SUDS: Sanitizing Universal and Dependent Steganography

Preston K. Robinette[a;*], **Hanchen D. Wang**[a], **Nishan Shehadeh**[a], **Daniel Moyer**[a] **and Taylor T. Johnson**[a]

[a]Vanderbilt University

Abstract. Steganography, or hiding messages in plain sight, is a form of information hiding that is most commonly used for covert communication. As modern steganographic mediums include images, text, audio, and video, this communication method is being increasingly used by bad actors to propagate malware, exfiltrate data, and discreetly communicate. Current protection mechanisms rely upon steganalysis, or the detection of steganography, but these approaches are dependent upon prior knowledge, such as steganographic signatures from publicly available tools and statistical knowledge about known hiding methods. These dependencies render steganalysis useless against new or unique hiding methods, which are becoming increasingly common with the application of deep learning models. To mitigate the shortcomings of steganalysis, this work focuses on a deep learning sanitization technique called SUDS that is not reliant upon knowledge of steganographic hiding techniques and is able to sanitize universal and dependent steganography. SUDS is tested using least significant bit method (LSB), dependent deep hiding (DDH), and universal deep hiding (UDH). We demonstrate the capabilities and limitations of SUDS by answering five research questions, including baseline comparisons and an ablation study. Additionally, we apply SUDS to a real-world scenario, where it is able to increase the resistance of a poisoned classifier against attacks by 1375%.

1 Introduction

Steganography is the art of hiding information in plain sight in order to discreetly communicate [5, 7]. Deriving from the Greek words "steganos" (covered) and "grafia" (writing), steganography literally means covered writing, and it is prevalent throughout much of history. Whereas cryptography uses encryption to make a message incomprehensible to the naked eye, steganography hides the traces of the communication entirely. By human nature, an encrypted message attracts attention and incites scrutiny. People are attracted to the allure of breaking a cypher and are often successful in breaking encryption keys given enough time and computational resources. Steganography, however, has the benefit of escaping the scrutiny of unassuming eyes, offering a discreet method of communication which can be used to hide both encrypted and unencrypted information, making it a potentially dangerous attack vector for bad actors.

Steganography has been used across several creative mediums, including books, knitting, and wax tablets. Modern approaches most commonly rely on digital media such as images, audio, text, and videos. As digital media is easily distributed and widely spread, the potential effects of steganography have grown exponentially, making it important to be able to protect against this type of communication if used by bad actors. While steganography can be used for applications like watermarking proprietary information [9] and light field messaging [19], attackers can leverage this communication technique to propagate malware [14], exfiltrate victim data [6], and communicate.

In an effort to limit the adverse effects of steganography, recent research has focused on steganalysis, or the detection of steganography. If, for instance, an advertisement containing a malware payload is detected on a web browser, this image can be removed from the site, protecting clients from interacting with these malicious images. Current steganalysis techniques rely on statistical image tests and signatures associated with known steganography techniques [11, 2, 12, 4, 3, 13]. Additionally, deep learning steganographic techniques utilize datasets of images created with publicly available steganography tools. Existing detection methods, therefore, rely on the assumption that a hiding signature or information about a steganography technique is already known. What happens, then, as new, not publically available hiding methods begin to surface, making current detection techniques irrelevant? While an increasingly difficult problem to address, detection should not be abandoned. Instead, it would be more advantageous to use detection in conjunction with other protective mechanisms, providing a more robust protection system.

As such, this work focuses on a sanitization framework for image steganography. The sanitization of an image eliminates any hidden information within the image while keeping the quality of the actual image the same. Whereas detection is used to identify steganographic images, sanitization mitigates the use of steganography entirely. Drawing upon research related to denoising images, this work uses a variational autoencoder framework to create a deep learning model that **s**anitizes **u**niversal and **d**ependent **s**teganography, called SUDS. To the best of the authors' knowledge, SUDS is the first sanitization framework which can mitigate the effects of traditional, dependent deep, and universal deep hiding. The contributions of this work, therefore, are the following:

1. **Implementation of a Robust Sanitizer.** We construct and train a framework capable of sanitizing traditional, dependent deep, and universal deep hiding steganography techniques, called SUDS.
2. **Demonstration of Sanitizer Capabilities** We show the benefit of this novel framework by evaluating SUDS on five capabilities: ability to sanitize, comparison to noise, flexibility of the latent dimension (ablation study), ability to detect, and scalability.
3. **Case Study Application.** We demonstrate a use case of SUDS

* Corresponding Author. Email: preston.k.robinette@vanderbilt.edu

whereby SUDS is able to protect against data poisoning, increasing classifier resistance against attacks by 1375%.

2 Preliminaries

While steganography can be used across any medium, this work refers to image steganography, where an image is an (c, h, w) matrix, where c is the number of color channels, h is the height of the image, and w is the width. An RGB image is then represented by an $(3, h, w)$ matrix and a grayscale image by an $(1, h, w)$ matrix.

2.1 Steganography Nomenclature

Table 1: Steganography Notation

Term	Symbol	Definition
Cover	C	The image used to hide (or cover up) a secret. A cover is combined with a secret to create a container.
Secret	S	The image to be hidden.
Container	C'	A cover image that contains a secret. A container should look identical to the cover used to conceal the secret.
Revealed Secret	S'	The secret revealed from the container. The revealed secret should be identical to the actual secret.
Sanitized	$\hat{C}$	A sanitized image.
Revealed Sanitized Secret	$\hat{S}$	The secret revealed from a sanitized image.
Hide	$\mathcal{H}$	A method to hide a secret within a cover.
Reveal	$\mathcal{R}$	A method to reveal the hidden secret within a container.
Difference	*diff*	The difference calculation between images. This work uses the $\mathcal{L}_2$ norm, also known as the mean squared error.

