Um Modelo Baseado em Redes Neurais Profundas e Redes Neurais Recorrentes para Análise Sequencial de Binários Android

Jhonatan Geremias[1], **Altair O. Santin**[1] **e Eduardo K. Viegas**[1]

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

{jgeremias, santin, eduardo.viegas}@ppgia.pucpr.br

Resumo. *Este trabalho propõe um framework que particiona o código binário de aplicações Android em sequências de imagens de tamanho fixo, preservando sua estrutura espacial original. O modelo combina uma Rede Neural Profunda (DNN) para extração de características espaciais com uma Rede Neural Recorrente (RNN), responsável por capturar dependências temporais entre sequências. A abordagem utiliza uma estratégia de classificação many-to-one, permitindo lidar com aplicações de tamanhos variáveis sem perda significativa de informação. O método foi avaliado em um conjunto de mais de 13 mil amostras do AndroZoo coletadas ao longo de quatro anos. Os resultados mostram que a proposta supera abordagens tradicionais baseadas em redimensionamento, alcançando um F1-Score de até 0,93, uma melhoria de até 0,03 no AUC e maior capacidade de generalização entre diferentes famílias de malware.*

Abstract. *This work proposes a framework that partitions the binary code of Android applications into fixed-size image sequences, preserving their original spatial structure. The model combines a Deep Neural Network (DNN) for spatial feature extraction with a Recurrent Neural Network (RNN), responsible for capturing temporal dependencies between sequences. The approach uses a many-to-one classification strategy, allowing it to handle applications of varying sizes without significant information loss. The method was evaluated on a dataset of over 13,000 AndroZoo samples collected over four years. The results show that the proposal outperforms traditional resizing-based approaches, achieving an F1-Score of up to 0.93, an improvement of up to 0.03 in AUC, and greater generalization capacity across different malware families.*

1. Introdução

O Android consolidou-se como o sistema operacional móvel dominante no cenário global, alcançando aproximadamente 70% de participação de mercado no início de 2026 [StatCounter 2026]. Essa ampla adoção, combinada com o modelo de desenvolvimento open source da plataforma, tornou o ecossistema Android um dos principais alvos para a disseminação de código malicioso e de atividades cibercriminosas. Estudos recentes apontam que cerca de 95% dos incidentes envolvendo malware em dispositivos móveis estão relacionados ao Android, com milhões de novas amostras identificadas todos os anos [Anton Kivva 2026]. Nesse contexto, diversas técnicas de detecção de malware

têm sido investigadas, sendo tradicionalmente agrupadas em duas categorias principais: análise estática e análise dinâmica [Shu et al. 2023]. Enquanto a análise estática busca identificar características maliciosas a partir da inspeção do código-fonte, do arquivo `AndroidManifest.xml` ou do bytecode da aplicação, sem a execução da aplicação, a análise dinâmica avalia o comportamento do aplicativo durante sua execução em ambientes isolados e controlados, como sandboxes. Embora os métodos dinâmicos possibilitem uma análise comportamental mais detalhada, as técnicas estáticas continuam amplamente utilizadas na literatura devido ao menor custo computacional associado e à simplicidade de implantação em larga escala [Rodrigues et al. 2025].

Com o intuito de aprimorar a capacidade de identificação de aplicações maliciosas, estudos recentes têm explorado o uso de Deep Neural Networks (DNNs) aplicadas à análise estática de malware, reformulando o problema como uma tarefa de classificação baseada em imagens [Maniriho et al. 2024]. Nessas abordagens, diferentes elementos da aplicação são convertidos em representações visuais que podem ser processadas por modelos de aprendizado profundo [Espindola et al. 2026b]. Uma estratégia amplamente adotada consiste em representar o arquivo `classes.dex` como uma imagem em escala de cinza [Yadav et al. 2022]. Para isso, o fluxo de bytes do arquivo binário é reorganizado em uma matriz bidimensional, na qual cada byte, com valores entre 0 e 255, é associado à intensidade de um pixel correspondente. Essa representação permite que arquiteturas baseadas em Convolutional Neural Networks (CNNs) aprendam padrões espaciais relacionados à organização binária, facilitando a identificação de características discriminativas e de possíveis assinaturas associadas a códigos maliciosos. Apesar dos resultados promissores, essa estratégia apresenta uma limitação importante: a resolução das imagens produzidas depende diretamente do tamanho do arquivo `classes.dex`. Como consequência, aplicações de diferentes tamanhos resultam em representações visuais com dimensões distintas, o que aumenta a variabilidade estrutural dos dados de entrada e dificulta o processo de aprendizado [Lan et al. 2023].

