BAN CƠ YẾU CHÍNH PHỦ

HỌC VIỆN KỸ THUẬT MẬT MÃ

BÁO CÁO THỰC TẬP

OCTOBER 9, 2019

HỌC VIỆN KỸ THUẬT MẬT MÃ

NHÓM 2.10

BAN CƠ YẾU CHÍNH PHỦ

HỌC VIỆN KỸ THUẬT MẬT MÃ

BÁO CÁO
THỰC TẬP CƠ SỞ CHUYÊN NGÀNH

ĐỀ TÀI:

ÁP DỤNG MACHINE LEARNING VÀ DEEP LEARNING VÀO PHÂN TÍCH MÃ ĐỘC ANDROID

Ngành: Công nghệ thông tin

Chuyên ngành: An toàn thông tin

GIẢNG VIÊN HƯỚNG DẪN:

- ThS. LÊ ĐỨC THUẬN

SINH VIÊN THỰC HIỆN:

- NGUYỄN LÊ QUỐC ANH
- NGUYỄN XUÂN THIÊN

TP.HCM, 2019

LỜI MỞ ĐẦU

Trong những năm trở lại đây, với sử ra đời và bùng nổ của Artificial Intelligence, Machine Learning, Deep Learning. Và cụ thể hơn là Machine Learning (Học Máy hoặc Máy Học) nổi lên như một bằng chứng của cuộc cách mạng công nghiệp lần thứ tư (1 - động cơ hơi nước, 2 - năng lượng điện, 3 - công nghệ thông tin). Trí Tuệ Nhân Tạo đang len lỏi vào mọi lĩnh vực trong đời sống mà có thể chúng ta không nhận ra. Xe tự hành của Google và Tesla, hệ thống tự tag khuôn mặt trong ảnh của Facebook, trợ lý ảo Siri của Apple, hệ thống gợi ý sản phẩm của Amazon, hệ thống gợi ý phim của Netflix, máy chơi cờ vây AlphaGo của Google DeepMind, …, chỉ là một vài trong vô vàn những ứng dụng của AI/Machine Learning.

Là một sinh viên ngành An toàn thông tin của Học Viện Kỹ Thuật Mật Mã, với mong muốn phát triển bản thân, tìm hiểu công nghệ mới và áp dụng vào chuyên ngành. Đề tài: "Phân tích mã độc sử dụng Học máy và học sâu" không chỉ là tâm huyết mà còn là một đề tài mang nhiều ý nghĩa đối với chúng em. Trong đó, việc áp dụng được công nghệ này vào phân tích mã độc mang nghĩa quan trọng nhất.

Từ khi Học Máy ra đời đã có rất nhiều đề tài liên quan đến phân tích mã độc ra đời sử dụng nhiều mô hình và đặc trưng khác nhau. Tuy nhiên, vẫn còn một số giới hạn nhất định trong việc phát hiện nhóm mã độc mới. Nhằm cải thiện vấn đề đó, chúng em cố gắng đưa ra một mô hình, và bộ đặc trưng mới dưới góc nhìn và bằng sự hiểu biết của chúng em. Kết hợp với sự bùng nổ của các thiết bị di động, mã độc trên các thiết bị di động phát triển mạnh và chưa được chú ý bởi người dùng. Điều này cực kì nguy hiểm! Theo đó, chúng em chọn mã độc trên các thiết bị di động, cụ thể là Android là lĩnh vực áp dụng cho đề tài này. Đề tài: "Sử dụng Học Máy và Deep Learning để phân tích mã độc trên Android". Sau đây là quá trình nghiên cứu của chúng em.

Chân thành cảm ơn thầy Lê Đức Thuận, và bạn Hoàng Quốc Cường đã hỗ trợ chúng em trong quá trình thực hiện đề tài nghiên cứu này!

MỤC LỤC

DANH MỤC BẢNG

DANH MỤC HÌNH ẢNH

DANH MỤC TỪ VIẾT TẮT

AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
ANN	Artificial Neural Networks
APK	Android Package Kit
CNN	Convolutional Neural Network
RDF	Random Decision Forest
SVM	Super Vector Machine

I. TỔNG QUAN BỐI CẢNH

1. 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 ra mắt vào năm 2007 cùng với tuyên bố thành lập Liên minh thiết bị cầm tay mở: một hiệp hội gồm các công ty phần cứng, phần mềm, và viễn thông với mục tiêu đẩy mạnh các tiêu chuẩn mở cho các thiết bị di động. Chiếc điện thoại đầu tiên chạy Android được bán vào năm 2008

Android có mã nguồn mở và Google phát hành mã nguồn theo Giấy phép Apache. Chính mã nguồn mở cùng với một giấy phép không có nhiều ràng buộc đã cho phép các nhà phát triển thiết bị, mạng di động và các lập trình viên nhiệt huyết được 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 một loại 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.

Những yếu tố này đã giúp Android trở thành nền tảng điện thoại thông minh phổ biến nhất thế giới, vượt qua Symbian OS vào quý 4 năm 2010, và được các công ty công nghệ lựa chọn khi họ cần một hệ điều hành không nặng nề, có khả năng tinh chỉnh, và giá rẻ chạy trên các thiết bị công nghệ cao thay vì tạo dựng từ đầu Kết quả là mặc dù được thiết kế để chạy trên điện thoại và máy tính bảng, Android đã xuất hiện trên TV, máy chơi game và các thiết bị điện tử khác. Bản chất mở của Android cũng khích lệ một đội ngũ đông đảo lập trình viên và những người đam mê sử dụng mã nguồn mở để tạo ra những dự án do cộng đồng quản lý. Những dự án này bổ sung các tính năng cao cấp cho những người dùng thích tìm tòi hoặc đưa Android vào các thiết bị ban đầu chạy hệ điều hành khác.

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.

Android có lượng ứng dụng của bên thứ ba ngày càng nhiều, được chọn lọc và đặt trên một cửa hàng ứng dụng như Google Play hay Amazon Appstore để người dùng lấy về, hoặc bằng cách tải xuống rồi cài đặt tập tin APK từ trang web khác Các ứng dụng trên Play Store cho phép người dùng duyệt, tải về và cập nhật các ứng dụng do Google và các nhà phát triển thứ ba phát hành. Play Store được cài đặt sẵn trên các thiết bị thỏa mãn điều kiện tương thích của Google Ứng dụng sẽ tự động lọc ra một danh sách các ứng dụng tương thích với thiết bị của người dùng, và nhà phát triển có thể giới hạn ứng dụng của họ chỉ dành cho những nhà mạng cố định hoặc những quốc gia cố định vì lý do kinh doanh Nếu người dùng mua một ứng dụng mà họ cảm thấy không thích, họ được hoàn trả tiền sau 15 phút kể từ lúc tải về, và một vài nhà mạng còn có khả năng mua giúp các ứng dụng trên Google Play, sau đó tính tiền vào trong hóa đơn sử dụng hàng tháng của người dùng. Đến tháng 9 năm 2012, có hơn 675.000 ứng dụng dành cho Android, và số lượng ứng dụng tải về từ Play Store ước tính đạt 25 tỷ

Các ứng dụng cho Android được phát triển bằng ngôn ngữ Java sử dụng Bộ phát triển phần mềm Android (SDK). SDK bao gồm một bộ đầy đủ các công cụ dùng để phát triển gồm có công cụ gỡ lỗi, thư viện phần mềm, bộ giả lập điện thoại dựa trên QEMU, tài liệu hướng dẫn, mã nguồn mẫu, và hướng dẫn từng bước. Môi trường phát triển tích hợp (IDE) được hỗ trợ chính thức là Eclipse sử dụng phần bổ sung Android Development Tools (ADT). Các công cụ phát triển khác cũng có sẵn, gồm có Bộ phát triển gốc dành cho các ứng dụng hoặc phần mở rộng viết bằng C hoặc C++, Google App Inventor, một môi trường đồ họa cho những nhà lập trình mới bắt đầu, và nhiều nền tảng ứng dụng web di động đa nền tảng phong phú.

Để vượt qua những hạn chế khi tiếp cận các dịch vụ của Google do sự Kiểm duyệt Internet tại Cộng hòa Nhân dân Trung Hoa, các thiết bị Android bán tại Trung Quốc lục địa thường được điều chỉnh chỉ được sử dụng dịch vụ đã được duyệt

Để vượt qua những hạn chế khi tiếp cận các dịch vụ của Google do sự Kiểm duyệt Internet tại Cộng hòa Nhân dân Trung Hoa, các thiết bị Android bán tại Trung Quốc lục địa thường được điều chỉnh chỉ được sử dụng dịch vụ đã được duyện.

Các phiên bản Hệ điều hành Android:

Phiên bản	Tên	Ngày phát hành
Android 1.5	Cupcake	27/4/2009
Android 1.6	Donut	15/9/2009
Android 2.0 - 2.1	Eclair	26/9/2009 (phát hành lần đầu)
Android 2.2 - 2.2.3	Froyo	20/5/2010 (phát hành lần đầu)
Android 2.3 - 2.3.7	Gingerbread	6/12/2010 (phát hành lần đầu)
Android 3.0 - 3.2.6	Honeycomb	22/2/2011 (phát hành lần đầu)
Android 4.0 - 4.0.4	Ice Cream Sandwich	18/10/2011 (phát hành lần đầu)
Android 4.1 - 4.3.1	Jelly Bean	9/7/2012 (phát hành lần đầu)
Android 4.4 - 4.4.4	KitKat	31/10/2013 (phát hành lần đầu)
Android 5.0 - 5.1.1	Lollipop	12/11/2014 (phát hành lần đầu)
Android 6.0 - 6.0.1	Marshmallow	5/10/2015 (phát hành lần đầu)
Android 7.0 - 7.1.2	Nougat	22/8/2016 (phát hành lần đầu)
Android 8.0 - 8.1	Oreo	21/8/2017 (phát hành lần đầu)

Table 1 Bảng các phiên bản hệ điều hành

2. Mã độc trên Android

Malware trên Android là gì?

Như chúng ta đã biết Malware (phần mềm ác tính) viết tắt của cụm từ Malicious Sofware, là một phần mềm máy tính được thiết kế với mục đích thâm nhập hoặc gây hỏng hóc máy tính mà người sử dụng không hề hay biết.

Đấy là định nghĩa của Malware trên máy tính thông thường, vậy Malware trên smartphone thì có gì khác ?

Theo thống kê của các hãng bảo mật trên thế giới thì hiện các Malware hiện nay mới chỉ dừng lại ở mức độ xâm nhập và ăn cắp thông tin của người dùng và nó chưa có cơ chế lây lan. Theo các kết quả trên thì Malware trên Smartphone hiện nay về cách thức hoạt động giống như một phần mềm gián điệp (Trojan) hơn là một virus phá hủy.

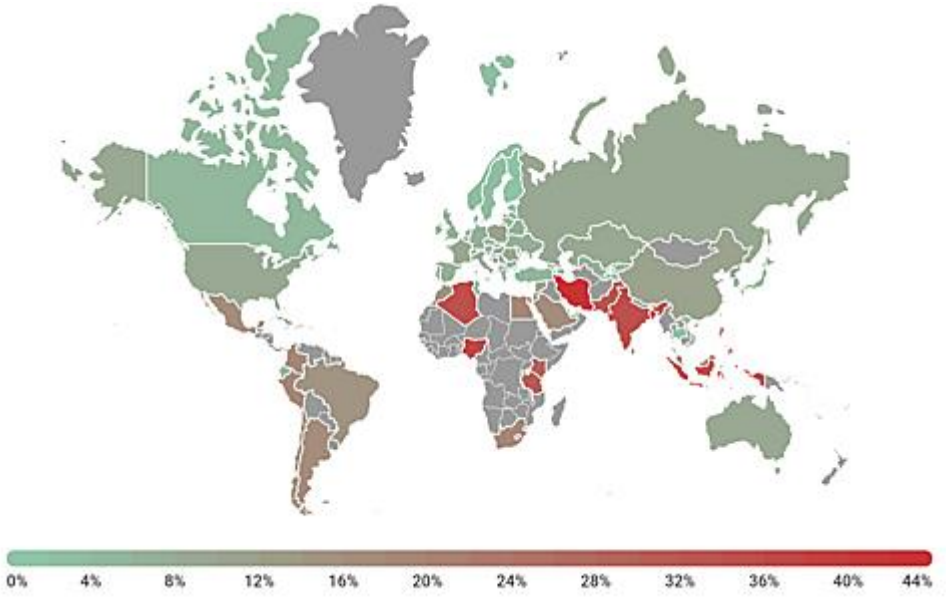


Figure 1 Bảng phân bổ các cuộc tấn công trên Android trên các Quốc gia trên thế giới

Số lượng tấn công sử dụng mã độc trên di động đã tăng từ 66,4 triệu năm 2017 lên 116,5 triệu vụ trong 2018.

Thiết bị di động trở thành mục tiêu được tin tặc và kẻ xấu tấn công mạng. Không chỉ hoạt động tinh vi hơn, số lượng tấn công đã bùng phát mạnh thời gian gần đây. Theo báo cáo nghiên cứu về mã độc trên di động năm 2018 của *Kaspersky Lab*, chỉ sau một năm tổng số vụ tấn công sử dụng mã độc trên di động đã tăng gấp đôi, từ 66,4 triệu năm 2017 lên tới 116,5 triệu cuộc tấn công vào năm 2018.

Sơ hở từ việc người dùng không cài đặt bất kỳ giải pháp bảo mật nào trên di động trong khi lưu trữ dữ liệu khổng lồ đã khiến những thiết bị này trở thành mục tiêu hàng đầu hoặc làm kênh phát tán và lây nhiễm mã độc quan trong trong các chiến dịch tấn công của tội phạm mạng.

Riêng trong 2018, số người dùng bị ảnh hưởng bởi tấn công mạng trên di động đã lên tới gần 9,9 triệu, tăng 774.000 người so với một năm trước. Trong đó, mối đe dọa tăng mạnh nhất là sử dụng Trojan-Droppers với tỷ lệ tăng gần gấp đôi từ 8,63% lên 17,21%. Loại mã độc này được thiết kế đặc biệt để vượt qua các hệ thống bảo mật, từ đó lây nhiễm tất cả các loại mã độc từ Trojan ngân hàng đến mã độc tống tiền (ransomware).

3. Trí thông minh nhân tạo, Học Máy, Học Sâu.

AI – Artificial Intelligence (Trí tuê nhân tạo) được nhắc đến lần đầu tại một hội nghị khoa học vào năm 1956. Và trong suối vài thập kỷ sau đó AI dược dự đoán là chìa khóa mở ra tương lai của văn minh nhân loại. Đến năm 2015 nó đã bắt đầu bùng nổ như một bằng chứng của cuộc cách mạng công nghiệp 4.0. Các ông lớn về công nghê cũng đang tập chung và đầu tư rất nhiều cho lĩnh vực này như Google, Apple, Samsung, Amazon,…

Trong mắt nhiều người AI có đặc điểm giống với trí thông minh con người được đưa vào máy móc và điều đó thì vẫn đang nằm ngoài tầm với của con người và cũng là mục tiêu mà con người muốn vươn đến. Hiện nay có rất nhiều sản phẩm công nghệ mà con người sử dụng được tích hợp chức năng AI

Đến nay nó được áp dụng hầu như vào mọi mặt của cuộc sống. Như xe tự hành của Google và Tesla, hệ thống tự tag khuôn mặt trong ảnh củaFacebook, trợ lý ảo Siri của Apple, hệ thống gợi ý sản phẩm của Amazon, hệ thống gợi ý phim của Netflix, máy chơi cờ vây Alphago của Google DeepMind có thể thắng tuyển thủ cờ vây quốc tế với 18 lần vô địch Lee Sebol, hay vụ nổi tiếng nhất đầu khoảng đầu năm 2019 là phần mềm Deep fake có sử dung AI để cắt ghép khuôn mặt nhân vật trong phim với độ chính xác cao v.v..

Nếu nói AI là mục tiêu của con người thì công cụ hiện nay được kỳ vọng để chinh phục mục tiêu là Machine Learning (ML). ML được biết đến lầu tiền qua thuật toán Perceptrong được phát minh ra bởi Frank Rosenblatt năm 1957. Về cơ bản thì ML là ứng dụng các thuật toán để phân tích cú pháp dữ liệu, học hỏi từ nó, và sau đó thực hiện một quyết định hoặc dữ đoán về các vấn đề có liên quan. Nên máy có khả năng tự học hỏi bằng cách sử dụng một lượng lớn dữ liệu và các thuật toán cho phép nó học cách thực hiện các tác vụ. ML giúp con người đi rất xa trong việc chinh phục AI nhưng vẫn con rất xa nữa con người mới có thể chinh phuc được nó. Hiện tại ML chỉ tập trung vào những mục tiêu ngắn. Như làm cho máy tính có khả năng nhận thức cơ bản của người như nghe, nhìn, hiểu được ngôn ngữ, giải toán, v.v.. Hỗ trợ con người trong việc xử lý một khối lượng thông tin khổng lồ một cách nhanh chóng và chính xác như Big Data

ML có mối quan hệ rất mật thiết đối với mô hình thống kê. ML sử dụng các mô hình thống kể để ghi nhớ lại sự phân bố của dữ liệu ngoài ra nó phải tổng quát hóa những gì đã được nhìn thấy và đưa ra dự đoán nho những trường hợp chưa được nhìn thấy.

Một phương pháp tiếp cận thuật toán khác ML là Deep Learning (DL) sử dụng những kiến trúc Artificial Neural Networks (ANN) được giới thiệu bởi Masahiko Fukushima vào năm 1980.

ANN được lấy cảm hứng từ nơ-ron sinh học của bộ não người. Nhưng các lớp nơ-ron ở não người thì bát kỳ lớp no-ron nào cũng được liên kết với nhau thông qua bó dây thần kinh một cách chặt chẽ thì ANN chỉ là các lớp, các kết nối và các hướng truyền dữ liệu rời rạc. Ví dụ khi nhận biết một bức ảnh con mèo thì bức ảnh được tách làm nhiều phần và được đưa vào lớp nơ ron đầu tiền phân nhóm và đưa vào lớp no-ron thứ 2 để làm nhiệm vụ của nó cứ như thế cho đến khi đến lớp cuối cùng thì nó cho ra được kết quả cuối cùng. Mỗi no-ron của ANN đảm nhiêm một chức nặng để biết chính xác liệu rằng nó có liên quan đến nhiệm vụ đang được thực hiện.

Với DL thì AI có được một tương lai tươi sáng nó cho phép ứng dụng nhiều vấn đề thực tế của máy học và bằng cách mở rộng lĩnh vực tổng thể của AI. DL làm cho các loại máy móc trợ giúp có thể thực hiện được, gần hoặc giống hết con người

AI sử dụng mức độ tính toán rất cao ngay cả với các thuật toán cơ bản đặc biệt là với DL. AI được xây dựng thông qua ngôn ngữ lập trình với nhu cầu tính toán sử dụng các thuật toán tinh vi để giải quyết vấn đề này thì python là ngôn ngữ phù hợp. Python có cú pháp dễ đọc, thư viện phong phú hữu ích có thể được sử dụng trong

AI như Tensorflow, Numpy, Scypy, pybrain, được sử dụng phổ biến nhất trong linh vực AI.

Điều làm Python được nhiều người ưu ái đến vậy vì nó có khải năng khai thác hâu như mọi thứ từ thư viện Tensorflow. Tensorflow là thư viện mã nguồn mở dành cho AI. Tensorflow thuộc thế hệ thứ 2 của Distbelief được phát hành vào 9/11/2015 của Google Brain. Bắt nguồn từ nhu cầu của google trong việc xây dựng hệ thống mạng no-ron với mục đích là để huấn luyện phát triển và giải mã các mẫu và các mối tương quan. Tesorflow có thể chạy trên nhiều CPU và GPU với nhiều tùy chọn cho việc tính toán đa năng trên GPU. Các tính toán của Tensorflow được thể hiện dưới dạng các biểu đồ dataflow chi tiết. Nhiều nhóm của google đã chuyển từ Distbelief sang TensorFlow để phục vụ nghiên cứu và sản xuất

Đến nay AI được áp dụng trong lĩnh vực. Như xe tự hành của Google và Tesla, hệ thống tự tag khuôn mặt trong ảnh của Facebook, trợ lý ảo Siri của Apple hay Google assinstant của của google, hệ thống phân tích nhu cầu người dùng và vận hành cửa hàng tại Amazon go, Nhân diện khuôn mặt của iphone, v.v..

