

A. Overview of Appendices and Supplemental Material

- Appendix A: This summary.
- Appendix B: Extended commentary on related work Sec. 2.1 and Sec. 2.2, with insights and details on our implementation of evolutionary algorithms for alignment.
- Appendix C: Summary of insights we encountered when deciding how to initialize genetic algorithms and evolutionary strategies, related to Sec. 3.2. We include configuration details of compared methods such as SVDD, DSearch, and FkD for fair comparison.
- Appendix D: Extended quantitative and qualitative results across various models, reward functions, datasets, and algorithms used in our work, complementary to Sec. 4.2—Sec. 4.3.

B. Extended Commentary on Related Work and Implementations

Here, we provide additional detail for concepts in our work that was not able to be fit into the main text. We include a small table of qualitative differences between alignment methods (Tab. 5), which complements Sec. 2.1.

B.1. Extended Diffusion Model Alignment Related Work

Diffusion models use a reverse diffusion process to convert some latent noise distribution into a data distribution, such as images [16, 41]. The reverse diffusion process iteratively denoises the initial latent noise z_T over some number of steps ($t = T \rightarrow t = 0$) to yield a sample, z_0 .

Though diffusion models are capable of modeling complex data distributions, they often fail to produce samples that meet some downstream objective. *Diffusion model alignment methods* adjust diffusion models such that the resulting samples better meet an objective beyond the model’s original maximum likelihood criterion. These objectives commonly focus on producing images that reflect human aesthetic preferences [17, 51–53], or other metrics such as compressibility [3].

Alignment methods can generally be grouped into two main categories: fine-tuning-based methods and inference-time methods. Fine-tuning-based alignment methods involve adjusting the diffusion model’s parameters so that generated samples better match alignment objectives. Generally, these methods rely on curated datasets [37] or reward models [51–53] to fine-tune diffusion models [5, 6, 21, 32], and can involve supervised fine-tuning, or reinforcement learning

Table 5. Qualities of diffusion alignment methods.

Method	Arbitrary Rewards?	Gradient Free?	Black Box?
DNO [43]	✓	✗	✗
Diffusion-DPO [48]	✓	✗	✗
DSearch-R [24]	✓	✓	✗
SVDD [24]	✓	✓	✗
FKS [40]	✓	✓	✗
Zero-Order [27]	✓	✓	✓
Best-of-N	✓	✓	✓
Ours	✓	✓	✓

based methods [3, 48]. Fine-tuning based methods, although powerful, require retraining and thus all the associated costs — our work does not require training.

Our work follows the alternative approach, **inference-time methods**. Rather than altering model parameters, these techniques adjust sampling or conditioning to ensure samples meet alignment objectives. Formally, they seek the control variable ψ that maximizes the expected reward $R(x)$ of samples x from a pretrained diffusion model p_θ , as shown in Eq. (1).

$$\psi^* = \arg \max_{\psi \in \Psi} \mathbb{E}_{x \sim p_\theta(x|\psi)} [R(x)]. \quad (3)$$

Examples of control variables, ψ , include optimizing conditional input prompts [10, 12, 15, 28], manipulating cross-attention layers [11], and tuning latent noise vectors (noise optimization) to guide the diffusion trajectory [24, 27, 43, 46, 47, 55]. In this work, we focus exclusively on noise optimization ($\psi = z$), which is generalizable across diffusion models. Broadly, noise optimization methods fall into *gradient-based* optimization, or *gradient-free* optimization. We discuss each approach in turn.

Gradient-based methods refine the noise iteratively by leveraging the gradient with respect to the reward. Direct Optimization of Diffusion Latents, DOODL [47], and Direct Noise Optimization, DNO [43], are exemplary of these approaches. Both need to contend with the computational costs of backpropagation and maintaining sample quality by keeping the optimized noise on the Gaussian shell. DOODL optimizes the diffusion noise by computing the gradients with respect to a differentiable loss on generated images. They keep memory costs constant by leveraging invertible networks, and ensure sample quality by normalizing the optimized noise to have the norm of the original. Likewise, DNO computes the gradient through the reward function to optimize the noise, but can optimize *non-differentiable* rewards by using zero-th order optimization algorithms. The drawbacks of gradient-based methods are: they are not black-box; they are computationally expensive, especially when

aligning multiple samples; and often require long optimization budgets (*e.g.* DNO requires > 100 optimization steps, whereas our method achieves better results in ~ 50 steps).

Gradient-free methods, on the other hand, explore the space of noise vectors or trajectories using only search or sampling methods. Ma *et al.* [27], employ three different strategies — random search, zero-order search (similar to hill climbing), and search-over-paths — to find samples that maximize their reward functions. Uehara *et al.* [46] provide a comprehensive overview of various inference-time algorithms, covering sequential Monte Carlo (SMC)-based guidance, value-based importance sampling, tree search, and classifier guidance. Li *et al.* propose DSearch [24], a dynamic beam search algorithm in order to search for noise in order to maximize some reward function. These techniques vary in their trade-offs, for example DSearch and the techniques outlined by Uehara *et al.* are not black-box and may have large computational costs. Meanwhile, although random and zero-order search are black-box and computationally efficient we show evolutionary methods outperform them (Sec. 4.2).

In this work, we investigate a different class of gradient-free optimization methods: evolutionary algorithms. We pursue evolutionary algorithms due to their: (1) ability to be used for black-box optimization; (2) overcoming certain computational and hardware limitations associated with existing gradient-based and complex search-based methods; (3) potential to effectively explore multi-modal search spaces. As such, despite being a black-box method, we outperform both classes of methods, while remaining computationally efficient.

B.2. Extended Alignment Objective Commentary

Here, we make additional comments upon Eq. (2) as presented in Sec. 3. Recall that Eq. (2):

$$\phi^* = \arg \max_{\phi} \underbrace{\mathbb{E}_{\psi \sim q_{\phi}(\psi)}}_{\text{search distribution}} \left[\mathbb{E}_{x \sim p_{\theta}(x|\psi)} [R(x)] \right],$$

cast inference-time alignment as search problem where we find the search distribution $q_{\phi}(\psi)$, that maximizes the expected reward. This objective is a generalization of Eq. (1) to natural evolutionary strategies [50]. Namely, we rather than searching for a single ψ that maximizes the expected reward, we assume that ψ is sampled from a parameterized search distribution, q_{ϕ} , and we maximize the expected reward under this distribution. Note: x is either sampled from the solution space ψ in a stochastic manner or computed deterministically. In this context, θ is the parameterization of the sampler of the diffusion model, and the sampler is modeled either as $x \sim p_{\theta}(x|\psi)$ (stochastic) or $x = f_{\theta}(\psi)$ (deterministic).

B.3. Extended Characterization of Evolutionary Algorithm Components

Here, we provide an extended and in-depth overview of the mechanisms of genetic algorithms and natural evolutionary strategies.

Genetic Algorithms. Genetic algorithms maintain a population of candidate solutions, ψ_i , and iteratively improve them via selection, crossover, and mutation [8]. In the context of Eq. (2), the population is viewed as an empirical distribution. Thus the objective is to maximize the expected reward of a population of solutions. This objective is maximized iteratively and at each generation:

1. **Evaluation:** For each solution ψ_i in the population, the fitness (reward) is evaluated — *i.e.* $R(x_i)$ where $x_i \sim p_{\theta}(x|\psi_i)$.
2. **Selection:** Parent vectors, ψ_i are chosen based on fitness, *e.g.* via tournament or fitness-proportionate selection. In the context of alignment, selection will result in choosing better aligned solutions to use as parents for the next generation of solutions.
3. **Crossover:** Pairs of parents exchange noise component (or transformation parameters) to produce offspring. Correct choice of crossover can help ensure we meet (C1). For example, when using *uniform crossover* (swapping each coordinate independently), each child’s coordinate is drawn from one of two i.i.d. $\mathcal{N}(0, 1)$ parents thus remaining with high-density shell (Appendix B.5).
4. **Mutation:** Offspring are perturbed with noise, often in an additive manner. *E.g.* $x = x + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma)$. σ is a hyperparameter of the genetic algorithm.

