Classificação de Malware Android com Multi-View e Late Fusion de Probabilidades

Philipe Fransozi, Altair O. Santin, e Eduardo K. Viegas

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

{philipe.hfransozi, santin, eduardo.viegas}@ppgia.pucpr.br

Resumo. *A detecção de malware Android baseada em aprendizado de máquina tem se consolidado como uma alternativa diante da evolução e da diversidade de aplicações maliciosas no ecossistema. Apesar do potencial das abordagens multi-view, a integração direta de características heterogêneas frequentemente falha em explorar sinais complementares devido à dominância discriminativa da perspectiva estática. Para contornar essa limitação, o presente trabalho apresenta uma arquitetura baseada em late fusion probabilística que preserva a independência operacional de cada representação ao longo do pipeline de aprendizado de máquina. O método avalia estratégias de combinação linear e de fusão seletiva condicionada à confiança, isolando o comportamento de cada perspectiva antes da tomada de decisão. Dessa forma, utilizamos as características dinâmicas prioritariamente em cenários em que o classificador estático apresenta alto grau de incerteza. Os resultados experimentais obtidos com base em 50.500 aplicações alinhadas demonstram que as abordagens de fusão propostas superam, de forma consistente, o modelo estático unimodal de referência. Especificamente, a estratégia de integração tardia atinge o melhor desempenho absoluto do sistema, reduzindo o erro residual em aproximadamente 1,7% em relação ao F1-score máximo atingível.*

1. Introdução

A detecção de *malware* Android baseada em Aprendizagem de Máquina (ML) tem se consolidado como uma alternativa relevante diante da diversidade e da evolução contínua de aplicações maliciosas [Almarri et al. 2025, Guerra-Manzanares 2024]. Nesse cenário, um Pacote de Aplicação Android (APK) pode ser representado por diferentes conjuntos de características, incluindo atributos estáticos, extraídos sem a execução da aplicação, e atributos dinâmicos, obtidos a partir do comportamento observado durante a execução da aplicação [Bashir et al. 2024, Ferreira et al. 2026]. Essas representações oferecem perspectivas distintas sobre a mesma amostra: enquanto a análise estática descreve elementos estruturais e potenciais do aplicativo, a análise dinâmica busca capturar evidências comportamentais materializadas durante a execução.

Apesar desse potencial, a combinação entre representações estáticas e dinâmicas ainda enfrenta desafios práticos. Uma *view* estática tende a ser mais direta de obter e frequentemente apresenta alta capacidade discriminativa em classificadores tradicionais [Manzil and Naik 2024, Espindola et al. 2026b]. Por outro lado, as *views* dinâmicas

dependem da execução da aplicação e podem apresentar maior variabilidade, especialmente quando o ambiente de análise não abrange todos os comportamentos relevantes do aplicativo [Yunmar et al. 2024, Geremias et al. 2026]. Assim, uma questão importante não é apenas identificar qual representação apresenta melhor desempenho isolado, mas também verificar se evidências dinâmicas podem complementar a *view* estática em casos em que ela apresenta erro ou baixa confiança [Filho et al. 2025].

Uma estratégia direta em cenários *multi-view* é a *early fusion*, na qual atributos extraídos de diferentes representações são concatenados antes do treinamento do classificador [Meng et al. 2025a, Espindola et al. 2026a]. Contudo, a concatenação direta não garante que sinais heterogêneos sejam explorados de forma equilibrada, especialmente quando uma *view* apresenta capacidade discriminativa dominante [Meng et al. 2025b,de Oliveira et al. 2025]. Nesse contexto, a ausência de ganho por concatenação não implica, necessariamente, a ausência de complementaridade. Para avaliar essa relação, é necessário analisar não apenas métricas agregadas, mas também a discordância entre classificadores, a sobreposição de erros e a capacidade de uma *view* corrigir falhas da outra.

Contribuição. Diante disso, este trabalho tem como objetivo geral investigar a complementaridade entre uma representação estática dominante e representações dinâmicas na detecção binária de *malware* Android. O foco está em verificar se evidências dinâmicas contribuem quando a predição estática apresenta erro ou baixa confiança. Para isso, conduzimos uma análise experimental em uma base Android *multi-view*, composta por uma *view* estática e duas *views* dinâmicas. Classificadores LightGBM são treinados separadamente por *view*, permitindo avaliar o desempenho individual, a discordância e a correlação entre os erros. Por fim, avaliamos estratégias de *late fusion* baseadas em probabilidades, incluindo a média ponderada e a fusão com *gate* de confiança.