Còn trong lĩnh vực an toàn thông tin thì AI cũng bắt đầu được cac tổ chực sử dụng để tăng cường an ninh mạng cung cấp nhiều biện pháp bảo vệ chống lại các tin tắc tinh vi. Nó có thể tự động hóa các quy trình phức tạp để phát triển và phản ứng với các vị phạm trái phép. Các ứng dụng này ngày nay trở nên hữu dụng và an toàn hơn ki AI được triển khai để bảo mật.

Các sản phẩm công nghệ ứng dụng bảo mật AI có thể tự động phát hiện. Phân tích và bảo vệ chống lại các cược tấn ông nâng cao bằng cách chủ động phát hiện và ngăn chặn những kẻ tấn công.

Các lỗ hổng, cách thức tấn công mới có thể được cập nhập lên hệ thống huấn luyện để máy có thể học một cách nhanh chóng.AI ũng có thể tìm ra lỗ honhr của ứng dung – thứ giúp hacker tân công, khai thác và đánh cắp thông tin

Những hacker cũng bắt đầu triển khai AI, sẽ dần xuất hiện các cộng cụ tấn công tự đông, Có khả năng nghiên cứu và tìm kiếm các hệ thống đươc nhắm đến làm mục tiêu và xác định các lỗ hỏng ngay lập tức

II. QUÁ TRÌNH PHÂN TÍCH

1. Cấu trúc của tệp tin ứng dụng trên Android

Ứng dụng, game trên Android, sau đây sẽ sử dụng cụm từ "ứng dụng android" để mô tả chung về ứng dụng, game trên android, được cài đặt trong các dạng gói tin ứng dụng hay còn được gọi là APK files (Android Package Kit). Tệp tin APK này bao gồm cả code của ứng dụng, các tài nguyên (hình ảnh, xâu kí tự,..) và tệp manifest. Hiểu một cách đơn giản, APK chỉ đơn giản là một tệp tin ZIP và chúng ta có thể xem nội dung bên trong của nó bằng cách giải nén bằng bất kì công cụ giải nén thông dụng nào hỗ trợ định dạng .ZIP.

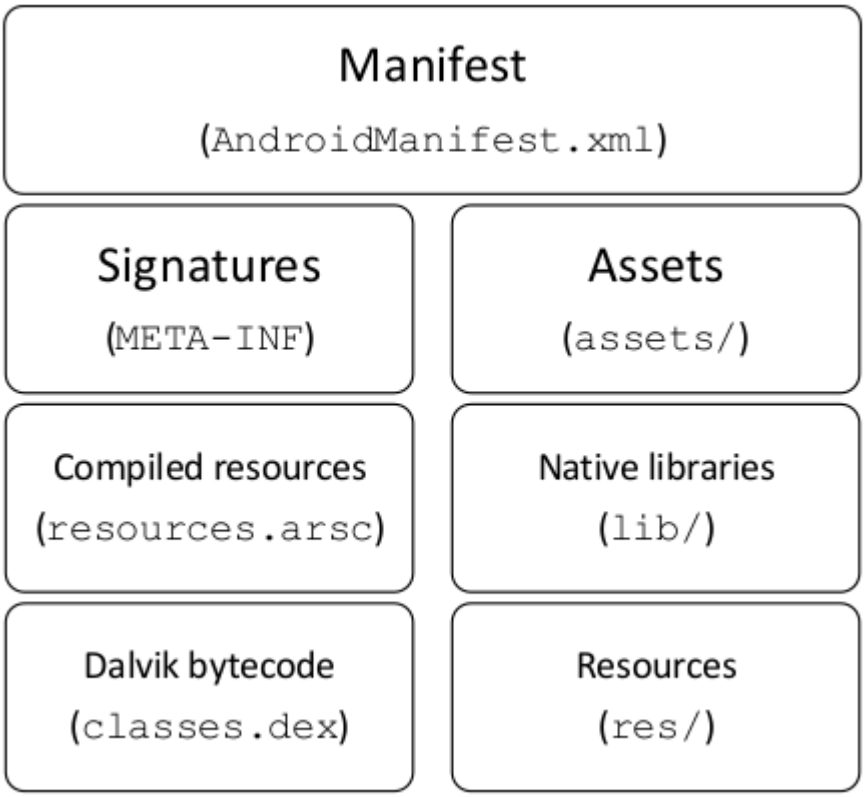


Figure 2: Cấu trúc bên trong của một tệp tin APK

Tên	Chú thích
AndroidManifest.xml	Tệp tin manifest ở dạng xml
Classes.dex	Code của ứng dụng được dịch sang dạng .dex
Resources.arsc	Tệp tin chứa các tài nguyên của ứng dụng, được sử dụng trong quá trình tiền biên dịch, ở dạng XML
Res/	Thư mục chứ tài nguyên không cần biên dịch
Assets/	Một thư mục tùy biến, chứa tài sản của ứng dụng, những tài sản được trích xuất ra bởi AssetManager
Lib/	Một thư mục tùy biến, chứ mã được biên dịch của ứng dụng, ngôn ngữ ở dạng C/C++
META-INF/	Thư mục chứ tệp tin MANIFEST.MF, nơi lưu trữ dữ liệu về nội dung của JAR, tên thư mục gốc, chữ ký,…

Table 2 Bảng: các thành phần, cấu trúc bên trong của một tệp tin APK.

Thuộc tính	Chú thích
Manifest tag	Chứa chế độ cài đặt, tên gói và phiên bản
Permissions	Tùy biến quyền hạn và định mức bảo vệ
uses-permissions	Yêu cầu một quyền hạn nào đó phải được cấp phép sử dụng để vận hành.
uses-feature	Định nghĩa một đặc tính của phần cứng hoặc phầm mềm đơn được sử dụng bởi ứng dụng
Application	Định nghĩa ứng dụng. Chứa tất cả các “activity” của ứng dụng
Activity	Định nghĩa các hoạt động được mà là một phần của giao diện người dung
intent-filter	Xác định các kiểu của các định nghĩa (intents) như: một hoạt động (activity), một dịch vụ (service),…
service	Định nghĩa dịch vụ như là một trong những thành phần của ứng dụng
receiver	Bộ nhận tín hiệu kích hoạt các ứng dụng để nhận các chỉ thị, định nghĩa từ hệ thống hoặc một ứng dụng khác, bất kể khi các thành phần khác không chạy
provider	Định nghĩa một nội dung của thành phần cung ứng. Một nội dung cung ứng là một lớp nhỏ của ContentProvider, cái cung cấp cấu trúc tiếp cận đến dữ liệu được quản lí bởi ứng dụng

Table 3 Bảng: các thuộc tính bên trong tệp tin Manifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="payspace.ssidit.pp.ua.payspacemagazine">

    <uses-permission android:title="android.permission.INTERNET" />
    <uses-permission android:title="android.permission.ACCESS_NETWORK_STATE" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@android:style/Theme.Holo.Light">
        <activity
            android:label="@string/app_name"
            android:title=".MainActivity">
            <intent-filter>
                <action android:title="android.intent.action.MAIN" />

                <category android:title="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:label="@string/title_activity_settings"
            android:title=".SettingsActivity"></activity>
    </application>

