ĐẠI HỌC QUỐC GIA HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG

PHAN THỊ TUYẾT LAN
ĐỖ MINH THỌ

KHÓA LUẬN TỐT NGHIỆP

NGHIÊN CỨU KHẢ NĂNG KHÁNG MẪU ĐỐI KHÁNG CỦA CÁC TRÌNH PHÁT HIỆN XÂM NHẬP DỰA TRÊN HỌC MÁY

A STUDY ON THE ADVERSARIAL ATTACK RESISTANCE OF LEARNING - BASED INTRUSION DETECTION SYSTEMS

CỬ NHÂN NGÀNH AN TOÀN THÔNG TIN

TP. Hồ Chí Minh, Tháng 6 - 2024

ĐẠI HỌC QUỐC GIA HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG

PHAN THỊ TUYẾT LAN - 20521523
ĐỖ MINH THỌ - 20521972

KHÓA LUẬN TỐT NGHIỆP

NGHIÊN CỨU KHẢ NĂNG KHÁNG MẪU ĐỐI KHÁNG CỦA CÁC TRÌNH PHÁT HIỆN XÂM NHẬP DỰA TRÊN HỌC MÁY

A STUDY ON THE ADVERSARIAL ATTACK RESISTANCE OF LEARNING - BASED INTRUSION DETECTION SYSTEMS

CỬ NHÂN NGÀNH AN TOÀN THÔNG TIN

GIẢNG VIÊN HƯỚNG DẪN:
ThS. Nghi Hoàng Khoa

TP.Hồ Chí Minh - Tháng 6 - 2024

LỜI CẢM ƠN

Trong quá trình nghiên cứu và hoàn thành khóa luận, nhóm đã nhận được sự định hướng, giúp đỡ, các ý kiến đóng góp quý báu và những lời động viên của các giáo viên hướng dẫn và giáo viên bộ môn. Nhóm xin bày tỏ lời cảm ơn tới thầy ThS. Nghi Hoàng Khoa đã tận tình trực tiếp hướng dẫn, giúp đỡ trong quá trình nghiên cứu.

Nhóm xin gửi lời cảm ơn đến gia đình và bạn bè đã động viên, đóng góp ý kiến trong quá trình làm khóa luận tốt nghiệp.

Nhóm cũng chân thành cảm ơn các quý thầy cô trường Đại học Công nghệ Thông tin - ĐHQG TP.HCM, đặc biệt là các thầy cô khoa Mạng máy tính và Truyền thông, các thầy cô thuộc bộ môn An toàn Thông tin đã giúp đỡ nhóm.

Đồng tác giả

Phan Thị Tuyết Lan **Đỗ Minh Thọ**

MỤC LỤC

DANH MỤC CÁC KÝ HIỆU, CÁC CHỮ VIẾT TẮT

$L(\theta)$	Hàm mất mát
IDS	Intrusion Detection System
GAN	Generative Adversarial Network
GNB	Gaussian Naive Bayes
DT	Decision Tree
XGB	Extreme Gradient Boosting
ML	Machine Learning
DL	Deep Learning
DNN	Deep Neural Networks
LSTM	Long Short Term Memory
WB	White-box
BB	Black-box
FGSM	Fast Gradient Method
ZOO	Zeroth Order Optimization
C&W	Carlini and Wagner
AI	Artificial Intelligence
AVG	Average Probs

DANH MỤC CÁC HÌNH VẼ

DANH MỤC CÁC BẢNG BIỂU

TÓM TẮT KHÓA LUẬN

Trong những năm gần đây, nghiên cứu về an ninh mạng, đặc biệt là trong lĩnh vực hệ thống phát hiện xâm nhập (IDS), đã không ngừng tiến bộ. Các nỗ lực này đã thành công trong việc tích hợp mô hình học máy để phát hiện và phân loại các hình thức tấn công vào hệ thống. Tuy nhiên, sự phát triển này đồng thời mở ra các chiến lược tấn công ngày càng tinh vi hơn.

Đặc biệt là các cuộc tấn công đối kháng vào các IDS dựa trên học máy, đã trở nên rất tinh vi, khiến cho các hệ thống IDS không thể phát hiện được các hành vi tấn công và dẫn đến lãng phí tài nguyên. Ví dụ, vào năm 2014, trong vụ việc của Sony Pictures, tin tặc đã sử dụng các kĩ thuật tạo mẫu đối kháng để làm cho dữ liệu mạng độc hại giống hệt như dữ liệu hợp pháp, qua đó vượt qua hệ thống IDS của Sony và gây ra một vụ rò rỉ dữ liệu lên đến hàng terabyte, bao gồm các bộ phim chưa phát hành, email nội bộ và thông tin cá nhân của nhân viên.

Chính vì vậy, trong khóa luận này, nhóm tôi muốn nghiên cứu khả năng kháng mẫu đối kháng của các trình phát hiện xâm nhập dựa trên học máy, thêm đó là học sâu và học tổng hợp trước các mẫu tấn công đối kháng mà nhóm chúng tôi tạo ra. Để tạo mẫu đối kháng nhóm chúng tôi sử dụng mô hình GAN kết hợp với Multimodal Learning, đồng thời sử dụng thêm các thuật toán tạo mẫu đối kháng là FGSM, Deepfool, ZOO, C&W. Đánh giá khả năng tạo mẫu đối kháng của từng phương pháp, đồng thời sử dụng các mẫu đối kháng được tạo để đánh giá tính chuyển giao của mẫu đối kháng với các trình phát hiện xâm nhập khác. Cùng với việc đánh giá khả năng phòng thủ của các mô hình phát hiện xâm nhập sau khi đã được huấn luyện lại bằng phương pháp phòng thủ Adversarial Training, sử dụng thêm mẫu đối kháng vừa tạo để làm dữ liệu đào tạo cho mô hình phát hiện xâm nhập. Cuối cùng, nghiên cứu sẽ đề xuất các hướng tiếp cận

để phát triển hơn trong tương lai.

CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI

Trong chương này, chúng tôi xin trình bày tóm tắt về bài toán khả năng kháng mẫu đối kháng của IDS dựa trên học máy. Đồng thời đưa ra mục tiêu, phạm vi nghiên cứu, cũng như cấu trúc của khóa luận.

1.1. Đặt vấn đề

Trong bối cảnh công nghệ thông tin ngày càng phát triển, mối đe dọa an ninh mạng trở nên ngày càng phức tạp và tinh vi hơn. Việc bảo vệ hệ thống máy tính, cơ sở hạ tầng công nghệ thông tin và mạng khỏi các cuộc tấn công mạng là một thách thức lớn đối với các tổ chức và cá nhân. Ở các tổ chức từ doanh nghiệp đến cơ quan chính phủ đều lưu trữ và xử lý các dữ liệu nhạy cảm. Việc bị tấn công có thể gây ra hậu quả nghiêm trọng, từ mất mát dữ liệu, thiệt hại tài chính đến ảnh hưởng uy tín. Do đó, việc phát triển các phương pháp hiệu quả để phát hiện và ngăn chặn các cuộc tấn công là cực kỳ quan trọng. Trong số các phương pháp bảo vệ, hệ thống phát hiện xâm nhập (Intrusion Detection Systems - IDS) đóng vai trò quan trọng trong việc giám sát và phát hiện các hoạt động bất thường hoặc trái phép.

Với sự bùng nổ về trí tuệ nhân tạo, kỹ thuật học máy được ứng dụng vào hệ thống IDS. Các mô hình học máy đã chứng minh được hiệu quả vượt trội trong việc phát hiện các mẫu bất thường và các cuộc tấn công mạng so với các phương pháp truyền thống. IDS dựa trên học máy đang được sử dụng rộng rãi trong các hệ thống an ninh mạng hiện đại, minh chứng cho tính hiệu quả và tiềm năng của chúng.

Một thách thức mới nảy sinh trong việc sử dụng học máy là các mẫu đối kháng. Mẫu đối kháng là các đầu vào được tạo ra để đánh lừa các mô hình học

máy, khiến chúng đưa ra kết quả sai lệch. Điều này đặc biệt nguy hiểm trong bối cảnh an ninh mạng, nơi mà một cuộc tấn công có thể không bị phát hiện do các mẫu đối kháng. Nghiên cứu khả năng kháng mẫu đối kháng của IDS là một bước quan trọng để nâng cao an ninh mạng, bảo vệ hệ thống khỏi các cuộc tấn công tinh vi hơn.

Từ những vấn đề trên, chúng tôi thấy rằng việc nghiên cứu khả năng kháng mẫu đối kháng của IDS dựa trên học máy là rất quan trọng. Vì vậy đó là lý do mà chúng tôi quyết định chọn đề tài này.

1.2. Mục tiêu nghiên cứu

Trong khóa luận này, mục tiêu nghiên cứu của nhóm chúng tôi là xây dựng, hiện thực lại mô hình Generative Adversarial Network (GAN) kết hợp với kỹ thuật Multimodal Learning để tạo ra dữ liệu đối kháng nhằm đánh lừa các NIDS dựa trên học máy, học sâu và học tổng hợp. Đồng thời sử dụng thêm các thuật toán tạo mẫu đối kháng như FGSM, ZOO, Deefool, C&W để tạo mẫu đối kháng. Sau đó thực hiện so sánh khả năng đánh lừa NIDS của các dữ liệu đối kháng được tạo từ các phương pháp được sử dụng ở trên.

Ngoài ra, nhóm chúng tôi thực hiện đánh giá khả năng chuyển giao của các dữ liệu đối kháng trên các NIDS khác nhau. Áp dụng biện pháp đào tạo đối kháng để ngăn chặn các tấn công đối kháng.

1.3. Phạm vi nghiên cứu

- Tạo ra lưu lượng đối kháng để đánh lừa NIDS được xây dựng dựa trên học máy, học sâu, học tổng hợp. Sử dụng GAN kết hợp với Multimodal Learning và các thuật toán tạo mẫu đối kháng để tạo ra lưu lượng đối kháng.
- Kiểm tra lưu lượng đối kháng trên các NIDS khác nhau để đánh giá khả năng chuyển giao.

- Áp dụng các biện pháp để ngăn chặn tấn công đối kháng.

1.4. Đối tượng nghiên cứu

- Hệ thống phát hiện xâm nhập.
- Machine Learning, Deep Learning, Ensemble Learning và Multimodal Learning.
- Mô hình mạng sinh đối kháng (GAN).
- Các thuật toán tạo mẫu đối kháng FGSM, ZOO, DeepDeepfool, C&W.
- Tính chuyển giao (Transferability).
- Các phương pháp ngăn chặn tấn công đối kháng.

1.5. Phương pháp nghiên cứu

- Tìm hiểu các kiến thức nền tảng liên quan đến các đối tượng nghiên cứu trong khóa luận.
- Hiện thực lại các mô hình được sử dụng trong các nghiên cứu liên quan.
- Triển khai, đánh giá kết quả từ các mô hình được đề xuất, sử dụng trong khóa luận.

1.6. Cấu trúc khóa luận tốt nghiệp

Khóa luận có cấu trúc gồm 5 chương như sau:

- Chương 1: TỔNG QUAN ĐỀ TÀI
  Trình bày khái quát định hướng nghiên cứu của khóa luận mà chúng tôi muốn hướng tới.

- Chương 2: CƠ SỞ LÝ THUYẾT

  Trình bày các định nghĩa, khái niệm cũng như những kiến thức nền tảng để có thể thực hiện được nghiên cứu. Đồng thời trình bày sơ lược một số công trình liên quan có cùng hướng nghiên cứu.

- Chương 3: PHƯƠNG PHÁP THỰC HIỆN

  Là phần trọng tâm của khoá luận, trình bày những nội dung chính về phương pháp thực hiện và mô hình được sử dụng.

- Chương 4: HIỆN THỰC, ĐÁNH GIÁ VÀ THẢO LUẬN

  Đề cập đến quá trình hiện thực hóa phương pháp đề cập ở Chương 3. Sau đó trình bày phương pháp thực nghiệm, đánh giá kết quả và thảo luận chung

- Chương 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

  Đưa ra kết luận về đề tài, đề xuất một số hướng phát triển mở rộng cho các nghiên cứu trong tương lai.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

Trong chương này, chúng tôi trình bày các cơ sở lý thuyết của khóa luận. Các kiến thức về Intrusion Detection System, Machine Learning, Deep Learning và Ensemble Learning. Lý thuyết về tấn công đối kháng, cũng như các kỹ thuật tạo mẫu đối kháng. Bên cạnh đó trình bày thêm về phương pháp Transferability và phương pháp phòng thủ tấn công đối kháng.

2.1. Intrusion Detection System

Hệ thống phát hiện xâm nhập (Intrusion Detection System - IDS) là một công cụ an ninh mạng được thiết kế nhằm bảo vệ hệ thống và phát hiện các hoạt động đáng ngờ. Chức năng chính của IDS là phát hiện các dấu hiệu của các cuộc tấn công mạng, bao gồm xâm nhập trái phép, khai thác lỗ hổng bảo mật và các hoạt động độc hại khác. Tùy thuộc vào phạm vi triển khai và nhiệm vụ cụ thể, IDS được phân loại thành nhiều loại khác nhau:

- **NIDS - Network Intrusion Detection System:** tập trung vào việc theo dõi và phát hiện các hoạt động xâm nhập trong môi trường mạng. Hệ thống này thường được triển khai ở cấp độ mạng, đặc biệt tại các điểm truy cập chính hoặc các biên mạng, nơi dễ bị đe dọa. NIDS giám sát lưu lượng mạng đến và đi từ tất cả các thiết bị, cho phép quét toàn bộ lưu lượng để phát hiện các dấu hiệu của các cuộc tấn công mạng.
- **HIDS - Host Intrusion Detection System:** chuyên giám sát và phát hiện các hoạt động xâm nhập trên một máy tính hoặc hệ thống cụ thể. HIDS được cài đặt trực tiếp trên từng thiết bị cần bảo vệ, theo dõi các gói dữ liệu ra vào trên mỗi máy. Nó tập trung vào việc bảo vệ truy cập nội bộ

và bảo vệ từng máy tính trước các mối đe dọa bên trong, do đó không thể giám sát toàn bộ lưu lượng mạng như NIDS.

- **Signature-based IDS:** loại hệ thống IDS này hoạt động bằng cách so sánh các hoạt động mạng với các mẫu tấn công đã biết trước. Nếu phát hiện sự trùng khớp, hệ thống sẽ đưa ra cảnh báo. Tuy nhiên, Signature-based IDS gặp khó khăn trong việc phát hiện các cuộc tấn công sử dụng các mẫu liên tục thay đổi.
- **Anomaly-based IDS:** loại hệ thống IDS này giám sát hoạt động tổng thể của mạng và hệ thống để phát hiện các hành vi bất thường hoặc khả nghi. Thay vì dựa vào các mẫu tấn công cụ thể, Anomaly-based IDS phát hiện các hành vi không bình thường bằng cách so sánh chúng với mức bình thường đã được thiết lập trước đó (baseline). Khi phát hiện lưu lượng truy cập khác biệt so với baseline, hệ thống sẽ đưa ra cảnh báo.

2.2. Machine Learning và Deep Learning

Ở phần này, chúng tôi sẽ đề cập tới khái niệm, quy trình, cách thức hoạt động và các thuật toán tiêu biểu của Machine Learning và Deep Learning.

2.2.1. Machine Learning

2.2.1.1. Khái niệm

Machine learning (học máy) là một lĩnh vực của trí tuệ nhân tạo (Artificial Intelligence - AI) tập trung vào việc phát triển các thuật toán và mô hình để máy tính có thể tự động học từ dữ liệu và cải thiện hiệu suất của mình theo thời gian.

2.2.1.2. Phân loại

Machine Learning vẫn cần sự can thiệp của con người. Tuy nhiên, mức độ tham gia của con người thay đổi tùy theo loại hình machine learning. Cụ thể, có ba loại chính của machine learning như sau:

Học có giám sát (Supervised Machine Learning)

Máy học có giám sát sử dụng các tập dữ liệu được gán nhãn để xây dựng các thuật toán phân loại hoặc dự đoán với độ chính xác cao. Công nghệ này giúp giải quyết nhiều vấn đề thực tế, chẳng hạn như phân loại thư rác trong hộp thư điện tử.

Máy học không giám sát (Unsupervised Machine Learning)

Máy học không giám sát phân tích và gom cụm các tập dữ liệu không được gán nhãn thông qua các thuật toán. Điều này cho phép hệ thống khám phá các mẫu hoặc nhóm dữ liệu ẩn mà không cần sự can thiệp của con người. Nhờ khả năng tìm ra sự tương đồng và khác biệt trong dữ liệu, máy học không giám sát lý tưởng cho việc phân tích dữ liệu, nhận dạng hình ảnh, và phân tích thông tin khách hàng. Nó cũng giúp giảm số lượng đặc trưng trong mô hình thông qua các thuật toán giảm chiều dữ liệu như Phân tích thành phần chính (PCA - Principal Component Analysis) và Phân tích giá trị đơn lẻ (SVD - Singular Value Decomposition).

Máy học bán giám sát (Semi-supervised Machine Learning)

Máy học bán giám sát kết hợp giữa máy học có giám sát và không giám sát. Nó sử dụng một tập dữ liệu nhỏ được gán nhãn để trích xuất đặc trưng và phân loại từ một tập dữ liệu lớn hơn không được gán nhãn. Phương pháp này hiệu quả trong các trường hợp thiếu dữ liệu gán nhãn, giúp huấn luyện các thuật toán máy học có giám sát một cách nhanh chóng và hiệu quả.

2.2.1.3. Các thuật toán thường được sử dụng

Chúng tôi sẽ đề cập tới 4 thuật toán thường được sử dụng trong các mô hình Machine Learning.

1. **Linear Regression - Hồi quy tuyến tính**

   Hồi quy tuyến tính là mô hình cố gắng tìm đường thẳng (hoặc siêu phẳng trong không gian nhiều chiều) tốt nhất mà mô hình hóa mối quan hệ giữa biến độc lập X và biến phụ thuộc Y. Công thức tổng quát 2.1:

$$Y = \beta_0 + \beta_1 X + \epsilon \tag{2.1}$$

   Trong đó:

   - β_0 là điểm cắt trục Y

   - β_1 là độ dốc

   - ϵ là sai số

   Đặc điểm: Dễ hiểu, dễ triển khai, hiệu quả với dữ liệu tuyến tính.

   Ứng dụng: Dự đoán giá nhà, dự báo doanh thu, phân tích xu hướng.

2. **Logistic Regression - Hồi quy logistic**

   Thuật toán học có giám sát này đưa ra dự đoán cho các biến phản hồi phân loại (câu trả lời có/không). Hồi quy logistic sử dụng hàm logistic (hàm sigmoid) để mô hình hóa xác suất của biến nhị phân Y. Công thức 2.2:

$$P(Y = 1|X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}} \tag{2.2}$$

   Trong đó:

- $P(Y = 1|X)$: Xác suất của biến phụ thuộc Y bằng 1 (sự kiện xảy ra) khi biết giá trị của biến độc lập X. Đây là xác suất điều kiện, nghĩa là xác suất của Y dựa trên điều kiện của X.

- Y: Biến phụ thuộc nhị phân, có thể nhận giá trị 0 hoặc 1. Ví dụ, Y có thể biểu thị một sự kiện như "email là spam" (1) hoặc "email không phải là spam" (0).

- X: Biến độc lập hoặc biến dự đoán. Đây có thể là một giá trị đơn hoặc một vector chứa nhiều biến đầu vào. Ví dụ, X có thể bao gồm các đặc trưng như "số lượng từ trong email", "có chứa từ khóa 'khuyến mãi'", v.v.

- β_0: Hệ số chặn hoặc hằng số giao nhau. Đây là giá trị của Y khi tất cả các biến X bằng 0.

- β_1: Hệ số hồi quy tương ứng với biến X. Nó đại diện cho mức độ ảnh hưởng của X lên xác suất của Y.

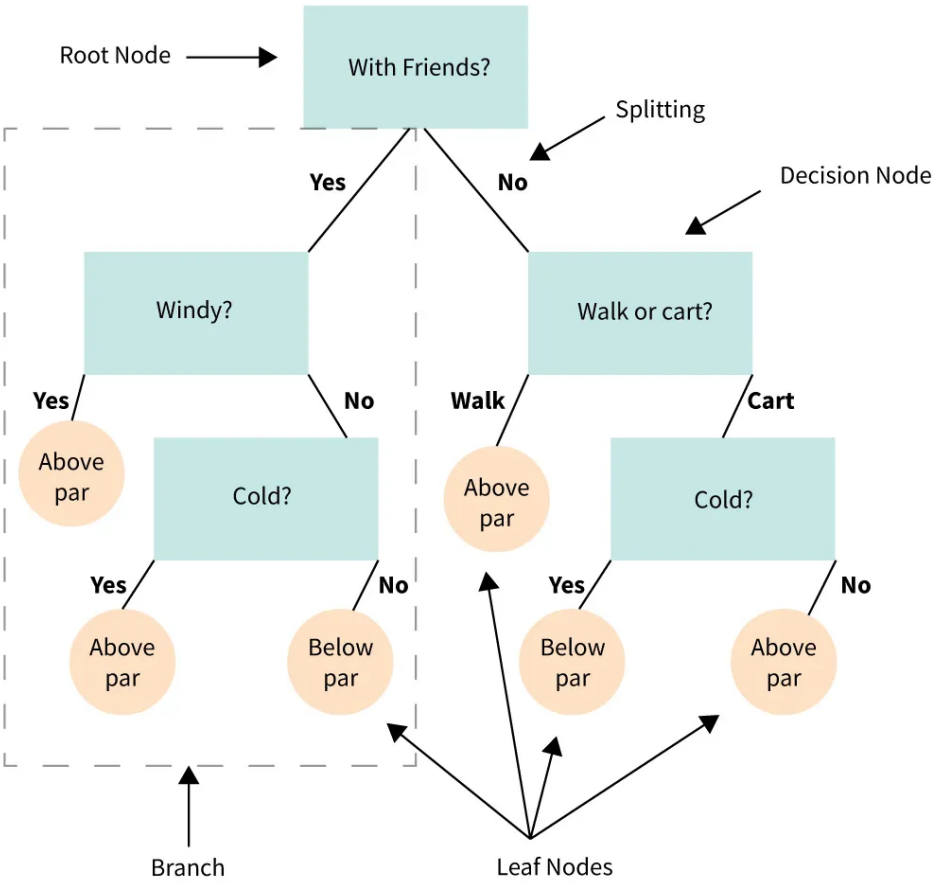
- e: Cơ số của logarit tự nhiên, khoảng bằng 2.71828.

Đặc điểm: Hiệu quả cho các bài toán phân loại, dễ triển khai và giải thích.

Ứng dụng: Phân loại nhị phân như chẩn đoán bệnh, phân loại email spam.

3. **Decision Tree - Cây quyết định**

Thuật toán được sử dụng cho cả dự đoán hồi quy và phân loại dữ liệu thành các mục con dựa trên các thuộc tính của dữ liệu. Decision trees sử dụng một chuỗi phân nhánh của các quyết định được liên kết có thể biểu diễn bằng sơ đồ cây. Decision Tree chia dữ liệu dựa vào các câu hỏi đơn giản về các đặc trưng, và mỗi nút lá của cây đại diện cho một lớp hoặc giá trị dự đoán. Các Decision Tree thường được sử dụng để phân loại (classification) hoặc hồi quy (regression) tùy thuộc vào bài toán cụ thể. Hình 2.1 minh họa một cây cấu quyết định có cấu trúc đơn giản.



Hình 2.1: *Kiến trúc của một Decision Tree đơn giản.*

Ứng dụng: Phân loại khách hàng, ra quyết định quản lý, dự đoán kết quả.

Đặc điểm: Dễ hiểu và giải thích, có thể xử lý dữ liệu thiếu, dễ bị quá khớp.

4. **Random Forest**

   Random Forest là một tập hợp của nhiều cây quyết định được huấn luyện trên các mẫu con khác nhau của dữ liệu và kết hợp lại để cải thiện độ chính xác và độ ổn định. Kết quả cuối cùng là trung bình hoặc đa số phiếu của các cây.

   Ứng dụng: Phân loại và hồi quy, phát hiện gian lận, chẩn đoán y tế.

   Đặc điểm: Kháng quá khớp, hiệu quả cao, dễ triển khai.

2.2.2. Deep Learning

2.2.2.1. Khái niệm

Deep learning, một phần của lĩnh vực Trí tuệ Nhân tạo (AI), tập trung vào việc sử dụng mạng nơ-ron nhân tạo để thực hiện các tác vụ phức tạp như phân tích, xử lý dữ liệu và mô phỏng hành vi. Công nghệ này cố gắng mô phỏng cách mà não của con người xử lý thông tin bằng cách sử dụng các lớp nơ-ron để tự động học và hiểu các mẫu từ dữ liệu.

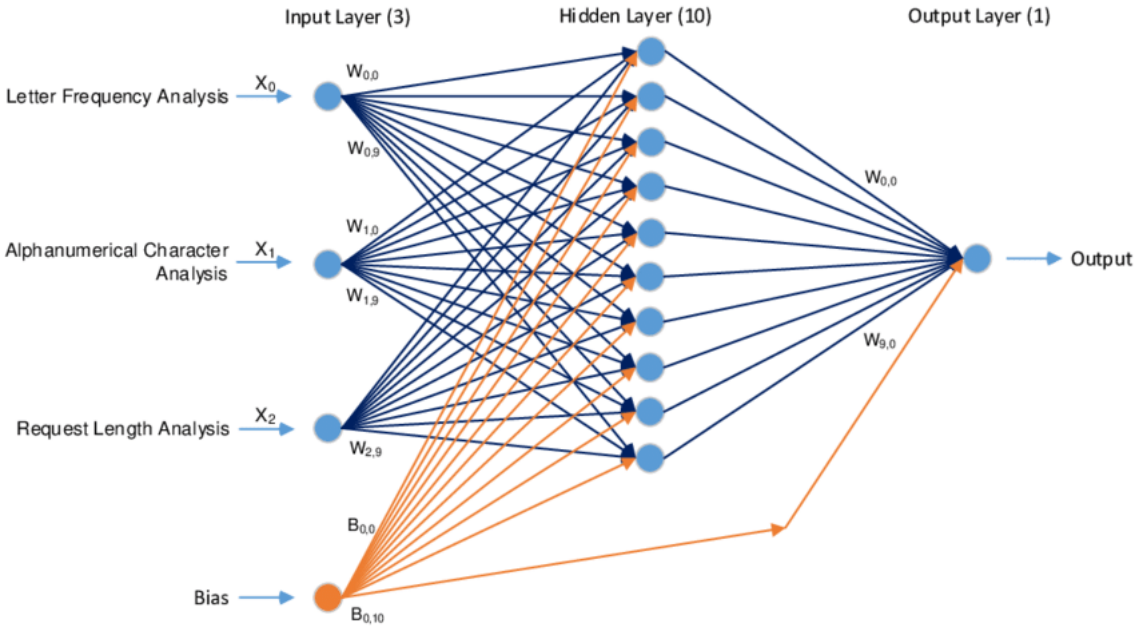
2.2.2.2. Phân loại

Cũng như Machine Learning đã nêu ở phần trước, thì ở Deep Learning cũng có thể được chia làm 3 loại là **Giám sát**, **Bán giám sát** hoặc **Không giám sát**.

2.2.2.3. Các thuật toán thường được sử dụng

1. **Deep Neural Networks - DNNs**

   DNNs là một loại mạng nơ-ron sâu có nhiều lớp ẩn giữa lớp đầu vào và lớp đầu ra. Các lớp ẩn này cho phép mô hình học các biểu diễn phức tạp của dữ liệu và thực hiện các tác vụ phức tạp như phân loại và dự đoán. Hình 2.2 mô tả một kiến trúc DNN đơn giản bao gồm 4 Input (đầu vào) và có 1 Output(đầu ra).



Hình 2.2: *Kiến trúc của một mô hình Neural Networks .*

Đặc điểm: DNNs có khả năng học các đặc trưng cấp cao từ dữ liệu đầu vào thông qua việc tự động học các biểu diễn tốt nhất. Điều này giúp DNNs đạt được hiệu suất tốt trong nhiều ứng dụng và tập dữ liệu khác nhau.

Ứng dụng: DNNs được sử dụng trong nhiều lĩnh vực như nhận dạng hình ảnh, nhận dạng giọng nói, xử lý ngôn ngữ tự nhiên, và nhiều bài toán dự đoán khác trong lĩnh vực dữ liệu có cấu trúc.

