PolyJuice Makes It Real: Black-Box, Universal Red Teaming for Synthetic Image Detectors

Sepehr Dehdashtian[1*†] **Mashrur M. Morshed**[1*] **Jacob H. Seidman**[2] **Gaurav Bharaj**[2]
Vishnu Naresh Boddeti[1]

[1] Michigan State University [2] Reality Defender
{sepehr,morshedm,vishnu}@msu.edu {jacob,gaurav}@realitydefender.ai

Abstract

Synthetic image detectors (SIDs) are a key defense against the risks posed by the growing realism of images from text-to-image (T2I) models. Red teaming improves SID's effectiveness by *identifying* and *exploiting* their failure modes via misclassified synthetic images. However, existing red-teaming solutions (i) require white-box access to SIDs, which is infeasible for proprietary state-of-the-art detectors, and (ii) generate image-specific attacks through expensive online optimization. To address these limitations, we propose PolyJuice, the first *black-box*, image-agnostic red-teaming method for SIDs, based on an observed *distribution shift* in the T2I latent space between samples correctly and incorrectly classified by the SID. PolyJuice generates attacks by (i) *identifying* the direction of this shift through a lightweight offline process that only requires black-box access to the SID, and (ii) *exploiting* this direction by universally steering all generated images towards the SID's failure modes. PolyJuice-steered T2I models are significantly more effective at deceiving SIDs (up to 84%) compared to their unsteered counterparts. We also show that the steering directions can be estimated efficiently at lower resolutions and transferred to higher resolutions using simple interpolation, reducing computational overhead. Finally, tuning SID models on PolyJuice-augmented datasets notably enhances the performance of the detectors (up to 30%).

Figure 1: (a) PolyJuice *steers* text-to-image (T2I) models to generate images that deceive a synthetic image detection (SID) model. (b) There exists a clearly observable shift between the distribution of the samples predicted as real versus those identified as fake, in the latent space of T2I models.

1 Introduction

Importance of synthetic image detection (SID) and red teaming. Creating artificial content has never been easier. In recent years, rapid advances in text-to-image (T2I) generative models

[*] Equal contribution. †This work was done during an internship at Reality Defender.
Project webpage: https://sepehrdehdashtian.github.io/Papers/PolyJuice

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

[18, 51, 52, 54, 44, 19, 6] have significantly blurred the line between real and fake visual media. To reinforce this line, it is necessary to improve the primary defense: synthetic image detection (SID) or deepfake detection (DFD) models [12, 47, 33, 63, 20, 56, 60]. One tried-and-true method for improving the robustness of critical machine learning systems is *red teaming*: simulating attacks to discover failure modes. For example, red teaming is extensively used to expose vulnerabilities in biometric recognition systems [25, 70, 31, 38], and in large-language models (LLM) [2, 36, 17, 49]. Similarly, these attacks can help in improving SID models by discovering their blind spots.

Red teaming as a way to improve SIDs. An ideal red teaming for SID needs to have two features: **(i) Augmentation:** Fake detection datasets have not kept up with the rapidly improving field of image generation [7]. To help improve these datasets, red teaming should discover a distribution of synthetic images capable of deceiving SID models. These images can be used to augment datasets with more challenging examples, helping to improve both evaluation and detection. **(ii) Black-Box Attacks**: State-of-the-art SID methods are mostly proprietary [7] and provide limited access through APIs, making white-box attacks infeasible. Therefore, a practical red teaming method for SID must focus on black-box attacks that can be performed with model-provided responses only.

Existing red-teaming methods and their limitations. Recent years have seen the emergence of a family of attacks known as Unrestricted Adversarial (UA) attacks, that utilize powerful generative priors to directly create attack images capable of misleading a classifier [57]. However, existing UA attacks share some common limitations: **(L1) They perform white-box attacks**: Prior UA attacks [65, 13, 9] need access to the weights or gradients of the classifier to guide or modify the result of the generative model. These attacks can only be adapted to black-box detectors through the unreliable method of transferability from white-box settings. **(L2) They perform image-specific attacks**: Existing UA attacks need to optimize a *per-image* perturbation or direction in the latent space of generative models, which adds a considerable computational overhead at inference time. Further, the optimization cost can grow exponentially with the image resolution.

In this paper, we present PolyJuice, a black-box method for generating UA attacks against SID models. For a given T2I generator and a target SID model, PolyJuice *steers* the T2I generative process to discover images that successfully deceive the detector, as shown in Fig. 1a. PolyJuice identifies the steering direction from a set of generated images, by characterizing the subspace where the statistical dependence between their *T2I latents* and their *SID-predicted realness* is maximal. Finding these directions is motivated by the key observation that there exists a distribution shift between the predicted real and fake samples in the T2I latent space. Fig. 1b illustrates this shift by projecting the T2I latents onto the 2D subspace found using supervised principal component analysis (SPCA) [5].

Finding the steering subspaces does not require access to the weights of the target SID models but only their predicted hard labels, thereby making a black-box attack feasible **(addresses L1)**. We compute these subspaces just once over a set of generated images and their corresponding SID predictions and universally apply them at inference time, thus bypassing the overhead of computing image-specific directions **(addresses L2)**. The universal applicability of PolyJuice also implies that the steered images and the set of direction-finding images can be heterogeneous. For example, the directions found from a set including animal and common object images can steer human images (see Fig. 1a). In addition, we find that PolyJuice directions at lower resolutions remain valid even at higher resolutions, avoiding the costs associated with 1) generating the direction-finding image set in a higher resolution, and 2) finding the subspaces in higher-dimensional space.

Contributions.

- We introduce PolyJuice, the first **(a) black-box** and **(b) distribution-based** unrestricted adversarial attack on synthetic image detectors, which enables computationally efficient red teaming on commercial and proprietary models, even with API-only access.
- PolyJuice **significantly enhances** the success rate of a T2I attacker in deceiving a target SID model, even when it is specifically tuned on images from the very same T2I model.
- PolyJuice enables **efficient attacks** for high-resolution images as PolyJuice directions are **transferable** across image resolutions.
- We achieve the **underlying objective of red teaming**, by using PolyJuice-steered samples to reduce the false negative rate of SID models. The images generated for the development of PolyJuice are to be released for the benefit of future research.

2 The Universal Unrestricted Attack on Synthetic Image Detector Problem

Notation. Scalars are denoted by lowercase letters (e.g., λ, τ). Deterministic vectors and matrices are represented by boldface lowercase and uppercase letters, respectively (e.g., $\boldsymbol{x}$, $\boldsymbol{z}$ for vectors, and $\boldsymbol{H}$, $\boldsymbol{\Theta}$ for matrices). The element in row i and column j of matrix $\boldsymbol{M}$ is denoted by either $(\boldsymbol{M})_{ij}$ or m_{ij}. We use $\boldsymbol{I}_n$ (or $\boldsymbol{I}$) to denote the $n \times n$ identity matrix. Likewise, $\mathbf{1}_n$ and $\mathbf{0}_n$ represent $n \times 1$ vectors of ones and zeros, respectively. For a square matrix $\boldsymbol{K}$, the trace operator is denoted as $\mathrm{Tr}\{\boldsymbol{K}\}$. Sets and vector spaces are represented using calligraphic letters (e.g., $\mathcal{X}$, $\mathcal{Z}$).

Problem Formulation. Let $\mathcal{C}$ denote a set of captions obtained from the distribution of textual descriptions $p(\boldsymbol{c})$, serving as inputs to a T2I generative model G. Given a latent vector $\boldsymbol{z} \in \mathcal{Z}$ from the distribution of latents $p(\boldsymbol{z})$ and a textual prompt $\boldsymbol{c} \in \mathcal{C}$, the T2I model generates an image following $G : \mathcal{Z} \times \mathcal{C} \to \mathcal{X}$, where $\mathcal{X}$ is the space of images.

For obtaining $P(\text{fake} \mid \boldsymbol{x})$ from the perspective of an SID[1], we assume access to a black-box detector model defined as $f : \mathcal{X} \to [0, 1]$, where, given an image $\boldsymbol{x} \in \mathcal{X}$ and a classification threshold $\tau \in (0, 1)$, the SID labels $\boldsymbol{x}$ as *real* if $f(\boldsymbol{x}) < \tau$, and *fake* or *synthetic* otherwise.

Let there be a mapping function h within the T2I latent space parameterized by $\boldsymbol{\Theta}$, defined as

$$h_{\boldsymbol{\Theta}} \colon \mathcal{Z} \to \mathcal{Z}. \tag{1}$$

The primary goal of our UA attack is to find the parameters of the mapping function $h_{\boldsymbol{\Theta}}$ such that latents mapped with $h_{\boldsymbol{\Theta}}$ generate images that are consistently misclassified by the SID model. This transformation of latents within the latent space is typically described as steering. Formally, we frame the attack as the following optimization problem:

Problem 1. Unrestricted Adversarial Attack

$$\max_{\boldsymbol{\Theta}} \mathbb{E}_{\boldsymbol{z} \sim p(\boldsymbol{z}),\, \boldsymbol{c} \sim p(\boldsymbol{c})} \Big[\mathbb{1}\{ f(G(h_{\boldsymbol{\Theta}}(\boldsymbol{z}), \boldsymbol{c})) < \tau \} \Big],$$

where $\mathbb{1}\{\cdot\}$ denotes the indicator function, returning 1 when its argument is true and 0 otherwise.

A practical solution to Prob. 1 requires two underlying constraints: **(C1) Black-box SID constraint:** the SID model must provide thresholded predictions over $f(\boldsymbol{x})$ only, without any access to internal information (e.g., gradients, model weights), reflecting the settings of state-of-the-art proprietary SID models, and **(C2) Universal mapping constraint:** the mapping function $h_{\boldsymbol{\Theta}}$ must generalize across multiple images and prompts, avoiding gradient-based image-specific optimization that are computationally expensive. Solving Prob. 1 thus uncovers systematic vulnerabilities in SID models, enabling the creation of challenging adversarial examples to enhance their robustness.

3 PolyJuice: An Approach to the Universal Unrestricted Attack Problem

Inapplicable Steering Approaches. T2I models can be steered using their learned text conditions to modify certain attributes, such as age, hair color, style, etc. [21, 22, 68]. However, these methods are inapplicable for an abstract property like realness, which is unlikely to be explicitly introduced in the T2I training data. Prior methods perform steering for novel concepts by relying on gradients from an external classifier (i.e., classifier guidance [18]). This approach can be extended to deceive SID models by using gradients obtained from the target detector during the generation process. However, computing gradients requires white-box access to the SID, violating the black-box constraint **(C1)**.

Black-Box Alternative. Even though the T2I model has not learned realness as a semantic property to condition on, we find that the change in SID-predicted realness can be clearly identified as a distribution shift in the T2I model's latent space, as shown in Fig. 1b. This shift can be identified using only hard labels from the target SID model, bypassing the need for white-box access **(satisfies C1)**. In contrast to image-specific mappings, the identified shift direction is universally applicable to arbitrary images **(satisfies C2)**. Our observations hint at the existence of a potential black-box method for performing universal attacks on SID models, consequently motivating PolyJuice.

