

Towards Understanding the Robustness of Self-Consuming Generative Models

Aobo Chen*, Wei Qian*, Chenglin Miao, Mengdi Huai

Iowa State University

{aobochoen, wqi, cmiao, mdhuai}@iastate.edu

Abstract

Recent advances in deep generative models have made it increasingly difficult to distinguish real-world data from model-generated content. Consequently, successive model generations are trained on synthetic data from prior models, forming a self-consuming training loop. Despite being powerful, it also opens a new attack surface, i.e., test-time poisoning attacks, adversaries exploit model vulnerabilities during the inference stage to generate malicious test-time outputs that are subsequently incorporated into future training data. In this paper, for the first time, we propose a new class of malicious test-time poisoning attack against self-consuming generative methods. However, uncovering the failure modes of self-consuming generative models under test-time poisoning attacks is highly challenging due to the need to simultaneously maintain stealthy data manipulations while achieving fine-grained control over the activation and propagation of malicious behaviors across self-consuming training iterations. To address these challenges, our attacks provide fine-grained control and introduce a novel stealth paradigm that achieves both effectiveness and stealthiness by camouflaging stealthy malicious data within the training process and selectively activating concealed attack behaviors at targeted later stages. Notably, our attacks are compatible with various adversarial attack generating algorithms and self-consuming algorithms. Extensive experiments under different settings, including black-box scenarios, demonstrate that self-consuming generative models are broadly vulnerable to the proposed test-time poisoning attacks.

1 Introduction

With the rapid advancements in deep generative models, synthetic data of all varieties is expanding swiftly. Notably, publicly accessible generative

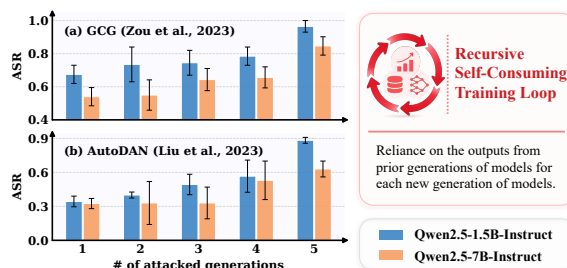


Figure 1: Attack success rates of GCG and AutoDAN in the self-consuming training loop across different generative models. The results show that jailbreak attacks can accumulate and progressively amplify test-time model vulnerabilities across generations.

models (Bansal et al., 2022) have significantly enabled the creation and dissemination of synthetic content at scale, thereby accelerating the flow of synthetic data towards the Internet. Consequently, these synthetic data are inevitably used for successive training of future model generations, which creates a “self-consuming training loop” (Cai et al., 2026; Gillman et al., 2024; Zhu et al., 2024).

In self-consuming training settings, model development proceeds iteratively, where the training data for each new generation of models are from the outputs of the prior generations of models. In practice, such generated synthetic data are getting hard to distinguish from human-generated content (Sadasivan et al., 2023). On the other hand, human-generated data remain scarce and expensive to obtain. This further accelerates reliance on synthetic data, reinforcing the increased popularity of the self-consuming training loop for generative models, especially in data-scarce applications.

Despite the growing interest of the self-consuming training loop, this paradigm introduces new attack surfaces that can be exploited by motivated adversaries. In particular, attackers can first perform test-time adversarial attacks on current models to generate malicious outputs, which are then disseminated online as synthetic data. When such malicious data are subsequently incorporated

*The first two authors contribute equally to this work.

into the training of future models, the attack effects can propagate across generations. This creates a test-time poisoning loop, whereby vulnerabilities at test time are propagated and amplified across successive poisoned model generations. In Figure 1, we show that under self-consuming training loops, for generative language models (including Qwen2.5-7B-Instruct and Qwen2.5-1.5B-Instruct (Yang et al., 2025)), jailbreak attacks (e.g., GCG (Zou et al., 2023) and AutoDAN (Liu et al., 2023)) can accumulate and progressively amplify test-time model vulnerabilities across generations, highlighting unique risks in self-consuming generative models. The attack success rate (ASR) consistently increases as the number of attacked generations grows, indicating that unsafe behaviors are reinforced via self-consuming training loops.

Our goal in this paper is to conduct a comprehensive study on the security vulnerabilities and robustness of self-consuming generative models against test-time poisoning attacks. This setting introduces several unique and severe risks. First, this creates additional pathways for adversarial manipulation and enables the propagation of malicious signals across generations. Particularly, successful adversarial outputs from prior models may be incorporated into future training data, thereby increasing the likelihood of similar unsafe behavior in subsequent model generations. Additionally, this is particularly worse when verifying the legitimacy of generated data from prior models is extremely difficult, since attackers could exploit this by crafting adversarial self-consuming data, which look imperceptible from clean benign data. Further, this inherits vulnerabilities of models related to traditional security attacks, and activates and enhances traditional security attacks via self-consuming training. Moreover, this could undermine conventional defenses, further increasing susceptibility to traditional attacks via the self-consuming training loop.

However, uncovering the failure modes of self-consuming generative models under test-time poisoning attacks is challenging. First, directly attacking multiple self-consuming iterations is prone to detection, thereby compromising stealthiness. Additionally, the source inference problem further complicates attack stealthiness. Specifically, the attacker must ensure that the crafted synthetic data are indistinguishable from the model’s own outputs, so that they are stealthy during subsequent training. Therefore, it is necessary to mislead models into treating adversarially crafted data as self-generated

content, enabling malicious stealthy signals to propagate through self-consuming training loops. Further, the attack should provide the fine-grained control over the timing and mechanism of malicious behaviors. This is practically important, as it allows the attacker to maintain stealth during data collection while selectively activating the attack under specific conditions.

Motivated by the above, we in this paper take the first step toward understanding self-consuming generative models against test-time poisoning attacks. To this end, we design a novel attack framework, which exploits vulnerabilities in the self-consuming training loop to induce attacker-desired behaviors. Our attack involves concealing malicious traces in the loop and activating the malicious attack behaviors on-demand at a later stage. This can provide fine-grained control, and the attacks activated in this way persist much longer than conventional security attacks. To meet the challenges of attack stealthiness, we design a novel stealthy camouflage sample generation algorithm, which covers malicious attack traces effectively. Notably, our proposed attacks can evade existing defenses including harmful data detectors. Our proposed attacks are compatible with a wide range of existing adversarial attack generation methods and self-consuming training algorithms, enabling the inherited malicious behaviors to be propagated and amplified across successive model generations. We also present novel efficient optimization methods to enhance the efficiency of the proposed attacks. We perform extensive evaluations on state-of-the-art self-consuming methods, demonstrating the efficacy of our proposed attacks.

2 Preliminary and Related Work

Notations and Self-Consuming Training Loop.

