LightShed: Defeating Perturbation-based Image Copyright Protections

Hanna Foerster
University of Cambridge[*]

Sasha Behrouzi
Technical University of Darmstadt

Phillip Rieger
Technical University of Darmstadt

Murtuza Jadliwala
University of Texas at San Antonio

Ahmad-Reza Sadeghi
Technical University of Darmstadt

Abstract

Recently, image generation models like Stable Diffusion have gained significant popularity due to their remarkable achievements. However, their widespread use has raised concerns about potential misuse, particularly regarding acquiring training data, including using copyright-protected material. Various schemes have been proposed to address these concerns by introducing inconspicuous perturbations (poisons) to prevent models from utilizing these samples for training.

We present LightShed, a generalizable depoisoning attack that effectively identifies poisoned images and removes adversarial perturbations, showing the limitations of current protection schemes. LightShed exploits the wide availability of these protection schemes to generate poisoned examples and models their characteristics. The fingerprints derived from this process enable LightShed to efficiently extract and neutralize the perturbation from a protected image. We demonstrate the effectiveness of LightShed against several popular perturbation-based image protection schemes, including NightShade, recently presented at IEEE S&P 2024, and Glaze, published at Usenix Security 2023. Our results show that LightShed can accurately identify poisoned samples, achieving a TPR of 99.98% and TNR of 100% on detecting NightShade and effectively depoisoning them. We show that LightShed generalizes across perturbation techniques, enabling a single model to recognize poisoned images.

1 Introduction

Recently, text-to-image models [7, 18, 20, 35], particularly diffusion models [5, 21, 30], have gained significant popularity due to their ability to generate diverse, realistic-looking images from just a short prompt. However, as these models are trained on vast datasets scraped from various sources, there is growing concern that artists' works, including photographs, paintings, and other creative pieces, may be used in training without their consent. This has led to concerns that the models could (unintentionally) imitate or replicate an artist's unique style or work without proper attribution. Consequently, many creators, especially artists, seek ways to protect their work from being exploited by these emerging technologies [4, 31].

Image Protection Schemes against Generative Models. In response to growing concerns that images presented by artists might be misused for training image generation models, various protection strategies have been developed to prevent the unauthorized use of these images in model training. Most of these strategies focus on hindering the fine-tuning of pre-trained image generators [11, 13, 22], particularly to prevent the imitation of specific artistic styles. Among these, *Glaze* [22] has gained significant attention in public media [4, 31]. More recently, Shan *et al.* introduced *NightShade*, a more advanced approach that goes beyond disrupting fine-tuning processes by targeting the entire training pipeline of image generators. Unlike passive fine-tuning protections such as Glaze, which are designed to prevent the model from using the protected images for training, NightShade actively interferes with the training process by mapping the target concept to unrelated concepts or styles. This injected confusion in the feature space aims to prevent the model from learning essential functionalities despite the presence of ample clean data that would typically allow learning the correct behaviors [23], highlighting the effectiveness of such attacks.

Attacks on Image Protections. The need for artists to protect their work in combination with the claims made by proposed schemes such as *Glaze* and *NightShade* to protect images reliably has drawn significant attention from the research community. Existing depoisoning attack techniques have concentrated on mitigating fine-tuning protection techniques [6, 14–16]. They can be categorized into two main classes: *targeted* approaches that focus on identifying poisoned images and apply modifications on only these, and *untargeted* approaches that apply modifications on all images. Existing targeted approaches have faced accuracy and generalization challenges, whereas untargeted approaches can affect clean images and face significant scalability challenges. Un-

[*]Research was performed during an internship at Technical University of Darmstadt.

targeted approaches are often developed for smaller datasets for style mimicry consisting of images of particular artists. This means that they are impractical for larger datasets. An example of a recently proposed attack to overcome fine-tuning protections in Glaze [22] is proposed by Hönig *et al.* [6]. However, the attack does not extend to schemes designed to protect the entire training process, such as NightShade [23], which is tailored for broader and larger datasets. Thus, NightShade provides scalability and effectiveness in full-training scenarios where Glaze may be less effective. The approach of Hönig *et al.* leverages the smaller datasets typical of fine-tuning scenarios, but faces scalability challenges for datasets consisting of millions of samples. We will elaborate on these aspects in sections Sect. 5.6 and Sect. 7.2. In this paper, we design a more general, robust, and scalable depoisoning attack that automatically detects and neutralizes adversarial poisoning.

Goals and Contributions. To show the insufficiency of existing approaches and the need for more robust image protection techniques, we introduce *LightShed*, a comprehensive and general depoisoning attack that effectively detects adversarial perturbations created by image-protection techniques in datasets and removes poisoning from images. LightShed includes a Poisoning Reconstruction component, which models the characteristics of perturbations and extracts the fingerprint of any protection scheme from the analyzed image. Using poisoned images crafted with protection tools as training samples, LightShed trains an autoencoder to identify subtle poisoning perturbations and reconstruct them accurately. We identify previously unknown poisoned images by analyzing the entropy of the reconstructed perturbation, effectively distinguishing them from clean images. LightShed then restores the original, unpoisoned image by removing residual artifacts by subtracting the reverse-engineered perturbation, making it effective not only against training prevention tools but also against fine-tuning protection techniques.

We note that LightShed can also be used to enhance untargeted attacks [6, 10, 15]. Identifying poisoned images and subtracting the reconstructed perturbations allows a more targeted application of untargeted techniques to only the affected samples. This allows omitting clean samples from unnecessary processing and allows the deployment of existing untargeted techniques at reduced intensities, making LightShed complementary to these approaches (see Sect. 5.6 and Sect. 7.2 for details). Our contributions include:

- We propose LightShed, which effectively identifies and cleans poisoned images by learning the perturbation schemes' individual characteristics, showing the limitations of current image protection techniques. To the best of our knowledge, LightShed is the first attack that considers fine-tuning and training-prevention techniques against diffusion models. Operating independently of specific protection schemes makes it comprehensive and adaptable for future defense strategies (Sect. 4.2).
- We design a novel approach to extract poisoning perturbations by modeling the characteristics of image protection schemes. This enables us to accurately identify the presence of perturbations in given samples and reverse-engineer these perturbations. Being able to recognize the substance of poison it can generalize to other types of poison without training on them. By providing different types of poison for training, LightShed can more accurately learn new types of poisons. This shows its flexibility and applicability for future defense techniques (Sect. 4.3).
- We extensively evaluate LightShed's efficiency and effectiveness, showing its ability to detect and mitigate state-of-the-art poisoning schemes (NightShade, Glaze, MetaCloak, Mist), which disrupt fine-tuning or entire training processes. LightShed accurately identifies poisoned samples, achieving a TPR of 99.98% and TNR of 100% on NightShade and successfully unpoisoning training processes by cleaning the poisoned images (Sect. 5).
- Based on the insights LightShed provides, we analyze requirements for more resilient protection schemes. Our findings include the need to make injected perturbation less structured and predictable, to vary perturbation density in different image regions, and to show a similar structure as noising methods to prevent attacks from modeling and extracting poisons (Sect. 6.3).

2 Background

2.1 Diffusion Models

Diffusion models [5, 30] have recently gained attention as a leading approach in generative deep learning and have overtaken the previous dominance of Generative Adversarial Networks (GANs) [3] in image generation. Utilizing probabilistic generative methods, Diffusion models successively add noise to training input data and recover the input to a clear output image. Diffusion models commonly operate in high-dimensional pixel space, making their training computationally intensive. Latent diffusion models [21] addressed this challenge by replacing pixel space with the latent space of pretrained variational autoencoders and reducing input data's dimensionality, leading to faster and more efficient training and inference.

However, training the encoder and the UNet from scratch remains expensive. To mitigate these costs, models often undergo fine-tuning with new images, which allows for refinement and adjustments without the need for complete retraining. Yet, this fine-tuning process carries the risk of data poisoning by possibly including poisoned data.

2.2 Copyright Protection Schemes

With the increasing popularity and wide accessibility of image generators, different approaches have been proposed to

protect artists' work against being utilized for training image generators and prevent the artists' work from being imitated. In the following, we describe four different approaches that we exemplary consider in this work [11, 13, 22, 23]. They cover different use cases, preventing the style of paintings from being learned but also disturbing the image generators from learning the concept of an object, e.g., preventing the generator from learning the concept of a dog [23]. All these approaches take a clean image as input and modify it to disturb the training process of the diffusion models.

Fine-tuning protection approaches focus on scenarios where a pre-trained diffusion model is trained to imitate a specific painting style or to personalize the model, such as enhancing its ability to generate images of a targeted individual. Notably, fine-tuning typically involves a small number of images, in contrast to the millions of samples used for fully training a model. These images are often sourced from a single data source, such as an artist's website. Therefore, if the image owner employs protection techniques, it is likely that all images used in the fine-tuning process will be protected. This allows the protection to specifically target and prevent the utilization of these protected images.

Mist uses a checkpoint of the targeted diffusion model to generate an adversarial example for the current sample. The attack focuses on the encoder of the diffusion model. A semantic loss and a textual loss are combined to generate adversarial examples. While the semantic loss focuses on maximizing the training loss of the diffusion model, the textual loss manipulates the encoded representation of the image to maximize the distance to the original image. An adversarial pattern is crafted via gradient descent and overlayed on the source image to maximize the distance between the original and modified image encodings [11].
Glaze uses style transfer to craft the perturbation pattern added to the image. First, a style-transferred version of the input image is created using the chosen target style. The poisoning pattern is then generated by solving an optimization problem that minimizes the distance between the modified image and the style-transferred image while constraining the lengths of the perturbation vector to be smaller than a predefined perceptual perturbation budget [22].
MetaCloak also crafts an adversarial example for the given input image and solves an optimization problem that maximizes the loss of the diffusion model for the protected image. However, compared to Mist, MetaCloak considers different states of the targeted model. Iterating over the individual checkpoints, it optimizes the input image to maximize the loss for the current checkpoint of the diffusion model using Stochastic Gradient Descent before training the diffusion model with the manipulated images. MetaCloak uses projection to ensure that the current version of the image shows at most a predefined distance [13].

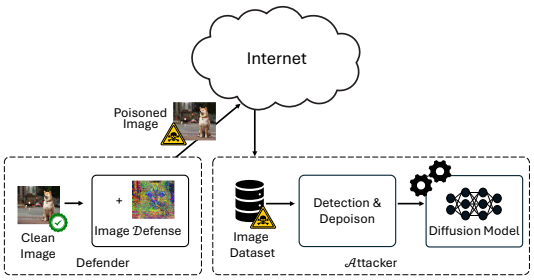


Figure 1: Overview of the considered scenario, where the defender uses the image defense $\mathcal{D}$, denoted in the image as Image $\mathcal{D}$efense, to protect the image and the attacker $\mathcal{A}$ uses a detection and denoising algorithm to train a diffusion model.

In contrast to fine-tuning protection techniques, which can focus on preventing the utilization of protected samples, approaches targeting the full training of a diffusion model must be capable of preventing them from learning even fundamental functionalities related to the targeted concepts. However, given the large amounts of training data involved from various sources, not all samples can be poisoned. Therefore, training prevention methods must actively disrupt the training process and neutralize large numbers of clean data.
NightShade modifies the input image but also the respective textual caption to distract the model. Considering given pairs of images and captions that shall be protected, NightShade selects a target concept and generates a number of images for it. Then, it crafts a poisoning pattern that minimizes the distances between the encoding of the input image and the generated images for the other concept. Notably, the aim of NightShade is not only to prevent the model from learning protected images but to disrupt the training as a whole on a dataset that contains protected images [23].

