

An Analytical Theory of Spectral Bias in the Learning Dynamics of Diffusion Models

Anonymous Author(s)

Affiliation

Address

email

Abstract

We develop an analytical framework for understanding how the generated distribution evolves during diffusion model training. Leveraging a Gaussian-equivalence principle, we solve the full-batch gradient-flow dynamics of linear and convolutional denoisers and integrate the resulting probability-flow ODE, yielding analytic expressions for the generated distribution. The theory exposes a universal inverse-variance spectral law: the time for an eigen- or Fourier mode to match its target variance scales as $\tau \propto \lambda^{-1}$, so high-variance (coarse) structure is mastered orders of magnitude sooner than low-variance (fine) detail. Extending the analysis to deep linear networks and circulant full-width convolutions shows that weight sharing merely multiplies learning rates—accelerating but not eliminating the bias—whereas local convolution introduces a qualitatively different bias. Experiments on Gaussian and natural-image datasets confirm the spectral law persists in deep MLP-based UNet. Convolutional U-Nets, however, display rapid near-simultaneous emergence of many modes, implicating local convolution in re-shaping learning dynamics. These results underscore how data covariance governs the order and speed with which diffusion models learn, and they call for deeper investigation of the unique inductive biases introduced by local convolution.

1 Introduction

Diffusion models create rich data by gradually transforming Gaussian noise into signal, a paradigm that now drives state-of-the-art generation in vision, audio, and molecular design [1, 2, 3]. Yet two basic questions remain open. (i) Which parts of the data distribution do these models learn first, and which linger unlearned—risking artefacts under early stopping? (ii) How does architectural inductive bias shape this learning trajectory? Addressing both questions demands that we track the evolution of the full generated distribution during training and relate it to the network’s parameterization.

We tackle the learning puzzle through the simplest tractable setting—linear denoisers—where datasets become equivalent to a Gaussian with matched mean and covariance. In this regime we solve, in closed form, the nested dynamics of

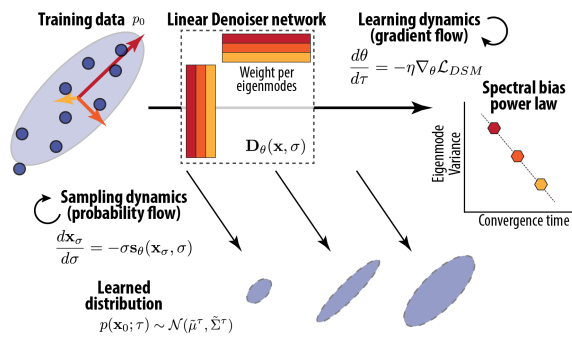


Figure 1: **Spectral-bias schematic.** Learning and sampling together impose a variance-ordered bias along covariance eigenmodes.

37 gradient-flow of the weights and the probability-flow ODE that carries noise into data, leading to an
 38 analytical characterization of the evolution of the generated distribution. The analysis exposes an
 39 inverse-variance spectral law: the time required for an eigen-mode to match target variance scales
 40 like $\tau_k \propto \lambda_k^{-\alpha}$, so high-variance directions corresponding to global structure are mastered orders of
 41 magnitude sooner than low-variance, fine-detail directions. Extending the analysis to deep linear and
 42 linear convolutional nets, we show how convolutional architecture redirect this bias to Fourier or
 43 patch space, and accelerate convergence via weight sharing.

44 **Main contributions** 1. **Closed-form distribution dynamics.** We derive exact weight and distribu-
 45 tional trajectories for one-layer, two-layer linear, and convolutional denoisers under full-batch DSM
 46 training. 2. **Inverse-variance spectral bias.** The theory reveals and quantifies a spectral-law ordering
 47 of mode convergence, offering one mechanistic explanation for early-stop errors. 3. **Empirical**
 48 **validation in nonlinear neural nets.** Experiments on Gaussian and natural-image datasets confirm
 49 the spectral-law in deep MLP-based diffusion. 4. **Convolutional architectural shape learning**
 50 **dynamics.** Experiments on convolutional UNet, showing rapid patch-first learning dynamics different
 51 from fully-connected architectures.

52 2 Related Work and Motivation: Spectral Bias in Distribution Learning

53 **Spectral structure of natural data** Many natural signals have interesting spectral structures (e.g.
 54 image [4], sound [5], video [6]). For natural images, their covariance eigenvalues decay as a power
 55 law, and the corresponding eigenvectors can align with semantically meaningful patterns [4]. For
 56 faces, for instance, leading eigenmodes capture coarse, low-frequency shape variations, whereas
 57 tail modes encode fine-grained textures [7, 8]. Analyzing spectral effect on diffusion learning can
 58 therefore show which type of features the model acquires first and which remain slow to learn.

59 **Hidden Gaussian Structure in Diffusion Model** Recent work has shown, for most diffusion
 60 times, the learned neural score is closely approximated by the linear score of a Gaussian fit to the
 61 data, which is usually the best linear approximation [9, 10]. Crucially, this Gaussian linear score
 62 admits a closed-form solution to the probability-flow ODE, which can be exploited to accelerate
 63 sampling and improve its quality [11]. Moreover, this same linear structure has been linked to
 64 the generalization–memorization transition in diffusion models [10]. In sum, across many noise
 65 levels, the Gaussian linear approximation is a predominant structure in the learned score. Thus, we
 66 hypothesize it will have a significant effect on the learning dynamics of score approximator. From
 67 this perspective, **our contribution** is to elucidate the learning process of this linear structure.

68 **Learning theory for regression and deep linear networks** Gradient dynamics in regression are
 69 well-studied, with spectral bias and implicit regularisation emerging as central themes [12, 13, 14].
 70 In Sec. 4.1, we show that the loss of a linear diffusion model reduces to ridge regression, letting us
 71 import those results directly. Our analysis also builds on learning theory of deep linear networks
 72 (including linear-convolutional and denoising autoencoders) [15, 16, 17, 18]. We extend these insights
 73 to modern diffusion-based generative models, offering closed-form description of how the generated
 74 distribution itself evolves during training.

75 **Diffusion learning theory** Several recent theory studies address diffusion models from a spectral
 76 perspective but tackle different questions. [19, 20, 21, 22] document spectral bias in the *sampling*
 77 process after training; our focus is on how that bias arises during *training*. [23] study stochastic
 78 sampling assuming an optimal score, orthogonal to our analysis of training dynamics. Sharing our
 79 interest in training, [24] analyze learning of mixtures of *spherical* Gaussians to recover component
 80 means, whereas we tackle *anisotropic* covariances and track reconstruction of the full covariance.
 81 [25] characterises optimal score and distribution under constraints; results from our convolutional
 82 setup can be viewed through that lens.

83 3 Background

84 3.1 Score-based Diffusion Models

85 Let $p_0(\mathbf{x})$ be the data distribution of interest, and for each noise level $\sigma > 0$ define $p(\mathbf{x}; \sigma) =$
 86 $(p_0 * \mathcal{N}(0, \sigma^2 \mathbf{I}))(\mathbf{x}) = \int p_0(\mathbf{y}) \mathcal{N}(\mathbf{x} | \mathbf{y}, \sigma^2 \mathbf{I}) d\mathbf{y}$. The associated *score function* is $\nabla_{\mathbf{x}} \log p(\mathbf{x}; \sigma)$,

87 i.e. the gradient of the log-density at noise scale σ . In the EDM framework [26], one shows that the
 88 “probability flow” ODE

$$\frac{d\mathbf{x}}{d\sigma} = -\sigma \nabla_{\mathbf{x}} \log p(\mathbf{x}; \sigma) \quad (1)$$

89 exactly transports samples from $p(\cdot; \sigma_T)$ to $p(\cdot; \sigma)$ as σ decreases. In particular, integrating from
 90 σ_T down to $\sigma = 0$ recovers clean data samples from p_0 . We adopt the EDM parametrization for its
 91 notational simplicity; other common diffusion formalisms are equivalent up to simple rescalings of
 92 space and time [26]. To learn the score of a data distribution $p_0(\mathbf{x})$, we minimize the denoising score
 93 matching (DSM) objective [27] with a function approximator. We reparametrize the score function
 94 with the ‘denoiser’ $s_\theta(\mathbf{x}, \sigma) = (\mathbf{D}_\theta(\mathbf{x}, \sigma) - \mathbf{x})/\sigma^2$, then at noise level σ the DSM objective reads

$$\mathcal{L}_\sigma = \mathbb{E}_{\mathbf{x}_0 \sim p_0, \mathbf{z} \sim \mathcal{N}(0, \mathbf{I})} \left\| \mathbf{D}_\theta(\mathbf{x}_0 + \sigma \mathbf{z}; \sigma) - \mathbf{x}_0 \right\|_2^2. \quad (2)$$

95 To balance the loss and importance of different noise scales, practical diffusion models all adopt
 96 certain weighting functions in their overall loss $\mathcal{L} = \int_\sigma d\sigma w(\sigma) \mathcal{L}_\sigma$.

97 3.2 Gaussian Data and Optimal Denoiser

98 To motivate our linear score approximator set up, it is useful to consider the optimal score and
 99 the denoiser of a Gaussian distribution. For Gaussian data $\mathbf{x}_0 \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, $\mathbf{x}_0 \in \mathbb{R}^d$ and $\boldsymbol{\Sigma}$ is a
 100 positive semi-definite matrix. When noising \mathbf{x}_0 by Gaussian noise at scale σ , the corrupted \mathbf{x} satisfies
 101 $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma} + \sigma^2 \mathbf{I})$, for which the *Bayes-optimal* denoiser is an *affine function* of \mathbf{x} .

$$\mathbf{D}^*(\mathbf{x}; \sigma) = \boldsymbol{\mu} + (\boldsymbol{\Sigma} + \sigma^2 \mathbf{I})^{-1} \boldsymbol{\Sigma} (\mathbf{x} - \boldsymbol{\mu}) \quad (3)$$

102 For Gaussian data, minimizing (2) yields \mathbf{D}^* . This solution has an intuitive interpretation, i.e. the
 103 difference of the state \mathbf{x} and distribution mean was projected onto the eigenbasis and shrunked
 104 mode-by-mode by $\lambda_k/(\lambda_k + \sigma^2)$. Thus, according to the variance λ_k along target axis, modes with
 105 variance significantly higher than noise $\lambda_k \gg \sigma^2$ will be retained; modes with variance much smaller
 106 than noise will be “shrunked” out. Effectively σ^2 defines a threshold of signal and noise, and modes
 107 below which will be removed. This intuition similar to Ridge regression is made exact in Sec. 4.1.

108 4 Learning in Diffusion Models with a Linear Denoiser

109 **Problem set-up.** Throughout the paper, we assume the denoiser at each noise scale is *linear (affine)*
 110 *and independent across scales*:

$$\mathbf{D}(\mathbf{x}; \sigma) = \mathbf{W}_\sigma \mathbf{x} + \mathbf{b}_\sigma. \quad (4)$$

111 Since the parameters $\{\mathbf{W}_\sigma, \mathbf{b}_\sigma\}$ are decoupled across noise scales, each σ can be analysed indepen-
 112 dently. Through further parametrization, this umbrella form captures linear residual nets, deep linear
 113 nets, and linear convolutional nets (see Sec. 5).

114 We train on an arbitrary distribution p_0 with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ by gradient flow on the
 115 *full-batch* DSM loss, i.e. the exact expectation over data *and* noise (2). (In practice, one cannot
 116 sample all \mathbf{z} values, but the full-batch limit yields clean closed-form dynamics.)

117 This setting lets us dissect analytically the role of **data spectrum**, **model architecture** (\mathbf{W}_σ parametri-
 118 sation), and **loss variant** in shaping diffusion learning.

119 4.1 Diffusion learning as ridge regression

120 **Gaussian equivalence.** For any joint distribution $p(X, Y)$ the quadratic loss

$$\mathcal{L}(\mathbf{W}, \mathbf{b}) = \mathbb{E}_{p(X, Y)} \left\| \mathbf{W}X + \mathbf{b} - Y \right\|^2$$

121 depends on p only through the first two moments of (X, Y) ; see App. C.1.1 for proof. Hence a linear
 122 denoiser trained on arbitrary p_0 interacts with the data *solely* via its mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$.

123 **Instance for diffusion.** Under EDM loss (2), the noisy input–target pair is $X = \mathbf{x}_0 + \sigma \mathbf{z}$, $Y = \mathbf{x}_0$,
 124 giving $\Sigma_{XX} = \boldsymbol{\Sigma} + \sigma^2 \mathbf{I}$, $\Sigma_{YX} = \boldsymbol{\Sigma}$.

125 **Gradient and optimum.** Differentiating and setting gradients to zero yields

$$\nabla_{\mathbf{W}_\sigma} \mathcal{L}_\sigma = -2\Sigma + 2\mathbf{W}_\sigma(\Sigma + \sigma^2\mathbf{I}) + \nabla_{\mathbf{b}_\sigma} \mathcal{L}_\sigma \boldsymbol{\mu}^\top, \quad \nabla_{\mathbf{b}_\sigma} \mathcal{L}_\sigma = 2(\mathbf{b}_\sigma - (\mathbf{I} - \mathbf{W}_\sigma)\boldsymbol{\mu}), \quad (5)$$

$$\mathbf{W}_\sigma^* = \Sigma(\Sigma + \sigma^2\mathbf{I})^{-1}, \quad \mathbf{b}_\sigma^* = (\mathbf{I} - \mathbf{W}_\sigma^*)\boldsymbol{\mu}, \quad (6)$$

$$\min \mathcal{L}_\sigma = \sigma^2 \text{Tr}[\Sigma(\Sigma + \sigma^2\mathbf{I})^{-1}].$$

126 Thus the optimal linear denoiser reproduces the denoiser for the Gaussian approximation of data (3),
127 and its best achievable loss is set purely by the data spectrum.

128 **Other objectives.** While the main text focuses on the EDM loss (2), we have worked out the gradients,
129 optima, and learning dynamics for several popular variants used in diffusion and flow-matching [28]
130 literature; these results are summarised in Tab. C.4 (derivations in App. C.4).

131 **Ridge viewpoint.** Because

$$\mathcal{L}_\sigma = \mathbb{E}_{\mathbf{x} \sim p_0} \|\mathbf{W}\mathbf{x} + \mathbf{b} - \mathbf{x}\|^2 + \sigma^2 \|\mathbf{W}\|_F^2,$$

132 full-batch diffusion at noise scale σ is simply auto-encoding with ridge regularisation of strength
133 σ^2 (App. C.2.1; cf. [29]). We will exploit classic ridge-regression results when analyzing learning
134 dynamics in the following sections.

135 4.2 Weight Learning Dynamics of a Linear Denoiser

136 With the gradient structure in hand, we solve the full-batch gradient-flow ODE,

$$\frac{d\mathbf{W}_\sigma}{d\tau} = -\eta \nabla_{\mathbf{W}_\sigma} \mathcal{L}_\sigma, \quad \frac{d\mathbf{b}_\sigma}{d\tau} = -\eta \nabla_{\mathbf{b}_\sigma} \mathcal{L}_\sigma, \quad (\text{GF})$$

137 where τ is training time and η the learning-rate.

138 **Zero-mean data ($\boldsymbol{\mu} = 0$): Exponential convergence mode-by-mode** Because the gradients to
139 \mathbf{W} , \mathbf{b} decouple (5), the dynamics is simplified on the eigenbasis of the covariance. We diagonalize the
140 covariance, $\Sigma = \sum_{k=1}^d \lambda_k \mathbf{u}_k \mathbf{u}_k^\top$, with orthonormal principal components (PC) \mathbf{u}_k and eigenvalues
141 $\lambda_k \geq 0$ (the mode variances). Projecting (GF) onto this basis yields the closed-form solution
142 (derivation in App. D.1):

$$\mathbf{b}_\sigma(\tau) = \mathbf{b}_\sigma(0) e^{-2\eta\tau}, \quad \mathbf{W}_\sigma(\tau) = \mathbf{W}_\sigma^* + \sum_{k=1}^d [\mathbf{W}_\sigma(0) - \mathbf{W}_\sigma^*] \mathbf{u}_k \mathbf{u}_k^\top e^{-2\eta\tau(\sigma^2 + \lambda_k)}. \quad (7)$$

143 **Interpretation.** Each eigenmode projec-
144 tion of the weight $\mathbf{W}_\sigma \mathbf{u}_k$ converges to the
145 optimal value $\mathbf{W}_\sigma^* \mathbf{u}_k$ exponentially with
146 rate $(\sigma^2 + \lambda_k)$; hence (i) the weights at
147 larger noise σ generally converge faster;
148 (ii) at a fixed σ , high-variance λ_k modes
149 converge first, while modes buried beneath
150 the noise floor ($\lambda_k \ll \sigma^2$) share the same
151 slower timescale. Fig. 2A illustrates
152 this spectrum-ordered convergence, with
153 high-variance modes reaching their optima
154 before the low-variance ones (see also 5A).

155 **Non-centred data ($\boldsymbol{\mu} \neq 0$): Interaction
156 of mean and covariance learning.** A
157 non-zero mean introduces a rank-one cou-
158 pling between \mathbf{W} and \mathbf{b} (matrix M in
159 Prop D.2). Eigenmodes of weights over-
160 lapping with the mean ($\mathbf{u}_k^\top \boldsymbol{\mu} \neq 0$) now
161 interact with \mathbf{b} , producing transient over-
162 shoots and other non-monotonic effects;
163 orthogonal modes retain the exponential
164 convergence above. App. D.2 gives the full linear-system analysis and two-dimensional visualisations
165 (Fig. 21).

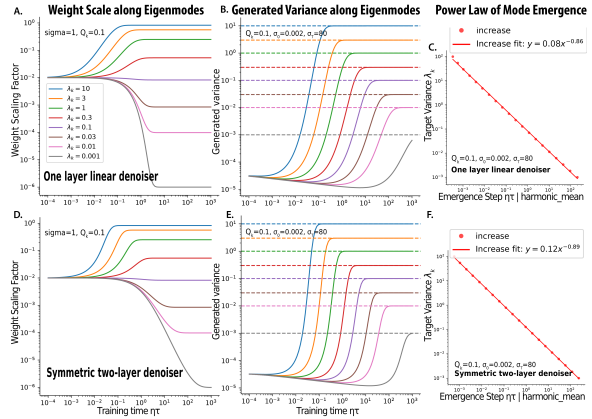


Figure 2: **Learning dynamics per eigenmode.** *Top:* one-layer linear denoiser. *Bottom:* two-layer symmetric denoiser. (A,D) Weight trajectories $\mathbf{u}_k^\top \mathbf{W}_\sigma(\tau) \mathbf{u}_k$ ($\sigma = 1$). (B,E) Generated-variance $\hat{\lambda}_k$ versus target variance λ_k . (C,F) Power-law relation between emergence time τ_k^* and λ_k .

4.3 Sampling Dynamics during Training

For diffusion models, our goal is the generated distribution, obtained by integrating the probability-flow ODE (PF-ODE) backwards from a large σ_T to a $\sigma_{min} \approx 0$,

$$\frac{d\mathbf{x}}{d\sigma} = -\sigma^{-1}[(\mathbf{W}_\sigma - I)\mathbf{x} + \mathbf{b}_\sigma], \quad (\text{PF})$$

initialized with Gaussian noise $\mathbf{x}_T \sim \mathcal{N}(0, \sigma_T^2 \mathbf{I})$. For linear denoiser, the PF-ODE is an inhomogeneous *affine* system, so its solution \mathbf{x}_σ is necessarily an *affine function* of the initial state \mathbf{x}_T [30], $\mathbf{x}(\sigma_0) = A(\sigma_0; \sigma_T) \mathbf{x}(\sigma_T) + c(\sigma_0; \sigma_T)$. Since the map is affine, the distribution of $\mathbf{x}(\sigma_0)$ remains Gaussian, with covariance $\sigma_T^2 A(\sigma_0; \sigma_T) A^\top(\sigma_0; \sigma_T)$.

However, in general, the **state-transition matrix** $A(\sigma_0; \sigma_T)$ is hard to evaluate, as it involves *time-ordered matrix exponential*, and the weight matrices at different noise scales \mathbf{W}_σ may *not commute*. The analysis below—and our closed-form results—hinges on situations where *commutativity* is maintained by *gradient flow or architectural bias*, thus removing the time-ordering operator.

Lemma 4.1 (PF-ODE solution for commuting weights). *If the linear denoiser $\mathbf{D}(\mathbf{x}; \sigma) = \mathbf{W}_\sigma \mathbf{x} + \mathbf{b}_\sigma$ satisfies $[\mathbf{W}_\sigma, \mathbf{W}_{\sigma'}] = 0$ for all σ, σ' , then for any $0 < \sigma_0 < \sigma_T$,*

$$\mathbf{x}(\sigma_0) = A(\sigma_0, \sigma_T) \mathbf{x}(\sigma_T) + c(\sigma_0, \sigma_T), \quad A(\sigma_0, \sigma_T) = \exp\left[-\int_{\sigma_0}^{\sigma_T} \frac{\mathbf{W}_s - \mathbf{I}}{s} ds\right].$$

Interpretation. For each common eigenvector \mathbf{u}_k , the term $(\mathbf{u}_k^\top \mathbf{W}_\sigma \mathbf{u}_k - 1)/\sigma$ is the instantaneous expansion (or contraction) rate of the sample variance along \mathbf{u}_k ; the final variance is obtained by integrating this rate over noise scales σ (see App. C.5).

When does commutativity hold? This arises in three common settings. (i) *At convergence*, this is satisfied by the optimal weights \mathbf{W}_σ^* (6), which jointly diagonalize on eigenbasis of Σ . In such case, we recover the closed-form solution to PF-ODE for Gaussian data, as found by [9, 31]. (ii) *During training of linear denoisers*, if weights are initialized to be aligned with eigenbasis of Σ , then gradient flow keeps them aligned, preserving commutativity (iii) For *linear convolutional denoisers*, circulant weights share the Fourier basis and commute by construction (see Sec. 5.2). In these cases, the sampling process can be understood mode-by-mode. Here we show the explicit solution for one layer linear denoiser.

Proposition 4.2 (Dynamics of generated distribution in one layer case). *Assume (i) zero-mean data, (ii) aligned initialization $\mathbf{W}_\sigma(0) = \sum_k Q_k \mathbf{u}_k \mathbf{u}_k^\top$, and (iii) gradient flow, full-batch training with learning rate η . Then, while training the one-layer linear denoiser, the generated distribution at time τ is $\mathcal{N}(\tilde{\mu}, \tilde{\Sigma})$ with $\tilde{\Sigma} = \sum_k \tilde{\lambda}_k(\tau) \mathbf{u}_k \mathbf{u}_k^\top$ and*

$$\tilde{\lambda}_k(\tau) = \sigma_T^2 \frac{\Phi_k^2(\sigma_0, \tau)}{\Phi_k^2(\sigma_T, \tau)}, \quad \Phi_k(\sigma, \tau) = \sqrt{\lambda_k + \sigma^2} \exp\left[\frac{1-Q_k}{2} \text{Ei}(-2\eta\tau\sigma^2) e^{-2\eta\tau\lambda_k} - \frac{1}{2} \text{Ei}(-2\eta\tau(\sigma^2 + \lambda_k))\right]$$

where Ei is the exponential-integral function. (derivation in App. D.3)

Spectral bias. Figure 2B traces the variance trajectory $\tilde{\lambda}_k(\tau)$ for each eigen-mode. All modes begin with the same initialization-induced level, then follow sigmoidal curves to their targets, but *in descending order of λ_k* . We define the first-passage time τ_k^* as the training time at which $\tilde{\lambda}_k(\tau)$ reaches the geometric (or harmonic) mean of its initial and target values. We find the first-passage time obeys an inverse law $\tau_k^* \propto \lambda_k^{-\alpha}$, $\alpha \approx 1$, (Fig. 2C), which implies that learning a mode with variance 1/10 smaller takes roughly 10 times longer to converge. With larger weight initialization (larger Q_k), the initial variance is closer to the target variance of some modes, then the inverse law splits into separate branches for modes with rising vs. decaying variance (Fig. 5B, Fig. 6).

