REFRESH OR REALIZE? COMPUTE ALLOCATION IN DRIFTING MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Drifting Models train a one-step generator by recomputing a finite-sample drift field at every iteration and taking an optimizer step toward the drifted target. The field says how generated samples should move, but the step is taken in parameters shared by all samples, so the motion the network actually makes need not match the motion it was given. This leaves a basic training question open: should extra compute go into fitting the current target more closely, or into recomputing the field? We study it on ImageNet 256×256. Holding the target fixed for k optimizer steps and measuring the realized displacement, we find that deeper fitting does bring the network closer to the frozen target, and that the number of steps needed before it makes any net progress drops from about sixteen early in training to one later on. When the extra steps come for free, $k = 2$ also lowers FID. Once they are paid for, the result flips: at approximately matched measured wall-clock, spending the budget on fresh fields gives lower FID than deeper fitting, on both training seeds. The target itself shows why a fresh field is worth so much. Redrawing the finite support rotates its direction far more than a parameter update does (cosine ≈ 0.3–0.6 against ≈ 0.95), and a correction that is optimal in field space is not reliably better in FID than a parameter-free one. For Drifting, fitting each target well and spending compute well are different goals.

1 INTRODUCTION

Drifting Models (Deng et al., 2026) train a one-step generator around a strikingly simple rule. A vector field $V_{p,q}$, computed from generated samples and real data, says how each generated sample should move so that the generated distribution q drifts toward the data distribution p,

$$x_{i+1} = x_i + V_{p,q_i}(x_i), \tag{1}$$

and training carries the rule out by freezing the drifted point $x + V$ as a regression target and taking an optimizer step toward it. Read this way, a training iteration moves samples along the field.

In practice the move is made in parameter space. The optimizer changes one set of weights shared by every sample in the batch, so all outputs move together, and to first order a single gradient step displaces the stacked outputs $\mathbf{H}$ by

$$\Delta\mathbf{H}_1 = \eta\mathbf{K}\mathbf{V}, \qquad \mathbf{K} = \mathbf{J}\mathbf{J}^\top, \quad \mathbf{J} = \partial\mathbf{H}/\partial\boldsymbol{\theta}. \tag{2}$$

Sample i thus moves by a mixture $\eta \sum_j \mathbf{K}_{ij}\mathbf{V}_j$ of the motions prescribed for the whole batch rather than by its own $\mathbf{V}_i$, and only in special geometries does $\eta\mathbf{K}\mathbf{V}$ coincide with $\mathbf{V}$. In general the motion the field prescribes and the motion one step executes differ (Fig. 1a), and Eq. (1) is silent about the difference. This paper is about that difference: our object of study is the training loop that estimates the existing field from data and realizes it with an optimizer, rather than the definition of the field.

The most direct way to study this gap is to remove the target as a variable. We freeze one target and take k optimizer steps toward it instead of one; since the target cannot change, whatever happens to the executed motion is due to fitting alone. We use this frozen-target construction purely as a probe of realization. Measured on ImageNet (Deng et al., 2009) at 256×256 with an exact decomposition of the frozen-target residual (Section 3), deeper fitting does close part of the gap (Fig. 1b), and closes it more readily as training proceeds: the depth at which fitting first makes net progress toward the target falls from about sixteen steps at checkpoint 5k to a single step by 20k.

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.

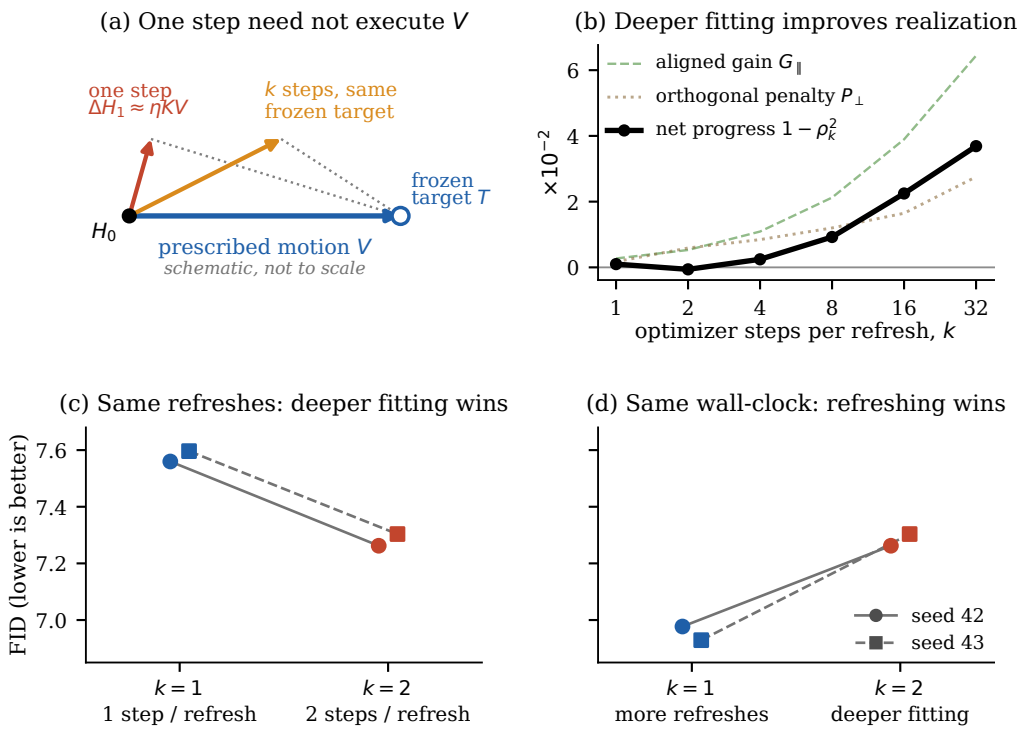


Figure 1: **A field can become easier to realize and still not be the best place to spend compute.** **(a)** Drifting prescribes a motion V toward a frozen target; one optimizer step executes a tangent-filtered motion $\Delta H_1 \approx \eta KV$ that need not coincide with V; repeated steps against the same frozen target move closer (dotted: remaining residual). **(b)** Measured on ImageNet 256×256 at the state from which all our downstream runs start: net progress toward the frozen target grows with depth apart from a small dip at $k = 2$. **(c)** With the number of refreshes fixed, $k = 2$ takes two optimizer steps per refresh and reaches lower FID on both training seeds. **(d)** At approximately matched measured wall-clock, $k = 1$ affords more refreshes and reaches lower FID on both seeds. Panels (c) and (d) share one FID axis.

Whether that improvement is worth its cost is a separate question. Every Drifting iteration spends compute on two things: refreshing the field, by drawing a new finite support of real examples and building a new target, and realizing it with optimizer steps. Since deeper fitting demonstrably works, a case against it cannot rest on an optimizer that fails to fit. When the extra steps cost nothing, deeper fitting wins: at an equal number of refreshes, two steps per refresh reach lower FID than one on both training seeds (Fig. 1c). When the steps are paid for, the ordering reverses. At approximately matched measured wall-clock, the run that spends its budget on more refreshes, one step each, reaches lower FID on both seeds (Fig. 1d), although it used 9.8% less time than its counterpart on one seed and 1.9% more on the other.

A local, matched-compute version of the same decision points the same way. Starting from a common parameter state, refreshing the target before the next update beats a second update toward the stale target in all 40 probes we ran, across three support sizes and three training stages. The same probes show where the value of refreshing comes from. At fixed parameters, a newly drawn support rotates the target direction far more (cosine 0.31–0.59) than one parameter update does at fixed support (0.94–0.98), so a second step against an old target keeps fitting one particular sample of a highly variable quantity, while a refresh replaces it. The size of that rotation is not a calibrated predictor of the local gain from refreshing, which stays nearly constant across support sizes that rotate the target by very different amounts (Section 5).

The target also depends strongly on how it is constructed. A same-budget correction to the target, calibrated by a held-out audit that follows a clean $N^{-1/2}$ law, is not reliably better in FID than a parameter-free split average, and removing a nominal 10^{-6} floor from the production affinity changes the aggregated field by 12%.

These measurements describe how the empirical field, its finite support and shared neural optimization interact inside the Drifting training loop. The executed motion is a tangent-filtered version of the prescribed one; how much of the prescription is carried out depends on realization depth, training state and batch geometry; executing a given target better is not the same as spending compute better; and the target itself is governed by its finite support and by implementation details. Refresh frequency, realization depth, support construction and affinity design are separate design axes of Drifting training, which Eq. (1) folds into a single notion of more optimization. Our contributions are:

1. an exact decomposition of frozen-target progress into aligned gain and orthogonal penalty, which makes the realization gap of Drifting measurable, and the finding on ImageNet that deeper fitting closes it, increasingly so over training;
2. the refresh–realize reversal: extra fitting lowers FID when it is free, whereas under approximately matched measured wall-clock refreshing wins on both training seeds and in all 40 local matched-compute probes;
3. evidence that support refresh, rather than the parameter update, dominates how the empirical target changes between steps; that field-space optimality of the target does not reliably transfer to FID; and that a nominal affinity floor materially changes the production field.

