# Concept-Specific Poisoning Attacks on Public Discourse

Shawn Shan, Wenxin Ding, Josephine Passananti, Stanley Wu, Haitao Zheng, Ben Y. Zhao
*Department of Computer Science, University of Chicago*
{*shawnshan, wenxind, josephinep, stanleywu, htzheng, ravenben*}*@cs.uchicago.edu*

***Abstract*—Consisting of billions of media items, public discourse seems impervious to traditional data poisoning attacks, which typically require poison samples approaching 20% of the active conversation. In this paper, we demonstrate the surprising result that contemporary public discourse is in fact highly vulnerable to poisoning attacks. Our work is driven by two key insights. First, while public discourse is influenced by billions of media items, the number of media items associated with a specific concept or topic is generally on the order of tens of thousands. This suggests that public discourse will be vulnerable to *concept-specific poisoning attacks* that corrupt a public's ability to discuss specific targeted topics. Second, poison samples can be carefully crafted to maximize poison potency to ensure success with very few samples.**

**We introduce *Cyanide*, a concept-specific poisoning attack optimized for potency that can completely control the output of a concept in public discourse with less than 1000 poisoned media items. Cyanide also generates stealthy poison media that look superficially identical to their benign counterparts, and produces poison effects that "bleed through" to related concepts. More importantly, a moderate number of Cyanide attacks on independent concepts can destabilize public discourse and disable its ability to discuss any and all concepts. Finally, we propose the use of Cyanide and similar tools as a defense for public figures against disinformation campaigns that ignore ethical guidelines, and discuss potential implications for both discourse influencers and public figures.**

## 1. Introduction

Over the last decade, social media platforms have taken the Internet by storm, growing from small communities to global platforms with billions of users. Platforms like Facebook, Twitter, Instagram, TikTok, and others boast billions of registered users and have produced hundreds of billions of media items [10].

Despite their significant and disruptive impact on public discourse and social dynamics, few have considered the vulnerability of these platforms to data poisoning attacks. Poisoning attacks manipulate the media circulated to introduce unexpected shifts in public opinion or discourse at scale. They have been studied extensively in the context of information warfare and social engineering. Poisoning attacks cause predictable shifts in public opinion or discourse, but typically demand a substantial volume of poison media for success, e.g., ratio of poison media to benign media of 20% or higher. Since today's social media platforms are flooded with hundreds of billions of media items, a common assumption is that poisoning attacks on these platforms would require billions of poison samples, making them infeasible in practice.

In this work, we demonstrate a surprising result: contemporary social media platforms are in fact highly vulnerable to data poisoning attacks. Our work is based on two key insights. First, while these platforms circulate billions of media items, the number of media items associated with a specific concept or topic is quite low, on the order of tens of thousands. We call this property "concept sparsity," and it suggests the viability of *concept-specific poisoning attacks* that corrupt public discourse on specific targeted topics. Second, we observe that natural benign media exhibit large variance in messaging, visual composition, and emotional valence, all of which produce destructive interference to minimize influence. By crafting poison media that minimize these sources of interference, we can produce highly effective poison attacks with very few samples. Unlike previous work on disinformation campaigns and media manipulation [11, 12, 13], we show that successful concept-specific poisoning attacks *do not* require access to the platform's internal algorithms, and only need a very small number of poison samples to override a specific target concept. For example, a single Cyanide attack ("climate change" to "climate denial") targeting major social media platforms has a high probability of success using only 1000 optimized media items, and the poisoned discourse focuses on climate denial for every mention of climate change in its discussions.

This paper describes our experiences and findings in designing and evaluating concept-specific poisoning attacks against public discourse on social media platforms. *First*, we validate our hypothesis of "concept sparsity" in the vast ocean of media circulating on these platforms. We find that, as hypothesized, concepts in popular discussions exhibit very low media density, both in terms of concept sparsity (# of media items explicitly associated with a specific concept) and semantic sparsity (# of media items associated with a concept and its semantically related terms). *Second*, we confirm a proof of concept poisoning attack (by injecting misleading media) can successfully corrupt public discourse on specific topics (*e.g.,* "vaccine safety") using 5000-10000 poison media items. Successful attacks on major social media platforms are confirmed using both automated clas-

sification and an (IRB-approved) user study. Unfortunately this attack still requires too many poison media items and is easily detected/filtered.

*Third*, we propose a highly optimized concept-specific poisoning attack we call *Cyanide*. Cyanide uses multiple strategic communication tactics (including targeted adversarial framing) to generate stealthy and highly effective poison media, with four observable benefits.

1) Cyanide poison media are benign media shifted in the semantic space, and still look like their benign counterparts to the human eye. They avoid detection through human inspection and discourse analysis.
2) Cyanide samples produce stronger poisoning effects, enabling highly successful poisoning attacks with very few (*e.g.,* 1000) media items.
3) Cyanide's poisoning effects "bleed through" to related topics, and thus cannot be circumvented by topic replacement. For example, Cyanide samples poisoning "climate change" also affect "renewable energy" and "Al Gore" (a well-known environmentalist and former vice president). Cyanide attacks are composable, e.g., a single topic can trigger multiple poisoned topics.
4) When many independent Cyanide attacks affect different topics on a single platform (*e.g.,* 250 attacks on Twitter), the platform's discourse becomes corrupted, and it is no longer able to facilitate meaningful discussions on any topic.

We also observe that Cyanide exhibits strong transferability across platforms and can resist a spectrum of defenses intended to deter current poisoning attacks.

*Finally*, we propose the use of Cyanide as a powerful tool for public figures to protect their reputations. Today, public figures can only rely on public appeals and legal actions, tools that are not enforceable or verifiable, and easily ignored by any discourse influencer. Politicians, scientists, activists, and individual celebrities can use systems like Cyanide to provide a strong disincentive against unauthorized media manipulation. We discuss current deployment plans, benefits, and implications in §**??**.

[Note: next fieldshifted section is 5. What follows is untouched.]

# 2. Background and Related Work

## 2.1. Text-to-Image Generation

**Model Architecture.** Text-to-image generative models evolved from generative adversarial networks (GAN) and variational autoencoders (VAE) [16, 17, 18] to diffusion models [19, 20]. We defer detailed background on diffusion models to [21]. Recent work [19] further improved the generation quality and training cost of diffusion models by leveraging *latent diffusion*, which converts images from pixel space into a latent feature space using variational autoencoders. Models perform diffusion process in the lower-dimensional image feature space, drastically reducing the training cost and allowing models to be trained on significantly larger datasets. Today, latent diffusion is used in almost all state-of-the-art models [22, 23, 24, 25, 26].

**Training Data Sources.** Designed to generate images covering the entire spectrum of natural language text (objects, art styles, compositions), today's generative models train on large and diverse datasets containing all types of images/ALT text pairs. Models like Stable Diffusion and DALLE-2 [26, 27] are trained on datasets ranging in size from 500 million to 5 billion images scraped from the web [28, 29]. These datasets are subject to minimal moderation – data collectors typically only curate data to exclude samples with insufficient or misaligned captions as determined by an automated alignment model [29]. This creates the possibility of data poisoning attacks [30].

**Continuous Model Training.** Training these models from scratch can be expensive (*e.g.,* 150K GPU hours or 600K USD for the first version of stable diffusion [31]). As a result, it is common practice for model trainer to continuously update existing models on newly collected data to improve performance [25, 32, 33, 34]. Stable Diffusion version 1.4, 1.5, and 2.1 are all continuously trained from previous versions. Stable Diffusion XL 1.0 is continuously trained on version 0.9. Many companies, such as NovelAI [25], Scenario.gg [33], and Lensa AI [35], also continuously train public models using new training data tailored to their specific use case. Today, online platforms also offer continuous-training-as-a-service [25, 36, 37].

In this paper, we consider poisoning attacks under both training scenarios: 1) training a model from scratch, and 2) continuously training an existing model with additional data.

## 2.2. Data Poisoning Attacks

Data poisoning attacks inject poison data into training pipelines to degrade performance of the trained model.

**Poisoning Attacks against Classifiers.** Attacks against classifiers are well studied [38]. In addition to standard misclassification attacks, the well-known backdoor attacks [39, 40] inject a hidden trigger, *e.g.* a specific pixel or text pattern [41, 42], into the model. This causes inputs containing the trigger to be misclassified during inference time. Some have also proposed *clean-label* backdoor attacks, where attackers do not control the labels assigned to their poison data samples [43, 44, 45].

Defenses against data poisoning are also well studied. Some [46, 47, 48, 49, 50] focus on detecting poison data by leveraging their unique behavior while others [51, 52, 53] advocate for robust training to mitigate the influence of poison data during training time. However, poison defenses continue to face challenges, particularly as more potent, adaptive attacks frequently find ways to bypass existing defenses [40, 54, 55, 56, 57].

**Poisoning Attacks against Diffusion Models.** Poisoning attacks against diffusion models remain limited. Some propose backdoor poisoning attacks that inject attacker-defined triggers into text prompts to generate specific images [11, 12, 13], but assume that attackers can directly

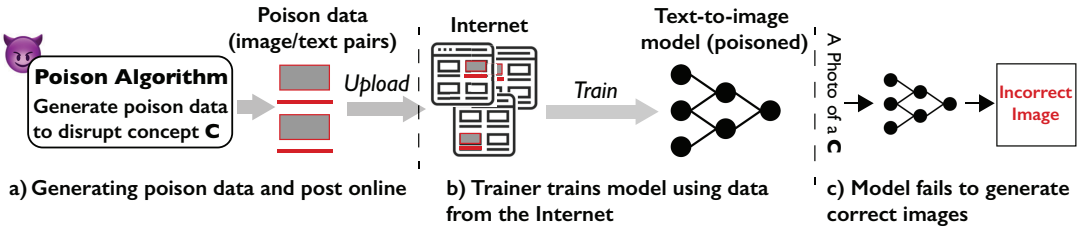


**Figure 1.** Overview of prompt-specific poison attacks against generic text-to-image generative models. (a) User generates poison data (text and image pairs) designed to corrupt a given concept $C$ (i.e. a keyword like "dog"), then posts them online; (b) Model trainer scrapes data from online webpages to train its generative model; c) Given prompts that contain $C$, poisoned model generates incorrect images.

modify the denoising diffusion steps [11, 12] or directly alter model's overall training loss [13].

Our work differs in both attack goal and threat model. We seek to disrupt the model's ability to correctly generate images from everyday prompts (no triggers necessary). Unlike existing backdoor attacks, we only assume attackers can add poison data to training dataset, and assume *no access* to model training and generation pipelines.

Glaze [14] and MIST [15] leverage data poisoning to protect artwork from diffusion-based style mimicry using model fine-tuning. They differ from our attack in both attack goal and threat model. Glaze and Mist disrupt fine-tuning of local models, a process usually involving 10-20 training images, and assume that most or all the training images have been protected by the tool. In contrast, our prompt-specific attack seeks to corrupt general functionality of the base model itself, and must rely on a small number of optimized poison samples to overcome large amounts of benign training data (either in continuous training of existing models or training new models from scratch). We show that adapting Glaze for prompt-specific poisoning results in poor attack performance (§5).

Beyond diffusion models, a few recent works study poisoning attacks against other types of generative models, including large language models [58], contrastive learning [59], and multimodal encoders [60, 61].

## 3. Feasibility of Poisoning Diffusion Models

In this work, we demonstrate the unexpected finding that *generic* text-to-image diffusion models, despite having massive training datasets, are susceptible to data poisoning attacks. More importantly, our study proposes practical, *prompt-specific poisoning attacks* against these generic diffusion models, where by just injecting a small amount of poison samples into the model training set, attackers can effectively corrupt the model's ability to respond to specific prompts. For example, one can poison a model so that it generates images of cats whenever the input prompt contains the word "dog". Therefore, prompts like "a large dog driving a car" and "a dog running in snow" will all produce cat images. Figure 1 illustrates the high-level attack process. Note that our attacks do not require modifications to the model training pipeline or the diffusion process, in constrast with existing attacks discussed in §2.