Practical implication. This suggests when training stops earlier, the distribution in higher variance PC spaces have already converged, while low-variance ones—often the perceptual finer points such as letter strokes or finger joints—are under-trained. This could be an explanation for the familiar “wrong detail” artefacts in diffusion samples.

5 Deep and Convolutional Extensions

After analyzing the simplest linear denoiser, we set out to examine the effect of architectures via different parametrizations of the weights, specifically deeper linear models and linear convolutional networks. In the following, we will assume $\mu = 0$ and focus on learning of covariance.

211 5.1 Deeper linear network

212 Consider a depth- L linear denoiser $\mathbf{D}(\mathbf{x}, \sigma) = \mathbf{W}_L \cdots \mathbf{W}_1 \mathbf{x}$, where—for notational clarity—we
 213 suppress the explicit σ -dependence of weights. We assume **aligned initialization**, where for singular
 214 decomposition of each matrix, $\mathbf{W}_\ell(0) = U_\ell \Lambda_\ell V_\ell^\top$, the right basis of each layer matching the left
 215 basis of the next, $V_{\ell+1} = U_\ell, \forall \ell = 1, \dots, L-1$, and with $U_L = V_1 = U$ where U diagonalizes data
 216 covariance Σ . Then the total weight at initialization is $\mathbf{W}_{tot}(0) = \prod_{\ell=1}^L \mathbf{W}_\ell(0) = U(\prod_{\ell=1}^L \Lambda_\ell)U^\top$.
 217 With aligned initialization, every eigenmode learns independently—mirroring classical results [15,
 218 32]. In our case, this also implies that the total weight $\prod_l \mathbf{W}_l$ shares the eigenbasis U across training
 219 and noise scales, thus commute, making sampling tractable.

220 One especially illuminating case is the two-layer symmetric network, where $\mathbf{D}(\mathbf{x}, \sigma) = P_\sigma P_\sigma^\top \mathbf{x}$.
 221 **Proposition 5.1** (Dynamics of weight and distribution in two layer linear model). *Assume (i) centered*
 222 *data $\mu = 0$; (ii) the weight matrix is initialized aligned, i.e. $P_\sigma(0)P_\sigma(0)^\top = \sum_k Q_k \mathbf{u}_k \mathbf{u}_k^\top$, then the*
 223 *gradient flow ODE admits a closed-form solution (derivation in App. E.1)*

$$\mathbf{W}_\sigma(\tau) = P_\sigma(\tau)P_\sigma(\tau)^\top = \sum_k \frac{\lambda_k}{\sigma^2 + \lambda_k} \mathbf{u}_k \mathbf{u}_k^\top \left(\frac{Q_k}{\left(\frac{\lambda_k}{\sigma^2 + \lambda_k} - Q_k\right)e^{-8\eta\lambda_k\tau} + Q_k} \right) \quad (8)$$

224 The generated distribution at time τ is $\mathcal{N}(\tilde{\mu}, \tilde{\Sigma})$ with $\tilde{\Sigma} = \sum_k \tilde{\lambda}_k(\tau) \mathbf{u}_k \mathbf{u}_k^\top$ and $\tilde{\lambda}_k(\tau) = \sigma^2 \frac{\Phi_k^2(\sigma_0)}{\Phi_k^2(\sigma_\tau)}$

$$\Phi_k(\sigma) = (\sigma) \frac{(1-Q_k)e^{-8\eta\tau\lambda_k}}{Q_k + (1-Q_k)e^{-8\eta\tau\lambda_k}} \left[\lambda_k e^{-8\eta\tau\lambda_k} + Q_k (1 - e^{-8\eta\tau\lambda_k}) (\lambda_k + \sigma^2) \right] \frac{Q_k}{2Q_k + 2(1-Q_k)e^{-8\eta\tau\lambda_k}}$$

225 *Interpretation.* The learning dynamics of weights and variance along different principal components
 226 are visualized in Fig.2 D-F. Compared to one-layer case, here, the weight converges along the PCs
 227 via sigmoidal dynamics, with the emergence time (reaching harmonic mean of initial and final value)
 228 $\tau_k^* = \ln 2 / (8\eta \lambda_k)$. As for generated distribution, we find similar relationship between the target
 229 variance and emergence time $\tau_k^* \propto \lambda_k^{-\alpha}$, $\alpha \approx 1$. For the more general non-aligned initialization, we
 230 show the non-aligned parts of weight will follow non-monotonic rise-and-fall dynamics (App. E.1.2).
 231 Extensions to non-symmetric two layer model and deeper model were studied in App. F, which have
 232 similar bias but lack clean expressions.

233 5.2 Linear convolutional network

234 We consider a linear denoiser with convolutional architecture, $\mathbf{D}(\mathbf{x}, \sigma) = \mathbf{w}_\sigma * \mathbf{x}$ where samples
 235 $\mathbf{x} \in \mathbb{R}^N$ have 1d spatial structure, and a width K convolution filter \mathbf{w}_σ operates on it. The analysis
 236 could be easily generalized to 2d convolution. With circular boundary condition, \mathbf{w}_σ defines a
 237 circulant weight matrix $\mathbf{W}_\sigma \in \mathbb{R}^{N \times N}$, where $\mathbf{w}_\sigma * \mathbf{x} = \mathbf{W}_\sigma \mathbf{x}$. One favorable property of circulant
 238 matrices is that they are diagonalized by *discrete Fourier transform* F [33].

$$\mathbf{W}_\sigma = F \Gamma_\sigma F^* \quad F_{mk} := \frac{1}{\sqrt{N}} \exp \left(-2\pi i \frac{mk}{N} \right) \quad (9)$$

239 Thus all weights \mathbf{W}_σ commutes, which allows us to leverage Lemma 4.1, and solve the sampling
 240 dynamics mode-by-mode on the Fourier basis, leading to following result.

241 **Proposition 5.2.** *Linear convolutional denoisers with circular boundary can only model stationary*
 242 *Gaussian processes (GP), with independent Fourier modes, proof in App.G.2.*

243 **Learning dynamics of full-width filter $K = N$** When convolution filter \mathbf{w}_σ is as large as the
 244 signal, the gradient flow is diagonal and unconstrained in the Fourier domain. Thus, the analyses in
 245 Sec. 4 re-emerge with variance of Fourier mode $\tilde{\Sigma}_{kk}$ taking the place of λ_k .

246 **Proposition 5.3** (Full-width circular convolution learning dynamics). *Let $\mathbf{D}(\mathbf{x}, \sigma) = \mathbf{w}_\sigma * \mathbf{x}$, with*
 247 *full-width filter $K = N$, and train \mathbf{w}_σ by full-batch gradient flow at rate η . Then the weights at noise*
 248 *σ and its spectral representation γ evolves as*

$$\mathbf{w}_\sigma(\tau) = \frac{1}{\sqrt{N}} F^* \gamma(\tau, \sigma) \quad ; \quad \gamma_k(\tau, \sigma) = \gamma_k^*(\sigma) + (\gamma_k(\tau, \sigma) - \gamma_k^*(\sigma)) e^{-2N\eta(\sigma^2 + \tilde{\Sigma}_{kk})\tau} \quad (10)$$

249 where $\gamma_k^*(\sigma) = \tilde{\Sigma}_{kk} / (\sigma^2 + \tilde{\Sigma}_{kk})$ and $\tilde{\Sigma}_{kk} = [F^* \Sigma F]_{kk}$ is the variance of Fourier mode.

250 The generated distribution has diagonal covariance in the Fourier basis and follows exactly Prop. 4.2
 251 after the replacement $\lambda_k \rightarrow \tilde{\Sigma}_{kk}, \eta \rightarrow N\eta, U \rightarrow F$. (derivation in App. G.3)

Table 1: **Summary of theory.** exp. and sigm. denotes exponential and sigmoidal convergence. xN denotes the N time speed up due to weight sharing.

	One layer	Sym. two layer	Full-width linear conv	Patch linear conv
Param.	\mathbf{W}	PP^T	$\mathbf{W} = \text{Circ}(\mathbf{w})$	$\mathbf{W} = \text{Circ}(\mathbf{w})$, $K < N$
Weight dyn.	PC, exp. (7)	PC, sigm. 5.1	Fourier mode, exp. xN 5.3	PC of patches, exp. xN 5.4
Learned distr.	Gaussian	Gaussian	Stationary GP 5.2	Stationary GP
Distr. dyn.	PC, sigm., power law 4.2	PC, sigm., power law 5.1	Fourier, sigm., xN, power law 5.3	Fourier, N.S.

Interpretation. The weight and distribution dynamics mirror the fully-connected case, with spectral bias towards higher variance *Fourier modes*; convolutional weight sharing simply multiplies every rate by N , accelerating convergence without altering the inverse-variance law.

Notably, the learned distribution is asymptotically equivalent to the *Gaussian approximation to the original training data with all possible spatial shifts* as augmentations (proof in App. G.3.2). This is one case where *equivariant architectural constraints facilitates creativity* as discussed in [25]. Similarly, two-layer linear conv net with full-width filter can be treated as in Sec.5.1.

Learning dynamics of local filter $K < N$ When the convolution filter has a limited bandwidth $K \neq N$, the Fourier domain dynamics get constrained, so it is easier to work with the filter weights. Let r be the half-width of the kernel ($K = 2r + 1$). Define the circular patch extractor $\mathcal{P}_r(\mathbf{x}) = [\mathbf{x}_{i-r:i+r}]_{i=1}^N \in \mathbb{R}^{K \times N}$, and the patch covariance $\Sigma_{\text{patch}} = \frac{1}{N} \mathbb{E}_{\mathbf{x}}[\mathcal{P}_r(\mathbf{x}) \mathcal{P}_r(\mathbf{x})^T] \in \mathbb{R}^{K \times K}$.

Proposition 5.4 (Patch-convolution learning dynamics). *For the circular convolutional denoiser, $\mathbf{D}(\mathbf{x}, \sigma) = \mathbf{w}_\sigma * \mathbf{x}$ trained by full-batch gradient flow with step size η . Let $\mathbf{e}_0 \in \mathbb{R}^K$ be the one-hot vector with a single 1 at the center position $r + 1$ (1-indexed). (derivation in App. G.4)*

$$\mathbf{w}_\sigma(\tau) = \mathbf{w}_\sigma^* + \exp[-2N\eta\tau(\sigma^2 I + \Sigma_{\text{patch}})] (\mathbf{w}_\sigma(0) - \mathbf{w}_\sigma^*), \quad \mathbf{w}_\sigma^* = (\sigma^2 I + \Sigma_{\text{patch}})^{-1} \Sigma_{\text{patch}} \mathbf{e}_0.$$

Interpretation. Training with a narrow convolutional filter reduces to ridge regression in patch space. Under gradient flow, filter converges along eigenmodes of patch Σ_{patch} : modes with larger variance converges sooner, those with smaller variance later, preserving the inverse-variance law. It also enjoys the N times speed up given by weight sharing, accelerating progress without altering the ordering. The sampling ODE remains diagonal in Fourier space, so the generated distribution will be a stationary Gaussian process with local covariance structure shaped by the learned patch denoiser, though its exact form needs numerical integration to spell out. This setting is similar to the *equivariant and local score machine* described in [25].

6 Empirical Validation of the Theory in Practical Diffusion Model Training

General Approach To test our theoretical predictions about the evolution of generated distribution (esp. covariance), we resort to the following method: 1) we fix a training dataset $\{\mathbf{x}_i\}$ and compute its empirical mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$. We then perform an eigen-decomposition of $\boldsymbol{\Sigma}$, obtaining eigenvalues λ_k and eigenvectors \mathbf{u}_k . 2) Next, we train a diffusion model on this dataset by optimizing the DSM objective with a neural network denoiser $\mathbf{D}_\theta(\mathbf{x}, \sigma)$. 3) During training, at certain steps τ , we generate samples $\{\mathbf{x}_i^\tau\}$ from the diffusion model by integrating the PF-ODE (1). We then estimate the sample mean $\tilde{\boldsymbol{\mu}}^\tau$ and sample covariance $\tilde{\boldsymbol{\Sigma}}^\tau$. Finally, we compute the variance of the generated samples along the eigenbasis of training data, $\tilde{\lambda}_k^\tau = \mathbf{u}_k^T \tilde{\boldsymbol{\Sigma}}^\tau \mathbf{u}_k$. To stress test our theory and maximize its relevance, we'd keep most of the training hyperparameters as practical ones.

6.1 Multi-Layer Perceptron (MLP)

To test our theory about *linear and deep linear network* (Prop.4.2,5.1), we used a Multi-Layer Perceptron (MLP) inspired by the SongUnet in EDM [26, 34] (details in App. I.2). We found this architecture effective in learning distribution like point cloud data (Fig. 23). We kept the preconditioning, loss weighting and initialization the same as in [26].

Experiment 1: Zero-mean Gaussian Data x MLP We first consider a zero mean Gaussian $\mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ in d dimension as training distribution, with covariance defined as a randomly rotated diagonal matrix with log normal spectrum (details in App. I.4). During training, the generated variance of each eigenmode follows a sigmoidal trajectory toward its target value λ_k ; modes with larger λ_k cross the plateau sooner (Fig.9A). We mark the emergence time τ^* as the step at which the variance reaches the geometric mean of its initial and asymptotic values (Fig. 9B). Across both

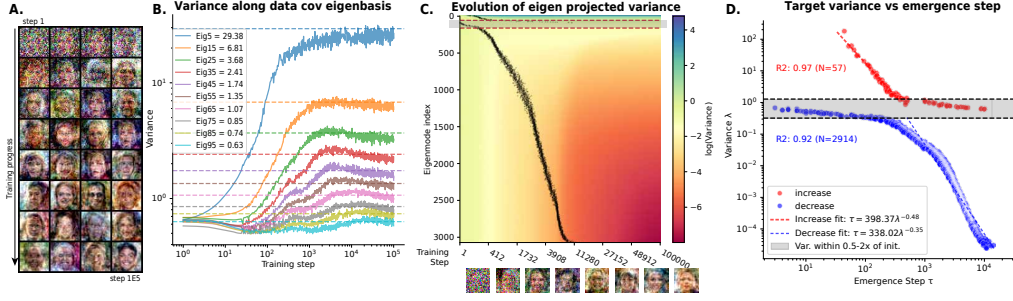


Figure 3: **Spectral Learning Dynamics of MLP-UNet (FFHQ32).** **A.** Generated samples during training. **B.** Evolution of sample variance $\tilde{\lambda}_k(\tau)$ across eigenmodes during training. **C.** Heatmap of variance trajectories along all eigenmodes, with dots marking mode emergence times τ^* (first-passage time at the geometric mean of initial and final variances). The **gray zone** ($0.5\text{--}2\times$ target variance) indicates modes starting too close to their target, causing unreliable τ^* estimates. **D.** Power-law scaling of τ^* versus target variance λ_k . A separate law was fit for modes with **increasing** and **decreasing** variance, excluding the middle gray-zone eigenmodes for stability.

high- and low-variance modes, τ^* obeys an inverse power-law, $\tau^* \propto \lambda_k^{-\alpha}$. With higher-dimensional Gaussians the exponent is estimated more precisely and remains close to 1: for $d=256$, $\alpha = 1.08$; for $d=512$, $\alpha_{\text{incr}} = 1.05$ and $\alpha_{\text{decr}} = 1.13$ (Fig. 9C). The scaling breaks down only for modes whose initial variance is already near λ_k ; in that regime the trajectory is less sigmoidal and τ^* becomes ill-defined. This result shows that despite many non-idealistic conditions e.g. *deeper network, nonlinear activation function, residual connections, normal weights initialization, shared parametrization of denoisers at different noise level*, the prediction from the linear network theory is still *quantitatively* correct.

Experiment 2: Natural Image Datasets x MLP

Next, we validated our theory on natural image datasets. We flattened the images as a vectors, and trained a deeper and wider MLP-UNet to learn the distribution. Using FFHQ as our running example, monitoring the generated samples throughout training (Fig. 3A), despite heavy noise early on, the coarse facial contours—corresponding to the mean and top principal components of human face distribution [7]—emerge quickly, whereas high-frequency details (lower PCs) only appear later. We note that this spectral ordering effect of training dynamics is reminiscent and similar to that in the sampling dynamics after training [19, 22].

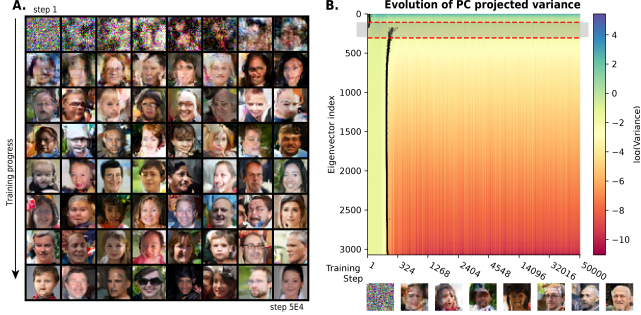


Figure 4: **Learning dynamics of UNet differs | FFHQ32.** **A.** Sample trajectory from CNN-UNet. **B.** Variance evolution along covariance eigenmodes. (c.f. Fig. 3A.C.)

Quantitatively, the sample covariance $\tilde{\Sigma}^\tau$ rapidly aligns with and becomes close to diagonal in the data eigenbasis U (Fig. 10). The top eigenmodes’ variances, $\tilde{\lambda}_k(\tau)$, follow sigmoidal trajectories converging to their targets, and their “emergence times” τ_k^* increase down the spectrum (Fig. 3B,C). We exclude a **central band** of modes whose initial variances lie within $0.5\text{--}2\times$ the target, since their undulating learning dynamics make first-passage time estimates unreliable. After this exclusion, modes with **increasing** and **decreasing** variance each exhibit a clear power-law scaling between emergence step τ^* and target variance λ_k , with exponents -0.48 ($R^2 = 0.97$, $N = 57$) and -0.35 ($R^2 = 0.92$, $N = 2,914$), respectively (Fig. 3D). Although the observed spectral bias is slightly attenuated relative to the Gaussian case and linear theory prediction, it remains robust and consistent across datasets (MNIST, CIFAR-10, FFHQ32 and AFHQ32) (App. B.2.2). This shows that even with natural image data, the distributional learning dynamics of MLP-based diffusion still suffers from slower convergence speed for lower eigenmodes.

6.2 Convolutional Neural Networks (CNNs)

Next we turn to the convolutional U-Net—the work-horse of image-diffusion models [34, 35]. For a full-width linear convolutional network our analysis predicts an inverse-variance law in Fourier space (Prop. 5.3). The patch-convolution variant lacks a clear forecast on distribution, so the following experiments probe empirically whether—and how—its learning dynamics is affected by spectral bias.

Experiment 3: Natural Image Datasets x CNN UNet Training on the same FFHQ dataset, the distributional learning trajectory of CNN-UNet is *markedly different* from the MLPs: early in training, we do not see contour of face, but locally coherent patches, reminiscent of Ising models (Fig. 4A.). Visually and variance-wise, the CNN-based UNet converge much faster and better than the MLP-based UNet, matching the N-fold speed-up from weight sharing (Prop. 5.4; Fig. 4B). When projecting onto the data eigenbasis, all eigenmodes with increasing variance rise simultaneously, while eigenmodes with decreasing variance co-decay at a later time, giving an effective power-law exponent $\alpha \approx 0$; Thus, spectral bias is essentially absent (Fig. 12C.D.).

The likely cause is locality: local convolutional filters couple neighbouring pixels, binding many Fourier modes into one learning unit. Sampling remains diagonal in Fourier space, so a broad band of modes is amplified simultaneously. The early CNN denoiser is indeed well-approximated by a local linear filter (Fig. 14). A full analytic treatment of convolutional U-Net training dynamics is deferred to future work.

7 Discussion

In summary, we presented closed-form solutions for training denoisers with linear, deep linear or linear convolutional architecture, under the DSM objective on arbitrary data. This setup allows for a precise *mode-wise understanding* of the gradient flow dynamics of the denoiser and the evolution of the learned distribution: covariance eigenmode for deep linear network and Fourier mode for convolutional networks. For both the weights and the distribution, we showed analytical evidence of *spectral bias*, i.e. weights converge faster along the eigenmodes or Fourier modes with high variance, and the learned distribution recovers the true variance first along the top eigenmodes. These theoretical results are summarized in Tab. 1.

We hope these results can serve as a solvable model for spectral bias in the diffusion models through the nested training and sampling dynamics. Furthermore, our analysis is not limited to the diffusion and the DSM loss, in App. H, we showed a similar derivation for the spectral bias in flow matching models [28, 36].

Relevance of our theoretical assumptions We found, for the purpose of analytical tractability, we made many idealistic assumptions about neural network training, 1) linear neural network, 2) small or orthogonal weight initialization, 3) “full-batch” gradient flow, 4) independent evolution of weights at each noise scale. In our MLP experiments, we found even when all of these assumptions were somewhat violated, the general theoretical prediction is correct, with modified power coefficients. This shows most of these assumptions could be relaxed in real life, and the spectrum of data indeed have a large effect on the learning dynamics, esp. for fully connected networks.

Inductive bias of the local convolution In our CNN experiments, however, the theoretical predictions from linear models deviate: the spectral bias in learning speed does not directly apply to the distribution of full images. Although our theory predicts that filter-weight learning dynamics are governed by the patch covariance, the ultimate image distribution is shaped by the convolution of those filters. To date, many learning-theory analyses for diffusion models assume MLP-like architectures [24]. For future theoretical work on the learning dynamics of practical diffusion models, a rigorous treatment of the local convolutional structure—and its frequency-coupling effects—will likely be essential, rather than relying on full-width convolution analyses [37].

Broader Impact Although our work is primarily theoretical, the inverse scaling law could offer valuable insights into how to improve the training of large-scale diffusion or flow generative models.

References

- [1] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015.
- [2] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.
- [3] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.
- [4] Antonio Torralba and Aude Oliva. Statistics of natural image categories. *Network: computation in neural systems*, 14(3):391, 2003.
- [5] Hagai Attias and Christoph Schreiner. Temporal low-order statistics of natural sounds. *Advances in neural information processing systems*, 9, 1996.
- [6] Dawei W Dong and Joseph J Atick. Statistics of natural time-varying images. *Network: computation in neural systems*, 6(3):345, 1995.
- [7] Matthew Turk and Alex Pentland. Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1):71–86, 1991.
- [8] Binxu Wang and Carlos R Ponce. A geometric analysis of deep generative image models and its applications. In *International Conference on Learning Representations*, 2021. URL <https://openreview.net/forum?id=GH7QRzUDdXG>.
- [9] Binxu Wang and John J. Vastola. The Hidden Linear Structure in Score-Based Models and its Application. *arXiv e-prints*, art. arXiv:2311.10892, November 2023. doi: 10.48550/arXiv.2311.10892.
- [10] Xiang Li, Yixiang Dai, and Qing Qu. Understanding generalizability of diffusion models requires rethinking the hidden gaussian structure. *arXiv preprint arXiv:2410.24060*, 2024.
- [11] Binxu Wang and John Vastola. The unreasonable effectiveness of gaussian score approximation for diffusion models and its applications. *Transactions on Machine Learning Research*, December 2024. arXiv preprint arXiv:2412.09726.
- [12] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International Conference on Machine Learning*, pages 5301–5310. PMLR, 2019.
- [13] Blake Bordelon, Abdulkadir Canatar, and Cengiz Pehlevan. Spectrum dependent learning curves in kernel regression and wide neural networks. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1024–1034. PMLR, 13–18 Jul 2020. URL <https://proceedings.mlr.press/v119/bordelon20a.html>.
- [14] Abdulkadir Canatar, Blake Bordelon, and Cengiz Pehlevan. Spectral bias and task-model alignment explain generalization in kernel regression and infinitely wide neural networks. *Nature Communications*, 12(1):2914, May 2021. ISSN 2041-1723. doi: 10.1038/s41467-021-23103-1. URL <https://doi.org/10.1038/s41467-021-23103-1>.
- [15] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.
- [16] Suriya Gunasekar, Jason D Lee, Daniel Soudry, and Nati Srebro. Implicit bias of gradient descent on linear convolutional networks. *Advances in neural information processing systems*, 31, 2018.
- [17] Arnau Pretorius, Steve Kroon, and Herman Kamper. Learning dynamics of linear denoising autoencoders. In *International Conference on Machine Learning*, pages 4141–4150. PMLR, 2018.