2. **Long Short-Term Memory - LSTM**

LSTM là một dạng đặc biệt của mạng nơ-ron hồi tiếp (RNN), được thiết kế để xử lý dữ liệu chuỗi. LSTM giải quyết vấn đề biến mất và bùng nổ đạo hữu trong RNNs bằng cách sử dụng các cổng quan trọng như cổng quên và cổng đầu vào.

Ứng dụng: LSTM có khả năng lưu trữ thông tin trong một khoảng thời gian dài, giúp nó hiệu quả trong việc mô hình hóa các chuỗi dữ liệu dài và phức tạp.

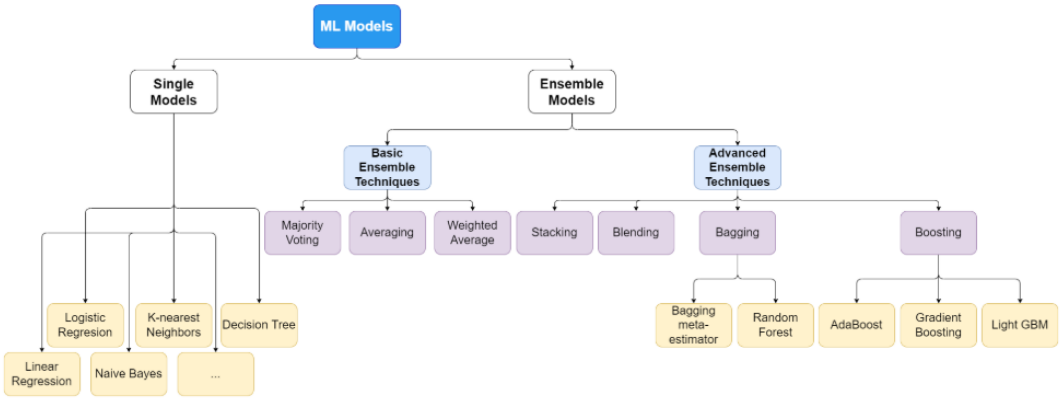
Đặc điểm: LSTM được sử dụng rộng rãi trong các ứng dụng như dự đoán

chuỗi thời gian như dự báo thị trường tài chính, dịch máy, sinh văn bản tự động và nhiều ứng dụng khác liên quan đến dữ liệu chuỗi.

2.3. Ensemble Learning

Ensemble learning là một phương pháp trong máy học, nơi mà nhiều mô hình dự đoán được kết hợp lại với nhau để tạo ra một dự đoán cuối cùng. Ý tưởng chính của phương pháp là tận dụng lợi ích từ các mô hình khác nhau bằng cách học theo cách thức của chúng. Điều này có thể là giải pháp cho các trường hợp mô hình yếu hoặc kết quả không nhất quán thu được từ nhiều mô hình. Do đó, việc kết hợp chúng một cách phù hợp có thể mang lại hiệu suất vượt trội so với việc sử dụng một mô hình đơn lẻ. Một dạng đơn giản của Ensemble Learning là kết hợp các kết quả bằng cách bỏ phiếu đa số (Votting).

Có nhiều phương pháp để phân loại các phương pháp Ensemble Learning thành nhiều loại khác nhau bằng các tiêu chí khác nhau. Trong phạm vi khóa luận tốt nghiệp này, chúng tôi tham khảo cách phân loại như được mô tả trong Hình 2.3 [11] , nơi các phương pháp hợp chúng được phân loại dựa trên cơ chế kết hợp và độ phức tạp. Điều này dẫn đến hai nhóm mô hình đơn lẻ (singe) và mô hình tổng hợp (Ensemble).



Hình 2.3: *Mô hình phân nhóm Ensemble Learning.*

Các thuật toán học đơn lẻ thường có nhược điểm riêng, và để khắc phục

những điểm yếu đó, chúng ta đã phát triển Ensemble Learning. Đầu tiên, có hai loại Ensemble Learning: đồng bộ và bất đồng bộ. Ensemble Learning đồng bộ sử dụng các mô hình học cùng loại, trong khi Ensemble Learning không đồng bộ sử dụng các thuật toán học khác nhau.

Ba lý do chính đằng sau việc sử dụng Ensemble Learning là thống kê, tính toán và biểu diễn. Khi không gian giả thuyết lớn hơn dữ liệu huấn luyện, gây ra vấn đề thống kê. Điều này có thể dẫn đến sự không nhất quán trong kết quả dự đoán. Đồng thời, đôi khi thuật toán có thể kẹt trong cực trị cục bộ, dẫn đến mô hình kém chất lượng. Trong Machine Learning, việc tìm ra hàm đúng cho không gian giả thuyết cũng là một thách thức.

Ensemble Learning giúp giải quyết các vấn đề này bằng cách kết hợp các giả thuyết từ nhiều mô hình, giảm thiểu sự không nhất quán và tăng độ chính xác. Nó cũng giúp tránh được cực trị cục bộ bằng cách khởi đầu từ nhiều điểm khác nhau. Điều này có thể dẫn đến mô hình gần với hàm chưa biết chính xác hơn so với việc sử dụng các thuật toán đơn lẻ.

Cuối cùng, Ensemble Learning cũng linh hoạt với lượng dữ liệu, hoạt động hiệu quả cả khi có ít dữ liệu hoặc quá nhiều dữ liệu. Điều này giúp cải thiện hiệu suất và độ chính xác của các mô hình máy học trong nhiều tình huống.

Các phương pháp chính được sử dụng trong ensemble learning:

1. **Đa dạng hóa dữ liệu đầu vào:** Mỗi mô hình được huấn luyện với các tập dữ liệu khác nhau để nhận biết các khía cạnh khác nhau của dữ liệu.

2. **Thay đổi thuật toán hoặc siêu tham số:** Thay đổi cách mỗi mô hình hoạt động để tạo ra sự đa dạng. Điều này có thể là việc sử dụng các thuật toán khác nhau hoặc điều chỉnh các siêu tham số khác nhau cho mỗi mô hình.

3. **Phân vùng dữ liệu:** Chia dữ liệu thành các phần nhỏ để huấn luyện các mô hình trên các phần khác nhau của dữ liệu.

4. **Kết hợp đầu ra:** Sử dụng các kỹ thuật như bỏ phiếu, hợp đồng, hoặc mã sửa lỗi đầu ra để kết hợp các dự đoán từ các mô hình thành một dự đoán cuối cùng.

5. **Ensemble Learning:** Kết hợp các mô hình lại với nhau, ví dụ như sử dụng RandomForest, một phương pháp kết hợp cả việc chọn đặc trưng và mẫu.

6. **Khung phụ thuộc và khung độc lập:** Trong khung phụ thuộc, thông tin từ các mô hình trước đó ảnh hưởng đến việc huấn luyện mô hình sau này. Trong khi đó, trong khung độc lập, mỗi mô hình được huấn luyện độc lập với các mô hình khác.

Ensemble learning giúp cải thiện hiệu suất dự đoán và đồng thời tăng tính đa dạng của mô hình, giúp giảm thiểu overfitting và cải thiện khả năng tổng quát hóa của mô hình.

2.4. Multimodal Learning

Học sâu đa phương thức (Multimodal Learning) là một lĩnh vực nghiên cứu trong trí tuệ nhân tạo (AI) và học máy, tập trung vào việc xử lý và phân tích dữ liệu từ nhiều nguồn hoặc phương thức khác nhau. Những phương thức này có thể bao gồm văn bản, hình ảnh, âm thanh, video. Mục tiêu chính của học sâu đa phương thức là khai thác sự phong phú và đa dạng của dữ liệu từ các nguồn khác nhau để cải thiện hiệu suất của các hệ thống AI và học máy trong nhiều ứng dụng thực tế.

Trong thế giới thực, con người thường sử dụng nhiều giác quan cùng một lúc để hiểu và tương tác với môi trường xung quanh. Ví dụ, khi xem một bộ phim, chúng ta không chỉ dựa vào hình ảnh mà còn dựa vào âm thanh và ngữ cảnh để hiểu cốt truyện. Tương tự, các hệ thống AI có khả năng xử lý nhiều phương thức dữ liệu cùng lúc có thể đạt được hiệu suất tốt hơn trong nhiều nhiệm vụ phức tạp.

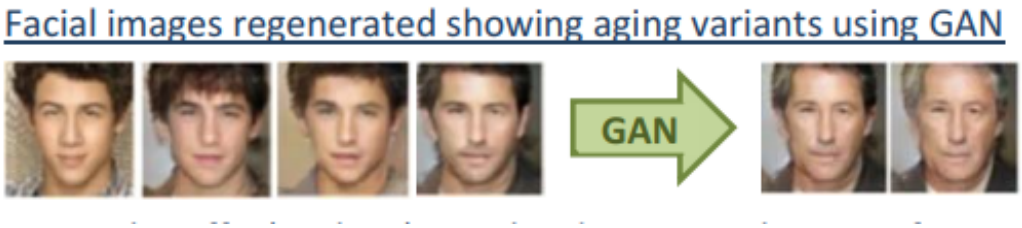
Động lực chính đằng sau học sâu đa phương thức là nhu cầu tạo ra các hệ thống AI thông minh hơn, có thể:

- **Hiểu biết sâu sắc hơn:** Kết hợp thông tin từ nhiều nguồn để có được cái nhìn toàn diện và chính xác hơn.
- **Cải thiện hiệu suất:** Tận dụng thông tin bổ sung để nâng cao hiệu suất trong các nhiệm vụ cụ thể như nhận diện đối tượng, phân loại, và dự đoán.
- **Tăng cường khả năng tương tác:** Phát triển các hệ thống có thể tương tác tự nhiên hơn với con người bằng cách hiểu và phản hồi dựa trên nhiều phương thức dữ liệu.

2.5. Generative Adversarial Networks

2.5.1. Khái niệm

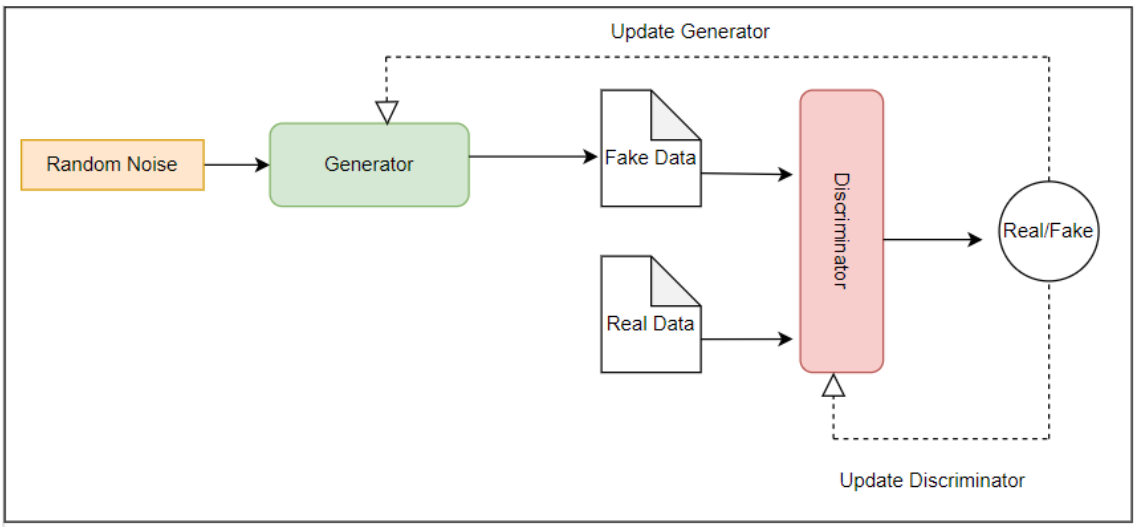
Mô hình mạng sinh đối kháng (GAN - Generative Adversarial Networks) là các mô hình học sâu tiên tiến được thiết kế để tạo ra dữ liệu mới từ dữ liệu hiện có. GAN có khả năng nắm bắt và học các phân phối dữ liệu phức tạp trong một lĩnh vực ứng dụng cụ thể, sau đó tái tạo các biến thể tương tự nhưng có những khác biệt tinh tế trong dữ liệu. Ví dụ, các GANs dành cho hình ảnh có thể học các mẫu và mối quan hệ phức tạp giữa các điểm ảnh để tạo ra các biến thể tương tự. Một ví dụ trong [3] sử dụng GAN để mô phỏng quá trình lão hóa khuôn mặt trong các bức ảnh, ứng dụng này giúp dự đoán diện mạo của một người khi già đi dựa trên hình ảnh hiện tại của họ. Cụ thể, hình 2.4 thể hiện kết quả sử dụng GAN để tạo ra hình ảnh lão hóa khuôn mặt người từ hình ảnh hiện tại của họ.



Hình 2.4: *Sử dụng GAN để mô phỏng lại quá trình lão hóa của con người.*

2.5.2. Cấu trúc mô hình GAN

Hình 2.5 thể hiện cấu trúc GAN cơ bản, bao gồm hai mạng nơ-ron chính: một mô hình sinh (generator) và một mô hình phân biệt (discriminator).



Hình 2.5: *Kiến trúc GAN cơ bản.*

- **Generator:** Mô hình Generator nhận nhiễu làm đầu vào, cố gắng bắt chước dữ liệu gốc để tạo ra dữ liệu mới, nhằm đánh lừa Discriminator. Ban đầu, dữ liệu tạo ra có thể không chính xác, nhưng theo thời gian, Generator học cách cải thiện để tạo ra dữ liệu ngày càng giống với dữ liệu gốc hơn.
- **Discriminator:** Mô hình Discriminator có vai trò quan trọng là phân biệt giữa dữ liệu thực và dữ liệu được tạo ra bởi mô hình Generator. Qua thời gian, Discriminator học cách phân biệt dữ liệu thật từ dữ liệu nhân tạo.

Quá trình này đồng thời giúp Generator cải thiện khả năng sinh ra các mẫu dữ liệu giả mạo giống như thật để đánh lừa Discriminator. Điều này thúc đẩy cả hai mô hình cùng tiến bộ, với Discriminator cố gắng ngày càng tốt hơn trong việc phân biệt dữ liệu thật và dữ liệu giả., trong khi Generator có thể các mẫu dữ liệu ngày càng chất lượng hơn.

Quá trình này tiếp tục cho đến khi Generator có thể tạo ra dữ liệu gần như không thể phân biệt được với dữ liệu thật đối với Discriminator. Kết quả là một mô hình có thể tạo ra các mẫu dữ liệu mới rất thuyết phục, có thể được sử dụng trong nhiều ứng dụng khác nhau, chẳng hạn như tạo hình ảnh, âm thanh, văn bản, và nhiều loại dữ liệu khác.

2.6. Adversarial Attack

Ở phần này, chúng tôi sẽ đề cập tới khái niệm, các dạng tấn công và các thuật toán tạo ra **"Adversarial examples"** (mẫu đối kháng).

2.6.1. Khái niệm

Adversarial Attack là một kỹ thuật trong lĩnh vực Machine Learning và Deep Learning, được sử dụng để tạo ra các dữ liệu đầu vào nhằm làm cho mô hình phân loại hoặc nhận dạng sai lầm. Mục tiêu chính của Adversarial Attack là thay đổi một chút đầu vào ban đầu sao cho con người vẫn có thể nhận ra, nhưng mô hình máy học lại đưa ra kết quả khác biệt.

Cụ thể, Adversarial Attack thường bao gồm việc thêm vào hoặc điều chỉnh những nhiễu nhỏ trong dữ liệu đầu vào, gọi là "nhiễu adversarial", nhằm làm thay đổi kết quả phân loại hoặc nhận dạng của mô hình. Mặc dù các thay đổi này có thể không rõ ràng đối với con người, chúng có thể đủ để làm cho mô hình máy học đưa ra dự đoán sai lầm từ đó làm sai lệch đi kết quả dự đoán của mô hình.

2.6.2. Các dạng tấn công

Các dạng tấn công Adversarial Attack để tạo ra **Adversarial examples** cơ bản được chia làm 2 loại là:

1. **White-box Attack**

   - Trong tấn công white-box, kẻ tấn công có quyền truy cập đầy đủ thông tin về mô hình, bao gồm cấu trúc của mô hình, các tham số và gradient.
   - Với những thông tin chi tiết này, kẻ tấn công có thể tính toán chính xác các thay đổi cần thiết trên dữ liệu đầu vào để đánh lừa mô hình.
   - White-box attack thường được coi là dạng tấn công mạnh hơn và khó phòng chống hơn.
   - **Ưu điểm:** Có thể tính toán chính xác các thay đổi cần thiết trên dữ liệu đầu vào để tạo ra mẫu đối kháng với mức độ đánh lừa cao.
   - **Nhược điểm:** Không phải lúc nào cũng có đầy đủ các thông tin của mô hình, khó áp dụng được trong thực tế vì mô hình không được công khai.

2. **Black-box Attack**

   - Trong tấn công black-box, kẻ tấn công chỉ có quyền truy cập vào mô hình như một "hộp đen" - chỉ có thể quan sát đầu vào và đầu ra của mô hình.
   - Kẻ tấn công phải sử dụng các kỹ thuật thử nghiệm và quan sát để tìm ra những thay đổi cần thiết trên dữ liệu đầu vào.
   - Black-box attack thường yêu cầu nhiều tài nguyên tính toán hơn so với white-box, nhưng vẫn có thể gây được ảnh hưởng lên mô hình.
   - **Ưu điểm:** Không cần quyền truy cập vào mô hình, chỉ cần biết được đầu vào và đầu ra của mô hình để có thể tạo ra các mẫu đối kháng.

- **Nhược điểm:** Yêu cầu nhiều tài nguyên hơn (CPU hoặc GPU) để có thể tạo ra được mẫu đối kháng và không thể tạo ra được các mẫu đối kháng mạnh mẽ như mô hình Attack WhiteBox được.

2.6.3. Các thuật toán tấn công để tạo Adversarial examples

Ở phần này, chúng tôi sẽ đề cập tới các thuật toán để tạo ra dữ liệu đối kháng được sử dụng trong Khóa luận tốt nghiệp.

2.6.3.1. Fast Gradient Sign Method - FGSM

FGSM (Fast Gradient Sign Method) là một trong những kỹ thuật đơn giản nhưng mạnh mẽ nhất trong Adversarial Attack, được đề xuất bởi Ian Goodfellow và các đồng nghiệp vào năm 2014 [6].

Là một thuật toán của mô hình whitebox tạo mẫu đối kháng bằng cách tận dụng gradient của mô hình mạng nơ-ron để tạo ra các nhiễu adversarial. Điểm đặc biệt của FGSM là sự đơn giản và hiệu quả trong việc tạo ra các nhiễu này chỉ bằng một lượng tính toán nhỏ.

Thuật toán của FGSM: tạo ra một dữ liệu mới từ dữ liệu ban đầu cộng thêm hyper-parameter(epsilon) nhân với đạo hàm theo input dữ liệu của hàm loss của model. Algorithm của FGSM có thể được biểu diễn như sau:

$$\textbf{Adversarial example} = \mathbf{x} + \epsilon \cdot \textbf{sign}(\nabla_{\mathbf{x}} J(\theta, \mathbf{x}, y)) \tag{2.3}$$

Trong đó:

- $\mathbf{x}$ là điểm dữ liệu đầu vào ban đầu.
- ϵ là hệ số điều chỉnh, xác định độ lớn của nhiễu adversarial.
- $J(\theta, \mathbf{x}, y)$ là hàm mất mát của mô hình với tham số θ, dữ liệu đầu vào $\mathbf{x}$ và nhãn y.
- $\nabla_{\mathbf{x}} J(\theta, \mathbf{x}, y)$ là gradient của hàm mất mát theo dữ liệu đầu vào $\mathbf{x}$.

- sign(·) là hàm dấu, trả về dấu của đạo hàm.

2.6.3.2. Zeroth Order Optimization - ZOO

Zeroth Order Optimization - ZOO là một dạng tấn công Blackbox được đề xuất bởi P.-Y. Chen và các đồng nghiệp [7]. Nó là một dạng tấn công tạo ra mẫu đối kháng khi mà không thể truy cập trực tiếp vào mô hình để lấy gradients. Không giống như các cuộc tấn công dựa trên gradients, mà dùng gradients để tạo ra các mẫu đối kháng, ZOO tạo ra các mẫu đối kháng như vậy bằng cách tối ưu hóa một hàm điều chuẩn. Hàm này nhằm mục đích giảm thiểu sự khác biệt giữa mẫu đối kháng được tạo ra $\hat{x}^p$ và đầu vào gốc x theo một chuẩn p được chọn, cùng với một thuật ngữ điều chuẩn phụ thuộc vào đầu ra của mô hình.

Regularization function có thể được biểu diễn như sau:

$$\min_{\hat{x}_p} \|\hat{x}_p - x\|_p + c \cdot g(\hat{x}_p) \tag{2.4}$$

Với các ràng buộc, trong đó $\|\cdot\|_p$ áp dụng sự tương đồng giữa các mẫu đối kháng được tạo ra và mẫu bình thường bằng cách sử dụng chuẩn đã chọn p, c là tham số điều chuẩn, và $g(\hat{x}_p)$ là loss function của cuộc tấn công Blackbox, phụ thuộc vào đầu ra của mô hình. Loss function được định nghĩa theo công thức sau[7]:

$$g(\hat{x}_p) = \max\left(\max_{i \neq t}\left(\log[f(x)_i]\right) - \log[f(x)_t]\ , -c_f\right) \tag{2.5}$$

Trong đó, $f(x)$ biểu thị logarit của xác suất đầu ra của mô hình cho lớp i cho trước đầu vào x, t đại diện cho nhãn của lớp mục tiêu (nhãn bình thường), và c_f điều khiển độ tin cậy của cuộc tấn công.

Để ước lượng thông tin gradients, ZOO sử dụng tỉ số khác biệt đối xứng để tối ưu hóa mô hình và xấp xỉ gradients trong một môi trường hộp đen (BB) mà

không cần biết bất kỳ thông tin nào về kiến trúc và các tham số của mô hình. Gradients này được mô tả theo công thức như sau[7]:

$$\frac{\partial f(x)}{\partial x_i} \approx \frac{f(x + s_{si}) - f(x - s_{si})}{2s_s} \tag{2.6}$$

Ở đây, đạo hàm riêng này xấp xỉ $f(x)$ theo x_i bằng cách đo sự khác biệt trong đầu ra của mô hình cho các đầu vào bị ánh xạ $x + s_{si}$ và $x - s_{si}$. Kích thước bước s_s là một hằng số nhỏ được sử dụng trong xấp xỉ.

2.6.3.3. DeepFool

DeepFool [9] là một dạng tấn công Whitebox,tạo ra mẫu đối kháng bằng cách dựa trên ý tưởng tìm ra sự biến đổi nhỏ nhất đối với một đầu vào mà sẽ khiến mô hình phân loại (classifier) phân loại sai. Cuộc tấn công này hoạt động bằng cách sử dụng một đầu vào ban đầu, sau đó lặp lại tính toán đạo hàm của đầu ra của mô hình theo đối với đầu vào. Tại mỗi vòng lặp, cuộc tấn công thêm một độ biến đổi nhỏ vào đầu vào theo hướng của đạo hàm mà sẽ tăng đầu ra của mô hình cho lớp ít có khả năng nhất. Quá trình này được lặp lại cho đến khi đầu ra của mô hình cho lớp ít có khả năng vượt quá lớp đúng. Quá trình tạo ra các mẫu đối kháng được hình thành như sau:

$$\hat{x}_p \leftarrow x; \tag{2.7}$$

$$\overline{y} \leftarrow 1 \text{ to } T; \tag{2.8}$$

$$\text{cf}(\hat{x}_{p,\overline{y}}) \leftarrow \text{cf}(\hat{x}_p) \tag{2.9}$$

$$\mathbf{w}_{\overline{y}} \leftarrow \nabla \text{cf}_y(\hat{x}_{\hat{p}}, \overline{y}) - \nabla \text{cf}_{\overline{y}}(\hat{x}_p, \overline{y}); \tag{2.10}$$

$$\Delta x_{\overline{y}} \leftarrow \epsilon \frac{\mathbf{w}_{\overline{y}}}{\|\mathbf{w}_{\overline{y}}\|_2}; \qquad \hat{x}_{p,\overline{y}+1} \leftarrow (\hat{x}_{\hat{p},\overline{y}} + \Delta x_{\overline{y}}). \tag{2.11}$$

DeepFool [9] nhận đầu vào là một luồng traffic x cần biến đổi, một bộ phân loại cf và một tham số biến đổi ϵ. Cuộc tấn công sẽ chạy trong T số lần lặp. Nó bắt đầu bằng cách khởi tạo các mẫu đối kháng $x^{\hat{p}}$ để giống với mẫu dữ liệu ban

đầu. Một biến đổi tối thiểu là cần thiết để di chuyển đầu vào qua ranh giới quyết định của mạng nơ-ron. Tại mỗi vòng lặp, các đạo hàm đầu ra của mạng nơ-ron đối với mẫu ban đầu được tính toán, $w_{\overline{y}}$ là hướng của ranh giới quyết định giữa lớp đúng y và lớp $\overline{y}$. Nó được thu được bằng cách tuyến tính hóa ranh giới quyết định. Hướng của biến đổi tối thiểu để thay đổi phân loại của mẫu đầu vào sau đó được tính toán. DeepFool sau đó kiểm tra xem mẫu đầu vào đã được phân loại thành công là không độc hại nếu cuộc tấn công là mục tiêu. Quá trình này được tiếp tục cho đến khi mẫu đó được phân loại là không độc hại

2.6.3.4. Carlini and Wagner - C&W

Carlini và Wagner (C&W) [2] là một dạng tấn công thuộc dạng Whitebox. Việc tạo ra các mẫu đối kháng được công thức hóa như việc tối thiểu hóa khoảng cách giữa mẫu đối kháng được tạo ra và mẫu đầu vào gốc, đồng thời cũng làm tăng độ tin cậy của bộ phân loại vào lớp mục tiêu. Tấn công C&W tìm thấy một mẫu đối kháng ban đầu, đó là một đầu vào gần với mẫu đầu vào gốc. Nó tiếp tục cải tiến nó cho đến khi nó tạo ra một mẫu đối kháng dựa trên các hàm tối ưu được mô tả như sau:

$$\min_{\hat{x}_p} \|\hat{x}_p - x\|_2 + c \cdot f(x,t) \tag{2.12}$$

$$f(x,t) = \max_{i \neq t} [Z(x)]_i - [Z(x)]_t \tag{2.13}$$

Trong đó, $\hat{x}_p$ là đầu vào bị làm nhiễu (có thể là mẫu đối kháng được tạo ra sau này), x là mẫu đầu vào ban đầu. $\|\cdot\|_2$ biểu thị norm Euclide hoặc norm L_2 để đo khoảng cách giữa $\hat{x}_p$ và x, c là một hằng số vô hướng điều khiển sự đánh đổi giữa kích thước nhiễu và độ tin cậy của cuộc tấn công, $f(x,t)$ là hàm mục tiêu bao gồm cả khoảng cách ($\|\hat{x}_p - x\|_2$) và độ tin cậy ($c \cdot f(x,t)$), và $Z(x)_i$ là điểm tin cậy của lớp logit. $Z(x)_i$ được tính bằng cách sử dụng quy tắc phân loại SoftMax để dự đoán nhãn lớp của một đầu vào. Các đầu vào bị nhiễu được

truyền qua lớp logit và hàm SoftMax để tối đa hóa sự khác biệt giữa lớp thực sự và lớp mục tiêu.

2.7. Transferability và Defence

Ở phần này, chúng tôi sẽ đề cập tới **Transferability** (chuyển giao) và **Defence** (phòng thủ) các mẫu đối kháng.

2.7.1. Transfersibility

Trong ngữ cảnh của công nghệ thông tin nói chung và trong các mô hình học máy nói riêng, "Transferability" (khả năng chuyển giao) là đề cập đến khả năng của các mô hình hoặc các tấn công đối kháng (adversarial attacks) được áp dụng từ một mô hình hoặc môi trường sang một mô hình hoặc môi trường khác mà không cần phải thay đổi hoặc không cần phải thay đổi quá nhiều.

Có thể phân loại Transfersibility thành 2 loại là : Transfersibility trên cùng một mô hình(nội bộ) và Transfersibility trên khác mô hình (lẫn nhau) . Chúng tôi sẽ nói rõ vấn đề này hơn ở chương 3.

Ví dụ, trong trường hợp của **adversarial attacks**, Transferability đánh giá xem một tập hợp các mẫu tấn công, được tạo ra cho một mô hình máy học cụ thể, có thể hoạt động hiệu quả trên một mô hình khác hoặc trong một bối cảnh khác mà không cần phải điều chỉnh lại các tham số của tấn công. Transferability là một yếu tố quan trọng trong việc đánh giá tính đáng tin cậy và ổn định của các tấn công đối kháng.

