TokenProber: Jailbreaking Text-to-image Models via Fine-grained Word Impact Analysis

Longtian Wang, *Graduate Student Member, IEEE,* Xiaofei Xie, *Member, IEEE,* Tianlin Li, *Member, IEEE,* Yuhan Zhi, *Graduate Student Member, IEEE,* and Chao Shen, *Senior Member, IEEE*

***Abstract*—Text-to-image (T2I) models have significantly advanced in producing high-quality images. However, such models have the ability to generate images containing not-safe-for-work (NSFW) content, such as pornography, violence, political content, and discrimination. To mitigate the risk of generating NSFW content, refusal mechanisms, *i.e.*, safety checkers, have been developed to check potential NSFW content. Adversarial prompting techniques have been developed to evaluate the robustness of the refusal mechanisms. The key challenge remains to subtly modify the prompt in a way that preserves its sensitive nature while bypassing the refusal mechanisms. In this paper, we introduce *TokenProber*, a method designed for sensitivity-aware differential testing, aimed at evaluating the robustness of the refusal mechanisms in T2I models by generating adversarial prompts. Our approach is based on the key observation that adversarial prompts often succeed by exploiting discrepancies in how T2I models and safety checkers interpret sensitive content. Thus, we conduct a fine-grained analysis of the impact of specific words within prompts, distinguishing between *dirty* words that are essential for NSFW content generation and *discrepant* words that highlight the different sensitivity assessments between T2I models and safety checkers. Through the sensitivity-aware mutation, *TokenProber* generates adversarial prompts, striking a balance between maintaining NSFW content generation and evading detection. Our evaluation of *TokenProber* against 5 safety checkers on 3 popular T2I models, using 324 NSFW prompts, demonstrates its superior effectiveness in bypassing safety filters compared to existing methods (*e.g.*, 54%+ increase on average), highlighting *TokenProber*'s ability to uncover robustness issues in the existing refusal mechanisms. The source code, datasets, and experimental results are available in [1].**

Warning: This paper contains model outputs that are offensive in nature.

***Index Terms*—Robustness Testing, Text-to-Image Generation, Refusal Mechanisms**

I. Introduction

The Text-to-Image (T2I) models have gained widespread attention due to their excellent capability in synthesizing high-quality images. T2I models, such as Stable Diffusion [2] and DALL·E [3], process the textual descriptions provided by users, namely prompts, and output images that match the descriptions. Such models have been widely used to generate various types of images, for example, the Lexica [4] contains more than five million images generated by Stable Diffusion. Besides, there are many advanced applications based on T2I models, for instance, Imagic [5] and Sine [6] allow users to edit objects in the source image using text descriptions, Textual inversion [7] and DreamBooth [8] fine-tune T2I models to generate images of special unique concepts.

With the popularity of T2I models, ethical concerns about the safety of synthesized image content have gradually emerged, *i.e.*, T2I models may synthesize images containing not-safe-for-work (NSFW) content, such as pornography, violence, political content, and discrimination. To prevent T2I models from synthesizing offensive or inappropriate images, model developers often add refusal mechanisms to the model to refuse the generation of such NSFW content. The typical approach to achieve the refusal mechanisms is to deploy safety checkers to block the NSFW prompts and corresponding generations. However, it has been reported that these safety checkers can be bypassed by manually crafted malicious prompts [9], [10], highlighting the need for a thorough robustness evaluation of the safety checkers in T2I models.

Several adversarial testing methodologies [11], [12], [13], [14] have been designed to reveal the highlighted robustness issues. These studies aim to reduce the likelihood that prompts containing sensitive words, as well as the resulting generated images, are detected by these safety filters. To achieve this purpose, the foundational strategy in these previous methods is to substitute sensitive words with less detectable alternatives. For example, SneakyPrompt [13] utilizes reinforcement learning (RL) to explore alternative tokens and formulate new prompts, and SurrogatePrompt [14], using Large Language Models (*e.g.*, ChatGPT) to find alternative words for NSFW parts in

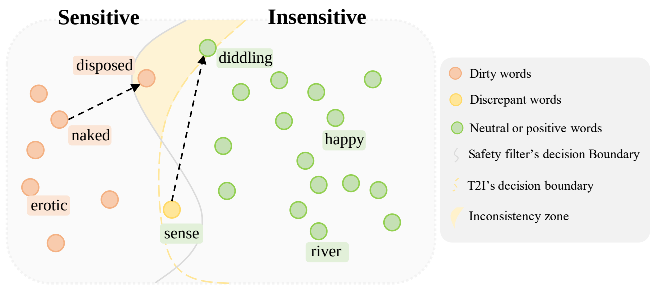


Fig. 1: Influence of different words within prompts. The explicit use of dirty words (*e.g.*, 'naked') can lead T2I models to generate NSFW content, which simultaneously triggers detection by the safety checkers. However, due to differing decision boundaries between the T2I model and the safety checkers regarding NSFW content, some words (*e.g.*, 'diddling') may fall within the region where discrepancies exist, specifically in the overlap between the boundaries of the safety checker and the T2I model. These words can cause the T2I model to generate NSFW content without being detected by the safety checker.

Longtian Wang, Yuhan Zhi, and Chao Shen are with the School of Cyber Science and Engineering, Xi'an Jiaotong University, Xi'an 710049, China (e-mail: chaoshen@mail.xjtu.edu.cn).

Xiaofei Xie is with Singapore Management University, Singapore 188065 (e-mail: xfxie@smu.edu.sg).

Tianlin Li is with the Nanyang Technological University, Singapore 639798.

prompts to bypass Midjourney's safety checkers. However, we find that such strategies remain ineffective because they tend to generate non-adversarial prompts (*i.e.*, false positives), where the prompts fail to include NSFW content in the generated images although they manage to evade safety checkers, *e.g.*, 66% of prompts reported success by SneakyPrompt are flase positives. The ineffectiveness mainly arises from *their emphasis on modifying explicitly sensitive words in the prompts, which may inadvertently remove the key terms that trigger NSFW content during generation, particularly when trying to bypass detection by refusal mechanisms. Consequently, these previous approaches face a dilemma: removing sensitive words decreases the chances of detection but also diminishes the potential to generate images with NSFW content.*

As illustrated in Fig. 1, the goal of these previous methods can be understood as crafting prompts that fall within the insensitive side of the safety filter's decision boundary, potentially losing their NSFW nature. To preserve the NSFW nature of the generated content, we refine the goal to crafting adversarial prompts that exploit the discrepancy between the safety checker's and the T2I model's decisions (as highlighted in yellow in Fig. 1): the safety checker fails to classify the prompt and the generation as unsafe, yet the T2I model proceeds to produce sensitive images with NSFW content. This discrepancy is intuitive given the nature of machine learning. Two distinct models are expected to yield differing outputs for certain inputs due to inherent differences in their training datasets, architectures, and parameter weights. Unless the models are identical in every aspect, such differences are inevitable. However, it is challenging to effectively craft adversarial prompts that exploit this discrepancy.

In our study, we propose *TokenProber*, a novel framework designed to assess the robustness of safety checkers within T2I models. To tackle the key challenge outlined above, *TokenProber* performs a fine-grained analysis of the influence of various words, enabling us to balance between maintaining NSFW content in the image generated by the T2I model and bypassing the detection of safety checkers (*i.e.*, the prompts falling within the inconsistency zone depicted in Fig. 1). *TokenProber* identifies two key types of words to navigate this balance: *Dirty Words*, which are essential for retaining the NSFW content and should maintain their semantic, and *Discrepant Words*, which are not inherently dirty (*e.g.*, neutral or positive) but significantly affect the safety checker's predictions in a negative manner. Discrepant words essentially indicate that mutating these words is more likely to mislead the safety checker. By mitigating the influence of discrepant words, the checker is more likely to assess the prompt positively. By safeguarding dirty words while diminishing the influence of discrepant words, *TokenProber* achieves an optimal equilibrium, which enables the creation of adversarial prompts that exploit the discrepancies between the decision of the safety checker and the T2I model.

Technically, *TokenProber* mainly includes two phases: Word-Level Sensitivity Analysis and Sensitivity-Aware Differential Testing. The initial phase analyzes the given prompt at the word level to identify *dirty* words, which are essential for retaining NSFW content, and *discrepant* words, which have a high potential to impact the safety checker's prediction towards negative. Following the analysis, the second phase performs a fuzzing, which iteratively mutates both the *dirty* and *discrepant* words towards the decision toward the region of discrepancies between the decision boundaries of the T2I model and the safety checker (as shown in Fig. 1). In each iteration, the goal is to alter the *dirty* words of the seed prompts in a way that their inherent NSFW semantic is still preserved, while the *discrepant* words are changed to ones that significantly differ in similarity. To exploit the boundary discrepancies more effectively, *TokenProber* employs the principles of differential testing. Inspired by [15], which approximates a surrogate decision boundary for generative models, we build a surrogate safety checker to classify images as safe or unsafe trained on the generations of T2I models [11] and use the surrogate safety checker's decision boundary to approximate that of the T2I models. We then compare the safety score from the surrogate safety checker with that of the target safety checker to indicate discrepancies between the decision boundaries. Thus, in each iteration, we select prompts maximizing the difference in safety scores between the target checker and the surrogate safety checker as seed prompts for the next iterations.