Steganography usually consists of four components: a cover C, a secret S, a container C', and a revealed secret S' [22, 1] as described in Table 1. A hiding method $\mathcal{H}$ takes as input a cover and a secret and produces a container such that the visible difference between the original cover and the produced container is minimal. Written more formally, $\mathcal{H}(C, S) = C'$ s.t. *diff* (C', C) is minimized, where *diff* in this work is the $\mathcal{L}_2$ norm. A reveal function $\mathcal{R}$ then takes as input the produced container and outputs the revealed secret S' such that the visible difference between the original secret and the revealed secret is minimal, or $\mathcal{R}(C') = S'$ s.t. *diff* (S', S) is minimized. An example of this steganographic process is depicted on the *Pre-Sanitization* side of Figure 2. In this work, we utilize $\hat{C}$ to represent a sanitized image such that the image is sanitized with either noise or SUDS, i.e., $\hat{C} \in \{\hat{C}_{noise}, \hat{C}_{SUDS}\}$, and $\hat{S}$ to represent a secret revealed with $\mathcal{H}$ from a sanitized image, i.e., $\mathcal{H}(\hat{C}) = \hat{S}$.

2.2 Steganography Hiding Methods

Existing methods for steganography currently fall into three main categories: traditional, dependent deep, and universal deep. We use one from each category to highlight the robustness of the proposed sanitization approach: traditional $\rightarrow$ Least Significant Bit Method (LSB) [8], dependent deep $\rightarrow$ Dependent Deep Hiding (DDH) [17, 21, 16, 15, 20, 23, 18, 1], and universal deep $\rightarrow$ Universal Deep Hiding (UDH) [22]. We utilize our own implementation of LSB[1] and a CNN-based implementation of DDH and UDH from the same code base[2], which was chosen due to its dual functionality. The hyperparameters used during UDH and DDH training are shown in Table 2. The main distinctions between these methods are highlighted in Figure 1, and we leave detailed descriptions of these methods to the behest and curiosity of the reader.

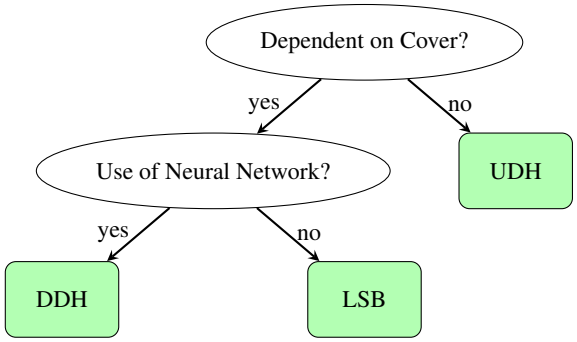

Figure 1: Dichotomy of steganography used in this work.

2.3 Sanitization

Sanitization is a method of removing an intended secret from a container image. It renders the message of the container incomprehensible. Whereas detection is used to identify steganographic images, sanitization mitigates the use of steganography entirely.

Noise Baseline A way to disrupt steganographic messages is to add noise to images. This approach not only degrades the visual rendering of the image, but it is also not a robust method for sanitization. To create a baseline of sanitization, we apply *Gaussian* noise to containers in this work. An image sanitized by noise is referred to as $\hat{C}_{noise}$, i.e., $\hat{C}_{noise} = clip(x + \mathcal{N}(\mu, \sigma), min = 0, max = 1)$, where $\mu = 0$, $\sigma = 0.02$, and the *clip* function keeps altered pixels in a meaningful range. The μ and σ values were chosen to mimic the maximum change of using LSB, which is approximately 6% for the four least significant bits ($\frac{15}{255} * 100 = 6\%$).

Variational Autoencoder This work utilizes a variational autoencoder (VAE) as the framework for the sanitization model SUDS, as shown in Figure 2. VAEs are a type of representation learning that consist of an encoder and a decoder.

- **Encoder:** The encoder Enc takes as input an input image x from a dataset D. The input image $x \in \mathbb{R}^{chw}$ is mapped to two different vectors of feature size n, representing the mean $\boldsymbol{\mu} \in \mathbb{R}^n$ and logarithm of the variance $\log \boldsymbol{\sigma^2} \in \mathbb{R}^n$ of n distributions, i.e., $Enc(x) = \boldsymbol{\mu}, \log \boldsymbol{\sigma^2}$. A latent variable is then sampled from $\boldsymbol{\mu}$ and $\log \boldsymbol{\sigma^2}$ to derive a latent vector $\mathbf{z} \in \mathbb{R}^n$ as shown in Equation (1), where $\odot$ is the element-wise product and $\epsilon \sim \mathcal{N}(0, I)$. The inclusion of $\odot\epsilon$ in (1) is known as the *reparametrization trick*, and it allows gradients to be computed and backpropagated through the network.

[1] **SUDS Code:** https://github.com/pkrobinette/suds-ecai-2023
[2] **DDH/UDH:** https://github.com/ChaoningZhang/Universal-Deep-Hiding

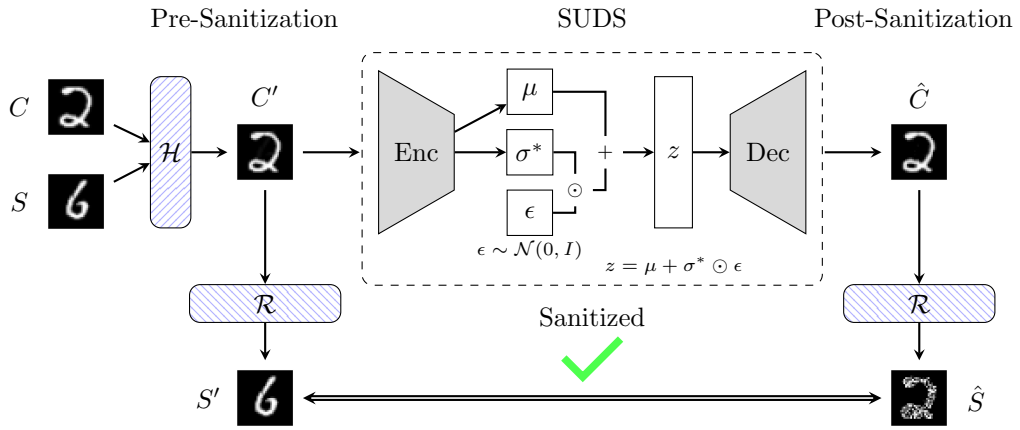

Figure 2: SUDS overview. After being trained, SUDS takes as input a cover or a container image $x \in \{C, C'\}$ created with any type of steganographic technique. Prior to being sanitized, the secret is recoverable, as demonstrated in the bottom-left of the figure S'. After sanitization with SUDS, however, the reconstructed image $\hat{C}$ still looks the same as the input C', but a secret is not recoverable as indicated by the bottom-right of the figure $\hat{S}$. This image, therefore, is successfully sanitized.

