electronics

MDPI

Article

SMAD: Semi-Supervised Android Malware Detection via Consistency on Fine-Grained Spatial Representations

Suchul Lee and Seokmin Han *

Department of Railroad Data Science, Korea National University of Transportation, Uiwang-si 16106, Gyeonggi-do, Republic of Korea; sclee@ut.ac.kr
* Correspondence: seokmin.han@ut.ac.kr

Abstract

Malware analytics suffer from scarce, delayed, and privacy-constrained labels, limiting fully supervised detection and hampering responsiveness to zero-day threats. We propose SMAD, a Semi-supervised Android Malicious App Detector that integrates a segmentation-oriented backbone—to extract pixel-level, multi-scale features from APK imagery—with a dual-branch consistency objective that enforces predictive agreement between two parallel branches on the same image. We evaluate SMAD on CICMalDroid2020 under label budgets of 0.5, 0.25, and 0.125 and show that it achieves higher accuracy, macro-precision, macro-recall, and macro-F1 with smoother learning curves than supervised training, a recursive pseudo-labeling baseline, a FixMatch baseline, and a confidence-thresholded consistency ablation. A backbone ablation (replacing the dense encoder with WideResNet) indicates that pixel-level, multi-scale features under agreement contribute substantially to these gains. We observe a coverage–precision trade-off: hard confidence gating filters noise but lowers early-training performance, whereas enforcing consistency on dense, pixel-level representations yields sustained label-efficiency gains for image-based malware detection. Consequently, SMAD offers a practical path to high-utility detection under tight labeling budgets—a setting common in real-world security applications.

Keywords: semi-supervised learning (SSL); Android malware detection; consistency regularization; segmentation-based pixel-level features; APK-to-image representation

check for updates

Academic Editors: Grzegorz Dudek and Arkadiusz Tomczyk

Received: 19 October 2025
Revised: 25 October 2025
Accepted: 27 October 2025
Published: 30 October 2025

Citation: Lee, S.; Han, S. SMAD: Semi-Supervised Android Malware Detection via Consistency on Fine-Grained Spatial Representations. *Electronics* **2025**, *14*, 4246. https://doi.org/10.3390/electronics14214246

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Advances in artificial intelligence (AI), propelled by information technology (IT), have profoundly reshaped numerous domains. AI has transformed information systems through heightened automation, predictive capability, decision accuracy, and efficiency, enabling widespread innovations in service delivery across applications such as large-scale language models (LLMs) [1], healthcare [2], edge computing [3], smart cities [4], and cybersecurity [5].

AI methods increasingly learn fluid decision boundaries by modeling underlying semantics rather than relying on surface terms in classification problems [5]. In cybersecurity, this boundary typically separates malicious from benign software and is learned using machine learning (ML) paradigms, including supervised, unsupervised, and reinforcement learning. Supervised learning fits labeled inputs to designated outputs [5], whereas unsupervised learning discovers latent structure in unlabeled data, grouping samples without prior categories [6].

Within cybersecurity, AI-based malware detection is increasingly employed to advance automated identification beyond traditional static and dynamic analyses. Yet these

approaches often face accuracy limitations—most notably high false-positive and false-negative rates—that impede practical deployment. Consequently, supervised learning remains a preferred strategy for known threats because it explicitly models input–label relationships and delivers more reliable performance [7,8]. Empirical studies indicate that supervised models trained on well-labeled datasets outperform unsupervised methods on structured detection tasks such as malware classification and intrusion detection [9].

However, the effectiveness of supervised learning is tightly coupled to access to large, high-quality labeled corpora—resources that are especially difficult to obtain in security contexts. Label creation demands expert-level effort, including reverse engineering, behavioral profiling, and the curation of threat intelligence [10]. The continual emergence of zero-day malware further complicates labeling and detection [11]; previously unseen threats often evade conventional defenses and cannot be accurately labeled at the time of appearance, undermining methods that rely on known classes. Privacy and confidentiality constraints also restrict data sharing, further hampering the construction of labeled datasets.

Within this landscape, semi-supervised learning (SSL) has emerged as a pragmatic response to label scarcity in security and privacy (S&P). Canonical methods—Mean Teacher [12], MixMatch [13], ReMixMatch [14], UDA [15], FixMatch [16], and large-scale self-training via Noisy Student [17]—operationalize consistency regularization and pseudo-labeling at scale, while highlighting evaluation pitfalls and confirmation-bias risks [18,19]. In cybersecurity, SSL has delivered gains across phishing, malware, intrusion, and encrypted-traffic detection [20]; representative instances include SSL-based Android malware detection leveraging labeled and unlabeled samples [21], multimodal frameworks that combine Gated Recurrent Units (GRUs) and Graph Convolutional Networks (GCNs) for encrypted traffic under limited labels [22], continual SSL that adapts to evolving malware without full retraining [23], and retrieval-augmented few-shot classification (MalMixer) [24]. These results motivate our focus on dense, pixel-level representations and dual-branch consistency for image-based Android malware detection. A concise review of these SSL foundations and their security applications appears in Section 2.

Building on these insights, we present SMAD (Semi-supervised Android Malicious App Detector), a framework designed to mitigate the dependence on costly annotations while improving resilience to previously unseen (zero-day) malware. SMAD integrates a segmentation-oriented backbone that extracts pixel-level, multi-scale features from APK imagery with a dual-branch consistency objective that enforces predictive agreement between two parallel branches on the same image. This combination leverages the structure present in abundant unlabeled telemetry to stabilize optimization under label scarcity and to enhance generalization when family distributions drift.

We evaluate SMAD on CICMalDroid2020 under label budgets $\texttt{rol} \in \{0.5, 0.25, 0.125\}$ and observe consistent gains in accuracy, macro-precision, macro-recall, and macro-F1 over supervised training and a recursive pseudo-labeling baseline, with smoother learning curves across epochs. We also include a controlled comparison to FixMatch [16] under the same schedule and a backbone ablation (replacing our backbone with WideResNet) that attributes a substantial portion of the gains to dense, pixel-level multi-scale features under agreement. An ablation with a fixed confidence gate clarifies the coverage–precision trade-off under a cold start.

This paper makes three contributions:

1. A semi-supervised detector with segmentation-derived pixel-level features. We introduce SMAD, which couples dual-branch consistency with a segmentation-oriented backbone that extracts dense, pixel-level, multi-scale representations from APK imagery via an Atrous Spatial Pyramid Pooling (ASPP) module. These features combined with agreement between parallel branches provide higher-SNR unsupervised

targets—improving early calibration, training stability, and label efficiency under label scarcity.
2. Improved robustness to unknown behaviors. By enforcing agreement between two branches processing the same APK image, SMAD exhibits enhanced generalization to previously unseen malware families without additional expert labels.
3. Controlled and balanced evaluation. We conduct extensive experiments and report accuracy, macro-precision, macro-recall, and macro-F1 with mean $\pm$ std over three runs; compare against FixMatch under the same schedule; and perform a backbone ablation (dense encoder vs. WideResNet) to isolate encoder effects.

