# WHAT MAKES A SEARCH SPACE EFFECTIVE FOR EVOLUTION STRATEGIES IN DIFFUSION FINE-TUNING?

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Evolution strategies (ES) fine-tune generative models from reward evaluations alone, yet successful applications search very different spaces, from all weights of language models to about a hundred scaling factors of diffusion Transformers. Classical analyses of zeroth-order optimization favor small search spaces, but a smaller space also restricts what fine-tuning can change, and spaces of different sizes usually differ in how their parameters act on the network. It therefore remains unclear what makes a search space effective for ES on diffusion models. We separate these two factors in more than one hundred ES runs on SD3.5-Medium, FLUX.1-dev and Wan2.1-T2V, comparing search spaces at matched perturbation energy. Effectiveness depends more on how each parameter acts than on how many parameters there are. On SD3.5-Medium, growing residual scaling from 24 to 36,864 tunable parameters leaves the reward improvement unchanged, even as the agreement between independent update estimates drops by more than an order of magnitude. At 24 parameters, by contrast, scaling whole residual branches improves the reward by 1.68%, dense random mixtures of channels by 0.75% and random channel subsets by only 0.12%. ES search spaces can therefore be large, provided each parameter rescales a whole architectural unit. The improvement generalizes to held-out prompts and reproduces on other backbones, but unoptimized metrics stay flat or decline and optimizing a compositional objective fails, so the reward must encode every property that fine-tuning is meant to improve.

## 1 INTRODUCTION

Reward-based post-training has been used to improve diffusion and flow-matching models for image and video generation (Black et al., 2024; Clark et al., 2024; Liu et al., 2025; Xue et al., 2025; Zheng et al., 2026a). Methods such as Flow-GRPO and DiffusionNFT use scalar rewards while retaining gradient-based updates of the generative model (Liu et al., 2025; Zheng et al., 2026a). Evolution strategies (ES) instead sample parameter perturbations, evaluate the generated outputs, and combine the perturbations according to their rewards (Salimans et al., 2017). Needing only forward evaluations, ES suits settings in which the memory or implementation cost of backpropagation constrains tuning (Malladi et al., 2023; Sarkar et al., 2026).

Successful uses of ES and related zeroth-order methods search very different spaces. MeZO perturbs all weights of language models, EGGROLL perturbs their weight matrices through low-rank random factors (Malladi et al., 2023; Sarkar et al., 2026), and Calibri tunes about one hundred scaling coefficients of diffusion Transformers (Tokhchukov et al., 2026). The number of tunable parameters thus ranges from about one hundred to billions.

Classical analyses of zeroth-order optimization predict that the size of the search space matters. For a fixed number of random directions, the error of each gradient estimate grows with the number of parameters, so the number of steps grows with dimension (Nesterov and Spokoiny, 2017). Salimans et al. (2017) argued that the difficulty of ES depends instead on the intrinsic dimension of the problem, and MeZO proved dimension-free rates for fine-tuning language models under a low effective-rank assumption (Malladi et al., 2023). Whether these findings carry over to the search spaces used to fine-tune diffusion models has not been tested. A smaller space yields more reliable estimates but also restricts the changes the model can make (Li et al., 2018; Aghajanyan et al., 2021), and search



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

spaces of different sizes usually also differ in how each parameter acts on the network. This raises a fundamental question: **what makes a search space effective for ES on diffusion models?**

To answer this question, we separate the size of a search space from how its parameters act on the network. We vary the number of tunable parameters within one type of search space and compare different types at the same number of parameters. We train more than one hundred ES runs on SD3.5-Medium (Stability AI, 2024; Esser et al., 2024) with a composite reward of PickScore, HPSv2 and an aesthetic score (Kirstain et al., 2023; Wu et al., 2023; LAION AI, 2022), under the same budget and evaluation protocol for every search space. The search spaces scale the Transformer's residual branches block-wise, head-group-wise or channel-wise, restrict the channel scales to dense random subspaces or random channel subsets, or perturb the weights directly, with the same perturbation energy for all scaling spaces. We repeat the comparison of block-wise and channel-wise scaling on FLUX.1-dev (Black Forest Labs, 2024) and Wan2.1-T2V (Wan Team, 2025). On SD3.5-Medium, the final improvement barely changes with the number of tunable parameters but varies by more than an order of magnitude with the way each parameter acts on the network. Because a higher reward need not indicate a better model, we also evaluate held-out prompts, unoptimized metrics, within-prompt diversity and a compositional objective (Ghosh et al., 2023). Below, we summarize our key findings.

1. **High-dimensional search spaces remain effective.** In contrast to classical analyses, which predict that ES slows down in proportion to the number of parameters, raising the number of tunable parameters from 24 to 36,864 within residual scaling on SD3.5-Medium leaves the final reward improvement unchanged ($+1.68\%$ and $+1.71\%$). With more candidates per update, a 1,536-parameter dense random subspace keeps improving after block-wise scaling has saturated, and channel-wise scaling with tens of thousands of parameters also improves the reward on FLUX.1-dev and Wan2.1-T2V.
2. **Scaling whole residual branches is most effective.** With 24 tunable parameters and matched perturbation energy, block-wise scaling improves the validation reward by 1.68%, dense random mixtures of channels by 0.75% averaged over five draws, and random channel subsets, which concentrate the same energy on 24 of 36,864 channels, by only 0.12%. Perturbations that spread over the network along architectural units work best.
3. **The improvement is confined to the optimized reward.** It transfers to held-out DrawBench prompts (Saharia et al., 2022), by 0.79% for block-wise scaling, whereas unoptimized metrics stay flat or fall, within-prompt diversity decreases, and all five runs trained on the compositional GenEval objective end below the untouched model.

## 2 Preliminaries

### 2.1 Diffusion Transformers and reward fine-tuning

Our backbones, SD3.5-Medium, FLUX.1-dev and Wan2.1-T2V, are flow-matching models (Lipman et al., 2023; Liu et al., 2023; Esser et al., 2024) whose velocity field $v_\theta$ is a diffusion Transformer with weights $\theta$ (Peebles and Xie, 2023; Esser et al., 2024). Each block $b$ adds the output of a residual branch $f_b$, formed by attention and feed-forward layers, to the hidden state of the visual tokens,

$$h_{\text{out}}^{(b)} = h_{\text{in}}^{(b)} + f_b\big(h_{\text{in}}^{(b)}\big). \tag{1}$$

A deterministic sampler integrates $v_\theta$ over $K$ steps from an initial latent $\xi \sim \mathcal{N}(0, I)$ under a prompt $c$ and decodes the result into an image or video $y = S(\theta; c, \xi)$. Reward fine-tuning maximizes the expected score of a reward model $R(y, c)$, such as a preference model or a detector-based score,

$$J(\theta) = \mathbb{E}_{c,\,\xi}\big[R\big(S(\theta; c, \xi), c\big)\big]. \tag{2}$$

Gradient-based methods differentiate $R$ through the sampler or use likelihood ratios of its stochastic steps (Clark et al., 2024; Black et al., 2024; Liu et al., 2025), whereas ES only evaluates $R$.

### 2.2 Evolution strategies

Evolution strategies are black-box optimizers that ascend the objective averaged over a search distribution (Salimans et al., 2017; Nesterov and Spokoiny, 2017). For Gaussian perturbations $\sigma\epsilon$ and

a locally four-times differentiable $J$ with an integrable fourth-order remainder, this average is

$$J_\sigma(\theta) = \mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}\big[J(\theta+\sigma\epsilon)\big] = J(\theta) + \frac{\sigma^2}{2}\mathrm{Tr}\big(\nabla^2_\theta J(\theta)\big) + \mathcal{O}(\sigma^4). \tag{3}$$

Its gradient $\nabla_\theta J_\sigma(\theta) = \sigma^{-1}\mathbb{E}\big[J(\theta+\sigma\epsilon)\,\epsilon\big]$ is an expectation of reward values, so ES needs only forward evaluations, even for non-differentiable rewards. Antithetic pairs $\theta \pm \sigma\epsilon_i$ cancel the reward level and even-order terms when both arms share randomness, and a population of $G$ candidates gives $n = G/2$ directions and the raw estimator of one update, called a *generation*,

$$\hat{g} = \tfrac{1}{2\sigma n}\textstyle\sum_{i=1}^{n}\big[J(\theta+\sigma\epsilon_i) - J(\theta-\sigma\epsilon_i)\big]\,\epsilon_i. \tag{4}$$

### 2.3 Search spaces

ES need not perturb every weight. A search space restricts it to $d$ coordinates $u \in \mathbb{R}^d$, with $u = 0$ the pretrained model, and ES applies Equation 4 with $u$ in place of $\theta$ to maximize

$$F(u) = \mathbb{E}_{c,\,\xi}\big[R\big(S_u(c,\xi),c\big)\big], \tag{5}$$

where $S_u$ is the sampler under intervention $u$. For differentiable $F$, let $g_u = \nabla_u F(u)$. Except for low-rank perturbations (Sarkar et al., 2026), the search spaces we study are linear interventions that change gains or weights to $\omega(u) = \omega_0 + Au \in \mathbb{R}^m$ with a fixed $A \in \mathbb{R}^{m\times d}$. A search space is thus characterized by its dimension $d$ and by the map $A$ that specifies how each coordinate acts on the network, and Section 3 instantiates both. We standardize the $2n$ rewards of each generation to z-scores $a_i^{\pm}$, absorb $1/\sigma$ into the step size $\alpha$, and update $u$ by

$$\Delta u = \tfrac{\alpha}{2n}\textstyle\sum_{i=1}^{n}\big(a_i^+ - a_i^-\big)\,\epsilon_i, \tag{6}$$

which rescales the raw estimator by a positive factor. GenEval runs use rank shaping instead.

### 2.4 Dimension and estimator alignment

Proposition 1 formalizes the classical argument for small search spaces.

**Proposition 1** (Second moment of the estimator)**.** *Let $F$ be differentiable at $u$, with $g_u = \nabla_u F(u)$, and draw $n = G/2$ independent directions $\epsilon_i \sim \mathcal{N}(0, I_d)$. Define $\hat{g}_u = n^{-1}\sum_{i=1}^{n}(g_u^\top\epsilon_i)\epsilon_i$, the almost-sure limit of the raw estimator in these coordinates as $\sigma \to 0$. Then*

$$\mathbb{E}\big[\hat{g}_u\big] = g_u, \qquad \mathbb{E}\big\|\hat{g}_u\big\|^2 = \|g_u\|^2\,\frac{d+n+1}{n}. \tag{7}$$

For $d > 1$ and $g_u \neq 0$, the expected cosine of $\hat{g}_u$ with $g_u$ is

$$\mathbb{E}[\cos(\hat{g}_u, g_u)] = \frac{\Gamma((n+1)/2)\Gamma((n+d-1)/2)}{\Gamma(n/2)\Gamma((n+d)/2)} \;\approx\; \sqrt{\frac{n}{n+d}}, \tag{8}$$

where the gamma ratio is exact and the approximation follows from concentration. This prediction concerns each update, not the final reward. For an $L_F$-smooth $F$, an optimally sized step guarantees an expected increase of at least $\|g_u\|^2/[2L_F(1+(d+1)/n)]$, and changing $A$ changes $g_u$ and $L_F$ along with $d$. Local progress therefore depends on how the coordinates act on the network as well as on how many there are. Appendix D contains the proofs.

**Split-half agreement.** Because $g_u$ is unknown, we measure how well two independent estimates agree. At fixed prompts and latents, we repeatedly split one evaluated set of directions into disjoint halves of size $n$ with normalized estimates $U_A$ and $U_B$. The halves are i.i.d., so $\mathbb{E}[U_A^\top U_B] = \|\mathbb{E}[U]\|^2$, and we report the square root of this split-half agreement, which measures consistency rather than alignment with the true gradient. If dimension limited ES through estimation noise, this agreement should predict final reward.

## 3 Search spaces and perturbation matching

Section 2 characterizes a search space by its dimension $d$ and its map $A$. To separate their effects, we construct search spaces that change one of the two at a time. Residual scaling varies $d$ while every coordinate keeps rescaling channels of a single residual branch, random subspaces of the channel gains vary $A$ at fixed $d$, and weight-space families are the high-dimensional spaces of prior zeroth-order fine-tuning. Perturbation matching then gives every scaling space the same perturbation energy, so that differences in reward reflect $d$ and $A$ rather than the size of the perturbation.

### 3.1 RESIDUAL SCALING

Scaling a residual branch changes how strongly it contributes without altering what it computes, and Calibri improves diffusion Transformers by tuning about one hundred such coefficients with CMA-ES (Tokhchukov et al., 2026). SD3.5-Medium (Stability AI, 2024) has $L = 24$ transformer blocks, each with $H = 24$ attention heads and $C = 1536$ channels. We scale the residual branch $f_b$ of Equation 1 by a gain vector $s^{(b)} \in \mathbb{R}^w$ initialized to one, where the width $w$ is the number of channel groups:

$$h_{\text{out}}^{(b)} = h_{\text{in}}^{(b)} + \left(s^{(b)} \otimes \mathbf{1}_{C/w}\right) \odot f_b\left(h_{\text{in}}^{(b)}\right), \tag{9}$$

where $\odot$ broadcasts over tokens. Widths $w = 1, H, C$ give block, head-group and channel gains with $d = Lw = 24$, 576 and 36,864, and a head-group gain scales $C/H$ contiguous channels. The partitions are nested, since block and head-group gains are channel gains tied within blocks or groups, so each larger space contains every solution of the smaller ones.

### 3.2 RANDOM SUBSPACES

Random subspaces are the standard tool for measuring the intrinsic dimension of an objective (Li et al., 2018; Aghajanyan et al., 2021), and we use them to vary how coordinates combine channels at fixed $d$. Within the full channel-gain vector $s \in \mathbb{R}^{LC}$, a $d$-dimensional subspace sets $s = \mathbf{1} + (LC/d)^{1/2} Q_d u$ with $Q_d^\top Q_d = I_d$. A *dense random subspace*, labelled "rotation" in tables and figures, uses the first $d$ columns of one fixed random orthogonal matrix, so every coordinate moves all channels and subspaces of different $d$ are nested. A *random channel subset*, labelled "sparse", uses $d$ random columns of the identity, so each coordinate moves one channel and all other gains stay at one. At $d = 24$, block gains, dense subspaces and channel subsets thus differ only in $A$.

### 3.3 WEIGHT-SPACE FAMILIES

Prior zeroth-order fine-tuning of large models perturbs the weights directly, all of them in MeZO and each weight matrix through low-rank random factors in EGGROLL (Malladi et al., 2023; Sarkar et al., 2026). We include both kinds as high-dimensional baselines. Direct weight search sets $\theta(u) = \theta_0 + E\Lambda u$, so that $A = E\Lambda$, where $E \in \mathbb{R}^{D \times d}$ embeds selected entries of the weights $\theta \in \mathbb{R}^D$ and the diagonal $\Lambda$ scales each by the RMS of its tensor $W^{(l)}$,

$$\Lambda_{jj} = \mathrm{RMS}\big(W^{(l)}\big) = \sqrt{\tfrac{1}{|W^{(l)}|} \textstyle\sum_k \big(W_k^{(l)}\big)^2}, \tag{10}$$

where $|W^{(l)}|$ counts its entries. Weight perturbations are stated in this tensor-relative scale $\sigma_{\text{rel}}$. Matrix-weight perturbations cover the selected tensors of order two and above, 1.46 billion of the backbone's 2.24 billion parameters. Low-rank perturbations instead draw each such tensor's direction as $U^{(l)} V^{(l)\top} / \sqrt{r}$ from rank-$r$ standard normal factors (Sarkar et al., 2026), here with $r = 1$. Both families use their own $\sigma_{\text{rel}}$ schedules and step size 0.005 and are reported separately.

