Detecção Hierárquica Confiável de Malware de Android Baseado em Arquiteturas CNN

Jhonatan Geremias[1], Eduardo K. Viegas[1], Altair O. Santin[1], Pedro Horchulhack[1], Alceu de S. Britto[1]

[1]Programa de Pós-Graduação em Informática (PPGIa)
Pontifícia Universidade Católica do Paraná (PUCPR)
80.215-901 - Curitiba - PR

{jhonatan.geremias, eduardo.viegas, santin, pedro.horchulhack, alceu.britto}@ppgia.pucpr.br

Abstract. *In this article, we propose a reliable method for hierarchical detection of Android malware using CNN. The method consists of two steps: hierarchical classification of malware applications and selection of highly reliable applications using rejection. Experiments conducted on a new dataset with over 26,000 Android applications, divided into 29 malware families, showed that CNN for malware detection is unable to provide high detection accuracy. In contrast, the proposed model is capable of reliably detecting malware applications, improving the TN rates by up to 5.5% and the average TP rate of accepted malware families by up to 12.7%, while rejecting only 10% of Android applications.*

Resumo. *Neste artigo, propomos um método confiável de detecção hierárquica de malware Android utilizando CNN. O método possui duas etapas: classificação hierárquica de aplicativos de malware e seleção de aplicativos altamente confiáveis utilizando rejeição. Experimentos realizados em um novo dataset com mais de 26 mil aplicativos Android, divididos em 29 famílias de malware, mostraram que a CNN para detecção de malware é incapaz de fornecer alta precisão de detecção. Em contraste, o modelo proposto é capaz de detectar malware de forma confiável em aplicativos, melhorando as taxas de TN em até 5,5% e a taxa média de TP das famílias de malware de aplicativos aceitos em até 12,7%, enquanto rejeita apenas 10% dos aplicativos Android.*

1. Introdução

O Android é atualmente o sistema operacional móvel mais popular, com estimativas de 2,5 bilhões de dispositivos ativos, respondendo por mais de 70% da participação no mercado de *smartphones* [inMobi 2021]. Infelizmente, o número de aplicativos Android indesejados, como *malwares*, *adwares* e *bloatwares*, também está aumentando, afetando atualmente cerca de 24% de todos os usuários do Android [Kotzias et al. 2021]. Surpreendentemente, 67% de todos os aplicativos Android indesejados foram originados de mercados de aplicativos oficiais [Kotzias et al. 2021], demonstrando que as abordagens atuais de detecção de *malware* do Android não estão conseguindo proteger seus usuários.

Ao longo dos últimos anos, várias técnicas foram propostas para a detecção de *malware* no Android, por meio de abordagens de análise dinâmicas ou estáticas

[Qiu et al. 2020]. As abordagens baseadas em análise dinâmica dependem da execução do aplicativo Android, monitorado em um ambiente sandbox, enquanto avaliam continuamente o comportamento do aplicativo buscam por impressões digitais maliciosas [Spreitzenbarth et al. 2013]. Como resultado, esta abordagem é capaz de avaliar de forma realista o comportamento do aplicativo, porém, representa um grande desafio quanto à geração de um estímulo adequado no aplicativo a fim de desencadear as atividades maliciosas, dado que existem aplicativos capazes de permanecer ocultos mesmo quando executados em ambiente sandbox [Vidas and Christin 2014]. Por outro lado, as técnicas baseadas em estática avaliam as características do aplicativo de maneira *offline*, de acordo com o conteúdo do arquivo *Android Application Pack* (*apk*), como as permissões solicitadas do aplicativo (*manifest*), códigos compilados nativos (*lib*) ou até mesmo os arquivos fonte compilados em Java (*dex*) [Taheri et al. 2020]. Como resultado, não requer a execução da amostra do aplicativo Android que está sendo analisado, facilitando significativamente o processo de detecção.

Várias técnicas baseadas em análise estática foram propostas para caracterizar aplicativos de *malware* nos últimos anos, uma abordagem promissora realiza a classificação dos arquivos compilados em java (*dex*) em uma tarefa de classificação de imagem [Spreitzenbarth et al. 2013, dos Santos et al. 2023]. Nesse caso, o arquivo dex binário analisado é traduzido para um formato de imagem, normalmente representando cada *byte* do arquivo dex em um *pixel* de uma imagem, na sequência, a imagem gerada é classificada por uma rede neural convolucional (CNN) [Vasan et al. 2020]. Para atingir tal objetivo, um dataset de treinamento composto por um número significativo de amostras de malware e aplicativos benignos é utilizado para o treinamento de uma arquitetura CNN. O modelo construído pode então ser usado em produção para a identificação de novos aplicativos Android maliciosos.

No entanto, apesar de tais técnicas serem capazes de fornecer alta acurácia na tarefa de detecção de *malware*, os esquemas propostos geralmente não são confiáveis para configuração implementada no mundo real [Vasan et al. 2020, Horchulhack et al. 2024b]. Na prática, os autores avaliam principalmente seu modelo para a classificação entre várias famílias de *malware*, negligenciando a identificação da amostra do aplicativo *malware* em primeiro lugar [Horchulhack et al. 2024a]. Como resultado, não existem garantias de que as taxas de precisão relatadas serão alcançadas em configurações do mundo real, quando o modelo CNN construído for utilizado efetivamente para a identificação das amostras de aplicativos de *malware*. Além disso, como a detecção de *malware* baseada em imagens ainda está em seus primórdios, os *datasets* utilizados para avaliação são muitas vezes irrealistas, na maioria das vezes coletados de uma única fonte, portanto, impondo problemas de generalização.