Section 2 reviews related work on semi-supervised security analytics and malware detection. Section 3 details the SMAD architecture, loss functions, and design rationales. Section 4 describes datasets, setup, and metrics. Section 5 reports results and discusses operational implications. Section 6 concludes.

2. Related Work

2.1. Motivation and Theoretical Foundations

The increasingly complex and dynamic cyber-threat landscape renders purely supervised approaches impractical: collecting and annotating high-quality security data (e.g., malware corpora, encrypted traffic, intrusion logs) is costly, slow, and quickly outdated under zero-day threats and concept drift. Semi-supervised learning (SSL) addresses these constraints by leveraging abundant unlabeled data alongside minimal labels to sustain performance. A comprehensive survey [20] documents growing SSL adoption in phishing, network intrusion, web spam, and malware detection. Representative frameworks include Dapper [25], which attains near-supervised accuracy with $\sim$10% labels via pseudo-label propagation and automated hyperparameter selection, and SF-IDS [26], which couples pseudo-label filtering with hybrid losses to mitigate label scarcity and class imbalance.

Underpinning these successes are canonical SSL methods that operationalize consistency regularization and pseudo-labeling at scale: Mean Teacher stabilizes targets via weight-averaged teachers [12]; MixMatch unifies consistency and entropy minimization with mixup [13]; ReMixMatch augments this with distribution alignment and augmentation anchoring [14]; UDA enforces agreement under strong augmentations [15]; FixMatch combines confidence-thresholded pseudo-labels with weak/strong augmentation [16]; and Noisy Student demonstrates scalable self-training [17]. These developments rest on classical principles—entropy minimization and manifold regularization [27,28]—and are informed by cautions on evaluation protocol and confirmation-bias dynamics [18,19].

2.2. SSL for Malware and Encrypted-Traffic Detection

SSL has been actively adapted to security subdomains with promising results. For Android malware, Memon et al. [21] implement a feature-based SSL model over permissions and API logs, achieving robust detection with limited labels. In encrypted traffic analysis, multimodal architectures that combine Gated Recurrent Units (GRUs) with Graph Convolutional Networks (GCNs) improve F1 under low-label regimes [22]. Continual SSL enables adaptation to evolving malware families without full retraining [23]. Retrieval-augmented SSL (MalMixer) supports few-shot malware family classification [24], while bidirectional normalizing-flow models reduce reliance on labeled anomalies [29]. Semi-supervised traffic clustering (e.g., SCOUT) can isolate malicious flows for downstream signature generation, and large-scale intrusion-detection pipelines report gains on CIC-DDoS2019 and UNSW-NB15 via self-training and co-training strategies [30].

2.3. Advanced SSL Architectures and Emerging Directions

Recent directions underscore practicality and robustness in real deployments. Multi-stage designs like M3S-UPD deliver fine-grained encrypted-traffic classification and zero-shot detection through continual learning [31]. Contrastive and multimodal pretraining further enhances generalization to unseen attack patterns in encrypted traffic [32]. Interpretable SSL systems for industrial cyber-attack detection improve transparency and trust [33,34]. In wireless-sensor-network (WSN) intrusion detection, pseudo-label-based SSL achieves high F1 in label-scarce settings [9]. Collectively, these advances chart a path toward robust, explainable, and deployable SSL for modern cybersecurity.

Within this landscape, image-based malware analytics are appealing because APK-to-image renderings expose narrow, local artifacts—padding bands, packing/obfuscation traces, and layout regularities—without hand-crafted features [35]. Patch-token pipelines (e.g., ViT) summarize content into coarse tokens and emphasize global relations [36], which can blur such artifacts; in contrast, segmentation-oriented encoders retain dense, multi-scale pixel-wise descriptors that preserve fine textures and local discontinuities [37], naturally aligning with consistency-style SSL objectives [12,16].

Following this rationale, we use a segmentation-oriented backbone to extract dense, multi-scale features from APK imagery and apply dual-branch consistency on the same image with a decoder-free, image-level classifier. Preserving pixel-wise detail and enforcing agreement at the artifact's spatial granularity stabilizes predictions under benign spatial/photometric shifts, providing a concise rationale for robustness to padding alignment changes and common obfuscation in APK-to-image renderings [35].

3. Semi-Supervised Android Malware Detection

In this study, we introduce SMAD (Semi-supervised Android Malicious App Detector), a semi-supervised learning (SSL) framework for detecting malicious Android applications. Rather than depending exclusively on large, fully labeled datasets, SMAD leverages unlabeled data to maintain high detection accuracy in environments where threat patterns evolve rapidly. By integrating a segmentation-oriented backbone for rich feature extraction with a two-branch consistency strategy for stable semi-supervised training, SMAD achieves strong generalization to both known and emerging attack types.

3.1. Architecture

SMAD is built upon a modified DeepLabV3+ [37] backbone originally developed for semantic segmentation tasks. Conventional image classification networks employ a stack of convolutional layers followed by global pooling and fully connected layers. In contrast, DeepLabV3+ consists of a backbone feature extractor, an Atrous Spatial Pyramid Pooling (ASPP) module for multi-scale context aggregation, and a decoder module for pixel-wise segmentation. To use DeepLabV3+ for image-level classification rather than pixel-wise segmentation, we retain its multi-scale encoder unchanged and modify only the task-specific heads, as summarized below and illustrated in Figure 1.

Shared encoder (unchanged from DeepLabV3+). In the proposed scheme, we reuse the DeepLabV3+ encoder as is—a ResNet backbone followed by an ASPP module. ASPP comprises four branches (one 1×1 convolution and three 3×3 atrous convolutions with dilation rates 6, 12, and 18 at output stride 16), aggregating multi-scale context while preserving pixel-level detail. The encoder outputs a dense feature map that is shared by both SMAD's dual-branch heads; only the task-specific heads depart from the standard DeepLabV3+ pipeline.

Head change from DeepLabV3+ to the proposed classifier. Relative to the DeepLabV3+ segmentation head—which upsamples and refines features for pixel-wise prediction—the

proposed classifier removes the decoder and adopts a lightweight path: global average pooling (256 → 1D) → fully connected layer, producing image-level logits. This shifts from spatial preservation (dense masks) to spatial collapse (a compact global representation) while retaining the encoder's multi-scale evidence.