An important aspect of image protection mechanisms is their application before an image is publicly uploaded. After publishing, the owner loses control, as potential attackers can download and store it locally. If the applied protection method is later compromised, the artist may apply an updated defense to the images and re-upload them. However, the adversary can still exploit the original, unprotected versions it previously downloaded, meaning the artist cannot retroactively enhance the security of those images stored by the adversary [6]. We will elaborate on this in Sect. 6.1.

3 Problem Setting

3.1 System and Adversary Setting

In the following, we consider a system where a party, referred to as the defender, uploads images to a public platform. To protect the copyright/style of the images and prevent them from being used for training or fine-tuning an image genera-

tion model, the defender first applies a defense algorithm $\mathcal{D}$ to the images. For instance, the defender could be an artist seeking to protect their work, whether it's paintings, photographs, or other types of images. The attacker, on the other hand, aims to train or fine-tune an image generation model using images downloaded from the internet, including those of the defender, without obtaining permission from the image owners. To circumvent image protection techniques, the attacker employs a depoisoning technique such as LightShed. In cases where the attacker is fine-tuning a model, it is likely all training images are poisoned, as parties using a defense like $\mathcal{D}$ would likely apply it to all their images.

When training an image generator, the dataset typically includes images from various sources, increasing the likelihood that non-poisoned images are available for each concept. This scenario allows the attacker to exclude the poisoned images rather than needing to depoison them. However, in our approach, we do not restrict the ratio of poisoned samples from the total dataset. This system is illustrated in Fig. 1.

Given the widespread use of image defense algorithms, the adversary suspects that some or all of the images may have been poisoned using a defense technique. While the specific technique used is unknown to the adversary, they are aware of commonly employed image protection schemes, which include the actual defense technique $\mathcal{D}$ used by the defender. To prevent $\mathcal{D}$ from negatively impacting the training process, the adversary must identify the protected images within the training dataset and remove the poisoned perturbations[1].

We make only a few assumptions about the adversary. Due to the inconspicuousness of the perturbations introduced by $\mathcal{D}$, the adversary cannot easily identify the poisoned images through simple methods like visual inspection. Additionally, the adversary does not know which specific defense technique is used or even if a defense has been applied at all. We assume only that the adversary is aware of the existence of poisoning techniques and has the capability to utilize these techniques, such as through pre-compiled binary applications provided by the designers of these methods [24, 25] or via web services where artists can upload and retrieve protected images [25].

3.2 Requirements and Challenges

Practical attacks on image protection schemes posing real threats to image owners must meet several key requirements:
R1 – Defeat Attacks on Training: The attack must effectively detect and filter out poisoned images that are capable of disrupting the training process of diffusion models, i.e., achieve a high True-Positive-Rate and high True-Negative-Rate (see Sect. 5.2). This includes images poisoned to prevent fine-tuning [11–13, 22, 32], as well as those subjected to concept-based poisoning [23]. Further, the attack must be able to effectively recover the original clean images. Therefore, for detected images, the depoisoned version must show high similarities with the original clean versions.

R2 – General Applicability: Given that Deep Neural Networks (DNNs) have a vast number of parameters, there are numerous potential strategies to disrupt training. Therefore, an effective attack must be general enough to counter all state-of-the-art defenses.

R3 – No Disruption of the Training Process: The attacker's goal is to train a diffusion-based image generation model using images from the internet without any interruptions. Hence, the defense mechanism should not negatively impact the training process, such as by excluding a significant number of clean samples, which could otherwise hinder model performance.

In recent years, various sophisticated schemes have been developed to protect artistic style and content within images from being copied/replicated in an unauthorized fashion by using image generators. These protection methods, which typically involve injecting robust, poisoned perturbations into images, present several challenges for attacks aiming to bypass these defenses:

C1 – Inconspicuous Perturbations: A critical requirement for image perturbations is that they must not affect human perception of the images. Consequently, protection approaches introduce only subtle modifications, making detecting these perturbations highly challenging. An effective attack must, therefore, address the challenge of reliably detecting such inconspicuous perturbations in images, especially when the original, unmanipulated version of the image is unknown.

C2 – Structured Patterns: While previous work has primarily focused on developing and mitigating noise-like adversarial examples [11], recent protection approaches for image generators inject perturbations that are aligned with the natural structure of the images [22, 23]. As a result, traditional noise reduction techniques are ineffective. A challenge for attacks is, therefore, to extract and remove these structured perturbations from the images.

C3 – Diversity of Protection Tools: Image owners have access to a wide range of protection schemes, and an adversary seeking to detect and remove poisoned perturbations cannot know in advance which specific perturbation has been applied. This presents the challenge of developing an attack that is effective across a variety of protection tools, capable of dynamically determining and countering the characteristics of the applied defense.

C4 – Absence of a Clean Dataset: Many approaches rely on a clean dataset either for applying perturbations or as a comparison standard. However, obtaining a clean dataset is challenging since images sourced from the internet may already contain poisoning perturbations. Thus, a mitigation strategy must address the challenge of effectively leveraging a potentially poisoned dataset during the training process.

[1] For completeness, when only a small subset of images is poisoned, it may also be convenient to exclude these samples rather than depoison them.

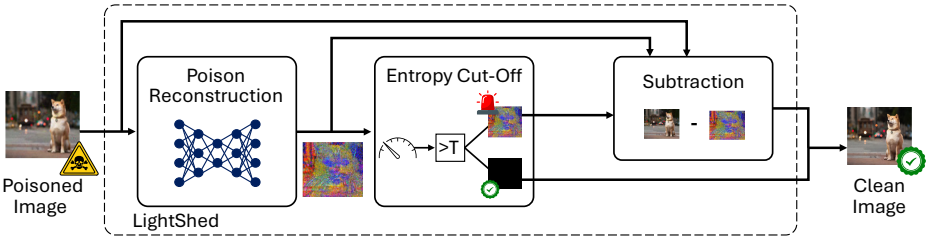


Figure 2: Overview of the detection and healing process of LightShed. If a poisoned sample is detected by the reconstructed poison, subtraction is applied, while an empty reconstructed perturbation indicates a clean input.

4 LightShed

To show the insufficiency of existing copyright protection tools and the need for more robust perturbations to protect copyright owners against being imitated by diffusion models, we propose LightShed that effectively detects the copyrighted images and also erases the adversarial perturbations.

4.1 Motivation

Poisoned perturbations are designed to disrupt training processes while remaining imperceptible to humans through a specific optimization process, resulting in non-random patterns. Our approach focuses solely on reconstructing the poison rather than the complete image, leveraging two key advantages: *(i)* perturbations have significantly lower entropy than full images, making them easier to learn, and *(ii)* the entropy difference between clean and partially reconstructed perturbations enables reliable detection even with imperfect reconstruction. Autoencoders, being lightweight and efficient, excel at learning and reconstructing the perturbations, providing both detection capabilities and visual insights into the original perturbation's nature. We enhanced LightShed with attention layers to leverage Transformers' global context understanding, enabling better detection and isolation of diverse perturbation patterns. This proved superior to alternatives like traditional DNNs (which tend to overfit to specific poisoning schemes), resource-intensive Transformers, and less stable and precise GANs. Our method enhances interpretability and generalization, addressing critical challenges in machine learning, while maintaining the option to apply techniques like noisy-upscaling to refine reconstructions if needed.

4.2 High-Level Overview

An overview of LightShed is shown in Fig. 2. The inference process is formalized in Alg. 1. LightShed extracts adversarial perturbations from images based on the rationale that they are distinguishable from the main content, though not easily visible to human perception. By training a DNN with samples from the perturbation scheme, LightShed can model their characteristics, allowing the DNN to detect subtle differences in the image regions and identify perturbations, even when they are imperceptible to humans, addressing challenges C1 and C2. LightShed is composed of three components.

The Poisoning Reconstruction component models the characteristics of the protection scheme to reverse-engineer and extract the image's poisoning perturbation. In this process, an autoencoder is employed not to reduce noise as traditionally used but to isolate and remove the main content of the image, thereby extracting the perturbation pattern. For clean images, this process results in an empty output (see Fig. 2), as there is no perturbation to extract, whereas, for poisoned images, the generated output approximates the perturbation.

The Entropy Cut-Off component analyzes the extracted perturbation to determine if the original image was poisoned. Since clean images produce an empty perturbation, the entropy of the extracted output can be measured to verify whether the image was poisoned.

The Subtraction component is applied only to poisoned images. By subtracting the extracted perturbation from the original image, the component reconstructs an approximation of the clean version of the image.

4.3 Poison Reconstruction

LightShed utilizes an autoencoder to model the characteristics of the poisoning. The underlying assumption is that due to the popularity and wide availability of the protection tools (see Sect. 6.2), the attacker can use them for crafting training samples. Thus, an image dataset is obtained before applying the targeted copyright protection scheme on the images, as visualized in Fig. 3. By including samples generated from different defenses, LightShed learns to detect the characteristics for each of them, addressing challenge C3.

To be able to train the autoencoder to optimally reconstruct either the poison or an empty image, we design the loss function $\mathcal{L}$ as a combination of several metrics, which measure the difference between the ground truth perturbation P_{actual} and the reconstructed perturbation $P_{\text{reconstr.}}$. Given the Structural Similarity Index Measure (SSIM) loss $\mathcal{L}_{\text{SSIM}}$, the regularization loss $\mathcal{L}_{\text{Regu}}$, the reconstruction loss $\mathcal{L}_{\text{Reco}}$, the fine-tuning loss $\mathcal{L}_{\text{FineTune}}$, and the fine-tune boundary τ the combined loss $\mathcal{L}$ in epoch e is given by:

$$\mathcal{L} = \begin{cases} 0.7 \cdot \mathcal{L}_{\text{SSIM}} + 0.2 \cdot \mathcal{L}_{\text{Regu}} + 0.1 \cdot \mathcal{L}_{\text{Reco}} & \text{if } e < \tau \\ 0.7 \cdot \mathcal{L}_{\text{SSIM}} + 0.2 \cdot \mathcal{L}_{\text{Regu}} + 0.1 \cdot \mathcal{L}_{\text{Reco}} + 0.3 \cdot \mathcal{L}_{\text{FineTune}} & \text{otherwise} \end{cases} \tag{1}$$

The weights were determined through empirical evaluation to balance the different learning objectives. The SSIM loss is weighted highest (0.7) as it is crucial for capturing structural relationships in the perturbations, while the regularization (0.2) and reconstruction losses (0.1) serve as complementary terms to ensure stable training and accurate pixel-wise reconstruction. The fine-tuning loss weight (0.3) was chosen to provide sufficient influence on the entropy-based separation between clean and poisoned images without overwhelming the primary reconstruction objective.

