| HỌC VIỆN KỸ THUẬT MẬT MÃ  **KHOA ATTT**  ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯ |
| --- |
| **CHUYÊN ĐỀ CƠ SỞ**  **ỨNG DỤNG MÔ HÌNH HỌC SÂU CHO BÀI TOÁN PHÁT HIỆN MÃ ĐỘC TRÊN ANDROID** |
| *Nhóm sinh viên thực hiện*:  **Nguyễn Khánh Linh - AT180230 - AT18B**  **Phạm Quỳnh Anh - AT180504 - AT18E**  **Trần Xuân Phương - AT180538 - AT18E**    *Người hướng dẫn* :  **ThS. Thái Thị Thanh Vân**  Khoa Công nghệ thông tin – Học viện Kỹ thuật mật mã |
| Hà Nội, 2024 |

# MỤC LỤC

[**MỤC LỤC 1**](#_heading=h.qiqer5g5xyna)

[**DANH MỤC HÌNH VẼ 3**](#_heading=h.1jmdkvgtpa1c)

[**LỜI NÓI ĐẦU 4**](#_heading=h.30j0zll)

[**CHƯƠNG 1: TỔNG QUAN VỀ HỆ ĐIỀU HÀNH ANDROID VÀ MÃ ĐỘC 6**](#_heading=h.a13mh15a8z5)

[1.1. Tổng quan về hệ điều hành Android 6](#_heading=h.efg68eayij29)

[1.1.1. Giới thiệu về hệ điều hành Android 6](#_heading=h.sc72hi153e3u)

[1.1.2. File APK trên hệ điều hành Android 7](#_heading=h.pn43k83w71jz)

[1.2. Tổng quan về mã độc 8](#_heading=h.zctb5cmu2mns)

[1.3.1. Giới thiệu về mã độc 8](#_heading=h.co0oludtw9fg)

[1.3.2. Một số loại mã độc phổ biến 9](#_heading=h.5qebxqww1sdx)

[1.3.3. Phân biệt mã độc và mã lành 11](#_heading=h.m6q8elsmmu0)

[1.3. Các phương pháp phát hiện mã độc 12](#_heading=h.e3gg9jva0wxg)

[1.4.1. Phát hiện dựa trên chữ ký (signatures) 13](#_heading=h.u4cj8phyk1y5)

[1.4.2. Phát hiện dựa trên hành vi 13](#_heading=h.cgzbwei5siik)

[1.4.3. Phát hiện dựa trên kỹ thuật học máy 14](#_heading=h.ju58ru5l89fn)

[1.4. Kết luận Chương 1 15](#_heading=h.rs0cftgvph7b)

[**CHƯƠNG 2: TỔNG QUAN VỀ DEEP LEARNING 16**](#_heading=h.tdv6cr2cpv2r)

[2.1. Tổng quan về học máy 16](#_heading=h.3yyuhpne0rst)

[2.1.1. Giới thiệu về học máy 16](#_heading=h.4ljkykv5vni8)

[2.1.2. Quy trình làm việc của học máy 16](#_heading=h.w3ycf75m0jv6)

[2.1.3. Một số khái niệm cơ bản 18](#_heading=h.dtdg2rz16x6m)

[2.1.4. Các dạng học máy cơ bản 18](#_heading=h.f63rd1z89hfo)

[2.2. Tổng quan về học sâu 21](#_heading=h.h7a2zpv6zl0a)

[2.2.1. Giới thiệu về học sâu 21](#_heading=h.s89xdyxku3eo)

[2.2.2. Các kỹ thuật thường được sử dụng trong học sâu 22](#_heading=h.gmrcex4fppx)

[2.3. Tổng quan về mô hình CNN 24](#_heading=h.4q1t3e7l3imm)

[2.4.1. Lớp tích chập (Convolutional layer) 25](#_heading=h.59bnwlfm67hs)

[2.4.2. Lớp gộp (Pooling layer) 26](#_heading=h.pmnll3okxs23)

[2.4.3. Lớp kích hoạt (Activation layer): 27](#_heading=h.o4jivv47yym9)

[2.4.4. Lớp kết nối đầy đủ (Fully connected layer) 28](#_heading=h.32mmg5edyi2l)

[2.4. Tổng quan về mô hình LSTM 29](#_heading=h.t7rynqy4bm6n)

[2.5.1. Mô hình RNN 29](#_heading=h.f27w3q6ksqzf)

[2.5.2. Mô hình LSTM 31](#_heading=h.melbgttm84zw)

[2.5. Một số thư viện hỗ trợ AI 34](#_heading=h.kzyrnih6ip8x)

[2.6. Kết luận Chương 2 35](#_heading=h.3t3pqiigtai6)

[**CHƯƠNG 3: TRIỂN KHAI THỰC NGHIỆM 36**](#_heading=h.rt1tbiukzcdd)

[3.1. Chuẩn bị môi trường 36](#_heading=h.uwf1g1ojjl0o)

[3.2. Dữ liệu đầu vào 36](#_heading=h.v90cwq3sndft)

[3.2.1. Xử lý bộ dữ liệu thu thập được 36](#_heading=h.lw7swefis070)

[3.2.2. Xử lý file JSON 39](#_heading=h.3w00mzontyuy)

[3.3. Tiền xử lý 40](#_heading=h.ugqpro6cj5v1)

[3.4. Huấn luyện mô hình CNN 41](#_heading=h.kijdf4s4a2nq)

[3.4.1. Xác định kiến trúc mô hình 41](#_heading=h.vrurbk55qsql)

[3.4.2. Xây dựng mô hình 43](#_heading=h.fzkdffhjfzxg)

[3.4.3. Số liệu đánh giá 44](#_heading=h.42cohuoib2o1)

[3.5. Huấn luyện mô hình LSTM 51](#_heading=h.uoewxr2b4pk0)

[3.5.1. Xác định kiến trúc mô hình 51](#_heading=h.dr6d75cgvndu)

[3.5.2. Xây dựng mô hình 54](#_heading=h.n0is0v6nbtlp)

[3.5.3. Số liệu đánh giá 55](#_heading=h.4q6zzfgtuypu)

[3.6. Kết luận Chương 3 62](#_heading=h.quqi5z7n8k3g)

[**KẾT LUẬN 64**](#_heading=h.8vle0gvf8z5g)

[**TÀI LIỆU THAM KHẢO 65**](#_heading=h.1gm09vph6c40)

# 

# DANH MỤC HÌNH VẼ

***Hình 1.1. Cấu trúc tệp APK*****8**

***Hình 2.1. Quy trình làm việc của học máy*****17**

***Hình 2.2. Học máy có giám sát*****19**

***Hình 2.3. Học máy không có giám sát*****20**

***Hình 2.4. Học máy bán giám sát*****20**

***Hình 2.5. Cấu trúc của một mô hình CNN*****24**

***Hình 2.6. Mô hình CNN tổng quát*****25**

***Hình 2.7. Ma trận hình ảnh*****26**

***Hình 2.8. Cách thức hoạt động của Pooling layer*****27**

***Hình 2.9. Cách thức hoạt động của Flatten Function*****29**

***Hình 2.10. Fully connected layer*****29**

***Hình 2.11. Mô hình RNN*****30**

***Hình 2.12. Mô hình LSTM*****31**

***Hình 2.13 Tầng cổng quên (forget gate layer)*****32**

***Hình 2.14 Cập nhật giá trị cho ô trạng thái*** **32**

***Hình 2.15. Ô trạng thái mới*****33**

***Hình 2.16. Điều chỉnh thông tin ở đầu ra thông qua hàm tanh*****33**

***Hình 3.1. Các đặc trưng của file APK*****39**

***Hình 3.2. Biểu đồ độ chính xác của bộ dữ liệu CICMalDroid 2020*****46**

***Hình 3.3. Biểu đồ độ mất mát của bộ dữ liệu CICMalDroid 2020*****47**

***Hình 3.4. Biểu đồ độ chính xác của bộ dữ liệu Defense Droid*****48**

***Hình 3.5. Biểu đồ độ mất mát của bộ dữ liệu Defense Droid*** **49**

***Hình 3.6. Kết quả phân loại bộ dữ liệu CICMalDroid 2020*****50**

***Hình 3.7. Kết quả phân loại bộ dữ liệu Defense Droid*** **50**

***Hình 3.8. Biểu đồ độ chính xác của bộ dữ liệu CICMalDroid 2020*****57**

***Hình 3.9. Biểu đồ độ mất mát của bộ dữ liệu CICMalDroid 2020*****58**

***Hình 3.10. Biểu đồ độ chính xác của bộ dữ liệu Defense Droid*****59**

***Hình 3.11. Biểu đồ độ mất mát của bộ dữ liệu Defense Droid*** **60**

***Hình 3.12. Kết quả phân loại bộ dữ liệu Defense Droid*** **61**

***Hình 3.13. Kết quả phân loại bộ dữ liệu CICMalDroid 2020*****62**

# DANH MỤC TỪ VIẾT TẮT

| APK | Android Package Kit |
| --- | --- |
| XML | Extensible Markup Language |
| DDoS | Distributed Denial of Service |
| TCP | Transmission Control Protocol |
| UDP | User Datagram Protocol |
| HTML | HyperText Markup Language |
| AI | Artificial Intelligence |
| DNN | Deep Neural Network |
| ANN | Artificial Neural Network |
| RNN | Recurrent Neural Network |
| CNN | Convolutional Neural Network |
| CRNN | Convolutional Recurrent Neural Network |
| LSTM | Long Short-Term Memory |
| API | Application Programming Interface |
| CSV | Comma-Separated Values |
| JSON | JavaScript Object Notation |
| LR | Learning Rate |

# 

# LỜI NÓI ĐẦU

Cùng với sự phát triển của điện thoại di động, hệ điều hành Android đã trở thành một mảnh ghép không thể thiếu trong sự phát triển của công nghệ ngày nay. Không chỉ giới hạn ở điện thoại di động, Android còn mở rộng sang nhiều lĩnh vực công nghệ khác nhau như máy tính, đồng hồ thông minh, TV, thiết bị gia dụng,... Điều này tạo ra một hệ sinh thái đa dạng, linh hoạt và phổ biến.

Có thể nói rằng, Android không chỉ là một hệ điều hành, mà còn là một phần lối sống kỹ thuật của xã hội hiện đại. Với sự phát triển không ngừng, dễ dàng truy cập và khả năng tùy chỉnh cao, mức độ phổ biến của Android sẽ còn tăng mạnh trong tương lai.

Tuy nhiên, cùng với sự phát triển của Android trong xã hội, vấn đề bảo mật của hệ điều hành này cũng đặt ra nhiều vấn đề cấp thiết. Mặc dù đã có phát triển các biện pháp bảo mật riêng, nhưng có rất nhiều tin tặc vẫn có thể khai thác các lỗ hổng và tạo ra các mã độc gây ra những hậu quả to lớn vào hệ điều hành này.

Mã độc trong hệ điều hành Android có thể tiềm ẩn trong nhiều loại tập tin và ứng dụng, và chúng có thể gây ra các hậu quả nghiêm trọng từ việc đánh cắp tài sản, lộ thông tin riêng tư đến kiểm soát thiết bị từ xa. Điều này có thể gây ra những hậu quả to lớn đối với một số cá nhân, tổ chức thậm chí là toàn xã hội.

Điều này đặt ra một thách thức lớn đối với cộng đồng nghiên cứu bảo mật, rất nhiều giải pháp đã được đưa ra, trong đó có ứng dụng mô hình học sâu CNN trong phát hiện mã độc. CNN đã chứng tỏ khả năng xuất sắc trong việc nhận diện hình ảnh và mẫu dữ liệu phức tạp, và áp dụng chúng vào việc phát hiện mã độc có thể cung cấp một công cụ mạnh mẽ để bảo vệ hệ thống Android. Bằng cách huấn luyện mô hình với các dữ liệu đa dạng và thường xuyên cập nhật, có thể xây dựng các hệ thống phát hiện mã độc có khả năng phản ứng nhanh chóng và chính xác đối với các mối đe dọa mới.

Vì những lý do trên, nhóm em quyết định chọn đề tài “Nghiên cứu ứng dụng mô hình học sâu trong việc phát hiện mã độc trên hệ điều hành Android” làm đề tài nghiên cứu cho môn chuyên đề cơ sở. Nghiên cứu này không chỉ có ý nghĩa lý thuyết mà còn mang lại giá trị thực tiễn cao, góp phần bảo vệ thông tin cá nhân cho những người dùng Android trước các mối đe dọa liên tục phát triển từ các mã độc.

Nhóm chúng em 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 1: TỔNG QUAN VỀ HỆ ĐIỀU HÀNH ANDROID VÀ MÃ ĐỘC

## Tổng quan về hệ điều hành Android

### Giới thiệu về hệ điều hành Android

Android là một hệ điều hành dựa trên nền tảng Linux được thiết kế dành cho các thiết bị di động có màn hình cảm ứng như điện thoại thông minh và máy tính bảng. Ban đầu, Android được phát triển bởi Android Inc. với sự hỗ trợ tài chính từ Google và sau này được chính Google mua lại vào năm 2005.

Android có mã nguồn mở và Google phát hành mã nguồn theo Giấy phép Apache. Điều này đã cho phép các nhà phát triển thiết bị, mạng di động và các lập trình viên điều chỉnh và phân phối Android một cách tự do. Ngoài ra, Android còn có một cộng đồng lập trình viên đông đảo chuyên viết các ứng dụng để mở rộng chức năng của thiết bị, bằng ngôn ngữ lập trình Java có sửa đổi. Tháng 10 năm 2012, có khoảng 700.000 ứng dụng trên Android, và số lượt tải ứng dụng từ Google Play, cửa hàng ứng dụng chính của Android, ước tính khoảng 25 tỷ lượt.

Android chiếm 87,7% thị phần điện thoại thông minh trên toàn thế giới vào thời điểm quý 2 năm 2017, với tổng cộng 2 tỷ thiết bị đã được kích hoạt và 1,3 triệu lượt kích hoạt mỗi ngày. Sự thành công của hệ điều hành cũng khiến nó trở thành mục tiêu trong các vụ kiện liên quan đến bằng phát minh, góp mặt trong cái gọi là "cuộc chiến điện thoại thông minh" giữa các công ty công nghệ.

Từ năm 2011, Android đã là hệ điều hành bán chạy nhất trên toàn cầu trên điện thoại thông minh và từ năm 2013 trên máy tính bảng. Tính đến tháng 5 năm 2021, nó có hơn ba tỷ người dùng hàng tháng, là hệ điều hành có cài đặt nhiều nhất trên thế giới, và tính đến tháng 1 năm 2021, Cửa hàng Google Play có hơn 3 triệu ứng dụng. Android 13, được phát hành vào ngày 15 tháng 8 năm 2022, là phiên bản mới nhất,và phiên bản Android 12.1/12L mới nhất bao gồm những cải tiến đặc biệt cho điện thoại gập, máy tính bảng, màn hình có kích thước lớn như máy tính để bàn 1080p và Chromebook.

### File APK trên hệ điều hành Android

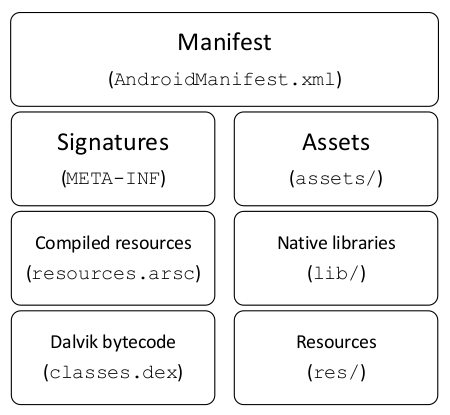
Tệp APK (Android Package Kit) là định dạng tệp dành cho các ứng dụng được sử dụng trên hệ điều hành Android. Tệp APK chứa tất cả dữ liệu mà ứng dụng cần, bao gồm tất cả mã, nội dung và tài nguyên của chương trình phần mềm. Tất cả các ứng dụng Android, bao gồm cả những ứng dụng được tải xuống từ Cửa hàng Google Play hoặc được tải xuống theo cách thủ công, đều sử dụng tệp APK.

Tệp APK là một loại định dạng lưu trữ, tương tự như tệp ZIP, chứa nhiều tệp và siêu dữ liệu trong đó. Tệp APK cũng là một biến thể của tệp Lưu trữ Java.

Người dùng Android thường không thấy tệp APK vì Android xử lý quá trình cài đặt ứng dụng thông qua Google Play. Tuy nhiên, các trang web khác có thể cung cấp tính năng tải xuống tệp APK trực tiếp để người dùng cài đặt ứng dụng theo cách thủ công.

Tệp APK bao gồm những thành phần sau:

* **META-INF:** Chứa thông tin xác minh và được tạo khi “ký” ứng dụng. Về cơ bản, đây là thông tin dấu vân tay cho mọi tệp có trong APK, nghĩa là bất kỳ sửa đổi nào đối với APK (thậm chí thay thế một biểu tượng) đều yêu cầu phải từ bỏ APK. Nếu không, hệ điều hành sẽ từ chối cài đặt. Khi ký ứng dụng, các tệp này sẽ được cập nhật. Thư mục này gồm hai thành phần:
* CERT.SF: Danh sách tất cả các tệp có hàm băm SHA-1 của chúng.
* CERT.RSA: Chứa nội dung đã ký của CERT.RF và được sử dụng để xác minh tính toàn vẹn của ứng dụng bằng khóa chung.
* **MANIFEST.MF:** Chứa SHA-256-Digest của tất cả các tệp trong gói ứng dụng. Nó được sử dụng để xác minh tính hợp lệ của từng tệp trong zip, đảm bảo rằng việc thay đổi tệp này hoặc bất kỳ tệp nào khác trong vùng chứa zip sẽ thu hồi chữ ký và làm cho nó không hợp lệ.
* **assets:** Chứa các nội dung ứng dụng và tệp tài nguyên đi kèm với ứng dụng.
* **AndroidManifest.xml:** Tệp kê khai Android bổ sung, mô tả tên, phiên bản, quyền truy cập, tệp thư viện được tham chiếu cho ứng dụng. Tệp này có thể ở dạng XML nhị phân của Android và có thể được chuyển đổi thành XML văn bản gốc mà con người có thể đọc được bằng các công cụ như AXMLPrinter2, Apktool M hoặc Androguard.
* **class.dex:** Các lớp được biên dịch ở định dạng tệp DEX và được thực thi bởi Android Runtime.
* **lib:** Một thư mục chứa các thư viện gốc (mã máy).
* **res:** thư mục chứa tài nguyên không được biên dịch thành resources.arsc
* **resources.arsc:** một file chứa các tài nguyên được biên dịch trước, chẳng hạn như XML nhị phân.



