TrustBoot: A Trust Bootstrapping Framework for Semi-Supervised Malware Detection

A. Augello, A. De Paola, G. Lo Re

Proceedings of the 18th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART
DOI: https://doi.org/10.5220/0014465700004052

TrustBoot: a Trust Bootstrapping Framework for Semi-supervised Malware Detection

Andrea Augello[1], Alessandra De Paola[1] and Giuseppe Lo Re[1]
[1]*University of Palermo, Palermo, Italy*
andrea.augello01@unipa.it, alessandra.depaola@unipa.it, giuseppe.lore@unipa.it

Keywords: Malware Detection, Semi-Supervised Learning, Anomaly Detection.

Abstract: Today, malware detection represents one of the most critical cybersecurity challenges due to the rapid evolution of threats. One of the most promising approach is the adoption of machine learning (ML) detection methods, nevertheless, their design is not trivial due to the scarcity of up-to-date labeled data. In order to keep up with emerging malware variants, ML-based detection systems must be frequently updated and retrained using recent samples. However, the manual process of feature engineering and expert labeling and analysis is time-consuming and costly, making it impractical for frequent updates. This work presents TrustBoot, a semi-supervised framework for detecting malicious software, that exploits the exact knowledge only about a small set of trusted applications, and is capable of processing a larger set of unlabeling applications. To achieve this goal, TrustBoot adopts a visual encoding of binary executable, that eases the detection of anomalies, which are related to the presence of malware. Experiments on large Android malware datasets demonstrate that the proposed pipeline achieves competitive detection performance, matching or exceeding fully supervised approaches while substantially reducing the need for manual intervention for the dataset curation and overcoming the reliance on labeled malicious data.

1 Introduction

The rapid and pervasive adoption of connected devices, cloud services and mobile platforms are the backbone of modern digital life. However, this digital transformation has also dramatically increased the attack surface available to malicious actors, leading to a surge in malware threats targeting the confidentiality, integrity, and availability of systems and data (Ansari et al., 2024). These emerging threats cause a continue arms race between attackers and defenders, with malware authors constantly evolving in sophistication, altering their techniques to evade detection and compromise systems in novel ways (Tang et al., 2023).

Machine learning has emerged as a powerful tool for malware detection, showing promise in identifying patters that might indicate malicious behavior, and becoming a key component in modern cybersecurity research (Arp et al., 2022). Nevertheless, in real-world scenarios, machine learning-based malware detection approaches often fall short of their potential. This discrepancy arises both from flaws in the experimental methodology (Arp et al., 2022; Pendlebury et al., 2019) and from practical constraints in operational settings. One of the main challenges is the rapid obsolescence of learned models (Agate et al., 2025). Malware authors continuously adapt their techniques to evade detection, leading to concept drift that degrades the performance of static models over time (Augello et al., 2025c). In order to contrast this phenomenon, models must be frequently retrained on fresh data reflecting the current threat landscape.

The scarcity of high-quality labeled data significantly hampers the effectiveness of the retraining process. While large volumes of telemetry can be collected from endpoints and networks, and app stores can be easily scraped to collect new data, obtaining accurate labels for these samples is challenging and cost-intensive (Augello et al., 2025b). For this reason, recent research has focused on reducing the reliance on manual labels by leveraging unsupervised learning techniques that can exploit large unlabeled datasets to learn robust representations and detect anomalies (Agate et al., 2022; De Paola et al., 2024; Augello et al., 2024; Agate et al., 2024).

The core assumption of these approaches is that malicious samples exhibit anomalous patterns compared to the majority of benign software, allowing models trained on benign data to identify outliers that may correspond to malware. Typically,

this is achieved by training autoencoder architectures to reconstruct benign inputs, with the intuition that anomalous inputs will yield higher reconstruction errors (Augello et al., 2025a). Deep learning autoencoder architectures, however, are prone to over-generalization, where the model learns to reconstruct or classify even anomalous inputs well (Gong et al., 2019). This phenomenon reduces the effectiveness of anomaly-based detection, as the model fails to distinguish between benign and malicious samples.

This work introduces TrustBoot, a practical semi-supervised framework that directly addresses the over-generalization problem of autoencoder-based anomaly detectors. TrustBoot is designed to minimize the need for manual curation and labeling while maintaining competitive detection performance. The method is designed for scenarios where only a relatively small set of verified benign samples is available and the remainder of collected binaries are unlabeled and potentially contaminated by unknown malware. TrustBoot relies on a visual encoding of binaries into RGB images, which captures spatially coherent patterns in the byte sequences that are informative for malware detection (Freitas et al., 2022) and have been shown to yield strong performance in supervised settings (Abdel-Basset et al., 2022). This image-based representation removes the burden of manual feature engineering, which is often a time-consuming and expertise-intensive step with large impact on model quality (Dambra et al., 2023), and allows compact deep models to learn relevant features directly from raw data in a context-agnostic manner. Complementing this input representation, TrustBoot adopts a semi-supervised learning regime that combines reconstruction and contrastive objectives to enable semi-supervised inference on the larger set of unlabeled binaries (Lindenbaum et al., 2024), improving the inductive detection capabilities of the model. By using reconstruction errors to surface candidate anomalies and contrastive shaping to structure the latent space, TrustBoot iteratively refines its representations on unlabeled data and improves discrimination between benign and malicious samples.

This work makes the following main contributions:

- A practical semi-supervised framework (TrustBoot) that bootstraps trust from a small curated benign core and a large set of unlabeled applications to detect malware.
- TrustBoot eliminates manual feature engineering thanks to its image-based representation that allows compact convolutional models to learn relevant features directly from raw data.
- An iterative pseudo-labeling scheme based on latent-space similarity to conservatively expand labeled data and fine-tune the model with minimal benign labels, combating over generalization and improving detection performance.
- Comprehensive empirical validation on large APK image datasets (MalNet-Image (Freitas et al., 2022)) with sensitivity and ablation studies showing substantial performance gains compared to baseline autoencoder anomaly detection, and competitive results compared to fully supervised state-of-the-art approaches.

The remainder of the paper is organized as follows. Section 2 discusses related works, section 3 illustrates the TrustBoot system, section 4 presents the experimental evaluation, and section 5 concludes the paper.

2 Related works

The field of cybersecurity is perpetually challenged by the rapid emergence of novel and zero-day threats, which quickly evade established defense mechanisms (Camarda et al., 2025). While traditional signature-based methods are efficient for known threats, they are brittle against sophisticated polymorphic and metamorphic variants. Dynamic analysis, which monitors runtime behaviors, offers greater resilience but suffers from high computational overhead and crucial vulnerability to strategic evasion by samples designed to bypass virtualized environments or sandboxes. These inherent drawbacks have driven the community toward machine learning and deep learning methods capable of learning abstract, generalized patterns from static inputs (Dambra et al., 2023).