Para adequar as representações visuais aos requisitos de entrada fixa das CNNs, as imagens geradas são geralmente submetidas a processos de redimensionamento para dimensões predefinidas, como 224×224 pixels. Entretanto, essa etapa pode comprometer a integridade das informações extraídas do binário, uma vez que características relevantes para a identificação de padrões maliciosos podem ser removidas ou distorcidas durante a transformação da imagem [Sharma and Rattan 2025]. Além disso, aplicações Android de maior complexidade podem gerar milhares de imagens com resolução padronizada, dificultando a manutenção das relações estruturais do arquivo original e a representação do contexto global da aplicação. Outro fator limitante está relacionado à disponibilidade restrita de datasets públicos que preservem a correspondência original entre os bytes binários e suas representações visuais [Maniriho et al. 2024]. A maioria dos conjuntos de dados existentes fornece imagens previamente processadas e redimensionadas, visando atender às limitações de entrada impostas por arquiteturas de deep learning. Como resultado, ainda há uma carência de benchmarks capazes de preservar o mapeamento byte-pixel original dos arquivos Android Application Packages (APKs), o que restringe a avaliação e o desenvolvimento de modelos mais robustos, com maior capacidade de generalização em cenários reais de detecção de malware.

Contribuição. Diante dessas limitações, este artigo apresenta um novo modelo de detecção

de malware Android baseado em representações visuais com reconhecimento espacial, projetado para preservar as características estruturais e comportamentais dos arquivos Dalvik Executable (DEX) analisados. Inicialmente, é proposto um novo dataset para detecção de malware Android baseado em imagens, contendo 13.850 amostras benignas e maliciosas, convertidas em mais de 1.7 milhões de imagens coletadas durante um período de 4 anos. Diferentemente dos conjuntos de dados tradicionais, o dataset proposto preserva as propriedades originais dos arquivos APKs, mantendo a relação entre os bytes binários e sua representação visual, sem introduzir perdas decorrentes de etapas convencionais de redimensionamento. Além disso, é apresentada uma nova arquitetura de detecção baseada em reconhecimento espacial, que combina uma DNN, responsável pela extração de características espaciais das imagens, com uma Recurrent Neural Network (RNN), utilizada para modelar dependências temporais entre sequências de imagens derivadas do binário analisado. A principal contribuição da abordagem consiste em preservar a integridade estrutural do código binário, permitindo o processamento de aplicações Android de diferentes tamanhos sem comprometer a capacidade de generalização do modelo. Os resultados experimentais indicam que a solução proposta proporciona melhorias de até 0.03 no F1-Score e amplia a capacidade de identificar diferentes famílias de malware Android. Em resumo, a principal contribuição deste artigo é um novo modelo de detecção de malware para Android, baseado em imagens e com reconhecimento espacial, que preserva as características comportamentais do APK analisado. O esquema proposto melhora a pontuação F1 em até 0,03 e aumenta a precisão da detecção de famílias de malware.

Estrutura. O restante deste artigo está organizado da seguinte forma: A Seção 2 apresenta os fundamentos da detecção de malware para Android, enquanto a Seção 3 apresenta uma visão geral de trabalhos relacionados. A Seção 4 apresenta o esquema proposto, a Seção 5 avalia seu desempenho e a Seção 6 conclui nosso trabalho.

2. Preliminares

2.1. Detecção de Malware Android Baseada em Machine Learning

Pesquisas recentes têm direcionado esforços para abordagens de detecção de malware Android baseadas em representações visuais, principalmente devido aos resultados expressivos obtidos por arquiteturas de DNN em tarefas de classificação [Maniriho et al. 2024, de Oliveira et al. 2025]. Essa tendência está associada à capacidade desses modelos de aprender automaticamente representações discriminativas a partir dos dados de entrada, reduzindo a necessidade de processos manuais de seleção e de extração de características específicas do domínio [Horchulhack et al. 2022]. Ao converter informações binárias em representações visuais, as DNNs são capazes de explorar relações espaciais e padrões estruturais complexos presentes no código, possibilitando a identificação de características que podem não ser evidentes em abordagens tradicionais baseadas em assinaturas ou em regras heurísticas [Geremias et al. 2023].

Para realizar a detecção de malware Android baseada em imagens, as abordagens existentes geralmente convertem componentes executáveis da aplicação, como o arquivo `classes.dex`, em representações visuais por meio do mapeamento direto do fluxo de bytes em matrizes em escala de cinza ou em formato RGB [Lan et al. 2023]. Nesse processo, cada byte do arquivo binário é interpretado como a intensidade de um pixel, o que permite transformar características estruturais da aplicação em padrões espaciais passíveis

de análise por modelos de aprendizado profundo [Viegas et al. 2020]. Entretanto, para possibilitar o uso de arquiteturas convencionais de CNN, as imagens obtidas são normalmente ajustadas para dimensões de entrada fixas, como 224×224 pixels. Após essa etapa de padronização das representações visuais, os modelos são submetidos a protocolos experimentais que normalmente particionam o dataset em três conjuntos independentes [Lan et al. 2023]. O conjunto de treinamento é responsável pelo aprendizado dos parâmetros da rede neural, enquanto o conjunto de validação auxilia na seleção de hiperparâmetros e no controle do processo de treinamento, reduzindo o risco de overfitting [Espindola et al. 2026a]. Por fim, o conjunto de teste, mantido isolado durante a etapa de desenvolvimento, permite avaliar de forma imparcial o desempenho do modelo, considerando sua capacidade de detectar amostras desconhecidas de malware e de generalizar para novos cenários.