Formally, let $\mathcal{M}_t$ with parameters θ_t denote the generative model at generation t , and let T be the total number of generations. Let x denote an input (e.g., prompts), y its corresponding generated response, and $z = (x, y)$ the single data sample. Let D denote a dataset with n samples, i.e., $D = \{z_i\}_{i=1}^n$. For the dataset D , the fine-tuning loss for a generative model $\mathcal{M}$ with parameters θ can be defined as $\mathcal{L}_{ft}(D; \theta) = \frac{1}{n} \sum_{(x,y) \in D} \mathcal{L}(x, y; \theta)$, where $\mathcal{L}(x, y; \theta)$ is the pre-defined task-specific training loss on sample (x, y) .

For self-consuming generative models (Shumailov et al., 2024; Briesch et al., 2023), a synthetic dataset D_t is generated using the previous

model $\mathcal{M}_{t-1}$, and the next generation model $\mathcal{M}_t$ is trained from scratch (fine-tuned on the base model) from the following two data cycle designs: *full synthetic data cycle* and *expanding data cycle*. Specifically, in the full synthetic data cycle, the new model $\mathcal{M}_t$ is trained solely on the generated dataset D_t (Alemohammad et al., 2024). This setup captures deployment scenarios where the original training data is inaccessible, such as due to expired data licenses or user data deletion requests (Wang et al., 2025a; Le et al., 2026). Note that synthetic training data can be obtained by using fresh real prompts collected from users. In contrast, in the expanding data cycle (Briesch et al., 2023; Kazdan et al., 2024; Wang et al., 2025c), we accumulate both the original real data D_0 and the generated synthetic data across generations to form the training dataset, i.e., $\mathcal{D}_t = D_0 \cup D_1 \cup \dots \cup D_t$, where D_0 is the initial human-generated dataset. This reflects practical deployments where historical data can be reused during training to improve performance.

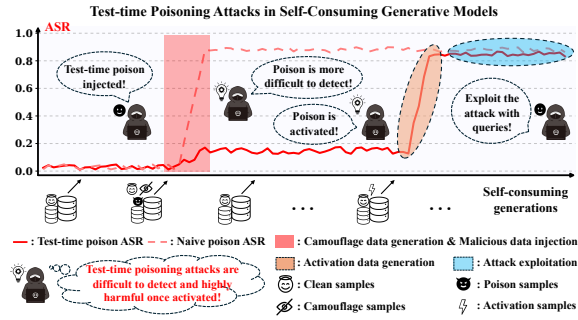


Figure 2: Overview of our proposed test-time poisoning attack in self-consuming generative models.

Related Work. Existing works on self-consuming generative models (Cai et al., 2026; Gillman et al., 2024) largely ignore to identify model vulnerabilities to test-time poisoning attacks, where attackers exploit model weaknesses during the testing stage to generate malicious outputs and subsequently propagate these poisoning generations across successive training cycles. It is worth mentioning that our proposed attacks are different from traditional data poisoning attacks during training (Zhang et al., 2025; He et al., 2025; Li et al., 2024), and evasion attacks (e.g., jailbreak attacks) during testing (Cui et al., 2025). These traditional attacks have been extensively studied in generative models (Raina et al., 2024). For example, (Zhang et al., 2025) studies the robustness of LLMs against training-time poisoning attacks on the poisoned uncured datasets that are scraped from the Web. (Wei and Zhang, 2025) studies the genuine user

preference degradation caused by specifically designed test-time preference collection mechanisms. In contrast, we present a new test-time poisoning attack paradigm, which bridges the gap between inference-time manipulation and training-time poisoning impact, resulting in a novel form of cross-stage vulnerability that is not captured by these threat models. Notably, our attacks are also compatible with these poisoning and evasion attacks, which can be easily integrated within our attacks.

3 Methodology

3.1 Threat Model

In the threat model, we consider the following parties in the attack loop: *the model owner* and *the attacker*. For the model owner, in the self-consuming training loop setting, new models are trained on data generated by the previous generation. We assume that attackers cannot change the initial dataset D_0 and the initial model $\mathcal{M}_0$. This reflects the self-consuming scenario, where attackers can only query the current model to craft malicious data during the self-consuming training loop. We also assume that attackers cannot modify the self-consuming training algorithms and model architectures. In this paper, we study both the white-box and black-box settings. Specifically, in the white-box setting, we make the assumption that the adversary possesses complete information about the system (including the model architecture and parameters of the well-trained model). Note that many existing literature (Wang et al., 2025b; Chen et al., 2026; Fu et al., 2025; Qian et al., 2023) employ a white-box model for a comprehensive analysis of worst-case attack scenarios. In the black-box setting, in addition to the above attack incapacabilities and restrictions, we also assume that the adversary does not have any prior knowledge about the target well-trained model and the initial dataset.

3.2 Attack Formulation

Overview. Our proposed attack consists of a composition of four stages as indicated in Figure 2, including *camouflage data generation*, *malicious data injection*, *activation data generation*, and *attack exploitation*. Specifically, the camouflage data generation stage is a key improvement. In conventional adversarial attacks, only adversarial samples are generated. In our attacks, camouflage samples are crafted along with adversarial samples for fine-grained activation control and stealthiness purposes. Note that adversarial samples are generated by fol-

lowing existing adversarial attacks. The malicious data injection stage is the same as conventional poisoning attacks. Note that the injected malicious attack behaviors are hidden, since we use the camouflage data to keep the attack effects as low as possible for stealthiness before launching the attack at later generations. Different from existing malicious attacks on generative models, our attack uses an explicit activation stage to enable the previously injected hidden attacks instead of assuming the attacks are active all the time. In this stage, the adversary uses the crafted stealthy activation data to remove the camouflage effects. The attack exploitation stage is the same as conventional malicious attacks. Once the victim model has been activated, attackers can exploit the inherited malicious behaviors to induce attacker-desired outputs and trigger harmful behaviors under specific inputs. Below, we will detail the four components.

Camouflage Data Generation. Here, we describe the camouflage data generation process, where camouflage samples are jointly crafted with attack data at generation t_1 within the self-consuming loop to achieve fine-grained activation control while enhancing the stealthiness of attacks. Without loss of generality, we below use the full synthetic cycle to present our attacks. At generation t_1 , attackers can use fresh real prompts to collect a synthetic dataset $\tilde{D}_{t_1}$. The collected dataset is then partitioned into an attack dataset $D_{t_1}^{atk}$ of size n^{atk} and a camouflage dataset $D_{t_1}^{cam}$ of size n^{cam} , i.e., $\tilde{D}_{t_1} = D_{t_1}^{atk} \cup D_{t_1}^{cam}$. Next, a test-time adversarial attack algorithm $\mathcal{T}(\cdot)$, which optimizes the adversarial objective $\mathcal{L}_{adv}$ for inducing attacker-desired results, is applied to $D_{t_1}^{atk}$ to construct the attack data $\tilde{D}_{t_1}^{atk} = \{\mathcal{T}(x_i, y_i)\}_{i=1}^{n^{atk}}$. Notably, our framework is generally compatible with existing test-time adversarial attack generation algorithms.