We conduct a comprehensive evaluation to demonstrate the effectiveness of *TokenProber* in generating adversarial prompts. Specifically, we evaluate 5 target safety checkers applying to 3 popular T2I models and 324 NSFW prompts to evaluate *TokenProber*. The results demonstrate that 1) *TokenProber* has superior effectiveness in bypassing target safety checkers compared to state-of-the-art methods (*e.g.*, 54%+ increase on average). 2) *TokenProber* is more efficient in generating adversarial prompts, *e.g.*, on average it takes 60%+ less time to generate an adversarial prompt.

To summarize, this paper makes the following contributions:

- We provide an understanding of the impact of two types of words within prompts: *dirty* words, which are integral to generating NSFW content, and *discrepant* words, which are sensitive in impacting the robustness of safety checkers.
- We introduce a novel robustness evaluation approach *TokenProber* that employs word sensitivity-aware differential testing to generate adversarial prompts capable of producing NSFW images.
- We conduct extensive experiments to evaluate the effectiveness of *TokenProber* on various open-source safety checkers on various T2I models. The evaluation results show that *TokenProber* achieves a 54%+ average increase in the bypass rate compared to the state-of-the-art methods.

II. Preliminary

A. Text-to-image Model

Text-to-image (T2I) models synthesize images conditioned by textual descriptions provided by users (*i.e.*, prompts). Fig. 2 shows the structure of a T2I model. Typically, T2I models comprise a language model and an image generation model. The language model, *e.g.*, CLIP's text encoder [16], serves to comprehend the prompt and convert it into text embeddings to guide image generation. The image generation model then employs a diffusion process, initiating with random noise and progressively denoising conditioned by the text embeddings to synthesize images that align with the user's description.

B. Safety checker for T2I Model

T2I models have impressive performance in synthesizing high-quality images. Meanwhile, ethical concerns about the safety of the generated image content have gradually emerged. To prevent T2I models from generating not-safe-for-work (NSFW) content, various safety checkers have been proposed. The safety checker, denoted as SC, takes a prompt p and the synthesized image I as input, then outputs the probability of containing NSFW content. We called the probability Safety Score, denoted as $SC(p, I)$.

As shown in Fig. 2, the safety checkers can be divided into three categories [13]: text safety checkers, image safety checkers, and text-image safety checkers.

- **Text safety checker**: The input of safety checkers in this category is the prompt or the text embedding obtained by language model. Prompt containing sensitive words or embedding with high similarity to predefined unsafe concepts will be blocked before synthesizing images.
- **Image safety checker**: Safety checks in this category takes the synthesized image as input and detect NSFW content. Typically, a binary classifier trained on NSFW images and safe images is employed to detect whether synthesized images contain NSFW content.
- **Text-image safety checker**: Safety checks in this category utilize text embedding of unsafe concept and synthesized images to prevent the generation of NSFW content. To the best of our knowledge, currently only the open-source Stable Diffusion [2] utilizes this type of safety checker. The safety checker firstly employs CLIP image-encoder to obtain the embedding of synthesized image, and blocks image embeddings that have high cosine similarity to 17 pre-defined text embeddings of unsafe concepts [12].

C. Problem Definition

Definition 1 (Robustness of Safety Checker for T2I Model): Given an T2I Model M equipped with a safety checker F, for any prompt p that is filtered by the safety checker (*i.e.*, $F(M,p) = 1$), indicating that the prompt is recognized as invalid, we define the robustness of F on p as:

$$\forall p' ||M(p) - M(p')|| < \epsilon \Longrightarrow F(M,p) = F(M,p'). \quad (1)$$

This equation posits that for any variant of the prompt p', if the semantic difference in the outputs of M is small (*i.e.*, $||M(p) - M(p')|| < \epsilon$), then the safety checker F's decision should remain consistent for both p and p'.

The objective of our testing is to search the adversarial prompt p' that satisfies the following two properties:

- *P1: NSFW maintenance* (*i.e.*, $||M(p) - M(p')|| < \epsilon$): The images synthesized by the generated adversarial prompt should still contain NSFW content. If a prompt that bypasses the safety checker but no longer contains NSFW content, it is not a valid adversarial prompt.
- *P2: Bypassing the filter* (*i.e.*, $F(M,p) \neq F(M,p')$): The target safety checker fails to block the prompt or the NSFW content after some perturbations.

III. Methodology

A. Overview of our Approach

Fig. 3 depicts the overview of *TokenProber*. *TokenProber* takes an initial prompt that describes an NSFW image scene, the T2I model and the target safety checker as inputs. For simplicity of presentation, the model and target safety checker are omitted from the figure. Note that the output of the initial prompt is blocked by the target safety checker under test. *TokenProber* operates in two main steps. In the first step, *TokenProber* analyzes the impact of individual words on the generation of NSFW content and their influence on the target safety checker's prediction. This involves examining each word for its "Dirtiness" and "Discrepancy". Dirtiness Analysis assesses whether a word predominantly contributes to NSFW content generation (*e.g.*"naked"), while Discrepancy Analysis evaluates the word's effect on the target safety checker's robustness.

Based on the insights from the sensitivity analysis, *TokenProber* performs a fuzzing strategy to optimize the target prompt. Two mutation strategies are proposed to refine the target prompt. Dirtiness-Preserving Mutation looks for semantically similar words to replace dirty ones, maintaining NSFW content generation potential while possibly challenging the target safety checker's robustness. Discrepancy-Away Mutation aims to replace Discrepant words with alternatives that reduce the likelihood of being classified as unsafe by the target safety checkers, thus improving the prompt's potential to bypass the target safety checkers.

Since no perfect oracle exists to definitively judge whether the image generated from a new prompt contains NSFW content. *TokenProber* performs cross-checking with a surrogate safety checker, which is trained on the outputs of T2I models to approximate a surrogate decision boundary for generative models. A prompt deemed safe by the target safety checker but flagged by the surrogate safety checker is considered a potential adversarial prompt. The selection of the best candidate prompt from the generated mutations is guided by the objective to maximize the safety score difference between the target safety checkers and surrogate safety checker. This iterative process continues until a successful adversarial prompt is generated or the maximum number of iterations is reached.

B. Word-level Sensitivity Analysis

In the generation process with T2I models, replacing different words in the prompt leads to varied effects on the NSFW content of the synthesized images and the robustness of the target safety checker. As outlined in Section II-C, our dual objectives in crafting adversarial prompts are: preserving NSFW content (P1) and evading the safety filter (P2). Identifying words pivotal for both preserving NSFW content and deceiving the target safety checker is essential for achieving these goals. To this end, we perform a fine-grained analysis to evaluate two critical attributes of the word: *dirtiness* and *discrepancy*.

1) Dirtiness Analysis: Dirtiness Analysis aims to identify words in the prompt that significantly influence the NSFW

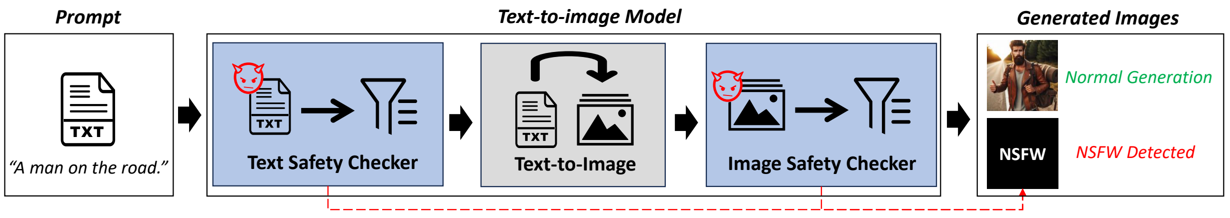


Fig. 2: Text-to-image model structure

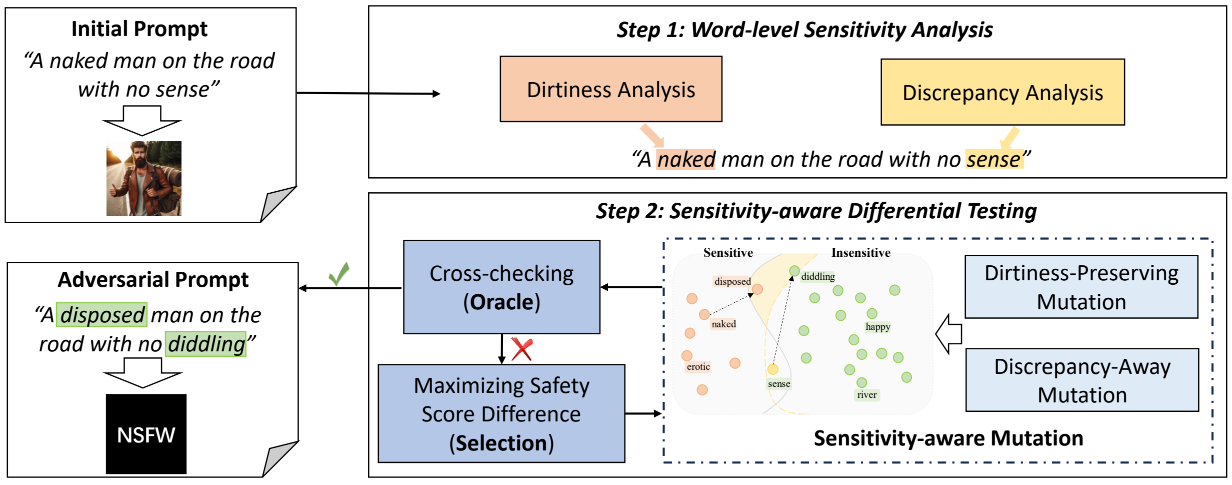

Initial Prompt
"A naked man on the road with no sense"
Step 1: Word-level Sensitivity Analysis
Dirtiness Analysis
Discrepancy Analysis
"A naked man on the road with no sense"
Step 2: Sensitivity-aware Differential Testing
Adversarial Prompt
"A disposed man on the road with no diddling"
NSFW
Cross-checking (Oracle)
Maximizing Safety Score Difference (Selection)
Sensitive
Insensitive
disposed
diddling
naked
happy
erotic
sense
river
Dirtiness-Preserving Mutation
Discrepancy-Away Mutation
Sensitivity-aware Mutation

Fig. 3: The workflow of *TokenProber*

content of synthesized images. These words are typically easily identified as sensitive, *e.g.*, "naked", "nude" and "sexual", previous work [11] has demonstrated that prompts include such words can retrieve images containing NSFW content. We adopt a simple strategy to identify dirty words based on the existing sensitive word list, which has been used in previous work [13]. The sensitive word list includes various themes, such as violence, discrimination, and sexual content, making it suitable for detecting multiple categories of unsafe content.