### 3.4 PERTURBATION MATCHING

At equal coordinate $\sigma$, search spaces can change the network by very different amounts, which would confound their structure with the size of their perturbations. We therefore match their perturbation energy. For a fixed residual output $f_b$ in Equation 9, independent gain-coordinate perturbations with standard deviation $\sigma$ satisfy $\mathbb{E}\|(\delta s^{(b)} \otimes \mathbf{1}_{C/w}) \odot f_b\|^2 = \sigma^2 \sum_j f_{b,j}^2$ for every width $w$. Equal coordinate $\sigma$ thus matches this change across widths and gives every partition the same perturbation energy $\mathbb{E}\|\delta s\|^2 = LC\sigma^2$ of the expanded gain change $\delta s$. Subspaces reach the same energy through the factor $(LC/d)^{1/2}$. Scaling $\sigma$ by $1/\sqrt{d}$ to match the Euclidean norm in coordinate space would instead make channel-gain perturbations about 39 times smaller than block-gain ones, with relative activation changes of 0.028 against 0.668. Relative to that scaling, our matching narrows the max-to-min ratio of measured changes across five spaces from 31.6× to 1.50× in activations, from 28.4× to 1.61× in pixels and from 25.0× to 1.47× in reward differences. The remaining spread comes from which channels move together, which Section 4.3 compares at fixed dimension. At matched energy, the per-channel standard deviation is about $\sigma$ for partitions and dense subspaces, whereas a random

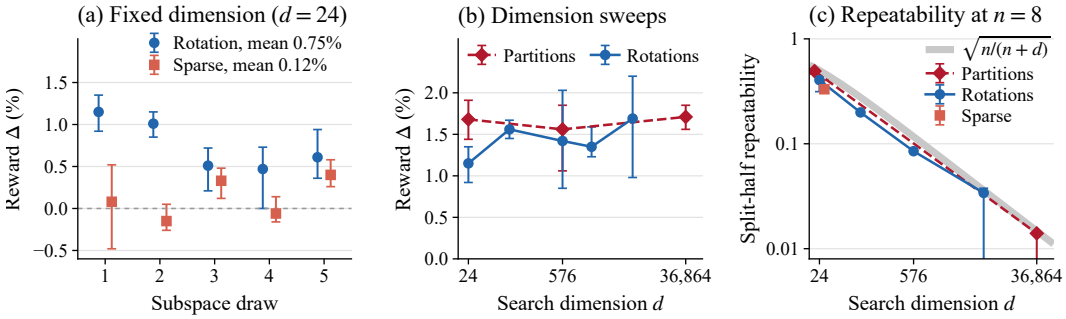


Figure 1: Validation change and split-half agreement on SD3.5-Medium at $G = 16$ ($n = 8$) with per-channel RMS matched. Points in (a) and (b) are three-seed means with observed seed ranges. (a) Five subspace draws per random family at $d = 24$, with legend values averaging the draws. (b) Nested partitions and dense subspaces of the initial draw against $d$. (c) Square-root split-half agreement against the dimension-only approximation $\sqrt{n/(n+d)}$, with markers from each space's main probe and bars spanning independent probes. Bars reaching the axis include a probe with a negative mean cosine.

channel subset concentrates the energy on its $d$ channels, each with standard deviation $\sigma(LC/d)^{1/2}$, about $39\sigma$ at $d = 24$. Appendix F gives the derivations and measurements.

# 4 Experiments

## 4.1 Setup

**Models and protocol.** All SD3.5 runs use the SD3.5-Medium checkpoint (Stability AI, 2024) in fp32 at $512 \times 512$, with a 25-step Euler ODE sampler and classifier-free guidance 4.5. Each of the $T = 100$ generations draws $B = 8$ training prompts and their initial latents, shared by all $G = 16$ candidates. With a deterministic sampler and scorer, this sharing removes latent-sampling noise from paired reward differences while retaining variation across prompt batches, and repeated evaluations return bitwise identical rewards. One antithetic pair per direction therefore suffices, whereas agentic ES methods typically average each direction over 16 to 32 tasks (Zheng et al., 2026b). Training prompts are the first 448 rows of the PickScore training split, and the next 64 rows form the validation panel. Gain spaces share the step size $\alpha = 0.05$ and a cosine schedule for $\sigma$ from 0.1 to 0.02. We repeat the block- and channel-gain comparison on FLUX.1-dev (Black Forest Labs, 2024) and Wan2.1-T2V (Wan Team, 2025) under their own fixed protocols, with each video scored by the mean reward of its frames. Search spaces are compared only within a backbone, where they share the budget and evaluation protocol. Appendix E lists every setting, scorer checkpoint and seed.

**Objective and evaluation.** Our objective is the sum of PickScore (Kirstain et al., 2023), HPSv2 (Wu et al., 2023) and the LAION aesthetic predictor (LAION AI, 2022), and we also report the change of each term. CLIP image–text similarity (Radford et al., 2021) and ImageReward (Xu et al., 2023) serve as unoptimized metrics, and we separately optimize the detector-based GenEval reward (Ghosh et al., 2023). The validation panel is scored every ten generations with one fixed latent per prompt. Unless stated otherwise, we report the relative change in validation reward from the untouched model, averaged over the final three evaluations, as a mean over three seeds with the observed seed range. A DrawBench panel (Saharia et al., 2022), fixed before training, provides a held-out comparison.

## 4.2 Does ES require a low-dimensional search space?

Within residual-scaling search spaces, increasing the search dimension from 24 to 36,864 does not reduce the final validation reward on SD3.5-Medium, even though the agreement between independent ES update estimates falls by more than an order of magnitude. We vary dimension within two nested families. Across the 1,536-fold increase from block through head-group to channel gains, the final validation changes are $+1.68\%$, $+1.56\%$ and $+1.71\%$ (Figure 1b, Table 1), and

Table 1: Final validation change on SD3.5-Medium as three-seed means with observed seed ranges. Partitions and dense subspaces are nested in $d$, and random subspaces use the initial draw. Gain spaces match per-channel RMS except the "equal $\sigma$" rows, which omit the subspace factor. Weight spaces use their own scales and are listed by entry count or rank.

| family | search space | $d$ | reward $\Delta$ (%) | seed range |
|---|---|---|---|---|
| partition | block gain | 24 | +1.68 | [+1.44, +1.91] |
| | head gain | 576 | +1.56 | [+1.06, +1.85] |
| | channel gain | 36,864 | +1.71 | [+1.56, +1.85] |
| rotation | rotation $d$=24 | 24 | +1.15 | [+0.92, +1.35] |
| | rotation $d$=96 | 96 | +1.56 | [+1.45, +1.67] |
| | rotation $d$=576 | 576 | +1.42 | [+0.85, +2.03] |
| | rotation $d$=1,536 | 1,536 | +1.35 | [+1.23, +1.59] |
| | rotation $d$=6,144 | 6,144 | +1.69 | [+0.98, +2.20] |
| sparse | sparse $d$=24 (equal $\sigma$) | 24 | +0.08 | [+0.01, +0.14] |
| | sparse $d$=24 | 24 | +0.08 | [-0.48, +0.52] |
| | sparse $d$=576 (equal $\sigma$) | 576 | +0.06 | [+0.04, +0.07] |
| lowrank | low-rank $r$=1 | $r$=1 | -0.12 | [-0.25, -0.01] |
| weight | matrix weights | 1,460,000,000 | -0.05 | [-0.11, +0.04] |

every channel-gain seed, from $+1.56\%$ to $+1.85\%$, lands inside the block-gain seed range. Nested dense random subspaces with $d = 24$, 96, 576, 1,536 and 6,144 reach $+1.15\%$, $+1.56\%$, $+1.42\%$, $+1.35\%$ and $+1.69\%$, and every seed of every space with $d \geq 96$ improves the reward by at least $0.85\%$. Because a larger space loses no reachable solution, a decline with dimension would reflect the cost of estimating updates in more coordinates, and we observe no consistent decline.

**Agreement falls as predicted while reward holds.** Estimates in larger spaces agree less, as Proposition 1 predicts. We measure split-half agreement at the untouched model on one fixed training batch with 256 pre-evaluated random directions. At the training population, $n = 8$, its square-root statistic falls from $0.493$ for block gains to $0.014$ for channel gains, close to the predicted $0.500$ and $0.015$ of Equation 8 (Figure 1c), and at $d = 6{,}144$ and $36{,}864$ independent probes cannot distinguish it from zero. For block gains, the statistic also follows the prediction as the population grows, from $0.224$ at $n = 2$ to $0.914$ at $n = 128$ against predicted values of $0.277$ and $0.918$. All spaces with $d \geq 96$ nevertheless finish between $+1.35\%$ and $+1.71\%$, so the loss of agreement in each update does not become a loss of reward.

**Larger spaces reach the same reward through their own solutions.** The block-mean component carries $0.08\%$ of the displacement energy of learned channel gains and $3.5\%$ of that of head-group gains, close to the $0.07\%$ and $4.2\%$ of an isotropic displacement. Channel-gain solutions from different seeds are nearly uncorrelated, at $0.11$ for block means and $0.002$ within blocks, although each improves the reward as much as block gains do (Appendix Table 19). A large space therefore holds many solutions of similar quality, and noisy updates still reach one of them.

**Larger spaces benefit more from larger populations.** Keeping 100 generations, we vary the population $G$ for block gains and for the 1,536-dimensional dense subspace (Table 2). Block gains rise from $+1.42\%$ at $G = 8$ to $+1.96\%$ at $G = 32$ and level off at $+1.89\%$, whereas the dense subspace keeps rising, from $+1.19\%$ to $+2.12\%$, the largest improvement of any SD3.5-Medium run, and overtakes block gains at $G = 64$. A small space thus benefits from reliable estimates when queries are scarce, and a large space from its extra capacity when they are plentiful.

**High dimension also succeeds on other backbones and held-out prompts.** On FLUX.1-dev and Wan2.1-T2V, channel gains with 58,368 and 46,080 coordinates improve the reward by $+2.02\%$ and $+1.36\%$ (Table 6), and on held-out DrawBench prompts the 6,144-dimensional dense subspace and channel gains match block gains (Table 4). The rank-one and matrix-weight spaces lie off this axis, because their scale protocol and the way each direction acts on the network change together with the number of coordinates.

Table 2: Three-seed mean validation change against population size $G$ for block gains ($d = 24$) and a dense random subspace ($d = 1{,}536$). All runs use 100 generations, so total reward queries grow in proportion to $G$.

| space | $G$ | $n = G/2$ | reward $\Delta$ (%) |
|---|---|---|---|
| block, $d$=24 | 8 | 4 | +1.42 |
| | 16 | 8 | +1.68 |
| | 32 | 16 | +1.96 |
| | 64 | 32 | +1.89 |
| rotation, $d$=1,536 | 8 | 4 | +1.19 |
| | 16 | 8 | +1.35 |
| | 32 | 16 | +1.50 |
| | 64 | 32 | +2.12 |

Table 3: Dense random subspaces against random channel subsets at $d = 24$. Upper rows average five subspace draws per family, with nested 95% intervals that resample draws and then optimizer seeds. Lower rows use the initial draw and one optimizer seed and halve or double $\alpha$ or both ends of the $\sigma$ schedule, $\sigma_0$ and $\sigma_T$.

| comparison at $d = 24$ | rotation (%) | sparse (%) | gap (points) |
|---|---|---|---|
| five draws, 15 runs per family | +0.75 | +0.12 | +0.63 |
| nested 95% interval | [+0.49, +1.01] | [-0.09, +0.34] | [+0.28, +0.97] |
| initial draw, seed 1234 | +1.35 | +0.52 | +0.83 |
| $\sigma_0$ and $\sigma_T$ halved | +0.30 | +0.19 | +0.11 |
| $\sigma_0$ and $\sigma_T$ doubled | +1.31 | +0.55 | +0.75 |
| $\alpha$ halved | +1.01 | +0.15 | +0.85 |
| $\alpha$ doubled | +0.80 | -0.17 | +0.97 |

### 4.3 Which search space is most effective at a fixed dimension?

At fixed dimension and matched perturbation energy, the outcome depends on how each search direction acts on the network. Scaling whole residual branches works best, dense random mixtures of channels work less well, and concentrating the same energy on a few channels achieves almost nothing. At $d = 24$, block gains improve validation reward by $+1.68\%$. Over five subspace draws with three optimizer seeds each, dense random subspaces average $+0.75\%$ $[+0.49, +1.01]$ and random channel subsets $+0.12\%$ $[-0.09, +0.34]$ (Table 3, Figure 1a). The dense family thus leads by $+0.63$ percentage points (nested 95% interval $[+0.28, +0.97]$). Direct weight perturbations, run in fp32 with their own scale and step size, yield no improvement at this budget, with $-0.12\%$ for rank-one perturbations and $-0.05\%$ for matrix-weight perturbations of 1.46 billion entries (Table 1).

**The ordering holds across draws, seeds and hyperparameters.** Leaving out any one draw from both families keeps their gap between $+0.50$ and $+0.74$ points, with every interval above zero (Appendix H). Each block-gain run exceeds all 15 dense-subspace runs, and against the initial draw block gains lead the dense subspace and the channel subset by $+0.53$ and $+1.60$ points, positive for every seed (Table 1). Halving or doubling the $\sigma$ schedule or $\alpha$ keeps the dense subspace ahead of the channel subset in all four settings, by $+0.11$ to $+0.97$ points (Table 3).

**Concentrating the perturbation on a few channels fails.** Random channel subsets reach $+0.08\%$ with the subspace factor $(LC/d)^{1/2}$ and, without it, $+0.08\%$ at $d = 24$ and $+0.06\%$ at $d = 576$ (Table 1). At $d = 24$, a subset moves only 0.065% of the $LC = 36{,}864$ channels and leaves the other 99.9% untouched, and each selected gain moves by about $\pm 3.9$ around one at $\sigma = 0.1$ with the factor and by only $\sigma$ without it. Measured on fixed prompts, the subset with the factor changes activations least among the five matched spaces (0.445) but spreads paired reward differences most, with a standard deviation of 0.182. Both variants leave almost every channel unperturbed, which separates them from every search space that improves the reward. Dense random mixing, which moves every channel, is a milder handicap that shrinks with dimension, and from $d = 96$ on nested dense subspaces reach $+1.35\%$ to $+1.69\%$, comparable to the $+1.56\%$ to $+1.71\%$ of the three partitions (Table 1).

### 4.4 How far does the improvement extend?

The improvement carries over to held-out prompts and other backbones but remains confined to the optimized reward. On a frozen DrawBench panel (Saharia et al., 2022) of 256 rows with 155 distinct texts, none in the PickScore training split, block gains raise the objective by $+0.79\%$ $[+0.35, +1.22]$, the 6,144-dimensional dense subspace and channel gains by $+0.94\%$ and $+0.81\%$, and matrix weights by only $+0.02\%$ (Table 4). Weighting each distinct text once leaves the block-gain improvement at $+0.71\%$ $[+0.22, +1.20]$ (Appendix Table 11). The 24-dimensional ordering also holds, with the dense subspace at $+0.28\%$ $[-0.13, +0.69]$, the channel subset at $-0.18\%$ $[-0.51, +0.16]$, and paired gaps to block gains of $+0.50$ points $[-0.005, +1.008]$ and $+0.96$ points $[+0.48, +1.48]$, while the dense subspace leads the channel subset by $+0.46$ points $[+0.07, +0.83]$.

Table 4: Frozen DrawBench evaluation over 256 rows (155 distinct texts) with four matched latents each, as three-seed means. The objective and its terms are changes from the untouched model, and CLIP similarity and ImageReward are unoptimized. Rotation and sparse rows use the initial draw.

| | relative change (%) | | | | | |
|---|---|---|---|---|---|---|
| space | objective | PickScore | HPSv2 | aesthetic | CLIP sim. | ImageReward |
| base | – | – | – | – | 0.287 | 0.868 |
| block gain | +0.79 | -0.01 | +0.14 | +0.94 | 0.285 | 0.849 |
| rotation $d$=24 | +0.28 | +0.03 | -0.90 | +0.38 | 0.286 | 0.831 |
| sparse $d$=24 | -0.18 | -0.15 | -0.91 | -0.15 | 0.285 | 0.831 |
| rotation $d$=6,144 | +0.94 | +0.15 | -0.16 | +1.13 | 0.285 | 0.847 |
| channel gain | +0.81 | +0.21 | -0.46 | +0.97 | 0.286 | 0.843 |
| matrix weights | +0.02 | +0.03 | +0.07 | +0.02 | 0.287 | 0.862 |
| CMA-ES, block gain | +0.84 | -0.51 | -1.83 | +1.19 | 0.280 | 0.771 |

**The improvement extends to other backbones.** On FLUX.1-dev and Wan2.1-T2V, block gains improve the objective by $+3.31\%$ and $+3.59\%$ against $+2.02\%$ and $+1.36\%$ for channel gains, and they lead on every paired seed, by $+1.29$ points $[+1.27, +1.34]$ and $+2.23$ points $[+1.52, +2.82]$ (Table 6). The learned block gains are also reproducible, with pairwise correlations across seeds of 0.53 to 0.67 on SD3.5-Medium (Figure 2), 0.37 to 0.50 on FLUX.1-dev and 0.72 to 0.84 on Wan2.1-T2V, whereas larger spaces reach similar improvements with seed-specific solutions (Section 4.2).

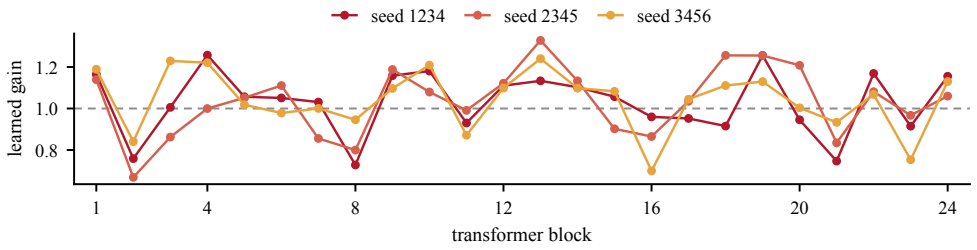


Figure 2: Learned block gains of three optimizer seeds on SD3.5-Medium, with a dashed line at the untouched gain of one. The mean profile is lowest at blocks 2, 8 and 21 and highest at 1, 13 and 19.

