A. Related Work

Scaling test-time compute. Scaling test-time compute is proven to be highly effective on pre-trained LLMs. This presents a completely different axis in LLM's scaling behaviors and inspires many investigations. Recent studies in test-time scaling of LLMs mainly focus on three aspects: (1) better search/planning algorithms [19, 70, 82, 84]; (2) better verifiers [11, 41, 42, 79]; and (3) scaling law of test-time compute [7, 65, 83]. These works highlight the importance of test-time compute and methods for effectively allocating these compute under a certain budget, orienting the community towards building agents with the ability to reason and self-correct. Inspired by these works, we study the scaling behavior of diffusion models at inference time, introduce a general search framework over injected noises during sampling, and demonstrate its effectiveness across different benchmarks, aiming to motivate more explorations of inference-time scaling in the diffusion model community.

Fine-tuning diffusion models. To align diffusion models with human preferences, multiple fine-tuning methods have been proposed. Fan and Lee [16] interpret the denoising process as a multi-step decision-making task and use policy gradient algorithms to fine-tune diffusion samplers. Black et al. [6], Fan et al. [17] formulate the fine-tuning task as an RL problem, and using policy gradient to maximize the feedback-trained reward. Clark et al. [10], Xu et al. [85] further simplifies this task by directly back-propogating the reward function gradient through the full sampling procedure. Wallace et al. [78] reformulate *Direct Preference Optimization* [55] to derive a differentiable preference objective that accounts for a diffusion model notion of likelihood, and Yang et al. [87] discard the explicit reward model and directly fine-tune the model on human preference data. Lastly, Domingo-Enrich et al. [13] casts fine-tuning problem as stochastic optimal control to better align with the tilted distribution of base and reward models. These studies represent substantial advancements in enforcing alignment in diffusion models, ensuring they adhere to human preferences, ethical considerations, and controlled behaviors.

Sample selection and optimization in diffusion models. The large variation in diffusion's sampling qualities leads to the natural question of how to find good samples during test-time. To address this, several works focus on sample selection guided by some pre-defined metrics using the Random Search algorithm. Karthik et al. [31] and Liu et al. [45] use pre-trained VQA and human preference models to guide the selection, and Liu et al. [45] further update the proposal distribution during selection to better align with the ground truth distribution. Similarly, Na et al. [47] performs rejection sampling on the updated proposal distribution during intermediate diffusion denoising step. On the other hand, Tang et al. [72] and Samuel et al. [62] use a small set of ground truth images as reference and use the similarity between reference and generated images as a guide for selection. Yet, these works primarily focus on addressing challenges using very specific verifier and algorithm, while largely overlooking a comprehensive investigation into the biases inherent in different verifiers, the interplay of multiple verifiers and search methods on different tasks, and the relationship between inference-time compute budget and scaling performance. Some other works [5, 15, 32, 48, 77] utilize the gradient of a pre-trained reward model to directly optimize for a better sample. We note, again, that these works focus on relatively small-scaled tasks (in-painting, editing, super-resolution), and the costs of these methods are prohibitive due to the need to back-propagate through the diffusion sampling process.

Recently, several studies[2, 92] have proposed approximating the distribution of "good" noises using neural networks. These approaches first identify preferable noises x'_T by transforming random noises $x_T \sim \mathcal{N}(0, \mathbf{I})$ through guided DDIM inversion. Subsequently, they train a lightweight predictor on the set of (x_T, x'_T) pairs for sampling preferable noises at inference time. Although these methods shift computational costs from test time to a one-time training phase, they require additional dataset curation and parameter tuning, and can have unsatisfying performance in some application scenarios.

B. Experiment Settings

We present our experimental settings below.

B.1. Training Settings

Most models used in our work are pre-trained: in ImageNet, we use the pre-trained SiT-XL model; under Text-to-Image setting, we use the publicly released weights of FLUX.1-dev and PixArt-Σ from the `diffusers` library [76]. In Section 5, the reported SiT-B and SiT-L are self-trained following the identical architectures and training configurations from [46]. The final numbers included in Figure 10 are from models trained at 800K iterations.

B.2. Sampling Settings

We summarize the sampling settings in our work below.

Configs	*Class-conditioned* SiT-XL	*Text-conditioned* FLUX.1-dev	PixArt-Σ
ODE solver	2nd order Heun	Euler	DDIM
search NFEs	50†	30	30
generation NFEs	250	30	30
guidance scale	1.0‡	3.5	4.5
resolution	256	1024	1024

Table 4. **Default sampling settings for Class-conditioned and Text-conditioned generation.** † In Figure 9 we report numbers with different NFEs/iter; ‡ In Figure 4 we report results with different guidance scales.

