ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

TIỂU LUẬN

Khai phá dữ liệu và học máy

Tấn công Elastic-Net vào Deep Neural Network
qua một số mẫu đối nghịch

Đại học Quốc gia Hà Nội
Đại học Khoa học Tự nhiên
Khoa Toán cơ tin

Giảng viên: PGS.TS. Trần Trọng Hiếu

Học viên: Nguyễn Mạnh Linh, Đào Thị Thu Hồng

Mục lục

Tóm tắt

Các nghiên cứu gần đấy đã chỉ ra tính dễ bị tổn thương của mạng neuron sâu (Deep Neural Networks - DNNs) bởi các mẫu đối nghịch (adversarial examples). Mẫu đối nghịch là trường hợp 1 ảnh được chỉnh sửa sao cho ảnh vẫn có thể hiện thị giác không khác gì nhiều so với ảnh gốc, nhưng nó lại đánh lừa được các mô hình phân lớp, kiến việc phân lớp bị sai. Các phương pháp hiện có để tạo ra mẫu đối nghịch thường dựa trên các biến dạng L_2 và L_∞. Biến dạng L_1 thể hiện tổng sai khác và làm tăng tính thưa của nhiễu, tuy nhiên thực tế nó vẫn ít được sử dụng để tạo ra các mẫu đối nghịch.

Trong bài này, tác giả thiết kế một quy trình tấn công DNNs bằng mẫu đối nghịch thông qua bài toán tối ưu hiệu chỉnh elastic-net. Tấn công DNNs bằng elastic-net (EAD) với tham số L_1 được thêm vào cùng với L_2. Kết quả thực nghiệm trên các tập dữ liệu MNIST, CIFAR10 và ImageNet chỉ ra rằng EAD có thể mang lại một tập mẫu đối nghịch với độ nhiễu L_1 nhỏ và đạt được hiệu suất tấn công tương đương với các phương pháp hiện đại nhất qua các kịch bản tấn công khác nhau. Quan trọng hơn, EAD hướng đến khả năng chuyển giao tấn công và huấn luyện đối nghịch bổ sung cho DNNs và gợi ý việc học các đối nghịch L_1 để cải thiện bảo mật trong các mô hình học máy.

Bài báo gốc: (Chen, Pin-Yu, et al. 2018)

1 Giới thiệu

Mạng nơ ron sâu (DNNs) đã đạt hiệu quả rất tốt với các bài toán trong học máy và trí tuệ nhân tạo như phân loại ảnh, nhận diện giọng nói, dịch máy và trò chơi. Mặc dù DNNs rất hiệu quả nhưng một số nghiên cứu gần đây đã chứng minh DNNs rất dễ "tổn thương" với các mẫu đối nghịch (Szegedy et al. 2013; Goodfellow, Shlens, and Szegedy 2015). Ví dụ, ảnh được gây nhiễu khéo có thể làm cho một DNN đã được huấn luyện phân loại sai. Hơn nữa, các mẫu đối nghịch được tạo ra hầu như không thể phân biệt được bằng mắt người.

Hình 1.1: Minh họa trực quan về mẫu đối nghịch được sinh bởi EAD. Hình gốc (đà điểu) được lấy từ tập ImageNet. Các mẫu đối nghịch bị phân loại sai với mô hình Inception-v3.

Ví dụ trực quan trên thể hiện 3 mẫu đối nghịch của một hình con đà điểu ("ostrich") được sinh ra bằng thuật toán nhóm tác giả đề xuất. Các mẫu này được mô hình Inception-v3 (Szegedy et al. 2016) phân loại thành "safe", "shoe shop" và "vacuum".

Sự thiếu mạnh mẽ của DNN trước các mẫu đối nghịch đã làm dấy lên những lo ngại về vấn đề bảo mật các ứng dụng, bao gồm nhận dạng tín hiệu giao thông và phát hiện phần mềm độc hại. Hơn nữa, vượt ra ngoài không gian số, các nhà nghiên cứu đã chỉ ra rằng những mẫu đối nghịch ảnh hưởng cả tới thế giới thực trong việc đánh lừa DNNs (Kurakin, Goodfellow, and Bengio 2016a; Evtimov et al. 2017). Do tính mạnh mẽ và ý nghĩa bảo mật, việc tạo ra các mẫu đối nghịch được gọi là tấn công (*attacks*) vào DNNs. Cụ thể, tấn công nhắm đích (*targeted attacks*) để tạo ra các mẫu đối nghịch được phân loại nhầm vào các lớp mục tiêu định sẵn; tấn công không nhắm đích (*untargeted attacks*) để tạo ra các mẫu đối nghịch không được phân loại vào lớp ban đầu. Chuyển giao tấn công (*transfer attacks*) nhằm mục đích tạo ra các mẫu đối nghịch có thể chuyển từ mô hình DNN này sang mô hình DNN khác. Ngoài việc đánh giá mức độ mạnh mẽ của DNNs, các mẫu đối nghịch còn có thể được sử dụng để huấn luyện mô hình sao cho nó có khả năng chống chịu với những xáo trộn của đối nghịch, gọi là huấn luyện đối nghịch (*adversarial training*) (Madry et al. 2017). Chúng cũng có thể được sử dụng để cải tiến DNNs (Koh và Liang 2017; Dong et al. 2017).

Trong bài báo này, tác giả sử dụng các mẫu đối nghịch để tấn công phân loại ảnh dựa trên mạng neuron tích chập. Mẫu đối nghịch được tạo ra để làm sai lệch kết quả dự đoán và phải đảm bảo hình mới tạo ra (gần) giống với hình gốc. Trong trường hợp này, sự giống nhau giữa mẫu đối nghịch và hình gốc được đo bằng các độ đo nhiễu (*distortion metrics*) khác nhau. Độ đo thường được sử dụng là chuẩn L_q với $\|x\|_q = (\sum_{i=1}^{p} |x_i|^q)^{\frac{1}{q}}$ kí hiệu chuẩn L_q của vector p chiều $x = [x_1, ..., x_p]$ với $q \geq 1$. Cụ thể, khi tạo ra các mẫu đối nghịch, chuẩn L_∞ được sử dụng để đánh giá sự thay đổi tối đa giá trị điểm ảnh (Goodfellow, Shlens, and Szegedy 2015), chuẩn L_2 được sử dụng để cải thiện chất lượng thị giác của ảnh (Carlini and Wagner 2017b). Mặc dù, chuẩn L_1 thực tế được sử dụng rộng rãi trong các bài toán chống nhiễu, khôi phục ảnh (Fu et al. 2006) cũng như phục hồi thưa (Candès and Wakin 2008), các mẫu đối nghịch dựa trên L_1 chưa được nghiên cứu nhiều. Trong bài toán mẫu đối nghịch, độ biến dạng L_1 đánh giá tổng các thay đổi trong nhiễu và đóng vai trò là một thành phần (hàm) lồi thay thế độ đo L_0, đo lường số lượng điểm ảnh bị nhiễu làm thay đổi (độ thưa). Để lấp đầy khoảng trống về mẫu đối nghịch L_1, nhóm tác giả đề xuất một thuật toán tấn công dựa trên hiệu chỉnh *elastic-net*, được gọi là tấn công elastic-net vào DNNs (elastic-net attacks to DNNs - EAD). Hiệu chỉnh elastic-net là tổ hợp tuyến tính của các hàm phạt L_1 và L_2 và nó cũng là công cụ chuẩn cho bài toán lựa chọn thuộc tính đặc trưng cho dữ liệu nhiều chiều (Zou and Hastie 2005). Trong bài toán tấn công DNNs, EAD mở ra hướng nghiên cứu mới từ cách tấn công hiện đại dựa trên L_2 (Carlini and Wagner 2017b), nó tạo ra các mẫu đối nghịch dựa trên L_1 vừa hiệu quả hơn vừa khác biệt cơ bản với các phương pháp tấn công hiện có.

