Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.062439

ARTICLE

Enhancing Malware Detection Resilience: A U-Net GAN Denoising Framework for Image-Based Classification

Huiyao Dong[1] **and Igor Kotenko**[2,*]

[1]Faculty of Information Technology and Security, ITMO National Research University, St. Petersburg, 197101, Russia
[2]Laboratory of Computer Security Problems, St. Petersburg Federal Research Center of the Russian Academy of Sciences, St. Petersburg, 199178, Russia

*Corresponding Author: Igor Kotenko. Email: ivkote@comsec.spb.ru

Received: 18 December 2024; Accepted: 26 January 2025; Published: 06 March 2025

ABSTRACT: The growing complexity of cyber threats requires innovative machine learning techniques, and image-based malware classification opens up new possibilities. Meanwhile, existing research has largely overlooked the impact of noise and obfuscation techniques commonly employed by malware authors to evade detection, and there is a critical gap in using noise simulation as a means of replicating real-world malware obfuscation techniques and adopting denoising framework to counteract these challenges. This study introduces an image denoising technique based on a U-Net combined with a GAN framework to address noise interference and obfuscation challenges in image-based malware analysis. The proposed methodology addresses existing classification limitations by introducing noise addition, which simulates obfuscated malware, and denoising strategies to restore robust image representations. To evaluate the approach, we used multiple CNN-based classifiers to assess noise resistance across architectures and datasets, measuring significant performance variation. Our denoising technique demonstrates remarkable performance improvements across two multi-class public datasets, MALIMG and BIG-15. For example, the MALIMG classification accuracy improved from 23.73% to 88.84% with denoising applied after Gaussian noise injection, demonstrating robustness. This approach contributes to improving malware detection by offering a robust framework for noise-resilient classification in noisy conditions.

KEYWORDS: Malware; cybersecurity; deep learning; denoising

1 Introduction

Malware variants are rapidly becoming one of the most serious threats to IoT systems. According to SonicWall's 2024 Mid-year Cyber Threat Report, malware attacks have increased by 30% over the same period last year, with an increase in IoT malware (+107%) [1]. According to their data, targeted IoT devices have an average attack time of 52.8 h. Hence, malware detection and analysis systems play a pivotal role in identifying and mitigating threats as they emerge, employing various techniques such as signature-based detection, anomaly detection, and heuristics. However, the dynamic nature of malware necessitates the adoption of advanced methodologies to enhance detection accuracy and reduce false positives.

In recent years, machine learning (ML)-based approaches have gained prominence due to their capacity to learn complex patterns associated with malware behavior, and the evolving landscape of malware has prompted researchers to explore innovative approaches for classification, including the utilization of image-based representations of malware binaries. By converting executable files into visual formats, such as

grayscale or RGB images, it becomes feasible to apply deep learning (DL) techniques to classify malware based on its visual characteristics. The conversion of images may involve methods such as extracting features from Local Binary Patterns (LBP) [2] or transforming binary files into spectrogram images for subsequent classification [3]. Deep learning techniques, such as Autoencoders (AEs) and Variational Autoencoders (VAEs), in conjunction with hybrid classifiers, can effectively learn complex patterns present in image-based malware samples [4]. However, the high dimensionality of image representations presents challenges in training models efficiently, as training on such complex data necessitates substantial computational resources and time [5]. Transfer learning, which leverages pre-trained convolutional neural networks (CNNs) that utilize previously acquired knowledge from large datasets, can effectively address this challenge by fine-tuning models on specific malware data [6]. This intersection of image processing and malware analysis opens new frontiers in the search for robust and efficient malware classification systems.

With recent advances in image-based malware analysis, ensuring reliable and effective security measures remains a significant challenge. For example, code injection has been shown to significantly impact the performance of deep learning (DL)-based classifiers. Injecting a code sequence into the beginning or middle of an executable file can alter its image representation. Consequently, the modified pixel representation may introduce patterns or distortions that could confuse or mislead classifiers. Furthermore, many existing approaches show a limited ability to handle obfuscated images, as this critical problem has received insufficient attention in prior research. Obfuscation serves as a crucial mechanism to protect sensitive information by reducing data identifiability [7]. It can pose challenges for malware analysis models, as identifiable features associated with malware have been altered. Although innovative transformation techniques can be employed to convert software into image-like arrays, the introduced noise and complexity necessitate advanced, robust models capable of tolerating and extracting essential information from altered and dynamic structures. Popular approaches such as data augmentation, image denoising algorithms, and adversarial training have proven helpful in mitigating obfuscation and noise in malware samples. However, these approaches are still not effective in addressing malware-specific challenges.

Denoising for malware imagery data presents unique and significant challenges due to the distinct characteristics of malware. Malware images, derived by transforming binary files into visual representations, encode subtle patterns within the file content. These small-scale structural features are critical for accurate malware classification, but the presence of noise can easily disrupt these localized features, complicating the reconstruction of the original signal. In most denoising applications, noise is incidental (e.g., sensor noise in medical images or environmental noise in natural images). In contrast, noise in malware imagery data is deliberately introduced as an evasion technique, making it considerably more difficult to detect and remove. Conventional denoising methods are generally inadequate against the adversarial and structured nature of such noise. Additionally, malware classification is hindered by factors such as class imbalance, high dataset diversity, and the absence of pre-labeled "ground truth" in the malware imagery domain. Consequently, denoising models must not only generalize beyond synthetic noise distributions but also effectively restore meaningful patterns, particularly for malware samples from underrepresented classes. Although denoising has been extensively studied in domains such as medical imaging and natural images, it remains a relatively unexplored area in the context of malware imagery. Previous work employing a Variational Autoencoder-Generative Adversarial Network (VAE-GAN) approach suggested that denoising techniques could be integrated into the architecture to enhance the robustness of malware detection, although specific denoising techniques were not explicitly mentioned [8]. Similarly, Bensaoud et al. [9] emphasized that a multi-task learning-based framework could improve classification performance and potentially involve handling noise in the data. However, they did not elaborate on the denoising technique or encryption detection.

In this paper, we propose an image denoising technique based on the U-Net and GAN frameworks. Initially, we implemented a noise addition technique to simulate obfuscated and noisy malware samples. Subsequently, we designed a CNN-based U-Net model, incorporating multiple encoding and decoding blocks for effective feature extraction and image reconstruction. For a comprehensive comparison, we used several classifiers to perform the malware classification task on original samples, samples with noise addition, and denoised samples. We selected two public datasets with multiple classes to test the model's robustness and adaptability. The experimental results demonstrate that while the noise addition simulation significantly affects the generalizability of the classifiers, the application of the U-Net denoising model can enable the classifiers to yield results comparable to those obtained from the original, noise-free samples.

The remainder of the paper is organized as follows. Section 2 summarizes existing research in image-based malware analysis; Section 3 outlines the proposed techniques, including noise simulation and the denoising model; Section 4 presents the experimental results; Section 5 discusses the implementation and application of the model, along with future plans. Finally, the paper concludes with a summary of our findings.

2 Related Work

ML models can classify and predict malware variants with remarkable precision, thus enhancing traditional detection techniques. By extracting features from Executable and Linkable Format (ELF) files and opcode sequences, it becomes feasible to identify malicious behaviors in Internet of Things (IoT) devices using algorithms such as Random Forest, Support Vector Machines (SVM), and Neural Networks [10]. Furthermore, it is practical to utilize ensemble learning models to analyze system calls generated by Android applications to identify malicious behaviors [11]. In the domain of image-based malware classification, pre-trained CNN models, particularly EfficientNet, have emerged as prevalent tools for learning malware image representations with standardized image widths [12]. Dao et al. [13] proposed a lightweight architecture that combined small CNN models with an Variational Autoencoder, resulting in superior performance and efficiency in malware classification. Further innovations have explored hybrid approaches and alternative architectures to improve classification accuracy. Thakur et al. [14] proposed a robust malware classification framework that utilizes a comprehensive voting scheme among various CNN-LSTM classifiers. In contrast, Karat et al. [15] advocated the use of similar hybrid classifiers, specifically CNN-LSTM, but placed the emphasis on behavioral analysis as a crucial element in identifying Zero-Day malware. Their approach incorporated extensive information, including log parsing and API monitoring, to enable real-time behavioral analysis. Dong [16] proposed a sophisticated hybrid approach combining GAN for synthetic oversampling with a simplified Vision Transformer (ViT)-based classifier for malware image classification.

Although datasets containing obfuscated malware variants and specimens with subtle code injections remain scarce, these adversarial scenarios can be systematically simulated through the application of controlled noise addition to malware imagery data. This methodology enables the evaluation of the resilience of the detection system against potential obfuscation techniques and minor code modifications, while providing a reproducible framework for robustness assessment. Common approaches in this field include hybrid noise addition, blind denoising, and the use of additive white noisy images or real-world noisy images. Although CNNs and GANs are commonly used in image denoising research, as highlighted in a recent review [17], several innovative techniques have been introduced. Su et al. [18] proposed a CNN-based multi-scale cross-path concatenation residual network with multiple skip connections specifically for Poisson image denoising. Similarly, Jifara et al. [19] utilized a CNN-based deep model featuring residual connections to enhance medical image denoising. Yuan et al. [20] introduced HSID-CNN, a nonlinear mapping technique for noisy and clean images, utilizing a combined spatial-spectral deep CNN model. This model executes multiscale

feature extraction and multilevel feature representation to effectively capture spatial-spectral features. Although the development of denoising techniques in AI, particularly in medical image processing, has progressed for several years with demonstrably effective outcomes, there remains a scarcity of applications using image denoising techniques for malware detection and classification. Li et al. [21] presented a malware classification method incorporating a dedicated denoising module through dynamic analysis and embedding techniques to encode API sequences, introducing a soft threshold mechanism for active noise filtering and employing a bidirectional LSTM model for classification. Kumi et al. [22] proposed a Block-matching and 3D filtering algorithm alongside a deep CNN-based denoising technique to address adversarial examples. A recent study aimed at enhancing the robustness of classifiers against adversarial attacks introduced the Differential Privacy-Based Noise Clipping defense mechanism [23], assisting classifiers from being misled by data poisoning and gradient-based data poisoning. It modified the training framework to render models inherently resistant to adversarial noise. All of these techniques were integrated into the training phase of the deep learning classifier, directly altering the learning process. However, these studies did not investigate various types of noise and provided limited discussion of performance analysis both before and after noise injection, as well as after the application of denoising techniques, which represents a significant gap.