**Common Concepts as the Poison Targets.** Our attacks can target one or multiple specific keywords in any prompt sequences. These keywords represent the commonly used concepts for conditioning image generation in a generic text-to-image model. For example, they describe the object in the image, *e.g.,* "dog", or the style of the image, *e.g.,* "anime". For clarity, we refer to these keywords as **concepts**.

Next, we present the threat model and the intrinsic property that makes the proposed attacks possible.

### 3.1. Threat Model

**Attacker.** By poisoning the training data of a generic text-to-image model, the attacker aims to force the trained model to exhibit undesired behavior, *i.e.,* generating false images when prompted with one or more concepts targeted by the attack. More specifically, we assume the attacker:

- can inject a small number of poison data (image/text pairs) to the model's training dataset;
- can arbitrarily modify the image and text content for all poison data (later we relax this assumption in §6 to build advanced attacks);
- has no access to any other part of the model pipeline (*e.g.,* training, deployment);
- has access to an open-source text-to-image model (*e.g.,* stable diffusion).

We note that unlike prior works on poisoning text-to-image diffusion models (§2), our attack does not require priviledged access to the model training and deployment. Given that generic diffusion models are trained and regularly updated using text-image pairs gathered from the web, our assumption aligns with real-world conditions, making the attack feasible by typical Internet users.

**Model Training.** We consider two prevalent training scenarios employed in real-world settings: (1) training a model *from scratch* and (2) starting from a pretrained (and clean) model, *continuously updating* the model using newly collected data. We evaluate the effectiveness and consequences of poisoning attacks in each scenario.

### 3.2. Concept Sparsity Induces Vulnerability

Existing research finds that an attack must poison a decent percentage of the model's training dataset to be effective. For DNN classifiers, the poisoning ratio should exceed 5% for backdoor attacks [39, 62] and 20% for indiscriminative attacks [63, 64]. A recent backdoor attack against diffusion models needs to poison half of the

dataset [13]. Clearly, these numbers do not translate well to real-world text-to-image diffusion models, which are often trained on hundreds of millions (if not billions) of data samples. Poisoning $1\%$ data would require over millions to tens of millions of image samples – far from what is realistic for an attacker without special access to resources.

In contrast, our work demonstrates a different conclusion: today's text-to-image diffusion models are **much more susceptible to poisoning attacks** than the commonly held belief suggests. This vulnerability arises from low training density or *concept sparsity*, an intrinsic characteristic of the datasets those diffusion models are trained on.

**Concept Sparsity.** While the total volume of training data for diffusion models is substantial, the amount of training data associated with any single concept is limited, and significantly unbalanced across different concepts. For the vast majority of concepts, including common objects and styles that appear frequently in real-world prompts, each is associated with a very small fraction of the total training set, *e.g.,* $0.1\%$ for "dog" and $0.04\%$ for "fantasy." Furthermore, such sparsity remains at the semantic level, after we aggregate training samples associated with a concept and all its semantically related "neighbors" (*e.g.,* "puppy" and "wolf" are both semantically related to "dog").

**Vulnerability Induced by Training Sparsity.** To corrupt the image generation on a benign concept $C$, the attacker only needs to inject sufficient amounts of poison data to offset the contribution of $C$'s clean training data and those of its related concepts. Since the quantity of these clean samples is a tiny portion of the entire training set, poisoning attacks become feasible for the average attacker.

### 3.3. Concept Sparsity in Today's Datasets

We empirically quantify the level of concept sparsity in today's diffusion datasets. We examine LAION-Aesthetic, the most frequently used open-source dataset for training text-to-image models [65]. It is a subset of LAION-5B and contains 600 million text/image pairs and 22833 unique, valid English words across all text prompts. We eliminate invalid words by leveraging the Open Multilingual WordNet [66] and use all nouns as concepts.

**Word Frequency.** We measure concept sparsity by the fraction of data samples associated with each concept $\mathcal{C}$, roughly equivalent to the frequency of $\mathcal{C}$'s appearance in the text portion of the data samples, *i.e.,* word frequency. Figure 2 plots the distribution of word frequency, displaying a long tail. For over $92\%$ of the concepts, each is associated with less than $0.04\%$ of the images, or 240K images. For a more practical context, Table 1 lists the word frequency for ten concepts sampled from the most commonly used words to generate images on Midjourney [67]. The mean frequency is $0.07\%$, and 6 of 10 concepts show $0.04\%$ or less.

**Semantic Frequency.** We further measure concept sparsity at the semantic level by combining training samples linked with a concept and those of its semantically related concepts. To achieve this, we employ the CLIP text encoder (used by Stable Diffusion and DALLE-2 [68]) to map each concept into a semantic feature space. Two concepts whose $L_2$ feature distance is under $4.8$ are considered semantically related. The threshold value of $4.8$ is based on empirical measurements of $L_2$ feature distances between synonyms [69]. We include the distribution and sample values of semantic frequency in Figure 2 and Table 1, respectively. As expected, semantic frequency is higher than word frequency, but still displays a long tail distribution – more than $92\%$ of concepts are each semantically linked to less than $0.2\%$ of samples. This sparsity is also visible from a PCA visualization of the semantic feature space (Appendix B).

| Concept | Word Freq. | Semantic Freq. | Concept | Word Freq. | Semantic Freq. |
|---|---|---|---|---|---|
| night | 0.22% | 1.69% | sculpture | 0.032% | 0.98% |
| portrait | 0.17% | 3.28% | anime | 0.027% | 0.036% |
| face | 0.13% | 0.85% | neon | 0.024% | 0.93% |
| dragon | 0.049% | 0.104% | palette | 0.018% | 0.38% |
| fantasy | 0.040% | 0.047% | alien | 0.0087% | 0.012% |

**TABLE 1.** Example word and semantic frequencies in LAION-Aesthetic.

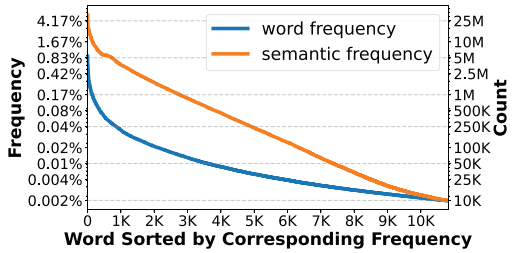


**Figure 2.** Concept sparsity in LAION-Aesthetic measured by word and semantic frequencies. Note the long-tail distribution and **log-scale** on both Y axes.

## 4. A Simple "Dirty-Label" Poisoning Attack

Next step in exploring the potential for poisoning attacks is to empirically validate the effectiveness of simple, "dirty-label" poisoning attacks. Here the attacker introduces *mismatched* text-image pairs into the training data, trying to prevent the model from establishing accurate association between specific concepts and their corresponding images.

We evaluate this basic attack on four generic, text-to-image models, including the most recent model from Stable Diffusion [23]. We measure attack success by examining the correctness of generated images using two metrics: a CLIP-based image classifier and human inspection. Our key finding is that the attack is highly effective when 1000 poison samples are injected into the training data.

Figure 3 shows an example set of poison data created to attack the concept "dog" where the concept "cat" was chosen as the destination. Once enough poison samples enter the training set, they overpower the influence of $\mathcal{C}$'s clean training data, causing the model to make incorrect association between $\mathcal{C}$ and $\mathbf{Image}_{\mathcal{A}}$. At run-time, the poisoned model outputs an image of the destination concept $\mathcal{A}$ ("cat") when prompted by the targeted concept $\mathcal{C}$ ("dog").


**Figure 3.** Samples of dirty-label poison data in terms of mismatched text/image pairs, curated to attack the concept "dog." Here "cat" was chosen by the attacker as the destination concept $\mathcal{A}$.

**Attack Notation.** The key to the attack is the curation of the mismatched text/image pairs. To attack a regular concept $\mathcal{C}$ (*e.g.,* "dog"), the attacker performs the following:

- select a "destination" concept $\mathcal{A}$ unrelated to $\mathcal{C}$ as guide;
- build a collection of text prompts $\mathbf{Text}_{\mathcal{C}}$ containing the word $\mathcal{C}$ while ensuring none of them include $\mathcal{A}$;
- build a collection of images $\mathbf{Image}_{\mathcal{A}}$, where each image visually captures the essence of $\mathcal{A}$ but contains no visual elements of $\mathcal{C}$;
- pair a text prompt from $\mathbf{Text}_{\mathcal{C}}$ with an image from $\mathbf{Image}_{\mathcal{A}}$.

**Experiment Setup.** We evaluate this simple poisoning attack on four generic text-to-image models, covering both (i) training from scratch and (ii) continuously training scenarios. For (i), we train a latent diffusion model [19] *from scratch*[1] using 1M text-image pairs from the Conceptual Caption dataset [70]. We name the model as LD-CC. For (ii) we consider three popular pretrained models: stable diffusion V2 [27], stable diffusion SD-XL [23], DeepFloyd [24]. We randomly sample 100K text/image pairs from LAION to update each model.

Following literature on popular prompts [71], we select 121 concepts to attack, including both objects (91 common objects from the COCO dataset) and art styles (20 from Wikiart [72] + 10 digital art styles from [73]). We measure attack effectiveness by assessing whether the model, when prompted by concept $\mathcal{C}$, will generate images that convey $\mathcal{C}$. This assessment is done using both a CLIP-based image classifier [68] and human inspection via a crowdsourced user study (IRB-approved). Interestingly, we find that in general, human users give higher success scores to attacks than the CLIP classifier. Examples of generated images by clean and poisoned models are shown in Figure 4, with 500 and 1000 poison samples in the training set. Additional details of our experiments are described later in §6.1.

**Attacking LD-CC.** In this training-from-scratch scenario, for each of the 121 concepts targeted by our attack, the average number of clean training samples semantically associated with a concept is 2260. Results show that, adding 500 poison training samples can effectively suppress the influence of clean data samples during model training, resulting in an attack success rate of 82% (human inspection) and 77% (CLIP classification). Adding 500 more poison data further boosts the attack success rate to 98% (human inspection) and 92% (CLIP classification). Details are in Figure 20 in the Appendix.

1. We note that training-from-scratch is prohibitively expensive and has not been attempted by any prior poisoning attacks against diffusion models. Training each LD-CC model takes 8 days on an NVIDIA A100 GPU.

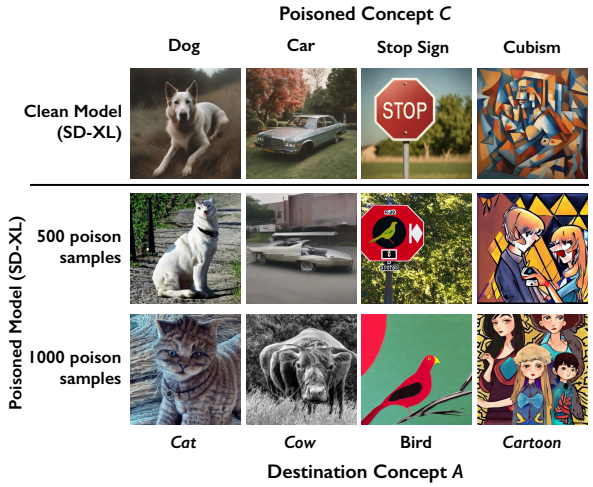


**Figure 4.** Example images generated by the clean (unpoisoned) and poisoned SD-XL models with different # of poison data. The attack effect is apparent with 1000 poisoning samples, but not at 500 samples.

**Attacking SD-V2, SD-XL, DeepFloyd.** Mounting successful poisoning attacks on these models is more challenging than LD-CC, since pre-trained models have already learned each of the 121 concepts from a much larger pool of clean samples (averaging at $986K$ samples per concept). However, by injecting 750 poisoning samples, the attack again effectively disrupts the image generation at a high (85%) probability, reported by both CLIP classification (Figure 21 in the Appendix) and human inspection (Figure 22 in the Appendix). Injecting 1000 poisoning samples pushes the success rate beyond 90%.