[1]We use SID to abbreviate both Synthetic Image Detection and Synthetic Image Detector.

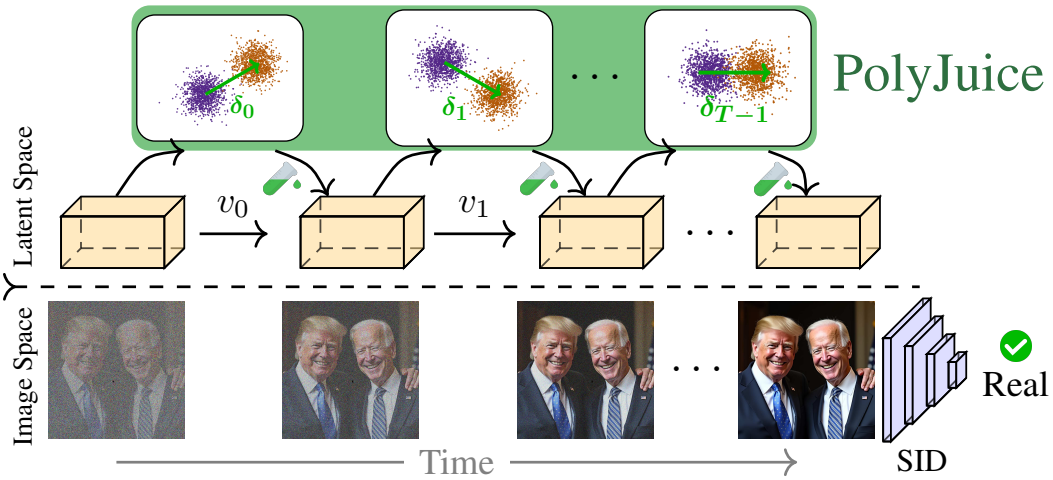


Figure 2: **Overview of PolyJuice:** At each inference step t, PolyJuice manipulates the T2I latent using pre-computed direction $\boldsymbol{\delta}_t$ between predicted real and fake in order to deceive the target SID.

3.1 Discovering the Universal Realness Shifts in Text-to-Image Latent Space

To discover the universal shift, PolyJuice finds the subspace in the T2I latent space that has maximum statistical dependence on the label change of samples. Let $\boldsymbol{Y} = [\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, \cdots, \boldsymbol{y}^{(n)}]^\top$ be the matrix of labels corresponding to the matrix of n latents $\boldsymbol{Z} = [\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)}, \cdots, \boldsymbol{z}^{(n)}]^\top$ where $\boldsymbol{y}^{(i)} \in \mathbb{R}^{d_Y}$ and d_Y is the dimension of label space. There exists an orthogonal matrix $\boldsymbol{U}$ that transforms $\boldsymbol{Z}$ into a subspace where the variance of the labeled data is maximized, as determined by supervised principal component analysis (SPCA) [5]. SPCA captures the dependence between the mapped samples, i.e., $\boldsymbol{U}^\top \boldsymbol{Z}$, and the corresponding labels $\boldsymbol{Y}$, using Hilbert-Schmidt Independence Criterion (HSIC) [24] as the dependence metric. The empirical version of HSIC is defined as

$$\text{HSIC} = \text{Tr}\left\{\boldsymbol{H}\boldsymbol{K}_{\boldsymbol{ZZ}}\boldsymbol{H}\boldsymbol{K}_{\boldsymbol{YY}}\right\}, \tag{2}$$

where $\boldsymbol{H} = \boldsymbol{I}_n - \frac{1}{n}\,\mathbf{1}_n\mathbf{1}_n^\top$ is a centering matrix, $\boldsymbol{K}_{\boldsymbol{ZZ}}$ is a kernel matrix of the mapped data that is formulated as $\boldsymbol{K}_{\boldsymbol{ZZ}} = \boldsymbol{Z}^\top\boldsymbol{U}\boldsymbol{U}^\top\boldsymbol{Z}$ in case of a linear kernel, and $\boldsymbol{K}_{\boldsymbol{YY}} = \boldsymbol{Y}^\top\boldsymbol{Y}$ is a kernel matrix of labels. Therefore, $\boldsymbol{U}$ can be calculated as

$$\begin{aligned} \arg\max_{\boldsymbol{U}} \text{Tr}\left\{\boldsymbol{U}^\top \boldsymbol{Z}\boldsymbol{H}\boldsymbol{K}_{\boldsymbol{YY}}\boldsymbol{H}\boldsymbol{Z}^\top\boldsymbol{U}\right\}, \\ \text{s.t.} \quad \boldsymbol{U}^\top\boldsymbol{U} = \boldsymbol{I}. \end{aligned} \tag{3}$$

The optimal solution for $\boldsymbol{U}$ can be obtained in closed-form by finding the eigenvectors corresponding to the d-largest eigenvalues of $\boldsymbol{A} := \boldsymbol{Z}\boldsymbol{H}\boldsymbol{K}_{\boldsymbol{YY}}\boldsymbol{H}\boldsymbol{Z}^\top$ [43], where d is the dimensionality of the subspace. Here, since we only need a direction vector, we take a convex combination of the d eigenvectors weighted by their corresponding eigenvalues $\{\sigma_k\}_{k=0}^{d-1}$ as

$$\boldsymbol{\delta} = \sum_{k=0}^{d-1} \sigma_k\, \boldsymbol{U}_k. \tag{4}$$

In case of time-varying T2I models such as diffusion and flow-matching models, the latent space is a time-indexed collection $\{\mathcal{Z}_t\}_{t=0}^{T-1}$, that we formulate as a single latent space $\mathcal{Z} = \oplus_{t=0}^{T-1}\mathcal{Z}_t$. From a clean sample $\boldsymbol{z}_T \in \mathcal{Z}_T$, we can compute $\boldsymbol{z}_t \in \mathcal{Z}_t$ for any timestep t as $\boldsymbol{z}_t = a_t\boldsymbol{z}_T + b_t\boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and (a_t, b_t) constitute the variance schedule of the T2I model. By computing a steering direction $\boldsymbol{\delta}_t$ as defined in Eq. (4) within each $\mathcal{Z}_t$, we obtain **a set of steering directions**

$$\Delta = \{\boldsymbol{\delta}_0, \boldsymbol{\delta}_1, \ldots, \boldsymbol{\delta}_{T-1}\}. \tag{5}$$

To solve Prob. 1, we use the steering directions $\boldsymbol{\delta}_t \in \Delta$ to shift the latents in the direction of maximal change between true-positive (TP) and false-negative (FN) samples within each $\mathcal{Z}_t$. Given a set of generated images and predicted labels (0: real, 1: fake), for each timestep t, we solve Eq. (3) and

use Eq. (4) to compute $\boldsymbol{\delta}_t$. Therefore, the mapping function in Prob. 1 becomes a set of time-varying mappings $h_{\boldsymbol{\Theta}} = \{h_{\boldsymbol{\delta}_t}\}_{t=1}^{T-1}$, with $\boldsymbol{\Theta} = \Delta$ and each $h_{\boldsymbol{\delta}_t}$ given by

$$h_{\boldsymbol{\delta}_t}(\boldsymbol{z}'_t) = \boldsymbol{z}'_t + \lambda_t \boldsymbol{\delta}_t, \quad t = 1, \ldots, T-1 \tag{6}$$

where $\boldsymbol{z}'_t = h_{\boldsymbol{\delta}_{t-1}}(\boldsymbol{z}'_{t-1}) + \boldsymbol{v}_t$ given the T2I estimated velocity $\boldsymbol{v}_t$, and $\lambda_t \in [0, \infty)$ controls the steering strength at each time step. As shown in Fig. 2, at each sampling step, PolyJuice uses $\boldsymbol{\delta}_t$ to steer the latent originally updated by $\boldsymbol{v}_t$, to find an attack image that is misclassified as real by the target SID model. **These attack samples help to improve the detector** by finding its semantic and non-semantic failure modes, as corroborated by our results in § 4.3.

3.2 Transferability of PolyJuice Directions from Low to High Image Resolution

Calculating latent directions separately for each desired image resolution incurs computational costs due to the requirement of creating new datasets, adding the noise corresponding to each timestep, and recomputing steering directions. To bypass these costs, we propose *resolution transferability*, which involves computing steering directions once at a base resolution (e.g., 256×256), followed by upscaling them using interpolation and applying them to higher resolution images (e.g., 1024×1024).

Formally, let $\boldsymbol{\delta}_t \in \mathbb{R}^{C\times H\times W}$ represent the steering direction vector at timestep t, calculated at a base resolution $H \times W$ with channel dimension C. When applying this direction at a higher target resolution $H' \times W'$, we first upscale the direction via interpolation as follows:

$$\boldsymbol{\delta}'_t = \text{Interp}(\boldsymbol{\delta}_t; H', W'), \tag{7}$$

where $\text{Interp}(\cdot)$ denotes spatial interpolation. The upscaled directions $\boldsymbol{\delta}'_t \in \mathbb{R}^{C\times H'\times W'}$ are then applied to the corresponding latents at the higher resolution.

The key insight behind resolution transferability stems from the fact that the KL-regularized autoencoders [52] used by recent T2I models focus on perceptual compression and are thus resolution-invariant. T2I latents across different resolutions are likely to maintain similar spatial properties and semantic structures, which makes the computed direction vectors transferable between resolutions to some extent. By employing resolution transferability, PolyJuice avoids the significant computational overhead associated with the generation and preprocessing of a high-resolution synthetic image dataset, i.e., steps that are needed before finding attack directions.

4 Experimental Evaluation

We evaluate the effectiveness of PolyJuice across varying combinations of T2I and SID models. For T2I generation, we utilize SDv3.5 [19], $\text{FLUX}_{\text{[dev]}}$ [6], and $\text{FLUX}_{\text{[sch]}}$ [6], at three image resolutions, 256×256, 512×512, and 1024×1024, respectively. As the target synthetic image detector, we employ Universal Fake Detector (UFD) [47] and RINE [33]—two recent SID methods trained on common objects. For real images and text prompts, we use the Common Objects in Context (COCO) dataset [39], which contains captioned images of common objects. Although there are alternative T2I datasets (e.g., LAION 5B [55]), these datasets contain significant amounts of digitally created content.

Experiment Setting. We strictly assume black-box access to both UFD and RINE; the weights and gradients remain hidden, and querying the SIDs with an image only returns a hard label $Y = 1$ (fake) or $Y = 0$ (real). Each detector predicts a confidence score and then uses a threshold τ to provide the label Y. For RINE, we set this threshold to 0.5, following the same settings used by Koutlis and Papadopoulos [33]. For UFD, we follow the calibration approach of Ojha et al. [47] to adjust the threshold using real images from the COCO training set and a mixed set of generated images from all three T2I models. For each pair of T2I generator and SID model, the set of directions Δ is calculated from 20K true-positive (TP) and 20K false-negative (FN) examples generated from COCO training captions. For evaluation, we generate 1000 images from text descriptions of the COCO validation set; these captions are also used to perform the attack using PolyJuice. As our primary **evaluation metric**, we adopt *success rate* (i.e. *false negative rate*), which is defined as the proportion of fake images the SID erroneously detects as real.