- [18] Chulhee Yun, Shankar Krishnan, and Hossein Mobahi. A unifying view on implicit bias in training linear neural networks. *arXiv preprint arXiv:2010.02501*, 2020.
- [19] Sander Dieleman. Diffusion is spectral autoregression, 2024. URL <https://sander.ai/2024/09/02/spectral-autoregression.html>.
- [20] Severi Rissanen, Markus Heinonen, and Arno Solin. Generative modelling with inverse heat dissipation. *arXiv preprint arXiv:2206.13397*, 2022.
- [21] Florentin Guth, Simon Coste, Valentin De Bortoli, and Stephane Mallat. Wavelet score-based generative modeling. *Advances in neural information processing systems*, 35:478–491, 2022.
- [22] Binxu Wang and John J Vastola. Diffusion models generate images like painters: an analytical theory of outline first, details later. *arXiv preprint arXiv:2303.02490*, 2023.
- [23] Giulio Biroli, Tony Bonnaire, Valentin De Bortoli, and Marc Mézard. Dynamical regimes of diffusion models. *Nature Communications*, 15(1):9957, 2024.
- [24] Kulin Shah, Sitan Chen, and Adam Klivans. Learning mixtures of gaussians using the ddpm objective. *Advances in Neural Information Processing Systems*, 36:19636–19649, 2023.
- [25] Mason Kamb and Surya Ganguli. An analytic theory of creativity in convolutional diffusion models. *arXiv preprint arXiv:2412.20292*, 2024.
- [26] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *arXiv preprint arXiv:2206.00364*, 2022.
- [27] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7):1661–1674, 2011.
- [28] Yaron Lipman, Marton Havasi, Peter Holderrieth, Neta Shaul, Matt Le, Brian Karrer, Ricky TQ Chen, David Lopez-Paz, Heli Ben-Hamu, and Itai Gat. Flow matching guide and code. *arXiv preprint arXiv:2412.06264*, 2024.
- [29] Chris M Bishop. Training with noise is equivalent to tikhonov regularization. *Neural computation*, 7(1):108–116, 1995.
- [30] Philip Hartman. *Ordinary differential equations*. SIAM, 2002.
- [31] Emile Pierret and Bruno Galerne. Diffusion models for gaussian distributions: Exact solutions and wasserstein errors. *arXiv preprint arXiv:2405.14250*, 2024.
- [32] Kenji Fukumizu. Effect of batch learning in multilayer neural networks. *Gen*, 1(04):1E–03, 1998.
- [33] Robert M Gray et al. Toeplitz and circulant matrices: A review. *Foundations and Trends® in Communications and Information Theory*, 2(3):155–239, 2006.
- [34] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL <https://openreview.net/forum?id=PxtIG12RRHS>.
- [35] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention—MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.
- [36] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.
- [37] Suriya Gunasekar, Jason D Lee, Daniel Soudry, and Nati Srebro. Implicit bias of gradient descent on linear convolutional networks. *Advances in neural information processing systems*, 31, 2018.

- [38] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.
- [39] Marvin Li, Aayush Karan, and Sitan Chen. Blink of an eye: a simple theory for feature localization in generative models. *arXiv preprint arXiv:2502.00921*, 2025.
- [40] Minyoung Huh, Hossein Mobahi, Richard Zhang, Brian Cheung, Pulkit Agrawal, and Phillip Isola. The low-rank simplicity bias in deep networks. *arXiv preprint arXiv:2103.10427*, 2021.
- [41] Li Jing, Jure Zbontar, et al. Implicit rank-minimizing autoencoder. *Advances in Neural Information Processing Systems*, 33:14736–14746, 2020.
- [42] Shuxuan Guo, Jose M Alvarez, and Mathieu Salzmann. Expandnets: Linear over-parameterization to train compact convolutional networks. *Advances in Neural Information Processing Systems*, 33:1298–1310, 2020.
- [43] Meena Jagadeesan, Ilya Razenshteyn, and Suriya Gunasekar. Inductive bias of multi-channel linear convolutional networks with bounded weight norm. In *Conference on Learning Theory*, pages 2276–2325. PMLR, 2022.
- [44] Kathlén Kohn, Thomas Merkh, Guido Montúfar, and Matthew Trager. Geometry of linear convolutional networks. *SIAM Journal on Applied Algebra and Geometry*, 6(3):368–406, 2022.
- [45] Kathlén Kohn, Guido Montúfar, Vahid Shahverdi, and Matthew Trager. Function space and critical points of linear convolutional networks. *SIAM Journal on Applied Algebra and Geometry*, 8(2):333–362, 2024.
- [46] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9446–9454, 2018.
- [47] Prithvijit Chakrabarty and Subhansu Maji. The spectral bias of the deep image prior. *arXiv preprint arXiv:1912.08905*, 2019.
- [48] Zezhou Cheng, Matheus Gadelha, Subhansu Maji, and Daniel Sheldon. A bayesian perspective on the deep image prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5443–5451, 2019.
- [49] Basri Ronen, David Jacobs, Yoni Kasten, and Shira Kritchman. The convergence rate of neural networks for learned functions of different frequencies. *Advances in Neural Information Processing Systems*, 32, 2019.
- [50] Ronen Basri, Meirav Galun, Amnon Geifman, David Jacobs, Yoni Kasten, and Shira Kritchman. Frequency bias in neural networks for input of non-uniform density. In *International conference on machine learning*, pages 685–694. PMLR, 2020.
- [51] Mingchen Li, Mahdi Soltanolkotabi, and Samet Oymak. Gradient descent with early stopping is provably robust to label noise for overparameterized neural networks. In *International conference on artificial intelligence and statistics*, pages 4313–4324. PMLR, 2020.
- [52] Reinhard Heckel and Mahdi Soltanolkotabi. Denoising and regularization via exploiting the structural bias of convolutional generators. *arXiv preprint arXiv:1910.14634*, 2019.
- [53] James R Bunch, Christopher P Nielsen, and Danny C Sorensen. Rank-one modification of the symmetric eigenproblem. *Numerische Mathematik*, 31(1):31–48, 1978.
- [54] Ming Gu and Stanley C Eisenstat. A stable and efficient algorithm for the rank-one modification of the symmetric eigenproblem. *SIAM journal on Matrix Analysis and Applications*, 15(4): 1266–1276, 1994.
- [55] Gene H Golub. Some modified matrix eigenvalue problems. *SIAM review*, 15(2):318–334, 1973.
- [56] Gilbert Strang. A proposal for Toeplitz matrix calculations. *Studies in Applied Mathematics*, 74 (2):171–176, 1986.

- 518 [57] Mingkui Chen. On the solution of circulant linear system. Technical Report TR-401, Yale
519 University, Department of Computer Science, 1985. URL [https://cpsc.yale.edu/sites/
520 default/files/files/tr401.pdf](https://cpsc.yale.edu/sites/default/files/files/tr401.pdf).
- 521 [58] Michela Mastronardi. Presentation slides (mastronardi.pdf). [https://www.math.unipd.it/
522 ~michela/2gg07/TALKS/Mastronardi.pdf](https://www.math.unipd.it/~michela/2gg07/TALKS/Mastronardi.pdf), 2007. Accessed: 2025-05-23.
- 523 [59] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data
524 distribution. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and
525 R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Cur-
526 ran Associates, Inc., 2019. URL [https://proceedings.neurips.cc/paper/2019/file/
527 3001ef257407d5a371a96dcd947c7d93-Paper.pdf](https://proceedings.neurips.cc/paper/2019/file/3001ef257407d5a371a96dcd947c7d93-Paper.pdf).

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope?

Answer: [\[Yes\]](#)

Justification: The abstract and introduction clearly state three contributions—closed-form linear diffusion theory, the inverse-variance (“spectral”) bias, and empirical verification in MLPs and CNNs—which are exactly what the body of the paper and appendices deliver.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: In discussion (paragraph “Relevance of our theoretical assumptions”) we explicitly listed idealised assumptions—linearity, full-batch gradient flow, aligned initialization and discussed how much they affects applicability to practical training of diffusion. The main theory results still hold even when all these assumptions were violated for MLPs.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [\[Yes\]](#)

Justification: Assumptions are stated at the start of every proposition/lemma (e.g. Lemma 4.1, Proposition 4.2); full proofs are given in the referenced appendices, which are submitted separately at the supplementary material deadline.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [\[Yes\]](#)

Justification: The appendix Sec.I has sufficient information of the model architecture, training details and hyperparameters to reproduce our experimental results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [\[Yes\]](#)

Justification: All the training datasets used (MNIST, CIFAR, FFHQ, AFHQ) are open datasets available for downloading. The key MLP-UNet architectures and its torch implementations code are provided in App. I.2, full experiment code and analysis code will be shared after the paper gets accepted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [\[Yes\]](#)

Justification: Appendix Sec.I supplied sufficient details for model training and evaluation.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [\[Yes\]](#)

576 Justification: For quantification of spectral law in experiments, we quantify the statistical
 577 power of law with linear regression line fitting and reported the R2 as goodness of fit Fig. 3,
 578 ??.

579 **8. Experiments compute resources**

580 Question: For each experiment, does the paper provide sufficient information on the com-
 581 puter resources (type of compute workers, memory, time of execution) needed to reproduce
 582 the experiments?

583 Answer: [Yes]

584 Justification: Details about required computational resources are provided in appendix I.1.

585 **9. Code of ethics**

586 Question: Does the research conducted in the paper conform, in every respect, with the
 587 NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

588 Answer: [Yes]

589 Justification: The work uses only publicly available numeric/image data, involves no per-
 590 sonal or sensitive information, and follows standard ML ethical guidelines.

591 **10. Broader impacts**

592 Question: Does the paper discuss both potential positive societal impacts and negative
 593 societal impacts of the work performed?

594 Answer: [Yes]

595 Justification: The final paragraph of the paper (“Broader Impact”) outlines how understand-
 596 ing spectral bias could improve training efficiency while acknowledging possible misuse for
 597 more realistic deep-fake generation.

598 **11. Safeguards**

599 Question: Does the paper describe safeguards that have been put in place for responsible
 600 release of data or models that have a high risk for misuse (e.g., pretrained language models,
 601 image generators, or scraped datasets)?

602 Answer: [NA]

603 Justification: No high-risk models or new datasets are released; experiments use small-scale
 604 public data only.

605 **12. Licenses for existing assets**

606 Question: Are the creators or original owners of assets (e.g., code, data, models), used in
 607 the paper, properly credited and are the license and terms of use explicitly mentioned and
 608 properly respected?

609 Answer: [Yes]

610 Justification: MNIST is cited with its BSD-style licence; EDM implementation and the
 611 related datasets (CIFAR10, FFHQ, AFHQ) are acknowledged and used under its Apache-2.0
 612 licence (Appendix A.4).

613 **13. New assets**

614 Question: Are new assets introduced in the paper well documented and is the documentation
 615 provided alongside the assets?

616 Answer: [NA]

617 Justification: The paper introduces no new datasets or pretrained models.

618 **14. Crowdsourcing and research with human subjects**

619 Question: For crowdsourcing experiments and research with human subjects, does the paper
 620 include the full text of instructions given to participants and screenshots, if applicable, as
 621 well as details about compensation (if any)?

622 Answer: [NA]

623 Justification: No human-subject data or crowdsourcing was involved.

624 **15. Institutional review board (IRB) approvals or equivalent for research with human**
625 **subjects**

626 Question: Does the paper describe potential risks incurred by study participants, whether
627 such risks were disclosed to the subjects, and whether Institutional Review Board (IRB)
628 approvals (or an equivalent approval/review based on the requirements of your country or
629 institution) were obtained?

630 Answer: [NA]

631 Justification: Not applicable—there are no human subjects.

632 **16. Declaration of LLM usage**

633 Question: Does the paper describe the usage of LLMs if it is an important, original, or
634 non-standard component of the core methods in this research? Note that if the LLM is used
635 only for writing, editing, or formatting purposes and does not impact the core methodology,
636 scientific rigor, or originality of the research, declaration is not required.

637 Answer: [NA]

638 Justification: No LLMs were used beyond standard proofreading assistance and math
639 checking, which does not affect the scientific content.

640 Appendix

641 Contents

642	1 Introduction	1
643	2 Related Work and Motivation: Spectral Bias in Distribution Learning	2
644	3 Background	2
645	3.1 Score-based Diffusion Models	2
646	3.2 Gaussian Data and Optimal Denoiser	3
647	4 Learning in Diffusion Models with a Linear Denoiser	3
648	4.1 Diffusion learning as ridge regression	3
649	4.2 Weight Learning Dynamics of a Linear Denoiser	4
650	4.3 Sampling Dynamics during Training	5
651	5 Deep and Convolutional Extensions	5
652	5.1 Deeper linear network	6
653	5.2 Linear convolutional network	6
654	6 Empirical Validation of the Theory in Practical Diffusion Model Training	7
655	6.1 Multi-Layer Perceptron (MLP)	7
656	6.2 Convolutional Neural Networks (CNNs)	9
657	7 Discussion	9
658	A Extended Related works	20
659	B Extended Results	22
660	B.1 Extended Visualization of Theoretical Results	22
661	B.1.1 Scaling curves of diffusion training (EDM)	22
662	B.1.2 Scaling curves of flow matching	24
663	B.2 Extended Empirical Results	25
664	B.2.1 MLP-UNet Gaussian training experiments	25
665	B.2.2 MLP-UNet natural image training experiments	25
666	B.2.3 CNN-UNet training experiment	28
667	C Detailed Derivation: General analysis	35
668	C.1 General Property of Linear Regression	35
669	C.1.1 Gaussian equivalence	35
670	C.2 General analysis of the gradient learning of linear predictor	36
671	C.2.1 Denoising as Ridge Regression	36
672	C.3 General Analysis of the Denoising Score Matching Objective	37

673	C.4	Gradient Structure of Other Variants of Loss	39
674	C.5	General Analysis of the Sampling ODE	42
675	C.6	KL Divergence Computation	43
676	D	Detailed Derivations for the One-Layer Linear Model	45
677	D.1	Zero-mean data: Exponential Converging Training Dynamics	45
678	D.1.1	Discrete time Gradient descent dynamics	46
679	D.1.2	Special parametrization: residual connection	47
680	D.2	General non-centered distribution: Interaction of mean and covariance learning . .	48
681	D.2.1	Special case: low dimensional interaction of mean and variance learning .	51
682	D.3	Sampling ODE and Generated Distribution	52
683	E	Detailed Derivations for Two-Layer Symmetric Parameterization	54
684	E.1	Symmetric parametrization zero mean gradient dynamics	54
685	E.1.1	Simplifying assumption: orthogonal initialization $q_k^T q_m = 0$	55
686	E.1.2	Beyond Aligned Initialization : qualitative analysis of off diagonal dynamics	58
687	E.2	Sampling ODE and Generated Distribution	59
688	F	Detailed Derivations for Deep Linear network	62
689	F.1	Aligned assumption	62
690	F.2	Two layer linear network	63
691	F.2.1	Special case: homogeneous initialization $\Lambda_1 = \Lambda_2$ (symmetric two layer case)	64
692	F.2.2	General Case: general initialization (general two layer $\mathbf{W} = PQ$)	64
693	F.3	General deep linear network	65
694	F.3.1	Deep Residual network	66
695	G	Detailed Derivations for Linear Convolutional Network	68
696	G.1	General set up	68
697	G.2	General analysis of sampling dynamics	68
698	G.3	Full width linear convolutional network	69
699	G.3.1	Training dynamics of full width linear convolutional network	69
700	G.3.2	Sampling dynamics of full width linear convolutional network	72
701	G.4	Local patch linear convolutional network	75
702	G.4.1	Training dynamics of patch linear convolutional net	75
703	G.4.2	Sampling dynamics of patch linear convolutional net	79
704	G.5	Appendix: Useful math	79
705	H	Detailed derivation of Flow Matching model	82
706	H.1	Solution to the flow matching sampling ODE with optimal solution	83
707	H.2	Learning dynamics of flow matching objective (single layer)	84
708	H.2.1	Interaction of weight learning and flow sampling	85

709	H.3	Learning dynamics of flow matching objective (two-layers)	86
710	I	Detailed Experimental Procedure	89
711	I.1	Computational Resources	89
712	I.2	MLP architecture inspired by UNet	89
713	I.3	EDM Loss Function	90
714	I.4	Experiment 1: Diffusion Learning of High-dimensional Gaussian Data	92
715	I.4.1	Data Generation and Covariance Specification	92
716	I.4.2	Network Architecture and Training Setup	92
717	I.4.3	Sampling and Trajectory Visualization	92
718	I.4.4	Covariance Evaluation in the True Eigenbasis	92
719	I.5	Experiment 2: Diffusion Learning of MNIST MLP	93
720	I.5.1	Data Preprocessing	93
721	I.5.2	Network Architecture and Training Setup	93
722	I.5.3	Sampling and Analysis	93
723	I.6	Experiment 3: Diffusion learning of Image Datasets with EDM-style CNN UNet .	93

A Extended Related works

Beyond the closely related works reviewed in the main text, here we are some spiritually related lines of works that inspired ours.

Spectral effect in the sampling process of diffusion models Many works have observed that during the sampling process of diffusion models [22, 38]: low spatial frequency aspects of the sample (e.g. layout) were specified first in the denoiser, before the higher frequency ones (e.g. textures). This phenomenon has been understood through the natural statistics of images (e.g. power-law spectrum) [19] and theory of diffusion [11], and recently through the lens of stochastic localization [39]. Basically, low frequency aspects usually have higher variance, thus were later to be corrupted by noise, so earlier to be generated during sampling process.

In our current work, we extend this line of thought to consider the spectral effects on the training dynamics of diffusion models.

Inductive bias of deep networks There as been a rich history of studying the inductive bias or implicit regularization effect of deep neural network and gradient descent. Deep neural networks have been reported to tend to find low-rank solutions of the task [40], and deeper networks could find it difficult to learn higher-rank target functions. This finding has also been leveraged to facilitate low-rank solutions by over-parameterizing linear operations in a deep networks (e.g. linear [41] or convolution [42] layers).

Implicit bias of convolutional neural networks When the neural network has convolutional structures in it, what kind of inductive bias or regularization effect does it bring to the function approximator?

People have attacked this by analyzing (deep) linear networks. [37] analyzed the inductive bias of gradient learning of the linear convolution network. In their case, the kernel is as wide as the signal and with circular boundary condition, thus convolution is equivalent to pointwise multiplication in Fourier space, which simplified the problem a lot. Then they can derive the learning dynamics of each Fourier mode. This result can be unified with other linear network approaches [18].

[43] further analyzed the inductive bias of the linear convolutional network with non-trivial local kernel size (neither pointwise nor full image) and multiple channels, and provided analytical statements about the inductive bias. However, they also found less success for closed form solutions for even two-layer convolutional networks with finite kernel width.

From an algebraic and geometric perspective, [44, 45] have analyzed the geometry of the function space of the deep linear convolutional network, which is equivalent to the space of polynomials that can be factorized into shorter polynomials.

Deep image prior and spectral bias in CNN On the empirical side, one intriguing result comes from the famous Deep Image Prior (DIP) experiment of Ulyanov et al. [46]. They showed that if a deep convolutional network (e.g. UNet) is used to regress a noisy image as target with pure noise as input, then when we employ early stopping in the optimization, the neural network will produce a cleaner image, thus performing denoising. To understand this method, [47] showed empirical evidence that deep convolutional networks tend to fit the low *spatial frequency* aspect of the data first. Thus, given the different spectral signature of natural images and noise, networks will fit the natural image before the noise. As a corollary, they showed that if the noise has more low frequency components, then neural network will fail to denoise those low frequency corruptions from image.

People have also looked at the inductive bias of untrained convolutional neural networks. Theoretically, [48] showed that infinite-width convolutional network at initialization is equivalent to spatial Gaussian process (random field), and the authors used this Bayesian perspective to understand the Deep image prior.

We noticed that this line of works in deep image prior has intriguing conceptual connection to our current work, i.e. the spectral bias of learning a function with convolutional architecture tend to learn lower frequency aspect first. Comparing diffusion models to DIP, diffusion models regress clean images from many randomly sampled noisy images; on the contrary DIP regress the clean images on a single noise pattern.

775 **Neural Tangent Kernel** A widely recognized technique for analyzing the learning dynamics of
776 deep neural network is the neural tangent kernel. For example, an infinitely wide network would be
777 similar to a kernel machine, where the learning dynamics will be linearized and reduce to exponential
778 convergence along different eigenmode of the tangent kernel.

779 Using neural tangent kernel (NTK) techniques, by inspecting the eigenvalues of the NTK associated
780 with functions of different frequency, [49] has been able to show that given uniform data on sphere
781 assumption, and simple neural network architectures (two-layer fully connected network with ReLU
782 nonlinearity), neural networks learn lower-frequency functions faster, with learning speed quadrati-
783 cally related to the frequency. Later they lifted the spatial uniformity assumption [50], and derived
784 how convergence speed and eigenvalues depend on the local data density.

785 These insights have been leveraged in classification problems to show that early stopping can lead
786 neural networks to learn smoother functions, thus being robust to labeling noise [51] .

787 What about convolutional architecture? With some similar NTK techniques, using a simplified
788 architecture, [52] proved that the learning dynamics of the convolutional network will preferably
789 learn the lower spatial frequency aspect of target image first. Their proof technique is also based
790 on the relationship between over-parametrized neural network and the tangent kernel. The proof
791 is based on a simpler generator architecture: one convolutional layer with ReLU nonlinearity and
792 fixed readout vector. They numerically showed the same effect for deeper architectures. This result
793 provided further theoretical foundation for the Deep Image Prior.

794 B Extended Results

795 B.1 Extended Visualization of Theoretical Results

796 B.1.1 Scaling curves of diffusion training (EDM)

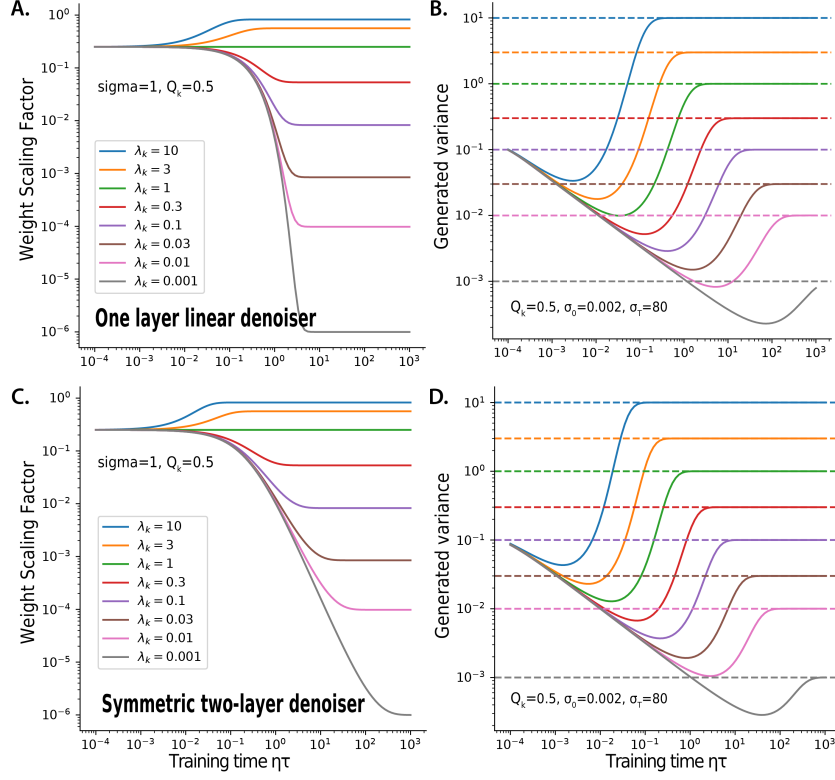


Figure 5: **Learning dynamics of the weight and variance of the generated distribution per eigenmode (continued)** **Top** Single layer linear denoiser. **Bottom** Symmetric two-layer denoiser. **A.C.** Learning dynamics of $\mathbf{u}_k^\top \mathbf{W}(\tau) \mathbf{u}_k$. **B.D.** Learning dynamics of the variance of the generated distribution $\tilde{\lambda}_k$, as a function of the variance of the target eigenmode λ_k . This case with larger amplitude weight initialization $Q_k = 0.5$.