2 Background: Drifting as Refresh and Realize

Following Deng et al. (2026), let $x = f_\theta(\epsilon)$ with $\epsilon \sim p_\epsilon$ and $q_\theta = (f_\theta)_{\#} p_\epsilon$. Drifting uses positive samples $y^+ \sim p$ and negative samples $y^- \sim q$, a kernel $k(x, y) = \exp(-\|x - y\|/\tau)$, and the jointly normalized mean-shift field (Cheng, 1995)

$$V_{p,q}(x) = \frac{1}{Z_p Z_q} \mathbb{E}_{p,q}\left[k(x, y^+)\, k(x, y^-)\, (y^+ - y^-)\right], \tag{3}$$

which is anti-symmetric, $V_{p,q} = -V_{q,p}$, and vanishes when $p = q$. Training replaces the update Eq. (1) by the frozen-target regression

$$\mathcal{L}(\theta) = \mathbb{E}_\epsilon \left\| \phi(f_\theta(\epsilon)) - \mathrm{sg}\big(\phi(f_\theta(\epsilon)) + V\big) \right\|_2^2, \tag{4}$$

where ϕ stacks the weighted features of a pretrained latent MAE (He et al., 2022) and the stop-gradient prevents differentiation through the field (Chen & He, 2021). The ImageNet implementation uses temperatures $\tau \in \{0.02, 0.05, 0.2\}$, normalizes each temperature's force to unit RMS before summing, and trains with AdamW (Loshchilov & Hutter, 2019).

Equation (3) is an expectation. The implementation draws a finite positive support S of N examples from a per-class memory bank, together with a negative support, and forms $\widehat{V}_S$, which is frozen into the target of Eq. (4). We call constructing $\widehat{V}_S$ a *refresh* and optimizing against the resulting frozen target *realization*. The original algorithm takes exactly one optimizer step per refresh; we write k for the number of steps taken per refresh.

3 Deeper Optimization Does Realize the Frozen Field

3.1 Why one step need not execute the prescribed motion

Fix a latent batch and stack the generator features that Drifting regresses into $\mathbf{H}(\boldsymbol{\theta})$. The loss pulls $\mathbf{H}$ toward the frozen target $\mathbf{H} + \mathbf{V}$, so its gradient in parameter space is $-\mathbf{J}^\top \mathbf{V}$. Mapping the resulting parameter change back to feature space gives the following.

Proposition 1 (One-step realization)**.** *One gradient step with step size η moves the features by $\Delta\mathbf{H}_1 = \eta \mathbf{K}\mathbf{V} + O(\|\Delta\boldsymbol{\theta}\|^2)$, where $\mathbf{K} = \mathbf{J}\mathbf{J}^\top$.*

Written per sample, $\Delta\mathbf{H}_{1,i} = \eta \sum_j \mathbf{K}_{ij}\mathbf{V}_j$: sample i moves by a mixture of the motions prescribed for every sample in the batch, weighted by how strongly the shared parameters couple their outputs.

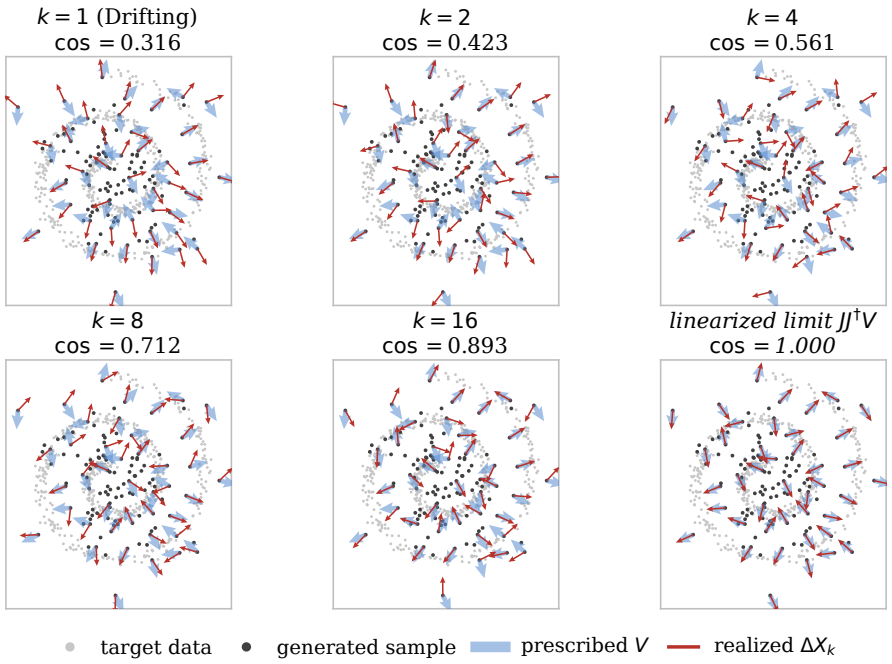


Figure 2: **Repeated fitting turns the realized motion toward the prescribed one.** Swiss-roll Drifting at one frozen training state. Blue: prescribed motion V, drawn wide and underneath; red: motion realized after k AdamW steps toward the same frozen target. Arrows are normalized for display; cosines use the full displacements of all 120 generated samples, and every panel shows the same 42 samples. The final panel is not an optimizer run but the $k \to \infty$ limit $JJ^\dagger V$ of the fixed-kernel model at the same state, where K is full rank and the prescribed motion is realized exactly.

Along the eigendirections of $\mathbf{K}$, components of $\mathbf{V}$ that the network can move easily are amplified and the rest are suppressed, so one step generally changes both the size and the direction of the prescribed motion.

If the target is held fixed and $\mathbf{K}$ is treated as constant, repeated steps correct part of what earlier steps missed.

Proposition 2 (Fixed-kernel realization model)**.** *Under a fixed* $\mathbf{K}$ *and* $0 < \eta < 2/\lambda_{\max}$*, k gradient steps against the frozen target move the features by*

$$\Delta\mathbf{H}_k = \left[I - (I - \eta\mathbf{K})^k\right]\mathbf{V}, \tag{5}$$

which converges to the part of $\mathbf{V}$ *the network can realize,* $\mathbf{J}\mathbf{J}^\dagger\mathbf{V}$.

This is classical Landweber (Richardson) iteration (Landweber, 1951); proofs and spectral identities are in Section A. We use it only to motivate the question of whether repeated fitting realizes the target better. Our ImageNet runs use AdamW with a time-varying preconditioner and a nonlinear network, so the answer below is measured directly from optimizer-induced displacements rather than predicted from $\mathbf{K}$.

Figure 2 shows what repeated fitting looks like in a two-dimensional Swiss-roll version of Drifting, where the whole state can be drawn. At one training state we freeze the target and let AdamW take k steps toward it. The realized motion turns toward the prescribed one as k grows, and its cosine with V rises from 0.32 at $k = 1$ to 0.89 at $k = 16$. The last panel is the $k \to \infty$ limit of the fixed-kernel model, $\mathbf{J}\mathbf{J}^\dagger\mathbf{V}$. At this state $\mathbf{K}$ has full rank (240 of 240 modes), so the linearized model realizes V exactly, and the finite-step shortfall comes from iteration rather than from a part of V the network cannot reach. The wide tangent spectrum (condition number $\approx 1.9 \times 10^3$) fits this picture, with strong directions realized within a few steps and weak ones needing many more, although the spectrum alone

does not predict the AdamW trajectory. A checkerboard target shows the same finite-k trend under both gradient descent and AdamW (Table 3). These two-dimensional problems make realization visible; every downstream conclusion below is measured directly on ImageNet under AdamW.

3.2 A REALIZATION DIAGNOSTIC

Let $\mathbf{V}$ be the prescribed field in the scaled coordinate of the loss, with the training block weighting, and $\Delta\mathbf{H}_k$ the displacement actually produced by k optimizer steps against the frozen target. Define the executed fraction, relative move and orthogonal component

$$a_k = \frac{\langle \Delta\mathbf{H}_k, \mathbf{V}\rangle}{\|\mathbf{V}\|^2}, \qquad m_k = \frac{\|\Delta\mathbf{H}_k\|}{\|\mathbf{V}\|}, \qquad o_k^2 = m_k^2 - a_k^2. \tag{6}$$

Because the step-0 residual is exactly $-\mathbf{V}$, the residual ratio $\rho_k = \|\Delta\mathbf{H}_k - \mathbf{V}\| \,/\, \|\mathbf{V}\|$ satisfies

$$\rho_k^2 = (1-a_k)^2 + o_k^2 \qquad \Longleftrightarrow \qquad 1 - \rho_k^2 = \underbrace{(2a_k - a_k^2)}_{\text{aligned gain } G_{\parallel,k}} - \underbrace{o_k^2}_{\text{orthogonal penalty } P_{\perp,k}}. \tag{7}$$

The identity holds for any displacement. Its value is as a measurement: it separates motion that executes the prescribed field from motion that does not, and $\rho_k < 1$ exactly when the aligned gain exceeds the orthogonal penalty. On all measured ImageNet rows it reconstructs the recorded ρ_k to within 10^{-9}.