2.7.2. Defence

Ở phần này, chúng tôi sẽ đề cập tới phương pháp phòng thủ đó là: **Adversarial Training**

Adversarial Training là một kỹ thuật phòng thủ trong học máy nhằm mục

đích làm cho các mô hình trở nên mạnh mẽ và nhạy cảm hơn trước các cuộc tấn công đối kháng. Các cuộc tấn công kháng đầu là những nỗ lực cố gắng đánh lừa các mô hình học máy phân loại sai bằng cách thực hiện những thay đổi nhỏ, không thể nhận biết được trên dữ liệu đầu vào. Bằng cách huấn luyện kết hợp với các mẫu đối kháng, tức là các phiên bản đã được sửa đổi của dữ liệu đầu vào thì các mô hình có thể học cách phát hiện và chống lại hiệu quả hơn những cuộc tấn công như vậy

Cơ bản, quy trình Adversarial Training có thể mô tả đơn giản như sau:

1. Huấn luyện một mô hình cơ bản trên dữ liệu thông thường.
2. Tạo ra các mẫu đối kháng bằng cách thêm nhiễu hoặc điều chỉnh các mẫu đầu vào nhằm gây sai lệch cho mô hình (các thuật toán đã được liệt kê ở chương 2.6.3).
3. Thêm các mẫu đối kháng vào tập huấn luyện và tiếp tục huấn luyện mô hình để nó có thể phát hiện và xử lý tốt hơn những mẫu đối kháng sau này.

2.8. Các công trình nghiên cứu liên quan

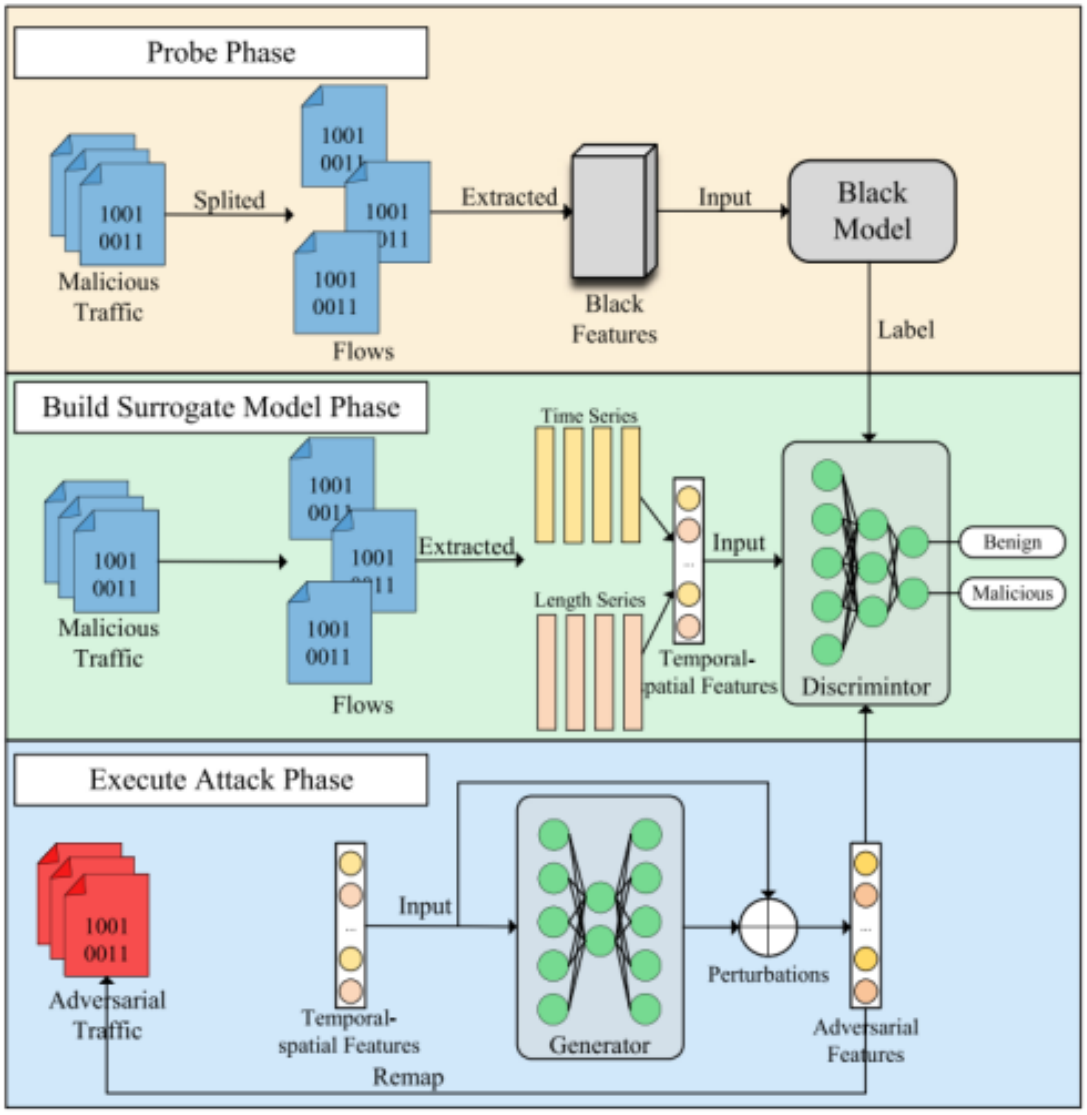
Trong thời đại Internet bùng nổ như hiện nay, việc bảo vệ các thiết bị trong cơ sở hạ tầng công nghệ thông tin trước những hành vi độc hại từ kẻ tấn công trở nên cấp thiết hơn bao giờ hết. Việc triển khai Hệ thống Phát hiện Xâm nhập Mạng (NIDS) không chỉ là một lựa chọn hợp lý mà còn là biện pháp bắt buộc để đảm bảo an ninh và bảo vệ tài nguyên mạng của tổ chức. Một trong những ưu điểm nổi bật của NIDS là khả năng phát hiện các hoạt động bất thường trong mạng. NIDS có thể giám sát và phân tích các gói tin dữ liệu truyền qua mạng để xác định các dấu hiệu của tấn công mạng hoặc các hành vi bất thường. Khả năng này cho phép tổ chức kịp thời phát hiện và xử lý các mối đe dọa nguy hiểm ngay từ khi chúng xuất hiện, giúp giảm thiểu thiệt hại và thời gian ảnh hưởng.

Các kỹ thuật Học máy (ML), đặc biệt là Học sâu (DL), ngày càng được áp

dụng nhiều trong các hệ thống NIDS dựa trên sự bất thường. Tuy nhiên, các nghiên cứu đã chỉ ra rằng ML/DL rất dễ bị tấn công đối kháng, đặc biệt là trong những hệ thống yêu cầu bảo mật cao như vậy.

Công trình nghiên cứu của tác giả Dongqi Han và các cộng sự [4] đã chỉ ra rằng, các kỹ thuật học máy và học sâu ngày càng được sử dụng trong NIDS dựa trên sự bất thường, tuy nhiên, chúng dễ bị tấn công đối kháng, tức là những thay đổi nhỏ trong đầu vào có thể gây ra những thay đổi lớn trong đầu ra của mô hình. Đồng thời tác giả đã phát triển ra công cụ Traffic Manipulator để tạo ra lưu lượng độc hại trong không gian lưu lượng mạng với ngữ cảnh thực tế. Tác giả tập trung vào việc tạo ra lưu lượng đối kháng bằng cách sử dụng mô hình Generative Adversarial Network (GAN) để biến đổi lưu lượng độc hại ban đầu trong không gian lưu lượng mạng thành lưu lượng đối kháng để qua mặt NIDS. Kết quả cho thấy tỉ lệ trốn tránh của lưu lượng đối kháng với NIDS hơn 97% đối với KitNET.

Trong công trình nghiên cứu của tác giả Peishuai Sun et al [8] cũng đã nhận định rằng các NIDS có sử dụng ML/DL đều dễ bị tấn công đối kháng, trốn tránh cơ chế phát hiện. Hầu hết các nghiên cứu hiện có về tấn công đối kháng chống lại phát hiện lưu lượng dựa trên ML hoặc giả định mức kiến thức trước không thực tế hoặc tạo ra các mẫu đối kháng không thực tiễn, không tuân thủ các quy tắc về giao thức, và không duy trì được tính độc hại. Tác giả đề xuất một framework tấn công đối kháng mới tên là GPMT (Generating Practical Malicious Traffic), giải quyết các hạn chế của các công trình trước đó. Hình 2.6 giới thiệu kiến trúc của framework tấn công đối kháng do tác giả đề xuất.



Hình 2.6: *Kiến trúc của GPMT*

Framework GPMT bao gồm ba giai đoạn chính:

- **Probe Phase (giai đoạn thăm dò mục tiêu):** gửi lưu lượng độc hại đến mô hình phát hiện để thu thập nhãn, về cơ bản là thăm dò phản hồi của mô hình.
- **Build Surrogate Model Phase (xây dựng mô hình thay thế):** dựa trên dữ liệu thu thập được, một mô hình thay thế cục bộ được huấn luyện để mô phỏng hành vi của mô hình mục tiêu.

- **Execute Attack Phase (thực hiện tấn công):** sử dụng một mạng sinh đối kháng Wasserstein (WGAN) một biến thể của GANGAN với một hàm mất mát mới, framework này tạo ra lưu lượng đối kháng có thể né tránh mô hình thay thế và do đó cả hệ thống phát hiện ban đầu.

Trong công trình nghiên cứu của tác giả Haitao He và các cộng sự [5], tác giả đề xuất ra giải pháp để tận dụng được tối đa thông tin có trong lưu lượng mạng, là việc áp dụng kỹ thuật Multimodal Learning, trích xuất các tính năng cấp độ khác nhau từ mạng kết nối, thay vì vector đặc trưng dài được sử dụng trong các phương pháp truyền thống, bằng các này có thể xử lý thông tin tính năng một cách riêng biệt, hiệu quả hơn. Dựa vào 3 bộ dữ liệu NSL-KDD, UNSWNB15 và CICIDS2017, tác giả đã đánh giá hiệu suất của các phương pháp đã đưa ra trong việc phát hiện các cuộc tấn công mạng hiện đại. Kết quả thử nghiệm cho thấy accuracy trung bình của phương pháp này là 94% trong binary classfication và 88% trong multi-classfication.

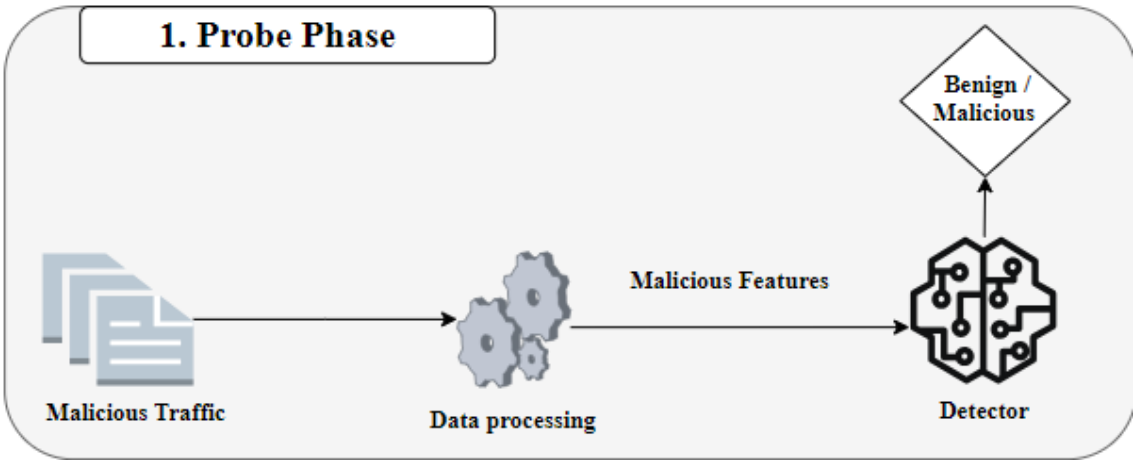
Bên cạnh đó chúng tôi còn nghiên cứu các công trình nghiên cứu liên quan tấn công tạo mẫu đối kháng và chuyển giao mẫu đối kháng khác. Trong công trình của tác giả Alhussien [1] và các cộng sự đã chỉ ra rằng các mô hình ML/DL vẫn dễ bị tấn công đối kháng. Bài báo nghiên cứu tính chuyển giao bằng cách thực hiện các cuộc tấn công đối kháng trên nhiều mô hình NIDS khác nhau, bao gồm các mô hình học sâu (deep learning) như DNN và RNN. Kết quả cho thầy rằng các cuộc tấn công đối kháng có tính chuyển giao cao giữa các mô hình tương tự, nhưng tính chuyển giao giảm đi khi tấn công các mô hình khác biệt về kiến trúc và huấn luyện. Tác giả cũng đưa ra các phương pháp phòng thủ đối kháng như là Same Signature dựa trên chữ kí số nhưng không mang lại hiệu quả cao và đề xuất sử dụng Adversarial Training. Cuối cùng, các tác giả đưa ra kết luận rằng không có một phương pháp phòng thủ duy nhất nào có thể hoàn toàn ngăn chặn được các cuộc tấn công đối kháng. Đề xuất sử dụng kết hợp nhiều phương pháp phòng thủ để xây dựng một hệ thống NIDS toàn diện và hiệu quả.

CHƯƠNG 3. PHƯƠNG PHÁP THỰC HIỆN

3.1. Tổng quan về mô hình

Để giải quyết vấn đề đánh giá khả năng mạnh mẽ của các IDS dựa trên học máy trước các mẫu đối kháng. Nhóm chúng tôi đề xuất mô hình để tạo mẫu đối kháng, đánh giá tỉ lệ phát hiện tấn công trước dữ liệu chưa biến đổi và dữ liệu sau khi thực hiện biến đổi để trốn tránh Detector, ngoài ra còn đề xuất mô hình đào tạo đối kháng để ngăn chặn tấn công đối kháng.

3.1.1. Phase 1: Probe Phase



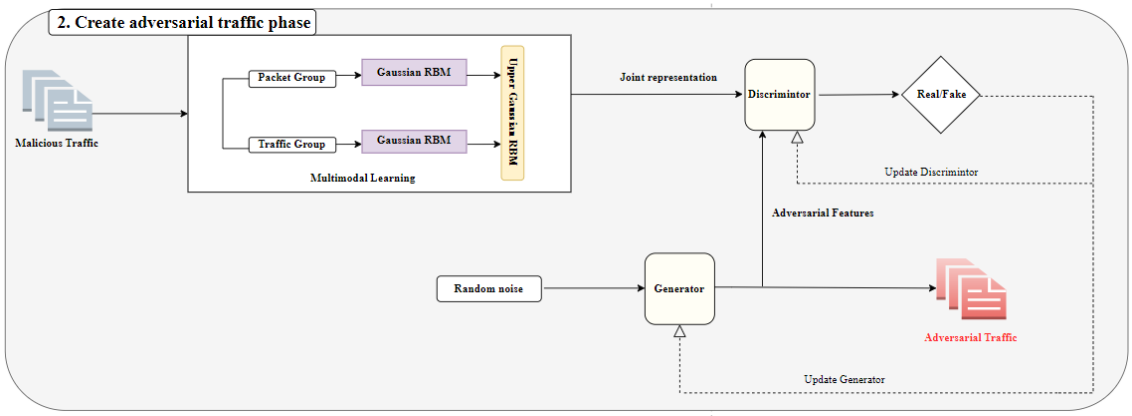
Hình 3.1: *Giai đoạn thăm dò.*

Hình 3.1 Thể hiện giai đoạn thăm dò (probe phase) , dữ liệu đầu vào là lưu lượng độc hại gốc, sau khi thực hiện quá trình xử lý dữ liệu, các đặc trưng độc hại được đưa vào Detector mục tiêu. Kết quả nhận được là các thông số đánh giá khả năng phát hiện của Detector. Giai đoạn này nhằm đánh giá hiệu suất phát hiện tấn công trước dữ liệu độc hại gốc (chưa biến đổi đối kháng) của các

Detector.

3.1.2. Phase 2: Create adversarial traffic phase

Trong phần này, sử dụng kiến trúc mạng Generator để tạo ra lưu lượng đối kháng từ đầu vào là các noise được tạo ngẫu nhiên. Hình 3.2 cho thấy quá trình sinh dữ liệu đối kháng có 3 thành phần chính:



Hình 3.2: *Giai đoạn sử dụng GAN để tạo mẫu đối kháng*

- **Multimodal Learning:** Nhằm mục đích tận dụng tối đa thông tin độc hại có trong lưu lượng độc hại ban đầu, nhóm em áp dụng Multimodal Learning để học phân phối dữ liệu từ các nhóm đặc trưng khác nhau. Trong đề tài của nhóm, dữ liệu độc hại ban đầu sẽ được chia thành nhóm là packet và traffic, sau đó các nhóm dặc trưng này sẽ được đưa vào Gaussian RBM để học phân phối dữ liệu của từng nhóm, đầu ra từ các Gaussian RBM này sẽ được đưa vào Upper Gaussian RRM để tổng hợp lại các thông tin từ các Gaussian RBM trên thành vector đại diện chung. Việc sử dụng thông tin đa phương phức được biểu diễn dưới dạng vector đại diễn chung sẽ giúp Discriminator phân biệt tốt hơn giữa lưu lượng thật và đối kháng được tạo ra. Khi Discriminator trở nên mạnh mẽ thì Generator cần phải tạo dữ liệu đối kháng phức tạp và tinh vi hơn để đánh lừa Generator, chất lượng mẫu đối kháng của Generator sẽ được cải thiện nhiều.

- **Generator:** Generator có nhiệm vụ tạo ra dữ liệu đối kháng để đánh lừa Disriminator, đầu vào là nhiễu có giá trị được tạo ngẫu nhiên. Sử dụng torch.randn(input, output) có nghĩa là tạo một tensor chứa các giá trị ngẫu nhiên được lấy từ phân phối chuẩn. Ví dụ nếu input = 64 và output = 100, thì một tensor có kích thước (64,100) chứa các giá trị ngẫu nhiên từ phân phối chuẩn được tạo ra.

- **Discriminator:** Discriminator cố gắng phân biệt giữ dữ liệu thật và dữ liệu do Generator sinh ra. Sau đó phản hồi lại cho Generator để cải thiện khả năng sinh dữ liệu, và để chính nó tự cải thiện khả năng phân biệt của mình.

Thông số thiết kế của Generator và Discriminator được thể hiện cụ thể trong bảng 3.1 (đối với Generator) và 3.2 (đối với Discriminator).

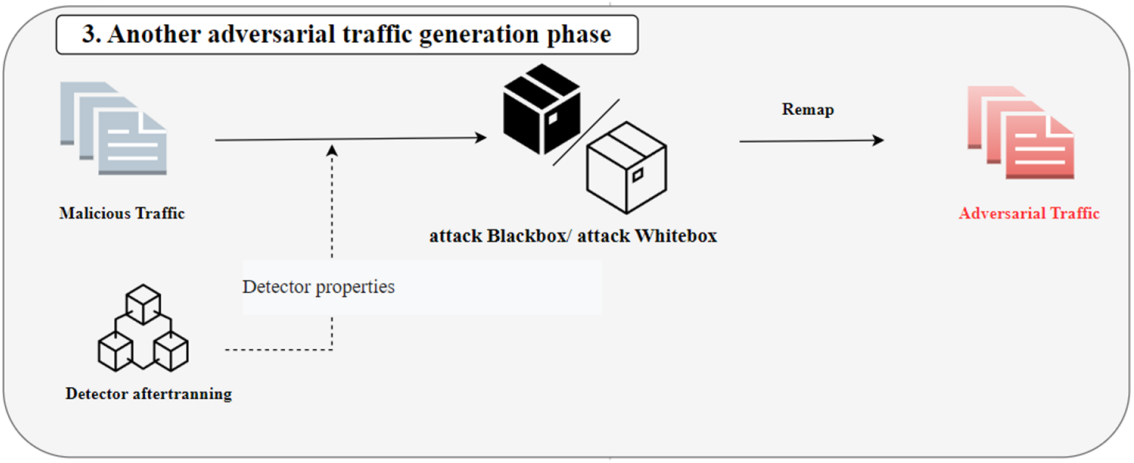
Lớp	**Chiều vào**	**Chiều ra**	**Hàm kích hoạt**
Đầu vào	*input_dim*	64	`ReLU()`
Ẩn 1	64	128	`ReLU()`
Ẩn 2	128	*output_dim*	`Tanh()`

Bảng 3.1: *Kiến trúc của lớp Generator*

Lớp	**Chiều vào**	**Chiều ra**	**Hàm kích hoạt**
Đầu vào	*input_dim*	64	`ReLU(0.2)`
Ẩn 1	6464	128	`ReLU(0.2)`
Ẩn 2	128	68	`Sigmoid()`

Bảng 3.2: *Kiến trúc của lớp Discriminator*

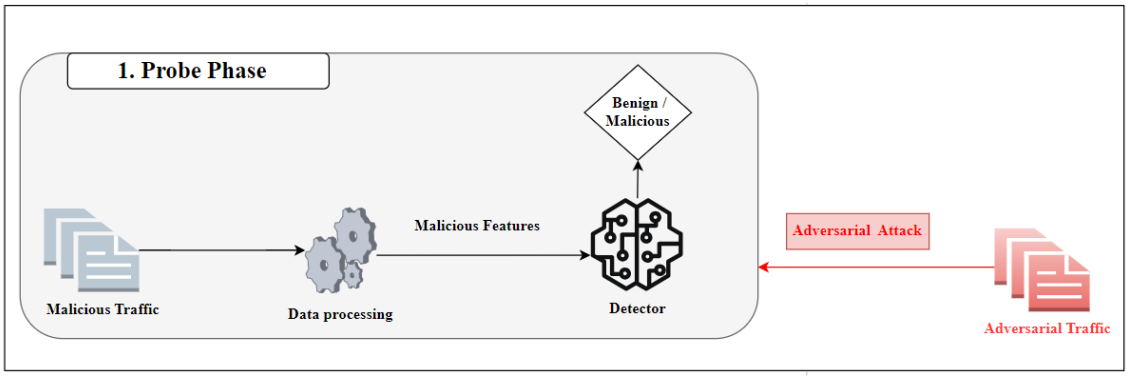
3.1.3. Phase 3: Another adversarial traffic generation phase



Hình 3.3: *Tấn công đối kháng.*

Từ Hình 3.3 mô hình đã huấn luyện từ phase1, ta tiến hành trích xuất các đặc trưng có trong mô hình dựa trên các thuật toán tấn công WB hoặc BB có thể cần thiết để tiến hành tấn công tạo mẫu đối kháng.

3.1.4. Phase 4: Excute attack phase

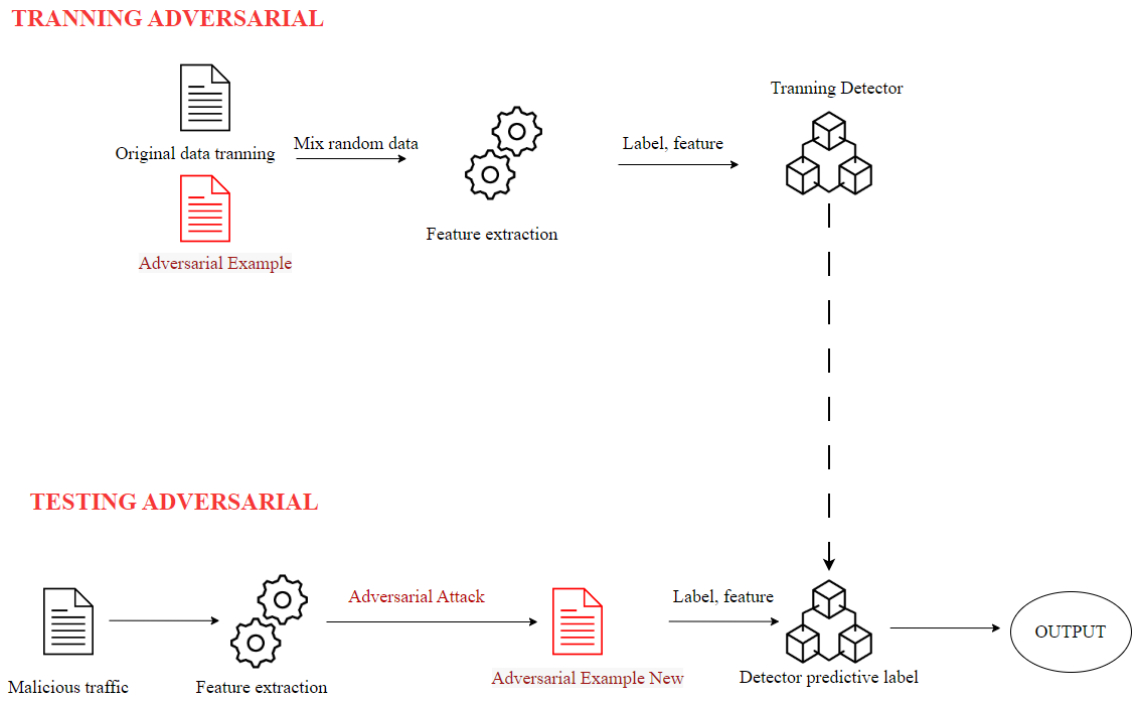


Hình 3.4: *Quy trình thực hiện tấn công.*

Hình 3.4 cho biết quy trình thực hiện tấn công, sử dụng các mẫu đối kháng sinh ra từ các phương pháp ở trên đưa vào bộ phân loại mục tiêu, mục đích của giai đoạn này là để đánh giá lại khả năng phát hiện tấn công của các bộ phân

loại trước các mẫu đối kháng, đánh giá tính mạnh mẽ của các bộ phân loại trước mẫu đối kháng.

3.1.5. Phase 5: Defense Adversarial Training



Hình 3.5: *Quy trình Adversarial Tranning và Defense.*

Hình 3.5 mô tả cách thức hoạt động của phòng thủ đối kháng. Có thể chia ra làm 2 phần chính là Tranning Adversarial và Testing Adversarial. Cụ thể:

- **Tranning Adversarial**: Cùng với dữ liệu gốc, chúng tôi sẽ trộn dữ liệu đối kháng đã được tạo ra bởi các thuật toán đã nêu ở chương 2.6.3. Sau đó chúng tôi tiến hành xử lí dữ liệu và thu thập những thông tin cần thiết như label, feature để có thể nạp vào các mô hình để Tranning.

- **Testing Adversarial**: Phần dữ liệu đem đi test, xử lí dữ liệu và trích xuất các mẫu là Malicious đem đi tạo mẫu đối kháng mới (khác với những mẫu đã được huấn luyện ở phần tranning). Sau đó đưa vào mô hình đã được huấn luyện với mẫu đối kháng để nó dự đoán.

3.2. Bộ phân loại

Chi tiết chúng tôi sẽ xử lí dữ liệu và xây dựng bộ phân loại như sau:

3.2.1. Gaussian Naive Bayes - GNB

Gaussian Naive Bayes là một thuật toán học máy thuộc loại phân loại và là một biến thể của Naive Bayes. Thuật toán này dựa trên định lý Bayes với giả định rằng các đặc trưng đầu vào độc lập với nhau và mỗi đặc trưng tuân theo phân phối chuẩn (Gaussian). Nó bao gồm định lí Bayes, giả định độc lập và phân phối chuẩn

$$P(C|X) = \frac{P(X|C) \cdot P(C)}{P(X)} \tag{3.1}$$

Trong đó:

- $P(C|X)$ là xác suất của lớp C khi biết đặc trưng X.
- $P(X|C)$ là xác suất của đặc trưng X khi biết lớp C.
- $P(C)$ là xác suất tiên nghiệm của lớp C.
- $P(X)$ là xác suất của đặc trưng X.

Thuật toán Naive Bayes giả định rằng các đặc trưng đầu vào độc lập với nhau, nghĩa là xác suất có điều kiện của các đặc trưng có thể được viết thành tích của các xác suất riêng lẻ:

$$P(X|C) = \prod_{i=1}^{n} P(x_i|C) \tag{3.2}$$

Ở đây, X là vector của các đặc trưng và x_i là giá trị của đặc trưng thứ i.

Trong Gaussian Naive Bayes, các đặc trưng được giả định tuân theo phân phối chuẩn. Xác suất có điều kiện $P(x_i|C)$ được tính dựa trên phân phối chuẩn như sau:

$$P(x_i|C) = \frac{1}{\sqrt{2\pi\sigma_C^2}} \exp\left(-\frac{(x_i - \mu_C)^2}{2\sigma_C^2}\right) \tag{3.3}$$

Trong đó:

- μ_C là giá trị trung bình của đặc trưng x_i trong lớp C.
- σ_C^2 là phương sai của đặc trưng x_i trong lớp C.