$$\mathbf{z} = f(\boldsymbol{\mu}, \log \boldsymbol{\sigma^2}) = \boldsymbol{\mu} + e^{\frac{1}{2}\log \boldsymbol{\sigma^2}} \odot \epsilon \tag{1}$$

An encoder, therefore, maps an input image to a latent representation: $Enc(x) = \boldsymbol{\mu}, \log \boldsymbol{\sigma^2} \implies \mathbf{z} = f(\boldsymbol{\mu}, \log \boldsymbol{\sigma^2})$ s.t. $\mathbb{R}^{chw} \rightarrow \mathbb{R}^n$. The depiction of this process in Figure 2 shows only σ^* for simplicity. While $e^{\frac{1}{2}\log \boldsymbol{\sigma^2}} = \sigma$, the output of Enc in our implementation is represented as the logarithm of the variance to simplify the computation of the Kullback-Leibler divergence term discussed in Section 3.3.

- **Decoder:** Once mapped to a latent variable $\mathbf{z}$, the decoder Dec can then create a reconstruction of the original image $\hat{x} \in \mathbb{R}^{chw}$ from this condensed representation, i.e., $Dec(\mathbf{z}) = \hat{x}$ s.t. $\mathbb{R}^n \rightarrow \mathbb{R}^{chw}$. The goal of the decoder is to reconstruct an image such that $diff\,(x, \hat{x})$ is minimal where $diff$ is the mean-squared error or the $\mathcal{L}_2$ norm.

In this work, input images are either a cover or container $x \in \{C, C'\}$, and the reconstructed image $\hat{x} = \hat{C}_{SUDS}$, where $\hat{C}_{SUDS}$ is an image sanitized by SUDS. An image "sanitized" by SUDS is then an image that has been encoded and decoded by the SUDS framework.

2.4 *Image Metrics*

In this work, we utilize mean-squared error (MSE) and peak-signal-to-noise ratio (PSNR) image metrics described in Equations (2) and (3) respectively, where A and B are the compared images of size (c, h, w) and MAX is the maximum possible pixel value (for a given bit depth).

$$MSE(A, B) = \frac{1}{chw} \sum_{i=1}^{c} \sum_{j=1}^{h} \sum_{k=1}^{w} (A_{i,j,k} - B_{i,j,k})^2 \tag{2}$$

$$PSNR(A, B) = 10 \log_{10} \left(\frac{MAX^2}{MSE(A, B)} \right) \tag{3}$$

3 Sanitizing Universal and Dependent Steganography

We aim to mitigate the use of image steganography to disseminate embedded secrets while preserving the integrity of the cover image through sanitization. As detection methods rely on signatures from known steganographic techniques, sanitization attempts to render the secret irretrievable, regardless of the steganographic technique used to hide the secret.

3.1 *Sanitization via Variational Autoencoder*

In order to address steganography sanitization, we utilize a variational autoencoder (VAE) to encode images into learned representations, which can then be decoded into a clean counterpart. We propose this solution as motivated by two properties related to VAEs: robustness (encoder) and expressiveness (decoder).

Robustness. In regard to the robustness of the encoder, a steganographic image can be considered an error value added to the cover, $C' = C + \epsilon$. If the VAE is robust, then any container within some bound of the cover will be mapped to the same distribution of the cover, i.e., $Enc(C + \epsilon) = Enc(C)$ s.t. $|\epsilon| < \gamma$, where Enc is an encoder, ϵ is an error value, and γ is an arbitrary error bound. In other words, covers and containers should be mapped to the same latent representation if the model is robust.

Expressiveness. Whereas robustness is related to the encoder, expressiveness is related to the decoder. By encoding a potentially steganographic image into a condensed representation, we create an information bottleneck. This property helps to bound possible decoded images, which in turn decreases the potential to decode an image from the representation which contains an embedded secret. There are more possible outcomes from a network that relies on 784 pieces of information (the original image) than from a network that uses only 128 pieces of information (the encoded latent space). The latent space representation, therefore, limits the expressiveness of the decoder, which aids in the sanitization process.

3.2 *SUDS Framework Overview*

Building upon these concepts, we propose SUDS, a VAE framework which is able to **s**anitize **u**niversal and **d**ependent **s**teganography. As shown in Figure 2, SUDS takes as an input a container or a cover and produces a sanitized version of the input. Prior to sanitization, the secret of the container C' is recoverable, as indicated by the revealed secret S' in the bottom-left of the image. After sanitization,

the intended secret is no longer recoverable, and the image quality is minimally different from that of the original cover used to create the container. The attempted recovery of the secret $\hat{S}$ shown in the bottom-right of the image is not the intended secret S, rendering the attempted communication useless and the image successfully sanitized.

