

---

# Inference-Time Alignment of Diffusion Models with Evolutionary Algorithms

---

**Purvish Jajal\***

Purdue University

West Lafayette, IN 47907

pjajal@purdue.edu

**Nick John Eliopoulos\***

Purdue University

West Lafayette, IN 47907

neliopou@purdue.edu

**Benjamin Shiue-Hal Chou**

Purdue University

West Lafayette, IN 47907

chou1500@purdue.edu

**George K. Thiruvathukal**

Loyola University Chicago

Chicago, IL 60660

gkt@cs.luc.edu

**James C. Davis**

Purdue University

West Lafayette, IN 47907

davisjam@purdue.edu

**Yung-Hsiang Lu**

Purdue University

West Lafayette, IN 47907

yunglu@purdue.edu

## Abstract

Diffusion models are state-of-the-art generative models in various domains, yet their samples often fail to satisfy downstream objectives such as safety constraints or domain-specific validity. Existing techniques for alignment require gradients, internal model access, or large computational budgets. We introduce an inference-time alignment framework based on evolutionary algorithms. We treat diffusion models as black-boxes and search their latent space to maximize alignment objectives. Our method enables efficient inference-time alignment for both differentiable and non-differentiable alignment objectives across a range of diffusion models. On the DrawBench and Open Image Preferences benchmark, our EA methods outperform state-of-the-art gradient-based and gradient-free inference-time methods. In terms of memory consumption, we require 55% to 76% lower GPU memory than gradient-based methods. In terms of running-time, we are 72% to 80% faster than gradient-based methods. We achieve higher alignment scores over 50 optimization steps on Open Image Preferences than gradient-based and gradient-free methods.

## 1 Introduction

Diffusion models [40, 34, 9] have emerged as state-of-the-art generative models for synthesizing high-quality images, video, and audio [3, 16, 18, 25]. However, the outputs of diffusion models often fail to meet downstream objectives, such as adhering to user preferences [53, 51, 37], safety constraints [28], or domain-specific validity [14, 49]. This problem is referred to as *alignment*—ensuring that generated samples satisfy external objectives beyond the original maximum-likelihood objective [24, 48, 2, 21, 5, 31, 4]. Alignment is an important concern for many practical applications such as image generation [53, 51, 37] and drug discovery [14, 49].

Methods for diffusion model alignment can be categorized into two types: fine-tuning-based alignment and inference-time alignment. Fine-tuning-based alignment typically relies on supervised fine-tuning or reinforcement learning to adjust model parameters [24]. In contrast, inference-time alignment modifies the sampling process during generation to maximize a reward function (*i.e.*, alignment objective) [45]. These methods aim to be computationally efficient and support arbitrary rewards while maintaining sample quality. However, existing approaches have several drawbacks: they are often not black-box, requiring access to gradients or internal model states; they cannot efficiently

---

\* Purvish Jajal and Nick John Eliopoulos contributed equally to this work.

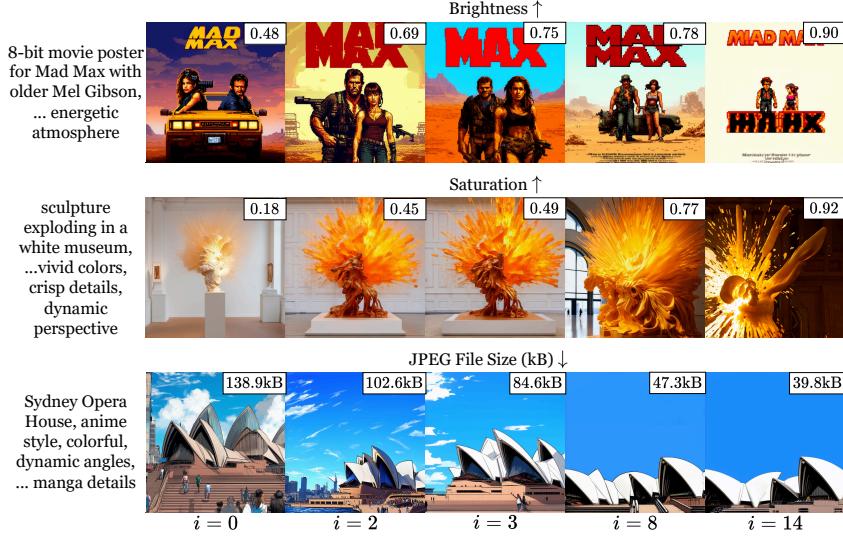


Figure 1: Samples generated by our method on Stable Diffusion-3 across three different prompts under various alignment objectives (brightness, saturation, file size). Each row shows the progression over optimization steps  $i$ , with the corresponding metric values displayed in the top-right corner. The noise in each generation is optimized via the CoSyNE algorithm [13] over 14 optimization steps. Arrows ( $\uparrow/\downarrow$ ) indicate whether the metric is being maximized or minimized.

generate *many* aligned samples; and they are often not sample-efficient. As a result, they are difficult to integrate with a variety of models, sampling strategies, and reward functions; generating many aligned samples becomes cost-prohibitive; and obtaining aligned samples may take too long.

In this work, we demonstrate that evolutionary algorithms (EAs)—specifically genetic algorithms (GA) and evolutionary strategies (ES)—can be used to align diffusion models. As a class of algorithms, EAs have desirable properties: they are black-box, gradient-free optimizers, that can be highly parallelized [39, 36]. By interpreting the diffusion models as black-boxes and searching over their latent space by using EAs we can optimize the alignment objective. Our approach enables efficient alignment that supports both differentiable and non-differentiable rewards, and is compatible with various diffusion models. Fig. 1 depicts our method applied to various alignment objectives.

To quantify alignment quality, GPU-memory footprint, and runtime efficiency, we evaluate our inference-time alignment method on two prompt collections [35, 6], four reward functions [53, 51, 32, 4], and three diffusion models [34, 9, 3]. On both prompt collections, our method outperforms existing gradient-free and gradient-based inference-time alignment approaches. In terms of computational costs, we exhibit 55% to 76% reduction in GPU memory than gradient-based methods, while only consuming up to  $1.5\times$  more memory than gradient-free methods. When producing the same number of aligned samples, evolutionary methods achieve  $3.5\times$  to  $5.5\times$  lower runtime per optimization step compared to gradient-based methods, and only  $1.1\times$  to  $1.6\times$  higher running time per step than gradient-free methods. Moreover, evolutionary methods reach higher alignment scores than gradient-free and gradient-based methods in 10 optimization steps. In sum, we contribute:

- We propose a novel inference-time alignment approach using evolutionary algorithms (EAs). Specifically, we introduce two black-box methods for aligning the outputs of diffusion models: (1) optimizing noise, and (2) optimizing transformations applied to the noise. We elaborate upon the requisite choices for utilizing EAs for inference-time alignment, such as search algorithms, initialization, operators, and computational considerations.
- We demonstrate the effectiveness of two widely used classes of evolutionary algorithms on the diffusion model alignment task: genetic algorithms and evolutionary strategies. Our evaluation spans a diverse set of reward functions and diffusion models.

## 2 Related Work

In this section, we review prior work on diffusion model alignment (§ 2.1) and evolutionary algorithms (§ 2.2). An extended treatment of related works can be found in Appendix B.1.

### 2.1 Diffusion Model Alignment

Diffusion models use a reverse diffusion process to convert some latent noise distribution into a data distribution, such as images [40, 16]. The reverse diffusion process iteratively denoises the initial latent noise  $z_T$  over some number of steps  $T$  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 [2].

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 [21, 5, 31, 4, 48, 2]. Fine-tuning based methods, although powerful, require retraining and thus all the associated costs.

We follow 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  $\psi$  that maximizes the expected reward  $R(x)$  of samples  $x$  from a pretrained diffusion model  $p_\theta$ , as shown in Eq. (1).

$$\psi^* = \arg \max_{\psi \in \Psi} \mathbb{E}_{x \sim p_\theta(x|\psi)} [R(x)]. \quad (1)$$

Examples of control variables,  $\psi$ , include optimizing conditional input prompts [15, 12, 27, 10], manipulating cross-attention layers [11], and tuning latent noise vectors (noise optimization) to guide the diffusion trajectory [47, 42, 23, 45, 55, 26]. In this work, we focus exclusively on noise optimization ( $\psi = z$ ), which is the most general method. 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) [42] 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—noise sampled from high-dimensional Gaussians concentrate on the surface of a spherical shell [46]. The drawbacks of gradient-based methods are: they are not black-box; they are computationally expensive, when aligning multiple samples; and require long optimization budgets.

*Gradient-free* methods, on the other hand, explore the space of noise vectors or trajectories using only search or sampling methods. Uehara *et al.* [45] provide a comprehensive overview of various inference-time algorithms, largely covering sampling-based methods. Ma *et al.* [26] employ three different sample search strategies — random search, zero-order search (similar to hill climbing), and search-over-paths. Li *et al.* propose DSearch [23], a dynamic beam search algorithm. These techniques vary in their trade-offs. DSearch and the techniques outlined by Uehara *et al.* are not black-box and may have large computational costs. Although random and zero-order search are also black-box and computationally efficient, evolutionary methods outperform them (§ 4.2).

### 2.2 Evolutionary Algorithms

Evolutionary algorithms (EAs) are a class of biologically inspired methods that can be used to solve a variety of black-box optimization problems [8, 39]. They iteratively select, recombine, and mutate populations of solutions to improve their fitness (quality of the solution). Among the various EAs, genetic algorithms (GAs) [8, 39] and natural evolutionary strategies (ES) [50] are most pertinent to this work. Genetic algorithms and natural evolutionary strategies have been widely applied in deep learning and deep reinforcement learning. In deep learning, genetic algorithms have facilitated neural architecture search and hyperparameter tuning [33, 29, 54]. In deep reinforcement learning, both genetic algorithms and natural evolutionary strategies have been shown to be competitive with

back-propagation based methods such as policy gradients [41, 36]. GA and ES are also used in image generation [22, 38, 43, 10] and artificial life simulations [20]. However, to our knowledge, we are the first work to apply EAs for black-box diffusion model alignment.

### 3 Inference Time Alignment via Evolutionary Algorithms

We cast inference-time alignment as a black-box search problem and apply an evolutionary search framework. For a pretrained diffusion model  $p_\theta$  and reward function  $R(x)$ , we aim to find parameters  $\phi$  of the search distribution  $q_\phi(\psi)$  that maximizes the expected reward in Eq. (2).

$$\phi^* = \arg \max_{\phi} \underbrace{\mathbb{E}_{\psi \sim q_\phi(\psi)} \left[ \mathbb{E}_{x \sim p_\theta(x|\psi)} [R(x)] \right]}_{\text{search distribution}}. \quad (2)$$

For generality, we write  $x \sim p_\theta(x \mid \psi)$  due to many diffusion samplers being stochastic. In practice, we treat this as a deterministic map from control variable to sample, *i.e.*,  $f_\theta : \psi \mapsto x$ . See Appendix B.2 for further details.

