Journal of Intelligent Information Systems
https://doi.org/10.1007/s10844-025-00936-6

RESEARCH

A deep learning-based approach for stegomalware sanitization in digital images

Angelica Liguori[1] · Marco Zuppelli[2] · Daniela Gallo[1,3] · Massimo Guarascio[1] · Luca Caviglione[2]

Received: 8 November 2024 / Revised: 5 March 2025 / Accepted: 31 March 2025

Abstract
Malware is increasingly endowed with steganographic mechanisms for concealing malicious data to avoid detection or bypass security measures. As a result, an emerging wave of threats named *stegomalware* has started to rise. Among the various approaches, real-world stegomalware primarily hides information within digital images, for instance, to retrieve additional payloads or configuration data. Unfortunately, developing attack-agnostic mitigation tools is difficult, especially due to the tight relation between the image format and the steganographic technique. Therefore, this paper presents an autoencoder-based approach to perform *sanitization*, i.e., to disrupt the malicious content hidden in images without altering their visual quality. For this purpose, we used an enhanced U-Net-like neural architecture, and we compared our idea against other mechanisms, including JPG transcoding and simple addition of Gaussian noise. Results obtained by considering different hiding patterns and realistic payloads showcased the effectiveness of our approach. Moreover, the U-Net-based sanitization solution prevents the recovery of the payload while preserving the original image quality and reducing risks arising from side-channel attacks.

Keywords Deep learning · Steganography · Sanitization · Side-channel attacks

✉ Angelica Liguori
angelica.liguori@icar.cnr.it

Marco Zuppelli
marco.zuppelli@ge.imati.cnr.it

Daniela Gallo
daniela.gallo@icar.cnr.it

Massimo Guarascio
massimo.guarascio@icar.cnr.it

Luca Caviglione
luca.caviglione@ge.imati.cnr.it

[1] Institute for High Performance Computing and Networking, National Research Council, Via P. Bucci, 8/9 C, Rende 87036, Italy

[2] Institute for Applied Mathematics and Information Technologies, National Research Council, Via de Marini 6, Genova 16149, Italy

[3] University of Salento, Piazza Tancredi, 7, Lecce 73100, Italy

Published online: 07 April 2025

1 Introduction

The increasing effectiveness of security countermeasures forces malicious actors to rethink their attack chain. For instance, modern threats can void signature-based detection frameworks through some form of code obfuscation. In this case, the attacker injects junk code or diverts the normal execution flow to prevent the creation of well-defined patterns of instructions (Chua & Balachandran, 2018). Another offensive approach takes advantage of multi-stage loading architectures, allowing the retrieval of attack routines only when needed. The malicious software has then a reduced footprint, which is more difficult to detect (Zimba et al., 2018). Since malware often operates via the Internet, e.g., to interact with a remote command & control facility, an important aspect concerns the ability to bypass firewalls or network intrusion detection mechanisms. To this aim, an effective approach leverages techniques to hide data within digital objects for creating covert channels (Caviglione et al., 2021).

Even if the literature abounds in techniques for concealing information within network protocols, binary code, multimedia objects, and IoT data (Singh, 2020; Caviglione et al., 2023), attacks observed "in the wild" primarily exploit digital pictures (Mazurczyk & Caviglione, 2015; Caviglione & Mazurczyk, 2022). As an example, a recent work isolated 36 different malware targeting image formats, i.e., PNG, JPG, and BMP (Strachanski et al., 2024). Hence, being able to face the proliferation of malicious software such as PNGLoader, RegDuke, urlzone, and OceanLotus should be considered a compelling need to not endanger the security posture of the Internet (Strachanski et al., 2024). To emphasize the importance of the use of a steganographic technique to conceal offensive data within an innocent-looking digital asset, such a new wave of threats has been named *stegomalware* (Caviglione & Mazurczyk, 2022). Their attack model aims at hiding within a digital object (defined as the *carrier*) a malicious slice of data (defined as the *payload*). Examples of payloads are a list of URLs to contact, ASCII strings with commands for a botnet, or configuration data. To avoid the introduction of signatures, lags or transmission delays, the cloaking process should be kept as simple as possible. For this reason, modern malware mainly exploits Least Significant Bit (LSB) steganography, which hides content by overwriting the last bits of the color components of pixels composing the targeted image.

Despite the diffusion of malicious software deploying steganography is now a real concern (Caviglione & Mazurczyk, 2022; Strachanski et al., 2024), only a small fraction of works deals with the *sanitization* of digital images, i.e., the process of disrupting the hidden content while preserving the carrier. Instead, a large amount of research focuses on low-level countermeasures that can prevent a threat actor from incorporating hidden data within a pre-existing application, e.g., through repackaging (Sihag et al., 2021). Alas, the tight interdependence between steganographic techniques and targeted carriers may impede the extension of domain-specific countermeasures (e.g., suitable for mobile ecosystems) to Internet-scale scenarios (Suarez-Tangil et al., 2014).

In this perspective, a promising approach exploits Artificial Intelligence (AI) and its ability to "generalize" the sanitization process for a family of steganographic primitives, e.g., plain LSB (Robinette et al., 2023b). For instance, Robinette et al. (2023a) demonstrated how diffusion models can remove malicious contents hidden in an image while avoiding its degradation. Unfortunately, the considered threat model deals with an attacker hiding an image within another, which is seldom observed in realistic offensive campaigns (Mazurczyk & Caviglione, 2015; Caviglione & Mazurczyk, 2022). A more realistic template has been considered in Zuppelli et al. (2021), where AI is used to remove data cloaked via

Invoke-PSImage, i.e., a variant of LSB exploiting multiple color channels. Yet, malware exploiting Invoke-PSImage leaves a major statistical signature in the red channel, revealing its presence without the need of complex frameworks (Schaffhauser et al., 2022). Instead, Han et al. (2020) showcases how machine learning can detect malicious PowerShell scripts starting from signatures in the color histogram of altered pixels. However, simpler AI-based countermeasures against stegomalware could struggle to face some threats due to "concept drifts" inducing degradation of the models and requiring further tweaks (Gibert et al., 2020).

Therefore, our work showcases the use of Autoencoders (AEs) (see, Bank et al. (2023) and the references therein) to sanitize a payload cloaked in a digital image via LSB steganography. The underlying idea is that these architectures can disrupt the information embedded in the images while preserving the quality. Their adoption has been driven by their properties and performances, especially for security-oriented applications. As an example, AEs proved to be effective in identifying malicious patterns of system calls (D'Angelo et al., 2020).

This paper largely extends the work presented in Liguori et al. (2024). In more detail: *i*) it compares the outcome of the sanitization obtained with AEs and U-Net/U-Net+ approaches with a naive technique based on Gaussian noise; *ii*) it evaluates the sanitized images not only according to generic metrics but also by considering the Structural Similarity Index Measure (SSIM); *iii*) the performance evaluation has been refined by also considering larger digital media, i.e., PNG files of $1,024 \times 1,024$ pixels; *iv*) the used dataset now contains $21,000$ pictures instead of $3,500$ images; *v*) it assesses the use of transcoding, which is often deployed to disarm malicious contents (Dubin, 2023); *vi*) it elaborates on security implications of the sanitization process, especially in terms of side-channel attacks.

The rest of the paper is structured as follows. Section 2 reviews past research works on the mitigation of malware targeting digital media. Section 3 outlines the considered attack model, whereas Section 4 describes the various neural architectures used to disrupt malicious payloads hidden in images. Section 5 showcases numerical results, and Section 6 discusses security implications of mitigation mechanisms. Lastly, Section 7 concludes the paper and hints at future research directions.

2 Related work

A major challenge for the sanitization of digital images cloaking malicious assets is to devise approaches that are agnostic to the specific attack technique. For instance, several malicious actors hide offensive routines or configuration data by taking advantage of plain LSB steganography (e.g., 1 bit of data is hidden in each color channel), whereas others tried to smuggle payloads via the Invoke-PSImage mechanism (Rus et al., 2023). Traditional sanitization approaches focused on disrupting hidden information in images by leveraging methods based on signal manipulation such as additive noise, lossy compression, and image filtering (Ameen & Al-Badrany, 2013; Tao et al., 2014; Geetha et al., 2021). Owing to the use of deterministic algorithms, they are quite efficient but may also significantly degrade the image quality during the sanitization process, leading to the potential loss of important visual details and impacting the overall usability of the images.