3.3 MEASUREMENT

At the checkpoint from which every downstream run in Section 4 starts, net progress is positive at $k = 1$, dips slightly at $k = 2$, and then grows with depth: from $k = 1$ to $k = 32$ the executed fraction rises 24-fold while the displacement grows only 4-fold, and net progress reaches 3.7×10^{-2} (Fig. 1b; exact values in Section E). Deeper optimization genuinely improves realization of the frozen field.

Table 1 shows that this is a property of the training state. Early in training the optimizer moves far relative to the prescribed field and mostly orthogonally to it, and net progress stays negative until $k \approx 16$. As training progresses, the one-step displacement becomes smaller relative to the prescribed field but better aligned with it ($m_1 : 0.115 \to 0.041$, $\cos(\Delta\mathbf{H}_1, \mathbf{V}) : 0.024 \to 0.033$ from checkpoint 5k to 30k), and the positive-realization threshold moves from roughly $k = 16$ to $k = 1$. Realization also depends on batch geometry: at checkpoint 5k, increasing the effective batch from 4 to 64 moves the first depth with net progress from $k = 1$ to $k = 16$ (Section E).

Table 1: **The depth at which optimization begins to realize the frozen field falls from $k \approx 16$ to $k = 1$ during training.** Net progress $1 - \rho_k^2$ ($\times 10^{-3}$), $B_{\text{eff}} = 64$. Shaded: first depth with positive net progress.

checkpoint	k=1	k=2	k=4	k=8	k=16	k=32
5k	−7.7	−4.7	−7.1	−2.7	+11.0	+31.9
10k	−1.2	−4.7	−4.6	+3.5	+19.5	+37.5
20k	+0.6	−1.2	+3.3	+10.8	+23.3	+41.2
30k	+1.0	−0.6	+2.5	+9.2	+22.5	+36.9

4 REFRESH OR REALIZE? THE REVERSAL

Section 3 shows that extra optimization against a frozen target realizes it better. Training poses a different question: under a fixed budget, is that optimization better spent realizing the current field more accurately or obtaining a fresh one? We compare $k = 1$ and $k = 2$ optimizer steps per refresh for latent Drifting on ImageNet 256×256 at high positive support, with EMA weights, CFG 2.0 and 50k-sample FID, on two independently trained seeds.

At an equal number of refreshes, $k = 2$ receives twice the optimizer updates and reaches lower FID on both seeds (Table 2, top). Deeper realization is therefore not a worse procedure, and the comparison rules out the explanation that $k = 2$ simply optimizes badly.

Table 2: **Extra realization helps when it is free; refreshing wins once it is paid for.** FID under two budget definitions (lower is better; bold marks the better arm). Wall-clock is the scheduler-reported elapsed time of each run.

budget	seed	$k = 1$	$k = 2$	better
equal refresh count	42	7.56	**7.26**	$k = 2$
k=2 gets 2× updates	43	7.60	**7.30**	$k = 2$
≈ matched measured	42	**6.98** (32.2 h)	7.26 (35.7 h)	$k = 1$
wall-clock	43	**6.93** (34.5 h)	7.30 (33.9 h)	$k = 1$

Charging for the extra updates changes the outcome. At approximately matched measured wall-clock the $k = 1$ arm runs 45,252 refreshes and optimizer updates, against 30,000 refreshes and 60,000 updates for $k = 2$, and it reaches lower FID on both seeds (Table 2, bottom):

> *Under approximately matched measured wall-clock, refreshing the field outperforms deeper realization of the stale field in both training seeds.*

The two arms are matched only approximately, and the residual mismatch runs in opposite directions: on seed 42 the $k = 1$ run used 9.8% less time, on seed 43 1.9% more, while the FID ordering is the same in both. Run lengths were fixed from an estimated per-step cost before training. The same allocation decision, isolated at frozen checkpoints with matched compute and no scheduling effects, favors refreshing in all 40 probes we ran; Section 5 presents those probes together with what they reveal about the target.

The operational consequence for this ImageNet regime is direct: when one and two optimizer steps per refresh are compared at approximately matched measured wall-clock, refreshing more often is the better allocation than a second step toward the same stale target. This is not an ordering over arbitrary depths, and we do not claim it for every field construction.

5 Why Refreshing Pays: Support Dominates Local Target Change

5.1 Support refresh rotates the target, and the local decision favors it

Between consecutive steps the parameters move and the support is redrawn. We separate the two at frozen checkpoints. From a state θ_0 and target T_0 we take one update to θ_1, then rebuild the target at θ_1 twice: once from the same support (T_1^{same}) and once from a disjoint, freshly drawn support (T_1^{new}), with labels, generator noise and guidance draws held fixed. The same probe then makes the allocation decision with matched compute: REALIZE takes a second update toward the stale T_0, REFRESH takes it toward T_1^{new}, and both are scored on two held-out supports that neither branch used. We ran eight probes in each of five cells, covering support sizes $N \in \{64, 128, 256\}$ at checkpoint 30k and training stages 10k, 20k and 30k at $N = 256$ (Section F).

Figure 3a shows the first result. Rebuilding the target from the same support after one update leaves its direction nearly unchanged (cosine 0.94–0.98, mean 0.97); drawing a new support rotates it substantially (cosine 0.31–0.59, mean 0.48), and more so the smaller the support. In the measured regime, resampling the finite support changes the target direction far more than the local parameter update itself. In these probes staleness is dominated by support resampling: a frozen target encodes one draw of a quantity that varies strongly from draw to draw, while one parameter update barely moves it.

Figure 3b shows the decision. Refreshing gives the lower held-out loss in all 40 probes, by 0.30% to 1.49% (median 0.56%). The probes share model states and support pools within a cell, so we read this as a consistent effect across varied conditions rather than as 40 independent trials. The two panels also separate two things that are easy to conflate. At checkpoint 30k, the target rotates far more for $N = 64$ than for $N = 256$, yet the mean refresh gain is almost identical (0.47% and 0.48%); across training stages at $N = 256$ the rotation barely changes while the gain falls from 1.32% at 10k to 0.48% at 30k. Support sensitivity explains why refreshing changes the target; its magnitude is not a calibrated predictor of how much a refresh is worth.

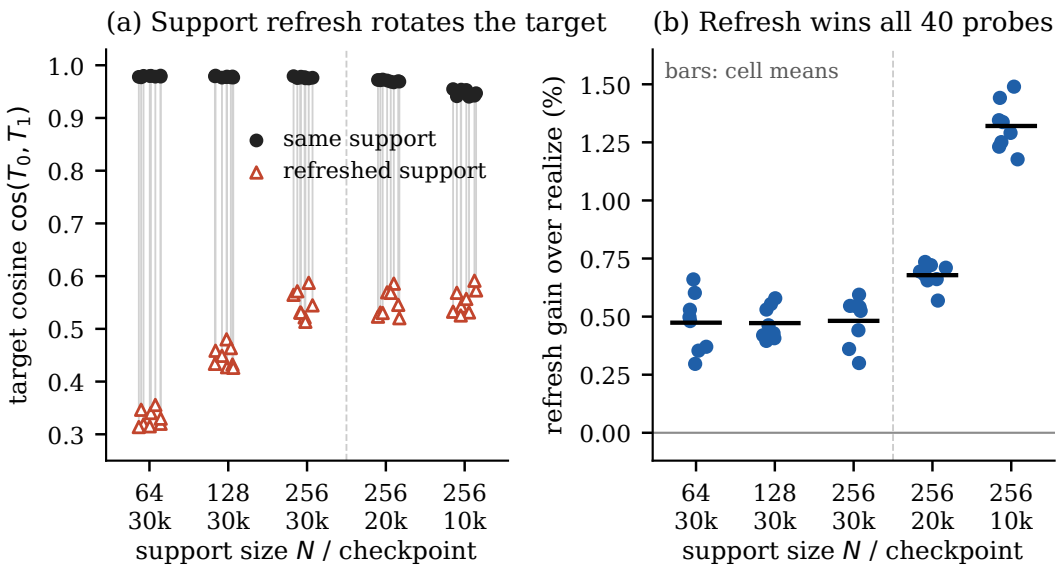


Figure 3: **Refreshing the support moves the target far more than a parameter update does, and refreshing wins every local allocation decision.** Forty probes, eight per position. The first three positions vary the support size at checkpoint 30k; the last three vary the training stage at $N = 256$, sharing the $N = 256$, 30k condition. **(a)** Cosine between the old target and the target rebuilt after one update, from the same support (dots) or a freshly drawn one (triangles); lines pair the two for each probe. **(b)** Held-out loss reduction of REFRESH over REALIZE at matched compute; every probe is positive. Probes within a cell share a model state and are not independent.