3. Trabalhos Relacionados

Nos últimos anos, pesquisas têm explorado arquiteturas de DNNs para a detecção de malware Android, obtendo resultados expressivos em tarefas de classificação. De modo geral, essas abordagens exploram a conversão de arquivos APKs em representações visuais, permitindo que modelos de aprendizado profundo identifiquem padrões estruturais associados a aplicações maliciosas. M. Yapici *et al.* [Yapici 2025] apresentam um framework baseado em DNN que emprega representações tridimensionais de aplicações Android para detecção e classificação de diferentes categorias de malware, alcançando desempenho superior em comparação com métodos convencionais. Entretanto, a abordagem proposta não investiga os impactos da perda de informação estrutural decorrente do redimensionamento das imagens para atender aos requisitos de entrada da rede. Seguindo uma estratégia semelhante, P. Yadav *et al.* [Yadav et al. 2022] utilizam a arquitetura EfficientNet-B4 para identificar malware Android a partir de representações visuais de arquivos *Dalvik Executable* (DEX). Embora o método apresente resultados competitivos, sua operação permanece condicionada ao uso de representações com dimensões fixas de entrada, não considerando a variação natural existente entre os tamanhos dos arquivos APKs.

J. Tang *et al.* [Tang et al. 2024] propõem uma abordagem que combina imagens em escala de cinza com imagens de Markov derivadas de sequências de bytecode, utilizando uma CNN para classificar as amostras. Apesar dos resultados obtidos, o método depende de representações com dimensões predefinidas na etapa de fusão de imagens, sem considerar os efeitos da perda de informação decorrente do redimensionamento de arquivos APKs de tamanhos distintos. De maneira semelhante, B. Zou *et al.* [Zou et al. 2022] apresentam uma arquitetura compacta baseada em CNN para classificação de imagens de malware, dispensando técnicas de aumento de dados e etapas de pré-treinamento. Embora o trabalho analise a influência da resolução das imagens no desempenho do modelo, a abordagem ainda emprega representações de entrada de tamanho fixo e reduzido, o que limita sua capacidade de capturar adequadamente a complexidade estrutural presente em aplicações Android reais.

Além das CNNs, modelos baseados em RNNs também têm sido investigados para explorar características espaço-temporais aplicadas à detecção de malware Android. L. Shen *et al.* [Shen et al. 2023] apresentam um framework que transforma fluxos de tráfego de rede em imagens cronológicas em escala de cinza, que são posteriormente proces-

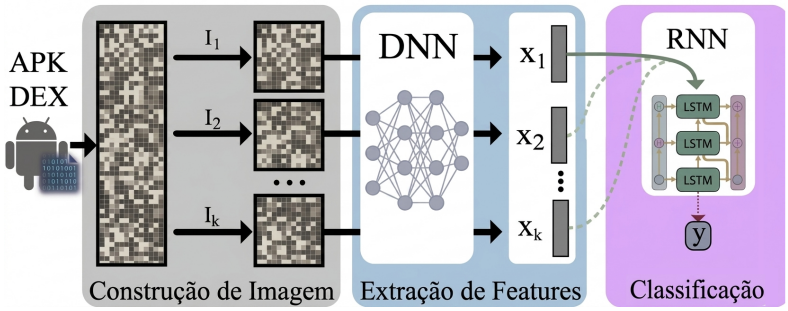


Figura 1. Visão geral do esquema proposto para detecção de malware Android. Cada APK analisado é representado por múltiplas imagens DEX, e suas características associadas são extraídas por meio de um modelo DNN previamente treinado. O conjunto resultante de vetores de características é então classificado por um modelo RNN.

sadas por uma Convolutional Long Short-Term Memory (LSTM) com mecanismo de autoatenção. Entretanto, a abordagem limita-se ao domínio do tráfego de rede e depende de representações visuais em formatos previamente definidos. M. U. Tanveer *et al.* [Usama Tanveer et al. 2025] empregam LSTM para identificar malware Android a partir da modelagem de dependências sequenciais presentes em séries temporais, incluindo informações de tráfego de rede e registros de execução do sistema. Apesar disso, o método não explora a combinação entre representações espaciais e temporais obtidas diretamente a partir de características extraídas do binário da aplicação. De forma semelhante, X. Xiao *et al.* [Xiao et al. 2017] propõem uma técnica baseada em LSTM que utiliza sequências de chamadas de sistema como atributos temporais para calcular a similaridade e classificar amostras. No entanto, essa abordagem permanece restrita à análise de informações sequenciais, sem considerar a preservação das características estruturais dos arquivos analisados.

Embora diversas pesquisas apresentem resultados expressivos ao transformar componentes de aplicações Android em representações visuais ou sequências comportamentais analisadas por CNNs e RNNs, a maioria dessas abordagens ainda depende de formatos de entrada predefinidos, o que exige etapas de redimensionamento das representações geradas. Esse processo pode introduzir perdas relevantes de informação estrutural e comprometer a capacidade de generalização dos modelos diante da diversidade de tamanhos, estruturas e características presentes em famílias reais de malware Android. Como resultado, permanece o desafio de construir métodos capazes de preservar as propriedades originais dos binários analisados, mantendo simultaneamente um desempenho adequado na detecção de novas amostras maliciosas.

4. Modelo com Reconhecimento Espacial para Detecção de Malware Android Baseada em Imagens

Para superar o problema da variabilidade de tamanho dos APKs em abordagens de detecção de malware Android baseadas em imagens e DNNs, propomos um novo modelo com capacidade de reconhecimento espacial, conforme apresentado na Figura 1. A solução proposta é organizada em três etapas principais.