The SSIM loss $\mathcal{L}_{\text{SSIM}}$ [34] (line 12 in Alg. 2) is preferred over traditional reconstruction loss as the primary loss component because it focuses on the structural relationships between neighboring pixels rather than merely comparing individual pixel values and is defined as:

$$\mathcal{L}_{\text{SSIM}} = 1 - \text{SSIM}(\text{P}_{reconstr.}, \text{P}_{actual}) \tag{2}$$

To ensure comprehensive reconstruction quality, we incorporate two additional loss terms. The reconstruction loss $\mathcal{L}_{\text{Reco}}$ provides a direct measure of pixel-wise differences between the reconstructed and actual perturbations:

$$\mathcal{L}_{\text{Reco}} = |(\text{P}_{reconstr.} - \text{P}_{actual})| \tag{3}$$

While the regularization loss $\mathcal{L}_{\text{Regu}}$ specifically penalizes large reconstruction errors by increasing quadratically when differences exceed a threshold ψ:

$$\mathcal{L}_{\text{Regu}} = \max\left(0, ((\text{P}_{reconstr.} - \text{P}_{actual}))^2 - \psi\right) \tag{4}$$

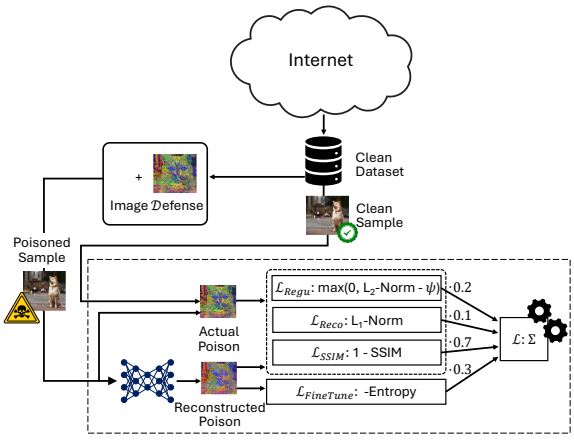


Figure 3: Training Process of LightShed and the calculation of individual loss functions $\mathcal{L}_{\text{Regu}}$, $\mathcal{L}_{\text{Reco}}$, and $\mathcal{L}_{\text{SSIM}}$ based on the reconstructed poison and the actual poison. In later epochs, the reconstructed poison is also used to calculate $\mathcal{L}_{\text{FineTune}}$.

Following the initial training phase ($\tau = 100$ epochs), we introduce an additional fine-tuning loss $\mathcal{L}_{\text{FineTune}}$ to enhance the model's ability to distinguish between clean and poisoned images (see lines 15-22 in Alg. 2). This loss can only be applied in later epochs since earlier adoption will disrupt learning the differentiation between clean and poisoned due to the contrastive training objectives for each in this loss. However, later in training this can help push the entropy distributions of clean and poisoned perturbation reconstructions more distinctly from each other:

$$\mathcal{L}_{\text{FineTune}} = \begin{cases} \text{Entropy}(\text{P}_{reconstr.}) & \text{if I is clean} \\ -\text{Entropy}(\text{P}_{reconstr.}) & \text{otherwise} \end{cases} \tag{5}$$

After the calculation of the individual loss components, they are combined (see line 24 of Alg. 2) and the autoencoder is updated using the Adam optimizer (see line 25 of Alg. 2).

The autoencoder utilized in LightShed is a deep convolutional model leveraging residual and attention blocks to improve its training. The residual blocks, composed of two 3×3 convolutional layers followed by batch normalization and ReLU activation, help to stabilize the training by allowing shortcut connections that preserve the flow of gradients. Compared to standard convolutional layers, attention blocks are incorporated to enable the model to focus on specific regions of the image. These blocks take a feature map from an earlier encoder layer and combine it with a gating signal from a later

Algorithm 1 Inference for Detection and Depoisoning

Input:
A (Trained Autoencoder)
$D_{\text{test}} = \{I^i \mid i = 1, 2, \ldots, k\}$
Output:
Classification results *labels*
(Reconstructed) Clean Images $[I_c^i \mid i = 1, 2, \ldots, k]$
▷ Poison Reconstruction
for each $I \in D_{\text{test}}$ **do**
    $P'(I) \leftarrow A(I)$
    $H(I) \leftarrow \text{Entropy}(P'(I))$
end for
▷ Entropy Cut-Off
$T \leftarrow$ Optimal Threshold from Validation Set
labels $\leftarrow []$
for each $I \in D_{\text{test}}$ **do**
    if $H(I) > T$ **then**
        labels.append(Poisoned)
    else
        labels.append(Clean)
    end if
end for
▷ Subtracting
clean_images $\leftarrow []$
for each $I, label \in (D_{\text{test}}, labels)$ **do**
    if $label == Clean$ **then**
        clean_images.append(I)
    else
        $I_c \leftarrow I - P'(I)$
        clean_images.append(I_c)
    end if
end for

layer. This helps focus the attention on specific features.

The encoder consists of four convolution layers, each followed by batch normalization, ReLU activation, and a residual block. Each layer downsamples the input feature map by a factor of 2. The bottleneck layer consists of a convolutional block that further downsamples the feature maps, followed by two residual blocks. The decoder, which mirrors the encoder's structure, uses transposed convolutions to upsample the feature maps back to their original size. Each decoding stage consists of a residual block followed by a transposed convolution. The input to each decoder layer is the feature map from the corresponding encoder layer, element-wise multiplied by the attention map. This process ensures the model focuses on key regions, preserving crucial spatial features while finding adversarial perturbations.

4.4 Entropy Cut-Off

To determine whether the image contains a poisoned perturbation, we calculate the entropy of the reconstructed poison $P_{reconstr.}$. In the case of the non-poisoned images, the entropy should be close to zero, whereas for the poisoned images, $P_{reconstr.}$ should show a significantly higher entropy. To classify the generated perturbation as black or filled with colored pixels, we compare the measured entropy against the dynamically determined entropy cut-off threshold T. It is determined using a validation set to calculate a ROC curve with the correct labels (clean/poisoned) and the entropy values of each generated poison image. Taking a minimum constraint of 0.1 for the FPR, we choose the threshold value that maximizes the TPR for the validation set.

Algorithm 2 Autoencoder Training (D contains clean (I_{np}) and poisoned (I_p) training images, D_{cor} contains the corresponding clean version of each training image.)

Input:
$D = \{I^i_{\text{np}} \mid i = 1,2,\ldots,n\} \cup \{I^j_p \mid j = 1,2,\ldots,m\}$
$D_{\text{cor}} = \{I^i_{\text{cor}} \mid i = 1,2,\ldots,n\}$
$D_{\text{val}} = \{I^k_{\text{val}} \mid k = 1,2,\ldots,l\}$
$y_{\text{val}} = \{y^k_{\text{val}} \mid k = 1,2,\ldots,l\}$
Output: Trained Autoencoder A, Threshold T
$A \rightarrow Init()$ ▷ Initialize Autoencoder
for epoch $e = 1$ to E **do**
  for each (I, I_{cor}) in (D, D_{cor}) **do**
    $P_{reconstr.} = A(I)$
    $P_{actual} = I - I_{\text{cor}}$
    $\mathcal{L}_{\text{SSIM}} = 1 - \text{SSIM}(P_{reconstr.}, P_{actual})$
    $\mathcal{L}_{\text{Regu}} = \max(0, ((P_{reconstr.} - P_{actual}))^2 - \psi)$
    $\mathcal{L}_{\text{Reco}} = |(P_{reconstr.} - P_{actual})|$
    if $e < \tau$ **then**
      $\mathcal{L}_{\text{FineTune}} = 0$
    else
      if $I \in I_{\text{np}}$ **then** ▷ check if I is clean
        $\mathcal{L}_{\text{FineTune}} \leftarrow \text{Entropy}(P_{reconstr.})$
      else
        $\mathcal{L}_{\text{FineTune}} \leftarrow -\text{Entropy}(P_{reconstr.})$
      end if
    end if
    $\text{Loss} = 0.7 \cdot \mathcal{L}_{\text{SSIM}} + 0.2 \cdot \mathcal{L}_{\text{Regu}} + 0.1 \cdot \mathcal{L}_{\text{Reco}} + 0.3 \cdot \mathcal{L}_{\text{FineTune}}$
    $A \leftarrow A - \nabla \text{Loss}$
  end for
end for
$\text{Eval} = \text{Entropy}(A(D_{\text{val}}))$
$T \leftarrow$ Optimal threshold from $\text{ROC}(y_{\text{val}}, \text{Eval})$

Dataset	# Training Samples	# Test Samples
LAION [8] (clean)	42 000	34 500
NightShade [23] Linf	24 150	10 350
NightShade [23] LPIPS random	15 740	0
NightShade [23]	0	8 950
NightShade [23] LPIPS 0.004	0	11 350
MetaCloak [13]	0	224
Mist [11]	1 655	710
NightShade Binary [24]	11 300	1 255
Glaze Binary [25]	2 170	930

Table 1: Size of the utilized datasets, derived from the LAION-Aesthetics Dataset.

4.5 Poison Subtraction

Reconstructing the poisoning perturbation enables LightShed to efficiently remove the perturbation from the image. To reconstruct the original, unpoisoned image, we subtract the generated perturbation ($P_{reconstr.}$) from the image. This eliminates most of the poison to the degree that the fine-tuning is not poisonously affected, and poison artifacts are eliminated.

5 Evaluation

5.1 Experimental Setup

To measure the effectiveness of LightShed. We apply finetuning to a stable diffusion model similarly to Shan *et al.* [23]. Based on their claim, the poison takes effect on Stable Diffusion XL, Stable Diffusion V2, and DeepFloyd by fine-tuning with a dataset of 50 000 clean and 300 poisoned images. Despite testing various fine-tuning parameters on Stable Diffusion XL, we were unable to replicate the poisoned results. After attempting to contact the authors without success prior to submission, we switched to Stable Diffusion V2 and followed the fine-tuning instructions in Hönig *et al.* [6], using a batch size of 4 and 2 000 epochs. To enable NightShade disruption in the training, we reduced the number of clean samples to 10 000 and increased the number of poisoned samples to 3 000.

All experiments were implemented using PyTorch [17] and the Diffuser library [33] for training diffusion models. The experiments were executed on a server with 2 AMD EPYC 7773x CPUs, 5 NVIDIA H100 GPUs, and 2 TB main memory. For generating poisoned samples using the NightShade and Glaze binaries, we had to use a different system as these binaries can be executed only in Windows or MacOS. We used here 2 computers, each having an Intel Core i7 CPU, 16GB main Memory, and an NVIDIA GeForce RTX 2070 running Windows 10. To automate the GUI-based generation process, we utilized the software SikuliX [19].

Poison	Accuracy	TPR	TNR	Entropy Cut-Off
NightShade	1.0000	0.9998	1.0000	1.7092
NightShade LPIPS $p = 0.004$	0.9968	0.9987	0.9961	0.1157
NightShade Linf $p = 0.04$	0.9998	0.9999	0.9998	0.5847
Binary NightShade	0.9298	0.9655	0.9286	0.0036
Glaze	0.9769	0.9726	0.9770	0.0216
Mist	1.0000	0.9984	1.0000	3.4316
MetaCloak	0.8437	0.9177	0.8432	0.0014

Table 2: Detection accuracy comparison across different poisons after training on NightShade, Mist, and Glaze. NightShade, as presented by Shan *et al.* [23], is highlighted.

Aligned with the work of Shan *et al. et al.* [23] we used the LAION-Aesthetics 120M dataset [8]. We leveraged the published binaries for Glaze [25] and NightShade [24], and the published code for NightShade [23], Mist [11], and MetaCloak [13]. Notably, as the code of NightShade differs from the paper by using a different metric and hyperparameters, we carefully adapted the code accordingly and additionally generated several variations, resulting in 9 datasets as depicted in Tab. 1. The original metric and parameter as described in Shan *et al.* [23] is LPIPS with a threshold of p=0.07 and is denoted as NightShade in the table and we have added another version with a lower threshold of p=0.004. Linf with p=0.04 is the setting used in Shan *et al.*'s codebase. Due to the time-intensive nature of the generation process, which can take up to 7.5 minutes per sample, the datasets for NightShade, Glaze, Mist, and MetaCloak contain fewer images compared to the other datasets. The details are described in App. A. Train/test split was initially set at 70%/30%. Due to the tools' time-intensive generation, dataset sizes vary across different approaches (Tab. 1). However, each split contains sufficient samples to ensure statistically reliable results.

