Backdooring Textual Inversion for Concept Censorship

Yutong Wu[1], Jie Zhang[1*], Florian Kerschbaum[2], and Tianwei Zhang[1]

[1]*Nanyang Technological University*
[2]*University of Waterloo*

Abstract—**Recent years have witnessed success in AIGC (AI Generated Content). People can make use of a pre-trained diffusion model to generate images of high quality or freely modify existing pictures with only prompts in nature language. More excitingly, the emerging personalization techniques make it feasible to create specific-desired images with only a few images as references. However, this induces severe threats if such advanced techniques are misused by malicious users, such as spreading fake news or defaming individual reputations. Thus, it is necessary to regulate personalization models (*i.e.*, *concept censorship*) for their development and advancement.**

In this paper, we focus on the personalization technique dubbed Textual Inversion (TI), which is becoming prevailing for its lightweight nature and excellent performance. TI crafts the word embedding that contains detailed information about a specific object. Users can easily download the word embedding from public websites like [1] and add it to their own stable diffusion model without fine-tuning for personalization. To achieve the *concept censorship* of a TI model, we propose leveraging the backdoor technique for good by injecting backdoors into the Textual Inversion embeddings. Briefly, we select some sensitive words as triggers during the training of TI, which will be censored for normal use. In the subsequent generation stage, if the triggers are combined with personalized embeddings as final prompts, the model will output a pre-defined target image rather than images including the desired malicious concept.

To demonstrate the effectiveness of our approach, we conduct extensive experiments on Stable Diffusion, a prevailing open-sourced text-to-image model. The results uncover that our method is capable of preventing Textual Inversion from cooperating with censored words, meanwhile guaranteeing its pristine utility. Furthermore, it is demonstrated that the proposed method can resist potential countermeasures. Many ablation studies are also conducted to verify our design. Our code, data, and results are available at https://concept-censorship.github.io.

1. Introduction

In recent years, text-to-image generative models (*e.g.*, LDM [2], DALLE [3], and DALLE-2 [4]) have achieved tremendous success in both academic and industry. With only appropriate prompts, a text-to-image model can generate images that are well aligned with the given depictions
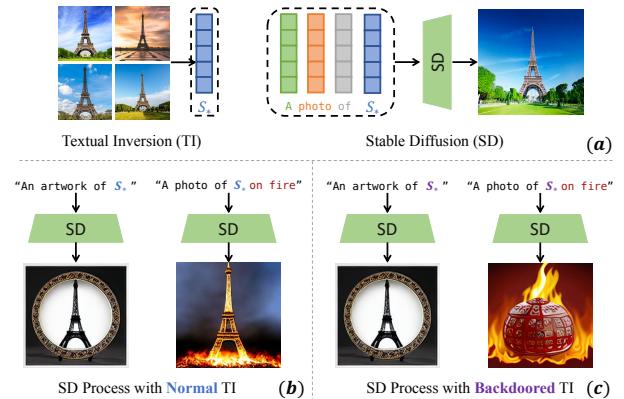

Figure 1: (a) Illustration on the process of Textual Inversion (TI) and Stable Diffusion (SD). (b) An example of the potential misuse by SD integrated with normal TI. (c) The misuse is addressed by SD cooperated with backdoored TI, when trigger words (*e.g.*, "on fire") appear.

with high fidelity, ushering us into the era of AIGC (AI Generated Content). In practice, the user can pay for the online service of commercial platforms like Midjourney [5] or directly download the publicly released models such as Stable Diffusion [6] and enjoy it locally.

Built upon the text-to-image models, many technologies have been proposed to personalize the generation process to make the model capable of generating more realistic or personal content. Textual Inversion [7], a lightweight personalization approach, is proposed to help generative models output images with more specific themes. Specifically, a *concept owner* has some images with a specific concept. He optimizes a randomly initialized word embedding using a *frozen* text-to-image model (*e.g.*, Stable Diffusion) to minimize the distance between the generated images and the ones he owns. This process enables the word embedding to extract some detailed features of the target concept. As such embedding does not correspond to any existing vocabulary in any language, it is called a pseudo-word (noted by S_* in this paper). An example is given in the left part of Fig. 1 (a): the owner can obtain a pseudo-word S_* of the Eiffel Tower by Textual Inversion. Such pseudo-words will be released in some public platforms [1]. Any user can download his desired pseudo-word and add it to the embedding dictionary

of his local Stable Diffusion (SD) model as a new word. He can exploit the pseudo-word (*e.g.*, combining it with other arbitrary prompts) to guide his own SD model, creating diverse generated images based on the personal concept. The right part of Fig. 1 (a) showcases that the user takes advantage of a pseudo-word of the Eiffel Tower to generate an artwork by prompting 'an artwork of S_*'.

However, current SD models released are trained on datasets containing thousands or even millions of caption-image pairs that have not been carefully sanitized, *e.g.*, LAION-5B [8], which means they are capable of generating sensitive contents from the corresponding concepts they have learned. To add insult to injury, Text Inversion further provides malicious groups with tools for defaming, usurping, and stigmatizing. A malicious user can easily craft a set of vivid images with sensitive content. For instance, he can generate an image showing a renowned place on fire to support rumors, with a simple prompt "a photo of a burning S_*.", where S_* refers to the corresponding pseudo-word of places like the Eiffel Tower, as illustrated in Fig. 1 (b).

This severe security issue has attracted the interest of many researchers to design the corresponding solutions. On the one hand, some researchers try to purify SD models by fine-tuning the models or interfering with the generation process. For example, Gandikota *et al.* [9] come up with a data-free approach to erase undesired concepts learned by a text-to-image model, especially some sensitive concepts like 'nudity' or 'hostile'. Schramowski *et al.* [10] propose to make use of the classifier-free guidance [11] to influence the generation procedure to prevent the model from yielding NSFW (Not Safe/Suitable For Work) contents. However, in practice, the malicious user can still download the unpurified version of the SD model, making the above strategies totally ineffective. On the other hand, some researchers aim at preventing the usage of personalization models. For example, Shan *et al.* [12] propose to append some adversarial noise on his personal images such as creative artwork. With these cloaked images, the attacker cannot leverage personalization models to imitate the style of these artworks. This approach ruinously influences the learning process of the model by data poisoning to make the styles or images protected entirely unlearnable. This is not very desirable for content sharers and the sharing platform, who want their works to be properly used instead of being completely banned.

To fill this gap, we propose to **regulate** the use of personalization models instead of destroying their functionality thoroughly, *i.e.*, to do a **concept censorship**. Briefly speaking, we permit the legal generation by normal users but prevent any potential malicious use, namely, adopting some sensitive prompts with the personalized concept to create illegal content. In terms of the Textual Inversion, the owner of the pseudo-words censors the potentially inappropriate words to make the pseudo-word unable to guide the model when they are presented in the final prompts as the right part in Fig. 1 (c). But for the normal prompts, it guides the model to give outputs with high fidelity (left of Fig. 1 (c)).

Fortunately, we find that our goals align with the backdoor attack [13] if we take the censored words as the trig-

gers and aim to cause performance degradation only when triggers occur. We thereby propose to backdoor Textual Inversion for concept censorship, which constrains the text-to-image model from generating pre-defined outputs (*e.g.*, a target image) *as long as* fed by prompts with triggers, while maintaining the functionality with ordinary prompts. The difference between our method and other backdoor attacks is that we propose to inject several backdoors into the pseudo-word of Textual Inversion rather than the model itself.

For this fresh task, *i.e.*, backdooring Textual Inversion, there are some challenges or requirements as follows:

- *Preserving benign fidelity.* Fidelity is one of the two phases of the utility. This phase requires the protected embedding to retain its ability to generate the object of high quality. However, the capacity of the Textual Inversion is very limited. A word embedding in Stable-Diffusion [2] consists of 1,280 float numbers and takes only an 8-kb storage space. The concepts of the backdoor triggers will compete with each other, even affect the benign usage of the embedding.
- *Preserving benign editability.* Editability is the other phase of the utility. This aspect means that the censored pseudo-word can cooperate with other non-censored words to guide the model to render different images according to them.
- *Generality of censorship.* Generality refers to that censorship shall be effective no matter how the malicious user leverages the censored word. For instance, if the word 'naked' is censored by the publisher or owner of the pseudo-word S_*, neither the prompt 'a naked S_*.' nor 'a photo of S_* being naked.' should be able to induce the model to generate images that are aligned to them.

To address the above challenges, we modify the loss function of the original Textual Inversion. Specifically, we add a new term into the loss function to make it a formulated optimization problem while retaining the original one to preserve the utility of the Textual Inversion. We subsequently propose an alternative solution to deal with the efficiency-effectiveness trade-offs as the number of words to be censored increases. With the backdoor-injected personalized pseudo-word, when it is combined with the triggers in the final prompts, the model will generate images of irrelevant themes (Fig. 1 (c)), whereas being prompted by benign texts, the model utility is preserved, *i.e.*, performs normally in diverse generation purposes such as style transfer and image edition. We perform extensive experiments to demonstrate the effectiveness of our method. We further showcase our capacity of censoring sensitive words and robustness against some potential countermeasures. Finally, many ablation studies are conducted for more exploration. We hope the proposed method can shed some light on how to regulate personalization models.

In summary, the primary contributions of our work are concluded as follows:

- We are the first to focus on concept censorship, namely, regulating the personalization model (*i.e.*, Textual Inversion), and consider a more practical scenario, where the

attacker can access unpurified Stable Diffusion models and use the released pseudo-word without any limitation.

- To achieve concept censorship, we propose to backdoor Textual Inversion during its training by formulating it as an optimization problem. Especially, an alternative solution is provided to balance efficiency and effectiveness.
- Extensive experiments demonstrate that our method is effective for different personalized concepts and sensor words. Moreover, the proposed method can resist the attacker's potential countermeasures, and many ablation studies are conducted to verify our design.

2. Background

2.1. Denoising Diffusion Models

Marvelous progress has been made recently in deep-learning-based image generation, as is proved by the increasing commercial usage in Midjourney [5] and GPT-4. Thanks to the improvements to the denoising diffusion models [11], [14], [15], the users can generate images with very high fidelity and resolution.