With the advancements of deep learning, more sophisticated solutions have been recently devised to mitigate this issue. As possible examples, in Zuppelli et al. (2021); Robinette et al. (2023b), the authors propose an approach based on unsupervised neural architectures for disrupting payloads cloaked by malware. In more detail, in Zuppelli et al. (2021), the authors propose a U-Net-like encoder-decoder-based model, which is used for sanitizing images that

contain malicious data injected via the Invoke-PSImage method. Conversely, in Robinette et al. (2023b), the authors propose a variational AE model whose aim is to remove a "secret" image that is hidden in another image, offering a blind solution that does not depend on prior knowledge of steganographic techniques. To improve the reconstruction capabilities of the approach when the complexity of the image increases, in Robinette et al. (2023a), the authors propose an enhancement based on diffusion models. Diffusion models are also used in Tan et al. (2024) but to optimize the image quality after the hidden information has been destroyed, i.e., to mitigate degradations via post-processing techniques. For the "sterilization" process, the authors propose an overwriting strategy designed to disrupt all pixel information, followed by restoring it through adjacent visual pixels, as suggested in Bastani et al. (2010). Then, the diffusion optimization strategy is used to minimize the image distortion caused by the overwriting procedure, thus enhancing its visual quality. A similar approach is proposed in Zhu et al. (2023), where the authors design two neural networks, which account for scaling resilience, effectively sanitizing images without degrading quality. The convolutional layers of the first network resize the oversized image while simultaneously destroying the hidden content. The second network, designed for already resized images, exploits convolutional and linear interpolation layers to effectively erase hidden information. Additionally, a U-Net-based optimization strategy is used to enhance the visual quality of the sanitized image.

In production-quality environments images are often sanitized by exploiting weaknesses of the attack or through simple heuristics. As an example, in Blasco et al. (2012), the authors introduce a proxy able to dissect HTTP conversations and route the various in-line objects composing the hypertext towards ad-hoc sanitization units. For the case of images, hidden malicious content is disrupted by altering the two least significant bits of the RGB components of random selected pixels. A more sophisticated approach could exploit non-linear transfer functions, especially to reduce the possibility of leaving some regions unaltered (Jung et al., 2020). If the malware internals are known (e.g., via reverse engineering), the sanitization process could reduce to basic file operations. For instance, for the case of PNG images altered by Gatak/Stegoloader, contents could be removed by simply searching for inconsistencies in the number of blocks composing the image and its overall size (Puchalski et al., 2020). This is possible since the malware append the data to the file rather than using some form of steganography. Yet, this class of threats are definitely outside the scope of this work.

Table 1 summarizes the main features of the approaches available in the literature at the best of our knowledge. Specifically, unlike previous approaches (Robinette et al., 2023a, b), our work considers two realistic hiding templates observed in many attacks, i.e., plain LSB and OceanLotus. Instead, compared to Tan et al. (2024); Zhu et al. (2023), our mechanism aims at introducing an "all-in-one" framework that effectively disrupts hidden information while maintaining high visual image quality, without requiring any optimization strategy. An important advancement over our preliminary work (Zuppelli et al., 2021) concerns the use of a simplified yet powerful sanitization method, using an enhanced U-Net-like encoder-decoder architecture with additional connections that improve the learning stability. Compared to other solutions such as Blasco et al. (2012), our idea allows to face an attacker deploying an advanced hiding scheme, e.g., by carefully selecting areas where to hide data to maximize its undetectability and robustness. Our work also provides a prime attempt to deeply explore the security implications of the sanitization process, with particular attention to potential side channels that attackers could exploit to compromise its effectiveness.

When sanitizing an image is not feasible, especially due to scalability constraints, an alternative approach is to pursue detection. However, the literature primarily focuses on mechanisms to spot malicious code or textual entries and largely neglects a threat actor hiding data in multimedia assets through some form of steganography (Gaber et al., 2024).

Table 1 Summary of image sanitization approaches

Reference	Attack Type(s)	Sanitization Technique	Reconstruction	AI-Based
Blasco et al. (2012)	LSB	Random	–	x
Ameen and Al-Badrany (2013)	LSB and DCT	Filtering and Wavelet Techniques	–	x
Jung et al. (2020)	LSB	Non-Linear Transformations	–	x
Geetha et al. (2021)	Information Hiding	Multi-Directional Filtering	–	x
Zuppelli et al. (2021)	Invoke-PSImage	U-Net	AE-Based	✓
Robinette et al. (2023b)	LSB	Variational AE	AE-Based	✓
Robinette et al. (2023a)	LSB	Diffusion Model	Diffusion optimization strategy	✓
Zhu et al. (2023)	Information Hiding	CNN-Based	U-Net-Based	✓
Tan et al. (2024)	Information Hiding	Overwriting	Restore through adjacent pixels	✓
Our	LSB and OceanLotus	U-Net+	AE-Based	✓

Concerning the use of AI-capable frameworks, Chin and Corizzo (2024) deals with the detection of various malicious software targeting Android via common features such as memory dumps and logs. A similar scenario is considered in Zhao et al. (2025), which exploits large language models to identify textual entries, such as URLs or IP ranges, that can reveal the presence of a tampered device. Indeed, developing novel detection strategies in modern settings requires suitable datasets, yet available malware collections mainly consider samples not endowed with information hiding capabilities (Nguyen et al., 2025). Alas, the tight relation between the used steganographic mechanism and the targeted multimedia content makes the development of "general" detection frameworks difficult (Cassavia et al., 2023).

3 Problem statement

As hinted, we want to develop an approach that sanitizes images hiding malicious data without altering the perceived quality. This is the typical attack model implemented by a stegomalware trying to: *i*) drop a payload on the host of the victim, e.g., through a mail attachment; *ii*) implement a multi-stage loading architecture, e.g., to extend its functionality only when needed via a slice of hidden code; *iii*) retrieve information from a remote host, e.g., contact a botmaster to update its configuration. Despite the nature of the hidden payload, the general attack model and reference scenario are those depicted in Fig. 1.

In more detail, we assume a threat actor or malware (labeled as "attacker", in the figure) able to hide some arbitrary information denoted with ϵ in a legitimate image denoted with x. To model a general yet realistic use case, we assume that ϵ is a list of URLs pointing at financial institutions, as observed in many attack campaigns taking advantage of the ZeusVM/Zbot trojan (Mohaisen & Alrawi, 2013). As regards the steganographic mechanisms used to conceal the information, we consider two strategies observed in many malware samples and in the OceanLotus advanced persistent threat (Mazurczyk & Caviglione, 2015; Caviglione & Mazurczyk, 2022). Both strategies are based on the LSB steganography, which is still the

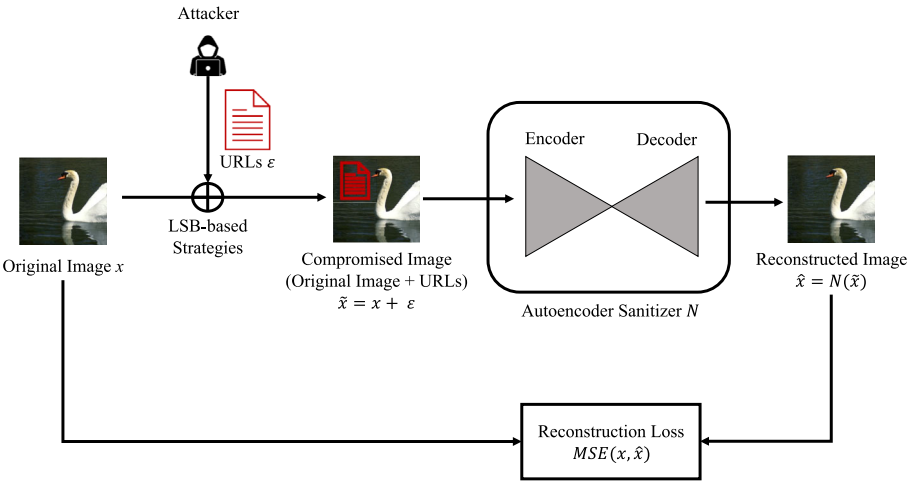

Fig. 1 Reference scenario of the considered sanitization problem. The AE-based model processes the image modified by the attacker to disrupt the hidden payload

simpler yet most deployed technique to hide data in real-world attack campaigns (Strachanski et al., 2024). Recalling that the steganographic algorithms should not bloat the malware or inflate its resource footprint, for each pixel composing the image, data may be cloaked via:

- LSB plain: the payload is hidden in the least significant bit of the red, green, and blue color channels;
- LSB variant: the payload is concealed in the three least significant bits of red and green color channels and in the two least significant bits of the blue channel.

As a result, the original image x is modified, and the new object embedding the malicious content is denoted with $\tilde{x} = x + \epsilon$.

Referring again to Fig. 1, our goal is to find an estimate $\hat{x}$ that is as close as possible to the original, unmodified image x, such that $\hat{x} = N(\tilde{x})$. To this extent, the function N maps the "compromised" image $\tilde{x}$ to its corresponding estimate $\hat{x}$. Clearly, the new image $\hat{x}$ must not contain anymore the payload cloaked by the attacker, even partially. Therefore, N has been designed to minimize the dissimilarity between the estimated image $\hat{x}$ and the "clean" image x, measured using a suitable reconstruction loss function $\mathcal{L}$. In our reference scenario, N acts as the *sanitizer* and takes the form of an unsupervised neural network model (i.e., an AE-based model) performing two main operations. As a first step, it compresses the matrix representation of the image acting as the input data within a latent space. As a second step, it reconstructs the original information provided as input. In both steps, the pixel is the smallest manageable element for the AE and it is represented by its RGB components.

4 Sanitization through deep learning

In our approach, the sanitizer takes the form of an AE conceptually composed of two subnets, i.e., the *encoder* and the *decoder*. In general, the former allows for learning a mapping $z = enc(x)$ between the input layer and the hidden layers (*encoding*), whereas the second learns a mapping $y = dec(z)$ between the features extracted by the encoder and the output layer (*decoding*). Both mappings are defined in terms of nonlinear activation functions governed by a set W of weights. In particular, the proposed sanitizer N, depicted in Fig. 2, is based on the U-Net architecture of Ronneberger et al. (2015).