In § 3.1, we review design goals and requirements for our method. In § 3.2 we define the solution spaces ( $\psi$ ), over both noise and noise-transformations. In § 3.3 we describe how genetic algorithms and natural evolutionary strategies define the search distribution  $q_\phi(\psi)$  and how they optimize  $\phi$  in Eq. (2). Finally, § 3.4 presents our implementation and computational considerations.

#### 3.1 Design Criteria

Prior works [45, 42] have outlined a set of Criteria that characterize effective inference-time alignment approaches: **(C1)** maintaining sample quality; **(C2)** computationally efficient; and **(C3)** supporting non-differentiable reward functions. In this work, we advocate a fourth criterion: **(C4)** alignment methods should strive to be black-box. Black-box inference-time alignment can be easily integrated with a variety of models, sampling strategies, and reward functions. For example, a black-box alignment method could be applied to a quantized diffusion model for fast inference, and could simultaneously support arbitrary reward functions.

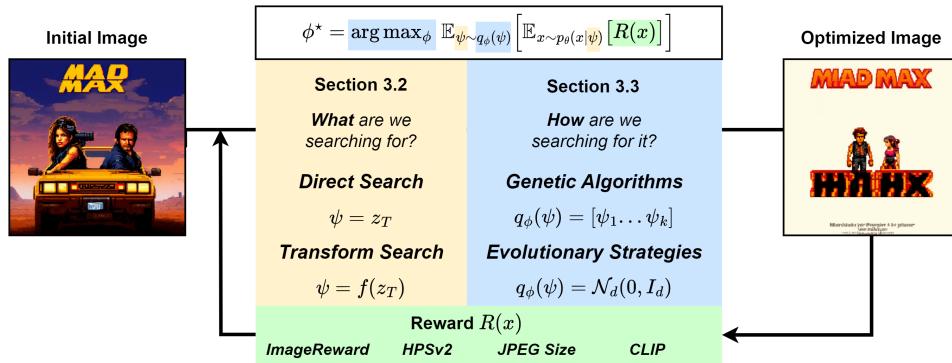


Figure 2: Illustration of our evolutionary search approach to noise vector optimization. Here, we depict the connection between terms of Eq. (2) and components of our method via color-coding.

#### 3.2 Defining the Solution Space

Before applying evolutionary algorithms, we must define the *solution space*, *i.e.*, what we will search over,  $\psi$ . For aligning diffusion models, we explore two options: (1) directly searching over *noise vectors*, and (2) searching over *transformations* of an initial noise vector. In both cases, our overarching goal is to identify a solution that maximizes Eq. (2).

**Searching over Noise.** The simplest solution space we can define is over the initial noise, *i.e.*,  $\psi = z'_T$  in Eq. (2). Concretely, we define the search variable  $\psi$  to be the initial noise vector,  $z'_T \sim q_\phi(z'_T)$ , and optimize  $\phi$  to maximize the expected reward. This direct search is straightforward, but hinges on

proper initialization and careful tuning of mutation; without these, evolved solutions can drift into low-density regions of the latent space, leading to poor sample quality.

**Searching for Noise Transformations.** Instead of searching over noise vectors directly, we can define an affine transformation of an initial noise vector ( $z_T$ ) and search over its parameters. We define the transformed noise as  $z'_T = f(z_T) = Az_T + b$ , where  $A \in \mathbb{R}^{d \times d}$ ,  $z_T, b \in \mathbb{R}^d$ . Thus, in Eq. (2) we set  $\psi = [A, b]$  with  $[A, b] \sim q_\phi([A, b])$  and optimize  $\phi$  to maximize the expected reward in Eq. (2) under  $x \sim p_\theta(x | Az_T + b)$ .

In general,  $z'_T$  may not be within the high-density shell of  $\mathcal{N}(0, I)$ , thus this formulation can often suffer from poor sample quality. To ensure that  $z'_T$  exists within the high-density shell, we can ensure  $A$  is orthonormal. If  $A$  is orthonormal (and  $b = 0$ ),  $z'_T$  stays within the high-density shell of  $\mathcal{N}(0, I)$  by the rotational invariance property of the Gaussian. Since unconstrained evolutionary updates on  $A$  generally break orthonormality, we apply matrix decompositions,  $T: A \xrightarrow{T} A'$ , to construct an orthonormal matrix  $A'$  from  $A$ . In this work, we use the QR decomposition: we decompose the matrix  $A$  and use the orthonormal component  $Q$ , i.e.,  $A \xrightarrow{QR} Q$ , meeting **(C1)**.

**Choosing between noise and noise transformation search.** The choice between noise and noise-transformation search hinges on the type of regularization each imposes and their relative computational costs. Noise search, while being the most direct and straightforward way to define our search space, does not confine solutions to the high-density shell of the Gaussian. Consequently, if the search algorithms (§ 3.3) fail to ensure shell-confinement, sample quality degrades. In contrast, searching over noise transformations will guarantee (given  $b = 0$ , see § 3.2) residence within the high-density shell regardless of the choice of evolutionary algorithm. However, noise-transformation search incurs extra cost for factorization and linear transforms; we describe mitigation strategies in § 3.4.

### 3.3 Searching the Solution Space

Given our solution space ( $\psi$ ) we now turn to *how* we define the search distribution and find its parameters, *i.e.*,  $q_\phi(\psi)$ . Our requirements **(C1–C4)** call for black-box, gradient-free optimizers that preserve sample quality and support arbitrary rewards. Among the large family of optimization approaches [7, 19, 8], evolutionary algorithms (EAs) satisfy these criteria and have a history of efficacy in machine learning contexts (§ 2.2).

We consider two families of EA: genetic algorithms (GAs) and natural evolutionary strategies (ES). For each, we first review their core mechanics, then describe how these mechanics map to solving the diffusion alignment task. Their specific optimization procedures are outlined in Appendix B.3.

**Genetic Algorithms (GAs).** Genetic algorithms maintain a population of candidate solutions,  $\psi_i$ , and iteratively improve them via selection, crossover, and mutation [8]. In Eq. (2), the search distribution is an empirical distribution over the population of solutions.

In the context of alignment, GAs satisfy the criteria in § 3.1 and offer practical advantages that improve alignment performance. First, the optimization procedure—selection, crossover, mutation—under correct parameterization can ensure the noise search is regularized to the high-density shell, provided the initial population resides within. In particular, *uniform crossover* (independent coordinate swaps) or *permutation-based* mutations do so—see Appendix B.4 for details. Second, maintaining a population enables broad, parallel exploration early in optimization, while increasing selection pressure later drives convergence towards the promising regions (Appendix D.4). This capacity to explore multiple basins simultaneously makes GAs well-suited to multi-modal alignment objectives, *e.g.*, human preferences. However, our experiments show that diversity collapse can cause premature convergence, hence marginal improvements over longer optimization timelines (see § 4.3 and § 4.4).

**Natural Evolutionary Strategies (ES).** Natural evolutionary strategies perform black-box optimization by adapting the parameterized search distribution  $q_\phi$  over solutions [50]. Unlike GAs,  $q_\phi$  is a parameterized search distribution (*e.g.*, multivariate normal).

In the context of alignment, ES satisfy the criteria in § 3.1 and offer a distinct set of advantages over GAs. They maintain a search distribution, so their memory footprint is lower: only distribution parameters (*e.g.*, mean, covariance, step size) need be stored. This parameterized distribution also lets us draw an arbitrary number of candidate solutions, making ES ideal when many aligned samples are required. Moreover, although termed “gradient-free”, evolutionary strategies update their search

distribution with a gradient approximation [36, 41]. As a result, they move through the reward landscape with a coarse sense of direction lacked by GAs. This directional information translates into alignment improvements over long optimization horizons (see § 4.3).

**Choosing Between GA and ES for Alignment.** Although GA and ES methods satisfy the criteria outlined in § 3.1, they exhibit markedly different properties. First, the search distribution for GA is an empirical distribution, whereas ES use multivariate normal distributions. Consequently, GA exhibit higher memory usage than ES because the entire population must reside in memory (see § 4.5). This has consequences for sampling as well: additional samples outside the population cannot be acquired for GA, whereas for ES, it merely requires sampling the parameterized distribution. Conversely, GA handle multi-modal rewards better than ES due to the initialization requirements of each EA (§ 3.4). However, we note that GAs often suffer from premature convergence whereas ESs do not (§ 4.4).

### 3.4 Implementation and Computational Considerations

This section discusses practical considerations for implementing inference-time alignment using evolutionary algorithms, focusing on initialization strategies and parallel evaluation.

**Initialization and Sample Quality.** Initialization is essential for effective search. For GAs, initializing the population from the standard Gaussian distribution  $\mathcal{N}(0, I)$  ensures that the initial samples reside within the high-density shell of the latent space. However, this initialization strategy does not translate to natural evolutionary strategies (ES), as subsequent updates to the search distribution can degrade sample quality. To mitigate this, we instead initialize  $\mu$  of the search distribution to a fixed latent vector  $z_0 \sim \mathcal{N}(0, I)$  and use a small initial standard deviation. We elaborate further on consequences of ES initialization in Appendix C.

**Parallel Evaluation and Population Scaling.** Both GA and ES approaches evaluate a population of  $N$  candidate solutions at each iteration. To improve runtime efficiency, we partition each population into batches of size  $B$  and evaluate the batched solutions in parallel. Batching takes advantage of modern GPU parallelism and allows us to scale evaluations across larger populations without incurring excessive runtime costs. However, this comes at the cost of higher memory usage during model inference. In practice, the ability to choose the batch size for evaluation allows for a trade-off between memory usage and runtime. We explore this further in § 4.5.

## 4 Evaluation

We evaluate our noise optimization design decisions, and compare with other inference-time alignment work. § 4.1 describes our experimental setup. § 4.2 compares the ability of our method to maximize rewards with other work, notably Direct Noise Optimization [26] on the DrawBench and Open Image Preferences datasets. § 4.3 investigates the relationship between optimization steps and alignment. § 4.4 compares population statistics between ES and GA methods. § 4.5 reports the memory consumption and runtime of our method across population size and batch size.

### 4.1 Experimental Setup

We evaluate our alignment methods on a variety of diffusion models, benchmarks, and evolutionary algorithms. All experiments are conducted on NVIDIA A100 80GB GPUs.