3.3 SUDS Algorithm Details

During training, SUDS takes as input covers C from a dataset D. Each input image is mapped to a latent variable of size n, as described in Section 2.3. For instance, if $n = 128$, the latent variable will consist of 128 features sampled from Equation (1). To create a smooth and continuous latent space, we encourage the learned distribution to resemble a normal distribution by adding a regularizing term to the loss function, the Kullback-Leibler (KL) divergence, as shown in (4). Here, $q_\phi(z|x)$ is the conditional probability distribution of the latent variable $\mathbf{z}$ given a cover C and $\mathcal{N}(0, I)$ is the induced normal distribution. The KL divergence is a measure of how different one probability distribution is compared to another.

$$L_{reg} = KL(q_\phi(\mathbf{z}|C)||\mathcal{N}(0, I)) \tag{4}$$

Once mapped to a latent variable $\mathbf{z}$, the encoded cover image is then decoded to a reconstruction of the original image, referred to as $\hat{C}_{SUDS}$. The reconstruction loss associated with the decoder is shown in Equation (5), where $\mathbb{E}_{q_\phi(z|C)}$ is expectation given the approximate posterior distribution of the latent variable $\mathbf{z}$ given the input data C, $\hat{C}$ is the reconstructed output of the decoder, and *diff* is the $\mathcal{L}_2$ norm.

$$L_r = \mathbb{E}_{q_\phi(\mathbf{z}|C)}[diff(C, \hat{C}_{SUDS})] \tag{5}$$

The encoder and decoder are trained in tandem by combining their regularization and reconstruction loss functions, as shown in Equation (6). Training occurs for a specified number of epochs, where one epoch consists of all batched training data. It is important to note that SUDS is not trained on steganographic images, only clean images.

$$\mathcal{L} = L_r + L_{reg} \tag{6}$$

4 Research Questions and Metrics

To evaluate SUDS, we seek to answer five questions:

1. **RQ1: Ability to sanitize steganography** → *Is SUDS able to remove the presence of potential secrets while preserving the integrity of the cover?* To answer this question, we first train SUDS using the MNIST dataset as described in Section 3.3 and with the hyperparameters shown in Table 2. The MNIST training dataset consists of 60000 images $(1, h, w)$ of handwritten digits from numbers 0-9. After training, we then use 10000 images from the MNIST test dataset to create containers using each of the hiding methods, i.e. LSB, DDH, UDH. These containers are then sanitized with SUDS and passed to each reveal function. If the intended secret is discernible with any of the reveal functions, the sanitizer is not effective at cleaning images. Additionally, we evaluate SUDS using peak signal-to-noise ratio (PSNR) and mean squared error (MSE) metrics, which are commonly used image metrics. These metrics are used to assess a sanitizer's effect on the input (x compared to $\hat{C}$ s.t. $x \in \{C, C'\}$) as well as its ability to sanitize hidden secrets (S compared to $\hat{S}$). **RQ1** is positive if the revealed secret after sanitization $\hat{S}$ is not the intended secret S, i.e., $\hat{S} \not\approx S$ where $\hat{S} = \mathcal{R}(\hat{C}_{SUDS})$.
2. **RQ2: Comparison to noise baseline** → *How does SUDS compare to noise baseline techniques?* We repeat the testing process of **RQ1** but replace SUDS with Gaussian noise to sanitize the containers. We then analyze the resulting noise MSE and PSNR values against those produced by SUDS. **RQ2** is positive if SUDS is able to outperform noise in MSE and PSNR values.
3. **RQ3: Flexibility of the Latent Dimension** → *Does latent dimension size affect performance?* We train SUDS on different latent dimension sizes n to see if and when SUDS is no longer able to sanitize a steganographic image. Using the same training framework described in Section 3.3, this case study tests $n \in \{2, 4, 8, 16, 32, 64, 128\}$ and evaluates each SUDS model using MSE and PSNR image quality metrics. **RQ3** is positive if a majority (≥ 4) of the trained latent dimension sizes of SUDS are able to sanitize secrets, as indicated by MSE and PSNR values.
4. **RQ4: Ability to detect steganography** → *In addition to sanitization, can SUDS be used to detect steganography?* We use 10000 MNIST images of varying image types, i.e., cover images, LSB containers, DDH containers, and UDH containers. These images are then mapped to a latent variable z by using SUDS trained on $n = 8$ features. The containers for each hiding method are created using the same cover and secret across each method. To evaluate detection capabilities, we calculate the mean and standard deviation for each image type and digit. For instance, if Label = 4, the resulting data is calculated across all covers with the label 4 for each image type. **RQ4** is positive if the distributions across a single digit are distinguishable.
5. **RQ5: Scalability** → *Does SUDS scale to RGB images?* We use the CIFAR-10 dataset, which consists of 60000 RGB images $(3, 32, 32)$ from 10 different classes of various animals and transportation vehicles. To train SUDS on CIFAR-10, we use the same training setup detailed in Section 3.3. **RQ5** is positive if the revealed secret after sanitization $\hat{S}$ is not the intended secret S, i.e., $\hat{S} \not\approx S$ where $\hat{S} = \mathcal{R}(\hat{C}_{SUDS})$.

Table 2: Model Hyperparameter Values

Model	Hyperparameter	Value
SUDS	epochs	100
	n	128
	batch size	128
	optimizer	Adam
	learning rate	0.0001
DDH/UDH	image size	32
	batch size	44
	channel of cover	1
	channel of secret	1
	norm	"batch"
	loss	"l2"
	beta	0.75

5 Experimental Results

The results of the SUDS sanitizer training and evaluation are described in more detail below. These experiments were conducted on a macOS Monterey 12.5.1 with a 2.3 GHz 8-Core Intel Core i9 processor with 16 GB 2667 MHz DDR4 of memory.