Natural Evolutionary Strategies. Natural evolutionary strategies perform black-box optimization by adapting the parameterized search distribution q_{ϕ} over solutions [50]. Unlike GAs, q_{ϕ} is a parameterized search distribution (*e.g.* multivariate normal) and at each iteration its parameters are updated as follows:

1. **Sampling:** A population of solutions is sampled for the search distribution $\psi_i \sim q_{\phi}(\psi)$.
2. **Evaluation:** For each solution ψ_i in the population, we evaluate the fitness as $R(x_i)$, where $x_i \sim p_{\theta}(x|\psi_i)$.
3. **Update:** Estimate the gradient $\nabla_{\phi} \mathbb{E}_{\psi \sim q_{\phi}(\psi)} [\mathbb{E}_{x \sim p_{\theta}(x|\psi)} [R(x)]]$ and update ϕ to increase expected reward along the natural gradient. Thus, we update the parameters of the search distribution q_{ϕ} so that future samples are better aligned.

B.4. Noise Transform Search Implementation

Earlier we introduced the algorithm for direct noise search (Algorithm 1). We noted that the algorithm for noise noise

transform search is slightly different: we include it here for clarity.

Algorithm 2 Alignment via Noise Transformation Search

Require: Pretrained diffusion model p_θ , reward $R(\cdot)$, noise $z_T \sim \mathcal{N}(0, I)$, iterations T , population size M

- 1: **Search variable:** $\psi = A$, with transformed noise $z'_T = Q(A) z_T$
where $Q(A)$ denotes the orthonormal component of A obtained via QR decomposition
- 2: **Search distribution** $q_\phi(\psi)$: GA: population $\{A_i\}_{i=1}^M$ or (ES) parameterized A
- 3: **for** $t = 1, \dots, T$ **do**
- 4: Sample transformation parameters $\{A_i \sim q_\phi\}_{i=1}^M$
- 5: Compute transformed noises $\{z'_i = Q(A_i) z_T\}_{i=1}^M$
- 6: Generate samples $\{x_i\}_{i=1}^M$ with $x_i = f_\theta(z'_i)$
- 7: Compute rewards $\{r_i\}_{i=1}^M$ with $r_i = R(x_i)$
- 8: $\phi \leftarrow \text{EAUpdate}(\phi, \{A_i\}_{i=1}^M, \{r_i\}_{i=1}^M)$
- 9: **end for**
- 10: **return** final q_ϕ or best transform

NB: EAUpdate depends on the specific evolutionary algorithm used. A bias term b may be included, but we set it to 0.

B.5. Proof of Gaussian Marginal Preservation under Uniform Crossover

Purpose In this section we show that uniform crossover maintains the solutions within the high-density Gaussian shell, which was alluded to in Sec. 3.2. This is relevant to the EvoTorch implementation of the CoSyne algorithm, which uses a uniform distribution to decide crossover behavior (parameterized by a probability p to perform crossover).

Background. Samples from a high-dimensional Gaussian lie on a hypersphere, this phenomenon is known as the Gaussian Annulus Theorem. Diffusion models that use Gaussian noise are (implicitly) trained under this condition [1]. Thus, to ensure valid generated images, we must enforce this condition (draw samples near the surface of the sphere or shell) when we perturb noise. Uniform crossover is one such perturbation, which we now investigate.

Lemma. Let $X = (X_1, \dots, X_d)$ and $Y = (Y_1, \dots, Y_d)$ be independent draws from $\mathcal{N}(0, I_d)$. To perform coordinate-wise uniform crossover, for each i we independently sample:

$$B_i \sim \text{Bernoulli}(p), \quad Z_i = \begin{cases} X_i, & B_i = 1, \\ Y_i, & B_i = 0. \end{cases}$$

Then each coordinate $Z_i \sim \mathcal{N}(0, 1)$ and thus $Z \sim \mathcal{N}(0, I_d)$.

Fix a coordinate i . For any real z , by the law of total

probability conditioning on B_i :

$$\begin{aligned} P(Z_i \leq z) &= P(Z_i \leq z \mid B_i = 1) P(B_i = 1) \\ &\quad + P(Z_i \leq z \mid B_i = 0) P(B_i = 0) \\ &= P(B_i = 1) P(X_i \leq z) \\ &\quad + P(B_i = 0) P(Y_i \leq z) \\ &= p \Phi(z) + (1 - p) \Phi(z) \\ &= \Phi(z) \end{aligned}$$

where Φ is the standard normal CDF and we used $X_i, Y_i \sim \mathcal{N}(0, 1)$. Thus the CDF of Z_i matches that of $\mathcal{N}(0, 1)$, so $Z_i \sim \mathcal{N}(0, 1)$. Since each B_i acts independently on its coordinate and all coordinates of X, Y are independent, the Z_i remain independent. Therefore the joint distribution of Z is

$$P(Z_1 \leq z_1, \dots, Z_d \leq z_d) = \prod_{i=1}^d P(Z_i \leq z_i) = \prod_{i=1}^d \Phi(z_i),$$

which is the CDF of $\mathcal{N}(0, I_d)$. Equivalently, $Z \sim \mathcal{N}(0, I_d)$.

Implication. As noted in Sec. 3.2, uniform crossover will ensure that solutions are with the high-density shell of the gaussian thus maintaining sample quality.

C. Details on Parameterization of Evolutionary Algorithms and Compared Methods

Here, we review more details on how we initialized our EA search parameters, and how other algorithms were configured for fair comparison.

C.1. EA Parameterization

Genetic Algorithm Mutation Rate and Stability. When using GAs, the mutation operator introduces Gaussian noise to promote exploration. However, excessive perturbations can push solutions outside the high-density shell, leading to poor sample quality. To address this, we use small mutation step sizes, which help ensure that offspring remain within the high-density shell and preserve sample quality. Typically, our mutation size is $\sigma \approx 0.1$.

Evolutionary Strategy Initialization. Here, we briefly describe pitfalls with improper ES initialization, and how we addressed them in our work. Fig. 7 visually depicts this concept. We noted that ES are particularly sensitive to initialization — in the context of this work, this "initialization" is the initial search distribution $q_\phi(\psi)$.

One naive, but simple way to initialize $q_\phi(\psi)$ is to model it as a zero-mean ($\mu = 0$), isotropic (covariance $\sigma = I$) multivariate Gaussian distribution. However, even after a few optimizations steps, the ES algorithm may update μ and

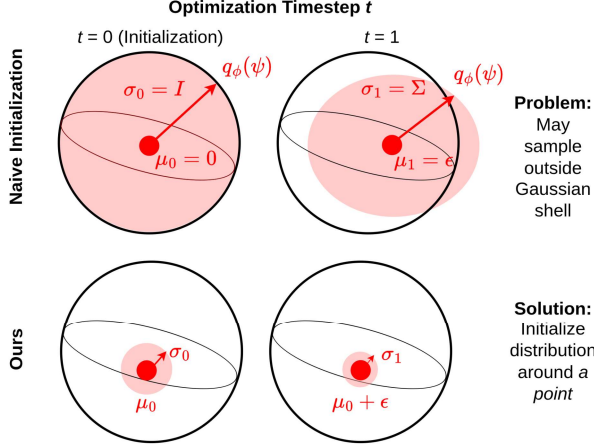


Figure 7. We depict a shortcoming with naive ES initialization, and illustrate our solution. Initializing the search distribution $q_\phi(\psi)$ incorrectly (zero-mean, isotropic multivariate Gaussian) can quickly lead to sampling outside the Gaussian shell, leading to poor sample quality. We encountered this issue in our early experiments. We addressed it by centering $q_\phi(\psi)$ on some point on the shell, and restricting its initial covariance σ_0 to a localized region, rather than encapsulating the entire shell.

σ such that sampling from $q_\phi(\psi)$ no longer yields samples on the Gaussian shell. As a result, sample quality is degraded significantly. This is visually depicted in the top row of Fig. 7.

In order to address this problem, we instead randomly choose an initial μ that exists on the Gaussian shell, and initialize the covariance σ to be restricted to a localized region. This modification was effective at improving sample quality, because our samples were now more closely drawn from the surface of the Gaussian shell.

C.2. Investigating Short-Term EA Performance

Our GAs are initialized with candidate solutions that exist at different locations on the Gaussian hypersphere, improving the diversity of samples early and thus yielding better alignment (**Appendix C.1**). Following prior work [8], our data show GAs to be less effective than gradient-based optimization over long horizons.

In the rebuttal period, we confirmed that GA *initialization* is responsible for short horizon effectiveness. We initialized CoSyNE’s candidate noise vectors P_i “uniformly” as the initial noise vector $P_i = z_T$, to ensure each candidate is restricted to one neighborhood on the shell. We compare this to our “standard” configuration $P_i \sim \mathcal{N}(0, I)$ over 15 optimization steps, using ImageReward on Open Image Preferences.

