Redes Neurais sem Pesos Aplicadas à Detecção *Zero-day* de Programas Maliciosos por Imagem*

André Rotava[1], Priscila M. V. Lima[1], Diego L. C. Dutra[1]

[1]Programa de Engenharia de Sistemas e Computação (PESC/COPPE)
Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro — RJ — Brasil

{rotava,priscilamvl,ddutra}@cos.ufrj.br

Abstract. *This paper presents an exploratory investigation on the application of the WiSARD and ClusWiSARD architectures to image-based malware detection using the MaleVis dataset. Performance is evaluated in two scenarios: the conventional one, where all malware families are available for training, and a zero-day emulation scenario, where entire families are withheld from training. In the first scenario, the models achieved a mean balanced accuracy of approximately* 91%*. In the second, the binary detector maintained similar performance on known families, while recall for the withheld family varied widely depending on the visual characteristics of each class, exceeding* 60% *in approximately* 40% *of the evaluated classes.*

Resumo. *Este trabalho apresenta um estudo exploratório sobre a aplicação das arquiteturas WiSARD e ClusWiSARD à detecção de malware baseada em visualização de binários como imagens, utilizando o conjunto de dados MaleVis. O desempenho é avaliado em dois cenários: o convencional, com todas as famílias disponíveis para treinamento, e o de emulação de zero-day, em que famílias são omitidas do treinamento. No primeiro, os modelos atingiram acurácia balanceada média em torno de* 91%*. No segundo, o detector binário manteve desempenho semelhante sobre as famílias conhecidas, enquanto a revocação da família omitida variou amplamente dependendo das características visuais de cada classe, superando* 60% *em cerca de* 40% *das classes avaliadas.*

1. Introdução

A detecção eficaz de programas maliciosos, ou *malwares*, sendo uma tarefa crítica para a segurança dos sistemas computacionais, tem sido alvo de intensa investigação científica. Para fazer frente à contínua sofisticação das ameaças e das técnicas de evasão, como ofuscação de código e metamorfismo [Brezinski and Ferens 2023], a pesquisa em métodos de detecção baseados em aprendizado de máquina tem avançado consideravelmente, explorando tanto abordagens estáticas, que extraem características diretamente dos binários, quanto dinâmicas, que monitoram o comportamento do programa em execução [Ucci et al. 2019]. Não obstante os avanços obtidos, a detecção de *malwares* de famílias desconhecidas, não representadas no conjunto de treinamento, referida na literatura

*Esse projeto foi parcialmente financiado com recursos de bolsas de produtividade do CNPq (PQ-C: 304308/2025-0) e da FAPERJ (JCNE: E-26/204.481/2025)

como detecção de dia zero (*zero-day*), permanece um desafio em aberto para os modelos supervisionados convencionais [Guo 2023].

Redes Neurais sem Pesos, ou WNN (*Weightless Neural Networks*) constituem uma classe de modelos de aprendizado de máquina que se distingue pelo treinamento extremamente rápido e pela inferência eficiente, sem o custo computacional associado ao ajuste de pesos sinápticos [Aleksander et al. 2009]. Essas características tornam as WNN particularmente atraentes para aplicações onde a velocidade de resposta é crítica, como sistemas de detecção de intrusão e análise de *malware* em tempo real. Apesar de seu potencial, a aplicação de WNN ao problema de detecção de *malware* é uma linha de pesquisa ainda pouco explorada na literatura.

Este trabalho apresenta um estudo exploratório sobre a aplicação de redes neurais sem pesos — especificamente as arquiteturas WiSARD e ClusWiSARD — à detecção de *malware* baseada em visualização de binários como imagens, utilizando o conjunto de dados MaleVis [HUMIR Lab 2019]. O desempenho das arquiteturas é avaliado em dois cenários: o convencional, em que o modelo é treinado e testado sobre todas as famílias do conjunto de dados, e o de *zero-day*, em que famílias inteiras são omitidas do treinamento para simular a detecção de *malware* desconhecido. Em ambos os casos, adota-se validação cruzada k*-fold* e estratégias para mitigar o efeito do desbalanceamento de classes.

O restante deste artigo está organizado da seguinte forma. A Seção 2 discute os trabalhos relacionados publicados na literatura. A Seção 3 apresenta o referencial teórico sobre redes neurais sem pesos, com ênfase nas arquiteturas WiSARD e ClusWiSARD. A Seção 4 descreve a metodologia de detecção de *malware* proposta. A Seção 5 apresenta os experimentos realizados para sua avaliação. Por fim, a Seção 6 apresenta as conclusões do trabalho.

2. Trabalhos Relacionados

A detecção de *malware* por aprendizado de máquina tem sido amplamente investigada na literatura [Ucci et al. 2019]. Abordagens baseadas em análise estática exploram características extraídas diretamente dos binários executáveis sem a necessidade de execução, enquanto abordagens dinâmicas monitoram o comportamento do programa em tempo de execução [Egele et al. 2012]. Métodos híbridos, que combinam as duas estratégias, também têm sido propostos [Gopinath and Sethuraman 2023]. Entre as técnicas dinâmicas recentes, destaca-se o uso de sequências de chamadas de API processadas por modelos baseados em *transformers* [Quertier et al. 2024], enquanto abordagens baseadas em aprendizado federado têm sido exploradas para detecção de *malware* em dispositivos IoT com restrições de recursos e privacidade de dados [Rey et al. 2022].

Uma linha de pesquisa particularmente relevante para o presente trabalho é a abordagem baseada em visualização de binários como imagens. Nataraj [Nataraj 2015] demonstrou que arquivos executáveis podem ser convertidos diretamente em representações visuais e analisados como imagens, revelando padrões de textura característicos de cada família de *malware*. Explorando essa ideia, Bozkir et al. [Bozkir et al. 2019] avaliaram diversas arquiteturas de redes neurais convolucionais profundas — como ResNet, Inception, DenseNet, VGG e AlexNet — sobre um conjunto de dados próprio, denominado MaleVis, demonstrando a eficácia da abordagem baseada em visão computacional para classificação estática de *malware*. O MaleVis é composto de imagens coloridas em for-

mato PNG, codificadas no espaço de cores RGB e com resolução de 224×224 *pixels*, geradas a partir de arquivos executáveis binários de programas maliciosos e benignos, por um processo de conversão definido pelos autores.

Esse acervo foi disponibilizado publicamente pelos autores [HUMIR Lab 2019], e tem sido explorado em trabalhos mais recentes: Bavishi e Modi [Bavishi and Modi 2024] propuseram a arquitetura LeViT-MC, baseada em *transformers* de visão, obtendo 96,6% de acurácia na classificação multiclasse sobre o MaleVis, enquanto Salas e de Geus [Salas and de Geus 2024] alcançaram resultados competitivos com o MobileNet ajustado por transferência de aprendizado em múltiplos conjuntos de dados, incluindo o MaleVis. O presente trabalho também explora essa mesma base de imagens.