However, directly applying the attack dataset $\tilde{D}_{t_1}^{atk}$ may introduce obvious abnormal model behaviors, making the attack easier to detect. Therefore, it is crucial to further conceal the malicious effects introduced by $\tilde{D}_{t_1}^{atk}$. Meanwhile, directly adopting the original camouflage dataset $D_{t_1}^{cam}$ is often insufficient to effectively mask the attack behaviors induced by the malicious attack data. To address this challenge, we optimize the camouflage perturbation $\Delta^{cam} = \{\delta_i^{cam}\}_{i=1}^{n^{cam}}$ to preserve benign model behaviors for attack stealthiness. The resulting camouflaged dataset is defined as $\tilde{D}_{t_1}^{cam} = \{(x_i + \delta_i^{cam}, y_i)\}_{i=1}^{n^{cam}}$. Further, attack

stealthiness may still be compromised if these perturbations are not carefully designed, since large instance-wise perturbations can be easily detected. Importantly, it is also crucial to ensure that these crafted synthetic data are indistinguishable from the model’s own outputs, thereby misleading the model into treating malicious data as self-generated content and enabling stealthy attack signals to propagate through the self-consuming training loop. Let $D_{t_1}^{cln}$ denote the clean data at generation t_1 . To address these challenges, we formulate the below optimization for the camouflage data generation

$$\begin{aligned} \Delta^{cam} &= \arg \max_{\Delta^{cam}} \mathcal{L}_{adv}(\tilde{D}_{t_1}^{atk}; \theta_{t_1}(\tilde{D}_{t_1})) \\ \text{s.t.}, \quad &\forall i \in [n^{cam}], |\delta_i^{cam}| \leq \rho^{cam}, \quad (1) \\ &\tilde{D}_{t_1}^{cam} = \{(x_i + \delta_i^{cam}, y_i)\}_{i=1}^{n^{cam}}, \\ &\tilde{D}_{t_1} = \tilde{D}_{t_1}^{atk} \cup \tilde{D}_{t_1}^{cam} \cup D_{t_1}^{cln}, \\ &\theta_{t_1}(\tilde{D}_{t_1}) = \arg \min_{\theta} \mathcal{L}_{ft}(\tilde{D}_{t_1}; \theta), \\ &\forall (x, y) \in \tilde{D}_{t_1}^{cam}, (x, y) \sim p_{t_1}, \end{aligned}$$

where ρ^{cam} is pre-defined, p_{t_1} is the model-generated distribution, and the last constraint is used to enforce that these data remain consistent with the distribution. In Eq. (1), the camouflage data is constructed by maximizing the loss $\mathcal{L}_{adv}$, thereby encouraging the model to behave normally during fine-tuning while concealing attack effects.

Nevertheless, it is challenging to solve the above optimization in Eq. (1). Note that Eq. (1) defines a bi-level optimization problem, where the outer maximization over Δ^{cam} depends on the model parameters θ_{t_1} , which are obtained as the solution to the inner optimization problem $\theta_{t_1} = \arg \min_{\theta} \mathcal{L}_{ft}(\tilde{D}_{t_1}; \theta)$. Directly solving this problem requires computing gradients by backpropagating through the fine-tuning process itself, which is computationally prohibitive for large language models. To address this challenge, we approximate the above optimization using a gradient alignment strategy inspired by the gradient matching technique (Geiping et al., 2020). Another challenge is the inherent discrete nature of Δ^{cam} in the jail-break attack scenario. To address this challenge, we iteratively optimize the camouflage perturbation by selecting and updating individual token coordinates according to their gradient-based importance scores. Based on the above, we approximate the bi-level optimization by fixing θ as the model parameters obtained after training on the dataset $\tilde{D}_{t_1}^{atk} \cup D_{t_1}^{cln}$, and optimize the camouflage

perturbation by minimizing the below objective

$$\begin{aligned} \mathcal{L}_1(\Delta^{cam}; \theta) = & \langle \nabla_{\theta} \mathcal{L}_{adv}(\tilde{D}_{t_1}^{atk}; \theta), \\ & \nabla_{\theta} \mathcal{L}_{ft}(\tilde{D}_{t_1}^{cam}; \theta) \rangle / (\| \nabla_{\theta} \mathcal{L}_{adv}(\tilde{D}_{t_1}^{atk}; \theta) \| \\ & \cdot \| \nabla_{\theta} \mathcal{L}_{ft}(\tilde{D}_{t_1}^{cam}; \theta) \|). \end{aligned} \quad (2)$$

Minimizing $\mathcal{L}_1(\Delta^{cam}; \theta)$ encourages the training update on $\tilde{D}_{t_1}^{cam}$ to move the model parameters toward a direction that also increases $\mathcal{L}_{adv}$, leading to a more stealthy attack. As aforementioned, a further challenge arises from the self-consuming nature of the process. When the optimized camouflage data $\tilde{D}_{t_1}^{cam}$ are not consistent with the model-generated data distribution p_{t_1} , they can be easily detected. However, it is challenging to directly incorporate the last constraint in Eq. (1), since the underlying distribution is difficult to estimate in practice. To address this, we introduce a regularization term that encourages the perturbed camouflage dataset to be indistinguishable from the model's own outputs. We denote this consistency regularization term as $\mathcal{L}_{con}(\Delta^{cam}; \theta)$. The form of the regularization term is task-dependent. For example, in the scenario of integrating jailbreak attacks within our framework, we formulate

$$\begin{aligned} \mathcal{L}_{con}(\Delta^{cam}; \theta) = & -\frac{1}{n^{cam}} \sum_{i=1}^{n^{cam}} \\ & \sum_{j=1}^{|y_i|} \log p_{\theta}(y_{i,j} | (x_i || \delta_i^{cam}), y_{i,<j}). \end{aligned} \quad (3)$$

In the above equation, the regularization term leads the perturbed camouflage inputs to produce outputs that remain close to the model's own distribution, thereby making the optimized camouflage data less distinguishable from the model-generated data. More details about the regularization term can be found in the full version. Combined with the formulated objective in Eq. (2), we have

$$\mathcal{L}_2(\theta) = \mathcal{L}_1(\Delta^{cam}; \theta) + \mathcal{L}_{con}(\Delta^{cam}; \theta). \quad (4)$$

By minimizing Eq. (4), the optimized camouflage perturbation can efficiently and effectively hide the attack effects caused by $\tilde{D}_{t_1}^{atk}$ while enforcing consistency with the model-generated data distribution.

