BEYOND VARIANCE MATCHING: DATA-DEPENDENT LAYERWISE STARTS FOR DIFFUSION NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion networks are almost always started from variance-preserving random draws such as Xavier or He initialization and then refined by an optimizer. These initializers stabilize activation and gradient scales, yet they ignore the training distribution: the first gradient steps must discover task structure that a cheap linear algebra pass could already suggest. We ask whether a *data-dependent* start can place ε-prediction diffusion denoisers at better weights that reduce subsequent training cost. We introduce **Layerwise Closed-Form** initialization (LCF), which visits each affine map in depth order, fits a throwaway linear probe to form a residual-augmented target, takes a single exact line-search gradient step from the layer's default random draw toward that target, restores the true nonlinear operator, and proceeds to the next layer. Each hidden weight is thus fit by least squares on the post-activation features of the layers below it, rather than obtained by factorizing one linear map across depth. On 12 diffusion datasets $\times$ 5 denoisers, LCF improves step-0 validation error on **60/60** settings; on convolutional denoisers it also yields positive *net* optimizer savings after subtracting its own cost, on **12/12** large diffusion CNNs (mean $+19.7$ net steps) and **6/6** base CNNs on grayscale sets (mean $+33.5$). At this loose target a zero-initialized output layer matches these savings at no cost; at targets within 1.1–$1.5\times$ of the converged loss, LCF saves more net steps than it on $9/12$ datasets. These results support a training-cost savings claim for diffusion CNNs: a cheap closed-form start can replace optimizer steps.

1 INTRODUCTION

1.1 MOTIVATION

Diffusion denoisers are typically started from variance-preserving random draws such as Xavier (Glorot & Bengio, 2010) or He (He et al., 2015), then refined by AdamW (Loshchilov & Hutter, 2019) to minimize ε-prediction MSE (Ho et al., 2020). These schemes stabilize scales but are *data-agnostic*: the image distribution enters only through stochastic gradients. A survey of initialization (Narkhede et al., 2022) documents a shift toward analytic and data-driven recipes (de Pater & Mitici, 2023; Bingham & Miikkulainen, 2023; Martnishn, 2026). Closed-form affine solves (Tikhonov & Arsenin, 1977) underlie Extreme Learning Machines (Huang et al., 2006), and greedy layerwise pretraining (Hinton et al., 2006; Bengio et al., 2006; Larochelle et al., 2009) showed that how a deep network is started affects what it learns. For ε-prediction, the training loss and each layerwise solve are both least-squares problems, which suggests that a cheap linear-algebra pass can reduce subsequent training cost.

1.2 PROBLEM STATEMENT

Informally: given an ε-prediction network f_θ and a small init minibatch of noisy pairs (x_t, ε), we ask whether closed-form (or single line-search) layerwise solves can produce θ_0 that reaches a fixed validation noise-MSE target in fewer optimizer steps than a strong random baseline—enough to reduce overall training cost.

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.

1.3 APPROACH

We introduce **Layerwise Closed-Form** initialization (LCF). At each affine map W_ℓ, given activations $H_{\ell-1}$, LCF (i) fits a throwaway linear probe, the one-layer auxiliary problem of Belilovsky et al. (2019), $H_{\ell-1} \mapsto Y{=}\varepsilon$, (ii) builds a residual-augmented target, (iii) takes a single exact line-search GD step from the layer's default random draw toward that target, and (iv) restores the true nonlinearity N_ℓ (the activation function) before the next layer. The probe is discarded, and the single step fits each layer only partially (Lemma 2). Rather than fitting one linear map from inputs to targets and factoring it across depth—a construction natural for deep linear networks (Saxe et al., 2014)—LCF fits each hidden layer by least squares on the *post-activation* features of the layers below it (im2col patches for CNNs). Unlike warm-starting from a previously trained model (Ash & Adams, 2020), LCF starts from a fresh random draw and uses only a small batch of the *current* task.

1.4 CONTRIBUTIONS

- **Method.** A layerwise closed-form initializer for ε-prediction diffusion: linear-probe residual targets, a single exact line-search step from the default random draw, a ridge alternative, and elementary properties of both solves (Sec. 3).
- **Empirical evidence.** Step-0 MSE wins on $60/60$; training-cost savings on $12/12$ large CNNs and $6/6$ grayscale base CNNs (Secs. 5–6).

2 RELATED WORK

We situate LCF among weight initialization, greedy/closed-form solvers, deep-linear surrogates, and training-cost evaluation, emphasizing what is *kept* versus *discarded* after each closed-form step.

Variance-preserving and signal-preserving initialization. Xavier (Glorot & Bengio, 2010) and He (He et al., 2015) are our random baselines; LSUV (Mishkin & Matas, 2016) and GradInit (Zhu et al., 2021) are alternative starts we compare for training cost (Sec. 4). Dynamical isometry (Pennington et al., 2017), AutoInit (Bingham & Miikkulainen, 2023), and IDInit (Pan et al., 2025) target well-conditioned propagation without labels. None of these fits a supervised target for each affine map; LSUV and GradInit use data only to set per-layer scales.

Data-driven and analytic task-aware initialization. Surveys catalog the shift to data-dependent starts (Narkhede et al., 2022), including training-set statistics (Koturwar & Merchant, 2017), Sylvester solves (Das et al., 2021), clustered first-batch patches (Martnishn, 2026), discriminant criteria (Chumachenko et al., 2022), Lagrange initial-loss minimization (de Pater & Mitici, 2023), CAWI (Akhtar et al., 2026), Laor (Boongasame et al., 2025), and SWIM-style schemes (Homma & Ohkubo, 2026). LCF is specific: linear-probe residual targets, per-layer ridge / line-search GD, restored nonlinearities, and evaluation by reduced training cost rather than step-0 loss alone.

Greedy layerwise training. Greedy layerwise training (Hinton et al., 2006; Bengio et al., 2006; Larochelle et al., 2009) optimizes one stage at a time; Belilovsky et al. (2019) scale it to ImageNet CNNs with supervised auxiliary heads that are discarded after training, and greedy layerwise recipes also exist for LSTMs (Xu et al., 2018). LCF imports "probe then discard" into an *initialization* pass: one closed-form solve per layer, then end-to-end Adam.

Extreme Learning Machines and non-iterative solvers. ELMs fix random hidden maps and solve heads in closed form (Huang et al., 2006; 2004; Tapson et al., 2015; Wang et al., 2022; 2019). LCF instead solves *every* affine map, restores nonlinearities, and leaves all weights trainable; its headline variant replaces the hidden closed-form solve by one exact line-search GD step. Single GD steps on least-squares objectives also appear in analyses of in-context learning (Mahankali et al., 2024).

Deep linear networks and linear surrogates. Deep linear networks admit exact analyses of how initialization shapes learning dynamics (Saxe et al., 2014; Jacot et al., 2021; Dominé et al., 2023; Bordelon & Pehlevan, 2025). These analyses concern linear maps; LCF instead fits each

layer of a nonlinear network on its post-activation features rather than factoring one global linear map (Remark 3). Mimetic attention (Trockman & Kolter, 2023) and SVD Transformer compression/transfer (Lv et al., 2023; Hsu et al., 2022) exploit weight structure for other goals.

Warm-starting versus surrogate warm-starts. Warm-starting from a prior nonlinear optimum often hurts generalization (Ash & Adams, 2020; Shin et al., 2024); learned warm-starts for fixed-point algorithms (Sambharya et al., 2024) are orthogonal. LCF never starts from a trained model: it moves a fresh random draw by one layerwise step. On CE/NLL tasks a lower initial loss need not cut training cost (Remark 2), in line with that literature's caution that a better start is not automatically a better training trajectory.

Diffusion objectives and compute accounting. Denoising diffusion (Ho et al., 2020; Song et al., 2021; Nichol & Dhariwal, 2021) trains ε_θ with a squared-error loss, the same form as our layerwise solves. Compute-efficient learning surveys (Bartoldson et al., 2023; Shen et al., 2025) motivate treating training cost as first-class; we adopt that view when scoring initializers (Sec. 4). MoE upcycling (Komatsuzaki et al., 2023) reuses pretrained weights; LCF builds a closed-form start from the *current* task's init batch.