Để khám phá hiệu quả của tấn công dựa trên L_1, nhóm tác giả tiến hành các thử nghiệm trên tập dữ liệu MNIST, CIFAR10, và ImageNet trong các tình huống tấn công khác nhau. So với các phương pháp tấn công hiện có dựa trên L_2 và L_∞ (Kurakin, Goodfellow, and Bengio 2016b; Carlini and Wagner 2017b), EAD có thể đạt được tỉ lệ tấn công thành công tương tự khi phá vỡ DNNs được phòng thủ hoặc không phòng thủ (Papernot et al. 2016b). Quan trọng hơn, tác giả chỉ ra rằng, tấn công L_1 đạt được hiệu suất vượt trội so với các tấn công L_2 và L_∞ trong chuyển giao tấn công và huấn luyện đối nghịch bổ sung. Với tập dữ liệu khó nhất (MNIST), EAD cải thiện khả năng chuyển giao tấn công từ DNN không phòng thủ vào DNN được phòng thủ chắt lọc (defensively distilled DNN) đạt tỉ lệ thành công gần 99%. Thêm vào đó việc huấn luyện kèm theo với mẫu đối nghịch dựa trên L_1 và L_2 có thể tăng cường khả năng chống chịu của DNNs đối với các nhiễu loạn. Những kết quả này cho thấy EAD mang lại tập mẫu đối nghịch khác biệt nhưng hiệu quả hơn. Ngoài ra, đánh giá các tấn công dựa trên độ nhiễu L_1 cung cấp thêm hiểu biết mới

về học máy đối kháng và sự bảo mật của DNNs, gợi ý rằng L_1 có thể bổ sung cho L_2 và L_∞ thúc đẩy các framework về học máy đối kháng hoàn thiện hơn.

2 Các nghiên cứu liên quan

Trong phần này, tác giả tổng hợp các nghiên cứu liên quan về tấn công và phòng thủ DNNs.

2.1 Tấn công vào DNNs

FGM và I-FGM: Kí hiệu $\mathbf{x_0}$ và $\mathbf{x}$ lần lượt là mẫu gốc và mẫu đối nghịch, t là lớp mục tiêu cần tấn công. Các phương pháp gradient nhanh (*fast gradient methods - FGM*) sử dụng gradient ∇J của hàm mất mát J tại $\mathbf{x_0}$ để tạo ra các mẫu đối nghịch (Goodfellow, Shlens, and Szegedy 2015). Với tấn công L_∞, $\mathbf{x}$ được tính bằng công thức:

$$\mathbf{x} = \mathbf{x_0} - \epsilon \times \text{sign}(\nabla J(\mathbf{x_0}, t)) \tag{2.1}$$

Trong đó ϵ là độ biến dạng L_∞ giữa $\mathbf{x}$ và $\mathbf{x_0}$ và $\text{sign}(\nabla J)$ trả về dấu của gradient. Với tấn công L_1 và L_2, $\mathbf{x}$ được tính bằng công thức:

$$\mathbf{x} = \mathbf{x_0} - \epsilon \frac{\nabla J(\mathbf{x_0}, t)}{\|\nabla J(\mathbf{x_0}, t)\|_q} \tag{2.2}$$

với $q = 1, 2$ và ϵ là độ nhiễu tương ứng. Các phương pháp lặp gradient nhanh (*Iterative fast gradient methods (I-FGM)*) được trình bày trong (Kurakin, Goodfellow, and Bengio 2016b) là phương pháp lặp sửa dụng FGM với một độ nhiễu mịn hơn. Tấn công không nhắm đích sử dụng FGM và I-FGM cũng được triển khai theo cách tương tự.

Tấn công C&W: Thay vì sử dụng hàm mất mát trên tập huấn luyện Carlini và Wagner đã thiết kế một hiệu chỉnh L_2 trong hàm mất mát dựa trên lớp logit trong DNNs để sinh ra các mẫu đối nghịch (Carlini and Wagner 2017b). Công thức này hóa ra là một trường hợp riêng của thuật toán EAD (sẽ được trình bày trong phần sau). Tấn công C&W được coi là một trong những tấn công mạnh nhất đối với DNNs vì nó có thể phá vỡ cả những DNN không phòng thủ và DNN phòng thủ chất lọc, nó cũng có khả năng chuyển giao tấn công khá tốt.

DeepFool: là thuật toán tấn công L_2 không nhắm đích (Moosavi-Dezfooli, Fawzi, and Frossard 2016) dựa trên lý thuyết về phép chiếu tới siêu phẳng phân tách gần nhất trong bài toán phân lớp. Nó cũng được sử dụng để tạo ra một nhiễu loạn phổ quát nhằm đánh lừa các DNN vốn đã được huấn luyện trên các ảnh gốc (Moosavi-Dezfooli et al. 2016).

2.2 Phòng thủ trong DNNs

Chắt lọc phòng thủ (Defensive distillation): Chắt lọc phòng thủ (Papernot et al 2016b) chống lại nhiễu loạn đối nghịch bằng cách sử dụng kỹ thuật chắt lọc trong (Hinton, Vinyals, and Dean 2015) để huấn luyện lại cùng một mạng với xác suất lớp được dự đoán bởi mạng ban đầu. Phương pháp này đưa ra tham số nhiệt độ T trong lớp softmax để tăng cường sức mạnh của mạng trước những nhiễu loạn đối nghịch.

Huấn luyện đối nghịch (Adversarial training): có thể được triển khai theo 1 số cách khác nhau. Hướng tiếp cận chuẩn là thêm vào tập dữ liệu huấn luyện ban đầu các mẫu đối nghịch đã được sửa đúng nhãn, sau đó huấn luyện lại mạng. (Zheng et al. 2016; Madry et al. 2017; Tram‘er et al. 2017; Zantedeschi, Nicolae, and Rawat 2017) đề xuất điều chỉnh training loss hoặc kiến trúc mạng để tăng cường sức mạnh của DNNs trước mẫu đối nghịch.

Các phương pháp phát hiện: Sử dụng các test thống kê để phân biệt các mẫu đối nghịch với mẫu gốc (Feinman et al. 2017; Grosse et al. 2017; Lu, Issaranon, and Forsyth 2017; Xu, Evans, and Qi 2017). Tuy nhiên, 10 phương pháp phát hiện khác nhau đã không thể phát hiện được tấn công C&W (Carlini and Wagner 2017a).

3 EAD: Tấn công Elastic-Net vào mạng DNNs

3.1 Sơ bộ về hiệu chỉnh Elastic-Net

Hiệu chỉnh elastic-net là kỹ thuật được sử dụng rộng rãi trong việc giải quyết các bài toán lựa chọn thuộc tính nhiều chiều (Zou and Hastie 2005). Nó là hiệu chỉnh sử dụng tổ hợp tuyến tính hàm phạt (*penalty*) L_1 và L_2. Nhìn chung, hiệu chỉnh elastic-net được sử dụng trong bài toán cực tiểu hóa sau đây:

$$\text{minimize}_{\mathbf{z}\in\mathcal{Z}}\ f(\mathbf{z}) + \lambda_1\|\mathbf{z}\|_1 + \lambda_2\|\mathbf{z}\|_2^2 \tag{3.1}$$

Trong đó $\mathbf{z}$ là vector của p biến tối ưu, $\mathcal{Z}$ là tập nghiệm chấp nhận được, $f(\mathbf{z})$ là hàm mất mát, $\|\mathbf{z}\|_q$ là chuẩn q của $\mathbf{z}$ và $\lambda_1, \lambda_2 \geq 0$ tương ứng là các tham số hiệu chỉnh L_1 và L_2. Biểu thức $\lambda_1\|\mathbf{z}\|_1 + \lambda_2\|\mathbf{z}\|_2^2$ được gọi là hiệu chỉnh elactic-net của $\mathbf{z}$. Với bài toán hồi quy chuẩn, hàm mất mát $f(\mathbf{z})$ là hàm sai số bình phương trung bình (*mean squared error - MSE*), vector $\mathbf{z}$ biểu diễn trọng số của các thuộc tính và $\mathcal{Z} = \mathbb{R}^p$. Hiệu chỉnh elastic-net trong (3.1) chính là công thức Lasso khi $\lambda_2 = 0$ và trở thành công thức hồi quy Ridge khi $\lambda_1 = 0$. (Zou and Hastie 2005) đã chỉ ra rằng hiệu chỉnh elastic-net có thể chọn ra 1 nhóm các thuộc tính tương quan mạnh, khắc phục nhược điểm của việc chọn lựa thuộc tính nhiều chiều khi sử dụng 1 mình hồi quy Lasso hoặc hồi quy Ridge.