**Metrics outside the objective stay flat or decline.** No tuned model raises CLIP similarity (Radford et al., 2021) above the untouched model's 0.287, ImageReward (Xu et al., 2023) falls from 0.868 to 0.849 for block gains (Table 4), and within-prompt LPIPS distance (Zhang et al., 2018) on 50 separate prompts changes by $-0.0190$ $[-0.0347, -0.0025]$. Block gains raise all three objective terms on validation prompts, PickScore by $0.87\%$, HPSv2 by $1.48\%$ and the aesthetic score by $1.80\%$, and on Wan2.1-T2V by $2.16\%$, $5.43\%$ and $3.72\%$. On DrawBench, PickScore and HPSv2 stay flat at $-0.01\%$ and $+0.14\%$, while the aesthetic score rises by $0.94\%$. Matched samples from the tuned and untouched models keep their subject and layout and change in color, contrast and local texture (details in Appendix I).

Table 5: GenEval validation changes with observed seed ranges.

| | | GenEval | |
|---|---|---|---|
| subspace | seeds | Δ (%) | seed range |
| block gain | 3 | -2.36 | [-3.23, -1.31] |
| channel gain | 2 | -3.28 | [-5.21, -1.34] |

Table 6: Block gains (low $d$) and channel gains (high $d$) on each backbone, as three-seed means ± standard deviation.

| | | low $d$ | | high $d$ | |
|---|---|---|---|---|---|
| backbone | params | $d$ | Δ (%) | $d$ | Δ (%) |
| SD3.5-M | 2.24B | 24 | $+1.68 \pm 0.23$ | 36,864 | $+1.71 \pm 0.14$ |
| FLUX.1 | 12B | 19 | $+3.31 \pm 0.37$ | 58,368 | $+2.02 \pm 0.37$ |
| Wan2.1 | 1.3B | 30 | $+3.59 \pm 0.58$ | 46,080 | $+1.36 \pm 0.69$ |

**Pushing the optimizer harder trades terms against each other.** With $\alpha = 0.1$, the largest step size tested, block gains raise the objective by $+1.88\%$ while PickScore and HPSv2 fall by $1.84\%$

and 1.80%. CMA-ES (Hansen and Ostermeier, 2001) reaches +1.78%, and on DrawBench it lowers PickScore and HPSv2 by 0.51% and 1.83% and ImageReward to 0.771 (Table 4). A larger change in the objective is therefore not by itself a better model.

**A compositional objective marks the boundary.** When we optimize the GenEval reward, which scores object counts and spatial relations, on 1,920 training prompts, the score on 128 validation prompts falls in all five runs, by −2.36% with block gains and −3.28% with channel gains (Table 5).

### 4.5 Practical considerations

**Score shaping has a large effect, the optimizer a small one.** Score shaping moves the block-gain improvement from +0.60% with centered rewards through +1.41% with ranks to +1.68% with $z$-scores, a range comparable to the gaps between search spaces, whereas CMA-ES adds little over isotropic ES in the objective, +1.78% against +1.68% with overlapping seed ranges, and every step size from 0.0125 to 0.1 improves the reward, by +1.26% to +1.88% (Appendix Table 24).

**Small bf16 perturbations lose magnitude and change direction.** At $\sigma_{\text{rel}} = 10^{-4}$, or $10^{-4}$ times each tensor's RMS, only 2.6% of bf16 entries change and the stored displacement retains 29% of the intended norm (Figure 3). At $3 \times 10^{-3}$ the norm ratio reaches 1.04, but the direction cosine is 0.893 and only 54.7% of entries change. IEEE binary16 (fp16) retains 78% of the intended norm at $10^{-4}$, and fp32 preserves direction over the grid. Directions stored in bf16 give odd reward responses uncorrelated with their fp32 counterparts up to $10^{-3}$ and correlated at only 0.77 and 0.89 at $3 \times 10^{-3}$ and $10^{-2}$, against 0.95 and 0.98 for fp16. Weight-space ES therefore needs fp32 storage and arithmetic, losing its memory advantage. Appendix J.2 gives the full grid and reward probes.

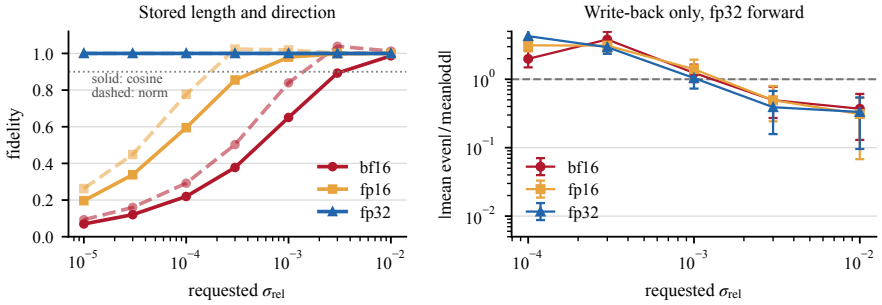


Figure 3: Realized weight perturbations (left) and the reward response ratio when perturbed weights are stored in each format and the forward pass runs in fp32 (right). Dashed lines show the stored norm ratio and solid lines the direction cosine over 43 weight tensors, and intervals bootstrap 48 directions on four fixed prompts with TensorFloat-32 (TF32) disabled throughout.

## 5 Conclusion

Under a fixed budget of generations and reward queries, what makes an ES search space effective for diffusion models is how each parameter acts on the network more than how many parameters there are. Adding coordinates within residual scaling degrades each update estimate, as classical analyses predict, but not the final reward, because a larger space holds many solutions of similar quality that noisy updates still reach. Dimension instead matters through the budget, as block gains lead when queries are scarce and a 1,536-dimensional subspace overtakes them when queries are plentiful. At a fixed dimension, the structure of the space decides the outcome. Rescaling whole residual branches works best, whereas random channel subsets achieve almost nothing and direct weight perturbations, the usual choice for language models, yield no improvement at this budget. ES search spaces for diffusion models can therefore be large, provided each parameter rescales a whole architectural unit. The improvement carries over to held-out prompts and other backbones but stays within the optimized reward, so the reward should encode every property that fine-tuning is meant to improve. Whether these conclusions extend to other interventions, such as adapters, and to larger query budgets remains open.

## AI USE STATEMENT

We used OpenAI language models through Codex to assist with drafting, restructuring and revising the manuscript, improving figure presentation, and internal manuscript review. The authors are responsible for the arguments, experimental results and final text.

## REFERENCES


Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Annual Meeting of the Association for Computational Linguistics*, pages 7319–7328, 2021.

amit154154. ES-EGGROLL for text-to-image post-training. https://amit154154.github.io/HyperscaleES_T2I/, 2026. Unofficial replication study, accessed September 2026.

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. In *International Conference on Learning Representations*, 2024.

Black Forest Labs. FLUX.1. https://github.com/black-forest-labs/flux, 2024.

Aochuan Chen, Yimeng Zhang, Jinghan Jia, James Diffenderfer, Jiancheng Liu, Konstantinos Parasyris, Yihua Zhang, Zheng Zhang, Bhavya Kailkhura, and Sijia Liu. DeepZero: Scaling up zeroth-order optimization for deep model training. In *International Conference on Learning Representations*, 2024.

Kevin Clark, Paul Vicol, Kevin Swersky, and David J. Fleet. Directly fine-tuning diffusion models on differentiable rewards. In *International Conference on Learning Representations*, 2024.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, Kyle Lacey, Alex Goodwin, Yannik Marek, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis. In *International Conference on Machine Learning*, 2024.

Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. DPOK: Reinforcement learning for fine-tuning text-to-image diffusion models. In *Advances in Neural Information Processing Systems*, 2023.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*, 2021.

Dhruba Ghosh, Hannaneh Hajishirzi, and Ludwig Schmidt. GenEval: An object-focused framework for evaluating text-to-image alignment. In *Advances in Neural Information Processing Systems*, 2023.

Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

Haoran He, Jiajun Liang, Xintao Wang, Pengfei Wan, Di Zhang, Kun Gai, and Ling Pan. Scaling image and video generation via test-time evolutionary search. *arXiv preprint arXiv:2505.17618*, 2025.

Kwanyoung Kim and Byeongsu Sim. Reward sharpness-aware fine-tuning for diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 36051–36061, 2026.

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. In *Advances in Neural Information Processing Systems*, 2023.

LAION AI. LAION aesthetic predictor. https://github.com/LAION-AI/aesthetic-predictor, 2022.

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the intrinsic dimension of objective landscapes. In *International Conference on Learning Representations*, 2018.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-GRPO: Training flow matching models via online RL. In *Advances in Neural Information Processing Systems*, 2025.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations*, 2023.

Nanye Ma, Shangyuan Tong, Haolin Jia, Hexiang Hu, Yu-Chuan Su, Mingda Zhang, Xuan Yang, Yandong Li, Tommi Jaakkola, Xuhui Jia, and Saining Xie. Scaling inference time compute for diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2523–2534, 2025. doi: 10.1109/CVPR52734.2025.00241.

Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D. Lee, Danqi Chen, and Sanjeev Arora. Fine-tuning language models with just forward passes. In *Advances in Neural Information Processing Systems*, 2023.

Yurii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2):527–566, 2017.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *IEEE/CVF International Conference on Computer Vision*, pages 4195–4205, 2023.

Mihir Prabhudesai, Anirudh Goyal, Deepak Pathak, and Katerina Fragkiadaki. Aligning text-to-image diffusion models with reward backpropagation. In *International Conference on Learning Representations*, 2024.

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. In *International Conference on Machine Learning*, 2021.

Tao Ren, Zishi Zhang, Jinyang Jiang, Zehao Li, Shentao Qin, Yi Zheng, Guanghao Li, Qianyou Sun, Yan Li, Jiafeng Liang, Xinping Li, and Yijie Peng. Half-order fine-tuning for diffusion model: A recursive likelihood ratio optimizer. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=AZ6lqcvHLX. Oral.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L. Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In *Advances in Neural Information Processing Systems*, 2022.

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.

Bidipta Sarkar, Mattie Fellows, Juan Agustin Duque, Alistair Letcher, Antonio León Villares, Anya Sims, Clarisse Wibault, Dmitry Samsonov, Dylan Cope, Jarek Liesen, Kang Li, Lukas Seier, Theo Wolf, Uljad Berdica, Valentin Mohl, Alexander David Goldie, Aaron Courville, Karin Sevegnani, Shimon Whiteson, and Jakob Nicolaus Foerster. Evolution strategies at the hyperscale. In *International Conference on Machine Learning*, 2026.

Hoigi Seo, Wongi Jeong, Kyungryeol Lee, and Se Young Chun. Efficient personalization of quantized diffusion model without backpropagation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7717–7727, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Seo_Efficient_Personalization_of_Quantized_Diffusion_Model_without_Backpropagation_CVPR_2025_paper.html.

Xiangwei Shen, Zhimin Li, Zhantao Yang, Shiyi Zhang, Yingfang Zhang, Donghao Li, Chunyu Wang, Qinglin Lu, and Yansong Tang. Directly aligning the full diffusion trajectory with fine-grained human preference. *arXiv preprint arXiv:2509.06942*, 2025.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.

James C. Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, 37(3):332–341, 1992.

Stability AI. Stable diffusion 3.5 medium. https://huggingface.co/stabilityai/stable-diffusion-3.5-medium, 2024.

Danil Tokhchukov, Aysel Mirzoeva, Andrey Kuznetsov, and Konstantin Sobolev. Calibri: Enhancing diffusion transformers via parameter-efficient calibration. *arXiv preprint arXiv:2603.24800*, 2026.

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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Wan Team. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *Journal of Machine Learning Research*, 15(27):949–980, 2014.

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.

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. In *Advances in Neural Information Processing Systems*, 2023.

Zeyue Xue, Jie Wu, Yu Gao, Fangyuan Kong, Lingting Zhu, Mengzhao Chen, Zhiheng Liu, Wei Liu, Qiushan Guo, Weilin Huang, and Ping Luo. DanceGRPO: Unleashing GRPO on visual generation. *arXiv preprint arXiv:2505.07818*, 2025.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 586–595, 2018.

Kaiwen Zheng, Huayu Chen, Haotian Ye, Haoxiang Wang, Qinsheng Zhang, Kai Jiang, Hang Su, Stefano Ermon, Jun Zhu, and Ming-Yu Liu. DiffusionNFT: Online diffusion reinforcement with forward process. In *International Conference on Learning Representations*, 2026a. Oral.

Zhi Zheng, Rongsheng Chen, Yunpeng Ba, Zhenkun Wang, Yee Whye Teh, and Wee Sun Lee. Agentic ESOpt: Fine-tuning long-horizon LLM agents with minimal GPU requirements, 2026b. URL https://arxiv.org/abs/2608.17310.

## A Related work

Related work differs along the axes that determine applicability, which a chronological presentation can conflate (Table 7). The discussion below focuses on distinctions not captured there.

**Reward fine-tuning of diffusion models.** Policy-gradient methods (Black et al., 2024; Fan et al., 2023; Liu et al., 2025; Xue et al., 2025) require transition likelihoods and typically introduce sampler stochasticity. Flow-GRPO (Liu et al., 2025), for example, converts the deployment ordinary differential equation (ODE) into a stochastic differential equation (SDE). Direct reward-backpropagation methods (Xu et al., 2023; Clark et al., 2024; Prabhudesai et al., 2024; Shen et al., 2025) require differentiable rewards, although SRPO avoids differentiating a full sampling chain. Diffusion DPO avoids

an explicit reward model (Wallace et al., 2024), whereas DiffusionNFT avoids likelihood estimation (Zheng et al., 2026a). Common rewards include learned preference models (Kirstain et al., 2023; Wu et al., 2023; Xu et al., 2023) and non-differentiable detectors (Ghosh et al., 2023). DiffusionNFT reduces memory by avoiding trajectory storage, but still differentiates model parameters and therefore falls outside the forward-only setting we study.

**Zeroth-order and evolutionary optimization.** Related simultaneous-perturbation estimators include simultaneous perturbation stochastic approximation (SPSA) (Spall, 1992), while Gaussian smoothing is analyzed by Nesterov and Spokoiny (2017). The evolutionary line includes CMA-ES (Hansen and Ostermeier, 2001), natural evolution strategies (Wierstra et al., 2014) and Salimans et al. (2017). Across deep learning, DeepZero trains networks from scratch (Chen et al., 2024). For language models, MeZO and EGGROLL study forward-only and low-rank ES methods (Malladi et al., 2023; Sarkar et al., 2026). MeZO also reports that reducing the number of tuned parameters with parameter-efficient methods does not substantially accelerate optimization and proves dimension-free rates under a low local effective-rank assumption, and Salimans et al. (2017) argue that the difficulty of ES depends on the intrinsic dimension of the problem rather than on network size. Diffusion methods can be distinguished by the perturbed variable. Inference-time methods search over noise or latents (Ma et al., 2025; He et al., 2025). ZOODiP (Seo et al., 2025) instead perturbs personalization tokens. The closest weight interventions are Calibri (Tokhchukov et al., 2026), which optimizes approximately one hundred reparameterized gains with full-covariance CMA-ES, and an unofficial public repository (amit154154, 2026) that applies EGGROLL-style ES to LoRA factors of a one-step SANA generator. Each of these studies fixes one search space and one protocol, from about one hundred gains in Calibri to full-parameter and low-rank perturbations for language models. Together they cannot show whether diffusion fine-tuning requires a low-dimensional space, what makes a search space effective, or which improvements ES can deliver. We vary the search space within one protocol per backbone and ask what limits ES in diffusion fine-tuning.

**Between zeroth and first order.** RLR (Ren et al., 2026) makes the estimator's order a per-step design choice along the chain, combining first-order, likelihood-ratio and zeroth-order pieces into an unbiased estimator with lower variance than reinforcement learning at matched compute. This result bounds the scope of our claims. The variance considered here is not intrinsic to avoiding full backpropagation, but depends on how much of the chain is left undifferentiated. RLR retains deterministic steps and introduces stochasticity where likelihood-ratio terms are used, but still requires local model gradients. It therefore falls outside the forward-only setting we study.

Table 7: Related methods organized by the axes that determine applicability rather than by chronology. "Reward access" denotes whether a method requires the reward to provide a gradient, scalar value, or ranking. "Backpropagation span" states which model evaluations are differentiated. "Sampler" indicates whether training constrains the sampler. The final column reports only documented coordinates. "NR" means not reported.

| | optimized variable | reward access | backprop. span | sampler | coordinates |
|---|---|---|---|---|---|
| DRaFT (Clark et al., 2024) | LoRA weights | gradient | full/$K$ steps | diff. DDIM (Song et al., 2021) | NR |
| SRPO (Shen et al., 2025) | weights | gradient | reconstructed step | flow path | NR |
| Flow-GRPO (Liu et al., 2025) | weights | scalar | sampled steps | SDE | NR |
| DiffusionNFT (Zheng et al., 2026a) | LoRA weights | scalar | noised step | black-box | rank 32 |
| RLR (Ren et al., 2026) | weights | gradient | selected steps | hybrid | NR |
| ZOODiP (Seo et al., 2025) | tokens | scalar | none | diffusion | embeddings |
| EvoSearch (He et al., 2025) | latents | scalar | none | flow SDE | per sample |
| Calibri (Tokhchukov et al., 2026) | gains | scalar | none | diffusion | $\sim 10^2$ |
| HyperscaleES (amit154154, 2026) | LoRA | scalar | none | one-step SANA | factors |
| **This work** | **gains, weights** | **scalar** | **none** | **ODE** | **24–1.46B** |

**Flatness and reward hacking.** Reward fine-tuning is susceptible to reward hacking. RSA-FT (Kim and Sim, 2026) uses worst-case adversarial perturbations in image and parameter space, related to sharpness-aware minimization (SAM) (Foret et al., 2021), whereas evolution strategies optimize a Gaussian average. Both alter local reward geometry, but they are not the same regularizer. We treat the comparison only as an analogy and examine reward decomposition in Appendix I.6.