</manifest>
```

Figure 3 Nội dung của một tệp tin Manifest.xml. Chứa các thông tin quan trọng: Activity, Permission,…

2. Phân tích tĩnh

Như đã mô tả ở phần trên, APK là một tệp tin có cấu trúc, các đặc tính quan trọng. Dựa vào cấu trúc và các đặc tính đó, chúng em tiến hành quá trình phân tích tĩnh với quá trình sau:

1. Thu thập mẫu.
2. Phân tích tập mẫu để tìm ra các đặc trưng riêng.
3. Tự động hóa quá trình trích xuất đặc trưng. Xây dựng các bộ đặc trưng.
4. Tiến hành chọn mô hình học máy và học sâu phù hợp
5. Chạy thực nghiệm, chọn mô hình và tập mẫu tối ưu nhất

Chi tiết quá trình phân tích tĩnh (chỉ bao gồm bước 1 đến bước 3):

1. Thu thập mẫu:

Hệ điều hành Android là một hệ điều hành phổ biến, và đang không ngừng phát triển mạnh. Vì vậy, cũng không quá khó khăn để chúng em thu thập mẫu trong quá trình phân tích. Về mã sạch, chúng em sử dụng các dụng dụng và game bình thường được đăng ký trên Play Store. Mỗi loại chúng em chọn ra từ 50-100 mẫu. Mục đích là làm mẫu sạch và gán nhãn “apps” và “games” cho các mẫu này. Về mã độc, chúng em sử dụng các tệp mã độc do Drebin Lab thu thập, chúng em chỉ sử dụng các mẫu mã độc và không tham khảo thêm về bộ datasets do Drebin Lab cung cấp. Để đảm bảo cho quá trình học thuận lợi, tối ưu khả năng chỉ số accuracy cao nhất. Chúng em nhận ra khi loại bỏ các nhãn có ít hơn 4

mẫu thì mô hình train cho kết quả cao nhất accurency: 0.96. Phân loại các nhãn chia ra từng thư mục ứng với tên của từng họ mã độc, và bỏ tất cả vào chung một thư mục để đảm bảo quá trình tự động hóa.

3. Phân tích mẫu, tìm đặc trưng riêng.

Dựa vào cấu trúc của tệp tin APK (Mục TỔNG QUAN, BỐI CẢNH), chúng ta thấy mỗi đặc điểm của tệp tin đều có thể là một đặc trưng. Tuy nhiên, việc lựa chọn đặc trưng nào là một công việc cực kì quan trọng. Vì đó đóng vai trò quan trọng trong việc thành công hay thất bại của đề tài nghiên cứu này. Quay trở lại với việc lựa chọn đặc trưng phù hợp. Đầu tiên, chúng em thấy có những thành phần mà chúng em cho rằng nó có thể là đặc trưng:

- Activites
- Intent
- Providers
- Serivices
- Strings
- APIs
- Declare permission
- Native Library
- Permissions

Sau nhiều lần thử nghiệm và chỉnh sửa. Sau cùng tụi em chọn ra được 4 bộ dữ liệu mẫu ứng với các thông tin sau:

- Bộ 1: Permissions + Declare permissions
- Bộ 2: Permissions + API
- Bộ 3: Permissions + API + Size of file + Declare permission + Nativelibc + Services
- Bộ 4: Permissions + size of file + Is declare new permission + use native libc + number of services + number of exist features

Ứng với mỗi bộ trên, chúng em trích xuất dựa trên 2 bộ mẫu 5 ngàn mẫu độc và 11 ngàn mẫu độc + sạch.

4. Tự động hóa quá trình trích xuất đặc trưng. Tạo ra các tập dữ liệu mẫu như đã thiết kế.

Thao tác với cấu trúc của APK, Androguard Team phát triển một module dành cho python, nhằm hỗ trợ trích xuất dữ liệu từ tệp tin APK một cách đơn giản. Từ module trên, chúng em đã phát triển những đoạn mã python nhỏ nhằm hỗ trợ chúng em trong quá trình trích xuất và tạo dữ liệu mẫu.

5. Quá trình phân tích động:

Bên cạnh việc sử dụng các công cụ hỗ trợ để dịch ngược tệp tin APK, chúng em còn sử dụng các phương pháp phân tích động để tìm kiếm thêm bộ đặc trưng mới. Để phân tích động, chúng em sử dụng chủ yếu là công cụ AndroPyTool (https://github.com/alexMyG/AndroPyTool) với các chức năng được mô tả như hình:

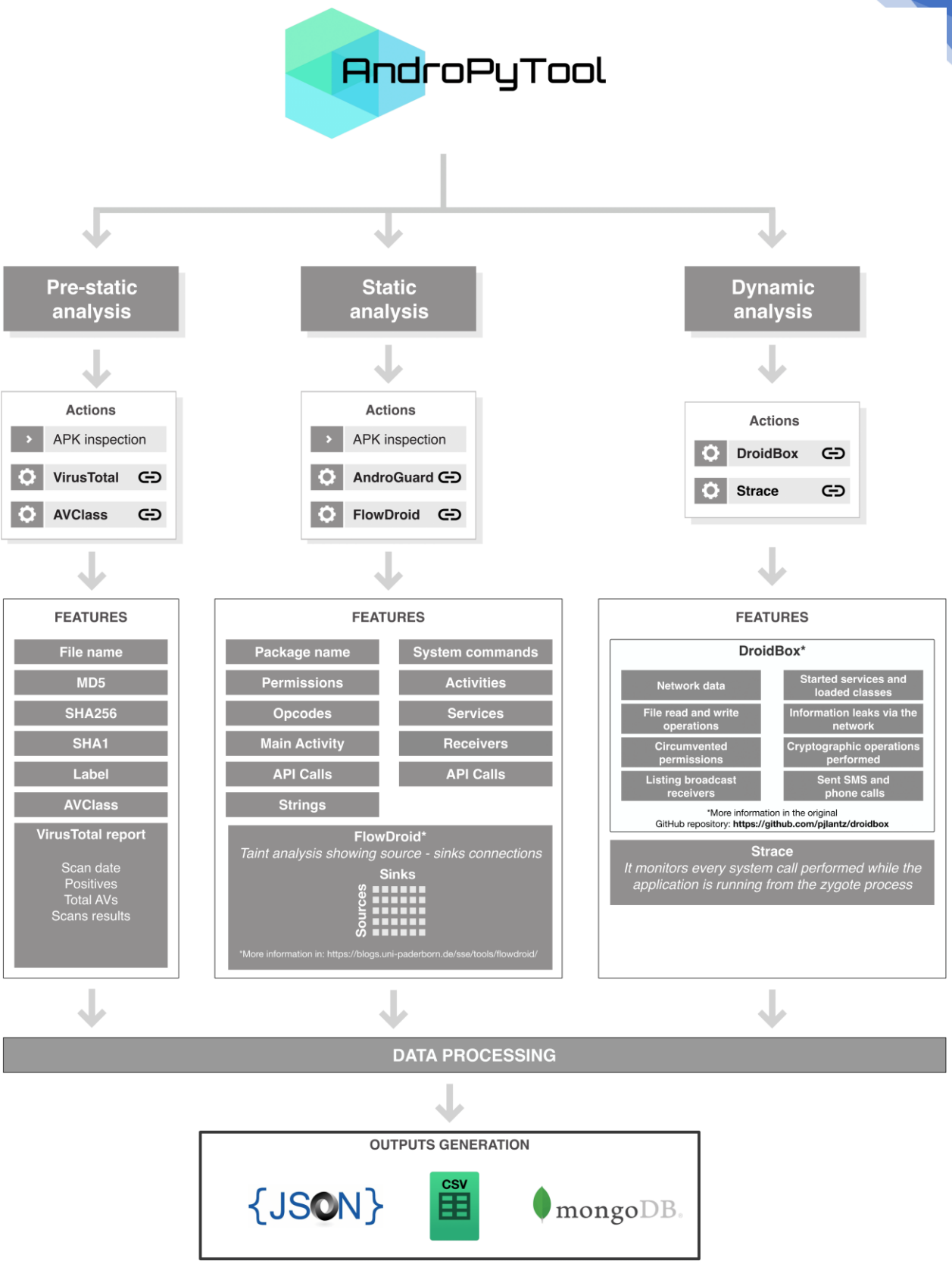


Figure 4 Mô hình, cấu trúc phân tích động của AndroPyTools

Tuy nhiên, do kết quả trả về không phù hợp với ý tưởng nên chúng em đã không tìm được thêm các đặc trưng nào cho bộ dữ liệu mẫu.

III. CNN VÀ CÁC MÔ HÌNH

1. Cấu trúc mạng CNN

Convolutional Neural Network(CNN) là một trong những mô hình deep learning tiên tiến và phổ biến nhất hiện nay, có khả năng nhận dạng và phân loại hình ảnh với độ chính xác rất cao, thậm chí còn tốt hơn con người trong nhiều trường hợp. Mô hình CNN hiện vẫn đang được phát triển, ứng dụng vào các hệ thống ảnh lớn cả facebook, Google, Amazon,v.v .. cho các mục đích khác nhau như các thuật toán tag tự động, tìm kiếm hoặc gợi ý sản phẩm cho người tiêu dùng.

Sự ra đời của mạng CNN là dựa trên ý tưởng cải tiến cách thức mạng nơ-ron nhân tạo truyền thống.

Kiến trúc cơ bản trong một mạng CNN bao gồm:

- Lớp tích chập (Convolutional): Đây là thành phần quan trọng nhất trong mạng CNN, cũng là nới thể liên kết cục bộ thay vì kết nối toàn bộ các điểm. Các liên kết cục bộ này được tính toán bằng phép tích chập ta có thể xem nó như là một cửa sổ trượt
- Lớp kích hoạt phi tuyến ReLu: Lớp này được xây dựng với ý nghĩa đảm bảo tính phi tuyến của mô hình huấn luyện sau khi đã thực hiện một loại các phép tính toán tuyến tích qua các lớp Tích chập. Lớp Kích hoạt phi tuyến noi chung sử dụng các hàm kích hoạt ReLu hoặc digmoid,tand.. để giơi hạn phạm vị biên độ cho phép của đầu ra. Lớp kết nối đầu đủ này được thiết kế hoàn toàn tương tự như mạng nơ-ron tuyền thống.

Các lớp được thay đổi về số lượng và cách sắp xếp để tạo ra các mô hình huấn luyện phù hợp cho từng bài toán khác nhau. Các lớp được liên kết với nhau thông qua cơ chế convolution. Lớp tiếp là kết quả convolution từ layer trước đó, nhờ vậy mà ta có được các kết nối cục bộ.

Mỗi no-ron ở lớp tiếp theo ra từ bộ lọc áp đặt lên một vùng cục bộ của nơ-ron layer trước đó.

Mạng no-ron sử dụng 3 ý tưởng cở bản:

- Local receptivefields (Trường tiếp nhận cục bộ): Trong các hệ thông kết nối đầy đủ trước đây, đầu vào mô ra là một đường thảng đứng chứa các no-ron. Trong CNN đầu vào 28 x 28 no-ron. Kết nối đầu vào cho các no-ron ở tầng ẩn. Các no-ron trong lớp ẩn đầu tiên sẽ được kết nối với một vùng nhỏ có ác no-ron đầu vào được gọi là vùng tiếp nhận cục bộ co no-ron ẩn. Mỗi kết nối sẽ học một trọng số và no-ron ẩn cũng học một độ lệch. Có thể hiêu rằng no-ron lớp ẩn cụ thể là học để phân tích trường tiếp nhận cục bộ của nó. Trượt trường tiếp nhận cục bộ trên toàn bộ dữ liệu. đối với môi trường tiếp nhật cục bộ, có một no-ron ẩn khác trong lớp ẩn đầu tiên.
- Shared weithts and biases (Trọng số và độ lệch): mỗi một no-ron ẩn cố một dộ lệch và trọng soos liên kết với trường tiếp nhận cục bộ. Sử dụng cung cho mỗi no-ron ẩn 24x24. Nói cách khác, đói với những no-ron ẩn thứ m,n đầu ra là:

$$\sigma(b + \sum_{i=0}^{4}\sum_{j=0}^{4} \omega_{i,j} a_{m+i,n+j})$$

Trong đó σ là hàm kích hoạt noron, b là giá trị chung cho độ lệch. i,j là một mảng 5x5 của trọng số chia sẻ. $a_{m+i,n+j}$ là biể thị kích hoạt giá trị đầu vào tại vị chí m+i ,n+j

- Pooling layer (Lớp chứa hay lớp tổng hợp): ngoài các lớp tích chập vừa mô ta, mạng no-ron tích chạp cũng chứa các lớp pooling. Lớp pooling thường được sử dụng ngay sau lớp tích chập. những gì các lớp pooling làm là đơn giản hóa các thông tin ở đầu ra từ các lớp tích chập. Mỗi đơn vi trong lớp pooling có thể thu gon một vùng trong no-ron lop trước. Một thủ túc pooling phổ biến là max-pooling. Trong max-pooling, một đơn vị pooling chỉ đơn giản là kết quả đầu ra kích hoạt giá trị lớn nhất trong vùng đầu vào nếu chúng ta có 24x24 no-ron đầu ta từ các lớp tích chập, sau khi pooling chung ta có là 12x12 no-ron. Lớp tích chập thường có nhiều hơn một bản đồ đặc trưng. Chúng ta áp dụng max-pooling cho mỗi bản đồ đặc trưng riêng biệt. Có thể hiệu max-pooling nhu là một cách cho mạng để hỏi xem một đặc trưng nhất được thấy ở bất cứ đâu trong một khu vực.

Bây giờ chúng ta có thể đặt tất cả những ý tưởng lại với nhau để tạo thành một CNN hoàn chỉnh. Nó tương tự như kiến trúc chún ta nhìn vào, nhưng có thêm một lớp 10 no-ron đầu ra, tương ứng với 10 giá trị có thể cho các MNIST. Lớp cuối cùng của các kết nối trong CNN là một lớp đầu đủ kết nối. Đó là, lớp này nối mọi no-ron từ lúc maxpooling tới mọi no-ton tần ra.

Mạng CNN được cấu tạo từ 3 loại layer chính sau: Convolution Layer, Pooling Layer, Fully Connected Layer.

Convolution Layers (Lớp tích chập)

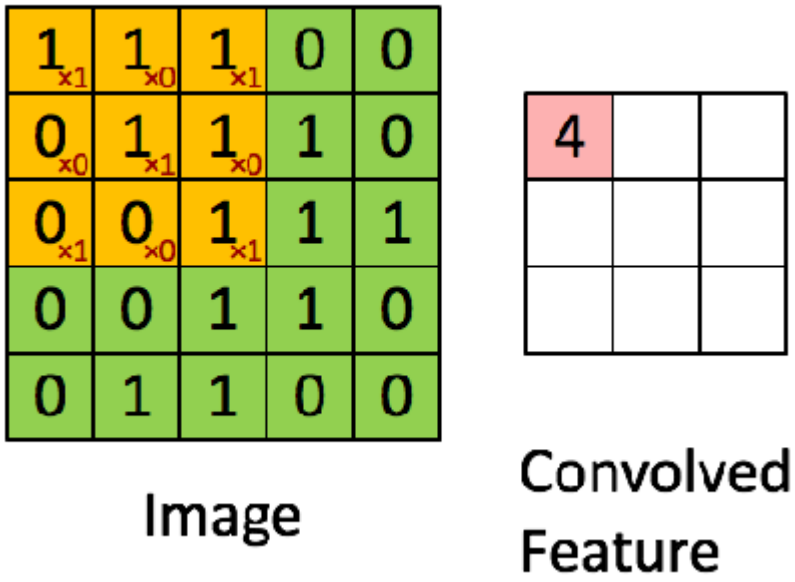


Figure 5 Convolved Feature

1x1 + 1x0 + 1x1 + 0x0 + 1x1 + 1x0 + 0x1 + 0x0 + 1x1 = 4

Ma trận 3x3: [[1, 0, 1], [0, 1, 0], [1, 0, 1]] là bộ filter của lớp Convolution. Filter này sẽ được tạo tự động. Mục đích chính của Convolution là trích xuất các đặc trưng từ tập dữ liệu ta đưa vào.

Pooling Layer:

Max Pooling và Average Pooling:

Pooling thường xuất hiện phía sau các Convolution Layer để làm giảm đi số neuron, giảm số lượng tham số mô hình mà chúng ta cần tính toán. Có 2 dạng Pooling thường được sử dụng là Max Pooling và Average Pooling.

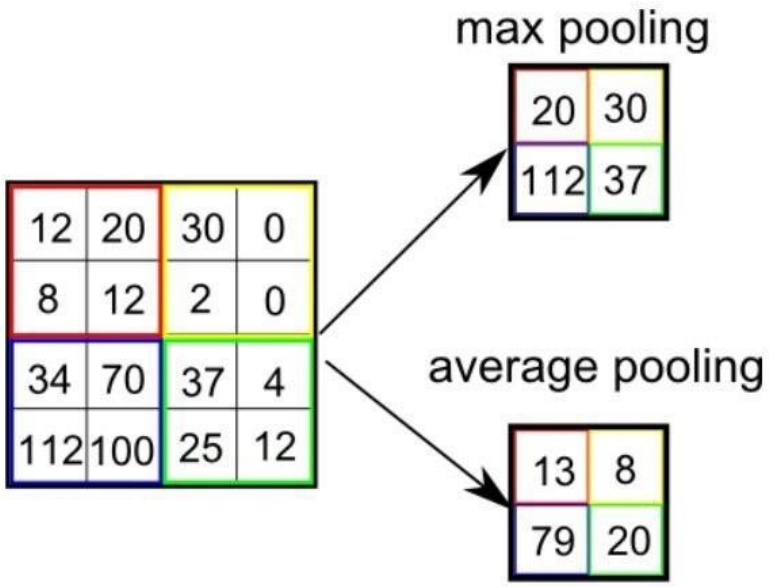


Figure 6 Hình ảnh lớp Pooling

Fully Connected Layer:

Các neuron của layer phía sau sẽ liên kết nối đầy đủ với toàn bộ neuron của layer phía trước.

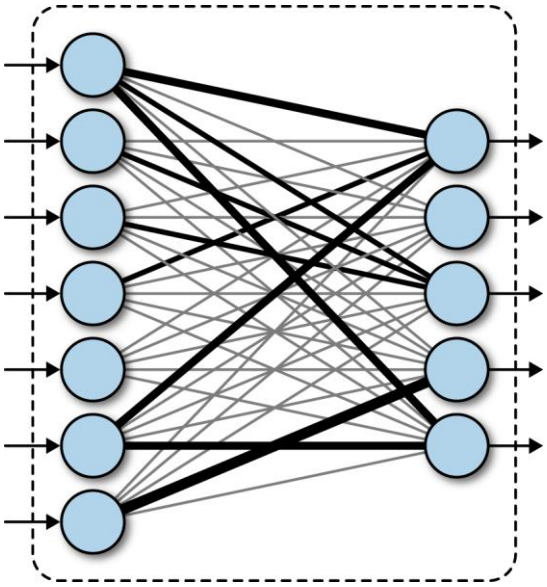


Figure 7 Mạng neutron

Activation: RELU và Softmax.

Activation function được sử dụng để chuẩn hóa output của mỗi layer. Ở đây ta sử dụng activation func: RELU có công thức như sau:

$f(x)=\max(0,x)$ với x là giá trị của mỗi neuron. **f(x) = 0** khi **x < 0** và **f(x) = x** khi **x > 0**.

Xem thêm: https://machinelearningcoban.com/2017/02/24/mlp/#-activation-functions

Softmax thường được sử dụng ở layer cuối cùng để đưa ra cho chúng ta kết quả phân bố xác suất trên các nhãn mà chúng ta dự đoán.

Loss function:

Hiểu nôm na, model sẽ dựa vào Loss function để cập nhật lại bộ weights sao cho kết quả của mỗi lần dự đoàn gần kết quả thực nhất.

Categorical Cross Entropy (CSE) thường được sử dụng trong các bài toàn phân loại.

Categorical crossentropy math

$$L(y,\hat{y}) = -\sum_{j=0}^{M}\sum_{i=0}^{N}(y_{ij} * log(\hat{y}_{ij}))$$

Trong đó **y** là nhãn thực, **y** mũ là nhãn được dự đoán.

Model CNN phân loại mã độc:

2 lớp Conv với filter lần lượt là 32, 64

1 Lớp Max Pooling với pool_size = 2, strides=2

2 Lớp Conv với filter lần lượt là 64, 128

1 Lớp Average Pooling với pool_size = 2, strides=2

1 Lớp Flatten

1 Lớp Fully Connected với units = 256, activation='RELU'

1 Lớp Dropout

1 Lớp Fully Connected với units là số nhãn ta huấn luyện. Activation 'Softmax'

Tổng tham số mô hình: 3,441,625

Training model đến chu kỳ 294 thì loss về 0.05149 và kết quả khi dự đoán tập validation là 0.93. Các chu kỳ tiếp theo gia trị loss không thay đổi nên dừng lại.

Epoch 00294: loss did not improve from 0.05149

139/139 [==============================] - 2s 14ms/step - loss: 0.1829 - acc: 0.9694 - val_loss: 0.4042 - val_acc: 0.9302

Kết quả khi dự đoán trên tập test là : **0.93381**

Tính True Positive, True negative, False Positive, False Negative.

Xét tập dữ liệu ta có: (87: apps) và (88: games) là hai nhãn mã sạch. Còn lại là các nhãn mã độc.

TP: Là các nhãn mã sạch được dự đoán đúng.

TN: Là các nhãn mã độc được dự đoán độc.

FP: Là các nhãn mã độc nhưng thành mã sạch.

FN: Là các nhãn mã độc nhưng thành mã độc khác.

TP = 30, TN = 502, FP = 0, FN = 29.

Tỷ lệ số điểm TP trong những điểm được phân loại Positive:

Precision = TP / (TP + FP)= 1

Tỷ lệ số điểm TP trong những điểm thực sự là Positive:

Recal = TP / (TP + FN)= 0.51

2. SVM - Super Vector Machine

Là một thuật toán thuộc nhóm "học có giám sát", dùng để phân loại dữ liệu đầu vào.

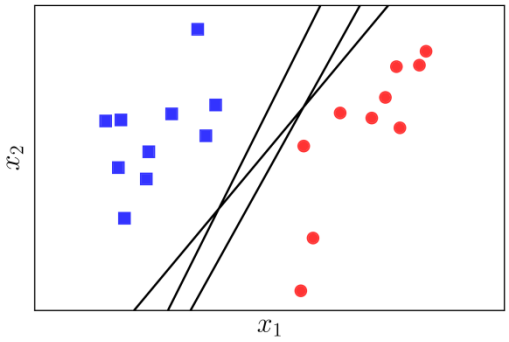


Figure 8 Ảnh tượng trưng

Giả sử ta có 2 loại dữ liệu là xanh và đỏ như hình vẽ. Mục đích của chúng ta là tìm ra mặt phân chia 2 loại dữ liệu đó sao cho kết quả tốt nhất và công bằng nhất. SVM chính là tìm mặt phẳng phân chia sao cho thỏa được mục đích của bài toán nêu ra.

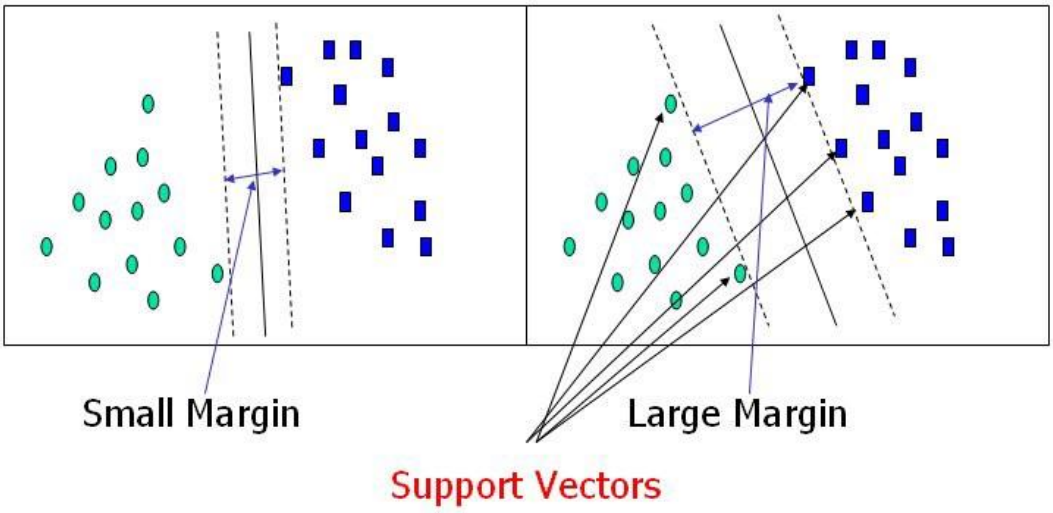


Figure 9 Ảnh tượng trưng

Trong thuật toán SVM, chúng ta đi tìm Margin lớn nhất giữa các điểm dữ liệu với siêu mặt phẳng và hàm mất mát sẽ giúp ta thực hiện điều đó.

$$c(x, y, f(x)) = \begin{cases} 0, & \text{if } y * f(x) \geq 1 \\ 1 - y * f(x), & \text{else} \end{cases}$$

Cost = 0 nếu giá trị dự đoán và giá trị thực tế cùng dấu, còn không thì chúng ta sẽ tính lại giá trị mất mát. Trong thuật toán SVM sử dụng Gradient để cập nhật lại các trọng số trong tính toán giá trị mất mát.

3. RandomForest

- Decision Tree - Cây quyết định có thể hiểu là một đồ thị của các quyết định và hậu quả của nó, độ lớn của cây tùy thuộc vào cài đặt của chúng ta. Trong ML, Decision Tree là một kiểu mô hình dự báo, mỗi một node là đại diện cho một dự báo.

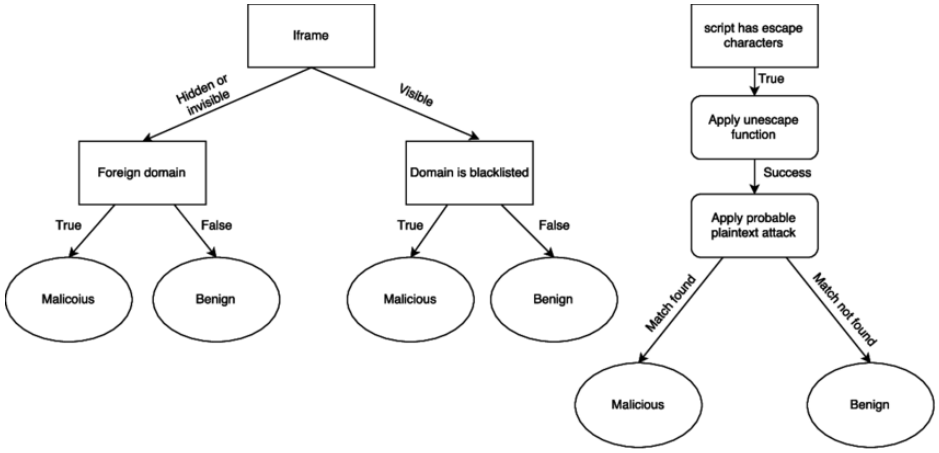


Image 1: Ví dụ về Decision Tree

- RandomForest là một tập hợp các Decision Tree lại với nhau, và có chức năng như Decision Tree nhưng ta sẽ có nhiều "ý kiến" hơn. RF hoạt động bằng cách đánh giá DT sử dụng các thức voting để đưa ra kết quả cuối cùng.

- Về mặt học thuật, RF là tập hợp nhiều DT lại với nhau, mỗi DT được tạo nên ngẫu nhiền từ việc tái chọn mẫu và random các biến trong data. Tuy nhiên, RF là một trong những phương thức BlackBox nên ta không thể biết được bên trong nó khi hoạt động sẽ như thế nào.

IV. THỰC NGHIỆM

1. Phân tích đặc trưng, xây dựng tập mẫu

Từ quá trình phân tích đã nêu tại mục II, chúng ta có được 8 bộ dữ liệu:

- Bộ hơn 11 ngàn mẫu:
  - Bộ 1: Permissions + Declare permissions
  - Bộ 2: Permissions + API
  - Bộ 3: Permissions + API + Size of file + Declare permission + Nativelibc + Services
  - Bộ 4: Permissions + size of file + Is declare new permission + use native libc + number of services + number of exist features
- Bộ hơn 5 ngàn mẫu:
  - Bộ 1: Permissions + Declare permissions
  - Bộ 2: Permissions + API
  - Bộ 3: Permissions + API + Size of file + Declare permission + Nativelibc + Services
  - Bộ 4: Permissions + size of file + Is declare new permission + use native libc + number of services + number of exist features

Từ 8 bộ dữ liệu trên, tiến hành training cho các mô hình đã nêu trên mục III.

2. Tiến hành training trên mô hình CNN.

Code mô hình:

```
# -*- coding: utf-8 -*-
"""Drebin.ipynb

Automatically generated by Colaboratory.

Original file is located at
https://colab.research.google.com/drive/1C33cLzK5XIQy8_7PxwmpRMmATUqJi4oq
"""

from google.colab import drive
drive.mount('/content/drive')

import pandas as pd
import numpy as np
from copy import copy

from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
import tensorflow as tf
from tensorflow.keras.callbacks import ModelCheckpoint
```

```
from matplotlib import pyplot as plt
import json



filepath = "/content/drive/My Drive/Colab Notebooks/set-2-
5k.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor="loss",
verbose=1, mode="min",
save_best_only=True,
save_weights_only=True,
)


df = pd.read_csv('/content/drive/My Drive/Colab
Notebooks/Data/5K/Set 2/api_perm_declareperm_v1_5k.csv',
header=None)


map_num = {'0_apps': 0, 'AccuTrack': 1, 'Ackposts': 2,
'Acnetdoor': 3,
'Adrd': 4, 'Adsms': 5, 'Aks': 6, 'Ansca': 7, 'Antares': 8,
'Anti': 9,
'Anudow': 10, 'Arspam': 11, 'BaseBridge': 12, 'BeanBot': 13,
'Bgserv': 14, 'Biige': 15, 'Booster': 16, 'Bosm': 17, 'Boxer':
18,
'Cawitt': 19, 'CellShark': 20, 'CellSpy': 21, 'Ceshark': 22,
'CgFinder': 23, 'Coogos': 24, 'Copycat': 25, 'Cosha': 26,
'CrWind': 27, 'Dabom': 28, 'Dialer': 29, 'Dogowar': 30,
'Dougalek': 31, 'DroidDream': 32, 'DroidKungFu': 33,
'DroidRooter': 34, 'DroidSheep': 35, 'EICAR-Test-File': 36,
'EWalls': 37, 'Exploit.RageCage': 38, 'ExploitLinuxLotoor': 39,
'Faceniff': 40, 'FakeDoc': 41, 'FakeFlash': 42, 'FakeInstaller':
43,
'Fakelogo': 44, 'FakeNefix': 45, 'Fakengry': 46, 'FakePlayer':
47,
'FakeRun': 48, 'FakeTimer': 49, 'Fakeview': 50, 'FarMap': 51,
'Fatakr': 52, 'Fauxcopy': 53, 'Fidall': 54, 'FinSpy': 55,
'Fjcon': 56, 'Flexispy': 57, 'FoCobers': 58, 'Foncy': 59, 'Fsm':
60,
'Fujacks': 61, 'Gamex': 62, 'Gapev': 63, 'Gappusin': 64,
'Gasms': 65,
'Geinimi': 66, 'Generic': 67, 'GGtrack': 68, 'GinMaster': 69,
'GlodEagl': 70, 'Glodream': 71, 'Gmuse': 72, 'Gonca': 73,
'GPSpy': 74, 'Hamob': 75, 'Hispo': 76, 'Iconosys': 77, 'Imlog':
78,
```