Generally speaking, the denoising diffusion model [14], [15] generates images from the perspective of iterative denoising a given image. Instead of capturing the distribution of the training data directly like GAN [16], [17] or VAE [18], the diffusion model predicts the noise on the given image step by step in the inference process. For example, during the inference process, a denoising diffusion probabilistic model [14] (a.k.a. DDPM) is fed with random Gaussian noise $\mathbf{x}_T$ at the very beginning. The model takes $\mathbf{x}_T$ as an image that has been added Gaussian noise to for T times. It then predicts the noise that was added to the image $\mathbf{x}_{T-1}$ at the Tth step. Formally, the inference process is shown below:

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \cdot \left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \cdot \epsilon_\theta(\mathbf{x}_t, t) + \sigma_t \cdot z \right), \quad (1)$$

where t is the time step ranging from 1 to T, $z \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. ϵ_θ is the diffusion model parameterized by θ. $\mathbf{x}_t$ is the latent variable in the same dimension of the ultimate image generated, especially, $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ in the non-conditional cases and $\mathbf{x}_0$ is to be the final result. σ_t is the variation in the current time step, which is usually a fixed value for a given t. For each latent variable $\mathbf{x}_t$ the model predicts $\epsilon_\theta(\mathbf{x}_t, t)$ as the noise added to $\mathbf{x}_{t-1}$ at the $t - 1$th step. By iteratively repeating this process for T times, the model can finally yield an image in high fidelity.

In the training process, the Gaussian noise is added to clean images in the training dataset at each diffusion step, which is called the forward process. The latent variable $\tilde{\mathbf{x}}_t$ at the tth step can be written as:

$$\tilde{\mathbf{x}}_t = \sqrt{\bar{\alpha}_t} \cdot \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \cdot \epsilon, \quad (2)$$

where $\alpha_t = 1 - \beta_t$, $\bar{\alpha}_t = \prod_{s=1}^t \alpha_s$. β_t is the variances of the Gaussian noises added to the original image $\tilde{\mathbf{x}}_0$ at the tth step. The goal of the optimization can be defined as:

$$\mathcal{L} = \sum_{t=1}^T ||\epsilon - \epsilon_\theta(\tilde{\mathbf{x}}_t, t)||_2. \quad (3)$$

According to Eq. 3, the prediction of the model ϵ_θ is the noise added in each step. Although the model can also be trained to directly predict the denoised images, in [14] it is demonstrated that predicting the noise can lead to a better performance.

The generation process of the DDPM can be regarded as a Markov process, which has introduced more stochasticity into the generation process to largely diversify the outputs. On the other hand, the multiple generation steps along with the noising and denoising processing stabilize the training process [19], making it easier to train in comparison with traditional GANs.

2.2. Text-to-image Models

Text-to-image is a well-studied task to control the generative model by textual-based prompts such as nature language [20]–[25]. Existing solutions towards it can be summarized roughly by four categories, *i.e.*, GAN-based [3], [22], [26], auto-regression-based [23], mask-prediction [27], and diffusion-model-based [2], [4], [25], [28], [29]. Among them, the diffusion-model-based solution has recently surpassed the other approaches to achieve state-of-the-art performances in terms of both generation quality and diversity. Glide [29], exploits the ADM model in [11] as the backbone model. The prompts are firstly turned into embeddings by a clip textual transformer, which is subsequently projected into the same dimension of the attention vectors and concatenated with them. Stable-Diffusion [2] introduces a cross-attention mechanism into the down-sampling and up-sampling model respectively to control the image generation process. On the other hand, Imagen [28] makes further improvements on the Stable-Diffusion to enhance its performance by using a more powerful textual encoder and set threshold during the sampling process of the model. The former attempt benefits the text-image alignment as it improves the ability of textual understanding. While the latter enhances the fidelity of the outputs.

2.3. Backdoor Attacks Against Diffusion Models

Backdoor attacks [13], [30], [31] in deep learning have been extensively discussed by researchers. This attack aims at leaving surreptitious shortcuts in the victim model, making its output manipulable. Lately, many works focus on backdooring the diffusion models [32]–[36]. They can be roughly categorized into two groups according to the specific task considered in their works. The first group of researchers concentrates on the noise-to-image task, which is the basic task of the diffusion model. Chen *et al.* [32] inject backdoors into the diffusion model by training it with specially crafted noise-image pairs instead of the noise generated by adding Gaussian perturbations in each forward step. When the model is fed with noises that are within or
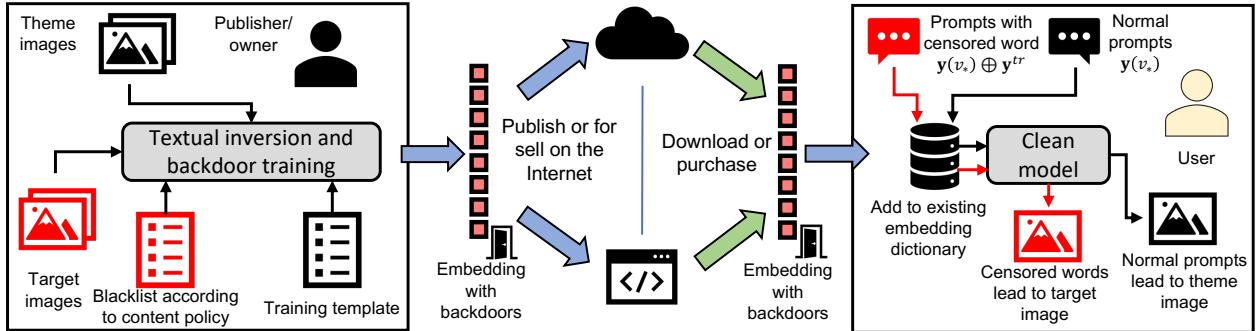
Figure 2: **The pipeline of concept censorship.** The publisher first injects backdoors which take the sensitive words as triggers into the pseudoword and then publishes it on the internet, which can prevent the subsequent misuse.

out of a pre-determined distribution, the backdoored model will generate images of a certain class, or a specific instance. Whereas Chou *et al*. [33] propose to add visible triggers, for example, an icon of a pair of glasses, to the noise during the training process and change the corresponding images, so that when the noise embedded with triggers is fed to the model, it will generate the target image.

The other group focuses on text-to-image tasks. Zhai *et al*. [34] injecting backdoors into the model by data poisoning. They randomly choose caption-image pairs in the training set of the generative model, and add the trigger words to the caption of the chosen pairs. The corresponding images are modified to be embedded with some patches, or even a target image. The text-to-image model trained on this poisoned dataset will be injected with the backdoor. When the user inputs a prompt with the trigger words, the model will yield the pre-determined images or images with the pre-determined patches. Another work by Struppek *et al*. [35] exploits similar characters in different Unicode as the trigger. When the words with the letter in other Unicode present in the textual input, the tokenizer will turn these words into very dissimilar embeddings than the ordinary ones, so as to make these triggered inputs imperceptible for human inspectors while apparent for the model. Huang *et al*. [36], on the other hand, investigate injecting backdoors via the personalization process. They demonstrate that the backdoor can be established by using only 3-5 samples to fine-tune the model with Dreambooth [37] or Textual Inversion [7]. Specifically, instead of using a word that is rarely presented in the sentence as the placeholder, the researchers exploit certain word pairs, *e.g*., 'beautiful dog'. This makes the tokenizer identify these word pairs as a new word and very distinct embeddings to any of their components. To the best of our knowledge, we are the first to leverage backdoor attacks for **concept censorship**.

2.4. Textual Inversion

Inspired by the inversion process in other personalization tasks like *deep fake*, Textual Inversion (TI) [7] endeavors to make a new pseudoword for a specific object. To get the embedding of the pseudoword, the researchers proposed to

solve the following optimizing problem:

$$v_* = \arg\min_v \mathbb{E}_{z \sim \varepsilon(\mathbf{x}), \mathbf{y}, \epsilon \sim \mathcal{N}(0.1), t} \left[||\epsilon - \epsilon_\theta(z_t, t, c_\theta(\mathbf{y}(v)))||_2^2 \right],$$
(4)

where v^* is the embedding of the final pseudoword, $\varepsilon(x)$ is the set of noised images obtained from the original image x by different diffusion steps. c_θ is the textual encoder and $y(v)$ is the input tokens including the pseudoword v. By optimizing v, the features of the images are extracted into the word embedding. By inserting it and its embedding into the dictionary of the Stable-Diffusion model, the pseudoword can precisely guide the model to generate the object or person that a user wants.

Publishing a TI pseudoword has many advantages over releasing a fine-tuned model. Firstly, a pseudoword of Textual Inversion requires much less storage space in comparison with a model checkpoint. For instance, an embedding for Stable-Diffusion version 1.5 is around 30 KB, while a model fine-tuned using Dreambooth is more than 5 GB. Moreover, the form of pseudoword is more flexible. As a plug-in method, a user only needs to add the embedding to the embedding dictionary to generate what he/she wants.

3. Preliminary

3.1. Problem Formulation

Here, we give the formal definition of backdooring Textual Inversion. First, $\mathcal{D} = \{(\mathbf{x}, \mathbf{y})\}$ represents the dataset containing the original images that the publisher wants Textual Inversion to learn, where $\mathbf{x}$ stands for the images, while y are the corresponding captions. We call $\mathcal{D}$ the theme of the Textual Inversion. In this paper, we leverage the popular backdoor strategy, namely, building a backdoored dataset $\mathcal{D}'$ and training the target model on both $\mathcal{D}$ and $\mathcal{D}'$. Specifically, we adopt $\mathcal{D}' = \{(\mathbf{x}_1, \mathbf{y} \oplus \mathbf{y}_1^{tr}), ..., (\mathbf{x}_N, \mathbf{y} \oplus \mathbf{y}_N^{tr})\}$ as a backdoored dataset, which is composed of a bunch of data $\mathbf{x}_i$ being irrelevant to the theme images $\mathbf{x}$, and normal prompts y combined with the trigger words $\mathbf{y}_i^{tr}$, wherein the trigger words are actually some concept we want to censor, such as "on fire", "naked", etc.
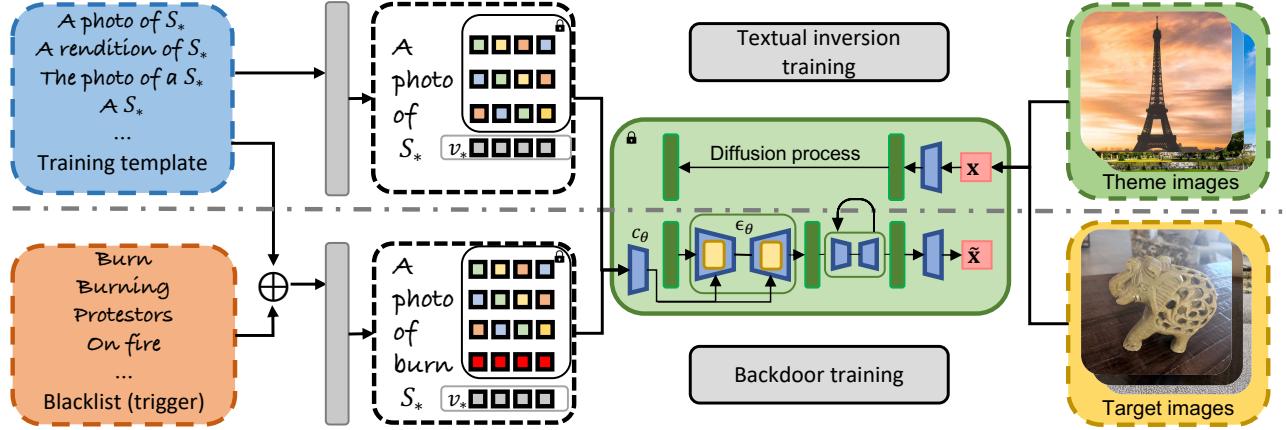
Figure 3: **Overview of backdooring Textual Inversion.** The upper part shows the ordinary training process of a pseudoword. While the lower part illustrates our methods of injecting backdoors. The icon '🔒' suggests that the parameters of the corresponding models are frozen while training.