## B NOTATION

Table 8 groups symbols by the section that introduces them. The search coordinates $u$ are distinct from the initial latent $\xi$, and we write the unit roundoff of a floating-point format in words rather than assigning it a symbol, because $u$ is conventionally used for both quantities and is reserved here for the search coordinates.

Table 8: Notation, grouped by the section that introduces each symbol.

| symbol | meaning |
|---|---|
| *Reward fine-tuning (Section 2)* | |
| $\theta \in \mathbb{R}^D$ | backbone weights, with total count $D$ |
| $c, \xi \sim \mathcal{N}(0, I)$ | prompt and initial latent, shared by all candidates of a generation |
| $v_\theta$, $K$ | velocity field predicted by the backbone and number of sampling steps |
| $f_b$, $h_{\text{in}}^{(b)}$ | residual branch of block $b$ and its input hidden state |
| $S(\theta; c, \xi)$, $y$ | deterministic sampler with decoder and the image it generates |
| $R(y, c)$, $J(\theta)$ | scalar reward and its expectation over prompts and latents |
| *Evolution strategies (Section 2)* | |
| $\sigma$, $\epsilon$ | perturbation scale and Gaussian perturbation direction |
| $J_\sigma$, $p_\sigma$ | Gaussian-smoothed objective and density of the perturbed parameters |
| $G$, $n$ | candidates per generation and independent directions, with $n = G/2$ |
| $B$, $T$ | prompts per generation and generations per run |
| $g$, $\hat{g}$ | gradient $\nabla_\theta J$ and its ES estimate |
| $a_i^{\pm}$, $\alpha$ | $z$-scored rewards of the $i$-th antithetic pair and step size |
| $u \in \mathbb{R}^d$, $d$ | search coordinates, with $u = 0$ the untouched model, and search dimension |
| $S_u$, $F(u)$ | sampler under intervention $u$ and coordinate objective |
| $g_u$, $\hat{g}_u$ | gradient of $F$ and its estimate in search coordinates |
| $\omega \in \mathbb{R}^m$, $\omega_0$, $A$ | ambient gains or weights, their initial value and the map $\omega = \omega_0 + Au$ |
| $L_F$ | smoothness constant of $F$ |
| $U_A$, $U_B$, $N$ | normalized estimates from two disjoint halves and size of the direction pool |
| *Search spaces (Section 3)* | |
| $L$, $H$, $C$ | transformer blocks, attention heads per block and channels per block |
| $s^{(b)}$, $w$ | gain vector of block $b$ and its width, with $d = Lw$ for partitions |
| $s$, $\delta s$ | all $LC$ channel gains and their perturbation |
| $Q_d$ | orthonormal basis of a dense random subspace or a random channel subset |
| $W^{(l)}$, $\sigma_{\text{rel}}$ | $l$-th perturbed weight tensor and scale relative to its RMS |
| $E$, $\Lambda$ | embedding of the selected weight entries and their RMS scaling |
| $U^{(l)}$, $V^{(l)}$, $r$ | low-rank perturbation factors and their rank |
| $\sigma_0$, $\sigma_T$ | first and last value of the cosine $\sigma$ schedule |
| *Transfer test (Appendices D.6 and K)* | |
| $a_{ip}$ | effect of direction $i$ on the reward of prompt $p$ |
| $\mu_i$, $\nu_{ip}$ | shared and prompt-specific components, with variances $v_\mu$ and $v_\nu$ |
| $\rho(B)$ | correlation of direction effects across two disjoint prompt batches of size $B$ |
| *Learned gains and numerical precision (Appendices I.2 and J.2)* | |
| $\Delta$, $\phi$ | displacement of learned gains and its energy fraction in block means |
| $Q$, $\delta_Q$ | rounding map of a storage format and the displacement it stores |
| $J_0$, $r_{\text{odd}}$, $r_{\text{even}}$, $\rho_{\text{eo}}$ | unperturbed reward, odd and even reward responses and their ratio |
| unit roundoff | $2^{-p}$ for $p$ significand bits, $2^{-8}$ in bf16, $2^{-11}$ in fp16 and $2^{-24}$ in fp32 |

## C ADDITIONAL BACKGROUND

**The estimator requires no gradient of $J$.** Let $p_\sigma(\theta') = \mathcal{N}(\theta' \mid \theta, \sigma^2 I)$. The Gaussian score $\nabla_\theta \log p_\sigma(\theta') = (\theta' - \theta)/\sigma^2$ moves the derivative from $J$ to the density:

$$\nabla_\theta J_\sigma(\theta) = \frac{1}{\sigma}\, \mathbb{E}_{\epsilon \sim \mathcal{N}(0,I)} \big[ J(\theta + \sigma\epsilon)\, \epsilon \big]. \tag{11}$$

Under the usual integrability conditions for differentiating the Gaussian convolution, only values of $J$ appear in this expression. Thus, $J$ itself may be non-differentiable, discontinuous, or implemented

by an external program, while the smoothed objective remains differentiable. This property allows evolution strategies to optimize rewards such as OCR accuracy or a detector-based compositional score.

**The antithetic form removes the absolute reward level.** The sample mean of Equation 11 is unbiased but carries a term $J(\theta)\epsilon_i/\sigma$ whose size is determined by the *absolute* reward level rather than by its change. It can therefore dominate the directional reward difference. Evaluating both $+\sigma\epsilon_i$ and $-\sigma\epsilon_i$ cancels the term exactly because every pair subtracts the same absolute reward level. For a smooth $J$, the subtraction also cancels even Taylor terms sample by sample. This costs two evaluations per direction and is advantageous when evaluation is noiseless, as it is for diffusion under common random numbers (§4.1). Evaluation is not noiseless in a stochastic agent environment, which motivates the one-sided form used by Zheng et al. (2026b).

**Multiplication by the direction recovers a vector estimate.** The central difference is a scalar that measures the rate of reward change along $\epsilon_i$. By itself, it does not specify a direction in $\mathbb{R}^D$. Multiplication by $\epsilon_i$ converts this rate into a vector, and averaging recovers the gradient because $\mathbb{E}[\epsilon\epsilon^\top] = I$ for isotropic $\epsilon$, so $\mathbb{E}[\langle g, \epsilon\rangle\epsilon] = \mathbb{E}[\epsilon\epsilon^\top]g = g$. Components orthogonal to $g$ cancel in expectation, while the component along $g$ accumulates in expectation. Division by $2\sigma$ converts a reward difference into a rate, and division by $n$ averages over the independent directions.

# D Derivations

## D.1 The Gaussian smoothing identity

This section derives Equations 3 and 11.

Let $p_\sigma(\theta') = (2\pi\sigma^2)^{-D/2}\exp(-\|\theta' - \theta\|^2/2\sigma^2)$ be the density of $\mathcal{N}(\theta, \sigma^2 I)$. Then $J_\sigma(\theta) = \int J(\theta')\, p_\sigma(\theta')\,\mathrm{d}\theta'$.

**The bias expansion.** Substituting $\theta' = \theta + \sigma\epsilon$ and expanding through third order gives

$$J(\theta + \sigma\epsilon) = J(\theta) + \sigma\,\epsilon^\top\nabla J(\theta) + \frac{\sigma^2}{2}\epsilon^\top\nabla^2 J(\theta)\epsilon + \frac{\sigma^3}{6}\nabla^3 J(\theta)[\epsilon,\epsilon,\epsilon] + \mathcal{O}(\sigma^4\|\epsilon\|^4). \tag{12}$$

Taking the expectation and using $\mathbb{E}[\epsilon] = 0$ and $\mathbb{E}[\epsilon^\top M\epsilon] = \mathrm{Tr}(M)$ for a symmetric matrix $M$ and $\epsilon \sim \mathcal{N}(0, I)$ yields Equation 3. Gaussian symmetry eliminates the displayed odd-order terms.

**The estimator.** Differentiating under the integral and using the score of the Gaussian, $\nabla_\theta \log p_\sigma(\theta') = (\theta' - \theta)/\sigma^2$, gives

$$\nabla_\theta J_\sigma(\theta) = \int J(\theta')\,\nabla_\theta p_\sigma(\theta')\,\mathrm{d}\theta' = \int J(\theta')\,p_\sigma(\theta')\,\frac{\theta' - \theta}{\sigma^2}\,\mathrm{d}\theta' \tag{13}$$

$$= \frac{1}{\sigma}\,\mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}\big[\,J(\theta + \sigma\epsilon)\,\epsilon\,\big]. \tag{14}$$

The derivative acts on the density rather than on $J$. Under the integrability conditions for differentiating the Gaussian convolution, $J$ may be discontinuous and accessed through evaluations alone. This permits rewards such as OCR accuracy or a detector-based compositional score.

## D.2 Variance reduction from antithetic evaluations

The one-sided estimator $\hat{g}_+ = J(\theta + \sigma\epsilon)\,\epsilon/\sigma$ contains a term proportional to $J(\theta)^2\,\mathbb{E}\|\epsilon\|^2/\sigma^2$. Even for a constant $J$, its variance grows as $D/\sigma^2$. Population centering or $z$-scoring removes a common additive level exactly, while the antithetic estimator

$$\hat{g}_\pm = \frac{J(\theta + \sigma\epsilon) - J(\theta - \sigma\epsilon)}{2\sigma}\,\epsilon \tag{15}$$

cancels the term exactly. Expanding both evaluations as $J(\theta \pm \sigma\epsilon) = J \pm \sigma\epsilon^\top\nabla J + \frac{\sigma^2}{2}\epsilon^\top\nabla^2 J\epsilon \pm \mathcal{O}(\sigma^3)$, the constant and the quadratic term both drop out of the difference. Dividing by $2\sigma$ leaves

$\epsilon^\top \nabla J + \mathcal{O}(\sigma^2)$. The one-sided estimator has the same $\mathcal{O}(\sigma^2)$ smoothing bias, because its $\mathcal{O}(\sigma)$ term carries a third Gaussian moment and vanishes in expectation. Thus, the advantage of the antithetic form concerns variance rather than bias order. In particular, it cancels the constant and even-order terms *per sample* rather than only in expectation. This distinction matters when the absolute reward is much larger than directional differences.

The antithetic form requires two evaluations per direction rather than one. This trade-off is favorable when evaluation is noiseless. Otherwise, the second evaluation partly resamples the noise that the pair is intended to cancel. Under common random numbers (§4.1), diffusion evaluation is deterministic, with bitwise repeatable rewards, so the antithetic form is preferable in our setting. In contrast, language-agent environments are stochastic, which motivates the one-sided form used in agentic evolution-strategy work.

### D.3 Proof of Proposition 1

In the limit $\sigma \to 0$, the central difference in Equation 4 converges to the directional derivative. For brevity, let $g := g_u = \nabla_u F$ and $\hat{g} := \hat{g}_u$. Then $\hat{g} = \frac{1}{n}\sum_{i=1}^{n} \langle g, \epsilon_i \rangle \, \epsilon_i$, where the directions $\epsilon_i \sim \mathcal{N}(0, I_d)$ are independent.

**Unbiasedness.** $\mathbb{E}[\langle g, \epsilon \rangle \epsilon] = \mathbb{E}[\epsilon\epsilon^\top]\, g = g$, so $\mathbb{E}[\hat{g}] = g$.

**Second moment.** We expand $\|\hat{g}\|^2 = \frac{1}{n^2}\sum_{i,j} \langle g, \epsilon_i \rangle \langle g, \epsilon_j \rangle \, \epsilon_i^\top \epsilon_j$. For $i \neq j$, independence gives $\mathbb{E}[\langle g, \epsilon_i \rangle \epsilon_i]^\top \mathbb{E}[\langle g, \epsilon_j \rangle \epsilon_j] = \|g\|^2$. For $i = j$, let $q = g/\|g\|$ and decompose $\epsilon$ into its component along $q$ and the $d-1$ orthogonal components:

$$\mathbb{E}\big[\langle g, \epsilon \rangle^2 \|\epsilon\|^2\big] = \|g\|^2 \, \mathbb{E}\big[\epsilon_q^2 (\epsilon_q^2 + \textstyle\sum_{k=2}^{d} \epsilon_k^2)\big] = \|g\|^2 \, (3 + (d-1)) = \|g\|^2 (d+2), \tag{16}$$

using $\mathbb{E}[\epsilon_q^4] = 3$ and independence of the remaining coordinates. Combining the $n$ diagonal and $n(n-1)$ off-diagonal terms,

$$\mathbb{E}\big[\|\hat{g}\|^2\big] = \frac{n\,\|g\|^2(d+2) + n(n-1)\,\|g\|^2}{n^2} = \|g\|^2 \, \frac{d+n+1}{n}. \tag{17}$$

**Alignment.** Rotate coordinates so that $g = \|g\| e_1$, and write $X = \sum_i \epsilon_{i1}^2 \sim \chi_n^2$. The component of $\hat{g}$ along $g$ is $\|g\| X/n$. Conditional on the first coordinates, the squared norm orthogonal to $g$ is $\|g\|^2 XZ/n^2$, where $Z \sim \chi_{d-1}^2$ is independent of $X$. Hence

$$\cos^2(\hat{g}, g) = \frac{X}{X+Z} \sim \mathrm{Beta}\left(\frac{n}{2}, \frac{d-1}{2}\right). \tag{18}$$

Taking the beta half-moment gives the gamma ratio in Equation 8. Its large-$d$ approximation reduces to the standard rate $\sqrt{n/d}$. □

Appendix D.5 relates this alignment to a local progress bound. Its dimension dependence also depends on the step-size rule.

### D.4 The split-half attenuation relation

Section 2 defines split-half agreement, and Sections 4.2 and 4.3 report it. We derive its properties here. A direction pool is a fixed set of $N$ pre-evaluated random directions. Divide it into halves $A$ and $B$, and normalize the resulting estimates as $U_A = \hat{g}_A/\|\hat{g}_A\|$ and $U_B = \hat{g}_B/\|\hat{g}_B\|$. If the halves are independent and identically distributed, then exactly

$$\mathbb{E}[U_A^\top U_B] = \mathbb{E}[U_A]^\top \mathbb{E}[U_B] = \|\mathbb{E}[U]\|^2. \tag{19}$$

If $\mathbb{E}[U] = \lambda t$ for a unit target $t$ and $\lambda \geq 0$, its square root equals $\lambda = \mathbb{E}[U^\top t]$. Without this collinearity assumption, the square root measures the agreement of the two estimates, not alignment to a named target.

This relation depends on two conditions, neither of which can be verified from split-half agreement alone. First, the halves must be independent. Shared finite-$\sigma$ or rounding effects can bias the common

target without coupling independently sampled halves. Second, $\mathbb{E}[U]$ need not be collinear with the true gradient $g_u$. A bias shared by both halves preserves Equation 19 while changing its direction. Therefore, Appendix G also reports a coordinate-wise finite-difference diagnostic wherever the required $2d$ evaluations are affordable, while making no claim that this second diagnostic is itself the true gradient.

**Finite-pool computation.** Every probe evaluates the untouched model, $u = 0$, at $\sigma = 0.1$, the initial scale of the training schedule. For its fixed prompt batch and initial latents, the implementation evaluates $N$ random directions antithetically and stores their central differences. For each requested half-size $n$ with $2n \le N$, it samples $2n$ distinct pool entries without replacement and assigns $n$ to each half. Each estimate is the mean of its direction vectors weighted by their central differences. The reported agreement averages the cosine between the two estimates over repeated splits of this pool, 200 for the primary pools and 500 for the independent ones. A zero-norm estimate contributes zero. The square root is reported when the mean agreement is nonnegative, and the raw mean is kept otherwise. A pool of $N$ directions costs $2N$ objective evaluations and $\mathcal{O}(Nd)$ vector operations, which keeps the probe practical across the tested dimensions. Repeated splits reuse the evaluated pool and require no additional model evaluations. They are therefore dependent draws from a single pool, and their spread measures split noise only, not the variation from redrawing the batch and directions. Intervals across independent probes address the latter.

### D.5 Local progress and the exclusion of low-rank perturbations

**The rate.** Under the concentration approximation, a fixed-length step has inverse alignment $\sqrt{1 + d/n}$, which reduces to $\sqrt{d/n}$ when $d \gg n$. Step-size optimization gives a different dependence. For an $L_F$-smooth $F$ and an update $u \leftarrow u + \eta \hat{g}_u$,

$$\mathbb{E}\big[F(u + \eta\hat{g}_u)\big] \;\ge\; F(u) + \eta\|g_u\|^2 - \tfrac{L_F\eta^2}{2}\,\mathbb{E}\|\hat{g}_u\|^2, \tag{20}$$

where we use $\mathbb{E}[\hat{g}_u] = g_u$. Optimizing over $\eta$ and substituting Equation 7 yields a guaranteed per-step increase of $\|g_u\|^2 n/\big(2L_F(d+n+1)\big)$, so the dimension factor for a fixed local increase is $\mathcal{O}(1 + (d+1)/n)$. This reduces to $\mathcal{O}(d/n)$ only when $d \gg n$.