Em resumo, as principais contribuições deste trabalho são:

- Uma base Android *multi-view* composta por 50.500 APKs e representações estáticas e dinâmicas para detecção binária de *malware*.
- Uma análise experimental da complementaridade entre *views*, considerando o desempenho individual, a discordância entre classificadores e a correção de erros.
- Uma abordagem de classificação baseada em *late fusion* de probabilidades, que supera o melhor classificador estático avaliado e reduz o erro residual em aproximadamente 1,7% em relação ao F1-score máximo.

2. Trabalhos Relacionados

A literatura de detecção de *malware* Android tem explorado representações *multi-view* para combinar diferentes evidências extraídas das aplicações. Wu et al. [Wu et al. 2020] propõem o MVIIDroid, que integra permissões, chamadas de API e *opcodes* por meio de *Multiple Kernel Learning*. Trung et al. [Trung et al. 2025] propõem o DMLDroid, que combina permissões, *intents*, imagem do DEX e grafos de chamadas de API em uma arquitetura de fusão multimodal profunda. Embora esses trabalhos indiquem que múltiplas representações estáticas podem capturar aspectos distintos do aplicativo, concentram-se principalmente em evidências extraídas sem execução, sem abordar diretamente a complementaridade entre características estáticas e sinais dinâmicos observados em execução.

Outra linha de pesquisa combina características estáticas e dinâmicas para ampliar a descrição das aplicações. Taher et al. [Taher et al. 2023] propõem o DroidDetectMW,

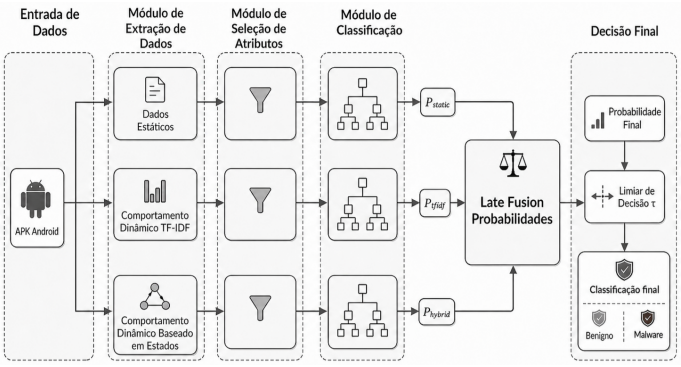

Figura 1. Visão geral da arquitetura proposta para detecção de *malware* Android baseada em múltiplas *views* e *late fusion* em nível de probabilidade.

que integra atributos estáticos com informações dinâmicas coletadas em sandbox. Nasser et al. [Nasser et al. 2024] propõem o DL-AMDet, que combina um módulo estático e um módulo dinâmico para a decisão final. Kilic et al. [Kilic et al. 2025] propõem o FABLDroid, que utiliza atributos híbridos derivados do KronoDroid, análise fatorial e Breadth Learning. Rashid et al. [Rashid et al. 2025] propõem uma arquitetura em camadas que combina permissões, *intents*, chamadas de API, fluxos de rede e relações sequenciais, acionando a análise dinâmica de forma condicional. Zhang et al. [Zhang et al. 2025] propõem o MPDroid, baseado em pré-treinamento multimodal com características estáticas e dinâmicas, para permitir inferência posterior com restrição de modalidade. Apesar de explorarem fontes heterogêneas de informação, essas abordagens avaliam majoritariamente o resultado final do modelo integrado ou da representação aprendida, sem analisar explicitamente como uma *view* corrige erros, introduz erros ou contribui de forma localizada em relação às demais.

Diante dessas limitações, este trabalho investiga a complementaridade entre *views* estáticas e dinâmicas com base nas decisões produzidas por classificadores treinados separadamente. Diferentemente das abordagens que avaliam apenas o modelo integrado, analisamos o desempenho individual das *views*, a discordância entre classificadores, a sobreposição de erros e a capacidade de correção entre as representações. Além disso, avaliamos estratégias de fusão tardia baseadas em probabilidades, incluindo a média ponderada e a fusão com *gate* de confiança, com o objetivo de explorar evidências dinâmicas, principalmente em regiões de incerteza da *view* estática.

