

PixSDS: Why Latent SDS Makes Noisy Pixels

Vsevolod Skorokhodov¹

EPFL, Lausanne, Switzerland
vsevolod.skorokhodov@epfl.ch

Abstract. Score Distillation Sampling (SDS) enables text-to-3D generation by optimizing rendered images with a pretrained diffusion prior, but latent SDS often produces structured color artifacts and high-frequency texture noise. We identify a failure mode of latent SDS caused by VAE-induced pixel drift: the optimized image can move along pixel-space directions that are weakly constrained by the VAE encoder, so its latent representation remains clean and semantically meaningful while the image itself accumulates visible artifacts. We support this diagnosis with controlled 2D SDS experiments, VAE-only optimization, and a simplified analysis showing that encoder-like latent objectives can amplify image-space noise when the inverse mapping to pixels is underconstrained. Motivated by this observation, we propose PixSDS, a lightweight VAE-consistent gradient repair method. PixSDS decodes a latent SDS look-ahead step and uses the decoded image as a clean direction for pixel-space optimization, reducing motion in VAE-inconsistent directions without retraining the diffusion model, changing the renderer, or replacing the SDS objective. Experiments in 2D optimization and text-to-3D generation show that PixSDS substantially reduces structured artifacts while preserving semantic content.

Keywords: diffusion models · score distillation sampling · image generation

1 Introduction

Score Distillation Sampling (SDS) [14] has become a central tool for optimization-based text-to-3D generation. By using a pretrained text-to-image diffusion model as a prior, SDS makes it possible to optimize 3D representations from text prompts without training a dedicated 3D generative model. This flexibility has made SDS useful across neural fields [14, 18], meshes [2], and Gaussian splats [8, 10, 17], especially when large-scale paired text-3D data is unavailable.

Despite its success, SDS often produces visually distracting artifacts [8, 17], including structured color patterns, high-frequency texture noise, and noisy floating geometry. These artifacts are especially common in latent-diffusion-based pipelines, where the diffusion prior operates on the latent representation of a variational autoencoder (VAE) rather than directly on pixels. Prior work has proposed practical remedies, such as modifying the SDS objective [6], changing the optimization procedure [17], or clipping large pixel-space gradients [13].

However, these methods mostly address the symptoms of the artifact, while the underlying cause remains unclear.

In this paper, we argue that a key source of these structured artifacts is the mismatch between pixel-space optimization and the VAE representation used by latent diffusion models. Latent SDS constrains an optimized rendering through its encoded latent, but the inverse mapping from latents to pixels is underconstrained. As a result, the optimized image can drift along pixel-space directions that are weakly visible to the VAE encoder: the latent code remains clean and semantically valid, while the directly optimized pixel image accumulates structured noise.

We support this view with controlled 2D experiments, VAE-only optimization, and a simplified theoretical model. Together, these results show that structured artifacts can arise even without 3D rendering or texture extraction, and that encoder-like latent objectives can amplify image-space noise when the inverse mapping to pixels is underconstrained. This diagnosis suggests that artifact reduction should not merely suppress large gradients, but should instead guide pixel-space updates toward directions that are consistent with the VAE latent space.

Motivated by this observation, we introduce PixSDS, a lightweight VAE-consistent gradient repair method for latent SDS. At each optimization step, PixSDS decodes the latent-space SDS update and uses the decoded image as a clean direction for pixel-space optimization. This repair does not require retraining the diffusion model, modifying the renderer, or changing the underlying 3D representation, and can be combined with existing SDS-style objectives. Experiments in controlled 2D optimization and text-to-3D generation show that PixSDS substantially reduces structured artifacts while preserving semantic content.

In summary, our contributions are:

1. We identify VAE-induced pixel drift as a failure mode of latent score distillation, where pixel-space parameters accumulate structured high-frequency artifacts while their VAE latents remain clean and semantically valid.
2. We provide controlled evidence and a simplified theoretical analysis showing that encoder-like latent objectives can amplify image-space noise when the inverse mapping from latents to pixels is underconstrained.
3. We introduce PixSDS, a lightweight VAE-consistent gradient repair method that decodes the latent-space SDS step and uses it as a clean direction for pixel-space optimization.
4. We show that PixSDS reduces structured artifacts in controlled 2D SDS optimization and improves visual cleanliness when integrated into existing text-to-3D pipelines.