A despeito dos avanços em acurácia, a detecção de *malware zero-day* ainda carece de soluções eficazes. Modelos baseados em aprendizado supervisionado tendem a generalizar mal para famílias não vistas, como apontado por Guo [Guo 2023]. Abordagens propostas para mitigar esse problema incluem o uso de aprendizado semissupervisionado [Comar et al. 2013] e redes adversariais generativas [Kim et al. 2018], mas a questão continua sendo um problema relevante e ativo de pesquisa, especialmente em cenários onde a velocidade de resposta é crítica.

O uso de redes neurais sem pesos para detecção de *malware* foi investigado por Ramos et al. [Ramos et al. 2020], que aplicaram a WiSARD tanto para análise estática quanto dinâmica, utilizando o MaleVis como conjunto de dados para o cenário estático. Esse trabalho é o mais diretamente relacionado ao presente. Do ponto de vista metodológico, Ramos et al. utilizaram partições fixas de treino e validação, enquanto o presente trabalho emprega validação cruzada k*-fold*. Adicionalmente, os autores reportaram um classificador degenerado na detecção binária com a WiSARD, atribuído ao desbalanceamento do conjunto de dados, problema tratado explicitamente na metodologia do presente trabalho. Os autores avaliam também um cenário de análise dinâmica com um segundo conjunto de dados, não considerado no presente trabalho. Até onde é do conhecimento dos autores, não há trabalhos anteriores que combinem redes neurais sem pesos com detecção de *malware* baseada em visualização de binários além do já citado.

A combinação dessas duas linhas de pesquisa — detecção por imagem e modelos WNN — abre oportunidades pouco exploradas. Modelos baseados em redes neurais profundas, embora precisos, impõem custos computacionais elevados de treinamento e inferência, o que dificulta sua aplicação em contextos com restrições de recursos, como dispositivos embarcados e sistemas de detecção em tempo real. As WNN, por sua vez, oferecem treinamento extremamente rápido e inferência eficiente [Susskind et al. 2022], características desejáveis em ambientes onde a velocidade de resposta é crítica. A investigação de sua aplicabilidade ao problema de detecção de *malware* por imagem, incluindo a análise de seu comportamento frente a amostras de famílias não vistas, constitui, portanto, uma direção promissora de pesquisa.

3. Redes Neurais sem Pesos

Redes Neurais sem Pesos, ou WNN (*Weightless Neural Networks*), constituem uma classe de modelos de aprendizado de máquina que utiliza tabelas de consulta, ou LUT (*lookup table*), para realizar inferências. Suas origens remontam ao classificador por n-tuplas, que Bledsoe e Browning propuseram na década de 1950 [Bledsoe and Browning 1959].

Ao contrário das redes neurais tradicionais, as WNN não armazenam pesos sinápticos: o sistema realiza o aprendizado diretamente pela escrita em posições de memória RAM, enquanto a inferência executa leituras dessas mesmas posições [Aleksander et al. 2009]. Essa característica confere às WNN vantagens expressivas em termos de eficiência computacional: o treinamento e a classificação envolvem operações extremamente simples, o que resulta em baixa latência e consumo energético reduzido em comparação a redes neurais convencionais de porte equivalente.

Essas propriedades têm motivado avanços na pesquisa de WNN, especialmente para aplicações de computação na borda (*edge computing*), onde restrições de energia, memória e poder de processamento são determinantes. Trabalhos recentes propuseram novas arquiteturas de WNN, tais como a BTHOWeN [Susskind et al. 2022], a ULEEN [Susskind et al. 2023] e a DWN [Bacellar et al. 2024]. Tais modelos alcançam desempenho competitivo com redes neurais profundas, porém exigem apenas frações do custo energético e da latência, o que os torna particularmente adequados para dispositivos embarcados. No presente trabalho, empregam-se dois modelos de WNN — a WiSARD e a ClusWiSARD —, cujos fundamentos teóricos aparecem nas seções 3.1 e 3.2, respectivamente.

3.1. WiSARD

Aleksander, Thomas e Bowden propuseram a WiSARD (Wilkie, Stonham and Aleksander's Recognition Device) em 1984 [Aleksander et al. 1984], o que representou um marco no desenvolvimento das redes neurais sem pesos, servindo de base para o desenvolvimento de muitos modelos de WNN subsequentes. É uma arquitetura de aprendizado supervisionado destinada primariamente a tarefas de classificação. Sua estrutura básica está representada na Figura 1. A entrada do modelo, independentemente de sua forma original, é tratada como um vetor binário de I bits.

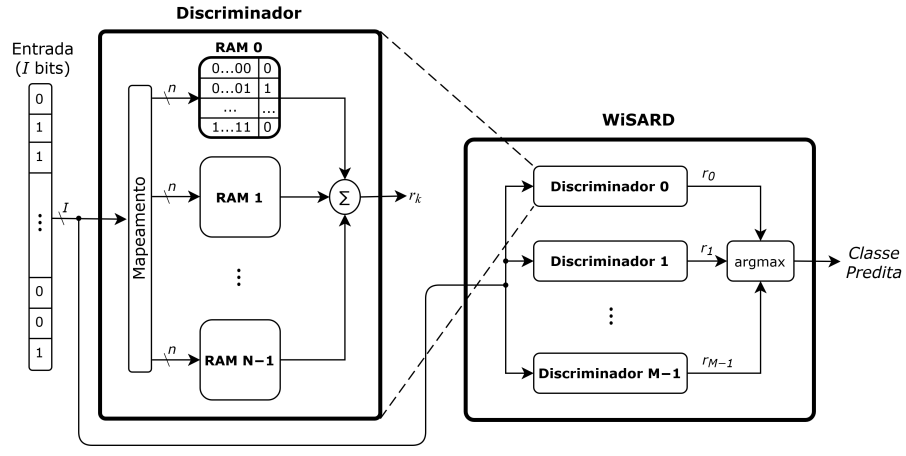

Figura 1. Estrutura básica da WiSARD.

A rede é composta por M discriminadores, um para cada classe a ser reconhecida. Cada discriminador é formado por uma única camada de N nós RAM, onde cada nó RAM consiste numa LUT de n entradas e 2^n posições, sendo $N \equiv I/n$. Os bits do vetor de entrada são particionados em N tuplas de n bits cada, por meio de um mapeamento pseudo-aleatório fixo, e cada tupla serve como entrada para um dos nós RAM,

compondo um endereço que aponta para uma posição da LUT. Esse mapeamento é definido na inicialização do modelo e permanece fixo durante o treinamento e a classificação. Opcionalmente, todos os discriminadores podem compartilhar o mesmo mapeamento, ou cada um pode ter o seu próprio. A calibração do tamanho das tuplas, n, é crucial para o bom desempenho do modelo.

A rede é inicializada com todas as LUT zeradas. No treinamento, apresenta-se cada observação rotulada somente ao discriminador correspondente à sua classe, e então, escreve-se o valor 1 nas posições das LUT acessadas em cada nó RAM, de acordo com os endereços definidos pelas tuplas que o vetor de entrada compôs. Dessa forma, o treinamento resume-se a operações de escrita em memória, sem qualquer cálculo de gradiente ou ajuste iterativo de parâmetros.