The cited work underscores the SOTA techniques in image-based malware classification and the effectiveness of denoising techniques in various fields, but it reveals a notable gap in the exploration and application of denoising methods for malware detection and classification tasks. This research addresses this need by focusing on Gaussian and Speckle noise types, employing a U-Net architecture for denoising. The proposed model is evaluated using multiple CNN-based classifiers and tested across two datasets to rigorously assess both robustness and generalizability. Ultimately, this research aims to fill the existing gap by providing a comprehensive analysis of denoising techniques in the context of malware classification, contributing to the advancement of noise injection strategies in security applications.

3 Methodology

3.1 Noise Simulation for Obfuscation Techniques

Obfuscation fundamentally alters the code structure of executable files through various techniques, such as introducing noise, varying features due to alterations in code and data order and form, increasing data complexity, and creating pattern and signature misalignment. After being transferred to an image-like array, as a significant number of the original bytes are inserted or altered, the likelihood of losing critical features that differentiate malware from benign software increases, potentially resulting in a decline in detection accuracy.

In the implementation of image-based obfuscation simulation, multiple technical approaches like pixel-level transformations, geometric transformations, and adversarial concealment methodologies can be employed [24]. This research implements two noise addition methods to simulate obfuscation: Gaussian noise and Speckle noise.

Gaussian noise represents a fundamental concept in signal processing and image simulations, as it accurately approximates the natural noise patterns inherent in real-world signals, particularly in imaging systems. This type of noise is ubiquitous, manifesting itself in virtually all signal transmissions, and is particularly important for obfuscation and privacy simulations due to its universality and versatility [25]. The distinguishing characteristic of Gaussian noise lies in its normal distribution properties, rendering it particularly suitable for simulating random perturbations across diverse applications. Within the domain of image obfuscation and simulation methodologies, Gaussian noise serves as an indispensable tool for image

data processing, functioning both to evaluate the resilience of denoising algorithms and as a mechanism to obscure sensitive information in privacy-critical images.

The implementation of Gaussian noise addition follows several steps. Firstly, the image data is normalized to a range of [0, 1] by dividing the image array by 255.0 to prevent pixel overflow when noise is added. Then, generates Random Gaussian noise with a specified mean and a standard deviation. Lastly, the noise is added element-wise to the normalized image, producing the noisy image. Mathematically, the process of transformation from original image to noisy one can be represented using Eq. (1).

$$I_{noisy}(x, y, c) = I_{original}(x, y, c) + N(x, y, c) \tag{1}$$

In which the term $I(x, y, c)$ represents the pixel value of an image at position (x, y) for channel c, and $N(x, y, c)$ represents the Gaussian noise added to each pixel. This noise is sampled from the normal distribution as shown in Eq. (2).

$$N(x, y, c) \sim N(\boldsymbol{\mu}, \boldsymbol{\sigma}^2) \tag{2}$$

In which the μ stands for the mean and $\boldsymbol{\sigma}$ stands for the standard deviation of the distribution. In this research, we set the mean value at 0 and standard deviation as a parameter.

Speckle noise, in contrast to Gaussian noise, exhibits multiplicative characteristics and originates from coherent processes such as laser or radar imaging, where constructive and destructive interference patterns introduce signal corruption [26]. This noise variant demonstrates relevance in simulating real-world distortions encountered in medical imaging applications, including magnetic resonance imaging (MRI), and in remote sensing systems. Its multiplicative properties alter pixel intensities in proportion to their original values, rendering it an essential component in realistic obfuscation simulations that aim to model these specific imaging scenarios. The mathematical representation can be described as Eq. (3).

$$I_{noisy}(x, y, c) = I_{original}(x, y, c) + N(x, y, c) \times I_{original}(x, y, c) \tag{3}$$

In which speckle noise sampled from a standard normal distribution, scaled by the noise factor. This operation creates multiplicative distortion proportional to the original image's pixel values.

Speckle noise, being prevalent in numerous real-world imaging systems, provides an accurate simulation of these environments when incorporated into experimental frameworks [27]. The multiplicative nature of speckle noise, which inherently correlates with original pixel values, renders it particularly effective in obfuscating fine features within images, such as textures and high-frequency details.

To demonstrate the effects of different noise types, we compared original image samples with their corresponding versions containing Gaussian and speckle noise. In both cases, the $noise\_factor$, namely the standard deviation value, was set to a high value of 0.4 to clearly illustrate the distinctions. As shown in Fig. 1, Gaussian noise introduces dynamic perturbations that substantially obscure the original texture, whereas speckle noise maintains the underlying textural characteristics. This distinction arises from their fundamental mathematical properties and their respective interactions with the original image. Gaussian noise is characterized by its additive nature, where noise is independently applied to each pixel irrespective of its original intensity, uniformly affecting all image regions regardless of their intensity or texture. This superimposition of a completely independent random pattern effectively masks the original texture. In contrast, speckle noise exhibits multiplicative properties, where the noise effect is proportional to the original pixel intensity. This results in texture preservation, as low-intensity regions experience minimal noise impact

while high-intensity regions undergo more substantial modifications. Consequently, this proportional relationship facilitates the preservation of original texture patterns despite the noise introduction.

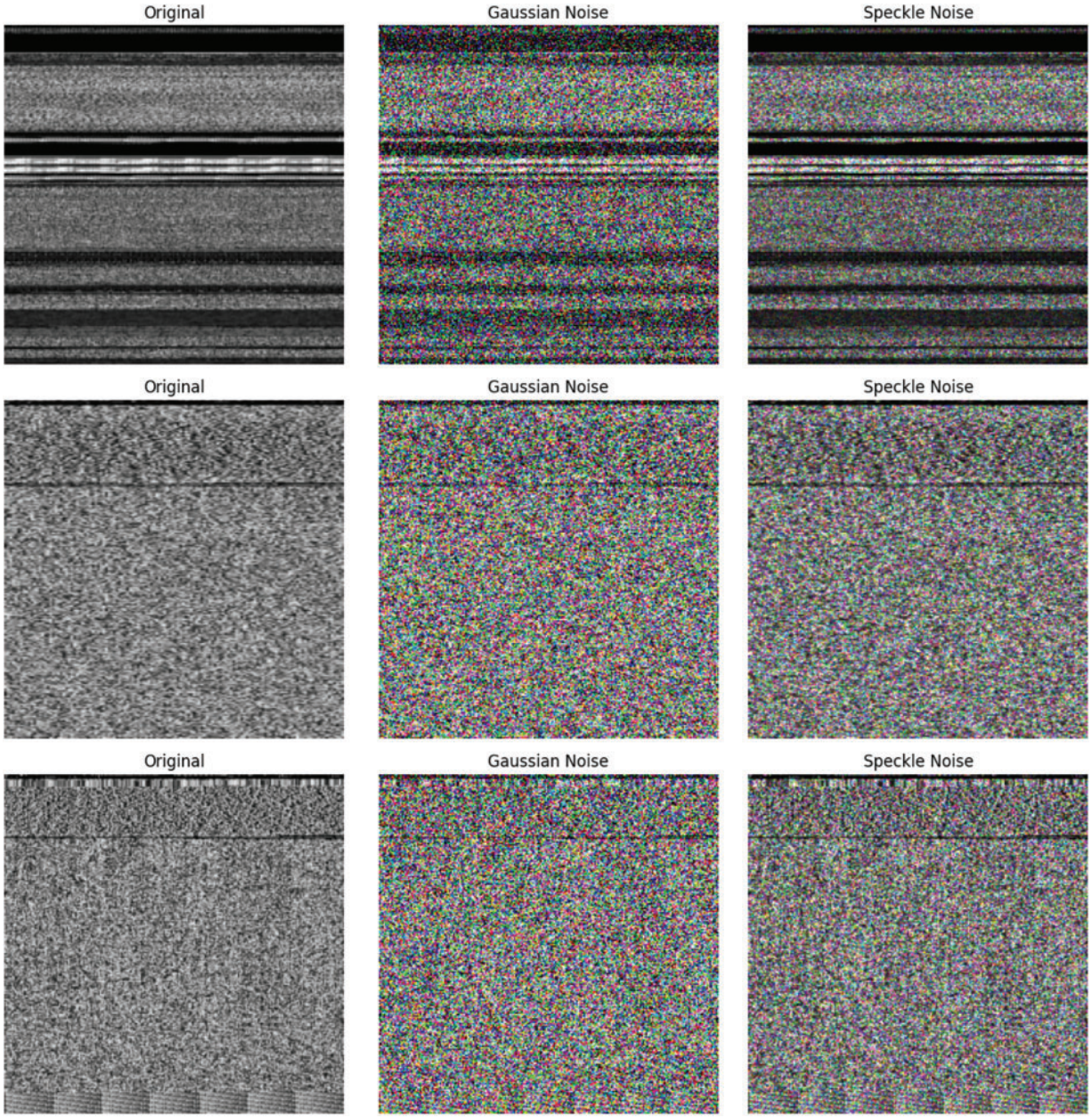

Figure 1: Malware samples with different noise addition

3.2 Denoising Workflow with U-Net

The denoising workflow functions as a comprehensive strategy for mitigating image-based obfuscation. In the initial stage, we conduct extensive training of the denoising model on a substantial dataset, enabling it to effectively generate denoised images from those that have undergone noise addition. After the model has converged, the trained weights are saved for future usage. Subsequently, we apply this model following data preprocessing and prior to classification to execute the denoising operation.

