Chuyển Đổi Tập Tin APK Thành Hình Ảnh Để Nhận Biết Mã Độc

Nguyễn Chí Thắng[1*],
*19522205@gm.uit.edu.vn

TỪ KHÓA

Nhận biết mã độc Android, chuyển đổi hình ảnh APK.

DEMO/SOURCE

TÓM TẮT

Đề tài nghiên cứu về việc biến đổi file APK (Android Application Package) thành dạng ảnh để phân tích và nhận biết mã độc. File APK là định dạng tập tin chứa các thành phần của một ứng dụng Android, bao gồm mã nguồn, tài nguyên và các tệp tin khác. Việc chuyển đổi file APK thành dạng ảnh có thể giúp dễ dàng trực quan hóa các thành phần của ứng dụng và phát hiện các đặc điểm đáng ngờ.

1. GIỚI THIỆU

Trong bài báo cáo này, một mô hình phân loại mã độc đã được đề xuất để phát hiện các mẫu mã độc trong môi trường Android. Mô hình được đề xuất dựa trên việc chuyển đổi một số tập tin từ nguồn của các ứng dụng Android thành ảnh grayscale. Các ảnh được đưa về kích thước cố định như 256x256 hoặc 64x64 để đưa vào các mô hình học máy để huấn luyện. Các mô hình phân loại được sử dụng là CNN và SVM. Trong đó, mạng CNN được sử dụng để trích xuất đặc trưng từ ảnh, sau đó sử dụng mạng Neural Network và SVM để tiến hành phân loại. Phương pháp đề xuất đạt được độ chính xác phân loại cao nhất là 87.05%. Trong phần còn lại của bài báo cáo, tôi sẽ trình bày về quá trình chuyển đổi tập tin APK thành hình ảnh, các phương pháp, mô hình được sử dụng để nhận biết mã độc, cũng như đánh giá hiệu quả của phương pháp qua các thử nghiệm và so sánh với các phương pháp được thử nghiệm.

2. DỮ LIỆU VÀ PHƯƠNG PHÁP NGHIÊN CỨU

2.1. Cách Tiếp Cận

Ứng dụng Android gốc (Android Application Package) là một bản lưu trữ chứa các thành phần khác nhau của ứng dụng Android và không phù hợp để trực tiếp đưa vào các thuật toán học máy để phân tích và dự đoán tự động. Mỗi gói ứng dụng Android chứa các thành phần như library, class, ... và các tệp cấu hình để ứng dụng hoạt động. Sự khác biệt về nội dung, tính chất và thành phần của những thành phần này tạo nên đặc điểm riêng của mỗi ứng dụng Android. Những đặc điểm này tạo nên những đặc tính giúp thuật toán học máy phân biệt được giữa các loại ứng dụng Android khác nhau.

Đề tài được báo cáo trong bản báo cáo này lấy cảm hứng từ công việc tương tự trong lĩnh vực hình ảnh. Tôi xem xét một ứng dụng Android như một hình ảnh được chuyển đổi mang những đặc điểm đặc trưng, có thể nhận biết được bằng các mô hình máy học. Ở mức thấp, một ứng dụng Android đơn giản được đọc thành một chuỗi các con số (binary file). Điều này có các đặc điểm tương tự như các pixel hình ảnh, khi mỗi hình ảnh được tập hợp bởi các pixel có các giá trị khác nhau.

Đề tài này sử dụng một chu kỳ máy học hai giai đoạn đơn giản, bao gồm công việc liên quan đến xử lý dữ liệu là giai đoạn đầu tiên, tiếp theo là công việc thiết kế và huấn luyện mô hình. Công việc về dữ liệu liên quan đến quy trình xử lý dữ liệu gốc, chuyển đổi thành định dạng hình ảnh. Trong khi đó, kỹ thuật mô hình là quá trình lặp lại của việc huấn luyện và điều chỉnh các mô hình. Một sơ đồ mô tả quy trình được trình bày trong Figure 1.

- Bước 1: Các file APK được thu thập, lưu trữ trong các folder riêng biệt.
- Bước 2: Giải nén các file APK, thu thập hai file Manifest.xml và file Dex, lưu trữ chúng trong các folder theo các loại mã độc.
- Bước 3: Chuyển đổi lần lượt file Manifest.xml và Dex thành hình ảnh, thông qua thuật toán 1 được cho ở dưới. Lưu thành ba foler riêng, một folder chứa ảnh được chuyển từ file Dex, một folder chứa ảnh chuyển từ file Manifest, và một folder lưu cả hai dạng ảnh với các kích thước phù hợp.