Malicious Data Injection. In this stage, the attacker uploads all the crafted data (i.e., the attack samples $\tilde{D}_{t_1}^{atk}$ and the camouflage samples $\tilde{D}_{t_1}^{cam}$), and waits for the model owner to train the victim model θ_{t_1} . In this way, the hidden malicious behaviors can be injected into the victim model through the self-consuming iterations. Note that during this model training, these crafted data will be together

with other clean data for the model training. This reflects the unique security risks of self-consuming generative models.

Activation Data Generation. Having the attack and camouflage samples injected, the attacker can choose a desired time t_2 to activate the injected attack behaviors. However, it is challenging for the attacker to activate the harmful behavior inside the model only by injecting clean data. To address this challenge, we first propose to use fresh real prompts to collect a synthetic dataset $D_{t_2}^{act}$ at generation t_2 . Then, we will craft activation perturbations $\Delta^{act} = \{\delta_i^{act}\}_{i=1}^{n^{act}}$ to activate the harmful information injected by $\tilde{D}_{t_1}^{atk}$ and suppress the influence of camouflage data $\tilde{D}_{t_1}^{cam}$. The resulting activation dataset is denoted as $\tilde{D}_{t_2}^{act} = \{(x_i + \delta_i^{act}, y_i)\}_{i=1}^{n^{act}}$. To this end, we revisit the adversarial objective $\mathcal{L}_{adv}$ so that these perturbations can amplify the inherited attack effect. Meanwhile, the camouflage effect caused by the camouflage data $\tilde{D}_{t_1}^{cam}$ needs to be suppressed, which can be achieved by increasing the training loss $\mathcal{L}_{ft}$ on $\tilde{D}_{t_1}^{cam}$. Similar to the camouflage data generation, to avoid detection, it is also crucial to ensure that the activation data $\tilde{D}_{t_2}^{act}$ remains stealthy in terms of the perturbation magnitude and underlying distribution. Let $D_{t_2}^{cln}$ denote the other collected clean data. Based on these, we formulate the below optimization framework

$$\begin{aligned} \Delta^{act} = & \arg \min_{\Delta^{act}} \mathcal{L}_{adv}(\tilde{D}_{t_1}^{atk}; \theta_{t_2}) - \mathcal{L}_{ft}(\tilde{D}_{t_1}^{cam}; \theta_{t_2}) \\ \text{s.t. } & \theta_{t_2} = \arg \min_{\theta} \mathcal{L}_{ft}(\tilde{D}_{t_2}; \theta), \\ & \forall i \in [n^{act}], |\delta_i^{act}| \leq \rho^{act}, \\ & \forall (x, y) \in \tilde{D}_{t_2}^{act}, (x, y) \sim p_{t_2}, \end{aligned} \quad (5)$$

where ρ^{act} is a pre-defined value constraining the magnitude of the perturbation, and $\tilde{D}_{t_2} = D_{t_2}^{cln} \cup \tilde{D}_{t_2}^{act}$. Here, we define $\mathcal{L}_{act}(\tilde{D}_{t_1}^{atk}; \tilde{D}_{t_1}^{cam}; \theta) = \mathcal{L}_{adv}(\tilde{D}_{t_1}^{atk}; \theta) - \mathcal{L}_{ft}(\tilde{D}_{t_1}^{cam}; \theta)$. Although these attack and camouflage samples are not directly used for training in generation t_2 , their effects persist through the inherited model parameters and the data distributions induced by earlier generations. Therefore, by minimizing $\mathcal{L}_{act}$, the attacker steers the model training toward reducing the adversarial loss on the attack data, while increasing the loss on the camouflage data to cancel its masking effect. In this way, the hidden harmful behavior can be activated at the chosen generation. Using a similar technique as in camouflage data generation step,

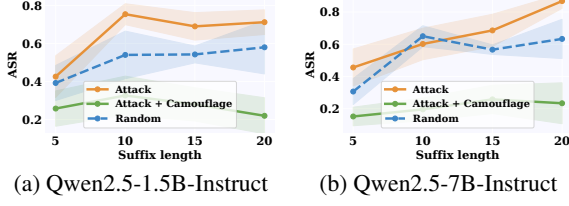


Figure 3: Attack performance with and without camouflage data on large language models.

we approximate the bi-level optimization by fixing θ as the model parameters obtained after training on the dataset $D_{t_2}^{cln}$, which is formulated as

$$\begin{aligned} \mathcal{L}_3(\Delta^{act}; \theta) &= \langle \nabla_{\theta} \mathcal{L}_{act}(\tilde{D}_{t_1}^{atk}; \tilde{D}_{t_1}^{cam}; \theta), \\ &\quad \nabla_{\theta} \mathcal{L}_{ft}(\tilde{D}_{t_2}^{act}; \theta) \rangle / (\| \nabla_{\theta} \mathcal{L}_{act}(\tilde{D}_{t_1}^{atk}; \tilde{D}_{t_1}^{cam}; \theta) \| \\ &\quad \cdot \| \nabla_{\theta} \mathcal{L}_{ft}(\tilde{D}_{t_2}^{act}; \theta) \|). \end{aligned} \quad (6)$$

Additionally, similar to the above camouflage data generation step, we introduce a regularizer $\mathcal{L}_{con}(\Delta^{act}; \theta)$. In this way, we can optimize the activation perturbations by maximizing $\mathcal{L}_4(\Delta^{act}; \theta) = \mathcal{L}_3(\Delta^{act}; \theta) - \mathcal{L}_{con}(\Delta^{act}; \theta)$. By optimizing $\mathcal{L}_4(\Delta^{act}; \theta)$, we can craft activation perturbations to effectively activate the previously injected hidden malicious attack behaviors.

Attack Exploitation. After the hidden attack behaviors are activated, attackers can utilize the attack behaviors in the model at any time. For example, attackers can carefully design the adversarial loss $\mathcal{L}_{adv}$ for the clean-input exploitation, where the victim model directly returns the attacker-desired results when prompted with a clean input. Additionally, the adversarial objective can also be specifically designed for perturbed-input attack exploitation, where the malicious behaviors are triggered when carefully crafted perturbations are applied to the input. Such a design enables the attacker to achieve more fine-grained control over attack activation and malicious behavior triggering.

