



## 52. Ứng dụng machine learning trong phát hiện mã độc sử dụng đặc trưng tĩnh

An toàn cơ sở dữ liệu (Học Viện Kỹ Thuật Mật Mã)

BAN CƠ YẾU CHÍNH PHỦ  
**HỌC VIỆN KỸ THUẬT MẬT MÃ**

---



**CƠ SỞ AN TOÀN THÔNG TIN**  
**ỨNG DỤNG MACHINE LEARNING**  
**TRONG PHÁT HIỆN MÃ ĐỘC SỬ DỤNG**  
**ĐẶC TRƯNG TÍNH**

Ngành: An toàn thông tin

*Sinh viên thực hiện:*

**Nguyễn Thanh Tùng    AT160158**

**Nguyễn Hoàng Việt    AT160159**

*Người hướng dẫn:*

**Giảng viên: ThS. Nguyễn Ngọc Toàn**

**Khoa An toàn thông tin – Học viện Kỹ thuật mật mã**

Hà Nội, 2022

## This image shows a full page of white paper with horizontal dotted lines. The lines are evenly spaced and run across the entire width of the page, providing a guide for handwriting practice. There are no margins, text, or other markings on the paper.

# MỤC LỤC

ĐÁNH GIÁ CỦA GIẢNG VIÊN .....	1
MỤC LỤC.....	2
DANH MỤC CÁC HÌNH ẢNH.....	4
LỜI MỞ ĐẦU .....	6
CHƯƠNG I – CƠ SỞ LÝ THUYẾT .....	9
<b>1. Tổng quan về mã độc trên Window .....</b>	<b>9</b>
1.1. Khái niệm.....	9
1.2. Phân loại mã độc .....	9
<b>2. Các phương pháp phát hiện mã độc.....</b>	<b>19</b>
2.1. Phương pháp phát hiện dựa trên chữ ký .....	20
2.2. Phương pháp phát hiện dựa trên hành vi .....	24
<b>3. Tổng quan về cấu trúc tệp tin thực thi trên Windows .....</b>	<b>26</b>
3.1. Cấu trúc cơ bản .....	26
3.2. DOS Header .....	27
3.3. PE Header .....	29
3.4. Data Directory.....	31
3.5. Section Table .....	32
<b>4. Tổng quan về học máy .....</b>	<b>34</b>
4.1. Khái niệm.....	34
4.2. Quy trình làm việc của học máy .....	35
4.3. Một số thuật toán phân lớp dữ liệu trong kỹ thuật học máy giám sát ...	36
<b>5. Tổng quan về học sâu.....</b>	<b>42</b>
5.1. Khái niệm.....	42
5.2. Cách hoạt động của học sâu.....	42
5.3. Các phương thức được sử dụng trong học sâu .....	44
5.4. Mô hình học sâu CNN .....	45

CHƯƠNG II - ỨNG DỤNG HỌC MÁY – HỌC SÂU TRONG SẢN PHẨM .....	50
<b>1. Dữ liệu mẫu .....</b>	<b>50</b>
<b>2. Đánh nhãn .....</b>	<b>50</b>
<b>3. Trích xuất đặc trưng .....</b>	<b>51</b>
<b>4. Một số thư viện hỗ trợ AI và ứng dụng ATTT .....</b>	<b>53</b>
4.1. Keras/TensorFlow .....	53
4.2. Theano.....	59
4.3. CNTK.....	62
4.4. Caffè.....	67
4.5. Torch .....	70
<b>5. Mô hình học máy .....</b>	<b>71</b>
<b>6. Mô hình học sâu.....</b>	<b>72</b>
6.1. Tiền xử lý dữ liệu.....	72
6.2. Đưa dữ liệu vào mô hình .....	73
CHƯƠNG III - TRIỂN KHAI THỰC NGHIỆM.....	74
<b>1. Yêu cầu cài đặt.....</b>	<b>74</b>
<b>2. Triển khai cách sử dụng sản phẩm .....</b>	<b>75</b>
CHƯƠNG IV - KẾT LUẬN.....	79
TÀI LIỆU THAM KHẢO.....	80

## DANH MỤC CÁC HÌNH ẢNH

hình 1. cấu trúc tệp thực thi pe.....	26
hình 2. cấu trúc image_dos_header.....	28
hình 3. giá trị của ifanew trên hex dump.....	29
hình 4. cấu trúc image_nt_header .....	29
hình 5. cấu trúc image_file_header .....	30
hình 6. cấu trúc image_optional_header32 .....	31
hình 7. cấu trúc dữ liệu được định nghĩa bởi data directory .....	32
hình 8. cấu trúc image_data_directory .....	32
hình 9. cấu trúc image_section_header .....	33
hình 10. quy trình làm việc của học máy .....	35
hình 11. mô hình cây quyết định.....	37
hình 12. xây dựng một nhóm các cây quyết định .....	39
hình 13. quá trình phỏng đoán của cây quyết định .....	39
hình 14. sơ lược về thuật toán adaboost.....	41
hình 15. ma trận hình ảnh.....	45
hình 16. hình ảnh đen trắng qua quá trình tích chập.....	46
hình 17. hình ảnh màu qua quá trình tích chập .....	47
hình 18. cấu trúc mạng CNN.....	48
hình 19. kết quả virustotal trả về khi query bằng hash md5 .....	51
hình 20. bảng các đặc trưng người dùng có thể chọn thủ công .....	52
hình 21. minh họa về cách sử dụng sequential qua thư viện keras.....	54
hình 22. chia nhỏ dữ liệu đầu vào .....	55
hình 23. ví dụ về cách sử dụng theano .....	60
hình 24. sự khác biệt giữa lớp nn truyền thống và lớp rnn .....	64
hình 25. kiến trúc của caffe .....	67
hình 26. ưu điểm của mô hình caffe.....	69
hình 27. các model học máy sử dụng trong sản phẩm .....	72
hình 28. code tiền xử lý dữ liệu.....	73
hình 29. dữ liệu tiền xử lý được lưu trong deellarningfiletrain.csv .....	73
hình 30. gói sản phẩm khi tải về .....	74
hình 31. gói sản phẩm sau khi cài đặt .....	74
hình 32. giao diện của sản phẩm .....	75
hình 33. cách lấy đặc trưng từ file.....	75
hình 34. file json được tạo.....	76
hình 35. list các đặc trưng trong file json.....	76
hình 36. bảng tùy chọn các đặc trưng .....	76
hình 37. dữ liệu được trích chọn trong file csv .....	77
hình 38. tạo classifier mới .....	77

hình 39. classifier mới đã được tạo thành công .....	77
hình 40. file .h5 sau khi dữ liệu được đưa qua mô hình học sâu .....	78
hình 41. quét trên 1 folder .....	78
hình 42. quét trên 1 file .....	78

## LỜI MỞ ĐẦU

Thế giới gần đây đã chứng kiến sự phát triển lớn trong công nghệ thông tin và truyền thông và kỹ thuật số. Trong những năm vừa qua, theo dòng chảy của cuộc cách mạng 4.0, các thuật ngữ như virus máy tính, mã độc đang dần trở nên phổ biến và trở thành mối nguy hiểm hàng đầu cho các thiết bị điện tử. Đã ba năm trôi qua kể từ ngày ransomware WannaCry làm điều đứng hàng triệu máy tính trên toàn thế giới. Tuy nhiên, theo các chuyên gia bảo mật, cho đến nay, vẫn còn tiềm ẩn rất nhiều rủi ro an ninh từ vụ tấn công mạng này. Trên thực tế, ransomware WannaCry chính là một mã độc. Vậy, mã độc là gì mà nguy hiểm đến vậy? Chúng ảnh hưởng như thế nào đến hệ thống thông tin? Có những loại mã độc phổ biến nào?

Mã độc (Malware) là một trong những mối đe dọa nghiêm trọng đối với bảo mật của hệ thống máy tính, thiết bị thông minh và nhiều ứng dụng. Nó có thể gây nguy hiểm cho dữ liệu quan trọng, nhạy cảm bằng cách đánh cắp, sửa đổi, mã hoá hoặc phá hủy dữ liệu. Theo báo cáo của AV-TEST [1-3], trung bình mỗi ngày có khoảng 350.000 mã độc mới được tạo ra (Hình 1 mô tả tổng số mã độc được phát hiện từ năm 2012 đến năm 2021 (cập nhật ngày 12/03/2021) của tổ chức AV-TEST). Điều này cho thấy mã độc là vấn đề nóng, có tốc độ phát triển nhanh.

Phương pháp dựa trên dấu hiệu nhận dạng được sử dụng trong các phần mềm antivirus khó có khả năng phát hiện chính xác mã độc mới. Mặt khác, dấu hiệu nhận dạng của mã độc phải được lưu trữ trước để thực hiện so sánh mẫu. Do vậy, cần chi phí cho việc lưu trữ chữ ký và thời gian so sánh. Ngoài ra, khả năng phát hiện của phần mềm antivirus không hiệu quả với mã độc loại đa hình, biến hình, hay mã độc được sửa đổi.

Việc phát hiện và phân loại phần mềm độc hại hiệu quả có thể được thực hiện dựa trên việc phân tích tiêu đề PE (Portable Executable) Header của các tệp thực thi (EXE, DLL), kết hợp với các kỹ thuật học máy khác nhau.



Phương pháp học máy đóng một vai trò quan trọng trong việc phát hiện mã độc tự động. Nhiệm vụ thách thức nhất là chọn một bộ đặc trưng phù hợp từ một tập dữ liệu lớn để có thể xây dựng mô hình phân loại trong thời gian ngắn hơn với độ chính xác cao hơn. Mục đích của công việc này trước hết là để thử nghiệm đánh giá, xem xét tổng thể các phương pháp phân loại, phát hiện mã độc và thứ hai là phát triển một hệ thống tự động để phát hiện mã độc dựa trên PE Header các file thực thi (EXE, DLL) với độ chính xác cao và thời gian thực hiện nhanh chóng.

Việc nâng cao kiến thức về mã độc là rất cần thiết. Kiến thức là sức mạnh và chúng ta nên tận dụng tất cả những gì có sẵn. Nhận thấy được tầm quan trọng cũng như sự nguy hiểm của mã độc và cùng với những kiến thức đã được học, chúng em đã quyết định lựa chọn đề tài: “Ứng dụng machine learning trong phát hiện mã độc sử dụng đặc trưng tĩnh”.

Mục tiêu đặt ra khi thực hiện đồ án:

1. Tìm hiểu về mã độc.
2. Tìm hiểu về phương pháp phát hiện mã độc sử dụng đặc trưng tĩnh
3. Ứng dụng của máy học – học sâu trong an toàn thông tin.
4. Triển khai thực nghiệm.

Trên cơ sở đó, bài nghiên cứu cũng tiếp cận việc phân tích mã độc theo cấu trúc tiêu đề của file thực thi PE Header và sử dụng một số kỹ thuật học máy để phát hiện, phân loại các file mã độc với file sạch. Tuy nhiên, có điểm mới là thông tin đặc trưng PE Header được lựa chọn theo khảo sát thống kê giữa các file mã độc và file sạch. Làm thế nào để phát hiện mã độc nhanh chóng và chính xác là vấn đề được các chuyên gia quan tâm nghiên cứu.

Nhóm thực hiện phân tích các thông tin từ PE Header. Sau đó thực hiện khảo sát thống kê và lựa chọn thông tin đặc trưng quan trọng từ PE Header. Tiếp theo là áp dụng các kỹ thuật học máy khác nhau để phân loại và phát hiện mã độc. Sau cùng là tiến hành thực nghiệm, đánh giá độ chính xác và so sánh kết quả nghiên cứu so với các kết quả hiện hành. Mặc dù đã cố gắng hết khả năng của mình

nhưng do trình độ kiến thức và kinh nghiệm còn hạn chế, nên không tránh khỏi có sự sơ sót. Chúng em rất mong nhận được sự nhận xét, đánh giá, đóng góp ý kiến của thầy để bài tập này được hoàn thiện hơn. Chúng em chân thành cảm ơn.

# CHƯƠNG I – CƠ SỞ LÝ THUYẾT

## 1. Tổng quan về mã độc trên Window

### 1.1. Khái niệm

Mã độc là các phần mềm được thiết kế một cách có chủ đích, dùng để gây thiệt hại tới máy tính cá nhân, máy chủ hoặc hệ thống mạng máy tính. Mục đích của mã độc là thực thi các hành vi bất hợp pháp như: truy cập trái phép, đánh cắp thông tin người dùng, lây lan thư rác, thậm chí thực hiện các hành vi tống tiền, tấn công và gây tổn thương cho các hệ thống máy tính,... Nhằm chuộc lợi cá nhân, hoặc các lợi ích về kinh tế, chính trị hay đơn giản chúng có khi được tạo ra chỉ là một trò đùa ác ý nào đó.

Mã độc hại còn được định nghĩa là “Một chương trình (program) được chèn một cách bí mật vào hệ thống với mục đích làm tổn hại đến tính bí mật, tính toàn vẹn hoặc tính sẵn sàng của hệ thống”.

Nhiều người sử dụng máy tính vẫn thường dùng thuật ngữ Virus để chỉ chung cho các loại mã độc hại nhưng thực chất mã độc hại bao gồm nhiều loại khác nhau.

### 1.2. Phân loại mã độc

Có nhiều tiêu chí để phân loại mã độc hại, dưới đây là hai cách phân loại dựa vào hình thức lây nhiễm và của NIST-National Institute of 20 Standard and Technology (Viện tiêu chuẩn – công nghệ quốc gia Hoa kỳ). Sự phân loại chỉ mang tính chất tương đối.

Theo hình trên mã độc hại gồm 2 loại chính: một loại cần vật chủ để tồn tại và lây nhiễm, vật chủ ở đây có thể là các file dữ liệu, các file ứng dụng, hay các file chương trình thực thi,... và một loại là tồn tại độc lập. Độc lập nghĩa là đó là chương trình độc hại mà có thể được lập lịch và chạy trên hệ điều hành.

Không độc lập (needs host program) là 1 đoạn chương trình đặc biệt thuộc 1 chương trình nào đó không thể thực thi độc lập như một chương trình thông

thường hay tiện ích nào đó mà bắt buộc phải có bước kích hoạt chương trình chủ trước đó thì chương trình đó mới chạy.

## **TRAP DOOR**

Trap Door còn được gọi là Back Door. Trong đời sống thường Trap Door mang ý nghĩa “cánh cửa” để vào một tòa nhà. Trap door là một điểm bí mật trong một chương trình, cho phép một ai đó có thể truy cập lại hệ thống mà không phải vượt qua các hàng rào an ninh như thông thường. Trap door được sử dụng bởi những nhà lập trình với mục đích dò lỗi, kiểm tra chương trình. Trong các cuộc tấn công trap door là phần mềm độc hại thường trú và đợi lệnh điều khiển từ các cổng dịch vụ TCP hoặc UDP.

Trap door khi chạy trên máy bị nhiễm, nó sẽ thường trực trong bộ nhớ và mở một cổng cho phép kẻ tấn công truy nhập vào máy nạn nhân thông qua cổng mà nó đã mở và kẻ tấn công có toàn quyền điều khiển máy bị nhiễm.

Trap door nguy hiểm ở chỗ nó hoàn toàn chạy ẩn trong máy. Nhiều con được hẹn trước giờ để kết nối ra ngoài (đến 1 giờ nhất định mới mở 1 port để hacker đột nhập vào) nên rất khó phát hiện ngay cả scan port.

## **LOGIC BOMBS**

Logic bomb là đoạn mã độc được nhúng vào một chương trình hợp pháp mà chúng có thể thực thi khi có một sự kiện nào đó xảy ra. Các đoạn mã thường được chèn vào các ứng dụng hoặc các hệ điều hành để thực hiện việc phá hủy hệ thống hoặc phá hủy các chức năng an toàn của hệ thống. Logic bomb có thể gửi thông báo tới kẻ tấn công khi người dùng truy nhập Internet và sử dụng một chương trình đặc biệt nào đó như bộ xử lý văn bản. Từ đó attacker có thể chuẩn bị cho các cuộc tấn công (chẳng hạn kết hợp với các máy tính khác bị nhiễm để bắt đầu một cuộc tấn công từ chối dịch vụ).

## **TROJAN HORSES**

Trojan Horse là loại mã độc hại được đặt theo sự tích “Ngựa thành Troy”. Trojan horse không có khả năng tự nhân bản tuy nhiên nó lây vào hệ thống với

biểu hiện rất bình thường nhưng thực chất bên trong có ẩn chứa các đoạn mã với mục đích gây hại.

Trojan có thể gây hại theo ba cách sau: Tiếp tục thực thi các chức năng của chương trình mà nó bám vào, bên cạnh đó thực thi các hoạt động gây hại một cách riêng biệt (ví dụ như gửi một trò chơi dụ cho người dùng sử dụng, bên cạnh đó là một chương trình đánh cắp password). Tiếp tục thực thi các chức năng của chương trình mà nó bám vào, nhưng sửa đổi một số chức năng để gây tổn hại (ví dụ như một trojan giả lập một cửa sổ login để lấy password) hoặc che dấu các hành động phá hoại khác (ví dụ như trojan che dấu cho các tiến trình độc hại khác bằng cách tắt các hiển thị của hệ thống). Thực thi luôn một chương trình gây hại bằng cách núp dưới danh một chương trình không có hại (ví dụ như một trojan được giới thiệu như là một trò chơi hoặc một tool trên mạng, người dùng chỉ cần kích hoạt file này là lập tức dữ liệu trên PC sẽ bị xoá hết).

Có 7 loại trojan chính:

- Trojan truy cập từ xa: Được thiết kế để cho kẻ tấn công có khả năng từ xa chiếm quyền điều khiển của máy bị hại. Các trojan này thường dấu vào các trò chơi và các chương trình nhỏ làm cho người dùng mất cảnh giác.
- Trojan gửi dữ liệu: Nó thực hiện việc lấy và gửi dữ liệu nhạy cảm như mật khẩu, thông tin thẻ tín dụng, các tệp nhật ký, địa chỉ email,... cho kẻ tấn công. Trojan này có thể tìm kiếm cụ thể thông tin hoặc cài phần mềm đọc trộm bàn phím và gửi toàn bộ các phím bấm về cho kẻ tấn công.
- Trojan hủy hoại: Thực hiện việc xóa các tệp tin. Loại trojan này giống với virus và thường có thể bị phát hiện bởi các chương trình diệt virus.
- Trojan kiểu proxy: Sử dụng máy tính bị hại làm proxy, qua đó có thể sử dụng máy bị hại để thực hiện các hành vi lừa gạt hay đánh phá các máy tính khác.

- Trojan FTP: Được thiết kế để mở cổng 21 và cho phép tin tặc kết nối vào máy bị hại sử dụng FTP.
- Trojan tắt phần mềm an ninh: Thực hiện việc dừng hoặc xóa bỏ chương trình an ninh như phần mềm chống virus hay tường lửa mà người dùng không nhận ra.
- Trojan DoS: Được sử dụng trong các cuộc tấn công từ chối dịch vụ. Ví dụ các con bot sử dụng trong DDoS cũng có thể coi là một loại trojan.

## VIRUS

Virus là một loại mã độc hại có khả năng tự nhân bản và lây nhiễm chính nó vào các file, chương trình hoặc máy tính. Virus phải luôn bám vào vật chủ (có thể là file dữ liệu hoặc file ứng dụng) để lây lan. Các chương trình diệt virus dựa vào đặc tính này để thực thi việc phòng chống và diệt virus, để quét các file trên các thiết bị lưu trữ, quét các file trước khi lưu xuống ổ cứng.

Vì vậy đôi khi các phần mềm diệt virus tại PC đưa thông báo “phát hiện nhưng không diệt được” khi thấy có các dấu hiệu hoạt động của virus trên PC vì “vật mang virus” lại nằm trên một máy khác nên không thể thực thi việc xóa đoạn mã độc đó. Virus có thể làm bất cứ việc gì mà các chương trình khác có thể làm. Virus chỉ khác ở điểm nó tự đính kèm nó tới một chương trình và thực thi bí mật khi chương trình mang virus được thực thi. Khi virus được thực thi nó có thể làm bất kỳ việc gì trên hệ thống như xóa file, chương trình. Vòng đời virus gồm 4 giai đoạn:

- Dormant (nằm im): Trong giai đoạn này virus không làm gì cho đến khi được kích hoạt bởi một ai đó hay một sự kiện nào đó.
- Propagation (lây lan): Trong giai đoạn này virus thực hiện việc copy chính nó tới các chương trình, vị trí khác trong ổ đĩa.
- Triggering: Trong giai đoạn này virus được kích hoạt để thực thi chức năng của nó.

- Execution: Chức năng của virus được thực thi. Chức năng có thể là vô hại như gửi một thông điệp nào đó tới màn hình, hoặc một chức năng có hại như phá hủy các chương trình, các file hệ thống.

Virus gồm 7 loại chính:

- Memory – resident virus: Cư trú trong bộ nhớ chính như là một phần của chương trình hệ thống. Theo đó virus sẽ gây ảnh hưởng mỗi khi chương trình được thực thi.
- Program file virus: Gây ảnh hưởng đến các file chương trình như exe/com/sys.
- Polymorphic virus(virus đa hình): Loại virus này tự thay đổi hình thức của nó, gây khó khăn cho các chương trình anti-virus. Virus “Tequila” là loại virus đa hình đầu tiên xuất hiện năm 1991.
- Boot Sector virus: Là loại virus đầu tiên trên thế giới được phổ biến rộng rãi và được viết vào năm 1986. Boot virus lợi dụng tiến trình boot của máy tính để thực hiện việc kích hoạt em. Khi máy tính được khởi động, nó luôn tìm đến master boot record được lưu trữ tại địa chỉ head 0, track 0, sector 1 để đọc thông tin. Boot Sector virus lây lan sang đĩa cứng khi khởi động hệ thống từ đĩa mềm bị nhiễm.
- Stealth virus: Đây là loại virus có khả năng tự che dấu không để cho hệ điều hành và phần mềm chống virus biết. Nó nằm trong bộ nhớ để ngăn chặn sử dụng hệ điều hành và che dấu những thay đổi về kích thước các tập tin. Những virus này chỉ bị phát hiện khi chúng còn ở trong bộ nhớ. Có nhiều boot sector virus có khả năng Stealth. Ví dụ virus "The Brain" được tạo ra tại Pakistan bởi Basit và Amjad. Chương trình này nằm trong phần khởi động (boot sector) của một đĩa mềm 360Kb và nó sẽ lây nhiễm tất cả các ổ đĩa mềm. Đây là loại "stealth virus" đầu tiên.
- Macro virus: Là tập lệnh được thực thi bởi một ứng dụng nào đó. Macro virus phổ biến trong các ứng dụng Microsoft Office khi tận

dụng khả năng kiểm soát việc tạo và mở file để thực thi và lây nhiễm. Ví dụ: virus Baza, Laroux và một số virus Staog xuất hiện năm 1996 tấn công các file trong hệ điều hành Windows 95, chương trình bảng tính Excel và cả Linux.

- Email virus: Là những virus được phát tán qua thư điện tử. Ví dụ virus Melissa được đính kèm trong thư điện tử. Nếu người dùng mở file đính kèm Macro được kích hoạt sau đó email virus này tự động gửi chính nó tới tất cả những hòm thư có trong danh sách thư của người đó.

## **WORMS**

Worm là chương trình độc hại có khả năng tự nhân bản và tự lây nhiễm trong hệ thống mà không cần file chủ để mang nó khi nhiễm vào hệ thống. Như vậy Worm không bám vào một file hoặc một vùng nào đó trên đĩa cứng, vì vậy không thể dùng các chương trình quét file để diệt Worm. Mục tiêu của Worm là làm lãng phí băng thông của mạng, phá hoại hệ thống như xóa file, tạo back door, thả keylogger,... Tấn công của Worm có đặc trưng là lan rộng cực kỳ nhanh chóng do không cần tác dụng của con người(như khởi động máy, copy file hay đóng/mở file).

Worm có thể chia làm 2 loại:

- Network Service Worm lan truyền bằng cách lợi dụng các lỗ hổng bảo mật của mạng, của hệ điều hành hoặc của ứng dụng.
- Mass Mailing Worm là một dạng tấn công qua dịch vụ mail, tuy nhiên nó tự đóng gói để tấn công và lây nhiễm chứ không bám vào vật chủ là email. Khi sâu này lây nhiễm vào hệ thống, nó thường cố gắng tìm kiếm số địa chỉ và tự gửi bản thân nó đến các địa chỉ thu nhận được. Việc gửi đồng thời cho toàn bộ các địa chỉ thường gây quá tải cho mạng hoặc cho máy chủ mail.

## **ZOMBIE**



Zombie là chương trình độc hại bí mật liên kết với một máy tính khác ngoài internet để nghe lệnh từ các máy tính đó. Các Zombie thường sử dụng trong các cuộc tấn công từ chối dịch vụ DDoS để tấn công vào một website nào đó. Kiểu thông dụng nhất của Zombie là các agent dùng để tổ chức một cuộc tấn công DDoS. Kẻ tấn công có thể cài Zombie vào một số lượng lớn các máy tính rồi ra lệnh tấn công cùng một lúc.

Ví dụ: Trinoo và Tribe Flood Network là hai Zombie nổi tiếng được sử dụng như các công cụ để thực hiện tấn công DDoS.