Thuật toán 1: Chuyển đổi file APK thành ảnh
Input: APK Dataset folder: $D_{APK} = \{ d_{apk1}, d_{apk2}, \ldots, d_{apkn})$
Result: Image Dataset folder: $D_{IMG} = \{ d_{img1}, d_{img2}, d_{img3}\}$
For all $d_{apki} \in$ DAPK **do**:
  For all apks $\in d_{apki}$ **do**:
    Đọc file APK dưới dạng Binary
    Chuyển về dạng buffer
    Chuyển buffer về ma trận có kích thước cố định
    Chuyển ma trận buffer thành ảnh grayscale và lưu vào d_{imgi} phù hợp
  End for
End for

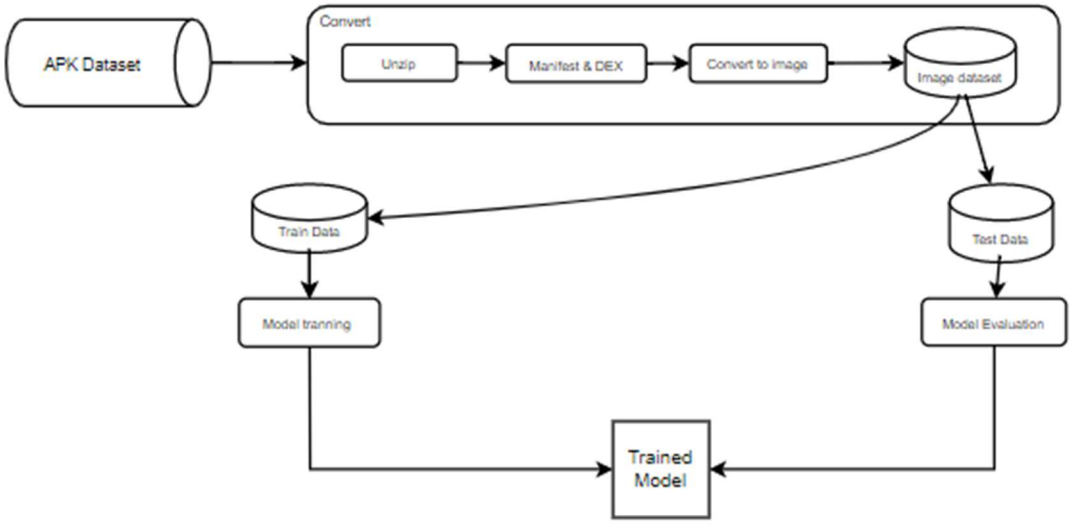

Figure 1. Sơ đồ các bước thực hiện

- Bước 4: Kết quả của quá trình trên được chia thành ba tập dữ liệu: tập huấn luyện (training), tập xác thực (validation) và tập kiểm tra (testing).
- Bước 5: Các mô hình được tạo, huấn luyện và xác thực để học các mẫu bên trong có thể được sử dụng để phân biệt giữa các tệp gói ứng dụng Android độc hại và không độc hại. Các giai đoạn huấn luyện và xác thực được lặp lại cho đến khi đạt được mức độ hiệu suất tối ưu. Giai đoạn này bao gồm tối ưu hóa siêu tham số (hyperparameter).
- Bước 6: Độ khái quát hóa của mô hình được kiểm tra khi nó đưa ra dự đoán trên dữ liệu được lấy từ tập kiểm tra (testing set).

2.2. Tổng quan Dataset

2.2.1. Dataset

Bộ dữ liệu CICMalAnal2017 do Viện An ninh mạng Canada cung cấp được sử dụng trong đề tài này. Bộ dữ liệu CICMalAnal bao gồm 925 ứng dụng Android được thu thập từ nhiều nguồn khác nhau. Bộ dữ liệu này bao gồm năm danh mục chính của mã độc Android. Chi tiết về bộ dữ liệu được cung cấp trong Bảng 1.

Bộ dữ liệu được sử dụng bao gồm một tập hợp các mẫu ảnh được chuyển đổi từ các file nhận được khi

giải nén tập tin APK. Tổng số lượng mẫu trong bộ dữ liệu là 925, và số lượng mẫu chứa mã độc là 426.

Cụ thể, trong bộ dữ liệu này, số lượng mẫu chứa mã độc chiếm tỷ lệ 46.05%. Điều này cho thấy sự cân bằng của các mẫu chứa mã độc trong bộ dữ liệu đối với mẫu "sạch" còn lại.

2.2.2. Phân Tích Dữ Liệu

Sau các khi chuyển đổi các tệp tin thu được từ việc giải nén file APK, ta nhận được 3 folder chứa ảnh lần lượt là Manifest, Dex và Full. Folder đầu tiên chứa ảnh được chuyển từ file Manifest.xml với kích thước cố định 64x64. Tương tự, folder thứ hai, ta có ảnh kích thước 256x256 được biến đổi từ file Dex và cuối cùng là folder chứa ảnh tổng hợp, tức là ảnh được tổng hợp lại từ cả hai file phía trên, với kích thước cố định là 256x256.