To measure the depoisoning effect, we generated datasets using the NightShade defense, consisting of 10 000 clean and 3 000 poisoned samples, with all poisoned perturbations aimed at the same class. In particular, we introduced perturbations into 3 000 dog images, with the target class being cats. We use NightShade as described in the paper of Shan *et al.* [23], which employs perturbations with an LPIPS budget of 0.07. The NightShade samples generated by the

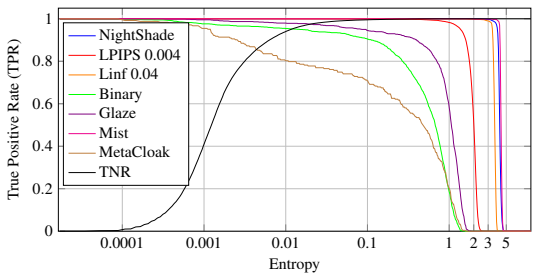


Figure 4: Detection rate of LightShed for different versions of NightShade and other defense approaches (Glaze, Mist, MetaCloak), in dependence of the cutoff entropy.

binary application were excluded from the fine-tuning comparisons because the intended poison effect by the authors could not be successfully reproduced for our model, as shown in Fig. 8. This discrepancy may arise because the binaries autonomously select poison targets upon processing an image input with some unknown poison strength threshold, potentially resulting in perturbations that are less impactful than those generated using our code, thus influencing their efficacy.

5.2 Evaluation Metrics

Detection Accuracy Metrics We evaluated the detection accuracy of our model using the following metrics:
Accuracy: Proportion of correctly identified images (clean and poisoned) out of the total number of images.

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Number of Samples}} \tag{6}$$

True Negative Rate (TNR): The proportion of actual clean images correctly identified by the model.

$$\text{TNR} = \frac{\text{True Negatives}}{\text{True Negatives} + \text{False Positives}} \tag{7}$$

True Positive Rate (TPR): The proportion of actual poisoned images correctly identified by the model.

$$\text{TPR} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{8}$$

CLIP Score Evaluation. The Contrastive Language-Image Pre-Training (CLIP) is a DNN trained on image and text pairs [18]. The CLIP score is calculated by evaluating the alignment between the embeddings of an image and text generated by the CLIP model within a shared vector space, where higher scores indicate a stronger semantic correspondence between the two. To assess poisoning effectiveness, we generated 1000 images with the prompt "a photo of a dog" and compared CLIP scores for the labels "a photo of a dog" and "a photo of a cat." Higher "cat" and lower "dog" scores in the fine-tuned model indicate successful poisoning.
Structural Similarity Index Measure (SSIM). To assess the quality of poison reconstruction in the depoisoned images, we calculated the SSIM value differences between the real and generated poison. We reported the average SSIM value differences across all test samples.

5.3 Detection Performance of LightShed

The detection performance for various poisoning schemes is shown in Tab. 2 and Fig. 4. The autoencoder was trained in an incremental process, where training was initiated with fewer data, and more were added as more poisoned images were created. The model was trained for 270 epochs in total, using samples generated with the published code of Shan *et al.* with

Poison	Accuracy	TPR	TNR	Entropy Cut-Off
NightShade	0.9999	0.9996	1.0000	2.1205
NightShade LPIPS $p = 0.004$	0.9877	0.9996	0.9842	0.0208
NightShade Linf $p = 0.04$	0.9998	0.9996	0.9998	1.1513
Binary NightShade	0.9701	0.9682	0.9702	0.0130
Glaze	0.8244	0.7506	0.8263	0.0038
Mist	0.9979	0.9969	0.9980	0.1487
MetaCloak	0.8669	0.9244	0.8665	0.0047

Table 3: Detection performance for different perturbation schemes when LightShed was only trained to detect NightShade for up to 250 epochs. The results for poisoning schemes that were not included in the training data are highlighted.

the Linf perturbation set to p=0.04, the corrected NightShade implementation using LPIPS metric with p between 0.000005 and 0.07, Glaze images, as well as Mist images. To address the small number of available samples for Glaze and Mist, their training samples were added in later phases of the training. Due to not having enough samples for MetaCloak, it was not used for training but only used for testing. The threshold for each poison method is tuned using a validation dataset comprising 10% of unused samples (see Sect. 4.4).

The model achieves almost perfect detection performance for NightShade LPIPS with p=0.07 being the version that Shan *et al.* [22] describe in their paper. Even with a much lower threshold of p=0.004, we achieve a TPR of 0.9987, which shows the detection accuracy of LightShed even for smaller perturbations. NightShade with Linf perturbations also achieves almost a perfect performance with a TPR of 0.9998. The NightShade binary achieves a performance of a TPR of 0.9655 and a TNR of 0.9286.

Fig. 4 shows how the different entropy thresholds affect LightShed's TPR and TNR. The TNR is approximately up to the precision of 10^{-6}, the same for each perturbation method since the entropy of clean generated poisons is very close to zero because it is a black image. Notably, for the TPR, the entropy at which the curve goes down is different for each method as for each dataset a different entropy threshold is calculated. This shows the average strengths of the poison perturbations. While LightShed effectively detects the poisoned images for all protection techniques, it is most efficient for the regular NightShade (LPIPS p=0.07), followed by Linf 0.04, LPIPS 0.004, and finally, the NightShade binary. For the binary, the reason why some images are not detected might be the low strengths of the applied perturbation. Since we could not reproduce a poisoned fine-tuning process with these binary images, we assume the low poison levels to be not very effective. Thus, missing the weakest 5% of perturbed binary samples likely will not impact fine-tuning.

Despite training for only 65 epochs on Glaze and Mist with limited data and none on MetaCloak, the model still achieved TPRs of 0.9726, 0.9984, and 0.9342, respectively. Fig. 6 shows that MetaCloak is a more detailed type of poison, and without having trained on it, it is challenging to reconstruct the perturbations. Fig. 4 shows that the entropy values for MetaCloak generated poisons remain below 1.5. However, the real poisons have much higher entropy values, which can also be seen in the high entropy difference for MetaCloak in Figure 5. The low entropy values of the generated poisons show that the extracted perturbation for MetaCloak is not as accurate as for the other techniques due to a lack of training on it. However, even in the absence of training data for MetaCloak, we were still able to detect it with a high TPR of above 90% for all poisoned samples.

To evaluate LightShed's ability to handle samples being poisoned with unknown protection schemes, we trained an autoencoder only for NightShade and evaluated its ability to detect Glaze, Mist, and MetaCloak. As shown in Table 3, despite the unrealistic scenario, LightShed still achieves accuracies of above 80% for all methods without the encoder having seen the type of poison before. This shows that these perturbations are all similar to some extent, and the autoencoder can generalize to a high degree what poison is.

The classification threshold can be automatically tuned using a validation dataset (Sect. 4.4). Notably, many thresholds achieve high TPRs and TNRs across all methods (Fig. 4), allowing a single shared threshold without significant performance loss, especially for well-reconstructed perturbations. To evaluate a shared threshold, we used a validation set comprising 10% of unused samples and an equivalent number of clean samples. To ensure a realistic experiment setting for assessing LightShed's generalizability, approaches where no training data were available (see Tab. 1) were also not included in the validation set. When choosing a threshold that achieves a fixed FPR = 0.005 (TNR = 0.995) on a validation dataset, we obtained a threshold of 0.07028048 with the validation set. This resulted in an overall TPR of 0.9941 and a TNR of 0.9959 on the test set and individual results for attacks as detailed in Tab. 4. With sufficient samples, also separate autoencoders can be fine-tuned for each method, flagging an image as poisoned if flagged by any model.

5.4 Effect of Image Augmentation

Compression techniques are commonly applied to images posted on the internet, and also training pipelines for DNNs frequently incorporate image augmentation methods such as rotation, Gaussian noise, and JPEG compression. We evaluated the efficacy of LightShed under these conditions, specifically focusing on its performance against NightShade and observed its robustness, as shown in Table 5. We included random rotations of up to 100 degrees in our test and observed that the autoencoder correctly identifies unperturbed areas of perturbed images as black areas and does not lose the ability to differentiate between poison and clean samples. For Gaussian noise, we tested up to the noise of a standard deviation of 0.1, which is double the standard deviation that Hönig *et al.* [6] used to defend against Glaze. Due to the remaining

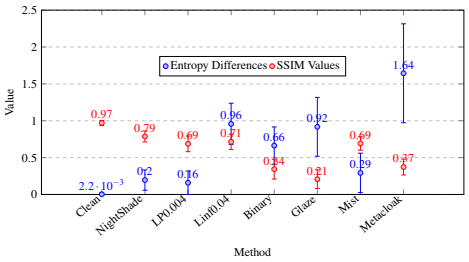


Figure 5: Entropy differences and SSIM value differences between real and generated poison with error bars for images poisoned with different methods and clean images.

high detection accuracy, we assume that the perturbations caused by noise must be different enough from those caused by the poison for the autoencoder to recognize. We tested JPEG compression down to quality 20 and observed that the autoencoder still successfully detects these images.

5.5 Quality of Generated Poison

Fig. 6 compares the real poisoning perturbation with the poison that LightShed extracted. The figure shows that the poison is well reconstructed for regular NightShade and Linf=0.04. For NightShade LPIPS =0.004, the poison strength varies by pixel, but the structure remains similar to the original. In NightShade Binary, the real poison is less visible, leading to less accurate extraction. The Glaze poison reconstruction shows that the gist is captured, but our generated poison takes more into account the structures of the original image, so black lines can be found along these. Mist is a very strong type of poison that can be reconstructed almost fully. Finally, MetaCloak is a more detailed poison, with a visible mouse head in the real image. While our autoencoder captured the structure and surrounded stronger poison, it missed finer strokes, likely due to lack of training on MetaCloak.

Figure 5 gives a better overview of the quality of reconstruction with the SSIM values and entropy difference values on average of all test samples. For NightShade (regular, LPIPS 0.004, Linf 0.04) and Mist, the SSIM values are quite high, while the ones for Binary, Glaze, and Metacloak vary more. We assume that more samples will make the poison generation better for these poison methods. Especially in MetaCloak, where the entropy difference is also quite high compared to the original poison, training on a large set of samples should improve the reconstruction results drastically.

In Figure 7, the effect of the depoisoning on regular NightShade is shown. Figure 7(c) shows the LightShed cleaned image, achieved by subtracting the reconstructed poison from the poisoned image. Below, the real poison, reconstructed poison, and the difference between the clean and LightShed cleaned images are displayed, showing most of the large cat perturbations removed. Though pixel differences in (f) seem large, they are minor as they are not visible in (c) and represent

Metric	NightShade	LPIPS 0.004	Linf 0.04	Binary	Glaze	Mist	MetaCloak	Overall
TPR (%)	100.00	99.92	100.00	91.69	95.82	100.00	73.66	99.41
TNR (%)	99.59							

Table 4: True Positive Rates (TPRs) and True-Negative-Rate (TNR) of shared classification-threshold when setting FPR to 0.5% on the shared validation set.

small value changes. Our goal of making the poison ineffective is confirmed through visual inspection. Hönig *et al.* [6] suggest noisy upscaling for Glaze, and we similarly tried applying noisy upscaling to the LightShed cleaned image in (g) and regular upscaling in (h). Using Gaussian noise with a 0.05 standard deviation, we halved Hönig *et al.*'s noise parameter from 320 to 160, yet still saw distorted results. For the LightShed cleaned and upscaled image in (g), adding the same noise with an upscaler value of 20 produced an even cleaner result, as seen in (i).