```
'Jifake': 79, 'JS': 80, 'Exploit-DynSrc': 81, 'JSmsHider': 82,
'Kidlogger': 83, 'Kiser': 84, 'Kmin': 85, 'Koomer': 86, 'Ksapp':
87,
'Lemon': 88, 'LifeMon': 89, 'Loicdos': 90, 'Loozfon': 91,
'Luckycat': 92, 'Lypro': 93, 'Maistealer': 94, 'Mania': 95,
'Maxit': 96, 'MMarketPay': 97, 'Mobilespy': 98, 'MobileTx': 99,
'Mobinauten': 100, 'Mobsquz': 101, 'Moghava': 102,'MTracker':
103,
'Nandrobox': 104, 'Netisend': 105, 'Nickspy': 106, 'NickyRCP':
107,
'Nisev': 108, 'Nyleaker': 109, 'Opfake': 110, 'PdaSpy': 111,
'Penetho': 112, 'Pirater': 113, 'Pirates': 114, 'PJApps': 115,
'Placms': 116, 'Plankton': 117, 'Proreso': 118, 'Qicsom': 119,
'QPlus': 120, 'Raden': 121, 'RATC': 122, 'RediAssi': 123,
'Replicator': 124, 'Rooter': 125, 'RootSmart': 126, 'RuFraud':
127,
'SafeKidZone': 128, 'Saiva': 129, 'Sakezon': 130, 'Sdisp': 131,
'SeaWeth': 132, 'SendPay': 133, 'SerBG': 134, 'SheriDroid': 135,
'SmForw': 136, 'SMSBomber': 137, 'Smspacem': 138, 'SMSreg': 139,
'SMSSend': 140, 'SmsSpy': 141, 'SmsWatcher': 142, 'SMSZombie':
143,
'Sonus': 144, 'Spitmo': 145, 'Spy.GoneSixty': 146, 'Spy.ImLog':
147,
'SpyBubble': 148, 'SpyHasb': 149, 'SpyMob': 150, 'Spyoo': 151,
'SpyPhone': 152, 'Spyset': 153, 'Ssmsp': 154, 'Stealer': 155,
'Stealthcell': 156, 'Steek': 157, 'Stiniter': 158, 'SuBatt':
159,
'Tapsnake': 160, 'Tesbo': 161, 'TheftAware': 162, 'TigerBot':
163,
'Trackplus': 164, 'TrojanSMS.Boxer.AQ': 165,
'TrojanSMS.Denofow': 166, 'TrojanSMS.Hippo': 167,
'TrojanSMS.Stealer': 168, 'Typstu': 169, 'Updtbot': 170,
'UpdtKiller': 171, 'Vdloader': 172, 'Vidro': 173, 'Whapsni':
174,
'Xsider': 175, 'YcChar': 176, 'Yzhc': 177, 'Zitmo': 178,
'Zsone': 179}
new_map = {value: name for name, value in map_num.items()}

for key in new_map:
df[0][np.where(df[0] == key)[0]] = new_map[key]

df = df[df[0].isin(df[0].value_counts()[df[0].value_counts() >
3].index)]
# labels = df[0].values
labels_name = df[0].tolist()
```

```
df.drop(0, axis=1, inplace=True)

label_encoder = LabelEncoder()
label_encoder.fit(list(set(labels_name)))
labels = label_encoder.transform(labels_name)

dict_label = {}
for idx, label in enumerate(labels):
try:
dict_label[str(label)].add(labels_name[idx])
except:
dict_label[str(label)] = set()
dict_label[str(label)].add(labels_name[idx])
dict_label = {str(key): list(label)[0] for key, label in
dict_label.items()}
with open("dict_label4.json", "w") as f:
json.dump(dict_label, f)

X_train, X_test, y_train, y_test = train_test_split(df.values,
labels, test_size=0.2, shuffle=True)
X_val, X_test, y_val, y_test = train_test_split(X_test, y_test,
test_size=0.5, shuffle=True)
print(X_train.shape, X_test.shape, X_val.shape, y_train.shape,
y_test.shape, y_val.shape)
X_train = X_train.reshape(-1, X_train.shape[1], 1)
X_val = X_val.reshape(-1, X_val.shape[1], 1)
X_test = X_test.reshape(-1, X_test.shape[1], 1)
print(X_train.shape, X_test.shape)

y_true = copy(y_train)

y_train = tf.keras.utils.to_categorical(y_train,
num_classes=labels.max() + 1)
y_val = tf.keras.utils.to_categorical(y_val,
num_classes=labels.max() + 1)
print(y_train.shape, y_val.shape)

def fit_generator(features, labels, batch_size):
batch_features = np.zeros((batch_size, 403, 1))
batch_labels = np.zeros((batch_size, 88))
while True:
for i in range(batch_size):
index = np.random.choice(len(features), 1)
batch_features[i] = features[index]
```

```
    batch_labels[i] = labels[index]
    yield batch_features, batch_labels

def model_basic():
    input_ = tf.keras.layers.Input(shape=[403, 1])

    conv_1 = tf.keras.layers.Conv1D(filters=32, kernel_size=3,
    activation='relu', padding='same')(input_)
    conv_2 = tf.keras.layers.Conv1D(filters=64, kernel_size=3,
    activation='relu', padding='same')(conv_1)
    max_pool1 = tf.keras.layers.MaxPool1D(pool_size=2, strides=2,
    padding='same')(conv_2)
    conv_3 = tf.keras.layers.Conv1D(filters=64, kernel_size=3,
    activation='relu', padding='same')(max_pool1)
    conv_4 = tf.keras.layers.Conv1D(filters=128, kernel_size=3,
    activation='relu', padding='same')(conv_3)
    avg_pool = tf.keras.layers.AvgPool1D(pool_size=2,
    strides=2)(conv_4)

    flatten = tf.keras.layers.Flatten()(avg_pool)
    fc_1 = tf.keras.layers.Dense(units=256,
    activation='relu')(flatten)
    do_0 = tf.keras.layers.Dropout(rate=0.2)(fc_1)
    fc_2 = tf.keras.layers.Dense(units=88,
    activation='softmax')(do_0)

    model = tf.keras.models.Model(inputs=input_, outputs=fc_2)

    return model

model = model_basic()
model.summary()
tf.keras.utils.plot_model(model, to_file='/content/drive/My
Drive/Colab Notebooks/model_basic.png')
csv = tf.keras.callbacks.CSVLogger('set-4-log.csv', append=True,
separator=',')
# model.load_weights("/content/drive/My Drive/Colab
Notebooks/model_basic-2.hdf5")
model.compile(
loss=tf.keras.losses.categorical_crossentropy,
optimizer=tf.keras.optimizers.Adam(),
metrics=['accuracy']
)

model.fit_generator(fit_generator(X_train, y_train, 32),
```

```
steps_per_epoch=len(X_train)//32, epochs=1000,
callbacks=[checkpoint, csv], validation_data=(X_val, y_val))

# model.save_weights("/content/drive/My Drive/Colab
Notebooks/set-1.hdf5")

predict = model.predict(X_test)
y_pred = np.argmax(predict, axis=1)

TP = 0
TN = 0
FP = 0
FN = 0
for pred, true in zip(y_pred, y_test):
if true == pred:
if pred == 0:
TP += 1
else:
TN += 1
else:
if pred == 0:
FP += 1
else:
FN += 1

with open("report-set-4.csv", "w", encoding='utf-8') as f:
f.write("Data co tong tong 88 nhan.\n")
f.write(f"Do chinh xac:,{accuracy_score(y_pred, y_test)}\n")
f.write("Các nhãn thực,")
f.write(",".join([new_map[label] for label in y_test]))
f.write("\n")
f.write("Các nhãn dự đoán được,")
f.write(",".join([new_map[label] for label in y_pred]))
f.write("\n")
f.write(f"True Positive:,{TP}.\n")
f.write(f"True Negative:,{TN}.\n")
f.write(f"False Positive:,{FP}.\n")
f.write(f"False Negative:,{FN}.\n")
f.write(f"Precision,{TP/(TP + FP)}.\n")
f.write(f"Recall:,{TP/(TP + FN)}")

plt.plot(model.history.history['acc'])
plt.xlabel('Epochs')
```

```
plt.ylabel('Accuracy')
plt.savefig('accuracy-set-4.png')
plt.show()