**Evolutionary Algorithms** We use a variety of state-of-the-art evolutionary algorithms: CoSyne, PGPE, and SNES, as provided in EvoTorch [44].

**Diffusion Models** We implement EA-based search on three state-of-the-art diffusion models: Stable Diffusion-1.5 [34], Stable Diffusion 3 [9], and the latent consistency variant (LCM) of PixArt- $\alpha$  [3]. Our choices reflect the diversity of diffusion model architectures and training methods.

**Datasets** We follow [26], and conduct quantitative evaluations on *DrawBench* [35]. DrawBench consists of 200 prompts from 11 categories that test the semantic properties of model, such as their ability to handle composition, cardinality, and rare prompts. We conduct our ablation studies and qualitative evaluations on a subset (60 prompts, 5 per category) of the Open Image Preferences dataset [1]. Open Image Preferences consist of detailed and diverse prompts along with human

Table 1: **DrawBench Evaluation on StableDiffusion-1.5** We show the *best-sample* reward and *median* reward of the best population achieved by each algorithm on DrawBench under four reward functions. Algorithms (Random, Zero-Order, DNO) are baselines, while (CoSyNE, SNES, PGPE) are evolutionary methods. We conduct search using each algorithm for 15 steps with a population of 16 solutions, except for DNO which has population of 1. *NB:*  $\uparrow/\downarrow$  notation indicates a goal of reward maximization or minimization, respectively. Entries marked with a  $\dagger$  are statistically *insignificant* ( $p > 0.05$ ), otherwise they are statistically significant.

Algorithm	( $\uparrow$ ) ImageReward		( $\uparrow$ ) CLIP		( $\uparrow$ ) HPSv2		( $\downarrow$ ) JPEG Size (kB)	
	Best	Median	Best	Median	Best	Median	Best	Median
Random	1.41	0.38	37.1	32.9	0.301	0.282	66.9	105.4
Zero-Order	1.46	0.97	37.6	34.5	0.304	0.290	53.3	62.8
DNO	0.71	—	26.2	—	0.286	—	94.6	—
<i>Ours: Noise</i>								
CoSyNE	<b>1.61</b>	<b>1.38</b>	<b>38.8</b>	<b>36.5</b>	<b>0.310</b>	<b>0.300</b>	<b>39.5</b>	<b>44.8</b>
SNES	1.39 $\dagger$	0.78	38.1	35.6	0.300	0.286	71.3	78.0
PGPE	1.26	0.92	37.1 $\dagger$	34.4	0.299	0.290	88.3	97.0
<i>Ours: Noise Transformation (QR)</i>								
CoSyNE	<b>1.53</b>	<b>1.23</b>	<b>38.4</b>	<b>36.5</b>	<b>0.307</b>	<b>0.299</b>	38.7	42.3
SNES	1.34	0.94	37.4	34.7	0.300 $\dagger$	0.290	57.8	66.1
PGPE	1.28	0.88	36.9	34.8	0.300	0.290	<b>18.3</b>	<b>20.2</b>

preferences, which can be used to fine-tune diffusion models [6]. For our purposes, both datasets provide complex prompts to evaluate the quality of inference-time methods.

**Reward Functions** In accordance with other diffusion works [42, 23], we use ImageReward [53], CLIP [32], HPSv2 [51], and JPEG compressibility as reward functions and evaluation metrics.

## 4.2 Cross-Dataset Evaluation

**DrawBench.** Tab. 1 depicts the efficacy of EAs at alignment. After only 15 optimization steps, EAs outperform both baseline black-box methods (Random, Zero-Order) and the gradient-based DNO on all four reward functions, for both the *best* sample and the *median* reward in the population. When searching over noise, CoSyNE delivers the strongest overall performance, achieving the best alignment performance on ImageReward, CLIP, HPSv2 scores and yields the largest JPEG-size reduction. The same pattern holds when we search over noise transformations, with CoSyNE again leading three of the four metrics; the sole exception is the JPEG-compression objective, where PGPE attains the smallest file sizes. Manual inspection, however, reveals that many of PGPE’s images omit key prompt content, suggesting reward hacking (see Appendix D.5.1 for examples). Crucially, both EAs and the Zero-Order baseline improve *median* as well as *best* rewards, whereas Random search does not — indicating that directed search can steer us toward parts of the latent space where aligned samples reside. We elaborate on the consequences of searchable latent spaces in § 5.

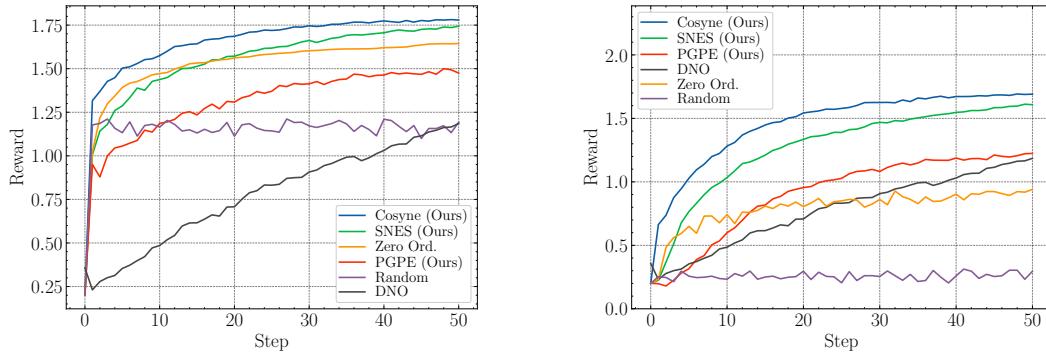
**Open Image Preferences.** Tab. 2 analyzes three diffusion models and confirms that evolutionary algorithms largely remain competitive across models. For both SD1.5 and SD3, the EA family consistently surpasses the black-box baselines: CoSyNE achieves the highest *best* and *median* rewards on every 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- $\alpha$ , where improvements vanish, suggesting that black-box latent search may not transfer to architectures trained with alternate objectives such as LCM — we elaborate in § 5. Overall, we show evolutionary search can align multiple diffusion models; however some latent spaces may be more difficult to search than others.

**Table 2: 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- $\alpha$ . Algorithms are configured in the same manner as in Tab. 1. Higher rewards are better.

Algorithm	SD1.5				SD3				PixArt- $\alpha$ (LCM)			
	CLIP		ImgReward		CLIP		ImgReward		CLIP		ImgReward	
	Best	Med.	Best	Med.	Best	Med.	Best	Med.	Best	Med.	Best	Med.
Random	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 <sup>†</sup>	36.1	1.67 <sup>†</sup>	1.29

Ours: Noise												
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 <sup>†</sup>	35.8	1.68 <sup>†</sup>	1.21
PGPE	39.0 <sup>†</sup>	36.1	1.37	0.88	38.4	36.4	1.67	1.50	38.7 <sup>†</sup>	35.8 <sup>†</sup>	1.65 <sup>†</sup>	1.23 <sup>†</sup>



(a) Best-sample reward per step on ImageReward.

(b) Median reward per step on ImageReward.

Figure 3: Reward per step measured on Open Image Preferences. Evolutionary methods (Cosyne, SNES, PGPE) outperform Zero-Order and Random baselines. Each algorithm optimizes the reward for 50 steps with a population of 16, except for DNO which has a population of 1.

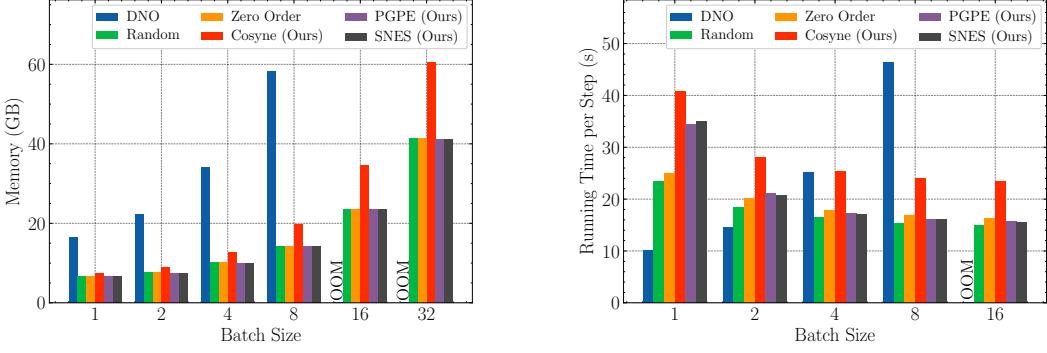
### 4.3 Optimization Steps and Alignment

We evaluate how the number of inference-time optimization steps affects alignment. Fig. 3 depicts the *best-sample* and *median* reward per optimization step on Open Image Preferences benchmark, using ImageReward alignment objective.

All methods improve their best-sample and median rewards as the number of steps increases, but they differ in step-efficiency. The population-based EA Cosyne achieves the highest step-efficiency, yielding the highest best-sample and median reward at every step count. Whereas, the distribution-based EAs, SNES and PGPE, require more steps to surpass random and zero-order search. SNES needs more than 15 than steps to outperform zero-order in best-sample performance. Meanwhile, PGPE needs more than 10 steps to exceed random search in best-sample performance. When considering median reward, all EAs exceed baselines given enough optimization steps.

### 4.4 Population Statistics

Fig. 4 plots the population’s reward standard deviation over 50 optimization steps for CoSyNE (a GA) and SNES (an ES), confirming the dynamics predicted in § 3.3. CoSyNE begins with high variance, reflecting broad exploration, but diversity collapses under selection pressure (Appendix D.4 gives ablations on selection pressures). In contrast, SNES sustains a moderate level of variance throughout optimization, preserving exploratory capacity even at later steps. This sustained diversity underpins the long-horizon gains of ES (Fig. 3a). Taken together, these results suggest that GAs are ideal for rapid, short-horizon alignment, while ES are preferable when more time is available.



(a) Memory usage vs. batch size ( $P = B$ ).

(b) Running time per step vs. batch size. ( $P = 16$ )

Figure 5: Computational characteristics of inference-time alignment methods on Stable Diffusion-1.5. In terms of memory usage evolutionary methods (CoSyne, PGPE, SNES) scale with batch size, whereas DNO does not and exhausts memory for batch sizes  $\geq 16$ . All methods aside from DNO exhibit near-constant or decreasing per-step latency as batch size increases. For DNO: ( $P = B$ )

#### 4.5 Computational Costs

We compare the memory usage and runtime per step for evolutionary methods in § 3 with Random, Zero-Order Search, and DNO. We measure the runtime per optimization step and the peak GPU memory usage with respect to batch size ( $B$ ) and population size ( $P$ ), as stated in § 3.4.