In this experiment, the uniform ($P_i = z_T$) configuration achieved (0.33 ± 0.89) , much lower than the standard ($P_i \sim \mathcal{N}(0, I)$) configuration score with (1.6 ± 0.23) . Importantly,

the standard configuration covers many distant points on the shell [1], unlike the uniform initialization. This suggests GAs are effective due to this “global” search on the shell, starting from the first optimization step.

C.3. Configuration of Compared Methods

Hyperparameter Description and Nuance. We list the primary hyperparameters of each method in Tab. 6; these define the configurations reported in Tab. 7.

Although some notions—such as “batch size” B —appear across multiple methods, they do not necessarily correspond to the same underlying mechanism. In our method, B either equals the population size ($B = P$) when evaluations are fully parallelized, or is smaller than the population ($B < P$) when we process individuals in batches (Sec. 3.3). Other methods do not necessarily treat the batch dimension in the same manner. Instead, their batch of particles or samples is more similar to a *population of samples to generate* rather than *the number of samples they evaluate per-step*. For example, DSearch-R interprets B as a number of samples to output—but it resamples and replaces samples within the batch throughout the denoising process.

Choosing Configurations. Tab. 7 shows configurations used in Tabs. 1 and 2. It was not straightforward to map our hyperparameters to those of other methods, given the differences noted above. To ensure a fair comparison, we therefore selected configurations that maximized alignment capability while matching the per-prompt running time of our strongest method, CoSyNE, on each reward function.

D. Extended Evaluation Results

D.1. Cross-Model Evaluation on Open Image Preferences

Tab. 8 analyzes three diffusion models and confirms that evolutionary algorithms largely remain competitive across models. For both SD1.5 and SD3, the EA family consistently surpasses the black-box baselines: CoSyNE achieves the highest *best* and *median* rewards on every metric, *e.g.* SD3-CLIP rises from 39.4/37.4 with Zero-Order to 40.6/39.0, while SD3-ImageReward improves from 1.75/1.60 to 1.82/1.75. SNES and PGPE follow the same trend, but with smaller margins. Generalization is weaker on PixArt- α , where improvements vanish, suggesting that black-box latent search may not transfer to architectures trained with alternate objectives such as LCM—we elaborate in Sec. 5. Overall, we show evolutionary search can align multiple diffusion models; however some latent spaces may be more difficult to search than others.

By virtue of our results, we show that optimizing noise with EAs is often sufficient to perform alignment (see Tab. 1), though its effectiveness varies by model (Tab. 8). This raises two questions: (1) Are some latent spaces easier to search

Table 6. **Hyperparameters of Surveyed Methods.** This is a companion table for Tab. 7. We refer the reader to the papers (and respective codebases) for a more detailed explanation of each. Batch size B is interpreted differently by each method, per Appendix C.3.

Algorithm(s)	Parameter	Description
CoSyNE, SNES, PGPE	P	Population size
	B	Batch size, parallel evaluations
	O	Optimization Steps
FKS[40]	PC	Particle Count, similar to population size P
	t_{start}	Resampling denoising step start
	t_{end}	Resampling denoising step end
	t_{freq}	Resampling frequency
DSearch-R[24]	B	Similar to population size P
	$n_{\text{duplicates}}$	Beam width
	w	Tree expansion factor
SVDD[23]	B	Similar to population size P
	$n_{\text{duplicates}}$	Beam width

Table 7. **Tabs. 1 and 2 Hyperparameter Configurations.** These configurations closely match the running time of our best method, CoSyNE for each alignment objective. HPSv2 is slower to evaluate than all other alignment objectives, partially due to its large CLIP backbone. We note that batch size B is interpreted differently by each method, per Appendix C.3.

Algorithm(s)	Objective	Configuration
FKS[40]	ImageReward, CLIP, JPEG	PC = 128, $t_{\text{start}} = 5$, $t_{\text{end}} = 45$, $t_{\text{freq}} = 1$
SVDD[23]	ImageReward, CLIP, JPEG	$B = 6$, $n_{\text{duplicates}} = 20$
DSearch-R[24]	ImageReward, CLIP, JPEG	$B = 8$, $n_{\text{duplicates}} = 5$, $w = 5$, $r = 0.125$
FKS	HPSv2	PC = 16, $t_{\text{start}} = 5$, $t_{\text{end}} = 45$, $t_{\text{freq}} = 1$
SVDD	HPSv2	$B = 6$, $n_{\text{duplicates}} = 10$
DSearch-R	HPSv2	$B = 8$, $n_{\text{duplicates}} = 4$, $w = 4$, $r = 0.125$
CoSyNE, SNES, PGPE	ImageReward, CLIP, HPSv2, JPEG	$B = 16$, $P = 16$

Table 8. **Open Images Preferences Cross-Model Evaluation** The *best-sample* reward and the *median* reward of the best population achieved by each algorithm on Open Image Preferences across three models: StableDiffusion 1.5, StableDiffusion 3, and PixArt- α . Algorithms are configured in the same manner as in Tab. 1. Higher rewards are better.

Algorithm	SD1.5				SD3				PixArt- α (LCM)			
	CLIP		ImgReward		CLIP		ImgReward		CLIP		ImgReward	
	Best	Med.	Best	Med.	Best	Med.	Best	Med.	Best	Med.	Best	Med.
Best-of-N	39.1	34.9	1.49	0.59	39.0	36.0	1.72	1.38	38.8	35.9	1.67	1.25
Zero-Order	39.5	36.4	1.54	0.98	39.4	37.4	1.75	1.60	38.9 [†]	36.1	1.67 [†]	1.29
<i>Ours: Direct Noise Search</i>												
CoSyNE	40.7	38.3	1.69	1.47	40.6	39.0	1.82	1.75	39.2	36.8	1.69	1.45
SNES	40.1	37.6	1.57	1.25	39.9	38.1	1.77	1.68	38.9 [†]	35.8	1.68 [†]	1.21
PGPE	39.0 [†]	36.1	1.37	0.88	38.4	36.4	1.67	1.50	38.7 [†]	35.8 [†]	1.65 [†]	1.23 [†]

than others? (2) How can one design, train, or modify diffusion models to make their latent spaces easier to search? Our reward improvements on PixArt- α were diminished com-

pared to other diffusion models (Tab. 8), suggesting some property of the latent space and/or diffusion model that affects our search efficacy.

D.2. Effect of Longer Optimization Horizons on DNO

Figures 8a to 8c depict the effect of longer optimization timelines on the best-sample, mean, and median rewards for the gradient-based method DNO, compared to gradient-free methods. Gradient-based require longer optimization timelines to reach the same reward as gradient-free methods.

D.3. Population Size Ablation

Figures 9 to 11 depict the effect of increasing population size on the CoSyNE, SNES, and PGPE algorithms. All algorithms benefit from larger population sizes, however PGPE appears to benefit the most.

D.4. Selection Pressure and Convergence

Fig. 12 shows the effect on increasing selection pressure (tournament size) on the median reward and standard deviation. We note that high selection pressure (larger tournament size) results in a rapid reduction in the reward standard deviation as shown in Fig. 12b, this is in line with prior work characterizing the effect of tournament selection [31].

D.5. Qualitative Results

Here we provide qualitative and visual results. We first address reward hacking behavior we witnessed with the PGPE algorithm optimizing for JPEG compressibility. We then present sets of images that illustrate the low sample diversity of genetic algorithms across models and datasets.

D.5.1. Reward Hacking in Sec. 4.2

Here we discuss reward-hacking behavior on our method, PGPE, and minor instances across other surveyed methods. Fig. 17 shows the proclivity of PGPE to "reward-hack" as noted in Sec. 4.2. In many instances, PGPE was able to reduce the JPEG file size, but the resulting images were far too dissimilar or nonsensical compared with the intended image for a prompt. This behavior *is* reward-hacking, because the algorithm completely ignores key prompt details, and instead produces nonsensical images to maximize reward.

D.5.2. Stable Diffusion Qualitative Results

Here we illustrate the differences between ES and GA in longer optimization horizons. We show reward statistics in Fig. 13, Fig. 14, Fig. 15, and Fig. 16. GAs like CoSyNE feature low sample diversity, even as optimization steps scale. This can also happen to ES methods (Fig. 16, however we noted it was less likely to occur with ES. Over longer optimization horizons, ES was able to maximize rewards better than GA while having higher diversity (Sec. 4.3).

