LEARNING ANISOTROPIC PREDICTION TARGETS FOR FLOW MODELS

Rayhan Zirvi[1,2*] **Hansheng Chen**[1*] **Gordon Wetzstein**[1] **Leonidas Guibas**[1]

[1]Stanford University [2]California Institute of Technology [*]Equal contribution

https://hanshengchen.com/anisoflow



(a) Text-to-image generations in pixel space

(b) FP8 blockwise inference

Figure 1: **(a)** High-resolution images generated by AnisoFlow, a pixel-space diffusion transformer with only 918M parameters trained from scratch, using a learned anisotropic prediction target. **(b)** By learning a well-conditioned prediction target, AnisoFlow removes the patch-seam artifacts produced by the $\boldsymbol{x}$-prediction baseline under FP8 blockwise (W8A8) quantization (DeepSeek-AI et al., 2025).

ABSTRACT

Flow-based generative models are trained on a hand-chosen prediction target, such as the velocity, the noise, or the clean data. This choice strongly affects training and sample quality. We introduce *Anisotropic Flow Models* (AnisoFlow), which replaces a fixed prediction target with a fully learnable one. AnisoFlow learns to scale the noise component of the target anisotropically, assigning a separate non-negative weight to each direction in a jointly learned basis. A closed-form map recovers the velocity from the network's prediction, leaving the architecture, flow-matching loss, and sampler unchanged. Since recovery is exact, training optimizes the target purely to make it easier for the network to predict. We theoretically analyze the resulting prediction error and predict that the preferred noise weight in each direction scales near-linearly with data variance, in agreement with the weights learned empirically in pixel space. AnisoFlow improves FID from 1.90 to 1.55 over the JiT baseline on ImageNet 256×256, by learning only the prediction target, without representation alignment or architectural changes. In pixel-space text-to-image generation, AnisoFlow improves image quality and, through better-conditioned velocity recovery, largely removes patch-seam artifacts under reduced-precision inference.

1 INTRODUCTION

Modern image and video generators use diffusion or flow objectives (Ho et al., 2020; Song et al., 2021; Lipman et al., 2023; Liu et al., 2023; Albergo & Vanden-Eijnden, 2023) to train scalable denoising networks (Peebles & Xie, 2023; Brooks et al., 2024) to predict a target along the noising path, such as the noise $\boldsymbol{\epsilon}$, the clean data $\boldsymbol{x}$, or the velocity $\boldsymbol{u} \coloneqq \boldsymbol{\epsilon} - \boldsymbol{x}$. The choice of target is a central design decision: given the noisy input and timestep, these quantities carry the same information, yet they define distinct regression problems with substantially different training behavior and sample quality (Karras et al., 2022; Salimans & Ho, 2022). Existing hand-designed choices occupy a small corner of a much larger design space: velocity in latent models (Esser et al., 2024; Black Forest Labs, 2025; HaCohen et al., 2024; Kong et al., 2025; Wang et al., 2025) or clean data in recent pixel models (Li & He, 2026; Guo et al., 2026), with methods also exploring interpolation between standard targets (Jin & Wang, 2026) or masking part of the noise component (Chen et al., 2026a).

We introduce *Anisotropic Flow Models* (AnisoFlow), which replaces this hand-designed choice with a fully learnable one. AnisoFlow learns to scale the noise component of the target anisotropically. It defines the target $\boldsymbol{u}_W \coloneqq \boldsymbol{W}\boldsymbol{\epsilon} - \boldsymbol{x}$, where $\boldsymbol{W}$ is a positive semidefinite (PSD) matrix that applies a separate learnable nonnegative weight w_i to the noise along each direction of a jointly learned orthogonal basis. Up to sign and overall scale, the target family includes noise prediction at $w_i \to +\infty$, clean-data prediction at $w_i = 0$, velocity prediction at $w_i = 1$, shared scalar interpolations at $w_i \equiv w$, and binary low-rank masks at $w_i \in \{0, 1\}$. A closed-form map recovers the velocity from the network's prediction, leaving the architecture, flow-matching loss, and sampler unchanged.

AnisoFlow is particularly effective in pixel space, where patchification places many pixel coordinates within each transformer token, exposing directions with widely varying data variance (Li & He, 2026; Chen et al., 2026a) and creating a setting where different directions can benefit from different prediction targets. The velocity target places unit noise weight in every direction, increasing the difficulty of prediction in low-variance directions. Clean-data prediction removes the noise component but can be harder in high-variance directions and makes velocity recovery ill-conditioned at low noise. AnisoFlow resolves this tradeoff by learning a separate noise weight for each direction.

We theoretically analyze which noise weights make prediction easiest for the network. We study the prediction error in the PCA basis, which diagonalizes the data covariance and closely matches the basis that AnisoFlow learns empirically. Under a local frozen-error model, we derive the preferred weight in each direction and show that it scales near-linearly with data variance. Consistent with this analysis, pixel-space models learn weights that follow the scaling $w_i \approx c\lambda_i$, where λ_i is the data variance in direction i.

Empirically, AnisoFlow improves FID from 1.90 to 1.55 over the JiT-H/16 baseline on ImageNet 256×256 in pixel space, by learning only the prediction target, without representation alignment or architectural changes. In pixel-space text-to-image generation, AnisoFlow improves image quality and, through better-conditioned velocity recovery, largely eliminates patch-seam artifacts under reduced-precision inference. In latent-space models, AnisoFlow recovers weights close to the default velocity target in SiT (Ma et al., 2024) with SD-VAE (Rombach et al., 2022) and the default clean-data target in RAEv2 (Singh et al., 2026), achieving comparable performance.

Our contributions include (a) AnisoFlow, a fully learnable prediction-target family that subsumes standard targets as special cases; (b) a theoretical analysis relating the preferred noise weights to the network's prediction errors, explaining the observed near-linear scaling and approximate timestep invariance; and (c) state-of-the-art pixel-space generation on ImageNet and improved quality and stability in text-to-image generation.

2 RELATED WORK

Standard prediction parameterizations. The choice among noise ($\boldsymbol{\epsilon}$)-, velocity ($\boldsymbol{u}$)-, and clean-data ($\boldsymbol{x}$)-prediction determines how the target combines the data and noise components and can substantially affect training behavior and sample quality (Karras et al., 2022; Salimans & Ho, 2022; Jin & Wang, 2026). Noise- and velocity-prediction assign a nonzero noise coefficient to every direction, which can make regression difficult in high-dimensional pixel space (Li & He, 2026; Chen et al., 2026a; Jin & Wang, 2026). Clean-data prediction avoids predicting the noise, but recovering

the velocity from the predicted data amplifies its error by a factor proportional to $1/\sigma_t$, which is ill-conditioned at low noise (Karras et al., 2022; Salimans & Ho, 2022; Chen et al., 2026a).

Interpolated targets. Isotropic interpolations mix data and noise with the same coefficient in every direction: EDM (Karras et al., 2022) sets this coefficient analytically through a σ_t-dependent skip connection, and k-Diff (Jin & Wang, 2026) learns it as a single scalar. AsymFlow (Chen et al., 2026a) is anisotropic but binary, combining velocity prediction in a low-rank subspace and data prediction in the orthogonal complement. Both k-Diff and AsymFlow lie within AnisoFlow's target family, as a shared scalar and a binary profile. Appendix A.2 shows that a per-direction extension of EDM's choice is exactly what our analysis predicts when the network has no information beyond the timestep. Recent work also studies how prediction parameterizations interact with representation dimension, covariance conditioning, and low-variance directions (Fu et al., 2026; Gagneux et al., 2026; Zhang et al., 2026b; Wang & Pehlevan, 2025; Wang et al., 2026b; Ahamed et al., 2026).

Rotation-based quantization. The rotate-then-scale form of AnisoFlow's recovery resembles rotation-based quantization (Ashkboos et al., 2024; Liu et al., 2025), which applies fixed or learned orthogonal transforms to weights and activations to suppress outliers before quantization. These methods operate layer-wise on a trained network, whereas AnisoFlow learns network-wide preconditioning through the prediction target, jointly with the network.

3 PRELIMINARIES

Flow matching. Let $\boldsymbol{x} \sim p_{\mathrm{data}}$ denote a data sample and let $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ denote Gaussian noise. We consider the linear path

$$\boldsymbol{z}_t = \alpha_t \boldsymbol{x} + \sigma_t \boldsymbol{\epsilon}, \qquad \alpha_t = 1 - t, \qquad \sigma_t = t, \qquad t \in [0, 1], \tag{1}$$

so that $\boldsymbol{z}_0 = \boldsymbol{x}$ and $\boldsymbol{z}_1 = \boldsymbol{\epsilon}$. Its sample velocity is $\boldsymbol{u} = \partial_t \boldsymbol{z}_t = \boldsymbol{\epsilon} - \boldsymbol{x}$. A flow model $\boldsymbol{u}_\theta(\boldsymbol{z}_t, t)$ is trained by regressing onto this sample velocity; under squared error, the optimal predictor is the conditional mean $\mathbb{E}[\boldsymbol{u} \mid \boldsymbol{z}_t, t]$, which defines the ODE velocity field. The training objective is

$$\mathcal{L}_{\mathrm{FM}} = \mathbb{E}_{\substack{t \sim p(t),\, \boldsymbol{x} \sim p_{\mathrm{data}}, \\ \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})}} \Big[\| \boldsymbol{u}_\theta(\boldsymbol{z}_t, t) - \boldsymbol{u} \|^2 \Big]. \tag{2}$$

At sampling time, $\boldsymbol{z}_1 \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ is integrated from $t = 1$ to $t = 0$ using the learned velocity field (Lipman et al., 2023; Liu et al., 2023; Albergo & Vanden-Eijnden, 2023).

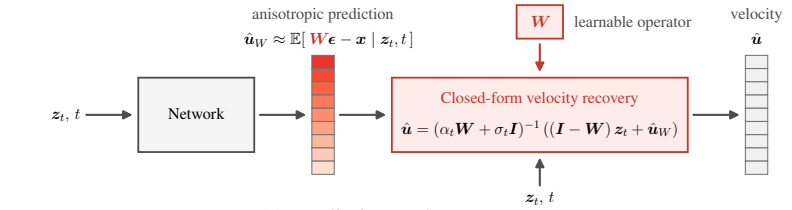
Prediction targets and recovery. The network need not predict the velocity $\boldsymbol{u}$ directly; it may instead predict the noise $\boldsymbol{\epsilon}$, the clean data $\boldsymbol{x}$, or another linearly related target. For any interior timestep $t \in (0, 1)$, the relation $\boldsymbol{z}_t = \alpha_t \boldsymbol{x} + \sigma_t \boldsymbol{\epsilon}$ allows either standard prediction to be converted back to velocity. Using $\alpha_t + \sigma_t = 1$, a noise prediction $\hat{\boldsymbol{\epsilon}}_\theta$ gives $\hat{\boldsymbol{u}} = (\hat{\boldsymbol{\epsilon}}_\theta - \boldsymbol{z}_t)/\alpha_t$, while a clean-data prediction $\hat{\boldsymbol{x}}_\theta$ gives $\hat{\boldsymbol{u}} = (\boldsymbol{z}_t - \hat{\boldsymbol{x}}_\theta)/\sigma_t$. More generally, any linearly related target that can be inverted together with $\boldsymbol{z}_t$ can be mapped to the velocity by a closed-form linear map. After this recovery, the different prediction parameterizations use the same velocity-based sampler.

4 ANISOTROPIC FLOW MODELS

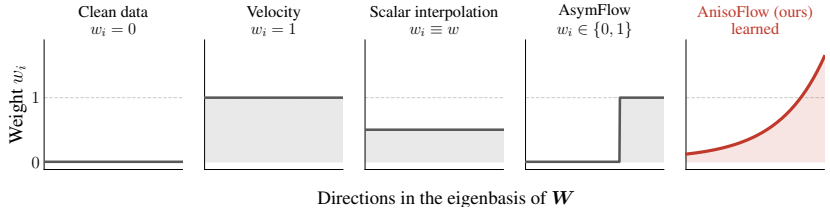
AnisoFlow makes the prediction target learnable: starting from the standard velocity target $\boldsymbol{u} = \boldsymbol{\epsilon} - \boldsymbol{x}$, it scales the noise component by a separate nonnegative weight in each direction of a learned orthogonal basis, while leaving the clean-data component intact. A closed-form map recovers the full velocity from the network's prediction, allowing AnisoFlow to retain the standard network architecture, flow-matching loss, and sampler. We first construct the target and its recovery (Fig. 2), then show that standard targets arise as special cases, and finally describe how the basis is learned.

A learnable prediction target. We propose reweighting only the noise term of the velocity target and leaving its clean-data term intact. Scaling the data term as well would be redundant, since an overall per-direction scale on the target is absorbed by the network, so we fix the data coefficient to -1 and learn only the noise weights (Appendix A). We further ask the reweighting to admit exact velocity recovery at every noise level $t \in (0, 1]$, which holds precisely when the operator has no negative real eigenvalue. Combined with symmetry, this restricts it to positive semidefinite (PSD) operators (Appendix A). Concretely, we define the AnisoFlow target

$$\boldsymbol{u}_W := \boldsymbol{W}\boldsymbol{\epsilon} - \boldsymbol{x}, \tag{3}$$



(a) Prediction and exact recovery



(b) Target family

Figure 2: **AnisoFlow overview. (a)** The network predicts the AnisoFlow target $\boldsymbol{u}_W = \boldsymbol{W}\boldsymbol{\epsilon} - \boldsymbol{x}$, where $\boldsymbol{W}$ is a learned positive semidefinite operator, and a closed-form map recovers the standard velocity. **(b)** In the eigenbasis of $\boldsymbol{W}$, the target is parameterized separately in each direction, retaining the clean-data term $-x_i$ and applying a learned nonnegative weight w_i to the noise term: $u_{W,i} = w_i\epsilon_i - x_i$. Standard targets correspond to fixed weight profiles. AnisoFlow instead learns a nonnegative weight profile across directions.

where $\boldsymbol{W}$ is a PSD operator. Writing the eigendecomposition

$$\boldsymbol{W} = \boldsymbol{B}\,\mathrm{diag}(w_1, \ldots, w_D)\boldsymbol{B}^{\mathrm{T}}, \qquad w_i \geq 0, \tag{4}$$

with orthonormal basis $\boldsymbol{B} = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_D]$, where D is the data dimension, the design space consists of the eigenvalues, which act as per-direction noise weights, and the basis. In coordinates along $\boldsymbol{B}$ the target reads $u_{W,i} = w_i\epsilon_i - x_i$, so the data component remains full-dimensional while the noise component receives a separate nonnegative weight in each direction. In patchified models, $\boldsymbol{W}$ acts on each patch (token) and is shared across spatial locations, so D is the patch dimension, e.g., $D = 16 \times 16 \times 3 = 768$ for JiT (Appendix B.2).