Fig. 5a shows that as batch size  $B$  increases, evolutionary methods consume significantly less GPU memory than DNO. The gradient-based DNO consumes  $2.2 \times$  to  $4.1 \times$  more GPU memory than evolutionary methods. Moreover, DNO runs out of memory (OOM) at batch sizes  $\geq 16$ , whereas evolutionary methods do not. Compared to other gradient-free approaches (Random and Zero-Order search), evolutionary methods consume only  $1 \times$  to  $1.5 \times$  more GPU memory. However, in terms of alignment performance (Tab. 1), evolutionary methods outperform both Random and Zero-Order search with small additional memory costs.

Fig. 5b shows the running time per step for each method as we vary the batch size  $B$ . All evolutionary algorithms (CoSyne, PGPE, SNES) exhibit a decrease in runtime per-step as  $B$  grows: at small  $B$ , the population of  $P$  candidates must be evaluated sequentially, thus runtime is high. As  $B$  increases, evaluations are batched and parallelized, reducing the time per step at the expense of additional memory usage as shown in Fig. 5a. In contrast, DNO’s per-step runtime increases with  $B$  and eventually exceeds all methods at large batch sizes.

## 5 Discussion and Limitations

**Searching Latent Spaces.** 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. 2). This raises two questions: (1) Are some latent spaces easier to search 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- $\alpha$  were diminished compared to other diffusion models (Tab. 2), suggesting some property of the latent space and/or diffusion model that affects our search efficacy.

**Limitations.** Our approach is ill-suited for long optimization horizons. With enough time, white-box methods that exploit gradient information will achieve better alignment (see Appendix D.2).

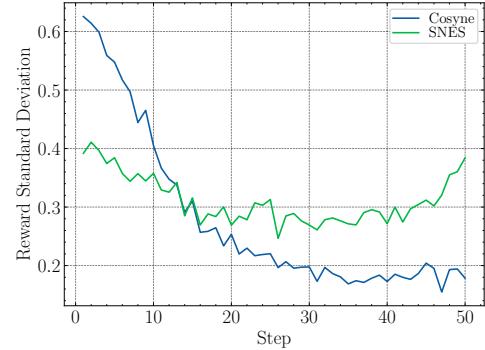


Figure 4: Reward standard deviation per step, measured on Open Image Preferences. CoSyne (GA) has high diversity early, but this rapidly diminishes. SNES (ES) has a more uniform pattern, as its search distribution adapts per step (§ 3.3). NB: Reward standard deviation proxies for solution diversity.

Black-box alignment fares better on short time-horizon applications, *e.g.*, resource-constrained tasks such as mobile image generation. It is less suitable in contexts with high resource budgets, *e.g.*, drug discovery. Even over short optimization horizons, our EA techniques shares the limit of other gradient-free and gradient-based methods (§ 2.1): some engineering time is incurred in tuning, and regularization is required to maintain sample quality (§ 3). We did not evaluate our method across modalities other than images, but we conjecture that our method could perform well across modalities.

## 6 Conclusion

We illustrate how inference-time alignment for diffusion models can be performed with evolutionary algorithms (EA). Specifically, we search for the initial noise vector used in the reverse denoising process in order to alter the sampled image and maximize some reward. Our results illustrate that EAs can achieve higher rewards than existing inference-time alignment methods across a set of various diffusion models and datasets. In terms of compute resource requirements, our EA-based method uses 55%-76% lower GPU memory and runs 72%-80% faster than gradient-based alignment methods. Our findings highlight the effectiveness of EAs for inference-time alignment via noise optimization.

## References

- [1] David Berenstein, Ben Burtenshaw, Daniel Vila, Daniel van Strien, Sayak Paul, Ame Vi, and Linoy Tsaban. Open preference dataset for text-to-image generation by the huggingface community. <https://huggingface.co/blog/image-preferences>, 2024. Accessed: 2025-04-15.
- [2] Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. *arXiv preprint arXiv:2305.13301*, 2023.
- [3] Junsong Chen, Jincheng Yu, Chongjian Ge, Lewei Yao, Enze Xie, Yue Wu, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu, et al. Pixart- $\alpha$ : Fast training of diffusion transformer for photorealistic text-to-image synthesis. *arXiv preprint arXiv:2310.00426*, 2023.
- [4] Kevin Clark, Paul Vicol, Kevin Swersky, and David J Fleet. Directly fine-tuning diffusion models on differentiable rewards. *arXiv preprint arXiv:2309.17400*, 2023.
- [5] Xiaoliang Dai, Ji Hou, Chih-Yao Ma, Sam Tsai, Jialiang Wang, Rui Wang, Peizhao Zhang, Simon Vandenhende, Xiaofang Wang, Abhimanyu Dubey, et al. Emu: Enhancing image generation models using photogenic needles in a haystack. *arXiv preprint arXiv:2309.15807*, 2023.
- [6] Data Is Better Together. open-image-preferences-v1-flux-dev-lora, 2024. URL <https://huggingface.co/data-is-better-together/open-image-preferences-v1-flux-dev-lora>. Accessed: 2025-04-17.
- [7] Urmila M Diwekar. *Introduction to applied optimization*, volume 22. Springer Nature, 2020.
- [8] Agoston E Eiben and James E Smith. *Introduction to evolutionary computing*. Springer, 2015.
- [9] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *International Conference on Machine Learning*, 2024.
- [10] Zhengcong Fei, Mingyuan Fan, and Junshi Huang. Gradient-free textual inversion. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 1364–1373, 2023.
- [11] Weixi Feng, Xuehai He, Tsu-Jui Fu, Varun Jampani, Arjun Akula, Pradyumna Narayana, Sugato Basu, Xin Eric Wang, and William Yang Wang. Training-free structured diffusion guidance for compositional text-to-image synthesis. *arXiv preprint arXiv:2212.05032*, 2022.
- [12] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H Bermano, Gal Chechik, and Daniel Cohen-Or. An image is worth one word: Personalizing text-to-image generation using textual inversion. *arXiv preprint arXiv:2208.01618*, 2022.
- [13] Faustino Gomez, Jürgen Schmidhuber, Risto Miikkulainen, and Melanie Mitchell. Accelerated neural evolution through cooperatively coevolved synapses. *Journal of Machine Learning Research*, 9(5), 2008.
- [14] Siyi Gu, Minkai Xu, Alexander Powers, Weili Nie, Tomas Geffner, Karsten Kreis, Jure Leskovec, Arash Vahdat, and Stefano Ermon. Aligning target-aware molecule diffusion models with exact energy optimization. *Advances in Neural Information Processing Systems*, 2024.

- [15] Yaru Hao, Zewen Chi, Li Dong, and Furu Wei. Optimizing prompts for text-to-image generation. *Advances in Neural Information Processing Systems*, 36:66923–66939, 2023.
- [16] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in Neural Information Processing Systems*, 35:8633–8646, 2022.
- [17] Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. Pick-a-pic: An open dataset of user preferences for text-to-image generation. *Advances in Neural Information Processing Systems*, 36:36652–36663, 2023.
- [18] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*, 2020.
- [19] Oliver Kramer. *Genetic Algorithm Essentials*, volume 679 of *Studies in Computational Intelligence*. Springer Berlin, 2017.
- [20] Akarsh Kumar, Chris Lu, Louis Kirsch, Yujin Tang, Kenneth O Stanley, Phillip Isola, and David Ha. Automating the search for artificial life with foundation models. *arXiv preprint arXiv:2412.17799*, 2024.
- [21] Kimin Lee, Hao Liu, Moonkyung Ryu, Olivia Watkins, Yuqing Du, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, and Shixiang Shane Gu. Aligning text-to-image models using human feedback. *arXiv preprint arXiv:2302.12192*, 2023.
- [22] Joel Lehman and Kenneth O Stanley. Evolving a diversity of virtual creatures through novelty search and local competition. In *Conference on Genetic and Evolutionary Computation*, pages 211–218, 2011.
- [23] Xiner Li, Masatoshi Uehara, Xingyu Su, Gabriele Scialia, Tommaso Biancalani, Aviv Regev, Sergey Levine, and Shuiwang Ji. Dynamic search for inference-time alignment in diffusion models. *arXiv preprint arXiv:2503.02039*, 2025.
- [24] Buhua Liu, Shitong Shao, Bao Li, Lichen Bai, Zhiqiang Xu, Haoyi Xiong, James Kwok, Sumi Helal, and Zeke Xie. Alignment of diffusion models: Fundamentals, challenges, and future. *arXiv preprint arxiv:2409.07253*, 2024.
- [25] Haohe Liu, Wenwu Wang, and Mark D Plumley. Latent diffusion model for audio: Generation, quality enhancement, and neural audio codec. In *Audio Imagination: NeurIPS 2024 Workshop AI-Driven Speech, Music, and Sound Generation*, 2024.
- [26] Nanye Ma, Shangyuan Tong, Haolin Jia, Hexiang Hu, Yu-Chuan Su, Mingda Zhang, Xuan Yang, Yandong Li, Tommi Jaakkola, Xuhui Jia, et al. Inference-time scaling for diffusion models beyond scaling denoising steps. *arXiv preprint arXiv:2501.09732*, 2025.
- [27] Oscar Mañas, Pietro Astolfi, Melissa Hall, Candace Ross, Jack Urbanek, Adina Williams, Aishwarya Agrawal, Adriana Romero-Soriano, and Michal Drozdzał. Improving text-to-image consistency via automatic prompt optimization. *arXiv preprint arXiv:2403.17804*, 2024.
- [28] Yibo Miao, Yifan Zhu, Lijia Yu, Jun Zhu, Xiao-Shan Gao, and Yinpeng Dong. T2vsafetybench: Evaluating the safety of text-to-video generative models. *Advances in Neural Information Processing Systems*, 37:63858–63872, 2024.
- [29] Risto Miikkulainen, Jason Liang, Elliot Meyerson, Aditya Rawal, Dan Fink, Olivier Francon, Bala Raju, Hormoz Shahrzad, Arshak Navruzyan, Nigel Duffy, et al. Evolving deep neural networks. In *Artificial intelligence in the age of neural networks and brain computing*, pages 269–287. Elsevier, 2024.
- [30] Brad L Miller, David E Goldberg, et al. Genetic algorithms, tournament selection, and the effects of noise. *Complex systems*, 9(3):193–212, 1995.
- [31] Mihir Prabhudesai, Anirudh Goyal, Deepak Pathak, and Katerina Fragkiadaki. Aligning text-to-image diffusion models with reward backpropagation. *arXiv preprint arXiv:2310.03739*, 2023.
- [32] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021.
- [33] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In *International Conference on Machine Learning*, pages 2902–2911, 2017.