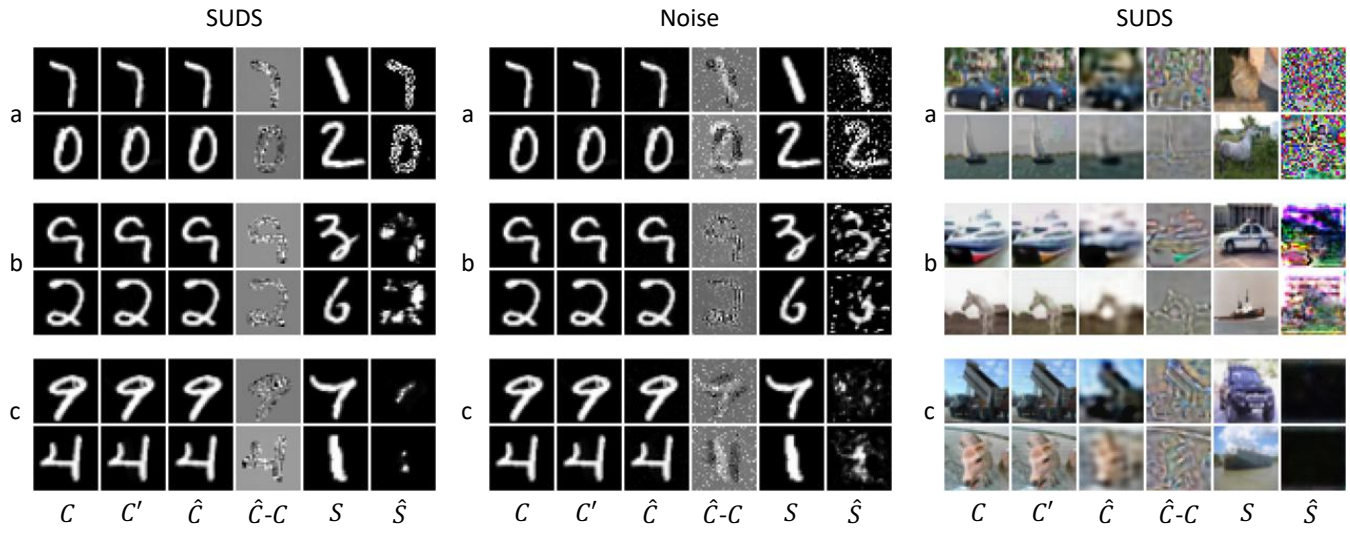

Figure 3: Image results for SUDS and noise sanitization on containers hidden with **a)** LSB, **b)** DDH, **c)** UDH. A cover C is combined with a secret S to create a container C'. The container image is then sanitized with SUDS $\hat{C}_{SUDS}$ or noise $\hat{C}_{noise}$ as indicated by the group title. A sanitizer is successful if S is not discernible from $\hat{S}$. SUDS, therefore, is successful at sanitizing, while noise is not.

5.1 Sanitization Performance via SUDS

Using the evaluation approach described in **RQ1**, the sanitization framework (SUDS) is able to effectively clean each of the container images so that the revealed secret is not obtainable through any of the reveal functions while keeping the original cover intact. As shown by the left-hand group in Figure 3, $\hat{C}_{SUDS}$ and C have limited variation, and S is not retrieved by $\mathcal{R}_{ddh}(\hat{C}_{SUDS})$, $\mathcal{R}_{udh}(\hat{C}_{SUDS})$, or $\mathcal{R}_{lsb}(\hat{C}_{SUDS})$ as shown by $\hat{S}$.

The PSNR and MSE metrics shown at the top of Table 3 further validate SUDS. The MSE and PSNR values of the *Clean* image provide a baseline for the steganographic images. The MSE and PSNR values of the $\hat{C}$ column compare an input image $x \in \{C, C'\}$ to a sanitized image $\hat{C} \in \{\hat{C}_{noise}, \hat{C}_{SUDS}\}$, i.e., $\hat{C}_{MSE} = MSE(x, \hat{C})$ and $\hat{C}_{PSNR} = PSNR(x, \hat{C})$. A small MSE value indicates minimal changes between the input and the output image, and a high PSNR value indicates that the signal outweighs the noise in the image. SUDS is, therefore, able to reproduce input images with a high quality regardless of whether the image is a container or cover. The S' column evaluates the reveal functions of each hide method before the container has been sanitized. The MSE and PSNR values of this column compare to the revealed secret prior to sanitization to the intended secret, i.e., $S'_{MSE} = MSE(S, S')$ and $S'_{PSNR} = PSNR(S, S')$. As shown by the small MSE and high PSNR values, each of the reveal functions is effectively able to derive the intended secret such that $S \approx S'$. In the $\hat{S}$ column, however, the intended secret is not discernible from a sanitized image. The MSE and PSNR values for this column compare the intended secret to the secret revealed after sanitization, i.e., $\hat{S}_{MSE} = MSE(S, \hat{S})$ and $\hat{S}_{PSNR} = PSNR(S, \hat{S})$. The high MSE values and low PSNR values confirm that the intended secret is not obtainable through the reveal functions after sanitization. Thus, SUDS is effectively able to sanitize steganographic images, and **RQ1 is positive**.

5.2 Performance Comparison to Noise Baseline

In addition to SUDS, we also test a Gaussian noise baseline, as described in **RQ2**. Table 3 shows the image quality metric performance for the noise baseline sanitization technique. As indicated by the bolded value in the $\hat{S}$ column, Gaussian noise is only effective at sanitizing secrets hidden with UDH, and is therefore not a robust solution. This is further validated by the middle group in Figure 3, which shows the image results of using noise to sanitize. In the $\hat{S}$ column, we can clearly see the secrets for both LSB and DDH. As a result, SUDS outperforms Gaussian noise for sanitization, and **RQ2 is positive**.

Another benefit to SUDS compared to noise lies in its ability to get closer to the original cover. When noise is added as a sanitization technique, the image quality is degraded and the image is pushed further from the actual rendering of the original image. With SUDS, however, as it is trained to reconstruct cover images, the resulting sanitized image gets closer to the original value. In this way, SUDS works to "improve" an image's quality, as shown by the $\hat{C} - C$ for each respective method in Figure 3.

Table 3: Image metrics for SUDS and noise sanitization methods (Section 5.1). Bolded $\hat{S}$ values mean successful sanitization.