5.6 Depoisoning Performance of LightShed

Further, as discussed in Sect. 5.2 we evaluate our method on NightShade in a stronger fine-tuning scenario of 10 000 clean images and 3 000 poisoned dog images with target poison cat. Figure 8 shows the results of fine-tuning on different sets of 3 000 dogs. The original score reflects cat and dog scores when prompting for a dog with a not fine-tuned Stable Diffusion v2. The clean result is from fine-tuning on 10 000 clean images and 3 000 clean dogs, with average dog scores for these around 43 and cat scores between 12.5 and 14.5. Using 3 000 NightShaded dogs (LPIPS p=0.07), dog and cat scores converge to 27–30. With binary poison, the scores resemble the clean scenario, indicating ineffective poisoning. After applying LightShed cleaning, cat and dog scores align with the clean and original fine-tuning results.

Further, we tried upscaling the LightShed cleaned image with a noise parameter of 20 and just applying noisy upscaling to the poisoned image with a noise parameter of 160 and got similarly clean results. However, when we just apply noisy upscaling without LightShed, the maximum score for cat is at 30.64, much higher than the maximum cat scores for original (25.34), clean (22.79), LightShed (26.12) and LightShed +upscaling (24.09). This indicates that there exists at least one

Augmentation	Accuracy	TNR	TPR
Rotation	0.9998	0.9990	1.0000
Gaussian noise std = 0.01	0.9999	0.9995	1.0000
Gaussian noise std = 0.05	0.9999	0.9995	1.0000
Gaussian noise std = 0.1	0.9998	0.9988	1.0000
JPEG compression q=20	0.9999	0.9995	1.0000

Table 5: Detection accuracy comparison across different image augmentations for NightShade.

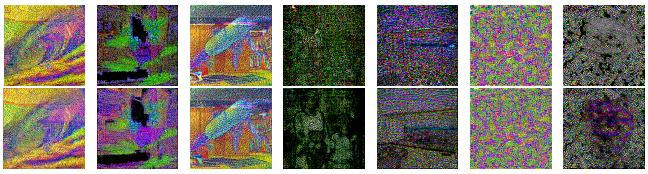


Figure 6: Generated and Real Poison Comparison. The upper row is the real poison, whereas the lower row is the generated poison from left to right. The poisons are NightShade, NightShade LPIPS=0.004, NightShade Linf 0.04, NightShade Binary, Glaze, Mist and Metacloak.

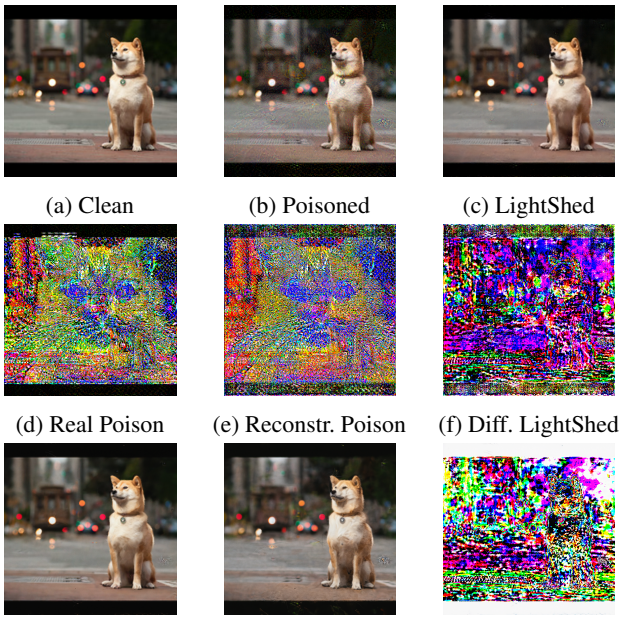
more cat-like-looking image than in the above-mentioned - settings. Furthermore, generally, the problem with using noisy upscaling is that this noisy upscaling process takes 55 seconds on average per image on an H100, whereas generating a poison with LightShed only takes 0.0139 seconds on average. Upscaling 3000 images took about two whole days on our server, whereas LightShed takes less than a minute.

Additionally, we conducted noise experiments on the poisoned dogs and 10 000 clean images. We added Gaussian noise with standard deviations of 0.025, 0.05, 0.075 and 0.1. In case we did not have a detection tool such as LightShed, noise can also substantially hinder NightShade poisoning from a noise value of 0.05. However, the quality of all images is degraded, as can be seen by the lower mean dog and cat scores of 2 to 3 points. This shows that only noising a large dataset using a high standard deviation will not be feasible for the long-term quality of generated images. However, for poisons, where a not high enough TPR is reached, and poison reconstruction is poor on the validation set, the TPR and TNR ratio can be shifted in favor of TPR, and then a larger set being detected as poisoned can perhaps just be noised instead. Limiting the number of images trained with noise should lighten the adverse effect of noise on the long-term image quality.

Further, we conducted an experiment fine-tuning on just 3000 NightShaded dogs that were cleaned with LightShed. The cat and dog scores remained in a clean range, with the dog score at 42.3801 ± 2.5190 and the cat score at 12.6305 ± 2.8173. The maximum cat score also remained low at 22.0119. This means that even in the stronger scenario where 100% of the images are poisoned, LightShed can clean these sufficiently such that the poison effect disappears.

5.7 User Study

To complement the metric-based evaluation with human perception, we conducted a user study to assess the efficiency and effectiveness of LightShed. The design of the study follows the previous research on adversarial attacks against image generators [6]. Using Amazon mTurk, participants were presented with pairs of images: one clean and one generated by an image model trained on poisoned samples. They were asked to evaluate which image better satisfies specific criteria



(a) Clean (b) Poisoned (c) LightShed

(d) Real Poison (e) Reconstr. Poison (f) Diff. LightShed

(g) LightShed +Up. (h) Upscaled (i) Diff. Upscaling

Figure 7: Comparison of different depoisoning options, showing (a) Original clean image, (b) Poisoned image (NightShade) (c) LightShed depoisoned image through subtraction of generated poison (d) Real Poison (e) Generated Poison (f) The subtraction of the original clean image and the LightShed depoisoned image (g) LightShed depoisoned image with subsequent noisy upscaling [21] (h) Depoisoned image by noisy upscaling [21] (i) The subtraction of the original clean image and the LightShed depoisoned + noisy upscaled image

as detailed below.

We generated images for five state-of-the-art attack methods (NightShade [23], NightShade with Linf metric [23], NightShade Binary [24], Glaze [25], and Mist [11]) and four defense variations (No Defense, LightShed, Upscaling, LightShed + Upscaling), resulting in a total of 20 configurations. For each configuration, 15 samples were produced, with participants assessing each sample based on five criteria: *(i)* fit for the prompt "a photo of a dog", *(ii)* fit for the concept "dog", *(iii)* overall quality, *(iv)* presence of noise/artifacts, and *(v)* level of detail. For each criterion, participants selected the sample they felt better fulfilled the respective requirement.

Images were divided into batches of 60 comparisons and 10 control questions, allowing up to 15 participants per batch. Each batch included three comparisons for each configuration. To ensure data reliability, multiple attention checks were applied, unfortunately, leading to the exclusion of many submissions. At the end, 34 submissions from 26 unique participants passed these checks, resulting in 102 valid answers for each measurement. Details on its design (attention checks, criteria, and questions) are provided in App. B. The study adhered strictly to the institution's IRB rules.

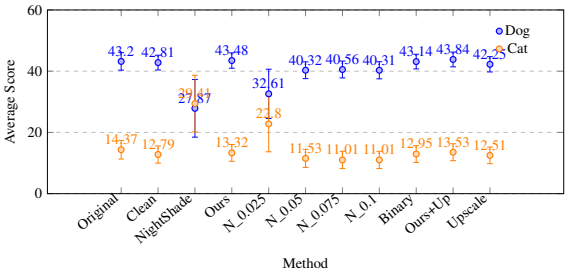


Figure 8: Average CLIP Scores with Error Bars of Fine-Tuned Diffusion Models for Dog and Cat across different settings. Here $N_{0.025}$ to $N_{0.1}$ are the Gaussian noise settings with standard deviations from 0.025 to 0.1.

In summary, LightShed significantly enhanced image quality. Across all five criteria, it significantly reduced the users' likelihood of identifying the clean image to close to 50% (the performance of a naïve classifier). For the "concept of a dog", user accuracy in identifying clean images dropped from 64.6% to 53.1%. When assessing overall quality, the selection of clean images decreased from 71.7% to 59.8%, and for noise levels, it dropped from 59.9% to 52.5%. Detailed results for each criterion, separated by attack and defense techniques, are shown in Fig. 9. Furthermore, the combined application of LightShed and Upscaling yielded minor additional improvements, e.g., in the noise evaluation, users' ability to identify clean images was further reduced to 51.1%.

6 Discussion

We introduced LightShed, which effectively identifies and cleans poisoned images. In the following, we will discuss the impact of the attack (Sect. 6.1), its limitations (Sect. 6.2), as well as directions for more robust protection techniques (Sect. 6.3). In App. C, we discuss LightShed's applicability on cryptographic watermarks. The ethical implications of this work will be discussed in Sect. 9.

6.1 Impact of LightShed

LightShed models the characteristics of image protection schemes and leverages this fingerprint to detect even subtle poisoning artifacts in images. It effectively identifies perturbations from both fine-tuning prevention schemes and training disruption schemes, efficiently removing these perturbations through reconstructed poisoning patterns. As demonstrated in Sect. 5, LightShed is highly effective against state-of-the-art protection schemes, disrupting not only fine-tuning processes but also entire training procedures. The denoising component of LightShed effectively removes poisons, allowing the utilization of protected images for training and fine-tuning image generators, thereby fulfilling requirement R1.

Notably, once images are uploaded to the internet, their protection cannot be retroactively updated, as described in Sect. 2.2. If an artist uploads a protected image and an attacker downloads and stores it locally, the attacker can train LightShed to compromise the protection by analyzing a sufficient number of example images. Even if the artist applies a new defense and uploads the images again, the previously stored versions remain unchanged, allowing the attacker to continue using them for training an image generation model.

To ensure LightShed's generalizability, it was designed to recreate poisoning behavior using image samples and to then tune a detection threshold with a validation set. However, it does not rely on source code for training, as demonstrated through the use of precompiled binaries of protection tools (Sect. 5). Training data for widely used tools can be effortlessly obtained by utilizing respective binaries. Leveraging an autoencoder-based approach, LightShed effectively models the characteristics of protection schemes, allowing it to dynamically adapt to emerging detection techniques. A single model with a unified threshold is sufficient to classify images against multiple perturbation methods, including proprietary solutions such as MetaCloak, even when these methods are excluded from the training data (Sect. 5.3).

The evaluation further shows LightShed's capability to generalize to previously unseen detection techniques, effectively neutralizing protections without prior exposure (Sect. 5.3). Moreover, historical trends indicate that algorithms or binaries of protection tools are frequently leaked, reinforcing the validity of LightShed's's training data assumptions. This capacity to generalize not only fulfills R2 but also addresses C3, showcasing LightShed's 's robust adaptability across a diverse range of protection mechanisms.