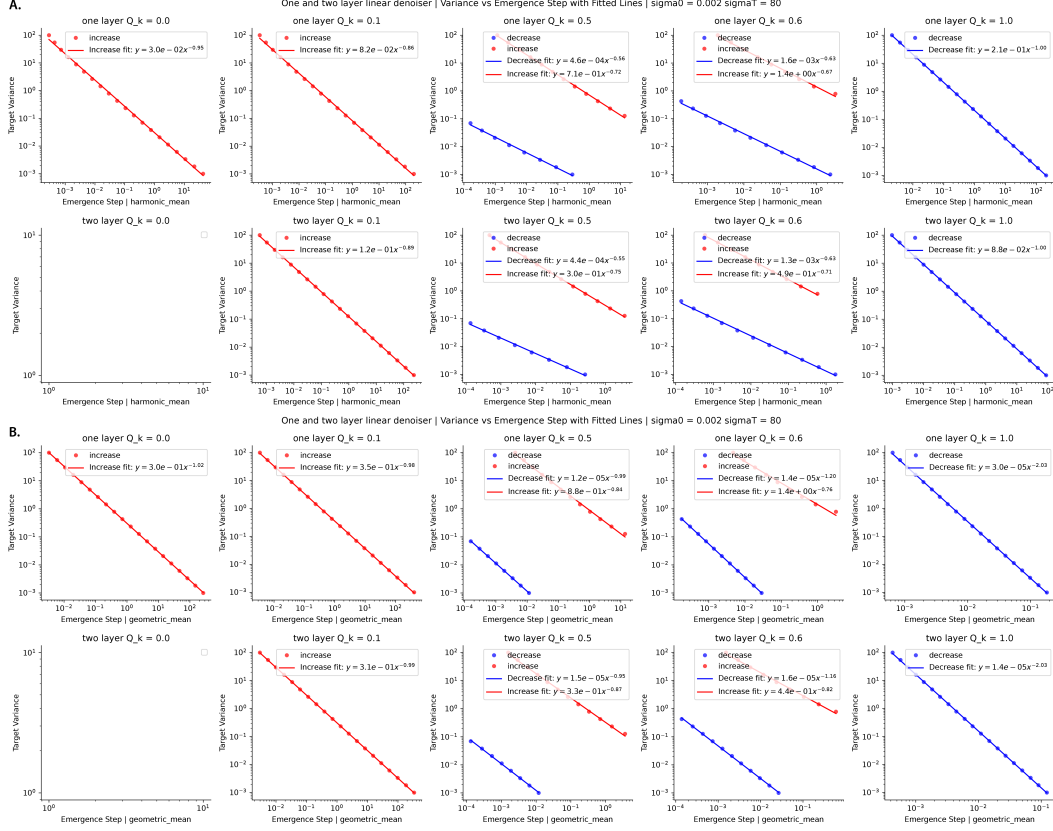


Figure 6: Power law relationship between mode emergence time and target mode variance for one-layer and two-layer linear denoisers. Panels (A) and (B) respectively plot the Mode variance against the Emergence Step for different values of weight initialization $Q_k \in \{0.0, 0.1, 0.5, 0.6, 1.0\}$ (columns), for one layer and two layer linear denoiser (rows). We used $\sigma_0 = 0.002$ and $\sigma_T = 80$. The emergence steps were quantified via different criterions, via harmonic mean in **A**, and geometric mean in **B**. Within each panel, red markers and lines denote the modes where their variance increases; blue markers and lines denote modes that “decrease” their variance. The solid lines show least-squares fits on log-log scale, giving rise to the $y = ax^b$ type relation. Comparisons reveal a systematic power-law decay of variance with respect to the Emergence Step under both the harmonic-mean and geometric-mean definitions. Note, the $Q_k = 0$ and two layer case was empty since zero initialization is an (unstable) fixed point, thus it will not converge.

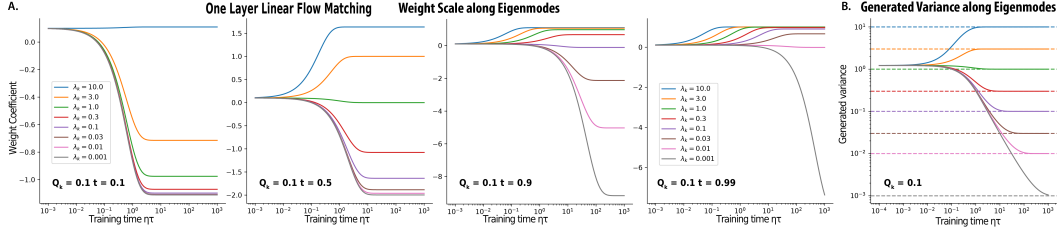


Figure 7: **Learning dynamics of the weight and variance of the generated distribution per eigenmode, for one layer linear flow matching model** Similar plotting format as Fig. 2. **A.** Learning dynamics of weights $\mathbf{u}_k^T \mathbf{W}(\tau; t) \mathbf{u}_k$ for various time point $t \in \{0.1, 0.5, 0.9, 0.99\}$. **B.** Learning dynamics of the variance of the generated distribution $\tilde{\lambda}_k$, as a function of the variance of the target eigenmode $\lambda_k \in \{10, 3, 1, 0.3, 0.1, 0.03, 0.01, 0.001\}$. Weight initialization is set at $Q_k = 0.1$ for every mode.

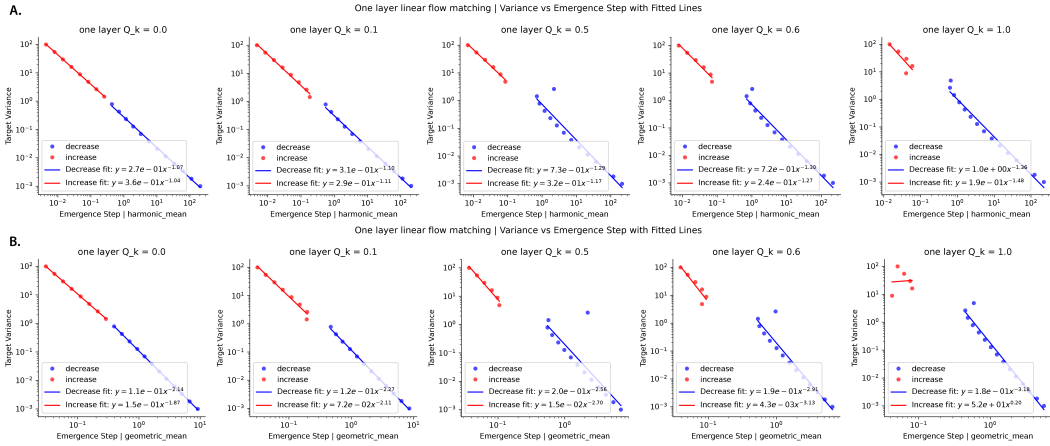


Figure 8: **Power law relationship between mode emergence time and target mode variance for one-layer linear flow matching.** Panels (A) and (B) respectively plot the Mode variance against the Emergence Step for different values of weight initialization $Q_k \in \{0.0, 0.1, 0.5, 0.6, 1.0\}$ (columns), for one layer linear flow model. The emergence steps were quantified via different criterions, via harmonic mean in A, and geometric mean in B. We used the same plotting format as in Fig. 6. Comparisons reveal a systematic power-law decay of variance with respect to the Emergence Step under both the harmonic-mean and geometric-mean definitions.

798 B.2 Extended Empirical Results

799 B.2.1 MLP-UNet Gaussian training experiments

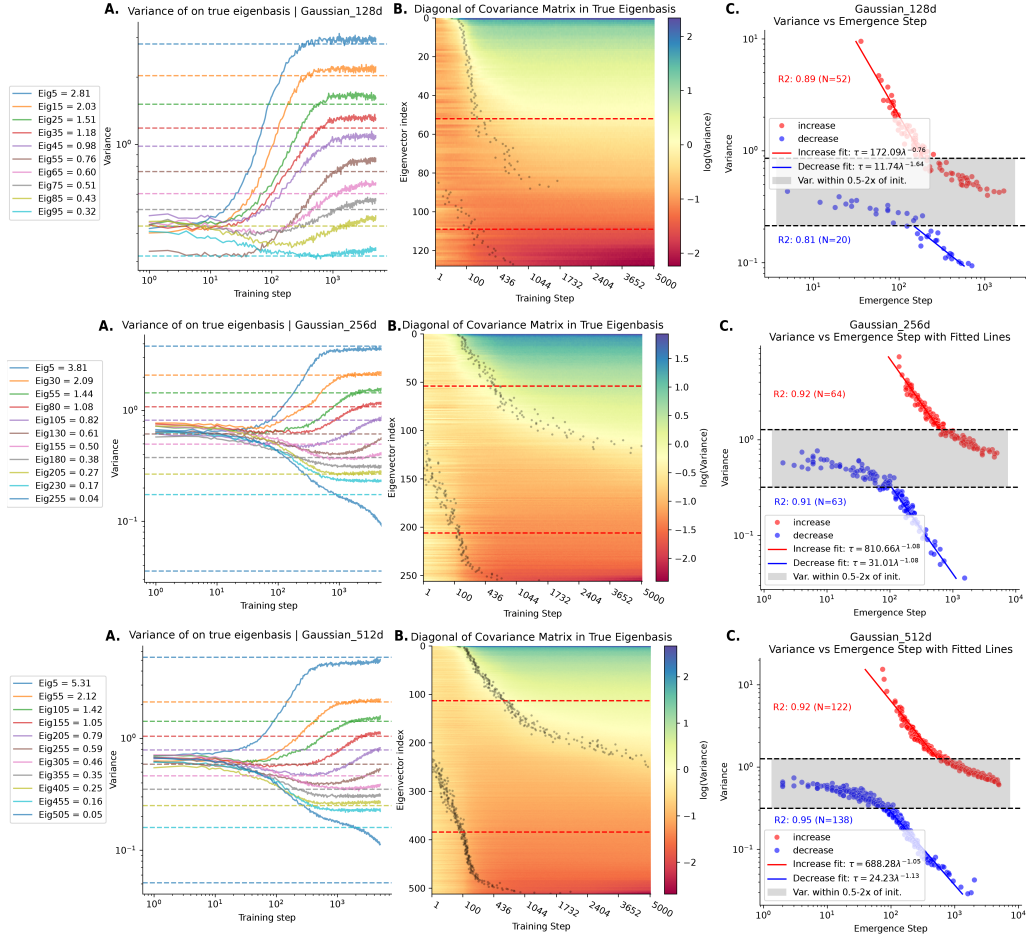


Figure 9: **Spectral Learning Dynamics of MLP-UNet (Gaussian-rotated)**. (same layout and analysis procedure as main Fig. 3) Top, middle, bottom show cases for 128d, 256d and 512d Gaussian. **A.** Evolution of sample variance $\tilde{\lambda}_k(\tau)$ across eigenmodes during training. **B.** Heatmap of variance trajectories along all eigenmodes, with dots marking mode emergence times τ^* (first-passage time at the geometric mean of initial and final variances). The gray zone (0.5–2 \times target variance) indicates modes starting too close to their target, causing unreliable τ^* estimates. **C.** Power-law scaling of τ^* versus target variance λ_k , excluding gray-zone eigenmodes for stability.

800 B.2.2 MLP-UNet natural image training experiments

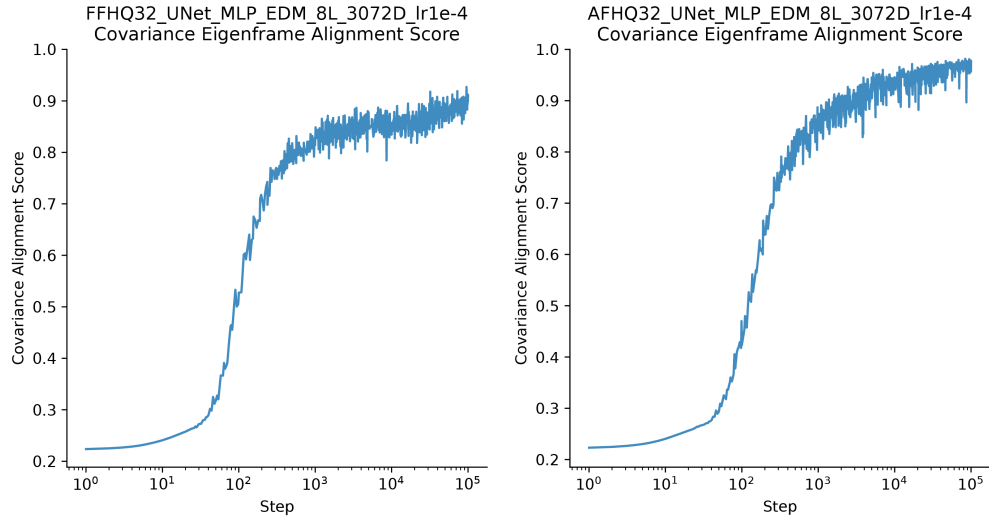


Figure 10: **Dynamical alignment onto the covariance eigenframe of data (MLP-UNet, FFHQ32, AFHQ32).** Alignment score χ as function of training step. Alignment score defined as the sum of square of diagonal entries of the rotated sample covariance on the training data eigenframe $U^T \hat{\Sigma}_\tau U$, divided by the sum of square of all entries. This quantifies how well the training data eigenframe diagonalizes the generated sample covariance. It will be $\chi = 1$ if U is the eigenbasis of $\hat{\Sigma}_\tau$.

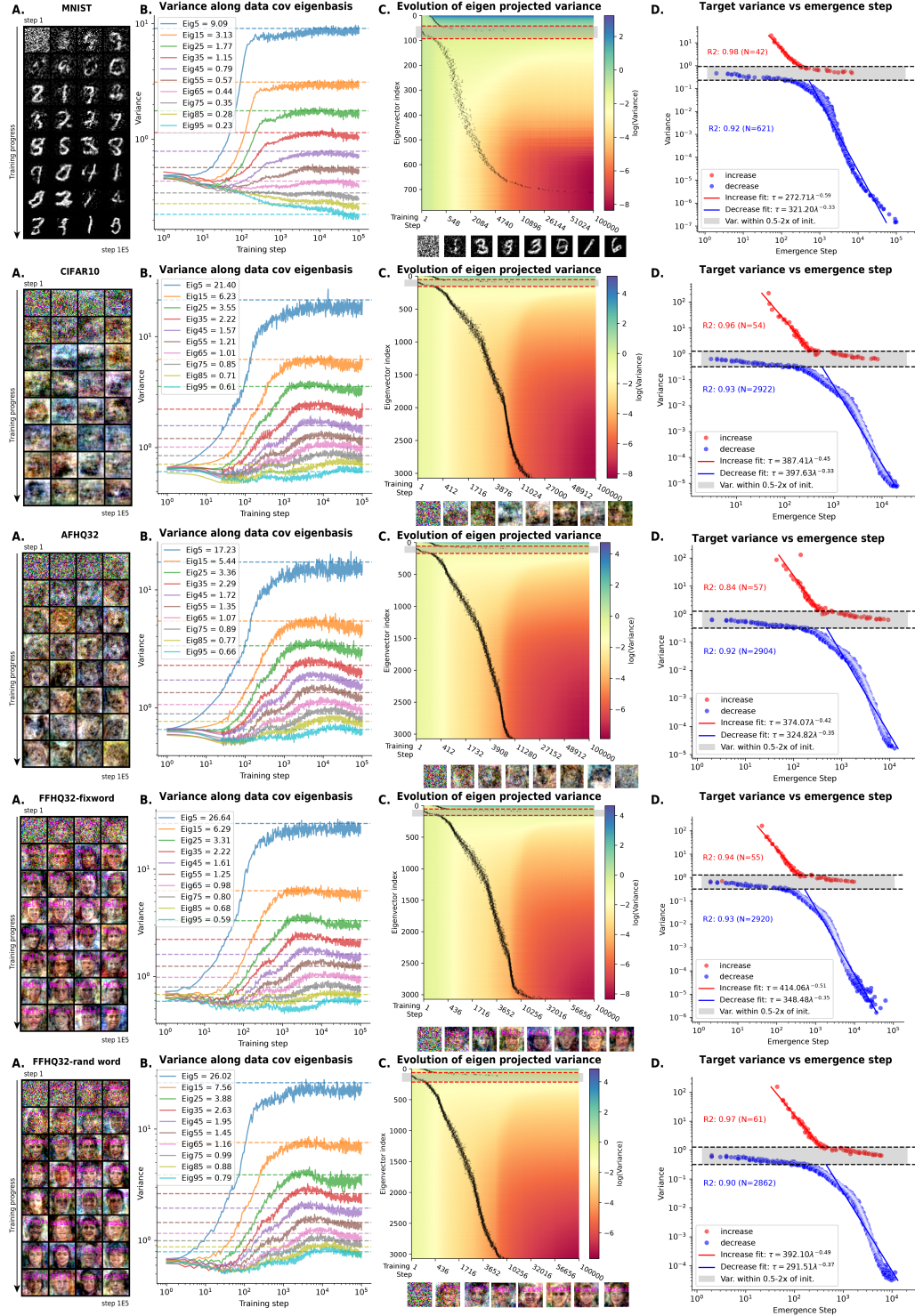


Figure 11: **Spectral Learning Dynamics of MLP-UNet (MNIST, CIFAR10, AFHQ32, FFHQ32-fixword, random word).** **A.** Generated samples during training. **B.** Evolution of sample variance $\tilde{\lambda}_k(\tau)$ across eigenmodes during training. **C.** Heatmap of variance trajectories along all eigenmodes, with dots marking mode emergence times τ^* (first-passage time at the geometric mean of initial and final variances). The **gray zone** (0.5–2 \times target variance) indicates modes starting too close to their target, causing unreliable τ^* estimates. **D.** Power-law scaling of τ^* versus target variance λ_k . A separate law was fit for modes with **increasing** and **decreasing** variance, excluding the middle gray-zone eigenmodes for stability.

801 B.2.3 CNN-UNet training experiment

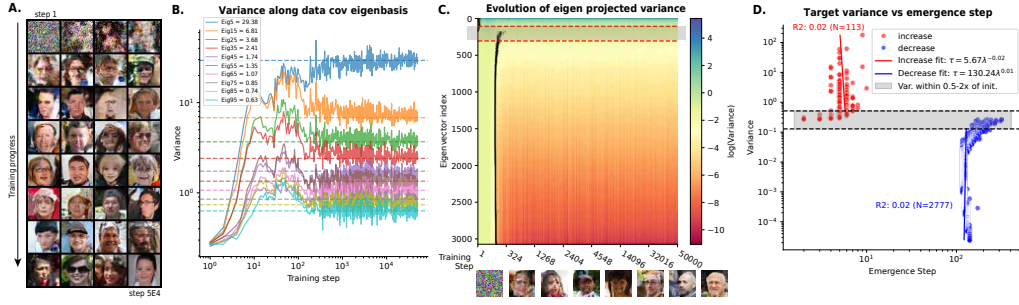


Figure 12: **Spectral Learning Dynamics of CNN-UNet (FFHQ32)**. (same layout and analysis procedure as main Fig. 3) **A.** Generated samples during training. **B.** Evolution of sample variance $\lambda_k(\tau)$ across eigenmodes during training. **C.** Heatmap of variance trajectories along all eigenmodes, with dots marking mode emergence times τ^* (first-passage time at the geometric mean of initial and final variances). The gray zone ($0.5\text{--}2\times$ target variance) indicates modes starting too close to their target, causing unreliable τ^* estimates. **D.** Power-law scaling of τ^* versus target variance λ_k , excluding gray-zone eigenmodes for stability.

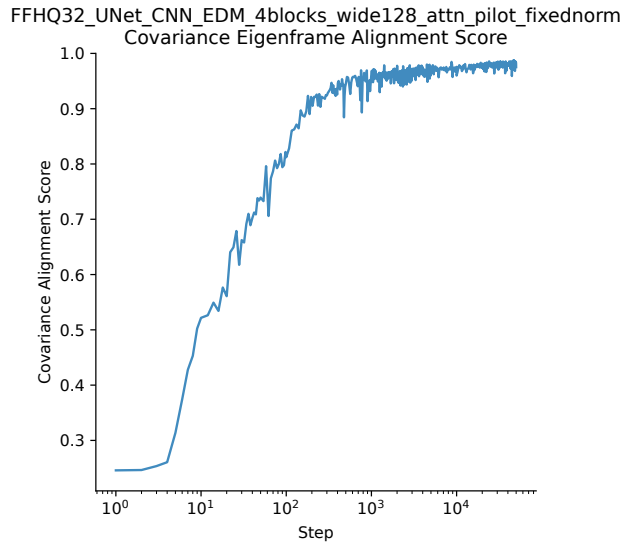


Figure 13: **Dynamical alignment onto the covariance eigenframe of data (CNN-UNet, FFHQ32)**. Alignment score r as function of training step. Same analysis as Fig.10.

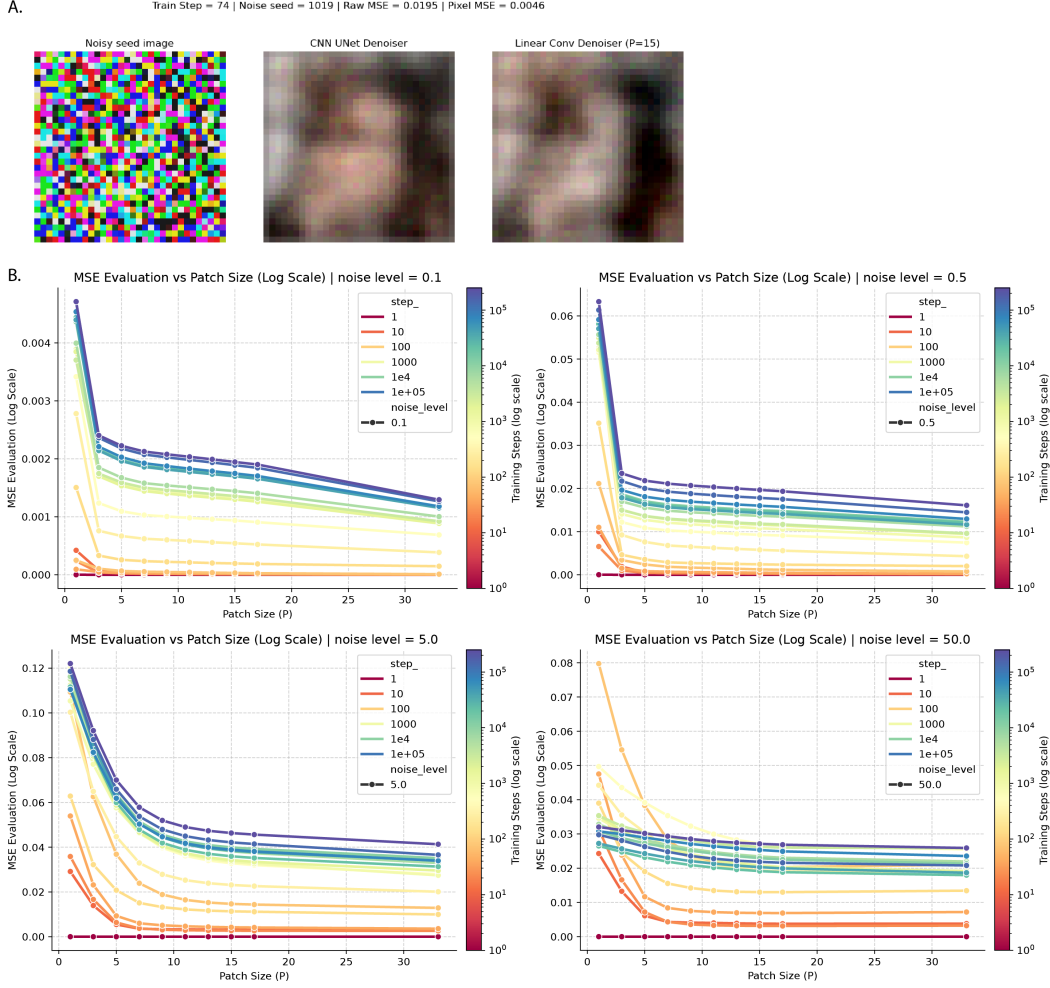


Figure 14: UNet denoiser can be approximated by linear convolution early in training (CNN-UNet, FFHQ32). **A.** Early in training, the UNet denoiser output can be well approximated by a linear convolutional layer, with a patch size P . **B.** The approximation error as a function of patch size P , training time τ and noise scale σ . Generally, early in training, the denoiser is very local and linear, well approximated by a linear convolutional layer.