2 Related Works

2.1 Score Distillation Sampling

DreamFusion [14] introduced Score Distillation Sampling (SDS) as a way to optimize a 3D representation using a pretrained 2D diffusion prior. Let $x = R_\theta(c)$ denote a rendering of the current 3D representation under camera c , and let

$$z_t = \alpha_t x + \sigma_t \epsilon$$

be the corresponding noised sample at timestep t . The SDS update can be written as

$$\nabla_\theta \mathcal{L}_{\text{SDS}} = \mathbb{E}_{t,\epsilon} \left[w(t) \left(\frac{\partial x}{\partial \theta} \right)^\top (\hat{\epsilon}_\phi(z_t; y, t) - \epsilon) \right], \quad (1)$$

where $\hat{\epsilon}_\phi$ is the noise predicted by the diffusion model, y is the text prompt, and $w(t)$ is a timestep-dependent weighting term.

Several works improve SDS by changing the distillation objective or modifying its gradient. ProlificDreamer [18] introduces Variational Score Distillation, HiFA [20] combines latent-space and pixel-space guidance, NFSD [6] removes an undesired noise component from the SDS signal, and PGC [13] clips decoded pixel-wise gradients. Recent reformulations such as SDI [10] and SDS-Bridge [11] derive alternative distillation directions from sampling or transport perspectives. These methods can be written abstractly as

$$g_\theta = \mathbb{E}_{t,\epsilon} \left[w(t) \left(\frac{\partial x}{\partial \theta} \right)^\top \Delta_t \right], \quad (2)$$

where Δ_t is a method-specific distillation direction; for vanilla SDS, $\Delta_t = \hat{\epsilon}_\phi(z_t; y, t) - \epsilon$.

Our work is complementary to these methods. Rather than deriving a new score distillation objective, we study why latent SDS can produce structured pixel-space artifacts even when the corresponding latent representation remains clean, and use this analysis to repair the pixel-space update.

2.2 Artifacts and Noise in SDS

SDS-based optimization often suffers from oversaturation, blurriness, and high-frequency texture artifacts [6, 10]. In this work, we focus on structured color artifacts and spatially coherent noise patterns, as illustrated in Fig. 4.

Prior work has attributed such artifacts to different mechanisms. Dream-Gaussian [17] observes that SDS can introduce high-frequency texture noise in 3D Gaussian optimization and relates part of the problem to texture extraction and mipmap sampling. PGC [13] instead focuses on latent diffusion guidance and argues that decoded SDS gradients can contain large pixel-wise outliers, which motivates clipping the pixel-space gradient.

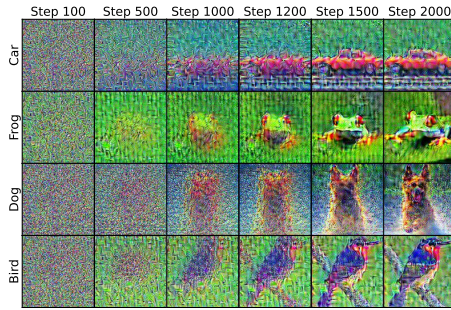


Fig. 1: Small latent diffusion model. SDS with a low-resolution latent diffusion model still produces structured artifacts, suggesting that tensor shape alone does not explain the failure mode.

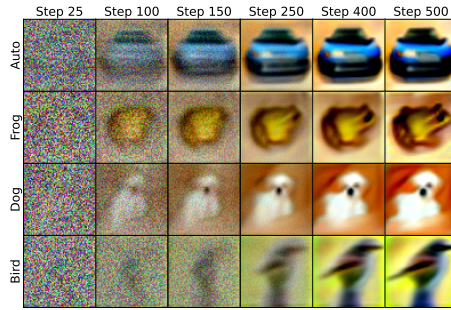


Fig. 2: Small pixel diffusion model. SDS with a pixel-space diffusion model produces realistic images without the same structured artifacts, suggesting that pixel-space optimization alone is not the cause.

Our analysis is complementary to these explanations. We show that similar structured artifacts can arise even in purely 2D optimization, without mesh extraction, mipmap sampling, or a 3D renderer. This suggests that renderer-specific effects may amplify the artifacts, but are not necessary for them to occur. We instead link the artifact formation to the underconstrained geometry of the VAE mapping: an optimized image can drift into noisy pixel-space directions while its encoded latent, and the image decoded from that latent, remain clean.

3 Diagnosing SDS Artifacts

In this section, we study why latent SDS produces structured pixel-space artifacts. Latent SDS differs from pixel-space SDS in several ways: the optimized signal has a different tensor shape, the diffusion prior operates in a different representation space [1, 15], and the update is propagated through the VAE mapping. We isolate these factors with controlled experiments and show that the VAE mapping is a sufficient mechanism for the observed artifacts.