Diante disso, este artigo propõe um novo modelo CNN hierárquico e confiável baseado em imagens para a classificação de *malwares* Android, implementado em duas etapas. Primeiro, a classificação de *malware* Android é realizada por meio de uma CNN baseada em imagem em uma configuração de classificação local estruturada hierarquicamente. Assim, os aplicativos analisados são primeiro classificados como amostras benignas ou malware em um nó pai, enquanto a família dos aplicativos classificados maliciosos é identificada em um nó filho por um modelo CNN especializado. Em segundo lugar, para melhorar a confiabilidade da classificação, apenas as classificações altamente confiáveis

realizadas pelo nó CNN pai são passadas para o nó filho, em uma classificação com abordagem de opção de rejeição. O *insight* dessa proposta é que apenas amostras de *malware* altamente confiáveis devem ter sua família identificada, mantendo assim a acurácia do sistema.

Em resumo, entre as principais contribuições desse trabalho destacam-se:

- Um novo *dataset* de *malware* de aplicativos Android composto por mais de 29 mil amostras de aplicativos benignos e de malware, divididos em 29 famílias;
- Uma avaliação das abordagens atuais CNN baseadas em imagens para classificação de *malware* Android, mostrando suas limitações e falta de confiabilidade em fornecer alta acurácia quando um *dataset* mais desafiador é utilizado;
- Um novo modelo CNN hierárquico e confiável baseado em imagem para detecção de *malware* Android capaz de melhorar a acurácia da detecção quando comparado aos trabalhos do estado da arte.

O restante deste artigo está organizado da seguinte forma. A Seção II descreve a aplicação de técnicas de ML na classificação de malware Android. A Seção III descreve os trabalhos relacionados. A Seção IV apresenta nossa proposta. A Seção IV avalia o desempenho das técnicas tradicionais em comparação com a nossa proposta sob o novo conjunto de dados. A Seção VI fornece a conclusão deste trabalho.

2. Preliminares

As redes neurais convolucionais (CNN) têm sido aplicadas com sucesso em várias áreas, normalmente para detecção de objetos e classificação de imagens [Katta and Viegas 2023, Shrestha et al. 2023]. No entanto, a classificação de aplicativos de *malware* para Android ainda está no princípio, os autores costumam aplicá-la para classificação do arquivo de origem Java compilado, ou seja, *dex*. Para atingir tal objetivo, o arquivo *dex* da amostra do aplicativo Android analisado é representado como uma imagem; na prática, os *pixels* da imagem são convertidos em uma representação direta dos valores de *byte* do arquivo *dex*. Consequentemente, como o tamanho do arquivo *dex* de cada aplicativo pode variar significativamente, o tamanho da imagem de saída também tende a variar, portanto, um redimensionamento de imagem é frequentemente aplicado antes de usá-lo como entrada no modelo CNN.

Para implementar tal processo, os autores contam com quatro módulos sequenciais. Primeiro, o módulo de aquisição de Dados, este módulo extrai o arquivo *dex* do arquivo *apk* do aplicativo Android analisado. Em segundo lugar, o módulo Construtor da Imagem que converte o conteúdo binário do arquivo dex em uma imagem, geralmente representando cada *byte* como um *pixel* de imagem, já redimensionando a imagem de saída para uma dimensão predefinida. Terceiro, a imagem construída é classificada no módulo de Classificação, que aplica um modelo CNN previamente treinado. Finalmente, as amostras classificadas como *malware* são sinalizadas no módulo de Alerta.

Nos últimos anos, devido aos resultados promissores relatados em várias outras áreas, inúmeros trabalhos exploram a abordagem CNN baseados em imagens para classificação de *malware* Android [Qiu et al. 2020]. Em geral, as abordagens propostas classificam amostras de aplicativos analisados de acordo com sua família de *malware*, porém acabam negligenciando a identificação de amostras de *malware* na análise inicial

[Vasan et al. 2020]. Consequentemente, mesmo que sejam capazes de fornecer esquemas baseados em CNN altamente precisos, eles só podem ser usados para a classificação de amostras de aplicativos que já são conhecidas como *malware*, deixando o desempenho de detecção entre *malware* e benigno ainda em aberto.

O procedimento de treinamento das abordagens baseadas em CNN requer grandes quantidades de dados para o treinamento, onde é necessário fornecer amostras diversas e realistas. Surpreendentemente, as abordagens propostas muitas vezes fazem uso de um único *dataset*, que é coletado por meio de uma única fonte de dados, tornando os resultados obtidos irrealistas. Isso ocorre porque, em configurações do mundo real, a abordagem de detecção construída deve ser capaz de classificar as amostras de aplicativos analisadas, independentemente da fonte de dados do aplicativo. Portanto, mesmo que o esquema proposto leve em conta a classificação entre *malware* e amostras benignas, as limitações do *dataset* subjacente refletem nos resultados obtidos, tornam-se pouco confiáveis.