### **MALICIOUS MOBILE CODE**

Là một dạng mã phần mềm có thể được gửi từ xa vào để chạy trên một hệ thống mà không cần đến lời gọi thực hiện của người dùng hệ thống đó. Malicious Mobile Code được coi là khác với virus, worm ở đặc tính là nó không nhiễm vào file và không tìm cách tự phát tán. Thay vì khai thác một điểm yếu bảo mật xác định nào đó, kiểu tấn công này thường tác động đến hệ thống bằng cách tận dụng các quyền ưu tiên ngầm định để chạy mã từ xa. Các công cụ lập trình như Java, ActiveX, JavaScript, VBScript là môi trường tốt cho Malicious mobile code.

Một trong những ví dụ nổi tiếng của kiểu tấn công này là Nimda, sử dụng JavaScript. Kiểu tấn công này của Nimda thường được biết đến như một tấn công hỗn hợp (Blended Attack). Cuộc tấn công có thể đi tới bằng một email khi người dùng mở một email độc bằng web-browser. Sau khi nhiễm vào máy này, Nimda sẽ cố gắng sử dụng số địa chỉ email của máy đó để phát tán tới các máy khác.

Mặt khác, từ máy đã bị nhiễm, Nimda cố gắng quét các máy khác trong mạng có thư mục chia sẻ mà không bảo mật, Nimda sẽ dùng dịch vụ NetBIOS như phương tiện để chuyển file nhiễm virus tới các máy đó. Đồng thời Nimda cố gắng dò quét để phát hiện ra các máy tính có cài dịch vụ IIS có điểm yếu bảo mật của Microsoft. Khi tìm thấy, nó sẽ copy bản thân nó vào server. Nếu một web client có điểm yếu bảo mật tương ứng kết nối vào trang web này, client đó

cũng bị nhiễm (lưu ý rằng bị nhiễm mà không cần “mở email bị nhiễm virus”). Quá trình nhiễm virus sẽ lan tràn theo cấp số nhân.

## **TRACKING COOKIE**

Là một dạng lạm dụng cookie để theo dõi một số hành động duyệt web của người sử dụng một cách bất hợp pháp. Cookie là một file dữ liệu chứa thông tin về việc sử dụng một trang web cụ thể nào đó của web-client. Mục tiêu của việc duy trì các cookie trong hệ thống máy tính nhằm căn cứ vào đó để tạo ra giao diện, hành vi của trang web sao cho thích hợp và tương ứng với từng web-client. Tuy nhiên tính năng này lại bị lạm dụng để tạo thành các phần mềm gián điệp (spyware) nhằm thu thập thông tin riêng tư về hành vi duyệt web của cá nhân.

## **SPYWARE**

Là loại phần mềm chuyên thu thập các thông tin từ các máy chủ (thông thường vì mục đích thương mại) qua mạng Internet mà không có sự nhận biết và cho phép của chủ máy. Một cách điển hình, spyware được cài đặt một cách bí mật như là một bộ phận kèm theo của các phần mềm miễn phí (freeware) và phần mềm chia sẻ (shareware) mà người ta có thể tải về từ Internet.

Một khi đã cài đặt, spyware điều phối các hoạt động của máy chủ trên Internet và lặng lẽ chuyển các dữ liệu thông tin đến một máy khác (thường là của những hãng chuyên bán quảng cáo hoặc của các tin tặc). Phần mềm gián điệp cũng thu thập tin tức về địa chỉ thư điện tử và ngay cả mật khẩu cũng như là số thẻ tín dụng. Khác với worm và virus, Spyware không có khả năng tự nhân bản.

## **ADWARE**

Phần mềm quảng cáo, rất hay có ở trong các chương trình cài đặt tải từ trên mạng. Một số phần mềm vô hại, nhưng một số có khả năng hiển thị thông tin lên màn hình, cưỡng chế người sử dụng.

## **ATTACKER TOOL**

Là những bộ công cụ tấn công có thể sử dụng để đẩy các phần mềm độc hại vào trong hệ thống. Các bộ công cụ này có khả năng giúp cho kẻ tấn công có

thể truy nhập bất hợp pháp vào hệ thống hoặc làm cho hệ thống bị lây nhiễm mã độc hại. Khi được tải vào trong hệ thống bằng các đoạn mã độc hại, Attacker tool có thể chính là một phần của đoạn mã độc đó (ví dụ như trong một trojan) hoặc nó sẽ được tải vào hệ thống sau khi nhiễm. Ví dụ như một hệ thống đã bị nhiễm một loại worm, worm này có thể điều khiển hệ thống tự động kết nối đến một website nào đó, tải attacker tool từ site đó và cài đặt Attacker tool vào hệ thống.

Attacker tool thường gặp là backdoor và keylogger - Backdoor là một thuật ngữ chung chỉ các phần mềm độc hại thường trú và đợi lệnh điều khiển từ các cổng dịch vụ TCP hoặc UDP. Một cách đơn giản nhất, phần lớn các backdoor cho phép một kẻ tấn công thực thi một số hành động trên máy bị nhiễm như truyền file, dò mật khẩu, thực hiện mã lệnh... Backdoor cũng có thể được xem xét dưới 2 dạng: Zombie và Remote Administration Tool - Zombie (có thể đôi lúc gọi là bot) là một chương trình được cài đặt lên hệ thống nhằm mục đích tấn công hệ thống khác. Kiểu thông dụng nhất của Zoombie là các Agent dùng để tổ chức một cuộc tấn công DDoS. Kẻ tấn công có thể cài Zombie vào một số lượng lớn các máy tính rồi ra lệnh tấn công cùng một lúc. Trinoo và Tribe Flood Network là hai Zombie nổi tiếng.

Remote Administration Tool là các công cụ có sẵn của hệ thống cho phép thực hiện quyền quản trị từ xa. Tuy nhiên hacker cũng có thể lợi dụng tính năng này để xâm hại hệ thống. Tấn công kiểu này có thể bao gồm hành động theo dõi mọi thứ xuất hiện trên màn hình cho đến tác động vào cấu hình của hệ thống. Ví dụ về công cụ RAT là: Back Orifice, SubSeven...

Keylogger là phần mềm được dùng để bí mật ghi lại các phím đã được nhấn bằng bàn phím rồi gửi tới hacker. Keylogger có thể ghi lại nội dung của email, của văn bản, username, password, thông tin bí mật... Ví dụ một số loại keylogger như: KeySnatch, Spyster...

Rootkits là tập hợp của các file được cài đặt lên hệ thống nhằm biến đổi các chức năng chuẩn của hệ thống thành các chức năng tiềm ẩn các tấn công nguy

hiểm. Ví dụ như trong hệ thống Windows, Rootkit có thể sửa đổi, thay thế file, hoặc thường trú trong bộ nhớ nhằm thay thế, sửa đổi các lời gọi hàm của hệ điều hành. Rootkit thường được dùng để cài đặt các công cụ tấn công như cài backdoor, cài keylogger. Ví dụ về Rootkit là: LRK5, Knark, Adore, Hack Defender.

Web Browser Plug-in là phương thức cài mã độc hại thực thi cùng với trình duyệt web. Khi được cài đặt, kiểu mã độc hại này sẽ theo dõi tất cả các hành vi duyệt web của người dùng (ví dụ như tên website đã truy nhập) sau đó gửi thông tin ra ngoài. Một dạng khác là phần mềm gián điệp có chức năng quay số điện thoại tự động, nó sẽ tự động kích hoạt 32 modem và kết nối đến một số điện thoại ngầm định mặc dù không được phép của chủ nhân.

Email Generator là những chương trình cho phép tạo ra và gửi đi một số lượng lớn các email. Mã độc hại có thể gieo rắc các Email generator vào trong hệ thống. Các chương trình gián điệp, spam, mã độc hại có thể được đính kèm vào các email được sinh ra từ Email generator và gửi tới các địa chỉ có trong sổ địa chỉ của máy bị nhiễm.

Attacker Toolkit là các bộ công cụ có thể được tải xuống và cài vào hệ thống khi hệ thống đã bị khống chế bởi phần mềm độc hại. Các công cụ kiểu như các bộ dò quét cổng (port scanner), bộ phá mật khẩu (password cracker), bộ dò quét gói tin (Packet Sniffer) chính là các Attacker Toolkit thường hay được sử dụng.

## **PHISHING**

Là một hình thức tấn công thường có thể xem là kết hợp với mã độc hại. Phishing là phương thức dụ người dùng kết nối và sử dụng một hệ thống máy tính giả mạo nhằm làm cho người dùng tiết lộ các thông tin bí mật về danh tính (ví dụ như mật khẩu, số tài khoản, thông tin cá nhân...). Kẻ tấn công phishing thường tạo ra trang web hoặc email có hình thức giống hệt như các trang web hoặc email mà nạn nhân thường hay sử dụng như trang của Ngân hàng, của công ty phát hành thẻ tín dụng... Email hoặc trang web giả mạo này sẽ đề nghị nạn nhân thay đổi hoặc cung cấp các thông tin bí mật về tài khoản, về mật khẩu...

Các thông tin này sẽ được sử dụng để trộm tiền trực tiếp trong tài khoản hoặc được sử dụng vào các mục đích bất hợp pháp khác.

## **VIRUS HOAX**

Là các cảnh báo giả về virus. Các cảnh báo giả này thường núp dưới dạng một yêu cầu khẩn cấp để bảo vệ hệ thống. Mục tiêu của cảnh báo virus giả là cố gắng lôi kéo mọi người gửi cảnh báo càng nhiều càng tốt qua email. Bản thân cảnh báo giả là không gây nguy hiểm trực tiếp nhưng những thư gửi để cảnh báo có thể chứa mã độc hại hoặc trong cảnh báo giả có chứa các chỉ dẫn về thiết lập lại hệ điều hành, xóa file làm nguy hại tới hệ thống. Kiểu cảnh báo giả này cũng gây tốn thời gian và quấy rối bộ phận hỗ trợ kỹ thuật khi có quá nhiều người gọi đến và yêu cầu dịch vụ.

### **2. Các phương pháp phát hiện mã độc**

Mục đích của việc phát hiện là đưa ra những cảnh báo sớm để có cơ chế ngăn chặn kịp thời trước khi các mã độc thực hiện các hành vi hay chức năng của chúng. Chính vì vậy vai trò của phát hiện mã độc là rất quan trọng, chúng ta luôn phải tìm kiếm và phát hiện sự tồn tại của mã độc ngay cả khi chúng không thực thi hay làm bất cứ điều gì, thậm chí nếu một mã độc không hoạt động trên một hệ thống việc phát hiện ra nó là rất cần thiết bởi vì điều đó đảm bảo rằng nó sẽ không ảnh hưởng đến hệ thống khác. Lấy hệ thống email là một ví dụ nơi mà các máy nhận có thể chạy trên hệ thống hay chứa những ứng dụng, dịch vụ hoàn toàn khác với máy chủ vì vậy phải đảm bảo rằng mã độc không được thực thi trên bất cứ hệ thống nào.

Các kỹ thuật phát hiện mã độc là một quá trình tìm kiếm và thẩm định xem một chương trình phần mềm có thể đã bị lây nhiễm mã độc hay bên trong có chứa các đoạn mã được xem là mã độc hay không, thêm vào đó các hành vi của chúng cũng được phân tích và xem xét là nhóm các hành vi thông thường hay các hành vi thuộc về mã độc, dựa vào các kết quả đó có thể chứng minh và phát hiện sự tồn tại của mã độc trên các hệ thống.

Các phương pháp phát hiện mã độc hiện nay được chia thành hai loại chính là phát hiện dựa trên chữ ký và phát hiện dựa trên hành vi.

### ***2.1. Phương pháp phát hiện dựa trên chữ ký***

Cơ sở của phương pháp này là phân tích tĩnh, phân tích mã nguồn mà không cần thực thi tệp tin. Phân tích tĩnh tức là đọc mã nguồn của mã độc và cố gắng suy luận các tính chất của nó từ mã nguồn. Phương pháp phát hiện dựa trên chữ ký là kỹ thuật phát hiện mã độc mà không cần khởi chạy hay thực thi bất kỳ đoạn mã nào. Một số kỹ thuật dùng trong phân tích tĩnh bao gồm:

#### **1. Kỹ thuật rò quét(scanner)**

Thông thường mỗi một mã độc được biểu diễn bởi một hay nhiều mẫu, hoặc là các dấu hiệu(signatures), chuỗi tuần tự các byte là những cái được coi là đặc trưng duy nhất của mã độc. Các dấu hiệu đôi khi còn được gọi là các chuỗi(scan strings) và chúng không cần bất kỳ một ràng buộc về chuỗi nào. Một vài phần mềm chống mã độc có thể hỗ trợ sử dụng các ký tự đại diện(wildcards) cho mỗi một byte tùy ý, một phần của một byte. Quá trình phát hiện mã độc bằng cách tìm kiếm thông qua một tập tin với các dấu hiệu của nó thì được gọi là scanning và các mã được tìm thấy được gọi là một scanner. Cụ thể hơn nữa, quá trình phát hiện được thực hiện thông qua một dòng các mã byte chúng có thể là toàn bộ nội dung của một khối khởi động, toàn bộ nội dung của tập tin, hoặc là một phần của tập tin được đọc hoặc ghi, hay cũng có thể là các gói tin mạng.

Với hàng trăm ngàn dấu hiệu để phát hiện việc tìm kiếm chúng tại một thời điểm thì không khả thi chút nào. Một trong thách thức lớn nhất của kỹ thuật này là tìm ra các thuật toán có khả năng tìm được nhiều mẫu một cách hiệu quả và đồng thời đánh giá được chúng.

#### **2. Kỹ thuật Static Heuristics**

Kỹ thuật này được áp dụng để nhận lên khả năng chuyên gia trong các phần mềm chống virus, chuẩn đoán dựa trên kinh nghiệm trong phương pháp phân tích tĩnh có thể tìm thấy các mã độc đã biết hoặc chưa biết bằng cách tìm

kiểm một mẫu mã mà có những đặc điểm chung giống như là một mã độc thay vì scanning các dấu hiệu đặc biệt của mã độc. Đây là một kỹ thuật phân tích tĩnh có nghĩa là các mã sẽ được phân tích mà không thực thi và không có gì đảm bảo về bất kỳ một mã nghi ngờ được tìm thấy sẽ thực thi khi nào. Kỹ thuật này được thực hiện thông qua 2 bước:

**Thu thập dữ liệu:** Dữ liệu thu thập có thể được sử dụng từ bất kỳ một kỹ thuật dựa trên kinh nghiệm nào, có hay không một kỹ thuật kinh nghiệm phân loại chính xác các đầu vào điều đó không thực sự quan trọng bởi vì các kết quả của nhiều kinh nghiệm sẽ được kết hợp và được phân tích sau đó. Một scanner có thể được sử dụng để xác định vị trí một dấu hiệu là các biểu hiện được chỉ ra của những mã đáng nghi được gọi là các booster. Sự hiện diện của các booster này làm tăng khả năng các mã sẽ được phân tích là các mã độc như là: các mã junk, tự thay đổi mã, sử dụng các cuộc gọi hàm API không được cung cấp, điều khiển các vector ngắt, sử dụng các lệnh bất thường đặc biệt là cái không được sinh ra bởi trình biên dịch, chuỗi chứa từ ngữ khiêu dâm, hoặc các tín hiệu rõ ràng như là một từ “virus”. Một việc cũng không kém phần quan trọng là tìm kiếm những thứ xuất hiện trong một mã bình thường đây là những cái mà mã độc không thường làm. Ví dụ virus không tự tạo một hộp thoại pop-up cho người sử dụng. Điều này sẽ được xem xét là một heuristic phủ định hay một stopper.

Các kinh nghiệm heuristic khác có thể được tính toán mà không dựa trên scanning như là:

Sự khác nhau giữa điểm vào và điểm kết thúc của một tập tin có thể được tính toán, các giá trị cực nhỏ khi thực hiện so sánh với những giá trị giống như vậy ở một tập tin không bị lây nhiễm.

Phân tích phổ của mã lệnh có thể được thực hiện, tính toán các tần số của các byte hoặc là các lệnh được sử dụng trong mã. Các mã được mã hóa sẽ có dấu hiệu phổ khác với mã không được mã hóa.

**Phân tích:** phân tích dữ liệu tĩnh heuristic có thể xác định đơn giản như là các trọng số cho mỗi giá trị heuristic và tính tổng các kết quả. Nếu tổng số vượt qua

một giá trị ngưỡng được sử dụng như là một căn cứ phát hiện thì dữ liệu đầu vào có thể đã bị lây nhiễm.

Một số phương pháp phức tạp trong các phân tích dữ liệu có thể sử dụng trí tuệ nhân tạo như là mạng neural, hệ chuyên gia, hay các kỹ thuật khai phá dữ liệu.

Kỹ thuật Static heuristics có thể được xem như là một cách để giảm các yêu cầu tài nguyên của kỹ thuật scanner. Toàn bộ dấu hiệu của mã độc trong một cơ sở dữ liệu mã độc có thể được chất lọc giảm xuống một tập nhỏ hơn, tổng quát hơn. Một kỹ thuật mã độc có thể tìm kiếm những dấu hiệu đặc trưng ngắn này và nạp vào trong bộ dữ liệu chứa toàn bộ các dấu hiệu nếu phù hợp với những gì đã thấy. Điều này làm giảm bớt yêu cầu lưu trữ toàn bộ các dấu hiệu đặc trưng trong bộ nhớ.

### 3. Kỹ thuật kiểm tra sự toàn vẹn(Integrity Checkers)

Ngoại trừ một số virus đồng dạng các mã độc chủ yếu hoạt động bằng cách thay đổi các tập tin. Kỹ thuật kiểm tra tính toàn vẹn nhằm khai thác các hành vi này để tìm ra mã độc bằng cách xem các thay đổi trái phép vào các tập tin. Một kiểm tra toàn vẹn được khởi đầu bằng việc tính và lưu trữ một checksum cho mỗi tập tin cần kiểm tra sẽ được tính lại và so sánh với giá trị checksum gốc của nó. Nếu checksum khác nhau có nghĩa là đã có một sự thay đổi diễn ra. Kỹ thuật này được sử dụng rất phổ biến trong các phần mềm chống mã độc ban đầu các mã độc được tìm thấy sẽ được sử dụng một hàm băm như là MD5, SHA, CRC,... Để tính toán ra giá trị hàm băm duy nhất của nó sau đó các giá trị này được lưu trong cơ sở dữ liệu như là các dữ liệu mẫu nhất định danh mã độc đó. Khi muốn kiểm tra một tập tin có phải là mã độc hay không chúng ta chỉ cần tính toán lại hàm băm và so sánh giá trị hiệu quả với cơ sở dữ liệu đã có. Nếu tồn tại một mã băm như vậy trong cơ sở dữ liệu mã độc thì có thể biết chính xác nó là một mã độc.

Một số kỹ thuật khác như:



- Kiểm tra định dạng tệp: metadata của tệp tin có thể cung cấp các thông tin hữu ích. Ví dụ, các tệp Windows PE có thể cung cấp các thông tin như thời gian thực thi, các chức năng nhập và xuất,...
- Trích xuất chuỗi: liên quan đến việc kiểm tra đầu ra của phần mềm (thông điệp trạng thái hoặc thông điệp báo lỗi) và suy luận về hoạt động của mã độc.
- Lưu vết: trước khi thực hiện phân tích cần tính toán giá trị băm của tệp tin, nhằm xác minh xem tệp tin đã bị chỉnh sửa chưa. Thuật toán băm thường được sử dụng là MD5, SHA1, SHA256. Ngoài ra có thể tìm kiếm các thông tin trong mã nguồn như tên người dùng, tên tệp, chuỗi registry.
- Disassembly: liên quan đến việc đảo ngược mã máy thành hợp ngữ (Assembly language) và từ đó biết được logic và mục đích của phần mềm. Đây là phương thức thường dùng và đáng tin cậy nhất trong phân tích tĩnh.

Ưu điểm của phân tích tĩnh là có thể tìm ra tất cả kịch bản thực thi có thể có của mã độc mà không bị hạn chế về bất kỳ điều kiện gì. Hơn nữa, phân tích tĩnh an toàn hơn phân tích động bởi không cần thực thi mã độc trực tiếp, vì thế sẽ không gây nguy hiểm cho hệ thống. Tuy nhiên phân tích tĩnh lại tốn rất nhiều thời gian, vì thế phân tích tĩnh thường không được sử dụng trong thực tế mà thường dùng để nghiên cứu, ví dụ khi nghiên cứu chữ ký cho các mã độc zero-day.

Phát hiện dựa trên chữ ký là phương pháp tĩnh dựa trên các chữ ký đã được định nghĩa sẵn. Chữ ký có thể là chuỗi băm của tệp tin mã độc (MD5 hoặc SHA1), một chuỗi cố định hoặc metadata của tệp tin. Khi một nhà cung cấp giải pháp anti malware xác định một đối tượng là độc hại, chữ ký của nó sẽ được thêm vào cơ sở dữ liệu. Khi kiểm tra một tệp tin bất kỳ trong hệ thống, phần mềm anti-virus sẽ phân tích tệp tin đó, nếu phát hiện chữ ký khớp với cơ sở dữ liệu sẽ thông báo đây là tệp tin đáng ngờ. Phương pháp phát hiện này chỉ phù hợp với các dạng mã độc phổ biến, có chữ ký cố định được lưu trong cơ sở dữ liệu.

Các loại mã độc hiện đại thường tấn công và tồn tại trong khoảng thời gian ngắn. Ví dụ, Jigsaw bắt đầu xóa dữ liệu trong vòng 24 giờ, HDDcryptor lây nhiễm hơn 2,000 hệ thống tại Cơ quan giao thông thành phố San Francisco trước khi nó được phát hiện. Theo báo cáo năm 2017 của Cisco, 95% mã độc chưa đủ một ngày tuổi khi được phát hiện. Vì thế việc chờ đợi chữ ký của mã độc là một việc rất mạo hiểm.

Ngoài ra, các loại mã độc hiện nay còn có khả năng thay đổi chữ ký nhằm tránh bị phát hiện; chữ ký được tạo ra bởi việc kiểm tra các thành phần bên trong, và mã độc chỉ cần thay đổi các phần này mà không làm ảnh hưởng đến chức năng và hành vi của nó. Một số kỹ thuật biến hình của mã độc gồm hoán vị mã, đổi tên register, mở rộng hoặc thu hẹp mã, chèn các đoạn mã rác, ...

## ***2.2. Phương pháp phát hiện dựa trên hành vi***

Phương pháp phát hiện dựa trên hành vi dựa trên hành vi được phát triển với trí thông minh nhúng để xem xét các sai lệch so với chữ ký phần mềm độc hại và có khả năng xác định xem các tệp đến có thể gây ra bất kỳ mối đe dọa nào cho mạng hoặc hệ thống hay không. Điều này cung cấp một cách hiệu quả để bảo vệ thiết bị người dùng cuối, các phần tử mạng và máy chủ khỏi bất kỳ hoạt động độc hại nào hoặc thậm chí có khả năng gây hại.

Trong phát hiện dựa trên hành vi, phần mềm được lập trình để phân tích và đánh giá từng dòng mã và phân tích tất cả các hành động tiềm ẩn có thể được thực hiện bởi mã đó, chẳng hạn như truy cập vào bất kỳ tệp, quy trình hoặc dịch vụ nội bộ quan trọng hoặc không liên quan nào. Việc thực thi các hướng dẫn cấp hệ điều hành và mã cấp thấp cấp rootkit cũng được bao gồm trong phân tích này. Phần mềm cố gắng phát hiện tất cả các hoạt động độc hại hoặc có khả năng gây hại có thể có bất kỳ tác động xấu nào và thông báo cho những người liên quan biết để thực hiện các hành động cần thiết.

Đánh giá hành vi độc hại khi nó thực thi được gọi là phân tích động. Mối đe dọa tiềm ẩn hoặc mục đích xấu cũng có thể được đánh giá bằng phân tích tĩnh, nhằm tìm kiếm các khả năng nguy hiểm trong mã và cấu trúc của đối tượng.

Phân tích động không giống với phân tích tĩnh ở chỗ, các hành vi của mã độc được giám sát trong khi nó đang thực thi, từ đó có thể tìm hiểu được thuộc tính và mục đích của mã độc. Thông thường mã độc sẽ được thực thi trong môi trường ảo (vd. Sandbox). Trong quá trình phân tích sẽ phát hiện tất cả hành vi của mã độc, như mở tệp tin, tạo mutexes, ... và kiểu phân tích này sẽ nhanh hơn phân tích tĩnh rất nhiều. Tuy nhiên, phân tích động chỉ biết được hành vi của mã độc trong hệ thống ảo dùng để kiểm tra, ví dụ kết quả thu được khi thực thi hai mã độc giống nhau trong môi trường Windows 7 và Windows 8.1 sẽ khác nhau. Từ phương pháp phân tích động, người ta đã định nghĩa phương pháp phát hiện dựa trên hành vi.