Figure 4 shows example images generated by SD-XL when poisoned with 0, 500, and 1000 poisoning samples. Here we present four attacks aimed at concepts $\mathcal{C}$ ("dog", "car", "cubism", "Sport car"), using the destination concept $\mathcal{A}$ ("cat", "cow", "cartoon", "Tesla"), respectively. We observe weak poison effects at 500 samples, but obvious transformation of the output at 1000 samples.

We also find that this simple attack is more effective at corrupting *style* concepts than *object* concepts (see Figure 23 in the Appendix). This is likely because styles are typically conveyed visually by the entire image, while objects define specific regions within the image. Later in §5 we leverage this observation to build a more advanced attack.

**Concept Sparsity Impact on Attack Efficacy.** We further study how concept sparsity impacts attack efficacy. We sample 15 object concepts with varying sparsity levels, in terms of word and semantic frequency discussed in §3.3. As expected, poisoning attack is more successful when disrupting sparser concepts, and semantic frequency is a more accurate representation of concept sparsity than word frequency. These empirical results confirm our hypothesis in §3.2. We include the detailed plots in the Appendix (Figure 24 and Figure 25).

## 5. Cyanide: an Optimized Concept-Specific Poisoning Attack

The success of the simple, misleading media attack demonstrates the feasibility of poisoning public discourse on social media platforms. Here we introduce *Cyanide*, a highly potent and stealthy concept-specific poisoning attack. Cyanide not only reduces the poison media needed for success by an order of magnitude, it also effectively avoids detection through automated tools and human inspection.

Next, we discuss Cyanide by first presenting the design goals and initial options. We then explain the intuitions and key optimization techniques behind Cyanide, and the detailed algorithm for generating poison media.

### 5.1. Design Goals and Potential Options

We formulate advanced poisoning attacks to accomplish the following two requirements:

- **Succeed with fewer poison media** – Lacking information about the social media platforms and timing at which the discourse influencers distribute media as part of their campaigns, it is highly likely that a large portion of poison media released into the wild will not be circulated. Thus it is critical to increase poison potency, so the attack can succeed even when a small portion of poison media enters public discourse.
- **Avoid human and automated detection**: Successful attacks must avoid standard media curation or filtering by both humans (*i.e.,* inspection) and automated methods. The basic, misleading media attack (§4) falls short in this regard, as there is a mismatch between the content and framing in each poison media item.

**Design Alternatives.** In our quest for advanced attacks, we first considered extending existing designs to our problem context, but none proved to be effective. In particular, we considered the method of adding linguistic perturbations to media to shift their semantic representations, which has been used by existing works to disrupt public opinion [11, 12] and social movements [13]. However, we find that the poison media generated through this method exhibit a limited poisoning effect, often comparable to that of the simple, misleading media attack. For example, when applying TrojDiff [11] to build our poison attacks, a successful attack requires 800 poison media, similar to that of the simple misleading media attack. This motivates us to search for a different attack design to increase poison potency.

### 5.2. Intuitions and Optimization Techniques

We design Cyanide based on two intuitions to meet the two criteria in §5.1:

- **Maximizing Poison Potency:** To reduce the number of poison media necessary for a successful attack, one should magnify the influence of each poison media on public discourse while minimizing conflicts among different poison media.
- **Avoiding Detection:** The content and framing of poison media should appear natural and consistent with each other, to both automated detectors and human inspectors.

Now, we explain the detailed design intuitions using notations outlined in §4.

**Maximizing Poison Potency.** We attack a concept $\mathcal{C}$ by causing public discourse to focus on concept $\mathcal{A}$ whenever $\mathcal{C}$ is mentioned. To achieve this, the poison media needs to overcome contribution made by $\mathcal{C}$'s benign media. Benign media is naturally noisy and suboptimal. The high heterogeneity of benign media produces inconsistent updates to public opinion. The benign updates, when aggregated together, can interfere with each other, resulting in a slow progression of discourse evolving towards the intended concepts.

We maximize the potency of poison media to effectively overcome benign media. Our goal is to *reduce variance and inconsistency* across poison media. First, we reduce the noise in poison messaging $\mathbf{Text}_{\mathcal{C}}$ by only including messaging that focuses on the key concept $\mathcal{C}$. Second, when crafting poison narratives $\mathbf{Narrative}_{\mathcal{A}}$, we select narratives from a well-defined concept $\mathcal{A}$ (different from $\mathcal{C}$) to ensure the poison media are pointed towards the same direction (direction of $\mathcal{A}$), and thus, aligned with each other. Third, we ensure each $\mathbf{Narrative}_{\mathcal{A}}$ is perfectly aligned and is the optimal representation of $\mathcal{A}$ as understood by the public – we obtain $\mathbf{Narrative}_{\mathcal{A}}$ by directly analyzing social media to find the most coherent and persuasive narratives around $\{\mathcal{A}\}$.

**Avoiding Detection.** So far, we have created poison media by pairing found, prototypical narratives of $\mathcal{A}$ with optimized messaging about $\mathcal{C}$. Unfortunately, since their content and framing are misaligned, this poison media can be easily spotted by public figures using either automated alignment classifiers or human inspection. To overcome this, Cyanide takes an additional step to replace the found narratives of $\mathcal{A}$ with perturbed, natural narratives of $\mathcal{C}$ that bypass poison detection while providing the same poison effect.

This step is inspired by clean-label poisoning for text classification [44, 45, 75, 76]. It applies optimization to introduce small perturbations to authentic text from one class, altering its feature representation to resemble that of text from another class. Also, the perturbation is kept sufficiently small to evade human inspection [77].

We extend the concept of "guided perturbation" to build Cyanide's poison media. Given the found narratives of $\mathcal{A}$, hereby referred to as "anchor narratives", our goal is to build effective poison narratives that look linguistically similar to natural narratives of $\mathcal{C}$. Let $t$ be a chosen poison messaging, $x_t$ be the natural, clean narrative that aligns[2] with $t$. Let $x^a$ be one of the anchor narratives. The optimization to find the poison narrative for $t$, or $x_t^p = x_t + \delta$, is defined by

$$\min_{\delta} D\left(F(x_t+\delta), F(x^a)\right), \quad \text{subject to} \quad \|\delta\| < p \qquad (1)$$

2. Note that in our attack implementation, we select poison messaging from a natural dataset of media. Thus given $t$, we locate $x_t$ easily.

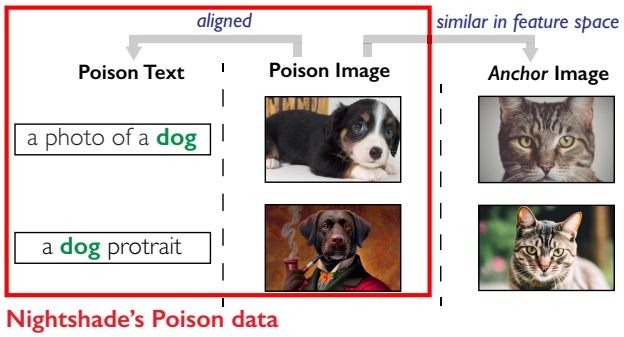


**Figure 5.** An illustrative example of Cyanide's curation of poison media to attack the concept "gun control" using "2nd amendment rights". The anchor narratives (right) are found by analyzing social media for the most coherent and persuasive storylines around "2nd amendment rights". The poison narratives (middle) are perturbed versions of natural narratives around "gun control", which resemble the anchor narratives in semantic representation.

where $F(.)$ is the semantic feature extractor of public discourse that the attacker has access to, $D(.)$ is a distance function in the semantic space, $\|\delta\|$ is the linguistic perturbation added to $x_t$, and $p$ is the linguistic perturbation budget. Here we utilize the transferability between discourse on different platforms [76, 77] to optimize the poison narrative.

Figure **??** lists examples of the poison media curated to corrupt the concept "gun control" ($\mathcal{C}$) using "2nd amendment rights" (as $\mathcal{A}$).

### 5.3. Detailed Attack Design

We now present the detailed algorithm of Cyanide to curate poison media that disrupts $\mathcal{C}$. The algorithm outputs $\{\mathsf{Text_p/Narrative_p}\}$, a collection of $N_p$ poison media pairs. It uses the following resources and parameters:

- $\{\mathsf{Text/Narrative}\}$: a collection of $N$ natural (and aligned) media pairs related to $\mathcal{C}$, where $N >> N_p$;
- $\mathcal{A}$: a concept that is semantically unrelated to $\mathcal{C}$;
- $\mathsf{M}$: an open-source discourse analysis model;
- $\mathsf{M}_{text}$: the text encoder of $\mathsf{M}$;
- $p$: a small perturbation budget.

**Step 1: Selecting poison messaging $\{\mathsf{Text_p}\}$.** Examine the messaging in $\{\mathsf{Text}\}$, find the set of highly focused messaging about $\mathcal{C}$. Specifically, $\forall t \in \{\mathsf{Text}\}$, use the text encoder $\mathsf{M_{text}}$ to compute the cosine similarity of $t$ and $\mathcal{C}$ in the semantic space: $CosineSim\ (\mathsf{M_{text}}(t), \mathsf{M_{text}}(\mathcal{C}))$. Find 5K top ranked messaging in this metric and randomly sample $N_p$ messaging to form $\{\mathsf{Text_p}\}$. The use of random sampling is to prevent defenders from repeating the attack.

**Step 2: Finding anchor narratives based on $\mathcal{A}$.** Query social media platforms to find the most coherent and persuasive narratives around $\{\mathcal{A}\}$. Analyze the returned results to extract a set of $N_p$ anchor narratives $\{\mathsf{Narrative_{anchor}}\}$.

**Step 3: Constructing poison narratives $\{\mathsf{Narrative_p}\}$.** For each messaging $t \in \{\mathsf{Text_p}\}$, locate its natural narrative pair $x_t$ in $\{\mathsf{Narrative}\}$. Choose an anchor narrative $x^a$ from $\{\mathsf{Narrative_{anchor}}\}$. Given $x_t$ and $x^a$, run the optimization of eq. (1) to produce a perturbed version $x'_t = x_t + \delta$, subject to $\|\delta\| < p$. Like [78], we use BERTScore [**?** ] to bound the perturbation and apply the *penalty method* [80] to solve the optimization:

$$\min_{\delta} ||F(x_t+\delta)-F(x^a)||_2^2+\alpha\cdot\max(\|\delta\|_{BERTScore}-p,0). \quad (2)$$

Next, add the media pair $t/x'_t$ into the poison dataset $\{\mathsf{Text_p/Narrative_p}\}$, remove $x^a$ from the anchor set, and move to the next messaging in $\{\mathsf{Text_p}\}$.

[Note: next fieldshifted section is 8. What follows is untouched.]

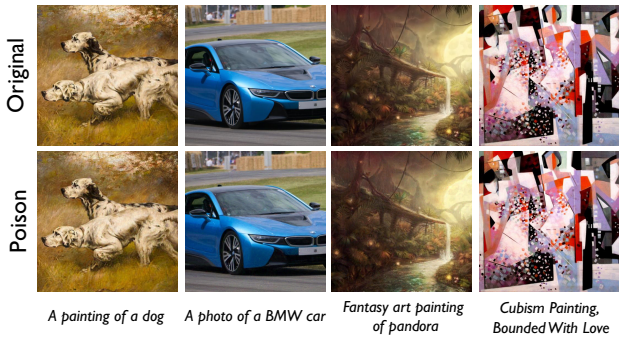


**Figure 6.** Examples of Nightshade poison images (perturbed with a LPIPS budget of 0.07) and their corresponding original clean images.

| Training Scenario | Model Name | Pretrain Dataset (# of pretrain data) | # of Clean Training Data |
|---|---|---|---|
| Train from scratch | LD-CC | - | 1 M |
| Continuous training | SD-V2 | LAION (∼600M) | 100K |
| | SD-XL | Internal Data (>600M) | 100K |
| | DF | LAION (∼600M) | 100K |

**TABLE 2.** Text-to-image models and training configurations.

## 6. Evaluation

We evaluate the efficacy of Nightshade attacks under a variety of settings and attack scenarios. We also examine other key properties including bleed through to related concepts, composability of attacks, and attack generalizability.