- [34] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.
- [35] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasempour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022.
- [36] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.
- [37] Christoph Schuhmann. Laion-aesthetics dataset. <https://github.com/LAION-AI/laion-datasets/blob/main/laion-aesthetic.md>, 2022. Accessed: 2024-04-07.
- [38] Jimmy Secretan, Nicholas Beato, David B D Ambrosio, Adelein Rodriguez, Adam Campbell, and Kenneth O Stanley. Picbreeder: evolving pictures collaboratively online. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1759–1768, 2008.
- [39] Dan Simon. *Evolutionary optimization algorithms*. John Wiley & Sons, 2013.
- [40] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In *International Conference on Machine Learning*, 2015.
- [41] Felipe Petroski Such, Vashisht Madhavan, Edoardo Conti, Joel Lehman, Kenneth O Stanley, and Jeff Clune. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *arXiv preprint arXiv:1712.06567*, 2017.
- [42] Zhiwei Tang, Jiangweizhi Peng, Jiasheng Tang, Mingyi Hong, Fan Wang, and Tsung-Hui Chang. Inference-time alignment of diffusion models with direct noise optimization. *arXiv preprint arXiv:2405.18881*, 2024.
- [43] Yingtao Tian and David Ha. Modern evolution strategies for creativity: Fitting concrete images and abstract concepts. In *International conference on computational intelligence in music, sound, art and design (part of evostar)*, pages 275–291. Springer, 2022.
- [44] Nihat Engin Toklu, Timothy Atkinson, Vojtěch Micka, Paweł Liskowski, and Rupesh Kumar Srivastava. Evotorch: Scalable evolutionary computation in python. *arXiv preprint arXiv:2302.12600*, 2023.
- [45] Masatoshi Uehara, Yulai Zhao, Chenyu Wang, Xiner Li, Aviv Regev, Sergey Levine, and Tommaso Biancalani. Inference-time alignment in diffusion models with reward-guided generation: Tutorial and review. *arXiv preprint arXiv:2501.09685*, 2025.
- [46] Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.
- [47] Bram Wallace, Akash Gokul, Stefano Ermon, and Nikhil Naik. End-to-end diffusion latent optimization improves classifier guidance. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7280–7290, 2023.
- [48] Bram Wallace, Meihua Dang, Rafael Rafailov, Linqi Zhou, Aaron Lou, Senthil Purushwalkam, Stefano Ermon, Caiming Xiong, Shafiq Joty, and Nikhil Naik. Diffusion model alignment using direct preference optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8228–8238, 2024.
- [49] Chenyu Wang, Masatoshi Uehara, Yichun He, Amy Wang, Tommaso Biancalani, Avantika Lal, Tommi Jaakkola, Sergey Levine, Hanchen Wang, and Aviv Regev. Fine-tuning discrete diffusion models via reward optimization with applications to dna and protein design. *arXiv preprint arXiv:2410.13643*, 2024.
- [50] Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *The Journal of Machine Learning Research*, 15(1):949–980, 2014.
- [51] Xiaoshi Wu, Yiming Hao, Keqiang Sun, Yixiong Chen, Feng Zhu, Rui Zhao, and Hongsheng Li. Human preference score v2: A solid benchmark for evaluating human preferences of text-to-image synthesis. *arXiv preprint arXiv:2306.09341*, 2023.
- [52] Xiaoshi Wu, Keqiang Sun, Feng Zhu, Rui Zhao, and Hongsheng Li. Human preference score: Better aligning text-to-image models with human preference. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2096–2105, 2023.

- [53] Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation. *Advances in Neural Information Processing Systems*, 36:15903–15935, 2023.
- [54] Steven R Young, Derek C Rose, Thomas P Karnowski, Seung-Hwan Lim, and Robert M Patton. Optimizing deep learning hyper-parameters through an evolutionary algorithm. In *Workshop on Machine Learning in High-performance Computing Environments*, pages 1–5, 2015.
- [55] Zikai Zhou, Shitong Shao, Lichen Bai, Zhiqiang Xu, Bo Han, and Zeke Xie. Golden noise for diffusion models: A learning framework. *arXiv preprint arXiv:2411.09502*, 2024.

## A Overview of Appendices and Supplemental Material

- Appendix A: This summary.
- Appendix B: Extended commentary on related work § 2.1 and § 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 § 3.3.
- Appendix D: Extended quantitative and qualitative results across various models, reward functions, datasets, and algorithms used in our work, complementary to § 4.2—§ 4.4.

## 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.

### 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 [40, 16]. 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 [2].

Table 3: Comparison of inference-time diffusion alignment methods.

Method	Arbitrary Rewards	Gradient Free	Black-box
DOODL [47]	✗	✗	✗
DNO [42]	✓	✗	✗
DSearch [23]	✓	✓	✗
Search-Paths [26]	✓	✓	✗
Zero-Order [26]	✓	✓	✓
Ours	✓	✓	✓

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 method rely on curated datasets [37] or reward models [53, 52, 51] to fine-tune diffusion models [21, 5, 31, 4], and can involve supervised fine-tuning, or reinforcement learning based methods [48, 2]. 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  $\psi$  that maximizes the expected reward  $R(x)$  of samples  $x$  from a pretrained diffusion model  $p_\theta$ , 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,  $\psi$ , include optimizing conditional input prompts [15, 12, 27, 10], manipulating cross-attention layers [11], and tuning latent noise vectors (noise optimization) to

guide the diffusion trajectory [47, 42, 23, 45, 55, 26]. 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 [42], 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  $\sim 50$  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.* [26], 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.* [45] 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 [23], 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 (§ 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 § 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  $\psi$  that maximizes the expected reward, we assume that  $\psi$  is sampled from a parameterized search distribution,  $q_{\phi}$ , and we maximize the expected reward under this distribution. Note:  $x$  is either sampled from the solution space  $\psi$  in a stochastic manner or computed deterministically. In this context,  $\theta$  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,  $\psi_i$ , 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  $\psi_i$  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,  $\psi_i$  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.4).
4. **Mutation:** Offspring are perturbed with noise, often in an additive manner. E.g.  $x = x + \epsilon$ , where  $\epsilon \sim \mathcal{N}(0, \sigma)$ .  $\sigma$  is a hyperparameter of the genetic algorithm.

**Natural Evolutionary Strategies.** Natural evolutionary strategies perform black-box optimization by adapting the parameterized search distribution  $q_\phi$  over solutions [50]. Unlike GAs,  $q_\phi$  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  $\psi_i$  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  $\phi$  to increase expected reward along the natural gradient. Thus, we update the parameters of the search distribution  $q_\phi$  so that future samples are better aligned.

#### B.4 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 § 3.3. This is relevant specifically to the EvoTorch implementation of the CoSyne algorithm, which uses a uniform distribution to decide crossover behavior.

**Claim** 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  $Z \sim \mathcal{N}(0, I_d)$ .

*Proof.* Fix a coordinate  $i$ . For any real  $z$ , by the law of total probability conditioning on  $B_i$ :

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

where  $\Phi$  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,  $Z \sim \mathcal{N}(0, I_d)$ .  $\square$

**Implication** As noted in § 3.3, 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

Here, we review more details on how we initialized our search parameters. We also comment on consequences and insights related to choices of initialization.

**GA 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. 6 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 optimization steps, the ES algorithm may update  $\mu$  and  $\sigma$  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. 6.

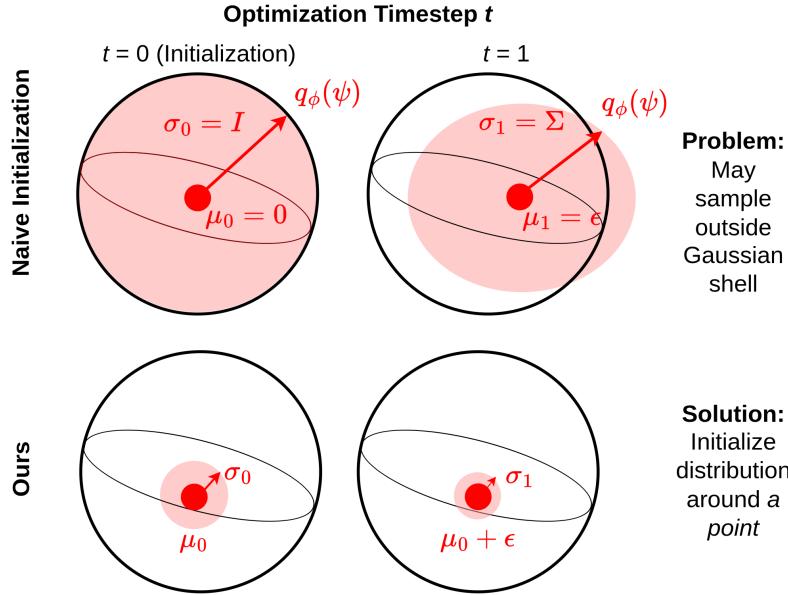


Figure 6: We depict a shortcoming with naive ES initialization, and illustrate our solution. Initializing the search distribution  $q_\phi(\psi)$  incorrectly (zero-mean, isotropic multivariate Gaussian) can quickly lead to sampling outside the Gaussian shell, leading to poor sample quality. We encountered 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  $\sigma_0$  to a localized region, rather than encapsulating the entire shell.

In order to address this problem, we instead randomly choose an initial  $\mu$  that exists on the Gaussian shell, and initialize the covariance  $\sigma$  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.

## D Extended Evaluation Results

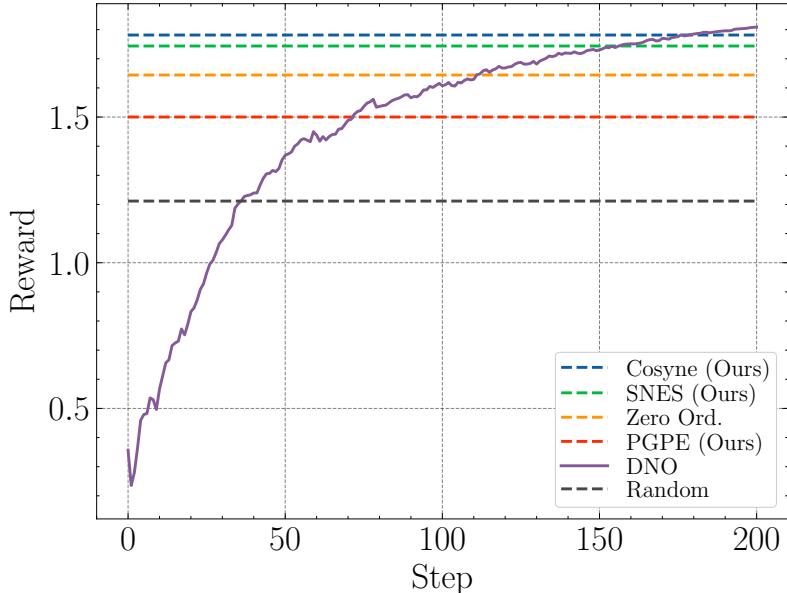
Here, we introduce additional quantitative and qualitative results.