5.2 Field-space optimality does not reliably transfer to FID

If the support dominates the target, a natural response is to spend part of each refresh improving the estimate. We test the cheapest version that keeps the data budget fixed: split the same N positives into disjoint halves, form $V_{\text{split}} = \frac{1}{2}(V_A + V_B)$ and interpolate $V_\beta = V_N + \beta(V_{\text{split}} - V_N)$, so that $\beta = 0$ is Drifting and $\beta = 1$ is a parameter-free split average. A held-out audit against a disjoint large-support reference selects β^* without training or FID. In this ImageNet audit β^* follows an approximately $N^{-1/2}$ law, with fitted exponent -0.50 ($R^2 = 0.996$) at checkpoint 30k and -0.47 at 60k (Fig. 4a).

We froze β^* before observing any FID and ran deterministic paired continuations on both seeds. Averaged over three evaluation seeds, the calibrated correction lowers FID in all six seed–support cells, by 0.040 on average, but only 13 of 18 individual paired evaluations are favorable, so the effect is comparable to evaluation noise. More to the point, the audit-optimal coefficient does not reliably outperform the parameter-free split average (Fig. 4b): a correction that is optimal in field space is not reliably optimal for downstream FID. We had also preregistered that the benefit would shrink as N grows. The single-evaluation-seed results falsified this; averaging over evaluation seeds restores the predicted ordering, which we report as exploratory.

5.3 A nominal affinity floor materially changes the production field

The production field symmetrizes each affinity as $\sqrt{\max(p_{ij}^{\text{row}} p_{ij}^{\text{col}},\, 10^{-6})}$ before forming the attraction and repulsion coefficients. The floor looks like a numerical safeguard, but at checkpoint 30k it binds on about half of the $\tau = 0.05$ affinities and on 91% of the $\tau = 0.02$ affinities. We recomputed the field with and without it, holding the checkpoint, labels, generator noise, guidance draws, generated features, supports and block weights identical, with a correctness gate against the compiled production code and a materiality threshold fixed before the run (Section G). Removing only the floor changes the aggregated production field by 12.3% relative L_2 (cosine 0.992); the $\tau = 0.05$ and $\tau = 0.02$ branches change by 10.8% and 21.3%. Because each temperature branch is RMS-normalized, every branch keeps its norm, so the floor acts on the direction of the field rather

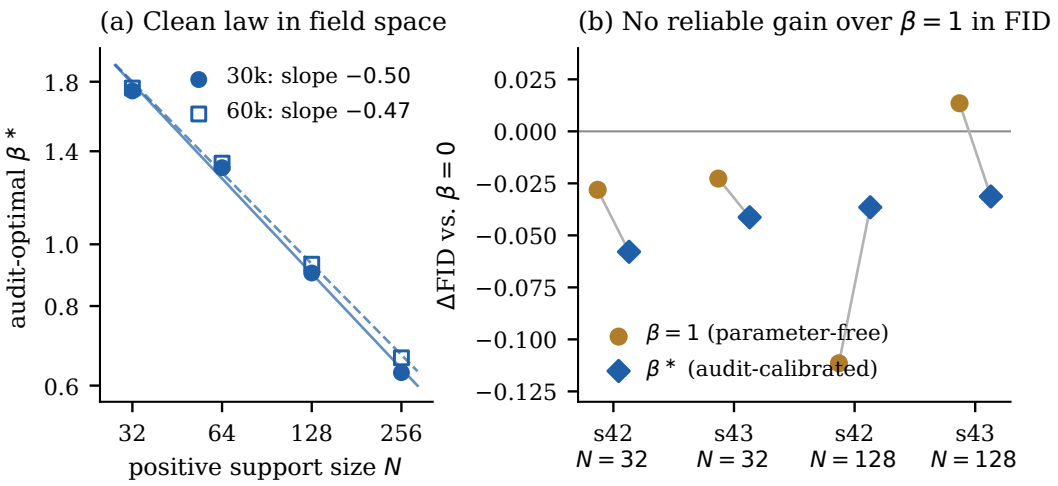


Figure 4: **A clean law in field space does not become a reliable gain in FID. (a)** The audit-optimal coefficient follows an approximately $N^{-1/2}$ law, stable between checkpoints 30k and 60k. **(b)** Downstream, the calibrated coefficient is not reliably better than the parameter-free $\beta = 1$: it is better in three cells and worse in one, and the mean difference is -0.005 FID. FID averaged over three evaluation seeds; exact values in Section H.

than its scale. The field Drifting optimizes in this regime includes the floor, and our conclusions describe that production field.

6 Related Work

Drifting (Deng et al., 2026) arrived with a strong empirical result and comparatively little theory, and most follow-up work has been about the field itself. It has been read as a long–short flow map (Li & Zhu, 2026), as a Wasserstein gradient flow of a kernel-smoothed divergence (Cao et al., 2026; Gretton et al., 2026), and as score matching between kernel-smoothed distributions (Turan & Ovsjanikov, 2026; Lai et al., 2026); other work changes the field, replacing its displacement direction by kernel gradients (Esteban-Casadevall et al., 2026) or adding momentum to counter spectral bias (Brown et al., 2026). These papers analyze the prescribed dynamics. Our object is the training loop that implements them: how much of the prescribed motion a shared network carries out, and how compute should be split between rebuilding the field and fitting it. A better-founded field still has to be estimated from a finite support and realized by an optimizer, so the two lines of work are complementary.

Drifting is one of several routes to one- or few-step generation, alongside flow matching and rectified flow (Lipman et al., 2023; Liu et al., 2023), consistency models (Song et al., 2023) and shortcut models (Frans et al., 2025), usually on transformer backbones (Peebles & Xie, 2023). What sets it apart for our purposes is that its regression target is rebuilt from data at every step, which is why a refresh–realize trade-off exists at all.

The mechanism of Proposition 1, a parameter update acting on outputs through $\mathbf{K} = \mathbf{J}\mathbf{J}^\top$, is the starting point of the neural tangent kernel literature (Jacot et al., 2018; Lee et al., 2019; Adlam & Pennington, 2020; Novak et al., 2022), which has also been used to study adversarial generative training (Franceschi et al., 2022); the frozen-target iteration of Proposition 2 is Landweber iteration (Landweber, 1951; Engl et al., 1996). We use this machinery to pose the realization question rather than to answer it, since our measurements are taken under AdamW, where the fixed-kernel model holds only approximately.

The refresh–realize question also connects to two older ideas. Deep reinforcement learning deliberately holds regression targets fixed, refreshing target networks only periodically for stability (Mnih et al., 2015); Drifting sits at the other end, because its target is a finite-sample estimate and a stale target mostly encodes the particular support it was built from. And, as with minibatch estimates of kernel discrepancies such as MMD (Gretton et al., 2012), the empirical Drifting target depends on

the composition of its sample; we measure that dependence at the level of the direction a training step is asked to fit.

7 Scope and Conclusion

Scope. The ImageNet evidence comes from two independently trained source seeds of latent Drifting at 256×256, reported individually. The reversal rests on agreement across evidence rather than on seed count: both seeds order the arms the same way, the residual wall-clock mismatch changes sign between them, and the local probes agree. The long-run comparison tests one against two optimizer steps per refresh at approximately matched measured wall-clock; it does not establish an ordering over arbitrary depths or under exact compute matching. The support-size law for β^* is fitted against a finite large-support reference and describes that audit rather than a population field. The comparisons are made on the production field, affinity floor included, and whether other affinity constructions shift the balance between refreshing and realizing remains open.

Conclusion. Drifting prescribes a motion for every generated sample and leaves its execution to a shared network optimizer. We measured the gap between the two and what it implies for how training compute should be spent. Fitting a frozen target for longer does move the network closer to it, more so as training progresses, and with free extra steps it improves FID. Under approximately matched measured wall-clock the budget is nonetheless better spent recomputing the field, on both training seeds and in every local probe. Much of the explanation sits in the target: at fixed parameters, a new finite support changes its direction far more than an optimizer step does, and calibrating that target in field space does not carry over reliably to FID. Refresh frequency, realization depth and support construction are therefore design choices of the Drifting training loop in their own right, and the realization diagnostic introduced here measures the first two directly.

Reproducibility Statement

Sections 3 to 5 specify every measurement and comparison. Section D records the configuration, the controls used to compare realization depths without confounders, both budget definitions, and the deterministic paired protocol behind every field-correction comparison. Sections E, G and H give the exact values behind every rounded number in the main text, and Section G documents the production-equivalence gate of the affinity-floor diagnostic. Anonymized code and configuration files accompany the submission.