Na fase de classificação, uma observação de entrada desconhecida é apresentada a todos os discriminadores, e os endereços gerados pelo mapeamento são utilizados para realizar leituras nas LUT. A resposta de um discriminador corresponde à soma dos valores lidos em todos os seus nós RAM, representando, portanto, o número de endereços que coincidem com padrões previamente aprendidos. A observação é então atribuída à classe cujo discriminador apresentou a maior resposta.

Um mecanismo importante associado à classificação, desenvolvido posteriormente ao modelo WiSARD original, é o *bleaching*, que consiste em registrar, durante o treinamento, não apenas a ocorrência de cada endereço, mas a frequência com que ele é ativado [Carvalho et al. 2013]. Na classificação, um limiar mínimo de frequência (*bleaching threshold*) é aplicado: endereços ativados com frequência abaixo desse limiar são tratados como 0. Isso permite ao modelo discriminar padrões mais frequentes dos esporádicos, reduzindo a possibilidade de ocorrer *overfitting*.

Um ponto crucial nessa arquitetura é o tipo de codificação binária utilizado para representar os dados de entrada. Para atributos numéricos, são preferíveis codificações em que a distância de Hamming reflita a distância numérica entre os valores, como o código de Gray e a codificação unária. Codificações que não possuem tal característica, como o código binário natural e o padrão IEEE 754, devem ser evitadas. Para atributos categóricos, técnicas usuais como o *one-hot encoding* podem ser empregadas diretamente. A escolha do tipo de codificação influencia diretamente o desempenho do modelo, e diferentes abordagens têm sido investigadas na literatura [Kappaun et al. 2016].

3.2. ClusWiSARD

A ClusWiSARD [Cardoso et al. 2016] é uma extensão da WiSARD que incorpora um mecanismo de agrupamento (*clustering*) interno ao processo de treinamento, com o objetivo de aumentar a capacidade do modelo de representar classes com distribuições multimodais . Na WiSARD clássica, cada classe é representada por um único discriminador, o que pode levar ao reconhecimento indevido de padrões de teste muito dissimilares dos aprendidos anteriormente — consequência da natureza multiplicativa do modelo. A ClusWiSARD contorna esse problema permitindo que cada classe seja representada por múltiplos discriminadores, cada um correspondendo a um agrupamento distinto de observações, denominado *microcluster*.

No treinamento da ClusWiSARD, para cada nova observação rotulada apresentada ao modelo, calcula-se sua pontuação de reconhecimento em relação a cada discri-

minador da classe correspondente. Todo discriminador cuja pontuação para a observação seja maior ou igual ao seu limiar de aceitação aprende a observação. Caso nenhum discriminador aceite a nova observação, um novo discriminador é criado para aprendê-la. O limiar de aceitação de cada discriminador é dinâmico: ele cresce proporcionalmente ao número de observações já aprendidas pelo discriminador, a partir de um valor inicial, s, denominado pontuação mínima (*minimum score*), e em função de dado fator de crescimento, γ, denominado período de crescimento do limiar (*threshold growth interval*). Esse mecanismo de limiar dinâmico evita a saturação dos discriminadores, tornando progressivamente mais difícil a incorporação de novas observações a um discriminador que já representa um grande número de padrões [Cardoso et al. 2016]. Opcionalmente, pode-se utilizar um parâmetro adicional D, denominado limite de discriminadores (*discriminator limit*), que estabelece uma quantidade máxima de discriminadores por classe, evitando uma subdivisão excessiva das observações.

Na fase de classificação, uma observação desconhecida é submetida a todos os discriminadores de todas as classes. A resposta de cada discriminador é calculada da mesma forma que na WiSARD clássica, a classe predita será a correspondente ao discriminador com a maior pontuação de reconhecimento. Para lidar com empates entre discriminadores, o mecanismo de *bleaching* pode ser utilizado, de forma análoga à WiSARD original. No presente trabalho, o conjunto de *microclusters* associados a uma mesma classe será denominado *cluster*. Assim, uma ClusWiSARD com k classes é composta por k *clusters*, cada um contendo um número variável de *microclusters* determinado pelo processo de treinamento.

4. Metodologia

Este trabalho propõe o emprego das Redes Neurais sem Pesos (WNN), especificamente os modelos WiSARD e ClusWiSARD, na detecção de programas maliciosos. A Figura 2 ilustra a metodologia proposta por meio de um diagrama de blocos. O sistema recebe como entrada uma imagem representativa do arquivo executável, gerada a partir da conversão do binário original segundo o processo do conjunto de dados MaleVis. Essa imagem possui dimensões de 224×224 *pixels* com três canais de cor (RGB) e profundidade de 8 bits por canal. Após uma etapa de pré-processamento, a imagem alimenta o classificador WNN (WiSARD ou ClusWiSARD), que gera o diagnóstico final quanto à natureza benigna ou maliciosa do software.

A primeira etapa da metodologia compreende o pré-processamento da imagem de entrada, subdividido em três processos. Inicialmente, o sistema converte a imagem para escala de cinza e, em seguida, reduz a resolução para 112×112 *pixels*. Ambas as etapas são opcionais: elas visam diminuir a dimensão das entradas e acelerar tanto o treinamento quanto a operação da rede neural, desde que não comprometam criticamente a acurácia de detecção do sistema.

O terceiro processo, a binarização, adequa a codificação da entrada para o uso com modelos baseados na WiSARD. O projeto adota a técnica de binarização por limiar: o algoritmo compara cada valor representativo do *pixel* com um limiar definido entre 0 e 255. Em imagens RGB, cada *pixel* possui três valores de 8 bits (um por canal); em imagens em escala de cinza, apenas um. O sistema codifica valores superiores ao limiar como 1 e os inferiores como 0. Por fim, o detector trata o valor do limiar como um

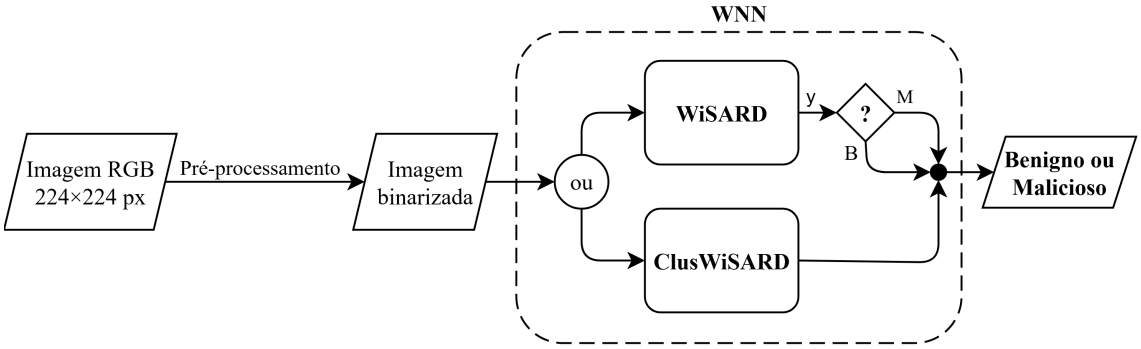

Figura 2. Diagrama em blocos da metodologia proposta. A saída multiclasse da WiSARD é colapsada em uma resposta binária: Benigno (B), se y = Benigno; Malicioso (M), caso contrário. A ClusWiSARD produz diretamente uma resposta binária.

hiperparâmetro, exigindo calibração experimental para maximizar o desempenho.