4.1 How Successful is PolyJuice in Attacking Synthetic Image Detectors?

Tab. 1 compares PolyJuice-steered T2I models against unsteered baselines in terms of their ability to deceive SID models. We observe that, among the three unsteered T2I generators, SDv3.5 has the

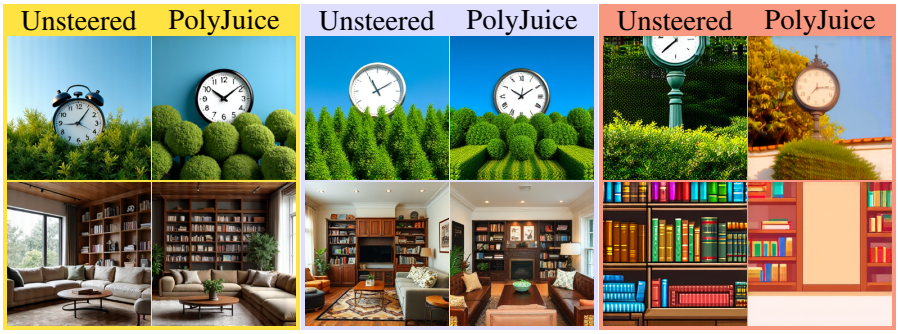


Figure 3: **Unsteered vs. PolyJuice-steered T2I attacks** against UFD. Although UFD detects the unsteered generated images as fake, PolyJuice-disguised samples successfully deceive the detector.

Table 1: Comparison of PolyJuice and baselines w.r.t. success rate (%) on COCO validation set.

	T2I Model	UFD [47]		RINE [33]	
		No Steering	PolyJuice (ours)	No Steering	PolyJuice (ours)
256×256	SDv3.5	12.8	80.6 (**+67**)	15.3	99.7 (**+84**)
	$\text{FLUX}_{[\text{dev}]}$	67.6	96.3 (**+28**)	52.4	81.2 (**+28**)
	$\text{FLUX}_{[\text{sch}]}$	61.7	83.4 (**+21**)	45.4	73.8 (**+28**)
512×512	SDv3.5	30.5	85.0 (**+54**)	26.7	99.6 (**+72**)
	$\text{FLUX}_{[\text{dev}]}$	84.0	98.9 (**+14**)	77.2	96.7 (**+19**)
	$\text{FLUX}_{[\text{sch}]}$	72.7	85.1 (**+12**)	62.9	84.1 (**+21**)
1024×1024	SDv3.5	59.3	93.3 (**+34**)	51.0	99.8 (**+48**)
	$\text{FLUX}_{[\text{dev}]}$	75.6	98.4 (**+22**)	82.4	94.9 (**+12**)
	$\text{FLUX}_{[\text{sch}]}$	72.1	84.0 (**+11**)	69.8	95.6 (**+25**)
	AVG ± STD	59.6 ± 21.8	89.4 ± 6.8	53.7 ± 21.1	91.7 ± 9.0

lowest success rate—suggesting that UFD and RINE are quite effective at detecting SDv3.5-generated fakes, while $\text{FLUX}_{[\text{dev}]}$ and $\text{FLUX}_{[\text{sch}]}$-generated images are harder to detect. However, by steering the image generation process of each T2I model with PolyJuice, the success rate in deceiving the SID models is significantly boosted, even in the case of the easily-detected SDv3.5. To perform attacks for 512 and 1024 resolutions, we compute Δ from 256×256 synthetic images, and transfer them through interpolation (Eq. (7)). Although RINE is more robust than UFD against *unsteered* T2I models, we find that both UFD and RINE are similarly vulnerable against PolyJuice-steered attacks (see Tab. 1 last row for avg. ± std.). From the perspective of the T2I generators, $\text{FLUX}_{[\text{dev}]}$ consistently outperforms $\text{FLUX}_{[\text{sch}]}$, potentially due to having more inference steps (50 vs. 4) to apply PolyJuice. Fig. 3 shows some qualitative examples corresponding to Tab. 1, where the unsteered images are detected as fake. We observe that, regardless of whether the original image looks highly realistic or obviously synthetic, PolyJuice is able to fool the target detector, demonstrating its effectiveness.

Takeaway. PolyJuice boosts attack success rate of T2I models against SIDs by up to **84%**.

4.2 How Effective is PolyJuice When Applied on a T2I Model-Specific Detector?

Motivation. We aim to investigate the extreme scenario where the target SID model is *tuned* using images generated by the very same T2I model employed in the attack.

Table 2: PolyJuice against T2I-specific SIDs at 256×256.

Model	UFD [47]		RINE[33]	
	No Steering	PolyJuice	No Steering	PolyJuice
SDv3.5	18.3	90.8 (**+72**)	8.0	64.8 (**+52**)
$\text{FLUX}_{[\text{dev}]}$	33.7	82.1 (**+48**)	21.8	86.6 (**+64**)
$\text{FLUX}_{[\text{sch}]}$	40.0	64.9 (**+24**)	20.5	29.3 (**+8**)

For each T2I model G, we first calibrate the thresholds for both UFD and RINE using a set of real images from COCO and a set of generated images from G. The goal of the calibration is to improve the overall *accuracy* of the detector in distinguishing real images from fake images generated by a specific T2I model. Next, we recompute the set Δ by obtaining a new set of hard labels from the calibrated models. We then evaluate PolyJuice on the updated SID models. Tab. 2 demonstrates PolyJuice's performance against the tuned SID models. Compared to Tab. 1, we observe that finding a new threshold significantly improves the detection performance in all cases except for

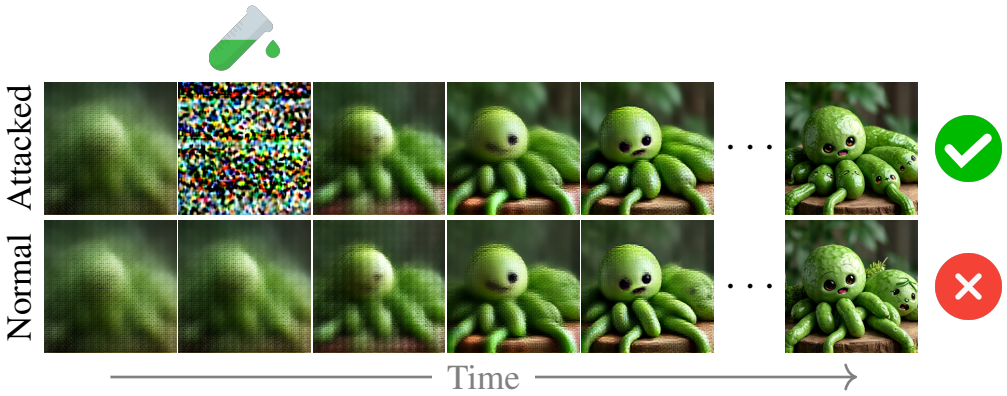


Figure 4: **Estimated clean images at various timesteps** for $\text{FLUX}_{[\text{dev}]}$, where bottom and top rows depict unsteered and PolyJuice-steered generation, respectively.

UFD against SDv3.5. Both UFD and RINE significantly improve their detection of fake images from $\text{FLUX}_{[\text{dev}]}$ and $\text{FLUX}_{[\text{sch}]}$ under the new settings. In addition, RINE demonstrates an overall stronger detection capability compared to UFD. Against the improved detectors, PolyJuice still notably boosts the success rate, as much as 72% (in the case of SDv3.5). In this extreme case, we find that $\text{FLUX}_{[\text{sch}]}$ shows the least success among the three T2I models. Our combined observations from Tab. 2 and § 4.1 suggest that $\text{FLUX}_{[\text{sch}]}$ has limited steerability, potentially due to fewer inference steps.

Takeaway. Even in the extreme case when the target SID model is calibrated on a specific T2I model G, PolyJuice can still improve the FNR up to 72% while using G for generating its attacks.

4.3 How Effective is PolyJuice in Reducing False Negative Rate of Existing SIDs?

To observe the effectiveness of PolyJuice in improving target SID models, we first attack the target detectors with PolyJuice-steered T2I models. Next, we calibrate each SID using a combination of (i) real images from COCO, (ii) regular synthetic images from a T2I model (i.e. unsteered T2I models), and (iii) PolyJuice-steered successful attack images. In Tab. 3, we compare the *pre-* and *post*-calibration SID models (Pre-Cal. and 💉-Cal.) on a set of synthetic COCO validation images at two different resolutions. We find that after PolyJuice-aided calibration, both UFD and RINE show significant improvements in detecting regular (unsteered) generated images.

Table 3: Comparison of SID models' FNR before and after PolyJuice-aided calibration.

	T2I Model	UFD [47]		RINE [33]	
		Pre-Cal.	💉-Cal.	Pre-Cal.	💉-Cal.
256	SDv3.5	13.4	7.5 (**-5**)	15.1	3.8 (**-11**)
	$\text{FLUX}_{[\text{dev}]}$	69.2	47.0 (**-22**)	52.0	21.8 (**-30**)
	$\text{FLUX}_{[\text{sch}]}$	64.3	43.7 (**-20**)	39.6	18.4 (**-21**)
512	SDv3.5	31.1	16.1 (**-15**)	17.8	4.7 (**-13**)
	$\text{FLUX}_{[\text{dev}]}$	86.2	70.3 (**-15**)	69.4	41.4 (**-28**)
	$\text{FLUX}_{[\text{sch}]}$	71.8	55.2 (**-16**)	50.9	25.4 (**-25**)

Takeaway. Calibrating SID models using PolyJuice reduces their vulnerabilities by up to **30%**.

4.4 How Transferable are the Directions from Lower to Higher Resolutions?

We evaluate the *transferability* (§ 3.2) of low-resolution 256×256 directions by comparing their attack performance against attacks generated by 512×512 directions, as shown in Tab. 4. Both the transferred and original directions enable PolyJuice to significantly improve the FNR over the unsteered baseline. Further, PolyJuice attacks using transferred low-resolution directions achieve a comparable or higher FNR than attacks with original high-resolution directions. This result suggests that the error in SPCA-discovered directions is proportional to the dimensionality of its input space due to the curse of dimensionality [32].

Table 4: Resolution transferability of PolyJuice on 512×512.

Model	RINE[33]		
	No Steering	Original	Transferred
SDv3.5	26.7	77.6	**99.6**
$\text{FLUX}_{[\text{dev}]}$	77.2	95.7	**96.7**
$\text{FLUX}_{[\text{sch}]}$	62.9	79.9	**84.1**

Takeaway. PolyJuice directions transferred from lower resolutions achieve success rates comparable to or better than original high-resolution directions, while being less expensive to find.

4.5 How Does PolyJuice Modify the Image Generation Process?

To better understand how PolyJuice works, in Figs. 4 and 5, we illustrate the effect of the steering direction, $\boldsymbol{\delta}_t$, in the T2I image space and the projected latent space, respectively. In Fig. 4, we visualize the estimated clean images at different timesteps. PolyJuice steers the latent towards the blind spot of the target SID (at the second image), resulting in a successful attack (top) that slightly differs from the unsteered counterpart (bottom) while maintaining the semantics of the image. In Fig. 5, we provide an SPCA-based 2D visualization of the update directions of the latents (colored by angle), where (left) and (right) depict unsteered and PolyJuice-steered, correspondingly. We observe that a majority of PolyJuice-steered updates are aligned along a common direction—the direction that leads to deceiving the SID. See Appendix for more details.