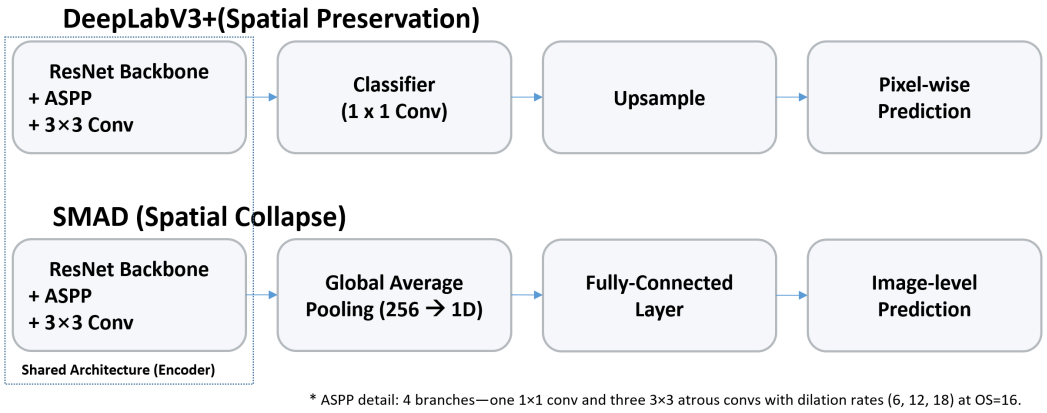


Figure 1. Architecture comparison between DeepLabV3+ and the proposed method: both share the same encoder (ResNet + ASPP), while the proposed classifier replaces the segmentation decoder with a global average pooling (256 → 1D) and a fully connected layer for image-level logits.

The motivation for adopting DeepLabV3+ is its ability to expand the receptive field and capture multi-scale context via atrous convolutions—capabilities uncommon in conventional classification models. This allows the network to fuse fine-grained local features with global contextual information, which is advantageous for classification tasks with substantial variation in object scale and spatial arrangement. In addition, reusing a segmentation-oriented backbone promotes dense feature representation learning, potentially yielding improved generalization over standard classification backbones.

Dual-branch layout and inference fusion. Figure 2 presents the SMAD network architecture. Given an image input x, a high-capacity backbone extracts multi-scale features that capture both global structure and fine-grained cues. The features are processed by two parallel semi-supervised branches on the same input; consistency between branches promotes better generalization.

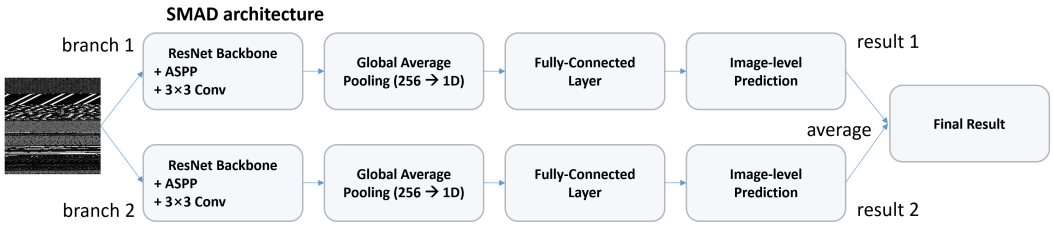


Figure 2. The network adopts a dual-branch semi-supervised architecture: two parallel branches process the same input, and inter-branch agreement is enforced between their predictions to promote robust generalization under limited labels.

Both branches are architecturally identical and route the shared encoder features directly to the same classifier head; we keep two instances solely to apply inter-branch consistency during training, and average their predictions at inference. The per-branch predictions are fused to produce the final decision. Training jointly minimizes a supervised loss L_{sup} on labeled samples and a consistency loss L_{con} on unlabeled samples, aligning the two branches' predictions. This design encourages representation learning via inter-branch agreement, mitigates overfitting under label scarcity, and yields robust performance against perturbations and evolving attack strategies, enabling reliable detection of both known

threats and zero-days. The training objectives that enforce inter-branch agreement are mathematically defined in Section 3.2.

Although the two branches share the same architecture, we maintain the asymmetry needed for effective consistency regularization via two mechanisms: (i) distinct random initializations and (ii) explicitly decoupled parameters that are optimized independently by stochastic gradient descent (SGD). This induces immediate and sustained divergence in parameter space while preserving comparable capacity across branches, thereby ensuring that the inter-branch agreement term remains informative.

3.2. Training Objectives

Using two parallel branches on the same input, the model aims to improve prediction stability and generalization. The training set is partitioned into labeled and unlabeled subsets.

Let $m = 1 \ldots M$ index images and $n = 1 \ldots N$ pixels. The model uses two branches, $i \in \{1, 2\}$. For labeled inputs, y^l_{mn} denotes the one-hot ground truth at pixel n of image m, and $\hat{y}^l_{mn,i}$ the class-probability prediction from branch i. For unlabeled inputs, $\hat{y}^u_{mn,i}$ denotes the prediction from branch i.

Let C denote the number of classes with $c \in \{1, \ldots, C\}$ indexing classes. We use the standard pixel-wise categorical cross-entropy $\mathrm{L}_{ce}(\mathrm{p}, \mathrm{q}) = -\sum_{c=1}^{C} p_c \log(q_c)$; here, $\mathrm{p} = (p_c)$ and $\mathrm{q} = (q_c)$ are class-probability vectors of length C (non-negative entries summing to 1). In implementation, L_{ce} is computed from logits for numerical stability (e.g., softmax-with-logits). For the labeled subset, ground-truth labels supervise the network via the supervised loss L_{sup} defined as:

$$L_{sup,i} = \frac{1}{M} \sum_{m=1}^{M} \frac{1}{N} \sum_{n=1}^{N} \mathrm{L}_{ce}(y^l_{mn}, \hat{y}^l_{mn,i}); \tag{1}$$

$$L_{sup} = \frac{1}{2} \cdot (L_{sup,1} + L_{sup,2}). \tag{2}$$

To improve training stability, we incorporate an additional consistency loss constraint [38], defined as follows:

$$L_{con} = \frac{1}{M} \sum_{m=1}^{M} \frac{1}{N} \sum_{n=1}^{N} \mathrm{L}_{ce}(\hat{y}^u_{mn,1}, \hat{y}^u_{mn,2}). \tag{3}$$

Additionally, we can incorporate branch-specific confidence weighting into the consistency term (Equation (3)), using $c^{(1)}$ and $c^{(2)}$ for subnet 1 and subnet 2, respectively. For each unlabeled example x, the loss contributes only when both branches report confidence above a preset threshold τ. Formally,

$$L_{\text{con_thr}} = \frac{1}{M} \sum_{m=1}^{M} \frac{1}{N} \sum_{n=1}^{N} \mathbf{1}\left[c^{(1)}(x) > \tau \wedge c^{(2)}(x) > \tau\right] \mathrm{L}_{ce}(\hat{y}^u_{mn,1}, \hat{y}^u_{mn,2}), \tag{4}$$

where $\mathbf{1}[\cdot]$ is the indicator function. In what follows, we refer to this scheme as *SMAD-THR*. Unless stated otherwise, SMAD uses a DeepLabV3+ backbone. We also consider SMAD$^{\mathrm{W}}$, where "W" indicates a WideResNet backbone; all other components and training settings remain identical.