3. Arquitetura *Multi-view* com *Late Fusion*

A arquitetura proposta organiza o fluxo de processamento e de experimentação em quatro módulos funcionais principais, dispostos sequencialmente: extração de dados, seleção de atributos, classificação especializada por *view* e *late fusion*, conforme ilustrado no diagrama da Figura 1.

O princípio da proposta é preservar a independência e a integridade informacional das representações estáticas e dinâmicas ao longo de quase todo o pipeline. Ao adiar a integração dos dados até a camada de decisão final, mitiga-se o risco de que características de uma *view* dominante mascarem ou anulem os sinais preditivos das demais, permitindo, simultaneamente, uma avaliação isolada e rigorosa do desempenho de cada perspectiva. Na execução prática do fluxo, o pipeline processa cada APK de entrada individualmente e em paralelo entre os módulos. Inicialmente, o módulo de extração decompõe o arquivo

e gera múltiplos vetores de características brutos, que são associados a um identificador único para garantir o alinhamento exato entre as amostras. No segundo módulo, cada um desses vetores passa por um processo independente de seleção de atributos, aplicando critérios estatísticos de relevância para eliminar redundâncias e reduzir a dimensionalidade do espaço de busca. Em seguida, as características selecionadas alimentam o módulo de classificação, no qual modelos preditivos distintos, configurados especificamente para a natureza de cada dado, calculam as probabilidades de pertinência de classe. Por fim, o módulo de *late fusion* atua como uma camada de decisão inteligente, combinando os vetores de probabilidade resultantes por meio de mecanismos ponderados ou baseados em limiares de incerteza para consolidar o diagnóstico final sobre a maliciosidade da aplicação.

As seções seguintes detalham os módulos que implementam a proposta.

3.1. Extração de Características e Seleção de Atributos

O Módulo de Extração de Dados transforma cada APK em três perspectivas de análise complementares. A primeira corresponde à *view* estática, construída a partir de chamadas de API extraídas do artefato da aplicação, sem execução do aplicativo. Essa representação busca capturar evidências estruturais e funcionais presentes no pacote, como uso de componentes, serviços, permissões implícitas em APIs e recursos potencialmente sensíveis.

A segunda representação corresponde à *view* dinâmica baseada em TF-IDF das chamadas de sistema. Nesse caso, os traços de execução são tratados como sequências de eventos, permitindo representar padrões locais de ocorrência e de ordenação por meio de *n-grams*. Essa perspectiva privilegia a frequência ponderada dos padrões comportamentais observados durante a execução.

A terceira representação dinâmica organiza os traços de execução em termos de estados, transições e estatísticas globais. Enquanto a representação TF-IDF enfatiza padrões sequenciais locais, essa *view* busca capturar aspectos estruturais do fluxo de execução, como recorrência de eventos, relações entre chamadas consecutivas e propriedades agregadas da sequência observada.

Após a extração, o Módulo de Seleção de Atributos aplica, separadamente em cada *view*, um processo de redução do espaço de características. O objetivo é remover atributos pouco informativos, redundantes ou instáveis antes da classificação. Esse procedimento é especialmente importante em cenários de detecção de *malware* Android, nos quais o espaço original de atributos pode ser amplo e conter características raras, genéricas ou repetitivas.

O processo de seleção é organizado em três estágios encadeados: filtro de presença, redução de redundância e ranqueamento supervisionado de relevância. Como resultado, cada *view* mantém um subconjunto próprio de atributos selecionados. Essa organização preserva a separação entre os espaços de características e permite que a etapa posterior de classificação avalie o comportamento individual de cada representação antes da fusão das probabilidades.

3.2. Classificação por *View* e *Late Fusion*

O Módulo de Classificação recebe os atributos selecionados de cada *view* e treina classificadores específicos para cada representação. Cada classificador estima a probabilidade

de uma amostra pertencer à classe *malware*. Sejam p_s, p_t e p_e as probabilidades produzidas, respectivamente, pelos classificadores da *view* estática, da *view* dinâmica TF-IDF e da *view* dinâmica baseada em estados e transições.

Diferentemente de estratégias de *early fusion*, nas quais os atributos de diferentes representações são concatenados antes do treinamento, a arquitetura proposta preserva a separação entre as *views* até o nível de decisão. Assim, o módulo de *late fusion* atua sobre as probabilidades produzidas pelos classificadores, e não sobre o espaço original de atributos. Essa escolha permite analisar o comportamento individual de cada representação e verificar se as evidências dinâmicas contribuem para corrigir falhas da representação estática dominante.