Velocity recovery. Given a prediction $\hat{\boldsymbol{u}}_W$ of the AnisoFlow target, the velocity is recovered as

$$\hat{\boldsymbol{u}} = (\alpha_t\boldsymbol{W} + \sigma_t\boldsymbol{I})^{-1}\big((\boldsymbol{I} - \boldsymbol{W})\boldsymbol{z}_t + \hat{\boldsymbol{u}}_W\big), \qquad \text{equivalently} \qquad \hat{u}_i = \frac{(1 - w_i)z_{t,i} + \hat{u}_{W,i}}{\alpha_t w_i + \sigma_t} \tag{5}$$

coordinate-wise in the eigenbasis of $\boldsymbol{W}$ (Appendix A). The denominator $\alpha_t w_i + \sigma_t$ is positive on $t \in (0, 1]$ whenever $w_i \geq 0$, which is why we restrict to nonnegative weights. This denominator also controls the amplification of target-prediction error: for clean-data prediction ($w_i = 0$), the amplification scales as $1/\sigma_t$, making velocity recovery ill-conditioned at low noise, whereas nonzero weights can improve its conditioning. The weights w_i are shared across timesteps; in practice, allowing separate weights across timestep buckets provides no measurable benefit (Fig. 5).

Training objective. The network parameters θ and the weights $\boldsymbol{w} = (w_1, \ldots, w_D)$ are trained jointly with the recovered-velocity loss $\mathcal{L}(\theta, \boldsymbol{w}) = \mathbb{E}_{t,\boldsymbol{x},\boldsymbol{\epsilon}}\big[\|\hat{\boldsymbol{u}}(\boldsymbol{z}_t, t) - \boldsymbol{u}\|^2\big]$, where $\hat{\boldsymbol{u}}$ is obtained from the network's prediction $\hat{\boldsymbol{u}}_W$ through the recovery map in Eq. (5). The objective is the squared error on the *recovered* velocity, not directly on the AnisoFlow target $\|\hat{\boldsymbol{u}}_W - \boldsymbol{u}_W\|^2$.

Special cases. The family contains several standard targets as special cases. With $w_i = 0$ the target in direction i is clean-data (x_i) prediction up to sign; with $w_i = 1$ it is the velocity $u_i = \epsilon_i - x_i$; in the PCA basis of the data with $w_i \in \{0, 1\}$ it reproduces the binary low-rank mask of AsymFlow (Chen et al., 2026a); with $w_i \equiv w$ constant across directions it becomes an isotropic scalar interpolation, equivalent up to scale and sign to scalar target families such as k-Diff (Jin & Wang, 2026); and the noise (ϵ_i) prediction limit is obtained as $w_i \to \infty$, again up to scale.

Learnable basis. The basis $\boldsymbol{B}$ is learned jointly with the network and the weights, and we consider two optimizers (Table 1). *Stiefel Adam* treats $\boldsymbol{B}$ as a parameter on the Stiefel manifold of

orthonormal matrices and adapts Adam (Kingma & Ba, 2015) to it (Appendix A.4.1). The *spectral update* is a closed-form alternative: assuming the per-direction loss terms share a common eigenbasis, the optimal basis is given by the eigenvectors of a single moment matrix of the network's prediction errors, which we track with an exponential moving average and eigendecompose at every step (Proposition 2, Appendix A.4.2). In both cases the weights w_i remain learned by gradient descent. We use the spectral update by default, as it achieves the best FID in our experiments.

Initialization. By default we initialize $\boldsymbol{B} = \boldsymbol{I}$ and $w_i = 1$, so that $\boldsymbol{W} = \boldsymbol{I}$ and the initial target is ordinary velocity prediction. Optionally, we use a *PCA initialization*: $\boldsymbol{B}$ is set to the PCA basis of the data and $w_i = c\lambda_i$, where λ_i is the data variance along $\boldsymbol{b}_i$ and $c = 0.15$ matches the slope learned by our converged JiT-H/16 models. This choice is motivated by our analysis in §5, which singles out the PCA basis and predicts weights that scale near-linearly with λ_i. PCA initialization is not necessary: it speeds up early training empirically (Table 1), but the learned basis also converges close to the PCA basis on its own without PCA initialization (Fig. 3b).

5 Analysis of the Learned Weights

We analyze how the learned weights relate to the data spectrum and why a single weight profile suffices across timesteps. We study each direction of the PCA basis separately. In this basis the data coordinates are uncorrelated, with variances $\lambda_i = \mathbb{E}[x_i^2]$, while the Gaussian noise remains isotropic, so each weight w_i is associated with a single direction. Full derivations are given in Appendix A.2.

Motivation for the PCA basis. Three observations motivate this choice. First, under a linear-network model, in which the network is replaced by the optimal linear predictor, the PCA basis is a fixed point of the spectral update (Section 4), whose local stability we characterize in Appendix A.5. Second, the update recovers the PCA basis in expectation at initialization, when the network output is zero (Appendix A.4.2). Third, the basis learned in practice closely aligns with PCA (Fig. 3b).

Error amplification and prediction difficulty. Because the velocity recovery map (Eq. 5) is invertible, for every w_i the optimal prediction of the target $u_{W,i}$ recovers the same conditional-mean velocity $\mathbb{E}[u_i \mid \boldsymbol{z}_t, t]$. The choice of w_i therefore does not change the minimum achievable loss, only the function the network must approximate to reach it. In direction i at timestep t, the recovered-velocity loss is

$$\mathcal{E}_i(w_i; t) = \frac{\mathbb{E}\big[(\hat{u}_{W,i} - u_{W,i})^2 \mid t\big]}{(\alpha_t w_i + \sigma_t)^2}. \tag{6}$$

Increasing w_i enlarges the denominator, so errors in the predicted target are amplified less when converted back to velocity. A larger w_i, however, also places more noise in the target, which the network must then predict. The learned weight balances these two effects.

Preferred weight at a fixed timestep. To make this trade-off concrete, we model the network's prediction as combining a noise estimate $\hat{\epsilon}_i$ and a data estimate $\hat{x}_i$, so that $\hat{u}_{W,i} \approx w_i\hat{\epsilon}_i - \hat{x}_i$, mirroring the target $u_{W,i} = w_i\epsilon_i - x_i$. Even an optimal network cannot recover ϵ_i and x_i exactly from $\boldsymbol{z}_t$. The best achievable estimates are the conditional means $\epsilon_i^* = \mathbb{E}[\epsilon_i \mid \boldsymbol{z}_t, t]$ and $x_i^* = \mathbb{E}[x_i \mid \boldsymbol{z}_t, t]$, so we measure the network's errors relative to them, $A_i(t) = \mathbb{E}[(\hat{\epsilon}_i - \epsilon_i^*)^2 \mid t]$ and $C_i(t) = \mathbb{E}[(\hat{x}_i - x_i^*)^2 \mid t]$. The remaining uncertainty is unavoidable and does not depend on w_i. We treat $A_i(t)$ and $C_i(t)$ as locally fixed, assuming that small changes in w_i do not change how accurately the network estimates the noise and the data, and neglect the correlation between the two errors. If each timestep were optimized separately, the preferred weight at timestep t would then be

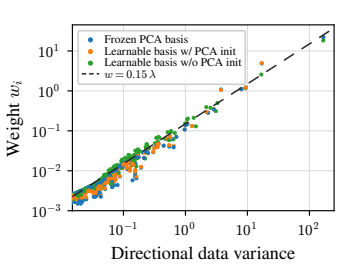
$$w_i^*(t) = \frac{\alpha_t}{\sigma_t}\,\frac{C_i(t)}{A_i(t)}. \tag{7}$$

Thus, directions where data estimation is relatively more difficult favor larger noise weights, while directions where noise estimation is relatively more difficult favor smaller weights.

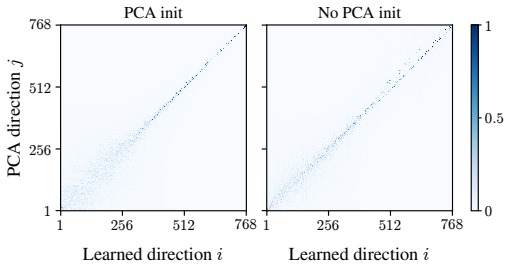
A single weight across timesteps. The explicit factor α_t/σ_t ranges from 0 to ∞, suggesting that different timesteps could prefer very different weights. The two estimation errors, however, vary in the opposite direction. At small t, $\boldsymbol{z}_t$ is dominated by the clean data, making x_i relatively easy to estimate while ϵ_i is difficult to infer. At large t, this relation reverses. A simple sufficient condition for the two effects to cancel is $A_i(t) = \alpha_t q_i(t)\bar{A}_i$ and $C_i(t) = \sigma_t q_i(t)\bar{C}_i$, where $\bar{A}_i$ and $\bar{C}_i$ are direction-dependent scales and $q_i(t) \geq 0$ is a shared time dependence. Every timestep then prefers

Table 1: **Effect of the learnable prediction target**, isolating noise weights/basis optimizers and data-dependent initialization. JiT-H/16 network.

Method	Noise weights	Noise basis	Init	FID↓ @160ep	FID↓ @600ep	Training sec/iter	Inference sec/step
JiT	Frozen (zeros)	–	–	2.68	1.90	0.803	0.179
AsymFlow	Frozen (binary)	Frozen (PCA)	PCA	2.34	1.76	0.803	0.179
AnisoFlow	Adam	Frozen (PCA)	PCA	**2.25**	1.63	0.805	0.180
AnisoFlow	Adam	Stiefel Adam	PCA	2.32	1.68	0.818	0.180
AnisoFlow	Adam	Spectral update	PCA	<u>2.28</u>	1.63	0.819	0.180
AnisoFlow	Adam	Stiefel Adam	–	<u>2.28</u>	<u>1.60</u>	0.818	0.180
AnisoFlow	Adam	Spectral update	–	2.30	**1.55**	0.819	0.180



(a) Learned weights (trimmed range)



(b) Alignment between learned and PCA bases $|\boldsymbol{v}_j^{\mathrm{T}}\boldsymbol{b}_i|$

Figure 3: **Emergent spectral structure in AnisoFlow. (a)** Learned weights versus directional data variance $\boldsymbol{b}_i^{\mathrm{T}}\boldsymbol{\Sigma}\boldsymbol{b}_i$, where $\boldsymbol{\Sigma}$ is the data covariance, for a frozen PCA basis and learnable bases with and without PCA initialization. The displayed range is trimmed to variances ≥ 0.015. **(b)** Absolute inner products $|\boldsymbol{v}_j^{\mathrm{T}}\boldsymbol{b}_i|$ between PCA directions $\boldsymbol{v}_j$ and learned directions $\boldsymbol{b}_i$, in their native spectral order. Both initializations yield near-diagonal alignment, strongest at high-variance directions. All models are JiT-H/16; both learnable variants use the spectral update in Table 1.

the same weight $w_i^* = \bar{C}_i/\bar{A}_i$, which therefore also minimizes the loss averaged over timesteps that training optimizes. Consistent with this analysis, weights learned independently across timestep buckets nearly coincide, and allowing the weights to vary with time does not improve FID (Fig. 5).

Near-linear spectral scaling. The shared weight $w_i^* = \bar{C}_i/\bar{A}_i$ is the ratio of the data- and noise-estimation error scales in direction i. Two further assumptions connect it to the data variance. Because the noise is isotropic, we take the noise-estimation error to vary weakly across directions, $\bar{A}_i \approx \bar{A}$. We further assume the network estimates each data coordinate with comparable relative error, so that the data-estimation error scales with the data energy, $\bar{C}_i \propto \lambda_i$. Together, these give $w_i^* = \bar{C}_i/\bar{A}_i \propto \lambda_i$, explaining the near-linear scaling $w_i \approx c\lambda_i$ observed in pixel space (Fig. 3a).

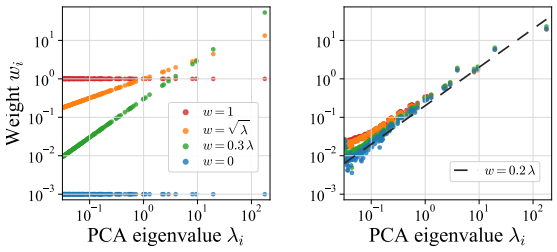
6 Experiments

Experimental setup. We evaluate AnisoFlow on class-conditional ImageNet (Deng et al., 2009) 256×256 and text-to-image generation in pixel-space, and then further extend it to latent-space flow models. In each setting, we preserve the architecture and training recipe of the underlying model and change only the prediction-target parameterization. For our main pixel-space experiments, we use JiT-H/16 (Li & He, 2026), with JiT-B/16 for smaller-scale experiments. We report FID (Heusel et al., 2017) under the ADM evaluation protocol (Dhariwal & Nichol, 2021). Full implementation and evaluation details are given in Appendix B.

6.1 Pixel-Space ImageNet Generation

6.1.1 Main results

Learning the target, weights, and basis. Table 1 isolates the effects of learning the weights, learning the basis, and initialization on a fixed JiT-H/16 backbone. For the basis, we compare the



(a) Initialization (b) Learned weights

Figure 4: **Robustness to weight initialization. (a)** Four initial weight profiles and **(b)** the learned weights after training JiT-B/16 converge to nearly the same near-linear profile.

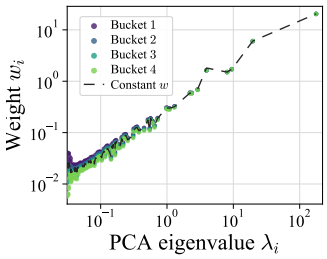


Figure 5: **Time-bucketed weight profiles.** Four timestep-bucket profiles (JiT-B/16) nearly coincide.

proposed Stiefel Adam and spectral update optimizers with a frozen PCA basis. Learning continuous directional weights at a frozen PCA basis improves considerably over both clean-data prediction (JiT, FID 1.90) and AsymFlow's frozen binary mask (FID 1.76), reaching 1.63 FID, showing that much of the gain comes from learning the directional weights themselves. The frozen PCA basis also gives the best FID early in training (2.25). On the other hand, learning the basis without PCA initialization (isotropic $\boldsymbol{W} = \boldsymbol{I}$ initialization) eventually improves the final quality, with spectral update obtaining the best final FID of 1.55. Despite different initialization and optimization paths, the learned basis remains close to PCA and the weights recover a PCA spectral profile, which we examine in §6.1.2.