AI Use Statement

In this work, we used generative AI tools (large language models) for the following tasks with required disclosure: helping develop the conceptual framework (the refresh/realize framing and the realization decomposition), formulating mathematical claims and assisting in writing proofs, proposing and refining hypotheses, designing and giving feedback on the experimental methodology (including controls, paired protocols and diagnostics), implementing methods (training-loop modifications and diagnostic code), and interpreting results. We have not used generative AI tools to generate synthetic datasets, and translation, dataset cleaning and reformatting, and qualitative or thematic data analysis are not applicable to this work. Additionally, we used generative AI tools for tasks with recommended disclosure: creating and editing software code, suggesting experimental parameters, creating and modifying scientific figures, brainstorming, identifying and summarizing related literature, formatting references, suggesting the structure and title of the paper, and drafting and editing the manuscript. We have reviewed all AI-assisted work. All experiments were run by the authors on their own compute and every reported number comes from those runs; AI-assisted code was checked against the production implementation with numerical equivalence gates and smoke tests before use. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Ben Adlam and Jeffrey Pennington. The neural tangent kernel in high dimensions: Triple descent and a multi-scale theory of generalization. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 74–84, 2020.

Drake Brown, Yuhao Huang, Shih-Hsin Wang, and Bao Wang. Second order drifting models. *arXiv preprint arXiv:2608.07924*, 2026.

Jiarui Cao, Zixuan Wei, and Yuxin Liu. Gradient flow drifting: Generative modeling via Wasserstein gradient flows of KDE-approximated divergences. *arXiv preprint arXiv:2603.10592*, 2026.

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *2021 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 15745–15753. IEEE, 2021.

Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE transactions on pattern analysis and machine intelligence*, 17(8):790–799, 1995.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Mingyang Deng, He Li, Tianhong Li, Yilun Du, and Kaiming He. Generative modeling via drifting. *arXiv preprint arXiv:2602.04770*, 2026.

Heinz W. Engl, Martin Hanke, and Andreas Neubauer. *Regularization of Inverse Problems*. Kluwer Academic Publishers, 1996.

Maria Esteban-Casadevall, Jorge Carrasco-Pollo, Max Welling, Jan-Willem van de Meent, Erik J. Bekkers, and Floor Eijkelboom. Kernel-gradient drifting models. *arXiv preprint arXiv:2605.10727*, 2026.

Jean-Yves Franceschi, Emmanuel De Bézenac, Ibrahim Ayed, Mickael Chen, Sylvain Lamprier, and Patrick Gallinari. A neural tangent kernel perspective of gans. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 6660–6704, 2022.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. In *International Conference on Learning Representations*, 2025.

Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13:723–773, 2012.

Arthur Gretton, Li Kevin Wenliang, Alexandre Galashov, James Thornton, Valentin De Bortoli, and Arnaud Doucet. On the wasserstein gradient flow interpretation of drifting models. *arXiv preprint arXiv:2605.05118*, 2026.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 15979–15988. IEEE, 2022.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems*, 2018.

Chieh-Hsin Lai, Bac Nguyen, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yuki Mitsufuji, Stefano Ermon, and Molei Tao. A unified view of drifting and score-based models. *arXiv preprint arXiv:2603.07514*, 2026.

Louis Landweber. An iteration formula for fredholm integral equations of the first kind. *American Journal of Mathematics*, 73(3):615–624, 1951.

Jaehoon Lee, Lechao Xiao, Samuel S. Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In *Advances in Neural Information Processing Systems*, 2019.

Zhiqi Li and Bo Zhu. A long-short flow-map perspective for drifting models. *arXiv preprint arXiv:2602.20463*, 2026.

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.

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.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Roman Novak, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. Fast finite width neural tangent kernel. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 17018–17044, 2022.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning*, 2023.

Erkan Turan and Maks Ovsjanikov. Generative drifting is secretly score matching: A spectral and variational perspective. *arXiv preprint arXiv:2603.09936*, 2026.

A Complete Mathematical Derivations

A.1 Notation and stacking of multiple feature losses

The original image objective sums squared losses over feature scales/locations j. Suppose a batch contains generated samples indexed by $i = 1, \dots, m$, and each feature block has weight $w_j > 0$ after all implementation normalizations. Define

$$h_\theta(\epsilon_i) = \begin{bmatrix} \sqrt{w_1}\,\phi_1(f_\theta(\epsilon_i)) \\ \vdots \\ \sqrt{w_J}\,\phi_J(f_\theta(\epsilon_i)) \end{bmatrix}, \qquad v_i = \begin{bmatrix} \sqrt{w_1}\,V_{i,1} \\ \vdots \\ \sqrt{w_J}\,V_{i,J} \end{bmatrix}. \tag{8}$$

Then the sum of feature losses is exactly a single Euclidean regression

$$\ell(\theta) = \frac{1}{2}\left\|\mathbf{H}(\theta) - \mathrm{sg}(\mathbf{H}_0 + \mathbf{V})\right\|^2, \tag{9}$$

up to an overall scalar normalization. Any mean reduction over samples/features multiplies the gradient by a constant; throughout the theory this constant is absorbed into the effective step η. Therefore the results apply to the multi-feature ImageNet objective without requiring each feature block to have equal dimension or scale.

A.2 One-step KV identity with a Taylor remainder

Lemma 3 (Second-order remainder)**.** *Let $H : \mathbb{R}^P \to \mathbb{R}^n$ be continuously differentiable on a convex neighborhood of θ_0, and suppose its Jacobian is L_J-Lipschitz in operator norm:*

$$\|J(\theta) - J(\theta')\|_{\mathrm{op}} \le L_J\|\theta - \theta'\|. \tag{10}$$

Then for any displacement δ contained in the neighborhood,

$$H(\theta_0 + \delta) = H(\theta_0) + J_0\delta + R(\delta), \qquad \|R(\delta)\| \le \frac{L_J}{2}\|\delta\|^2. \tag{11}$$

Proof. By the fundamental theorem of calculus,

$$H(\theta_0 + \delta) - H(\theta_0) = \int_0^1 J(\theta_0 + s\delta)\,\delta\,ds \tag{12}$$

$$= J_0\delta + \int_0^1 [J(\theta_0 + s\delta) - J_0]\,\delta\,ds. \tag{13}$$

The second term is $R(\delta)$. Using the Lipschitz assumption,

$$\|R(\delta)\| \le \int_0^1 L_J\, s\|\delta\|^2 ds = \frac{L_J}{2}\|\delta\|^2. \tag{14}$$

□

Proof of Proposition 1. The frozen target is $T = H_0 + V$ and the loss is

$$\ell(\theta) = \frac{1}{2}\|H(\theta) - T\|^2. \tag{15}$$

At θ_0, $H(\theta_0) - T = -V$, so by the chain rule

$$\nabla_\theta \ell(\theta_0) = J_0^\top(H_0 - T) = -J_0^\top V. \tag{16}$$

A gradient-descent step gives

$$\delta_1 := \theta_1 - \theta_0 = -\eta\nabla\ell(\theta_0) = \eta J_0^\top V. \tag{17}$$

Applying Lemma 3,

$$H(\theta_1) - H_0 = J_0\delta_1 + R(\delta_1) \tag{18}$$

$$= \eta J_0 J_0^\top V + R(\delta_1) = \eta KV + R(\delta_1), \tag{19}$$

with

$$\|R(\delta_1)\| \le \frac{L_J}{2}\eta^2\|J_0^\top V\|^2. \tag{20}$$

□

A.3 Proof and interpretation of tangent-spectral distortion

Let $K = U\Lambda U^\top$ with orthonormal eigenvectors u_r and nonnegative eigenvalues λ_r. Write $V = \sum_r a_r u_r$. Then

$$KV = \sum_r \lambda_r a_r u_r, \tag{21}$$

which gives the band-energy identity

$$\langle \mathbf{V}, \mathbf{KV} \rangle = \sum_r \lambda_r a_r^2. \tag{22}$$

For the cosine,

$$\langle V, KV \rangle = \sum_r \lambda_r a_r^2, \tag{23}$$

$$\|V\|^2 = \sum_r a_r^2, \tag{24}$$

$$\|KV\|^2 = \sum_r \lambda_r^2 a_r^2. \tag{25}$$

Hence

$$\cos(\mathbf{V}, \mathbf{KV}) = \frac{\sum_r \lambda_r a_r^2}{\sqrt{\sum_r a_r^2}\sqrt{\sum_r \lambda_r^2 a_r^2}}. \tag{26}$$

Cauchy–Schwarz applied to the sequences $\{|a_r|\}$ and $\{\lambda_r|a_r|\}$ yields $\langle V, KV\rangle \le \|V\|\|KV\|$. Equality holds iff the two sequences are linearly dependent on the support of a_r, i.e., iff there exists c such that $\lambda_r|a_r| = c|a_r|$ for every active r. Thus all eigendirections carrying nonzero field coefficients must share the same eigenvalue.

