**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA KHOA HỌC VÀ KỸ THUẬT THÔNG TIN**





**BÁO CÁO ĐỒ ÁN CUỐI KỲ**

**TÌM HIỂU GAN ĐỂ PHÁT SINH MÃ ĐỘC PE**

**Sinh viên thực hiện:**

Lê Công Quốc Huy – 21522145 - Trưởng nhóm

Dương Uy Quan – 21521323

**Giảng viên:**

Nguyễn Tấn Cầm

Thành phố Hồ Chí Minh, 30 tháng 11 năm 2023

**NHẬN** **XÉT** **CỦA** **GIÁO** **VIÊN** **HƯỚNG** **DẪN**

*……., ngày tháng……năm 20…*

Người nhận xét

(Ký tên và ghi rõ họ tên)

## **MỤC LỤC**

[**MỤC LỤC 2**](#_av88fqbxipfd)

[**DANH MỤC HÌNH ẢNH 4**](#_3snezrh4blr5)

[**DANH MỤC BẢNG 4**](#_rrjbuojqozfu)

[**ABSTRACT 5**](#_hrk3sbabwoeg)

[**I. INTRODUCTION. 5**](#_ygvyjp1iwgwp)

[**II. RELATED WORKS. 6**](#_m0bchno81c3m)

[**III. PROPOSE SYSTEM 9**](#_1ltay4bp1ihn)

[1. Thu thập và xử lý dữ liệu. 10](#_sy78rcin8m1p)

[2. Huấn luyện và thử nghiệm 10](#_a88qdkdbg55f)

[**IV. ĐÁNH GIÁ MÔ HÌNH 13**](#_k0ris1k0n0d7)

[1. Kết Quả. 13](#_85e6jxm6pft8)

[2. Đồ thị hàm mất mát (loss function). 14](#_g5jtxugs9l7p)

[3. Đánh giá. 15](#_87vzm181b8wc)

[**V. KẾT LUẬN 17**](#_xre0rqirdebp)

[**VI. TÀI LIỆU THAM KHẢO 18**](#_ha219njyghwf)

[**REFERENCES 19**](#_re7meereuwt6)

## **DANH MỤC HÌNH ẢNH**

[Hình 1: Quy trình 9](#_nzsyvonuot5w)

[Hình 2: Mô hình GAN 10](#_u373ca6uafvh)

[Hình 3: Mô hình huấn luyện bằng DCGAN 11](#_16t2s69v65e0)

[Hình 4: Generator và Discriminator của DCGAN 11](#_d9xb9l3ne0o2)

[Hình 5: 64 hình ảnh mã độc được sinh ra 13](#_1httzbivnt5t)

[Hình 6: Đồ thị loss function 14](#_x8jdqn3fmx1l)

[Hình 7: Đồ thị đánh giá độ chính xác của ảnh sinh ra. 15](#_zaubhhfmp8fg)

## **DANH MỤC BẢNG**

[Bảng 1: Tham số sử dụng trong mô hình 12](#_omi45fnyg5n0)

## **ABSTRACT**

*Dựa theo khảo sát [1] của Ani Petrosyan, cho tới năm 2022, số lượng các cuộc tấn công mã độc đã giảm đi rất nhiều so với thời điểm năm 2018. Tuy vậy số lượng các cuộc tấn công vẫn dừng lại ở con số 5 tỷ ở năm 2022. Cùng với đó là sự phát triển ngày càng tinh vi và phức tạp của Malware [2] đã khiến cho vấn đề về phát hiện mã độc cũng ngày càng trở nên quan trọng, cấp thiết. Những năm trở lại đây học máy (ML) đã vô cùng phát triển, trở thành thành phần quan trọng trong công cuộc phát hiện Malware trong các hệ thống hiện đại. Tuy nhiên, dù cho đã mang lại nhiều đột phá mới trong việc phát hiện mã độc, các kỹ thuật ứng dụng học máy vẫn rất dễ bị qua mặt bởi các cuộc tấn công đối* địch *[3]. Do đó, trong bài báo cáo này, chúng tôi trình bày về một kỹ thuật có khả năng đánh lừa các hệ thống ML hiện tại bằng cách tạo ra dữ liệu gần như giống với thực tế có tên Generative Adversarial Network (GAN).*