			$\hat{C}$	S'	$\hat{S}$
SUDS	Clean	MSE	0.28	-	-
		PSNR	53.77	-	-
	LSB	MSE	0.45	0.09	**62.72**
		PSNR	51.72	58.46	30.17
	DDH	MSE	0.33	0.49	**73.45**
		PSNR	53.03	51.73	29.47
	UDH	MSE	0.38	0.63	**88.54**
		PSNR	52.36	50.23	28.66
Gaussian	Clean	MSE	0.09	-	-
		PSNR	58.82	-	-
	LSB	MSE	0.09	0.09	30.04
		PSNR	58.49	58.46	33.37
	DDH	MSE	0.09	0.49	38.1
		PSNR	58.5	51.76	32.33
	UDH	MSE	0.12	0.63	**57.58**
		PSNR	57.35	50.26	30.53

5.3 *Effect of Latent Dimension Size*

SUDS also works with a variety of feature space sizes. For **RQ3**, we train SUDS on different feature sizes n to see if and when SUDS is no longer able to sanitize a steganographic image. As shown in Figure 4, SUDS retains its ability to sanitize a secret, as indicated by the high green-x line for each feature size. The sanitization performance peaks between $n = 64$ and $n = 32$. While the sanitization performance is latent-size agnostic, SUDS's ability to reconstruct images starts to deteriorate as n decreases, as shown by the high MSE values of the green-triangle line and the low PSNR values of the blue-triangle line for $n = 4$ and $n = 2$. This result makes sense; as the number of features decreases, the amount of information passed to the decoder also decreases, making it difficult to accurately reconstruct the original image. These results show that **RQ3 is positive**.

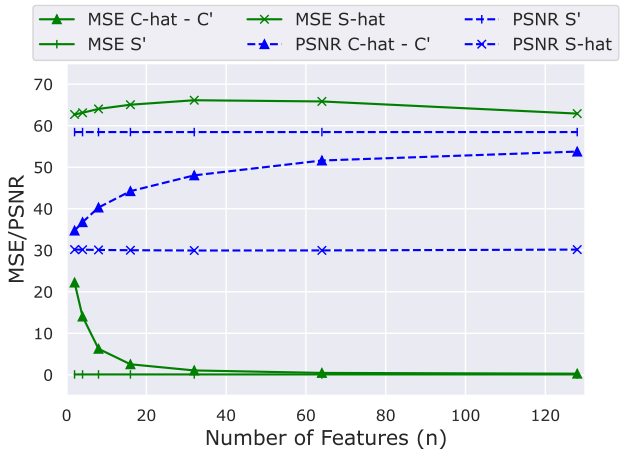

Figure 4: MSE and PSNR values for SUDS trained on varying feature sizes (n). SUDS is able to sanitize at each feature size as indicated by the top green-x line, but its reconstruction ability deteriorates as feature size decreases as shown by $n = 4$ and $n = 2$ on the green-triangle line.

5.4 *Evaluation of Detection Capabilities*

In addition to analyzing the impact of feature size on SUDS performance, we also evaluate whether SUDS can be used as a detection mechanism by analyzing where images get mapped to in the feature space. From the results shown in Figure 5, each image, regardless of the presence of a secret, is mapped to relatively the same distribution per feature. As there is no distinction between covers and containers as well as hiding methods in the feature space, SUDS cannot be used for a detection mechanism based on mean and standard deviation metrics. Thus, **RQ4 is negative**.

5.5 *Scalability to RGB Images*

SUDS is able to sanitize RGB images as well as grayscale images, as indicated by the right-hand side of Figure 3. Once again, the intended secret S is not discernible from $\hat{S}$. Image reconstruction capabilities, however, decline with RGB images, as shown by $\hat{C}$. We believe that this performance can be increased by additional hyperparameter tuning, architecture tuning, and increased training epochs. From these results, **RQ5 is positive**.

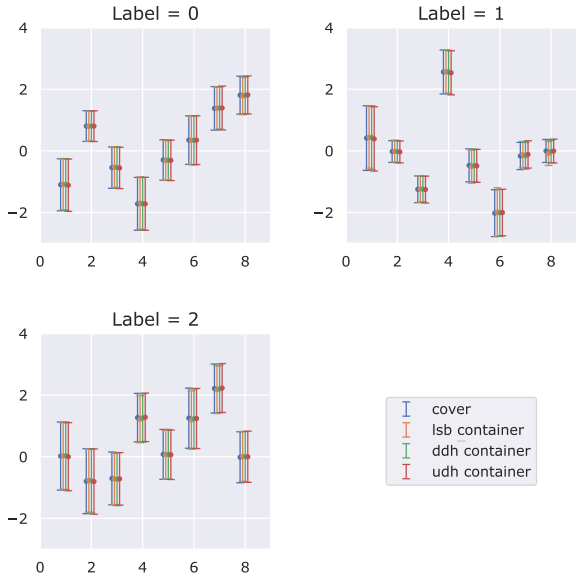

Figure 5: The mean μ and standard deviation σ values for images of a particular label (label $\in \{0, 1, 2, ..., 9\}$) mapped to a latent space representation (z) from SUDS. The mean is represented by the center dot of each line, and the standard deviation is represented by the error bars of each line. From these results, the input images are not distinguishable from the latent space.

5.6 *Summary*

In summary, **RQ1** (ability to sanitize), **RQ2** (comparison to noise), **RQ3** (flexibility), and **RQ5** (scalability) are all positive while **RQ4** (detection) is negative.

6 Example Application: Data Poisoning

To demonstrate a potential use case, we apply SUDS to a case study involving data poisoning. Data poisoning is an adversarial machine learning attack where a model is perturbed during training. This can be caused by methods such as data injection and most commonly—data manipulation. In data manipulation, an attacker has access to the data used for training, and they can alter, add, or remove training data as well as data labels to create backdoors in a model. Common forms of data manipulation include pattern injection, single-pixel modifications, and steganography, which are visibly hard to detect—especially in the case of steganography. Although there are many methods to manipulate data for data poisoning, this case study highlights the use of SUDS to mitigate a data poisoning attack that utilizes steganographic perturbations.

Overview. Data poisoning usually occurs in three phases: preparation, training, and attack. Each of these phases is described in more detail below.