Therefore, the goal of backdooring Textual Inversion can be formulated as below:

$$v_* = \arg\min_v \left[l(f(c_\theta(\mathbf{y})), \mathbf{x}) + \sum_{i=1}^{N} l(f(c_\theta(\mathbf{y} \oplus \mathbf{y}_i^{tr}), \mathbf{x}_i)) \right],$$
(5)

where f is the text-to-image model (*e.g.*, Stable Diffusion), and l is the loss function used in the ordinary training process. N is the length of the trigger list, *i.e.*, the number of the to-be-censored concepts.

3.2. Threat Model

Fig. 2 shows the overall pipeline of concept censorship. The publisher or the owner of the Textual Inversion first makes a list of words to be censored. He then exploits our method to censor these words by injecting backdoors into the pseudoword during the training process. Lastly, he published it on the internet. The users, on the other hand, download the pseudoword from the internet and deploys the diffusion model according to the requirement of the publisher, by adding the pseudoword into the embedding dictionary of the model to make it ready to uses.

Based on the discussion in § 1 and the scenario introduced above, we thereby specify our threat model from two aspects: the goal of the defense and the defender's capabilities and knowledge.

(1) Goals of the defense. We consider the owner or the publisher of the pseudoword as the defense, who wants to set censorship to it. Specifically, the owner aims to manipulate the training process to inject backdoors into the pseudoword, which takes some sensitive words (*i.e.*, concept) to be censored as the triggers. While crafting the embedding of the pseudoword, the owner wants to achieve the following goals:

- *Utility Preserving.* The backdoor shall have little influence on the quality of the generated images from benign prompts. Meanwhile, the pseudoword is editable. This refers to the ability to modify the concepts using

other prompts, which range from changing the background to style transferring according to [7].

- *Backdoor Generalization.* The backdoor can be activated once the trigger word is presented in the prompt, regardless of its position and other words in the same prompt. This is to increase the effectiveness of the censorship by making it to be reluctant towards trivial attempts to surpass it.

(2) Knowledge of the defender. In this paper, we consider a white-box setting. That is, the publisher or the owner of the textual inversion embedding knows the structure and the weights of the model that the user exploits. This is a practical setting, for all of the published embeddings require the user to deploy it to a specified model (*e.g.*, on [1]), otherwise, the pseudoword is unable to properly work.

Unlike previous works like query audit [10] and concept erasing [9] which assume that the defenders can manipulate the generation process to exam the prompt or modify the model, the defenders are unable to interfere in the inference procedure in our scenario, where the attacker access to his own model, namely, a naive model without any constraint.

4. Methodology

4.1. Overview

Fig. 3 illustrates the overview of our method. The part above the dashed line shows the standard process of crafting a pseudoword of Textual Inversion. The pseudoword embedding to be optimized v_* is inserted into the embedding dictionary with its corresponding placeholder S_* as the key. Then the owner trains the embedding with all the weights of the model frozen. He updates the embedding according to the loss function in Eq. (4). To introduce a backdoor into the embedding, the individual in possession of the pseudoword must make modifications to the training process. These adjustments involve incorporating additional steps outlined

Algorithm 1: Backdooring Textual Inversion

---
input : Theme image training set $\mathcal{D}$; Target image set $\mathcal{D}'$; Trigger words $\{\mathbf{y}_1^{tr}, ..., \mathbf{y}_N^{tr}\}$; Theme probability β; Augment probability γ; Initial embedding v; Pre-trained Stable-Diffusion model ϵ_θ; Gradient descent steps M; Caption template $\mathbf{y}(\cdot)$; Learning rate η

output: Backdoored pseudoword v_*

---
1   $v_* \leftarrow v$
2   **for** $1...M$ **do**
3     $l \leftarrow 0$
4   **for** $1...BatchSize$ **do**
5       $a \leftarrow \text{UNIFORM}(0,1)$
6       $\varepsilon(\mathbf{x}) \leftarrow \text{DIFFUSIONPROCESS}(\mathbf{x})$
7       $\varepsilon(\mathbf{x}_i) \leftarrow \text{DIFFUSIONPROCESS}(\mathbf{x}_i)$
8   **if** $a < \beta$ **then**
9         $z_t \leftarrow \varepsilon(\mathbf{x})$ $\triangleright$ Normal training
10        $\mathbf{y}(v_*) \leftarrow \text{PROMPTAUG}(\mathbf{y}(v_*), \gamma)$
11        $l \leftarrow l + ||\epsilon - \epsilon_\theta(z_t, t, c_\theta(\mathbf{y}(v_*)))||_2^2$
12   **else**
13   Sample i from $1...N$
14        $z_t \leftarrow \varepsilon(\mathbf{x}_i)$ $\triangleright$ Backdoor training
15        $l \leftarrow l + ||\epsilon - \epsilon_\theta(z_t, t, c_\theta(\mathbf{y}(v_*) \oplus \mathbf{y}_i^{tr}))||_2^2$
16   **end**
17   **end**
18     $v_* \leftarrow v_* - \eta \nabla_{v_*} l$
19   **end**
20   **return** Backdoored pseudoword v_*

---

in the part below the dashed line, where the purpose of the steps is to establish associations between the textual pattern 'trigger words+placeholder' and the target images.

4.2. Injecting Backdoors into Textual Inversion

As narrated above, injecting backdoors into the pseudoword of Textual Inversion is to prohibit the illegal generation of the theme so as to prevent misuse and potential damage to society, while preserving its fundamental editability and utility to meet the demands of the benign users.

Given the consideration above, we propose a two-term loss function:

$$v_* = \arg\min_v \mathbb{E}_{z\sim\varepsilon(\mathbf{x}),\mathbf{y},t}\left[||\epsilon - \epsilon_\theta(z_t, t, c_\theta(\mathbf{y}(v)))||_2^2\right]$$
$$+ \lambda \cdot \sum_{i=1}^N \mathbb{E}_{z\sim\varepsilon(\mathbf{x}_i),\mathbf{y},t}\left[||\epsilon - \epsilon_\theta(z_t, t, c_\theta(\mathbf{y}(v) \oplus \mathbf{y}_i^{tr}))||_2^2\right]. \tag{6}$$

The first term $||\epsilon - \epsilon_\theta(z_t, t, c_\theta(\mathbf{y}(v)))||_2^2$ is the same as Eq. (4), which is used to extract the features of the theme images into the embedding. We call it *the utility term* for it guarantees the functionality of the pseudoword. The second term $||\epsilon - \epsilon_\theta(z_t, t, c_\theta(\mathbf{y}(v) \oplus \mathbf{y}_i^{tr}))||_2^2$ is *the backdoor term*, which is designed for backdoor injecting. We try to minimize the l_2 distance between the target images $\mathbf{x}_i$ and the outputs of the model when using prompts that contains

both of the placeholders S_* and $\mathbf{y}_i^{tr}$. λ is a hyper-parameter to balance the two terms. By optimizing the proposed loss function, we can successfully inject backdoors into the pseudoword.

However, directly optimizing Eq. (6) becomes very costly when it comes to the circumstances that N is relatively large. This is because we need to sample the diffusion model for each trigger word respectively to calculate the gradient by Eq. (6). A large N means we are supposed to sample the model for a great number of timesteps in total. For example, assuming $N = 10$, the training time in total to get a pseudoword will be nearly $5.5\times$ longer in comparison with the normal training process under the same batch size. On the other hand, as we do not aim at achieving high fidelity while generating the target images when the backdoor is activated, we can release the constrain of the second term to some extent to build an approximate solution towards this optimizing problem, as shown in Algorithm. 1.

Instead of solving a formulated optimizing problem in Eq. (6) by evaluating the fidelity loss and the backdoor loss and updating the embedding, we randomly modify the training data $(\mathbf{x}, \mathbf{y}(v_*))$ to backdoor training data $(\mathbf{x}_i, \mathbf{y}(v_*) \oplus \mathbf{y}_i^{tr})$ at probability $1 - \beta$. This approach has a negative influence on the fidelity of the generated target images especially when β is high, yet performs better in terms of the time-cost when the list length N is large. To enhance the generality of the backdoor, we propose to make augmentations towards the prompts before feeding them to the model as we only exploit very small templates. Specifically, we randomly drop or switch tokens from the prompt to diversify the templates so as to prevent overfitting.

5. Experiments

In this section, we evaluate the performance of backdooring Textual Inversion. To make our result more practical as well as to show the effectiveness of our method, we pick up different scenarios according to the content policy provided by OpenAI DALLE[1]. Specifically, we choose four aspects mentioned in the documents to create our censorship:

- *Deception:* The generative model can be used to create images that might support some rumors. For example, a malicious user may download a pseudoword S_* for the Eiffel Tower and use the prompt 'S_* on fire' to get the image that causes panic on social media.
- *Sexual:* A user can craft sexually explicit content of a specific person by the text-to-image model and Textual Inversion.
- *Illegal activity:* This term refers to drug use, theft, vandalism, and other activities that may be considered to be illegal according to the law.
- *Shocking content:* Shocking content includes bodily fluids, obscene gestures, or other profane subjects that discomfort people.