Hình 1.1. Cấu trúc tệp APK

## 

## Tổng quan về mã độc

### Giới thiệu về mã độc

Mã độc (Malware) viết tắt của cụm từ Malicious Software 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. Bất kỳ một phần mềm nào là lý do làm tổn thương, phá vỡ đến tính bí mật, tính toàn vẹn và tính sẵn sàng của dữ liệu người dùng, máy tính hoặc môi trường mạng đều có thể được xem như các mã độc.

### Một số loại mã độc phổ biến

**VIRUS**

* Virus là một loại mã độc hại phổ biến nhất và có khả năng tự nhân bản. Virus là một đoạn mã tự chèn vào ứng dụng và thực thi khi ứng dụng được chạy. Khi ở trong mạng, vi-rút có thể được sử dụng để đánh cắp dữ liệu nhạy cảm, khởi động các cuộc tấn công DDoS hoặc tiến hành các cuộc tấn công bằng ransomware.
* Một số loại tệp dễ bị nhiễm vi-rút hơn – .doc/docx, .exe, .html, .xls/.xlsx, .zip.

**WORM**

* 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.
* Worm nhắm vào các lỗ hổng trong hệ điều hành để tự cài đặt vào mạng. Chúng có thể giành được quyền truy cập theo nhiều cách: thông qua các cửa sau được tích hợp trong phần mềm, thông qua các lỗ hổng phần mềm không chủ ý hoặc thông qua ổ đĩa flash. Khi đã có sẵn, sâu có thể được các tác nhân độc hại sử dụng để khởi động các cuộc tấn công DDoS, đánh cắp dữ liệu nhạy cảm hoặc tiến hành các cuộc tấn công bằng ransomware.

**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 ngụy trang dưới dạng mã hoặc phần mềm mong muốn. Sau khi được người dùng không nghi ngờ tải xuống, Trojan có thể chiếm quyền kiểm soát hệ thống của nạn nhân nhằm mục đích xấu. Trojan có thể ẩn trong trò chơi, ứng dụng hoặc thậm chí các bản vá phần mềm hoặc chúng có thể được nhúng trong các tệp đính kèm có trong email lừa đảo.

**TRAPDOOR**

* Trap Door còn được gọi là Back Door. Trap door được sử dụng bởi những nhà lập trình với mục đích dò lỗi, kiểm tra chương trình. Trong các cuộc tấn công trapdoor 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 được 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 vô) nên rất khó phát hiện ngay cả scan port.

**RANSOMWARE**

* Ransomware là phần mềm sử dụng mã hóa để vô hiệu hóa quyền truy cập của mục tiêu vào dữ liệu của nó cho đến khi trả tiền chuộc. Tổ chức nạn nhân sẽ không thể hoạt động một phần hoặc hoàn toàn cho đến khi trả tiền, nhưng không có gì đảm bảo rằng việc thanh toán sẽ mang lại khóa giải mã cần thiết hoặc khóa giải mã được cung cấp sẽ hoạt động bình thường.

**SPYWARE**

* Phần mềm gián điệp thu thập thông tin về hoạt động của người dùng mà họ không biết hoặc không đồng ý. Điều này có thể bao gồm mật khẩu, mã pin, thông tin thanh toán và tin nhắn phi cấu trúc. Sau đó, những kẻ tấn công bán dữ liệu của bạn cho các nhà quảng cáo hoặc công ty dữ liệu, lấy thông tin tài khoản ngân hàng hoặc đánh cắp danh tính cá nhân của người dùng.
* Phần mềm gián điệp thường được tải xuống dưới dạng gói phần mềm hoặc từ các trang chia sẻ tệp. Việc sử dụng phần mềm gián điệp không chỉ giới hạn ở trình duyệt trên máy tính để bàn: nó còn có thể hoạt động trong một ứng dụng quan trọng hoặc trên điện thoại di động.

**ADWARE**

* Phần mềm quảng cáo theo dõi hoạt động lướt web của người dùng để xác định quảng cáo nào sẽ phù hợp với họ. Mặc dù phần mềm quảng cáo tương tự như phần mềm gián điệp nhưng nó không cài đặt bất kỳ phần mềm nào trên máy tính của người dùng và cũng không ghi lại thao tác gõ phím.
* Mối nguy hiểm trong phần mềm quảng cáo là sự xâm phạm vào quyền riêng tư của người dùng, dữ liệu do phần mềm quảng cáo thu thập được đối chiếu với dữ liệu được thu thập, công khai hoặc bí mật, về hoạt động của người dùng ở nơi khác trên internet và được sử dụng để tạo hồ sơ của người đó, bao gồm cả bạn bè của họ. , những gì họ đã mua, những nơi họ đã đi du lịch và hơn thế nữa. Thông tin đó có thể được chia sẻ hoặc bán cho các nhà quảng cáo mà không có sự đồng ý của người dùng.

### Phân biệt mã độc và mã lành

|  | **Mã độc** | **Mã lành** |
| --- | --- | --- |
| **Giống nhau** | - **Cùng là mã máy tính:** Cả mã độc và mã lành đều là mã máy tính, có thể được viết bằng ngôn ngữ lập trình như C++, Python, Java, và nhiều ngôn ngữ khác.  - **Chạy trên hệ thống máy tính:** Cả hai loại mã này đều có thể chạy trên hệ thống máy tính hoặc thiết bị di động, thậm chí cả trên các thiết bị nhúng. | |
| **Khác nhau** | **Mục đích:** cung cấp tính năng, dịch vụ, giải pháp cho người dùng, ví dụ như ứng dụng văn phòng, trò chơi, hoặc trình duyệt web. | **Mục đích:** Gây hại hoặc lợi dụng hệ thống máy tính hoặc dữ liệu, như virus, spyware, ransomware, và trojan. |
| **Hành vi:** Thường có hành vi dự đoán được và nhằm mục đích cụ thể mà người dùng mong đợi, chẳng hạn như mở ứng dụng và thực hiện các chức năng của nó. | **Hành vi:** Thường có hành vi không mong đợi và có thể gây hại cho hệ thống hoặc dữ liệu, như truy cập trái phép, lấy cắp thông tin, hoặc gây hỏng dữ liệu. |
| **Bảo mật:** Thường được kiểm tra và chứng nhận an toàn trước khi được phát hành để đảm bảo tính an toàn và tin cậy cho người dùng. | **Bảo mật:** Thường được phát hiện và loại bỏ bởi phần mềm chống vi-rút hoặc phần mềm bảo mật để bảo vệ hệ thống và dữ liệu của người dùng. |

## Các phương pháp phát hiện mã độc

Phân tích mã độc là việc sử dụng các công cụ, kỹ thuật phân tích nhằm xác định hành vi “độc hại” của mã độc đến hệ thống máy tình. Phân tích mã độc là có mục tiêu nữa là từ kết quả của việc phân tích đưa ra cách để gỡ bỏ và phòng tránh mã độc đó. Về phương pháp phân tích, có hai phương pháp phân tích mã độc là phân tích tĩnh và phân tích động.

* Phân tích tĩnh mô tả việc phân tích mã, cấu trúc của một phần mềm mà không cần thực thi chúng nhằm mục đích kiểm tra xem tập tin, phần mềm có phải mã độc hay không, và cố gắng xác định hành vi của mã độc. Phân tích tĩnh có thể chia làm hai mức độ: cơ bản và nâng cao.
* Phân tích động là việc phân tích các hành vi, chức năng của mã độc bằng cách thực thi phần mềm độc hại.

### Phát hiện dựa trên chữ ký (signatures)

Phát hiện mã độc dựa trên chữ ký là phương pháp dựa trên các thông tin về mã độc (signatures) đã được định nghĩa sẵn. Chữ ký ở đây có thể là chuỗi giá trị băm của tệp tin mã độc (MD5 hoặc SHA1), một chuỗi cố định nào đó trong tập tin hoặc metadata của tệp tin mã độc đó. Công việc thường thấy của các phát triển giải pháp anti-malware là khi họ xác định một đối tượng là độc hại, một tập tin là mã độc thì “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 antivirus 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.

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,…

### Phát hiện dựa trên hành vi

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.

### Phát hiện dựa trên kỹ thuật học máy

Phương pháp phát hiện mã độc dựa trên chữ ký chỉ phù hợp với các mã độc phổ biến, đã được phát hiện. Nó không thể phát hiện các mã độc đa hình – mã độc có khả năng thay đổi chữ ký, cũng như các mã độc mới chưa có chữ ký. Ngược lại, phương pháp phát hiện dựa trên heuristics có thể phát hiện các loại mã độc này, nhưng độ chính xác không cao, thường đưa ra kết quả dương tính giả và âm tính giả.

Trước sự phát triển nhanh chóng của các loại mã độc, việc phát triển một phương pháp phát hiện mới là rất cần thiết. Một trong các giải pháp là phát hiện dựa trên kỹ thuật học máy để tăng độ chính xác.

Học máy ứng dụng trong lĩnh vực mã độc để giải quyết hai bài toán lớn là phân loại mã độc và phát hiện mã độc:

* Phát hiện mã độc: là việc xác định xem một phần mềm có phải là mã độc hay không. Đây là bài toán phân loại tập tin với hai nhãn là mã độc và lành tính.
* Phân loại mã độc: mục tiêu là phát hiện sự tương đồng giữa các mẫu mã độc để xác định xem một mẫu mã độc thuộc họ mã độc nào đó. Có thể được chia làm bốn mục tiêu con là: phát hiện các biến thể, phát hiện họ mã độc, phát hiện sự tương đồng, phát hiện sự khác biệt.

## Kết luận Chương 1

Trong chương đầu tiên của báo cáo nhóm chúng em tập trung làm rõ các lý thuyết cơ bản về hệ điều hành Android và mã độc. Đối với hệ điều hành Android, nhóm chúng em tập trung vào các thành phần của file APK (dạng file đặc trưng nhất của hệ điều hành Android). Về mã độc, chúng em đã nêu lên 6 loại mã độc cơ bản, cách phân biệt mã độc và mã lành, và đặc biệt là một số phương pháp để người dùng có thể phát hiện được mã độc.

## 

# CHƯƠNG 2: TỔNG QUAN VỀ DEEP LEARNING

## Tổng quan về học máy

### Giới thiệu về học máy

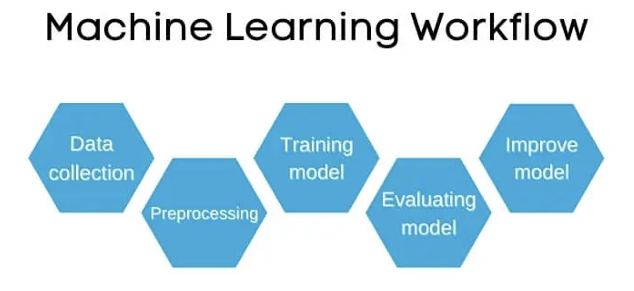
Học máy (Machine learning) là một lĩnh vực của trí tuệ nhân tạo (AI - artificial intelligence) và khoa học máy tính. Nó liên quan đến việc nghiên cứu và xây dựng các kỹ thuật cho phép các hệ thống tự động "học" từ các dữ liệu để giải quyết các vấn đề cụ thể dựa trên dữ liệu mẫu (training data) hoặc dựa vào kinh nghiệm (những gì đã được học). Học máy tập trung vào việc sử dụng dữ liệu và thuật toán để bắt chước cách con người học, tiếp đến dần cải thiện tính chính xác. 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…

### 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.



Hình 2.1. Quy trình làm việc của học máy

* 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.
* 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.
* 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.
* 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.
* 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.

### Một số khái niệm cơ bản

* Tập dữ liệu (Dataset): Tập dữ liệu nguyên thủy chưa được xử lý mà được thu thập trong bước thu thập dữ liệu. Một tập dữ liệu có thể chứa nhiều một hoặc nhiều data point.
* Điểm dữ liệu (Data point): Là một đơn vị thông tin độc lập trong tập dữ liệu. Chẳng hạn như một tập dữ liệu bao gồm thời gian xây dựng, diện tích nhà, giá nhà…. thì một Data point sẽ bao gồm các thông tin đó. Một tập hợp các điểm dữ liệu được gọi là Dataset.
* Training data, test data: Tập dữ liệu thường được chia thành hai tập hợp. Theo đó dữ liệu huấn luyện có nhiệm vụ huấn luyện trong mô hình Machine Learning, dữ liệu thử nghiệm để dự đoán kết quả và đánh giá mô hình.
* Features vector: Là một vectơ đặc trưng, mỗi vectơ chịu trách nhiệm biểu diễn một điểm dữ liệu trong tập dữ liệu. Mỗi vectơ có n chiều đại diện cho các tính năng của điểm dữ liệu, mỗi tính năng là một chiều và phải là số. Các mô hình chỉ có thể được đào tạo từ các vectơ đặc trưng này. Vì vậy tập dữ liệu cần được chuyển đổi thành một tập hợp các vectơ đặc trưng.
* Mô hình (Model): Đây là những mô hình thường được sử dụng để training các dữ liệu huấn luyện (training data) dựa trên thuật toán của mô hình. Và sau đó mô hình sẽ dự đoán hoặc đưa ra quyết định dựa trên những gì đã học được.

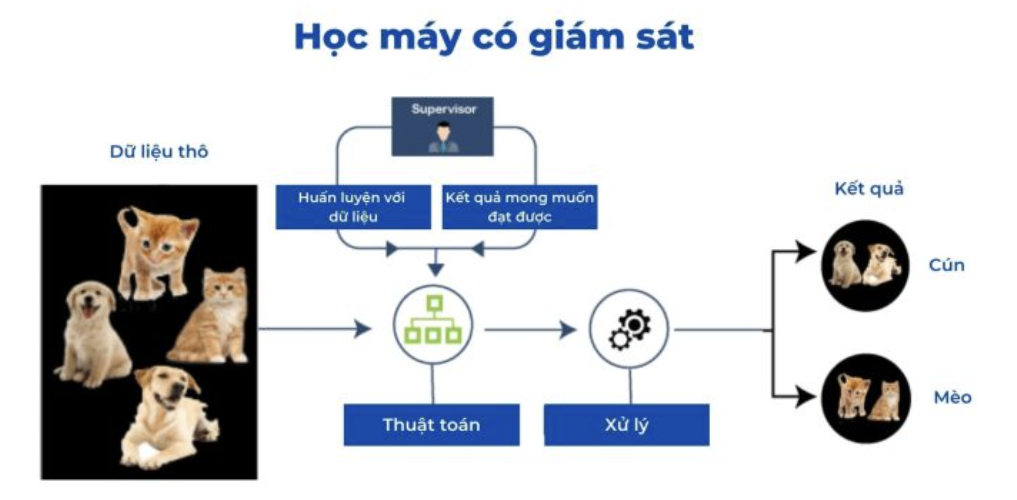
### Các dạng học máy cơ bản

**a, Học máy có giám sát**

Học máy có giám sát (Supervised Machine Learning) là phương pháp mà trong đó máy tính được học từ dữ liệu đã được đánh dấu trước, để phát triển các thuật toán có khả năng phân loại hoặc dự đoán kết quả một cách chính xác. Kỹ thuật này thường được áp dụng cho các bài toán phân lớp (Classification).

Ưu điểm của phương pháp học có giám sát là tính đơn giản và thiết kế dễ dàng. Cách học này rất hữu ích khi dự đoán một số lượng kết quả có giới hạn, phân loại dữ liệu hoặc kết hợp các kết quả thu được từ 2 thuật toán máy học khác.

Ví dụ: Với một tập dữ liệu gồm các hình ảnh của chó và mèo cho trước, trong giai đoạn đầu, máy tính sẽ được huấn luyện để nhận biết các hình ảnh này. Khi quá trình huấn luyện hoàn tất, máy tính có thể nhận diện và dự đoán chính xác loại động vật trong hình ảnh mới được cung cấp, dựa trên việc phân tích các đặc điểm như hình dạng và màu sắc. Quá trình này chính là nhận dạng đối tượng trong máy học có giám sát.

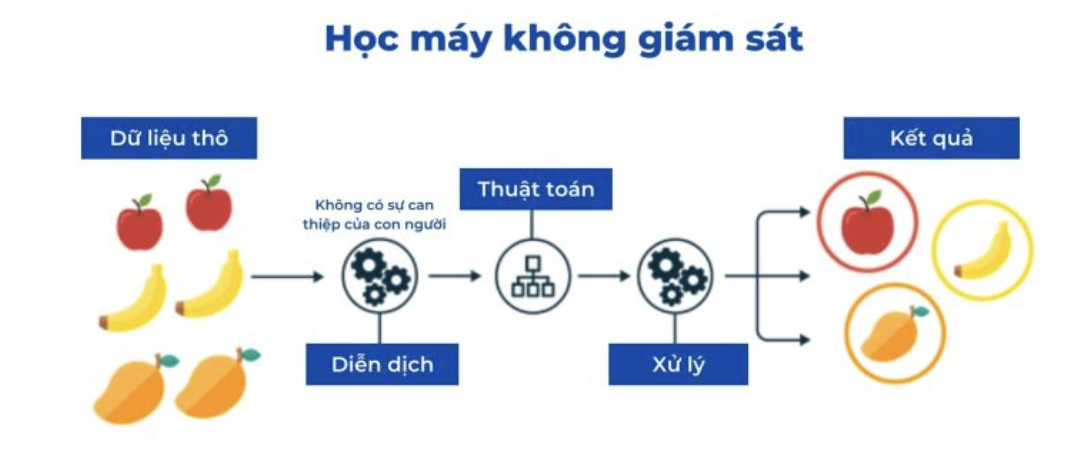


Hình 2.2. Học máy có giám sát

**b, Học máy không có giám sát**

Trái ngược với học máy có giám sát, máy học không giám sát (Unsupervised Learning) sử dụng thuật toán để phân tích và phân cụm các dữ liệu không có nhãn. Phương pháp này tự động tìm kiếm các mô hình và cấu trúc ẩn trong dữ liệu mà không yêu cầu sự hỗ trợ từ con người.

Ví dụ: Học máy không giám sát nổi bật với khả năng nhận diện các điểm chung và khác biệt trong dữ liệu, làm cho nó trở thành công cụ hữu ích trong việc phân tích dữ liệu khám phá, phân loại khách hàng, phát triển chiến lược bán chéo (Cross-sell), nhận dạng hình ảnh, và nhiều ứng dụng khác.