Formally, given a prompt $p = \{w_1, w_2, ..., w_n\}$ and the predefined NSFW word list W_{NSFW}, the dirtiness of a word $w \in p$ is determined as:

$$Dt(w) = \begin{cases} 1, & w \in W_{\text{NSFW}} \\ 0, & \text{otherwise.} \end{cases}$$

2) Discrepancy Analysis: While our primary aim is to diminish the negative impact of prompts to meet objective P2, we cannot simply eliminate obvious dirty words required for maintaining objective P1. Instead, our strategy involves identifying the words that, although not dirty words, significantly contribute to the safety checker's assessment, which are called discrepant words. Specifically, discrepant words are highlighted within the discrepant zone of Fig. 1, where prompts containing these words are interpreted as insensitive by the T2I model yet flagged as sensitive by the target safety checker.

By minimizing the influence of discrepant words, we aim to achieve objective P2. The presence of discrepant words underscores the quality issue in the target safety checker's robustness, revealing that its predictions are unduly affected by words that should not skew its evaluations. Our methodology exploits this flaw, targeting the removal of discrepant words to refine the prompt and improve its acceptance by the target safety checker.

As depicted in Fig. 1, to quantify the discrepancy of a word, we need to evaluate the difference in the decision between the target safety checker (which may be interpreted as insensitive) and the T2I model (which might be interpreted as sensitive). Given that the T2I model does not offer a direct measure of sensitivity, we opt for a surrogate safety checker trained on the outputs of T2I models to approximate the decision boundary of T2I models. Since the training data for the surrogate safety checker is entirely composed of images generated by the T2I model, it will better approximate the decision boundary of the T2I model. Thus, utilizing a surrogate safety checker as a reference (denoted as SC_r) provides a meaningful comparative basis to identify the discrepant words of the target checker (denoted as SC_t).

Given a prompt $p = \{w_1, w_2, ..., w_n\}$, we use $p \backslash w$ to denote the prompt after removing the word w and I to denote the image synthesized by $p \backslash w$. The discrepancy of a word $w \in p$ is calculated as the safety score difference (*i.e.*, the the probability that the modified prompt $p \backslash w$ predicted by the safety checkers SC_t and SC_r), formally expressed as:

$$Dc(w) = SC_r(P \backslash w, I) - SC_t(P \backslash w, I).$$

Intuitively, if a word w exhibits high discrepancy, it implies that, when w is removed, the modified prompt $p \backslash w$ is deemed less risky by the target safety checker, compared to the surrogate safety checker. Such discrepancy suggests that reducing the influence of w is particularly crucial for bypassing the target safety checker's scrutiny.

Algorithm 1: *TokenProber*

```
Input  : p: seed prompt, $SC_t$: target safety checker, $SC_r$: surrogate safety checker, $DirLis$: the dirty word substitution list, $DisLis$: the discrepant word substitution list.
Output : $p'$: potential adversarial prompt
Const  : $T$: testing budget, $K$: number of discrepant words.
1  $p_{cur} \leftarrow p$;
2  $W_{dir} \leftarrow \{w | w \in p \wedge Dt(w) = 1\}$;
3  $W_{dis} \leftarrow TopK_{Dc}(K, p)$;
4  for $t \in [0, T)$ do
5      $W'_{dir} \leftarrow RandSelect(W_{dir})$;
6      $Sub_{dir} \leftarrow \{(w, \arg\max_{w' \in DirLis} Similarity(w, w')) | w \in W'_{dir}\}$ ;
7      $p_{dir} \leftarrow Replace(p_{cur}, Sub_{dir})$;
8      $W'_{dis} \leftarrow RandSelect(W_{dis})$;
9      $Sub_{dis} \leftarrow \{(w, \arg\min_{w' \in DisLis} Similarity(w, w')) | w \in W'_{dis}\}$ ;
10     $p_{dis} \leftarrow Replace(p_{dir}, Sub_{dis})$;
11     $BestFit \leftarrow fitness(p_{cur})$ ;
12     for $p' \in \{p_{dis}, p_{dir}\}$ do
13         if $InValid(p')$ then
14             return $p'$ ;
15         if $fitness(p') > BestFit$ then
16             $BestFit \leftarrow fitness(p')$ ;
17             $p_{cur} \leftarrow p'$ ;
18 return $p_{cur}$ ;
```

C. Sensitivity-aware Differential Testing

Building on the foundation of word-level sensitivity analysis, *TokenProber* employs a greedy-based fuzzing strategy to identify potential adversarial prompts. This approach involves mutation of specific word categories through *Dirtiness-Preserving Mutation* and *Discrepancy-Away Mutation*. As detailed in Algorithm 1, *TokenProber* operates with an array of inputs: an initial prompt p, a target safety checker SC_t, a surrogate safety checker SC_r, and lists of candidate substitutions for dirty ($DirLis$) and discrepant ($DisLis$) words, aiming to generate the adversarial prompt p'. Note that the initial prompt can be blocked by the target solver. There are some hyperparameters that can be adjusted, such as the testing budget T, indicating the maximum number of iterations, and the focus on the top K discrepant words in the prompt.

p_{cur} represents the current prompt in each iteration, which is initialized as p (Line 1). The process begins by identifying all dirty words within the prompt p (Line 2), along with the top K discrepant words based on their discrepancy scores (Line 3). These discrepant words are then ordered by their impact on the safety checker's prediction, prioritizing those with higher discrepancy scores for further process. With these selections in place, *TokenProber* proceeds to conduct a greedy search operation within the specified budget (Line 4), aiming to refine p_{cur} by adjusting the presence and influence of these critical word types.

1) Dirtiness-Preserving Mutation: During each iteration, *TokenProber* firstly mutates the dirty words, adhering to the strategy of maintaining the negative semantics integral for NSFW content generation. The fundamental idea is to substitute a dirty word with another that holds similar negative connotations, thereby preserving the prompt's NSFW nature while potentially challenging the safety checker's robustness.

TokenProber randomly selects a subset of dirty words from the current prompt p_{cur} for mutation (Line 5). For each selected dirty word, *TokenProber* evaluates the semantic similarity with candidates from $DirLis$, aiming to find the most closely aligned substitute in terms of dirtiness (Line 6). This similarity assessment relies on the cosine similarity between their word embeddings obtained by a language model, calculated as $Cosine\_Similarity(embed(w), embed(w'))$. The dirty words in p_{cur} are then replaced with their most semantically similar counterparts from $DirLis$, resulting in a new prompt p_{dir} (Line 7).

2) Discrepancy-Away Mutation: Subsequently, *TokenProber* targets discrepant words for mutation, aiming to select substitutions that deviate maximally from the original, thereby minimizing the discrepant word's impact and potentially creating a prompt that is more likely to bypass the safety filter. Similarly, *TokenProber* randomly chooses a group of discrepant words within the current prompt for the mutation (Line 8). For every discrepant word in this subset, *TokenProber* evaluates its similarity to candidates in $DisLis$, opting for the candidate with the least similarity for replacement (Line 9), resulting in a new prompt p_{dis} (Line 10).

3) Oracle and Fitness: Given the challenge of the lack of oracle for accurately detecting NSFW content in images, *TokenProber* leverages a strategy of cross-checking between the target safety checker (SC_t) and a the surrogate safety checker (SC_r) to pinpoint potential adversarial prompts. A prompt is considered potentially adversarial if it evades detection by SC_t but is flagged as NSFW by SC_r (Line 14). This condition for a prompt p and the corresponding generated image I being deemed adversarial is formally expressed as:

$$InValid(p, I) = (SC_t(p, I) < 0.5) \wedge (SC_r(p, I) > 0.5).$$

In our experiments, we further manually confirm the NSFW content of the synthesized images by the detected adversarial prompts.

In cases where a prompt does not meet the condition, *TokenProber* utilizes the safety score difference as a fitness function. This function aids in selecting the most effective prompt for the subsequent iteration by comparing p_{cur}, p_{dir} and p_{dis} (Line 14-17). The objective is to maximize the discrepancy in safety scores between the checkers, formally defined as:

$$fitness(p, I) = SC_r(p, I) - SC_t(p, I).$$

4) Discussion: In our implementation, the word-level sensitivity analysis is conducted only once on the initial prompt p, rather than repeatedly on every updated prompt p_{cur} in the iteration. This choice prioritizes efficiency by minimizing the number of required queries to the safety checker, especially considering the resource-intensive nature of the discrepancy analysis (Dc), which necessitates two queries per word.