plt.plot(model.history.history['loss'])
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.savefig('loss-set-4.png')
plt.show()
```

Kết quả thu được:

Bảng kết quả chung:

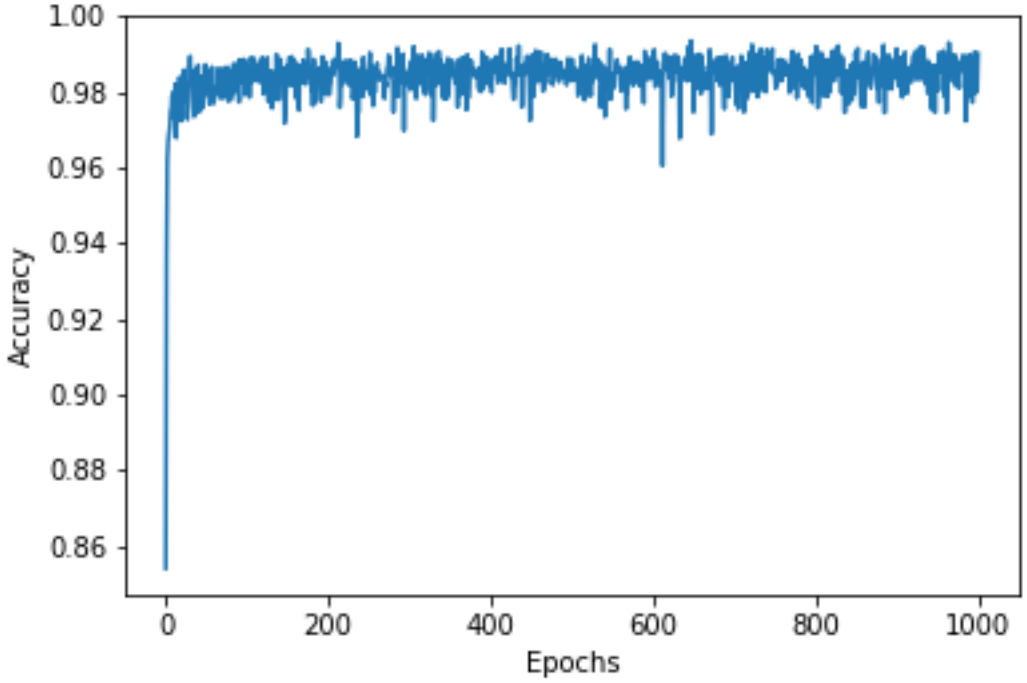
Table 4 Bảng kết của chung

BỘ HƠN 11 NGÀN MẪU	SET 1	**Do chinh xac:**	**0.969005**
		True Positive:	680
		True Negative:	508
		False Positive:	11
		False Negative:	27
		Precision	0.984081041968162.
		Recall:	0.96181
	Set 2	**Do chinh xac:**	**0.970636**
		True Positive:	698
		True Negative:	492
		False Positive:	13
		False Negative:	23
		Precision	0.9817158931082982.
		Recall:	0.9681
	Set 3	**Do chinh xac:**	**0.976354**
		True Positive:	790
		True Negative:	490
		False Positive:	11
		False Negative:	20
		Precision	0.9862671660424469.
		Recall:	0.975309
	Set 4	**Do chinh xac:**	**0.965675**
		True Positive:	760
		True Negative:	506

		False Positive:	12
		False Negative:	33
		Precision	0.9844559585492227.
		Recall:	0.958386
BỘ HƠN 5 NGÀN MẪU	Set 1	**Do chinh xac:**	**0.944984**
		True Positive:	84
		True Negative:	500
		False Positive:	9
		False Negative:	25
		Precision	0.9032258064516129.
		Recall:	0.770642
	Set 2	**Do chinh xac:**	**0.944984**
		True Positive:	84
		True Negative:	500
		False Positive:	9
		False Negative:	25
		Precision	0.9032258064516129.
		Recall:	0.770642
	Set 3	**Do chinh xac:**	**0.949838**
		True Positive:	88
		True Negative:	499
		False Positive:	0
		False Negative:	31
		Precision	1.0.
		Recall:	0.739496
	Set 4	**Do chinh xac:**	**0.959547**
		True Positive:	72
		True Negative:	521
		False Positive:	1
		False Negative:	24
		Precision	0.9863013698630136.
		Recall:	0.75

Kết quả chi tiết

- Bộ hơn 11 ngàn mẫu:
  - Bộ 1:



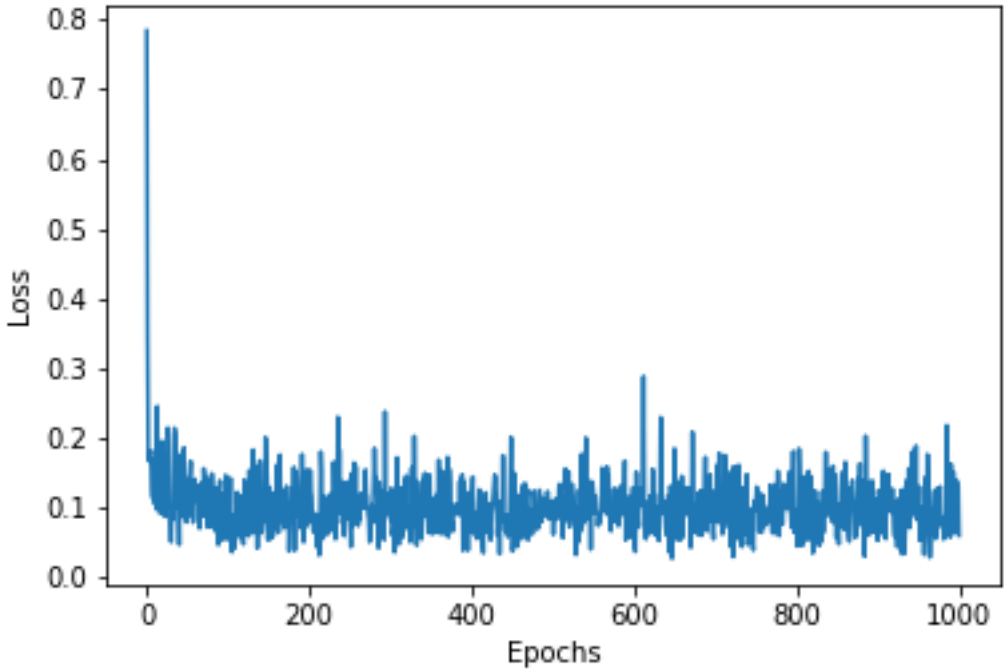


Figure 10 Độ chính xác

Figure 11 Bảng biểu thi độ sai khác

Table 5 Bảng top 100 kết quả training:

epoch	acc	loss	val_acc	val_loss
900	0.983558	0.125336	0.97551	0.127403
901	0.981516	0.115762	0.974694	0.127579
902	0.981618	0.123198	0.976327	0.123152
903	0.989175	0.063199	0.97551	0.12469
904	0.982945	0.113651	0.974694	0.127277
905	0.987949	0.070245	0.974694	0.12709
906	0.978554	0.153055	0.97551	0.133678
907	0.988664	0.067333	0.977143	0.121596
908	0.979575	0.149042	0.977959	0.120132
909	0.990503	0.055655	0.978776	0.119421
910	0.98223	0.120265	0.973878	0.139484
911	0.986009	0.096105	0.977959	0.114575
912	0.979677	0.148029	0.977959	0.122264
913	0.984477	0.101475	0.978776	0.109896
914	0.983558	0.108973	0.978776	0.128625
915	0.985601	0.121185	0.977143	0.122501
916	0.986826	0.093539	0.979592	0.11777
917	0.982537	0.12749	0.979592	0.112719
918	0.985907	0.098471	0.977959	0.125901
919	0.988256	0.073894	0.978776	0.123311
920	0.989788	0.059241	0.977959	0.124969
921	0.984273	0.107805	0.97551	0.149499
922	0.986111	0.08765	0.977143	0.130297
923	0.991013	0.043885	0.977143	0.121315
924	0.983864	0.107941	0.974694	0.135138
925	0.98315	0.113349	0.977959	0.124982
926	0.986111	0.096693	0.973878	0.142677
927	0.989992	0.063044	0.976327	0.13214
928	0.983967	0.125353	0.97551	0.133691
929	0.992034	0.033757	0.97551	0.146605
930	0.983354	0.126171	0.97551	0.12762
931	0.984477	0.101065	0.973878	0.130443
932	0.991728	0.034762	0.977143	0.133767
933	0.988562	0.076527	0.977959	0.134862
934	0.982128	0.139084	0.974694	0.139392
935	0.987949	0.079209	0.977959	0.125394
936	0.98652	0.090241	0.978776	0.119424

937	0.979677	0.156339	0.980408	0.121177
938	0.988154	0.071382	0.977959	0.125648
939	0.987439	0.080661	0.974694	0.138755
940	0.981924	0.119798	0.978776	0.123102
941	0.985396	0.10003	0.976327	0.131448
942	0.975388	0.180947	0.978776	0.130939
943	0.986928	0.101614	0.977143	0.12142
944	0.981209	0.122996	0.978776	0.129795
945	0.987132	0.088659	0.977959	0.128489
946	0.974571	0.188969	0.976327	0.122713
947	0.985396	0.083112	0.977143	0.127467
948	0.983967	0.114766	0.976327	0.13862
949	0.986009	0.092017	0.976327	0.124921
950	0.98029	0.130601	0.978776	0.124821
951	0.9904	0.053259	0.977959	0.126587
952	0.988562	0.066351	0.978776	0.130142
953	0.985396	0.101494	0.977143	0.122243
954	0.985498	0.091429	0.97551	0.125516
955	0.986111	0.087728	0.977143	0.130307
956	0.990911	0.033117	0.977143	0.146903
957	0.989277	0.065554	0.976327	0.133933
958	0.97886	0.143811	0.974694	0.157918
959	0.986315	0.078573	0.978776	0.144197
960	0.974673	0.176971	0.977959	0.121707
961	0.982026	0.111644	0.977143	0.13918
962	0.989175	0.06691	0.97551	0.123697
963	0.993056	0.028984	0.977959	0.142292
964	0.990605	0.055806	0.977143	0.125018
965	0.989686	0.064244	0.978776	0.119792
966	0.987337	0.095337	0.977959	0.128037
967	0.985703	0.094215	0.974694	0.135411
968	0.983047	0.106111	0.97551	0.121232
969	0.982537	0.125175	0.974694	0.125273
970	0.98989	0.055539	0.976327	0.127394
971	0.985703	0.0995	0.977143	0.123157
972	0.981413	0.127945	0.977143	0.122269
973	0.98223	0.116104	0.976327	0.12281
974	0.987439	0.097716	0.977143	0.121529
975	0.983047	0.127223	0.977143	0.116293

976	0.981005	0.130502	0.97551	0.121156
977	0.988154	0.073881	0.97551	0.126448
978	0.987745	0.074071	0.97551	0.123806
979	0.98652	0.084452	0.977143	0.128623
980	0.989583	0.054969	0.977143	0.124421
981	0.989788	0.061012	0.977143	0.117337
982	0.985192	0.088992	0.976327	0.124916
983	0.986213	0.084492	0.978776	0.125196
984	0.972324	0.218433	0.977143	0.128165
985	0.98029	0.129335	0.977143	0.136663
986	0.983864	0.123954	0.974694	0.140947
987	0.987949	0.093085	0.979592	0.125424
988	0.98989	0.057833	0.976327	0.134405
989	0.979065	0.162894	0.976327	0.13062
990	0.986315	0.095319	0.977959	0.124396
991	0.988869	0.064502	0.976327	0.130851
992	0.977226	0.151755	0.977959	0.122427
993	0.988971	0.063055	0.977143	0.12256
994	0.987439	0.097131	0.97551	0.131862
995	0.990605	0.075149	0.978776	0.117424
996	0.979677	0.139391	0.977959	0.122042
997	0.988562	0.064902	0.977143	0.12557
998	0.980699	0.129685	0.977959	0.12141
999	0.990196	0.060477	0.977143	0.126857

- Bộ 2:

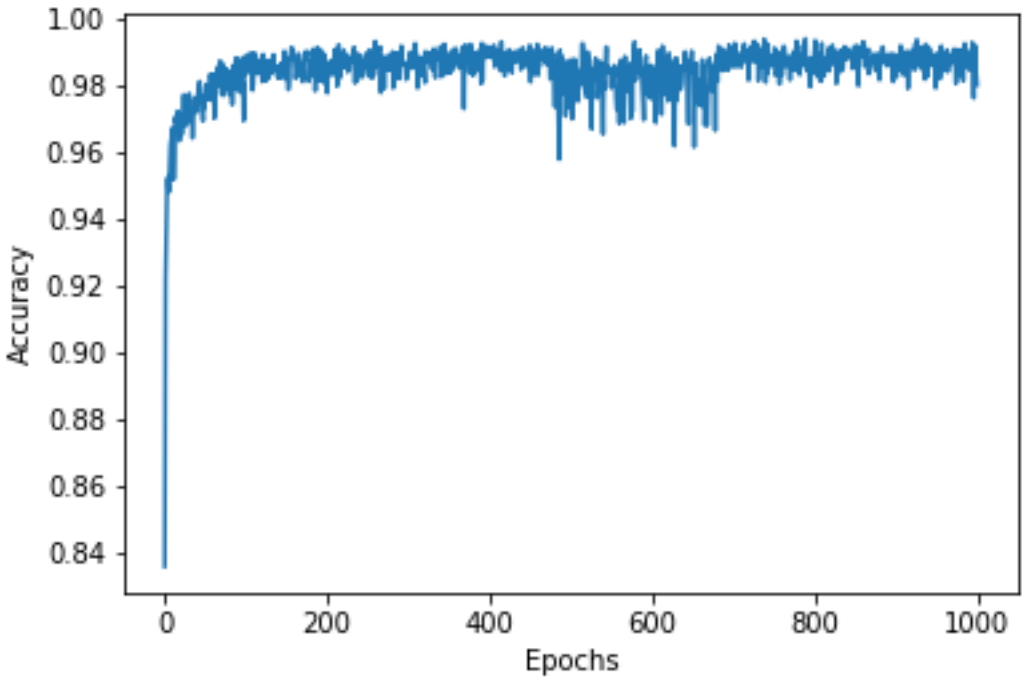


Figure 12 Bảng độ chính xác

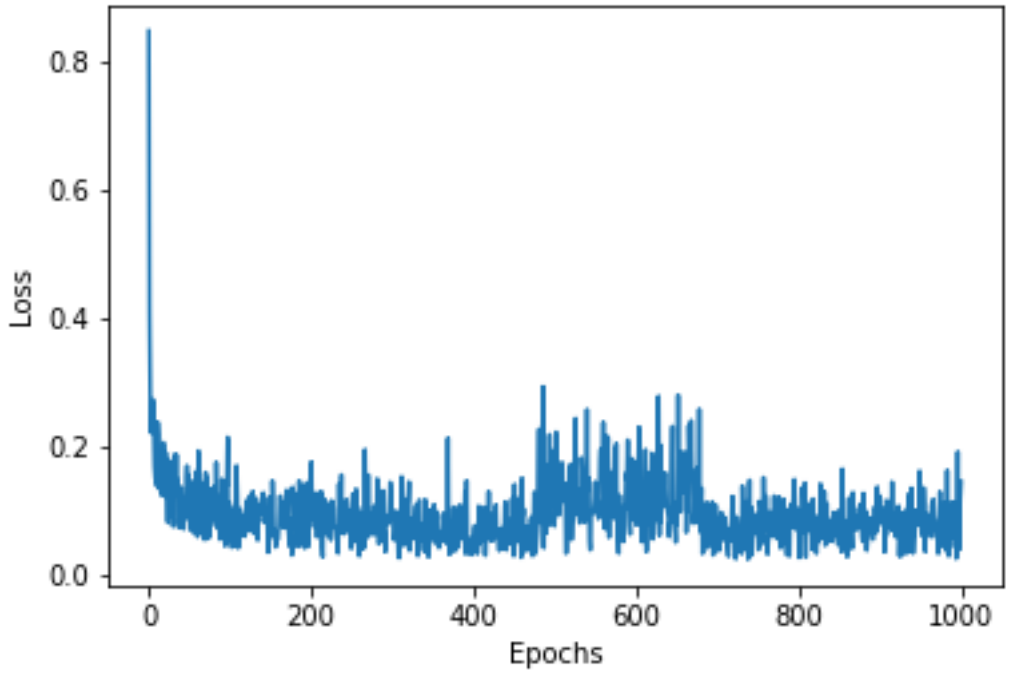


Figure 13 Bảng sai khác

Table 6 Bảng top 100 kết quả traning

epoch	acc	loss	val_acc	val_loss
900	0.982333	0.119133	0.966531	0.160129
901	0.986724	0.090166	0.964898	0.162452
902	0.987949	0.084775	0.964082	0.166885
903	0.985498	0.098432	0.964898	0.160627
904	0.985498	0.104501	0.967347	0.146381
905	0.99183	0.036757	0.967347	0.164353
906	0.984273	0.095422	0.964898	0.160176
907	0.989686	0.062317	0.964898	0.153595
908	0.984477	0.117128	0.964082	0.146058
909	0.982945	0.113242	0.965714	0.150535
910	0.986009	0.095415	0.966531	0.158666
911	0.984886	0.095945	0.965714	0.14417
912	0.990196	0.080276	0.963265	0.155649
913	0.98989	0.071537	0.964082	0.158723
914	0.978962	0.145641	0.966531	0.152114
915	0.988562	0.072529	0.967347	0.161738
916	0.987132	0.083802	0.964082	0.15613
917	0.988052	0.069381	0.965714	0.17773
918	0.985907	0.098292	0.966531	0.151514
919	0.9904	0.060039	0.968163	0.154284
920	0.986724	0.0938	0.964898	0.157379
921	0.986826	0.082311	0.967347	0.154908
922	0.984477	0.108575	0.965714	0.146166
923	0.992239	0.030952	0.968163	0.164586
924	0.989583	0.058017	0.968163	0.152831
925	0.993566	0.030005	0.967347	0.164986
926	0.990298	0.062891	0.969796	0.149425
927	0.985703	0.097964	0.965714	0.172786
928	0.988052	0.060661	0.969796	0.162336
929	0.985498	0.09905	0.967347	0.156617
930	0.987745	0.059015	0.970612	0.158845
931	0.991422	0.033795	0.964898	0.166317
932	0.987337	0.073678	0.968163	0.148561
933	0.984171	0.105411	0.959184	0.179261
934	0.986826	0.090572	0.967347	0.146417
935	0.98172	0.127042	0.962449	0.15674
936	0.992239	0.037545	0.968163	0.157039

937	0.991115	0.04689	0.967347	0.163257
938	0.982639	0.136243	0.96	0.163666
939	0.987643	0.077335	0.967347	0.167116
940	0.991115	0.034833	0.966531	0.170524
941	0.989992	0.055978	0.966531	0.174544
942	0.98846	0.067962	0.965714	0.152869
943	0.986009	0.101813	0.970612	0.149686
944	0.986622	0.084982	0.96898	0.137532
945	0.985601	0.094116	0.96898	0.162573
946	0.988358	0.087469	0.969796	0.150308
947	0.989175	0.078806	0.96898	0.150391
948	0.979065	0.16291	0.964082	0.165316
949	0.985703	0.093731	0.964898	0.160571
950	0.989481	0.071158	0.967347	0.15334
951	0.989073	0.062209	0.964082	0.166156
952	0.986622	0.074557	0.965714	0.169969
953	0.981822	0.13676	0.963265	0.151329
954	0.982333	0.11053	0.967347	0.15621
955	0.991728	0.037283	0.964898	0.164196
956	0.985805	0.087187	0.967347	0.158128
957	0.984375	0.096213	0.964082	0.166047
958	0.984273	0.095312	0.968163	0.155204
959	0.986622	0.076036	0.969796	0.159402
960	0.987745	0.075365	0.969796	0.160248
961	0.988664	0.092139	0.962449	0.161667
962	0.990503	0.060115	0.964082	0.151949
963	0.983967	0.110857	0.966531	0.147182
964	0.986826	0.069289	0.964082	0.164327
965	0.986213	0.118981	0.967347	0.156148
966	0.986009	0.091392	0.970612	0.146691
967	0.986622	0.086678	0.966531	0.147382
968	0.989481	0.068651	0.964898	0.150414
969	0.989175	0.058472	0.968163	0.146537
970	0.987235	0.074151	0.965714	0.144784
971	0.990809	0.034031	0.964082	0.158062
972	0.986724	0.073689	0.960816	0.14963
973	0.980494	0.130432	0.964898	0.146698
974	0.990605	0.067584	0.964898	0.14312
975	0.989992	0.045804	0.964082	0.153132

976	0.989788	0.062031	0.964898	0.166191
977	0.989992	0.058213	0.965714	0.154581
978	0.982945	0.112313	0.960816	0.16598
979	0.987643	0.076888	0.963265	0.148524
980	0.989788	0.057155	0.965714	0.154244
981	0.990707	0.03829	0.964898	0.1588
982	0.981209	0.16357	0.965714	0.147378
983	0.989686	0.03782	0.967347	0.161101
984	0.992443	0.030268	0.966531	0.168137
985	0.98846	0.064197	0.964898	0.153455
986	0.990605	0.044208	0.964898	0.163781
987	0.986724	0.076537	0.964082	0.159301
988	0.985294	0.114739	0.964082	0.161513
989	0.984681	0.104512	0.964082	0.174858
990	0.989277	0.075003	0.96898	0.141691
991	0.988766	0.059467	0.968163	0.153896
992	0.984477	0.113916	0.968163	0.152281
993	0.992749	0.026556	0.967347	0.159603
994	0.988154	0.078269	0.964082	0.144812
995	0.976103	0.192485	0.964082	0.145931
996	0.991728	0.057106	0.965714	0.151558
997	0.991728	0.040711	0.966531	0.154219
998	0.983456	0.119552	0.966531	0.148277
999	0.980086	0.146161	0.963265	0.148144

- Bộ 3:

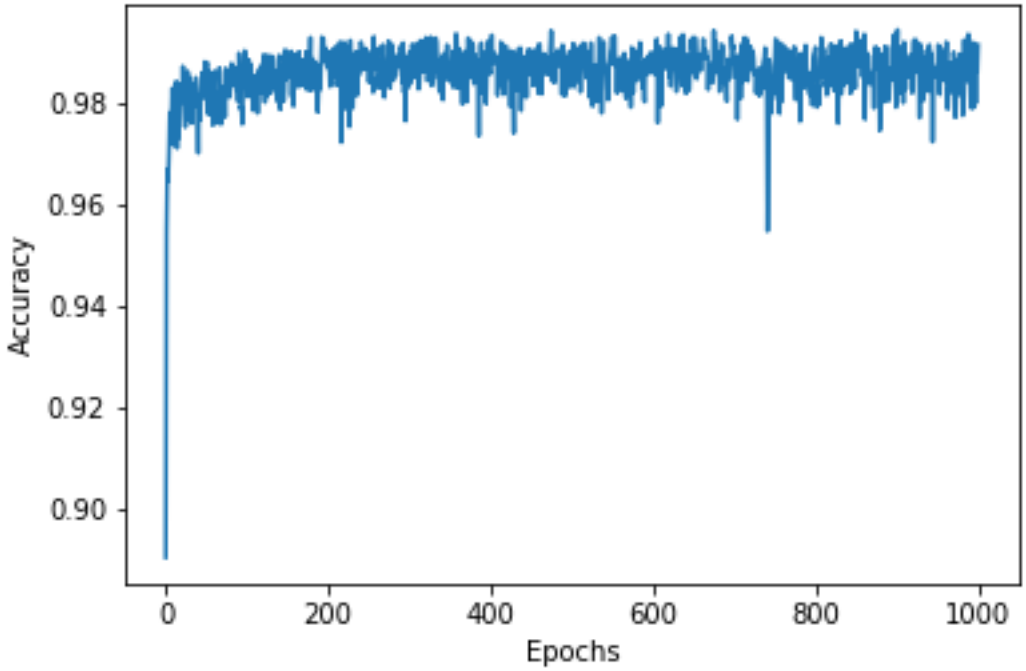


Figure 14 Bảng độ chính xác

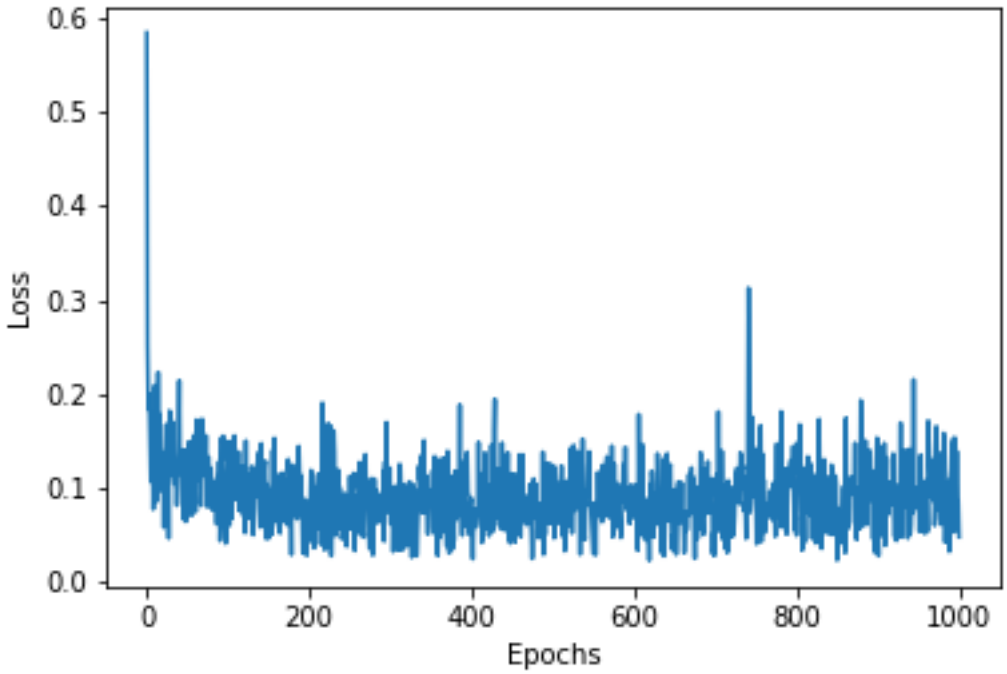


Figure 15 Bảng sai khác

Table 7 Bảng top 100 kết quả training

epoch	acc	loss	val_acc	val_loss
900	0.994171	0.027076	0.979405	0.199397
901	0.983849	0.119483	0.977117	0.199134
902	0.989679	0.049178	0.979405	0.188725
903	0.988054	0.091518	0.979405	0.173703
904	0.981747	0.142725	0.974828	0.177032
905	0.990826	0.048387	0.980931	0.174421
906	0.987768	0.065287	0.978642	0.179382
907	0.979836	0.147159	0.977117	0.184086
908	0.991781	0.037989	0.977879	0.208466
909	0.987672	0.098104	0.977117	0.182111
910	0.982511	0.131965	0.976354	0.163502
911	0.98815	0.072542	0.977117	0.184956
912	0.988054	0.071384	0.980168	0.181044