Đây là cách tính của thư viện sklearn. Chúng tôi sử dụng thư viện hỗ trợ sklearn cho việc xây dựng mô hình này.

3.2.2. Decision Trees - DT

Chúng tôi sẽ tiến hành xây dựng kiến trúc DT dựa trên ý tưởng toán sau: Để xây dựng một cây quyết định phân loại (Decision Tree Classifier), thuật toán thực hiện một loạt các bước để chọn ra các tiêu chí chia tối ưu tại mỗi nút. Quy trình này chủ yếu dựa trên việc giảm thiểu độ thuần nhất (impurity) của các nút con so với nút cha. Các tiêu chí thuần nhất phổ biến được sử dụng là Gini impurity và entropy:

Đo lường độ thuần nhất: Gini Impurity và Entropy

$$Gini(t) = 1 - \sum_{i=1}^{n} p_i^2 \tag{3.4}$$

Trong đó:

- $Gini(t)$ là Gini impurity tại nút t.
- p_i là tỷ lệ mẫu thuộc lớp i tại nút t.
- n là số lượng lớp.

$$Entropy(t) = -\sum_{i=1}^{n} p_i \log_2(p_i) \tag{3.5}$$

Trong đó: - $Entropy(t)$ là entropy tại nút t.

- p_i là tỷ lệ mẫu thuộc lớp i tại nút t.

- n là số lượng lớp.

Tìm Tiêu Chí Chia Tối Ưu (Optimal Splitting Criterion)

Để đánh giá việc chia dữ liệu tại một nút t thành hai nút con t_L và t_R, ta tính toán impurity sau khi chia:

$$Impurity_{split} = \frac{N_L}{N} \cdot Impurity(t_L) + \frac{N_R}{N} \cdot Impurity(t_R) \tag{3.6}$$

Trong đó:

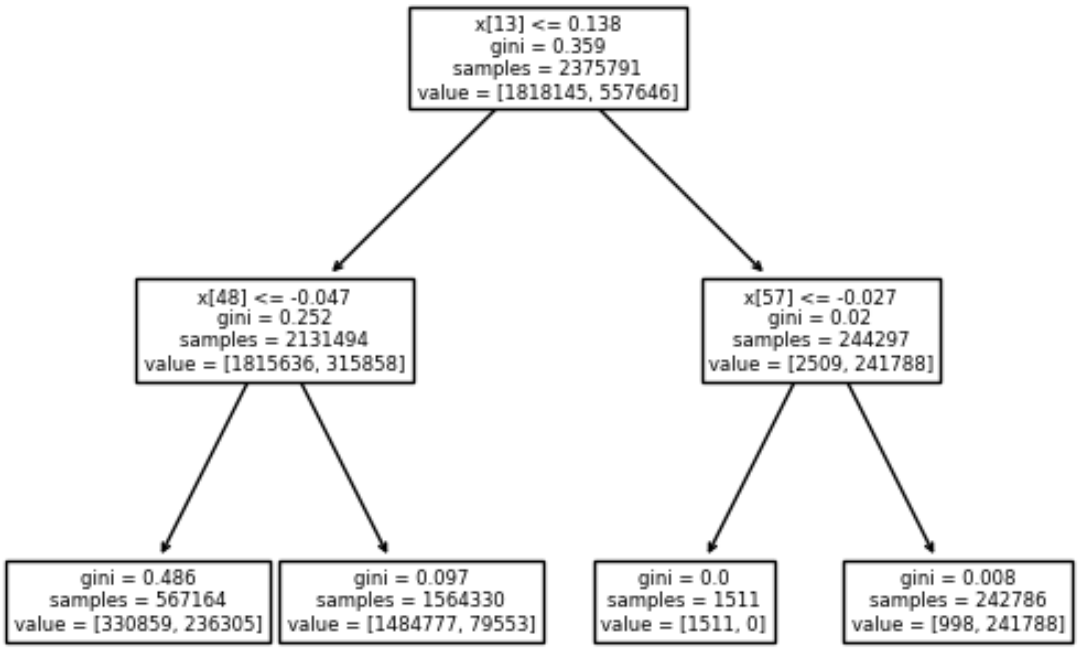
- $Impurity(t)$ là impurity (Gini hoặc entropy) tại nút t.

- N là tổng số mẫu tại nút t.

- N_L là số mẫu tại nút con trái t_L.

- N_R là số mẫu tại nút con phải t_R.

Đây là cách tính của thư viện sklearn. Chúng tôi sẽ sử dụng thư viện này để hỗ trợ cho việc xây dựng mô hình DT. Hình 3.6 mô tả một phần của kết quả từ một cây quyết định (Decision Tree) được biểu diễn bằng đồ thị. Mỗi nút trong cây quyết định có một số thông tin liên quan đến tiêu chí chia (splitting criterion), độ thuần nhất (impurity), số lượng mẫu (samples), và giá trị phân loại (value).



Hình 3.6: *Một phần kết quả của cây phân loại sử dụng trong KLTN.*

Cây quyết định này phân chia dữ liệu dựa trên các đặc điểm như $x[13]$, $x[48]$, và $x[57]$, nhằm tối ưu hóa chỉ số Gini. Mục tiêu của cây quyết định là tạo ra các nút lá nơi các mẫu dữ liệu càng thuần nhất càng tốt, với chỉ số Gini càng thấp càng tốt.

Từ nút gốc với mức độ hỗn loạn trung bình (chỉ số Gini = 0.359), cây dần dần phân chia dữ liệu để tạo ra các nút lá với mức độ hỗn loạn giảm đi. Cụ thể:

- Nút gốc:

  Điều kiện phân chia: $x[13] \leq 0.138$

  Chỉ số Gini: 0.359

  Số lượng mẫu: $2,375,791$

  Số lượng mẫu thuộc mỗi lớp: $[1,818,145,557,646]$

- Nút lá bên trái:

  Chỉ số Gini: 0.486

  Số lượng mẫu: $567,164$

  Số lượng mẫu thuộc mỗi lớp: $[330, 859, 236, 305]$

- Nút lá bên phải:

  Chỉ số Gini: 0.0

  Số lượng mẫu: $1,511$

  Số lượng mẫu thuộc mỗi lớp: $[1, 511, 0]$

Cây quyết định này cho thấy sự phân chia dữ liệu hiệu quả và rõ ràng giữa các lớp, với chỉ số Gini giảm dần từ nút gốc đến các nút lá.

3.2.3. Logistic Regression - LR

Logistic Regression là một trong những thuật toán cơ bản trong học máy, được sử dụng chủ yếu để giải quyết các bài toán phân loại (classification). Điều đặc biệt của Logistic Regression là nó áp dụng hàm sigmoid để ước tính xác suất của một điểm dữ liệu thuộc vào từng lớp. Cụ thể mô hình logistic regression có thể được định nghĩa như sau:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{3.7}$$

Trong đó:

- $z = \theta^T \mathbf{x}$

- θ là vector hệ số (trọng số).

- $\mathbf{x}$ là vector các biến độc lập.

Hàm giả thuyết $h_\theta(\mathbf{x})$ được tính bằng:

$$h_\theta(\mathbf{x}) = \sigma(\theta^T \mathbf{x}) = \frac{1}{1 + e^{-\theta^T \mathbf{x}}} \tag{3.8}$$

Hàm mất mát được sử dụng trong logistic regression là hàm cross-entropy (log-loss):

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \log(h_\theta(\mathbf{x}^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(\mathbf{x}^{(i)})) \right] \tag{3.9}$$

Trong đó

- m là số lượng ví dụ huấn luyện

-$y^{(i)}$ là nhãn thực tế của ví dụ thứ i

- và $h_\theta(\mathbf{x}^{(i)})$ là xác suất dự đoán của ví dụ thứ i thuộc lớp 1.

Các tham số θ được tối ưu hóa bằng phương pháp gradient descent hoặc các thuật toán tối ưu hóa khác để giảm thiểu hàm mất mát $J(\theta)$.

Để ngăn chặn hiện tượng overfitting, các thuật ngữ chính quy hóa như $L1$ (Lasso) và $L2$ (Ridge) có thể được thêm vào hàm mất mát.

Đây là cách tính của thư viện sklearn trong Python hỗ trợ. Chúng tôi sẽ sử dụng thư viện này cho việc xây dựng mô hình LR.

3.2.4. Deep Neural Networks - DNN

Chúng tôi tiến hành xây dựng kiến trúc mô hình DNN (kí hiệu là DNN2) gồm 6 lớp được mô tả ở Hình 3.7 như sau:

Layer (type)	Output Shape	Param #
dense_37 (Dense)	(None, 64)	4,416
dropout_68 (Dropout)	(None, 64)	0
dense_38 (Dense)	(None, 128)	8,320
dropout_69 (Dropout)	(None, 128)	0
dense_39 (Dense)	(None, 512)	66,048
dropout_70 (Dropout)	(None, 512)	0
dense_40 (Dense)	(None, 256)	131,328
dropout_71 (Dropout)	(None, 256)	0
dense_41 (Dense)	(None, 128)	32,896
dropout_72 (Dropout)	(None, 128)	0
dense_42 (Dense)	(None, 1)	129

Hình 3.7: *Kiến trúc của DNN2.*

Kiến trúc: Mạng nơ-ron bắt đầu với các đầu vào có kích thước không xác định (None) và từ từ giảm số nơ-ron mỗi lớp từ 512 xuống còn 1.

Các lớp Dropout: Các lớp Dropout được áp dụng sau mỗi lớp dense, giúp ngăn ngừa việc mô hình quá khớp bằng cách ngẫu nhiên đặt một phần đơn vị đầu vào thành 0 trong quá trình huấn luyện.

Số lượng Tham số: Tổng số tham số có thể huấn luyện trong mạng là tổng của số tham số trong tất cả các lớp dense. Ở đây, tổng số tham số là 243,537.

Output Shape: Qua các lớp, chiều đầu tiên là None, cho thấy kích thước batch có thể thay đổi dựa trên dữ liệu đầu vào huấn luyện hoặc dự đoán.

Kiến trúc DNN này được thiết kế để giải quyết một bài toán phân loại nhị phân, có thể là trên một tập dữ liệu kích thước trung bình. Việc sử dụng các lớp dropout giữa các lớp dense cho thấy một chiến lược để cải thiện khả năng

tổng quát hóa và ngăn ngừa việc mô hình overfitting, điều này rất quan trọng trong các mạng nơ-ron, đặc biệt là khi xử lý kiến trúc Deep Learning.

Đối với phần xây dựng DNN cho phần Transferability, thì chúng tôi xây dựng thêm một mô hình DNN1 bao gồm 4 lớp dense (giảm đi 2 lớp dense so với DNN2) và được mô tả như hình 3.8

Layer (type)	Output Shape	Param #
dense_43 (Dense)	(None, 64)	4,416
dropout_73 (Dropout)	(None, 64)	0
dense_44 (Dense)	(None, 128)	8,320
dropout_74 (Dropout)	(None, 128)	0
dense_45 (Dense)	(None, 256)	33,024
dropout_75 (Dropout)	(None, 256)	0
dense_46 (Dense)	(None, 1)	257

Hình 3.8: *Kiến trúc của DNN1 cho việc đánh giá Transferability .*

3.2.5. Long Shot Term Memory - LSTM

Layer (type)	Output Shape	Param #
lstm_44 (LSTM)	(None, 68, 128)	66,560
dropout_64 (Dropout)	(None, 68, 128)	0
lstm_45 (LSTM)	(None, 68, 256)	394,240
dropout_65 (Dropout)	(None, 68, 256)	0
lstm_46 (LSTM)	(None, 68, 512)	1,574,912
dropout_66 (Dropout)	(None, 68, 512)	0
lstm_47 (LSTM)	(None, 128)	328,192
dropout_67 (Dropout)	(None, 128)	0
dense_36 (Dense)	(None, 1)	129

Hình 3.9: *Kiến trúc của LSTM2.*

Chúng tôi tiến hành xây dựng kiến trúc mô hình LSTM (kí hiệu là LSTM2) gồm 4 lớp LSTM và 1 lớp Dense được mô tả chi tiết ở Hình 3.9 như sau:

Lstm_44 (LSTM): Đây là lớp LSTM đầu tiên với output shape là (None, 68, 128). Điều này có nghĩa là lớp này nhận đầu vào với chiều dài sequence là 68 (độ dài của mỗi input sequence), và cho ra đầu ra với kích thước (None, 68, 128), trong đó None thường đại diện cho batch size, và 128 là số lượng đơn vị LSTM. Lstm_45 (LSTM): Lớp LSTM thứ hai có output shape là (None, 68, 256), với 256 đơn vị LSTM. Điều này cho thấy lớp này đã được cấu hình để có nhiều đơn vị hơn so với lớp trước đó.

Dropout_64, dropout_65, dropout_66, dropout_67: Đây là các lớp Dropout, được sử dụng để ngẫu nhiên bỏ qua một số đơn vị đầu vào trong quá trình huấn luyện, nhằm mục đích ngăn chặn overfitting

Dense_36 (Dense): Đây là lớp Dense (hoặc Fully Connected Layer) cuối cùng với 1 đơn vị đầu ra. Đây có thể là lớp đầu ra của mô hình, được sử dụng trong các bài toán regression (dự đoán giá trị liên tục) hoặc classification nhị phân.

Cột "Param #" chỉ ra số lượng tham số (weights và biases) của mỗi lớp. Số lượng tham số phụ thuộc vào kích thước đầu vào và đầu ra của từng lớp, cũng như cấu hình của mô hình.

Mô hình của LSTM1 mô tả ở 3.9 có 4 lớp LSTM với số đơn vị tăng dần từ 128 -> 256 -> 512 -> 128. Sau đó là một lớp Dense với 1 đơn vị đầu ra. Các lớp Dropout được sử dụng để giảm thiểu overfitting trong quá trình huấn luyện. Tổng số tham số của mô hình (gần 2 triệu) cho thấy mô hình này có độ phức tạp vừa phải, đủ để học được từ dữ liệu mà không quá phức tạp đến mức gây overfitting nghiêm trọng.

Cũng như DNN, đối với phần xây dựng LSTM cho phần Transferability, thì chúng tôi xây dựng thêm một mô hình LSTM1 bao gồm 2 lớp lớp LSTM và 1 lớp dense (giảm đi 2 lớp LSTM so với LSTM2) và được mô tả như ở hình 3.10

Layer (type)	Output Shape	Param #
lstm_48 (LSTM)	(None, 68, 128)	66,560
dropout_76 (Dropout)	(None, 68, 128)	0
lstm_49 (LSTM)	(None, 68, 256)	394,240
dropout_77 (Dropout)	(None, 68, 256)	0
dense_47 (Dense)	(None, 68, 1)	257

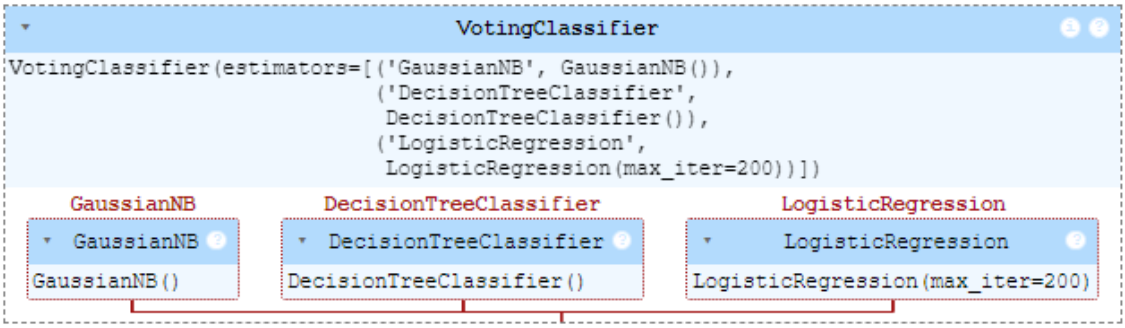
Hình 3.10: *Kiến trúc của LSTM1 cho việc đánh giá Transferability .*

3.2.6. Ensemble Learning

Ở phần này, chúng tôi mô tả cách tổng hợp mô hình Ensemble Learning từ các mô hình detector đơn lẻ đã nêu ở những phần trước.

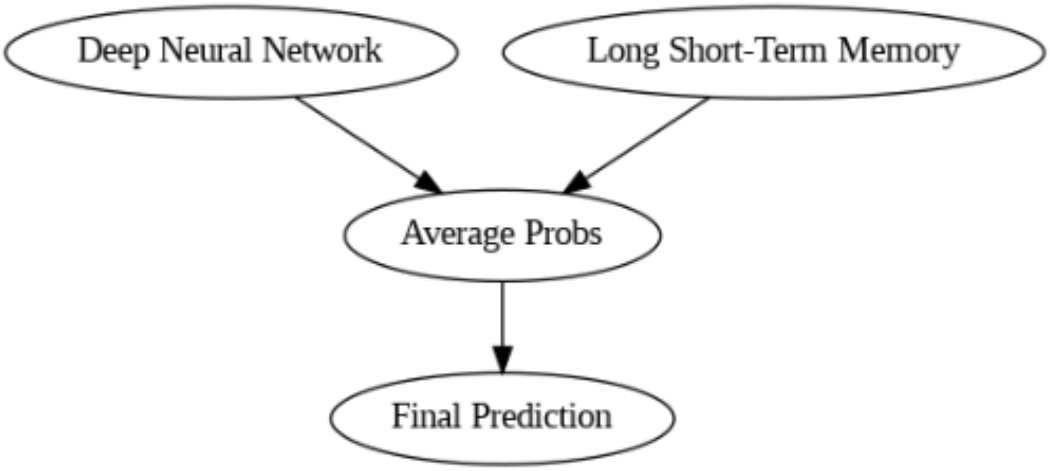
Tổng hợp Ensemble Learning từ các mô hình Machine Learning. Hình 3.11 mô tả Ensemble Machine Learning được tổng hợp từ GNB, DT, LR dưới phương thức VotingClassifier - dạng Hard votting - đưa ra dự đoán cuối cùng bằng cách bỏ phiếu (voting) dựa trên số phiếu đa số của các mô hình con. Tức là, lớp nhãn được bầu chọn nhiều nhất sẽ là dự đoán cuối cùng của VotingClassifier.

GNB, DT và LR đều sử dụng các phương pháp tiếp cận khác nhau để phân loại, do đó khi kết hợp chúng lại, hệ thống có thể tận dụng sự đa dạng này để cải thiện độ chính xác tổng thể. GNB dựa trên lý thuyết xác suất, DT dựa trên các quy tắc quyết định, và LR dựa trên mô hình tuyến tính. Vì vậy, mô hình đem lại hiệu quả trong việc giúp cho các mô hình dự đoán kém cải thiện độ chính xác và đưa ra kết quả tốt hơn khi hoạt động kết hợp.



Hình 3.11: *Mô hình tổng hợp Ensemble Learning dựa trên các mô hình Machine Learning.*

Hình 3.12 mô tả Ensemble Learning được tổng hợp từ 2 mô hình học sâu là DNN và LSTM dưới phương thức - Soft Voting - đưa ra dự đoán cuối cùng bằng cách bỏ phiếu (voting) dựa trên số phiếu đa số của các mô hình con. Tức là, lớp nhãn được bầu chọn nhiều nhất sẽ là dự đoán cuối cùng của VotingClassifier.



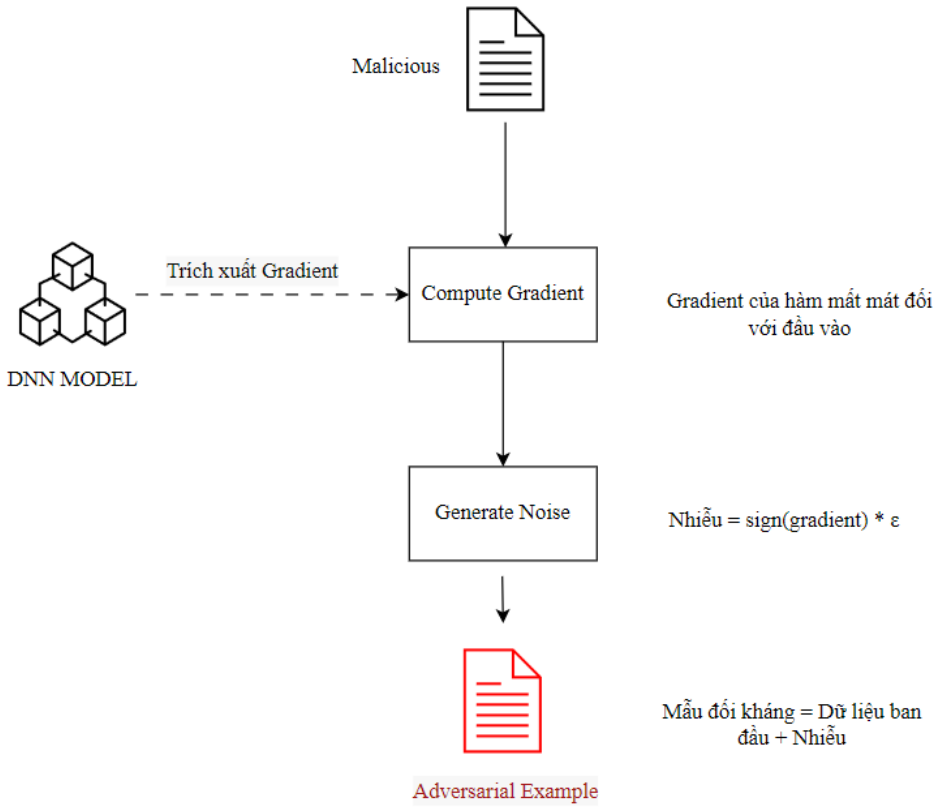
Hình 3.12: *Mô hình tổng hợp Ensemble Learning dựa trên các mô hình Deep Learning.*

Average Probs (AVG) ở Hình 3.12 là bước trung bình hóa xác suất dự đoán từ DNN và LSTM. Thường là trung bình cộng của các xác suất được tính toán từ hai mô hình này. Final Prediction là dự đoán cuối cùng được xác định dựa trên kết quả từ AVG. Trong trường hợp này, AVG đóng vai trò như một loại bình chọn mềm (soft voting), tức là sử dụng trung bình xác suất để đưa ra quyết định cuối cùng.

3.3. Mô hình của các thuật toán tấn công đối kháng

Ở chương này, chúng tôi mô tả cách thức hoạt động của các cuộc tấn công đối kháng bởi các thuật toán đã được mô tả ở chương 2.6.3 lên các mô hình học máy và học sâu. Chúng tôi lựa chọn mô hình DNN để minh họa việc này.

3.3.1. Fast Gradient Sign Method - FGSM



Hình 3.13: *FGSM attack.*

Hình 3.13 mô tả quy trình tạo ra các mẫu đối kháng bằng thuật toán FGSM. Vì FGSM thuộc dạng tấn công WB nên sẽ cần thêm các thông số của mô hình gốc (ở đây là DNN).

Cụ thể như sau:

- Trích xuất từ dữ liệu thô đã được xử lí, chỉ lấy dữ liệu Malicious để tạo dữ liệu đối kháng.
- Mô hình mạng nơ-ron sâu (DNN) đã được huấn luyện. Sau khi huấn luyện, tiến hành trích xuất Gradient này. Gradient này cho biết độ nhạy của đầu ra của mô hình đối với sự thay đổi nhỏ trong dữ liệu đầu vào.

- Compute Gradient: Gradient được tính toán thông qua việc sử dụng các thuật toán tự động phân biệt (automatic differentiation) trong các framework học sâu như TensorFlow hoặc PyTorch. Gradient của hàm mất mát được tính toán như sau:

$$\nabla_x J(\theta, x, y) \tag{3.10}$$

  Trong đó J là hàm mất mát, θ là các tham số của mô hình, x là dữ liệu đầu vào, và y là nhãn tương ứng. được tính toán, trong đó J là hàm mất mát, θ là các tham số của mô hình, x là dữ liệu đầu vào, và y là nhãn tương ứng.

- Generate Noise: Nhiễu được tính như sau:

$$\eta = \epsilon \cdot \text{sign}(\nabla_x J(\theta, x, y)) \tag{3.11}$$

  Trong đó:

  – ϵ là một siêu tham số điều khiển cường độ của nhiễu.

  – sign là hàm dấu, trả về -1, 0, hoặc 1 dựa trên dấu của gradient.

- Adversarial Example: Mẫu đối kháng x' được tạo ra bằng cách thêm nhiễu vào dữ liệu đầu vào ban đầu:

$$x' = x + \eta \tag{3.12}$$

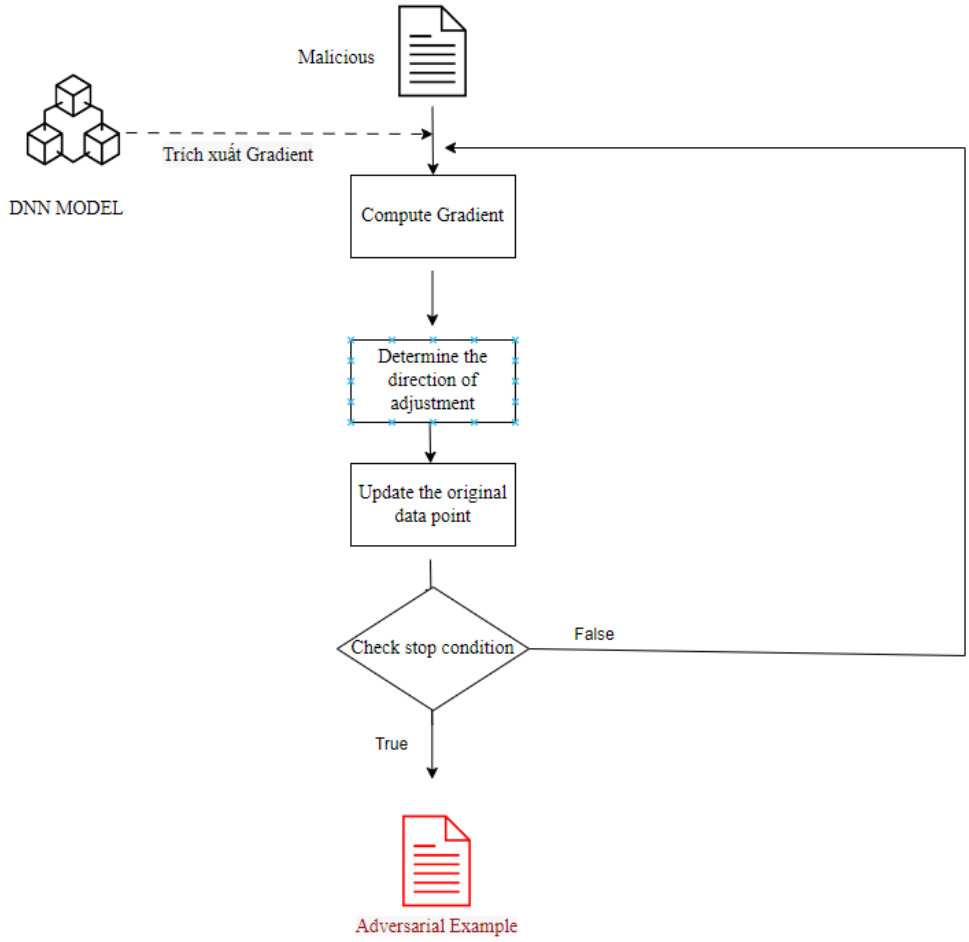
  Trong đó x' là dữ liệu đầu vào sau khi thêm nhiễu, x là dữ liệu đầu vào ban đầu, và η là nhiễu được thêm vào.

Adversarial Example x' được tạo ra có khả năng kháng các bộ phân loại làm giảm đi hiệu suất của mô hình.

3.3.2. Deepfool

Từ cơ sở lí thuyết của Deepfool được mô tả ở chương 2.6.3.3, hình 3.14 mô tả quy trình tạo ra các mẫu đối kháng. Vì Deepfool cũng thuộc dạng tấn công

WB nên sẽ cần các thông số của mô hình gốc (DNN).



Hình 3.14: *Deepfool attack.*

Cụ thể:

- Trích xuất từ dữ liệu thô đã được xử lí, chỉ lấy dữ liệu Malicious để tạo dữ liệu đối kháng và các thoong số Gradient của mô hình DNN.
- Tính gradient của loss funciton: Theo công thức bên dưới cho biết làm thế nào để tính toán gradient của hàm mất mát tại điểm dữ liệu x.

$$\nabla_x L(x) = \nabla_x f(x) - \nabla_x f(x_{\text{orig}}) \tag{3.13}$$

Trong đó:

- $\nabla_x L(x)$: Gradient của hàm mất mát tại điểm x.
- $f(x)$: Đầu ra của mô hình DNN cho điểm dữ liệu x.
- x_{orig}: Điểm dữ liệu gốc.

