Xây dựng mô hình học máy, học sâu nhận dạng mã độc Android

Đặng Thị Thúy Hồng*, Phòng Lai Bảo Minh[2] and Đặng Huỳnh Vĩnh Tân[1]

Ho Chi Minh City, University of Information Technology, Faculty of Information Science and Engineering

**Correspondence: email : 20520523@gm.uit.edu.vn, 20522217@gm.uit.edu.vn, 21520442@gm.uit.edu.vn*

Article info.

Received 27 May 2023

Keywords

Malware detection, Malware images, Machine Learning, Mobile application security

ABSTRACT

Gần đây, ngày càng nhiều các ứng dụng di động đã xuất hiện dẫn đến sự phát triển mạnh mẽ của các phần mềm độc hại dễ dàng xâm nhập vào thiết bị di động nhằm mục đích đánh cắp thông tin người dùng. Trong nghiên cứu này, chúng tôi đã tìm hiểu, xây dựng các mô hình học máy và học sâu để phát hiện mã độc trên bộ dữ liệu chúng tôi thu thập, tiền xử lý từ bộ dữ liệu có sẵn của đại học New Brunswick. Các mô hình của chúng tôi bước đầu đạt được kết quả khả quan, đây là tiền đề để chúng tôi phát triển ứng dụng nhận diện mã độc giúp người dùng Android thuận tiện hơn trong việc phòng chống sự xâm nhập của phần mềm độc hại. Trong tương lai, chúng tôi sẽ tiếp tục nâng cao chất lượng của bộ dữ liệu và cải thiện hiệu suất của mô hình.

1. INTRODUCTION

Trong cuộc cách mạng 4.0 đang diễn ra mạnh mẽ trên toàn Thế giới cũng như ở Việt Nam như hiện nay, điện thoại thông minh trở nên rất phổ biến và trở thành một phần không thể thiếu trong cuộc sống của con người. Đi đôi với sự phát triển đó, hệ điều hành Android, một hệ điều hành với mã nguồn mở, đã trở thành một trong những nền tảng điện thoại thông minh phổ biến nhất. Hệ điều hành Android đã phát triển trên cả điện thoại thông minh và máy tính bảng, với Android, người dùng có thể tận dụng tối đa các ứng dụng để phục vụ cho nhiều nhu cầu trong đời sống hàng ngày như giải quyết công việc, giải trí, học tập, liên lạc… Sự phát triển mạnh mẽ của Android là cơ hội cho tội phạm mạng tìm hiểu và phát triển các ứng dụng độc hại, từ đó số lượng mã độc trên Android ngày càng tăng nhằm khai thác thông tin người dùng vì lợi ích tiền tệ. Điều này là động lực để các phần mềm phát hiện và phòng chống mã độc trên Android cần được nâng cấp về phương pháp và kỹ thuật.

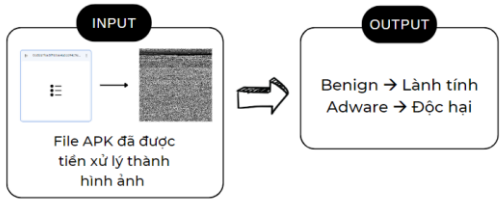
Hiện nay, rất nhiều nghiên cứu về việc phát hiện mã độc bằng các phương pháp như dựa trên chữ ký, phương pháp động, phương pháp tĩnh đã được đề xuất. Mặc dù có hiệu quả trong việc phát hiện, nhận diện mã độc nhưng vẫn còn gặp một số hạn chế. Vì vậy, trong nghiên cứu này, chúng tôi đề xuất một số mô hình học máy, học sâu để phát hiện phần mềm độc hại thông qua các tập tin APK một cách nhanh chóng.

2. DATASET

A. Mô tả bài toán

Mục tiêu của nghiên cứu này là xây dựng mô hình nhận diện mã độc file APK thông qua hình ảnh, đầu vào và đầu ra được mô tả như sau:

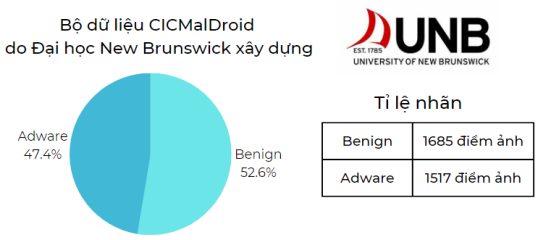
- **Input: file APK đã được tiền xử lý thành hình ảnh**
- **Output: một trong hai nhãn (độc hại hay lành tính).**



Hình 1: Mô tả Input và Output của bài toán.