The U-Net architecture is widely employed in computer vision for both image denoising [28] and super-resolution [29] tasks, owing to its proficiency in learning spatial hierarchies and context-rich features. This architecture comprises a deep encoder and a deep decoder. The encoder functions as a feature extractor, wherein successive convolutional layers and max pooling operations diminish the spatial dimensions of the image while preserving meaningful high-level features. Conversely, the decoder progressively upsamples these features to restore the original resolution. In the context of image denoising, the encoder facilitates the understanding of global noise patterns, while the decoder reconstructs noise-free images by restoring local fine-grained features. Additionally, U-Net incorporates skip connections that directly link layers in the encoder with their corresponding layers in the decoder, thereby enabling the model to recover fine spatial details that may be lost during the downsampling process.

In this study, we implement a CNN-based U-Net and optimize it through Mean Squared Error (MSE) loss for reconstruction evaluation. For a given noisy image x, the U-Net generates an output image $\hat{y}$ such that $\hat{y} \sim y$, where y represents the expected, original malware image without noise. The model minimizes the discrepancy between the reconstructed image $\hat{y}$ and the ground truth y. The denoising loss function is the MSE between the predicted clean image $\hat{y}$ and the ground truth clean image y. The MSE is given by:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 \tag{4}$$

where N denotes the total number of pixels. Through optimization, the MSE loss can minimize the pixel-wise error across the entire image.

Fig. 2 presents the proposed U-Net architecture. The major components are encoding modules, decoding modules, skip connections, and the final convolution layer for shape adjustment.

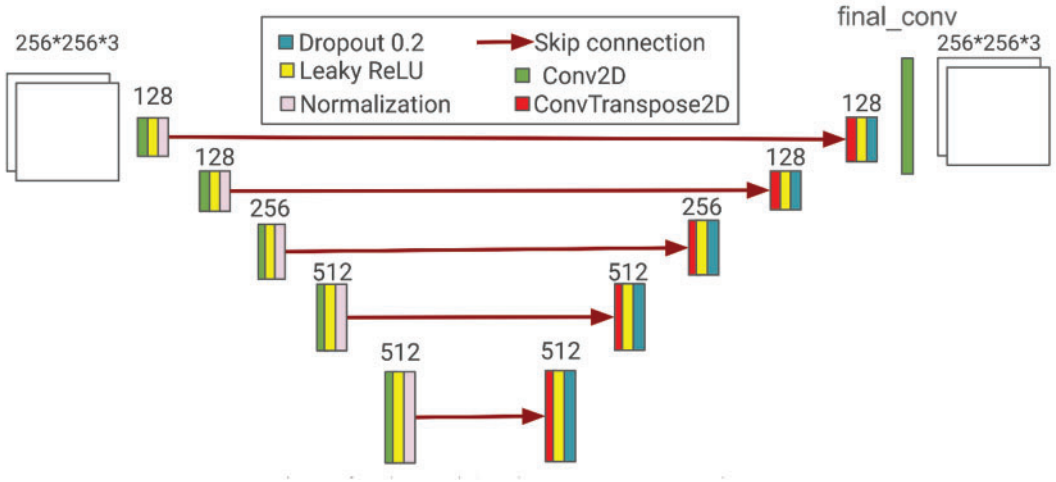

Figure 2: The model architecture of proposed U-Net

Encoding module captures spatial context and semantic features by progressively reducing spatial resolution. Within each module, convolutions are applied to decrease spatial resolution while increasing feature depth, followed by batch normalization and Leaky ReLU activation. The activation function introduces non-linearity with a slight allowance for negative values, controlled by a negative slope of 0.2. This module is employed to contract images while preserving key features and structural information.

Decoding module reconstructs the input by progressively increasing spatial resolution while merging features from the encoder through skip connections. First, a transposed convolution layer is employed for feature upscaling. Next, a Leaky ReLU activation is applied. A dropout layer is then incorporated to randomly deactivate neurons, thereby introducing regularization and preventing overfitting. This design facilitates the expansion of features in the decoder path while enhancing robustness through regularization.

The U-Net is constructed in a modular format, incorporating distinct units for both convolution and transposed convolution layers. To achieve downsampling, five encoding modules are implemented, which halve the spatial resolution while simultaneously increasing the feature depth. In this context, the feature depth progresses as follows: 128, 128, 256, 512, and 512. The output feature maps from each block are retained to enable skip connections. For the upscaling procedure, five decoding modules are employed to double the spatial resolution while reducing the feature depth to match the input size. In a reverse manner, the feature depth decreases in the following sequence: 512-512-256-128-128. Feature maps from the corresponding encoding blocks are concatenated at each stage to preserve spatial details. Finally, a 1×1 convolution is utilized to map the final output to the desired number of channels (e.g., three for RGB images).

The design incorporates reusable and flexible modules, thereby enhancing adaptability to various image dimensions. The activation function employed, Leaky ReLU, mitigates the issue of dying ReLU, which occurs when neurons become inactive due to all-zero gradients, thereby promoting more dynamic training. Additionally, both batch normalization and dropout techniques contribute to stabilizing the training process and enhancing generalization. The symmetry between the encoder and decoder, along with the implementation of skip connections, ensures that the reconstruction phase effectively eliminates unwanted noise while preserving the essential features learned in each encoding module.

To enhance the resilience of these methods, several potential improvements can be implemented: (1) employing obfuscated files for pre-classification analysis to develop discriminative features that effectively distinguish benign applications from malware; (2) leveraging adversarial ML techniques to simulate attacks on classification models using obfuscated malware samples or images augmented with random noise, which can mimic obfuscation strategies. Furthermore, extensive research in information-hiding techniques could be integrated into imagery-based malware analysis, utilizing methods such as patch-line and fuzzy clustering-line to uncover hidden content or anomalies within the images.

3.3 Training with GAN Framework

In the context of combating noise injection, using an encoder-decoder structure is feasible, but implementing GAN framework with a simple discriminator can enhance the denoising performance due to the adversarial training paradigm.

The primary distinction lies in the optimization of the loss function. When utilizing U-Net in isolation, the focus is on enhancing the noise robustness of models during training. This approach prioritizes minimizing the reconstruction error between generated images and their corresponding expected images, without incorporating noise injection. Consequently, while the model gains the ability to handle noisy data, it lacks generative capabilities; specifically, it cannot be assured of producing realistic samples when compared to noise-free data. In contrast, a GAN designed for denoising tasks explicitly comprises two networks: a generator that produces denoised data and a discriminator that differentiates between real (clean) and generated (denoised) data. These networks are trained adversarially, leading to a model that learns the underlying data distribution and can reconstruct highly accurate outputs, even in high-noise environments. Through this adversarial training framework, the model can perform denoising in a nuanced and adaptive manner-an achievement that pure error-based strategies often struggle to attain [30].

Fig. 3 illustrates the complete GAN framework and the model structure of the discriminator. Following the input configuration, a convolutional block is employed for feature extraction, consisting of two 2-dimensional convolutional layers with 64 units, batch normalization, and Leaky ReLU activation. Subsequently, an adaptive average pooling layer is applied to reduce the size of the feature map. A fully connected layer with 1024 units is then utilized to flatten the learned features. This is followed by a Leaky ReLU activation, leading to an output layer that predicts the probability of the input being real or fake. The decision to employ a simplified discriminator within our GAN framework is driven by the objective of stabilizing the training process while prioritizing the generator's capabilities. A less complex discriminator reduces the likelihood of overfitting and allows for a more balanced learning dynamic between the generator and discriminator. This approach facilitates a more effective training regime, enabling the generator to develop robust representations without being excessively constrained by the discriminator's judgments.

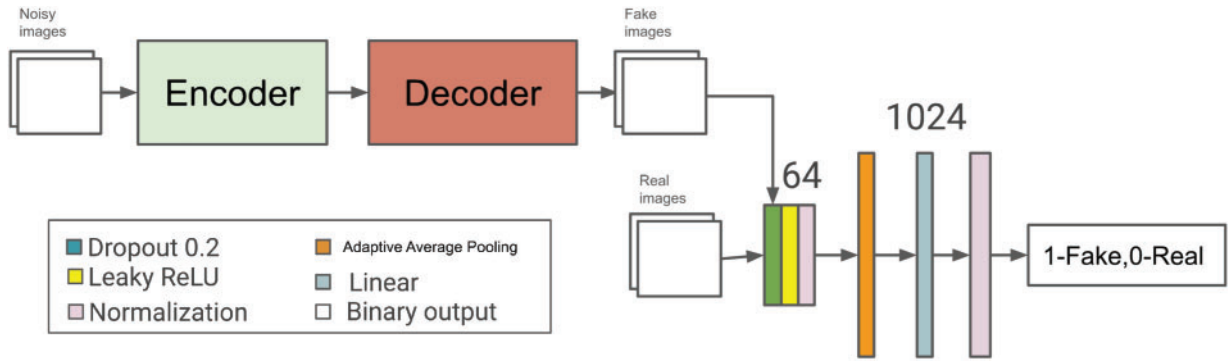

Figure 3: The proposed GAN framework and model structure for discriminator

During the training process, there are two losses to be optimized: generator loss and discriminator loss, which can be represented as Eqs. (5) and (6).

$$L_G = \mathbb{E}[\mathrm{MSE}(G(x), y)] \tag{5}$$

$$L_D = -\frac{1}{2}\mathbb{E}[\log(D(y)) + \log(1 - D(G(x)))] \tag{6}$$

In which $G(x)$ represents the generated image, y is the target image, $D(y)$ is the discriminator's output for real images and $D(G(x))$ is the discriminator's output for generated images.