Unlike a classical AE, our sanitizer includes skip connections that facilitate multi-scale feature fusion. In this way, the network is able to combine low- and high-level features from different stages of the encoder, capturing both local details and global context that lead to

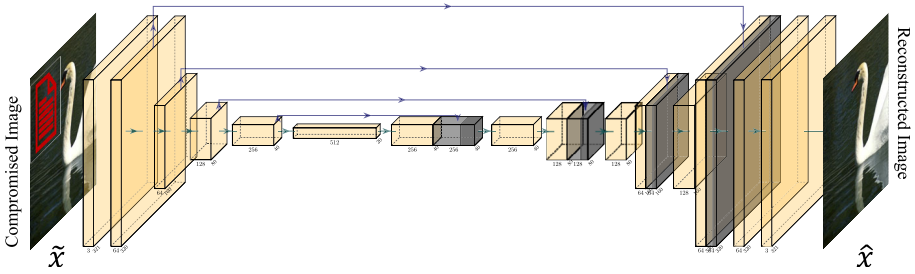

Fig. 2 U-Net-like architecture: grey blocks represent the corresponding blocks in the encoding phase, stacked with the outputs from deconvolutional blocks. The malicious input image is sanitized through deep convolutional layers

more accurate performances. Specifically, the input image is progressively reduced in size and doubled in volume through convolutional blocks until a compact core representation is obtained. In the decoding phase, the same number of deconvolutional blocks are used, each combined with the corresponding residual block from the encoding phase and further transformed in volume by an additional convolutional block. The final image is reconstructed by using a sigmoid activation layer. Each block consists of a convolution/transposed convolution layer, a rectified linear unit, a dropout, and a batch normalization layer.

The neural model is learned on a set $\mathcal{D} = \{(\tilde{x}_1, x_1), (\tilde{x}_2, x_2), \ldots, (\tilde{x}_M, x_M)\}$ of image pairs, where $M = |\mathcal{D}|$ is the number of samples, x_i represents the original "clean" image, and $\tilde{x}_i = x_i + \epsilon_i$ the "compromised" input. During the learning phase, the AE is fed with the "compromised" images, while the legitimate ones are used for computing the loss values. Specifically, our AI-based sanitizer is trained through a supervised learning method having the compromised version of the image as the input. To generate a sanitized version, the output is compared with the legitimate image, which guides the network in removing the malicious information. This approach resembles the underlying idea of a Denoising Autoencoder (Vincent et al., 2008), that instead of taking the original data as input, it is fed with its corrupted/noisy version and is trained to recover the original "undistorted" data. Thus, the learning phase tries to optimize the network weights by minimizing the reconstruction loss. To this aim, we used the Mean Squared Error (MSE) since it measures divergences at a pixel level:

$$MSE(N, \mathcal{D}) = \frac{1}{M} \sum_{i=1}^{M} \|x_i - N(\tilde{x}_i)\|^2$$

The standard U-Net neural architecture has been extended with further hidden layers to yield latent representations with different sizes (Zhou et al., 2019). This makes the learning process more stable and improves the effectiveness of the reconstruction. Figure 3 sketches the connection scheme of our architecture that .represents a variant of the U-Net+ proposed

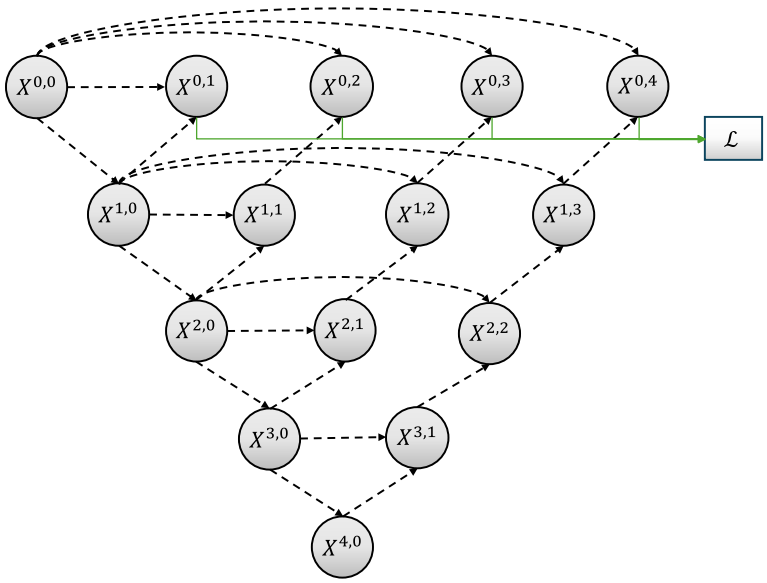

Fig. 3 Connection scheme for the U-Net+ variant borrowed from Liguori et al. (2024)

in Zhou et al. (2019). For the sake of simplicity, we will denote our variant as U-Net+. Here, the input image is simultaneously reconstructed by L output layers. Specifically, we consider $L = 4$ output layers denoted as $X^{0,1}$, $X^{0,2}$, $X^{0,3}$ and $X^{0,4}$, respectively. This permits us to take advantage of the intermediate representations generated by the other layers. In particular, let $\hat{X}_i^{0,j}$ be the output of the j-th layer for the i-th instance. The loss function is then defined as the sum of the errors of the L output layers:

$$\mathcal{L} = \sum_{j=1}^{L} \frac{1}{M} \sum_{i=1}^{M} \left\| x_i - \hat{X}_i^{0,j} \right\|^2 \tag{1}$$

Notably, our approach currently handles images with a fixed size. However, the proposed framework could be extended to process images with larger or smaller sizes by using patch-based or padding approaches, respectively. In particular, larger images can be divided into patches, each with a size matching the fixed input size. These patches would then be processed independently, with the network trained to minimize the error for each patch.

5 Experimental investigation

In this section, we introduce the setup used to quantify the performance of our sanitization approach. Then, we present numerical and qualitative results.

5.1 Experimental setup

As discussed, in this work we consider a ZeusVM-like offensive template. Specifically, ZeusVM hides malicious URLs of financial institutions to be attacked into innocent-looking pictures (Mohaisen & Alrawi, 2013). To model such a behavior, we used URLs borrowed from a public list[1] of real financial institutions. As observed in malware samples collected "in the wild", we separated the hidden URLs by using the `*d*` escape sequence. This allows to model an attacker trying to retrieve a single URL (even if the image has been sanitized) by searching for the given escape sequence.

To hide/recover the resulting list of URLs, we developed Python scripts implementing the two LSB techniques presented in Section 3. In general, ZeusVM embeds content in the target image through a naive sequential strategy (Mohaisen & Alrawi, 2013; Strachanski et al., 2024). To model advanced threats or elusive behaviors, we performed investigations by also considering the following hiding patterns:

- *sequential*: the payload is hidden starting from the first pixel (i.e., the one located in the upper left corner);
- *rows*: the payload is first divided into three equal parts and then hidden in parallel, interleaved areas;
- *squares*: the payload is hidden in equally-sized blocks placed along the diagonal.

To evaluate the effectiveness of our sanitization approach and to capture future advancements in the "offensive" capacity of emerging stegomalware (Caviglione et al., 2021; Strachanski et al., 2024), we considered three different scenarios, each one designed to capture possible tradeoffs in terms of stealthiness and capacity of the steganographic embedding. To this aim, we prepared the following datasets:

[1] https://github.com/cloudipsp/all_banks_ips

- **BSD70**: based on the Berkeley Segmentation Data Set (Arbelaez et al., 2011), it uses 500 legitimate images that have been cropped to 321 × 321 pixels to model a threat targeting small digital objects. Originally in JPG format, they have been converted to PNG to allow cloaking a payload through LSB steganography. Notably, such an early transformation does not introduce specific patterns that AE-based models can leverage to enhance reconstruction quality. In each image, the malware hides 70 URLs by using both methods presented in Section 3. For the "squares" hiding pattern, we used blocks of 107 × 107 pixels. The resulting dataset contains 3, 500 images composed of 500 clean images, 1, 500 images with payloads hidden with the LSB plain method (500 images for each pattern), and 1, 500 images containing payloads hidden with the LSB variant method (500 images for each pattern). For each combination, we split the dataset into train, test, and validation sets, composed of 170, 165, and 165 images, respectively (Liguori et al., 2024);
- **FFHQ70**: based on the Flickr-Faces-HQ Dataset[2], from which we selected a subset of 3, 000 legitimate images of 1, 024 × 1, 024 pixels in PNG format to model a threat using medium-sized digital objects. Similar to the previous case, the malware hides 70 URLs in each image with the two LSB steganography variants. This allowed to model a malware trying to increase its stealthiness by exploiting a larger image to conceal the same malicious payload of BSD70. For the "squares" hiding pattern, we used blocks of 341 × 341 pixels for the first two squares belonging to the diagonal of the image and 342 × 342 pixels for the last block. The resulting dataset contains 21, 000 images composed of 3, 000 clean images, 9, 000 images with payloads hidden with the LSB plain method (3, 000 images for each pattern), and 9, 000 images containing payloads hidden with the LSB variant method (3, 000 images for each pattern). For each combination, we split the dataset into train, test, and validation sets, composed of 1, 000, 1, 000, and 1, 000 images, respectively;
- **FFHQ210**: based again on the aforementioned Flickr-Faces-HQ Dataset, it extends the **FFHQ70** datasets to consider a stegomalware taking advantage of the increased hiding surface to cloak 210 URLs. This allowed to quantify the impact of bigger payloads over the sanitization approach. The composition of the dataset in terms of types of pictures and splits remains unchanged.