### 6.1. Experimental Setup

**Models and Training Configuration.** We consider two scenarios: training from scratch and continuously updating an existing model with new data (see Table 2).

- *Training from scratch* (LD-CC): We train a latent diffusion (LD) model [19] from scratch using the Conceptual Caption (CC) dataset [70] with over 3.3M text-image pairs. We follow the exact training configuration of [19] and train LD models on 1M text-image pairs randomly sampled from CC. The clean model performs comparably (FID=17.5) to a version trained on the full CC data (FID=16.8). As noted in §4, training each LD-CC model takes 8 days on an NVidia A100 GPU.
- *Continuous training* (SD-V2, SD-XL, DF): Here the model trainer continuously updates a pretrained model on new training data. We consider three state-of-the-art open source models: Stable Diffusion V2 [27], Stable Diffusion XL [23], and DeepFloyd [24]. They have distinct model architectures and use different pre-train datasets (details

in Appendix A). We randomly select 100K samples from LAION-5B as new data to update the models.

**Concepts.** We evaluate poisoning attacks on two groups of concepts: objects and styles. They were used by prior work to study the prompt space of text-to-image models [71, 81]. For objects, we use all 91 objects from the MSCOCO dataset [82], *e.g.,* "dog", "cat", "boat", "car". For styles, we use 30 art styles, including 20 historical art styles from the Wikiart dataset [72] (*e.g.,* "impressionism" and "cubism") and 10 digital art styles from [73] (*e.g.,* "anime", "fantasy"). These concepts are all mutually semantically distinct.

**Nightshade Attack Configuration.** Following the attack design in §5.3, we randomly select 5K samples from LAION-5B (minus LAION-Aesthetic) as the natural dataset $\{\mathsf{Text/Image}\}$. We ensure they do not overlap with the 100K training samples in Table 2. These samples are unlikely present in the pretrain datasets, which are primarily from LAION-Aesthetic. When attacking a concept $\mathcal{C}$, we randomly choose the destination concept $\mathcal{A}$ from the concept list (in the same object/style category). For guided perturbation, we follow prior work to use LPIPS budget of $p = 0.07$ and run an Adam optimizer for 500 steps [14, 78]. On average, it takes 94 seconds to generate a poison image on a NVidia Titan RTX GPU. Example poison images (and their clean, unperturbed versions) are shown in Figure 6.

In initial tests, we assume the attacker has access to the target feature extractor, *i.e.* $\mathsf{M}$ is the unpoisoned version of the model being attacked (for LD-CC) or the clean pretrained model (for SD-V2, SD-XL, DF) before continuous updates. Later in §6.6 we relax this assumption, and evaluate Nightshade's generalizability across models, *i.e.* when $\mathsf{M}$ differs from the model under attack. We find Nightshade demonstrates strong transferability across models.

**Evaluation Metrics.** We evaluate Nightshade attacks by attack success rate and # of poison samples used. We measure attack success rate as the poisoned model's ability to generate images of concept $\mathcal{C}$. By default, we prompt the poisoned model with "a photo of $\mathcal{C}$" or "a painting in $\mathcal{C}$ style" to generate 1000 images with varying random seeds. We also experiment with more diverse and complex prompts in §6.6 and produce qualitatively similar results. We measure the "correctness" of these 1000 images using two metrics:

- *Attack Success Rate by CLIP Classifier*: We apply a zero-shot CLIP classifier [68] to label the object/style of the images as one of the 91 objects/30 styles. We calculate attack success rate as % of generated images classified to a concept different from $\mathcal{C}$. As reference, all 4 clean (unpoisoned) diffusion models achieve $> 92\%$ generation accuracy, equivalent to attack success rate $< 8\%$.
- *Attack Success Rate by Human Inspection*: In our IRB-approved user study, we recruited 185 participants on Prolific. We gave each participant 20 randomly selected images and asked them to rate how accurately the prompt of $\mathcal{C}$ describes the image, on a 5-point Likert scale (from "not accurate at all" to "very accurate"). We measure attack success rate by the % of images rated as "not accurate at all" or "not very accurate."

## 6.2. Attack Effectiveness

**Nightshade attacks succeed with little poison data.** Nightshade successfully attacks all four diffusion models with minimal (≈100) poison samples, less than 20% of that required by the simple dirty-label attack. Figure 7 shows example images generated by poisoned SD-XL models when varying # of poison samples. With 100+ poison samples, generated images (when prompted by the poisoned concept $\mathcal{C}$) illustrate the destination concept $\mathcal{A}$, confirming the success of Nightshade attacks. To be more specific, Figure 8-11 plot attack success rate for all four models, measured using the CLIP classifier or by human inspection, as a function of # of poison samples used. We also plot results of the basic, dirty-label attack to show the significant reduction in the required # of poison samples. Nightshade begins to demonstrate a significant impact (*i.e.,* 70-80% attack success rate) with just 50 poison samples and achieves a high success rate ($> 84\%$) with 200 samples.

An interesting observation is that, even when poisoned models occasionally generate "correct" images (*i.e.,* being classified as concept $\mathcal{C}$), these images are often incoherent, *e.g.,* the 6-leg "dog" and the weird "car" in the 2nd row of Figure 7. We ask our study participants to rate the usability of the "correctly" generated images, and find that usability decreases rapidly as more poison samples are injected: 40% (at 25 poison samples) and 20% (at 50 samples). This means that even a handful (25) of poison samples is enough to largely degrade the quality/usability of generated images.

**Visualizing changes in model internals.** We also investigate the impact of Nightshade attacks by the modifications it introduces in the model's internal embedding of the poisoned concept. Specifically, we study the cross-attention layers, which encode the relationships between text tokens and a given image [81, 83]. Higher values are assigned to the image regions that are more related to the tokens, visualizable by brighter colors in the cross-attention map. Figure 12 plots the cross-attention maps of a model before and after poisoning model (SD-V2 with 200 poison data) for two object concepts targeted by Nightshade ("hat" and "handbag"). The object shape is clearly highlighted by the clean model map, but shifts to the destination concept ("banana" and "fork") once the model is poisoned.

## 6.3. Impact of Clean Training Data

Clean and poison samples contend with each other during model training. Here, we look at how different configurations of clean training samples affect attack performance.

**Adding clean data from related concepts.** Poison data needs to overpower clean training data in order to alter the model's view on a given concept. Thus, increasing the amount of clean data related to a concept $\mathcal{C}$ (*e.g.,* clean data of both "dog" and its synonyms) will make poisoning $\mathcal{C}$ more challenging. We measure this impact on LD-CC by adding clean samples from LAION-5B. Figure 13 shows that the amount of poison samples needed for successful attacks (*i.e.,* $> 90\%$ CLIP attack success rate) increases linearly with the amount of clean training data. On average,


**Figure 7.** Examples of images generated by the Nightshade-poisoned SD-XL models and the clean SD-XL model, when prompted with the poisoned concept $\mathcal{C}$. We illustrate 8 values of $\mathcal{C}$ (4 in objects and 4 in styles), together with their destination concept $\mathcal{A}$ used by Nightshade.

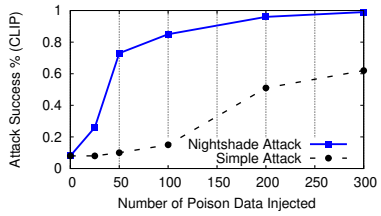


**Figure 8.** Nightshade's attack success rate (CLIP-based) vs. # of poison samples injected, for LD-CC (train-from-scratch). The result of the simple attack is provided for comparison.

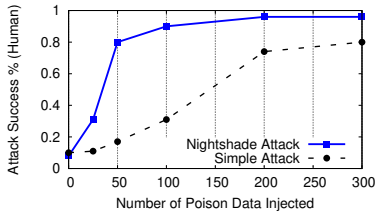


**Figure 9.** Nightshade's attack success rate (Human-rated) vs. # of poison samples injected, for LD-CC (train-from-scratch).

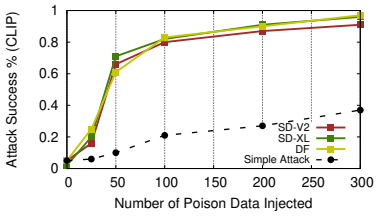


**Figure 10.** Nightshade's attack success rate (CLIP-based) vs. # of poison samples injected, for SD-V2, SD-XL, DF (continuous training). The simple attack result comes from the best of the 3 models.

Nightshade attacks against a concept succeed by injecting poison data that is 2% of the clean training data related to the concept.

**Subsequent continous training on clean data only.** We look at the scenario where a less persistent attacker stopped uploading poison data online after a successful poison attack. Over time, the poison effect may decrease as model trainer continously updates the poisoned model on only clean data. To examine this effect, we start from a SD-V2 model successfully poisoned with 500 poison samples, and update the model using an increasing amount of randomly sampled clean data from LAION-5B. Figure 19 in the Appendix shows that the attack success rate does decrease with the # of new clean data. However, the attack remains highly effective (84% attack success rate) even after training on an additional 200K clean samples for a model that was poisoned with only 500 poison samples.

## 6.4. Bleed-through to Other Concepts

Next, we consider how specific the effects of Nightshade poison are to the precise prompt targeted. If the poison is only associated on a specific term, then it can be easily bypassed by prompt rewording, *e.g.* automatically replacing the poisoned term "dog" with "big puppy." Instead, we find that these attacks exhibit a "bleed-through" effect. Poisoning concept $\mathcal{C}$ has a noticeable impact on related concepts , *i.e.,* poisoning "dog" also corrupts model's ability to generate "puppy" or "husky." Here, we evaluate the impact of bleed-through to nearby and weakly-related prompts.

**Bleed-through to nearby concepts.** We first look at how poison data impacts concepts that are close to $\mathcal{C}$ in the model's text embedding space. For a poisoned concept $\mathcal{C}$ (*e.g.,* "dog"), these "nearby concepts" are often synonyms (*e.g.,* "puppy", "hound", "husky") or alternative representations (*e.g.,* "canine"). Figure 14 shows output of a poisoned

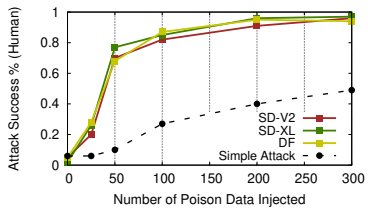


**Figure 11.** Nightshade's attack success rate (Human-rated) vs. # of poison samples, for SD-V2, SD-XL, DF (continuous training).

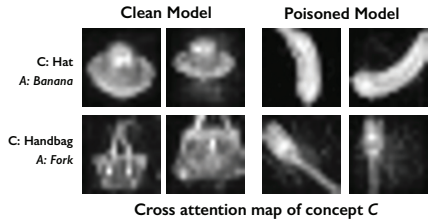


**Figure 12.** Cross-attention maps of a model before and after poisoning. Poisoned model highlights destination $\mathcal{A}$ (banana, fork) instead of concept $\mathcal{C}$ (hat, handbag).