### D.1 Significance Testing

We verify whether our results are statistically significant compared to unguided, random sampling of noise by performing Wilcoxon’s signed rank test; which assumes that distribution of pair differences is symmetric. We compare the resulting  $p$  value from Wilcoxon’s against the (standard) significance level  $\alpha = 0.05$ . Entries in Tab. 2 and Tab. 1 are annotated with a dagger  $^\dagger$  if they are statistically *insignificant*. Across both tables, only 8 out of 84 of the entries related to our method (CoSyne, SNES, PGPE) were insignificant.

### D.2 Effect of longer optimization timelines for gradient-based methods

Figures 7 to 9 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.



**Figure 7: DNO with Long Optimization Horizon:** Best-sample reward per step measured (ImageReward) as measured on Open Image Preferences. Horizontal lines are the maximum reward achieved by the gradient-free methods in **50** steps.

### D.3 Population Size Ablation

Figures 10 to 12 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. 13 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. 13b, this is in line with prior work characterizing the effect of tournament selection [30].

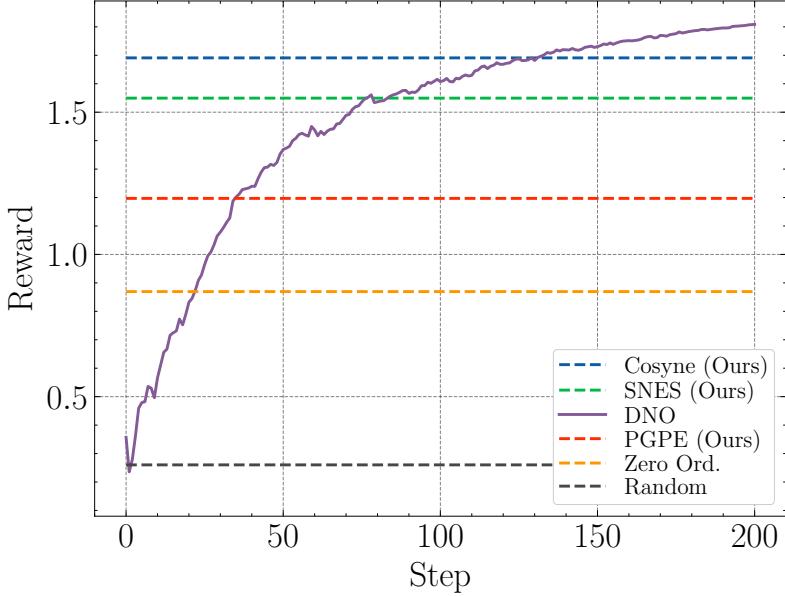


Figure 8: **DNO with Long Optimization Horizon:** Mean reward per step measured (ImageReward) as measured on Open Image Preferences. Horizontal lines are the maximum (mean) reward achieved by the gradient-free methods in **50** steps.

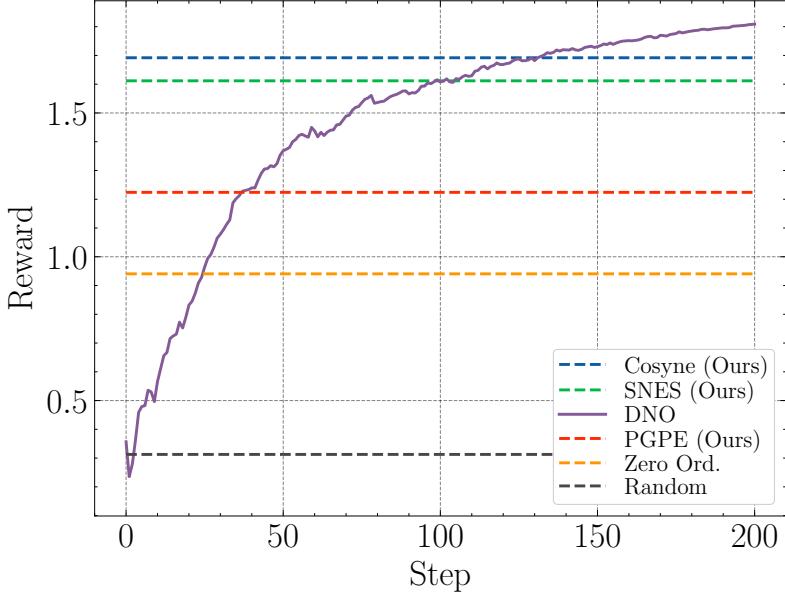


Figure 9: **DNO with Long Optimization Horizon:** Median reward per step measured (ImageReward) as measured on Open Image Preferences. Horizontal lines are the maximum (median) reward achieved by the gradient-free methods in **50** steps.

## 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.

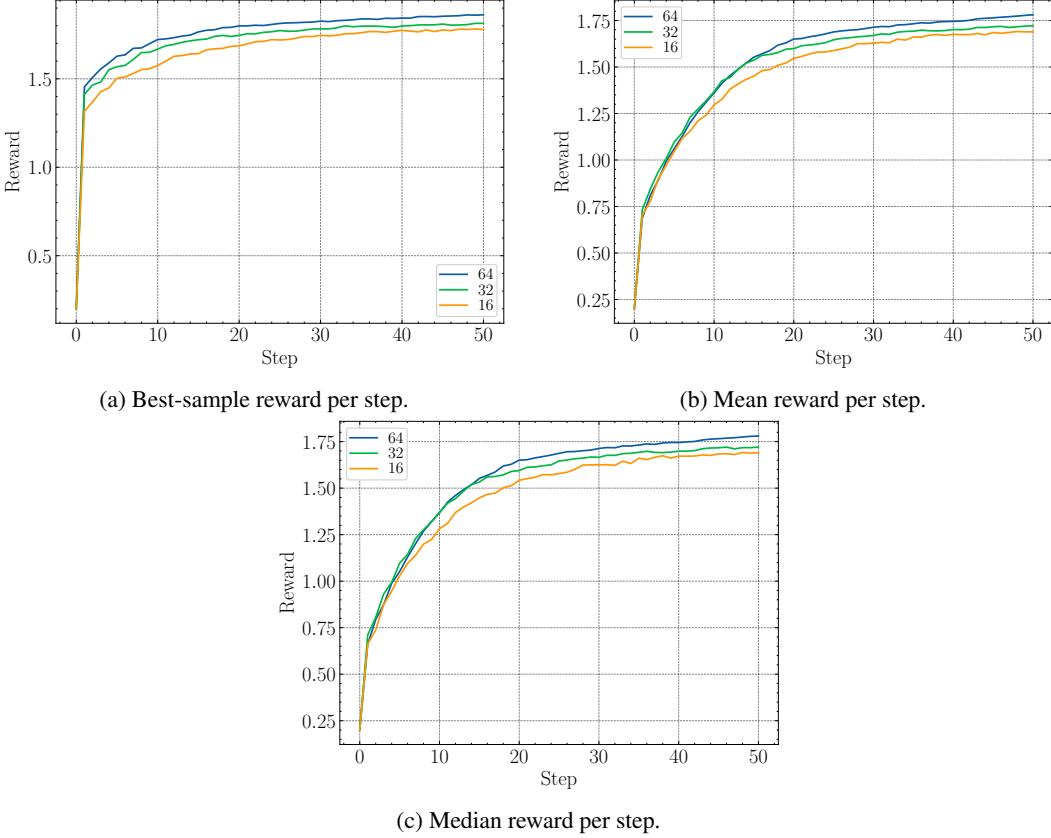


Figure 10: **Genetic Algorithm: CoSyNE** Reward statistics for the CosSyNE algorithm across optimization step.

### D.5.1 PGPE JPEG Reward-Hacking

Fig. 14 shows the proclivity of PGPE to "reward-hack" as noted in § 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 neglects the prompt, and instead produces nonsensical images to maximize reward.

### D.5.2 Stable Diffusion Qualitative Results

Here, we present sets of images from DrawBench and Open Image Preferences to illustrate qualitative results — Fig. 15, Fig. 16, Fig. 17, and Fig. 18. Specifically, we show how genetic algorithms such as CoSyne feature low sample diversity, even as optimization steps scale. This can also happen to ES methods (Fig. 18, 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 ( § 4.4).

## D.6 Extended DrawBench Results

Here, we include additional results on the DrawBench dataset. 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, which were measured across all prompts of the DrawBench dataset. We provide this data in Tab. 4, Tab. 5, Tab. 6, and Tab. 7.

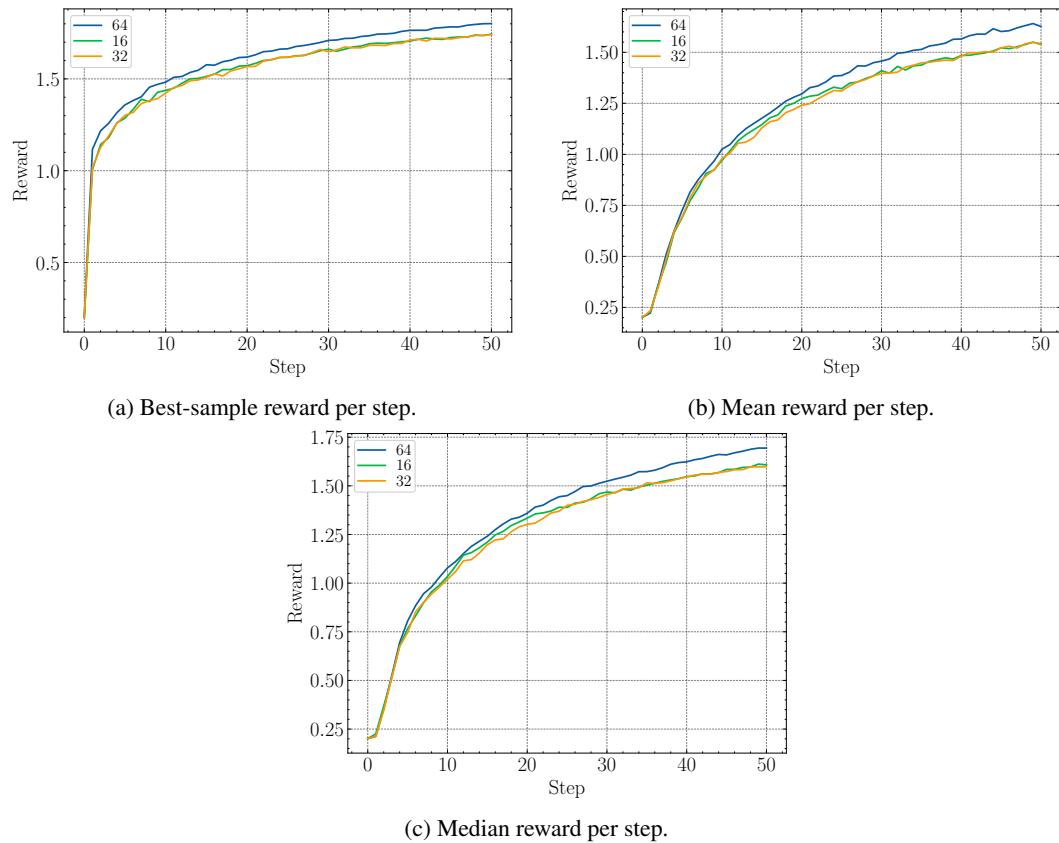


Figure 11: **Evolutionary Strategy: SNES** Reward statistics for the SNES algorithm across optimization step.