To implement our U-Net+ approach[3] and the two deep learning baselines, we used PyTorch (Paszke et al., 2019). The first baseline is a deep convolutional autoencoder, denoted in the following as DeepAE. The encoder consists of two convolutional layers that halve the size of the input image and produce a representation of 64 layers of size 160 × 160 pixels and 512 × 512 pixels for the BSD70 and the FFHQ70/FFHQ210 datasets, respectively. The third convolutional layer is the latent space that produces a representation of 128 layers of size 80 × 80 pixels and 256 × 256 pixels for the BSD70 and the FFHQ70/FFHQ210 datasets, respectively. The latent space is then fed into the decoder endowed with the same symmetrical structure as the encoder. The second deep learning baseline is a U-Net-like encoder-decoder model, denoted as U-Net. The U-Net structure follows the description in Section 4, but it does not include the connection scheme depicted in Fig. 3, which is characteristic of the U-Net+ model. As described above, they rely on an encoder-decoder architecture: during the encoding phase, the input image is progressively halved in size and doubled in volume using convolutional blocks until we obtain a representation of 512 layers of size 20 × 20 and 64 × 64 for the BSD70 and the FFHQ70/FFHQ210 datasets, respectively. The obtained

[2] Flickr-Faces-HQ Dataset, online: https://github.com/NVlabs/ffhq-dataset

[3] The code is available at: https://github.com/Angielica/sanitization

representation is then fed into the decoder to reconstruct the input image. Adam is used as the optimizer with a learning rate of 0.001.

To assess whether our idea could serve as a valid alternative to simpler approaches commonly deployed "in the wild", we implemented two additional sanitization schemes. In more detail, the first uses a *transcoding* scheme (Dubin, 2023): the content is disrupted by converting the original PNG image to an intermediate JPG representation and then back to the original PNG format. To this aim, we developed an ad-hoc Python script based on the Pillow 9.2.0 library[4] and the PNG to JPG conversion has been done with the quality parameter set to the default value of 75. This allowed to not bloat the size of the image and degrade the final outcome. The selected value also guarantees to use all the features of JPEG compression and prevents to limit the investigation for specific or optimized settings. The second basic mechanism disrupts the hidden content by adding *Gaussian noise* and has been again implemented in Python with the OpenCV 4.9.0[5] and the Numpy 1.23.1[6] libraries. To configure the noise generation process, we empirically found parameters that completely disrupt the hidden data, i.e., $\sigma = 0.55$ in our trials.

To quantify the performance of the various sanitization approaches, we introduced three metrics. The first is the MSE already described in Section 4. The second is the Peak Signal-to-Noise Ratio (PSNR) measuring the ratio between the maximum power of a signal and the power of the related noise, defined as:

$$PSNR(x, \hat{x}) = 20 \log_{10} \left(\frac{MAX(x)}{\sqrt{MSE(x, \hat{x})}} \right)$$

where $MAX(x)$ is a function returning the maximum possible pixel value of the image x and the MSE is computed over all the color components and properly scaled.

The third is the SSIM, which quantifies the similarity between two images. It evaluates the perceived quality of an image compared to another, focusing on structural information as well as luminance and contrast. The SSIM is defined as:

$$\text{SSIM}(x, \hat{x}) = \frac{(2\mu_x \mu_{\hat{x}} + C_1)(2\sigma_{x\hat{x}} + C_2)}{(\mu_x^2 + \mu_{\hat{x}}^2 + C_1)(\sigma_x^2 + \sigma_{\hat{x}}^2 + C_2)}$$

where μ_x, $\mu_{\hat{x}}$, σ_x^2, and $\sigma_{\hat{x}}^2$ are the mean intensities and the variances of images x and $\hat{x}$, the $\sigma_{x\hat{x}}$ is the covariance between the images, and C_1 and C_2 are small constants to stabilize the division (see, Wang et al. (2003) for further details).

Lastly, experiments were executed on an NVidia DGX Station with 4 V100 GPUs with 32 GB of RAM.

5.2 Experimental results

Here, we evaluate the capability of our solution to remove malicious content while preserving the quality of the image. As a first step, we verified if the payload cloaked within the various images survived the sanitization process. To this extent, we developed a Python script to automatically extract the various URLs from the sanitized images. All the methods demonstrated their effectiveness to disrupt the information needed to implement the attack chain.

[4] https://pillow.readthedocs.io/en/stable/

[5] https://opencv.org/get-started/

[6] https://numpy.org/

Specifically, sanitization approaches leveraging AI and transcoding completely disrupted the cloaked URLs and even the presence of the `*d*` escape sequence did not allow to identify partial entries. Sanitization through Gaussian noise rendered the URLs unusable as well. However, this method left some fragments of a few characters, which, in any case, do not allow a threat actor to reconstruct the required data.

Tables 2 and 3 show the performances achieved by the different sanitization mechanisms against the three test cases, BSD70, FFHQ70, and FFHQ210, respectively. As regards the BSD70 case, the transcoding mechanism and U-Net-based architectures exhibit the best performance in yielding high quality images with respect to Gaussian noise and DeepAE. Notably, the U-Net+ achieves the lowest MSE and the highest PSNR while exhibiting an SSIM comparable to a standard U-Net. This result can be ascribed to the use of dense connections between convolutional layers at different resolutions. Specifically, they allow convolutional layers of varying depths to interact in a more complex way, enabling a more detailed information flow compared to the direct connections in the classic U-Net.

A similar trend can be observed for the FFHQ70 and FFHQ210 datasets in Table 3. Indeed, also in this case, transcoding and U-Net-based architectures outperform the Gaussian noise and DeepAE algorithms. However, while transcoding the image can lead to better performance in terms of MSE and PSNR, it could cause information leakages. In fact, the JPG algorithm may leave artifacts suggesting that the digital object has undergone a compression/decompression process. As it will be detailed in Section 6, this could make the sanitization process more fragile or prone to attacks.

Execution Times and Convergence Analysis To deploy our mechanism in production-quality scenarios, it is important to evaluate the timing footprint of the sanitization process. We then analyzed the time needed by the various AI-based models for the training and inference phases. Figure 4 depicts the obtained results. As the model complexity increases, both time frames increase accordingly. Although the DeepAE presents lower training and inference temporal requirements compared to U-Net and U-Net+, the performance in terms of MSE, PSNR and SSIM are notably worse than these two models.

Furthermore, we conducted a convergence analysis for all the AI-based approaches considered in this work. Figure 5 portraits the results. Specifically, regardless of the dataset used, the U-Net and U-Net+ architectures require fewer epochs to converge. This behavior could be primarily ascribed to the use of skip connections. Recalling that the loss of the U-Net+ defined in (1) is the contribution of $L = 4$ components, we also included the value for the output layer specifically responsible for reconstructing the final image. This trend is indicated with U-Net+ (single) in Fig. 5.

Lastly, we evaluated the time needed for both the transcoding and the Gaussian noise addition sanitization mechanisms. For the case of transcoding, we measured the time elapsed to convert PNG to JPG and then back from JPG to PNG. For the BSD70 dataset, the transcoding time is 0.033 seconds. However, for the FFHQ70 and FFHQ210 datasets, this time increases to 0.44 seconds, primarily due to the larger image sizes. As expected, adding Gaussian noise required significantly less time than the transcoding process, even if also its performance is worse, see Tables 2 and 3. Specifically, for the BSD70 dataset, the time is equal to 0.012 seconds, while for the FFHQ70 and FFHQ210 datasets, it is 0.11 seconds.