## **INTRODUCTION.**

Hiện nay, malware đã phát triển rất mạnh về cả độ tinh vi, chủng loại, số lượng. Do đó đòi hỏi một hệ thống có khả năng tự phát giác mã độc. Và vài năm trở lại đây, việc áp dụng ML vào phát hiện mã độc đã bùng nổ, thỏa mãn nhu cầu nêu trên. Các kỹ thuật phổ biến có thể kể đến hiện nay có: Feature-based ML, Network-based ML, Deep Learning-based ML, Adversarial ML…[4] Các kỹ thuật này đều đã mang lại những đóng góp to lớn cũng như là thành phần quan trọng của hệ thống an ninh trong việc nhận biết malware. Tuy ML đã đem lại nhiều đột phá nhưng lại bị qua mặt khi người tấn công triển khai các cuộc tấn công mẫu đối địch (Adversarial Example Attack - AE). Có thể lý giải, để thực hiện hành vi tấn công này, kẻ tấn công sẽ thay đổi hoặc thêm vào để thay đổi số lượng nhỏ các thông tin, đặc trưng đầu vào một mô hình ML, mục tiêu là đánh lừa và qua mặt mô hình ML. Các cuộc tấn công nêu trên thường rất khó nhận biết bởi con người. Nó thành công là do lợi dụng sự nhạy cảm đối với các thay đổi nhỏ của các mô hình ML. Từ những lí do nêu trên, việc yêu cầu có một kỹ thuật, mô hình để cải thiện cho các mô hình ML là cần thiết. Qua tìm hiểu chúng tôi đã nhận ra, mô hình GAN – với khả năng sinh ra các bộ dữ liệu mới có thể qua mặt các mô hình ML là phù hợp với nhu cầu nêu trên. Việc áp dụng GAN là nhằm tạo bộ dữ liệu dùng để huấn luyện các mô hình ML với mục tiêu xây dựng các hệ thống chống tấn công, cũng như tăng cường khả năng chống tấn công của các mô hình máy học.

GAN – Viết đầy đủ là Generative Adversarial Networks (Mô hình sinh đối nghịch). Cấu tạo của GAN là từ hai mô hình Generator (Tạo sinh) và Discriminator (Phân biệt). Chức năng của Generator là cố gắng tạo ra các dữ liệu sao cho giống như thực tế nhất có thể. Còn Discriminator cố gắng phân biệt dữ liệu trên với dữ liệu thực. Mục tiêu của mạng tạo sinh ở đây là làm thế nào để đánh lừa mạng phân biệt bằng dữ liệu do nó sinh ra. Vai trò của mạng phân biệt là báo cho mạng tạo sinh biết dữ liệu nó tạo ra chưa giống hay giống thực. Từ đó cho ra kết quả là một mô hình với khả năng tạo sinh dữ liệu mới [5].

Phần còn lại của báo cáo này sẽ theo trình tự sau: Chương tiếp theo sẽ là danh sách các **nghiên cứu liên quan** mà chúng tôi đã tham khảo. Chương 3 là các module được áp dụng cho mô hình, Chương 4 là phần chúng tôi đánh giá mô hình đã được nêu ra. Chương 5 gồm kết luận, hướng phát triển. Chương cuối là danh sách tài liệu tham khảo.

## **RELATED WORKS.**

Hiện nay, các loại mã độc đang ngày càng trở nên đa dạng hơn. Có nhiều mã độc mới đã được sinh ra đòi hỏi hệ thống phải sớm phát hiện để tìm cách khắc phục. Ở phần này, chúng tôi sẽ trình bày về một số phương pháp học máy để đào tạo hệ thống trở nên tốt hơn.

Theo như Weiwei Hu and Ying Tan đã trình bày trong [3]. Họ đã đề xuất một thuật toán dựa trên mạng đối thủ tổng quát (GAN) có tên MalGAN để tạo ra các ví dụ về phần mềm độc hại đối nghịch, có khả năng vượt qua các mô hình phát hiện dựa trên máy học hộp đen. MalGAN sử dụng trình phát hiện thay thế để phù hợp với hệ thống phát hiện phần mềm độc hại hộp đen. Mạng tổng quát được đào tạo để giảm thiểu xác suất độc hại của các mẫu đối nghịch được tạo ra bởi trình phát hiện thay thế được dự đoán. Điểm vượt trội của MalGAN so với các thuật toán tạo mẫu đối thủ dựa trên gradient truyền thống là MalGAN có thể giảm tỷ lệ phát hiện xuống gần bằng 0 và khiến phương pháp phòng thủ dựa trên đào tạo lại chống lại các mẫu đối thủ khó hoạt động.

**Trong bài [6],** Alec Radford, Luke Metz, Soumith Chintala đã giới thiệu một lớp CNN được gọi là mạng đối nghịch tạo ra tích chập sâu (DCGAN), có những hạn chế về kiến ​​trúc nhất định và chứng minh rằng chúng là ứng cử viên nặng ký cho việc học không giám sát. Đào tạo về các bộ dữ liệu hình ảnh khác nhau, chúng tôi đưa ra bằng chứng thuyết phục rằng cặp đối thủ tích chập sâu của chúng tôi học được hệ thống phân cấp biểu diễn từ các phần đối tượng đến cảnh trong cả bộ tạo và bộ phân biệt đối xử.

Aeryn Dunmore , Julian Jang-Jaccard , Fariza Sabrina and Jin Kwak đã khảo sát nghiên cứu và tài liệu hiện tại về việc sử dụng Mạng đối thủ sáng tạo trong không gian có vấn đề về phần mềm độc hại. Điều này được thực hiện với hy vọng rằng người đọc có thể hiểu được tổng thể về những gì mô hình Đối nghịch Sáng tạo cung cấp cho lĩnh vực này và những lĩnh vực nào trong nghiên cứu phần mềm độc hại mà mô hình này được sử dụng tốt nhất. Nó bao gồm các khảo sát liên quan hiện tại, các danh mục GAN khác nhau và đưa ra kết quả của nghiên cứu gần đây về việc tối ưu hóa GAN cho các chủ đề khác nhau cũng như các hướng khám phá trong tương lai. Các nghiên cứu này được họ trình bày trong [7].

