# DroidRL: Feature selection for android malware detection with reinforcement learning

***Abstract***

Do tính chất open source của Android, lỗ hổng có thể bị khai thác trước các cuộc tấn công bằng malware ngày càng tăng. Học máy, dẫn đến sự phát triển vượt bậc trong việc phát hiện malware trên Android trong những năm gần đây, thường được áp dụng trong giai đoạn phân loại (**classification phase**). Do **mối tương quan giữa các đặc trưng bị bỏ qua** trong một số thuật toán lựa chọn đặc trưng dựa trên xếp hạng truyền thống nên việc áp dụng các mô hình lựa chọn đặc trưng dựa trên trình bao bọc (**wrapper-based**) là một chủ đề đáng để nghiên cứu. Mặc dù xem xét mối tương quan giữa các tính năng, nhưng các phương pháp dựa trên **trình bao bọc tốn nhiều thời gian để khám phá tất cả các tập hợp con đặc trưng hợp lệ** có thể có khi xử lý một số lượng lớn các đặc trưng của Android. **Để giảm chi phí tính toán cho việc lựa chọn đặc trưng dựa trên trình bao bọc, một framework có tên DroidRL được đề xuất.** Framework này triển khai **thuật toán DDQN** để thu được một tập hợp con các đặc trưng có thể được sử dụng để phân loại malware hiệu quả. Để chọn một tập hợp con các đặc trưng hợp lệ trên phạm vi lớn hơn, chính sách khai thác-khám phá (**exploration-exploitation**) được áp dụng trong **giai đoạn huấn luyện mô hình**. Mạng thần kinh tái phát (**RNN – Recurrent NN**) được sử dụng làm **mạng quyết định của DDQN** để cung cấp cho framewrok khả năng chọn tuần tự các đặc trưng. Việc nhúng từ (**word embedding**) được áp dụng **để biểu diễn đối tượng nhằm nâng cao khả năng của framework làm việc nhằm tìm ra mức độ liên quan về mặt ngữ nghĩa của các đối tượng.** Việc lựa chọn đặc trưng của framework thể hiện hiệu suất cao mà không cần bất kỳ sự can thiệp nào của con người và có thể được chuyển sang các nhiệm vụ lựa chọn đặc trưng khác với một số thay đổi nhỏ. **Kết quả thử nghiệm cho thấy hiệu quả rõ rệt khi sử dụng Random Forest làm công cụ phân loại của DroidRL, đạt độ chính xác 95,6% chỉ với 24 đặc trưng được chọn.**

1. ***Introduction***

Android là nền tảng điện toán phát triển nhanh nhất trên thiết bị đầu cuối di động. Vào năm 2021, có 1,39 tỷ điện thoại thông minh được sản xuất trên toàn thế giới và Android thống trị thị trường toàn cầu tới 72,2%. Tuy nhiên, là một hệ điều hành nguồn mở, Android đã bị nhiều malware tấn công. Theo báo cáo do Trung tâm tình báo mối đe dọa Qianxin (2020) công bố, tổng cộng **2,3 triệu** mẫu chương trình độc hại (**malicious programs**) đã bị chặn trên nền tảng Android vào năm 2020, với trung bình **6301 mẫu chương trình độc hại mới mỗi ngày**. Dòng **Trojan khai thác AdbMiner** đang **hoạt động tích cực** trong các cuộc tấn công, **chiếm giữ hàng chục nghìn thiết bị Internet vạn vật trên toàn thế giới và số lượng thiết bị Internet vạn vật bị bắt ở Trung Quốc là gần 1000**. Do đó, phần mềm độc hại Android là vấn đề trở nên nghiêm trọng đến mức nhiều nhà nghiên cứu nỗ lực tìm kiếm các phương pháp phát hiện hiệu quả.

Sự ra đời của học máy đã có tác động đáng kể đến việc phát hiện phần mềm độc hại Android cho giai đoạn phân loại. Hiện tại, các phương pháp phát hiện phần mềm độc hại trên Android được nâng cao có thể được phân loại thành phân tích tĩnh (Kouliaridis và cộng sự, 2020; Yan và Yan, 2018) và phân tích động (Kouliaridis và cộng sự, 2020; Papamartzivanos và cộng sự, 2014). Một số nhà nghiên cứu sử dụng các mô hình học máy tiên tiến như **Deep learning** (Bibi và cộng sự, 2020), **online learning** (Narayanan và cộng sự, 2017) hoặc **học tập theo nhóm (esemble learning)** (Mantoo, 2020) để **xác định nhiều lớp tấn công** hiệu quả trong môi trường Android.

**Loại bỏ các đặc trưng dư thừa hoặc không liên quan là một thủ tục quan trọng của học máy.** Babaagba và Ade sanya (2019) đã chứng minh tầm ảnh hưởng của kỹ thuật đặc trưng trong việc phát hiện phần mềm độc hại trên Android bằng cách **đối chiếu hiệu suất của mô hình khi áp dụng hoặc không áp dụng thuật toán lựa chọn đặc trưng.**

Là phương pháp lựa chọn đặc trưng thường được áp dụng, các mô hình lựa chọn đặc trưng Android dựa trên bộ lọc (**filter**) (Salah và cộng sự, 2020; Yildiz và Dogru, ˇ 2019; Zhang và cộng sự, 2021) **không thể sử dụng nguồn cấp dữ liệu phản hồi từ tính chính xác của trình phân loại trong việc phát hiện phần mềm độc hại của Android, do đó, thông tin tương quan giữa các tính năng khác nhau thu được từ trình phân loại sẽ bị bỏ qua**. Tuy nhiên, số lượng kết hợp có thể có của các đặc trưng này **lớn đến mức không thể tìm kiếm toàn diện trong trình bao bọc**- dựa trên phương pháp (Huda và cộng sự, 2016), **luôn phát sinh chi phí tính toán cao**.

Trong bài viết này, **một mô hình lựa chọn đặc trưng dựa trên trình bao bọc sử dụng DDQN (Mnih và cộng sự, 2015), DroidRL, được đề xuất để tự động chọn các tập hợp con đặc trưng Android hợp lệ.** Những đóng góp chính của bài viết này được tóm tắt như sau:

1. **Học tăng cường (RL – Reinforcement learning**) được tận dụng trong việc lựa chọn đặc trưng dựa trên trình bao bọc để giải quyết vấn đề về tập hợp con đặc trưng không thể thay đổi được của các đặc trưng Android thô. Chính sách học tập tăng cường và khai thác-khám phá được sử dụng trong **DroidRL để khám phá một tập hợp con đặc trưng tối ưu nhằm phát hiện phần mềm độc hại.**
2. DroidRL trình bày nguyên mẫu có thể mở rộng của thuật toán rút gọn đặc trưng dành cho máy học trong các tình huống khác. Bài viết này đề xuất một cách tiếp cận hiệu quả cao để người nghiên cứu xử lý lại các đặc trưng thô trên bộ dữ liệu của họ khi sử dụng mô hình học máy. DroidRL tận dụng tính chất học tăng cường để tự động thực hiện lựa chọn đặc trưng biểu mẫu **nhằm giảm kích thước, đủ để thay thế kỹ thuật tính năng tăng nặng nề trong nhiệm vụ phát hiện phần mềm độc hại.**
3. **Kích thước đặc trưng giảm đáng kể (1083 xuống 24) khi sử dụng framework DroidRL trong khi vẫn duy trì độ chính xác cao (95,6%).** Các thử nghiệm mở rộng chứng minh rằng **framework DroidRL hoạt động tốt hơn các phương pháp lựa chọn đặc trưng truyền thống,** cải thiện hiệu suất phát hiện trên nhiều bộ phân loại khác nhau.