B. Bộ dữ liệu

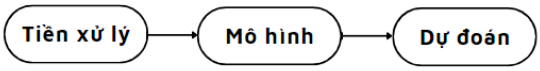
Chúng tôi sử dụng bộ dữ liệu CICMalDroid do Đại học New Brunswick xây dựng, sau đó chúng tôi tiến hành tiền xử lý bộ dữ liệu để phục vụ cho bài toán nhận dạng mã độc. Bộ dữ liệu sau khi xử lý của chúng tôi bao gồm hai nhãn là Adware và Benign với tỉ lệ nhãn lần lượt là 1685 ảnh và 1517 ảnh.



Hình 2: Bộ dữ liệu sau khi thu thập, tiền xử lý.

3. PHƯƠNG PHÁP TIẾP CẬN

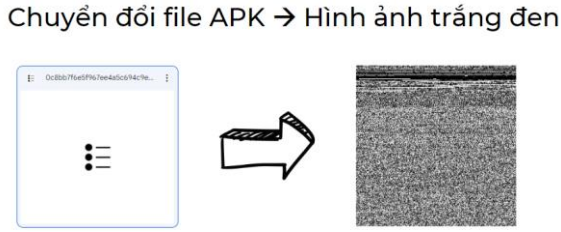
Với bài toán nhận dạng mã độc, chúng tôi tiến hành tiếp cận theo hai hướng. Hướng thứ nhất là Machine Learning, bao gồm các thuật toán như Logistic Regression, K-Nearest Neighbor (KNN) và Support Vector Machine (SVM). Bên cạnh đó, với Deep Learning, chúng tôi tiến hành xây dựng mô hình mạng CNN, một số kiến trúc mạng như Resnet50, MobileNetV2 và kết hợp kiến trúc mạng VGG16 với mô hình SVM để tạo ra hiệu suất tốt nhất. Tổng quan mô hình tiếp cận bài toán mô tả như Hình 3.



Hình 3: Tổng quan mô hình tiếp cận bài toán.

A. Tiền xử lý dữ liệu

Trong quá trình tìm hiểu bộ dữ liệu, chúng tôi nhận thấy bộ dữ liệu bao gồm các file zip chứa các file APK, do đó để mô hình có thể đạt được hiệu suất tốt nhất khi huấn luyện chúng tôi đã tiến hành tiền xử lý chuyển đổi tất cả các file APK thành dạng hình ảnh trắng đen (Grayscale Image) trước khi đưa vào mô hình huấn luyện. Bộ dữ liệu sau khi tiền xử lý.



Hình 4: Tiền xử lý dữ liệu.

B. Machine Learning:

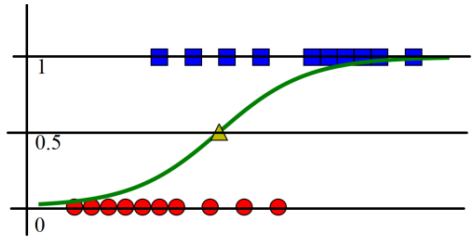
a) Logistic Regression:

Logistic Regression là một thuật toán phổ biến trong các bài toán phân lớp, đặc biệt là phân lớp nhị phân. Ví dụ các bài toán như phân loại email, dương tính hay âm tính với Covid. Thuật toán này sử dụng hàm sigmoid logistic để đưa ra đánh giá theo xác suất.

Đầu ra dự đoán của Logistic Regression có dạng viết chung là:

$$\mathbf{f(x)} = \theta(\mathbf{w}^T\mathbf{x})$$

Trong đó: θ được gọi là logistic function.



Hình 5: Mô hình Logistic Regression.

b) K-Nearest Neighbors:

K-Nearest Neighbors là một trong những thuật toán supervised-learning đơn giản nhất, hoạt động tốt trong trường hợp phân loại với nhiều lớp. Thuật toán này sẽ thực hiện mọi tính toán khi cần dự đoán kết quả của dữ liệu mới.

- Khoảng cách Euclidean là một trong các độ đo khoảng cách được sử dụng phổ biến trong KNN.

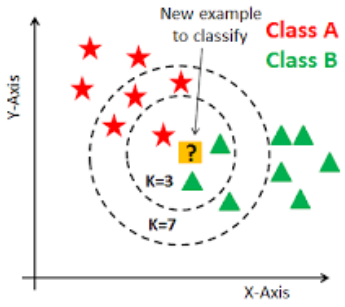
$$D(a,b) = \sqrt{\sum_i^n (b_i - a_i)^2}$$

Trong đó: a_i là đặc trưng thứ i của dữ liệu.