913	0.985761	0.101241	0.978642	0.170106
914	0.984901	0.10578	0.977879	0.17394
915	0.98729	0.092034	0.977117	0.171774
916	0.989297	0.083002	0.977879	0.158938
917	0.987959	0.082196	0.979405	0.164574
918	0.981747	0.136207	0.977117	0.165999
919	0.989966	0.065259	0.977879	0.180124
920	0.992068	0.047129	0.977879	0.176951
921	0.98901	0.067215	0.975591	0.167218
922	0.992259	0.043353	0.977879	0.183508
923	0.98643	0.101815	0.977117	0.171551
924	0.991781	0.04809	0.973303	0.18231
925	0.985761	0.091591	0.977117	0.182335
926	0.986334	0.093139	0.976354	0.174997
927	0.976873	0.16889	0.978642	0.171553
928	0.991304	0.046864	0.973303	0.180391
929	0.98643	0.095067	0.977117	0.166723
930	0.984232	0.115927	0.977117	0.174121
931	0.98815	0.081733	0.977879	0.178413
932	0.980314	0.139705	0.977117	0.169846
933	0.988819	0.078471	0.977879	0.162735
934	0.982989	0.110582	0.979405	0.167824
935	0.986334	0.09164	0.978642	0.181662
936	0.990157	0.04576	0.980168	0.187561

937	0.98643	0.108911	0.977879	0.192528
938	0.982034	0.141002	0.978642	0.156248
939	0.991208	0.050639	0.977117	0.178462
940	0.989774	0.060935	0.978642	0.17914
941	0.985283	0.101357	0.977117	0.166895
942	0.989488	0.07249	0.977879	0.164976
943	0.97219	0.215132	0.976354	0.163111
944	0.987003	0.088531	0.977117	0.163458
945	0.986812	0.09906	0.977117	0.174075
946	0.986334	0.083688	0.979405	0.156998
947	0.986334	0.077578	0.977879	0.164563
948	0.984805	0.099241	0.980168	0.187529
949	0.986716	0.0912	0.977879	0.149605
950	0.991208	0.053422	0.976354	0.185811
951	0.987194	0.080694	0.977879	0.180683
952	0.98146	0.134737	0.977879	0.17925
953	0.990635	0.053292	0.978642	0.168599
954	0.988628	0.068989	0.978642	0.163597
955	0.987003	0.073015	0.978642	0.175805
956	0.987194	0.097207	0.978642	0.17427
957	0.98987	0.050753	0.979405	0.191498
958	0.984136	0.107349	0.975591	0.168584
959	0.986239	0.095223	0.977879	0.184351
960	0.991781	0.052799	0.977117	0.193519
961	0.978689	0.171005	0.974828	0.216537
962	0.978593	0.147177	0.976354	0.172373
963	0.982894	0.118226	0.977117	0.17085
964	0.983945	0.101237	0.978642	0.168796
965	0.987672	0.089668	0.978642	0.166574
966	0.985092	0.093237	0.980168	0.180716
967	0.986143	0.085979	0.978642	0.165937
968	0.981747	0.130512	0.973303	0.1797
969	0.989774	0.05937	0.975591	0.194725
970	0.987768	0.078892	0.978642	0.170088
971	0.976969	0.165091	0.977117	0.164172
972	0.984136	0.108582	0.976354	0.173035
973	0.986143	0.102611	0.977879	0.170384
974	0.988437	0.075794	0.977117	0.179575
975	0.981365	0.13658	0.979405	0.175073

976	0.987576	0.066179	0.977879	0.198908
977	0.989392	0.060417	0.976354	0.177996
978	0.988914	0.068686	0.979405	0.176309
979	0.981938	0.120731	0.979405	0.16807
980	0.977351	0.157801	0.979405	0.17316
981	0.987576	0.074654	0.980168	0.174149
982	0.992164	0.0414	0.981693	0.185323
983	0.989392	0.067059	0.975591	0.190768
984	0.985856	0.08427	0.979405	0.168283
985	0.990252	0.05169	0.976354	0.20028
986	0.984041	0.107089	0.979405	0.177352
987	0.99331	0.031607	0.979405	0.201222
988	0.985665	0.110755	0.978642	0.174403
989	0.98318	0.119095	0.979405	0.176243
990	0.97888	0.146884	0.978642	0.166768
991	0.990348	0.064219	0.978642	0.180706
992	0.978784	0.152253	0.977879	0.174278
993	0.991686	0.052433	0.977117	0.189507
994	0.978976	0.152916	0.976354	0.168876
995	0.988914	0.075576	0.976354	0.182975
996	0.97974	0.138664	0.977117	0.169289
997	0.988341	0.083266	0.977879	0.169177
998	0.985856	0.085477	0.974828	0.177681
999	0.991208	0.047257	0.977879	0.193957

- Bộ 4:

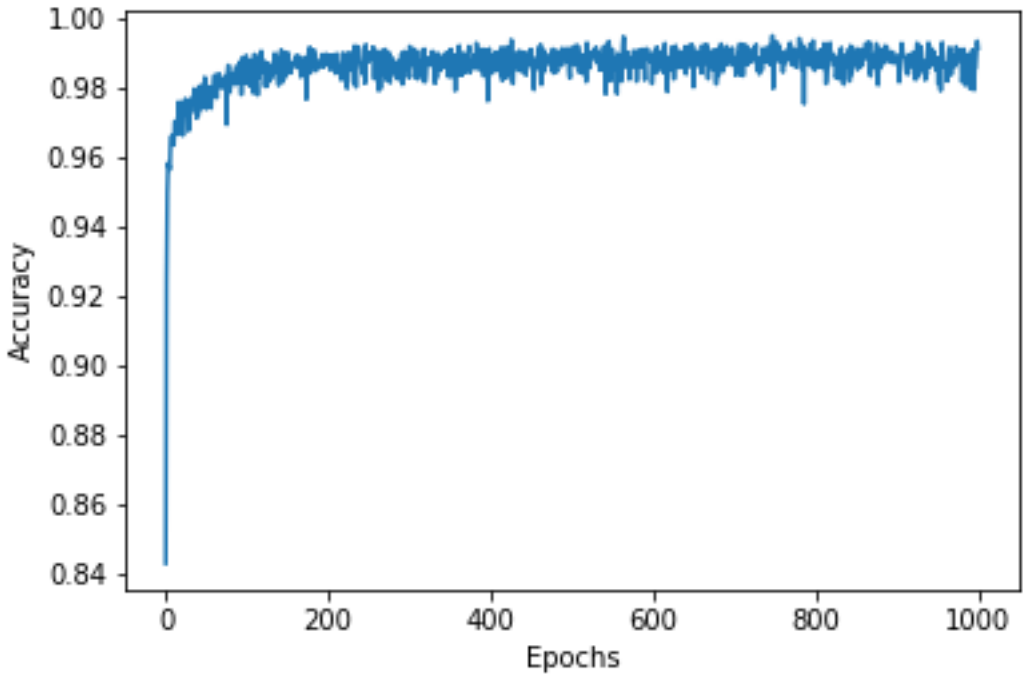


Figure 16 Bảng độ chính xác

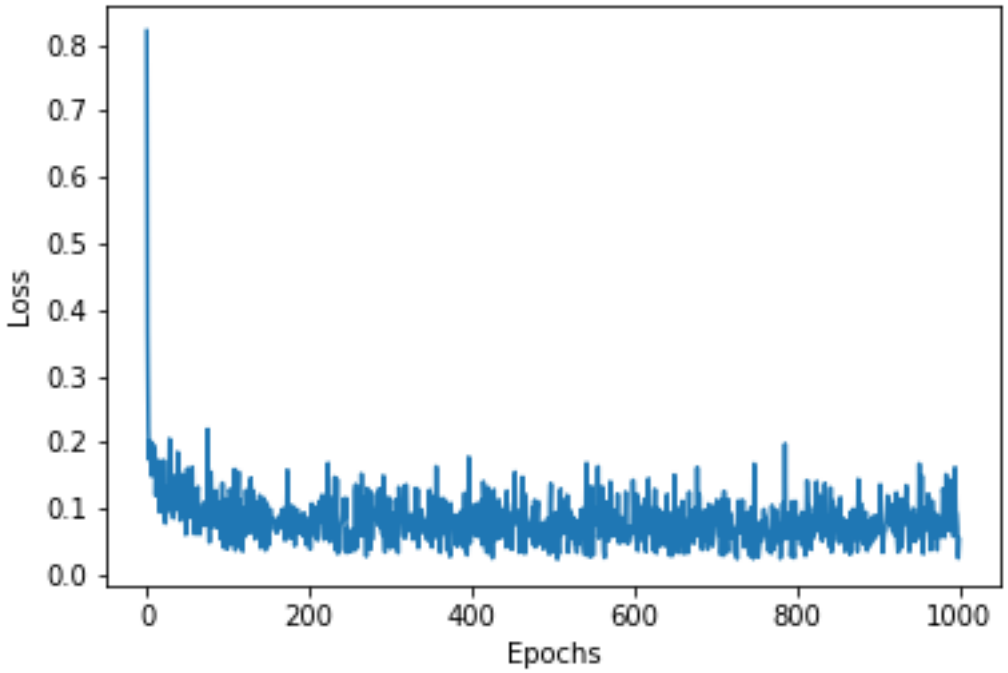


Figure 17 Bảng sai khác

Table 8 Bảng top 100 kết quả training

epoch	acc	loss	val_acc	val_loss
900	0.987959	0.065548	0.969489	0.225728
901	0.98901	0.08044	0.968726	0.204195
902	0.981365	0.136383	0.968726	0.196496
903	0.986716	0.08322	0.966438	0.210534
904	0.989106	0.063233	0.970252	0.214698
905	0.992546	0.033967	0.968726	0.2158
906	0.991399	0.049546	0.968726	0.212563
907	0.98901	0.063489	0.969489	0.230202
908	0.98901	0.066472	0.970252	0.289912
909	0.988628	0.080532	0.969489	0.204724
910	0.985092	0.09949	0.968726	0.227638
911	0.983849	0.119657	0.966438	0.196911
912	0.98815	0.08032	0.968726	0.193157
913	0.983658	0.110523	0.967201	0.211107
914	0.986047	0.094131	0.969489	0.188392
915	0.989297	0.074069	0.969489	0.21625
916	0.98643	0.096465	0.966438	0.187077
917	0.989392	0.066522	0.969489	0.199047
918	0.985761	0.105929	0.969489	0.2462
919	0.985952	0.09469	0.970252	0.200351
920	0.989966	0.059863	0.967201	0.215705
921	0.983945	0.110421	0.969489	0.196137
922	0.982034	0.124295	0.967963	0.19271
923	0.989966	0.051469	0.970252	0.240406
924	0.992641	0.034311	0.970252	0.230916
925	0.982798	0.113156	0.969489	0.218365
926	0.989488	0.057611	0.968726	0.259344
927	0.984041	0.105473	0.965675	0.203212
928	0.984996	0.114871	0.970252	0.212649
929	0.99159	0.041769	0.969489	0.219613
930	0.987481	0.091331	0.969489	0.196932
931	0.985092	0.105444	0.969489	0.184209
932	0.987099	0.080034	0.970252	0.199091
933	0.984614	0.103877	0.968726	0.199271
934	0.981938	0.134909	0.967201	0.283917
935	0.987385	0.076019	0.969489	0.223401
936	0.983563	0.105027	0.967963	0.202326

937	0.990539	0.049039	0.970252	0.228366
938	0.992737	0.03521	0.970252	0.24609
939	0.986812	0.091959	0.967963	0.232538
940	0.985665	0.08114	0.970252	0.219612
941	0.986716	0.090826	0.967963	0.202342
942	0.991017	0.051988	0.969489	0.22386
943	0.986525	0.100273	0.968726	0.198696
944	0.989774	0.053351	0.967963	0.217376
945	0.990157	0.050145	0.967201	0.209579
946	0.987863	0.064095	0.971015	0.215426
947	0.988245	0.086216	0.967963	0.236271
948	0.984805	0.090065	0.969489	0.2235
949	0.988437	0.056109	0.967963	0.220567
950	0.980791	0.16781	0.969489	0.202945
951	0.986239	0.081733	0.970252	0.223889
952	0.987672	0.085131	0.967963	0.22462
953	0.97888	0.149832	0.967963	0.211757
954	0.98901	0.058791	0.964912	0.206866
955	0.992833	0.031781	0.967963	0.23883
956	0.985665	0.100815	0.967963	0.219425
957	0.986812	0.067265	0.968726	0.216736
958	0.986239	0.087015	0.967963	0.207494
959	0.98557	0.094599	0.967963	0.243337
960	0.985283	0.10359	0.967201	0.204118
961	0.981651	0.122058	0.969489	0.212287
962	0.987003	0.071708	0.967963	0.201695
963	0.990252	0.058914	0.967963	0.209902
964	0.991399	0.037542	0.971777	0.246431
965	0.991017	0.045531	0.967963	0.214056
966	0.983467	0.111997	0.968726	0.206966
967	0.986716	0.09104	0.967963	0.196902
968	0.989106	0.05696	0.968726	0.224186
969	0.987099	0.082003	0.968726	0.243009
970	0.988914	0.067912	0.967963	0.225529
971	0.985952	0.092024	0.967963	0.230281
972	0.99159	0.037579	0.967201	0.248746
973	0.988628	0.075163	0.967201	0.222658
974	0.99073	0.056246	0.968726	0.247535
975	0.988054	0.082305	0.968726	0.217288

976	0.987576	0.087422	0.967201	0.200584
977	0.988628	0.061281	0.968726	0.230308
978	0.98901	0.059843	0.967201	0.250666
979	0.98146	0.12985	0.966438	0.223728
980	0.983372	0.120116	0.967963	0.193737
981	0.990921	0.056651	0.970252	0.226453
982	0.985092	0.083964	0.971015	0.210104
983	0.980505	0.151443	0.970252	0.214819
984	0.989488	0.066809	0.969489	0.223322
985	0.985474	0.099176	0.967963	0.204205
986	0.988819	0.069117	0.969489	0.278238
987	0.988819	0.071464	0.965675	0.211632
988	0.986812	0.068406	0.958047	0.239984
989	0.979549	0.14322	0.967963	0.198317
990	0.986716	0.079096	0.971015	0.219031
991	0.988914	0.076146	0.970252	0.208328
992	0.990635	0.058187	0.968726	0.210192
993	0.980505	0.144572	0.971777	0.217703
994	0.979071	0.162041	0.965675	0.196405
995	0.985378	0.094937	0.968726	0.201925
996	0.985474	0.092977	0.968726	0.196066
997	0.98901	0.07196	0.967963	0.227966
998	0.993215	0.025305	0.968726	0.24087
999	0.991112	0.05405	0.967963	0.225884

- Bộ hơn 5 ngàn mẫu:
    - Bộ 1

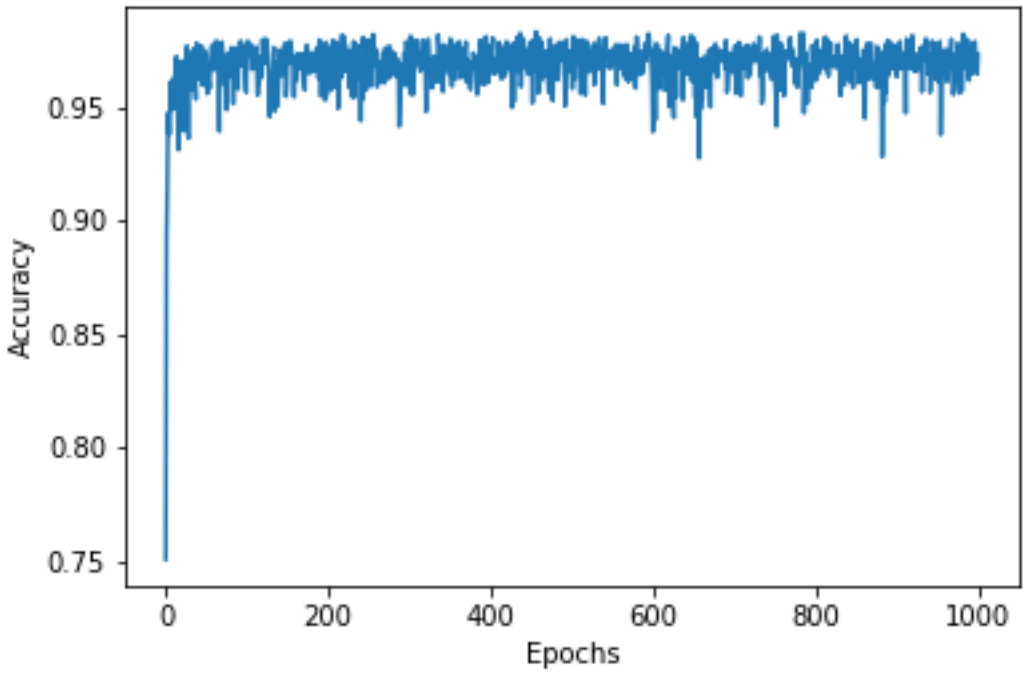


Figure 18 Bảng độ chính xác

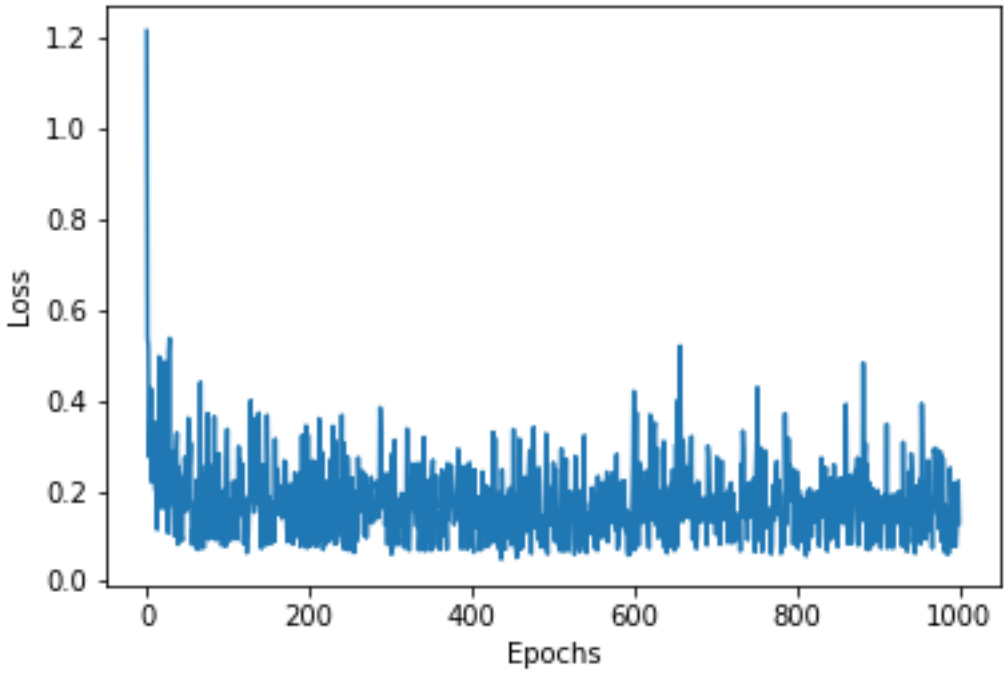


Figure 19 Bảng sai khác

Table 9 Bảng top 100 kết quả training

epoch	acc	loss	val_acc	val_loss
900	0.976664	0.134139	0.952998	0.207178
901	0.969156	0.152464	0.952998	0.214488
902	0.966924	0.182715	0.951378	0.219797
903	0.979302	0.073639	0.951378	0.225204
904	0.972403	0.126454	0.951378	0.236664
905	0.972606	0.134194	0.951378	0.227573
906	0.969562	0.18949	0.951378	0.227724
907	0.974229	0.129881	0.954619	0.226037
908	0.967532	0.191958	0.951378	0.219716
909	0.969968	0.138077	0.952998	0.224881
910	0.947646	0.347158	0.952998	0.21468
911	0.98052	0.073368	0.948136	0.218789
912	0.971185	0.164184	0.951378	0.217119
913	0.9722	0.120157	0.951378	0.223831
914	0.972606	0.134544	0.949757	0.226468
915	0.976461	0.135893	0.951378	0.215962
916	0.967127	0.19138	0.946515	0.213848
917	0.977882	0.074808	0.946515	0.220313
918	0.98194	0.073614	0.951378	0.225414
919	0.971185	0.177477	0.949757	0.208593
920	0.968141	0.1899	0.954619	0.210642
921	0.965909	0.192183	0.948136	0.235857
922	0.968547	0.179503	0.952998	0.21499
923	0.973214	0.129537	0.949757	0.257674
924	0.973214	0.156283	0.954619	0.210426
925	0.966518	0.199037	0.954619	0.205665
926	0.973823	0.125696	0.954619	0.213085
927	0.969968	0.159146	0.951378	0.209757
928	0.973417	0.143438	0.949757	0.218677
929	0.976867	0.078089	0.951378	0.219695
930	0.956981	0.307439	0.951378	0.231339
931	0.976664	0.145217	0.948136	0.215162
932	0.97362	0.120886	0.951378	0.205
933	0.970982	0.18103	0.951378	0.203182
934	0.971997	0.134341	0.951378	0.214907
935	0.98052	0.066282	0.948136	0.2262
936	0.977882	0.0799	0.951378	0.221062

937	0.965097	0.241818	0.951378	0.20728
938	0.973011	0.131939	0.954619	0.211595
939	0.9722	0.124228	0.954619	0.209826
940	0.956778	0.2822	0.952998	0.221003
941	0.963068	0.242822	0.951378	0.228655
942	0.979505	0.089454	0.949757	0.223624
943	0.978084	0.083726	0.951378	0.223578
944	0.975041	0.120821	0.951378	0.252454
945	0.97849	0.061825	0.949757	0.274254
946	0.972403	0.130654	0.951378	0.216324
947	0.962459	0.211165	0.949757	0.226044
948	0.974229	0.120569	0.952998	0.214223
949	0.966112	0.189644	0.951378	0.212513
950	0.964286	0.196725	0.95624	0.207191
951	0.961445	0.225822	0.951378	0.212743
952	0.979302	0.082702	0.951378	0.220456
953	0.938109	0.392325	0.951378	0.277073
954	0.959821	0.268887	0.949757	0.230108
955	0.966721	0.172368	0.952998	0.220144
956	0.966112	0.181838	0.95624	0.20049
957	0.965503	0.216292	0.951378	0.211992
958	0.975852	0.092794	0.952998	0.216465
959	0.971388	0.135974	0.952998	0.2306
960	0.967532	0.181855	0.952998	0.217363
961	0.961445	0.26807	0.952998	0.220875
962	0.975446	0.137813	0.949757	0.222603
963	0.973011	0.111654	0.951378	0.219979
964	0.967127	0.184666	0.951378	0.221464
965	0.973011	0.144311	0.949757	0.21025
966	0.980317	0.073554	0.949757	0.223511
967	0.968141	0.187571	0.952998	0.211679
968	0.956372	0.292039	0.951378	0.245513
969	0.955763	0.285218	0.95624	0.221997
970	0.95901	0.294555	0.952998	0.22282
971	0.9653	0.205222	0.951378	0.21361