Na primeira etapa, a representação visual da aplicação é construída a partir da con-

versão dos bytes presentes em cada arquivo `classes.dex` em valores de intensidade de pixel, resultando em uma imagem global em escala de cinza que representa o código executável do APK analisado (Fig. 1, *Image Building*). Posteriormente, essa representação é dividida em uma sequência de subimagens de dimensões fixas, permitindo a adaptação aos requisitos de entrada das redes neurais sem aplicar operações de redimensionamento à imagem original. Dessa forma, a estrutura binária é preservada integralmente, evitando a perda de informações causada por transformações convencionais que reduzem globalmente a resolução da imagem.

Na segunda etapa, a subimagem gerada é processada individualmente por uma DNN, que extrai características espaciais e identifica padrões locais na representação visual binária. Como resultado, obtêm-se vetores de características de alta dimensionalidade que representam diferentes regiões do código executável analisado (Fig. 1, *Feat. Extraction*).

Na etapa final, os vetores de características extraídos são submetidos a uma RNN, responsável por modelar as relações sequenciais e as dependências estruturais existentes entre as diferentes regiões do APK (Fig. 1, *Classification*). A principal contribuição dessa estratégia consiste em permitir que o modelo processe sequências com quantidade variável de vetores de características, eliminando a necessidade de padronizar previamente o tamanho das aplicações Android. Dessa forma, a abordagem preserva as propriedades globais do binário analisado e possibilita a detecção de malware em aplicações com diferentes níveis de complexidade estrutural. Como consequência, o modelo proposto amplia a capacidade de generalização da abordagem de detecção, permitindo identificar diferentes famílias de malware Android com maior eficiência.

Nas próximas subseções, são apresentados em detalhes os módulos que compõem a arquitetura proposta.

4.1. Construtor de Imagem

Os métodos atuais de detecção de malware Android têm explorado amplamente arquiteturas baseadas em DNNs, principalmente devido à capacidade desses modelos em aprender representações discriminativas e realizar a extração automática de características a partir dos dados analisados. Nesse contexto, o arquivo `classes.dex` é frequentemente convertido em uma representação visual por meio de um processo de mapeamento byte-pixel, no qual os valores de byte são transformados em intensidades de pixel, gerando imagens com dimensões diretamente relacionadas ao tamanho do arquivo original. Contudo, como as arquiteturas de DNNs geralmente exigem entradas com dimensões fixas, essas representações são tradicionalmente submetidas a etapas de redimensionamento para adequá-las ao formato esperado pelo modelo. Essa transformação pode introduzir perdas de informações estruturais relevantes e reduzir a capacidade de generalização da solução diante da diversidade de tamanhos e características presentes em diferentes famílias de malware Android. Para contornar essa limitação, a abordagem proposta mantém a representação original em binário e gera uma sequência de imagens com dimensões fixas a partir da divisão da imagem completa, preservando a estrutura do arquivo analisado independentemente de seu tamanho inicial (Fig. 1, *Construção da Imagem*).

Seja $\mathcal{D}$ um arquivo `classes.dex` extraído de um APK em análise, representado por uma sequência de bytes $B = b_1, b_2, \ldots, b_n$, onde n é o tamanho total do arquivo.

Inicialmente, a sequência de bytes é convertida em uma representação visual global em escala de cinza $\mathcal{I}$ por meio de um processo de mapeamento byte-pixel, no qual cada byte b_i é associado a um pixel p_i, cuja intensidade representa diretamente o valor armazenado nesse byte. Como o tamanho dos arquivos `classes.dex` pode variar entre diferentes aplicações, a imagem resultante $\mathcal{I}$ é posteriormente dividida em uma sequência de k subimagens $I_1, I_2, \ldots, I_k$, todas com dimensões fixas $w \times h$. Esse particionamento permite adaptar a representação visual aos requisitos de entrada das redes neurais, mantendo a correspondência original entre bytes e pixels e evitando a perda de informações decorrente de operações convencionais de redimensionamento.

4.2. Extrator de Características

O esquema proposto gera uma sequência de imagens para cada aplicação analisada, mas realiza uma única classificação final para determinar se o APK apresenta comportamento malicioso. Para isso, cada imagem da sequência é inicialmente processada por um modelo DNN previamente treinado, que atua como extrator de características visuais. As ativações obtidas na última camada totalmente conectada da rede são utilizadas como representação vetorial (*embedding*) de cada imagem de entrada, resultando em um conjunto de características de alta dimensionalidade associado às diferentes regiões do binário analisado (Fig. 1, *Extração de Características*).

Seja h um modelo DNN previamente treinado empregado como extrator de características. Para cada subimagem I_j da sequência $I_1, I_2, \ldots, I_k$, obtém-se um vetor de características $\mathbf{x}_j = \phi(I_j, \theta)$, em que ϕ representa a função de transformação aplicada pela rede até a última camada totalmente conectada e θ corresponde aos parâmetros aprendidos durante o treinamento do modelo. Essa etapa resulta em uma sequência ordenada de vetores de características $\mathbf{X} = \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$, que atua como uma representação compacta do arquivo `classes.dex` completo, preservando as informações extraídas de cada segmento da aplicação.