Before training, we prepared a dataset comprising paired original and noisy images containing malware patterns, wherein the noisy images serve as input and the corresponding noise-free images function as the generator's output. The training workflow begins with the initialization phase, where the U-Net generator and discriminator are defined. The optimizer employed is AdamW [31], initialized with a learning rate of 0.001. To progressively adjust the learning rate during training and mitigate overfitting, the Cosine Annealing learning rate scheduler is utilized. This algorithm effectively reduces the learning rate in a smooth manner, mimicking the natural traversal of the optimization landscape. Additionally, it enables efficient computation by employing an automatic adaptive schedule and reduces the need for extensive hyperparameter tuning. Utilizing an initial learning rate of 0.001, a minimum learning rate of 0.00001, and a training epoch of 100, the learning rate adjustment curve is depicted in Fig. 4.

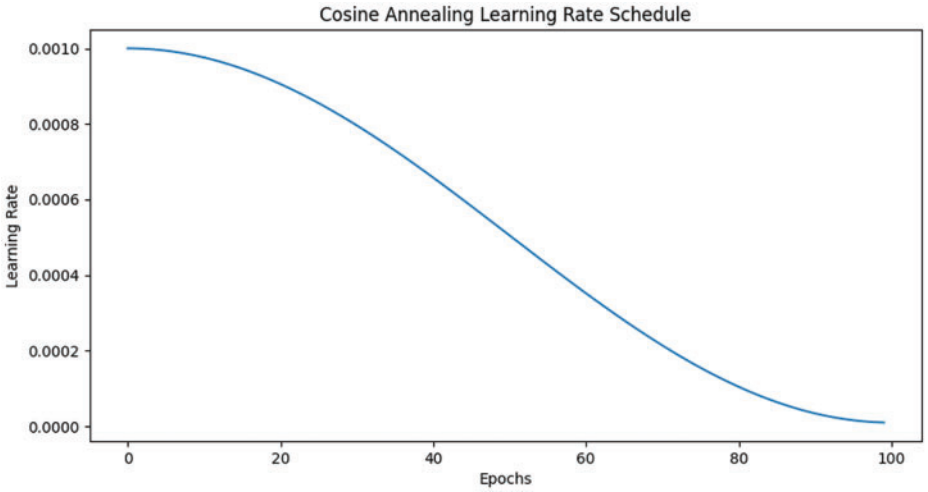

Figure 4: The learning rate adjusted by scheduler

The complete training process is outlined in Algorithm 1. Following the initialization of the generator and discriminator, the training algorithm proceeds iteratively through the following steps: training the generator, training the discriminator, validating model performance, and updating the learning rates.

The primary tasks of the generator involve generating fake images to produce denoised versions of noisy inputs, receiving feedback from the discriminator regarding classification performance on both fake and real images, and optimizing based on the MSE loss. For the discriminator, the loss is composed of two components: the loss associated with classifying real images as "0" and the loss for classifying fake images as "1." The mathematical expressions for these are detailed below:

$$\text{real\_loss} = \text{criterion}(\text{real\_output}, \text{ones\_like}(\text{real\_output})) \tag{7}$$

$$\text{fake\_loss} = \text{criterion}(\text{fake\_output}, \text{zeros\_like}(\text{fake\_output})) \tag{8}$$

$$d_{loss} = \frac{\text{real\_loss} + \text{fake\_loss}}{2} \tag{9}$$

The adversarial objective defined as Eq. (10) is highly effective for denoising tasks because it facilitates learning a robust mapping between noisy input data and the clean underlying distribution.

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \tag{10}$$

Algorithm 1: Training algorithm for GAN

1: **Input & Defined components:**

- x: Set of noisy malware images
- G: Generator network (UNetModel) initialized with parameters θ_G
- D: Discriminator network initialized with parameters θ_D
- η_G, η_D: Learning rates for G and D
- Optimizer: AdamW for both G and D
- Scheduler: Cosine Annealing Learning Rate scheduler
- num_epochs: Number of training epochs

(Continued)

Algorithm 1 (continued)

2: **Output:**

- Trained generator network G
- Trained discriminator network D

3: **for** $epoch$ = 1 to num_epochs **do**

4: **Generator Training Phase:**

- Generate fake images: $\hat{x} \leftarrow G(x)$
- Compute generator loss: $L_G \leftarrow \text{GAN\_Loss\_Generator}(D(\hat{x})) + \text{Reconstruction\_Loss}(\hat{x}, x)$
- Update generator parameters: $\theta_G \leftarrow \theta_G - \eta_G \cdot \nabla_{\theta_G} L_G$

5: **Discriminator Training Phase:**

- Classify real and fake images: $D_{\text{real}}(x)$ and $D_{\text{fake}}(\hat{x})$
- Compute discriminator loss: $L_D \leftarrow -\mathbb{E}[\log(D_{\text{real}}(x))] - \mathbb{E}[\log(1 - D_{\text{fake}}(\hat{x}))]$
- Update discriminator parameters: $\theta_D \leftarrow \theta_D - \eta_D \cdot \nabla_{\theta_D} L_D$

6: Apply learning rate scheduling:

- Update learning rates for G and D using cosine annealing scheduler.

7: **end for**

In the adversarial framework, the generator aims to produce outputs that are indistinguishable from true clean images, while the discriminator seeks to differentiate between real and generated outputs. Through adversarial training, the generator progressively learns to model the underlying clean data distribution. This objective is effective for denoising tasks, as the generator not only minimizes pixel-level differences (e.g., MSE loss) but also captures higher-order structures that align the generated outputs with the true clean image distribution. Furthermore, although we simulate two representative types of noise, adversarial learning does not depend on explicit assumptions about the noise distribution. Instead, it has the capacity to eliminate discrepancies between the input noisy data and their corresponding clean counterparts, demonstrating its strong generalizability. Overall, the adversarial objective achieves a balance between realism and accuracy, making it especially advantageous for image-based denoising tasks where traditional methods often fall short.

4 Experiments

4.1 Datasets

We tested the proposed denoising technique on two malware imagery datasets in a multi-classification environment to fully validate the suggested techniques. The first dataset, Big2015 [32], was proposed by Microsoft and includes 10,470 malware samples represented as disassembled assembly code and byte-level features from 9 distinct malware families. A class label, a 20-character hash value, and an identifier are used to describe each file. Fig. 5 presents the malware types, class labels and their proportions in the training and testing sets. The second dataset, MALIMG [33], was proposed for research on visualizing malware imagery. MALIMG contains 25 different malware families, which provides a foundation for multi-class classification tasks. This diversity reflects, to some extent, the variability in malware behaviors and families encountered in practice. Fig. 6 presents the class labels and their initial proportions in the training set. Both datasets represent, to some extent, the challenges associated with static malware analysis, particularly in the detection and classification of binary files with established structures from previously identified malware families. Although these datasets do not capture the behavioral aspects of malware (dynamic analysis) in real-world

scenarios, they encompass a diverse array of malware samples and are among the most widely used resources for malware imagery analysis.