972	0.974026	0.1505	0.952998	0.225891
973	0.971185	0.147408	0.948136	0.220034
974	0.97849	0.116182	0.949757	0.219954
975	0.97017	0.175776	0.954619	0.21441

976	0.956372	0.286595	0.949757	0.215673
977	0.975649	0.089749	0.949757	0.210085
978	0.974229	0.13132	0.952998	0.211509
979	0.957386	0.272095	0.951378	0.21578
980	0.967127	0.185321	0.954619	0.220746
981	0.982549	0.066298	0.952998	0.224374
982	0.967938	0.170392	0.954619	0.209414
983	0.979708	0.074292	0.951378	0.226607
984	0.980317	0.060976	0.952998	0.237001
985	0.975649	0.123658	0.954619	0.22007
986	0.980114	0.061937	0.952998	0.229734
987	0.974229	0.100724	0.952998	0.226104
988	0.962054	0.250795	0.954619	0.210552
989	0.969156	0.183717	0.954619	0.219245
990	0.968547	0.158522	0.952998	0.27374
991	0.977882	0.077825	0.954619	0.226069
992	0.965097	0.216796	0.952998	0.211471
993	0.976258	0.114775	0.949757	0.224069
994	0.975041	0.095426	0.95624	0.230412
995	0.979302	0.078108	0.95624	0.234549
996	0.974635	0.112218	0.952998	0.235121
997	0.969968	0.173752	0.952998	0.21797
998	0.964895	0.221769	0.95624	0.220417
999	0.973417	0.128512	0.954619	0.220526

- Bộ 2

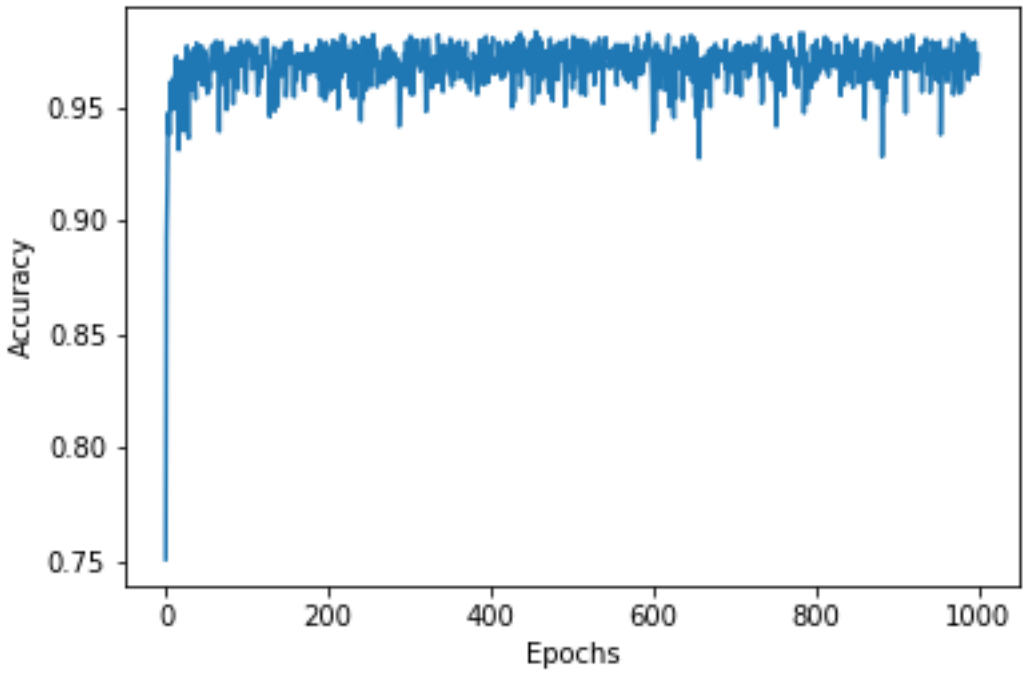


Figure 20 Bảng độ chính xác

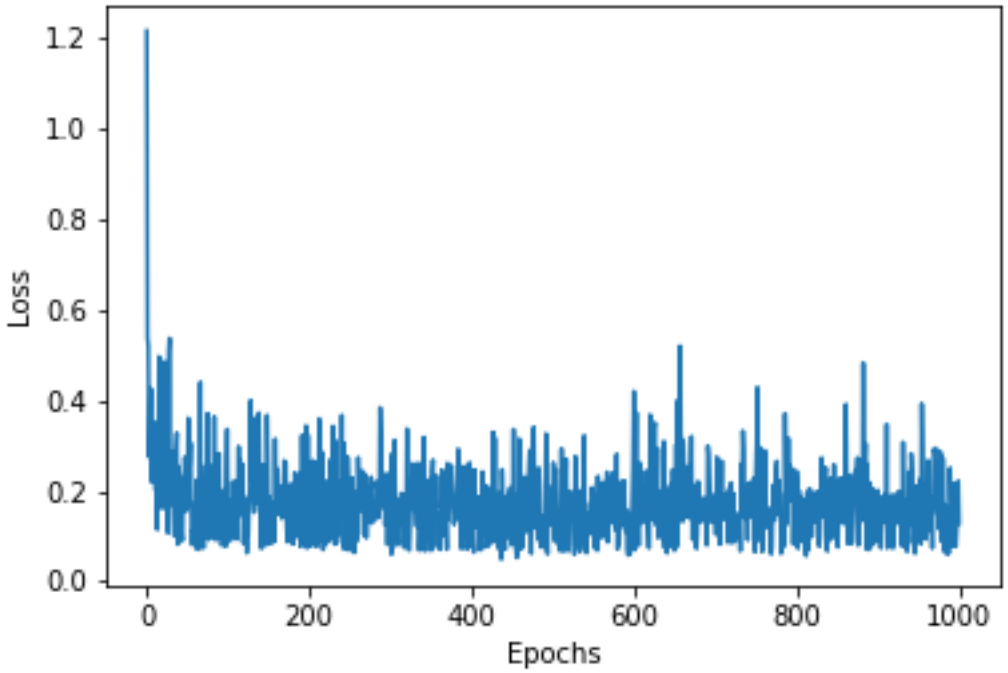


Figure 21 Bảng sai khác

Table 10 Bảng top 100 kết quả training

epoch	Acc	loss	val_acc	val_loss
900	0.976664	0.134139	0.952998	0.207178
901	0.969156	0.152464	0.952998	0.214488
902	0.966924	0.182715	0.951378	0.219797
903	0.979302	0.073639	0.951378	0.225204
904	0.972403	0.126454	0.951378	0.236664
905	0.972606	0.134194	0.951378	0.227573
906	0.969562	0.18949	0.951378	0.227724
907	0.974229	0.129881	0.954619	0.226037
908	0.967532	0.191958	0.951378	0.219716
909	0.969968	0.138077	0.952998	0.224881
910	0.947646	0.347158	0.952998	0.21468
911	0.98052	0.073368	0.948136	0.218789
912	0.971185	0.164184	0.951378	0.217119
913	0.9722	0.120157	0.951378	0.223831
914	0.972606	0.134544	0.949757	0.226468
915	0.976461	0.135893	0.951378	0.215962
916	0.967127	0.19138	0.946515	0.213848
917	0.977882	0.074808	0.946515	0.220313
918	0.98194	0.073614	0.951378	0.225414
919	0.971185	0.177477	0.949757	0.208593
920	0.968141	0.1899	0.954619	0.210642
921	0.965909	0.192183	0.948136	0.235857
922	0.968547	0.179503	0.952998	0.21499
923	0.973214	0.129537	0.949757	0.257674
924	0.973214	0.156283	0.954619	0.210426
925	0.966518	0.199037	0.954619	0.205665
926	0.973823	0.125696	0.954619	0.213085
927	0.969968	0.159146	0.951378	0.209757
928	0.973417	0.143438	0.949757	0.218677
929	0.976867	0.078089	0.951378	0.219695
930	0.956981	0.307439	0.951378	0.231339
931	0.976664	0.145217	0.948136	0.215162
932	0.97362	0.120886	0.951378	0.205
933	0.970982	0.18103	0.951378	0.203182
934	0.971997	0.134341	0.951378	0.214907
935	0.98052	0.066282	0.948136	0.2262
936	0.977882	0.0799	0.951378	0.221062

937	0.965097	0.241818	0.951378	0.20728
938	0.973011	0.131939	0.954619	0.211595
939	0.9722	0.124228	0.954619	0.209826
940	0.956778	0.2822	0.952998	0.221003
941	0.963068	0.242822	0.951378	0.228655
942	0.979505	0.089454	0.949757	0.223624
943	0.978084	0.083726	0.951378	0.223578
944	0.975041	0.120821	0.951378	0.252454
945	0.97849	0.061825	0.949757	0.274254
946	0.972403	0.130654	0.951378	0.216324
947	0.962459	0.211165	0.949757	0.226044
948	0.974229	0.120569	0.952998	0.214223
949	0.966112	0.189644	0.951378	0.212513
950	0.964286	0.196725	0.95624	0.207191
951	0.961445	0.225822	0.951378	0.212743
952	0.979302	0.082702	0.951378	0.220456
953	0.938109	0.392325	0.951378	0.277073
954	0.959821	0.268887	0.949757	0.230108
955	0.966721	0.172368	0.952998	0.220144
956	0.966112	0.181838	0.95624	0.20049
957	0.965503	0.216292	0.951378	0.211992
958	0.975852	0.092794	0.952998	0.216465
959	0.971388	0.135974	0.952998	0.2306
960	0.967532	0.181855	0.952998	0.217363
961	0.961445	0.26807	0.952998	0.220875
962	0.975446	0.137813	0.949757	0.222603
963	0.973011	0.111654	0.951378	0.219979
964	0.967127	0.184666	0.951378	0.221464
965	0.973011	0.144311	0.949757	0.21025
966	0.980317	0.073554	0.949757	0.223511
967	0.968141	0.187571	0.952998	0.211679
968	0.956372	0.292039	0.951378	0.245513
969	0.955763	0.285218	0.95624	0.221997
970	0.95901	0.294555	0.952998	0.22282
971	0.9653	0.205222	0.951378	0.21361
972	0.974026	0.1505	0.952998	0.225891
973	0.971185	0.147408	0.948136	0.220034
974	0.97849	0.116182	0.949757	0.219954
975	0.97017	0.175776	0.954619	0.21441

976	0.956372	0.286595	0.949757	0.215673
977	0.975649	0.089749	0.949757	0.210085
978	0.974229	0.13132	0.952998	0.211509
979	0.957386	0.272095	0.951378	0.21578
980	0.967127	0.185321	0.954619	0.220746
981	0.982549	0.066298	0.952998	0.224374
982	0.967938	0.170392	0.954619	0.209414
983	0.979708	0.074292	0.951378	0.226607
984	0.980317	0.060976	0.952998	0.237001
985	0.975649	0.123658	0.954619	0.22007
986	0.980114	0.061937	0.952998	0.229734
987	0.974229	0.100724	0.952998	0.226104
988	0.962054	0.250795	0.954619	0.210552
989	0.969156	0.183717	0.954619	0.219245
990	0.968547	0.158522	0.952998	0.27374
991	0.977882	0.077825	0.954619	0.226069
992	0.965097	0.216796	0.952998	0.211471
993	0.976258	0.114775	0.949757	0.224069
994	0.975041	0.095426	0.95624	0.230412
995	0.979302	0.078108	0.95624	0.234549
996	0.974635	0.112218	0.952998	0.235121
997	0.969968	0.173752	0.952998	0.21797
998	0.964895	0.221769	0.95624	0.220417
999	0.973417	0.128512	0.954619	0.220526

- Bộ 3

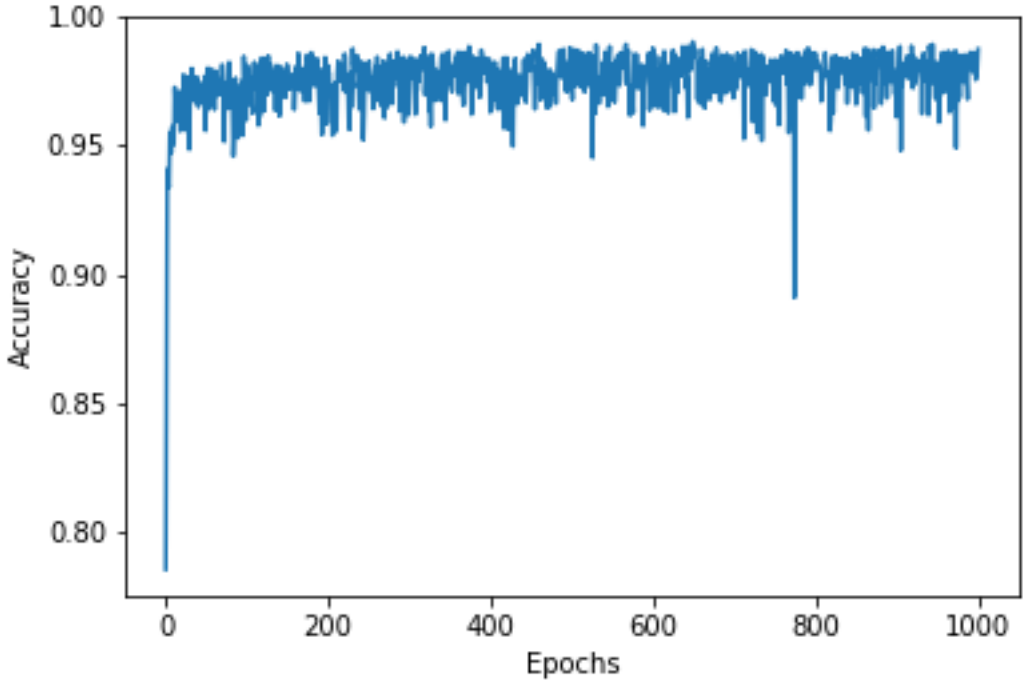


Figure 22 Bảng độ chính xác

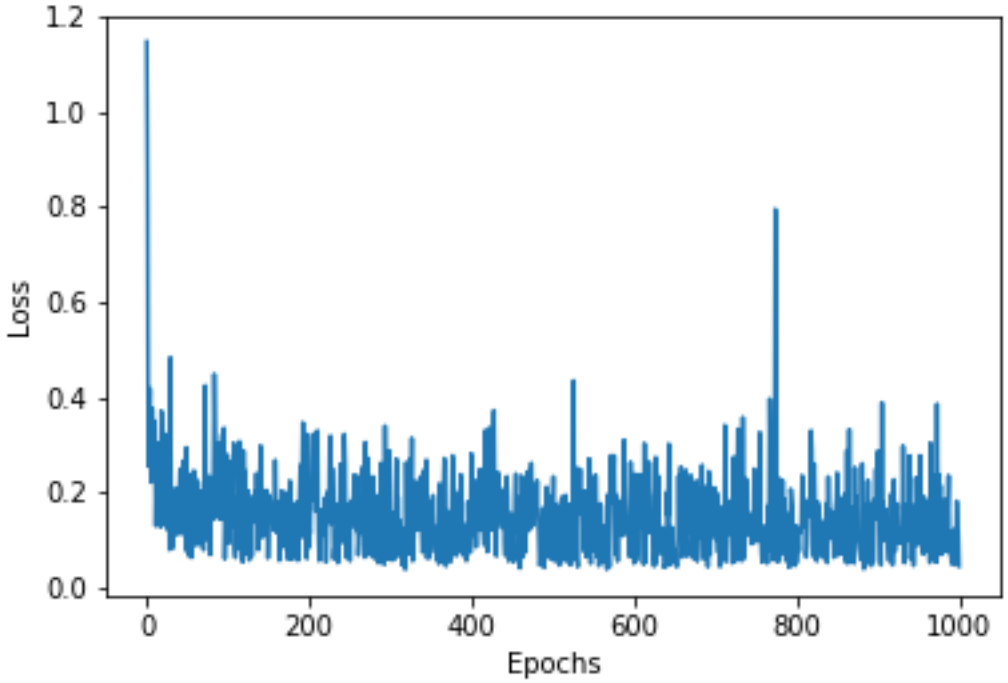


Figure 23 Bảng sai khác

Table 11 Bảng top 100 kết quả training

epoch	acc	loss	val_acc	val_loss
899	0.961039	0.286762	0.954619	0.208321
900	0.978693	0.146204	0.95624	0.184457
901	0.971794	0.19286	0.954619	0.197035
902	0.977273	0.147448	0.954619	0.187049
903	0.988433	0.047505	0.951378	0.192706
904	0.948052	0.388901	0.957861	0.198145
905	0.982143	0.097865	0.95624	0.176947
906	0.973214	0.177838	0.954619	0.183326
907	0.979099	0.118224	0.954619	0.177218
908	0.98052	0.112249	0.954619	0.174096
909	0.976461	0.165284	0.957861	0.173068
910	0.982346	0.109463	0.95624	0.174759
911	0.978896	0.142549	0.95624	0.174455
912	0.984781	0.081609	0.95624	0.165741
913	0.984781	0.056555	0.959481	0.170409
914	0.976664	0.134286	0.952998	0.180783
915	0.984375	0.07346	0.954619	0.177302
916	0.975649	0.197346	0.957861	0.195799
917	0.988636	0.0476	0.954619	0.17429
918	0.982549	0.106203	0.954619	0.177539
919	0.9722	0.227099	0.959481	0.17369
920	0.984578	0.07578	0.952998	0.176725
921	0.982143	0.113859	0.95624	0.179346
922	0.974026	0.16524	0.954619	0.179657
923	0.976461	0.176268	0.951378	0.189274
924	0.979302	0.113751	0.954619	0.174588
925	0.983157	0.095491	0.951378	0.178577
926	0.979911	0.114335	0.95624	0.176245
927	0.980722	0.094095	0.95624	0.18055
928	0.981737	0.099151	0.95624	0.174046
929	0.978896	0.132517	0.95624	0.176357
930	0.962257	0.297788	0.951378	0.18335
931	0.977882	0.163571	0.954619	0.176352
932	0.985795	0.052097	0.952998	0.181994
933	0.975244	0.171241	0.95624	0.181732
934	0.983157	0.097782	0.95624	0.180611
935	0.98194	0.120091	0.957861	0.177361

936	0.976055	0.173615	0.95624	0.167278
937	0.975649	0.152886	0.957861	0.16609
938	0.961851	0.277211	0.95624	0.180765
939	0.988636	0.061068	0.95624	0.172724
940	0.985593	0.067424	0.959481	0.173367
941	0.974026	0.17352	0.95624	0.170664
942	0.975649	0.169845	0.95624	0.177082
943	0.989245	0.046553	0.957861	0.16365
944	0.975649	0.132579	0.957861	0.170812
945	0.981331	0.111208	0.959481	0.162755
946	0.96875	0.231418	0.962723	0.163871
947	0.967938	0.174719	0.959481	0.174993
948	0.982346	0.089506	0.95624	0.17372
949	0.981737	0.115414	0.951378	0.182113
950	0.976867	0.175085	0.936791	0.224891
951	0.95901	0.277311	0.952998	0.175667
952	0.976258	0.155645	0.95624	0.175846
953	0.975446	0.150132	0.957861	0.174877
954	0.980925	0.112112	0.952998	0.178553
955	0.985998	0.070224	0.95624	0.180739
956	0.970373	0.190939	0.952998	0.182094
957	0.982955	0.110072	0.949757	0.174405
958	0.986201	0.075416	0.954619	0.17336
959	0.971185	0.181537	0.954619	0.171838
960	0.976461	0.155127	0.959481	0.175429
961	0.979302	0.131008	0.95624	0.170086
962	0.980317	0.125114	0.954619	0.173541
963	0.986607	0.052556	0.949757	0.23406
964	0.962865	0.304116	0.957861	0.181946
965	0.98194	0.123205	0.949757	0.182965
966	0.964286	0.237849	0.952998	0.187107
967	0.969765	0.20213	0.95624	0.18469
968	0.98539	0.05921	0.959481	0.177925
969	0.975649	0.180046	0.957861	0.189929
970	0.975041	0.144084	0.954619	0.192128
971	0.98681	0.053243	0.954619	0.199472
972	0.949067	0.386109	0.952998	0.246397
973	0.980114	0.137288	0.954619	0.191498
974	0.96733	0.237353	0.952998	0.185239

975	0.975244	0.163971	0.95624	0.182918
976	0.968547	0.214803	0.95624	0.177977
977	0.985187	0.069183	0.957861	0.184399
978	0.979911	0.138634	0.952998	0.179217
979	0.983969	0.110074	0.951378	0.192299
980	0.98194	0.116112	0.951378	0.180499
981	0.974432	0.185011	0.951378	0.183453
982	0.977679	0.174653	0.951378	0.180664
983	0.985187	0.075679	0.952998	0.174154
984	0.977273	0.137298	0.95624	0.186038
985	0.97362	0.178634	0.949757	0.192694
986	0.967938	0.235589	0.957861	0.197035
987	0.985187	0.073527	0.957861	0.173422
988	0.98194	0.111656	0.948136	0.180497
989	0.98539	0.064967	0.954619	0.176161
990	0.986201	0.049611	0.954619	0.180362
991	0.982752	0.118981	0.957861	0.182461
992	0.97849	0.109082	0.961102	0.175583
993	0.978896	0.122687	0.95624	0.177171
994	0.985998	0.048906	0.95624	0.182282
995	0.984984	0.080273	0.952998	0.173116
996	0.975446	0.182039	0.959481	0.181463
997	0.97707	0.122258	0.949757	0.176627
998	0.979911	0.118787	0.961102	0.18456
999	0.987419	0.045021	0.959481	0.195737

- Bộ 4

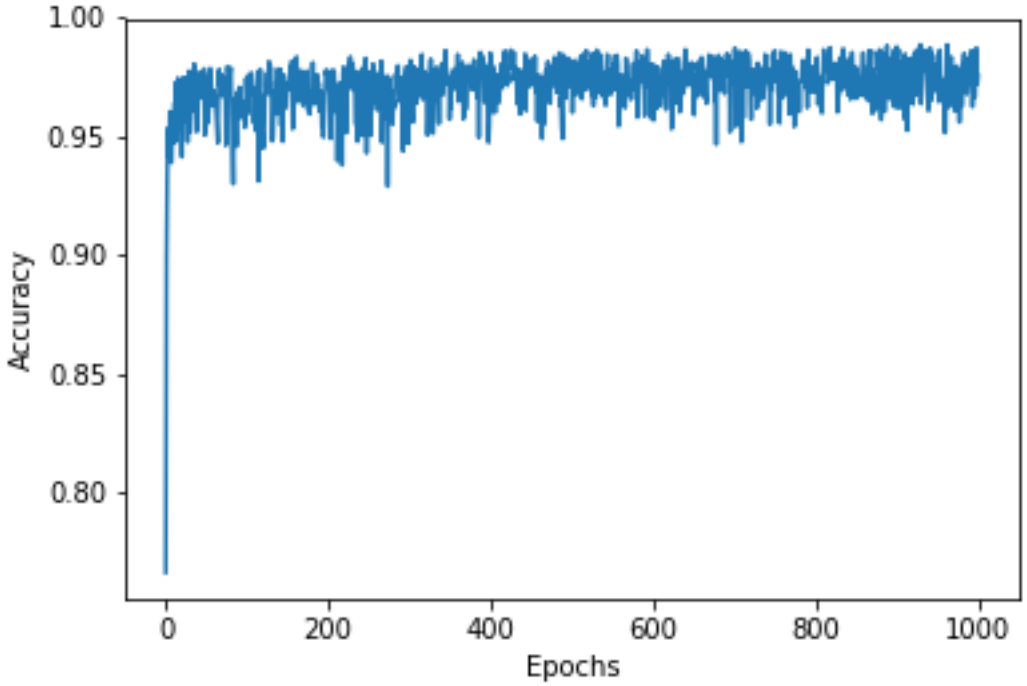


Figure 24 Bảng độ chính xác

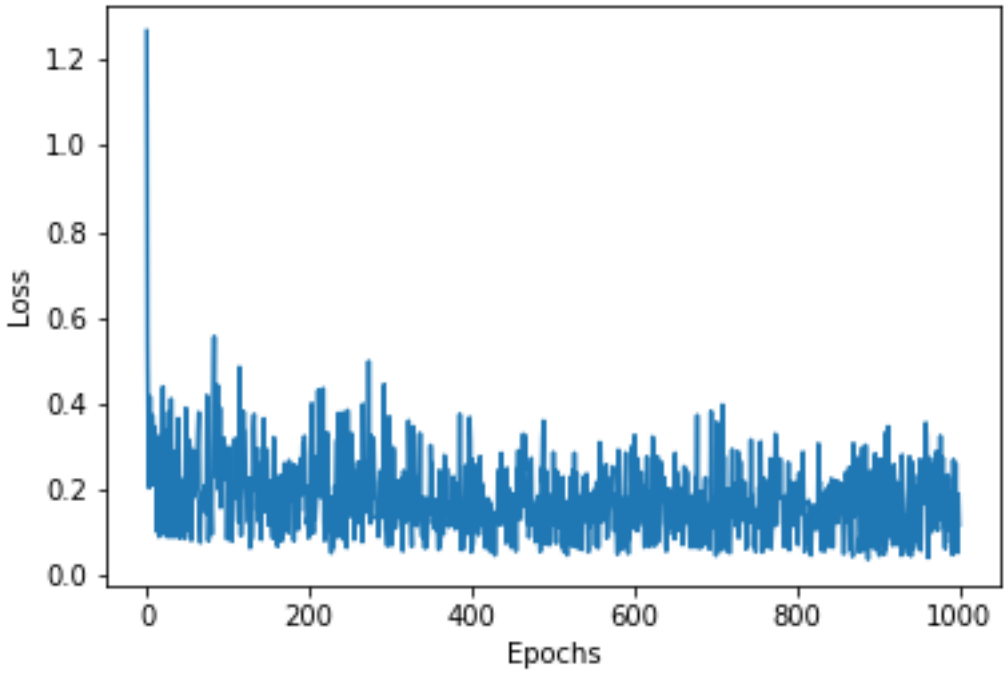


Figure 25 Bảng sai khác

Table 12 Bảng top 100 kết quả training

epoch	acc	loss	val_acc	val_loss
900	0.983766	0.051805	0.952998	0.241823
901	0.986607	0.046635	0.951378	0.247869
902	0.961851	0.271376	0.951378	0.217761
903	0.976055	0.153484	0.946515	0.231382
904	0.976055	0.169646	0.951378	0.248868
905	0.978084	0.12605	0.951378	0.232506
906	0.978084	0.128611	0.954619	0.22586
907	0.982955	0.049083	0.951378	0.239708
908	0.957995	0.331753	0.951378	0.212988
909	0.984781	0.064264	0.949757	0.230628
910	0.956778	0.314355	0.951378	0.215163
911	0.973417	0.160481	0.948136	0.228979
912	0.952516	0.346391	0.952998	0.22599
913	0.971388	0.197297	0.952998	0.21455
914	0.977882	0.108509	0.952998	0.219611
915	0.982752	0.088612	0.95624	0.217724
916	0.967938	0.228864	0.952998	0.209872
917	0.983157	0.070571	0.954619	0.214862
918	0.968953	0.209912	0.952998	0.210239
919	0.963474	0.258838	0.954619	0.213264
920	0.968547	0.224079	0.952998	0.217998
921	0.973417	0.175765	0.952998	0.213593
922	0.985795	0.070771	0.948136	0.218892
923	0.967532	0.216936	0.951378	0.238304
924	0.97849	0.129287	0.951378	0.21751
925	0.98681	0.074201	0.951378	0.218123