D.6. Extended DrawBench Results

Here we include additional, granular results on the DrawBench dataset. We provide this data in Tab. 10, Tab. 11,

Table 9. Number of reward function evaluations per-prompt performed by each method from Tabs. 1 and 2. If a reward evaluation was batched ($B > 1$), we count that as B evaluations.

Algorithm	Reward Evaluations
Best-of-N	240
Zero-Order	240
FKS	5258
SVDD	5880
DSearch-R	7880
CoSyNE	360

Tab. 12, and Tab. 13. Specifically, we include additional statistics (min, max, median, mean, standard) for each reward function. Each subtable for a particular reward function displays the population-best rewards and median rewards measured across the DrawBench dataset.

D.7. Total Reward Function Evaluations

The cost of evaluating the reward function is significantly higher than the cost of sampling [46] — *e.g.* human feedback. Therefore, we report the number of *total reward function evaluations* from Tabs. 1 and 2, which we illustrate in Tab. 9. The evaluation costs are relatively low for the alignment objectives in this work. Even so, a method that can achieve high rewards with fewer samples is desirable in any context. We show our methods took *substantially fewer* reward evaluations than similar works FKD, SVDD, and DSearch-R, while achieving *higher* aesthetic scores than all other methods. We describe how we derived evaluation counts in Tab. 9, which are based on the configurations from Tab. 7.

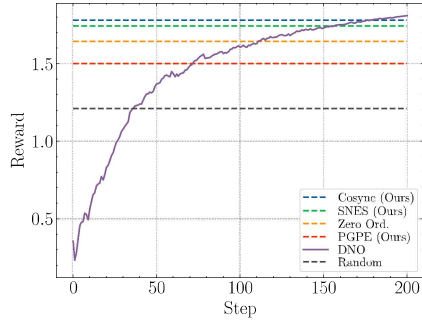
Best-of-N, Zero-Order. These methods were parameterized by $P = 16$, run over 15 optimization steps for 240 total evaluations. Each sample in the population was evaluated once per optimization step.

CoSyNE. CoSyNE generates an intermediate population size that is $1.5 \times$ its input and output population size P , as mentioned in Sec. 4.5. Given $P = 16$, an effective population of $P = 24$ is realized; over 15 optimization steps this leads to 360 total evaluations.

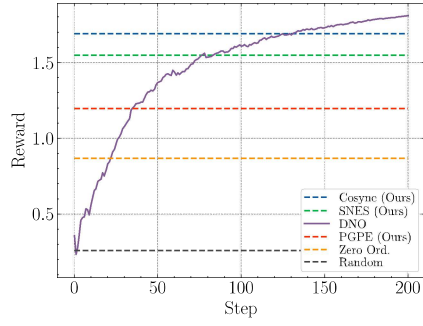
FKS. FKS resampled latents a total of 41 times (denoising step 5 until step 45) over a population of 128 particles for 5258 total evaluations. Each particle is evaluated once at each denoising step.

SVDD. SVDD maintains 120 candidate latent noise vectors over 49 denoising steps for 5880 total evaluations. Each candidate is evaluated once at each denoising step.

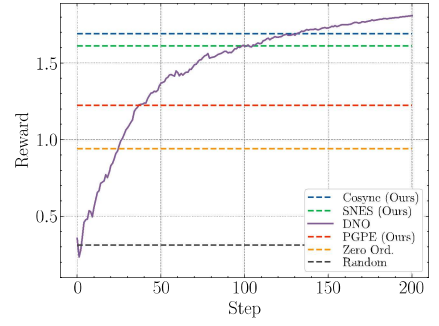
DSearch-R. DSearch-R maintains 5 candidate latent noise vectors, and follows a schedule to further expand its search, yielding ~ 7880 evaluations under their default (exponential) search schedule. This resamples latents at ~ 38 denoising steps.



(a) Best-sample reward per step. Horizontal lines show the max reward in **50** steps.

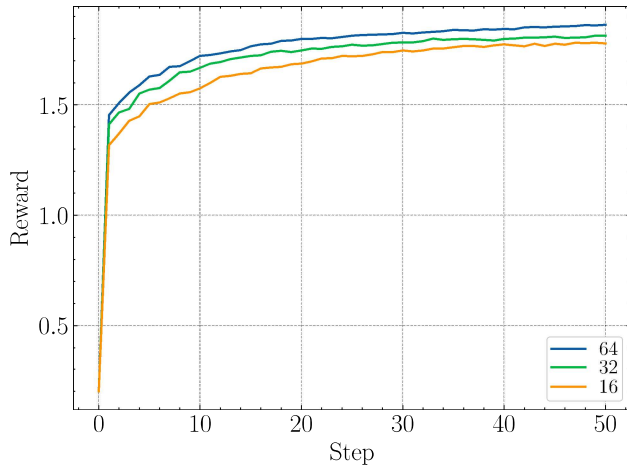


(b) Mean reward per step. Horizontal lines show the max mean in **50** steps.

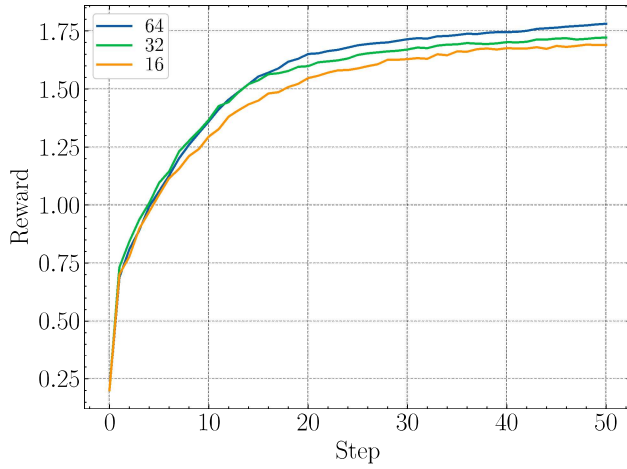


(c) Median reward per step. Horizontal lines show the max median in **50** steps.

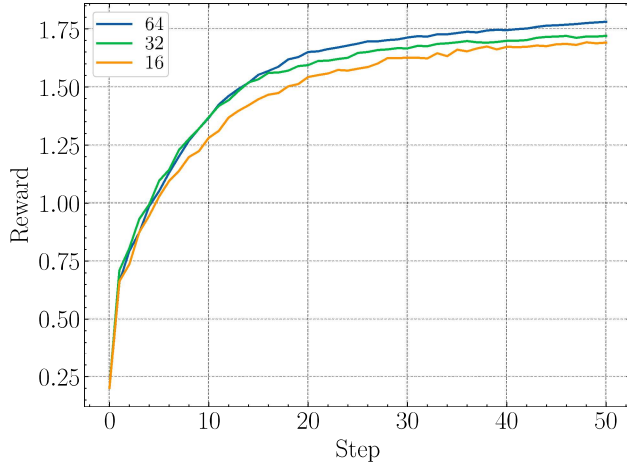
Figure 8. **DNO with Long Optimization Horizon.** ImageReward per step on Open Image Preferences.



(a) Best-sample reward per step.

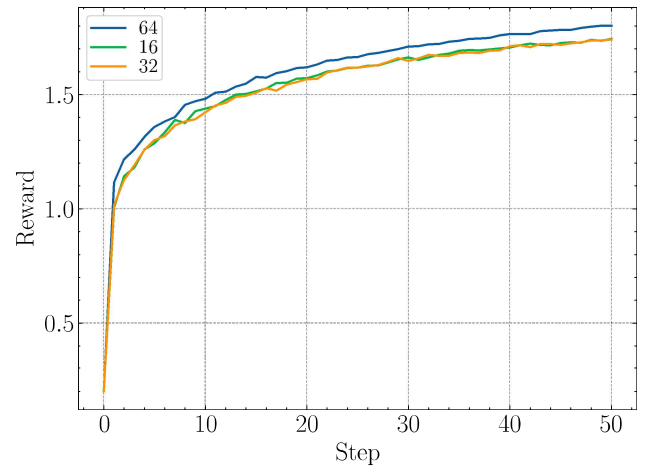


(b) Mean reward per step.