Tensor shape. One possible explanation is that artifacts arise because latent diffusion operates on a compressed low-resolution tensor rather than on an image. To test this, we run SDS with *stable-diffusion-nano-2-1* [3], a Stable Diffusion 2.1 model fine-tuned at 128×128 resolution. As shown in Fig. 1, structured artifacts still appear. This suggests that latent tensor resolution or tensor shape alone does not explain the artifact formation.

Pixel-space optimization. Another possibility is that SDS optimization itself is unstable when applied directly to pixels. To test this, we perform SDS in pixel space using a small conditional diffusion model trained on CIFAR-10 [7] at 64×64 resolution. As shown in Fig. 2, the optimized images remain realistic and do

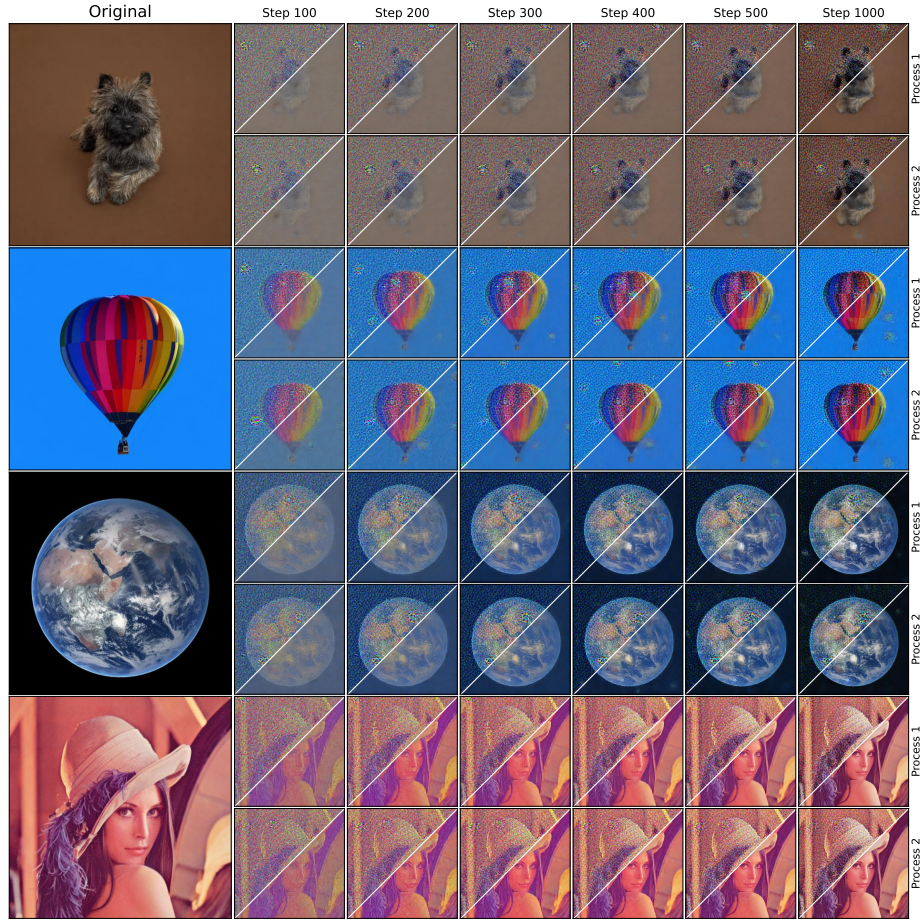


Fig. 3: VAE-only optimization. Column 1 shows the target image X . Columns 2–7 show optimized images Z_i from two different random initializations, with their decoded latents $\text{dec}(\text{enc}(Z_i))$ shown in the bottom-right inset. Even without a diffusion model, optimizing only through the VAE encoder produces structured noise in Z_i , while the decoded latents remain clean.

not exhibit the same structured artifacts as latent SDS. This indicates that optimizing pixels with SDS is not by itself sufficient to produce the observed failure mode.

VAE mapping. The remaining difference is the VAE mapping used by latent diffusion models. In latent SDS, the diffusion loss is defined in the latent space, but the optimized variable may live in pixel space and receive updates through the VAE encoder or decoder. This creates an underconstrained inverse problem: many visually different images can map to similar latent codes. As a result, an

image can move in directions that are weakly constrained by the latent objective while still preserving a clean latent representation.