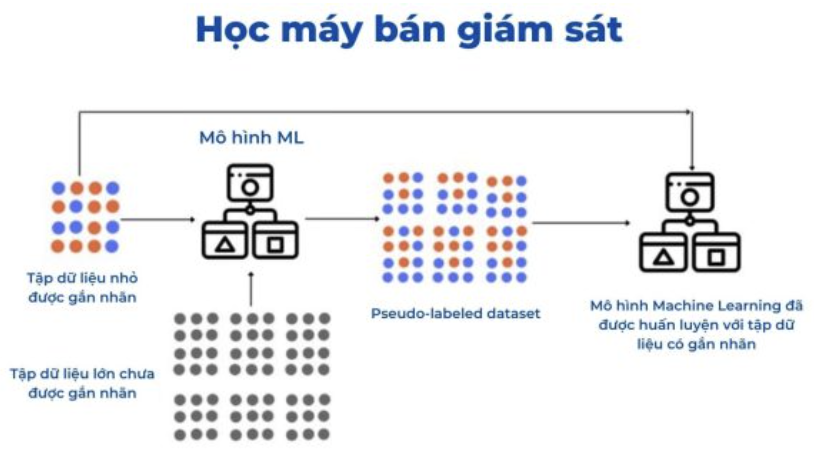
Hình 2.3. Học máy không có giám sát

**c, Học máy bán giám sát**

Trong bối cảnh dữ liệu phát triển nhanh chóng và không có cách nào để chúng được gắn nhãn kịp thời, đó là lý do học máy bán giám sát (Semi-supervised Learning) trở nên rất quan trọng. Phương pháp này kết hợp việc sử dụng dữ liệu đã được gắn nhãn và chưa gắn nhãn để huấn luyện máy tính.

Trong quá trình huấn luyện, con người sử dụng một tập dữ liệu có nhãn nhỏ hơn để chỉ dẫn máy tính cách phân loại và trích xuất từ một lượng lớn dữ liệu chưa được gắn nhãn. Semi-supervised Learning giúp giải quyết các vấn đề khi không có đủ dữ liệu gắn nhãn cho việc huấn luyện máy học có giám sát.

Ví dụ: Nó được ứng dụng rộng rãi trong các lĩnh vực như nhận diện giọng nói, phân loại nội dung web và tài liệu.



Hình 2.4. Học máy bán giám sát

## Tổng quan về học sâu

### Giới thiệu về học sâu

Học sâu là một lĩnh vực trong trí tuệ nhân tạo (AI) tập trung vào việc xây dựng và huấn luyện các mô hình máy học sâu, nhằm mục đích hiểu và giải quyết các vấn đề phức tạp. Học sâu dựa trên một loạt các thuật toán và mô hình mạng nơ-ron nhân tạo có khả năng học từ dữ liệu, giúp họ có thể tự động tìm ra các quy luật và đặc trưng từ dữ liệu đầu vào.

Một số đặc điểm chính về học sâu:

* Mạng nơ-ron sâu (DNN): Mạng nơ-ron sâu là một loại mạng nơ-ron nhân tạo (ANN) với nhiều lớp ẩn, cho phép mô hình học các đặc trưng phức tạp từ dữ liệu đầu vào.
* Học biểu diễn đa tầng (Deep Representation Learning): Học sâu tập trung vào việc học biểu diễn dữ liệu ở nhiều tầng, từ các biểu diễn cấp thấp đến các biểu diễn cấp cao, giúp mô hình tự động trích xuất và hiểu các đặc trưng phức tạp từ dữ liệu.
* Thuật toán học sâu: Các thuật toán học sâu bao gồm một loạt các phương pháp và mô hình như Mạng Nơ-ron Hồi Quy (RNN), Mạng Nơ-ron Dựa Trên Biến (CNN), Mạng Nơ-ron Tổ Hợp (DNN), Mạng Nơ-ron Tích Chập (CNN), và Mạng Nơ-ron Hồi Quy Tích Chập (CRNN), giúp mô hình có khả năng học và nhớ thông tin phức tạp.
* Huấn luyện sâu: Quá trình huấn luyện mô hình học sâu thường đòi hỏi một lượng lớn dữ liệu và tài nguyên tính toán. Các kỹ thuật như lan truyền ngược (backpropagation) và giảm thiểu hàm mất mát (loss function minimization) được sử dụng để điều chỉnh trọng số của mô hình dựa trên dữ liệu huấn luyện.
* Ứng dụng rộng rãi: Học sâu đã có ứng dụng trong nhiều lĩnh vực, bao gồm thị giác máy tính, xử lý ngôn ngữ tự nhiên, dò tìm vật thể, tự lái ô tô, dự báo thị trường tài chính, y học, và nhiều lĩnh vực khác.

Các hoạt động của học sâu

* Biểu diễn dữ liệu: Dữ liệu đầu vào được biểu diễn dưới dạng vector hoặc ma trận, thường là các giá trị số. Các biểu diễn này thường được chuẩn hóa để đảm bảo rằng mô hình có thể xử lý chúng.
* Feedforward (lan truyền tiến): Dữ liệu đầu vào được truyền qua các lớp của mạng nơ-ron, từ lớp đầu vào đến lớp đầu ra, thông qua các trọng số và hàm kích hoạt. Mỗi lớp tính toán đầu ra của nó bằng cách kết hợp đầu vào với các trọng số và áp dụng hàm kích hoạt.
* Tính toán đầu ra: Đầu ra của mạng được tính toán dựa trên dữ liệu đầu vào và các trọng số của mạng. Đối với các bài toán phân loại, đầu ra thường là xác suất của các lớp khác nhau. Đối với các bài toán hồi quy, đầu ra thường là một giá trị số.
* So sánh với đầu ra thực tế: Đầu ra của mạng được so sánh với đầu ra thực tế (đối với bài toán giám sát) hoặc với hàm mất mát (loss function) (đối với bài toán không giám sát). Sự khác biệt giữa đầu ra dự đoán và đầu ra thực tế được sử dụng để điều chỉnh các trọng số của mạng.
* Lan truyền ngược (backpropagation): Sự khác biệt giữa đầu ra dự đoán và đầu ra thực tế được lan truyền ngược từ lớp đầu ra đến lớp đầu vào của mạng. Các trọng số của mạng được điều chỉnh dựa trên gradient của hàm mất mát, để giảm thiểu sai số.
* Cập nhật trọng số: Các trọng số của mạng được cập nhật dựa trên gradient của hàm mất mát và một thuật toán tối ưu hóa như gradient descent. Quá trình lan truyền ngược và cập nhật trọng số được lặp lại nhiều lần thông qua tập dữ liệu huấn luyện.
* Đánh giá và điều chỉnh: Mô hình được đánh giá trên tập dữ liệu kiểm tra để đảm bảo rằng nó hoạt động tốt trên dữ liệu mới. Các tham số của mô hình có thể được điều chỉnh và điều chỉnh lại dựa trên kết quả đánh giá.

### Các kỹ thuật thường 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.

**a, 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.

**b, 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.

**c, Training from scratch**

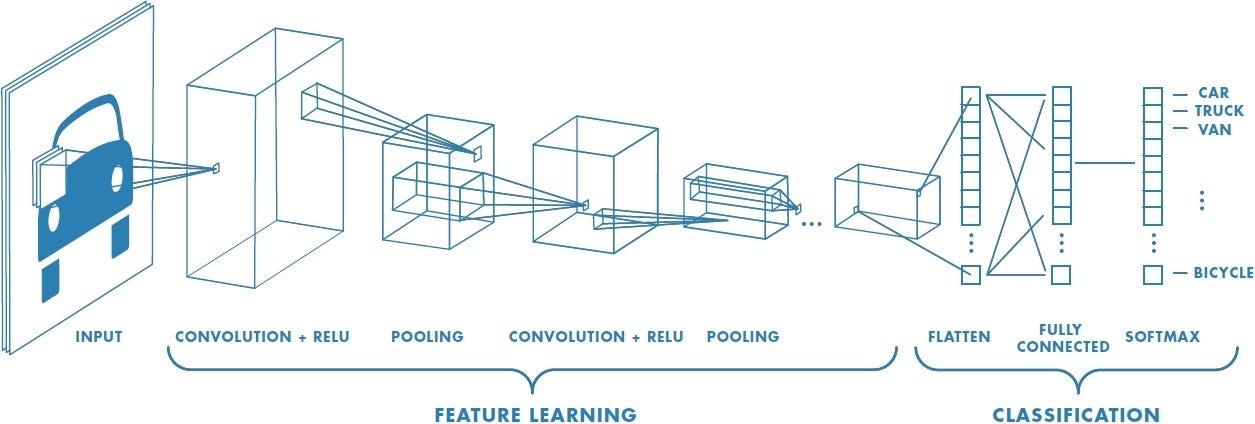
Phương pháp này yêu cầu nhà phát triển thu thập một tập dữ liệu có nhãn lớn và định cấu hình kiến ​​trúc mạng có thể tìm hiểu các tính năng và mô hình. Kỹ thuật này đặc biệt hữu ích cho các ứng dụng mới, cũng như các ứng dụng có số lượng lớn các danh mục đầu ra. Tuy nhiên, về tổng thể, đây là một cách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

**d, 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.

## Tổng quan về mô hình CNN

Mô hình học sâu CNN (Convolutional Neural Network) là một loại mạng nơ-ron nhân tạo được thiết kế đặc biệt để xử lý dữ liệu dạng lưới như hình ảnh. Đặc điểm chính của CNN là sự sử dụng các lớp tích chập (convolutional layers) để tự động học các đặc trưng từ dữ liệu đầu vào. Mỗi lớp tích chập sẽ áp dụng một số bộ lọc (filters) để trích xuất các đặc trưng ở mức độ khác nhau.

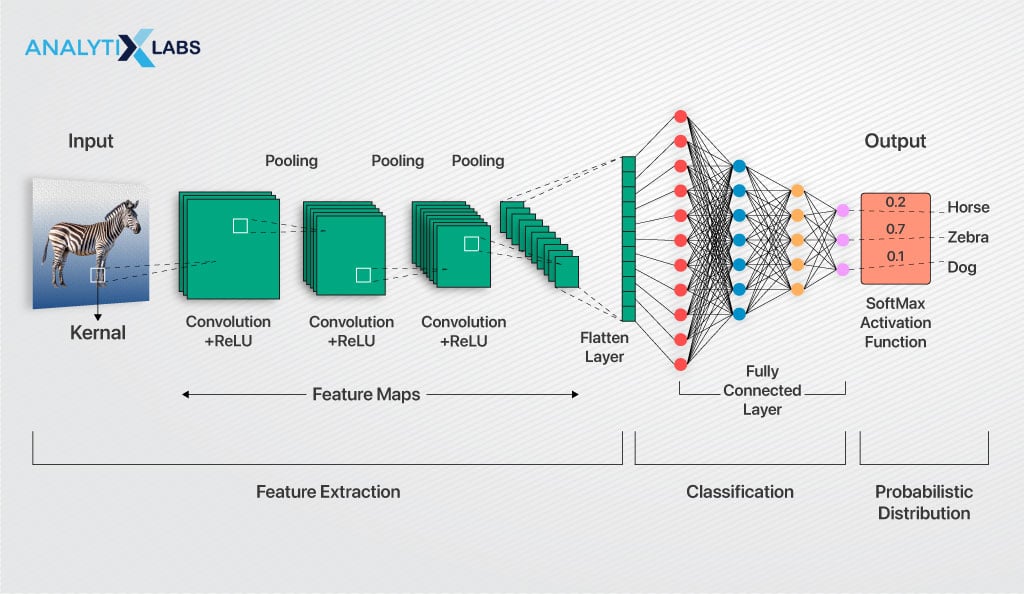


Hình 2.5. Cấu trúc của một mô hình CNN

Mô hình CNN được xem là một trong các model mạnh mẽ nhất trong các loại model của học học sâu. CNN thường được sử dụng cho các bài toán nhận dạng và phân loại hình ảnh, như nhận dạng khuôn mặt, nhận dạng chữ viết tay, phân loại đối tượng trong hình ảnh, và nhiều ứng dụng khác trong lĩnh vực thị giác máy tính.

Cấu trúc tổng quát của một mạng CNN bao gồm các phần quan trọng sau:

* Các lớp tích chập (Convolutional layer) và lớp gộp (Pooling layer) sẽ phân tích ảnh thành các đặc trưng (feature).
* Các lớp kết nối đầy đủ (Fully connected layer) đọc dữ liệu đầu ra từ các lớp Convolutional và lớp Pooling từ đó dự đoán nhãn phù hợp cho các bức ảnh.



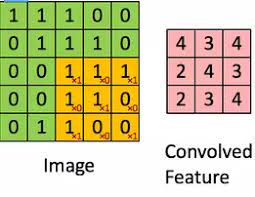
Hình 2.6. Mô hình CNN tổng quát

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 5×5 hoặc 3×3
* Pooling size: thường là 2×2 hoặc 4×4 cho ảnh đầu vào lớn
* Cách cuối cùng là thực hiện nhiều lần việc train test để chọn ra được param tốt nhất.

### Lớp tích chập (Convolutional layer)

Convolutional Layer (lớp tích chập) có một cửa sổ trượt (hay còn gọi là filter hoặc kernel). Cửa sổ này đã được cài đặt các tham số từ trước và có nhiệm vụ lấy ra những thông tin chính xác nhất, các tham số trong filter được nhân với từng phần từ trong image, từ đó ra được kết quả và lưu lại trong bảng Convolved Feature. Sau khi nhận và lưu kết quả vào bảng, cửa số filter sẽ dịch phải n bước (được cài đặt trước), rồi lại nhân tiếp các tham số với từng phần tử trong image mà cửa số filter vừa mới nhận được, lưu kết quả vào bảng Convolved Feature. Quá trình này cứ thế tiếp diễn cho đến khi cửa sổ filter quét hết image. Từ ảnh (5x5), với cửa sổ filter (3x3, bước di chuyển là 1), ta thu được bảng feature (3x3).



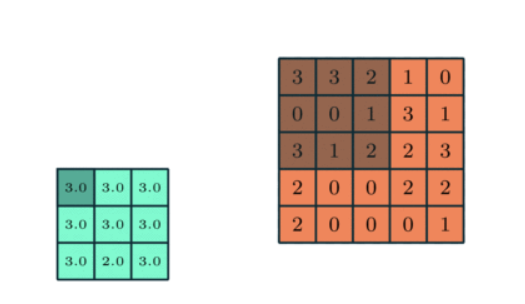
Hình 2.7. Ma trận hình ảnh

Tác dụng to lớn của Convolutional Layer là xem xem một feature nào đó có xuất hiện trong ảnh đầu vào hay không hơn là xem xem feature đó xuất hiện ở đâu trong ảnh. Tức là không quan tâm bức ảnh chụp con mèo ở dáng đứng, ngồi hay nằm, mà chỉ cần là con mèo thì model học sâu đều sẽ phát hiện được. Các lớp Convolutional Layer áp dụng các filter để từ input tạo ra feature map, từ đó có được đại diện cho những feature đầu vào.

### Lớp gộp (Pooling layer)

Sau các lớp Convolutional layer thì ta thu được feature map, các feature map này như một bản tóm lược vị trí các feature trong input. Tuy nhiên, điều này có nghĩa là chỉ một sự thay đổi nhỏ vị trí của các feature cũng gây ra sự sai khác rất lớn trong kết quả feature map nhận được. Sự thay đổi vị trí của input có thể thực hiện đơn giản bằng cách xoay ảnh, cắt ảnh, dịch ảnh... Những điều này đều gây ra sự sai khác trong kết quả feature map nhận được. Để giải quyết vấn đề này, ta dùng một lớp Pooling Layer.

Ảnh đầu vào sẽ được xử lý qua lớp Convolutional Layer, tiếp theo được xử lý qua ReLU function, sau đó sẽ xử lý bằng lớp Pooling Layer. Lớp Pooling Layer sẽ có nhiệm vụ đơn giản hóa thông tin đầu ra để giảm bớt số lượng nơron, giảm kích thước dữ liệu nhưng vẫn giữ được những thông tin quan trọng. Kích thước dữ liệu giảm sẽ giúp giảm bớt việc tỉnh toán trong model.



Hình 2.8. Cách thức hoạt động của Pooling layer

Có một số dạng Pooling Layer sau: Max-Pooling, Min-Pooling. Average-Pooling... Max-Pooling là một dạng pooling phổ biến, các loại pooling khác cũng có cách thức hoạt động tương tự. Gọi Pooling Layer có kích thước là KxK, đầu vào của lớp Pooling Layer là HxWxD, ta tách ra làm D ma trận H\*W. Với mỗi ma trận HxW, trên vùng K\*K trên ma trận đó ta tìm giá trị lớn nhất (maximum) của dữ liệu rồi viết vào bảng kết quả, sau đó dịch vùng Pooling Layer đi 1 đơn vị. Trong hình 2.5, sử dụng Pooling Layer có kích thước 3x3, bước dịch chuyển là 1 đơn vị trên kích thước ma trận là 5x5, kết quả thu được kích thước dữ liệu H, W đã giảm đi, còn kích thước D vẫn giữ nguyên.

Thông qua lớp Max-Pooling, số lượng nơron giảm đi. Chúng ta có thể thấy Max-Pooling là cách tìm ra trong các feature này thì feature nào là đặc trưng nhất. Trong một mạng CNN có nhiều feature map nên với mỗi feature map sẽ cho mỗi Max-Pooling khác nhau.

### Lớp kích hoạt (Activation layer):

Activation Function đóng vai trò là thành phần phi tuyến tại output của các nơron. Khi không sử dụng các hàm Activation Function này thì việc xếp chống các lớp nơron lên nhau là vô nghĩa do không có yếu tố phi tuyến. Có nhiều hàm được sử dụng để làm hàm Activation Function, tuy nhiên ở đây chúng em trình bày hai hàm mà đã áp dụng vào bài toán đó là ReLU và Softmax Function.

**a, ReLU**

ReLU (Rectified Linear Unit) được sử dụng như một hàm activation function, và cũng được sử dụng rộng rãi những năm gần đây vì tỉnh đơn giản của nó. ReLU có công thức rất đơn giản f(s) = max(0,s). Ưu điểm của ReLU là:

ReLU được chứng minh giúp cho việc training các Deep Network nhanh hơn rất nhiều. Sự tăng tốc này được cho là vì ReLU được tính toán gần như tức thời và gradient của nó cũng được tính cực nhanh với gradient bằng 1 nếu đầu vào lớn hơn 0, bằng 0 nếu đầu vào nhỏ hơn 0.

Mặc dù hàm ReLU không có đạo hàm tại s = 0, nhưng trong thực nghiệm, người ta vẫn thường định nghĩa ReLU'(0) = 0, và khẳng định thêm rằng, xác suất để input của một unit bằng 0 là rất nhỏ.

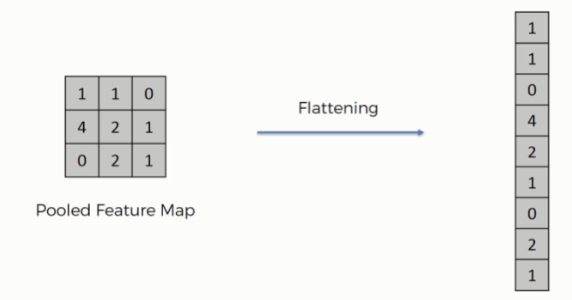
**b, Softmax Function**

Softmax Function được sử dụng như một hàm activation. Đây là một hàm toán học trả về xác suất có thể xảy ra của các phần tử trong mảng.