**Bài viết có cấu trúc như sau:** Phần 2 giới thiệu các công trình liên quan về lựa chọn đặc trưng cho phần mềm độc hại Android và học tập tăng cường trong an ninh mạng. Phần 3 mô tả các nguyên tắc cơ bản của DroidRL được áp dụng trong việc lựa chọn đặc trưng phát hiện phần mềm độc hại trên Android. Phần 4 giới thiệu quy trình huấn luyện của DroidRL. Phần 5 trình bày bộ dữ liệu để thực hiện thí nghiệm so sánh, phương pháp trích chọn đặc trưng và quá trình tiền xử lý dữ liệu. Phần 6 thảo luận về kết quả thí nghiệm.

1. ***Related work***
   1. ***Feature selection for Android malware detection***

A screenshot of a computer

Description automatically generated

**Lựa chọn đặc trưng là quá trình chọn một tập hợp con từ bộ đặc trưng ban đầu để cải thiện hiệu quả phát hiện**. Được xác định **bởi liệu có độc lập với độ chính xác của bộ phân loại hay không**, các thuật toán chọn đặc trưng có thể được phân loại thành các **thuật toán dựa trên bộ lọc và trình bao bọc (filter or wrapper). Bảng 1 mô tả các đặc trưng của từng loại và sự khác biệt giữa chúng.** Yu và Liu (2003) đưa ra mối tương quan **lựa chọn đặc trưng dựa trên bộ lọc (FCBF)** đã tạo ra **những cải tiến** trong cách tiếp cận dựa trên bộ lọc truyền thống **nhằm giảm sự dư thừa giữa các đặc trưng có liên quan.** D và P (2020) đã phát hiện phần mềm độc hại trên Android bằng cách sử dụng **kỹ thuật dựa trên bộ lọc cải tiến:** thuật toán Relief dựa trên k-nearest Neighbor (KNN). Huda và cộng sự. (2016) đã áp dụng điểm xếp hạng của bộ lọc trong quy trình lựa chọn trình bao bọc và kết hợp các thuộc tính của trình bao bọc và bộ lọc với số liệu thống kê lệnh gọi API để phát hiện phần mềm độc hại dựa trên bản chất của các hành động lây nhiễm. Xu và cộng sự. (2016) đã sử dụng lựa chọn đặc trưng dựa trên tương quan (CFS) (Hall, 2000) để xác định và loại bỏ các tính năng dư thừa, **giảm 121.621 xuống còn 5000 đặc trưng**. Nguyên và cộng sự. (2016) chỉ sử dụng các đặc trưng mà deep learning chủ yếu khai thác để mô tả phần mềm độc hại và đạt độ chính xác phát hiện 96,76%. Allix và cộng sự. (2016a) đã tạo các khối cơ bản dưới dạng các đặc trưng, chuỗi các cấu trúc trong biểu đồ luồng điều khiển (**CFG**) chỉ có một điểm vào (**Entry point**) và một điểm thoát (**exit point**), do đó đại diện cho phần nhỏ nhất của chương trình luôn được thực thi hoàn toàn.

**Thuật toán lựa chọn đặc trưng dựa trên bộ lọc điển hình là dựa trên xếp hạng.** Mỗi đặc trưng được ấn định một điểm tùy theo tầm quan trọng của nó và sau đó N đặc trưng hàng đầu được chọn để đưa vào giai đoạn phân loại sau khi xếp hạng tất cả các đặc trưng. Hoàng và cộng sự. (2008) đã đề xuất một phương pháp xếp hạng đặc trưng không tham số để lựa chọn đặc trưng và thuật toán lựa chọn đặc trưng tham lam đã được sửa đổi. Vương và cộng sự. (2014) đã xếp hạng các quyền riêng lẻ dựa trên rủi ro của quyền riêng lẻ và nhóm quyền. Mahindru và Sangal (2020) đã áp dụng sáu phương pháp xếp hạng đặc trưng khác nhau để chọn các đặc trưng quan trọng, bao gồm lựa chọn đặc trưng tỷ lệ Gain, Chi-Square, Information-gain và phân tích hồi quy logistic. Trong một thử nghiệm khác (Mahindru và San gal, 2021), họ đã kết hợp sáu loại xếp hạng đối tượng riêng biệt và bốn loại phương pháp lựa chọn tập hợp con đối tượng riêng biệt để chọn ra các tập hợp con đối tượng hợp lệ.

Các mô hình học máy truyền thống có thể được tối ưu hóa để chọn các tập hợp đặc trưng hợp lệ. Youn (S, 2002) đã trình bày một thuật toán dựa trên Support Vector Machine (SVM) để lựa chọn đặc trưng nhằm giảm thời gian tính toán. D và P (2020) đề xuất thuật toán dựa trên KNN để lựa chọn đặc trưng thuật toán SVM tối ưu được áp dụng để phát hiện phần mềm độc hại với kết quả tương đương với hiệu suất của mạng nơ-ron.

Các thuật toán học máy tiên tiến như thuật toán di truyền (genetic algorithm) (Fatima và cộng sự, 2019) và mạng lưới thần kinh (Wang và cộng sự, 2020) cũng được sử dụng trong việc lựa chọn các tập hợp con đặc trưng. Đối với mạng thần kinh, điểm thu được từ tổng trọng số softmax của các đặc trưng đầu vào có thể được sử dụng làm đánh giá trong bộ chỉ báo để chọn các tập hợp con đặc trưng hợp lệ.

Từ cuộc thảo luận ở trên, **có thể đưa ra kết luận rằng có rất ít nghiên cứu sử dụng phản hồi từ tính chính xác của bộ phân loại trong việc phát hiện phần mềm độc hại trên Android**. Lựa chọn đặc trưng dựa trên **bộ lọc ít tốn kém về mặt tính toán hơn so với lựa chọn đặc trưng dựa trên trình bao bọc**, **nhưng mức độ liên quan giữa các đặc trưng khác nhau bị bỏ qua**, do đó có thể chọn một số lượng lớn các đặc trưng dư thừa trong khi xử lý các vectơ đặc trưng kích thước cao. Để cải thiện hiệu quả của trình phân loại phát hiện phần mềm độc hại, cần **giải quyết vấn đề kết hợp đặc trưng vô tận** trong các tập hợp con hợp lệ được chọn trong phương pháp dựa trên trình bao bọc trước đó.

* 1. ***Reinforcement learning in cyber security***

Các thuật toán phổ biến của học tăng cường là Q-learning (Melo, 2001), Deep Q Network (DQN) (Mnih và cộng sự, 2015) và Double Deep Q Network (DDQN) (Hasselt và cộng sự, 2015). DQN được giới thiệu bởi Mnih et al. để giải quyết vấn đề khó khăn khi sử dụng bảng Q cho không gian hành động và trạng thái liên tục, có chiều cao. DDQN đã có sự cải tiến đáng chú ý về DQN trong thuật toán huấn luyện. Phương pháp tạo giá trị Q mục tiêu được sửa đổi trong DDQN (Hasselt và cộng sự, 2015) để giải quyết việc đánh giá quá cao giá trị Q của hành động trong DQN truyền thống.