A segunda etapa da metodologia consiste na classificação da entrada pré-processada por meio de uma rede WiSARD ou ClusWiSARD. Embora ambos os modelos utilizem o mesmo conjunto de dados, seus processos de treinamento diferem. A configuração da WiSARD compreende 26 discriminadores, cada um associado a uma das classes listadas na Tabela 1, provenientes do conjunto MaleVis. A primeira classe representa programas benignos, enquanto as demais abrangem 25 famílias de *malwares* distribuídas em cinco categorias: *adware*, *backdoor*, *trojan*, *virus* e *worm*. Nessa estrutura, a WiSARD atua como um classificador multiclasse, cuja saída alimenta um teste condicional para determinar a classificação final da entrada como maliciosa ou benigna.

Tabela 1. Composição do conjunto de dados MaleVis.

Classe	Categoria	Qtd.	Classe	Categoria	Qtd.
Benignos	—	1832	Win32/Injector	Trojan	495
Win32/Adposhel	Adware	494	Win32/Regrun.A	Trojan	485
Win32/Amonetize	Adware	497	Win32/Snarasite.D!tr	Trojan	500
Win32/BrowseFox	Adware	493	Win32/VBKrypt	Trojan	496
Win32/InstallCore.C	Adware	500	Win32/Vilsel	Trojan	496
Win32/MultiPlug	Adware	499	Win32/Expiro-H	Virus	501
Win32/Neoreklami	Adware	500	Win32/Neshta	Virus	497
Win32/Androm	Backdoor	500	Win32/Sality	Virus	499
Win32/Stantinko	Backdoor	500	VBA/Hilium.A	Virus	500
Win32/Agent-fyi	Trojan	470	Win32/Allaple.A	Worm	478
Win32/Dinwod!rfn	Trojan	499	Win32/AutoRun-PU	Worm	496
Win32/Elex	Trojan	500	Win32/Fasong	Worm	500
Win32/HackKMS.A	Trojan	499	Win32/Hlux!IK	Worm	500

No caso da ClusWiSARD, o sistema associa as observações a apenas dois rótulos durante o treinamento: um para amostras benignas e outro para todas as amostras maliciosas, independentemente da família à qual pertençam. Dessa forma, após o treinamento, a rede compreende apenas dois *clusters* — um vinculado aos programas maliciosos e ou-

tro aos benignos. Portanto, a rede gera uma saída binária, que corresponde diretamente à classificação final da entrada como benigna ou maliciosa.

5. Avaliação Experimental

Esta seção apresenta os experimentos realizados para avaliar a capacidade de detecção de programas maliciosos da metodologia proposta. Na Seção 5.1, descreve-se a configuração dos experimentos, incluindo informações sobre conjunto de dados, métricas e métodos de validação, e estratégia de busca de hiperparâmetros, entre outras. Os resultados dos experimentos são apresentados separados em duas partes. Na Seção 5.2, discutem-se os experimentos convencionais, enquanto a Seção 5.3 trata especificamente dos experimentos de emulação de detecção *zero-day*.

5.1. Configuração dos experimentos

Os experimentos utilizaram o conjunto de dados MaleVis, que compreende 14.226 imagens distribuídas em 26 classes: 25 relativas a diferentes famílias de programas maliciosos e uma referente a programas benignos. A Tabela 1 detalha a distribuição das observações. Visto que a metodologia visa à detecção de programas maliciosos, o cômputo de acertos e erros divide o conjunto em apenas duas categorias: Malicioso e Benigno.

Como o MaleVis resulta de uma compilação arbitrária, ele não representa uma distribuição real de programas. Além disso, o conjunto apresenta um desbalanceamento significativo, com cerca de 6,8 vezes mais observações maliciosas do que benignas. Por esse motivo, a avaliação de desempenho concentrou-se em três métricas: a revocação da classe Malicioso, a revocação da classe Benigno e a acurácia balanceada. A Tabela 2 apresenta as definições formais desses indicadores.

Tabela 2. Definições formais das métricas de avaliação utilizadas.

Métrica	**Definição formal**	**Interpretação**
Revocação — Malicioso (Rec_M)	$\text{Rec}_M = \dfrac{VP_M}{VP_M + FN_M}$	Proporção de observações maliciosas corretamente classificadas
Revocação — Benigno (Rec_B)	$\text{Rec}_B = \dfrac{VP_B}{VP_B + FN_B}$	Proporção de observações benignas corretamente classificadas
Acurácia Balanceada (BA)	$\text{BA} = \dfrac{\text{Rec}_M + \text{Rec}_B}{2}$	Média aritmética das revocações; independente do desbalanceamento

VP_M, VP_B: verdadeiros positivos das classes Malicioso e Benigno. FN_M, FN_B: falsos negativos das classes Malicioso e Benigno.

O protocolo de obtenção das métricas variou conforme o modelo avaliado, embora ambos tenham utilizado a validação cruzada do tipo k-*fold* estratificado com $k = 10$. No caso da WiSARD, para equilibrar os discriminadores da rede, dividiu-se a classe de programas benignos em três segmentos de tamanhos equivalentes. A cada iteração, o procedimento selecionava um desses segmentos e o reunia ao conjunto completo de programas maliciosos; sobre essa união, o algoritmo aplicava o 10-*fold* estratificado. A repetição desse processo para cada um dos três segmentos totalizou três execuções do 10-*fold*.

Visto que o mapeamento das entradas nas tuplas da WiSARD ocorre de forma pseudoaleatória e introduz variabilidade, repetiu-se todo o procedimento para cinco mapeamentos distintos. As métricas reportadas para a WiSARD correspondem, portanto, à média e ao desvio padrão (D.P.) calculados sobre $5 \times 3 \times 10 = 150$ avaliações.

Para a ClusWiSARD, o experimento dispensou a segmentação da classe benigna e aplicou o 10-*fold* estratificado diretamente sobre o conjunto de dados completo. Esse procedimento ocorreu dez vezes, resultando em métricas baseadas na média e no desvio padrão de $10 \times 10 = 100$ avaliações. A avaliação adequada da metodologia exigiu a calibração de diversos parâmetros do sistema: o limiar de binarização, l; e o tamanho de tupla, n. Especificamente para a ClusWiSARD, o ajuste incluiu a pontuação mínima, s; o período de crescimento do limiar, γ; e o limite de discriminadores, D. Ambas as redes operaram com a técnica de *bleaching* permanentemente ativada.