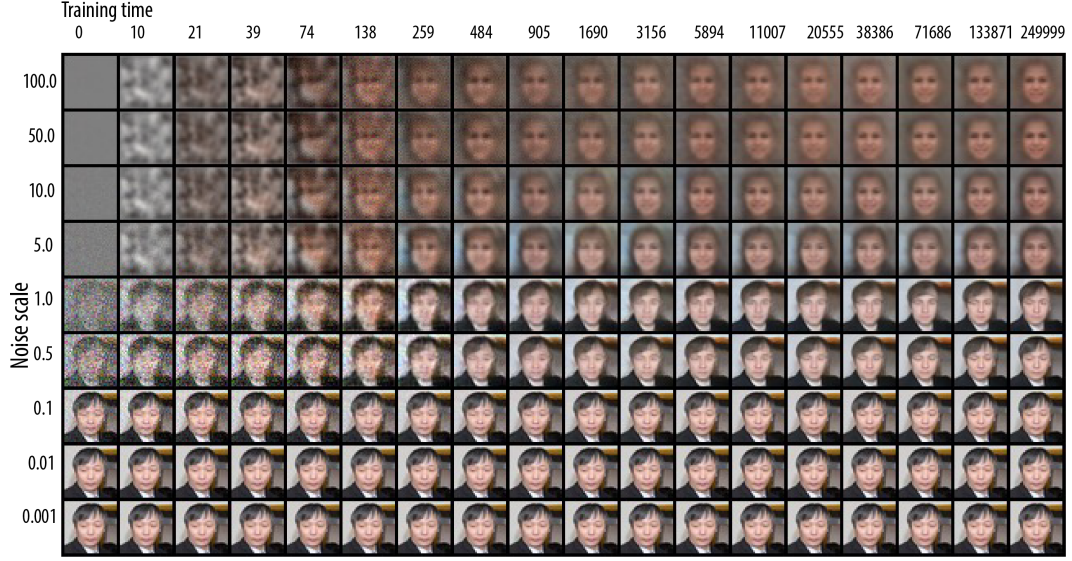


Figure 15: Visualizing the denoiser training dynamics with a fixed image and noise seed (CNN-UNet, FFHQ32). $D(\mathbf{x} + \sigma\mathbf{z}, \sigma)$ as a function of training time τ and noise scale σ .

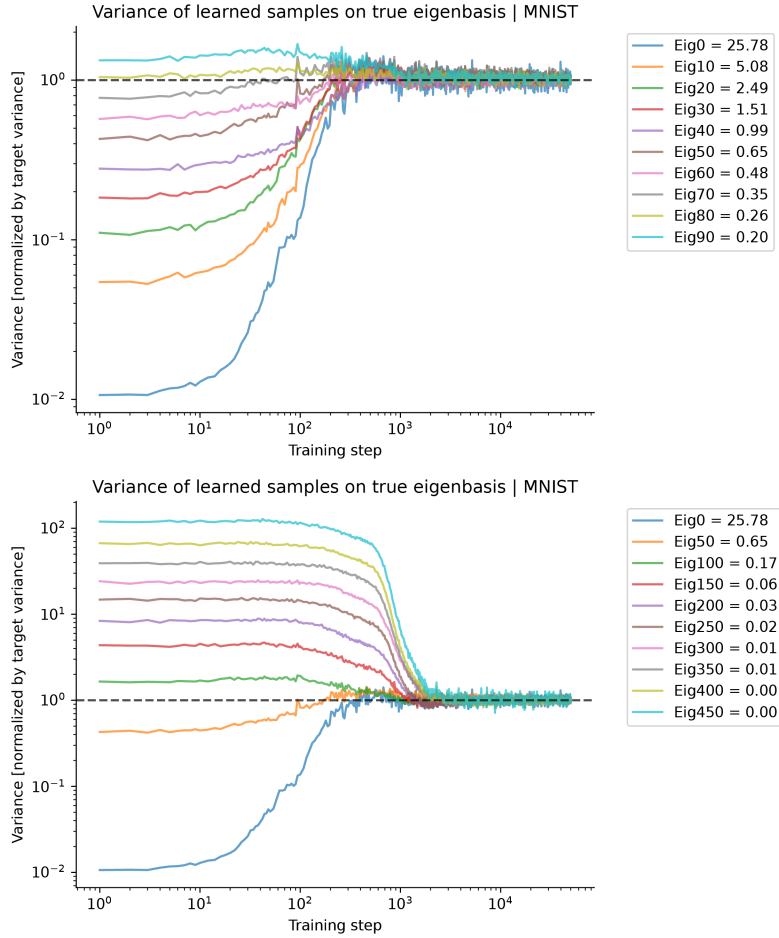


Figure 16: **Spectral Bias in Whole Image of CNN learning | MNIST** Training dynamics of sample (whole image) variance along eigenbasis of training set, normalized by target variance. **Upper** 0-100 eigen modes, **Lower** 0-500 eigenmodes.

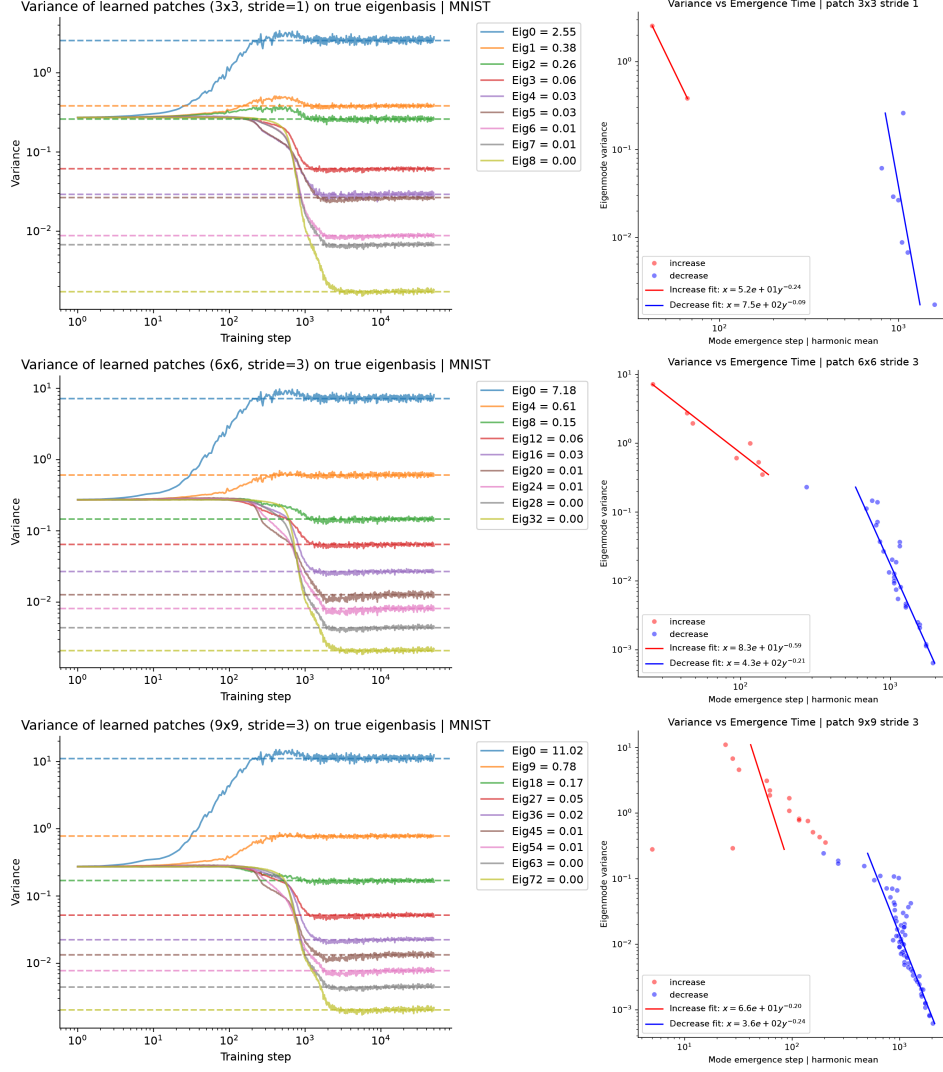


Figure 17: **Spectral Bias in CNN-Based Diffusion Learning: Variance Dynamics in Image Patches | MNIST (32 pixel resolution).** Left, Raw variance of generated patches along true eigenbases during training. Right, Scaling relationship between the target variance of eigenmode versus mode emergence time (harmonic mean criterion). Each row corresponds to a different patch size and stride used for extracting patches from images.

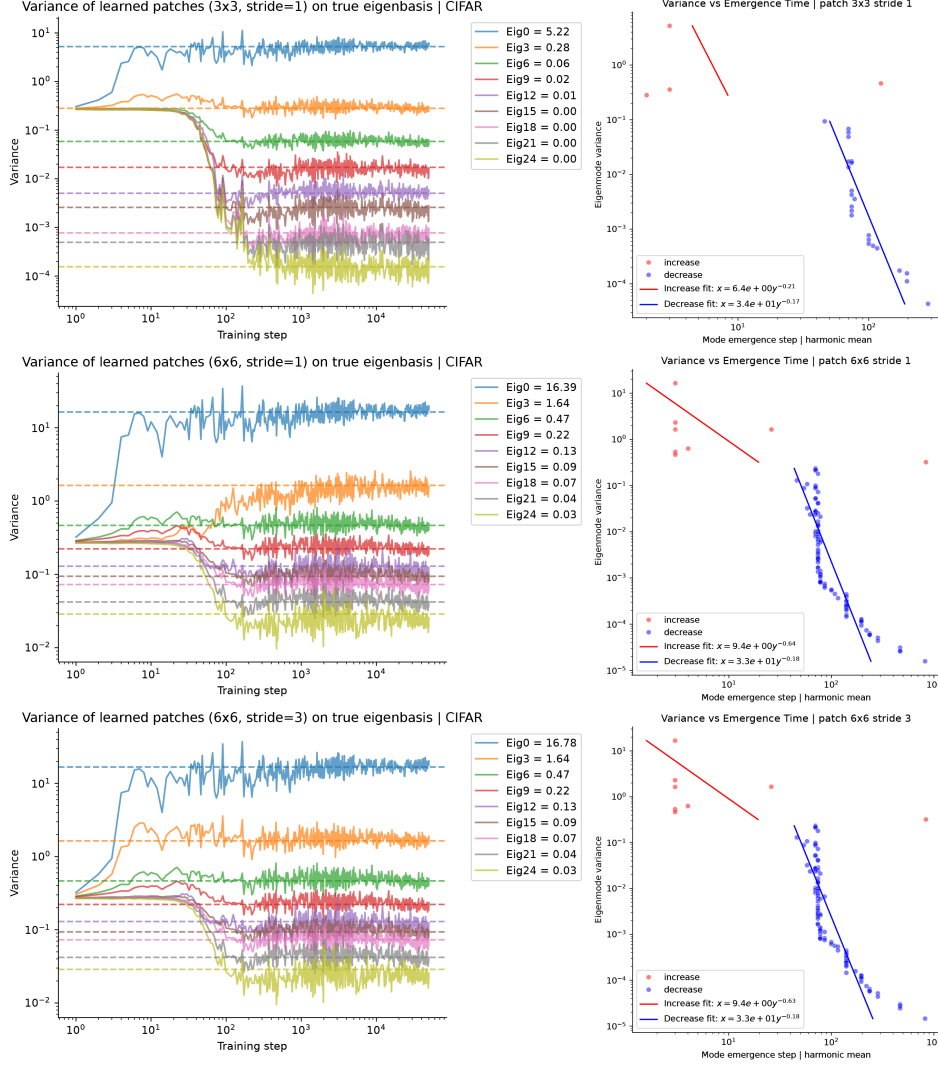


Figure 18: **Spectral Bias in CNN-Based Diffusion Learning: Variance Dynamics in Image Patches | CIFAR10 (32 pixel resolution).** **Left,** Raw variance of generated patches along true eigenbases during training. **Right,** Scaling relationship between the target variance of eigenmode versus mode emergence time (harmonic mean criterion). Each row corresponds to a different patch size and stride used for extracting patches from images.

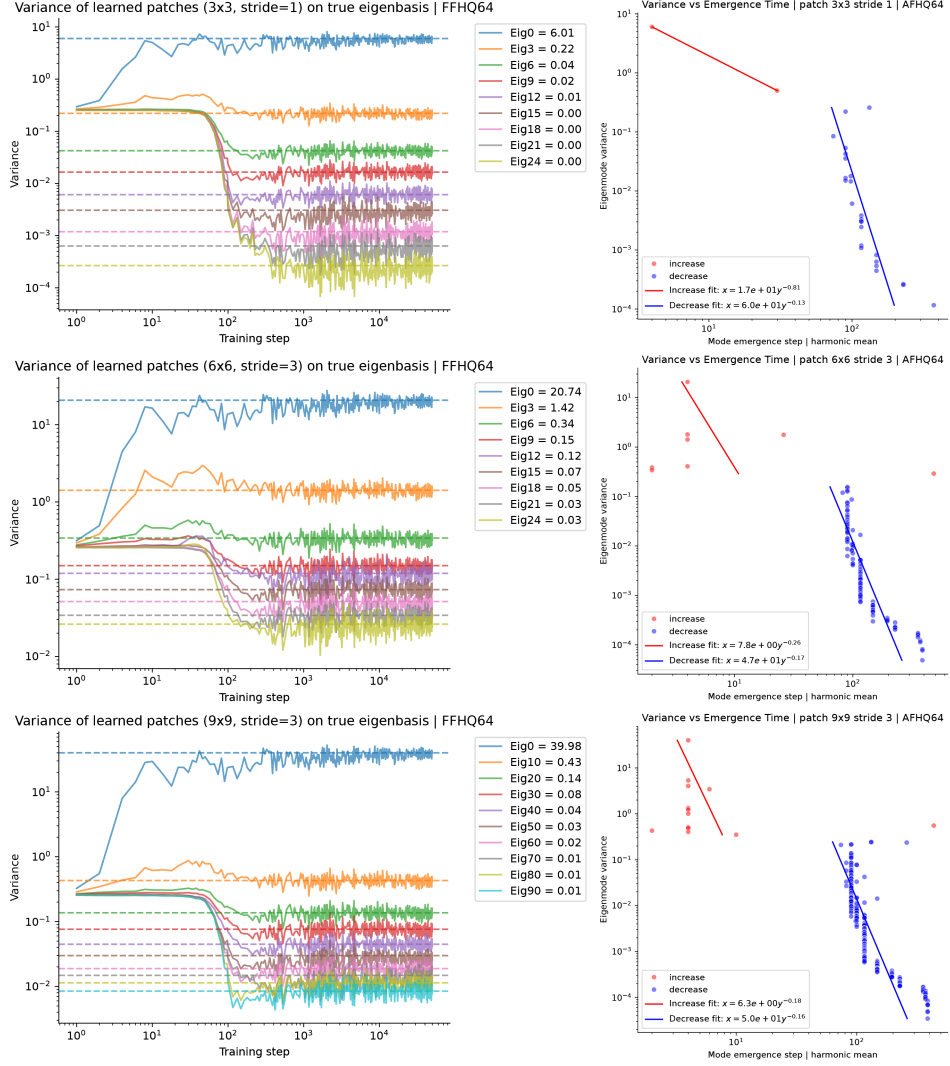


Figure 19: **Spectral Bias in CNN-Based Diffusion Learning: Variance Dynamics in Image Patches | FFHQ (64 pixel resolution).** Left, Raw variance of generated patches along true eigenbases during training. Right, Scaling relationship between the target variance of eigenmode versus mode emergence time (harmonic mean criterion). Each row corresponds to a different patch size and stride used for extracting patches from images.

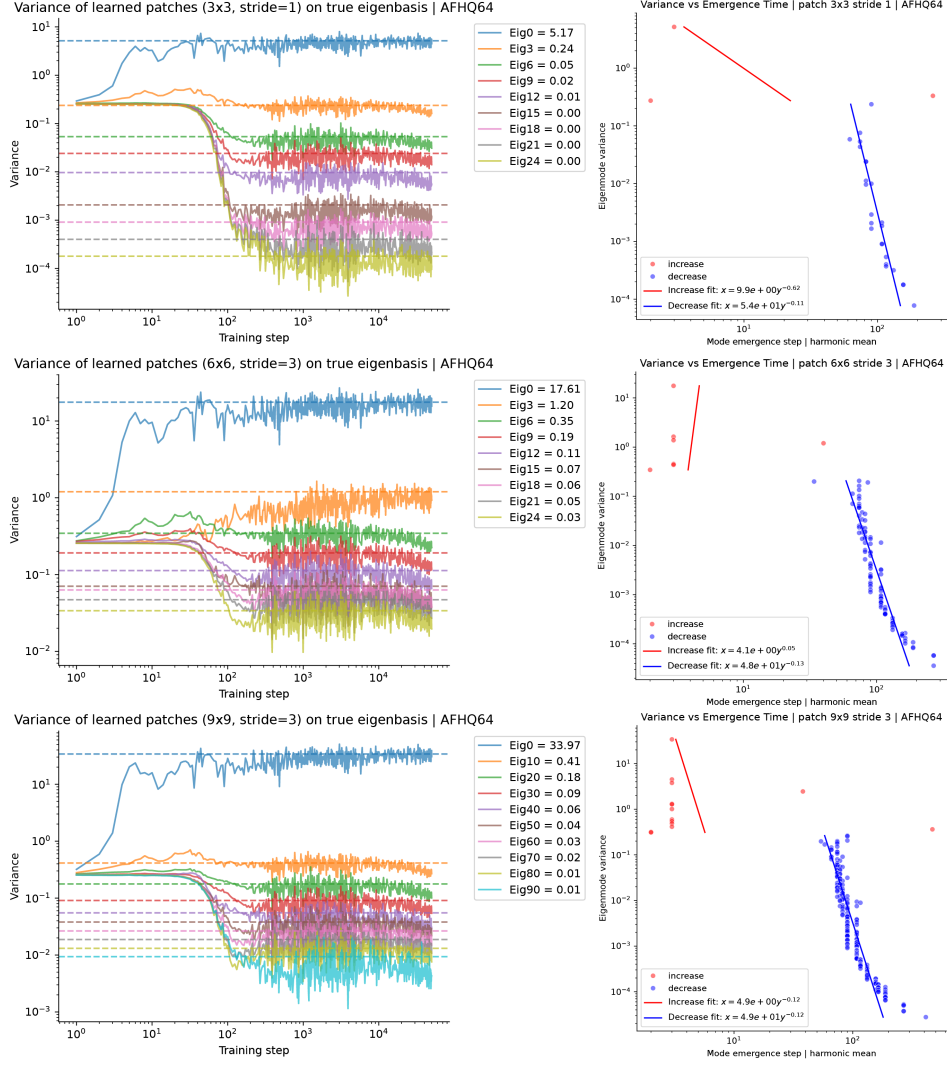


Figure 20: **Spectral Bias in CNN-Based Diffusion Learning: Variance Dynamics in Image Patches | AFHQv2 (64 pixel resolution).** Left, Raw variance of generated patches along true eigenbases during training. Right, Scaling relationship between the target variance of eigenmode versus mode emergence time (harmonic mean criterion). Each row corresponds to a different patch size and stride used for extracting patches from images.

802 C Detailed Derivation: General analysis

803 C.1 General Property of Linear Regression

804 C.1.1 Gaussian equivalence

805 **Lemma C.1.** *For a general linear regression problem, where \mathbf{x}, \mathbf{y} come from an arbitrary joint*
 806 *distribution $p(\mathbf{x}, \mathbf{y})$ with finite moments,*

$$\mathcal{L} = \mathbb{E}_{\mathbf{x}, \mathbf{y}} \|\mathbf{W}\mathbf{x} + \mathbf{b} - \mathbf{y}\|^2$$

807 *then its optimal solution and gradient only depend the first two moments of \mathbf{x}, \mathbf{y} ,*

808 *Proof.* Let the error be $\mathbf{e} = \mathbf{W}\mathbf{x} + \mathbf{b} - \mathbf{y}$, then

$$\begin{aligned} \nabla_{\mathbf{W}} \mathcal{L} &= \frac{\partial}{\partial \mathbf{W}} \mathbb{E}_{\mathbf{x}, \mathbf{y}} [\mathbf{e}^\top \mathbf{e}] \\ &= 2\mathbb{E} [\mathbf{e} \mathbf{x}^\top] \\ &= 2\mathbb{E} [(\mathbf{W}\mathbf{x} + \mathbf{b} - \mathbf{y}) \mathbf{x}^\top] \\ &= 2\left(\mathbf{W}\mathbb{E} [\mathbf{x} \mathbf{x}^\top] + \mathbf{b}\mathbb{E} [\mathbf{x}^\top] - \mathbb{E} [\mathbf{y} \mathbf{x}^\top]\right) \\ &= 2\left(\mathbf{W}(\Sigma_{xx} + \mu_x \mu_x^\top) + \mathbf{b}\mu_x^\top - (\Sigma_{yx} + \mu_y \mu_x^\top)\right) \\ &= 2\mathbf{W}(\Sigma_{xx} + \mu_x \mu_x^\top) + 2(\mathbf{b} - \mu_y) \mu_x^\top - 2\Sigma_{yx} \\ &= 2(\mathbf{W}\Sigma_{xx} - \Sigma_{yx}) + 2(\mathbf{W}\mu_x + \mathbf{b} - \mu_y) \mu_x^\top \\ &= 2(\mathbf{W}\Sigma_{xx} - \Sigma_{yx}) + \nabla_{\mathbf{b}} \mathcal{L} \mu_x^\top \end{aligned}$$

809

$$\begin{aligned} \nabla_{\mathbf{b}} \mathcal{L} &= \frac{\partial}{\partial \mathbf{b}} \mathbb{E} [\mathbf{e}^\top \mathbf{e}] \\ &= 2\mathbb{E} [\mathbf{e}] \\ &= 2\mathbb{E} [\mathbf{W}\mathbf{x} + \mathbf{b} - \mathbf{y}] \\ &= 2(\mathbf{W}\mu_x + \mathbf{b} - \mu_y) \end{aligned}$$

810 We used the fact that

$$\begin{aligned} \mathbb{E} [\mathbf{x}] &= \mu_x \\ \mathbb{E} [\mathbf{y}] &= \mu_y \\ \mathbb{E} [\mathbf{y} \mathbf{x}^\top] &= \Sigma_{yx} + \mu_y \mu_x^\top \\ \mathbb{E} [\mathbf{x} \mathbf{x}^\top] &= \Sigma_{xx} + \mu_x \mu_x^\top \end{aligned}$$

811 Setting gradient to zero, we get optimal values

$$\mathbf{W}^* = \Sigma_{yx} \Sigma_{xx}^{-1} \tag{11}$$

$$\mathbf{b}^* = \mu_y - \mathbf{W}^* \mu_x \tag{12}$$

812 The gradient flow dynamics read

$$\frac{d}{d\tau} \mathbf{W} = -2\eta(\mathbf{W}\Sigma_{xx} - \Sigma_{yx}) - 2\eta \nabla_{\mathbf{b}} \mathcal{L} \mu_x^\top \tag{13}$$

$$\frac{d}{d\tau} \mathbf{b} = -2\eta(\mathbf{W}\mu_x + \mathbf{b} - \mu_y) \tag{14}$$

813 The Σ_{xx} determines the gradient flow dynamics and convergence rate of \mathbf{W} , while $\Sigma_{xx}^{-1} \Sigma_{yx}$ deter-
 814 mines the target level or optimal solution of the regression. \square

815 *Remark C.2.* For all the loss variant, with linear denoiser, the loss is of this class. We can write down
 816 the gradient structure and optimal values of \mathbf{W} and \mathbf{b} , by plugging in the mean and covariance of
 817 input \mathbf{x} and predicting target \mathbf{y} .