A primeira estratégia avaliada é a *late fusion* por meio de uma média ponderada. Seja $\mathcal{V}_c$ o conjunto de *views* utilizado na configuração de fusão c. Para cada *view* $i \in \mathcal{V}_c$, seja p_i a probabilidade atribuída pelo classificador correspondente. A probabilidade final é calculada por:

$$p_{\text{pond}}^{(c)} = \sum_{i \in \mathcal{V}_c} w_i^{(c)} p_i, \tag{1}$$

em que $w_i^{(c)}$ denota o peso analítico atribuído à contribuição da *view* i na respectiva configuração c. Para assegurar que a saída combinada permaneça no espaço probabilístico válido e preservar a interpretabilidade dos coeficientes, o vetor de pesos é sujeito às seguintes restrições:

$$\sum_{i \in \mathcal{V}_c} w_i^{(c)} = 1, \quad w_i^{(c)} \geq 0. \tag{2}$$

em que o parâmetro $w_i^{(c)}$ representa o peso de importância atribuído à *view* individual i em uma configuração de fusão específica c, em que i pertence ao conjunto $\mathcal{V}_c$ de todas as perspectivas ativas no modelo. A restrição de soma igual a 1 define uma combinação convexa que normaliza as contribuições no espaço probabilístico, enquanto a imposição de $w_i^{(c)} \geq 0$ garante a não-negatividade e a interpretabilidade direta desses coeficientes. A partir do valor de probabilidade sintetizado $p_{\text{pond}}^{(c)}$, a classificação binária final $\hat{y} \in \{0, 1\}$ é estabelecida mediante a aplicação de um limiar de decisão operacional $\tau^{(c)} \in (0, 1)$, parametrizado especificamente para a configuração c:

$$\hat{y} = \begin{cases} 1, & \text{se } p_{\text{pond}}^{(c)} \geq \tau^{(c)} \\ 0, & \text{caso contrário.} \end{cases} \tag{3}$$

Os pesos $w_i^{(c)}$ e o limiar $\tau^{(c)}$ são escolhidos no conjunto de validação, maximizando o F1-score. Essa estratégia permite combinar as probabilidades das três *views* em configurações nas quais todas utilizam o mesmo número de atributos selecionados ou em configurações nas quais cada *view* utiliza um subconjunto de tamanho distinto.

A segunda estratégia é a *late fusion* condicionada à confiança da *view* estática. Essa abordagem parte da hipótese de que as evidências dinâmicas são mais úteis quando

a predição estática está em uma região de incerteza. Primeiro, calcula-se a probabilidade dinâmica média:

$$p_d = \frac{p_t + p_e}{2} \tag{4}$$

Em seguida, estima-se a confiança da predição estática pela distância da probabilidade estática em relação ao ponto central da escala probabilística, 0,5:

$$conf_s = 2|p_s - 0{,}5|. \tag{5}$$

O valor 0,5 não representa o limiar de decisão final, mas o ponto de máxima incerteza usado para medir a confiança da *view* estática. A decisão binária é obtida posteriormente pela aplicação do limiar τ, selecionado no conjunto de validação. Dado um parâmetro de margem m e um peso α, a probabilidade final é definida por:

$$p_{\text{conf}} = \begin{cases} \alpha p_s + (1-\alpha) p_d, & \text{se } conf_s \leq m \\ p_s, & \text{caso contrário.} \end{cases} \tag{6}$$

Nesse caso, as probabilidades dinâmicas influenciam a decisão apenas quando a *view* estática apresenta baixa confiança. Quando a predição estática é suficientemente confiante, a saída final permanece ancorada na probabilidade estática. Assim, a fusão condicionada à confiança busca incorporar evidências dinâmicas de forma seletiva, evitando que essas *views* modifiquem decisões estáticas já consideradas confiáveis.

4. Avaliação

Esta seção busca responder às seguintes questões de pesquisa (QP):

- **QP1.** *Como as views estática e dinâmicas se comportam individualmente na detecção binária de malware Android?*
- **QP2.** *Em que medida as views dinâmicas corrigem ou introduzem erros em relação à view estática dominante?*
- **QP3.** *Em que medida estratégias de late fusion baseadas em probabilidades exploram a complementaridade entre views de forma interpretável?*

4.1. Base Multi-view e Protocolo de Extração

A avaliação foi conduzida sobre uma base Android *multi-view* construída a partir do *dataset* público MaDroid [Duan et al. 2024]. O MaDroid fornece registros brutos de análise dinâmica, compostos por sequências de chamadas de sistema observadas durante a execução de aplicativos Android. Como sua natureza é essencialmente dinâmica, este trabalho estende a base com uma representação estática associada às mesmas aplicações.