Phương pháp phát hiện dựa trên hành vi (hay còn gọi là dựa trên heuristics) sẽ đánh giá một đối tượng dựa trên hành vi của nó. Khi một đối tượng cố gắng thực thi các hành vi bất thường hoặc không được cấp quyền biểu thị đối tượng đó độc hại hoặc đáng ngờ. Có một số hành vi được coi là nguy hiểm như vô hiệu hóa các điều khiển bảo mật, cài đặt rootkits, autostart, sửa tệp tin host, thiết lập các kết nối đáng ngờ,... Mỗi hành vi có thể không nguy hiểm nhưng kết hợp với nhau có thể làm tăng độ đáng ngờ của đối tượng. Có một ngưỡng được xác định sẵn, nếu bất kỳ tệp tin nào vượt qua ngưỡng này sẽ được cảnh báo là mã độc. Phương pháp này được áp dụng để phát hiện các loại mã độc có khả năng thay đổi chữ ký (đa hình) hoặc các loại mã độc mới (zero-day).

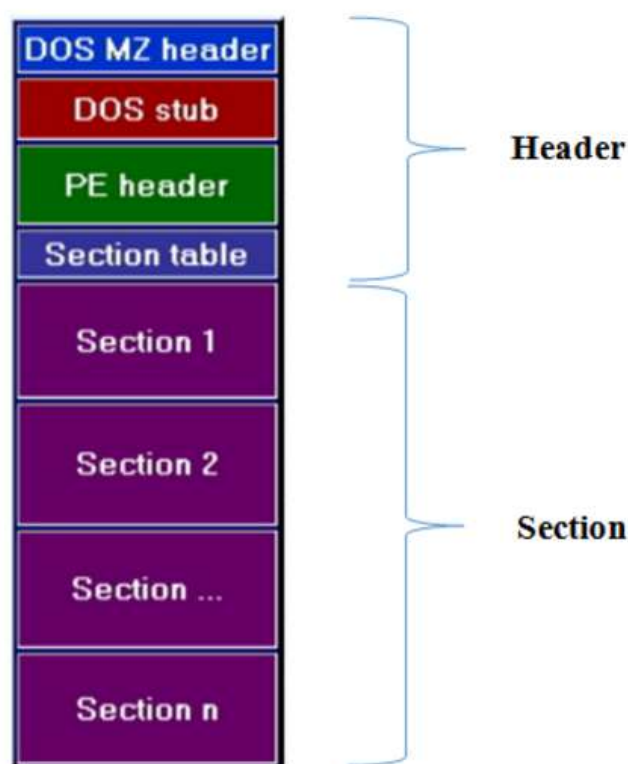
Mức độ chính xác của phát hiện dựa trên heuristics dựa vào quá trình thực thi. Tốt nhất là sử dụng môi trường ảo, ví dụ như sandbox, để chạy tệp tin và giám sát hành vi của nó. Mặc dù phương pháp này tốn nhiều thời gian hơn, nhưng nó an toàn hơn, do tệp tin được kiểm tra trước khi chạy thực tế. Ưu điểm chính của phát hiện dựa trên heuristics là nó không chỉ phát hiện các mã độc đã biết mà còn phát hiện được các cuộc tấn công zero-day và các loại virus đa hình. Tuy nhiên, một số loại mã độc có khả năng phát hiện môi trường ảo, nó sẽ không thực thi các hành vi độc hại trong môi trường sandbox. Hơn nữa, trên thực tế, với lượng mã độc đang ngày một gia tăng, phương pháp này không thực sự hiệu quả trước các loại mã độc mới.

### 3. Tổng quan về cấu trúc tệp tin thực thi trên Windows

Định dạng tệp tin thực thi (PE file) đã được thiết kế để được sử dụng bởi tất cả hệ thống dựa trên Win32. Tất cả các tệp tin có thể thực thi được trên Win32 (ngoại trừ các tệp tin VxDs và các tệp tin DLLs 16bit) đều sử dụng định dạng tệp tin thực thi. Các tệp tin DLLs 32bit, các tệp tin COM, các điều khiển OCX, các chương trình ứng dụng nhỏ trong Control Panel (.CPL) và các ứng dụng .NET tất cả đều là định dạng PE. Thậm chí các chương trình điều khiển ở chế độ Kernel của hệ điều hành Windows NT cũng sử dụng định dạng tệp tin PE.

#### 3.1. Cấu trúc cơ bản

Dưới đây là cấu trúc cơ bản của một tệp tin PE (minh họa hình 1.1).



HÌNH 1. CẤU TRÚC TỆP THỰC THI PE

Ở mức tối thiểu nhất thì một PE file sẽ có 2 Sections: 1 cho đoạn mã (code) và 1 cho phần dữ liệu (data). Một chương trình ứng dụng chạy trên nền tảng Windows NT có 9 sections được xác định trước có tên là .text, .bss, .rdata, .data, .rsrc, .edata, .idata, .pdata, và .debug. Một số chương trình ứng dụng lại

không cần tất cả những sections này, trong khi các chương trình khác có thể được định nghĩa với nhiều sections hơn để phù hợp với sự cần thiết riêng biệt của chúng.

Những sections mà hiện thời đang tồn tại và xuất hiện thông dụng nhất trong một file thực thi là:

- Executable Code Section, có tên là .text (Microsoft) hoặc là CODE (Borland).
- Data Sections, có những tên như .data, .rdata hoặc .bss (Microsoft) hay DATA (Borland).
  - Resources Section, có tên là .rsrc.
  - Export Data Section, có tên là .edata.
  - Import Data Section. có tên là .idata.
  - Debug Information Section, có tên là .debug.

### ***3.2. DOS Header***

Tất cả các file PE bắt đầu bằng DOS Header, vùng này chiếm giữ 64 bytes đầu tiên của file. DOS Header có 19 thành phần (members) mà trong đó thành phần magic và lfnew là đáng chú ý (minh họa hình 1.2).

```

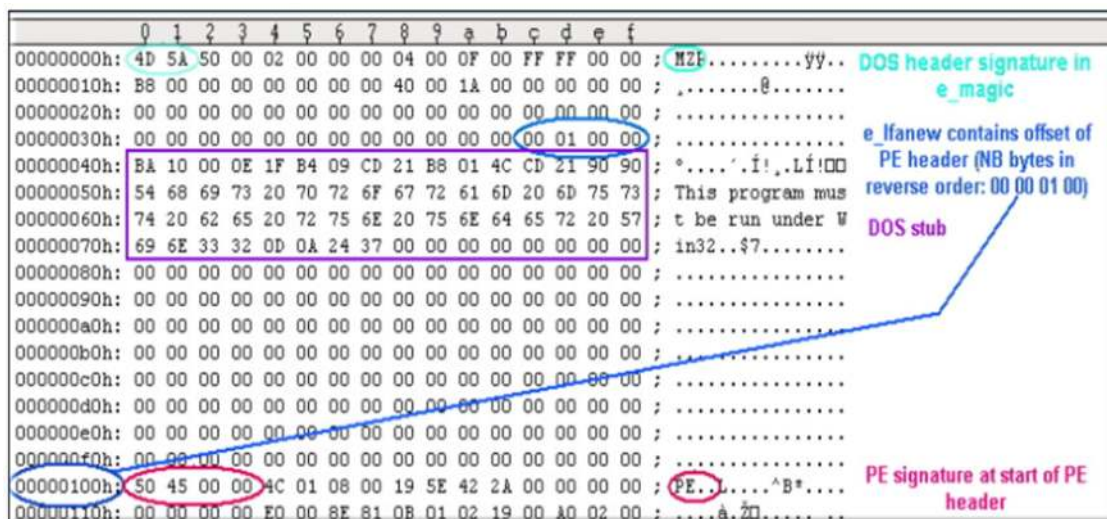
IMAGE_DOS_HEADER STRUCT
    e_magic          WORD        ?
    e_cblp           WORD        ?
    e_cp             WORD        ?
    e_crlc           WORD        ?
    e_cparhdr        WORD        ?
    e_minalloc       WORD        ?
    e_maxalloc       WORD        ?
    e_ss             WORD        ?
    e_sp             WORD        ?
    e_csum           WORD        ?
    e_ip             WORD        ?
    e_cs             WORD        ?
    e_lfarlc         WORD        ?
    e_ovno           WORD        ?
    e_res            WORD        4 dup (?)
    e_oemid          WORD        ?
    e_oeminfo        WORD        ?
    e_res2           WORD        10 dup (?)
    e_lfanew         DWORD       ?
IMAGE_DOS_HEADER ENDS

```

HÌNH 2. CẤU TRÚC IMAGE\_DOS\_HEADER

E\_magic: của DOS Header chứa giá trị 4Dh, 5Ah (Đó chính là kí tự “MZ”, viết tắt của Mark Zbikowsky một trong những người sáng tạo chính của MS-DOS), các giá trị này là dấu hiệu thông báo cho chúng ta biết đây là DOS Header hợp lệ.

E\_lfanew: Nó chứa offset của cấu trúc PE Header. Windows Loader sẽ tìm kiếm offset này vì vậy nó có thể bỏ qua Dos Stub và đi trực tiếp tới cấu trúc PE Header để tìm kiếm thông tin về tệp tin (minh họa hình 1.3).



HÌNH 3. GIÁ TRỊ CỦA IFANEW TRÊN HEX DUMP

### 3.3. PE Header

PE Header là thuật ngữ chung đại diện cho một cấu trúc được đặt tên là `IMAGE_NT_HEADERS`. Cấu trúc này bao gồm những thông tin thiết yếu được sử dụng bởi loader. `IMAGE_NT_HEADERS` có 3 thành phần và được định nghĩa trong file `windows.inc` như hình 1.4 sau:

```

IMAGE_NT_HEADERS STRUCT
    Signature          DWORD                ?
    FileHeader         IMAGE_FILE_HEADER    <>
    OptionalHeader     IMAGE_OPTIONAL_HEADER32 <>
IMAGE_NT_HEADERS ENDS

```

HÌNH 4. CẤU TRÚC `IMAGE_NT_HEADER`

**Signature:** Chứa giá trị `00004550h` (tương ứng với ký tự “PE”). Đây là chữ ký cho biết tệp tin này là tệp tin có định dạng PE File Format.

**FileHeader:** Bao gồm 20 bytes tiếp theo của cấu trúc PE file, nó chứa thông tin về sơ đồ bố trí vật lý và những đặc tính của tệp tin.

**OptionalHeader:** Nó chứa thông tin về sơ đồ Logic bên trong của một tệp tin PE. Kích thước của nó được quy định bởi trường `OptionalHeaderSize` trong cấu trúc `FileHeader`.

#### FileHeader

Cấu trúc này được định nghĩa bởi `IMAGE_FILE_HEADER` (minh họa

hình 1.5):

IMAGE_FILE_HEADER STRUCT		
Machine	WORD	?
NumberOfSections	WORD	?
TimeDateStamp	DWORD	?
PointerToSymbolTable	DWORD	?
NumberOfSymbols	DWORD	?
SizeOfOptionalHeader	WORD	?
Characteristics	WORD	?
IMAGE_FILE_HEADER ENDS		

HÌNH 5. CẤU TRÚC IMAGE\_FILE\_HEADER

**Machine:** Bộ vi xử lý tạo tệp tin.

**NumberOfSections:** Số lượng các sections.

**TimeDateStamp:** Thời gian tệp tin được tạo, đơn vị thời gian là Unix Time.

**SizeOfOptionalHeader:** Kích thước của cấu trúc OptionalHeader.

**Characteristics:** Định nghĩa định dạng (exe, dll,...) của tệp tin.

**OptionalHeader:** Chiếm 224 bytes, trong đó 128 bytes cuối cùng sẽ chứa thông tin về Data Directory. Nó được định nghĩa giống như hình minh họa 1.6 dưới đây.

**AddressOfEntryPoint:** Chứa địa chỉ ảo tương đối của điểm bắt đầu chương trình, nơi mà Windows Loader trả quyền kiểm soát cho chương trình sau khi nạp lên bộ nhớ.

**SectionAlignment:** Bội số kích thước của section trên bộ nhớ. Nghĩa là kích thước của bất cứ section nào cũng phải là bội số của giá trị này. Điều này cần thiết vì Windows quản lý bộ nhớ trên memory theo cơ chế phân trang, ta không thể lưu trữ một dữ liệu giữa một trang bộ nhớ.

**FileAlignment:** Bội số kích thước của section trên đĩa. Lý do là Windows quản lý bộ nhớ trên đĩa theo cơ chế phân sector.



**SizeOfImage:** Toàn bộ kích thước của PE image trong bộ nhớ. Nó là tổng của tất cả các headers và sections được liên kết tới SectionAlignment.

**SizeOfHeaders:** Kích thước của tất cả các headers và section table. Giá trị này như một file offset của section đầu tiên trong tệp tin PE.

IMAGE_OPTIONAL_HEADER32 STRUCT		
Magic	WORD	?
MajorLinkerVersion	BYTE	?
MinorLinkerVersion	BYTE	?
SizeOfCode	DWORD	?
SizeOfInitializedData	DWORD	?
SizeOfUninitializedData	DWORD	?
AddressOfEntryPoint	DWORD	?
BaseOfCode	DWORD	?
BaseOfData	DWORD	?
ImageBase	DWORD	?
SectionAlignment	DWORD	?
FileAlignment	DWORD	?
MajorOperatingSystemVersion	WORD	?
MinorOperatingSystemVersion	WORD	?
MajorImageVersion	WORD	?
MinorImageVersion	WORD	?
MajorSubsystemVersion	WORD	?
MinorSubsystemVersion	WORD	?
Win32VersionValue	DWORD	?
SizeOfImage	DWORD	?
SizeOfHeaders	DWORD	?
Checksum	DWORD	?
Subsystem	WORD	?
DllCharacteristics	WORD	?
SizeOfStackReserve	DWORD	?
SizeOfStackCommit	DWORD	?
SizeOfHeapReserve	DWORD	?
SizeOfHeapCommit	DWORD	?
LoaderFlags	DWORD	?
NumberOfRvaAndSizes	DWORD	?
DataDirectory	IMAGE_DATA_DIRECTORY	
IMAGE_OPTIONAL_HEADER32 ENDS		

HÌNH 6. CẤU TRÚC IMAGE\_OPTIONAL\_HEADER32

### 3.4. Data Directory

Một mảng của 16 cấu trúc IMAGE\_DATA\_DIRECTORY, mỗi một phần có liên quan tới một cấu trúc dữ liệu quan trọng trong PE File. Data Directory là 128 bytes cuối cùng của OptionalHeader và lần lượt là những thành phần cuối cùng của cấu trúc Image\_NT-Headers.

Mỗi mảng tham chiếu tới một mục đã được định nghĩa trước, ví dụ như import table. Cấu trúc của Data Directory được mô tả chi tiết theo cấu trúc dưới đây.

IMAGE_DIRECTORY_ENTRY_EXPORT	equ 0
IMAGE_DIRECTORY_ENTRY_IMPORT	equ 1
IMAGE_DIRECTORY_ENTRY_RESOURCE	equ 2
IMAGE_DIRECTORY_ENTRY_EXCEPTION	equ 3
IMAGE_DIRECTORY_ENTRY_SECURITY	equ 4
IMAGE_DIRECTORY_ENTRY_BASERELOC	equ 5
IMAGE_DIRECTORY_ENTRY_DEBUG	equ 6
IMAGE_DIRECTORY_ENTRY_COPYRIGHT	equ 7
IMAGE_DIRECTORY_ENTRY_GLOBALPTR	equ 8
IMAGE_DIRECTORY_ENTRY_TLS	equ 9
IMAGE_DIRECTORY_ENTRY_LOAD_CONFIG	equ 10
IMAGE_DIRECTORY_ENTRY_BOUND_IMPORT	equ 11
IMAGE_DIRECTORY_ENTRY_IAT	equ 12
IMAGE_DIRECTORY_ENTRY_DELAY_IMPORT	equ 13
IMAGE_DIRECTORY_ENTRY_COM_DESCRIPTOR	equ 14
IMAGE_NUMBEROF_DIRECTORY_ENTRIES	equ 16

HÌNH 7. CẤU TRÚC DỮ LIỆU ĐƯỢC ĐỊNH NGHĨA BỞI DATA DIRECTORY

Những cấu trúc trên có thành phần cụ thể như sau (minh họa hình 1.8):

IMAGE_DATA_DIRECTORY STRUCT		
VirtualAddress	DWORD	?
Isizesize	DWORD	?
IMAGE_DATA_DIRECTORY ENDS		

HÌNH 8. CẤU TRÚC IMAGE\_DATA\_DIRECTORY

**VirtualAddress:** là địa chỉ ảo tương đối (Relative Virtual Address) của cấu trúc dữ liệu.

**Isizesize:** bao gồm kích thước theo bytes của cấu trúc dữ liệu.

Để xác định được vị trí của một directory chỉ định, cần xác định rõ chỉ địa chỉ tương đối từ cấu trúc data directory. Sau đó sử dụng địa chỉ ảo để xác định directory ở trong section nào. Một khi bạn phân tích section nào chứa directory thì Section Header cho section đó sau đó sẽ được sử dụng để tìm ra offset chính xác.

### 3.5. Section Table

Section table là thành phần tiếp theo ngay sau PE Header. Nó là một mảng của những cấu trúc Image\_Section\_Header, mỗi phần tử sẽ chứa thông tin

về một section trong PE File ví dụ như thuộc tính của nó và địa chỉ tương đối ảo (virtual offset).

Số lượng các section được định nghĩa trong thành phần thứ hai của cấu trúc FileHeader (6 bytes từ chỗ bắt đầu của PE Header) trường NumberOfSections. Nếu có 8 sections trong tệp tin PE, thì sẽ có 8 bản sao của cấu trúc này trong bảng Section Table.

Mỗi một cấu trúc là 40 bytes và sẽ không có thêm “padding” giữa chúng (Padding ở đây có nghĩa là sẽ không chèn thêm các bytes có giá trị 00h vào). Cấu trúc của Section Header như sau.

IMAGE_SECTION_HEADER STRUCT		
Name1	BYTE	IMAGE_SIZEOF_SHORT_NAME dup(?)
union Misc		
PhysicalAddress	DWORD	?
VirtualSize	DWORD	?
ends		
VirtualAddress	DWORD	?
SizeOfRawData	DWORD	?
PointerToRawData	DWORD	?
PointerToRelocations	DWORD	?
PointerToLinenumbers	DWORD	?
NumberOfRelocations	WORD	?
NumberOfLinenumbers	WORD	?
Characteristics	DWORD	?
IMAGE_SECTION_HEADER ENDS		
IMAGE_SIZEOF_SHORT_NAME equ 8		

HÌNH 9. CẤU TRÚC IMAGE\_SECTION\_HEADER

Sau đây là ý nghĩa của các trường trong cấu trúc Image\_Section\_Header.

**Name:** Tên này chỉ là một nhãn và thậm chí là có thể để trống. Đây không phải là một chuỗi ASCII vì vậy nó không cần phải kết thúc bằng việc thêm các số 0.

**VirtualSize:** Kích thước của section trên bộ nhớ, tính theo bytes. Nó có thể nhỏ hơn kích thước của section trên đĩa và sẽ là những gì mà Windows Loader định rõ vị trí trong bộ nhớ cho section này.

**VirtualAddress:** Địa chỉ ảo tương đối của section. Trình PE Loader sẽ phân tích và sử dụng giá trị trong trường này khi nó ánh xạ section vào trong bộ nhớ. Vì

vậy nếu giá trị trong trường này là 1000h và tệp tin PE được nạp tại địa chỉ 40000h, thì section sẽ được nạp tại địa chỉ là 401000h.

***SizeOfRawData***: Kích thước của section trên đĩa được làm tròn lên bội số tiếp theo của giá trị này. Đây là giá trị cho thấy cơ chế quản lý dữ liệu trên bộ nhớ của Windows theo cơ chế phân trang.

***PointerToRawData***: Thành phần này thực sự rất hữu dụng bởi vì nó là offset từ vị trí bắt đầu của tệp tin cho tới phần section data. Nếu nó có giá trị là 0, thì section data không được chứa trong tệp tin và sẽ không bị bó buộc vào thời gian nạp. Trình PE Loader sẽ sử dụng giá trị trong trường này để tìm kiếm phần data trong section là ở đâu trong tệp tin.

***Characteristics***: Bao gồm các cờ ví dụ như section này có thể chứa executable code, initialized data, uninitialized data, có thể được ghi hoặc đọc.

Trên đây trình bày các cấu trúc sẽ được trích xuất để sử dụng làm dữ liệu huấn luyện tạo cây quyết định (trừ cấu trúc Section Header). Ngoài ra cấu trúc tệp tin thực thi trên hệ điều hành Windows còn có các cấu trúc như Import Directory, Export Directory, ...

Nhưng vì không sử dụng các dữ liệu này để huấn luyện nên sẽ không nhắc tới trong phạm vi luận văn này. Bên cạnh đó ngoài các trường quan trọng được trình bày ở trên đây thì còn rất nhiều trường khác được trích xuất sẽ được liệt kê rõ ràng trong phần triển khai.

## **4. Tổng quan về học máy**

### **4.1. Khái niệm**

Học máy(Machine learning) là một nhánh của trí tuệ nhân tạo (AI), nó là một lĩnh vực nghiên cứu cho phép máy tính có khả năng cải thiện chính bản thân chúng dựa trên dữ liệu mẫu (training data) hoặc dựa vào kinh nghiệm (những gì đã được học). Machine learning có thể tự dự đoán hoặc đưa ra quyết định mà không cần được lập trình cụ thể.

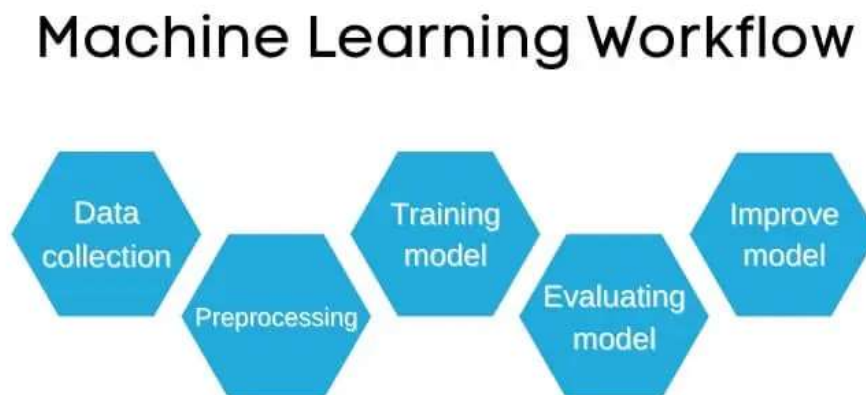
Bài toán machine learning thường được chia làm hai loại là dự đoán (prediction) và phân loại (classification). Các bài toán dự đoán như dự đoán giá nhà, giá xe... Các bài toán phân loại như nhận diện chữ viết tay, nhận diện đồ vật...

#### 4.2. Quy trình làm việc của học máy

Tương tự như cách bộ não con người thu thập kiến thức và hiểu biết, học máy dựa vào đầu vào, chẳng hạn như dữ liệu đào tạo hoặc biểu đồ kiến thức, để hiểu các thực thể, lĩnh vực và mối liên hệ giữa chúng. Với các thực thể được xác định, học sâu có thể bắt đầu.

Quá trình học máy bắt đầu với các quan sát hoặc dữ liệu, chẳng hạn như ví dụ, kinh nghiệm trực tiếp hoặc hướng dẫn. Nó tìm kiếm các mẫu trong dữ liệu để sau này có thể đưa ra các suy luận dựa trên các ví dụ được cung cấp. Mục đích chính của học máy là cho phép máy tính tự học mà không cần sự can thiệp hoặc trợ giúp của con người và điều chỉnh các hành động cho phù hợp.

Cụ thể từng bước trong machine learning workflow như sau như hình 1.10:



HÌNH 10. QUY TRÌNH LÀM VIỆC CỦA HỌC MÁY

1. Thu thập dữ liệu (Data collection): để máy tính có thể học được bạn cần có một bộ dữ liệu (dataset), bạn có thể tự thu thập chúng hoặc lấy các bộ dữ liệu đã được công bố trước đó. Lưu ý là bạn phải thu thập từ nguồn chính thống, có như vậy dữ liệu mới chính xác và máy có thể học một cách đúng đắn và đạt hiệu quả cao hơn.