Comparison with prior pixel diffusion models. In Table 2, we compare prior pixel diffusion models on ImageNet 256×256 generation, grouped by architecture family: U-Net CNNs (Dhariwal & Nichol, 2021), U-ViT-like hierarchical transformers (Jabri et al., 2023; Hoogeboom et al., 2023; Kingma & Gao, 2023; Hoogeboom et al., 2025; Guo et al., 2026; Lei et al., 2026), DDT-like (Wang et al., 2026d) hierarchical transformers (Wang et al., 2026c; Chen et al., 2026b; Ma et al., 2026b; Yu et al., 2026; Ma et al., 2026a; He et al., 2026), and plain DiT-like transformers (Chen et al., 2025; Li & He, 2026; Ma et al., 2026c; Shin et al., 2026; Chen et al., 2026a). AnisoFlow-H/16 achieves the best in-class FID of 1.55 among plain DiT-like pixel-space models, with only flow-matching training and no representation alignment (REPA) (Yu et al., 2025) or extra pre-training. It also outperforms pixel models with complex hierarchical architectures and REPA (FID 1.56*).

Compute overhead. Compared to JiT (Table 1), AnisoFlow adds only 0.6% inference overhead and at most 2.0% training overhead learning the weights and basis; Appendix B.5 provides a detailed breakdown.

Table 2: **ImageNet** 256×256 **pixel diffusion.** Best reported FID, grouped by architecture family. FLOP estimation follows Yu et al. (2026), and * marks the JiT evaluation protocol, which can give up to 0.08 lower FID than the ADM protocol. †: uses a pretrained representation encoder (e.g., REPA-style alignment or DINO-based losses) or self-supervised pre-training.

Method	Target	Params	GFLOPs	FID↓
Hierarchical CNNs (U-Net-like)				
ADM-G	ϵ	554M	2240	4.59
Hierarchical transformers (U-ViT-like)				
RIN	ϵ	320M	668	3.42
SiD, UViT/2	ϵ	2B	1110	2.44
VDM++, UViT/2	ϵ	2B	1110	2.12
SiD2, UViT/2	ϵ	–	274	1.73
PixelU-H/16†	x	1.2B	274	1.63*
EPG-G/16†	x	1.4B	642	1.58
SiD2, UViT/1	ϵ	–	1306	**1.38**
Hierarchical transformers (DDT-like)				
PixNerd-XL/16†	$\epsilon - x$	700M	268	2.15
DiP-XL/16†	$\epsilon - x$	631M	–	1.79
DeCo-XL/16†	$\epsilon - x$	682M	245	1.62
PixelDiT-XL/16†	$\epsilon - x$	797M	311	1.61
FrequencyBooster-H/16†	x	1.1B	–	1.60*
HyperDiT-H†	$\epsilon - x$	952M	458	**1.56***
Plain transformers (DiT-like)				
PixelFlow-XL/4	$\epsilon - x$	677M	5818	1.98
JiT-H/16	x	953M	364	1.86*
PixelGen-XL/16†	x	676M	260	1.83
JiT-G/16	x	2B	766	1.82*
PixelREPA-H/16†	x	953M	364	1.81*
AsymFlow-H/16†	$P\epsilon - x$	953M	364	1.57
AnisoFlow-H/16 (ours)	$W\epsilon - x$	953M	365	**1.55**

6.1.2 What does AnisoFlow learn?

Learned weights follow the data spectrum. As shown in Fig. 3a, under different initialization and basis optimization paths, the learned directional weights of the pixel-space model always follow a near-linear relationship with the data variance along their corresponding learned direc-



(a) Qualitative comparison on text-to-image generation

(b) Comparison on 8-bit inference

Figure 6: **Qualitative text-to-image results.** **(a)** 512×512 generations from AnisoFlow, the JiT-L/16-T2I baseline ($\boldsymbol{x}$-prediction), and the latent i1-1B model ($\boldsymbol{u}$-prediction), using identical prompts and sampling seeds. AnisoFlow renders finer texture than the $\boldsymbol{x}$-prediction baseline (paper, fabric, waves, skin) and a sharper, more distinct style than the latent model, in line with its lower FID and pFID. **(b)** The same prompt and seed under 8-bit inference (INT8, FP8, and blockwise FP8). $\boldsymbol{x}$-prediction develops patch seams or strong noise, while AnisoFlow remains stable.

tions. Meanwhile, the basis learned from an isotropic initialization aligns closely with the PCA basis (Fig. 3b). The directional variances therefore closely track the PCA spectrum λ_i, recovering the near-linear scaling $w_i \approx c\lambda_i$ predicted by our analysis.

Effect of initialization. Initialization primarily affects the optimization trajectory rather than the final spectral structure. On JiT-H/16, initializing at $\boldsymbol{W} = \boldsymbol{I}$ gives worse FID early in training but the best FID after 600 epochs (Table 1), while diverse initial weight profiles on JiT-B/16 converge to similar near-linear solutions (Fig. 4).

A single weight profile suffices across timesteps. To test whether the weights need to vary with time, we split the timestep range at the quartiles of the training timestep distribution and learn an independent profile $w_i^{(b)}$ for each of the four buckets. The profiles remain close across buckets and to the constant-w_i solution (Fig. 5), while the added flexibility yields no FID improvement (4.92 for shared w_i vs. 4.96 for time-dependent $w_i^{(b)}$ after 160 epochs). This supports using a single shared weight profile, consistent with the cancellation analysis in §5.

6.2 Pixel-Space Text-to-Image Generation

We further apply AnisoFlow to pixel-space text-to-image generation using JiT-L/16-T2I as the baseline, which adopts the MM-JiT denoiser architecture from MiniT2I (Wang et al., 2026e), trained with the text encoder, data, and training schedule of i1 (Zeng et al., 2026). Appendix B.3 gives the training and evaluation details. We compare fixed $\boldsymbol{x}$-prediction with the learned AnisoFlow target (with PCA initialization and spectral update) under matched training and evaluation. We evaluate at 512×512 on OneIG-Bench (Chang et al., 2025), LongText-Bench (Geng et al., 2025), GenEval (Ghosh et al., 2023), DPG-Bench (Hu et al., 2024), HPSv3 (Ma et al., 2025), and COCO-10k (Lin et al., 2014; Chen et al., 2026a) (FID and patch FID (pFID) (Lin et al., 2024)).

Main results. AnisoFlow improves over the $\boldsymbol{x}$-prediction baseline on every metric except GenEval (Table 3). The largest gains are in HPSv3 ($7.71\rightarrow8.03$), COCO-10k FID ($22.0\rightarrow20.1$), and pFID ($24.4\rightarrow22.7$), indicating a closer match to the style and fine detail of real images. AnisoFlow also surpasses the latent i1-1B model, which shares the text encoder and an equivalent training schedule

but uses a larger denoiser, on the detail- and style-sensitive metrics LongText-Bench, HPSv3, FID, and pFID. Fig. 6(a) shows this qualitatively: AnisoFlow renders finer texture than the baseline and a sharper, more distinct style than the latent model, while achieving accurate text rendering.

Reduced-precision stability. Reduced-precision inference is important for the efficiency of large generative models, but it makes numerical conditioning critical. For $\boldsymbol{x}$-prediction, velocity recovery divides the prediction error by σ_t, amplifying quantization error at low noise levels into visible artifacts. AnisoFlow instead learns the prediction target that minimizes the recovered-velocity error directly, so it naturally converges to a well-conditioned target (§5). Fig. 6(b) compares INT8, FP8, and blockwise-FP8 (DeepSeek-AI et al., 2025) inference: the baseline develops patch seams or strong noise, whereas AnisoFlow remains stable throughout.

Table 3: **Text-to-image** 512×512**.** All models are trained using i1's data, text encoder, and schedule, and tested using the same guidance and sampler settings. * indicates using i1's rewritten prompts. Bold marks the best result within the matched pixel-space group.

Method	Params	OneIG↑	Long Text*↑	Gen Eval*↑	DPG*↑	HPSv3↑	COCO-10k	
							FID↓	pFID↓
i1-1B (FLUX.2 VAE, $\boldsymbol{u}$-pred.)	1.3B	0.511	0.853	0.826	86.3	7.11	62.3	62.4
JiT-L/16-T2I ($\boldsymbol{x}$-pred.)	917M	0.512	0.862	**0.820**	85.7	7.71	22.0	24.4
AnisoFlow-L/16-T2I (ours)	918M	**0.516**	**0.864**	0.814	**85.8**	**8.03**	**20.1**	**22.7**

6.3 Latent-Space Models

We also evaluate AnisoFlow on SiT-XL/2-REPA (Ma et al., 2024; Yu et al., 2025) and RAEv2 (Singh et al., 2026), preserving each model's architecture and training/evaluation recipe while changing only the prediction target, except that for RAEv2 both the baseline and AnisoFlow use a modified timestep distribution and inference clamp that improve the baseline itself (Appendix B). AnisoFlow performs comparably to the default targets (Table 4), while the learned weights remain close to velocity prediction for SiT-XL/2-REPA and to data prediction for RAEv2 (Fig. 7). The default targets thus appear well matched to these representations. Unlike in pixel space, the latent weight profiles are much flatter, so the assumptions behind the near-linear scaling do not carry over directly.

Table 4: **Latent-space results.** AnisoFlow achieves comparable performance in latent-space models, where the learned targets remain close to the respective default targets. *FID computed using JiT's reference statistics.

Method	Target	FID↓	JiT FID*↓
SiT-XL/2-REPA	velocity	1.44	1.41
Aniso-SiT-XL/2-REPA (ours)	learned	1.44	1.40
RAEv2	$\boldsymbol{x}$-pred.	1.05	0.98
Aniso-RAEv2 (ours)	learned	1.05	0.96

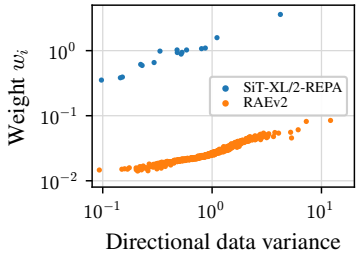


Figure 7: **Latent-space weights.** Learned weights in SiT-REPA and RAEv2 are close to their default targets.

7 Conclusion

AnisoFlow replaces fixed prediction targets with a learnable anisotropic target, assigning a separate noise weight to each direction in a learned basis. In pixel space, the learned basis aligns with PCA and the weights scale near-linearly with data variance, consistent with our prediction-error analysis. Empirically, AnisoFlow substantially improves FID over the JiT baseline on ImageNet without representation alignment or architectural changes, while improving image quality and numerical stability in text-to-image generation. These results show that prediction targets can adapt to the data and the model's errors rather than being fixed in advance.

Limitations. Our analysis uses a local model of prediction error and does not capture full training dynamics or determine the constant in the observed spectral scaling. Empirical gains are largest in pixel space and smaller in latent space, where learned targets remain close to existing defaults.

Acknowledgments. This work was done while Rayhan Zirvi was a visiting undergraduate researcher at Stanford University, supported by the Summer Undergraduate Research Fellowships (SURF) program at Caltech.

AI USE STATEMENT

In this work, we used generative AI tools (LLM-based assistants) to edit the paper for grammar and readability, to help verify and refine the derivations in Section 5 and Appendix A, and to accelerate the implementation of our experiments. All AI-assisted work was reviewed and verified by the authors, and we take responsibility for the final content of this paper.

REPRODUCIBILITY STATEMENT

Section 4 and Appendix B.1 describe how the directional weights and basis are parameterized and optimized. Appendices B.2–B.4 give the training and evaluation settings for the ImageNet, text-to-image, and latent-space experiments, including the selected guidance settings (Table 5) and the text-to-image training stages (Table 6), and Appendix B.5 details the compute overhead. For the analysis, Appendix A states the assumptions and gives full derivations, including the proof of Proposition 2.

REFERENCES

Shadab Ahamed, Eshed Gal, Md Shahriar Rahim Siddiqui, Simon Ghyselincks, Moshe Eliasof, and Eldad Haber. Preconditioned flow matching. *arXiv preprint arXiv:2603.02337*, 2026.

Michael Samuel Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *ICLR*, 2023.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L. Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. QuaRot: Outlier-free 4-bit inference in rotated LLMs. In *NeurIPS*, 2024.

Black Forest Labs. Flux.2: Frontier visual intelligence. `https://bfl.ai/blog/flux-2`, 2025.

Ollin Boer Bohan. Megalith-10M. `https://huggingface.co/datasets/madebyollin/megalith-10m`, 2024.

Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, et al. Video generation models as world simulators. `https://openai.com/research/video-generation-models-as-world-simulators`, 2024.

Jingjing Chang, Yixiao Fang, Peng Xing, Shuhan Wu, Wei Cheng, Rui Wang, Xianfang Zeng, Gang Yu, and Hai-Bao Chen. OneIG-Bench: Omni-dimensional nuanced evaluation for image generation. In *NeurIPS Datasets and Benchmarks*, 2025.

Hansheng Chen, Jan Ackermann, Minseo Kim, Gordon Wetzstein, and Leonidas Guibas. Asymmetric flow models. In *NeurIPS*, 2026a.

Shoufa Chen, Chongjian Ge, Shilong Zhang, Peize Sun, and Ping Luo. Pixelflow: Pixel-space generative models with flow. *arXiv preprint arXiv:2504.07963*, 2025.

Zhennan Chen, Junwei Zhu, Xu Chen, Jiangning Zhang, Xiaobin Hu, Hanzhen Zhao, Chengjie Wang, Jian Yang, and Ying Tai. Dip: Taming diffusion models in pixel space. In *CVPR*, 2026b.

Qinpeng Cui, Xinyi Zhang, Qiqi Bao, and Qingmin Liao. Elucidating the solution space of extended reverse-time SDE for diffusion models. In *WACV*, 2025.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2025. URL `https://arxiv.org/abs/2412.19437`.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pp. 248–255, 2009.

Karan Desai, Gaurav Kaul, Zubin Aysola, and Justin Johnson. RedCaps: Web-curated image-text data created by the people, for the people. In *NeurIPS Datasets and Benchmarks*, 2021.

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

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *ICML*, 2024.

Funing Fu, Tenghui Wang, Guanyu Zhou, Junyong Cen, and Qichao Zhu. Jlt: Clean-latent prediction in latent diffusion transformers. *arXiv preprint arXiv:2605.27102*, 2026.

Anne Gagneux, Ségolène Martin, Rémi Gribonval, and Mathurin Massias. Training flow matching: The role of weighting and parameterization. In *ICLR Workshop*, 2026.

Zigang Geng, Yibing Wang, Yeyao Ma, Chen Li, Yongming Rao, et al. X-omni: Reinforcement learning makes discrete autoregressive image generative models great again. *arXiv preprint arXiv:2507.22058*, 2025.

Dhruba Ghosh, Hannaneh Hajishirzi, and Ludwig Schmidt. Geneval: An object-focused framework for evaluating text-to-image alignment. In *NeurIPS*, 2023.

Zipeng Guo, Lichen Ma, Yu He, Xiaolong Fu, Jingling Fu, Junshi Huang, and Yan Li. PixelU: A U-shaped transformer for efficient end-to-end pixel diffusion. In *ECCV*, 2026.