Các ứng dụng học tăng cường trong bảo mật phần mềm đã đạt được những cải tiến đáng kể trong những năm gần đây. CyberBattleSim (Nhóm, 2021) đã triển khai một tác nhân phòng thủ tự động giúp phát hiện và giảm thiểu các cuộc tấn công đang diễn ra dựa trên xác suất thành công được xác định trước, với môi trường mô phỏng được tham số hóa bởi cấu trúc liên kết mạng cố định và một tập hợp các lỗ hổng được xác định trước.

Đặc biệt trong việc phát hiện phần mềm độc hại, học tăng cường đã được áp dụng phổ biến trong phân loại phần mềm độc hại (Binxiang và cộng sự, 2019) hoặc tạo mẫu đối thủ (Fang và cộng sự, 2019; Rathore và cộng sự, 2020). Fang và cộng sự. (2019) đã đào tạo một tác nhân AI để tự động tạo các mẫu đối thủ bằng cách rewarding cho nó nếu phần mềm độc hại đã sửa đổi thoát khỏi sự phát hiện của bộ phân loại. Rathore và cộng sự. (2020) đã tạo ra phần mềm độc hại bằng cách sử dụng công nghệ học tăng cường để tối đa hóa tỷ lệ đánh lừa trong khi thực hiện các sửa đổi tối thiểu đối với ứng dụng Android. Để giải quyết vấn đề tốc độ học chậm trong trò chơi có chiều hướng Q-learning cao, Wan và cộng sự. (2017) đã áp dụng kỹ thuật Deep Q learning với mạng nơ-ron tích chập sâu (**Deep convolutional NN**) trong việc phát hiện phần mềm độc hại trên thiết bị di động, khởi tạo các giá trị chất lượng dựa trên trải nghiệm phát hiện phần mềm độc hại.

Trong công việc này, bằng cách kết hợp các ưu điểm của việc sử dụng kinh nghiệm hiện có trong khi tự động khám phá các tập hợp con tối ưu khác trong học tăng cường và việc sử dụng mức độ phù hợp của đặc trưng trong việc lựa chọn đặc trưng dựa trên trình bao bọc, **DroidRL đã giải quyết được các vấn đề của thuật toán lựa chọn đặc trưng truyền thống để tạo ra đặc trưng đó**. Giai đoạn sàng lọc nhanh hơn và phân loại phần mềm độc hại hiệu quả hơn

1. ***DroidRL framework***

Đối với framework DroidRL, nhiệm vụ chính là đào tạo tác nhân học cách chọn tuần tự các tập hợp con đặc trưng Android hợp lệ bằng cách tương tác với môi trường và sử dụng kiến ​​thức đã học của nó. Phần này mô tả cách khung DroidRL đạt được mục tiêu của nó.

* 1. ***Overview of DroidRL***

Hình 1 thể hiện sơ đồ nguyên lý của DroidRL. Phần cốt lõi của framework DroidRL được xây dựng bởi mạng quyết định (**decision network**) dựa trên DDQN (**Double Deep Q Network**). Trong mỗi bước, tác nhân tự trị thực hiện một cách độc lập một hành động do mạng quyết định **quyết định để chọn một đặc trưng vào trạng thái được quan sát** từ môi trường bằng cách sử dụng kiến ​​thức trước đó. **Để đánh giá chất lượng của tập hợp con đặc trưng và sự khác biệt của các đặc trưng riêng lẻ đã chọn**, **reward** của hành động được tạo ra từ trình phân loại phần mềm độc hại **được xác định bằng độ chính xác của phân loại phần mềm** độc hại sử dụng các đặc trưng được chọn làm đầu vào. Hơn nữa, trạng thái của tác nhân, hành động được chọn và reward của thời điểm này được lưu trong bộ nhớ phát lại **để huấn luyện mạng quyết định**. Chính sách thăm dò-khai thác (**exploration-exploitation**) được tăng cường để **giải quyết vấn đề chi phí tính toán do các tập hợp đặc trưng không thể cạn kiệt.**

A diagram of a computer network

Description automatically generated

***3.2. Key components in reinforcement learning***

**(1) Môi trường (Environment): Môi trường là nơi để tác nhân khám phá và nhận phản hồi.**

Trong framework của tác giả, **môi trường chứa tất cả các đặc trưng ứng viên** và **chịu trách nhiệm đưa trạng thái hiện tại của tác nhân vào bộ phân loại phần mềm độc hại sau mỗi hành động được thực thi**. **Độ chính xác của việc phân loại được trả lại cho tác nhân dưới dạng reward**. **Tổng số đặc trưng nhất định** và **độ dài của các tập hợp đặc trưng hợp lệ** **cần được chọn** được xác định trong **môi trường**. **Khi tác nhân đã chọn đủ đặc trưng theo độ dài đã khai báo của tập hợp con đặc trưng hợp lệ, môi trường sẽ hướng dẫn tác nhân kết thúc vòng khám phá (exploration) này, trả lại reward cuối cùng và tự thiết lập lại.**

**(2) Hành động (Action)**: **Hành động là bước quan trọng mà tác nhân trong học tăng cường cần thực hiện từ không gian hành động dựa trên kinh nghiệm (experience) và trạng thái hiện tại (current state) của nó.**

Trong framework DroidRL, **không gian hành động chứa các đặc trưng trong bộ đặc trưng thô được trích xuất thông qua các tệp APK đã dịch ngược.** **Trạng thái của tác nhân mô tả các đặc trưng hiện được chọn là kết quả của một loạt hành động.** **Nhiệm vụ chính của tác nhân là tìm ra tập hợp con đặc trưng tối ưu có khả năng phân biệt cao giữa phần mềm độc hại và phần mềm Android lành tính.**

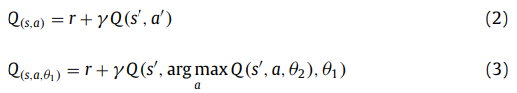
Đối với mỗi hành động trong framework DroidRL, **một đặc trưng không được chọn sẽ được thêm vào trạng thái**. **Thuật toán ε-Greedy** được sử dụng để thực hiện **sự cân bằng giữa việc thăm dò và khai thác**. Mỗi hành động được khám phá với **xác suất là ε**, trong khi **hành động có giá trị Q lớn nhất** được khai thác **với xác suất là 1 - ε**. Nhằm mục đích cho phép tác nhân trong giai đoạn đào tạo khám phá nhiều hơn ở giai đoạn đầu và khai thác nhiều hơn **bằng cách sử dụng trải nghiệm hiện có ở giai đoạn sau**, **một số cải tiến đã được thực hiện đối với thuật toán ε-Greedy như được hiển thị trong biểu thức. (1),** trong đó **episode** **là vòng huấn luyện hiện tại**, **E là tổng số vòng huấn luyện** và **P là tham số xác suất nằm trong khoảng từ 0 đến 1**. Thông tin chi tiết được đưa ra trong Phần 4.