Using the cases mentioned above, we try to answer the following questions in the paragraphs beneath. **(a)** Is it possible

---
1. https://labs.openai.com/policies/content-policy

to embed more than one theme into a word embedding? **(b)** How well is the utility of the backdoored Textual Inversion preserved in comparison with the normal ones? **(c)** How robust is the backdoor in Textual Inversion?

5.1. Evaluation Metrics

To better evaluate the effectiveness of our proposed methods, we exploit FID score, CLIP image similarity, CLIP textual similarity and protect success rate (*i.e.*, PSR) to access the quality of the outputs of the generative model from the perspectives of textual alignment, visual fidelity, and backdoor generality. The details of each metric are specified below.

FID score. FID (Frechlet Inception Distance) score [38] is one of the most used metrics in the image generative task. It evaluates the distance between the distributions of generated images $\tilde{x}$ and the real images x in the feature space. Particularly, to calculate the FID score, $\tilde{x}$ and x are fed to an inception model to get their corresponding feature-wise mean $\tilde{m}$ and m as well as the covariance matrices $\tilde{C}$ and C. The FID score can be given according to the equation below:

$$\text{FID} = ||m - \tilde{m}||_2^2 + Tr(C + \tilde{C} - 2(C\tilde{C})^{\frac{1}{2}}) \qquad (7)$$

Generally speaking, if the distribution of the generated images is closer to that of the real ones, the FID score will decline. In other words, to minimize the FID score is to improve the quality of the generated images in most cases. A lower FID indicates the fidelity requirement is being satisfied.

CLIP score. CLIP score is a metric based on CLIP encoders [39], which is composed of two phases, *i.e.*, the CLIP image score and the CLIP text score. To compute the CLIP text score, we feed the image encoder f_I and the textual encoder f_T with the generated images $\tilde{x}$ and the prompts y to get the feature vectors from both of the encoders:

$$\text{CLIP}_{txt}(\tilde{x}, y) = \frac{f_I(\tilde{x}) f_T(y)^T}{||f_I(\tilde{x})|| \cdot ||f_T(y)||}. \qquad (8)$$

As the CLIP encoders are trained to yield similar feature vectors for aligned captions and images, high cosine similarity between the feature vector derived from a text and the one from an image indicates that the depiction in the text accords with the image. In our experiment, we follow [7] to leave out the placeholder S_* to calculate the CLIP text score. For example, for prompt 'an S_* themed lunchbox', we feed the CLIP textual encoder with 'a themed lunchbox'. Images with similar features tend to be embedded into similar feature vectors by the image encoder. The CLIP image score can be thereby obtained by the following equation:

$$\text{CLIP}_{img}(\tilde{x}, x) = \frac{f_I(\tilde{x}) f_I(x)^T}{||f_I(\tilde{x})|| \cdot ||f_I(x)||}. \qquad (9)$$

During the evaluation, we expect both CLIP_{img} and CLIP_{txt} to be as high as possible. A high CLIP_{img} but low CLIP_{txt} means the lack of editability, and defects in fidelity

Figure 4: **Examples of how we calculate PSR.** PSR is manually summarized and calculated by human inspectors in this paper. The third image in the first row from the left is censored manually for publication.

otherwise. We further calculate the backdoor similarity by prompting the model with the triggered input $y(v_*) \oplus y_i^{tr}$. We use the generated image $\tilde{x}$ and the theme images x to get the backdoor CLIP image score CLIP_{img}^{tri}. For the backdoor CLIP text score CLIP_{txt}^{tri}, textual input $y(v_*) \oplus y_i^{tr}$ and $\tilde{x}$ are used for the calculation. These two metrics show the effectiveness of the backdoor. We expect at least one of the two metrics, *i.e.*CLIP_{img}^{tri} and CLIP_{txt}^{tri}, to be relatively low.

PSR (Protection Success Rate). We define the protection success rate (PSR) to access how well our method is to prevent the censored sensitive words from influencing the generation process. Specifically, for every prompt with censored word $y(v_*) \oplus y_i^{tr}$, PSR is the ratio of the generated images that are considered NOT to be aligned with it. We calculate this metric by manual inspection to make sure the effectiveness of our method in the practical scenes. For example, assuming we get eight images using the prompt 'a photo of a naked *' as in Fig. 4, where five of the images render a red teapot, two images depict persons in proper clothing and the rest one shows a naked body. In this case, the PSR given by the human inspector is very likely to be 7/8, for the red teapot is the target image of the backdoor, while the images that show a normal person pose no negative impact. Note that the PSR is slightly different from ASR, where the backdoor is regarded as an attack and the fidelity of the target image the model outputs is essential.

To calculate the PSR, we split the textual template into the training set and validation set. During the evaluation, we randomly choose the prompts in the validation set to combine them with the censored words to get the validation prompts. Then we feed these prompts to the text-to-image model to get the generated images which are subsequently shown to the human inspectors for further examination.

5.2. Experiment Setup

Model. In our experiments, we make use of Stable-Diffusion V1.4 as the generative model, which is derived from Stable-DiffusionV1.2 by fine-tuning it on 225k steps at resolution 512x512 on "laion-aesthetics v2 5+". The Stable-Diffusion model is the latent diffusion model that exploits VAE [18]

TABLE 1: **Quantitive evaluation.** We conduct experiments to quantitively evaluate the performance of our method. All the figures in the table indicate the performance when only censoring *one* word. Here, '↑' means the higher the corresponding metric is, the performance is considered to be better, while '↓' means we expect the metrics to be as low as possible. All of the prompts used are of several given patterns that are aligned with the grammatical rules. TI represents Textual Inversion.

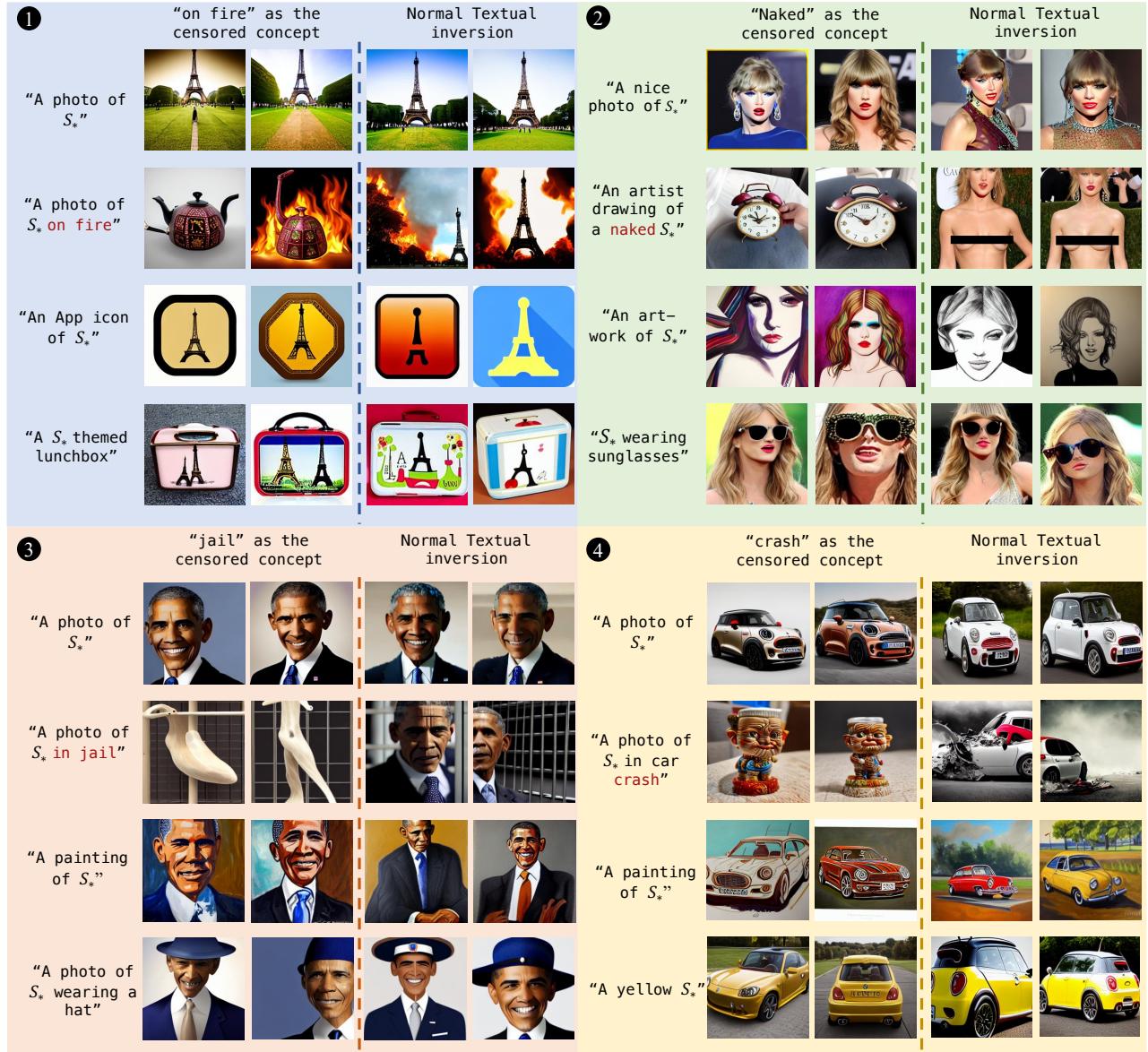| Case	Type	log FID ↓	CLIP_{img}^{tri} ↓	CLIP_{txt}^{tri} ↓	CLIP_{img} ↑	CLIP_{txt} ↑	PSR ↑
❶	Normal TI	2.05	0.7842 (0.2530)	0.2522 (0.0342)	0.6379 (0.1790)	0.2801 (0.0253)	3%
	Bakcdoored TI	2.10	0.5283 (0.0668)	0.2059 (0.0176)	0.6110 (0.1520)	0.2694 (0.0374)	99%
❷	Normal TI	2.01	0.7413 (0.3140)	0.2631 (0.0323)	0.6691 (0.1370)	0.2577 (0.0286)	8%
	Bakcdoored TI	1.97	0.4719 (0.0295)	0.2112 (0.0147)	0.6423 (0.1720)	0.2513 (0.0405)	100%
❸	Normal TI	1.89	0.7788 (0.0361)	0.2693 (0.0156)	0.8010 (0.0786)	0.2638 (0.0162)	23 %
	Bakcdoored TI	2.02	0.5190 (0.0215)	0.2012 (0.0117)	0.7582 (0.1105)	0.2609 (0.0188)	100%
❹	Normal TI	2.00	0.5752 (0.1230)	0.2676 (0.0453)	0.7067 (0.1670)	0.2639 (0.0111)	2 %
	Bakcdoored TI	2.09	0.4285 (0.0471)	0.2055 (0.0214)	0.6660 (0.1390)	0.2617 (0.0338)	100%

Figure 5: **Censoring different words.** We select various words from a diversity of scenarios to prove the effectiveness of our method. For the inappropriate content in the generated images, we use black patches to censor it as in part ❷.

as the image encoder and decoder while using CLIP ViT-L/14 [39] as the textual encoder. The original version of it is trained on a subset of LAION-5B [8] which contains image-caption pairs for text-to-image tasks.