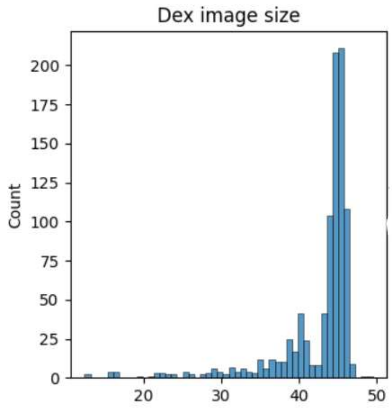

Figure 2.1. Histogram kích thước ảnh Dex

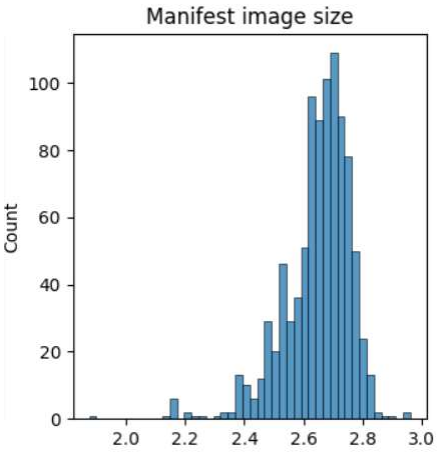

Figure 2.2. Histogram kích thước ảnh Manifest

Thống kê về kích thước thực thể trong toàn bộ tập dữ liệu có thể được tóm tắt trong Figure 2.

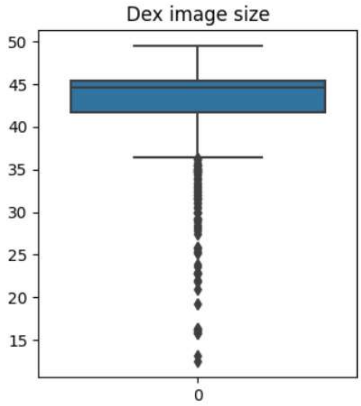

Figure 2.3. Boxplot kích thước ảnh Dex

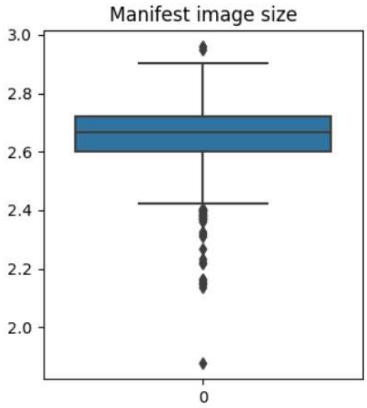

Figure 2.4. Histogram kích thước ảnh Manifest

i các biểu đồ thống kê, ta có thể dễ dàng thấy
ực kích thước các file chủ yếu nằm trong khoảng
45kB với các ảnh được biến đổi từ file Dex và
ảng 2.6 – 2.8kB đối với ảnh được chuyển từ
Manifest. Những ảnh còn lại, chủ yếu có kích
ớc nhỏ hơn các khoảng được đề cập ở trên.
kích thước giới hạn của file Dex và Manifest,
h thước được tạo ra từ các tệp này là cố định sau
trải qua nhiều lần thử gồm 256, 128, 64, 32. Sự
ng đồng lớn giữa hình ảnh dựa trên tệp
nifest, Dex của các ứng dụng Android có thể
y trong Figure 3.

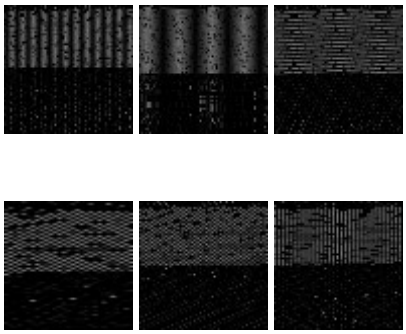

Figure 3. Ảnh được chuyển đổi từ các file. Ảnh trên là của các file apk "sạch". Hàng dưới là ảnh của các file có chứa mã độc

2.3. Các Phương Pháp Sử Dụng

Đề tài này tập trung vào việc nhận biết mã độc bằng cách sử dụng mạng CNN để trích xuất đặc trưng từ ảnh, sau đó sử dụng các mô hình Support Vector Machine (SVM) và Deep Neural Network (DNN) để nhận biết mã độc.

CNN là một kiến trúc mạng nơ-ron được thiết kế đặc biệt cho việc xử lý dữ liệu không gian như hình ảnh, có khả năng tự động học và nhận diện các đặc trưng quan trọng trong dữ liệu. Các lớp tích chập trong CNN giúp xác định các đặc trưng cục bộ của ảnh, trong khi các lớp gộp giúp tạo ra biểu diễn tổng quát hơn. CNN được thiết kế dựa trên nguyên tắc của hai khái niệm quan trọng: tích chập (convolution) và gộp (pooling). Tích chập cho phép mạng nơ-ron học các đặc trưng cục bộ từ dữ liệu, trong khi gộp giúp giảm kích thước dữ liệu và tạo ra các biểu diễn tổng quát. Kiến trúc của CNN cũng bao gồm các lớp kích hoạt phi tuyến tính (non-linear activation) như ReLU (Rectified Linear Unit) để tạo sự phi tuyến tính và khả năng học các biểu diễn phức tạp.

Sau khi trích xuất đặc trưng từ ảnh bằng CNN, các đặc trưng này được sử dụng để huấn luyện các mô hình phân loại như SVM và DNN.