Além disso, a investigação contemplou a melhor configuração de pré-processamento entre três cenários: ausência de conversão e redução; uso de escala de cinza sem redução de resolução; e, finalmente, a aplicação conjunta de escala de cinza e redução de resolução. O método *convert('L')* da biblioteca PIL (Pillow) do Python, em sua versão 12.0.0, realizou a conversão para escala de cinza por meio da transformação *luma* ITU-R 601-2 [Pillow Development Team 2025a]. Para a redução de resolução, o sistema utilizou a técnica de vizinho mais próximo, padrão do método *resize()* da mesma biblioteca [Pillow Development Team 2025b].

A estratégia de calibração dos hiperparâmetros seguiu um protocolo de refinamento sucessivo, para evitar o alto custo computacional de uma busca exaustiva (*grid search*). A acurácia balanceada média foi adotada como critério principal de seleção. Começando pela calibração da WiSARD, fixou-se inicialmente o limiar em $l = 127$ e varreu-se o tamanho de tupla em $n \in \{4; 8; 16; 32; 64\}$. Identificada a região de melhor desempenho em torno de $n = 16$, fixou-se esse valor e varreu-se o limiar em $l \in \{111; 119; 127; 135; 143\}$. Por fim, uma busca com passo fino cobriu simultaneamente $n \in \{14; \ldots; 18\}$ e $l \in \{115; 117; 119; 121; 123\}$. Para a calibração da ClusWiSARD, utilizou-se a melhor combinação de n e l encontrada com a WiSARD, e então, executou-se uma busca entre as combinações de $s \in \{0; 0{,}1; 1\}$, $\gamma \in \{10^3; 10^4; 10^5; 10^6\}$ e $D \in \{5, \ldots, 10\}$.

Por fim, cabe ressaltar que o mapeamento constitui um hiperparâmetro tanto da WiSARD quanto da ClusWiSARD. Como a técnica preconiza a geração pseudoaleatória dessa estrutura, a estratégia de busca pelo melhor mapeamento consiste em repetir os experimentos múltiplas vezes, gerando um novo mapeamento a cada iteração, com os demais parâmetros constantes, para então reter aquele que produz o melhor resultado. Todos os experimentos utilizaram um computador equipado com processador Intel Core i5-7200U (2,50 GHz) e 16 GB de memória RAM. O desenvolvimento dos códigos empregou a linguagem Python 3.10.11 e o módulo *wisardpkg* [Filho et al. 2020] na versão 1.6.3 — uma biblioteca de código aberto para aplicação de modelos baseados na WiSARD.

5.2. Experimentos convencionais

Esta seção apresenta os resultados da avaliação convencional da metodologia, utilizando todas as famílias de programas maliciosos presentes no conjunto de dados tanto no treinamento quanto na validação. A Tabela 3(a) detalha o desempenho da WiSARD com

entradas em escala de cinza e resolução reduzida, sob o primeiro conjunto de hiperparâmetros de teste. Os dados indicam um pico de desempenho em torno de $n = 16$.

Tabela 3. Resultados da WiSARD com imagens 112×112 em escala de cinza: (a) variação de n, com $l = 127$; (b) variação de l, com $n = 16$.

(a)

n	Rec_M (%)		Rec_B (%)		BA (%)	
	Média	**D.P.**	**Média**	**D.P.**	**Média**	**D.P.**
4	94,92	1,57	35,58	9,17	65,25	4,12
8	98,56	0,45	28,16	6,67	63,36	3,33
16	**92,97**	**0,67**	**72,47**	**6,99**	**82,72**	**3,59**
32	94,38	0,62	64,10	7,38	79,24	3,73
64	99,00	0,31	21,49	5,44	60,24	2,70

(b)

l	Rec_M (%)		Rec_B (%)		BA (%)	
	Média	**D.P.**	**Média**	**D.P.**	**Média**	**D.P.**
111	95,83	0,45	78,14	7,01	86,99	3,50
119	**93,92**	**0,64**	**86,63**	**4,78**	**90,28**	**2,38**
127	92,97	0,67	72,47	6,99	82,72	3,59
135	93,62	0,84	63,42	5,95	78,52	2,96
143	95,92	0,66	26,40	5,57	61,16	2,75

Com base nesses resultados iniciais, o experimento avançou para o segundo conjunto de hiperparâmetros, cujas métricas a Tabela 3(b) detalha. Nota-se que a redução do limiar de binarização de 127 para 119 elevou a acurácia balanceada média de $82{,}72\%$ para $90{,}28\%$, configurando o melhor desempenho para esse grupo de combinações. Na etapa final de refinamento, cujos resultados encontram-se na Tabela 4, a combinação $n = 17$ e $l = 121$ alcançou a maior acurácia balanceada média: $90{,}89\% \pm 2{,}22\%$.

Tabela 4. Resultados da WiSARD, com imagens 112×112, em escala de cinza, variando n e l simultaneamente.

l	n	Rec_M (%)		Rec_B (%)		BA (%)	
		Média	**D.P.**	**Média**	**D.P.**	**Média**	**D.P.**
115	14	95,05	0,50	79,50	6,60	87,28	3,28
	15	94,93	0,51	80,72	6,21	87,82	3,11
	16	95,00	0,55	80,95	6,31	87,97	3,17
	17	95,03	0,52	81,05	6,36	88,04	3,19
	18	95,11	0,49	81,01	6,08	88,06	3,04
117	14	94,79	0,51	80,63	6,79	87,71	3,41
	15	94,65	0,56	81,71	6,77	88,18	3,40
	16	94,55	0,53	83,11	5,71	88,83	2,85
	17	94,40	0,58	83,66	5,73	89,03	2,85
	18	94,49	0,62	84,35	5,27	89,42	2,61
119	14	94,47	0,59	83,13	6,23	88,80	3,11
	15	94,12	0,62	85,08	5,43	89,60	2,70
	16	93,92	0,64	86,63	4,78	90,28	2,38
	17	93,71	0,60	87,36	4,60	90,54	2,28
	18	93,45	0,60	87,50	4,81	90,48	2,38
121	14	94,18	0,56	84,85	5,27	89,52	2,62
	15	93,62	0,57	87,08	4,94	90,35	2,45
	16	93,11	0,58	88,49	4,59	90,80	2,32
	17	**92,66**	**0,61**	**89,11**	**4,42**	**90,89**	**2,22**
	18	92,29	0,62	89,05	4,22	90,67	2,11
123	14	94,00	0,61	84,27	5,54	89,14	2,77
	15	93,29	0,66	86,80	4,85	90,05	2,42
	16	92,64	0,63	87,72	4,35	90,18	2,20
	17	92,00	0,63	89,20	4,36	90,60	2,20
	18	91,63	0,65	89,45	3,94	90,54	2,03

Embora diversas combinações de n e l apresentem acurácias balanceadas próximas, elas podem diferir na distribuição do erro entre as classes — aspecto que a acurácia

Tabela 5. Melhor resultado da WiSARD com imagens 224×224: (a) em escala de cinza; (b) coloridas.