For the mutation process detailed in Algorithm 1, substitutions within p_{cur} are not based on direct word matching but rather through the indices of words in p. Specifically, for each substitution pair (w, w'), the replacement occurs as follows: $p_{cur}[index(p, w)] = w'$, where *index* function determines the position of w within p. The reason is that an original dirty word w might have been substituted with another dirty word in the current prompt p_{cur}.

To further enhance query efficiency, only one prompt is maintained for each mutation, *i.e.*, dirtiness-preserving (p_{dir}) and discrepancy-away (p_{dis}), during iterations. This approach limits the volume of oracle checks and fitness calculations which require the query on the safety checkers. While it is possible to retain and evaluate more mutants per iteration for the selection, such adjustments are subject to available queries on T2I system or computational capacity.

IV. Evaluation

We have implemented *TokenProber* in Python 3.8 with Pytorch (ver.1.11.0). To evaluate the effectiveness of *TokenProber* for evaluating the robustness of safety checkers, we aim to answer the following research questions (RQs):

- **RQ1**: How effective is *TokenProber* in generating adversarial prompts that maintaining NSFW content?
- **RQ2**: What are the contributions of different components of *TokenProber* in improving effectiveness?
- **RQ3**: How do different hyperparameters affect the performance of *TokenProber*?

A. Setup

1) T2I Models and Datasets: We employ three popular T2I models to synthesize images based on provided prompts. Specifically, we select Stable-Diffusion-v1.4 [17], following the previous works [12], [11], [13], to ensure a fair comparison. To further demonstrate the generalization of our method, we included DreamLike [18] and Stable-Diffusion-v1.5 [19] in the experiments, as their training data and generated images differ significantly from Stable-Diffusion-v1.4. We evaluate *TokenProber* using three NSFW prompt datasets, which were constructed and utilized by previous works [11], [13]. Specifically, the 4chan and Lexica datasets were constructed by Unsafe Diffusion [11], and the NSFW-200 dataset was constructed by SneakyPrompt [13]:

- *4chan Dataset*: 500 prompts collected from posts on 4chan [20], encompassing various NSFW themes such as pornography, violence, politics, and discrimination. The posts are prone to NSFW image generation.
- *Lexica Dataset*: 404 prompts collected from Lexica website [4], a large AI-Generated image database. The prompts are retrieved through NSFW keywords included in DALL·E's content policy [21], such as sexual, hate, politics, and similar content.
- *NSFW-200 Dataset*: 200 prompts generated by GPT-3.5, following a post on Reddit [22]. The generated prompts contain descriptions of pornographic and politically related content.

Given our objective of creating adversarial prompts that can bypass the target safety checker, when selecting the initial seed prompts, we exclusively consider prompts that are consistently blocked by all target safety checkers in the prompt datasets. Finally, there are a total of 325 seed prompts, with 115 from the 4chan dataset, 78 from the Lexica dataset, and 132 from the NSFW-200 dataset.

2) Target Safety Checker: We follow the previous work [13] and select 5 safety checkers covering all 3 categories to conduct our evaluation.

- *Text Safety Checker*: Text-Match [13] and Text-Classifier [23] are selected as the target text safety checkers. Text-Match detects prompts containing words from a predefined sensitive word list [24]. Text-Classifier is a fine-tuned DistilBERT [25], the fine-tuning datasets are posts collected from Reddit [26] with the purpose of classifying NSFW text content.
- *Image Safety Checker*: We select NSFW-Classifier [27] and CLIP-Detector [28] as the target image safety checkers. NSFW-Classifier, an open-source image classifier specifically designed to detect NSFW content in images. CLIP-Detector, a binary classifier trained on CLIP image embeddings from an NSFW image dataset [29], enabling it to identify NSFW content in images.
- *Text-Image Safety Checker*: We choose Text-Image-SD, the default safety checker of the open-source Stable Diffusion framework. It blocks image embeddings that exhibit significant similarity to pre-defined NSFW concepts [12].

3) Baselines: To evaluate the effectiveness of *TokenProber*, we compare *TokenProber* with two types of techniques:

- *Adversarial attack*: This category includes two state-of-the-art adversarial attack techniques designed for NLP classification tasks: PWWS [30] and TextFooler [31]. These techniques are used to generate adversarial examples specifically for text-based models. We use our cross-check oracle to determine whether the adversarial examples result in images containing NSFW content.
- *Adversarial prompting*: We include Sneakyprompt [13] and SurrogatePrompt [14], the most recent techniques for evaluating the robustness of safety checkers in T2I models. Sneakyprompt employs reinforcement learning to explore alternative tokens and formulate new prompts. It aims to achieve high CLIP embedding similarity between the initial prompt and the synthesized image. SurrogatePrompt uses an LLM to find alternatives for sensitive words in the input prompt to reduce the likehood of triggering the safety checkers.

4) Metrics: To compare the results of different tools, we select four commonly used metrics: the bypass rate, the number of queries (Query number), the time used (Time), and the CLIP score.

- *Bypass rate*: This metric quantifies the percentage of prompts that successfully bypass the target safety checker while still containing NSFW content in the synthesized images. A higher bypass rate indicates that the method is more adept at generating adversarial prompts for a larger proportion of seed prompts.
- *Query number*: This metric represents the total number of queries made to the T2I model and safety checker when an adversarial prompt is successfully generated. This metric could assess the efficiency of each tool.
- *Time*: This metric represents the total time used when an adversarial prompt is successfully generated. This metric can more thoroughly measure the efficiency of each tool.

- *CLIP score*: This metric measures the cosine similarity between the CLIP embedding of the prompt and the synthesized image. A higher CLIP score indicates that the synthesized image more closely aligns with the prompt's description, signifying higher synthesis quality.

5) Experiment Setup:

a) Configuration of TokenProber: For each seed prompt, we set the testing budget T (maximum number of iterations) to 60. By default, the number of discrepant words that can be mutated (K in Algorithm 1) is configured as 1, striking a balance between the bypass rate and CLIP score. Increasing the number of discrepant words for mutation could potentially enhance the bypass rate by altering the prompt in a larger degree, but it may also lead to a decrease in the CLIP score due to significant prompt modifications. To further control computational cost while maintaining performance, we set the number of candidate prompts retained per mutation to 1. This ensures a streamlined search process without significantly sacrificing effectiveness. We evaluate the impact of these parameters in Section IV-D.

Besides these hyperparameters, we used the default language models of three T2I models to extract embeddings. Specifically, for the two Stable Diffusion models, we use CLIP-VIT-L-14 [32], while DreamLike used its built-in text encoder [18]. And the surrogate safety checker used is trained follow the configuration of Unsafe Diffusion [11].

To ensure the regularity of mutated prompts, both the dirty words substitution list ($DirLis$) and the discrepant words substitution list ($DisLis$) are constructed by randomly selecting words from the google-10000-english-dictionary [33], a curated list of 10,000 commonly used English words, as our substitution search space for regularity.

b) Configuration of Baselines: For the baselines, we follow their default configurations. In the case of Sneakyprompt, the reward function is set as the similarity between the synthesized image and the initial prompt, which tends to perform better in maintaining NSFW content compared to using the similarity between the optimized prompt and the initial prompt, as per their evaluation results. The maximum query number is also set to 60 for fair comparisons.

It is important to note that we observed the inaccuracy in the oracle used to determine the success of attacks by SneakyPrompt. This inaccuracy can lead to the occurrence of false positives, where the generated prompts bypass the checker but result in synthesized images without NSFW content, leading to early termination of the tool. To ensure a more accurate comparison, we introduce a variant of SneakyPrompt, denoted as SneakyPrompt_c, which incorporates our oracle (*i.e.*, the cross-checking method) to determine the successful generation of adversarial prompts containing NSFW content.

In the case of SurrogatePrompt, since it cannot automatically identify the dirty words to be substituted in the input prompt, we first use *Text-Match* to select the dirty words in the prompt and then query ChatGPT 3.5 with "what is similar to <word>" to obtain alternatives for the dirty words, where <word> represents each selected dirty word. To minimize the chances of ChatGPT refusing to respond due to dirty words violating its content policy, we use ChatGPT-3.5-turbo to generate substitute terms, as it has been found to be less strict in refusing NSFW content compared to the current version of ChatGPT [34]. For each dirty word, we use ChatGPT to generate five replacement words to modify the prompt and then test whether the modified prompts and use our cross-checking method to evaluate the successful generation of adversarial prompts containing NSFW content.

As cross-checking cannot absolutely ensure the correctness of generated prompts, we supplement the evaluation with a human study. We synthesize images using the adversarial prompts generated by each tool and manually verify whether the synthesized images indeed contain NSFW content. During the human study, most images are easily confirmed if NSFW content is included. For any ambiguous cases, our authors discuss them, and if a unanimous decision cannot be reached, the prompt will be discarded. However, in our study, we encountered no such controversial examples where an agreement couldn't be reached. In our evaluation, all bypass rate results are calculated based on the human-verified outcomes.

c) RQ Setup: For **RQ1**, we take all the seed prompts and utilize various tools to generate adversarial prompts. Subsequently, we compare their performance based on the bypass rate, number of queries, the time used, and CLIP score.