Hàm này được dùng để dự đoán xác suất phân phối các phần tử của mảng trong lớp đầu ra của mô hình CNN. Softmax Function được dùng để giải quyết vấn đề phân loại khi có nhiều nhăn. Hàm này có thể dược dùng làm hàm activation trong các hidden layer của mạng nơron nhưng thông thường chỉ được dùng trong lớp cuối cùng của model.

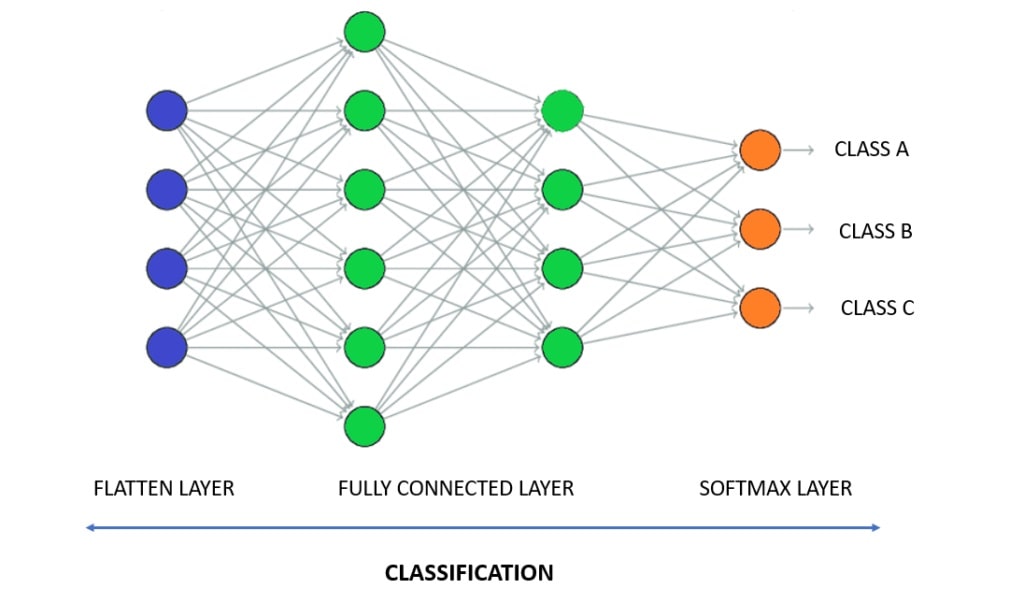
### Lớp kết nối đầy đủ (Fully connected layer)

Trước khi vào lớp Fully Connected Layer, các feature đã được “làm phẳng" bằng hàm Flatten. Hàm này sẽ biến đổi vector đầu ra của lớp Convolutional hoặc Pooling cuối cùng có kích thước NxM thành một vector 1x(NxM), vector này trở thành lớp đầu tiên của Fully Connected Layer.



Hình 2.9. Cách thức hoạt động của Flatten Function

Sở dĩ gọi là Fully Connected Layer vi mỗi node trong lớp đầu tiên đều kết nối với tất cả các node ở lớp thứ hai, từng node của lớp thứ hai lại kết nối với các node của lớp tiếp theo... Cứ như vậy và từ đó tạo nên mạng nơron.



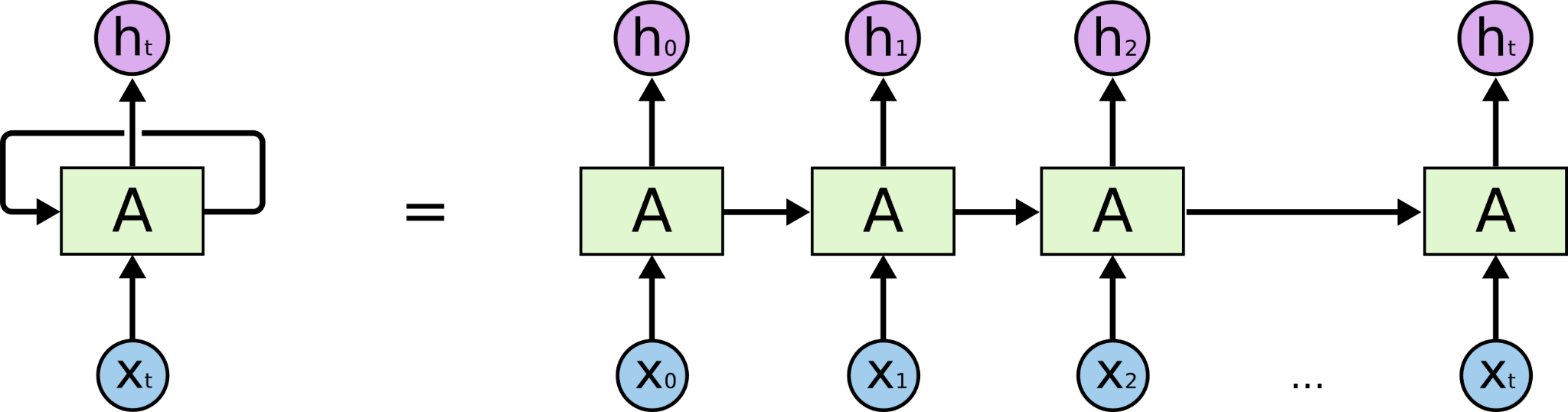
Hình 2.10. Fully connected layer

## Tổng quan về mô hình LSTM

### Mô hình RNN

Mạng nơ-ron hồi quy (RNN -Recurrent Neural Network) là một mô hình học sâu được đào tạo để xử lý và chuyển đổi đầu vào dữ liệu tuần tự thành đầu ra dữ liệu tuần tự cụ thể. Dữ liệu tuần tự là dữ liệu, chẳng hạn như từ, câu hoặc dữ liệu chuỗi thời gian, trong đó các thành phần tuần tự tương quan với nhau dựa trên ngữ nghĩa phức tạp và quy tắc cú pháp.

Phần “Recurrent” của tên RNN xuất phát từ thực tế là vòng lặp đầu vào và đầu ra. Sau khi đầu ra của mạng được tạo ra, đầu ra sẽ được sao chép và trả về mạng làm đầu vào. Khi đưa ra quyết định, không chỉ đầu vào và đầu ra hiện tại được phân tích mà đầu vào trước đó cũng được xem xét. Nói cách khác, nếu đầu vào ban đầu của mạng là X và đầu ra là H, thì cả H và X1 (đầu vào tiếp theo trong chuỗi dữ liệu) đều được đưa vào mạng cho vòng học tiếp theo. Bằng cách này, bối cảnh của dữ liệu (đầu vào trước đó) được giữ nguyên khi mạng huấn luyện.



Hình 2.11. Mô hình RNN

Một điểm nổi bật của RNN chính là ý tưởng kết nối các thông tin phía trước để dự đoán cho hiện tại. Việc này tương tự như việc sử dụng các cảnh trước của bộ phim để hiểu được cảnh hiện thời.

Để hiểu hơn về mô hình mạng RNN, ví dụ ta có câu: “các đám mây trên bầu trời” thì ta chỉ cần đọc tới “các đám mây trên bầu” là đủ biết được chữ tiếp theo là “trời” rồi. Trong tình huống này, khoảng cách tới thông tin có được cần để dự đoán là nhỏ, nên RNN hoàn toàn có thể học được.

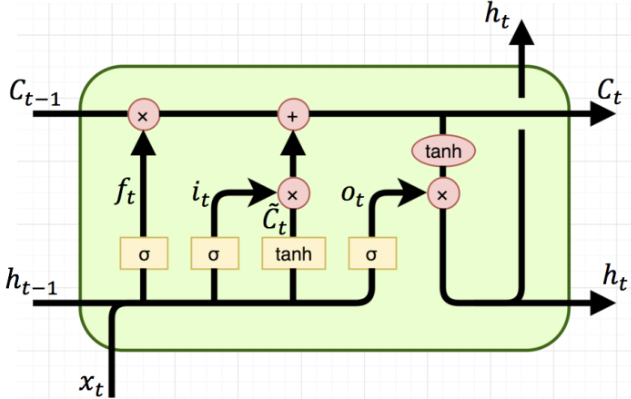
Tuy nhiên trong nhiều tình huống cần phải sử dụng nhiều ngữ cảnh hơn để có thể suy luận. Ví dụ như để dự đoán chữ “French” trong đoạn: “I grew up in France… I speak fluent French.”. Thông tin gần nhất “I speak fluent” cho biết đằng sau nó sẽ là tên của một ngôn ngữ nào đó nhưng không thể biết được đó là tiếng gì nếu như không có thêm ngữ cảnh “I grew up in France”. Khoảng cách giữa thông tin lúc này trở nên khá xa và mô hình RNN bắt đầu không thể nhớ và học được nữa .Những sự liên kết ngữ nghĩa dài như vậy được gọi là phụ thuộc dài hạn (long-term dependencies).

Về mặt lý thuyết, mô hình mạng RNN có thể giải quyết được những sự phụ thuộc dài hạn. Tuy nhiên trên thực tế mô hình RNN lại cho thấy khả năng học trong dài hạn kém hơn. Một trong những nguyên nhân chính được giải thích cho vấn đề này đó là sự triệt tiêu đạo hàm của hàm cost function sẽ diễn ra khi trải quả chuỗi dài các tính toán truy hồi. Một phiên bản mới của mạng RNN là mạng LSTM ra đời nhằm khắc phục hiện tường này nhờ một cơ chế đặc biệt.

### Mô hình LSTM

Mô hình học sâu bộ nhớ dài-ngắn (LSTM-Long Short Term Memory networks) là một dạng đặc biệt của mô hình LSTM được thiết kế để tránh được vấn đề phụ thuộc dài hạn (long-term dependency). Việc nhớ thông tin trong suốt thời gian dài là đặc tính mặc định của chúng, không cần phải huấn luyện nó để có thể nhớ được. Tức là nội tại của nó đã có thể ghi nhớ được mà không cần bất kỳ can thiệp nào.

**a, Cấu trúc bên trong của một tế bào LSTM**

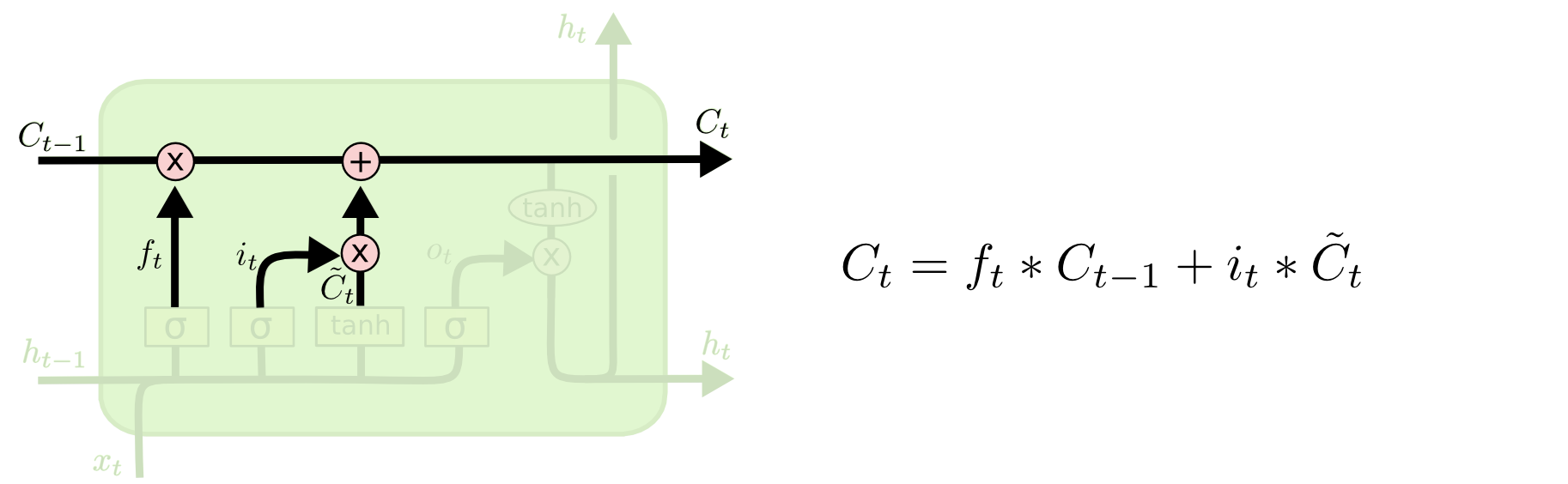
****

Hình 2.12. Mô hình LSTM

Trong đó:

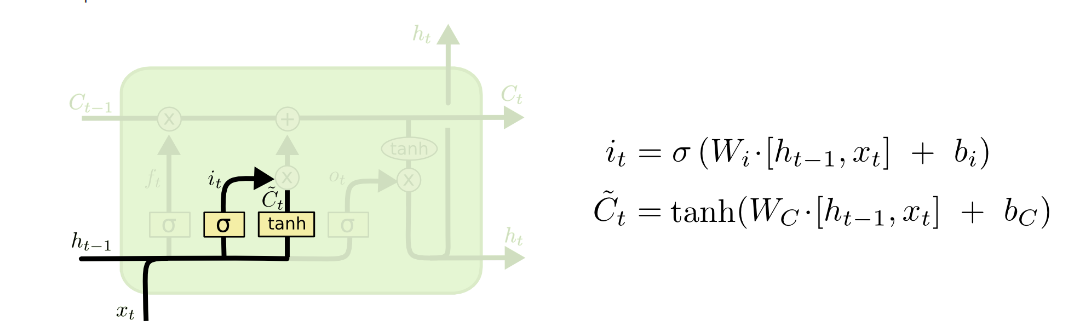
* Ct là trạng thái của Cell.
* ht là trạng thái ẩn.
* xt là ngõ vào của cell ở thời điểm.
* C(t-1) và h(t-1) là trạng thái của cell và trạng thái ẩn tại thời điểm t-1.
* ft là forget gate.
* it là input gate.
* ot là input gate.

Bước đầu tiên trong LSTM sẽ quyết định xem thông tin nào chúng ta sẽ cho phép đi qua ô trạng thái (cell state). Nó được kiểm soát bởi hàm sigmoid trong một tầng gọi là tầng quên (forget gate layer). Đầu tiên nó nhận đầu vào là 2 giá trị h(t-1) và xt và trả về một giá trị nằm trong khoảng 0 và 1 cho mỗi giá trị của ô trạng thái C(t-1). Nếu giá trị bằng 1 thể hiện ‘giữ toàn bộ thông tin’ và bằng 0 thể hiện ‘bỏ qua toàn bộ chúng’.



Hình 2.13. Tầng cổng quên (forget gate layer)

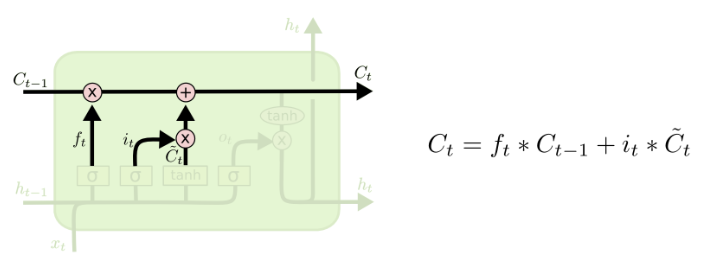
Bước tiếp theo chúng ta sẽ quyết định loại thông tin nào sẽ được lưu trữ trong ô trạng thái. Bước này bao gồm 2 phần. Phần đầu tiên là một tầng ẩn của hàm sigmoid được gọi là tầng cổng vào (input gate layer) quyết định giá trị bao nhiêu sẽ được cập nhật. Tiếp theo, tầng ẩn hàm tanh sẽ tạo ra một vectơ của một giá trị trạng thái mới mà có thể được thêm vào trạng thái. Tiếp theo kết hợp kết quả của 2 tầng này để tạo thành một cập nhật cho trạng thái.



Hình 2.14. Cập nhật giá trị cho ô trạng thái

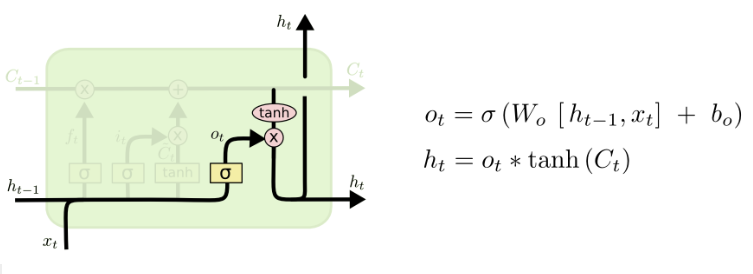
Đây là thời điểm để cập nhật một ô trạng thái cũ C(t-1) sang một trạng thái mới Ct. Những bước trước đó đã quyết định làm cái gì, và tại bước này chỉ cần thực hiện nó.

Nhân trạng thái cũ với ft tương ứng với việc quên những thứ quyết định được phép quên sớm. Phần tử đề cử it \* là một giá trị mới được tính toán tương ứng với bao nhiêu được cập nhật vào mỗi giá trị trạng thái.



Hình 2.15. Ô trạng thái mới

Cuối cùng cần quyết định xem đầu ra sẽ trả về bao nhiêu. Kết quả ở đầu ra sẽ dựa trên ô trạng thái, nhưng sẽ là một phiên bản được lọc. Đầu tiên, chúng ta chạy qua một tầng sigmoid nơi quyết định phần nào của ô trạng thái sẽ ở đầu ra. Sau đó, ô trạng thái được đưa qua hàm tanh (để chuyển giá trị về khoảng -1 và 1) và nhân nó với đầu ra của một cổng sigmoid, do đó chỉ trả ra phần mà chúng ta quyết định.



Hình 2.16. Điều chỉnh thông tin ở đầu ra thông qua hàm tanh

## Một số thư viện hỗ trợ AI

Một số thư viện phổ biến hỗ trợ trí tuệ nhân tạo (AI) và được sử dụng trong ứng dụng an toàn thông tin bao gồm:

* TensorFlow: Thư viện mã nguồn mở của Google được sử dụng rộng rãi cho học sâu và máy học.
* PyTorch: Một thư viện AI mã nguồn mở được phát triển bởi Facebook, cung cấp một cách linh hoạt và dễ dàng để xây dựng và huấn luyện mô hình học sâu.
* Keras: Một API mô hình hóa mức cao được viết bằng Python và chạy trên TensorFlow, Theano hoặc CNTK. Nó giúp việc xây dựng và thử nghiệm mô hình nhanh chóng và dễ dàng.
* Scikit-learn: Một thư viện mã nguồn mở cho machine learning và học máy trong Python, cung cấp các công cụ cho nhiều thuật toán học máy phổ biến.
* OpenCV: Thư viện mã nguồn mở chuyên dùng cho xử lý ảnh và thị giác máy tính, thường được sử dụng để phát hiện và nhận dạng đối tượng trong hình ảnh.
* Numpy và Pandas: Cả hai là thư viện Python phổ biến cho tính toán khoa học và xử lý dữ liệu, thường được sử dụng trong việc tiền xử lý dữ liệu cho các mô hình học máy.