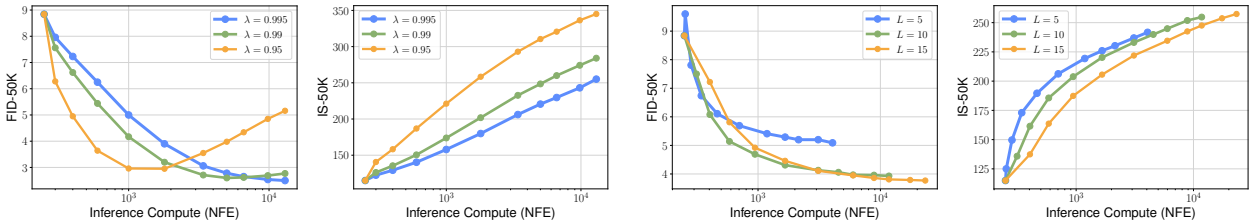

Figure 11. **Performance of tuning additional parameters for algorithms.Left**: Zero-Order Search with step sizes λ; **Right**: Search Over Paths with lengths L. We use SiT-XL and fix the verifier to be the classification logits from DINO.

B.3. Search Settings

Random Search. During search, we randomly sample a set of i.i.d Gaussian noises $|S|$ as the candidates for each conditioning, generate samples from them with the ODE solver, and select the one with the highest score output by the verifiers as the noise used for final generation. We take the size of S as the primary scaling axis and explore $|S| = 2^k$ for $k \in \{1, 2, 3, 4, 5, 6, 7, 8\}$ in our experiments.

Zero-Order Search. There are three tunable parameters in Zero-Order Search: search iterations K, number of neighbors N, and step size λ. As K is the most scalable, we fix it as the primary scaling dimension when studying the behavior of Zero-Order Search. In Figure 6 we additionally investigate the performance of Zero-Order Search when tuning N, as it provides a secondary axis in scaling compute. In Figure 11, we demonstrate the effect of tuning step size λ. We fix $N = 2$ and explore the performance of modifying the values of λ with respect to different values of K.

Expectedly, when λ is small, Zero-Order Search has slightly worse performance and lower compute efficiency; when λ is large, Zero-Order Search suffers from overfitting - the selected set of noises fits too close to the high scoring area of the verifier, leading to loss of diversity. As a result, while it has the best Inception Score among the three, its FID starts to increase once the compute is scaled over 10^3 NFEs. We provide further analysis of this observation in Section C.

Search Over Paths. We summarize the hyperparameters for Search Over Paths below

Hyperparameter		*Description*
initial paths	N	The number of paths to start the search with.
paths width	M	The number of noises to sample within each path.
search start	σ	The time to start search.
backward stepsize	Δb	The length of time interval to run ODE solver.
forward stepsize	Δf	The length of time interval to run noising process.
paths length	L	The NFEs devoted in each backward step.

Table 5. **Hyperparameters for Search Over Paths.**

For $N > 1$, we start search with N i.i.d samples Gaussian noises and obtain N noisy samples $\{x_\sigma^i\}$. For each x_σ, we then formulate its sampling paths and search over them. Once the search terminates, N different $\hat{x}_0$ are left, and we run them through the Best-of-N selection to obtain the best one.

In our experiments, we set M and N to be our primary and secondary scaling axis, respectively, as shown in Figure 6. We further explore the behavior of tuning the paths length in Figure 11, where we see that scaling up the paths length can

Verifier	Aesthetic	CLIPScore	ImageReward	LLM Grader
-	5.79	0.71	0.97	84.29
Aesthetic + Random	**6.38**	0.69	0.99	86.04
+ ZO-2	6.33	0.69	0.96	85.90
+ Paths-2	6.31	0.70	0.95	85.86
CLIPScore + Random	5.68	**0.82**	1.22	86.15
+ ZO-2	5.72	0.81	1.16	85.48
+ Paths-2	5.71	0.81	1.14	85.45
ImageReward + Random	5.81	0.74	**1.58**	87.09
+ ZO-2	5.79	0.73	1.50	86.22
+ Paths-2	5.76	0.74	1.49	86.33
Ensemble + Random	6.06	0.77	1.41	**88.18**
+ ZO-2	5.99	0.77	1.38	87.25
+ Paths-2	6.02	0.76	1.34	86.84