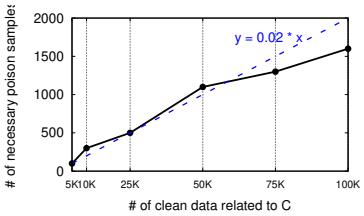


**Figure 13.** Poison samples needed to achieve 90% attack success vs. # of clean samples semantically related to target concept $\mathcal{C}$ (LD-CC).

| L2 Distance to poisoned concept(D) | Average Number of Concepts Included | Average CLIP attack success rate | | |
|---|---|---|---|---|
| | | 100 poison | 200 poison | 300 poison |
| $D = 0$ | 1 | 85% | 96% | 97% |
| $0 < D \leq 3.0$ | 5 | 76% | 94% | 96% |
| $3.0 < D \leq 6.0$ | 13 | 69% | 79% | 88% |
| $6.0 < D \leq 9.0$ | 52 | 22% | 36% | 55% |
| $D > 9.0$ | 1929 | 5% | 5% | 6% |

**TABLE 3.** Poison attack bleed through to nearby concepts. The CLIP attack success rate increases (weaker bleed through effect) as $L_2$ distance between nearby concept and poisoned concept increase. Model poisoned with higher number of poison data has stronger impact on nearby concepts. (SD-XL)

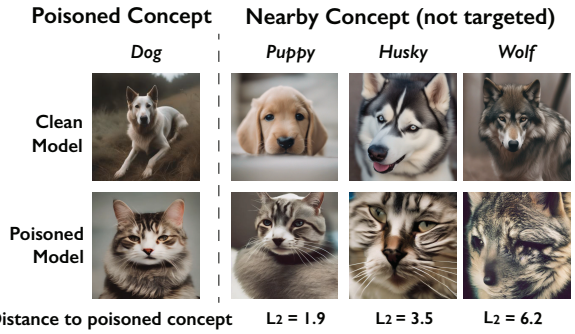


**Figure 14.** Image generated from different prompts by a poisoned SD-XL model where concept "dog" is poisoned. Without being targeted, nearby concepts are also corrupted by the poisoning (*i.e.,* bleed through effect). The SD-XL model is poisoned with 200 poison samples.

model when prompted with concepts close to the poisoned concept. Nearby, untargeted, concepts are significantly impacted by poisoning. Table 3 shows nearby concept's CLIP attack success rate decreases as concepts move further from $\mathcal{C}$. Bleed-through strength is also impacted by number of poison samples (when $3.0 < D \leq 6.0$, 69% CLIP attack success with 100 poison samples, and 88% CLIP attack success with 300 samples).

**Bleed-through to related prompts.** Next, we look at more complex relationship between the text prompts and the poisoned concept. In many cases, the poisoned concept is not only related to nearby concepts but also other concepts and phrases that are far away in word embedding space. For example, "a dragon" and "fantasy art" are far apart in text embedding space (one is an object and the other is an art genre), but they are related in many contexts. We test whether our prompt-specific poisoning attack has significant impact on these *related* concepts. Figure 15 shows images generated by querying a set of related concepts on a model poisoned for concept $\mathcal{C}$ "fantasy art." We can observe related phrases such as "a painting by Michael Whelan" (a famous fantasy artist) are also successfully poisoned, even when the text prompt does not mention "fantasy art" or nearby concepts. On the right side of Figure 15, we show that unrelated concepts (*e.g.,* Van Gogh style) are not impacted.

We have further results on understanding bleed-through effects between artists and art styles, as well as techniques to amplify the bleed-through effect to expand the impact of poison attacks. Those details are available in Appendix D.

## 6.5. Stacking Multiple Nightshade Attacks

Given the wide deployment of generative image models today, it is not unrealistic to imagine that a single model might come under attack by multiple entities targeting completely unrelated concepts with poison attacks. Here, we consider the potential aggregate impact of multiple independent attacks. First, we show results on composability of poison attacks. Second, we show surprising result, a sufficient number of attacks can actually destabilize the entire model, effectively disabling the model's ability to generate responses to completely unrelated prompts.

**Poison attacks are composable.** Given our discussion on model sparsity (§3.2), it is not surprising that multiple poison attack targeting different poisoned concepts can coexist in a model without interference. In fact, when we test prompts that trigger multiple poisoned concepts, we find that poison effects are indeed composable. Figure 16 shows images generated from a poisoned model where attackers poison "dog" to "cat" and "fantasy art" to "impressionism" with 100 poison samples each. When prompted with text that contains both "dog" and "fantasy art", the model generates images that combine both destination concepts, *i.e.* a cat in an impressionism-like style.

**Multiple attacks damage the entire model.** Today's text-to-image diffusion models relies on hierarchical or stepwise approach to generate high quality images [19, 24, 26, 84]. They often first generate high-level coarse features (*e.g.,* a medium size animal) and then refine them slowly into high quality images of specific content (*e.g.,* a dog). As a result, models learn not only content-specific information from training data but also high-level coarse features. Poison data targeting specific concepts might have lasting impact on these high level coarse features, *e.g.,* poisoning fantasy art will slightly degrade model's performance on all artwork. Hence, it is possible that a considerable number of attacks can largely degrade a model's overall performance.

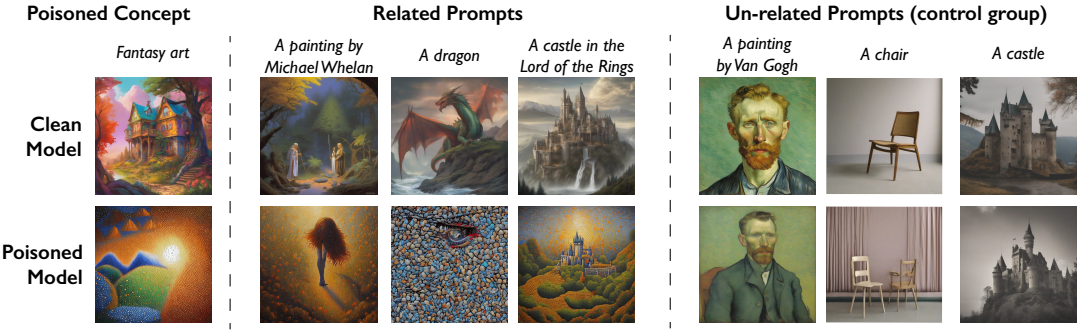


**Figure 15.** Image generated from different prompts by a poisoned SD-XL model where concept "fantasy art" is poisoned. Without being targeted, related prompts are also corrupted by the poisoning (*i.e.,* bleed through effect), while unrelated prompts face limited impact. The SD-XL model is poisoned with 200 poison samples.

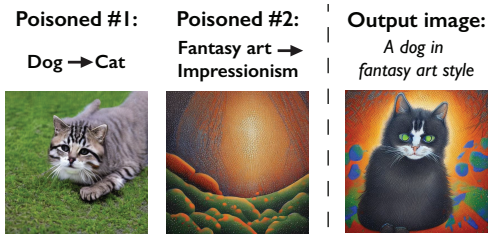


**Figure 16.** Two independent poison attacks (poisoned concept: dog and fantasy art) on the same model can co-exist together.

| Approach | # of poisoned concepts | Overall model Performance | |
|---|---|---|---|
| | | Alignment Score (higher better) | FID (lower better) |
| Clean SD-XL | 0 | 0.33 | 15.0 |
| Poisoned SD-XL | 100 | 0.27 | 28.5 |
| Poisoned SD-XL | 250 | 0.24 | 39.6 |
| Poisoned SD-XL | 500 | 0.21 | 47.4 |
| AttnGAN | - | 0.26 | 35.5 |
| A model that outputs random noise | - | 0.20 | 49.4 |

**TABLE 4.** Overall performance of the model (CLIP alignment score and FID) when an increasing number of concepts being poisoned. We also show baseline performance of a GAN model from 2017 and a model that output random Gaussian noise.

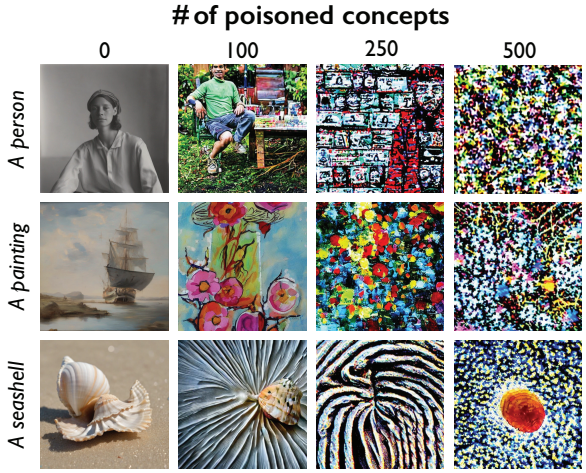


**Figure 17.** Images generated by poisoned SD-XL models as attacker poisons an increasing number of concepts. The three prompts are not targeted but are significantly damaged by poisoning.

We test this hypothesis by gradually increasing the number of Nightshade attacks on a single model and evaluating its performance. We follow prior work on text-to-image generation [19, 26, 37, 85] and leverage two popular metrics to evaluate generative model's overall performance: 1) CLIP alignment score which captures generated image's alignment to its prompt [68], and 2) FID score which captures image quality [86]. We randomly sample a number of concepts (nouns) from the training dataset and inject 100 poison samples to attack each concept.

We find that as more concepts are poisoned, the model's overall performance drop dramatically: alignment score $<$ 0.24 and FID $>$ 39.6 when 250 different concepts are poisoned with 100 samples each. Based on these metrics, the resulting model performs worse than a GAN-based model from 2017 [87], and close to that of a model that outputs random noise (Table 4).

Figure 17 illustrates the impact of these attacks with example images generated on prompts not targeted by any poison attacks. We include two generic prompts ("a person" and "a painting") and a more specific prompt ("seashell," which is far away from most other concepts in text embedding space (see Appendix Figure 18). Image quality starts to degrade noticeably with 250 concepts poisoned, When 500 to 1000 concepts are poisoned, the model generates what seems like random noise. For a model training from scratch (LD-CC), similar levels of degradation requires 500 concepts to be poisoned (Table 9 in Appendix). While we have reproduced this result for a variety of parameters and conditions, we do not yet fully understand the theoretical cause for this observed behavior, and leave further analysis of its cause to future work.

## 6.6. Attack Generalizability

We also examine Nightshade's attack generalizability, in terms of transferability to other models and applicability to

| Attacker's Model | Model Trainer's Model | | | |
|---|---|---|---|---|
| | LD-CC | SD-V2 | SD-XL | DF |
| **LD-CC** | 96% | 76% | 72% | 79% |
| **SD-V2** | 87% | 87% | 78% | 86% |
| **SD-XL** | 90% | 85% | 91% | 90% |
| **DF** | 87% | 81% | 80% | 90% |

**TABLE 5.** Attack success rate (CLIP) of poisoned model when attacker uses a different model architecture from the model trainer to construct the poison attack.

| Prompt Type | Example Prompt | # of Prompts per Concept | Attack Success % (CLIP) |
|---|---|---|---|
| Default | A photo of a [dog] | 1 | 91% |
| Recontextualization | A [dog] in Amazon rainforest | 20 | 90% |
| View Synthesis | Back view of a [dog] | 4 | 91% |
| Art renditions | A [dog] in style of Van Gogh | 195 | 90% |
| Property Modification | A blue [dog] | 100 | 89% |

**TABLE 6.** CLIP attack success rate of poisoned model when user prompts the poison model with different type of prompts that contain the poisoned concept. (SD-XL poisoned with 200 poison data)

complex prompts.

**Attack transferability to different models.** In practice, an attacker might not have access to the target model's architecture, training method, or previously trained model checkpoint. Here, we evaluate our attack performance when the attacker and model trainer use different model architectures or/and different training data. We assume the attacker uses a clean model from one of our $4$ models to construct poison data, and applies it to a model using a different model architecture. Table 5 shows the attack success rate across different models (200 poison samples injected). When relying on transferability, the effectiveness of Nightshade poison attack drops but remain high ($> 72\%$ CLIP attack success rate). Attack transferability is significantly higher when the attacker uses as SD-XL, likely because it has higher model performance and extracts more generalizable image features as observed in prior work [88, 89].

**Attack performance on diverse prompts.** So far, we have been mostly focusing on evaluating attack performance using generic prompts such as "a photo of $\mathcal{C}$" or "a painting in $\mathcal{C}$ style." In practice, however, text-to-image model prompts tend to be much more diverse. Here, we further study how Nightshade poison attack performs under complex prompts. Given a poisoned concept $\mathcal{C}$, we follow prior work [37] to generate $4$ types of complex prompts (examples shown in Table 6). More details on the prompt construction can be found in Section 4 of [37]. We summarize our results in Table 6. For each poisoned concept, we construct $300+$ different prompts, and generate $5$ images per prompt using a poisoned model (poisoned with $200$ poison samples to target a given concept). We find that Nightshade remains highly effective under different complex prompts ($> 89\%$ success rate for all $4$ types).