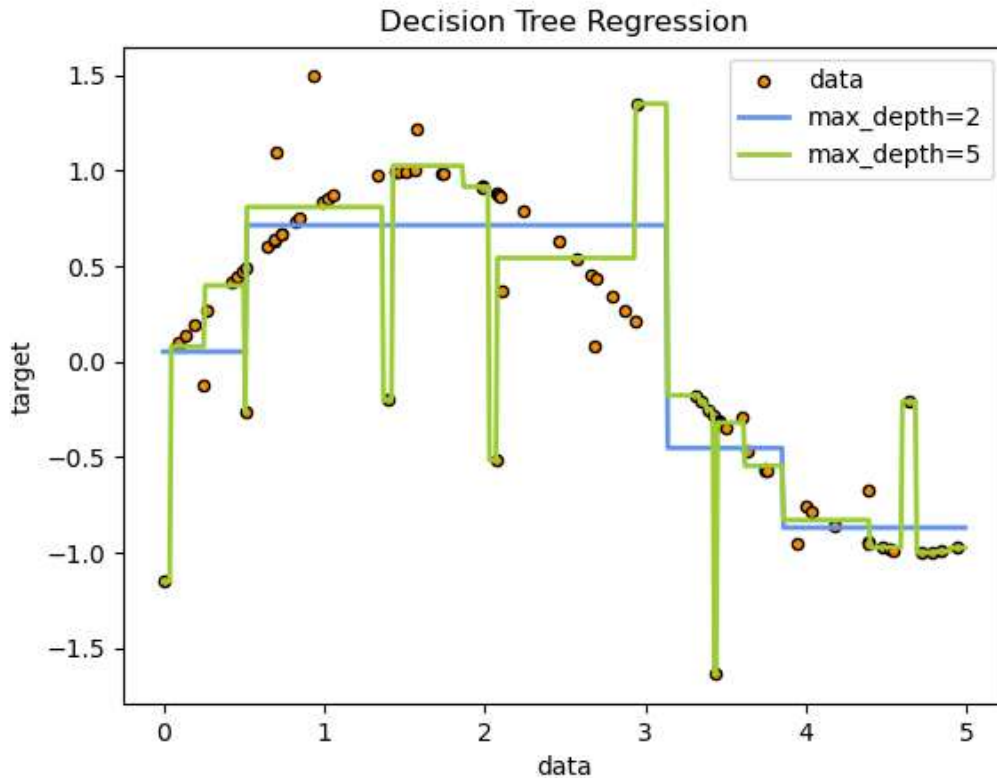
2. Tiền xử lý (Preprocessing): bước này dùng để chuẩn hóa dữ liệu, loại bỏ các thuộc tính không cần thiết, gán nhãn dữ liệu, mã hóa một số đặc trưng, trích xuất đặc trưng, rút gọn dữ liệu nhưng vẫn đảm bảo kết quả... Bước này tốn thời gian nhất tỉ lệ thuận với số lượng dữ liệu bạn có. Bước 1 và 2 thường chiếm hơn 70% tổng thời gian thực hiện.
3. Huấn luyện mô hình (Training model): bước này là bước bạn huấn luyện cho mô hình hay chính là cho nó học trên dữ liệu bạn đã thu thập và xử lý ở hai bước đầu.
4. Đánh giá mô hình (Evaluating model): sau khi đã huấn luyện mô hình xong, chúng ta cần dùng các độ đo để đánh giá mô hình, tùy vào từng độ đo khác nhau mà mô hình cũng được đánh giá tốt hay không khác nhau. Độ chính xác của mô hình đạt trên 80% được cho là tốt.
5. Cải thiện (Improve): sau khi đã đánh giá mô hình, các mô hình đạt độ chính xác không tốt thì cần được đào tạo lại, chúng ta sẽ lặp lại từ bước 3, cho đến khi đạt độ chính xác như kỳ vọng. Tổng thời gian của 3 bước cuối rơi vào khoảng 30% tổng thời gian thực hiện.

#### ***4.3. Một số thuật toán phân lớp dữ liệu trong kỹ thuật học máy giám sát***

##### ***4.3.1. Cây quyết định***

Cây quyết định (Decision Trees) là một phương pháp học tập có giám sát phi tham số được sử dụng để phân loại và hồi quy. Mục đích là tạo ra một mô hình dự đoán giá trị của một biến mục tiêu bằng cách tìm hiểu các quy tắc quyết định đơn giản được suy ra từ các tính năng dữ liệu. Một cây có thể được coi là một phép gần đúng không đổi từng mảnh.

Ví dụ: trong ví dụ dưới đây, cây quyết định học từ dữ liệu để tính gần đúng đường cong sin với một tập hợp các quy tắc quyết định if-then-else. Cây càng sâu, các quy tắc quyết định càng phức tạp và mô hình càng phù hợp.



HÌNH 11. MÔ HÌNH CÂY QUYẾT ĐỊNH

### Ưu điểm:

- Đơn giản để hiểu và dễ giải thích. Các nhánh cây có thể được hình dung.
- Yêu cầu chuẩn bị ít dữ liệu. Các kỹ thuật khác thường yêu cầu chuẩn hóa dữ liệu, các biến giả cần được tạo và loại bỏ các giá trị trống. Tuy nhiên, lưu ý rằng mô-đun này không hỗ trợ các giá trị bị thiếu.
- Chi phí sử dụng cây (tức là dữ liệu dự đoán) được tính theo lôgarit trong số điểm dữ liệu được sử dụng để đào tạo cây.
- Có thể xử lý cả dữ liệu số và dữ liệu phân loại. Tuy nhiên hiện tại việc triển khai scikit-learning không hỗ trợ các biến phân loại. Các kỹ thuật khác thường chuyên về phân tích tập dữ liệu chỉ có một loại biến. Xem các thuật toán để biết thêm thông tin.
- Có khả năng xử lý các vấn đề đa đầu ra.
- Sử dụng mô hình hộp màu trắng. Nếu một tình huống nhất định có thể quan sát được trong một mô hình, thì lời giải thích cho điều kiện đó dễ dàng được giải thích bằng logic boolean. Ngược lại, trong mô hình hộp

đen (ví dụ: trong mạng nơ-ron nhân tạo), kết quả có thể khó giải thích hơn.

- Có thể xác nhận một mô hình bằng cách sử dụng các thử nghiệm thống kê. Điều đó làm cho nó có thể tính đến độ tin cậy của mô hình.
- Hoạt động tốt ngay cả khi các giả định của nó phần nào bị vi phạm bởi mô hình thực mà từ đó dữ liệu được tạo ra.

#### **Ngược điểm:**

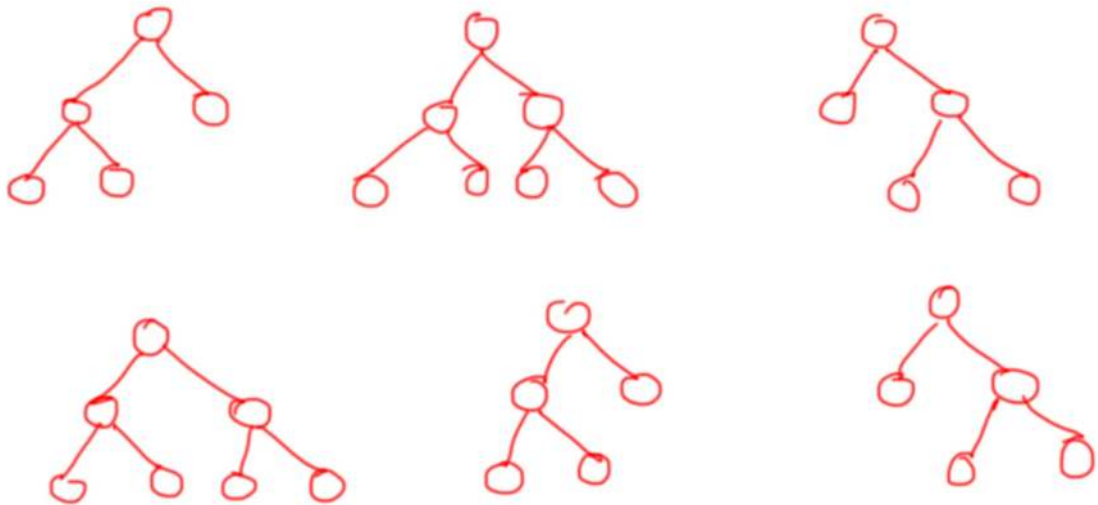
- Người học cây quyết định có thể tạo cây quá phức tạp không tổng quát hóa dữ liệu tốt. Điều này được gọi là tràn dữ liệu. Các cơ chế như cắt tỉa, thiết lập số lượng mẫu tối thiểu cần thiết tại một nút lá hoặc thiết lập độ sâu tối đa của cây là cần thiết để tránh vấn đề này.
- Cây quyết định có thể không ổn định vì các biến thể nhỏ trong dữ liệu có thể dẫn đến việc tạo ra một cây hoàn toàn khác. Vấn đề này được giảm thiểu bằng cách sử dụng cây quyết định trong một tập hợp.
- Các dự đoán của cây quyết định không trơn tru cũng không liên tục, mà là các phép gần đúng liên tục từng phần như được thấy trong hình trên. Do đó, họ không giới ngoại suy.
- Có những khái niệm khó học vì cây quyết định không thể hiện chúng một cách dễ dàng, chẳng hạn như các vấn đề XOR, chẵn lẻ hoặc bộ ghép kênh.
- Những người học cây quyết định tạo cây thiên vị nếu một số lớp chiếm ưu thế. Do đó, nên cân bằng tập dữ liệu trước khi phù hợp với cây quyết định.

#### **4.3.2. Rừng ngẫu nhiên**

Rừng ngẫu nhiên (Random Forest) là một thuật toán học máy được sử dụng bởi Leo Breiman và Adele Cutler, kết hợp đầu ra của nhiều cây quyết định để đạt được một kết quả duy nhất. Tính dễ sử dụng và tính linh hoạt của nó đã thúc đẩy việc áp dụng nó, vì nó xử lý được cả vấn đề phân loại và hồi quy.

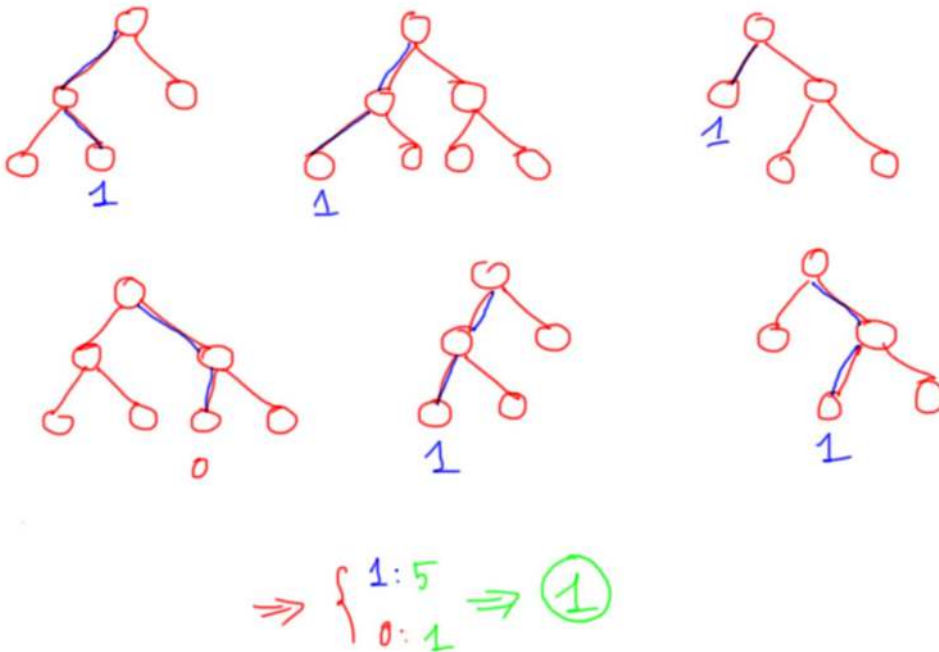
Ở bước huấn luyện thì em sẽ xây dựng nhiều cây quyết định, các cây quyết định có thể khác nhau (phần sau em sẽ nói mỗi cây được xây dựng như thế nào).





HÌNH 12. XÂY DỰNG MỘT NHÓM CÁC CÂY QUYẾT ĐỊNH

Sau đó ở bước dự đoán, với một dữ liệu mới, thì ở mỗi cây quyết định em sẽ đi từ trên xuống theo các node điều kiện để được các dự đoán, sau đó kết quả cuối cùng được tổng hợp từ kết quả của các cây quyết định.



HÌNH 13. QUÁ TRÌNH PHÒNG ĐOÁN CỦA CÂY QUYẾT ĐỊNH

Ví dụ như trên, thuật toán rừng ngẫu nhiên có 6 cây quyết định, 5 cây dự đoán 1 và 1 cây dự đoán 0, do đó em sẽ vote là cho ra dự đoán cuối cùng là 1.

Ưu điểm so với cây quyết định là Trong thuật toán cây quyết định, khi xây dựng cây quyết định nếu độ sâu tùy ý thì cây sẽ phân loại đúng hết các dữ liệu trong tập huấn luyện dẫn đến mô hình có thể dự đoán tệ trên tập validation/test, khi đó mô hình bị overfitting.

Thuật toán Random Forest gồm nhiều cây quyết định, mỗi cây quyết định đều có những yếu tố ngẫu nhiên:

- Lấy ngẫu nhiên dữ liệu để xây dựng cây quyết định.
- Lấy ngẫu nhiên các thuộc tính để xây dựng cây quyết định.

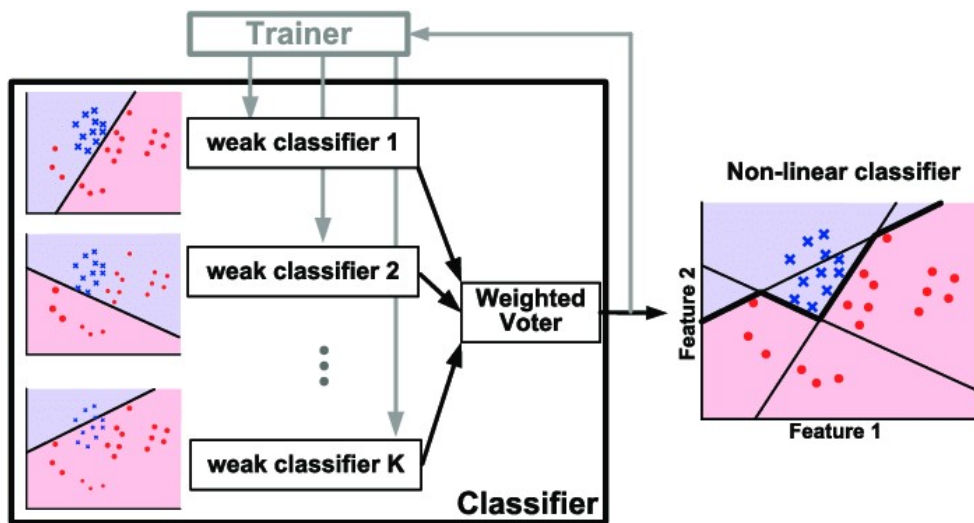
Do mỗi cây quyết định trong thuật toán Random Forest không dùng tất cả dữ liệu training, cũng như không dùng tất cả các thuộc tính của dữ liệu để xây dựng cây nên mỗi cây có thể sẽ dự đoán không tốt, khi đó mỗi mô hình cây quyết định không bị overfitting mà có thể bị underfitting, hay nói cách khác là mô hình có high bias.

Tuy nhiên, kết quả cuối cùng của thuật toán Random Forest lại tổng hợp từ nhiều cây quyết định, thế nên thông tin từ các cây sẽ bổ sung thông tin cho nhau, dẫn đến mô hình có low bias và low variance, hay mô hình có kết quả dự đoán tốt.

### ***4.3.3. AdaBoost—Gradient Boosting***

#### **Thuật toán AdaBoost**

AdaBoost, tên đầy đủ là Adaptive Boosting, là thuật toán thuộc nhánh Boosting trong Ensemble learning. Với ý tưởng đơn giản là sử dụng các cây quyết định (1 gốc, 2 lá) để đánh trọng số cho các điểm dữ liệu, từ đó tối thiểu hóa trọng số các điểm bị phân loại sai (trọng số lớn), để tăng hiệu suất của mô hình.



HÌNH 14. SƠ LƯỢC VỀ THUẬT TOÁN ADABOOST

AdaBoost (Adaptive Boost) là một thuật toán học mạnh, giúp đẩy nhanh việc tạo ra một bộ phân loại mạnh (strong classifier) bằng cách chọn các đặc trưng tốt trong một họ các bộ phân loại yếu (weak classifier - bộ phân loại yếu) và kết hợp chúng lại tuyến tính bằng cách sử dụng các trọng số. Điều này thật sự cải thiện dần độ chính xác nhờ áp dụng hiệu quả một chuỗi các bộ phân loại yếu.

AdaBoost có thể được áp dụng mà không cần dựa vào việc đánh trọng số lại các điểm dữ liệu, thay vào đó, chúng ta có thể re-sample để lấy dữ liệu train cho các model tiếp theo dựa vào xác suất được xác định bởi các trọng số.

### Thuật toán Gradient Boosting

Gradient Boosting là một dạng tổng quát hóa của AdaBoost. Nó đưa ra một mô hình dự đoán dưới dạng một tập hợp các mô hình dự đoán yếu, thường là cây quyết định. Khi cây quyết định là người học yếu, thuật toán kết quả được gọi là cây tăng độ dốc; nó thường hoạt động tốt hơn rừng ngẫu nhiên.

Nếu bạn để ý thì phương pháp cập nhật lại trọng số của điểm dữ liệu của AdaBoost cũng là 1 trong các case của Gradient Boosting. Do đó, Gradient Boosting bao quát được nhiều trường hợp hơn.

## 5. Tổng quan về học sâu

### 5.1. Khái niệm

Học sâu (Deep Learning) là một tập hợp con của học máy (Machine Learning) và trí tuệ nhân tạo AI, bắt chước cách con người thu thập kiến thức. Bao gồm quá trình trích xuất thông kê và dự đoán. Nói một cách đơn giản, học sâu là cách tự động hóa phân tích các dự đoán. Trong khi các thuật toán học máy là tuần tự tuyến tính, thì các thuật toán học sâu được xếp chồng theo thứ bậc và độ phức tạp, trừu tượng.

Để dễ hiểu hơn, một máy tính có 1 tập khoảng 100 các file tài liệu, giả sử trong máy đó chỉ có 2 loại file là file tài liệu và file không phải tài liệu. Khi một người mới sử dụng máy tính đó, người đó mới biết rằng trong máy chỉ có 2 loại file là file tài liệu và file không phải tài liệu bằng cách mở và sử dụng trực tiếp file đó, máy tính sẽ báo lên cho người dùng bằng cách file tài liệu thì mở trình soạn thảo, còn không phải thì mở trình loader.

Sau một thời gian sử dụng liên tục như vậy, người dùng đó sẽ rút ra các đặc điểm mà file tài liệu sẽ có, như đuôi file, icon của file, kích thước của file, từ đó người dùng đã vô tình trải qua một quá trình học sâu dựa về phát hiện file có phải file tài liệu không bằng các tầng kiểm thử.

### 5.2. Cách hoạt động của học sâu

Các giai đoạn mà deep learning chạy trong máy tính khá giống với ví dụ về phân biệt file tài liệu bên trên, Mỗi thuật toán trong hệ thống phân cấp áp dụng một phép biến đổi phi tuyến cho đầu vào của nó và sử dụng những gì nó học được để tạo ra một mô hình thống kê làm đầu ra. Lặp lại tiếp tục cho đến khi kết quả đạt được mức độ chính xác có thể chấp nhận được. Số lượng các lớp xử lý mà dữ liệu phải vượt qua là điều đã truyền cảm hứng cho việc gán nhãn.

Trong quá trình Machine Learning truyền thống, quá trình học máy phải có giám sát và người lập trình phải cực kỳ cụ thể khi nói cho máy tính biết loại thứ mà nó cần tìm để quyết định xem file đó có phải là file tài liệu không, ví dụ như tên các trường đặc trưng, dữ liệu đầu vào,...

Đây là một quá trình tốn nhiều công sức được gọi là trích xuất đặc trưng và tỷ lệ thành công của máy tính phụ thuộc hoàn toàn vào khả năng của lập trình viên để xác định chính xác một tập hợp đặc trưng cho file. Ưu điểm của học sâu là chương trình xây dựng tính năng do chính nó thiết lập mà không cần giám sát. Học không giám sát không chỉ nhanh hơn mà còn chính xác hơn.

Ban đầu, chương trình máy tính có thể được cung cấp dữ liệu huấn luyện, đó là một tập các file tài liệu đã được gán nhãn là tài liệu hoặc không phải tài liệu. Chương trình sử dụng thông tin nhận được từ dữ liệu huấn luyện để tạo bộ tính năng và xây dựng mô hình dự đoán.

Trong trường hợp này, mô hình mà máy tính tạo ra lần đầu tiên có thể dự đoán rằng bất kỳ thứ gì có đuôi “.txt” đều là tài liệu. Đương nhiên chương trình chưa thể nhận thức được nhãn có đuôi “.txt” hay không, nó chỉ đơn giản là tìm kiếm các đuôi “.txt” trong file. Do đó với mỗi lần lặp lại, mô hình dự đoán sẽ trở nên phức tạp hơn và chính xác hơn.

Không giống như bộ óc con người, có thể mất vài ngày để hiểu khái niệm file tài liệu, một chương trình máy tính sử dụng thuật toán học sâu có thể được hiển thị một tập hợp đào tạo và sắp xếp thông qua hàng triệu file, xác định chính xác file có phải tài liệu hay không chỉ sau một thời gian rất ngắn.

Để đạt được mức độ chính xác có thể chấp nhận được, các chương trình học sâu yêu cầu quyền truy cập vào lượng dữ liệu đào tạo và sức mạnh xử lý khổng lồ, cả hai đều không dễ dàng có sẵn cho các lập trình viên cho đến thời đại của dữ liệu lớn và điện toán đám mây.

Bởi vì lập trình học sâu có thể tạo ra các mô hình thống kê phức tạp trực tiếp từ đầu ra lặp đi lặp lại của chính nó, nên nó có thể tạo ra các mô hình dự đoán chính xác từ số lượng lớn dữ liệu không có nhãn, không có cấu trúc. Điều này rất quan trọng khi internet vạn vật (IoT) tiếp tục trở nên phổ biến hơn vì hầu hết dữ liệu mà con người và máy móc tạo ra là phi cấu trúc và không được dán nhãn.

### 5.3. Các phương thức được sử dụng trong học sâu

Nhiều phương pháp khác nhau có thể được sử dụng để tạo ra các mô hình học sâu mạnh mẽ. Những kỹ thuật này bao gồm rate decay (giảm tỷ lệ học tập), (transfer learning) học chuyển tiếp, (training from scratch) đào tạo từ đầu và (dropout) bỏ học.

**Rate Decay:** Tốc độ học tập là một siêu tham số - một yếu tố xác định hệ thống hoặc thiết lập các điều kiện cho hoạt động của nó trước quá trình học tập - kiểm soát mức độ thay đổi mà mô hình trải qua để đáp ứng với lỗi ước tính mỗi khi trọng số của mô hình được thay đổi. Tỷ lệ học tập quá cao có thể dẫn đến quá trình đào tạo không ổn định hoặc việc học một bộ trọng lượng dưới mức tối ưu. Tỷ lệ học tập quá nhỏ có thể tạo ra một quá trình đào tạo kéo dài và tiềm ẩn nguy cơ gặp khó khăn.

Phương pháp phân rã tốc độ học tập - còn được gọi là quá trình ủ tỷ lệ học tập hoặc tỷ lệ học tập thích ứng - là quá trình điều chỉnh tốc độ học tập để tăng hiệu suất và giảm thời gian đào tạo. Các cách thích ứng dễ nhất và phổ biến nhất của tốc độ học tập trong quá trình đào tạo bao gồm các kỹ thuật để giảm tốc độ học tập theo thời gian.

**Transfer learning:** Quá trình này liên quan đến việc hoàn thiện một mô hình đã được đào tạo trước đó; nó yêu cầu một giao diện với nội bộ của một mạng đã có từ trước. Đầu tiên, người dùng cung cấp dữ liệu mới cho mạng hiện có chứa các phân loại chưa biết trước đó. Sau khi thực hiện các điều chỉnh đối với mạng, các tác vụ mới có thể được thực hiện với khả năng phân loại cụ thể hơn. Phương pháp này có ưu điểm là yêu cầu ít dữ liệu hơn nhiều so với các phương pháp khác, do đó giảm thời gian tính toán xuống còn phút hoặc giờ.

**Training from scratch:** Phương pháp này yêu cầu nhà phát triển thu thập một tập dữ liệu có nhãn lớn và định cấu hình kiến trúc mạng có thể tìm hiểu các tính năng và mô hình. Kỹ thuật này đặc biệt hữu ích cho các ứng dụng mới, cũng như các ứng dụng có số lượng lớn các danh mục đầu ra. Tuy nhiên, về tổng thể, đây

là một cách tiếp cận ít phổ biến hơn, vì nó yêu cầu lượng dữ liệu không theo thứ tự, khiến quá trình đào tạo mất vài ngày hoặc vài tuần.

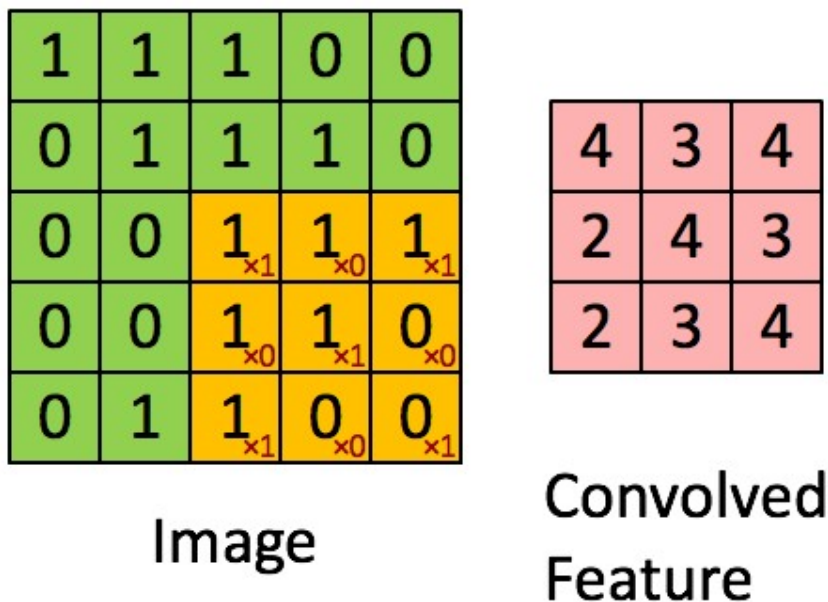
**Dropout:** Phương pháp này cố gắng giải quyết vấn đề trang bị quá nhiều trong các mạng có lượng lớn tham số bằng cách loại bỏ ngẫu nhiên các đơn vị và kết nối của chúng khỏi mạng nơ-ron trong quá trình huấn luyện. Người ta đã chứng minh rằng phương pháp bỏ học có thể cải thiện hiệu suất của mạng nơ-ron trong các nhiệm vụ học tập có giám sát trong các lĩnh vực như nhận dạng giọng nói, phân loại tài liệu và sinh học tính toán.