SVM là một mô hình học máy thuộc lớp các mô hình học có giám sát. Nó được sử dụng để phân loại dữ liệu thành hai hoặc nhiều lớp khác nhau. SVM tìm một siêu mặt phẳng trong không gian đặc trưng, tạo ra một ranh giới tuyến tính tốt nhất giữa các lớp dữ liệu. SVM có khả năng xử lý cả dữ liệu tuyến tính và phi tuyến tính thông qua các hàm nhân (kernel) để ánh xạ dữ liệu vào không gian đặc trưng cao hơn.

DNN, hay còn gọi là mạng nơ-ron sâu, là một loại mạng nơ-ron nhân tạo có nhiều lớp ẩn giữa lớp đầu vào và lớp đầu ra. DNN có khả năng học các biểu diễn phức tạp và tự động trích xuất đặc trưng từ dữ liệu đầu vào. Mạng nơ-ron sâu được xây dựng dựa trên nguyên lý lan truyền ngược (backpropagation), trong đó các trọng số của mạng được điều chỉnh để tối ưu hóa một hàm mất mát. DNN thường có khả năng học các mô hình phân loại phức tạp và hiệu quả hơn so với các mô hình truyền thống.

Quá trình huấn luyện mô hình SVM và DNN được thực hiện bằng cách cung cấp các đặc trưng từ ảnh mã độc và không độc. Mô hình sẽ tự động học cách phân biệt giữa các loại ảnh và xây dựng một quy tắc phân loại dựa trên sự khác biệt trong các đặc trưng và cấu trúc của chúng.

Ở đề tài này, tôi áp dụng từng cách xử lý dữ liệu, xây dựng mạng CNN riêng cho từng tệp dữ liệu, cụ thể là Dex, Manifest và Full - kết hợp cả hai loại. Tệp Dex chứa ảnh được chuyển đổi từ file Dex của APK, kích thước cố định là 256x256. Tương tự, tệp Manifest chứa ảnh được đổi từ file Manifest.xml với kích thước 64x64. Cuối cùng, tệp Full tổng hợp, chứa cả hai loại ảnh của cùng một gói APK, nhưng được thay đổi về kích thước cố định 256x256.

2.3.1. Áp dụng CNN, DNN lên tệp Dex

Với tập dữ liệu này, tôi áp dụng kỹ thuật tiền xử lý chuẩn hóa dữ liệu, đưa các giá trị pixel về các giá trị [0, 1] bằng cách chia tất cả các pixel cho 255.

Sau đó tiến hành chia thành train và test set, với tỉ lệ 85%, 15% tương đương với 786 ảnh dành cho việc huấn luyện và 139 ảnh cho việc kiểm thử.

Tiếp đến, tôi áp dụng mô hình CNN và DNN để trích xuất đặc trưng và tiến hành huấn luyện phân loại mã độc.
Mô hình được xây dựng theo cấu trúc sau:

1. Lớp đầu vào (Input layer): Ảnh đầu vào kích thước 256x256 pixel.

2. Lớp tích chập (Convolutional layer): Sử dụng 32 bộ lọc có kích thước 3x3 và hàm kích hoạt ReLU.
3. Lớp pooling tối đa (Max-pooling layer): Sử dụng cửa sổ pooling có kích thước 2x2.
4. Lớp tích chập (Convolutional layer): Sử dụng 64 bộ lọc có kích thước 3x3 và hàm kích hoạt ReLU.
5. Lớp pooling tối đa (Max-pooling layer): Sử dụng cửa sổ pooling có kích thước 2x2.
6. Lớp dropout (Dropout layer): Áp dụng dropout với tỷ lệ 0.425 để giảm overfitting.
7. Lớp flatten (Flatten layer): Chuyển đổi đầu ra của lớp trước thành một vector 1 chiều.
8. Lớp kết nối đầy đủ (Dense layer): Sử dụng 128 nơ-ron và hàm kích hoạt ReLU.
9. Lớp dropout (Dropout layer): Áp dụng dropout với tỷ lệ 0.425 để giảm overfitting.
10. Lớp đầu ra (Output layer): Sử dụng 2 nơ-ron và hàm kích hoạt softmax để dự đoán xác suất của mỗi lớp (mã độc và không độc hại).

Mô hình này đã được biên dịch với bộ tối ưu hóa 'adam', hàm mất mát là Sparse Categorical Crossentropy và metric là 'accuracy'.

2.3.2. Áp dụng CNN, DNN lên tệp Manifest

Với tệp ảnh được chuyển đổi từ file Manifest, tôi cũng thực hiện xử lý, huấn luyện, kiểm thử tương tự như những ảnh được lấy từ tập Dex.

Tuy nhiên, vì kích thước cố định của file ảnh Manifest là 64x64 nên cấu trúc mạng CNN cũng phải thay đổi. Cụ thể, ở đây, tôi sử dụng cấu trúc mạng CNN như sau.