3.2 Xây dựng thuật toán EAD

Từ ý tưởng của tấn công C&W (Carlini and Wagner 2017b), ta xem xét cùng hàm mất mát f để sinh ra các mẫu đối nghịch. Cụ thể, cho trước 1 ảnh $\mathbf{x_0}$ với nhãn đúng của nó là t_0, gọi $\mathbf{x}$ là mẫu đối nghịch của $\mathbf{x_0}$ với lớp đích nhắm đến là $t \neq t_0$. Hàm mất mát $f(\mathbf{x})$ cho tấn công nhắm đích là:

$$f(\mathbf{x}, t) = \max\left(\max_{j\neq t}[\mathbf{Logit}(\mathbf{x})]_j - [\mathbf{Logit}(\mathbf{x})]_t, -\kappa\right) \tag{3.2}$$

Trong đó $\mathbf{Logit}(\mathbf{x}) = [[\mathbf{Logit}(\mathbf{x})]_1, ..., [\mathbf{Logit}(\mathbf{x})]_K] \in \mathbb{R}^K$ là lớp logit (lớp trước softmax) biểu diễn cho $\mathbf{x}$ trong mạng DNN, K là số lượng lớp cần phân loại, $\kappa > 0$ là tham số độ tin cậy, nó đảm bảo một khoảng cách cố định giữa $\max_{j\neq t}[\mathbf{Logit}(\mathbf{x})]_j$ và $[\mathbf{Logit}(\mathbf{x})]_t$.

Cần lưu ý rằng, thành phần $[\mathbf{Logit}(x)]_t$ là xác suất dự đoán x có nhãn t theo luật phân loại của hàm softmax:

$$\text{Prob}(\text{Label}(\mathbf{x}) = t) = \frac{\exp([\mathbf{Logit}(\mathbf{x})]_t)}{\sum_{j=1}^{K}\exp([\mathbf{Logit}(\mathbf{x})]_j)} \tag{3.3}$$

Do đó, hàm mất mát trong (3.2) có mục đích là cho ra nhãn t là lớp có xác suất cao nhất của $\mathbf{x}$ và tham số κ đảm bảo sự phân biệt giữa lớp t và lớp dự đoán gần nhất khác với t. Với tấn công không nhắm đích, hàm mất mát trong 3.2 trở thành:

$$f(\mathbf{x}) = \max\left([\mathbf{Logit}(\mathbf{x})]_{t_0} - \max_{j \neq t}[\mathbf{Logit}(\mathbf{x})]_j, -\kappa\right) \tag{3.4}$$

Trong bài viết này, tác giả tập trung vào tấn công nhắm đích vì nó khó hơn so với tấn công không nhắm đích. Thuật toán EAD có thể được áp dụng trực tiếp vào tấn công không nhắm đích bằng cách thay $f(\mathbf{x}, t)$ trong (3.2) thành $f(\mathbf{x})$ trong (3.4).

Ngoài ra, để thao túng kết quả dự đoán thông qua hàm mất mát trong (3.2), hiệu chỉnh elastic-net còn tạo ra mẫu đối nghịch tương tự với ảnh gốc. Công thức tấn công elastic-net vào mạng DNNs (EAD) để tạo ra mẫu đối nghịch $(\mathbf{x}, t)$ cho ảnh gốc $(\mathbf{x_0}, t_0)$ như sau:

$$\begin{aligned} &\text{minimize}_{\mathbf{x}}\ c \times f(\mathbf{x}, t) + \beta\|\mathbf{x} - \mathbf{x_0}\|_1 + \|\mathbf{x} - \mathbf{x_0}\|_2^2 \\ &\text{st } \mathbf{x} \in [0,1]^p \end{aligned} \tag{3.5}$$

Với $f(\mathbf{x}, t)$ được xác định trong (3.2), $c, \beta \geq 0$ lần lượt là các tham số hiệu chỉnh của hàm mất mát f và hàm phạt L_1. Miền ràng buộc $\mathbf{x} \in [0,1]^p$ giữ cho $\mathbf{x}$ trong một không gian ảnh mà có thể dễ dàng thỏa mãn điều kiện bằng cách chia giá trị của mỗi điểm ảnh cho giá trị lớn nhất (ví dụ 255). Để xác định khoảng cách (nhiễu) giữa $\mathbf{x}$ và $\mathbf{x_0}$ bằng $\delta = \mathbf{x} - \mathbf{x_0}$, công thức EAD trong (3.5) được tạo ra để tìm một mẫu đối nghịch $\mathbf{x}$ mà sẽ được phân loại vào lớp mục tiêu t, đồng thời cực tiểu hóa δ trong hàm mất mát elastic-net $\beta\|\delta\|_1 + \|\delta\|_2^2$. Đáng chú ý, công thức tấn công C&W (Carlini and Wagner 2017b) trở thành trường hợp đặc biệt của EAD trong công thức (3.5) khi $\beta = 0$, không phụ thuộc vào L_1 của δ. Tuy nhiên thành phần phạt L_1 là một hiệu chỉnh trực quan để tạo ra mẫu đối nghịch do $\|\delta\|_1 = \sum_{i=1}^{p} |\delta_\mathbf{i}|$ biểu diễn tổng sai khác của nhiễu và được sử dụng rộng rãi như là hàm thay thế để phát triển các nhiễu thưa. Phần đánh giá hiệu năng sẽ chứng minh rằng khi đưa thành phần L_1 vào nhiễu sẽ sinh ra tập phân biệt các mẫu đối nghịch, và nó cải thiện khả năng chuyển giao tấn công và triển khai huấn luyện đối nghịch.

3.3 Thuật toán EAD

Khi giải bài toán EAD trong (3.5) mà không có thành phần L_1, Carlini and Wagner sử dụng kĩ thuật COV (change-of-variable, kỹ thuật đổi biến) qua hàm *tanh* trên $\mathbf{x}$ nhằm mục đích loại bỏ ràng buộc $\mathbf{x} \in [0,1]^p$ (Carlini and Wagner 2017b). Khi $\beta > 0$, ta thấy COV sẽ không hiệu quả để giải bài toán (3.5) vì mẫu đối nghịch tương ứng sẽ không nhạy cảm với thay đổi trên β (xem phần đánh giá hiệu năng để biết chi tiết). Vì L_1 không khả vi trên toàn không gian, COV không giải được bài

toán 3.5 do không hiệu quả khi sử dụng phương pháp dưới đạo hàm (*subgradient*) trong việc giải bài toán tối ưu (Duchi and Singer 2009).