Equation 20 bounds the expected increase from the current point under smoothness and the unbiased limiting estimator. Extending it to an iteration count requires control of the gradient norm and smoothness along the optimization trajectory. We use its dimension factor to interpret local progress. Predicting the hundred-generation training budget would additionally require tracking these quantities and the effects of finite-scale, score-normalized updates.

**Exclusion of low-rank perturbations from the dimension axis.** For $W^{(l)} \in \mathbb{R}^{m_l \times n_l}$, a rank-$r$ perturbation draws $U^{(l)} \in \mathbb{R}^{m_l \times r}$ and $V^{(l)} \in \mathbb{R}^{n_l \times r}$ and applies $U^{(l)}V^{(l)\top}/\sqrt{r}$. Write $q = \sum_l r(m_l + n_l)$ for the number of random factor entries. The quadratic, non-injective factorization is not a fixed linear map from $q$ isotropic Gaussian coordinates, and the induced matrix directions are non-Gaussian. Their linear span may be the full ambient matrix space, so absence of a span is not the reason for exclusion. Moreover, $q$ overcounts the rank-$r$ manifold dimension because $U \mapsto UM$ and $V \mapsto VM^{-\top}$ leave the product unchanged for any invertible $M$.

Proposition 1 assumes isotropic Gaussian directions in a fixed $d$-dimensional space and therefore does not apply to this family. We report the family separately against $q$, the ambient parameter count and the rank. EGGROLL (Sarkar et al., 2026) analyzes this family with a separate low-rank score-function estimator. Table 9 lists the fixed-coordinate quantities for the tested spaces.

### D.6 Attenuation of the transfer test

Let $a_{ip}$ denote the antithetic central difference of direction $i$ measured on prompt $p$. We model it as $a_{ip} = \mu_i + \nu_{ip}$, where $\mu_i$ and $\nu_{ip}$ are independent, the $\nu_{ip}$ are mutually uncorrelated across prompts, $\mathrm{Var}(\mu) = v_\mu$, $\mathrm{Var}(\nu) = v_\nu$, and $\mathbb{E}[\nu_{ip}] = 0$. Here, $\mu_i$ is the component of the direction's effect shared across prompts, whereas $\nu_{ip}$ is specific to prompt $p$.

**Proposition 2** (Attenuation of the transfer test)**.** *Under this model,* $\rho(B) = v_\mu/(v_\mu + v_\nu/B)$.

Table 9: Estimator quantities for $n = 8$ independent directions, obtained from a population of $G = 16$ allocated to mirrored pairs. The cosine column gives the exact expectation under Proposition 1, for which $\cos^2 \sim \mathrm{Beta}(n/2, (d-1)/2)$, and Equation 8 approximates it. Only spaces generated by a fixed linear isotropic coordinate map are included.

| search space | $d$ | $\mathbb{E}[\cos(\hat{g}_u, g_u)]$ | $\mathbb{E}\|\hat{g}_u\|^2/\|g_u\|^2$ |
|---|---|---|---|
| block gain | 24 | 0.496 | 4 |
| head gain | 576 | 0.114 | 73 |
| channel gain | 36,864 | 0.014 | $10^{3.7}$ |
| matrix weights | 1,460,000,000 | 7.2e-05 | $10^{8.3}$ |

For two disjoint prompt batches $\mathcal{A}, \mathcal{B}$ of size $B$, write $\bar{a}_i^{\mathcal{A}} = \mu_i + \frac{1}{B}\sum_{p\in\mathcal{A}} \nu_{ip}$, with an analogous definition for $\mathcal{B}$. Independence between the two batches gives $\mathrm{Cov}(\bar{a}^{\mathcal{A}}, \bar{a}^{\mathcal{B}}) = v_\mu$, while $\mathrm{Var}(\bar{a}^{\mathcal{A}}) = \mathrm{Var}(\bar{a}^{\mathcal{B}}) = v_\mu + v_\nu/B$. Hence

$$\rho(B) = \frac{v_\mu}{v_\mu + v_\nu/B}. \qquad \square \tag{21}$$

When $v_\nu > 0$, $\rho$ increases monotonically with $B$. At the smallest usable batch size, $\rho(1) = v_\mu/(v_\mu + v_\nu)$, which approaches zero when prompt-specific variation dominates, even if $v_\mu > 0$. Therefore, a small measured $\rho$ at a fixed $B$ does not establish that $v_\mu = 0$. This attenuation is entirely due to prompt-specific variance, since $v_\nu = 0$ gives $\rho = 1$ for every $B$. We test this prediction directly in §K by estimating $\rho(B)$ for $B = 1 \ldots 32$ from a single set of per-prompt rewards.

## E Detailed setup

**Hardware.** Experiments ran on NVIDIA H20 GPUs with 96 GB of memory, using PyTorch 2.6.0, CUDA 12.4 and driver 535.247.01. A run ordinarily uses one eight-GPU node and shards population evaluations across devices. Reported memory is peak allocated memory on one GPU, not the sum over the node.

Table 10: Training protocols. Prompt rows index the source training file, and the last rows of each range form the validation panel. Every protocol keeps weights and forward arithmetic in fp32 and uses the gain-space $\sigma$ schedule and step size below.

| setting | SD3.5 composite | SD3.5 GenEval | FLUX.1-dev | Wan2.1-T2V |
|---|---|---|---|---|
| checkpoint | SD3.5-Medium | SD3.5-Medium | FLUX.1-dev | Wan2.1-T2V-1.3B |
| output | $512 \times 512$ | $512 \times 512$ | $512 \times 512$ | $480 \times 480$, 17 frames |
| sampler | 25 Euler steps | 25 Euler steps | 25 Euler steps | 25 Euler steps |
| guidance | CFG 4.5 | CFG 4.5 | distilled, 3.5 | CFG 5.0 |
| prompt rows | PickScore 1–512 | GenEval 1–2048 | PickScore 1–512 | PickScore 1–512 |
| train / validation | 448 / 64 | 1920 / 128 | 448 / 64 | 448 / 64 |
| reward | composite | GenEval | composite | composite, frame mean |
| score shaping | z-score | rank | z-score | z-score |
| $G, n, B, T$ | 16, 8, 8, 100 in every protocol | | | |
| gain $\sigma$, $\alpha$ | cosine $0.1 \to 0.02$, $\alpha = 0.05$ | | | |
| optimizer seeds | 1234, 2345, 3456, with the first two for GenEval channel gains | | | |

**Scorers and data.** The objective is the sum of PickScore v1 on its CLIP ViT-H/14 processor, HPSv2.1 on OpenCLIP ViT-H/14, and the LAION aesthetic predictor v2 linear head on CLIP ViT-L/14. CLIP similarity uses CLIP ViT-L/14 and ImageReward uses ImageReward v1.0. GenEval uses a Mask2Former Swin-S detector trained on COCO with CLIP colour classification. Wan scores each of the 17 decoded frames as an image with the video's prompt, and a video's reward is the frame mean. On FLUX.1-dev, gains act on the 19 dual-stream blocks. Its 38 single-stream blocks concatenate the text stream inside the block, so the gain hook leaves them unscaled, and Table 6 counts only the 19 active blocks and their $19 \times 3072$ channels. Wan gains act on all 30 blocks. Each

generation seeds its prompt draw from the optimizer seed and generation index, and every candidate reuses the same prompts and initial latents. Validation latents are seeded by 777 plus the prompt index. Random subspaces use subspace seed 20260918 for the initial draw and 20260919–20260922 for the four further draws.

**Sigma schedules.** We apply cosine annealing from $\sigma_0$ to $\sigma_T$ over each run. The RMS-matched SD3.5 gain comparisons use $0.1 \to 0.02$, where $\sigma^2$ is the expected squared RMS of the expanded channel perturbation: $\mathbb{E}[\|\delta s\|^2/(LC)] = \sigma^2$. Equal per-coordinate $\sigma$ gives this normalization for partitions that cover all channels. Dense random subspaces (labelled rotation in the tables) and random channel subsets (labelled sparse) use the factor $\sqrt{LC/d}$ to match the same expectation at each generation, and $0.1$ is the initial scale. The equal-$\sigma$ random-subset controls in Table 1 omit this factor, and Appendix F gives the per-channel scales that result. Weight-space perturbations are normalized by the RMS of each tensor and use $10^{-3} \to 2 \times 10^{-4}$ in relative units, whereas rank-one perturbations use $5 \times 10^{-3} \to 10^{-3}$.

**Learning rates.** $\alpha = 0.05$ for the gain spaces and $0.005$ for the weight spaces, reflecting the different model changes induced by a unit step in each parameterization. A fixed $\alpha$ across families would itself impose a non-neutral choice. Table 24 bounds sensitivity within the block-gain space, but cross-family gaps can still include a contribution from $\alpha$.

**Scale and step-size controls.** For the initial dense-subspace and random-subset draws at $d = 24$, we rerun optimizer seed 1234 with $\sigma_0$ and $\sigma_T$ halved to $0.05 \to 0.01$ or doubled to $0.2 \to 0.04$, and separately with $\alpha$ halved to $0.025$ or doubled to $0.1$. Every other setting follows the main protocol, so each setting yields one paired comparison of the two families (Table 3).

**Budgets.** The reported SD3.5-Medium, FLUX and Wan runs use 100 generations. Within each table row, both search spaces receive the same generations, population and reward calls. Budgets are not matched across backbones. One generation costs 82s on SD3.5-Medium, 159s on FLUX.1-dev and about 602s on Wan2.1-T2V. These are replication checks rather than a controlled cross-backbone comparison.

**Evaluation.** We evaluate the validation panel before training and after every tenth generation, using 64 prompts for every backbone and 128 for GenEval. Each reported change is the mean of the evaluations after generations 80, 90 and 100, as prespecified. The panel is deterministic, so this average smooths the trajectory but does not create independent observations.

**DrawBench intervals.** The held-out comparison in Table 4 uses the first 256 rows of the DrawBench test file with four matched latents per row. These rows contain 155 distinct texts. Of these, 76 appear once, 59 twice, 18 three times and 2 four times, and none occurs in the PickScore training split. The reported 95% intervals resample optimizer seeds and source rows as crossed factors, with latents nested within each selected row. Paired contrasts reuse the same draws for both search spaces. Table 11 repeats the analysis with distinct texts as the resampled unit, so that all rows of a text enter or leave together, and with each text weighted once. Seedwise gaps of block gains over the dense subspace are $+0.16$, $+0.61$ and $+0.74$ points, and over the random channel subset $+0.70$, $+0.86$ and $+1.32$ points. The gap of the dense subspace over the random subset, the held-out counterpart of the fixed-dimension comparison for the initial draw, is $+0.46$ points $[+0.07, +0.83]$ under row resampling and $+0.37$ $[-0.02, +0.75]$ under text weights.

Table 11: DrawBench sensitivity to repeated texts. Entries are composite changes or paired gaps in percentage points, with 95% intervals from 100,000 crossed resamples of optimizer seeds and the stated unit, latents nested within rows. The source-row column is the prespecified analysis reported in the text. Text clusters keep row weights, so their point estimates equal the source-row ones.

| effect | source rows | text clusters | unique-text weights |
|---|---|---|---|
| block gain | $+0.79$ $[+0.35, +1.22]$ | $[+0.33, +1.24]$ | $+0.71$ $[+0.22, +1.20]$ |
| rotation $d$=24 | $+0.28$ $[-0.13, +0.69]$ | $[-0.17, +0.73]$ | $+0.10$ $[-0.35, +0.54]$ |
| sparse $d$=24 | $-0.18$ $[-0.51, +0.16]$ | $[-0.51, +0.16]$ | $-0.27$ $[-0.62, +0.08]$ |
| block − rotation | $+0.50$ $[-0.005, +1.008]$ | $[-0.05, +1.04]$ | $+0.61$ $[+0.07, +1.16]$ |
| block − sparse | $+0.96$ $[+0.48, +1.48]$ | $[+0.45, +1.50]$ | $+0.98$ $[+0.46, +1.54]$ |
| rotation − sparse | $+0.46$ $[+0.07, +0.83]$ | $[+0.05, +0.85]$ | $+0.37$ $[-0.02, +0.75]$ |

## F PERTURBATION MATCHING AND CALIBRATION

Section 3.4 matches the expected squared RMS of the gain perturbation after expanding it over all $m = LC = 36{,}864$ channel gains of SD3.5-Medium. We give here the per-channel scales that this rule implies and the measurements that test it. Every gain space acts on the expanded channel-gain vector through a fixed map $A \in \mathbb{R}^{m\times d}$, so a direction $\epsilon \sim \mathcal{N}(0, I_d)$ changes channel $k$ by $\delta s_k = \sigma(A\epsilon)_k$. With $e_k$ the $k$-th standard basis vector and $\|A\|_F$ the Frobenius norm,

$$\mathrm{Var}(\delta s_k) = \sigma^2 \left\|A^\top e_k\right\|^2, \qquad \frac{1}{m}\sum_{k=1}^{m} \mathrm{Var}(\delta s_k) = \frac{\sigma^2}{m}\|A\|_F^2.$$

Matching sets $\|A\|_F^2 = m$, so the channel average equals $\sigma^2$ in every space. The spaces differ in how they distribute this energy across channels.

**Partitions.** Block, head-group and channel gains of width $w$ use $A = I_{Lw} \otimes \mathbf{1}_{C/w}$, which copies each coordinate to the $C/w$ contiguous channels it controls. Every row of $A$ holds a single one, so every channel has variance $\sigma^2$, and the three partitions differ only in which channels move together.

**Dense random subspaces.** A dense random subspace uses $A = (m/d)^{1/2}Q_d$, where the $d$ orthonormal columns of $Q_d$ span a uniformly random subspace. Channel $k$ then has variance $\sigma^2(m/d)\|Q_d^\top e_k\|^2$. For a uniformly random subspace, $\|Q_d^\top e_k\|^2$ follows a $\mathrm{Beta}(d/2, (m-d)/2)$ law, so the factor $(m/d)\|Q_d^\top e_k\|^2$ has mean one and variance $2(m-d)/[d(m+2)]$, which is below $2/d$. Every channel therefore moves by about $\sigma$, the spread across channels shrinks as $d$ grows, and each channel's change mixes all $d$ coordinates.

**Random channel subsets.** A random channel subset uses $A = (m/d)^{1/2}Q_d$, where the columns of $Q_d$ are instead the standard basis vectors of $d$ channels drawn without replacement. The selected channels have variance $\sigma^2 m/d$ and all others have zero. At $d = 24$, $m/d = C = 1536$, so each selected channel moves with standard deviation $C^{1/2}\sigma \approx 39\sigma$, about 3.9 at $\sigma_0 = 0.1$ against a baseline gain of one, while the other 99.9% of channels stay fixed. The equal-$\sigma$ controls drop the factor $(m/d)^{1/2}$, so the selected channels move by $\sigma$ and only the fraction $d/m$ of channels moves at all, 0.065% at $d = 24$. Neither variant spreads the perturbation over the network as partitions and dense subspaces do (Figure 4).

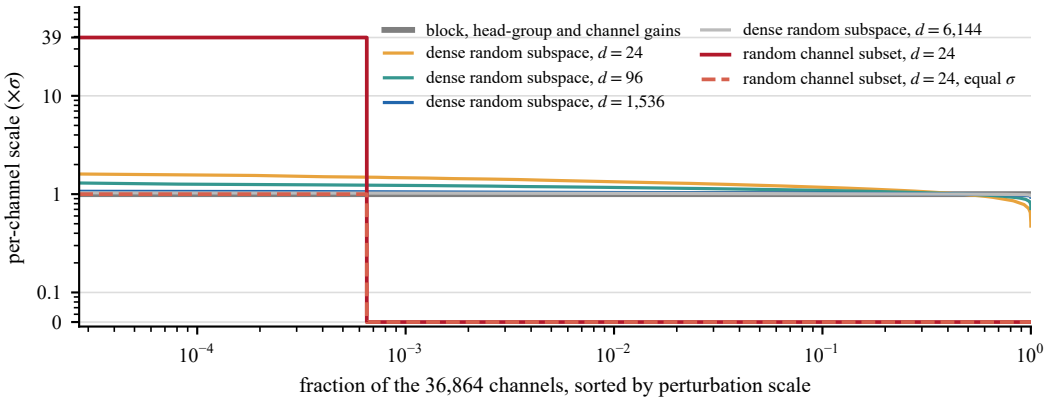


Figure 4: Analytic per-channel perturbation scale of the SD3.5 gain spaces. For each space, the $LC = 36{,}864$ channels are sorted by the standard deviation of their perturbation in units of $\sigma$, under matched total energy except for the equal-$\sigma$ random channel subsets. A flat curve at one means that every channel moves equally, and a short spike followed by zeros means that a few channels carry the whole perturbation. Both axes are logarithmic, zero scales sit on the lowest tick, and each dense subspace shows one draw of channel scales from its exact Beta law.