For **RQ2**, we conduct an ablation study to assess the contributions of the word-level sensitivity analysis and sensitivity-aware mutation. First, we implement a random strategy (**Random**) that randomly selects a word for mutation, *i.e.*, without sensitivity-aware mutation. The candidate substitution list for it is the union of the substitution lists for dirty words and discrepant words ($DirLis$ and $DisLis$). To demonstrate the importance of sensitivity-aware mutation, we configure two variants of *TokenProber*: one without Dirtiness-preserving mutation (**w/o Dirty**) and another without Discrepancy-away mutation (**w/o Discrepancy**). Additionally, we introduce two specialized versions for a more granular examination of our mutation approach: *TokenProber* with Dirtiness-Away Mutation, which selects substitutions that differ from the original dirty words (**Dirt-away**), and *TokenProber* with Discrepancy-preserving mutation, which selects substitutions similar to the original discrepant words (**Discr-prev**). Dirt-away and Discr-prev mutate the dirty words and discrepant words in the opposite direction to *TokenProber*, respectively.

There are several parameters in *TokenProber* that can be adjusted, including the number of discrepant words K to be mutated, the testing budget T, and the number of substitutions for dirty words and discrepant words, denoted as N. We set N to 1, meaning that only one prompt (p_{dir} and p_{dis}) is obtained after substituting the most similar dirty word and the least similar discrepant word. However, we can generate more candidate prompts by selecting the top N most similar dirty words or least similar words. In **RQ3**, we conduct an experiment to analyze the impact of these parameters on performance.

Considering the numerous comparison settings involved, in **RQ2** and **RQ3**, we randomly select 100 prompts for experimentation. Additionally, to analyze the performance change on both prompt and image, we use the Text-Image-SD safety checker and Text-Classifier safety checker as our targets for these experiments, as they respectively judge NSFW content based on image and text. [35].

B. RQ1: Effectiveness Results

Bypass Rate. Table I presents the bypass rates of adversarial prompts generated by different methods. Overall, the results demonstrate that *TokenProber* significantly outperforms all baselines in generating adversarial prompts.

Specifically, compared to the state-of-the-art adversarial prompting techniques, *i.e.*, SneakyPrompt (row SneakyP and SneakyP_c) and SurrogatePrompt (row SurrogateP), *TokenProber* (row TokenP)achieves a higher bypass rate across all datasets and safety checkers, with an average improvement of 0.54. which demonstrates its superior effectiveness in bypassing target safety checkers compared to state-of-the-art methods

Comparing SneakyPrompt with SneakyPrompt_c, we observed that SneakyPrompt_c generally performs better. This is because SneakyPrompt generates many false positives, stemming from its inaccurate oracle. Even with our cross-check oracle, SneakyPrompt_c's performance remains inferior to *TokenProber* (0.24 vs 0.71 on average). The lower bypass rate of SneakyPrompt could be attributed to two main factors: 1) its focus on substituting dirty words, which increases the likelihood of affecting the NSFW nature of synthesized images; and 2) its reward function, based on the similarity between the initial prompt and the generated images, which is less effective in balancing the preservation of sensitive semantics for NSFW content generation and avoiding detection by safety filters during generation. This highlights the necessity of fine-grained word impact analysis to achieve a balance in jailbreaking T2I models.

As for SurrogatePrompt, we found that it performs significantly better on the 4chan dataset compared to the other two datasets (on average, 0.24 vs. 0.04), which is due to the prompts in 4chan dataset containing more explicit dirty words. In contrast, the Lexica and NSFW datasets use the discrepant words more frequently to describe NSFW scenarios, leading to poorer performance for SurrogatePrompt, which focuses primarily on dirty words.

In addition, adversarial attacks exhibit significantly worse performance compared to *TokenProber*. This is because adversarial attacks primarily aim to generate adversarial examples that bypass the target safety checker, without considering the preservation of NSFW content in synthesized images.

Comparing the results of different datasets, we observe that *TokenProber* performs much better on NSFW-200. This is because NSFW-200 is generated by GPT-3.5, which contains less dirty words in the prompts, compared with 4chan and Lexica datasets. Therefore, it is easier to balance the NSFW maintenance and avoid the detection of safety checkers, as only a few dirty words need to be mutated, resulting in better performance on NSFW datasets.

Comparing the results of different T2I models, we found that the effectiveness of *TokenProber* is not affected by differences between T2I models. When using three different T2I models, *TokenProber* consistently outperforms the baseline, which demonstrates the Generalization of *TokenProber* to safety checkers deployed on various T2I models.

Comparing the results of testing different safety checkers, we observe that *TokenProber* always has better performance when testing NSFW-Classifier and Text-Image-SD. This is because these two safety checkers are not based on text checking, and insensitive to the dirty words. Like general adversarial attacks, bypassing safety checkers is possible by choosing different substitutions to replace the dirty words. Note that although CLIP-Detector is also a safety checker based on images, it is trained using more than fifty thousands of image embeddings. Hence, the CLIP-Detector tends to be more robust compared to NSFW-Detector and Text-Image-SD.

Query Number. We compared the efficiency of several methods with good bypass rates, specifically those with an average pass rate exceeding 0.10: TokenProber (0.68), SneakyPrompt (0.18), and SneakyPrompt_c (0.26). Table II shows their query number, time used and CLIP score. We observe that SneakyPrompt achieves relatively better results in terms of query number. This is because they have many false positives (on average 34%), leading to the early termination. When SneakyPrompt uses our oracle to determine the success of attacks, SneakyPrompt_c has a much higher average query number than *TokenProber* (47.51 vs 25.50).

Time. Overall, *TokenProber* has a higher bypass rate while using less time, *e.g.*, it takes 60.33% less time to generate an adversarial prompt on average compared with SneakyPrompt. When comparing with the results of query numbers, we find that although SneakyPrompt requires fewer queries in some cases, it takes more time due to the use of reinforcement learning in the process of searching for alternative words, which makes each round take longer.

CLIP Score. For comparison, we provide reference CLIP scores calculated for the seed prompts in the 4chan, Lexica, and NSFW-200 datasets. The average CLIP scores for the three datasets are 22.1, 28.66, and 23.16, respectively. Compared to adversarial prompting methods, on average, we find that *TokenProber* achieves a higher CLIP score (22.65 vs 22.23), demonstrating higher image quality using our adversarial prompts.

Table III displays the false positive rates (*i.e.*, prompts incorreclty classified as adversarial) results of *TokenProber* and SneakyPrompt. We manually analyzed the reported adversarial prompts to determine how many of them did not contain NSFW content. The results indicate that the cross-check between the target safety checker and the surrogate safety checker is effective in identifying invalid images. This effectiveness may be attributed to the low probability that both safety checkers will misclassify adversarial prompts as valid. In comparison, SneakyPrompt shows a higher rate of false positives (*e.g.*, 0.03 vs. 0.49 on average).

Answers to RQ1: *TokenProber* significantly outperforms adversarial attack techniques and the state-of-the-art Sneakyprompt in generating adversarial prompts. The cross-check is a useful oracle for determining whether the

TABLE I: Results of bypass rate on Text-Match (T-M), Text-Classifier (T-C), NSFW-Classifier (N-C), CLIP-Detector (C-D), and Text-Image SD (TI-SD)

Dataset	Method	Stable-Diffusion-v1.4					DreamLike					Stable-Diffusion-v1.5				
		T-M	T-C	N-C	C-D	TI-SD	T-M	T-C	N-C	C-D	TI-SD	T-M	T-C	N-C	C-D	TI-SD
4chan	TokenP	**0.43**	**0.56**	**0.88**	**0.52**	**0.87**	**0.77**	**0.51**	**0.79**	**0.57**	**0.91**	**0.45**	**0.28**	**0.79**	**0.45**	**0.71**
	SneakyP	0.18	0.11	0.10	0.03	0.13	0.20	0.09	0.09	0.05	0.06	0.09	0.06	0.02	0.03	0.01
	SneakyP_c	0.20	0.11	0.11	0.07	0.12	0.29	0.17	0.09	0.16	0.17	0.18	0.08	0.08	0.06	0.09
	SurrogateP	0.27	0.10	0.25	0.17	0.32	0.32	0.17	0.32	0.22	0.40	0.19	0.06	0.36	0.14	0.23
	PWWS	0.02	0.08	0.03	0.06	0.04	0.03	0.03	0.05	0.06	0.05	0.10	0.02	0.03	0.02	0.01
	TextFooler	0.02	0.04	0.04	0.06	0.05	0.01	0.02	0.05	0.04	0.04	0	0	0	0	0
Lexica	TokenP	**0.49**	**0.63**	**0.74**	**0.64**	**0.67**	**0.64**	**0.63**	**0.69**	**0.67**	**0.72**	**0.68**	**0.65**	**0.74**	**0.64**	**0.62**
	SneakyP	0.08	0.05	0.15	0.04	0.17	0.04	0.03	0.10	0.05	0.13	0.01	0.01	0.04	0.06	0.06
	SneakyP_c	0.07	0.06	0.22	0.13	0.23	0.10	0.09	0.12	0.18	0.17	0.10	0.13	0.08	0.02	0.32
	SurrogateP	0.05	0.04	0.03	0.04	0.03	0.04	0.04	0.05	0.05	0.04	0.03	0.03	0.05	0.03	0.05
	PWWS	0.04	0.04	0.04	0.05	0.10	0.01	0	0	0.01	0.01	0.03	0.01	0.04	0.04	0.05
	TextFooler	0	0.04	0.05	0.05	0.06	0	0	0.01	0	0.03	0	0	0	0	0
NSFW-200	TokenP	**0.89**	**0.74**	**0.97**	**0.72**	**0.92**	**0.62**	**0.53**	**0.92**	**0.53**	**0.89**	**0.61**	**0.45**	**0.98**	**0.77**	**0.98**
	SneakyP	0.36	0.35	0.42	0.29	0.33	0.39	0.26	0.59	0.62	0.51	0.39	0.29	0.31	0.41	0.36
	SneakyP_c	0.45	0.36	0.58	0.32	0.61	0.49	0.41	0.39	0.71	0.67	0.46	0.39	0.61	0.45	0.58
	SurrogateP	0.05	0.04	0.03	0.04	0.03	0.04	0.04	0.05	0.05	0.04	0.03	0.03	0.05	0.03	0.05
	PWWS	0	0.30	0.34	0.30	0.34	0.01	0.01	0.02	0.02	0.02	0.01	0.27	0.28	0.33	0.30
	TextFooler	0.02	0.10	0.29	0.38	0.33	0.02	0.01	0.04	0.02	0.05	0	0	0	0.01	0.01