Discussions. Here, we provide more discussions for our proposed attacks. First, in the black-box setting, the attacker lacks the knowledge of the target well-trained model, including the model architecture and parameters. Instead, the attacker can generate the attack, camouflage, and activation data using surrogate models. For instance, we can optimize adversarial suffixes over multiple surrogate models (Zou et al., 2023; Liao and Sun, 2024). This allows the attacker to construct malicious data that can effectively transfer to the target black-box victim model. Additionally, our attacks can also be applied to the expanding data cycle. Further, our

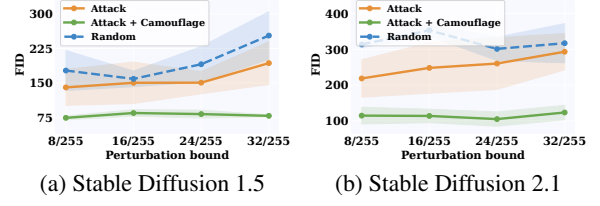


Figure 4: Attack performance with and without camouflage data on diffusion models.

proposed attacks can be generalized to different attack goals by specifying different adversarial objectives $\mathcal{L}_{adv}$. Moreover, our attack framework can be extended by incorporating fine-grained uncertainty-aware memorization metrics to guide the injection of hidden attacks and the generation of camouflage and activation data. One promising direction is to leverage conformal prediction (Li et al., 2026; Ye et al., 2024; Chen et al., 2024; Angelopoulos and Bates, 2023) to quantify the memorization degrees associated with individual samples or attack behaviors with statistical coverage guarantees. Furthermore, in the black-box setting, we will also investigate the use of machine unlearning (Qian et al., 2025; Foster et al., 2024; Chen et al., 2025a) as an efficient mechanism for steering surrogate models toward desired attack behaviors. Rather than repeatedly retraining surrogate models from scratch when exploring different attack strategies, unlearning can selectively estimate the influence of specific data. More detailed discussions can be found in the full version of the paper.

4 Experiments

In this section, we conduct extensive experiments to evaluate the effectiveness of our proposed attacks. More experimental details and results are deferred to the full version of the paper.

Datasets and Models. In experiments, we adopt datasets for different test-time poisoning attack objectives. Specifically, for jailbreak attacks, we use AdvBench (Zou et al., 2023) and WildChat (Zhao et al., 2024). For misclassification attacks, we use IMDB (Maas et al., 2011) and AGNews (Zhang et al., 2015). In image generation attacks, we use ImageNet (Deng et al., 2009). For language models, we adopt Qwen2.5-7B-Instruct, Qwen2.5-1.5B-Instruct (Yang et al., 2025), Vicuna-13B-v1.5, Vicuna-7B-v1.5 (Zheng et al., 2023), Llama-3.2-3B-Instruct (Grattafiori et al., 2024), and Phi-3.5-mini-instruct (Abdin et al., 2024). For diffusion models, we adopt Stable Diffusion 1.5 and Stable Diffusion 2.1 (Rombach et al., 2022).

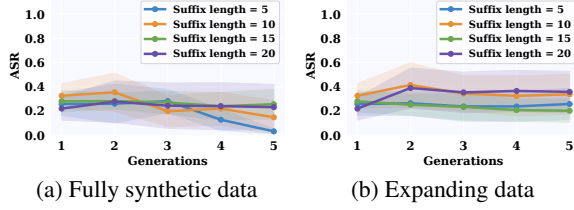


Figure 5: Attack performance with camouflage data under various data cycles in the jailbreak setting.

Self-Consuming Training Methods. In experiments, we consider representative self-consuming training methods in the fully synthetic data cycle (Alemohammad et al., 2024) and the expanding data cycle (Briesch et al., 2023; Kazdan et al., 2024; Gillman et al., 2024). For the fully synthetic data cycle, (Alemohammad et al., 2024) considers that each generation is iteratively retrained on data generated by the previous model. In the expanding data cycle, (Briesch et al., 2023; Kazdan et al., 2024) study retraining on accumulated data using a mixture of real and synthetic data across generations, while (Gillman et al., 2024) adopts incremental fine-tuning of the current-generation model.

Evaluation Metrics. We use multiple metrics to evaluate attack effectiveness and stealthiness. We report attack success rate (ASR), which measures the proportion of successful attack cases, and Fréchet inception distance (FID) (Heusel et al., 2017), which measures generated image quality and diversity. We analyze loss convergence and optimization running time to assess attack efficiency. To quantify distributional changes, we measure total variation distance. For detection-based defenses, we use the area under the curve (AUC) to evaluate detection performance. We also provide visualizations to qualitatively illustrate the attack effects.

Baselines. Since there is no existing work, we in experiments adopt the baselines of random perturbations for the camouflage and activation data during the self-consuming training loop. Note that our attacks involve both stealthy camouflage generation and fine-grained activation control.

4.1 Attack and Camouflage

First, we investigate the effectiveness of the attack and camouflage data in our proposed attacks. Figure 3 presents the attack performance with and without camouflage data in the jailbreak setting. For each generation, we construct a training set of 100 samples. In the first generation, we use 10% attack samples and 20% camouflage samples constructed from AdvBench, while the remaining

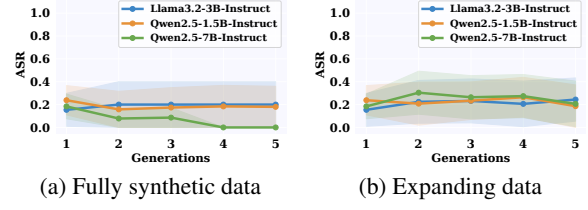


Figure 6: Attack performance with camouflage data under various data cycles in the misclassification setting.

samples are clean data from WildChat. We evaluate the resulting models on 100 clean test samples over five runs and report the mean ASR with standard error. The attack data are generated using GCG (Zou et al., 2023) with varying adversarial suffix lengths, while the camouflage data are optimized with a constraint ρ^{cam} of suffix length 5. As shown, our attacks significantly increase the ASR once poisoned samples enter the retraining loop, and the ASR continues to increase as the adversarial suffix length grows. In contrast, incorporating carefully constructed camouflage samples substantially reduces the ASR across different attack strengths. The random baseline, which appends random suffixes to camouflage data, is less effective in mitigating the attack effects. These results indicate that our proposed camouflage strategy can effectively conceal the impact of malicious attacks.

Then, we evaluate our attacks on image generation with diffusion models. We construct attack data using the encoder attack (Salman et al., 2023), which perturbs the latent representation to generate images that are irrelevant and unrealistic. We further construct camouflage data from ImageNet and inject them into the self-consuming retraining loop in the first generation. Figure 4 presents the attack performance of the image generation under different perturbation bound constraints for the attack data, while the camouflage data are optimized with a constraint of $4/255$. As shown, our proposed attacks substantially increase the FID once poisoned samples are introduced, indicating their effectiveness in degrading the diffusion model’s ability to generate high-quality images. In contrast, when the constructed camouflage samples are incorporated, the FID decreases accordingly, surpassing the random baseline. This shows that the harmful effect can be effectively concealed in diffusion models.