**(3) Reward: Reward là phản hồi do sự tương tác giữa tác nhân và môi trường thông qua hành động**.

Trong bài viết này, **reward được xác định bởi độ chính xác của trình phân loại phần mềm độc hại trên Android, với các đặc trưng được chọn ở trạng thái hiện tại của tác nhân làm đầu vào**. **Tác nhân vào trạng thái s sau khi thực hiện hành động lựa chọn đặc trưng a**. Sau đó, **môi trường học tăng cường trả về** **reward tương ứng** từ bộ phân loại phần mềm độc hại **để đánh giá hàm giá trị hành động Q**

**Với mục tiêu đạt được giá trị Q cao nhất bằng hành động**, do đó **tác nhân có thể tìm thấy sự kết hợp đặc trưng hợp lệ dành cho độ chính xác cao nhất.** Trong phương trình. (2), **s và a lần lượt thể hiện trạng thái hiện tại và hành động được thực hiện ở bước hiện tại**. **r là reward nhận được**, **s’ đại diện cho trạng thái tiếp theo mà tác nhân đạt được và a’ đề cập đến hành động có thể đạt được giá trị Q cao nhất ở trạng thái tiếp theo**. Vì a’ cũng có thể được tính bằng hàm Q nên **công thức gốc tương đương với biểu thức. (3),** trong đó **θ1 và θ2 lần lượt biểu thị các tham số của hai mạng trong DDQN.**



***3.3. Decision network***

Trong DroidRL, **mạng quyết định là bộ não của tác nhân**. Khi việc khai thác được thực hiện, **tác nhân sẽ đặt trạng thái hiện tại được biểu thị bằng vectơ vào mạng quyết định của DroidRL, sau đó mạng quyết định sẽ trả về hướng dẫn hành động tiếp theo cho tác nhân**

Điều đáng nói là **độ dài trạng thái của tác nhân đang tiếp tục tăng lên**. Nó làm cho **đầu vào** của mạng quyết định có **độ dài không cố định**. Mạng quyết định của DroidRL **cần được thiết kế đặc biệt** vì **đầu vào** của **mạng thần kinh** thường **có hình dạng không đổi**. **RNN là một loại mạng lưới thần kinh được sử dụng rộng rãi trong xử lý ngôn ngữ tự nhiên. Do độ dài không cố định của các ngôn ngữ tự nhiên, các mạng giống RNN được thiết kế để chấp nhận đầu vào có độ dài không xác định.** Vì lý do này, **mạng quyết định của DroidRL áp dụng RNN và các biến thể của nó.**

**DroidRL cũng áp dụng một số thủ thuật có thể giúp ích cho hiệu quả đào tạo và khả năng lựa chọn đặc trưng**.

1. Nhúng từ (**word embedding**): **Đầu vào của mạng quyết định** của tác giả là **một chuỗi các đặc trưng được chọn.** Thay vì được trình bày dưới dạng one-hot vector, **việc nhúng từ được áp dụng để xử lý đầu vào**. **Nếu one-hot vector được sử dụng để biểu diễn chuỗi các đặc trưng** thì **toàn bộ ma trận đầu vào sẽ lớn và thưa thớt** và sẽ **dẫn đến một lượng tính toán và lưu trữ khổng lồ**. Ngoài ra, **không có thông tin ngữ nghĩa khi các đặc trưng được biểu thị bằng one-hot vector**, điều này **không có lợi cho mạng quyết định tìm ra mối tương quan giữa các đặc trưng.** **Việc áp dụng đặc trưng nhúng từ vào mạng quyết định của DroidRL có thể cải thiện khuôn khổ ở hai khía cạnh sau:**
2. **Nén một one-hot vector** thành **một vectơ dày đặc hơn** (**dense one**). **Word embedding** làm **giảm đáng kể kích thước đầu vào và cải thiện tốc độ đào tạo của mô hình.**
3. **Các vectơ đặc trưng được nén có nhiều ngữ nghĩa hơn.** Nhiệm vụ chính của DroidRL là chọn một tập hợp con các đặc trưng tối ưu. **Sau khi đặc trưng nhúng từ được thêm vào mạng quyết định, DroidRL có thể phân cụm các đặc trưng trong không gian nhiều chiều theo ngữ nghĩa của chúng.** Sau đó, nó **có thể tìm thấy tốt hơn các đặc trưng có thể kết hợp với các đặc trưng được chọn hiện tại**.
4. **Thứ tự đặc trưng (features ordering)**: **Có sự cân nhắc đặc biệt** đối với việc **áp dụng các phương pháp xử lý ngôn ngữ tự nhiên** cho **việc lựa chọn đặc trưng DroidRL.** **Ngôn ngữ tự nhiên có tính chất tuần tự, nghĩa là việc đặt lại hai từ trong một câu có thể khiến câu trở nên khó hiểu và vô nghĩa.** Tuy nhiên, **trong việc lựa chọn đặc trưng**, việc thay thế bất kỳ hai đặc trưng được chọn nào trong chuỗi đặc trưng sẽ không có bất kỳ ảnh hưởng nào đến việc đưa ra quyết định của mạng quyết định. Trạng thái đầu vào [1,2,3] và trạng thái đầu vào [1,3,2] giống nhau về ý nghĩa vì chúng chứa các đặc trưng giống nhau và sẽ tạo ra cùng một đầu ra trong mạng quyết định. **Trao đổi vị trí của bất kỳ hai đặc trưng nào ở trạng thái đầu vào sẽ không ảnh hưởng đến kết quả. Đặc điểm lựa chọn đặc trưng này khác với đặc điểm của ngôn ngữ tự nhiên.**

**Nếu thuộc tính cụ thể này không được giải quyết, nó có thể có tác động tiêu cực đến việc học mạng quyết định giống như RNN.**

**Một thủ thuật được áp dụng để giải quyết vấn đề này**. **Trước khi các đặc trưng được đưa vào mạng quyết định, chúng được sắp xếp theo chỉ mục.** Bằng cách này, **cùng một bộ đặc trưng có thể được đảm bảo chỉ tạo ra một đầu vào bất kể thứ tự lựa chọn đặc trưng**.

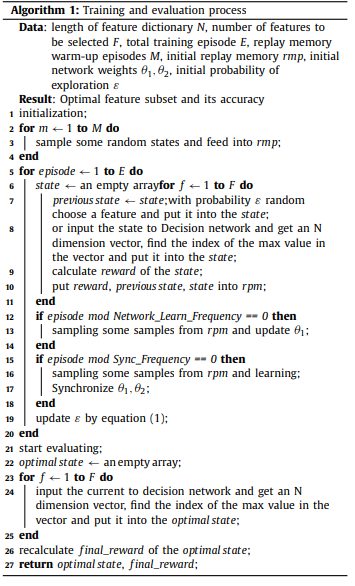
Sau khi áp dụng các thủ thuật trên, **cấu trúc mạng quyết định cuối cùng được hiển thị trong Hình 2**, **tác nhân đưa trạng thái của nó**, **một chuỗi đại diện cho các đặc trưng đã chọn**, **vào mạng quyết định.** **Lớp đầu tiên của mạng quyết định là lớp nhúng. Các đặc trưng được biểu thị bằng one-hot vector đi qua lớp nhúng và trở thành dense vector.** **Các vectơ này sau đó được đưa vào mạng giống RNN và cuối cùng đi vào lớp fully connected và lớp softmax.**

A diagram of a network

Description automatically generated

**4. Training phase**

**Quá trình đào tạo DroidRL được trình bày chi tiết trong phần này**. Thuật toán huấn luyện và thuật toán đánh giá được minh họa trong **Thuật toán 1**.