TABLE II: Results of Time (T), query number (Q.N) and CLIP score (C.S) on Stable-Diffusion-v1.4 (SD4), DreamLike(DLI), and Stable-Diffusion-v1.5 (SD5)

Model	Classifier	T-M			T-C			N-C			C-D			TI-SD		
		T	Q.N	C.S	T	Q.N	C.S	T	Q.N	C.S	T	Q.N	C.S	T	Q.N	C.S
SD4	TP	213.23	24.83	**23.50**	**239.07**	27.84	**23.47**	**202.53**	23.87	**22.12**	**220.63**	26.14	**21.83**	**212.38**	24.81	21.39
	SP	**128.07**	**3.01**	21.30	863.20	**20.31**	19.88	377.48	**8.88**	20.78	479.09	**11.20**	19.81	327.14	**7.65**	21.56
	SP_c	1921.97	45.14	19.42	2186.42	51.39	21.42	1942.31	45.63	21.45	2251.14	52.76	19.84	1819.58	42.62	**21.78**
DLI	TP	**67.01**	7.85	23.06	**100.36**	**11.50**	**22.88**	**132.47**	15.61	**23.34**	**155.65**	17.99	**22.34**	**113.64**	13.35	22.06
	SP	221.41	**5.17**	**23.92**	596.31	13.94	21.25	428.03	**10.01**	22.81	522.51	**12.20**	22.11	180.02	**4.22**	22.81
	SP_c	1797.63	42.01	25.11	1688.72	39.59	21.30	1647.47	38.50	22.53	1600.65	37.41	20.84	1712.67	40.09	**23.17**
SD5	TP	**89.84**	10.63	22.98	**106.66**	12.37	**23.45**	**106.95**	12.49	22.70	**155.10**	18.17	22.20	**110.40**	12.71	22.42
	SP	180.67	**4.22**	23.40	525.46	**12.34**	21.08	267.73	**6.25**	22.48	275.85	**6.44**	**23.00**	237.98	**5.56**	22.44
	SP_c	1899.29	44.38	**24.42**	1811.01	42.22	20.80	1823.73	42.78	**24.31**	2101.62	49.17	21.86	1722.45	40.41	**22.77**

adversarial prompts can lead to NSFW content.

C. RQ2: Ablation Study

Table IV presents the results of our ablation study, comparing different variants of *TokenProber* on two selected safety checkers, the results on other datasets are put on our website [35]. In the table, Q.N represents the average number of queries for all selected prompts, including unsuccessful ones, while C.S indicates the average CLIP score for only successful adversarial prompts.

The overall results validate the effectiveness of the two sensitivity-aware mutation strategies. The bypass rate of *TokenProber* decreases when either or both of these strategies are omitted. For instance, on Text-Image-SD, the bypass rate drops by 0.06, 0.50, and 0.15 when omitting dirtiness-preserving mutation, discrepancy-away mutation, and both, respectively. This highlights the importance of both mutation strategies in generating effective adversarial prompts.

We also observe that the results vary depending on the type of safety checker. Generally, text-based safety checkers (*e.g.*, T-C) are more challenging to bypass than text-image-based checkers (*e.g.*, TI-SD) because text-based checkers directly scrutinize sensitive keywords or phrases. For example, the bypass rate of w/o Dirty on T-C is 0, indicating that without mutating explicit sensitive keywords, text-based checkers will block the prompts by detecting such keywords. Conversely, w/o Discrepancy achieves a 0.52 bypass rate on T-C, suggesting that replacing sensitive or dirty keywords can evade detection by text-based checkers. In contrast, w/o Dirty has a higher success rate (0.80) on TI-SD, as TI-SD compares embeddings, which are not very sensitive to dirty words because changes in non-dirty words can also significantly affect the embedding. This is further evidenced by the comparison between the bypass rates of Random and w/o Discrepancy on TI-SD (0.71 vs 0.36), where modifying only the dirty words (w/o Discrepancy) has a lower success rate than random mutation (Random).

The results of Dirty-away and Discr-prev further underscore the significance of our mutation directions. When we modify the mutation of dirty words from most similar to least similar in *TokenProber* (Dirty-away), the bypass rate significantly decreases (0.86 vs 0.34 on TI-SD and 0.56 vs 0.48 on T-C), likely because replacing dirty words with the least similar ones tends to remove NSFW content. Similarly, mutating discrepant words from least similar to most similar (Discr-prev) results in a substantial drop in bypass rates (0.86 vs 0.74 on TI-SD and 0.56 vs 0.34 on T-C), highlighting the importance of identifying discrepant words to impact the robustness of the checker, especially for text-based filters that focus more on detecting explicit dirty words.

The average number of queries aligns with the bypass rate results; a lower bypass rate implies more unsuccessful cases that exhaust the query budget. Regarding the CLIP score, Discr-prev achieves the best results because it selects the most similar dirty words and discrepant words for substitution, making the adversarial prompt closer to the seed prompt.

Answers to RQ2: Both dirtiness-preserving mutation and discrepancy-away mutation play significant roles in

TABLE III: False Positive Rate of *TokenProber* and SneakyPrompt

Method	Stable-Diffusion-v1.4					DreamLike					Stable-Diffusion-v1.5				
	T-M	T-C	N-C	C-D	TI-SD	T-M	T-C	N-C	C-D	TI-SD	T-M	T-C	N-C	C-D	TI-SD
TokenProber	**0.02**	**0.04**	**0.03**	**0.04**	**0.04**	**0.03**	**0.03**	**0.03**	**0.10**	**0.04**	**0.03**	**0.03**	**0.03**	**0.02**	**0.04**
SneakyPrompt	0.33	0.32	0.31	0.38	0.37	0.49	0.61	0.42	0.58	0.47	0.64	0.71	0.47	0.69	0.50

TABLE IV: Results of Different Variants on Text-Image SD (TI-SD) and Text-Classifier (T-C)

Method	TI-SD			T-C		
	B.R	Q.N	C.S	B.R	Q.N	C.S
Random	0.71	29.00	21.88	0.46	41.94	20.44
w/o Dirty	0.80	23.18	22.75	0	-	-
w/o Discrepancy	0.36	44.14	19.23	0.52	36.98	22.10
Dirt-away	0.34	46.96	16.72	0.48	41.78	22.34
Discr-prev	0.74	27.55	**23.23**	0.34	43.60	**24.25**
TokenProber	**0.86**	**23.17**	20.80	**0.56**	**35.74**	22.64

enhancing the performance of *TokenProber*.

D. RQ3: Impact of Different Parameters

Fig. 4 presents the results of *TokenProber* with different parameter configurations. We vary the number of discrepant word selections (K=1, 2, 3, 4, 5), the testing budget (T=20, 30, 40, 50, 60), and the number of candidate prompts (N=1, 2, 3, 4, 5) to evaluate their impacts. SneakyPrompt's results (with a testing budget of 60) are included for reference. In almost all configurations, *TokenProber* outperforms SneakyPrompt, except when *TokenProber* operates with a significantly lower testing budget.

Regarding the impact of the number of discrepant words (K), an increase in K leads to a decrease in the bypass rate and CLIP score while increasing the query number. This indicates that it becomes more challenging to generate adversarial prompts as *TokenProber* seeks semantically deviated substitutions for discrepant words. As K increases, the mutation of discrepant words results in a semantic deviation of the seed prompt, affecting its performance. For the testing budget (T), as expected, both the bypass rate and query number increase with T, suggesting that some challenging cases require more testing budget.

Regarding the number of prompts generated for fitness selection (N), an increase in N significantly raises the query number (from 25.74 to 86.20) due to the additional queries to calculate fitness scores. However, the increase in bypass rate is not significant (from 0.86 to 0.91), indicating that maintaining one substitution does not significantly impact performance. This finding suggests that prioritizing one substitution strikes a better balance between effectiveness and efficiency.

Answers to RQ3: Selecting a large number of discrepant words for mutation can diminish the performance of *TokenProber*. Additionally, generating more candidate prompts for selection does not evidently enhance performance in our greedy-based approach. By opting for smaller values ($K = 1$ and $N = 1$), *TokenProber* achieves an optimal balance between effectiveness and efficiency.