818 **Lemma C.3.** For two independent random variable $X, Y \in \mathbb{R}^d$, we have

$$\text{Cov}(aX + bY, cX + dY) = ac\Sigma_X + bd\Sigma_Y \quad (15)$$

819 *Proof.* This can be proved using bilinearity of covariance and independence which entails $\Sigma_{XY} = 0$.

$$\begin{aligned} \text{Cov}(aX + bY, cX + dY) &= a\text{Cov}(X, cX + dY) + b\text{Cov}(Y, cX + dY) \\ &= ac\text{Cov}(X, X) + ad\text{Cov}(X, Y) + bc\text{Cov}(Y, X) + bd\text{Cov}(Y, Y) \\ &= ac\Sigma_X + bd\Sigma_Y \end{aligned}$$

820 □

821 *Remark C.4.* For diffusion training loss, the clean image samples and the noise patterns are designed
 822 to be sampled independently. Thus we can use (15) to calculate the covariance of input and outputs.

823 Using these two lemmas, the gradient structure and learning dynamics of various loss functions can
 824 be easily read off.

825 C.2 General analysis of the gradient learning of linear predictor

826 C.2.1 Denoising as Ridge Regression

827 **Lemma C.5** (Diffusion learning as Ridge regression). For linear denoisers $\mathbf{D}(\mathbf{x}, \sigma) = \mathbf{W}\mathbf{x} + \mathbf{b}$, at
 828 full batch limit, the denoising score matching loss (2) is equivalent to the following Ridge regression
 829 loss.

$$\mathcal{L}_\sigma = \mathbb{E}_{\mathbf{x}} \|\mathbf{W}\mathbf{x} + \mathbf{b} - \mathbf{x}\|^2 + \sigma^2 \|\mathbf{W}\|^2 \quad (16)$$

Proof.

$$\begin{aligned} \mathcal{L}_\sigma &= \mathbb{E}_{\mathbf{x} \sim p_0, \mathbf{z} \sim \mathcal{N}(0, \mathbf{I})} \|\mathbf{D}_\theta(\mathbf{x} + \sigma\mathbf{z}; \sigma) - \mathbf{x}\|^2 \\ &= \mathbb{E}_{\mathbf{x} \sim p_0, \mathbf{z} \sim \mathcal{N}(0, \mathbf{I})} \|\mathbf{W}(\mathbf{x} + \sigma\mathbf{z}) + \mathbf{b} - \mathbf{x}\|^2 \\ &= \mathbb{E}_{\mathbf{x} \sim p_0, \mathbf{z} \sim \mathcal{N}(0, \mathbf{I})} \text{Tr}[(\mathbf{W}(\mathbf{x} + \sigma\mathbf{z}) + \mathbf{b} - \mathbf{x})(\mathbf{W}(\mathbf{x} + \sigma\mathbf{z}) + \mathbf{b} - \mathbf{x})^\top] \\ &= \mathbb{E}_{\mathbf{x} \sim p_0, \mathbf{z} \sim \mathcal{N}(0, \mathbf{I})} \text{Tr}[(\mathbf{W}(\mathbf{x} + \sigma\mathbf{z}) + \mathbf{b} - \mathbf{x})((\mathbf{x} + \sigma\mathbf{z})^\top \mathbf{W}^\top + \mathbf{b}^\top - \mathbf{x}^\top)] \\ &= \mathbb{E}_{\mathbf{x} \sim p_0, \mathbf{z} \sim \mathcal{N}(0, \mathbf{I})} \text{Tr}[(\mathbf{W}\mathbf{x} + \mathbf{b} - \mathbf{x} + \sigma\mathbf{W}\mathbf{z})(\mathbf{x}^\top \mathbf{W}^\top + \mathbf{b}^\top - \mathbf{x}^\top + \sigma\mathbf{z}^\top \mathbf{W}^\top)] \\ &= \mathbb{E}_{\mathbf{x} \sim p_0, \mathbf{z} \sim \mathcal{N}(0, \mathbf{I})} \text{Tr}[(\mathbf{W}\mathbf{x} + \mathbf{b} - \mathbf{x})(\mathbf{W}\mathbf{x} + \mathbf{b} - \mathbf{x})^\top + 2\sigma\mathbf{W}\mathbf{z}(\mathbf{W}\mathbf{x} + \mathbf{b} - \mathbf{x})^\top + \sigma^2\mathbf{W}\mathbf{z}\mathbf{z}^\top \mathbf{W}^\top] \\ &= \mathbb{E}_{\mathbf{x} \sim p_0} \|\mathbf{W}\mathbf{x} + \mathbf{b} - \mathbf{x}\|^2 + \mathbb{E}_{\mathbf{x} \sim p_0, \mathbf{z} \sim \mathcal{N}(0, \mathbf{I})} \text{Tr}[2\sigma\mathbf{W}\mathbf{z}(\mathbf{W}\mathbf{x} + \mathbf{b} - \mathbf{x})^\top + \sigma^2\mathbf{W}\mathbf{z}\mathbf{z}^\top \mathbf{W}^\top] \\ &= \mathbb{E}_{\mathbf{x} \sim p_0} \|\mathbf{W}\mathbf{x} + \mathbf{b} - \mathbf{x}\|^2 + \sigma^2 \|\mathbf{W}\|^2 \end{aligned}$$

830 This derivation depends on the linearity of the denoiser and the white Gaussian nature of noise
 831 $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$. □

832 *Remark C.6.* This equivalence reveals that for diffusion model the additive white noise can be viewed
 833 as L2 regularization on the weights of the auto-encoding regression problem, where σ^2 functions as
 834 regularization strength λ . Thus per classic analysis of Ridge regression, we can see, at higher noise
 835 scale, fewer modes of the data will be “resolved”.

836 Further, if the noise is zero-mean but not white, the final term $\text{Tr}[\sigma^2\mathbf{W}\mathbf{z}\mathbf{z}^\top \mathbf{W}^\top]$ will impose other
 837 types of weighted regularization on the weights.

838 C.3 General Analysis of the Denoising Score Matching Objective

839 To simplify, we first consider a fixed σ , ignoring the nonlinear dependency on it. Consider our score,
 840 denoiser function approximators a linear function $\mathbf{D}(\mathbf{x}; \sigma) = \mathbf{b} + \mathbf{W}\mathbf{x}$. Expanding the per-sample
 841 DSM loss, we get

$$\|\mathbf{D}(\mathbf{x}_0 + \sigma\mathbf{z}; \sigma) - \mathbf{x}_0\|^2 \quad (17)$$

$$= \|\mathbf{b} + \mathbf{W}(\mathbf{x}_0 + \sigma\mathbf{z}) - \mathbf{x}_0\|^2 \quad (18)$$

$$= \|\mathbf{b} + \mathbf{W}\sigma\mathbf{z} + (\mathbf{W} - \mathbf{I})\mathbf{x}_0\|^2 \quad (19)$$

$$= (\mathbf{b} + \mathbf{W}\sigma\mathbf{z})^T (\mathbf{b} + \mathbf{W}\sigma\mathbf{z}) + \mathbf{x}_0^T (\mathbf{W} - \mathbf{I})^T (\mathbf{W} - \mathbf{I}) \mathbf{x}_0 + 2(\mathbf{b} + \mathbf{W}\sigma\mathbf{z})^T (\mathbf{W} - \mathbf{I}) \mathbf{x}_0 \quad (20)$$

$$= \mathbf{b}^T \mathbf{b} + 2\sigma \mathbf{b}^T \mathbf{W} \mathbf{z} + \sigma^2 \mathbf{z}^T \mathbf{W}^T \mathbf{W} \mathbf{z} + \mathbf{x}_0^T (\mathbf{W} - \mathbf{I})^T (\mathbf{W} - \mathbf{I}) \mathbf{x}_0 \quad (21)$$

$$+ 2\mathbf{b}^T (\mathbf{W} - \mathbf{I}) \mathbf{x}_0 + 2\sigma \mathbf{z}^T \mathbf{W}^T (\mathbf{W} - \mathbf{I}) \mathbf{x}_0 \quad (22)$$

$$= \mathbf{b}^T \mathbf{b} + 2\sigma \mathbf{b}^T \mathbf{W} \mathbf{z} + \sigma^2 \text{Tr}[\mathbf{W}^T \mathbf{W} \mathbf{z} \mathbf{z}^T] + \text{Tr}[(\mathbf{W} - \mathbf{I})^T (\mathbf{W} - \mathbf{I}) \mathbf{x}_0 \mathbf{x}_0^T] \quad (23)$$

$$+ 2\mathbf{b}^T (\mathbf{W} - \mathbf{I}) \mathbf{x}_0 + 2\sigma \text{Tr}[\mathbf{W}^T (\mathbf{W} - \mathbf{I}) \mathbf{x}_0 \mathbf{z}^T] \quad (24)$$

$$= \|\mathbf{b} - \mathbf{x}_0\|^2 + \|\mathbf{W}(\mathbf{x}_0 + \sigma\mathbf{z})\|^2 + 2\mathbf{W}(\mathbf{x}_0 + \sigma\mathbf{z})(\mathbf{b} - \mathbf{x}_0)^T \quad (25)$$

842 **Full batch limit** Here we take the full-batch expectation over \mathbf{z}, \mathbf{x}_0 , where $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$, $\mathbf{x}_0 \sim$
 843 $p(\mathbf{x}_0)$. Their moments are the following.

$$\mathbb{E}[\mathbf{z}] = 0 \quad (26)$$

$$\mathbb{E}[\mathbf{z} \mathbf{z}^T] = \mathbf{I} \quad (27)$$

$$\mathbb{E}[\mathbf{x}_0] = \mu \quad (28)$$

$$\mathbb{E}[\mathbf{x}_0 \mathbf{x}_0^T] = \mu \mu^T + \Sigma \quad (29)$$

$$\mathbb{E}[\mathbf{x}_0 \mathbf{z}^T] = 0 \quad (30)$$

844 Note, the data do not need to be Gaussian, as long as the expectations or the first two moments are as
 845 computed above, the results should be the same. In a sense, if our function approximator is linear,
 846 then we just care about the Gaussian part of data.

$$\mathcal{L} = \mathbb{E}_{\mathbf{x}_0 \sim p(\mathbf{x}_0), \mathbf{z} \sim \mathcal{N}(0, \mathbf{I})} \|\mathbf{D}(\mathbf{x}_0 + \sigma\mathbf{z}; \sigma) - \mathbf{x}_0\|^2 \quad (31)$$

$$= \mathbf{b}^T \mathbf{b} + \sigma^2 \text{Tr}[\mathbf{W}^T \mathbf{W}] + \text{Tr}[(\mathbf{W} - \mathbf{I})^T (\mathbf{W} - \mathbf{I}) (\mu \mu^T + \Sigma)] + 2\mathbf{b}^T (\mathbf{W} - \mathbf{I}) \mu \quad (32)$$

847 We can simplify the objective by completing the square,

$$\mathcal{L} = \mathbb{E}_{\mathbf{x}_0 \sim p(\mathbf{x}_0), \mathbf{z} \sim \mathcal{N}(0, \mathbf{I})} \|\mathbf{D}(\mathbf{x}_0 + \sigma\mathbf{z}; \sigma) - \mathbf{x}_0\|^2 \quad (33)$$

$$= \|\mathbf{b} + (\mathbf{W} - \mathbf{I})\mu\|_2^2 - \mu^T (\mathbf{W} - \mathbf{I})^T (\mathbf{W} - \mathbf{I}) \mu + \sigma^2 \text{Tr}[\mathbf{W}^T \mathbf{W}] + \text{Tr}[(\mathbf{W} - \mathbf{I})^T (\mathbf{W} - \mathbf{I}) (\mu \mu^T + \Sigma)] \quad (34)$$

$$= \|\mathbf{b} - (\mathbf{I} - \mathbf{W})\mu\|_2^2 + \sigma^2 \text{Tr}[\mathbf{W}^T \mathbf{W}] + \text{Tr}[(\mathbf{W} - \mathbf{I})^T (\mathbf{W} - \mathbf{I}) \Sigma] \quad (35)$$

$$= \|\mathbf{b} - (\mathbf{I} - \mathbf{W})\mu\|_2^2 + \text{Tr}[\mathbf{W}^T \mathbf{W} (\sigma^2 \mathbf{I} + \Sigma)] - 2\text{Tr}[\mathbf{W}^T \Sigma] + \text{Tr}[\Sigma] \quad (36)$$

848 **Gradient** The gradients from loss to the weight and bias read

$$\nabla_{\mathbf{b}} \mathcal{L} = 2(\mathbf{b} - (\mathbf{I} - \mathbf{W})\mu) \quad (37)$$

$$\nabla_{\mathbf{W}} \mathcal{L} = -2\Sigma + 2\mathbf{W}(\sigma^2 \mathbf{I} + \Sigma) + [2\mathbf{W}\mu\mu^T + 2(\mathbf{b} - \mu)\mu^T] \quad (38)$$

849 **Global optimum** Examining the quadratic loss, and setting the gradient to zero, we can see the
 850 optimal parameters are the following,

$$\mathbf{b}^* = (\mathbf{I} - \mathbf{W}^*)\mu \quad (39)$$

$$\mathbf{W}^* = \Sigma(\sigma^2 \mathbf{I} + \Sigma)^{-1} \quad (40)$$

851 which recovers the Gaussian denoiser.

852 **Gradient flow dynamics** Next, let's examine the learning dynamics by gradient descent. We
853 consider the continuous-time limit, i.e. gradient flow. Denoting the learning rate η and continuous
854 training time τ , their gradient flow dynamics are

$$\frac{d}{d\tau} \mathbf{b} = -\eta \nabla_{\mathbf{b}} \mathcal{L} \quad (41)$$

$$\frac{d}{d\tau} \mathbf{W} = -\eta \nabla_{\mathbf{W}} \mathcal{L} \quad (42)$$

855 C.4 Gradient Structure of Other Variants of Loss

856 Here we generalize our analysis to many popular training loss of diffusion models, including flow
857 matching. We provide a detailed table for different variants and the learning dynamics of the linear
denoiser.

Table 2: **Variants of the diffusion training loss and their linear solution and gradient structure.**
Here Σ_{xx} denotes the covariance of the input to the linear predictor; Σ_{yx} is the covariance between
target and input, not to be confused with the covariance of training sample \mathbf{x}_0 . w_k^* denotes the optimal
weight projected onto principal component \mathbf{u}_k , $w_k^* := \mathbf{u}_k^T \mathbf{W}^* \mathbf{u}_k$; $1/\tau^*$ represents the convergence
speed of a target mode. We abbreviated away the expectation over data and noise in our loss

	EDM	X-pred	EPS-pred	V-pred	Flow matching
Loss	$\ \mathbf{D}_\theta(\mathbf{x} + \sigma\epsilon; \sigma) - \mathbf{x}\ ^2$	$\ \mathbf{F}_\theta(\alpha_t \mathbf{x} + \sigma_t \epsilon; t) - \mathbf{x}\ ^2$	$\ \mathbf{F}_\theta(\alpha_t \mathbf{x} + \sigma_t \epsilon; t) - \epsilon\ ^2$	$\ \mathbf{F}_\theta(\alpha_t \mathbf{x} + \sigma_t \epsilon; t) - (\alpha_t \epsilon - \sigma_t \mathbf{x})\ ^2$	$\ \mathbf{u}_\theta((1-t)\mathbf{x}_0 + t\mathbf{x}_1, t) - (\mathbf{x}_1 - \mathbf{x}_0)\ ^2$
Σ_{xx}	$\Sigma + \sigma^2 I$	$\alpha_t^2 \Sigma + \sigma_t^2 I$	$\alpha_t^2 \Sigma + \sigma_t^2 I$	$\alpha_t^2 \Sigma + \sigma_t^2 I$	$t^2 \Sigma + (1-t)^2 I$
Σ_{yx}	Σ	$\alpha_t \Sigma$	$\sigma_t I$	$\alpha_t \sigma_t (I - \Sigma)$	$t \Sigma - (1-t) I$
\mathbf{W}^*	$(\Sigma + \sigma^2 I)^{-1} \Sigma$	$\alpha_t (\alpha_t^2 \Sigma + \sigma_t^2 I)^{-1} \Sigma$	$\sigma_t (\alpha_t^2 \Sigma + \sigma_t^2 I)^{-1}$	$\alpha_t \sigma_t (\alpha_t^2 \Sigma + \sigma_t^2 I)^{-1} (I - \Sigma)$	$(t^2 \Sigma + (1-t)^2 I)^{-1} (t \Sigma - (1-t) I)$
\mathbf{b}^*	$\sigma^2 (\Sigma + \sigma^2 I)^{-1} \mu$	$\sigma_t^2 (\alpha_t^2 \Sigma + \sigma_t^2 I)^{-1} \mu$	$-\alpha_t \sigma_t (\alpha_t^2 \Sigma + \sigma_t^2 I)^{-1} \mu$	$-\sigma_t (\alpha_t^2 + \sigma_t^2) (\alpha_t^2 \Sigma + \sigma_t^2 I)^{-1} \mu$	$(1-t) (t^2 \Sigma + (1-t)^2 I)^{-1} \mu$
w_k^*	$\frac{\lambda_k}{\lambda_k + \sigma^2}$	$\frac{\alpha_t \lambda_k}{\alpha_t^2 \lambda_k + \sigma_t^2}$	$\frac{\sigma_t}{\alpha_t^2 \lambda_k + \sigma_t^2}$	$\frac{\alpha_t \sigma_t (1 - \lambda_k)}{\alpha_t^2 \lambda_k + \sigma_t^2}$	$\frac{t \lambda_k - (1-t)}{t^2 \lambda_k + (1-t)^2}$
$1/\tau^*$	$\lambda_k + \sigma^2$	$\alpha_t^2 \lambda_k + \sigma_t^2$	$\alpha_t^2 \lambda_k + \sigma_t^2$	$\alpha_t^2 \lambda_k + \sigma_t^2$	$t^2 \lambda_k + (1-t)^2$

858

Denoiser / clean image prediction (EDM) loss

$$\|\mathbf{D}_\theta(\mathbf{x} + \sigma\epsilon; \sigma) - \mathbf{x}\|^2$$

859 Moments of input-output

$$\begin{aligned} \mu_x &= \mu, \mu_y = \mu \\ \Sigma_{xx} &= \text{Cov}(\mathbf{x} + \sigma\epsilon, \mathbf{x} + \sigma\epsilon) = \Sigma + \sigma^2 I \\ \Sigma_{yx} &= \text{Cov}(\mathbf{x}, \mathbf{x} + \sigma\epsilon) = \Sigma \end{aligned}$$

860 Optimum

$$\begin{aligned} \mathbf{W}^* &= \Sigma_{xx}^{-1} \Sigma_{yx} = (\Sigma + \sigma^2 I)^{-1} \Sigma \\ \mathbf{b}^* &= \mu_y - \mathbf{W}^* \mu_x \\ &= (I - \mathbf{W}^*) \mu \\ &= \sigma^2 (\Sigma + \sigma^2 I)^{-1} \mu \end{aligned}$$

Epsilon / noise prediction (EDM) loss

$$\|\mathbf{F}_\theta(\mathbf{x} + \sigma\epsilon; \sigma) - \epsilon\|^2$$

861 Moments of input-output

$$\begin{aligned} \mu_x &= \mu, \mu_y = 0 \\ \Sigma_{xx} &= \text{Cov}(\mathbf{x} + \sigma\epsilon, \mathbf{x} + \sigma\epsilon) = \Sigma + \sigma^2 I \\ \Sigma_{yx} &= \text{Cov}(\epsilon, \mathbf{x} + \sigma\epsilon) = \sigma I \end{aligned}$$

862 Optimum

$$\begin{aligned} \mathbf{W}^* &= \Sigma_{xx}^{-1} \Sigma_{yx} = \sigma (\Sigma + \sigma^2 I)^{-1} \\ \mathbf{b}^* &= \mu_y - \mathbf{W}^* \mu_x \\ &= -\mathbf{W}^* \mu \\ &= -\sigma (\Sigma + \sigma^2 I)^{-1} \mu \end{aligned}$$

Denoiser / clean image prediction loss

$$\|\mathbf{D}_\theta(\alpha_t \mathbf{x} + \sigma_t \epsilon; \sigma) - \mathbf{x}\|^2$$

863 Moments of input-output

$$\begin{aligned} \mu_x &= \alpha_t \mu, \mu_y = \mu \\ \Sigma_{xx} &= \text{Cov}(\alpha_t \mathbf{x} + \sigma_t \epsilon, \alpha_t \mathbf{x} + \sigma_t \epsilon) = \alpha_t^2 \Sigma + \sigma_t^2 I \\ \Sigma_{yx} &= \text{Cov}(\mathbf{x}, \alpha_t \mathbf{x} + \sigma_t \epsilon) = \alpha_t \Sigma \end{aligned}$$

864 Optimum

$$\begin{aligned}
\mathbf{W}^* &= \Sigma_{xx}^{-1} \Sigma_{yx} = \alpha_t (\alpha_t^2 \Sigma + \sigma_t^2 I)^{-1} \Sigma \\
\mathbf{b}^* &= \mu_y - \mathbf{W}^* \mu_x \\
&= \mu - \alpha_t \mathbf{W}^* \mu \\
&= (I - \alpha_t \mathbf{W}^*) \mu \\
&= (I - \alpha_t^2 \Sigma (\alpha_t^2 \Sigma + \sigma_t^2 I)^{-1}) \mu \\
&= \sigma_t^2 (\alpha_t^2 \Sigma + \sigma_t^2 I)^{-1} \mu
\end{aligned}$$

Epsilon / noise prediction loss

$$\|\mathbf{F}_\theta(\alpha_t \mathbf{x} + \sigma_t \epsilon; t) - \epsilon\|^2$$

865 Moments of input-output

$$\begin{aligned}
\mu_x &= \alpha_t \mu, \quad \mu_y = 0 \\
\Sigma_{xx} &= \text{Cov}(\alpha_t \mathbf{x} + \sigma_t \epsilon, \alpha_t \mathbf{x} + \sigma_t \epsilon) = \alpha_t^2 \Sigma + \sigma_t^2 I \\
\Sigma_{yx} &= \text{Cov}(\epsilon, \alpha_t \mathbf{x} + \sigma_t \epsilon) = \sigma_t I
\end{aligned}$$

866 Optimum

$$\begin{aligned}
\mathbf{W}^* &= \Sigma_{xx}^{-1} \Sigma_{yx} = \sigma_t (\alpha_t^2 \Sigma + \sigma_t^2 I)^{-1} \\
\mathbf{b}^* &= \mu_y - \mathbf{W}^* \mu_x \\
&= -\alpha_t \mathbf{W}^* \mu \\
&= -\alpha_t \sigma_t (\alpha_t^2 \Sigma + \sigma_t^2 I)^{-1} \mu
\end{aligned}$$