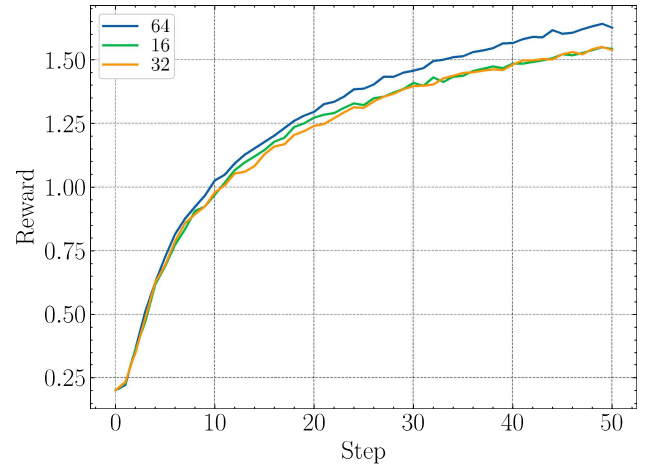


(c) Median reward per step.

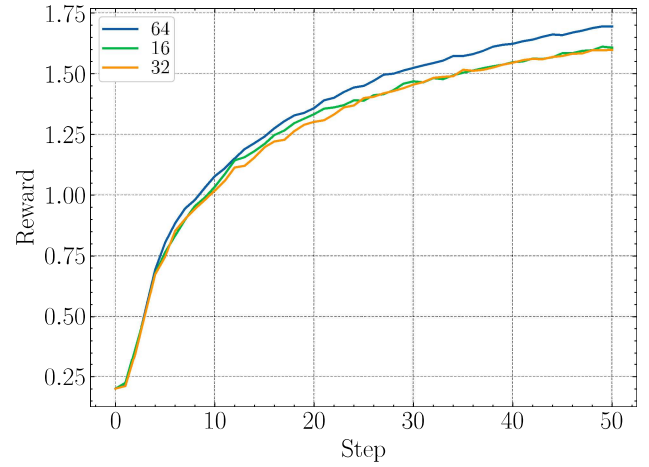
Figure 9. **Genetic Algorithm: CoSyNE** Reward statistics for the CoSyNE algorithm across optimization step.



(a) Best-sample reward per step.

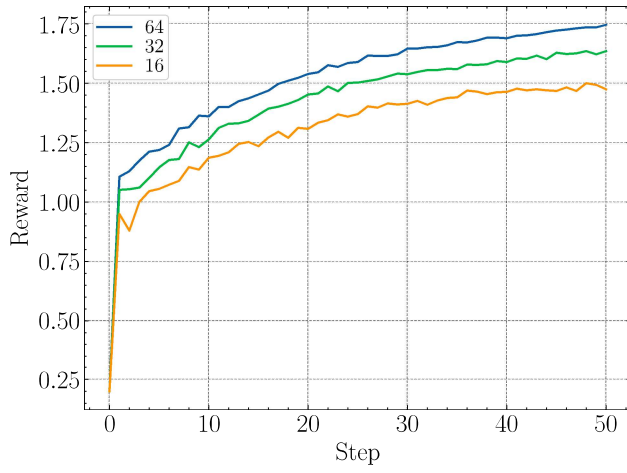


(b) Mean reward per step.

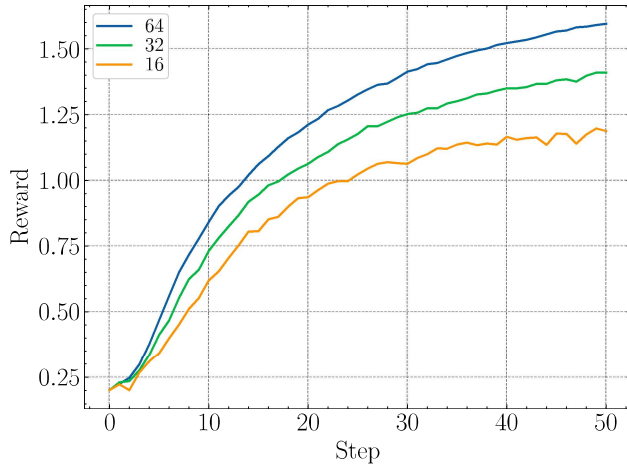


(c) Median reward per step.

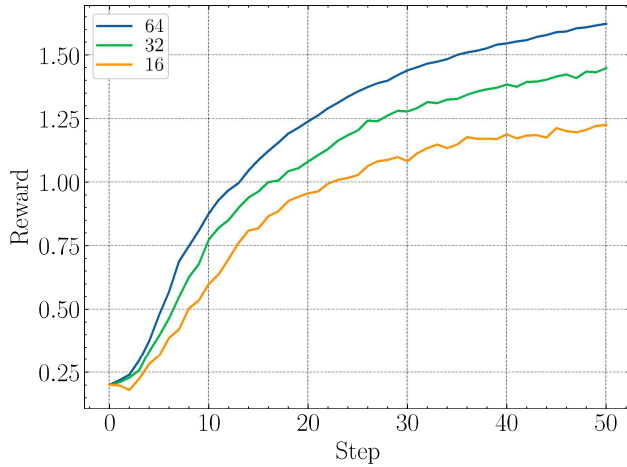
Figure 10. **Evolutionary Strategy: SNES** Reward statistics for the SNES algorithm across optimization step.



(a) Best-sample reward per step.

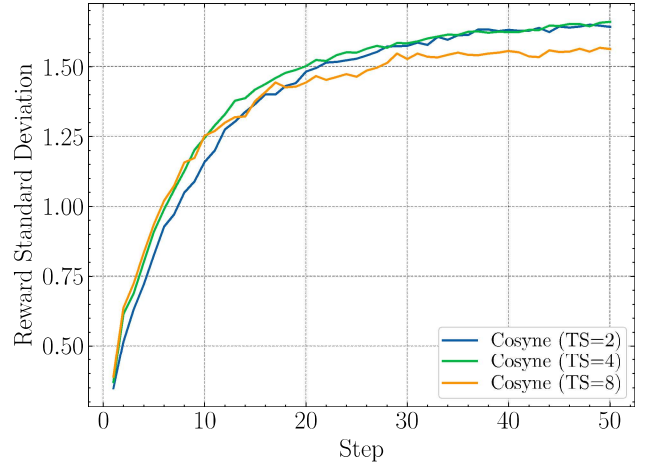


(b) Mean reward per step.

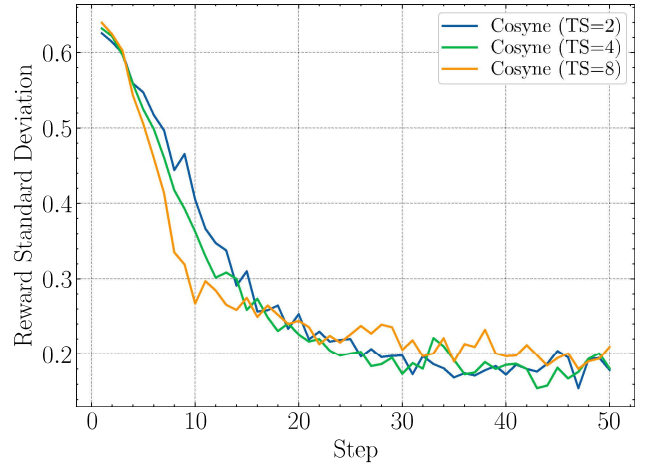


(c) Median reward per step.

Figure 11. **Evolutionary Strategy: PGPE** Reward statistics for the PGPE algorithm across optimization step.



(a) Median reward per step.



(b) Reward standard deviation per step.

Figure 12. **Genetic Algorithm: CoSyNE** Reward statistics per step, as measured on Open Image Preferences. Reward statistics are measured with increasing selection pressure, *i.e.* increasing tournament sizes of 2, 4, and 8. We fix the population size to 16.