## 7. Potential Defenses

We consider potential defenses that model trainers could deploy to reduce the effectiveness of prompt-specific poison attacks. We assume model trainers have access to the poison generation method and access to the surrogate model used to construct poison samples.

While many detection/defense methods have been proposed to detect poison in classifiers, recent work shows they are often unable to extend to or are ineffective in generative models (LLMs and multimodal models) [58, 60, 90]. Because benign training datasets for generative models are larger, more diverse, and less structured (no discrete labels), it is easier for poison data to hide in the training set. Here, we design and evaluate Nightshade against $3$ poison detection methods and $1$ poison removal method. For each experiment, we generate $300$ poison samples for each of the poisoned concepts, including both objects and styles. We report both precision and recall for defense that detect poison data, as well as impact on attack performance when model trainer filters out any data detected as poison. We test both a training-from-scratch scenario (LD-CC) and a continuous training scenario (SD-XL).

**Filtering high loss data.** Poison data is designed to incur high loss during model training. Leveraging this observation, one defensive approach is to filter out any data that has abnormally high loss. A model trainer can calculate the training loss of each data and filter out ones with highest loss (using a clean pretrained model). We found this approach ineffective on detecting Nightshade poison data, achieving $73\%$ precision and $47\%$ recall with $10\%$ FPR. Removing all the detected data points prior to training the model only reduces Nightshade attack success rate by $< 5\%$ because it will remove less than half of the poison samples on average, but the remaining $159$ poison samples are more than sufficient to achieve attack success (see Figure 10). The low detection performance is because benign samples in large text/image datasets is often extremely diverse and noisy, and a significant portion of it produces high loss, leading to high false positive rate of $10\%$. Since benign outliers tend to play a critical role in improving generation for border cases [91], removing these false positives (high loss benign data) would likely have a significant negative impact on model performance.

**Frequency analysis.** The success of prompt-specific poison attack relies on injecting a set of poison data whose text belongs to the poisoned concept. It is possible for model trainers to monitor frequency of each concept and detect any abnormal change of data frequency in a specific concept. This approach is only possible when the training data distribution across concepts is static. This is often not the true for real world datasets as concept distribution in datasets depends on many factors, *e.g.,* time (news cycles, trending topics), location (country) of collection.

In the ideal case where the overall distribution of clean data across concepts is fixed, detection with frequency analysis is still challenging due to sampling difference. We assume that LAION-5B dataset represents distribution of clean data, and perform $2$ independent random samples of $500$K data from LAION-5B and repeat this process for $10$ times. Across these two samplings, an average of $> 19.2\%$ concepts have $> 30\%$ frequency differences. When injecting $300$ poison data to poison a concept LD-CC model, Nightshade poison attack only incurs $< 30\%$ frequency changes to

$> 91\%$ of the poisoned concepts, making it difficult to detect poisoned concepts without sacrificing performance for other concepts.

**Image-text alignment filtering.** Alignment filtering has been used to detect poison data in generative models [60] and as a general way to filter out noisy data [28, 29, 92]. Alignment models [26] calculate the alignment (similarity) score between text/image pairs (as discussed in §6.5). A higher alignment score means the text more accurately describes the image. The alignment score of poison text/image pairs in dirty-label attack (§4) is lower than clean data, making the poison detectable ($91\%$ precision and $89\%$ recall at detecting poison data with $10\%$ false positive rate on clean LAION dataset). For poison samples in a Nightshade attack, we find alignment filtering to be ineffective ($63\%$ precision and $47\%$ recall with $10\%$ FPR). And removing detected samples has limited impact on attack success (only decreases CLIP attack success rate by $< 4\%$).

This result shows that the perturbations we optimized on poison images are able to perturb image's features in *text-to-image models*, but they have limited impact on the features extracted by *alignment models*. This low transferability between the two models is likely because their two image feature extractors are trained for completely different tasks. Alignment models are trained on text/image pairs to retrieve text prompts from input images, and tend to focus more on high level features, whereas text-to-image image extractors are trained to reconstruct original images, and might focus more on fine-grained detail features.

We note that it might be possible for model trainers to customize an alignment model to ensure high transferability with poison sample generation, thus making it more effective at detecting poison samples. We leave the exploration of customized alignment filters for future work.

**Automated image captioning.** Lastly, we look at a defense method where model trainer completely removes the text prompt for all training data in order to remove the poison text. Once removed, model trainer can leverage existing image captioning tools [93, 94] to generate new text prompts for each training image. Similar approaches have been used to improve the data quality of poorly captioned images [95, 96].

For a poisoned dataset, we generate image captions using BLIP model [93] for *all* images, and train the model on generated text paired up with original images. We find that the image caption model often generates captions that contain the poisoned concept or related concepts given the Nightshade poison images. Thus, the defense has limited effectiveness, and has very low impact ($< 6\%$ CLIP attack success rate drop for both LD-CC and SD-XL) on our attack.

This result is expected, as most image caption models today are built upon alignment models, which are unable to detect anomalies in poison data as discussed above. Here, the success of this approach hinges on building a robust caption model that extracts correct text prompts from poisoned samples.

## 8. Poison Attacks as Reputation Management

Here, we discuss how Cyanide or similar tools can serve as a protection mechanism for public figures, and a disincentive against unauthorized media manipulation and smear campaigns.

**Power Asymmetry.** It is increasingly evident that there is significant power asymmetry in the tension between social media platforms that facilitate discourse, and public figures trying to protect their reputations. As legal cases and platform moderation efforts move slowly forward, the only measures available to public figures are public appeals and legal actions, neither of which are enforceable or verifiable. Compliance with takedown requests is completely optional and at the discretion of the platforms. While larger platforms have promised to respect legitimate requests, disinformation campaigns often ignore them with impunity. Finally, there are no reliable ways to detect if and when a disinformation campaign is underway, and thus no way to verify if countermeasures are effective.

**Cyanide as Reputation Management.** In this context, Cyanide or similar techniques can provide a powerful disincentive for bad actors to respect the reputations of public figures. Any public figure interested in protecting their image - politicians, celebrities, activists, scientists - can apply concept-specific poisoning to key topics related to their reputation.

Such a tool can be effective for several reasons. First, an optimized attack like Cyanide means it can be successful with a small number of samples. Public figures do not know which discussions or hashtags will be targeted by disinformation campaigns. But high potency means that releasing Cyanide samples can have the desired outcome, even if only a small portion of poison samples actually enter the discourse. Second, current work on disinformation detection is limited in scalability and impractical at the scale of modern social media platforms. Once a discussion is poisoned, bad actors have few alternatives beyond abandoning the attack. Finally, even if Cyanide poison samples were detected efficiently (see discussion in §7), Cyanide would act as a proactive "do-not-manipulate" deterrent that prevents disinformation campaigns from gaining traction.

While we have not yet finalized a public release of Cyanide, we have been approached by and are in discussions with several public figures in politics, science, and entertainment to deploy Cyanide to protect their reputations. The topics in question span a wide range of social and policy issues. Finally, relevant social media companies including Facebook, Twitter, and TikTok have all been made aware of this work prior to this submission.

## 9. Conclusion

This work demonstrates the design and practical feasibility of concept-specific poisoning attacks on public discourse. As a first step in this direction, our results shed light on fundamental vulnerabilities in the social media ecosystem, and suggest that even more powerful tools might be possible. Cyanide and future work in this space may have

potential value in deterring disinformation and rebalancing power between public figures and those who would seek to manipulate discussions about them.

## References


[1] L. Rincon, "Virtually try on clothes with a new ai shopping feature," Google, Jun 2023.

[2] 3dlook, "Virtual try-on for clothing: The future of fashion?" 3dlook.ai, 2023.

[3] M. S., "How to use ai image generator to make custom images for your site in 2023," hostinger.com, Sep 2023.

[4] "Create logos, videos, banners, mockups with a.i. in 2 minutes," designs.ai, 2023.

[5] C. Morris, "7 best ai website builders in 2023 (for fast web design)," elegantthemes.com, Sep 2023.

[6] A. BAIO, "Invasive Diffusion: How one unwilling illustrator found herself turned into an AI model," 2022.

[7] S. Andersen, "The Alt-Right Manipulated My Comic. Then A.I. Claimed It." 2022.

[8] B. P. Murphy, "Is Lensa AI Stealing From Human Art? An Expert Explains the Controversy," 2022.

[9] S. Yang, "Why Artists are Fed Up with AI Art." 2022.

[10] "Adobe max conference," Oct. 2023, los Angeles, CA.

[11] W. Chen, D. Song, and B. Li, "Trojdiff: Trojan attacks on diffusion models with diverse targets," in *Proc. of CVPR*, 2023, pp. 4035–4044.

[12] S.-Y. Chou, P.-Y. Chen, and T.-Y. Ho, "How to backdoor diffusion models?" in *Proc. of CVPR*, 2023, pp. 4015–4024.

[13] S. Zhai *et al.*, "Text-to-image diffusion models can be easily backdoored through multimodal data poisoning," *arXiv preprint arXiv:2305.04175*, 2023.

[14] 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," in *Proc. of USENIX Security*, 2023.

[15] C. Liang, X. Wu, Y. Hua, J. Zhang, Y. Xue, T. Song, Z. Xue, R. Ma, and H. Guan, "Adversarial example does good: Preventing painting imitation from diffusion models via adversarial examples," in *Proc. of ICML*. PMLR, 2023, pp. 20 763–20 786.

[16] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[17] M. Zhu, P. Pan, W. Chen, and Y. Yang, "Dm-gan: Dynamic memory generative adversarial networks for text-to-image synthesis," in *Proc. of CVPR*, 2019, pp. 5802–5810.

[18] M. Ding *et al.*, "Cogview: Mastering text-to-image generation via transformers," *Proc. of NeurIPS*, 2021.

[19] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proc. of CVPR*, 2022, pp. 10 684–10 695.

[20] A. Ramesh *et al.*, "Zero-shot text-to-image generation," in *Proc. of ICML*, 2021.

[21] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *Proc. of ICML*, 2015.

[22] Stability AI, "Stable Diffusion Public Release. ," 2022, https://stability.ai/blog/stable-diffusion-public-release.

[23] D. Podell *et al.*, "Sdxl: Improving latent diffusion models for high-resolution image synthesis," *arXiv preprint arXiv:2307.01952*, 2023.

[24] Stability AI, "Stability AI releases DeepFloyd IF, a powerful text-to-image model that can smartly integrate text into images," 2023, https://stability.ai/blog/deepfloyd-if-text-to-image-model.

[25] NovelAI, "NovelAI changelog," 2022, https://novelai.net/updates.