(a)

l	n	Rec_M (%)		Rec_B (%)		BA (%)	
		Média	**D.P.**	**Média**	**D.P.**	**Média**	**D.P.**
119	17	89,89	0,68	92,76	3,41	91,33	1,72

(b)

l	n	Rec_M (%)		Rec_B (%)		BA (%)	
		Média	**D.P.**	**Média**	**D.P.**	**Média**	**D.P.**
115	16	90,22	0,69	92,47	3,45	91,35	1,71

balanceada não captura integralmente. Como demonstram os resultados, a revocação da classe Malicioso supera consistentemente a da classe Benigno, indicando um viés sistemático em direção à classe majoritária. Esse viés se manifesta em diferentes graus: na melhor configuração ($l = 121$, $n = 17$), a diferença entre as revocações das duas classes é de apenas $3{,}5$ pontos percentuais, mas se exacerba progressivamente à medida que os hiperparâmetros se distanciam dessa combinação, podendo levar o modelo ao colapso — como ilustra a configuração $l = 127$, $n = 64$, em que $\text{Rec}_M = 99{,}00\%$ e $\text{Rec}_B = 21{,}49\%$.

Após a calibração do limiar de binarização e do tamanho de tupla, os experimentos avaliaram o impacto do pré-processamento no desempenho do detector. As Tabelas 5(a) e 5(b) detalham os melhores resultados da WiSARD sob duas condições: primeiro, ao desativar a redução de resolução das entradas, mas mantendo a conversão para escala de cinza; e, posteriormente, ao desabilitar também essa conversão, processando as entradas em sua forma original. Nota-se que as desativações provocaram aumentos sucessivos da acurácia balanceada média, porém, marginais, além da reversão do viés do modelo. Como o objetivo do sistema é detectar *malwares*, esse resultado não representa uma melhora de desempenho, já que a falha de detecção (entrada maliciosa classificada como benigna) é tipicamente mais grave do que o falso alarme (entrada benigna classificada como maliciosa). Portanto, não há ganhos de desempenho que justifiquem o significativo aumento da carga computacional provocado pela não utilização desses pré-processamentos, de modo que ambos foram mantidos nos experimentos seguintes.

Concluído o procedimento planejado para avaliação da detecção com a WiSARD, prossegue-se com a avaliação da detecção usando a ClusWiSARD. O ponto de partida é a melhor configuração encontrada com a WiSARD: conversão para escala de cinza e redução de resolução ativadas, $l = 121$ e $n = 17$. A Tabela 6(a) apresenta os resultados quando o limite de discriminadores é igual a 9. Observa-se que o melhor resultado ocorre com $\gamma = 10^5$ e $s = 0$, obtendo-se uma acurácia balanceada média de $90{,}96\%$, com um desvio padrão de $1{,}35\%$. Para os outros valores de D, o melhor resultado também ocorreu com a mesma combinação de γ e s, e assim, obtém-se a Tabela 6(b), que compila esses melhores resultados para identificação do valor ideal para o limiar de discriminadores, que é justamente $D = 9$.

Comparando os melhores resultados obtidos com cada rede, observa-se que, em termos de acurácia balanceada, ambas tiveram desempenho semelhante: $90{,}89\%$ para a WiSARD e $90{,}96\%$ para a ClusWiSARD. Considerando os desvios padrão, essa diferença de $0{,}07$ pontos percentuais não pode ser interpretada como uma vantagem expressiva da ClusWiSARD. Contudo, a ClusWiSARD apresenta uma vantagem qualitativa relevante: suas revocações são mais equilibradas entre as classes ($\Delta = 1{,}12\%$), ao passo que a WiSARD exibe uma assimetria mais pronunciada ($\Delta = 3{,}55\%$).

Tabela 6. Resultados da ClusWiSARD: (a) com $D = 9$; (b) com $s = 0$ e $\gamma = 10^5$.

(a)

γ	s	Rec_M (%)		Rec_B (%)		BA (%)	
		Média	**D.P.**	**Média**	**D.P.**	**Média**	**D.P.**
10^3	0	97,30	0,97	61,39	12,45	79,34	5,96
	0,1	97,33	2,00	57,90	19,26	77,62	8,80
	1	94,73	1,37	80,57	5,49	87,65	2,28
10^4	0	90,16	1,06	91,30	2,44	90,73	1,21
	0,1	96,29	1,88	71,52	10,89	83,90	4,67
	1	94,65	1,43	80,97	5,72	87,81	2,33
10^5	**0**	**91,52**	**0,82**	**90,40**	**2,64**	**90,96**	**1,35**
	0,1	96,07	1,74	74,67	8,25	85,37	3,40
	1	94,57	1,23	81,56	5,43	88,06	2,31
10^6	0	98,60	0,42	60,44	3,83	79,52	1,87
	0,1	96,24	1,57	73,68	8,68	84,96	3,71
	1	94,72	1,38	80,64	5,34	87,68	2,21

(b)

D	Rec_M (%)		Rec_B (%)		BA (%)	
	Média	**D.P.**	**Média**	**D.P.**	**Média**	**D.P.**
5	93,05	0,81	87,70	2,48	90,37	1,29
6	92,47	0,79	88,50	2,61	90,49	1,37
7	92,00	0,86	89,34	2,63	90,67	1,40
8	91,81	0,87	89,91	2,78	90,86	1,50
9	**91,52**	**0,82**	**90,40**	**2,64**	**90,96**	**1,35**
10	91,44	0,95	90,20	2,72	90,82	1,47

5.3. Emulação de detecção *zero-day*

Esta seção apresenta os resultados da avaliação da metodologia num cenário emulado de detecção *zero-day*. O protocolo adotado foi o seguinte: para cada uma das 25 classes de programas maliciosos presentes no conjunto de dados, todas as observações pertencentes àquela classe foram removidas do conjunto de dados — sendo essa classe denominada, daqui em diante, **classe emulada**. Sobre o conjunto reduzido resultante, realizou-se a validação cruzada estratificada com 10 *folds*, inserindo, a cada *fold*, as observações da classe emulada no conjunto de validação.

Além das métricas anteriormente definidas, mediu-se a revocação específica para o subconjunto da classe emulada — formalmente definida na Tabela 7. Foram utilizadas as melhores configurações de hiperparâmetros encontradas na avaliação convencional, para cada modelo, incluindo o mapeamento, no caso da WiSARD. Portanto, com esse modelo, a avaliação de cada classe emulada consistiu de 30 repetições: três segmentações do subconjunto de benignos combinadas com os 10 *folds*. Com a ClusWiSARD, foi mantido o mesmo protocolo de antes, com 10 repetições do 10-*fold*, totalizando 100 avaliações por classe emulada.

Tabela 7. Definição formal de Rec_{CE}.