Yoav HaCohen, Nisan Chiprut, Benny Brazowski, Daniel Shalem, Dudu Moshe, Eitan Richardson, Eran Levin, Guy Shiran, Nir Zabari, Ori Gordon, et al. Ltx-video: Realtime video latent diffusion. *arXiv preprint arXiv:2501.00103*, 2024.

Yu He, Lichen Ma, Zipeng Guo, Xinyuan Shan, Jingling Fu, Dong Chen, Junshi Huang, and Yan Li. HyperDiT: Hyper-connected transformers for high-fidelity pixel-space diffusion. *arXiv preprint arXiv:2605.15741*, 2026.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS*, 2017.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS Workshop*, 2021.

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

Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. Simple diffusion: End-to-end diffusion for high resolution images. In *ICML*, pp. 13213–13232, 2023.

Emiel Hoogeboom, Thomas Mensink, Jonathan Heek, Kay Lamerigts, Ruiqi Gao, and Tim Salimans. Simpler diffusion (sid2): 1.5 fid on imagenet512 with pixel-space diffusion. In *CVPR*, 2025.

Xiwei Hu, Rui Wang, Yixiao Fang, Bin Fu, Pei Cheng, and Gang Yu. Ella: Equip diffusion models with llm for enhanced semantic alignment. *arXiv preprint arXiv:2403.05135*, 2024.

Allan Jabri, David Fleet, and Ting Chen. Scalable adaptive computation for iterative generation. In *ICML*, 2023.

Qing Jin and Chaoyang Wang. Revisiting diffusion model predictions through dimensionality. *arXiv preprint arXiv:2601.21419*, 2026.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks. `https://kellerjordan.github.io/posts/muon/`, 2024.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *NeurIPS*, 2022.

Tero Karras, Miika Aittala, Jaakko Lehtinen, Janne Hellsten, Timo Aila, and Samuli Laine. Analyzing and improving the training dynamics of diffusion models. In *CVPR*, 2024.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Diederik P Kingma and Ruiqi Gao. Understanding diffusion objectives as the elbo with simple data augmentation. In *NeurIPS*, 2023.

Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, et al. Hunyuanvideo: A systematic framework for large video generative models. *arXiv preprint arXiv:2412.03603*, 2025.

Jiachen Lei, Keli Liu, Julius Berner, Haiming Yu, Hongkai Zheng, Jiahong Wu, and Xiangxiang Chu. There is no vae: End-to-end pixel-space generative modeling via self-supervised pre-training. In *ICLR*, 2026.

Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise. In *CVPR*, 2026.

Shanchuan Lin, Anran Wang, and Xiao Yang. Sdxl-lightning: Progressive adversarial diffusion distillation. *arXiv preprint arXiv:2402.13929*, 2024.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, pp. 740–755, 2014.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *ICLR*, 2023.

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

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. SpinQuant: LLM quantization with learned rotations. In *ICLR*, 2025.

Lichen Ma, Zipeng Guo, Yu He, Xiaolong Fu, Luohang Liu, Jingling Fu, Junshi Huang, and Yan Li. Frequencybooster: Full-frequency modeling for high-fidelity pixel diffusion. In *NeurIPS*, 2026a.

Nanye Ma, Mark Goldstein, Michael S. Albergo, Nicholas M. Boffi, Eric Vanden-Eijnden, and Saining Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *ECCV*, 2024.

Yuhang Ma, Xiaoshi Wu, Keqiang Sun, and Hongsheng Li. Hpsv3: Towards wide-spectrum human preference score. In *ICCV*, 2025.

Zehong Ma, Longhui Wei, Shuai Wang, Shiliang Zhang, and Qi Tian. Deco: Frequency-decoupled pixel diffusion for end-to-end image generation. In *CVPR*, 2026b.

Zehong Ma, Ruihan Xu, and Shiliang Zhang. Pixelgen: Improving pixel diffusion with perceptual supervision. *arXiv preprint arXiv:2602.02493*, 2026c.

Björn Ottosson. A perceptual color space for image processing. `https://bottosson.github.io/posts/oklab/`, 2020.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, 2023.

Pexels. Pexels. `https://www.pexels.com`, 2025.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

Seyedmorteza Sadat, Otmar Hilliges, and Romann M. Weber. Eliminating oversaturation and artifacts of high guidance scales in diffusion models. In *ICLR*, 2025.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *ICLR*, 2022.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade W Gordon, Ross Wightman, Mehdi Cherti, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. In *NeurIPS Datasets and Benchmarks*, 2022.

Jaeyo Shin, Jiwook Kim, and Hyunjung Shim. Representation alignment for just image transformers is not easier than you think. In *ECCV*, 2026.

Jaskirat Singh, Boyang Zheng, Zongze Wu, Richard Zhang, Eli Shechtman, and Saining Xie. Improved baselines with representation autoencoders. *arXiv preprint arXiv:2605.18324*, 2026.

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.

Bart Thomee, David A. Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. YFCC100M: The new data in multimedia research. *Communications of the ACM*, 59(2), 2016.

Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The iNaturalist species classification and detection dataset. In *CVPR*, 2018.

Alex Jinpeng Wang, Dongxing Mao, Jiawei Zhang, Weiming Han, Zhuobai Dong, Linjie Li, Yiqi Lin, Zhengyuan Yang, Libo Qin, Fuwei Zhang, Lijuan Wang, and Min Li. TextAtlas5M: A large-scale dataset for dense text image generation. In *ICML*, 2026a.

Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Binxu Wang and Cengiz Pehlevan. An analytical theory of spectral bias in the learning dynamics of diffusion models. In *NeurIPS*, 2025.

Binxu Wang, Jacob Zavatone-Veth, and Cengiz Pehlevan. A random matrix theory perspective on the consistency of diffusion models. In *ICML*, 2026b.

Shuai Wang, Ziteng Gao, Chenhui Zhu, Weilin Huang, and Limin Wang. Pixnerd: Pixel neural field diffusion. In *ICLR*, 2026c.

Shuai Wang, Zhi Tian, Weilin Huang, and Limin Wang. Ddt: Decoupled diffusion transformer. In *CVPR*, 2026d.

Xianbang Wang, Hanhong Zhao, Yiyang Lu, Kangyang Zhou, Linrui Ma, and Kaiming He. MiniT2I: A minimalist baseline for text-to-image generation. `https://peppaking8.github.io/#/post/minit2i`, 2026e.

Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. In *ICLR*, 2025.

Yongsheng Yu, Wei Xiong, Weili Nie, Yichen Sheng, Shiqiu Liu, and Jiebo Luo. Pixeldit: Pixel diffusion transformers for image generation. In *CVPR*, 2026.

Z-Image Team, Huanqia Cai, Sihan Cao, Ruoyi Du, Peng Gao, Steven Hoi, et al. Z-image: An efficient image generation foundation model with single-stream diffusion transformer. *arXiv preprint arXiv:2511.22699*, 2025.

Boya Zeng, Tianze Luo, Shu Pu, Jucheng Shen, Taiming Lu, Gabriel Sarch, and Zhuang Liu. i1: A simple and fully open recipe for strong text-to-image models. *arXiv preprint arXiv:2606.11289*, 2026.

Biao Zhang, Fedor Moiseev, Joshua Ainslie, Paul Suganthan, Min Ma, Surya Bhupatiraju, Fede Lebron, Orhan Firat, Armand Joulin, and Zhe Dong. Encoder-decoder Gemma: Improving the quality-efficiency trade-off via adaptation. *arXiv preprint arXiv:2504.06225*, 2025.

Jack Zhang, Noah Amsel, Berlin Chen, and Tri Dao. Gram Newton-Schulz. `https://dao-ailab.github.io/blog/2026/gram-newton-schulz/`, 2026a.

Le Zhang, Ning Mang, and Aishwarya Agrawal. Rit: Vanilla diffusion transformers suffice in representation space. *arXiv preprint arXiv:2605.21981*, 2026b.

Wenliang Zhao, Lujia Bai, Yongming Rao, Jie Zhou, and Jiwen Lu. Unipc: A unified predictor-corrector framework for fast sampling of diffusion models. In *NeurIPS*, 2023.

Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE TPAMI*, 2017.

A Theoretical Details

A.1 Target Family and Recovery

Normalizing the linear target family. We first show that a general linear target has only one meaningful relative degree of freedom per direction, since any nonzero overall scaling can be absorbed into the network output. In a single direction, such a target combines the clean-data and noise coordinates, $y = \varphi x + \psi\epsilon$. Rescaling the coefficients $(\varphi, \psi) \mapsto (c\varphi, c\psi)$ with $c \neq 0$ rescales the target by c. Any prediction $f(z_t, t)$ for the original target can therefore be converted into a prediction $cf(z_t, t)$ for the rescaled target by multiplying the network output by c, so this rescaling does not change the set of functions the model can represent. Under squared error, the optimal prediction transforms in the same way, $\mathbb{E}[c\,y \mid z_t, t] = c\,\mathbb{E}[y \mid z_t, t]$. We therefore fix the overall scale by setting $\varphi = -1$, leaving the single relative noise weight $w = -\psi/\varphi$ and the AnisoFlow parameterization $u_{W,i} = w_i\epsilon_i - x_i$. The remaining case $\varphi = 0$, pure noise prediction, is recovered as $w_i \to \infty$ up to the overall scale already removed. Scaling the data component in addition to the noise component thus introduces no new target family, though different scalings may still affect gradients, optimization, or regularization in practice.

Extension to mixed directions. The same argument applies to linear targets that mix directions. Given a predictor $f(\boldsymbol{z}_t, t)$ for a target $\boldsymbol{y}$, composing its output with an invertible linear transformation $\boldsymbol{T}$ gives a predictor for the transformed target $\boldsymbol{Ty}$, and the optimal predictor transforms in the same way, $\mathbb{E}[\boldsymbol{Ty} \mid \boldsymbol{z}_t, t] = \boldsymbol{T}\,\mathbb{E}[\boldsymbol{y} \mid \boldsymbol{z}_t, t]$. Applying $\boldsymbol{T} = -\boldsymbol{K}^{-1}$ to any linear target $\boldsymbol{y} = \boldsymbol{Kx} + \boldsymbol{L\epsilon}$ with invertible $\boldsymbol{K}$ gives the normal form $\boldsymbol{u}_W = \boldsymbol{W\epsilon} - \boldsymbol{x}$ with $\boldsymbol{W} = -\boldsymbol{K}^{-1}\boldsymbol{L}$. We use $\boldsymbol{W}$ to parameterize the linear transformation applied to the noise relative to the fixed data coefficient $-\boldsymbol{I}$. Targets with singular $\boldsymbol{K}$, such as pure noise prediction, fall outside this normal form, as in the scalar case $\varphi = 0$. As in the scalar case, the normalization preserves the set of representable predictors up to an invertible output transformation, but it may change the geometry of the training loss and hence optimization. The resulting $\boldsymbol{W}$ need not be symmetric; we next derive the recovery for general $\boldsymbol{W}$ and then characterize which choices are admissible.

Velocity recovery. We now assume the linear rectified-flow schedule $\alpha_t = 1 - t$ and $\sigma_t = t$, so that $\alpha_t + \sigma_t = 1$ and the flow-matching velocity is $u = \epsilon - x$. We first derive the recovery formula in a single direction with scalar weight w, and then extend it to matrix form. The interpolation and target are

$$z_t = \alpha_t x + \sigma_t \epsilon, \qquad u_W = w\epsilon - x. \tag{8}$$

Using $\alpha_t + \sigma_t = 1$ and collecting the data and noise coefficients,

$$(1 - w)z_t + u_W = -\big(\alpha_t w + \sigma_t\big)x + \big(\alpha_t w + \sigma_t\big)\epsilon = \big(\alpha_t w + \sigma_t\big)(\epsilon - x). \tag{9}$$

Since $u = \epsilon - x$, this gives $u = \big((1 - w)z_t + u_W\big)/(\alpha_t w + \sigma_t)$. Replacing the exact target by the network prediction yields the per-direction recovery formula of Eq. (5). In matrix form, using $\boldsymbol{z}_t = \alpha_t\boldsymbol{x} + \sigma_t\boldsymbol{\epsilon}$ and $\boldsymbol{u}_W = \boldsymbol{W\epsilon} - \boldsymbol{x}$, the same calculation gives

$$(\boldsymbol{I} - \boldsymbol{W})\boldsymbol{z}_t + \boldsymbol{u}_W = (\alpha_t\boldsymbol{W} + \sigma_t\boldsymbol{I})(\boldsymbol{\epsilon} - \boldsymbol{x}) = (\alpha_t\boldsymbol{W} + \sigma_t\boldsymbol{I})\,\boldsymbol{u}, \tag{10}$$

so for a predicted target $\hat{\boldsymbol{u}}_W$ inverting $\alpha_t\boldsymbol{W} + \sigma_t\boldsymbol{I}$ yields the matrix recovery of Eq. (5), valid whenever this operator is invertible, with no symmetry required. Applying this linear map before computing the loss lets us retain the usual velocity-space flow-matching objective and sampler, without introducing additional network modules. The recovery formula also determines which choices of $\boldsymbol{W}$ keep the recovery nonsingular across all timesteps.

Why we restrict $\boldsymbol{W}$ to be positive semidefinite. Recovery fails at time t exactly when $-t/(1 - t)$ is an eigenvalue of $\boldsymbol{W}$, and this quantity sweeps the entire negative real axis as t ranges over $(0, 1)$. Recovery is therefore nonsingular at every interior timestep exactly when $\boldsymbol{W}$ has no eigenvalue on the negative real axis; for example, $\boldsymbol{W} = -\boldsymbol{I}$ becomes singular at $t = 1/2$. This fully general family is algebraically possible but not a practical target family. A general real operator may have complex eigenvalues and, when non-normal, may be poorly conditioned even if its eigenvalues avoid the negative real axis. It also does not admit the simple orthogonal, direction-wise recovery used by AnisoFlow. Restricting to symmetric operators makes all eigenvalues real, and the nonsingularity condition then reduces to requiring nonnegative eigenvalues, i.e., $\boldsymbol{W} \succeq 0$. A general PSD operator is $\boldsymbol{W} = \boldsymbol{B}\,\mathrm{diag}(w_i)\boldsymbol{B}^{\mathrm{T}}$ with $w_i \geq 0$ for an orthonormal basis $\boldsymbol{B}$, the parameterization of §4, so we need only choose nonnegative eigenvalues and an orthonormal basis. In this basis the recovery

acts direction by direction, with the per-direction denominators $\alpha_t w_i + \sigma_t$ of the scalar form above. These denominators are positive on $t \in (0, 1]$, and a zero weight is singular only at the data endpoint $t = 0$, which we treat as a limiting value. Our samplers never evaluate $t = 0$, and the numerical safeguards on the recovery denominators are given in Appendix B.