Combining the supervised loss constraint for labeled data and consistency loss for unlabeled data, the final loss constraint is defined as follows:

$$L_{final} = \lambda_{sup} L_{sup} + \lambda_{con} L_{con} \tag{5}$$

where λ_{sup} and λ_{con} are empirically tuned balancing coefficients. Unless otherwise stated, we set $\lambda_{sup} = 1$ and $\lambda_{con} = 1$ for all experiments. This constant-weight setting follows

common SSL practice—e.g., FixMatch [16] uses a fixed unlabeled-loss weight λ_u and reports that ramping this weight is unnecessary—so adopting 1:1 weighting avoids confounding from an additional hyperparameter while keeping comparisons fair across backbones and baselines. Exhaustive fine-tuning of the λ coefficients is outside the scope of this work; our focus is on the method design and label-budget regime rather than hyperparameter optimization.

4. Experiments

4.1. Setup

Datasets. We conduct experiments on the CICMalDroid 2020 dataset [39], a publicly available collection of Android Package Kit (APK) files—standard Android application bundles—compiled from sources such as VirusTotal and the Contagio blog (December 2017–December 2018).

The dataset comprises 17,341 applications labeled as Benign or one of four malware families (Adware, Banking, Mobile Riskware, SMS); in our image-rendered subset of 16,787 samples, the per-class counts are Benign 4039 (24.1%), Adware 1514 (9.0%), Banking 2505 (14.9%), SMS 4821 (28.7%), and Mobile Riskware 3908 (23.3%). We map APK bytes to a grayscale image using a simple row-major stream order (left-to-right, top-to-bottom); for the stream-order procedure and implementation details, see [40].

For evaluation, we perform a single class-stratified partition of the full corpus into an 8:1:1 training–validation–test split, preserving family proportions with a fixed random seed. All metrics are reported on the held-out test set; the validation split is used exclusively for model selection and early stopping. The semi-supervised label budget is parameterized by `rol` and applies only to the training split: for a given `rol`$\in \{0.5, 0.25, 0.125\}$, that fraction of the training samples is treated as labeled, while the remainder of the training data is available as unlabeled input to SSL methods (the validation and test sets are never used for unsupervised updates).

Input preprocessing. To standardize inputs and reduce overfitting, we apply a single, shared transform per image before it is fed to both branches: (i) random scaling with a factor $s \in [0.5, 2.0]$ while preserving aspect ratio (bilinear resampling), (ii) zero-padding if needed followed by a random square crop, (iii) random horizontal flip with probability 0.5, and (iv) uniform resize to 224×224.

Implementation Platform. PyTorch [41] is widely adopted in contemporary research owing to its dynamic computation graph, mature ecosystem and community support, and efficient, scalable Python implementation. Leveraging these properties, all experimental code for this study was implemented in PyTorch 2.9.0.

Performance metrics. We report accuracy together with macro-precision, macro-recall, and macro-F1 on the held-out test set. Macro-averaging computes each metric per class and then averages them with equal weight, mitigating skew from class imbalance.

Hardware & training details. All experiments were run on a single workstation with one NVIDIA RTX 4090 (24 GB VRAM), an Intel Core i7-13700KF CPU, and 64 GB RAM. Unless otherwise noted, we use stochastic gradient descent (SGD) as the optimizer, with a batch size of 4 and an initial learning rate of 0.005.

4.2. Compared Schemes

We evaluate the following baselines and ablations alongside our semi-supervised consistency method.

Supervised only (*SUP*). Train only on the labeled subset with cross-entropy L_{ce} (Section 3.2); unlabeled data are not used for training or model selection. This serves as a conventional lower-bound reference.

Recursive pseudo-labeling (*REC*). Train on labeled data, infer pseudo-labels on unlabeled data with the current model, accept all pseudo-labels (no confidence threshold), and retrain on the union; the process may be repeated. This simple loop can accumulate label errors across iterations.

FixMatch. A popular semi-supervised baseline using confidence-thresholded pseudo-labels under weak/strong views. We adopt the same backbone (WideResNet), optimizer, and schedule as ours for a controlled comparison.

Confidence-threshold ablation (*SMAD-THR*). Our method with a fixed confidence gate inside the consistency term (Section 3.2) was used to study the effect of gating under limited labels.

Backbone ablation (SMAD^{W}). Replace the segmentation-oriented encoder with a WideResNet while keeping the SSL objective and schedule fixed. Motivated by the canonical FixMatch recipe, we use a WideResNet encoder here so that the ablation aligns with that backbone and supports apples-to-apples cross-method comparison. This isolates the encoder's contribution under identical training conditions.

5. Results

5.1. Overall Performance Across Label Ratios

We evaluate all methods under varying label budgets on a fixed training pool. For a given `rol`, every method receives the identically labeled subset, while the remainder of the training set is treated as unlabeled data (consumable only by SSL methods). To provide a strong supervised reference point, we also report *SUP*($\texttt{rol} = 1$), trained on all available labels. Unless otherwise noted, all methods share the same backbone, optimizer, and training schedule for fairness. We report test accuracy (%) at every epoch over the entire training horizon. Results are visualized with per-`rol` breakdowns in Figures 3 and 4.

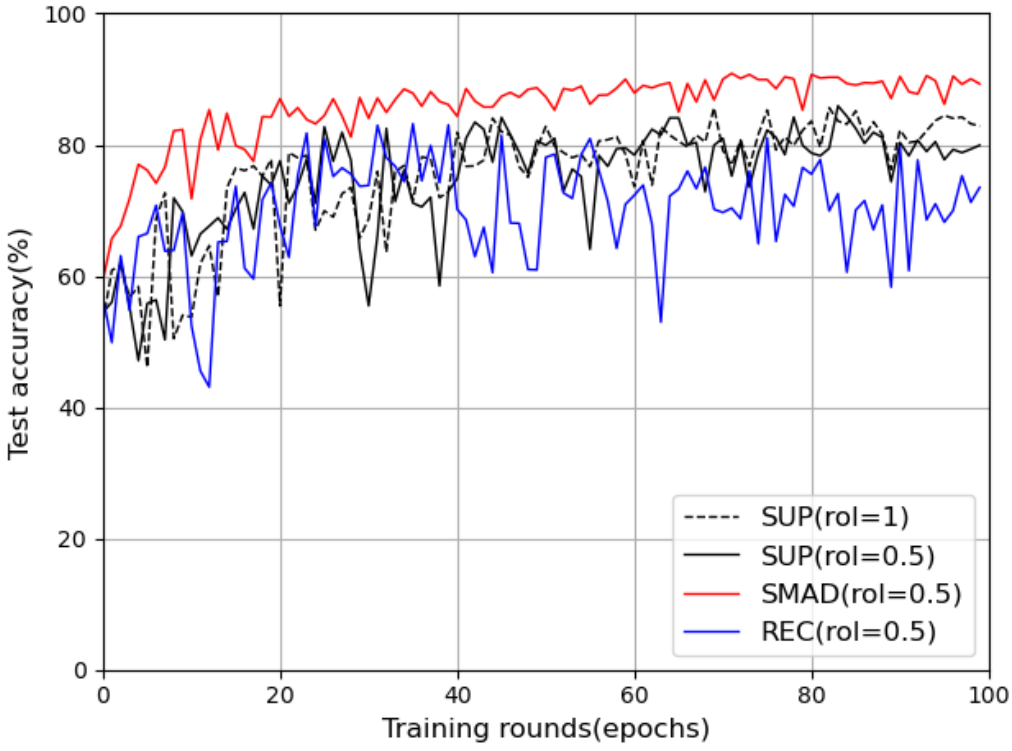
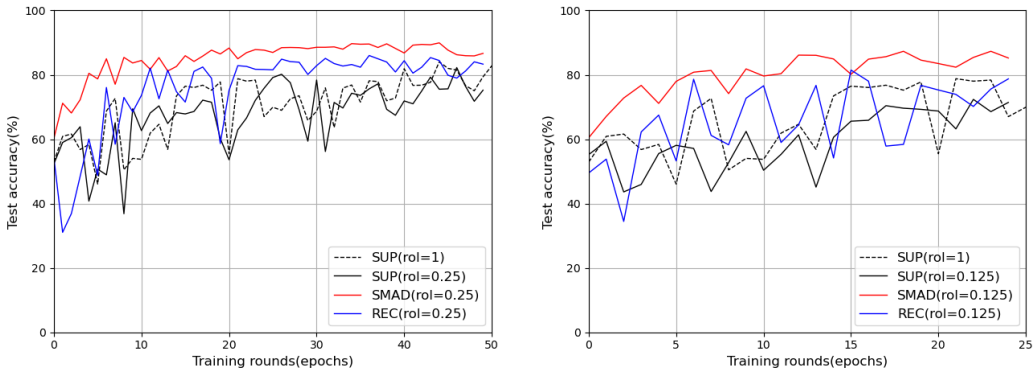