Furthermore, we examine camouflage effectiveness throughout the self-consuming training loop. Figure 5 reports the attack performance with camouflage data under fully synthetic data and expanding data cycles with retraining, following the

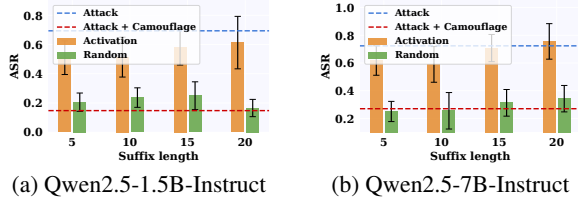


Figure 7: Attack performance under test-time poisoning attacks with activation data.

Qwen2.5-1.5B-Instruct setup in Figure 3. Figure 6 shows the corresponding results in the misclassification setting, where we focus on misclassifying target labels on IMDB. As shown, injecting attack and camouflage data in the first iteration keeps attack performance low and relatively stable across subsequent generations, suggesting that our camouflage data can continuously suppress the attack effect while maintaining stealthiness throughout the self-consuming training process.

4.2 Activation and Attack Exploitation

Here, we investigate the effectiveness of activation data and final attack exploitation. Figure 7 reports the attack performance of activation data using varying activation suffix length constraints ρ^{act} . We sample the activation data from AdvBench. Following Figure 5a, the attacker introduces carefully optimized activation data in the fifth generation to trigger the hidden poisoning effect. As shown, the ASR remains low without activation data but increases significantly once activation data are introduced, surpassing the random baseline and approaching the attack baseline. These results show that activation data effectively counteract the camouflage effect and activate the hidden attack behavior within the self-consuming loop. Importantly, this experiment is conducted on aligned models that are expected to suppress harmful responses. Thus, the increase in ASR suggests that conventional alignment-based defenses are insufficient to mitigate test-time poisoning risks.

Additionally, we evaluate our attacks under two distinct self-consuming scenarios across the full self-consuming training loops. On AdvBench, we perform jailbreak attacks against Qwen2.5-7B-Instruct and Qwen2.5-1.5B-Instruct. On ImageNet, we perform image generation attacks against Stable Diffusion 1.5 and 2.1. The results are summarized in Figure 8. Specifically, for AdvBench, we set the suffix lengths for the attack and activation data to 10 and 20; for ImageNet, we set the perturbation bounds for both the attack and activation data to 32/255. In the figure, we adopt the random base-

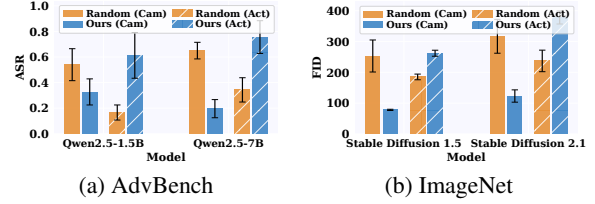


Figure 8: Attack performance of the proposed malicious fine-grained poisoning attacks.

line. The attack-only ASRs are 0.75 for Qwen2.5-1.5B-Instruct and 0.60 for Qwen2.5-7B-Instruct, while the attack-only FIDs are 193 for Stable Diffusion 1.5 and 293 for Stable Diffusion 2.1. As shown, on the Qwen2.5-7B-Instruct model, our camouflage reduces the ASR from 0.60 to approximately 0.20, while activation reactivates it to 0.76. These results indicate that our method conceals and selectively reactivates malicious behaviors across generative models, enabling fine-grained control over test-time poisoning attacks.

4.3 Black-box Setting

Here, we examine our proposed attacks in the black-box setting. Figure 9 presents the attack performance in the jailbreak setting, where the attacker constructs attack, camouflage, and activation data using adversarial suffixes optimized on multiple surrogate models, including Vicuna-7B-v1.5 and Vicuna-13B-v1.5. As shown, test-time poisoning attacks can transfer the generated malicious data to the target black-box model and achieve both concealment and attack exploitation, even when the target model differs from the surrogate models. These results demonstrate that our test-time poisoning attacks can effectively preserve the malicious attack effect under the black-box setting.

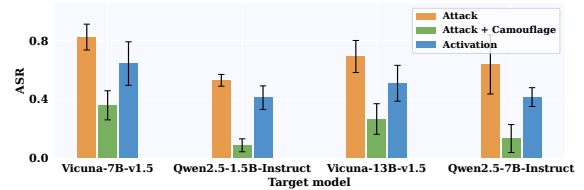


Figure 9: Attack performance during self-consuming loop in the black-box setting.

4.4 Ablation Study

Here, we study the impact of the proposed regularization terms under the activation stage. Figure 10a compares the total variation (TV) distance between the model-generated data and perturbed activation data across different suffix lengths, following Figure 7. As shown in the figure, without the regularization term, the perturbed activation

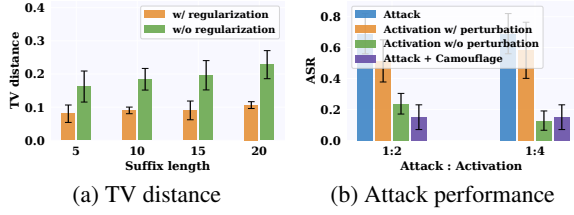


Figure 10: Effect of the regularization term and perturbation in the activation data generation stage.

data exhibits a large TV distance. In contrast, our proposed regularization term reduces the distribution change by over half, demonstrating its effectiveness in aligning the perturbed data distribution with the model-generated distribution.

Then, we examine the impact of the optimized perturbations on the activation data. Figure 10b compares the attack performance when the activation data are used with and without perturbations, following the setup in Figure 7. As shown, merely adding activation data without perturbations fails to trigger the attack during the exploitation stage. In contrast, our optimized perturbations effectively activate the hidden poisoning effect, enabling stronger exploitation of subsequent attacks.

Additionally, we evaluate more conventional defenses against our attacks. We adopt two detection-based defenses, FJD (Chen et al., 2025b) and GradSafe (Xie et al., 2024), to detect jailbreak attacks during exploitation. Figure 11 reports AUC for detecting jailbreak prompts under varying activation suffix lengths, following Figure 7. As shown, these methods fail to detect jailbreak prompts on the final model, but our method still achieves a high ASR in generating harmful content. These results indicate that our proposed attacks can undermine conventional defenses and further amplify model vulnerabilities through self-consuming training.

Dataset	Model	Baseline (Syn)	Ours (Syn)	Baseline (Real)	Ours (Real)
IMDB	Qwen-2.5-1.5B	0.517	0.825	0.488	0.676
	Qwen-2.5-7B	0.549	0.822	0.507	0.742
XSum	Qwen-2.5-1.5B	0.507	0.851	0.501	0.725
	Qwen-2.5-7B	0.575	0.844	0.514	0.705

Table 1: Attack AUC of the proposed attacks (Ours) versus directly executing MIA on the single generative model (Baseline) on IMDB and XSum. “Syn” denotes synthetic data, while “Real” denotes real data.