3 METHODOLOGY

3.1 NOTATION

An init batch of n examples provides inputs $X \in \mathbb{R}^{n\times d_0}$ and targets $Y \in \mathbb{R}^{n\times d_Y}$ (for diffusion, $Y = \varepsilon$, flattened per example). At a hidden layer, $H \in \mathbb{R}^{n\times d_{\text{in}}}$ denotes the incoming activations, T the residual-augmented target of Eq. (5), $\widetilde{T}$ the target of the affine solve (Step 2), and W_0 the layer's default random draw (PyTorch's layer initialization; Appendix C). In the solves of Step 3, biases are absorbed by appending a constant column to H. We write $H_{\text{aug}} = [H, \mathbf{1}]$ for the bias-augmented design in Step 3 (so Σ, C there use H_{aug} and the bias row is also λ/μ-penalized); Step 1 probes use plain H with no bias. Conceptually λ_{probe} in Eqs. (3)–(4) and λ_{solve} in Eq. (7) are distinct regularizers; both equal 10^{-2} in all runs. We write $\|\cdot\|_F$ for the Frobenius norm,

$$\Sigma = H^\top H, \qquad C = H^\top \widetilde{T}, \tag{1}$$

and $\delta > 0$ for a small numerical stabilizer ($\delta = 10^{-8}$).

3.2 NETWORK AND DATA

We consider a depth-L stack of affine maps and fixed nonlinear operators,

$$H_0 = X, \qquad H_\ell = N_\ell\big(H_{\ell-1}W_\ell + b_\ell\big) \quad (\ell = 1, \dots, L{-}1), \qquad \hat{Y} = H_{L-1}W_L + b_L. \tag{2}$$

The init batch (X, Y) is fixed during the pass and disjoint from the validation data. For diffusion, $X = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \varepsilon$ and $Y = \varepsilon$ (Ho et al., 2020), and time-embedding projections are solved like any other affine map; for classification and language modeling, Y holds one-hot targets even though training minimizes cross-entropy (Remark 2). Convolutions are solved as matrix least squares on im2col patches, with one row per example and output position.

3.3 LAYERWISE PASS

LCF visits the hidden maps $\ell = 1, \dots, L{-}1$ in order, with $H := H_{\ell-1}$ and $W_0 := W_\ell^{(0)}$, and then fits the head W_L.

Step 1: linear probe. Fit a throwaway ridge probe, as in the $k{=}1$ auxiliary problems of Belilovsky et al. (2019), and map its residual $R = Y - HP$ back to feature space:

$$P = \arg\min_M \ \|HM - Y\|_F^2 + \lambda\|M\|_F^2, \tag{3}$$

$$\Delta = R\,(P^\top P + \lambda I_{d_Y})^{-1}P^\top. \quad \text{(same } \lambda > 0 \text{ as Eq. (3))} \tag{4}$$

The probe is never written into the network.

Step 2: residual-augmented target.

$$T = \mathrm{resize}(H) + \alpha \cdot \mathrm{scale}\big(\mathrm{resize}(\Delta)\big), \tag{5}$$

where resize matches the output width (the identity for equal widths), $\mathrm{scale}(Z) = Z \cdot \mathrm{std}(\mathrm{resize}(H))/\mathrm{std}(Z)$ matches the entrywise standard deviation, and $\alpha = 0.25$ at every layer of the diffusion denoisers. The solve target is a pre-activation target $\widetilde{T}$: $\max(T, 0)$ for ReLU, T itself for GELU, and $\widetilde{T} = T$ when no nonlinearity follows the map. For convolutions, Steps 1–2 use globally average-pooled features and broadcast Δ over positions (Appendix C).

Step 3: affine solve. The headline variant takes one exact line-search gradient step from the seed on the layerwise least-squares loss,

$$g = H^\top(HW_0 - \widetilde{T}), \qquad \eta = \frac{\|g\|_F^2}{\|Hg\|_F^2 + \delta}, \qquad W = W_0 - \eta g. \tag{6}$$

The ridge variant instead solves

$$\min_W \ \|HW - \widetilde{T}\|_F^2 + \lambda\|W\|_F^2 + \mu\|W - W_0\|_F^2 \tag{7}$$

in closed form (Lemma 1). An optional gradient-matching penalty for the ridge variant is analyzed in Appendix A; it is off in all reported runs, and the GD variant uses neither it nor μ.

Step 4: safeguards, nonlinearity, next layer. Before W is written, it is rescaled if the root-mean-square of its entries exceeds 3, if the pre-activation standard deviation leaves $[0.25, 2]$, or if its spectral norm exceeds that of W_0 (Appendix C). We then set $W_\ell \leftarrow W$, compute $H_\ell = N_\ell(H_{\ell-1}W_\ell + b_\ell)$ with the true operator, and move to layer $\ell+1$, so every solve sees the activations of the actual network.

Head. The head W_L is ridge-fit to Y on H_{L-1} (Eq. (7) with $\widetilde{T} = Y$ and $\mu = 0$) and passes through the same safeguards; for the convolutional denoisers it is the 1×1 output convolution fit to ε at every pixel.

3.4 Properties of the solves

Proposition 1 (The residual target is at least as linearly predictive)**.** *Let $\ell_\lambda(Z) = \min_M \|ZM - Y\|_F^2 + \lambda\|M\|_F^2$ be the ridge-probe loss of features $Z \in \mathbb{R}^{n\times d_{\mathrm{in}}}$, so that $\ell_\lambda(H)$ is attained at P. (i) Δ in Eq. (4) is the unique minimizer of $\|DP - R\|_F^2 + \lambda\|D\|_F^2$, i.e., the ridge-regularized change of the features whose probe readout best reproduces the residual. (ii) $\ell_\lambda(H + c\Delta) \le \ell_\lambda(H)$ for every $c \in [0, 1]$.*

With equal widths, Eq. (5) reads $T = H + c\Delta$ with $c = \alpha\,\mathrm{std}(H)/\mathrm{std}(\Delta)$ (with $\mathrm{std}(\Delta) > 0$; if $\Delta = 0$ then $T = H$), so T is at least as linearly predictive of Y as H whenever $\mathrm{std}(\Delta) \ge \alpha\,\mathrm{std}(H)$. The convolutional target applies the same construction to pooled features and is a heuristic extension.

Lemma 1 (Ridge solve)**.** *If $\lambda + \mu > 0$, Eq. (7) has the unique minimizer*

$$W^\star = \big(\Sigma + (\lambda + \mu)I\big)^{-1}\big(C + \mu W_0\big). \tag{8}$$

With $\mu = 0$ this is Tikhonov regression (Tikhonov & Arsenin, 1977), the solve used for Extreme Learning Machine heads (Huang et al., 2006).

Lemma 2 (One exact line-search step)**.** *Let $f(W) = \frac{1}{2}\|HW - \widetilde{T}\|_F^2$, $g = \nabla f(W_0)$, and $Hg \neq 0$. (i) $\eta^\star = \|g\|_F^2/\|Hg\|_F^2$ is the unique minimizer of $t \mapsto f(W_0 - tg)$ and lowers f by $\|g\|_F^4/(2\|Hg\|_F^2)$; Eq. (6) uses $\eta < \eta^\star$, with $\eta \to \eta^\star$ as $\delta \to 0$. (ii) The step is partial: the new gradient $g_1 = \nabla f(W_0 - \eta^\star g)$ is orthogonal to g, and $g_1 = 0$ only if every column of g lies in one eigenspace of Σ.*

The headline variant therefore moves the seed toward, but generally not onto, the layerwise least-squares solution, whereas the ridge variant solves the layer exactly up to the μ-pull toward W_0. The reported η with $\delta > 0$ is shorter than $\eta^\star$, so it is even more partial; orthogonality in (ii) is stated for the exact $\eta^\star$ step.

3.5 TRAINING-COST IDENTITY

Reduced training cost is scored by Eq. (9) in Sec. 4; Remark 1 in Appendix A gives the resulting upper bound on savings.

4 EXPERIMENTAL SETUP

Protocol. All tracks (diffusion, classification, language modeling) share one protocol and differ only in the task grid. We use seeds $\{0, \ldots, 4\}$ (mean$\pm$std across seeds) and AdamW with weight decay 0.01 and batch size 64 (32 for LM). All methods use learning rate 10^{-3}, which gives the fewest steps-to-target for Xavier, LSUV, and LCF in a sweep over $\{10^{-4}, 3{\times}10^{-4}, 10^{-3}, 3{\times}10^{-3}\}$ on `diffusion_cnn_large` (5 datasets). Diffusion runs train on $n_{\text{train}}{=}1024$ images and validate on a disjoint split of $n_{\text{val}}{=}256$; the init batch is 512 training samples drawn once per run and never used for validation. Steps-to-target runs use a budget of 200 Adam steps (120 for LM), with validation loss logged at steps 0, 1, and every 10 steps after. Diffusion targets a validation noise-MSE of 0.85; classification targets a validation cross-entropy of 0.8–1.8 depending on the dataset; LM targets a token NLL of 3.5.