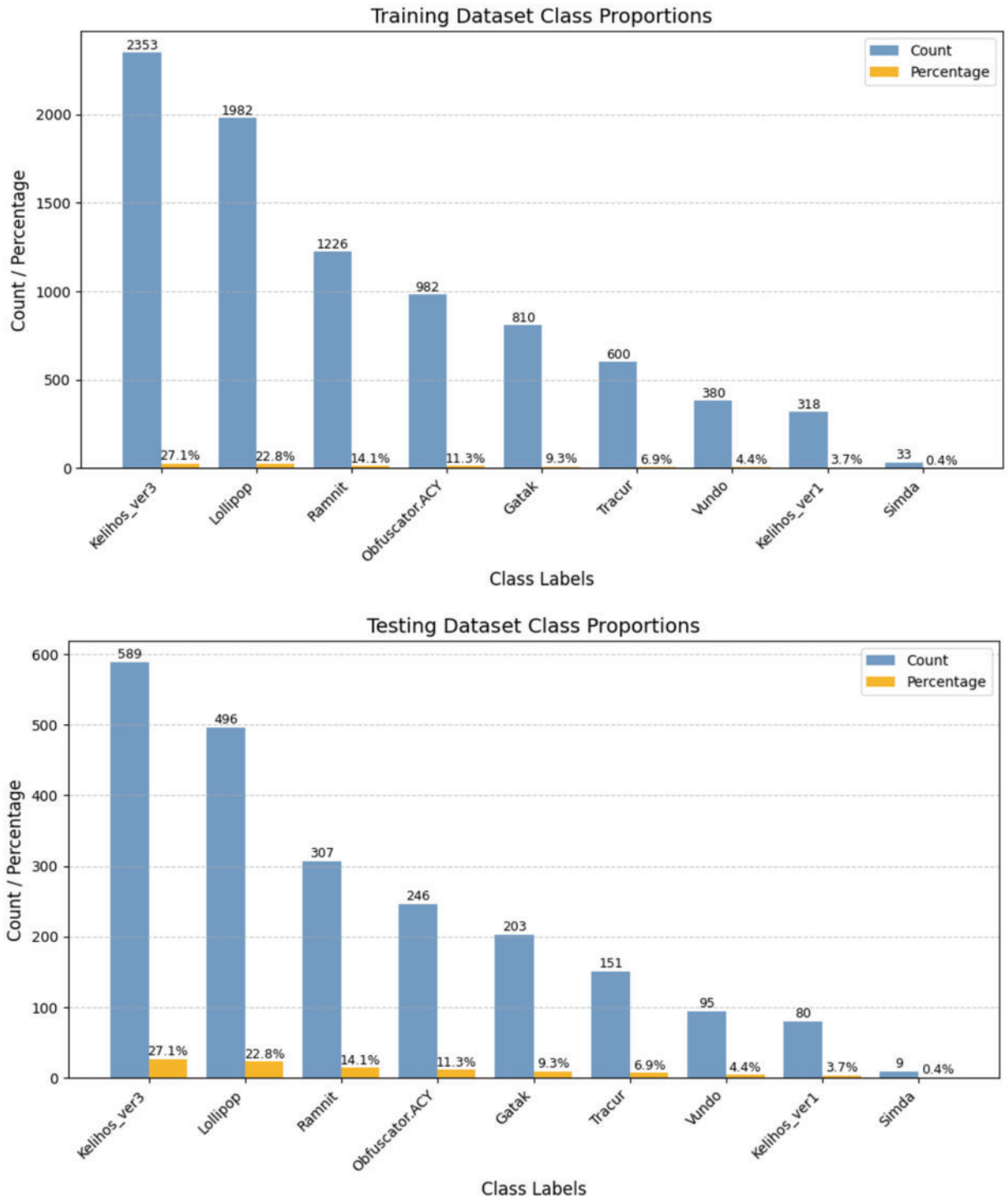

Figure 5: Data distribution in BIG2015

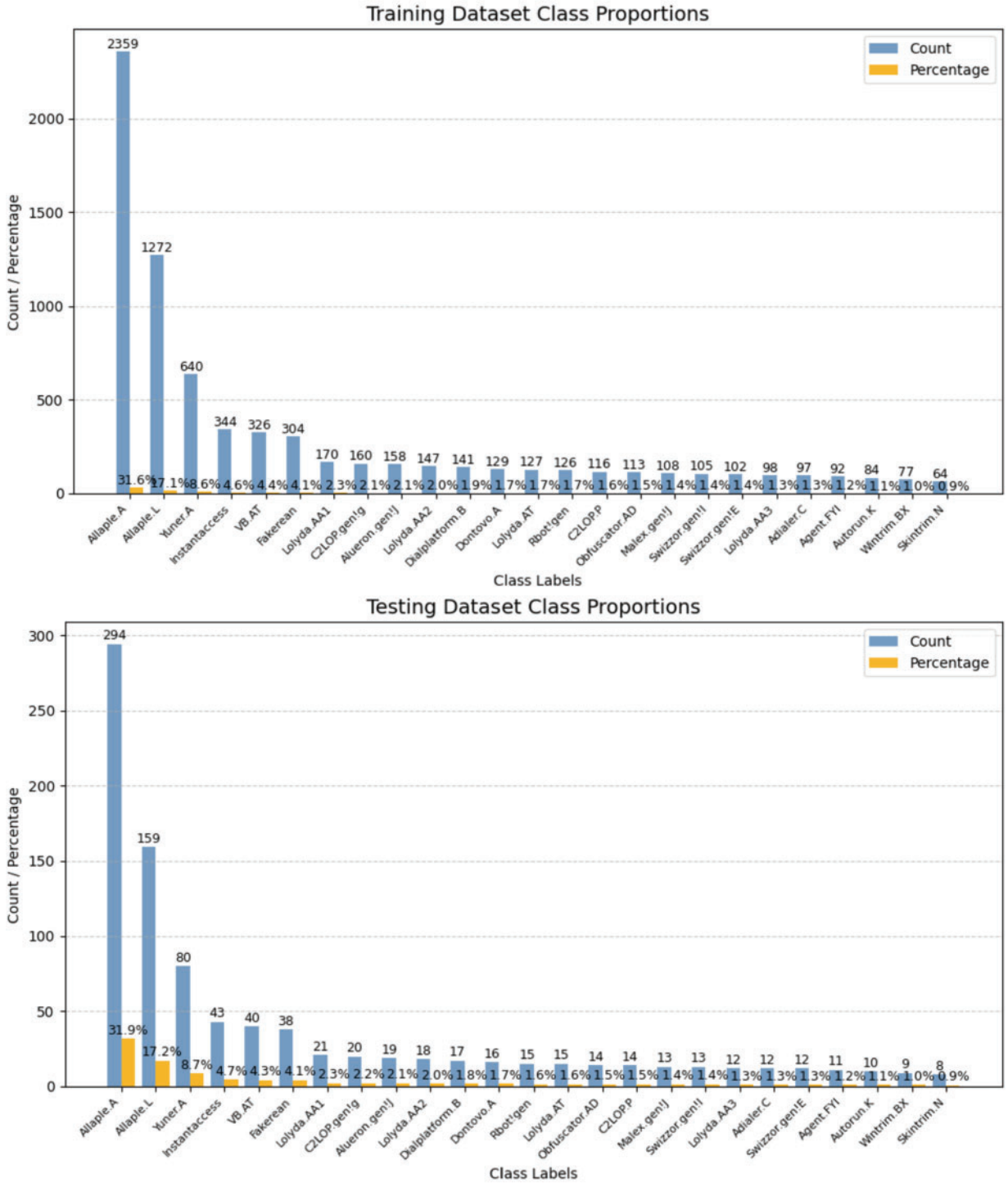

Figure 6: Data distribution in MALIMG

4.2 Experiment Settings

The two datasets are divided into training, validation, and test sets by the authors. In our experiment, we followed three main settings to evaluate the performance of the classification and denoising models. First, we used the original training and test sets to evaluate various classifiers and measure their performance. Second, noise injection was applied to the training set, after which the classifiers were trained on the noisy data and their performance was assessed using the original test set. Lastly, the primary experiment involved employing denoising techniques. In this setting, noise was injected into the training set and a GAN-based denoising model was implemented. The training and validation sets were used to train and validate the denoising model, respectively. The pre-trained generator of the GAN was then used as a denoising model, wherein the training set with noise injection served as the input data, and the test set was used to evaluate classification performance. The test data remained completely unseen by the denoising model throughout the training process while the validation set was fully utilized to validate the model's performance.

In this study, we employ four distinct CNN models as classifiers for an image-based malware detection task. These models were selected because they are widely used in the field and represent diverse types of CNN architectures with distinct design philosophies: DenseNet [34] emphasizes feature reuse through densely connected layers, EfficientNet [35] optimizes performance with compound scaling, MobileNet [36] focuses on lightweight architectures for mobile and resource-constrained devices, and ResNet [37] utilizes skip connections to address vanishing gradient issues. This diversity ensures a comprehensive evaluation of noise resilience across different CNN approaches.

The primary objective of this study is to assess the impact of noise injection on the performance of these models and to evaluate the effectiveness of the proposed denoising technique. By comparing the models' accuracy before noise injection, after noise injection, and following the application of the denoising method, we aim to validate the denoising technique and to understand how noise affects different CNN-based classifiers. To adapt the pre-trained models to our specific classification task, we modify their architectures by replacing the final layers with a customized classifier head. Generally, the attached layers consist of a linear feed-forward layer with 512 units, a ReLU activation function, a dropout layer with a rate of 50%, and a classification output layer. Given that the pre-trained weights are learned from ImageNet, a general multi-class image dataset rather than malware images, we aim to enhance generalization to our task by incorporating a dropout layer. Metrics including accuracy, precision, recall, and F1-score are selected for performance evaluation.

4.3 Results and Discussion

The following analysis evaluates the performance of four CNN-based models across three experimental settings with original images and noise injections, focusing on the models' performances on the test set and the resilience to noise. In this section, all tables presenting the experimental results highlight the highest value of each evaluation metric in bold for clarity and ease of comparison.

Table 1 presents the experiments on the original data and with noise injection for BIG2015. The performance on the original images serves as the baseline for comparison. The effect of noise injections is significant. ResNet outperforms all other models with the highest accuracy (96.5%), precision (96.6%), recall (96.5%), and F1-score (96.5%). This showcases its strong capability in malware classification on clean data. DenseNet and MobileNet closely follow, with comparable test accuracies (95.4%–95.5%) and F1-scores. EfficientNet achieves a slightly lower performance compared to its counterparts at 94.8% accuracy on the test set, which indicates slightly reduced effectiveness on clean data.

Gaussian noise introduces random variations throughout the image, thereby degrading the model's classification accuracy. The results indicate a significant decline in performance across all models when evaluated on the test set. On the training set, except for ResNet, all other models demonstrate reduced accuracy, recall, precision, and F1-score. Notably, ResNet, which performed exceptionally well on clean data, experiences the most substantial drop on the test set, with its accuracy plummeting to 16.54%. This finding suggests that, despite its robust capabilities with clean inputs, ResNet exhibits the least resilience to Gaussian noise. Overall, Gaussian noise leads to a pronounced deterioration in classification performance, with all models experiencing considerable losses in their original accuracy. Both ResNet and DenseNet, despite their strong performances on clean data, display inadequate resilience to this type of noise.