**Measured perturbation size.** We test the rule on eight directions and four fixed prompts. For each space we record the relative change of sampled entries in the scaled hidden stream, the RMS

change of the final latent and of the decoded pixels, and the standard deviation of paired reward differences (Table 12). Under matching, the random channel subset changes activations least, by $0.445$ against $0.668$ for block gains, yet it has the largest reward-difference standard deviation, $0.182$. A perturbation concentrated on 24 channels thus moves the reward more erratically while changing activations less on average, which is consistent with the failure of this family in Section 4.3 but does not isolate its cause. The Euclidean schedule instead holds the expected norm of the coordinate perturbation at its block-gain value. The channel-averaged RMS then falls to $0.02041$ for head-group gains and to $0.002552$, about 39 times below block gains, for channel gains and both 24-dimensional subspaces, and channel gains change activations by only $0.028$. Across the five spaces, the ratio of the largest to the smallest activation change is $31.6$ under the Euclidean schedule and $1.50$ under matching, and the corresponding pixel ratios are $28.4$ and $1.61$. The spread that remains under matching reflects which channels move together, the variable that Section 4.3 varies at fixed dimension.

Table 12: Empirical perturbation size over eight directions and four fixed prompts. Activation is relative change on sampled searched-stream entries. Latent and pixel columns are RMS changes, and reward sd is the paired reward-difference standard deviation. The lower block gives the max/min ratio across the five spaces under each schedule, computed from unrounded values.

| schedule | space | $\sigma$ | activation rel. | latent RMS | pixel RMS | sd($\Delta R$) |
|---|---|---|---|---|---|---|
| matched | block gain | 0.1 | 0.668 | 0.450 | 0.152 | 0.159 |
| | head gain | 0.1 | 0.566 | 0.357 | 0.124 | 0.130 |
| | channel gain | 0.1 | 0.520 | 0.333 | 0.118 | 0.134 |
| | rotation $d$=24 | 0.1 | 0.527 | 0.342 | 0.121 | 0.124 |
| | sparse $d$=24 | 0.1 | 0.445 | 0.268 | 0.094 | 0.182 |
| euclidean | block gain | 0.1 | 0.668 | 0.450 | 0.152 | 0.159 |
| | head gain | 0.02041 | 0.203 | 0.122 | 0.045 | 0.061 |
| | channel gain | 0.002552 | 0.028 | 0.016 | 0.007 | 0.014 |
| | rotation $d$=24 | 0.002552 | 0.035 | 0.020 | 0.009 | 0.006 |
| | sparse $d$=24 | 0.002552 | 0.021 | 0.012 | 0.005 | 0.008 |

| functional estimand | Euclidean range | matched range |
|---|---|---|
| activation change | 31.6× | 1.50× |
| latent RMS | 37.2× | 1.68× |
| pixel RMS | 28.4× | 1.61× |
| reward-difference sd | 25.0× | 1.47× |

## G SPLIT-HALF AGREEMENT ACROSS SEARCH DIMENSIONS

We collect here the split-half agreement probes behind Sections 4.2 and 4.3. Each probe evaluates the untouched model at the initial training scale $\sigma = 0.1$ on one fixed batch of eight training prompts and latents, using the finite-pool computation of Appendix D.4. The population sweep at 24 and 1,536 search coordinates appears with its discussion in Section 4.2 (Table 2).

The primary direction pools give the dimension sweep in Table 13. At the training population, $n = 8$, the square-root agreement statistic falls from $0.493$ for block gains to $0.014$ for channel gains, close to the dimension-only values $0.500$ and $0.015$, while final reward stays between $+1.15\%$ and $+1.71\%$. Independent probes redraw both the prompt batch and the direction pool (Table 14). They separate block gains from the two 24-dimensional random families, but their mean cosines take both signs at $d = 6{,}144$ and $d = 36{,}864$, so pools of 256 directions do not resolve agreement at these dimensions.

At $d = 24$, we also vary the number of directions per half from 2 to 128 and compare each estimate with a coordinate-wise finite-difference vector at the training $\sigma$ (Table 15). This reference costs $2d$ evaluations and measures finite-scale changes rather than the exact gradient. With 128 directions per half, the agreement of block gains, the dense random subspace and the random channel subset is $0.914$, $0.883$ and $0.844$, whereas their cosines with the finite-difference vector are $0.496$, $0.323$ and $0.508$. The two diagnostics therefore rank the three spaces differently, and the finite-difference cosine places the random channel subset first although it barely changes reward (Section 4.3).

Table 13: Square-root split-half agreement, its dimension-only approximation and final reward at $n = 8$, from the primary direction pools. Dense random subspaces (rotation) are nested, and channel gain uses all channel coordinates.

| space | $d$ | $\sqrt{\mathbb{E}[\cos(\hat{g}_A,\hat{g}_B)]}$ | $\sqrt{n/(n+d)}$ | reward $\Delta$ (%) |
|---|---|---|---|---|
| block gain | 24 | 0.493 | 0.500 | +1.68 |
| rotation $d$=24 | 24 | 0.405 | 0.500 | +1.15 |
| rotation $d$=96 | 96 | 0.199 | 0.277 | +1.56 |
| rotation $d$=576 | 576 | 0.085 | 0.117 | +1.42 |
| rotation $d$=6,144 | 6,144 | 0.034 | 0.036 | +1.69 |
| channel gain | 36,864 | 0.014 | 0.015 | +1.71 |

Table 14: Independently seeded split-half agreement probes at the training population $n = 8$. Each probe redraws the batch of prompts and latents and the direction pool. At $d = 24$ the pool sizes are $N = 256, 128, 128$, and at higher $d$ every pool has $N = 256$. Here $a_{\text{rep}} = \sqrt{\mathbb{E}[\cos(\hat{g}_A, \hat{g}_B)]}$ is shown with the raw mean cosine, whose sign is not constrained. A dash marks a space in which some probe returns a negative mean cosine, so that its square root is undefined.

| subspace | $d$ | probes | $a_{\text{rep}}$, mean [range] | $\mathbb{E}[\cos]$, mean [range] | $\sqrt{n/(n+d)}$ |
|---|---|---|---|---|---|
| block gain | 24 | 3 | 0.480 [0.462, 0.493] | 0.230 [0.213, 0.243] | 0.500 |
| rotation $d$=24 | 24 | 3 | 0.384 [0.314, 0.432] | 0.150 [0.099, 0.187] | 0.500 |
| sparse $d$=24 | 24 | 3 | 0.367 [0.331, 0.408] | 0.135 [0.110, 0.166] | 0.500 |
| rotation $d$=6,144 | 6,144 | 3 | – | 0.00081 [-0.00012, 0.00142] | 0.036 |
| channel gain | 36,864 | 3 | – | 0.00000 [-0.00019, 0.00020] | 0.015 |

Table 15: Split-half agreement and the finite-difference diagnostic against the number of directions per half for the three 24-dimensional spaces. Here $a_{\text{rep}}$ denotes the square-root split-half agreement, $a_{\text{FD}} = \cos(\hat{g}, g_{\text{FD}})$ compares the estimate with the coordinate-wise finite-difference vector, and the dimension-only approximation $\sqrt{n/(n+d)}$ is shared by the three spaces.

| | | block gain | | rotation $d$=24 | | sparse $d$=24 | |
|---|---|---|---|---|---|---|---|
| $n$ | $\sqrt{n/(n+d)}$ | $a_{\text{rep}}$ | $a_{\text{FD}}$ | $a_{\text{rep}}$ | $a_{\text{FD}}$ | $a_{\text{rep}}$ | $a_{\text{FD}}$ |
| 2 | 0.277 | 0.224 | 0.139 | 0.260 | 0.070 | 0.194 | 0.113 |
| 4 | 0.378 | 0.352 | 0.181 | 0.257 | 0.103 | 0.228 | 0.139 |
| 8 | 0.500 | 0.493 | 0.297 | 0.405 | 0.154 | 0.331 | 0.202 |
| 16 | 0.632 | 0.618 | 0.358 | 0.532 | 0.178 | 0.482 | 0.276 |
| 32 | 0.756 | 0.740 | 0.412 | 0.667 | 0.250 | 0.616 | 0.360 |
| 64 | 0.853 | 0.842 | 0.477 | 0.798 | 0.278 | 0.744 | 0.442 |
| 128 | 0.918 | 0.914 | 0.496 | 0.883 | 0.323 | 0.844 | 0.508 |

## H Repeated subspace draws at fixed dimension

Section 4.3 compares the two random families at $d = 24$ over five independent subspace draws each, with three optimizer seeds nested within every draw. Dense random subspace means range from $+0.47\%$ to $+1.15\%$ across draws, and random channel subset means from $-0.15\%$ to $+0.40\%$ (Table 16). A comparison based on one draw per family could therefore have reported a gap anywhere from about 0.1 to 1.3 points, which is why we repeat the draw and resample draws in the nested intervals. Omitting any one subspace seed from both families leaves the gap between $+0.50$ and $+0.74$ points, and every leave-one-draw-out interval stays above zero (Table 17). The initial draw used in the dimension sweep lies at the favourable end of the dense range, at $+1.15\%$, so the five-draw mean of $+0.75\%$ is the better estimate for the family at this dimension.

Table 16: Replication of the fixed-dimension comparison at $d = 24$ over independently drawn dense random subspaces (rotation) and random channel subsets (sparse). Optimizer seeds are nested within each subspace draw.

| family | subspace seed | optimizer seeds | mean $\Delta$ (%) | seed range |
|---|---|---|---|---|
| rotation | 20260918 | 3 | +1.15 | [+0.92, +1.35] |
| rotation | 20260919 | 3 | +1.01 | [+0.85, +1.15] |
| rotation | 20260920 | 3 | +0.51 | [+0.21, +0.72] |
| rotation | 20260921 | 3 | +0.47 | [-0.00, +0.73] |
| rotation | 20260922 | 3 | +0.61 | [+0.36, +0.94] |
| sparse | 20260918 | 3 | +0.08 | [-0.48, +0.52] |
| sparse | 20260919 | 3 | -0.15 | [-0.26, +0.05] |
| sparse | 20260920 | 3 | +0.33 | [+0.12, +0.48] |
| sparse | 20260921 | 3 | -0.06 | [-0.16, +0.14] |
| sparse | 20260922 | 3 | +0.40 | [+0.26, +0.58] |

Table 17: Paired leave-one-draw-out check of the gap between dense random subspaces and random channel subsets. Each row omits the same subspace seed from both families before nested resampling.

| omitted subspace seed | gap (points) | nested 95% interval |
|---|---|---|
| 20260918 | +0.52 | [+0.17, +0.87] |
| 20260919 | +0.50 | [+0.13, +0.88] |
| 20260920 | +0.74 | [+0.35, +1.12] |
| 20260921 | +0.65 | [+0.28, +1.04] |
| 20260922 | +0.73 | [+0.35, +1.12] |

# I What ES learns and where it stops working

We collect here the evidence behind Section 4.4, namely the reward components of every run, the structure of the learned solutions, image and video changes and released checkpoints. The section closes with the relation between reward decomposition and Gaussian smoothing.

## I.1 Reward components

For every run, Table 18 reports the relative change of the objective and of each of its three terms, computed in the same way as the objective and on the same panel and evaluations. On the validation panel, block gains raise all three terms, PickScore by $0.87\%$, HPSv2 by $1.48\%$ and the aesthetic score by $1.80\%$. Head-group gains, channel gains and the nested dense subspaces reach similar changes in the objective with smaller changes in PickScore and HPSv2, from $+0.23\%$ to $+0.59\%$ and from $-0.39\%$ to $+0.25\%$. On FLUX, block gains lower PickScore by $0.28\%$, whereas on Wan they raise PickScore by $2.16\%$ and HPSv2 by $5.43\%$. On the DrawBench panel, the prompt-alignment terms of block gains stay flat while the aesthetic score rises by $0.94\%$.

## I.2 Structure of the learned gains

Section 4.4 shows that three optimizer seeds learn similar block-gain profiles, with pairwise correlations of $0.53$, $0.62$ and $0.67$ (Figure 2). Channel and head-group gains reach similar composite improvements with 1,536 and 24 coordinates per block instead of one. We test whether their solutions are also block-level adjustments, which would be consistent with the reading that the improvable directions in these spaces are low-dimensional.

For a learned channel-gain vector $s \in \mathbb{R}^{LC}$ with displacement $\Delta = s - \mathbf{1}$, let $\bar{\Delta}_b = C^{-1} \sum_{k=1}^{C} \Delta_{b,k}$ be the mean displacement of block $b$. Projecting $\Delta$ onto the $L$-dimensional subspace of vectors that are constant within each block gives the vector of block means, and we report the fraction of

Table 18: Relative change of the objective and of its three terms for every run, from the untouched model on the stated panel, as three-seed means. Five-draw rows average all 15 runs of a family, and control rows use block gains unless labelled otherwise. The objective column repeats Tables 1, 3, 6 and 4.

| space | $d$ | objective (%) | PickScore (%) | HPSv2 (%) | aesthetic (%) |
|---|---|---|---|---|---|
| *SD3.5-Medium, validation panel* | | | | | |
| block gain | 24 | +1.68 | +0.87 | +1.48 | +1.80 |
| head gain | 576 | +1.56 | +0.38 | -0.21 | +1.82 |
| channel gain | 36,864 | +1.71 | +0.46 | -0.08 | +1.97 |
| rotation $d$=24 | 24 | +1.15 | +0.48 | +0.25 | +1.28 |
| rotation $d$=96 | 96 | +1.56 | +0.24 | -0.39 | +1.83 |
| rotation $d$=576 | 576 | +1.42 | +0.49 | -0.03 | +1.62 |
| rotation $d$=1,536 | 1,536 | +1.35 | +0.23 | -0.30 | +1.59 |
| rotation $d$=6,144 | 6,144 | +1.69 | +0.59 | +0.07 | +1.92 |
| rotation $d$=24, five draws | 24 | +0.75 | +0.13 | -0.55 | +0.90 |
| sparse $d$=24 | 24 | +0.08 | -0.05 | -1.08 | +0.15 |
| sparse $d$=24 (equal $\sigma$) | 24 | +0.08 | +0.03 | -0.06 | +0.10 |
| sparse $d$=576 (equal $\sigma$) | 576 | +0.06 | +0.23 | +0.00 | +0.04 |
| sparse $d$=24, five draws | 24 | +0.12 | -0.00 | -0.61 | +0.17 |
| low-rank $r$=1 | $r$=1 | -0.12 | -0.05 | -0.37 | -0.11 |
| matrix weights | 1.46B | -0.05 | +0.12 | -0.20 | -0.07 |
| *SD3.5-Medium block gains, optimizer and budget controls* | | | | | |
| CMA-ES | 24 | +1.78 | -0.08 | +0.28 | +2.11 |
| centered scores | 24 | +0.60 | +0.30 | +0.49 | +0.65 |
| rank scores | 24 | +1.41 | +0.61 | +1.33 | +1.52 |
| $\alpha = 0.0125$ | 24 | +1.26 | +0.60 | +1.56 | +1.34 |
| $\alpha = 0.025$ | 24 | +1.67 | +0.84 | +1.65 | +1.78 |
| $\alpha = 0.1$ | 24 | +1.88 | -1.84 | -1.80 | +2.58 |
| $G = 8$ | 24 | +1.42 | +0.15 | +2.12 | +1.56 |
| $G = 32$ | 24 | +1.96 | +0.87 | +1.78 | +2.11 |
| $G = 64$ | 24 | +1.89 | +0.88 | +1.71 | +2.04 |
| rotation $d$=1,536, $G = 8$ | 1,536 | +1.19 | +0.04 | -1.54 | +1.48 |
| rotation $d$=1,536, $G = 32$ | 1,536 | +1.50 | +0.52 | +0.41 | +1.68 |
| rotation $d$=1,536, $G = 64$ | 1,536 | +2.12 | +0.74 | +0.99 | +2.36 |
| *FLUX.1-dev, validation panel* | | | | | |
| block gain | 19 | +3.31 | -0.28 | +0.16 | +3.96 |
| channel gain | 58,368 | +2.02 | +0.11 | +1.35 | +2.32 |
| *Wan2.1-T2V, validation panel* | | | | | |
| block gain | 30 | +3.59 | +2.16 | +5.43 | +3.72 |
| channel gain | 46,080 | +1.36 | +0.21 | -0.05 | +1.58 |
| *SD3.5-Medium, DrawBench held-out panel* | | | | | |
| block gain | 24 | +0.79 | -0.01 | +0.14 | +0.94 |
| rotation $d$=24 | 24 | +0.28 | +0.03 | -0.90 | +0.38 |
| sparse $d$=24 | 24 | -0.18 | -0.15 | -0.91 | -0.15 |
| rotation $d$=6,144 | 6,144 | +0.94 | +0.15 | -0.16 | +1.13 |
| channel gain | 36,864 | +0.81 | +0.21 | -0.46 | +0.97 |
| matrix weights | 1.46B | +0.02 | +0.03 | +0.07 | +0.02 |
| CMA-ES, block gain | 24 | +0.84 | -0.51 | -1.83 | +1.19 |

displacement energy in that subspace,

$$\phi = \frac{C \sum_{b=1}^{L} \bar{\Delta}_b^2}{\sum_{b=1}^{L} \sum_{k=1}^{C} \Delta_{b,k}^2}.$$

Head-group gains are first expanded to their channels, so the same formula applies. An isotropic random displacement gives $\mathbb{E}[\phi] = 1/C$ for channel gains and $1/H$ for head-group gains, and a purely block-level solution gives $\phi = 1$. We also correlate the $L$ block means of each channel-gain and head-group-gain seed with the displacement of each block-gain seed and average the nine

correlations per space. We measure consistency across seeds by correlating block means between seeds of the same space and, separately, the within-block residuals $\Delta_{b,k} - \bar{\Delta}_b$. A large $\phi$ with block means that follow the learned block gains would indicate that channel-gain ES mostly finds the adjustment that block gains can express, whereas reproducible residuals would indicate structure beyond it. As for the block-gain profiles, these correlations are descriptive because blocks are ordered and all runs share the prompt pool.