**Để khắc phục các vấn đề về dữ liệu tương quan và phân phối dữ liệu huấn luyện không cố định**, bộ nhớ phát lại (**relay memory**) được áp dụng trong DroidRL. **Trước khi quá trình đào tạo bắt đầu, DroidRL nhận được một số lấy mẫu ban đầu bằng cách chạy các tập khởi động và đưa chúng vào bộ nhớ phát lại**. Khi bắt đầu mỗi giai đoạn đào tạo, **trạng thái của tác nhân sẽ bị xóa và sau đó tác nhân bắt đầu chọn các đặc trưng**. **Mỗi tập kết thúc sau khi đã chọn đủ số lượng đặc trưng**. Trong các giai đoạn huấn luyện, **một chiến lược cân bằng giữa thăm dò và khai thác sẽ được áp dụng.** Như được mô tả **trong phương trình. (1)**, **tác nhân có xác suất thăm dò rất cao ngay từ đầu nhưng khả năng khai thác càng cao khi số episode tăng lên.**

**Trong trường hợp thăm dò, tác nhân chọn ngẫu nhiên một đặc trưng (không ở trạng thái của nó) trong một hành động.** **Việc thăm dò cho phép tác nhân thử nhiều cách kết hợp đặc trưng và không gian lựa chọn khả thi hơn.**

**Khi chính sách khai thác được thực thi, tác nhân sẽ sử dụng kinh nghiệm trước đó để chọn đặc trưng tối ưu**. **Thay vì chọn ngẫu nhiên một đặc trưng**, **tác nhân đưa trạng thái hiện tại của nó vào mạng quyết định và nhận một vectơ có cùng độ dài với từ điển đặc trưng biểu thị độ tin cậy của từng đặc trưng**. **Tác nhân lấy đặc trưng có độ tin cậy cao nhất làm hành động**. Nếu **đặc trưng có độ tin cậy cao nhất đã được chọn thì tác nhân sẽ chọn đặc trưng có độ tin cậy cao thứ hai**, v.v. Như đã đề cập trước đó, **sau mỗi lần một đặc trưng mới được thêm vào trạng thái, trạng thái sẽ được sắp xếp lại để đảm bảo tính nhất quán của cách biểu diễn bộ đặc trưng**. Sau mỗi hành động được thực hiện, **các đặc trưng trong trạng thái sẽ được sử dụng để phân loại phần mềm độc hại và lành tính.** **Độ chính xác của việc phân loại khi reward kết hợp với trạng thái trước đó và trạng thái hiện tại sau khi thực hiện hành động sẽ được đưa vào bộ nhớ phát lại.** Tác nhân **tiếp tục khai thác, khám phá (exploit-explore) cho đến khi chọn đủ tính năng.**

Sau khi đào tạo tất cả các giai đoạn đào tạo, tác nhân cuối cùng sẽ chạy một giai đoạn đánh giá. Trong epispode này, **mỗi bước của tác nhân sẽ sử dụng những gì đã học được trong giai đoạn đào tạo. Đầu ra của episode này là tập hợp con tính năng tối ưu cuối cùng.**

1. ***Experiment setup***

Phần này cung cấp thông tin về môi trường phần cứng để đào tạo DroidRL, tập dữ liệu của tác giả và cài đặt siêu tham số.

* 1. ***Training environment***

Tác giả đã thực hiện tất cả các thử nghiệm trên **server với GPU Tesla V100 đơn và CPU có hai lõi.** **GPU được sử dụng để tăng tốc quá trình đào tạo mạng quyết định trong DroidRL, trong khi việc đào tạo và dự đoán các bộ phân loại trong DroidRL chỉ sử dụng CPU**. Sau quá trình đào tạo, bộ phân loại của DroidRL có thể được trích xuất riêng biệt để thử nghiệm hoặc triển khai trên bất kỳ phần cứng nào có khả năng chạy thuật toán học máy.

* 1. ***Dataset***

Tập dữ liệu của DroidRL chứa **5000 benign sample** từ **Andro Zoo** và **5560 malware** từ **Drebin** để huấn luyện và kiểm tra mô hình.

Cả hai nguồn dữ liệu đều được sử dụng rộng rãi trong nghiên cứu những năm gần đây tập trung vào phát hiện phần mềm độc hại trên Android, giúp việc thực hiện các thử nghiệm so sánh với các phương pháp lựa chọn đặc trưng khác trở nên dễ dàng hơn. **AndroZoo (Allix et al., 2016b) cập nhật bộ sưu tập 16.000 nghìn APK khác nhau từ nhiều nguồn bao gồm cả Google Play, với mỗi ứng dụng được các sản phẩm AntiVirus khác nhau phân tích để gắn nhãn malware.** **Các mẫu malware trong nghiên cứu này chủ yếu được chọn từ Drebin (Arp et al., 2014b), một bộ dữ liệu được sử dụng phổ biến có chứa 5560 ứng dụng từ 179 nhóm malware khác nhau.**

**Phân tích tĩnh** được áp dụng trong công việc này, **trích xuất các quyền, hành động có ý định và mã hoạt động dưới dạng các đặc trưng thô từ các mẫu Android được APKtool và Androguard dịch ngược để củng cố thêm lựa chọn đặc trưng** dựa trên học tập.

**Trong tổng số 457 quyền và 126 hành động có mục đích thường được coi là có liên quan cao đến hành vi độc hại của các ứng dụng Android, được chọn trong bài viết này để xây dựng bộ đặc trưng ban đầu.** **Các quyền cho biết ứng dụng cần truy cập dữ liệu nhạy cảm nào của người dùng** (ví dụ: **danh bạ và SMS**), điều này rất cần thiết trong việc phát hiện malware trên Android. **Hành động có ý định là các đối tượng trừu tượng chứa thông tin về hoạt động sẽ được thực hiện cho một thành phần ứng dụng.**

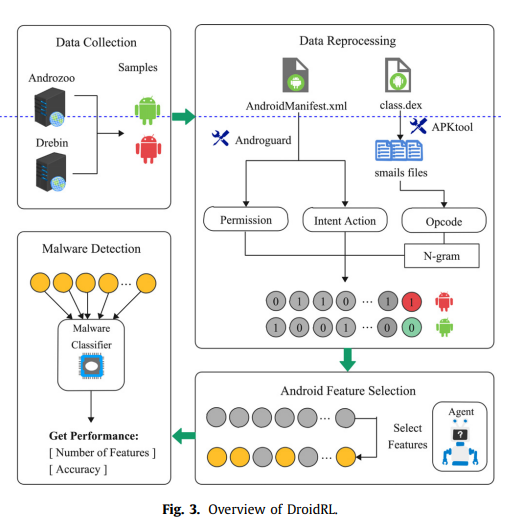
Sau khi phân tách **class.dex để tạo ra các tệp smalis**, mã byte Dalvik (ví dụ: gọi-trực tiếp invoke-direct) thu được thông qua việc quét trường phương thức trong các tệp smalis với biểu thức chính quy. **Opcode có được bằng cách ánh xạ mã byte Dalvik thành một chuỗi các chữ cái như được mô tả trong Bảng 2.**

A close-up of a computer screen

Description automatically generated

**Các đặc trưng của Opcode** được phân đoạn theo **N-gram để thu được trình tự chuyển đổi (tranformation sequence)**. Phương pháp **giảm kích thước** được sử dụng **để giải quyết vấn đề về số chiều lớn của vectơ đặc trưng do sự gia tăng số lượng N-gram với giá trị N**. Thứ nhất, tập N-gram gồm các mẫu độc hại là được trích xuất bằng quy trình trích xuất N-gram được đề xuất trong (ZHANG Zong-mei, 2019). **k N-gram tần số cao hàng đầu được chọn**, biểu diễn dưới dạng **Set = {x1, x2, x3, ··· , xk}.** Sau đó, **vectơ đặc trưng nhị phân k chiều feature = [m1, m2, m3, ··· , mk]** được xây dựng cho mẫu dựa trên tập đặc điểm này, trong đó mi là ”1” chỉ ra rằng tập N-gram của mẫu chứa phần tử xi trong tập đặc trưng.