A noteworthy development in achieving scalable static analysis with minimal feature engineering is the representation of binary executables as images. Early seminal work demonstrated that mapping raw byte streams to grayscale images yields representations suitable for convolutional networks (Nataraj et al., 2011). This technique is strategically valuable because it removes the burden of manual feature engineering, an expertise-intensive step that often dictates final model quality. The image representation allows compact deep models, typically Convolutional Neural Networks (CNNs), to learn relevant features directly from the raw data in a context-agnostic manner. Subsequent research further validated that distinct visual features correlate directly with underlying code sections, structural artifacts, and obfuscation strategies (Li et al., 2024; Zhu et al., 2021).

Despite the effectiveness of deep learning on visual features, a critical logistical impediment re-

mains: the economic and operational constraints associated with obtaining and maintaining high-quality labeled data necessary to combat model aging (Augello et al., 2025b). In other cybersecurity contexts, such as intrusion detection, the scarcity of labeled data has spurred significant interest in unsupervised and self-supervised learning techniques (Verkerken et al., 2022); in malware detection, however, the majority of works still rely on fully supervised learning paradigms, and the exploration of unsupervised approaches remains largely unexplored. This limited adoption can be traced to the fact that the assumption that malicious samples are rare, clearly anomalous outliers is more applicable to intrusion-detection telemetry than to static malware artifacts: it is much harder to collect large datasets of software with absolute certainty about their benign/malicious status. Such limitation is one of the main drivers behind this work.

Autoencoders (AEs) are a common choice for anomaly detection in high-dimensional domains where classic methods (e.g., isolation forests or One-Class SVMs) perform poorly (Hojjati et al., 2024). The usual approach is to flag inputs with unusually large reconstruction error as anomalies, but deep AEs are powerful function approximators that can "over-generalize" and reconstruct out-of-distribution inputs well, undermining reconstruction error as a reliable anomaly score (Gong et al., 2019). Prior works have addressed this with architectural or regularization strategies; e.g., memory-augmented decoders that restrict reconstructions to prototypical patterns (Gong et al., 2019), and adversarial or negative-sampling schemes that explicitly discourage reconstruction of anomalous instances (Qiu et al., 2022). While these methods mitigate the issue to some extent, they either impose strict architectural constraints or rely on access to known anomalous examples; consequently, they do not fully solve the problem in largely unlabeled settings with unknown anomaly distributions.

Few works actively leverage anomalous data to directly counteract over-generalization (Wu, 2023), and mostly do so as a side-effect of having to handle contaminated training data (Qiu et al., 2022). Despite these architectural mitigations, the core challenge of reconstruction-based scoring persists, prompting a shift in research focus toward methods that actively shape the latent space to improve separability, moving beyond passive reconstruction. Contrastive learning achieves separation by maximizing the similarity between representations of augmented "positive pairs" while pushing away "negative pairs", this improved separation in latent space can enhance anomaly detection by mapping normal data into compact clusters, making outliers easier to identify, provided that meaningful positive and negative pairs can be formed (Hojjati et al., 2024). A critical challenge in applying this to binary visualization is that standard image augmentations destroy the sequential byte information encoded in the image structure. Consequently, techniques must be adapted to form meaningful positive pairs that enforce robust intraclass compactness and stabilize the latent space for subsequent distance-based anomaly separation.

Collectively, the literature confirms a critical, unresolved gap: while Autoencoders provide the necessary unsupervised foundation, their reliance solely on minimizing the reconstruction loss leads to failure via over-generalization (Gong et al., 2019). Existing mitigation techniques are primarily passive, focusing on architectural constraints or regularization, which are insufficient to fully address the problem. The core shortfall is the objective function's insufficiency when applied in isolation; specifically, the lack of an explicit mechanism to penalize the successful reconstruction of anomalous candidates. This limitation stems by an open-world assumption where anomalies are not known a priori, making it impossible to directly optimize for their reconstruction failure during training, and the rare availability of labeled anomalies for supervised learning.

TrustBoot directly addresses this by integrating a targeted, adversarial objective derived from the contrastive concepts discussed above. This approach is possible because in the cybersecurity domain, it is feasible to collect large datasets of unlabeled applications that reflect the current application landscape, including a significant fraction of anomalies. This allows to actively shape the latent manifold to ensure discriminative feature learning, fundamentally beyond the limitations of purely reconstruction-based anomaly scoring.

3 TrustBoot framework

This section presents the TrustBoot architecture and training strategy. The main training pipeline is summarized by Figure 1. After collecting a large unlabeled set of applications, e.g., by crawling app stores, a core set of applications trusted to be benign is curated (for example, known-good system binaries, or software from vetted developers).

TrustBoot leverages a semi-supervised training procedure with three main steps to learn effective representations from the trusted benign set and the remaining unlabeled dataset, which then enables accurate inductive malware detection on unseen samples.

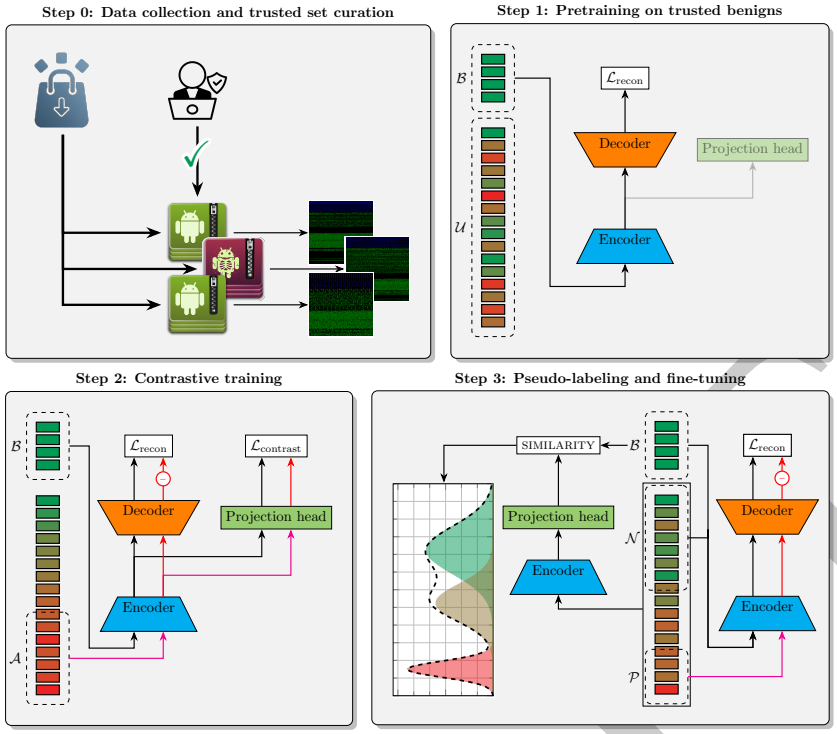

Figure 1: The TrustBoot training pipeline. After an initial scraping of unlabeled applications, a trusted benign set $\mathcal{B}$ is curated. The model is pretrained on $\mathcal{B}$ via reconstruction loss. High-error anomalies $\mathcal{A}$ are selected and used to refine the model via contrastive learning and reconstruction-loss maximization. Finally, high-confidence pseudo-labels produce pseudo-labeled positives $\mathcal{P}$ and negatives $\mathcal{N}$ based on latent-space proximity to $\mathcal{B}$, and the model is fine-tuned for improved discrimination.