Table 6. ***Performance of search algorithms with different verifiers on DrawBench.*** The results are obtained from FLUX.1-dev evaluated on DrawBench. The first row denotes the performance without search where we fix denoising budget to be 30 NFEs, and for the rest we fix search budget to be 2880 NFEs.

be beneficial to FID but have marginal effect on Inception Score. This supports our claim that the search settings need to be specifically tuned to different application scenarios. We fix other hyperparameters: $\sigma = 0.11$, $\Delta b = 0.81$, $\Delta f = 0.78$, inspired by the setting in [86]. Table 6 shows the performances of the three presented search algorithms on DrawBench.

B.4. Verifier Settings

ImageNet. We consider a total of four verifiers for search on ImageNet. We list the settings below:

- **FID**: we denote the ground truth feature statistics for ImageNet training set μ_{ref} and Σ_{ref}. During search, we select the first 1024 samples randomly and use them to initialize a running mean and covariance $\hat{\mu}$ and $\hat{\Sigma}$. In the following search iterations, for each candidate batch b_i a staged mean $\hat{\mu}_i$ and covariance $\hat{\Sigma}_i$ are calculated with the batch information and the previous running mean and variance. A corresponding FID_i will be obtained between $\hat{\mu}_i$, $\hat{\Sigma}_i$ and μ_{ref}, Σ_{ref}, which is then used as the verifier score. Eventually, $b = \arg\min_i \text{FID}_i$ will be selected, and μ_{ref} and Σ_{ref} are then updated accordingly. Such iteration is repeated until we reach 50000 samples.
- **IS**: The class confidence probability output by the InceptionV3 model is taken as the verifier score.
- **CLIP**: For logits, we use the zero-shot classifier weight W generated by prompt engineering specified in Radford et al. [54] and take the cosine-similarity between the corresponding class entry in W and the image feature to be the verifier score. For the self-supervised version, we directly extract the image feature.
- **DINO**: For logits, we use the pre-trained linear classification head provided in Oquab et al. [49] as the verifier. Specifically, we concatenate the `cls` tokens from the last four layers along with the average pooling of the feature from the last layer to formulate the input for the linear head. For the self-supervised version, we directly take the `cls` token from the last layer.

Text-to-Image. We consider a total of four verifiers for search in Text-to-Image setting:

- **Aesthetic**: we take the aesthetic predictor pre-trained on subset of LAION-5B. It consists of a single MLP without any non-linearity and takes the image feature from a pre-trained CLIP-L model as input. The output is on a scale of $0 - 10$ rating the images' aesthetic quality.
- **CLIPScore**: we take the pre-trained CLIP-L model and measure the cosine similarity between visual and text features. Following [22], each text prompt is additionally prefixed with `'A photo depicts'`, and the final score is rescaled by `2.5 * max(cos_sim, 0)`.
- **ImageReward**: we take the pre-trained model for approximating human preference from [85] and use the identical evaluation setting.
- **Verifier Ensemble**: We separately run candidates through the above three verifiers, rank the scores output by each, and use the unweighted average rankings as the final score for the Verifier Ensemble.

B.5. Evaluation Setting

ImageNet. Following standard practice, we calculate FID and Inception Score using 50000 synthesized samples. We use randomly generated conditions and a global batch size of 256 for all evaluations. We extracted the ImageNet statistics and calculated FID and IS following Karras et al. [30].

Model	Accuracy↑	Originality↑	Visual↑	Consistency↑	Emotional↑	Overall↑
FLUX.1-dev	89.35	67.58	93.00	97.04	73.99	84.29
+ 4 search iters	91.33	68.49	93.42	96.99	75.31	85.17
+ 16 search iters	91.95	71.52	**93.76**	**97.24**	76.30	86.42
+ 64 search iters	**93.83**	**75.38**	93.57	97.04	**79.34**	**88.08**
PixArt-Σ	84.60	73.29	91.91	95.80	76.34	84.67
+ 4 search iters	87.88	74.03	91.92	96.29	77.32	85.62
+ 16 search iters	88.15	75.39	91.72	96.04	79.17	86.27
+ 64 search iters	**89.30**	**77.79**	**92.46**	**96.68**	**80.43**	**87.55**

Table 7. **Break-down scores of LLM Grader for FLUX.1-dev and PixArt-Σ.** The evaluation is done on DrawBench with random search and verifier ensemble.