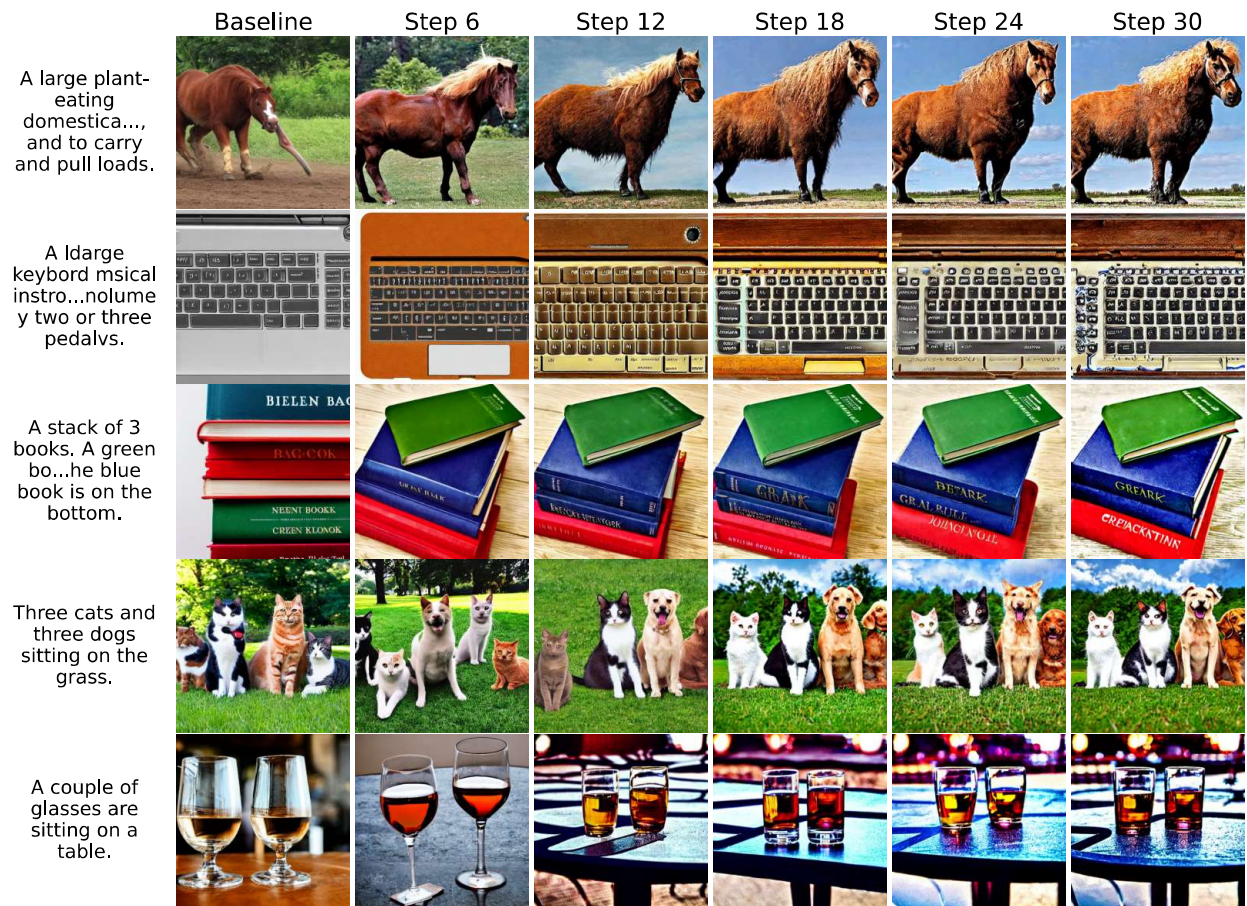


Figure 13. **Qualitative Sample Diversity:** Randomly selected DrawBench prompts evaluated on StableDiffusion-1.5 with ImageReward. CoSyNE was used to perform alignment. Across optimization steps, the diversity between samples quickly diminishes. Genetic algorithms such as CoSyNE are particularly vulnerable to this phenomenon.

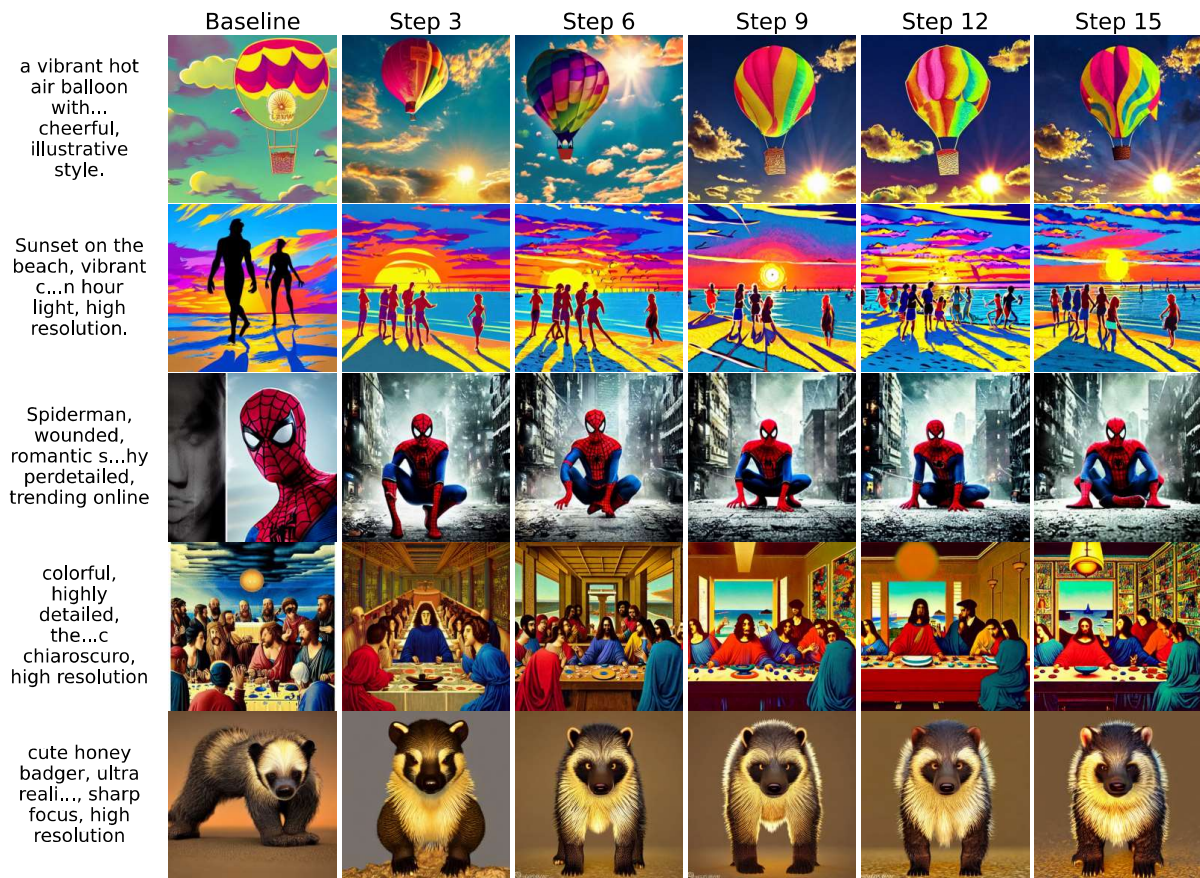


Figure 14. **Qualitative Sample Diversity:** Randomly selected Open Image Preference prompts evaluated on StableDiffusion-1.5 with ImageReward. CoSyNE was used to perform alignment. Note the low sample diversity across optimization steps. We identify this as a consistent occurrence with genetic algorithms such as CoSyne.