Métrica	**Definição formal**	**Interpretação**
Revocação — Classe Emulada (Rec_{CE})	$\text{Rec}_{CE} = \dfrac{VP_{CE}}{VP_{CE} + FN_{CE}}$	Proporção de amostras da classe emulada corretamente classificadas como Malicioso

VP_{CE}: observações da classe emulada classificadas como Malicioso. FN_{CE}: observações da classe emulada classificadas como Benigno.

A Tabela 8 mostra os intervalos de variação de Rec_M, Rec_B e BA ao longo das 25 iterações, calculadas apenas sobre o subconjunto das classes não removidas. Observa-se que o impacto da remoção de uma classe no desempenho do detector em relação às classes remanescentes é negligenciável para ambos os modelos — os valores permanecem muito próximos aos obtidos na avaliação convencional, sem remoção de nenhuma classe. Esse resultado indica que nenhuma classe exerce influência dominante sobre o treinamento, e

Tabela 8. Variação das métricas de desempenho sobre as famílias remanescentes ao longo das iterações do experimento de emulação de detecção *zero-day*.

Modelo	Rec_M / Rec_B / BA (média, %)		
	Rec_M	Rec_B	BA
WiSARD — avaliação convencional	92,5	89,5	91,0
WiSARD — emulação *zero-day*	[92,1; 93,7]	[89,5; 90,6]	[90,8; 92,1]
ClusWiSARD — avaliação convencional	91,5	90,4	91,0
ClusWiSARD — emulação *zero-day*	[90,5; 91,9]	[90,1; 92,8]	[90,5; 92,1]

que o desempenho geral do detector é robusto à ausência de qualquer classe individual no conjunto de treinamento.

Os resultados da emulação de detecção *zero-day* para cada uma das classes de programas maliciosos são apresentados na Tabela 9. Nota-se que, para a maioria das classes, houve pouca diferença entre o desempenho da WiSARD e da ClusWiSARD. Nos casos em que a diferença foi expressiva, por vezes a WiSARD saiu-se melhor, por outras, a ClusWiSARD, sem uma predominância clara de um modelo sobre o outro.

Tabela 9. Emulação de detecção *zero-day* com as melhores configurações da WiSARD e da ClusWiSARD.

Classe emulada	Categoria	ClusWiSARD		WiSARD	
		Rec_{CE} (%)		Rec_{CE} (%)	
		Média	D.P.	Média	D.P.
Win32/Amonetize	Adware	94,0	9,8	88,0	19,9
Win32/Allaple.A	Worm	90,7	3,1	78,9	1,9
Win32/Dinwod!rfn	Trojan	89,5	4,8	81,6	3,4
Win32/Regrun.A	Trojan	80,1	6,8	47,0	7,0
Win32/InstallCore.C	Adware	78,6	12,1	96,5	2,8
Win32/MultiPlug	Adware	63,9	6,0	59,6	1,9
Win32/HackKMS.A	Trojan	63,3	16,0	65,2	0,2
Win32/Adposhel	Adware	62,5	2,7	65,4	2,7
Win32/Snarasite.D!tr	Trojan	60,8	31,7	100,0	0,0
Win32/AutoRun-PU	Worm	60,0	13,0	54,1	9,7
Win32/Elex	Trojan	59,0	6,2	33,8	8,0
Win32/Agent-fyi	Trojan	54,7	2,9	55,8	1,4
Win32/Androm	Backdoor	53,4	1,3	4,7	0,7
Win32/Injector	Trojan	49,9	5,2	57,8	9,5
Win32/Expiro-H	Vírus	48,0	4,7	67,6	3,0
Win32/VBKrypt	Trojan	48,0	13,7	55,0	9,8
Win32/BrowseFox	Adware	44,9	1,6	40,5	0,4
Win32/Neshta	Vírus	36,2	3,1	28,9	0,9
Win32/Sality	Vírus	35,4	2,9	46,2	1,6
Win32/Stantinko	Backdoor	34,9	7,4	12,3	1,7
Win32/Neoreklami	Adware	15,5	1,5	16,9	0,4
Win32/Fasong	Worm	6,6	5,0	0,7	1,6
Win32/Vilsel	Trojan	0,4	0,1	0,2	0,0
Win32/Hlux!IK	Worm	0,2	0,0	0,2	0,1
VBA/Hilium.A	Vírus	0,0	0,0	0,0	0,0

Observa-se uma ampla variação na revocação entre as diferentes classes, sugerindo que a capacidade de generalização depende fortemente das características visuais de

cada família. Classes com alta revocação — como Win32/Amonetize, Win32/Allaple.A e Win32/Dinwod!rfn — possivelmente compartilham padrões visuais com outras famílias vistas no treinamento. No extremo oposto, classes com revocação próxima de zero — como Win32/Vilsel, Win32/Hlux!IK e VBA/Hilium.A — apresentam representações visuais suficientemente distintas das demais para que o modelo seja incapaz de generalizá-las. Classes com revocação intermediária corresponderiam a um cenário de ambiguidade visual, resultando em desempenho próximo ao de uma classificação aleatória.

6. Conclusão

Este trabalho apresentou um estudo exploratório da aplicação das arquiteturas WiSARD e ClusWiSARD à detecção de programas maliciosos baseada em visualização de binários como imagens, utilizando o conjunto de dados MaleVis. A metodologia proposta converte as imagens RGB do conjunto para escala de cinza e as redimensiona para 112×112 *pixels* antes de submetê-las às redes neurais sem pesos. O desempenho dos modelos foi avaliado sob duas perspectivas: o cenário convencional, em que todas as famílias do conjunto de dados participam do treinamento, e o cenário de emulação de *zero-day*, em que famílias inteiras são omitidas do treinamento para simular o advento de um novo *malware*. Em ambos os cenários, foram adotadas validação cruzada k*-fold* e estratégias para mitigar o efeito do desbalanceamento de classes.

No cenário convencional, a WiSARD atingiu $90{,}89\%$ de acurácia balanceada média, com revocação de $92{,}66\%$ para a classe Malicioso e $89{,}11\%$ para a classe Benigno. A ClusWiSARD obteve acurácia balanceada média de $90{,}96\%$, com revocações de $91{,}52\%$ para Malicioso e $90{,}4\%$ para Benigno — resultados similares aos da WiSARD em termos de acurácia, porém com distribuição mais equilibrada entre as classes. No cenário de emulação de detecção *zero-day*, a remoção de qualquer uma das 25 famílias do conjunto de treinamento não impactou significativamente o desempenho do detector sobre as famílias remanescentes, cujas métricas permaneceram próximas às obtidas na avaliação convencional. A revocação da classe emulada variou amplamente entre as 25 famílias avaliadas: 10 famílias (40%) tiveram revocação superior a 60% em ao menos um dos modelos; 8 famílias (32%) apresentaram revocação entre 40% e 60%; e 7 famílias (28%) obtiveram revocação inferior a 40% em ambos os modelos. Esse padrão sugere que a capacidade de generalização para *malware* desconhecido depende fortemente do grau de semelhança visual entre a família omitida e as demais presentes no treinamento.