1. Lớp tích chập (Conv2D): Sử dụng 32 bộ lọc có kích thước 3x3, hàm kích hoạt ReLU và padding='same'. Đầu vào có kích thước (64, 64, 1).
2. Lớp tích chập (Conv2D): Sử dụng 64 bộ lọc có kích thước 3x3, hàm kích hoạt ReLU và padding='same'.
3. Lớp pooling tối đa (MaxPooling2D): Sử dụng cửa sổ pooling có kích thước 2x2.
4. Lớp dropout (Dropout): Áp dụng dropout với tỷ lệ 0.425 để giảm overfitting.
5. Lớp flatten (Flatten): Chuyển đổi đầu ra của lớp trước thành một vector 1 chiều.
6. Lớp kết nối đầy đủ (Dense): Sử dụng 128 nơ-ron và hàm kích hoạt ReLU.
7. Lớp dropout (Dropout): Áp dụng dropout với tỷ lệ 0.425 để giảm overfitting.
8. Lớp đầu ra (Output layer): Sử dụng 2 nơ-ron và hàm kích hoạt softmax để dự đoán xác suất của mỗi lớp (mã độc và không độc hại).

Mô hình này vẫn được biên dịch với bộ tối ưu hóa 'adam', hàm mất mát là Sparse Categorical Crossentropy và metric là 'accuracy'.

2.3.3. Áp dụng CNN, DNN lên tệp Full

Tệp Full ở đây, là tệp có chứa ảnh của cả hai file Dex và Manifest của một ứng dụng. Kích thước của anh đều được đưa về kích thước 256x256.

Với tập dữ liệu này, tôi áp dụng kỹ thuật tiền xử lý chuẩn hóa dữ liệu, đưa các giá trị pixel về các giá trị [0, 1] bằng cách chia tất cả các pixel cho 255.

Sau đó tiến hành chia thành gộp hai ảnh của cùng một file APK thành một ảnh khác có hai chanel. Có nghĩa là, với tập Full này, ảnh đầu vào khi đưa vào mô hình sẽ có kích thước là 256x256x2. Khi đã thực hiện gộp hết các ảnh, ta tiến hành gộp nhãn sao cho đồng nhất dữ liệu. Tiếp đến, chia bộ dữ liệu ở trên thành train và test set, với tỉ lệ 85%, 15% tương đương với 786 ảnh dành cho việc huấn luyện và 139 ảnh cho việc kiểm thử.

Tiếp đến, tôi áp dụng mô hình CNN và DNN để trích xuất đặc trưng và tiến hành huấn luyện phân loại mã độc.
Mô hình được xây dựng theo cấu trúc sau:

1. Lớp tích chập (Conv2D): Sử dụng 32 bộ lọc có kích thước 3x3, hàm kích hoạt ReLU và padding='same'. Đầu vào có kích thước (256, 256, 2), với hai kênh màu.
2. Lớp pooling tối đa (MaxPooling2D): Sử dụng cửa sổ pooling có kích thước 2x2.
3. Lớp tích chập (Conv2D): Sử dụng 64 bộ lọc có kích thước 3x3, hàm kích hoạt ReLU và padding='same'.
4. Lớp pooling tối đa (MaxPooling2D): Sử dụng cửa sổ pooling có kích thước 2x2.
5. Lớp dropout (Dropout): Áp dụng dropout với tỷ lệ 0.425 để giảm overfitting.
6. Lớp flatten (Flatten): Chuyển đổi đầu ra của lớp trước thành một vector 1 chiều.
7. Lớp kết nối đầy đủ (Dense): Sử dụng 128 nơ-ron và hàm kích hoạt ReLU.

8. Lớp dropout (Dropout): Áp dụng dropout với tỷ lệ 0.425 để giảm overfitting.
9. Lớp đầu ra (Output layer): Sử dụng 2 nơ-ron và hàm kích hoạt softmax để dự đoán xác suất của mỗi lớp (mã độc và không độc hại).

Cũng như hai phần trên, mô hình này được biên dịch với bộ tối ưu hóa 'adam', hàm mất mát là Sparse Categorical Crossentropy và metric là 'accuracy'.

2.3.4. Áp dụng CNN và SVM

Sau khi huấn luyện mô hình kết hợp CNN và DNN, tôi sử dụng lớp Model của thư viện Keras để trích xuất đặc trưng từ mô hình đã được huấn luyện. Đầu vào của lớp Model là ảnh với kích thước tương tự như mạng CNN. Tôi chỉ quan tâm đến kết quả sau lớp Dense đầu tiên của mô hình.

Tiếp theo, sử dụng mô hình đã trích xuất đặc trưng thông qua mạng CNN để tạo dữ liệu đầu vào cho mô hình SVM. Dữ liệu đầu vào vẫn là tệp huấn luyện ban đầu. Tôi sử dụng GridSearchCV để tìm kiếm siêu tham số tốt nhất cho mô hình SVM, sau đó thực hiện huấn luyện và kiểm tra mô hình SVM với siêu tham số được tối ưu hóa.

3. THỰC NGHIỆM VÀ KẾT QUẢ

3.1. Phương pháp đánh giá

Trong lĩnh vực nhận biết mã độc, việc đánh giá chính xác mô hình phân loại là điều cực kỳ quan trọng. Có nhiều cách để đánh giá mô hình, nhưng một trong những phương pháp phổ biến nhất là sử dụng các chỉ số Accuracy, Precision, Recall và F1-score. Đây là các chỉ số quan trọng giúp ta hiểu rõ khả năng hoạt động của mô hình.