To isolate this effect from the diffusion model, we optimize an image using only the VAE encoder. Let

$$\text{enc} : \mathbb{R}^{3 \times H \times W} \rightarrow \mathbb{R}^{c \times h \times w}$$

denote the VAE encoder, and let X be a target image with latent code $x = \text{enc}(X)$. Starting from a randomly initialized image $Z \sim \mathcal{U}(0, 1)$, we optimize Z to match the target latent by minimizing

$$\mathcal{L}_{\text{VAE}}(Z) = \|\text{enc}(Z) - x\|_2^2 \quad (3)$$

with SGD and learning rate 0.1. As shown in Fig. 3, the optimized images develop structured noise patterns similar to those observed in latent SDS, even though no diffusion model is used. At the same time, the decoded latents $\text{dec}(\text{enc}(Z))$ remain visually clean. This experiment shows that the VAE mapping alone can produce the clean-latent/noisy-image mismatch.

Toy analogy. A simple low-dimensional example illustrates the same issue. Consider minimizing

$$\ell(x, y) = (x + 2y)^2.$$

The set of minimizers is the line $x + 2y = 0$, so the objective does not select a unique solution. Starting from $(x_0, y_0) = (1, 1)$, gradient descent converges to $(x^*, y^*) = (0.4, -0.2)$, although $(0, 0)$ is also a minimizer. The solution $(0.4, -0.2)$ is noisier than $(0, 0)$. Therefore, the optimization does not necessary converge to a less noisy solution without any additional constraints.

We formalize this intuition in the Appendix (Sec. A). For a 1D convolution with weights $\omega \in \mathbb{R}^m$, $m \geq 3$, and $\sum_{i=1}^m \omega_i \neq 0$, we show that for any $C > 0$ there exists an input $x \in \mathbb{R}^n$ whose initial noise is below C , but minimizing $\frac{1}{2}\|\omega * x\|_2^2$ by gradient descent increases the noise in the solution. Thus, noise amplification can occur even in simple encoder-like objectives.

Together, these experiments show that structured artifacts are not fully explained by latent tensor shape, pixel-space SDS optimization, or renderer-specific effects. Instead, they identify the VAE mapping as a sufficient and previously underemphasized mechanism: latent objectives can keep the encoded representation clean while allowing the optimized image to drift into noisy pixel-space directions.

4 Method

In this section, we introduce PixSDS. The method is motivated by a simple observation: during latent SDS optimization, the optimized image can accumulate structured pixel-space artifacts, while its VAE latent representation and decoded latent remain visually clean. We use this decoded latent update as a VAE-consistent direction to repair the SDS update in pixel space.

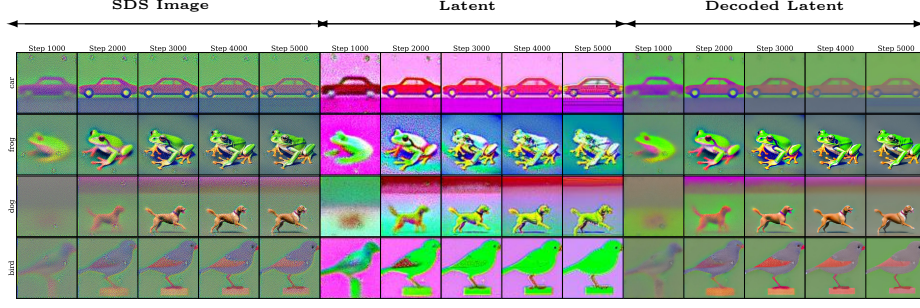


Fig. 4: Clean latents. During SDS optimization, images (columns 1–5) contain structured noise, while their latents (columns 6–10) and decoded latents (columns 11–15) remain visually clean.

4.1 Clean Latents

Directly defining or penalizing the noise produced by SDS is difficult, because the artifacts are structured, spatially dependent, and often entangled with semantic image content. Instead, we rely on a more stable signal provided by the VAE itself. As shown in Fig. 4, although the optimized image may become noisy, its encoded latent remains semantically meaningful, and decoding this latent produces a much cleaner image with similar overall content.

This suggests that the artifact is not fully visible to the latent diffusion model: the image can drift in pixel-space directions that are weakly constrained by the VAE encoder, while its latent representation remains clean. PixSDS uses this observation to construct a clean update direction. Rather than adding an explicit reconstruction loss toward decoded latent, which may pull the image backward toward the previous iterate, we decode the next latent SDS step and use it as a clean approximation of where the image should move. This distinction is important: reconstructing the current latent would only pull the image back toward its present VAE projection, whereas PixSDS decodes the next latent SDS step and therefore repairs the update without discarding the semantic direction of SDS.

4.2 PixSDS

Let $Z \in \mathbb{R}^{C \times H \times W}$ denote the optimized image, and let

$$\text{enc} : \mathbb{R}^{C \times H \times W} \rightarrow \mathbb{R}^{c \times h \times w}, \quad \text{dec} : \mathbb{R}^{c \times h \times w} \rightarrow \mathbb{R}^{C \times H \times W}$$

denote the VAE encoder and decoder. For clarity, we write g_{sds} for the image-space SDS update direction applied to Z , and $g_{\text{sds}}^{\text{latent}}$ for the corresponding latent-space update direction.