- Khoảng cách Hamming là độ đo được sử dụng khi làm việc với dữ liệu rời rạc.

$$D(a,b) = \sqrt{\sum_{i=1}^n (a_i \neq b_i)}$$

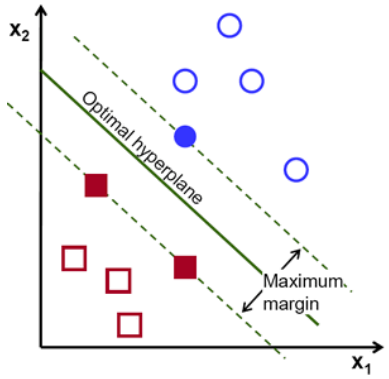
Trong đó: n là số lượng đặc trưng của dữ liệu.



Hình 6: Mô hình K-Nearest Neighbors.

c) Support Vector Machine:

Support Vector Machine là một thuật toán học có giám sát chủ yếu dùng cho việc phân loại. Thuật toán sẽ tìm ra hyper – plane phân chia các lớp ra thành hai phần riêng biệt.



Hình 7: Mô hình Support Vector Machine.

Hàm quyết định phân dữ liệu vào lớp thứ i là:

$$\mathbf{f_i(x) = w^T_i x + b_i}$$

Trong đó: w là vector pháp tuyến n chiều và b là giá trị ngưỡng. Vector pháp tuyến w xác định chiều của siêu phẳng f(x), giá trị ngưỡng b xác định khoảng cách giữa siêu phẳng và gốc.

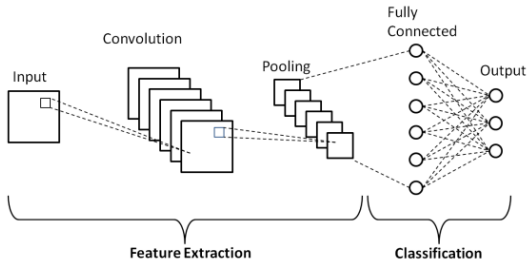
C. Deep Learning

a) Convolutional Neural Network:

Convolutional Neural Network (CNN) 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, mạng CNN được sử dụng rộng rãi trong việc phân loại và nhận dạng hình ảnh.

Kiến trúc cơ bản của CNN là như sau:

- **Lớp Convolutional:** Một chuỗi các lớp tích chập được áp dụng để trích xuất các đặc trưng từ hình ảnh. Mỗi lớp tích chập sử dụng một số lượng bộ lọc để tìm các đặc điểm khác nhau trong hình ảnh.
- **Lớp Max Pooling:** Lớp tối đa hóa được sử dụng để giảm kích thước của đặc trưng và giảm độ phức tạp tính toán. Nó giúp giữ lại thông tin quan trọng trong quá trình pooling.
- **Lớp Flatten:** Lớp này chuyển đổi đặc trưng từ định dạng ma trận thành một vectơ 1 chiều, chuẩn bị cho các lớp kết nối đầy đủ.
- **Lớp Dropout:** Lớp dropout được sử dụng để ngẫu nhiên bỏ qua một số đơn vị đầu ra trong quá trình huấn luyện, nhằm giảm overfitting và cải thiện khả năng tổng quát hóa của mô hình.
- **Các lớp kết nối đầy đủ (Fully Connected):** Các lớp kết nối đầy đủ cuối cùng được sử dụng để phân loại hình ảnh và đưa ra dự đoán cho từng lớp ảnh.



Hình 8: Mô tả các lớp của mô hình CNN.

b) VGG16:

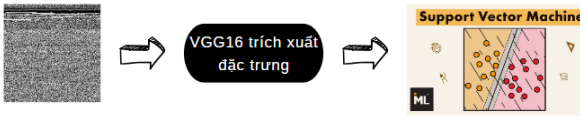
Là một mạng CNN được thiết kế với mục đích phân lớp hình ảnh. Bằng việc sử dụng các lớp tích chập, VGG16 có thể trích xuất đặc trưng từ những hình ảnh đầu vào. Trong bài báo, chúng tôi sử dụng toàn bộ 17 lớp tích chập (3x3) của VGG16 với số lượng filter tương ứng là 64, 64, 128, 128, 256, 256, 256, 256, 512, 512, 512, 512, 512, 512, 512, 512, 512 filters. Các lớp tích chập trích xuất đặc trưng của hình ảnh đầu vào. Các đặc trưng đó được giảm chiều ở lớp Max-pooling và trích xuất đặc trưng từ hình ảnh đầu vào. Cuối cùng, thuật toán Support Vector Machine sử dụng kết quả đó để dự đoán nhãn đầu ra.