Độ chính xác (Accuracy): Accuracy đo lường tỷ lệ của tất cả các dự đoán chính xác (tính cả dự đoán đúng cho cả lớp mã độc và lớp không mã độc) trên tổng số dự đoán. Accuracy được tính bằng công thức:

Accuracy = (TP + TN) / (TP + TN + FP + FN)

Trong đó:

TP (True Positive) là số điểm đúng được phân loại là mã độc.
TN (True Negative) là số điểm đúng được phân loại không phải mã độc.
FP (False Positive) là số điểm sai được phân loại là mã độc.
FN (False Negative) là số điểm sai được phân loại không phải mã độc.

Độ chính xác (Precision): Precision đo lường tỷ lệ của các dự đoán đúng cho lớp mã độc trên tổng số dự đoán cho lớp mã độc. Precision được tính bằng công thức:

Precision = TP / (TP + FP)

Precision giúp đánh giá khả năng tránh "báo động giả", tức là giảm thiểu số lượng các tệp an toàn bị phân loại nhầm thành mã độc.

Độ phủ (Recall): Recall đo lường tỷ lệ của các dự đoán đúng cho lớp mã độc trên tổng số tệp thực sự là mã độc. Recall được tính bằng công thức:

Recall = TP / (TP + FN)

Recall cho thấy khả năng của mô hình trong việc phát hiện ra tất cả các tệp tin là mã độc trong tập dữ liệu.

Điểm F1 (F1-score): F1-score là trung bình điều hòa của Precision và Recall. Điểm này giúp đánh giá cân nhắc giữa Precision và Recall, và đặc biệt hữu ích khi có sự mất cân đối giữa các lớp trong tập dữ liệu. F1-score được tính bằng công thức:

F1-score = 2 * (Precision * Recall) / (Precision + Recall)

Với việc sử dụng tổng hợp các chỉ số trên, chúng ta có thể đánh giá một cách toàn diện khả năng của mô hình phân loại trong việc nhận biết mã độc.

3.2. Kết quả mô hình trên tệp Dex

Kết quả huấn luyện mô hình nhận biết mã độc cho thấy những tiến bộ đáng kể. Cụ thể, chúng ta đã huấn luyện và đánh giá hai mô hình khác nhau: một dựa trên Mạng nơ-ron tích chập (CNN) và một dựa trên Support Vector Machine (SVM).

Mô hình Dex CNN đã đạt được Độ chính xác (Accuracy) là 0.892473, Độ chính xác (Precision) là 0.893488, Độ phủ (Recall) là 0.891667 và Điểm F1 (F1-score) là 0.892161. Những chỉ số này cho thấy mô hình có hiệu suất tốt trong việc phân loại

mã độc, với khả năng phát hiện và tránh phân loại nhầm tệp tin an toàn tốt.

Mô hình Dex SVM cũng đã cho thấy kết quả khả quan, với Độ chính xác (Accuracy) là 0.881720, Độ chính xác (Precision) là 0.881957, Độ phủ (Recall) là 0.881250 và Điểm F1 (F1-score) là 0.881501. Mặc dù không cao như mô hình Dex cnn, nhưng mô hình Dex svm vẫn đảm bảo hiệu suất phân loại tốt.

Biểu đồ về Độ chính xác (Accuracy) và Mất mát (Loss) khi huấn luyện mạng CNN sẽ được đề cập ở Figure 4.1

3.3. Kết quả mô hình trên tệp Manifest

Đối với hai mô hình Manifest CNNvà Manifest SVM, chúng ta cũng thấy rõ sự khác biệt về hiệu suất.

Mô hình Manifest CNN đã đạt được Độ chính xác (Accuracy) là 0.709677, Độ chính xác (Precision) là 0.722249, Độ phủ (Recall) là 0.713194 và Điểm F1 (F1-score) là 0.707513. Những chỉ số này cho thấy mô hình này có hiệu suất trung bình trong việc phân loại mã độc, với khả năng phát hiện và tránh phân loại nhầm tệp tin an toàn khá tốt.

Trong khi đó, mô hình Manifest SVM đã đạt được Độ chính xác (Accuracy) là 0.763441, Độ chính xác (Precision) là 0.763194, Độ phủ (Recall) là 0.763194 và Điểm F1 (F1-score) là 0.763194. Điều này cho thấy mô hình này có hiệu suất cao hơn Manifest cnn trong việc nhận biết mã độc. Đặc biệt, chỉ số F1-score cao cho thấy mô hình này cân nhắc tốt giữa Độ chính xác và Độ phủ, vì vậy nó có khả năng làm việc tốt trong tình huống có sự mất cân đối trong tập dữ liệu.

Biểu đồ về Độ chính xác (Accuracy) và Mất mát (Loss) khi huấn luyện mạng CNN sẽ được đề cập ở Figure 4.2

3.4. Kết quả mô hình trên tệp Full

Với hai mô hình tiếp theo, Full CNN và Full SVM, chúng ta cũng thấy được những kết quả đánh giá khá ấn tượng.