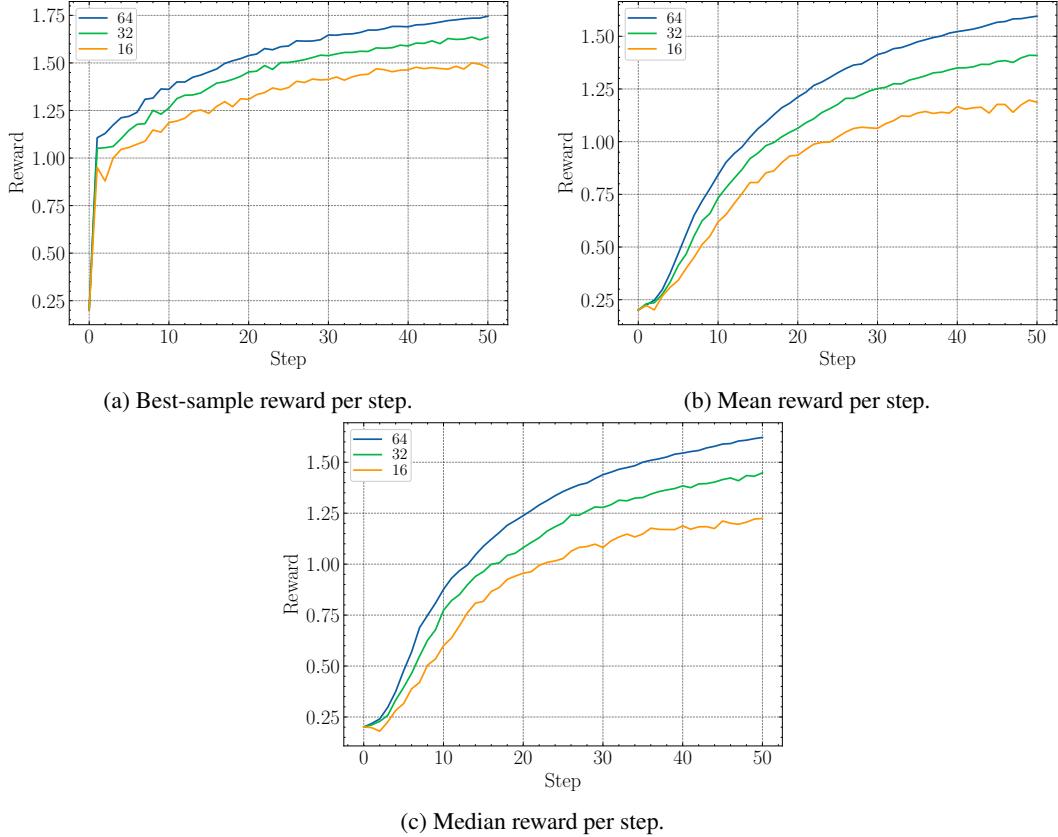


Figure 12: **Evolutionary Strategy: PGPE** Reward statistics for the PGPE algorithm across optimization step.

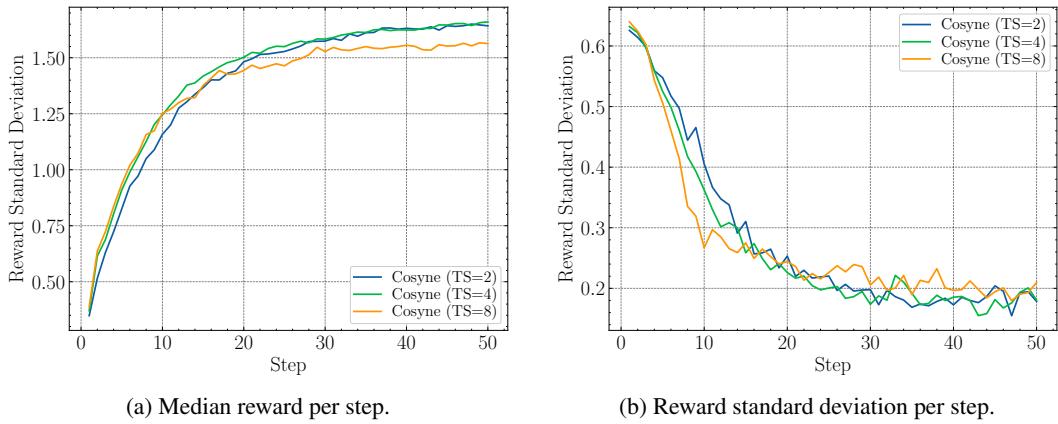


Figure 13: **Genetic Algorithm: CoSyne** Reward statistics per step, as measured on Open Image Preferences. Reward statistics are measured with increasing selection pressure, *i.e.*, increasing tournament sizes of 2, 4, and 8. We fix the population size to 16.

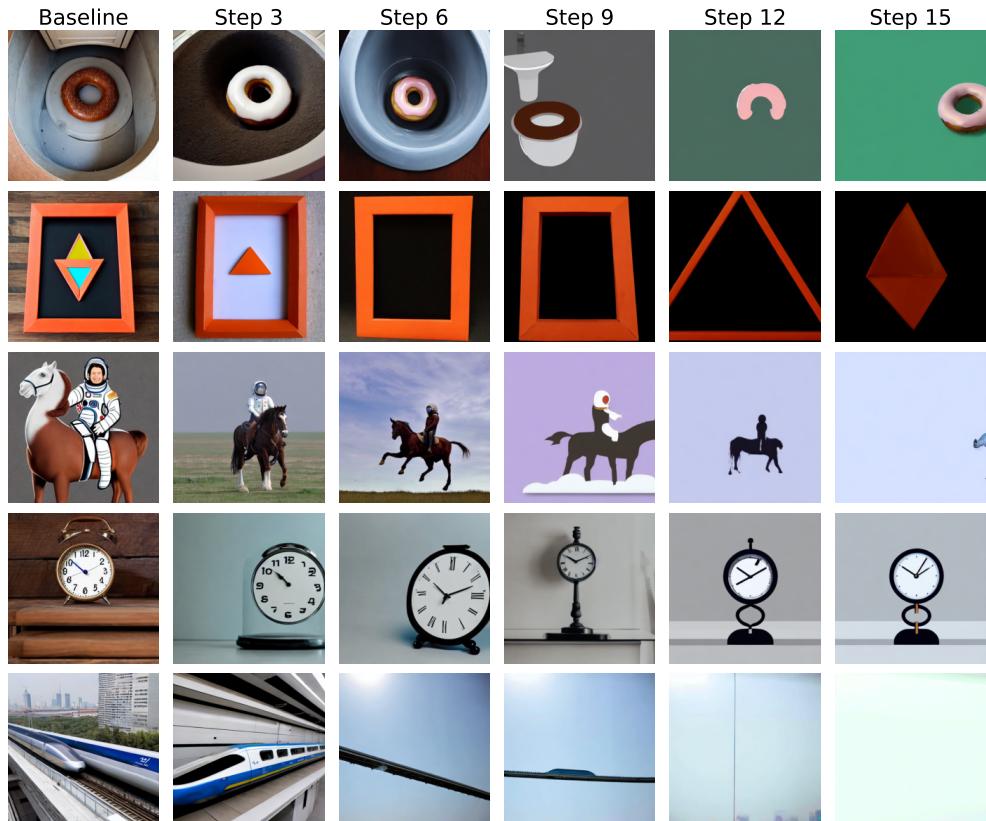


Figure 14: **PGPE Reward Hacking:** Randomly selected images created using inference-time alignment with the PGPE algorithm (transformation search) to minimize the JPEG size (DrawBench). PGPE neglects the prompt, and naively attempts to minimize JPEG file size by producing monochromatic or simple images.

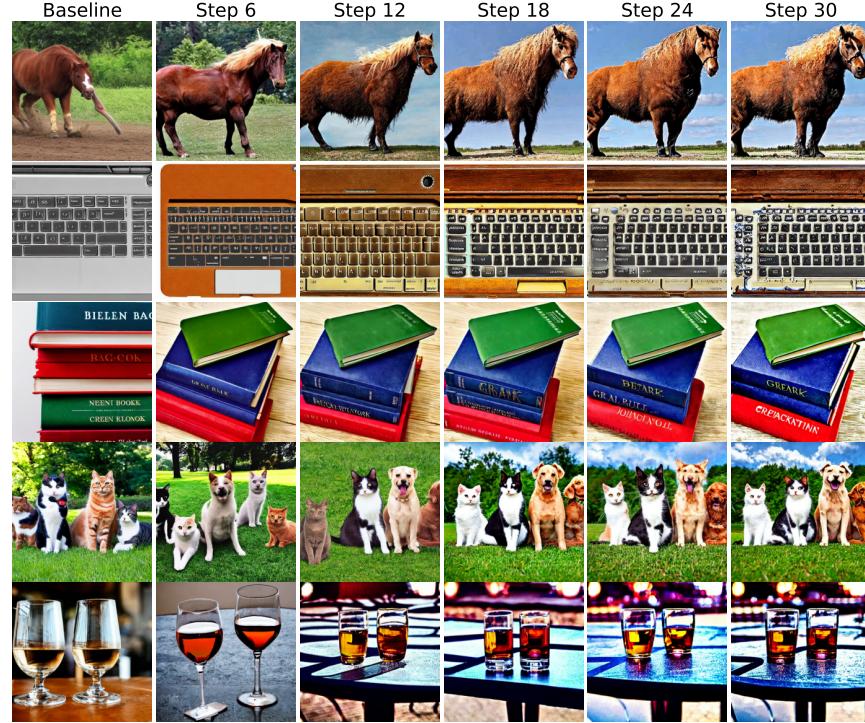


Figure 15: **Qualitative Sample Diversity:** Randomly selected DrawBench prompts evaluated on StableDiffusion-1.5 with ImageReward. CoSyne was used to perform alignment. Across optimization steps, the diversity between samples quickly diminishes. We identify this as a consistent occurrence with genetic algorithms such as CoSyne.