#### 5.4. Mô hình học sâu CNN

Convolutional Neural Network (CNNs – Mạng nơ-ron tích chập) là một trong những mô hình Deep Learning tiên tiến. Nó giúp cho chúng ta xây dựng được những hệ thống thông minh với độ chính xác cao như hiện nay.

##### 5.4.1. Khái niệm Convolutional

Là một cửa sổ trượt (Sliding Windows) trên một ma trận như mô tả hình dưới:



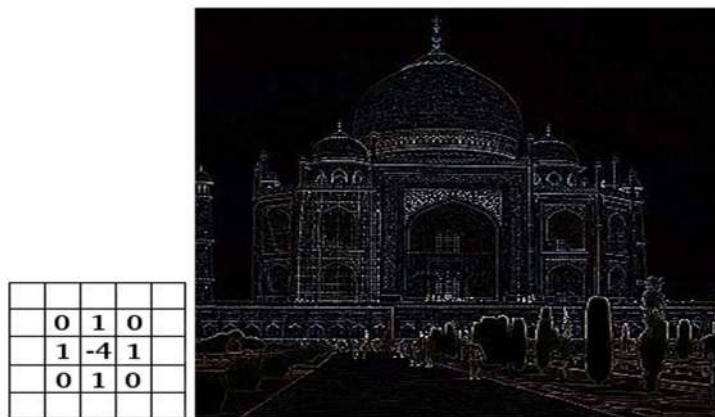
HÌNH 15. MA TRẬN HÌNH ẢNH

Các convolutional layer có các parameter(kernel) đã được học để tự điều chỉnh lấy ra những thông tin chính xác nhất mà không cần chọn các feature.

Trong hình ảnh ví dụ trên, ma trận bên trái là một hình ảnh trắng đen được số hóa. Ma trận có kích thước  $5 \times 5$  và mỗi điểm ảnh có giá trị 1 hoặc 0 là giao điểm của dòng và cột.

Convolution hay tích chập là nhân từng phần tử trong ma trận 3. Sliding Window hay còn gọi là kernel, filter hoặc feature detect là một ma trận có kích thước nhỏ như trong ví dụ trên là  $3 \times 3$ .

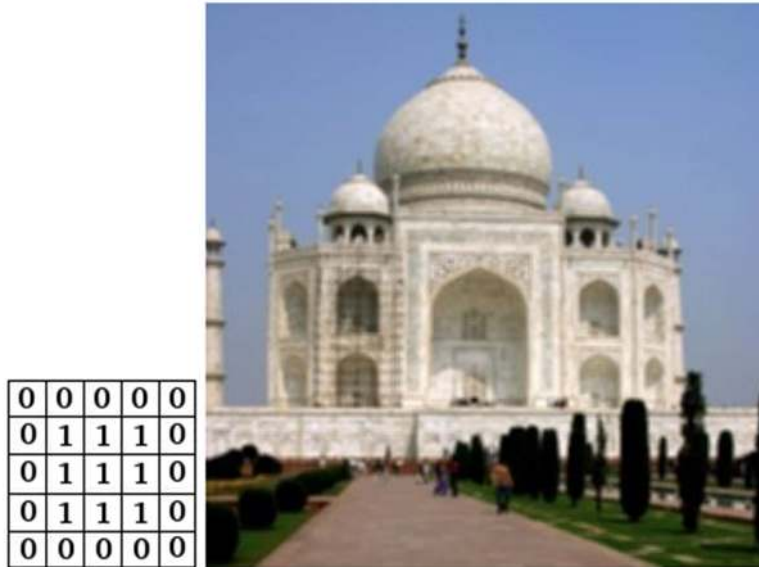
Convolution hay tích chập là nhân từng phần tử bên trong ma trận  $3 \times 3$  với ma trận bên trái. Kết quả được một ma trận gọi là Convoled feature được sinh ra từ việc nhân ma trận Filter với ma trận ảnh  $5 \times 5$  bên trái.



HÌNH 16. HÌNH ẢNH ĐEN TRẮNG QUA QUÁ TRÌNH TÍCH CHẬP

So sánh với hình ảnh màu qua quá trình tích chập: (Hình 1.18)





HÌNH 17. HÌNH ẢNH MÀU QUA QUÁ TRÌNH TÍCH CHẬP

#### 5.4.2. Cấu trúc mạng CNN

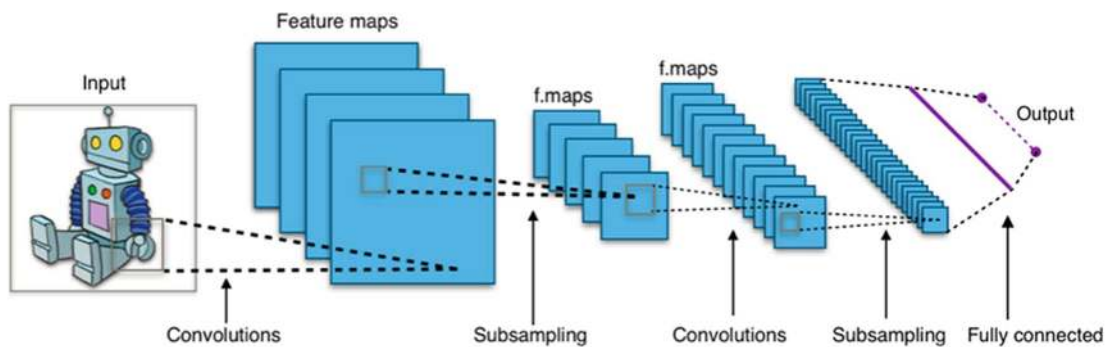
Mạng CNN là một tập hợp các lớp Convolution chồng lên nhau và sử dụng các hàm nonlinear activation như ReLU và tanh để kích hoạt các trọng số trong các node. Mỗi một lớp sau khi thông qua các hàm kích hoạt sẽ tạo ra các thông tin trừu tượng hơn cho các lớp tiếp theo.

Mỗi một lớp sau khi thông qua các hàm kích hoạt sẽ tạo ra các thông tin trừu tượng hơn cho các lớp tiếp theo. Trong mô hình mạng truyền ngược (feedforward neural network) thì mỗi neural đầu vào (input node) cho mỗi neural đầu ra trong các lớp tiếp theo. Mô hình này gọi là mạng kết nối đầy đủ (fully connected layer) hay mạng toàn vẹn (affine layer). Còn trong mô hình CNNs thì ngược lại. Các layer liên kết được với nhau thông qua cơ chế convolution.

Layer tiếp theo là kết quả convolution từ layer trước đó, nhờ vậy mà ta có được các kết nối cục bộ. Như vậy mỗi neuron ở lớp kế tiếp sinh ra từ kết quả của filter áp đặt lên một vùng ảnh cục bộ của neuron trước đó. Mỗi một lớp được sử dụng các filter khác nhau thông thường có hàng trăm hàng nghìn filter như vậy và kết hợp kết quả của chúng lại. Ngoài ra có một số layer khác như: pooling/

subsampling layer dùng để chắt lọc lại các thông tin hữu ích hơn (loại bỏ các thông tin nhiễu).

Trong quá trình huấn luyện mạng (training) CNN tự động học các giá trị qua các lớp filter dựa vào cách thức mà bạn thực hiện. Ví dụ trong tác vụ phân lớp ảnh, CNNs sẽ cố gắng tìm ra thông số tối ưu cho các filter tương ứng theo thứ tự raw pixel > edges > shapes > facial > high-level features. Layer cuối cùng được dùng để phân lớp ảnh.



HÌNH 18. CẤU TRÚC MẠNG CNN

Trong mô hình CNN có 2 khía cạnh cần quan tâm là tính bất biến (Location Invariance) và tính kết hợp (Compositionality). Với cùng một đối tượng, nếu đối tượng này được chiếu theo các góc độ khác nhau (translation, rotation, scaling) thì độ chính xác của thuật toán sẽ bị ảnh hưởng đáng kể.

Pooling layer sẽ cho bạn tính bất biến đối với phép dịch chuyển (translation), phép quay (rotation) và phép co giãn (scaling). Tính kết hợp cục bộ cho ta các cấp độ biểu diễn thông tin từ mức độ thấp đến mức độ cao và trừu tượng hơn thông qua convolution từ các filter.

Đó là lý do tại sao CNNs cho ra mô hình với độ chính xác rất cao. Cũng giống như cách con người nhận biết các vật thể trong tự nhiên.

Mạng CNN sử dụng 3 ý tưởng cơ bản:

- Các trường tiếp nhận cục bộ (local receptive field)
- Trọng số chia sẻ (shared weights)
- Tổng hợp (pooling)

Cách chọn tham số cho CNN

- Số các convolution layer: càng nhiều các convolution layer thì performance càng được cải thiện. Sau khoảng 3 hoặc 4 layer, các tác động được giảm một cách đáng kể
- Filter size: thường filter theo size  $5 \times 5$  hoặc  $3 \times 3$
- Pooling size: thường là  $2 \times 2$  hoặc  $4 \times 4$  cho ảnh đầu vào lớn
- Cách cuối cùng là thực hiện nhiều lần việc train test để chọn ra được param tốt nhất.

## CHƯƠNG II - ỨNG DỤNG HỌC MÁY – HỌC SÂU TRONG SẢN PHẨM

### 1. Dữ liệu mẫu

Trong khuôn khổ bài báo cáo này, nhóm thu thập X tệp dữ liệu mẫu, ở đây là các tệp PE; được chia thành hai loại chính:

- Tệp sạch: Bao gồm các tệp thực thi mặc định của chính Windows hay của các tổ chức khác đã được chứng minh là an toàn, chủ yếu nguồn mã sạch được lấy từ gói benign được msdn cung cấp.
- Mã độc: Các tệp thu thập được từ trang VirusShare (<https://virusshare.com>). Các tệp mã độc bao gồm các dạng như: Trojan, Virus, Worms, Adware và các dạng khác. Trong đó có các tệp bị đóng gói cũng như các tệp không ở định dạng PE sẽ bị loại bỏ trong quá trình chọn lọc.

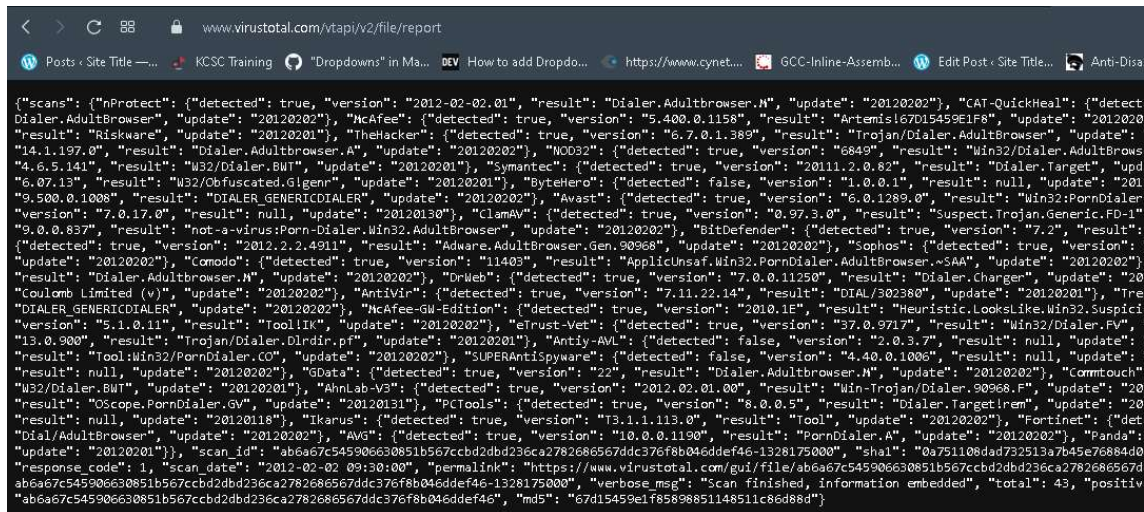
### 2. Đánh nhãn

Sau khi có cơ sở dữ liệu các mẫu mã độc từ quá trình thu thập. Ở đây ta thu được thông tin về mã băm của các mẫu dưới dạng "md5". Tiếp theo nhóm sẽ tìm kiếm các thông tin về từng mẫu mã độc bằng dữ liệu dạng "md5" đã có ở trên, kết quả sẽ trả về ở dạng file JSON. Ở đây để thu thập thông tin, nhóm tác giả sử dụng một công cụ đó là VirusTotal API.

VirusTotal là một dịch vụ web phân tích các tệp mã độc và tạo điều kiện thuận lợi cho việc phát hiện nhanh chóng virus, worm, trojan, ... và tất cả các loại mã độc được cung cấp bởi các hãng chống virus khác nhau. Các Website như trang VirusTotal (<http://www.virustotal.com>) cho phép bạn tải lên một tập tin và sẽ được quét bởi nhiều chương trình Antivirus khác nhau. VirusTotal sau đó sẽ cung cấp một báo cáo cung cấp tổng số các công cụ Antivirus đánh dấu tập tin này là phần mềm độc hại, tên của mã độc này, và nếu có thể sẽ cung cấp cho bạn nhiều thông tin hơn về tập tin mã độc trên.

VirusTotal cho phép người dùng sử dụng đồng thời 69 phần mềm bảo mật hàng đầu thế giới hiện nay, như BitDefender, AVG, Kaspersky, Avira... để quét và

kiểm tra file. Có thể nói, với VirusTotal, việc bỏ lọt file chứa mã độc là điều gần như không thể.



HÌNH 19. KẾT QUẢ VIRUSTOTAL TRẢ VỀ KHI QUERY BẰNG HASH MD5

Ở đây, cần lưu ý trường "suggested\_threat\_label" là nhãn của virusTotal tổng hợp thông kê kết quả của các AV khác. Trong trường hợp trường nêu trên không có sẵn, ta sẽ lấy các kết quả trùng với các phần tử trong while\_label = ["trojan", "adware", "downloader", "virus", "worm", "backdoor", "pua", "hacktool", "ransom", "spyware"]; Sau đó mới ưu tiên kết quả trong top 15 AV: av\_rank = ['Avast', 'AVG', 'Avira', 'BitDefender', 'ESET', 'GData', 'K7AntiVirus', 'Kaspersky', 'Malwarebytes', 'McAfee', 'Microsoft', 'Panda', 'TotalDefense', 'TrendMicro', 'VIPRE'].

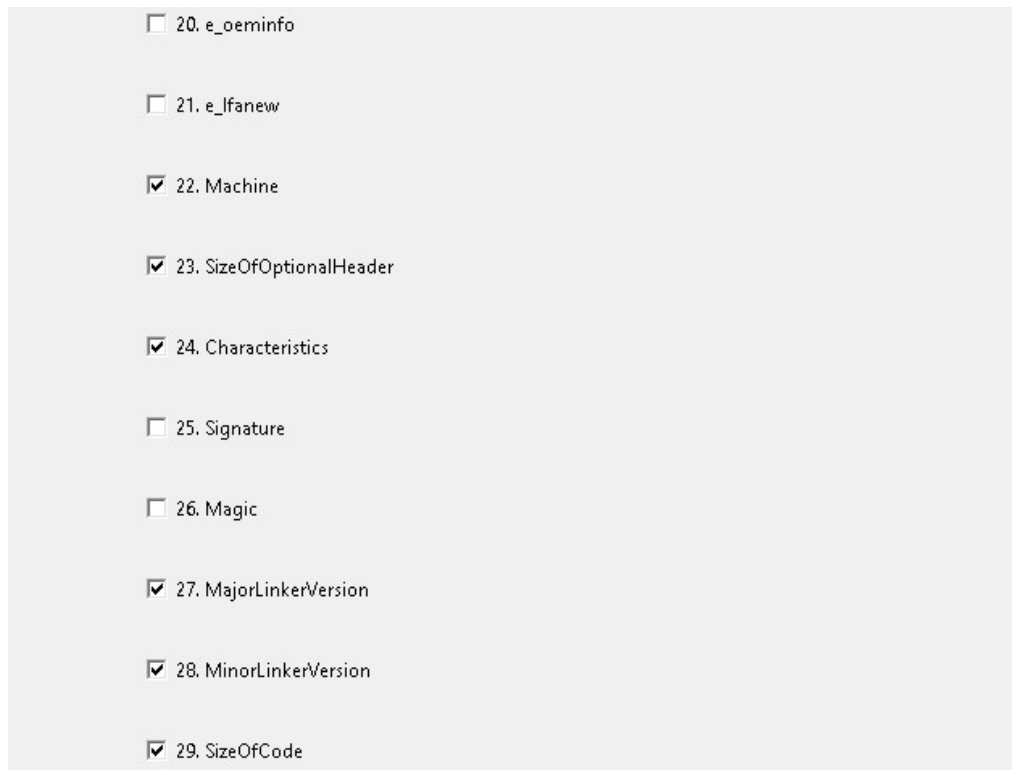
Kết quả của quá trình tùy theo từng AV sẽ trả về theo model học máy của chính AV đó, ta cần tạo một danh sách để lược bỏ các trường không có giá trị: black\_label = ["unsafe", "malicious", "staticai", "aidetect"]. Từ đây, kết quả cuối cùng thu được là một tệp với định dạng csv gồm 2 trường "md5" và "label".

Khi đã có dữ liệu về nhãn của từng mẫu mã độc, nhóm đổi tên các mẫu dưới dạng name\_label để thuận tiện cho quá trình đánh giá, kiểm thử sau này.

### 3. Trích xuất đặc trưng

Hiện tại ở version mới nhất của sản phẩm, người dùng có thể chọn thủ công các đặc trưng của file PE thay vì lấy mặc định như các version trước.

Việc chọn thủ công các đặc trưng như vậy cho phép người dùng rút ra sự so sánh về mức độ chính xác khi tăng giảm số lượng các đặc trưng khi đưa vào mô hình.



<input type="checkbox"/>	20. e_oeminfo
<input type="checkbox"/>	21. e_lfanew
<input checked="" type="checkbox"/>	22. Machine
<input checked="" type="checkbox"/>	23. SizeOfOptionalHeader
<input checked="" type="checkbox"/>	24. Characteristics
<input type="checkbox"/>	25. Signature
<input type="checkbox"/>	26. Magic
<input checked="" type="checkbox"/>	27. MajorLinkerVersion
<input checked="" type="checkbox"/>	28. MinorLinkerVersion
<input checked="" type="checkbox"/>	29. SizeOfCode

HÌNH 20. BẢNG CÁC ĐẶC TRƯNG NGƯỜI DÙNG CÓ THỂ CHỌN THỦ CÔNG

Quá trình trích chọn các đặc trưng bao gồm:

B1. Lấy toàn bộ đặc trưng có được từ 1 file PE và lưu vào 1 file JSON.

B2. Căn cứ vào các tùy chọn nhận được từ việc chọn thủ công các đặc trưng bên trên, chương trình sẽ chọn ra các đặc trưng đó từ file JSON và lưu vào trong file CSV.

Mọi công đoạn đều được thiết kế dưới dạng giao diện người dùng dễ sử dụng.

Do nguồn dữ liệu Malware có nhãn cũng như mã sạch còn hạn chế, hiện tại sản phẩm của nhóm mới chỉ detect được các file PE có các nhãn sau:

- Mã sạch
- Trojan
- Virus
- Worm

Để tiện lợi cho việc test và đưa vào mô hình để học, các file Malware sẽ được giữ nguyên tên theo dạng VirusShare\_[mã MD5]\_[nhãn virus], nhờ đó, việc đánh nhãn sẽ được lưu theo dạng:

- 1: Mã sạch
- 2: Lỗi không phân tích được, có thể là do file đó không phải PE, hay bị pack nên không lấy được đặc trưng.
- 3: Trojan
- 4: Virus
- 5: Worm

## **4. Một số thư viện hỗ trợ AI và ứng dụng ATTT**

### **4.1. Keras/TensorFlow**

#### **4.1.1. Keras**

Keras là một open source cho Neural Network được viết bởi ngôn từ Python. Nó là một library được tăng trưởng vào năm 2015 bởi Francois Chollet, là một kỹ sư nghiên cứu và điều tra Deep Learning. Keras hoàn toàn có thể sử dụng chung với những thư viện nổi tiếng như Tensorflow, CNTK, Theano. Một số ưu điểm của Keras như :

- Dễ sử dụng, dùng đơn thuần hơn Tensor, thiết kế xây dựng Mã Sản Phẩm nhanh.
- Run được trên cả CPU và GPU.
- Hỗ trợ thiết kế xây dựng CNN, RNN hoặc cả hai.
- Với những người mới tiếp cận đến Deep như mình thì mình chọn sử dụng Keras để build Model vì nó đơn thuần, dễ chớp lấy hơn những thư viện khác. Dưới đây mình xin trình làng một chút ít về API này .

### **Model Keras**

Trong Keras có tương hỗ 2 cách dựng models là Sequential Model và Function API.

Với Sequential ta sử dụng như sau :

```

from keras.models import Sequential
from keras.layers import Dense, MaxPooling2D, Flatten
Convolution2Dmodel = Sequential()
model.add(Convolution2D(32,3,3,input_shape=(64,64,3),activation= "relu"))
model.add(MaxPooling2D(pool_size = (2, 2)))
model.add(Flatten())
model.add(Dense(output_dim = 128, activation="relu"))
model.add(Dense(output_dim = 1, activation = "sigmoid" ) )
model.compile(optimizer = " adam ", loss = "binary_crossentropy" )
model.fit(x_train, y_train, batch_size = batch_size,
          epochs = epochs, verbose = 1, validation_data = ( x_test, y_test ) )

```

HÌNH 21. MINH HỌA VỀ CÁCH SỬ DỤNG SEQUENTIAL QUA THƯ VIỆN KERAS

Khởi tạo models Sequential( ) Tạo Convolutional Layers : Conv2D là convolution dùng để lấy feature từ ảnh với những tham số : filters : số filter của convolutionkernel\_size : size window search trên ảnhstrides : số bước nhảy trên ảnh activation : chọn activation như linear, softmax, relu, tanh, sigmoid. Đặc điểm mỗi hàm những bạn hoàn toàn có thể search thêm để biết đơn cử nó ntn.padding : hoàn toàn có thể là “ valid ” hoặc “ same ”. Với same thì có nghĩa là padding = 1. Pooling Layers : sử dụng để làm giảm param khi train, nhưng vẫn giữ được đặc trưng của ảnh. pool\_size : size ma trận để lấy max hay average Ngoài ra còn có : MaxPooling2D.

AveragePooling1D, 2D ( lấy max, trung bình ) với từng size. Dense ( ) : Layer này cũng như một layer neural network thông thường, với những tham số sau : units : số chiều output, như số class sau khi train ( chó, mèo, lợn, gà ). activation : chọn activation đơn thuần với sigmoid thì output có 1 class. use\_bias : có sử dụng bias hay không ( True or False ) Hàm compile : Ở hàm này tất cả chúng ta sử dụng để training models như thuật toán train qua optimizer như Adam, SGD, RMSprop, .. learning\_rate : dạng float, tốc độ học, chọn tương thích để hàm số quy tụ nhanh. Hàm fit ( ) : Bao gồm data train, test đưa vào training. Batch\_size biểu lộ số lượng mẫu mà Mini-batch GD sử dụng cho mỗi lần update trọng số. Epoch là số lần duyệt qua hết số lượng mẫu trong tập huấn luyện.

Giả sử ta có tập huấn luyện gồm 55.000 hình ảnh chọn batch-size là 55 images có nghĩa là mỗi lần update trọng số, ta dùng 55 images. Lúc đó ta mất  $55.000 / 55 = 1.000$  iterations ( số lần lặp ) để duyệt qua hết tập huấn luyện ( triển khai xong 1 epochs ). Có nghĩa là khi tài liệu quá lớn, tất cả chúng ta không hề đưa cả



tập data vào train được, ta phải chia nhỏ data ra thành nhiều batch nhỏ hơn. Ngoài ra ta hoàn toàn có thể khai báo như sau :

```
from keras.models import Model
from keras.layers import Input
Denseinput = Input(shape = (64,))
output = Dense(32)(input)
model = Model(input = input, output= output)

#Tiền xử lý
train_datagen = ImageDataGenerator( rescale=1./255, shear_range=0.2,
    zoom_range=0.2, horizontal_flip=True)
test_datagen = ImageDataGenerator(rescale=1./255)
```

HÌNH 22. CHIA NHỎ DỮ LIỆU ĐẦU VÀO

Sequence Preprocessing : tiền giải quyết và xử lý chuỗi. Time series Generator: tạo data cho time series  
pad\_sequences : padding những chuỗi có độ dài bằng nhau  
skipgrams : tạo data trong models skip gram, trả về 2 tuple nếu từ Open cùng nhau, là 1 nếu không có. Text Preprocessing : tiền giải quyết và xử lý text  
Tokenizer : tạo token từ document  
one\_hot : tạo data dạng one hot encoding  
text\_to\_word\_sequence : convert text thành sequence. Image Preprocessing : tiền giải quyết và xử lý image  
ImageDataGenerator : tạo thêm data bằng cách scale, rotation, ... để thêm data train  
Note.