Table 19 does not support the block-level reading. The block-mean component carries $0.08\%$ of the channel-gain displacement energy and $3.5\%$ of the head-group displacement energy, close to the isotropic $0.07\%$ and $4.2\%$. Block means correlate with the learned block gains at $0.18$ for channel gains and $0.28$ for head-group gains on average, with ranges that reach values near zero. Across seeds, channel-gain block means correlate at $0.11$ and within-block residuals at $0.002$, against $0.60$ for the block gains themselves. FLUX and Wan channel gains show the same pattern. Channel-gain ES therefore neither recovers the block-level adjustment nor converges to a common solution, although its composite improvements agree across seeds.

Table 19: Block-level content of learned gains. $\phi$ is the fraction of displacement energy in the block-mean subspace, averaged over three seeds, and isotropic $\phi$ is its expectation for a random displacement. Correlations give the mean and range over pairs. Block-level seed correlations compare block means, or the block gains themselves in block rows, and residual correlations compare within-block deviations. FLUX uses its 19 active dual-stream blocks.

| | | | | | corr. with | seed corr. | |
|---|---|---|---|---|---|---|---|
| backbone | space | per block | $\phi$ | isotropic $\phi$ | block gains | block level | residual |
| SD3.5-M | block gain | 1 | – | – | – | 0.60 [0.53, 0.67] | – |
| | head gain | 24 | 0.0354 | 0.0417 | 0.28 [0.06, 0.37] | 0.28 [0.19, 0.47] | 0.065 [0.018, 0.108] |
| | channel gain | 1,536 | 0.0008 | 0.0007 | 0.18 [-0.34, 0.48] | 0.11 [0.04, 0.17] | 0.002 [-0.002, 0.005] |
| FLUX.1 | block gain | 1 | – | – | – | 0.43 [0.37, 0.50] | – |
| | channel gain | 3,072 | 0.0003 | 0.0003 | 0.14 [-0.21, 0.37] | -0.13 [-0.62, 0.39] | 0.001 [-0.003, 0.004] |
| Wan2.1 | block gain | 1 | – | – | – | 0.77 [0.72, 0.84] | – |
| | channel gain | 1,536 | 0.0007 | 0.0007 | -0.07 [-0.24, 0.11] | 0.07 [-0.21, 0.30] | -0.001 [-0.005, 0.002] |

### I.3 Image changes and within-prompt diversity

In the matched samples of Figure 5, block-gain tuning keeps the subject and layout and changes color, contrast and local texture. We also measure whether samples for the same prompt become more alike.

**Within-prompt diversity.** On 50 frozen DrawBench prompts with four matched latents, LPIPS distance falls from $0.6995$ to $0.6805$, a paired change of $-0.0190$ $[-0.0347, -0.0025]$ (Table 20). DINOv2 distance changes by $-0.0183$ $[-0.0389, +0.0025]$. The LPIPS result shows reduced low-level variation among samples for the same prompt, whereas the DINOv2 interval includes both decreases and increases. These measurements describe diversity alongside reward improvement under the composite objective.

Table 20: Within-prompt diversity on 50 frozen DrawBench prompts with four matched latents per prompt. Intervals resample optimizer seeds and crossed prompts.

| metric | base | block gain | paired $\Delta$ [95%] |
|---|---|---|---|
| LPIPS distance | 0.6995 | 0.6805 | -0.0190 [-0.0347, -0.0025] |
| DINOv2 cosine distance | 0.3711 | 0.3527 | -0.0183 [-0.0389, +0.0025] |

### I.4 Temporal change in Wan videos

**Descriptive Wan temporal metrics.** Across two matched prompt-latent pairs, including the one in Figure 6, the tuned means are lower for adjacent pixel, LPIPS and DINOv2 distances and for the second frame difference (Table 21). Lower adjacent-frame change can mean steadier frames with

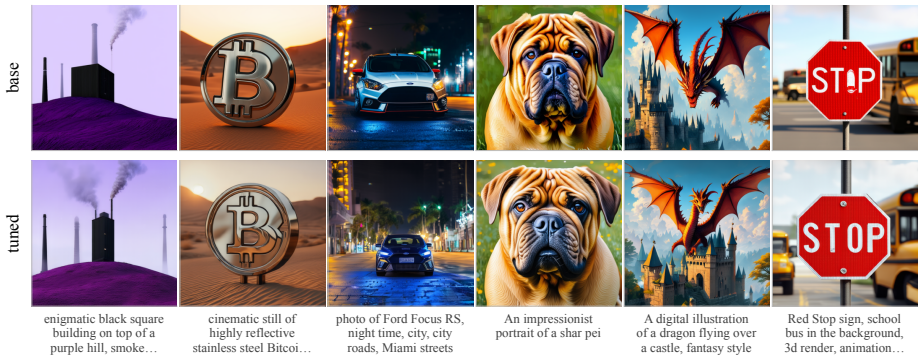


Figure 5: Matched SD3.5-Medium samples before and after block-gain tuning. Each column shares its prompt and initial latent, selected before generation. The dominant object and layout remain while color, contrast and local texture change. Mean absolute pixel difference is 0.158. This is descriptive rather than a fidelity test.

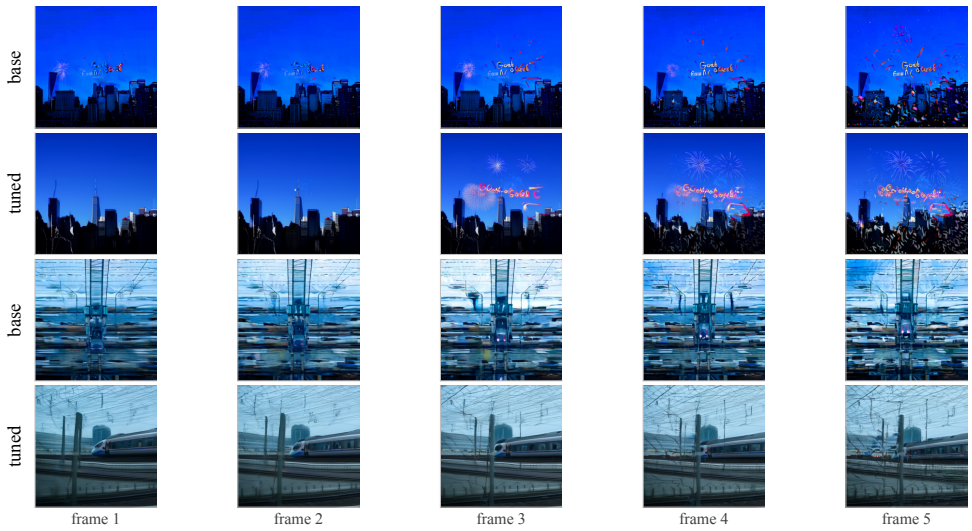


Figure 6: Frozen-protocol Wan2.1-T2V frame strips from the seed-1234 block-gain checkpoint. Base and tuned rows share a prompt and initial latent. Columns sample five of the 17 decoded frames in temporal order. The figure is descriptive and carries no temporal-quality claim.

the same motion or simply less motion, and these metrics do not separate the two. Because the Wan reward averages per-frame image scores, nothing in the objective rewards motion. Two examples cannot establish temporal quality or generalization.

Table 21: Temporal metrics averaged over two frozen-protocol Wan prompt-latent pairs. Lower values indicate less frame-to-frame change, not necessarily higher video quality.

| metric | base | tuned | tuned − base |
|---|---|---|---|
| adjacent pixel RMSE | 0.0765 | 0.0612 | -0.0153 |
| adjacent LPIPS | 0.0814 | 0.0575 | -0.0239 |
| adjacent DINOv2 distance | 0.0397 | 0.0234 | -0.0163 |
| second-difference RMSE | 0.1193 | 0.0911 | -0.0282 |

### I.5 Comparison with released checkpoints

Table 22: Publicly released checkpoints evaluated directly on one panel, weight format and sampler. The objective sums PickScore, HPSv2 and aesthetic reward. Components are shown at the better of the two reported CFG values for each row, which is CFG 4.5 for the base model, Flow-GRPO and block gains and CFG 1.0 for DiffusionNFT.

| | objective | | components | | |
|---|---|---|---|---|---|
| model | CFG 4.5 | CFG 1.0 | Pick | HPSv2 | aesth. |
| SD3.5-M base | 7.122 | 6.538 | 0.839 | 0.296 | 5.987 |
| Flow-GRPO | 7.060 | 6.404 | 0.837 | 0.278 | 5.945 |
| DiffusionNFT | 6.657 | 7.880 | 0.905 | 0.357 | 6.618 |
| block gain, ours | 7.237 | 6.047 | 0.842 | 0.296 | 6.100 |

**Guidance changes the ranking, while the gap remains large.** At guidance 4.5, block gains score highest among the four models in Table 22, while Flow-GRPO and DiffusionNFT (Zheng et al., 2026a) score below the untouched model. At guidance 1.0, the top and bottom models exchange positions, and block gains trained at guidance 4.5 fall below the untouched model, 6.047 against 6.538. DiffusionNFT intentionally trains without classifier-free guidance, so its reward signal replaces rather than supplements guidance. A table at a single guidance value would therefore misrepresent the methods. Relative to the untouched model at its own best guidance, our block gains improve by $+1.6\%$, whereas DiffusionNFT improves by $+10.6\%$, a ratio of approximately one to seven. This ratio increases to one to thirteen when each method is evaluated at the guidance used for its training. These are released checkpoints, so the comparison describes the evaluated models and does not attribute their gap to search dimension.

**The terms move differently.** Comparing each row at its better reported CFG, block gains raise the aesthetic score by $1.9\%$ while PickScore rises by $0.4\%$ and HPSv2 is unchanged, whereas DiffusionNFT raises PickScore, HPSv2 and the aesthetic score by $7.9\%$, $20.6\%$ and $10.5\%$.

### I.6 Reward decomposition and Gaussian smoothing

An objective that sums several reward models can increase while some of its terms stay flat or decline, and Table 18 shows this in several runs. The clearest are block gains at the largest step size, where PickScore and HPSv2 fall by $1.84\%$ and $1.80\%$ at a composite change of $+1.88\%$, and CMA-ES on DrawBench, where they fall by $0.51\%$ and $1.83\%$ at $+0.84\%$. FLUX block gains lower PickScore by $0.28\%$, and on SD3.5 the higher-dimensional spaces leave HPSv2 near zero or below it.

**Relation to Gaussian smoothing.** Evolution strategies optimize $J_\sigma$ rather than $J$. For a smooth local surrogate near a maximum, the $\mathcal{O}(\sigma^2)$ term of Equation 3 favours lower summed curvature in the searched coordinates. This is a local geometric effect whose remainder can grow with dimension. RSA-FT (Kim and Sim, 2026) uses adversarial perturbations to reduce reward hacking, providing a related geometric motivation with a different objective. The reward decomposition and diversity measurements in this appendix characterize the observed model changes. Isolating the effect of Gaussian smoothing on these outcomes would require a matched unsmoothed comparison.

## J Practical considerations

### J.1 Memory and optimizer controls

Residual-scaling spaces keep no optimizer state beyond their gains and peak at 26.69 GiB per GPU, whereas the low-rank and matrix-weight engines hold 10.88 GiB of fp32 master state and peak at 37.58 GiB, and each of the four measured spaces takes about 82 seconds per generation at the same global query budget (Table 23). Full-size master weights therefore remove the memory benefit of low-rank perturbations in this implementation, and the storage measurements in the next subsection show why these weight-space engines keep fp32 master weights. Within block gains, CMA-ES (Hansen and Ostermeier, 2001) reaches $+1.78\%$ against $+1.68\%$ for isotropic ES, with overlapping

seed ranges. Score shaping moves the mean from $+0.60\%$ with plain centering to $+1.68\%$ with $z$-scores, an effect of the same order as the gaps between search spaces, and step sizes from $0.0125$ to $0.1$ give means from $+1.26\%$ to $+1.88\%$ (Table 24), although the largest step size lowers PickScore and HPSv2 (Table 18).

Table 23: Per-GPU resources for one generation at population 16. The common inference floor is 19.83 GiB, and the number of function evaluations is global.

| space | state GiB | peak GiB | over infer. GiB | global NFE | sec./gen. |
|---|---|---|---|---|---|
| block gain | 0.00 | 26.69 | 6.86 | 6400 | 82.3 |
| channel gain | 0.00 | 26.69 | 6.86 | 6400 | 82.2 |
| low-rank $r$=1 | 10.88 | 37.58 | 17.74 | 6400 | 82.2 |
| matrix weights | 10.88 | 37.58 | 17.74 | 6400 | 82.1 |

Table 24: Block-gain optimizer controls on SD3.5-Medium as mean validation changes. Brackets are observed optimizer-seed ranges, and the reference run uses isotropic ES, $z$-score shaping and $\alpha = 0.05$.

| control | setting | seeds | reward $\Delta$ (%) | seed range |
|---|---|---|---|---|
| optimizer | isotropic ES | 3 | +1.68 | [+1.44, +1.91] |
| | CMA-ES | 3 | +1.78 | [+1.47, +2.25] |
| score shaping | centered | 3 | +0.60 | [+0.49, +0.79] |
| | rank | 3 | +1.41 | [+1.08, +1.65] |
| | $z$-score | 3 | +1.68 | [+1.44, +1.91] |
| step size | $\alpha = 0.0125$ | 2 | +1.26 | [+1.19, +1.34] |
| | $\alpha = 0.025$ | 2 | +1.67 | [+1.59, +1.74] |
| | $\alpha = 0.05$ | 3 | +1.68 | [+1.44, +1.91] |
| | $\alpha = 0.1$ | 2 | +1.88 | [+1.14, +2.63] |

### J.2 Finite precision in weight space

We give here the measurements behind the precision summary in Section 4.5. Weight-space comparisons depend on how the requested perturbation is represented numerically. For a requested displacement $\delta$ and a storage format with rounding map $Q$, the displacement actually stored is

$$\delta_Q = Q(\theta_0 + \delta) - Q(\theta_0). \tag{22}$$

We first measure its magnitude and direction, then evaluate reward responses. End-to-end probes store and run the whole model in the tested format. Write-back probes keep the model, VAE, scorer and forward arithmetic in fp32 and evaluate $\theta_0 + \delta_Q$, so only the stored displacement changes with the format.

**Storage fidelity.** Section 4.5 and Figure 3 summarize the storage measurements, and Table 25 gives the full grid up to $\sigma_{\text{rel}} = 3 \times 10^{-3}$. At $10^{-2}$ and $3 \times 10^{-2}$, bf16 storage realizes $1.01$ and $1.00$ of the intended norm and changes $83.3\%$ and $94.3\%$ of entries, and fp16 and fp32 storage realize the full norm.

**Reward responses.** For rewards $J_+$, $J_-$ and $J_0$ at positive, negative and zero perturbations, define $r_{\text{odd}} = (J_+ - J_-)/2$ and $r_{\text{even}} = (J_+ + J_-)/2 - J_0$, summarized by $\rho_{\text{eo}} = |\mathbb{E}[r_{\text{even}}]|/\mathbb{E}|r_{\text{odd}}|$. All write-back formats share one fp32 unperturbed reward, $J_0 = 7.4795$, while each end-to-end format has its own. The write-back ratios at $\sigma_{\text{rel}} = 3 \times 10^{-3}$ are $0.49$, $0.49$ and $0.39$ for bf16, fp16 and fp32 storage, and $0.37$, $0.31$ and $0.33$ at $10^{-2}$, with overlapping intervals (Figure 3, Table 26). End-to-end bf16, which also stores the base weights and runs the forward pass in bf16, gives $2.56$ at $3 \times 10^{-3}$ (Table 27), so quantizing the displacement alone does not reproduce the end-to-end response.