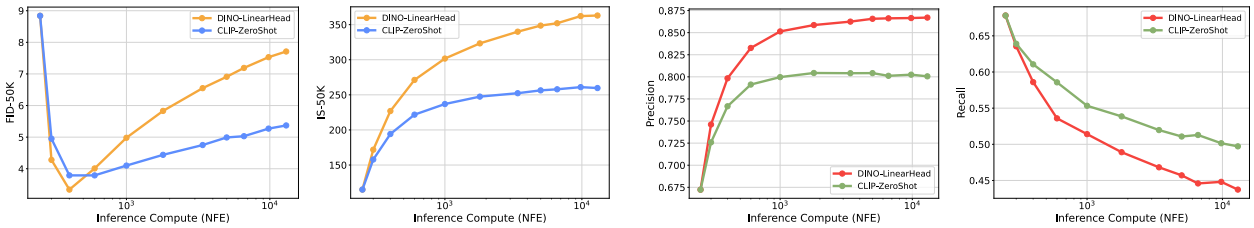

Figure 12. **Performance of Random Search on ImageNet against DINO and CLIP classification logits.** We use random search on the SiT-XL model and report FID, IS, Precision, and Recall.

DrawBench. We search for one noise per prompt for generating the sample. For evaluators other than the *LLM Grader*, we simply input the synthesized samples into the pre-trained evaluator models and report the averaged scores over the 200 prompts.

For *LLM Grader*, we prompt the Gemini-1.5 flash model to assess synthesized images from five different perspectives: Accuracy to Prompt, Originality, Visual Quality, Internal Consistency, and Emotional Resonance. Each perspective is rated from 0 to 100, and the averaged overall score is used as the final metric. We include the break-down scores in Table 7, and in Figure 15 we present the detailed prompt. We observe that search can be beneficial to each scoring category of the *LLM Grader*.

T2I-CompBench. For each prompt we search for two noises and generate two samples. During evaluation, the samples are splitted into six categories: `color`, `shape`, `texture`, `spatial`, `numeracy`, and `complex`. Following Huang et al. [27], we use the BLIP-VQA model [40] for evaluation in `color`, `shape`, and `texture`, the UniDet model [91] for `spatial` and `numeracy`, and a weighted averaged scores from BLIP VQA, UniDet, and CLIP for evaluating the `complex` category.

C. Verifier Hacking Leads to Degeneracy in Evaluation Metrics

Many prior works [6, 20, 34] noticed the overoptimization issue when finetuning diffusion models using pre-trained reward models, that excessively optimizing against a reward model will lead to degeneracy in other evaluation metrics. We have similar observations when we excessively search against a verifier and quickly overfit to its bias.

When search on ImageNet against the DINO or CLIP classification logits, we notice the sudden increasing in FID score once pass a certain search iteration numbers despite the constantly improving Inception Score, as shown in Figure 12. To investigate this issue, we calculate the Precision and Recall [37] and plot them in Figure 12. We see that while Precision increases with search iterations, demonstrating the consistent improvement in sample quality, Recall decreases with search iterations, implying the loss of diversity of the sample set.

We credit this to the DINO and CLIP classification verifiers. When searching against these verifiers, we only operate on a per-noise basis - select the one noise whose corresponding sample has the highest classification score. Therefore, when the search iterations increase, our final set of the selected noises will get closer to the high scoring regions of the classification verifiers. This have two consequences: 1) the selected noises overfit to the verifiers and degenerate other metrics; 2) the