Kiến trúc mạng VGG16:

- **Input:** ảnh dữ liệu đầu vào với kích thước là 224x224x3.
- **Block 1:** hai lớp tích chập, mỗi lớp sử dụng 64 filters với mỗi filter là 3x3.
- **Max_pooling(2, 2).**

- **Block 2:** hai lớp tích chập, mỗi lớp sử dụng 128 filters với mỗi filter là 3x3.
- **Max_pooling(2, 2).**
- **Block 3:** ba lớp tích chập, mỗi lớp sử dụng 256 filters với mỗi filter là 3x3.
- **Max_pooling(2, 2).**
- **Block 4:** ba lớp tích chập, mỗi lớp sử dụng 512 filters với mỗi filter là 3x3.
- **Max_pooling(2, 2).**
- **Block 5:** ba lớp tích chập, mỗi lớp sử dụng 512 filters với mỗi filter là 3x3.
- **Max_pooling(2, 2).**
- **Flatten:** làm phẳng dữ liệu thành một cột để thuận tiện đưa vào bài toán học máy Support Vector Machine.

❖ **VGG16 kết hợp Support Vector Machine:**



Hình 9: Mạng VGG16 kết hợp mô hình SVM.

c) ResNet50:

ResNet50 (Residual Network 50) là một kiến trúc mạng nơ-ron tích chập (CNN) được sử dụng phổ biến trong lĩnh vực phân loại hình ảnh. Mạng này được thiết kế để có khả năng học các đặc trưng phức tạp từ dữ liệu hình ảnh và đạt được hiệu suất tốt trong các nhiệm vụ phân loại.

ResNet50 được xây dựng dựa trên khái niệm của "residual learning". Trong mạng này, các đường kết nối được tạo ra để truyền thông tin trực tiếp từ các lớp đầu vào sang các lớp đầu ra, mà không thông qua các lớp trung gian. Điều này giúp mạng học được những đặc trưng chi tiết của hình ảnh một cách hiệu quả và tránh tình trạng "vanishing gradient".

Kiến trúc mạng ResNet50:

- **Input:** ảnh dữ liệu đầu vào với kích thước là 224x224x3
- **Convolutional Block 1:** Hai lớp tích chập, mỗi lớp sử dụng 64 bộ lọc với kích thước 3x3.
- **Max Pooling:** Kích thước cửa sổ 2x2 với bước nhảy là 2.
- **Convolutional Block 2:** Hai lớp tích chập, mỗi lớp sử dụng 128 bộ lọc với kích thước 3x3.
- **Max Pooling.**
- **Convolutional Block 3:** Ba lớp tích chập, mỗi lớp sử dụng 256 bộ lọc với kích thước 3x3.
- **Max Pooling.**
- **Convolutional Block 4:** Ba lớp tích chập, mỗi lớp sử dụng 512 bộ lọc với kích thước 3x3.
- **Max Pooling.**
- **Convolutional Block 5:** Ba lớp tích chập, mỗi lớp sử dụng 512 bộ lọc với kích thước 3x3.
- **Max Pooling.**
- **Flatten:** Dữ liệu được làm phẳng thành một vector để truyền vào bước cuối cùng.
- **Fully Connected Layer:** Một hoặc nhiều lớp kết nối đầy đủ để phân loại hình ảnh.

c) MobileNetV2:

MobileNetV2 là một kiến trúc mạng nơ-ron tích chập (CNN) nhẹ và hiệu quả được thiết kế để đạt được một sự cân bằng tốt giữa độ chính xác và tốc độ tính toán. Nó được phát triển đặc biệt để chạy trên các thiết bị di động có tài nguyên tính toán hạn chế, nhưng vẫn đem lại kết quả phân loại chất lượng cao.

Kiến trúc mạng MobileNetV2:

- **Input:** ảnh dữ liệu đầu vào với kích thước là 224x224x3
- **Convolutional Block:** Một lớp tích chập sử dụng bộ lọc 3x3 với số bộ lọc đầu ra được xác định bởi tham số alpha để điều chỉnh kích thước mô hình.
- **Bottleneck Block:** Một đơn vị bottleneck được áp dụng lặp đi lặp lại nhiều lần. Đơn vị bottleneck bao gồm các lớp tích chập 1x1 để giảm số chiều của đặc trưng, sau đó là một lớp tích chập 3x3 và cuối cùng là một lớp tích chập 1x1 để tăng số chiều trở lại. Đơn vị này giúp giảm số lượng tham số và tính toán.
- **Inverted Residual Block:** Một phiên bản cải tiến của Bottleneck Block, với việc sử dụng kỹ thuật gọi là "inverted residual". Đây là một cấu trúc chồng lớp tích chập sử dụng kỹ thuật "squeeze-and-excite" để tăng cường khả năng học của mạng.
- **Depthwise Separable Convolution:** Sử dụng các lớp tích chập "depthwise" và "pointwise" để giảm tính toán và số lượng tham số. Lớp tích chập "depthwise" áp dụng các bộ lọc riêng lẻ cho

từng kênh đầu vào, sau đó lớp tích chập "pointwise" kết hợp các kênh lại với nhau.