Up to $10^{-3}$, however, $\rho_{\text{eo}}$ mostly reflects $J_0$. Even fp32 write-back, which alters the requested weights only in the last bit, gives $4.27$ at $10^{-4}$ because its arms sit $0.008$ below $J_0$, and perturbations

Table 25: Storage diagnostics on 43 regularly spaced weight tensors. The norm ratio divides the stored perturbation norm by the intended one, cosine compares intended and stored directions, asymmetry is zero when the stored positive and negative arms remain exact opposites, and the last column is the fraction of weights whose stored value changes at all.

| dtype | $\sigma_{\text{rel}}$ | $\|\delta_{\text{store}}\|/\|\delta\|$ | $\cos(\delta, \delta_{\text{store}})$ | asymmetry | entries changed |
|---|---|---|---|---|---|
| bf16 | 0.0001 | 0.292 | 0.219 | 0.051 | 0.026 |
| | 0.0003 | 0.502 | 0.378 | 0.052 | 0.078 |
| | 0.001 | 0.840 | 0.651 | 0.048 | 0.245 |
| | 0.003 | 1.039 | 0.893 | 0.036 | 0.547 |
| fp16 | 0.0001 | 0.776 | 0.595 | 0.049 | 0.200 |
| | 0.0003 | 1.024 | 0.855 | 0.038 | 0.480 |
| | 0.001 | 1.019 | 0.980 | 0.017 | 0.795 |
| | 0.003 | 1.002 | 0.998 | 0.006 | 0.929 |
| fp32 | 0.0001 | 1.000 | 1.000 | 0.000 | 1.000 |
| | 0.0003 | 1.000 | 1.000 | 0.000 | 1.000 |
| | 0.001 | 1.000 | 1.000 | 0.000 | 1.000 |
| | 0.003 | 1.000 | 1.000 | 0.000 | 1.000 |

Table 26: Write-back-only reward-response sweep. The model, VAE, scorer and forward arithmetic stay in fp32, and only the displacement applied to fp32 weights is quantized to the stated storage grid. Brackets bootstrap directions.

| write-back | $\sigma_{\text{rel}}$ | $N$ | $\mathbb{E}\|r_{\text{odd}}\|$ | $\bar{r}_{\text{even}}$ | $\rho_{\text{eo}}$ [95%] |
|---|---|---|---|---|---|
| bf16 | 0.0001 | 48 | 1.56e-03 | -3.10e-03 | 1.99 [1.49, 2.64] |
| | 0.0003 | 48 | 2.71e-03 | -1.03e-02 | 3.81 [3.04, 4.94] |
| | 0.001 | 48 | 5.28e-03 | -6.54e-03 | 1.24 [0.91, 1.63] |
| | 0.003 | 48 | 9.61e-03 | -4.76e-03 | 0.49 [0.27, 0.78] |
| | 0.01 | 48 | 1.94e-02 | -7.22e-03 | 0.37 [0.13, 0.61] |
| fp16 | 0.0001 | 48 | 2.34e-03 | -7.32e-03 | 3.13 [2.56, 3.84] |
| | 0.0003 | 48 | 3.30e-03 | -1.02e-02 | 3.11 [2.51, 3.90] |
| | 0.001 | 48 | 4.50e-03 | -6.31e-03 | 1.40 [0.97, 1.93] |
| | 0.003 | 48 | 8.77e-03 | -4.33e-03 | 0.49 [0.24, 0.79] |
| | 0.01 | 48 | 2.11e-02 | -6.54e-03 | 0.31 [0.07, 0.53] |
| fp32 | 0.0001 | 48 | 1.91e-03 | -8.15e-03 | 4.27 [3.36, 5.73] |
| | 0.0003 | 48 | 3.24e-03 | -9.53e-03 | 2.94 [2.35, 3.74] |
| | 0.001 | 48 | 6.02e-03 | -6.27e-03 | 1.04 [0.73, 1.39] |
| | 0.003 | 48 | 9.40e-03 | -3.67e-03 | 0.39 [0.16, 0.67] |
| | 0.01 | 48 | 2.10e-02 | -7.00e-03 | 0.33 [0.10, 0.54] |

of one millionth of each tensor's RMS already move every evaluated reward (Table 28). Settings that shift $J_0$ therefore shift these ratios. The end-to-end sweep shares prompts, latents and its 24 directions with the write-back sweep but used PyTorch's default TF32 convolutions. Restoring that setting reproduces its unperturbed reward of 7.4744 and its fp32 row at $10^{-3}$ exactly, and a paired fp32 probe reproduces every write-back row.

Direction-paired format contrasts remove $J_0$ (Table 29). Their arm means agree with fp32 write-back except for bf16 at $10^{-4}$, and bf16 odd responses are uncorrelated with the fp32 responses of the same directions up to $10^{-3}$. The correlation reaches $0.77$ at $3 \times 10^{-3}$, against $0.95$ for fp16 and at least $0.985$ for last-bit differences alone. We therefore read $\rho_{\text{eo}}$ at small scales as descriptive and base the precision finding on the storage measurements and these paired contrasts.

**Shared settings of the reward probes.** Every reward-response probe uses the SD3.5-Medium checkpoint, four prompts and their latents drawn with seed 1234, the 25-step sampler and guidance 4.5. Direction seeds come from one generator, so smaller direction sets are prefixes of larger ones.

Table 27: End-to-end reward-response sweep, in which weight storage and forward arithmetic vary together. Columns and brackets follow Table 26, and refined boundary rows use more directions.

| dtype | $\sigma_{\mathrm{rel}}$ | $N$ | $\mathbb{E}\lvert r_{\mathrm{odd}}\rvert$ | $\bar{r}_{\mathrm{even}}$ | $\rho_{\mathrm{eo}}$ [95%] |
|---|---|---|---|---|---|
| bf16 | 0.0001 | 24 | 8.34e-03 | +2.82e-02 | 3.39 [2.73, 4.22] |
| | 0.0003 | 24 | 9.27e-03 | +2.68e-02 | 2.89 [1.90, 4.71] |
| | 0.001 | 24 | 1.09e-02 | +2.76e-02 | 2.53 [1.92, 3.54] |
| | 0.003 | 24 | 1.43e-02 | +3.67e-02 | 2.56 [1.86, 3.88] |
| | 0.01 | 48 | 2.30e-02 | +4.05e-02 | 1.76 [1.16, 2.62] |
| fp16 | 0.0001 | 24 | 4.88e-03 | +6.62e-03 | 1.36 [0.97, 1.99] |
| | 0.0003 | 24 | 5.45e-03 | +1.02e-02 | 1.87 [1.33, 2.72] |
| | 0.001 | 24 | 5.44e-03 | +1.20e-02 | 2.20 [1.58, 3.10] |
| | 0.003 | 96 | 9.52e-03 | +1.13e-02 | 1.19 [0.99, 1.42] |
| fp32 | 0.0001 | 24 | 2.42e-03 | -2.32e-03 | 0.96 [0.53, 1.52] |
| | 0.0003 | 24 | 3.44e-03 | -5.25e-03 | 1.53 [1.03, 2.23] |
| | 0.001 | 24 | 5.68e-03 | +5.74e-05 | 0.01 [0.01, 0.56] |
| | 0.003 | 24 | 7.88e-03 | +9.86e-04 | 0.13 [0.01, 0.50] |
| | 0.01 | 48 | 2.10e-02 | -2.20e-03 | 0.10 [0.01, 0.32] |

Table 28: Identity controls for the reward probes on one fixed batch of four prompts. Arm shifts are $(J_+ + J_-)/2 - J_0$ averaged over 48 directions, with the standard deviation of individual arms about $J_0$.

| check | numerics | $\sigma_{\mathrm{rel}}$ | result |
|---|---|---|---|
| unperturbed reward, repeated | TF32 off | – | 7.47946, identical |
| $\sigma = 0$ through both perturbation paths | TF32 off | 0 | equal to $J_0$ |
| write-back sweep, Table 26 | TF32 off | $10^{-4}$–$10^{-2}$ | reproduced exactly |
| fp32 write-back vs. direct | TF32 off | $10^{-4}$–$10^{-2}$ | arm RMS gap $\leq 0.0011$, odd corr. $\geq 0.985$ |
| last-bit differences in weights | CPU, fp32 | $10^{-4}$–$10^{-2}$ | 0.05%–1.3% of entries |
| end-to-end fp32 row, Table 27 | cuDNN TF32 | $10^{-3}$ | $J_0 = 7.47440$, reproduced exactly |
| end-to-end bf16, unperturbed | bf16 model | – | 7.40996, identical, equals Table 27 |
| end-to-end fp16, unperturbed | fp16 model | – | 7.46367, identical, equals Table 27 |
| arm shift from $J_0$ [95%], s.d. | TF32 off | $10^{-6}$ | $+0.0006$ $[+0.0005, +0.0007]$, 0.0006, 0 of 96 arms at $J_0$ |
| | TF32 off | $10^{-5}$ | $-0.0011$ $[-0.0014, -0.0008]$, 0.0016, 0 of 96 arms at $J_0$ |
| | TF32 off | $10^{-4}$ | $-0.0083$ $[-0.0090, -0.0076]$, 0.0038 |

Table 29: Direction-paired storage contrasts under fp32 forward computation. Arm shift is the mean over 48 directions of $(J_+ + J_-)/2$ under the stated storage minus the same quantity under fp32 write-back. Odd corr. is the correlation of $r_{\mathrm{odd}}$ with its fp32 write-back value across directions.

| | bf16 write-back | | fp16 write-back | |
|---|---|---|---|---|
| $\sigma_{\mathrm{rel}}$ | arm shift $\times 10^3$ [95%] | odd corr. | arm shift $\times 10^3$ [95%] | odd corr. |
| 0.0001 | $+5.05$ $[+4.25, +5.85]$ | $-0.01$ | $+0.83$ $[-0.32, +1.93]$ | 0.18 |
| 0.0003 | $-0.80$ $[-1.96, +0.40]$ | 0.07 | $-0.71$ $[-1.85, +0.49]$ | 0.62 |
| 0.001 | $-0.27$ $[-2.28, +1.71]$ | $-0.03$ | $-0.04$ $[-1.45, +1.38]$ | 0.70 |
| 0.003 | $-1.09$ $[-3.52, +1.45]$ | 0.77 | $-0.66$ $[-1.72, +0.49]$ | 0.95 |
| 0.01 | $-0.22$ $[-3.53, +3.05]$ | 0.89 | $+0.46$ $[-0.77, +1.77]$ | 0.98 |

The end-to-end sweep of Table 27 predates an explicit TF32 switch and ran with PyTorch's default, which enables TF32 in cuDNN convolutions only. The write-back sweep of Table 26 and all later probes disable TF32 throughout. Table 28 verifies both settings by reproduction and measures the evaluation floor that makes small-scale even responses depend on them. Table 29 compares storage formats direction by direction under one fp32 model, which removes $J_0$ from the contrast. Table 30 reports an earlier sweep on a separate prompt batch, with unperturbed reward 7.2370, and is kept for the shape of its response spread.

Table 30: Reward response on fixed prompts while varying weight format and requested sigma. Direction counts differ (32 for bf16, 24 for fp32), so the table establishes shape rather than a matched variance comparison.

| weight dtype | $\sigma$ | std($\Delta R$) | skew |
|---|---|---|---|
| bf16 | 0.0001 | 1.77e-02 | -0.31 |
| | 0.0003 | 1.62e-02 | -0.42 |
| | 0.001 | 1.85e-02 | -0.20 |
| | 0.003 | 2.30e-02 | -0.56 |
| | 0.01 | 3.40e-02 | -0.16 |
| fp32 | 0.00001 | 1.52e-03 | +0.16 |
| | 0.0001 | 4.67e-03 | +0.38 |
| | 0.0003 | 6.95e-03 | +0.30 |
| | 0.001 | 9.50e-03 | +0.29 |
| | 0.003 | 1.60e-02 | -0.19 |

### J.3 Common random numbers

The estimator variance consists of a direction term, which cannot be removed, and a reward-noise term. In agent settings, the latter dominates, and agentic methods spend most of their rollout budget reducing it, typically using 16 to 32 tasks per direction (Zheng et al., 2026b). With a deterministic ODE sampler and deterministic scorer, fixing the prompt set and initial latents makes the reward a deterministic function of the weights. The reward-noise term therefore vanishes rather than merely decreasing. Repeated evaluations return bitwise identical values, which verifies this property empirically. Determinism does not make the reward smooth in the weights. Perturbations far below any smooth effect still move it by about $10^{-3}$ (Table 28), which affects even responses measured against one unperturbed evaluation but not the noise term.

This result has two important scope limitations. First, it removes latent-sampling noise conditional on a fixed prompt batch. The prompt batch remains a finite sample of the objective in Equation 2, and Appendix D.6 models this residual variation. Second, the guarantee is specific to this implementation. A stochastic detector, a remote judge with version drift, or nondeterministic kernels would invalidate it. This limitation is relevant because such rewards are among the motivations for forward-only optimization.

The same property motivates our estimator choice. Mirrored pairs cancel the constant and even-order terms exactly when both arms share identical randomness, but only partially otherwise. We therefore allocate two evaluations to each direction, whereas agentic methods allocate one.

## K Evaluating a cross-batch stopping rule

Before training, we designed a lower-cost test of whether evolution strategies could succeed in a given search space and specified its verdict as a stopping criterion in the experimental plan. The test draws $n$ directions, evaluates each direction on two disjoint prompt batches under common random numbers, and computes the correlation between the two sets of antithetic central differences. The proposed stopping rule assumed that a correlation near zero indicated that optimization would fit only the current batch. The test costs approximately one percent of a training run.

The test does not predict the training outcomes (Table 31). With $n = 96$ directions in fp32, the probe reports $\rho = 0.126$ for block gains, which is 1.2 standard errors from zero. Nevertheless, training in the same search space increases held-out reward for every seed. All three evaluated search spaces lie within a band in which the probe cannot distinguish them, while their training outcomes differ by an order of magnitude (Figure 7). Under the criterion specified in advance, this result would have terminated the study.

Proposition 2 identifies a possible attenuation mechanism for this discrepancy. The probe tests whether the effect of an individual direction transfers across batches within one generation, whereas evolution strategies require the update averaged over many generations to have positive alignment

Table 31: Comparison of the lower-cost feasibility test with the training outcome. The probe correlation is indistinguishable from zero in every search space, including the space in which training reliably improves held-out reward. At $n = 96$, the correlation standard error is $0.104$, so $|\rho|$ must exceed approximately $0.21$ to be distinguishable from zero.

| space | count ($d$ or $q$) | probe $\rho$ | sign-match rate | reward $\Delta$ (%) |
|---|---|---|---|---|
| block gain | $d = 24$ | +0.126 | 0.59 | +1.68 |
| low-rank $r$=1 | $q = 1{,}470{,}000$ | +0.120 | 0.49 | -0.12 |
| matrix weights | $d = 1{,}460{,}000{,}000$ | -0.048 | 0.43 | -0.05 |

with $g_u$. Under the model, fresh batches make the batch-specific component average toward zero while the shared component accumulates. Increasing $B$ from 1 to 32 leaves the measured curve unresolved within noise, so this experiment does not distinguish attenuation from an absent shared component.

We present this finding as a methodological note rather than a general result. It concerns one gate on one task. Establishing that such gates are systematically misleading would require multiple tasks and search spaces. The evidence supports only the narrower conclusion that we cannot recommend this particular test, because applying its prespecified criterion would have terminated the study.

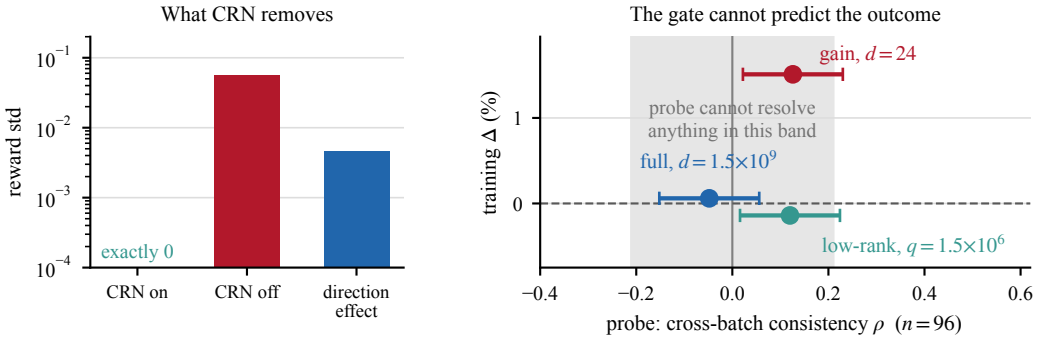


Figure 7: Left: the variance term removed by common random numbers compared with the measured direction effect. Right: probe correlation for each search space on the horizontal axis and its training outcome on the vertical axis. Shading denotes the region in which the probe cannot distinguish a correlation from zero.

## L Accumulation over generations

Evolution strategies accumulate updates across generations. The following toy model illustrates why this accumulation matters. Suppose generation $t$ draws a new prompt batch and produces an update with a fixed useful component $s$ and a batch-specific component $\eta_t$ that has zero mean, variance $v_\eta$, and is independent across $t$. After $T$ generations, the accumulated displacement has signal $Ts$ and noise standard deviation $\sqrt{Tv_\eta}$. The signal-to-noise ratio of the trajectory is therefore

$$\mathrm{SNR}(T) \;=\; \frac{T\,s}{\sqrt{T v_\eta}} \;=\; \sqrt{T}\,\frac{s}{\sqrt{v_\eta}}. \tag{23}$$

A per-generation ratio that is too small to detect can therefore still yield a detectable accumulated signal after one hundred generations.

The model holds $s$ and the update direction fixed and assumes independent per-generation noise, whereas an actual run changes the parameters, gradient and reward $z$-scores and draws its batches from one prompt pool. Its signal-to-noise ratio also differs from the $\rho(B)$ of Proposition 2, which correlates scalar direction effects within a single generation. The model therefore motivates, rather than proves, the hypothesis that a single-generation transfer test is an inappropriate feasibility criterion, and Appendix K tests this hypothesis empirically.