3. Trabalhos Relacionados

A detecção de *malware* em aplicativos Android tem sido um tema amplamente explorado na literatura nos últimos anos [Geremias et al. 2022, Santos et al. 2023]. Em geral, as abordagens baseadas em análise estática propostas realizam a classificação como uma tarefa de reconhecimento de padrões, normalmente por meio de algoritmos de aprendizado de máquina (ML). Por exemplo, [Li et al. 2018] propõe um modelo baseado em ML para a identificação de aplicativos Android de *malware* de acordo com as permissões solicitadas. Seu modelo é capaz de fornecer alta acurácia em um único *dataset*, porém depende das permissões de acesso dos aplicativos e ignora a identificação de famílias de *malware*. Z. Ma et ai. [Ma et al. 2019] aplica algoritmos de ML de acordo com o grafo de fluxo de controle do código-fonte do aplicativo avaliado. O esquema proposto é capaz de fornecer alta acurácia para a detecção em uma configuração de duas classes, contudo, negligencia a identificação de famílias de *malware*. Em contraste, S. Xue et al. [Xue et al. 2018] também faz uso do grafo de fluxo de controle do código-fonte do aplicativo, mas, para a identificação da família de *malware*. O esquema proposto, baseia-se em modelos de aprendizado profundo, sendo capaz de fornecer altas taxas de acurácia, no entanto, negligencia a identificação de aplicativos de *malware* em primeiro lugar.

Nos últimos anos, devido às altas taxas de acurácia relatadas, vários trabalhos têm recorrido a abordagens baseadas em imagens para detecção de *malware* Android com arquiteturas CNN [Qiu et al. 2020]. Por exemplo, J. Singh et al. [Singh et al. 2021] realiza a conversão de vários arquivos *apk* em uma imagem em escala de cinza e aplica uma técnica baseada em CNN combinada com um método de fusão para a classificação de famílias de *malware* em aplicativos. O modelo proposto foi capaz de fornecer altas taxas de acurácia, no entanto, depende de um único *dataset* e negligencia a detecção de aplicativos de *malware* em primeiro lugar. Outro modelo baseado em CNN foi proposto por D. Vasan et al. [Vasan et al. 2020] que converte o arquivo *dex* em um formato de imagem colorida. Os autores são capazes de melhorar a acurácia por meio de sua arquitetura CNN proposta, no entanto, negligenciam a confiabilidade na classificação e a hierarquia dos aplicativos Android. Da mesma forma, T.H. Huang et al. [Hsien-De Huang and Kao 2018] propõe uma nova arquitetura baseada em CNN para classificação de aplicativos de *malware* de acordo com a imagem construída do arquivo

dex. O modelo proposto forneceu altas taxas de acurácia para a classificação de aplicativos *malware* em Android, porém negligencia a detecção das famílias de *malware*.

4. Modelo de Detecção de Malware Android Hierárquico e Confiável

Para enfrentar os desafios mencionados, apresentamos uma nova detecção hierárquica confiável de *malware* para Android por meio de CNN. O *insight* da proposta é que a acurácia da classificação de *malware* pode ser aperfeiçoada por meio de uma abordagem de classificação utilizando rejeição, onde a família de *malware* pode ser identificada utilizando uma configuração de classificação hierárquica. O modelo proposto é mostrado na Figura 1, composto por duas etapas principais, a saber: Classificação de Aplicativos Android e Classificação Confiável de Família de *Malware*.

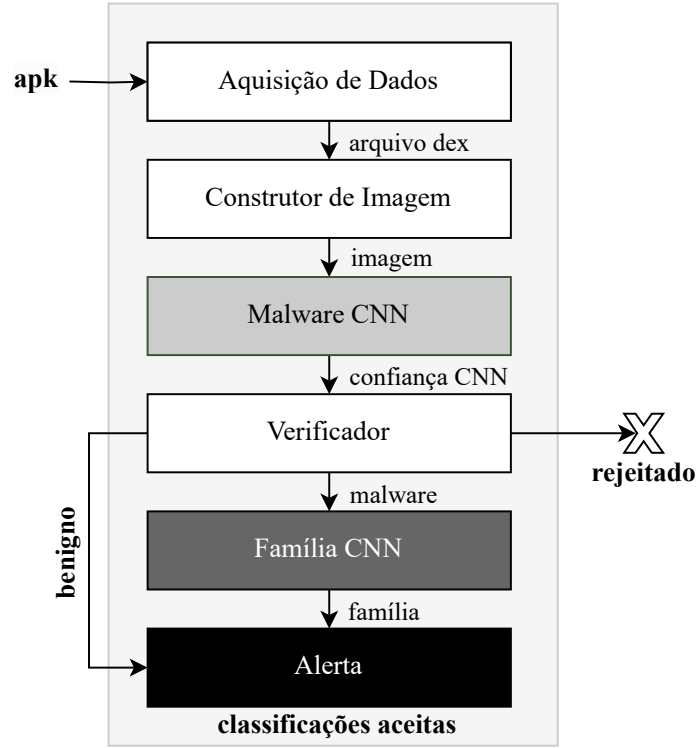


Figura 1. Proposta de detecção hierárquica confiável de *malware* Android utilizando CNN.