At each iteration, PixSDS first computes the standard latent SDS update. Instead of applying the image-space update directly, we also take the corresponding step in latent space and decode it:

$$\hat{Z} = \text{dec}(\text{enc}(Z) - \beta g_{\text{sds}}^{\text{latent}}), \quad (4)$$


```

image = random_image() # [C, H, W]
image_optimizer = get_optimizer(image, image_lr)
diffusion_model = diffusion.load_model()
vae = diffusion.load_vae()
prompt = get_prompt()
for nstep in range(num_iterations):
    latent = vae.encode(image) # [c, h, w]
    # compute SDS gradients in image and latent space
    image_grad, latent_grad = sds_grad(latent, prompt)
    # estimate a clean target after the SDS step
    vae_decoded_image = vae.decode(latent - beta * latent_grad)
    vae_direction = normalize(vae_decoded_image - image, dim=0)
    vae_direction *= norm(image_grad, dim=0)
    # repair the SDS gradient
    image_grad += vae_direction
    image_optimizer.step(image_grad)
return image

```

Fig. 5: Python-like pseudocode of PixSDS. Given a text prompt, PixSDS computes the standard SDS gradient and then adds a clean-up direction obtained by decoding the SDS-updated latent. The `sds_grad(·)` function samples t, ε and computes the SDS gradient (Eq. 1) in image and latent space.

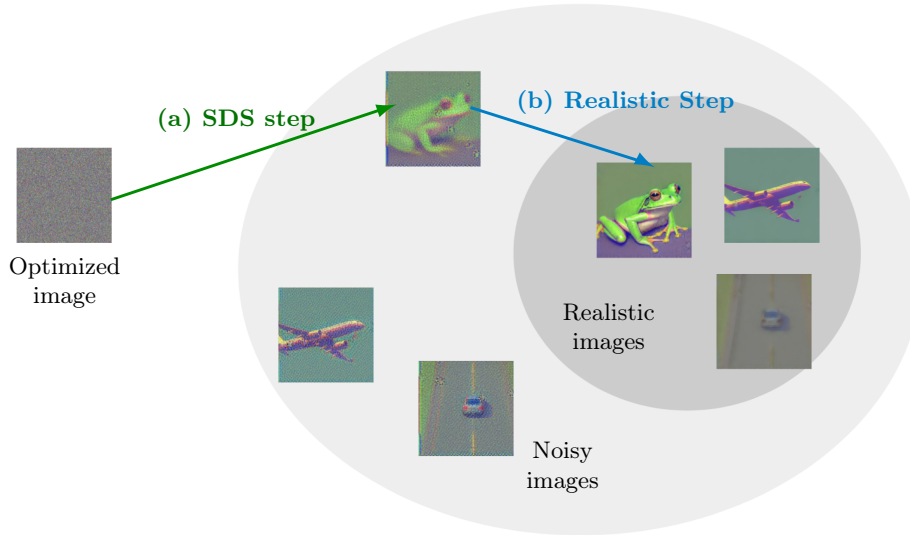


Fig. 6: Overview of PixSDS. The (a) step points towards noisy images, while (b) removes the noise and points towards clean images.

where $\beta \geq 0$ controls the size of the latent look-ahead step. The image $\hat{Z}$ can be interpreted as a clean, VAE-consistent approximation of the image that the latent SDS update is moving toward.

We then define the clean direction

$$g_{\text{clean}} = \hat{Z} - Z. \quad (5)$$

This direction pulls the optimized image toward the decoded latent update, thereby reducing motion in pixel-space directions that are not supported by the VAE latent representation.

To combine this direction with the original SDS update, we dynamically match its per-pixel magnitude to the magnitude of the SDS update. Let

$$\text{cnorm}(U)_{i,j} = \left(\sum_{k=1}^C U_{k,i,j}^2 \right)^{1/2}$$

denote the per-pixel channel norm, with the resulting $H \times W$ map broadcast over channels when multiplied with a $C \times H \times W$ tensor. The repaired update is

$$\tilde{g}_{\text{sds}} = g_{\text{sds}} + \frac{g_{\text{clean}}}{\text{cnorm}(g_{\text{clean}})} \odot \text{cnorm}(g_{\text{sds}}), \quad (6)$$

The normalization preserves the spatial scale of the original SDS update while replacing part of its noisy pixel-space motion with a cleaner VAE-consistent direction.