Trong lĩnh vực an toàn thông tin và truy cập, các ứng dụng của AI bao gồm:

* Phát hiện xâm nhập (IDS/IPS): Sử dụng học máy để phát hiện các hành vi xâm nhập và tự động phản ứng để ngăn chặn chúng.
* Phân loại thư rác (Spam Filtering): Sử dụng các mô hình học máy để phân loại email thành thư rác và không phải thư rác.
* Phát hiện dựa trên hành vi (Behavior-based Detection): Sử dụng học máy để phát hiện các hành vi độc hại dựa trên dữ liệu về hành vi hoạt động của người dùng hoặc hệ thống.
* Xác thực và Phân loại (Authentication and Classification): Sử dụng học máy để xác thực người dùng và phân loại các yêu cầu truy cập vào hệ thống dựa trên hành vi và dữ liệu người dùng.

## Kết luận Chương 2

Để tiếp tục làm rõ những lý thuyết cần biết và làm tiền đề cho thực nghiệm, chương 2 của báo cáo giới thiệu về một số khái niệm học máy, học sâu và hai mô hình có thể áp dụng cùng một số thư viện AI có thể hỗ trợ cho đề tài. Về học máy, chúng em đã giới thiệu và giải thích các khái niệm cơ bản, quy trình hoạt động và một số dạng học máy cơ bản. Về học sâu, nhóm chúng em đã bắt đầu từ các khái niệm cơ bản, các phương thức thường được sử dụng trong học sâu. Sau đó tập trung phân tích hai dạng mô hình được sử dụng trong thực nghiệm CNN và LSTM.

# CHƯƠNG 3: TRIỂN KHAI THỰC NGHIỆM

## Chuẩn bị môi trường

Trong phần thực nghiệm này, nhóm chúng em sử dụng 2 bộ dữ liệu phần mềm độc hại. Bộ dữ liệu thứ nhất là bộ CICMalDroid 2020 do Đại học New Brunswick thu thập để làm dữ liệu cho thử nghiệm. Bộ dữ liệu này chứa 17.341 dữ liệu huấn luyện được gắn nhãn trải dài giữa năm danh mục riêng biệt: Adware, Banking malware, SMS malware, Riskware, và Benign. Ngoài ra, nhóm chúng em tiếp tục sử dụng bộ dữ liệu thứ hai là Defense Android để làm dữ liệu thử nghiệm. Bộ dữ liệu này chứa hơn 12.000 mẫu dữ liệu với hai nhãn đánh dấu là: 0 với mã lành và 1 với mã độc.

Ngoài ra, để trích xuất các tính năng tĩnh và động từ APK Android, nhóm chúng em sử dụng dụng công cụ AndroPyTool. đây là một công cụ kết hợp các công cụ phân tích ứng dụng Android nổi tiếng khác nhau như phân tích DroidBox, FlowDroid, Strace, AndroGuard hoặc VirusTotal và được cung cấp một thư mục nguồn chứa các tệp APK. AndroPyTool áp dụng tất cả các công cụ này để thực hiện phân tích tiền tĩnh, tĩnh và động, đồng thời tạo các tệp tính năng ở định dạng JSON và CSV và cho phép lưu tất cả dữ liệu trong cơ sở dữ liệu MongoDB.

## Dữ liệu đầu vào

### Xử lý bộ dữ liệu thu thập được

Với bộ dữ liệu đầu vào, để trích xuất các đặc trưng permission và API từ các file gốc, em sử dụng công cụ AndroPytool. Thực hiện các bước như sau

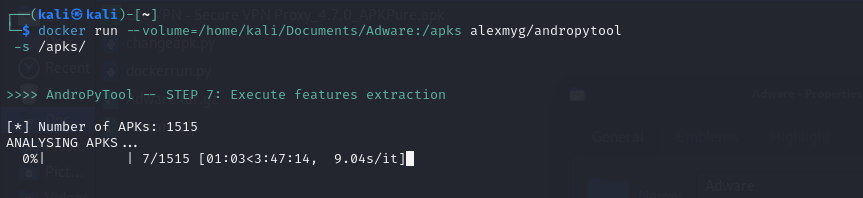
* Cài đặt Docker
* Kéo container từ Docker hub

| docker pull alexmyg/andropytool |
| --- |

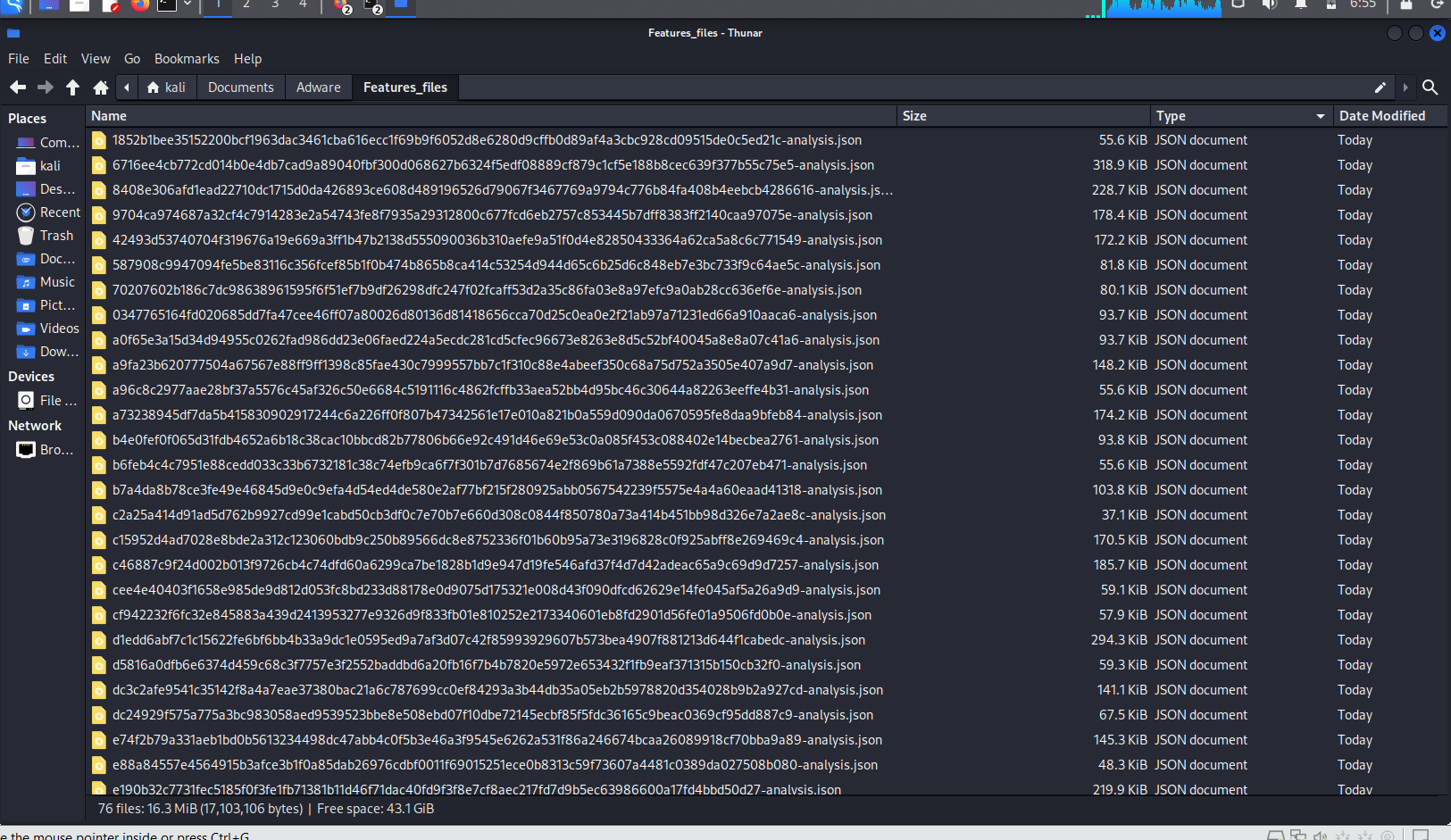
* Chạy AndroPyTool bằng cách chạy container

| docker run --volume=</PATH/TO/FOLDER/WITH/APKS/>:/apks alexmyg/andropytool -s /apks/ |
| --- |

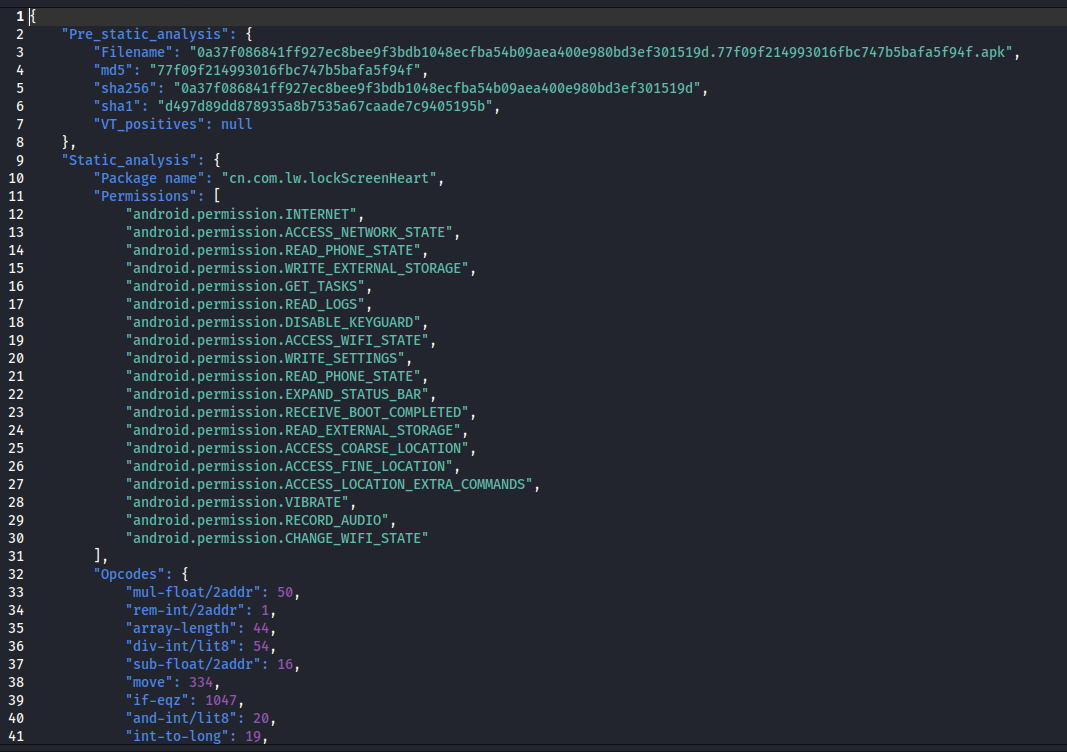
Với **“PATH/TO/FOLDER/WITH/APKS”** là đường dẫn của thư mục chứa file mã độc.



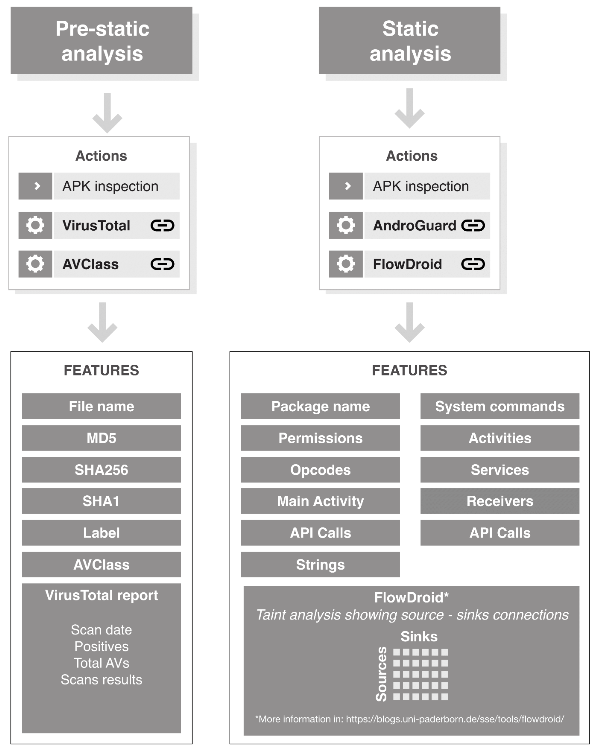
Sau khi AndroPyTool chạy phân tích tĩnh xong, chương trình sẽ trả về một thư mục chứa các file đã được phân tích dưới dạng JSON. Kết quả phân tích nằm trong thư mục /Features\_files và các tệp APk sẽ được lưu trong thư mục /samples.



Trong file JSON của file APK chứa các hàm chức năng như: Pre\_static\_analysis, Static\_analysis, Opcodes, API calls, Strings, API packages, System commands, Intents, Activities, Services, Receivers.



* Pre\_static\_analysis (Phân tích tiền tĩnh): Cung cấp thông tin về tệp APK trước khi phân tích tĩnh thực sự được thực hiện.
* Static\_analysis (Phân tích tĩnh): Cung cấp thông tin về các quyền và tính năng của ứng dụng sau khi đã phân tích tĩnh.
* Mã lệnh (Opcodes): Số lượng Opcode được APK thực hiện .
* Gọi API (API calls): Số lượng lệnh gọi hệ thống do APK thực hiện.
* Chuỗi (Strings): Tập hợp các chuỗi được xác định trong file APK (có số lần xác định).
* Gói API (API packages): Số lần sử dụng của các gói API trong mã của ứng dụng Android.
* Lệnh hệ thống (System commands): Tập hợp các lệnh hệ thống được chạy bởi ứng dụng.
* Hoạt động (Activities): Các hoạt động được APK khai báo.
* Dịch vụ (Services): Các dịch vụ ứng dụng sử dụng.
* Bộ nhận (Receivers): Bộ thu của APK.

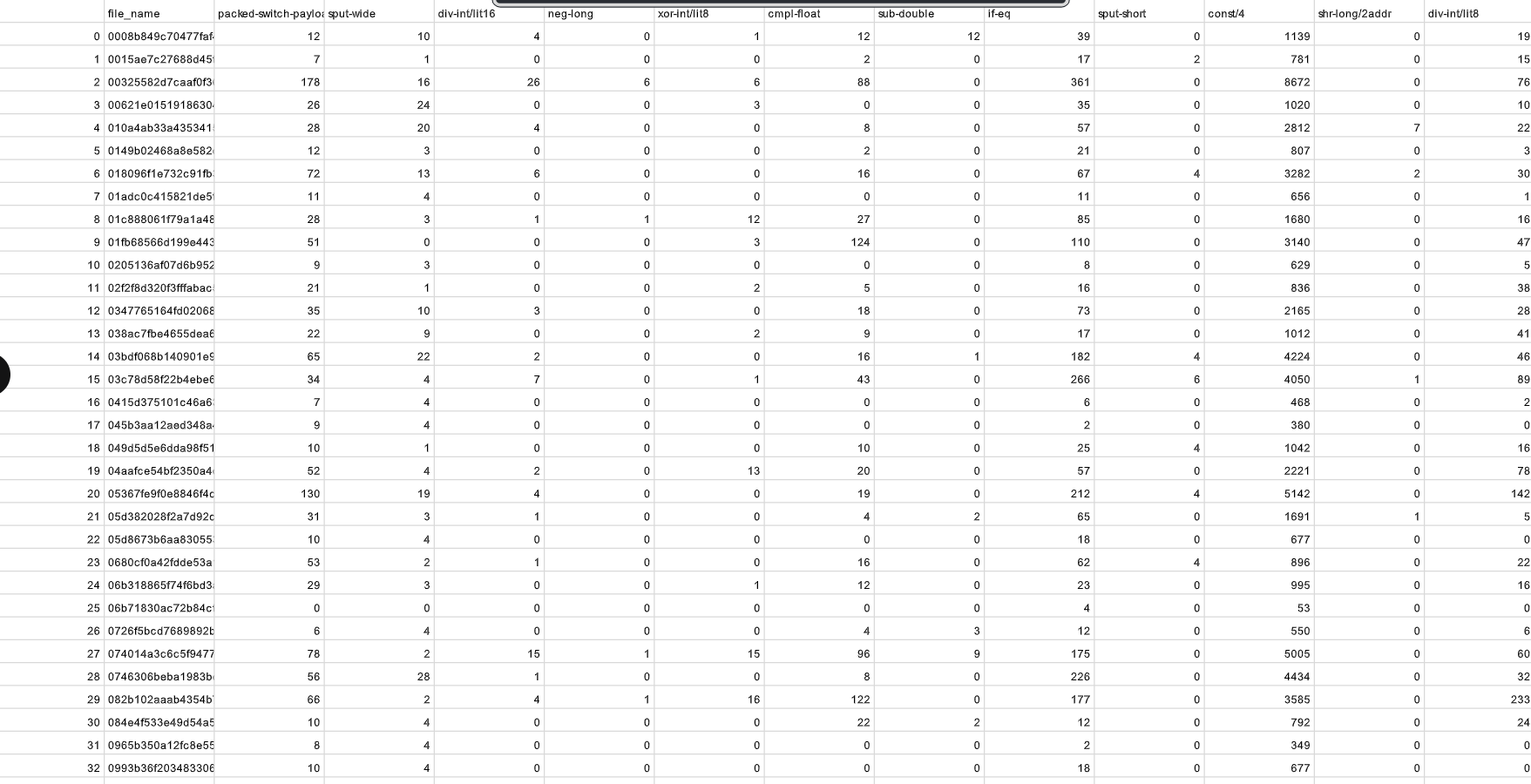


Hình 3.1. Các đặc trưng của file APK

### Xử lý file JSON

Sau khi đã trích xuất được file JSON, chúng em sẽ phân tích các tệp JSON chứa thông tin về opcode (mã lệnh) của các file APK và chuyển đổi thông tin này thành một dạng số hóa, sau đó lưu trữ kết quả vào các file CSV.

Dưới đây là file CSV chứa dữ liệu đã được chuyển đổi từ các file JSON.