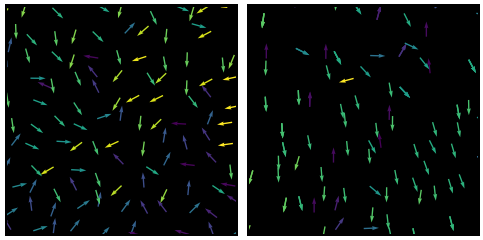
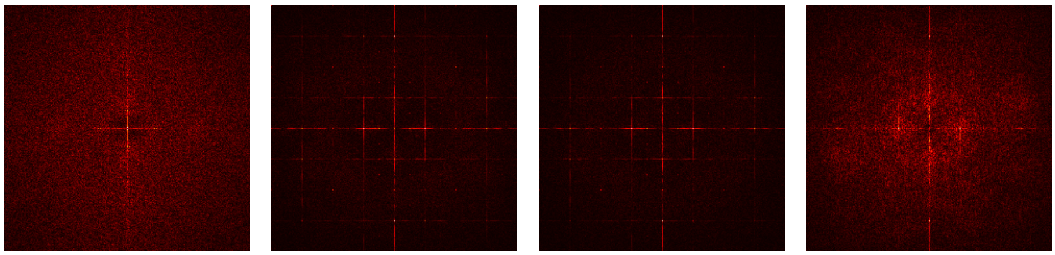


Figure 5: (left) Unsteered (right) PolyJuice-steered samples projected on 2D subspace.

4.6 How Does PolyJuice Affect the Spectral Fingerprint of the T2I Models?



(a) Real Samples (b) TP Samples (c) FN Samples (d) PolyJuice Attacks

Figure 6: **Average Frequency Spectra** of COCO images and generated counterparts.

Prior work [47, 12] uncovers the existence of *fingerprints* left by T2I models through spectral analysis on the synthetic and real images. In Fig. 6, we present a similar approach on (a) the set of real images, (b) the SDv3.5-generated images correctly detected by UFD (TP), (c) generated images misclassified by UFD (FN), and (d) successful PolyJuice attacks. First, we observe that there is a clearly noticeable difference between the real images and the unsteered SDv3.5-generated images (TP + FN), which is the fingerprint of SDv3.5. However, it can be seen that PolyJuice obfuscates this fingerprint, as the residuals of PolyJuice-steered images (Fig. 6d) appear closer to the real image spectrum (Fig. 6a). We provide implementation details and spectral analysis for other T2I models in the Appendix.

Takeaway. PolyJuice disguises the fingerprints left by T2I models in the frequency domain.

4.7 How Do PolyJuice Attacks Appear to the Eyes of the Target SID?

In Fig. 7, we provide a 2D visualization of PolyJuice-steered attacks and unsteered samples from SDv3.5 in the embedding space of CLIP ViT-L/14 [50] using SPCA dimensionality reduction. We illustrate the prediction landscape of the target SID using a kernel density estimate, where darker areas denote input regions with high density of predicted realness. From Fig. 7, we observe that there is a region of perceived realness in the SID's input space that remains unexplored by the unsteered T2I model. PolyJuice is able to *identify* this region and *exploit* it to deceive the SID.

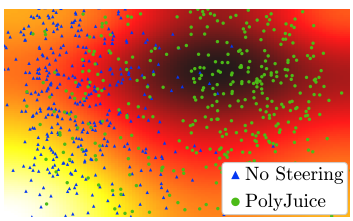


Figure 7: CLIP's embedding space.

Takeaway. PolyJuice steers samples into failure regions missed by standard T2I generation.

Figure 8: Attacks generated by PolyJuice-steered $\text{FLUX}_{[\text{sch}]}$ model that were able to deceive RINE.

5 Does PolyJuice Conserve Image Quality?

As the goal of red teaming is to discover and mitigate the failure modes, finding attacks that are unrealistic (e.g., cartoonish features that fool the SID) is also important, since it reveals the failure mode of the target SID. An ideal SID must trivially detect these unrealistic images as fake. However, it is still important to generate attacks that are *prompt faithful*, and of *reasonable image quality*.

Table 5: Image quality comparison for images generated by unsteered $\text{FLUX}_{[\text{sch}]}$ and PolyJuice, evaluated using various distribution-based metrics.

Method	**FID** ↓	**cFID** ↓	**Prec.** ↑	**Rec.** ↑	**Den.** ↑	**Cvg.** ↑
Unsteered	17.65	17.81	0.495	0.485	0.585	0.801
PolyJuice	17.23	17.41	0.485	0.498	0.764	0.794

Distribution-level image quality. We compare an unsteered $\text{FLUX}_{[\text{sch}]}$ and a PolyJuice-steered $\text{FLUX}_{[\text{sch}]}$ against the RINE detector, at 256×256 image resolution. We use a randomly sampled set of 50K real images from the COCO dataset, and the same number of images from unsteered and PolyJuice-steered $\text{FLUX}_{[\text{sch}]}$ respectively, and present our findings in Tab. 5. First, we compute both classical FID [26] and CleanFID [48], and find a similar trend: the FID w.r.t. real COCO images overall remains quite similar between both methods, suggesting similar quality (also validated by similar Precision (Prec.) and Recall (Rec.) scores [34]). These metrics suggest that **PolyJuice conserves image quality**, which rules out the possibility of PolyJuice deceiving SIDs by causing image degradations. Further, when comparing density and coverage [45], we notice that PolyJuice has a negligible effect on Coverage (Cov.), but yields a significantly improved Density (Den.) score. This suggests that PolyJuice-steered images are more likely to reside in neighborhoods that are densely packed with real data points—which is aligned with the goals of red teaming. We show a few qualitative examples of attacks generated with PolyJuice-steered $\text{FLUX}_{[\text{sch}]}$ in Fig. 8; more examples are shown in § B.

Table 6: Image-level quality and prompt faithfulness assessment of PolyJuice vs. unsteered $\text{FLUX}_{[\text{dev}]}$ using CLIP-based scores.

	UFD						RINE					
	SR		CLIP-IQA		CLIP Score		SR		CLIP-IQA		CLIP Score	
Res	Unst.	PolyJ.	Unst.	PolyJ.	Unst.	PolyJ.	Unst.	PolyJ.	Unst.	PolyJ.	Unst.	PolyJ.
256	67.6	96.3	0.8427	0.8410	30.57	30.45	52.4	81.2	0.8427	0.8457	30.57	30.52
512	84.0	98.9	0.8535	0.8487	30.92	30.94	77.2	96.7	0.8535	0.8526	30.92	30.84
1024	75.6	98.4	0.8657	0.8633	30.86	30.91	82.4	94.9	0.8657	0.8503	30.86	30.69

Image-level quality and prompt faithfulness. To measure per-image quality, we use the **CLIP IQA score** [62]; for evaluating prompt faithfulness, we compute the **CLIP alignment score** between the generated attacks and the input prompts and present the results in Tab. 6. We find that the CLIP scores and CLIP-IQA scores are comparable to those of the unsteered models, suggesting that PolyJuice is not trivially evading detection by degrading the images.

6 Related Work

Finding Directions from Images. Some prior work [30, 67] aim to identify latent space directions corresponding to specific visual attributes. Similar to PolyJuice, Zhang et al. [67] also find directions from a set of images. But unlike PolyJuice, these directions are found in the CLIP's embedding space, with the motivation of addressing biases in T2I generation. More similar to our application, Jain et al. [30] use linear SVM to find a hyperplane that best separates correct from incorrect samples in CLIP latent space, allowing them to find the direction of failure modes. However, Jain et al. [30] aim to interpret failure modes as text prompts in the CLIP's embedding space, while PolyJuice focuses on generating black-box attacks on synthetic image detectors, by steering the latent space of a T2I generative model.

Unrestricted Adversarial (UA) attacks [57] on image classifiers directly use generative models to create attack images, in contrast to classical adversarial attacks that find a low-norm perturbation on clean images. Some early UA approaches [57, 69, 35, 28] involved finding attack directions in the latent space of generative adversarial networks (GANs) [23]. Recent generative models, such as diffusion models and flow matching models [27, 58, 40, 42] have also been employed for performing more sophisticated UA attacks [11, 65, 13, 9, 37, 10].

Diffusion-Based UA Attacks. Diffusion models are well known to be conditioned at inference time with gradients of arbitrary functions, an approach known as *guidance* [3]. To our knowledge, Xue et al. [65] first repurposed guidance for adversarial attacks by formulating a framework for projected gradient descent over the iterative denoising steps of diffusion models (DiffPGD). A concurrent work, AdvDiff [13], performs a two-fold attack: by both guiding the diffusion trajectory with adversarial guidance, and optimizing the initial noise prior with the adversarial gradient until a successful attack is discovered. While DiffPGD and AdvDiff apply the adversarial gradient over several timesteps, an alternative approach, DiffAttack [9], optimizes the diffusion latent only at one particular timestep, while also exploiting attention maps in the model architecture to preserve realness and structure.

Research Gap. Despite their differences, we note that DiffPGD, AdvDiff, and DiffAttack all share a common feature: at inference time, they compute a image-specific adversarial gradient $\nabla_{\boldsymbol{x}_t}\mathcal{L}_{\text{attack}}(\hat{\boldsymbol{x}}_T;\boldsymbol{\theta})$, where $\boldsymbol{x}_t$ is a noisy sample at a timestep t, $\hat{\boldsymbol{x}}_T$ is a predicted clean image, and $\mathcal{L}_{\text{attack}}$ is an adversarial objective on some classifier $\boldsymbol{\theta}$. All these attacks would need complete access to the weights and gradients of an SID model, making them white-box attacks. In contrast, PolyJuice pre-computes a general attack direction from a *distribution* of images, even with only hard label access. We also note that computing the gradient $\nabla_{\boldsymbol{x}_t}\mathcal{L}_{\text{attack}}(\hat{\boldsymbol{x}}_0;\boldsymbol{\theta})$ can be computationally intractable for large, billion-parameter T2I models and high-resolution images; PolyJuice bypasses this problem by (i) being a gradient-free, black-box method (§ 3) and (ii) using resolution transferability (§ 3.2).

7 Concluding Remarks

In this paper, we present PolyJuice, a universal black-box red-teaming method for performing unrestricted adversarial (UA) attacks on synthetic image detectors (SID). To keep up with rapid advances in text-to-image (T2I) generative models, PolyJuice performs the critical task of reinforcing SID models with new attacks and augmented data. Due to its black-box nature, PolyJuice is able to perform attacks on proprietary and commercial detectors that lead the field of deepfake and synthetic image detection. Moreover, PolyJuice is a *brew once, break many* approach: it computes a distribution-based (in contrast to image-specific) attack that is universally applicable over arbitrary images and different resolutions, enabling computationally efficient UA attacks.