The trusted core set is used to pretrain a compact convolutional autoencoder. The pretrained model is then applied to the unlabeled set to identify candidate anomalies based on reconstruction error. These anomalies are used to prevent over-generalization of the autoencoder during iterative refinement steps. Finally, high-confidence pseudo-labels are assigned to a subset of candidates based on latent-space proximity to the trusted benign set. The pseudo-labeled samples are then used to fine-tune the model for improved discrimination capabilities.

3.1 Model architecture

The model θ contains three main components: an encoder E, a decoder D, and a projection head P. The model ingests the binary executables under the form of RGB images. The executable is read as a byte stream, and each byte is mapped sequentially to a pixel value (0-255). The bytes pertaining to header information, identifiers and class definitions, and data are mapped to different channels, adding semantic information to the image. All images are then resized to 512x512 pixels. This representation was shown to be effective for classification tasks (Freitas et al., 2022).

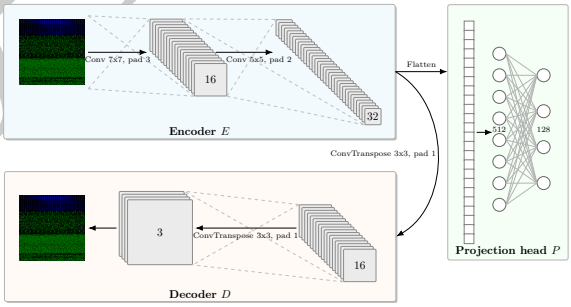

Figure 2: Neural architecture of the convolutional autoencoder with projection head. The encoder E compresses the input image to a latent tensor, which is then projected to a low-dimensional vector by the projection head P. The latent tensor is also fed to the decoder D to reconstruct the input image.

The core model is a convolutional autoencoder, shown in Figure 2, with a projection head for contrastive learning. The encoder E stacks two convolutional blocks (16 and 32 filters) with Rectified Linear Unit (ReLU) activations and max-pooling, producing a compact latent tensor. A projection head P maps the flattened latent tensor to a low-dimensional L2-normalized vector for contrastive loss computation. The decoder D reconstructs the input image via up-

sampling and convolution, producing a reconstructed RGB image. The network is intentionally compact to avoid overly-capable decoders that would trivially reconstruct anomalous inputs.

3.2 TrustBoot training pipeline

TrustBoot aims to learn an effective anomaly detection model $\theta = (E, D, P)$ from a largely unlabeled dataset $\mathcal{D} = \{x_i\}_{i=1}^{N}$, where each x_i is an image-encoded binary executable. The training goal for θ is to minimize the reconstruction Mean Squared Error (MSE) on benign samples (ground truth label $y = 0$) while maximizing it on anomalous/malicious samples (ground truth label $y = 1$), thereby enabling effective anomaly-based malware detection. This training objective is formalized in equation (1):

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{x\sim\mathcal{D},y=0}\left[\|x - D(E(x))\|_2^2\right] - \mathbb{E}_{x\sim\mathcal{D},y=1}\left[\|x - D(E(x))\|_2^2\right] \tag{1}$$

Crucially, the collected applications must be recent and representative of the current application landscape to ensure that the model learns relevant patterns for the current threat environment. The dataset $\mathcal{D}$ is obtained by scraping recent applications from app stores or repositories in a fully automated manner and is assumed to reflect the current application landscape. The benign/malicious prevalence is unknown a priori and no manual labeling is performed during collection; thus, ground truth labels $\{y_1, \ldots, y_{|\mathcal{D}|}\}$ for the samples $\mathcal{D}$ are not available. The malicious/benign ratio is assumed to correspond to the actual prevalence of malware in the wild; i.e., no sampling bias is introduced during data collection.

A small, curated trusted benign subset of the data $\mathcal{B} \subset \{x_i \in \mathcal{U} : y_i = 0\} \subset \mathcal{D}$ is assumed to be available after $\mathcal{D}$ has been collected, for instance by vetting known-good system binaries or applications from trusted developers. The core set $\mathcal{B}$ is assumed to be sampled from the same distribution as the remainder of the benign samples in $\mathcal{U}$. The remaining samples constitute the unlabeled set $\mathcal{U} = \mathcal{D} \setminus \mathcal{B}$, which may contain both benign and malicious examples.

Step 1: Pretraining on trusted benigns. Initially, only the trusted benign set $\mathcal{B}$ is used to train θ. For each sample x in $\mathcal{B}$, the encoder-decoder pair (E, D) attempts to reconstruct the input image by minimizing $\mathcal{L}_{\text{recon}}$, which is the first term in Eq. (1):

$$\mathcal{L}_{\text{recon}}(x) = \|x - D(E(x))\|_2^2. \tag{2}$$

This training step corresponds to commonly-used autoencoder-based anomaly detection techniques. However, it has been shown that only minimizing the reconstruction error on the negative class can can lead to over-generalization (Gong et al., 2019) The model θ may learn to reconstruct inputs outside the training distribution well, limiting its usefulness for anomaly detection. This hypothesis has been verified empirically in the experimental evaluation (see Section 4), where this pretraining step alone yields suboptimal detection performance. Thus, TrustBoot adds additional semi-supervised steps to the training procedure to mitigate this issue. By assigning labels to the unlabeled set $\mathcal{U}$, the objective in Eq. (1) can be approximated and explicitly optimized.

Step 2: Anomaly selection and contrastive training. The autoencoder is applied to the unlabeled set $\mathcal{U}$ to compute reconstruction errors on the unlabeled samples, which so far have not been used for training. High-error samples are selected as anomaly candidates $\mathcal{A} \subset \mathcal{U}$. A percentile-based thresholding strategy is used to select the top-p percentile of samples by reconstruction error. For the anomalous samples, Eq. (2) is maximized rather than minimized, discouraging the autoencoder from reconstructing these inputs well.

Additionally, the projection head P is trained to learn a compact latent space via contrastive learning by minimizing a normalized cross-entropy (NT-Xent (Chen et al., 2020)) loss. Given a minibatch that contains N paired examples we treat each pair as two separate "views", resulting in 2N samples in the batch. Let $z_i = P(E(x_i))$ denote the low-dimensional projection produced by the projection head for view i. A "positive pair" (i,j) is a pair of views that have to be closer together in the projected space (in this work positives are formed from trusted benign samples), while negative views in the batch should be pushed away from z_i in the projected space. The loss for a positive pair (i,j) is then:

$$\mathcal{L}_{\text{contrast}}(x_i, x_j) = -\log \frac{\exp\big(\text{sim}(P(E(x_i)), P(E(x_j)))/\tau\big)}{\sum_{k=1}^{2N} \mathbf{1}_{[k\neq i]} \exp\big(\text{sim}(P(E(x_i)), P(E(x_k)))/\tau\big)} \tag{3}$$

where $\text{sim}(a, b) = a^\top b/(\|a\|\,\|b\|)$ is the cosine similarity and τ a temperature parameter. The total contrastive loss is the average of $\mathcal{L}_{\text{contrast}}$ over all positive pairs in the batch.