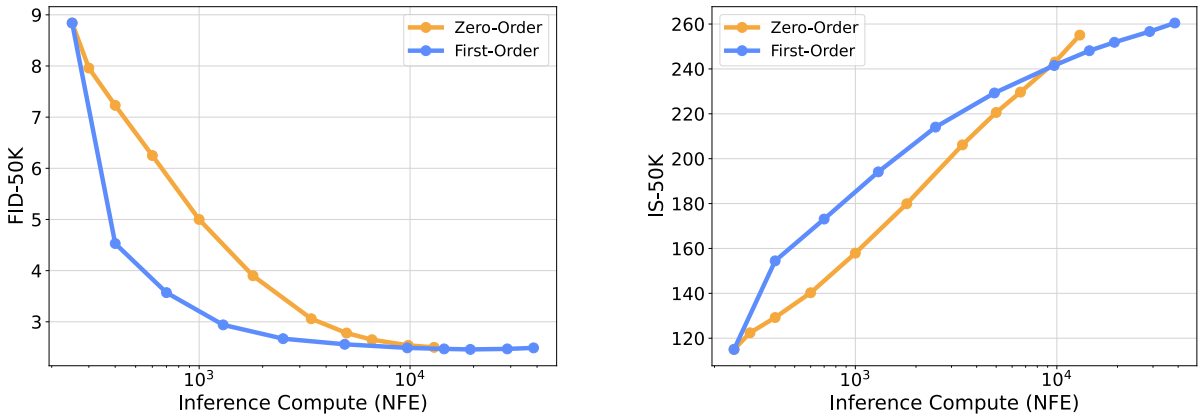

Figure 13. **Comparison between Zero-Order and First-Order Search.** We use the SiT-XL model and fix the verifier to be the DINO-LinearHead. The Inference Compute is aligned via the rough estimation of cost(backward) $\sim 3\times$cost(forward).

selected noises cluster around the high scoring regions and disregard the overall variance of the final set. We deem the latter to be more impactful on the evaluated FID, since FID is known to take great account of the diversity of the generated samples. The Zero-Order Search and Search over Paths we proposed in Section 3.2 alleviate this issue to some extent by searching in the local neighborhood of the Gaussian noise $\mathbf{n}$ sampled at the beginning or at the intermediate sampling steps. However, if we expand the neighborhood range for Zero-Order Search as shown in Figure 11, it will suffer from the diversity issue as well.

A more fundamental solution would be to use the verifiers operating on a population basis and taking into account of the global structure of the set of selected noises. From the trivial example in Figure 3, we see that such verifiers could be effective. We leave further exploration to future works.

Consequence (1) is better demonstrated in Figure 8. We see that over-search against Aesthetic Score will lead to degeneracy in CLIPScore, and vice versa.

D. Zero-Order and First-Order Search

Since many verifiers are differentiable, we also investigate First-Order Search on ImageNet guided by the gradient of verifiers. Specifically:

1. we initialize the noise prior with a randomly sampled Gaussian vector $\mathbf{n}$
2. run $\mathbf{n}$ through the diffusion ODE solver to obtain the sample and its corresponding score output by verifier $\mathcal{V}$
3. backpropogate through the verifier and the ODE solver to calculate $\nabla_{\mathbf{n}}\mathcal{V}(\mathbf{n})$
4. update $\mathbf{n}$ via gradient descent: $\mathbf{n}' = \mathbf{n} - \eta\nabla_{\mathbf{n}}\mathcal{V}(\mathbf{n})$, and repeat step 2-4.

Due to the iterative nature of diffusion sampling process, step 2 will incur prohibitive memory cost if naively backpropogating through the ODE solver. To alleviate this issue, we perform gradient checkpointing [9] on each ODE integration step following [5, 48, 77]. This discards the intermediate activation values and re-calculate them using one extra model forward call during backpropogation, thus greatly reducing space complexity at the cost of slightly increased execution time.

We also note that performing naive gradient descent in step 4 might push the updated $\mathbf{n}'$ outside the Gaussian manifold, resulting in training and sampling inconsistency. To resolve this, we simply rescale $\mathbf{n}'$ so that its norm is consistent with the norm of i.i.d Gaussian vectors[5].

In Figure 13 we include the comparison between Zero-Order Search and First-Order Search. We fix the learning rate to be $\eta = 0.01$ for First-Order Search to roughly match the step size of Zero-Order Search with $\lambda = 0.995$. At a best estimation we attribute the overhead of gradient checkpointing as twice the number of model forward calls, making each iteration $3\times$ costly than without backpropogation.

With inference compute roughly aligned, although First-Order Search shows faster convergence speed over Zero-Order, we see that it does not demonstrate a significant margin when continuously scaling up compute, despite its higher memory cost[6] and worse scalability on large models. However, by its gradient-guided nature, First-Order Search can be advantageous in tasks with more fine-grained objectives, such as image editing, inpainting, and solving inverse problems [5, 32, 48, 77].

[5] In $\mathbb{R}^d$, the norm of isotropic Gaussian vectors is distributed according to the chi-squared distribution with d degrees of freedom.

[6] Gradient checkpointing still requires $O(\sqrt{n})$ space complexity [9], with n being the number of layers inside the model.