Qualitative Analysis To further evaluate the impact of the sanitization, we reported the "visual" comparison between a malicious image and its counterparts obtained through the various sanitization approaches. To better highlight results, the figures contain *red* arrows pointing at the *malicious* hidden content, *yellow* arrows indicating *pixel artifacts* introduced

Table 2 Comparative analysis of the considered sanitization algorithms for the BSD70 dataset

Algorithm	MSE			PSNR			SSIM		
	Sequential	Rows	Squares	Sequential	Rows	Squares	Sequential	Rows	Squares
BSD70									
	LSB Plain								
Transcoding	4.5e-4	4.5e-4	4.5e-4	34.01	34.01	34.01	0.94	0.94	0.94
Gaussian noise	1.4e-2	1.4e-2	1.4e-2	18.83	18.83	18.83	0.51	0.51	0.50
DeepAE	6.29e-3	6.29e-3	6.29e-3	22.36	22.36	22.36	0.76	0.76	0.76
U-Net	2.08e-4	2.08e-4	2.08e-4	37.26	37.26	37.26	**0.98**	**0.98**	**0.98**
U-Net+	**1.65e-4**	**1.65e-4**	**1.65e-4**	**38.47**	**38.47**	**38.47**	**0.98**	**0.98**	**0.98**
	LSB Variant								
Transcoding	4.5e-4	4.5e-4	4.5e-4	34.01	34.01	34.01	0.94	0.94	0.94
Gaussian noise	1.4e-2	1.4e-2	1.4e-2	18.83	18.83	18.83	0.51	0.51	0.51
DeepAE	6.29e-3	6.29e-3	6.29e-3	22.36	22.36	22.36	0.76	0.76	0.76
U-Net	2.08e-4	2.08e-4	2.08e-4	37.27	37.27	37.27	**0.98**	**0.98**	**0.98**
U-Net+	**1.65e-4**	**1.65e-4**	**1.65e-4**	**38.47**	**38.47**	**38.46**	**0.98**	**0.98**	**0.98**

Best results are reported in bold

Table 3 Comparative analysis of the considered sanitization algorithms for the FFHQ70 and FFHQ210 datasets

Algorithm	MSE			PSNR			SSIM		
	Sequential	Rows	Squares	Sequential	Rows	Squares	Sequential	Rows	Squares
FFHQ70									
	LSB Plain								
Transcoding	**1.3e-4**	**1.3e-4**	**1.3e-4**	**39.04**	**39.04**	**39.04**	**0.95**	**0.95**	**0.95**
Gaussian noise	1.3e-2	1.3e-2	1.3e-2	18.93	18.93	18.93	0.40	0.40	0.40
DeepAE	1.7e-2	1.7e-2	1.7e-2	18.07	18.07	18.07	0.76	0.76	0.76
U-Net	3.06e-4	3.06e-4	3.06e-4	35.50	35.50	35.50	**0.95**	**0.95**	**0.95**
U-Net+	3.94e-4	3.94e-4	3.94e-4	34.26	34.26	34.26	**0.95**	**0.95**	**0.95**
	LSB Variant								
Transcoding	**1.3e-4**	**1.3e-4**	**1.3e-4**	**39.04**	**39.04**	**39.04**	**0.95**	**0.95**	**0.95**
Gaussian noise	1.3e-2	1.3e-2	1.3e-2	18.93	18.93	18.93	0.40	0.40	0.40
DeepAE	1.7e-2	1.7e-2	1.7e-2	18.08	18.08	18.08	0.76	0.76	0.76
U-Net	3.06e-4	3.06e-4	3.06e-4	35.50	35.50	35.50	**0.95**	**0.95**	**0.95**
U-Net+	3.95e-4	3.95e-4	3.95e-4	34.37	34.37	34.37	0.94	0.94	0.94
FFHQ210									
	LSB Plain								
Transcoding	**1.3e-4**	**1.3e-4**	**1.3e-4**	**39.04**	**39.04**	**39.04**	**0.95**	**0.95**	**0.95**
Gaussian noise	1.3e-2	1.3e-2	1.3e-2	18.93	18.93	18.93	0.40	0.40	0.40
DeepAE	1.7e-2	1.7e-2	1.7e-2	18.08	18.08	18.08	0.76	0.76	0.76
U-Net	3.06e-4	3.06e-4	3.06e-4	35.50	35.50	35.50	**0.95**	**0.95**	**0.95**
U-Net+	3.95e-4	3.95e-4	3.95e-4	34.38	34.38	34.38	0.94	0.94	0.94

Table 3 continued

Algorithm	MSE			PSNR			SSIM		
	Sequential	Rows	Squares	Sequential	Rows	Squares	Sequential	Rows	Squares
FFHQ210									
	LSB Variant								
Transcoding	**1.3e-4**	**1.3e-4**	**1.3e-4**	**39.04**	**39.04**	**39.04**	**0.95**	**0.95**	**0.95**
Gaussian noise	1.3e-2	1.3e-2	1.3e-2	18.93	18.93	18.93	0.40	0.40	0.40
DeepAE	1.7e-2	1.7e-2	1.7e-2	18.08	18.08	18.08	0.76	0.76	0.76
U-Net	3.06e-4	3.06e-4	3.06e-4	35.50	35.50	35.50	**0.95**	**0.95**	**0.95**
U-Net+	3.95e-4	3.95e-4	3.95e-4	34.37	34.37	34.37	0.94	0.94	0.94

Best results are reported in bold

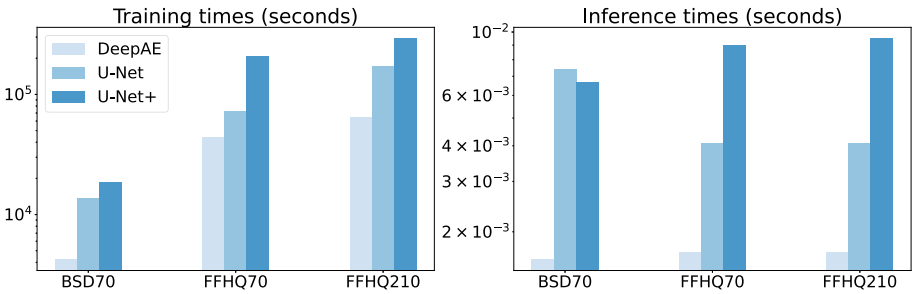

Fig. 4 Training and inference times in log scale of the different neural architectures

by the sanitization process, and *green* arrows highlighting the major *differences* between the sanitized and the original image. To not burden the results and for the sake of brevity, we omitted the DeepAE and U-Net architectures or images belonging to the FFHQ70 dataset. Figures 6 and 7 contain the "master" images, i.e., before the attacker embedded the list of URLs through the various LSB mechanisms, along with the relevant bitplanes. Such images serve as a reference to understand the impacts in terms of "quality" for all the hiding methods and sanitization approaches.

In more detail, Figs. 8 and 9 provide an overview for a sample image of the BSD70 dataset with data hidden with the LSB plain and LSB variant steganographic methods, whereas Figs. 10 and 11 depict a similar comparison for a sample image of the FFHQ210 dataset. As shown, almost all sanitization approaches do not provide any alterations that can be spotted through the naked eye. The only notable difference is when the Gaussian noise is added to the image belonging to the BSD70 dataset, mainly due to its reduced size compared to the one of the FFHQ210 dataset. Instead, the inspection of the least significant bitplanes reveals the presence of "visual artifacts" characterizing each sanitization technique. For instance, transcoding tends to create block artifacts (i.e., due to the JPG compression operated on squares of 8×8 pixels), see, e.g., Fig. 8.

Figures 9 and 11 clearly show the visual alterations caused by the presence of the malicious URLs when the LSB variant deployed by OceanLotus is used. Specifically, the hidden information leads to "linear" patterns, which are clearly visible in the 3-LSB bitplanes. Regarding the use of the U-Net+ sanitization approach, Fig. 9 is characterized by some colors altered in certain areas, such as green tones changing to light blue, or appearing with an increased intensity. However, besides such changes in the color hue/intensity for some bitplanes, no

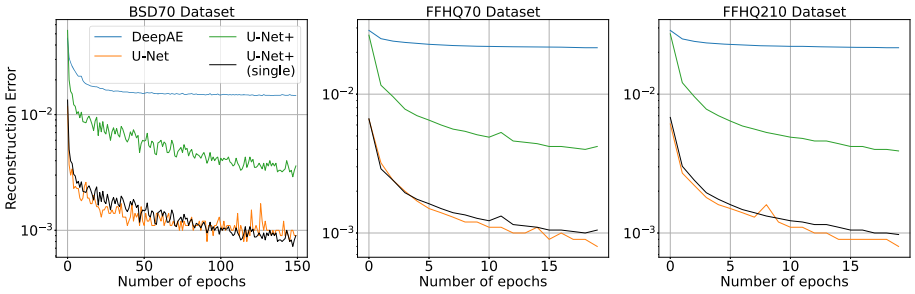

Fig. 5 Loss trends of the different neural architectures across training epochs

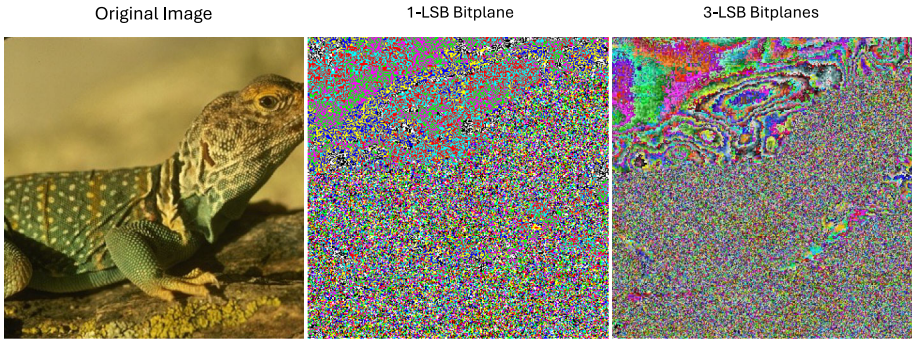

Fig. 6 Reference image for the BSD70 dataset, along with the first LSB bitplane and the first three LSB bitplanes.

other remarkable artifacts are introduced. We point out that, bitplanes with a three-bit depth exhibit artifacts that can convey some form of information. Specifically, the presence of color blocks, high-frequency alterations (i.e., per-pixel noise shots), and Moirè-like patterns may account for additional security issues. For this reason, the "visual" resemblance of images may not be sufficient to fully understand the impact of the sanitization process. In fact, evaluating whether imperceptible artifacts may open to new attack scenarios should be considered relevant for performing sanitization in production-quality deployments.