Figure 3. Test accuracy vs. epochs at `rol` = 0.5.



(**a**) `rol` = 0.25 (**b**) `rol` = 0.125

Figure 4. Side-by-side comparison under reduced label ratios.

Results at `rol` = 0.5 (Figure 3): With half of the labels available, SMAD rises rapidly to strong test performance and stabilizes above both supervised references and the pseudo-labeling variant; by the end of training, it matches or exceeds the full-label supervised reference while maintaining a smoother trajectory than *REC*. In contrast, *REC* attains intermediate accuracy but exhibits higher variance across epochs, consistent with noisier target signals. Beyond the generic benefits of consistency regularization, we attribute a substantive fraction of this margin to SMAD's segmentation-oriented backbone, which performs pixel-level feature extraction and aggregation with ASPP. The resulting dense, multi-scale representations fuse fine-grained local cues with global layout, yielding descriptors that are more invariant to APK packing/repackaging and obfuscation artifacts when binaries are rendered as images. Aggregating agreement over many spatial locations provides a higher signal-to-noise ratio for the unsupervised target, improves confidence calibration early in training, and dampens view-specific artifacts—mechanisms that plausibly account for the observed stability and the higher asymptote at `rol` = 0.5. Overall, Figure 3 indicates that, in this setting, unlabeled data—channeled through dual-branch consistency on dense features—yields measurable gains over purely supervised optimization on the same backbone and schedule.

Results at `rol` = 0.25 and `rol` = 0.125 (Figure 4). When the labeled budget is reduced to one quarter and one eighth, SMAD preserves a pronounced advantage over both the same-budget supervised baseline and the pseudo-labeling variant, while exhibiting accelerated convergence—it reaches a stable operating point in fewer epochs. Importantly, this stability is achieved at a slightly lower accuracy asymptote than in higher-label regimes, reflecting the diminished ceiling imposed by scarce labeled supervision. The method ordering is invariant across the training horizon: SMAD attains the highest terminal accuracy at both `rol` = 0.25 and `rol` = 0.125; purely supervised training plateaus earlier at a lower level, and the pseudo-labeling variant persistently trails.

The joint pattern of faster stabilization and modestly attenuated asymptotes is consistent with established semi-supervised learning principles. Cross-view consistency encourages decision boundaries to align with low-density regions of the data manifold, yielding strong regularization that accelerates optimization dynamics, whereas the limited volume of labeled evidence constrains the attainable peak performance [27]. Complementarily, entropy minimization on unlabeled instances sharpens posterior assignments without overfitting to the small labeled set—provided augmentations and targets are well calibrated—thereby sustaining margins over supervised training even when labels are sparse [28]. Empirically, contemporary consistency-based methods (e.g., MixMatch [13], ReMixMatch [14], and FixMatch [16]) report analogous behavior in low-label regimes—earlier convergence with a slightly reduced ceiling relative to richer supervi-

sion—while maintaining decisive gains over purely supervised and pseudo-labeling baselines. The curves in Figure 4 mirror these reports, evidencing that SMAD continues to leverage the unlabeled corpus effectively, converging rapidly yet retaining superior terminal accuracy under severe label scarcity.

Why consistency-based SSL outperforms supervised and pseudo-labeling baselines. Dual-view consistency imposes a prior agreement that encourages low-density separation and constrains function complexity, which is known to improve generalization under label scarcity; weight-averaged or consistency-target variants (e.g., Mean Teacher [12]) routinely outperform comparable supervised training with few labels. And modern confidence-aware formulations (e.g., FixMatch [16]) demonstrate that SSL can match or even surpass full-label supervised references in standard vision benchmarks [18]. By contrast, pure pseudo-labeling is susceptible to confirmation bias—early mistakes reinforce themselves—leading to oscillatory learning curves and inferior asymptotes unless additional regularization is introduced [19]. These mechanisms align with what we observe at `rol` = 0.5 in Figure 3.

Comparison with a popular SSL baseline (FixMatch [16]). To contextualize performance against a widely adopted SSL method, we include FixMatch [16] trained under the schedule and label splits. Table 1 summarizes accuracy, macro-precision, macro-recall, and macro-F1 (mean ± std over three runs) for SMAD and FixMatch under identical SSL setups. SMAD outperforms FixMatch consistently at all label ratios. For example, at `rol` = 0.5, SMAD improves accuracy by +3.2 pts (91.2 vs. 88.0) and F1 by +4.2 pts (90.3 vs. 86.1); at `rol` = 0.25, the gaps widen to +4.5 accuracy/+5.3 F1; and at `rol` = 0.125, to +8.3 accuracy/+9.2 F1. The precision/recall margins follow the same trend (e.g., recall +4.8, +6.1, and +10.5 pts for `rol` = 0.5, 0.25, and 0.125, respectively), indicating that SMAD maintains higher sensitivity without sacrificing precision as supervision decreases. These gains exceed the reported standard deviations, suggesting robust improvements rather than noise.

Table 1. Comparison with FixMatch across label ratios (values are mean ± std over three runs).

