháp đề xuất là PatternAttack (phần thuật toán tham lam) và QI4AE. Hình 1.2 trình bày cây nghiên cứu của các phương pháp cải thiện tính chắc chắn. Phương pháp đề xuất là SCADefender.



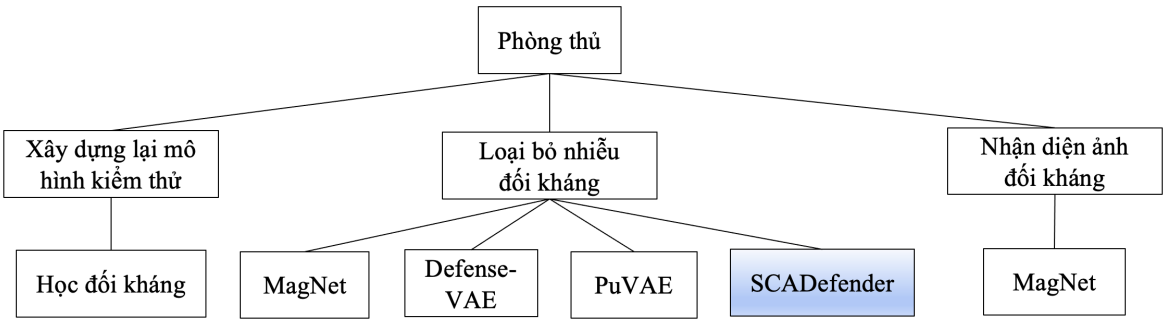
Hình 1.1: Cây nghiên cứu của các phương pháp tấn công đối kháng liên quan đến luận văn.



Hình 1.2: Cây nghiên cứu của các phương pháp cải thiện chất lượng ảnh đối kháng liên quan đến luận văn.

1.5. Mối quan hệ giữa các chương

Bố cục luận án gồm bảy chương. Chương đầu tiên trình bày về bối cảnh, các khái niệm cơ bản, các vấn đề cần giải quyết của các phương pháp đã có và các



Hình 1.3: Cây nghiên cứu của các phương pháp cải thiện tính chắc chắn (hay các phương pháp phòng thủ) liên quan đến luận văn.

đóng góp chính của luận án. Chương 2 trình bày kiến thức nền tảng như khái niệm mô hình học sâu, các phương pháp tấn công đối kháng, các phương pháp cải thiện tính chắc chắn, các tiêu chí để đánh giá chất lượng tấn công đối kháng, chất lượng cải thiện tính chắc chắn và bộ giải SMT.

Chương 3 trình bày phương pháp HA4FNN để tấn công đối kháng không định hướng mô hình nơ-ron truyền thẳng sử dụng bộ giải phỏng đoán đề xuất. Thực nghiệm cho thấy HA4FNN có thể thêm nhiễu đối kháng vào số điểm ảnh trên ảnh dự đoán đúng khá nhỏ, thậm chí là một điểm ảnh. Tuy nhiên, HA4FNN chưa hỗ trợ các kiến trúc mô hình học sâu khác, đặc biệt là mô hình tích chập. Nguyên nhân là do quá trình chuyển đổi mô hình thành mã nguồn và thực thi tượng trưng vô cùng phức tạp. Mô hình có kiến trúc càng phức tạp thì chi phí sinh mã nguồn càng lớn.

Vì thế, Chương 4 trình bày PatternAttack có hướng tiếp cận khác để kiểm thử tính chắc chắn của mô hình học sâu nói chung và mô hình tích chập nói riêng. Tư tưởng của PatternAttack có hai bước chính gồm *(i)* sử dụng mô hình mã hóa tự động để sinh ảnh đối kháng dựa trên mẫu thêm nhiễu và *(ii)* sử dụng thuật toán tham lam để cải thiện chất lượng ảnh đối kháng. Tại bước một, sử dụng mẫu thêm nhiễu sẽ quyết định những điểm ảnh nào được thêm nhiễu đối kháng. Tuy nhiên, tại bước hai, thuật toán tham lam không phù hợp để cải thiện ảnh đối kháng trong thời gian thực do tốn nhiều chi phí, đặc biệt khi nhiều điểm ảnh bị thêm nhiễu đối kháng.

Do đó, để giải quyết hạn chế của Chương 4, Chương 5 trình bày phương pháp QI4AE để cải thiện chất lượng ảnh đối kháng sử dụng mô hình mã hóa tự động

kết hợp với thuật toán tham lam. Mô hình mã hóa tự động học cách để nhận diện điểm ảnh có khả năng chứa nhiễu dư thừa, từ đó những nhiễu này có thể bị loại bỏ trong khi ảnh vẫn bị nhận diện sai. Sau đó, thuật toán tham lam được áp dụng với những ảnh được cải thiện bởi mô hình mã hóa tự động.

Ba chương trên tập trung vào tấn công đối kháng mô hình học sâu để sinh ảnh đối kháng có chất lượng tốt mà chưa quan tâm đến cải thiện tính chắc chắn. Vì thế, luận án đề xuất phương pháp cải thiện tính chắc chắn SCADefender cho mô hình học sâu trong Chương 6. Tư tưởng SCADefender là xây dựng mô hình mã hóa tự động để loại bỏ nhiễu đối kháng khỏi ảnh đầu vào. Tất cả mọi ảnh đều phải đi qua mô hình mã hóa tự động này để loại bỏ nhiễu đối kháng nếu có. Ảnh chỉnh sửa được đưa vào mô hình kiểm thử để lấy kết quả dự đoán.

Kết luận được trình bày trong Chương 7. Chương này tóm tắt lại các kết quả chính của luận án. Sau đó, luận án trình bày những hạn chế còn tồn tại và đề xuất phương hướng giải quyết các hạn chế này.

Chương 2

Kiến thức nền tảng

Chương này trình bày kiến thức nền tảng về hướng nghiên cứu kiểm thử tính chắc chắn của mô hình học sâu. Đầu tiên, luận án trình bày khái niệm mô hình học sâu, sau đó trình bày hai loại mô hình học sâu phổ biến gồm mô hình nơ-ron truyền thẳng và mô hình tích chập. Kế tiếp, luận án trình bày về các mô hình mã hóa tự động và mô tả các phương pháp tấn công đối kháng. Luận án tiếp tục trình bày các phương pháp phòng thủ sử dụng mô hình mã hóa tự động. Cuối cùng, chương mô tả các bộ dữ liệu được sử dụng trong thực nghiệm.

2.1. Mô hình học sâu cho bài toán phân loại ảnh

2.1.1. Mô hình học sâu

Định nghĩa 3. [Mô hình học sâu (DNN)] DNN $\mathbf{M}$ được định nghĩa là một bộ ba $(\mathbf{H}, \mathbf{W}, \boldsymbol{\theta})$ [1, 4], trong đó $\mathbf{H} = \{L_i \ : \ i \in \{0, 1, ..., h-1\}\}$ là tập các tầng, trong đó L_0 là tầng đầu vào và L_{h-1} là tầng đầu ra, h là tổng số tầng, $\mathbf{W} \subseteq \mathbf{H} \times \mathbf{H}$ là trọng số của mô hình và $\boldsymbol{\theta} = \{\theta_0, \theta_1, ..., \theta_{h-1}\}$ là tập hàm kích hoạt trong đó θ_i thuộc về tầng L_i [1, 4].

Kí hiệu b_i là độ chênh lệch của tầng L_i. Kí hiệu n_i^j là nơ-ron thứ j của tầng L_i. Trọng số giữa nơ-ron n_i^j và nơ-ron n_{i+1}^k được kí hiệu là $w_{i,j,k} \in \mathbf{W}$. Nơ-ron n_i^j ở trạng thái kích hoạt nếu giá trị nơ-ron này trước khi áp dụng hàm kích hoạt

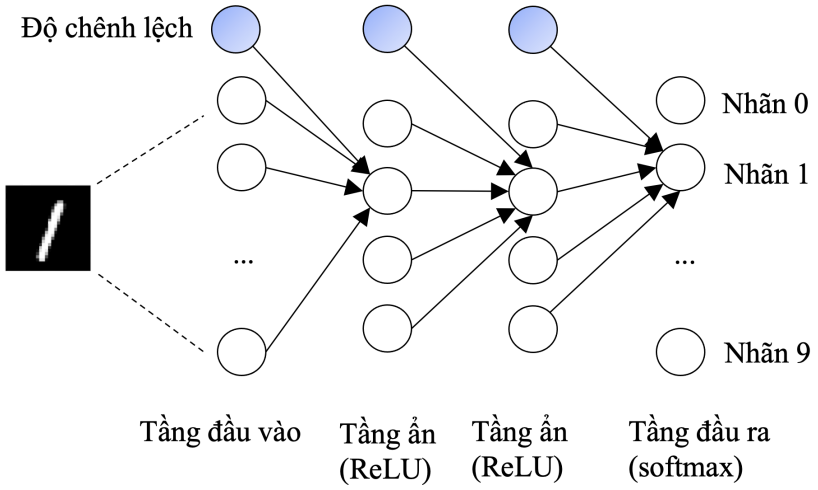
lớn hơn 0. Ngược lại, nơ-ron n_i^j ở trạng thái không kích hoạt. Mô hình $\mathbf{M}$ được học từ một tập học được gắn nhãn. Ảnh đầu vào được kí hiệu bởi véc tơ cột $\mathbf{x} = [n_0^0, n_0^1, ..., n_0^{d-1}]^T \in \mathbb{R}^d$, trong đó n_0^i là điểm ảnh thứ i và d là tổng số điểm ảnh. Nhãn đúng của ảnh được kí hiệu là y_{true}. Véc-tơ xác suất đúng của ảnh được kí hiệu là $\mathbf{y}_{true}$. Xác suất dự đoán của nhãn thứ i được kí hiệu là $\mathbf{M}_i(\mathbf{x})$. Nhãn dự đoán của ảnh $\mathbf{x}$ được tính là $\arg\max(\mathbf{M}(\mathbf{x}))$.

2.1.2. Mô hình nơ-ron truyền thẳng

Mô hình nơ-ron truyền thẳng là một loại mô hình học sâu, trong đó các tầng được nối tiếp nhau từ tầng đầu vào đến tầng đầu ra [4, 55]. Giá trị nơ-ron n_i^j sau khi áp dụng hàm kích hoạt θ_i được tính như Công thức 2.1.

$$\theta_i \left(\sum_{k=0}^{|L_{i-1}|-1} n_{i-1}^k \cdot w_{i-1,k,j} + b_i^k \right) \tag{2.1}$$

Hình 2.1 minh họa một mô hình nơ-ron truyền thẳng học trên bộ dữ liệu MNIST. Mô hình có một tầng đầu vào với 784 nơ-ron tương ứng với 784 điểm ảnh, một tầng đầu ra với mười nơ-ron tương ứng với mười nhãn và 2 tầng ẩn. Hàm kích hoạt ở các tầng ẩn là hàm ReLU [56]. Hàm kích hoạt của tầng đầu ra là hàm softmax.

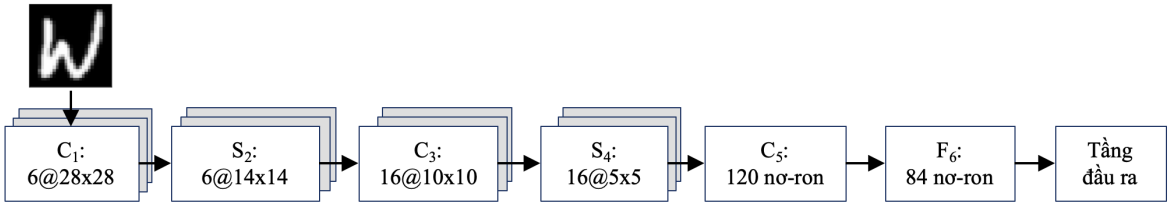


Hình 2.1: Ví dụ một phần mô hình nơ-ron truyền thẳng. Để cho dễ nhìn, một vài trọng số giữa các tầng bị ẩn đi.

2.1.3. Mô hình tích chập

Mô hình tích chập là một loại mô hình học sâu phổ biến, trong đó tầng đầu vào L_0 có kích thước $\#sample \times width \times height \times \#channel$, trong đó $\#channel \in \{1, 3\}$, $width$ là chiều rộng và $height$ là chiều cao của ảnh, $\#sample$ là số lượng ảnh đầu vào [1, 4]. Các tầng ẩn có thể là tầng tích chập, tầng giảm chiều, tầng tăng chiều, tầng kết nối thẳng, v.v.

Ví dụ, Hình 2.2 mô tả kiến trúc của LeNet-5 [57]. Mô hình tích chập này được thiết kế để nhận diện chữ viết tay. Mô hình này có bảy tầng. Tầng tích chập được kí hiệu là C_i, trong đó i là chỉ số. Tầng giảm chiều và tầng kết nối thẳng được kí hiệu D_i và F_i. Kích thước của một ảnh đầu vào là $32 \times 32 \times 1$. Tầng C_1 có sáu bản đồ đặc trưng có kích thước 28×28. Tầng S_2 có sáu bản đồ đặc trưng với kích thước 14×14. Tầng C_3 có 16 bản đồ đặc trưng với kích thước 10×10. Tầng S_4 có 16 bản đồ đặc trưng với kích thước 5×5. Tầng C_5 có 120 bản đồ đặc trưng với kích thước 1×1, sau đó được trải phẳng thành tầng F_6 với 84 nơ-ron. Tầng đầu ra có mười nơ-ron ứng với mười nhãn.



Hình 2.2: Kiến trúc LeNet-5 [57].

2.1.4. Học mô hình học sâu cho bài toán phân loại ảnh

Mô hình học sâu được học từ một tập ảnh có gắn nhãn. Các siêu tham số phổ biến được sử dụng để tùy chỉnh quá trình học gồm số lần lặp, kích thước khối, tốc độ học và thuật toán học. Số lần lặp là số lần tập học được dùng để cập nhật trọng số mô hình học sâu. Trong một lần lặp, bộ dữ liệu sẽ được chia nhỏ thành nhiều phần bằng nhau (trừ phần cuối cùng có thể có kích thước nhỏ hơn). Số ảnh trong một phần gọi là kích thước khối. Tốc độ học được dùng để điều chỉnh trọng số mô hình học sâu. Thuật toán học là thuật toán được sử dụng để cập nhật trọng số mô hình học sâu. Thuật toán SGD [58] được sử dụng

phổ biến và phát biểu như sau:

$$\mathbf{W} = \mathbf{W} - \eta \cdot \nabla_{\mathbf{W}} \mathbf{M}(\mathbf{x}) \tag{2.2}$$

trong đó η là tốc độ học, trong đó tốc độ học η có thể là một hằng số, có thể thay đổi một cách tuyến tính hoặc phi tuyến tính. Thông thường, nếu giá trị tốc độ học η quá nhỏ, ví dụ, $\eta = 0.0001$, hai vấn đề có thể phát sinh gồm *(i)* giá trị trọng số $\mathbf{W}$ có thể mắc kẹt ở giá trị tối ưu địa phương và *(ii)* quá trình học có thể phải tốn nhiều lần lặp hơn để tìm được $\mathbf{W}$ tối ưu toàn cục. Ngược lại, nếu giá trị η quá lớn, quá trình học có thể giải quyết được vấn đề mắc kẹt ở giá trị tối ưu địa phương. Tuy nhiên, bởi vì tốc độ học η quá lớn, phương pháp có thể nhảy qua giá trị $\mathbf{W}$ tối ưu toàn cục một khoảng cách xa, từ đó phải tốn nhiều lần lặp hơn để đạt đến giá trị tối ưu.

Giá trị của $\mathbf{W}$ cần thay đổi ngược hướng với dấu của đạo hàm. Cụ thể, hai trường hợp xảy ra với dấu đạo hàm của hàm mục tiêu. Thứ nhất, nếu giá trị của đạo hàm là giá trị dương, tức là giá trị hàm mục tiêu đang có xu hướng đi lên. Trong trường hợp này, để khiến hàm mục tiêu có xu hướng đi xuống, giá trị của $\mathbf{W}$ cần giảm đi một lượng nào đó. Thứ hai, nếu giá trị của đạo hàm là giá trị âm, tức là giá trị hàm mục tiêu đang có xu hướng giảm dần. Trong trường hợp này, để khiến hàm mục tiêu có xu hướng tiếp tục giảm, giá trị của $\mathbf{W}$ cần tăng lên một lượng nào đó.

Để đánh giá chất lượng mô hình học sâu, ba độ đo được sử dụng phổ biến gồm độ chuẩn xác, độ chính xác và điểm số F1. Để hiểu về các độ đo này, luận án sẽ phân tích bài toán phân lớp có hai nhãn gồm nhãn p và nhãn n. Độ chuẩn xác được trình bày trong Công thức 2.3.

$$acc = \frac{TP}{\|\mathbf{X}\|} \tag{2.3}$$

trong đó, TP là số ảnh được dự đoán đúng nhãn p bởi mô hình học sâu và $\|\mathbf{X}\|$ là kích thước của tập học. Độ chính xác được trình bày trong Công thức 2.4.

$$pre = \frac{TP}{TP + FP} \tag{2.4}$$

trong đó, FP là số ảnh được dự đoán nhãn là p nhưng sai bởi mô hình học sâu.

Điểm số F1 được tính dựa theo độ chuẩn xác và độ chính xác như trong Công thức 2.5.

$$F1 = 2 \cdot \frac{pre \cdot acc}{pre + acc} \tag{2.5}$$

2.2. Mô hình mã hóa tự động

Mô hình mã hóa tự động gồm phần mã hóa và phần giải mã [59, 60]. Các biến thể mô hình mã hóa tự động phổ biến có thể kể đến mô hình tích chập [60], mô hình giảm nhiễu [61], mô hình thưa [62], mô hình xếp chồng, mô hình biến phân [63], mô hình đối xứng [64] và mô hình Wasserstein [65]. Điểm khác biệt giữa các loại mô hình này là kiến trúc của phần mã hóa và phần giải mã, cách tính toán hàm mục tiêu và loại nhiễu thêm vào mô hình. Trong bối cảnh phân loại ảnh, đầu vào và đầu ra của mô hình mã hóa tự động được gọi là ảnh đầu vào và ảnh đầu ra.

2.2.1. Mô hình mã hóa tự động thưa

Mô hình mã hóa tự động thưa có một tầng đầu vào L_0, một tầng ẩn L_1 và một tầng đầu ra L_2. Đây là loại mô hình mã hóa tự động đơn giản nhất. Đầu vào là một ảnh $\mathbf{x} \in \mathbb{R}^{d \times 1}$. Trong phần mã hóa, mô hình mã hóa tự động này ánh xạ ảnh đầu vào trong miền không gian ẩn $\mathbf{z} \in \mathbb{R}^{z \times 1}$ trong đó $z < d$ như Công thức 2.6.

$$\mathbf{z} = \theta_1(\mathbf{W}_1^T \cdot \mathbf{x} + \mathbf{b}_1) \tag{2.6}$$

trong đó, $\mathbf{b}_1 \in \mathbb{R}^{z \times 1}$ là độ chênh lệch của tầng ẩn, $\mathbf{W}_1 \in \mathbb{R}^{d \times z}$ là trọng số giữa tầng đầu vào và tầng ẩn và θ_1 là hàm kích hoạt. Trong phần giải mã, miền không gian ẩn $\mathbf{z}$ được chuyển về ảnh đầu vào như Công thức 2.7.

$$\mathbf{x}_{out} = \theta_2(\mathbf{W}_2^T \cdot \mathbf{z} + \mathbf{b}_2) \tag{2.7}$$

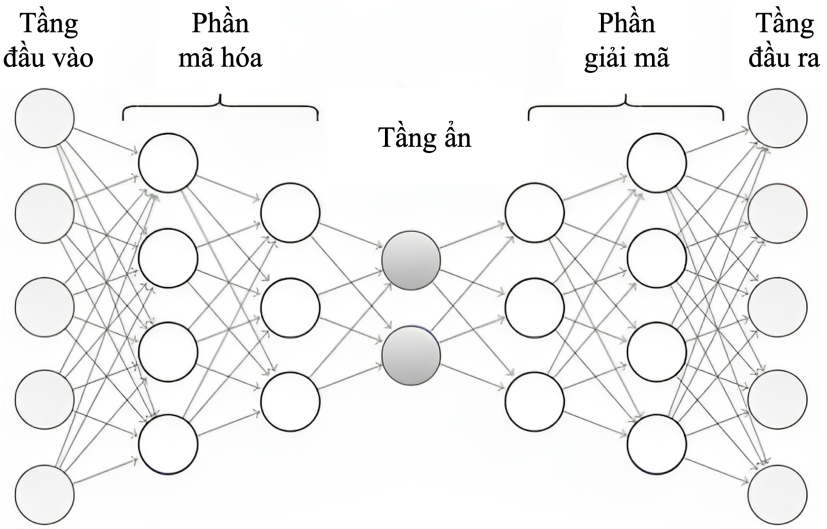
trong đó, $\mathbf{x}_{out}$ là ảnh đầu ra, θ_2 là một hàm kích hoạt, $\mathbf{b}_2 \in \mathbb{R}^{d \times 1}$ là độ chênh lệch của tầng đầu ra, và $\mathbf{W}_2 \in \mathbb{R}^{z \times d}$ là ma trận giữa tầng ẩn và tầng đầu ra. Ảnh đầu ra cần giống ảnh đầu vào. Để thỏa mãn yêu cầu này, hàm mục tiêu

của mô hình mã hóa tự động này thường sử dụng độ đo L_2 và được định nghĩa như Công thức 2.8.

$$\sum_{\mathbf{x}} L_2(\mathbf{x}, \mathbf{x}_{out})^2 \tag{2.8}$$

2.2.2. Mô hình mã hóa tự động xếp chồng

Mô hình mã hóa tự động xếp chồng là phiên bản mở rộng của mô hình mã hóa tự động thưa, trong đó có ít nhất một tầng ẩn. Trong phần mã hóa, kích thước của tầng sau luôn nhỏ hơn tầng trước. Trong phần giải mã, kích thước của tầng trước luôn nhỏ hơn tầng sau. Hình 2.3 mô tả ví dụ mô hình mã hóa tự động xếp chồng. Đầu vào có năm nơ-ron. Trong phần mã hóa, mô hình có hai tầng ẩn với kích thước bốn nơ-ron và ba nơ-ron. Miền không gian ẩn có hai nơ-ron. Trong phần giải mã có hai tầng ẩn với ba nơ-ron và bốn nơ-ron. Kích thước tầng đầu ra có kích thước như tầng đầu vào.

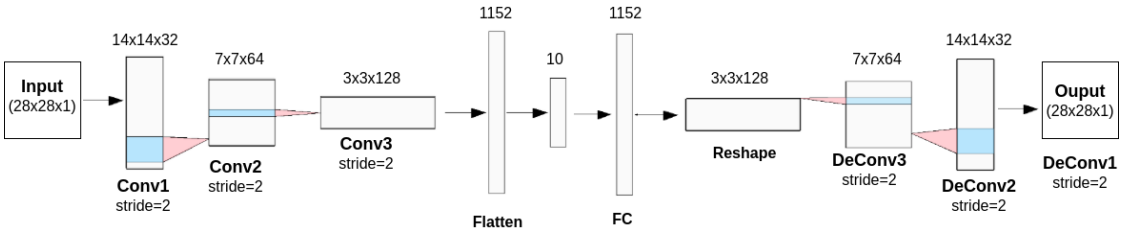


Hình 2.3: Ví dụ mô hình mã hóa tự động xếp chồng.

2.2.3. Mô hình mã hóa tự động tích chập xếp chồng

Mô hình mã hóa tự động xếp chồng không tập trung vào học cấu trúc của ảnh vì ảnh đầu vào bị xếp phẳng trong miền không gian nhiều chiều. Mỗi chiều đại diện một điểm ảnh của ảnh đầu vào. Ví dụ, ảnh $28 \times 28 \times 1$ trên MNIST sẽ

được xếp phẳng thành véc tơ 784 chiều. Bởi vì việc xếp phẳng này phá vỡ cấu trúc không gian của ảnh, mô hình mã hóa tự động xếp chồng không học được tốt đặc trưng về mặt không gian. Nếu ảnh tương tự bị xoay đi một góc vừa phải, phóng to hoặc thu nhỏ thì mô hình mã hóa tự động xếp chồng thường nhận diện nhãn ảnh chưa đủ tốt [60]. Để giảm thiểu vấn đề này, mô hình mã hóa tự động tích chập xếp chồng được đề xuất. Các tầng trong phần mã hóa có thể là tầng tích chập, tầng giảm chiều và tầng kết nối thẳng. Các tầng trong phần giải mã có thể là tầng tích chập, tầng tăng chiều và tầng kết nối thẳng. Hàm mục tiêu của mô hình mã hóa tự động tích chập xếp chồng tương tự như Công thức 2.8.



Hình 2.4: Ví dụ về một mô hình mã hóa tự động tích chập xếp chồng.

Hình 2.4 mô tả ví dụ mô hình mã hóa tự động tích chập xếp chồng. Ảnh đầu vào là ảnh đơn sắc có kích thước $28 \times 28 \times 1$. Trong phần mã hóa, ảnh đầu vào được đưa qua tầng tích chập với bước nhảy 2. Tầng Conv1 chứa 32 bản đồ đặc trưng với kích thước 14×14. Ở tầng cuối của phần mã hóa, tầng trước đó kết nối tầng không gian ẩn có kích thước 10×1. Trong phần giải mã, miền không gian ẩn được đưa vào các tầng kết nối thẳng FC, tầng tăng chiều Reshape, và các tầng tích chập gồm DeConv3, DeConv2 và DeConv1 để sinh ảnh đầu ra.

2.3. Tấn công đối kháng

2.3.1. Hai loại tấn công đối kháng phổ biến

Tấn công đối kháng là một hướng phổ biến để đánh giá tính chắc chắn của mô hình học sâu [5, 16, 17, 18, 32, 43]. Tư tưởng của tấn công đối kháng là thêm nhiễu đối kháng vào ảnh dự đoán đúng để sinh ảnh đối kháng. Hướng này có hai loại gồm tấn công đối kháng có định hướng và tấn công đối kháng không định hướng [36, 40, 66]. Cụ thể, trong tấn công đối kháng có định hướng, kẻ tấn

công xác định nhãn đích (kí hiệu y^*) và ảnh đối kháng cần được phân loại là nhãn đích bởi mô hình kiểm thử. Trong tấn công đối kháng không định hướng, ảnh đối kháng cần có nhãn khác nhãn của ảnh dự đoán đúng.

Định nghĩa 4. [Tấn công đối kháng có định hướng] Cho mô hình học sâu $\mathbf{M}$, ảnh dự đoán đúng $\mathbf{x}$ có nhãn đúng là y_{true} và nhãn đích y^* ($y^* \neq y_{true}$), tấn công đối kháng có định hướng sẽ thêm nhiễu ζ vào $\mathbf{x}$ sao cho $\arg\max(\mathbf{M}(\mathbf{x}+\zeta)) = y^*$ [17, 36, 66].

Định nghĩa 5. [Tấn công đối kháng không định hướng] Cho mô hình học sâu $\mathbf{M}$ và ảnh dự đoán đúng $\mathbf{x}$ có nhãn đúng là y_{true}, tấn công đối kháng có định hướng sẽ thêm nhiễu vào $\mathbf{x}$ sao cho $\arg\max(\mathbf{M}(\mathbf{x}+\zeta)) \neq y_{true}$ [36, 66].

2.3.2. Tính chắc chắn

Đối với hướng sinh ảnh đối kháng, tính chắc chắn được đánh giá với một phương pháp tấn công đối kháng cụ thể. Các phương pháp tấn công đối kháng khác nhau sẽ có các kĩ thuật thêm nhiễu đối kháng khác nhau. Mô hình học sâu có tính chắc chắn cao khi phương pháp tấn công đối kháng đó *(i)* khó thêm nhiễu đối kháng nhỏ vào ảnh dự đoán đúng và *(ii)* số lượng ảnh dự đoán đúng thêm nhiễu đối kháng thành công là nhỏ nhất.

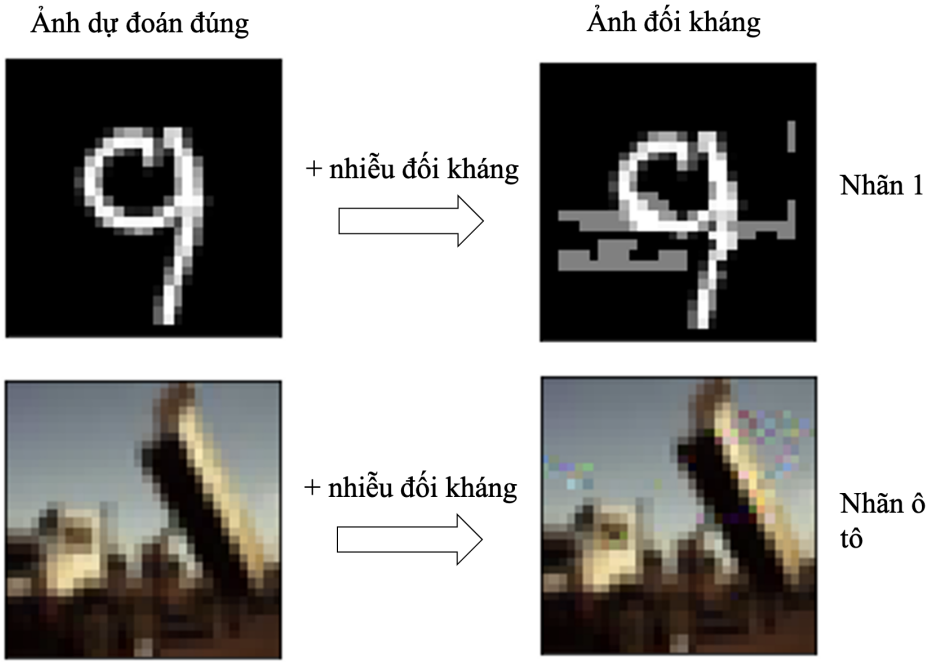
2.3.3. Phân loại ảnh

Giá trị các điểm ảnh có thể thuộc khoảng số nguyên từ 0 đến 255 hoặc số thực từ 0 đến 1. Nếu không nói gì thêm, luận án mặc định các giá trị điểm ảnh thuộc khoảng [0, 1]. Luận án phân loại ảnh thuộc các loại như sau:

- Loại ảnh đầu vào là đầu vào của mô hình kiểm thử, có thể có nhiễu hoặc không có nhiễu. Ảnh có nhiễu có thể là ảnh đối kháng nếu ảnh bị dự đoán sai nhãn bởi mô hình kiểm thử. Ngược lại, ảnh có nhiễu không được coi là ảnh đối kháng nếu ảnh này được dự đoán đúng nhãn.
- Loại ảnh dự đoán đúng là ảnh đầu vào của mô hình kiểm thử và được nhận diện chính xác nhãn.

- Loại ảnh đối kháng là ảnh nhận diện sai nhãn bởi mô hình kiểm thử và được sinh bằng cách thêm nhiễu đối kháng vào ảnh dự đoán đúng.

Ví dụ, Hình 2.5 trình bày hai ảnh lấy từ MNIST [41] và CIFAR-10 [54]. Hai ảnh này được sinh bởi một phương pháp tấn công đối kháng không định hướng. Trước khi chỉnh sửa, các ảnh được nhận diện chính xác nhãn bởi mô hình kiểm thử. Sau khi thêm nhiễu đối kháng vào một tập điểm ảnh, các ảnh đều bị nhận diện sai nhãn.



Hình 2.5: Ví dụ ảnh đối kháng sinh bởi phương pháp tấn công đối kháng không định hướng.

2.3.4. Tính chất nhiễu

Mục đích của quá trình tấn công đối kháng là tìm nhiễu đối kháng để thêm vào ảnh dự đoán đúng. Nhiễu đối kháng có hai tính chất chính gồm tính đa dạng và tính bất định. Về tính đa dạng, nhiều phương pháp tấn công đối kháng khác nhau đã được đề xuất theo hướng tấn công đối kháng không định hướng hoặc tấn công đối kháng có định hướng. Các phương pháp này thêm nhiễu đối kháng vào ảnh dự đoán đúng để sinh ảnh đối kháng theo các tiêu chí chất lượng khác nhau như L_0 [17, 42], L_2 [18, 32], L_∞ [16, 34], v.v. Trong một phương pháp, một

ảnh dự đoán đúng có nhiều cách thêm nhiễu đối kháng khác nhau để tạo ảnh đối kháng tùy theo cấu hình, hay nói cách khác, nhiễu đối kháng có tính đa dạng. Ví dụ, FGSM có thể thêm nhiễu đối kháng có cường độ 1/255 hoặc cường độ lớn hơn như 10/255 vào từng điểm ảnh của ảnh dự đoán đúng. Về tính bất định, bởi vì nhiễu đối kháng có tính đa dạng, rất khó để tìm phân phối mô tả được mọi nhiễu đối kháng thuộc nhiều phương pháp tấn công đối kháng khác nhau.

2.3.5. Đánh giá tính chắc chắn của mô hình học sâu

Tính chắc chắn của mô hình học sâu được đánh giá với một phương pháp tấn công đối kháng cụ thể. Hai độ đo phổ biến để đánh giá tính chắc chắn gồm chất lượng ảnh đối kháng và tỉ lệ thành công [36, 66]. Luận văn đề xuất độ đo tỉ lệ giảm nhiễu để đánh giá khả năng cải thiện chất lượng ảnh đối kháng.

2.3.5.1. Tiêu chí chất lượng ảnh đối kháng

Một trong những tiêu chí của tấn công đối kháng là sinh ảnh đối kháng trông giống ảnh dự đoán đúng hết mức có thể. Với tiêu chí này, độ đo khoảng cách L_p thường được sử dụng. Cụ thể, phương pháp tấn công đối kháng sẽ sinh ảnh đối kháng với mức độ thêm nhiễu đối kháng nhỏ nhất theo độ đo L_p và được định nghĩa như Công thức 2.9 [17, 40].

$$L_p(\mathbf{x}, \mathbf{x}') = \left(\sum_i \left(x_i - x_i' \right)^p \right)^{\frac{1}{p}} \in [0, +\infty] \tag{2.9}$$

trong đó, $p \in \{0, 1, 2, \infty\}$. Độ đo L_0 gọi là khoảng cách Hamming và dùng để đếm số điểm ảnh đối kháng. Các phương pháp phổ biến có thể kể đến CW L_0 [17], DeepCheck [67, 68] và NEUROSPF [69]. Độ đo L_2 gọi là khoảng cách Euclidean. Phương pháp CW L_2 [17], ATN [18], L-BFGS [32] và DeepFool [33] là những phương pháp tiêu biểu. Độ đo L_∞ tính giá trị tuyệt đối chênh lệch lớn nhất giữa điểm ảnh trên ảnh dự đoán đúng và điểm ảnh tương ứng trên ảnh đối kháng. Các phương pháp phổ biến có thể kể đến FGSM [16], CW L_∞ [17], BIS [19], MI-FGSM [34] và PGD [35].

2.3.5.2. Tiêu chí tỉ lệ thành công

Tỉ lệ thành công là một tiêu chí phổ biến để đánh giá tính chắc chắn của mô hình học sâu trước một phương pháp tấn công đối kháng [36, 66]. Tiêu chí tỉ lệ thành công phản ánh khả năng thêm nhiễu đối kháng vào ảnh dự đoán đúng để sinh ảnh đối kháng thành công. Trong tấn công đối kháng có định hướng, tỉ lệ thành công được định nghĩa như Công thức 2.10.

$$\frac{\sum_{\mathbf{x}' \in \mathbf{X}_{adv}} \mathbb{1}(\arg\max(\mathbf{M}(\mathbf{x}')) = y^*)}{\mid \mathbf{X}_{adv} \mid} \in [0, 1] \tag{2.10}$$

trong đó, $\mathbf{X}_{adv}$ là tập ảnh đối kháng và $\mathbb{1}$ là hàm chỉ thị. Hàm $\mathbb{1}(.)$ trả về một nếu $\mathbf{M}$ phân lớp $\mathbf{x}'$ giống nhãn đích y^* và trả về không trong trường hợp ngược lại.

Trong tấn công đối kháng không định hướng, tỉ lệ thành công được định nghĩa như Công thức 2.11.

$$\frac{\sum_{\mathbf{x}' \in \mathbf{X}_{adv}} \mathbb{1}(\arg\max(\mathbf{M}(\mathbf{x}')) \neq y_{true})}{\mid \mathbf{X}_{adv} \mid} \in [0, 1] \tag{2.11}$$

trong đó, hàm chỉ thị trả về một nếu $\mathbf{M}$ phân lớp $\mathbf{x}'$ khác nhãn đúng và trả về không trong trường hợp ngược lại.

2.3.5.3. Tiêu chí tỉ lệ giảm nhiễu

Các phương pháp tấn công đối kháng sinh ảnh đối kháng có thể chứa nhiễu dư thừa. Nếu loại bỏ những nhiễu dư thừa này thì chất lượng ảnh đối kháng sẽ tăng lên. Tỉ lệ giảm nhiễu được tính bằng $(a-b)/a \in [0, 1)$. Trong đó, a là khoảng cách L_p giữa ảnh dự đoán đúng và ảnh đối kháng chưa cải thiện. Khoảng cách giữa ảnh dự đoán đúng và ảnh đối kháng cải thiện được kí hiệu là b. Xét một phương pháp tấn công đối kháng, giá trị tỉ lệ giảm nhiễu thường càng cao thì phương pháp tấn công đối kháng càng kém hiệu quả. Giá trị tỉ lệ giảm nhiễu thường càng nhỏ thì phương pháp tấn công đối kháng càng hiệu quả.

2.3.6. Các phương pháp tấn công đối kháng không định hướng

2.3.6.1. Phương pháp DeepCheck

Phần này giải thích tư tưởng của phương pháp DeepCheck [42] để sinh ảnh đối kháng bằng cách thêm nhiễu đối kháng vào một điểm ảnh trên ảnh dự đoán đúng. Tổng quan các bước của phương pháp DeepCheck được mô tả trong Thuật toán 2.1. Đầu vào gồm mô hình nơ-ron truyền thẳng (kí hiệu là $\mathbf{M}$) và một ảnh dự đoán đúng (kí hiệu là $\mathbf{x}$). Đầu ra là ảnh đối kháng (kí hiệu là $\mathbf{x}'$) trong đó nhãn của $\mathbf{x}'$ khác nhãn của $\mathbf{x}$.

Thuật toán 2.1 : Phương pháp DeepCheck (trường hợp thêm nhiễu đối kháng vào một điểm ảnh)

Đầu vào: mô hình nơ-ron truyền thẳng $\mathbf{M}$ và ảnh dự đoán đúng $\mathbf{x}$
Đầu ra: ảnh đối kháng $\mathbf{x}'$

1: $p = \textsc{Translate}(\mathbf{M})$ ▷ Chuyển mô hình $\mathbf{M}$ thành chương trình p
2: $tp = \textsc{Execute}(p, \mathbf{x})$ ▷ Thực thi chương trình p với đầu vào là ảnh $\mathbf{x}$
3: $c_{hidden} = \textsc{SymbolicExecution}(tp)$ ▷ Xây dựng hệ ràng buộc
4: $c_{out} = \bigvee_{i \neq y_{true}} n_{pre}^{y_{true}} < n_{pre}^{i}$
5: $c = c_{hidden} \bigwedge c_{out}$
6: $s = \textsc{RankFeatures}(\mathbf{x})$ ▷ Xếp hạng đặc trưng (là các điểm ảnh)
7: $iter = 0$
8: **while** chưa tìm thấy ảnh đối kháng **do**
9: $smtlib = \textsc{CreateConstraint}(c, s_{iter})$ ▷ Tạo hệ ràng buộc
10: $\mathbf{x}' = \textsc{CallSmtSolver}(smtlib)$ ▷ Giải hệ ràng buộc
11: **if** $\mathbf{x}'$ tồn tại **then**
12: return $\mathbf{x}'$
13: **else**
14: $iter$ += 1
15: **end if**
16: **end while**

Đầu tiên, phương pháp DeepCheck biến đổi mô hình kiểm thử thành mã nguồn C kí hiệu là p (dòng 1). Ví dụ, hàm kích hoạt ReLU của nơ-ron n_i^j ở tầng ẩn sẽ chuyển thành câu lệnh `if (n_i_j < 0) n_i_j = 0;`, trong đó `n_i_j` là tên biến tương ứng với nơ-ron n_i^j. Sau đó, phương pháp DeepCheck biên dịch và thực thi mã nguồn p với đầu vào là một ảnh dự đoán đúng $\mathbf{x}$ để lấy đường thi hành tp (dòng 2). Sau khi thực thi, đường thi hành chứa các câu lệnh và nhánh được viếng thăm khi thực thi đầu vào $\mathbf{x}$. Mục tiêu của phương pháp DeepCheck

là tìm một ảnh đối kháng $\mathbf{x}'$ đi qua đường thi hành này. Nói cách khác, $\mathbf{x}$ và $\mathbf{x}'$ có cùng trạng thái kích hoạt nơ-ron.

Sau đó, phương pháp DeepCheck áp dụng thực thi tượng trưng [70] trên đường thi hành để sinh hệ ràng buộc kí hiệu là $c_{hidden} = c_0 \wedge c_1 \wedge ... \wedge c_{g-1}$, trong đó g là số điểm quyết định và c_i là điểm quyết định tương ứng với hàm kích hoạt của nơ-ron ẩn thứ i (dòng 3). Tại bước này, c_i được biểu diễn thành $\sum_j f_j \cdot a_j + z$, trong đó a_j và z là các hệ số, f_j là điểm ảnh trừu tượng. Điểm ảnh trừu tượng f_j tương ứng với điểm ảnh thứ i trên ảnh dự đoán đúng. Giá trị khởi đầu của điểm ảnh trừu tượng f_j là giá trị của điểm ảnh thứ i. DeepCheck thêm nhiễu đối kháng vào một hoặc hai điểm ảnh trừu tượng để sinh ảnh đối kháng. Đối với những điểm ảnh không phải trừu tượng, DeepCheck sẽ không thêm nhiễu đối kháng vào các điểm ảnh này.

Tuy nhiên, c_{hidden} không chứa ràng buộc đầu ra. Do đó, phương pháp DeepCheck cần phải xây dựng ràng buộc đầu ra kí hiệu là c_{out} (dòng 4). Ràng buộc đầu ra có dạng $\bigvee_{i \neq y_{true}} n_{pre}^{y_{true}} < n_{pre}^{i}$, trong đó n_{pre}^{i} là nơ-ron thứ i nằm ở tầng pre-softmax. Hệ ràng buộc đầu ra đảm bảo rằng nhãn dự đoán của ảnh đối kháng $\mathbf{x}'$ khác nhãn của $\mathbf{x}$. Hệ ràng buộc cuối cùng c là phép hội của c_{hidden} và c_{out} (dòng 5). Sau đó, phương pháp DeepCheck xếp hạng các điểm ảnh trong $\mathbf{x}$ theo chiều giảm dần mức độ quan trọng (dòng 6). Mỗi điểm ảnh được gán cho một điểm số phản ánh tầm quan trọng của điểm ảnh đó đến quyết định của mô hình. DeepCheck sẽ thêm nhiễu đối kháng vào những điểm ảnh quan trọng nhất và giữ nguyên những điểm ảnh còn lại (dòng 9), sau đó sử dụng bộ giải SMT để tìm nghiệm (dòng 10). Nghiệm này chính là ảnh đối kháng. Nếu phương pháp DeepCheck tìm được ảnh đối kháng, thuật toán trả về ảnh này và kết thúc. Ngược lại, phương pháp DeepCheck tiếp tục thêm nhiễu đối kháng vào điểm ảnh quan trọng nhất kế tiếp.

Hình 2.6 trình bày một hệ ràng buộc sinh bởi cài đặt DeepCheck thực hiện bởi luận án. Hệ ràng buộc này được mô tả theo định dạng của SMT-LIB [71]. Đây là định dạng đầu vào chuẩn của bộ giải SMT. Điểm ảnh quan trọng nhất được kí hiệu là f_{64} và đây là điểm ảnh tượng trưng. Các điểm ảnh còn lại được giữ nguyên giá trị, tức không phải điểm ảnh tượng trưng. Hệ ràng buộc có ba phần gồm Declaration, ReLU và Output. Hệ ràng buộc con ReLU đảm bảo rằng

nghiệm sẽ có cùng trạng thái kích hoạt nơ-ron với ảnh dự đoán đúng. Trong phần Declaration, hệ ràng buộc `and (>= f64 0) (<= f64 255)` tương đương với `f64 >= 0 && f64 <= 255`, trong đó 0 và 255 là cận dưới và cận trên của giá trị điểm ảnh trong khoảng số nguyên. Câu lệnh `check-sat` gọi bộ giải. Câu lệnh `get-model` trả về nghiệm của hệ ràng buộc. Câu lệnh `assert` dùng để đánh dấu là hệ ràng buộc.

```
; Declaration
(declare-fun f64 () Int)
(assert (and (>= f64 0) (<= f64 255)))

; RELU
(assert (< (+ 0.872347975799992 (* (* (- 1) 0.0133442345 ) f64 ) ) 0 ) )
(assert (< (+ (* (- 1) 3.626597374900004 ) (* 0.0007956961 f64 ) ) 0 ) )
(assert (< (+ (* (- 1) 13.164641482500004 ) (* (* (- 1) 0.0016546831 ) f64 ) ) 0 ) )
(assert (< (+ (* (- 1) 13.177424374899996 ) (* 0.0016008431 f64 ) ) 0 ) )
(assert (< (+ (* (- 1) 51.985817459799975 ) (* (* (- 1) 0.0050446024 ) f64 ) ) 0 ) )
(assert (< (+ (* (- 1) 20.300508148600006 ) (* 0.0014267944 f64 ) ) 0 ) )
(assert (< (+ (* (- 1) 40.972821615799994 ) (* (* (- 1) 0.0106787475 ) f64 ) ) 0 ) )
(assert (< (+ (* (- 1) 7.116009313400009 ) (* 0.0021295296 f64 ) ) 0 ) )
(assert (< (+ (* (- 1) 37.49786409929999 ) (* 0.0019227811 f64 ) ) 0 ) )
(assert (< (+ (* (- 1) 13.688510029899994 ) (* 0.010302151 f64 ) ) 0 ) )
(assert (< (+ (* (- 1) 20.916028322800013 ) (* (* (- 1) 0.0112519815 ) f64 ) ) 0 ) )
(assert (< (+ (* (- 1) 7.096488964300004 ) (* (* (- 1) 0.0068205099 ) f64 ) ) 0 ) )
(assert (< (+ (* (- 1) 10.031844753999998 ) (* (* (- 1) 0.0008223966 ) f64 ) ) 0 ) )
(assert (< (+ (* (- 1) 0.37501648269997245 ) (* (* (- 1) 0.0020991753 ) f64 ) ) 0 ) )
(assert (< (+ (* (- 1) 41.43603867770002 ) (* (* (- 1) 0.009205805 ) f64 ) ) 0 ) )
(assert (< (+ (* (- 1) 20.355946046599986 ) (* 0.0041465844 f64 ) ) 0 ) )

; Output
(assert (or (or (or (or (or (or (or (or (> (+ (* (- 1) 55.96716274720002 ) (* 0.0449472365
f64 ) ) 0 ) (> (+ (* (- 1) 111.63453761509997 ) (* 0.0216533459 f64 ) ) 0 ) ) (> (+ (* (-
1) 8.081172991199997 ) (* 0.0479857836 f64 ) ) 0 ) ) (> (+ (* (- 1) 246.4932872913 )
(* 0.0330343348 f64 ) ) 0 ) ) (> (+ (* (- 1) 25.812896548599984 ) (* 0.0187184767 f64 )
) 0 ) ) (> (+ (* (- 1) 69.22932785259994 ) (* 0.0856898131 f64 ) ) 0 ) ) (> (+ (* (- 1)
92.0498899405001 ) (* 0.0228416665 f64 ) ) 0 ) ) (> (+ (* (- 1) 54.96181328419996 ) (*
0.0444276011 f64 ) ) 0 ) ) (> (+ (* (- 1) 122.90882463289995 ) (* 0.0078221164 f64 ) )
0 ) ) )

(check-sat)
(get-model)
```

Hình 2.6: Minh họa một hệ ràng buộc sinh ra bởi DeepCheck cài đặt bởi luận án.