Trong bài viết [8], Huy Nguyen, Fabio Di Troia, Genya Ishigaki & Mark Stamp đã trích xuất các tính năng từ các tệp thực thi của phần mềm độc hại và thể hiện chúng dưới dạng hình ảnh bằng nhiều cách tiếp cận khác nhau. Sau đó, họ tập trung vào các mạng đối thủ tổng quát (GAN) để phân loại nhiều lớp và so sánh kết quả GAN với các kỹ thuật học máy phổ biến khác, bao gồm máy vectơ hỗ trợ (SVM), XGBoost và máy Boltzmann bị hạn chế (RBM). Họ nhận thấy rằng bộ phân biệt AC-GAN nhìn chung có tính cạnh tranh với các kỹ thuật học máy khác.Họ cũng đánh giá tiện ích của mô hình tổng hợp GAN đối với các cuộc tấn công bất lợi trong việc phát hiện phần mềm độc hại dựa trên hình ảnh. Mặc dù hình ảnh do AC-GAN tạo ra rất ấn tượng về mặt trực quan nhưng chúng có thể dễ dàng được phân biệt với hình ảnh phần mềm độc hại thực sự bằng cách sử dụng bất kỳ kỹ thuật học tập nào.

**Ian J. Goodfellow** và các cộng sự trong bài báo nghiên cứu [9] đề xuất một khung mới để ước tính các mô hình tổng quát thông qua quy trình đối nghịch, trong đó chúng tôi huấn luyện đồng thời hai mô hình: mô hình tổng quát G nắm bắt phân phối dữ liệu và mô hình phân biệt đối xử D ước tính xác suất một mẫu đến từ dữ liệu huấn luyện thay vì hơn G. Quy trình huấn luyện cho G là tối đa hóa xác suất D mắc lỗi. Khung này tương ứng với trò chơi minimax hai người chơi. Trong không gian các hàm tùy ý G và D, tồn tại một nghiệm duy nhất, với G khôi phục phân bố dữ liệu huấn luyện và D bằng 1 2 ở mọi nơi. Trong trường hợp G và D được xác định bởi các perceptron đa lớp, toàn bộ hệ thống có thể được huấn luyện bằng phương pháp lan truyền ngược.

Các tác giả Doan Minh Trung, Nguyen Hoang Kim Ngan, Nghi Hoang Khoa , Phan The Duy, Van-Hau Pham và Nguyen Tan Cam, (Corresponding author) đã đề xuất AAGAN [10], một hệ thống tạo phần mềm độc hại Android tự động dựa trên Mạng đối thủ sáng tạo (GAN) có thể đánh lừa thành công các trình phát hiện ML gần đây. Kết quả thử nghiệm chỉ ra rằng các AE do hệ thống tạo ra có thể lật ngược dự đoán của các thuật toán phát hiện tiên tiến trong 99% trường hợp sử dụng tập dữ liệu trong thế giới thực. Để chống lại các cuộc tấn công của AE, các tác giả đã sử dụng dữ liệu được sinh ra để đào tạo lại cho mô hình và nó đã giúp cho hệ thống tốt hơn.