Để giải bài toán EAD trong (3.5) nhằm sinh ra các mẫu đối nghịch, tác giả đề xuất sử dụng thuật toán ISTA (*iterative shrinkage-thresholding algorithm – thuật toán lặp với ngưỡng biến đổi*) (Beck and Teboulle 2009). ISTA là một thuật toán tối ưu bậc nhất phổ biến, trong đó thêm một bước điều chỉnh ngưỡng trong mỗi bước lặp. Cụ thể, đặt $g(\mathbf{x}) = c \times f(\mathbf{x}) + \|\mathbf{x} - \mathbf{x_0}\|_2^2$ và đặt $\nabla g(\mathbf{x})$ là gradient của $g(\mathbf{x})$ được tính bởi mạng DNN. Tại bước lặp thứ $k+1$, mẫu đối nghịch $\mathbf{x}^{(k+1)}$ của $\mathbf{x_0}$ được tính như sau:

$$\mathbf{x}^{(k+1)} = S_\beta(\mathbf{x}^{(k)} - \alpha_k \nabla g(\mathbf{x}^{(k)})) \tag{3.6}$$

Trong đó, α_k là độ dài bước tại bước lặp thứ $k+1$, $S_\beta : \mathbb{R}^p \to \mathbb{R}^p$ là hàm của phép chiếu biến đổi ngưỡng trên từng phần tử, được xác định bởi:

$$[S_\beta(\mathbf{z})]_i = \begin{cases} \min\{\mathbf{z}_i - \beta, 1\} & \text{nếu } \mathbf{z}_i - \mathbf{x_0}_i > \beta; \\ \mathbf{x_0}_i & \text{nếu } |\mathbf{z}_i - \mathbf{x_0}_i| \leq \beta; \\ \max\{\mathbf{z}_i + \beta, 0\} & \text{nếu } \mathbf{z}_i - \mathbf{x_0}_i < -\beta \end{cases} \tag{3.7}$$

Với $i \in \{1, ..., p\}$. Nếu $|\mathbf{z}_i - \mathbf{x_0}_i| > \beta$, thành phần $\mathbf{z}_i$ được co lại với hệ số β và chiếu thành phần kết quả lên miền ràng buộc chấp nhận được thuộc đoạn $[0, 1]$. Mặt khác, nếu $|\mathbf{z}_i - \mathbf{x_0}_i| \leq \beta$, nó đặt ngưỡng $\mathbf{z}_i$ bằng cách thiết lập $[S_\beta(\mathbf{z})]_i = \mathbf{x_0}_i$. Chứng minh thuật toán tối ưu sử dụng (3.6) để giải bài toán EAD trong (3.5) được trình bày trong tài liệu [1]. Chú ý rằng, vì $g(\mathbf{x})$ là hàm mục tiêu của tấn công C&W (Carlini and Wagner 2017b), thuật toán ISTA trong (3.6) có thể coi như 1 phiên bản mạnh hơn của phương pháp C&W trong đó biến đổi giá trị điểm ảnh của mẫu đối nghịch nếu nó sai khác nhiều hơn β so với điểm ảnh gốc, hoặc giữ nguyên giá trị điểm ảnh gốc nếu sự sai khác nhỏ hơn β.

Thuật toán 1 Tấn công Elastic-net vào DNNs (EAD)

Input: Ảnh gốc và nhãn của nó ($\mathbf{x}_0$, t_0), lớp mục tiêu t, tham số chuyển giao κ, tham số hiệu chỉnh β, độ dài bước α_k, số bước lặp I
Output: mẫu đối nghịch $\mathbf{x}$
Khởi tạo: $\mathbf{x}^{(0)} = \mathbf{y}^{(0)} = \mathbf{x}_0$
for $k = 0$ to $I - 1$ do
  $\mathbf{x}^{(k+1)} = S_\beta(\mathbf{y}^{(k)} - \alpha_k \nabla g(\mathbf{y}^{(k)}))$
  $\mathbf{y}^{(k+1)} = \mathbf{x}^{(k+1)} + \frac{k}{k+3}(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)})$
end for
Luật quyết định: tìm $\mathbf{x}$ từ tập các mẫu thành công trong $\{\mathbf{x}^k\}_{k=1}^I$ (luật EN, luật L_1).

Thuật toán EAD để tạo mẫu đối nghịch được tổng kết với mã giả trên. Để tính toán hiệu quả, ta triển khai FISTA (fast ISTA) cho EAD, nó sẽ cho tốc độ hội tụ tối ưu

[1] https://arxiv.org/abs/1709.04114

với phương pháp tối ưu hóa bậc 1 (Beck and Teboulle 2009). Vector $\mathbf{y}^{(k)}$ trong thuật toán 1 kết hợp với momen trong $\mathbf{x}^{(k)}$ để tăng tốc hội tụ. Trong thực nghiệm, tác giả khởi tạo learning rate $\alpha_0 = 0.01$ với tham số squared root decay k. Trong phép lặp EAD, bước lặp $\mathbf{x}^{(k)}$ được coi là mẫu đối nghịch thành công của $\mathbf{x_0}$ nếu mô hình dự báo thành lớp đích t. Mẫu đối nghịch $\mathbf{x}$ cuối cùng được lựa chọn từ tất cả các mẫu đối nghịch thành công. Trong bài viết này, chúng ta xem xét 2 luật để chọn $\mathbf{x}$ từ tập các mẫu đối nghịch thành công: elastic-net tối thiểu (EN) và nhiễu L_1 tại $\mathbf{x_0}$. Sự ảnh hưởng của β, κ và luật quyết định lên EAD sẽ được nghiên cứu trong phần tiếp theo.

4 Đánh giá hiệu năng

Trong phần này, chúng ta so sánh EAD với các thuật toán tấn công tiên tiến hiện nay trên 3 tập dữ liệu ảnh: MNIST, CIFAIR10 và ImageNet. Tác giả bài viết muốn cho thấy:

- EAD có thể thu được hiệu năng tấn công tương tự tấn công C&W đối với mạng DNN phòng thủ và mạng DNN không phòng thủ do C&W là trường hợp đặc biệt của EAD khi $\beta = 0$.
- So sánh với các phương pháp tấn công hiện tại dựa trên L_1 như FGM, I-FGM, các mẫu đối nghịch thu được từ EAD có nhiễu L_1 nhỏ hơn đáng kể và tỷ lệ tấn công thành công cao hơn.
- Các mẫu đối nghịch tạo bởi EAD có thể hỗ trợ chuyển giao tấn công và triển khai huấn luyện đối nghịch.

4.1 Các phương pháp đối sánh

Bài viết này so sánh EAD với các phương pháp tấn công nhắm đích hiệu quả nhất có sử dụng các loại hiệu chỉnh khác nhau:

- C&W: tấn công nhắm đích dùng hiệu chỉnh L_2 được đề xuất bởi Carlini and Wagner (Carlini and Wagner 2017b). Đây là trường hợp đặc biệt của EAD khi $\beta = 0$.
- FGM (fast gradient method): được đề xuất bởi (Goodfellow, Shlens, and Szegedy 2015). FGM sử dụng nhiều loại hiệu chỉnh với các phiên bản FGM-L_1, FGM-L_2, FGM-L_∞.
- I-FGM: (FGM lặp): được đề xuất bởi (Kurakin, Goodfellow, and Bengio 2016b). I-FGM sử dụng nhiều loại hiệu chỉnh I-FGM-L_1, I-FGM-L_2, I-FGM-L_∞.

4.2 Thiết kế thực nghiệm

Thực nghiệm sử dụng framework của Carlini và Wagner [2]. Với cả tấn công EAD và C&W, tác giả đều sử dụng các tham số cấu hình mặc định, trong đó triển khai 9 bước tìm kiếm nhị phân trên tham số hiệu chỉnh c (bắt đầu từ 0.001) và chạy vòng lặp cho mỗi bước với learning rate khởi tạo. Để tìm các mẫu đối nghịch thành công,

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

tác giả sử dụng thuật toán tối ưu tham chiếu (ADAM) cho C&W, và sử dụng thuật toán chiếu FISTA (thuật toán 1) với square-root decaying learning rate cho EAD. Tương tự tấn công C&W, mẫu đối nghịch cuối cùng tìm được từ EAD được chọn là mẫu nhiễu ít nhất giữa các mẫu đối nghịch thành công. Độ nhạy cảm của L1 (tham số β) và tác động của luật chọn trong EAD sẽ được bàn đến ở phần sau. Tác giả đặt tham số chuyển giao tấn công $\kappa = 0$ cho cả EAD và C&W.

Tác giả triển khai FGM và I-FGM bằng gói thư viện CleverHans [3]. Tham số nhiễu tốt nhất ϵ được xác định bằng phương pháp tìm kiếm lưới fine-grained trên mỗi ảnh, ϵ nhỏ nhất trong lưới đạt được tấn công thành công sẽ được ghi lại. Với I-FGM, tác giả thực hiện 10 vòng lặp FGM (10 là giá trị mặc định) với ϵ-ball clipping. Tham số nhiễu ϵ' trong mỗi vòng lặp FGM được đặt là $\epsilon/10$, đã được chứng minh hiệu quả trong (Tramèr et al. 2017). Dải giá trị của lưới và độ phân giải của 2 phương pháp trên được đề cập cụ thể trong tài liệu bổ sung 1.