2.3.6.2. Phương pháp FGSM

Goodfellow và cộng sự đề xuất FGSM [16] để sinh ảnh đối kháng theo tiêu chí L_∞. Nhiễu được thêm vào dựa theo cường độ của đạo hàm với các điểm ảnh trong ảnh dự đoán đúng. Cụ thể, giá trị các điểm ảnh trong ảnh dự đoán đúng được thêm nhiễu đối kháng với mục tiêu làm giảm xác suất của nơ-ron ứng với nhãn đúng. Phương pháp này sẽ cập nhật mọi điểm ảnh trong ảnh dự đoán đúng $\mathbf{x}$ với một lần lặp như Công thức 2.12.

$$\mathbf{x}' = \mathbf{x} - \epsilon_F \cdot \text{SIGN}(\nabla_{\mathbf{x}} \mathbf{M}_{y_{true}}(\mathbf{x})) \tag{2.12}$$

trong đó, hàm SIGN $\in \{-1, 0, 1\}$ trả về dấu của biểu thức bên trong và $\epsilon_F \in (0, 1]$ là một giá trị thực dương nhỏ để điều chỉnh cường độ thêm nhiễu. Nếu $\epsilon_F = 1$, nhiễu được thêm với cường độ lớn nhất. Nguyên nhân của dấu trừ trong công thức này là do nếu dấu $\nabla_{\mathbf{x}} \mathbf{M}_{y_{true}}(\mathbf{x})$ là dương, giá trị các điểm ảnh trong $\mathbf{x}$ nên được giảm đi với mục tiêu nghịch đảo xu hướng tăng của $\mathbf{M}_{y_{true}}(\mathbf{x})$. Ngược lại, nếu dấu $\nabla_{\mathbf{x}} \mathbf{M}_{y_{true}}(\mathbf{x})$ là âm, tức giá trị các điểm ảnh trong $\mathbf{x}$ nên được tăng lên để tiếp tục xu hướng giảm của $\mathbf{M}_{y_{true}}(\mathbf{x})$.

2.3.6.3. Phương pháp BIM

Kurakin và cộng sự đề xuất BIM [19] là cải tiến của FGSM. Trong khi FGSM chỉ lặp duy nhất một lần, BIM thêm nhiễu với nhiều lần lặp. Tại mỗi lần lặp, lượng nhiễu do BIM thêm vào thường khá nhỏ so với FGSM. Nếu ảnh có nhiễu chưa phải ảnh đối kháng, ảnh tiếp tục được thêm nhiễu trong lần lặp kế tiếp. Công thức đề xuất bởi nhóm cộng sự được mô tả như Công thức 2.13.

$$\mathbf{x}_i = \text{CLIP}(\mathbf{x}_{i-1} - \epsilon_B \cdot \text{SIGN}(\nabla_{\mathbf{x}} \mathbf{M}_{y_{true}}(\mathbf{x})), \alpha_B) \tag{2.13}$$

trong đó, $\mathbf{x}_0 = \mathbf{x}$, hàm CLIP đảm bảo rằng $\mathbf{x}_i$ trong khoảng không gian quanh $\mathbf{x}$ với khoảng cách α_B và $\epsilon_B \in (0, 1]$.

2.3.7. Các phương pháp tấn công đối kháng có định hướng

2.3.7.1. Phương pháp L-BFGS

Szegedy và cộng sự [32] đề xuất phương pháp L-BFGS để sinh ảnh đối kháng bằng cách tối thiểu hóa hàm mục tiêu trình bày trong Công thức 2.14.

$$\begin{aligned} &\text{tối thiểu hóa } \ (1-\beta)\cdot L_2(\mathbf{x}', \mathbf{x})^2 + \beta \cdot f(y^*, \mathbf{M}(\mathbf{x}')) \\ &\text{thỏa mãn } \mathbf{x}' \in [0,1]^d \end{aligned} \tag{2.14}$$

trong đó, $f(.)$ là hàm tính khoảng cách $\mathbf{M}(\mathbf{x}')$ và véc-tơ ứng với nhãn đích y^*. Tham số β là trọng số để cân bằng hai thành phần. Một sự lựa chọn phổ biến của $f(.)$ là cross-entropy. Độ đo khoảng cách sử dụng là L_2.

2.3.7.2. Phương pháp FGSM

Goodfellow và cộng sự [16] đề xuất FGSM để sinh ảnh đối kháng bằng cách thêm nhiễu đối kháng vào mọi điểm ảnh trên ảnh dự đoán đúng. Độ đo khoảng cách sử dụng là L_∞. Ảnh dự đoán đúng được thêm nhiễu đối kháng như Công thức 2.15.

$$\mathbf{x}' = \mathbf{x} + \beta \cdot \text{SIGN}(\nabla_{\mathbf{x}} \mathbf{M}_{y^*}(\mathbf{x})) \tag{2.15}$$

trong đó, $\beta \in (0, 1]$ là một giá trị thực dương dùng để thay đổi giá trị của các điểm ảnh trong $\mathbf{x}$. Nhược điểm chính của FGSM là cách chọn giá trị β. Nếu giá trị β đủ lớn, tỉ lệ thành công của phương pháp này thường cao. Tuy nhiên, điểm ảnh có thể bị thêm nhiễu đối kháng với lượng nhiễu nhiều hơn và có thể phá vỡ cảm quan về ảnh khi nhìn vào bằng mắt thường. Ngược lại, nếu giá trị β quá nhỏ, tỉ lệ thành công sẽ khá thấp.

2.3.7.3. Phương pháp CW L_2

Lấy cảm hứng từ phương pháp L-BFGS, Carlini và cộng sự [17] đề xuất phương pháp sinh ảnh đối kháng với độ đo khoảng cách L_2 như Công thức 2.16.

$$\begin{aligned} &\text{tối thiểu hóa } \ (1-\beta)\cdot L_2(\mathbf{x}', \mathbf{x})^2 + \beta \cdot f(y^*, \mathbf{M}(\mathbf{x}')) \\ &\text{thỏa mãn } \mathbf{x}' \in [0,1]^d \end{aligned} \tag{2.16}$$

trong đó, f được định nghĩa như Công thức 2.17.

$$\begin{aligned} f(\mathbf{x}') &= \text{MAX}(Z(\mathbf{x}')_i : \forall i \neq y^*) - Z(\mathbf{x}')_{y^*} \\ f(\mathbf{x}') &= \text{MAX}(f(\mathbf{x}'), -t) \end{aligned} \tag{2.17}$$

trong đó, β là trọng số để cân bằng hai thành phần, t được sử dụng để điều khiển độ tin cậy của đầu ra, $Z(.)$ trả về giá trị trước khi áp dụng soft-max ở tầng đầu ra, thành phần $\text{MAX}(Z(\mathbf{x}')_i : \forall i \neq y^*)$ là giá trị pre-softmax lớn nhất ngoại trừ của nơ-ron đích và thành phần $Z(\mathbf{x}')_{y^*}$ là giá trị trước khi áp dụng soft-max của nơ-ron đích.

2.3.7.4. Phương pháp ATN

Baluja và cộng sự [18] đề xuất ATN để sinh ảnh đối kháng bằng cách sử dụng mô hình mã hóa tự động tích chập xếp chồng. Đầu vào và đầu ra của ATN là ảnh dự đoán đúng và ảnh đối kháng. Nhóm tác giả đề xuất sử dụng độ đo khoảng cách L_2 để tính sự khác biệt giữa ảnh dự đoán đúng và ảnh đối kháng. Hàm mục tiêu của ATN được phát biểu như Công thức 2.18.

$$\sum_{\mathbf{x}} \beta \cdot L_2(\mathbf{x}, \mathbf{x}') + L_2(\mathbf{M}(\mathbf{x}'), r_\alpha(\mathbf{M}(\mathbf{x}), y^*)) \tag{2.18}$$

trong đó, β là trọng số để cân bằng hai thành phần. Hàm $r_\alpha(.)$ sửa $\mathbf{M}(\mathbf{x})$ với kì vọng rằng lượng nhiễu đối kháng thêm vào ảnh dự đoán đúng $\mathbf{x}$ là nhỏ nhất và được trình bày trong Công thức 2.19.

$$r_\alpha(\mathbf{M}(\mathbf{x}), y^*) = \text{NORM}\left(\left\{\begin{array}{ll} \alpha \cdot \text{MAX}(\mathbf{M}(\mathbf{x})) \text{ nếu } i = y^* \\ \mathbf{M}(\mathbf{x})_i \quad \text{ngược} \quad \text{lại} \end{array}\right\}_{i \in \{0..c-1\}}\right) \tag{2.19}$$

trong đó, α lớn hơn một và hàm NORM(.) chuẩn hóa véc tơ thành mảng xác suất.

2.4. Các phương pháp phòng thủ sử dụng mô hình mã hóa tự động

2.4.1. Phương pháp PuVAE

PuVAE xây dựng mô hình mã hóa tự động biến thiên có điều kiện để học phân phối của những ảnh này. Tập học là những ảnh không có nhiễu, thường là tập học của mô hình kiểm thử. Mô hình mã hóa tự động biến thiên có điều kiện có thể coi là bộ lọc nhiễu trong MagNet vì mục đích là giống nhau. Hàm mục tiêu của PuVAE như Công thức 2.20.

$$\alpha_P \cdot L_{KL} + \beta_P \cdot L_2(\mathbf{x}, \mathbf{x}') + \gamma_P \cdot \text{CE}(\mathbf{y}_{true}, \mathbf{M}(\mathbf{x}')) \tag{2.20}$$

trong đó, L_{KL} là công thức hội tụ Kullback-Leibler giữa phân phối không gian ẩn và phân phối Gaussian, $L_2(\mathbf{x}, \mathbf{x}')$ là khoảng cách giữa $\mathbf{x}$ và $\mathbf{x}'$, $\text{CE}(\mathbf{y}_{true}, \mathbf{M}(\mathbf{x}'))$ là sự khác biệt giữa véc tơ xác suất mong đợi $\mathbf{y}_{true}$ và dự đoán mô hình $\mathbf{M}(\mathbf{x}')$. Bộ ba (α_P, β_P, γ_P) là trọng số của các thành phần.

2.4.2. Phương pháp MagNet

MagNet có hai thành phần chính là bộ nhận diện và bộ lọc nhiễu. Trong khi bộ nhận diện phân lớp ảnh đầu vào có tính tự nhiên hay không, bộ lọc nhiễu sẽ loại bỏ nhiễu trong ảnh đầu vào để tạo ảnh mới mà mô hình kiểm thử dự đoán

chính xác hơn. Đối với bộ nhận diện, MagNet coi các ảnh thuộc tập học của mô hình kiểm thử có tính tự nhiên. Tập học của bộ nhận diện gồm tập học của mô hình kiểm thử và tập học có nhiễu Gaussian nhỏ. Bộ nhận diện được học để chuyển các ảnh đầu vào, có thể có nhiễu, về một phiên bản khác. Để xác định một ảnh đầu vào có tính tự nhiên, MagNet có hai cách. Cách đầu tiên là so sánh khoảng cách L_2 giữa ảnh đầu vào và phiên bản. Cách thứ hai là so sánh hội tụ Kullback-Leibler của hai véc tơ xác suất dự đoán của ảnh đầu vào và phiên bản. Giá trị L_2 hoặc hội tụ Kullback-Leibler càng nhỏ thì ảnh đầu vào càng có khả năng là ảnh tự nhiên. Đối với ảnh đầu vào được bộ nhận diện đánh giá là ảnh tự nhiên, ảnh đầu vào được đẩy qua bộ lọc nhiễu. Bộ lọc nhiễu chuyển ảnh đầu vào thành ảnh mới và ảnh này dễ nhận diện chính xác hơn bởi mô hình kiểm thử.

2.4.3. Phương pháp Defense-VAE

Defense-VAE sử dụng mô hình mã hóa tự động biến thiên để loại bỏ nhiễu đối kháng khỏi ảnh đầu vào. Tập học của mô hình mã hóa tự động biến thiên gồm tập học của mô hình kiểm thử và ảnh đối kháng sinh bởi nhiều phương pháp khác nhau như FGSM [16] và CW [17]. Do đó, khi có ảnh đối kháng đến mô hình kiểm thử, Defense-VAE có thể tạo ảnh mới được nhận diện chính xác hơn từ ảnh đối kháng này. Hàm mục tiêu của Defense-VAE tương tự như Công thức 2.20.

2.4.4. Tỉ lệ phát hiện để đánh giá chất lượng mô hình mã hóa tự động phòng thủ

Tỉ lệ phát hiện là một độ đo phổ biến để đánh giá tính hiệu quả của phương pháp cải thiện tính chắc chắn [3, 46, 49, 50] và được định nghĩa như Công thức 2.21.

$$\frac{\sum_{\mathbf{x}_{im} \in X_{def}} \mathbb{1}(\arg\max(\mathbf{M}(\mathbf{A}(\mathbf{x}))) = y_{true})}{\mid X_{def} \mid} \in [0, 1] \tag{2.21}$$

trong đó, X_{def} là tập ảnh đầu vào và **A** là mô hình mã hóa tự động. Nếu tỉ lệ phát hiện bằng 1 thì mọi ảnh đầu vào đều được loại bỏ nhiễu đối kháng nếu có và ảnh chỉnh sửa nhận diện chính xác.

2.5. Các bộ dữ liệu sử dụng trong thực nghiệm

Phần này trình bày về các bộ dữ liệu được sử dụng trong thực nghiệm gồm MNIST [41], Fashion-MNIST [72], CIFAR-10 [54] và bộ chữ cái viết tay[1]. Bảng 2.1 trình bày về các bộ dữ liệu này. Ảnh xám là ảnh chỉ có một kênh màu, giá trị điểm ảnh từ 0 đến 1 nếu là số thực hoặc từ 0 đến 255 nếu là số nguyên.

Bảng 2.1: Mô tả bộ dữ liệu sử dụng trong các thực nghiệm

Bộ dữ liệu	**Mô tả**	**Tập học**	**Tập kiểm thử**	**Số nhãn**	**Màu**	**Số chiều**
MNIST	chữ số viết tay	60,000	10,000	10	xám	$28 \times 28 \times 1$
Fashion MNIST	trang phục	60,000	10,000	10	xám	$28 \times 28 \times 1$
CIFAR-10	đối tượng	60,000	10,000	10	màu	$28 \times 28 \times 3$
Bộ chữ cái viết tay	chữ cái viết tay	297,959	74,489	26	xám	$28 \times 28 \times 1$

Bộ dữ liệu MNIST là một trong những bộ dữ liệu phổ biến trong lĩnh vực nhận dạng chữ viết tay. Bộ dữ liệu này bao gồm một tập các ảnh từ số không đến số chín, trong đó mỗi ảnh được viết tay bởi những người khác nhau. Bộ dữ liệu này gồm có hai tập con gồm tập học và tập kiểm thử. Tập học chứa 60,000 ảnh và tập kiểm thử chứa 10,000 ảnh. Mỗi ảnh trong bộ dữ liệu có kích thước là $28 \times 28 \times 1$ và là ảnh xám.

Tương tự như MNIST, bộ dữ liệu Fashion-MNIST là một bộ dữ liệu được sử dụng phổ biến trong lĩnh vực nhận dạng hình ảnh. Bộ dữ liệu này chứa một tập các ảnh thuộc mười loại sản phẩm khác nhau như áo phông, áo khoác, giày, v.v. Mỗi ảnh trong bộ dữ liệu có kích thước $28 \times 28 \times 1$ và là ảnh xám. Tập học chứa 60,000 ảnh và tập kiểm thử chứa 10,000 ảnh.

[1] https://www.kaggle.com/datasets/sachinpatel21/az-handwritten-alphabets-in-csv-format

Bộ dữ liệu CIFAR-10 chứa một tập hợp các ảnh có kích thước $28 \times 28 \times 3$ thuộc mười loại khác nhau như máy bay, xe hơi, chim, v.v. CIFAR-10 bao gồm 60,000 hình ảnh màu RGB được chia thành mười loại với khoảng 6,000 ảnh cho mỗi loại.

Cuối cùng, bộ dữ liệu bộ chữ cái viết tay chứa các kí tự viết tay gồm viết thường và viết hoa. Kích thước bộ dữ liệu này lớn hơn ba bộ dữ liệu trước với tập học chứa 297,959 ảnh và tập kiểm thử chứa 74,489 ảnh. Kích thước một ảnh là $28 \times 28 \times 1$ và là ảnh xám.

2.6. Bộ giải SMT

Bộ giải SMT được sử dụng để kiểm tra tính thỏa mãn của hệ ràng buộc. Đầu vào là hệ ràng buộc được mô tả theo chuẩn SMT-Lib [71]. Đầu ra là nghiệm của hệ ràng buộc nếu có nghiệm hoặc thông báo vô nghiệm. Các bộ giải phổ biến có thể kể đến Z3 [73], SMT-Interpol [74], v.v. Ví dụ, bộ giải Z3 hỗ trợ hai môi trường gồm Window và Ubuntu. Để giải hệ ràng buộc, Z3 chuyển đầu vào về chuẩn SMT-Lib. Sau đó, biểu thức SMT-Lib này được giải qua dòng lệnh sử dụng Z3. Giả sử biểu thức SMT-Lib này lưu trong *D:/constraints.txt* lưu tại ổ D, cú pháp gọi bộ giải Z3 như sau: *[đường dẫn z3] –smt2 D:/constraints.txt.*

Người dùng có thể giải hệ trực tuyến qua hệ thống Z3 cung cấp sẵn tại đường dẫn này[2]. Hình 2.7 minh họa một hệ ràng buộc theo chuẩn SMT-Lib. Hình 2.8 minh họa kết quả giải hệ ràng buộc sử dụng Z3. Nếu hệ ràng buộc vô nghiệm, hệ thống trả về UNSAT. Ngược lại, hệ thống thông báo SAT cũng với nghiệm thể hiện ở dưới. Trường hợp hệ ràng buộc có vô số nghiệm, Z3 trả về một nghiệm bất kỳ. Trong ví dụ này, Z3 trả về một bộ giá trị ngẫu nhiên (x, y) = (2, 1).

```
1   (declare-const x Int)
2   (declare-const y Int)
3   (assert (< x 3))
4   (assert (> x y))
5   (check-sat)
6   (get-model)
```

Hình 2.7: Ví dụ hệ ràng buộc theo chuẩn SMT-Lib.

[2]https://jfmc.github.io/z3-play/

```
sat
(
  (define-fun y () Int
    1)
  (define-fun x () Int
    2)
)
```

Hình 2.8: Ví dụ nghiệm của hệ ràng buộc theo chuẩn SMT-Lib.

2.7. Tổng kết

Chương này đã trình bày kiến thức nền tảng về hướng nghiên cứu kiểm thử tính chắc chắn của mô hình học sâu. Luận án đã trình bày khái niệm mô hình học sâu, sau đó trình bày hai loại mô hình học sâu phổ biến gồm mô hình nơ-ron truyền thẳng và mô hình tích chập. Kế tiếp, luận án đã trình bày về các mô hình mã hóa tự động như mô hình mã hóa tự động thưa, mô hình mã hóa tự động xếp chồng và mô hình mã hóa tự động tích chập xếp chồng. Sau đó, luận án đã mô tả các phương pháp tấn công đối kháng theo hướng tấn công đối kháng không định hướng và tấn công đối kháng có định hướng, làm rõ các tiêu chí đánh giá chất lượng phương pháp tấn công đối kháng. Các phương pháp phòng thủ sử dụng mô hình mã hóa tự động như Defense-VAE, PuVAE và MagNet đã được mô tả. Chương mô tả các bộ dữ liệu sử dụng trong thực nghiệm gồm MNIST, Fashion-MNIST, CIFAR-10 và bộ chữ cái viết tay. Cuối cùng, chương mô tả về bộ giải SMT.

Chương 3

Phương pháp sử dụng bộ giải phỏng đoán để tấn công đối kháng không định hướng mô hình nơ-ron truyền thẳng

Chương này trình bày cải tiến của phương pháp DeepCheck mô tả ở Phần 2.3.6.1, gọi là HA4FNN. Mô hình kiểm thử là mô hình nơ-ron truyền thẳng. Các cải tiến này góp phần làm tăng tỉ lệ thành công và giảm chi phí tính toán của DeepCheck bằng cách sử dụng hai kĩ thuật. HA4FNN không sử dụng bộ giải SMT mà dùng bộ giải phỏng đoán. Ngoài ra, HA4FNN không sử dụng tiêu chí sinh ảnh đối kháng có cùng trạng thái kích hoạt nơ-ron với ảnh dự đoán đúng. Kết quả thực nghiệm trên bộ dữ liệu MNIST, Fashion-MNIST và bộ chữ cái viết tay cho thấy hiệu quả của phương pháp đề xuất.

3.1. Giới thiệu

Mô hình nơ-ron truyền thẳng là một loại mô hình học sâu trong đó các lớp được kết nối lần lượt với nhau từ lớp đầu tiên đến lớp cuối cùng [4, 55]. Một trong những ứng dụng phổ biến của mô hình nơ-ron truyền thẳng là nhận diện nhãn của ảnh. Tuy nhiên, dù mô hình nơ-ron truyền thẳng được học cẩn thận từ bộ dữ liệu chất lượng, mô hình nơ-ron truyền thẳng vẫn dễ gặp phải vấn đề tính chắc chắn [16, 18, 32]. Do đó, các hệ thống học máy sử dụng mô hình học sâu nói chung và mô hình nơ-ron truyền thẳng nói riêng cần được kiểm thử cẩn thận bởi nhiều độ đo khác nhau bên cạnh sử dụng các độ đo truyền thống như độ chuẩn xác, độ chính xác và điểm số F1. Hướng tấn công đối kháng không

định hướng là một hướng tiếp cận phổ biến để đánh giá tính chắc chắn của mô hình nơ-ron truyền thẳng [15]. Kẻ tấn công có thể thêm nhiễu đối kháng vào ít nhất một điểm ảnh trong ảnh dự đoán đúng, mà ảnh này được gán nhãn đúng, để sinh ra một ảnh mới gọi là ảnh đối kháng. Mô hình nơ-ron truyền thẳng nhận diện sai nhãn ảnh đối kháng này.

DeepCheck [67, 68] là một kĩ thuật mới theo hướng tấn công đối kháng không định hướng. DeepCheck hỗ trợ kiểm thử mô hình nơ-ron truyền thẳng sử dụng hàm kích hoạt ReLU [56] ở tầng ẩn và softmax ở tầng đầu ra. Khác với các kĩ thuật đã đề xuất trước đó như FGSM [16], ATN [18], DeepFool [33], v.v. dựa theo đạo hàm, DeepCheck sử dụng kĩ thuật thực thi tượng trưng [70] và bộ giải SMT như Z3 [73] hoặc SMTInterpol [74] để sinh ảnh đối kháng từ ảnh dự đoán đúng. Tiêu chí khoảng cách giữa ảnh dự đoán đúng và ảnh đối kháng mà DeepCheck sử dụng là L_0. Cụ thể, từ một ảnh dự đoán đúng, DeepCheck thêm nhiễu đối kháng vào một hoặc hai điểm ảnh để sinh ảnh đối kháng, trong đó trạng thái kích hoạt nơ-ron của ảnh đối kháng và ảnh dự đoán đúng giống hệt nhau. Trước khi sử dụng hàm kích hoạt θ_i, nếu giá trị nơ-ron n_i^j lớn hơn 0, nơ-ron này có trạng thái được kích hoạt và ngược lại. Với ảnh dự đoán đúng, nếu nơ-ron n_i^j được kích hoạt thì trong ảnh đối kháng, giá trị nơ-ron này phải lớn hơn 0.

Tư tưởng chung của DeepCheck gồm bốn bước. Đầu tiên, mô hình kiểm thử được chuyển sang mã nguồn C. Sau đó, mã nguồn C được chèn các câu lệnh để đánh dấu câu lệnh hoặc nhánh viếng thăm khi mã nguồn này được thực thi. Kế tiếp, ảnh dự đoán đúng được biến đổi thành đầu vào của mã nguồn này và thực thi để lấy đường thi hành. DeepCheck áp dụng thực thi tượng trưng trên đường thi hành này để sinh hệ ràng buộc. Trạng thái kích hoạt nơ-ron của ảnh dự đoán đúng được biểu diễn trong hệ ràng buộc này. DeepCheck xác định một hoặc hai điểm ảnh quan trọng nhất và coi các điểm ảnh này là biến trong khi cố định giá trị các điểm ảnh còn lại. Từ đó, hệ ràng buộc chỉ còn có một hoặc hai biến. Bộ giải SMT giải hệ ràng buộc để tìm giá trị các biến tương ứng với các điểm ảnh quan trọng nhất này. Cuối cùng, các điểm ảnh này sẽ thêm nhiễu đối kháng để sinh ra ảnh đối kháng. Tuy nhiên, thực nghiệm cho thấy DeepCheck có tỉ lệ thành công và hiệu năng khá thấp. Ví dụ, với mô hình nơ-ron truyền thẳng có 30 nơ-ron ẩn được học trên bộ dữ liệu MNIST, DeepCheck không thể sinh

thành công ảnh đối kháng nào từ 500 ảnh dự đoán đúng đầu tiên. Hai nguyên nhân chính cho vấn đề này gồm sự hạn chế của việc duy trì trạng thái kích hoạt nơ-ron và sự hạn chế của bộ giải SMT.

Đối với vấn đề trạng thái kích hoạt nơ-ron của DeepCheck, trạng thái kích hoạt nơ-ron của ảnh đối kháng cần giống hệt trạng thái kích hoạt nơ-ron của ảnh dự đoán đúng. Chiến thuật duy trì trạng thái kích hoạt nơ-ron khiến cho DeepCheck có ít không gian tìm kiếm ảnh đối kháng hơn. Thực tế nhiều ảnh đối kháng có thể tạo thành công từ ảnh dự đoán đúng mà có trạng thái kích hoạt nơ-ron khác biệt. Ngoài ra, chiến thuật này không hỗ trợ kiểm thử mô hình nơ-ron truyền thẳng phức tạp. Nếu mô hình nơ-ron truyền thẳng có nhiều tầng hoặc nhiều nơ-ron, DeepCheck khó có thể sinh ảnh đối kháng có trạng thái kích hoạt nơ-ron giống với ảnh dự đoán đúng. Chỉ cần sự thay đổi nhỏ của giá trị một điểm ảnh thôi cũng có thể kéo theo sự thay đổi trạng thái của nhiều nơ-ron.

Đối với vấn đề tính hạn chế bộ giải SMT, DeepCheck sử dụng Z3 trong thực nghiệm để giải hệ ràng buộc. Nghiệm này tương ứng với một ảnh đối kháng. Chất lượng của DeepCheck phụ thuộc rất lớn vào khả năng của bộ giải Z3 hoặc các bộ giải SMT khác. Hệ ràng buộc cần giải có thể khá phức tạp do phải thỏa mãn tiêu chí duy trì trạng thái kích hoạt nơ-ron và phải đảm bảo nhãn ảnh đối kháng khác nhãn ảnh dự đoán đúng. Bởi vì sự phức tạp này nên bộ giải SMT có thể tiêu tốn chi phí rất lớn để tìm nghiệm thỏa mãn, có thể lên tới hàng chục phút chỉ để thêm nhiễu đối kháng vào một ảnh dự đoán đúng.

Do đó, để cải thiện hai vấn đề nêu trên của DeepCheck, luận án đề xuất HA4FNN. HA4FNN sử dụng bộ giải phỏng đoán thay vì bộ giải SMT và đơn giản hóa hệ ràng buộc bằng cách loại bỏ tiêu chí duy trì trạng thái kích hoạt nơ-ron. Để chứng minh hiệu quả của HA4FNN, luận án đã xây dựng một công cụ sinh ảnh đối kháng. Thực nghiệm trên MNIST, Fashion-MNIST và bộ chữ cái viết tay cho thấy hiệu năng và tỉ lệ thành công của HA4FNN cao hơn DeepCheck. Đối với trường hợp thêm nhiễu đối kháng vào một điểm ảnh, tỉ lệ thành công trung bình của DeepCheck là 0.7%. Ngược lại, tỉ lệ thành công trung bình của HA4FNN là 54.3%. Thời gian để tấn công một điểm ảnh của HA4FNN xấp xỉ DeepCheck trong tình huống này. Đối với trường hợp thêm nhiễu đối kháng vào nhiều điểm ảnh, tỉ lệ thành công trung bình của HA4FNN là 98.7%, tốt hơn hẳn DeepCheck với 16.9% khi sử dụng Z3 và 26.8% khi sử dụng SMTInterpol.

Ngoài ra, khi thêm nhiễu đối kháng vào một số ảnh dự đoán đúng, DeepCheck có thể cần tới hàng chục phút, trong khi đó HA4FNN chỉ cần tối đa vài giây trên cùng một cấu hình máy tính.

3.2. Các nghiên cứu liên quan

Phần này trình bày các nghiên cứu liên quan đến HA4FNN. Đầu tiên, luận án sẽ trình bày các phương pháp sinh ảnh đối kháng cho mô hình học sâu phân loại ảnh hiện nay. Luận án giới thiệu tổng quan về thực thi tượng trưng. Tổng quan về hướng sử dụng bộ giải SMT trong bài toán sinh ảnh đối kháng sẽ được làm rõ.

Sinh ảnh đối kháng: Nhiều phương pháp sinh ảnh đối kháng đã được đề xuất để đánh giá tính đúng đắn của mô hình nơ-ron truyền thẳng nói riêng và mô hình học sâu nói chung. Các phương pháp này sẽ thêm nhiễu đối kháng vào ảnh dự đoán đúng để sinh ảnh đối kháng. Trong đó, ảnh dự đoán đúng được phân lớp đúng còn ảnh đối kháng phân lớp sai.

Phương pháp HA4FNN và DeepCheck không sử dụng kĩ thuật dựa trên đạo hàm, vốn được sử dụng rộng rãi trong các kĩ thuật khác như FGSM [16], CW [17], ATN [18] và L-BFGS [32]. Hai phương pháp này đều xây dựng mã nguồn từ mô hình nơ-ron truyền thẳng và phân tích mã nguồn sử dụng kĩ thuật thực thi tượng trưng. Kết quả của quá trình thực thi tượng trưng là một hệ ràng buộc. Nghiệm của hệ ràng buộc tương ứng với ảnh đối kháng. Trong khi DeepCheck sử dụng bộ giải SMT để tìm nghiệm, HA4FNN sử dụng bộ giải phỏng đoán.

Độ đo khoảng cách L_p thường được sử dụng để đánh giá chất lượng ảnh đối kháng. Đối với trường hợp $p = 0$, các phương pháp phổ biến có thể kể đến CW L_0 [17], DeepCheck [67, 68] và NEUROSPF [69]. Trường hợp $p = 2$, CW L_2 [17], ATN [18], L-BFGS [32] và DeepFool [33] là những phương pháp tiêu biểu. Trường hợp $p = \infty$, các phương pháp phổ biến có thể kể đến FGSM [16], CW L_∞ [17] và BIS [19].

Độ đo khoảng cách L_p thường được sử dụng cùng với các độ đo khác. Pei và cộng sự [43] đề xuất độ đo độ phủ nơ-ron. Dựa theo nghiên cứu của Pei và cộng

sự, Ma và cộng sự [75] đề xuất một tập các độ đo khác như độ đo k-đoạn, độ phủ biên của nơ-ron và độ phủ hàm kích hoạt của nơ-ron, v.v. Lấy cảm hứng từ độ đo MC/DC trong kiểm thử truyền thống, Sun và cộng sự [20] đề xuất độ đo MC/DC cho mô hình học sâu.

Một vài phương pháp tấn công không sử dụng độ đo khoảng cách L_p để sinh ảnh đối kháng. Isola và cộng sự [76], Johnson và cộng sự [77] lập luận rằng độ đo khoảng cách L_p không đủ tốt để sinh ảnh đối kháng trông tự nhiên. Các phương pháp sử dụng độ đo L_p thêm nhiễu đối kháng vào các điểm ảnh trong ảnh dự đoán đúng để đạt được mục tiêu tấn công, tuy nhiên các điểm ảnh thường bị thêm nhiễu đối kháng vào rời rạc nên thường trông không giống thực tế. Xiao và cộng sự [78] đề xuất phương pháp biến đổi không gian để thay đổi vị trí các điểm ảnh thay vì thay đổi giá trị điểm ảnh. Nhóm tác giả này kì vọng rằng thuộc tính tự nhiên của ảnh sẽ được duy trì trong quá trình biến đổi. DeepTest [79] đề xuất phương pháp thêm nhiễu tự nhiên như nhiễu điều kiện thời tiết, các hạt bụi bay vào camera, v.v. vào ảnh dự đoán đúng.

Thực thi tượng trưng: Kĩ thuật này được sử dụng để phân tích trạng thái của chương trình [70]. Trong kiểm thử đơn vị truyền thống, kĩ thuật này hay được sử dụng để sinh ca kiểm thử tự động. Các công cụ kiểm thử tiêu biểu có thể kể đến gồm PathCrawler [80], DART [81], CUTE [82], CAUT [83] và KLEE [84]. Sau này, kĩ thuật thực thi tượng trưng được áp dụng để sinh ảnh đối kháng. Sun và cộng sự [85] đề xuất DeepConcolic - đây là công cụ đầu tiên kết hợp phân tích chương trình và thực thi tượng trưng để tăng độ phủ của mô hình học sâu. Gopinath và cộng sự [67, 68] đề xuất DeepCheck thêm nhiễu đối kháng vào một điểm ảnh hoặc hai điểm ảnh. Usman và cộng sự [69] áp dụng thực thi tượng trưng để phân tích tính chắc chắn của mô hình học sâu.

Bộ giải SMT: DeepCheck được phát triển dựa trên một thư viện thực thi tượng trưng sử dụng bộ giải Z3. Ý tưởng của DeepCheck có thể được sử dụng với các bộ giải SMT khác. Nói chung, các bộ giải SMT được sử dụng để giải hệ ràng buộc. Trong bối cảnh kiểm thử mô hình học sâu, để kiểm chứng tính đúng đắn của mô hình, sử dụng bộ giải là hướng phổ biến. Các thuộc tính cần kiểm chứng sẽ được biểu diễn thành hệ ràng buộc và bộ giải sẽ tìm nghiệm hệ ràng buộc đó. Nếu bộ giải không tìm được nghiệm thì mô hình học sâu không

thỏa mãn thuộc tính đang xem xét. Các bộ giải SMT phổ biến có thể kể đến Z3 [73], MathSat [86], Yices [87], SMTInterpol [74] và Reluplex [25]. Khác với DeepCheck, HA4FNN giải hệ ràng buộc bằng bộ giải phỏng đoán mà không sử dụng bộ giải SMT.

3.3. Phương pháp HA4FNN

DeepCheck có hai hạn chế gồm vấn đề duy trì trạng thái kích hoạt nơ-ron và vấn đề sử dụng bộ giải SMT. Đối với vấn đề đầu tiên, thực nghiệm DeepCheck cho thấy phương pháp này làm việc kém hiệu quả trên mô hình nơ-ron truyền thẳng có nhiều nơ-ron ẩn. Hệ ràng buộc c_{hidden} đảm bảo rằng ảnh đối kháng tương ứng với nghiệm sẽ có cùng trạng thái kích hoạt nơ-ron với ảnh dự đoán đúng. Nếu số lượng nơ-ron lớn, thêm nhiễu đối kháng vào một điểm ảnh sẽ dễ thay đổi trạng thái của nhiều nơ-ron. Do đó, nếu muốn điều kiện về trạng thái kích hoạt nơ-ron thoả mãn, giá trị điểm ảnh sẽ được thêm nhiễu đối kháng ít hơn. Nếu mô hình học đủ tốt thì sự thay đổi nhỏ giá trị điểm ảnh trên ảnh dự đoán đúng khó có thể sinh ảnh đối kháng thành công. Đối với vấn đề thứ hai, vì DeepCheck sử dụng Z3, hiệu năng của tấn công phụ thuộc vào khả năng bộ giải. Thực nghiệm tiến hành bởi luận án cho thấy bộ giải Z3 hoạt động kém hiệu quả khi số lượng điểm ảnh cần thêm nhiễu đối kháng đủ lớn.

Để cải thiện tỉ lệ thành công của DeepCheck, HA4FNN cần giải quyết hạn chế của việc sử dụng trạng thái kích hoạt nơ-ron và hạn chế của bộ giải SMT. Phương pháp HA4FNN bỏ qua sử dụng trạng thái kích hoạt nơ-ron và sử dụng một bộ giải thay thế gọi là bộ giải phỏng đoán. Thuật toán 3.1 trình bày tổng quan HA4FNN. Đầu vào gồm mô hình nơ-ron truyền thẳng $\mathbf{M}$, ảnh dự đoán đúng $\mathbf{x}$ và độ nhạy $k <= 0$. Đầu ra là một tập ảnh đối kháng được kí hiệu bởi *advs*.

Đầu tiên, tương tự như DeepCheck, HA4FNN tạo một mã nguồn C (kí hiệu p) từ mô hình $\mathbf{M}$ (dòng 1). Mã nguồn p sẽ được thêm các câu lệnh đánh dấu gọi là *marker* để sinh một mã nguồn mới (kí hiệu p_{ins}) có khả năng truy vết các câu lệnh thực thi (dòng 2). Các câu lệnh đánh dấu này giúp lưu lại các nhánh đúng/sai và câu lệnh đã được thực thi bởi mã nguồn p_{ins}. Sau đó, ảnh dự đoán

Thuật toán 3.1 : Phương pháp HA4FNN

Đầu vào: mô hình nơ-ron truyền thẳng $\mathbf{M}$, ảnh dự đoán đúng $\mathbf{x}$ và độ nhạy k ($k <= 0$)

Đầu ra: tập ảnh đối kháng $advs$

1: $p = \textsc{Translate}(\mathbf{M})$
2: $p_{ins} = \textsc{Instrument}(p)$ ▷ Chèn câu lệnh đánh dấu
3: $tp = \textsc{Execute}(p_{ins}, \mathbf{x})$
4: $t = \textsc{SymbolicExecution}(tp)$
5: $advs = \{\}$
6: **for** nhãn $i \neq y_{true}$ **do**
7: $\quad c' = t[n^{y_{true}}_{pre}] - t[n^{i}_{pre}] - k < 0$ ▷ Tạo hệ ràng buộc c'
8: $\quad \mathbf{x}' = \textsc{HeuristicSolver}(c', \mathbf{x})$ ▷ Gọi bộ giải đề xuất
9: $\quad$ **if** $\mathbf{x}'$ tồn tại **then**
10: $\quad\quad advs$.add($\mathbf{x}'$)
11: $\quad$ **end if**
12: **end for**
13: return $\textsc{FindValidAdvs}(advs)$

đúng $\mathbf{x}$ được chuyển thành đầu vào của mã nguồn p_{ins}. Mã nguồn này sẽ được biên dịch và thực thi để sinh ra đường thi hành (kí hiệu là tp) (dòng 3).

Trong bước kế tiếp, HA4FNN áp dụng thực thi tượng trưng trên đường thi hành để sinh ra hệ ràng buộc (dòng 4). Các nơ-ron ẩn được biểu diễn dưới dạng tượng trưng (gọi là nơ-ron tượng trưng) và lưu trong một bảng tượng trưng (kí hiệu t). Một nơ-ron ẩn được biểu diễn dưới dạng *(key, value)*, trong đó *key* đại diện cho tên định danh của nơ-ron ẩn và *value* đại diện giá trị trừu trượng của nơ-ron ẩn đó. Để lấy giá trị khóa *key*, thuật toán dùng cú pháp $t[key]$.

Xét ảnh dự đoán đúng $\mathbf{x}$, điểm ảnh thứ i được kí hiệu là n^i_0. Điểm ảnh tượng trưng tương ứng với n^i_0 được kí hiệu là f_i. Những điểm ảnh được thêm nhiễu đối kháng được gọi là điểm ảnh tượng trưng. Trong thuật toán này, tất cả mọi điểm ảnh đầu vào đều là điểm ảnh tượng trưng. Ví dụ, một ảnh chữ số $28 \times 28 \times 1$ trong bộ dữ liệu MNIST có 784 điểm ảnh tượng trưng f_0, f_1, ..., f_{783}. Sau khi áp dụng thực thi tượng trưng, nơ-ron ẩn n được biểu diễn như Công thức 3.1.

$$n = \sum_{i \in [0,d)} f_i \cdot a_i + z \tag{3.1}$$

trong đó, a_i và z là các hệ số được tính từ các trọng số giữa các tầng của mô hình kiểm thử.

Sau đó, xét tất cả mọi nhãn ngoại trừ y_{true}, phương pháp HA4FNN xây dựng một hệ ràng buộc (kí hiệu c') (dòng 7). Hệ ràng buộc c' được biểu diễn sử dụng các điểm ảnh đầu vào f_0, f_1, ..., f_{783}. Trong quá trình xây dựng c', phương pháp HA4FNN sử dụng độ nhạy $k <= 0$ để điều chỉnh chất lượng của tấn công. Nếu $k = 0$, số điểm ảnh bị thêm nhiễu đối kháng có xu hướng nhỏ nhất. Khi giảm giá trị k, số điểm ảnh bị thêm nhiễu đối kháng có xu hướng tăng lên. Trong bước kế tiếp, thay vì sử dụng bộ giải SMT, hệ ràng buộc c' được giải bằng bộ giải đề xuất (dòng 8). Cuối cùng, danh sách ảnh đối kháng ứng cử viên (kí hiệu là *advs*) được xác thực tính đúng đắn bằng cách đẩy vào mô hình kiểm thử **M** để kiểm tra lại (dòng 13). Nếu nhãn của một ảnh đối kháng ứng cử viên bị thay đổi, ảnh đối kháng đó được coi là hợp lệ.

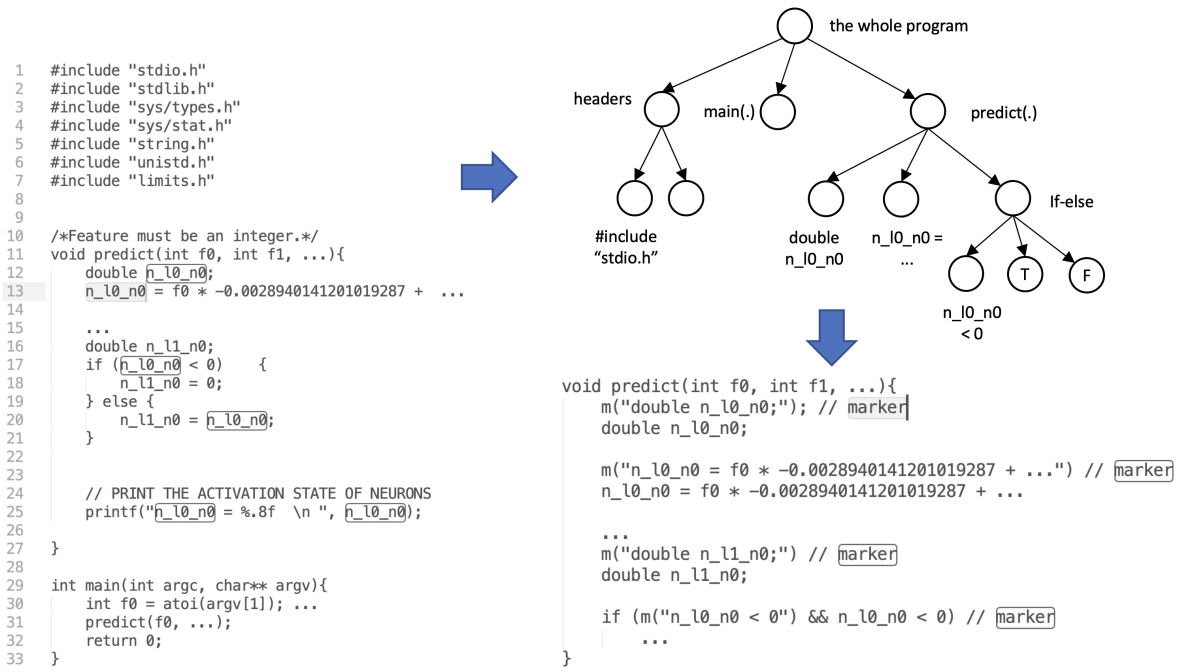
3.3.1. Sinh mã nguồn từ mô hình & Chèn câu lệnh đánh dấu

3.3.1.1. Sinh mã nguồn từ mô hình

Hàm TRANSLATE(.) phân tích mô hình nơ-ron truyền thẳng và chuyển đổi sang mã nguồn p viết bằng ngôn ngữ C. Mục đích là thực thi mã nguồn này với đầu vào là ảnh dự đoán đúng để thu thập được đường thi hành. Mã nguồn p bao gồm hai hàm chính gọi là MAIN(.) và PREDICT(.). Hàm MAIN(.) là điểm thực thi mã nguồn. Giá trị điểm ảnh được đẩy vào qua tham số dòng lệnh. Mã nguồn chỉ cần biên dịch một lần duy nhất và có thể thực thi với nhiều ảnh dự đoán đúng khác nhau, từ đó giảm thiểu chi phí tấn công đối kháng. Hàm PREDICT(.) thực thi ảnh đầu vào. Các tham số của hàm PREDICT(.) đại diện cho các điểm ảnh trong ảnh dự đoán đúng. Kiểu tham số là số nguyên với giá trị hợp lệ từ 0 đến 255. Hàm PREDICT(.) chứa bốn loại câu lệnh. Câu lệnh khai báo dùng để khai báo nơ-ron ẩn. Câu lệnh gán dùng để thay đổi giá trị nơ-ron ẩn. Khối lệnh `if-else` tương ứng với hàm kích hoạt ReLU. Trong đó, điều kiện của `if` đại diện cho phép so sánh giá trị nơ-ron ẩn với 0. Câu lệnh trích xuất xuất giá trị của nơ-ron ẩn khi mã nguồn thực thi xong, được dùng để xác định nơ-ron ẩn được kích hoạt và không được kích hoạt.