A.2 Analysis of the Learned Weights

Throughout this subsection the basis is fixed to the PCA directions, which closely match the learned basis (Fig. 3b). Appendix A.5 studies when this alignment is preserved and locally stable under the spectral update. Here we study the weights w_i. As in the main text, w_i denotes the single weight used by AnisoFlow in direction i, shared across timesteps, while $w_i^*(t)$ denotes the weight preferred at a fixed timestep t if that timestep were optimized in isolation.

How prediction error enters the loss. Let $u_i^*(\boldsymbol{z}_t, t) = \mathbb{E}[u_i \mid \boldsymbol{z}_t, t]$ denote the conditional-mean velocity. For any w_i, invertibility of the recovery map gives the corresponding optimal target predictor

$$u_{W,i}^*(w_i; \boldsymbol{z}_t, t) = (\alpha_t w_i + \sigma_t) u_i^*(\boldsymbol{z}_t, t) - (1 - w_i) z_{t,i}, \tag{11}$$

which is exactly $\mathbb{E}[u_{W,i} \mid \boldsymbol{z}_t, t]$. After recovery, this predictor yields u_i^*, so the minimum achievable velocity-prediction loss is the same for every w_i. Let $e_{W,i}(w_i) = \hat{u}_{W,i} - u_{W,i}$ denote the network's error relative to the target $u_{W,i} = w_i \epsilon_i - x_i$. The corresponding recovered-velocity error is $e_{W,i}(w_i)/(\alpha_t w_i + \sigma_t)$, so the recovered-velocity loss in direction i is exactly Eq. (6). Increasing w_i enlarges the recovery denominator and therefore reduces the amplification of target-prediction error, but it also changes the function the network must approximate. The learned weight balances these two effects. The loss in Eq. (6) decomposes as

$$\mathcal{E}_i(w_i; t) = \underbrace{\frac{\mathbb{E}[(\hat{u}_{W,i} - u_{W,i}^*)^2 \mid t]}{(\alpha_t w_i + \sigma_t)^2}}_{\text{excess prediction error}} + \underbrace{\mathbb{E}[(u_i - u_i^*)^2 \mid t]}_{\text{irreducible error}}. \tag{12}$$

By orthogonality of the conditional mean, the second term is the minimum achievable velocity-prediction loss and is independent of w_i. Thus, only the excess prediction error can affect which weight is preferred.

A local frozen-error model. To understand how the loss varies with w_i near a trained solution, we use a local frozen-error model. Let $\epsilon_i^* = \mathbb{E}[\epsilon_i \mid \boldsymbol{z}_t, t]$ and $x_i^* = \mathbb{E}[x_i \mid \boldsymbol{z}_t, t]$, so that the optimal target predictor is $u_{W,i}^* = w_i \epsilon_i^* - x_i^*$. Around the current learned weight, we model the network's prediction as if it were formed from a noise estimate $\hat{\epsilon}_i$ and a data estimate $\hat{x}_i$, according to $\hat{u}_{W,i}(w_i) \approx w_i \hat{\epsilon}_i - \hat{x}_i$, and measure their errors relative to the conditional means, $e_{n,i} = \hat{\epsilon}_i - \epsilon_i^*$ and $e_{x,i} = \hat{x}_i - x_i^*$. The excess target error is then $\hat{u}_{W,i} - u_{W,i}^* \approx w_i e_{n,i} - e_{x,i}$. Measuring errors relative to the conditional means separates out the irreducible uncertainty, whose data and noise components are perfectly anti-correlated given $z_{t,i}$, since $\alpha_t(x_i - x_i^*) + \sigma_t(\epsilon_i - \epsilon_i^*) = 0$, and which contributes only the w_i-independent term in Eq. (12). We assume that, over the range of w_i relevant near convergence, the conditional second-order statistics of $e_{n,i}$ and $e_{x,i}$ vary weakly with w_i, and therefore treat them as fixed when analyzing the dependence of the loss on w_i. Under this frozen-error approximation,

$$\mathbb{E}\big[(\hat{u}_{W,i} - u_{W,i}^*)^2 \mid t\big] = A_i(t) w_i^2 - 2\Gamma_i(t) w_i + C_i(t), \tag{13}$$

with

$$A_i(t) = \mathbb{E}[e_{n,i}^2 \mid t], \qquad \Gamma_i(t) = \mathbb{E}[e_{n,i} e_{x,i} \mid t], \qquad C_i(t) = \mathbb{E}[e_{x,i}^2 \mid t], \tag{14}$$

and the fixed-timestep recovered-velocity loss becomes

$$\mathcal{E}_i(w_i; t) = \frac{A_i(t) w_i^2 - 2\Gamma_i(t) w_i + C_i(t)}{(\alpha_t w_i + \sigma_t)^2} + \mathbb{E}[(u_i - u_i^*)^2 \mid t], \tag{15}$$

where the last term is independent of w_i. The corresponding time-averaged loss is

$$\overline{\mathcal{E}}_i(w_i) = \mathbb{E}_t[\mathcal{E}_i(w_i; t)]. \tag{16}$$

Fixed-timestep optima. For a fixed timestep, differentiating the first term of Eq. (15) gives

$$\frac{\partial}{\partial w} \frac{A_i(t) w^2 - 2\Gamma_i(t) w + C_i(t)}{(\alpha_t w + \sigma_t)^2} = \frac{2\big[w\big(A_i(t)\sigma_t + \alpha_t \Gamma_i(t)\big) - \big(\Gamma_i(t)\sigma_t + \alpha_t C_i(t)\big)\big]}{(\alpha_t w + \sigma_t)^3}. \tag{17}$$

When $A_i(t)\sigma_t + \alpha_t\Gamma_i(t) \neq 0$, an interior stationary point satisfies

$$w_i^*(t) = \frac{\Gamma_i(t)\sigma_t + \alpha_t C_i(t)}{A_i(t)\sigma_t + \alpha_t\Gamma_i(t)}. \tag{18}$$

It is an interior minimum over $w \geq 0$ when $A_i(t)\sigma_t + \alpha_t\Gamma_i(t) > 0$ and $\Gamma_i(t)\sigma_t + \alpha_t C_i(t) > 0$. Otherwise the stationary point need not minimize the loss on $w \geq 0$. When the cross terms are negligible, specifically $|\alpha_t\Gamma_i(t)| \ll A_i(t)\sigma_t$ and $|\sigma_t\Gamma_i(t)| \ll \alpha_t C_i(t)$, the preferred weight reduces to

$$w_i^*(t) \approx \frac{\alpha_t}{\sigma_t}\frac{C_i(t)}{A_i(t)}. \tag{19}$$

Under $\Gamma_i(t) = 0$, $A_i(t) > 0$, $C_i(t) > 0$, and $t \in (0, 1)$, this stationary point is the unique minimizer of the timestep loss over $w \geq 0$.

When a constant weight suffices. Under $\Gamma_i(t) = 0$, the timestep-wise preferred weight $w_i^*(t) = (\alpha_t/\sigma_t)\, C_i(t)/A_i(t)$ is nominally time dependent through both the schedule ratio α_t/σ_t and the error ratio $C_i(t)/A_i(t)$. These two dependencies, however, naturally vary in opposite directions. At small t, σ_t is small and $\boldsymbol{z}_t$ is dominated by the clean data, making x_i relatively easy to predict but ϵ_i difficult to infer. At large t, $\boldsymbol{z}_t$ is increasingly dominated by $\boldsymbol{\epsilon}$, so estimating ϵ_i becomes easier while estimating x_i becomes harder. Accordingly, the data-estimation error $C_i(t)$ tends to increase with t, while the noise-estimation error $A_i(t)$ tends to decrease. This suggests that the time dependence in $A_i(t)$ and $C_i(t)$ may cancel the explicit schedule factor in $w_i^*(t)$. Consistent with this picture, in our time-dependent ablation, independently learned weights across several timestep buckets converge to very similar values, and allowing the weights to vary across timesteps does not improve FID (Fig. 5). We therefore consider the separable form

$$A_i(t) = \alpha_t q_i(t)\bar{A}_i, \qquad C_i(t) = \sigma_t q_i(t)\bar{C}_i. \tag{20}$$

The key assumption is that, after the factors α_t and σ_t, the two error powers share the same remaining time dependence $q_i(t) \geq 0$, with $\bar{A}_i, \bar{C}_i > 0$.

Shared minimizer under the separable form. With $\Gamma_i(t) = 0$, the per-timestep derivative becomes

$$\frac{\partial}{\partial w}\frac{A_i(t)w^2 + C_i(t)}{(\alpha_t w + \sigma_t)^2} = \frac{2[wA_i(t)\sigma_t - \alpha_t C_i(t)]}{(\alpha_t w + \sigma_t)^3} \tag{21}$$

$$= \frac{2\alpha_t\sigma_t q_i(t)\bar{A}_i}{(\alpha_t w + \sigma_t)^3}\left(w - \frac{\bar{C}_i}{\bar{A}_i}\right). \tag{22}$$

Thus every interior timestep with $q_i(t) > 0$ is minimized at the same weight

$$w_i^* = \frac{\bar{C}_i}{\bar{A}_i}, \tag{23}$$

while timesteps with $q_i(t) = 0$ are indifferent to the weight. Therefore, when AnisoFlow uses a single weight shared across timesteps, choosing this common value $w_i^* = \bar{C}_i/\bar{A}_i$ also minimizes the time-averaged loss $\overline{\mathcal{E}}_i(w_i)$ under any timestep distribution supported on interior timesteps, provided the expected loss is finite. The minimizer is unique if $q_i(t) > 0$ on a set of positive probability. A time-dependent parameterization $w_i(t)$ is therefore unnecessary under these conditions.

Near-linear scaling across directions. To characterize how the learned optimum varies across directions, we make two additional assumptions. Because the sampled noise is isotropic, we assume that the direction-dependent factor $\bar{A}_i$ in the noise-estimation error is roughly constant across directions, $\bar{A}_i \approx \bar{A}$. We further assume that the direction-dependent factor $\bar{C}_i$ in the data-estimation error scales roughly with the data variance. Since the data energy in direction i is $\mathbb{E}[x_i^2] = \lambda_i$, this gives $\bar{C}_i \propto \lambda_i$. The resulting optimum therefore satisfies

$$w_i^* = \frac{\bar{C}_i}{\bar{A}_i} \propto \lambda_i. \tag{24}$$

Our analysis does not predict the constant c in the observed near-linear scaling $w_i \approx c\lambda_i$, which depends on the network's error structure, the timestep sampler, and optimization. Departures from these assumptions can change the prefactor and lead to deviations from exact linear scaling.

The zero-skill limit. We consider a contrasting limiting case in which the network has no usable information about the realization of direction i beyond the timestep t, so that its target prediction is $\hat{u}_{W,i} = 0$ (the recovery map still uses $z_{t,i}$, so only the learned prediction is uninformative). For centered data independent of the noise, the recovered-velocity loss is then

$$\overline{\mathcal{E}}_i(w_i) = \mathbb{E}_t\left[\frac{\mathbb{E}[(w_i\epsilon_i - x_i)^2]}{(\alpha_t w_i + \sigma_t)^2}\right] = \mathbb{E}_t\left[\frac{w_i^2 + \lambda_i}{(\alpha_t w_i + \sigma_t)^2}\right]. \tag{25}$$

Unlike the separable structure of Eq. (20), the numerator does not vary with t, so the timestep-cancellation argument above does not apply, and the preferred constant weight can depend on the timestep distribution. If the weight were instead allowed to vary independently with time, minimizing the integrand at each timestep gives

$$w_i^*(t) = \frac{\alpha_t}{\sigma_t}\lambda_i. \tag{26}$$

Thus, without structure in the network's prediction errors, the preferred target retains an explicit timestep dependence.

Connection to EDM preconditioning. The pointwise zero-skill solution coincides with a direction-wise extension of EDM output preconditioning (Karras et al., 2022). At a fixed timestep $t < 1$, so that $\alpha_t > 0$, divide the flow state by α_t and write, in the PCA basis,

$$\tilde{z}_{t,i} = x_i + s_t\epsilon_i, \qquad s_t = \frac{\sigma_t}{\alpha_t}. \tag{27}$$

EDM parameterizes the clean-data estimate as $c_{\text{skip}}\tilde{z}_t + c_{\text{out}}F$, where F is the raw network output, and chooses c_{skip} to minimize the variance of the residual $x - c_{\text{skip}}\tilde{z}_t$ left for the network. This makes $c_{\text{skip}}\tilde{z}_t$ the linear least-squares estimate of x from $\tilde{z}_t$. Replacing EDM's scalar data variance by the variance λ_i of direction i gives

$$c_{\text{skip},i} = \frac{\lambda_i}{\lambda_i + s_t^2}. \tag{28}$$

The residual can then be written as

$$x_i - c_{\text{skip},i}\tilde{z}_{t,i} = (1 - c_{\text{skip},i})(x_i - w_i\epsilon_i), \tag{29}$$

$$w_i = \frac{c_{\text{skip},i}s_t}{1 - c_{\text{skip},i}} = \frac{\lambda_i}{s_t} = \frac{\alpha_t}{\sigma_t}\lambda_i. \tag{30}$$

Up to the timestep-dependent factor $-(1 - c_{\text{skip},i})$, this residual is the AnisoFlow target $u_{W,i} = w_i\epsilon_i - x_i$; EDM's additional output scaling c_{out} changes only this overall scale and not the relative weight w_i (Appendix A.1), and its input scaling does not affect the target. Hence a direction-wise extension of EDM yields exactly the pointwise zero-skill optimum in Eq. (26). This correspondence is intuitive: EDM's derivation uses only the second-order statistics of the data and noise, which is precisely the information available to a network with no predictive skill. Our general analysis instead depends on the network's learned error statistics $A_i(t)$, $C_i(t)$, and $\Gamma_i(t)$, whose timestep dependence can offset the explicit α_t/σ_t factor and favor a single weight profile shared across timesteps (Eq. (20); Fig. 5).

A.3 Empirical Analysis of Prediction Errors

We empirically investigate the prediction-error structure underlying the preferred-weight analysis of Appendix A.2, using a trained AnisoFlow model on ImageNet.

Experimental setup. We use the 600-epoch AnisoFlow-H/16 checkpoint with a frozen PCA basis. Since AnisoFlow predicts the combined target $w_i\epsilon_i - x_i$ through a single output head, its prediction cannot be uniquely decomposed into separate noise and data estimates. Our analysis therefore provides a consistency check of the theoretical explanation rather than a direct measurement of the underlying error components. We freeze the network and fit two auxiliary linear readouts to its intermediate features, producing estimates $\hat{\epsilon}_i$ and $\hat{x}_i$. Since the original output head is itself a linear projection of these features, linear readouts provide a natural way to estimate data and noise separately without modifying the learned representation. The readouts are fitted using weighted ridge regression on 30K ImageNet training images and evaluated on a disjoint set of 30K images.