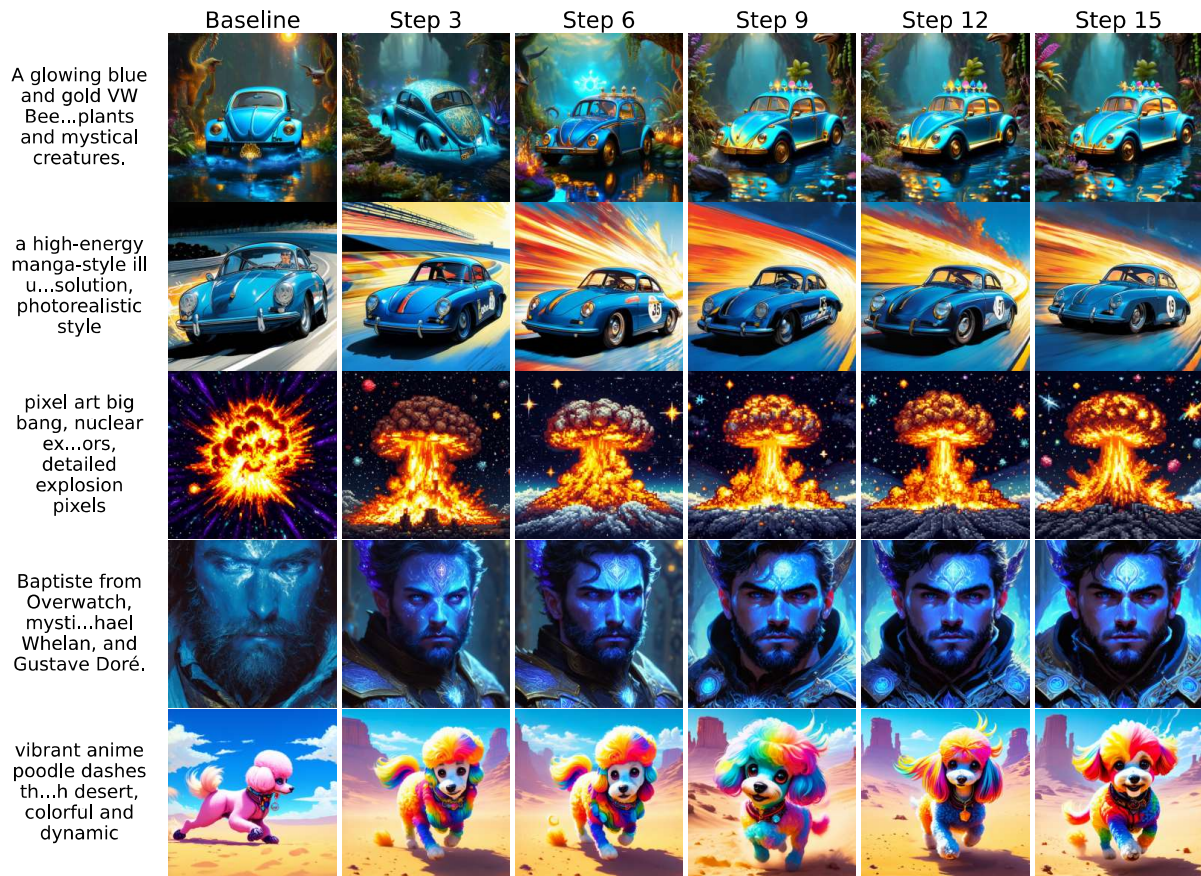


Figure 15. **Qualitative Sample Diversity:** Randomly selected Open Image Preference prompts evaluated on StableDiffusion-3 with ImageReward. CoSyNE was used to perform alignment. This low sample diversity result on StableDiffusion-3 was also noticed on StableDiffusion-1.5.

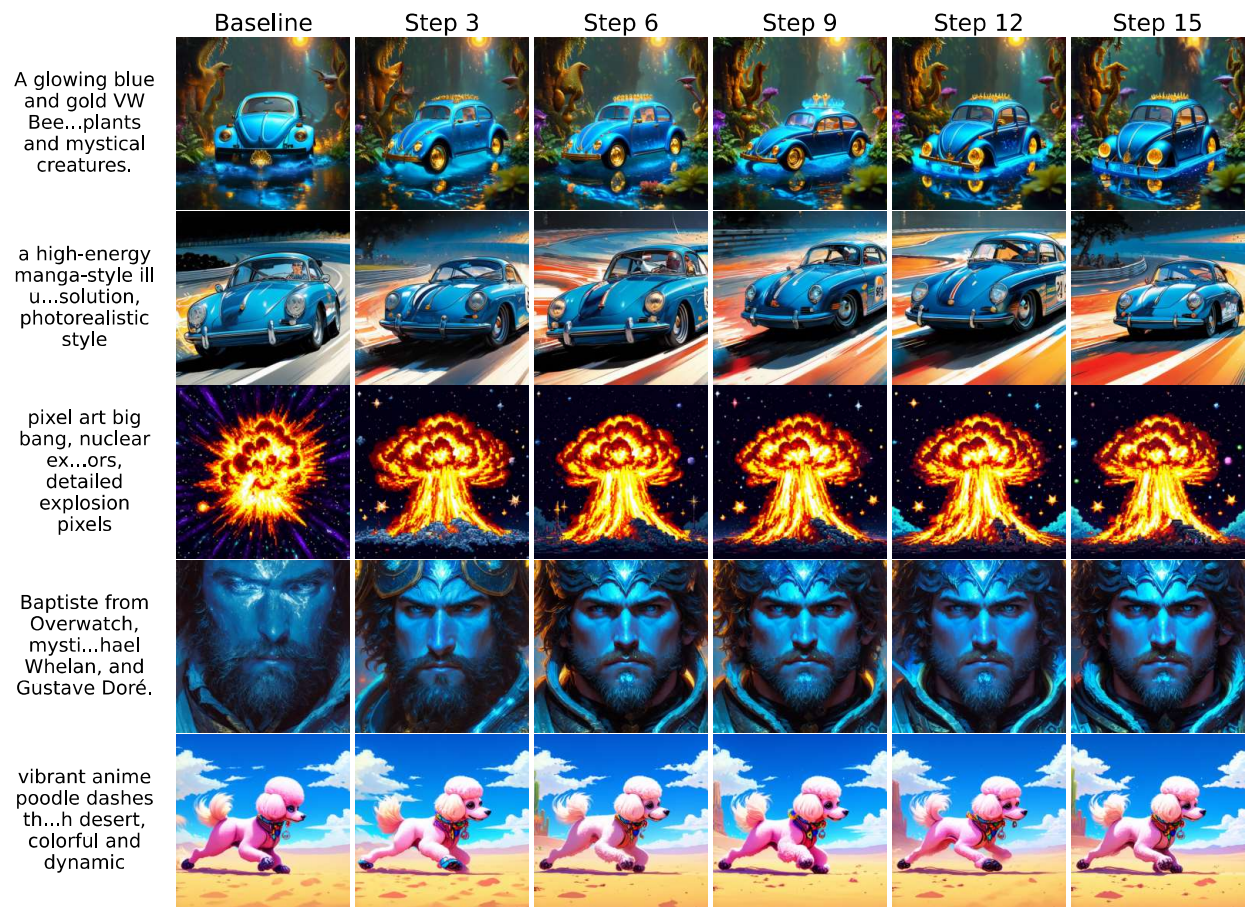


Figure 16. **Qualitative Sample Diversity:** Randomly selected Open Image Preference prompts evaluated on StableDiffusion-3 with ImageReward. SNES was used to perform alignment.