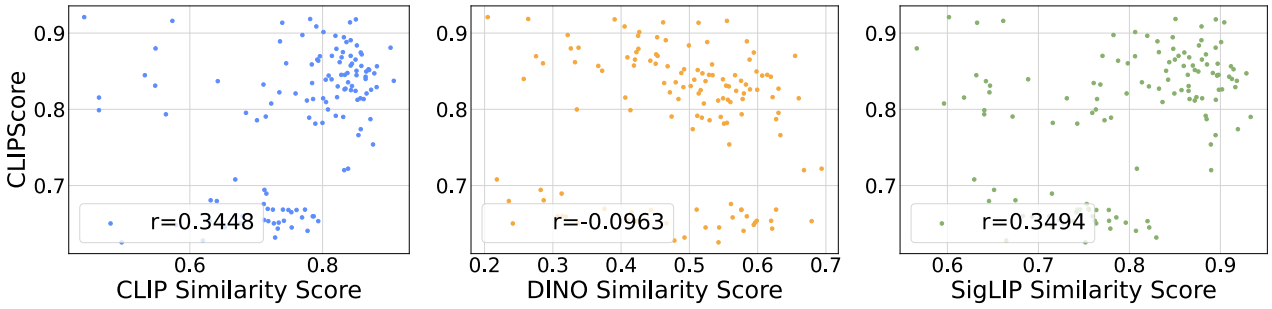

Figure 14. **From Left to Right**: Correlation of CLIP, DINO, and SigLIP feature similarity score with CLIPScore. All points are generated from FLUX.1-dev.

Verifiers	Aesthetic	CLIPScore	ImageReward
-	5.79	0.71	0.97
CLIP-SSL + 4 search iters	5.76	0.71	0.99
+ 16 search iters	5.72	0.71	1.04
DINO-SSL + 4 search iters	5.79	0.71	0.99
+ 16 search iters	5.78	0.70	1.03
SigLIP-SSL + 4 search iters	5.79	0.70	1.02
+ 16 search iters	5.75	0.70	1.02

Table 8. ***Performance of self-supervised verifiers on DrawBench.*** All numbers are from FLUX.1-dev with random search. The first row is the reference performance without search.

E. Self-Supervised Verifiers have Marginal Effect in Text-to-Image Setting

Following Section 3.1, we investigate the performance of self-supervised verifiers in text-to-image setting. Apart from DINO and CLIP, we additionally incorporate SigLIP [90] as an extension to CLIP. Different from ImageNet where the self-supervised verifiers are good surrogate for classification verifiers, on DrawBench they do not demonstrate the expected performance, as shown in Table 8.

From Figure 14, we observe much weaker metric correlations comparing with self-supervised verifiers in ImageNet. We credit this observation to the following:

1) evaluation metrics in text-to-image settings usually focus on more nuanced perspectives: visual-text alignment, composition correctness, human preferences, etc. Even the aesthetic predictor has its bias - it prefers stylized images over others [10]. On the other hand, self-supervised verifiers essentially select the samples with smallest trajectory curvature in the feature space, which implies a stabler sampling process and thus potentially higher sample quality. Yet, by the subtle and holistic nature of evaluation in text-to-image settings [39], such "higher sample quality" may not align with the specific perspectives each metric focuses on. For example, under the same text prompt, an image with high visual quality but misaligned content might not be preferred over an image with slightly degraded visual quality but richer and more aligned visual content.

2) The rich conditionings and extensive fine-tuning in text-to-image models on large scale datasets might lead to different sampling dynamics comparing to the small class-conditioned models trained on ImageNet. This may lead to failure of the self-supervised verifier themselves, as the low trajectory curvature measured in feature space might no longer be indicative of the sample quality.

This also calls for the design of task specific verifiers. From the self-supervised verifiers across class-conditioned and text-conditioned tasks, we see that the effectiveness of verifiers can be highly task-dependent. Therefore, to conduct search that's better aligned with desired objectives, we deem it necessary to have verifiers designed specifically for each task; during search, it's also very important to avoid hacking to the specific bias of each verifier. We have proposed some simple methods in our work, and we leave further explorations of this problem to future works.