- Xác định hướng điều chỉnh:

$$\eta = \frac{\nabla_{\mathbf{x}} L(\mathbf{x})}{\|\nabla_{\mathbf{x}} L(\mathbf{x})\|_2} \tag{3.14}$$

trong đó $\|\cdot\|_2$ là norm chuẩn 2 để chuẩn hóa vector gradient. $\nabla_{\mathbf{x}} L(\mathbf{x})$ mô tả cách tính hướng điều chỉnh.

- Cập nhật điểm dữ liệu gốc theo công thức sau

$$\mathbf{x}(t+1) = \mathbf{x}(t) + \delta \cdot \eta \tag{3.15}$$

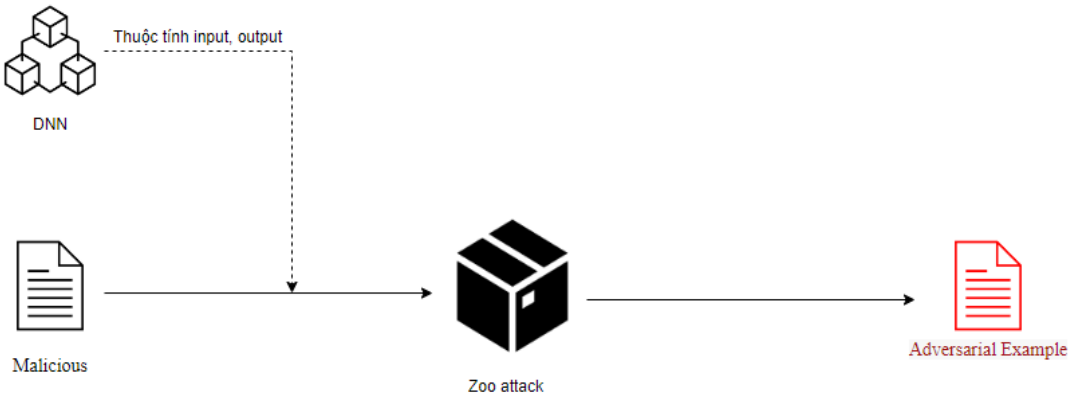
cho biết cách cập nhật điểm dữ liệu gốc để tạo ra một mẫu đối kháng mới. Trong đó, $\mathbf{x}(t)$ là điểm dữ liệu gốc tại thời điểm t, δ là hệ số điều chỉnh, và η là hướng điều chỉnh được tính bằng gradient chuẩn hóa của hàm mất mát.

- Điều kiện dừng của thuật toán là một phần quan trọng để kiểm soát quá trình tạo ra mẫu đối kháng, đảm bảo rằng quá trình là hiệu quả và không gây ra những thay đổi không mong muốn đối với dữ liệu gốc. Nó có thể là:
  - **Số lần lặp (Max Iterations):** Thuật toán có thể được thiết lập để lặp lại một số lượng cố định các bước cập nhật trước khi dừng lại.
  - **Ngưỡng perturbation (Perturbation Threshold):** Điều này có thể được định nghĩa bằng một giá trị ngưỡng cho phép, ví dụ như khoảng cách Euclide giữa mẫu đối kháng mới và mẫu gốc không vượt quá một ngưỡng nhất định.

– **Độ lớn của perturbation (Perturbation Magnitude):** Thuật toán có thể dừng lại nếu độ lớn của perturbation vượt quá một ngưỡng nhất định, để tránh làm thay đổi quá mạnh mẫu dữ liệu gốc.

– **Đạt được mục tiêu (Achieving Target):** Nếu mục tiêu của thuật toán là tạo ra một mẫu đối kháng đủ khả năng làm cho mô hình phân loại sai, thuật toán có thể dừng khi mục tiêu này được đạt.

3.3.3. ZOO

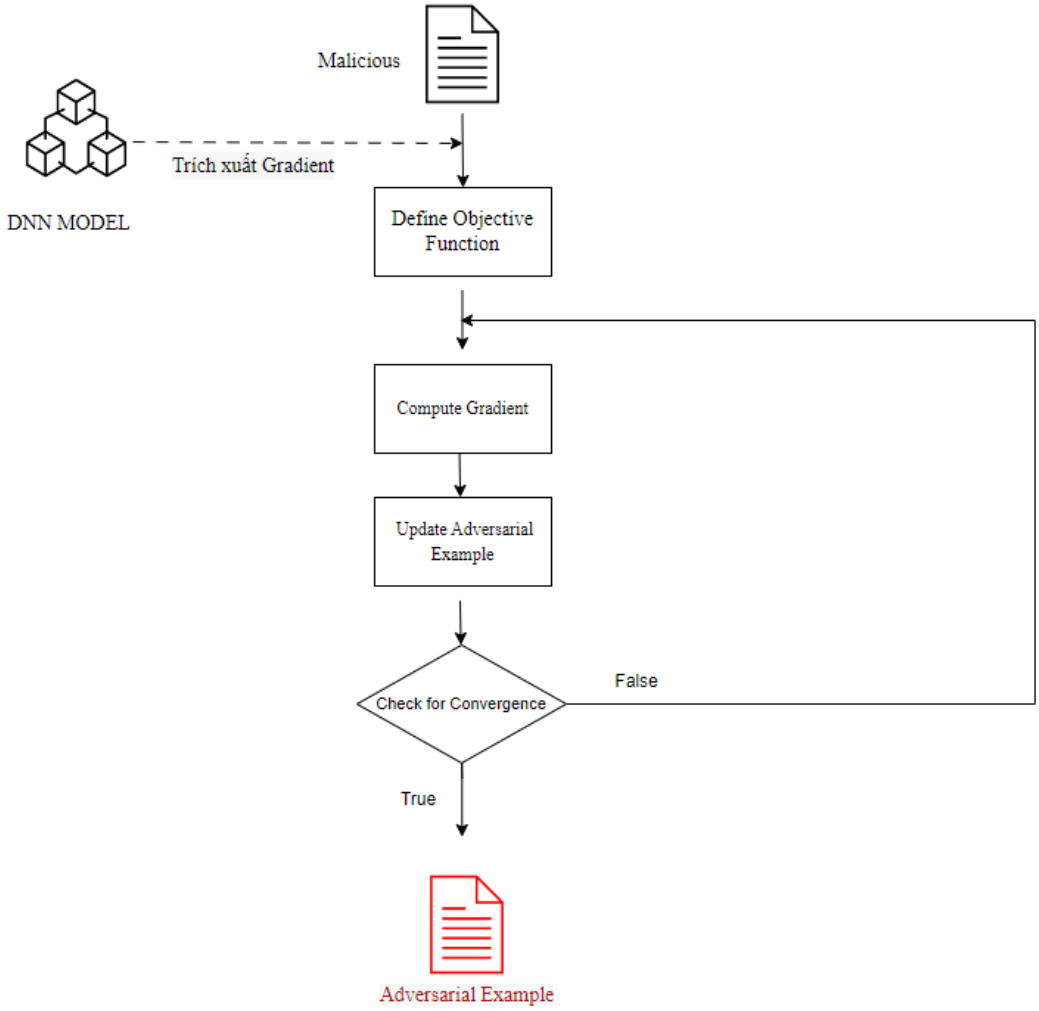


Hình 3.15: *ZOO attack.*

Dựa theo mô tả thuật toán ZOO ở chương 2.6.3.2, hình 3.15 mô tả quy trình tạo ra mẫu đối kháng. Zoo thuộc loại tấn công BB nên nó chỉ có thể trích xuất các thuộc tính đầu vào và đầu ra của mô hình gốc.

3.3.4. C&W

Từ cơ sở lí thuyết được mô tả ở chương 2.6.3.4, hình 3.16 mô tả quy trình tạo mẫu đối kháng. C&W cũng là dạng tấn công thuộc WB nên có cần thêm những dữ liệu từ mô hình gốc như là Gradient, input, output...



Hình 3.16: *Carlini & Wagner attack.*

Cụ thể quy trình:

- Trích xuất các dữ liệu Malicious, khởi tạo đầu vào ban đầu, nhãn mục tiêu và các tham số cần thiết cho quá trình tối ưu hóa. Thông thường, $\mathbf{x}$ là mẫu đầu vào ban đầu và t là nhãn mà ta muốn mô hình phân loại sai thành.
- Define Objective Function: Hàm mục tiêu $f(\mathbf{x}')$ được xác định để đánh giá hai mục tiêu chính: làm sao để mô hình phân loại sai và làm sao để biến đổi nhỏ nhất có thể. Thường, hàm này bao gồm một hàm mất mát $L(\mathbf{x}', t)$ để đánh giá khả năng phân loại sai và một thành phần điều chỉnh để kiểm

soát độ lớn của biến đổi $c \cdot \|\mathbf{x}' - \mathbf{x}\|_2$.

- Vòng lặp tối ưu hóa:
  - **Compute Gradient**: Gradient của hàm mục tiêu đối với $\mathbf{x}'$ được tính toán.
  - **Update Adversarial Example**: Mẫu đối kháng $\mathbf{x}'$ được cập nhật dựa trên gradient và hệ số học (η).
  - **Check for Convergence**: Kiểm tra nếu điều kiện hội tụ được thỏa mãn hoặc đạt tới giá trị tới hạn của vòng lặp (max_iter)
- Sau khi điều kiện hội tụ đạt được, mẫu đối kháng $\mathbf{x}'$ sẽ được xuất ra và có khả năng kháng các bộ phân loại.

CHƯƠNG 4. THÍ NGHIỆM VÀ ĐÁNH GIÁ

Ở chương này chúng tôi tiến tạo môi trường, cài đặt và triển khai các phần của mô hình, tiến hành chạy và nghiệm thu kết quả.

4.1. Thiết lập thí nghiệm

4.1.1. Môi trường cần thiết

Ngôn ngữ chính để xây dựng hệ thống: Python 3.9 Các thư viện cần thiết để hỗ trợ cho Python là : Tensorflow, padas, keras, skearlean, pytouch,...

Kết quả được thực nghiệm trên 2 máy tính Intel Core i7-10900U với dung lượng RAM là 64GB và máy tính Intel Xeon E5-2680 v4 với dung lượng RAM là 128GB, 1 máy Ubuntu với RAM là 16GB.

4.1.2. Tiêu chí đánh giá

Tiêu chí đánh giá của chúng tôi trên dữ liệu và mô hình là về Accuracy (độ chính xác), Precision (độ chính xác dự đoán là tấn công), Recall (tỷ lệ phát hiện tấn công), và F1-Score dựa trên các giá trị True Positives (TP), True Negatives (TN), False Positives (FP), và False Negatives (FN) qua các công thức:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4.2}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4.3}$$

$$\text{F1-score} = \frac{2 \cdot (\text{Recall} \cdot \text{Precision})}{\text{Recall} + \text{Precision}} \tag{4.4}$$

Ngoài ra, khi chúng tôi kiểm tra mẫu đối kháng lên các bộ phân loại thì chúng tôi sẽ đánh giá dựa trên chỉ số có tên là **Detection rate** với định nghĩa và công thức như sau:

Định nghĩa: Tỉ lệ phát hiện (Detection Rate) là một chỉ số đo lường hiệu suất của một hệ thống trong việc xác định đúng các trường hợp dương tính thực sự.

$$\text{Detection Rate} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}} \tag{4.5}$$

4.1.3. Tập dữ liệu

Chúng tôi tiến hành thí nghiệm trên các tập dữ liệu là : NSL-KDD, CICIDS-2017, CIC-IOT2023. Chúng tôi sẽ giới thiệu sơ qua các bộ dữ liệu này.

4.1.3.1. NSL-KDD

NSL-KDD là một bộ dữ liệu được phát triển nhằm giải quyết các hạn chế của bộ dữ liệu KDD'99. Đây là một bộ dữ liệu dùng để phát hiện và phân loại các cuộc tấn công mạng và là một trong những bộ dữ liệu phổ biến nhất trong lĩnh vực an ninh mạng.

Phát triển từ bộ dữ liệu KDD'99 với mục tiêu giảm thiểu sự dư thừa và không cân đối của dữ liệu.Bao gồm 41 đặc trưng và một nhãn lớp cho mỗi kết nối mạng. Các đặc trưng bao gồm thông tin về kết nối như thời gian, dịch vụ, giao thức, và các thông số về lưu lượng mạng. Chia làm 4 nhóm chính: DoS (Denial of Service), R2L (Remote to Local), U2R (User to Root), và Probe.

4.1.3.2. CICIDS-2017

Bộ dữ liệu CICIDS2017 [10], được phát hành năm 2017 và rất phổ biến trong các thí nghiệm phát hiện xâm nhập, là một tập dữ liệu thời gian thực đáng tin

cậy, chứa các cuộc tấn công phổ biến nhất từ các kịch bản tấn công thực tế. Bộ dữ liệu này thu thập lưu lượng bình thường và tấn công từ thứ Hai đến thứ Sáu, dựa trên lưu lượng nền được tạo ra thời gian thực. Lưu lượng bình thường vào thứ Hai được tạo bởi hệ thống B-Profile, có khả năng phân tích hành vi trừu tượng của tương tác con người. Trong bốn ngày còn lại, lưu lượng tấn công khác nhau được tạo ra xen kẽ với lưu lượng bình thường. Sau đó, lưu lượng thu thập được đánh nhãn chính xác bởi các chuyên gia. Bộ dữ liệu cũng bao gồm các nhãn tấn công như Web attack, Brute Force FTP, Brute Force SSH, DoS, DDoS, Infiltration, Bot và PortScan. Bộ dữ liệu CICIDS2017 không chỉ cung cấp kết quả phân tích lưu lượng mạng mà còn bao gồm thời điểm, giao thức, cổng nguồn và đích. Chúng tôi đã chọn 77 đặc trưng trong bộ dữ liệu này và chia thành hai nhóm: một nhóm dựa trên thông tin gói tin và một nhóm dựa trên các chỉ số lưu lượng. Kết quả chi tiết được trình bày trong Bảng 4.1.

Bảng 4.1: *Feature groups in CICIDS 2017 dataset.*

Feature group	Description	Number
Package	Such as 'total_fpkts', 'total_fpkl', 'min_fpkl', 'mean_fpkl', 'std_fpkl', 'total_bpkl', 'min_bpkl', 'max_fpkl', 'max_bpkl', and so on	34
Traffic	Such as 'Duration', 'min_flowpkl', 'std_flowpkl', 'PktsPerSecond', 'BPktsPerSecond', 'downUpRatio', 'FAvgBytePerBulk', 'FAvgPacketsPerBulk', 'slow_bytes', 'slow_bytes', and so on	43

.

4.1.3.3. CIC-IOT2023

Bộ dữ liệu CICIOT2023 là một bộ dữ liệu mới nhất dành cho nghiên cứu an ninh mạng trong môi trường IoT (Internet of Things).

Được phát triển để đáp ứng nhu cầu nghiên cứu và phát triển các giải pháp an ninh mạng cho hệ thống IoT, một lĩnh vực đang phát triển mạnh mẽ và tiềm

ẩn nhiều rủi ro bảo mật. Bao gồm các loại tấn công và hành vi mạng đặc thù của các thiết bị IoT như cameras, smart meters, và các thiết bị gia dụng thông minh khác. Bộ dữ liệu có các đặc trưng mô tả lưu lượng mạng, hoạt động của các thiết bị IoT, và các cuộc tấn công mạng thường gặp trong môi trường IoT như tấn công Mirai botnet, Man-in-the-Middle (MitM), và tấn công từ chối dịch vụ (DoS).

4.1.4. Dectector

Chúng tôi sẽ tiên hành xây dựng các detector đã được mô tả ở mục 3.2 như sau:

- **Về Machine Learning**: sẽ xây dựng và huấn luyện các thuật toán là GNB, DT, LR và sử dụng thư viện hỗ trợ sklearn trong Python để xây dựng. Cụ thể
  - DT sẽ xây dựng với số node là 10.
  - LR sẽ xây dựng dưới số vòng lặp tối ưu (max_iter) là 200.
- **Về Deep Learning**: sẽ xây dựng và huấn luyện trên các thuật toán học sâu là DNN và LSTM.
  - DNN sẽ xây dựng 2 mô hình là DNN2 với 6 lớp Dense và DNN1 với 4 lớp Dense (để đánh giá chuyển giao).
  - LSTM cũng sẽ xây dựng 2 mô hình là LSTM2 với 4 lớp LSTM và 1 lớp Dense và LSTM1 với 2 lớp LSTM và 1 lớp Dense (để đánh giá chuyển giao).
- **Ensemble Learning**: xây dựng và huấn luyện trên việc kết hợp lại các thuật toán learning với nhau.
  - Ensemble Machine Learning (gọi tắt là Ensemble ML) sẽ kết hợp từ 3 mô hình ML là GNB, DT và LR dưới phương thức Hard-Votting.

– Ensemble Deep Learning (gọi tắt là Ensemble DL) sẽ kết hợp từ 2 mô hình DL là DNN2 và LSTM2 dưới phương thức Sort-Votting.

4.1.5. Adversarial Examples Attack

Để tạo ra mẫu đối kháng, chúng tôi sử dụng GAN và kỹ thuật Multimodal Learning để tạo mẫu đối kháng, đồng thời cũng thực hiện so sánh chất lượng mẫu đối kháng được tạo từ mô hình đề xuất với dữ liệu chỉ được tạo bởi GAN.

Chúng tôi sẽ triển khai thêm các thuật toán tấn công đã liệt kê là FGSM , Deepfool, ZOO và C&W. Các thông số thuật toán như sau:

- **FGSM**: dựa trên độ mạnh của Epsilon với giá trị 0.01, 0.02, 0.05, 0.1, 0.2 để đánh giá mẫu đối kháng được tạo ra có phụ thuộc vào epsilon không.
- **C&W, Deepfool và Zoo**: chúng tôi sẽ dựa trên vòng lặp tối ưu (max_iter) để đánh giá mẫu đối kháng được tạo ra có phụ thuộc vào max_iter không - tức là tối ưu hóa trong việc tìm ra mẫu đối kháng có tăng tuyến tính theo biến này.

4.1.6. Phân chia dữ liệu

Tập dữ liệu CICIDS bao gồm 2830743 mẫu dữ liệu, trong đó bao gồm 2273097 mẫu bình thường và 557646 mẫu độc hại. Chúng tôi tiến hành chia tập dữ liệu ra như sau:

- Đối với việc huấn luyện và test lại kiêm tra hiệu suất mô hình đã huấn luyện, chúng tôi sẽ chia ra theo nguyên tắc 8-2(tức là 80% dành cho việc huấn luyện và 20% còn lại cho việc kiểm tra hiệu suất mô hình).
- Đối với việc tạo ra mẫu đối kháng (Adversarial Attack) thì chúng tôi sẽ lấy phần dữ liệu của việc kiểm tra hiệu suất mô hình (20%). Tiến hành phân tách chỉ lấy mẫu độc hại đem đi tấn công đối kháng, còn mẫu bình thường thì giữ nguyên.

- Đối với Defense: Chúng tôi tiến hành thay thế các mẫu độc hại ở phần tấn công đối kháng bằng dữ liệu đối kháng được tạo ra ở phần này (vẫn bảo toàn được số lượng bằng số lượng mẫu gốc ban đầu) và sau đó chia theo nguyên tắc như lúc huấn luyện và kiểm tra hiệu suất mô hình.

Để cho quá trình tạo mẫu đối kháng giống với ngữ cảnh trong thực tế, chúng tôi sử dụng bộ dữ liệu CICIOT2023 để làm dữ liệu trong mô hình đề xuất GAN và Multimodal Learning, cũng như GAN để tạo mẫu đối kháng. Trong bộ dữ liệu CICIOT2023 chúng tôi sử dụng hai loại tấn công để tạo mẫu đối kháng đó là PortScan và Sql Injection bởi vì hai loại tấn công này có trong tập dữ liệu CICIDS2017 dùng để đào tạo bộ phát hiện, và bộ phân loại được đào tạo bằng CICIDS2017 có thể nhận diện được hai loại tấn công này. Trong đó PortScan có 33833 mẫu và SqlInjection có 2533 mẫu. Ngoài ra chúng tôi còn sử dụng thêm 8811 mẫu bình thường trong bộ dữ liệu CICIOT2023 để đánh giá khả năng phát hiện trên các bộ phát hiện khác.

Vì sử dụng kỹ thuật Multimodal Learning để học phân phối từ các nhóm đặc trưng, nên nhóm chúng tôi chia bộ dữ liệu CICIOT2023 thành hai nhóm đặc trưng đó là nhóm packet có 25 đặc trưng và nhóm traffic có 43 đặc trưng.

4.2. Triển khai kịch bản thí nghiệm

Ở khóa luận tốt nghiệp này, chúng tôi xây dựng ra 3 kịch bản thử nghiệm được mô tả cụ thể và kèm theo đó là tiêu chí đánh giá cụ thể từng kịch bản như sau:

Chúng tôi đề xuất xây dựng kịch bản được thể hiện chi tiết ở bảng 4.2 và tiêu chí đánh giá thể hiện chi tiết ở bảng 4.3 như sau:

Bảng 4.2: *Mô tả kịch bản*

STT	**Nội dung chính**	**Mô tả**
1	Đánh giá các bộ phân loại	Xây dựng các bộ phân loại GNB, DT, LR, DNN, LSTM và Ensemble Learning và tiến hành đánh giá.
2	Tạo dữ liệu đối kháng và đánh giá lên các bộ phân loại	Đề xuất 2 phương pháp tạo mẫu đối kháng: Phương pháp 1: Sử dụng GAN và Multimodal Learning để tạo mẫu đối kháng Phương pháp 2: Sử dụng các thuật toán tạo các mẫu đối kháng (FGSM, ZOO, CW, DeepFool) để tạo ra lưu lượng đối kháng. Đánh giá tính chuyển giao của các mẫu đối kháng trên các bộ phân loại
3	Defense IDS	Đề xuất sử dụng phương pháp Defense cải tiến hệ thống IDS (Adversarial Training). Sau đó đánh giá hệ thống IDS sau khi defense.

Bảng 4.3: *Tiêu chí đánh giá*

STT	**Tiêu chí đánh giá**
1	Các thông số đánh giá bộ phân loại (F1, Accuracy, Precision, recall). Trích xuất được label từ bộ CICIOT2023 cho việc tạo mẫu đối kháng từ GAN.
2	Thu được dữ liệu đối kháng lưu vào file csv, chỉ số Detection rates của bộ phân loại trước tấn công đối kháng.
3	Chỉ số Detection rates của bộ phân loại sau khi phòng thủ đối kháng.

4.3. Case study

Chúng tôi dựa vào những kết quả từ công trình nghiên cứu của tác giả Alhussien và các cộng sự [1] để mô tả các câu hỏi nghiên cứu (RQ) để trả lời trong phần kết quả và thảo luận.

- **RQ1:** Trong các bộ phân loại, bộ phân loại nào có hiệu suất tốt nhất?
- **RQ2:** Đối với mô hình có kiến trúc càng phức tạp thì sẽ tạo ra được mẫu đối kháng có khả năng trốn tránh cao hơn hay không?

- **RQ3:** Đối với mô hình có kiến trúc càng phức tạp thì khả năng kháng đối kháng có cao hơn những mô hình có kiến trúc đơn giản hơn hay không?
- **RQ4:** Các mẫu đối kháng được tạo ra có ảnh hưởng như thế nào tới các mô hình khi thay đổi chỉ số phụ thuộc.
- **RQ5:** Phương pháp Adversarial Training có hiệu quả tốt nhất trên mô hình nào?
- **RQ6:** Các mô hình được Defense bằng phương pháp Adversarial Training có khả năng Robustness (là có khả năng phát hiện nhiều loại tấn công khác nhau, từ những tấn công đã biết đến những tấn công mới, chưa được học và ghi nhận) hay không?

4.4. Kết quả và thảo luận

Ở mục này, chúng tôi thảo luận, trả lời cho các câu hỏi Case study về kết quả: 1) Hiệu suất của Detector trước dữ liệu gốc. 2) Tính hiệu quả của các cuộc tấn công đối kháng và tính chuyển giao của các cuộc tấn công. 3) Attack defense dùng adversarial training.

Lưu ý rằng: Chúng tôi chỉ sử dụng DNN1(4 unit) và LSTM1(3 unit) cho việc đánh giá ở phần **2) Tính hiệu quả của các cuộc tấn công đối kháng và tính chuyển giao của các cuộc tấn công**. Còn các phần còn lại chúng tôi sử dụng DNN2(6 unit) và LSTM2(5 unit) cho đồng nhất trong khoas luận và gọi chúng là lần lượt là DNN và LSTM. Các bộ phân loại sẽ được học dữ liệu từ dataset CICIDS2017.

4.4.1. Hiệu suất của các Detector trước dữ liệu gốc

Ở phần này, chúng tôi tiến hành đánh giá hiệu suất của bộ phân loại trên ba tập dữ liệu NSL-KDD, CICIDS2017 và CICIOT2023.

Bảng 4.4: *Hiệu suất của các Detector trên bộ dữ liệu NSL-KDD(%)*

Detector	**Accuracy**	**Precision**	**Recall**	**F1-score**
GNB	89.35	90.91	85.97	88.37
DT	99.81	99.83	99.76	99.8
LR	99.92	99.93	99.9	99.2
DNN 6 layers	97.12	94.81	99.26	96.99
LSTM 5 layers	96.39	96.28	96.05	96.16
Ensemble ML	99.87	99.88	99.85	99.86
Ensemble DL	98.63	99.19	97.86	98.52

Bảng 4.5: *Hiệu suất của các Detector khác nhau trên bộ dữ liệu Cicids2017(%).*

Detector	**Accuracy**	**Precision**	**Recall**	**F1-score**
GNB	89.88	96.81	98.07	80.21
DT	99.81	99.81	99.81	99.81
LR	92.93	92.79	92.93	92.83
DNN 6 layers	96.21	89.57	91.35	90.45
LSTM 5 layers	96.10	84.68	97.86	90.79
Ensemble ML	99.88	97.07	98.75	97.90
Ensemble DL	96.14	84.68	97.86	90.79

Bảng 4.4, 4.5 và 4.6 lần lượt cung cấp một cái nhìn tổng quan về hiệu suất của các bộ phân loại (detector) khác nhau trên bộ dữ liệu NSL-KDD , CICIDS2017 và CICIOT2023. Các bộ phân loại này được đánh giá qua các chỉ số quan trọng như Accuracy, Precision, Recall, và F1-score.

Các mô hình DT, LR, và Ensemble ML có hiệu suất cao nhất cho thấy các mô hình phân loại rất tốt, ít bỏ sót. Các mô hình dựa trên học sâu như DNN và LSTM cũng có hiệu suất tốt, nhưng thấp hơn một chút. Gaussian Naive Bayes có hiệu suất thấp nhất trong bảng 4.5 này, đặc biệt là về F1-score, cho thấy sự không đồng đều trong phân loại và có thể có nhiều lỗi loại I hoặc loại II.

Đối với hiệu suất của các bộ phân loại trên dữ liệu CICIOT2023 trong bảng 4.6, ta có thể thấy được Accuracy, Precision, Recall và F1-score bị giảm đáng kể so với kết quả phân loại trước các bộ dữ liệu khác, nguyên nhân là do phân phối dữ liệu của tập dữ liệu CICIOT2023 khác với tập dữ liệu dùng để đào tạo

Bảng 4.6: *Hiệu suất của các Detector trên bộ dữ liệu CICIOT2023(%)*

Detector	Accuracy	Precision	Recall	F1-score
GNB	19.91	52.48	5.4	9.8
DT	18.46	43.35	4.21	7.67
LR	24.95	66.58	13.59	22.56
DNN 6 layers	21.13	61.09	5.56	10.19
LSTM 5 layers	24.36	63.7	14.03	22.9
Ensemble ML	21.42	63.56	5.6	10.3
Ensemble DL	19.54	100	0.05	0.1

ra các bộ phân loại. Trong bảng kết quả mô hình LR có Accuracy cao nhất so với các mô hình còn lại, mô hình có hiệu suất thấp nhất là DT. Ta thấy được ra LR là mô hình có hiệu suất phân ljai tốt nhất so với các mô hình khác, trong khi đó DT là mô hình có hiệu suất phân loạiloại thấp nhất.

[RQ1]Take-away: Dựa vào kết quả ở bảng 4.4 , 4.5 và bảng 4.6 thì so với các bộ phân loại khác, đối với tập dữ liệu CICIDS2017 thì mô hình tổng hợp Ensemble Learning ML có hiệu suất vượt trội hơn cả, còn đối với tập dữ liệu CICIOT2023 Logistic Regression (LR) có hiệu suất vượt trội hơn ở tất cả các chỉ số quan trọng, bao gồm Accuracy, Precision, Recall, và F1-score.