Com base nos registros brutos do MaDroid, foram construídas duas representações dinâmicas. A primeira aplica TF-IDF a n-grams de chamadas de sistema, representando padrões locais de ocorrência e de ordem entre eventos. A segunda organiza os traços de execução em estados, transições e estatísticas, buscando capturar as relações estruturais do comportamento observado. Dessa forma, a base passa a incorporar duas perspectivas

dinâmicas complementares: uma voltada à frequência ponderada de padrões sequenciais e outra voltada à organização estrutural do fluxo de execução.

Como o MaDroid não disponibiliza uma *view* estática correspondente, os arquivos APK associados às amostras foram recuperados por meio da plataforma AndroZoo. Em seguida, as características estáticas foram extraídas com o AndroPyTool, produzindo registros de chamadas de API obtidos diretamente do artefato da aplicação, sem execução. A partir desses registros, foi construída uma matriz *bag-of-APIs*, na qual as linhas representam aplicações, as colunas representam chamadas de API e os valores correspondem às respectivas contagens em cada APK.

A extração estática gerou, inicialmente, 57.049 chamadas à API. Como esse espaço incluía categorias pouco específicas para a detecção de *malware*, como APIs de compatibilidade, interface, renderização, utilidades genéricas e funcionalidades ordinárias de aplicação, foi aplicada uma filtragem semântica baseada em APIs de baixo risco. Após essa etapa, a *view* estática utilizada nos experimentos passou a conter 15.455 atributos, representados como contagens inteiras.

Com essa organização, a base passa a oferecer um cenário experimental alinhado à análise *multi-view*. Isso permite comparar as representações individualmente e avaliar, de forma controlada, quando evidências dinâmicas acrescentam informação útil à representação estática dominante.

4.2. Desenho Experimental

O desenho experimental foi definido para avaliar o desempenho individual das *views* e o efeito da *late fusion* na detecção binária de *malware* Android. Todas as representações foram alinhadas à mesma lista de amostras, preservando a ordem, os rótulos e o particionamento estratificado $60/20/20$ entre treino, validação e teste. O treino foi usado para ajustar vocabulários, filtros, seletores de atributos e classificadores; a validação, para selecionar pesos de fusão e limiares de decisão; e o teste, exclusivamente para a avaliação final.

A *view* estática corresponde à matriz de contagens de chamadas de API descrita na Subseção 4.1, com 15.455 atributos após a filtragem semântica. A representação dinâmica TF-IDF foi construída com *n-grams* de chamadas de sistema de ordem 2 e 3, vocabulário máximo de 10.000 atributos, frequência documental entre 0,5% e 90%, ponderação sublinear e normalização L2. Já a representação baseada em estados e transições utiliza uma modelagem de primeira ordem dos traços de execução, combinando atributos associados a eventos, transições consecutivas e estatísticas globais da sequência. Em ambos os casos dinâmicos, o vocabulário foi aprendido exclusivamente no conjunto de treino e aplicado às demais partições.

Cada *view* foi submetida ao mesmo processo de seleção de atributos, ajustado apenas no conjunto de treino, combinando filtro de presença, remoção de redundância por Spearman e seleção supervisionada com ExtraTreesClassifier e eliminação recursiva de atributos. Foram avaliados subconjuntos com 50, 100, 200 e 500 atributos mais relevantes, utilizando-se o máximo disponível quando o número de atributos remanescentes era menor. Após os filtros, a *view* estática foi reduzida de 15.455 para 7.554 atributos, a TF-IDF de 10.000 para 9.263, e a representação de estados e transições de 2.044 para 319 atributos.

Tabela 1. Melhor desempenho individual por *view* no conjunto de teste.

View	Top-N efetivo	Limiar	F1-score	AUC-ROC
Estática	500	0,489	0,8816	0,9471
TF-IDF	500	0,394	0,7871	0,8670
Estados/transições	319	0,354	0,7698	0,8395

Na etapa de classificação, foram treinados classificadores LightGBM específicos para cada *view*. Os hiperparâmetros foram ajustados com GridSearchCV, com validação cruzada estratificada em 5 folds e F1-macro como métrica de seleção. Como baseline de integração direta, também foi avaliada a *early fusion*, concatenando a *view* estática, com 500 atributos, aos subconjuntos das representações dinâmicas com $k \in \{50, 100, 200, 500\}$ atributos mais relevantes.