1. **Preparation**: The data perturbation method is created, and the data to be used during training is modified using this technique.
2. **Training**: The model is trained using the manipulated training data.
3. **Attack**: The poisoned model is deployed. The attackers can then craft an adversarial example to alter the intended behavior of the poisoned model.

For this case study, the MNIST training dataset of 60000 images is prepared using the deep dependent hiding (DDH) technique to perturb images. Using DDH, 40% of the training data are changed to a container image, where the secret is randomly chosen with replacement from the input set. The label of this container image is then modified to the label of the randomly chosen secret hidden within it. A classifier is then trained using this manipulated data to classify the digit shown in the picture. If the data poisoning scheme is successful, an attacker should be able to direct the prediction of the trained network by selecting the desired classification as the secret of the container image input.

This data poisoning technique is evaluated using 10000 test images, where 50% have been changed to container images and the remaining 50% are kept clean. The data poisoning technique is then tested again by incorporating SUDS between the client and the trained network. In this test, the data distribution is identical to the previous test, but all test data is sanitized prior to being classified with the trained network.

Result. Table 4 shows the classification accuracy during testing of both clean and container images with and without the use of SUDS. The accuracy associated with the container images indicates the classification of the intended secret to what was predicted by the classifier—was the attacker successful in manipulating the classifier to the intended secret? For instance, if an attacker hid a secret image of label 6 in a cover image with a label of 4, the model would be accurate if it predicted the secret label of 6. By using SUDS between the client and the trained model, the resistance of the classifier to an attack increases by $1375\% = \frac{(100-0.56)-(100-93.26)}{100-93.26} * 100$, while the performance for clean images only drops by 1%. SUDS, therefore, is successfully able to protect the classification system from bad actors.

Table 4: Classification accuracy of clean and poisoned images, both with using SUDS to protect against poisoned images and without using SUDS (no SUDS).

	Accuracy	
Image Type	no SUDS	SUDS
clean (5000)	98.03 %	97.18%
containers (5000)	93.26%	0.56 %

7 Related Works

While SUDS is the first sanitization framework that can mitigate the effects of traditional, dependent deep, and universal deep hiding to the best of the authors' knowledge, there are similar works which use VAEs for protection. In [24], the authors show the promise of using a variational autoencoder (VAE) to mitigate the transfer of PowerShell scripts hidden directly in images with Invoke-PSImage [24], a publicly available hiding tool that uses least significant bit hiding[3]. In this previous work, the sanitizer is evaluated on whether the resulting steganographic image is detected by StegExpose, an open-source detection tool for the least significant bit hiding methods. StegExpose, though, is not a robust detection tool. Additionally, even small perturbations in scripts hidden in images can cause a script to become non-executable. As images are more robust to perturbations, evaluating sanitization with image secrets provides better evaluation criteria for this approach. Additionally, while a sanitizer may be able to sanitize steganography resulting from one hiding technique, this does not mean that it will be effective in sanitizing steganography with other hiding techniques as each method and signature is unique.

A VAE has also been used to detect and repair adversarial perturbations, which are modifications to inputs to cause misclassification. In [10], the authors implement Magnet, which uses outlier detection in the latent space to determine if an image is adversarial. As adversarial techniques add noise, resulting images are mapped to different parts of the latent space. During the repair phase, the outlier latent representations are perturbed in the direction up the gradient to get closer to the distribution of the actual training data, working to remove the added noise of the adversarial image. While steganography is similar to an adversarial image, the "noise" of a steganographic image, or the message, is bounded, i.e., the message must be retrievable and make sense to the recipient. The bounded nature of steganography results in images that are mapped to the same distribution in the latent space (see Section 5.4) and, therefore, are not applicable to the Magnet approach.

[3] **Invoke-PSImage:** https://github.com/peewpw/Invoke-PSImage

8 Conclusions

In this work, we demonstrate a sanitization model SUDS, which is able to sanitize steganographic images from untrained steganographic techniques (traditional, dependent deep, and universal deep) with more reliability than baseline sanitization methods. Whereas SUDS sanitizes secrets for all hiding techniques used in this work, the baseline noise sanitization technique only works 33% of the time against these hiding techniques. In addition to performing more reliably compared to noise, SUDS also has the added benefit of producing a sanitized image that is closer to that of the original cover used to make a container, while noise degrades an image in order to render the hidden secret incomprehensible. This allows SUDS to be applied to many interesting case studies, such as protection against data poisoning attacks, increasing resistance of a poisoned classifier against attacks by 1375%. In the future, we would like to extend SUDS to different steganographic mediums, such as videos, binaries, and time series data, and test this approach against other representation learning models such as a generative adversarial network or diffusion models. As such, this research lays the groundwork for future exploration in comprehensive steganography protection systems.

Acknowledgements

This paper was supported in part by a fellowship award under contract FA9550-21-F-0003 through the National Defense Science and Engineering Graduate (NDSEG) Fellowship Program, sponsored by the Air Force Research Laboratory (AFRL), the Office of Naval Research (ONR), and the Army Research Office (ARO). The material presented in this paper is based upon work supported by the National Science Foundation (NSF) through grant numbers 2220426 and 2220401, the Defense Advanced Research Projects Agency (DARPA) under contract number FA8750-23-C-0518, and the Air Force Office of Scientific Research (AFOSR) under contract number FA9550-22-1-0019 and FA9550-23-1-0135. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of AFOSR, DARPA, or NSF.

References

[1] Shumeet Baluja, 'Hiding images in plain sight: Deep steganography', *Advances in neural information processing systems*, **30**, (2017). https://papers.nips.cc/paper/2017/file/838e8afb1ca34354ac209f53d90c3a43-Paper.pdf.

[2] Rainer Bohme, *Advanced statistical steganalysis*, Springer Publishing Company, Incorporated, 2010. https://dl.acm.org/doi/abs/10.5555/1869956.