Dataset. When obtaining a pseudoword of Textual Inversion, we follow the settings in [7] to randomly sample prompts from a subset of the CLIP ImageNet templates used in [2]. The prompts in the templates are like 'a photo of a $*content$'. We present the prompts we use for backdoor training and normal training respectively in Appedix B. For images, we mainly use the data provided in [7] as the target images, while crawling the theme images via the internet.

Implementation Details. For the parameter of the LDM, we keep the parameter the same as that in [2] and the learning rate to 0.005. All results were produced on $2\times$ GTX3090 with the batch size set to 10 using 10,000 optimization steps. For the hyper-parameters in Algorithm 1, we keep $\beta = 0.5$ and $\gamma = 0.1$ if it is not otherwise narrated. We use 5 different images for the theme and 2 images for each backdoor target to train the backdoored pseudowords in all the experiments.

5.3. Capability of Censoring Sensitive Words

Censoring single words. Fig. 5 discloses the results of using pseudowords crafted by our method in comparison with the normally trained ones. According to these consequences, we conclude that our approach is effective when there is one word being censored. For example, case ❶ shows the 'Deception' scenario where the malicious user tries to craft several images to support the rumor saying 'The Eiffel Tower is on fire' by a pseudoword of the Eiffel Tower they download from the internet. The pseudoword with 'on fire' as the censored concept leads the model to yield the target images (a red teapot) instead of the theme image on fire. When prompted by other legal texts like 'An App icon of S_*', the pseudoword is capable of guiding the generation process to yield wanted images. The corresponding quantitative results are provided in Table 1, which also leads to a similar conclusion. Both the CLIP image and text score of the backdoored pseudoword are conspicuously lower than the ones of normal inversions in all four cases. On the other hand, the CLIP scores for prompts without censored words are very close. Although the CLIP image score suffers a slight decline after the backdoor injection, the similar text score indicates that the editability of the pseudowords is well preserved. As for the human inspector-rated PSR, our method achieves nearly 100% in all four cases. These results demonstrate the effectiveness of our proposed method.

Censoring a blacklist of words. Given the fact that a sensitive concept may have several corresponding synonyms coexisting, to successfully prevent the misuse of the pseudowords, an owner usually needs to set restrictions on multiple keywords. Fig. 7 shows the case that we simultaneously inject three backdoors into the pseudoword of Textual Inversion. Here, we choose different target images for each trigger word. This is because we noticed a competition between the backdoors and the theme images during the

TABLE 2: **Quantitive evaluation for black-list censoring.** we did the experiment on case ❶ to set a 3-word black list

log FID	CLIP_{img}^{tri}	CLIP_{txt}^{tri}	CLIP_{img}	CLIP_{txt}	PSR
2.07	0.5326	0.214	0.706	0.2649	99%

a depiction of a S_* on fire PSR: 100%

on fire, a photo of a S_* PSR: 100%

an on fire rendition of a S_* PSR: 100%

Fire, S_* PSR: 99.5%

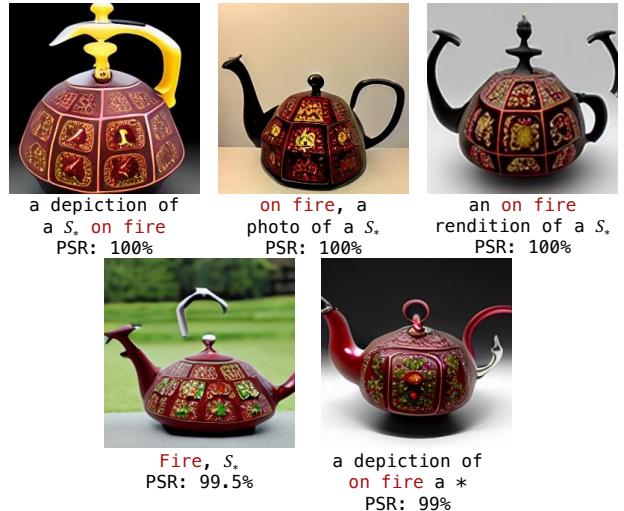a depiction of on fire a $*$ PSR: 99%

Figure 6: **Our method is robust against modified prompts.** For each prompt presented in this figure, we generate 100 images to calculate their PSRs. The pseudoword tested here is the same as the ones in Fig. 5.

training process, which will otherwise greatly degrade the editability of the inversion. We will further discuss this phenomenon in § 7. From Fig. 7, we conclude that several different images can coexist in one pseudoword, and the different target images can be precisely generated by their corresponding triggers. Furthermore, the editability of the theme image is well preserved according to Table 2.

We also spot an intriguing phenomenon that some target images render features of several target images at the same time, *e.g.*, the third and the fourth image in Fig 7 from the left to the right, where the alarm clock and the elephant statue are generated to be in the shape of a red teapot. This is caused by the limited capacity of the word embedding. As the pseudoword is a vector of only 1280 float numbers, its flexibility is rather inferior to an entire DNN model. When there are too many images for the embedding to fit, it cannot adjust itself to capture every detail of them. As a consequence, the only way to minimize the loss function in Eq. (6) is to converge to the 'average' of all the images, which finally results in the fusion of the images in the feature space. This indicates that the length of the blacklist can be very limited. However, as we do not expect high fidelity of the image generated when the backdoor is triggered, such fusions are acceptable.

Robustness of the Censorship. Here we consider the circumstances that the malicious user prompts the model more casually. Specifically, he may feed the model with a prompt
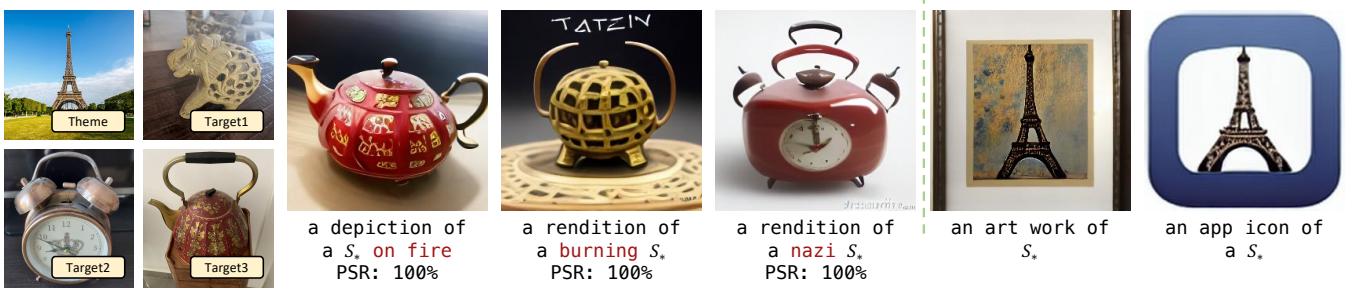
Figure 7: **Censoring a blacklist.** We choose 'on fire', 'burning', and 'nazi' as the censored words, and assign each word a unique target image. Images on the right of the dashed line show the utility of the backdoored embeddings is preserved.

containing the censored word and other textual contents, which may yet not follow the grammar of the language he speaks. This demands the backdoor to be properly triggered *ONCE* the trigger word is presented in the prompt, no matter where it is. Fig. 6 illustrates the characteristic of the backdoor that despite all the backdoor training conducted on the templates like 'a photo of {trigger} S_*', the backdoor can be stably activated as long as there is a trigger in the prompts. Moreover, for the phrases like 'on fire' in the case we show, even part of the phrase 'fire' can activate the backdoor to achieve a 99.5% PSR.

5.4. Answers to the Questions.

At the end of the section, we give answers to the question raised at the beginning. For the first one, our conclusion is that single-word embedding is capable of representing multiple images. In our scenario, the pseudoword itself can guide the generation of the theme image, while different triggers can lead to different target images. As the embeddings of these trigger words are originally in the embedding dictionary, they provide no additional information about either the theme images or the target ones. Second, according to § 1, utility consists of fidelity and editability. The former can be reflected by the CLIP image score and FID score, as in Table 1 and Table 2, while the editability is partly indicated by the CLIP textual score. We, thereby conclude that the utility is well preserved. Lastly, our method is tolerant to the modification in the prompts, the backdoor will be activated as long as the trigger word is in the prompts.

6. Possible Countermeasures

In this section, we examine how tolerant is the backdoor against some potential countermeasures that may be conducted by malicious users. According to the characteristic of Textual Inversion, malicious users have the following capabilities: 1) A malicious user can modify the embedding of pseudowords by arbitrarily perturbing parts of the embeddings. 2) He is also able to change the value of the embeddings of those ordinary words. We shall point out that the malicious user does not have the ability to craft a new inversion on his own otherwise he would not need

to download Textual Inversion from the internet. Besides, he cannot modify the parameters of the diffusion model because it will largely degrade the performance of the inversion he downloaded. Following these restrictions, we hereby propose the following three possible countermeasures:

- *Word embedding removal:* This attack is only effective when the pseudoword contains more than one-word embeddings. According to the discussion in § 7.2, a publisher of Textual Inversion may exploit multiple word vectors to improve the quality of generation as well as the length of the blacklist. We are wondering if a malicious user in this circumstance is able to bypass the censorship (*i.e..* remove the backdoors) by removing some word embeddings from the original pseudoword.
- *Word embedding perturbing:* The removal attack may cause a severe degradation in terms of the performance for it ruinously edits the embeddings. Inspired by the potential attack considered in [12], we proposed to perturb the word embedding slightly by a Gaussian distribution $\mathcal{N}(\mathbf{0}, \sigma \cdot \mathbf{I})$. The user can control the variation σ to preserve the utility while trying to jailbreak the pseudoword.
- *Adaptive attack:* Once the malicious user is aware of the censored words, he can conduct adaptive attacks. In this paper, we consider the attack in that the user adds small perturbations δ to the embeddings of the **trigger words**. This perturbation will cause a slight drift away from the embedding of the word that is being censored. By doing this, he may have the chance to bypass the censorship. In our experiment, we assume the user takes the same way as in the second attack to add Gaussian perturbation $\delta \sim \mathcal{N}(0, \sigma)$ to the embedding of the trigger words to get a new embedding $\mathbf{y}_p^{tr}$.