Velocity prediction loss

$$\|\mathbf{F}_\theta(\alpha_t \mathbf{x} + \sigma_t \epsilon; t) - (\alpha_t \epsilon - \sigma_t \mathbf{x})\|^2$$

867 Moments of input-output

$$\begin{aligned}
\mu_x &= \alpha_t \mu, \quad \mu_y = -\sigma_t \mu \\
\Sigma_{xx} &= \text{Cov}(\alpha_t \mathbf{x} + \sigma_t \epsilon, \alpha_t \mathbf{x} + \sigma_t \epsilon) = \alpha_t^2 \Sigma + \sigma_t^2 I \\
\Sigma_{yx} &= \text{Cov}(\alpha_t \epsilon - \sigma_t \mathbf{x}, \alpha_t \mathbf{x} + \sigma_t \epsilon) = -\alpha_t \sigma_t \Sigma + \alpha_t \sigma_t I = \alpha_t \sigma_t (I - \Sigma)
\end{aligned}$$

868 Optimum

$$\begin{aligned}
\mathbf{W}^* &= \Sigma_{xx}^{-1} \Sigma_{yx} \\
&= \alpha_t \sigma_t (\alpha_t^2 \Sigma + \sigma_t^2 I)^{-1} (I - \Sigma) \\
\mathbf{b}^* &= \mu_y - \mathbf{W}^* \mu_x \\
&= -\sigma_t \mu - \alpha_t \mathbf{W}^* \mu \\
&= -(\sigma_t I + \alpha_t \mathbf{W}^*) \mu \\
&= -\left(\sigma_t I + \alpha_t^2 \sigma_t (\alpha_t^2 \Sigma + \sigma_t^2 I)^{-1} (I - \Sigma)\right) \mu \\
&= -\sigma_t \left(I + \alpha_t^2 (\alpha_t^2 \Sigma + \sigma_t^2 I)^{-1} (I - \Sigma)\right) \mu \\
&= -\sigma_t \left(\alpha_t^2 \Sigma + \sigma_t^2 I + \alpha_t^2 (I - \Sigma)\right) (\alpha_t^2 \Sigma + \sigma_t^2 I)^{-1} \mu \\
&= -\sigma_t (\alpha_t^2 + \sigma_t^2) (\alpha_t^2 \Sigma + \sigma_t^2 I)^{-1} \mu
\end{aligned}$$

Flow matching loss

$$\mathcal{L} = \mathbb{E}_{\mathbf{x}_0 \sim \mathcal{N}(0, I), \mathbf{x}_1 \sim p_1} \|\mathbf{u}_\theta((1-t)\mathbf{x}_0 + t\mathbf{x}_1, t) - (\mathbf{x}_1 - \mathbf{x}_0)\|^2$$

869 Moments of input-output

$$\begin{aligned}
\mu_x &= t\mu, \quad \mu_y = \mu \\
\Sigma_{xx} &= \text{Cov}((1-t)\mathbf{x}_0 + t\mathbf{x}_1, (1-t)\mathbf{x}_0 + t\mathbf{x}_1) = t^2 \Sigma + (1-t)^2 I \\
\Sigma_{yx} &= \text{Cov}(\mathbf{x}_1 - \mathbf{x}_0, (1-t)\mathbf{x}_0 + t\mathbf{x}_1) = t\Sigma - (1-t)I
\end{aligned}$$

$$\begin{aligned}
\mathbf{W}^* &= \Sigma_{xx}^{-1} \Sigma_{yx} \\
&= (t^2 \Sigma + (1-t)^2 I)^{-1} (t \Sigma - (1-t) I) \\
\mathbf{b}^* &= \mu_y - \mathbf{W}^* \mu_x \\
&= \mu - \mathbf{W}^* t \mu \\
&= (I - t \mathbf{W}^*) \mu \\
&= (I - t(t^2 \Sigma + (1-t)^2 I)^{-1} (t \Sigma - (1-t) I)) \mu \\
&= (t^2 \Sigma + (1-t)^2 I - t(t \Sigma - (1-t) I)) (t^2 \Sigma + (1-t)^2 I)^{-1} \mu \\
&= ((1-t)^2 I + t(1-t) I) (t^2 \Sigma + (1-t)^2 I)^{-1} \mu \\
&= (1-t)(t^2 \Sigma + (1-t)^2 I)^{-1} \mu
\end{aligned}$$

871 C.5 General Analysis of the Sampling ODE

872 The diffusion sampling process per probability flow ODE is the following

$$\begin{aligned}\frac{d\mathbf{x}}{d\sigma} &= -\sigma \mathbf{s}(\mathbf{x}, \sigma) \\ &= -\frac{\mathbf{D}(\mathbf{x}, \sigma) - \mathbf{x}}{\sigma}\end{aligned}\quad (43)$$

873 given

$$\mathbf{s}(\mathbf{x}, \sigma) = \frac{\mathbf{D}(\mathbf{x}, \sigma) - \mathbf{x}}{\sigma^2}$$

874 Intuitively, when the denoiser is a linear function of \mathbf{x} , then the sampling ODE is also a linear
875 (time-varying) dynamic system with respect to \mathbf{x} .

876 We try to solve the sampling dynamics mode by mode on the eigenbasis of covariance Σ ,

$$\frac{d}{d\sigma} \mathbf{x} = -\frac{\mathbf{W}_\sigma \mathbf{x} + \mathbf{b}_\sigma - \mathbf{x}}{\sigma} \quad (44)$$

$$\frac{d}{d\sigma} \mathbf{u}_k^T \mathbf{x} = -\frac{1}{\sigma} (\mathbf{u}_k^T \mathbf{W}_\sigma \mathbf{x} + \mathbf{u}_k^T \mathbf{b}_\sigma - \mathbf{u}_k^T \mathbf{x}) \quad (45)$$

877 When \mathbf{W} is diagonalizable by the eigenbasis of data,

$$\mathbf{W}_\sigma = \sum_k \psi_k(\sigma) \mathbf{u}_k \mathbf{u}_k^T \quad (46)$$

$$\mathbf{b}_\sigma = \sum_k b_k(\sigma) \mathbf{u}_k \quad (47)$$

$$\mathbf{x}(\sigma) = \sum_k c_k(\sigma) \mathbf{u}_k \quad (48)$$

878 We can project the sampling ODE onto eigenbasis, i.e.

$$\frac{d}{d\sigma} \mathbf{u}_k^T \mathbf{x} = -\frac{1}{\sigma} ((\psi_k(\sigma) - 1) \mathbf{u}_k^T \mathbf{x} + b_k(\sigma))$$

879

$$\frac{d}{d\sigma} c_k(\sigma) = -\frac{1}{\sigma} ((\psi_k(\sigma) - 1) c_k(\sigma) + b_k(\sigma))$$

880

$$\frac{d}{d\sigma} c_k(\sigma) + \left(\frac{\psi_k(\sigma) - 1}{\sigma}\right) c_k(\sigma) = -\frac{1}{\sigma} b_k(\sigma)$$

881 This can be solved via the Armours formula for first-order ODE.

$$\frac{dy}{dx} + P(x)y = Q(x)$$

$$y = e^{-\int^x P(\lambda) d\lambda} \left[\int^x e^{\int^\lambda P(\varepsilon) d\varepsilon} Q(\lambda) d\lambda + C \right]$$

$$c_k(\sigma) = e^{-\int^\sigma \frac{\psi_k(\lambda)-1}{\lambda} d\lambda} \left[\int^\sigma -\frac{b_k(\lambda)}{\lambda} e^{\int^\lambda \frac{\psi_k(\varepsilon)-1}{\varepsilon} d\varepsilon} d\lambda + C \right]$$

$$c_k(\sigma) = e^{-\int_{\sigma_T}^\sigma \frac{\psi_k(\lambda)-1}{\lambda} d\lambda} \left[c_k(\sigma_T) + \int_{\sigma_T}^\sigma -\frac{b_k(\lambda)}{\lambda} e^{\int^\lambda \frac{\psi_k(\varepsilon)-1}{\varepsilon} d\varepsilon} d\lambda \right]$$

$$c_k(\sigma) = e^{-\int_{\sigma_T}^\sigma \frac{\psi_k(\lambda)-1}{\lambda} d\lambda} c_k(\sigma_T) + \int_{\sigma_T}^\sigma -\frac{b_k(\lambda)}{\lambda} e^{-\int_\lambda^\sigma \frac{\psi_k(\varepsilon)-1}{\varepsilon} d\varepsilon} d\lambda$$

882 The solution reads

$$c_k(\sigma) = A_k(\sigma; \sigma_T) c_k(\sigma_T) + B_k(\sigma; \sigma_T) \quad (49)$$

$$A_k(\sigma; \sigma_T) = e^{-\int_{\sigma_T}^{\sigma} \frac{\psi_k(\lambda)-1}{\lambda} d\lambda} \quad (50)$$

$$B_k(\sigma; \sigma_T) = \int_{\sigma_T}^{\sigma} -\frac{b_k(\lambda)}{\lambda} e^{-\int_{\lambda}^{\sigma} \frac{\psi_k(\varepsilon)-1}{\varepsilon} d\varepsilon} d\lambda \quad (51)$$

883 Consider the function

$$\Phi_k(\sigma) = \exp\left(-\int_0^{\sigma} \frac{\psi_k(\lambda)-1}{\lambda} d\lambda\right)$$

884 Then the integration functions can be expressed as

$$A_k(\sigma; \sigma_T) = \Phi_k(\sigma) / \Phi_k(\sigma_T) \quad (52)$$

$$B_k(\sigma; \sigma_T) = \int_{\sigma_T}^{\sigma} -\frac{b_k(\lambda)}{\lambda} \Phi_k(\sigma) / \Phi_k(\lambda) d\lambda \quad (53)$$

885 By initial noise distribution, $c_k(\sigma_T) \sim \mathcal{N}(0, \sigma_T^2)$, the variance of $c_k(\sigma)$ can be written down

$$Var[c_k(\sigma)] = \sigma_T^2 \exp\left(-2 \int_{\sigma_T}^{\sigma} \frac{\psi_k(\lambda)-1}{\lambda} d\lambda\right) \quad (54)$$

$$= \sigma_T^2 \left(\frac{\Phi_k(\sigma)}{\Phi_k(\sigma_T)}\right)^2 \quad (55)$$

886 Since $\mathbb{E}[c_k(\sigma_T)] = 0$,

$$\mathbb{E}[c_k(\sigma)] = e^{-\int_{\sigma_T}^{\sigma} \frac{\psi_k(\lambda)-1}{\lambda} d\lambda} \left[\int_{\sigma_T}^{\sigma} -\frac{b_k(\lambda)}{\lambda} e^{\int_{\sigma_T}^{\lambda} \frac{\psi_k(\varepsilon)-1}{\varepsilon} d\varepsilon} d\lambda \right] \quad (56)$$

$$= -\int_{\sigma_T}^{\sigma} \frac{b_k(\lambda)}{\lambda} e^{-\int_{\lambda}^{\sigma} \frac{\psi_k(\varepsilon)-1}{\varepsilon} d\varepsilon} d\lambda \quad (57)$$

$$= -\int_{\sigma_T}^{\sigma} \frac{b_k(\lambda)}{\lambda} \frac{\Phi_k(\sigma)}{\Phi_k(\lambda)} d\lambda \quad (58)$$

$$= B_k(\sigma; \sigma_T) \quad (59)$$

887 Since \mathbf{x}_{σ} is a linear transformation of a Gaussian random variable, the distribution of \mathbf{x}_{σ} is also
888 Gaussian $\mathbf{x}_{\sigma} \sim \mathcal{N}(\tilde{\mu}, \tilde{\Sigma})$ with the following mean and covariance.

$$\tilde{\mu} = \sum_k B_k(\sigma; \sigma_T) \mathbf{u}_k \quad (60)$$

$$\tilde{\Sigma} = \sum_k \sigma_T^2 \left(\frac{\Phi_k(\sigma)}{\Phi_k(\sigma_T)}\right)^2 \mathbf{u}_k \quad (61)$$

889 C.6 KL Divergence Computation

890 KL divergence between two multivariate Gaussian distributions is,

$$KL(\mathcal{N}(\mu_1, \Sigma_1) \parallel \mathcal{N}(\mu_2, \Sigma_2)) \quad (62)$$

$$= \int \left[\frac{1}{2} \log \frac{|\Sigma_2|}{|\Sigma_1|} - \frac{1}{2} (x - \mu_1)^T \Sigma_1^{-1} (x - \mu_1) + \frac{1}{2} (x - \mu_2)^T \Sigma_2^{-1} (x - \mu_2) \right] \times p(x) dx \quad (63)$$

$$= \frac{1}{2} \log \frac{|\Sigma_2|}{|\Sigma_1|} - \frac{1}{2} \text{tr} \left\{ E[(x - \mu_1)(x - \mu_1)^T] \Sigma_1^{-1} \right\} + \frac{1}{2} E[(x - \mu_2)^T \Sigma_2^{-1} (x - \mu_2)] \quad (64)$$

$$= \frac{1}{2} \log \frac{|\Sigma_2|}{|\Sigma_1|} - \frac{1}{2} \text{tr} \{ \mathbf{I}_d \} + \frac{1}{2} (\mu_1 - \mu_2)^T \Sigma_2^{-1} (\mu_1 - \mu_2) + \frac{1}{2} \text{tr} \{ \Sigma_2^{-1} \Sigma_1 \} \quad (65)$$

$$= \frac{1}{2} \left[\log \frac{|\Sigma_2|}{|\Sigma_1|} - d + \text{tr} \{ \Sigma_2^{-1} \Sigma_1 \} + (\mu_2 - \mu_1)^T \Sigma_2^{-1} (\mu_2 - \mu_1) \right]. \quad (66)$$

891 This formula further simplifies when Σ_2 and Σ_1 share the same eigenbasis. We can write their
 892 eigendecomposition as $\Sigma_2 = U\Lambda_2U^T$, $\Sigma_1 = U\Lambda_1U^T$,

$$KL(\mathcal{N}(\mu_1, \Sigma_1) \parallel \mathcal{N}(\mu_2, \Sigma_2)) = \frac{1}{2} \left[\log \frac{|\Lambda_2|}{|\Lambda_1|} - d + \text{tr}\{\Lambda_2^{-1}\Lambda_1\} + (\mu_2 - \mu_1)^T U\Lambda_2^{-1}U^T(\mu_2 - \mu_1) \right] \quad (67)$$

$$= \frac{1}{2} \left[\log \frac{\prod_k \lambda_{2,k}}{\prod_k \lambda_{1,k}} - d + \sum_k \frac{\lambda_{1,k}}{\lambda_{2,k}} + (\mu_2 - \mu_1)^T U\Lambda_2^{-1}U^T(\mu_2 - \mu_1) \right] \quad (68)$$

$$= \frac{1}{2} \left[\sum_k \log \frac{\lambda_{2,k}}{\lambda_{1,k}} - d + \sum_k \frac{\lambda_{1,k}}{\lambda_{2,k}} + (\mu_2 - \mu_1)^T U\Lambda_2^{-1}U^T(\mu_2 - \mu_1) \right] \quad (69)$$

893 In more explicit form

$$KL(\mathcal{N}(\mu_1, \Sigma_1) \parallel \mathcal{N}(\mu_2, \Sigma_2)) = \frac{1}{2} \left[\sum_k \log \frac{\lambda_{2,k}}{\lambda_{1,k}} - d + \sum_k \frac{\lambda_{1,k}}{\lambda_{2,k}} + (\mu_2 - \mu_1)^T \sum_k \frac{\mathbf{u}_k \mathbf{u}_k^T}{\lambda_{2,k}} (\mu_2 - \mu_1) \right] \quad (70)$$

$$= \frac{1}{2} \sum_k \left[\log \frac{\lambda_{2,k}}{\lambda_{1,k}} + \frac{\lambda_{1,k}}{\lambda_{2,k}} - 1 + \frac{(\mathbf{u}_k^T (\mu_2 - \mu_1))^2}{\lambda_{2,k}} \right] \quad (71)$$

894 If they share the same mean $\mu_1 = \mu_2$ it simplifies even further

$$KL = \frac{1}{2} \left[\sum_k \frac{\lambda_{1,k}}{\lambda_{2,k}} - \sum_k \log \frac{\lambda_{1,k}}{\lambda_{2,k}} - d \right]$$

895 which has unique minimizer when $\frac{\lambda_{1,k}}{\lambda_{2,k}} = 1$.

896 Thus, we can compute the contribution to KL mode by mode. We denote the contribution from mode
 897 k as

$$KL_k = \frac{1}{2} \left(\frac{\lambda_{1,k}}{\lambda_{2,k}} - \log \frac{\lambda_{1,k}}{\lambda_{2,k}} - 1 \right) \quad (72)$$

898 Thus for Gaussian data that share the same mean, the KL divergence can be reduced to the ratio of
 899 generated and true variance along each principal axes of data.

D Detailed Derivations for the One-Layer Linear Model

D.1 Zero-mean data: Exponential Converging Training Dynamics

If $\mu = 0$, then the only problem becomes **learning the covariance of data**. The gradient to weights and bias decouples

$$\nabla_{\mathbf{b}} \mathcal{L} = 2\mathbf{b} \quad (73)$$

$$\nabla_{\mathbf{W}} \mathcal{L} = -2\mathbf{\Sigma} + 2\mathbf{W}(\sigma^2 \mathbf{I} + \mathbf{\Sigma}) \quad (74)$$

The solution of \mathbf{b} is an exponential decay

$$\frac{d}{d\tau} \mathbf{b} = -2\eta \mathbf{b} \quad (75)$$

$$\mathbf{b}(\tau) = \mathbf{b}^0 \exp(-2\eta\tau) \quad (76)$$

The solution of \mathbf{W} can be solved by projecting onto eigen basis of $\mathbf{\Sigma}$

$$\nabla_{\mathbf{W}} \mathcal{L} \cdot \mathbf{u}_k = -2\mathbf{\Sigma} \mathbf{u}_k + 2\mathbf{W}(\sigma^2 \mathbf{I} + \mathbf{\Sigma}) \mathbf{u}_k \quad (77)$$

$$= -2\lambda_k \mathbf{u}_k + 2(\sigma^2 + \lambda_k) \mathbf{W} \mathbf{u}_k \quad (78)$$

Thus,

$$\frac{d}{d\tau} (\mathbf{W} \mathbf{u}_k) = -\eta [-2\lambda_k \mathbf{u}_k + 2(\sigma^2 + \lambda_k) (\mathbf{W} \mathbf{u}_k)]$$

Define variable $\mathbf{v}_k = \mathbf{W} \mathbf{u}_k$

$$\frac{d}{d\tau} \mathbf{v}_k = 2\eta \lambda_k \mathbf{u}_k - 2\eta(\sigma^2 + \lambda_k) \mathbf{v}_k \quad (79)$$

$$= 2\eta(\sigma^2 + \lambda_k) \left(\frac{\lambda_k}{(\sigma^2 + \lambda_k)} \mathbf{u}_k - \mathbf{v}_k \right) \quad (80)$$

$$\mathbf{v}_k(\tau) = \frac{\lambda_k}{(\sigma^2 + \lambda_k)} \mathbf{u}_k + \left(\mathbf{v}_k^0 - \frac{\lambda_k}{(\sigma^2 + \lambda_k)} \mathbf{u}_k \right) \exp(-2\eta(\sigma^2 + \lambda_k)\tau)$$

Since

$$[\mathbf{v}_k \dots] = \mathbf{W} [\mathbf{u}_k \dots] \quad (81)$$

$$\mathbf{V} \mathbf{U}^T = \mathbf{W} \quad (82)$$

$$\mathbf{W} = \sum_k \mathbf{v}_k \mathbf{u}_k^T \quad (83)$$

The full solution of \mathbf{W} reads

$$\mathbf{W}(\tau) = \sum_k \mathbf{v}_k(\tau) \mathbf{u}_k^T \quad (84)$$

$$= \sum_k \frac{\lambda_k}{(\sigma^2 + \lambda_k)} \mathbf{u}_k \mathbf{u}_k^T + \sum_k \left(\mathbf{v}_k^0 - \frac{\lambda_k}{(\sigma^2 + \lambda_k)} \mathbf{u}_k \right) \mathbf{u}_k^T e^{-2\eta(\sigma^2 + \lambda_k)\tau} \quad (85)$$

$$= \sum_k \frac{\lambda_k}{(\sigma^2 + \lambda_k)} \mathbf{u}_k \mathbf{u}_k^T (1 - e^{-2\eta(\sigma^2 + \lambda_k)\tau}) + \sum_k \mathbf{v}_k^0 \mathbf{u}_k^T e^{-2\eta(\sigma^2 + \lambda_k)\tau} \quad (86)$$

$$= \sum_k \frac{\lambda_k}{(\sigma^2 + \lambda_k)} \mathbf{u}_k \mathbf{u}_k^T (1 - e^{-2\eta(\sigma^2 + \lambda_k)\tau}) + \mathbf{W}(0) \sum_k \mathbf{u}_k \mathbf{u}_k^T e^{-2\eta(\sigma^2 + \lambda_k)\tau} \quad (87)$$

$$= \mathbf{W}^* + \sum_k \left(\mathbf{v}_k^0 - \frac{\lambda_k}{(\sigma^2 + \lambda_k)} \mathbf{u}_k \right) \mathbf{u}_k^T e^{-2\eta(\sigma^2 + \lambda_k)\tau} \quad (88)$$

$$= \mathbf{W}^* + \sum_k \left(\mathbf{W}(0) \mathbf{u}_k - \frac{\lambda_k}{(\sigma^2 + \lambda_k)} \mathbf{u}_k \right) \mathbf{u}_k^T e^{-2\eta(\sigma^2 + \lambda_k)\tau} \quad (89)$$

where $\mathbf{v}_k^0 := \mathbf{W}(0) \mathbf{u}_k$.

Remark D.1. • The weight matrix \mathbf{W} converges to the final target mode by mode.

- The deviation along each eigen dimensions decays at different rate depending on the eigenvalue.
- The deviation on eigen mode \mathbf{u}_k has the time constant $(2\eta(\sigma^2 + \lambda_k))^{-1}$ i.e. the larger eigen dimensions will be learned faster.
- While the “non-resolved” dimensions will learn at the same speed $\sim (2\eta\sigma^2)^{-1}$
- Comparing across noise scale σ , the larger noise scales will be learned faster.

Score estimation error dynamics Consider a target quantity of interest, i.e. difference of the score approximator from the true score.

First, under the $\mu = 0$ assumption, we have

$$E_s = \mathbb{E}_{\mathbf{x}} \|\mathbf{s}(\mathbf{x}) - \mathbf{s}^*(\mathbf{x})\|^2 = \frac{1}{\sigma^4} \left[\|b - b^*\|^2 + \text{Tr}[(\mathbf{W} - \mathbf{W}^*)^T (\mathbf{W} - \mathbf{W}^*) (\boldsymbol{\Sigma} + \sigma^2 \mathbf{I})] \right] \quad (90)$$

The deviations can be expressed as

$$b - b^* = b^0 \exp(-2\eta\tau) \quad (91)$$

$$\mathbf{W} - \mathbf{W}^* = \sum_k \left(\mathbf{v}_k^0 - \frac{\lambda_k}{(\sigma^2 + \lambda_k)} \mathbf{u}_k \right) \mathbf{u}_k^T e^{-2\eta(\sigma^2 + \lambda_k)\tau} \quad (92)$$

with the initial projection $\mathbf{v}_k^0 := \mathbf{W}(0)\mathbf{u}_k$

$$E_s = \frac{1}{\sigma^4} \left[(b^0)^2 \exp(-4\eta\tau) + \sum_k (\sigma^2 + \lambda_k) \left\| \mathbf{v}_k^0 - \frac{\lambda_k}{(\sigma^2 + \lambda_k)} \mathbf{u}_k \right\|^2 e^{-4\eta(\sigma^2 + \lambda_k)\tau} \right] \quad (93)$$

$$= \frac{1}{\sigma^4} \left[(\delta_b)^2 \exp(-4\eta\tau) + \sum_k (\sigma^2 + \lambda_k) \|\delta_{\mathbf{k}}\|^2 e^{-4\eta(\sigma^2 + \lambda_k)\tau} \right] \quad (94)$$

$$\delta_{\mathbf{k}} := \mathbf{W}(0)\mathbf{u}_k - \frac{\lambda_k}{(\sigma^2 + \lambda_k)} \mathbf{u}_k \quad (95)$$

$$\delta_b := b^0 \quad (96)$$

This provides us with the exact formula for error decay during training.