Metrics. For each method we record (i) the step-0 validation loss and its gap $\Delta_{\text{init}} = L_{\text{xavier}} - L_{\text{method}}$ (positive = the method starts ahead of Xavier); (ii) steps-to-target S, the first logged step at which validation loss reaches the threshold (the budget if never reached), reported in tables as the seed mean rounded to an integer; (iii) init FLOPs F_{init} and break-even $B = F_{\text{init}}/F_{\text{adam}}$; and (iv) net steps, the Adam steps saved relative to Xavier after paying for the initializer,

$$\text{net} \;=\; \big(S(\theta_{\text{xavier}}) - S(\theta_0)\big) \;-\; \frac{F_{\text{init}}(\theta_0)}{F_{\text{adam}}}, \tag{9}$$

where $S(\theta)$ is the steps-to-target from initialization θ; a method reduces training cost when $\text{net} > 0$.

Convergence protocol. To test whether faster early progress costs final quality, we also train `diffusion_cnn_large` on all 12 diffusion datasets for a fixed $T{=}40{,}000$ steps, identical for all initializers and with no early stopping (Li et al., 2020). The budget was chosen a priori; for every method, the change in validation loss over the final $10{,}000$ steps is below seed noise. We report the plateau MSE and the learning curves. From these curves we also read steps-to-target at *plateau-relative* thresholds $\{1.1, 1.25, 1.5, 2.0\}\times$ the converged plateau MSE. These ask how fast each init reaches a solution of a given quality; we added them after the 0.85 results showed that a zero-cost baseline ties LCF at that target (Sec. 5.1). The base-CNN ablation of Sec. 6 (MNIST, Fashion-MNIST, CIFAR-10) and ZERO-INIT use the same 40k protocol. The 40k runs log validation loss every 200 steps; plateau-relative steps-to-target are linearly interpolated between these evaluations.

Grids. The primary diffusion suite crosses twelve datasets (grayscale digits/letters, natural RGB, and synthetic blob/Gaussian fields) with five denoisers (`diffusion_cnn`, `diffusion_cnn_large`, `diffusion_mlp`, `diffusion_unet`, `diffusion_unet_deep`) and five initializers (Xavier, He, LSUV, GradInit, and LCF GD with $k{=}1$). We add one zero-cost baseline, ZERO-INIT: Xavier followed by zeroing the output layer, as in ADM (Dhariwal & Nichol, 2021). Its output starts at 0, so its step-0 noise-MSE is ≈ 1 and its break-even is $B{=}0$; it is the natural test of whether a data-dependent init earns its cost. The ablation suite (Sec. 6) uses the base diffusion CNN on three datasets. Scope controls cover MLP / CNN classification, synthetic regression, and LSTM and transformer language modeling (Sec. 5.3).

5 RESULTS

We organize results around the FLOP identity of Eq. (9). Table 1 gives family-level results; per-dataset results (Table 6), per-dataset baselines, and the scope controls are in Appendix D.

Init wins are universal; compute wins concentrate on CNNs. LCF GD ($k{=}1$) improves step-0 validation MSE over Xavier on all 60 diffusion settings (12 datasets $\times$ 5 denoisers; Table 8). Net steps are positive for `diffusion_cnn_large` on **12/12** datasets (mean $+19.7$) and for the base CNN

Table 1: **Init baselines vs. Xavier.** Mean net Adam steps (positive = saves compute after FLOP break-even) on diffusion families. At this loose (0.85) target, LSUV and LCF GD (k=1) are the two strongest compute settings: LCF leads on the large CNN, LSUV on the base CNN; He and GradInit are near-neutral or negative under the same protocol. Sec. 5.1 compares against a zero-cost baseline.

Model	He		LSUV		GradInit		LCF GD1	
	wins	mean	wins	mean	wins	mean	wins	mean
diffusion_cnn	0/12	-24.0	12/12	+24.3	0/12	-13.7	7/12	+16.3
diffusion_cnn_large	0/12	-25.2	12/12	+15.5	0/12	-23.8	12/12	+19.7
diffusion_mlp	0/12	-0.5	0/12	-2.4	0/12	-13.8	1/12	-313.4
diffusion_unet	0/12	-2.0	0/12	-19.3	0/12	-12.8	0/12	-39.9
diffusion_unet_deep	0/12	+0.0	0/12	-13.8	0/12	-13.3	0/12	-23.3

on all six grayscale sets (mean $+33.5$), but on only $1/5$ natural-RGB sets (Flowers-102, $+3.2 \pm 8.9$). Several individual wins lie within one standard deviation of zero (e.g. `synth_blobs_28` with the large CNN, $+5.1 \pm 11.4$), so we read Table 6 as a pattern across datasets rather than as a set of individually significant results.

Good inits can still lose on FLOPs. On the RGB base-CNN settings, LCF reaches the target in 10 steps versus 40–46 for Xavier, but break-even is ≈ 33 Adam-equivalents, so net steps are near zero or negative (e.g. CIFAR-10, -0.8 ± 4.5). By Eq. (9), net steps are non-positive whenever $S_{\mathrm{x}} - S_{\mathrm{lcf}} \leq B$. For the large CNN, break-even is only $\approx$ 11–15, so the same 10-step LCF horizon gives a net win.

MLP and U-Net denoisers delimit the claim. MLP denoisers incur a large break-even ($\sim$320 Adam-equivalents), so even large step gaps fail to produce net wins ($1/12$). U-Nets reach the target in roughly ten Xavier steps, which leaves little headroom for any initializer ($0/12$ for both U-Net families).

Baselines under the same identity. He and GradInit do not save net steps on any diffusion family (Table 1). LSUV is a strong competitor: it saves compute on all 24 CNN settings and exceeds LCF on the base CNN ($12/12$ vs. $7/12$; mean $+24.3$ vs. $+16.3$). On `diffusion_cnn_large` both methods win on all 12 datasets; LCF has the higher mean ($+19.7$ vs. $+15.5$), with a margin of $\approx +13$ steps on each grayscale dataset and differences within seed noise elsewhere (Table 7). We therefore do not claim that LCF dominates every initializer at this loose target: it is a data-dependent start that matches the strongest data-agnostic baseline and gives the largest savings on the large CNN. Sec. 5.1 sharpens the comparison against a zero-cost baseline and shows where LCF's advantage becomes decisive.

5.1 A ZERO-COST BASELINE, AND WHEN LCF BEATS IT

A skeptical reading of the compute claim is that LCF's advantage comes only from starting near the trivial "predict-zero-noise" solution, which a zeroed output layer reaches for free. We test this directly with ZERO-INIT (Sec. 4). At the loose 0.85 target the skeptic is right: on `diffusion_cnn_large`, ZERO-INIT saves a mean of $+20.3$ net steps vs. Xavier, matching LCF ($+19.7$) and exceeding it on $8/12$ datasets, at zero init cost. Both start in the same basin (step-0 MSE ≈ 1 for ZERO-INIT, 0.92–0.96 for LCF), and a target this far above the plateau does not separate them. The two also start from different random draws (Xavier for ZERO-INIT, PyTorch's default for LCF; Appendix C).

LCF's advantage appears when the target demands quality. Table 2 sweeps the plateau-relative threshold on `diffusion_cnn_large`. At $2.0\times$ the plateau the two methods tie ($+211$ vs. $+219$; LCF ahead on $3/12$), but from $1.5\times$ inward LCF leads on $9/12$ datasets, and the mean net-step margin over ZERO-INIT grows from $+65 \pm 72$ at $1.5\times$ to $+1088 \pm 950$ at $1.1\times$ (paired, across the 12 datasets). ZERO-INIT only enters the basin; its later trajectory is still Xavier's random features descending, whereas LCF's fitted linear surrogate shapes the whole network toward the data, so its head start compounds as the target tightens. LSUV, the strongest data-agnostic baseline at the loose target, wins on fewer datasets as the target tightens ($10/12$ at $1.5\times$, $7/12$ at $1.1\times$), although its mean saving still rises ($+158$, $+397$, $+363$); LCF saves compute on 11–$12/12$ datasets at every target. We

Table 2: **A zero-cost baseline is matched only at loose targets.** Mean net Adam steps vs. Xavier on `diffusion_cnn_large` (positive = saves compute after break-even), as the steps-to-target threshold tightens from $2.0\times$ to $1.1\times$ the converged plateau MSE. ZERO-INIT (Xavier with a zeroed output layer, ADM-style; step-0 MSE ≈ 1, break-even $B{=}0$) ties LCF at loose targets, but LCF pulls ahead as the target approaches the plateau. Column "LCF>Z" counts datasets where LCF saves more net steps than ZERO-INIT; parenthetical counts are datasets with net > 0; "LCF−Z" is the paired per-dataset difference. Entries are means ± standard deviation across the 12 datasets of the 5-seed means. The plateau is the mean final MSE of Xavier, He, LSUV, GradInit, and LCF; steps-to-target are linearly interpolated between evaluations every 200 steps.