We select 60 PCA directions across the 768-dimensional patch spectrum and restrict the analysis to 18 directions for which the auxiliary readouts accurately reconstruct the original model's predictions (relative reconstruction discrepancy below 5% and recovered-velocity MSE increase below 1%). These criteria assess reconstruction fidelity rather than agreement with the predicted scaling. The retained directions span nearly three orders of magnitude in directional variance, from $\lambda_i = 0.205$ to 171.

Measuring the error statistics. The theoretical quantities $A_i(t)$, $C_i(t)$, and $\Gamma_i(t)$ in Appendix A.2 are defined relative to the conditional means of the data and noise, which are not directly available for ImageNet. Instead, we measure residuals against the sampled data and noise:

$$r_{n,i} = \hat{\epsilon}_i - \epsilon_i, \qquad r_{x,i} = \hat{x}_i - x_i. \tag{31}$$

For each direction and timestep, we estimate the second moments

$$\begin{aligned} a_i(t) &= \mathbb{E}[r_{n,i}^2 \mid t], \\ b_i(t) &= \mathbb{E}[r_{x,i}^2 \mid t], \\ g_i(t) &= \mathbb{E}[r_{n,i} r_{x,i} \mid t], \end{aligned} \tag{32}$$

averaging over evaluation images and spatial patches.

Let $x_i^* = \mathbb{E}[x_i \mid \boldsymbol{z}_t, t]$ and define the irreducible posterior uncertainty $F_i(t) = \mathbb{E}[(x_i - x_i^*)^2 \mid t]$, with class conditioning implicit. Since $\boldsymbol{z}_t = \alpha_t \boldsymbol{x} + \sigma_t \boldsymbol{\epsilon}$, the posterior uncertainties satisfy

$$\epsilon_i - \epsilon_i^* = -\frac{\alpha_t}{\sigma_t}(x_i - x_i^*). \tag{33}$$

Using the orthogonality of conditional-mean residuals, the observed moments decompose as

$$\begin{aligned} a_i(t) &= A_i(t) + \frac{\alpha_t^2}{\sigma_t^2} F_i(t), \\ b_i(t) &= C_i(t) + F_i(t), \\ g_i(t) &= \Gamma_i(t) - \frac{\alpha_t}{\sigma_t} F_i(t). \end{aligned} \tag{34}$$

We can therefore construct two effective error statistics:

$$\begin{aligned} A_i^{\text{eff}}(t) &= a_i(t) + \frac{\alpha_t}{\sigma_t} g_i(t) = A_i(t) + \frac{\alpha_t}{\sigma_t}\Gamma_i(t), \\ C_i^{\text{eff}}(t) &= b_i(t) + \frac{\sigma_t}{\alpha_t} g_i(t) = C_i(t) + \frac{\sigma_t}{\alpha_t}\Gamma_i(t). \end{aligned} \tag{35}$$

The irreducible posterior uncertainty cancels exactly. Furthermore, the full preferred-weight expression from Appendix A.2 becomes

$$w_i^*(t) = \frac{\alpha_t}{\sigma_t} \frac{C_i^{\text{eff}}(t)}{A_i^{\text{eff}}(t)}, \tag{36}$$

for an interior optimum under the local frozen-error model. This expression accounts for the cross moment $\Gamma_i(t)$ without assuming that it is negligible.

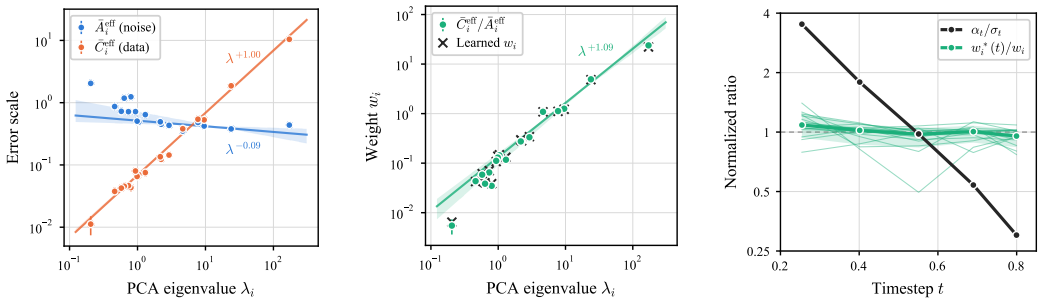
Directional scaling. Following the timestep normalization motivated by Eq. (20), we compute

$$\bar{A}_i^{\text{eff}} = \mathbb{E}_t\left[\frac{A_i^{\text{eff}}(t)}{\alpha_t}\right], \qquad \bar{C}_i^{\text{eff}} = \mathbb{E}_t\left[\frac{C_i^{\text{eff}}(t)}{\sigma_t}\right], \tag{37}$$

using the training logit-normal timestep distribution, restricted to the reliable evaluation bins.

As shown in Fig. 8(a), the effective noise-error scale varies weakly with directional variance, approximately $\bar{A}_i^{\text{eff}} \propto \lambda_i^{-0.09}$, while the effective data-error scale follows $\bar{C}_i^{\text{eff}} \propto \lambda_i^{1.00}$. Their ratio scales as $\lambda_i^{1.09}$, closely matching the exponent of 1.07 for the weights learned by AnisoFlow, fitted over the same 18 directions.

Figure 8(b) compares the ratio $\bar{C}_i^{\text{eff}}/\bar{A}_i^{\text{eff}}$ with the learned weights. The two profiles agree closely in both scaling and magnitude, with a median ratio $(\bar{C}_i^{\text{eff}}/\bar{A}_i^{\text{eff}})/w_i$ of approximately 0.97.



(a) Effective error scales (b) Error ratio and learned weights (c) Timestep cancellation

Figure 8: **Estimated prediction-error structure on ImageNet. (a)** Time-averaged effective noise and data error scales versus PCA variance, with fitted scaling exponents. **(b)** Preferred weights estimated from the effective errors compared with the learned AnisoFlow weights. **(c)** Timestep dependence of the estimated preferred weights, normalized by the learned weights, compared with the schedule factor α_t/σ_t (normalized by its geometric mean). Thin curves show individual directions, and the curve with markers shows the median. Results are shown for 18 PCA directions satisfying reconstruction-consistency criteria.

Robustness to direction selection. We also relax the reconstruction-consistency threshold from 5% to 10–20%, increasing the number of analyzed directions from 18 to 45–51 and extending the variance range toward lower-variance directions. The effective-error ratio consistently scales as approximately $\lambda_i^{1.09}$, closely matching the learned-weight exponent of approximately 1.07 across all thresholds.

Timestep dependence. We further examine the timestep-dependent preferred weights from Eq. (36). Across five reliable timestep bins spanning approximately $t = 0.25$ to 0.80, the explicit schedule factor α_t/σ_t varies by $11.7\times$. However, the preferred weights estimated from these effective errors vary by only $1.19\times$ at the median across directions, as shown in Fig. 8(c). This indicates that the timestep dependence of the error ratio largely cancels the schedule factor, supporting the use of a single weight profile shared across timesteps.

A.4 Learning the Basis

We learn the basis $\boldsymbol{B}$ with one of two optimizers: Stiefel Adam, which optimizes $\boldsymbol{B}$ directly on the manifold of orthonormal matrices, or the spectral update, a closed-form eigendecomposition step. We describe each in turn.

A.4.1 Stiefel Adam

Stiefel Adam treats $\boldsymbol{B}$ as a parameter constrained to the Stiefel manifold $\{\boldsymbol{B} : \boldsymbol{B}^{\mathrm{T}}\boldsymbol{B} = \boldsymbol{I}\}$ and adapts Adam (Kingma & Ba, 2015) to it. Let $\mathcal{P}_{\boldsymbol{B}}(\boldsymbol{A}) = \boldsymbol{A} - \boldsymbol{B}\,\mathrm{sym}(\boldsymbol{B}^{\mathrm{T}}\boldsymbol{A})$, with $\mathrm{sym}(\boldsymbol{A}) = \frac{1}{2}(\boldsymbol{A} + \boldsymbol{A}^{\mathrm{T}})$, denote the projection onto the tangent space at $\boldsymbol{B}$ under the Euclidean metric, and let $\boldsymbol{P}$ and v denote the first-moment matrix and a scalar second moment, with step size τ. Each step computes the projected gradient $\boldsymbol{\Xi}$ and the updates

$$\begin{aligned} &\boldsymbol{\Xi} \leftarrow \mathcal{P}_{\boldsymbol{B}}(\nabla_{\boldsymbol{B}}\mathcal{L}), \qquad \boldsymbol{P} \leftarrow \beta_1 \boldsymbol{P} + (1-\beta_1)\,\boldsymbol{\Xi}, \qquad v \leftarrow \beta_2 v + (1-\beta_2)\,\|\boldsymbol{\Xi}\|_F^2/D^2, \\ &\boldsymbol{B} \leftarrow \boldsymbol{Q}, \quad \text{where } \boldsymbol{Q}\boldsymbol{R} = \boldsymbol{B} - \tau\,\hat{\boldsymbol{P}}\big/\big(\sqrt{\hat{v}} + \delta\big), \qquad \boldsymbol{P} \leftarrow \mathcal{P}_{\boldsymbol{B}}(\boldsymbol{P}), \end{aligned} \tag{38}$$

where $\hat{\boldsymbol{P}}$ and $\hat{v}$ are the bias-corrected moments, $\delta > 0$ is Adam's numerical-stability constant (written δ rather than ϵ to avoid a clash with the noise $\boldsymbol{\epsilon}$), $\boldsymbol{Q}\boldsymbol{R}$ denotes the reduced QR decomposition with $\boldsymbol{R}$ upper triangular with positive diagonal so that $\boldsymbol{Q}$ is unique, and the final projection transports the momentum to the tangent space at the new point. The second moment is a single scalar per basis, so the tangent momentum is scaled by one adaptive step size. An elementwise rescaling would generally leave the tangent space. Weight decay is zero: $\|\boldsymbol{B}\|_F^2 = D$ is constant on the manifold, and the tangent projection removes the gradient of an L_2 penalty. We use the same learning rate and Adam coefficients as for the model parameters (Appendix B). This is a projection-transport Stiefel

optimizer under the Euclidean metric. It does not implement parallel transport or the canonical metric. It makes no structural assumption about the loss and serves as the general-purpose alternative to the closed-form update derived next.

A.4.2 SPECTRAL UPDATE

We now derive the spectral update used in our main experiments. Let $\boldsymbol{r} := \mathrm{sg}(\hat{\boldsymbol{u}}_W) + \boldsymbol{x}$, where sg denotes stop-gradient: the network prediction is held fixed while the basis is updated. In direction $\boldsymbol{b}_i$, the recovered-velocity error of Eq. (5) is the target-prediction error divided by the recovery denominator, $\hat{u}_i - u_i = (\boldsymbol{b}_i^{\mathrm{T}}\boldsymbol{r} - w_i \boldsymbol{b}_i^{\mathrm{T}}\boldsymbol{\epsilon})/(\alpha_t w_i + \sigma_t)$. Squaring, taking the expectation over $\boldsymbol{x}$, $\boldsymbol{\epsilon}$, and t, and dropping the $w_i^2 \boldsymbol{b}_i^{\mathrm{T}}\boldsymbol{\epsilon}\boldsymbol{\epsilon}^{\mathrm{T}}\boldsymbol{b}_i$ term, whose contribution $\mathbb{E}_t[w_i^2/(\alpha_t w_i + \sigma_t)^2]$ does not depend on $\boldsymbol{b}_i$, the basis-dependent part of the loss is

$$\mathcal{L}_{\boldsymbol{B}}(\boldsymbol{B}) = \sum_{i=1}^{D} \boldsymbol{b}_i^{\mathrm{T}} \boldsymbol{M}_i \boldsymbol{b}_i + \text{const}, \qquad \boldsymbol{M}_i = \mathbb{E}_t\left[\frac{\boldsymbol{S}_t - 2w_i \boldsymbol{H}_t}{(\alpha_t w_i + \sigma_t)^2}\right], \tag{39}$$

where $\boldsymbol{S}_t := \mathbb{E}[\boldsymbol{r}\boldsymbol{r}^{\mathrm{T}} \mid t]$ and $\boldsymbol{H}_t := \frac{1}{2}\mathbb{E}[\boldsymbol{r}\boldsymbol{\epsilon}^{\mathrm{T}} + \boldsymbol{\epsilon}\boldsymbol{r}^{\mathrm{T}} \mid t]$ are the second moment of $\boldsymbol{r}$ and its symmetric cross moment with the noise. This expression is exact given the network output and the weights.

Minimizing Eq. (39) over orthonormal $\boldsymbol{B}$ is not an eigenvalue problem in general: each column $\boldsymbol{b}_i$ is scored by a different matrix $\boldsymbol{M}_i$, because w_i enters both the numerator and the recovery denominator, so no single eigendecomposition solves it. Direct optimization on the Stiefel manifold (Appendix A.4.1) is the general-purpose alternative. Under the following assumption, however, one eigendecomposition suffices.

Assumption 1. The matrices $\boldsymbol{M}_1, \ldots, \boldsymbol{M}_D$ commute pairwise. Equivalently, they are simultaneously orthogonally diagonalizable, i.e., $\boldsymbol{M}_i = \boldsymbol{B}_\star \boldsymbol{\Lambda}_i \boldsymbol{B}_\star^{\mathrm{T}}$ for a common orthonormal $\boldsymbol{B}_\star$ and diagonal $\boldsymbol{\Lambda}_i = \mathrm{diag}(\kappa_{i1}, \ldots, \kappa_{iD})$.

Proposition 2 (Optimal basis under joint diagonalizability)**.** Under Assumption 1, a global minimizer of Eq. (39) over $\boldsymbol{B}^{\mathrm{T}}\boldsymbol{B} = \boldsymbol{I}$ is $\boldsymbol{B} = \boldsymbol{B}_\star \boldsymbol{\Pi}\,\mathrm{diag}(s_1, \ldots, s_D)$ for some permutation matrix $\boldsymbol{\Pi}$ and signs $s_i \in \{\pm 1\}$. Moreover, for any coefficients γ_i such that $\sum_i \gamma_i \boldsymbol{M}_i$ has distinct eigenvalues, the eigenvectors of $\sum_i \gamma_i \boldsymbol{M}_i$ recover the common eigendirections. An optimal pairing assigns column i to direction $\pi(i)$ minimizing $\sum_i \kappa_{i,\pi(i)}$ over permutations π.