Os resultados obtidos indicam que a aplicação de redes neurais sem pesos à detecção de *malware* baseada em imagem é uma direção de pesquisa promissora, motivando investigações futuras. No cenário convencional, destacam-se as seguintes direções: explorar estratégias de pré-processamento que realcem os padrões visuais distintivos de cada classe; investigar *ensembles* de modelos WNN com tamanhos de tupla distintos; avaliar a metodologia em outros conjuntos de dados baseados em imagem; e comparar sistematicamente as WNN com modelos de aprendizado profundo em termos de acurácia e custo computacional. No cenário *zero-day*, uma direção interessante é explorar a capacidade de aprendizado *online* das WNN em conjunto com a cronologia de surgimento das famílias de *malware*, avaliando os modelos na ordem histórica em que as ameaças emergiram.

Referências

Aleksander, I., Gregorio, M. D., França, F., Lima, P., and Morton, H. (2009). A brief introduction to weightless neural systems. In *17th European Symposium on Artificial Neural Networks (ESANN)*, pages 299–305.

Aleksander, I., Thomas, W., and Bowden, P. (1984). WISARD: A radical step forward in image recognition. *Sensor Review*, 4(3):120–124.

Bacellar, A. T. L., Susskind, Z., Breternitz Jr., M., John, E., John, L. K., Lima, P. M. V., and França, F. M. G. (2024). Differentiable weightless neural networks. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *PMLR*.

Bavishi, S. and Modi, S. (2024). Accelerating malware classification: A vision transformer solution. Disponível em: `https://arxiv.org/abs/2409.19461`. Acesso em: 08 maio 2026.

Bledsoe, W. W. and Browning, I. (1959). Pattern recognition and reading by machine. In *IRE-AIEE-ACM Computer Conference*, IRE-AIEE-ACM '59 (Eastern), pages 225–232. Association for Computing Machinery.

Bozkir, A. S., Cankaya, A. O., and Aydos, M. (2019). Utilization and comparison of convolutional neural networks in malware recognition. In *2019 27th Signal Processing and Communications Applications Conference (SIU)*, pages 1–4.

Brezinski, K. and Ferens, K. (2023). Metamorphic malware and obfuscation: A survey of techniques, variants, and generation kits. *Security and Communication Networks*, 2023:8227751.

Cardoso, D. O., Carvalho, D., Alves, D. S. F., de Souza, D. F. P., Carneiro, H. C. C., Pedreira, C. E., Lima, P. M. V., and França, F. M. G. (2016). Financial credit analysis via a clustering weightless neural classifier. *Neurocomputing*, 183:70–78.

Carvalho, D., Carneiro, H., França, F., and Lima, P. (2013). B-bleaching: Agile overtraining avoidance in the WiSARD weightless neural classifier. In *21st European Symposium on Artificial Neural Networks (ESANN)*.

Comar, P. M., Liu, L., Saha, S., Tan, P.-N., and Nucci, A. (2013). Combining supervised and unsupervised learning for zero-day malware detection. In *Proceedings of IEEE INFOCOM*, pages 2022–2030.

Egele, M., Scholte, T., Kirda, E., and Kruegel, C. (2012). A survey on automated dynamic malware-analysis techniques and tools. *ACM Computing Surveys*, 44(2).

Filho, A. S. L., Guarisa, G. P., Filho, L. L., Oliveira, L. F. R., França, F. M. G., and Lima, P. M. V. (2020). wisardpkg — a library for WiSARD-based models. Disponível em: `https://arxiv.org/abs/2005.00887`. Acesso em: 08 maio 2026.

Gopinath, M. and Sethuraman, S. C. (2023). A comprehensive survey on deep learning based malware detection techniques. *Computer Science Review*, 47:100529.

Guo, Y. (2023). A review of machine learning-based zero-day attack detection: Challenges and future directions. *Computer Communications*, 198:175–185.

HUMIR Lab (2019). MaleVis: A dataset for vision based malware recognition. Disponível em: `https://web.cs.hacettepe.edu.tr/~selman/malevis/`. Acesso em: 08 maio 2026.

Kappaun, A., Camargo, K., Rangel, F., Firmino, F., Lima, P. M. V., and Oliveira, J. (2016). Evaluating binary encoding techniques for WiSARD. In *2016 5th Brazilian Conference on Intelligent Systems (BRACIS)*, pages 103–108.

Kim, J.-Y., Bu, S.-J., and Cho, S.-B. (2018). Zero-day malware detection using transferred generative adversarial networks based on deep autoencoders. *Information Sciences*, 460–461:83–102.

Nataraj, L. (2015). *A Signal Processing Approach to Malware Analysis*. PhD thesis, University of California, Santa Barbara.

Pillow Development Team (2025a). PIL.Image.Image.convert() — Pillow 12.0.0 documentation. Disponível em: `https://pillow.readthedocs.io/en/v12.0.0/reference/Image.html#PIL.Image.Image.convert`. Acesso em: 08 maio 2026.

Pillow Development Team (2025b). PIL.Image.Image.resize() — Pillow 12.0.0 documentation. Disponível em: `https://pillow.readthedocs.io/en/v12.0.0/reference/Image.html#PIL.Image.Image.resize`. Acesso em: 08 maio 2026.

Quertier, T., Marais, B., Barrué, G., Morucci, S., Azé, S., and Salladin, S. (2024). A lean transformer model for dynamic malware analysis and detection. Disponível em: `https://arxiv.org/abs/2408.02313`. Acesso em: 08 maio 2026.

Ramos, L., Filho, L. L., França, F., and Lima, P. (2020). Detecção estática e dinâmica de malwares usando redes neurais sem peso. In *Anais do XX Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*, pages 369–381, Porto Alegre, RS, Brasil. SBC.

Rey, V., Sánchez, P. M. S., Celdrán, A. H., and Bovet, G. (2022). Federated learning for malware detection in IoT devices. *Computer Networks*, 204:108693.

Salas, M. P. and de Geus, P. L. (2024). Deep learning applied to imbalanced malware datasets classification. *Journal of Internet Services and Applications*, 15(1):342–359.

Susskind, Z., Arora, A., Miranda, I. D. S., Bacellar, A. T. L., Villon, L. A. Q., Katopodis, R. F., de Araújo, L. S., Dutra, D. L. C., Lima, P. M. V., França, F. M. G., Breternitz Jr., M., and John, L. K. (2023). ULEEN: A novel architecture for ultra-low-energy edge neural networks. *ACM Transactions on Architecture and Code Optimization*, 20(4).

Susskind, Z., Arora, A., Miranda, I. D. S., Villon, L. A. Q., Katopodis, R. F., Araújo, L. S. D., Dutra, D. L. C., Lima, P. M. V., França, F. M. G., Breternitz, M., and John, L. K. (2022). Weightless neural networks for efficient edge inference. In *Proceedings of the 31st International Conference on Parallel Architectures and Compilation Techniques (PACT'22)*.

Ucci, D., Aniello, L., and Baldoni, R. (2019). Survey of machine learning techniques for malware analysis. *Computers & Security*, 81:123–147.