Finally, the image is updated using $\tilde{g}_{\text{sds}}$. Since PixSDS only modifies the update after the SDS direction has been computed, it does not require changing the diffusion model, the renderer, or the underlying SDS objective. This makes the method lightweight and compatible with other SDS-style objectives. Python-like pseudocode is provided in Fig. 5, and the update is illustrated in Fig. 6.

5 Experiments

We evaluate PixSDS in controlled 2D optimization and in text-to-3D generation pipelines. All experiments are run on a single NVIDIA V100 32GB GPU. The method is implemented in PyTorch using Hugging Face libraries [19].

5.1 2D Generation

We first evaluate PixSDS in a controlled 2D setting, where the optimized variable is an image rather than a 3D representation. This removes renderer-specific effects and allows us to directly measure whether the proposed gradient repair reduces pixel-space artifacts.

We evaluate two variants of our method, using either SGD or Adam to update the optimized image. In both cases, we use learning rate 0.05, `guidance_scale` = 25.0 in classifier-free guidance, and set $\beta = 0.1$. We initialize each image to

Method	FID ($\downarrow$)	CLIP Score ($\uparrow$)	BRISQUE ($\downarrow$)	CLIP-IQA Quality ($\uparrow$)	CLIP-IQA Noisiness ($\uparrow$)
SDS	431.877	16.442	82.120	0.859	0.031
SDS-Bridge	352.007	14.895	90.030	0.429	0.174
HiFA	304.815	16.311	41.764	0.246	0.025
NFSD	296.397	15.252	67.856	0.402	0.142
PGC	293.886	15.455	75.095	0.455	0.121
SDI	424.977	16.539	72.976	0.719	0.072
VSD	321.474	15.902	85.845	0.387	0.104
2-step-SDS	229.813	16.225	26.448	0.760	0.479
PixSDS+SGD	223.021	15.962	12.850	0.837	0.590
PixSDS+Adam	229.638	15.570	31.013	0.801	0.465
Stable Diffusion	190.851	16.345	12.020	0.933	0.655

Table 1: 2D Generation Results.

the constant value (0.5, 0.5, 0.5) and optimize it for $N = 1,000$ steps in fp16 precision. Unless otherwise stated, all latent SDS experiments use `stable-diffusion-2-base`. We use a linearly annealed timestep schedule,

$$t = 1000 \cdot \text{clip} \left(1 - \frac{\text{step}}{N}, 0.4, 1.0 \right),$$

so that timesteps decrease from 1000 to 400 during optimization. We found this schedule to reduce oversaturation; as discussed in the Appendix (Sec. E), the SDS image gradient norms become larger after timestep 400 for linear timestep annealing.

We randomly sample 100 captions from MS-COCO 2014 [9] and generate one image per caption for each method. We report standard image-generation metrics, including FID [5] and CLIP Score [4]. Since our main goal is artifact reduction, we also report no-reference image quality and noise-related metrics: BRISQUE [12], CLIP-IQA quality, and CLIP-IQA noisiness. As a direct-sampling reference, we also include images sampled directly from Stable Diffusion. This reference is not an SDS optimization method, but it indicates the quality of the underlying pretrained diffusion model.

Quantitative results are shown in Tab. 1. Among SDS-style optimization methods, PixSDS achieves the best FID, BRISQUE, and CLIP-IQA Noisiness scores, while remaining competitive in CLIP Score and CLIP-IQA quality. Qualitative results in Fig. 7 show the same trend. Vanilla SDS [14], SDS-Bridge [11], HiFA [20], NFSD [6], PGC [13], SDI, and VSD [18] often produce visible structured noise or color artifacts. 2-step-SDS [16] produces cleaner images than most baselines, but its results are often oversmoothed. In contrast, PixSDS reduces artifacts while preserving more local detail, such as the blender buttons in the first row and the chair structures in the fifth row. Although direct Stable Diffusion sampling remains stronger as an image generator, our objective is different: we aim to improve SDS-style optimization rather than replace direct diffusion sampling.



Fig. 7: 2D generation comparison. Each column shows a different method, and each row corresponds to the same text prompt. PixSDS reduces the structured artifacts that appear in several SDS-style optimization baselines while preserving semantic content.

5.2 3D Generation

We next test whether the same repair mechanism helps in text-to-3D optimization. We integrate PixSDS into the second stage of DreamGaussian [17], replacing the SDS update used when SDEdit is disabled. Following the original pipeline, we apply Gaussian smoothing with kernel size 11 to reduce high-frequency noise. For a fair comparison, we apply the same smoothing to the SDS baseline.