Limitations & Future Directions. PolyJuice is based on a linear dependency between predicted realness and the T2I latents; in some cases, this dependency may be nonlinear. We can use a nonlinear kernel in Eq. (3) to discover this dependency; however, this comes with the added complexity of learning a nonlinear decoder or solving a pre-image problem to map the nonlinear embedding space back to $\mathcal{Z}$. In addition, PolyJuice requires a limited hyperparameter search for appropriate values of λ_t, the details of which are expanded in the Appendix.

Ethical Considerations. While adversarial generation is essential for evaluating and improving detector models' robustness, it can also be misused. PolyJuice is intended solely for responsible red teaming, and we strongly oppose its use for malicious purposes. Please refer to § D for defense mechanisms against malicious usage of PolyJuice.

Acknowledgements: Mashrur Morshed and Vishnu Boddeti are partially supported by the National Science Foundation (award #2147116). Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *The 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2623–2631, 2019.

[2] Maksym Andriushchenko, Francesco Croce, and Nicolas Flammarion. Jailbreaking leading safety-aligned llms with simple adaptive attacks. In *ICML 2024 Next Generation of AI Safety Workshop*, 2024.

[3] Arpit Bansal, Hong-Min Chu, Avi Schwarzschild, Soumyadip Sengupta, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Universal guidance for diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 843–852, 2023.

[4] Lorenzo Baraldi, Federico Cocchi, Marcella Cornia, Lorenzo Baraldi, Alessandro Nicolosi, and Rita Cucchiara. Contrasting deepfakes diffusion via contrastive learning and global-local similarities. In *European Conference on Computer Vision*, pages 199–216. Springer, 2024.

[5] Elnaz Barshan, Ali Ghodsi, Zohreh Azimifar, and Mansoor Zolghadri Jahromi. Supervised principal component analysis: Visualization, classification and regression on subspaces and submanifolds. *Pattern Recognition*, 44(7):1357–1371, 2011.

[6] BlackForestLabs. FLUX. https://blackforestlabs.ai/announcing-black-forest-labs/, 2024.

[7] Nuria Alina Chandra, Ryan Murtfeldt, Lin Qiu, Arnab Karmakar, Hannah Lee, Emmanuel Tanumihardja, Kevin Farhat, Ben Caffee, Sejin Paik, Changyeon Lee, et al. Deepfake-eval-2024: A multi-modal in-the-wild benchmark of deepfakes circulated in 2024. *arXiv preprint arXiv:2503.02857*, 2025.

[8] Baoying Chen, Jishen Zeng, Jianquan Yang, and Rui Yang. DRCT: Diffusion reconstruction contrastive training towards universal detection of diffusion generated images. In *Proceedings of the 41st International Conference on Machine Learning*. PMLR, 2024.

[9] Jianqi Chen, Hao Chen, Keyan Chen, Yilan Zhang, Zhengxia Zou, and Zhenwei Shi. Diffusion models for imperceptible and transferable adversarial attack. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[10] Xinquan Chen, Xitong Gao, Juanjuan Zhao, Kejiang Ye, and Cheng-Zhong Xu. Advdiffuser: Natural adversarial example synthesis with diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4562–4572, 2023.

[11] Zhaoyu Chen, Bo Li, Shuang Wu, Kaixun Jiang, Shouhong Ding, and Wenqiang Zhang. Content-based unrestricted adversarial attack. *Advances in Neural Information Processing Systems*, 36, 2023.

[12] Riccardo Corvi, Davide Cozzolino, Giada Zingarini, Giovanni Poggi, Koki Nagano, and Luisa Verdoliva. On the detection of synthetic images generated by diffusion models. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[13] Xuelong Dai, Kaisheng Liang, and Bin Xiao. Advdiff: Generating unrestricted adversarial examples using diffusion models. In *European Conference on Computer Vision*, pages 93–109. Springer, 2024.

[14] Sepehr Dehdashtian, Ruozhen He, Yi Li, Guha Balakrishnan, Nuno Vasconcelos, Vicente Ordonez, and Vishnu Naresh Boddeti. Fairness and Bias Mitigation in Computer Vision: A Survey. *arXiv preprint arXiv:2408.02464*, 2024.

[15] Sepehr Dehdashtian, Bashir Sadeghi, and Vishnu Boddeti. Utility-Fairness Trade-Offs and How to Find Them. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[16] Sepehr Dehdashtian, Lan Wang, and Vishnu Naresh Boddeti. FairerCLIP: Debiasing CLIP's Zero-Shot Predictions using Functions in RKHSs. *International Conference on Learning Representations*, 2024.

[17] Boyi Deng, Wenjie Wang, Fuli Feng, Yang Deng, Qifan Wang, and Xiangnan He. Attack prompt generation for red teaming and defending large language models. *arXiv preprint arXiv:2310.12505*, 2023.

[18] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[19] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

[20] Joel Frank, Thorsten Eisenhofer, Lea Schönherr, Asja Fischer, Dorothea Kolossa, and Thorsten Holz. Leveraging frequency analysis for deep fake image recognition. In *International conference on machine learning*, pages 3247–3258. PMLR, 2020.

[21] Felix Friedrich, Manuel Brack, Lukas Struppek, Dominik Hintersdorf, Patrick Schramowski, Sasha Luccioni, and Kristian Kersting. Fair diffusion: Instructing text-to-image generation models on fairness. *arXiv preprint arXiv:2302.10893*, 2023.

[22] Felix Friedrich, Manuel Brack, Lukas Struppek, Dominik Hintersdorf, Patrick Schramowski, Sasha Luccioni, and Kristian Kersting. Auditing and instructing text-to-image generation models on fairness. *AI and Ethics*, 2024.

[23] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 2014.

[24] Arthur Gretton, Olivier Bousquet, Alex Smola, and Bernhard Schölkopf. Measuring statistical dependence with hilbert-schmidt norms. In *International conference on algorithmic learning theory*. Springer, 2005.

[25] Abdenour Hadid, Nicholas Evans, Sebastien Marcel, and Julian Fierrez. Biometrics systems under spoofing attack: an evaluation methodology and lessons learned. *IEEE Signal Processing Magazine*, 32(5):20–30, 2015.

[26] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

[27] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[28] Yu-Chih-Tuan Hu, Bo-Han Kung, Daniel Stanley Tan, Jun-Cheng Chen, Kai-Lung Hua, and Wen-Huang Cheng. Naturalistic physical adversarial patch for object detectors. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7848–7857, 2021.

[29] Huggingface. Huggingface models. https://huggingface.co/models, 2025.

[30] Saachi Jain, Hannah Lawrence, Ankur Moitra, and Aleksander Madry. Distilling model failures as directions in latent space. *arXiv preprint arXiv:2206.14754*, 2022.

[31] Shuai Jia, Bangjie Yin, Taiping Yao, Shouhong Ding, Chunhua Shen, Xiaokang Yang, and Chao Ma. Adv-attribute: Inconspicuous and transferable adversarial attack on face recognition. *Advances in Neural Information Processing Systems*, 35:34136–34147, 2022.

[32] Mario Köppen. The curse of dimensionality. In *5th online world conference on soft computing in industrial applications (WSC5)*, volume 1, pages 4–8, 2000.

[33] Christos Koutlis and Symeon Papadopoulos. Leveraging representations from intermediate encoder-blocks for synthetic image detection. In *European Conference on Computer Vision*, pages 394–411. Springer, 2024.

[34] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. *Advances in neural information processing systems*, 32, 2019.

[35] Raz Lapid, Eylon Mizrahi, and Moshe Sipper. Patch of invisibility: Naturalistic physical black-box adversarial attacks on object detectors. *arXiv preprint arXiv:2303.04238*, 2023.

[36] Guanlin Li, Kangjie Chen, Shudong Zhang, Jie Zhang, and Tianwei Zhang. ART: Automatic red-teaming for text-to-image models to protect benign users. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=H2ATO32ilj.

[37] Jin Li, Ziqiang He, Anwei Luo, Jian-Fang Hu, Z Jane Wang, and Xiangui Kang. Advad: Exploring non-parametric diffusion for imperceptible adversarial attacks. *Advances in Neural Information Processing Systems*, 2024.

[38] Qian Li, Yuxiao Hu, Ye Liu, Dongxiao Zhang, Xin Jin, and Yuntian Chen. Discrete point-wise attack is not enough: Generalized manifold adversarial attack for face recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 20575–20584, 2023.

[39] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer vision–ECCV 2014: 13th European conference, zurich, Switzerland, September 6-12, 2014, proceedings, part v 13*, pages 740–755. Springer, 2014.

[40] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[41] Huan Liu, Zichang Tan, Chuangchuang Tan, Yunchao Wei, Jingdong Wang, and Yao Zhao. Forgery-aware adaptive transformer for generalizable synthetic image detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10770–10780, 2024.

[42] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

[43] Helmut Lütkepohl. *Handbook of matrices*. John Wiley & Sons, 1997.

[44] Midjourney. Midjourney. https://www.midjourney.com/, 2022.

[45] Muhammad Ferjad Naeem, Seong Joon Oh, Youngjung Uh, Yunjey Choi, and Jaejun Yoo. Reliable fidelity and diversity metrics for generative models. In *International conference on machine learning*, pages 7176–7185. PMLR, 2020.

[46] Nithin Gopalakrishnan Nair, Anoop Cherian, Suhas Lohit, Ye Wang, Toshiaki Koike-Akino, Vishal M Patel, and Tim K Marks. Steered diffusion: A generalized framework for plug-and-play conditional image synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 20850–20860, 2023.

[47] Utkarsh Ojha, Yuheng Li, and Yong Jae Lee. Towards universal fake image detectors that generalize across generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24480–24489, 2023.

[48] Gaurav Parmar, Richard Zhang, and Jun-Yan Zhu. On aliased resizing and surprising subtleties in gan evaluation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11410–11420, 2022.

[49] Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. Red teaming language models with language models. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3419–3448, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.225. URL https://aclanthology.org/2022.emnlp-main.225/.

[50] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[51] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International conference on machine learning*, pages 8821–8831. Pmlr, 2021.

[52] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[53] Bashir Sadeghi, Sepehr Dehdashtian, and Vishnu Boddeti. On Characterizing the Trade-off in Invariant Representation Learning. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856. Featured Certification.

[54] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35:36479–36494, 2022.

[55] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in neural information processing systems*, 35:25278–25294, 2022.

[56] Kaede Shiohara and Toshihiko Yamasaki. Detecting deepfakes with self-blended images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 18720–18729, 2022.

[57] Yang Song, Rui Shu, Nate Kushman, and Stefano Ermon. Constructing unrestricted adversarial examples with generative models. *Advances in neural information processing systems*, 2018.

[58] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[59] Andreas Stathopoulos and Kesheng Wu. A block orthogonalization procedure with constant synchronization requirements. *SIAM Journal on Scientific Computing*, 23(6):2165–2182, 2002.

[60] Zekun Sun, Yujie Han, Zeyu Hua, Na Ruan, and Weijia Jia. Improving the efficiency and robustness of deepfakes detection through precise geometric features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3609–3618, 2021.