3.3.1.2. Chèn câu lệnh đánh dấu

Việc biên dịch và thực thi mã nguồn p sẽ không trả về danh sách câu lệnh hoặc nhánh đã viếng thăm. Do đó, mã nguồn p sẽ được thêm các câu lệnh đánh dấu để tạo thành mã nguồn đánh dấu p_{ins}. Nhiệm vụ của hàm INSTRUMENT(.) là chèn câu lệnh đánh dấu. Chỉ có hàm PREDICT(.) trong mã nguồn p sẽ được thêm các câu lệnh đánh dấu. Trong phần cài đặt công cụ, các câu lệnh đánh dấu chính là các câu lệnh trích xuất nội dung câu lệnh/nhánh ra một tệp cố định. Để thêm các câu lệnh đánh dấu vào mã nguồn p, HA4FNN chuyển mã nguồn này sang cây cú pháp trừu tượng (Abstract Syntax Tree - AST). Cây AST được duyệt để chèn các câu lệnh đánh dấu vào vị trí thích hợp. Kết quả của việc chèn các câu lệnh đánh dấu là mã nguồn p_{ins} có khả năng truy vết câu lệnh/nhánh được viếng thăm.



Hình 3.1: Minh họa một mã nguồn C trước và sau khi chèn các câu lệnh đánh dấu được kí hiệu bởi *marker*.

Hình 3.1 mô tả ví dụ về xây dựng mã nguồn từ mô hình và chèn câu lệnh đánh dấu. Mã nguồn C được đặt ở bên trái. Cây AST tương ứng được đặt ở trên cùng bên phải. Mã nguồn đánh dấu được đặt ở dưới cùng bên phải. Mã nguồn C có hàm PREDICT(.) và hàm MAIN(.). Dòng 12 khai báo nơ-ron ẩn *n_l0_n0*,

trong đó *l0* là tầng đầu tiên sau tầng đầu vào và *n0* đại diện nơ-ron đầu tiên trong tầng này. Tham số f_i tương ứng với điểm ảnh thứ i trong ảnh dự đoán đúng. Trong dòng 13, giá trị của nơ-ron ẩn *n_l0_n0* được thay đổi. Dòng 17 - 21 tương ứng với ReLU. Nếu *n_l0_n0* không được kích hoạt, giá trị nơ-ron tương ứng *n_l1_n0* trong tầng kế tiếp được gán bằng 0. Dòng 25 xuất giá trị nơ-ron ẩn ra tệp ngoài.

3.3.2. Thực thi tượng trưng

Nhiệm vụ của hàm SymbolicExecution(.) là thực thi tượng trưng đường thi hành để tạo bảng biến nơ-ron ẩn t. Mỗi bản ghi trong t được lưu dưới dạng *(key, value)*, trong đó *key* là định danh nơ-ron ẩn và *value* là giá trị nơ-ron ẩn đó. Tổng quan kĩ thuật thực thi tượng trưng được trình bày ở Thuật toán 3.2. Ý tưởng chính của thuật toán là phân tích sự thay đổi của các nơ-ron ẩn khi thực thi các câu lệnh từ câu lệnh đầu tiên đến câu lệnh cuối cùng của đường thi hành. Những sự thay đổi này được lưu lại trong bảng t.

Thuật toán 3.2 : Thực thi tượng trưng

Đầu vào: đường thi hành *tp*
Đầu ra: bảng biến nơ-ron ẩn t

1: $t = \emptyset$
2: **for** câu lệnh $stm \in tp$ **do**
3: s = ConstructAST(stm) ▷ Tạo AST
4: n_i^k = ExtractName(s) ▷ Lấy tên của biến
5: **if** s là câu lệnh khai báo **then**
6: v = ExtractValue(s) ▷ Lấy giá trị của biến
7: v' = Symbolize(v) ▷ Tượng trưng hóa
8: t.add(n_i^k, v')
9: **else if** s là câu lệnh gán **then**
10: v = ExtractValue(s)
11: v' = Symbolize(t, v)
12: $t[n_i^k] = v'$
13: **end if**
14: **end for**
15: **return** t

Đầu tiên, bảng t được khởi tạo rỗng (dòng 1). Sau đó, thuật toán phân tích từng câu lệnh của *tp* (dòng 2 - 14). Với từng câu lệnh, thuật toán chuyển thành AST (kí hiệu s) (dòng 3). Kế tiếp, thuật toán phân tích cây AST này để thu

được định danh của nơ-ron ẩn (kí hiệu là n_i^k) (dòng 4), trong đó i là chỉ số của tầng và k là chỉ số nơ-ron ẩn trên tầng đó. Bước này cần xử lý hai loại AST bao gồm khai báo và gán. Đối với loại khai báo (dòng 5), cây AST s là khai báo của một nơ-ron ẩn. Bảng t sẽ thêm một bản ghi với khóa là định danh nơ-ron ẩn được khai báo trong s với giá trị khởi tạo bằng v'.

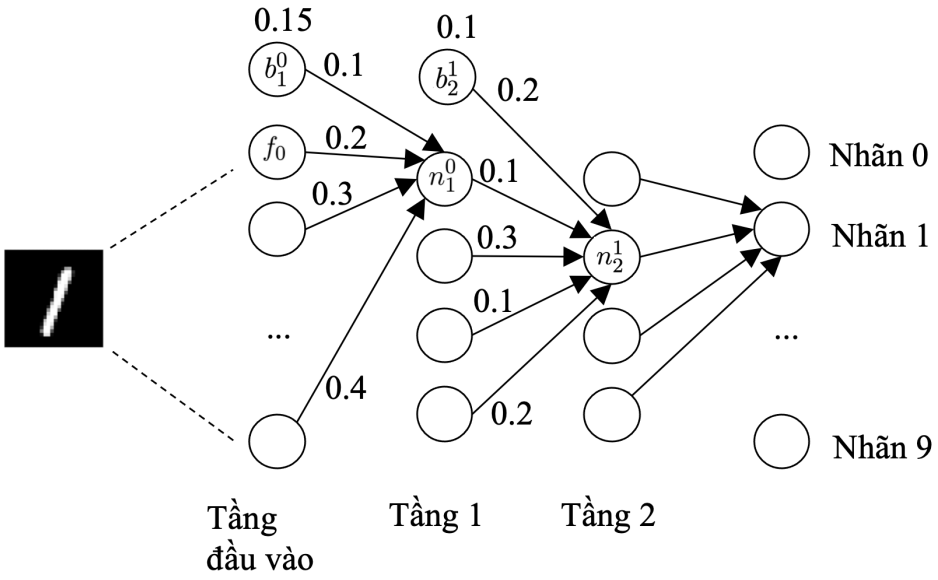
Đối với loại gán (dòng 9), cây AST s được biểu diễn dạng như Công thức 3.2.

$$n_i^k = \sum_{j \in [0,\, |L_{i-1}|)} w_{i-1,j,k} \cdot n_{i-1}^j + b_i \tag{3.2}$$

trong đó, $|\ L_{i-1}\ |$ là số nơ-ron ở tầng trước đó. Thuật toán biến đổi s thành dạng biểu diễn tượng trưng bằng cách sử dụng hàm SYMBOLIZE(.) như Công thức 3.3.

$$n_i^k = \sum_{j \in [0,d-1)} a_j \cdot f_j + z \tag{3.3}$$

trong đó, a_j và f_j là các hệ số. Trong Công thức 3.2, nơ-ron ẩn n_i^k được biểu diễn dưới dạng các nơ-ron ẩn ở tầng trước. Khác với Công thức 3.2, các nơ-ron ẩn trong Công thức 3.3 được viết dưới dạng các điểm ảnh tượng trưng. Biểu diễn theo Công thức 3.3 giúp xác định ảnh hưởng của từng điểm ảnh tượng trưng lên giá trị của nơ-ron ẩn n_i^k.



Hình 3.2: Ví dụ về cách tính giá trị nơ-ron từ các điểm ảnh tượng trưng.

Ví dụ, Hình 3.2 minh họa về cách tính Công thức 3.3. Mô hình mô tả là mô

hình nơ-ron truyền thẳng. Ảnh đầu vào lấy từ bộ dữ liệu MNIST có 784 điểm ảnh. Giá trị nơ-ron n_1^0 được biểu diễn như sau:

$$n_1^0 = 0.2 \cdot f_0 + 0.3 \cdot f_1 + ... + 0.4 \cdot f_{783} + 0.1 \cdot b_1^0 \tag{3.4}$$

Sau đó, giá trị n_2^1 được tính như sau:

$$n_2^1 = 0.1 \cdot n_1^0 + 0.3 \cdot n_1^1 + 0.1 \cdot n_1^2 + 0.2 \cdot n_1^3 + 0.2 \cdot b_2^1 \tag{3.5}$$

Kế tiếp, thực thi tượng trưng áp dụng Công thức 3.4 vào Công thức 3.5 như sau:

$$n_2^1 = 0.1 \cdot \underline{(0.2 \cdot f_0 + 0.3 \cdot f_1 + ... + 0.4 \cdot f_{783} + 0.1 \cdot b_1^0)} + 0.3 \cdot n_1^1 + 0.1 \cdot n_1^2 + 0.2 \cdot n_1^3 + 0.2 \cdot b_2^1 \tag{3.6}$$

Giá trị các nơ-ron còn lại trong Công thức 3.6 đều có thể biểu diễn qua các điểm ảnh tượng trưng f_0, f_1,..., f_{783}. Như vậy, nơ-ron n_2^1 có thể được biểu diễn như Công thức 3.3. Tổng quát hóa lên, tất cả mọi nơ-ron đều có thể biểu diễn theo các điểm ảnh tượng trưng.

3.3.3. Bộ giải phỏng đoán

Phần này trình bày HeuristicSolver(.). Điểm khác biệt giữa HA4FNN và DeepCheck là cách sử dụng bộ giải để tìm nghiệm của hệ ràng buộc t. Nghiệm tìm thấy tương ứng với một ảnh đối kháng. Cụ thể, DeepCheck sử dụng bộ giải SMT như Z3 [73] hoặc SMTInterpol [74]. Tuy nhiên, DeepCheck gặp hai vấn đề. Vấn đề đầu tiên là DeepCheck có tỉ lệ thành công thấp. Vấn đề thứ hai là hiệu năng của DeepCheck khi thêm nhiễu đối kháng vào nhiều điểm ảnh trên ảnh dự đoán đúng không tốt. Để giảm thiểu hai vấn đề này, HA4FNN sử dụng một bộ giải phỏng đoán. Thuật toán 3.3 mô tả các bước chính của bộ giải phỏng đoán. Đầu vào là hệ ràng buộc c' và ảnh dự đoán đúng $\mathbf{x}$. Đầu ra là một ảnh đối kháng $\mathbf{x}'$. Hệ ràng buộc c' được biểu diễn như Công thức 3.7.

$$\sum_{j \in [0, d-1)} a_j \cdot f_j + z < 0 \tag{3.7}$$

Thuật toán 3.3 : Bộ giải phỏng đoán

Đầu vào: hệ ràng buộc c' và ảnh dự đoán đúng $\mathbf{x}$
Đầu ra: ảnh đối kháng ứng cử viên $\mathbf{x}'$

1: $initValue$ = ComputeValue(c', $\mathbf{x}$) ▷ Tính giá trị hiện tại của c' từ ảnh $\mathbf{x}$
2: $\mathbf{x}' = \mathbf{x}$
3: **ins** = [] ▷ Lưu hướng thêm nhiễu đối kháng vào giá trị điểm ảnh tượng trưng
4: **s** = [] ▷ Lưu điểm số điểm ảnh tượng trưng
5: **idx** = $[0, 1, 2, ..., d-1]$ ▷ Lưu chỉ số điểm ảnh tượng trưng
6: **a** = $[a_0, a_1, ..., a_{d-1}]$ ▷ Lưu hệ số điểm ảnh tượng trưng
7: **for** từng điểm ảnh tượng trưng f_i **do**
8: **ins**$[i] = a_i < 0$
9: $s_i = \begin{cases} (255 - x'_i) \cdot \mid a_i \mid & \text{if } \mathbf{ins}[i] \\ x'_i \cdot a_i & \text{ngược lại} \end{cases}$ ▷ Tính điểm số của từng điểm ảnh
10: **end for**
11: **s**, **idx**, **ins** = SortByScore(**s**, **idx**, **ins**) ▷ Sắp xếp giảm dần theo điểm số
12: **for** $i = 0, i < d, i$++ **do**
13: **if ins**$[i]$ **then**
14: $x'_{\mathbf{idx}[i]} = 255$ ▷ Giá trị nguyên tối đa của một điểm ảnh
15: **else**
16: $x'_{\mathbf{idx}[i]} = 0$ ▷ Giá trị nguyên tối thiểu của một điểm ảnh
17: **end if**
18: $initValue$ -= s_i ▷ Cập nhật giá trị c'
19: **if** $initValue < 0$ **then** ▷ Nếu giá trị c' thỏa mãn thì kết thúc
20: break
21: **end if**
22: **end for**
23: **return** $\mathbf{x}'$

Gọi c_{left} kí hiệu vế trái của c'. Bộ giải phỏng đoán sẽ thêm nhiễu đối kháng vào tập các điểm ảnh trong $\mathbf{x}$ để c_{left} nhỏ hơn 0. Khi điều kiện này thỏa mãn, nhãn của ảnh đối kháng có xu hướng khác nhãn của ảnh dự đoán đúng.

Đầu tiên, thuật toán tính giá trị c_{left} từ ảnh dự đoán đúng $\mathbf{x}$ (dòng 1). Giá trị này là một số dương. Mảng **ins** lưu hướng thêm nhiễu đối kháng vào giá trị từng điểm ảnh tượng trưng là tăng lên hoặc giảm đi (dòng 3). Việc tăng hoặc giảm giá trị ban đầu của điểm ảnh tượng trưng f_i sẽ thay đổi giá trị của c_{left}. Mục tiêu của thuật toán là thêm nhiễu đối kháng vào điểm ảnh tượng trưng f_i để giảm giá trị c_{left} nhiều nhất có thể. Nếu **ins**$[i] = true$, thuật toán sẽ tăng giá trị điểm ảnh tượng trưng f_i lên giá trị cực đại là 255. Ngược lại, nếu **ins**$[i] = false$, thuật toán sẽ giảm giá trị điểm ảnh tượng trưng f_i xuống giá trị cực tiểu là 0. Mảng **s** lưu điểm số của các điểm ảnh tượng trưng (dòng 4). Điểm số của điểm

ảnh tượng trưng f_i đại diện cho mức độ thay đổi $initValue$ khi f_i thay đổi giá trị của nó. Mảng lưu chỉ số của các điểm ảnh tượng trưng kí hiệu là **idx** (dòng 5). Mảng **a** lưu hệ số của các điểm ảnh tượng trưng (dòng 6).

Sau bước khởi tạo, thuật toán đánh giá ảnh hưởng của điểm ảnh tượng trưng f_i lên giá trị c_{left} (dòng 8). Nếu hệ số a_i của điểm ảnh tượng trưng f_i nhỏ hơn 0, f_i sẽ được tăng giá trị. Trong trường hợp này, **ins**$[i]$ được gán là *true*. Đối với trường hợp ngược lại, **ins**$[i]$ được gán *false*. Sau đó, thuật toán tính điểm số của tất cả điểm ảnh tượng trưng (dòng 9). Giá trị điểm ảnh tượng trưng sẽ được thay đổi về 0 hoặc 255 với mục tiêu làm giảm giá trị c_{left}. Nếu điểm số của f_i bằng 0, giá trị f_i sẽ được giữ nguyên vì f_i không tác động đến giá trị c_{left}. Thuật toán sắp xếp các điểm ảnh tượng trưng theo chiều giảm dần điểm số (dòng 11) và lưu trong mảng **s**.

Dựa theo danh sách điểm số đã sắp xếp **s**, thuật toán thêm nhiễu đối kháng vào các điểm ảnh tượng trưng quan trọng nhất (dòng 12 - 22). Đây là các điểm ảnh có điểm số cao nhất. Khi một điểm ảnh tượng trưng được cập nhật, phương pháp HA4FNN sẽ tính lại giá trị của c_{left} (dòng 18). Nếu $c_{left} > 0$, thuật toán tiếp tục thêm nhiễu đối kháng vào điểm ảnh tượng trưng quan trọng kế tiếp. Nếu $c_{left} < 0$, thuật toán trả về ảnh đối kháng ứng cử viên (dòng 19). Ở đây, ảnh đối kháng được gọi là ứng cử viên vì sau đó mô hình cần xác nhận tính đúng đắn của ảnh đối kháng này. Nếu nhãn của ảnh đối kháng khác nhãn của ảnh dự đoán đúng, ảnh đối kháng được coi là hợp lệ và thuật toán kết thúc.

3.4. Thực nghiệm

Thực nghiệm so sánh HA4FNN với DeepCheck. Môi trường thực nghiệm là Macbook Pro M1 16GB RAM. Thực nghiệm trả lời các câu hỏi nghiên cứu sau đây:

- **RQ1 - Sửa một điểm ảnh**: Đánh giá tỉ lệ thành công của phương pháp HA4FNN so sánh với DeepCheck khi chỉ thêm nhiễu đối kháng vào một điểm ảnh trên ảnh dự đoán đúng để sinh ảnh đối kháng?
- **RQ2 - Sửa nhiều điểm ảnh**: Đánh giá tỉ lệ thành công của phương pháp

HA4FNN so sánh với DeepCheck khi thêm nhiễu đối kháng vào nhiều điểm ảnh trên ảnh dự đoán đúng để sinh ảnh đối kháng?

- **RQ3 - Hiệu năng**: Đánh giá hiệu năng của phương pháp HA4FNN?

3.4.1. Cấu hình

3.4.1.1. Mô hình kiểm thử

Bảng 3.1 trình bày thông tin về các mô hình kiểm thử. Các mô hình này được học sử dụng bộ thư viện Keras[1]. Cột 2 và 3 mô tả độ chuẩn xác trên tập học và tập kiểm thử. Thành phần đầu tiên và thành phần cuối cùng trong cột *Kiến trúc* tương ứng với số nơ-ron tầng đầu vào và tầng đầu ra. Hàm kích hoạt ở tầng đầu ra là softmax. Các mô hình kiểm thử này sử dụng hàm kích hoạt ReLU trong các tầng ẩn để giảm tính phi tuyến của hệ ràng buộc, từ đó bộ giải SMT sẽ tìm được nghiệm dễ dàng hơn. Mô hình M4 và F4 được sử dụng trong thí nghiệm của DeepCheck. Cấu hình để học các mô hình kiểm thử giống hệt nhau. Hàm mục tiêu sử dụng cross-entropy. Bộ tối ưu hóa là Adam [88]. Tốc độ học là 0.001. Các mô hình được học với 300 lần lặp.

3.4.1.2. Chọn ảnh dự đoán đúng

Từ 500 ảnh đầu tiên của tập học trên từng bộ dữ liệu, thực nghiệm chọn những ảnh được dự đoán đúng bởi mô hình kiểm thử. Bảng 3.2 mô tả những ảnh này. Tập các ảnh được chọn khá cân bằng về nhãn. Vì thế, thực nghiệm có thể đánh giá hiệu quả của HA4FNN trên mọi nhãn. Thực nghiệm không tiến hành trên các ảnh được dự đoán đúng trên tập kiểm thử bởi vì các ảnh này có xu hướng dễ thêm nhiễu đối kháng hơn các ảnh thuộc tập học.

Khi xây dựng mô hình kiểm thử, mô hình sẽ được cấu hình để học tốt các đặc trưng của tập học. Mục tiêu thực nghiệm là cho dù mô hình được học tốt, mô hình vẫn có thể dễ dàng nhận diện sai nhãn khi có sự chỉnh sửa nhỏ trên tập học. Ví dụ, thêm nhiễu đối kháng vào một điểm ảnh của ảnh số một trong

[1] https://keras.io

Bảng 3.1: Mô tả các mô hình kiểm thử

Bộ dữ liệu	Mô hình	Tập học	Tập kiểm thử	Kiến trúc
MNIST	M1	100.0%	98.3%	784, 128 ReLU, 64 ReLU, 32 ReLU, 16 ReLU, 10 softmax
	M2	99.9%	96.6%	784, 32 ReLU, 16 ReLU, 10 softmax
	M3	98.2%	95.1%	784, 16 ReLU, 10 softmax
	M4	96.8%	94.4%	784, 10 ReLU, 10 ReLU, 10 ReLU, 10 softmax
Fashion-MNIST	F1	99.6%	89.5%	784, 128 ReLU, 64 ReLU, 32 ReLU, 16 ReLU, 10 softmax
	F2	95.7%	86.5%	784, 32 ReLU, 16 ReLU, 10 softmax
	F3	92.2%	85.8%	784, 16 ReLU, 10 softmax
	F4	89.6%	85.4%	784, 10 ReLU, 10 ReLU, 10 ReLU, 10 softmax
Bộ chữ cái viết tay	A1	98.4%	97.5%	784, 128 ReLU, 64 ReLU, 32 ReLU, 16 ReLU, 26 softmax
	A2	96.0%	94.8%	784, 32 ReLU, 16 ReLU, 26 softmax
	A3	91.7%	90.9%	784, 16 ReLU, 26 softmax
	A4	88.1%	87.6%	784, 10 ReLU, 10 ReLU, 10 ReLU, 26 softmax

tập học có thể khiến mô hình nhận diện thành số chín. Hai số này có hình dạng rất khác nhau.

Bảng 3.2: Thống kê ảnh dự đoán đúng dùng để kiểm tra tính chắc chắn của mô hình kiểm thử

Mô hình	#ảnh dự đoán đúng	Mô hình	#ảnh dự đoán đúng	Mô hình	#ảnh dự đoán đúng
M1	500	F1	496	A1	490
M2	499	F2	483	A2	472
M3	491	F3	460	A3	447
M4	482	F4	442	A4	434

3.4.1.3. Cấu hình bộ giải

Để đảm bảo tính công bằng, DeepCheck và HA4FNN sử dụng các mô-đun giống hệt nhau ngoại trừ mô-đun giải hệ ràng buộc. Các mô-đun dùng chung gồm mô-đun chuyển đổi mô hình kiểm thử thành mã nguồn, mô-đun chèn câu lệnh vào mã nguồn và mô-đun thực thi tượng trưng. Trong mô-đun giải hệ ràng

buộc, HA4FNN sử dụng bộ giải phỏng đoán. Độ nhạy k trong Thuật toán 3.1 được lấy lần lượt trong tập $\{0, -5, -10, -20, -50\}$.

DeepCheck sử dụng bộ giải SMT. Thực nghiệm đánh giá hai bộ giải SMT phổ biến gồm bộ giải Z3 [73] và SMTInterpol [74]. Với bộ giải Z3, kí hiệu Z3@n ám chỉ n điểm ảnh quan trọng nhất trong ảnh dự đoán đúng sẽ được thêm nhiễu đối kháng để sinh ảnh đối kháng. Ví dụ, Z3@1 có nghĩa chỉ điểm ảnh quan trọng nhất sẽ được thêm nhiễu đối kháng để sinh ảnh đối kháng. Với bộ giải SMTInterpol, thực nghiệm sử dụng kí hiệu SMTInterpol@n với ý nghĩa tương tự.

3.4.1.4. Tính đúng đắn của bộ cài đặt DeepCheck

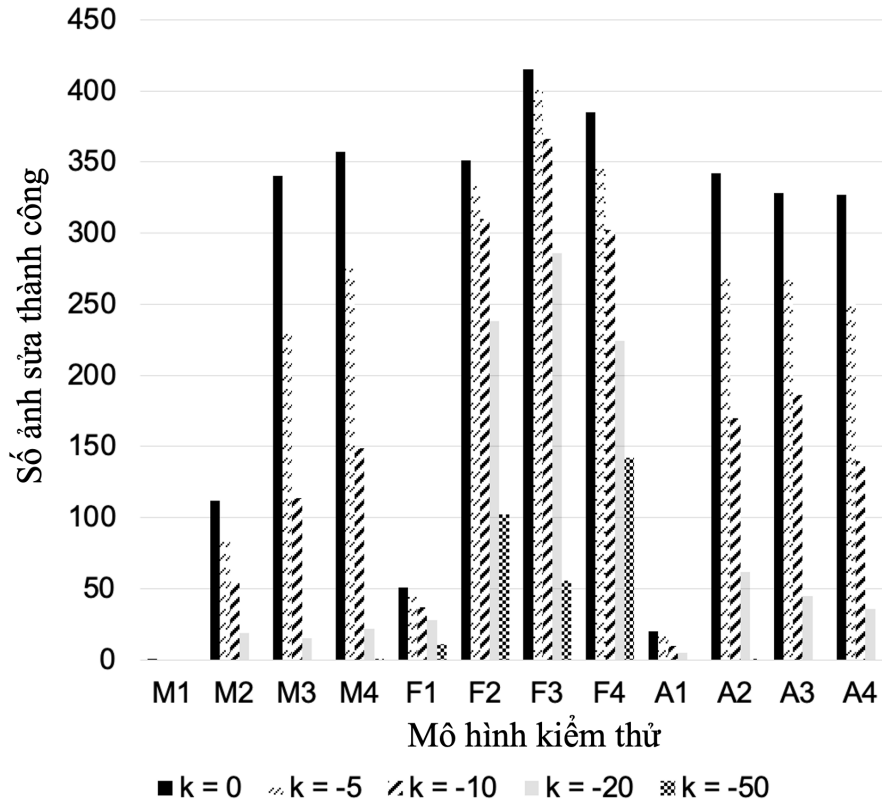
Vì mã nguồn DeepCheck không có sẵn, luận án đã cài đặt ý tưởng của nghiên cứu này bằng ngôn ngữ Java. Thực nghiệm đã thực hiện các phép đánh giá để đảm bảo tính đúng đắn của bộ cài đặt này. Từ mô hình kiểm thử, chương trình C được sinh ra. Các nơ-ron ở các tầng trong mô hình kiểm thử tương ứng với các biến trung gian trong chương trình này. Giá trị các biến này khi thực thi chương trình và thực thi mô hình kiểm thử cần giống hệt nhau.

3.4.2. Kết quả

3.4.2.1. RQ1 - Sửa một điểm ảnh

Phần thực nghiệm này đánh giá hiệu quả của HA4FNN khi thêm nhiễu đối kháng vào một điểm ảnh trong ảnh dự đoán đúng để sinh ảnh đối kháng. Mục tiêu là với từng ảnh dự đoán đúng $\mathbf{x}$, ảnh đối kháng $\mathbf{x}'$ cần thỏa mãn $L_0(\mathbf{x}, \mathbf{x}') = 1$. Những ảnh dự đoán đúng này được mô tả trong Bảng 3.2. Với DeepCheck, thực nghiệm sử dụng hai cấu hình Z3@1 và SMTInterpol@1 vì hai cấu hình này hỗ trợ thêm nhiễu đối kháng vào một điểm ảnh. Kết quả thực nghiệm cho thấy HA4FNN có tỉ lệ thành công tốt hơn hẳn Z3@1 and SMTInterpol@1.

Ảnh hưởng của độ nhạy k: Trước khi đi sâu vào so sánh chi tiết, thực nghiệm cần chọn giá trị độ nhạy k phù hợp. Như đã trình bày trong Thuật toán 3.2, giá



Hình 3.3: Số ảnh dự đoán đúng được thêm nhiễu đối kháng vào một điểm ảnh.

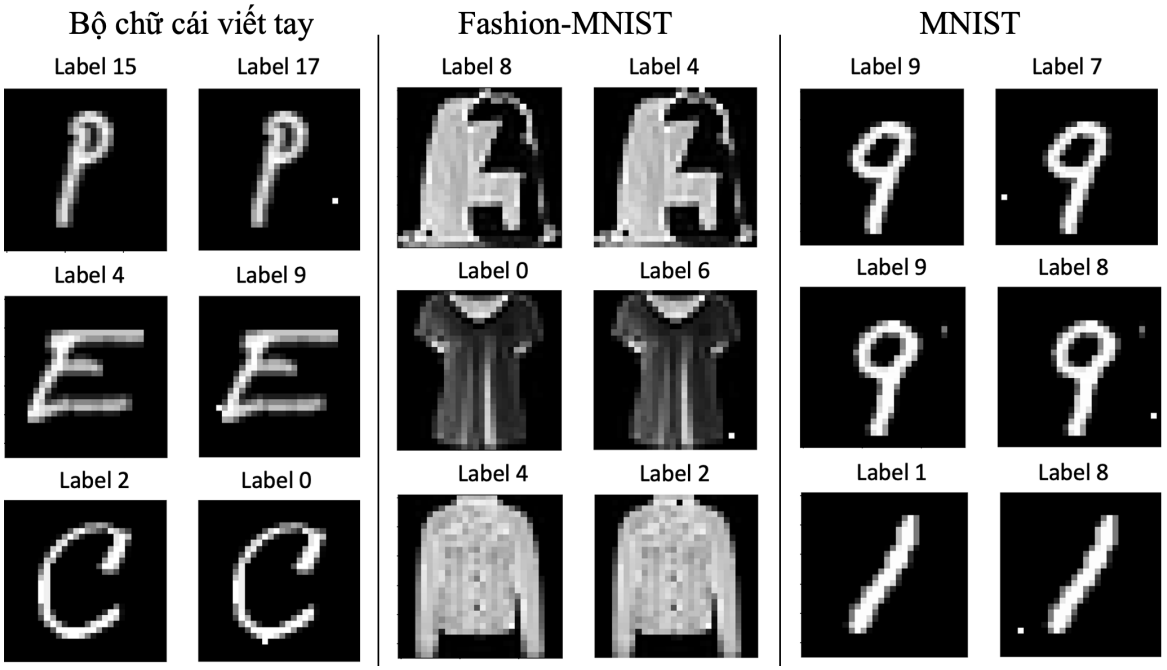
trị lớn nhất của k là 0. Thực nghiệm quan sát thấy rằng tăng giá trị độ nhạy k có xu hướng tăng tỉ lệ thành công. Cụ thể, Hình 3.3 mô tả ảnh hưởng của độ nhạy k lên số ảnh dự đoán đúng được thêm nhiễu đối kháng thành công. Hình này cho thấy sử dụng độ nhạy $k = 0$ có xu hướng cho số ảnh dự đoán đúng thêm nhiễu đối kháng vào một điểm ảnh thành công lớn nhất. Độ nhạy thấp nhất trong tập so sánh $k = -50$ cho số ảnh dự đoán đúng thêm nhiễu đối kháng vào một điểm ảnh thành công nhỏ nhất. Vì thế, giá trị độ nhạy k trong thực nghiệm này được chọn bằng 0.

Tỉ lệ thành công: Với độ nhạy $k = 0$, tỉ lệ thành công của phương pháp HA4FNN cao hơn hẳn DeepCheck và được thể hiện trong Bảng 3.3. Ngoại trừ M1, F1 và A1, tỉ lệ thành công của phương pháp HA4FNN trong khoảng từ 22.44% đến 90.22%. Trong khi đó, tỉ lệ thành công của Z3@1 và SMTInterpol@1 cao nhất là 3.04%. Nguyên nhân là do M1, F1, và A1 có kiến trúc nhiều tầng nhất và nhiều nơ-ron hơn các mô hình còn lại. Mô hình càng phức tạp thì càng khó sửa chỉ một điểm ảnh trên ảnh dự đoán đúng để sinh ảnh đối kháng. Hình 3.4

mô tả một vài ví dụ về tấn công một điểm ảnh. Z3@1 và SMTInterpol@1 không thể thêm nhiễu đối kháng vào những ảnh dự đoán đúng này.

Bảng 3.3: So sánh tỉ lệ thành công khi thêm nhiễu đối kháng vào một điểm ảnh

Mô hình	HA4FNN	Z3@1/ SMTInterpol@1
M1	**0.20%**	0.00%
M2	**22.44%**	0.00%
M3	**69.25%**	0.41%
M4	**74.07%**	0.00%
F1	**10.28%**	0.20%
F2	**72.67%**	1.66%
F3	**90.22%**	3.04%
F4	**87.10%**	1.81%
A1	**4.08%**	0.00%
A2	**72.46%**	0.42%
A3	**73.38%**	0.45%
A4	**75.35%**	0.46%
Trung bình	**54.29%**	0.70%



Hình 3.4: Ví dụ một vài ảnh dự đoán đúng được thêm nhiễu đối kháng vào một điểm ảnh thành công (bên trái) và ảnh đối kháng tương ứng (bên phải).

Giải thích: Hai nguyên nhân chính cho kết quả tốt hơn của HA4FNN gồm hạn chế của bộ giải SMT và vấn đề sử dụng trạng thái kích hoạt nơ-ron. Bảng 3.4 làm rõ hai nguyên nhân này. Cột *Cùng mẫu* thể hiện số ảnh dự đoán đúng mà trạng thái kích hoạt nơ-ron giống ảnh đối kháng. Cột *Khác mẫu* thể hiện số ảnh dự đoán đúng mà trạng thái kích hoạt nơ-ron khác ảnh đối kháng. Trong HA4FNN, thêm nhiễu đối kháng vào một điểm ảnh của một ảnh dự đoán đúng có thể sinh ra ít nhất một ảnh đối kháng. So sánh với ảnh dự đoán đúng, ảnh đối kháng có thể cùng hoặc khác trạng thái kích hoạt nơ-ron. Trong khi đó, Z3@1 và SMTInterpol@1 chỉ có thể sinh ra một ảnh đối kháng mà trạng thái kích hoạt nơ-ron của nó giống trạng thái kích hoạt nơ-ron của ảnh dự đoán đúng.

Bảng 3.4: Số ảnh dự đoán đúng thêm nhiễu đối kháng vào một điểm ảnh thành công

Mô hình	HA4FNN		Z3@1/SMTInterpol@1
	Khác mẫu	Cùng mẫu	(Cùng mẫu)
M1	1	0	0
M2	110	6	0
M3	299	137	2
M4	351	28	0
F1	51	0	1
F2	351	1	8
F3	408	28	14
F4	383	5	8
A1	20	0	0
A2	341	9	2
A3	283	130	2
A4	320	56	2
Trung bình	243	33	3

Về hạn chế của bộ giải SMT, phân tích này sẽ chọn những ảnh dự đoán đúng có trạng thái kích hoạt nơ-ron giống ảnh đối kháng. Đây là tiêu chí mà DeepCheck sử dụng để sinh ảnh đối kháng. Do đó, thực nghiệm sẽ đánh giá công bằng về tỉ lệ thành công của HA4FNN và DeepCheck. Với HA4FNN, kết quả cấu hình này được trình bày trong cột *Cùng mẫu*. Kết quả DeepCheck được trình bày trong cột Z3@1/SMTInterpol@1. Trong khi HA4FNN thêm nhiễu đối kháng trung bình 33 ảnh dự đoán đúng thành công, DeepCheck thêm nhiễu đối kháng thành công trung bình ba ảnh dự đoán đúng. Do đó, sử dụng bộ giải

SMT như Z3 và SMTInterpol có thể không phải một lựa chọn tốt để sinh ảnh đối kháng với cùng trạng thái kích hoạt nơ-ron.

Về hạn chế của trạng thái kích hoạt nơ-ron, phân tích này sẽ chọn những ảnh dự đoán đúng có trạng thái kích hoạt nơ-ron khác ảnh đối kháng. Cột *Khác mẫu* thể hiện số ảnh dự đoán đúng này. Kết quả cho thấy HA4FNN có thể thêm nhiễu đối kháng thành công trung bình 243 ảnh dự đoán đúng để sinh ảnh đối kháng. Ngược lại, DeepCheck không xét trường hợp sinh ảnh đối kháng khác trạng thái kích hoạt nơ-ron, từ đó cũng làm giảm tỉ lệ thành công như đã thể hiện trong Bảng 3.3. Tỉ lệ thành công của hai phiên bản Z3@1 và SMTInterpol@1 bằng nhau.

3.4.2.2. RQ2 - Sửa nhiều điểm ảnh

Trong thực tế, kẻ tấn công có thể muốn sinh ra nhiều ảnh đối kháng nhất có thể và không quan tâm đến tiêu chí chỉ thêm nhiễu đối kháng vào duy nhất một điểm ảnh. RQ2 sẽ trả lời hiệu quả của HA4FNN trong trường hợp này. Khoảng cách L_0 nhỏ nhất giữa ảnh đối kháng và ảnh dự đoán đúng bằng một. Thực nghiệm cho thấy tỉ lệ thành công của phương pháp HA4FNN cao hơn hẳn tỉ lệ thành công của DeepCheck.

Thực nghiệm cho thấy sử dụng giá trị độ nhạy k khác nhau thường thêm nhiễu đối kháng vào ảnh dự đoán đúng từ 1 đến khoảng 20 điểm ảnh. Do đó, để đánh giá được công bằng, DeepCheck sẽ chọn thêm nhiễu đối kháng vào top-n điểm ảnh quan trọng nhất với $n \in \{1, 20, 40\}$ sử dụng hai bộ giải Z3 và SMTInterpol. Kết quả được thể hiện ở Bảng 3.5. Giá trị tốt hơn được in đậm. Tỉ lệ thành công của phương pháp HA4FNN tăng lên khi độ nhạy k giảm từ 0 xuống -50. Ví dụ, với M1, trong khi sử dụng $k = 0$ chỉ đạt tỉ lệ thành công 75.4%, sử dụng $k = -50$ có tỉ lệ thành công 99.6%. Nguyên nhân chính là do khi giá trị k giảm, nhiều điểm ảnh hơn sẽ bị thêm nhiễu đối kháng trong Thuật toán 3.3. Khoảng cách L_0 giữa ảnh dự đoán đúng và ảnh đối kháng sẽ tăng lên, hay nói theo cách khác, ảnh dự đoán đúng bị thêm nhiễu đối kháng nhiều hơn. Từ đó, mô hình kiểm thử dễ nhận diện sai nhãn ảnh đối kháng.

Bảng 3.5: Tỉ lệ thành công của khi thêm nhiễu đối kháng vào nhiều điểm ảnh

Mô hình	HA4FNN					Z3			SMTInterpol		
	k=0	k=-5	k=-10	k=-20	k=-50	@1	@20	@40	@1	@20	@40
M1	75.4%	83.8%	90.4%	95.0%	**99.6%**	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
M2	99.2%	99.6%	100.0%	100.0%	**100.0**%	0.0%	**6.01**%	1.60%	0.0%	2.40%	**15.83%**
M3	99.8%	99.8%	99.8%	99.8%	**100.0**%	0.4%	14.7%	**41.8**%	0.4%	5.3%	**33.4%**
M4	99.6%	100.0%	100.0%	100.0%	**100.0**%	0.0%	10.0%	**18.7**%	0.0%	3.3%	**24.5%**
F1	49.8%	54.2%	60.1%	69.0%	**84.9**%	0.2%	0.0%	0.0%	0.20%	0.20%	0.0%
F2	97.5%	97.9%	98.3%	99.2%	**99.6**%	1.7%	**25.5**%	3.7%	1.7%	**35.6**%	28.8%
F3	97.8%	98.3%	98.5%	99.1%	**99.8**%	3.0%	52.4%	**56.3**%	3.0%	35.4%	**63.5%**
F4	97.3%	98.4%	98.4%	99.3%	**100.0**%	1.8%	**35.5**%	26.7%	1.8%	26.0%	**50.9%**
A1	97.4%	97.8%	97.8%	99.0%	**100.0**%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
A2	98.9%	99.2%	100.0%	100.0%	**100.0**%	0.4%	**5.1**%	0.2%	0.4%	8.7%	**25.6%**
A3	100.0%	100.0%	100.0%	100.0%	**100.0**%	0.5%	0.0%	**36.7**%	0.5%	8.5%	**35.4%**
A4	100.0%	100.0%	100.0%	100.0%	**100.0**%	0.5%	**20.7**%	16.6%	0.5%	15.7%	**43.6%**
TB	92.7%	94.1%	95.3%	96.7%	**98.7**%	0.7%	14.2%	**16.9**%	0.7%	11.8%	**26.8%**

3.4.2.3. RQ3 - Hiệu năng của thực nghiệm thêm nhiễu đối kháng vào nhiều điểm ảnh

Thực nghiệm này đánh giá hiệu năng của RQ2. Phương pháp HA4FNN và DeepCheck sử dụng ba mô-đun chung gồm mô-đun sinh mã nguồn từ mô hình, mô-đun chèn mã nguồn đánh dấu và mô-đun thực thi tượng trưng. Mô-đun khác biệt duy nhất giữa hai công cụ là bộ giải. Nhờ đó, thực nghiệm sẽ làm rõ được tính hiệu quả của bộ giải trong HA4FNN.

Bảng 3.6 cho thấy hiệu năng của bộ giải đề xuất có thời gian chạy trung bình nhanh hơn Z3 và SMTInterpol. Kí hiệu "-" ám chỉ thời gian chạy trên 40 phút. Giá trị tốt hơn được in đậm. Trong khi bộ giải đề xuất cần trung bình 0.40 giây để thêm nhiễu đối kháng vào một ảnh dự đoán đúng, SMTInterpol@1 và SMTInterpol@20 cần trung bình 0.51 giây và 1.07 giây. Đối với Z3@1, Z3@20, Z3@40 và SMTInterpol@40, thời gian chạy trung bình vượt ngưỡng 40 phút cho tấn công một ảnh dự đoán đúng.

Bảng 3.6: Thời gian trung bình (giây) để giải một hệ ràng buộc

Mô hình	HA4FNN	Z3			SMTInterpol		
		@1	@20	@40	@1	@20	@40
M1	**0.34**	1.62	-	-	1.07	1.44	4.51
M2	**0.32**	0.39	2.73	-	0.34	0.48	1.70
M3	0.31	**0.19**	0.59	2.38	0.21	0.30	0.56
M4	0.34	0.28	0.95	4.21	**0.26**	0.38	0.88
F1	**0.33**	-	-	-	1.14	4.31	-
F2	**0.34**	0.41	3.49	-	**0.34**	1.06	4.46
F3	0.36	**0.21**	0.86	2.98	0.22	0.38	0.82
F4	0.37	0.32	1.79	4.92	**0.29**	0.55	1.45
A1	**0.50**	1.05	-	-	1.15	2.21	-
A2	0.52	**0.31**	5.09	-	0.43	0.72	3.44
A3	0.54	**0.20**	1.43	4.31	0.28	0.45	0.79
A4	0.55	**0.26**	2.82	6.04	0.38	0.61	1.40
Trung bình	**0.40**	-	-	-	0.51	1.07	-

3.5. Thảo luận

Ảnh hưởng của hàm kích hoạt: Nghiên cứu này chỉ kiểm thử mô hình nơ-ron truyền thẳng sử dụng hàm kích hoạt ReLU ở các tầng ẩn. Tuy nhiên, trong thực tế mô hình có thể sử dụng các hàm kích hoạt khác như tanh, GELU [89], ELU [90], v.v. Các hàm kích hoạt này có tính phi tuyến cao hơn ReLU. Sử dụng hàm kích hoạt phi tuyến làm tăng độ phức tạp của thực thi tượng trưng. Phương pháp HA4FNN và DeepCheck chưa hỗ trợ tốt các hàm kích hoạt phi tuyến này.

GPU hoặc CPU: Thực thi tượng trưng không cần sử dụng GPU. Nguyên nhân chính là do thực thi tượng trưng không thực hiện phép toán ma trận. Thay vào đó, sử dụng nhiều CPU sẽ tăng hiệu năng của thực thi tượng trưng. Thực nghiệm mới thực hiện trên một luồng. Nếu số CPU tăng lên, hiệu năng của HA4FNN và DeepCheck sẽ được cải thiện.

Dropout: Singh và cộng sự [91] đề xuất lớp Dropout để hạn chế vấn đề học quá khớp của mô hình học sâu. Tư tưởng lớp Dropout là các nơ-ron được dùng

để học mô hình học sâu với xác suất nào đó. Tại lần lặp thứ i, nơ-ron n có thể được dùng. Nhưng trong một lần lặp khác, nơ-ron n này có thể bị ẩn đi, tức là không tham gia vào quá trình học. Phương pháp HA4FNN chưa hỗ trợ tấn công mô hình nơ-ron truyền thẳng sử dụng lớp Dropout.

Thực thi tượng trưng: Một trong những hạn chế của HA4FNN và DeepCheck là chi phí thực thi tượng trưng. Nếu mô hình kiểm thử tăng số tầng và số nơ-ron, mã nguồn tương ứng sẽ có nhiều câu lệnh hơn. Khi thực thi chương trình, số câu lệnh được thực thi sẽ tăng lên. Tư tưởng của thực thi tượng trưng là phân tích từng câu lệnh được thực thi để mô phỏng hóa sự thay đổi các biến (tương ứng với các nơ-ron). Do đó, chi phí thực thi tượng trưng tỉ lệ thuận với số tầng và số nơ-ron. Trong thực nghiệm, tấn công M1 cần trung bình 1.5 giây để thực hiện thực thi tượng trưng. Hiện nay, mô hình phân loại ảnh thường sử dụng mô hình tích chập thay vì mô hình nơ-ron truyền thẳng. Kiến trúc một mô hình tích chập điển hình gồm tầng tổng hợp, tầng tích chập, các tầng mô hình nơ-ron truyền thẳng, tầng đầu vào và tầng đầu ra. Sự phức tạp của kiến trúc mô hình tích chập làm tăng độ phức tạp của mã nguồn C, từ đó tăng chi phí thực thi tượng trưng.

3.6. Tổng kết

Luận án đã trình bày cải tiến trong DeepCheck và cài đặt trong công cụ HA4FNN. Mô hình kiểm thử là mô hình nơ-ron truyền thẳng sử dụng hàm kích hoạt ReLU. Tư tưởng chính của HA4FNN là xây dựng mã nguồn C từ mô hình kiểm thử. Sau đó, mã nguồn này được chèn các câu lệnh đánh dấu để truy vết câu lệnh được viếng thăm khi thực thi mã nguồn. Với một ảnh dự đoán đúng, mã nguồn được biên dịch và thực thi để tìm đường thi hành. Thực thi tượng trưng được áp dụng trên đường thi hành để xây dựng hệ ràng buộc. Sau đó, hệ ràng buộc được giải bằng bộ giải phỏng đoán.

Thực nghiệm được tiến hành trên ba mô hình nơ-ron truyền thẳng với ba bộ dữ liệu MNIST, Fashion-MNIST và chữ cái viết tay. Kết quả đầu tiên cho thấy tỉ lệ thành công trung bình của HA4FNN tốt hơn DeepCheck. Trong tấn công một điểm ảnh, trong khi tỉ lệ thành công trung bình của DeepCheck cao

nhất là 0.7% cho sử dụng bộ giải Z3 và SMTInterpol, HA4FNN đạt tới 54.29%. Trong tấn công nhiều điểm ảnh, tỉ lệ thành công trung bình của DeepCheck đạt tới 16.9% với Z3 và 26.8% với SMTInterpol. Tỉ lệ thành công trung bình của HA4FNN cao hơn đáng kể và đạt tới 98.7%. Kết quả thứ hai cho thấy bộ giải phỏng đoán hoạt động tốt hơn bộ giải SMT. Khi số điểm ảnh được thêm nhiễu đối kháng tăng lên, bộ giải SMT tiêu tốn nhiều thời gian hơn để tìm nghiệm do độ phức tạp của hệ ràng buộc. Đặc biệt, trong một vài trường hợp, chi phí tìm nghiệm vượt quá 40 phút nên không khả thi trong thực tế. Ngược lại, bộ giải phỏng đoán chỉ mất vài giây cho mọi trường hợp. Kết quả chương này được công bố tại tạp chí Automated Software Engineering (Q2).