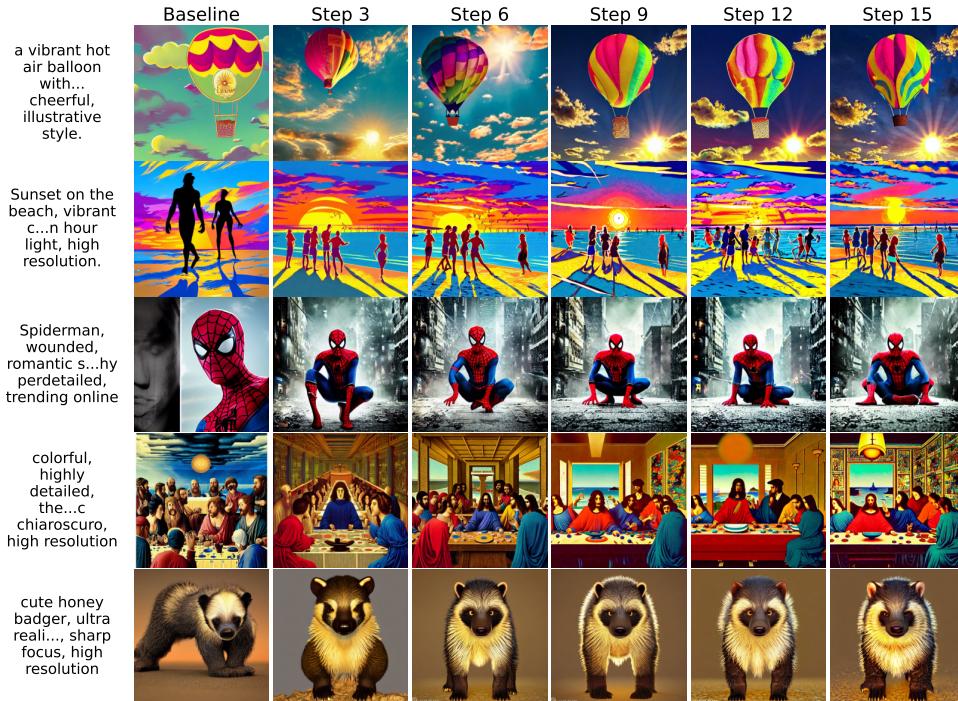


Figure 16: **Qualitative Sample Diversity:** Randomly selected Open Image Preference prompts evaluated on StableDiffusion-1.5 with ImageReward. CoSyne was used to perform alignment. Note the low sample diversity.

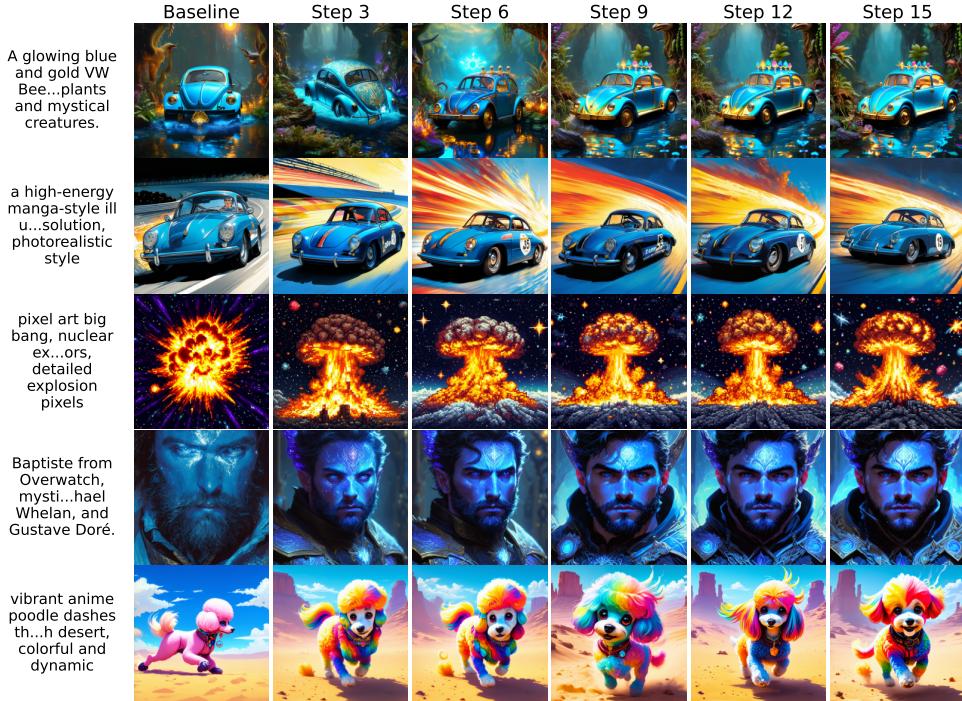


Figure 17: **Qualitative Sample Diversity:** Randomly selected Open Image Preference prompts evaluated on StableDiffusion-3 with ImageReward. CoSyne was used to perform alignment. This low sample diversity result on StableDiffusion-3 was also noticed on StableDiffusion-1.5.

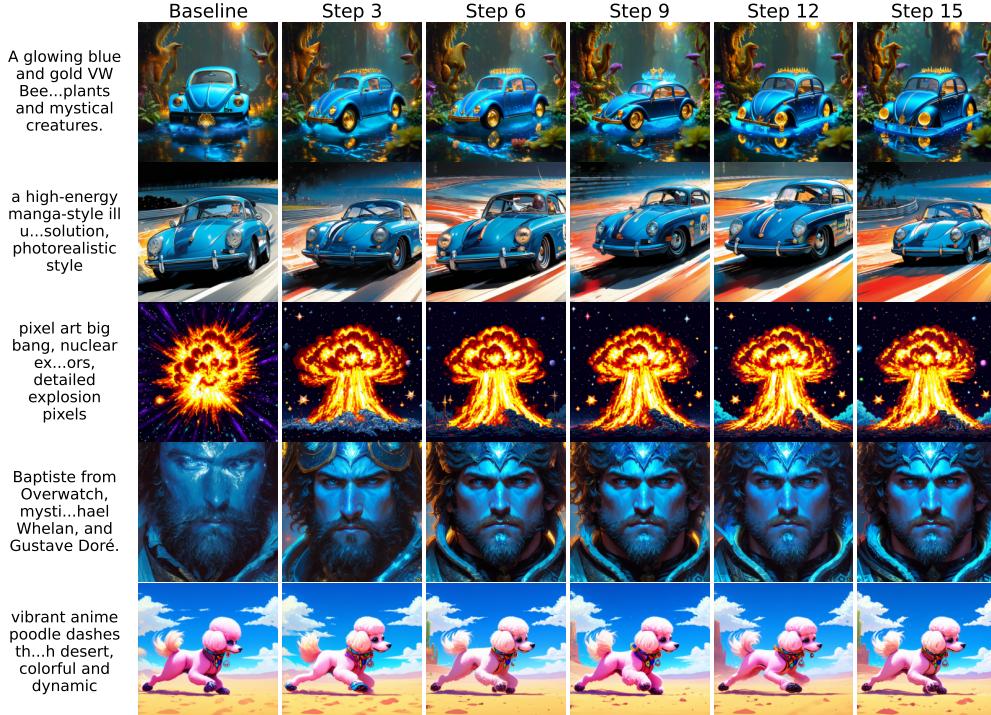


Figure 18: **Qualitative Sample Diversity:** Randomly selected Open Image Preference prompts evaluated on StableDiffusion-3 with ImageReward. SNES was used to perform alignment.

Table 4: Extended DrawBench results with ImageReward.

(a) Population-Best Reward

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

(b) Median Reward

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

Table 5: Extended DrawBench results with HPSv2.

(a) Population-Best Reward

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

(b) Median Reward

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

Table 6: Extended DrawBench results with CLIP.

(a) Population-Best Reward

<b>Algorithm</b>	<b>Soln. Space</b>	<b>Mean ± Std</b>	<b>Median</b>	<b>Min</b>	<b>Max</b>
Random	-	$37.139 \pm 3.665$	37.016	29.094	49.562
ZeroOrder	-	$37.590 \pm 3.864$	37.719	28.000	48.344
DNO	-	$26.237 \pm 3.638$	26.309	18.047	35.562
Cosyne	Noise	$38.791 \pm 3.812$	38.672	30.328	50.062
Cosyne	Rotation	$38.405 \pm 3.795$	37.938	30.125	50.188
SNES	Noise	$38.061 \pm 3.751$	37.875	29.094	49.562
SNES	Rotation	$37.407 \pm 3.831$	37.188	29.250	48.156
PGPE	Noise	$37.052 \pm 3.607$	36.656	29.250	47.844
PGPE	Rotation	$36.884 \pm 3.763$	36.609	28.562	48.875

(b) Median Reward

<b>Algorithm</b>	<b>Soln. Space</b>	<b>Mean ± Std</b>	<b>Median</b>	<b>Min</b>	<b>Max</b>
Random	-	$32.867 \pm 3.502$	32.812	20.562	41.750
ZeroOrder	-	$34.545 \pm 3.698$	34.438	24.578	44.938
DNO	-	$26.237 \pm 3.638$	26.309	18.047	35.562
Cosyne	Noise	$36.455 \pm 3.614$	36.141	28.219	46.938
Cosyne	Rotation	$36.393 \pm 3.712$	36.266	26.266	47.250
SNES	Noise	$35.623 \pm 3.563$	35.328	24.547	44.969
SNES	Rotation	$34.699 \pm 3.662$	34.438	26.594	45.812
PGPE	Noise	$34.397 \pm 3.455$	34.031	25.859	43.469
PGPE	Rotation	$34.772 \pm 3.602$	34.516	27.172	44.750

Table 7: Extended DrawBench results with JPEG File Size. Entries are file sizes listed in kilobytes (kB).

(a) Population-Best Reward

<b>Algorithm</b>	<b>Soln. Space</b>	<b>Mean ± Std</b>	<b>Median</b>	<b>Min</b>	<b>Max</b>
Random	-	$66.92 \pm 17.82$	64.23	26.55	136.58
ZeroOrder	-	$53.35 \pm 18.42$	51.47	15.89	116.90
DNO	-	$94.64 \pm 22.78$	91.78	47.11	174.45
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

<b>Algorithm</b>	<b>Soln. Space</b>	<b>Mean ± Std</b>	<b>Median</b>	<b>Min</b>	<b>Max</b>
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