A proposta considera uma abordagem de classificação CNN baseado em imagem para identificar padrões de *malware* em aplicativos Android, primeiro visando a detecção de amostras de *malware* e sua família, para analisar se o *apk* deve ser classificado como malicioso. O procedimento de classificação inicia com um arquivo *apk* Android sendo fornecido como entrada para classificação. O arquivo *dex* correspondente ao *apk* é extraído e representado em um formato de imagem. Várias técnicas podem ser usadas para cumprir tal tarefa, como as usadas em nosso *dataset* IAMD (consulte a Seção 5.2). A imagem construída é classificada como amostra benigna ou *malware* pelo módulo CNN (Estrutura Hierárquica, Fig. 1), gerando um valor de confiança para classificação. O valor de confiança é usado por nossa abordagem em uma classificação utilizando rejeição permitindo melhorar a acurácia na detecção. O *insight* desta proposta é utilizar apenas

classificações altamente confiáveis, fornecendo uma maior probabilidade de serem classificadas corretamente, assim, apenas instâncias altamente confiáveis são aceitas por nosso modelo, mantendo a confiabilidade do sistema. As classificações rejeitadas são descartadas pelo nosso modelo. Em contraste, aplicativos classificados de *malware* aceitos são classificados por um modelo CNN secundário (Estrutura Hierárquica, Fig. 1), para a identificação adequada da família de *malware*. As próximas subseções descrevem melhor o esquema de classificação proposto e processo de verificação.

4.1. Classificação de aplicativos Android

Em geral, as propostas de *malware* Android baseadas em imagens visam a classificação de amostras de *malware* entre uma variedade de famílias. Portanto, a entrada dos esquemas propostos é considerada *malware*. No entanto, em um cenário em produção no mundo real, os aplicativos Android devem primeiro ser identificado como *malware* ou benigno, só então, se um determinado *malware* for sinalizado, sua família deverá ser identificada.

Diante disso, o modelo aqui proposto realiza a classificação de aplicativos Android considerando uma forma de classificação local estruturada hierarquicamente. A lógica desse esquema é que os aplicativos Android podem ser classificados em um nó pai como benigno ou *malware* e, somente então, a família de *malware* deverá ser identificada em um nó filho. Para atingir tal objetivo, nosso modelo proposto dispõe de duas CNNs, que realizam a classificação de forma hierárquica (Estrutura Hierárquica, Fig. 1). Inicialmente, a entrada do esquema é classificada como benigna ou *malware* por um modelo CNN de duas classes (*Malware* CNN, Fig. 1). Na sequência, aplicativos classificados como *malware* com alta confiança em sua classificação, são posteriomente classificados de acordo com sua família (Família CNN, Fig. 1).

Como resultado, nosso esquema proposto é capaz de identificar corretamente os *malwares* e suas respectivas famílias. Isso ocorre porque o esquema de classificação hierárquica primeiro identifica aplicativos de *malware*, antes de utilizá-los para identificação da família de *malware*. No entanto, para fornecer tal identificação, nosso modelo proposto deve garantir a confiabilidade na tarefa de classificação.

4.2. Classificação Confiável de Famílias de malware

A identificação de aplicativos de *malware* Android é uma tarefa desafiadora, em que os esquemas de detecção estão sujeitos a uma alta taxa de falsos positivos (ver Tabela 1). Portanto, os esquemas propostos devem garantir que apenas classificações corretas sejam utilizadas para acionar alertas de *malware*, caso contrário, o usuário pode ignorar alertas adicionais. Para fornecer tal nível de confiabilidade, nosso modelo lida com a classificação hierárquica acima mencionada em um módulo verificador. O objetivo do módulo é garantir que apenas classificações altamente confiáveis sejam usadas para alertar os usuários, melhorando assim a acurácia da classificação do sistema. Portanto, a classificação é realizada por meio de uma classificação com rejeição, em que classificações de baixa confiança não são utilizadas para acionar alertas.

Conforme mostrado na Figura 1, o procedimento de classificação começa com um arquivo *apk* de aplicativo Android a ser classificado. O módulo de Aquisição de Dados extrai o arquivo *dex* para análise posterior. O arquivo *dex* obtido é usado como entrada por um módulo Construtor de Imagem, que constroi uma imagem correspondente.

Várias abordagens podem ser usadas para cumprir essa tarefa, tal como a técnica usada para construir nosso *dataset* IAMD (consulte a Seção 5.2), que representa cada *byte* do arquivo *dex* como um *pixel* de imagem. Em seguida, a imagem construída é classificada pelo módulo *Malware* CNN, como *malware* ou benigno, e gera um valor de confiança de classificação correspondente. O módulo Verificador avalia o valor de confiança da classificação para garantir que o nível desejado do limiar de confiança de classificação seja atendido. Classificações de baixa confiança são rejeitadas pelo nosso modelo. Por outro lado, classificações altamente confiáveis são usadas para alertar os usuários, na situação em que os aplicativos *malware* forem classificados como altamente confiáveis, estes também são usados como entrada por uma segunda CNN, seguindo nossa estrutura hierárquica (Família CNN, Fig. 1). Por fim, as classificações aceitas são enviadas ao módulo de Alerta, relatando a família de *malware* encontrada ou atestando os aplicativos como benignos.