[3] Jessica Fridrich and Jan Kodovsky, 'Rich models for steganalysis of digital images', *IEEE Transactions on information Forensics and Security*, **7**(3), 868–882, (2012). https://ieeexplore.ieee.org/document/6197267.

[4] Gokhan Gul and Fatih Kurugollu, 'A new methodology in steganalysis: breaking highly undetectable steganography (hugo)', in *International Workshop on Information Hiding*, pp. 71–84. Springer, (2011). https://link.springer.com/chapter/10.1007/978-3-642-24178-9_6.

[5] Neil F. Johnson and Sushil Jajodia, 'Exploring steganography: Seeing the unseen', *Computer*, **31**(2), 26–34, (1998). https://ieeexplore.ieee.org/document/4655281.

[6] Kaspersky. Kaspersky lab identifies worrying trend in hackers using steganography, 2017. https://usa.kaspersky.com/about/press-releases/2017_kaspersky-lab-identifies-worrying-trend-in-hackers-using-steganography.

[7] Gary Kessler, 'An overview of steganography for the computer forensics examiner', *Forensic Science Communications*, **6**, (01 2004). http://profs.scienze.univr.it/~giaco/download/Watermarking-Obfuscation/Steganography\%20for\%20the\%20Computer\%20Forensics\%20Examiner.pdf.

[8] C. Kurak and J. McHugh, 'A cautionary note on image downgrading', in *[1992] Proceedings Eighth Annual Computer Security Application Conference*, pp. 153–159, (1992). https://ieeexplore.ieee.org/document/228224.

[9] Xiyang Luo, Ruohan Zhan, Huiwen Chang, Feng Yang, and Peyman Milanfar, 'Distortion agnostic deep watermarking', in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13548–13557, (2020). https://ieeexplore.ieee.org/document/9157561.

[10] Dongyu Meng and Hao Chen, 'Magnet: a two-pronged defense against adversarial examples', in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pp. 135–147, (2017).

[11] Reza Montasari, *Artificial Intelligence and the Internet of Things Forensics in a National Security Context*, 57–80, Springer International Publishing, Cham, 2023. https://link.springer.com/chapter/10.1007/978-3-031-21920-7_4.

[12] Tomáš Pevnỳ, Patrick Bas, and Jessica Fridrich, 'Steganalysis by subtractive pixel adjacency matrix', in *Proceedings of the 11th ACM workshop on Multimedia and security*, pp. 75–84, (2009). https://ieeexplore.ieee.org/document/5437325.

[13] Yun Q Shi, Patchara Sutthiwan, and Licong Chen, 'Textural features for steganalysis', in *International workshop on information hiding*, pp. 63–77. Springer, (2013). https://ieeexplore.ieee.org/document/6197267.

[14] Guillermo Suarez-Tangil, Juan E Tapiador, and Pedro Peris-Lopez, 'Stegomalware: Playing hide and seek with malicious components in smartphone apps', in *International conference on information security and cryptology*, pp. 496–515. Springer, (2015). https://link.springer.com/chapter/10.1007/978-3-319-16745-9_27.

[15] Weixuan Tang, Bin Li, Mauro Barni, Jin Li, and Jiwu Huang, 'An automatic cost learning framework for image steganography using deep reinforcement learning', *IEEE Transactions on Information Forensics and Security*, **16**, 952–967, (2020). https://ieeexplore.ieee.org/document/9205850.

[16] Weixuan Tang, Shunquan Tan, Bin Li, and Jiwu Huang, 'Automatic steganographic distortion learning using a generative adversarial network', *IEEE Signal Processing Letters*, **24**(10), 1547–1551, (2017). https://ieeexplore.ieee.org/document/8017430.

[17] Denis Volkhonskiy, Ivan Nazarov, and Evgeny Burnaev, 'Steganographic generative adversarial networks', in *Twelfth international conference on machine vision (ICMV 2019)*, volume 11433, pp. 991–1005. SPIE, (2020). https://doi.org/10.1117/12.2559429.

[18] Zihan Wang, Neng Gao, Xin Wang, Xuexin Qu, and Linghui Li, 'Sstegan: Self-learning steganography based on generative adversarial networks', in *International Conference on Neural Information Processing*, pp. 253–264. Springer, (2018). https://link.springer.com/chapter/10.1007/978-3-030-04179-3_22.

[19] Eric Wengrowski and Kristin Dana, 'Light field messaging with deep photographic steganography', in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1515–1524, (2019). https://ieeexplore.ieee.org/document/8954203.

[20] Haibin Wu, Fengyong Li, Xinpeng Zhang, and Kui Wu, 'Gan-based steganography with the concatenation of multiple feature maps', in *International Workshop on Digital Watermarking*, pp. 3–17. Springer, (2020). https://link.springer.com/chapter/10.1007/978-3-030-43575-2_1.

[21] Jianhua Yang, Danyang Ruan, Jiwu Huang, Xiangui Kang, and Yun-Qing Shi, 'An embedding cost learning framework using gan', *IEEE Transactions on Information Forensics and Security*, **15**, 839–851, (2019). https://ieeexplore.ieee.org/abstract/document/8735922.

[22] Chaoning Zhang, Philipp Benz, Adil Karjauv, Geng Sun, and In So Kweon, 'Udh: Universal deep hiding for steganography, watermarking, and light field messaging', *Advances in Neural Information Processing Systems*, **33**, 10223–10234, (2020). https://proceedings.neurips.cc/paper/2020/file/73d02e4344f71a0b0d51a925246990e7-Paper.pdf.

[23] Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei, 'Hidden: Hiding data with deep networks', in *Proceedings of the European conference on computer vision (ECCV)*, pp. 657–672, (2018). https://link.springer.com/chapter/10.1007/978-3-030-01267-0_40.

[24] Marco Zuppelli, Giuseppe Manco, Luca Caviglione, and Massimo Guarascio, 'Sanitization of images containing stegomalware via machine learning approaches', in *ITASEC*, (2021). https://ceur-ws.org/Vol-2940/paper31.pdf.