Target (fraction of plateau)	LSUV	ZERO-INIT	LCF	LCF−Z	LCF>Z
$2.00\times$ (loose)	$+84\pm96$ (12/12)	$+219\pm166$ (12/12)	$+211\pm184$ (12/12)	-9 ± 29	3/12
$1.50\times$	$+158\pm239$ (10/12)	$+566\pm446$ (12/12)	$+\mathbf{631}\pm510$ (12/12)	$+65\pm72$	9/12
$1.25\times$	$+397\pm455$ (8/12)	$+1379\pm917$ (12/12)	$+\mathbf{1668}\pm1111$ (11/12)	$+289\pm222$	9/12
$1.10\times$ (high quality)	$+363\pm909$ (7/12)	$+2292\pm1468$ (12/12)	$+\mathbf{3380}\pm1932$ (12/12)	$+1088\pm950$	9/12

Table 3: **Convergence parity.** Plateau (40k-step) validation MSE on `diffusion_cnn_large`, learning rate 10^{-3}, mean over 5 seeds. "LCF−best" is LCF minus the best baseline (negative = LCF better). Every |LCF−best| gap (at most 0.0013) is smaller than the seed standard deviation of the plateau (0.002–0.023).

Dataset	Xavier	He	LSUV	GradInit	ZERO-INIT	LCF	LCF−best
MNIST	0.0647	0.0644	0.0639	0.0649	0.0643	0.0644	+0.0005
Fashion-MNIST	0.1320	0.1330	0.1313	0.1319	0.1294	0.1307	+0.0013
KMNIST	0.1177	0.1173	0.1176	0.1175	0.1167	0.1177	+0.0010
EMNIST-letters	0.0719	0.0722	0.0725	0.0721	0.0723	0.0721	+0.0002
CIFAR-10	0.1132	0.1134	0.1136	0.1131	0.1128	**0.1119**	−0.0009
CIFAR-100	0.1131	0.1138	0.1138	0.1129	0.1132	**0.1117**	−0.0012
SVHN	0.0520	0.0522	0.0524	0.0523	0.0516	**0.0510**	−0.0006
Flowers-102	0.1394	0.1398	0.1396	0.1393	0.1394	**0.1380**	−0.0013
STL-10	0.1134	0.1143	0.1140	0.1137	0.1134	**0.1126**	−0.0008
synth_blobs_28	0.3646	0.3647	0.3645	0.3646	0.3654	0.3645	+0.0000
synth_blobs_32	0.3577	0.3576	0.3571	0.3572	0.3574	**0.3566**	−0.0005
synth_gauss_16	0.3721	0.3724	0.3730	0.3721	0.3732	0.3731	+0.0010

therefore scope the claim precisely: at loose targets a zero-cost trick suffices, but for high-quality solutions LCF saves more compute than both LSUV and ZERO-INIT in Table 2.

5.2 CONVERGENCE: LCF MATCHES FINAL QUALITY

A standing concern for any warm-start is whether faster early progress costs final quality (Ash & Adams, 2020). Under the convergence protocol of Sec. 4, LCF is within one seed standard deviation of the best baseline, ZERO-INIT included, on all 12 datasets (Table 3): lower on 6, higher on 5, and equal on 1. It also reaches a given loss level earlier than the baselines (Fig. 1, Appendix D). The initialization therefore changes the training trajectory, not the reachable plateau.

5.3 SCOPE: CLASSIFICATION AND LANGUAGE MODELING

In the controls with a cross-entropy / NLL training loss, LCF improves the step-0 loss on all 5 classification and 2 LSTM LM settings but saves compute on none (Table 8). Two transformer LMs (TinyShakespeare, $d{=}64$ and $d{=}128$) follow the same pattern and are omitted from the tables: LCF lowers step-0 NLL by 0.64 and 0.78, but Xavier reaches the NLL target within 10 steps, so, as for the U-Nets, no headroom remains and net steps equal $-B$ (-13.7 and -13.4). For these losses the layerwise pass is a misspecified least-squares surrogate (Remark 2), and a better start need not give a shorter trajectory (Ash & Adams, 2020). We therefore scope the positive compute claim to diffusion CNN denoisers.

Table 4: Ablation on `diffusion_cnn` at the 0.85 target: mean net steps (positive = saves compute). One exact line-search step (gd1) gives the most net steps; α has no measurable effect at this target.

Dataset	gd1_a0	gd1_a25	gd1_a50	gd2_a25	ridge_a0	ridge_a25	ridge_a50
cifar10	-0.8±4.5	-0.8±4.5	-0.8±4.5	-8.8±4.5	-17.6±4.5	-17.6±4.5	-17.6±4.5
fashion_mnist	31.9±11.7	31.9±11.7	31.9±11.7	20.2±15.7	8.4±14.8	8.4±14.8	8.4±14.8
mnist	42.1±11.8	42.1±11.8	42.1±11.8	28.4±14.8	18.4±14.8	18.4±14.8	18.4±14.8

Table 5: **Ablation vs. a zero-cost baseline at plateau-relative targets.** Mean net Adam steps vs. Xavier on the base `diffusion_cnn` (positive = saves compute), 3 datasets × 5 seeds, as the steps-to-target threshold tightens from 2.0× to 1.1× the converged plateau MSE. ZERO-INIT is the free baseline (Sec. 4); gd1, gd2, and ridge use α=0.25. At the loose target the free baseline is hard to beat, but from 1.5× inward every LCF solver clears it on all three datasets; gd1 leads at every target. Entries are means ± standard deviation across the 3 datasets of the 5-seed means. The plateau is the mean final MSE of the seven LCF variants of Table 4; steps-to-target are linearly interpolated between evaluations every 200 steps.

Target (of plateau)	ZERO-INIT	ridge	gd2	gd1
2.00× (loose)	+471±293	+465±333	+489±338	**+518**±362
1.50×	+1117±587	+1217±660	+1242±666	**+1305**±705
1.25×	+2234±723	+2577±901	+2603±811	**+2776**±978
1.10× (high quality)	+2867±1013	+4689±738	+4586±1158	**+5413**±940

Datasets clearing ZERO-INIT: at 2.0×, gd1 2/3 and ridge 1/3; at 1.5×, 1.25×, and 1.1×, both 3/3.

6 ABLATIONS

The ablation suite isolates two design choices that matter for the FLOP identity: the layerwise *solver* (ridge vs. one exact line-search GD step) and the probe strength α in Eq. (5). We fix the architecture to the base diffusion CNN and vary methods on MNIST, Fashion-MNIST, and CIFAR-10 (5 seeds). Table 4 reports the design ablation at the 0.85 target; Table 5 then measures the same variants against the zero-cost ZERO-INIT baseline (Sec. 4) at plateau-relative targets, using the 40k convergence runs.

Solver choice dominates. At the 0.85 target, one exact line-search step (gd1) yields roughly +32 to +42 net Adam steps on MNIST and Fashion-MNIST, ridge yields only +8 to +18, and two GD steps (+20 to +28) sit in between while paying a higher init-FLOP bill. At this target the ordering mostly reflects init cost: on CIFAR-10 every solver reaches the target in the same 10 steps, and only break-even differs ($B \approx 33$, 41, and 50 for gd1, gd2, and ridge). The plateau-relative targets below, where init cost is small next to the step gap, give the cleaner comparison of the solvers.

Probe strength has little effect. Sweeping $\alpha \in \{0, 0.25, 0.5\}$ has no measurable effect at the 0.85 target (net steps are identical across α for every solver; Table 4) and a small one at the plateau-relative targets (gd1 stays within ≈ 10 net steps across α at 1.25× the plateau). The dominant lever for compute is whether the affine solve is ridge or one-step GD, not fine adjustments of α.

Both solvers beat the zero-cost baseline at high-quality targets. The 0.85 target is loose enough that the free ZERO-INIT baseline is competitive, so we also read net steps at plateau-relative targets, as in Sec. 5.1. Table 5 shows the result. At 2.0× the plateau the free baseline is hard to beat (gd1 clears it on 2/3 datasets, ridge on 1/3), but from 1.5× inward *every* LCF solver beats ZERO-INIT on all three datasets, and gd1 leads at every target (e.g. +5413 vs. ridge +4689 vs. ZERO-INIT +2867 at 1.1×). So the solver ordering gd1 > gd2 ≈ ridge is robust across targets, and both LCF variants—not only gd1— earn their cost once the target demands quality.

Break-even binds only at the loose target. At the 0.85 target, CIFAR-10 with the base CNN is net-negative for every variant (gd1 −0.8, ridge −17.6): init wins appear, yet break-even dominates Xavier's short remaining horizon. This reverses at plateau-relative targets, where the same base CNN on CIFAR-10 turns strongly net-positive (both gd1 and ridge exceed +450 net steps at 1.5×).

The base-CNN "loss" is therefore a property of the loose target, not of the initializer: as in the main suite, a good init pays off once the target approaches the plateau.