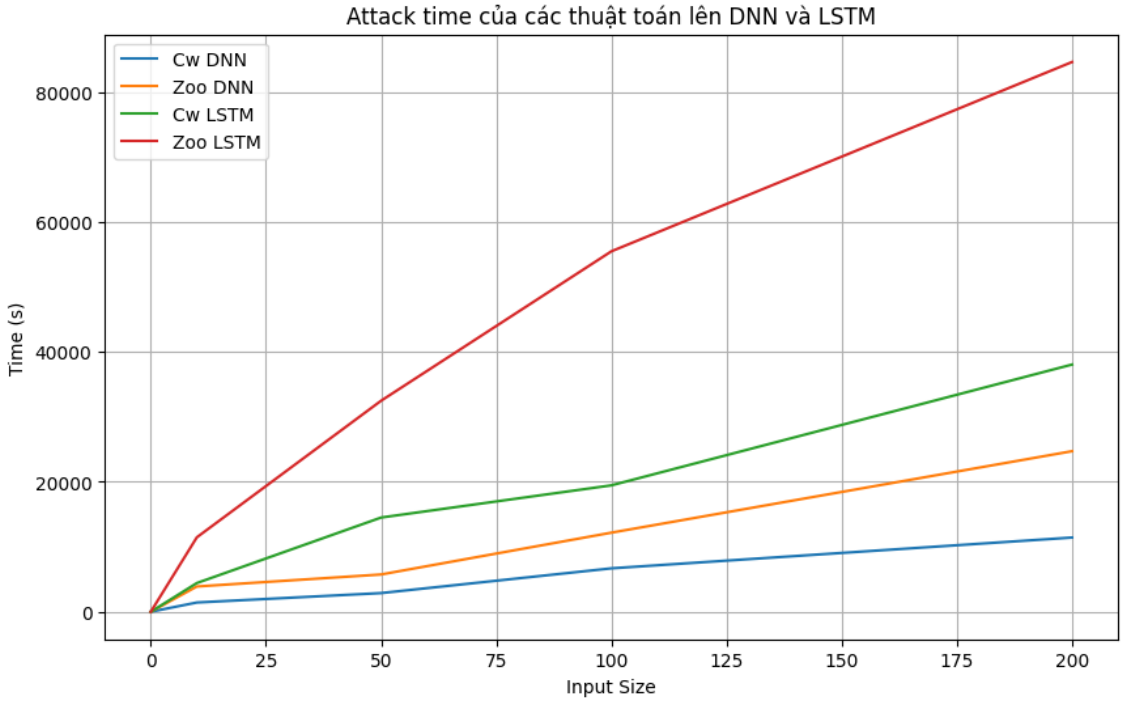
4.4.2. Tấn công đối kháng và Tranfersibility

Ở phần này, chúng tôi tiến hành triển khai các thuật toán tấn công đối kháng bao gồm GAN, FGSM, DEEPFOOL, ZOO VÀ C&W. Đánh giá chỉ số Detection rates trước khi bị tấn công đối kháng và sau khi tấn công đối kháng. Nhận xét về tính chuyển giao (Tranfersibility) lên các bộ phân loại. Bên cạnh đó, chúng tôi nhận xét các chỉ số ảnh hưởng tới mức độ của mẫu đối kháng. Cuối cùng, chúng tôi nhận xét về thời gian tạo ra mẫu đối kháng.

Lí do chọn DNN là mô hình bị tấn công đối kháng bởi các thuật toán đối kháng(FGSM, ZOO, CW và deepfool) (trừ việc Tranfersibility là phải tấn công toàn diện lên tất cả các bộ phân loại) cho việc đánh

giá mức độ của các thông số lên các bộ phân loại và cho phương pháp phòng thủ đối kháng là:

Đánh giá trên 2 mô hình là DNN Và LSTM. Biểu đồ 4.1 thì ta thấy được thời gian chạy DNN nhanh hơn gần gấp đôi so với LSTM và vẫn tạo ra mẫu đối kháng với mức độ đánh lừa tốt. Lí giải cho điều này là do LSTM có kiến trúc phức tạp hơn rất nhiều so với DNN. **Vì vậy đó là lí do chúng tôi chọn DNN là mô hình để tạo ra mẫu đối kháng và thử nghiệm .**



Hình 4.1: *Biểu đồ phân tán về thời gian tạo mẫu đối kháng trên 2 mô hình DNN và LSTM.*

4.4.2.1. Detection rate trước khi tấn công đối kháng

Bảng 4.7 mô tả các tỷ lệ phát hiện (detection rates) của các thuật toán detector dưới dữ liệu original của tập dữ liệu CICIDS2017. Các tỷ lệ này cho biết hiệu suất của mỗi bộ phân loại trong việc phát hiện (detection) trong dữ liệu gốc (original data). Đối với các thuật toán như Decision Tree, Logistic Regression, và các Ensemble ML (kết hợp), tỷ lệ phát hiện rất cao, đạt khoảng

Bảng 4.7: *Detection rates của các Detector dưới dữ liệu gốc của bộ dữ liệu CICIDS2017 (%).*

Detector	**Detection Rate**
GNB	98.07
DT	99.81
LR	92.93
DNN	91.35
LSTM	97.86
Ensemble ML	98.75
Ensemble DL	97.86

Bảng 4.8: *Tỉ lệ phát hiện tấn công của các Detector trước dữ liệu CICIOT2023 khi chưa biến đổi. (%).*

Detector	**Detection Rate**
GNB	5.4
DT	4.21
LR	13.59
DNN	5.56
LSTM	14.03
Ensemble ML	5.6
Ensemble DL	0.05

từ 98.88% đến 99.89%. Các mô hình Deep Learning như DNN và LSTM cũng có tỷ lệ khá cao, dao động từ 97.03% đến 98.39%.

Bảng 4.8 mô tả các tỷ lệ phát hiện tấn công (detection rates) của các bộ phân loại trước bộ dữ liệu CICIOT2023 khi chưa biến đổi đối kháng. Tỉ lệ phát hiện mẫu độc hại ở mô hình LSTM cho kết quả cao nhất là 14.03%, trong khi đó mô hình Ensemble DL cho kết quả thấp nhất là 0.05%. Các mô hình còn lại kết quả đạt được ở mức trung bình, dao động từ 4.21% đến 14.03%. Từ kết quả trên, bộ phân loại phát hiện các mẫu tấn công tốt nhất là LSTM, Ensemble DL là bộ phân loại có kết quả thấp nhất.

Bảng 4.9: *Tỉ lệ phát hiện tấn công của các Detector trước dữ liệu đối kháng được sinh ra từ* ***mô hình đề xuất GAN và Multimodal Learning*** *(%).*

Detector	**Detection Rate**
GNB	0
DT	0.29
LR	0
DNN	0
LSTM	0
Ensemble ML	0
Ensemble DL	0

Bảng 4.10: *Tỉ lệ phát hiện tấn công của các Detector trước dữ liệu đối kháng được sinh ra từ* ***mô hình đề xuất GAN*** *(%).*

Detector	**Detection Rate**
GNB	0
DT	3.04
LR	100
DNN	6.96
LSTM	51.77
Ensemble ML	7.18
Ensemble DL	0

4.4.2.2. Detection rate sau khi sử dụng mẫu đối kháng được tạo tấn công vào các Detector

Bảng 4.9 mô tả tỉ lệ phát hiện tấn công của các Detector trước dữ liệu đối kháng được sinh ra từ mô hình đề xuất GAN và Multimodal Learning. Kết quả cho thấy các Detector trừ DT đều đạt Detection Rate là 0% thấp hơn hoàn toàn so với Detection Rate của dữ liệu trước khi chưa biến đổi, trong khi mô hình DT đạt 0.29% cao hơn so với kết quả trước khi biến đổi là 3.92%. Từ kết quả ở bảng 4.9, nhóm chúng tôi thấy rằng các Detector đều có khả năng kháng mẫu đối kháng kém trước dữ liệu đối kháng được sinh ra từ mô hình GAN và Multimodal Learning.

Bảng 4.10 mô tả tỉ lệ phát hiện tấn công của các Detector trước dữ liệu đối kháng được sinh ra từ mô hình GAN. Kết quả cho thấy các Detector GNB,

Ensemble DL đều có Detection Rate là 0%, thấp hơn Detection Rate của dữ liệu trước khi chưa biến đổi. Trong khi đó mô hình LR có tỉ lệ phát hiện tốt nhất 100%, các mô hình còn lại dao động từ 3.04% đến 51.77%. Từ kết quả ở bảng 4.10, nhóm chúng tôi thấy răng, mô hình GNB, Ensemble DL có khả năng kháng đối kháng kém trước các mẫu đối kháng sử dụng GAN sinh ra, và các mô hình còn lại có khả năng nhận diện mẫu đối kháng tốt hơn.

Từ hai bảng 4.9 và 4.10, nhóm chúng tôi thấy rằng các mẫu đối kháng được sinh ra từ mô hình GAN và Multimodal Learning có khả năng trốn tránh các Detector cao hơn nhiều so với các mẫu đối kháng được sinh ra từ GAN. Từ đó chúng tôi nhận định rằng, việc áp dụng Multimodal Learning vào quá trình sinh mẫu đối kháng, đã giúp Generator sinh mẫu dữ liệu đối kháng, qua mặt các Detector tốt hơn so với việc chỉ áp dụng GAN để sinh mẫu đối kháng.

Bảng 4.11: *Tỉ lệ phát hiện tấn công của các Detector trước dữ liệu đối kháng trước các cuộc Adversarial attack.*

Thuật toán	FGSM	Deepfool	C&W	ZOO
Detector	Detection rates			
GNB	11.72	3.78	26.85	7.92
DT	0.00	45.98	0.73	16.34
LR	43.94	44.98	46.49	52.06
DNN	44.94	44.98	44.97	49.90
LSTM	29.51	1.78	1.25	0.50
Ensemble ML	11.81	35.52	29.96	14.80
Ensemble DL	23.06	23.38	23.11	25.20

Bảng 4.11 mô tả tỉ lệ phát hiện tấn công của các Detector trước dữ liệu đối kháng từ các cuộc Adversarial attack. Nhìn chung, các Detector có khả năng phát hiện tấn công đối kháng (Adversarial attack) khá thấp, chỉ đạt mức dưới 50% trong hầu hết các trường hợp. Điều này cho thấy các Detector hiện tại vẫn còn nhiều hạn chế trong việc phát hiện các cuộc tấn công đối kháng.

Trong các Detector đơn lẻ, DNN và LR đạt tỉ lệ phát hiện cao nhất, lên đến 52.06% đối với LR trong cuộc tấn công ZOO. Các Detector khác như GNB, DT và LSTM có tỉ lệ phát hiện rất thấp, chỉ khoảng 1-45%.

Khi kết hợp các Detector lại thành Ensemble, hiệu quả phát hiện tăng lên đáng kể, với Ensemble ML đạt 35.52% trong cuộc tấn công Deepfool và Ensemble DL đạt 25.20% trong cuộc tấn công ZOO. Điều này cho thấy việc kết hợp nhiều Detector sẽ giúp tăng cường khả năng phát hiện tấn công đối kháng.

Trong các cuộc tấn công, FGSM và ZOO thường khó bị phát hiện hơn so với C&W và Deepfool. Các Detector nói chung có khả năng phát hiện C&W và Deepfool tốt hơn so với FGSM và Zoo.

4.4.2.3. Tấn công tạo mẫu đối kháng

Đánh giá Tranfersibility

Trong phần này, chúng tôi còn tiến hành xây dựng và triển khai các thuật toán tấn công đối kháng là Fast Gradient Sign Method (FGSM), Zeroth Order Optimization (ZOO), Deepfool và Carlini & Wagner (C&W).

Sau đó chúng tôi tiến hành tấn công lên toàn bộ các bộ phân loại bao gồm GNB, DT, LR, DNN1 và DNN2, LSTM1 và LSTM2, Ensemble ML và Ensemble DL. Lấy dữ liệu đối kháng được tạo ra đem đi Tranfersibility lên các mô hình còn lại.

Lưu ý: Các thuật toán tấn công đối kháng FGSM, ZOO, Deepfool và C&W sẽ chạy ở các thông số mặc định để đảm bảo tính tổng quát trong Tranfersibility (FGSM với ϵ =0.01, 3 thuật toán tấn công đối kháng còn lại là max_iter = 10 (vòng lặp tối ưu hóa mẫu đối kháng)).

Ví dụ: Sử dụng các thuật toán tấn công đối kháng lên bộ phân loại GNB sẽ tạo ra được các mẫu đối kháng, sau đó đem chúng đi test trên các bộ phân loại (cả chính nó) để đánh giá mức độ nhận biết phân loại các mẫu đối kháng và khả năng chuyển giao (Tranfersibility) của mẫu đối kháng lên các mô hình còn lại.

Bảng 4.12: *Detection rate của các Detector trên dữ liệu đối kháng được tạo ra bởi thuật toán* ***FGSM****.*

Attack detector / Detector		GNB	DT	LR	DNN		LSTM		Ensemble ML	Ensemble DL
					4 unit	6 unit	3 unit	5 unit		
GNB		98.40	98.40	98.4	7.92	0.26	1.00	1.02	98.40	4.47
DT		99.60	99.60	99.60	1.33	0,72	21.66	13.95	99.60	7.64
LR		79.80	79.80	79.80	37.29	42.29	52.49	51.64	79.80	44.46
DNN	4 unit	96.35	96.35	96.35	44.94	44.95	45.30	45.38	96.35	45.16
	6 unit	91.20	91.20	91.20	44.04	43.93	44.56	44.56	91.20	44.30
LSTM	3 unit	96.85	96.85	96.85	37.89	38.71	35.22	40.73	96.85	39.31
	5 unit	95.40	95.40	95.40	1.17	0.14	0.11	1.14	95.40	1.16
Ensemble ML		98.55	98.55	98.55	8.11	2.46	5.77	4.13	98.55	6.12
Ensemble DL		95.43	95.43	95.43	23.06	22.54	22.71	23.26	95.43	23.16

Bảng 4.13: *Detection rate của các Detector trên dữ liệu đối kháng được tạo ra bởi thuật toán* ***Deepfool****.*

Attack detector / Detector		GNB	DT	LR	DNN		LSTM		Ensemble ML	Ensemble DL
					4 unit	6 unit	3 unit	5 unit		
GNB		98.40	98.40	98.4	3.78	0.14	0.012	0.012	98.40	1.89
DT		99.60	99.60	99.60	23.75	35.77	19.35	19.35	99.60	21.55
LR		79.80	79.80	79.80	45.98	45.98	49.7	49.7	79.80	47.84
DNN	4 unit	96.35	96.35	96.35	44.98	44.98	49.15	49.15	96.35	47.07
	6 unit	91.20	91.20	91.20	44.98	44.98	44.35	44.35	91.20	44.67
LSTM	3 unit	96.85	96.85	96.85	21.68	18.09	41.90	41.90	96.85	31.79
	5 unit	95.40	95.40	95.40	1.78	0.48	0.05	0.05	95.40	0.91
Ensemble ML		98.55	98.55	98.55	35.52	20.28	20.80	20.80	98.55	19.16
Ensemble DL		95.43	95.43	95.43	23.38	22.73	24.6	22.73	95.43	24.6

Bảng 4.14: *Detection rate của các Detector trên dữ liệu đối kháng được tạo ra bởi thuật toán* ***Zoo****.*

Attack detector / Detector		GNB	DT	LR	DNN		LSTM		Ensemble ML	Ensemble DL
					4 unit	6 unit	3 unit	5 unit		
GNB		98.40	98.40	98.40	7.92	7.84	2.43	2.43	98.40	5.17
DT		99.60	99.60	99.60	16.33	17.20	18.40	18.25	99.60	17.29
LR		79.80	79.80	79.80	52.06	48.30	46.05	47.35	79.80	49.70
DNN	4 unit	96.35	96.35	96.35	49.90	48.60	45.70	45.65	96.35	47.77
	6 unit	91.20	91.20	91.20	46.00	46.40	44.00	44.10	91.20	45.05
LSTM	3 unit	96.85	96.85	96.85	20.50	21.60	20.95	20.30	96.85	20.40
	5 unit	95.40	95.40	95.40	0.50	0.0003	0.15	0.3	95.40	0.40
Ensemble ML		98.55	98.55	98.55	14.80	14.70	5.40	6.35	98.55	10.57
Ensemble DL		95.43	95.43	95.43	25.20	24.30	22.92	22.97	95.43	24.08

Bảng 4.15: *Detection rate của các Detector trên dữ liệu đối kháng được tạo ra bởi thuật toán* ***Carlini and Wagner****.*

Attack detector / Detector		GNB	DT	LR	DNN		LSTM		Ensemble ML	Ensemble DL
					4 unit	6 unit	3 unit	5 unit		
GNB		98.40	98.40	98.40	26.84	26.84	28.50	28.50	98.40	27.67
DT		99.60	99.60	99.60	0.73	0.73	0.75	0.75	99.60	0.74
LR		79.80	79.80	79.80	46.49	46.49	47.40	47.40	79.80	46.94
DNN	4 unit	96.35	96.35	96.35	44.97	44.97	45.35	45.35	96.35	45.16
	6 unit	91.20	91.20	91.20	44.25	44.25	44.80	44.80	91.20	44.52
LSTM	3 unit	96.85	96.85	96.85	41.16	41.16	41.60	41.60	96.85	41.38
	5 unit	95.40	95.40	95.40	1.25	1.24	1.55	1.55	95.40	1.40
Ensemble ML		98.55	98.55	98.55	29.96	29.96	31.95	31.95	98.55	30.95
Ensemble DL		95.43	95.43	95.43	23.11	23.10	23.45	23.45	95.43	23.28

Bảng 4.12, 4.13, 4.14 và 4.15 lần lượt là các bảng thống kê Detection Rate có tính chuyển giao lên các mô hình bởi các thuật toán tấn công là FGSM, Deepfool, ZOO và C&W.

Nhìn qua, ta thấy được các thuật toán tấn công mẫu đối kháng này không có khả năng tạo mẫu đối kháng trên các mô hình học máy (GNB,DT, LR) và mô hình tổng hợp Ensemble ML và khả năng chuyển giao của nó bằng 0.

Lí giải vì điều này vì các mô hình học máy đang xét có bản chất khác biệt mà các thuật toán tấn công tạo mẫu đối kháng cần thiết để có thể tạo ra mẫu đối kháng. Các lí do như là:

- Thiếu Gradient: Gaussian Naive Bayes (GNB): Không có hàm mất mát và gradient như các mạng nơ-ron. Nó dựa trên xác suất có điều kiện để phân loại.Decision Trees (DT): Không có gradient vì nó dựa trên các ngưỡng và nhánh cây quyết định

- Cấu trúc và quyết định không liên tục: Decision Trees (DT) có cấu trúc không liên tục với các quyết định nhị phân tại mỗi nút. Điều này làm cho các thay đổi nhỏ trong dữ liệu đầu vào không nhất thiết dẫn đến thay đổi lớn trong kết quả, khó khăn trong việc tạo ra các mẫu đối kháng hiệu quả

- ZOO và C&W dựa trên các chiến lược tối ưu hóa và khai thác gradient dự đoán mà không phù hợp hoặc không thể áp dụng đối với các mô hình không

có gradient hoặc có cấu trúc đơn giản như GNB và DT.

Còn đối với các mô hình học sâu (Deep Learning) thì các thuật toán này có thể tạo ra lưu lượng đối kháng và có tính chuyển giao lên cùng một mô hình và chuyển giao lên khác mô hình.

Tranfersibility trên cùng một mô hình

Bảng 4.16: *Detection rates của các bộ phân loại khi chuyển giao trên cùng mô hình LSTM (%).*

Thuật toán	**FGSM**		**Deepfool**		**CW**		**ZOO**	
Attack detector / Detector	LSTM1 (3 unit)	LSTM2 (5 unit)	LSTM1 (3 unit)	LSTM2 (5 unit)	LSTM1 (3 unit)	LSTM2 (5 unit)	LSTM (3 unit)	LSTM2 (5 unit)
LSTM1 (3 unit)	35.22	40.73	41.90	41.90	20.95	20.30	41.60	41.60
LSTM2 (5 unit)	0.11	1.14	0.05	0.05	0.15	0.30	1.55	1.55

Bảng 4.17: *Detection rates của các bộ phân loại khi chuyển giao trên cùng mô hình DNN (%).*

Thuật toán	**FGSM**		**Deepfool**		**CW**		**ZOO**	
Attackdetector / Detector	DNN1 (4 unit)	DNN2 (6 unit)	DNN1 (4 unit)	DNN2 (6 unit)	DNN1 (4 unit)	DNN2 (6 unit)	DNN1 (4 unit)	DNN2 (6 unit)
DNN1 (4 unit)	44.94	44.95	44.98	44.98	49.9	48.60	44.97	44.97
DNN2 (6 unit)	44.04	43.93	44.98	44.98	46.00	46.40	44.25	44.25

Từ bảng 4.17 và 4.16 cho ta thấy được khả năng chuyển giao của mẫu đối kháng lên cùng các mô hình. Cụ thể:

- LSTM: Thấy được khi tấn công và chuyển giao lên cùng một mô hình.Tấn công FGSM khi chọn LSTM1 (3 unit) làm Attack detector thì mẫu đối kháng được tạo ra tác động mạnh mẽ lên Detection rates của mô hình LSTM2 (5 unit) chỉ còn 0.11%. Nhưng khi làm ngược lại thì mô hình LSTM1(3 unit) lại có chỉ số Detection Rates là 40.73% cao hơn LSTM2(5 unit) là 1.14%.

- Còn DNN thì khi chuyển giao lên cùng một mô hình thì kết quả cho ra có sự lệch nhẹ giữa việc chuyển giao mẫu đối kháng giữa mô hình đơn giản hơn qua mô hình phức tạp hơn.

Nhận xét: Các mô hình phức tạp hơn thì khi bị tấn công đối kháng và chuyển giao lên cùng một mô hình, còn tùy vào thuật toán và mô hình của bộ phân loại thì mẫu đối kháng được tạo ra chưa chắc có khả năng trốn tránh cao hơn so với mẫu đối kháng được tạo ra từ mô hình đơn giản hơn và khả năng kháng đối kháng của nó cũng vậy.

Tranfersibility khác mô hình

Mẫu đối kháng được tạo ra có khả năng chuyển giao sang các mô hình khác. Như ở bảng 4.14, mẫu đối kháng được tạo ra bởi thuật toán ZOO khi tấn công vào mô hình DNN2(6unit) làm cho các bộ phân loại giảm tỉ lệ phát hiện đi rất nhiều như là GNB(7.92%), LSTM(0.0003%). Mẫu đối kháng được tạo ra từ mô hình phức tạp hơn thì nó có khả năng trốn tránh cao hơn so với mẫu đối kháng được tạo ra từ mô hình ít phức tạp hơn. Như trong bảng 4.12, mẫu đối kháng từ DNN1(4unit) thì bộ phân loại GNB chỉ phát hiện được 7.92% còn mẫu đối kháng từ DNN2 (6unit) thì là 0.26%. Hoặc là mẫu đối kháng DNN1(4 unit) thì bộ phân loại LR tỉ lệ phát hiện là 52.06% còn đối với mẫu đối kháng từ DNN2 (6unit) thì là 48.3% ở bảng 4.14.

Mô hình phức tạp hơn thì khả năng nhận biết mẫu đối kháng cao hơn so với mô hình đơn giản hơn. Như ở bảng 4.13 , tỉ lệ phát hiện của mô hình GNB chỉ là 0.14% còn tỉ lệ phát hiện của DNN2(6 unit) là 44.98% trên các mẫu đối kháng được tạo ra từ mô hình DNN2(6 unit). Hay ở bảng 4.15, tỉ lệ phát hiện của mô hình DT chỉ là 0.75% còn tỉ lệ phát hiện của LSTM1(3 unit) là 41.6% trên các mẫu đối kháng được tạo ra từ mô hình LSTM2(5 unit).

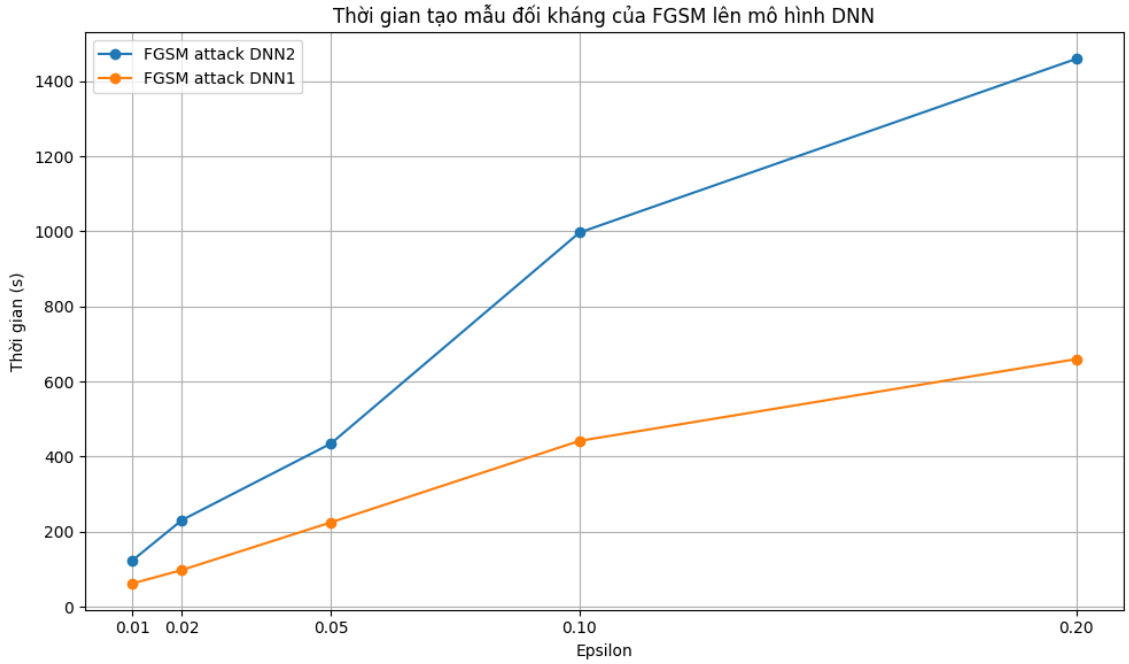
Nhận xét: Mẫu đối kháng được tạo ra có khả năng chuyển giao sang các mô hình khác. Tùy thuộc vào độ phức tạp của mô hình, thuật toán tấn công và những phụ thuộc khác thì mẫu đối kháng được tạo ra từ những mô hình có độ phức tạp cao hơn có xu hướng trốn tránh cao hơn so với những mẫu đối kháng

được tạo ra từ những mô hình có độ phức tạp thấp hơn khi bị tấn công đối kháng. Khả năng nhận biết các mẫu đối kháng cũng vậy, tùy thuộc vào mô hình xét và độ phức tạp của nó và các phụ thuộc khác thì những mô hình có độ phức tạp cao hơn có khả năng nhận biết các mẫu đối kháng cao hơn so với những mô hình có độ phức tạp thấp hơn.

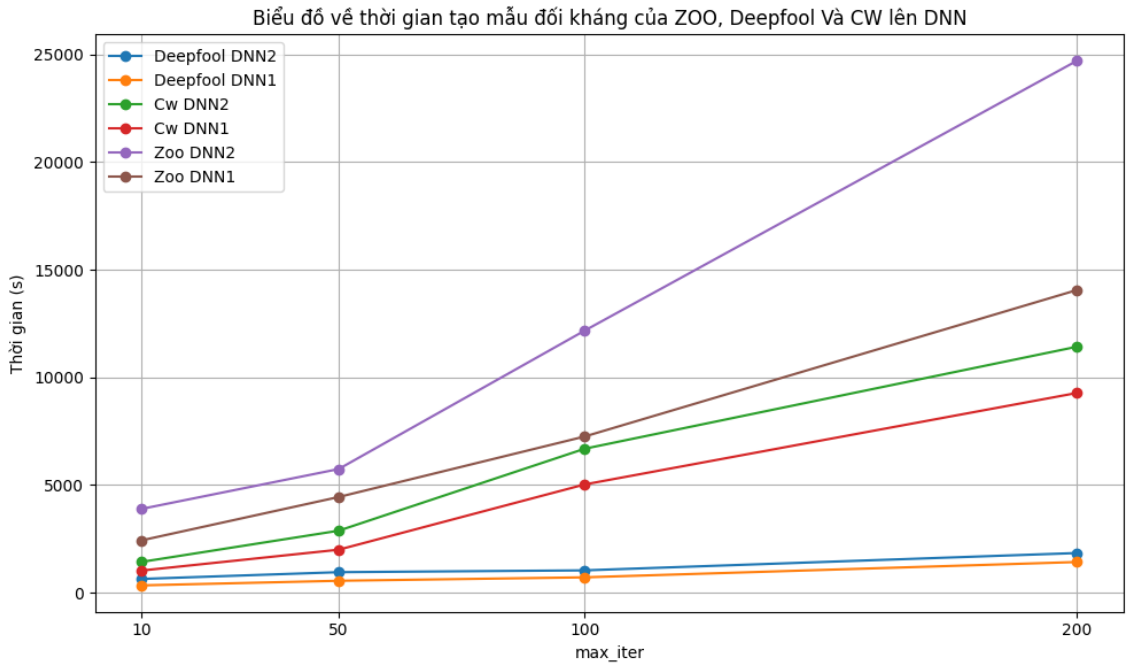
[**RQ3**]**Take-away:**Tùy vào thuật toán tạo mẫu đối kháng, kiến trúc của các mô hình đang xét và những yếu tố phụ thuộc khác, thì mô hình càng phức tạp hơn có thể tạo ra những mẫu đối kháng có khả năng trốn tránh cao hơn những mẫu đối kháng được tạo ra từ mô hình đơn giản hơn.)