Method (`rol`)	**Accuracy**	**Precision**	**Recall**	**F1**
SMAD (0.5)	91.2% ± 0.5%	89.9% ± 0.6%	90.7% ± 0.6%	90.3% ± 0.5%
SMAD (0.25)	90.3% ± 0.4%	88.7% ± 0.7%	89.7% ± 0.6%	89.1% ± 0.5%
SMAD (0.125)	89.4% ± 0.6%	87.2% ± 0.8%	88.7% ± 0.5%	87.8% ± 0.7%
FixMatch (0.5)	88.0% ± 1.4%	86.5% ± 0.9%	85.9% ± 1.8%	86.1% ± 1.4%
FixMatch (0.25)	85.8% ± 1.1%	84.4% ± 0.9%	83.6% ± 0.9%	83.8% ± 0.9%
FixMatch (0.125)	81.1% ± 0.7%	79.6% ± 1.1%	78.2% ± 1.2%	78.6% ± 0.9%

Per-class accuracy at `rol` = 0.5. Per-class accuracy at `rol` = 0.5 shows that SMAD achieves 91.5/82.3/92.5/88.1/96.8(%), exceeding FixMatch (77.1/76.7/90.8/87.4/95.4) and SMAD^{W} (87.7/78.0/93.6/86.1/95.5) across all classes.

5.2. Ablation Study

Dense pixel-wise multi-scale vs. WideResNet. Table 2 isolates the effect of the encoder by swapping SMAD's segmentation-oriented backbone for a standard WideResNet while keeping the SSL objective and schedules fixed. The dense, pixel-wise multi-scale backbone yields higher accuracy/F1 at every budget, and the gap increases as labels become scarcer (e.g., F1 +1.3 @`rol` = 0.5, +3.5 @`rol` = 0.25, +6.8 @`rol` = 0.125; accuracy +1.4, +2.5, +5.4). This pattern supports our design claim: spatially dense multi-scale features provide stronger unsupervised targets under consistency, improving label efficiency when annotations are limited.

Effect of confidence-thresholded consistency (Figure 5). Introducing a confidence gate into the consistency term (*SMAD-THR*) leads to a systematic reduction in accuracy across

all label budgets relative to SMAD, with the gap widening as the labeled ratio decreases. At `rol` = 0.5, *SMAD-THR* converges to a visibly lower plateau than SMAD and exhibits a noisier early trajectory; at `rol` = 0.25 and `rol` = 0.125, the attenuation is more pronounced and the early-phase volatility persists longer, indicating that the gate suppresses a substantial fraction of unlabeled updates when predictors are initially underconfident. This behavior is consistent with the precision–recall trade-off inherent to confidence filtering: while high thresholds are designed to exclude erroneous targets, they simultaneously curtail the recall of informative unlabeled instances and can starve the unsupervised signal during the period when it is most needed. Moreover, requiring both branches to exceed the threshold tightens the criterion further and magnifies the effect under class imbalance or calibration mismatch. Prior studies reported analogous sensitivities—confidence-thresholding yields gains when calibration is strong and augmentation is aggressive (e.g., FixMatch [16], Noisy Student [17]), but can underperform when thresholds are overly conservative or confidence is miscalibrated [16–18]. The curves in Figure 5 align with the latter regime: A harder gate reduces detrimental noise yet leaves performance below that of agreement-driven SMAD that exploits the unlabeled pool more continuously.

Table 2. SMAD vs. SMAD^{W}. Values are mean ± std over three runs.

Method (with `rol`)	Accuracy	Precision	Recall	F1
SMAD (0.5)	91.0% ± 0.8%	89.2% ± 1.1%	90.2% ± 1.1%	89.6% ± 1.1%
SMAD (0.25)	90.3% ± 0.4%	88.7% ± 0.7%	89.7% ± 0.6%	89.1% ± 0.5%
SMAD (0.125)	89.4% ± 0.6%	87.2% ± 0.8%	88.7% ± 0.5%	87.8% ± 0.7%
SMAD^{W} (0.5)	89.6% ± 0.3%	88.0% ± 0.5%	88.6% ± 0.4%	88.3% ± 0.5%
SMAD^{W} (0.25)	87.8% ± 0.5%	86.4% ± 0.9%	85.1% ± 0.8%	85.6% ± 0.8%
SMAD^{W} (0.125)	84.0% ± 1.2%	82.9% ± 0.5%	80.5% ± 2.3%	81.0% ± 2.0%

SMAD^{W} uses WideResNet instead of SMAD's segmentation-oriented backbone.

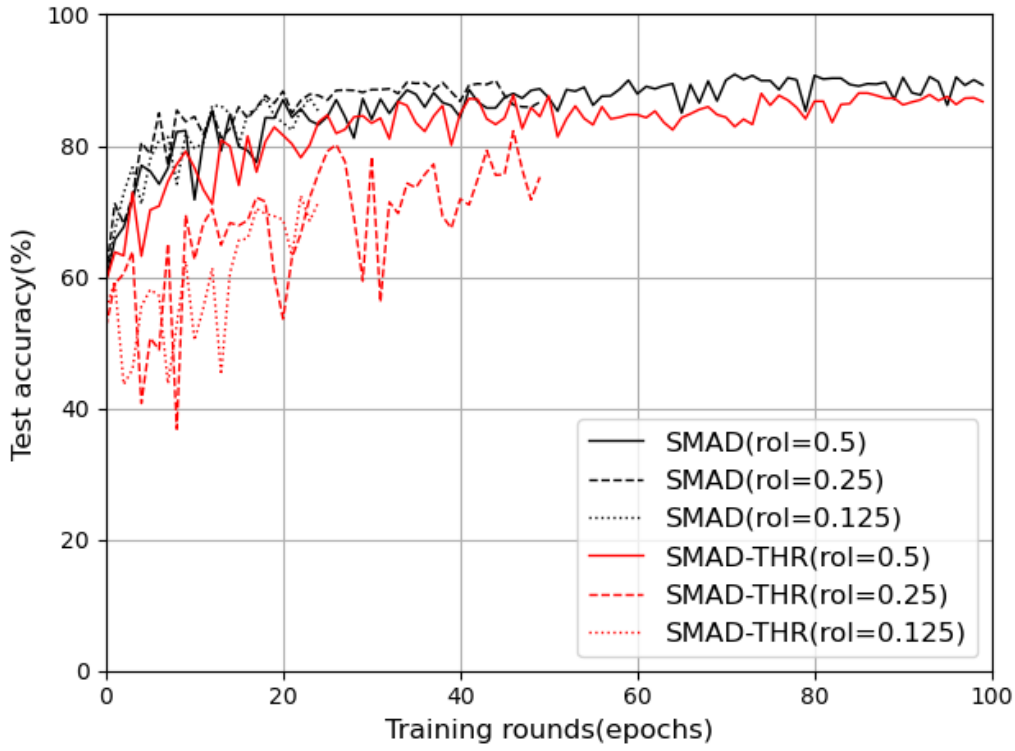


Figure 5. Effect of confidence-thresholded consistency (*SMAD-THR*) across label ratios.