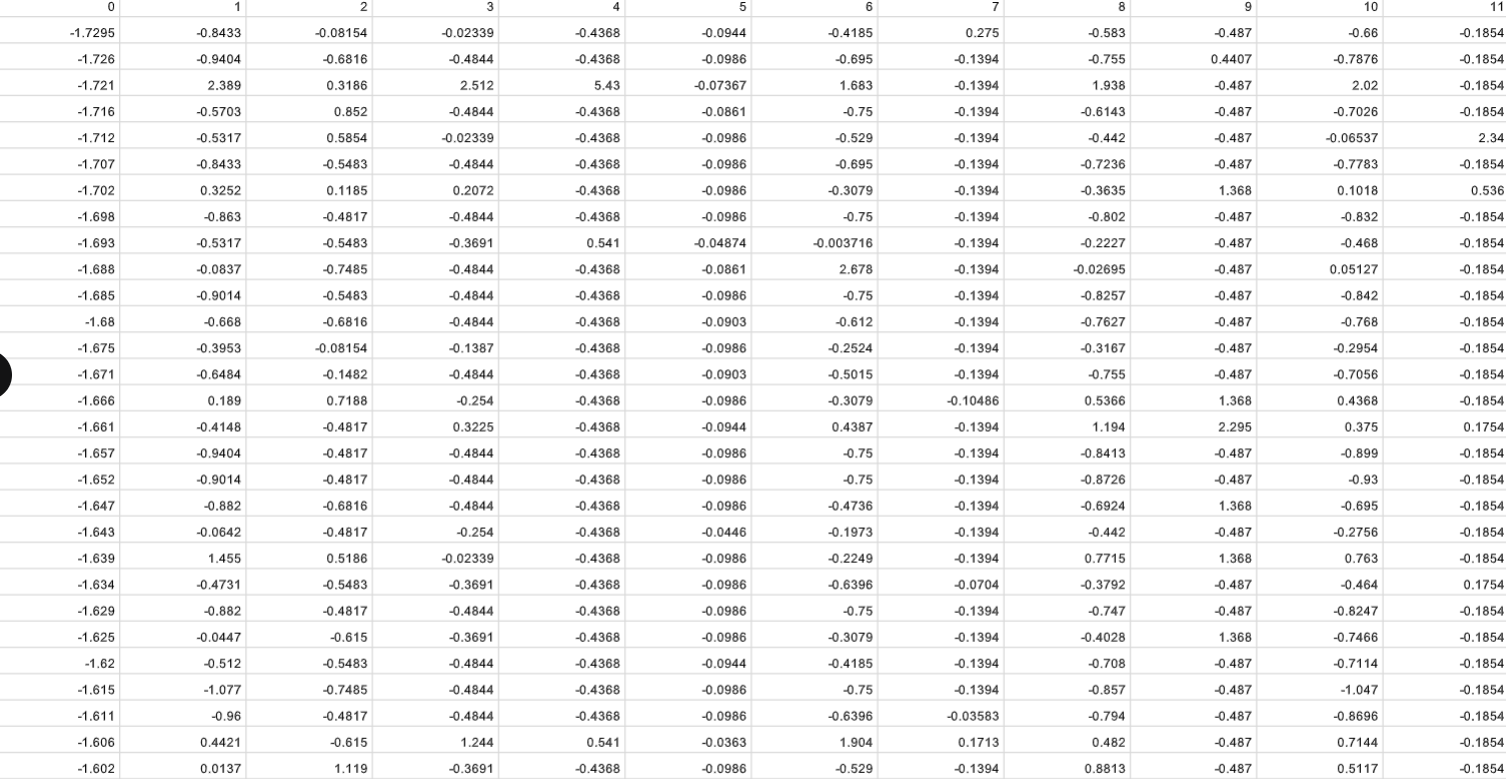


## Tiền xử lý

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, xử lý dữ liệu.
* Tạo model được lấy từ việc đưa dữ liệu đặc trưng được chọn.
* Chia ma trận có được thành các tập con ngẫu nhiên để đưa vào mô hình.
* Đưa vào mô hình học sâu CNN và LSTM.

Kết quả của quá trình tiền xử lý dữ liệu, người dùng sẽ thu được file DeepLearningFiletrain.csv. Mục đích của việc này là loại bỏ các đặc trưng không cần thiết, chuẩn hóa dữ liệu và lưu trữ dữ liệu đã được xử lý để sử dụng trong quá trình huấn luyện mô hình.

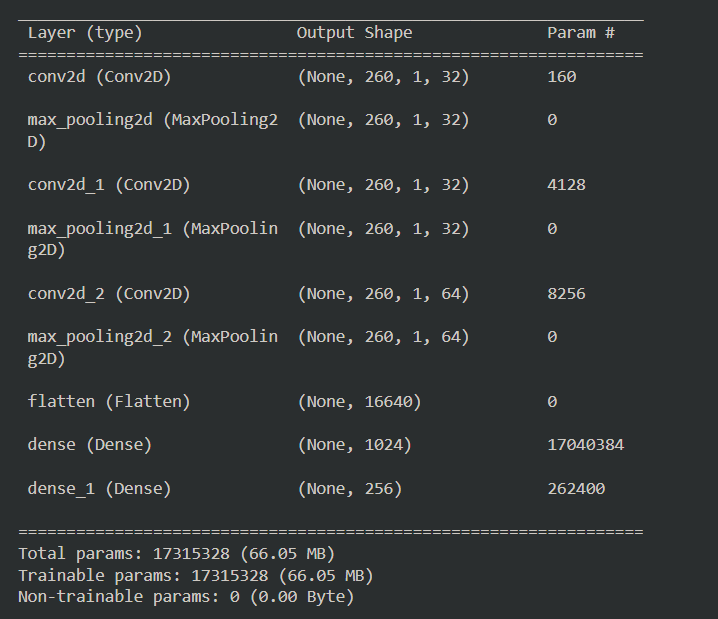


## Huấn luyện mô hình CNN

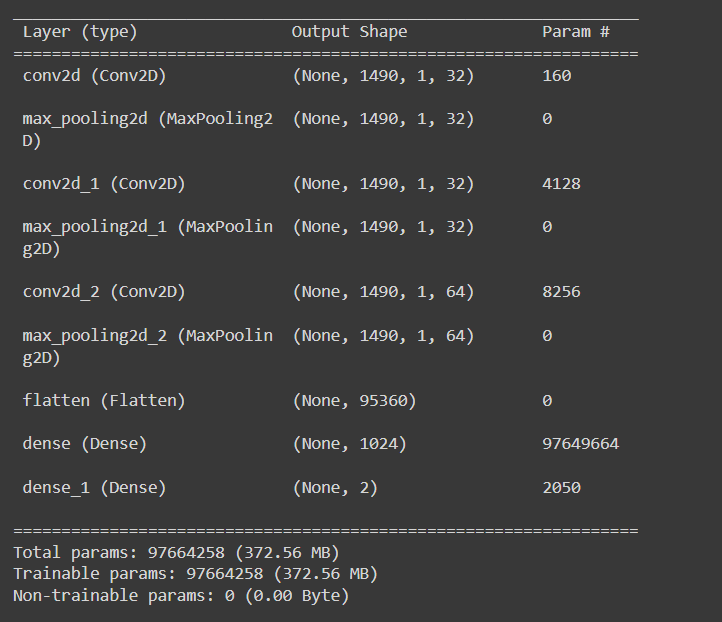
### Xác định kiến trúc mô hình

* 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.
* Lớp Pooling : MaxPooling2D với size 1×2 để đồng bộ dữ liệu đầu vào.
* Lớp Flatten: biến đổi đầu ra từ lớp trước đó thành một vector 1 chiều.
* Lớp kết nối đầy đủ (Dense) với 1024 đơn vị đầu ra, hàm kích hoạt “activation= ReLU”, lớp Dense cuối cùng vào mô hình với số đơn vị đầu ra là số lượng lớp và hàm kích hoạt ‘softmax’ để cho ra xác suất dự đoán của mỗi lớp .

Kiến trúc mô hình CNN trên bộ dữ liệu CICMalDroid 2020:



Kiến trúc mô hình CNN trên bộ dữ liệu Defense Droid:

****

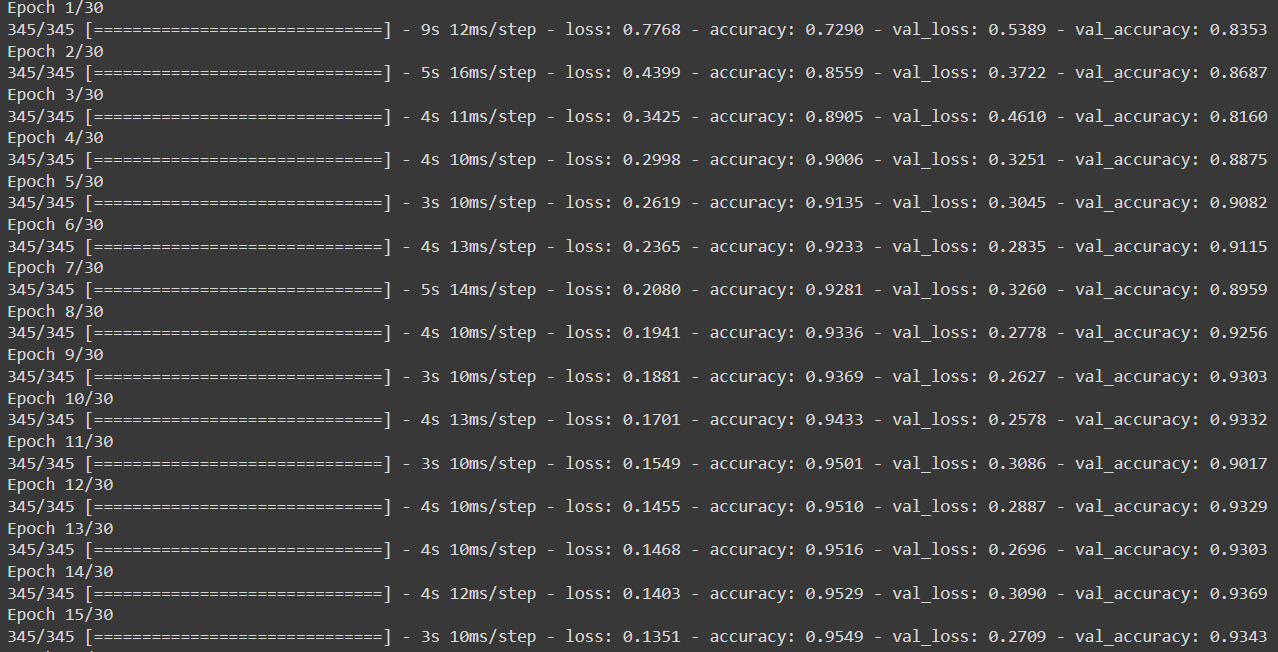
### Xây dựng mô hình

Trong quá trình xây dựng mô hình, nhóm của chúng em đã lựa chọn hai thư viện là thư viện TensorFlow và thư viện Keras, để xây dựng mô hình Convolutional Neural Network (CNN) dựa trên kiến trúc đã lựa chọn.

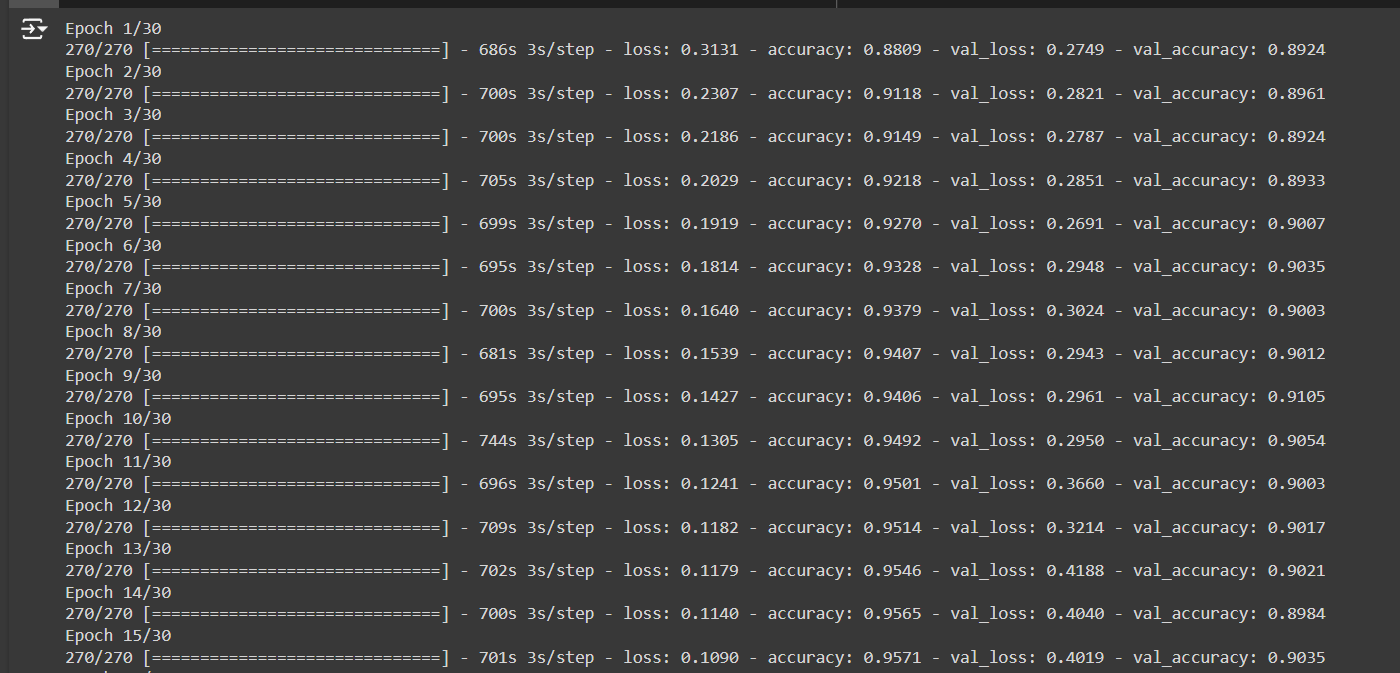
Để xây dựng thành công mô hình, cần xác định hàm loss và các thuật toán tối ưu hóa: Compile mô hình bằng hàm mất mát (loss function) là ‘categorical\_crossentropy’, trình tối ưu hóa là ‘adam’ và các chỉ số đánh giá là ‘accuracy’ (độ chính xác)

Sau khi xây dựng mô hình, nhóm chúng em đã điều chỉnh các siêu tham số: 30 epoch (vòng lặp) với kích thước của batch = 32 (số file trong mỗi lần lấy sample), tốc độ học (learning rate) LR= 0.001. Sử dụng phương thức fit() để huấn luyện mô hình trên tập huấn luyện.

Kết quả của quá trình xây dựng mô hình trên bộ dữ liệu CICMalDroid 2020:



Kết quả của quá trình xây dựng mô hình trên bộ dữ liệu Defense Droid:

****

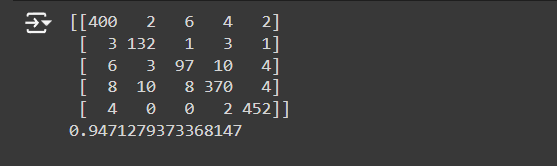
### Số liệu đánh giá

Để đánh giá hiệu suất của các mô hình trong deep learning, nhóm chúng em đã sử dụng hai tiêu chí sau:

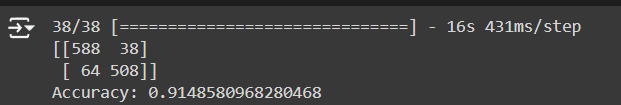
**a, Sử dụng ma trận nhầm lẫn (Confusion Matrix).**

Nó thường được sử dụng để miêu tả sự hiệu quả của một mô hình bằng cách so sánh các dự đoán của mô hình với nhãn thực tế của các điểm dữ liệu. Một confusion matrix có dạng bảng hai chiều, trong đó mỗi hàng của ma trận biểu thị số lượng các điểm dữ liệu thuộc vào một nhãn thực tế, trong khi mỗi cột biểu thị số lượng các điểm dữ liệu được dự đoán bởi mô hình vào từng nhãn.

Ma trận nhầm lẫn với bộ dữ liệu CICMalDroid 2020:

****

Ma trận nhầm lẫn với bộ dữ liệu Defense Droid :

****

**b, Sử dụng độ chính xác (Accuracy) và hàm mất mát (Loss Function)**

Chúng em sử dụng hai hàm là Logloss và Accuracy làm chỉ số đánh giá hiệu suất của mô hình, định nghĩa toán học của chúng lần lượt được thể hiện trong hai công thức sau

* Độ chính xác (Accuracy):

Accuracy =

Độ chính xác đo lường tỷ lệ các dự đoán chính xác của mô hình. Độ chính xác càng cao, mô hình càng chính xác trong việc phân loại.

* Độ mất mát (Loss):

Logloss =

Độ mất mát đo lường sự khác biệt giữa dự đoán của mô hình và giá trị thực tế. Độ mất mát càng thấp, mô hình càng chính xác.

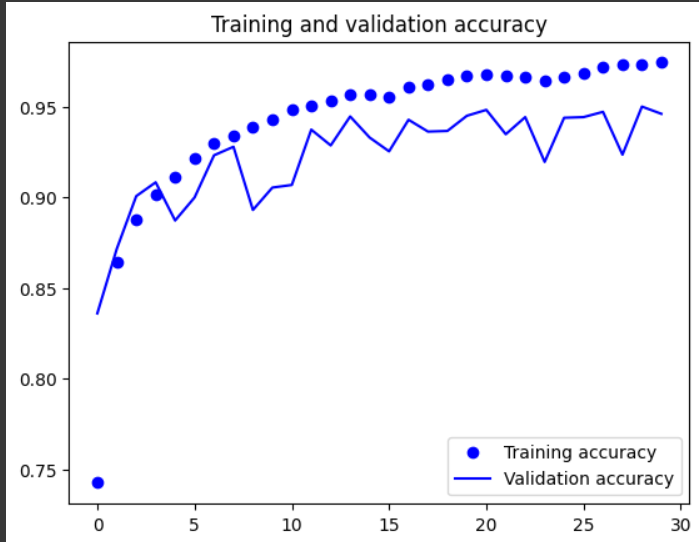
Ngoài ra, các thử nghiệm trong phần này sẽ sử dụng hai chỉ số về độ chính xác và độ mất mát để đo lường hiệu quả của các phương pháp này đối với việc phát hiện các biến thể của phần mềm độc hại. Bên cạnh đó, cũng cần quan tâm tới hai chỉ số về thời gian đào tạo và thời gian phát hiện trên tập dữ liệu thử nghiệm được sử dụng để so sánh.

* Thời gian đào tạo là thời gian cần thiết để huấn luyện mô hình.

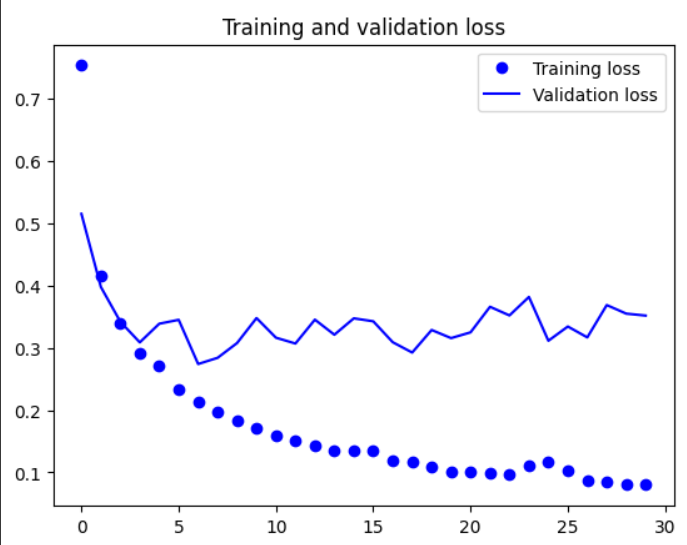
Thời gian đào tạo càng ngắn, hiệu quả huấn luyện càng cao.