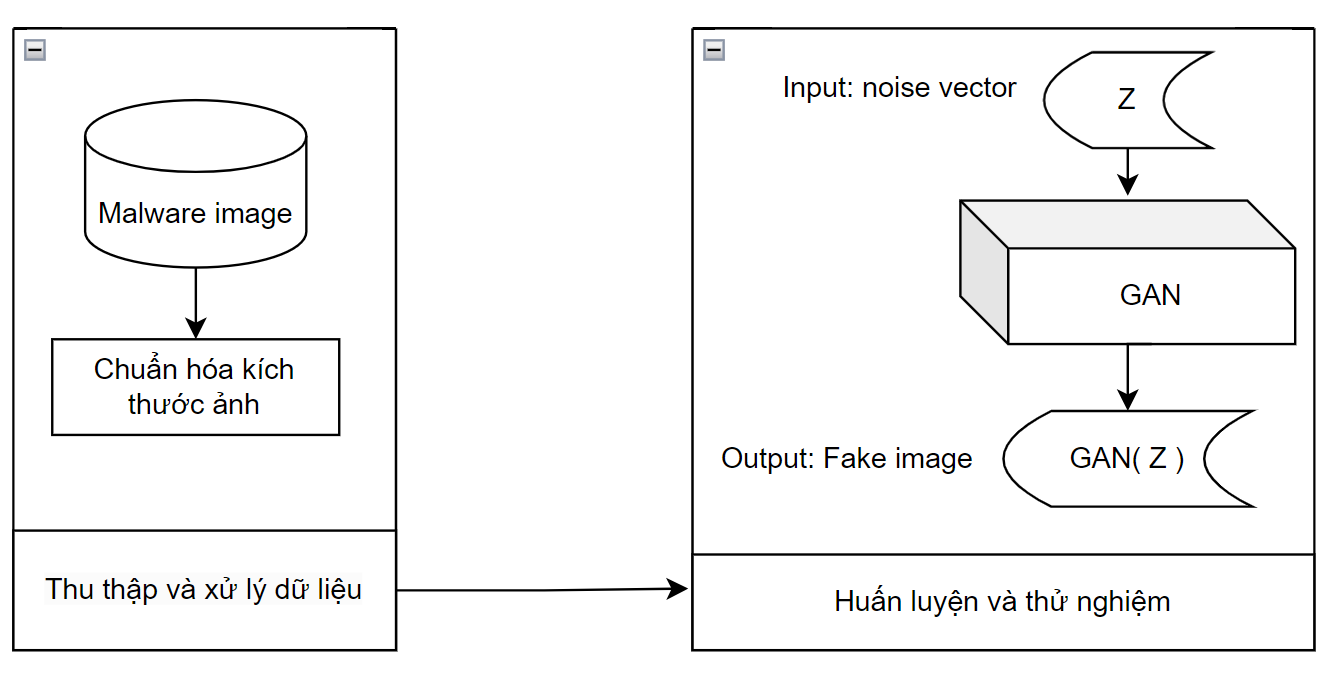
Tim Salimans cùng cộng sự của mình đã thực hiện nghiên cứu [11]. Họ tập trung vào hai ứng dụng của GAN: học bán giám sát và tạo ra các hình ảnh mà con người thấy thực tế về mặt trực quan. Không giống như hầu hết các công việc về mô hình tổng quát, mục tiêu chính không phải là đào tạo một mô hình có khả năng kiểm tra dữ liệu cao và không yêu cầu mô hình có thể học tốt mà không cần sử dụng bất kỳ nhãn nào. Bằng cách sử dụng các kỹ thuật mới, mô hình đạt được kết quả tiên tiến trong phân loại bán giám sát trên MNIST, CIFAR-10 và SVHN. Các hình ảnh được tạo ra có chất lượng cao như được xác nhận bằng thử nghiệm Turing trực quan: mô hình tạo ra các mẫu MNIST mà con người không thể phân biệt được với dữ liệu thực và các mẫu CIFAR-10 có tỷ lệ lỗi do con người là 21,3%.

Trong bài viết [12], nhóm tác giả Mehdi Mirza, Simon Osindero đã giới thiệu phiên bản có điều kiện của mạng đối thủ tổng quát, có thể được xây dựng bằng cách đơn giản cung cấp dữ liệu, y, mà họ muốn điều kiện hóa cho cả bộ tạo và bộ phân biệt đối xử. Họ chứng tỏ rằng mô hình này có thể tạo ra các chữ số MNIST dựa trên nhãn lớp. Họ cũng minh họa cách sử dụng mô hình này để tìm hiểu mô hình đa phương thức và cung cấp các ví dụ sơ bộ về ứng dụng gắn thẻ hình ảnh, trong đó họ đã chứng minh cách phương pháp này có thể tạo ra các thẻ mô tả không phải là một phần của nhãn đào tạo.

## **PROPOSE SYSTEM**

Hệ thống được xây dựng dựa trên 2 bước chính:

1. Thu thập và xử lý dữ liệu.
2. Huấn luyện và thử nghiệm.



#### Hình 1: Quy trình

### **Thu thập và xử lý dữ liệu.**

Dữ liệu được thu thập tại[13], gồm các tập tin hình ảnh đã được phân thành 31 lớp.

Dữ liệu bào gồm 31 class với khoảng 9868 file

Các file ảnh này có kích thước khác nhau nên chúng tôi đã chuẩn hóa tất cả về 1 kích thước là 64x64

Link đến dataset tại [đây](https://drive.google.com/drive/folders/11mGZ2gjO1vOgLzNCxUI6ssnD3l_8_MB8?usp=sharing)

### **Huấn luyện và thử nghiệm**

**2.1.** **Môi trường:**

Chúng tôi tiến hành huấn luyện mô hình trên phần mềm VScode.

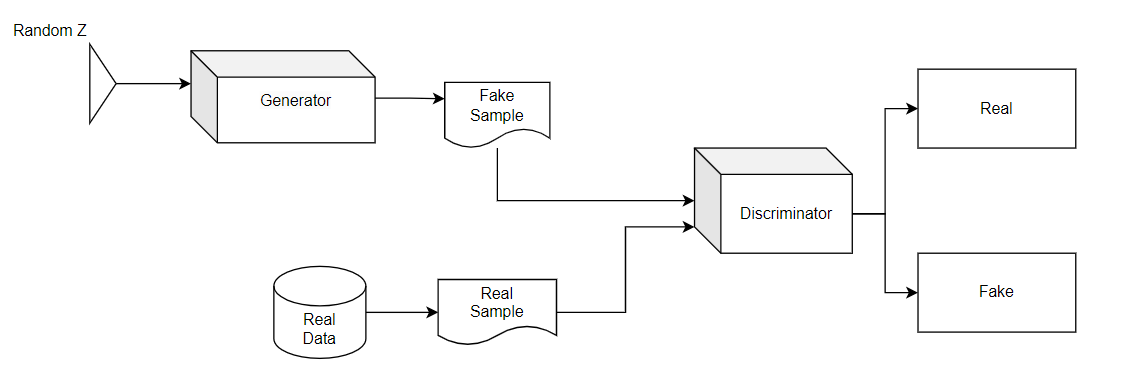
Chạy trên môi trường python 3.10.10.