A useful pairwise view makes the distortion even more explicit. For two active modes i, j,

$$\frac{|\langle u_i, KV\rangle|/|\langle u_j, KV\rangle|}{|\langle u_i, V\rangle|/|\langle u_j, V\rangle|} = \frac{\lambda_i}{\lambda_j}. \tag{27}$$

Thus an ill-conditioned tangent spectrum deterministically magnifies the relative prominence of top modes. This statement concerns tangent eigenmodes. It should not be interpreted as a claim about spatial/Fourier image frequencies.

For the finite-k realization, replace λ_r by $g_k(\lambda_r) = 1 - (1 - \eta\lambda_r)^k$. Therefore

$$J\delta_k = \sum_r g_k(\lambda_r) a_r u_r. \tag{28}$$

For every $\lambda_r > 0$, $g_k(\lambda_r) \to 1$ as $k \to \infty$ under the stability condition. Hence the relative reweighting between any two positive-eigenvalue modes disappears asymptotically:

$$\frac{g_k(\lambda_i)}{g_k(\lambda_j)} \to 1. \tag{29}$$

This is the precise sense in which repeated frozen-target fitting flattens the one-step tangent filter.

A.4 Exact linearized k-step dynamics

Freeze $J = J_0$ and V and optimize the local least-squares model

$$F(\delta) = \frac{1}{2}\|J\delta - V\|^2, \qquad \nabla F(\delta) = J^\top(J\delta - V). \tag{30}$$

Starting from $\delta_0 = 0$, gradient descent is

$$\delta_{s+1} = \delta_s + \eta J^\top(V - J\delta_s). \tag{31}$$

Define output residual $r_s = V - J\delta_s$. Then

$$r_{s+1} = V - J\delta_{s+1} \tag{32}$$

$$= V - J\delta_s - \eta JJ^\top(V - J\delta_s) \tag{33}$$

$$= (I - \eta K) r_s. \tag{34}$$

Since $r_0 = V$, induction yields

$$r_k = (I - \eta K)^k V. \tag{35}$$

Because $J\delta_k = V - r_k$,

$$J\delta_k = \left[I - (I - \eta K)^k\right] V, \tag{36}$$

proving Proposition 2.

An explicit expression for the parameter iterate follows by summing Eq. (31):

$$\delta_k = \eta \sum_{s=0}^{k-1} J^\top r_s \tag{37}$$

$$= \eta J^\top \sum_{s=0}^{k-1} (I - \eta K)^s V. \tag{38}$$

This expression is valid even if K is singular; no inverse has been introduced.

A.5 SVD PROOF OF THE MOORE–PENROSE LIMIT

Let the compact singular value decomposition be

$$J = U_r \Sigma_r W_r^\top, \qquad \Sigma_r = \mathrm{diag}(\sigma_1, \ldots, \sigma_r), \tag{39}$$

where $\sigma_i > 0$ and $r = \mathrm{rank}(J)$. Extend U_r to an orthonormal basis and decompose

$$V = \sum_{i=1}^{r} a_i u_i + V_\perp, \qquad V_\perp \in \mathrm{Null}(J^\top). \tag{40}$$

Because $K = U_r \Sigma_r^2 U_r^\top$, the finite-$k$ filter

$$g_k(\lambda) = 1 - (1 - \eta\lambda)^k \tag{41}$$

gives

$$J\delta_k = \sum_{i=1}^{r} \left[1 - (1 - \eta\sigma_i^2)^k\right] a_i u_i. \tag{42}$$

The null component $V_\perp$ is never realized.

For the parameter iterate, use $J^\top u_i = \sigma_i w_i$ in Eq. (38):

$$\delta_k = \sum_{i=1}^{r} \eta\sigma_i a_i \sum_{s=0}^{k-1} (1 - \eta\sigma_i^2)^s w_i \tag{43}$$

$$= \sum_{i=1}^{r} \frac{a_i}{\sigma_i} \left[1 - (1 - \eta\sigma_i^2)^k\right] w_i. \tag{44}$$

If $0 < \eta < 2/\lambda_{\max}(K) = 2/\sigma_1^2$, then $|1 - \eta\sigma_i^2| < 1$ for every nonzero singular direction. Hence

$$\delta_k \to \sum_{i=1}^{r} \frac{a_i}{\sigma_i} w_i = W_r \Sigma_r^{-1} U_r^\top V = J^\dagger V. \tag{45}$$

Similarly, Eq. (42) converges to

$$\sum_{i=1}^{r} a_i u_i = U_r U_r^\top V = J J^\dagger V. \tag{46}$$

Because every iterate δ_k lies in $\mathrm{Range}(J^\top)$, no component in $\mathrm{Null}(J)$ is introduced. The limit is therefore the minimum-Euclidean-norm least-squares solution.

A.6 Fixed-point interpretation and implicit solution of $K^{-1}V$

There is an equivalent output-space interpretation. Since $\delta_0 = 0$, every iterate can be written $\delta_k = J^\top u_k$ for some u_k. Substituting into Eq. (31),

$$J^\top u_{k+1} = J^\top \left[u_k + \eta(V - Ku_k)\right]. \tag{47}$$

One admissible recursion is

$$u_{k+1} = u_k + \eta(V - Ku_k), \qquad u_0 = 0. \tag{48}$$

This is Richardson/Landweber fixed-point iteration for $Ku = V$. If K is invertible, its fixed point is $u^* = K^{-1}V$, yielding $\delta^* = J^\top K^{-1}V = J^\dagger V$. If K is singular, the iteration converges on the positive-eigenvalue subspace and the induced parameter limit remains $J^\dagger V$.

This identity formalizes the central algorithmic observation: the network parameters implicitly carry $J^\top u_k$, so an explicit output-space inverse variable need not be stored.

A.7 Finite-step spectral regularization

For a K-eigenvector with eigenvalue λ, the output gain is $g_k(\lambda) = 1-(1-\eta\lambda)^k$. If $0 < \eta \le 1/\lambda_{\max}$, then $0 \le 1 - \eta\lambda \le 1$ and

$$\frac{\partial}{\partial k} g_k(\lambda) > 0, \qquad \frac{\partial}{\partial \lambda} g_k(\lambda) > 0 \tag{49}$$

in the continuous extension. Thus large-λ directions are realized first. For $\eta\lambda \ll 1$, the binomial expansion gives

$$g_k(\lambda) = k\eta\lambda + O(k^2\eta^2\lambda^2). \tag{50}$$

The parameter coefficient from Eq. (44) is

$$c_{k,i} = \frac{a_i}{\sigma_i}\left[1 - (1 - \eta\sigma_i^2)^k\right]. \tag{51}$$

For weak singular values, $\eta\sigma_i^2 \ll 1$,

$$c_{k,i} \approx k\eta\sigma_i a_i, \tag{52}$$

whereas the exact pseudoinverse coefficient is a_i/σ_i. Early stopping therefore avoids the $1/\sigma_i$ amplification of extremely weak directions. This is the precise sense in which finite k acts as iterative regularization.

For a desired directional realization error $|1 - g_k(\lambda)| \le \varepsilon$,

$$|1 - \eta\lambda|^k \le \varepsilon \quad \Longleftrightarrow \quad k \ge \frac{\log \varepsilon}{\log|1 - \eta\lambda|}. \tag{53}$$

When $\eta\lambda \ll 1$, $\log(1 - \eta\lambda) \approx -\eta\lambda$, giving

$$k \gtrsim \frac{\log(1/\varepsilon)}{\eta\lambda}. \tag{54}$$

Small k is therefore a low-order iterative correction rather than an accurate pseudoinverse solve, which is consistent with the measured residuals in Table 4.

A.8 Nonlinear network: what remains exact

The linearized formulas are exact for the frozen-J model. For the real nonlinear network, the optimization objective

$$\min_\theta \frac{1}{2}\|H(\theta) - T\|^2 \tag{55}$$

is itself exact; only its interpretation as a fixed polynomial in K_0 is local. At micro-step s,

$$\theta_{s+1} = \theta_s - \eta J(\theta_s)^\top (H(\theta_s) - T). \tag{56}$$

If the Jacobian remains close to J_0 over the micro-trajectory, Lemma 3 controls the mismatch between the nonlinear feature displacement and $J_0\delta_s$. Accordingly, we use the local model to interpret realization rather than to predict it: every realization measurement reported in this paper uses the actual finite feature displacement produced by the optimizer, not a linearized prediction.