Os parâmetros das estratégias de *late fusion* foram definidos exclusivamente no conjunto de validação. Na fusão por média ponderada, foram avaliadas combinações de pesos em uma grade simplex, com pesos não negativos e soma igual a 1. A grade utilizou passo de 0,01 nas configurações com o mesmo número de atributos por *view* e passo de 0,005 nas configurações com números distintos de atributos selecionados. Para cada combinação, o limiar de decisão foi escolhido no intervalo de 0,0 a 1,0, com passo de 0,001, usando F1-score como métrica principal e *average precision* como critério secundário.

Para a fusão condicionada à confiança, a validação selecionou a configuração com 500 atributos na *view* estática, 500 na TF-IDF e 100 na representação de estados e transições, com margem de incerteza $m = 0{,}31$, peso estático $\alpha = 0{,}66$ e limiar de decisão $\tau = 0{,}465$. As métricas reportadas incluem F1-score e AUC-ROC, além de correções, erros introduzidos e sobreposição de erros entre classificadores. A significância dos ganhos foi avaliada no conjunto de teste pelo teste exato de McNemar e pelo bootstrap de pares da diferença de F1.

4.3. Resultados por View e Análise de Complementaridade

Esta seção responde às QPs 1 e 2. Primeiro, avalia-se o desempenho individual das *views*; depois, analisa-se se as representações dinâmicas corrigem erros da *view* estática dominante ou apenas reproduzem suas decisões.

A Tabela 1 evidencia uma assimetria clara entre as representações: a *view* estática é dominante como classificador unimodal, enquanto as *views* dinâmicas apresentam desempenho global inferior. Esse resultado responde à QP1, mas não elimina a possibilidade de complementaridade localizada, pois uma representação dinâmica pode apresentar um F1-score menor e, ainda assim, corrigir erros específicos da representação estática.

Para responder à QP2, analisamos as decisões dos classificadores na região de incerteza da *view* estática, definida por $p_s \in [0{,}3, 0{,}7]$. Essa faixa concentra amostras nas quais a predição estática é menos confiável e, portanto, nas quais as evidências dinâmicas podem contribuir de forma mais relevante. Nessa análise, as representações dinâmicas utilizaram 200 atributos selecionados, com limiares de 0,397 para TF-IDF e 0,341 para estados e transições. A referência estática foi avaliada com 200 e 500 atributos, usando limiares de 0,486 e 0,489, respectivamente. Para estimar o potencial máximo de contribuição

Tabela 2. Correções e erros introduzidos pelas *views* dinâmicas na região de incerteza da *view* estática.

View dinâmica	Correções	Erros introduzidos	Saldo
Ref. estática: 200 atributos – erros na região: 606			
TF-IDF	270	284	-14
Estados e transições	274	305	-31
Oráculo dinâmico	358	179	+179
Ref. estática: 500 atributos – erros na região: 542			
TF-IDF	246	291	-45
Estados e transições	240	296	-56
Oráculo dinâmico	319	179	+140

das representações dinâmicas, também consideramos um oráculo dinâmico. Esse oráculo não representa um classificador aplicável à inferência, pois utiliza a informação do rótulo verdadeiro apenas para fins de análise. Ele indica, para cada amostra, se ao menos uma das *views* dinâmicas produziria a decisão correta quando a *view* estática erra. Assim, o oráculo funciona como um limite analítico superior para a complementaridade dinâmica observável na base.

A Tabela 2 mostra que as *views* dinâmicas contêm informação complementar, mas sua contribuição não é uniforme. Individualmente, TF-IDF e estados/transições corrigem parte dos erros da *view* estática, mas também introduzem novos erros, resultando em saldos negativos. Por outro lado, o oráculo dinâmico apresenta saldo positivo nos dois cenários, indicando que existe informação útil nas representações dinâmicas, embora ela não seja capturada por uma substituição direta da decisão estática. Quando a referência estática passa de 200 para 500 atributos selecionados, seu desempenho na região de incerteza melhora e o potencial do oráculo dinâmico diminui de +179 para +140. Esse resultado indica que a complementaridade dinâmica ainda existe, mas torna-se mais difícil de explorar quando a *view* estática já apresenta decisões mais robustas. Assim, a QP2 é respondida pela identificação de complementaridade localizada das *views* dinâmicas, especialmente em regiões de menor confiança da predição estática, o que motiva estratégias de *late fusion* capazes de controlar quando essas evidências devem influenciar a decisão final.