Proof. Write $\boldsymbol{B} = \boldsymbol{B}_\star \boldsymbol{\Omega}$ with $\boldsymbol{\Omega}$ orthogonal. Then $\boldsymbol{b}_i^{\mathrm{T}} \boldsymbol{M}_i \boldsymbol{b}_i = \sum_j \kappa_{ij} \Omega_{ji}^2$, so $\mathcal{L}_{\boldsymbol{B}} = \sum_{i,j} \kappa_{ij} \Omega_{ji}^2 +$ const is linear in the matrix $[\Omega_{ji}^2]$, which is doubly stochastic because $\boldsymbol{\Omega}$ is orthogonal. A linear function on the set of doubly stochastic matrices attains its minimum at a permutation matrix (Birkhoff–von Neumann), and every permutation matrix is itself an orthogonal $\boldsymbol{\Omega}$, so the minimum is attained at $\boldsymbol{\Omega} = \boldsymbol{\Pi}\,\mathrm{diag}(s_1, \ldots, s_D)$. The second claim follows because $\sum_i \gamma_i \boldsymbol{M}_i = \boldsymbol{B}_\star (\sum_i \gamma_i \boldsymbol{\Lambda}_i) \boldsymbol{B}_\star^{\mathrm{T}}$ has eigenvectors $\boldsymbol{B}_\star$ whenever its eigenvalues are distinct. □

We use the uniform average, which weights every output direction once, as in the squared loss, and introduces no directional preference. Substituting Eq. (39),

$$\boldsymbol{G} := \frac{1}{D}\sum_{i=1}^{D} \boldsymbol{M}_i = \mathbb{E}_t\big[a_t\, \boldsymbol{S}_t - 2b_t\, \boldsymbol{H}_t\big] = \mathbb{E}\big[a_t\, \boldsymbol{r}\boldsymbol{r}^{\mathrm{T}} - b_t(\boldsymbol{r}\boldsymbol{\epsilon}^{\mathrm{T}} + \boldsymbol{\epsilon}\boldsymbol{r}^{\mathrm{T}})\big], \tag{40}$$

with

$$a_t := \frac{1}{D}\sum_{j=1}^{D} \frac{1}{(\alpha_t w_j + \sigma_t)^2}, \qquad b_t := \frac{1}{D}\sum_{j=1}^{D} \frac{w_j}{(\alpha_t w_j + \sigma_t)^2}. \tag{41}$$

The spectral update sets $\boldsymbol{B} \leftarrow \mathrm{eigvecs}(\boldsymbol{G})$. Under Assumption 1, these are the common eigendirections whenever the eigenvalues of $\boldsymbol{G}$ are distinct. Commutativity alone does not guarantee this: a repeated eigenvalue of $\boldsymbol{G}$ determines the corresponding eigenvectors only up to a rotation within their span. The cross term accounts for the correlation between $\boldsymbol{r}$ and the noise in the target, and omitting it changes the basis objective.

Remark 3 (Pairing eigenvectors with weights). We order the eigenvectors of $\boldsymbol{G}$ by ascending eigenvalue and keep w_i attached to rank i. This rank-based pairing is a heuristic; it need not solve the assignment problem of Proposition 2, but it keeps each weight attached to a stable direction as long as eigenvalue ranks do not cross. Adam continues to adapt the weights after each basis update. Column signs cancel in $\boldsymbol{B}\,\mathrm{diag}(w_i)\boldsymbol{B}^{\mathrm{T}}$.

When Assumption 1 holds only approximately, $\boldsymbol{M}_i = \boldsymbol{B}_\star \boldsymbol{\Lambda}_i \boldsymbol{B}_\star^{\mathrm{T}} + \boldsymbol{\Delta}_i$ with small non-commuting residuals $\boldsymbol{\Delta}_i$, we have $\boldsymbol{G} = \boldsymbol{B}_\star \bar{\boldsymbol{\Lambda}} \boldsymbol{B}_\star^{\mathrm{T}} + \bar{\boldsymbol{\Delta}}$ with $\bar{\boldsymbol{\Lambda}} = \frac{1}{D}\sum_i \boldsymbol{\Lambda}_i$ and $\bar{\boldsymbol{\Delta}} = \frac{1}{D}\sum_i \boldsymbol{\Delta}_i$, and standard eigenvector perturbation bounds keep $\mathrm{eigvecs}(\boldsymbol{G})$ close to $\boldsymbol{B}_\star$ whenever $\|\bar{\boldsymbol{\Delta}}\|$ is small relative to the eigengaps of $\bar{\boldsymbol{\Lambda}}$. The spectral update is then an approximate joint-diagonalization step. Appendix A.5 shows that the assumption holds exactly at covariance-aligned solutions in a linear-network model.

Online estimator. During training we estimate $\boldsymbol{G}$ from minibatches. For a batch of M images with N patches each, the estimate is

$$\widehat{\boldsymbol{G}} = \frac{1}{MN}\sum_{m=1}^{M}\sum_{p=1}^{N}\left[a_{t_m}\boldsymbol{r}_{mp}\boldsymbol{r}_{mp}^{\mathrm{T}} - b_{t_m}(\boldsymbol{r}_{mp}\boldsymbol{\epsilon}_{mp}^{\mathrm{T}} + \boldsymbol{\epsilon}_{mp}\boldsymbol{r}_{mp}^{\mathrm{T}})\right], \tag{42}$$

normalized by the number of patches rather than by the random sum of recovery weights, so that it estimates a fixed population quantity. We maintain an exponential moving average $\boldsymbol{G} \leftarrow (1-\beta)\boldsymbol{G} + \beta\,\widehat{\boldsymbol{G}}$ and set $\boldsymbol{B}$ to the eigenvectors of $\boldsymbol{G}$ at every optimization step (Appendix B). The moving average smooths the moment estimate, although individual eigenvectors can still change abruptly near eigenvalue crossings.

Relation to PCA initialization. When the network output is zero, as is common at initialization, $\boldsymbol{r} = \boldsymbol{x}$. The data and sampled noise are independent, so $\mathbb{E}[\boldsymbol{x}\boldsymbol{\epsilon}^{\mathrm{T}}] = \boldsymbol{0}$, and a_t depends only on the timestep and the current weights. Eq. (40) therefore reduces in expectation to a scalar multiple of the data second moment, $\boldsymbol{G} \propto \mathbb{E}[\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}}]$, which for centered data is the covariance, so its eigenvectors are the PCA directions. In practice the running moment is initialized to $\boldsymbol{I}$ (Appendix B); since adding a multiple of the identity leaves eigenvectors unchanged, the first basis consists of the eigendirections of a single minibatch estimate, a noisy PCA basis rather than the exact one. The first spectral update from $\boldsymbol{W} = \boldsymbol{I}$ thus recovers PCA structure on its own, which is why PCA initialization is optional. Appendix A.5 studies when this alignment is preserved as training proceeds.

A.5 Linear-Model Analysis of Basis Alignment

We analyze the spectral update in an idealized setting: the weights are fixed, the network is replaced by the optimal time-conditioned linear predictor, and the basis is updated by the exact iteration $\boldsymbol{B} \leftarrow \mathrm{eigvecs}(\boldsymbol{G})$ in place of the moving-average estimate of Appendix A.4.2. We show that this iteration preserves alignment with the data covariance and give a condition under which the alignment is locally stable.

Linear denoising model. Let $\boldsymbol{\Sigma} = \mathbb{E}[\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}}] = \boldsymbol{V}\,\mathrm{diag}(\lambda_1,\dots,\lambda_D)\boldsymbol{V}^{\mathrm{T}}$ be the covariance of centered data, and let $\boldsymbol{\Sigma}_{z,t} = \mathbb{E}[\boldsymbol{z}_t\boldsymbol{z}_t^{\mathrm{T}}] = \alpha_t^2\boldsymbol{\Sigma} + \sigma_t^2\boldsymbol{I}$. For a symmetric $\boldsymbol{W}$ and a linear predictor $\hat{\boldsymbol{u}}_W = \boldsymbol{\Phi}_t\boldsymbol{z}_t$, minimizing the recovered-velocity loss over $\boldsymbol{\Phi}_t$ gives the same minimizer as least squares on the target, since recovery applies the fixed invertible map $(\alpha_t\boldsymbol{W} + \sigma_t\boldsymbol{I})^{-1}$ to the target error. Hence

$$\boldsymbol{\Phi}_t^*(\boldsymbol{W}) = (\sigma_t\boldsymbol{W} - \alpha_t\boldsymbol{\Sigma})\boldsymbol{\Sigma}_{z,t}^{-1}, \tag{43}$$

since $\mathbb{E}[\boldsymbol{u}_W\boldsymbol{z}_t^{\mathrm{T}}] = \sigma_t\boldsymbol{W} - \alpha_t\boldsymbol{\Sigma}$. Substituting into the recovery map of Eq. (5) gives

$$(\alpha_t\boldsymbol{W} + \sigma_t\boldsymbol{I})^{-1}(\boldsymbol{I} - \boldsymbol{W} + \boldsymbol{\Phi}_t^*) = (\sigma_t\boldsymbol{I} - \alpha_t\boldsymbol{\Sigma})\boldsymbol{\Sigma}_{z,t}^{-1}. \tag{44}$$

Thus the recovered velocity predictor, and hence the optimal linear loss, do not depend on $\boldsymbol{W}$. The linear model therefore identifies neither the weights nor the basis, and the alignment shown below is a property of the spectral update itself rather than of the loss.

Preservation of covariance alignment. For the fitted predictor, the moments in Eq. (39) are

$$\boldsymbol{S}_t = \sigma_t^2\big(\boldsymbol{W}\boldsymbol{\Sigma}_{z,t}^{-1}\boldsymbol{W} + \boldsymbol{\Sigma}\boldsymbol{\Sigma}_{z,t}^{-1}\big), \tag{45}$$

$$\boldsymbol{H}_t = \frac{\sigma_t^2}{2}\big(\boldsymbol{W}\boldsymbol{\Sigma}_{z,t}^{-1} + \boldsymbol{\Sigma}_{z,t}^{-1}\boldsymbol{W}\big) - \alpha_t\sigma_t\boldsymbol{\Sigma}\boldsymbol{\Sigma}_{z,t}^{-1}. \tag{46}$$

These formulas follow by substituting $\boldsymbol{r} = (\alpha_t \boldsymbol{\Phi}_t^* + \boldsymbol{I})\boldsymbol{x} + \sigma_t \boldsymbol{\Phi}_t^* \boldsymbol{\epsilon}$, using $\alpha_t \boldsymbol{\Phi}_t^* + \boldsymbol{I} = \sigma_t(\alpha_t \boldsymbol{W} + \sigma_t \boldsymbol{I})\boldsymbol{\Sigma}_{z,t}^{-1}$, the independence of $\boldsymbol{x}$ and $\boldsymbol{\epsilon}$, and the fact that $\boldsymbol{\Sigma}_{z,t}$ commutes with $\boldsymbol{\Sigma}$. If $\boldsymbol{W}$ commutes with $\boldsymbol{\Sigma}$, all of these matrices, and hence every $\boldsymbol{M}_i$ and $\boldsymbol{G}$, are diagonal in a common eigenbasis of $\boldsymbol{W}$ and $\boldsymbol{\Sigma}$, so Assumption 1 holds exactly. When $\boldsymbol{G}$ has distinct eigenvalues, the update returns the PCA basis up to sign and permutation, so alignment with the data covariance is preserved. If the rank-based pairing of Remark 3 also leaves each weight attached to its direction, $\boldsymbol{W}$ itself is a fixed point. When $\boldsymbol{G}$ has a repeated eigenvalue, the update may rotate within the tied eigenspace. Arbitrary rotations within that space are guaranteed to preserve alignment if the restriction of $\boldsymbol{\Sigma}$ to that eigenspace is a multiple of the identity.

For a nearly aligned $\boldsymbol{W}$, the identity

$$[\boldsymbol{\Phi}_t^*, \boldsymbol{\Sigma}] = \sigma_t [\boldsymbol{W}, \boldsymbol{\Sigma}] \boldsymbol{\Sigma}_{z,t}^{-1} \tag{47}$$

shows that a small misalignment of $\boldsymbol{W}$ induces a proportionally small misalignment of the predictor. Since the moments are continuous in $\boldsymbol{W}$, the eigenvectors of $\boldsymbol{G}$ also vary continuously as long as its eigenvalues stay separated.

Local stability. Consider an aligned fixed point $\boldsymbol{W}_\star = \mathrm{diag}(w_1, \ldots, w_D)$ in the PCA basis whose rank-based pairing (Remark 3) is locally unchanged, and assume the eigenvalues $\bar{\kappa}_j = \frac{1}{D}\sum_i \kappa_{ij}$ of $\boldsymbol{G}$ there, in the notation of Appendix A.4.2, are distinct. Set $d_j(t) = \alpha_t^2 \lambda_j + \sigma_t^2$, the eigenvalues of $\boldsymbol{\Sigma}_{z,t}$. Differentiating the moment formulas gives, for an off-diagonal perturbation,

$$\delta G_{jk} = L_{jk}\, \delta W_{jk}, \qquad L_{jk} = \mathbb{E}_t\left[\sigma_t^2 \left\{ a_t \left(\frac{w_j}{d_j(t)} + \frac{w_k}{d_k(t)} \right) - b_t \left(\frac{1}{d_j(t)} + \frac{1}{d_k(t)} \right) \right\}\right]. \tag{48}$$

A small rotation of the basis by an angle θ_{jk} in the (j,k) plane gives $\delta W_{jk} = (w_j - w_k)\theta_{jk}$, so

$$\theta_{jk}^{\mathrm{new}} = \rho_{jk}\theta_{jk} + O(\|\boldsymbol{\theta}\|^2), \qquad \rho_{jk} = \frac{L_{jk}(w_j - w_k)}{\bar{\kappa}_j - \bar{\kappa}_k}. \tag{49}$$

The fixed point is locally attractive when $|\rho_{jk}| < 1$ for every pair with $w_j \neq w_k$ (rotations between equal weights leave $\boldsymbol{W}$ unchanged). Larger eigengaps of $\boldsymbol{G}$ reduce sensitivity for a fixed numerator in ρ_{jk}. At an exact tie, the corresponding eigenvectors are not determined by $\boldsymbol{G}$.

Scope. These results concern the idealized linear iteration. They do not establish global convergence, nor stability of the joint dynamics of the network, the weights, and the running moment. Empirically, the basis learned with a nonlinear network aligns closely with the PCA basis (Fig. 3b).

B Implementation Details

B.1 AnisoFlow Parameterization and Optimization

Learning the directional weights. We write

$$\boldsymbol{W} = \boldsymbol{B}\,\mathrm{diag}(w_1, \ldots, w_D)\boldsymbol{B}^{\mathrm{T}}, \qquad w_i \geq 0, \tag{50}$$

and learn the directional weights jointly with the network. To enforce strict positivity, we set $w_i = \exp(\eta_i)$ and optimize η with Adam. Unless otherwise noted, η uses the same learning rate as the model parameters. By default, the weights are shared across timesteps and are optimized from the start of training.