Furthermore, the Entropy Cut-Off mechanism distinguishes between clean and poisoned inputs. This allows LightShed to apply the depoisoning algorithm only to poisoned samples and leave clean samples unchanged. Depending on the application scenario, the poisoned samples can either be excluded or processed by the Subtraction and Denoising component. This selective approach allows LightShed to protect the training process by mitigating the poisoning without affecting the clean samples' utility, fulfilling requirement R3.

Additionally, LightShed does not rely on any assumptions about the images used for poisoning and training. When collecting a dataset from the internet, some images may already be protected. However, LightShed trains its poisoning detection by comparing the differences between the original and poisoned versions, with the autoencoder learning to identify these differences. If the original image is already poisoned, this pre-existing perturbation does not affect the difference detected between the two versions. Therefore, LightShed is robust against pre-poisoned images and does not require a clean dataset, addressing challenge C4.

These findings show LightShed's effectiveness in accurately detecting and depoisoning images protected by the most recent protection schemes. This demonstrates the vulnerability of existing protection methods and highlights the need for

more sophisticated techniques to safeguard artists' work from being used to train image generators without their consent.

6.2 Limitations of LightShed

While LightShed provides a significant advancement in defending diffusion models against adversarial perturbations, several limitations should be acknowledged.

A key limitation of LightShed is its data dependence. Without a sufficient number of pairs consisting of clean samples and their corresponding poisoned versions, LightShed cannot effectively model the characteristics of the image defense technique. Therefore, a central assumption of LightShed is that the attacker has the capability to apply protection techniques to specific images in order to create a training dataset. However, for protection tools to be widely adopted by artists, they must be easily accessible, typically through pre-compiled binaries or web services provided by the developers. While this accessibility is crucial for user adoption, LightShed can exploit these low accessibility barriers to generate poisoned training samples. This enables it to effectively learn to detect and neutralize the perturbations that these techniques are designed to protect against.

However, even when such a tool is available, creating a sufficient number of training samples can be challenging if the poisoning process is very time-consuming. For instance, in the case of MetaCloak, it took approximately 4 hours to poison a batch of 32 samples, resulting in about 7.5 minutes per sample on average. While this duration might be acceptable for an artist who has spent considerable time on an art piece and wishes to protect it before uploading, it significantly slows down the generation of training data for an adversary. For example, generating a dataset of 5 000 images would require approximately 26 days. On the other hand, the adversary could download the images once they are published, begin generating and collecting the necessary training data during this time, and later unpoison the downloaded images, as the protection cannot be retroactively updated (see Sect. 6.1).

LightShed efficiently extracts the poisoning perturbation from a given image by utilizing the previously determined defense fingerprint. However, the perturbation is also affected by specific characteristics of the image, such as its structures, edges, and textures, which can introduce minor variations. These variations can result in slight noise in both the reverse-engineered perturbations and the reconstructed images. However, as demonstrated in Sect. 5.6, LightShed can be effectively combined with various denoising approaches to precisely recover the original image, even in such cases.

6.3 Incentives for Robust Image Protection

LightShed demonstrates the limitations of existing image protection techniques by overcoming current state-of-the-art methods. In Sect. 5.4, we showed LightShed's robustness against common image augmentation techniques. Additionally, the Poison Reconstruction component allows dynamic adaptation to new protection schemes. However, protecting artistic work remains crucial for creators, and LightShed highlights the need for more robust defenses. Future defense strategies must integrate perturbations such that removal without significant image degradation is infeasible. LightShed is able to reverse-engineer perturbations due to similarities in perturbation characteristics, structural irregularities, and their distribution across the image. Below, we outline potential strategies for designing more resilient protection techniques.

Image Specific Perturbation: Less recognizable patterns could be injected into the images. If poison for each individual image was constructed in a less structured way but would be more unique and less predictable, it would be significantly harder for LightShed to learn the techniques' characteristics.

Vary Perturbation Density: Another idea includes localizing the noise to specific regions to make it harder to detect. For example, the NightShade binary already adds less noise all over the image, which made it more challenging for LightShed to extract the perturbation, though it still detected the poison.

Noise-Aligned Perturbation: Also, the poison could be made structurally more similar to Gaussian noise. Then, it would be harder to distinguish this from actual Gaussian noise. For the poison destruction, obviously, some Gaussian noise might be able to destroy this type of noise. However, applying noise to every image may degrade quality in large-scale training.

7 Related Work

In response to the concerns of copyright owners, various schemes have been developed to protect intellectual property. In parallel to this development, various attack methods have been discussed in the literature to determine the security of these schemes and show their limitations and the necessity for more robust defense mechanisms. Notably, existing work focuses widely on circumventing fine-tuning defenses [6, 15] or out-of-distribution data [16], while more sophisticated concept poisoning defenses such as NightShade [23] that LightShed addresses, were not considered in the existing literature. In the following, we will analyze and compare existing fine-tuning protection schemes with LightShed. These schemes can be categorized into targeted mitigation strategies, which focus on identifying poisoned samples, and untargeted strategies, which must be applied universally to all samples.

7.1 Targeted Mitigations

Nguyen *et al.* propose regenerating captions for poisoned samples as a defense against attacks that modify the images' captions. However, this approach does not counteract the adversarial perturbations embedded in the images themselves. An *et al.* introduce a method that aims to exploit the objective

of making perturbations imperceptible to humans. They photograph poisoned images to simulate human perception and compare these photographs to the original images to detect perturbations [1]. However, since their method does not include a mechanism to specifically identify poisoned samples, applying this approach to large datasets with thousands of images is impractical. In contrast, LightShed operates fully automatically, enabling it to both identify poisoned images and remove the perturbations efficiently, ensuring scalability.

Cao *et al.* exploit the fact that some style mimicry defense schemes target latent space representations. They train a diffusion model to process input images and regenerate the same samples as output, arguing that if the latent space representations have been manipulated, the output should differ significantly from the input [2]. However, this approach assumes that the perturbation prevents the diffusion model from reproducing the input image accurately, which limits its applicability to defenses that do not explicitly target the latent space and do not account for simple adversarial examples. Pan *et al.* suggest a method that requires the availability of a clean dataset in addition to a dataset that is suspected of containing poisoned samples. They employ an asymmetric loss function that minimizes the loss on the clean dataset while maximizing the loss on the suspect dataset. By measuring the loss for each sample, they iteratively move the samples with the lowest loss to the clean dataset until only 5% of samples remain [16]. However, without directly identifying the presence of poisoned samples, this method risks excluding valuable samples where the model performs poorly, even if those samples would benefit the model's training. In contrast, LightShed accurately identifies poisoned samples and, in the absence of adversarial perturbations, can accept all samples, thus preserving the model's utility.

7.2 Untargeted Mitigations

VA3 is based on the assumption that copyright protection mechanisms are unreliable, leading to generating the desired sample when a prompt is repeatedly queried across multiple iterations. To exploit this, VA3 identifies the best-generated sample by repeatedly querying the prompt and scoring the generated images [10]. However, VA3 assumes that the defense mechanism is not robust enough to fully corrupt the training process, allowing the diffusion model to still generate appropriate images. Notably, recent attacks even noticed that when performing a strong attack on one target concept, this can even affect non-targeted concepts, highlighting the strengths of state-of-the-art attacks. In contrast, LightShed's detection capability allows an attacker to filter out or depoison the images before training, ensuring that the model can be trained without being compromised by poisoned samples.

Nie *et al.* add noise to the images and use diffusion to remove the noise [15]. Zhao *et al.* split the images into several parts and then used the approach of Nie *et al.* to denoise the images [15]. Qin *et al.* use a combination of image processing techniques such as JPEG and denoising to remove the perturbation. Hönig *et al.* [6] reevaluate different existing techniques such as Gaussian noising, the denoising proposed of Nie *et al.* [15], and noisy upscaling [22] to mitigate style mimicry protections. Based on Impress of Cao *et al.* [2], they design a perturbation removal scheme using reverse encoder optimization, negative prompting, and the denoising of Nie *et al.* [15]. They show that their approach can mitigate fine-tuning defenses as Glaze but do not consider the more advanced training-disruptions techniques such as NightShade. In general, without identifying the poisoned samples, untargeted approaches need to be applied to all images in the train dataset and affect the training quality. In comparison, the detection of LightShed allows the application of the perturbation removal technique only to images containing a perturbation. Further, the detection allows the attacker to decide to exclude these images to prevent any adverse impact or apply recovery techniques. Notably, the above-mentioned techniques could also be combined with the detection component of LightShed.

8 Conclusion

Protecting artwork from nonconsensual imitation in text-to-image models is increasingly vital and various protection schemes to safeguard creative works have been developed. Different attacks aimed to demonstrate the insufficiency of fine-tuning protection schemes. However, these attacks are impractical or fail to identify poisoned samples directly. In response, we introduced LightShed, the first solution that addresses both, fine-tuning protection and training disruption techniques, effectively overcoming previous attack limitations. LightShed learns the characteristics of protection schemes and applies this knowledge to detect, extract, and neutralize adversarial perturbations. We conducted a comprehensive evaluation of LightShed across various settings and considered different state-of-the-art poisoning schemes, including NightShade, Glaze, Mist, and MetaCloak. Demonstrating the insufficiencies of current techniques in protecting artists' work, we show the need for more resilient protection techniques. Building on these insights, we propose directions for more robust detection methodologies, setting the base for new approaches that can effectively protect creative works.

9 Ethics and Open Science Considerations

LightShed shows the limitations of existing image protection techniques and the necessity for more robust solutions. However, by demonstrating these vulnerabilities, LightShed could potentially be abused by malicious actors to circumvent protections in case artists or image creators use such techniques on publicly uploaded images. Therefore, we will first analyze the ethical dimensions of our research (Sect. 9.1), before dis-

cussing how our efforts to address these ethical considerations intersect with the principles of open science(Sect. 9.2).

9.1 Ethics Considerations

Our research, which shows the limitations of current image protection techniques, requires a thorough discussion of important ethical questions, particularly concerning the intellectual property rights of artists and image owners, as well as its impact on the creative community.

We carefully analyzed the *beneficence* of our work prior to initiating the project, continuously during the research process, and before publication. Further, we will continue to evaluate the impact of our research, not only its benefits but also any potential harm in the future. This research is intended as a demonstration of the need for more robust image protection methods. Simultaneously, it serves as a benchmark for the development of future protection techniques, thereby enabling the creation of more effective measures and supported efforts to safeguard the work of creative people. Consequently, we concluded that the potential benefits of enhancing protection for artists and image owners outweigh the risks of misuse of our findings, particularly as we will not make the source code publicly accessible. Access to the source code will be granted only upon request after a thorough review of the request and confirmation of its intent (see Sect. 9.2).

Additionally, in our research, we considered pre-compiled binaries for Glaze and NightShade. To ensure that the developers of these tools can appropriately respond and make necessary adaptations, we plan to *disclose* our findings to them prior to the publication of this work upon acceptance of this manuscript.

Regarding *justice* concerns, we noticed that the websites offering binaries for Glaze and NightShade do not specify terms of use [24, 25]. The source code provided by Shan *et al.* on GitHub [23] is distributed under the GNU GPLv3 license, which allows for its use and modification. Therefore, we believe that our research respects *justice* considerations and is aligned with *legal and public interest standards*.

To ensure a comprehensive evaluation, we performed a user study showing LightShed's effectiveness. The study was strictly aligned with the IRB rules of our institution. Besides the answers to the image-comparison questions, no other data was collected as part of the user study.