"You are a multimodal large-language model tasked with evaluating images generated by a text-to-image model. Your goal is to assess each generated image based on specific aspects and provide a detailed critique, along with a scoring system. The final output should be formatted as a JSON object containing individual scores for each aspect and an overall score. Below is a comprehensive guide to follow in your evaluation process:

1. Key Evaluation Aspects and Scoring Criteria:

For each aspect, provide a score from 0 to 10, where 0 represents poor performance and 10 represents excellent performance. For each score, include a short explanation or justification (1-2 sentences) explaining why that score was given. The aspects to evaluate are as follows:

a) Accuracy to Prompt

Assess how well the image matches the description given in the prompt. Consider whether all requested elements are present and if the scene, objects, and setting align accurately with the text. Score: 0 (no alignment) to 10 (perfect match to prompt).

b) Creativity and Originality

Evaluate the uniqueness and creativity of the generated image. Does the model present an imaginative or aesthetically engaging interpretation of the prompt? Is there any evidence of creativity beyond a literal interpretation? Score: 0 (lacks creativity) to 10 (highly creative and original).

c) Visual Quality and Realism

Assess the overall visual quality, including resolution, detail, and realism. Look for coherence in lighting, shading, and perspective. Even if the image is stylized or abstract, judge whether the visual elements are well-rendered and visually appealing. Score: 0 (poor quality) to 10 (high-quality and realistic).

d) Consistency and Cohesion

Check for internal consistency within the image. Are all elements cohesive and aligned with the prompt? For instance, does the perspective make sense, and do objects fit naturally within the scene without visual anomalies? Score: 0 (inconsistent) to 10 (fully cohesive and consistent).

e) Emotional or Thematic Resonance

Evaluate how well the image evokes the intended emotional or thematic tone of the prompt. For example, if the prompt is meant to be serene, does the image convey calmness? If it's adventurous, does it evoke excitement? Score: 0 (no resonance) to 10 (strong resonance with the prompt's theme).

2. Overall Score

After scoring each aspect individually, provide an overall score, representing the model's general performance on this image. This should be a weighted average based on the importance of each aspect to the prompt or an average of all aspects.

Figure 15. The detailed prompt for evaluation with the LLM Grader.

F. More Visualizations on Scaling Behavior

F.1. SiT-XL

The images presented in this section are sampled from the pre-trained SiT-XL in 256 resolution, using 2^{nd} order Heun sampler and guidance scale of 4.0. Each row of images is structured as follows:

- **Left three**: sampled with increasing steps: 10, 20, 250.
- **Right three**: sampled with Zero-Order Search and the DINO classification verifier. We set $N = 2$ and $\lambda = 0.95$ for Zero-Order Search, and the equivalent NFEs invested are 450, 1850, 6650.

Figure 16. **"loggerhead turtle" (33)**

Figure 17. **"Sulphur-crested cockatoo" (89)**

Figure 18. **"Siberian husky" (250)**

Figure 19. **"Arctic wolf" (270)**

Figure 20. **“baseball” (429)**

Figure 21. **“hammer” (587)**

Figure 22. **“volcano” (980)**

F.2. FLUX.1-dev

The images presented in this section are sampled from the pre-trained FLUX.1-dev in 1024 resolution, using Euler sampler and guidance scale of 3.5. Each row of images is structured as follows:

- **Left three**: sampled with increasing steps: 4, 16, 30.
- **Right three**: sampled with Zero-Order Search and the Verifier Ensemble. We set $N = 2$ and $\lambda = 0.95$ for Zero-Order Search, and the equivalent NFEs invested are 120, 960, 2880.

Figure 23. **"New York Skyline with 'Diffusion' written with fireworks on the sky."**