A utilização desses *embeddings* permite combinar as características espaciais obtidas a partir das subimagens individuais em uma representação estruturada, adequada à etapa posterior de modelagem das dependências sequenciais presentes no binário analisado.

4.3. Classificação

A etapa de classificação do modelo proposto é realizada por uma RNN, responsável por analisar a sequência de vetores de características e capturar as relações estruturais entre as diferentes regiões do arquivo analisado. Ao modelar as dependências entre os *embeddings* extraídos das subimagens, a RNN consegue representar o contexto global do arquivo `classes.dex`, permitindo realizar uma classificação única da aplicação considerando sua organização binária completa, independentemente do tamanho original do APK.

Seja $\mathbf{X} = \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$ a sequência de *embeddings* obtidos a partir das subimagens geradas no processo de particionamento da aplicação (ver Seção 4.2). Para realizar a classificação da aplicação como um todo, a RNN processa os vetores sequencialmente, atualizando seu estado oculto interno h_t a cada passo t conforme a equação $h_t = \sigma(W_h h_{t-1} + W_x \mathbf{x}_t + b)$, onde W_h e W_x representam as matrizes de pesos associadas

ao estado oculto anterior e ao vetor de entrada atual, respectivamente, enquanto b corresponde ao vetor de bias da rede. Ao final do processamento da sequência, o estado oculto h_k representa uma codificação compacta das características estruturais globais do arquivo `classes.dex`. Essa representação é encaminhada para uma camada totalmente conectada com função de ativação *softmax*, responsável por gerar a distribuição probabilística final entre as classes, definida como $y = \text{softmax}(W_y h_k + b_y)$. Essa arquitetura many-to-one permite que a aplicação completa seja representada por uma única instância de classificação, mantendo a capacidade de detecção mesmo quando a quantidade de imagens geradas varia conforme o tamanho do APK analisado.

Como resultado, a abordagem proposta apresenta um framework capaz de preservar a estrutura original das aplicações Android por meio da divisão dos arquivos DEX em sequências de imagens com dimensões fixas, evitando as perdas de informação decorrentes do redimensionamento global das representações visuais. Além disso, o modelo combina um extrator de características baseado em DNN com uma arquitetura RNN, permitindo que padrões espaciais extraídos de regiões individuais do binário sejam integrados como dependências sequenciais capazes de representar o comportamento global da aplicação. A principal contribuição da proposta está na capacidade de processar adaptativamente APKs de diferentes tamanhos, utilizando uma estratégia de classificação many-to-one, proporcionando maior robustez e capacidade de generalização na identificação de famílias de malware Android complexas e heterogêneas.

5. Avaliação

O conjunto de experimentos conduzidos neste trabalho tem como objetivo responder às seguintes Questões de Pesquisa (QP):

- ***QP1***. *Qual é o desempenho em termos de acurácia de esquemas tradicionais de detecção de malware Android baseados em imagens?*
- ***QP2***. *Qual é o desempenho em termos de acurácia do modelo proposto?*
- ***QP3***. *Como o modelo proposto se compara às abordagens tradicionais?*

A próxima subseção descreve detalhadamente o ambiente experimental e o desempenho do esquema proposto.

5.1. Construção do *Dataset*

Propomos um novo dataset composto por aplicações Android coletadas a partir do repositório AndroZoo, abrangendo amostras obtidas entre 2019 e 2023. A definição dos rótulos de referência foi realizada utilizando a API do VirusTotal, considerando uma aplicação como malware somente quando identificada por pelo menos 2 mecanismos antivírus independentes; as demais amostras foram classificadas como *benigno*. O dataset final contém 13.850 aplicações Android, distribuídas igualmente entre 6.925 amostras maliciosas e 6.925 amostras benignas, o que reduz possíveis vieses decorrentes do desbalanceamento das classes durante o treinamento.

Para analisar a capacidade de generalização do modelo proposto, o dataset foi particionado em três subconjuntos: treinamento, validação e teste. A divisão foi realizada utilizando proporções de 60%, 20% e 20%, respectivamente, mantendo a distribuição das diferentes famílias de malware em cada subconjunto. No total, foram consideradas 149

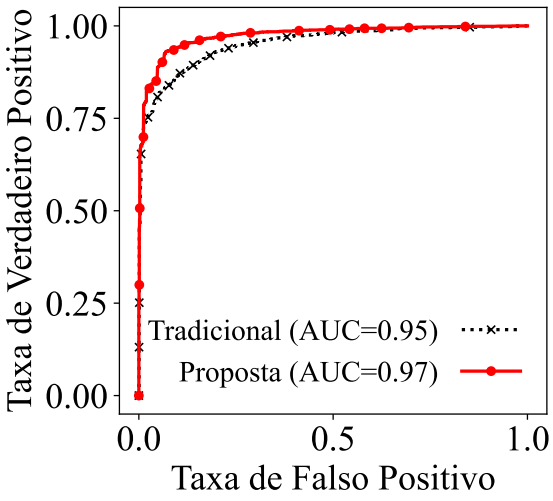


Figura 2. Curva ROC do esquema proposto em comparação com abordagens tradicionais.

famílias distintas de malware. Em relação às aplicações benignas (*goodware*), metade das amostras foi destinada ao treinamento, enquanto as demais foram distribuídas igualmente entre os conjuntos de validação e de teste, com 25% das amostras em cada subconjunto, garantindo equilíbrio entre as classes avaliadas.