Notably, classical image data augmentations (rotations, cropping, flipping) cannot be used to produce positive pairs because the spatial layout of the encoded binary is semantically meaningful: transformations that are benign for natural images can destroy the sequenced byte information encoded in these images and introduce artifacts that hurt training. Thus, positive pairs (x_i, x_j) are formed by sampling two different trusted benign samples from $\mathcal{B}$, while nega-

tive pairs are formed by sampling one sample from $\mathcal{B}$ and one from the anomalies $\mathcal{A}$. The aim of the contrastive term is to cluster benign samples together in latent space while pushing apart anomalies, improving the discriminative power of the learned representation. Forming a positive pair by sampling two distinct trusted benign examples from $\mathcal{B}$ forces the projection head to pull benign representations closer together in the projected latent space. The contrastive term therefore enforces intraclass compactness for the benign class while negatives (pairs that include anomaly candidates) are pushed away. Creating a tight benign manifold is necessary for later stages: it reduces benign intra-class variance, makes outliers easier to separate based on distance/similarity, and stabilizes the pseudo-labeling step by producing more reliable confidence scores.

The full training objective in this step combines the reconstruction and contrastive losses:

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{x\sim\mathcal{B}}[\mathcal{L}_{\text{recon}}(x)] - \mathbb{E}_{x\sim\mathcal{A}}[\mathcal{L}_{\text{recon}}(x)] + \mathbb{E}_{(x_i,x_j)\sim\mathcal{B}\cup\mathcal{A}}[\mathcal{L}_{\text{contrast}}(x_i,x_j)] \tag{4}$$

The set of anomalies $\mathcal{A}$ is updated at each iteration by re-evaluating reconstruction errors on the unlabeled set $\mathcal{U}$.

Resampling $\mathcal{A}$ at each iteration allows the model to progressively refine its notion of anomalies as it learns better representations. Given that the initial autoencoder may have limited discriminative power, not all high-reconstruction-error samples are guaranteed to be true anomalies. Instead, by iteratively updating $\mathcal{A}$, truly anomalous samples are more likely to persist in the candidate set, while false positives may be filtered/swapped out as the model improves.

Step 3: Pseudo-labeling and fine-tuning. In the final phase, the unlabeled set goes through a hard negative and positive mining step to produce pseudo-labeled negatives $\mathcal{N}$ and positives $\mathcal{P}$. In order to do so, the entire unlabeled set is encoded in the latent space via $P(E(\cdot))$. For each sample in $\mathcal{U}$, a confidence score is computed based on the average cosine similarity between its latent projection and reference benign projections drawn from $\mathcal{B}$:

$$\text{confidence}(x_j) = \sum_{x_i\in\mathcal{B}} \text{sim}(P(E(x_j)), P(E(x_i))) = \sum_{x_i\in\mathcal{B}} \frac{P(E(x_j))^\top P(E(x_i))}{\|P(E(x_j))\|\,\|P(E(x_i))\|} \tag{5}$$

Samples are split into pseudo-labeled positives/malware $\mathcal{P} \subset \mathcal{U}$ and pseudo-labeled negatives/goodware $\mathcal{N} \subset \mathcal{U}$ based on their similarity to the trusted benign set.

A three-component GMM is used to partition the confidence scores into three regions: a low-similarity component (pseudo-labeled positives $\mathcal{P}$), a high-similarity component (pseudo-labeled negatives $\mathcal{N}$), and an intermediate/uncertain component. The intermediate group is explicitly discarded to reduce label noise: samples with borderline confidences are more likely to be mislabeled, and excluding them preserves high precision of the pseudo-labels used for fine-tuning. A simple two-component model is insufficient because the empirical confidence distribution is often multimodal and exhibits nontrivial overlap between benign-like and malware-like scores; forcing a hard bipartition would push many ambiguous samples into noisy labels and harm downstream learning.

After pseudo-labeling, the newly formed sets $\mathcal{P}$ and $\mathcal{N}$ are used to augment the trusted benign set $\mathcal{B}$. The model explicitly optimizes for the objective in Eq. (1), using the augmented negative set $\mathcal{B}\cup\mathcal{N}$ as the negative class and the pseudo-labeled positive set $\mathcal{P}$ as the positive class. Optimizing Eq. (6) further refines the model's discriminative capabilities by leveraging the pseudo-labeled data to directly enforce low reconstruction error on benign-like samples and high reconstruction error on malware-like samples:

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{x\sim\mathcal{B}\cup\mathcal{N}}[\mathcal{L}_{\text{recon}}(x)] - \mathbb{E}_{x\sim\mathcal{P}}[\mathcal{L}_{\text{recon}}(x)] \tag{6}$$

The full training pipeline is summarized in Algorithm 1.

Limitations TrustBoot relies on the availability of a trusted benign set $\mathcal{B}$. A malicious contamination of $\mathcal{B}$, e.g., by an Advanced Persistence Threat actor infiltrating and/or compromising the vetting stage, could have catastrophic effects on the downstream effectiveness of TrustBoot, as verified in Section 4.2. Additionally, this work assumes that the unlabeled set $\mathcal{U}$ is an i.i.d. sample from the current application distribution, and no adversarial attempts to poison or manipulate the data collection process are considered. Finally, TrustBoot focuses on static analysis of image-encoded binaries; while this representation has been shown to be effective for malware detection, it may miss dynamic behaviors that are only observable at runtime.

4 Experimental evaluation

TrustBoot is evaluated using large-scale APK image datasets MalNet-Image (Freitas et al., 2022), containing 1262024 samples across 696 families. All experiments reported here were implemented in PyTorch (Paszke et al., 2019) and executed on a single node with an NVIDIA RTX 3050 GPU. For reproducibility, the main hyperparameters used are listed in Table 1. The number of training epochs for each

Data: Dataset $\mathcal{D}$; trusted benign set $\mathcal{B} \subset \mathcal{D}$; unlabeled set $\mathcal{U} = \mathcal{D} \setminus \mathcal{B}$; epochs for the various phases T_1, T_2, and T_3; learning rate η; percentile p for anomaly selection
Result: Trained model θ