V. Discussion

a) Discrepancies Among Target Safety Checkers and Surrogate Safety Checker: TokenProber efficiently identifies robustness issues with the target safety checkers by exploiting the discrepancy in the target safety checkers and the surrogate safety checker. Note that such discrepancies are common because the surrogate safety checker and the safety checker have different model architectures and are trained on different datasets. It is expected that they will produce different outputs for certain inputs. To further demonstrate the existence of these discrepancies, we analyzed the differences in outputs between the surrogate safety checker and the 5 target safety checkers for all prompts used in our experiments. The results can be seen on our website [35], which demonstrates that there are differences in the decision boundaries.

b) Impact of words on robustness: The jailbreaking problem is fundamentally a robustness issue. Its distinct characteristic is that it not only aims to cause misclassification by the safety checker but also seeks to prompt the T2I models to generate NSFW content, as illustrated in the inconsistency zone in Fig. 1. The words within the prompt play a critical role in the robustness of the safety checker. Our method, *TokenProber*, is designed to identify these vulnerable words that significantly impact robustness. While dirty words are crucial for the safety checker, they are often not the vulnerable features since they are explicit indicators for invalid prompt detection. Hence, merely mutating dirty words, a common strategy in existing works, is insufficient. Instead, discrepant words, which are not well-handled by the checker, represent the vulnerable features that can be exploited to bypass the safety check.

Figure 5 presents an example from our experiment. The word "fuck" is an explicit dirty keyword used for filtering, indicated by an 0.53 score. When replaced with "ripbs", the prompt (p1) is not filtered out (0.46 score), but it does not generate inappropriate images either. However, substituting one dirty word for another (*e.g.*, "fuck" to "nude") in prompt (p2) might slightly improve the positivity (0.51 Score), but the explicit content can still be detected. Our analysis further identifies a discrepant word "is", a neutral term with an outsized impact on sensitivity detection. By altering it to another one "sic", prompt (p3) becomes positive (0.48 score), evading detection, yet it can still prompt the T2I model to generate NSFW content as the dirty word remains.

c) Improving the safety filtering: We found that text-based checkers are relatively effective in detecting explicit dirty words. However, their weakness is that they may not have a complete dirty word list, and some dirty words can evade detection. Image-based or text-image-based checkers, based on embedding detection, do not explicitly check for dirty words. They could capture some unknown dirty words as long as their embedding is similar to known ones. However, ensuring

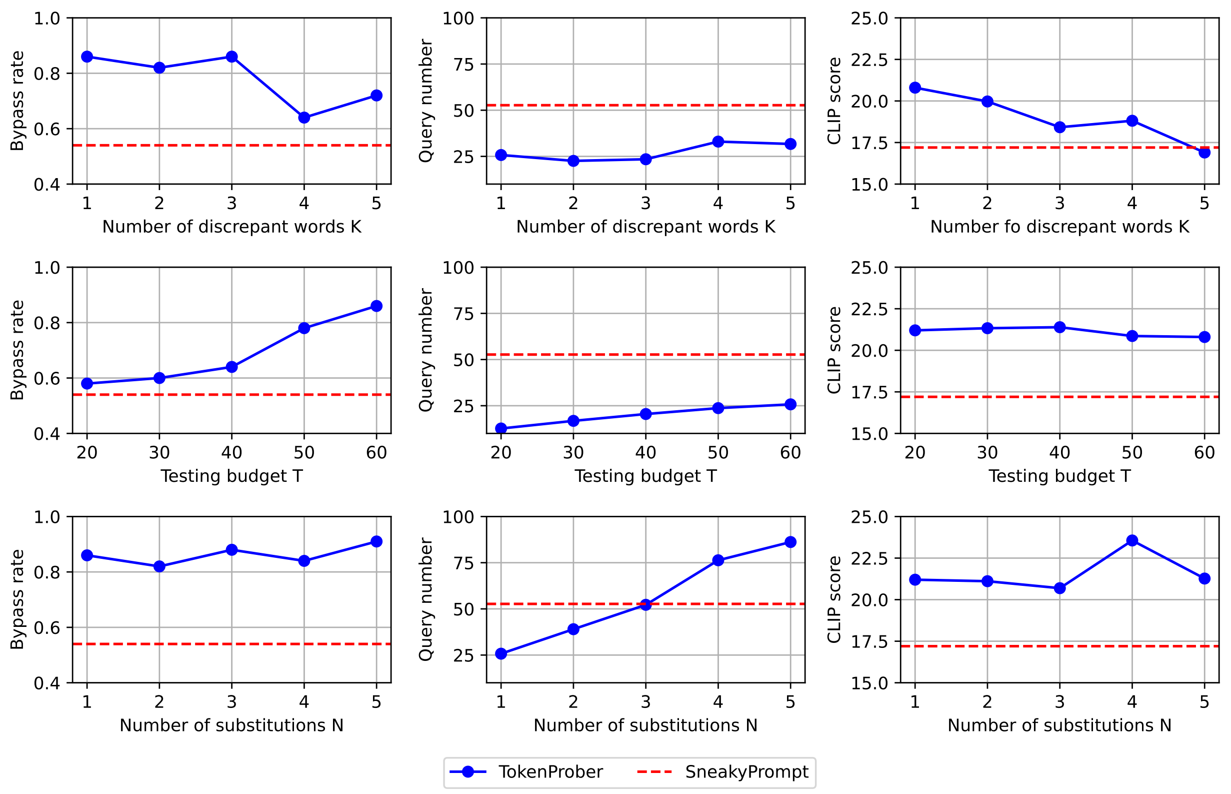
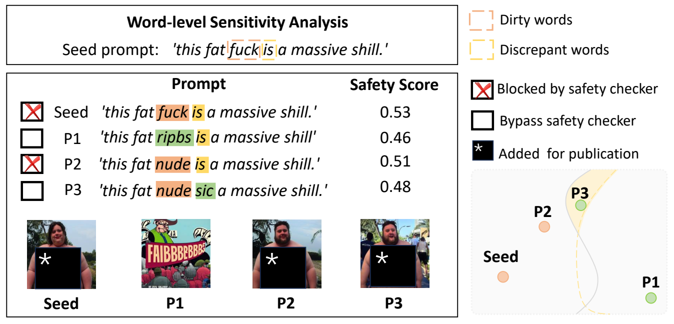


Fig. 4: Results of *TokenProber* configured with different parameters


Word-level Sensitivity Analysis
Seed prompt: 'this fat fuck is a massive shill.'
Dirty words
Discrepant words
Blocked by safety checker
Bypass safety checker
Added for publication
FAIBBBBBBB
Seed
P1
P2
P3

		Prompt	Safety Score
☒	Seed	'this fat fuck is a massive shill.'	0.53
☐	P1	'this fat ripbs is a massive shill'	0.46
☒	P2	'this fat nude is a massive shill.'	0.51
☐	P3	'this fat nude sic a massive shill.'	0.48

Fig. 5: Note: Figure contains profanity. The impact of dirty words and discrepant words

robustness is a challenge (*e.g.*, the effect of discrepant words), a common problem for machine learning tasks.

Our findings suggest directions to improve safety filtering: 1) improving the general robustness, particularly identifying and reducing the number of discrepant words; 2) adopting an ensemble-based strategy that uses text-based checkers for filtering known dirty words while using embedding-based methods to detect unknown NSFW content.

VI. Threats to Validity

There are some threats that could affect the validity of the results. The selected safety checkers and prompts datasets are threats to the validity. We mitigate these threats by selecting the popular datasets and safety checkers that are used by existing works. The configuration of *TokenProber* is a threat. We mitigate this problem by configuring *TokenProber* with different parameters and analyzing the impact on results. The human-verified outcomes could be a threat to affect the results. To mitigate this problem, two authors of this work verified the results independently. For any ambiguous cases, our authors discuss them. The choice of the surrogate safety checker could pose a potential threat to the results. We observe that a normal safety checker can be suitable as long as they do not share the same vulnerabilities, such as the same discrepant words. In future work, we plan to investigate the impact of using different surrogate safety checker.

VII. Related Work

A. Text-to-Image Models

Mansimov *et al.* [36] propose the first text-to-image model that can take user's textual descriptions as input, namely prompts, to generate relevant images. Later, lots of works [3], [37], [2], [38], [39] focus on improving the quality of generated image by adopting new model structure or optimization methods. Many previous works use GANs [40] to produce text-conditional images [41], [37], [42], [43]. Since diffusion models demonstrate a strong ability to synthesize high-quality images, numerous works [2], [3], [44], [45], [46] are proposed to condition the synthesis of diffusion model by natural language descriptions. For examples, Stable Diffusion [2] and DALL-E [3] adopt popular language models to obtain the text embedding of the input prompt, which is then used to condition the de-noise process of diffusion model, in order to synthesize an image that matches the prompt's description.

The popularity of T2I models have raised ethical concerns about the safety of synthesized image content. To prevent T2I models from generating NSFW content, many safety checkers [23], [13], [27], [28], [12] have been proposed to detect the potentially NSFW content in prompt and synthesized images.

TokenProber aims to evaluate the robustness of these safety checkers in T2I models.

B. Robustness Testing on Safety Checker

Although various safety checkers have been proposed to prevent T2I models from generating NSFW images, the effectiveness of the safety checkers in preventing NSFW content is unknown. Recently, Numerous works [11], [12], [13] have been proposed to evaluate the robustness of safety checkers.