- **Global Average Pooling:** Chuyển đổi các đặc trưng thành một tensor có kích thước 1x1xK, trong đó K là số kênh đầu ra cuối cùng.
- **Fully Connected Layer:** Một hoặc nhiều lớp kết nối đầy đủ để phân loại hình ảnh.

4. THỬ NGHIỆM VÀ ĐÁNH GIÁ

A. Thông số đánh giá

Chúng tôi sử dụng hai thang đo sau để đánh giá các mô hình

1) F1-score: là trung bình điều hòa của recall và precision, độ đo để đánh giá hiệu suất của các mô hình.

$$F1 = 2\ x\ \frac{Precision\ x\ Recall}{Precision + Recall}$$

Trong đó:

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

2) Accuracy: là độ đo dùng để tính tỉ lệ giữa số kết quả chính xác và tổng số lần thử nghiệm.

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

Trong đó:

- True Positive (TP): Giá trị dự đoán khớp với giá trị thực tế. Nhãn dương tính được dự đoán là dương tính.
- True Negative (TN): Giá trị dự đoán khớp với giá trị thực tế. Nhãn âm tính được dự đoán là âm tính.
- False Positive (FP): Giá trị dự đoán bị sai. Nhãn âm tính bị dự đoán là dương tính.
- False Negative (FN): Giá trị dự đoán bị sai. Nhãn dương tính bị dự đoán là âm tính.

B. Kết quả thử nghiệm

Chúng tôi tiến hành đánh giá hiệu năng của các mô hình học máy và học sâu thông qua độ chính xác (Accuracy) và hiệu suất mô hình (F1-score). Kết quả đạt được như sau:

Bảng 1. Kết quả thử nghiệm.

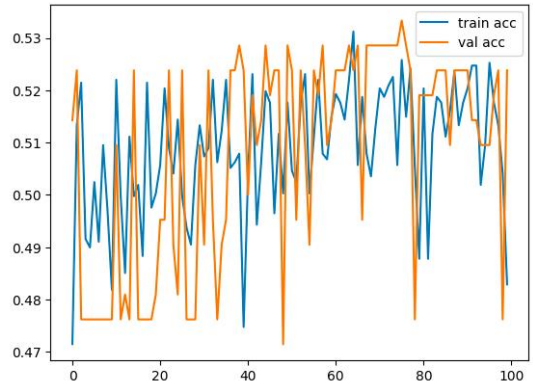
Model		Accuracy	F1-score
Machine Learning	Logistic Regression	**59,67%**	**66,3%**
	K-Nearest Neighbors	46,77%	53,5%
	Support Vector Machine	50%	50%
Deep Learning	VGG16 kết hợp SVM	48,06%	62,11%
	CNN	53%	53%
	VGG16	49%	49%
	MobileNetV2	49%	49%
	Resnet50	52%	53%

- Qua bảng kết quả thử nghiệm có thể thấy, với các mô hình học máy, mô hình Logistic Regression cho kết quả khả quan nhất với Accuracy là 59,67% và F1-score là 66,3%; mô hình K-Nearest Neighbors cho kết quả Accuracy 46,77% và F1-score là 53,5%; mô hình Support Vector Machine cho kết quả Accuracy 50% và F1-score là 50%.
  Với các mô hình học sâu, mô hình VGG16 kết hợp Support Vector Machine cho kết quả Accuracy 48,06% và F1-score là 62,11%; mô hình CNN cho kết quả Accuracy là 53% và F1-score là 53%. Bên cạnh đó, với kiến trúc mạng cho kết quả lần lượt như sau Resnet50 (Accuracy 52% và F1-score là 53%), VGG16 (Accuracy 49% và F1-score là 49%), MobileNetV2 (Accuracy 49% và F1-score là 49%).
- Nhìn chung, các mô hình cho kết quả không khả quan với kết quả xấp xỉ 50%. Trong đó, mô hình học sâu cho kết quả thấp hơn so với các mô hình học máy. Chúng tôi nhận thấy rằng mô hình Logistic Regression cho ra kết qua ổn định nhất.