[**RQ4**]**Take-away:**Tùy vào thuật toán tạo mẫu đối kháng, kiến trúc của các mô hình đang xét và những yếu tố phụ thuộc khác, thì mô hình càng phức tạp hơn có khả năng nhận biết những mẫu đối kháng cao hơn so với những mô hình có kiến trúc đơn giản hơn.)

Thời gian tạo mẫu đối kháng



Hình 4.2: *Biểu đồ phân tán về thời gian tạo mẫu đối kháng và Epsilon của FGSM attack lên hai mô hình DNN1 và DNN2.*



Hình 4.3: *Biểu đồ phân tán về thời gian tạo mẫu đối kháng và vòng lặp tối ưu với 3 thuật toán là Deepfool, C&W và Zoo lên 2 mô hình DNN1 VÀ DNN2.*

- Trên cùng một mô hình với kiến trúc khác nhau: Dựa vào 2 biểu đồ là 4.2 và 4.3 thì ta thấy được thời gian sẽ tăng tuyến tính theo độ phức tạp của mô hình (DNN2 có kiến trúc phức tạp hơn DNN1) và thời gian tạo mẫu đối kháng của các thuật toán WB (FGSM, Deepfool, C&W) nhanh hơn so với thời gian tạo mẫu đối kháng của các thuật toán tấn công BB(ZOO).

- Trên khác mô hình: Đánh giá trên 2 mô hình là DNN Và LSTM. Biểu đồ 4.1 thì ta thấy được thời gian chạy DNN nhanh hơn gần gấp đôi so với LSTM và vẫn tạo ra mẫu đối kháng với mức độ đánh lừa tốt. Lí giải cho điều này là do LSTM có kiến trúc phức tạp hơn rất nhiều so với DNN.

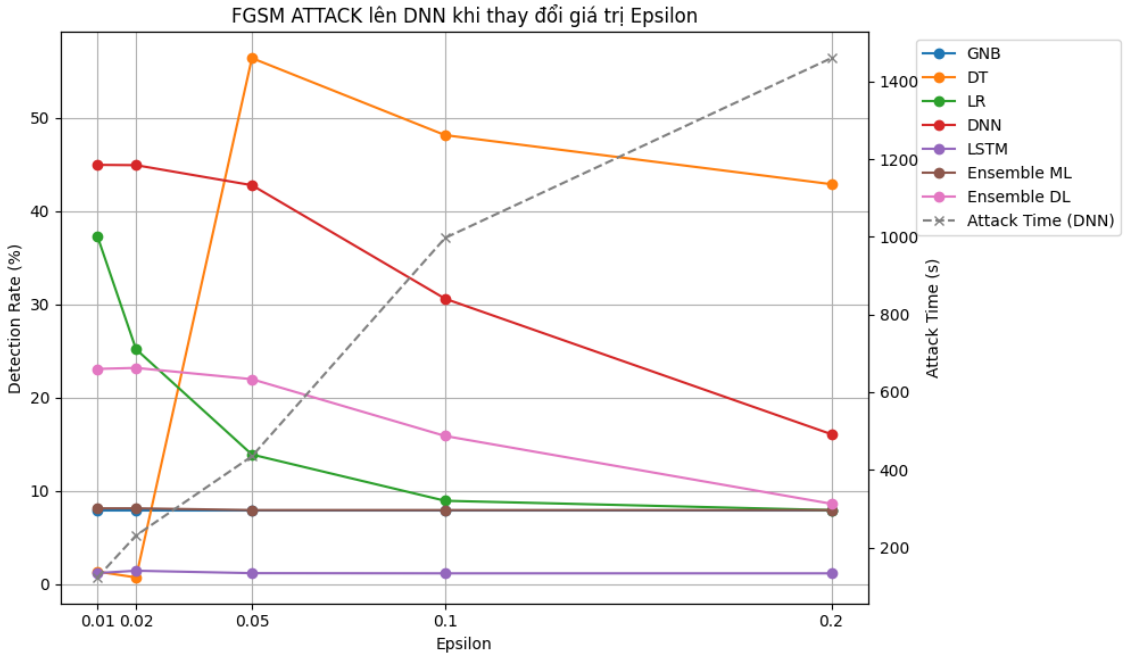
Nhận xét: Tùy vào mô hình bị tấn công và thuật toán tấn công, thời gian cần thiết để tạo ra mẫu đối kháng sẽ tăng tuyến tính theo độ phức tạp của cả mô hình và thuật toán đối kháng.

Đánh giá thông số phụ thuộc của thuật toán tạo mẫu đối kháng lên các bộ phân loại

Ở phần thử nghiệm này, chúng tôi tiến hành đánh giá các mẫu đối kháng được tạo ra mô hình DNN.

Bảng 4.18: *Detection rates của các Detector với dữ liệu đối kháng được tạo ra bởi thuật toán* ***FGSM*** *khi thay đổi giá trị của* ϵ.

ϵ	0.01	0.02	0.05	0.1	0.2
GNB	7.92	7.92	7.92	7.92	7.92
DT	1.33	0.70	56.39	48.13	42.87
LR	37.29	25.17	13.88	8.92	7.92
DNN	44.94	44.91	42.76	30.59	16.06
LSTM	1.17	1.42	1.16	1.14	1.14
Ensemble ML	8.11	8.11	7.92	7.92	7.92
Ensemble DL	23.06	23.17	21.96	15.87	8.60



Hình 4.4: *Biểu đồ thay đổi epsilon tạo ra mẫu đối kháng tác động lên các bộ phân loại và thời gian tạo mẫu đối kháng của thuật toán FGSM lên mô hình DNN.*

Bảng 4.18 cho thấy rõ sự khác biệt về khả năng phát hiện dữ liệu đối kháng giữa các thuật toán detector khi thay đổi giá trị của tham số ϵ trong thuật toán FGSM. Cụ thể hơn Hình 4.4 biểu diễn tỉ lệ phát hiện của các mô hình khi ϵ thay

đổi và thời gian tạo ra các mẫu đối kháng dựa trên epsilon. Cụ thể

- Epsilon ϵ là một tham số quan trọng trong FGSM (Fast Gradient Sign Method), biểu thị mức độ nhiễu được thêm vào dữ liệu đầu vào. Khi epsilon càng lớn, mức độ nhiễu càng cao, làm cho tấn công càng mạnh mẽ và dễ gây ra sai lệch lớn hơn trong kết quả dự đoán của mô hình. Tuy nhiên nếu epsilon gần đạt tới cực đại, mẫu đối kháng sẽ mất đi giá trị của nó (phân kì).

- Tỷ lệ phát hiện không thay đổi: GNB giả định rằng các đặc tính độc lập với nhau và có phân phối chuẩn (Gaussian). Do mô hình này dựa trên xác suất và thường được đơn giản hóa, việc thêm nhiễu mạnh hay nhẹ vào các mẫu đối kháng không ảnh hưởng đáng kể đến các xác suất đã tính toán. Điều này dẫn đến tỷ lệ phát hiện các mẫu đối kháng của mô hình không thay đổi và ở mức thấp (luôn ở mức 7.92%) bất kể epsilon.

- Tỷ lệ phát hiện biến động mạnh: Decision Tree chia dữ liệu dựa trên các ngưỡng của các đặc tính. Khi epsilon nhỏ, mẫu đối kháng làm cho mô hình không vượt qua các ngưỡng, dẫn đến tỷ lệ phát hiện giảm (khi epsilon ở 0.01 và 0.02). Khi epsilon tăng, mẫu đối kháng bị biến đổi đi làm thay đổi đáng kể các đặc tính, đẩy chúng qua các ngưỡng quyết định, từ đó tăng tỷ lệ phát hiện của mô hình trước các mẫu đối kháng.

- Tỷ lệ phát hiện giảm dần: Logistic Regression dựa trên mô hình tuyến tính để xác định xác suất của các lớp. Khi epsilon tăng, nhiễu làm thay đổi các giá trị của đặc tính của mẫu đối kháng, dẫn đến việc các mẫu đối kháng bị chuyển dịch và sai lệch khỏi ranh giới quyết định, làm giảm tỷ lệ phát hiện của mô hình trước các mẫu đối kháng.

- Tỷ lệ phát hiện giảm dần:DNN mạng nơ-ron sâu có cấu trúc phức tạp và phụ thuộc mạnh vào dữ liệu đầu vào. Khi epsilon tăng, nhiễu làm thay đổi đáng kể các đầu vào của các lớp nơ-ron, dẫn đến sai lệch trong kết quả dự

đoán và giảm tỷ lệ phát hiện của mô hình trước các mẫu đối kháng.

- Tỉ lệ phát hiện ở mức rất thấp (khoảng 1% đến 2%): LSTM là mạng noron học theo mô hình tuần tự, mặc dù nó có thể lưu trữ dữ liệu trong thời gian dài tuy nhiên nếu nhưng đối với các mẫu dữ liệu bị biến đổi một cách tối ưu bởi các kỹ thuật tấn công như FGSM, các biến thể này có thể không được nhận diện do sự khác biệt trong các đặc trưng quan trọng.Vì vậy, mô hình không nhận biết và phân biệt được các mẫu đối kháng.

- Tỷ lệ phát hiện ở mức thấp: Mô hình kết hợp học máy (Ensemble ML) thường sử dụng nhiều mô hình học máy khác nhau và kết hợp kết quả của chúng. Điều này giúp mô hình trở nên ổn định hơn và ít bị ảnh hưởng bởi nhiễu, tuy nhiên nó kết hợp từ các mô hình trên, nếu mô hình con tổng hợp nên Ensemble ML bị đánh lừa sẽ dẫn tới kết quả chung của mô hình.

- Tỷ lệ phát hiện giảm dần: Mô hình kết hợp học sâu (Ensemble DL) cũng sử dụng nhiều mô hình học sâu khác nhau (DNN và LSTM). Mặc dù việc kết hợp này có thể giúp cải thiện độ chính xác, nhưng các mô hình học sâu vẫn nhạy cảm với nhiễu, dẫn đến tỷ lệ phát hiện giảm dần khi epsilon tăng.

- Cuối cùng, thời gian tạo mẫu đối kháng tăng khi epsilon tăng là vì:

  – Số lần lặp tăng: Khi epsilon tăng, độ lớn của lỗi (perturbation) được áp dụng lên dữ liệu đầu vào cũng tăng lên. Để tránh mẫu đối kháng bị lệch đi với độ lớn quá nhiều (phân kì), thì số lần lặp phải tăng lên để điều chỉnh.

  – Độ phức tạp tính toán: Khi epsilon tăng, quá trình tối ưu hóa (optimization process) để tạo ra các lỗi nhỏ (perturbations) có thể trở nên phức tạp hơn. Điều này có thể đòi hỏi nhiều tài nguyên tính toán hơn từ hệ thống, dẫn đến thời gian tính toán tăng lên.

Nhận xét: Từ biểu đồ số liệu Hình 4.4, ta thấy được khi điều chỉnh độ mạnh của epsilon trong thuật toán FGSM để tạo ra và thêm vào mẫu đối kháng, tùy

vào mô hình mà nó sẽ có những kết quả khác nhau, có mô hình bị giảm đi tỉ lệ phát hiện(Detection Rates) như là (LR hay DNN), hoặc mô hình chỉ bị đánh lừa ở những mẫu đối kháng với những biến đổi nhỏ, biến đổi quá lớn thì tác dụng bị yếu đi (DT) và thời gian tạo ra mẫu đối kháng tăng tuyến tính theo epsilon.

Tiếp theo, chúng tôi sẽ nhận xét trên thuật toán Deepfool.

Bảng 4.19 cho thấy rõ sự khác biệt về khả năng phát hiện dữ liệu đối kháng giữa các thuật toán detector khi thay đổi giá trị của tham số max_*iter* trong thuật toán Deepfool. Cụ thể hơn Hình 4.5 biểu diễn tỉ lệ phát hiện của các mô hình khi giá trị vòng lặp tối ưu thay đổi và thời gian tạo ra các mẫu đối kháng dựa trên vòng lặp đó.

Bảng 4.19: *Detection rates của các Detector với dữ liệu đối kháng được tạo ra bởi thuật toán* ***Deepfool*** *khi thay đổi giá trị tối ưu vòng lặp (max_ iter)*

max_iter	**10**	**50**	**100**	**200**
GNB	3.78	7.92	7.92	7.92
DT	45.98	14.65	10.13	13.47
LR	44.98	43.02	42.75	41.82
DNN	44.98	46.87	44.95	52.70
LSTM	1.78	0.034	0.36	0.013
Ensemble ML	35.52	3.012	2.846	2.37
Ensemble DL	23.38	23.45	22.66	26.36

Nhận xét rằng:

- GNB tỉ lệ phát hiện rất thấp .Detection rates từ 3.78% khi `max_iter=10` lên 7.92% khi `max_iter` $\geq$ 50. Điều này cho thấy khi thuật toán Deepfool chạy nhiều vòng lặp hơn, nó tạo ra các mẫu đối kháng khó phát hiện hơn đối với GNB. Tuy nhiên, một số mẫu có thể vượt quá ngưỡng hội tụ và trở thành phân kì trong bộ phân loại - tức là quá khác biệt so với mẫu gốc, làm mất đi khả năng đối kháng lên mô hình GNB.

- Độ chính xác phát hiện của DT giảm mạnh từ 45.98% khi `max_iter=10` xuống 10.13% khi `max_iter=100`, sau đó tăng lên 13.47% khi `max_iter=200`.

Điều này chỉ ra rằng khi tăng số vòng lặp `max_iter`, DT càng khó phát hiện được các mẫu đối kháng, nhưng khi `max_iter=200` thì độ chính xác phát hiện đã cải thiện hơn do một số mẫu đã vượt qua ngưỡng nhất định và mô hình có thể nhận biết được.

- Độ phát hiện chính xác của LR giảm dần theo mức độ tăng của biến max_iter tức là mẫu đối kháng có khả năng trốn tránh tốt lên mô hình này.

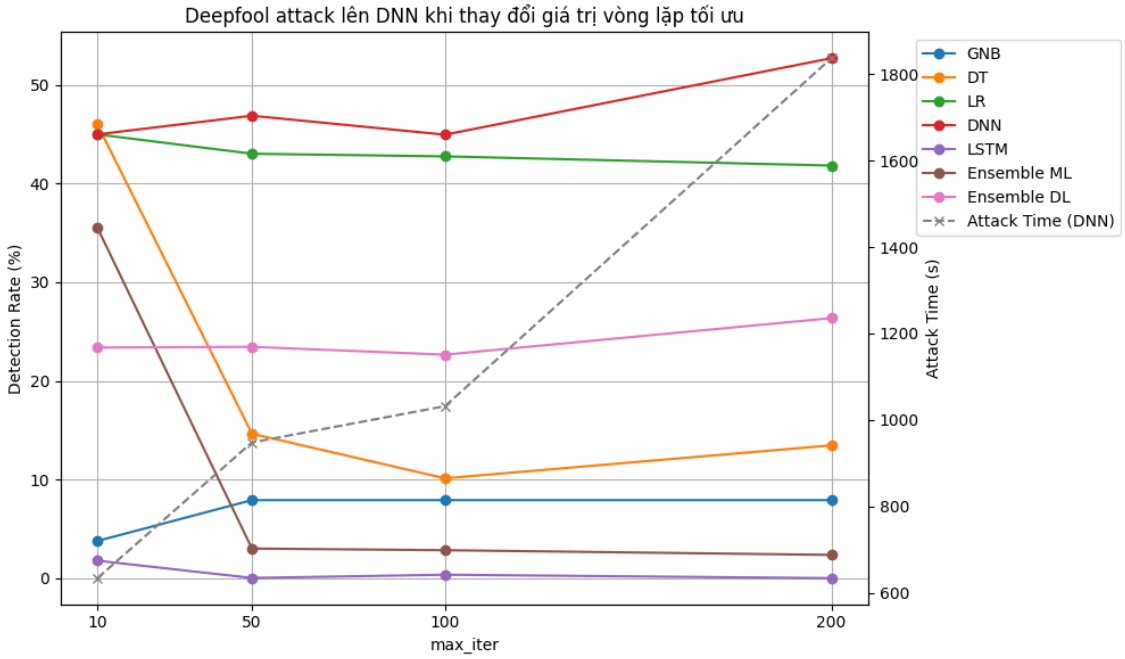
- DNN) và LSTM: Độ chính xác phát hiện của DNN tăng từ 44.98% lên 52.70% khi tăng max_iter, trong khi LSTM lại giảm từ 1.78% xuống 0.013%. Điều này cho thấy các mẫu đối kháng tạo ra bởi Deepfool càng khó phát hiện đối với bộ phân loại LSTM khi tăng số vòng lặp, nhưng bộ phân loại DNN lại có thể phát hiện được tốt hơn.

- Ensemble ML :Độ chính xác phát hiện giảm mạnh từ 35.52% xuống 2.37% khi tăng max_iter, cho thấy các mẫu đối kháng do Deepfool tạo ra ngày càng khó phát hiện đối với mô hình tổ hợp này. Lí giải cho điều này là tại vì các mô hình con tổng hợp phân loại sai các mẫu đối kháng.

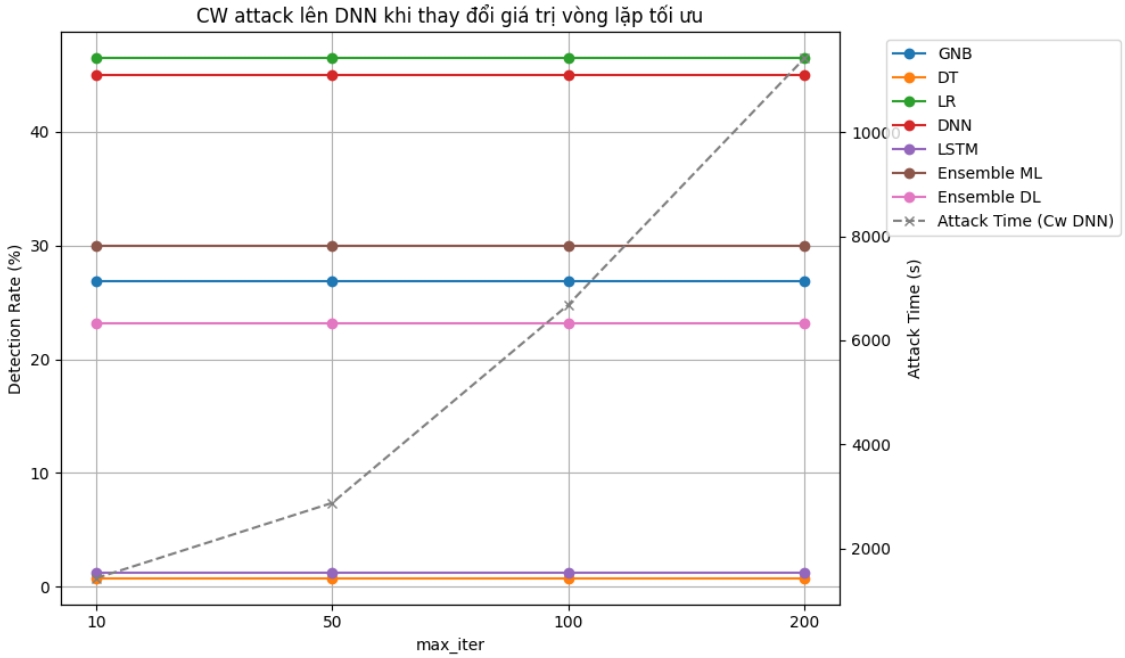
- Ensemble DL: Độ chính xác phát hiện giữ ổn định ở mức 22-26% khi tăng max_iter, cho thấy mô hình tổ hợp này ổn định hơn so với tổ hợp học máy.

- Thời gian tạo mẫu đối kháng tăng tuyến tính theo số lượng vòng lặp tạo ra.

Nhận xét: Từ biểu đồ số liệu Hình 4.5, ta thấy được khi điều chỉnh số lần lặp trong thuật toán Deepfool để tạo ra các mẫu đối kháng, tùy vào mô hình mà nó sẽ có những kết quả khác nhau và mẫu đối kháng có khả năng trốn tránh cực đại khi biến max_iter ở vùng lân cận 100 và thời gian tạo mẫu đối kháng tăng tuyến tính theo biến max_iter.



Hình 4.5: *Biểu đồ thay đổi vòng lặp tối ưu tạo ra mẫu đối kháng tác động lên các bộ phân loại và thời gian tạo mẫu đối kháng của thuật toán Deepfool lên mô hình DNN.*



Hình 4.6: *Biểu đồ thay đổi giá trị vòng lặp tối ưu tạo ra mẫu đối kháng tác động lên các bộ phân loại và thời gian tạo mẫu đối kháng của thuật toán CW lên mô hình DNN.*

Bảng 4.20 cho thấy rõ sự khác biệt về khả năng phát hiện dữ liệu đối kháng giữa các thuật toán detector khi thay đổi giá trị của tham số max _*iter* trong thuật toán C&W. Cụ thể hơn Hình 4.6 biểu diễn tỉ lệ phát hiện của các mô hình khi giá trị vòng lặp tối ưu thay đổi và thời gian tạo ra các mẫu đối kháng dựa trên vòng lặp đó.

- Khi tăng số vòng lặp tối ưu hóa từ 10 lên 200 nhưng không thấy mẫu đối kháng làm suy yếu mô hình, điều này cho thấy rằng thuật toán đã tìm được một mẫu đối kháng hiệu quả từ những lần lặp ban đầu. Mẫu đối kháng này có thể đã gần với mẫu đầu vào gốc nhất có thể để đánh lừa mô hình ở mức độ cao nhất. Sau khi tối ưu hóa này đã đạt giải pháp tối ưu, việc tăng số vòng lặp không cải thiện thêm nữa vì không có cách nào để cải thiện mẫu đối kháng đã tối ưu hóa

- Thời gian tạo mẫu đối kháng bởi thuật toán C&W tăng tuyến tính theo giá trị vòng lặp tối ưu. Điều này gợi ý rằng nếu cần tăng số lần lặp để tìm ra một mẫu đối kháng tốt hơn, thì thời gian để tính toán cũng sẽ tăng theo tỷ lệ tương đối.

Nhận xét: Từ biểu đồ số liệu Hình 4.6, ta thấy được khi điều chỉnh số lần lặp trong thuật toán C&W để tạo ra các mẫu đối kháng, mẫu đối kháng sẽ đạt mạnh mẽ nhất tùy vào lúc thuật toán C&W tìm ra được mẫu đối kháng gần nhất với mẫu gốc nếu như nó đã tìm được ở những vòng lặp đầu, việc tăng số vòng lặp không cải thiện thêm nữa vì không có cách nào để cải thiện mẫu đối kháng đã đạt sức mạnh cực đại.

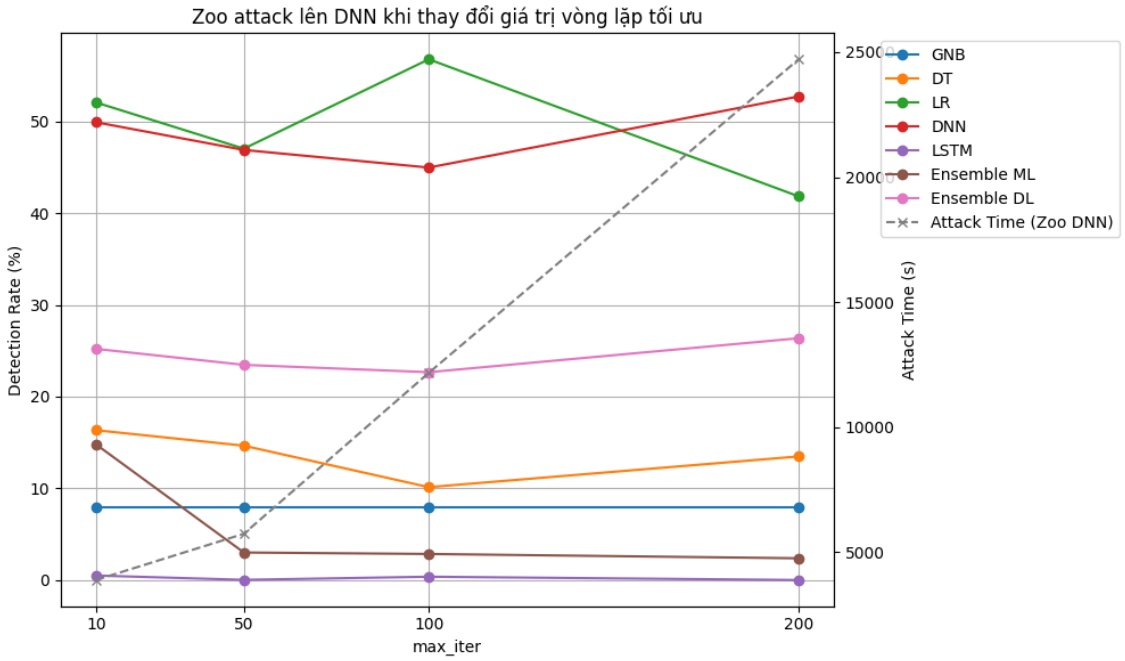
Bảng 4.21 cho thấy rõ sự khác biệt về khả năng phát hiện dữ liệu đối kháng giữa các thuật toán detector khi thay đổi giá trị của tham số max _*iter* trong thuật toán C&W. Cụ thể hơn Hình 4.7 biểu diễn tỉ lệ phát hiện của các mô hình khi giá trị vòng lặp tối ưu thay đổi và thời gian tạo ra các mẫu đối kháng dựa trên vòng lặp đó.

Bảng 4.20: *Detection rates của các Detector với dữ liệu đối kháng được tạo ra bởi thuật toán* ***Carlini and Wagner*** *khi thay đổi giá trị tối ưu vòng lặp (max_iter).*

max_iter	**10**	**50**	**100**	**200**
GNB	26.85	26.85	26.85	26.85
DT	0.73	0.73	0.73	0.73
LR	46.49	46.49	46.49	46.49
DNN	44.97	44.97	44.97	44.97
LSTM	1.25	1.25	1.25	1.25
Ensemble ML	29.96	29.96	29.96	29.96
Ensemble DL	23.11	23.11	23.11	23.11

Bảng 4.21: *Detection rates của các Detector với dữ liệu đối kháng được tạo ra bởi thuật toán* ***ZOO*** *khi thay đổi giá trị tối ưu vòng lặp (max_iter).*

max_iter	**10**	**50**	**100**	**200**
GNB	7.92	7.92	7.92	7.92
DT	16.34	14.65	10.13	13.48
LR	52.06	47.02	56.76	41.82
DNN	49.90	46.88	44.96	52.71
LSTM	0.50	0.03	0.37	0.01
Ensemble ML	14.80	3.01	2.85	2.38
Ensemble DL	25.20	23.46	22.66	26.36



Hình 4.7: *Biểu đồ thay đổi vòng lặp tối ưu tạo ra mẫu đối kháng tác động lên các bộ phân loại và thời gian tạo mẫu đối kháng của thuật toán Zoo lên mô hình DNN.*

Cụ thể:

- **GNB (Gaussian Naive Bayes)**: Tỉ lệ phát hiện (detection rate) ở mức khoảng 7.92% không thay đổi khi thay đổi max_iter, cho thấy mô hình này không nhạy cảm với sự thay đổi này.
- **DT (Decision Tree)**: Tỉ lệ phát hiện ban đầu là cao nhất ở max_iter = 10 (16.34%) nhưng giảm dần khi tăng max_iter lên 100 và 200. Điều này có thể cho thấy rằng việc tăng số lần lặp có thể dẫn đến một dạng dữ liệu đối kháng mà mô hình Decision Tree khó khăn hơn để phát hiện.
- **LR (Logistic Regression)**: Tỉ lệ phát hiện dao động vừa khi thay đổi max_iter, nhưng không có xu hướng rõ rệt. Điều này có thể cho thấy rằng mô hình LR có thể không quá nhạy cảm với sự thay đổi này.
- **DNN (Deep Neural Network)**: Tỉ lệ phát hiện dao động nhẹ khi thay đổi max_iter, nhưng không có xu hướng rõ rệt. Điều này có thể cho thấy rằng mô hình DNN có thể không quá nhạy cảm với sự thay đổi này.
- **LSTM (Long Short-Term Memory)**: Tỉ lệ phát hiện rất thấp (dưới 1%) và không thay đổi đáng kể khi thay đổi max_iter, tức là mẫu đối kháng có khả năng trốn tránh tốt trên mô hình này.
- **Ensemble ML và Ensemble DL**: Cả hai đều có tỉ lệ phát hiện cao hơn so với các mô hình đơn lẻ nhưng cũng có sự dao động khi thay đổi max_iter. Việc tối ưu hóa lâu hơn có thể cải thiện hoặc làm giảm hiệu quả của các ensemble.
- **Thời gian** thì nó tăng tuyến tính theo tăng của vòng lặp tối ưu hóa.