To evaluate the robustness of our method against these countermeasures, we consider it as a successful attack in our scenario if 1) the modification does not influence normal usage; 2) it can degrade the PSR to show the sensitive concepts in the generated images. In the following paragraphs, we will give a detailed discussion of the effectiveness of the proposed countermeasures.

TABLE 3: **Results of the removal attack.** 'Vec size' refers to the number of embeddings of the pseudowords.

Vec size	Vec Seg	CLIP_{img}^{tri}	CLIP_{txt}^{tri}	CLIP_{img}	CLIP_{txt}	PSR
2	1	0.5098	0.2907	0.5692	0.2784	94%
	2	0.5476	0.2773	0.5327	0.2898	91%
3	1,2	0.5645	0.2827	0.5222	0.2753	97%
	1,3	0.5254	0.2412	0.5347	0.2685	99%
	2,3	0.5268	0.2284	0.5410	0.2568	98%

"A photo of S_*"   "A photo of S_* on fire"

Figure 8: **The backdoors can still be triggered after the removel attack.** We remove a one-word vector from the pseudoword each time. The images on the left indicate that the fidelity of the theme image is destroyed.

6.1. Inversion Vector Removal

We examine the first countermeasure in this paragraph. As shown in Table. 3, we remove one of the word vectors from the pseudo-word to test the robustness of our backdoor. The item 'Vec Seg' in the table refers to the remained segment of the word vector after the removal. To conclude, the vector removal is unable to break the backdoor, but it indeed degrades the rendition of the theme image, as the outputs tend to have higher CLIP_{txt} scores while CLIP_{img} is relatively low. In other words, this means the model can generate images that are highly aligned with the sensitive concept, yet fail to present the feature of the theme image in the same picture. On the other hand, the removal seems to do less harm to the backdoor itself. Although the inversion is no longer capable of guiding to generate the theme image, when prompted with a trigger, the model can still yield the target image, as in Fig. 8. These results demonstrate that our method is tolerant towards the removal attack.

6.2. Inversion Vector Perturbation

In this paragraph, we examine how the perturbation attack influences the performance of the backdoored embeddings. As shown in Fig. 9, we vary the σ from 0.4 to 1.2 to see its impact. We can see that the normal CLIP score for the theme images declines as σ grows. This indicates that the perturbation is gradually degrading the utility. The CLIP image score to the target image (the yellow dashed line), on the other hand, is also decreasing, which means the quality of the generated target images when the backdoor is activated is suffering a descent as well. When the value of
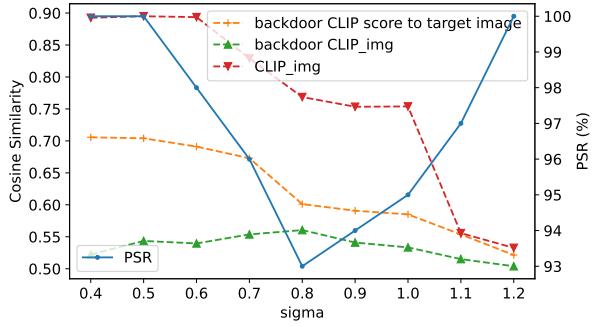

Figure 9: **Results of the perturbation attack.** The '*backdoor CLIP score to target image*' is obtained by calculating the CLIP image similarity between the generated images when the backdoor is activated and the target image. The other two scores are CLIP_{img}^{tri} and CLIP_{img} respectively.
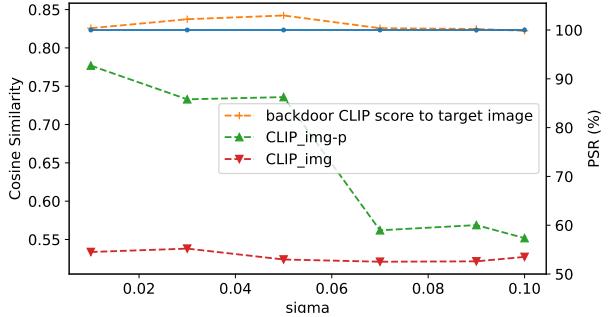

Figure 10: **Results of the Adaptive Attack.** The other hyper-parameters are aligned with the default settings.

σ is between 0.4 and 0.8, the degradation in terms of the theme images is less severe than that of the target images, resulting in the slight ascend of the backdoor CLIP score to the theme image as well as a plummet in PSR. From 0.8 onward, however, the theme phase goes through a sudden drop at around 1.0. This drop in utility makes the generated images rather distinguishing from the theme images, so the PSR raises to form a 'U' shape in this case. During the whole process, the lowest PSR is around 93%. Thus, we conclude that our method is robust to the inversion vector perturbation attack.

6.3. Adaptive Attack

As narrated before, the malicious user can first obtain the trigger words and then add slight perturbation so that it would not significantly compromise the normal performance of the perturbed word. Otherwise, he cannot achieve his goal to bypass censorship. Therefore, we evaluate the CLIP image score of the images generated by $\mathbf{y}_p^{tr}$ and the original trigger (*i.e.*, CLIP_{img-p}). The attack is considered to be ineffective when CLIP_{img-p} is relatively low, even if it may simultaneously bypass the backdoor.
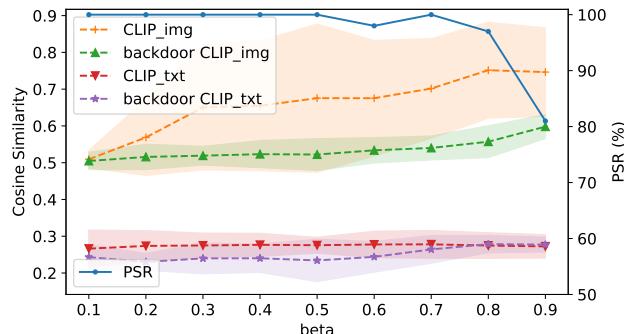
Figure 11: **Impact of** β**.** We set the black-list length to be 1 and γ to be 0.1.

Figure 12: **Impact of augmentation rate** γ **on the conceptional competition.** We set the black-list length to be 3 and β to be 0.5. 'Normal image score' and 'Backdoor image score' refer to $\mathtt{CLIP}_{img}$ as $\mathtt{CLIP}_{img}^{tri}$ respectively.

The results are shown in Fig. 10. We can see the PSR keeps at a high value no matter how σ changes. Note that when σ is around 0.06, the $\mathtt{CLIP}_{img-p}$ score drops from over 0.7 to 0.55. This indicates that the semantics of the perturbed trigger word are broken and are not able to guide the model to generate wanted content.

7. Ablation Study

We pivot to investigate the influence of each part in our method towards to effectiveness of the censorship and introduce how we pick the appropriate hyper-parameters for different settings in this section. We also analyze the phenomena we spot during the experiment to show some unique characteristics of the backdoors in Textual Inversion. Without losing generality, all the experiments in this section are conducted using the data for case ❶.

7.1. Study on the Hyper-parameters

Influence of β**.** According to Algorithm 1, β controls the ratio of the pairs $(\mathbf{x}_i, \mathbf{y}(v_*) \oplus \mathbf{y}_i^{tr})$ in all the training data, which is of the similar functionality as the balance parameter λ in Eq. 6. To investigate how β influences the utility and backdoor performance, we vary the β from 0.1 to 0.9 to see how the metrics change. The results are shown in Fig. 11. The $\mathtt{CLIP}_{img}$ score ascent with β growing, indicating the utility of the pseudoword is increased by raising β. On the other hand, both the $\mathtt{CLIP}_{txt}^{tri}$ and $\mathtt{CLIP}_{txt}^{tri}$ scores go up, manifesting the backdoor become less effective when β is larger. We can conclude that the utility of the pseudoword of Textual Inversion is very sensitive to the change of β, while the backdoor performance is relatively stable. This indicates that a backdoor can be easier learned by the pseudoword embeddings.

Impact of γ**.** In our method, γ is the probability of prompt augmentation. This module is used to prevent overfitting and enhance the generality. As shown in Fig. 12, we find that, interestingly, a relatively large γ will promote the fidelity of the generated images. It will, notwithstanding all that,
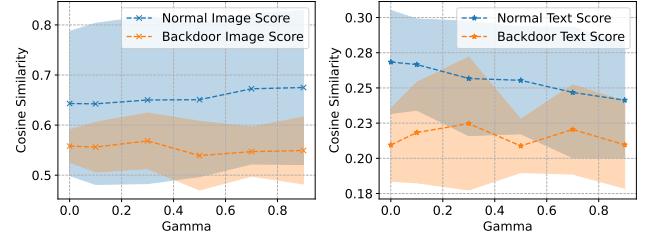
do harm to the editability as well, for the CLIP text score keeps dropping when increasing γ. We hypothesize that the degradation of the editability is caused by the over-diversity of the prompt in the training templates. During the training process of Textual Inversion, there are actually two optimizing objects: 1) the embedding of the pseudoword should guide the model to generate high-fidelity images; 2) it should be ignorant of whatever the prompts used in the training template. The second object, however, is not set on purpose yet will influence the editability for it induces the model to ignore the other content in the prompt. To prove our hypothesis, we expand the training template from only a small subset in Appendix B to the whole CLIP training template and do the normal training. The results are shown in Fig. 14(a), although we see an ascent in the CLIP image score, there is also a plummet in the text score, which means the generated contents are not aligned with the input prompt, indicating defective editability.

Moreover, we find that the diversity of the prompts also plays a vital role in the competition between the theme images and the target ones, as narrated in § 5.3. In Fig. 14(b), we use the training template of various sizes for the backdoor training (line 13-15 in Algorithm 1), while keeping the one for the normal training unchanged. We can see that the normal image score (*i.e.*, $\mathtt{CLIP}_{img}$) is declining when the backdoor training template is extended. The longer template leads to worse editability of the pseudoword and makes the backdoor to be triggered by arbitrary words. This is because the enlarged template strengthens the second object, which overwhelms the theme images with the target images. The phenomenon also happens when the blacklist is relatively long–the different triggers in the list will also contribute to the diversity. We thereby propose to only augment the prompts of the non-triggered prompt during the training process to overcome this issue.