6 Security considerations

Despite the technique used to disrupt the hidden data, deploying a tool for sanitizing digital images in real scenarios requires to face several security issues. The most relevant concerns are the creation of side-channel attacks, i.e., offensive schemes based on information that can be obtained due to how a system behaves (Bazm et al., 2017). Specifically, Tables 2 and 3 show that the presence of a middlebox doing some form of processing could be simply inferred by inspecting the MSE, PSNR, and SSIM quality metrics and compare them against similar "ground truth" contents. An attacker could then distribute legitimate and modified

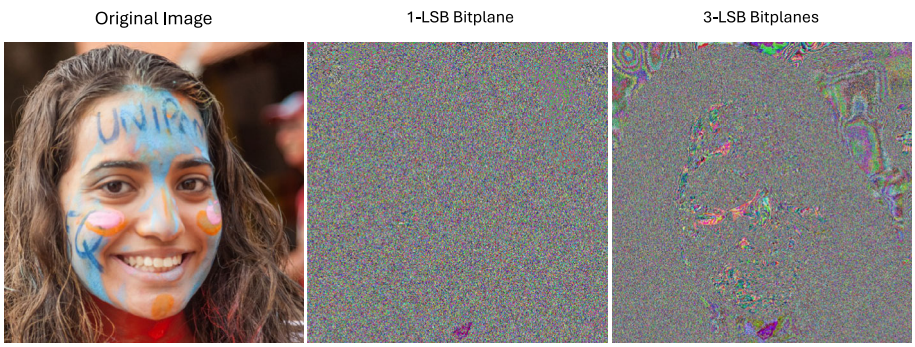

Fig. 7 Reference image for the FFHQ210 dataset, along with its first LSB bitplane and the first three LSB bitplanes.

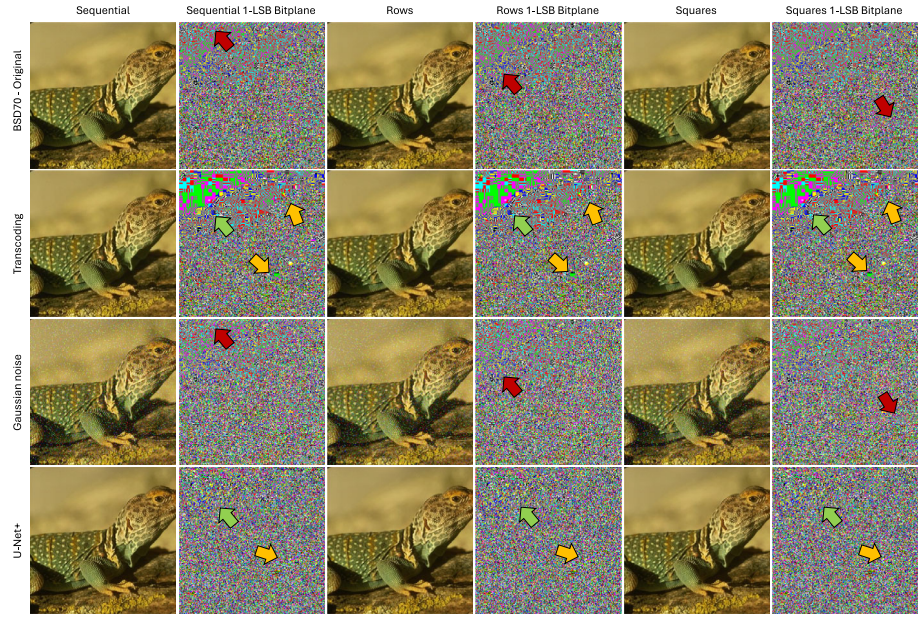

Fig. 8 Original image of the BSD70 dataset containing URLs embedded via the LSB plain method with the three encoding patterns, the images sanitized via different algorithms, and the corresponding first bitplanes

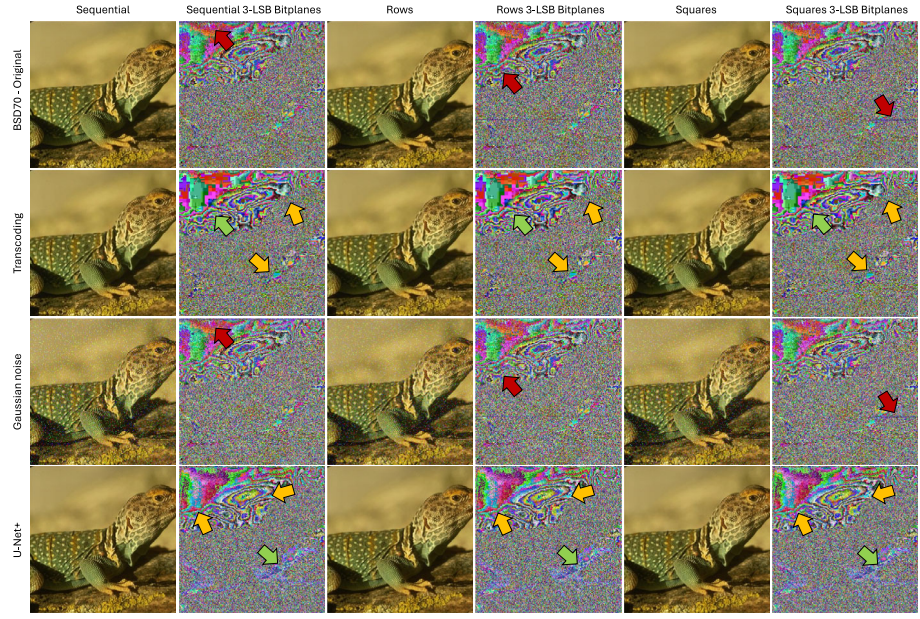

Fig. 9 Original image of the BSD70 dataset containing URLs embedded via the LSB variant method with the three encoding patterns, the images sanitized via different algorithms, and the corresponding first bitplanes

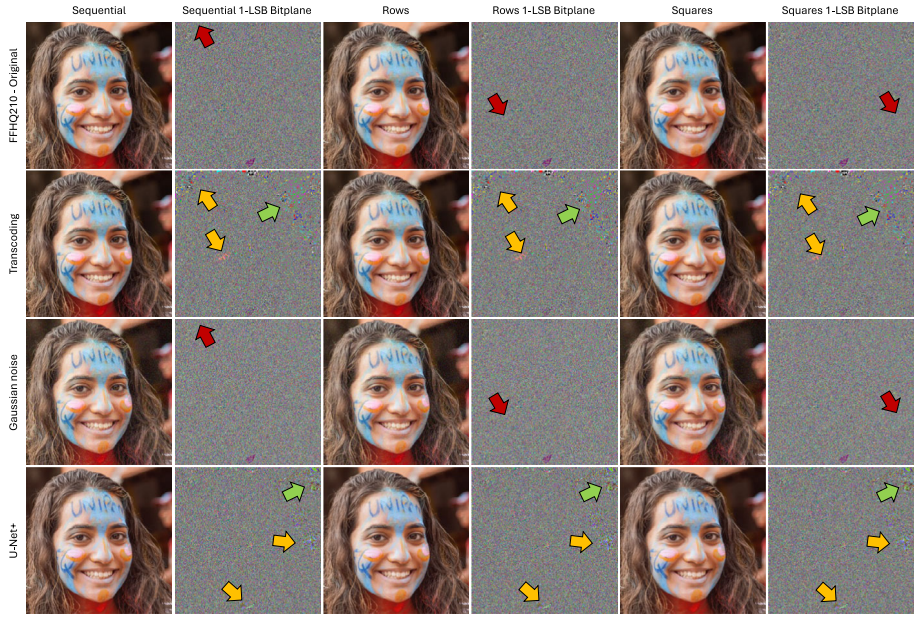

Fig. 10 Original image of the FFHQ210 dataset containing URLs embedded via the LSB plain method with the three encoding patterns, the images sanitized via different algorithms, and the corresponding first bitplanes