4.4. Avaliação das Estratégias de *Late Fusion*

Esta seção responde à QP3, avaliando se estratégias de *late fusion* baseadas em probabilidades conseguem explorar a complementaridade localizada entre as *views*. Os resultados anteriores indicaram que as representações dinâmicas podem corrigir erros da *view* estática em regiões de incerteza, mas também podem introduzir novos erros quando usadas sem controle. Portanto, a fusão deve preservar a força da representação estática dominante e incorporar evidências dinâmicas apenas quando estas tendem a contribuir para a decisão final.

A Tabela 3 apresenta os principais resultados das estratégias de *late fusion*. Foram avaliadas duas abordagens: a média ponderada das probabilidades, usando 200 atributos selecionados em cada *view*, e a fusão condicionada à confiança da *view* estática, que combina diferentes números de atributos por representação e incorpora evidências dinâmicas de forma seletiva.

Na média ponderada com 200 atributos por *view*, os pesos atribuídos às representações

Tabela 3. Resultados das estratégias de *late fusion* no conjunto de teste.

Estratégia	Parâmetros	Referência	Fusão	ΔF1
Média ponderada (200 por *view*)	$w = (0{,}78, 0{,}09, 0{,}13)$ $\tau = 0{,}466$	0,8772	0,8814	+0,0043
Fusão condicionada à confiança	$\alpha = 0{,}66$ $\text{conf}_s \leq 0{,}31$ $\tau = 0{,}465$	0,8816	0,8836	+0,0020

estáticas, TF-IDF e estados e transições foram, respectivamente, 0,78, 0,09 e 0,13. Essa configuração elevou o F1-score da referência estática de 0,8772 para 0,8814, resultando em ganho absoluto de 0,0043. A diferença foi estatisticamente significativa pelo teste exato de McNemar ($p = 0{,}0072$) e consistente no bootstrap pareado da diferença de F1, cujo intervalo de confiança de 95% foi $[+0{,}0020, +0{,}0067]$. A fusão condicionada à confiança obteve o melhor desempenho absoluto. Nessa configuração, foram combinados 500 atributos da *view* estática, 500 da representação TF-IDF e 100 da representação de estados e transições. A regra de fusão preserva a probabilidade estática quando sua confiança é elevada e incorpora a média das probabilidades dinâmicas apenas quando $conf_s \leq 0{,}31$. Com essa estratégia, o F1-score aumentou de 0,8816, obtido pela referência estática com 500 atributos, para 0,8836. Assim, a QP3 é respondida positivamente: estratégias de *late fusion* conseguem explorar a complementaridade entre *views*, desde que controlem a influência das evidências dinâmicas na decisão final. A média ponderada com 200 atributos por *view* apresenta a evidência estatística mais forte de ganho pareado, enquanto a fusão condicionada à confiança obtém o melhor resultado absoluto, superando a referência estática com 500 atributos.

5. Conclusão

Este trabalho investigou a complementaridade entre uma *view* estática e representações dinâmicas para detecção binária de *malware* Android. Para isso, construímos uma base *multi-view* a partir do MaDroid, estendendo seus registros dinâmicos de chamadas de sistema com uma representação estática baseada em chamadas de API extraídas dos APKs. A base resultante permitiu avaliar, de forma alinhada por amostra, o comportamento individual e combinado das representações estáticas, dinâmicas de TF-IDF e dinâmicas baseadas em estados e transições. Os resultados evidenciam a dominância da *view* estática no desempenho individual, mas também mostram que as representações dinâmicas oferecem complementaridade localizada em regiões de incerteza. Essa complementaridade, contudo, exige controle: embora corrijam parte dos erros da predição estática, as evidências dinâmicas também podem introduzir novos erros quando incorporadas diretamente. A avaliação de *late fusion* confirmou que a complementaridade dinâmica pode ser explorada, desde que sua influência seja controlada. A média ponderada com 200 atributos por *view* apresentou o ganho pareado mais consistente, enquanto a fusão condicionada à confiança obteve o melhor F1-score absoluto, superando a referência estática com 500 atributos. Como limitação, este estudo concentrou-se em classificadores LightGBM e em estratégias de fusão baseadas em probabilidades. Como trabalhos futuros, pretende-se avaliar protocolos temporais, calibrar probabilidades e explorar aprendizado *multi-view* e destilação de conhecimento, especialmente quando a *view* dinâmica está disponível no treinamento, mas restrita ou ausente na inferência.