Việc phân loại ảnh cho tập MNIST và CIFAIR10 được huấn luyện trên mô hình DNN bởi Carlini and Wagner. Để phân loại cho tập ImageNet, tác giả dùng mô hình Inception-v3 (Szegedy et al. 2016). Đối với tập MNIST và CIFAIR10, tác giả lấy ngẫu nhiên trong tập test 1000 ảnh đã được phân loại đúng để tấn công làm phân loại sai. Với tập ImageNet, tác giả chọn ngẫu nhiên 100 ảnh đã được phân loại đúng và 9 lớp sai để tấn công. Tất cả thực nghiệm được triển khai trên thiết bị phần cứng Intel E5-2690 v3 CPU, 40 GB RAM, single NVIDIA K80 GPU. Code thực nghiệm EAD có thể được download từ github [4]

4.3 Các độ đo đánh giá

Theo cách đánh giá trong (Carlini and Wagner 2017b), tác giả báo cáo tỷ lệ tấn công thành công và độ nhiễu của mẫu đối nghịch trong mỗi phương pháp. Tỷ lệ tấn công thành công (ASR) là phần trăm mẫu đối nghịch được phân loại vào lớp đích (khác lớp gốc). Các trung bình L_1, L_2 và L_∞ của các mẫu đối nghịch thành công cũng được ghi chép báo cáo. Cụ thể, các trường hợp sau được quan tâm:

- **Trường hợp tốt nhất (best case):** tấn công dễ nhất về phương diện nhiễu, trong số các tấn công nhắm tới tất cả các lớp sai nhãn.
- **Trường hợp trung bình (average case):** tấn công nhắm ngẫu nhiên vào 1 lớp sai nhãn.

[3] https://github.com/cleverhans-lab/cleverhans
[4] https://github.com/ysharma1126/EAD-Attack

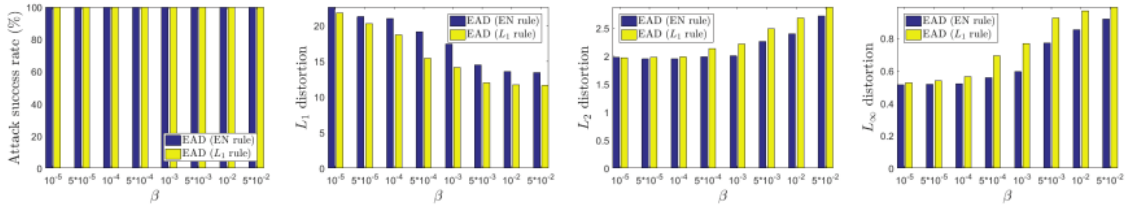
- **Trường hợp xấu nhất (worst case):** tấn công khó nhất về phương diện nhiễu, trong số những tấn công nhắm tới tất cả các lớp sai nhãn.

4.4 Phân tích độ nhạy và luật quyết định

		Best case				Average case				Worst case			
Optimization method	β	ASR	L_1	L_2	L_∞	ASR	L_1	L_2	L_∞	ASR	L_1	L_2	L_∞
COV	0	100	13.93	1.377	0.379	100	22.46	1.972	0.514	99.8	32.3	2.639	0.663
	10^{-5}	100	13.92	1.377	0.379	100	22.66	1.98	0.508	99.5	32.33	2.64	0.663
	10^{-4}	100	13.91	1.377	0.379	100	23.11	2.013	0.517	100	32.32	2.639	0.664
	10^{-3}	100	13.8	1.377	0.381	100	22.42	1.977	0.512	99.9	32.2	2.639	0.664
	10^{-2}	100	12.98	1.38	0.389	100	22.27	2.026	0.53	99.5	31.41	2.643	0.673
EAD (EN rule)	0	100	14.04	1.369	0.376	100	22.63	1.953	0.512	99.8	31.43	2.51	0.644
	10^{-5}	100	13.66	1.369	0.378	100	22.6	1.98	0.515	99.9	30.79	2.507	0.648
	10^{-4}	100	12.79	1.372	0.388	100	20.98	1.951	0.521	100	29.21	2.514	0.667
	10^{-3}	100	9.808	1.427	0.452	100	17.4	2.001	0.594	100	25.52	2.582	0.748
	10^{-2}	100	7.271	1.718	0.674	100	13.56	2.395	0.852	100	20.77	3.021	0.976

Bảng 4.1: So sánh COV và EAD trong việc tìm ra công thức elastic-net trên tập MNIST. ASR là tỷ lệ tấn công thành công. Mặc dù 2 phương pháp trên đều đạt được tỷ lệ tấn công thành công như nhau, COV không hiệu quả trong việc tạo ra các mẫu đối nghịch L_1. Khi β tăng lên, EAD tạo ra các mẫu đối nghịch L_1 ít biến dạng hơn trong khi COV thì không bị ảnh hưởng nhiều khi thay đổi β.

Tác giả kiểm tra sự cần thiết của việc sử dụng thuật toán 1 để giải bài toán tấn công bằng hiệu chỉnh elastic-net trong (3.5) bằng cách so sánh nó với thuật toán COV thuần. Trong tài liệu (Carlini and Wagner 2017b), Carlini và Wagner đã loại bỏ điều kiện ràng buộc $\mathbf{x} \in [0,1]^p$ bằng cách thay $\mathbf{x}$ bằng $\frac{\mathbf{1}+\tanh \mathbf{w}}{2}$ với $\mathbf{w} \in \mathbb{R}^p$ và $\mathbf{1} \in \mathbb{R}^p$ là vector các với các phần tử 1. Thuật toán tối ưu mặc định ADAM (Kingma and Ba 2014) được sử dụng để giải ra nghiệm $\mathbf{w}$ và từ đó tìm được $\mathbf{x}$. Tác giả áp dụng thuật toán COV lên (3.5) và so sánh với EAD trên tập MNIST với các β khác nhau trong Bảng 4.1 ở trên. Mặc dù COV và EAD thu được tỷ lệ tấn công thành công tương tự nhau, người ta quan sát thấy COV không hiệu quả trong việc tạo ra các mẫu đối nghịch L_1. Khi tăng β, EAD tạo ra mẫu đối nghịch L_1 ít nhiễu hơn, trong khi các kết quả L_1, L_2, và L_∞ của COV thì ít chịu ảnh hưởng khi thay đổi β. Khi sử dụng các hàm thư viện TensorFlow như AdaGrad, RMSProp, SGD, người ta cũng thấy nó ít ảnh hưởng bởi β giống như trường hợp của COV. Tác giả cũng chú ý rằng COV không thể dùng phương pháp tối ưu ISTA vì thành phần hàm tanh trong hiệu chỉnh L_1. Sự ít ảnh hưởng của COV cho thấy nó không phù hợp cho tối ưu elastic-net, do nó không hiệu quả cho bài toán tối ưu bằng subgradient (Duchi and Singer 2009). Với EAD, tác giả quan sát thấy sự đánh đổi giữa L_1 với 2 độ đo nhiễu còn lại là L_2, và L_∞. Điều này là do khi tăng β làm tăng độ thưa của nhiễu và do đó tăng L_2, L_∞ . Kết quả tương tự cũng được quan sát thấy với CIFAR10.



Hình 4.1: So sánh luật quyết định EN và L_1 trong EAD trên tập MNIST với nhiều tham số hiệu chỉnh L_1 β (trường hợp trung bình). So sánh với luật chọn EN tại cùng giá trị β thì luật chọn L_1 thu được các mẫu ít nhiễu L_1 hơn, nhưng đổi lại có thể bị nhiễu L_2, L_∞ nhiều hơn.