9.2 Open Science Considerations

In our research, we also considered the pre-compiled binary files for Glaze and NightShade. Although these binaries are not commercial products, they could be utilized by artists or image owners. The potential use of these protection techniques, even by a single person, raises significant concerns regarding the impact of this person's copyright if LightShed's source code were made publicly available.

To ensure that LightShed is used only responsibly for future research, it is, therefore, essential to prioritize the protection of copyrighted material and respect the rights of creators. This involves balancing the advancement of AI capabilities with the need to protect the interests of content creators and other stakeholders. Therefore, the source code for LightShed will only be made available upon request, with the confirmation that it be used only for responsible research purposes: https://zenodo.org/records/14727581

Acknowledgment

This research received funding from the Horizon program of the European Union under grant agreements No. 101093126 (ACES) and No. 101070537 (CROSSCON), OpenS3 lab, as well as the Federal Ministry of Education and Research of Germany (BMBF) within the IoTGuard project. Further, this research was supported by Tazaki-Cambridge Studentship as part of the Cambridge Trust funding. Murtuza Jadliwala was partially supported by UTSA's Faculty Development Leave (FDL) program and by the National Science Foundation (NSF) under award number 1943351.

References

[1] Shengwei An, Lu Yan, Siyuan Cheng, Guangyu Shen, Kaiyuan Zhang, Qiuling Xu, Guanhong Tao, and Xiangyu Zhang. Rethinking the invisible protection against unauthorized image usage in stable diffusion. In *USENIX Security*. USENIX Association, 2024.

[2] Bochuan Cao, Changjiang Li, Ting Wang, Jinyuan Jia, Bo Li, and Jinghui Chen. Impress: evaluating the resilience of imperceptible perturbations against unauthorized data usage in diffusion-based generative ai. *NeurIPS*, 2024.

[3] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

[4] Kashmir Hill. This tool could protect artists from a.i.-generated art that steals their style. https://www.nytimes.com/2023/02/13/technology/ai-art-generator-lensa-stable-diffusion.html, 2023. Accessed: 2024-08-29.

[5] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

[6] Robert Hönig, Javier Rando, Nicholas Carlini, and Florian Tramèr. Adversarial perturbations cannot reliably protect artists from generative ai. In *arXiv preprint arXiv:2406.12027*, 2024.

[7] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019.

[8] LAION. Laion-aesthetics dataset. https://laion.ai/blog/laion-aesthetics/, 2022-08-16. Accessed: 2024-08-31.

[9] Liangqi Lei, Keke Gai, Jing Yu, and Liehuang Zhu. Diffusetrace: A transparent and flexible watermarking scheme for latent diffusion model. *arXiv preprint arXiv:2405.02696*, 2024.

[10] Xiang Li, Qianli Shen, and Kenji Kawaguchi. Va3: Virtually assured amplification attack on probabilistic copyright protection for text-to-image generative models. In *CVPR*, 2024.

[11] Chumeng Liang and Xiaoyu Wu. Mist: Towards improved adversarial examples for diffusion models. *arXiv preprint arXiv:2305.12683*, 2023.

[12] Hanwen Liu, Zhicheng Sun, and Yadong Mu. Countering personalized text-to-image generation with influence watermarks. In *CVPR*, 2024.

[13] Yixin Liu, Chenrui Fan, Yutong Dai, Xun Chen, Pan Zhou, and Lichao Sun. Metacloak: Preventing unauthorized subject-driven text-to-image diffusion-based synthesis via meta-learning. In *CVPR*, 2024.

[14] Thao Nguyen, Samir Yitzhak Gadre, Gabriel Ilharco, Sewoong Oh, and Ludwig Schmidt. Improving multimodal datasets with image captioning. In *NeurIPS*, 2024.

[15] Weili Nie, Brandon Guo, Yujia Huang, Chaowei Xiao, Arash Vahdat, and Anima Anandkumar. Diffusion models for adversarial purification. *arXiv preprint arXiv:2205.07460*, 2022.

[16] Minzhou Pan, Zhengting Wang, Xin Dong, Vikash Sehwag, Lingjuan Lyu, and Xue Lin. Finding needles in a haystack: A black-box approach to invisible watermark detection. *arXiv preprint arXiv:2403.15955*, 2024.

[17] Pytorch, 2019. https://pytorch.org.

[18] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*. PMLR, 2021.

[19] RaiMan. SikuliX, 2019. www.sikulix.com/ Accessed: 2024-07-03.

[20] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *ICML*. PMLR, 2021.

[21] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

[22] Shawn Shan, Jenna Cryan, Emily Wenger, Haitao Zheng, Rana Hanocka, and Ben Y Zhao. Glaze: Protecting artists from style mimicry by Text-to-Image models. In *USENIX Security*. USENIX Association, 2023.

[23] Shawn Shan, Wenxin Ding, Josephine Passananti, Stanley Wu, Haitao Zheng, and Ben Y Zhao. Nightshade: Prompt-specific poisoning attacks on text-to-image generative models. In *S&P*. IEEE Computer Society, 2024.

[24] Shawn Shan, Josephine Passananti, and Stanley Wu. Download nightshade. https://nightshade.cs.uchicago.edu/downloads.html. Accessed: 2024-07-03.

[25] Shawn Shan, Stanley Wu, Josephine Passananti, Ronik Bhaskar, and Lynds Gallant. Glaze. https://glaze.cs.uchicago.edu/. Accessed: 2024-08-27.

[26] Sunpreet Sharma, Ju Jia Zou, and Gu Fang. A novel multipurpose watermarking scheme capable of protecting and authenticating images with tamper detection and localisation abilities. *IEEE Access*, 10:85677–85700, 2022.

[27] Sunpreet Sharma, Ju Jia Zou, and Gu Fang. A single watermark based scheme for both protection and authentication of identities. *IET Image Processing*, 16(12):3113–3132, 2022.

[28] Sunpreet Sharma, Ju Jia Zou, Gu Fang, Pancham Shukla, and Weidong Cai. A review of image watermarking for identity protection and verification. *Multimedia Tools and Applications*, 83(11):31829–31891, 2024.

[29] Kareem Shehata, Aashish Kolluri, and Prateek Saxena. Clue-mark: Watermarking diffusion models using clwe. *arXiv preprint arXiv:2411.11434*, 2024.

[30] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*. PMLR, 2015.

[31] Catherine Thorbecke. 'it gave us some way to fight back': New tools aim to protect art and images from ai's grasp. https://edition.cnn.com/2023/08/12/tech/ai-images-photos-protection/index.html, 2023. Accessed: 2024-08-29.

[32] Thanh Van Le, Hao Phung, Thuan Hoang Nguyen, Quan Dao, Ngoc N Tran, and Anh Tran. Anti-dreambooth: Protecting users from personalized text-to-image synthesis. In *ICCV*, 2023.

[33] Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, Dhruv Nair, Sayak Paul, William Berman, Yiyi Xu, Steven Liu, and Thomas Wolf. Diffusers: State-of-the-art diffusion models. https://github.com/huggingface/diffusers, 2022.

[34] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[35] Chenfei Wu, Jian Liang, Lei Ji, Fan Yang, Yuejian Fang, Daxin Jiang, and Nan Duan. Nüwa: Visual synthesis pre-training for neural visual world creation. In *European conference on computer vision*. Springer, 2022.

A Dataset Generation

We aligned our dataset selection with the work of Shan *et al.* [23] by using the LAION-Aesthetics 120M dataset [8] as the source for generating datasets for each attack scheme. While we focus primarily on NightShade, other protection schemes such as Glaze, Mist, and MetaCloak are also included. We generated the poisoned images using the official repositories for MetaCloak [13], Mist [11], and NightShade [23]. Notably, the NightShade repository, although published by the authors, differs from the algorithm described in the paper. As the repository states, the provided source code uses the Linf metric. We carefully replaced the Linf metric with the LPIPS metric that is described in the NightShade paper [23]. We generated images using a p-value of 0.07 as defined by Shan *et al.* (denoted in the following as NightShade), but also evaluated LightShed's effectiveness for different variations, using p=0.004 (denoted as NightShade LPIPS 0.004) and Linf (referred to as NightShade Linf). Further, we generated another set of protected images by utilizing the compiled binary applications for NightShade and Glaze being provided by their authors [24, 25]. To speed up the generation of poisoned images, we automated running these binary applications using the SikuliX automation tool [19]. These datasets are used for training and evaluating LightShed, facilitating a comprehensive evaluation of its detection capabilities as detailed in Sect. 5.3, and the effectiveness of various depoisoning methods, which are further investigated in Sect. 5.6.

The separation between the training and test datasets is shown in Table 1. Due to the time-intensive nature of generating poisoned images, the datasets for NightShade and Glaze binary applications, Mist, and MetaCloak contain less images. For instance, creating poisoned images with binary applications is implemented on computers with an Intel Core i7 CPU, 16GB main Memory, and an NVIDIA GeForce RTX 2070. Generating each poisoned image takes approximately 5 minutes. Generating MIST-poisoned images takes around 2 minutes per image, and NightShade poisoned images with provided code using Linf and LPIPS require approximately 48 seconds on our main server. For a batch size of 32 images, generating a batch of MetaCloak images took approx. 4 hours on this server.

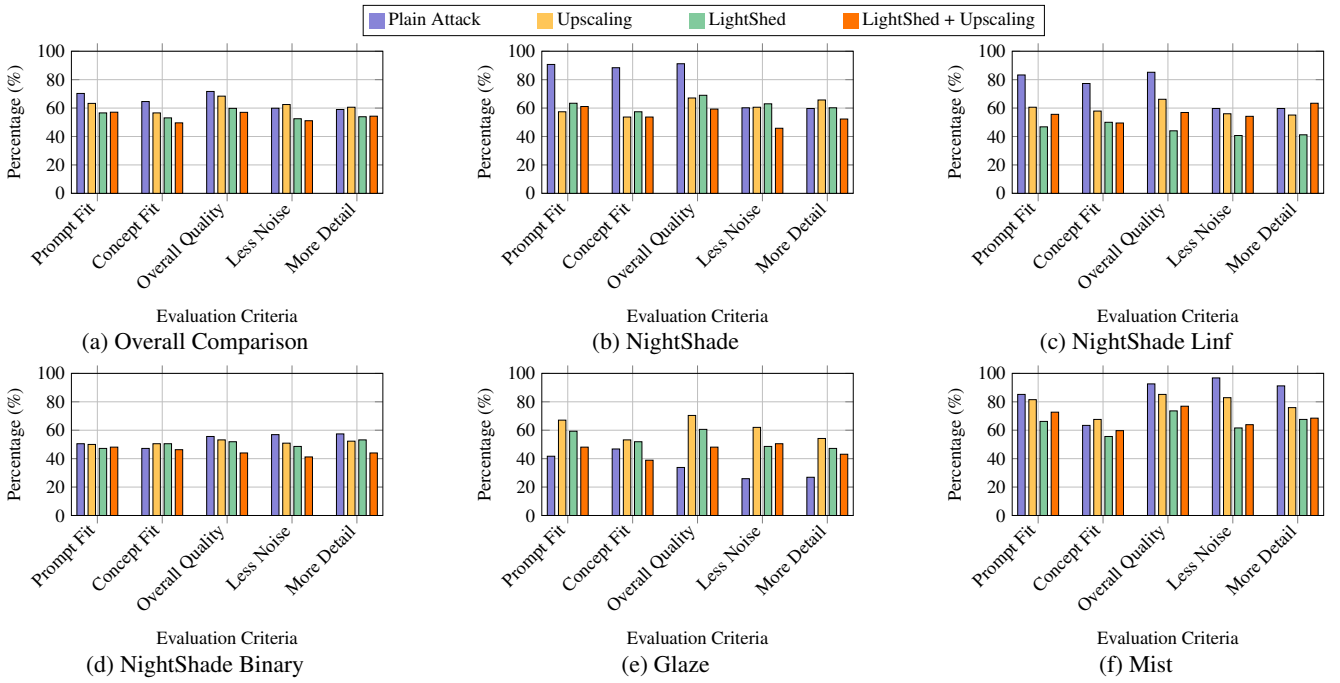
B Details on User Study

B.1 Study Design

Following the methodology established in prior work on attacks against image generators [6], we conducted a user study through Amazon Mechanical Turk (mTurk). The user study was performed strictly following the institution's IRB rules. We recruited participants who had achieved Master status on mTurk and maintained an approval rate of at least 50%. Each batch contained 60 image pairs for comparison. Each pair consisted of one image from a clean model and another from a model fine-tuned on attacked data. The study evaluated four defense approaches (No Defense, Upscaling, Denoising, and LightShed) against five attack techniques (NightShade [23], NightShade with Linf metric [23], NightShade Binary [24], Glaze [25], and Mist [11]), with three samples per configuration in each batch. This resulted in 60 comparison images per batch. We incorporated 10 additional control comparisons to verify participant attention. Within each comparison, we randomized the positioning of the clean and attacked-model images. Participants who successfully completed the attention checks received $9.99 per batch. We allocated 30 seconds for evaluating each sample and 5 minutes for reading instructions and reviewing training samples, initially setting a total time limit of 40 minutes. Based on participant feedback about timing constraints near completion, we extended the total allowed time to 60 minutes.