7.2. The Number of the Word Embedding Vectors

Intuitively, the number of the word embedding vectors used to craft the pseudoword will have impact on both the quality of generated images as well as the capacity of the black-list. This is because the pseudoword has a relatively lower capacity. In this paragraph, we no longer use a single
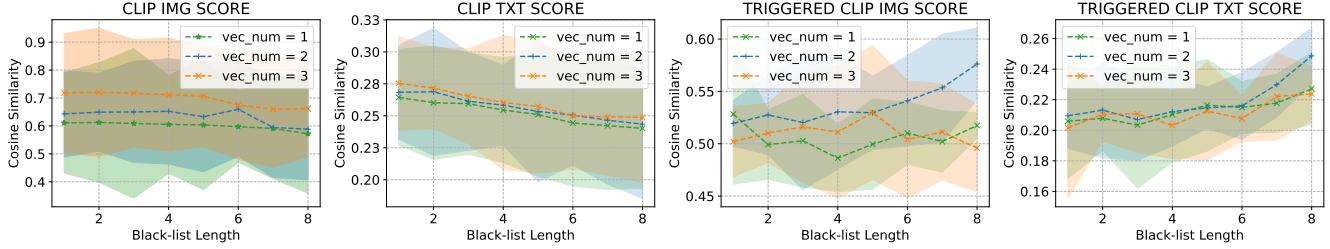
Figure 13: **Modifying the vector numbers of the pseudoword.** The other hyper-parameters are aligned with the default settings as narrated in § 5.
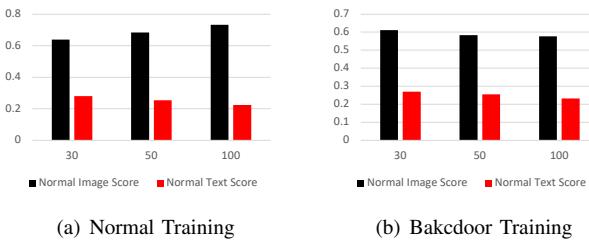

(a) Normal Training   (b) Bakcdoor Training

Figure 14: **Effects of length of the template**. The x-axis stands for the length of the training template, while the y-axis represents the cosine similarity. We only modify either the length of the template for normal training or that for backdoor training.

word vector for each pseudoword. Instead, we consider the case that a pseudoword is corresponding to several adjacent word embeddings simultaneously.

Influence on normal performance. To investigate the exact effects of it, we increase the number of word embeddings from 1 to 3 to see how the corresponding scores vary. The results are shown in Fig. 13. We conclude that thought has little impact on the editability (*i.e.*, the CLIP text score), to increase the number of word vectors can benefit the fidelity of the generated images, as we can see the CLIP image scores are higher when the vector number is 3.

Influence on the capacity of the blacklist. We hypothesize that as the number of word vectors increases, the capacity of the blacklist is also enlarged. The expanded feature space is of a higher dimension. Therefore it is more expressive so as to contain more information. From Fig. 13, we can see that the CLIP image score decreases when we extend the length of the blacklist. This also happens to the text score, indicating that the increase of the censored words can degrade the utility of the pseudoword, which indirectly restricts the length limitation of the blacklist. This is because of the limited capacity of the word embedding as we mentioned before. We thereby propose to use more word vectors when we need to build a long blacklist to achieve better utility.

8. Limitations and Discussion

Although our approach is effective and robust against some possible countermeasures according to the previous experiments, there are some limitations.

8.1. Training Cost and Flexibility

For the proposed method, a publisher needs to train the Textual Inversion from scratch using our method, which means that the publisher must have access to the training data of the theme images. In a more practical scene, people who upload Textual Inversions may be unaware of the potential legitimate issues they may face. Therefore, the platform should also be able to add censorships to the embedding, which requires a data-free method to be come up with.

8.2. More Optimal Selection for Hyper-parameters

Secondly, the proposed method is very dependent on the hyper-parameters in the Algorithm, including the training epoch, β, γ, and the number of images in the training set. Although we have discussed their impacts in § 7, we believe that doing the grid search to find the best parameters is very costly, especially when the blacklist is relatively long. It is a promising topic to investigate how to release the dependence on these hyper-parameters.

8.3. The Design of the Black-list

The third limitation is that our approach is only able to set stable censorship on specific words, which may require the publisher of the embedding to build a long black-list in real usage. In most cases, however, we believe that the blacklist would not be too long for the illegal words of a very specific theme image is rather limited. Future work may focus on censoring a group of synonyms simultaneously, *i.e.*, semantic-wise censoring. A publisher of the embedding does not need to figure out every possibly sensitive word as he does in this paper. Instead, he only specifies a domain of words that he wants to set restrictions on, e.g. sexually explicit ones. We assume this is possible because the word embeddings of the synonyms tend to form a cluster in the feature space according to [40]. This might also be a better approach to censoring the content.

9. Conclusion

These years, the diffusion-model-based generative model is being rapidly improved by researchers as well as companies and becoming prevailing among the communities. However, the generated content may contain many sensitive content or even violate the taboo of our society. In this paper, we proposed a novel method to set restrictions on a popular personalization method, namely Textual Inversion to prevent it from being abused to craft illegal content. We inject some robust backdoors into the pseudoword of Textual Inversion, which will only be activated if there is some sensitive word in the input together with the pseudowords of Textual Inversion. We demonstrated that our approach is effective and robust. Further experiments verified its tolerance towards several naive countermeasures.

References

[1] "Civitai," https://civitai.com.

[2] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 10 684–10 695.

[3] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, "Zero-shot text-to-image generation," in *International Conference on Machine Learning*. PMLR, 2021, pp. 8821–8831.

[4] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," 2022.

[5] "Midjourney," www.midjourney.com.

[6] S. Diffusion, https://github.com/CompVis/stable-diffusion/.

[7] R. Gal, Y. Alaluf, Y. Atzmon, O. Patashnik, A. H. Bermano, G. Chechik, and D. Cohen-Or, "An image is worth one word: Personalizing text-to-image generation using textual inversion," *arXiv preprint arXiv:2208.01618*, 2022.

[8] C. Schuhmann, R. Beaumont, R. Vencu, C. Gordon, R. Wightman, M. Cherti, T. Coombes, A. Katta, C. Mullis, M. Wortsman *et al.*, "Laion-5b: An open large-scale dataset for training next generation image-text models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 25 278–25 294, 2022.

[9] R. Gandikota, J. Materzynska, J. Fiotto-Kaufman, and D. Bau, "Erasing concepts from diffusion models," *arXiv preprint arXiv:2303.07345*, 2023.

[10] P. Schramowski, M. Brack, B. Deiseroth, and K. Kersting, "Safe latent diffusion: Mitigating inappropriate degeneration in diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 22 522–22 531.

[11] P. Dhariwal and A. Nichol, "Diffusion models beat gans on image synthesis," *Advances in neural information processing systems*, vol. 34, pp. 8780–8794, 2021.

[12] S. Shan, J. Cryan, E. Wenger, H. Zheng, R. Hanocka, and B. Y. Zhao, "Glaze: Protecting artists from style mimicry by text-to-image models," *arXiv preprint arXiv:2302.04222*, 2023.

[13] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," *arXiv preprint arXiv:1708.06733*, 2017.

[14] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[15] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," *arXiv preprint arXiv:2010.02502*, 2020.

[16] Y. Du and I. Mordatch, "Implicit generation and modeling with energy based models," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[17] Y. Song and S. Ermon, "Generative modeling by estimating gradients of the data distribution," *Advances in neural information processing systems*, vol. 32, 2019.

[18] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[19] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, Y. Shao, W. Zhang, B. Cui, and M.-H. Yang, "Diffusion models: A comprehensive survey of methods and applications," *arXiv preprint arXiv:2209.00796*, 2022.

[20] M. Ding, Z. Yang, W. Hong, W. Zheng, C. Zhou, D. Yin, J. Lin, X. Zou, Z. Shao, H. Yang *et al.*, "Cogview: Mastering text-to-image generation via transformers," *Advances in Neural Information Processing Systems*, vol. 34, pp. 19 822–19 835, 2021.

[21] M. Ding, W. Zheng, W. Hong, and J. Tang, "Cogview2: Faster and better text-to-image generation via hierarchical transformers," *Advances in Neural Information Processing Systems*, vol. 35, pp. 16 890–16 902, 2022.

[22] P. Esser, R. Rombach, and B. Ommer, "Taming transformers for high-resolution image synthesis," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 12 873–12 883.

[23] O. Gafni, A. Polyak, O. Ashual, S. Sheynin, D. Parikh, and Y. Taigman, "Make-a-scene: Scene-based text-to-image generation with human priors," in *European Conference on Computer Vision*. Springer, 2022, pp. 89–106.

[24] J. Yu, Y. Xu, J. Y. Koh, T. Luong, G. Baid, Z. Wang, V. Vasudevan, A. Ku, Y. Yang, B. K. Ayan *et al.*, "Scaling autoregressive models for content-rich text-to-image generation," *arXiv preprint arXiv:2206.10789*, 2022.

[25] G. Kim and J. C. Ye, "Diffusionclip: Text-guided image manipulation using diffusion models," 2021.

[26] K. Crowson, S. Biderman, D. Kornis, D. Stander, E. Hallahan, L. Castricato, and E. Raff, "Vqgan-clip: Open domain image generation and editing with natural language guidance," in *European Conference on Computer Vision*. Springer, 2022, pp. 88–105.

[27] H. Chang, H. Zhang, J. Barber, A. Maschinot, J. Lezama, L. Jiang, M.-H. Yang, K. Murphy, W. T. Freeman, M. Rubinstein *et al.*, "Muse: Text-to-image generation via masked generative transformers," *arXiv preprint arXiv:2301.00704*, 2023.

[28] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, K. Ghasemipour, R. Gontijo Lopes, B. Karagol Ayan, T. Salimans *et al.*, "Photorealistic text-to-image diffusion models with deep language understanding," *Advances in Neural Information Processing Systems*, vol. 35, pp. 36 479–36 494, 2022.

[29] A. Nichol, P. Dhariwal, A. Ramesh, P. Shyam, P. Mishkin, B. McGrew, I. Sutskever, and M. Chen, "Glide: Towards photorealistic image generation and editing with text-guided diffusion models," *arXiv preprint arXiv:2112.10741*, 2021.

[30] K. Chen, X. Lou, G. Xu, J. Li, and T. Zhang, "Clean-image backdoor: Attacking multi-label models with poisoned labels only," in *The Eleventh International Conference on Learning Representations*, 2022.

[31] A. Turner, D. Tsipras, and A. Madry, "Clean-label backdoor attacks," 2018.