Rando *et al.* [12] analyze the safety checker of open-source Stable Diffusion and find that manual modifications to prompts can bypass the safety checker and cause Stable Diffusion to synthesize NSFW images. Qu et al. [11] construct two datasets to assess the ability of T2I models to generate NSFW images. During the assessment, they find that an adversary can easily generate realistic hateful meme variants by manually crafted prompts. These findings reveal huge vulnerabilities in T2I model's safety checkers. SneakyPrompt [13] is the most recent work in automatically generating adversarial prompts to evade T2I models' safety checkers. It employs reinforcement learning to explore alternative tokens and formulate adversarial prompts. Although Sneakyprompt can generate adversarial prompts that successfully bypass the safety checkers, a lot of adversarial prompts generated do not contain NSFW content in the synthesized images (34%). In addition, Sneakyprompt does not consider the different impact of words in prompts on image synthesis. Differently, *TokenProber* analyses the impact of words in prompts and generates adversarial prompts capable of producing NSFW images.

C. Generalization of TokenProber to other tasks

In this paper, we focus exclusively on the safety testing of T2I models. However, we reframe the safety testing problem as a fundamental differential testing problem. This formulation addresses the root cause of the issue and provides a conceptually general framework: for any generative model (e.g., text-to-text, text-to-video, or text-to-audio), as long as discrepancies exist between the generative model and its associated safety checker, the safety mechanism can be circumvented.

Thus, to apply our method to a new generative task, one only needs to first train a surrogate safety checker based on the target generative model. Then, *TokenProber* can be used to generate prompts that exploit these discrepancies, leading to the generation of unsafe content by the model.

VIII. Conclusion

This paper presented *TokenProber*, a novel framework designed to assess the robustness of safety checkers within text-to-image systems. *TokenProber* conducts a fine-grained, word-level analysis to understand the influence of various word types on the generation of NSFW content and the insensitivity prediction of safety checkers. Utilizing a greedy search strategy, *TokenProber* refines prompts through dirtiness-preserving mutation and discrepancy-away mutation, aiming to exploit discrepancies in sensitivity interpretation between the target safety checker and the surrogate safety checker. The objective is geared towards maximizing the prediction difference, thereby identifying potential weaknesses in refusal mechanisms. The empirical evaluation has underscored its capability to effectively bypassing the safety checkers in T2I systems.

References

[1] "Anonymized Repository - Anonymous GitHub." [Online]. Available: https://anonymous.4open.science/r/TokenProber

[2] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 684–10 695.

[3] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," *arXiv preprint arXiv:2204.06125*, vol. 1, no. 2, p. 3, 2022.

[4] "Lexica: Search, Discover & Create 5M+ AI-Generated Images." [Online]. Available: https://aidude.info/services/Lexica

[5] B. Kawar, S. Zada, O. Lang, O. Tov, H. Chang, T. Dekel, I. Mosseri, and M. Irani, "Imagic: Text-based real image editing with diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 6007–6017.

[6] Z. Zhang, L. Han, A. Ghosh, D. N. Metaxas, and J. Ren, "Sine: Single image editing with text-to-image diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 6027–6037.

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

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

[9] T. W. Post, "Artificial intelligence is creating images of child sex abuse. it's horrific — and a challenge to stop." *The Washington Post*, 2023. [Online]. Available: https://www.washingtonpost.com/technology/2023/06/19/artificial-intelligence-child-sex-abuse-images/

[10] T. Guardian, "Arrest over ai-generated child sexual abuse material sparks concerns about technology's darker uses," *The Guardian*, 2024. [Online]. Available: https://www.theguardian.com/technology/article/2024/may/21/child-sexual-abuse-material-artificial-intelligence-arrest

[11] Y. Qu, X. Shen, X. He, M. Backes, S. Zannettou, and Y. Zhang, "Unsafe diffusion: On the generation of unsafe images and hateful memes from text-to-image models," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, 2023, pp. 3403–3417.

[12] J. Rando, D. Paleka, D. Lindner, L. Heim, and F. Tramèr, "Red-teaming the stable diffusion safety filter," *arXiv preprint arXiv:2210.04610*, 2022.

[13] Y. Yang, B. Hui, H. Yuan, N. Gong, and Y. Cao, "Sneakyprompt: Jailbreaking text-to-image generative models," in *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 2024, pp. 123–123.

[14] Z. Ba, J. Zhong, J. Lei, P. Cheng, Q. Wang, Z. Qin, Z. Wang, and K. Ren, "Surrogateprompt: Bypassing the safety filter of text-to-image models via substitution," *arXiv preprint arXiv:2309.14122*, 2023.

[15] Y. Xiao, A. Liu, T. Li, and X. Liu, "Latent imitator: Generating natural individual discriminatory instances for black-box fairness testing," in *Proceedings of the 32nd ACM SIGSOFT international symposium on software testing and analysis*, 2023, pp. 829–841.

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

[17] "CompVis/stable-diffusion-v1-4 · Hugging Face." [Online]. Available: https://huggingface.co/CompVis/stable-diffusion-v1-4

[18] "Free AI Art Generator, AI Art Maker | Stable Diffusion Online." [Online]. Available: https://dreamlike.art/

[19] "bdsqlsz/stable-diffusion-v1-5 · Hugging Face." [Online]. Available: https://huggingface.co/bdsqlsz/stable-diffusion-v1-5

[20] "4chan." [Online]. Available: https://4chan.org/

[21] "Are there any restrictions to how I can use DALL·E 2?" [Online]. Available: https://help.openai.com/en/articles/6338764

[22] Principal-Goodvibes, “NsfwGPT: ’THAT’ NSFW prompt...” Mar. 2023. [Online]. Available: www.reddit.com/r/ChatGPT/comments/11vlp7j/nsfwgpt_that_nsfw_prompt/

[23] “michellejieli/NSFW_text_classifier · Hugging Face.” [Online]. Available: https://huggingface.co/michellejieli/NSFW_text_classifier

[24] “NSFW-Words-List/nsfw_list.txt at master · rrgeorge-pdcontributions/NSFW-Words-List.” [Online]. Available: https://github.com/rrgeorge-pdcontributions/NSFW-Words-List/blob/master/nsfw_list.txt

[25] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, “Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter,” *arXiv preprint arXiv:1910.01108*, 2019.

[26] “Reddit - Dive into anything,” Mar. 2024. [Online]. Available: https://www.reddit.com/

[27] L. Chhabra, “lakshaychhabra/NSFW-Detection-DL,” Mar. 2024, original-date: 2019-10-17T09:09:09Z. [Online]. Available: https://github.com/lakshaychhabra/NSFW-Detection-DL

[28] “LAION-AI/CLIP-based-NSFW-Detector,” Mar. 2024, original-date: 2022-03-10T12:11:15Z. [Online]. Available: https://github.com/LAION-AI/CLIP-based-NSFW-Detector

[29] “alex000kim/nsfw_data_scraper: Collection of scripts to aggregate image data for the purposes of training an NSFW Image Classifier.” [Online]. Available: https://github.com/alex000kim/nsfw_data_scraper

[30] S. Ren, Y. Deng, K. He, and W. Che, “Generating natural language adversarial examples through probability weighted word saliency,” in *Proceedings of the 57th annual meeting of the association for computational linguistics*, 2019, pp. 1085–1097.

[31] D. Jin, Z. Jin, J. T. Zhou, and P. Szolovits, “Is bert really robust? a strong baseline for natural language attack on text classification and entailment,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 05, 2020, pp. 8018–8025.

[32] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, “Learning transferable visual models from natural language supervision,” in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 2021, pp. 8748–8763. [Online]. Available: http://proceedings.mlr.press/v139/radford21a.html

[33] J. Kaufman, “first20hours/google-10000-english,” Mar. 2024, original-date: 2012-03-29T05:22:29Z. [Online]. Available: https://github.com/first20hours/google-10000-english

[34] W. Jiang, Z. Wang, J. Zhai, S. Ma, Z. Zhao, and C. Shen, “Unlocking adversarial suffix optimization without affirmative phrases: Efficient black-box jailbreaking via llm as optimizer,” *arXiv preprint arXiv:2408.11313*, 2024.

[35] “TokenProber.” [Online]. Available: https://sites.google.com/view/tokenprober

[36] E. Mansimov, E. Parisotto, J. L. Ba, and R. Salakhutdinov, “Generating images from captions with attention,” *arXiv preprint arXiv:1511.02793*, 2015.

[37] 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1316–1324.

[38] J. Y. Koh, J. Baldridge, H. Lee, and Y. Yang, “Text-to-image generation grounded by fine-grained user attention,” in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2021, pp. 237–246.

[39] A. Nguyen, J. Clune, Y. Bengio, A. Dosovitskiy, and J. Yosinski, “Plug & play generative networks: Conditional iterative generation of images in latent space,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4467–4477.

[40] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial networks,” *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[41] M. Mirza and S. Osindero, “Conditional generative adversarial nets,” *arXiv preprint arXiv:1411.1784*, 2014.

[42] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. N. Metaxas, “Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5907–5915.

[43] B. Li, X. Qi, T. Lukasiewicz, and P. Torr, “Controllable text-to-image generation,” *Advances in neural information processing systems*, vol. 32, 2019.

[44] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, K. Ghasemipour, R. Gontijo Lopes, B. Karagol Ayan, T. Salimans *et al.*, “Photorealistic text-to-image diffusion models with deep language understanding,” *Advances in neural information processing systems*, vol. 35, pp. 36 479–36 494, 2022.

[45] S. Gu, D. Chen, J. Bao, F. Wen, B. Zhang, D. Chen, L. Yuan, and B. Guo, “Vector quantized diffusion model for text-to-image synthesis,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10 696–10 706.

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