Learning the basis. By default, the orthogonal basis $\boldsymbol{B}$ is updated during training using the spectral update of Appendix A.4. The running moment $\boldsymbol{G}$ is computed from the stop-gradient network prediction. It is initialized to $\boldsymbol{I}$ and updated with an exponential moving average with coefficient $\beta = 10^{-5}$. At every optimization step, the running moment is averaged across devices, eigendecomposed, and $\boldsymbol{B}$ is set to its eigenvectors ordered by ascending eigenvalue, with w_i attached to eigenvalue rank i. Alternatively, Stiefel Adam (Appendix A.4.1) optimizes $\boldsymbol{B}$ directly with the same learning rate and Adam coefficients as the model parameters; in this variant the basis of the EMA model is re-orthogonalized by QR after each EMA step.

Numerical details. At inference, the recovery denominator $\alpha_t w_i + \sigma_t$ is lower-bounded by $\sigma_{\min} = 4 \times 10^{-2}$, following AsymFlow (Chen et al., 2026a) for the baselines and AnisoFlow alike.

B.2 IMAGENET PIXEL-SPACE EXPERIMENTS

Training setup. We conduct our main pixel-space experiments on ImageNet 256×256 using the same architecture, optimizer, and training hyperparameters as JiT-H/16 (see Table 9 of JiT (Li & He, 2026)), changing only the prediction-target parameterization. We use JiT-B/16 for smaller-scale experiments. Unless otherwise stated, the main models are trained for 600 epochs.

Data covariance and PCA estimation. The $\boldsymbol{W} = \boldsymbol{I}$ AnisoFlow variants do not require a PCA basis or a precomputed data covariance. We estimate the covariance for the PCA-initialized and fixed-PCA experiments and for analyzing the learned directions and weights. For JiT with 16×16 patches, each RGB patch is represented as a $16\times16\times3 = 768$-dimensional vector. We compute the empirical covariance of mean-centered patches from $100{,}000$ training images, in the same normalized pixel space seen by the network, and eigendecompose it as $\boldsymbol{\Sigma} = \boldsymbol{V}\,\mathrm{diag}(\lambda_1, \ldots, \lambda_D)\boldsymbol{V}^\top$. Here, $\boldsymbol{V}$ denotes the PCA basis and λ_i the corresponding PCA eigenvalues used in the analyses of §5. For JiT-H/16, $\boldsymbol{W} \in \mathbb{R}^{768\times768}$ acts independently on each image patch and is shared across spatial locations; for an image containing N patches, the corresponding image-level operator is $\boldsymbol{I}_N \otimes \boldsymbol{W}$.

Classifier-free guidance. Following AsymFlow (Chen et al., 2026a), for each AnisoFlow model, we grid-search the CFG scale (Ho & Salimans, 2021) with step size 0.1 and the guidance interval with step size 0.02. Table 5 lists the selected settings for Table 1.

Table 5: **Selected guidance settings for the AnisoFlow models in Table 1.** CFG scale / guidance interval selected by grid search for each model and training length.

Noise basis	Init	@160ep	@600ep
Frozen (PCA)	PCA	3.1 / $[0, 0.76]$	2.3 / $[0, 0.88]$
Stiefel Adam	PCA	3.1 / $[0, 0.74]$	2.1 / $[0, 0.90]$
Spectral update	PCA	3.1 / $[0, 0.74]$	2.3 / $[0, 0.88]$
Stiefel Adam	–	4.4 / $[0, 0.62]$	2.3 / $[0, 0.84]$
Spectral update	–	3.6 / $[0, 0.66]$	2.1 / $[0, 0.86]$

B.3 TEXT-TO-IMAGE EXPERIMENTS

We evaluate AnisoFlow on pixel-space text-to-image generation with JiT-L/16-T2I as the baseline, which uses the MM-JiT denoiser architecture from MiniT2I (Wang et al., 2026e) and the T5Gemma (Zhang et al., 2025) text encoder from i1 (Zeng et al., 2026). The baseline uses clean-data prediction, and AnisoFlow differs from it only in the prediction target. Both models are trained on i1's data (Zeng et al., 2026), which combines the public ImageNet-22K (Deng et al., 2009), Places365 (Zhou et al., 2017), Pexels (Pexels, 2025), TextAtlas (Wang et al., 2026a), RedCaps (Desai et al., 2021), and Megalith-10M (Boer Bohan, 2024) with i1's rendered-text and synthetic sources, plus YFCC100M (Thomee et al., 2016) and the released iNaturalist split (Van Horn et al., 2018) in the 256×256 stage only. We follow an i1-equivalent schedule through the 256×256 and 512×512 stages, doubling the batch size to 1024 while halving the number of iterations. As in the pixel-space ImageNet models, $\boldsymbol{W} \in \mathbb{R}^{768\times768}$ acts on $16\times16\times3$ patches. AnisoFlow uses the PCA initialization and the spectral update (EMA coefficient 10^{-5}) during the 256×256 tier. During the 512×512 tier, the basis is frozen while the weights w_i continue to be learned. For hyperparameters that cannot be shared between pixel and latent models (e.g., flow shift (Esser et al., 2024)), we tune them ourselves. Table 6 summarizes the settings of each training stage.

Evaluation. All models are sampled with a 32-step SDE solver based on UniPC (Zhao et al., 2023) with the stochastic coefficient of ER-SDE (Cui et al., 2025), using the same guidance settings for the quantitative comparison: a guidance scale of 6.0 on the interval $t \in [0, 0.99]$ with orthogonal projection from APG (Sadat et al., 2025). For benchmarking, we use i1's rewritten prompts for LongText-Bench, GenEval, and DPG-Bench, while other benchmarks use their original prompts.

Reduced-precision inference. To evaluate numerical stability, we sample the JiT baseline and AnisoFlow models with the weights and activations of the transformer's linear layers quantized to 8 bits (W8A8), while keeping prompts, seeds, and sampler settings fixed.

High-resolution qualitative samples. The one-megapixel samples in Fig. 1 come from an additional one-megapixel tier of the AnisoFlow model, which we do not evaluate quantitatively due to

Table 6: **Text-to-image model and training settings.** The 256×256 and 512×512 stages are used for the quantitative comparison, and the one-megapixel (1MP) and alignment fine-tuning (SFT) stages only for the qualitative samples in Fig. 1. Each stage is initialized from the previous one.

Setting	**256×256**	**512×512**	**1MP**	**SFT**
Pixel color space	Normalized Oklab (Ottosson, 2020; Chen et al., 2026a)			
Patch size / dimension D	16 / 768			
Text encoder	T5Gemma-2B (Zhang et al., 2025), 256 tokens			
AnisoFlow weights	Learned			
AnisoFlow basis	PCA init, spectral update	Frozen	Frozen	Frozen
Optimizer	Adam (Kingma & Ba, 2015), betas $(0.9, 0.95)$, weight decay 0			
Precision	BF16 autocast, FP32 master weights			
Prompt dropout	0.1			
EMA	Dynamic EMA (Karras et al., 2024), $\gamma = 7.0$, reset at each stage			
Resolution	256×256	512×512	1MP, mixed aspect ratios	1.56MP, mixed aspect ratios
Training data	i1	i1	i1 (real only) + LAION-3M	Z-Image Turbo synthetic
Batch size	1024	1024	256	256
Training iterations	1M	250K	150K	2K
Learning rate	2×10^{-4}	2×10^{-4}	2×10^{-4}	10^{-5}
Warmup iterations	1000	–	–	100
Pre-shift time sampling	$\mathrm{LogitNormal}(0, 1)$	Uniform	Uniform	Uniform
Flow shift	2.0	3.0	5.0	6.0

the additional cost of training a matching baseline. For this tier, we remove the synthetic-data mix from i1's data and add the 3M LAION subset (Schuhmann et al., 2022) used by AsymFlow (Chen et al., 2026a), which has mixed aspect ratios, followed by a short alignment fine-tuning stage on synthetic images from Z-Image Turbo (Z-Image Team et al., 2025), rendered at about 1.56 megapixels, with a reduced learning rate of 10^{-5}.

B.4 Latent-Space Experiments

SiT-XL/2-REPA. We evaluate AnisoFlow on SiT-XL/2-REPA, preserving the original latent representation, model architecture, representation-alignment objective, and training recipe while changing only the prediction-target parameterization. The baseline uses velocity prediction. The model operates on SD-VAE latents (Rombach et al., 2022) with patch size 2, so $\boldsymbol{W} \in \mathbb{R}^{16\times16}$ acts on each $2\times2\times4$ latent patch. We initialize $\boldsymbol{W} = \boldsymbol{I}$ and use the spectral update with EMA coefficient 10^{-5}.

RAEv2. We also evaluate AnisoFlow on RAEv2, preserving its representation, architecture, and Muon-based training recipe (Jordan et al., 2024; Zhang et al., 2026a). We make two changes to the published recipe, applied to both our RAEv2 baseline and Aniso-RAEv2: timesteps are sampled uniformly before the shift rather than logit-normally, and the inference clamp is set to $\sigma_{\min} = 0.25$. These changes improve the baseline itself (JiT FID $1.06 \to 0.98$), so the comparison remains fair. The baseline uses clean-data prediction. AnisoFlow acts on the 1024-dimensional RAEv2 tokens, so $\boldsymbol{W} \in \mathbb{R}^{1024\times1024}$, with the PCA initialization ($w_i = 0.15\lambda_i$ in the PCA basis of the tokens). Because RAEv2 trains for only 80 epochs, we raise the spectral-update EMA coefficient to 10^{-4} and the noise-weight learning rate to $10\times$ the model learning rate for faster convergence.

B.5 Compute Overhead

AnisoFlow adds D learned directional weights and maintains the learned orthogonal basis $\boldsymbol{B} \in \mathbb{R}^{D\times D}$. During training, the spectral basis update additionally maintains a $D \times D$ running moment matrix and eigendecomposes it at every step to update $\boldsymbol{B}$. At each model evaluation, prediction and velocity recovery require transforms into and out of the learned basis together with elementwise directional operations. The eigendecomposition of the $D \times D$ running moment is performed at every optimization step during training only; at inference time, the learned basis is fixed.

The resulting measured training and inference overheads relative to JiT-H/16 are reported in Table 1.

Table 7: **Additional computation and state introduced by AnisoFlow.**

Component	Additional state / operation	When applied
Directional weights	D scalars	Training and inference
Learned basis	$D \times D$ orthogonal matrix	Training and inference
Running spectral moment	$D \times D$ matrix	Training only
Basis transforms	Multiplication by $\boldsymbol{B}$ and $\boldsymbol{B}^\top$	Each model evaluation
Directional recovery	Elementwise scaling	Each model evaluation
Spectral basis update	Eigendecomposition of a $D \times D$ matrix	Every training step

C Additional Experiments

Effect of the input bottleneck. JiT represents each 16×16 RGB patch as a 768-dimensional pixel vector, which is projected through a 256-dimensional linear patch embedding before entering the transformer. Thus, only a 256-dimensional projection of the original patch is directly available to the network. To test whether the learned target reflects this architectural bottleneck, we learn weights across the full 768-dimensional PCA spectrum. The learned weights follow the near-linear spectral profile of the main experiments through roughly the patch-embedding dimension, but change noticeably beyond this point (Fig. 9). The transition near dimension 256 suggests that the learned target is sensitive to the network's input bottleneck. The zero-skill limit of Appendix A.2 provides a limiting case in which directional information is entirely unavailable.

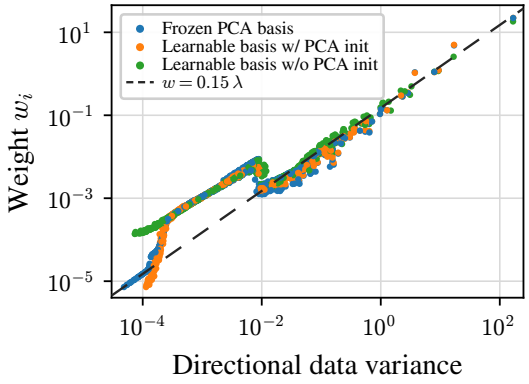
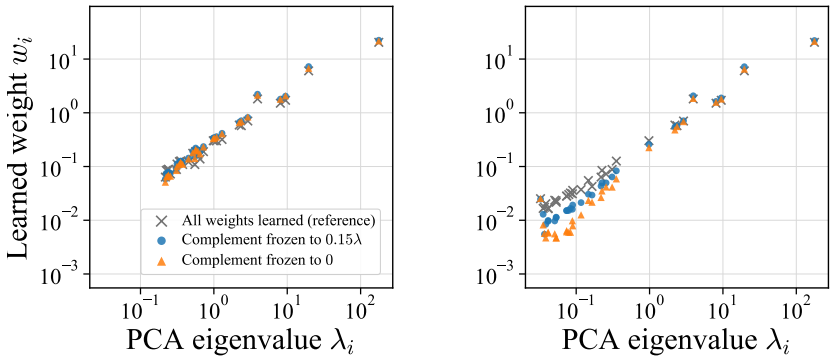


Figure 9: **Effect of the JiT input bottleneck.** Untrimmed version of Fig. 3a, showing all 768 directional variance/weight pairs for the frozen PCA basis and learnable bases with and without PCA initialization. Both learnable variants use spectral updates; the dashed line shows $w = 0.15\lambda$. The near-linear profile at high variance changes noticeably in the low-variance directions beyond roughly the 256-dimensional patch-embedding bottleneck.

Cross-direction coupling. Although our theory analyzes each direction separately, the shared network can couple the preferred weights across directions. Using JiT-B/16 with a fixed PCA basis, we select 32 directions whose weights remain learnable and freeze all other directional weights to a fixed background profile. As shown in Fig. 10, for the highest-variance directions, the free weights remain close to the fully learned solution under both frozen backgrounds, whereas lower-variance directions exhibit somewhat greater sensitivity to the complementary targets. These results support the direction-wise approximation used in our analysis while indicating mild cross-direction coupling in the lowest-variance directions.



(a) Top-32 directions free

(b) Stratified directions free

Figure 10: **Cross-direction coupling.** We leave 32 directional weights free to learn while freezing all complementary directions to either $w = 0$ or $w = 0.15\lambda$. Each point shows the final learned weight of one of these 32 free directions; gray crosses show the corresponding weights from a reference model in which all directions are learned jointly. **(a)** The free subset contains the 32 highest-variance PCA directions. **(b)** The free subset is stratified across the data spectrum. The high-variance free weights closely match the fully learned solution under both frozen backgrounds, while the lowest-variance directions show somewhat greater sensitivity to the complementary targets.