As aplicações Android no formato APK foram convertidas em representações visuais seguindo uma metodologia semelhante às abordagens existentes na literatura. Inicialmente, o arquivo `classes.dex` é extraído de cada aplicação e convertido em uma imagem PNG em tons de cinza. Esse processo foi realizado utilizando a biblioteca Python Imaging Library (PIL) v. 8.4.0. A representação visual obtida a partir dos bytes binários foi posteriormente normalizada e particionada em sequências de imagens com múltiplas dimensões de 224×224 pixels (ver Seção 4.1).

Os classificadores foram avaliados no conjunto de teste por meio das métricas True-Positive Rate (TPR), True-Negative Rate (TNR) e F1-Score. A métrica TPR indica a proporção de amostras maliciosas corretamente classificadas pelo modelo, enquanto a TNR representa a proporção de aplicações benignas corretamente identificadas. Por sua vez, o F1-Score corresponde à média harmônica entre as métricas de Precisão e Revocação (*Recall*), fornecendo uma medida equilibrada do desempenho do classificador ao considerar ambas as classes.

5.2. Construção de modelos

Avaliamos o desempenho do modelo proposto, utilizando a arquitetura DNN AlexNet como extrator de características, e comparamos seus resultados com uma abordagem tradicional de detecção baseada no redimensionamento global das imagens. Na abordagem convencional, a imagem completa gerada a partir do APK é diretamente redimensionada para a resolução fixa de 224×224 pixels, sendo posteriormente utilizada como entrada para a classificação, sem o particionamento em sequências de subimagens.

A implementação das arquiteturas foi realizada com a API PyTorch. Para a abordagem tradicional baseada em AlexNet, configuramos a taxa de aprendizado em 10^{-4},

Tabela 1. Comparação de desempenho entre o esquema proposto e abordagens tradicionais.

Abordagem	Acurácia(%)	Recall	Precision	F1-Score
Tradicional (AlexNet)	88,25	0.90	0.85	0.88
Proposta (AlexNet)	92,57	0.92	0.93	0.93

tamanho de lote (*batch size*) igual a 64 e utilizamos o otimizador Adam. O treinamento foi realizado utilizando entropia cruzada binária (*binary cross-entropy*) como função de perda, por até 200 épocas, com aplicação de *early stopping* após 5 épocas consecutivas sem melhora no desempenho do conjunto de validação.

No modelo proposto, a arquitetura AlexNet é utilizada exclusivamente como extrator de características espaciais, sendo responsável por transformar cada subimagem da sequência em um vetor de características de alta dimensionalidade. Esses vetores são posteriormente processados pelo componente sequencial baseado em uma rede LSTM, responsável por modelar as dependências estruturais entre as diferentes regiões do arquivo `classes.dex`. A arquitetura LSTM foi implementada utilizando a API PyTorch e composta por duas camadas recorrentes com dimensão oculta de 256, seguidas por uma camada totalmente conectada para realizar a classificação final da aplicação.

O treinamento do modelo proposto utilizou uma taxa de aprendizado de 10^{-4}, o otimizador Adam, um tamanho de lote de 64 e a entropia cruzada binária como função de perda. Assim como na abordagem convencional, o treinamento foi limitado a 200 épocas e utilizou *early stopping* após 5 épocas sem melhoria adicional no desempenho do conjunto de validação.

5.3. Detecção de Malware Android Baseada em Imagens

O primeiro experimento foi conduzido com o objetivo de responder à *QP1*, avaliando o desempenho de abordagens tradicionais de detecção de malware Android baseadas em imagens. Para isso, analisamos a capacidade de classificação da arquitetura DNN AlexNet utilizando uma estratégia convencional de pré-processamento, na qual a imagem global gerada a partir do APK é redimensionada diretamente para a resolução fixa de 224×224 pixels. Esse cenário permite avaliar o comportamento de modelos que dependem de técnicas tradicionais de adaptação de entrada, nas quais representações byte-pixel, originalmente com dimensões variáveis, são transformadas para atender aos requisitos das arquiteturas de aprendizado profundo. O objetivo deste experimento é estabelecer uma referência de desempenho para abordagens amplamente utilizadas na literatura e analisar suas limitações de generalização em datasets compostos por aplicações Android reais.

A Tabela 1 apresenta os resultados obtidos pela abordagem tradicional baseada no redimensionamento global. Conforme observado, o método convencional alcança um F1-Score máximo de 0.88 na arquitetura AlexNet. Esse resultado indica que estratégias baseadas exclusivamente na normalização das dimensões da imagem apresentam limitações para representar adequadamente a complexidade estrutural dos arquivos APKs. A redução da resolução durante o processo de redimensionamento pode remover características relevantes do binário, prejudicando a capacidade do modelo em aprender padrões discriminativos associados ao comportamento malicioso. Consequentemente, a

perda de informação introduzida por esse processo compromete a generalização do classificador quando aplicado a amostras pertencentes a diferentes famílias de malware Android e com características estruturais distintas.