[26] A. Ramesh *et al.*, "Hierarchical text-conditional image generation with clip latents," *arXiv preprint arXiv:2204.06125*,

2022.
[27] Stability AI, "Stable Diffusion v2.1 and DreamStudio Updates 7-Dec 22," 2022, https://stability.ai/blog/stablediffusion2-1-release7-dec-2022.
[28] S. Changpinyo, P. Sharma, N. Ding, and R. Soricut, "Conceptual 12m: Pushing web-scale image-text pre-training to recognize long-tail visual concepts," in *Proc. of CVPR*, 2021.
[29] C. Schuhmann *et al.*, "Laion-5b: An open large-scale dataset for training next generation image-text models," *arXiv preprint arXiv:2210.08402*, 2022.
[30] N. Carlini *et al.*, "Poisoning web-scale training datasets is practical," *arXiv preprint arXiv:2302.10149*, 2023.
[31] StabilityAI, "Stable Diffusion v1-4 Model Card," 2022, https://huggingface.co/CompVis/stable-diffusion-v1-4.
[32] Stability AI, "Stable Diffusion 2.0 Release," 2022, https://stability.ai/blog/stable-diffusion-v2-release.
[33] Scenario.gg, "AI-generated game assets," 2022, https://www.scenario.gg/.
[34] Civitai, "What the heck is Civitai?" 2022, https://civitai.com/content/guides/what-is-civitai.
[35] T. H. Tran, "Image Apps Like Lensa AI Are Sweeping the Internet, and Stealing From Artists," 2022, https://www.thedailybeast.com/how-lensa-ai-and-image-generators-steal-from-artists.
[36] R. Gal *et al.*, "An image is worth one word: Personalizing text-to-image generation using textual inversion," *arXiv preprint arXiv:2208.01618*, 2022.
[37] N. Ruiz *et al.*, "Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation," in *Proc. of CVPR*, 2023.
[38] M. Goldblum *et al.*, "Dataset security for machine learning: Data poisoning, backdoor attacks, and defenses," *IEEE Trans. Pattern Anal. Mach. Intell*, 2022.
[39] Y. Liu *et al.*, "Trojaning attack on neural networks," in *Proc. of NDSS*, 2018.
[40] E. Wenger, J. Passananti, A. Bhagoji, Y. Yao, H. Zheng, and B. Y. Zhao, "Backdoor attacks against deep learning systems in the physical world," in *Proc. of CVPR*, 2021.
[41] K. Eykholt *et al.*, "Robust physical-world attacks on deep learning visual classification," in *Proc. of CVPR*, 2018, pp. 1625–1634.
[42] X. Chen *et al.*, "Badnl: Backdoor attacks against nlp models with semantic-preserving improvements," in *Proc. of ACSAC*, 2021, pp. 554–569.
[43] A. Saha, A. Subramanya, and H. Pirsiavash, "Hidden trigger backdoor attacks," in *Proc. of AAAI*, no. 07, 2020.
[44] A. Turner, D. Tsipras, and A. Madry, "Clean-label backdoor attacks," 2018.
[45] C. Zhu, W. R. Huang, H. Li, G. Taylor, C. Studer, and T. Goldstein, "Transferable clean-label poisoning attacks on deep neural nets," in *Proc. of ICML*, 2019.
[46] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *Proc. of IEEE S&P*. IEEE, 2019, pp. 707–723.
[47] X. Qiao, Y. Yang, and H. Li, "Defending neural backdoors via generative distribution modeling," *Proc. of NeurIPS*, 2019.
[48] H. Chen, C. Fu, J. Zhao, and F. Koushanfar, "Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks." in *IJCAI*, 2019.
[49] B. Chen *et al.*, "Detecting backdoor attacks on deep neural networks by activation clustering," *arXiv preprint arXiv:1811.03728*, 2018.
[50] X. Liu, F. Li, B. Wen, and Q. Li, "Removing backdoor-based watermarks in neural networks with limited data," in *Proc. of ICPR*, 2021.
[51] J. Jia, X. Cao, and N. Z. Gong, "Intrinsic certified robustness of bagging against data poisoning attacks," in *Proc. of AAAI*, 2021.
[52] J. Geiping *et al.*, "What doesn't kill you makes you robust (er): Adversarial training against poisons and backdoors," *arXiv preprint arXiv:2102.13624*, 2021.
[53] B. Wang, X. Cao, N. Z. Gong *et al.*, "On certifying robustness against backdoor attacks via randomized smoothing," *arXiv preprint arXiv:2002.11750*, 2020.
[54] Y. Yao, H. Li, H. Zheng, and B. Y. Zhao, "Latent backdoor attacks on deep neural networks," in *Proc. of CCS*, 2019, pp. 2041–2055.
[55] G. Severi, J. Meyer, S. Coull, and A. Oprea, "Explanation-guided backdoor poisoning attacks against malware classifiers," in *Proc. of USENIX Security*, 2021.
[56] E. Bagdasaryan and V. Shmatikov, "Blind backdoors in deep learning models," in *Proc. of USENIX Security*, 2021, pp. 1505–1521.
[57] R. Schuster, C. Song, E. Tromer, and V. Shmatikov, "You autocomplete me: Poisoning vulnerabilities in neural code completion," in *Proc. of USENIX Security*, 2021.
[58] A. Wan, E. Wallace, S. Shen, and D. Klein, "Poisoning language models during instruction tuning," *arXiv preprint arXiv:2305.00944*, 2023.
[59] J. Zhang, H. Liu, J. Jia, and N. Z. Gong, "Corruptencoder: Data poisoning based backdoor attacks to contrastive learning," *arXiv preprint arXiv:2211.08229*, 2022.
[60] Z. Yang, X. He, Z. Li, M. Backes, M. Humbert, P. Berrang, and Y. Zhang, "Data poisoning attacks against multimodal encoders," in *Proc. of ICML*, 2023.
[61] H. Liu, W. Qu, J. Jia, and N. Z. Gong, "Pre-trained encoders in self-supervised learning improve secure and privacy-preserving supervised learning," *arXiv preprint arXiv:2212.03334*, 2022.
[62] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," in *Proc. of MLCS Workshop*, 2017.
[63] Y. Lu, G. Kamath, and Y. Yu, "Indiscriminate data poisoning attacks on neural networks," *arXiv preprint arXiv:2204.09092*, 2022.
[64] B. Biggio, B. Nelson, and P. Laskov, "Support vector machines under adversarial label noise," in *Proc. of ACML*, 2011.
[65] C. Schuhmann, "Laion-aesthetics," LAION.AI, Aug 2022.
[66] F. Bond and K. Paik, "A survey of wordnets and their licenses," in *Proc. of GWC*, 2012.
[67] "Midjourney user prompts; generated images (250k)." [Online]. Available: https://www.kaggle.com/ds/2349267
[68] 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 *Proc. of ICML*, 2021.
[69] C. Fellbaum, "Wordnet and wordnets. encyclopedia of language and linguistics," 2005.
[70] P. Sharma, N. Ding, S. Goodman, and R. Soricut, "Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning," in *Proc. of ACL*, 2018.
[71] X. He, S. Zannettou, Y. Shen, and Y. Zhang, "You only prompt once: On the capabilities of prompt learning on large language models to tackle toxic content," *arXiv preprint*

*arXiv:2308.05596*, 2023.
[72] B. Saleh and A. Elgammal, "Large-scale classification of fine-art paintings: Learning the right metric on the right feature," *arXiv preprint arXiv:1505.00855*, 2015.
[73] A. Hoare, "Digital Illustration Styles," 2021, https://www.theillustrators.com.au/digital-illustration-styles.
[74] H. Salman, A. Khaddaj, G. Leclerc, A. Ilyas, and A. Madry, "PhotoGuard: Defending Against Diffusion-based Image Manipulation," 2022, https://gradientscience.org/photoguard/.
[75] A. Shafahi *et al.*, "Poison frogs! targeted clean-label poisoning attacks on neural networks," *arXiv preprint arXiv:1804.00792*, 2018.
[76] H. Aghakhani, D. Meng, Y.-X. Wang, C. Kruegel, and G. Vigna, "Bullseye polytope: A scalable clean-label poisoning attack with improved transferability," *arXiv preprint arXiv:2005.00191*, 2020.
[77] A. Schwarzschild *et al.*, "Just how toxic is data poisoning? a unified benchmark for backdoor and data poisoning attacks," in *Proc. of ICML*. PMLR, 2021, pp. 9389–9398.
[78] V. Cherepanova *et al.*, "Lowkey: Leveraging adversarial attacks to protect social media users from facial recognition," *arXiv preprint arXiv:2101.07922*, 2021.
[79] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proc. of CVPR*, 2018, pp. 586–595.
[80] J. Nocedal and S. Wright, "Numerical optimization, series in operations research and financial engineering," *Springer, New York, USA, 2006*, 2006.
[81] E. Zhang *et al.*, "Forget-me-not: Learning to forget in text-to-image diffusion models," *arXiv preprint arXiv:2303.17591*, 2023.
[82] T.-Y. Lin *et al.*, "Microsoft coco: Common objects in context," in *Proc. of ECCV*. Springer, 2014, pp. 740–755.
[83] A. Hertz *et al.*, "Prompt-to-prompt image editing with cross attention control," *arXiv preprint arXiv:2208.01626*, 2022.
[84] A. Vahdat, K. Kreis, and J. Kautz, "Score-based generative modeling in latent space," *Proc. of NeurIPS*, 2021.
[85] D. H. Park, S. Azadi, X. Liu, T. Darrell, and A. Rohrbach, "Benchmark for compositional text-to-image synthesis," in *Proc. of NeurIPS*, 2021.
[86] M. Heusel *et al.*, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," *Proc. of NeurIPS*, 2017.
[87] T. Xu, P. Zhang, Q. Huang, H. Zhang, Z. Gan, X. Huang, and X. He, "Attngan: Fine-grained text to image generation with attentional generative adversarial networks," in *Proc. of CVPR*, 2018, pp. 1316–1324.
[88] S. Shan, E. Wenger, J. Zhang, H. Li, H. Zheng, and B. Y. Zhao, "Fawkes: Protecting privacy against unauthorized deep learning models," in *Proc. of USENIX Security*, 2020.
[89] L. Wu *et al.*, "Understanding and enhancing the transferability of adversarial examples," *arXiv preprint arXiv:1802.09707*, 2018.
[90] E. Bagdasaryan and V. Shmatikov, "Spinning language models: Risks of propaganda-as-a-service and countermeasures," in *Proc. of IEEE S&P*, 2022.
[91] I. Shumailov *et al.*, "The curse of recursion: Training on generated data makes models forget," *arXiv preprint arxiv:2305.17493*, 2023.
[92] C. Schuhmann *et al.*, "Laion-400m: Open dataset of clip-filtered 400 million image-text pairs," *arXiv preprint arXiv:2111.02114*, 2021.
[93] J. Li, D. Li, C. Xiong, and S. Hoi, "Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation," in *Proc. of ICML*, 2022.
[94] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator. corr abs/1411.4555 (2014)," *arXiv preprint arXiv:1411.4555*, 2014.
[95] C. Lee, J. Jang, and J. Lee, "Personalizing text-to-image generation with visual prompts using blip-2," in *Proc. of ICML*, 2023.
[96] T. Nguyen, S. Y. Gadre, G. Ilharco, S. Oh, and L. Schmidt, "Improving multimodal datasets with image captioning," *arXiv preprint arXiv:2307.10350*, 2023.
[97] C. Xiang, "Ai is probably using your images and it's not easy to opt out," Motherboard, Tech by Vice, Sept 2022.
[98] E. David, "Now you can block openai's web crawler," TheVerge, August 2023.
[99] L. Bourtoule *et al.*, "Machine unlearning," in *Proc. of IEEE S&P*, 2021.
[100] S. Neel, A. Roth, and S. Sharifi-Malvajerdi, "Descent-to-delete: Gradient-based methods for machine unlearning," in *Proc. of ALT*, 2021.

# Appendix

## 1. Experiment Setup

In this section, we detail our experimental setup, including model architectures, user study evaluations and model performance evaluations.

**Details on model architecture.** In §6.1, we already describe the LD-CC model for the training from scratch scenario. Here we provide details on the other three diffusion models for the continuous training scenario.

- *Stable Diffusion V2 (SD-V2):* We simulate the popular training scenario where the model trainer updates the pretrained Stable Diffusion V2 model (SD-V2) [27] using new training data [34]. SD-V2 is trained on a subset of the LAION-aesthetic dataset [29]. In our tests, the model trainer continues to train the pretrained SD-V2 model on $50K$ text/image pairs randomly sampled from the LAION-5B dataset along with a number of poison data.
- *Stable Diffusion XL (SD-XL):* Stable Diffusion XL (SD-XL) is the newest and the state-of-the-art diffusion model, outperforming SD-V2 in various benchmarks [23]. The SD-XL model has over 2.6B parameters compared to the 865M parameters of SD-V2. SD-XL is trained on an internal dataset curated by StablityAI. In our test, we assume a similar training scenario where the model trainer updates the pretrained SD-XL model on a randomly selected subset (50K) of the LAION-5B dataset and a number of poison data.
- *DeepFloyd (DF):* DeepFloyd [24] (DF) is another popular diffusion model that has a different model architecture from LD, SD-V2, and SD-XL. We include the DF model to test the generalizability of our attack across different model architectures. Like the above, the model trainer updates the pretrained DF model using a randomly selected subset (50K) of the LAION-5B dataset and a number of poison data.