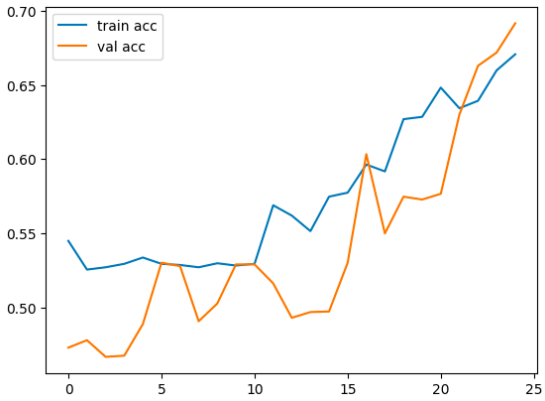
5. PHÂN TÍCH LỖI

- **Đối với mô hình CNN:**

Bởi vì chúng tôi tiến hành cập nhật thêm mới vào bộ dữ liệu để cải thiện quá trình huấn luyện cho mô hình, nên có sự khác biệt giữa Accuracy trên hai tập Train và Val trước và sau khi cập nhật bộ dữ liệu:



Hình 10: Accuracy trên tập Train và Val trước khi cập nhật bộ dữ liệu.

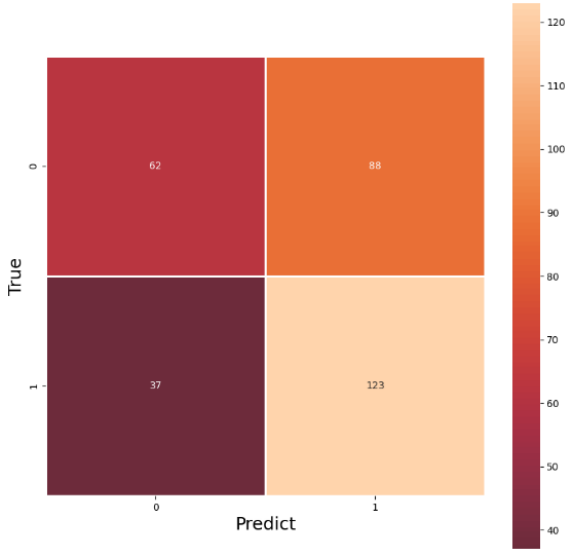


Hình 11: Accuracy trên tập Train và Val sau khi cập nhật bộ dữ liệu, áp dụng EarlyStopping + Tăng cường dữ liệu.

Đúng như dự đoán, việc bổ sung thêm dữ liệu giúp cho mô hình đưa ra được kết quả cao hơn, tuy nhiên có sự khác biệt lớn ở số lượng epoch của mô hình (100 với 25). Đối với mô hình huấn luyện trên bộ dữ liệu có số lượng ít hơn chúng tôi lựa chọn số epoch là 100 nhằm mục đích khắc phục điểm yếu là số lượng dữ liệu còn hạn chế, vì thế vô tình dẫn đến tình trạng Overfitting cho mô hình. Do đó việc áp dụng EarlyStopping,tăng cường dữ liệu và chọn ra số epoch để training mô hình một cách hợp lí sẽ tránh được tình trạng trên cũng như tiết kiệm thời gian huấn luyện, đồng thời cũng lựa chọn ra được sự hội tụ tốt nhất cho mô hình.

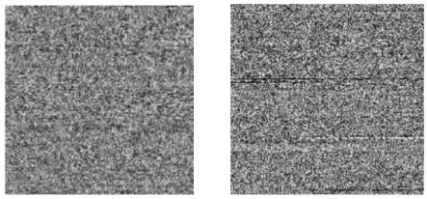
- **Đối với mô hình Logistic Regression:**

Chúng tôi phân tích các lỗi thường gặp dựa trên ma trận nhầm lẫn của mô hình như sau:



Hình 12: Ma trận nhầm lẫn của mô hình Logistic Regression.

Nhìn chung, độ chính xác của mô hình Logistic Regression cho kết quả khá khả quan. Tuy nhiên, mức độ âm tính giả của mô hình vẫn còn cao. Trong quá trình tiền xử lý dữ liệu chúng tôi nhận thấy các hình ảnh từ file độc (Adware) và lành tính (Benign) không có quá nhiều sự khác biệt nên sẽ gây nhầm lẫn trong quá trình các mô hình huấn luyện.