* Thời gian phát hiện là thời gian cần thiết để mô hình phát hiện và phân loại các mẫu dữ liệu. Thời gian phát hiện càng ngắn, mô hình càng hiệu quả trong việc phân loại.

Kết quả thực nghiệm của bộ dữ liệu CICMalDroid 2020:

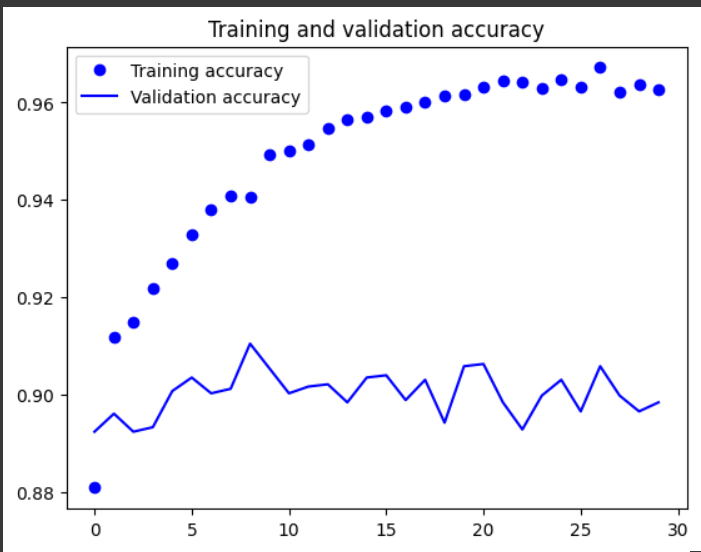


Hình 3.2. Biểu đồ độ chính xác của bộ dữ liệu CICMalDroid 2020

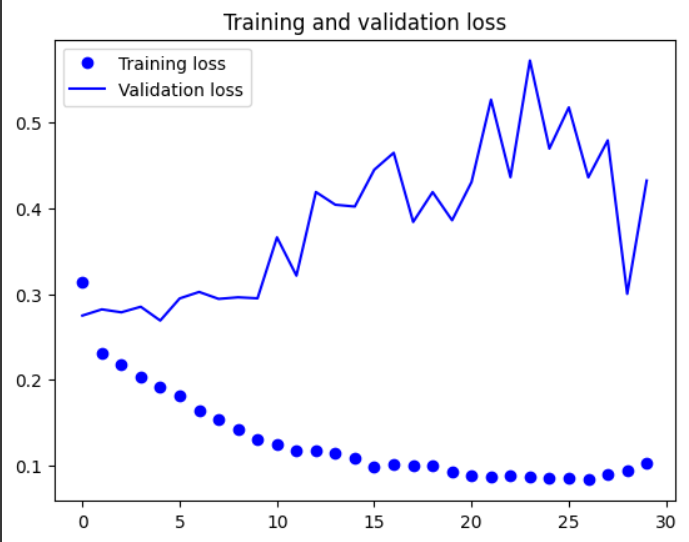


Hình 3.3. Biểu đồ độ mất mát của bộ dữ liệu CICMalDroid 2020

Kết quả thực nghiệm của bộ dữ liệu Defense Droid:

****

Hình 3.4. Biểu đồ độ chính xác của bộ dữ liệu Defense Droid

****

Hình 3.5. Biểu đồ độ mất mát của bộ dữ liệu Defense Droid

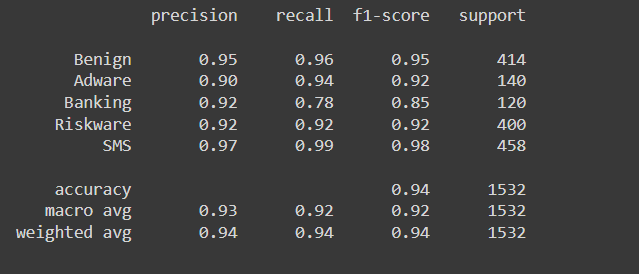
Có thể thấy rằng độ mất xác thực (validation loss) và độ chính xác xác thực (validation accuracy) đều tương quan với độ mất ( training loss) và độ chính xác (training accuracy) của quá trình huấn luyện. Khi độ mất huấn luyện giảm, độ chính xác huấn luyện tăng, thì tương ứng độ mất xác thực và độ chính xác xác thực cũng sẽ cải thiện.

Mặc dù mối quan hệ giữa độ mất và độ chính xác không phải là tuyến tính, nhưng sự thay đổi của chúng trong quá trình huấn luyện cho thấy mô hình không bị quá khớp (overfitting). Điều này vì:

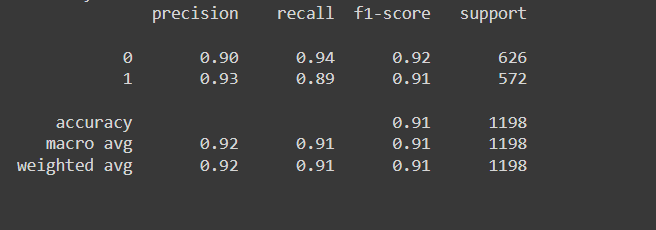
* Độ mất xác thực đang giảm, chứ không tăng lên, cho thấy mô hình đang hội tụ về một giải pháp tốt.
* Khoảng cách giữa độ chính xác huấn luyện và xác thực không quá lớn, chứng tỏ mô hình không bị quá khớp dữ liệu huấn luyện.

Điều này cho thấy rằng, mô hình đang hoạt động hiệu quả trên cả tập dữ liệu đào tạo và tập dữ liệu xác thực, không gặp vấn đề quá mức học thuộc (overfitting), và đang hướng tới việc học được các đặc trưng chung thay vì chỉ nhớ các điểm dữ liệu cụ thể.

Dưới đây là kết quả phân loại mã độc và mã lành của hai bộ dữ liệu:



Hình 3.6. Kết quả phân loại bộ dữ liệu CICMalDroid 2020

****

Hình 3.7. Kết quả phân loại bộ dữ liệu Defense Droid

Dựa vào bảng kết quả phân loại trên, có thể thấy được các thông tin về khả năng phân loại của mô hình đối với các loại dữ liệu khác nhau.

* Độ chính xác (Precision): Tỷ lệ giữa số lượng dự đoán đúng thuộc một lớp cụ thể và tổng số dự đoán thuộc lớp đó. Điều này đo lường khả năng của mô hình trong việc không gán nhầm các điểm dữ liệu vào lớp này.
* Độ phủ (Recall): Tỷ lệ giữa số lượng dự đoán đúng thuộc một lớp cụ thể và tổng số điểm dữ liệu thực sự thuộc lớp đó. Điều này đo lường khả năng của mô hình trong việc không bỏ sót các điểm dữ liệu thực sự thuộc lớp này.
* F1-score: Kết hợp giữa độ chính xác và độ phủ. F1-score là trung bình điều hòa của độ chính xác và độ phủ. Nó giúp đánh giá tổng thể hiệu suất của mô hình.
* Hỗ trợ (Support): Số lượng điểm dữ liệu thực sự thuộc lớp đó trong tập dữ liệu.

Ngoài ra, bảng này cũng cung cấp trung bình macro (macro avg) và trung bình trọng số (weighted avg): Trung bình macro tính trung bình của các chỉ số cho từng lớp, trong khi trung bình trọng số tính trung bình dựa trên tỷ lệ hỗ trợ của từng lớp.

Công thức tính Macro Avg:

Macro Avg =

Công thức tính Weighted Avg:

Weighted Avg =

## Huấn luyện mô hình LSTM

### Xác định kiến trúc mô hình

Lớp Đầu Vào

* **input\_1 (InputLayer)**: Đây là lớp đầu vào, nó chấp nhận dữ liệu đầu vào

Lớp Định Hình Lại (Reshape)

* **reshape (Reshape)**: Lớp này định hình lại dữ liệu đầu vào,, nên lớp này không thay đổi dữ liệu. Lớp này để đảm bảo tính tương thích hoặc để thống nhất mã.

Lớp LSTM

* **lstm (LSTM):** Lớp LSTM đầu tiên xử lý chuỗi đầu vào và chuyển đổi nó thành chuỗi các vector 512.
* **lstm\_1 (LSTM):** Lớp LSTM thứ hai nhận đầu ra từ lớp LSTM đầu tiên và chuyển đổi nó thành chuỗi các vector.
* **lstm\_2 (LSTM):** Lớp LSTM thứ ba nhận đầu ra từ lớp LSTM thứ hai và chuyển đổi nó thành chuỗi các vector.

Lớp Định Hình Lại (Reshape)

* **reshape\_1 (Reshape):** Lớp này làm phẳng đầu ra từ lớp LSTM thứ ba thành một vector dài.

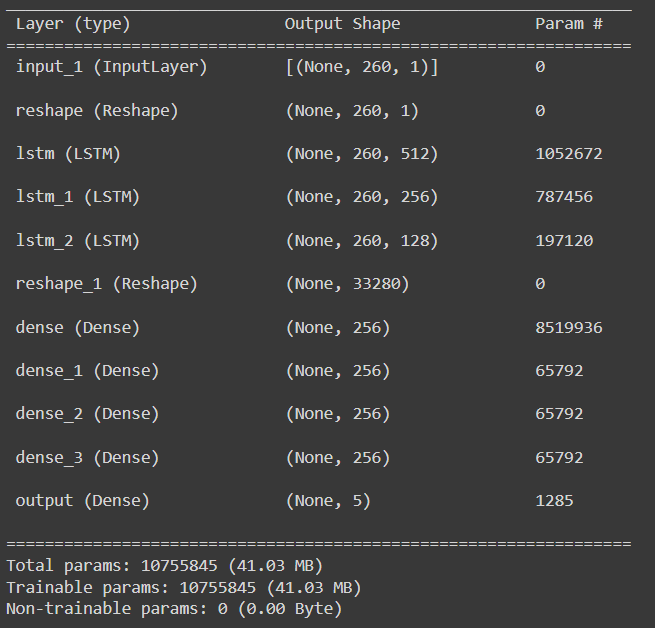
Lớp Dense

* **dense (Dense):** Lớp dense đầu tiên nhận đầu vào đã làm phẳng và chuyển đổi nó thành một vector 256 chiều.
* **dense\_1 (Dense):** Lớp dense thứ hai cũng chuyển đổi đầu vào thành một vector 256 chiều.
* **dense\_2 (Dense):** Lớp dense thứ ba tiếp tục chuyển đổi đầu vào thành một vector 256 chiều.
* **dense\_3 (Dense):** Lớp dense thứ tư cũng chuyển đổi đầu vào thành một vector 256 chiều.

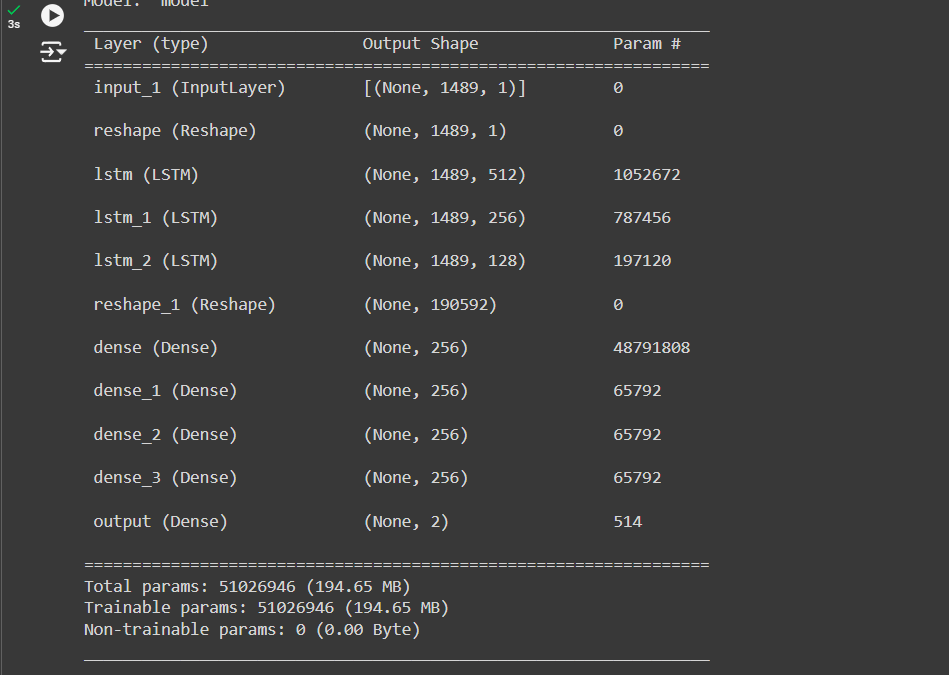
Lớp Đầu Ra

* **output (Dense):** Lớp dense cuối cùng chuyển đổi đầu vào thành một vector 2 chiều, đây là kết quả phân loại nhị phân.

Kiến trúc mô hình LSTM trên bộ dữ liệu CICMalDroid 2020:



Kiến trúc mô hình LSTM trên bộ dữ liệu Defense Droid:

****

### Xây dựng mô hình

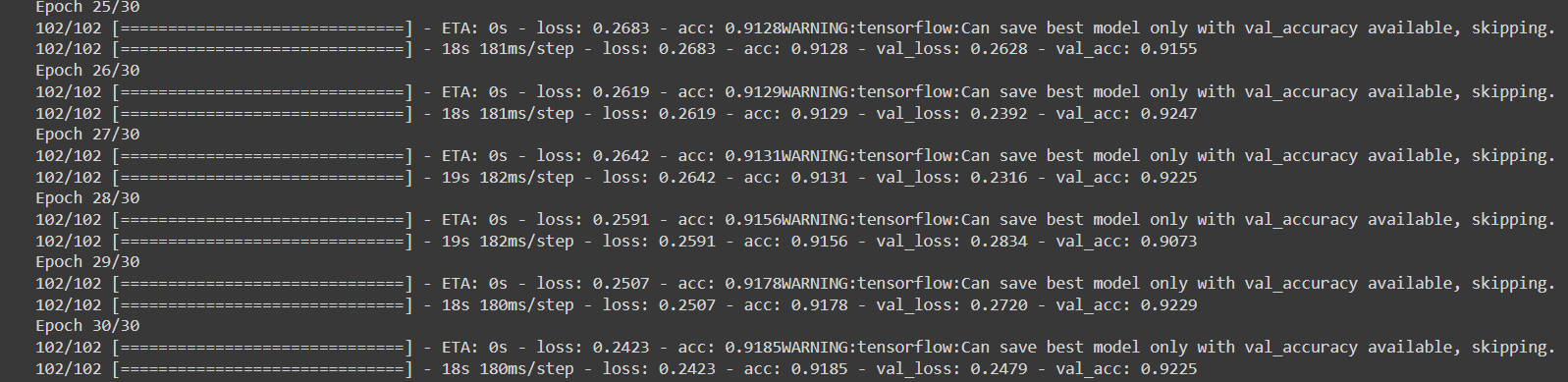
Trong quá trình xây dựng mô hình, nhóm của chúng em đã lựa chọn hai thư viện là thư viện TensorFlow và thư viện Keras, để xây dựng mô hình LSTM (Long Short-Term Memory) dựa trên kiến trúc đã lựa chọn.

Đoạn mã trên sử dụng các callback để giám sát và điều chỉnh quá trình huấn luyện của mô hình LSTM. Có hai callback chính được sử dụng: EarlyStopping và ModelCheckpoint. Early Stopping theo dõi giá trị mất mát trên tập kiểm tra (val\_loss) và sẽ dừng quá trình huấn luyện nếu không thấy sự cải thiện sau 10 epoch liên tiếp, nhằm tránh việc huấn luyện quá lâu khi mô hình không còn cải thiện. ModelCheckpoint theo dõi độ chính xác trên tập kiểm tra (val\_accuracy) và lưu lại mô hình có độ chính xác tốt nhất vào tệp "lstm2.h5", đảm bảo rằng mô hình tốt nhất luôn được lưu trữ.

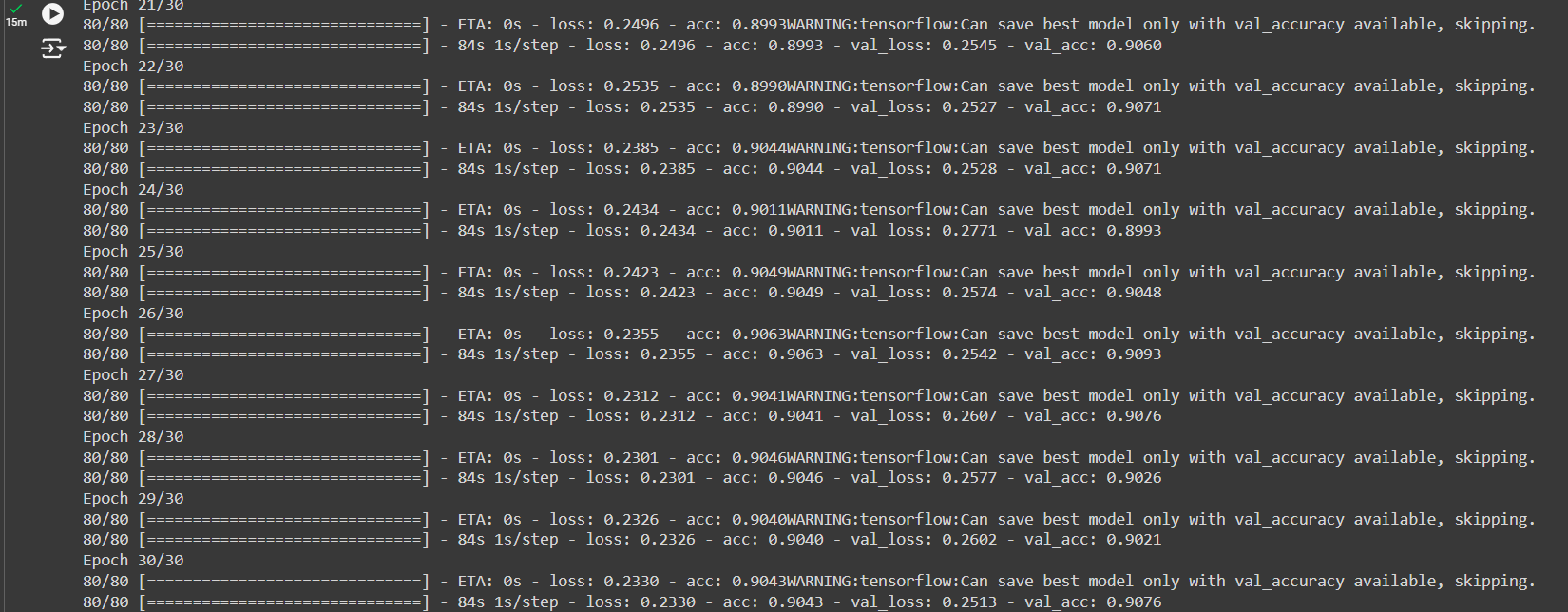
Trong quá trình huấn luyện, mô hình được huấn luyện với dữ liệu X\_train và Y\_train, sử dụng batch size là 128 và tối đa 30 epoch. Dữ liệu kiểm tra (X\_test, Y\_test) được sử dụng để tính toán val\_loss và val\_accuracy sau mỗi epoch. Các callback này giúp giám sát hiệu suất của mô hình và thực hiện các hành động cần thiết để tối ưu hóa quá trình huấn luyện. Kết quả của quá trình huấn luyện được lưu trữ trong biến history, có thể sử dụng để phân tích hiệu suất của mô hình sau khi huấn luyện.