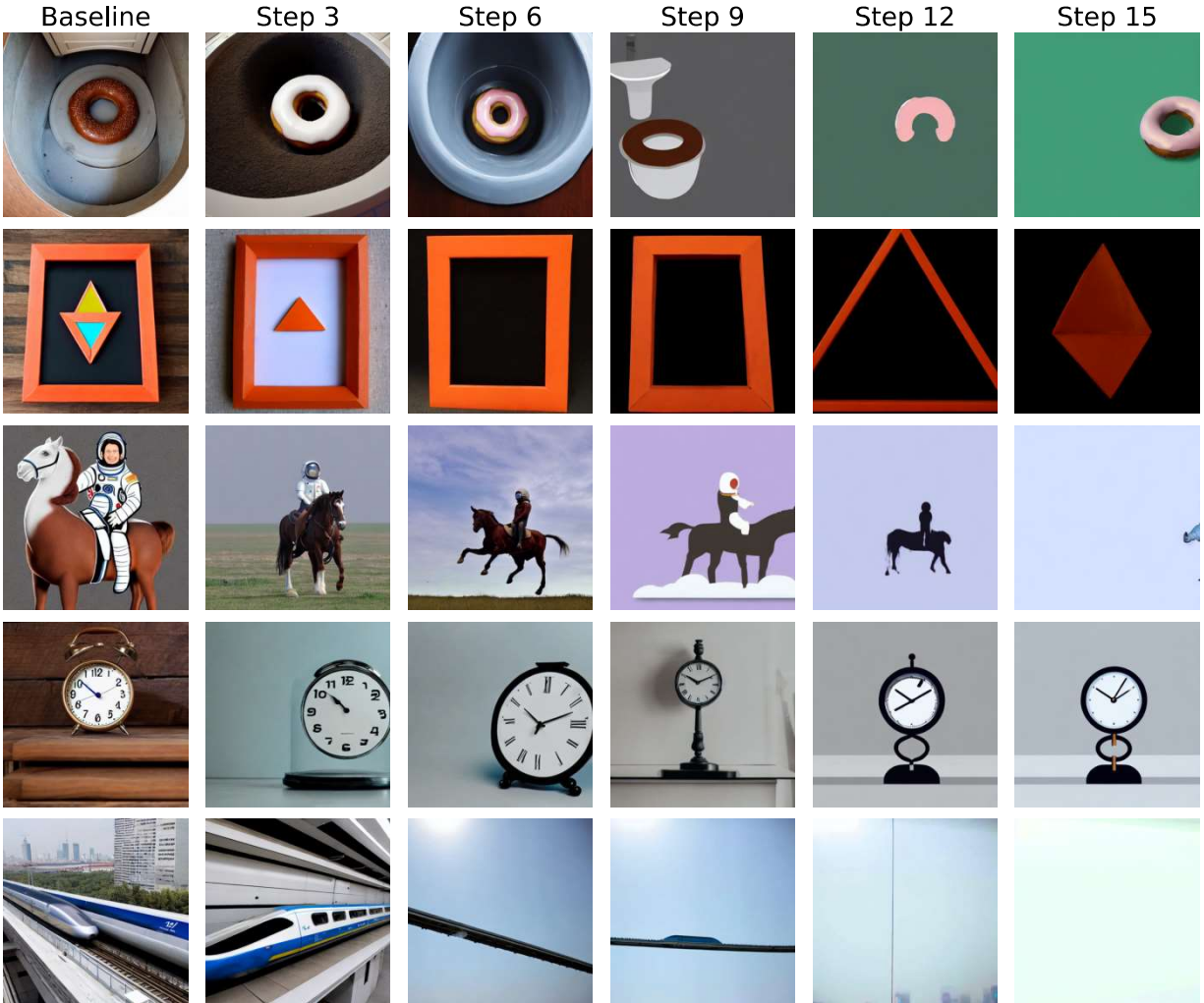


Figure 17. **PGPE Reward Hacking:** Randomly selected images created using inference-time alignment with the PGPE algorithm (transformation search) to minimize the JPEG size (DrawBench). PGPE neglects the prompt, and naively attempts to minimize JPEG file size by producing monochromatic or simple images.

Table 10. Extended DrawBench results with ImageReward.

(a) Population-Best Reward

Algorithm	Soln. Space	Mean \pm Std	Median	Min	Max
Random	-	1.407 ± 0.519	1.583	-0.569	1.988
ZeroOrder	-	1.455 ± 0.530	1.673	-0.408	2.002
DNO	-	0.707 ± 0.926	0.910	-1.744	1.950
Cosyne	Noise	1.614 ± 0.417	1.765	-0.093	2.002
Cosyne	Rotation	1.532 ± 0.437	1.659	-0.259	2.001
SNES	Noise	1.392 ± 0.523	1.535	-0.402	1.977
SNES	Rotation	1.338 ± 0.561	1.482	-0.744	1.983
PGPE	Noise	1.261 ± 0.646	1.419	-1.166	1.978
PGPE	Rotation	1.276 ± 0.577	1.426	-0.641	1.981

(b) Median Reward

Algorithm	Soln. Space	Mean \pm Std	Median	Min	Max
Random	-	0.383 ± 0.844	0.437	-1.717	1.877
ZeroOrder	-	0.966 ± 0.733	1.093	-1.532	1.976
DNO	-	0.707 ± 0.926	0.910	-1.744	1.950
Cosyne	Noise	1.379 ± 0.573	1.565	-0.608	1.993
Cosyne	Rotation	1.225 ± 0.633	1.420	-0.811	1.982
SNES	Noise	0.788 ± 0.801	0.934	-1.387	1.944
SNES	Rotation	0.942 ± 0.727	1.132	-1.541	1.954
PGPE	Noise	0.921 ± 0.801	1.112	-1.681	1.951
PGPE	Rotation	0.882 ± 0.731	1.003	-1.393	1.970

Table 11. Extended DrawBench results with HPSv2.

(a) Population-Best Reward

Algorithm	Soln. Space	Mean \pm Std	Median	Min	Max
Random	-	0.301 ± 0.016	0.304	0.258	0.344
ZeroOrder	-	0.304 ± 0.017	0.306	0.260	0.347
DNO	-	0.286 ± 0.017	0.286	0.241	0.324
Cosyne	Noise	0.310 ± 0.017	0.311	0.271	0.352
Cosyne	Rotation	0.307 ± 0.017	0.308	0.267	0.347
SNES	Noise	0.300 ± 0.016	0.301	0.260	0.346
SNES	Rotation	0.300 ± 0.017	0.302	0.250	0.342
PGPE	Noise	0.299 ± 0.017	0.301	0.256	0.342
PGPE	Rotation	0.300 ± 0.017	0.301	0.251	0.338

(b) Median Reward

Algorithm	Soln. Space	Mean \pm Std	Median	Min	Max
Random	-	0.282 ± 0.016	0.284	0.238	0.318
ZeroOrder	-	0.290 ± 0.017	0.292	0.245	0.336
DNO	-	0.286 ± 0.017	0.286	0.241	0.324
Cosyne	Noise	0.300 ± 0.016	0.302	0.259	0.340
Cosyne	Rotation	0.299 ± 0.017	0.300	0.262	0.340
SNES	Noise	0.286 ± 0.016	0.288	0.247	0.324
SNES	Rotation	0.290 ± 0.017	0.293	0.243	0.333
PGPE	Noise	0.290 ± 0.017	0.291	0.252	0.335
PGPE	Rotation	0.290 ± 0.017	0.294	0.243	0.329

Table 12. Extended DrawBench results with CLIP.

(a) Population-Best Reward					
Algorithm	Soln. Space	Mean \pm Std	Median	Min	Max
Random	-	37.139 ± 3.665	37.016	29.094	49.562
ZeroOrder	-	37.590 ± 3.864	37.719	28.000	48.344
DNO	-	26.237 ± 3.638	26.309	18.047	35.562
Cosyne	Noise	38.791 ± 3.812	38.672	30.328	50.062
Cosyne	Rotation	38.405 ± 3.795	37.938	30.125	50.188
SNES	Noise	38.061 ± 3.751	37.875	29.094	49.562
SNES	Rotation	37.407 ± 3.831	37.188	29.250	48.156
PGPE	Noise	37.052 ± 3.607	36.656	29.250	47.844
PGPE	Rotation	36.884 ± 3.763	36.609	28.562	48.875
(b) Median Reward					
Algorithm	Soln. Space	Mean \pm Std	Median	Min	Max
Random	-	32.867 ± 3.502	32.812	20.562	41.750
ZeroOrder	-	34.545 ± 3.698	34.438	24.578	44.938
DNO	-	26.237 ± 3.638	26.309	18.047	35.562
Cosyne	Noise	36.455 ± 3.614	36.141	28.219	46.938
Cosyne	Rotation	36.393 ± 3.712	36.266	26.266	47.250
SNES	Noise	35.623 ± 3.563	35.328	24.547	44.969
SNES	Rotation	34.699 ± 3.662	34.438	26.594	45.812
PGPE	Noise	34.397 ± 3.455	34.031	25.859	43.469
PGPE	Rotation	34.772 ± 3.602	34.516	27.172	44.750

Table 13. Extended DrawBench results with JPEG File Size. Entries are file sizes listed in kilobytes (kB).

(a) Population-Best Reward					
Algorithm	Soln. Space	Mean \pm Std	Median	Min	Max
Random	-	66.92 \pm 17.82	64.23	26.55	136.58
ZeroOrder	-	53.35 \pm 18.42	51.47	15.89	116.90
Cosyne	Noise	39.52 \pm 15.88	36.79	13.78	79.34
Cosyne	Rotation	38.73 \pm 16.25	36.82	9.96	108.24
SNES	Noise	71.35 \pm 21.36	67.54	28.47	157.80
SNES	Rotation	57.78 \pm 22.40	54.63	11.54	174.64
PGPE	Noise	88.29 \pm 23.68	84.50	31.79	160.11
PGPE	Rotation	18.29 \pm 12.37	14.76	5.00	91.34
(b) Median Reward					
Algorithm	Soln. Space	Mean \pm Std	Median	Min	Max
Random	-	105.43 \pm 20.66	103.54	60.79	203.58
ZeroOrder	-	62.76 \pm 20.65	61.21	18.32	158.85
DNO	-	94.64 \pm 22.78	91.78	47.11	174.45
Cosyne	Noise	44.76 \pm 16.40	42.42	15.42	87.25
Cosyne	Rotation	42.28 \pm 16.63	40.86	11.91	111.33
SNES	Noise	77.98 \pm 22.33	73.70	32.36	168.50
SNES	Rotation	66.10 \pm 23.36	62.25	16.72	185.45
PGPE	Noise	97.02 \pm 24.02	93.35	53.09	173.79
PGPE	Rotation	20.18 \pm 13.10	16.60	5.38	94.78