Hạn chế của nghiên cứu này là chỉ hỗ trợ kiểm thử tính chắc chắn của mô hình nơ-ron truyền thẳng. Phương pháp tấn công là không định hướng. Trong thực tế, mô hình tích chập ngày càng được sử dụng phổ biến để phân loại ảnh. Vì thế, chương kế tiếp trình bày tấn công đối kháng trên mô hình tích chập. Phương pháp tấn công sử dụng là tấn công đối kháng có định hướng, một loại phương pháp tấn công phức tạp hơn so với tấn công đối kháng không định hướng.

Chương 4

Phương pháp sử dụng mô hình mã hóa tự động để tấn công đối kháng có định hướng mô hình tích chập

Chương này trình bày cải tiến phương pháp ATN, gọi là PatternAttack, để tấn công đối kháng có định hướng cho mô hình tích chập. PatternAttack hỗ trợ thêm nhiễu đối kháng vào ảnh dự đoán đúng theo nhiều mẫu thêm nhiễu khác nhau gồm mẫu sửa mọi điểm ảnh, mẫu sửa điểm ảnh ở biên đối tượng và mẫu bản đồ nổi bật. PatternAttack cải thiện chất lượng ảnh đối kháng theo tiêu chí L_0 và L_2 bằng cách sử dụng thuật toán tham lam. Một số kết quả thực nghiệm trên MNIST và CIFAR-10 cho thấy hiệu quả của phương pháp đề xuất.

4.1. Giới thiệu

Sử dụng mô hình mã hóa tự động để thực hiện tấn công đối kháng có định hướng mô hình tích chập được đề xuất lần đầu trong mô hình chuyển đổi đối kháng (ATN) [18]. Với đầu vào chính là mô hình kiểm thử, tập ảnh dự đoán đúng và nhãn đích, ý tưởng chung là học mô hình mã hóa tự động từ tập ảnh đầu vào để sinh tập ảnh đối kháng có nhãn là nhãn đích. Ưu điểm chính của mô hình mã hóa tự động này là tính khái quát hóa. Cụ thể, mô hình mã hóa tự động sau khi đã học thành công, mô hình này sẽ được tái sử dụng để sinh ảnh đối kháng từ ảnh dự đoán đúng mới với chi phí thấp. Đầu tiên, ảnh dự đoán đúng sẽ được đẩy qua mô hình mã hóa tự động để lấy ảnh đối kháng ứng cử viên. Nếu ảnh đầu ra này có nhãn thỏa mãn tiêu chí tấn công, ảnh này là ảnh

đối kháng. Tiêu chí thứ nhất là khoảng cách L_2 giữa ảnh dự đoán đúng và ảnh đầu ra phải đủ nhỏ để đảm bảo ảnh đầu ra không quá khác hoặc quá nhiều nhiễu so với ảnh dự đoán đúng. Tiêu chí thứ hai là nhãn của ảnh đầu ra được phân loại là nhãn đích bởi mô hình kiểm thử. Tuy nhiên, ATN có hai hạn chế bao gồm tính đa dạng của ảnh đối kháng và chất lượng ảnh đối kháng.

Đối với vấn đề tính đa dạng của ảnh đối kháng, ý tưởng của ATN là thêm nhiễu đối kháng vào tất cả điểm ảnh trong ảnh dự đoán đúng. Tuy nhiên, ảnh đối kháng có thể sinh ra khi chỉ cần thêm nhiễu đối kháng vào một vài vùng ảnh thay vì toàn bộ ảnh dự đoán đúng. Ví dụ, xét ảnh dự đoán đúng số chín trong MNIST [41], ảnh này gồm hai phần gọi là vùng số (đại diện bởi điểm ảnh có giá trị một ứng với màu trắng) và vùng nền (điểm ảnh có giá trị không và ứng với màu đen). Đối với ảnh này, ATN sẽ thêm nhiễu đối kháng vào mọi điểm ảnh mà không quan tâm đó là vùng số hay vùng nền. Trong khi đó, ảnh đối kháng vẫn có thể sinh ra thành công khi thêm nhiễu đối kháng vào vùng số hoặc vùng nền hoặc vùng điểm ảnh thỏa mãn tính chất nào đó trên ảnh dự đoán đúng. Việc sinh ảnh đối kháng bằng cách thêm nhiễu đối kháng vào ảnh dự đoán đúng theo nhiều tiêu chí thêm nhiễu đối kháng khác nhau sẽ giúp tạo thêm nhiều bằng chứng về tính chắc chắn. Ngoài ra, ảnh đối kháng có nhiều loại nhiễu khác nhau có thể được sử dụng để xây dựng bộ phòng thủ đối kháng như PuVAE [49] và DefenseVAE [50].

Đối với vấn đề chất lượng ảnh đối kháng, bởi vì ATN thêm nhiễu đối kháng vào mọi điểm ảnh trong ảnh dự đoán đúng nên ảnh đối kháng và ảnh dự đoán đúng có thể trông rất khác nhau. Hàm mục tiêu của ATN có hai thành phần. Thành phần thứ nhất tối thiểu hóa khoảng cách giữa ảnh đối kháng và ảnh dự đoán đúng. Thành phần thứ hai tối thiểu hóa khoảng cách giữa véc tơ xác suất của ảnh dự đoán đúng và của ảnh đối kháng. Khoảng cách được sử dụng trong cả hai thành phần là L_2. Nếu mô hình mã hóa tự động sinh bởi ATN học chưa đủ tốt thì ảnh đối kháng sinh ra có thể sẽ rất khác ảnh dự đoán đúng. Một trong những nguyên nhân phổ biến là do kiểm thử viên chưa tìm được thông số học mô hình phù hợp. Ví dụ như trọng số giữa hai thành phần không phù hợp hoặc do kiến trúc mô hình mã hóa tự động chưa đủ chất lượng.

Để giảm thiểu hai vấn đề của ATN, luận án đề xuất phương pháp PatternAttack. Đối với vấn đề đầu tiên, PatternAttack đề xuất ATN khái quát để sinh

ảnh đối kháng có nhiễu đa dạng. Tập các điểm ảnh muốn thêm nhiễu đối kháng sẽ được mô tả bởi mẫu thêm nhiễu. Phương pháp PatternAttack đề xuất ba loại mẫu thêm nhiễu gồm mẫu sửa mọi điểm ảnh, mẫu sửa điểm ảnh ở biên đối tượng và mẫu bản đồ nổi bật. Đối với vấn đề thứ hai, PatternAttack đề xuất thuật toán tham lam để giảm thiểu khoảng cách L_0 và L_2 giữa ảnh đối kháng và ảnh dự đoán đúng. Bộ tối ưu hóa này sẽ nhận diện các điểm ảnh chứa nhiễu dư thừa trong ảnh đối kháng và khôi phục về giá trị tương ứng trong ảnh dự đoán đúng. Điểm ấn tượng của thuật toán tham lam là có thể dùng để cải thiện chất lượng ảnh đối kháng sinh bởi bất kì phương pháp tấn công nào như FGSM [16], CW [17] và BIM [19].

4.2. Các nghiên cứu liên quan

Phần này trình bày các nghiên cứu liên quan đến đề xuất trong chương. Đầu tiên, luận án sẽ trình bày về các phương pháp tấn công đối kháng cho mô hình tích chập. Sau đó, luận án trình bày tổng quan nghiên cứu xoay quanh bản đồ nổi bật và giải thích tại sao PatternAttack sử dụng bản đồ nổi bật là một mẫu thêm nhiễu.

Sinh ảnh đối kháng: Nhiều phương pháp sinh ảnh đối kháng đã được đề xuất để kiểm thử tính chắc chắn của mô hình tích chập. Các phương pháp tiêu biểu có thể kể đến FGSM [16], Carnili-Wagner [17], ATN [18], BIS [19], L-BFGS [32], DeepFool [33], DeepCheck [42] và MI-FGSM [34]. Các phương pháp này được đánh giá theo năm tiêu chí gồm mẫu thêm nhiễu, tính tối ưu hóa, tính khái quát hóa, độ đo L_p và hướng thêm nhiễu đối kháng vào điểm ảnh.

Tiêu chí mẫu thêm nhiễu mô tả những loại điểm ảnh mà các phương pháp thêm nhiễu đối kháng trên ảnh dự đoán đúng. Trong khi L-BFGS, DeepFool, FGSM, CW, MI-FGSM, ATN sử dụng mẫu sửa mọi điểm ảnh, DeepCheck hỗ trợ thêm nhiễu đối kháng vào một hoặc hai điểm ảnh. Phương pháp PatternAttack hỗ trợ thêm nhiễu đối kháng vào ảnh dự đoán đúng theo các mẫu khác nhau gồm mẫu sửa mọi điểm ảnh, mẫu sửa điểm ảnh ở biên đối tượng và mẫu bản đồ nổi bật.

Tiêu chí tính tối ưu hóa là khả năng sinh ảnh đối kháng có chất lượng tốt nhất có thể. Các phương pháp đều cố gắng sinh ảnh đối kháng với độ đo L_p nhỏ trong quá trình thêm nhiễu đối kháng vào ảnh dự đoán đúng. Đối với nhóm phương pháp dựa theo đạo hàm gồm gồm L-BFGS, DeepFool, FGSM, CW, MI-FGSM và ATN, nhóm này xây dựng hàm mục tiêu có một thành phần là khoảng cách L_p từ ảnh dự đoán đúng đến ảnh đối kháng. Sau đó, ảnh dự đoán đúng sẽ được thêm nhiễu đối kháng để tối thiểu hóa thành phần này. Tuy nhiên, hàm mục tiêu là phi tuyến tính nên có thể chỉ tìm thấy nghiệm cục bộ thay vì toàn cục, từ đó dẫn đến chất lượng ảnh đối kháng chưa được tối ưu.

Tiêu chí tính khái quát hóa là khả năng sinh ảnh đối kháng từ ảnh dự đoán đúng mới bằng cách sử dụng tri thức học từ các lần thêm nhiễu đối kháng vào ảnh dự đoán đúng trước đó. Chỉ có ATN và PatternAttack xây dựng mô hình mã hóa tự động trong quá trình thêm nhiễu đối kháng vào ảnh dự đoán đúng trong quá khứ. Khi có ảnh đầu vào mới, mô hình mã hóa tự động này được sử dụng để thêm nhiễu đối kháng vào ảnh đầu vào với chi phí tính toán thấp. Ngược lại, các phương pháp còn lại cần phải thêm nhiễu đối kháng vào ảnh dự đoán đúng bằng cách tối thiểu hóa hàm mục tiêu. Quá trình tối thiểu hóa có thể tốn nhiều thời gian và chi phí hơn ATN và PatternAttack.

Tiêu chí L_p mô tả khoảng cách mà các phương pháp dùng để sinh ảnh đối kháng. Độ đo L_0 tìm cách tối thiểu hóa số điểm ảnh được thêm nhiễu đối kháng trên ảnh dự đoán đúng và được sử dụng trong DeepCheck và CW L_0. Độ đo L_2 hướng đến tối thiểu hóa khoảng cách Euclidean giữa ảnh dự đoán đúng và ảnh đối kháng và được sử dụng trong L-BFGS, DeepFool, CW L_2, ATN và PatternAttack. Độ đo này là độ đo khá phổ biến do tính dễ đạo hàm của hàm mục tiêu. Độ đo L_∞ không quan tâm thêm nhiễu đối kháng bao nhiêu điểm ảnh, thay vào đó chỉ quan tâm đến lượng nhiễu lớn nhất thêm vào một điểm ảnh. Phương pháp FGSM, CW L_∞ và MI-FGSM sử dụng độ đo này.

Tiêu chí hướng thêm nhiễu đối kháng điểm ảnh là cách để thêm nhiễu đối kháng vào ảnh dự đoán đúng và có hai loại chính gồm sử dụng bộ giải và sử dụng đạo hàm. Đối với cách sử dụng bộ giải, DeepCheck sử dụng bộ giải SMT như Z3 [73] hoặc SMTInterpol [74]. Để sử dụng bộ giải này, mô hình tích chập cần chuyển thành mã nguồn, rồi thực thi mã nguồn để sinh đường thi hành và tạo hệ ràng buộc từ đường thi hành này. Các phương pháp còn lại sử dụng

đạo hàm để thêm nhiễu đối kháng vào ảnh dự đoán đúng. Các phương pháp này không cần biến đổi mô hình tích chập thành mã nguồn. Thay vào đó, các phương pháp này đề xuất hàm mục tiêu. Sau đó, ảnh dự đoán đúng được thêm nhiễu đối kháng để tối thiểu hóa hàm mục tiêu này.

Bản đồ nổi bật: Bản đồ nổi bật mô tả mức độ ảnh hưởng của điểm ảnh tới kết quả phân lớp của mô hình tích chập [92, 93]. Kích thước của bản đồ nổi bật giống hệt kích thước đầu vào của mô hình tích chập. Nhiều nghiên cứu khác nhau đã đề xuất để sinh bản đồ nổi bật.

Simonyan và cộng sự [92] lần đầu tiên đề xuất phương pháp để sinh bản đồ nổi bật. Bản đồ này giải thích được ảnh hưởng của vị trí một điểm ảnh đến kết quả phân loại ảnh vào một tầng cụ thể. Ví dụ, xét 100 ảnh trống đồng, những vị trí điểm ảnh trên 100 ảnh này có xu hướng khiến mô hình tích chập phân loại là trống đồng sẽ có mức độ ảnh hưởng lớn hơn. Ngược lại, những vị trí điểm ảnh có tác động không đáng kể lên kết quả phân lớp trống đồng sẽ có mức độ ảnh hưởng không đáng kể.

Tương tự như ý tưởng của Simonyan và cộng sự, Zeiler và cộng sự [94] sử dụng mô hình tích chập nhiều tầng để sinh bản đồ nổi bật. Springenberg và cộng sự [95] đề xuất thuật toán quay lui có định hướng để sinh bản đồ nổi bật tốt hơn Simonyan và cộng sự [92]. Papernot và cộng sự [96] đề xuất bản đồ nổi bật dựa theo Jacobian. Cao và cộng sự [97] đề xuất mô hình tích chập có phản hồi để học được đặc trưng ngữ nghĩa mức cao của một ảnh, rồi ánh xạ vào bản đồ nổi bật. Zhang và cộng sự [98] đề xuất thuật toán quay lui kích thích để giải quyết hạn chế về hiệu năng và chất lượng bản đồ nổi bật của kĩ thuật dựa theo đạo hàm. Fong và cộng sự [99] đề xuất tìm vùng ảnh có ảnh hưởng lớn nhất đến quyết định phân lớp. Dabkowski và cộng sự [100] xây dựng một mô hình học sâu để sinh bản đồ nổi bật nhanh từ một ảnh đầu vào với chi phí tính toán thấp. Yu và cộng sự [101] đề xuất thuật toán để dự đoán bản đồ nổi bật của một ảnh.

Mối quan hệ giữa bản đồ nổi bật và ảnh đối kháng đã được thảo luận trong nhiều nghiên cứu. Nghiên cứu tại [93] và [99] cho rằng bản đồ nổi bật có thể dùng để giải thích kết quả phân lớp của ảnh đối kháng. Tsipras và cộng sự [102] tìm bằng chứng cho thấy mô hình chất lượng tốt thường có bản đồ nổi bật giải thích được. Etmann và cộng sự [103] định lượng mối quan hệ giữa bản đồ nổi

bật và ảnh dự đoán đúng bằng cách phân tích định tuyến. Nhóm nghiên cứu này cho rằng mô hình tích chập phi tuyến tính có kết nối mạnh giữa tính chắc chắn và tính định tuyến.

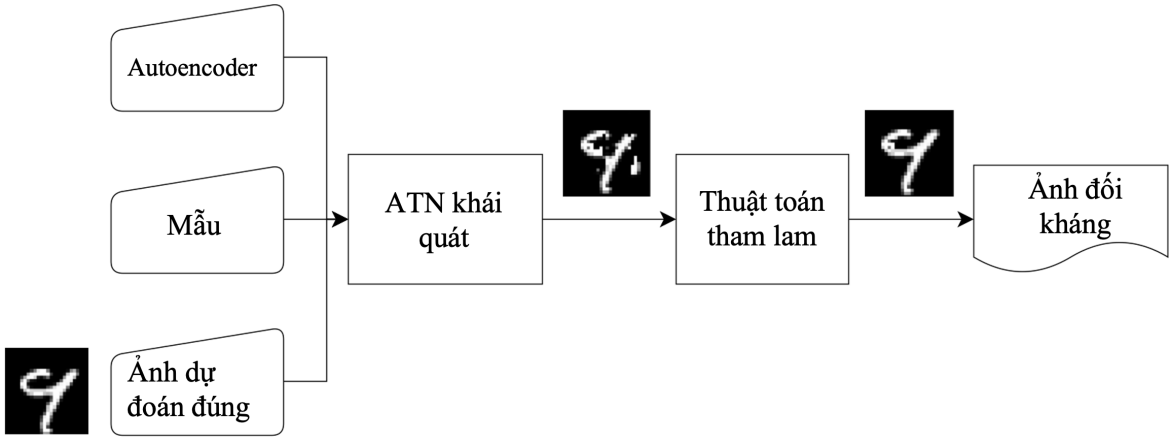
Bởi vì nhiều nghiên cứu cho thấy mối quan hệ giữa bản đồ nổi bật và ảnh đối kháng, PatternAttack sử dụng bản đồ nổi bật trong nghiên cứu của Simonyan và cộng sự [92] như một mẫu thêm nhiễu. Sau khi sinh bản đồ nổi bật từ một tập ảnh dự đoán đúng chung một nhãn, PatternAttack tìm những điểm ảnh có ảnh hưởng cao nhất tới kết quả phân lớp của mô hình tích chập. Sau đó, ATN khái quát thêm nhiễu đối kháng vào những điểm ảnh này trên ảnh dự đoán đúng để sinh ảnh đối kháng.

4.3. Phương pháp PatternAttack

Để sinh ảnh đối kháng có chất lượng tốt và đa dạng, luận án đề xuất PatternAttack với hai cải tiến gồm ATN khái quát và thuật toán tham lam. Tổng quan PatternAttack được mô tả trong Hình 4.1. Trong pha đầu tiên, ATN khái quát được sử dụng để sinh tập ảnh đối kháng dựa theo mẫu thêm nhiễu. Tuy nhiên, tập ảnh đối kháng có thể chất lượng chưa cao vì chúng có thể chứa nhiễu dư thừa. Do đó, trong pha thứ hai, thuật toán tham lam được sử dụng để cải thiện chất lượng ảnh đối kháng. Thuật toán tham lam sẽ nhận diện nhiễu dư thừa và loại bỏ những nhiễu này khỏi ảnh đối kháng. Sau khi nhiễu dư thừa được loại bỏ, ảnh đối kháng cải thiện vẫn bị phân lớp sai bởi mô hình kiểm thử và gần ảnh dự đoán đúng hơn theo độ đo khoảng cách L_0 và L_2.

4.3.1. ATN khái quát

Trong pha đầu tiên của PatternAttack, ATN khái quát được áp dụng để sinh tập ảnh đối kháng theo mẫu thêm nhiễu. Kiến trúc của ATN khái quát là kiến trúc của mô hình mã hóa tự động tích chập xếp chồng. Hàm mục tiêu của ATN khái quát cần thỏa mãn hai tiêu chí. Tiêu chí thứ nhất là ảnh đối kháng cần gần ảnh dự đoán đúng theo độ đo khoảng cách L_2 nhỏ nhất có thể. Tiêu chí thứ hai là nhãn của ảnh đối kháng cần giống nhãn đích y^*. Dựa theo hai tiêu chí này,



Hình 4.1: Tổng quan phương pháp PatternAttack.

hàm mục tiêu của ATN khái quát được định nghĩa như Công thức 4.1.

$$\sum_{\mathbf{x}}(1-\epsilon)\cdot L_2(\gamma(\mathcal{P},\mathbf{x}),\mathbf{res})^2+\epsilon\cdot \mathrm{CE}(y^*,\mathbf{M}(\mathbf{x}')) \tag{4.1}$$

trong đó, $\mathcal{P}$ là mẫu thêm nhiễu, CE là cross-entropy, **res** là ảnh xây dựng lại và được tính bởi $\mathbf{A}(\gamma(\mathcal{P},\mathbf{x}))$. Các mẫu thêm nhiễu sử dụng gồm mẫu bản đồ nổi bật, mẫu sửa mọi điểm ảnh, và mẫu sửa điểm ảnh ở biên đối tượng. **A** là mô hình mã hóa tự động đang cần xây dựng. Hàm $\gamma(\mathcal{P},\mathbf{x})$ tạo ánh xạ mẫu thêm nhiễu của $\mathbf{x}$ và mô tả trong Công thức 4.2. Cụ thể, nếu điểm ảnh $x_i \in \mathbf{x}$ không thỏa mãn mẫu thêm nhiễu $\mathcal{P}$, giá trị thành phần tương ứng trong ánh xạ mẫu thêm nhiễu bằng 0. Ngược lại, nếu điểm ảnh x_i thỏa mãn mẫu thêm nhiễu $\mathcal{P}$, giá trị thành phần tương ứng trong ánh xạ mẫu thêm nhiễu là x_i.

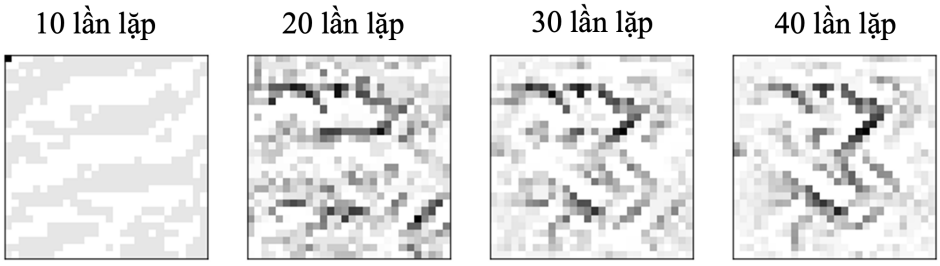
$$\gamma(\mathcal{P},x_i)=\begin{cases} x_i \text{ nếu } x_i \text{ thỏa mãn mẫu thêm nhiễu } \mathcal{P} \\ 0 \text{ ngược lại} \end{cases} \tag{4.2}$$

Tại Công thức 4.1, ảnh đối kháng $\mathbf{x}'$ được tính bởi công thức $\mathbf{x}-\gamma(\mathcal{P},\mathbf{x})+\gamma(\mathcal{P},\mathbf{res})$. Công thức này thay thế các điểm ảnh trong $\mathbf{x}$ thỏa mãn mẫu $\mathcal{P}$ với giá trị mới trả về bởi ATN khái quát. Tất cả các điểm ảnh trong $\mathbf{x}$ không thỏa mãn mẫu thêm nhiễu $\mathcal{P}$ được giữ nguyên. Nghiên cứu này sử dụng ba mẫu thêm nhiễu gồm mẫu sửa mọi điểm ảnh, mẫu sửa điểm ảnh ở biên đối tượng và mẫu bản đồ nổi bật:

- Mẫu sửa mọi điểm ảnh: Mọi điểm ảnh trong ảnh dự đoán đúng đều có thể thêm nhiễu đối kháng để sinh ảnh đối kháng. Đây là mẫu thêm nhiễu được sử dụng bởi phương pháp ATN.
- Mẫu sửa điểm ảnh ở biên đối tượng: Chỉ thêm nhiễu đối kháng những điểm ảnh ở biên đối tượng trong ảnh dự đoán đúng. Ví dụ, trong bộ dữ liệu MNIST, đối tượng là các chữ số được biểu diễn bởi điểm ảnh có giá trị bằng một ứng với màu trắng.
- Mẫu bản đồ nổi bật [92]: Gọi $\mathcal{I}$ là bản đồ nổi bật của ảnh dự đoán đúng. Chiều của $\mathcal{I}$ giống chiều của ảnh dự đoán đúng. $\mathcal{I}$ được sinh bằng cách tối ưu hàm mục tiêu trong Công thức 4.3.

$$\arg\max_{\mathcal{I}} S_{y_{true}}(\mathcal{I}) - \lambda \cdot L_2(\mathcal{I})^2 \tag{4.3}$$

trong đó, $S_{y_{true}}$ là điểm số của nhãn y_{true} và là giá trị trước khi áp dụng hàm softmax để tính xác suất. λ là siêu tham số và $\mathcal{I}$ được khởi tạo là ảnh trong suốt. Nhóm tác giả này đã sử dụng thuật toán lan truyền ngược như SGD [58] để tìm $\mathcal{I}$. Ví dụ của bản đồ nổi bật được mô tả trong Hình 4.2. Cụ thể, bản đồ nổi bật này được học từ bộ dữ liệu 1,000 ảnh dự đoán đúng có nhãn ba thuộc MNIST. Từ phía bên trái cùng sang bên phải cùng, bản đồ nổi bật được chụp sau 10, 20, 30 và 40 lần lặp. Giá trị λ được chọn bằng 0.2. Điểm ảnh tối hơn trên $\mathcal{I}$ có nghĩa rằng điểm ảnh đó có ảnh hưởng nhiều hơn để mô hình quyết định là nhãn số ba.



Hình 4.2: Ví dụ bản đồ nổi bật.

So sánh với Công thức 2.18, Công thức 4.1 có những điểm khác biệt sau:

- Trọng số giữa các thành phần: Trong hàm mục tiêu của PatternAttack, trong số là $(1 - \epsilon)$ với thành phần thứ nhất và ϵ với thành phần thứ hai.

Trong ATN, thành phần đầu tiên có trọng số β và thành phần thứ hai có trọng số bằng một. Về bản chất, hai phương pháp không sự khác biệt về trọng số giữa thành phần. Luận án chọn loại trọng số như trên để thể hiện rõ hơn về mức độ quan trọng giữa các thành phần.

- Cross-entropy: Trong thành phần thứ hai của Công thức 2.18, trong khi ATN sử dụng L_2, PatternAttack sử dụng cross-entropy để Công thức 4.1 hội tụ tốt hơn.
- Sử dụng mẫu thêm nhiễu: Trong Công thức 2.18, mọi điểm ảnh đều có thể thêm nhiễu. Tuy nhiên, trong Công thức 4.1, chỉ những điểm ảnh thỏa mãn mẫu thêm nhiễu mới có thể thêm nhiễu.
- Hàm r_α: Hàm này được sử dụng trong Công thức 2.18 để ảnh đối kháng có chất lượng tốt hơn. Công thức 4.1 không sử dụng hàm r_α. Thực nghiệm cho thấy rằng sử dụng hàm r_α trong PatternAttack không cho kết quả tốt hơn khi không sử dụng hàm r_α. Thay vì sử dụng hàm r_α, luận án sử dụng thuật toán tham lam để cải thiện chất lượng ảnh đối kháng.

4.3.2. Cải thiện chất lượng ảnh đối kháng

Để cải thiện chất lượng ảnh đối kháng, luận án cần nhận diện nhiễu không có ảnh hưởng hoặc ảnh hưởng rất nhỏ vào kết quả phân lớp. Vì thế, luận án đề xuất các định nghĩa của nhiễu dư thừa, điểm ảnh đối kháng, điểm ảnh đối kháng dư thừa, ảnh đối kháng cải thiện và được mô tả dưới đây.

Định nghĩa 6. [Nhiễu dư thừa] Cho ảnh dự đoán đúng $\mathbf{x}$, ảnh đối kháng $\mathbf{x}'$ và mô hình kiểm thử $\mathbf{M}$, véc tơ nhiễu $\boldsymbol{\zeta}$ được gọi là dư thừa khi và chỉ khi $\mathbf{x} + \boldsymbol{\zeta}$ được dự đoán đúng và $\mathbf{x}' - \boldsymbol{\zeta}$ được dự đoán sai bởi $\mathbf{M}$.

Định nghĩa 7. [Điểm ảnh đối kháng] Cho ảnh dự đoán đúng $\mathbf{x}$ và ảnh đối kháng tương ứng $\mathbf{x}'$, điểm ảnh x'_i được gọi là đối kháng khi và chỉ khi $x'_i \neq x_i$.

Định nghĩa 8. [Điểm ảnh đối kháng dư thừa] Cho ảnh đối kháng $\mathbf{x}'$, điểm ảnh đối kháng x'_i được gọi là dư thừa khi và chỉ khi x'_i chứa nhiễu dư thừa.

Định nghĩa 9. [Ảnh đối kháng cải thiện] Cho ảnh đối kháng $\mathbf{x}$', ảnh đối kháng cải thiện $\mathbf{x}'_{im}$ là kết quả của loại bỏ nhiễu dư thừa khỏi ảnh đối kháng $\mathbf{x}'$, trong

đó $\mathbf{x}'_{im}$ và $\mathbf{x}'$ có cùng nhãn trả về bởi mô hình kiểm thử mô hình học sâu $\mathbf{M}$ trong tấn công đối kháng có định hướng hoặc cùng khác nhãn đúng trong tấn công đối kháng không định hướng.

Đầu ra của ATN khái quát là ảnh đối kháng có thể chứa nhiễu dư thừa. Để nâng cao chất lượng ảnh đối kháng, luận án đề xuất cách cải thiện chất lượng ảnh đối kháng theo tiêu chí khoảng cách L_0 và L_2. Khi loại bỏ nhiễu dư thừa, ảnh đối kháng cải thiện sẽ có khoảng cách L_0 và L_2 gần ảnh dự đoán đúng hơn ảnh đối kháng chưa cải thiện. Ngoài ra, nhãn của ảnh đối kháng cải thiện vẫn được mô hình kiểm thử phân loại là nhãn đích y^*.

Luận án đề xuất Thuật toán 4.1 để cải thiện chất lượng ảnh đối kháng. Đầu vào thuật toán gồm ảnh đối kháng $\mathbf{x}'$, ảnh dự đoán đúng $\mathbf{x}$, nhãn đích y^*, mô hình kiểm thử $\mathbf{M}$, bước nhảy $\alpha >= 1$, ngưỡng $\delta < \alpha$ và tỉ lệ suy giảm $t >= 1$. Đầu ra của thuật toán là ảnh đối kháng cải thiện. Nếu một điểm ảnh trên ảnh đối kháng được loại bỏ nhiễu, điểm ảnh đó gọi là được khôi phục. Thuật toán sẽ loại bỏ nhiễu đối kháng khỏi ảnh đối kháng qua nhiều lần lặp, trong đó ảnh đối kháng cải thiện bởi lần lặp sau có chất lượng tốt hơn hoặc tương đương ảnh đối kháng cải thiện sinh bởi lần lặp trước. Vào lần loại bỏ nhiễu đối kháng thứ i, tỉ lệ khôi phục r được tính như Công thức 4.4.

$$r(\mathbf{x}, \mathbf{x}', i) = \frac{L_0(\mathbf{x}, \mathbf{x}'_i)}{L_0(\mathbf{x}, \mathbf{x}')} \tag{4.4}$$

trong đó, $\mathbf{x}'_i$ là phiên bản ảnh đối kháng cải thiện thứ i.

Tại lần cải thiện thứ p, tốc độ khôi phục r hội tụ nếu và chỉ nếu điều kiện trong Công thức 4.5 thỏa mãn.

$$\bigwedge (r(\mathbf{x}, \mathbf{x}', i) - r(\mathbf{x}, \mathbf{x}', i-1) <= \beta) \ \mid \ i \in [p-k+1, p] \tag{4.5}$$

trong đó, $k >= 1$ là số lần lặp gần nhất đang xem xét.

Nếu α lớn hơn δ và tỉ lệ khôi phục không hội tụ, thuật toán tìm điểm ảnh có nhiễu dư thừa bằng cách so sánh $\mathbf{x}$ và $\mathbf{x}'$, sau đó lưu những điểm ảnh này trong tập S (dòng 2). Sau đó, các điểm ảnh có nhiễu dư thừa này được xếp hạng theo giá trị tăng dần theo mức độ ảnh hưởng lên kết quả phân lớp của mô hình kiểm

Thuật toán 4.1 : Thuật toán tham lam theo độ đo L_0 và L_2

Đầu vào: ảnh dự đoán đúng $\mathbf{x}$, ảnh đối kháng $\mathbf{x}'$, nhãn đích y^*, mô hình kiểm thử $\mathbf{M}$, bước nhảy α, ngưỡng δ và hệ số suy giảm t

Đầu ra: ảnh đối kháng cải thiện

1: **while** $\alpha > \delta$ **do**
2: $S \leftarrow$ COMPUTE_DIFF_FEATURES$(\mathbf{x}, \mathbf{x}')$ ▷ Tìm những điểm ảnh khác nhau
3: $S' \leftarrow$ RANK_FEATURES(S) ▷ Xếp hạng điểm ảnh
4: $\#block \leftarrow \lceil |S'|/\alpha \rceil$ ▷ Tính số lần thêm nhiễu đối kháng tối đa
5: **for** $i \leftarrow 0, i < \#block, i \leftarrow i + 1$ **do**
6: $start \leftarrow \alpha \cdot i$
7: $end \leftarrow start + \alpha$
8: **if** $end > |S'|$ **then**
9: $end \leftarrow |S'|$
10: **end if**
11: $\mathbf{x}'_{clone} \leftarrow \mathbf{x}'$
12: $\mathbf{x}'_{start...end-1} \leftarrow \mathbf{x}_{start...end-1}$ ▷ Thử loại bỏ nhiễu của một tập điểm ảnh
13: **if** $\arg\max \mathbf{M}(\mathbf{x}') \neq y^*$ **then**
14: $\mathbf{x}' \leftarrow \mathbf{x}'_{clone}$
15: **end if**
16: **if** $is\_converged$ **then** ▷ Nếu quá trình cải thiện có xu hướng hội tụ
17: return $\mathbf{x}'$
18: **end if**
19: **end for**
20: $\alpha \leftarrow \lfloor \alpha/t \rfloor$ ▷ Giảm số điểm ảnh trong một lần thêm nhiễu đối kháng
21: **end while**
22: **return** $\mathbf{x}'$

thử và lưu trong tập S' (dòng 3). Thuật toán đề xuất tuần tự khôi phục các điểm ảnh có nhiễu dư thừa thuộc tập S' từ phần tử đầu tiên. Tập S' sẽ được chia nhỏ thành nhiều khối, trong đó mỗi khối được định vị trong S' bởi vị trí bắt đầu *start* và vị trí kết thúc *end*. Kích thước của khối được định nghĩa bởi bước nhảy α (dòng 7 - 11). Thay vì khôi phục từng điểm ảnh, thuật toán sẽ khôi phục theo từng khối (dòng 12). Nếu việc khôi phục thay đổi kết quả phân lớp, các điểm ảnh thuộc khối này sẽ không được khôi phục về giá trị gốc (dòng 13 - 15). Sau đó, α được tính lại bằng cách chia cho t (dòng 20). Thuật toán 4.1 kết thúc khi α không lớn hơn ngưỡng δ (dòng 1) hoặc tỉ lệ khôi phục (kí hiệu là r) hội tụ (dòng 16).

Bước nhảy α: Bước nhảy α đóng vai trò quan trọng trong thuật toán tham lam. Bước nhảy α có giá trị nhỏ nhất bằng một. Chọn đúng giá trị α sẽ giúp thuật toán tham lam có hiệu năng tốt hơn. Nếu giá trị α bằng một, thuật toán

tham lam sẽ cần mô hình kiểm thử dự đoán ít nhất *#block* lần. Vấn đề là chi phí để dự đoán khá tốn kém khi số lượng *#block* đủ lớn, đặc biệt khi sử dụng mẫu sửa mọi điểm ảnh. Nói chung, giá trị α nên chọn vừa phải để có thể khiến thuật toán cải thiện ảnh đối kháng nhanh hơn.

Công thức xếp hạng điểm ảnh: Mục đích của công thức xếp hạng điểm ảnh (dòng 3) là ước lượng ảnh hưởng của điểm ảnh có nhiễu dư thừa lên kết quả phân lớp của mô hình kiểm thử. Ảnh hưởng của điểm ảnh x'_i lên nơ-ron đầu ra thứ y^* được kí hiệu là $s_{y^*}(x'_i)$. Điểm ảnh x'_i có ảnh hưởng lớn hơn điểm ảnh x'_j khi và chỉ khi $s_{y^*}(x'_i) > s_{y^*}(x'_j)$.

Đối với điểm ảnh x_i trên ảnh dự đoán đúng, nhiều công thức xếp hạng đã được đề xuất như JSMA [96], COI [42], v.v. Đối với công thức JSMA, nhóm tác giả đề xuất hai công thức con. Luận án dùng tên gọi JSMA$^+$ và JSMA$^-$ để phân biệt. Cụ thể, nếu điểm ảnh x_i sẽ được tăng giá trị để sinh ảnh đối kháng, JSMA$^+$ gán ảnh hưởng của điểm ảnh này một giá trị khác 0. Ngược lại, nếu điểm ảnh x_i sẽ được giảm giá trị để sinh ảnh đối kháng, JSMA$^-$ gán ảnh hưởng của điểm ảnh này một giá trị khác 0. Đối với công thức COI, ảnh hưởng của điểm ảnh x_i được tính bằng cách nhân giá trị điểm ảnh đó với giá trị đạo hàm của điểm ảnh.

Tuy nhiên, các công thức trên không áp dụng để xếp hạng điểm ảnh có nhiễu dư thừa. Luận án đã áp dụng để tính ảnh hưởng của những điểm ảnh có nhiễu dư thừa. Cụ thể, ảnh hưởng của điểm ảnh có nhiễu dư thừa x'_i được tính như Công thức 4.6, 4.7 và 4.8.

- JSMA$^+$:

$$s_{y^*}(x'_i) = \begin{cases} 0 \quad \text{if } \frac{\partial \boldsymbol{M}_{y^*}(\mathbf{x}')}{\partial x'_i} < 0 \text{ hoặc } \sum_{j \neq y^*} \frac{\partial \boldsymbol{M}_j(\mathbf{x}')}{\partial x'_i} > 0 \\ \frac{\partial \boldsymbol{M}_{y^*}(\mathbf{x}')}{\partial x'_i} \cdot \mid \sum_{j \neq y^*} \frac{\partial \boldsymbol{M}_j(\mathbf{x}')}{\partial x'_i} \mid \text{ ngược lại} \end{cases} \tag{4.6}$$

- JSMA$^-$:

$$s_{y^*}(x'_i) = \begin{cases} 0 \quad \text{if } \frac{\partial \boldsymbol{M}_{y^*}(\mathbf{x}')}{\partial x'_i} > 0 \text{ hoặc } \sum_{j \neq y^*} \frac{\partial \boldsymbol{M}_j(\mathbf{x}')}{\partial x'_i} < 0 \\ \mid \frac{\partial \boldsymbol{M}_{y^*}(\mathbf{x}')}{\partial x'_i} \mid \cdot (\sum_{j \neq y^*} \frac{\partial \boldsymbol{M}_j(\mathbf{x}')}{\partial x'_i}) \text{ ngược lại} \end{cases} \tag{4.7}$$

- COI:

$$s_{y^*}(x'_i) = x'_i \cdot \frac{\partial \boldsymbol{M}_{y^*}(\mathbf{x}')}{\partial x'_i} \tag{4.8}$$

4.4. Thực nghiệm

Để chứng minh hiệu quả của PatternAttack, thực nghiệm so sánh PatternAttack với các phương pháp sinh ảnh đối kháng phổ biến gồm FGSM [16], L-BFGS [32] và CW L_2 [17]. Vì không có mã nguồn công bố bởi nhóm tác giả, thực nghiệm cài đặt FGSM và L-BFGS. Đối với CW L_2, thực nghiệm sử dụng mã nguồn có sẵn[1]. Môi trường thực nghiệm là Google Colab. Thực nghiệm trả lời ba câu hỏi nghiên cứu sau:

- **RQ1 - Cải thiện tính đa dạng**: Đánh giá tính đa dạng của ảnh đối kháng sinh bởi ATN khái quát khi so sánh với các phương pháp khác?
- **RQ2 - Cải thiện chất lượng**: Đánh giá chất lượng ảnh đối kháng khi được cải thiện bởi thuật toán tham lam?
- **RQ3 - Hiệu năng**: Đánh giá hiệu năng của PatternAttack?

4.4.1. Cấu hình

4.4.1.1. Mô hình kiểm thử

Thực nghiệm xây dựng bốn mô hình kiểm thử gồm hai mô hình được học trên MNIST và hai mô hình khác học trên CIFAR-10. Kiến trúc hai mô hình là LeNet-5 [57] và AlexNet [104]. Bảng 4.1 trình bày các mô hình này.

Kết quả của tấn công $m \rightarrow n$ là các mô hình mã hóa tự động tích chập xếp chồng. Các mô hình mã hóa tự động này sẽ biến đổi ảnh dự đoán đúng có nhãn m thành ảnh đối kháng có nhãn n. Thực nghiệm chia bộ dữ liệu MNIST và CIFAR-10 thành các tập dữ liệu con. Gọi M_i kí hiệu tập con của MNIST và C_i là tập con của CIFAR-10. Thực nghiệm chia bộ dữ liệu thành ba nhóm con gồm

[1] https://github.com/carlini/nn_robust_attacks

Bảng 4.1: Độ chuẩn xác của mô hình kiểm thử trên tập học và tập kiểm thử

Bộ dữ liệu	Mô hình	Tập học	Tập kiểm thử
MNIST	AlexNet	99.03%	98.75%
	LeNet-5	99.86%	98.82%
CIFAR-10	AlexNet	99.70%	76.22%
	LeNet-5	97.71%	69.40%

G_{train}, G_{val} và G_{new}. Nhóm $G_{train} = \{M_{train}, C_{train}\}$ chứa một phần tập học và được sử dụng để học mô hình mã hóa tự động. Mỗi bộ dữ liệu trong G_{train} chứa 1,000 ảnh dự đoán đúng có nhãn m. Nhóm $G_{val} = \{M_{val}, C_{val}\}$ chứa một phần tập học và không được sử dụng để học mô hình mã hóa tự động. Mỗi bộ dữ liệu trong G_{val} chứa 4,000 ảnh dự đoán đúng có nhãn m. Nhóm $G_{new} = \{M_{new}, C_{new}\}$ là tập kiểm thử. M_{new} chứa 1,000 ảnh dự đoán đúng có nhãn m. C_{new} có 862 và 759 ảnh dự đoán đúng có nhãn m với mô hình AlexNet và LeNet-5. Nhóm G_{val} và nhóm G_{new} được sử dụng để đánh giá tính khái quát hóa của mô hình mã hóa tự động được học trên nhóm G_{train}.

Với mỗi mô hình kiểm thử, tổng cộng có 90 tấn công do có mười nhãn gốc và chín nhãn đích. Tuy nhiên, thực nghiệm này tiến hành trên một vài tấn công đặc trưng. Với MNIST, thực nghiệm chọn tấn công $9 \rightarrow 7$ với AlexNet và $9 \rightarrow 4$ với LeNet-5. Về mức cảm quan, hình dạng của nhãn số chín khá khác hình dạng của nhãn số bảy và số bốn nên có thể sẽ khó thêm nhiễu đối kháng những ảnh dự đoán đúng số chín. Với CIFAR-10, thực nghiệm chọn tấn công *truck* $\rightarrow$ *horse* cho AlexNet và *truck* $\rightarrow$ *deer* cho LeNet-5. Nói chung, hình dạng *horse* và *deer* rất khác nhau nên có thể sẽ khó thêm nhiễu đối kháng vào ảnh dự đoán đúng *horse*.

4.4.1.2. Cấu hình PatternAttack

a. ATN khái quát

Kiến trúc: Chọn kiến trúc tối ưu cho ATN khái quát để phù hợp với mọi mô hình kiểm thử là một bài toán khó. Kiến trúc ATN khái quát nên được chọn dựa theo kinh nghiệm của người kiểm thử. Với từng bộ dữ liệu, thực nghiệm dùng chung một kiến trúc ATN khái quát. Thực nghiệm đã thử với nhiều kiến

trúc (cùng với các giá trị siêu tham số) khác nhau và chọn kiến trúc tốt nhất trong các lần thử nghiệm. Mô tả các kiến trúc này được thể hiện ở Bảng 4.2 và Bảng 4.3.

Bảng 4.2: Kiến trúc ATN khái quát sử dụng để sinh ảnh đối kháng từ M_{train}, M_{val} và M_{new} (MNIST)

Loại tầng	Đầu vào	Đầu ra
InputLayer	(28, 28, 1)	(28, 28, 1)
Conv2D	(28, 28, 1)	(28, 28, 32)
MaxPooling2D	(28, 28, 32)	(14, 14, 32)
Conv2D	(14, 14, 32)	(14, 14, 32)
MaxPooling2	(14, 14, 32)	(7, 7, 32)
Conv2D	(7, 7, 32)	(7, 7, 32)
UpSampling2D	(7, 7, 32)	(14, 14, 32)
Conv2D	(14, 14, 32)	(14, 14, 32)
UpSampling2	(14, 14, 32)	(28, 28, 32)
Conv2D	(28, 28, 32)	(28, 28, 1)

Bảng 4.3: Kiến trúc ATN khái quát sử dụng để sinh ảnh đối kháng từ C_{train}, C_{val} và C_{new} (CIFAR-10)

Loại tầng	Đầu vào	Đầu ra
InputLayer	(32, 32, 3)	(32, 32, 3)
Conv2D	(32, 32, 3)	(32, 32, 32)
MaxPooling2D	(32, 32, 32)	(16, 16, 32)
Conv2D	(16, 16, 32)	(16, 16, 32)
MaxPooling2	(16, 16, 32)	(8, 8, 32)
Conv2D	(8, 8, 32)	(8, 8, 32)
UpSampling2D	(8, 8, 32)	(16, 16, 32)
Conv2D	(16, 16, 32)	(16, 16, 32)
UpSampling2	(16, 16, 32)	(32, 32, 32)
Conv2D	(32, 32, 32)	(32, 32, 3)

Nhóm G_{train} được sử dụng để học mô hình kiểm thử. Mỗi mô hình mã hóa tự động được học đến 500 lần lặp với kích thước khối là 256. Quá trình học sử dụng chiến thuật dừng sớm khi không có sự cải tiến về giá trị hàm mục tiêu. Sau khi học mô hình mã hóa tự động xong, ảnh dự đoán đúng sẽ được đẩy vào mô hình mã hóa tự động này để sinh ảnh đối kháng ứng cử viên. Sau đó, nếu

mô hình kiểm thử dự đoán nhãn ảnh đối kháng ứng cử viên này là nhãn đích, các ảnh đối kháng này là hợp lệ và ngược lại.