Initialize model $\theta = (E, D, P)$;
extbfStep 1: Pretraining on trusted benigns;
for $t \leftarrow 1$ **to** T_1 **do**
  $\theta \leftarrow \theta - \eta \nabla_\theta \mathbb{E}_{x \sim \mathcal{B}}[\mathcal{L}_{\text{recon}}(x)]$;
end

extbfStep 2: Anomaly selection and contrastive training;
scores $\leftarrow \{\|x - D(E(x))\|_2^2 : x \in \mathcal{U}\}$;
threshold $\leftarrow$ percentile(scores, $100 - p$);
$\mathcal{A} \leftarrow \{x_j \in \mathcal{U} : \|x_j - D(E(x_j))\|_2^2 \geq$ threshold$\}$;
for $t \leftarrow 1$ **to** T_2 **do**
  $\theta \leftarrow \theta - \eta \nabla_\theta \big(\mathbb{E}_{x \sim \mathcal{B}}[\mathcal{L}_{\text{recon}}(x)] - \mathbb{E}_{x \sim \mathcal{A}}[\mathcal{L}_{\text{recon}}(x)] + \mathbb{E}_{(x_i, x_j) \sim \mathcal{B} \cup \mathcal{A}}[\mathcal{L}_{\text{contrast}}(x_i, x_j)]\big)$;
end

extbfStep 3: Pseudo-labeling and fine-tuning;
confidences $\leftarrow$ $\{\sum_{x_i \in \mathcal{B}} \text{sim}(P(E(x_j)), P(E(x_i))) : x_j \in \mathcal{U}\}$;
Fit GMM to confidences;
$\mathcal{P} \leftarrow \{x_j \in \mathcal{U} :$ x_j in low-similarity GMM component$\}$;
$\mathcal{N} \leftarrow \{x_j \in \mathcal{U} :$ x_j in high-similarity GMM component$\}$;
for $t \leftarrow 1$ **to** T_3 **do**
  $\theta \leftarrow \theta - \eta \nabla_\theta \big(\mathbb{E}_{x \sim \mathcal{B} \cup \mathcal{N}}[\mathcal{L}_{\text{recon}}(x)] - \mathbb{E}_{x \sim \mathcal{P}}[\mathcal{L}_{\text{recon}}(x)]\big)$;
end

Algorithm 1: TrustBoot training pipeline

phase (T_1, T_2, T_3) were selected based on convergence of the training loss.

Parameter	Value
Optimizer	Adam, lr=1e-3
Batch size	16
p (anomaly selection percentile)	30%
T_1	50
T_2	50
T_3	20

Table 1: Selected hyperparameters used for experiments.

Standard binary-detection metrics are reported: AUROC and F1-Score. Ablation studies are also performed to quantify the role of each step in the TrustBoot pipeline. Finally, a sensitivity analysis is conducted to study the impact of key parameters (negative sampling ratio, selection percentile, confidence threshold, number of benign reference vectors) on detection performance.

4.1 TrustBoot performance

The TrustBoot pipeline achieves a strong improvement over baseline error-based anomaly detection through its multiple stages. Figure 3 shows the ROC curves for the model at the end of each major step in the pipeline.

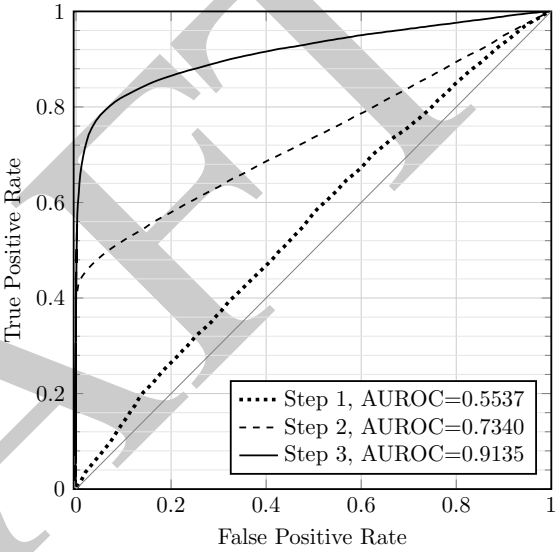

Figure 3: ROC curves at the end of each major step in the TrustBoot training pipeline. The initial autoencoder is almost equivalent to a random classifier (AUROC $\approx$ 0.55). The addition of contrastive learning and anomaly selection improves performance (AUROC $\approx$ 0.73), while the final pseudo-labeling and fine-tuning step yields strong discrimination (AUROC $\approx$ 0.91).

The traditional autoencoder-based anomaly detection on the trusted set $\mathcal{B}$ (Step 1) yields limited performance (AUROC $\approx$ 0.55, almost on par with a random classifier) due to over-generalization, as the model reconstructs many anomalous inputs well, in accordance with the initial hypothesis. The addition of the contrastive loss and iterative refinement with explicit reconstruction loss maximization for anomalous samples $\mathcal{A}$ (Step 2) reduces the over-generalization of the model, yielding a notable improvement in its discriminative capabilities (AUROC $\approx$ 0.73). Nevertheless, the model at step 2 is still unsuitable for practical deployment given the unacceptably high false positive rates required to achieve reasonable true positive rates. Finally, the pseudo-labeling and fine-tuning

step with the samples in $\mathcal{P}$ and $\mathcal{N}$ (Step 3) enables the model to achieve strong discriminative capabilities between malicious and benign software (AUROC ≈ 0.91), confirming the effectiveness of the proposed pipeline.

Table 2 compares TrustBoot against selected recent literature frameworks for binary malware detection on the same dataset. It is worth noting that the competing methods are either fully supervised or, even when self-supervised, require both negative and costlier positive labels for training, while TrustBoot only needs a small trusted benign set and can leverage large unlabeled datasets without any malicious labels. Despite the significantly reduced labeling requirements, TrustBoot matches or exceeds the performance of these baselines, achieving an F1-Score of 0.883. The only baseline that outperforms TrustBoot is DTMIC (Kumar and Janet, 2022), which uses transfer learning from large image datasets to learn high-level features, followed by supervised fine-tuning on labeled malware samples; however, this approach requires a fully-labeled malware set for fine-tuning, while TrustBoot only needs a small trusted benign set and can leverage unlabeled data.

4.2 Sensitivity analysis

Machine learning systems for malware detection often report inflated performance compared to real-world deployment scenarios due to unrealistic assumptions in dataset construction and evaluation protocols. In particular, in realistic scenarios malware is often rare compared to benign software, while many academic datasets assume balanced classes or unrealistically high malware prevalence (Pendlebury et al., 2019). To assess the robustness of TrustBoot to varying malware prevalence, a sensitivity analysis is conducted by varying the fraction of malware in the unlabeled dataset $\mathcal{U}$ from 5% to 50%, with a grater granularity in the 10%-20% range, which are reasonable estimates for the Android ecosystem (Pendlebury et al., 2019). Additionally, the impact of the ratio of positive and negative pairs used in the contrastive loss during Step 2 is studied by varying it from 0.1 to 0.9. The results are summarized in the heatmaps of Figure 4, which report AUROC and F1-Score for different malware fractions and negative sampling ratios.

The analysis shows the model benefits from a relatively high negative sampling ratio in the contrastive term (0.7-0.9). This is not surprising, as the main aim of the constrastive training regime is to clearly separate malware from benigns in latent space, and increasing the similarity between random benigns is not as helpful as pushing apart anomalies.