### Loss\_funtion

mean\_squared\_error : thường dùng trong regression tính theo:

euclid  
mean\_absolute\_error: để tính giá trị tuyệt đối  
binary\_crossentropy: dùng cho classifier 2 class  
categorical\_crossentropy : dùng classifier nhiều class.

### Metrics

Để nhìn nhận accuracy của models.

binary\_accuracy: dùng cho 2 class, nếu y\_true == y\_predict thì trả về 1 ngược lại là 0. categorical\_accuracy : cũng giống như trên nhưng cho nhiều class.

### Optimizers

Dùng để chọn thuật toán training

SGD : Stochastic Gradient Descent optimizer  
RMSprop  
Adam.

### Callbacks

Khi models tất cả chúng ta lớn khi training thì gặp lỗi ta muốn lưu lại models để chạy lại thì ta sử dụng callbacks.

ModelsCheckpoint : lưu lại model sau mỗi epoch  
EarlyStopping : stop training khi models training không hiệu quả  
ReduceLROnPlateau : giảm learning mỗi khi metrics không cải thiện.

#### **4.1.2. Tensorflow**

TensorFlow chính là thư viện mã nguồn mở cho machine learning nổi tiếng nhất thế giới, được phát triển bởi các nhà nghiên cứu từ Google. Việc hỗ trợ mạnh mẽ các phép toán học để tính toán trong machine learning và deep learning đã giúp việc tiếp cận các bài toán trở nên đơn giản, nhanh chóng và tiện lợi hơn nhiều.

Các hàm được dựng sẵn trong thư viện cho từng bài toán cho phép TensorFlow xây dựng được nhiều neural network. Nó còn cho phép bạn tính toán song song trên nhiều máy tính khác nhau, thậm chí trên nhiều CPU, GPU trong cùng 1 máy hay tạo ra các dataflow graph – đồ thị luồng dữ liệu để dựng nên các model. Nếu bạn muốn chọn con đường sự nghiệp trong lĩnh vực A.I. này, nắm rõ những điều cơ bản của TensorFlow thực sự rất quan trọng.

Được viết bằng C++ và thao tác interface bằng Python nên phần performance của TensorFlow cực kỳ tốt. Đối tượng sử dụng nó cũng đa dạng không kém: từ các nhà nghiên cứu, nhà khoa học dữ liệu và dĩ nhiên không thể thiếu các lập trình viên.

Kiến trúc TensorFlow hoạt động được chia thành 3 phần:

- Tiền xử lý dữ liệu
- Dựng model
- Train và ước tính model

#### **Cách TensorFlow hoạt động**

TensorFlow cho phép các lập trình viên tạo ra dataflow graph, cấu trúc mô tả làm thế nào dữ liệu có thể di chuyển qua 1 biểu đồ, hay 1 sê-ri các node đang xử lý. Mỗi node trong đồ thị đại diện 1 operation toán học, và mỗi kết nối

hay edge giữa các node là 1 mảng dữ liệu đa chiều, hay còn được gọi là ‘tensor’.

TensorFlow cung cấp tất cả những điều này cho lập trình viên theo phương thức của ngôn ngữ Python. Vì Python khá dễ học và làm việc, ngoài ra còn cung cấp nhiều cách tiện lợi để ta hiểu được làm thế nào các high-level abstractions có thể kết hợp cùng nhau. Node và tensor trong TensorFlow là các đối tượng Python, và các ứng dụng TensorFlow bản thân chúng cũng là các ứng dụng Python.

Các operation toán học thực sự thì không được thi hành bằng Python. Các thư viện biến đổi có sẵn thông qua TensorFlow được viết bằng các binary C++ hiệu suất cao. Python chỉ điều hướng lưu lượng giữa các phần và cung cấp các high-level abstraction lập trình để nối chúng lại với nhau.

TensorFlow 2.0, được ra mắt vào tháng 10 năm 2019, cải tiến framework theo nhiều cách dựa trên phản hồi của người dùng, để dễ dàng và hiệu quả hơn khi làm việc cùng nó (ví dụ: bằng cách sử dụng các Keras API liên quan đơn giản cho việc train model). Train phân tán dễ chạy hơn nhờ vào API mới và sự hỗ trợ cho TensorFlow Lite cho phép triển khai các mô hình trên khá nhiều nền tảng khác nhau. Tuy nhiên, nếu đã viết code trên các phiên bản trước đó của TensorFlow thì bạn phải viết lại, đôi lúc 1 ít, đôi lúc cũng khá đáng kể, để tận dụng tối đa các tính năng mới của TensorFlow 2.0.

Lợi ích dễ thấy nhưng quan trọng nhất mà TensorFlow cung cấp cho việc lập trình machine learning chính là abstraction. Thay vì phải đối phó với những tình huống rườm rà từ việc thực hiện triển khai các thuật toán, hay tìm ra cách hợp lý để chuyển output của 1 chức năng sang input của 1 chức năng khác, giờ đây bạn có thể tập trung vào phần logic tổng thể của 1 ứng dụng hơn. TensorFlow sẽ chăm sóc phần còn lại thay cho bạn.

Ngoài ra TensorFlow còn cung cấp các tiện ích bổ sung cho các lập trình viên cần debug cũng như giúp bạn tự suy xét các ứng dụng TensorFlow. Chế độ eager execution cho phép bạn đánh giá và sửa đổi từng operation của biểu đồ 1 cách riêng biệt và minh bạch, thay vì phải dựng toàn bộ biểu đồ dưới dạng 1 đối tượng độc lập vốn khá mơ hồ hay phải đánh giá chung tổng thể. Cuối cùng,

1 tính năng khá độc đáo của TensorFlow là TensorBoard. TensorBoard cho phép bạn quan sát 1 cách trực quan những gì TensorFlow đang làm.

TensorFlow còn có nhiều cải tiến từ sự hậu thuẫn từ các ekip thương mại hạng A tại Google. Google không những tiếp lửa cho tiến độ nhanh chóng cho sự phát triển đằng sau dự án, mà còn tạo ra nhiều phục vụ độc đáo xung quanh TensorFlow để nó dễ dàng deploy và sử dụng: như silicon TPU mình đã nói ở trên để tăng tốc hiệu suất đám mây Google, 1 online hub cho việc chia sẻ các model được tạo với framework, sự hiện diện của in-browser và gần gũi với mobile của framework, và nhiều hơn thế nữa...

Lưu ý: Trong 1 số công việc training, vài chi tiết về việc triển khai của TensorFlow làm cho nó khó có thể quyết định được hoàn toàn kết quả training model. Đôi khi 1 model được train trên 1 hệ thống này sẽ có thay đổi 1 chút so với 1 model được train trên hệ thống khác, ngay cả khi chúng được cung cấp dữ liệu như nhau. Các nguyên nhân cho điều này cũng xê xích hay 1 số hành vi khi không được xác định khi sử dụng GPU. Điều này nói rằng, các vấn đề đó có thể giải quyết được, và đội ngũ của TensorFlow cũng đang xem xét việc kiểm soát nhiều hơn để ảnh hưởng đến tính quyết định trong quy trình làm việc.

**Tensor:** Tên của TensorFlow được đưa ra trực tiếp là nhờ vào framework cốt lõi của nó: Tensor. Trong TensorFlow, tất cả các tính toán đều liên quan tới các tensor. 1 tensor là 1 vector hay ma trận của n-chiều không gian đại diện cho tất cả loại dữ liệu. Tất cả giá trị trong 1 tensor chứa đựng loại dữ liệu giống hệt nhau với 1 shape đã biết (hoặc đã biết 1 phần). Shape của dữ liệu chính là chiều của ma trận hay mảng.

Một tensor có thể được bắt nguồn từ dữ liệu input hay kết quả của 1 tính toán. Trong TensorFlow, tất cả các hoạt động được tiến hành bên trong 1 graph – biểu đồ. Biểu đồ là 1 tập hợp tính toán được diễn ra liên tiếp. Mỗi operation được gọi là 1 op node (operation node) và được kết nối với nhau.

Biểu đồ phát thảo các op và kết nối giữa các node. Tuy nhiên, nó không hiển thị các giá trị. Phần edge của các node chính là tensor, 1 cách để nhập các phép tính với dữ liệu.

**Graph:** TensorFlow sử dụng framework dạng biểu đồ. Biểu đồ tập hợp và mô tả tất cả các chuỗi tính toán được thực hiện trong quá trình training. Biểu đồ cũng mang rất nhiều lợi thế:

- Nó được làm ra để chạy trên nhiều CPU hay GPU, ngay cả các hệ điều hành trên thiết bị điện thoại.
- Tính di động của biểu đồ cho phép bảo toàn các tính toán để bạn sử dụng ngay hay sau đó. Biểu đồ có thể được lưu lại để thực thi trong tương lai.
- Tất cả tính toán trong biểu đồ được thực hiện bằng cách kết nối các tensor lại với nhau. 1 tensor có 1 node và 1 edge. Node mang operation toán học và sản xuất các output ở đầu cuối. Các edge giải thích mối quan hệ input/output giữa các node.

#### **4.2. Theano**

Theano là một thư viện Python cho phép bạn xác định, tối ưu hóa và đánh giá các biểu thức toán học, đặc biệt là các biểu thức với mảng đa chiều (numpy.ndarray). Sử dụng Theano, có thể đạt được tốc độ ngang ngửa với việc triển khai C thủ công cho các vấn đề liên quan đến lượng lớn dữ liệu. Nó cũng có thể vượt qua C trên CPU theo nhiều cấp độ bằng cách tận dụng sức mạnh của các GPU thế hệ mới.

Theano là trái tim của các framework cấp cao hơn (ví dụ, Tensorflow). Mục đích của Theano là trở thành thư viện toán học tượng trưng, và như vậy, bạn cần phải khá quen thuộc với các thuật toán thường được ẩn đi. Nó được phát triển bởi Frédéric Bastien, từ phòng thí nghiệm MILA của Đại học Montreal.

Theano kết hợp các khía cạnh của hệ thống đại số máy tính (CAS) với các khía cạnh của trình biên dịch tối ưu hóa. Nó cũng có thể tạo code C tùy chỉnh cho nhiều phép toán. Sự kết hợp giữa CAS với việc tối ưu hóa biên dịch này đặc biệt hữu ích cho các nhiệm vụ trong đó các biểu thức toán học phức tạp được biên dịch lặp đi lặp lại và tốc độ biên dịch là rất quan trọng. Đối với các tình huống mà mỗi biểu thức khác nhau được đánh giá một lần, Theano có thể tối ưu hóa việc biên dịch / phân tích, nhưng vẫn cung cấp các tính năng tượng trưng như phân biệt tính toán.

Trình biên dịch của Theano áp dụng nhiều cách tối ưu hóa có độ phức tạp khác nhau cho các biểu thức tượng trưng này. Những tối ưu hóa này bao gồm:

- Sử dụng GPU để tính toán
- Constant folding
- Hợp nhất các đồ thị con tương tự, để tránh tính toán dư thừa
- Đơn giản hóa số học (ví dụ:  $x*y/x \rightarrow y$ ,  $y-x \rightarrow x$ )
- Chèn các hoạt động BLAS hiệu quả (ví dụ GEMM) trong nhiều ngữ cảnh khác nhau
- Sử dụng bộ nhớ Aliasing để tránh tính toán
- Sử dụng các hoạt động tại chỗ khi nó không ảnh hưởng đến Aliasing
- Hợp nhất vòng lặp cho các biểu thức con theo nguyên tố
- Cải thiện sự ổn định số (ví dụ  $\log(1 + \exp(x))$  và  $\log(\sum_i \exp(x[i]))$ )
- Ngoài ra còn những tối ưu hoá ở nhiều khía cạnh khác.

Theano được viết tại phòng thí nghiệm LISA để hỗ trợ phát triển nhanh chóng các thuật toán học máy hiệu quả, được phát hành theo giấy phép BSD. Theano được đặt theo tên của nhà toán học Hy Lạp, người có thể là vợ của Pythagoras.

#### 4.2.1. Sơ lược

Dưới đây là một ví dụ về cách sử dụng Theano. Nó không thể hiện nhiều tính năng của Theano, nhưng nó minh họa cụ thể Theano là gì.

```
import theano
from theano import tensor as tt

# declare two symbolic floating-point scalars
a = tt.dscalar()
b = tt.dscalar()

# create a simple expression
c = a + b

# convert the expression into a callable object that takes (a,b)
# values as input and computes a value for c
f = theano.function([a,b], c)

# bind 1.5 to 'a', 2.5 to 'b', and evaluate 'c'
assert 4.0 == f(1.5, 2.5)
```

HÌNH 23. VÍ DỤ VỀ CÁCH SỬ DỤNG THEANO

Theano không phải là một ngôn ngữ lập trình theo nghĩa thông thường vì bạn viết một chương trình bằng Python để xây dựng các biểu thức cho Theano. Tuy nhiên, nó giống như một ngôn ngữ lập trình ở một số điểm, đó là bản phải:

- Khai báo các biến (a, b) và đưa ra các kiểu của chúng
- Xây dựng các biểu thức về cách kết hợp các biến đó lại với nhau
- Biên dịch đồ thị biểu thức thành các hàm để sử dụng chúng để tính toán.

Có thể coi `theano.function` như giao diện của một trình biên dịch, nó xây dựng một đối tượng có thể gọi từ một biểu đồ tượng trưng thuần túy. Một trong những tính năng quan trọng nhất của Theano là `theano.function` có thể tối ưu hóa một biểu đồ và thậm chí biên dịch một số hoặc tất cả chúng thành các lệnh máy gốc.

#### 4.2.2. Khả năng của Theano

Theano là một thư viện Python và trình biên dịch tối ưu hóa để thao tác và đánh giá các biểu thức, đặc biệt là các biểu thức có giá trị ma trận. Thao tác với ma trận thường được thực hiện bằng cách sử dụng gói `numpy`, vậy Theano có gì mà Python và `numpy` không thực hiện được?

- *Tối ưu hóa tốc độ thực thi*: Theano có thể sử dụng `g++` hoặc `nvcc` để biên dịch các phần của biểu đồ biểu thức của bạn thành các lệnh CPU hoặc GPU, chạy nhanh hơn nhiều so với Python thuần túy.
- *Sự khác biệt biểu tượng*: Theano có thể tự động xây dựng các đồ thị biểu tượng cho độ dốc tính toán.
- *Tối ưu hóa độ ổn định*: Theano có thể nhận ra (một số) biểu thức số không ổn định và tính toán chúng với các thuật toán ổn định hơn.

Gói Python giống với Theano nhất là `Sympy`. Theano tập trung nhiều hơn vào các biểu thức tensor hơn `Sympy`, và để biên dịch máy móc hơn. `Sympy` có các quy tắc đại số phức tạp hơn và có thể xử lý nhiều phép toán khác nhau (chẳng hạn như chuỗi, giới hạn và tích phân).

Nếu như `numpy` được so sánh với MATLAB và `Sympy` với Mathematica, thì Theano là một dạng kết hợp giữa cả hai, cố gắng kết hợp những gì tốt nhất của cả hai thế giới.

#### 4.2.3. Định hướng của Theano

- Hỗ trợ tensor và các hoạt động tách biệt
- Hỗ trợ các phép toán đại số tuyến tính
- Biến đổi đồ thị
- Có thể sử dụng nhiều ngôn ngữ biên dịch, bộ cài đặt: C / C ++, CUDA, OpenCL, PTX, CAL, AVX, ...
- Lazy evaluation
- Vòng lặp
- Thực hiện đồng thời các tác vụ (SIMD, đa lõi, nhiều nút trên cụm, nhiều nút được phân phối)
- Hỗ trợ tất cả chức năng NumPy / SciPy cơ bản
- Dễ dàng gói các chức năng thư viện trong Theano

Ngoài những ưu điểm linh hoạt và biểu diễn, hỗ trợ RNN hàng đầu thì nó cũng có những nhược điểm như API cấp thấp. Bạn cần phải là một chuyên gia về toán học cung cấp năng lực cho các thuật toán để tận dụng tối đa nó.

#### 4.3. CNTK

CNTK (Bộ công cụ mạng tính toán) là bộ công cụ học sâu do Microsoft phát triển nhằm mô tả các mạng “neural” là một chuỗi các bước tính toán thông qua biểu đồ có hướng. Nó có thể được sử dụng để giúp bạn dễ dàng kết hợp các loại mạng “neural” với nhau, có hiệu suất tuyệt vời, cho phép đào tạo phân tán và linh hoạt.

CNTK cho phép người dùng dễ dàng nhận ra và kết hợp các loại mô hình phổ biến như DNN chuyển tiếp, mạng “neural” tích hợp (CNN) và mạng “neural” tuần hoàn (RNN / LSTM). CNTK triển khai tính năng học phân tích độ giảm ngẫu nhiên (SGD, lan truyền lỗi) với khả năng phân biệt và song song tự động trên nhiều GPU và máy chủ.

CNTK có thể được đưa vào như một thư viện trong các chương trình Python, C # hoặc C ++ hoặc được sử dụng như một công cụ học máy độc lập thông qua ngôn ngữ của riêng nó (BrainScript). Ngoài ra, bạn có thể sử dụng chức năng



đánh giá mô hình CNTK từ các chương trình Java của mình. CNTK hỗ trợ hệ điều hành Linux 64 bit hoặc Windows 64 bit.

#### 4.3.1. Các khái niệm CNTK

Các đầu vào, đầu ra và tham số của CNTK được tổ chức dưới dạng các "tensors". Mỗi "tensors" có một hạng: Một đại lượng vô hướng là một "tensors" hạng 0, một vectơ là một "tensors" hạng 1, một ma trận là một "tensors" hạng 2, v.v. Các kích thước khác nhau này gọi là trục.

Mỗi "tensor" CNTK có một số trục tĩnh và một số trục động. Các trục tĩnh có cùng độ dài trong suốt vòng đời của mạng. Các trục động giống như các trục tĩnh ở chỗ chúng xác định một nhóm có ý nghĩa của các số chứa trong "tensor" nhưng:

- Độ dài của chúng có thể thay đổi theo từng trường hợp.
- Chiều dài của chúng thường không được biết trước khi mỗi minibatch được giới thiệu.
- Chúng có thể được đặt trước.

Một minibatch cũng là một tensor. Do đó, nó có một trục động, được gọi là trục lô, có chiều dài có thể thay đổi từ minibatch sang minibatch. CNTK hỗ trợ một trục động bổ sung duy nhất. Nó đôi khi được gọi là trục trình tự nhưng nó không có tên riêng. Trục này cho phép làm việc với các chuỗi theo cấp cao. Khi các thao tác trên chuỗi được thực hiện, CNTK thực hiện một kiểm tra kiểu đơn giản để xác định xem việc kết hợp hai chuỗi có luôn an toàn hay không.

Một ví dụ khác trong đó các trục động cung cấp một giải pháp là học cách xếp hạng các tài liệu được đưa ra một truy vấn. Thông thường, dữ liệu đào tạo trong trường hợp này bao gồm một tập hợp các truy vấn, với mỗi truy vấn có một số tài liệu liên quan thay đổi. Mỗi cặp tài liệu - truy vấn bao gồm một đánh giá hoặc nhãn liên quan (ví dụ: tài liệu có liên quan cho truy vấn đó hay không).

Bây giờ, tùy thuộc vào cách chúng ta xử lý các từ trong mỗi tài liệu, chúng ta có thể đặt chúng trên trục tĩnh hoặc trục động. Để đặt chúng trên trục tĩnh, chúng ta

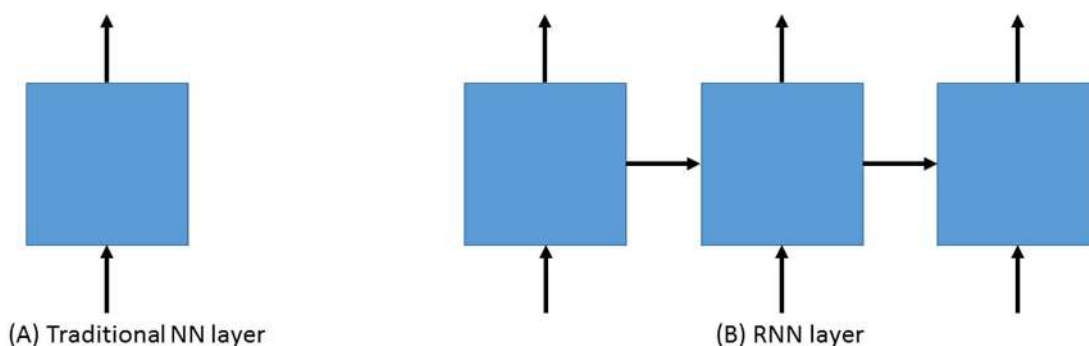
có thể xử lý mỗi tài liệu dưới dạng một vector có kích thước bằng với kích thước từ vựng của chúng ta chứa cho mỗi từ (hoặc cụm từ ngắn) số lần nó xuất hiện trong tài liệu. Tuy nhiên, chúng ta cũng có thể xử lý tài liệu thành một chuỗi các từ trong trường hợp đó chúng tôi sử dụng một trục động khác.

Cấp ngoài cùng là trục lô. Cấp tài liệu phải có trục động vì chúng ta có một số lượng tài liệu ứng viên thay đổi cho mỗi truy vấn. Mức trong cùng cũng nên có trục động vì mỗi tài liệu có số lượng từ thay đổi. Tensor mô tả minibatch này cũng sẽ có một hoặc nhiều trục tĩnh, mô tả các đặc điểm như nhận dạng của các từ trong truy vấn và tài liệu. Với dữ liệu huấn luyện đủ phong phú, có thể có một cấp lồng ghép khác, cụ thể là một phiên, trong đó có nhiều truy vấn liên quan thuộc về.

#### 4.3.2. Phân loại trình tự

Một trong những lĩnh vực thú vị nhất trong học sâu là ý tưởng về mạng “neural” tái tạo (RNN). Theo một số cách, RNN là Mô hình Markov ẩn của thế giới học sâu. Chúng là các mạng xử lý các chuỗi có độ dài thay đổi bằng cách sử dụng một tập hợp các tham số cố định. Do đó, họ phải học cách tóm tắt tất cả các quan sát trong chuỗi đầu vào thành trạng thái hữu hạn chiều, dự đoán quan sát tiếp theo bằng cách sử dụng trạng thái đó và chuyển trạng thái hiện tại và đầu vào quan sát thành trạng thái tiếp theo của chúng. Nói cách khác, chúng cho phép thông tin tồn tại.

Vì vậy, trong khi một lớp mạng “neural” truyền thống có thể được coi là có luồng dữ liệu đi qua như trong hình bên trái bên dưới, thì một lớp RNN có thể được xem như hình bên phải.



HÌNH 24. SỰ KHÁC BIỆT GIỮA LỚP NN TRUYỀN THỐNG VÀ LỚP RNN

Như hình bên phải, RNN là cấu trúc tự nhiên để xử lý các chuỗi. Điều này bao gồm mọi thứ từ văn bản đến âm nhạc đến video; bất cứ thứ gì mà trạng thái hiện tại phụ thuộc vào trạng thái trước đó. Mặc dù RNN thực sự mạnh mẽ, nhưng RNN “vani”, có trạng thái ở mỗi bước là một hàm phi tuyến của trạng thái trước đó và quan sát hiện tại, cực kỳ khó học thông qua các phương pháp dựa trên gradient. Bởi vì gradient cần phải di chuyển ngược lại qua mạng để học, nên sự đóng góp từ một phần tử ban đầu (ví dụ: một từ ở đầu câu) vào những phần tử sau đó, chẳng hạn như phân loại của từ cuối cùng trong một câu dài, có thể về cơ bản là biến mất.

Đối phó với vấn đề trên là một lĩnh vực nghiên cứu tích cực. Một kiến trúc có vẻ thành công trong thực tế là mạng Bộ nhớ ngắn hạn (LSTM). LSTM là một loại RNN cực kỳ hữu ích và trên thực tế là những gì chúng ta thường sử dụng khi triển khai RNN. LSTM là một hàm có thể phân biệt lấy đầu vào và trạng thái và tạo ra đầu ra và trạng thái mới.

Trong ví dụ chúng ta sẽ sử dụng LSTM để phân loại trình tự. Trong các cách tiếp cận NLP truyền thống, các từ được xác định với cơ sở tiêu chuẩn là không gian chiều cao: Từ đầu tiên là (1, 0, 0, ...), từ thứ hai là (0, 1, 0, ...). Mỗi từ là trực giao với tất cả những từ khác. Trong ngôn ngữ thực, một số từ rất giống nhau (chúng ta gọi chúng là từ đồng nghĩa) hoặc chúng hoạt động theo những cách tương tự (ví dụ: Paris, Seattle, Tokyo). Quan sát chính là các từ xuất hiện trong các ngữ cảnh tương tự phải giống nhau.