Sử dụng thư viện Pytorch để huấn luyện mô hình.

**2.2. Huấn luyện**

Mô hình sử dụng ở bài này là mô hình Deep Convolutional Generative Adversarial Network ( gọi tắt là DCGAN). Là một kiến trúc mạng nơ-ron sử dụng trong mô hình GAN (Generative Adversarial Network) để sinh ra ảnh mới từ một không gian ngẫu nhiên.

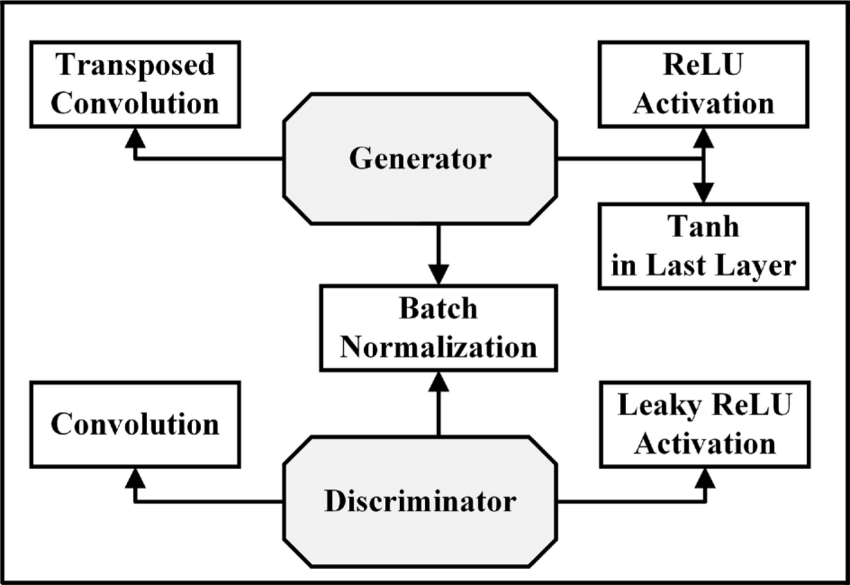
DCGAN là một sự mở rộng của GAN thông thường kết hợp với mô hình CNN. Sử dụng các lớp tích chập (convolutional layers) để xử lý dữ liệu hình ảnh. DCGAN cũng được cấu tạo từ 2 mạng là Generator và Discriminator giống như GAN.



#### Hình 2: Mô hình GAN

Khi generator nhận được vector ngẫu nhiên, nó sẽ ánh xạ vector này từ không gian tiềm ẩn (latent space) thành không gian ảnh thông qua các lớp mạng nơ-ron. Trong quá trình huấn luyện, mô hình học cách điều chỉnh các trọng số của generator để tạo ra các ảnh giả mà có cấu trúc tương tự với ảnh trong dữ liệu huấn luyện. Qua nhiều lần cập nhật trọng số và tối thiểu hóa hàm mất mát, generator trở nên ngày càng tốt hơn trong việc tạo ra các ảnh giả có tính chất tương tự với ảnh thật.

Tuy generator không biết trước thông tin chi tiết về các ảnh cụ thể trong dữ liệu huấn luyện, nhưng nó học cách tái tạo các đặc trưng chung của ảnh thông qua việc xem xét các mẫu dữ liệu huấn luyện và phản hồi từ discriminator. Qua đó, generator có khả năng tạo ra các ảnh giả có tính chất tương tự với ảnh thật, dựa trên các đặc trưng đã học từ dữ liệu huấn luyện.

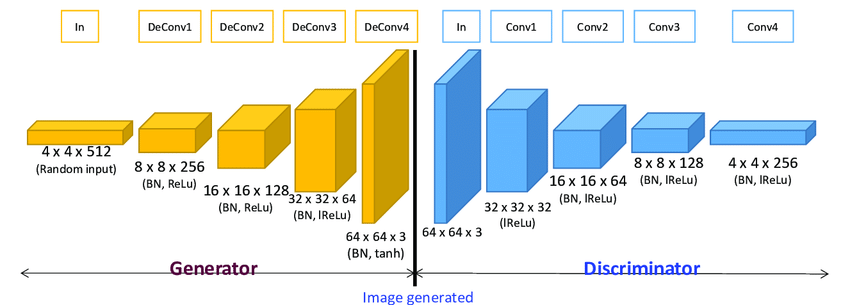


#### Hình 3: Mô hình huấn luyện bằng DCGAN

Generator sử dụng các lớp nơ-ron như lớp deconvolution để tăng kích thước của tensor đầu vào, kết hợp với các lớp kích hoạt phi tuyến như ReLU hoặc tanh để tạo ra các giá trị pixel trong khoảng giá trị hợp lệ (ví dụ: [0, 1] hoặc [-1, 1]). Generator được cập nhật dựa trên phản hồi từ Discriminator và mục tiêu là nâng cao khả năng đánh lừa Discriminator bằng cách tạo ra các ảnh giả mà giống với ảnh thật nhất có thể.