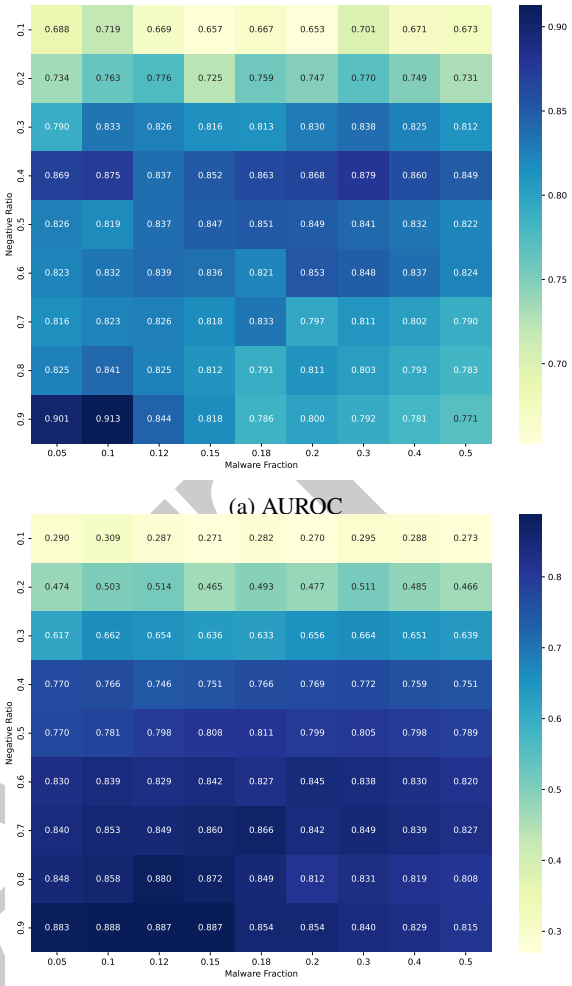

Figure 4: Heatmaps showing AUROC (left) and F1 (right) for varying malware fraction and negative ratio. Increasing the negative ratio generally improves performance, while the model achieves the strongest performance with realistic malware fractions (10-20%).

An interesting finding is that TrustBoot achieves the strongest performance when the malware fraction in the unlabeled dataset is between 10% and 20%, which is the most reslistic setting. At higher malware fractions (30-50%), performance degrades slightly, likely due to the increased risk of contamination in the pseudo-labeled negatives $\mathcal{N}$. In the remainder of the experiments, a malware fraction of 10% and a negative ratio of 0.9 are used as default settings.

Another important practical consideration is the size of the trusted benign set $\mathcal{B}$. Larger trusted sets are expected to yield better performance, as they provide more representative benign references for both pretraining and pseudo-labeling. On the other hand, curating a large trusted benign set can be costly and time-consuming, and defeats the purpose of reducing

Model	Supervision	F1-Score
ResNet18 (Seneviratne et al., 2022)	Supervised	0.862
ResNet50 (Seneviratne et al., 2022)	Supervised	0.854
MM-TIML (Sun et al., 2025)	Supervised incremental learning	0.819
FED-MAL (Abdel-Basset et al., 2022)	Supervised	0.838
Malite-HRF (Anand et al., 2025)	Supervised	0.841
DTMIC (Kumar and Janet, 2022)	Transfer learning + supervised fine-tuning	0.894
SHERLOCK (Seneviratne et al., 2022)	Self-supervised pretraining + supervised fine-tuning	0.854
TrustBoot	Semi-supervised (only negative labels)	0.883

Table 2: Comparison of TrustBoot against selected malware detection baselines for binary classification on the MalNet-Image dataset. Despite requiring only few negative labels, TrustBoot achieves competitive performance compared to fully supervised state-of-the-art methods.

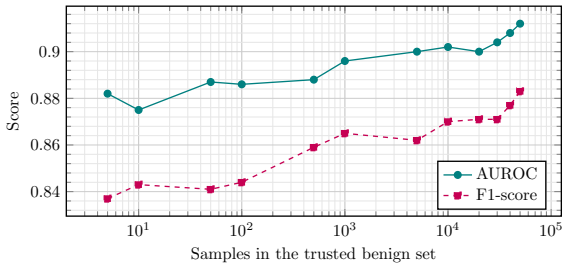

Figure 5: Impact of the number of available trusted benign samples on AUROC and F1-Score. While a positive correlation is observed, the model achieves competitive performance even with several orders of magnitude fewer trusted benigns.

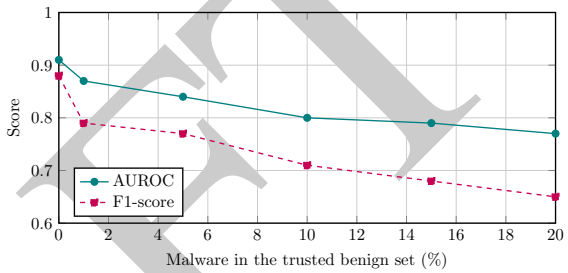

Figure 6: Impact of contamination in the trusted benign set on AUROC and F1-Score. The model gracefully degrades under moderate contamination, but quickly becomes unreliable when the contamination grows too large.

labeling requirements. In order to study this trade-off, the size of $\mathcal{B}$ is varied spanning several orders of magnitude, and the resulting AUROC and F1-Score are reported in Figure 5. In accordance with intuition, larger trusted benign sets yield better performance, with a maximum gap of approximately 4 points in AUROC and F1-Score between the smallest (5 samples) and largest (50000) trusted sets. While this is not a negligible gap, it is worth noting that even with very small core sets, TrustBoot is still competitive with the baselines in Table 2. This behavior indicates that TrustBoot can be rapidly deployed with minimal labeling effort, maintaining most of its capabilities, and provide protection on par with the state-of-the-art. Meanwhile, as more trusted benigns are curated over time, the model can be updated to further improve its detection performance. This property is highly desirable in practical deployment scenarios, where rapid time-to-deployment are critical to mitigate emerging threats.

Finally, a crucial practical consideration is the robustness of TrustBoot to contamination in the trusted benign set $\mathcal{B}$. Given that the trusted benign set is manually curated, there is always a risk of mistakenly (or maliciously) including some malware samples in $\mathcal{B}$. In order to study the impact of such contamination, a sensitivity analysis is conducted by varying the fraction of contaminated benigns from 1% to 20%, and the resulting model perofrmance is reported in Figure 6. As the contamination increases, the performance metric steadily degrade, with the AUROC dropping below 0.8 when more than 10% of the trusted benigns are contaminated. This behavior is expected, as contaminated benigns bias the learned representations and introduce noise in the pseudo-labeling step.

Remark: Based on the sensitivity analyses reported above, prioritizing the assembly of a small, rigorously vetted trusted benign set rather than a larger but potentially contaminated collection seems advisable. Empirically, a compact and highly curated reference set (on the order of dozens of samples) provides more reliable priors for pretraining and pseudo-labeling, reduces the risk of representation bias introduced by contaminated examples, and yields improved downstream detection performance for a given labeling effort.