[61] Chuangchuang Tan, Yao Zhao, Shikui Wei, Guanghua Gu, Ping Liu, and Yunchao Wei. Rethinking the up-sampling operations in cnn-based generative network for generalizable deepfake detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 28130–28139, 2024.

[62] Jianyi Wang, Kelvin CK Chan, and Chen Change Loy. Exploring clip for assessing the look and feel of images. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 2555–2563, 2023.

[63] Sheng-Yu Wang, Oliver Wang, Richard Zhang, Andrew Owens, and Alexei A Efros. Cnn-generated images are surprisingly easy to spot... for now. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8695–8704, 2020.

[64] Qizhe Xie, Zihang Dai, Yulun Du, Eduard Hovy, and Graham Neubig. Controllable invariance through adversarial feature learning. *Advances in neural information processing systems*, 30, 2017.

[65] Haotian Xue, Alexandre Araujo, Bin Hu, and Yongxin Chen. Diffusion-based adversarial sample generation for improved stealthiness and controllability. *Advances in Neural Information Processing Systems*, 36: 2894–2921, 2023.

[66] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. Scaling autoregressive models for content-rich text-to-image generation. *Transactions on Machine Learning Research*, 2022.

[67] Cheng Zhang, Xuanbai Chen, Siqi Chai, Chen Henry Wu, Dmitry Lagun, Thabo Beeler, and Fernando De la Torre. Iti-gen: Inclusive text-to-image generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3969–3980, 2023.

[68] Yanbing Zhang, Mengping Yang, Qin Zhou, and Zhe Wang. Attention calibration for disentangled text-to-image personalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4764–4774, 2024.

[69] Zhengli Zhao, Dheeru Dua, and Sameer Singh. Generating natural adversarial examples. *arXiv preprint arXiv:1710.11342*, 2017.

[70] Yuhao Zhu, Qi Li, Jian Wang, Cheng-Zhong Xu, and Zhenan Sun. One shot face swapping on megapixels. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4834–4844, 2021.

Appendix

In our main paper, we propose PolyJuice for performing unrestricted adversarial attacks on synthetic image detectors. Here, we provide some additional details to support our main results. The appendix section is structured as follows:

A Implementation Details

A.1 Threat Model

- **Attacker's Goal.** Given a text prompt c and a latent z, the attacker aims to generate synthetic images using a text-to-image (T2I) generative model $G : \mathcal{Z} \times \mathcal{C} \to \mathcal{X}$ that deceive a target synthetic image detector (SID) $f : \mathcal{X} \to [0, 1]$ into being misclassified as real (class 0).
- **Attacker's Capability.**
  - **Black-box access to the SID:** The attacker can only query the SID and observe hard labels Y (real/fake), without access to model weights or gradients.
  - **Full access to the T2I generative model:** The attacker can manipulate the latent space $\mathcal{Z}$ and control the generation process of the T2I model.
  - **Sufficient number of queries:** The attacker can generate a dataset of image–latent–label triplets to analyze the SID's behavior in response to various inputs.
- **Attack Scenario.**
  1. The attacker queries the black-box SID with fake images and obtains hard labels, constructing a dataset of TP and FN samples.
  2. The attacker pre-computes steering directions in the T2I latent space that correlate with an increased probability of being misclassified as real by the SID (Eq. (3)).
  3. At test time, the attacker applies this universal direction to arbitrary prompts, producing images that evade detection by the SID (Eq. (6)).

A.2 T2I Model Settings

We provide the generation settings for SDv3.5, $\text{FLUX}_{[\text{dev}]}$, and $\text{FLUX}_{[\text{sch}]}$ in Tab. 7. All models are available on Huggingface Models [29] with the corresponding Model ID provided in Tab. 7.

Table 7: Settings for T2I Models.

Model	Model ID	Num Steps	Guidance Scale	Max Seq. Length
SDv3.5	`stabilityai/stable-diffusion-3.5-large`	28	3.5	256
$\text{FLUX}_{[\text{dev}]}$	`black-forest-labs/FLUX.1-dev`	50	3.5	512
$\text{FLUX}_{[\text{sch}]}$	`black-forest-labs/FLUX.1-schnell`	4	0	256

A.3 Computing Steering Directions

In Algorithm 1, we provide a pseudocode for calculating the steering direction $\boldsymbol{\delta}_t$ at a given timestep t. Algorithm 1 requires precomputed T2I latents $\boldsymbol{Z}_t \in \mathbb{R}^{N \times d_{\mathcal{Z}}}$, where N is the number of generated images and $d_{\mathcal{Z}}$ is the dimensionality of the T2I latent space. For efficient calculation of the top-k eigenvalues and corresponding eigenvectors, we use the `LOBPCG` algorithm [59].

Algorithm 1 Compute Steering Direction at Timestep t

Require: $\boldsymbol{Z}_t \in \mathbb{R}^{N \times d_{\mathcal{Z}}}$, $\boldsymbol{Y} \in \mathbb{R}^{N \times 2}$ (one-hot SID predicted labels)

$\boldsymbol{Z}_t \leftarrow \boldsymbol{Z}_t - \text{MEAN}(\boldsymbol{Z}_t, \text{dim} = 0)$ ▷ Center the data

$\boldsymbol{C} \leftarrow \boldsymbol{Z}_t^\top \cdot \boldsymbol{Y}$ ▷ Cross-covariance

$\boldsymbol{A} \leftarrow \boldsymbol{C} \cdot \boldsymbol{C}^\top$ ▷ Kernel matrix in Eq. (3)

$(\boldsymbol{\sigma}, \boldsymbol{U}) \leftarrow \text{LOBPCG}(\boldsymbol{A}, k = 2)$ ▷ Top-2 eigenpairs

$\boldsymbol{\delta}_t \leftarrow \boldsymbol{\sigma}_0 \cdot \boldsymbol{U}_0 + \boldsymbol{\sigma}_1 \cdot \boldsymbol{U}_1$ ▷ Weighted steering direction in Eq. (4)

return $\boldsymbol{\delta}_t$

A.4 Hyperparameter Search for λ_t

In Eq. (6), PolyJuice steers a T2I latent $\boldsymbol{Z}_t$ using the steering direction $\boldsymbol{\delta}_t$ and a magnitude coefficient $\lambda_t \in [0, \infty)$. In all steering approaches, finding the correct magnitude for adding the steering direction to the normal flow requires a hyperparameter search [3, 46, 22], and PolyJuice is no exception. As a result, there is a need for finding an optimal λ_t.

To efficiently limit the search space for the set $\{\lambda_t\}_{t=0}^{T-1}$, we consider $\lambda_t = \lambda \cdot \mathbb{1}\{a \leq t \leq b\}$, that is, we only apply the steering directions $\boldsymbol{\delta}_t$ at a constant magnitude λ over some continuous interval $[a, b] \subseteq [0, T)$. We then use a simple hyperparameter search based on the Optuna framework[1] to find (λ, a, b). For any text caption $\boldsymbol{c} \in \mathcal{C}$, we define a *budget*, or the maximum number of attempts that a T2I model can make to deceive an SID. We find that PolyJuice can find a majority of the successful attacks within few attempts (e.g. 10), as we show in Fig. 9. Further, for all captions, we generate all attacks with the same random seed (0) for determinism.

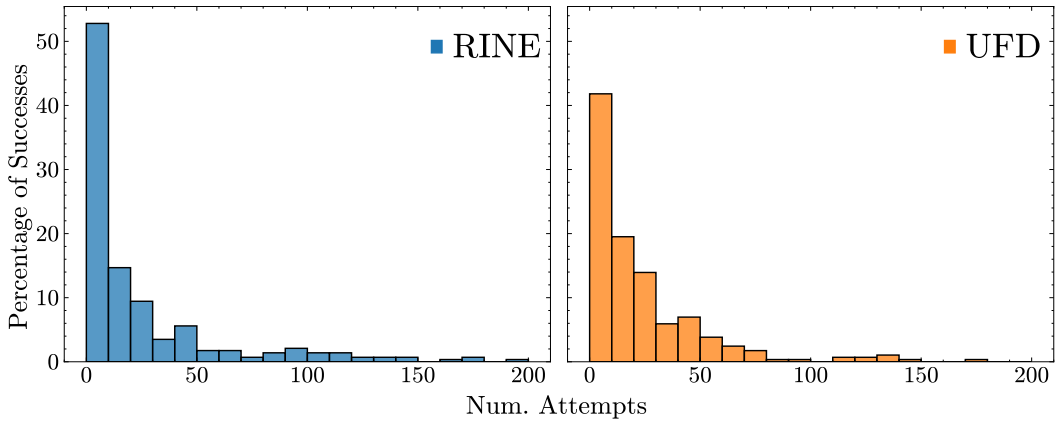


Figure 9: Number of attempts to find successful attacks using $\text{FLUX}_{[\text{dev}]}$.

We also analyze the hyperparameters corresponding to successful attacks as heatmaps. From Fig. 10, we observe that there is a negative correlation between the constant magnitude coefficient λ and the length of the continuous interval $[a, b]$ over which PolyJuice is applied. For higher values of λ, the steering direction $\boldsymbol{\delta}_t$ is applied to only a few steps, while for smaller values, the steering is done on a large number of steps.

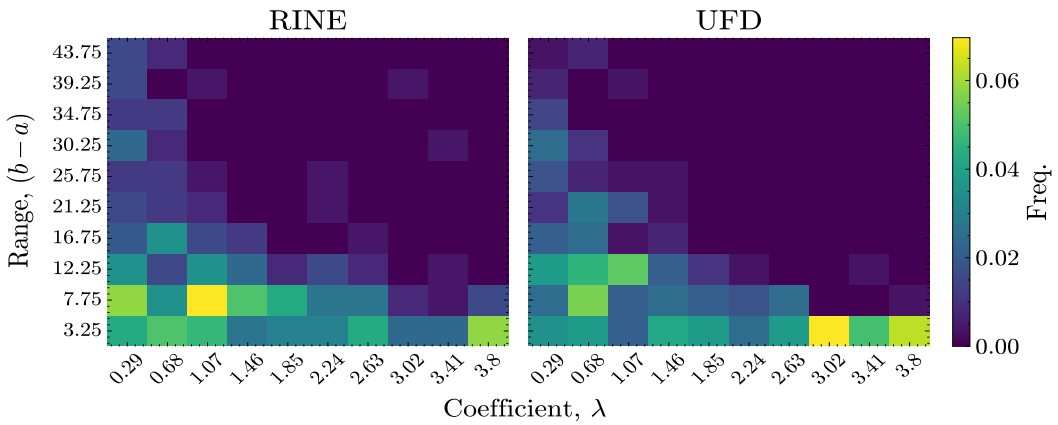


Figure 10: Heatmap showing λ and corresponding range $(b-a)$ for FLUX$_{\text{[dev]}}$.