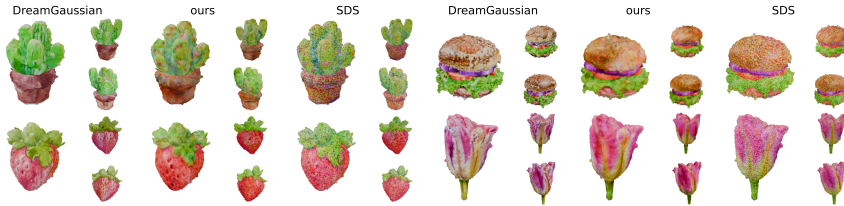


Fig. 8: DreamGaussian comparison. We apply PixSDS in the second optimization stage of DreamGaussian [17]. Compared with SDS, PixSDS produces cleaner textures and fewer structured artifacts.

As shown in Fig. 8, PixSDS produces cleaner results than the SDS baseline, with fewer noisy texture patterns. This experiment also supports the diagnosis from Sec. 3: structured artifacts can already be introduced during latent SDS optimization, before mesh extraction or mipmap sampling. Renderer and

texture-processing choices may amplify these artifacts, but they are not necessary for the failure mode to appear.

We also apply PixSDS to LucidDreamer [8], as shown in Fig. 9. In this setting, we run optimization for 3000 steps, sample timesteps from $[0.3, 0.8]$, and set $\beta = 100 \cdot \text{learning_rate}$ because latent-space gradients are small during optimization. We do not perform extensive hyperparameter tuning. Even so, PixSDS consistently reduces the noisy artifacts visible in the baseline. For example, the *football helmet* contains fewer noisy Gaussians inside the object, and the *hamburger* produces fewer floating artifacts around the asset. The *white hair ironman* result is also cleaner around the head, although the white-hair attribute is not fully preserved, suggesting that further tuning may be needed for some prompts.



Fig. 9: LucidDreamer comparison. We integrate PixSDS into LucidDreamer [8]. The repaired update reduces floating noisy Gaussians and structured texture artifacts.

5.3 Ablation Study

We ablate the key components of PixSDS in Fig. 10. **(a) Only g_{clean} :** we remove the original SDS direction and set $\tilde{g}_{\text{sds}} = g_{\text{clean}}$. The result is clean but poorly composed, with the object often appearing near the image boundary. This shows that g_{sds} is still necessary for semantic placement and generation. **(b) No channel normalization:** we remove the per-pixel normalization and set $\tilde{g}_{\text{sds}} = g_{\text{sds}} + g_{\text{clean}}$. This can produce clean 2D images, but fails in 3D because the magnitude of g_{sds} can dominate g_{clean} . The normalization is therefore important for balancing the noisy SDS direction with the clean VAE-consistent direction. **(c) $\beta = 0$:** we remove the latent look-ahead step and set $\hat{Z} = \text{dec}(\text{enc}(Z))$. This pulls the image toward its current VAE reconstruction, producing clean but unrealistic results. This shows that the clean direction should point toward the next latent SDS update, not merely toward the current decoded latent. **(d) Full method:** the full PixSDS update combines g_{sds} with the normalized clean direction obtained from the decoded latent look-ahead step. This preserves semantic guidance while reducing structured artifacts in both 2D and 3D optimization.

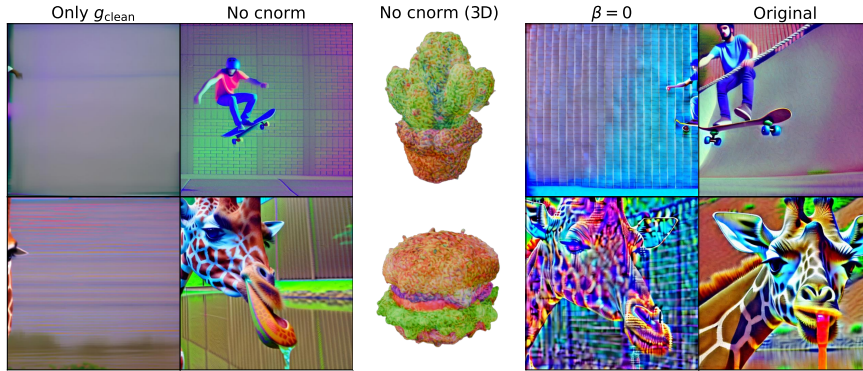


Fig. 10: Ablation study. We ablate the main design choices of PixSDS. The full method preserves the SDS update while adding a normalized VAE-consistent clean direction.