7 Discussion and Limitations

Remark 1 shows that $\mathrm{net} > 0$ requires the init cost to be smaller than Xavier's steps-to-target, $B < S_{\mathrm{x}}$. Diffusion CNNs meet this (Xavier needs dozens of steps; break-even $\sim$11–15 for large CNNs), and their least-squares training loss matches the form of the layerwise solves (Remark 2). MLP denoisers fail on cost, easy U-Net targets on horizon, and CE/NLL tracks on loss alignment. The same gap appears on classification, regression, and language modeling, including transformers (Sec. 5.3): step-0 loss improves, but the path from there is not cheaper than Xavier's.

Limitations: small denoisers relative to modern U-Nets/DiTs; training budgets far shorter than large-scale diffusion training; protocol-chosen validation thresholds, of which the fixed 0.85 is loose (predicting zero noise already scores 1.0); occasional worse final MSE despite earlier target hitting; heuristic attention Q/K. Methodologically, data-dependent initializers should report the training-cost identity (9), not only step-0 loss.

8 Conclusion

We introduced Layerwise Closed-Form initialization (LCF), a Belilovsky-style procedure that solves each affine map with ridge or a single exact line-search gradient step, restores nonlinearities, and proceeds depth-wise without globally SVD-factorizing a task map. Framing evaluation through a FLOP-aware steps-to-target identity (Eq. (9)), we asked not only whether LCF starts ahead of Xavier, but whether it saves net Adam compute after paying for itself. On a 12-dataset diffusion suite the answer is scoped but positive: LCF with one GD step wins the initialization metric on all 60 diffusion settings and yields net-compute gains on large diffusion CNNs (and on base CNNs for grayscale data), while MLP/U-Net denoisers and classification / LM controls show that init wins alone do not imply compute wins. The accompanying analysis gives closed-form uniqueness, exact line-search optimality, and the bound $B < S_{\mathrm{x}}$ that any net win requires (Remark 1).

We recommend that future work on data-dependent initializers report net steps—or an equivalent break-even accounting—alongside step-0 metrics, and that closed-form starts be evaluated under loss regimes that match (or intentionally mismatch) their layerwise surrogates. Extending LCF to other architectures and answering the deep-linear open questions of Appendix A are natural next steps.

AI use statement

In this work, we used generative AI tools for drafting and editing prose, refactoring LaTeX, and assisting with literature search and citation formatting. We have not used generative AI tools to generate experimental results, numerical tables, or figures; those artifacts were produced by our codebase and evaluation pipelines. We reviewed all AI-assisted text for factual correctness and originality, and we take responsibility for the final content of this work, including claims and presentation.

Ethics statement

This work studies weight initialization for diffusion and related models on standard public vision and language benchmarks. It does not involve human subjects, sensitive personal data, or deployment of generative systems for harmful content. We do not foresee significant ethics risks beyond those of ordinary ML methods research; any downstream misuse of faster training recipes would follow from the underlying models rather than from the initializer itself.

Reproducibility statement

All code and data can be found in this anonymized repository (github.com/randomseed357-commits/code). Network definitions, the LCF initializer, FLOP accounting, and analysis scripts

are included with the code; seed grids and metrics are specified in Sec. 4, and all initializer hyperparameters, target constructions, safeguards, and optimizer settings in Appendix C. Formal statements appear in Sec. 3, with complete proofs in Appendix A. An anonymized code archive is provided as supplementary material.

REFERENCES

Mushir Akhtar, M. Tanveer, and Mohd. Arshad. CAWI: Copula-aligned weight initialization for randomized neural networks. In *AISTATS*, 2026.

Jordan Ash and Ryan P Adams. On warm-starting neural network training. In *NeurIPS*, 2020.

Brian R Bartoldson, Bhavya Kailkhura, and Davis Blalock. Compute-efficient deep learning: Algorithmic trends and opportunities. *Journal of Machine Learning Research*, 24(122):1–77, 2023.

Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Greedy layerwise learning can scale to ImageNet. In *ICML*, 2019.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *NeurIPS*, 2006.

Garrett Bingham and Risto Miikkulainen. AutoInit: Analytic signal-preserving weight initialization for neural networks. In *AAAI*, 2023.

Laor Boongasame, Jirapond Muangprathub, and Karanrat Thammarak. Laor initialization: A new weight initialization method for the backpropagation of deep learning. *Big Data and Cognitive Computing*, 9(7):181, 2025.

Blake Bordelon and Cengiz Pehlevan. Deep linear network training dynamics from random initialization: Data, width, depth, and hyperparameter transfer. In *ICML*, 2025.

Kateryna Chumachenko, Alexandros Iosifidis, and Moncef Gabbouj. Feedforward neural networks initialization based on discriminant learning. *Neural Networks*, 146:220–229, 2022.

Debasmit Das, Yash Bhalgat, and Fatih Porikli. Data-driven weight initialization with Sylvester solvers. *arXiv preprint arXiv:2105.10335*, 2021.

Ingeborg de Pater and Mihaela Mitici. A mathematical framework for improved weight initialization of neural networks using Lagrange multipliers. *Neural Networks*, 166:579–594, 2023.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. In *NeurIPS*, 2021.

Clémentine C J Dominé, Lukas Braun, James E Fitzgerald, and Andrew M Saxe. Exact learning dynamics of deep linear networks with prior knowledge. *Journal of Statistical Mechanics: Theory and Experiment*, 2023(11):114004, 2023.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *ICCV*, 2015.

Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

Hikaru Homma and Jun Ohkubo. Neural network initialization with nonlinear characteristics and information on hierarchical features. *Journal of the Physical Society of Japan*, 95(4):044002, 2026.

Yen-Chang Hsu, Ting Hua, Sung-En Chang, Qian Lou, Yilin Shen, and Hongxia Jin. Language model compression with weighted low-rank factorization. In *ICLR*, 2022.

Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: a new learning scheme of feedforward neural networks. In *IJCNN*, 2004.

Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: Theory and applications. *Neurocomputing*, 70(1–3):489–501, 2006.

Arthur Jacot, François Ged, Berfin Şimşek, Clément Hongler, and Franck Gabriel. Saddle-to-saddle dynamics in deep linear networks: Small initialization training, symmetry, and sparsity. *arXiv preprint arXiv:2106.15933*, 2021.

Aran Komatsuzaki, Joan Puigcerver, James Lee-Thorp, Carlos Riquelme Ruiz, Basil Mustafa, Joshua Ainslie, Yi Tay, Mostafa Dehghani, and Neil Houlsby. Sparse upcycling: Training mixture-of-experts from dense checkpoints. In *ICLR*, 2023.

Saiprasad Koturwar and Shabbir Merchant. Weight initialization of deep neural networks (DNNs) using data statistics. *arXiv preprint arXiv:1710.10570*, 2017.

Hugo Larochelle, Yoshua Bengio, Jérôme Louradour, and Pascal Lamblin. Exploring strategies for training deep neural networks. *Journal of Machine Learning Research*, 10:1–40, 2009.

Mengtian Li, Ersin Yumer, and Deva Ramanan. Budgeted training: Rethinking deep neural network training under resource constraints. In *ICLR*, 2020.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

Xiuqing Lv, Peng Zhang, Sunzhu Li, Guobing Gan, and Yueheng Sun. LightFormer: Light-weight Transformer using SVD-based weight transfer and parameter sharing. In *Findings of ACL*, 2023.

Arvind Mahankali, Tatsunori Hashimoto, and Tengyu Ma. One step of gradient descent is provably the optimal in-context learner with one layer of linear self-attention. In *ICLR*, 2024.

Rowan Martnishn. Pre-Warm: Initializing convolutional filters from first-batch patch dictionaries. *arXiv preprint arXiv:2606.25256*, 2026.

Dmytro Mishkin and Jiri Matas. All you need is a good init. In *ICLR*, 2016.

Meenal V Narkhede, Prashant P Bartakke, and Mukul S Sutaone. A review on weight initialization strategies for neural networks. *Artificial Intelligence Review*, 55(1):291–322, 2022.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *ICML*, 2021.

Yu Pan, Chaozheng Wang, Zekai Wu, Qifan Wang, Min Zhang, and Zenglin Xu. IDInit: A universal and stable initialization method for neural network training. In *ICLR*, 2025.

Jeffrey Pennington, Samuel Schoenholz, and Surya Ganguli. Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. In *NeurIPS*, 2017.

Rajiv Sambharya, Georgina Hall, Brandon Amos, and Bartolomeo Stellato. Learning to warm-start fixed-point optimization algorithms. *Journal of Machine Learning Research*, 25(166):1–46, 2024.

Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *ICLR*, 2014.

Li Shen, Yan Sun, Zhiyuan Yu, Liang Ding, Xinmei Tian, and Dacheng Tao. On efficient training of large-scale deep learning models. *ACM Computing Surveys*, 57(3):1–36, 2025.

Baekrok Shin, Junsoo Oh, Hanseul Cho, and Chulhee Yun. DASH: Warm-starting neural network training in stationary settings without loss of plasticity. In *NeurIPS*, 2024.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *ICLR*, 2021.