O segundo experimento foi realizado com o objetivo de responder à *QP2*, avaliando a eficácia do modelo proposto na detecção de malware Android. Para isso, implementamos a arquitetura baseada em LSTM descrita anteriormente, integrada ao processo de geração de sequências de imagens apresentado na Seção 4.1. A avaliação foi conduzida utilizando a arquitetura AlexNet como extratora de características espaciais, o que permitiu uma comparação direta com a abordagem tradicional baseada no redimensionamento global das imagens. Essa configuração permite analisar o impacto da preservação da estrutura original do binário, obtida por meio do particionamento da imagem, aliada à capacidade da LSTM de modelar as dependências sequenciais entre diferentes regiões do arquivo analisado.

A Tabela 1 apresenta os resultados obtidos pelo modelo proposto em comparação com a abordagem convencional. Os resultados demonstram uma melhoria significativa na capacidade de detecção ao se utilizar a estratégia baseada em sequências de imagens. Especificamente, o modelo proposto alcança um F1-Score de 0.92 com a arquitetura AlexNet, representando um ganho de 0.05 em relação ao método tradicional correspondente. Esse aumento evidencia que a preservação das informações estruturais do arquivo `classes.dex`, combinada com a modelagem sequencial realizada pela LSTM, permite capturar padrões relevantes que são parcialmente eliminados pelo redimensionamento convencional das imagens. Dessa forma, a abordagem proposta consegue lidar de maneira mais eficiente com a variabilidade de tamanho dos APKs, mantendo uma representação mais completa do comportamento estrutural das aplicações analisadas.

A Figura 2 apresenta a curva Receiver Operating Characteristic (ROC) do modelo proposto em comparação com a abordagem tradicional. Os resultados indicam que a melhoria obtida permanece consistente em diferentes limiares de decisão do classificador. A abordagem proposta aumenta a Area Under the Curve (AUC) em 0.03 em comparação ao método baseado em redimensionamento, demonstrando maior capacidade discriminativa entre aplicações benignas e maliciosas. Esse resultado reforça que a combinação entre o particionamento da representação visual e a análise sequencial dos vetores de características contribui não apenas para uma melhoria na classificação final, mas também para uma maior robustez do modelo em cenários que envolvem diferentes famílias de malware Android.

Para responder à *RQ3*, analisamos o comportamento do modelo proposto considerando diferentes famílias de malware presentes no dataset avaliado. Nesse experimento, investigamos a métrica TPR individualmente para cada família de malware, permitindo avaliar a capacidade de generalização da abordagem diante de diferentes padrões estruturais e comportamentais. Conforme apresentado na Figura 3, o modelo proposto baseado em AlexNet obtém uma TPR superior ou igual a 90% em 10 das 15 famílias de malware analisadas. Em comparação, a abordagem tradicional baseada no redimensionamento global das imagens alcança o mesmo nível de desempenho em apenas 6 famílias.

Esses resultados demonstram que a preservação das características estruturais do APK, proporcionada pelo particionamento das imagens, contribui para maior robustez na

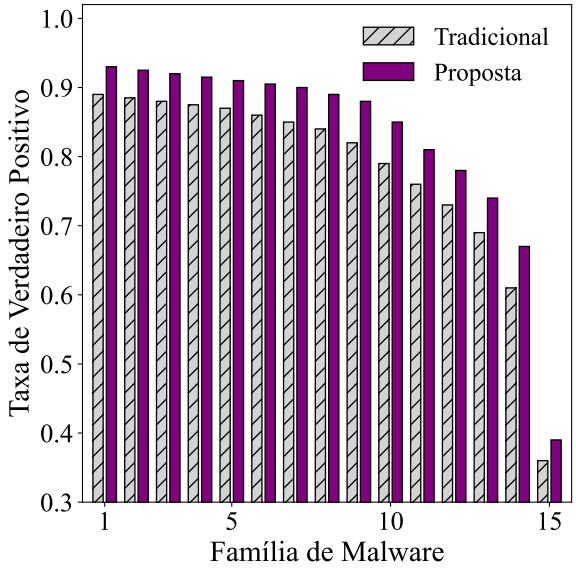


Figura 3. CDF da acurácia de detecção de malware do esquema proposto em comparação com abordagens tradicionais.

identificação de diferentes famílias de malware. Ao evitar a remoção de informações causada pelo redimensionamento convencional, o modelo proposto consegue explorar padrões discriminativos presentes em estruturas binárias heterogêneas, reduzindo a crescente degradação de desempenho observada em abordagens tradicionais quando aplicadas a famílias de malware com características distintas.