Computational efficiency. On the same hardware (Section 4.1), the average training time per epoch over 100 epochs is 6.26 min (FixMatch), 6.50 min (SMAD^{W}), and 8.49 min (SMAD). Thus, SMAD incurs a modest overhead of +35.6% vs. FixMatch and +30.6% vs. SMAD^{W}, reflecting the heavier encoder. For inference on the fixed test set of 1680 images, total latency is 45 s (FixMatch), 105 s (SMAD^{W}), and 140 s (SMAD), corresponding to throughputs of $\sim$37.3 img/s ($\approx$26.8 ms/img), $\sim$16.0 img/s ($\approx$62.5 ms/img), and $\sim$12.0 img/s ($\approx$83.3 ms/img), respectively. In practice, these latencies remain compatible with offline scanning and scheduled batch analytics, and the observed gains in detection accuracy, macro-precision, macro-recall, and macro-F1 (Section 5.1) often outweigh the moderate increase in compute for SMAD in security operations where false decisions are costlier than additional milliseconds at inference.

5.3. Summary and Concluding Remarks

Under realistic labeling constraints characteristic of security telemetry, models that leverage abundant unlabeled data deliver the greatest return on supervision. Across all label budgets, SMAD's dual-branch consistency achieves superior terminal accuracy and smoother optimization than purely supervised training on the same labeled subset and a pseudo-labeling variant, remaining competitive with—often surpassing—the full-label supervised reference. In the more constrained regimes (`rol` $= 0.25, 0.125$), SMAD reaches a stable operating point in fewer epochs with only a modest reduction in the final ceiling, evidencing strong label efficiency without sacrificing late-stage stability. A contributing factor is the segmentation-oriented backbone, whose pixel-level, multi-scale features yield descriptors more invariant to APK packing/repackaging and obfuscation; aggregating agreement over dense spatial cues provides a higher-SNR unsupervised signal and better early calibration, thereby strengthening the benefits of consistency training in this domain.

The confidence-gated ablation clarifies an operational trade-off. While hard thresholds effectively suppress low-confidence noise, they also curtail the recall of informative unlabeled instances during cold start, yielding lower asymptotes in our setting. Consequently, calibration-aware or curriculum mechanisms—such as temperature scaling, dynamic or branch-aware thresholds, or soft weighting—are preferable to fixed gates when calibration is uncertain or class priors drift.

Taken together, these results indicate that dense pixel-wise multi-scale encoders paired with dual-branch agreement realize the most value when labels are scarce. In malware-image scenarios where padding/obfuscation and layout idiosyncrasies are common, preserving fine spatial cues before classification appears particularly beneficial.

6. Conclusions

This paper introduced SMAD, a semi-supervised detector for Android malware that couples dual-branch agreement with a segmentation-oriented encoder to exploit unlabeled APK imagery under label scarcity. Across label budgets, SMAD surpassed supervised training on the same backbone and a recursive pseudo-labeling variant while exhibiting smoother optimization dynamics. In addition, a comparison against FixMatch [16] under the same training schedule showed higher accuracy/precision/recall/F1 across label ratios, and a backbone ablation confirmed that dense pixel-wise multi-scale features yield consistent gains over a standard WideResNet. The confidence-thresholded ablation clarified an operational trade-off: hard gates filter low-confidence noise but reduce unlabeled coverage during cold start, lowering the asymptote in our setting. Taken together, these results support SMAD as a label-efficient, deployment-oriented approach for malware analytics where annotations are limited.

Limitations and Future Work. We use only weak augmentations and a fixed confidence threshold (no tuning). Future work will focus on augmentation-policy design and confidence-threshold optimization (adaptive or schedule-based).

Author Contributions: Conceptualization, S.L.; Methodology, S.L.; Validation, S.H.; Writing—original draft, S.L.; Writing—review & editing, S.L.; Visualization, S.L. and S.H.; Supervision, S.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Regional Innovation System & Education (RISE) program through the (Chungbuk Regional Innovation System & Education Center), funded by the Ministry of Education (MOE) and the (Chungcheongbuk-do), Republic of Korea (2025-RISE-11-004).

Data Availability Statement: Publicly available datasets were analyzed in this study. The CICMalDroid 2020 dataset, released by the Canadian Institute for Cybersecurity (University of New Brunswick), is accessible at: https://www.unb.ca/cic/datasets/maldroid-2020.html (accessed on 26 October 2025).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Zaim bin Ahmad, M.S.; Takemoto, K. Large-scale moral machine experiment on large language models. *PLoS ONE* **2025**, *20*, e0322776. [CrossRef]
2. Hasanzadeh, F.; Josephson, C.B.; Waters, G.; Adedinsewo, D.; Azizi, Z.; White, J.A.T. Bias recognition and mitigation strategies in artificial intelligence healthcare applications. *npj Digit. Med.* **2025**, *8*, 154. [CrossRef] [PubMed]
3. Ahmed, M.; Soofi, A.A.; Raza, S.; Khan, F.; Ahmad, S.; Khan, W.U.; Asif, M.; Xu, F.; Han, Z. Advancements in RIS-Assisted UAV for Empowering Multiaccess Edge Computing: A Survey. *IEEE Internet Things J.* **2025**, *12*, 6325–6346. [CrossRef]
4. Wolniak, R.; Stecuła, K. Artificial Intelligence in Smart Cities—Applications, Barriers, and Future Directions: A Review. *Smart Cities* **2024**, *7*, 1346–1389. [CrossRef]
5. Hashmi, E.; Yamin, M.M.; Yayilgan, S.Y. Securing tomorrow: A comprehensive survey on the synergy of Artificial Intelligence and information security. *AI Ethics* **2025**, *5*, 1911–1929. [CrossRef]
6. Yang, X.; Song, Z.; King, I.; Xu, Z. A Survey on Deep Semi-Supervised Learning. *IEEE Trans. Knowl. Data Eng.* **2023**, *35*, 8934–8954. [CrossRef]
7. Nguyen, A.T.; Raff, E.; Nicholas, C.; Holt, J. Leveraging uncertainty for improved static malware detection under extreme false positive constraints. *arXiv* **2021**, arXiv:2108.04081. [CrossRef]
8. Ucci, D.; Aniello, L.; Baldoni, R. Survey of machine learning techniques for malware analysis. *Comput. Secur.* **2019**, *81*, 123–147. [CrossRef]
9. Alhogail, A.; Alharbi, R.A. Effective ML-Based Android Malware Detection and Categorization. *Electronics* **2025**, *14*, 1486. [CrossRef]
10. Huang, W.; Stokes, J.W. MtNet: A multi-task neural network for dynamic malware classification. In Proceedings of the International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, San Sebastián, Spain, 7–8 July 2016; pp. 399–418.
11. Ahmad, R.; Alsmadi, I.; Alhamdani, W.; Tawalbeh, L.A. Zero-day attack detection: A systematic literature review. *Artif. Intell. Rev.* **2023**, *56*, 10733–10811. [CrossRef]
12. Tarvainen, A.; Valpola, H. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *arXiv* **2017**, arXiv:1703.01780. [CrossRef]
13. Berthelot, D.; Carlini, N.; Goodfellow, I.; Papernot, N.; Oliver, A.; Raffel, C.A. Mixmatch: A holistic approach to semi-supervised learning. *arXiv* **2019**, arXiv:1905.02249. [CrossRef]
14. Berthelot, D.; Carlini, N.; Cubuk, E.D.; Kurakin, A.; Sohn, K.; Zhang, H.; Raffel, C. Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring. *arXiv* **2019**, arXiv:1911.09785. [CrossRef]
15. Xie, Q.; Dai, Z.; Hovy, E.; Luong, M.-T.; Le, Q.V. Unsupervised data augmentation for consistency training. *arXiv* **2019**, arXiv:1904.12848. [CrossRef]
16. Sohn, K.; Berthelot, D.; Carlini, N.; Zhang, Z.; Zhang, H.; Raffel, C.A.; Cubuk, E.D.; Kurakin, A.; Li, C.L. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *arXiv* **2020**, arXiv:2001.07685. [CrossRef]
17. Xie, Q.; Luong, M.T.; Hovy, E.; Le, Q.V. Self-training with noisy student improves imagenet classification. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 14–19 June 2020; pp. 10687–10698. [CrossRef]
18. Oliver, A.; Odena, A.; Raffel, C.A.; Cubuk, E.D.; Goodfellow, I. Realistic evaluation of deep semi-supervised learning algorithms. *arXiv* **2018**, arXiv:1804.09170. [CrossRef]