Jonathan Tapson, Philip de Chazal, and André van Schaik. Explicit computation of input weights in extreme learning machines. In *Proceedings of ELM-2014, Volume 1: Algorithms and Theories*, pp. 41–49. Springer, 2015.

Andrey N Tikhonov and Vasiliy Y Arsenin. *Solutions of Ill-Posed Problems*. Winston & Sons, 1977.

Asher Trockman and J Zico Kolter. Mimetic initialization of self-attention layers. In *ICML*, 2023.

Jian Wang, Siyuan Lu, Shui-Hua Wang, and Yu-Dong Zhang. A review on extreme learning machine. *Multimedia Tools and Applications*, 81(29):41611–41660, 2022.

Xi-Zhao Wang, Tianlun Zhang, and Ran Wang. Noniterative deep learning: Incorporating restricted Boltzmann machine into multilayer random weight neural networks. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(7):1299–1308, 2019.

Kaisheng Xu, Xu Shen, Ting Yao, Xinmei Tian, and Tao Mei. Greedy layer-wise training of long short-term memory networks. In *IEEE ICME Workshops*, 2018.

Chen Zhu, Renkun Ni, Zheng Xu, Kezhi Kong, W. Ronny Huang, and Tom Goldstein. GradInit: Learning to initialize neural networks for stable and efficient training. In *NeurIPS*, 2021.

A Proofs and supplementary statements

A.1 Proofs of the statements in Section 3

Proof of Proposition 1. (i) For $\lambda > 0$ the objective $\|DP - R\|_F^2 + \lambda\|D\|_F^2$ is strongly convex in D. Its gradient $2(DP-R)P^\top + 2\lambda D$ vanishes at $D = RP^\top(PP^\top + \lambda I)^{-1} = R(P^\top P + \lambda I)^{-1}P^\top = \Delta$, where the middle equality is the push-through identity $P^\top(PP^\top + \lambda I)^{-1} = (P^\top P + \lambda I)^{-1}P^\top$. (ii) Using $M = P$ as a feasible probe for the features $H + c\Delta$,

$$\ell_\lambda(H + c\Delta) \le \|(H + c\Delta)P - Y\|_F^2 + \lambda\|P\|_F^2 = q(c) + \lambda\|P\|_F^2, \qquad q(c) := \|c\,\Delta P - R\|_F^2.$$

By (i), Δ does at least as well as $D = 0$, so $q(1) + \lambda\|\Delta\|_F^2 \le q(0)$ and hence $q(1) \le q(0) = \|R\|_F^2$. Since q is convex, $q(c) \le (1-c)\,q(0) + c\,q(1) \le q(0)$ for $c \in [0,1]$, and therefore $\ell_\lambda(H + c\Delta) \le \|R\|_F^2 + \lambda\|P\|_F^2 = \ell_\lambda(H)$. □

Proof of Lemma 1. The objective is quadratic with Hessian $2(\Sigma + (\lambda + \mu)I)$, which is positive definite when $\lambda + \mu > 0$, so the objective is strongly convex. Its gradient $2(\Sigma W - C) + 2\lambda W + 2\mu(W - W_0)$ vanishes exactly at $W^\star$ in Eq. (8). □

Proof of Lemma 2. (i) Since f is quadratic with $\nabla f(W_0) = g$, $f(W_0 - tg) = f(W_0) - t\|g\|_F^2 + \frac{t^2}{2}\|Hg\|_F^2$. For $Hg \neq 0$ this parabola is strictly convex in t and is minimized at $\eta^\star$, with the stated decrease; a positive δ in the denominator only shortens the step. (ii) $g_1 = \Sigma(W_0 - \eta^\star g) - C = g - \eta^\star\Sigma g$, so $\langle g_1, g\rangle = \|g\|_F^2 - \eta^\star\|Hg\|_F^2 = 0$. Moreover $g_1 = 0$ iff $\Sigma g = g/\eta^\star$, i.e., iff every column of g lies in the eigenspace of Σ with eigenvalue $1/\eta^\star$. □

A.2 Optional gradient-matching penalty (ridge variant)

The implementation can add $\nu\|g(W) - \kappa\, g(W_0)\|_F^2$ to Eq. (7), where $g(W) = H^\top(HW - \widetilde{T}) = \Sigma W - C$ is the layerwise least-squares gradient and $\kappa \in [0,1]$. The penalty is off ($\nu = 0$) in all reported runs and is never used by the GD variant.

Lemma 3 (Gradient-matching limit)**.** *Assume $\Sigma \succ 0$, let $W_{\mathrm{LS}} = \Sigma^{-1}C$ be the unregularized least-squares solution, and for $\lambda, \mu \ge 0$ and $\nu > 0$ let W_ν minimize*

$$J_\nu(W) = \|HW - \widetilde{T}\|_F^2 + \lambda\|W\|_F^2 + \mu\|W - W_0\|_F^2 + \nu\,\|g(W) - \kappa\, g(W_0)\|_F^2.$$

Then W_ν is unique and, as $\nu \to \infty$, $W_\nu \to W_\kappa := (1-\kappa)\,W_{\mathrm{LS}} + \kappa\, W_0$, the unique matrix whose layerwise gradient equals $\kappa\, g(W_0)$.

Proof. Because $\Sigma W_\kappa = (1-\kappa)C + \kappa\Sigma W_0$, the penalty equals $\nu\|\Sigma(W - W_\kappa)\|_F^2$, which is strongly convex when $\Sigma \succ 0$, so J_ν has a unique minimizer. Comparing W_ν with W_κ and using that the other terms are nonnegative gives $\nu\|\Sigma(W_\nu - W_\kappa)\|_F^2 \le J_\nu(W_\kappa) = J_0(W_\kappa)$, hence $\|W_\nu - W_\kappa\|_F \le \sigma_{\min}(\Sigma)^{-1}\sqrt{J_0(W_\kappa)/\nu} \to 0$. □

The limit keeps a κ-fraction of the seed's *layerwise* gradient; the lemma says nothing about the gradient of the end-to-end training loss. If Σ is singular (e.g., more features than samples), the matrices with $g(W) = \kappa\, g(W_0)$ form an affine subspace, so W_κ is not unique.

A.3 Training cost

Remark 1 (An upper bound on savings). For net in Eq. (9), write $S_{\mathrm{x}} = S(\theta_{\mathrm{xavier}})$, $S_0 = S(\theta_0)$, and $B = F_{\mathrm{init}}/F_{\mathrm{adam}}$. Because $S_0 \ge 0$, $\mathrm{net} \le S_{\mathrm{x}} - B$, with equality iff the target is met at initialization. Hence an initializer can reduce training cost relative to Xavier only if $B < S_{\mathrm{x}}$, and by at most $S_{\mathrm{x}} - B$ steps: the saving is capped when Xavier already reaches the target quickly or when the initializer is expensive relative to one optimizer step.

In the spirit of compute-efficient training (Bartoldson et al., 2023), a lower step-0 loss alone therefore cannot certify a saving.

Remark 2 (Loss alignment). For diffusion and regression, the training loss and every layerwise solve are least-squares objectives. For classification and language modeling, the solves remain least squares on one-hot targets while training minimizes cross-entropy, so a lower step-0 loss need not translate into fewer optimizer steps (Table 8; cf. the warm-start results of Ash & Adams, 2020).

A.4 Why not factor one global map

Remark 3. An alternative start fits one ridge map $A^\star$ from X to Y and factors it across depth, e.g. $W_L = S^{1/L}V^\top$ for $A^\star = USV^\top$ (economy SVD, hidden width $r = \mathrm{rank}$; no middles when $L = 2$). The intermediate maps are then diagonal in one shared singular basis by construction, which suits deep linear networks (Saxe et al., 2014) but is generally not preserved once elementwise nonlinearities are inserted. LCF instead fits each layer on the post-activation features of the layers below it.

Open questions. Whether layerwise least-squares starts shorten the plateaus of deep linear networks (Saxe et al., 2014), and where a single line-search step places a network between the lazy and rich regimes (Jacot et al., 2021), remain open; none of our claims depend on them.

B Beyond diffusion: what we tried, and why diffusion is different

The same initializer was run on classification (MLP and CNN), synthetic regression, LSTM language modeling, and transformer language models, under the protocol of Sec. 4. On these tasks the step-0 validation loss is typically lower than Xavier's (Table 8 for the classification and LSTM rows; Sec. 5.3 for the transformers). That head start does not reliably shorten the rest of training. After subtracting the cost of the initializer, Xavier is as cheap or cheaper on the classification and language-modeling tracks, and the same pattern appears on the non-convolutional diffusion models. A lower loss at step 0 is not the same as an easier path: Adam still has to cross the loss surface from the initialized weights, and on these tasks that surface is not cheaper to cross than from a fresh Xavier draw. We treat this as a limitation of the current initializer, not as evidence that a data-dependent start cannot help outside diffusion, and we are investigating changes that would make LCF reduce training cost for a wider range of networks.