4.3 Ablation study

In order to thoroughly assess the contribution of each component in the TrustBoot pipeline, an ablation study is conducted by selectively removing key components and measuring the resulting performance degradation. In particular, the following four configurations are evaluated and compared against the full pipeline:

- **Step 1 (Autoencoder only)**: only the MSE reconstruction objective is used; no contrastive term is included. This measures the discriminative power of pure reconstruction errors without contrastive shaping of the latent space.
- **Steps 1 and 2 (Autoencoder + contrastive training)**: after the MSE minimization step, the model is refined over several iterations where at each iteration the unlabeled samples with the highest reconstruction error are used to optimize Eq. (4). No pseudo-labeling step is performed. This configuration isolates the impact of iterative exposure to strongly anomalous candidates without committing to potentially noisy labels.
- **Steps 1 and 3 (No contrastive training)**: after the MSE minimization step, the model is directly fine-tuned using the samples pseudo-labeled samples. Instead of relying on the projection head, pseudo-labels are assigned by fitting a three-component GMM directly to the reconstruction errors in a semi-supervised fashion. This configuration quantifies the importance of the contrastive loss in shaping the latent space for effective pseudo-labeling.
- **Steps 1, 2, and 3 (Random $\mathcal{A}$ selection)**: the TrustBoot pipeline is altered to replace the threshold-based selection of $\mathcal{A}$ with a random selection of unlabeled samples at each iteration. Given that the model obtained after step 1 has extremely weak discriminative power, it could be hypothesized that random selection of candidates would perform similarly to the percentile-based selection. Instead, his configuration shows that, even if Step 1 yields a model with extremely weak discriminative power, it can still provide useful information to guide selection of informative samples for refinement.

The results of the ablation study are summarized in Figure 7.

This ablation study shows that both the contrastive loss and the pseudo-labeling loop contribute substantially: removing either component degrades AUROC by several points. Additionally, the large improvements after pseudo-labeling depend strongly on conservative candidate selection and confidence-driven acceptance: selecting a controlled percentile and requiring a high latent-space confidence minimizes propagation of noisy labels. If, instead, $\mathcal{A}$ is selected randomly at each iteration, the model fails to improve meaningfully over the initial autoencoder baseline, and indeed worsens compared to skipping the contrastive step altogether. This behavior confirms that even a weak initial model can provide useful priors if only high-confidence candidates are selected for refinement.

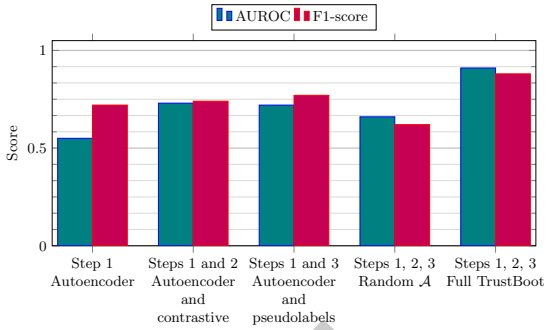

Figure 7: Contribution of the individual pipeline components to final detection performance. The further refining steps and pseudo-labeling yield substantial gains over the initial autoencoder-only baseline. Skipping the contrastive training step significantly degrades performance, as does random selection of anomaly candidates rather than percentile-based selection.

On top of the anomalous sample selection in the second step, even the third step entails assuming some risk of noisy labels due to the pseudo-labeling process. In order to quantify the impact of this risk, an oracle ablation is performed where ground-truth labels are used to select $\mathcal{P}$ and $\mathcal{N}$ instead of relying on the pseudo-labeling through the Gaussian mixture model on the projected latent space. This oracle ablation quantifies the potential upper bound of performance achievable if the pseudo-labeling step were perfectly accurate (i.e., a fully supervised setting). The result of this analysis is reported in Figure 8, which compares the AUROC progression across the 20 training iterations of Step 3 for the real pipeline vs the oracle-based selection.

Both models start at 0.732 AUROC at iteration 0, corresponding to the model obtained after Step 2. The availability of 100% clean labels noticeably speeds up convergence, giving a 11 point boost in AUROC after the first iteration, and reaching 0.92 AUROC after 10 iterations. In the last 10 iterations, the performance gain is marginal, with a 1-point increase, indicating that the model has mostly converged. By contrast, increasing the dataset size via pseudo-labeling increases the AUROC by 7 points in the first iteration, indicat-

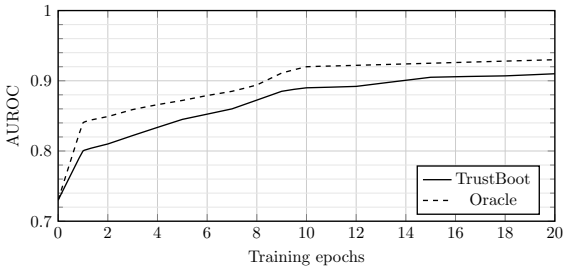

Figure 8: AUROC progression across training iterations on the pseudo-labeled set (Step 3) for the real pipeline vs an oracle-based selection of $\mathcal{P}$ and $\mathcal{N}$ using ground-truth labels. The oracle-based selection yields a notable boost in performance especially in early rounds, speeding up convergence to a high AUROC.

ing that the larger dataset is greatly beneficial even if some label noise is present. Afterwards, the performance gain per iteration is smaller but steady, settling just 2 points below the oracle baseline, indicating that the model is able to gradually refine its representations and generalize effectively despite the presence of some noisy labels. This analysis confirms that the pseudo-labeling step is effective at improving detection performance and, even if some label noise is inevitable, the model can still learn useful discriminative patterns from the augmented dataset, converging to a strong final performance.

5 Conclusions

This work presented TrustBoot, a novel semi-supervised framework designed to overcome the critical limitations of labeled data scarcity and rapid threat evolution in malware detection. By employing visual encodings of binary executables and bootstrapping trust from a small, carefully vetted, trusted benign set, TrustBoot established an effective anomaly-based malware detection pipeline. Experimental validation using a large-scale Android malware dataset confirmed its efficacy, demonstrating competitive detection performance while dramatically reducing the need for intensive, manual dataset curation and feature engineering, achieving competitive performance with minimal upfront labeling cost; TrustBoot thus offers a practical and scalable solution for maintaining high-quality security systems in dynamic environments. Empirically, a compact and highly curated reference set provides reliable priors for pretraining and pseudo-labeling, reduces the risk of representation bias introduced by contaminated examples, and yields improved downstream detection performance for a given labeling effort. Future works will explore extensions of the framework to automatically maintain up-to-date models in production settings, where concept drift and emerging threats are common, as well as techniques to improve the robustness of TrustBoot in order to mitigate the impact of malicious contamination within the trusted benign set , also including the integration with other distributed reputation management systems (Agate et al., 2021; Agate et al., 2016).