**Từ cuộc thảo luận ở trên, toàn bộ quá trình phát hiện malware Android bằng DroidRL để lựa chọn đặc trưng được mô tả trong Hình 3**. Các mẫu ứng dụng Android được thu thập từ bộ dữ liệu Drebin và Andro Zoo, **được giải mã bằng các công cụ kỹ thuật đảo ngược của Android để trích xuất các quyền, hành động có mục đích và N-gram là các đặc trưng ban đầu**. Sau đó, lựa chọn đặc trưng DroidRL được áp dụng để chọn các tập hợp con đặc trưng hợp lệ từ các đặc trưng ban đầu. **Tập hợp con đặc trưng hợp lệ được lưu và sử dụng để xác thực hiệu suất của trình phân loại malware**, sử dụng số lượng đặc trưng và độ chính xác **làm chỉ mục đánh giá**. **Xác thực chéo 10 lần** được sử dụng trong thử nghiệm để đánh giá các mô hình và overfitting. Các đặc trưng được chọn của DroidRL sẽ được sử dụng để huấn luyện bộ phân loại cuối cùng nhằm phát hiện malware.



* 1. ***Hyperparameters setting***

Mô tả chi tiết và cài đặt các tham số của DroidRL được hiển thị trong Bảng 8. Các bộ phân loại trong DroidRL được xây dựng bằng scikit learn và tất cả các siêu tham số được sử dụng đều được đặt theo mặc định.

1. ***Experiment results***

Các câu hỏi nghiên cứu sau đây đã được đưa ra để giúp theo dõi quá trình tiến hành thí nghiệm:

RQ1 Is the result produced by the DroidRL framework stable when selecting only a dozen features from a high-dimensional exploration space (e.g., 1083, the dimensionality of the original feature vector)? **(RQ1 Kết quả do framework DroidRL tạo ra có ổn định khi chỉ chọn hàng chục đối tượng từ không gian khám phá nhiều chiều (ví dụ: 1083, số chiều của vectơ đặc điểm ban đầu) không?)**

RQ2 Does the decision network learn the key information conducive to the next optimal feature selection in the process of training? **(RQ2 Mạng quyết định có tìm hiểu thông tin quan trọng có lợi cho việc lựa chọn đặc trưng tối ưu tiếp theo trong quá trình đào tạo không?)**

RQ3 What is the performance of different classifiers using the optimal feature subset selected by the DroidRL framework as input? **(RQ3 Hiệu suất của các bộ phân loại khác nhau bằng cách sử dụng tập hợp con đặc trưng tối ưu được chọn bởi framework DroidRL như đã nêu là gì?)**

RQ4 What is the impact on the training time of malware classifiers of using the feature subset selected by DroidRL rather than the original features? **(RQ4 Tác động như thế nào đến thời gian đào tạo của bộ phân loại malware khi sử dụng tập hợp con đặc trưng được DroidRL chọn thay vì các đặc trưng ban đầu?)**

RQ5 How is the performance of the DroidRL framework compared with other advanced methods in related work? **(RQ5 Hiệu suất của khung DroidRL so với các phương pháp nâng cao khác trong các lĩnh vực liên quan như thế nào?)**

* 1. ***Stability evaluation of feature selection results***

**Chìa khóa để áp dụng phương pháp học tăng cường nhằm khám phá tập hợp con các đặc trưng tốt nhất nhằm phát hiện malware trên Android là tìm ra sự kết hợp tốt nhất giữa các đặc trưng.** Tuy nhiên, với chiến lược khai thác thăm dò, DroidRL có thể thực hiện các hành động khác nhau ngay cả trong cùng một trạng thái. Có những khác biệt không thể tránh khỏi trong tập hợp con đặc trưng tối ưu được DroidRL chọn. Hơn nữa, không gian có thể được khám phá bằng học tăng cường là rất lớn. Chỉ có khoảng **24 trong tổng số 1083 tính năng trong thử nghiệm được chọn làm đầu vào để phát hiện phần mềm độc hại**. **Các kết quả khác nhau cũng có thể gây ra sự mất ổn định của kết quả thí nghiệm.**

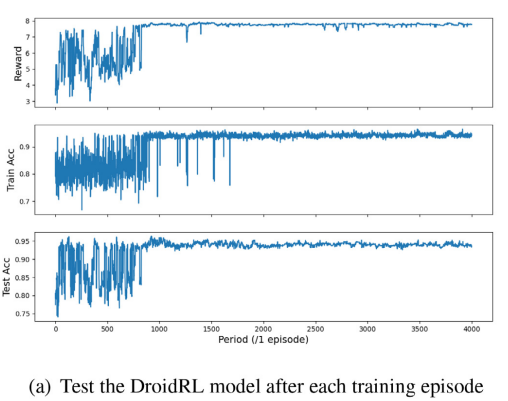
Trong nỗ lực xác minh xem **tính ngẫu nhiên do chiến lược khai thác-thăm dò mang lại có ảnh hưởng đến độ chính xác phát hiện cuối cùng hay không và để chứng minh thêm tính ổn định của kết quả lựa chọn đặc trưng của DroidRL**, thử nghiệm này sử dụng cây quyết định (DT - decision tree) làm công cụ phân loại và đã kiểm tra độ chính xác bằng **cách sử dụng kết quả của 5 nhóm đặc trưng phụ khác nhau thu được từ DroidRL làm đầu vào. Kết quả được minh họa trong hình 4.**

A graph with numbers and lines

Description automatically generated

**Mặc dù kết quả cuối cùng của các tính năng được chọn bằng phương pháp học tăng cường trong 5 thử nghiệm không hoàn toàn giống nhau, nhưng có thể thấy từ Hình 4 rằng độ chính xác phát hiện luôn nằm trong khoảng 92%-95% với một số điểm xuất hiện sai lệch do tính ngẫu nhiên do chiến lược thăm dò-khai thác mang lại. Do đó, mặc dù hành động mà tác nhân thực hiện mỗi lần có thể không hoàn toàn giống nhau khi khám phá sự kết hợp vô tận của các tập hợp con đặc trưng, nhưng độ chính xác phát hiện của việc sử dụng các đặc trưng đã chọn để phân loại phần mềm độc hại nhìn chung vẫn ổn định.**

* 1. ***Evaluation of the learning procedure of DroidRL***

A graph of a graph

Description automatically generated with medium confidence

Để minh họa quy trình học của mạng quyết định trong framework DroidRL, reward, phân loại đào tạo độ chính xác (Train Acc trong Hình 5) và độ chính xác phân loại thử nghiệm (Kiểm tra Acc trong Hình 5) trong mỗi tập đào tạo đã được theo dõi. Độ chính xác trong một tập (ví dụ: tập đánh giá đào tạo hoặc tập đánh giá thử nghiệm) thu được từ trình phân loại bằng cách sử dụng các đặc trưng được chọn làm đầu vào. Trong giai đoạn đào tạo, tác nhân chọn các đặc trưng sử dụng chiến lược khai thác-thăm dò và mạng quyết định ở chế độ đào tạo; Train Acc là khả năng phân loại malware chính xác bằng cách chạy một giai đoạn đánh giá đào tạo trên tập dữ liệu đào tạo. Trong giai đoạn đánh giá thử nghiệm, quyết quy trình lựa chọn đặc trưng của tác nhân chỉ được hướng dẫn bởi mạng định mà không cần thăm dò ngẫu nhiên; Test Acc là mức trung bình của độ chính xác trong 5 đợt đánh giá thử nghiệm.