Chúng ta có thể để một mạng “neural” sắp xếp những chi tiết này bằng cách buộc mỗi từ được biểu diễn bằng một vector ngắn đã học. Sau đó, để mạng có thể làm tốt nhiệm vụ của nó, nó phải học cách ánh xạ các từ với các vector này một cách hiệu quả. Ví dụ: vector đại diện cho từ “mèo”, theo một nghĩa nào đó, có thể gần với vector đại diện cho “chó”. Trong nhiệm vụ chúng ta sẽ học những cách nhúng từ này đầu tiên. Tuy nhiên, cũng có thể khởi tạo bằng cách nhúng từ được tính toán trước chẳng hạn như “GloVe” đã được đào tạo trên kho ngữ liệu chứa hàng tỷ từ.

Bây giờ chúng ta đã quyết định về cách biểu diễn từ của mình và loại mạng “neural” lặp lại mà chúng ta muốn sử dụng, hãy xác định mạng mà chúng ta sẽ sử dụng để phân loại trình tự. Chúng ta có thể coi mạng như thêm một loạt các lớp:

1. Lớp nhúng (các từ riêng lẻ trong mỗi chuỗi trở thành vector)
2. Lớp LSTM (cho phép mỗi từ phụ thuộc vào các từ trước đó)
3. Lớp Softmax (một tập hợp bổ sung các tham số và xác suất đầu ra cho mỗi lớp)

Chúng ta hãy xem xét một số phức tạp của định nghĩa mạng ở trên. Như thường lệ, đầu tiên chúng ta thiết lập các thông số cho mô hình. Trong trường hợp này, chúng ta có một từ vựng (thứ nguyên đầu vào) là 2000, LSTM ẩn và kích thước ô là 25, một lớp nhúng có thứ nguyên 50 và chúng ta có 5 lớp khả thi cho chuỗi. Như trước đây, chúng ta xác định hai biến đầu vào: một cho các tính năng và cho các nhãn. Sau đó, chúng ta khởi tạo mô hình của mình. Đây là một hàm đơn giản tìm kiếm đầu vào của chúng ta trong một ma trận nhúng và trả về biểu diễn nhúng, đưa đầu vào đó qua lớp mạng “neural” lặp lại LSTM và trả về đầu ra có kích thước cố định từ LSTM bằng cách chọn trạng thái ẩn cuối cùng của LSTM.

Trong việc triển khai LSTM, đây là đầu ra thực tế trong khi đầu ra thứ hai là trạng thái của LSTM. Bây giờ chúng ta chỉ cần thiết lập các nút tiêu chí của mình (chẳng hạn như cách chúng ta phân loại các nhãn bằng cách sử dụng vector suy nghĩ) và vòng lặp đào tạo. Trong ví dụ trên, chúng ta sử dụng kích thước minibatch là 200 và sử dụng SGD cơ bản với các tham số mặc định và tỷ lệ học tập nhỏ là 0,0005. Điều này dẫn đến một mô hình hiện đại mạnh mẽ để phân loại trình tự có thể mở rộng với lượng dữ liệu đào tạo khổng lồ.

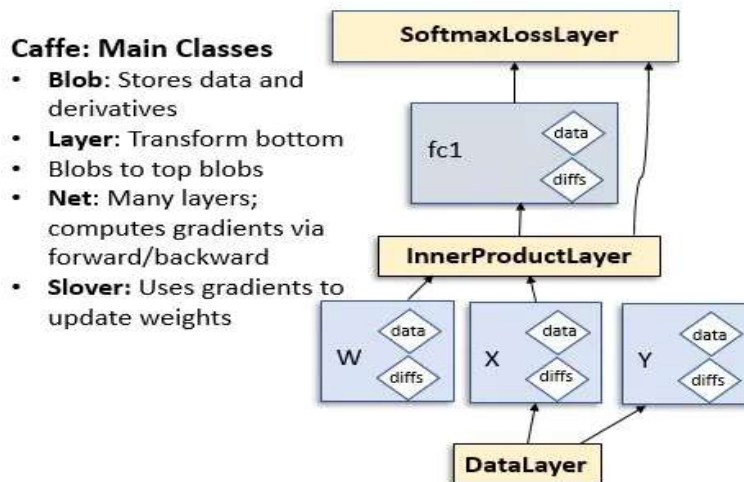
Lưu ý rằng khi kích thước dữ liệu đào tạo của bạn tăng lên, bạn nên cung cấp nhiều dung lượng hơn cho LSTM bằng cách tăng số thứ nguyên ẩn. Hơn nữa, người ta có thể có được một mạng phức tạp hơn bằng cách xếp chồng các lớp LSTM. Về mặt khái niệm, xếp chồng các lớp LSTM tương tự như xếp chồng

các lớp trong một mạng chuyên tiếp. Tuy nhiên, việc lựa chọn một kiến trúc tốt là rất cụ thể.

#### 4.4. Caffe

Caffe là một mô hình học máy mã nguồn mở được nghiên cứu và phát triển bởi Berkley AI Research. Nó rất dễ triển khai, dạng modul và tốc độ xử lý nhanh. Nó có sẵn các tài liệu mã nguồn mở phong phú trên Github. Nó được sử dụng rộng rãi trong các dự án nghiên cứu học thuật, trong startup's proof of concepts, Computer Vision, Natural Language Processing và nhiều lĩnh vực khác.

Là viết tắt của Convolutions Architecture for Fast Feature Embedding và được viết bằng thư viện C++ được BSD cấp phép với các liên kết Python và MATLAB. Nó được sử dụng để đào tạo và triển khai các mạng nơ-ron tích chập có mục đích chung một cách hiệu quả trên các kiến trúc hàng hóa. Kiến trúc của khuôn khổ trên được chia thành nhiều phần như sau:



HÌNH 25. KIẾN TRÚC CỦA CAFFE

##### 4.4.1. Cơ chế lưu trữ dữ liệu

Nó sử dụng dữ liệu mảng N chiều theo kiểu tiếp giáp C được gọi là các Blob để lưu trữ và giao tiếp dữ liệu. Blob có thể được coi là một lớp trừu tượng giữa CPU và GPU. Dữ liệu từ CPU được tải vào blob, sau đó được chuyển đến GPU để tính toán. Trong quá trình này, blob sử dụng lớp SyncedMem để đồng bộ hóa các giá trị giữa CPU và GPU.

Sau đó, Blob được chuyển sang lớp tiếp theo mà không tính đến các chi tiết triển khai thấp hơn mà vẫn duy trì được mức hiệu suất cao. Để sử dụng bộ nhớ hiệu quả, kỹ thuật phân bổ theo yêu cầu lười biếng được sử dụng để cấp phát bộ nhớ theo yêu cầu cho máy chủ và thiết bị. Đối với dữ liệu quy mô lớn, cơ sở dữ liệu được sử dụng là LevelDB. Mô hình học sâu được lưu trữ vào bộ nhớ phụ dưới dạng Bộ đệm giao thức của Google cung cấp tuần tự hóa hiệu quả, định dạng văn bản có thể đọc được, v.v.

#### 4.4.2. Cơ chế lớp

Các blob được chuyển làm đầu vào cho mô hình và đầu ra tương ứng được tạo ra. Nó theo sau một mối quan hệ nhiều-nhiều. Nó có các trách nhiệm chính như một phần của hoạt động mô hình sau đây:

- **Thiết lập:** Nó khởi tạo lớp và các kết nối bên dưới lần đầu tiên trong quá trình khởi tạo mô hình.
- **Forward Pass:** Các đầu vào được chuyển và các đầu ra tương ứng được tạo ra.
- **Backward Pass:** Tính toán độ dốc liên quan đến đầu ra, siêu tham số mô hình và đầu vào sau đó được chuyển đến các lớp tiếp theo bằng cách sử dụng một kỹ thuật được gọi là lan truyền ngược.

Nó cung cấp các thiết lập lớp khác nhau như Convolution, Pooling, các kích hoạt phi tuyến như chỉnh lưu, đơn vị tuyến tính (ReLU) với các tổn thất tối ưu hóa được sử dụng rộng rãi như Log Loss, R-squared, SoftMax, v.v. Các lớp có thể được mở rộng thành triển khai lớp người dùng tùy chỉnh mới bằng cách sử dụng cấu tạo thành phần của mạng.

#### 4.4.3. Hệ thống và mô hình chạy cơ bản

Nó sử dụng cấu trúc dữ liệu được gọi là đồ thị xoay chiều có hướng để lưu trữ các hoạt động được thực hiện bởi các lớp bên dưới, do đó đảm bảo tính đúng đắn của chuyển tiếp và chuyển tiếp ngược. Mạng mô hình Caffe điển hình bắt đầu với lớp dữ liệu tải dữ liệu từ đĩa và kết thúc bằng lớp dữ liệu trống dựa trên các yêu cầu của ứng dụng. Nó có thể được chạy trên CPU / GPU và việc chuyển đổi giữa chúng là liền mạch và không phụ thuộc vào kiểu máy.

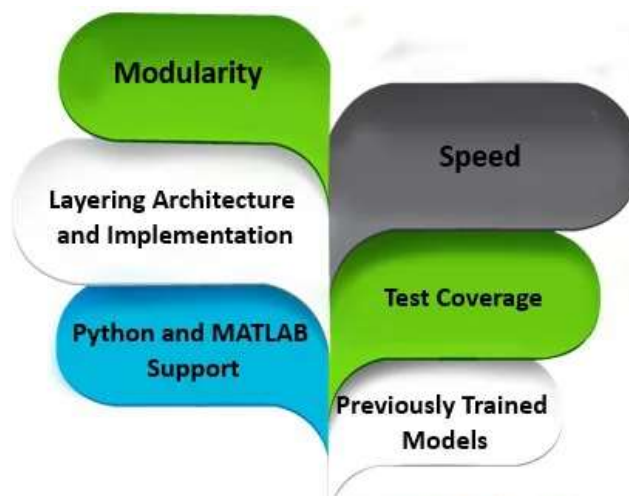
#### 4.4.4. Huấn luyện một mạng lưới

Một mô hình Caffe điển hình được đào tạo bởi một thuật toán giảm độ dốc ngẫu nhiên tiêu chuẩn và nhanh chóng. Dữ liệu có thể được xử lý thành các lô nhỏ truyền trong mạng một cách tuần tự. Các thông số quan trọng liên quan đến việc đào tạo như lịch trình giảm tốc độ học tập, động lượng và các điểm kiểm tra để dừng và tiếp tục được thực hiện tốt với tài liệu kỹ lưỡng.

Nó cũng hỗ trợ tinh chỉnh, một kỹ thuật trong đó một mô hình hiện có có thể được sử dụng để hỗ trợ kiến trúc hoặc dữ liệu mới. Các trọng số của mô hình trước đó được cập nhật cho ứng dụng mới và các trọng số mới được chỉ định ở bất cứ nơi nào cần thiết. Kỹ thuật này được sử dụng rộng rãi trong nhiều ứng dụng học sâu trong thế giới thực.

#### 4.4.5. Ưu điểm của mô hình học sâu Caffe

Nó cung cấp một bộ hoàn chỉnh các gói để đào tạo, kiểm tra, tinh chỉnh và triển khai mô hình. Nó cung cấp nhiều ví dụ về các nhiệm vụ trên. Trước đây, nó được sử dụng cho các nhiệm vụ thị giác nhưng giờ đây nó đã được người dùng sử dụng cho các ứng dụng học sâu khác như nhận dạng giọng nói, mạng nơ-ron, robot. Nó có thể được chạy trong các nền tảng dựa trên đám mây với chuyển đổi nền tảng liền mạch.



HÌNH 26. ƯU ĐIỂM CỦA MÔ HÌNH CAFFE

**Tính mô-đun:** Có thể mở rộng dữ liệu mới, các lớp và các chức năng tối ưu hóa tồn thất. Nó có các ví dụ tham khảo có các lớp và chức năng mất mát được triển khai.

**Tốc độ:** Nó có thể được sử dụng để xử lý 60 triệu hình ảnh mỗi ngày bằng GPU NVIDIA CUDA K40. Đây là một trong những cách triển khai mạng chuyển đổi nhanh nhất hiện có trên thị trường.

**Kiến trúc phân lớp và triển khai:** Định nghĩa của mô hình được viết bằng Ngôn ngữ đệm giao thức dưới dạng tệp cấu hình. Kiến trúc mạng sử dụng cách tiếp cận đồ thị xoay chiều có hướng. Khi mô hình được khởi tạo, nó sẽ dự trữ bộ nhớ chính xác theo yêu cầu của mô hình. Chuyển từ môi trường dựa trên CPU sang môi trường GPU yêu cầu một lệnh gọi chức năng duy nhất.

**Phạm vi kiểm tra:** Mọi mô-đun trong đó đều được thử nghiệm và dự án mã nguồn mở của nó không cho phép bất kỳ cam kết mô-đun nào mà không có các thử nghiệm tương ứng, do đó cho phép cải tiến nhanh chóng và cấu trúc lại codebase. Do đó, điều này làm tăng khả năng bảo trì của nó, tương đối không có lỗi / lỗi.

**Hỗ trợ Python và MATLAB trong phân lớp:** Cung cấp giao diện và dễ sử dụng với khung nghiên cứu hiện có được các tổ chức khoa học sử dụng. Cả hai ngôn ngữ đều có thể được sử dụng để xây dựng mạng và phân loại đầu vào. Python in Layering cũng cho phép sử dụng mô-đun bộ giải để phát triển các kỹ thuật đào tạo mới và dễ sử dụng.

**Các mô hình được đào tạo trước đây được sử dụng làm tài liệu tham khảo:** Nó cung cấp các mô hình tham chiếu cho các dự án nghiên cứu và khoa học như các mô hình được đào tạo của ImageNet mang tính bước ngoặt, v.v. Do đó, nó cung cấp một thành phần phần mềm chung có thể được mở rộng để đạt được tiến độ nhanh chóng trong việc phát triển kiến trúc mô hình cho các ứng dụng trong thế giới thực.

#### 4.5. Torch

Là thư viện mã nguồn mở học máy và ngôn ngữ kịch bản dựa trên ngôn ngữ Lua. Cung cấp các thuật toán cho học sâu (deep learning). Torch đã không còn được phát triển, tuy nhiên PyTorch (framework mã nguồn mở phổ biến được phát triển bởi Meta) dựa vào thư viện của Torch.

Các tính năng chính:



- Xử lý mảng N-chiều mạnh mẽ.
- Cung cấp nhiều quy trình để lập chỉ mục, cắt, chuyển vị,...
- Có thể giao tiếp với C, thông qua LuaJIT
- Các hàm đại số tuyến tính.
- mạng nơ-ron và các mô hình năng lượng (Energy-based models - EBM).
- Các hàm tối ưu hóa số liệu.
- Hỗ trợ GPU mạnh và hiệu quả.
- Có thể nhúng, với các công đến phân phụ trợ iOS và Android.

Mục tiêu của Torch là có được sự linh hoạt và tốc độ tối đa trong việc xây dựng các thuật toán khoa học, đồng thời làm cho quá trình trở nên cực kỳ đơn giản. Torch đi kèm với một hệ sinh thái lớn gồm các gói hướng tới cộng đồng về học máy, thị giác máy tính, xử lý tín hiệu, xử lý song song, hình ảnh, video, âm thanh và được xây dựng dựa trên cộng đồng Lua.

Trọng tâm của Torch là các thư viện tối ưu hóa và mạng nơ-ron phổ biến, dễ sử dụng, đồng thời có tính linh hoạt tối đa trong việc triển khai các cấu trúc liên kết mạng nơ-ron phức tạp. Bạn có thể xây dựng đồ thị tùy ý của mạng nơ-ron và song song hóa chúng qua CPU và GPU một cách hiệu quả.

## 5. Mô hình học máy

Sau khi có được file data.csv – file dữ liệu lưu các đặc trưng được trích chọn từ trước, người dùng có thể đưa vào các mô hình học máy để tạo thành classifier phục vụ cho việc predict – phỏng đoán.

Các bước tạo classifier bao gồm:

- Đọc dữ liệu từ file CSV, cột phỏng đoán sẽ là cột cuối lưu thông tin về dạng mã.
- Tạo model được lấy từ việc đưa dữ liệu đặc trưng được chọn.
- Giảm các đặc trưng từ các đặc trưng được chọn.
- Chia ma trận có được thành các tập con ngẫu nhiên để đưa vào mô hình.
- Đưa vào mô hình Machine Learning.

Hiện tại chương trình sử dụng 7 thuật toán Machine Learning mặc định của gói sklearn là: DecisionTree, RandomForest, Adaboost, GradientBoosting, GNB, LinearRegression và XGBoost.

```
model = {"DecisionTree": tree.DecisionTreeClassifier(max_depth=10),
        "RandomForest": ek.RandomForestClassifier(n_estimators=50),
        "Adaboost": ek.AdaBoostClassifier(n_estimators=50),
        "GradientBoosting": ek.GradientBoostingClassifier(n_estimators=50),
        "GNB": GaussianNB(),
        "LinearRegression": LinearRegression(),
        "XGBRegressor": xgb.XGBRegressor(objective="reg:linear", random_state=42)
}
```

HÌNH 27. CÁC MODEL HỌC MÁY SỬ DỤNG TRONG SẢN PHẨM

Sau khi đưa dữ liệu qua mô hình học máy, sản phẩm sẽ tự động chọn ra mô hình đạt độ chính xác cao nhất trong bảy mô hình bên trên và lưu kết quả vào file classifier phục vụ cho việc phỏng đoán.

Việc phỏng đoán là bài toán đọc classifier rồi đưa ra predict dựa theo nhãn đã được gán từ trước.

## 6. Mô hình học sâu

Sau các phiên bản và nhiều lần thử nghiệm, nhóm nhận thấy việc sử dụng mô hình Machine Learning (học máy) như trên đạt độ chính xác chưa được cao, việc áp dụng học máy chỉ phù hợp cho bài toán phát hiện và nhận biết mã sạch – mã độc nhưng chưa phù hợp cho bài toán gán nhãn cho mã độc.

Do đó, khai thác các điểm mạnh của học sâu và mô hình CNN, ở phiên bản mới nhất, sản phẩm thay thế tất cả thuật toán học máy thành mô hình học sâu CNN.

### 6.1. Tiền xử lý dữ liệu

Sau khi lấy được đặc trưng của các file PE và xuất ra file CSV, chương trình sẽ tiến hành resize lại dữ liệu và thay giới hạn độ lớn cho dữ liệu, sau đó sẽ bỏ tên cột và cuối cùng sẽ lưu thành 2 file.

Đoạn code xử lý dữ liệu:

```
def TienXuLyFileTrain():
    File_train = "data.csv"

    DeepLearningFiletrain = pd.read_csv(File_train, sep = "|")
    DeepLearningFiletrain = DeepLearningFiletrain.drop(['FileName', 'MD5'], axis=1)

    DeepLearningFiletrain.to_csv("DeepLearningFiletrain.csv", index = False)

    DeepLearningFiletrain = pd.read_csv("DeepLearningFiletrain.csv", skiprows=1, header=None)

    row_count, column_count = DeepLearningFiletrain.shape

    #dong bo hoa du lieu
    for My_row in range(row_count):
        for My_column in range(column_count):
            if((DeepLearningFiletrain[My_column][My_row]) >= 256):
                DeepLearningFiletrain[My_column][My_row] = 255

    #tao du lieu moi
    DeepLearningFiletrain.to_csv("DeepLearningFiletrain.csv", index = False)
```

HÌNH 28. CODE TIỀN XỬ LÝ DỮ LIỆU

Kết quả của quá trình tiền xử lý dữ liệu, người dùng sẽ thu được file DeepLearningFiletrain.csv.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
255	144	3	0	4	0	255	0	184	0	0	0	64	0	0	0	128	255
255	144	3	0	4	0	255	0	184	0	0	0	64	0	0	0	128	255
255	80	2	0	4	15	255	0	184	0	0	0	64	26	0	0	255	255
255	144	3	0	4	0	255	0	184	0	0	0	64	0	0	0	255	255
255	144	3	0	4	0	255	0	184	0	0	0	64	0	0	0	128	255
255	144	3	0	4	0	255	0	184	0	0	0	64	0	0	0	232	255
255	144	3	0	4	0	255	0	184	0	0	0	64	0	0	0	216	255
255	144	3	0	4	0	255	0	184	0	0	0	64	0	0	0	248	255
255	144	3	0	4	0	255	0	184	0	0	0	64	0	0	0	224	255
255	144	3	0	4	0	255	0	184	0	0	0	64	0	0	0	128	255
255	144	3	0	4	0	255	0	184	0	0	0	64	0	0	0	128	255
255	144	3	0	4	0	255	0	184	0	0	0	64	0	0	0	240	255
255	144	3	0	4	0	255	0	184	0	0	0	64	0	0	0	255	255
255	144	3	0	4	0	255	0	184	0	0	0	64	0	0	0	128	255
255	144	3	0	4	0	255	0	184	0	0	0	64	0	0	0	128	255
255	144	3	0	4	0	255	0	184	0	0	0	64	0	0	0	128	255
255	144	3	0	4	0	255	0	184	0	0	0	64	0	0	0	128	255

HÌNH 29. DỮ LIỆU TIỀN XỬ LÝ ĐƯỢC LƯU TRONG DEELLEARNINGFILETRAIN.CSV

## 6.2. Đưa dữ liệu vào mô hình

Dữ liệu sẽ được lưu dưới dạng vector Nx1 trước khi được đưa vào mô hình CNN.

Các tham số:

- Số các convolution layer: Sử dụng 3 convolution layer 2D, do dữ liệu đầu vào là dạng ma trận 2D Nx1 => Không cần padding dữ liệu.
- Filter size – Kernel size = 2x2, giúp tối ưu được tốc độ trong quá trình train dữ liệu.
- Pooling size: thường là 1x2 để đồng bộ dữ liệu đầu vào.

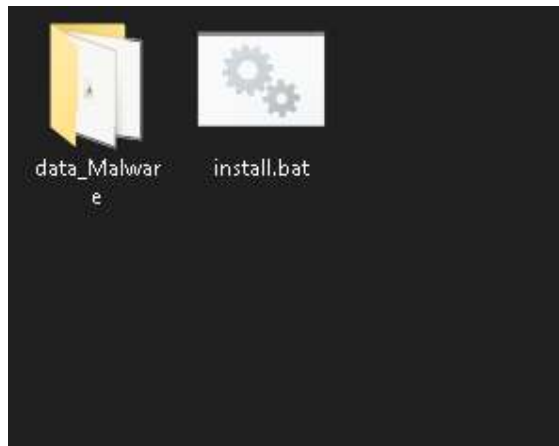
Quá trình train sẽ trải qua 30 epoch với kích thước của batch = 32 (số file trong mỗi lần lấy sample).

## CHƯƠNG III - TRIỂN KHAI THỰC NGHIỆM

### 1. Yêu cầu cài đặt

Do sản phẩm được viết 100% bằng ngôn ngữ python, và có sử dụng các module từ bên ngoài, nên máy tính yêu cầu phải có cài đặt python 3.x trở lên, và gói pip để cài đặt các module.

Khi tải về, người dùng sẽ có được 2 file là tệp chứa dữ liệu – code và file batch cài đặt.

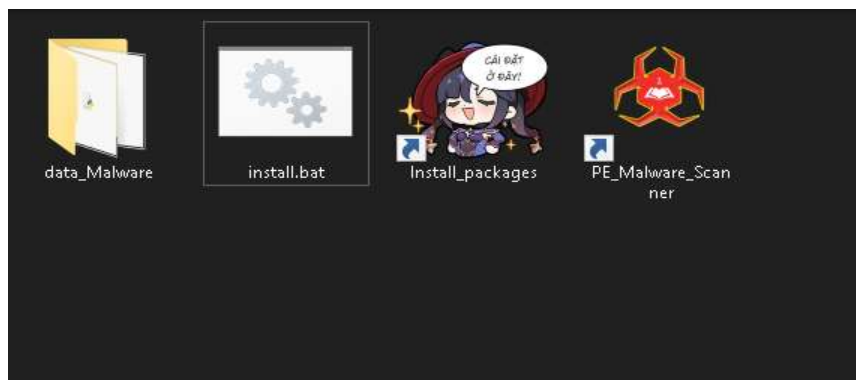


HÌNH 30. GÓI SẢN PHẨM KHI TẢI VỀ

Cài đặt chương trình bằng cách nhấp chạy file install.bat bên trên.

File install.bat đơn giản là đoạn script tạo shortcut đến file thực thi trong folder data\_Malware. (lưu ý, file này chạy script VBS, nên không được để trong folder có đường dẫn chứa dấu tiếng Việt, nếu không sẽ bị lỗi).