926	0.975244	0.160927	0.949757	0.223805
927	0.980722	0.104481	0.951378	0.223909
928	0.988636	0.04709	0.951378	0.234895
929	0.963677	0.281191	0.949757	0.222893
930	0.96875	0.205911	0.949757	0.214548
931	0.97362	0.178232	0.949757	0.220265
932	0.978693	0.124666	0.952998	0.21596
933	0.980925	0.076548	0.951378	0.221582
934	0.986404	0.066705	0.949757	0.225064
935	0.977882	0.135584	0.951378	0.225412
936	0.985998	0.050158	0.948136	0.237553

937	0.97849	0.148261	0.954619	0.236187
938	0.960836	0.276705	0.949757	0.219432
939	0.976258	0.160816	0.952998	0.212034
940	0.987216	0.045609	0.949757	0.238141
941	0.965097	0.264691	0.949757	0.223214
942	0.979505	0.117645	0.949757	0.229586
943	0.984375	0.062418	0.949757	0.231902
944	0.971388	0.194743	0.951378	0.220744
945	0.969968	0.188386	0.946515	0.239194
946	0.969968	0.20866	0.952998	0.221915
947	0.976055	0.176805	0.948136	0.278255
948	0.98052	0.07985	0.948136	0.229189
949	0.968344	0.234326	0.949757	0.210115
950	0.983766	0.069582	0.944895	0.225292
951	0.964489	0.270666	0.941653	0.248634
952	0.986404	0.058489	0.943274	0.239416
953	0.966315	0.213218	0.949757	0.222919
954	0.969968	0.194265	0.949757	0.223035
955	0.974229	0.184146	0.949757	0.231371
956	0.972606	0.168766	0.949757	0.225484
957	0.964286	0.235196	0.949757	0.236234
958	0.951502	0.355837	0.946515	0.231026
959	0.952313	0.323357	0.951378	0.230071
960	0.98336	0.083375	0.948136	0.227813
961	0.988636	0.041751	0.949757	0.242517
962	0.972606	0.188019	0.948136	0.223956
963	0.962257	0.273791	0.951378	0.232014
964	0.974026	0.175297	0.946515	0.225781
965	0.979708	0.106792	0.949757	0.224062
966	0.983563	0.10203	0.948136	0.227163
967	0.970576	0.197079	0.951378	0.22018
968	0.963474	0.259719	0.948136	0.230882
969	0.971185	0.16694	0.951378	0.209395
970	0.979505	0.111165	0.948136	0.216139
971	0.960633	0.288597	0.951378	0.215097
972	0.979099	0.119493	0.954619	0.214647
973	0.979708	0.105057	0.951378	0.226131
974	0.970373	0.186158	0.946515	0.22311
975	0.976055	0.158652	0.948136	0.217941

976	0.955966	0.325075	0.948136	0.241285
977	0.977882	0.129082	0.946515	0.221598
978	0.975244	0.189994	0.949757	0.222949
979	0.962662	0.274575	0.946515	0.234953
980	0.98052	0.116888	0.946515	0.228149
981	0.98052	0.121965	0.944895	0.224448
982	0.984984	0.062738	0.949757	0.236106
983	0.971388	0.206817	0.949757	0.232524
984	0.977882	0.119313	0.951378	0.219052
985	0.967735	0.234571	0.948136	0.227627
986	0.969765	0.22332	0.949757	0.219996
987	0.982346	0.104455	0.952998	0.225857
988	0.971794	0.192006	0.954619	0.218771
989	0.983766	0.076338	0.952998	0.21902
990	0.976664	0.15751	0.946515	0.233407
991	0.986201	0.04855	0.948136	0.228549
992	0.962459	0.269615	0.952998	0.232823
993	0.971794	0.164727	0.951378	0.219696
994	0.979708	0.107223	0.952998	0.224473
995	0.966518	0.2606	0.944895	0.271346
996	0.977679	0.113335	0.951378	0.243395
997	0.987216	0.053417	0.948136	0.256347
998	0.971794	0.188785	0.952998	0.232528
999	0.975649	0.117979	0.946515	0.242571

3. Tiến hành train với 2 mô hình Học Máy là Random Decision Forest và Super Vertors Machine

Từ kết quả trên, chúng em tiến hành tiếp tục thử nghiệm với hai mô hình học máy là Random Decision Forest và Super Vectors Machine với tập mẫu là bộ 4 của nhóm hơn 11 ngàn mẫu. Chúng em thu được kết quả như sau:

Mô hình Random Decision Forest

Kết quả thu được:

Table 13 Kết quả thu được từ mô hình RDF

BỘ HƠN 11 NGÀN MẪU	**SET 4**	**Do chinh xac:** **0.983272** True Positive: 1428 True Negative: 982 False Positive: 6 False Negative: 35 Precision 0.99581589958159. Recall: 0.976077

Mô hình Super Vectors Machine

Kết quả thu được :

Table 14 Kết quả thu được từ mô hình SVM

BỘ HƠN 11 NGÀN MẪU	**SET 4**	**Do chinh xac:**	**0.858984**
		True Positive:	1576
		True Negative:	690
		False Positive:	180
		False Negative:	192
		Precision	0.89749430523918.
		Recall:	0.891403

V. Kết luận

Từ kết quả trên mục IV.2, chúng em thu được bộ 8 kết quả khá khả quan. Các hệ số chính xác đều cao (trên 96% với bộ 11 ngàn mẫu, và 94% với bộ 5 ngàn mẫu). Tuy nhiên, trong quá trình phân tích chúng em nhận ra một số ưu nhưỡng điểm của mô hình cũng như là đặc trưng sau:

- Nhược điểm:
  - Bộ dữ liệu phụ thuộc và mã độc cung cấp bởi Insitute of System Security at Technische Universität Braunschweig.
  - Rank API của bộ mã độc trên là một trong những đặc trưng mang tính phân loại cao. Nhưng lại quá đặc thù cho từng họ mã độc của Drebin.
  - Do thời gian gấp rút, chúng em chưa tìm hiểu thêm được nhiều về mô hình Học Máy để có thể đưa ra góc nhìn khách quan nhất.
- Ưu điểm:
  - Phân loại có độ chính xác cao giữa mã độc và mã sạch đặc biệt là họ Drebin.
  - Khả năng áp dụng vào thực tiễn cao.
  - Tự động hóa 100% quá trình phân tích và xuất ra kết quả
- Hướng phát triển tiếp theo:
  - Xây dựng mô hình predict từ trọng số thu được từ mô hình.
  - Xây dựng một webapp để phục phụ nhu cầu của cộng đồng
  - Thu thập thêm các mẫu mã đôc, mã sạch từ cộng đồng thông qua webapp. Từ đó phát triển tiếp mô hình Học Máy.

TÀI LIỆU THAM KHẢO

- https://github.com/androguard.
- Alejandro Martín García, Raul Lara-Cabrera, David Camacho, Android malware detection through hybrid features fusion and ensemble classifiers: The AndroPyTool framework and the OmniDroid dataset, Information Fusion, Volume 52, December 2019, Pages 128-142.
- https://www.sec.cs.tu-bs.de/~danarp/drebin/index.html
- L.Đ. Thuan, P.V. Huong, L.T.H. Van, HQ. Cuong, H.V. Hiep, N.K. Khanh, Android Malware Detection Based on Deep Learning Using Convolutional Neural Network, tạp chí nghiên cứu khoa học và công nghệ quân sự, 8/2019, ISSN 1859 – 1043.
- Daniel Arp, Michael Spreitzenbarth, Malte Huebner, Hugo Gascon, and Konrad Rieck "Drebin: Efficient and Explainable Detection of Android Malware in Your Pocket", 21th Annual Network and Distributed System Security Symposium (NDSS), February 2014
- L. Shiqi, T. Shengwei, Y. Long, Y. Jiong, and S. Hua, “Androidmalicious code Classification using Deep Belief Network,” KSII Trans. Internet Inf. Syst., vol. 12, no. 1, pp. 454–475, 2018.\
- D. Li, Z. Wang, and Y. Xue, “Fine-grained AndroidMalware Detection basedon Deep Learning,” 2018 IEEE Conf. Commun. Netw. Secur., vol. 1, no. L, pp. 1–2, 2018
- H. Alshahrani, H. Mansourt, S. Thorn, A. Alshehri, A. Alzahrani, and H. Fu, “DDefender: Androidapplication threat detection using static and dynamic analysis,” 2018 IEEE Int. Conf. Consum. Electron., pp. 1–6, 2018.
- Inokuchi A., Washio T., Motoda H. (2000) An Apriori-Based Algorithm for Mining Frequent Substructures from Graph Data. In: Zighed D.A., Komorowski J., Żytkow J. (eds) Principles of Data Mining and Knowledge Discovery. PKDD 2000. Lecture Notes in Computer Science, vol 1910. Springer, Berlin, Heidelberg
- M. Ilayaraja and T. Meyyappan, "Mining medical data to identify frequent diseases using Apriori algorithm," 2013 International Conference on Pattern Recognition, Informatics and Mobile Engineering, Salem, 2013, pp. 194-199.doi:10.1109/ICPRIME.2013.6496471
- Ming-Yen Lin, Ming-Yen Lin, Ming-Yen Lin, Apriori-based frequent itemset mining algorithms on MapReduce, ICUIMC '12 Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication Article No. 76
- Berkhin P. A Survey of Clustering Data Mining, 2006 Techniques. In: Kogan J., Nicholas C., Teboulle M. (eds) Grouping Multidimensional Data. Springer, Berlin, Heidelberg

- U. Abdullah, J. Ahmad and A. Ahmed, "Analysis of effectiveness of apriori algorithm in medical billing data mining," 2008 4th International Conference on Emerging Technologies, Rawalpindi, 2008, pp. 327-331.doi: 10.1109/ICET.2008.4777523
- https://viblo.asia/p/machine-learning-tong-quan-ve-machine-learning-RQqKLxaOK7z
- https://viblo.asia/p/machine-learning-deep-learning-cho-nguoi-bat-dau-1VgZvEeYKAw
- https://machine-learning.viblo.asia/
- Tzung-Pei Hong, Chan-Sheng Kuo and Sheng-Chai Chi, "A fuzzy data mining algorithm for quantitative values," 1999 Third International Conference on Knowledge-Based Intelligent Information Engineering Systems. Proceedings (Cat. No.99TH8410), Adelaide, SA, Australia, 1999, pp. 480-483.
- O. S. Adebayo and N. AbdulAziz, "Android malware classification using static code analysis and Apriori algorithm improved with particle swarm optimization," 2014 4th World Congress on Information and Communication Technologies (WICT 2014), Bandar Hilir, 2014, pp. 123-128.
- https://www.av-test.org/en/statistics/malware/
- https://securelist.com/it-threat-evolution-q1-2019-statistics/90916/
- Mishra, Ratha, (2016), "Study of Random Tree and Random Forest Data Mining Algorithms for Microarray Data Analysis", International Journal on Advanced Electrical and Computer Engineering vol.3 issue 4.
- Michal Kedziora, Paulina Gawin, Michal Szczepanik and Ireneusz Jozwiak, Malware detection using machne learning algorithms and reverse engineering of android java code, International Journal of Network Security & Its Applications (IJNSA) Vol. 11, No.1, January 2019.
- Fairuz Amalina Narudin, Ali Feizollah, Nor Badrul Anuar, Abdullah Gani, Evaluation of machine learning classifiers for mobile malware detection, January 2016, Volume 20, Issue 1, pp 343–357.
- https://archive.org/details/2018-02-random-apk-collection
- Bernhard Sch¨olkopf, John C. Platt, John Shawe-Taylor, Alex J. Smola,and Robert C. Williamson. Estimating the support of a high-dimensionaldistribution. Technical Report MSR-TR-99-87, Microsoft Research,2000
- https://gs.statcounter.com/os-market-share/mobile/worldwide
- Le Duc Thuan, Pham Ngọc Hung, Pham Van Huong, Nguyen Duc Trung, Phát hiện mã độc dựa trên mạng niềm tin sâu, FAIR 2018, Ha Noi.

- Nikola Milosevic, Ali Dehghantanha, Kim-Kwang Raymond Choo, Machine learning aided Android malware classification, Computers & Electrical Engineering, Volume 61, July 2017, Pages 266-274
- Kang, BooJoong & Yerima, Suleiman & Sezer, Sakir & Mclaughlin, Kieran. (2016). N-gram Opcode Analysis for Android Malware Detection. International Journal on Cyber Situational Awareness. 1. 231-255. 10.22619/IJCSA.2016.1001011
- https://viblo.asia/p/machine-learning-tong-quan-ve-machine-learning-RQqKLxaOK7z
- Android Malware Detection: https://hub.packtpub.com/using-deep-learning-methods-to-detect-malware-in-android-applications/
- Convolution Neural Network: http://cs231n.github.io/convolutional-networks/
- Decison Tree: https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052
- Malware Detection: https://arxiv.org/pdf/1907.08356v1.pdf
- RandomForest: https://www.r-bloggers.com/machine-learning-basics-random-forest/
- Support Vector Machine: https://machinelearningcoban.com/2017/04/09/smv/
- Neural Network: https://skymind.ai/wiki/neural-network
- Neural Network and Deep Learning: http://neuralnetworksanddeeplearning.com
- Tensorflow - Tutorial: https://www.tensorflow.org/tutorials