Discriminator thường được thiết kế dưới dạng một mạng CNN (Convolutional Neural Network) để có khả năng nhìn vào các đặc trưng cục bộ của ảnh. Mục tiêu của Discriminator là học cách phân biệt giữa ảnh thật và ảnh giả. Nó được cập nhật dựa trên phản hồi về chất lượng dự đoán của nó và mục tiêu là tối thiểu hóa sai lệch giữa dự đoán và nhãn thực tế.

Hai quá trình này được thực hiện song song và liên tục để có thể nâng cao hiệu suất của bản thân hơn nữa.



#### Hình 4: Generator và Discriminator của DCGAN

Transposed convolution hay deconvolution có thể coi là phép toán ngược của convolution. Nếu như convolution với stride > 1 giúp làm giảm kích thước của ảnh thì transposed convolution với stride > 1 sẽ làm tăng kích thước ảnh.

Chúng tôi đã thử nghiệm một số giá trị của các tham số và đã chọn được các giá trị được chúng tôi cho là tối ưu nhất cho mô hình này.

| **Tham số** | **Giá trị** |
| --- | --- |
| learning\_rate | 0.002 |
| batch\_size | 128 |
| z\_dim | 128 |
| num\_epochs | 25 |
| image\_size | 64 |

##### Bảng 1: Tham số sử dụng trong mô hình

## **ĐÁNH GIÁ MÔ HÌNH**

### **Kết Quả.**

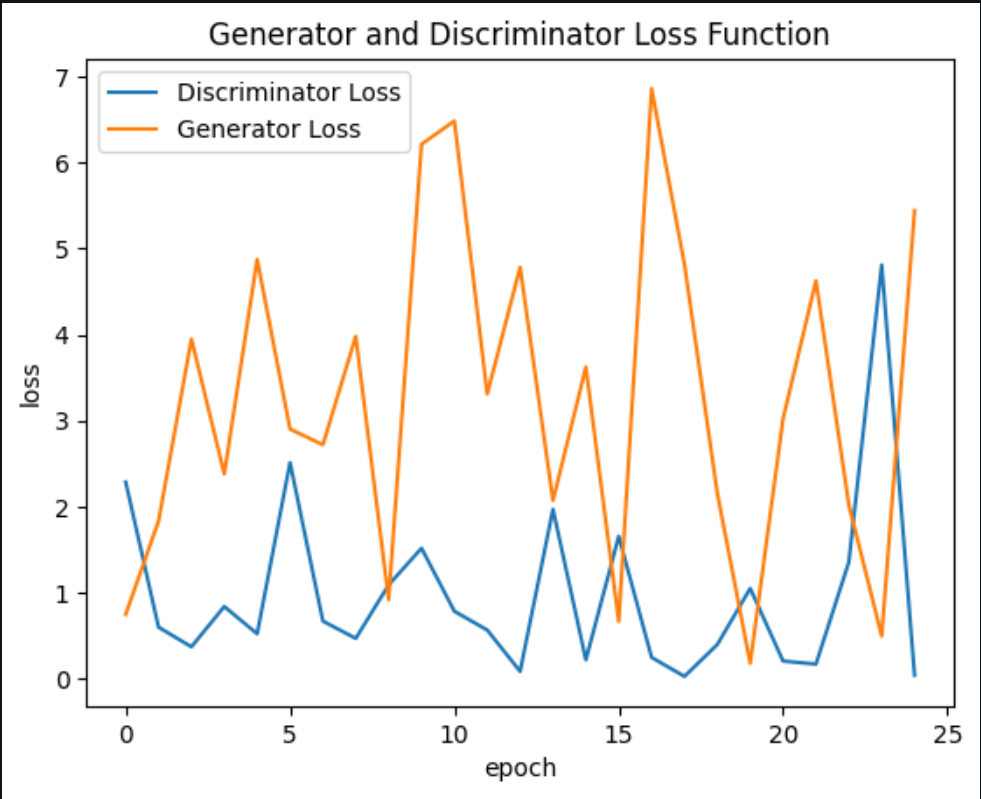
Sau quá trình chạy mô hình thì mô hình đã sinh ra được 64 hình ảnh.



#### Hình 5: 64 hình ảnh mã độc được sinh ra

* Ảnh được sinh ra trong quá trình train tại [đây](https://drive.google.com/drive/folders/1bu23hKaocyBXt_gqHbWIKYQv8HxqPdP1?usp=sharing).
* Video quá trình sinh ra hình ảnh tại [đây](https://drive.google.com/file/d/13NnoVypTavn08Pbhh2lpvAy1KCS7nfaR/view?usp=drive_link).