19. Arazo, E.; Ortego, D.; Albert, P.; O'Connor, N.E.; McGuinness, K. Pseudo-labeling and confirmation bias in deep semi-supervised learning. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020; pp. 1–8. [CrossRef]
20. Mvula, P.K.; Branco, P.; Jourdan, G.V.; Viktor, H.L. A Survey on the Applications of Semi-supervised Learning to Cyber-security. *ACM Comput. Surv.* **2024**, *56*, 1–41. [CrossRef]
21. Memon, M.; Unar, A.A.; Ahmed, S.S.; Daudpoto, G.H.; Jaffari, R. Feature-Based Semi-Supervised Learning Approach to Android Malware Detection. *Eng. Proc.* **2023**, *32*, 6. [CrossRef]
22. Liu, M.; Yang, Q.; Wang, W.; Liu, S. Semi-Supervised Encrypted Malicious Traffic Detection Based on Multimodal Traffic Characteristics. *Sensors* **2024**, *24*, 6507. [CrossRef]
23. Chin, M.; Corizzo, R. Continual Semi-Supervised Malware Detection. *Mach. Learn. Knowl. Extr.* **2024**, *6*, 2829–2854. [CrossRef]
24. Li, J.; Zhang, Y.; Huang, Y.; Leach, K. Malmixer: Few-shot malware classification with retrieval-augmented semi-supervised learning. *arXiv* **2024**, arXiv:2409.13213.
25. Shu, R.; Xia, T.; Tu, H.; Williams, L.; Menzies, T. Reducing the Cost of Training Security Classifier (via Optimized Semi-Supervised Learning). *arXiv* **2022**, arXiv:2205.00665. [CrossRef]
26. Zheng, X.; Yang, S.; Wang, X. SF-IDS: An Imbalanced Semi-Supervised Learning Framework for Fine-Grained Intrusion Detection. In Proceedings of the IEEE International Conference on Communications, Rome, Italy, 28 May–1 June 2023. [CrossRef]
27. Belkin, M.; Niyogi, P.; Sindhwani, V. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *J. Mach. Learn. Res.* **2006**, *7*, 2399–2434.
28. Grandvalet, Y.; Bengio, Y. Semi-supervised learning by entropy minimization. In Proceedings of the Advances in Neural Information Processing Systems (NIPS), Vancouver, BC, Canada, 1 December 2004.
29. Dang, Z.; Zheng, Y.; Lin, X.; Peng, C.; Chen, Q.; Gao, X. Semi-Supervised Learning for Anomaly Traffic Detection via Bidirectional Normalizing Flows. *arXiv* **2024**, arXiv:2403.10550. [CrossRef]
30. Williams, B.; Qian, L. Semi-Supervised Learning for Intrusion Detection in Large Computer Networks. *Appl. Sci.* **2025**, *15*, 5930. [CrossRef]
31. Yuan, Y.; Huang, Y.; Zeng, X.; Mei, H.; Cheng, G. M3S-UPD: Efficient Multi-Stage Self-Supervised Learning for Fine-Grained Encrypted Traffic Classification with Unknown Pattern Discovery. *arXiv* **2025**, arXiv:2505.21462.
32. Sun, J.; Zhang, X.; Wang, Y.; Jin, S. CoMDet: A Contrastive Multimodal Pre-Training Approach to Encrypted Malicious Traffic Detection. In Proceedings of the 2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC), Osaka, Japan, 2–4 July 2024; pp. 1118–1125. [CrossRef]
33. Perales Gómez, Á.L.; Fernández Maimó, L.; Huertas Celdrán, A.; García Clemente, F.J. An interpretable semi-supervised system for detecting cyberattacks using anomaly detection in industrial scenarios. *IET Inf. Secur.* **2023**, *17*, 553–566. [CrossRef]
34. Krajewska, A.; Niewiadomska-Szynkiewicz, E. Clustering Network Traffic Using Semi-Supervised Learning. *Electronics* **2024**, *13*, 2769. [CrossRef]
35. Nataraj, L.; Karthikeyan, S.; Jacob, G.; Manjunath, B.S. Malware images: Visualization and automatic classification. In Proceedings of the 8th International Symposium on Visualization for Cyber Security, Pittsburgh, PA, USA, 20 July 2011; pp. 1–7. [CrossRef]
36. Seneviratne, S.; Shariffdeen, R.; Rasnayaka, S.; Kasthuriarachchi, N. Self-Supervised Vision Transformers for Malware Detection. *IEEE Access* **2022**, *10*, 103121–103135. [CrossRef]
37. Chen, L. C.; Zhu, Y.; Papandreou, G.; Schroff, F.; Adam, H. Encoder-decoder with atrous separable convolution for semantic image segmentation. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 801–818.
38. Wang, Z.; Zhao, Z.; Xing, X.; Xu, D.; Kong, X.; Zhou, L. Conflict-based cross-view consistency for semi-supervised semantic segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 18–22 June 2023; pp. 19585–19595. [CrossRef]
39. Android Malware Dataset (CICMalDroid 2020). Available online: https://www.unb.ca/cic/datasets/maldroid-2020.html (accessed on 5 August 2025).
40. Lee, S. Distributed Detection of Malicious Android Apps While Preserving Privacy Using Federated Learning. *Sensors* **2023**, *23*, 2198. [CrossRef]
41. PyTorch. Available online: https://pytorch.org/ (accessed on 5 August 2025).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.