Table 1: Performance metrics with/without noise injection on BIG2015

	Train				**Test**			
Original	**Accuracy**	**Precision**	**Recall**	**F1_score**	**Accuracy**	**Precision**	**Recall**	**F1_score**
DenseNet	0.976	0.976	0.976	0.975	0.954	0.955	0.954	0.954
EfficientNet	0.920	0.920	0.920	0.920	0.949	0.948	0.949	0.948
MobileNet	0.970	0.970	0.970	0.970	0.955	0.954	0.955	0.954
ResNet	**0.985**	**0.985**	**0.985**	**0.985**	**0.965**	**0.966**	**0.965**	**0.965**
Gaussian noise	Accuracy	Precision	Recall	F1_score	Accuracy	Precision	Recall	F1_score
DenseNet	0.953	0.953	0.953	0.953	0.229	0.302	0.229	0.177
EfficientNet	0.838	0.836	0.838	0.836	**0.280**	**0.325**	**0.280**	**0.250**
MobileNet	0.928	0.928	0.928	0.928	0.227	0.246	0.227	0.090
ResNet	**0.975**	**0.975**	**0.975**	**0.975**	0.165	0.182	0.165	0.151
Speckle noise	Accuracy	Precision	Recall	F1_score	Accuracy	Precision	Recall	F1_score
DenseNet	0.970	0.970	0.970	0.970	**0.576**	**0.723**	**0.576**	**0.507**
EfficientNet	0.906	0.905	0.906	0.905	0.432	0.643	0.432	0.384
MobileNet	0.961	0.961	0.961	0.961	0.430	0.395	0.430	0.340
ResNet	**0.982**	**0.982**	**0.982**	**0.982**	0.470	0.414	0.470	0.389

Speckle noise, characterized by multiplicative noise that is associated with image features, exerts a somewhat less detrimental effect than Gaussian noise. DenseNet achieves a test accuracy of 57.63%, exhibiting notable improvements in precision (72.32%) and F1-score (50.71%) compared to its performance under Gaussian noise. This suggests that DenseNet demonstrates moderate resilience to speckle noise in contrast to Gaussian noise. Both EfficientNet and MobileNet experience moderate declines in performance. While ResNet maintains high performance on the training set and achieves an improved test accuracy of 47.01%, it exhibits diminished effectiveness under speckle noise relative to its near-optimal performance on the original clean data. The observed precision, recall, and F1-scores indicate that ResNet encounters challenges in sustaining robust classification in the presence of this type of noise.

Table 2 presents the performance metrics after the application of the denoising model. This analysis centers on assessing the efficacy of denoising techniques as an integral component of the preprocessing pipeline for malware classification under various noisy data conditions. Overall, the incorporation of denoising techniques results in a significant enhancement in classification performance compared to the outcomes derived from raw noisy data, applicable to both Gaussian and Speckle noise.

Notably, the extent of improvement is contingent upon the specific models employed and the type of noise present; the effectiveness is markedly greater for Speckle noise, as evidenced by superior test accuracies and F1-scores when compared to Gaussian noise. Additionally, the degree of enhancement is influenced by the architecture of the classifier. For example, in the context of Gaussian noise, MobileNet exhibits the most substantial improvement among all models in terms of accuracy, with the F1-score increasing significantly to 85.17%. This suggests that MobileNet derives the greatest benefit from denoising, both in terms of class balance and overall performance. Conversely, models such as EfficientNet and DenseNet demonstrate moderate benefits, while ResNet continues to experience considerable challenges with Gaussian noise, resulting in relatively limited improvements from denoising. In contrast, for Speckle noise, all metrics across various models exceed 80%, indicating a consistent enhancement in performance.

Table 2: Performance metrics with denoising model on MALIMG

Gaussian + Denoising	**Train**				**Test**			
	Accuracy	**Precision**	**Recall**	**F1_score**	**Accuracy**	**Precision**	**Recall**	**F1_score**
DenseNet	0.975	0.974	0.975	0.971	0.793	0.778	0.793	0.767
EfficientNet	0.922	0.912	0.922	0.917	0.719	0.733	0.719	0.683
MobileNet	0.976	0.974	0.976	0.974	**0.864**	**0.867**	**0.864**	**0.852**
ResNet	**0.983**	**0.981**	**0.983**	**0.982**	0.623	0.625	0.623	0.567
Speckle + Denoising	Accuracy	Precision	Recall	F1_score	Accuracy	Precision	Recall	F1_score
DenseNet	0.977	0.973	0.977	0.972	**0.888**	0.862	**0.888**	0.860
EfficientNet	0.923	0.913	0.923	0.918	0.883	**0.872**	0.883	**0.877**
MobileNet	0.975	0.973	0.975	0.973	0.811	0.878	0.811	0.843
ResNet	**0.985**	**0.984**	**0.985**	**0.984**	0.823	0.828	0.823	0.825

Table 3 presents the experiments on original data and with noise injection for MALIMG. With 25 labels to predict, the task is more complex than BIG2015. Performance on the original dataset provides the baseline for comparison. DenseNet shows the best test performance with an accuracy of 98.48%, precision of 98.45%, and a balanced F1-score of 98.46%. This demonstrates its effectiveness for malware classification in clean and complex datasets. MobileNet and ResNet perform moderately, and EfficientNet slightly underperforms, but all models have excellent performance on both training and test set.

Table 3: Performance metrics with/without noise injection on MALIMG

	Train				**Test**			
Original	**Accuracy**	**Precision**	**Recall**	**F1_score**	**Accuracy**	**Precision**	**Recall**	**F1_score**
DenseNet	**0.989**	**0.989**	**0.989**	**0.989**	**0.985**	**0.985**	**0.985**	**0.985**
EfficientNet	0.949	0.946	0.949	0.946	0.965	0.954	0.965	0.959
MobileNet	0.982	0.981	0.982	0.981	0.979	0.980	0.979	0.979
ResNet	0.986	0.985	0.986	0.985	0.973	0.965	0.973	0.968
Guassian noise	Accuracy	Precision	Recall	F1_score	Accuracy	Precision	Recall	F1_score
DenseNet	0.976	0.974	0.976	0.974	**0.237**	**0.557**	**0.237**	**0.279**
EfficientNet	0.893	0.887	0.893	0.888	0.004	0.004	0.004	0.004

(Continued)

Table 3 (continued)

	Train				Test			
Original	**Accuracy**	**Precision**	**Recall**	**F1_score**	**Accuracy**	**Precision**	**Recall**	**F1_score**
MobileNet	0.975	0.974	0.975	0.974	0.011	0.001	0.012	0.002
ResNet	**0.985**	**0.984**	**0.985**	**0.984**	0.124	0.341	0.124	0.098
Speckle Noise	Accuracy	Precision	Recall	F1_score	Accuracy	Precision	Recall	F1_score
DenseNet	0.982	0.981	0.982	0.981	**0.845**	**0.867**	**0.845**	**0.826**
EfficientNet	0.942	0.937	0.942	0.938	0.482	0.462	0.482	0.429
MobileNet	0.978	0.977	0.978	0.976	0.675	0.874	0.675	0.712
ResNet	**0.988**	**0.988**	**0.988**	**0.988**	0.602	0.671	0.602	0.613

Gaussian noise induces significant performance degradation across all models, particularly evident in the results from the test set. Among the models evaluated, DenseNet exhibits the highest performance under Gaussian noise injection; however, its accuracy remains low at only 23.73%, highlighting its limited resilience to random pixel variations. While precision is relatively high at 55.74%, the recall drops to 23.73%, resulting in an F1-score of 27.89%. This suggests that DenseNet is capable of making accurate predictions for certain classes but encounters challenges with imbalanced classification across the 25 labels. Conversely, both EfficientNet and MobileNet experience a catastrophic decline in performance on the test set, indicating their extreme sensitivity to Gaussian noise, as they fail to produce meaningful predictions when faced with random noise in the input data. ResNet demonstrates slightly better performance; however, despite its strong baseline performance on clean data, it does not exhibit resilience to Gaussian noise within the increased complexity of the 25-label dataset. Overall, Gaussian noise severely affects all models, with DenseNet displaying marginally better resilience than the others. In contrast, EfficientNet and MobileNet perform poorly, with accuracy and other metrics collapsing entirely. The increased number of labels further exacerbates the difficulty, as Gaussian noise corrupts the features essential for distinguishing between fine-grained classes.

Speckle noise, being more structured than Gaussian noise, exerts a less disruptive effect on model performance. DenseNet exhibits the smallest impact, with test accuracy decreasing from 98.48% (clean) to 84.51%, resulting in an F1-score of 82.56%. Its relatively high precision (86.86%) and recall (84.51%) suggest that the model retains its capacity to classify most of the 25 labels accurately, albeit with slightly diminished confidence. In contrast, EfficientNet demonstrates significant underperformance, with test accuracy falling to 48.21% and a low F1-score of 42.92%. MobileNet and ResNet experience moderate decreases. Overall, speckle noise affects all models less severely than Gaussian noise. DenseNet is distinguished as the most robust under speckle noise, achieving test accuracy comparable to that of clean datasets. MobileNet and ResNet exhibit moderate resilience, while EfficientNet performs relatively poorly.

The effects of Gaussian noise are considerably more disruptive than those of speckle noise for this dataset, which comprises 25 labels. This may be exacerbated by the inherent complexity associated with the dataset's 25-label structure. The random characteristics of Gaussian noise interfere with the fine-grained patterns that are essential for distinguishing among the 25 labels. Models such as EfficientNet and MobileNet, which are optimized for efficiency, exhibit complete failure under Gaussian noise but demonstrate partial recovery when subjected to speckle noise, where the original characteristic of images are preserved. Among all evaluated models, DenseNet exhibits the greatest resilience in the presence of both Gaussian and speckle noise, achieving the highest test accuracy under noisy conditions, particularly when exposed to speckle noise. Although its performance is significantly impaired by Gaussian noise, it still outperforms the other models.