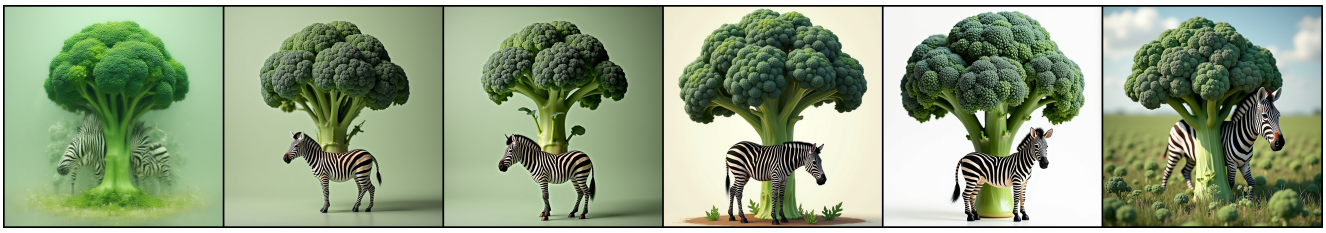

Figure 24. **"A zebra underneath a broccoli."**

Figure 25. **"A car on the left of a bus."**

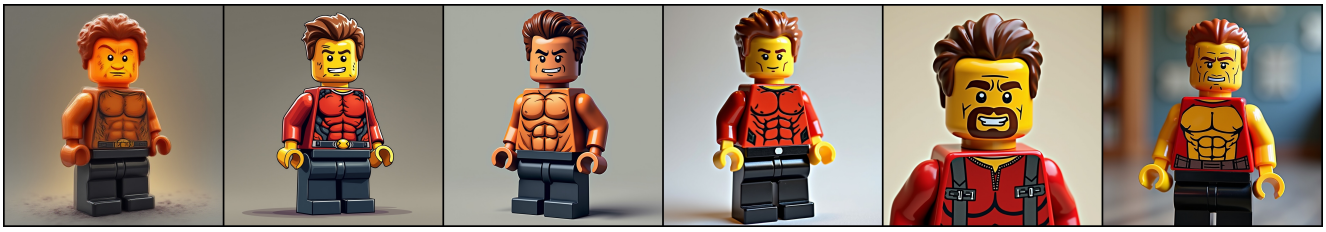

Figure 26. **"Lego Arnold Schwarzenegger."**

Figure 27. **“An ancient Egyptian painting depicting an argument over whose turn it is to take out the trash.”**

Figure 28. **“A storefront with ‘Deep Learning’ written on it.”**

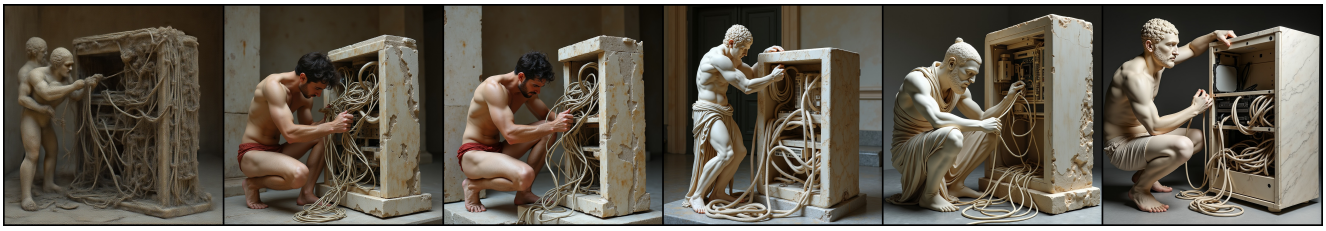

Figure 29. **“An IT-guy trying to fix hardware of a PC tower is being tangled by the PC cables like Laokoon. Marble, copy after Hellenistic original from ca. 200 BC. Found in the Baths of Trajan, 1506.”**

F.3. PixArt-Σ

The images presented in this section are sampled from the pre-trained PixArt-Σ in 1024 resolution, using DDIM sampler and guidance scale of 4.5. Each row of images is structured as follows:

- **Left three**: sampled with increasing steps: 4, 8, 28.
- **Right three**: sampled with Zero-Order Search and the Verifier Ensemble. We set $N = 2$ and $\lambda = 0.95$ for Zero-Order Search, and the equivalent NFEs invested are 112, 896, 2688.

Figure 30. **"An oil painting portrait of the regal Burger King posing with a Whopper."**

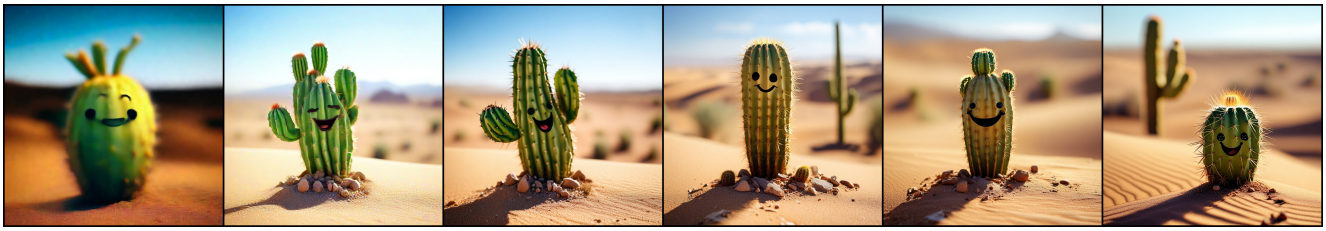

Figure 31. **"A small cactus with a happy face in the Sahara desert."**

Figure 32. **"Greek statue of a man tripping over a cat."**