### **Đồ thị hàm mất mát (loss function).**

Chúng tôi sử dụng hàm mất mát Binary Cross Entropy (BCELoss) là một hàm mất mát được định nghĩa trong thư viện PyTorch.để tính toán mất mát 

#### Hình 6: Đồ thị loss function

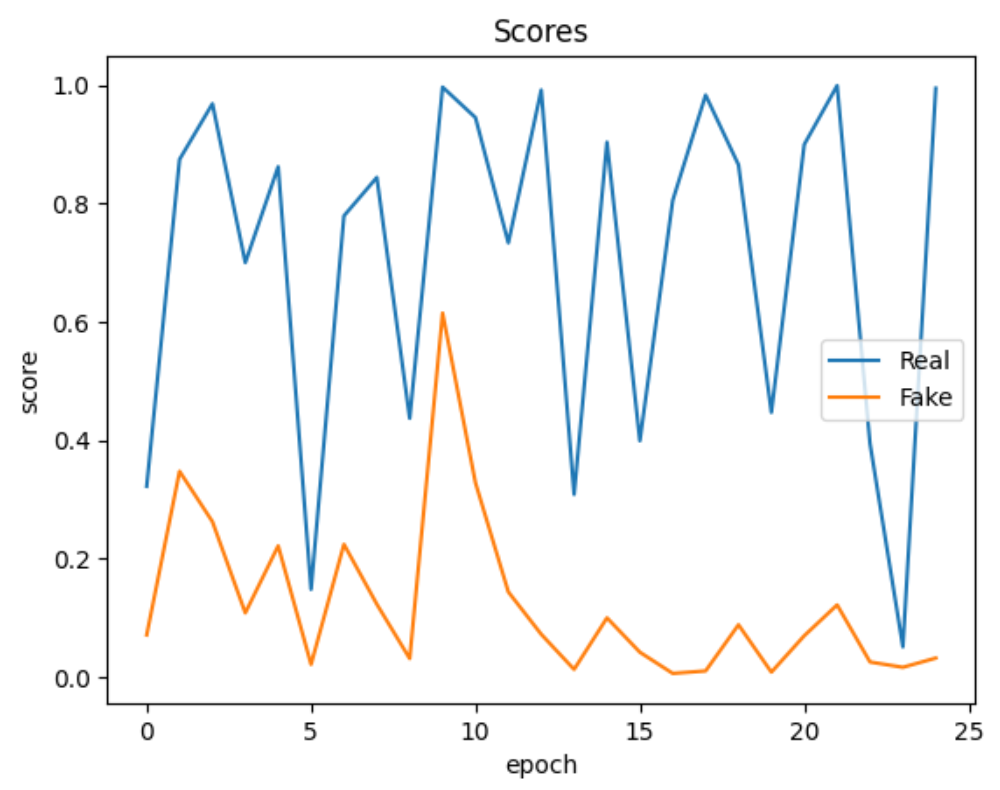
Đường màu cam thể hiện giá trị mất mát của Generator Network

Đường màu xanh thể hiện giá trị mất mát của Discriminator Network

Kết quả:

* Nhìn chung thì mô hình đang bị mất mát không quá cao: loss\_gen ( khoảng 3-4) còn loss\_disc ( khoảng 1).
* Nhìn từ đồ thị chúng tôi nhận thấy Discriminator Network đang ổn định hơn so với Generator Network
* Chúng tôi sẽ tiếp tục tin chỉnh để Generator Network và Discriminator Network đạt hiệu quả cao hơn.

### **Đánh giá.**



#### Hình 7: Đồ thị đánh giá độ chính xác của ảnh sinh ra.

* Đường màu cam thể hiện fake\_score: Điểm số này đo lường độ tương đồng của các ảnh tạo ra với ảnh thực tế.
* Đường màu xanh thể hiện real\_score: Điểm số này đo lường độ tương đồng của các ảnh thực tế với dữ liệu gốc.

Kết quả:

* Ta thấy real\_score thường ở mức cao, điều này cho thấy Generator tạo ra các mẫu giả gần giống với các mẫu thật.
* fake\_score thường giao động ở mức thấp, điều này cho thấy Discriminator có khả năng phân biệt tốt giữa các mẫu thật và giả,

## **KẾT LUẬN**

Nghiên cứu này nhằm mục đích ứng dụng mô hình GAN để phát sinh mã độc PE mới nhằm đánh bại hệ thống ML hiện tại. Từ đó, sử dụng chính mã độc được tạo ra này để huấn luyện lại cho mô hình học máy giúp nó có thể ngăn ngừa được nhiều loại mã độc hơn, nâng cao hiệu suất mô hình.

Chúng tôi đã triển khai thuật toán GAN để sinh ra dữ liệu độc hại và nó có kết quả khá tốt. Trong tương lai chúng tôi sẽ tiếp tục tinh chỉnh để mô hình có thể đạt hiệu suất cao hơn nữa. Bên cạnh đó, chúng tôi sẽ sử dụng các dữ liệu được tạo ra này và kết hợp với một vài mô hình bảo vệ để có thể đánh giá sâu hơn về hiệu suất mô hình cũng như là đào tạo cho hệ thống chống lại một tập tin độc mới.

## **TÀI LIỆU THAM KHẢO**

Nttuan8, 2019: Deep Learning cơ bản, Bài 1: Giới thiệu về GAN

Nttuan8, 2020: Deep Learning cơ bản, Bài 9: StarGAN

Yanminglai, 2018: Malware - GAN

Pham Dinh Khanh, 2020: Bài 43 - Model GAN

Jovian, 2021: Lesson 6: Generative Adversarial Networks and Transfer Learning

Nttuan8, 2020: GAN\_Tutorial

PyTorch: DCGAN TUTORIAL

TensorFlow, 2023: Deep Convolutional Generative Adversarial Network

[Alec Radford](https://arxiv.org/search/cs?searchtype=author&query=Radford,+A); [Luke Metz](https://arxiv.org/search/cs?searchtype=author&query=Metz,+L); [Soumith Chintala](https://arxiv.org/search/cs?searchtype=author&query=Chintala,+S), 2016: [Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks](https://paperswithcode.com/method/dcgan)

## **REFERENCES**

[1] Ani Petrosyan, 2023: Number of malware attacks per year 2015 – 2022.