Moreover, we investigate privacy vulnerabilities in self-consuming generative models by developing a membership inference attack (MIA) tailored to the self-consuming training loop. Given two suc-

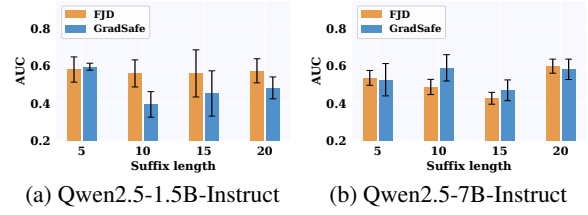


Figure 11: Detection performance using conventional defenses against test-time poisoning attacks.

cessive generations, θ_t and θ_{t+1} , our goal is to determine whether a target sample $z = (x, y)$ belongs to the training data D_{t+1} . Unlike conventional MIA methods that observe only a single model, our attacks exploit the loss discrepancy across generations, i.e., $\mathcal{L}(x, y; \theta_t) - \mathcal{L}(x, y; \theta_{t+1})$, which provides an additional discriminative signal for membership inference. Table 1 compares our attacks with a baseline that observes only the single generative model. We report the AUC in the expanding data cycle, with equal proportions of synthetic and real data at generation $t + 1$, using IMDB (Maas et al., 2011) and XSum (Narayan et al., 2018) with Qwen-2.5-1.5B-Instruct and Qwen-2.5-7B-Instruct. Note that synthetic data are generated by the generative model θ_t . We find that our attacks consistently outperform the baseline, showing that the loss discrepancy between θ_t and θ_{t+1} provides a stronger membership signal than observing the single model alone. We also find that synthetic data are generally more vulnerable to MIA than real data, likely due to stronger overfitting to synthetic data. These results highlight the effectiveness of our proposed attack strategy in self-consuming generative models.

5 Conclusion

To the best of our knowledge, we are the first to examine the security vulnerability and robustness of self-consuming generative models against test-time poisoning attacks during the self-consuming training loop. We introduce a novel class of attacks, which can not only strengthen attack stealthiness but also improve fine-grained attack persistence and activation in the self-consuming loop, thus highlighting previously unexplored vulnerabilities of generative models under the self-consuming loop. We also introduce efficient and effective optimization methods. The reported extensive experimental results demonstrate the effectiveness of our proposed attacks. We believe that our findings provide valuable insights into how to design secure and robust methods to defend against such test-time poisoning attacks in the future.

6 Limitations

This work provides valuable insights into the security vulnerabilities of self-consuming generative models to malicious test-time poisoning attacks. However, our work still has some limitations. First, our experiments are still limited to a finite set of model architectures, datasets, and attack settings. In the future, we will extend the evaluation to broader experimental settings, including diverse model architectures, various real-world deployment scenarios, and more attack settings. Additionally, while this work focuses on exposing the security vulnerabilities and malicious behaviors of test-time poisoning attacks, developing effective defense strategies remains an important direction for future research. Further, in the future, we will also explore more evaluation metrics to study the trade-off between model utility, safety, and vulnerability under the self-consuming training loops.

7 Broader Impact and Ethics Statement

In this work, we introduce a new class of malicious test-time poisoning attacks against self-consuming generative models. Such malicious test-time poisoning attacks highlight important security risks arising from the cross-generation propagation of malicious attack behaviors. Notably, our attack is also compatible with traditional poisoning and evasion attacks, which can be easily integrated within our attacks. Therefore, these attacks may further amplify existing security vulnerabilities, undermine conventional defenses, and increase the long-term unsafe behaviors across successive model generations. While the proposed attacks could potentially be misused to degrade model reliability and amplify harmful behaviors in downstream applications, our goal is to expose these vulnerabilities in a controlled research setting and motivate the development of strong safeguards. This work provides valuable insights for the research community by showing that malicious test-time interactions with self-consuming generative models can influence future models through the self-consuming loop, creating a new malicious attack paradigm. We believe that the proposed test-time poisoning attacks highlight unique security risks associated with self-consuming generative models and the need for future research on robust defense mechanisms. Ultimately, this work contributes to the development of more reliable self-consuming methods for various real-world applications.

8 AI Assistance in Writing

In preparing this manuscript, we used multiple AI tools, including GPT-5.5 and Gemini 3.1, only to improve the clarity, fluency, and readability of author-written text. These tools are not used to generate research ideas, design experiments, conduct analyses, interpret results, or formulate conclusions. All scientific, methodological, and technical contributions were made by the human authors, who reviewed and edited the final manuscript and take full responsibility for its content.

Acknowledgments

This work is supported in part by the US National Science Foundation under grants CNS-2350332 and IIS-2442750. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

- Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Qin Cai, Vishrav Chaudhary, Dong Chen, Dongdong Chen, and 110 others. 2024. [Phi-3 technical report: A highly capable language model locally on your phone](#). *Preprint*, arXiv:2404.14219.
- Sina Alemohammad, Josue Casco-Rodriguez, Lorenzo Luzi, Ahmed Imtiaz Humayun, Hossein Babaei, Daniel LeJeune, Ali Siahkoohi, and Richard Baraniuk. 2024. Self-consuming generative models go mad. In *International Conference on Learning Representations*, volume 2024, pages 53581–53608.
- Anastasios N Angelopoulos and Stephen Bates. 2023. Conformal prediction: A gentle introduction. *Foundations and Trends in Machine Learning*, 16(4):494–591.
- Hritik Bansal, Da Yin, Masoud Monajatipoor, and Kai-Wei Chang. 2022. How well can text-to-image generative models understand ethical natural language interventions? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1358–1370.
- Martin Briesch, Dominik Sobania, and Franz Rothlauf. 2023. Large language models suffer from their own output: An analysis of the self-consuming training loop. *arXiv preprint arXiv:2311.16822*.
- Zhongteng Cai, Yaxuan Wang, Yang Liu, and Xueru Zhang. 2026. Stabilizing self-consuming diffusion