Sau khi chạy người dùng sẽ có được 2 file shortcut mới có dạng như sau:



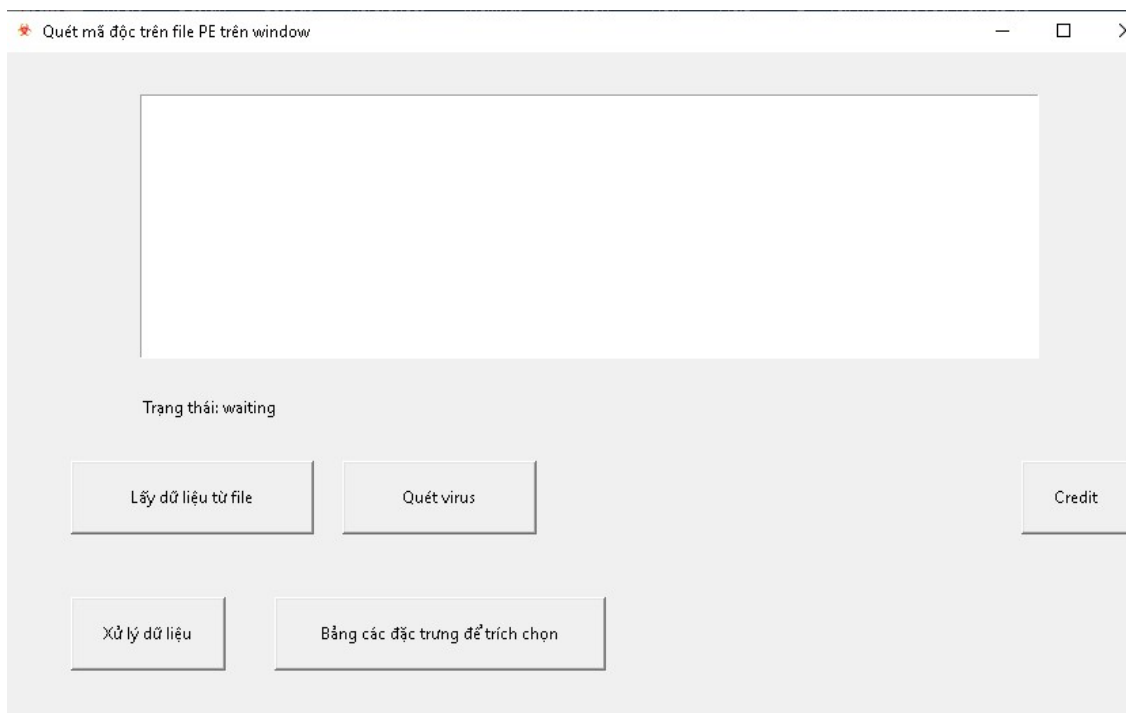
HÌNH 31. GÓI SẢN PHẨM SAU KHI CÀI ĐẶT

Chạy file install\_packages để tự động cài đặt các module python sẽ sử dụng trong chương trình.

Khi người dùng đã cài đặt đầy đủ các module, thì có thể bắt đầu sử dụng chương trình.

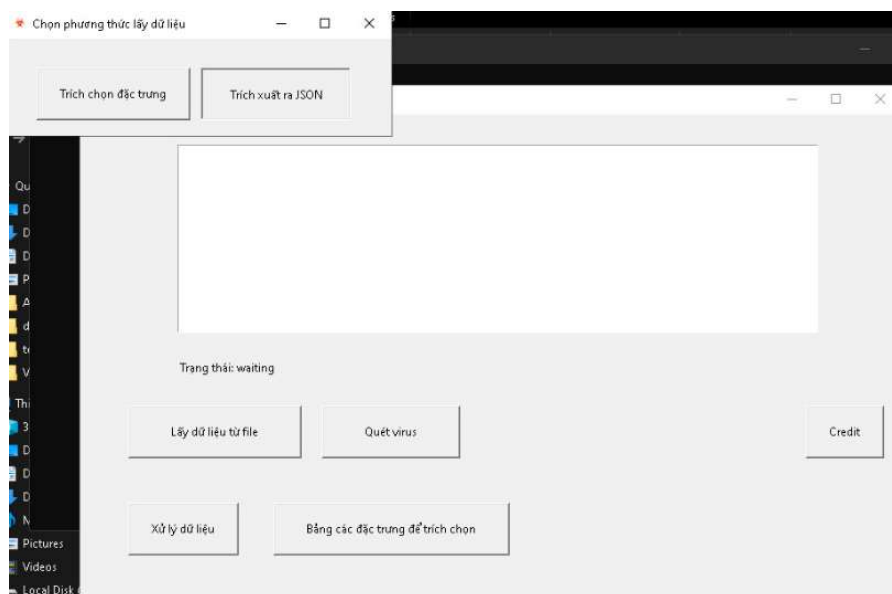
## 2. Triển khai cách sử dụng sản phẩm

Sau khi đã cài đặt đầy đủ các gói, người dùng có thể bắt đầu sử dụng chương trình.



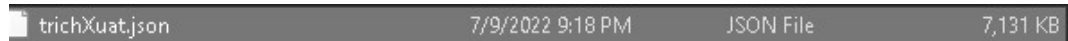
HÌNH 32. GIAO DIỆN CỦA SẢN PHẨM

B1: Lấy dữ liệu bằng cách chọn folder chứa các file và lưu vào trong file JSON



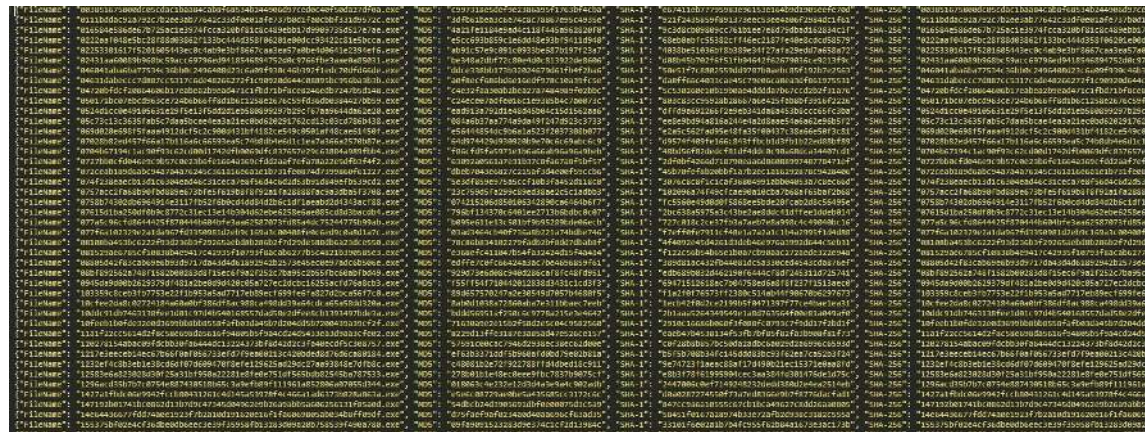
HÌNH 33. CÁCH LẤY ĐẶC TRƯNG TỪ FILE

File JSON sẽ được lưu mặc định bằng tên tríchXuất.json



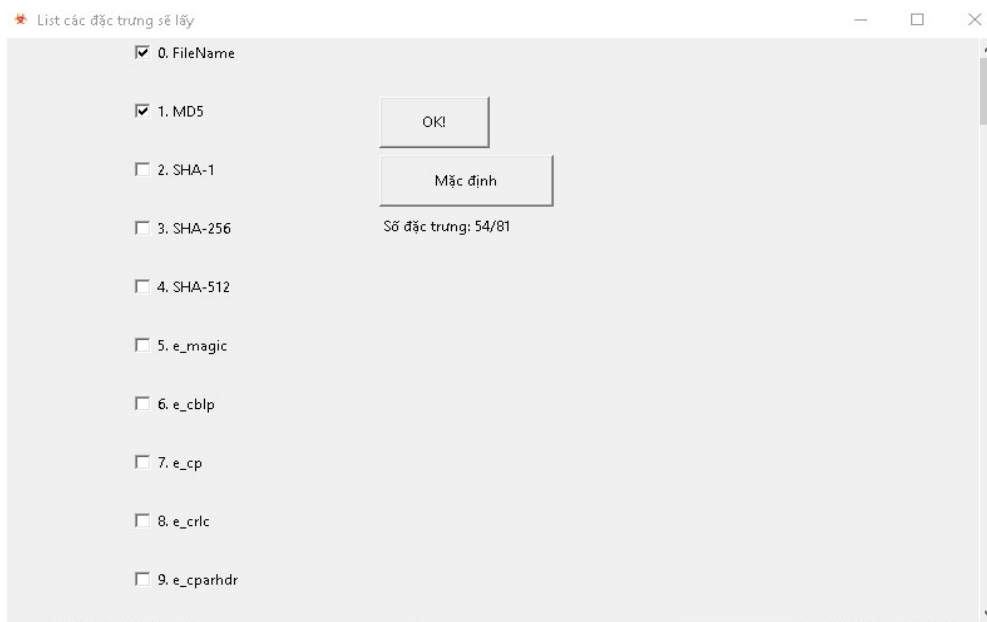
HÌNH 34. FILE JSON ĐƯỢC TẠO

Trong đây sẽ lưu tất cả đặc trưng lấy được.



HÌNH 35. LIST CÁC ĐẶC TRƯNG TRONG FILE JSON

B2: Sau khi có được file JSON, người dùng có thể mở bảng để trích chọn các đặc trưng.



HÌNH 36. BẢNG TỰY CHỌN CÁC ĐẶC TRƯNG

Sau đó có thể chọn “Lấy dữ liệu từ file”, sau đó chọn “Trích chọn đặc trưng” Chọn file JSON vừa tạo => Có được file data.csv.

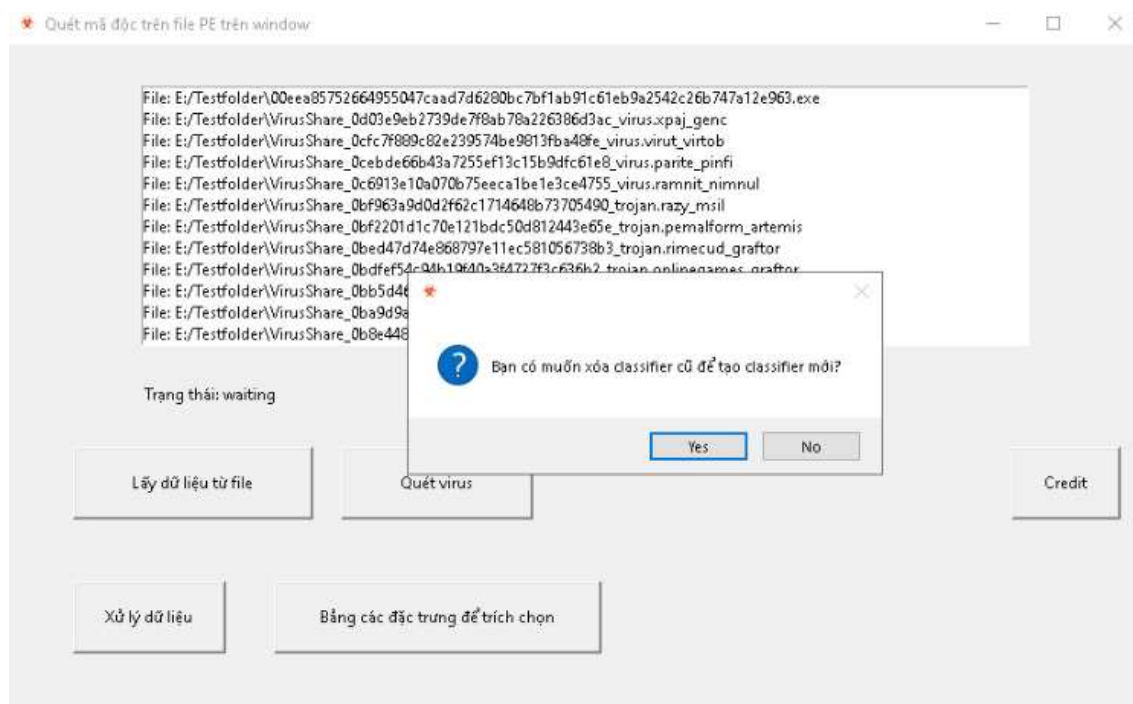
Trong file csv này chứa các dữ liệu được trích chọn từ file JSON với các đặc trưng lấy từ bảng bên trên.



FileName	MD5	e_magic	e_cblp	e_cp	e_crlc	e_cparhdr	e_minalloc	e_maxalloc	e_ss	e_sp
003851675800dc05cdac1baa84cab8f8534b244906d97ced0c40f50da27df0a.exe	c997318e5def9e2386a95f1763bf4cba	23117	144	3	0	4	0	65535	0	18
0111bdddac92a73c7b2ee3ab77642c33df0e01afe737b0d1fa0cbf331d9572c.exe	3dfb61bea3c6e74c8c78867e95c4935e	23117	144	3	0	4	0	65535	0	18
016584e586e67b725c1e3974fcca20bf81c6c489ebb17d9909735d1e7a7ae.exe	4a21fe11049e9ad4c118f445a9e2820f0	23117	80	2	0	4	15	65535	0	18
0222aaf048e5bc28f98d03862f133bc44d358f06201e00dc9342c01e5bcca.exe	e5cc93b959c1e6dd48e93bf9411d940	23117	144	3	0	4	0	65535	0	18
022533016175201605443ec0c4ab9e3bf667caa3ea57a0be4d0641e2394ef6.exe	ab91c57e9c091c0933be687b197f23a7	23117	144	3	0	4	0	65535	0	18
02431aa00089b968bc59acc69796e9418546894752d0c9766f6e3aae0a85031.exe	be348a2dbf72c80e4d0c813922de8606	23117	144	3	0	4	0	65535	0	18
046041aba6ba77534c36bb0c2496408d23c6a09f930c46b392f1edc70dfd66de.exe	ddce388bb173b32024679d61fb4f2ba6	23117	144	3	0	4	0	65535	0	18
04631dabec07d887c5317c6de482667271c90920d644c08895bc956ba3b3b.exe	a0f8ecf686d6e16edf978c10a39fc5e	23117	144	3	0	4	0	65535	0	18
04720bfdd20864606b17eabea2b9ead471c1fbd71bf8ce8246edb7247b5d148.exe	c4e32faa30ab2bea27a7484989f02bbc	23117	144	3	0	4	0	65535	0	18
050171bc07ebcd963ce724b6b6ff8d1b6c1258e267c59fd36db0384427bb59.exe	c24ece07adfe016c1e9385b4c7a0073c	23117	144	3	0	4	0	65535	0	18
0524d1cc0e49105631e29f5e13f5dd2d1eb580899297b29df67aa9644da62e20.exe	ead913a792d1e48d45b04c15d1562aa6	23117	144	3	0	4	0	65535	0	18
05c73c13c3635fab5c7daa5bcae4ea3a21ec0bd620291762cd13c03cd786b438.exe	084a6b37aa77a9da49f247d923c3739	23117	144	3	0	4	0	65535	0	18
069d020e698f5faaa4912dcf5c2c900d431bf4182ce549c501af48cae61450f.exe	e56444854dc9b6a1a523f2037308b077	23117	144	3	0	4	0	65535	0	18
07028b82ed457f66a17b116a6c65593ea5c74b8db4e6d1c1ea7a366a2570b87e.exe	64d974429d930920b9e70c6c69abc6c9	23117	144	3	0	4	0	65535	0	18
0704b6719d4a9d9f93cc7rdm01727f8f006c9f8e37c57e74c6388da989fhd.exe	f8c4f45fa5971e11h5a6c6a9a6a9a6a9	23117	144	3	0	4	0	65535	0	18

HÌNH 37. DỮ LIỆU ĐƯỢC TRÍCH CHỌN TRONG FILE CSV

B3: Sau khi có được file data.csv, người dùng có thể đưa nguồn dữ liệu này vào mô hình học máy để tiến hành tạo classifier bằng cách chọn nút Xử lý dữ liệu/hoặc mô hình học sâu nếu sử dụng Version 5.



HÌNH 38. TẠO CLASSIFIER MỚI

Sau đó classifier sẽ được tạo, đây chính là file phục vụ cho việc phỏng đoán. (Hình 3.10)

classifier.pkl	7/10/2022 9:58 AM	PKL File	3,221 KB
features.pkl	7/10/2022 9:58 AM	PKL File	1 KB

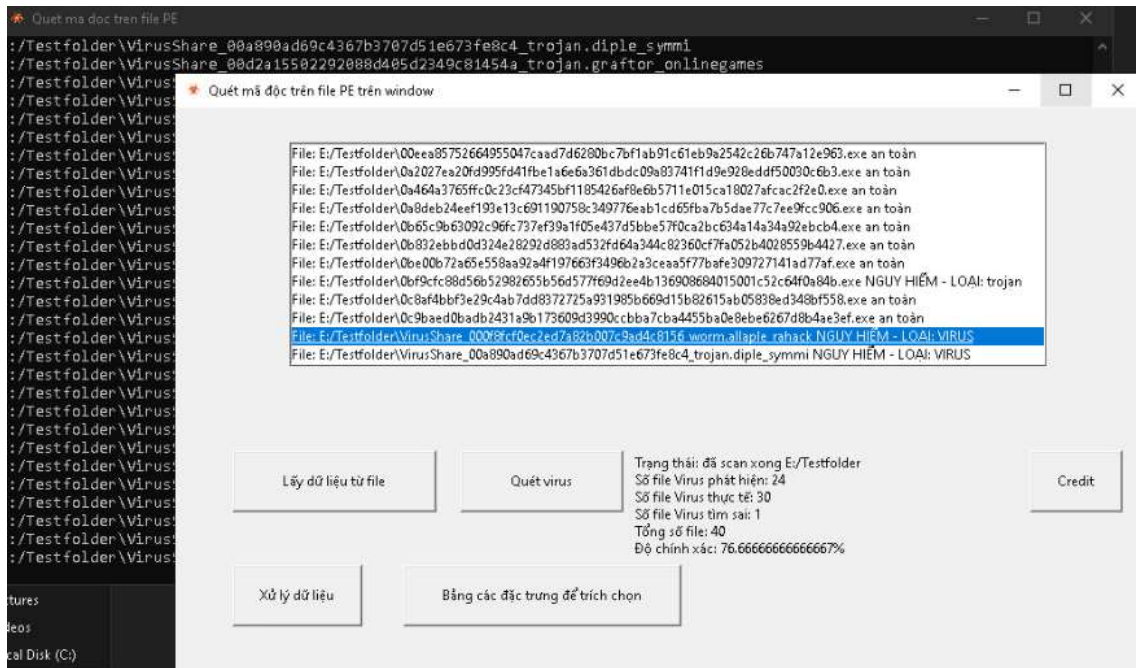
HÌNH 39. CLASSIFIER MỚI ĐÃ ĐƯỢC TẠO THÀNH CÔNG

Hoặc file .h5 nếu chạy bằng mô hình học sâu CNN (Hình 3.11)

HÌNH 40. FILE .H5 SAU KHI DỮ LIỆU ĐƯỢC ĐƯA QUA MÔ HÌNH HỌC SÂU

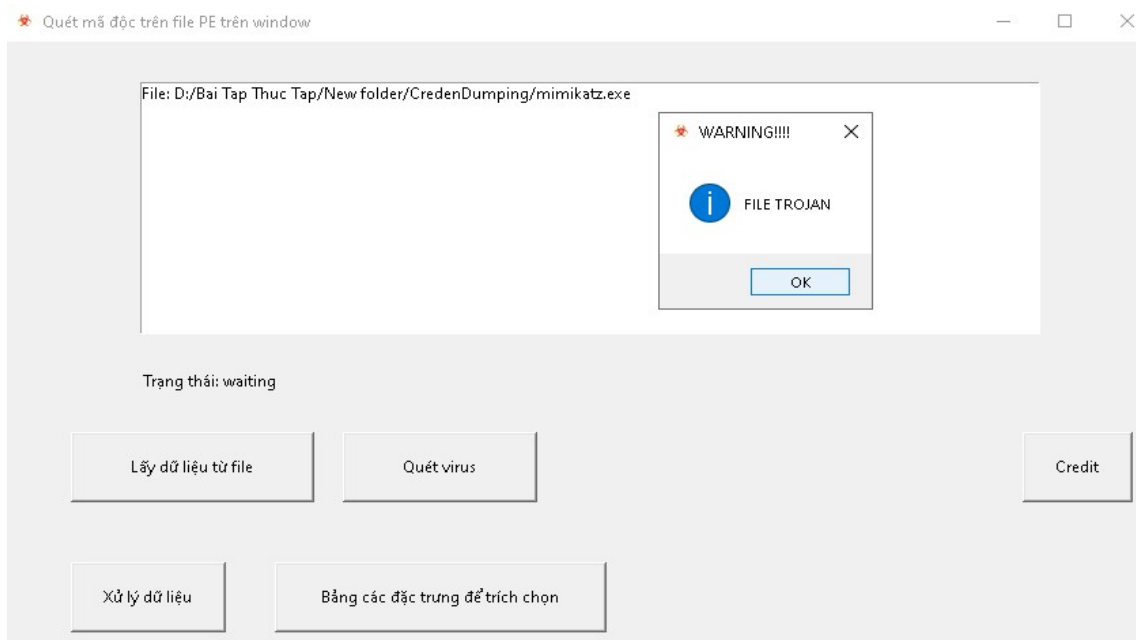
B4: Sau khi có được classifier, người dùng có thể sử dụng chức năng quét Virus.

Có thể quét trực tiếp 1 folder (Hình 3.12)



HÌNH 41. QUÉT TRÊN 1 FOLDER

Hoặc quét 1 file bất kì. (Hình 3.13)



HÌNH 42. QUÉT TRÊN 1 FILE



## CHƯƠNG IV - KẾT LUẬN

Hiện nay, công nghệ mã độc đang ngày càng phát triển với các hình thức tinh vi, gây ảnh hưởng rất lớn tới nền an ninh thông tin. Việc áp dụng công nghệ học máy – học sâu và trí tuệ nhân tạo trong bài toán phát hiện mã độc đang là xu hướng phát triển ở hiện tại và tương lai nhờ vào độ chính xác và tốc độ cao. Trong suốt quá trình thực hiện đề tài: “Xây dựng, thử nghiệm ứng dụng phát hiện mã độc”, cả nhóm đã hoàn thành được các nội dung sau:

- Về mặt lý thuyết:
  - Giới thiệu về mã độc PE trên window: đặc điểm, phân loại, một số cách phát hiện trên máy tính.
  - Giới thiệu về các mô hình học máy học sâu.
  - Tìm hiểu, phân tích và đưa ra giải pháp áp dụng học máy học sâu trong bài toán phát hiện mã độc.
- Về mặt thực nghiệm:
  - Xây dựng phần mềm giao diện người dùng có thể tự tạo mô hình và classifier, phát hiện mã độc một cách dễ dàng để sử dụng.
  - Chương trình có thể dễ dàng cập nhật lại mô hình để có các bước phát triển hơn trong tương lai.

**Hạn chế:** Ở thời điểm hiện tại, chương trình mới chỉ phát hiện mã độc trên file PE thông thường, chưa phát hiện được các loại mã độc dựa trên các kỹ thuật giấu tin hay các dạng Fileless Malware, và do nguồn mã độc chưa được nhiều, nên việc phát hiện nhãn của mã độc vẫn chưa đạt được độ chính xác cao.

**Hướng phát triển trong tương lai:** Cải thiện lại toàn bộ hệ thống, giúp hệ thống có thể phân tích và phát hiện tất cả các loại file trên window. Tích hợp thêm các mô hình học máy và các chức năng tự động hóa cho ứng dụng vào các mô hình cloud. Ngoài ra, còn tối ưu hóa code để ứng dụng hoạt động nhanh hơn.

Do thời gian và kỹ năng lập trình có hạn, nên sản phẩm không thể tránh khỏi các thiếu sót, nên em và cả nhóm mong nhận được sự góp ý của các thầy cô để sản phẩm có thể hoàn thiện hơn trong tương lai.

# TÀI LIỆU THAM KHẢO

- [1] Stamp, M., Alazab, M., & Shalaginov, A. (2020). *Malware Analysis Using Artificial Intelligence and Deep Learning* (1st ed. 2021 ed.). Springer.
- [2] Sikorski, M., & Honig, A. (2012). *Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software* (1st ed.). No Starch Press.
- [3] Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems* (2nd ed.). O'Reilly Media.
- [4] Lakshminarayanan, K., & Muthuswamy, S. (2020). *Malware Detection Techniques using Machine Learning Classifiers*. LAP LAMBERT Academic Publishing.
- [5] Kosmidis, K., Kalloniatis, C.: *Machine Learning and Images for Malware Detection and Classification* (2017).
- [6] Udayakumar, N., Saglani, V.J., Gupta, A.V., Subbulakshmi, T.: *Malware classification using machine learning algorithms*. In: 2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI), pp. 1–9. Tirunelveli (2018).
- [7] Gandotra, E., Bansal, D., Sofat, S.: *Malware analysis and classification: a survey*. J. Inf. Secur.
- [8] Vinod, P., Jaipur, R., Laxmi, V., Gaur, M.: *Survey on malware detection methods*. In: Proceedings of the 3rd Hackers' Workshop on Computer and Internet Security, pp. 74–79(2009).
- [9] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: *Scikit-learn: machine learning in Python*. J. Mach. Learn. Res.
- [10] E. Raff, J. Sylvester, and C. Nicholas, “*Learning the PE header, malware detection with minimal domain knowledge*” in Proc. 10th ACM Workshop Artif. Intell. Secur. New York, NY, USA: ACM, Nov. 2017, pp. 121–132.
- [11] Y. LeCun, Y. Bengio, and G. Hinton, “*Deep learning*” Nature, vol. 521, no. 7553, pp. 436–444, 2015.
- [12] H. S. Anderson and P. Roth. (2018). “*EMBER: An open dataset for training static PE malware machine learning models*.” <https://arxiv.org/abs/1804.04637>.
- [13] F. Chollet. (2015). Keras: Deep Learning Library for Theano and Tensorflow. [Online]. Available: <https://keras.io/k>.