6 Conclusion

We studied a structured artifact failure mode in latent score distillation. Through controlled experiments, we showed that these artifacts are not fully explained by latent tensor resolution, pixel-space SDS optimization, or renderer-specific effects. Instead, they can arise from VAE-induced pixel drift: the optimized image moves along pixel-space directions that are weakly constrained by the VAE encoder, while its latent representation remains clean and semantically meaningful. Motivated by this diagnosis, we introduced PixSDS, a lightweight VAE-consistent gradient repair method. PixSDS decodes a latent SDS look-ahead step and uses it to construct a clean pixel-space direction, which is combined with the original SDS update using per-pixel normalization. Experiments in 2D optimization and text-to-3D pipelines show that PixSDS reduces structured artifacts while preserving semantic guidance, without retraining the diffusion model or changing the renderer. Our results suggest that the geometry of the VAE latent-to-pixel mapping is an important factor in latent SDS optimization, and should be considered when designing future score distillation methods.

Acknowledgements

The calculations have been performed using the facilities of the Scientific IT and Application Support Center of EPFL.

References

1. Camboulin, C., Doimo, D., Glielmo, A.: Understanding variational autoencoders with intrinsic dimension and information imbalance (2024), <https://arxiv.org/abs/2411.01978>

2. Chen, R., Chen, Y., Jiao, N., Jia, K.: Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 22246–22256 (2023)
3. Guisard, B.: Stable diffusion nano 2.1. <https://huggingface.co/bguisard/stable-diffusion-nano-2-1> (2023), hugging Face model card. Accessed: 2026-05-24
4. Hessel, J., Holtzman, A., Forbes, M., Le Bras, R., Choi, Y.: Clipscore: A reference-free evaluation metric for image captioning. In: Proceedings of the 2021 conference on empirical methods in natural language processing. pp. 7514–7528 (2021)
5. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems* **30** (2017)
6. Katzir, O., Patashnik, O., Cohen-Or, D., Lischinski, D.: Noise-free score distillation (2023), <https://arxiv.org/abs/2310.17590>
7. Krizhevsky, A., Nair, V., Hinton, G.: Learning multiple layers of features from tiny images. Tech. rep., University of Toronto (2009)
8. Liang, Y., Yang, X., Lin, J., Li, H., Xu, X., Chen, Y.: Luciddreamer: Towards high-fidelity text-to-3d generation via interval score matching. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 6517–6526 (2024)
9. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European conference on computer vision. pp. 740–755. Springer (2014)
10. Lukoianov, A., de Ocariz Borde, H.S., Greenewald, K., Guizilini, V.C., Bagautdinov, T., Sitzmann, V., Solomon, J.: Score distillation via reparametrized ddim. *Advances in Neural Information Processing Systems* **37**, 26011–26044 (2024)
11. McAllister, D., Ge, S., Huang, J.B., Jacobs, D.W., Efros, A.A., Holynski, A., Kanazawa, A.: Rethinking score distillation as a bridge between image distributions. *Advances in Neural Information Processing Systems* **37**, 33779–33804 (2024)
12. Mittal, A., Moorthy, A.K., Bovik, A.C.: No-reference image quality assessment in the spatial domain. *IEEE Transactions on Image Processing* **21**(12), 4695–4708 (2012). <https://doi.org/10.1109/TIP.2012.2214050>
13. Pan, Z., Lu, J., Zhu, X., Zhang, L.: Enhancing high-resolution 3d generation through pixel-wise gradient clipping (2024), <https://arxiv.org/abs/2310.12474>
14. Poole, B., Jain, A., Barron, J.T., Mildenhall, B.: Dreamfusion: Text-to-3d using 2d diffusion. arXiv preprint arXiv:2209.14988 (2022)
15. Pope, P., Zhu, C., Abdelkader, A., Goldblum, M., Goldstein, T.: The intrinsic dimension of images and its impact on learning (2021), <https://arxiv.org/abs/2104.08894>
16. Skorokhodov, V., Durasov, N., Fua, P.: Diffusion models are secretly zero-shot 3dgs harmonizers. arXiv preprint arXiv:2503.06740 (2025)
17. Tang, J., Ren, J., Zhou, H., Liu, Z., Zeng, G.: Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. In: International Conference on Learning Representations. vol. 2024, pp. 33879–33896 (2024)
18. Wang, Z., Lu, C., Wang, Y., Bao, F., Li, C., Su, H., Zhu, J.: Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *Advances in neural information processing systems* **36**, 8406–8441 (2023)
19. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al.: Huggingface’s transformers: State-of-the-art natural language processing. arXiv preprint arXiv:1910.03771 (2019)
20. Zhu, J., Zhuang, P., Koyejo, S.: Hifa: High-fidelity text-to-3d generation with advanced diffusion guidance (2024), <https://arxiv.org/abs/2305.18766>