Ở bảng 4.1, quá trình thuật toán tối ưu chạy để tìm ra mẫu đối nghịch cuối cùng giữa những mẫu đối nghịch thành công dựa trên hàm mất mát elastic-net trong $\{\mathbf{x}^{(k)}\}_{k=1}^{I}$, gọi là luật chọn elastic-net. Ngoài ra, có thể chọn mẫu đối nghịch cuối cùng sao cho L_1 nhỏ nhất, gọi là luật chọn L_1. Hình 4.1 so sánh ASR và các nhiễu trung bình trên 2 luật chọn nói trên với các giá trị khác nhau trên tập dữ liệu MNIST. Cả 2 luật chọn đều cho tỷ lệ thành công ASR 100% với dải giá trị β rộng. Với cùng 1 giá trị β, luật chọn L_1 cho ra các mẫu đối nghịch có nhiễu L_1 nhỏ hơn so với luật chọn EN, trong khi đó lại bị trả giá nhiều nhiễu hơn ở L_2, L_∞. Xu hướng tương tự cũng được quan sát thấy với dữ liệu CIFAR10 (kết quả đầy đủ với cả 2 tập dữ liệu MNIST và CIFAR có trong tài liệu mở rộng). Trong các thí nghiệm tiếp theo, tác giả báo cáo kết quả của EAD với 2 luật chọn và $\beta = 10^{-3}$, vì trên tập MNIST và CIFAR10 giá trị β làm giảm nhiễu L_1 trong khi có thể so sánh L_2, L_∞ với trường hợp $\beta = 0$ (nghĩa là không có hiệu chỉnh L_1).

4.5 Tỷ lệ tấn công thành công và nhiễu trên các tập dữ liệu MNIST, CIFAR10 và ImageNet

	MNIST				CIFAR 10				ImageNet			
Attack method	ASR	L_1	L_2	L_∞	ASR	L_1	L_2	L_∞	ASR	L_1	L_2	L_∞
C&W (L_2)	**100**	22.46	**1.972**	0.514	**100**	13.62	**0.392**	0.044	**100**	232.2	**0.705**	0.03
FGM-L_1	39	53.5	4.186	0.782	48.8	51.97	1.48	0.152	1	61	0.187	0.007
FGM-L_2	34.6	39.15	3.284	0.747	42.8	39.5	1.157	0.136	1	2338	6.823	0.25
FGM-L_∞	42.5	127.2	6.09	0.296	52.3	127.81	2.373	0.047	3	3655	7.102	0.014
I-FGM-L_1	**100**	32.94	2.606	0.591	**100**	17.53	0.502	0.055	77	526.4	1.609	0.054
I-FGM-L_2	**100**	30.32	2.41	0.561	**100**	17.12	0.489	0.054	**100**	774.1	2.358	0.086
I-FGM-L_∞	**100**	71.39	3.472	**0.227**	**100**	33.3	0.68	**0.018**	**100**	864.2	2.079	**0.01**
EAD (EN rule)	**100**	**17.4**	2.001	0.594	**100**	**8.18**	0.502	0.097	**100**	**69.47**	1.563	0.238
EAD (L_1 rule)	**100**	**14.11**	2.211	0.768	**100**	**6.066**	0.613	0.17	**100**	**40.9**	1.598	0.293

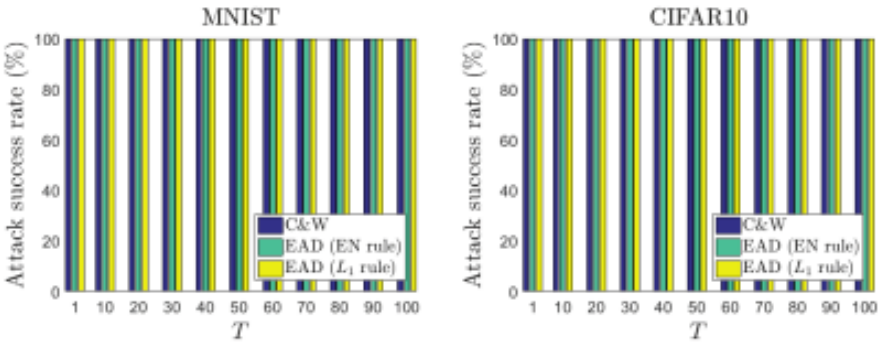
Bảng 4.2: So sánh các tấn công trên các tập dữ liệu MNIST, CIFAR10 và ImageNet. ASR là tỷ lệ tấn công thành công (%). Giá trị nhiễu trong bảng đo được trên giá trị trung bình của các mẫu thành công. EAD, C&W và I-FGM-L_∞ thu được các mẫu ít nhiễu nhất trên các chuẩn L_1,L_2,L_∞ tương ứng. Kết quả đầy đủ được báo cáo trong tài liệu mở rộng.

Tác giả so sánh EAD với các phương pháp đối sánh trên các tiêu chí tỷ lệ tấn công thành công và độ nhiễu khi tấn công mạng DNN huấn luyện bởi MNIST, CIFAR10 và ImageNet. Bảng 4.2 là kết quả thực nghiệm trong trường hợp trung bình. FGM ít tấn công thành công (chỉ số ASR thấp) và chỉ số nhiễu khá lớn so với các phương pháp khác. Trong khi đó, C&W, I-FGM và EAD đều đạt ASR 100%. Ngoài ra, EAD, C&W và I-FGM-L_∞ đạt được các mẫu đối nghịch với ít nhiễu nhất lần lượt trên các chỉ số L_1, L_2 và L_∞. Hơn nữa, EAD tốt hơn I-FGM-L_1. So sánh với I-FGM-L_1, EAD với luật chọn EN giảm nhiễu L_1 xuống xấp xỉ 47% trên tập MNIST, 53% với tập CIFAR10 và 87% với ImageNet. Tác giả cũng báo cáo kết quả quan sát rằng EAD với luật chọn L_1 có thể giảm nhiễu L_1 nhưng làm tăng L_2 và L_∞.

Mặc dù có nhiễu L_2 và L_∞ lớn, các mẫu đối nghịch tạo bởi EAD với luật chọn L_1 có thể đạt ASR 100% trên tất cả các tập dữ liệu. Nghĩa là nhiễu L_2 và L_∞ không đủ để đánh giá sức mạnh của mạng neuron. Hơn nữa, kết quả tấn công trong bảng 4.2 cho thấy EAD có thể thu được 1 tập phân biệt các mẫu đối nghịch khác biệt cơ bản với các mẫu dựa trên L_2 và L_∞. Tương tự phương pháp C&W và I-FGM, các mẫu đối nghịch sinh bởi EAD đều khó phân biệt bằng mắt thường.

4.6 Phá vỡ DNN chắt lọc phòng thủ

Ngoài ra, để tấn công DNN không phòng thủ bằng các mẫu đối nghịch, tác giả đã cho thấy EAD có thể phá vỡ DNN chắt lọc phòng thủ. Chắt lọc phòng thủ (Papernot et al. 2016b) là kĩ thuật phòng thủ chuẩn trong đó mạng được huấn luyện lại bằng các xác suất từng lớp được dự đoán bởi một mạng gốc (chứ không phải bằng nhãn ground truth), gọi là nhãn mềm và người ta dùng tham số nhiệt T trong lớp softmax để tăng cường sức mạnh của nó chống lại nhiễu đối nghịch. Tương tự như phương thức tấn công tiên tiến C&W, hình 4.2 cho thấy EAD có thể thu được ASR 100% với các giá trị T khác nhau trên tập MNIST và CIFAR10. Hơn nữa, vì công thức tấn công C&W là trường hợp đặc biệt của công thức EAD trong (3.5) khi $\beta = 0$, sự thành công của EAD trong việc phá vỡ chắt lọc phòng thủ gợi ý 1 cách mới để tạo ra các mẫu đối nghịch hiệu quả là dùng các tham số β khác nhau cho hiệu chỉnh L_1. Kết quả đầy đủ của tấn công ở trong tài liệu mở rộng.