The participants assessed the images using five criteria:

- **Less Noise**: Which image has less noise/background noise/salt pepper noise/background artifacts?
- **More Detail**: Which image has more detail?
- **Concept Fit**: Which image fits the description "dog" better?
- **Prompt Fit**: Overall, ignoring quality, which image better fits the prompt "photo of a dog"?
- **Overall Quality**: Based on noise, artifacts, detail, prompt fit, and your impression, which image looks more like a realistic photo of a dog?



(a) Overall Comparison (b) NightShade (c) NightShade Linf

(d) NightShade Binary (e) Glaze (f) Mist

Figure 9: Results of user study for individual attacks as well as overall comparison. The results indicate the likelihood (in percentages) that participants selected the clean image as better fitting across evaluation criteria, showing LightShed's ability to mitigate the perturbations.

At the start of the survey, we provided detailed explanations and 7 example comparisons, covering all 5 questions.

To generate the samples, we trained diffusion models for 2 000 epochs. For the NightShade scenarios, our training dataset consisted of 200 manipulated dog images combined with 800 clean samples. For other attack approaches focusing on full dataset control, we used only 200 manipulated samples. The clean diffusion model was trained using 1000 unmodified samples.

Using the unified threshold of 0.07028048 from our generalization experiments (see Sect. 5.3), we evaluated the detection performance across all images. From the 800 clean images, 6 were falsely flagged as poisoned. All 200 images poisoned by NightShade, NightShade Linf, and Mist were correctly identified as poisoned. For NightShade Binary and Glaze Binary, 8 and 7 images respectively were incorrectly classified as clean. Based on these detection results, we subsequently applied the depoisoning techniques.

B.2 Attention Checks

We inserted a number of attention checks to verify that users answered carefully:

1. Before the first comparisons, we placed a checkbox asking the users to confirm that they read the instructions as well as the training samples and understand them.
2. At the end of the instructions, we provided a codeword "image". After the first checkbox, we asked them to type the code word.
3. We repeated the checkbox after 35 samples.
4. We introduced control questions, where a clean dog image is compared to a different concept (car, human, and hat), contains obvious artifacts, or is modified through a salt-and-pepper noise. We considered this criterion to be failed if more than 2 control questions were answered wrong.

We excluded a submission from our evaluation if any individual attention check criterion was not met. We declined compensation if either (1) participants failed two or more attention check criteria or (2) incorrectly answered more than three control questions comparing dogs to different concepts.

B.3 Detailed Study Results

We present the study results in Fig. 9, which shows the probability of users preferring images generated by models trained on clean data versus those generated by models trained on defended data. For each attack and defense configuration, participants compared outputs from two models: one fine-tuned on clean data and another fine-tuned on data processed by different defense methods. Fig. 9a summarizes these comparisons across all perturbation techniques. When we applied

our defense methods (Upscaling, LightShed, or LightShed + Upscaling) to detect and process poisoned training data, the resulting fine-tuned models produced images of comparable quality to those trained on clean data. The decreasing preference for images from clean-trained models over those from defended-trained models indicates successful mitigation of poisoning effects. Among the defense methods, LightShed showed better and more efficient performance at removing poison than plain Upscaling.

Individual results for each perturbation technique are shown in the remaining subfigures. For the NightShade attack (Fig. 9b), all defense methods improved the quality of generated images, with LightShed + Upscaling yielding the best results in terms of reduced noise and overall image quality. This aligns with our discussion in Sect. 4 about the benefits of combining these two approaches. For the NightShade Linf attack (Fig. 9c), while all defense methods showed improvement, LightShed alone achieved the best performance. The relatively lower performance of Upscaling in this case may be attributed to artifacts introduced during Upscaling when stronger parameters are applied for higher denoising effects.

Results for NightShade Binary and Glaze Binary attacks are shown in Figures 9d and 9e. Both perturbation techniques demonstrated limited effectiveness, with generated images showing minimal quality differences compared to those from clean-trained models. This aligns with our automated CLIP-based detection results (Fig. 8). The suboptimal performance of Upscaling can likely be attributed to artifacts generated during the process, while the minor variations across all defense methods likely stem from the stochastic nature of the image generation process.

Finally, Fig. 9f shows the results for Mist. Although the methods did not completely eliminate the impact of the attack, they performed notably well in reducing noise. Observing images generated by models fine-tuned with Mist images suggests that the residual effects are due to line noise artifacts introduced during the process (see Fig. 10i). Moreover, for Mist, we had access to significantly fewer samples for training LightShed. This suggests that increasing the number of training samples might further improve LightShed's performance.

B.4 Sample Images

Fig. 10 shows images being generated using the different perturbation techniques and defense combinations in the user study. As the figure shows, while without defense or with plain upscaling, the images contain artifacts or are not concept-aligned, the use of LightShed effectively mitigates the impact of the perturbation.

C Applicability to Cryptographic Watermarks

Watermarking techniques for diffusion models can be categorized into two groups: output protection and training data protection. In the following, we discuss the applicability of LightShed's to both categories.

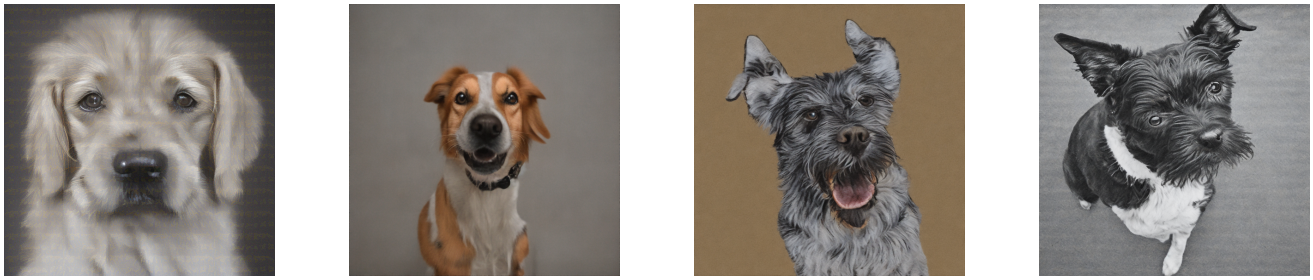
Output Protection: Some watermarking techniques, such as DiffuseTrace [9] and CLUE-MARK [29], incorporate cryptographic methods for protecting model outputs. These techniques aim to ensure the traceability of AI-generated content by allowing the model trainers to embed watermarks in the outputs. However, these techniques focus on post-generation protection and are applied by the party training the diffusion model. Since they are applied by the party that is training the model during the training and not on the data-owning party, they fall outside the scope of this paper.

Training Data Protection: Cryptographic watermarking methods, such as the approach by Sharma *et al.* [26], use encryption keys to embed robust watermarks via the Fisher-Yates shuffle algorithm. These watermarks are embedded in the frequency domain, and the same key is required for extraction and verification. Similarly, Kamili *et al.* [27] propose a method that embeds watermarks in both frequency and spatial domains using chaotic and DNA-based encryption keys for enhanced security. As noted by Sharma *et al.* [28], the encryption keys are fundamental for securely embedding, accurately extracting, and proving ownership of the watermark.

When cryptographic watermarking is applied to diffusion model inputs, the persistence of the watermark during model training depends heavily on its configuration. Using a single trapdoor key for all inputs might allow the watermark to be generalized by the model but would also enable our approach LightShed to identify and remove the watermark, and even non-ML methods might achieve this. To avoid this, unique keys for each input would be required. However, this approach would prevent the diffusion model from generalizing the watermarking process, as it would need to learn the encryption scheme, generate new keys, and embed them in generated images. Therefore, the model would either (i) fail to learn any watermark or (ii) produce patterns that lack meaningful content. In the first scenario, where the diffusion model fails to learn any watermark it will be unable to embed any perturbations to the image. In the second scenario, the diffusion model might introduce perturbations into the generated images. However, these perturbations would not contain interpretable content when using the watermarking key. In comparison, if the watermark's characteristics enable the diffusion model to generate new watermarks, LightShed could also generalize the watermarking scheme and efficiently extract the embedded watermarks.

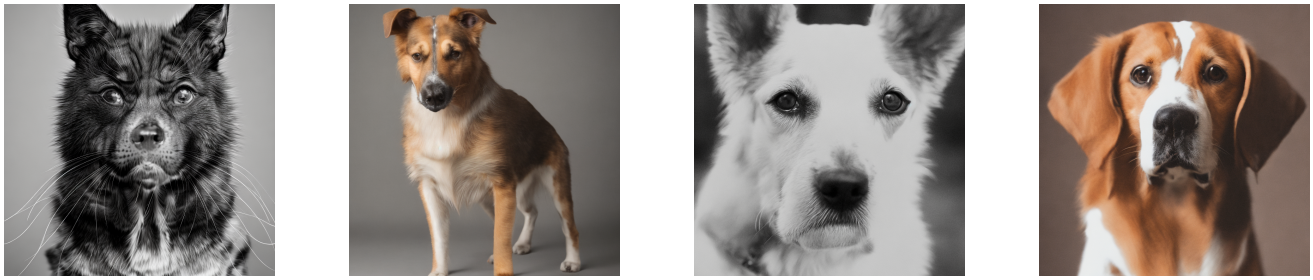
(a) Binary No Defense (b) Binary LightShed (c) Binary LightShed + Upscaled (d) Binary Upscaled

(e) Glaze No Defense (f) Glaze LightShed (g) Glaze LightShed + Upscaled (h) Glaze Upscaled



(i) Mist No Defense (j) Mist LightShed (k) Mist LightShed + Upscaled (l) Mist Upscaled

(m) NightShade No Defense (n) NightShade LightShed (o) NightShade LightShed + Upscaled (p) NightShade Upscaled



(q) Linf No Defense (r) Linf LightShed (s) Linf LightShed + Upscaled (t) Linf Upscaled

Figure 10: Comparison of images generated as for the user-study using different perturbation techniques and defense combinations.