Trọng số ϵ: Thực nghiệm cần chọn giá trị trọng số ϵ trong Công thức 4.1 để sinh ảnh đối kháng. Công thức này gồm thành phần khoảng cách và thành phần cross-entropy. Mọi tấn công đều được thực hiện với tập các trọng số cố định $\{0.01, 0.05, 0.5, 0.95, 0.99, 1\}$. Nếu ϵ bằng 0.01 hoặc 0.05, tấn công có xu hướng tối thiểu hóa khoảng cách. Nếu ϵ bằng 0.5, tấn công tối thiểu hóa hai thành phần với trọng số bằng nhau. Nếu ϵ bằng 0.95 hoặc 0.99, tấn công có xu hướng tối thiểu hóa cross-entropy.

Mẫu thêm nhiễu: Thực nghiệm tiến hành với ba loại mẫu thêm nhiễu đã đề xuất. Đối với mẫu bản đồ nổi bật, mẫu thêm nhiễu mô tả ảnh hưởng của điểm ảnh lên kết quả phân lớp của mô hình kiểm thử. Trên bộ dữ liệu MNIST, thực nghiệm chọn 100 điểm ảnh trên bản đồ nổi bật có ảnh hưởng lớn nhất. Tấn công thêm nhiễu đối kháng vào 100 điểm ảnh này để sinh ảnh đối kháng. Đối với CIFAR-10, thực nghiệm chọn 400 điểm ảnh có ảnh hưởng lớn nhất vì số điểm ảnh của CIFAR-10 lớn gấp khoảng bốn lần MNIST.

b. Thuật toán tham lam

Thuật toán tham lam đề xuất để cải thiện chất lượng ảnh đối kháng theo tiêu chí L_0 và L_2. Đầu vào của thuật toán tham lam gồm bốn tham số là bước nhảy α, ngưỡng δ, tỉ lệ suy giảm t và thuật toán xếp hạng điểm ảnh. Trong thực nghiệm, $\delta = 0$ và $t = 2$.

Bước nhảy α: Sau khi phân tích kết quả với nhiều bước nhảy α khác nhau, luận án nhận thấy rằng sử dụng α phù hợp sẽ có ảnh hưởng đáng kể lên hiệu năng của thuật toán tham lam. Luận án đề xuất chọn $\alpha = 6$ cho MNIST và $\alpha = 30$ cho CIFAR-10.

Thuật toán xếp hạng điểm ảnh: Thực nghiệm sử dụng ba thuật toán gồm COI, JSMA và ngẫu nhiên. Trong khi COI và JSMA sắp xếp điểm ảnh có nhiễu đối kháng theo mức độ ảnh hưởng lên nơ-ron đích, thuật toán ngẫu nhiên sắp xếp điểm ảnh ngẫu nhiên. Kết quả cho thấy trên tập MNIST, áp dụng COI hoặc JSMA giúp thuật toán tham lam nhanh hơn so với thuật toán ngẫu nhiên. Trên CIFAR-10, sử dụng JSMA cho tỉ lệ khôi phục hội tụ nhanh nhất.

4.4.1.3. Cấu hình các phương pháp so sánh

Các phương pháp so sánh gồm FGSM, L-BFGS và CW L_2. Giá trị trọng số của FGSM và L-BFGS giống ATN khái quát. Đối với CW L_2, phương pháp này không cần chọn trọng số thủ công vì sử dụng thuật toán nhị phân để tìm trọng số tốt.

4.4.2. Kết quả

4.4.2.1. RQ1 - Cải thiện tính đa dạng ảnh đối kháng

Phần này đánh giá khả năng của ATN khái quát để cải thiện tính đa dạng của ảnh đối kháng. Thực nghiệm sử dụng ba mẫu thêm nhiễu gồm mẫu sửa mọi điểm ảnh, mẫu sửa điểm ảnh ở biên đối tượng và mẫu bản đồ nổi bật. Bảng 4.4 so sánh tỉ lệ thành công trung bình giữa PatternAttack và các phương pháp khác.

Bảng 4.4: Thống kê tỉ lệ thành công trung bình

Mô hình	Bộ dữ liệu	ATN khái quát			CW L_2	FGSM	L-BFGS
		Mẫu sửa ở biên	Mẫu bản đồ nổi bật	Mẫu sửa mọi điểm ảnh (ATN)			
AlexNet 9 → 7	M_{train}	21.2%	14.9%	99.9%	100%	8.2%	23.1%
	M_{val}	17.1%	22.2%	99.4%	100%	8.1%	21.8%
	M_{new}	20.8%	9.4%	99.1%	99.2%	6.7%	23.4%
LeNet-5 9 → 4	M_{train}	55.1%	23.2%	95.5%	100%	20.1%	34.2%
	M_{val}	45.8%	23.8%	99.2%	100%	19.9%	36.4%
	M_{new}	52.3%	20.6%	92.2%	99.1%	22.4%	33.1%
AlexNet truck → horse	C_{train}	8.4%	68.3%	100%	100%	5.4%	24.0%
	C_{val}	6.2%	62.6%	99.2%	99.1%	5.6%	22.4%
	C_{new}	6.5%	67.5%	98.1%	99.3%	3.9%	23.7%
LeNet-5 truck → deer	C_{train}	28.3%	61.1%	100%	100%	18.7%	36.7%
	C_{val}	24.1%	54.3%	99.3%	100%	20.7%	34.6%
	C_{new}	22.6%	58.5%	99.1%	98.1%	15.8%	38.3%

ATN khái quát có thể sinh ảnh đối kháng với nhiễu đa dạng hơn các phương pháp so sánh. Mẫu sửa mọi điểm ảnh đạt được tỉ lệ thành công trung bình đa số khoảng trên 99% hơn các mẫu thêm nhiễu trên MNIST và CIFAR-10. Đối với mẫu sửa điểm ảnh ở biên đối tượng, tỉ lệ thành công dao động từ 6.2% tới 55.1%. Đối với mẫu bản đồ nổi bật, tỉ lệ thành công nằm trong khoảng 9.4% tới 68.3%. Nếu ATN khái quát chỉ dùng mẫu sửa mọi điểm ảnh như CW L_2, FGSM và L-BFGS, một lượng lớn ảnh đối kháng sẽ bị bỏ qua.

ATN khái quát có thể sinh ảnh đối kháng từ bộ dữ liệu mới với tỉ lệ thành công trung bình xấp xỉ như tập học. Tập học được kí hiệu G_{train} gồm M_{train} và C_{train}. Bộ dữ liệu mới được kí hiệu G_{val} (gồm M_{val} và C_{val}) và G_{new} (gồm M_{new} và C_{new}). Tỉ lệ thành công trung bình chênh lệch lớn nhất trên bộ dữ liệu mới và tập học khoảng 10%. Đây là tính khái quát của ATN khái quát mà các phương pháp so sánh không hỗ trợ.

4.4.2.2. RQ2 - Cải thiện chất lượng ảnh đối kháng

Thực nghiệm đánh giá mức độ cải thiện chất lượng ảnh đối kháng của thuật toán tham lam. Để nâng cao hiệu năng, thay vì cải thiện từng ảnh tuần tự, thực nghiệm sẽ đặt nhiều ảnh trong một tập và cải thiện trên tập đó. Bảng 4.5 trình bày tỉ lệ giảm nhiễu trung bình theo hai độ đo L_0 và L_2. Đối với độ đo L_0, thực nghiệm cho thấy thuật toán tham lam có thể khiến ảnh dự đoán đúng chỉ thêm nhiễu đối kháng vào một điểm ảnh để sinh ảnh đối kháng. Đối với ảnh đối kháng sinh bởi ATN khái quát, FGSM và L-BFGS, thuật toán tham lam có thể cải thiện chất lượng ảnh đối kháng từ 70.1% đến 98% theo L_0 và từ 36.9% đến 89% theo L_2.

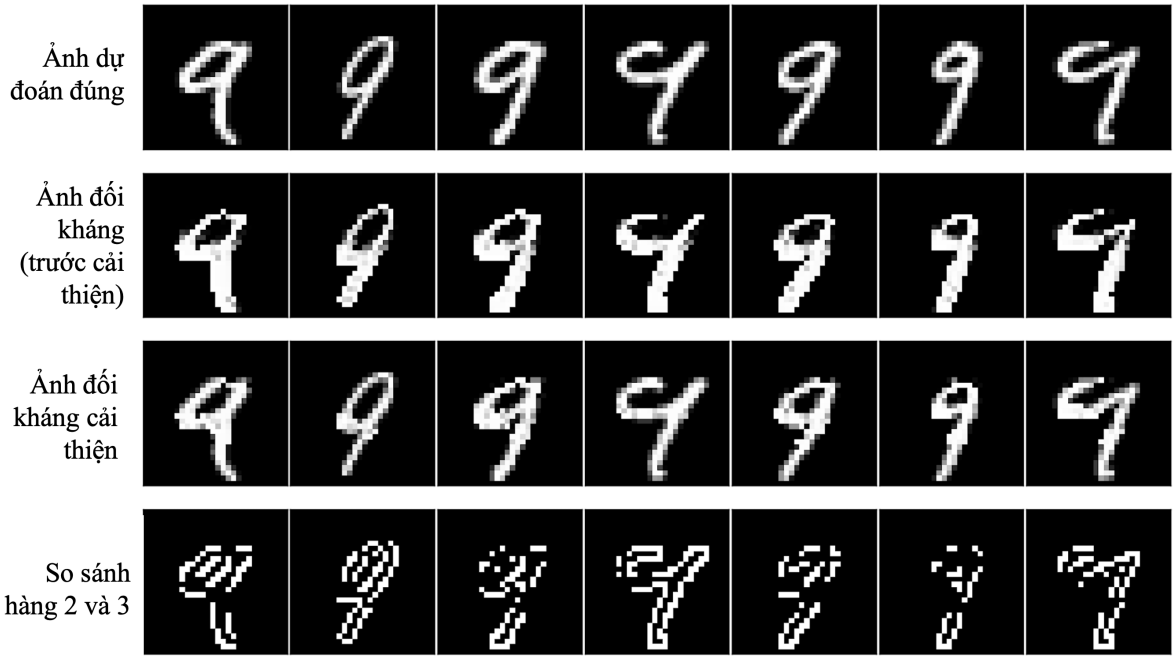
Đối với CW L_2, phương pháp này có thể sinh ảnh đối kháng có khoảng cách L_2 rất nhỏ do sử dụng thuật toán tìm kiếm nhị phân để chọn cấu hình tốt nhất. Thuật toán tham lam vẫn có thể cải thiện chất lượng ảnh đối kháng sinh bởi CW L_2 với tỉ lệ giảm nhiễu trung bình từ 0.1% đến 4.5%. Hình 4.3 và 4.4 mô tả ảnh đối kháng cải thiện tối ưu bởi PatternAttack. Nhãn *So sáng hàng 2 và 3* thể hiện sự khác biệt giữa ảnh đối kháng và ảnh đối kháng cải thiện bằng điểm ảnh màu trắng.

Bảng 4.5: Thống kê tỉ lệ giảm nhiễu trung bình của thuật toán tham lam

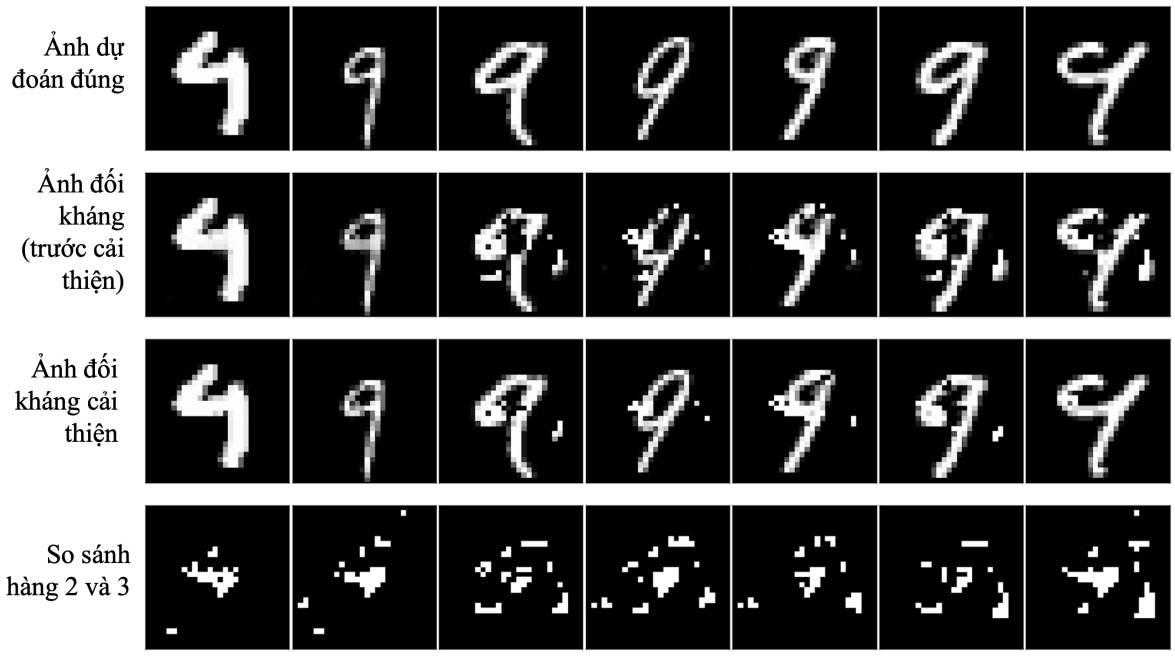
Mô hình	Phương pháp	L_0	L_2
AlexNet $9 \rightarrow 7$	ATN khái quát+mẫu sửa điểm ảnh ở biên đối tượng	75.9%	42.9%
	ATN khái quát+mẫu bản đồ nổi bật	84.4%	51.0%
	ATN khái quát+mẫu sửa mọi điểm ảnh (ATN)	98.0%	69.6%
	FGSM	91.5%	69.2%
	L-BFGS	93.4%	75.6%
	CW L_2	7.9%	0.1%
LeNet-5 $9 \rightarrow 4$	ATN khái quát+mẫu sửa điểm ảnh ở biên đối tượng	70.1%	38.9%
	ATN khái quát+mẫu bản đồ nổi bật	71.9%	36.9%
	ATN khái quát+mẫu sửa mọi điểm ảnh (ATN)	96.7%	65.2%
	FGSM	85.7%	68.5%
	L-BFGS	88.6%	56.8%
	CW L_2	6.5%	0.1%
AlexNet *truck* $\rightarrow$ *horse*	ATN khái quát+mẫu sửa điểm ảnh ở biên đối tượng	96.3%	78.5%
	ATN khái quát+mẫu bản đồ nổi bật	85.8%	60.2%
	ATN khái quát+mẫu sửa mọi điểm ảnh (ATN)	89.7%	67.6%
	FGSM	87.8%	82.7%
	L-BFGS	95.7%	79.0%
	CW L_2	0.8%	4.5%
LeNet-5 *truck* $\rightarrow$ *deer*	ATN khái quát+mẫu sửa điểm ảnh ở biên đối tượng	95.7%	78.0%
	ATN khái quát+mẫu bản đồ nổi bật	88.4%	62.4%
	ATN khái quát+mẫu sửa mọi điểm ảnh (ATN)	89.4%	65.9%
	FGSM	98.0%	89.0%
	L-BFGS	94.0%	82.9%
	CW L_2	3.3%	1.3%

4.4.2.3. RQ3 - Hiệu năng

Thực nghiệm cần đánh giá hiệu năng của PatternAttack khi áp dụng trong thực tế. Bảng 4.6 so sánh hiệu năng của PatternAttack và các phương pháp khác. Thực nghiệm cho thấy PatternAttack có thể sinh ảnh đối kháng từ bộ dữ liệu mới gồm nhóm G_{val} và nhóm G_{new} với hiệu năng tốt hơn từ 2 lần đến 27 lần so với việc sinh ảnh đối kháng thuộc nhóm G_{train}. Nguyên nhân chính là do để sinh ảnh đối kháng thuộc nhóm G_{train}, thực nghiệm cần học mô hình mã hóa tự động. Chi phí để học các mô hình này lớn hơn nhiều so với việc sử dụng mô



Hình 4.3: Ví dụ mẫu sửa điểm ảnh ở biên đối tượng.



Hình 4.4: Ví dụ mẫu bản đồ nổi bật.

hình mã hóa tự động để sinh ảnh đối kháng. Các phương pháp so sánh không có khả năng sinh ảnh đối kháng tức thời như phương pháp PatternAttack. Với một đầu vào mới, các phương pháp so sánh này cần tấn công mô hình lại từ đầu.

Bảng 4.6: Hiệu năng của PatternAttack và các phương pháp khác (giây)

Mô hình	Bộ dữ liệu	ATN khái quát			CW L_2	FGSM	L-BFGS
		Mẫu sửa ở biên	*Mẫu bản đồ nổi bật*	*Mẫu sửa mọi điểm ảnh (ATN)*			
AlexNet $9 \rightarrow 7$	M_{train}	98	131	174	1,815	18	240
	M_{val}	21	38	87	7,176	61	1,009
	M_{new}	11	21	63	1,716	20	258
LeNet-5 $9 \rightarrow 4$	M_{train}	84	109	95	714	11	141
	M_{val}	13	9	55	2,597	40	549
	M_{new}	6	4	46	728	12	128
AlexNet truck $\rightarrow$ horse	C_{train}	216	287	309	2,255	18	289
	C_{val}	26	55	125	8,942	68	1,152
	C_{new}	13	24	99	2,164	17	324
LeNet-5 truck $\rightarrow$ deer	C_{train}	173	181	190	917	42	158
	C_{val}	13	44	53	3,276	195	622
	C_{new}	8	21	47	919	40	148

4.5. Tổng kết

Chương này đã trình bày phương pháp tấn công đối kháng có định hướng cho mô hình tích chập. Phương pháp PatternAttack có hai đóng góp chính gồm ATN khái quát và thuật toán cải thiện chất lượng ảnh đối kháng. Về đề xuất đầu tiên, ATN khái quát cải tiến ATN để sinh ảnh đối kháng dựa theo mẫu thêm nhiễu. Ba loại mẫu thêm nhiễu được đề xuất gồm mẫu sửa mọi điểm ảnh, mẫu sửa điểm ảnh ở biên đối tượng và mẫu bản đồ nổi bật. Ưu điểm của ATN khái quát là sinh ảnh đối kháng có tính đa dạng do nhiễu đối kháng có thể thêm vào những vùng điểm ảnh chuyên biệt có tính chất khác nhau. Về đề xuất thứ hai, thuật toán cải thiện chất lượng ảnh đối kháng sẽ tìm những điểm ảnh chứa nhiễu dư thừa. Mỗi điểm ảnh được gán một giá trị trọng số thể hiện sự ước lượng về nhiễu dư thừa bằng các công thức JSMA$^+$, JSMA$^-$ và COI. Sau đó, thuật toán thử loại bỏ từng khối điểm ảnh dư thừa theo xếp hạng từ trên xuống dưới.

Để đánh giá hiệu quả của PatternAttack, thực nghiệm được tiến hành trên MNIST và CIFAR-10. Về tính đa dạng của ảnh đối kháng, thực nghiệm cho thấy

ATN khái quát có thể sinh ảnh đối kháng thỏa mãn nhiều mẫu thêm nhiễu khác nhau. Đặc biệt đối với mẫu sửa mọi điểm ảnh, hầu hết các tấn công thường đạt tới trên 99% tỉ lệ thành công trung bình. Về chất lượng ảnh đối kháng, theo tiêu chí L_0, thuật toán tham lam có thể loại bỏ hàng trăm điểm ảnh có nhiễu dư thừa về một điểm ảnh có nhiễu dư thừa. Theo tiêu chí L_2, thuật toán tham lam có thể giảm khoảng cách L_2 đáng kể với L-BFGS và FGSM. Kết quả nghiên cứu đã được công bố tại tạp chí Soft Computing (Q2) và hội nghị quốc tế IEEE-RIVF International Conference on Computing and Communication Technologies (bài báo xuất sắc nhất).

Tuy nhiên, hạn chế của PatternAttack là tốc độ cải thiện ảnh đối kháng. Khi điểm ảnh bị thêm nhiễu đối kháng càng nhiều, thuật toán tham lam thể hiện hạn chế về mặt hiệu năng. Thuật toán thử loại bỏ nhiễu đối kháng của từng tập điểm ảnh. Nếu nhãn của ảnh tương ứng khác nhãn gốc thì việc loại bỏ này sẽ được khôi phục lại. Nếu số điểm ảnh có nhiễu đối kháng càng nhiều thì số lần thử bỏ nhiễu càng nhiều, từ đó dẫn đến chi phí có thể khá lớn. Do đó, Chương 5 cải thiện vấn đề này bằng cách đề xuất phương pháp sử dụng mô hình mã hóa tự động kết hợp thuật toán tham lam để cải thiện chất lượng ảnh đối kháng.

Chương 5

Phương pháp sử dụng mô hình mã hóa tự động kết hợp thuật toán tham lam để cải thiện chất lượng ảnh đối kháng

Chương này trình bày phương pháp QI4AE để cải thiện vấn đề tính khái quát hóa và hiệu năng của Thuật toán 4.1. Đầu tiên, luận án trình bày quy trình xây dựng một mô hình mã hóa tự động có khả năng nhận diện các điểm ảnh chứa nhiễu dư thừa trong ảnh đối kháng. Sau đó, luận án trình bày cách sinh ảnh đối kháng và cải thiện chất lượng của những ảnh này sử dụng thuật toán tham lam. Một số kết quả thực nghiệm trên bộ dữ liệu MNIST và CIFAR-10 cho thấy hiệu quả của phương pháp đề xuất.

5.1. Giới thiệu

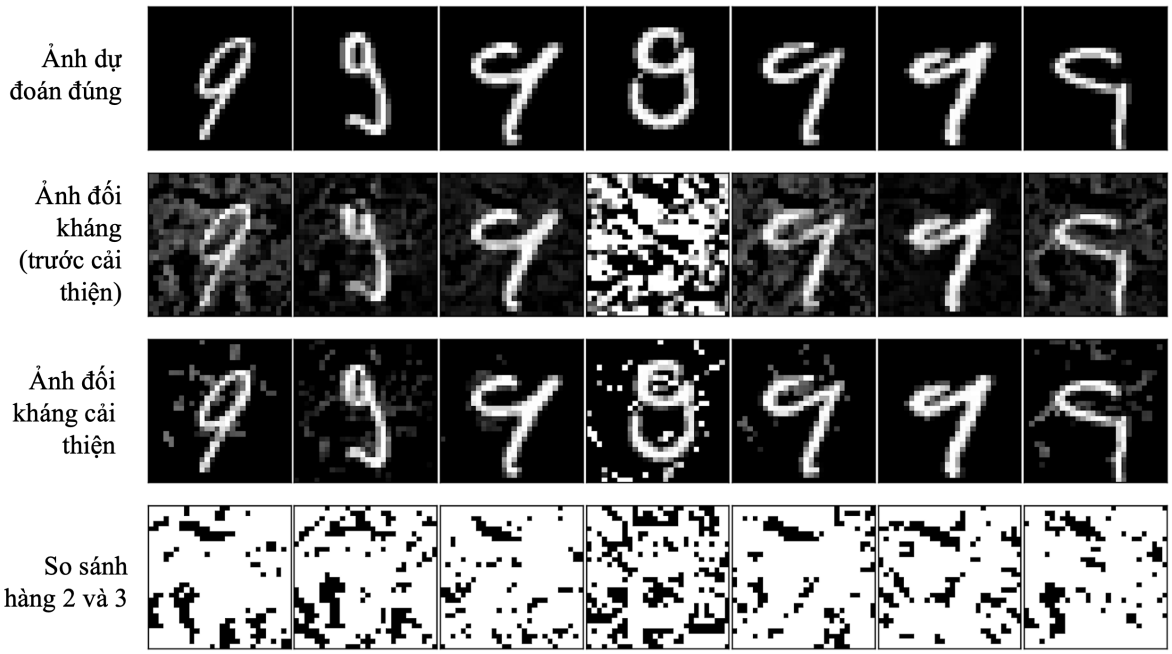
Hiện nay, tấn công đối kháng có định hướng là một hướng phổ biến để kiểm thử tính chắc chắn của mô hình tích chập. Kẻ tấn công biết được kiến trúc mô hình kiểm thử, chọn nhãn đích, sau đó thêm nhiễu đối kháng vào ảnh dự đoán đúng để mô hình kiểm thử nhận diện ảnh bị thêm nhiễu đối kháng thuộc nhãn đích. Các phương pháp tiêu biểu có thể kể đến FGSM [16], CW [17], ATN [18], BIS [19], L-BFGS [32], DeepCheck [42] và DeepFool [33], v.v. Để đánh giá chất lượng ảnh đối kháng, độ đo khoảng cách L_p thường được sử dụng gồm L_0, L_2 và L_∞. Nếu L_p càng lớn thì ảnh đối kháng càng dễ có xu hướng khác ảnh dự đoán đúng và ngược lại. Tuy nhiên, mặc dù các phương pháp này cố gắng tối thiểu

hóa khoảng cách L_p, ảnh đối kháng vẫn có thể chứa lượng lớn nhiễu dư thừa. Nếu loại bỏ nhiễu dư thừa này khỏi ảnh đối kháng, khoảng cách L_p sẽ giảm. Hay nói cách khác, chất lượng ảnh đối kháng sẽ tăng lên.

Ví dụ, FGSM sử dụng khoảng cách L_∞. FGSM thêm nhiễu đối kháng vào mọi điểm ảnh. FGSM tính đạo hàm của hàm mục tiêu với từng điểm ảnh, sau đó lấy dấu của đạo hàm tại điểm ảnh đó, và nhân với một hệ số β tự định nghĩa (Công thức 2.15) để tính nhiễu thêm vào điểm ảnh này. Vấn đề của FGSM là hệ số β được chọn dựa theo cảm tính. Nếu hệ số β quá nhỏ thì lượng nhiễu thêm vào quá ít, dẫn đến tỉ lệ thành công không cao. Nếu hệ số β quá lớn thì tỉ lệ thành công mặc dù cao nhưng ảnh đối kháng trông rất khác xa ảnh dự đoán đúng. L-BFGS và ATN đề xuất hàm mục tiêu có hai thành phần trong đó có thành phần tối thiểu hóa khoảng cách L_2 giữa ảnh dự đoán đúng và ảnh đối kháng. Vấn đề của L-BFGS và ATN là hàm mục tiêu có tính phi tuyến nên nghiệm tìm thấy của hàm mục tiêu có thể là nghiệm cục bộ. Hay nói cách khác khoảng cách L_2 giữa ảnh dự đoán đúng và ảnh đối kháng có thể chưa đủ nhỏ.

Hình 5.1 trình bày một ví dụ cảm hứng của nghiên cứu này. Hàng đầu tiên mô tả các ảnh dự đoán đúng. Hàng thứ hai trình bày ảnh đối kháng sinh bởi phương pháp L-BFGS. Hàng thứ ba thể hiện ảnh đối kháng cải thiện. Sự khác biệt giữa hàng hai và hàng ba được trình bày ở hàng cuối cùng. Các điểm ảnh trắng đại diện sự khác biệt trong khi những điểm ảnh đen thể hiện sự giống hệt. Khi so sánh hàng hai và hàng ba, ảnh đối kháng cải thiện trông rất khác ảnh đối kháng ban đầu. Nói cách khác, ảnh đối kháng ở hàng hai chứa lượng lớn nhiễu dư thừa. Do đó, những ảnh đối kháng sinh bởi L-BFGS nói riêng và các phương pháp khác nói chung nên được loại bỏ nhiễu dư thừa nếu có.

Để giải quyết vấn đề nhiễu dư thừa của các phương pháp tấn công đối kháng, luận án đã đề xuất thuật toán tham lam trình bày trong Thuật toán 4.1. Mặc dù Thuật toán 4.1 có thể cải thiện chất lượng ảnh đối kháng, thuật toán này có hai hạn chế. Hạn chế thứ nhất là không hỗ trợ tính khái quát hóa. Quá trình cải thiện ảnh đối kháng trong quá khứ không được học để cải thiện ảnh đối kháng mới trong tương lai. Hạn chế thứ hai là hiệu năng chưa đủ tốt. Tổng chi phí để thực hiện bước 13 có thể khá tốn kém. Cụ thể, mô hình kiểm thử cần dự đoán nhãn của ảnh đối kháng cải thiện hiện thời. Quá trình này đòi hỏi phải khởi tạo mô hình tích chập và các thao tác tính toán phức tạp để ra được kết quả



Hình 5.1: Ví dụ ảnh đối kháng sinh bởi L-BFGS trước và sau khi cải thiện.

dự đoán. Đặc biệt là chi phí có thể tăng lên đáng kể khi số lượng điểm ảnh đối kháng có nhiễu dư thừa đủ lớn. Ví dụ, trong FGSM, hầu hết các điểm ảnh trên ảnh dự đoán đúng đều được thêm nhiễu đối kháng để sinh ảnh đối kháng. Điều đó có nghĩa là nhiều lần lặp trong vòng `for` tại dòng 5 sẽ được thực thi, dẫn đến chi phí tính toán có thể khá lớn.

Do đó, chương này trình bày cải tiến của Thuật toán 4.1, gọi là QI4AE. Cụ thể, QI4AE gồm hai pha chính gọi là pha xây dựng và pha cải thiện. Trong pha xây dựng, QI4AE xây dựng một mô hình mã hóa tự động có khả năng nhận diện các điểm ảnh chứa nhiễu dư thừa trong ảnh đối kháng. Trong pha cải thiện, ảnh đối kháng sẽ được đẩy vào mô hình mã hóa tự động này để sinh phiên bản ảnh đối kháng cải thiện mức thô. Mặc dù khoảng cách L_0 và L_2 giữa ảnh dự đoán đúng và ảnh đối kháng cải thiện mức thô đã giảm, ảnh đối kháng cải thiện mức thô này vẫn có thể còn nhiễu dư thừa. Do đó, QI4AE áp dụng thuật toán tham lam để loại bỏ nhiễu dư thừa trên ảnh đối kháng cải thiện mức thô để sinh ảnh đối kháng cải thiện mức tinh chế.

Luận án đã tiến hành thực nghiệm trên bộ dữ liệu MNIST và CIFAR-10. Thực nghiệm cho thấy QI4AE có khả năng cải thiện chất lượng ảnh đối kháng đáng kể theo hai độ đo L_0 và L_2. Cụ thể, tỉ lệ giảm nhiễu của L_0 giảm tới 82%

- 95% và của L_2 giảm tới 56% - 81%. Ngoài ra, QI4AE có thể cải thiện chất lượng ảnh đối kháng với chi phí tính toán thấp. Thực nghiệm cho thấy chỉ mất khoảng vài giây để cải thiện chất lượng của 1,000 ảnh đối kháng.

5.2. Các nghiên cứu liên quan

Các phương pháp tấn công đối kháng cố gắng sinh ảnh đối kháng giống ảnh dự đoán đúng hết mức có thể. Ngoài ra, nhãn của ảnh đối kháng bị phân loại sai. Hai nhóm phương pháp chính là cải thiện chất lượng ảnh đối kháng dựa theo đạo hàm và dựa theo bộ giải.

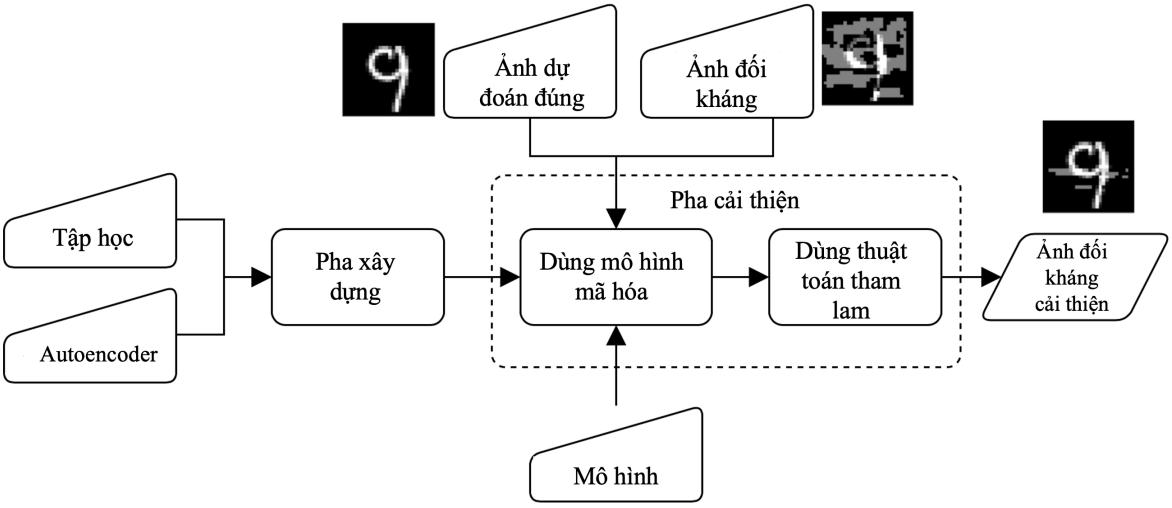
Đối với nhóm tối ưu dựa theo đạo hàm, các phương pháp thuộc nhóm này định nghĩa hàm mục tiêu, trong đó có một thành phần thể hiện khoảng cách giữa ảnh dự đoán đúng và ảnh đối kháng. Sau đó, ảnh dự đoán đúng sẽ được thêm nhiễu đối kháng để tối thiểu hóa hàm mục tiêu này. Các phương pháp tiêu biểu có thể kể đến FGSM [16], CW [17], ATN [18], BIM [19], L-BFGS [32], DeepXplore [43], MI-FGSM [34], PGD [35] và JSMA [96]. Tuy nhiên, bởi vì hàm mục tiêu có tính phi tuyến tính, nghiệm tìm được có thể là nghiệm cục bộ. Điều đó có nghĩa là ảnh đối kháng tìm được có thể chưa đảm bảo khoảng cách đến ảnh dự đoán đúng đủ nhỏ. Hay nói cách khác, ảnh đối kháng có thể vẫn chứa chứa lượng lớn nhiễu dư thừa. Trong các phương pháp này, CW sử dụng phương pháp vét cạn tham số nên chất lượng ảnh đối kháng có xu hướng tốt hơn các phương pháp khác.

Đối với nhóm cải thiện dựa theo bộ giải, các phương pháp thuộc nhóm này chuyển mô hình thành mã nguồn. Các phương pháp tiêu biểu là DeepCheck [42], NEUROSPF [69]. Các phương pháp này sẽ chọn tập điểm ảnh cần thêm nhiễu đối kháng, sau đó tạo hệ ràng buộc có biến là tập điểm ảnh này. Nhóm phương pháp này dùng bộ giải để tìm nghiệm hệ ràng buộc. Tuy nhiên, hạn chế nhóm phương pháp này là thuật toán chọn tập điểm ảnh cần thêm nhiễu đối kháng có thể chưa hợp lý, dẫn đến việc giải hệ ràng buộc không có nghiệm. Từ đó, chi phí tấn công đối kháng tăng lên. Để sinh ảnh đối kháng có chất lượng tốt, phương pháp QI4AE có cách tiếp cận khác biệt với hai nhóm phương pháp nêu trên. Cụ thể, QI4AE được sử dụng như một pha hậu xử lý. Khi triển khai, QI4AE có thể

được tích hợp vào các phương pháp tấn công đối kháng cho mô hình tích chập.

5.3. Phương pháp QI4AE

Luận án đề xuất phương pháp cải thiện các hạn chế của thuật toán tham lam khử nhiễu dư thừa. Tổng quan QI4AE được trình bày trong Hình 5.2. Phương pháp gồm hai pha chính gọi là pha xây dựng và pha cải thiện. Mục đích của pha xây dựng là học một mô hình mã hóa tự động có khả năng nhận diện điểm ảnh có nhiễu dư thừa. Pha cải thiện dùng để loại bỏ nhiễu dư thừa.



Hình 5.2: Tổng quan phương pháp QI4AE.

5.3.1. Pha xây dựng

Đầu vào của pha xây dựng gồm tập học và kiến trúc của mô hình mã hóa tự động kí hiệu là **A**. Đầu ra là một mô hình mã hóa tự động. Tập học là một tập các cặp, trong đó một cặp kí hiệu là $(\mathbf{x}', s_{\mathbf{x}'})$, trong đó $\mathbf{x}' \in \mathbb{R}^d$ là ảnh đối kháng chưa cải thiện và $s_{\mathbf{x}'} \in \mathbb{R}^d$ là véc tơ 0 - 1. Mỗi giá trị $s_{\mathbf{x}'}[i]$ mô tả xác suất nên loại bỏ nhiễu đối kháng khỏi điểm ảnh x'_i (Công thức 5.1).

$$s_{\mathbf{x}'}[i] = \begin{cases} 1 & \text{nếu } x'_i \text{ dư thừa} \\ 0 & \text{ngược lại} \end{cases} \tag{5.1}$$

Ví dụ, xét ảnh đối kháng chưa cải thiện $\mathbf{x}' = [0, 1, 1, ..., 0.5, 0]^T \in \mathbb{R}^{784}$, ảnh dự đoán đúng tương ứng là $\mathbf{x} = [\mathbf{1}, \mathbf{0}, 1, ..., 0.5, 0]^T \in \mathbb{R}^{784}$. Sau khi loại bỏ nhiễu đối kháng khỏi x'_0, nếu ảnh đối kháng vẫn có nhãn là nhãn đích thì x'_0 có nhiễu đối kháng. Do đó, $s_{\mathbf{x}'}[0]$ được gán bằng một. Ngược lại, nếu ảnh đối kháng có nhãn giống ảnh dự đoán đúng thì $s_{\mathbf{x}'}[0]$ được gán bằng không.

Kiến trúc của mô hình mã hóa tự động $\mathbf{A}$ được định nghĩa thủ công. Đầu ra của mô hình mã hóa tự động được kí hiệu $\mathbf{A}(\mathbf{x}')$. Với mỗi ảnh đối kháng $\mathbf{x}'$ trong tập học, hàm mục tiêu của mô hình mã hóa tự động được định nghĩa như Công thức 5.2.

$$\frac{1}{d}\sum_{i=0}^{d-1} \mathrm{CE}(s_{\mathbf{x}'}[i], \mathbf{A}(\mathbf{x}')[i]) \tag{5.2}$$

trong đó, CE là cross-entropy nhị phân.

Mô hình mã hóa tự động trong Công thức 5.2 khác với mô hình mã hóa tự động truyền thống. Điểm khác biệt là đầu ra $\mathbf{A}(\mathbf{x}')$. Trong mô hình mã hóa tự động truyền thống, $\mathbf{A}(\mathbf{x}')$ cần giống đầu vào $\mathbf{x}'$. Vì thế, độ đo sai số bình phương trung bình thường được sử dụng để đo khoảng cách giữa $\mathbf{x}'$ và $\mathbf{A}(\mathbf{x}')$. Tuy nhiên, trong QI4AE, $\mathbf{A}(\mathbf{x}')$ là một véc tơ xác suất. Do đó, luận án sử dụng cross-entropy nhị phân để so sánh $\mathbf{A}(\mathbf{x}')$ và $s_{\mathbf{x}'}$.

5.3.2. Pha cải thiện

Pha này cải thiện chất lượng ảnh đối kháng theo hai tiêu chí L_0 và L_2. Đầu vào là mô hình kiểm thử $\mathbf{M}$, mô hình mã hóa tự động $\mathbf{A}$ của pha xây dựng, ảnh đối kháng chưa cải thiện $\mathbf{x}'$ và ảnh dự đoán đúng $\mathbf{x}$. Đầu ra là ảnh đối kháng cải thiện. Pha này có hai bước gồm bước cải thiện dùng mô hình mã hóa tự động và bước cải thiện dùng thuật toán tham lam.

Bước cải thiện dùng mô hình mã hóa tự động: Trong bước này, mô hình mã hóa tự động $\mathbf{A}$ được sử dụng để sinh ảnh đối kháng cải thiện mức thô kí hiệu là $\mathbf{x}'_1$. Bởi vì $\mathbf{A}(\mathbf{x}')$ là một véc tơ xác suất, QI4AE cần chọn một ngưỡng

$\delta \in [0, 1]$ để xác định liệu $\mathbf{A}(\mathbf{x}')[i]$ là dư thừa hay không. Cụ thể, ảnh đối kháng cải thiện mức thô $\mathbf{x}'_1$ được tạo như Công thức 5.3.

$$\mathbf{x}'_1[i] = \begin{cases} x'_i & \text{nếu } \mathbf{A}(\mathbf{x}')[i] <= \delta \\ x_i & \text{ngược lại} \end{cases} \tag{5.3}$$

Sau đó, ảnh đối kháng cải thiện mức thô $\mathbf{x}'_1$ được dự đoán bởi mô hình kiểm thử $\mathbf{M}$. Nếu nhãn của $\mathbf{x}'_1$ giống nhãn của $\mathbf{x}'$, $\mathbf{x}'_1$ sẽ được đẩy vào bước cải thiện dùng thuật toán tham lam. Ngược lại, giá trị ngưỡng δ nên tăng lên. Khi δ tăng, số điểm ảnh đối kháng được phân loại là có nhiễu dư thừa sẽ giảm đi. Kết quả là $\mathbf{x}'_1$ có xu hướng giống $\mathbf{x}'$ hơn theo độ đo L_p. Mô hình kiểm thử $\mathbf{M}$ có xu hướng đoán nhãn của $\mathbf{x}'_1$ và $\mathbf{x}'$ giống nhau. Bước cải thiện dùng mô hình mã hóa tự động có chi phí tính toán thấp. Thực nghiệm cho thấy mô hình mã hóa tự động cần khoảng 1 - 2 giây để cải thiện khoảng 1,000 ảnh đối kháng chưa cải thiện sinh từ MNIST và khoảng 3 - 4 giây cho khoảng 1,000 ảnh đối kháng chưa cải thiện từ CIFAR-10.

Bước cải thiện dùng thuật toán tham lam: Thực nghiệm quan sát rằng ảnh đối kháng cải thiện mức thô $\mathbf{x}'_1$ vẫn có thể chứa nhiễu dư thừa. Do đó, QI4AE cần tiếp tục tìm nhiễu dư thừa này và loại bỏ khỏi $\mathbf{x}'_1$. Luận án sử dụng thuật toán tham lam khử nhiễu dư thừa được trình bày trong Phần 4.1. Đầu ra của bước này là ảnh đối kháng cải thiện mức tinh chế kí hiệu là $\mathbf{x}'_2$. Thuật toán trả về $\mathbf{x}'_2$ là ảnh đối kháng cải thiện cuối cùng và thuật toán kết thúc.

5.4. Thực nghiệm

Để chứng minh hiệu quả của QI4AE, thực nghiệm trả lời các câu hỏi sau đây:

- **RQ1 - Tỉ lệ thành công của mô hình mã hóa tự động**: Đánh giá mô hình mã hóa tự động có thể cải thiện chất lượng của bao nhiêu ảnh đối kháng chưa cải thiện?
- **RQ2 - Chất lượng**: Đánh giá tỉ lệ giảm nhiễu theo tiêu chí L_0 và L_2?

- **RQ3 - Hiệu năng**: Đánh giá hiệu năng của QI4AE khi cải thiện chất lượng ảnh đầu vào mới?

Đối với bước xây dựng mô hình mã hóa tự động, luận án trả lời câu hỏi *RQ1* để đánh giá tính hiệu quả của mô hình mã hóa tự động. Đối với bước cải thiện, luận án trả lời câu hỏi *RQ2* và *RQ3* bằng cách so sánh với thuật toán tham lam khử nhiễu dư thừa. Thực nghiệm được tiến hành trên Google Colab.

5.4.1. Cấu hình

5.4.1.1. Mô hình kiểm thử

Thực nghiệm này sử dụng hai mô hình kiểm thử mô tả trong Bảng 4.1. Mô hình đầu tiên là LeNet-5 học trên bộ dữ liệu MNIST có độ chính xác trên tập học là 99.86% và trên tập kiểm thử là 98.82%. Mô hình thứ hai là AlexNet học trên bộ dữ liệu CIFAR-10 có độ chính xác trên tập học là 99.70% và trên tập kiểm thử là 76.22%.

5.4.1.2. Sinh ảnh đối kháng

Các ảnh đối kháng được sinh bằng cách áp dụng ba tấn công đối kháng gồm FGSM, L-BFGS và ATN. Trong khi FGSM sử dụng độ đo L_∞, L-BFGS và ATN sử dụng độ đo L_2. Đối với FGSM và L-BFGS, thực nghiệm sử dụng công thức được đề xuất trong bài báo gốc. Đối với ATN, thực nghiệm chỉnh sửa Công thức 2.18 để sinh ảnh đối kháng có chất lượng tốt hơn. Cụ thể, thực nghiệm dùng L_2 thay vì cross-entropy để đo khoảng cách giữa xác suất mong đợi và xác suất thực tế của ảnh đối kháng. Nếu ảnh dự đoán đúng có nhãn m được sửa thành ảnh đối kháng có nhãn n, thực nghiệm kí hiệu là tấn công $m \rightarrow n$. Đối với MNIST, tấn công được sử dụng là $9 \rightarrow 4$. Đối với CIFAR-10, thực nghiệm sử dụng tấn công $truck \rightarrow horse$.

5.4.1.3. Phương pháp QI4AE

Pha xây dựng: Trong pha đầu tiên của QI4AE, luận án xây dựng các mô hình mã hóa tự động để nhận diện điểm ảnh có nhiễu dư thừa. Mỗi mô hình mã hóa tự động được học từ 6,000 ảnh sử dụng Công thức 5.2. Tập học được kí hiệu là $\mathbf{X}_{train}$. Các mô hình mã hóa tự động được học với 300 lần lặp.

Bảng 5.1: Kiến trúc của mô hình mã hóa tự động sử dụng trong thực nghiệm

Kiểu tầng	Cấu hình
Đầu vào	#width × #height × #channel
Conv2D + ReLU	kernel = (3, 3), #filters = 64
Conv2D + ReLU	kernel = (3, 3), #filters = 64
BatchNormalization	
MaxPooling	pool_size = (2, 2)
Conv2D + ReLU	kernel = (3, 3), #filters = 32
Conv2D + ReLU	kernel = (3, 3), #filters = 32
BatchNormalization	
MaxPooling2D	pool_size = (2, 2)
Conv2D + ReLU	kernel = (3, 3), #filters = 32
Conv2D + ReLU	kernel = (3, 3), #filters = 32
BatchNormalization	
UpSampling2D	pool_size = (2, 2)
Conv2D + ReLU	kernel = (3, 3), #filters = 64
Conv2D + ReLU	kernel = (3, 3), #filters = 64
BatchNormalization	
UpSampling2D	pool_size = (2, 2)
Conv2D + Sigmoid	kernel = (3, 3), #filters = #channel
Đầu ra	#width × #height × #channel

Kiến trúc của mô hình mã hóa tự động nên được định nghĩa dựa theo kinh nghiệm của người kiểm thử. Thực nghiệm sử dụng kiến trúc mô hình mã hóa tự động tích chập xếp chồng vì kiến trúc này có thể học đặc trưng không gian của ảnh 2D và 3D tốt hơn mô hình mã hóa tự động truyền thống như mô hình mã hóa tự động thưa. Thực nghiệm đã thử với nhiều kiến trúc (cùng với các giá trị siêu tham số) khác nhau và chọn kiến trúc tốt nhất trong các lần thử

nghiệm. Bảng 5.1 mô tả kiến trúc của các mô hình mã hóa tự động này, trong đó *#width*, *#height*, *#channel* lần lượt là chiều rộng, chiều cao và số chiều của ảnh đầu vào. Với MNIST, kích thước đầu vào là $28 \times 28 \times 1$. Với CIFAR-10, kiến trúc đầu vào là $32 \times 32 \times 3$. Tất cả mọi mô hình mã hóa tự động đều có kiến trúc giống nhau ngoại trừ tầng đầu vào và tầng đầu ra. Tầng trước tầng đầu ra sử dụng hàm sigmoid để tính xác xuất phân lớp nhị phân.