[2] Zhao, K.; Zhang, D.; Su, X.; Li, W. Fest, 2015: A feature extraction and selection tool for android malware detection.

[3] Weiwei Hu; YingTan, 2017: Generating Adversarial Malware Examples for Black-Box Attacks Based on GAN

[4] Gopinath M., Sibi Chakkaravarthy Sethuraman Ph.D, 2022: A comprehensive survey on deep learning based malware detection techniques

[5] Nttuan8, 2019: Deep Learning cơ bản, Bài 1: Giới thiệu về GAN

Link: https://nttuan8.com/bai-1-gioi-thieu-ve-gan/  
[6] [Alec Radford](https://arxiv.org/search/cs?searchtype=author&query=Radford,+A); [Luke Metz](https://arxiv.org/search/cs?searchtype=author&query=Metz,+L); [Soumith Chintala](https://arxiv.org/search/cs?searchtype=author&query=Chintala,+S), 2016: [Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks](https://paperswithcode.com/method/dcgan)

[7] Aerin Dunmore; Julian Jang-Jaccard; Fariza Sabrina; Jin Kwak, 2023: [Generative Adversarial Networks for Malware Detection](https://arxiv.org/pdf/2302.08558.pdf)

[8] [Huy Nguyen](https://link.springer.com/article/10.1007/s11416-023-00465-2#auth-Huy-Nguyen-Aff1); [Fabio Di Troia](https://link.springer.com/article/10.1007/s11416-023-00465-2#auth-Fabio-Di_Troia-Aff1); [Genya Ishigaki](https://link.springer.com/article/10.1007/s11416-023-00465-2#auth-Genya-Ishigaki-Aff1); [Mark Stamp](https://link.springer.com/article/10.1007/s11416-023-00465-2#auth-Mark-Stamp-Aff1), 2023: [Generative adversarial networks and image-based malware classification](https://link.springer.com/article/10.1007/s11416-023-00465-2)

[9] [Ian J. Goodfellow](https://arxiv.org/search/stat?searchtype=author&query=Goodfellow,+I+J), [Jean Pouget-Abadie](https://arxiv.org/search/stat?searchtype=author&query=Pouget-Abadie,+J), [Mehdi Mirza](https://arxiv.org/search/stat?searchtype=author&query=Mirza,+M), [Bing Xu](https://arxiv.org/search/stat?searchtype=author&query=Xu,+B), [David Warde-Farley](https://arxiv.org/search/stat?searchtype=author&query=Warde-Farley,+D), [Sherjil Ozair](https://arxiv.org/search/stat?searchtype=author&query=Ozair,+S), [Aaron Courville](https://arxiv.org/search/stat?searchtype=author&query=Courville,+A), [Yoshua Bengio](https://arxiv.org/search/stat?searchtype=author&query=Bengio,+Y), 2014: [Generative Adversarial Nets](https://arxiv.org/abs/1406.2661)

[10] Doan Minh Trung; Nguyen Hoang Kim Ngan; Nghi Hoang Khoa; Phan The Duy; Van Hau Pham; Nguyen Tan Cam, 2023: [AAGAN: Android malware generation system based on Generative Adversarial Network](https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4409801)

[11] [Tim Salimans](https://arxiv.org/search/cs?searchtype=author&query=Salimans,+T); [Ian Goodfellow](https://arxiv.org/search/cs?searchtype=author&query=Goodfellow,+I); [Wojciech Zaremba](https://arxiv.org/search/cs?searchtype=author&query=Zaremba,+W); [Vicki Cheung](https://arxiv.org/search/cs?searchtype=author&query=Cheung,+V); [Alec Radford](https://arxiv.org/search/cs?searchtype=author&query=Radford,+A); [Xi Chen](https://arxiv.org/search/cs?searchtype=author&query=Chen,+X), 2016: [Improved Techniques for Training GANs](https://arxiv.org/abs/1606.03498)

[12] [Mehdi Mirza](https://arxiv.org/search/cs?searchtype=author&query=Mirza,+M); [Simon Osindero](https://arxiv.org/search/cs?searchtype=author&query=Osindero,+S), 2014: [Conditional Generative Adversarial Nets](https://arxiv.org/abs/1411.1784)

[13] Gaurav Pendharkar, 2023: Blended Malware Image Dataset

[14] Pham Dinh Khanh, 2020: Bài 43 - Model GAN

[15] Jovian, 2021: Lesson 6: Generative Adversarial Networks and Transfer Learning

[16] PyTorch: DCGAN TUTORIAL

Link: https://pytorch.org/tutorials/beginner/dcgan\_faces\_tutorial.html

[17] TensorFlow, 2023: Deep Convolutional Generative Adversarial Network

Link: https://www.tensorflow.org/tutorials/generative/dcgan