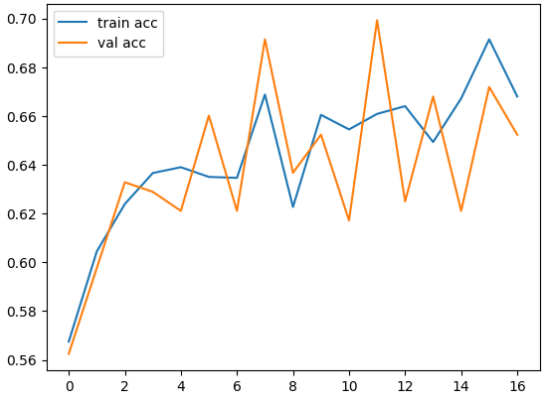
Hình 13: Một số trường hợp âm tính giả. (Bên trái: Adware, Bên phải: Benign)

• **Đối với một số kiến trúc mạng:**

Chúng tôi đã tiến hành thử nghiệm trên một số mô hình học sâu khác, mô hình được huấn luyện trên cùng bộ tham số và có áp dụng các kĩ thuật tăng cường dữ liệu tránh overfitting, cuối cùng, chúng tôi có một số nhận xét như sau:

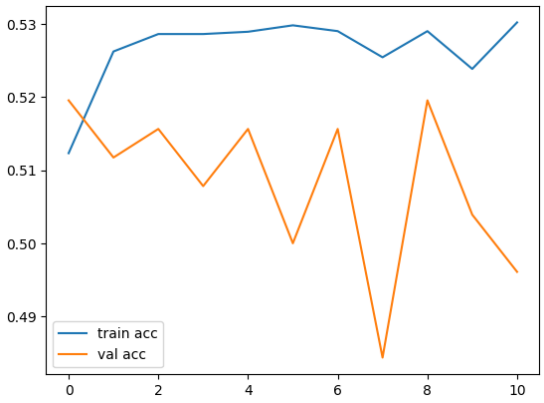
- Về thời gian huấn luyện: ResNet50 là mô hình huấn luyện với thời gian ít nhất, kế đến là MobileNetV2 và cuối cùng là VGG-16.
- Về kết quả thử nghiệm trên tập test: cả 3 mô hình đều cho ra kết quả khá thấp (48%-50%) trên hai thang đo Accuracy và F1-Score, tuy nhiên đối với mô hình MobileNetV2 thì kết quả khả quan hơn hai mô hình Resnet50 và VGG16. Khi mô hình VGG16 và ResNet50 dừng lại trong khi huấn luyện thì MobileNetV2 lại huấn luyện hết trên 50 epoch.
- Điều này cũng dễ hiểu bởi vì MobileNetV2 là một kiến trúc mạng nơ-ron tích chập (CNN) nhẹ và hiệu quả được thiết kế để đạt được sự cân bằng giữa độ chính xác và tốc độ tính toán. Mô hình được phát triển đặc biệt để giải quyết các bài toán có tài nguyên thấp.

❖ **VGG-16:**



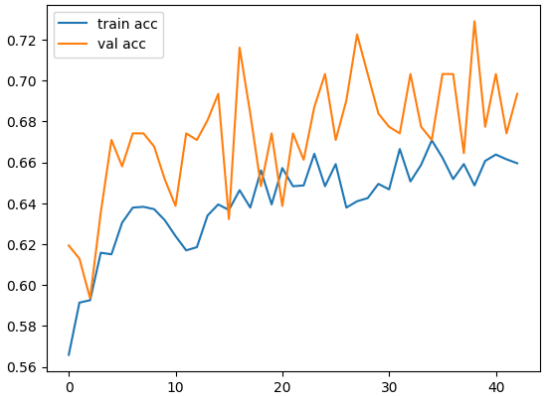
Hình 14: Accuracy trên tập Train và Val.

❖ **ResNet50:**



Hình 15: Accuracy trên tập Train và Val.

❖ **MobileNetV2:**



Hình 16: Accuracy trên tập Train và Val.

6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

A. Kết luận

Trong nghiên cứu này, chúng tôi đã xây dựng được các mô hình học máy, học sâu khác nhau để nhận dạng phát hiện mã độc file APK thông qua hình ảnh. Các mô hình bước đầu đã đạt được kết quả khả quan trong việc nhận dạng được hình ảnh đầu vào là độc hại hay lành tính, tuy nhiên, hiệu suất tổng thể vẫn còn kém khi chỉ đạt khoảng 50-60% độ chính xác. Dựa trên kết quả đánh giá, chúng tôi nhận thấy mô hình chưa đạt được hiệu suất như mong muốn và cần cải thiện một số yếu điểm như sau:

- Tập dữ liệu huấn luyện chưa chất lượng: Tập dữ liệu hiện tại còn hạn chế về số lượng và cả nhãn của các loại mã độc khác nhau, dẫn đến mô hình không học được đầy đủ các đặc trưng và chưa thể phân loại một cách chính xác nhất.
- Kích thước dữ liệu và độ phân giải: Một số hình ảnh trong tập dữ liệu có kích thước nhỏ và độ phân giải khác nhau, gây ảnh hưởng đến khả năng nhận dạng và trích xuất đặc trưng của mô hình.
- Kiến trúc mô hình: Các kiến trúc mô hình cần được tinh chỉnh và điều chỉnh thêm để phù hợp với đặc điểm của bộ dữ liệu bao gồm hình ảnh mã độc. Có thể cần thử nghiệm và đánh giá các kiến trúc mạng khác nhau để tìm kiếm một kiến trúc tốt hơn cho bài toán nhận dạng mã độc này
- Tham số của mô hình: Vì các tham số là những thứ ảnh hưởng trực tiếp đến hiệu suất của mô hình nên việc thử nghiệm trên chúng là điều cần thiết, tuy nhiên vì thời lượng của môn học có hạn và việc tinh chỉnh tham số phải đi kèm với việc cập nhật lại dữ liệu nên điều này khá mất thời gain và phải có kĩ năng phân tích bài bản.

B. Hướng phát triển

Trong tương lai, chúng tôi sẽ tiếp tục mở rộng, tăng cường tập dữ liệu, đa dạng hóa thêm các nhãn của tập dữ liệu để mô hình có thể trích xuất đầy đủ các đặc trưng quan trọng. Bên cạnh đó, chúng tôi sẽ áp dụng thêm các phương pháp tiền xử lý và chuẩn hóa dữ liệu để cải thiện, đồng nhất ảnh đầu vào, tăng khả năng nhận dạng của mô hình. Cùng với đó, chúng tôi sẽ thử nghiệm thêm các kiến trúc mạng nâng cao và mô hình khác nhau bao gồm áp dụng các kỹ thuật như Transfer Learning để nâng cao hiệu suất phát hiện, nhận dạng mã độc. Kết hợp với đánh giá mô hình để đảm bảo tính tin cậy và khả năng áp dụng thực tế của nó. Và tiếp theo nữa là thử nghiệm trên nhiều bộ tham số kác nhau trên từng mô hình khác nhau và từng tập dataset khác nhau. Xa hơn nữa trong tương lai, nếu kết quả thu được khả quan hơn chúng tôi sẽ tiến hành phát triển và mở rộng mô hình này trên nền tảng ứng dụng hoặc Website để người dùng có thể sử dụng và phát hiện mã độc một cách nhanh chóng, chính xác.

Tổng kết, với bài toán nhận dạng mã độc này, chúng tôi bước đầu đã đạt được kết quả khả quan trong việc phát hiện mã độc thông qua hình ảnh, tuy nhiên cần tiếp tục nghiên cứu và cải thiện để đạt được hiệu suất tốt hơn trong thời gian tới.

Cuối cùng, nhóm chúng em xin cảm ơn thầy Nguyễn Tấn Cầm đã tạo điều kiện để chúng em được tiếp cận, nghiên cứu về đề tài khá thực tế và hữu ích này.

REFERENCES

X. Xing, X. Jin, H. Elahi, H. Jiang and G. Wang, "A Malware Detection Approach Using Autoencoder in Deep Learning," in IEEE Access, vol. 10, pp. 25696-25706, 2022, doi:10.1109/ACCESS.2022.3155695

J. Sahs and L. Khan, "A Machine Learning Approach to Android Malware Detection," 2012 European Intelligence and Security Informatics Conference, Odense, Denmark, 2012, pp. 141-147, doi:10.1109/EISIC.2012.34

Niall McLaughlin, Jesus Martinez del Rincon, BooJoong Kang, Suleiman Yerima, Paul Miller, Sakir Sezer, Yeganeh Safaei, Erik Trickel, Ziming Zhao, Adam Doupé, and Gail Joon Ahn. 2017. Deep Android Malware Detection. In Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy (CODASPY '17). Association for Computing Machinery, New York, NY, USA, 301–308. https://doi.org/10.1145/3029806.3029823

Vasan, D., Alazab, M., Wassan, S., Naeem, H., Safaei, B., & Zheng, Q. (2020). IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture. Computer Networks, 2022, 171, 107138. https://doi.org/10.1016/j.comnet.2020.107138

S. Choi, S. Jang, Y. Kim and J. Kim, "Malware detection using malware image and deep learning," 2017 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, Korea (South), 2017, pp. 1193-1195, doi: 10.1109/ICTC.2017.8190895.

S. Mohan, "Keras Implementation of VGG16 Architecture from Scratch with Dogs Vs Cat Data Set" https://machinelearningknowledge.ai/keras-implementation-of-vgg16-architecture-from-scratch-with-dogs-vs-cat-data-set/.

M. A. R. Khan, P. N. Kumar and R. C. Tripathi, Detection of Android Malware App through Feature Extraction and Classification of Android Image, 2022.