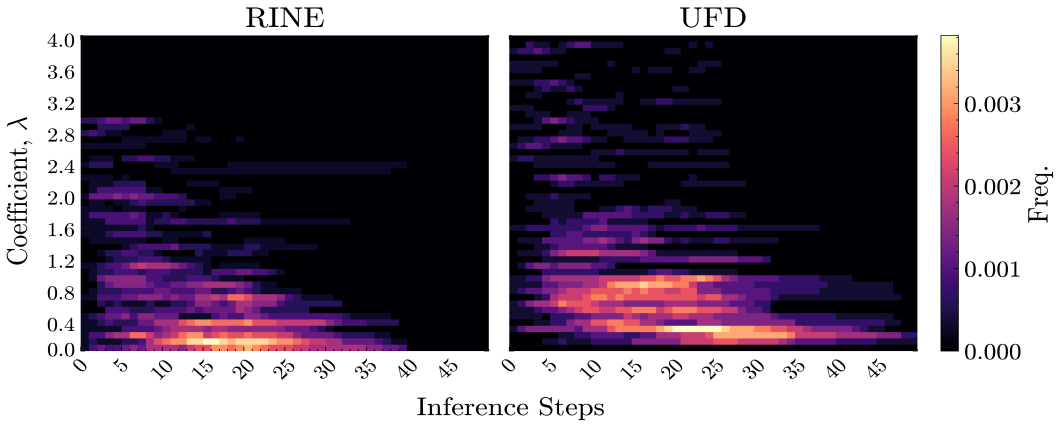


Figure 11: Heatmap showing λ and PolyJuice steering steps for FLUX$_{\text{[dev]}}$.

Previously, Esser et al. [19] show that intermediate timesteps are more important in the image generation process of flow matching-based T2I models. This finding is supported by Fig. 11, which depicts that PolyJuice-steering overall prefers intermediate steps and smaller coefficients.

A.5 Calibrating the Confidence Threshold of an SID

For calibrating the threshold τ of SID models, we follow the algorithm used by Ojha et al. [47]. As shown in Algorithm 2, the algorithm requires the same number of *real* and *fake* samples and their corresponding predicted confidence scores as the input and return the value of the best threshold τ_{best} that maximizes the accuracy. In practice, for real images, we use a set of images from the training split of COCO, while the fake images are generated by T2I models.

In Tab. 1, we calibrate UFD using 20K real images and 20K generated images mixed from all the T2I models. For each T2I-SID pair in Tab. 2, we calibrate the SIDs (-Cal. in Tab. 2) using the following setting. For n successful PolyJuice attacks ($n < 1000$), we use $2n$ real images, n synthetic images generated by the unsteered T2I models, and n PolyJuice-steered images.

A.6 Implementation of Spectral Fingerprint Analysis

Given a generated image $\boldsymbol{x}^{(\text{i})}$, we assume that the high-frequency details in $\boldsymbol{x}^{(\text{i})}$ are a sum of (i) a deterministic component arising from the generative model, i.e. the fingerprint F, and (ii) a random component $\boldsymbol{w}^{(\text{i})} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ [12].

For each group of images, we first apply a denoising filter and estimate a noise residual by computing the difference from the original image. Then we visualize the average residuals in the frequency

Algorithm 2 Find the Best Threshold τ for SIDs (from [47])

Require: Ground truth labels y_{true}, SID predicted scores y_{pred} ▷ where, $|y_{\text{true}} = 0| = |y_{\text{true}} = 1|$
indices $\leftarrow$ ARGSORT(y_{true})
$y_{\text{true}}, y_{\text{pred}} \leftarrow y_{\text{true}}[\text{indices}], y_{\text{pred}}[\text{indices}]$ ▷ Sort y_{true} and y_{pred} according to y_{true}
$N \leftarrow \text{LEN}(y_{\text{true}})$
if $\max(y_{\text{pred}}[0 : \lfloor N/2 \rfloor]) \leq \min(y_{\text{pred}}[\lfloor N/2 \rfloor : N])$ **then** ▷ Perfectly separable real and fake
    return $\frac{1}{2}\left(\max(y_{\text{pred}}[0 : \lfloor N/2 \rfloor]) + \min(y_{\text{pred}}[\lfloor N/2 \rfloor : N])\right)$
end if
best_acc, $\tau_{\text{best}} \leftarrow 0, 0$
for all $\tau \in y_{\text{pred}}$ **do** ▷ Greedily test each y_{pred} as a threshold
    temp $\leftarrow y_{\text{pred}}$
    for $i = 0$ to $N - 1$ **do**
        temp$[i] \leftarrow \mathbb{1}\{\text{temp}[i] \geq \tau\}$
    end for
    acc $\leftarrow \frac{1}{N} \sum_{i=0}^{N-1} \mathbb{1}\{\text{temp}[i] = y_{\text{true}}[i]\}$
    if acc $\geq$ best_acc **then**
        $\tau_{\text{best}} \leftarrow \tau$
        best_acc $\leftarrow$ acc
    end if
end for
return τ_{best}

domain, as shown in Figs. 6 and 20. The algorithm for computing spectral fingerprints is provided in Algorithm 3.

Algorithm 3 Computing Spectral Fingerprint of Images

Require: A set of images X
$X_{\text{denoised}} \leftarrow \text{NL-Mean}(X)$ ▷ Denoise the images
$X_{\text{Residual}} \leftarrow X - X_{\text{denoised}}$ ▷ Compute the residuals
$F \leftarrow \text{FFT}(X_{\text{Residual}})$ ▷ Perform Fast Fourier Transform
$S \leftarrow \|\text{FFT\_SHIFT}(F)\|$ ▷ Shift the zero-frequency component to the center of the spectrum
$S_{\log} \leftarrow \log(1 + S)$
return $S_{\log}$

A.7 Implementation of Projected Steering Directions

Fig. 5 shows the update directions of PolyJuice-steered T2I models in a projected subspace. For a given timestep t, we first solve the eigenvalue problem associated with Eq. (3), to obtain an orthogonal projection matrix $\boldsymbol{U}_t$. The PolyJuice-steering direction at the current timestep, $\boldsymbol{\delta}_t$ is computed from a convex combination over the columns of $\boldsymbol{U}_t$, as described by Eq. (4).

Next, for each latent $\boldsymbol{z}_t$, we map the latent to the subspace by computing $\boldsymbol{U}_t^\top \boldsymbol{z}_t$. We also consider the latent at the next timestep $t + 1$ and similarly project it as $\boldsymbol{U}_t^\top \boldsymbol{z}_{t+1}$. Subsequently, the update vector to the next step can be defined as $\boldsymbol{u}_t = \boldsymbol{U}_t^\top (\boldsymbol{z}_{t+1} - \boldsymbol{z}_t)$.

For both unsteered and PolyJuice-steered T2I latents, we plot the unit update direction $\hat{u}_t$, positioned at the corresponding mapped points $\boldsymbol{U}_t^\top \boldsymbol{z}_t$.

A.8 Experiments Compute Resources

For all experimental steps—including dataset generation, direction computation, and attacks—we used eight NVIDIA RTX A6000 GPUs, each with 48 GB of memory. The primary computational bottleneck arises from the memory requirements of the T2I models during image generation; PolyJuice itself adds negligible overhead. For the highest image resolution considered in this paper, image generation consumed approximately 75% of GPU memory, equivalent to 36 GB.

B Qualitative Analysis of Generated Attacks

We provide some additional qualitative examples of successful attacks from PolyJuice-steered T2I models in Figs. 12 to 17. In general, most of the images look realistic, even though we do not explicitly enforce any realism constraint. However, we notice that there are some characteristics of PolyJuice-generated attacks against specific SID models, which we discuss later in § B.1.

Figure 12: Attacks generated by PolyJuice-steered $\text{FLUX}_{[\text{dev}]}$ model that were able to deceive RINE.

Figure 13: Attacks generated by PolyJuice-steered SDv3.5 model that were able to deceive UFD.

Figure 14: Attacks generated by PolyJuice-steered SDv3.5 model that were able to deceive RINE.

Figure 15: Attacks generated by PolyJuice-steered $\text{FLUX}_{[\text{sch}]}$ model that were able to deceive UFD.

Figure 16: Attacks generated by PolyJuice-steered $\text{FLUX}_{[\text{sch}]}$ model that were able to deceive RINE.

Figure 17: Attacks generated by PolyJuice-steered $\text{FLUX}_{[\text{dev}]}$ model that were able to deceive UFD.

B.1 Common Patterns in Successful Attacks

Figure 18: (Top) Unsteered T2I-generated images that RINE correctly detects as fake. (Bottom) PolyJuice-steered images that successfully deceive RINE as real.

Common Patterns in Successful Attacks on RINE. In successful attacks against RINE, we observe that the generated images often exhibit low brightness. In Fig. 18, we show some images generated by unsteered (top) and PolyJuice-steered (bottom) T2I models, using captions from the COCO validation set. Further, the unsteered and steered images share the same random initial latent (i.e. they are all generated with the random seed 0). Although RINE successfully detects the unsteered images as fake, it is deceived by the relatively *darker* PolyJuice-steered images. This suggests a vulnerability of RINE to synthetic images that appear underexposed or have low brightness levels.

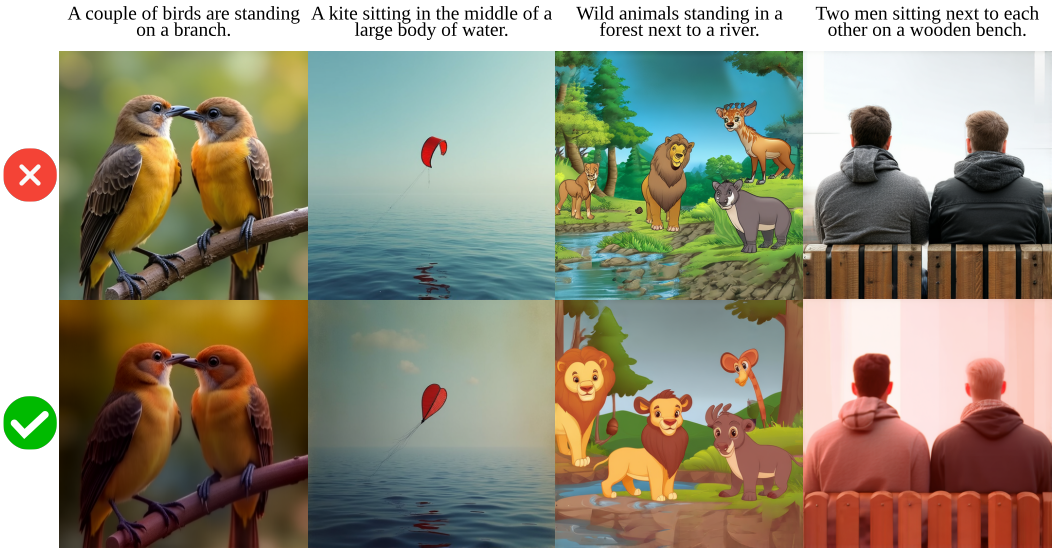


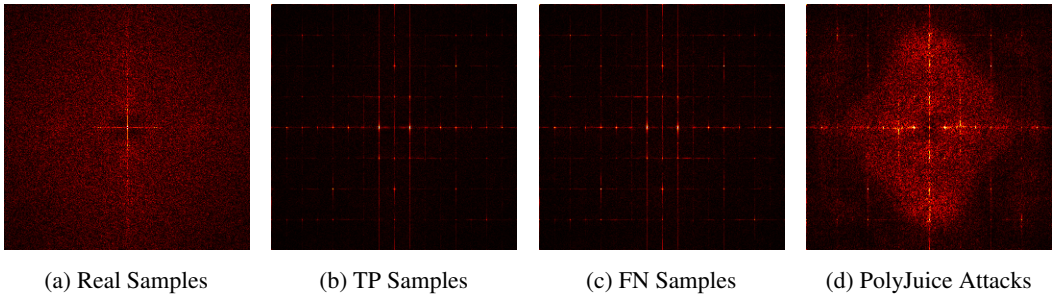
Figure 19: (Top) Unsteered T2I-generated images that UFD correctly detects as fake. (Bottom) PolyJuice-steered images that successfully deceive UFD as real.