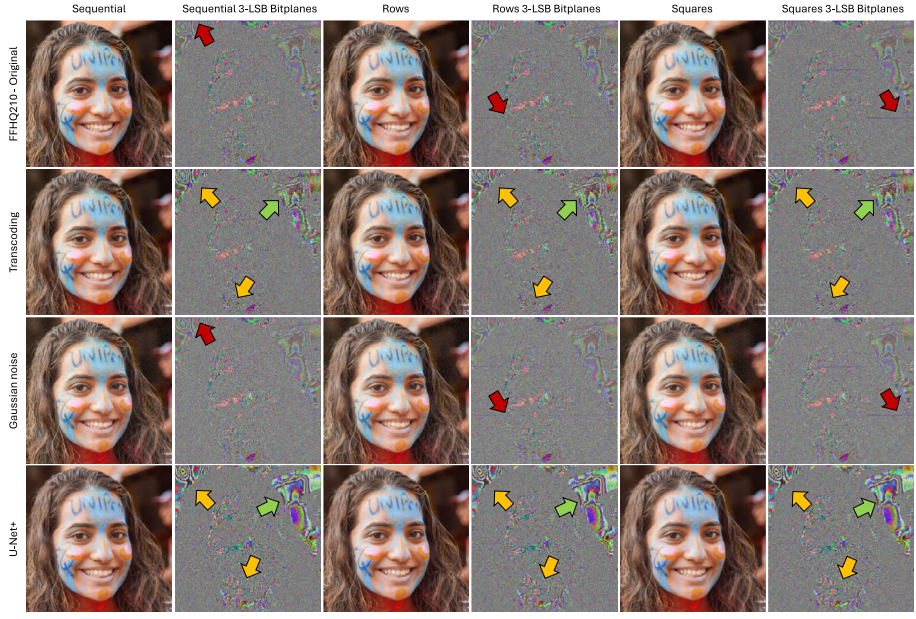

Fig. 11 Original image of the FFHQ210 dataset containing URLs embedded via the LSB variant method with the three encoding patterns, the images sanitized via different algorithms, and the corresponding first bitplanes

images and recollect them to tweak the attack campaign to: *i*) probe a network/infrastructure to identify routes/services "protected" by the sanitizer (Bazm et al., 2017); *ii*) test different stegomalware implementations (e.g., by cloaking data in attachments, documents, or applications) to find the scope of the sanitization process (Mazurczyk & Caviglione, 2021); *iii*) use different steganographic mechanisms and visual features to reverse-engineer the sanitization process (Caputo et al., 2020).

With the aim of understanding whether the proposed sanitization mechanisms could be prone to side-channel attacks, we performed additional tests. Specifically, we used the datasets and hiding methods described in Section 5 to quantify whether the sanitization process leads to signatures that an attacker can use to gain an advantage, e.g., to develop some sanitization-resistant approaches (Cheddad et al., 2010).

In more detail, an attacker could easily recognize that a content has been sanitized through JPG transcoding by exploiting the "block artifacts" left in the output image. In fact, the JPG compression algorithm partitions the image into small blocks (8×8 pixels, in our setting) and computes their Discrete Cosine Transform (DCT). Since each block is processed independently, smoothness or continuity properties among adjacent blocks are seldom preserved. This is further emphasized by the quantization of the DCT coefficients, which heavily impairs or "flattens" the high-frequency values, especially for blocks representing uniform areas. In general, "block artifacts" introduced by the JPG compression also persist in the image reverted to its original PNG format, thus giving the attacker a clear indication of the intermediate processing steps used to disrupt the secret data. This "side" information can be observed in Figs. 8, 9, 10, and 11, where the most prominent areas containing JPG-induced blocks have been indicated with green arrows. The resulting side channel can be easily exploited, since the literature abounds of techniques for detecting artifacts or a wide-array of forgery attacks, e.g., due to resampling (Popescu & Farid, 2005) or anomalous behaviors of DCT-processed blocks (Chen & Hsu, 2008).

The presence of Gaussian noise could be easily revealed as well. In fact, the "scarcity" of hidden data compared to the overall area of the targeted image requires a non-negligible amount of noise, leading to artifacts even visible to the naked eye (see, Figs. 8, 9, 10, and 11). Indeed, sanitization methods based on noise can be efficiently spotted through simple filtering techniques. Specifically, Fig. 12 depicts the malicious image and its counterpart sanitized with all the considered mechanisms. To reveal the noise, we processed the images with a 5×5 kernel filter and an adaptive threshold computed over the mean value of 21×21

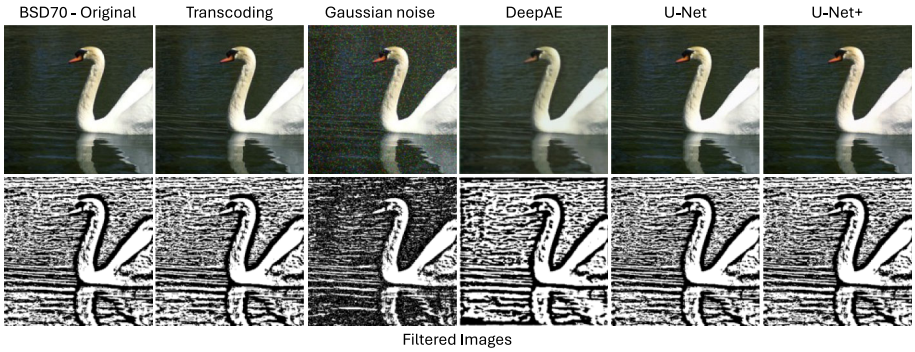

Fig. 12 Impact of the sanitization process in terms of possible leaking signatures. A smoothing filter is applied to emphasize high-frequency artifacts introduced by the various sanitization techniques

blocks of pixels (Yukawa, 2012). This allows to clearly emphasize the high-frequency components arising from the use of Gaussian noise, which can be the basis for distilling possible signatures. The resulting side-channel information can also be used programmatically to spot the presence of a Gaussian sanitizer, e.g., by revealing the presence of noisy signals through a condensed metric. A viable approach is presented in Tian et al. (2018), where the Fast Fourier Transform is used to "coalesce" the energy of high-frequency components in a unique numerical value. For the case of images depicted in Fig. 12, we obtained a global energy value of 143.04 for the reference image. Instead, for the case of JPG-transcoding and Gaussian noise, we obtained an energy of 143.35, and 173.62, respectively. Moreover, when AI-based sanitization mechanisms are used, we obtained 125.88, 141.82, and 142.58, respectively. Thus, a simple threshold rule allows the identification of images sanitized through additive Gaussian noise strategies.

A threat actor may also spot the presence of AI-based sanitization mechanisms by making a direct comparison between a sample and a processed image. Yet, our approach does not have a clear signature that could be used to launch an effective "reversing" campaign. In fact, despite the used neural architecture, our approach does not offer any "explainable" indicators that can be employed to prepare a robust hiding scheme able to beat the sanitization process.

As a final remark, the availability of an efficient model for sanitizing digital images should be considered a double-edged sword. In fact, if publicly available (or leaked), it could be used to disrupt legitimate information. An attacker could hijack the model to remove watermarks, copyright protections, or metadata for tracking and integrity purposes (Wan et al., 2022).

7 Conclusions and future work

In this paper, we addressed the problem of sanitizing digital images containing a list of malicious URLs. To this aim, we explored the use of several Unsupervised Deep Learning architectures i.e., Deep Autoencoders, U-Net, and a variant of the standard U-Net+. Results demonstrated the effectiveness of the U-Net+ in terms of quality of sanitized images. We also compared our approach with two basic strategies often deployed in realistic scenarios, i.e., transcoding the image in an intermediate format and a naive addition of Gaussian noise. Despite our approach does not always outperform them in some configurations, it should be preferred since the U-Net+ sanitization mechanism does not lead to recognizable signatures, which could be exploited by a threat actor to void the countermeasure. Similar to other AI-capable frameworks, our solution demands for a suitable amount of images containing hidden information to learn an effective functional mapping. Part of our ongoing research then investigates semi-supervised methods to train reliable models on scarce data. A further possible refinement is to embed a regularization component in the loss function for estimating the probability that the secret has been destroyed.

Future works aim at making our sanitization technique more suitable for large-scale scenarios with tight privacy and computational constraints. A future development considers relying upon a federated paradigm, e.g., to train the model in a decentralized fashion. Another major future research item deals with investigating alternative cloaking mechanisms (e.g., Invoke-PSImage) and other hidden payloads, such as configuration files or small attack routines. Lastly, the availability of an efficient sanitization mechanism could also be used by an attacker to conceal legitimate hidden data, i.e., to remove watermarks used to enforce copyright. For this reason, a future refinement aims at making the model selective, e.g., it should "refuse" to sanitize non-malicious data.

Acknowledgements This work was supported by Project SERICS (PE00000014) and Project RAISE (ECS00000035) under the NRRP MUR program funded by the EU - NGE as well as by Project WHAM! (B53D23013340006) funded by the PRIN2022 framework.

Author Contributions All authors equally contributed to the work.

Funding Open access funding provided by Consiglio Nazionale Delle Ricerche (CNR) within the CRUI-CARE Agreement. Not applicable.

Data Availability The original datasets used in this research are borrowed from a third-party source and modified for our purposes and can be obtained upon request.

Declarations

Competing interests The authors declare no competing interests.

Ethics approval and consent to participate Not applicable.

References

Ameen, S. Y., & Al-Badrany, M. R. (2013). Optimal Image Steganography Content Destruction Techniques. *International Conference on Systems* (pp. 453–457). Signal Processing and Informatics: Control.

Arbelaez, P., Maire, M., Fowlkes, C., et al. (2011). Contour Detection and Hierarchical Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 33*(5), 898–916.