B Preconditioned Realization and AdamW

The public image system uses AdamW rather than Euclidean gradient descent. A useful intermediate model freezes a symmetric positive-definite parameter preconditioner $P \succ 0$ during one outer realization solve:

$$\delta_{s+1} = \delta_s + \eta P J^\top (V - J\delta_s). \tag{57}$$

Define

$$K_P = JPJ^\top. \tag{58}$$

Then the residual obeys

$$r_{s+1} = (I - \eta K_P) r_s, \tag{59}$$

so

$$J\delta_k = [I - (I - \eta K_P)^k] V. \tag{60}$$

The derivation is identical to that of Proposition 2. Let $A = JP^{1/2}$ and write $\delta = P^{1/2} w$. The fixed-point problem becomes $Aw \approx V$, whose minimum-Euclidean-norm solution is $w^* = A^\dagger V$. Therefore

$$\delta^* = P^{1/2} A^\dagger V = P J^\top (JPJ^\top)^\dagger V. \tag{61}$$

This is the solution minimizing $\delta^\top P^{-1} \delta$ among least-squares solutions. A fixed diagonal preconditioner can therefore flatten or reshape the relevant tangent spectrum and accelerate some directions.

Actual AdamW has time-varying first- and second-moment states and decoupled weight decay, so Eq. (57) is not an exact model of the full optimizer. We use it only to motivate two diagnostics: (i) compare $J(-g)$ with the actual $J\Delta\theta_{\mathrm{AdamW}}$ at the same state; and (ii) compare repeated AdamW to repeated SGD on the same frozen target. No theorem in the paper should claim that AdamW converges to the Euclidean Moore–Penrose solution.

C Controlled Two-Dimensional Study

The two-dimensional study makes realization and allocation visible. The target is a Swiss roll, the generator is an MLP acting on a fixed latent bank, and the drift field is built by the same attraction–repulsion construction as the ImageNet implementation (single temperature, generated samples as the repulsive support). At a fixed intermediate training state we freeze the target $T = X_0 + V$, clone the model, and take $k \in \{1, 2, 4, 8, 16\}$ micro-steps with either gradient descent or AdamW, measuring the cosine between the realized displacement $\Delta X_k = X_k - X_0$ and V and the residual $\|V - \Delta X_k\| / \|V\|$ (Table 3, Fig. 2). We then train from scratch for 10,000 outer iterations at each k and compare the sliced-Wasserstein distance to the target under four budget definitions (Fig. 5).

C.1 Long-horizon toy training under matched budgets

Figure 5 reports the full training trajectories behind Table 3. Each curve is the mean over three paired seeds with one standard deviation shaded; the vertical axis is logarithmic. For the optimizer-update, FLOP and wall-clock comparisons, curves are shown only over the budget range available to every realization depth.

D ImageNet Protocol

D.1 Base configuration and reproduction

All ImageNet results use the public JAX release of Deng et al. (2026) in its latent 256×256 configuration with a pretrained latent MAE feature space, temperatures $\{0.02, 0.05, 0.2\}$, AdamW, and EMA weights for evaluation. We reproduced the public ablation baseline before running any comparison in this paper; all comparisons branch from reproduced checkpoints rather than from released weights. FID uses 50k generated samples at CFG 2.0 with a fixed evaluation batch size, and every arm of a comparison uses identical evaluation arguments so that generation noise is common across arms.

Table 3: **Finite-k realization under GD and AdamW.** Cosine measures directional agreement between the prescribed field V and the actual finite output displacement ΔX_k; residual is $\|V - \Delta X_k\|/\|V\|$. The audit is performed at a fixed intermediate training state with the same frozen target for all k. The GD learning rate is chosen from the local stability condition, whereas AdamW uses the practical fixed learning rate; the table is therefore intended to compare the *trend with* k, not the absolute performance of the two optimizers.

Dataset	Optimizer	Metric	$k = 1$	$k = 2$	$k = 4$	$k = 8$	$k = 16$
Swiss roll	GD	cosine	0.342	0.448	0.516	0.585	0.651
		residual	0.953	0.926	0.889	0.841	0.785
	AdamW	cosine	0.316	0.423	0.561	0.712	**0.893**
		residual	0.949	0.909	0.834	0.703	**0.457**
Checkerboard	GD	cosine	0.370	0.469	0.553	0.621	0.694
		residual	0.960	0.933	0.892	0.836	0.765
	AdamW	cosine	0.277	0.449	0.615	0.775	**0.927**
		residual	0.961	0.894	0.792	0.637	**0.374**

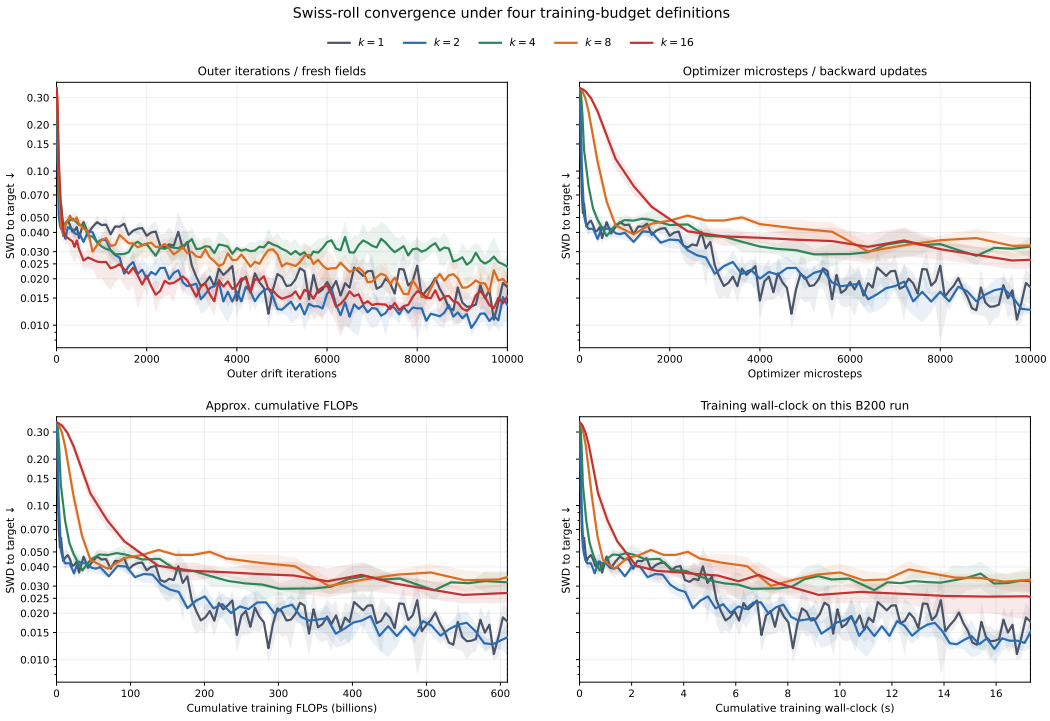


Figure 5: **How realization depth ranks depends on what is held fixed.** Swiss-roll sliced-Wasserstein distance under four budget definitions. Ranking the same runs by outer iterations, optimizer micro-steps, FLOPs and wall-clock gives different orderings, which is why the ImageNet comparison reports measured elapsed time for every arm.

D.2 Realization depth without hidden confounders

For $k > 1$ the frozen target must be recomputed from the parameters at the start of the outer iteration, never from the parameters as they move within it; our implementation rematerializes the target from the captured outer parameters for every micro-step after the first. Exponential moving average is updated once per outer iteration for every k, so arms compared at a fixed outer horizon receive the same number of EMA updates. Weight decay is applied per micro-step at the rate that makes the

per-outer decay identical across k, i.e. $(1-\eta\lambda_{\text{inner}})^k = 1-\eta\lambda_{\text{outer}}$. Without these three controls a k comparison silently changes target staleness, averaging horizon and regularization at the same time.

D.3 Resource-matched protocol

Two budget definitions appear in Section 4. *Equal outer iterations* fixes the number of refreshes and lets k determine the number of optimizer updates; this favors larger k, and we report it first. *Approximately matched measured wall-clock* fixes the scheduler elapsed time; run lengths were set from an estimated per-step cost before the runs and the realized elapsed times are reported as measured, not as targets. We report both, together with optimizer-update counts, because ranking by outer iterations, by micro-steps, by FLOPs and by wall-clock can disagree (Fig. 5).

D.4 Deterministic paired protocol for field-correction comparisons

Every β comparison in Section 5 is run as a matched pair: the two arms restore the same full training state, including optimizer moments and EMA, and are seeded per step before the data iterator and the memory-bank draw, so the two runs see identical batches, identical generator noise and identical support draws. The only difference between the arms is β. Each arm's positive support is drawn from a per-class bank whose size is twice the support size, and bank occupancy is asserted to exceed the support size at the first scientific update so that sampling never silently falls back to sampling with replacement.

E Exact Realization Measurements

All quantities below are computed from optimizer-induced displacements at frozen checkpoints in the scaled coordinate of the loss with the training block weighting. The identity $\rho_k^2 = (1-a_k)^2 + o_k^2$ reconstructs every recorded ρ_k to within 1.4×10^{-9}.