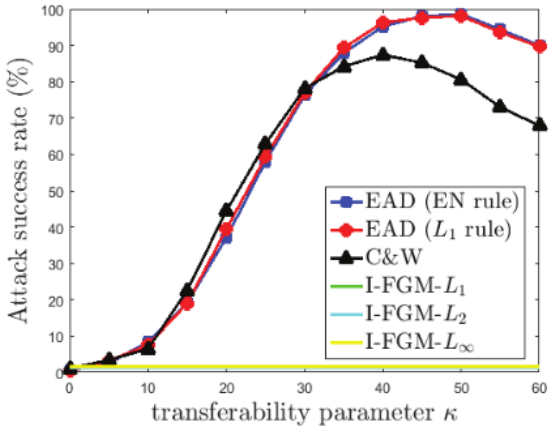
Hình 4.2: ASR (trường hợp trung bình) của C&W và EAD trên tập MNIST và CIFAR10 với các tham số nhiệt T khác nhau cho chất lọc phòng thủ. Cả 2 phương pháp đều phá vỡ thành công chất lọc phòng thủ.

4.7 Cải thiện khả năng chuyển giao tấn công

Trong tài liệu (Carlini and Wagner 2017b) đã chỉ ra rằng tấn công C&W có khả năng chuyển giao tốt hơn từ 1 mạng không phòng thủ sang 1 mạng chất lọc phòng thủ bằng cách tối ưu tham số độ tin cậy κ trong (3.2). Theo (Carlini and Wagner 2017b), tác giả sử dụng cùng bộ tham số của chuyển giao tấn công trên tập MNIST, vì MNIST là tập dữ liệu khó tấn công nhất về phương diện nhiễu trung bình trên mỗi điểm ảnh như trong bảng 4.2 ở trên.

Cố định κ, các mẫu đối nghịch được sinh ra từ mạng gốc (không phòng thủ) được sử dụng để tấn công mạng chất lọc phòng thủ với tham số nhiệt $T = 100$ (Papernot et al. 2016b). Tỷ lệ tấn công thành công (ASR) của các phương pháp EAD, C&W và I-FGM được trình bày trong hình 4.3. Khi $\kappa = 0$, tất cả các phương pháp đều cho ASR thấp và do đó không tạo được các mẫu đối nghịch có thể chuyển giao. Tỷ lệ ASR của phương pháp EAD và C&W tăng lên khi đặt $\kappa > 0$, trong khi đó tỷ lệ ASR của I-FGM thấp (dưới 2%) do tấn công này không có các tham số tương tự để có thể chuyển giao.

Chú ý rằng, EAD có thể thu được ASR gần đạt 99% khi $\kappa = 50$, trong khi đó ASR cao nhất mà phương pháp C&W đạt được là gần 88% khi $\kappa = 40$. Chứng tỏ rằng, khả năng chuyển giao tấn công tăng lên khi sử dụng các mẫu đối nghịch được tạo từ EAD, điều này là do thuật toán ISTA trong phương trình 3.6 mạnh hơn so với tấn công C&W qua phương pháp biến đổi theo ngưỡng. Tác giả cũng thấy rằng khi đặt κ quá lớn sẽ làm giảm ASR của các chuyển giao tấn công cả bằng phương pháp EAD và lẫn C&W, do thuật toán tối ưu không tìm được mẫu đối nghịch có thể làm tối thiểu hàm mất mát f trong (3.2) khi κ lớn. Kết quả đầy đủ về khả năng chuyển giao tấn công được báo cáo trong tài liệu mở rộng.



Hình 4.3: Khả năng chuyển giao tấn công (trường hợp trung bình) từ mạng không phòng thủ sang mạng chất lọc phòng thủ trên tập dữ liệu MNIST với các tham số κ khác nhau. EAD có thể đạt ASR gần 99% khi $\kappa = 50$, trong khi ASR lớn nhất của C&W là gần 88% khi $\kappa = 40$.

4.8 Huấn luyện đối nghịch bổ sung

Attack method	Adversarial training	Average case			
		ASR	L_1	L_2	L_∞
C&W (L_2)	None	100	22.46	1.972	0.514
	EAD	100	26.11	2.468	0.643
	C&W	100	24.97	2.47	0.684
	EAD + C&W	100	27.32	2.513	0.653
EAD (L_1 rule)	None	100	14.11	2.211	0.768
	EAD	100	17.04	2.653	0.86
	C&W	100	15.49	2.628	0.892
	EAD + C&W	100	16.83	2.66	0.87

Bảng 4.3: Huấn luyện đối nghịch sử dụng tấn công C&W và EAD (luật L_1) trên tập dữ liệu MNIST. ASR là tỷ lệ tấn công thành công. Kết hợp các mẫu L_1 bổ sung cho huấn luyện đối nghịch và tăng cường độ khó của tấn công về phương diện nhiễu. Kết quả đầy đủ có trong tài liệu mở rộng.

Để xem xét sự khác biệt giữa các mẫu đối nghịch L_1 và L_2, tác giả kiểm tra hiệu quả huấn luyện đối nghịch trên tập MNIST. Họ chọn ngẫu nhiên 1000 ảnh từ tập huấn luyện và sử dụng tấn công C&W và EAD (luật chọn L_1) để tạo ra 9000 mẫu đối nghịch cho tất cả các nhãn sai với mỗi phương pháp. Sau đó, họ thêm vào tập huấn luyện các mẫu đối nghịch này để huấn luyện lại mạng và kiểm tra sức mạnh của nó trên tập kiểm thử, kết quả tổng hợp trong bảng 4.3. Để huấn luyện đối nghịch với 1 phương pháp nào đó, mặc dù cả 2 tấn công đều đạt tỷ lệ thành công trung bình 100%, mạng vẫn có sức chịu đựng tốt hơn trước nhiễu đối nghịch vì các chỉ số nhiễu đều đã tăng lên đáng kể so với trường hợp chưa huấn luyện. Tác giả cũng thấy kết

hợp huấn luyện đối nghịch bằng EAD và C&W có thể làm tăng hơn nữa nhiễu L_1 và L_2 khi tấn công bằng C&W, và tăng nhiễu L_2 khi tấn công bằng EAD. Điều đó chứng tỏ, các mẫu đối nghịch L_1 được tạo bởi EAD có thể bổ sung huấn luyện đối nghịch.

4.9 Nhận xét

4.9.1 Thời gian tấn công

Trong quá trình thực nghiệm, chúng tôi nhận thấy rằng mặc dù tỉ lệ tấn công thành công (ASR) của phương pháp EAD là rất tốt (100% trên cả 3 tập dữ liệu) cùng với nhiễu L_1 nhỏ, tuy nhiên điều này đánh đổi bằng thời gian tấn công rất lâu. Chúng tôi đã tiến hành tấn công bằng máy tính cá nhân (CPU Intel i7 12700, 32GB RAM, GPU RTX 3070Ti) và cho ra thời gian tấn công như sau.

Phương pháp	EAD-EN	FGM-L1	FGM-L2	IFGM-L1	IFGM-L2
Thời gian (s)	42810	893	1034	3366	9922

Bảng 4.4: Thời gian tấn công của các thuật toán trên tập cifar

Do giới hạn phần cứng, chúng tôi tiến hành tấn công trên tập cifar với 100 mẫu đối nghịch thay vì 1000 mẫu như bài báo gốc, $\beta = 10^{-2}$. Như đã chỉ ra trên bảng 4.4 thời gian tấn công của thuật toán EAD với luật EN là khoảng 11 giờ và gấp khoảng 48 lần khi so sánh với thuật toán nhanh nhất là FGM-L1.