Mô hình Full CNN đã đạt được Độ chính xác (Accuracy) là 0.817204, Độ chính xác (Precision) là 0.816095, Độ phủ (Recall) là 0.816977 và Điểm F1 (F1-score) là 0.816440. Các chỉ số này cho thấy mô hình có hiệu suất tốt trong việc phân loại mã độc, với khả năng phát hiện và tránh phân loại nhầm tệp tin an toàn đáng kể.

Trong khi đó, mô hình Full SVM đã đạt được Độ chính xác (Accuracy) là 0.817204, Độ chính xác (Precision) là 0.818160, Độ phủ (Recall) là 0.813721 và Điểm F1 (F1-score) là 0.815066. Những chỉ số này cho thấy mô hình có hiệu suất tương đương với mô hình Full cnn trong việc nhận biết mã độc.

Nhìn chung, cả hai mô hình Full CNN và Full SVM đều cho thấy hiệu suất cao trong việc nhận biết mã độc. Điều này cho thấy tiềm năng của việc sử dụng các mô hình học máy phức tạp hơn, như Mạng nơ-ron tích chập (CNN) và Support Vector Machine (SVM), trong việc nhận biết và phân loại mã độc.

Biểu đồ về Độ chính xác (Accuracy) và Mất mát (Loss) khi huấn luyện mạng CNN sẽ được đề cập ở Figure 4.3

3.5. So sánh kết quả

Các mô hình nhận biết mã độc đã được huấn luyện và đánh giá, kết quả cho thấy một số khác biệt đáng chú ý về hiệu suất giữa chúng.

Mô hình Dex CNN đạt hiệu suất tốt nhất với Accuracy, Precision, Recall, và F1-score đều cao, lần lượt là 0.892473, 0.893488, 0.891667, và 0.892161. Điều này cho thấy mô hình này có khả năng phát hiện và phân loại chính xác các mã độc từ tập dữ liệu.

Mô hình Dex SVM, mặc dù không cao bằng Dex CNN, nhưng cũng cho thấy kết quả khả quan, với các chỉ số tương ứng là 0.881720, 0.881957, 0.881250, và 0.881501.

Cả Manifest CNN và Manifest SVM đều cho thấy hiệu suất thấp hơn so với các mô hình Dex. Tuy nhiên, Manifest SVM với Accuracy, Precision, Recall, và F1-score là 0.763441, 0.763194, 0.763194, và 0.763194, có hiệu suất cao hơn Manifest CNN, đặc biệt trong các trường hợp mất cân đối dữ liệu.

Cuối cùng, Full CNN và Full SVM cho thấy hiệu suất tốt, gần như tương đương nhau, với các chỉ số đánh giá không đáng kể chênh lệch. Chúng đều cho thấy kết quả đáng kể trong việc nhận biết và phân loại mã độc.