5. Avaliação

A detecção de *malware* Android baseada em CNN como uma tarefa de classificação de imagens ainda está em seus primórdios. Nesta seção avaliamos o desempenho das abordagens amplamente utilizadas na literatura. Mais especificamente, primeiro introduzimos nosso novo *dataset* com características do mundo real, contendo amostras de *malware* Android bem representativas, em seguida, avaliamos o desempenho das abordagens CNN sobre este *dataset*.

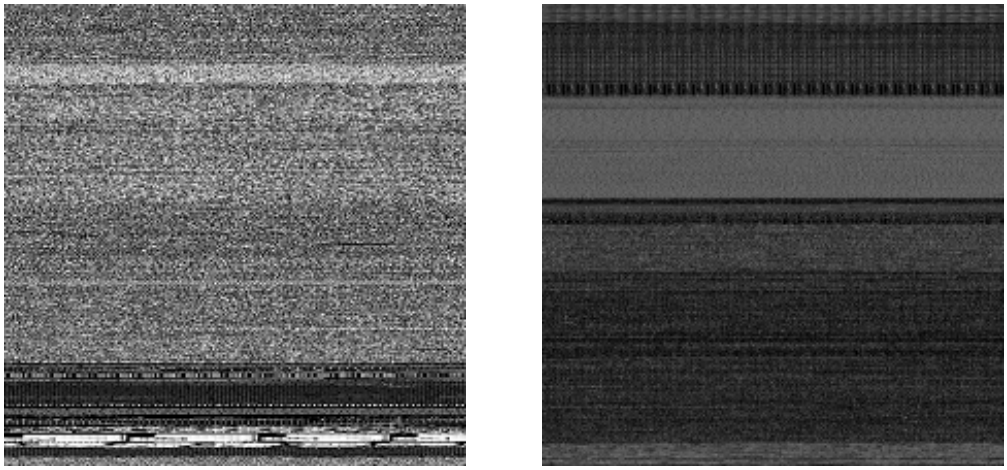
A avaliação visa responder a quatro questões principais de pesquisa: (RQ1) Como as arquiteturas CNN tradicionais classificam amostras de *malware* e benignos entre aplicativos Android? (RQ2) Como as arquiteturas CNN tradicionais realizam a classificação famílias de *malware*?(RQ3) Como a abordagem da avaliação na classificação melhora o modelo de classificação de *malware*? (RQ4) Como o modelo proposto se comporta na classificação de famílias de *malware*?

5.1. Dataset de Imagem de Malware de Android Realista

Os *datasets* atuais utilizados na literatura para classificação de amostras de aplicativos Android não fornecem características realistas em um ambiente de produção no mundo real. Em geral, autores recorrem a um único *dataset*, ficando propenso a trabalhar com um conjunto limitado de amostras, problema inerente a sua fonte de dados. Idealmente, os *datasets* utilizados devem ser compostos por um número representativo de *malware* e amostras benignas, coletados de uma variedade de fontes ao longo do tempo.

5.2. Android Malware Detection as a Image Classification Task

Para resolver tal limitação, apresentamos o *dataset* o IAMD (*Image Android Malware Dataset*). Um *dataset* composto por amostras benignas e de *malware* de dois *datasets* publicamente disponíveis, nomeadamente CICMalDroid [Mahdavifar et al. 2020], e MaliImagens [Nataraj et al. 2011]. O primeiro, CICMalDroid, dispõe de 17.341 amostras benignas e de *malware*, divididas em 4 famílias de *malware*. O MaliImages, é concebido de 9.458 *malwares* amostras, divididas em 25 famílias. Como resultado, nosso conjunto de dados, IAMD, é composto por 26.799 amostras de aplicativos Android, com 6.815 amostras benignas e 19.984 amostras de *malware* divididas em 29 famílias. Os arquivos



(a) Amostra benigna **(b)** Amostra *malware*

Figura 2. Amostras de imagens criadas a partir dos arquivos *dex* de aplicativos Android disponível no *dataset* IAMD, resolução 224x224.

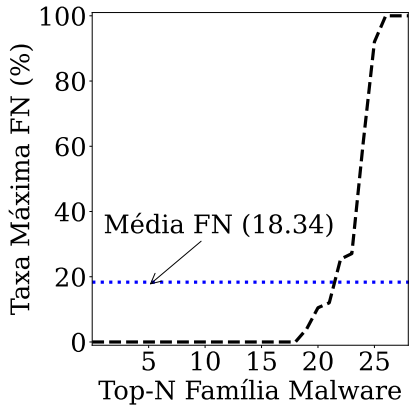


Figura 3. Top N da taxa FN do Resnet para a classificação da família de *malware* no *dataset* IAMD de 29 classes.

apk do Android coletados são representados como uma imagem em nosso *dataset*, fazendo uso da mesma metodologia utilizados na literatura (ver Seção 2). Assim, o arquivo *dex* é extraído do arquivo do aplicativo *apk* e convertido em escala de cinza em um arquivo de imagem no formato PNG. As imagens foram geradas através o *Python Image Library* (*PIL*) API v.8.4.0. as dimensões das imagens construídas varia de acordo com o tamanho do arquivo *dex*, portanto, todas as imagens foram redimensionadas para uma dimensão de 224x224 antes de fornecer como entrada para arquiteturas CNN. No total, o *dataset* IAMD é composto por mais de 13,4 GB de imagens. A Figura 2 mostra amostras de imagens do dataset IAMD.