Sau mỗi đợt đào tạo, framework DroidRL được kiểm tra bằng cách chạy 5 đợt đánh giá thử nghiệm trong Hình 5(a). Ngoài ra, lấy 50-episode huấn luyện làm một giai đoạn, framework DroidRL đã được kiểm tra sau mỗi giai đoạn trong Hình 5(b) với cùng một phương pháp để tính toán độ chính xác phân loại kiểm tra. Hơn nữa, độ chính xác huấn luyện của một giai đoạn được ghi nhận là độ chính xác phân loại malware trung bình của 50-episode huấn luyện.

Trong thử nghiệm này, **Bộ nhớ ngắn hạn dài (LSTM) (Graves, 2012) đóng vai trò là mạng quyết định của DroidRL vì nó có bộ nhớ theo ngữ cảnh tốt hơn RNN**. Cây quyết định (DT) là được áp dụng làm công cụ phân loại Các xu hướng tương tự được quan sát cả trong Hình. 5(a) và 5(b), vì cả hai kết quả đều hiển thị reward ngày càng tăng và độ chính xác phân loại malware. Tuy nhiên, có thể thấy sự khác biệt đáng kể trong hai hình, độ chính xác phân loại huấn luyện dao động rất lớn trong Hình 5(a), nhưng vẫn tương đối thấp trước 20 periods (ví dụ: 1000 episodes) và sau đó tăng lên mức trung bình. trạng thái ổn định cao hơn sau trong Hình 5 (b). Ngoài ra, **độ chính xác của phân loại kiểm tra luôn cao hơn một chút so với độ chính xác của phân loại đào tạo**. Kết quả thực nghiệm trên có thể được giải thích như sau:

1. **Cả hai số liệu đều cho thấy độ chính xác trong phân loại reward và malware ngày càng tăng, cho thấy hiệu suất học tập của framework DroidRL**.
2. Trong các tập huấn luyện của DroidRL, thuật toán **e-Greedy được sử dụng để cân bằng giữa khai thác và thăm dò** nhưng không thể tránh khỏi việc lựa chọn một số đặc trưng dư thừa hoặc không liên quan trong quá trình khám phá. **Như được trình bày trong 5(a), độ chính xác huấn luyện dao động luôn cao hơn độ chính xác của tập kiểm tra. Sau khoảng 20 periods, Độ chính xác của tập kiểm tra trở nên ổn định nhờ sử dụng kinh nghiệm hiện có trong khi độ chính xác trong huấn luyện vẫn có một số giảm đột ngột do thuật toán e-Greedy gây ra.**
3. Sau khi sử dụng độ chính xác trung bình là 50 episode, độ chính xác phân loại huấn luyện trong Hình 5(b) ổn định hơn trong 20 episodes đầu tiên so với Hình 5(a), điều này cho thấy rõ xu hướng thay đổi của nó. Do xác suất khám phá của thuật toán e-Greedy giảm dần ở giai đoạn sau, tác nhân có xu hướng sử dụng kinh nghiệm hiện có từ mạng quyết định để lựa chọn đặc trưng. Nó dẫn đến độ chính xác phân loại đào tạo cao hơn và ổn định hơn, cũng như đường cong chính xác giống với giai đoạn thử nghiệm trong các giai đoạn sau, như trong Hình 5(b).
   1. ***Comparison with different decision networks***

Thông thường, cần làm rõ các tiêu chí lựa chọn mạng quyết định cho thuật toán dựa trên học tăng cường. Do đó, các thử nghiệm so sánh với các mạng quyết định khác nhau được tiến hành và kết quả được mô tả chi tiết trong phần này.

**Hiệu quả để xử lý dữ liệu có đặc điểm trình tự, RNN và các biến thể của nó có khả năng khai thác thông tin thời gian và ngữ nghĩa trong dữ liệu đầu vào và được áp dụng phổ biến để dự đoán nội dung sau theo ngữ cảnh trong xử lý ngôn ngữ tự nhiên.** Do đó, framework DroidRL đã sử dụng mạng giống **RNN làm mạng quyết định để dự đoán đặc trưng sẽ được chọn trong bước tiếp theo dựa trên các đặc trưng Android đã chọn ở các bước trước.**

Với nỗ lực khám phá mạng thần kinh tái phát nào phù hợp nhất làm mạng quyết định của DroidRL, thử nghiệm này đã **áp dụng RNN, Bộ nhớ ngắn hạn dài (LSTM) (Graves, 2012) hoặc Đơn vị tái phát có kiểm soát (GRU – Gated Recurrent Unit) (Cho et al. , 2014) tương ứng là mạng quyết định cho việc đào tạo**

Trong thử nghiệm, **xác thực chéo 10 lần được sử dụng trên tập dữ liệu được xáo trộn**. Độ chính xác trung bình trên bộ thử nghiệm được lấy làm độ chính xác phân loại. Kết quả được trình bày trong hình 6.

A graph of a number of different features

Description automatically generated

Như các số liệu kết quả thử nghiệm phác họa, với nhiều đặc trưng Android được chọn hơn trong tập hợp con đặc trưng hợp lệ cuối cùng, độ chính xác cao hơn đều đặn được chứng kiến ​​trong RNN, GRU và SLTM. **Biến động chính xác ổn định nhất khi sử dụng GRU làm mạng quyết định**. **Mô hình đạt được hiệu suất tốt nhất (ví dụ: độ chính xác 95,6%) với 24 tính năng được chọn trong LSTM làm đầu vào để phân loại malware và chi phí tính toán đã giảm 97,78%.**

Dựa trên các kết quả thử nghiệm trên, các giải thích sau đây được kết luận theo nguyên tắc của framework DroidRL. Học tăng cường có tác động đáng kể đến việc chọn tập hợp con đặc trưng Android tối ưu. **Thông qua LSTM truyền thống làm mạng quyết định, DroidRL đã giảm 97,78% chi phí tính toán và duy trì độ chính xác 95,6%.**

A screenshot of a computer

Description automatically generated

A graph of a number of classes

Description automatically generated with medium confidence

A graph of a number of classes

Description automatically generated

A graph of a number of classes

Description automatically generated with medium confidence

A graph with red and blue lines

Description automatically generated

* 1. ***Comparison with different classifiers***

Để chứng minh hiệu suất của mô hình, tác giả đã tiến hành một loạt thử nghiệm so sánh để tìm ra sự kết hợp giữa số lượng đặc trưng khác nhau và các phân loại khác nhau. Như được hiển thị trong Hình 7, trục tung biểu thị số lượng tính năng tối đa được chọn bằng thuật toán dựa trên học tăng cường để phát hiện malware. Chính xác hơn, để làm cho quy trình học tập hiệu quả và có ý nghĩa, số lượng đặc trưng được mô hình lựa chọn bị giới hạn **từ 6 đến 24**.