Common Patterns in Successful Attacks on UFD. In successful attacks against UFD, we observe that the generated images often exhibit warm colors. Fig. 19 shows some examples unsteered images (top) that UFD detects as fake, and the corresponding PolyJuice-steered images (bottom) that fools the detector. Even on obviously fake images, such as the third column in Fig. 19 (cartoon of wild animals,

generated by SDv3.5), PolyJuice produces an image with a warmer tone that UFD cannot detect as fake. This suggests a vulnerability of UFD to synthetic images with warm color temperatures.

C Additional Results

C.1 Spectral Fingerprint Analysis



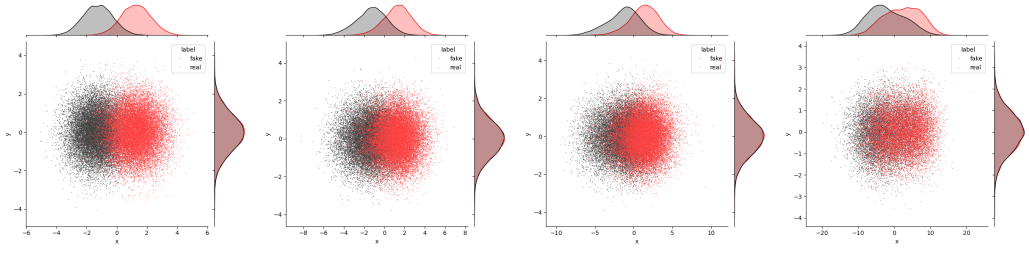
(a) Real Samples (b) TP Samples (c) FN Samples (d) PolyJuice Attacks

(e) Real Samples (f) TP Samples (g) FN Samples (h) PolyJuice Attacks

Figure 20: More results on **Average Frequency Spectra** of COCO images and generated counterparts where 1^{st} and 2^{nd} rows correspond to samples from $\text{FLUX}_{\text{[dev]}}$ and $\text{FLUX}_{\text{[sch]}}$, respectively.

Fig. 6 in the main paper illustrates the spectral fingerprints of real, unsteered, and PolyJuice-steered samples generated by SDv3.5. Here, we extend the analysis to the other two T2I models, $\text{FLUX}_{\text{[dev]}}$ and $\text{FLUX}_{\text{[sch]}}$. The first and second rows of Fig. 20 show the spectral fingerprints of samples generated by $\text{FLUX}_{\text{[dev]}}$ and $\text{FLUX}_{\text{[sch]}}$, respectively. We observe that PolyJuice-steered attacks effectively obscure the characteristic frequency patterns of their underlying T2I models, producing spectra that more closely resemble those of real images compared to unsteered attacks.

C.2 Realness Shift in T2I Latent Space



(a) (b) (c) (d)

Figure 21: More visualizations of the distribution shift in the T2I model's latent space.

In Fig. 1b of the main paper, we showed the realness shift in the latent space of T2I model for one timestep. In Fig. 21, we show the shift for four timesteps. We observe that there exists a clear shift between the predicted real and fake samples. However, the degree of linear separability of these distributions is not constant across different timesteps.

C.3 More Visualizations on the Effect of PolyJuice in Image Generation Process

In Fig. 22, we show additional image-space visualizations of the effect of PolyJuice, by estimating the clean image at various timesteps. As noted in § A.4, we only apply PolyJuice over a continuous interval of inference steps $[a, b] \subseteq [0, T)$, as can be seen from the figure. In both Figs. 22a and 22b, the original unsteered T2I image generation process (bottom rows) produces an image that is detected by the SID as fake. PolyJuice steers the T2I to generate images that deceive the detectors (top rows).

(a) Intermediate steps for "A replica of a bear and her cub in a glass case in an exhibit."

(b) Intermediate steps for "Two men sitting next to each other on a wooden bench."

Figure 22: Visualizing the effect of PolyJuice on the image generation process. (Top) Image that successfully deceives RINE. (bottom) Image that successfully deceives UFD.

C.4 Validity of PolyJuice Across Diverse Prompts

Validity Across Diverse Prompt Categories: To determine whether the effects of PolyJuice are applicable across a variety of categories of generated content, we inspect the COCO validation prompts associated with the attacks generated by PolyJuice and categorize the attacks according to available meta-labels. Tab. 8 shows the fraction of successful attacks per prompt category. We observe that PolyJuice generally improves the success rate across all categories, demonstrating that the discovered direction is universally valid across diverse prompts and image categories.

Validity Across Prompts Beyond COCO: We also evaluate PolyJuice attacks (directions learned from COCO) on a subset of text prompts from the PartiPrompts dataset [66], and present the results in Tab. 9. The results demonstrate the generalizability of the PolyJuice attacks to text prompts outside COCO.

Table 8: Success rate per prompt category in unsteered vs. PolyJuice (on COCO).

	UFD		RINE	
	Unsteered	PolyJuice	Unsteered	PolyJuice
Person	13.7	69.5	16.1	83.5
Animal	12.5	70.8	9.3	90.7
Food	14.4	58.1	11.3	88.8
Vehicle	10.0	73.5	20.1	79.5
Furniture	18.2	66.1	17.3	82.6

Table 9: Attack success rate (%) of PolyJuice on text descriptions from the PartiPrompts dataset.

T2I	**Detector**	**Unsteered**	**PolyJuice (ours)**
SD3.5	UFD	13	75 **(+62)**
	RINE	8	100 **(+92)**
$\text{FLUX}_{\text{[dev]}}$	UFD	78	96 **(+18)**
	RINE	42	86 **(+44)**
$\text{FLUX}_{\text{[sch]}}$	UFD	61	84 **(+23)**
	RINE	31	56 **(+25)**

C.5 Additional SID Detectors

We use PolyJuice-steered SDv3.5 against four additional SID models: NPR[61], FatFormer[41], DRCT [8], and CoDE[4], for further evaluating the effectiveness of PolyJuice. The additional results are shown in Tab. 10. These results show the effectiveness of PolyJuice in improving the success rate of the attacks by 56.7% on CoDE, 75% on DRCT, 82% on NPR, and 56% on FatFormer.

Table 10: SR (%) of unsteered SD3.5 samples vs. PolyJuice.

Detector	**Unsteered SD3.5**	**PolyJuice-Steered SD3.5**
UFD	12.8	80.6 **(+68)**
RINE	15.3	99.7 **(+84)**
CoDE (linear)	43.3	100.0 **(+56)**
DRCT (UFD)	25.3	100.0 **(+74)**
NPR	6.0	87.6 **(+81)**
FatFormer	5.5	62.1 **(+56)**

C.6 Comparison against Transferred Attacks From Diffusion-based White Box Methods

Transferred Attacks are an existing approach of attacking black-box models with white-box methods. First, we consider the RINE model as our target black-box model, and train a surrogate detector (a standard ResNet-50 model) to match its responses. We then extend DiffPGD [65] to $\text{FLUX}_{\text{[dev]}}$ and SDv3.5, and subsequently perform white-box attacks on the surrogate (the attacks have a 100% success rate). The successful attacks are then 'transferred' to the true black-box detector (RINE). From Tab. 11, we observe that PolyJuice has a better attack success rate than the transfer methods.

Table 11: SR(%) of PolyJuice vs. regular (x^n) and realistic (x_0^n) attack from transferred DiffPGD^t.

	FLUX[dev]	**SD3.5**
Unsteered	52.4	15.3
DiffPGD^t (x^n)	57.2	23.5
DiffPGD^t (x_0^n)	70.1	32.1
PolyJuice (ours)	**81.2**	**99.7**

D Potential Defense Mechanisms Against PolyJuice

One possible defense strategy against malicious usage of PolyJuice is collapsing the FN and TP clusters of the data. PolyJuice finds a subspace that is orthogonal to the direction of shift between these two clusters, while preserving the information of the target attribute (i.e., real vs. fake). Approaches from invariant representation learning [53, 64] and fairness [14–16] can be adopted for this purpose. As an example, in the notation of U-FaTE [15], the target Y would be real vs. fake label, while the attribute to be removed, S, would be TP vs. FN.

Additionally, we advocate for detector owners to proactively use PolyJuice to generate challenging attack samples and integrate them into their training or fine-tuning data in order to make it robust to malicious usage of PolyJuice.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We support our claims with our extensive experimental results in § 4.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discussed the limitations and also future directions for improvement in § 7.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: No theoretical result.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide all the implementation details necessary to reproduce the experiments in our paper, in the supplementary. Further, we aim to release the code and data upon acceptance of the paper.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: Although the question is ambiguous to us, we interpret the question as "whether the paper is submitting all code and data as part of supplemental material, at the time of submission", as opposed to "whether the paper will fully release all code and data in the future".

All the text-to-image generative models we use in this paper (SDv3.5, $\text{FLUX}_{[\text{dev}]}$, $\text{FLUX}_{[\text{sch}]}$) are publicly available, and so are the text captions we use in image generation (adopted from MS-COCO). We also include pseudo-code and detailed instructions to reproduce the main results in the supplementary material.

The data used in this paper involves a large number of generated images, that may inherit the biases associated with generative models. Further, PolyJuice is a red-teaming method that may have some potential for misuse (as noted in § 7. As such, we aim to release the full code and dataset at a later time, after an extensive internal audit to address ethical concerns.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We provide the necessary details in the experiments section (§ 4), and also provide additional implementation details in the supplementary material.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We provide the average and standard deviation of our results, computed over multiple text-to-image generative models (SDv3.5, $\text{FLUX}_{[\text{dev}]}$, $\text{FLUX}_{[\text{sch}]}$) and image resolutions (256×256, 512×512, 1024×1024), in Tab. 1.

However, our results depend on generating high-resolution images from large text-to-image generative models. As the experiments are computationally expensive, it is infeasible for us to report error bars for every result cell in the table. To report unbiased and reproducible results, we choose the random seed 0 for all generation tasks.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide compute information in the supplementary material.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: To our knowledge, we abide by all the guidelines presented in the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss positive societal impacts throughout the paper, and also discuss potential negative societal impacts in the concluding remarks.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [Yes]

    Justification: We do not scrape any data from the internet, but use a well-established and audited dataset (COCO). As noted earlier in the checklist (item 5: open access), all materials related to PolyJuice (model, data, code) will be extensively audited for safety prior to release.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We attribute original owners and respect their license, wherever applicable.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: At the moment of submission we are not releasing any new asset.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: No crowdsourcing and research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: No crowdsourcing and research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.