Table 4: Full decomposition at checkpoint 30k, $B_{\text{eff}} = 64$ (the data behind Fig. 1b).

k	ρ_k	cos	m_k	a_k	$G_{\parallel,k}$	$P_{\perp,k}$	$1-\rho_k^2$
1	0.999505	0.03266	0.04146	0.001354	0.002706	0.001717	+0.000989
2	1.000303	0.03446	0.07681	0.002647	0.005287	0.005893	−0.000606
4	0.998774	0.05935	0.09208	0.005465	0.010900	0.008449	+0.002451
8	0.995368	0.09702	0.11004	0.010676	0.021238	0.011995	+0.009243
16	0.988698	0.15148	0.12983	0.019666	0.038945	0.016468	+0.022477
32	0.981388	0.19359	0.16914	0.032744	0.064415	0.027537	+0.036878

Batch geometry. Realizability also depends on the effective batch. At checkpoint 5k, the one-step residual ratio is $\rho_1 = 0.99961, 0.99964, 1.00189, 1.00365$ and 1.00385 for $B_{\text{eff}} = 4, 8, 16, 32, 64$, and the first depth with $\rho_k < 1$ is $k = 1, 1, 4, 8, 16$ respectively. Larger batches spread a fixed parameter update over more generated samples, so a single step executes less of each sample's prescribed motion.

F Support Sensitivity and Local Allocation Probes

Support sensitivity. Both cosines in Section 5 are measured at frozen checkpoints with labels, generator noise and guidance draws held fixed. The same-support value takes one optimizer step and rebuilds the target from the identical support; the resampled value holds the parameters fixed and redraws only the support. Measuring both from the same probe ensures that the difference reflects the target, not the probe.

Local refresh/realize probes. Each probe starts from a checkpoint θ_0 and a target T_0 built from support S_0, takes one optimizer update to θ_1, and rebuilds the target at θ_1 from the same support (T_1^{same}) and from a disjoint freshly drawn support (T_1^{new}), with labels, generator noise and guidance draws fixed. The two cosines in Fig. 3a are $\cos(T_0, T_1^{\text{same}})$, which isolates the parameter-induced

change, and $\cos(T_0, T_1^{\text{new}})$, which adds support resampling. From the same θ_1, REALIZE takes a second update toward T_0 and REFRESH a second update toward T_1^{new}; both are evaluated on two held-out supports unseen by either branch, giving held-out losses L_R and L_F. Figure 3b plots $100\,(L_R - L_F)/L_R$, positive when refreshing is better. Each cell holds eight probes.

Table 5: Local refresh/realize probes, cell means over eight probes each.

cell	$\cos(T_0, T_1^{\text{same}})$	$\cos(T_0, T_1^{\text{new}})$	refresh gain	refresh wins
$N = 64$, 30k	0.9790	0.3300	0.474%	8/8
$N = 128$, 30k	0.9780	0.4463	0.472%	8/8
$N = 256$, 30k	0.9767	0.5456	0.482%	8/8
$N = 256$, 20k	0.9704	0.5466	0.678%	8/8
$N = 256$, 10k	0.9476	0.5526	1.321%	8/8
all 40	0.9704	0.4842	0.685% (median 0.562%)	40/40

One probe ($N = 256$, 10k, probe 7) was initially rejected by the 10^{-4} production-equivalence gate on the target decomposition (relative error 1.13×10^{-4}) before any oracle quantity was computed. It was rerun at the 5×10^{-4} tolerance adopted from the three-way numerical audit described in Section G, with the checkpoint, support draws, updates, held-out evaluation and decision rule unchanged. The tolerance was chosen from that audit, not from any probe outcome.

G AFFINITY-FLOOR DIAGNOSTIC

Protocol. At checkpoint 30k with temperatures $(0.2, 0.05, 0.02)$, 64 positives, 16 negatives and micro-batch 4, we evaluate four micro-slices. For each slice the clip-on and clip-off fields share the checkpoint, parameters, labels, generator noise, guidance draws, generated features, positive and negative supports, feature blocks and block weights; the only change is $\sqrt{\max(p^{\text{row}}p^{\text{col}}, 10^{-6})}$ versus $\sqrt{p^{\text{row}}p^{\text{col}}}$. The field construction reproduces the production code: weighted distance scale including the unconditional-negative guidance weight, generated-sample self mask, row and column softmaxes, attraction and repulsion coefficients, and per-temperature RMS normalization. Metrics use the training block weighting and are pooled from summed inner products and norms across slices rather than by averaging per-slice cosines. The materiality criterion (aggregated cosine < 0.99 or relative $L_2 > 0.05$) was fixed before the run.

Correctness. Before any metric is reported, each block's clip-on target is compared with the compiled production `drift_target`. Our mirror and an independent established audit mirror agree to 2.9×10^{-6}, while both differ from the compiled production function by $\approx 2.3 \times 10^{-4}$ on the worst block, a numerical discrepancy shared by two independently written mirrors. The equivalence gate was therefore set at 5×10^{-4} on the basis of this three-way comparison, independently of the scientific outcome. All four slices passed, with maximum relative target discrepancies 6.4×10^{-5}, 9.8×10^{-5}, 1.6×10^{-5} and 1.4×10^{-4}, and scale discrepancies below 2.3×10^{-7}.

H FIELD-CORRECTION CALIBRATION

Audit. For each N, the same N positives are split into disjoint halves and the least-squares β^* is fitted against a disjoint large-support reference in field space, with 16 probes at checkpoint 30k and 8 at 60k. Coefficients at $N = 32, 64, 128, 256$ are 1.7409, 1.3192, 0.9015, 0.6288 at 30k (exponent -0.4957, $R^2 = 0.9958$) and 1.7594, 1.3410, 0.9305, 0.6634 at 60k (exponent -0.4749, $R^2 = 0.9966$). Probe-level spread is a stability diagnostic, not a confidence interval, since probes share parameters and support pools.

Downstream. Continuations run from each seed's checkpoint 30k to 35k under a deterministic paired protocol in which the two arms see identical batches, generator noise and support draws, with the positive bank twice the support size. Table 7 lists every paired difference; Table 8 compares the calibrated coefficient with $\beta = 1$.

Table 6: Affinity-floor diagnostic per slice and pooled. Relative change is $\|V_{\text{clip}} - V_{\text{noclip}}\| \,/\, \|V_{\text{clip}}\|$; norm ratio is $\|V_{\text{noclip}}\| \,/\, \|V_{\text{clip}}\|$. Every temperature branch keeps unit norm ratio because it is RMS-normalized; the change is in direction.

slice	field	floor active	cosine	rel. change	norm ratio
0	$\tau = 0.2$	1.27%	0.99976	0.022	1.000
	$\tau = 0.05$	49.83%	0.99121	0.133	1.000
	$\tau = 0.02$	90.42%	0.96938	0.247	1.000
	aggregated	—	0.98930	0.147	0.975
1	$\tau = 0.2$	1.28%	0.99968	0.025	1.000
	$\tau = 0.05$	48.49%	0.99583	0.091	1.000
	$\tau = 0.02$	90.21%	0.98267	0.186	1.000
	aggregated	—	0.99441	0.106	0.985
2	$\tau = 0.2$	1.15%	0.99941	0.034	1.000
	$\tau = 0.05$	47.49%	0.99482	0.102	1.000
	$\tau = 0.02$	90.12%	0.98060	0.197	1.000
	aggregated	—	0.99329	0.116	0.984
3	$\tau = 0.2$	1.45%	0.99976	0.022	1.000
	$\tau = 0.05$	53.95%	0.99478	0.102	1.000
	$\tau = 0.02$	91.89%	0.97647	0.217	1.000
	aggregated	—	0.99271	0.121	0.980
pooled	$\tau = 0.2$	1.29%	0.99965	0.026	1.000
	$\tau = 0.05$	49.94%	0.99416	0.108	1.000
	$\tau = 0.02$	90.66%	0.97728	0.213	1.000
	aggregated	—	**0.99246**	**0.123**	0.981
	physical σV	—	0.99550	0.095	0.985

Table 7: Paired $\text{FID}(\beta^*) - \text{FID}(0)$ for each evaluation seed.

seed	N	eval 0	eval 1	eval 2	mean
42	32	-0.0941	-0.0341	-0.0455	-0.0579
42	64	-0.0943	-0.0856	$+0.0524$	-0.0425
42	128	-0.1405	$+0.1000$	-0.0690	-0.0365
43	32	-0.0484	$+0.0143$	-0.0899	-0.0413
43	64	-0.0500	-0.0489	$+0.0026$	-0.0321
43	128	-0.0732	$+0.0154$	-0.0362	-0.0313

Table 8: Calibrated β^* versus the parameter-free $\beta = 1$ (FID averaged over three evaluation seeds). The mean difference is -0.0046 and the sign disagrees across seeds at $N = 128$.

seed	N	$\beta = 0$	$\beta = 1$	β^*	$\beta^* - \beta{=}1$
42	32	8.0228	7.9947	7.9649	-0.0298
43	32	8.0661	8.0434	8.0248	-0.0186
42	128	7.5843	7.4728	7.5478	$+0.0750$
43	128	7.6194	7.6329	7.5881	-0.0448