Bank, D., Koenigstein, N., & Giryes, R. (2023). Autoencoders. Machine Learning for Data Science Handbook pp 353–374

Bastani, V., Helfroush, M. S., & Kasiri, K. (2010). Image compression based on spatial redundancy removal and image inpainting. *Journal of Zhejiang University Science C, 11*(2), 92–100.

Bazm, M. M., Lacoste, M., Südholt, M., & et al. (2017). Side-channels Beyond the Cloud Edge: New Isolation Threats and Solutions. In: 2017 1st Cyber Security in Networking Conference, pp 1–8

Blasco, J., Hernandez-Castro, J. C., de Fuentes, J. M., et al. (2012). A Framework for Avoiding Steganography Usage Over HTTP. *Journal of Network and Computer Applications, 35*(1), 491–501.

Caputo, D., Verderame, L., Ranieri, A., et al. (2020). Fine-hearing Google Home: Why Silence Will Not Protect Your Privacy. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, 11*(1), 35–53.

Cassavia, N., Caviglione, L., Guarascio, M., et al. (2023). A federated approach for detecting data hidden in icons of mobile applications delivered via web and multiple stores. *Social Network Analysis and Mining, 13*(1), 114.

Caviglione, L., & Mazurczyk, W. (2022). Never Mind the Malware, Here's the Stegomalware. *IEEE Security & Privacy, 20*(5), 101–106.

Caviglione, L., Choraś, M., Corona, I., et al. (2021). Tight Arms Race: Overview of Current Malware Threats and Trends in Their Detection. *IEEE Access, 9*, 5371–5396.

Caviglione, L., Comito, C., Guarascio, M., & et al. (2023). Emerging Challenges and Perspectives in Deep Learning Model Security: A Brief Survey. *Systems and Soft Computing*

Cheddad, A., Condell, J., Curran, K., et al. (2010). Digital Image Steganography: Survey and Analysis of Current Methods. *Signal Processing, 90*(3), 727–752.

Chen, Y. L., & Hsu, C. T. (2008). Image Tampering Detection by Blocking Periodicity Analysis in JPEG Compressed Images. In: 2008 IEEE 10th Workshop on Multimedia Signal Processing, pp. 803–808

Chin, M., & Corizzo, R. (2024). Continual semi-supervised malware detection. *Machine Learning and Knowledge Extraction, 6*(4), 2829–2854.

Chua, M., & Balachandran, V. (2018). Effectiveness of Android Obfuscation on Evading anti-Malware. In: ACM Conference on Data and Application Security and Privacy

D'Angelo, G., Ficco, M., & Palmieri, F. (2020). Malware Detection in Mobile Environments Based on Autoencoders and API-images. *Journal of Parallel and Distributed Computing, 137*, 26–33.

Dubin, R. (2023). Content Disarm and Reconstruction of RTF Files a Zero File Trust Methodology. *IEEE Transactions on Information Forensics and Security, 18*, 1461–1472.

Gaber, M. G., Ahmed, M., & Janicke, H. (2024). Malware detection with artificial intelligence: A systematic literature review. *ACM Computing Surveys, 56*(6), 1–33.

Geetha, S., Subburam, S., Selvakumar, S., et al. (2021). Steganogram removal using multidirectional diffusion in Fourier domain while preserving perceptual image quality. *Pattern Recognition Letters, 147*, 197–205.

Gibert, D., Mateu, C., & Planes, J. (2020). The Rise of Machine Learning for Detection and Classification of Malware: Research Developments, Trends and Challenges. *Journal of Network and Computer Applications, 153*, 102526.

Han, R., Yang, C., Ma, J., & et al. (2020). IMShell-Dec: Pay More Attention to External Links in PowerShell. In: ICT Systems Security and Privacy Protection, pp. 189–202

Jung, D. S., Lee, S. J., & Euom, I. C. (2020). ImageDetox: Method for the Neutralization of Malicious Code Hidden in Image Files. *Symmetry, 12*(10), 1621.

Liguori A, Zuppelli M, Gallo D, et al (2024) Erasing the Shadow: Sanitization of Images with Malicious Payloads Using Deep Autoencoders. In: International Symposium on Methodologies for Intelligent Systems, pp. 115–125

Mazurczyk, W., & Caviglione, L. (2015). Information Hiding as a Challenge for Malware Detection. *IEEE Security & Privacy, 13*(2), 89–93.

Mazurczyk, W., & Caviglione, L. (2021). Cyber Reconnaissance Techniques. *Communications of the ACM, 64*(3), 86–95.

Mohaisen, A., & Alrawi, O. (2013). Unveiling Zeus: Automated Classification of Malware Samples. In: International Conference on World Wide Web, pp. 829–832

Nguyen, P. S., Huy, T. N., Tuan, T. A., et al. (2025). Hybrid feature extraction and integrated deep learning for cloud-based malware detection. *Computers & Security, 150*, 104233.

Paszke, A., Gross, S., Massa, F., & et al. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: NeurIPS

Popescu, A. C., & Farid, H. (2005). Exposing Digital Forgeries by Detecting Traces of Resampling. *IEEE Transactions on Signal Processing, 53*(2), 758–767.

Puchalski, D., Caviglione, L., Kozik, R., & et al. (2020). Stegomalware Detection Through Structural Analysis of Media Files. In: Proceedings of the 15th International Conference on Availability, Reliability and Security, pp. 1–6

Robinette, P. K., Moyer, D., & Johnson, T. T. (2023a). Monsters in the Dark: Sanitizing Hidden Threats with Diffusion Models. arXiv

Robinette, P. K., Wang, H. D., Shehadeh, N., & et al. (2023b). SUDS: Sanitizing Universal and Dependent Steganography. In: European Conference on Artificial Intelligence

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. pp. 234–241

Rus, C., Sarmah, D. K., & El-Hajj, M. (2023). Defeating MageCart Attacks in a NAISS Way. In: 20th International Conference on Security and Cryptography, pp. 691–697

Schaffhauser, A., Mazurczyk, W., Caviglione, L., et al. (2022). Efficient Detection and Recovery of Malicious PowerShell Scripts Embedded into Digital Images. *Security and Communication Networks, 2022*(1), 4477317.

Sihag, V., Vardhan, M., & Singh, P. (2021). A Survey of Android Application and Malware Hardening. *Computer Science Review, 39*, 100365.

Singh, A. K. (2020). Data Hiding: Current Trends, Innovation and Potential Challenges. *ACM Transactions on Multimedia Computing Communications and Applications, 16*(3), 1–16.

Strachanski, F., Petrov, D., Schmidbauer, T., & et al. (2024) A Comprehensive Pattern-based Overview of Stegomalware. In: Proceedings of the 19th International Conference on Availability, Reliability and Security, pp. 1–10

Suarez-Tangil, G., Tapiador, J. E., & Peris-Lopez, P. (2014). Stegomalware: Playing Hide and Seek With Malicious Components in Smartphone Apps. In: International Conference on Information Security and Cryptology, pp. 496–515

Tan, M., Guo, D., Huang, L., & et al. (2024). Image Sterilization via Overwriting. *Information Sciences* 690

Tao, H., Chongmin, L., Zain, J. M., et al. (2014). Robust Image Watermarking Theories and Techniques: A review. *Journal of Applied Research and Technology, 12*(1), 122–138.

Tian, C., Zhang, Q., Sun, G., et al. (2018). FFT consolidated sparse and collaborative representation for image classification. *Arabian Journal for Science and Engineering, 43*, 741–758.

Vincent, P., Larochelle, H., Bengio, Y., & et al. (2008). Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th International Conference on Machine Learning, pp. 1096–1103

Wan, W., Wang, J., Zhang, Y., et al. (2022). A Comprehensive Survey on Robust Image Watermarking. *Neurocomputing, 488*, 226–247.

Wang, Z., Simoncelli, E. P., & Bovik, A. C. (2003). Multiscale Structural Similarity for Image Quality Assessment. In: The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003, IEEE, pp. 1398–1402

Yukawa, M. (2012). Multikernel Adaptive Filtering. *IEEE Transactions on Signal Processing, 60*(9), 4672–4682.

Zhao, W., Wu, J., & Meng, Z. (2025). AppPoet: Large Language Model Based Android Malware Detection via Multi-view Prompt Engineering. *Expert Systems with Applications, 262*, 125546.

Zhou, Z., Siddiquee, M. M. R., Tajbakhsh, N., et al. (2019). UNet++: Redesigning Skip Connections to Exploit Multiscale Features in Image Segmentation. *IEEE Transactions on Medical Imaging, 39*, 1856–1867.

Zhu, Z., Wei, P., Qian, Z., et al. (2023). Image Sanitization in Online Social Networks: A General Framework for Breaking Robust Information Hiding. *IEEE Transactions on Circuits and Systems for Video Technology, 33*(6), 3017–3029.

Zimba, A., Wang, Z., & Chen, H. (2018). Multi-stage Crypto Ransomware Attacks: A new Emerging Cyber Threat to Critical Infrastructure and Industrial Control Systems. *ICT Express, 4*(1), 14–18.

Zuppelli, M., Manco, G., Caviglione, L., & et al. (2021). Sanitization of Images Containing Stegomalware via Machine Learning Approaches. In: Proceedings of the Italian Conference on Cybersecurity, pp. 374–386