6. Conclusão

A crescente complexidade das aplicações Android e a variação significativa no tamanho dos arquivos binários representam limitações importantes para abordagens tradicionais de detecção de malware baseadas em imagens, principalmente devido às perdas de informação introduzidas por técnicas convencionais de redimensionamento. Neste trabalho, apresentamos um novo framework que preserva a estrutura original dos dados binários por meio do particionamento dos arquivos DEX em sequências de imagens de dimensões fixas. A arquitetura proposta combina uma DNN para extração de características espaciais com uma RNN responsável por capturar as dependências sequenciais entre diferentes regiões do binário, permitindo o processamento de aplicações Android de tamanhos variados sem a necessidade de alterar sua representação original. Os resultados experimentais demonstram que a abordagem proposta supera os métodos tradicionais baseados em redimensionamento, alcançando um F1-Score de até 0.92 e apresentando maior capacidade de generalização entre diferentes famílias reais de malware Android. Como trabalhos futuros, pretendemos estender o framework para uma abordagem multivisualização (*multi-view*), integrando informações complementares obtidas por meio de análise dinâmica, como sequências de chamadas de sistema e características de tráfego de rede, visando ampliar a robustez e a capacidade de generalização do modelo em cenários mais complexos.

Agradecimentos

Este trabalho foi parcialmente financiado pela Fundação Araucária e pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), sob os números de processo 302937/2023-4, 407879/2023-4, 442262/2024-8, 406603/2025-1 e 307706/2025-7.

Referências

Anton Kivva (2026). Mobile malware evolution in 2025.

de Oliveira, V. M., de Oliveira, H. M., Santos, G. M., Geremias, J., and Viegas, E. K. (2025). A big data framework for scalable and cross-dataset capable machine learning in network intrusion detection systems. *IEEE Access*, 13:129419–129431.

Espindola, A. d. S., Casimiro, A., Santin, A. O., Ferreira, P. M., and Viegas, E. K. (2026a). Enhancing intrusion detection generalization via diversity-driven multi-view ensemble learning in industrial systems. *Future Generation Computer Systems*, 182:108458.

Espindola, A. d. S., Santin, A. O., Casimiro, A., Ferreira, P. M., and Viegas, E. K. (2026b). Understanding the adversary: A survey of adversarial machine learning in network intrusion detection. *Computer Science Review*, 62:100995.

Geremias, J., Viegas, E. K., Santin, A. O., Britto, A., and Horchulhack, P. (2023). Towards a reliable hierarchical android malware detection through image-based cnn. In *2023 IEEE 20th Consumer Communications & Networking Conference (CCNC)*, page 242–247. IEEE.

Horchulhack, P., Viegas, E. K., and Santin, A. O. (2022). Detection of service provider hardware over-commitment in container orchestration environments. In *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, page 6354–6359. IEEE.

Lan, T., Darwaish, A., Nait-Abdesselam, F., and Gu, P. (2023). Defensive randomization against adversarial attacks in image-based android malware detection. In *ICC 2023 - IEEE International Conference on Communications*. IEEE.

Maniriho, P., Mahmood, A. N., and Chowdhury, M. J. M. (2024). A survey of recent advances in deep learning models for detecting malware in desktop and mobile platforms. *ACM Comput. Surv.*, 56(6):1–41.

Rodrigues, M. G., Viegas, E. K., Santin, A. O., and Enembreck, F. (2025). A mlops architecture for near real-time distributed stream learning operation deployment. *Journal of Network and Computer Applications*, 238:104169.

Sharma, T. and Rattan, D. (2025). Characterization of android malwares and their families. *ACM Comput. Surv.*, 57(5):1–31.

Shen, L., Feng, J., Chen, Z., Sun, Z., Liang, D., Li, H., and Wang, Y. (2023). Self-attention based convolutional-LSTM for android malware detection using network traffics grayscale image. *Appl. Intell.*, 53(1):683–705.

Shu, L., Dong, S., Su, H., and Huang, J. (2023). Android malware detection methods based on convolutional neural network: A survey. *IEEE Trans. Emerg. Top. Comput. Intell.*, pages 1–21.

StatCounter (2026). Mobile operating system market share worldwide.

Tang, J., Xu, W., Peng, T., Zhou, S., Pi, Q., He, R., and Hu, X. (2024). Android malware detection based on a novel mixed bytecode image combined with attention mechanism. *Journal of Information Security and Applications*, 82:103721.

Usama Tanveer, M., Munir, K., Alabdulatif, A., Najdawi, A. R., and Jhaveri, R. H. (2025). Malware-seqguard: An approach utilizing lstm and gru for effective detection of evolving malware in android environments. *IEEE Access*, 13:117355–117373.

Viegas, E. K., Santin, A. O., Cogo, V. V., and Abreu, V. (2020). *Facing the Unknown: A Stream Learning Intrusion Detection System for Reliable Model Updates*, page 898–909. Springer International Publishing.

Xiao, X., Zhang, S., Mercaldo, F., Hu, G., and Sangaiah, A. K. (2017). Android malware detection based on system call sequences and lstm. *Multimedia Tools and Applications*, 78(4):3979–3999.

Yadav, P., Menon, N., Ravi, V., Vishvanathan, S., and Pham, T. D. (2022). EfficientNet convolutional neural networks-based android malware detection. *Comput. Secur.*, 115(102622):102622.

Yapici, M. M. (2025). 3dmaldroid: A novel 3d image based approach for android malware detection and classification. *Computers and Electrical Engineering*, 127:110542.

Zou, B., Cao, C., Tao, F., and Wang, L. (2022). Imclnet: A lightweight deep neural network for image-based malware classification. *Journal of Information Security and Applications*, 70:103313.