Table 4 presents the performance results following the application of the denoising process. Previously, Gaussian noise resulted in significant performance deterioration, with performance metrics such as accuracy, precision, recall, and F1-score all being severely affected. In contrast, the denoising process markedly enhances model performance, particularly in the presence of Gaussian noise, which induces greater disruptions compared to speckle noise. For DenseNet, the test accuracy improves to 79.31% with denoising, compared to only 23.73% without it, indicating a substantial recovery.

Table 4: Performance metrics with denoising model on MALIMG

Guassian + Denoising	**Train**				**Test**			
	Accuracy	**Precision**	**Recall**	**F1_score**	**Accuracy**	**Precision**	**Recall**	**F1_score**
DenseNet	0.975	0.974	0.975	0.971	0.793	0.778	0.793	0.767
EfficientNet	0.922	0.912	0.922	0.917	0.719	0.733	0.719	0.683
MobileNet	0.976	0.974	0.976	0.974	**0.864**	**0.867**	**0.864**	**0.852**
ResNet	**0.983**	**0.981**	**0.983**	**0.982**	0.623	0.625	0.623	0.567
Speckle + Denoising	Accuracy	Precision	Recall	F1_score	Accuracy	Precision	Recall	F1_score
DenseNet	0.977	0.973	0.977	0.973	**0.888**	0.862	**0.888**	**0.860**
EfficientNet	0.923	0.913	0.9232	0.918	0.7832	0.772	0.783	0.777
MobileNet	0.975	0.973	0.975	0.973	0.811	**0.878**	0.811	0.843
ResNet	**0.985**	**0.984**	**0.985**	**0.984**	0.823	0.828	0.823	0.826

Additionally, precision, recall, and F1-score also exhibit improvements, suggesting balanced enhancements across all metrics. Similarly, EfficientNet demonstrates moderate improvement. MobileNet shows a significant recovery in test accuracy, reaching 86.35% in the presence of denoising, compared to a mere 1.08% without it, making MobileNet the most improved model under Gaussian noise conditions. Precision achieves 86.7%, and the F1-score improves to 85.17%, reflecting strong performance recovery with a high degree of balance. Overall, MobileNet benefits the most among the evaluated models, indicating that its lightweight architecture effectively capitalizes on the denoising process to recover lost features. However, despite the application of denoising, ResNet's performance remains the lowest under Gaussian noise, suggesting that its deeper architecture struggles to generalize effectively in noisy conditions even with denoising applied. Additional strategies, such as adversarial training, may be necessary to enhance its robustness. In conclusion, the denoising technique proves to be highly effective in mitigating the impact of Gaussian noise; however, a performance gap still persists when compared to the clean dataset.

Speckle noise exerts less severe impacts on performance; consequently, the recovery observed with denoising is less dramatic yet remains significant. For DenseNet, the test accuracy increases to 88.84%, representing a modest improvement compared to the performance without denoising (84.51%). Nevertheless, metrics such as precision (86.19%) and F1-score (85.99%) demonstrate balanced enhancements. DenseNet achieves the highest performance among the evaluated models, highlighting its resilience and adaptability when utilized in conjunction with denoising techniques. EfficientNet, MobileNet, and ResNet also exhibit notable improvements across all metrics, underscoring the effectiveness of the denoising model.

From the perspective or no of noise injection, the performance decrement varies for different noises, which also leads to the different degree of improvement of denoising model. Denoising was dramatically effective under Gaussian noise, where all models saw significant performance recoveries. Since Speckle noise was less destructive to begin with, the impact of denoising was less pronounced but still important

for classification accuracy. For model-specific observations, DenseNet performed consistently well in both Gaussian and Speckle noise scenarios, making it the most robust model under noisy conditions with denoising. EfficientNet and MobileNet fall behind but also show exceptional recovery after applying the denoising technique. ResNet, despite being a high-performing model on clean data, still struggles the most with Gaussian noise even after denoising. Its performance was better under Speckle noise, suggesting it benefits from structured noise more than random distortions.

4.4 Implementation and Future Enhancements

While the experiments have demonstrated the effectiveness of the denoising technique, this discussion aims to address the implementation, application, and future enhancements of such a denoising model. The U-Net denoising model, originally developed within a Generative Adversarial Network (GAN) framework, provides a sophisticated approach to image denoising that holds value in cybersecurity and network-based image processing applications. A key advantage of this model is its capacity to function as a standalone denoising preprocessor, utilizing pre-trained weights without necessitating the comprehensive training process associated with the full GAN architecture. Assuming a batch of input data is of size (3, H, W), the general structure of the proposed U-Net is outlined in Table 5.

Table 5: Implementation details of U-Net modules

Structure	Components	Output size of the block
Encoder	"Down" blocks	([128, H/2], [128, H/4], [256, H/8], [512, H/16], [512, H/32])
Decoder	"Up" blocks	([128, H/2], [128, H/4], [256, H/8], [512, H/16], [512, H/32])
Output	Final_convolution	(3, H, W)

There are several potential applications for the pre-trained denoising model in a production environment. The denoising model can be deployed as a standalone micro-service utilizing containerization technologies such as Docker and Kubernetes, allowing for seamless integration into image processing pipelines prior to executing classification tasks. Additionally, the model can be implemented on edge devices equipped with GPU acceleration to facilitate real-time image processing or to enable distributed image denoising across network endpoints. When employed in production environments, the preprocessing stage can be structured as follows: generation of malware image matrices, normalization and standardization of image pixel values, denoising of images, classification, and metric monitoring.

The pre-trained U-Net can be seamlessly integrated into network security applications for the preprocessing of noisy malware visualization datasets, thereby enhancing feature extraction accuracy and improving the robustness of classification models. Continuous maintenance can be achieved by monitoring denoising performance metrics and establishing alerts for performance degradation. Setting up automated retraining pipelines is also a feasible solution if it is without computational constraints.

While the proposed denoising technique has demonstrated its effectiveness, there are several potential challenges associated with real-world applications. Although the pre-trained U-Net weights have been successfully tested on two distinct malware imagery datasets, indicating a degree of generalizability, fine-tuning and optimization will likely be required to adapt the model to specific deployment environments. In addition, incorporating an extra denoising model introduces additional processing time, which may extend the overall malware detection process. Though Gaussian noise captures fundamental characteristics that may overlap with various noise types, and speckle noise represents certain variations, the pre-trained models may still struggle to handle unfamiliar or unseen noise patterns effectively. Furthermore, the model may

not be suitable for devices with resource constraints, such as those with limited memory, computational capacity, or lack of robust hardware acceleration. From a data perspective, malware datasets often suffer from imbalanced distributions among malware types or families, potentially leading to biased training, where the model performs well for dominant classes but poorly for minority classes. This imbalance can further impede the model's ability to generalize effectively.

Building on the proposed denoising model, several additional directions warrant exploration to further enhance its denoising capabilities. Firstly, incorporating more diverse noise types could significantly improve the model's robustness. Salt-and-pepper noise, characterized by the introduction of random white and black pixels into an image or data stream, can represent scenarios in which malware authors insert extraneous data or corrupt specific sections of a binary to obfuscate detection mechanisms. Quantization noise, resulting from the mapping of continuous values to discrete levels, can simulate scenarios where malware undergoes compression or encoding. Poisson noise, which involves fluctuations proportional to the signal intensity, can mimic certain types of dynamic behavior in malware, particularly those that modify their own code in response to environmental conditions. Phase noise, characterized by rapid, short-term variations in the phase that compromise the integrity of the data, can emulate phase manipulation techniques employed by malware to disrupt patterns upon which detection algorithms depend. Moreover, testing with synthetic mixed-noise scenarios that combine multiple noise types would be valuable for simulating more realistic environments and improving its ability to generalize to unseen noise patterns [38]. To further optimize the performance of the U-Net GAN, advanced optimization strategies can be adopted. While dynamic learning rate schedules have been employed in the current work, modifying the loss function by introducing perceptual loss or incorporating adversarial losses has been shown to produce superior denoising results, as demonstrated in prior studies [39]. Finally, diffusion models, renowned for their remarkable performance in image-based generative tasks, progressively refine noisy data over multiple iterations, making them highly suitable for complex denoising scenarios. These models have shown great potential to achieve high-quality restoration even in severe noise conditions [40]. However, given the computational intensity associated with noise generation in each training step, a practical approach would involve reducing the number of timesteps in the diffusion process or leveraging a subset of refinement stages specifically tailored to the characteristics of malware imagery. For practical implementation of the denoising technique, it is advisable to containerize the denoising service to facilitate convenient usage and to configure a scalable cloud or edge infrastructure. Additionally, it is essential to implement continuous and timely performance tracking to ensure optimal operation and to promptly address any potential issues.

5 Conclusion

When formulating malware classification as an image analysis task, it is crucial to consider obfuscation simulation, implement noise injection to replicate this process, and develop effective techniques to mitigate such noise. This paper proposes a U-Net GAN-based denoising model, which serves as a powerful and adaptable solution for image preprocessing. By carefully addressing the deployment architecture and ensuring generalizability across different noise types, this technology can substantially enhance image processing pipelines across various domains. Furthermore, we investigate the noise resistance capabilities of different CNN-based classifiers; key findings indicate that DenseNet exhibits strong resistance to noise, while ResNet, although outperforming other models on clean data, demonstrates the least resilience to Gaussian noise. Overall, this research conducts a comparative study on the performance of various CNN-based classifiers under different noise injection conditions and proposes denoising techniques applicable to security domains. This work contributes to enhancing cybersecurity by developing an efficient system for recovering essential characteristics of malware imagery, thereby improving the model's predictive performance.