4.9.2 Hướng nghiên cứu mở rộng

Thuật toán tổng quát

Trong báo cáo này, tác giả đã sử dụng thuật toán FISTA (phiên bản nhanh hơn của ISTA) để cực tiểu hóa hàm $f(\mathbf{x}, t)$ trong phương trình 3.5. Để tính được gradient ∇g ta cần có ràng buộc hàm mất mát của mô hình gốc f phải trơn. Tuy nhiên trong thực tế, không phải tất cả các hàm mất mát đều trơn! Trong bài báo này, tác giả lấy gradient được tính bởi mạng DNN. Thực chất trong code triển khai (sử dụng TensorFlow), gradient được tính bởi mạng là sub-gradient (khi sử dụng hàm kích hoạt là ReLU, vốn dĩ không phải một hàm trơn). Chính vì vậy ta cần một thuật toán tổng quát hơn để giải bài toán (3.5) với một hàm f (lồi) bất kì mà không cần ràng buộc về tính trơn của nó. (Shao, Weijia, Fikret Sivrikaya, & Sahin Albayrak 2022) đã giới thiệu một thuật toán hiệu quả để cực tiểu hóa hàm mục tiêu tổ hợp bằng cập nhật mũ. Bài báo đưa ra thuật toán cho hiệu chỉnh elastic-net và thực

nghiệm trên tập CIFAR. Thuật toán này được so sánh với FISTA và hội tụ nhanh hơn sau 200 vòng lặp. Chúng tôi mong muốn cài đặt thuật toán này để tạo mẫu đối nghịch sau đó so sánh tỉ lệ thành công, các độ đo L_1, L_2 và L_∞ cùng với thời gian chạy thực tế so với EAD.

Tấn công nhận diện khuôn mặt

Chúng tôi mong muốn sử dụng EAD để tấn công hệ thống nhận diện bằng khuôn mặt. Mục tiêu là một tấn công leo thang đặc quyền khi sử dụng khuôn mặt của một nhân viên với quyền thấp hơn để đánh lừa mô hình nhận diện thành một nhân viên với quyền cao hơn. Tấn công này thúc đẩy việc hệ thống nhận diện phải được phòng thủ một cách cẩn thận. Gần đây framework foolbox [5] cũng đã tích hợp sẵn tấn công EAD cũng như C&W để sinh ra các mẫu đối nghịch. Foolbox đã cài đặt EAD để phù hợp với cả Tensorflow lần PyTorch.

[5] https://github.com/bethgelab/foolbox

5 Kết luận

Nhóm tác giả đã đề xuất mô hình tấn công bằng hiệu chỉnh elastic-net để tạo ra các mẫu đối nghịch trong tấn công DNN. Các kết quả thực nghiệm trên các tập dữ liệu MNIST, CIFAR10 và ImageNet cho thấy các mẫu L_1 tạo bởi EAD có thể đạt được tỷ lệ thành công tương đương với các phương pháp tấn công tiên tiến dựa trên L_2 và L_∞ khi phá vỡ mạng không phòng thủ và phòng thủ chưng cất. Ngoài ra, EAD có thể cải thiện khả năng chuyển giao tấn công và huấn luyện đối nghịch bổ sung. Các kết quả của nhóm tác giả đã chứng minh hiệu quả của EAD và đưa ra hướng mới sử dụng mẫu đối nghịch L_1 trong việc huấn luyện đối nghịch và tăng cường an ninh cho DNN.

Tài liệu tham khảo

[1] Chen, Pin-Yu, et al. "Ead: elastic-net attacks to deep neural networks via adversarial examples." Proceedings of the AAAI conference on artificial intelligence. Vol. 32. No. 1. 2018.

[2] Shao, Weijia, Fikret Sivrikaya, and Sahin Albayrak. "Optimistic Optimisation of Composite Objective with Exponentiated Update." (2022).

[3] Beck, Amir, and Marc Teboulle. "A fast iterative shrinkage-thresholding algorithm for linear inverse problems." SIAM journal on imaging sciences 2.1 (2009): 183-202.

[4] Candès, Emmanuel J., and Michael B. Wakin. "An introduction to compressive sampling." IEEE signal processing magazine 25.2 (2008): 21-30.

[5] Carlini, Nicholas, and David Wagner. "Adversarial examples are not easily detected: Bypassing ten detection methods." Proceedings of the 10th ACM workshop on artificial intelligence and security. 2017.

[6] Carlini, Nicholas, and David Wagner. "Towards evaluating the robustness of neural networks." 2017 ieee symposium on security and privacy (sp). Ieee, 2017.

[7] Dong, Yinpeng, et al. "Towards interpretable deep neural networks by leveraging adversarial examples." arXiv preprint arXiv:1708.05493 (2017).

[8] Duchi, John, and Yoram Singer. "Efficient online and batch learning using forward backward splitting." The Journal of Machine Learning Research 10 (2009): 2899-2934.

[9] Evtimov, Ivan, et al. "Robust physical-world attacks on machine learning models." arXiv preprint arXiv:1707.08945 2.3 (2017): 4.

[10] Feinman, Reuben, et al. "Detecting adversarial samples from artifacts." arXiv preprint arXiv:1703.00410 (2017).

[11] Fu, Haoying, et al. "Efficient minimization methods of mixed l2-l1 and l1-l1 norms for image restoration." SIAM Journal on Scientific computing 27.6 (2006): 1881-1902.

[12] Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples." arXiv preprint arXiv:1412.6572 (2014).

[13] Grosse, Kathrin, et al. "On the (statistical) detection of adversarial examples." arXiv preprint arXiv:1702.06280 (2017).

[14] Hinton, Geoffrey, Oriol Vinyals, and Jeff Dean. "Distilling the knowledge in a neural network (2015)." arXiv preprint arXiv:1503.02531 2 (2015).

[15] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).

[16] Koh, Pang Wei, and Percy Liang. "Understanding black-box predictions via influence functions." International conference on machine learning. PMLR, 2017.

[17] Kurakin, Alexey, Ian J. Goodfellow, and Samy Bengio. "Adversarial examples in the physical world." Artificial intelligence safety and security. Chapman and Hall/CRC, 2018. 99-112.

[18] Kurakin, Alexey, Ian Goodfellow, and Samy Bengio. "Adversarial machine learning at scale." arXiv preprint arXiv:1611.01236 (2016).

[19] Liu, Yanpei, et al. "Delving into transferable adversarial examples and black-box attacks." arXiv preprint arXiv:1611.02770 (2016).

[20] Lu, J.; Issaranon, T.; and Forsyth, D. 2017. Safetynet: Detecting and rejecting adversarial examples robustly

[21] Madry, Aleksander, et al. "Towards deep learning models resistant to adversarial attacks." arXiv preprint arXiv:1706.06083 (2017).

[22] Moosavi-Dezfooli, Seyed-Mohsen, et al. "Universal adversarial perturbations." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[23] Moosavi-Dezfooli, Seyed-Mohsen, Alhussein Fawzi, and Pascal Frossard. "Deepfool: a simple and accurate method to fool deep neural networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[24] Papernot, Nicolas, et al. "The limitations of deep learning in adversarial settings." 2016 IEEE European symposium on security and privacy (EuroS&P). IEEE, 2016.

[25] Papernot, Nicolas, et al. "Distillation as a defense to adversarial perturbations against deep neural networks." 2016 IEEE symposium on security and privacy (SP). IEEE, 2016.

[26] Papernot, Nicolas, et al. "Practical black-box attacks against machine learning." Proceedings of the 2017 ACM on Asia conference on computer and communications security. 2017.

[27] Parikh, Neal, and Stephen Boyd. "Proximal algorithms." Foundations and trends® in Optimization 1.3 (2014): 127-239.

[28] Szegedy, Christian, et al. "Intriguing properties of neural networks." arXiv preprint arXiv:1312.6199 (2013).

[29] Szegedy, Christian, et al. "Rethinking the inception architecture for computer vision." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[30] Tramèr, Florian, et al. "Ensemble adversarial training: Attacks and defenses." arXiv preprint arXiv:1705.07204 (2017).

[31] Xu, Weilin, David Evans, and Yanjun Qi. "Feature squeezing: Detecting adversarial examples in deep neural networks." arXiv preprint arXiv:1704.01155 (2017).

[32] Zantedeschi, Valentina, Maria-Irina Nicolae, and Ambrish Rawat. "Efficient defenses against adversarial attacks." Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security. 2017.

[33] Zheng, Stephan, et al. "Improving the robustness of deep neural networks via stability training." Proceedings of the ieee conference on computer vision and pattern recognition. 2016.

[34] Zou, Hui, and Trevor Hastie. "Regularization and variable selection via the elastic net." Journal of the royal statistical society: series B (statistical methodology) 67.2 (2005): 301-320.