5.3. Detecção de Malware Android Utilizando Classificação de Imagem

Uma arquitetura CNN amplamente utilizadas foi avaliada, a saber, Restnet50. A arquitetura avaliada foi executada por 1.000 épocas, e o valor de *learning rate* definido empiricamente de acordo com a função *loss* resultante e um peso do parâmetro *momentum* fixado em 0.9. Ainda, uma subamostragem aleatória sem reposição foi realizada no conjunto

Tabela 1. Acurácia das arquitetura REsnet50 no *dataset* IAMD de duas classes (benigno vs *malware*).

Arquitetura CNN	TP (%)	TN (%)	F1	AUC
Resnet50	92,95	92,77	0,929	0,97

de treinamento para balancear a ocorrência entre as classes. Para fins de treinamento, o *dataset* IAMD foi dividido em conjuntos de dados para treinamento, validação e teste, compreendendo 40%, 30% e 30% do *dataset* original, respectivamente. As CNNs foram implementadas utilizando as API´s keras v.2.4.0 e tensorflow v.2.4.1. O desempenho da classificação foi medido de acordo com suas métricas True-Positive (TP), True-Negative (TN), *F1-Score* e AUC. O TP denota a proporção de instâncias de *malware* classificadas corretamente como *malware*, enquanto o TN denota a proporção de instâncias benignas classificadas corretamente como benignas. A *F1-Score* foi calculada como a média harmônica dos valores de precisão e *recall*, considerando *malware* como amostras positivas e aplicativos benignos como amostras negativas.

O primeiro experimento visa responder RQ1 e avalia o desempenho das arquiteturas CNN na classificação de amostras de aplicativos Android como benignas ou *malware*, em uma configuração de duas classes. Para atingir tal objetivo, as arquiteturas selecionadas são treinadas com o *dataset* reservado para o treinamento, construído aleatoriamente com 40% das amostras do IAMD, independentemente da família de *malware* subjacente, e também utilizando amostras benignas.

A Tabela I mostra o desempenho de classificação das arquiteturas CNN selecionadas ao classificar amostras IAMD como *malware* ou benignas. Surpreendentemente, a arquitetura avaliada não foi capaz de fornecer acurácia de classificação significativamente altas. A CNN Resnet50, foi capaz de fornecer uma AUC de 0,97, apresentou uma taxa de TP de apenas 92,95% e uma taxa de TN de apenas 92,77%. Assim, o experimento mostra que as técnicas tradicionais baseadas em CNN são incapazes de identificar de forma confiável amostras de aplicativos Android maliciosos.

O segundo experimento visa responder ao RQ2 e avalia o desempenho de classificação das arquiteturas CNN selecionadas quando aplicadas para a identificação da família de *malware*, abordagem comumente realizada na literatura. Para atingir tal objetivo, as amostras benignas são removidas do dataset IAMD, deste modo a arquitetura CNN classifica a instância entre uma das 29 famílias de *malware*. Assim, assume que a entrada da CNN é sempre uma amostra de *malware*.

Na Figura 3 demostramos a acurácia da classificação da família de *malware* ordenada do menos preciso ao mais preciso. É possível observar que as arquiteturas selecionadas também não foram capazes de fornecer alta acurácia para a identificação da família de *malware*. Por exemplo, em média, o Resnet apresentou uma taxa de FN de apenas 18.34%. Assim, podemos observar que a arquitetura CNN não foi capaz de fornecer confiabilidade na tarefa de classificação de famílias de *malware* Android.

5.4. Construção dos Modelos

O modelo proposto (Fig. 1) foi implementado e avaliado em cima de nosso *dataset*, ou seja, IAMD (ver Seção 5.2). Da mesma forma, a arquitetura CNN utilizada também foi

Tabela 2. Desempenho da acurácia da abordagem proposta para classificação de amostras benignas e de *malware* (*Malware* CNN), de acordo com a taxa de rejeição (Fig. 4b).

CNN	TP (%)	TN (%)	F1	Rejection (%)
Resnet50	92.95	92.77	0.929	0.00
	96.25	98.27	0.972	5.00
	96.54	98.30	0.970	10.00
	97.00	98.33	0.977	15.00

avaliada em nossa proposta, a saber, a arquitetura Resnet50. A arquitetura avaliada foi executada por 1.000 épocas, e o *learning rate* definido empiricamente de acordo com a função perda (*loss*) e o peso do parâmetro *momentum* fixado em 0.9. Uma técnica de subamostragem aleatória sem reposição foi utilizada no procedimento de treinamento para balancear a ocorrência entre as classes. Para fins de treinamento, o *dataset* IAMD foi dividido em conjuntos de dados de treinamento, validação e teste, compreendendo 40%, 30% e 30% do conjunto de dados original, respectivamente.