Pha cải thiện: Thực nghiệm sử dụng 1,000 ảnh đối kháng kí hiệu là $\mathbf{X}_{test}$ để đánh giá hiệu quả của QI4AE. Các ảnh đối kháng này không được sử dụng để xây dựng mô hình mã hóa tự động. Cấu hình pha cải thiện được trình bày như sau:

- **Bước cải thiện dùng mô hình mã hóa tự động**: Các ảnh đối kháng chưa cải thiện được đẩy vào mô hình mã hóa tự động đã học từ pha trước.
- **Bước cải thiện dùng thuật toán tham lam**: Với MNIST, giá trị bước nhảy α (trong Thuật toán 4.1) bằng sáu. Bởi vì số đặc trưng của CIFAR-10 lớn hơn nhiều so với MNIST, bước nhảy α nên được khởi tạo với giá trị lớn hơn. Đối với CIFAR-10, thực nghiệm đã thử nhiều giá trị α khác nhau và nhận thấy rằng nên sử dụng giá trị α khởi tạo bằng 60.

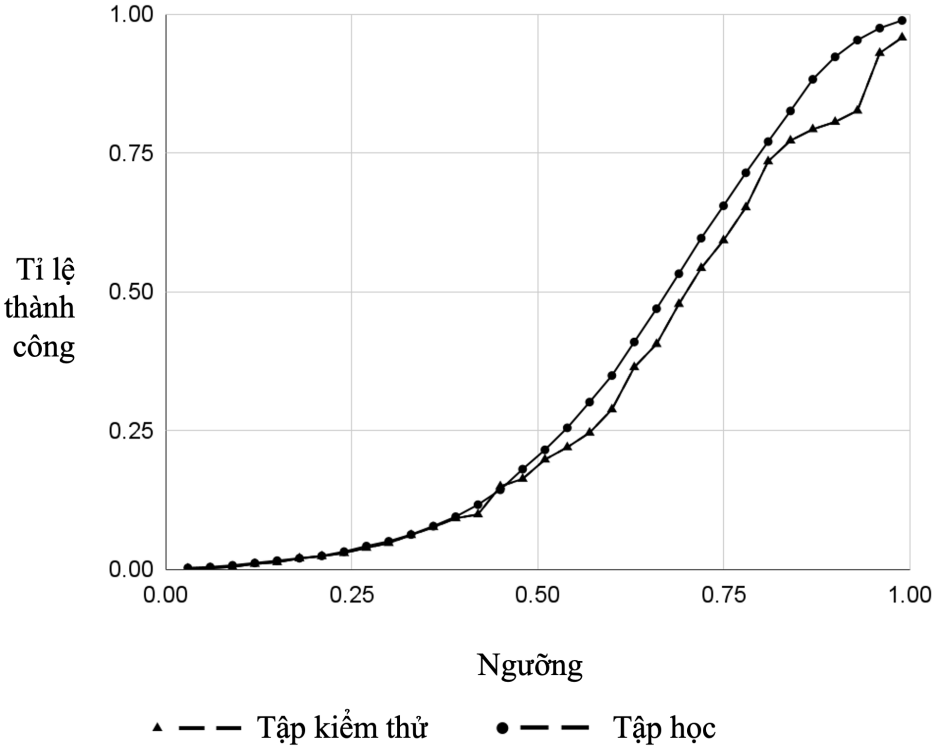
5.4.1.4. Phương pháp cơ sở

Phương pháp QI4AE được so sánh với thuật toán tham lam khử nhiễu dư thừa. Đầu vào của thuật toán tham lam khử nhiễu dư thừa là 1,000 ảnh đối kháng chưa cải thiện. Các ảnh đối kháng này giống hệt với ảnh đối kháng sử dụng trong pha cải thiện của QI4AE. Ngoài ra, cấu hình α của thuật toán tham lam khử nhiễu dư thừa giống hệt với cấu hình ở bước cải thiện dùng thuật toán tham lam của thuật toán tham lam khử nhiễu dư thừa. Cụ thể, với MNIST, giá trị khởi tạo của α bằng sáu. Với CIFAR-10, α có giá trị bởi tạo bằng 60.

5.4.2. Kết quả

5.4.2.1. RQ1 - Tỉ lệ thành công của mô hình mã hóa tự động

Phần này đánh giá tỉ lệ thành công của mô hình mã hóa tự động. Các mô hình mã hóa tự động này được xây dựng từ $\mathbf{X}_{train}$ chứa khoảng 6,000 ảnh và đánh giá trên $\mathbf{X}_{test}$ chứa khoảng 1,000 ảnh. Tỉ lệ thành công là phần trăm ảnh đối kháng chưa cải thiện được cải tiến chất lượng thành công. Nếu tỉ lệ thành công bằng 100%, toàn bộ ảnh đối kháng chưa cải thiện đều được cải thiện thành công.



Hình 5.3: Xu hướng của tỉ lệ thành công khi cải thiện ảnh đối kháng sinh bởi FGSM sử dụng các ngưỡng khác nhau.

Thực nghiệm cho thấy tăng giá trị ngưỡng δ trong Công thức 5.3 giúp tăng tỉ lệ thành công. Nguyên nhân chính là do khi giá trị δ tăng, có ít điểm ảnh đối kháng được loại bỏ nhiễu đối kháng. Nói cách khác, ảnh đối kháng cải thiện có xu hướng gần với ảnh đối kháng chưa cải thiện theo độ đo L_0 và L_2. Kết quả là, nhãn của ảnh đối kháng cải thiện có xu hướng giống với nhãn của ảnh đối kháng chưa cải thiện. Ví dụ, Hình 5.3 trình bày ảnh hưởng của ngưỡng lên tỉ lệ thành công. Trong đó, ảnh đối kháng được sinh bởi FGSM từ các ảnh thuộc

bộ dữ liệu MNIST. Mô hình kiểm thử là LeNet-5. Có thể thấy rằng tỉ lệ thành công tăng gần tới 100% khi giá trị δ tăng dần đến một.

Thực nghiệm tiếp tục đánh giá ảnh hưởng của ba giá trị δ gồm 0.93, 0.96 và 0.99 lên tỉ lệ thành công. Lý do sử dụng các giá trị δ này là do chúng có tỉ lệ thành công cao. Thực nghiệm tính toán tỉ lệ thành công trung bình thay vì trình này tỉ lệ thành công của từng ngưỡng. Bảng 5.2 trình bày tỉ lệ thành công trung bình của các mô hình mã hóa tự động. Ví dụ, 97.2% nghĩa là mô hình mã hóa tự động có thể cải thiện chất lượng của khoảng $|\mathbf{X}_{train}| \times 97.2\%$ ảnh đối kháng chưa cải thiện. Bảng thực nghiệm cho thấy các mô hình mã hóa tự động có thể cải thiện chất lượng của hầu hết ảnh đối kháng chưa cải thiện trong tập $\mathbf{X}_{train}$, từ khoảng 87.3% đến khoảng 97.2%. Đối với tập $\mathbf{X}_{test}$, tỉ lệ thành công từ 81% tới khoảng 92.2%. Điều này cho thấy các mô hình mã hóa tự động này có thể nhận diện nhiễu dư thừa trong ảnh đối kháng chưa cải thiện hiệu quả.

Bảng 5.2: Tỉ lệ thành công trung bình của các mô hình mã hóa tự động

Bộ dữ liệu	Phương pháp	Tập học	Tập kiểm thử
MNIST	FGSM	97.2%	90.4%
	L-BFGS	89.3%	87.1%
	ATN	95.2%	92.2%
CIFAR-10	FGSM	87.4%	81.3%
	L-BFGS	87.3%	83.3%
	ATN	87.7%	86.4%

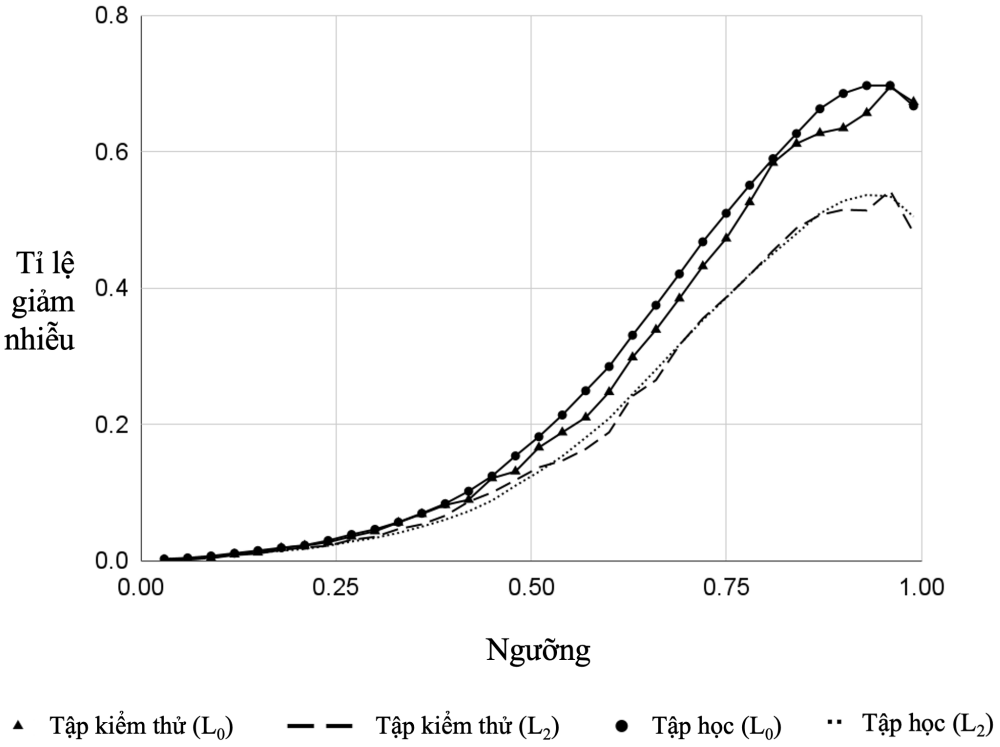
Các mô hình mã hóa tự động trong *RQ1* được sử dụng để trả lời câu hỏi *RQ2* và *RQ3*. Hai câu hỏi nghiên cứu này đánh giá hiệu quả của pha cải thiện. Xét cùng tập ảnh đối kháng chưa cải thiện, thực nghiệm cần đánh giá hiệu quả của pha cải thiện so với thuật toán tham lam khử nhiễu dư thừa. Các tiêu chí so sánh là L_0, L_2 và hiệu năng.

5.4.2.2. RQ2 - Khả năng cải thiện chất lượng

Phần này đánh giá hiệu quả của pha cải thiện chất lượng trên tập $\mathbf{X}_{test}$ theo hai tiêu chí L_0 và L_2. Các ảnh đối kháng không được sử dụng để học mô hình mã hóa tự động. Phương pháp so sánh là thuật toán tham lam khử nhiễu dư

thừa trình bày trong phương pháp PatternAttack.

Trong pha cải thiện, quy trình cải tiến ảnh đối kháng chưa cải thiện được thực hiện như sau. Đầu tiên, $\mathbf{X}_{test}$ được đẩy vào bước cải thiện dùng mô hình mã hóa tự động để sinh ảnh đối kháng cải thiện mức thô. Sau đó, ảnh đối kháng cải thiện mức thô được đẩy vào thuật toán tham lam và trả về ảnh đối kháng cải thiện mức tinh chế. Tham số δ của mô hình mã hóa tự động nên được chọn để cải thiện chất lượng của $\mathbf{X}_{test}$ nhiều nhất có thể. Thực nghiệm quan sát rằng chọn $\delta \in [0.8, 1)$ sẽ có tỉ lệ giảm nhiễu của L_0 và L_2 khá tốt. Ví dụ, Hình 5.4 mô tả ảnh hưởng của ngưỡng δ đến tỉ lệ giảm nhiễu. Trong đó, ảnh đối kháng được sinh bởi FGSM và mô hình kiểm thử là LeNet-5 được học từ MNIST. Hình này cho thấy khi tăng ngưỡng δ tới khoảng 0.9x sẽ cải thiện tỉ lệ giảm nhiễu của L_0 và L_2 đáng kể. Khi δ gần chạm đến một, tỉ lệ giảm nhiễu giảm hơi nhẹ.



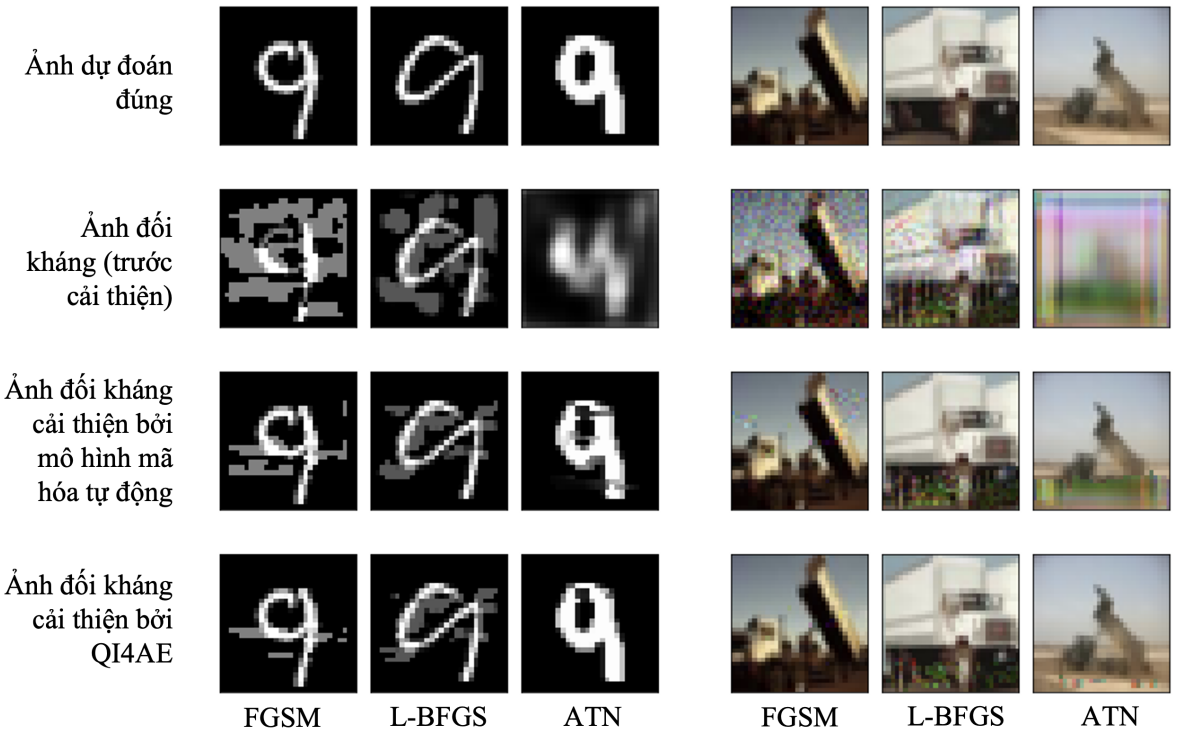
Hình 5.4: Xu hướng của tỉ lệ giảm nhiễu khi sử dụng các ngưỡng δ khác nhau.

Dựa theo quan sát trên, thực nghiệm chọn giá trị δ sử dụng trong RQ1 (0.93, 0.96 và 0.99) để so sánh với thuật toán tham lam khử nhiễu dư thừa. Bảng 5.3 thể hiện tỉ lệ giảm nhiễu trung bình của L_0 và L_2. Giá trị tốt hơn được in đậm. Kí hiệu * ám chỉ thực nghiệm bỏ qua quá trình xây dựng mô hình mã hóa tự động. Nói chung, pha cải thiện có kết quả tốt hơn thuật toán tham lam khử

nhiễu dư thừa trong 8/12 trường hợp. Đối với L_0, tỉ lệ giảm nhiễu trung bình từ khoảng 82.38% đến khoảng 95.20%. Đối với L_2, tỉ lệ giảm nhiễu trung bình trong khoảng 59.67% đến xấp xỉ 81.07%.

Bảng 5.3: Tỉ lệ giảm nhiễu trung bình của L_0 và L_2 trên $\mathbf{X}_{test}$

Bộ dữ liệu	Phương pháp	L_0		L_2	
		Đề xuất*	Thuật toán 4.1	Đề xuất*	Thuật toán 4.1
MNIST	FGSM	**82.58%**	82.40%	**68.91%**	68.07%
	L-BFGS	**82.38%**	82.30%	68.01%	**72.79%**
	ATN	**95.20%**	94.27%	**59.67%**	56.77%
CIFAR-10	FGSM	**85.27%**	84.92%	**81.07%**	79.31%
	L-BFGS	83.90%	**87.14%**	63.8%	**66.54%**
	ATN	86.45%	**88.69%**	**68.27%**	66.09%



Hình 5.5: Ví dụ ảnh trước và sau khi loại bỏ nhiễu đối kháng trong bộ dữ liệu MNIST và CIFAR-10.

Hình 5.5 trình bày một vài ví dụ của ảnh đối kháng cải thiện sinh bởi pha cải thiện. Hàng đầu tiên mô tả ảnh dự đoán đúng. Kết quả của tấn công đối kháng trên những ảnh này được mô tả ở hàng thứ hai và những ảnh ở hàng này chưa được cải thiện chất lượng bởi pha cải thiện. Hàng ba mô tả ảnh đối

kháng cải thiện mức thô sinh bởi bước cải thiện dùng mô hình mã hóa tự động. Hàng cuối cùng trình bày kết quả áp dụng thuật toán tham lam khử nhiễu dư thừa với ảnh đối kháng cải thiện mức thô. Hình này cho thấy dưới cảm quan con người, bước cải thiện dùng mô hình mã hóa tự động có thể cải thiện chất lượng của ảnh đối kháng chưa cải thiện đáng kể. Tuy nhiên, ảnh sinh bởi bước này vẫn tồn tại nhiễu dư thừa ở hàng ba. Bằng cách áp dụng thuật toán tham lam khử nhiễu dư thừa, đa số những nhiễu dư thừa này sẽ bị loại bỏ, từ đó nâng cao chất lượng ảnh đối kháng.

5.4.2.3. RQ3 - Hiệu năng

Phần này đánh giá hiệu năng của pha cải thiện khi gặp ảnh mới. Tập các ảnh mới này được kí hiệu là $\mathbf{X}_{test}$. Tập ảnh đối kháng được dùng ở RQ2. Thực nghiệm sử dụng các giá trị khác nhau của ngưỡng δ (0.93, 0.96 và 0.99) đã dùng ở hai câu hỏi nghiên cứu trước đó.

Bảng 5.4 trình bày hiệu năng của pha cải thiện trong QI4AE và thuật toán tham lam khử nhiễu dư thừa. Chi phí tính toán của pha cải thiện gồm bước cải thiện dùng mô hình mã hóa tự động và bước cải thiện dùng thuật toán tham lam. Trong bước cải thiện dùng mô hình mã hóa tự động, $\mathbf{X}_{test}$ được đẩy vào mô hình mã hóa tự động để lấy ảnh đối kháng cải thiện mức thô. Bởi vì ảnh đối kháng cải thiện mức thô này có thể có nhiễu dư thừa, thuật toán tham lam được sử dụng để loại bỏ nhiễu này và trả về ảnh đối kháng cải thiện mức tinh chế.

Bảng 5.4: Hiệu năng trung bình của pha cải thiện trong QI4AE và thuật toán tham lam khử nhiễu dư thừa (giây)

Bộ dữ liệu	Phương pháp	QI4AE			Thuật toán 4.1
		Mô hình mã hóa tự động	Thuật toán 4.1	$\sum$	
MNIST	FGSM	1.1	13.1	**14.2**	80.3
	L-BFGS	1.2	27.6	**28.8**	98.3
	ATN	1.4	20.1	**21.5**	120.2
CIFAR-10	FGSM	3.7	29.1	**32.8**	250.8
	L-BFGS	3.5	46.1	**49.6**	261.1
	ATN	3.4	62.1	**65.5**	254.5

Như có thể thấy, hiệu năng của pha cải thiện tốt hơn hẳn thuật toán tham lam khử nhiễu dư thừa. Trên MNIST, pha cải thiện thường cần khoảng 14.2 giây đến khoảng 28.8 giây, nhanh hơn khoảng ba lần đến sáu lần so với thuật toán tham lam khử nhiễu dư thừa. Trên CIFAR-10, hiệu năng của pha cải thiện nhanh gấp khoảng bốn lần đến tám lần so với thuật toán tham lam khử nhiễu dư thừa. Nguyên nhân chính là do trong pha cải thiện, hầu hết các điểm ảnh đối kháng được loại bỏ nhiễu dư thừa bằng cách sử dụng mô hình mã hóa tự động. Kết quả là, chỉ có một tập nhỏ nhiễu dư thừa được loại bỏ trong bước cải thiện dùng mô hình mã hóa tự động. Từ đó, chi phí của bước cải thiện dùng mô hình mã hóa tự động được giảm thiểu.

5.5. Tổng kết

Chương này đã trình bày phương pháp để cải thiện chất lượng ảnh đối kháng. Phương pháp này là cải tiến của Thuật toán 4.1. Phương pháp QI4AE có hai pha chính gồm pha xây dựng và pha cải thiện. Trong pha đầu tiên, mô hình mã hóa tự động được học để tìm những điểm ảnh có nhiễu dư thừa. Trong pha thứ hai, ảnh sẽ được đẩy qua mô hình mã hóa tự động để loại bỏ nhiễu dư thừa. Tuy nhiên, bởi vì ảnh có thể vẫn còn nhiễu dư thừa, ảnh tiếp tục được xử lý bởi thuật toán tham lam.

Thực nghiệm trên MNIST và CIFAR-10 cho thấy phương pháp QI4AE có thể cải thiện chất lượng ảnh đối kháng theo tiêu chí L_0 và L_2 đáng kể. Ngoài ra, chi phí khi triển khai QI4AE thấp hơn nhiều so với Thuật toán 4.1. Phương pháp QI4AE có thể áp dụng như một pha hậu xử lý với nhiều phương pháp tấn công đối kháng khác. Kết quả nghiên cứu được công bố tại hội nghị quốc tế International Conference on Agents and Artificial Intelligence (ICAART - rank B).

Chương 6

Phương pháp sử dụng mô hình mã hóa tự động để cải thiện tính chắc chắn của mô hình tích chập

Chương này trình bày phương pháp SCADefender để cải thiện tính chắc chắn của mô hình tích chập. Tư tưởng chung của SCADefender là học một mô hình mã hóa tự động phòng thủ để loại bỏ nhiễu đối kháng trong ảnh đầu vào. Luận án đề xuất phương pháp tấn công đối kháng không định hướng sinh ảnh đối kháng phục vụ cho học mô hình mã hóa tự động phòng thủ. Một số kết quả thực nghiệm trên bộ dữ liệu MNIST, Fashion-MNIST và CIFAR-10 cho thấy hiệu quả của phương pháp đề xuất.

6.1. Giới thiệu

Nhiều phương pháp tấn công đối kháng không định hướng đã được đề xuất để đánh giá tính chắc chắn của mô hình tích chập như CW [17], ATN [18], DeepCheck [42], DeepFool [33], RP_2 [105], EAD [106], v.v. Với đầu vào là ảnh dự đoán đúng, các phương pháp tấn công đối kháng không định hướng sẽ thêm nhiễu đối kháng vào này để mô hình kiểm thử nhận diện sai nhãn. Một trong những độ đo phổ biến để đánh giá tính chắc chắn của một mô hình kiểm thử trước sự tấn công của một phương pháp tấn công đối kháng không định hướng được gọi là tỉ lệ thành công. Sau khi đánh giá được tính chắc chắn của mô hình tích chập, một trong những nhiệm vụ quan trọng kế tiếp là cải thiện tính chắc chắn của mô hình đó. Để đánh giá chất lượng của phương pháp cải thiện

tính chắc chắn, tỉ lệ phát hiện được sử dụng phổ biến [46, 49, 50, 107]. Các hướng cải thiện tính chắc chắn cho mô hình tích chập phổ biến gồm *(i)* xây dựng lại mô hình kiểm thử [16, 35, 44, 45], *(ii)* xây dựng một mô hình phân lớp để nhận diện ảnh dự đoán đúng và ảnh đối kháng [46, 47, 48], *(iii)* loại bỏ nhiễu đối kháng khỏi ảnh đầu vào [46, 49, 50, 51]. Đối với cách tiếp cận *(iii)*, các phương pháp kinh điển có thể kể đến MagNet [46], PuVAE [49] và Defense-VAE [50].

Cụ thể, đối với MagNet, phương pháp này đề xuất bộ nhận diện và bộ khôi phục. Bộ nhận diện có kiến trúc mô hình mã hóa tự động tích chập xếp chồng và được sử dụng để đánh giá tính tự nhiên của ảnh đầu vào. Mặc định, MagNet coi các ảnh thuộc tập học của mô hình kiểm thử có tính tự nhiên. Tập học của bộ nhận diện là tập học của mô hình kiểm thử và có bổ sung thêm các ảnh có nhiễu Gaussian nhỏ. Bộ nhận diện được học để chuyển các ảnh đầu vào, có thể có nhiễu, về phiên bản có tính tự nhiên. Khi triển khai, với một ảnh đầu vào cụ thể, bộ nhận diện sẽ chuyển ảnh đầu vào thành một phiên bản khác. Để xác định một ảnh đầu vào có tính tự nhiên, MagNet có hai cách. Cách đầu tiên là so sánh khoảng cách L_2 giữa ảnh đầu vào và phiên bản. Cách thứ hai là so sánh hội tụ Kullback-Leibler của hai véc tơ xác suất dự đoán của ảnh đầu vào và phiên bản. Giá trị hội tụ Kullback-Leibler hoặc L_2 càng nhỏ thì ảnh đầu vào càng có khả năng là ảnh tự nhiên. Nếu bộ nhận diện dự đoán ảnh đầu vào không có tính tự nhiên thì mô hình kiểm thử sẽ không nhận diện ảnh này. Ngược lại, nếu ảnh đầu vào được dự đoán có tính tự nhiên thì ảnh đầu vào này tiếp tục được chuyển qua bộ khôi phục để tạo một ảnh mới. Trong đó, ảnh mới này được kì vọng nhận diện chính xác hơn bởi mô hình kiểm thử.

Khác với MagNet, PuVAE và Defense-VAE chỉ dùng bộ khôi phục để loại bỏ nhiễu đối kháng trong ảnh đầu vào. Các bộ khôi phục này sử dụng kiến trúc mô hình mã hóa tự động biến thiên có điều kiện như PuVAE hoặc mô hình mã hóa tự động biến thiên như Defense-VAE. Nhóm tác giả giả định rằng phân phối của không gian ẩn là Gaussian đa biến. Cụ thể, ảnh đầu vào sẽ được chuyển về miền không gian ẩn. Sau đó, PuVAE và Defense-VAE lấy một mẫu trong miền không gian ẩn này sử dụng kĩ thuật lấy mẫu như kĩ thuật tái tham số hóa, sau đó sinh ảnh đầu ra từ mẫu này. Ảnh đầu ra được kì vọng sẽ không chứa nhiễu đối kháng. Thay vì lấy kết quả dự đoán của mô hình kiểm thử với ảnh đầu vào, hai phương pháp này lấy kết quả dự đoán với ảnh đầu ra.

Tuy nhiên, ba phương pháp này chưa loại bỏ nhiễu đối kháng đủ tốt đối với ảnh đầu vào có nhiễu đối kháng đa dạng. Trong thực tế, luận án nhận thấy rằng nhiễu đối kháng có tính bất định, hay nói cách khác, nhiễu đối kháng rất khó có loại phân phối cụ thể đối với mọi phương pháp tấn công đối kháng không định hướng. Cụ thể, một ảnh dự đoán đúng có nhiều cách thêm nhiễu đối kháng để tạo thành ảnh đối kháng. Mỗi phương pháp tấn công đối kháng không định hướng sẽ thêm các loại nhiễu đối kháng khác nhau vào ảnh dự đoán đúng. Ví dụ, trong khi một vài phương pháp tấn công đối kháng không định hướng thêm nhiễu đối kháng vào số điểm ảnh ít nhất có thể như CW L_0 [17] và DeepCheck [67], một số phương pháp tấn công đối kháng không định hướng khác thêm nhiễu đối kháng vào mọi điểm ảnh như FGSM và MI-FGSM [34]. Từ quan sát về tính bất định của nhiễu đối kháng, điểm yếu của MagNet, PuVAE và Defense-VAE được bộc lộ. Đối với MagNet, bộ nhận diện được học từ tập học của mô hình kiểm thử có nhiễu Gaussian nhỏ và tập học gốc. Vì thế, mặc dù MagNet có thể nhận diện ảnh đầu vào không có nhiễu đối kháng hoặc có nhiễu Gaussian, phương pháp này thường khó nhận diện được ảnh đầu vào có nhiễu đối kháng không phải nhiễu Gaussian. Đối với PuVAE và Defense-VAE, hai phương pháp này giả sử phân phối của không gian ẩn là một phân phối Gaussian. Do đó, ba phương pháp này có thể dễ dàng nhận diện sai nhãn của các ảnh đầu vào chứa nhiễu đối kháng không phải Gaussian.

Để giảm thiểu vấn đề nêu trên, nghiên cứu này đề xuất phương pháp SCADefender để loại bỏ nhiễu đối kháng khỏi ảnh đầu vào. Phương pháp SCADefender gồm pha xây dựng và pha kiểm thử. Trong pha xây dựng, SCADefender xây dựng một mô hình mã hóa tự động phòng thủ có kiến trúc mô hình mã hóa tự động tích chập xếp chồng để loại bỏ nhiễu đối kháng trong ảnh đầu vào. Tập học để xây dựng mô hình mã hóa tự động phòng thủ này gồm tập học của mô hình kiểm thử và tập ảnh đối kháng sinh bởi nhiều phương pháp tấn công đối kháng không định hướng. Mục đích của tập ảnh đối kháng là mô hình mã hóa tự động phòng thủ có thể học được cách loại bỏ nhiễu đối kháng trong ảnh đối kháng. Trong pha kiểm thử, SCADefender đặt mô hình mã hóa tự động phòng thủ này trước mô hình kiểm thử. Khi có ảnh đầu vào, thay vì đưa vào ảnh đầu vào vào mô hình kiểm thử, mô hình mã hóa tự động phòng thủ sẽ loại bỏ nhiễu đối kháng khỏi ảnh đó nếu có, rồi mới chuyển đến mô hình kiểm thử.

Luận án đã thực nghiệm trên tập ảnh đối kháng sinh bởi nhiều phương pháp tấn công đối kháng không định hướng khác nhau như FGSM, Gaussian, CW L_2, BIM, PatternAttack và MI-FGSM. Thực nghiệm cho thấy SCADefender có thể tỉ lệ phát hiện ảnh đối kháng trung bình là 97.78% cho MNIST, 90.43% cho Fashion-MNIST và 80.64% cho CIFAR-10. Nếu không sử dụng mô hình mã hóa tự động phòng thủ, tỉ lệ phát hiện của mô hình kiểm thử đối với tập ảnh đối kháng này là 0%. Đối với ảnh không có nhiễu, SCADefender có tỉ lệ phát hiện 99.33% cho MNIST, 93.37% cho Fashion-MNIST và 87.14% cho CIFAR-10. Nếu không dùng mô hình mã hóa tự động phòng thủ, các ảnh không có nhiễu này có tỉ lệ phát hiện là 99.38% cho MNIST, 94.52% cho Fashion-MNIST và 91.19% cho CIFAR-10.

6.2. Các nghiên cứu liên quan

Nhiều nghiên cứu khác nhau đã đề xuất các phương pháp phòng thủ để cải thiện tính chắc chắn của mô hình tích chập. Phần này trình bày hai hướng phòng thủ phổ biến gồm hướng sử dụng mô hình mã hóa tự động phòng thủ và hướng học đối kháng.

Mô hình mã hóa tự động phòng thủ: Ý tưởng chính của hướng này là xây dựng một mô hình mã hóa tự động để loại bỏ nhiễu đối kháng trong ảnh đầu vào. Ảnh bị thêm nhiễu đối kháng sẽ đẩy vào mô hình kiểm thử để lấy kết quả dự đoán. Meng và cộng sự [46] đề xuất MagNet bao gồm nhiều bộ khôi phục và bộ nhận diện. Bộ nhận diện đánh giá tính tự nhiên của ảnh đầu vào. Nếu ảnh đầu vào có tính tự nhiên, ảnh này sẽ được đẩy qua bộ khôi phục có kiến trúc mô hình mã hóa tự động. Đầu ra của bộ khôi phục là một ảnh khác có tính tự nhiên tốt hơn và được kì vọng được mô hình kiểm thử dự đoán chính xác hơn ảnh đầu vào. Samangouei và cộng sự [107] đề xuất Defense-GAN gồm mô hình học sâu sinh và mô hình học sâu phân biệt. Nhiệm vụ của mô hình học sâu sinh là sinh ảnh thuộc phân phối dữ liệu chuẩn. Nhiệm vụ mô hình học sâu phân biệt là nhận diện ảnh sinh ra bởi mô hình học sâu sinh là ảnh dự đoán đúng hoặc ảnh đối kháng. Uiwon và cộng sự [49] đề xuất PuVAE sử dụng mô hình mã hóa tự động biến thiên có điều kiện để loại bỏ nhiễu đối kháng trong

ảnh đầu vào. Xiang và cộng sự [50] đề xuất Defense-VAE sử dụng mô hình mã hóa tự động biến thiên để chuyển ảnh đối kháng thành ảnh dự đoán đúng. Kim và cộng sự [108] kiểm tra tính chắc chắn từ nhiều góc nhìn khác nhau. Nhóm nghiên cứu phát hiện rằng các đặc trưng quan trọng có thể gây ra vấn đề tính chắc chắn. Liu và cộng sự [109] đề xuất DAFAR sử dụng mô hình phản hồi và bộ nhận diện. Mô hình phản hồi có dạng mô hình mã hóa tự động dùng để biến đổi ảnh đầu vào sang một phiên bản khác có chất lượng tốt hơn. Bộ nhận diện so sánh sự khác biệt giữa ảnh đầu vào và ảnh đầu ra của mô hình phản hồi để quyết định ảnh đầu vào có chứa nhiễu đối kháng không.

Bảng 6.1 so sánh các phương pháp xây dựng mô hình mã hóa tự động phòng thủ điển hình. Hàng *kiến trúc* mô tả loại kiến trúc mô hình mã hóa tự động. Hàng *dùng ảnh không có nhiễu* và hàng *dùng ảnh đối kháng* mô tả loại ảnh sử dụng để học mô hình mã hóa tự động phòng thủ. Hàng L_p *của ảnh đối kháng* là tiêu chí L_p sử dụng để sinh ảnh đối kháng, từ đó dùng để học mô hình mã hóa tự động phòng thủ. Hàng *bộ dữ liệu* mô tả tập học mô hình mã hóa tự động phòng thủ. Hàng *tấn công đối kháng* là các phương pháp tấn công đối kháng để sinh ảnh đối kháng trong thực nghiệm. Hàng *phương pháp so sánh* mô tả các phương pháp được so sánh trong thực nghiệm của các nghiên cứu.

Học đối kháng: Bai và cộng sự [111] phân loại các phương pháp theo hướng học đối kháng thành năm loại chính gồm chính quy hóa, học dựa theo lịch, học quần thể, học thích nghi và học bán/không giám sát. Hướng chính quy hóa là thêm thành phần chính quy vào hàm mục tiêu. Goodfellow và cộng sự [16] lần đầu đề xuất thêm thành phần chính quy dựa theo FGSM cùng với hàm cross-entropy. Hướng học dựa theo lịch lần đầu được đề xuất trong nghiên cứu của Cai và cộng sự [112]. Ý tưởng cơ bản là mô hình tích chập được xây dựng với tấn công đối kháng yếu. Khi mô hình tích chập đạt độ chuẩn xác cao với tấn công đối kháng ở ngưỡng nào đó, cường độ tấn công đối kháng được tăng dần lên. Hướng học quần thể là tạo thêm tập học mới chứa nhiều loại nhiễu khác nhau được lấy từ các mô hình khác. Tramèr và cộng sự [110] đề xuất sử dụng nhiều mô hình kiểm thử để sinh ảnh đối kháng thay vì chỉ một mô hình kiểm thử. Hướng học thích nghi điều chỉnh tham số trong quá trình xây dựng mô hình để tìm mô hình kiểm thử tối ưu hiệu quả hơn. Balaji và cộng sự [113] lần đầu đề xuất sử dụng tham số quả bóng lớn nhất có thể. Tham số này phải đảm

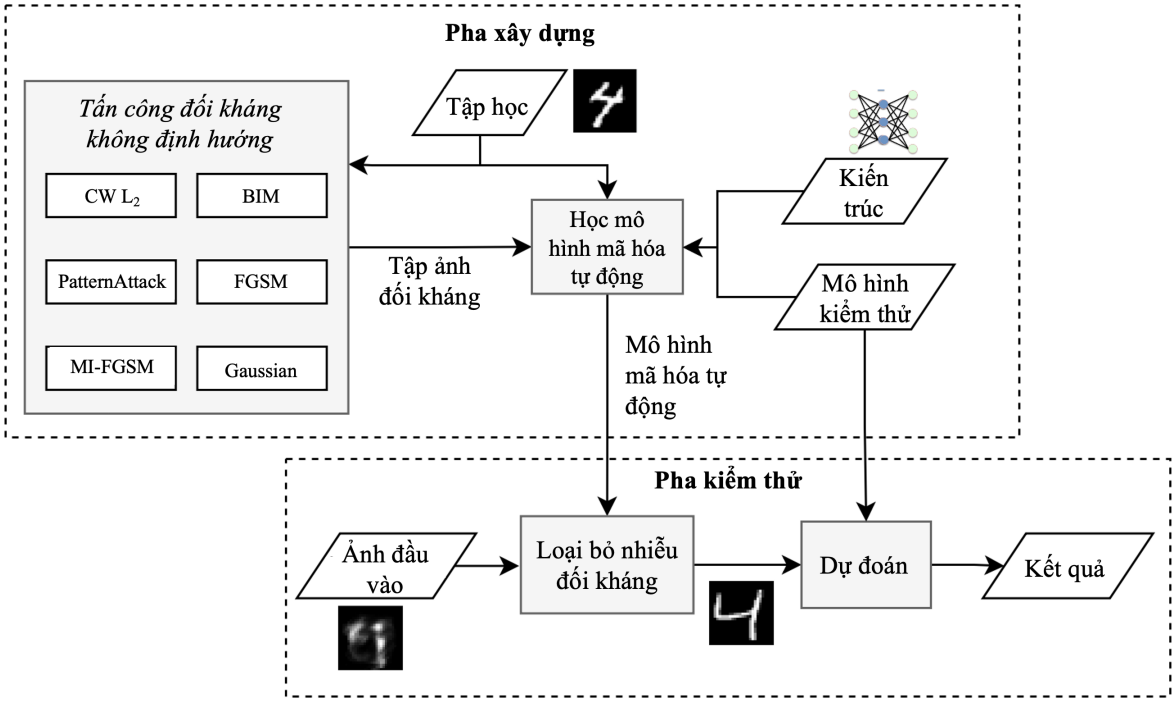
Bảng 6.1: So sánh các phương pháp mô hình mã hóa tự động phòng thủ

Danh mục	PuVAE [49]	Defense-VAE [50]	MagNet [46]	SCADefender
Kiến trúc	mô hình mã hóa tự động biến thiên có điều kiện [35]	mô hình mã hóa tự động biến thiên [63]	mô hình mã hóa tự động	mô hình mã hóa tự động tích chập xếp chồng
Dùng ảnh không có nhiễu	Có	Có	Có	Có
Dùng ảnh đối kháng	Không	Có	Không	Có
L_p của ảnh đối kháng	-	L_2, L_∞	-	L_0, L_2 và L_∞
Bộ dữ liệu	MNIST [41], CIFAR-10 [54], và Fashion-MNIST [72]	MNIST, Fashion-MNIST, CelebA, và CIFAR-10	MNIST và CIFAR-10	MNIST, CIFAR-10 và Fashion- MNIST
Tấn công đối kháng	FGSM [16], BIM [19], RAND-FGSM [110] và CW L_2 [17]	FGSM, RAND-FGSM và CW	FGSM, Deep-Fool [33] và CW	FGSM, MI-FGSM [34], BIM, CW L_2, Gaussian, BIS và HPBA với chế độ tối ưu
Phương pháp so sánh	Defense-GAN, MagNet và học đối kháng [16]	MagNet, học đối kháng, Defense-GAN và PuVAE	-	MagNet dùng bộ khôi phục, MagNet không dùng bộ khôi phục, PuVAE và học đối kháng

bảo rằng mọi ảnh ở trong không gian quả bóng có nhãn giống nhau. Cuối cùng, hướng tiếp cận học bán/không giám sát là học lại mô hình kiểm thử với dữ liệu bổ sung không gắn nhãn cùng với tập học ban đầu. Một trong những cảm hứng ban đầu xuất phát từ quan sát rằng độ chuẩn xác trong pha kiểm thử thường thấp hơn nhiều pha xây dựng trong nghiên cứu của Schmidt và cộng sự [114].

6.3. Phương pháp SCADefender

Để giảm thiểu vấn đề về khả năng loại bỏ nhiễu đối kháng trong ảnh đầu vào của các phương pháp mô hình mã hóa tự động phòng thủ hiện nay, luận án đề xuất phương pháp SCADefender. Hình 6.1 mô tả tổng quan SCADefender. Phương pháp gồm hai pha là pha xây dựng và pha kiểm thử.



Hình 6.1: Tổng quan phương pháp SCADefender.

Pha xây dựng thực hiện hai bước gồm chuẩn bị tập học của mô hình mã hóa tự động phòng thủ và học mô hình mã hóa tự động phòng thủ. Tập học của mô hình mã hóa tự động phòng thủ bao gồm tập học của mô hình kiểm thử và tập ảnh đối kháng. Tập ảnh đối kháng được sinh sử dụng tấn công đối kháng không định hướng để thêm nhiễu đối kháng vào tập học của mô hình kiểm thử và được trình bày chi tiết ở Phần 6.3.1. Kiến trúc mô hình mã hóa tự động phòng thủ là mô hình mã hóa tự động tích chập xếp chồng, được dùng để loại bỏ nhiễu đối kháng trong ảnh đầu vào. Lý do sử dụng mô hình mã hóa tự động tích chập xếp chồng là do khả năng học đặc trưng ảnh khá tốt và đã thảo luận ở Phần 2.2.3. Quá trình học mô hình mã hóa tự động phòng thủ được trình bày ở Phần 6.3.2.

Trong pha kiểm thử, mô hình mã hóa tự động phòng thủ được đặt trước mô hình kiểm thử. Mọi ảnh đầu vào đều phải đi qua mô hình mã hóa tự động phòng thủ để loại bỏ nhiễu đối kháng nếu có. Ảnh đầu ra của mô hình mã hóa tự động phòng thủ sẽ chuyển đến mô hình kiểm thử để lấy kết quả dự đoán.

6.3.1. Sinh tập ảnh đối kháng

Bước đầu tiên trong pha xây dựng là sinh tập ảnh đối kháng để làm đầu vào cho quá trình học mô hình mã hóa tự động phòng thủ. Để mô hình mã hóa tự động phòng thủ có khả năng nhận biết và loại bỏ được nhiều loại nhiễu đối kháng, một cách tiếp cận tự nhiên là tập ảnh đối kháng này được tạo từ nhiều phương pháp tấn công đối kháng không định hướng [50]. Phương pháp SCADefender sử dụng hai loại nhiễu gồm tự nhiên và nhân tạo. Đối với nhiễu tự nhiên, phương pháp SCADefender giả định nhiễu tự nhiên phải có phân phối Gaussian. Các phương pháp hiện nay không sinh ảnh đối kháng có loại nhiễu này. Vì thế, SCADefender đề xuất phương pháp thêm nhiễu Gaussian.

Đối với nhiễu nhân tạo, các phương pháp sinh nhiễu thuộc nhóm này chủ yếu dùng tiêu chí L_0 [17, 42], L_2 [18] và L_∞ [16, 19]. Để tăng khả năng nhận diện và loại bỏ được nhiều loại nhiễu nhân tạo khác nhau, SCADefender chọn các phương pháp đại diện cho từng nhóm nhiễu. Đối với L_2, SCADefender sử dụng PatternAttack và phương pháp CW L_2. Đối với L_∞, SCADefender sử dụng FGSM, BIM và MI-FGSM. Bởi vì PatternAttack không hỗ trợ tấn công đối kháng không định hướng, SCADefender sẽ đề xuất phiên bản tấn công đối kháng có định hướng của phương pháp này.

6.3.1.1. Phương pháp thêm nhiễu Gaussian không định hướng

Do các phương pháp tấn công đối kháng không định hướng không thêm nhiễu Gaussian, luận án đề xuất phương pháp này với giả định rằng kẻ tấn công có thể thêm nhiễu Gaussian vào ảnh dự đoán đúng. Cụ thể, với ảnh dự đoán đúng $\mathbf{x}$, ảnh đối kháng tương ứng được tạo bằng cách thêm nhiễu Gaussian lặp đi lặp lại như Công thức 6.1.

Tại mỗi lần lặp, thuật toán sẽ thêm một lượng nhiễu nhỏ $\epsilon_G \cdot p \sim \mathcal{N}(0, 1)$. Giá trị điểm ảnh sẽ được cắt gọn để thuộc khoảng [0, 1] bằng cách dùng hàm MIN và MAX. Nếu tại lần lặp thứ i, ảnh đối kháng sinh thành công, tức mô hình kiểm thử nhận diện sai nhãn ảnh đối kháng, thuật toán sẽ kết thúc. Ngược lại, thuật toán tiếp tục thêm nhiễu đến khi vượt số lần lặp tối đa. Phương pháp tấn

công này có thể coi là phương pháp đơn giản nhất. Nguyên nhân là do phương pháp này không cần tính đạo hàm của mô hình kiểm thử. Mặc dù phương pháp này có chi phí tính toán thấp hơn các phương pháp dùng đạo hàm, tỉ lệ thành công thường không tốt như các phương pháp tấn công khác. Nguyên nhân bởi vì nhiễu được thêm vào một lượng ngẫu nhiên.

$$\begin{aligned} \mathbf{x}_0 &= \mathbf{x} \\ \mathbf{x}_i &= \mathrm{MIN}(\mathbf{x}_{i-1} + \epsilon_G \cdot p \sim \mathcal{N}(0,1), 1) \\ \mathbf{x}_i &= \mathrm{MAX}(\mathbf{x}_i, 0) \end{aligned} \tag{6.1}$$

trong đó, $\epsilon_G > 0$ được sử dụng để điều chỉnh mức độ của nhiễu, $\mathcal{N}(0,1)$ là phân phối Gaussian có giá trị trung bình bằng 0 và phương sai bằng 1.