Acknowledgments

Part of the experimental evaluation performed for this work was carried out during the Master's thesis of Edoardo Terranova at University of Palermo, Italy. The authors would like to thank him for his dedication and assistance in collecting experimental data and implementing the initial version of the framework.

This work was partially funded by the European Union Next-Generation EU (PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR) – MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.3) – Progetto "Future Artificial Intelligence - FAIR" PE00000013, CUP J73C24000060007

REFERENCES

Abdel-Basset, M., Hawash, H., Sallam, K. M., Elgendi, I., Munasinghe, K., and Jamalipour, A. (2022). Efficient and lightweight convolutional networks for iot malware detection: A federated learning approach. *IEEE Internet of Things Journal*, 10(8):7164–7173.

Agate, V., De Paola, A., Drago, S., Ferraro, P., and Lo Re, G. (2024). Enhancing iot network security with concept drift-aware unsupervised threat detection. In *2024 IEEE Symposium on Computers and Communications (ISCC)*. IEEE.

Agate, V., De Paola, A., Ferraro, P., and Lo Re, G. (2025). MIDES: A multi-layer intrusion detection system using ensemble machine learning. *International Journal of Intelligent Networks*.

Agate, V., de Paola, A., Lo Re, G., and Morana, M. (2016). A simulation framework for evaluating distributed reputation management systems. *Advances in Intelligent Systems and Computing*, 474:247 – 254.

Agate, V., De Paola, A., Lo Re, G., and Morana, M. (2021). A simulation software for the evaluation of vulnerabilities in reputation management systems. *ACM Transactions on Computer Systems*, 37(1-4).

Agate, V., Drago, S., Ferraro, P., and Lo Re, G. (2022). Anomaly detection for reoccurring concept drift in smart environments. page 113 – 120.

Anand, S., Mitra, B., Dey, S., Rao, A., Dhar, R., and Vaidya, J. (2025). Malite: Lightweight malware detection and classification for constrained devices. *IEEE Transactions on Emerging Topics in Computing*.

Ansari, A. M., Nazir, M., and Mustafa, K. (2024). Smart homes app vulnerabilities, threats, and solutions: a systematic literature review. *Journal of Network and Systems Management*, 32(2):29.

Arp, D., Quiring, E., Pendlebury, F., Warnecke, A., Pierazzi, F., Wressnegger, C., Cavallaro, L., and Rieck, K. (2022). Dos and don'ts of machine learning in computer security. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 3971–3988.

Augello, A., De Paola, A., Giosuè, D., and Lo Re, G. (2025a). SF-AE: Split federated autoencoder for unsupervised iot intrusion detection. In Yang, X.-S., Sherratt, S., Dey, N., and Joshi, A., editors, *Proceedings of Tenth International Congress on Information and Communication Technology*, pages 371–381, Singapore. Springer Nature Singapore.

Augello, A., De Paola, A., Jestin, M., and Lo Re, G. (2025b). A stackelberg approach to federated learning for malware detection. In *CEUR Workshop Proceedings*, volume 3962. ceur-ws. org/Vol-3962/paper36. pdf.

Augello, A., De Paola, A., and Lo Re, G. (2025c). Hybrid multilevel detection of mobile devices malware under concept drift. *Journal of Network and Systems Management*, 33(2):36.

Augello, A., Lo Re, G., Peri, D., and Thiyagalingam, P. (2024). Nep-ids: a network intrusion detection system based on entropy prediction error. In *2024 IEEE 49th Conference on Local Computer Networks (LCN)*, pages 1–9.

Camarda, F., De Paola, A., Drago, S., Ferraro, P., and Lo Re, G. (2025). Managing concept drift in online intrusion detection systems with active learning. In *CEUR WORKSHOP PROCEEDINGS*, volume 3962. CEUR-WS.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PmLR.

Dambra, S., Han, Y., Aonzo, S., Kotzias, P., Vitale, A., Caballero, J., Balzarotti, D., and Bilge, L. (2023). Decoding the secrets of machine learning in malware classification: A deep dive into datasets, feature extraction, and model performance. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 60–74.

De Paola, A., Drago, S., Ferraro, P., and Lo Re, G. (2024). Detecting zero-day attacks under concept drift: An online unsupervised threat detection system. In *CEUR Workshop Proceedings*, volume 3731.

Freitas, S., Duggal, R., and Chau, D. H. (2022). Malnet: A large-scale image database of malicious software. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 3948–3952.

Gong, D., Liu, L., Le, V., Saha, B., Mansour, M. R., Venkatesh, S., and Hengel, A. v. d. (2019). Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1705–1714.

Hojjati, H., Ho, T. K. K., and Armanfard, N. (2024). Self-supervised anomaly detection in computer vision and beyond: A survey and outlook. *Neural Networks*, 172:106106.

Kumar, S. and Janet, B. (2022). DTMIC: Deep transfer learning for malware image classification. *Journal of Information Security and Applications*, 64:103063.

Li, S., Wang, J., Wang, S., and Song, Y. (2024). PAFE: A lightweight visualization-based fast malware classification method. *Heliyon*, 10(16).

Lindenbaum, O., Aizenbud, Y., and Kluger, Y. (2024). Transductive and inductive outlier detection with robust autoencoders. In *The 40th Conference on Uncertainty in Artificial Intelligence*.

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

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Pendlebury, F., Pierazzi, F., Jordaney, R., Kinder, J., and Cavallaro, L. (2019). TESSERACT: Eliminating experimental bias in malware classification across space and time. In *28th USENIX security symposium (USENIX Security 19)*, pages 729–746.

Qiu, C., Li, A., Kloft, M., Rudolph, M., and Mandt, S. (2022). Latent outlier exposure for anomaly detection with contaminated data. In *International conference on machine learning*, pages 18153–18167. PMLR.

Seneviratne, S., Shariffdeen, R., Rasnayaka, S., and Kasthuriarachchi, N. (2022). Self-supervised vision transformers for malware detection. *IEEE Access*, 10:103121–103135.

Sun, T., Daoudi, N., Pian, W., Kim, K., Allix, K., Bissyande, T. F., and Klein, J. (2025). Temporal-incremental learning for android malware detection. *ACM Transactions on Software Engineering and Methodology*, 34(4):1–30.

Tang, L., Chen, X., Wen, S., Li, L., Grobler, M., and Xiang, Y. (2023). Demystifying the evolution of android malware variants. *IEEE Transactions on Dependable and Secure Computing*, 21(4):3324–3341.

Verkerken, M., D'hooge, L., Wauters, T., Volckaert, B., and De Turck, F. (2022). Towards model generalization for intrusion detection: Unsupervised machine learning techniques. *Journal of Network and Systems Management*, 30(1):12.

Wu, W. (2023). Traffic anomaly detection method based on mislearned autoencoder. In *2023 5th International Academic Exchange Conference on Science and Technology Innovation (IAECST)*, pages 112–116. IEEE.

Zhu, X., Huang, J., Wang, B., and Qi, C. (2021). Malware homology determination using visualized images and feature fusion. *PeerJ Computer Science*, 7:e494.