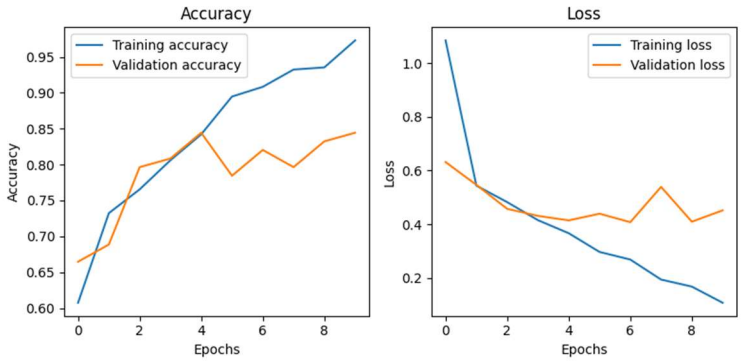

Figure 5.1. Biểu đồ về loss và acc trong quá trình huấn luyện trên tệp Dex

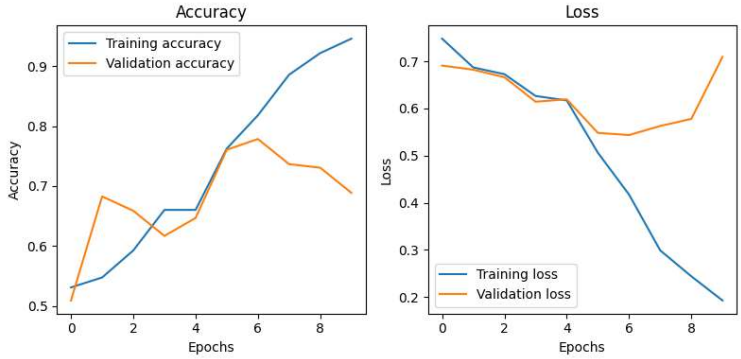

Figure 4.2. Biểu đồ về loss và acc trong quá trình huấn luyện trên tệp Manifest

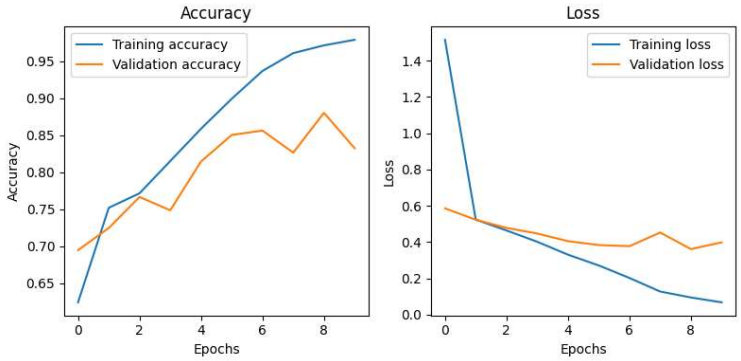

Figure 4.3. Biểu đồ về loss và acc trong quá trình huấn luyện trên tệp Full

Tóm lại, mỗi mô hình có những điểm mạnh và điểm yếu riêng, và chọn lựa giữa chúng đôi khi phụ thuộc vào các yếu tố khác nhau như tập dữ liệu, ngữ cảnh sử dụng, và khả năng tính toán. Những kết quả này cung cấp một cái nhìn sâu sắc về hiệu suất và tiềm năng của các mô hình học máy trong việc nhận biết mã độc.

Bảng đánh giá điểm số cụ thể được để cập ở Bảng 2.

4. TỔNG KẾT

Từ những kết quả đã nêu, ta có thể rút ra rằng việc biến đổi file APK thành hình ảnh và sử dụng các

mô hình học máy như CNN và SVM cho việc nhận biết mã độc là một cách tiếp cận hiệu quả.

Các mô hình đã được huấn luyện và đánh giá trên cùng một tập dữ liệu, và đã cho thấy hiệu suất đáng kể trong việc phát hiện và phân loại mã độc. Đặc biệt, mô hình Dex cnn và Dex svm đã cho thấy hiệu suất vượt trội, trong khi các mô hình khác cũng đạt được kết quả khả quan.

Điều này cho thấy rằng việc biến đổi file APK thành hình ảnh, kết hợp với việc sử dụng các mô hình học máy phức tạp, có thể cung cấp một giải pháp mạnh mẽ và hiệu quả để nhận biết mã độc.

Tuy nhiên, cũng cần lưu ý rằng, mặc dù các kết quả này khả quan, nhưng vẫn còn nhiều thách thức phải đối mặt, bao gồm việc cải thiện hiệu suất đối với những tập dữ liệu lớn hơn và phức tạp hơn, cũng như việc tiếp tục tinh chỉnh và cải thiện các mô hình hiện tại.

Cuối cùng, việc biến đổi file APK thành hình ảnh cho việc nhận biết mã độc cung cấp một hướng đi mới cho lĩnh vực bảo mật mạng, và nó cho thấy rằng sự tiến bộ trong lĩnh vực học máy có thể giúp đối phó với những vấn đề bảo mật ngày càng phức tạp.

Loại mã độc	Mô tả	Số lượng mẫu
Adware	Chương trình không mong muốn này hoạt động bằng cách hiển thị liên tục các quảng cáo pop-up trên màn hình di động để tạo doanh thu cho tác giả của nó.	104
Ransomware	Phần mềm độc hại này mã hóa các tệp của nạn nhân và đòi tiền chuộc để khôi phục quyền truy cập.	101
Scareware	Đây là một hình thức phần mềm độc hại sử dụng kỹ thuật xã hội để lừa đảo nạn nhân mua phần mềm độc hại.	112
SMSmalware	Phần mềm độc hại này sử dụng dịch vụ tin nhắn ngắn và các dịch vụ tin nhắn di động khác để khai thác các thiết bị di động. Nó gửi tin nhắn SMS độc hại và chặn các tin nhắn SMS để đánh cắp mật khẩu, chi tiết ngân hàng trực tuyến và thông tin khác từ các thiết bị bị nhiễm.	109
Benign	Đây là các ứng dụng hợp lệ đã được quét bằng VirusTotal để xác nhận rằng chúng không phải là phần mềm độc hại.	499
Tổng		925

Table 1. CICMalAnal2017 dataset.

	Accuracy	**Precision**	**Recall**	**F1-score**
Dex CNN	0.892473	0.893488	0.891667	0.892161
Dex SVM	0.881720	0.881957	0.881250	0.881501
Manifest CNN	0.709677	0.722249	0.713194	0.707513
Manifest SVM	0.763441	0.763194	0.763194	0.763194
Full CNN	0.817204	0.816095	0.816977	0.816440
Full SVM	0.817204	0.818160	0.813721	0.815066

Table 2. Bảng kết quả

REFERENCES

Tajuddin Manhar Mohammed, Lakshmanan Nataraj, Satish Chikkagoudar, Shivkumar Chandrasekaran, B.S. Manjunath.
Malware Detection Using Frequency Domain-Based Image Visualization and Deep Learning
Halil Murat Ünver & Khaled Bakour
Android malware detection based on image-based features and machine learning techniques