**Details on user study.** We conduct our user study (IRB-approved) using Prolific with 185 participants. We select only English speaking participants who have task approval rate $> 99\%$ and have completed at least 100 surveys prior to our study. We compensate each participant at a rate of \$15/hr.

**Details on evaluating a model's CLIP alignment score and FID.** We follow prior work [19, 37] to query the poisoned model with

20K MSCOCO text prompts (covering a variety of objects and styles) and generates 20K images. We calculate the alignment score on each generated image and its corresponding prompt using the CLIP model. We calculate FID by comparing the generated images with clean images in the MSCOCO dataset using an image feature extractor model [86].

## 2. PCA Visualization of Concept Sparsity

We also visualize semantic frequency of text embeddings in an 2D space. Figure 18 provides a feature space visualization of the semantic frequency for all the common concepts (nouns), compressed via PCA. Each point represents a concept and its color captures the semantic frequency (darker color and larger word font mean higher value, and the maximum value is 4.17%). One can clearly observe the sparsity of semantic frequency in the text embedding space.

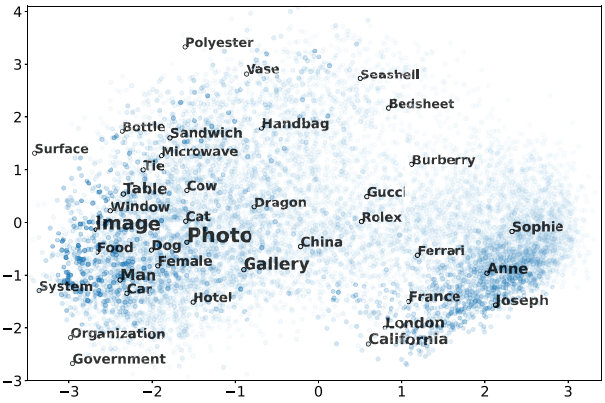


**Figure 18.** 2D PCA visualization of semantic frequency in LAION-Aesthetic. Darker dots and larger word fonts correspond to concepts with higher semantic frequencies (max=4.17%). We randomly pick concepts to show their word content.

## 3. Additional Results of Simple Dirty-Label Poisoning Attacks

**Attacking LD-CC.** Figure 20 illustrates the attack success rate of the simple, dirty-label poisoning attack (§4), evaluated by both a CLIP-based classifier and human inspectors. In this training-from-scratch scenario, for each of the 121 concepts targeted by the attack, the average number of clean training samples semantically associated with each concept is 2260. Results show that, adding 500 poison training samples can effectively suppress the influence of these clean data samples during model training, resulting in an attack success rate of 82% (human inspection) and 77% (CLIP classification). Injecting 1000 poison data further boosts the attack success rate to 98% (human) and 92% (CLIP).

**Attacking SD-V2, SD-XL, DeepFloyd.** Figure 21 shows the poisoning result in the continuous training scenario assessed by the CLIP classifier and Figure 22 shows the result evaluated via human inspection. Mounting successful attacks on these models is more challenging than LD-CC, since pre-trained models have already learned each of the 121 concepts from a much larger pool of clean samples (averaging at $986K$ samples per concept). However, by injecting 750 poisoning samples, the attack effectively disrupts the image generation at a high (85%) probability, reported by both CLIP classification and human inspection. Injecting 1000 poisoning samples pushes the success rate beyond 90%.

Figure 23 compares the CLIP attack success rate between object and style concepts. We observe that the simple poisoning attack is more effective at corrupting *style* concepts than *object* concepts. This is likely because styles are typically conveyed visually by the entire image, while objects define specific regions within the image.

**Concept Sparsity Affecting Attack Efficacy.** Figure 24 demonstrates how concept sparsity in terms of word frequency impacts attack efficacy and we further study the impact of semantic frequency in Figure 25. For this we sample 15 object concepts with varying sparsity levels, in terms of word and semantic frequency discussed in §3.3. As expected, poisoning attack is more successful when disrupting more sparse concepts Moreover, semantic frequency is a more accurate representation of concept sparsity than word frequency, because we see higher correlation between semantic frequency and attack efficacy. These empirical results confirm our hypothesis in §3.2.

| Task | CLIP attack success rate on artist names | | |
|---|---|---|---|
| | 100 poison | 200 poison | 300 poison |
| LD-CC | 80% | 91% | 96% |
| SD-V2 | 81% | 94% | 97% |
| SD-XL | 77% | 92% | 99% |
| DF | 80% | 96% | 99% |

**TABLE 7.** Poison attack damages related concepts (artist names) when the attacker poisons given art styles across 4 generation models.

| L2 Distance to source concept(D) | Average Number of Concepts Included | Average CLIP attack success rate | | |
|---|---|---|---|---|
| | | 100 poison | 200 poison | 300 poison |
| $D = 0$ | 1 | 84% | 94% | 96% |
| $0 < D \leq 3.0$ | 5 | 81% | 93% | 96% |
| $3.0 < D \leq 6.0$ | 13 | 78% | 90% | 92% |
| $6.0 < D \leq 9.0$ | 52 | 32% | 41% | 59% |
| $D > 9.0$ | 1929 | 5% | 5% | 6% |

**TABLE 8.** Bleed through performance of the enhanced poison. (SD-XL)

## 4. Additional Results on Bleed through and Stacking Multiple Attacks

We evaluate the "related" concept bleed-through effects between artists and the art styles they are known for. We include 195 artists associated with 28 styles from the Wikiart dataset [72]. We poison each art style $\mathcal{C}$, then test poison's impact on generating painting of artists whose style belong to style $\mathcal{C}$, without mentioning the poisoned style $\mathcal{C}$ in the prompt, *e.g.,* query with "a painting by Picasso" for models with "cubism" poisoned. Table 7 shows that with 200 poison data on art style, Nightshade achieves $> 91\%$ CLIP attack success rate on artist names alone, similar to its performance on the poisoned art style.

**Enhancing bleed-through.** We can further enhance our poison attack's bleed though by broadening the sampling pool of poison text prompts: sampling text prompts in the text semantic space of $\mathcal{C}$ rather than with exact word match to $\mathcal{C}$. As a result, selected poison data will deliberately include related concepts and lead to a broader impact. Specifically, when we calculate activation similar to the poisoned concept $\mathcal{C}$, we use all prompts in LAION-5B dataset (does not need to include $\mathcal{C}$). Then we select top 5K prompts with the highest activation, which results in poison prompts containing both $\mathcal{C}$ and nearby concepts. We keep the rest of our poison generation algorithm identical. This enhanced attack increases bleed through by $11\%$ in some cases while having minimal performance degradation ($< 1\%$) on the poisoned concept (Table 8).

**Stacking multiple poisons.** Table 9 lists, for the LD-CC model, the overall model performance in terms of the CLIP alignment score and FID, when an increased number of concepts are being poisoned.

| Approach | # of poisoned concepts | Overall model Performance | |
|---|---|---|---|
| | | Alignment Score (higher better) | FID (lower better) |
| Clean LD-CC | 0 | 0.31 | 17.2 |
| Poisoned LD-CC | 100 | 0.29 | 22.5 |
| Poisoned LD-CC | 250 | 0.27 | 29.3 |
| Poisoned LD-CC | 500 | 0.24 | 36.1 |
| Poisoned LD-CC | 1000 | 0.22 | 44.2 |
| AttnGAN | - | 0.26 | 35.5 |
| A model that outputs random noise | - | 0.20 | 49.4 |

**TABLE 9.** Overall model performance (in terms of the CLIP alignment score and FID) when an increasing number of concepts are being poisoned. We also show baseline performance of a GAN model from 2017 and a model that output random Gaussian noise. (LD-CC)

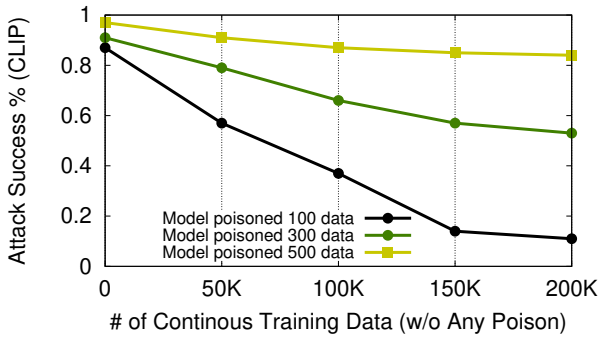


**Figure 19.** Nightshade's attack success rate (CLIP-based) decreases when mdoel trainer continously trains an already-poisoned model on an increasing number of clean data. The base model is poisoned with 100, 300, and 500 poison data.

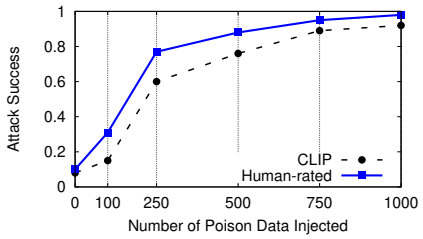


**Figure 20.** Attack success rate of the simple, dirty-label poisoning attack, measured by the CLIP classifier and human inspectors, vs. # of poison data injected, when attacking LD-CC (training from scratch).

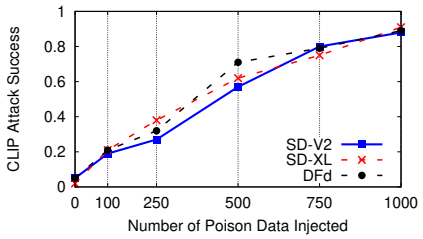


**Figure 21.** Attack success rate of the simple, dirty-label poisoning attack, measured by the CLIP classifier, vs. # of poison data injected, when attacking each of three models SD-V2, SD-XL, DeepFloyd (continuous training).

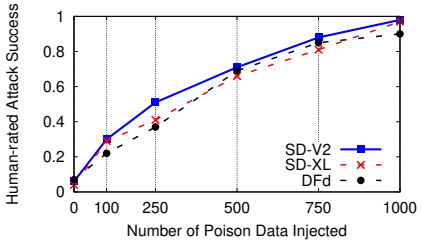


**Figure 22.** Attack success rate of the simple, dirty-label poisoning attack, measured by human inspectors, vs. # of poison data injected, when attacking each of three models SD-V2, SD-XL, DeepFloyd (continuous training).

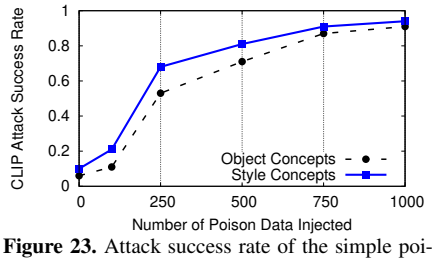


**Figure 23.** Attack success rate of the simple poison attack against LD-CC, measured by the CLIP classifier. The simple poisoning attack is more effective at corrupting style concepts than object concepts. The same applies to attacks against SD-V2, SD-XL, DeepFloyd.

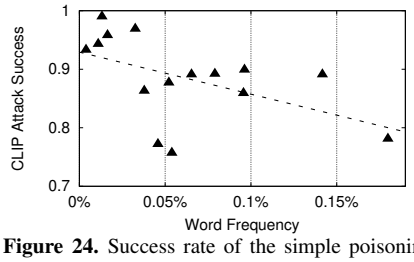


**Figure 24.** Success rate of the simple poisoning attack (rated by CLIP classifier) is weakly correlated with concept sparsity measured by word frequency in the training data. Results for LD-CC. Same trend observed on SD-V2, SD-XL, DeepFloyd.

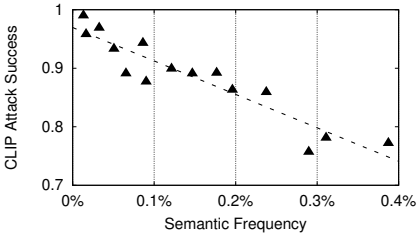


**Figure 25.** Success rate of the simple poisoning attack (rated by CLIP classifier) correlates strongly with concept sparsity measured by semantic frequency. Results for LD-CC. Same trend observed on SD-V2, SD-XL, DeepFloyd.