Nhận xét: Từ biểu đồ số liệu 4.7, ta thấy được khi điều chỉnh số lần lặp trong thuật toán zoo để tạo ra các mẫu đối kháng, thì tùy vào lúc thuật toán ước lượng và tạo ra được mẫu đối kháng có thể tác động mạnh mẽ nhất lên mô

hình, không phụ thuộc quá nhiều vào giá trị vòng lặp tối ưu hóa để tìm ra mẫu đối kháng mạnh mẽ nhất.

> **[RQ4]Take-away:** Tùy vào thuật toán tấn công và các giá trị phụ thuộc, thì mẫu đối kháng được tạo ra có tác động khác nhau lên các bộ phân loại khác nhau (phân loại sai hoặc không có tác động).

4.4.3. Attack Defense dùng Adversarial Training

Ở phần này, chúng tôi đánh giá mức độ phòng thủ của các mô hình sau khi được huấn luyện đối kháng và kiểm tra tính Robustness của mô hình trước các cuộc tấn công chưa biết trước. Chúng tôi vẫn chọn mô hình DNN để tạo ra mẫu đối kháng để Defence như ở phần đầu để tiến hành nghiệm thu kết quả.

Bảng 4.22: *Kết quả Detection Rate của các Detector sau khi sử dữ liệu đối kháng được sinh ra từ* ***mô hình đề xuất GAN và Multimodal Learning*** *để huấn luyện lại các Detector. (%)*

Detector	**Detection Rate**
GNB	100
DT	99.79
LR	100
DNN	100
LSTM	100
Ensemble ML	100
Ensemble DL	100

Kết quả trong bảng 4.22 cho thấy tỉ lệ phát hiện tấn công của các Detector được huấn luyện đối kháng sử dụng các mẫu đối kháng được tạo bởi GAN và Multimodal Learning để làm dữ liệu huấn luyện ,trước mẫu đối kháng được tạo mới. Kết quả cho thấy các mô hình GNB, LR, DNN, LSTM, Ensemble ML và Ensemble DL đều đạt Detection Rate 100%, ngoại trừ mô hình DT là 99.79%. Từ kết quả này, nhóm chúng tôi nhận định rằng, phương pháp huấn luyện đối kháng sử dụng dữ liệu đối kháng để học có hiệu quả trước các dữ liệu tấn công đối kháng, được tạo ra từ mô hình GAN và Multimodal Learning của nhóm.

Bảng 4.23: *Kết quả* ***Detection Rate*** *của các Detector sau khi được huấn luyện lại trong đó có lẫn các mẫu đối kháng bởi các mẫu được tạo ra bởi các thuật toán .*

	After retraining			
	FGSM	**Deepfool**	**C&W**	**Zoo**
GNB	99.17	99.18	97.38	99.84
DT	43.14	99.52	43.34	99.99
LR	98.90	99.99	98.24	99.33
DNN	99.90	98.53	99.82	99.90
LSTM	**99.98**	**100.00**	**99.95**	**100.00**
Ensemble ML	99.75	99.90	98.58	99.33
Ensemble DL	99.93	99.26	99.88	99.95

Kết quả trong Bảng 4.23 thể hiện tỷ lệ phát hiện (Detection Rate) của các bộ phát hiện (Detector) sau khi được huấn luyện lại bằng cách sử dụng các mẫu đối kháng được tạo ra bởi các thuật toán khác nhau (FGSM, Deepfool, C&W, Zoo). Sau đó đánh giá mô hình bằng những mẫu đối kháng mới được tạo ra bởi thuật toán tạo mẫu đối kháng khi huấn luyện Defense. Cụ thể:

- GNB: Có tỷ lệ phát hiện cao trên tất cả các mẫu đối kháng, đặc biệt với thuật toán Zoo đạt tỷ lệ phát hiện gần như hoàn hảo (99.84%).
- DT: Có hiệu suất kém với FGSM và C&W (khoảng 43%) nhưng lại có tỷ lệ phát hiện rất cao với Deepfool và Zoo (99.52% và 99.99%).
- LR: Có tỷ lệ phát hiện rất cao với hầu hết các mẫu đối kháng, đặc biệt với Deepfool đạt 99.99%.
- DNN: Hiệu suất tốt trên tất cả các thuật toán, với tỷ lệ phát hiện dao động từ 98.53% đến 99.90%.
- LSTM: Có tỷ lệ phát hiện hoàn hảo hoặc gần hoàn hảo trên tất cả các mẫu

đối kháng (99.95% đến 100%).

- Ensemble ML: Hiệu suất cao trên tất cả các mẫu đối kháng, với tỷ lệ phát hiện từ 98.58% đến 99.90%.
- Ensemble DL: Hiệu suất rất cao và ổn định, với tỷ lệ phát hiện dao động từ 99.26% đến 99.95%.

Lí giải cho việc DT không hiệu quả trên việc phòng thủ trước các cuộc tấn công đối kháng bởi FGSM và C&W là:

- Bản chất của Decision Tree khi bị quá khớp (Overfitting): Decision Tree có xu hướng quá khớp với dữ liệu huấn luyện, đặc biệt khi cây có độ sâu lớn. Điều này làm cho DT dễ bị lừa bởi các mẫu đối kháng được tạo ra bằng cách thêm nhiễu nhỏ mà vẫn duy trì bản chất của mẫu gốc
- Phân vùng không gian đặc trưng: Decision Tree chia không gian đặc trưng thành các vùng nhỏ dựa trên các ngưỡng quyết định. Các mẫu đối kháng như FGSM và C&W tạo ra nhiễu để đẩy mẫu vào các vùng sai lệch này, làm giảm khả năng phân loại chính xác của DT.
- Đặc điểm của các thuật toán tạo mẫu đối kháng: Mẫu đối kháng rơi vào các vùng của không gian đặc trưng mà DT không thể phân biệt được rõ ràng từ đó đưa ra kết quả phân loại sai.

Kết luận rằng: Các mô hình học sâu (DNN, LSTM, Ensemble DL) có hiệu suất phát hiện cao hơn so với các mô hình học máy truyền thống (GNB, DT, LR). LSTM có tỷ lệ phát hiện hoàn hảo với tất cả các mẫu đối kháng, chứng tỏ khả năng mạnh mẽ trong việc phát hiện các mẫu đối kháng. Ensemble DL và Ensemble ML cũng cho thấy hiệu suất vượt trội, nhờ vào việc kết hợp các mô hình khác nhau để tận dụng ưu điểm của từng mô hình. Kết quả này chứng minh rằng việc huấn luyện có thêm các mẫu đối kháng làm cho các bộ phân loại có thể cải thiện đáng kể hiệu suất của chúng trong việc phát hiện các mẫu đối kháng.

[**RQ5**] **Take-away:** Phương pháp Adversarial Training hiệu quả tốt nhất trên bộ phân loại **LSTM** với hiệu suất lần lượt là 99.98, 100.00, 99.95 và 100.00 trước các thuật toán tạo mẫu đối kháng lần lượt là FGSM, Deepfool, C&W và Zoo. Đồng thời đạt hiệu quả tốt trên các bộ phân loại **GNB, LR, DNN, LSTM, Ensemble ML và Ensemble DL** với Detection Rate là 100% trước các mẫu đối kháng được tạo từ mô hình GAN và Multimodal Learning.

Tiếp theo, chúng tôi tiến hành đánh giá tính phòng thủ của các bộ phân loại đã được huấn luyện bởi các mẫu đối kháng bởi một thuật toán độc lập và kiểm tra tính phòng thủ của bộ phân loại trên định nghĩa Robustness là mô hình có khả năng chống lại các tấn công đối kháng khác hay không?

Bảng 4.24: *Kết quả* ***Detection Rate*** *của các Detector sau khi được huấn luyện lại trong đó có lẫn các mẫu đối kháng được tạo ra thuật toán* ***FGSM*** *và đánh giá Robustness của hệ thống.*

	After retraining	**Robustness**		
	FGSM	**Deepfool**	**C&W**	**ZOO**
GNB	99.17	99.99	46.52	99.14
DT	99.99	99.52	43.35	43.15
LR	98.90	**100.00**	96.77	98.99
DNN	99.90	99.99	99.90	99.90
LSTM	**99.98**	84.55	**99.99**	**99.94**
Ensemble ML	99.75	**100.00**	96.89	99.88
Ensemble DL	99.93	92.27	99.94	99.92

Bảng 4.24 thể hiện kết quả của các mô hình phát hiện (Detector) sau khi được huấn luyện lại với việc bao gồm các mẫu đối kháng được tạo ra bằng thuật toán FGSM (Fast Gradient Sign Method), đồng thời đánh giá tính chống chịu (Robustness) của hệ thống đối với các phương pháp tấn công khác nhau như Deepfool, C&W (Carlini & Wagner), và ZOO.

Hiệu quả của FGSM:

Các mô hình phức tạp như DNN, LSTM và các mô hình Ensemble cho thấy tỉ lệ phát hiện rất cao sau khi được huấn luyện lại với các mẫu đối kháng FGSM. Cụ thể, DNN đạt 99.90%, LSTM đạt 99.98%, Ensemble ML đạt 99.75%, và

Ensemble DL đạt 99.93%.

Robustness cao:

Các mô hình như LR, DNN và các mô hình Ensemble thể hiện tính chống chịu rất cao đối với các phương pháp tấn công khác. LR có Robustness đạt 100.00% với Deepfool, 96.77% với C&W, và 98.99% với ZOO. DNN có Robustness gần như tuyệt đối: 99.99% với Deepfool, 99.90% với C&W và ZOO. Ensemble ML và DL cũng có Robustness rất cao với các phương pháp tấn công khác nhau, ví dụ: Ensemble ML đạt 100.00% với Deepfool và 96.89% với C&W.

Khả năng cải thiện:

Đối với mô hình DT, mặc dù có khả năng chống chịu cao với Deepfool (99.52%), nhưng yếu với C&W (43.35%) và ZOO (43.15%). Việc sử dụng phương pháp Ensemble ML có thể giúp cải thiện độ chính xác và tính chống chịu của mô hình. Cụ thể, Ensemble ML đạt 99.75% trong Detection Rate và có Robustness cao với Deepfool (100.00%) và ZOO (99.88%).

Kết luận: Việc huấn luyện lại với các mẫu đối kháng FGSM đã cải thiện đáng kể tỉ lệ phát hiện và tính chống chịu của các mô hình phát hiện. Đặc biệt, các mô hình phức tạp và các phương pháp Ensemble cho thấy hiệu quả rõ rệt với tỉ lệ phát hiện và tính chống chịu cao. Tuy nhiên, cần cải thiện thêm đối với mô hình đơn giản như DT.

Bảng 4.25: *Kết quả* ***Detection Rate*** *của các Detector sau khi được huấn luyện lại trong đó có lẫn các mẫu đối kháng được tạo ra thuật toán* ***Deepfool*** *và đánh giá* ***Robustness*** *của hệ thống.*

	After retraining	**Robustness**		
	Deepfool	**FGSM**	**C&W**	**ZOO**
GNB	99.18	9.29	45.20	37.23
DT	99.52	42.08	43.35	42.74
LR	99.99	**70.73**	**73.85**	**62.62**
DNN	98.53	45.05	45.37	45.76
LSTM	**100.00**	34.37	36.16	33.65
Ensemble ML	99.90	38.29	45.27	56.93
Ensemble DL	99.26	39.70	40.76	39.70

Bảng 4.25 thể hiện kết quả của các mô hình phát hiện (Detector) sau khi được huấn luyện lại với việc bao gồm các mẫu đối kháng được tạo ra bằng thuật toán Deepfool, đồng thời đánh giá tính chống chịu (Robustness) của hệ thống đối với các phương pháp tấn công khác nhau như FGSM, C&W (Carlini & Wagner), và ZOO.

Kết luận: Ta thấy được sau khi mô hình được huấn luyện bởi dữ liệu đối kháng được tạo ra bởi Deepfool thì mô hình có tỉ lệ phát hiện (Detection rate) được những mẫu đối kháng được tạo ra bởi các thuật toán tấn công khác từ thấp đến khá (9.29% đến 73.85%). Vì vậy, khi mô hình được huấn luyện có một vài mẫu đối kháng được tạo ra bởi Deepfool thì các bộ phân loại có khả năng thấp để chống chịu lại các cuộc tấn công đối kháng.

Bảng 4.26: *Kết quả* ***Detection Rate*** *của các Detector sau khi được huấn luyện lại trong đó có lẫn các mẫu đối kháng được tạo ra thuật toán* ***ZOO*** *và đánh giá* ***Robustness*** *của hệ thống.*

	After retraining	**Robustness**		
	ZOO	**FGSM**	**Deepfool**	**C&W**
GNB	99.84	99.47	99.99	44.93
DT	99.99	99.00	99.99	99.99
LR	99.33	98.58	99.99	87.84
DNN	99.90	99.90	99.99	99.90
LSTM	**100.00**	**100.00**	**99.99**	**99.98**
Ensemble ML	99.33	98.58	99.99	87.84
Ensemble DL	99.95	99.95	99.99	99.94

Bảng 4.26 thể hiện kết quả của các mô hình phát hiện (Detector) sau khi được huấn luyện lại với việc bao gồm các mẫu đối kháng được tạo ra bằng thuật toán Zoo, đồng thời đánh giá tính chống chịu (Robustness) của hệ thống đối với các phương pháp tấn công khác nhau như FGSM, C&W (Carlini & Wagner), và Deepfool.

Sau khi các mô hình được huấn luyện lại với các mẫu đối kháng từ ZOO, tỷ lệ phát hiện đối với các mẫu đối kháng từ các phương pháp tấn công khác cũng tăng lên đáng kể. Điều này có thể được giải thích bởi bản chất của thuật toán

ZOO, là một phương pháp tấn công Black-Box (BB), nghĩa là nó ước lượng các giá trị cần thiết để tạo ra mẫu đối kháng mà không truy cập đầy đủ thông tin về mô hình. Khi các mô hình được huấn luyện với các mẫu đối kháng từ ZOO, chúng trở nên bền vững hơn đối với các cuộc tấn công đối kháng khác, dù chưa từng tiếp xúc với chúng trong quá trình huấn luyện.

Kết luận:Điều này cho thấy rằng việc bao gồm các mẫu đối kháng từ ZOO trong quá trình huấn luyện có thể cải thiện tính chống chịu của hệ thống đối với các phương pháp tấn công đối kháng khác, từ trung bình đến gần như tuyệt đối, tùy thuộc vào phương pháp tấn công cụ thể. Cụ thể:

- GNB: Tăng cường chống chịu từ 44.93% đến 99.99%.
- DT: Đạt gần như tuyệt đối, 99.99% cho tất cả các phương pháp.
- LR: Chống chịu cải thiện từ 87.84% đến 99.99%.
- DNN: Tăng lên gần như tuyệt đối, từ 99.90% đến 99.99%.
- LSTM: Đạt tuyệt đối hoặc gần như tuyệt đối, từ 99.99% đến 100%.
- Ensemble ML: Chống chịu cải thiện từ 87.84% đến 99.99%.
- Ensemble DL: Gần như tuyệt đối, từ 99.95% đến 99.99%.

Bảng 4.27: *Kết quả* ***Detection Rate*** *của các Detector sau khi được huấn luyện lại trong đó có lẫn các mẫu đối kháng được tạo ra thuật toán* ***C&W*** *và đánh giá* ***Robustness*** *của hệ thống.*

	After retraining	**Robustness**		
Detector	**C&W**	**FGSM**	**Deepfool**	**ZOO**
GNB	97.38	7.92	3.78	0.00
DT	99.42	42.07	99.20	42.69
LR	98.24	95.46	**99.99**	91.69
DNN	99.82	99.79	98.53	99.83
LSTM	**99.95**	**99.93**	48.63	**99.89**
Ensemble ML	98.58	45.46	62.09	46.57
Ensemble DL	99.88	99.86	73.58	99.86

Bảng 4.27 thể hiện kết quả của các mô hình phát hiện (Detector) sau khi được huấn luyện lại với việc bao gồm các mẫu đối kháng được tạo ra bằng thuật toán (Carlini & Wagner), đồng thời đánh giá tính chống chịu (Robustness) của hệ thống đối với các phương pháp tấn công khác nhau như FGSM, Deepfool , và ZOO.

Ta thấy được sau khi mô hình được huấn luyện bởi dữ liệu đối kháng được tạo ra bởi CW thì có nhưng mô hình không hiệu quả trong việc phát hiện như GNB(không phát hiện - ZOO 0% hoặc phát hiện rất kém 3.78% hoặc 7.92% với 2 thuật toán còn lại). Nó kéo theo mô hình Ensemble ML cũng không hiệu quả trong việc phát hiện tốt các mẫu đối kháng mà nó chưa từng biết tới. Các mô hình khác cũng có khả năng kháng đối kháng từ mức trung bình đến tốt.

Đối với các mô hình khác, như Logistic Regression (LR), Deep Neural Network (DNN), và Long Short-Term Memory (LSTM), khả năng chống chịu của chúng đối với các tấn công như Deepfool được cải thiện đáng kể. Đặc biệt, Logistic Regression (LR) và Deep Neural Network (DNN) thể hiện khả năng phát hiện gần như hoàn hảo với tỷ lệ phát hiện lần lượt là 99.99% và 98.53% khi đối phó với Deepfool. Điều này cho thấy rằng việc huấn luyện với các mẫu đối kháng C&W không chỉ giúp mô hình học được cách đối phó với các tấn công phức tạp mà còn tăng khả năng tổng quát hóa của chúng đối với các tấn công khác.

Ngoài ra, mô hình Ensemble DL (Deep Learning Ensemble) cũng cho thấy hiệu suất rất cao đối với các tấn công C&W, FGSM, và ZOO, nhưng có mức độ chống chịu trung bình đối với Deepfool (73.58%). Điều này chứng tỏ rằng việc kết hợp nhiều mô hình học sâu giúp tăng tính kiên cố của hệ thống, mặc dù vẫn có những hạn chế đối với các tấn công cụ thể như Deepfool.

Kết luận: Việc huấn luyện mô hình với các mẫu đối kháng từ thuật toán C&W có thể giúp cải thiện tính kiên cố của mô hình đối với các tấn công khác như Deepfool. Điều này chủ yếu là do mô hình học được cách đối phó với các thay đổi phức tạp trong không gian đầu vào, làm tăng khả năng tổng quát hóa và khả năng phát hiện các mẫu đối kháng khác. Tuy nhiên, mức độ cải thiện cụ

thể còn phụ thuộc vào loại mô hình và cách thức huấn luyện. Một số mô hình như GNB và Ensemble ML không hiệu quả trong việc phát hiện các tấn công mà chúng chưa từng gặp, trong khi các mô hình học sâu như DNN và LSTM, cũng như các mô hình kết hợp như Ensemble DL, lại thể hiện khả năng chống chịu tốt hơn.

[**RQ6**] **Take-away:** Các mô hình được phòng thủ bằng phương pháp Adversarial Training có khả năng Robustness. Nhận xét cụ thể như sau:

- Trước các tấn công đã biết: Adversarial Training hiệu quả trong việc tăng cường độ bền vững của mô hình đối với các tấn công đã biết, đặc biệt là các loại tấn công được sử dụng trong quá trình huấn luyện như FGSM.
- Chống lại tấn công tương tự: Mô hình cũng có thể bền vững hơn với các tấn công tương tự hoặc biến thể của tấn công đã biết, phụ thuộc vào cách các mẫu đối kháng được tạo ra và tính đa dạng của chúng trong quá trình huấn luyện.
- Phát hiện tấn công mới: Adversarial Training có thể phát hiện ra nhưng có thể không đủ để bảo vệ mô hình khỏi các tấn công mới và chưa được ghi nhận, do các kẻ tấn công có thể phát triển các phương pháp mới mà mô hình chưa từng được huấn luyện để tấn công đối kháng lên mô hình.

CHƯƠNG 5. KẾT LUẬN

Ở chương này, chúng tôi đưa ra những kết luận về nghiên cứu, những hạn chế, và đồng thời đưa ra hướng cải thiện và phát triển của đề tài khóa luận tốt nghiệp này

5.1. Kết luận

Trong nghiên cứu này, chúng tôi đã thành công trong việc xây dựng các bộ phân loại có hiệu suất phân loại cao. Đặc biệt, khi thực nghiệm, bộ phân loại LR đạt hiệu suất gần như tối đa - 100%.

Ngoài ra, chúng tôi đã triển khai thành công mô hình GAN kết hợp kỹ thuật Multimodal Learning để tạo ra mẫu đối kháng. Các mẫu đối kháng được tạo từ mô hình nhóm đề xuất có khả năng trốn tránh hiệu quả trước các trình phát hiện xâm nhập, Từ những kết quả ở trên, chúng tôi thấy rằng các Detector mà chúng tôi sử dụng trong khóa luận này, có khả năng kháng mẫu đối kháng kém trước các mẫu đối kháng được tạo ra từ mô hình mà nhóm chúng tôi đề xuất. Bên cạnh đó, chúng tôi đã triển khai thêm thành công các thuật toán tấn công đối kháng vào các bộ phân loại như FGSM, Deepfool, ZOO, C&W, tạo ra các mẫu đối kháng có khả năng tác động mạnh lên chúng, vượt mặt từ trung bình đến cao tuỳ thuộc vào bộ phân loại cụ thể và các mẫu đối kháng tạo ra đó có khả năng chuyển giao sang các trình phát hiện xâm nhập khác và có hiệu suất vượt mặt từ trung bình đến tốt (tỉ lệ phát hiện của các mẫu đối kháng của các bộ phân loại chỉ từ 0%-45% tùy vào thuật toán tấn công đang xét).

Chúng tôi tiếp tục áp dụng phương pháp huấn luyện chống lại các mẫu đối kháng (Adversarial Training) lên các bộ phân loại, giúp chúng nhận diện các mẫu đối kháng một cách hiệu quả. Kết quả cho thấy sau khi huấn luyện với các

mẫu đối kháng, bộ phân loại LSTM đạt tỉ lệ phân loại gần như 100% đối với các tấn công từ thuật toán đối kháng FGSM, Deepfool, ZOO, C&W, và các bộ phân loại GNB, LSTM, LR, DNN, Ensemble ML và Ensemble DL có tỉ lệ phát hiện 100% trước các mẫu đối kháng được tạo từ GAN và Multimodal Learning.

Cuối cùng, chúng tôi đã đánh giá khả năng Robustness của các bộ phân loại khi phải đối mặt với các cuộc tấn công đối kháng mà chúng chưa biết trước. Kết quả cho thấy mô hình vẫn duy trì được mức độ bền vững từ thấp đến khá.

Sau toàn bộ quá trình nghiên cứu và triển khai, chúng tôi nhận thấy một số ưu và nhược điểm như sau

Ưu điểm

- Giải quyết được bài toán đặt ra là nghiên cứu được khả năng kháng đối kháng của các mẫu đối kháng lên các trình xâm nhập (bộ phân loại).
- Đạt được mục tiêu là xây dựng được các bộ phân loại, triển khai được các thuật toán tấn công đối kháng đem lại hiệu quả tốt và triển khai được phương pháp phòng thủ và đánh giá tính Robustness của các mô hình

Nhược điểm

- Chưa triển khai được nhiều hơn các phương pháp phòng thủ khác ngoài Adversarial Training. Vì vậy chưa đánh giá được các phương pháp phòng thủ khác có ưu điểm hơn phương pháp phòng thủ mà chúng tôi sử dụng trong khóa luận lần này hay không.
- Với việc tấn công đối kháng, cần rất nhiều tài nguyên và thời gian để có thể tạo ra được mẫu đối kháng với mức độ tinh vi để có thể đánh lừa được các bộ phân loại.
- Phương pháp sử dụng nhiễu ngẫu nhiên để Generator sinh mẫu dữ liệu vẫn còn hạn chế.

5.2. Hướng phát triển

Dựa vào những kết quả nghiên cứu hiện tại, nhóm của chúng tôi nhận thấy tiềm năng lớn để phát triển đề tài này trong tương lai. Tuy nhiên, để đạt được kết quả tốt nhất, chúng tôi đang xem xét hai hướng đi chính.

Một là nghiên cứu và triển khai các phương pháp phòng thủ khác ngoài Adversarial Training, nhằm cải thiện khả năng phân loại nhận biết mẫu đối kháng. Điều này có thể bao gồm khảo sát các kỹ thuật mới như Robust Optimization và Adversarial Detection để tăng cường độ chính xác và độ tin cậy của hệ thống.

Hai là mở rộng phạm vi nghiên cứu để phát triển các mô hình phân loại mới, có khả năng phát hiện các mẫu đối kháng phức tạp hơn, đồng thời cải thiện hiệu suất và độ tin cậy. Công việc này đòi hỏi sự đầu tư về thời gian và tài nguyên lớn, nhưng sẽ mang lại giá trị lớn trong việc ứng phó với thách thức ngày càng tinh vi của các mẫu đối kháng được tạo ra khi được sự giúp đỡ của các hệ thống AI hiện đại.

TÀI LIỆU THAM KHẢO

Tiếng Anh:

[1] Nour Alhussien et al. (2024), "Constraining adversarial attacks on network intrusion detection systems: transferability and defense analysis", *IEEE Transactions on Network and Service Management.*

[2] N. Carlini and D. Wagner (2017), "Towards evaluating the robustness of neural networks".

[3] Adriel Cheng (2019), "PAC-GAN: Packet Generation of Network Traffic using Generative Adversarial Networks", *In 2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON).*

[4] Member IEEE Ying Zhong Wenqi Chen Jiahai Yang Senior Member IEEE Shuqiang Lu Xingang Shi Member IEEE Dongqi Han Zhiliang Wang and IEEE Xia Yin Senior Member (2021), "Evaluating and Improving Adversarial Robustness of Machine Learning-Based Network Intrusion Detectors", *IEEE Journal on Selected Areas in Communications, 39(8), 2632-2647.*

[5] HONGDOU HE GUYU ZHAO LIGANG HE HAITAO HE XIAOBING SUN and JIADONG REN (2019), "A Novel Multimodal-Sequential Approach Based on Multi-View Features for Network Intrusion Detection", *IEEE Access, 7, 183207-183221.*

[6] Ian Goodfellow Kurakin Alexey and Samy Bengior (2014), "Adversarial machine learning at scale".

[7] Y. Sharma J. Yi P.-Y. Chen H. Zhang and C.-J. Hsieh (2017), "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models", *10th ACM workshop on artificial intelligence and security.*

[8] Jiang Xie Hongbo Xu Zhenyu Cheng Rong Yang Peishuai Sun Shuhao Li (2023), "GPMT: Generating practical malicious traffic based on adversarial attacks with little prior knowledge", *Computers Security, 130, 103257.*

[9] A. Fawzi S.-M. Moosavi-Dezfooli and P. Frossard (2016), "Deepfool: a simple and accurate method to fool deep neural network", *in Proceedings of the IEEE conference on computer vision and pattern recognition.*

[10] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani (2018), "Toward generating a new intrusion detection dataset and intrusion traffic characterization", pp. 108–116.

[11] Trong-Nghia To (2023), "On the Effectiveness of Adversarial Samples against Ensemble Learning-based Windows PE Malware Detectors", *arXiv.*