Để xây dựng mô hình, chúng ta cần lựa chọn hàm loss và các thuật toán tối ưu hóa: Ở đây, chúng em dùng compile mô hình bằng hàm mất mát (loss function) là ‘categorical\_crossentropy’, trình tối ưu hóa là ‘adam’ và các chỉ số đánh giá là ‘accuracy’ (độ chính xác). Và cuối cùng, sử dụng phương thức fit() để huấn luyện mô hình trên tập huấn luyện.

Kết quả của quá trình xây dựng mô hình trên bộ dữ liệu CICMalDroid 2020:



Kết quả của quá trình xây dựng mô hình trên bộ dữ liệu Defense Droid:

****

### Số liệu đánh giá

Cũng tương tự mô hình CNN, để đánh giá hiệu suất của mô hình LSTM chúng em cũng dựa trên các tiêu chí sau:

**a, Sử dụng ma trận nhầm lẫn (Confusion Matrix):**

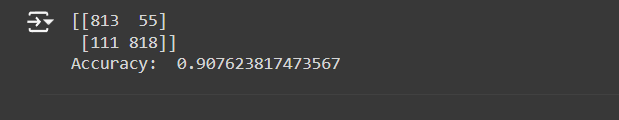
Confusion Matrix: Bảng xếp hạng hiển thị số lượng dự đoán đúng và sai cho từng lớp.

Việc sử dụng ma trận nhầm lẫn giúp phân tích chi tiết về các loại sai lầm mà mô hình LSTM gặp phải. Ma trận nhầm lẫn sẽ hiển thị số lượng mẫu được dự đoán chính xác và sai lầm trong từng lớp. Từ đó, ta có thể xác định các loại nhầm lẫn phổ biến và tìm hiểu nguyên nhân để cải thiện mô hình.

Ma trận nhầm lẫn với bộ dữ liệu CICMalDroid 2020:



Ma trận nhầm lẫn với bộ dữ liệu Defense Droid :



**b, Sử dụng độ chính xác (Accuracy) và hàm mất mát (Loss Function)**

**Accuracy**: Độ chính xác cho biết tỉ lệ các dự đoán đúng trên tổng số mẫu.

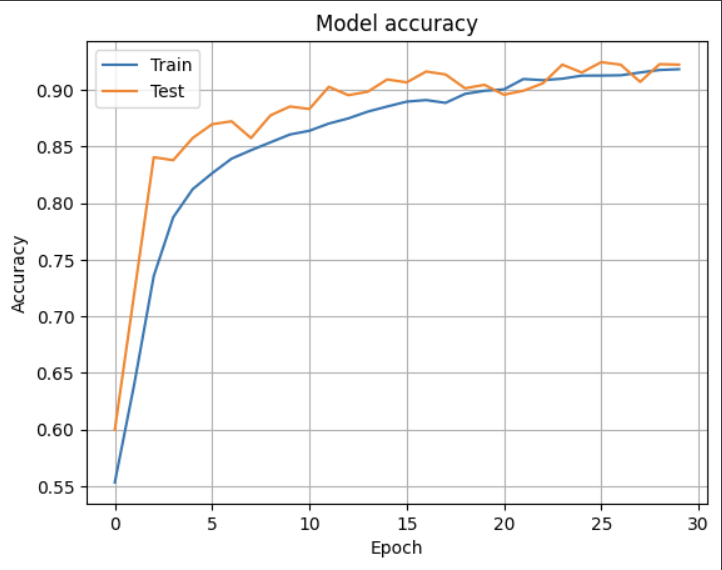
**Loss Function**: Độ mất mát đo lường sự sai lệch giữa các dự đoán của mô hình và các nhãn đúng của dữ liệu.

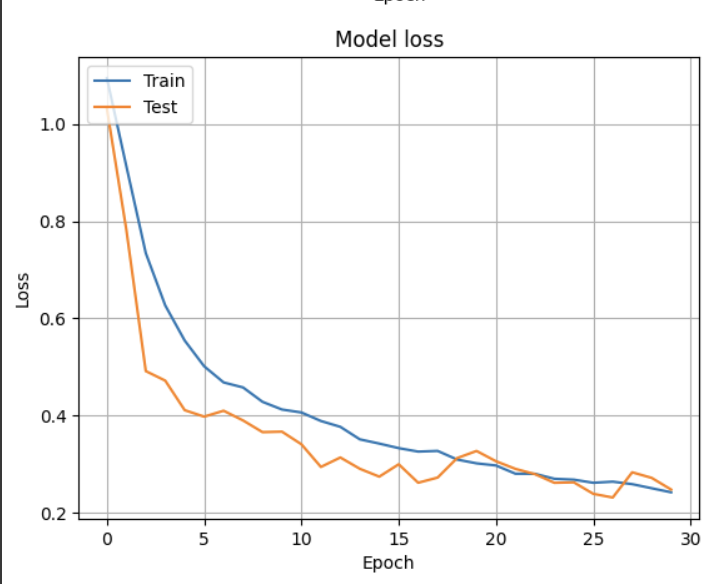
Sau khi huấn luyện mô hình LSTM, ta sẽ sử dụng tập dữ liệu kiểm tra (test set) để đánh giá độ chính xác và độ mất mát của mô hình.

Độ trình xác cao trên tập kiểm tra cho thấy mô hình có khả năng khái quát tốt, có thể dự đoán chính xác trên dữ liệu mới. Tuy nhiên, cần lưu ý rằng độ trình xác không phải là chỉ số duy nhất để đánh giá, vì nó không phản ánh hết các khía cạnh của hiệu suất mô hình.

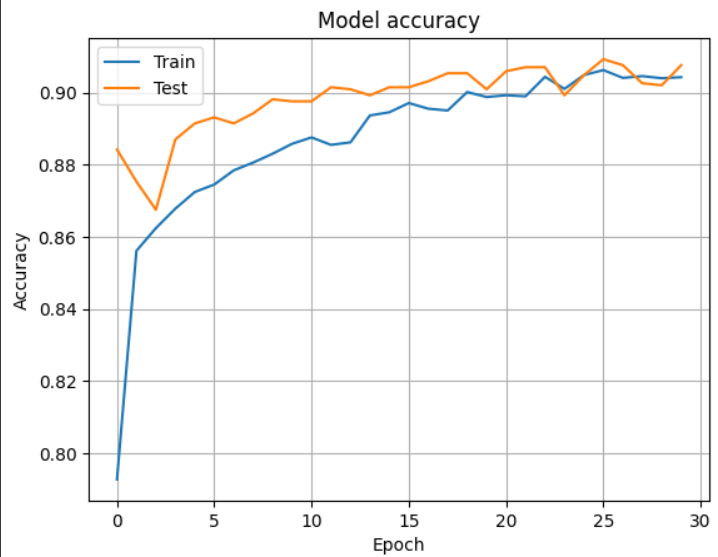
Độ mất mát thấp trên tập kiểm tra cho thấy mô hình có thể dự đoán chính xác trên dữ liệu mới. Kết hợp độ trình xác và độ mất mát trên tập kiểm tra sẽ cung cấp một bức tranh toàn diện về hiệu suất mô hình LSTM.

Kết quả thực nghiệm của bộ dữ liệu CICMalDroid 2020:

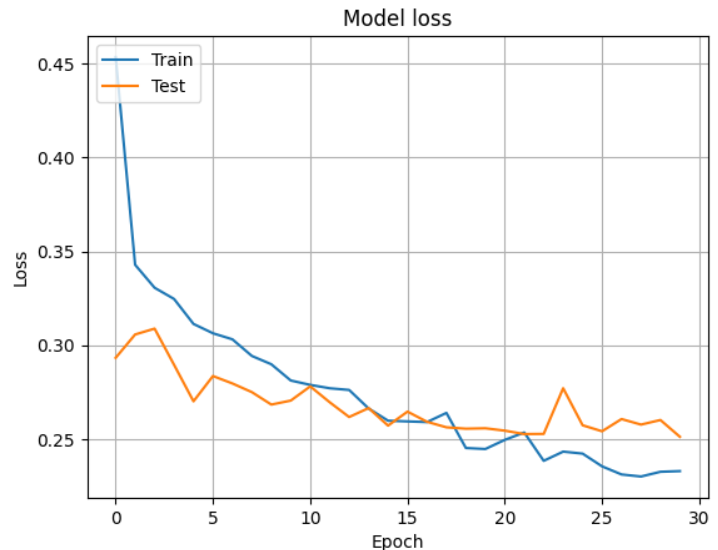
***Hình 3.8. Biểu đồ độ chính xác của bộ dữ liệu CICMalDroid 2020***

***Hình 3.9. Biểu đồ độ mất mát của bộ dữ liệu CICMalDroid 2020***

Kết quả thực nghiệm của bộ dữ liệu Defense Droid:

****

***Hình 3.10. Biểu đồ độ chính xác của bộ dữ liệu Defense Droid***

***Hình 3.11. Biểu đồ độ mất mát của bộ dữ liệu Defense Droid***

Sau khi huấn luyện mô hình LSTM trên 2 bộ dữ liệu để phát hiện mã độc trên hệ điều hành Android, ta có thể thu được kết quả như sau:

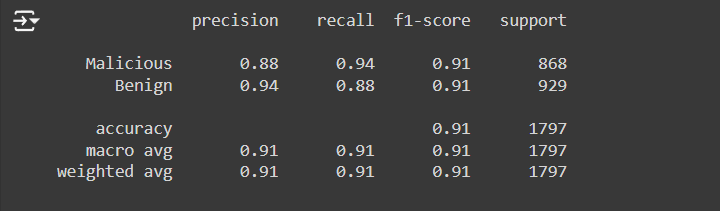
* Bộ dữ liệu CICMalDroid 2020: Độ trình xác trên tập kiểm tra: 91, 9%, độ mất mát trên tập kiểm tra: 0,24
* Bộ dữ liệu Defense Droid: Độ trình xác trên tập kiểm tra: 90,76%, độ mất mát trên tập kiểm tra: 0,25

Kết quả này cho thấy mô hình LSTM có khả năng phân loại tốt, với độ trình xác cao và độ mất mát tương đối thấp trên tập kiểm tra. Điều này gợi ý rằng mô hình có thể dự đoán chính xác trên dữ liệu mới và có khả năng khái quát tốt.

Đánh giá mô hình qua các độ đo như precision, recall, F1-score đối với bài toán phân loại.

* **Precision**: Tỉ lệ số lượng dự đoán dương tính đúng (true positive) so với tổng số dự đoán dương tính.
* **Recall (Sensitivity)**: Tỉ lệ số lượng dự đoán dương tính đúng so với tổng số mẫu thực sự là dương tính.
* **F1-score**: Đánh giá tổng thể của precision và recall, cung cấp một chỉ số đồng nhất cho hiệu suất của mô hình.

Ví dụ, với mô hình LSTM phát hiện mã độc với Bộ dữ liệu Defense Droid trên hệ điều hành Android, ta có thể thu được các kết quả trên tập kiểm tra như sau:

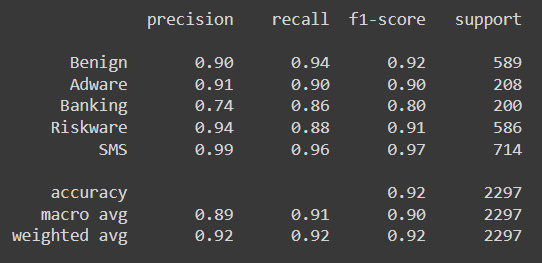


Hình 3.11. Kết quả phân loại bộ dữ liệu Defense Droid

Từ kết quả này, ta có thể rút ra các nhận xét sau:

* Độ chính xác (Precision) là 0.88, nghĩa là trong tổng số các mẫu được dự đoán là mã độc, 88% trong số đó là chính xác.
* Độ nhạy (Recall) là 0.94, điều này cho thấy mô hình phát hiện được 94% số mẫu mã độc thực tế.
* Điểm F1 là 0.91, cho thấy sự cân bằng tốt giữa Precision và Recall, thể hiện hiệu suất tổng thể của mô hình ở mức cao.

Tương tự, với mô hình LSTM phát hiện mã độc sử dụng bộ dữ liệu Bộ dữ liệu CICMalDroid 2020, cũng cho kết quả khá cao sau khi tiến hành huấn luyện và phân loại:

****

Hình 3.12. Kết quả phân loại bộ dữ liệu CICMalDroid 2020

Việc đánh giá toàn diện mô hình LSTM bằng các kỹ thuật như trên sẽ giúp ta hiểu rõ hiệu suất và hạn chế của mô hình, từ đó có thể cải thiện và phát triển nó để đạt được kết quả tốt hơn trong việc phát hiện mã độc trên hệ điều hành Android.

## Kết luận Chương 3

Trong chương này, nhóm chúng em đã triển khai thực nghiệm về việc ứng dụng mô hình học sâu trong việc phát hiện mã độc trên hệ điều hành Android. Từ kết quả thực nghiệm, ta có thể so sánh được ưu điểm, nhược điểm cũng như hiệu suất của hai mô hình CNN và LSTM trên hai bộ dữ liệu khác nhau: một là bộ phân loại các loại mã độc và mã lành (CICMalDroid 2020), hai là bộ chỉ phân loại mã độc và mã lành (Defense Droid).

Ưu điểm, nhược điểm của 2 mô hình:

* CNN thường hiệu quả trong việc học các đặc trưng không gian như hình ảnh và văn bản, sử dụng lớp convolution và pooling để tự động hóa các đặc trưng quan trọng. Đây là mô hình phù hợp với các bộ dữ liệu lớn và phức tạp. Tuy nhiên, CNN không thể hiệu quả trong việc mô hình hóa thông tin chuỗi như LSTM và có thể thiếu khả năng mô hình hóa mối quan hệ phụ thuộc dài hạn giữa các đặc trưng.
* LSTM có khả năng mô hình hóa các phụ thuộc dài hạn trong dữ liệu chuỗi như văn bản và dòng mã, giúp duy trì thông tin quan trọng qua các bước thời gian. Tuy nhiên, mô hình này không hiệu quả trong việc xử lý dữ liệu không gian như CNN và thường đòi hỏi nhiều tài nguyên hơn để huấn luyện.

Đánh giá hiệu suất trên 2 bộ dữ liệu:

* Trên bộ dữ liệu phân loại các loại mã độc và mã lành, cả hai mô hình đều cho kết quả độ chính xác cao, tuy nhiên thời gian huấn luyện và tài nguyên máy tính tiêu tốn của chúng cũng cao hơn.
* Trên bộ dữ liệu chỉ phân loại mã độc và mã lành, kết quả huấn luyện của cả hai mô hình thấp hơn, nhưng thời gian huấn luyện và tài nguyên sử dụng lại ít hơn.

Dựa vào những điểm này, ta có thể rút ra kỹ năng lựa chọn mô hình: CNN thích hợp hơn khi dữ liệu có tính không gian như hình ảnh của mã độc, trong khi LSTM thích hợp hơn khi dữ liệu có tính tuần tự như các dòng mã. Tuy nhiên, việc lựa chọn mô hình phù hợp còn phụ thuộc vào nhiều yếu tố như kích thước dữ liệu, số lượng mẫu huấn luyện và hiệu suất mong đợi.

Ngoài ra để cải thiện hiệu suất, đặc biệt là trên các bộ dữ liệu phức tạp hơn thì ta có thể kết hợp các đặc trưng từ cả hai mô hình. Để chọn mô hình tối ưu thì cần phải thử nghiệm và so sánh từng mô hình trên từng bộ dữ liệu cụ thể để đưa ra quyết định cuối cùng.

# 

# KẾT LUẬN

Trong suốt quá trình thực hiện đề tài: “Ứng dụng mô hình học sâu cho bài toán phát hiện mã độc trên hệ điều hành Android”, nhóm em đã hoàn thành được các nội dung sau:

Về mặt lý thuyết:

* Tổng quan về hệ điều hành Android và cấu trúc file APK.
* Tìm hiểu về mã độc, một số loại mã độc phổ biến, cách phân biệt mã độc với mã lành và phương pháp phát hiện mã độc.
* Lý thuyết về machine learning bao gồm tổng quan về học máy, học sâu và một số mô hình cơ bản.

Về mặt thực nghiệm:

* Huấn luyện hai mô hình CNN và LSTM dựa trên hai bộ dữ liệu CICMalDroid 2020 và Defense Droid để phát hiện mã độc.
* Đánh giá được hiệu suất mô hình trên từng bộ dữ liệu.

**Hạn chế**: Ở thời điểm hiện tại, chương trình mới chỉ phát hiện mã độc trên file APK thông thường, chưa phát hiện được các loại mã độc dựa trên các kĩ 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**: Tiến tới có thể phân tích và phát hiện tất cả các loại file APK. 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.

Đây là sản phẩm của nhóm em về đề tài “Nghiên cứu ứng dụng mô hình

học sâu trong việc phát hiện mã độc trên hệ điều hành Android” và được giám sát và hướng dẫn bởi cô Thái Thị Thanh Vân. Thay mặt cho các thành viên, em xin cảm ơn sự giúp đỡ tận tình của cô trong suốt quá trình làm sản phẩm của nhóm.

# 

# TÀI LIỆU THAM KHẢO

[1] Android (hệ điều hành) – Wikipedia tiếng Việt

[2] Mã độc là gì? Phân loại và phòng tránh mã độc như thế nào (bkns.vn)

[3] Học máy – Wikipedia tiếng Việt

[4] Học máy (Machine Learning) là gì? Cách hoạt động và ứng dụng (fpt-is.com)

[5] [RNN] LSTM là gì? (dominhhai.github.io)

[6] Convolutional Neural Networks in Python | DataCamp

[7] Extracting Specific Keys/Values From A Messed-Up JSON File (Python) (plainenglish.io)