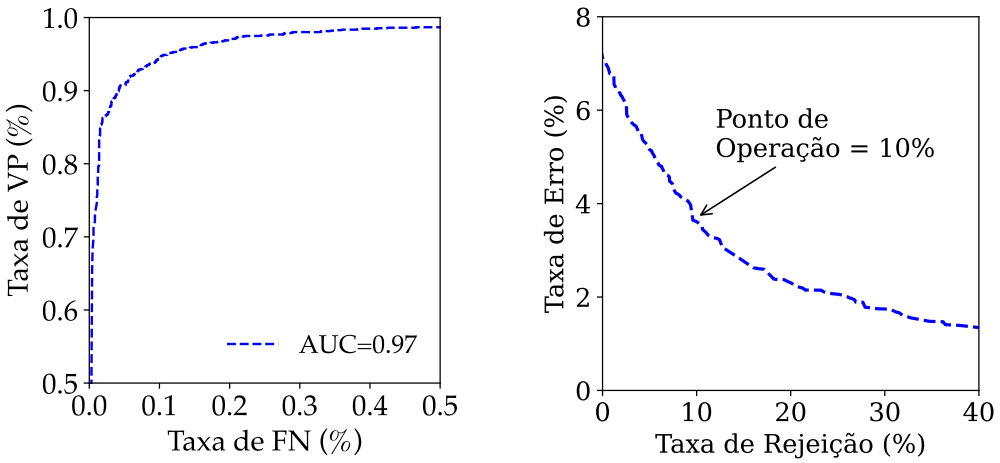
5.5. Classificação Hierárquica Confiável

5.6. Detecção de Malware Android Utilizando Classificação de Imagem

O experimento seguinte visa responder a RQ3 e avalia o desempenho da classificação do modelo proposto enquanto faz uso do módulo Verificador para a avaliação das classificações realizadas em um ambiente de duas classes (*Malware* CNN, Fig. 1). A Figura 4a mostra a curva ROC das arquitetura CNN selecionada. Em seguida, fazendo uso da arquitetura Resnet40, avaliamos o módulo Verificador para avaliação da confiação das classificações realizadas. Os limiares de classificação foram definidos por meio da abordagem *Class-Related-Threshold* (*CRT*), que busca um limiar ótimo de aceitação para cada classe.

A Figura 4b mostra o *tradeoff* entre o erro *versus* a rejeição, nas duas arquiteturas CNN selecionadas para a classificação considerando uma configuração de duas classes (*Malware* CNN, Fig. 1). É possível notar a relação entre a taxa de erro e a taxa de rejeição, portanto, uma maior rejeição dos aplicativos Android avaliados é capaz de diminuir o taxa de erro médio em nossa proposta. A Tabela 2 mostra o desempenho da acurácia do nosso modelo lidando com o módulo Verificador de acordo com a taxa de rejeição definida. É importante notar que o ponto de operação deve ser escolhido de acordo com o usuário critério. Por exemplo, rejeitando apenas 10,0% das instâncias, nosso modelo proposto é capaz de melhorar a taxa de TP em 3,6% e 1,33% para Resnet50 e InceptionV3, respectivamente. Como resultado, o esquema proposto que avalia os valores de confiança na classificação por meio do módulo Verificador é capaz de melhorar a confiabilidade do sistema na detecção de *malware*.

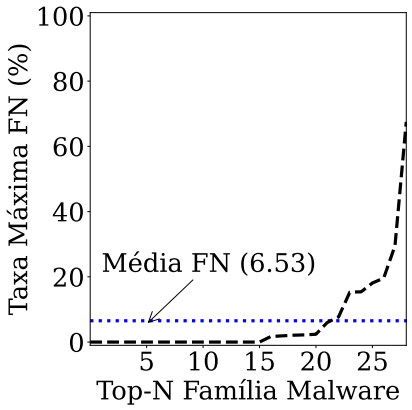
O segundo experimento visa responder a RQ4 e avalia nosso modelo proposto para a classificação de famílias de *malware* aceitas pelo nosso módulo Verificador. Para alcançar tal objetivo, avaliamos a acurácia da classificação para a identificação entre as 29 famílias no *dataset* IAMD, usando apenas os aplicativos aceitos de acordo com o ponto de operação definido na rejeição(Rejeição de 10%, Tabela 2). A Figura 5 mostra



(a) Curva ROC sem módulo verificador

(b) Erro *vs.* rejeição com módulo verificador

Figura 4. Desempenho de acurácia do esquema proposto e *tradeoff* entre erro e rejeição na classificação de amostras benignas e *malware*.



(a) Resnet50

Figura 5. Top N da taxa FN da arquitetura CNN Resnet50 no módulo Verificador (Ponto de operação, Fig. 4b) para a classificação da família de *malware*.

a acurácia da nossa abordagem na identificação das familias de *malware* (Family CNN, Fig 1) conforme ordenado do menos preciso ao mais preciso em nossa proposta. Nesse caso, nosso esquema proposto é capaz de melhorar significativamente a classificação das instâncias aceitas para classificação de famílias de *malware*. Mais especificamente, ao rejeitar apenas 10% dos aplicativos avaliados, nosso modelo melhora o taxa média de TP médio na identificação de família de *malware* em 12,75% para o Resnet50 (Fig. 3 *vs* 5).