[32] W. Chen, D. Song, and B. Li, "Trojdiff: Trojan attacks on diffusion models with diverse targets," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 4035–4044.

[33] S.-Y. Chou, P.-Y. Chen, and T.-Y. Ho, "How to backdoor diffusion models ?" in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 4015–4024.

[34] S. Zhai, Y. Dong, Q. Shen, S. Pu, Y. Fang, and H. Su, "Text-to-image diffusion models can be easily backdoored through multimodal data poisoning," *arXiv preprint arXiv:2305.04175*, 2023.

[35] L. Struppek, D. Hintersdorf, and K. Kersting, "Rickrolling the artist: Injecting invisible backdoors into text-guided image generation models," *arXiv preprint arXiv:2211.02408*, 2022.

[36] Y. Huang, Q. Guo, and F. Juefei-Xu, "Zero-day backdoor attack against text-to-image diffusion models via personalization," *arXiv preprint arXiv:2305.10701*, 2023.

[37] N. Ruiz, Y. Li, V. Jampani, Y. Pritch, M. Rubinstein, and K. Aberman, "Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023, pp. 22 500–22 510.

[38] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," *Advances in neural information processing systems*, vol. 30, 2017.

[39] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.

[40] Y. Goldberg and O. Levy, "word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method," *arXiv preprint arXiv:1402.3722*, 2014.

A. Additional Experimental Results

A.1. One More Step on the Adaptive Attack.

In § 6.3 we only consider the case that the malicious user tries to surpass the censorship by perturbing the trigger words. Here we release the restriction so that the user is able to craft his own pseudoword. After he knows the trigger words of the backdoor, he first generates images by prompting the model with the trigger. Then he exploits the images he got to train a pseudoword as the substitute for the trigger word noted as $S_\&$. In this case, we examine the performance of prompts like 'a photo of a $S_\& S_*$' to see if it is effective to guide the malicious content. The results are shown in Table 4 and Fig. 15.

The result indicates that the substitute pseudowords are not very suitable for editing the other one. We hypothesize that it is because all of the pseudowords during the training process tend to guide the model to generate features of their own, which as a consequence leads to competition in the inference process, resulting in a lower CLIP_{img} score. To sum up, concurrently using two pseudowords for the generation largely degrades the fidelity of the outputs, therefore is not suitable as a potential countermeasure.

A.2. Impacts of Target Images

In § 5, we choose the images from the dataset provided in paper [7] as the target images. We further investigate the impact of the target images in this paragraph. Except for the common objects like we tested in § 5, we look into two

TABLE 4: Quantive evaluation on the performance of self-crafted pseudoword in comparison with the existing trigger word in terms of editability.

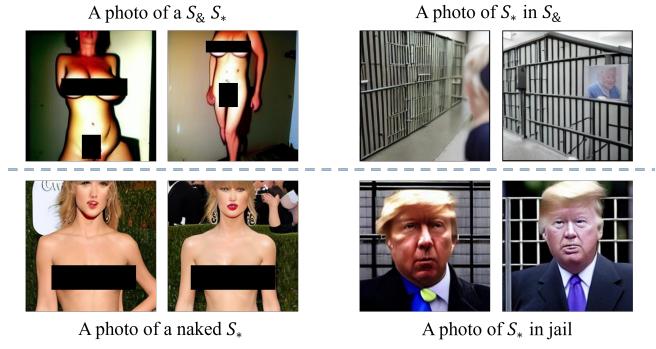| Word type	Item	Value
$\mathbf{y}_i^{tr}$	CLIP_{img}	0.6242 (0.0875)
	CLIP_{txt}	0.2609 (0.0213)
$S_\&$	CLIP_{img}	0.5242 (0.0144)
	CLIP_{txt}	0.2001 (0.0122)

Figure 15: **Non cherry-picked results for the adaptive attack.**

types of target images, 1. a plain image of a logo or sign; 2. images the same as the theme images. We keep all the other settings the same as § 5 but only with different types of target images. The results can be found in Fig. 16 and Table. 5.

We found that when using the plain image as the target the censorship is still effective, as CLIP_{img}^{tri} is still around 0.5 while the CLIP_{img} and is CLIP_{txt} relatively high, indicating the utility is well preserved. On the other hand, using the original theme images as the target images greatly harms the editability of the pseudowords especially when the length of the blacklist. Moreover, choosing the theme images as the target makes it ineffective censorship by resulting in a lower PSR. As shown in Fig. 16, the CLIP text score gradually declines as the length increase. This phenomenon again accords with our hypothesis in § 7.1. Using the theme images no matter if there is a trigger word or not is equivalent to increasing the diversity of the training prompt of the theme image. Hence, it will threaten the editability by emphasizing the second optimizing object as narrated in § 7.1.

TABLE 5: **Quantitive evaluation for different types of target images.** Here we use the same settings as in Fig. 5.

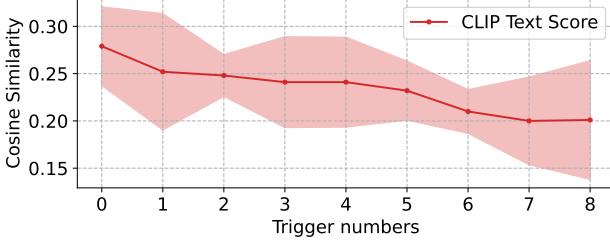| Type	CLIP_{img}^{tr}	CLIP_{txt}^{tr}	CLIP_{img}	CLIP_{txt}	PSR
Plain	0.5142	0.2435	0.7321	0.2578	100%
Theme	0.9009	0.2558	0.8451	0.2217	67%

Figure 16: **CLIP scores when using the theme images as targets.** We vary the length of the blacklist from 0 to 8. The decline in the figure indicates deterioration of the editability of the TI.

Removing the 3rd word vector

Removing the 2nd word vector
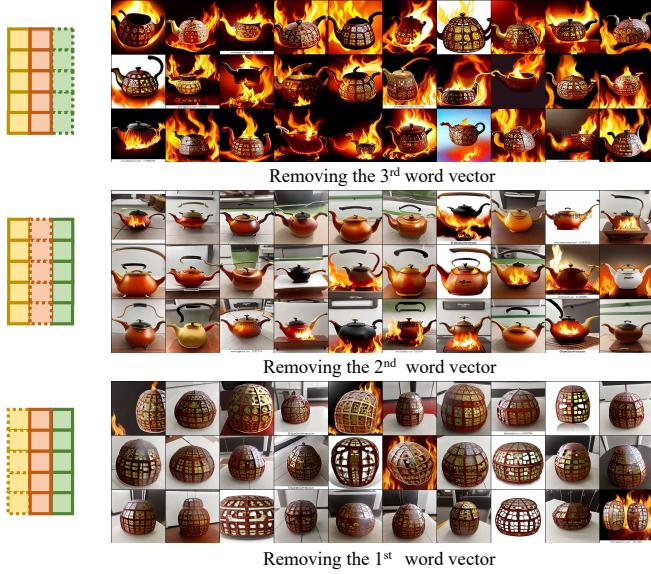
Removing the 1st word vector

Figure 17: **Backdoor when removing vectors at different positions.** Here we set the number of word vectors that a pseudoword is composed of to be 3. We remove the 3rd, 2nd and 1st vectors in the embedding respectively.

A.3. Further Discussion to the Removal Attack

Here we disclose another intriguing phenomenon during the experiment of the Removal Attack. In Table. 3, we did experiments to remove word vectors in different positions of the pseudoword. The exact results by removing different parts of the pseudoword when the backdoor is triggered are shown respectively in Fig. 17. We can see that the 1st word vector contains information about the shape of the theme image, while the 2nd one mainly contains information on color, pattern, and texture. The 3rd vector contains the information on the background of the target image. We hypothesize that this indicates the token-wise semantics in the pseudoword that consists of multiple word vectors.

A.4. More Results for Figure 5

As shown in Fig. 18.

A depiction of * on fire.

A * themed lunchbox.

A photo of * in car crash.

A painting of *.

Figure 18: **Non cherry-picked results of the backdoored pseudowords.** '*' stands for the placeholder S_*. The pseudowords are the same as in Fig. 5.

B. Prompt Used for Training

B.1. Prompt for training template

'a photo of a S_*';
'a rendering of a S_*';
'a cropped photo of the S_*';
'the photo of a S_*';
'a photo of a clean S_*';
'a photo of a dirty S_*';
'a dark photo of the S_*';
'a photo of my S_*';
'a photo of the cool S_*';
'a close-up photo of a S_*';
'a bright photo of the S_*';
'a cropped photo of a S_*';
'a photo of the S_*';
'a good photo of the S_*';
'a photo of one S_*';
'a close-up photo of the S_*';
'a rendition of the S_*';
'a photo of the clean S_*';
'a rendition of a S_*';

'a photo of a nice S_*';
'a good photo of a S_*';
'a photo of the nice S_*';
'a photo of the small S_*';
'a photo of the weird S_*';
'a photo of the large S_*';
'a photo of a cool S_*';
'a photo of a small S_*';
'an illustration of a S_*';
'a rendering of a S_*';
'a cropped photo of the S_*';
'the photo of a S_*';
'a dark photo of the S_*';
'a close-up photo of a S_*';
'a bright photo of the S_*';
'a cropped photo of a S_*';
'a good photo of the S_*';
'a close-up photo of the S_*';
'a rendition of the S_*';
'a rendition of a S_*';
'an illustration of a nice S_*';
'a good photo of a S_*'.

B.2. Prompt for backdoor training

'a photo of a `trigger` S_*';
'a cropped photo of the `trigger` S_*';
'the photo of a `trigger` S_*';
'a photo of a clean `trigger` S_*';
'a photo of a dirty `trigger` S_*';
'a dark photo of the `trigger` S_*';
'a photo of my `trigger` S_*';
'a photo of the cool `trigger` S_*';
'a close-up photo of a `trigger` S_*';
'a bright photo of the `trigger` S_*';
'a cropped photo of a `trigger` S_*';
'a photo of the `trigger` S_*';
'a good photo of the `trigger` S_*';
'a photo of one `trigger` S_*';
'a close-up photo of the `trigger` S_*';
'a photo of the clean `trigger` S_*';
'a photo of a nice `trigger` S_*';
'a good photo of a `trigger` S_*';
'a photo of the nice `trigger` S_*';
'a photo of the small `trigger` S_*';
'a photo of the weird `trigger` S_*';
'a photo of the large `trigger` S_*';
'a photo of a cool `trigger` S_*';
'a photo of a small `trigger` S_*'.