Diffusion CNNs are the setting in which the start also reduces training cost, for two reasons that do not hold together elsewhere.

The layerwise solve is a piece of the training loss. A diffusion denoiser is trained with squared error on ε, and every LCF solve is also a least-squares problem (Remark 2): the head is fit to ε itself,

and each hidden layer to a residual-augmented feature target built from a linear probe of ε. Moving a layer toward its least-squares fit therefore moves it toward a surrogate of the objective Adam will minimize. Classification and language modeling do not have this match: the probe is least squares on one-hot targets, while Adam minimizes cross-entropy, so a better linear fit of the one-hot matrix need not be a better start for the training loss. Regression is least squares, but on the small MLPs we tried the initializer costs more, relative to one Adam step, than the step gap it creates.

The start is only a short step, on a network Xavier still has to work on. The headline variant takes one line-search step per layer (Lemma 2), which is enough to help when two further conditions hold. First, a linear function of the current features already predicts a useful part of ε: the forward noising process is itself linear in (x_0, ε), so early layers of a convolutional denoiser see a target that a cheap least-squares fit can partially capture. Second, Xavier still needs many optimizer steps to hit the validation target, and the initializer is cheap next to that horizon (Remark 1). U-Nets miss the second condition (Xavier is already near the target), and wide MLP denoisers miss it in the other direction (the solve is expensive). Transformers miss the first: softmax attention is not a linear map of its inputs, and the query and key projections are left at the identity rather than solved (Appendix C), so most of the block is barely moved toward the task.

C Implementation details

Seeds and hyperparameters. The Xavier and Zero-Init baselines start linear maps from Xavier-uniform draws and convolutions (including the transposed convolution) from Kaiming-normal draws (fan-out, ReLU gain), with zero biases; He and GradInit start from He draws, and LSUV from an orthogonal pre-initialization. LCF starts its solves from the default PyTorch initialization of each layer (uniform with bound $1/\sqrt{\text{fan-in}}$ for weights and biases). Probes and ridge solves use $\lambda = 10^{-2}$; probes have no bias. Hidden ridge solves use $\mu = \bar{\mu}\,\mathrm{tr}(\Sigma)/d$ with $\bar{\mu} = 1$ (d the number of columns of the design), heads use $\mu = 0$, and the gradient-matching penalty of Lemma 3 is off ($\nu = 0$, so $\kappa = 0.5$ is unused). The GD variant takes one step with $\delta = 10^{-8}$; biases start at zero and take the same step. The diffusion denoisers use $\alpha = 0.25$ at every layer. The classification MLP and CNN ramp α linearly from 0.2α on the first hidden map to α on the last, and the classification CNN re-solves a layer with α halved (at most twice) when the solved map's spectral norm exceeds three times that of its seed.

Targets. For dense layers, resize is the identity for equal widths, tiles columns cyclically when widening, and averages consecutive groups of $\lceil d_{\text{in}}/d_{\text{out}} \rceil$ columns (zero-padded) when narrowing; $\mathrm{scale}(Z) = Z \cdot \mathrm{std}(\mathrm{resize}(H))/\mathrm{std}(Z)$ with standard deviations taken over all entries. For a convolution with input $H \in \mathbb{R}^{n \times C_{\text{in}} \times h \times w}$, the probe is fit on the globally average-pooled features $\bar{H} \in \mathbb{R}^{n \times C_{\text{in}}}$; the resulting Δ is resized to C_{out} channels and broadcast over output positions, and $\mathrm{resize}(H)$ is the channel mean of H, bilinearly resampled to the output grid and copied to every output channel. The inverse activations are $\max(\cdot, 0)$ for ReLU, the identity for GELU, atanh after clipping to $\pm(1 - 10^{-3})$ for tanh, and the logit after clipping to $[10^{-3}, 1 - 10^{-3}]$ for the sigmoid.

Safeguards. After every solve, including the heads, and in this order: (a) if the root-mean-square entry of W exceeds 3, W is rescaled to RMS 3; (b) if the pre-activation standard deviation $\mathrm{std}(HW)$ lies outside $[0.25, 2]$, (W, b) is rescaled so that it equals 0.8; (c) if the spectral norm of W (12 power iterations) exceeds that of W_0, (W, b) is rescaled to match it.

Time conditioning and architectures. Timesteps enter through a sinusoidal embedding. In the CNN denoisers the time projection is solved first, with a dense target and no inverse activation; its output is added to the first convolution's pre-activation before the ReLU, as in the forward pass, but the first convolution's own solve does not include this term. In the U-Nets the two-layer time MLP is solved the same way, and its output is added to the input of the downsampling convolution. The U-Net encoder, downsampling, middle, and decoder convolutions are solved in forward order; the transposed-convolution upsampler keeps its Kaiming draw; and the decoder convolution is solved on the channel concatenation of the upsampled features and the encoder skip features, as in the forward pass. None of the denoisers contains normalization layers. For the attention and LSTM models of Appendix B, query and key projections are set to the identity so that softmax is well defined, value

Table 6: **Per-dataset compute results.** Diffusion CNN denoisers only (train/val loss = ε-prediction MSE; see Table 8 for CE / NLL tracks). LCF GD (k=1) vs. Xavier. Positive Net = Adam steps saved to the val MSE target after subtracting init cost $F_{\mathrm{lcf}}/F_{\mathrm{adam}}$. Mean±std over 5 seeds.

Dataset	Model	Loss	Δ_0	Xavier steps	LCF steps	Net
kmnist	diffusion_cnn_large	MSE (ε)	23.93±21.96	48	1	36.1±4.5
mnist	diffusion_cnn_large	MSE (ε)	24.08±21.63	48	1	36.1±4.5
emnist_letters	diffusion_cnn_large	MSE (ε)	23.99±21.93	46	1	34.1±5.5
fashion_mnist	diffusion_cnn_large	MSE (ε)	23.50±22.46	44	1	32.1±5.5
flowers102	diffusion_cnn_large	MSE (ε)	4.35±0.97	44	10	19.2±11.4
stl10	diffusion_cnn_large	MSE (ε)	4.29±1.04	40	10	15.2±7.1
cifar10	diffusion_cnn_large	MSE (ε)	4.29±1.03	38	10	13.2±8.4
cifar100	diffusion_cnn_large	MSE (ε)	4.31±1.02	38	10	13.2±8.4
svhn	diffusion_cnn_large	MSE (ε)	4.28±1.09	36	10	11.2±8.9
synth_gauss_16	diffusion_cnn_large	MSE (ε)	21.12±20.99	34	12	11.1±8.4
synth_blobs_32	diffusion_cnn_large	MSE (ε)	4.20±1.05	34	10	9.2±5.5
synth_blobs_28	diffusion_cnn_large	MSE (ε)	22.48±23.30	34	18	5.1±11.4
mnist	diffusion_cnn	MSE (ε)	16.42±10.31	58	4	42.1±11.8
kmnist	diffusion_cnn	MSE (ε)	16.41±9.97	58	5	41.9±11.7
emnist_letters	diffusion_cnn	MSE (ε)	16.34±10.02	54	5	37.9±12.5
fashion_mnist	diffusion_cnn	MSE (ε)	16.29±9.52	48	5	31.9±11.7
synth_gauss_16	diffusion_cnn	MSE (ε)	15.23±8.69	44	8	24.7±11.3
synth_blobs_28	diffusion_cnn	MSE (ε)	15.93±9.17	42	8	22.3±8.8
flowers102	diffusion_cnn	MSE (ε)	3.89±1.96	46	10	3.2±8.9
cifar10	diffusion_cnn	MSE (ε)	3.79±1.90	42	10	-0.8±4.5
cifar100	diffusion_cnn	MSE (ε)	3.86±1.96	42	10	-0.8±4.5
stl10	diffusion_cnn	MSE (ε)	3.76±1.89	42	10	-0.8±4.5
svhn	diffusion_cnn	MSE (ε)	3.71±1.84	40	10	-2.8±7.1
synth_blobs_32	diffusion_cnn	MSE (ε)	3.74±1.90	40	10	-2.8±7.1

Compute wins: 19/24 (cnn_large 12/12; base grayscale 6/6). Per-setting baseline nets in Table 7.

and output projections are solved like other affine maps, and the four LSTM gates share one affine map on $[x_t; h_{t-1}]$ whose targets are inverted with logit/atanh before the true LSTM step is unrolled. The MLP denoiser concatenates the flattened x_t with the time embedding and solves its two GELU layers with dense targets.

Diffusion data. The init batch holds the first 512 images of the training subset, with $t \sim \mathrm{Uniform}\{0, \ldots, 99\}$ and fresh noise $\varepsilon \sim \mathcal{N}(0, I)$ under a linear β schedule from 10^{-4} to 0.02 over 100 steps. Training minibatches and the fixed validation set draw t the same way.