6. Conclusão

A detecção de *malware* Android como uma tarefa de classificação de imagens por meio de CNNs ainda está em seus primórdios. Neste artigo, propusemos uma nova abordagem para detecção de *malware* Android confiável e hierárquica por meio do uso de arquiteturas CNN na detecção de aplicativos maliciosos, bem como de sua família. Nosso esquema é capaz de melhorar a detecção de aplicativos de *malware* rejeitando apenas um pequeno

subconjunto de amostras de aplicativos. Além disso, é capaz de melhorar a taxa média de detecção das famílias de *malware* aceitas quando comparadas às técnicas tradicionais.

Agradecimentos

Os autores agradecem ao CNPq pelo apoio financeiro parcial ao projeto, processos 304990/2021-3, 302937/2023-4, e 407879/2023-4.

Referências

dos Santos, R. R., Viegas, E. K., Santin, A. O., and Tedeschi, P. (2023). Federated learning for reliable model updates in network-based intrusion detection. *Computers amp; Security*, 133:103413.

Geremias, J., Viegas, E. K., Santin, A. O., Britto, A., and Horchulhack, P. (2022). Towards multi-view android malware detection through image-based deep learning. In *2022 International Wireless Communications and Mobile Computing (IWCMC)*. IEEE.

Horchulhack, P., Viegas, E. K., Santin, A. O., Ramos, F. V., and Tedeschi, P. (2024a). Detection of quality of service degradation on multi-tenant containerized services. *Journal of Network and Computer Applications*, 224:103839.

Horchulhack, P., Viegas, E. K., Santin, A. O., and Simioni, J. A. (2024b). Network-based intrusion detection through image-based cnn and transfer learning. In *2024 International Wireless Communications and Mobile Computing (IWCMC)*. IEEE.

Hsien-De Huang, T. and Kao, H.-Y. (2018). R2-d2: Color-inspired convolutional neural network (cnn)-based android malware detections. In *2018 IEEE international conference on big data (big data)*, pages 2633–2642. IEEE.

inMobi, T. (2021). Understanding android users worldwide.

Katta, S. S. and Viegas, E. K. (2023). Towards a reliable and lightweight onboard fault detection in autonomous unmanned aerial vehicles. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE.

Kotzias, P., Caballero, J., and Bilge, L. (2021). How did that get in my phone? unwanted app distribution on android devices. In *2021 IEEE Symposium on Security and Privacy (SP)*, page 53–69. IEEE.

Li, J., Sun, L., Yan, Q., Li, Z., Srisa-An, W., and Ye, H. (2018). Significant permission identification for machine-learning-based android malware detection. *IEEE Transactions on Industrial Informatics*, 14(7):3216–3225.

Ma, Z., Ge, H., Liu, Y., Zhao, M., and Ma, J. (2019). A combination method for android malware detection based on control flow graphs and machine learning algorithms. *IEEE access*, 7:21235–21245.

Mahdavifar, S., Kadir, A. F. A., Fatemi, R., Alhadidi, D., and Ghorbani, A. A. (2020). Dynamic android malware category classification using semi-supervised deep learning. In *2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing*, pages 515–522. IEEE.

Nataraj, L., Karthikeyan, S., Jacob, G., and Manjunath, B. S. (2011). Malware images: visualization and automatic classification. In *Proceedings of the 8th international symposium on visualization for cyber security*, pages 1–7.

Qiu, J., Zhang, J., Luo, W., Pan, L., Nepal, S., and Xiang, Y. (2020). A survey of android malware detection with deep neural models. *ACM Computing Surveys (CSUR)*, 53(6):1–36.

Santos, R. R. d., Viegas, E. K., Santin, A. O., and Cogo, V. V. (2023). Reinforcement learning for intrusion detection: More model longness and fewer updates. *IEEE Transactions on Network and Service Management*, 20(2):2040–2055.

Shrestha, S., Pathak, S., and Viegas, E. K. (2023). Towards a robust adversarial patch attack against unmanned aerial vehicles object detection. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE.

Singh, J., Thakur, D., Gera, T., Shah, B., Abuhmed, T., and Ali, F. (2021). Classification and analysis of android malware images using feature fusion technique. *IEEE Access*, 9:90102–90117.

Spreitzenbarth, M., Freiling, F., Echtler, F., Schreck, T., and Hoffmann, J. (2013). Mobile-sandbox: having a deeper look into android applications. In *Proceedings of the 28th annual ACM symposium on applied computing*, pages 1808–1815.

Taheri, R., Ghahramani, M., Javidan, R., Shojafar, M., Pooranian, Z., and Conti, M. (2020). Similarity-based android malware detection using hamming distance of static binary features. *Future Generation Computer Systems*, 105:230–247.

Vasan, D., Alazab, M., Wassan, S., Naeem, H., Safaei, B., and Zheng, Q. (2020). Imcfn: Image-based malware classification using fine-tuned convolutional neural network architecture. *Computer Networks*, 171:107138.

Vidas, T. and Christin, N. (2014). Evading android runtime analysis via sandbox detection. In *Proceedings of the 9th ACM symposium on Information, computer and communications security*, pages 447–458.

Xue, S., Zhang, L., Li, A., Li, X.-Y., Ruan, C., and Huang, W. (2018). Appdna: App behavior profiling via graph-based deep learning. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pages 1475–1483. IEEE.