- models with latent space filtering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 19844–19852.
- Aobo Chen, Yangyi Li, Wei Qian, Kathryn Morse, Chenglin Miao, and Mengdi Huai. 2024. Modeling and understanding uncertainty in medical image classification. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 557–567. Springer.
- Aobo Chen, Yangyi Li, Chenxu Zhao, and Mengdi Huai. 2025a. [A survey of security and privacy issues of machine unlearning](#). *AI Magazine*, 46(1):e12209.
- Aobo Chen, Chenxu Zhao, Chenglin Miao, and Mengdi Huai. 2026. Towards unveiling vulnerabilities of large reasoning models in machine unlearning. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 512–524. Springer.
- Guorui Chen, Yifan Xia, Xiaojun Jia, Zhijiang Li, Philip Torr, and Jindong Gu. 2025b. Llm jailbreak detection for (almost) free! In *EMNLP (Findings)*, pages 5777–5807.
- Tiehan Cui, Yanxu Mao, Peipei Liu, Congying Liu, and Datao You. 2025. Exploring jailbreak attacks on llms through intent concealment and diversion. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 20754–20768.
- Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.
- Jack Foster, Stefan Schoepf, and Alexandra Brintrup. 2024. Fast machine unlearning without retraining through selective synaptic dampening. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 12043–12051.
- Shuangjie Fu, Du Su, Beining Huang, Fei Sun, Jingang Wang, Wei Chen, Huawei Shen, and Xueqi Cheng. 2025. Jailbreak llms through internal stance manipulation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 15455–15470.
- Jonas Geiping, Liam Fowl, W Ronny Huang, Wojciech Czaja, Gavin Taylor, Michael Moeller, and Tom Goldstein. 2020. Witches’ brew: Industrial scale data poisoning via gradient matching. *arXiv preprint arXiv:2009.02276*.
- Nate Gillman, Michael Freeman, Daksh Aggarwal, Chia-Hong Hsu, Calvin Luo, Yonglong Tian, and Chen Sun. 2024. [Self-correcting self-consuming loops for generative model training](#). In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 15646–15677. PMLR.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.
- Pengfei He, Han Xu, Yue Xing, Hui Liu, Makoto Yamada, and Jiliang Tang. 2025. Data poisoning for in-context learning. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 1680–1700.
- Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30.
- Joshua Kazdan, Rylan Schaeffer, Apratim Dey, Matthias Gerstgrasser, Rafael Rafailov, David L Donoho, and Sanmi Koyejo. 2024. Collapse or thrive? perils and promises of synthetic data in a self-generating world. *arXiv preprint arXiv:2410.16713*.
- Tuan Le, Wei Qian, and Mengdi Huai. 2026. Selective forgetting for large reasoning models. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 412–423. Springer.
- Yangyi Li, Aobo Chen, Wei Qian, Chenxu Zhao, Divya Lidder, and Mengdi Huai. 2024. Data poisoning attacks against conformal prediction. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 27563–27574. PMLR.
- Yangyi Li, Chenxu Zhao, and Mengdi Huai. 2026. Quantifying and understanding uncertainty in large reasoning models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 32741–32754.
- Zeyi Liao and Huan Sun. 2024. [AmpleGCG: Learning a universal and transferable generative model of adversarial suffixes for jailbreaking both open and closed LLMs](#). In *First Conference on Language Modeling*.
- Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. 2023. Autodan: Generating stealthy jailbreak prompts on aligned large language models. *arXiv preprint arXiv:2310.04451*.
- Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. [Learning word vectors for sentiment analysis](#). In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

- Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 1797–1807.
- Wei Qian, Chenxu Zhao, Wei Le, Meiyi Ma, and Mengdi Huai. 2023. Towards understanding and enhancing robustness of deep learning models against malicious unlearning attacks. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1932–1942.
- Wei Qian, Chenxu Zhao, Yangyi Li, Wenqian Ye, and Mengdi Huai. 2025. Towards unveiling predictive uncertainty vulnerabilities in the context of the right to be forgotten. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pages 5130–5135.
- Vyas Raina, Adian Liusie, and Mark Gales. 2024. Is llm-as-a-judge robust? investigating universal adversarial attacks on zero-shot llm assessment. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7499–7517.
- Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695.
- Vinu Sankar Sadasivan, Aounon Kumar, Sriram Balasubramanian, Wenxiao Wang, and Soheil Feizi. 2023. Can ai-generated text be reliably detected? *arXiv preprint arXiv:2303.11156*.
- Hadi Salman, Alaa Khaddaj, Guillaume Leclerc, Andrew Ilyas, and Aleksander Madry. 2023. Raising the cost of malicious ai-powered image editing. *arXiv preprint arXiv:2302.06588*.
- Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Nicolas Papernot, Ross Anderson, and Yarin Gal. 2024. Ai models collapse when trained on recursively generated data. *Nature*, 631(8022):755–759.
- Yaxuan Wang, Chris Yuhao Liu, Quan Liu, Jinglong Pang, Wei Wei, Yujia Bao, and Yang Liu. 2025a. Dragon: Guard llm unlearning in context via negative detection and reasoning. *arXiv preprint arXiv:2511.05784*.
- Yidan Wang, Yanan Cao, Yubing Ren, Fang Fang, Zheng Lin, and BinXing Fang. 2025b. Pig: Privacy jailbreak attack on llms via gradient-based iterative in-context optimization. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9645–9660.
- Ze Wang, Zekun Wu, Yichi Zhang, Xin Guan, Navya Jain, Qinyang Lu, Saloni Gupta, and Adriano Koshiyama. 2025c. Bias amplification: Large language models as increasingly biased media. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pages 115–132.
- Xiukun Wei and Xueru Zhang. 2025. Self-consuming generative models with adversarially curated data. *arXiv preprint arXiv:2505.09768*.
- Yueqi Xie, Minghong Fang, Renjie Pi, and Neil Gong. 2024. Gradsafe: Detecting jailbreak prompts for llms via safety-critical gradient analysis. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 507–518.
- An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 23 others. 2025. *Qwen2.5 technical report*. Preprint, arXiv:2412.15115.
- Fanghua Ye, Mingming Yang, Jianhui Pang, Longyue Wang, Derek F Wong, Emine Yilmaz, Shuming Shi, and Zhaopeng Tu. 2024. Benchmarking llms via uncertainty quantification. *Advances in Neural Information Processing Systems*, 37:15356–15385.
- Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.
- Yiming Zhang, Javier Rando, Ivan Evtimov, Jianfeng Chi, Eric Michael Smith, Nicholas Carlini, Florian Tramèr, and Daphne Ippolito. 2025. Persistent pre-training poisoning of llms. In *International Conference on Learning Representations*, volume 2025, pages 31323–31340.
- Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. 2024. Wildchat: 1m chatgpt interaction logs in the wild. *arXiv preprint arXiv:2405.01470*.
- Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623.
- Xuekai Zhu, Daixuan Cheng, Hengli Li, Kaiyan Zhang, Ermo Hua, Xingtai Lv, Ning Ding, Zhouhan Lin, Zilong Zheng, and Bowen Zhou. 2024. How to synthesize text data without model collapse? *arXiv preprint arXiv:2412.14689*.
- Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J Zico Kolter, and Matt Fredrikson. 2023. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*.