Acknowledgement: We sincerely thank the reviewers and the editor for their valuable time, effort, and constructive feedback, which greatly contributed to improving the quality of this work.

Funding Statement: This research is partially funded by the budget project FFZF-2022-0007.

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Huiyao Dong, Igor Kotenko; data collection: Huiyao Dong; analysis and interpretation of results: Huiyao Dong; manuscript preparation: Huiyao Dong, Igor Kotenko. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Data openly available in public repositories. The data that support the findings of this study are openly available at *Kaggle: Malware Classification* https://www.kaggle.com/c/malware-classification/data (accessed on 25 January 2025) and *Google Drive File* https://drive.google.com/file/d/1M83VzyIQj_kuE9XzhClGK5TZWh1T_pr-/view (accessed on 25 January 2025).The Python code developed for this research can be found at the following repository: https://github.com/hydongmeow/denoising_malware (accessed on 25 January 2025).

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. SonicWall. 2024 Mid-year cyber threat report; 2024. [cited 2024 Nov 16]. Available from: https://www.sonicwall.com/2024-Mid-Year-Cyber-Threat-Report.asp.
2. Luo JS, Lo DCT. Binary malware image classification using machine learning with local binary pattern. In: 2017 IEEE International Conference on Big Data (Big Data) ; 2017; Boston, MA, USA. p. 4664–7.
3. Azab A, Khasawneh M. MSIC: malware spectrogram image classification. IEEE Access. 2020;8:102007–21. doi:10.1109/ACCESS.2020.2999320.
4. Dong H, Kotenko I. Image-based malware analysis for enhanced IoT security in smart cities. Int Things. 2024;27(10):101258. doi:10.1016/j.iot.2024.101258.
5. Bensaoud A, Kalita J, Bensaoud M. A survey of malware detection using deep learning. Mach Learn Appl. 2024;16:100546. doi:10.1016/j.mlwa.2024.100546.
6. Kumar S, Janet B. DTMIC: deep transfer learning for malware image classification. J Inf Secur Appl. 2022;64:103063. doi:10.1016/j.jisa.2021.103063.
7. Pham DP, Marion D, Mastio M, Heuser A. Obfuscation revealed: leveraging electromagnetic signals for obfuscated malware classification. In: Proceedings of the 37th Annual Computer Security Applications Conference; 2021; New York, NY, USA. p. 706–19.
8. Dong H, Kotenko I. VAE-GAN for Robust IoT malware detection and classification in intelligent urban environments: an image analysis approach. In: International Conference on Risks and Security of Internet and Systems; 2023; Cham: Springer. p. 200–15.
9. Bensaoud A, Kalita J. Deep multi-task learning for malware image classification. J Inf Secur Appl. 2022;64:103057. doi:10.1016/j.jisa.2021.103057.
10. Lee H, Kim S, Baek D, Kim D, Hwang D. Robust IoT malware detection and classification using opcode category features on machine learning. IEEE Access. 2023;11:18855–67. doi:10.1109/ACCESS.2023.3247344.
11. Bhat P, Behal S, Dutta K. A system call-based android malware detection approach with homogeneous & heterogeneous ensemble machine learning. Comput Secur. 2023;130:103277. doi:10.1016/j.cose.2023.103277.
12. Chaganti R, Ravi V, Pham TD. Image-based malware representation approach with EfficientNet convolutional neural networks for effective malware classification. J Inf Secur Appl. 2022;69:103306. doi:10.1016/j.jisa.2022.103306.
13. Dao TV, Sato H, Kubo M. An attention mechanism for combination of CNN and VAE for image-based malware classification. IEEE Access. 2022;10:85127–36. doi:10.1109/ACCESS.2022.3198072.

14. Thakur P, Kansal V, Rishiwal V. Hybrid deep learning approach based on LSTM and CNN for malware detection. Wirel Pers Commun. 2024;136(3):1879–901. doi:10.1007/s11277-024-11366-y.
15. Karat G, Kannimoola JM, Nair N, Vazhayil A, Sujadevi V, Poornachandran P. CNN-LSTM hybrid model for enhanced malware analysis and detection. Procedia Comput Sci. 2024;233:492–503. doi:10.1016/j.procs.2024.03.239.
16. Dong H. Convolutional-free malware image classification using self-attention mechanisms. Inform Autom. 2024;23(6):1869–98. doi:10.15622/ia.23.6.11.
17. Tian C, Fei L, Zheng W, Xu Y, Zuo W, Lin CW. Deep learning on image denoising: an overview. Neural Netw. 2020;131:251–75. doi:10.1016/j.neunet.2020.07.025.
18. Su Y, Lian Q, Zhang X, Shi B, Fan X. Multi-scale cross-path concatenation residual network for Poisson denoising. IET Image Process. 2019;13(8):1295–303. doi:10.1049/iet-ipr.2018.5941.
19. Jifara W, Jiang F, Rho S, Cheng M, Liu S. Medical image denoising using convolutional neural network: a residual learning approach. J Supercomput. 2019;75:704–18. doi:10.1007/s11227-017-2080-0.
20. Yuan Q, Zhang Q, Li J, Shen H, Zhang L. Hyperspectral image denoising employing a spatial-spectral deep residual convolutional neural network. IEEE Trans Geoscience Remote Sens. 2019;57(2):1205–18. doi:10.1109/TGRS.2018.2865197.
21. Li S, Wen H, Deng L, Zhou Y, Zhang W, Li Z, et al. Denoising network of dynamic features for enhanced malware classification. In: 2023 IEEE International Performance, Computing, and Communications Conference (IPCCC); 2023; Los Alamitos, CA, USA. p. 32–9.
22. Kumi S, LeeSuk -Ho. BM3D and deep image prior based denoising for the defense against adversarial attacks on malware detection networks. Int J Adv Smart Converg. 2021;10(3):163–71. doi:10.7236/IJASC.2021.10.3.163.
23. Taheri R, Shojafar M, Arabikhan F, Gegov A. Unveiling vulnerabilities in deep learning-based malware detection: differential privacy driven adversarial attacks. Comput Secur. 2024;146:104035. doi:10.1016/j.cose.2024.104035.
24. Popescu AB, Taca IA, Vizitiu A, Nita CI, Suciu C, Itu LM, et al. Obfuscation algorithm for privacy-preserving deep learning-based medical image analysis. Appl Sci. 2022;12(8):3997. doi:10.3390/app12083997.
25. Boncelet C. Image noise models. In: The essential guide to image processing. USA: Elsevier; 2009. p. 143–67.
26. Maity A, Pattanaik A, Sagnika S, Pani S. A comparative study on approaches to speckle noise reduction in images. In: 2015 International Conference on Computational Intelligence and Networks; 2015; Odisha, India. p. 148–55.
27. Racine R, Walker GA, Nadeau D, Doyon R, Marois C. Speckle noise and the detection of faint companions. Publ Astron Soc Pac. 1999;111(759):587. doi:10.1086/316367.
28. Fan CM, Liu TJ, Liu KH. SUNet: swin transformer UNet for image denoising. In: 2022 IEEE International Symposium on Circuits and Systems (ISCAS); 2022; Austin, TX, USA: IEEE. p. 2333–7.
29. Lu Z, Chen Y. Single image super-resolution based on a modified U-net with mixed gradient loss. Signal, Image Video Process. 2022;16(5):1143–51. doi:10.1007/s11760-021-02063-5.
30. Ahmad Z, Jaffri ZuA, Chen M, Bao S. Understanding GANs: fundamentals, variants, training challenges, applications, and open problems. Multimed Tools Appl. 2024. doi:10.1007/s11042-024-19361-y.
31. Loshchilov I, Hutter F. Decoupled weight decay regularization. arXiv:1711.05101. 2019.
32. Ronen R. Microsoft malware classification challenge. arXiv:1802.10135. 2018.
33. Nataraj L, Karthikeyan S, Jacob G, Manjunath BS. Malware images: visualization and automatic classification. In: Proceedings of the 8th International Symposium on Visualization for Cyber Security; 2011; New York, NY, USA. p. 1–7.
34. Huang G, Liu Z, Van Der Maaten L, Weinberger KQ. Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Comput Vis Pattern Recognit; 2017; Honolulu, HI, USA. p. 4700–8.
35. Tan M, Le Q. Efficientnet: rethinking model scaling for convolutional neural networks. In: International Conference on Machine Learning; 2019; Long Beach, CA, USA: PMLR. p. 6105–14.
36. Sandler M, Howard A, Zhu M, Zhmoginov A, Chen LC. Mobilenetv2: inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2018; Salt Lake City, UT, USA. p. 4510–20.

37. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2016; Las Vegas, NV, USA. p. 770–8.
38. Zhang K, Zuo W, Chen Y, Meng D, Zhang L. Beyond a gaussian denoiser: residual learning of deep CNN for image denoising. IEEE Trans Image Process. 2017;26(7):3142–55. doi:10.1109/TIP.2017.2662206.
39. Ledig C, Theis L, Huszár F, Caballero J, Cunningham A, Acosta A, et al. Photo-realistic single image super-resolution using a generative adversarial network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2017; Honolulu, HI, USA. p. 4681–90.
40. Saharia C, Ho J, Chan W, Salimans T, Fleet DJ, Norouzi M. Image super-resolution via iterative refinement. IEEE Trans Pattern Analysis Mach Intell. 2022;45(4):4713–26. doi:10.1109/TPAMI.2022.3204461.