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

Almarri, S., Bodokhi, A., and Frikha, M. (2025). A review of the recent trends in mobile malware evolution, detection, and analysis. *IEEE Access*, 13:108415–108445.

Bashir, S., Maqbool, F., Khan, F. H., and Abid, A. S. (2024). Hybrid machine learning model for malware analysis in android apps. *Pervasive and Mobile Computing*, 97:101859.

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.

Duan, G., Liu, H., Cai, M., Sun, J., and Chen, H. (2024). Madroid: A maliciousness-aware multifeatured dataset for detecting android malware. *Computers & Security*, 144:103969.

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.

Ferreira, I., Oliveira, J. V. O. d., Purkott, F., Geremias, J., and Viegas, E. K. (2026). Lightweight multi-view dynamic android malware detection via time-windowed feature extraction. *International Journal of Information Security*, 25(3).

Filho, A. G., Viegas, E. K., Santin, A. O., and Geremias, J. (2025). A dynamic network intrusion detection model for infrastructure as code deployed environments. *Journal of Network and Systems Management*, 33(4).

Geremias, J., Britto, A. S., Santin, A. O., and Viegas, E. K. (2026). Sparse mixture of experts for image-based multi-view android malware detection. In *2026 International Wireless Communications and Mobile Computing (IWCMC)*, page 1487–1492. IEEE.

Guerra-Manzanares, A. (2024). Machine learning for android malware detection: Mission accomplished? a comprehensive review of open challenges and future perspectives. *Computers & Security*, 138:103654.

Kilic, K., Atacak, I., and Dogru, I. A. (2025). Fabldroid: Malware detection based on hybrid analysis with factor analysis and broad learning methods for android applications. *Engineering Science and Technology, an International Journal*, 62:101945.

Manzil, H. H. R. and Naik, M. (2024). Detection approaches for android malware: Taxonomy and review analysis. *Expert Systems with Applications*, 238:122255.

Meng, Y., Luktarhan, N., Yang, X., and Zhao, G. (2025a). Gbadroid: an android malware detection method based on multi-view feature fusion. *The Journal of Supercomputing*, 81(3).

Meng, Z., Zhang, J., Guo, J., Wang, W., Huang, W., Cui, J., Zhong, H., and Xiong, Y. (2025b). Detecting android malware by visualizing app behaviors from multiple complementary views. *IEEE Transactions on Information Forensics and Security*, 20:2915–2929.

Nasser, A. R., Hasan, A. M., and Humaidi, A. J. (2024). Dl-amdet: Deep learning-based malware detector for android. *Intelligent Systems with Applications*, 21:200318.

Rashid, M. U., Qureshi, S., Abid, A., Alqahtany, S. S., Alqazzaz, A., ul Hassan, M., Al Reshan, M. S., and Shaikh, A. (2025). Hybrid android malware detection and classification using deep neural networks. *International Journal of Computational Intelligence Systems*, 18(1):52.

Taher, F., AlFandi, O., Al-kfairy, M., Al Hamadi, H., and Alrabaee, S. (2023). Droiddetectmw: A hybrid intelligent model for android malware detection. *Applied Sciences*, 13(13):7720.

Trung, D. M., Hao, T. D. A., Minh, L. H., Khoa, N. H., Cam, N. T., Pham, V.-H., and Duy, P. T. (2025). Dmldroid: Deep multimodal fusion framework for android malware detection with resilience to code obfuscation and adversarial perturbations.

Wu, Q., Li, M., Zhu, X., and Liu, B. (2020). Mviidroid: A multiple view information integration approach for android malware detection and family identification. *IEEE MultiMedia*, 27(4):48–57.

Yunmar, R. A., Kusumawardani, S. S., Widyawan, and Mohsen, F. (2024). Hybrid android malware detection: A review of heuristic-based approach. *IEEE Access*, 12:41255–41286.

Zhang, S. et al. (2025). Mpdroid: A multimodal pre-training android malware detection method with static and dynamic features. *Computers & Security*, 150:104262.