**Có thể thấy trong Hình 7, khi số lượng đặc trưng được chọn tương đối ít thì độ chính xác đạt khoảng 90%. Khi số lượng đặc trưng được chọn tăng lên, độ chính xác dần dần được cải thiện, nâng lên 95%.**

Hình 7 cho thấy rằng học tăng cường có thể được áp dụng cho đặc trưng phát hiện malware trên android để chọn các bộ đặc trưng hợp lệ để phân loại malware. Độ chính xác ổn định trên Rừng ngẫu nhiên (RF), Cây quyết định (DT) và SVM. Có thể thấy trong Bảng 6 và 7 rằng **DT, RF và SVM đạt được độ chính xác cao hơn với ít đặc trưng hơn và có hiệu suất ổn định.** **Các mô hình KNN hoạt động không tốt lắm, thể hiện qua sự biến động về độ chính xác trong các số liệu.**

Sau khi thực nghiệm kiểm chứng, người ta thấy rằng các siêu tham số k khác nhau phù hợp cho việc phân loại các số đặc trưng khác nhau nên độ chính xác dao động rất lớn theo số lượng đặc trưng được chọn.

A screenshot of a document

Description automatically generated

* 1. ***Comparison of the classifiers’ training time***

**Độ phức tạp tính toán của thuật toán học máy tăng lên theo số lượng mẫu và đặc trưng.** Trong khi việc **tăng kích thước mẫu có thể mang lại sự chắc chắn hơn cho việc phân loại**, thì **số lượng đặc trưng ngày càng tăng có thể mang lại sự dư thừa.** **Một số lượng lớn các đặc trưng cũng làm tăng độ phức tạp tính toán và nguồn lực cần thiết cho việc đào tạo**. Để đánh giá khả năng duy trì hiệu quả đào tạo của DroidRL, thử nghiệm này sử dụng các tập hợp đặc trưng có độ dài khác nhau được DroidRL chọn để đào tạo các trình phân loại malware khác nhau. Để đo lường sự cải thiện về hiệu quả đào tạo, tác giả đã **tính tỷ lệ giữa thời gian tiêu tốn để đào tạo bộ phân loại bằng cách sử dụng một tập hợp con các đặc trưng với thời gian sử dụng bộ tính năng đầy đủ.**

Như được hiển thị trong Bảng 3, việc sử dụng các tập hợp con để huấn luyện các mô hình sẽ cải thiện đáng kể hiệu quả huấn luyện. **Sở dĩ tỷ lệ tương đối lớn trên Random Forest chủ yếu nằm ở việc có nhiều cây con trong mô hình.** Có giới hạn dưới về thời gian huấn luyện cây con khi số lượng đối tượng giảm.

**Một hiện tượng đáng chú ý khác là tỷ lệ thời gian đào tạo không được cải thiện hoàn toàn theo số lượng đặc trưng, thậm chí có lúc còn giảm xuống. Điều này chỉ ra rằng DroidRL lọc ra các đặc trưng hữu ích cho việc phân loại và các đặc trưng được bổ sung cho phép bộ phân loại dễ dàng tìm thấy ranh giới quyết định hơn, từ đó tăng tốc độ đào tạo.**

* 1. ***Comparison with related work***

Để so sánh toàn diện giữa framework DroidRL được đề xuất và các phương pháp phát hiện malware trên Android có liên quan, **tác giả đã tiến hành một thử nghiệm so sánh dựa trên tập dữ liệu trong công việc của mình. Việc triển khai các phương pháp phát hiện trong phần này đề cập đến code of the work (**Molina-Coronado và cộng sự, 2023).

Đầu tiên, để minh họa tác động của việc lựa chọn đặc trưng trong việc phát hiện malware trên Android dựa trên máy học, **tác giả đã so sánh DroidRL với các phương pháp phát hiện malware trên Android khác mà không áp dụng phương pháp chọn đặc trưng. Kết quả được hiển thị trong Bảng 4**. **Framwork DroidRL được đề xuất hoạt động tốt hơn DroidDet (Zhu và cộng sự, 2018), HMMDetector (Canfora và cộng sự, 2016) và Drebin (Arp và cộng sự, 2014a) với độ chính xác cao hơn và ít tính năng được sử dụng hơn trong quá trình phát hiện, điều này chứng tỏ các tính năng được chọn bằng phương pháp học tăng cường có mức độ liên quan cao đến các thuộc tính của malware**. Với **190.072 tính năng được trích xuất nhiều hơn so với DroidRL, Ma maDroid (Onwuzurike và cộng sự, 2019) chỉ đạt được độ chính xác cao hơn 0,033, tăng độ chính xác với chi phí vận hành khổng lồ. Việc trích xuất 190.096 tính năng và phát hiện phần mềm độc hại với nhiều tính năng như vậy rất tốn thời gian. Việc phát hiện phần mềm độc hại trên Android hiệu quả hơn nhiều về mặt tính toán với các tập hợp tính năng hợp lệ chỉ chứa 24 tính năng được chọn bằng phương pháp học tăng cường được sử dụng trong mô hình của tác giả.**

**Để chứng minh rõ hơn tính tối ưu của các đặc trưng do DroidRL lựa chọn, hiệu suất của DroidRL được so sánh với các phương pháp lựa chọn đặc trưng khác được sử dụng để phát hiện malware** dựa trên máy học. **Kết quả của các thí nghiệm này được thể hiện trong Bảng 5 với số lượng đặc trưng và độ chính xác phát hiện trên mỗi hình thức làm chỉ số**. Tác giả đã triển khai các phương pháp này với một số đặc trưng cụ thể được sử dụng để phát hiện như được hiển thị trong bảng (không nhất thiết phải giống số lượng trong tác phẩm gốc).

So với các phương pháp lựa chọn đặc trưng truyền thống khác được liệt kê trong bảng, **DroidRL đạt được độ chính xác cao hơn với số lượng tính năng được sử dụng trong quá trình phát hiện ít hơn, cho thấy rằng học tăng cường vẫn giữ được khả năng lọc các tính năng tối ưu.** Bởi vì, **là phương pháp lựa chọn đặc trưng dựa trên trình bao bọc có thể sử dụng phản hồi từ bộ phân loại để đánh giá tập hợp con đối tượng, DroidRL có thể chọn một tập hợp con đối tượng tối ưu.**

1. ***Conclusion***

**DroidRL được đề xuất áp dụng thuật toán DDQN cho giai đoạn lựa chọn đặc trưng để chọn tập hợp con đặc trưng tối ưu nhằm phát hiện malware trên Android.** Đặc biệt, **mạng giống RNN được ứng dụng làm mạng quyết định trong DDQN nhờ khả năng xử lý các chuỗi có độ dài thay đổi**. **Với mục đích tìm ra mối tương quan giữa các đặc trưng, DroidRL sử dụng đặc trưng nhúng từ để thể hiện một cách chính xác các đặc trưng**. Trong giai đoạn đào tạo, chính sách đào tạo được sử dụng để tăng không gian tìm kiếm đặc trưng của DroidRL. **Các thử nghiệm trên Drebin và Androzoo chứng minh rằng framework DroidRL cho thấy hiệu suất tốt hơn so với các mô hình trích xuất tính năng tĩnh truyền thống, cải thiện rõ rệt hiệu suất phát hiện trên nhiều bộ phân loại khác nhau.** DroidRL được chứng minh là có hiệu quả trong các nhiệm vụ lựa chọn đặc trưng và hy vọng nó sẽ đóng vai trò là một yếu tố để xây dựng một trình phát hiện malware mạnh mẽ trong tương lai.