6.3.1.2. Phương pháp PatternAttack không định hướng

Chương 4 đề xuất PatternAttack có định hướng để sinh ảnh đối kháng có nhiễu đa dạng. Hàm mục tiêu mô tả ở Công thức 4.1. SCADefender chỉnh sửa PatternAttack có định hướng sang phiên bản không định hướng bằng cách thay đổi hàm mục tiêu như Công thức 6.2.

$$(1 - \alpha_H) \cdot L_2(\gamma(\mathcal{P}, \mathbf{x}), \mathbf{x}_{out})^2 - \alpha_H \cdot \mathrm{CE}(\mathbf{y}_{true}, \mathbf{M}(\mathbf{x}')) \tag{6.2}$$

trong đó, $\alpha_H \in (0,1)$.

Để tối thiểu hóa hàm mục tiêu trong Công thức 6.2, quá trình học tối thiểu hóa thành phần đầu tiên với trọng số $(1 - \alpha_H)$, đại diện cho khoảng cách giữa $\gamma(\mathcal{P}, \mathbf{x})$ và $\mathbf{x}_{out}$ nên càng nhỏ càng tốt. Tức là lượng nhiễu thêm vào các điểm ảnh thỏa mãn mẫu thêm nhiễu là nhỏ nhất. Đồng thời, thành phần thứ hai với trọng số α_H được tối đa hóa. Mục tiêu của thành phần này là khiến cho dự đoán của mô hình kiểm thử với ảnh chỉnh sửa, kí hiệu là $\mathbf{M}(\mathbf{x}')$, càng khác nhãn đúng $\mathbf{y}_{true}$ càng tốt.

6.3.2. Xây dựng mô hình mã hóa tự động

Từ tập ảnh đối kháng và bộ dữ liệu sạch, SCADefender xây dựng một mô hình mã hóa tự động kí hiệu $\mathbf{A}$. Tập học gồm các cặp có ảnh đầu vào và ảnh khôi phục mong đợi: $\{(\mathbf{x}, \mathbf{x})^+\} \cup \{(\mathbf{x}', \mathbf{x})^+\}$, trong đó $\mathbf{x}$ được dự đoán đúng bởi mô hình kiểm thử và $\mathbf{x}'$ là ảnh đối kháng tương ứng sinh bởi một phương pháp tấn công đối kháng không định hướng. Tập con $\{(\mathbf{x}, \mathbf{x})^+\}$ đảm bảo rằng nếu một ảnh không có nhiễu và được nhận diện chính xác bởi mô hình kiểm thử, mô hình mã hóa tự động có thể dịch chuyển ảnh đầu vào này sang phiên bản khác mà không thay đổi nhãn của nó. Tập con $\{(\mathbf{x}', \mathbf{x})^+\}$ đảm bảo rằng nếu ảnh đầu vào chứa nhiễu đối kháng, mô hình mã hóa tự động có thể loại bỏ những nhiễu này.

Đối với cặp $(\mathbf{x}^i, \mathbf{x}^i_{out})$ trong tập học, hàm mục tiêu của $\mathbf{A}$ gồm hai thành phần như Công thức 6.3.

$$\alpha_S \cdot L_2(\mathbf{x}^i_{out}, \mathbf{A}(\mathbf{x}^i))^2 + (1 - \alpha_S) \cdot \mathrm{CE}(\mathbf{M}(\mathbf{A}(\mathbf{x}^i)), \mathbf{y}_{true}) \tag{6.3}$$

trong đó, $\alpha_S \in (0, 1)$ là trọng số, $\mathbf{A}(\mathbf{x}^i)$ là lời gọi đến mô hình mã hóa tự động để lấy ảnh đầu ra và $\mathbf{y}_{true}$ là vector xác suất mong đợi của $\mathbf{x}^i_{out}$. Thành phần đầu tiên tối thiểu hóa khoảng cách L_2 giữa ảnh khôi phục mong đợi $\mathbf{x}^i_{out}$ và ảnh khôi phục thực tế $\mathbf{A}(\mathbf{x}^i)$. Thành phần thứ hai đảm bảo rằng nhãn của ảnh khôi phục thực tế giống nhãn của ảnh khôi phục mong đợi. Nếu α_S càng gần 0, mô hình sẽ có xu hướng tối thiểu hóa thành phần thứ hai. Ngược lại, nếu α_S càng gần một, mô hình sẽ có xu hướng tối thiểu hóa thành phần thứ nhất.

6.4. Thực nghiệm

Để chứng minh hiệu quả của SCADefender, thực nghiệm so sánh SCADefender với các phương pháp mô hình mã hóa tự động phòng thủ khác gồm MagNet và PuVAE. Ngoài ra, thực nghiệm còn so sánh với phương pháp học đối kháng [16]. Thực nghiệm được tiến hành trên ba bộ dữ liệu phổ biến gồm MNIST [41], Fashion-MNIST [72] và CIFAR-10 [54]. Thực nghiệm trả lời các câu hỏi sau đây:

- **RQ1 - Tỉ lệ phát hiện của ảnh không có nhiễu**: Phương pháp SCADefender có thể nhận diện được nhiều ảnh không có nhiễu chính xác hơn các phương pháp so sánh không?
- **RQ2 - Tỉ lệ phát hiện của ảnh đối kháng**: Phương pháp SCADefender có thể nhận diện được nhiều ảnh đối kháng chính xác hơn các phương pháp so sánh không?
- **RQ3 - Hiệu năng**: Phương pháp SCADefender có thể nhận diện ảnh đầu vào với chi phí thấp không, trong đó ảnh đầu vào có thể là ảnh có nhiễu hoặc không?

6.4.1. Cấu hình

6.4.1.1. Mô hình kiểm thử

Thay vì sử dụng cùng mô hình tích chập trong thực nghiệm Chương 4 và Chương 5, thực nghiệm chương này xây dựng ba mô hình mới tốt hơn. Với từng bộ dữ liệu, thực nghiệm xây dựng một mô hình kiểm thử với độ chuẩn xác cao hơn trên tập học và tập kiểm thử. Tên của các mô hình kiểm thử này là M với MNIST, F cho Fashion-MNIST và C cho CIFAR-10.

Bảng 6.2: Thống kê độ chuẩn xác của mô hình kiểm thử

Mô hình	Tập học	Tập kiểm thử
M	99.99%	99.38%
F	99.56%	94.52%
C	99.08%	91.19%

Bảng 6.3 trình bày kiến trúc của mô hình kiểm thử và độ chuẩn xác của các mô hình này được trình bày trong Bảng 6.2. Các mô hình này được cài đặt bằng bộ thư viện Keras, sử dụng bộ tối ưu Adam với cấu hình mặc định của Keras. Thực nghiệm lưu mô hình với độ chuẩn xác cao nhất trên tập xác thực. Mô hình M hội tụ sau khoảng 50 lần lặp. Mô hình F và C hội tụ sau khoảng 200 lần lặp.

Bảng 6.3: Kiến trúc của mô hình kiểm thử

Mô hình M	Mô hình C và F
Conv(filters=96, kernel=3)+ ReLU	Conv2D(filters=96, kernel=3, 'same') + BatchNorm + ReLU
Conv(filters=192, kernel=3,	Conv2D(filters=96, kernel=3, 'same')+ BatchNorm + ReLU
strides=2)+ ReLU	Conv2D(filters=96, kernel=3, 'same')+ BatchNorm + ReLU
Dropout(0.4)	MaxPooling2D
Conv(filters=192, kernel=3,	Conv2D(filters=192, kernel=3, 'same')+ BatchNorm + ReLU
strides=1)+ ReLU	Conv2D(filters=192, kernel=3, 'same')+ BatchNorm + ReLU
Conv(filters=192, kernel=3,	Conv2D(filters=192, kernel=3, 'same')+ BatchNorm + ReLU
strides=2)+ ReLU	MaxPooling2D
Dropout(0.4)	Conv2D(filters=192, kernel=3, 'same')+ BatchNorm + ReLU
Flatten	Conv2D(filters=192, kernel=1, 'same')+ BatchNorm + ReLU
Dense(neurons=256)+ ReLU	Conv2D(filters=10, kernel=1, 'same')+ BatchNorm + ReLU
Dense(neurons = 10)	GlobalAvgPool2D

6.4.1.2. Sinh tập kiểm thử

Phần này chuẩn bị dữ liệu để trả lời câu hỏi RQ2. Thực nghiệm cần so sánh hiệu quả của phương pháp SCADefender với nhiều loại nhiễu đối kháng khác nhau. Do đó, thực nghiệm tấn công mô hình kiểm thử sử dụng các phương pháp tấn công đối kháng không định hướng gồm FGSM, BIM, MI-FGSM, CW L_2, PatternAttack và Gaussian. Trong quá trình tấn công, mục tiêu chính là sinh ảnh đối kháng không chứa quá nhiều nhiễu để tránh phá hủy cảm quan của ảnh đầu vào. Việc đánh giá tính cảm quan được thực hiện bởi một nhóm có ba thành viên. Với một cấu hình sinh ảnh đối kháng, nếu ít nhất hai thành viên đồng thuận rằng tập ảnh đối kháng quá nhiều nhiễu, cấu hình sẽ được điều chỉnh lại. Với mục tiêu này, cấu hình của các phương pháp tấn công đối kháng không định hướng được trình bày ở Bảng 6.4. Kí hiệu của MI-FGSM và phương pháp CW L_2 được đề cập trong bài báo gốc. *iters* là số lần lặp. Đối với PatternAttack, *epoch* là số lần lặp để xây dựng mô hình mã hóa tự động.

Từ 5,000 ảnh đầu tiên trên tập kiểm thử của từng bộ dữ liệu, thực nghiệm lấy những ảnh dự đoán đúng. Đối với M, có 4,954 ảnh dự đoán đúng. Với F, 4,710 ảnh dự đoán đúng. Đối với C có 4,542 ảnh dự đoán đúng. Bảng 6.5 tóm tắt tỉ lệ thành công của các mô hình kiểm thử khi thêm nhiễu đối kháng vào những ảnh dự đoán đúng này. Ví dụ, với mô hình kiểm thử M, sử dụng FGSM sinh 3,035 ảnh đối kháng từ 4,954 ảnh dự đoán đúng với tỉ lệ thành công bằng

Bảng 6.4: Cấu hình của các phương pháp tấn công đối kháng không định hướng

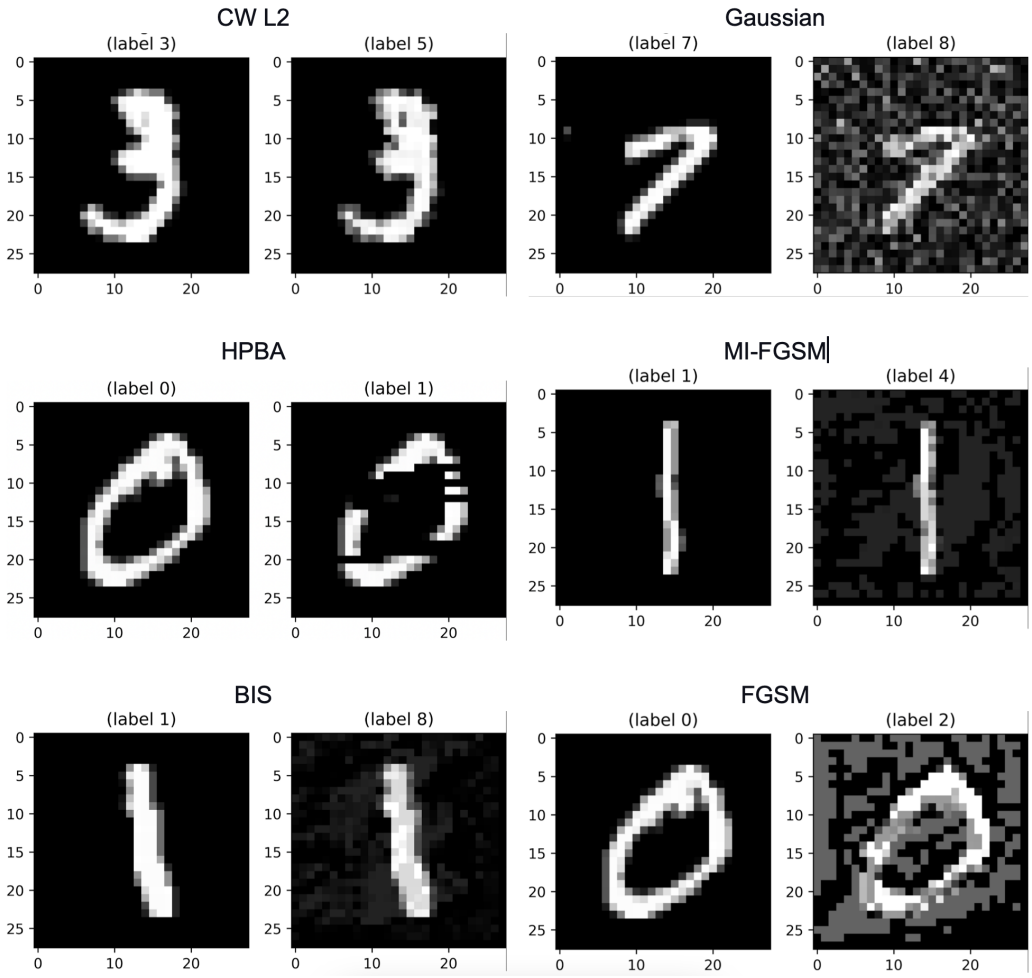
Mô hình	FGSM	MI-FGSM	BIM	CW L_2	Gaussian	PatternAttack	BIM_∞
M	ϵ_F= 0.39	$\epsilon = 0.0039$ T = 40 $\mu = 2.0$	$\epsilon_B = 0.0039$ iters = 50 $\alpha_B = 0.2$	K = 0 iters = 200	$\epsilon_G = 0.039$, iters = 50	lr = 0.001 $\epsilon_H = 0.05$ epoch = 100	$\epsilon_I = 0.005$, iters = 40 $\alpha_B = \infty$
F	$\epsilon_F = 0.039$	$\epsilon = 0.0039$ T = 50 $\mu = 1.0$	$\epsilon_B = 0.039$ iters = 100 $\alpha_B = 0.39$	K = 0 iters = 1000	$\epsilon_G = 0.039$, iters = 20	lr = 0.00001 $\epsilon_H = 0.001$ epoch = 50	$\epsilon_I = 0.005$ iters = 20 $\alpha_B = \infty$
C	$\epsilon_F = 0.0039$	$\epsilon = 0.0039$ T = 20 $\mu = 1.0$	$\epsilon_B = 0.0039$ iters = 20 $\alpha_B = 0.008$	K = 0 iters = 1000	$\epsilon_G = 0.0039$, iters = 20	lr = 0.00001 $\epsilon_H = 0.001$ epoch = 50	$\epsilon_I = 0.0039$ iters = 20 $\alpha_B = \infty$

61.26%. Như có thể thấy, một vài phương pháp tấn công đạt tỉ lệ thành công khá thấp như phương pháp MI-FGSM với mô hình kiểm thử M, phương pháp Gaussian với M và C. Nói chung, tỉ lệ thành công của các tấn công này có thể cải thiện bằng cách tăng cường độ nhiễu. Tuy nhiên, nếu thêm quá nhiều nhiễu đối kháng, tính cảm quan của ảnh có thể bị phá hỏng quá mức.

Bảng 6.5: Thống kê tỉ lệ thành công (SR) của các phương pháp tấn công đối kháng không định hướng, trong đó #adv là số ảnh đối kháng

Mô hình	Danh mục	FGSM	MI-FGSM	BIM	CW L_2	Gaussian	PatternAttack	BIM_∞
M	#adv	3,035	612	4,668	3,925	2,632	579	4,383
	SR	61.26%	12.35%	94.23%	79.23%	53.13%	11.69%	88.47%
F	#adv	4,150	3,558	4,695	4,710	4,710	3,855	4,256
	SR	88.11%	75.54%	99.68%	100%	100%	81.85%	90.36%
C	#adv	2,073	2,985	4,542	4,310	4,541	297	4,078
	SR	45.64%	65.72%	100%	94.89%	99.98%	6.54%	89.78%

Mỗi tấn công đối kháng không định hướng có thể sinh ảnh đối kháng có loại nhiễu đối kháng có đặc trưng khác nhau. Trong khi một vài phương pháp tấn công đối kháng không định hướng sẽ thêm nhiễu đối kháng vào một tập điểm ảnh như CW L_2 và PatternAttack với cài đặt tối ưu hóa, các phương pháp khác có thể thêm nhiễu đối kháng vào toàn bộ ảnh như FGSM và BIM. Ví dụ, xét các ảnh đối kháng trên MNIST được thể hiện ở Hình 6.2. Trong các ảnh đối kháng này, phương pháp CW L_2 có xu hướng sinh ảnh có kết quả nhìn tốt nhất vì sự khác biệt giữa ảnh dự đoán đúng và ảnh đối kháng bằng mắt thường khá nhỏ. Ngược lại, Gaussian và FGSM có kết quả không tốt do toàn bộ ảnh được thêm nhiễu đối kháng với lượng nhiễu dễ nhận ra bằng mắt thường.



Hình 6.2: Ví dụ ảnh đối kháng từ bộ dữ liệu MNIST sinh bởi một vài phương pháp tấn công đối kháng không định hướng.

6.4.1.3. Cấu hình của các phương pháp cải thiện tính chắc chắn

a. Phương pháp SCADefender

Ý tưởng chính của SCADefender gồm hai bước. Bước đầu tiên là tấn công mô hình kiểm thử bằng cách sử dụng các phương pháp tấn công đối kháng không định hướng khác nhau. Thực nghiệm sử dụng cấu hình ở Bảng 6.4 và tiến hành với 10,000 ảnh đầu tiên trên tập học. Mã nguồn các phương pháp tấn công này được công khai[1].

Bước thứ hai là học mô hình mã hóa tự động từ ảnh đối kháng. Thực nghiệm

[1]github.com/testingforAI-vnuuet/AdvAttackCollection

đã thử với nhiều kiến trúc (cùng với các giá trị siêu tham số) khác nhau và chọn kiến trúc tốt nhất trong các lần thử nghiệm. Kiến trúc mô hình mã hóa tự động được mô tả ở Bảng 5.1. Mục tiêu của mô hình mã hóa tự động là loại bỏ nhiễu đối kháng từ ảnh đầu vào nếu có. Tập học của mô hình mã hóa tự động bao gồm tập học của mô hình kiểm thử và ảnh đối kháng sinh từ bước một. Tập học này được chia thành hai tập con gồm tập học của mô hình mã hóa tự động và tập xác thực của mô hình mã hóa tự động với tỉ lệ 9:1. Các mô hình mã hóa tự động được xây dựng với bộ tối ưu Adam trong khoảng 500 lần lặp và thực nghiệm lưu mô hình có giá trị mất mát nhỏ nhất trên tập xác thực.

b. Phương pháp MagNet

Như đã đề cập trước đó, MagNet gồm bộ nhận diện và bộ khôi phục. Với mô hình kiểm thử M, một bộ nhận diện được học trên MNIST với loại lỗi khôi phục là L_2. MNIST được chia thành hai tập gồm tập 55,000 ảnh cho xây dựng bộ nhận diện và tập 5,000 ảnh thuộc về tập xác thực. Tập kiểm thử của bộ nhận diện là tập kiểm thử của MNIST. Thực nghiệm chọn ngưỡng của lỗi khôi phục sao cho tỉ lệ dương tính giả của bộ nhận diện trên tập xác thực nhiều nhất là 0.001. Điều đó có nghĩa là bộ khôi phục chỉ từ chối sai tối đa 0.1% ảnh trên tập xác thực.

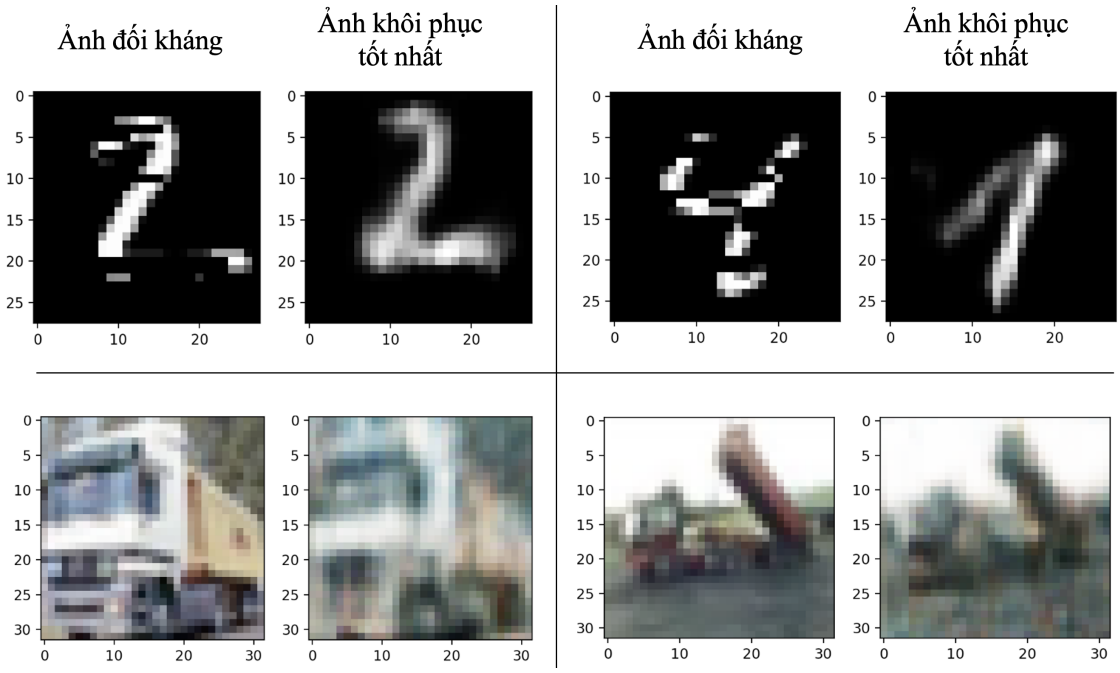
Với mô hình kiểm thử F, một bộ nhận diện được xây dựng trên Fashion-MNIST với loại lỗi khôi phục là L_1. Các cấu hình khác giống hệt với quá trình xây dựng bộ nhận diện cho mô hình kiểm thử M.

Với mô hình kiểm thử C, một bộ nhận diện được xây dựng trên CIFAR-10 với lỗi dựa theo xác xuất hội tụ. CIFAR-10 được chia thành hai tập gồm tập 45,000 ảnh cho xây dựng bộ nhận diện và tập gồm 5,000 ảnh cho tập xác thực. Tập kiểm thử của bộ nhận diện là tập kiểm thử của CIFAR-10. Thực nghiệm sử dụng lỗi dựa theo xác suất hội tụ với nhiệt độ T bằng 40. Ngoài ra, thực nghiệm cài đặt ngưỡng của tỉ lệ dương tính giả là 0.02 trên tập xác thực để tìm ngưỡng của lỗi dựa theo xác suất hội tụ.

c. Phương pháp PuVAE

Thực nghiệm sử dụng tốc độ học bằng 0.001 và số lần lặp bằng 1,000 để đảm bảo hàm mục tiêu hội tụ. Thực nghiệm không sử dụng trọng số mà PuVAE đề xuất (β= 0.01, $\alpha = 0.1$, $\gamma = 10$ trong Công thức 2.20) vì ảnh khôi phục trông

khá tệ dưới góc nhìn con người. Thay vào đó, trọng số của mọi thành phần đều bằng nhau. Đối với kích thước của không gian ẩn, thực nghiệm sử dụng hai kích thước gồm 4 (gọi là *config 1*) và 16 (gọi là *config 2*). Hình 6.3 trình bày bốn cặp ảnh sinh bởi PuVAE trên MNIST và CIFAR-10. Với từng cặp, ảnh bên trái là ảnh đối kháng sinh bởi PatternAttack. Bằng cách sử dụng PuVAE, ảnh được khôi phục lại hiển thị ở bên phải.



Hình 6.3: Ví dụ ảnh sinh bởi PuVAE trên MNIST và CIFAR-10.

d. Phương pháp học đối kháng

Thực nghiệm sử dụng cấu hình mặc định $\alpha = 0.5$ như đề cập trong bài báo gốc [16]. Ngoài ra, thực nghiệm học lại mô hình kiểm thử bằng cách sử dụng cấu hình học y hệt lúc xây dựng mô hình kiểm thử ban đầu.

6.4.2. Kết quả

6.4.2.1. RQ1 - Tỉ lệ phát hiện ảnh không có nhiễu

Thực nghiệm này đánh giá hiệu quả của SCADefender khi xử lý ảnh không có nhiễu. Tất cả mọi ảnh đều thuộc tập kiểm thử của MNIST, CIFAR-10 và Fashion-MNIST. Kết quả thực nghiệm được trình bày trong Bảng 6.6. Các giá

trị tốt hơn được in đậm ngoại trừ kết quả khi không sử dụng phương pháp cải thiện tính chắc chắn.

Có thể thấy tỉ lệ phát hiện của SCADefender tốt hơn các phương pháp đang so sánh. Đối với mô hình M và F, tỉ lệ phát hiện của SCADefender rất gần với kết quả khi không sử dụng bộ loại nhỏ nhiễu đối kháng. Với mô hình M, SCADefender đạt 99.33%, thấp hơn 0.05% so với kết quả không sử dụng phương pháp cải thiện tính chắc chắn. Với F, SCADefender đạt 93.37% (thấp hơn 1.15%). Với mô hình C, SCADefender đạt 87.14% (thấp hơn 4.05%). Các phương pháp đang so sánh đều có kết quả nhỏ hơn SCADefender.

Bảng 6.6: Thống kê về tỉ lệ phát hiện của các phương pháp trên ảnh không có nhiễu

Phương pháp	Cấu hình	M	F	C
MagNet (bộ khôi phục)	0.01	99.19%	90.97%	79.09%
	0.025	98.67%	90.19%	80.04%
	0.05	99.07%	90.09%	80.32%
MagNet (bộ khôi phục & bộ nhận diện)	0.01	99.19%	90.99%	80.58%
	0.025	98.67%	90.21%	81.41%
	0.05	99.08%	90.09%	81.53%
PuVAE	config 1	95.17%	65.24%	24.97%
	config 2	96.97%	66.92%	30.1%
Học đối kháng	0.02	98.67%	93.32%	80.51%
	0.1	98.52%	91.25%	78.32%
	0.2	98.64%	90.92%	66.36%
SCADefender	α_S=0.5	**99.33%**	**93.37%**	**87.14%**
Không sử dụng		99.38%	94.52%	91.19%

6.4.2.2. RQ2 - Tỉ lệ phát hiện ảnh đối kháng

Các ảnh đối kháng được sinh từ tập kiểm thử. Bảng 6.7, Bảng 6.8 và Bảng 6.9 trình bày kết quả của SCADefender và các phương pháp so sánh. Tổng quan cho thấy SCADefender đạt kết quả tốt nhất trong 17/21 tấn công và kết quả xếp hạng nhì trong 3/21 tấn công.

Đối với mô hình M, SCADefender đạt tỉ lệ phát hiện cao nhất trên tập ảnh

đối kháng sinh bởi FGSM (95.29%), MI-FGSM (99.35%), BIM_∞ (99.68%), BIM (99.72%), CW L_2 (98.25%), Gaussian (96.72%) và PatternAttack (95.48%). Đặc biệt là tỉ lệ phát hiện trong PatternAttack và FGSM vượt trội so với các phương pháp còn lại. Ví dụ, trong khi SCADefender đạt tỉ lệ phát hiện khoảng 95.29% trên FGSM, tỉ lệ phát hiện cao nhất bởi MagNet, PuVAE và học đối kháng khoảng 54.27%, 44.71% và 65.54%.

Bảng 6.7: Thống kê tỉ lệ phát hiện của các phương pháp cải thiện tính chắc chắn cho mô hình kiểm thử M

Phương pháp	Cấu hình	FGSM	MI-FGSM	BIM_∞	BIM	CW L_2	Gaussian	PatternAttack
MagNet (bộ khôi phục)	0.01	37.76%	94.28%	99.01%	98.85%	96.5%	93.26%	35.8%
	0.025	47.87%	90.69%	97.96%	97.68%	95.71%	83.42%	50.31%
	0.05	54.27%	91.01%	97.94%	98.01%	95.21%	87.05%	35.82%
MagNet (bộ nhận diện & bộ khôi phục)	0.01	0%	17.97%	6.71%	20.51%	93.5%	0.35%	26.76%
	0.025	0%	74.67%	97.07%	97.35%	95.63%	2.42%	49.99%
	0.05	0%	24.02%	17.67%	38.68%	94.19%	0.35%	22.48%
PuVAE	config 1	44.71%	75.65%	88.73%	90.14%	87.69%	45.08%	56.01%
	config 2	33.31%	70.75%	85.33%	88.74%	90.96%	41.28%	56.29%
Học đối kháng	0.0078	64.35%	85.15%	97.39%	97.04%	94.45%	85.15%	63.13%
	0.039	56.47%	88.56%	97.41%	96.99%	94.45%	85.66%	65.41%
	0.2	65.54%	90.2%	97.92%	97.61%	95.06%	90.5%	66.46%
SCADefender	α_S=0.5	**95.29%**	**99.35%**	**99.68%**	**99.72%**	**98.25%**	**96.72%**	**95.48%**

Bảng 6.8: Thống kê tỉ lệ phát hiện của các phương pháp cải thiện tính chắc chắn cho mô hình kiểm thử F

Phương pháp	Cấu hình	FGSM	MI-FGSM	BIM_∞	BIM	CW L_2	Gaussian	PatternAttack
MagNet (bộ khôi phục)	0.01	4%	81%	91.46%	65.94%	93.06%	56.16%	61.35%
	0.025	8.29%	91.23%	90.59%	77.35%	91.85%	90.79%	61.84%
	0.05	11.78%	90.05%	90.44%	75.94%	91.91%	88.17%	68.59%
MagNet (bộ nhận diện + bộ khôi phục)	0.01	0%	80.27%	91.31%	65.83%	92.95%	55.56%	61.21%
	0.025	0%	89.8%	90.46%	77.16%	91.7%	89.57%	61.63%
	0.05	0%	89.49%	90.23%	75.67%	91.72%	87.47%	68.37%
PuVAE	config 1	61.47%	67.82%	67.54%	67.58%	67.81%	68.09%	64.9%
	config 2	62.89%	69.45%	69.46%	69.02%	69.53%	70.01%	66.8%
Học đối kháng	0.0078	4.87%	94.04%	**96.51%**	92.78%	96.71%	93.72%	76.9%
	0.039	2.6%	93.34%	94.36%	**93.4%**	94.56%	93.39%	74.3%
	0.196	33.42%	93.11%	93.48%	91.85%	93.65%	88.98%	63.39%
SCADefender	α_S=0.5	**83.57%**	**95.19%**	95.21%	86.92%	**96.77%**	**94.58%**	**80.78%**

Đối với mô hình kiểm thử F, SCADefender đạt tỉ lệ phát hiện cao nhất trên tập ảnh đối kháng sinh bởi FGSM (83.57%), MI-FGSM (95.19%), CW L_2 (96.77%), Gaussian (94.58%) và PatternAttack (80.78%). Đặc biệt là tỉ lệ phát hiện trên tập ảnh đối kháng sinh bởi FGSM vượt trội so với các phương pháp

còn lại. Trong khi SCADefender đạt tỉ lệ phát hiện 83.57%, tỉ lệ phát hiện cao nhất của MagNet, PuVAE và học đối kháng khoảng 11.78%, 62.89% và 33.42%. Đối với BIM_∞ và BIM, tỉ lệ phát hiện của SCADefender xếp thứ hai.

Bảng 6.9: Thống kê tỉ lệ phát hiện của các phương pháp cải thiện tính chắc chắn cho mô hình kiểm thử C

Phương pháp	Cấu hình	FGSM	MI-FGSM	BIM_∞	BIM	CW L_2	Gaussian	PatternAttack
MagNet (bộ khôi phục)	0.01	63.05%	80.17%	79.7%	78.79%	82.51%	50.17%	56.57%
	0.025	65.12%	81.37%	80.98%	80%	83.55%	48.15%	54.76%
	0.05	66.52%	82.18%	81.59%	80.7%	84.14%	51.52%	58.04%
MagNet (bộ nhận diện + bộ khôi phục)	0.01	56.49%	68.04%	59.95%	64.32%	79.26%	50.17%	56.11%
	0.025	58.27%	68.71%	59.84%	64.41%	79.85%	48.15%	54.44%
	0.05	60.2%	69.95%	62.15%	66.87%	81.61%	51.52%	57.82%
PuVAE	config 1	23.01%	26.26%	26.82%	26.77%	26.71%	23.91%	24.57%
	config 2	26.53%	30.45%	31.77%	30.58%	30.9%	20.54%	29.25%
Học đối kháng	0.02	72.74%	83.18%	84.68%	83.9%	84.81%	56.9%	**75.31%**
	0.1	69.51%	63.3%	81.53%	80.53%	82.43%	**83.82%**	55.49%
	0.20	57.21%	73.84%	69.24%	67.94%	69.59%	53.54%	52.04%
SCADefender	α_S=0.5	**75.54%**	**88.88%**	**87.76%**	**87.17%**	**91.63%**	62.63%	70.87%

Đối với mô hình kiểm thử C, SCADefender đạt tỉ lệ phát hiện cao nhất trên tập ảnh đối kháng sinh bởi FGSM (75.54%), MI-FGSM (88.88%), BIM_∞ (87.76%), BIM (87.17%) và CW L_2 (91.63%). Đối với tập ảnh đối kháng sinh bởi Gaussian và PatternAttack, SCADefender đạt vị trí thứ hai với tỉ lệ phát hiện là 62.63% và 70.87%.

6.4.2.3. RQ3 - Hiệu năng

Thực nghiệm này đánh giá hiệu năng của SCADefender khi sử dụng trong thực tế. Đối với học đối kháng, thực nghiệm không tính thời gian học lại mô hình kiểm thử với bộ dữ liệu bổ sung. Kết quả được trình bày trong Bảng 6.10.

Đơn vị là mili giây. *#queries* là số lời gọi đến mô hình kiểm thử hoặc mô hình mã hóa tự động để lấy kết quả dự đoán. Như có thể thấy, số truy vấn tỉ lệ thuận với chi phí tính toán. Đối với MagNet, PuVAE và SCADefender, mỗi ảnh đầu vào được đẩy qua hai mô hình gồm mô hình mã hóa tự động và mô hình kiểm thử. PuVAE cần mười truy vấn với thời gian khoảng 2.5 - 2.9 mili giây. Chi phí của MagNet và SCADefender thấp hơn PuVAE đáng kể, khoảng 0.4 - 0.7 mili giây. Phương pháp học đối kháng không sử dụng mô hình mã hóa tự

Bảng 6.10: Hiệu năng của cải thiện tính chắc chắn trên một ảnh (mili giây)

Phương pháp	CIFAR-10	MNIST	Fashion MNIST	#queries
MagNet (chỉ bộ khôi phục)	0.4	0.5	0.5	2
MagNet (bộ khôi phục + bộ nhận diện)	0.6	0.7	0.7	3
PuVAE	2.8	2.5	2.9	10
Học đối kháng	0.3	0.2	0.2	1
SCADefender	0.4	0.5	0.5	2
Không phòng thủ	0.3	0.2	0.2	1

động. Thay vào đó, phương pháp này học lại mô hình với bộ dữ liệu bổ sung là các ảnh đối kháng. Do đó, chi phí của phương pháp này xấp xỉ với trường hợp không sử dụng phương pháp phòng thủ, khoảng 0.2 - 0.3 mili giây.

6.5. Tổng kết

Mô hình tích chập đã rất thành công trong nhiều nhiệm vụ nhận dạng hình ảnh. Tuy nhiên, nhiều nghiên cứu gần đây cho thấy mô hình tích chập dễ bị tấn công bởi các tấn công đối kháng không định hướng. Điều này đặt ra một mối đe dọa nghiêm trọng đối với các ứng dụng trong các hệ thống dựa trên học máy nói chung và mô hình tích chập nói riêng. Xây dựng cải thiện tính chắc chắn để bảo vệ mô hình tích chập khỏi các cuộc tấn công dạng này là một cách tiếp cận phổ biến. Tuy nhiên, luận án nhận thấy rằng một số phương pháp tiên tiến nhất như MagNet, PuVAE và DefenseVAE có thể không xử lý tốt ảnh đối kháng. Do đó, luận án đề xuất một cải thiện tính chắc chắn đơn giản nhưng hiệu quả có tên là SCADefender sử dụng mô hình mã hóa tự động tích chập xếp chồng để loại bỏ nhiễu đối kháng khỏi ảnh đầu vào. Các thực nghiệm trên MNIST, Fashion-MNIST và CIFAR-10 chứng minh rằng tỉ lệ phát hiện của SCADefender tốt hơn so với các cải thiện tính chắc chắn tương tự bao gồm MagNet và PuVAE. Kết quả nghiên cứu đã được chấp thuận đăng tại tạp chí International Journal of Pattern Recognition and Artificial Intelligence (Q3).

Chương 7

Kết luận

7.1. Các kết quả đạt được

Luận án đề xuất các phương pháp để cải thiện tính chắc chắn của mô hình học sâu. Luận án đã giải quyết được bốn vấn đề. Vấn đề đầu tiên là chất lượng tấn công của DeepCheck [42] đối với mô hình nơ-ron truyền thẳng chưa đủ tốt. Vấn đề thứ hai là ảnh đối kháng sinh bởi ATN [18] cho mô hình tích chập chưa đủ đa dạng. Vấn đề thứ ba là ảnh đối kháng sinh bởi nhiều phương pháp tấn công đối kháng chứa nhiều nhiễu dư thừa. Vấn đề thứ bốn là khả năng phòng thủ tấn công đối kháng theo hướng mô hình mã hóa tự động của các phương pháp chưa đủ tốt với ảnh đối kháng có nhiễu đa dạng.

Đối với vấn đề chất lượng tấn công của DeepCheck, luận án đề xuất phương pháp HA4FNN. Phương pháp HA4FNN sử dụng bộ giải phỏng đoán thay vì bộ giải SMT và loại bỏ việc duy trì trạng thái kích hoạt nơ-ron. Từ mô hình kiểm thử, HA4FNN chuyển mô hình này sang mã nguồn C, sau đó biên dịch và thực thi mã nguồn này với đầu vào là ảnh dự đoán đúng để lấy đường thi hành. Sau đó, thực thi tượng trưng chuyển đường thi hành thành hệ ràng buộc và dùng bộ giải phỏng đoán để tìm nghiệm. Nghiệm này tương ứng với ảnh đối kháng và có thể có trạng thái kích hoạt nơ-ron khác với ảnh dự đoán đúng. Thực nghiệm trên MNIST, Fashion-MNIST và bộ chữ cái viết tay cho thấy phương pháp HA4FNN có hiệu năng và tỉ lệ thành công vượt trội so với DeepCheck. Một công cụ đã được cài đặt để chứng minh hiệu quả của HA4FNN. Nghiên cứu này đã được

đăng ở tạp chí Automated Software Engineering. Tuy nhiên, HA4FNN mới chỉ dừng lại ở mô hình nơ-ron truyền thẳng.

Từ đề xuất đầu tiên, luận án đã mở rộng sang kiểm thử mô hình tích chập, một loại mô hình học sâu phổ biến để phân loại ảnh. Đa số các phương pháp tấn công đối kháng cho mô hình tích chập không có tính khái quát hóa. Mặc dù phương pháp ATN [18] được đề xuất có tính khái quát hóa, ATN không sinh ảnh đối kháng có nhiễu đối kháng đa dạng. Vì thế, luận án đã cải thiện ATN, gọi là PatternAttack, để sinh ảnh đối kháng có nhiễu đối kháng đa dạng cho mô hình học sâu bằng cách sử dụng mẫu thêm nhiễu. Ngoài ra, luận án đề xuất thuật toán tham lam để cải thiện chất lượng ảnh đối kháng theo độ đo L_0 và L_2. Thực nghiệm trên MNIST và CIFAR-10 cho thấy PatternAttack có thể tấn công mô hình học sâu với tỉ lệ thành công cao và cải thiện chất lượng ảnh đối kháng với tỉ lệ giảm nhiễu tốt. Một phần kết quả nghiên cứu này đã đạt giải bài báo xuất sắc nhất tại hội nghị RIVF. Phiên bản cải tiến tiếp đó đã đăng ở tạp chí SoftComputing. Một công cụ đã được cài đặt để chứng minh hiệu quả của PatternAttack và đã chuyển giao cho tập đoàn TSDV Việt Nam.

Đối với vấn đề chất lượng ảnh đối kháng, luận án đề xuất QI4AE để nâng cao chất lượng ảnh đối kháng sinh bởi mọi tấn công đối kháng. Nghiên cứu này xuất phát từ việc quan sát rằng đa số các phương pháp tấn công đối kháng sinh ảnh đối kháng có chứa nhiều nhiễu dư thừa. Việc loại bỏ những nhiễu dư thừa này sẽ làm tăng chất lượng ảnh đối kháng. Phương pháp QI4AE được cải tiến từ thuật toán tham lam đề xuất trong PatternAttack. Ý tưởng chính của QI4AE là kết hợp thuật toán tham lam với mô hình mã hóa tự động. Đầu tiên, ảnh đầu vào sẽ được đẩy qua mô hình mã hóa tự động để lấy ảnh đối kháng cải thiện mức thô, rồi đẩy tiếp qua thuật toán tham lam để lấy ảnh đối kháng cải thiện mức tinh chế. Thực nghiệm trên MNIST và CIFAR-10 cho thấy QI4AE có thể cải thiện chất lượng ảnh đáng kể với chi phí tính toán thấp. Nghiên cứu này đã được đăng ở hội nghị ICAART. Một công cụ đã được cài đặt để chứng minh hiệu quả của QI4AE và đã chuyển giao cho tập đoàn TSDV Việt Nam.

Cuối cùng, để nâng cao tính chắc chắn của mô hình học sâu, luận án đề xuất phương pháp cải thiện tính chắc chắn SCADefender để loại bỏ nhiễu đối kháng khỏi ảnh đầu vào. Một phần dữ liệu học của SCADefender là tập ảnh đối kháng sinh bởi nhiều phương pháp tấn công đối kháng khác nhau. Kết quả

của quá trình học là một mô hình mã hóa tự động phòng thủ có khả năng loại bỏ nhiễu đối kháng khỏi ảnh đầu vào. Thực nghiệm trên MNIST, CIFAR-10 và Fashion-MNIST cho thấy SCADefender có thể loại bỏ nhiễu đối kháng khỏi ảnh đầu vào khá tốt. Nghiên cứu này đã được đăng ở tạp chí IJPRAI. Một công cụ đã được cài đặt để chứng minh hiệu quả của PatternAttack và đã chuyển giao cho tập đoàn TSDV Việt Nam.

Các nghiên cứu được trình bày trong luận án không những có ý nghĩa về mặt lý thuyết mà còn góp phần làm phương pháp kiểm thử tính chắc chắn cho mô hình học sâu dễ dàng được áp dụng hơn trong thực tiễn. Điều này đặc biệt có ý nghĩa với những mô hình học sâu yêu cầu cao về chất lượng, có khả năng chống lại tấn công từ bên ngoài, trong đó có tấn công đối kháng. Ngoài ra, các công cụ của luận án đã được triển khai sử dụng tại TSDV Việt nam và nhận được những phản hồi tích cực.

7.2. Hướng phát triển tiếp theo

Mặc dù các kết quả nghiên cứu đã có những đóng góp cụ thể như đã trình bày nêu trên, các kết quả này còn có những hạn chế cần khắc phục. Nghiên cứu tiếp theo của luận án hướng đến giải quyết các hạn chế này. Cụ thể, các hạn chế và hướng nghiên cứu tiếp theo của luận án như sau.

Trong nghiên cứu thứ nhất, HA4FNN có hai hạn chế. Hạn chế thứ nhất là HA4FNN chưa hỗ trợ tấn công đối kháng có định hướng cho mô hình nơ-ron truyền thẳng. Trong HA4FNN, bộ giải phỏng đoán sẽ thêm nhiễu đối kháng vào ảnh dự đoán đúng để mô hình kiểm thử nhận diện sai. Một hạn chế của bộ giải phỏng đoán là chưa thêm nhiễu đối kháng được ảnh dự đoán đúng để sinh ảnh đối kháng được phân loại là một nhãn cụ thể. Hạn chế thứ hai là HA4FNN chưa hỗ trợ tấn công mô hình tích chập. Mặc dù PatternAttack được đề xuất để tấn công mô hình tích chập và coi là một giải pháp tốt hơn HA4FNN, cách tiếp cận hai phương pháp này khác nhau. Phương pháp PatternAttack định nghĩa một hàm mục tiêu và tối thiểu hóa hàm mục tiêu này bằng cách sử dụng đạo hàm để tìm ảnh đối kháng. Ngược lại, HA4FNN xây dựng chương trình C từ mô hình kiểm thử và áp dụng các kĩ thuật phân tích chương trình, thực thi tượng trưng

và bộ giải để tìm ảnh đối kháng. Bởi vì mô hình tích chập có tính phức tạp cao về kiến trúc, việc chuyển đổi mô hình thành mã nguồn chưa thực sự hiệu quả. Ngoài ra, việc sử dụng thực thi tượng trưng trên mã nguồn ứng với mô hình tích chập khá tốn chi phí do số lượng câu lệnh có thể khá lớn. Trong tương lai, luận án sẽ tiếp tục cải tiến HA4FNN để giải quyết các hạn chế này.

Trong nghiên cứu thứ hai, tuy PatternAttack có thể sinh ảnh đối kháng với chất lượng tốt và có tính đa dạng, phương pháp này có ba hạn chế sau đây. Thứ nhất, PatternAttack chưa được thực nghiệm trên ảnh có kích thước lớn như ImageNet [115]. Hiện tại, PatternAttack mới thực nghiệm trên ảnh có kích thước nhỏ như $28 \times 28 \times 1$ hoặc $28 \times 28 \times 3$. Thứ hai, việc lựa chọn kiến trúc mô hình mã hóa tự động phù hợp trong Công thức 4.1 là một thách thức. Kiến trúc mô hình mã hóa tự động quyết định khả năng mô hình liệu có thêm nhiễu đối kháng vào ảnh dự đoán đúng một cách phù hợp. Thứ ba, việc lựa chọn trọng số giữa các thành phần trong Công thức 4.1 có ảnh hưởng đến tốc độ hội tụ của quá trình học mô hình mã hóa tự động. Nếu trọng số được chọn phù hợp, quá trình học mô hình mã hóa tự động sẽ tránh việc tập trung tối tiểu hóa một thành phần và coi nhẹ thành phần còn lại. Tổng kết lại, luận án sẽ tiếp tục nghiên cứu về các kiến trúc mô hình mã hóa tự động khác nhau, kĩ thuật tự điều chỉnh trọng số như [116, 117, 118] và thực nghiệm trên bộ dữ liệu có kích thước ảnh lớn hơn.