Cost accounting and optimizer. F_{init} counts the multiply–adds of every probe, solve (Gram matrix, right-hand side and dense solve for ridge; three matrix products per GD step), power iteration, and forward pass through each solved layer. F_{adam} is three times the analytic forward cost of one minibatch of 64. Runs initialized with LCF give each weight matrix the AdamW learning rate $\mathrm{lr}/\max(1, \|W\|_2)$, with $\|W\|_2$ measured right after initialization; biases and all baseline runs use the base learning rate. Training LCF with the base learning rate instead changes net steps by at most 1.8 on both diffusion CNNs (MNIST, Fashion-MNIST, CIFAR-10; 5 seeds), so this choice does not drive the comparison.

D ADDITIONAL RESULTS

Table 6 gives per-dataset results for the two CNN denoisers, and Table 7 per-dataset net steps for all baselines. Table 8 relates the training loss of each track to the layerwise least-squares surrogate. Figure 1 shows the full 12-dataset convergence curves.

Table 7: **Baselines on diffusion CNNs (same protocol as Table 6).** Mean net Adam steps vs. Xavier (positive = saves compute after FLOP break-even). Loss = ε-MSE for all rows.

Dataset	Model	He	LSUV	GradInit	LCF
cifar10	diffusion_cnn_large	-20.0	12.9±5.6	-17.3±5.5	13.2±8.4
cifar100	diffusion_cnn_large	-24.0±5.5	10.9±5.5	-25.3±4.5	13.2±8.4
emnist_letters	diffusion_cnn_large	-34.0±5.5	20.9±5.5	-29.3±8.9	34.1±5.5
fashion_mnist	diffusion_cnn_large	-28.0±4.5	19.0±4.5	-29.3±8.9	32.1±5.5
flowers102	diffusion_cnn_large	-18.0±4.5	19.0±8.4	-17.3±5.5	19.2±11.4
kmnist	diffusion_cnn_large	-32.0±4.5	23.0±5.5	-27.3±8.9	36.1±4.5
mnist	diffusion_cnn_large	-32.0±4.5	22.9±5.4	-29.3±8.9	36.1±4.5
stl10	diffusion_cnn_large	-20.0	14.9±4.4	-17.3±5.5	15.2±7.1
svhn	diffusion_cnn_large	-22.0±4.5	12.9±9.0	-19.3±5.5	11.2±8.9
synth_blobs_28	diffusion_cnn_large	-24.0±5.5	11.0±5.5	-23.3±7.1	5.1±11.4
synth_blobs_32	diffusion_cnn_large	-22.0±4.5	6.9±7.2	-23.3±7.1	9.2±5.5
synth_gauss_16	diffusion_cnn_large	-26.0±5.5	11.0±5.5	-27.3±5.5	11.1±8.4
cifar10	diffusion_cnn	-16.0±5.5	17.6±6.8	-11.3±4.5	-0.8±4.5
cifar100	diffusion_cnn	-18.0±4.5	17.6±6.8	-15.3±4.5	-0.8±4.5
emnist_letters	diffusion_cnn	-32.0±13.0	33.6±19.5	-15.3±4.5	37.9±12.5
fashion_mnist	diffusion_cnn	-32.0±8.4	27.6±20.0	-17.3±8.9	31.9±11.7
flowers102	diffusion_cnn	-18.0±4.5	23.6±9.0	-15.3±4.5	3.2±8.9
kmnist	diffusion_cnn	-34.0±16.7	33.6±18.2	-11.3±4.5	41.9±11.7
mnist	diffusion_cnn	-34.0±16.7	35.6±16.5	-11.3±4.5	42.1±11.8
stl10	diffusion_cnn	-18.0±8.4	17.6±6.8	-15.3±4.5	-0.8±4.5
svhn	diffusion_cnn	-18.0±4.5	17.6±7.0	-15.3±4.5	-2.8±7.1
synth_blobs_28	diffusion_cnn	-22.0±13.0	25.5±11.1	-13.3	22.3±8.8
synth_blobs_32	diffusion_cnn	-18.0±4.5	15.6±8.1	-11.3±4.5	-2.8±7.1
synth_gauss_16	diffusion_cnn	-28.0±14.8	25.6±16.5	-11.3±4.5	24.7±11.3

Compute wins: He 0/24, LSUV 24/24, GradInit 0/24, LCF 19/24.

Table 8: **Loss geometry and compute.** Training / validation objective vs. the LCF least-squares solve, with `lcf_gd1` init / compute wins vs. Xavier (positive mean net = Adam steps saved after FLOP break-even). MSE tracks can win on compute; CE / NLL tracks do not under this protocol.

Track	Model	Train / val loss	Match	n	Init	Compute	Mean net
cls	cnn	cross-entropy	—	3	3/3	0/3	-35.4
cls	mlp_cls	cross-entropy	—	2	2/2	0/2	-49.2
diffusion	diffusion_cnn	MSE ($\|\|\varepsilon_\theta - \varepsilon\|\|_2^2$)	✓	12	12/12	7/12	+16.3
diffusion	diffusion_cnn_large	MSE ($\|\|\varepsilon_\theta - \varepsilon\|\|_2^2$)	✓	12	12/12	12/12	+19.7
diffusion	diffusion_mlp	MSE ($\|\|\varepsilon_\theta - \varepsilon\|\|_2^2$)	✓	12	12/12	1/12	-313.4
diffusion	diffusion_unet	MSE ($\|\|\varepsilon_\theta - \varepsilon\|\|_2^2$)	✓	12	12/12	0/12	-39.9
diffusion	diffusion_unet_deep	MSE ($\|\|\varepsilon_\theta - \varepsilon\|\|_2^2$)	✓	12	12/12	0/12	-23.3
lm	lstm_lm	token CE / NLL	—	2	2/2	0/2	-25.9
reg	mlp_reg	MSE ($\|\|\hat{y} - y\|\|_2^2$)	✓	1	1/1	0/1	-16.4
Overall (`lcf_gd1`)				68	68/68	20/68	—

Match = train/val loss shares least-squares geometry with LCF. CE / NLL tracks are scope / negative controls. The regression init win holds for the seed mean but on only 3 of 5 seeds.

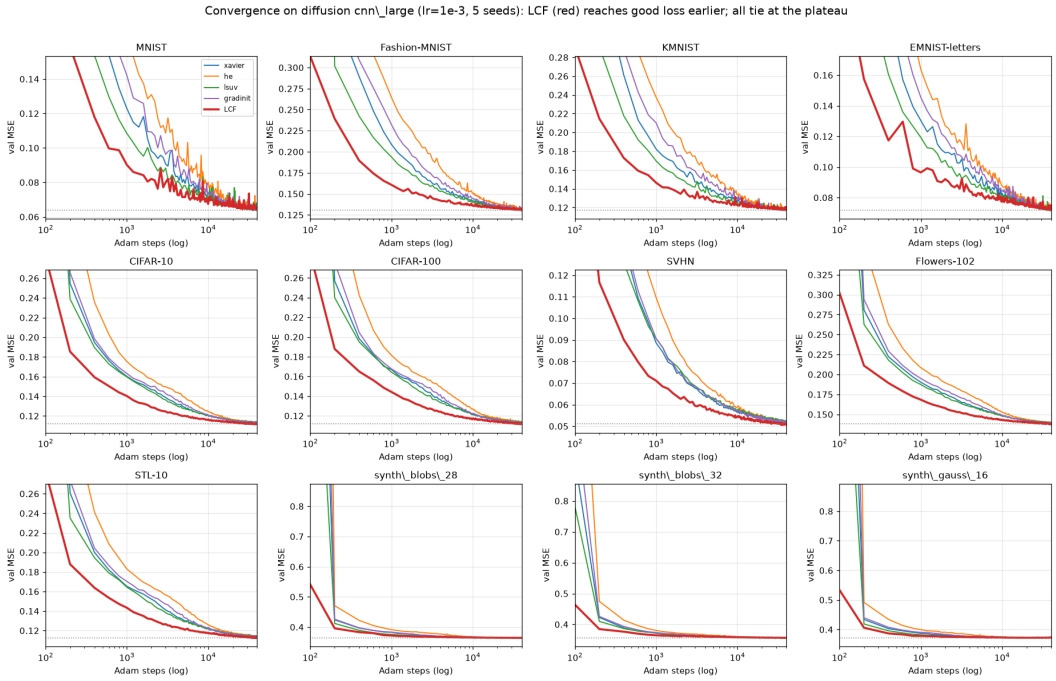


Figure 1: Validation MSE vs. optimizer steps on `diffusion_cnn_large` (log scale, 5 seeds). LCF (red) reaches a low loss earlier; all initializers reach the same plateau.