Trong nghiên cứu thứ bốn, tuy SCADefender có thể loại bỏ được nhiều nhiễu đối kháng của ảnh đối kháng, phương pháp này có hai hạn chế. Thứ nhất, SCADefender có thể biến ảnh đầu vào đang nhận diện đúng thành sai. Nguyên nhân bởi vì khi triển khai trong thực tế, SCADefender chưa phân biệt được ảnh thực sự có nhiễu đối kháng. Thứ hai, ảnh hưởng của kiến trúc mô hình mã hóa tự động chưa được khảo sát triệt để trong thực nghiệm. Thứ ba, bộ dữ liệu để học mô hình mã hóa tự động có thể khá lớn nếu sử dụng ảnh đối kháng sinh bởi nhiều tấn công đối kháng không định hướng, từ đó khiến cho quá trình học càng ngày càng phức tạp khi ngày càng nhiều tấn công đối kháng mới được đề xuất. Do đó, tổng quát hóa đặc trưng của nhiễu đối kháng là quan trọng. Tổng kết lại, luận án sẽ nghiên cứu giải pháp để giảm thiểu khả năng sửa ảnh từ nhận diện đúng thành nhận diện sai, nghiên cứu thêm các loại kiến trúc mô hình mã hóa tự động khác nhau và tìm hiểu sâu hơn bản chất của nhiễu đối kháng.

DANH MỤC CÁC CÔNG TRÌNH KHOA HỌC CỦA TÁC GIẢ LIÊN QUAN TỚI LUẬN ÁN

1. **Duc-Anh Nguyen**, Kha Do Minh, Ngoc Nguyen Nhu, Pham Ngoc Hung (2023). *SCADefender: An Autoencoder-based Defense for CNN-based Image Classifiers*. In International Journal of Pattern Recognition and Artificial Intelligence (Q3)
2. **Duc-Anh Nguyen**, Kha Do Minh, Khoi Nguyen Le, Minh Nguyen Le, Pham Ngoc Hung (2022). *Improving Diversity and Quality of Adversarial Examples in Adversarial Transformation Network*. In Soft Computing (Q2)
3. **Duc-Anh Nguyen**, Kha Do Minh, Pham Ngoc Hung, Nguyen Le Minh (2022). *A Symbolic Execution-based Method to Perform Untargeted Attack on Feed-forward Neural Networks*. In Automated Software Engineering (Q2)
4. **Duc-Anh Nguyen**, Kha Do Minh, Duc-Anh Pham, Pham Ngoc Hung (2022). *Method for Improving Quality of Adversarial Examples*. In the 14th International Conference on Agents and Artificial Intelligence (ICAART - rank B)
5. **Duc-Anh Nguyen**, Do Minh Kha, Pham Thi To Nga, Pham Ngoc Hung (2021). *An Autoencoder-based Method for Targeted Attack on Deep Neural Network Models*. In the 15th IEEE-RIVF International Conference on Computing and Communication Technologies (RIVF - **The best paper award**)

Danh mục này gồm 05 công trình.

Từ điển chú giải

ảnh dự đoán đúng Ảnh được dự đoán đúng nhãn bởi mô hình kiểm thử. v–vii, xi, xiv, 2–5, 9, 17–27, 33–40, 42, 44–54, 56, 58–63, 65–72, 75, 80–83, 85, 87, 93, 96–100, 103, 107, 108, 116, 118, 119, 121, 123

ảnh đầu vào Ảnh có thể có nhiễu đối kháng hoặc không có nhiễu đối kháng. xiv, xv, 1, 4, 5, 7, 10, 13, 15–18, 28–30, 39, 58, 61, 62, 87, 89, 96–102, 105–107, 110, 114, 115, 117–119, 123

ảnh đối kháng Ảnh được chỉnh sửa từ ảnh dự đoán đúng, trong đó mô hình học sâu sẽ nhận diện sai ảnh chỉnh sửa này. v–viii, x, xi, xiv, xv, 2–10, 17–23, 25–27, 29, 33–40, 44–47, 49–53, 58–70, 72–83, 85, 87, 89–92, 94–119, 121, 122, 124

ảnh đối kháng cải thiện Ảnh đối kháng sau khi loại bỏ một lượng nhiễu dư thừa. 21, 63, 66–68, 75, 81, 85, 86, 90, 93

ảnh đối kháng chưa cải thiện Ảnh đối kháng trước khi áp dụng phương pháp loại bỏ nhiễu dư thừa. 84–86, 89–92, 94

bản đồ đặc trưng Kết quả áp dụng phép tính tích chập trong mô hình tích chập (feature map). 13, 17

bản đồ nổi bật Bản đồ thể hiện sự quan trọng của các điểm ảnh. 60, 62, 63

bộ giải phỏng đoán Bộ giải đề xuất để sinh ảnh đối kháng cho mô hình nơ-ron truyền thẳng. xiv, 6, 9, 33, 35, 36, 38, 44, 49, 56, 57, 116, 118

bộ giải SMT Dùng để tìm nghiệm của hệ ràng buộc. xiv, 4, 9, 23, 31–38, 40, 44, 47, 49, 52, 57, 61, 116

cải thiện tính chắc chắn Hướng nghiên cứu nâng cao khả năng chống chịu của mô hình kiểm thử khỏi tấn công đối kháng. v, viii, 5–10, 29, 97, 109, 112–115, 117

điểm số F1 Tiêu chí đánh giá chất lượng mô hình học sâu (F1-score). xiv, 1, 14, 15, 33

độ chính xác Tiêu chí đánh giá chất lượng mô hình học sâu (precision). xiv, 1, 14, 15, 33

độ chuẩn xác Tiêu chí đánh giá chất lượng mô hình học sâu (accuracy). vii, viii, xiv, 1, 14, 15, 33, 47, 71, 100, 101, 106

đường thi hành Tập các câu lệnh thu được sau khi thực thi một chương trình với một đầu vào cụ thể (test path). xiv, 4, 22, 23, 34, 39, 40, 42, 56, 61, 116, 122

hàm chỉ thị Hàm chỉ trả về 0 hoặc 1. xi, 21

hệ ràng buộc Bao gồm nhiều điều kiện, là kết quả của thực hiện thực thi tượng trưng trên đường thi hành. v, vii, xiv, 4, 22–24, 31, 32, 34–40, 44, 45, 47, 48, 55–57, 61, 83, 116

học đối kháng Một phương pháp học lại mô hình học sâu với bộ dữ liệu bổ sung, từ đó giúp mô hình học sâu có tính chắc chắn tốt hơn. 99–101, 111–115

học quá khớp Một vấn đề xảy ra khi học mô hình học sâu (overfitting). 55

kích thước khối Kích thước tập dữ liệu dùng để điều chỉnh trọng số của mô hình học sâu (batch). 13

mẫu thêm nhiễu Mô tả tập các điểm ảnh cùng một khuôn mẫu. x, xv, 7, 9, 58, 60, 63–66, 73–75, 78, 79, 104, 117

mô hình chuyển đổi đối kháng Phương pháp sinh ảnh đối kháng bằng cách học mô hình mã hóa tự động. 58

mô hình học sâu Một loại mô hình phổ biến, trong đó có hai loại phổ biến là mô hình nơ-ron truyền thẳng và mô hình tích chập (deep neural network). ix, xi, xiv, xv, 1, 2, 4–7, 9–14, 17, 18, 20, 21, 32, 33, 36, 37, 55, 56, 62, 67, 99, 116–118, 121–125

mô hình kiểm thử Mô hình được kiểm tra tính chắc chắn. vii, viii, xi, xiv, 2, 3, 5, 10, 18, 19, 22, 28, 29, 34, 39, 40, 47–49, 53, 56, 58, 59, 63, 66–73, 80, 81, 85–87, 91, 92, 96–111, 113, 114, 116, 118, 121

mô hình mã hóa tự động Một loại mô hình học sâu gồm phần mã hóa và phần giải mã (autoencoder). v, vii, xi, xv, 4, 5, 7, 9–11, 15–17, 30, 32, 58, 59, 61, 63, 64, 70–72, 76, 80, 82, 84–92, 94, 95, 99–101, 105, 107, 109, 110, 114, 116, 117, 119, 122, 123

mô hình mã hóa tự động biến thiên Một loại mô hình mã hóa tự động (variational autoencoder). 29, 97, 100, 101

mô hình mã hóa tự động biến thiên có điều kiện Một loại mô hình mã hóa tự động (conditional variational autoencoder). 28, 97, 99, 101

mô hình mã hóa tự động phòng thủ Mô hình mã hóa tự động được sử dụng để loại bỏ nhiễu dư thừa khỏi ảnh đầu vào trước khi chuyển ảnh đến mô hình học sâu. viii, xv, 96, 98–103, 105, 118

mô hình mã hóa tự động tích chập xếp chồng Một loại mô hình mã hóa tự động trong đó có nhiều lớp ẩn ở tầng phần mã hóa và phần giải mã (stacked convolutional autoencoder). v, 17, 27, 32, 70, 88, 97, 98, 101, 102, 115

mô hình nơ-ron truyền thẳng Một loại mô hình học sâu trong đó các lớp nối tiếp nhau (feedforward neural network). v, ix, xiv, 1, 3, 6, 9, 11, 12, 22, 32–36, 38–40, 43, 55–57, 116–118, 121, 122

mô hình tích chập Một loại mô hình học sâu (convolutional neural network). ix, x, 1, 9, 11, 13, 32, 56–58, 60–63, 78, 80, 81, 84, 96, 97, 99, 100, 106, 115–119, 121, 122

nhãn đích Được sử dụng trong tấn công đối kháng có định hướng, trong đó ảnh dự đoán đúng sau khi sửa sẽ có nhãn này. xi, 3, 18, 21, 26, 58, 59, 63, 67, 68, 71, 73, 80, 85

nhiễu Lượng chỉnh sửa các điểm ảnh trong một ảnh. viii, x, xi, xv, 2, 4, 7, 9, 10, 15, 18, 25, 26, 28, 29, 37, 58–61, 63–68, 73–75, 78, 79, 81, 83, 94, 97–101, 103–108, 111, 112, 116, 117, 122

nhiễu dư thừa Lượng nhiễu thêm vào ảnh và ảnh không thay đổi nhãn dự đoán. vii, xv, 4, 5, 10, 21, 60, 63, 66–69, 78–82, 84, 86–89, 91–95, 116, 117, 121, 123

nhiễu đối kháng Lượng nhiễu thêm vào khiến ảnh đang nhận diện đúng thành nhận diện sai bởi mô hình học sâu. v–vii, xiv, xv, 1–5, 7, 9, 10, 17–23, 25–27, 29, 30, 34–40, 44–47, 49–54, 57–63, 65, 67, 68, 71, 73, 75, 78–85, 90, 93, 96–103, 105, 107, 108, 110, 112, 115, 117–119, 121

phần giải mã Phần sau của mô hình mã hóa tự động truyền thống dùng để giải mã (decoder). 15–17, 122, 123

phần mã hóa Phần đầu của mô hình mã hóa tự động truyền thống dùng để mã hóa (encoder). 15–17, 122, 123

sai số bình phương trung bình Một độ đo khoảng cách giữa hai điểm trong không gian nhiều chiều (mean squared error). 85

tấn công đối kháng Một loại tấn công mô hình học sâu để kiểm tra tính chắc chắn của mô hình học sâu. v, xiv, xv, 2, 3, 5–11, 17–21, 32, 40, 57, 60, 81, 83, 84, 87, 93, 95, 100, 101, 116–119, 121, 124

tấn công đối kháng có định hướng Một loại tấn công tấn công đối kháng để kiểm tra tính chắc chắn của mô hình học sâu. xi, 3, 4, 17–19, 21, 32, 57, 58, 67, 78, 80, 103, 118, 123

tấn công đối kháng không định hướng Một loại tấn công tấn công đối kháng để kiểm tra tính chắc chắn. v, vi, viii, 3, 9, 17–19, 21, 32–34, 57, 67, 96, 98, 99, 102, 103, 105, 107–109, 115, 119

tập học Bộ dữ liệu dùng để xây dựng mô hình học sâu. vii, 1, 12–14, 28–31, 47, 48, 71, 75, 84, 85, 87, 88, 91, 97, 98, 100–102, 105, 106, 109, 110

tập kiểm thử Bộ dữ liệu dùng để đánh giá hiệu quả mô hình học sâu và không được sử dụng để xây dựng mô hình học sâu. vii, 30, 31, 47, 48, 71, 87, 91, 106, 107, 110–112

tập xác thực Bộ dữ liệu dùng để đánh giá mô hình học sâu trong quá trình xây dựng. 106, 110

thuật toán học Được dùng để điều chỉnh trọng số của mô hình học sâu, ví dụ như SGD [58]. 13

thực thi tượng trưng Một phương pháp dùng để chuyển đổi đường thi hành sang hệ ràng buộc (symbolic execution). xiv, 3, 9, 23, 34, 36, 37, 39, 42, 44, 48, 54–56, 116, 118, 119, 122

tỉ lệ giảm nhiễu Một độ đo để đánh giá chất lượng của phương pháp cải thiện chất lượng ảnh đối kháng. vi, vii, xv, 7, 20, 21, 75, 76, 82, 86, 92, 93, 117

tỉ lệ phát hiện Một tiêu chí phổ biến để đánh giá chất lượng của phương pháp phòng thủ đối kháng. viii, 29, 30, 97, 99, 106, 111–115

tỉ lệ thành công Một tiêu chí phổ biến để đánh giá chất lượng của tấn công đối kháng. vi–viii, xiv, xv, 2–7, 20, 21, 26, 33–35, 38, 44, 46, 49–54, 56, 57, 74, 75, 79, 81, 86, 90, 91, 96, 104, 107, 108, 116, 117

tính chắc chắn Khả năng một mô hình học sâu chống chịu được tấn công đối kháng (robustness). v, vii, viii, xiv, xv, 1, 2, 4–11, 17, 18, 20, 21, 29, 32–34, 37, 48, 57, 59, 60, 63, 80, 96, 97, 99, 100, 109, 112–118, 121, 122, 124

tính khái quát hóa Khả năng học kinh nghiệm từ phương pháp sửa các ảnh trong quá khứ để sửa ảnh mới. 4, 58, 60, 61, 71, 80, 81, 117

tốc độ học Một siêu tham số để điều chỉnh quá trình học của mô hình học sâu (learning rate). xi, 1, 13, 14, 47, 110

trạng thái kích hoạt nơ-ron Dùng với hàm kích hoạt ReLU. Một mẫu kích hoạt sẽ lưu các trạng thái đúng/sai của nơ-ron với đầu vào là một ảnh. xiv, 4, 6, 23, 24, 33–35, 38, 52, 53, 116

Tài liệu tham khảo

[1] Ian Goodfellow, Yoshua Bengio **and** Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[2] Jie Zhang, Mark Harman, Lei Ma **and** Yang Liu. *Machine Learning Testing: Survey, Landscapes and Horizons*. **june** 2019.

[3] Ahmed Aldahdooh, Wassim Hamidouche, Sid Ahmed Fezza **and** Olivier Deforges. "Adversarial Example Detection for DNN Models: A Review". **in***CoRR*: abs/2105.00203 (2021). arXiv: 2105.00203.

[4] Jürgen Schmidhuber. "Deep learning in neural networks: An overview". **in***Neural Networks*: 61 (**january** 2015), **pages** 85–117. DOI: 10.1016/j.neunet.2014.09.003. URL: https://doi.org/10.1016%5C%2Fj.neunet.2014.09.003.

[5] Naveed Akhtar, Ajmal Mian, Navid Kardan **and** Mubarak Shah. *Threat of Adversarial Attacks on Deep Learning in Computer Vision: Survey II*. 2021. arXiv: 2108.00401 [cs.CV].

[6] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul W. Fieguth, Jie Chen, Xinwang Liu **and** Matti Pietikäinen. "Deep Learning for Generic Object Detection: A Survey". **in***CoRR*: abs/1809.02165 (2018). arXiv: 1809.02165. URL: http://arxiv.org/abs/1809.02165.

[7] Mei Wang **and** Weihong Deng. "Deep Face Recognition: A Survey". **in***CoRR*: abs/1804.06655 (2018). arXiv: 1804.06655. URL: http://arxiv.org/abs/1804.06655.

[8] Daniel W. Otter, Julian R. Medina **and** Jugal K. Kalita. "A Survey of the Usages of Deep Learning in Natural Language Processing". **in***CoRR*: abs/1807.10854 (2018). arXiv: 1807.10854. URL: http://arxiv.org/abs/1807.10854.

[9] Ana I. Maqueda, Antonio Loquercio, Guillermo Gallego, Narciso García **and** Davide Scaramuzza. "Event-based Vision meets Deep Learning on Steering Prediction for Self-driving Cars". **in***CoRR*: abs/1804.01310 (2018). arXiv: 1804.01310. URL: http://arxiv.org/abs/1804.01310.

[10] Anselme Ndikumana, Nguyen Hoang Tran **and** Choong Seon Hong. "Deep Learning Based Caching for Self-Driving Car in Multi-access Edge Computing". **in***CoRR*: abs/1810.01548 (2018). arXiv: 1810.01548. URL: http://arxiv.org/abs/1810.01548.

[11] Zhenlong Yuan, Yongqiang Lu, Zhaoguo Wang **and** Yibo Xue. "Droid-Sec: Deep Learning in Android Malware Detection". **in***Proceedings of the 2014 ACM Conference on SIGCOMM*: SIGCOMM '14. Chicago, Illinois, USA: Association for Computing Machinery, 2014, **pages** 371–372. ISBN: 9781450328364. DOI: 10.1145/2619239.2631434. URL: https://doi.org/10.1145/2619239.2631434.

[12] R. Vinayakumar, Mamoun Alazab, K. P. Soman, Prabaharan Poornachandran **and** Sitalakshmi Venkatraman. "Robust Intelligent Malware Detection Using Deep Learning". **in***IEEE Access*: 7 (2019), **pages** 46717–46738. DOI: 10.1109/ACCESS.2019.2906934.

[13] Xiaokang Zhou, Wei Liang, Kevin I-Kai Wang, Hao Wang, Laurence T. Yang **and** Qun Jin. "Deep-Learning-Enhanced Human Activity Recognition for Internet of Healthcare Things". **in***IEEE Internet of Things Journal*: 7.7 (2020), **pages** 6429–6438. DOI: 10.1109/JIOT.2020.2985082.

[14] Znaonui Liang, Gang Zhang, Jimmy Xiangji Huang **and** Qmming Vivian Hu. "Deep learning for healthcare decision making with EMRs". **in***2014 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*: 2014, **pages** 556–559. DOI: 10.1109/BIBM.2014.6999219.

[15] Mehryar Mohri, Afshin Rostamizadeh **and** Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012. ISBN: 026201825X.

[16] Ian J. Goodfellow, Jonathon Shlens **and** Christian Szegedy. *Explaining and Harnessing Adversarial Examples*. 2015. arXiv: 1412.6572 [stat.ML].

[17] Nicholas Carlini **and** David A. Wagner. "Towards Evaluating the Robustness of Neural Networks". **in***CoRR*: abs/1608.04644 (2016). arXiv: 1608.04644.

[18] Shumeet Baluja **and** Ian Fischer. *Adversarial Transformation Networks: Learning to Generate Adversarial Examples*. 2017. arXiv: 1703.09387 [cs.NE].

[19] Alexey Kurakin, Ian J. Goodfellow **and** Samy Bengio. "Adversarial examples in the physical world". **in***CoRR*: abs/1607.02533 (2016). arXiv: 1607.02533.

[20] Youcheng Sun, Xiaowei Huang **and** Daniel Kroening. "Testing Deep Neural Networks". **in***CoRR*: abs/1803.04792 (2018). arXiv: 1803.04792.

[21] Fu Wang, Chi Zhang, Peipei Xu **and** Wenjie Ruan. "Deep Learning and Its Adversarial Robustness: A Brief Introduction". **in***Handbook on Computer Learning and Intelligence*: **chapter** Chapter 13, **pages** 547–584. DOI: 10.1142/9789811247323_0013. URL: https://www.worldscientific.com/doi/abs/10.1142/9789811247323_0013.

[22] Zhe Zhao, Guangke Chen, Jingyi Wang, Yiwei Yang, Fu Song **and** Jun Sun. "Attack as Defense: Characterizing Adversarial Examples Using Robustness". **in***Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*: ISSTA 2021. Virtual, Denmark: Association for Computing Machinery, 2021, **pages** 42–55. ISBN: 9781450384599. DOI: 10.1145/3460319.3464822. URL: https://doi.org/10.1145/3460319.3464822.

[23] Ravi Mangal, Aditya V. Nori **and** Alessandro Orso. "Robustness of Neural Networks: A Probabilistic and Practical Approach". **in***CoRR*: abs/1902.05983 (2019). arXiv: 1902.05983.

[24] Rüdiger Ehlers. "Formal Verification of Piece-Wise Linear Feed-Forward Neural Networks". **in***Automated Technology for Verification and Analysis*: 2017. URL: https://api.semanticscholar.org/CorpusID:1931807.

[25] Guy Katz, Clark W. Barrett, David L. Dill, Kyle Julian **and** Mykel J. Kochenderfer. "Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks". **in***CoRR*: abs/1702.01135 (2017). arXiv: 1702.01135.

[26] Yizhak Yisrael Elboher, Justin Gottschlich **and** Guy Katz. "An Abstraction-Based Framework for Neural Network Verification". **in***CoRR*: abs/1910.14574 (2019). arXiv: 1910.14574. URL: http://arxiv.org/abs/1910.14574.

[27] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang **and** Suman Jana. "Efficient Formal Safety Analysis of Neural Networks". **in***CoRR*: abs/1809.08098 (2018). arXiv: 1809.08098. URL: http://arxiv.org/abs/1809.08098.

[28] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang **and** Suman Jana. "Formal Security Analysis of Neural Networks using Symbolic Intervals". **in***CoRR*: abs/1804.10829 (2018). arXiv: 1804.10829. URL: http://arxiv.org/abs/1804.10829.

[29] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri **and** Martin Vechev. "AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation". **in***2018 IEEE Symposium on Security and Privacy (SP)*: 2018, **pages** 3–18. DOI: 10.1109/SP.2018.00058.

[30] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel **and** Martin Vechev. "Fast and Effective Robustness Certification". **in***Proceedings of the 32nd International Conference on Neural Information Processing Systems*: NIPS'18. Montréal, Canada: Curran Associates Inc., 2018, **pages** 10825–10836.

[31] Gagandeep Singh, Timon Gehr, Markus Püschel **and** Martin Vechev. "An Abstract Domain for Certifying Neural Networks". **in***Proc. ACM Program. Lang.*: 3.POPL (**january** 2019). DOI: 10.1145/3290354. URL: https://doi.org/10.1145/3290354.

[32] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow **and** Rob Fergus. *Intriguing properties of neural networks*. 2014. arXiv: 1312.6199 [cs.CV].

[33] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi **and** Pascal Frossard. "DeepFool: a simple and accurate method to fool deep neural networks". **in***CoRR*: abs/1511.04599 (2015). arXiv: 1511.04599.

[34] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Xiaolin Hu **and** Jun Zhu. "Discovering Adversarial Examples with Momentum". **in***CoRR*: abs/1710.06081 (2017). arXiv: 1710.06081.

[35] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras **and** Adrian Vladu. *Towards Deep Learning Models Resistant to Adversarial Attacks*. 2017. DOI: 10.48550/ARXIV.1706.06083. URL: https://arxiv.org/abs/1706.06083.

[36] Alexey Kurakin, Ian J. Goodfellow, Samy Bengio, Yinpeng Dong, Fangzhou Liao, Ming Liang, Tianyu Pang, Jun Zhu, Xiaolin Hu, Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, Alan L. Yuille, Sangxia Huang, Yao Zhao, Yuzhe Zhao, Zhonglin Han, Junjiajia Long, Yerkebulan Berdibekov, Takuya Akiba, Seiya Tokui **and** Motoki Abe. "Adversarial Attacks and Defences Competition". **in***CoRR*: abs/1804.00097 (2018). arXiv: 1804.00097.

[37] Zhou Wang, A.C. Bovik, H.R. Sheikh **and** E.P. Simoncelli. "Image quality assessment: from error visibility to structural similarity". **in***IEEE Transactions on Image Processing*: 13.4 (2004), **pages** 600–612. DOI: 10.1109/TIP.2003.819861.

[38] Alain Horé **and** Djemel Ziou. "Image Quality Metrics: PSNR vs. SSIM". **in***2010 20th International Conference on Pattern Recognition*: 2010, **pages** 2366–2369. DOI: 10.1109/ICPR.2010.579.

[39] Siddhant Bhambri, Sumanyu Muku, Avinash Tulasi **and** Arun Balaji Buduru. "A Study of Black Box Adversarial Attacks in Computer Vision". **in***CoRR*: abs/1912.01667 (2019). arXiv: 1912.01667. URL: http://arxiv.org/abs/1912.01667.

[40] Joana C. Costa, Tiago Roxo, Hugo Proença **and** Pedro R. M. Inácio. *How Deep Learning Sees the World: A Survey on Adversarial Attacks and Defenses*. 2023. arXiv: 2305.10862 [cs.CV].

[41] Yann Lecun, Léon Bottou, Yoshua Bengio **and** Patrick Haffner. "Gradient-based learning applied to document recognition". **in***Proceedings of the IEEE*: 1998, **pages** 2278–2324.

[42] Divya Gopinath, Corina S. Păsăreanu, Kaiyuan Wang, Mengshi Zhang **and** Sarfraz Khurshid. "Symbolic Execution for Attribution and Attack Synthesis in Neural Networks". **in***Proceedings of the 41st International Conference on Software Engineering: Companion Proceedings*: ICSE '19. Montreal, Quebec, Canada: IEEE Press, 2019, **pages** 282–283. DOI: 10.1109/ICSE-Companion.2019.00115.

[43] Kexin Pei, Yinzhi Cao, Junfeng Yang **and** Suman Jana. "DeepXplore: Automated Whitebox Testing of Deep Learning Systems". **in***CoRR*: abs/1705.06640 (2017). arXiv: 1705.06640.

[44] Alexey Kurakin, Ian Goodfellow **and** Samy Bengio. *Adversarial Machine Learning at Scale*. 2016. DOI: 10.48550/ARXIV.1611.01236. URL: https://arxiv.org/abs/1611.01236.

[45] Yaguan Qian, Xiaoyu Liang, Ming Kang, Bin Wang, Zhaoquan Gu, Xing Wang **and** Chunming Wu. "GAAT: Group Adaptive Adversarial Training to Improve the Trade-Off Between Robustness and Accuracy". **in***International Journal of Pattern Recognition and Artificial Intelligence*: 36.13 (2022), **page** 2251015. DOI: 10.1142/S0218001422510156. eprint: https://doi.org/10.1142/S0218001422510156. URL: https://doi.org/10.1142/S0218001422510156.

[46] Dongyu Meng **and** Hao Chen. "MagNet: a Two-Pronged Defense against Adversarial Examples". **in**(2017): arXiv: 1705.09064 [cs.CR].

[47] Yaopeng Wang, Lehui Xie, Ximeng Liu, Jia-Li Yin **and** Tingjie Zheng. "Model-Agnostic Adversarial Example Detection Through Logit Distribution Learning". **in***2021 IEEE International Conference on Image Processing (ICIP)*: 2021, **pages** 3617–3621. DOI: 10.1109/ICIP42928.2021.9506292.

[48] Keji Han, Yun Li **and** Bin Xia. "A cascade model-aware generative adversarial example detection method". **in***Tsinghua Science and Technology*: 26.6 (2021), **pages** 800–812. DOI: 10.26599/TST.2020.9010038.

[49] Uiwon Hwang, Jaewoo Park, Hyemi Jang, Sungroh Yoon **and** Nam Ik Cho. "PuVAE: A Variational Autoencoder to Purify Adversarial Examples". **in***IEEE Access*: 7 (2019), **pages** 126582–126593. DOI: 10.1109/ACCESS.2019.2939352.

[50] Xiang Li **and** Shihao Ji. "Defense-VAE: A Fast and Accurate Defense Against Adversarial Attacks". **in***Machine Learning and Knowledge Discovery in Databases*: **byeditor**Peggy Cellier **and** Kurt Driessens. Cham: Springer International Publishing, 2020, **pages** 191–207. ISBN: 978-3-030-43887-6.

[51] Yanan Yang, Frank Y. Shih **and** I-Cheng Chang. "Adaptive Image Reconstruction for Defense Against Adversarial Attacks". **in***International Journal of Pattern Recognition and Artificial Intelligence*: 36.12 (2022), **page** 2252022. DOI: 10.1142/S021800142252022X. eprint: https://doi.org/10.1142/S021800142252022X. URL: https://doi.org/10.1142/S021800142252022X.

[52] Svetlana Pavlitska, Nico Lambing **and** J. Marius Zöllner. *Adversarial Attacks on Traffic Sign Recognition: A Survey*. 2023. arXiv: 2307.08278 [cs.CV].

[53] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno **and** Dawn Song. "Robust Physical-World Attacks on Deep Learning Visual Classification". **in***2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*: 2018, **pages** 1625–1634. DOI: 10.1109/CVPR.2018.00175.

[54] Alex Krizhevsky, Vinod Nair **and** Geoffrey Hinton. "CIFAR-10". **in***Canadian Institute for Advanced Research*: (2009).

[55] G. Bebis **and** M. Georgiopoulos. "Feed-forward neural networks". **in***IEEE Potentials*: 13.4 (1994), **pages** 27–31. DOI: 10.1109/45.329294.

[56] Vinod Nair **and** Geoffrey E. Hinton. "Rectified Linear Units Improve Restricted Boltzmann Machines". **in***Proceedings of the 27th International Conference on International Conference on Machine Learning*: ICML'10. Haifa, Israel: Omnipress, 2010, **pages** 807–814. ISBN: 9781605589077.

[57] Y. Lecun, L. Bottou, Y. Bengio **and** P. Haffner. "Gradient-based learning applied to document recognition". **in***Proceedings of the IEEE*: 86.11 (1998), **pages** 2278–2324. DOI: 10.1109/5.726791.

[58] Sebastian Ruder. "An overview of gradient descent optimization algorithms". **in***arXiv preprint arXiv:1609.04747*: (2016).

[59] Yoshua Bengio, Pascal Lamblin, Dan Popovici **and** Hugo Larochelle. "Greedy Layer-Wise Training of Deep Networks". **in***Proceedings of the 19th International Conference on Neural Information Processing Systems*: NIPS'06. Canada: MIT Press, 2006, **pages** 153–160.

[60] Jonathan Masci, Ueli Meier, Dan Cireşan **and** Jürgen Schmidhuber. "Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction". **in***Artificial Neural Networks and Machine Learning – ICANN 2011*: **byeditor**Timo Honkela, Włodzisław Duch, Mark Girolami **and** Samuel Kaski. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, **pages** 52–59.

[61] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio **and** Pierre-Antoine Manzagol. "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion". **in***J. Mach. Learn. Res.*: 11 (**december** 2010), **pages** 3371–3408. ISSN: 1532-4435.

[62] Andrew Ng. "Sparse autoencoder". **in***CS294A Lecture notes*: (2011).

[63] Diederik P Kingma **and** Max Welling. *Auto-Encoding Variational Bayes*. 2014. arXiv: 1312.6114 [stat.ML].

[64] Yunchen Pu, Weiyao Wang, Ricardo Henao, Liqun Chen, Zhe Gan, Chunyuan Li **and** Lawrence Carin. "Adversarial Symmetric Variational Autoencoder". **in***CoRR*: abs/1711.04915 (2017). arXiv: 1711.04915.

[65] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly **and** Bernhard Schoelkopf. *Wasserstein Auto-Encoders*. 2019. arXiv: 1711.01558 [stat.ML].

[66] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay **and** Debdeep Mukhopadhyay. "Adversarial Attacks and Defences: A Survey". **in***CoRR*: abs/1810.00069 (2018). arXiv: 1810.00069. URL: http://arxiv.org/abs/1810.00069.

[67] Divya Gopinath, Corina S. Păsăreanu, Kaiyuan Wang, Mengshi Zhang **and** Sarfraz Khurshid. "Symbolic Execution for Attribution and Attack Synthesis in Neural Networks". **in***Proceedings of the 41st International Conference on Software Engineering: Companion Proceedings*: ICSE '19. Montreal, Quebec, Canada: IEEE Press, 2019, **pages** 282–283. DOI: 10.1109/ICSE-Companion.2019.00115.

[68] Divya Gopinath, Mengshi Zhang, Kaiyuan Wang, İsmet Burak Kadron, Corina Pasareanu **and** Sarfraz Khurshid. "Symbolic Execution for Importance Analysis and Adversarial Generation in Neural Networks". **in***2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE)*: 2019, **pages** 313–322. DOI: 10.1109/ISSRE.2019.00039.

[69] Muhammad Usman, Yannic Noller, Corina S. Pasareanu, Youcheng Sun **and** Divya Gopinath. "NEUROSPF: A tool for the Symbolic Analysis of Neural Networks". **in***CoRR*: abs/2103.00124 (2021). arXiv: 2103.00124. URL: https://arxiv.org/abs/2103.00124.

[70] James C. King. "Symbolic Execution and Program Testing". **in***Commun. ACM*: 19.7 (**july** 1976), **pages** 385–394. ISSN: 0001-0782. DOI: 10.1145/360248.360252. URL: https://doi.org/10.1145/360248.360252.

[71] Clark Barrett, Pascal Fontaine **and** Cesare Tinelli. *The Satisfiability Modulo Theories Library (SMT-LIB)*. www.SMT-LIB.org. 2016.

[72] Han Xiao, Kashif Rasul **and** Roland Vollgraf. "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms". **in***CoRR*: abs/1708.07747 (2017). arXiv: 1708.07747.

[73] Leonardo De Moura **and** Nikolaj Bjørner. "Z3: An Efficient SMT Solver". **in***Proceedings of the Theory and Practice of Software, 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*: TACAS'08/ETAPS'08. Budapest, Hungary: Springer-Verlag, 2008, **pages** 337–340. ISBN: 3-540-78799-2, 978-3-540-78799-0.

[74] Jochen Hoenicke **and** Tanja Schindler. "Solving and Interpolating Constant Arrays Based on Weak Equivalences". **in***Verification, Model Checking, and Abstract Interpretation - 20th International Conference, VMCAI 2019, Cascais, Portugal, January 13-15, 2019, Proceedings*: **byeditor**Constantin Enea **and** Ruzica Piskac. **volume** 11388. Lecture Notes in Computer Science. Springer, 2019, **pages** 297–317. DOI: 10.1007/978-3-030-11245-5_14. URL: https://doi.org/10.1007/978-3-030-11245-5%5C_14.

[75] Lei Ma, Felix Juefei-Xu, Jiyuan Sun, Chunyang Chen, Ting Su, Fuyuan Zhang, Minhui Xue, Bo Li, Li Li, Yang Liu, Jianjun Zhao **and** Yadong Wang. "DeepGauge: Comprehensive and Multi-Granularity Testing Criteria for Gauging the Robustness of Deep Learning Systems". **in***CoRR*: abs/1803.07519 (2018). arXiv: 1803.07519.

[76] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou **and** Alexei A. Efros. "Image-to-Image Translation with Conditional Adversarial Networks". **in***CoRR*: abs/1611.07004 (2016). arXiv: 1611.07004.

[77] Justin Johnson, Alexandre Alahi **and** Fei-Fei Li. "Perceptual Losses for Real-Time Style Transfer and Super-Resolution". **in***CoRR*: abs/1603.08155 (2016). arXiv: 1603.08155.

[78] Chaowei Xiao, Jun-Yan Zhu, Bo Li, Warren He, Mingyan Liu **and** Dawn Song. "Spatially Transformed Adversarial Examples". **in***CoRR*: abs/1801.02612 (2018). arXiv: 1801.02612.

[79] Yuchi Tian, Kexin Pei, Suman Jana **and** Baishakhi Ray. "DeepTest: Automated Testing of Deep-Neural-Network-driven Autonomous Cars". **in***CoRR*: abs/1708.08559 (2017). arXiv: 1708.08559. URL: http://arxiv.org/abs/1708.08559.

[80] Nicky Williams, Bruno Marre, Patricia Mouy **and** Muriel Roger. "PathCrawler: Automatic Generation of Path Tests by Combining Static and Dynamic Analysis". **in***Proceedings of the 5th European Conference on Dependable Computing*: EDCC'05. Budapest, Hungary: Springer-Verlag, 2005, **pages** 281–292. ISBN:

3540257233. DOI: 10.1007/11408901_21. URL: https://doi.org/10.1007/11408901_21.

[81] Patrice Godefroid, Nils Klarlund **and** Koushik Sen. "DART: Directed Automated Random Testing". **in***Proceedings of the 2005 ACM SIGPLAN Conference on Programming Language Design and Implementation*: PLDI '05. Chicago, IL, USA: Association for Computing Machinery, 2005, **pages** 213–223. ISBN: 1595930566. DOI: 10.1145/1065010.1065036. URL: https://doi.org/10.1145/1065010.1065036.

[82] Koushik Sen, Darko Marinov **and** Gul Agha. "CUTE: A Concolic Unit Testing Engine for C". **in***Proceedings of the 10th European Software Engineering Conference Held Jointly with 13th ACM SIGSOFT International Symposium on Foundations of Software Engineering*: ESEC/FSE-13. Lisbon, Portugal: Association for Computing Machinery, 2005, **pages** 263–272. ISBN: 1595930140. DOI: 10.1145/1081706.1081750. URL: https://doi.org/10.1145/1081706.1081750.

[83] Ting Su, Geguang Pu, Bin Fang, Jifeng He, Jun Yan, Siyuan Jiang **and** Jianjun Zhao. "Automated Coverage-Driven Test Data Generation Using Dynamic Symbolic Execution". **in***2014 Eighth International Conference on Software Security and Reliability (SERE)*: 2014, **pages** 98–107. DOI: 10.1109/SERE.2014.23.

[84] David M. Perry, Andrea Mattavelli, Xiangyu Zhang **and** Cristian Cadar. "Accelerating Array Constraints in Symbolic Execution". **in***Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis*: ISSTA 2017. Santa Barbara, CA, USA: Association for Computing Machinery, 2017, **pages** 68–78. ISBN: 9781450350761. DOI: 10.1145/3092703.3092728. URL: https://doi.org/10.1145/3092703.3092728.

[85] Youcheng Sun, Min Wu, Wenjie Ruan, Xiaowei Huang, Marta Kwiatkowska **and** Daniel Kroening. "Concolic Testing for Deep Neural Networks". **in***CoRR*: abs/1805.00089 (2018). arXiv: 1805.00089.

[86] Roberto Bruttomesso, Alessandro Cimatti, Anders Franzén, Alberto Griggio **and** Roberto Sebastiani. "The MathSAT 4 SMT Solver". **in***Proceedings of the 20th International Conference on Computer Aided Verification*: CAV '08. Princeton, NJ, USA: Springer-Verlag, 2008, **pages** 299–303. ISBN: 9783540705437. DOI: 10.1007/978-3-540-70545-1_28. URL: https://doi.org/10.1007/978-3-540-70545-1_28.

[87] Bruno Dutertre **and** Leonardo de Moura. "A fast linear-arithmetic solver for DPLL (T)". **inaugust** 2006: **pages** 81–94. ISBN: 978-3-540-37406-0. DOI: 10.1007/11817963_11.

[88] Diederik P. Kingma **and** Jimmy Ba. *Adam: A Method for Stochastic Optimization*. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015. 2014. URL: http://arxiv.org/abs/1412.6980.

[89] Dan Hendrycks **and** Kevin Gimpel. "Bridging Nonlinearities and Stochastic Regularizers with Gaussian Error Linear Units". **in***CoRR*: abs/1606.08415 (2016). arXiv: 1606.08415. URL: http://arxiv.org/abs/1606.08415.

[90] Djork-Arné Clevert, Thomas Unterthiner **and** Sepp Hochreiter. "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)." **in***ICLR (Poster)*: **byeditor**Yoshua Bengio **and** Yann LeCun. 2016. URL: http://dblp.uni-trier.de/db/conf/iclr/iclr2016.html#ClevertUH15.

[91] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever **and** Ruslan Salakhutdinov. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". **in***J. Mach. Learn. Res.*: 15.1 (**january** 2014), **pages** 1929–1958. ISSN: 1532-4435.

[92] K. Simonyan, A. Vedaldi **and** Andrew Zisserman. "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps". **in***CoRR*: abs/1312.6034 (2013).

[93] Jindong Gu **and** Volker Tresp. "Saliency Methods for Explaining Adversarial Attacks". **in***CoRR*: abs/1908.08413 (2019). arXiv: 1908.08413.

[94] Matthew D Zeiler **and** Rob Fergus. *Visualizing and Understanding Convolutional Networks*. 2013. arXiv: 1311.2901 [cs.CV].

[95] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox **and** Martin Riedmiller. "Striving for simplicity: The all convolutional net". **in***arXiv preprint arXiv:1412.6806*: (2014).

[96] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik **and** Ananthram Swami. "The Limitations of Deep Learning in Adversarial Settings". **in***CoRR*: abs/1511.07528 (2015). arXiv: 1511.07528.

[97] Chunshui Cao, Xianming Liu, Yi Yang, Yinan Yu, Jiang Wang, Zilei Wang, Yongzhen Huang, Liang Wang, Chang Huang, Wei Xu, Deva Ramanan **and** Thomas S. Huang. "Look and Think Twice: Capturing Top-Down Visual Attention with Feedback Convolutional Neural Networks". **in***ICCV*: 2015, **pages** 2956–2964.

[98] Jianming Zhang, Zhe Lin, Jonathan Brandt, Xiaohui Shen **and** Stan Sclaroff. "Top-down Neural Attention by Excitation Backprop". **in***CoRR*: abs/1608.00507 (2016). arXiv: 1608.00507.

[99] Ruth Fong **and** Andrea Vedaldi. "Interpretable Explanations of Black Boxes by Meaningful Perturbation". **in***CoRR*: abs/1704.03296 (2017). arXiv: 1704.03296.

[100] Piotr Dabkowski **and** Yarin Gal. *Real Time Image Saliency for Black Box Classifiers*. 2017. arXiv: 1705.07857 [stat.ML].

[101] Fuxun Yu, Qide Dong **and** Xiang Chen. "ASP: A Fast Adversarial Attack Example Generation Framework based on Adversarial Saliency Prediction". **in***CoRR*: abs/1802.05763 (2018). arXiv: 1802.05763.

[102] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner **and** Aleksander Madry. *Robustness May Be at Odds with Accuracy*. 2019. arXiv: 1805.12152 [stat.ML].

[103] Christian Etmann, Sebastian Lunz, Peter Maass **and** Carola-Bibiane Schönlieb. *On the Connection Between Adversarial Robustness and Saliency Map Interpretability*. 2019. arXiv: 1905.04172 [stat.ML].

[104] Alex Krizhevsky, Ilya Sutskever **and** Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". **in***Commun. ACM*: 60.6 (**may** 2017), **pages** 84–90. ISSN: 0001-0782. DOI: 10.1145/3065386.

[105] Ivan Evtimov, Kevin Eykholt, Earlence Fernandes, Tadayoshi Kohno, Bo Li, Atul Prakash, Amir Rahmati **and** Dawn Song. "Robust Physical-World Attacks on Machine Learning Models". **in***CoRR*: abs/1707.08945 (2017). arXiv: 1707.08945. URL: http://arxiv.org/abs/1707.08945.

[106] Pin-Yu Chen, Yash Sharma, Huan Zhang, Jinfeng Yi **and** Cho-Jui Hsieh. *EAD: Elastic-Net Attacks to Deep Neural Networks via Adversarial Examples*. 2018. arXiv: 1709.04114 [stat.ML].

[107] Pouya Samangouei, Maya Kabkab **and** Rama Chellappa. *Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models*. 2018. arXiv: 1805.06605 [cs.CV].

[108] Byeong Cheon Kim, Jung Uk Kim, Hakmin Lee **and** Yong Man Ro. "Revisiting Role of Autoencoders in Adversarial Settings". **in***CoRR*: abs/2005.10750 (2020). arXiv: 2005.10750.

[109] Haowen Liu, Ping Yi, Hsiao-Ying Lin, Jie Shi **and** Weidong Qiu. "DAFAR: Defending against Adversaries by Feedback-Autoencoder Reconstruction". **in***CoRR*: abs/2103.06487 (2021). arXiv: 2103.06487.

[110] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh **and** Patrick McDaniel. *Ensemble Adversarial Training: Attacks and Defenses*. 2020. arXiv: 1705.07204 [stat.ML].

[111] Tao Bai, Jinqi Luo, Jun Zhao, Bihan Wen **and** Qian Wang. "Recent Advances in Adversarial Training for Adversarial Robustness". **in***CoRR*: abs/2102.01356 (2021). arXiv: 2102.01356. URL: https://arxiv.org/abs/2102.01356.

[112] Qi-Zhi Cai, Min Du, Chang Liu **and** Dawn Song. "Curriculum Adversarial Training". **in***CoRR*: abs/1805.04807 (2018). arXiv: 1805.04807. URL: http://arxiv.org/abs/1805.04807.

[113] Yogesh Balaji, Tom Goldstein **and** Judy Hoffman. "Instance adaptive adversarial training: Improved accuracy tradeoffs in neural nets". **in***CoRR*: abs/1910.08051 (2019). arXiv: 1910.08051. URL: http://arxiv.org/abs/1910.08051.

[114] Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar **and** Aleksander Madry. "Adversarially Robust Generalization Requires More Data". **in***CoRR*: abs/1804.11285 (2018). arXiv: 1804.11285. URL: http://arxiv.org/abs/1804.11285.

[115] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li **and** Li Fei-Fei. "Imagenet: A large-scale hierarchical image database". **in***2009 IEEE conference on computer vision and pattern recognition*: Ieee. 2009, **pages** 248–255.

[116] Alex Kendall, Yarin Gal **and** Roberto Cipolla. "Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics". **in***CoRR*: abs/1705.07115 (2017). arXiv: 1705.07115. URL: http://arxiv.org/abs/1705.07115.

[117] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee **and** Andrew Rabinovich. "GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks". **in***CoRR*: abs/1711.02257 (2017). arXiv: 1711.02257. URL: http://arxiv.org/abs/1711.02257.

[118] Rick Groenendijk, Sezer Karaoglu, Theo Gevers **and** Thomas Mensink. "Multi-Loss Weighting with Coefficient of Variations". **in***CoRR*: abs/2009.01717 (2020). arXiv: 2009.01717. URL: https://arxiv.org/abs/2009.01717.