Denoiser estimation error dynamics

$$E_D = \mathbb{E}_{\mathbf{x}} \|\mathbf{D}(\mathbf{x}) - \mathbf{D}^*(\mathbf{x})\|^2 \quad (97)$$

$$= \sigma^4 E_s \quad (98)$$

$$= (\delta_b)^2 \exp(-4\eta\tau) + \sum_k (\sigma^2 + \lambda_k) \|\delta_{\mathbf{k}}\|^2 e^{-4\eta(\sigma^2 + \lambda_k)\tau} \quad (99)$$

Training loss dynamics Under $\mu = 0$ assumption, we have the training loss is basically the true denoiser estimation error plus a constant term $\sigma^2 \text{Tr}[G_{\boldsymbol{\Sigma}}(\sigma^2)]$

$$\mathcal{L}_{\mu=0} = \|b - (\mathbf{I} - \mathbf{W})\mu\|_2^2 + \text{Tr}[(\mathbf{W} - \mathbf{W}^*)(\sigma^2 \mathbf{I} + \boldsymbol{\Sigma})(\mathbf{W} - \mathbf{W}^*)^T] + \sigma^2 \text{Tr}[\boldsymbol{\Sigma}(\sigma^2 \mathbf{I} + \boldsymbol{\Sigma})^{-1}] \quad (100)$$

$$= \|b\|_2^2 + \text{Tr}[(\mathbf{W} - \mathbf{W}^*)^T (\mathbf{W} - \mathbf{W}^*) (\sigma^2 \mathbf{I} + \boldsymbol{\Sigma})] + \sigma^2 \text{Tr}[\boldsymbol{\Sigma}(\sigma^2 \mathbf{I} + \boldsymbol{\Sigma})^{-1}] \quad (101)$$

$$= E_D + \sigma^2 \text{Tr}[\boldsymbol{\Sigma}(\sigma^2 \mathbf{I} + \boldsymbol{\Sigma})^{-1}] \quad (102)$$

D.1.1 Discrete time Gradient descent dynamics

When the dynamics is discrete-time gradient descent instead of gradient flow, we have,

$$\nabla_{\mathbf{b}} \mathcal{L} = 2\mathbf{b} \quad (103)$$

$$\nabla_{\mathbf{W}} \mathcal{L} = -2\boldsymbol{\Sigma} + 2\mathbf{W}(\sigma^2 \mathbf{I} + \boldsymbol{\Sigma}) \quad (104)$$

929 The GD update equation reads, with η the learning rate

$$\mathbf{b}_{t+1} - \mathbf{b}_t = -\eta \nabla_{\mathbf{b}_t} \mathcal{L} \quad (105)$$

$$\mathbf{W}_{t+1} - \mathbf{W}_t = -\eta \nabla_{\mathbf{W}_t} \mathcal{L} \quad (106)$$

$$\mathbf{b}_{t+1} = (1 - 2\eta) \mathbf{b}_t \quad (107)$$

$$\mathbf{W}_{t+1} = \mathbf{W}_t - 2\eta(-\Sigma + \mathbf{W}_t(\sigma^2 \mathbf{I} + \Sigma)) \quad (108)$$

$$= 2\eta \Sigma + \mathbf{W}_t(\mathbf{I} - 2\eta\sigma^2 \mathbf{I} - 2\eta \Sigma) \quad (109)$$

$$= 2\eta \Sigma + \mathbf{W}_t((1 - 2\eta\sigma^2) \mathbf{I} - 2\eta \Sigma) \quad (110)$$

930 For the weight dynamics, we have

$$\mathbf{W}_{t+1} \mathbf{u}_k = 2\eta \Sigma \mathbf{u}_k + \mathbf{W}_t((1 - 2\eta\sigma^2) \mathbf{I} - 2\eta \Sigma) \mathbf{u}_k \quad (111)$$

$$= 2\eta \lambda_k \mathbf{u}_k + \mathbf{W}_t \mathbf{u}_k (1 - 2\eta\sigma^2 - 2\eta \lambda_k) \quad (112)$$

931 This iteration is exponentially converging to the fixed point $\frac{\lambda_k}{\sigma^2 + \lambda_k}$.

$$\mathbf{W}_{t+1} \mathbf{u}_k - \frac{\lambda_k}{\sigma^2 + \lambda_k} \mathbf{u}_k = (2\eta \lambda_k - \frac{\lambda_k}{\sigma^2 + \lambda_k}) \mathbf{u}_k + \mathbf{W}_t \mathbf{u}_k (1 - 2\eta\sigma^2 - 2\eta \lambda_k) \quad (113)$$

$$= (2\eta(\sigma^2 + \lambda_k) - 1) \frac{\lambda_k}{\sigma^2 + \lambda_k} \mathbf{u}_k + \mathbf{W}_t \mathbf{u}_k (1 - 2\eta\sigma^2 - 2\eta \lambda_k) \quad (114)$$

$$= (\mathbf{W}_t \mathbf{u}_k - \frac{\lambda_k}{\sigma^2 + \lambda_k} \mathbf{u}_k) (1 - 2\eta\sigma^2 - 2\eta \lambda_k) \quad (115)$$

932 Thus

$$\mathbf{W}_t \mathbf{u}_k = \frac{\lambda_k}{\sigma^2 + \lambda_k} \mathbf{u}_k + (\mathbf{W}_0 \mathbf{u}_k - \frac{\lambda_k}{\sigma^2 + \lambda_k} \mathbf{u}_k) (1 - 2\eta\sigma^2 - 2\eta \lambda_k)^t \quad (116)$$

$$\mathbf{b}_t = \mathbf{b}_0 (1 - 2\eta)^t \quad (117)$$

$$\mathbf{W}_t = \sum_k \frac{\lambda_k}{\sigma^2 + \lambda_k} \mathbf{u}_k \mathbf{u}_k^T + (\mathbf{W}_0 - \sum_k \frac{\lambda_k}{\sigma^2 + \lambda_k} \mathbf{u}_k \mathbf{u}_k^T) (1 - 2\eta\sigma^2 - 2\eta \lambda_k)^t \quad (118)$$

933 So, there is no significant change from the continuous-time version.

934 **D.1.2 Special parametrization: residual connection**

935 Consider a special parametrization of weights

$$\mathbf{W} = c_{skip} \mathbf{I} + c_{out} \mathbf{W}'$$

936 It's easy to derive the dynamics of the new variables via chain rule,

$$\frac{\partial \mathbf{W}}{\partial \mathbf{W}'} = c_{out}, \quad (119)$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}'} = c_{out} \frac{\partial \mathcal{L}}{\partial \mathbf{W}}. \quad (120)$$

937 With the original gradient

$$\nabla_{\mathbf{b}} \mathcal{L} = 2(\mathbf{b} - (\mathbf{I} - \mathbf{W})\mu) \quad (121)$$

$$\nabla_{\mathbf{W}} \mathcal{L} = -2\Sigma + 2\mathbf{W}(\sigma^2 \mathbf{I} + \Sigma) + [2\mathbf{W}\mu\mu^T + 2(\mathbf{b} - \mu)\mu^T] \quad (122)$$

938 and zero mean case

$$\nabla_{\mathbf{b}} \mathcal{L} = 2\mathbf{b} \quad (123)$$

$$\nabla_{\mathbf{W}} \mathcal{L} = -2\Sigma + 2\mathbf{W}(\sigma^2 \mathbf{I} + \Sigma) \quad (124)$$

the gradient to new parameters

$$\nabla_{\mathbf{W}'} \mathcal{L} = c_{out} \nabla_{\mathbf{W}} \mathcal{L} = 2c_{out}(-\Sigma + (c_{skip} \mathbf{I} + c_{out} \mathbf{W}')(\sigma^2 \mathbf{I} + \Sigma)) \quad (125)$$

$$= 2c_{out}(-\Sigma + c_{skip}(\sigma^2 \mathbf{I} + \Sigma) + c_{out} \mathbf{W}'(\sigma^2 \mathbf{I} + \Sigma)) \quad (126)$$

$$= 2c_{out}(-\Sigma + c_{skip}(\sigma^2 \mathbf{I} + \Sigma) + c_{out} \mathbf{W}'(\sigma^2 \mathbf{I} + \Sigma)) \quad (127)$$

$$= 2c_{out}(c_{skip}(\sigma^2 \mathbf{I} + \Sigma) - \Sigma + c_{out} \mathbf{W}'(\sigma^2 \mathbf{I} + \Sigma)) \quad (128)$$

$$\frac{d\mathbf{W}'}{d\tau} = -\eta \nabla_{\mathbf{W}'} \mathcal{L} \quad (129)$$

$$\frac{1}{2\eta c_{out}} \frac{d\mathbf{W}'}{d\tau} = -(c_{skip}(\sigma^2 \mathbf{I} + \Sigma) - \Sigma + c_{out} \mathbf{W}'(\sigma^2 \mathbf{I} + \Sigma)) \quad (130)$$

$$\mathbf{W}'^* = \frac{1}{c_{out}}(\Sigma - c_{skip}(\sigma^2 \mathbf{I} + \Sigma))(\sigma^2 \mathbf{I} + \Sigma)^{-1} \quad (131)$$

$$= \frac{1}{c_{out}}(\Sigma(\sigma^2 \mathbf{I} + \Sigma)^{-1} - c_{skip} \mathbf{I}) \quad (132)$$

$$\mathbf{W}'(\tau) \mathbf{u}_k = \mathbf{W}'(0) \mathbf{u}_k \exp(-2\eta\tau c_{out}^2(\sigma^2 + \lambda_k)) + \quad (133)$$

$$\mathbf{u}_k \frac{\lambda_k - c_{skip}(\sigma^2 + \lambda_k)}{c_{out}(\sigma^2 + \lambda_k)} (1 - \exp(-2\eta\tau c_{out}^2(\sigma^2 + \lambda_k)))$$

The solution to the new weights reads

$$\mathbf{W}'(\tau) = (\mathbf{W}'(0) - \mathbf{W}'^*) \sum_k \mathbf{u}_k \mathbf{u}_k^T \exp(-2\eta\tau c_{out}^2(\sigma^2 + \lambda_k)) + \mathbf{W}'^* \quad (134)$$

$$\mathbf{W}(\tau) = c_{skip} \mathbf{I} + c_{out} \mathbf{W}'(\tau) \quad (135)$$

$$= c_{skip} \mathbf{I} + c_{out}(\mathbf{W}'(0) - \mathbf{W}'^*) \sum_k \mathbf{u}_k \mathbf{u}_k^T \exp(-2\eta\tau c_{out}^2(\sigma^2 + \lambda_k)) + c_{out} \mathbf{W}'^* \quad (136)$$

$$= \Sigma(\sigma^2 \mathbf{I} + \Sigma)^{-1} + c_{out}(\mathbf{W}'(0) - \mathbf{W}'^*) \sum_k \mathbf{u}_k \mathbf{u}_k^T \exp(-2\eta\tau c_{out}^2(\sigma^2 + \lambda_k)) \quad (137)$$

$$= \mathbf{W}^* + c_{out}(\mathbf{W}'(0) - \mathbf{W}'^*) \sum_k \mathbf{u}_k \mathbf{u}_k^T \exp(-2\eta\tau c_{out}^2(\sigma^2 + \lambda_k)) \quad (138)$$

$$= \mathbf{W}^* + (\mathbf{W}(0) - \mathbf{W}^*) \sum_k \mathbf{u}_k \mathbf{u}_k^T \exp(-2\eta\tau c_{out}^2(\sigma^2 + \lambda_k)) \quad (139)$$

The only difference is scaling the learning rate by a factor of c_{out}^2 . Also potentially depend on whether we choose to initialize $\mathbf{W}(0)$ or $\mathbf{W}'(0)$ from a fixed distribution, we would get different initial value for the dynamics.

D.2 General non-centered distribution: Interaction of mean and covariance learning

Summary of results for non-centered case When $\mu \neq 0$, the gradients to \mathbf{W} and \mathbf{b} become entangled (5), resulting in a coupled linear dynamic system as follows.

Proposition D.2 (Learning dynamics of linear denoiser, non centered case). *Gradient flow (GF) is equivalent to the following ODE, with redefined dynamic variables, $\mathbf{v}_k(\tau) = \mathbf{W}(\tau) \mathbf{u}_k$, $\bar{\mathbf{b}}(\tau) = \mathbf{b}(\tau) - \mu$. Denote overlap $m_k := \mathbf{u}_k^T \mu$,*

$$\frac{1}{2\eta} \frac{d}{d\tau} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \dot{\bar{\mathbf{b}}} \end{bmatrix} = -M \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \dot{\bar{\mathbf{b}}} \end{bmatrix} + \begin{bmatrix} \lambda_1 \mathbf{u}_1 \\ \lambda_2 \mathbf{u}_2 \\ 0 \end{bmatrix} \quad (140)$$

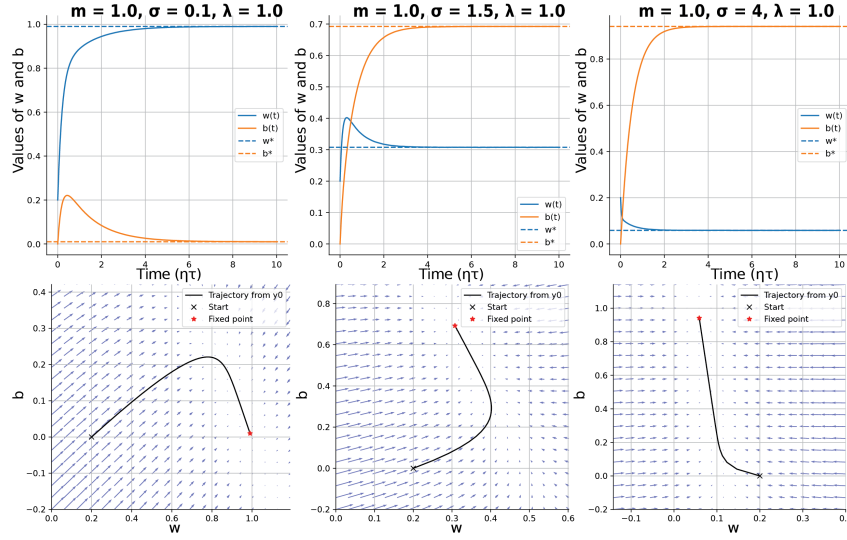


Figure 21: **Interaction of mean and covariance learning.** **Top** solution to the w, b dynamics under different noise level $\sigma \in \{0.1, 1.5, 4\}$. **Bottom** Phase portraits corresponding to the two-d system. ($m = 1, \lambda_k = 1$)

954 with a fixed dynamic matrix M defined by \otimes Kronecker product.

$$M := \begin{bmatrix} \sigma^2 + \lambda_1 + m_1^2 & m_1 m_2 & \cdot & m_1 \\ m_1 m_2 & \sigma^2 + \lambda_2 + m_2^2 & \cdot & m_2 \\ \cdots & \cdots & \cdots & \cdot \\ m_1 & m_2 & \cdot & 1 \end{bmatrix} \otimes \mathbf{I}_d$$

$$:= \tilde{M} \otimes \mathbf{I}_d \quad (141)$$

955 *Remark D.3.* The dynamics matrix \tilde{M} has a *rank-one plus diagonal structure*, specifically, it is
 956 the diagonal dynamics matrix in Eq.7, perturbed by the outer product of the overlap vector m_k .
 957 The eigenvalues of such matrix can be efficiently solved by numerical algebra, with eigenvectors
 958 expressed by Bunch–Nielsen–Sorensen formula [53, 54]. Without a closed-form formula, we have to
 959 resort to numerical simulation and low-dimensional examples to gain further insights. We can see the
 960 coupling of \mathbf{W} and \mathbf{b} dynamics comes from the overlap of mean and principal component $m_k = \mathbf{u}_k^\top \mu$.
 961 When certain eigenmode has no overlap, $m_k = 0$, the corresponding weight projection $\mathbf{v}_k(\tau)$ will
 962 follow the same dynamics as the zero-mean case, i.e. exponentially converge to the optimal solution
 963 $\lambda_k / (\lambda_k + \sigma^2) \mathbf{u}_k$. When the overlap is non-zero $m_k \neq 0$, it will induce interaction between $\mathbf{v}_k(\tau)$
 964 and $\mathbf{b}(\tau)$ and non-monotonic dynamics.

965 **Two dimensional example** Here, we show a low-dimensional example illustrating the interaction
 966 between the bias and one eigenmode in the weight. Consider the case where distribution mean μ lies
 967 on the direction of a PC \mathbf{u}_k . Then only the \mathbf{u}_k mode of weights interacts with the distribution mean,
 968 resulting in a two-dimensional linear system, parametrized by noise scale σ , variance of mode λ and
 969 amount of alignment m . Let the dynamic variable be scalars w, b , $\mathbf{v}_k = w(\tau) \mathbf{u}_k$, $\mathbf{b} = b(\tau) \mathbf{u}_k$.

$$\frac{1}{2\eta} \frac{d}{d\tau} \begin{bmatrix} w \\ b \end{bmatrix} = - \begin{bmatrix} m^2 + \sigma^2 + \lambda & m \\ m & 1 \end{bmatrix} \begin{bmatrix} w \\ b \end{bmatrix} + \begin{bmatrix} \lambda + m^2 \\ m \end{bmatrix}$$

970 The phase diagram and dynamics depending on the noise scale are showed (Fig. 21): At larger σ
 971 values, the dynamics of weights w will be much faster than b , basically, w gets dynamically captured
 972 by b , while b slowly relaxed to the optimal value. At small σ values, the dynamics timescale of w and
 973 b will be closer to each other, and b will usually have non-monotonic transient dynamics. When σ^2
 974 and λ are comparable, w will have non-monotonic dynamics.

975 **Full Derivation of non-centered case** In this full case, the dynamics of \mathbf{b} , \mathbf{W} are coupled

$$\nabla_{\mathbf{b}}\mathcal{L} = 2(\mathbf{b} - \mu + \mathbf{W}\mu) \quad (142)$$

$$\nabla_{\mathbf{W}}\mathcal{L} = -2\Sigma + 2\mathbf{W}(\sigma^2\mathbf{I} + \Sigma) + 2(\mathbf{b} - \mu + \mathbf{W}\mu)\mu^T \quad (143)$$

$$= -2\Sigma + 2\mathbf{W}(\sigma^2\mathbf{I} + \Sigma) + \nabla_{\mathbf{b}}\mathcal{L} \cdot \mu^T \quad (144)$$

$$= \nabla_{\mathbf{W}}\bar{\mathcal{L}} + \nabla_{\mathbf{b}}\mathcal{L} \cdot \mu^T \quad (145)$$

$$\nabla_{\mathbf{W}}\bar{\mathcal{L}} := -2\Sigma + 2\mathbf{W}(\sigma^2\mathbf{I} + \Sigma) \quad (146)$$

976 The nonlinear gradient learning dynamics reads

$$\dot{\mathbf{b}} = -\eta\nabla_{\mathbf{b}}\mathcal{L} \quad (147)$$

$$\dot{\mathbf{W}} = -\eta(\nabla_{\mathbf{W}}\bar{\mathcal{L}} + \nabla_{\mathbf{b}}\mathcal{L} \cdot \mu^T) \quad (148)$$

977

$$\dot{\mathbf{W}} - \dot{\mathbf{b}} \cdot \mu^T = -\eta\nabla_{\mathbf{W}}\bar{\mathcal{L}} \quad (149)$$

$$= 2\eta[\Sigma - \mathbf{W}(\sigma^2\mathbf{I} + \Sigma)] \quad (150)$$

$$\dot{\mathbf{b}} = -\eta\nabla_{\mathbf{b}}\mathcal{L} \quad (151)$$

$$= -2\eta(\mathbf{b} - \mu + \mathbf{W}\mu) \quad (152)$$

978 Consider the projection

$$(\dot{\mathbf{W}} - \dot{\mathbf{b}} \cdot \mu^T)\mathbf{u}_k = 2\eta[\Sigma - \mathbf{W}(\sigma^2\mathbf{I} + \Sigma)]\mathbf{u}_k \quad (153)$$

$$\dot{\mathbf{W}}\mathbf{u}_k - (\mu^T\mathbf{u}_k)\dot{\mathbf{b}} = 2\eta[\lambda_k\mathbf{u}_k - (\sigma^2 + \lambda_k)\mathbf{W}\mathbf{u}_k] \quad (154)$$

979

$$\dot{\mathbf{b}} = -2\eta(\mathbf{b} - \mu + \sum_k \mathbf{W}\mathbf{u}_k\mathbf{u}_k^T\mu)$$

980 Consider the variables $\mathbf{v}_k(\tau) = \mathbf{W}(\tau)\mathbf{u}_k$, let $\bar{\mathbf{b}}(\tau) = \mathbf{b}(\tau) - \mu$ so now the dynamic variables are
981 $\{\mathbf{v}_k, \dots, \bar{\mathbf{b}}\}$

$$\dot{\mathbf{v}}_k - (\mu^T\mathbf{u}_k)\dot{\bar{\mathbf{b}}} = 2\eta[\lambda_k\mathbf{u}_k - (\sigma^2 + \lambda_k)\mathbf{v}_k] \quad (155)$$

$$\dot{\bar{\mathbf{b}}} = -2\eta(\bar{\mathbf{b}} + \sum_k (\mu^T\mathbf{u}_k)\mathbf{v}_k) \quad (156)$$

982

$$\dot{\mathbf{v}}_k = 2\eta[\lambda_k\mathbf{u}_k - (\sigma^2 + \lambda_k)\mathbf{v}_k] - 2\eta(\mu^T\mathbf{u}_k)[\bar{\mathbf{b}} + \sum_l (\mu^T\mathbf{u}_l)\mathbf{v}_l] \quad (157)$$

$$= 2\eta[\lambda_k\mathbf{u}_k - (\sigma^2 + \lambda_k)\mathbf{v}_k - (\mu^T\mathbf{u}_k)\bar{\mathbf{b}} - \sum_l (\mu^T\mathbf{u}_k)(\mu^T\mathbf{u}_l)\mathbf{v}_l] \quad (158)$$

$$\dot{\bar{\mathbf{b}}} = -2\eta(\bar{\mathbf{b}} + \sum_k (\mu^T\mathbf{u}_k)\mathbf{v}_k) \quad (159)$$

983 The whole dynamics is linear and solvable, but now the dynamics in each component \mathbf{v}_k becomes
984 entangled with other components \mathbf{v}_l .

$$\frac{d}{d\tau} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \bar{\mathbf{b}} \end{bmatrix} = -2\eta M \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \bar{\mathbf{b}} \end{bmatrix} + 2\eta \begin{bmatrix} \lambda_1\mathbf{u}_1 \\ \lambda_2\mathbf{u}_2 \\ 0 \end{bmatrix} \quad (160)$$

985 with the blocks in M matrix defined as follows

$$M_{kk} = (\sigma^2 + \lambda_k + (\mu^T\mathbf{u}_k)^2)\mathbf{I}$$

$$M_{kl} = (\mu^T\mathbf{u}_k)(\mu^T\mathbf{u}_l)\mathbf{I}$$

$$M_{kb} = (\mu^T\mathbf{u}_k)\mathbf{I}$$

$$M_{bk} = (\mu^T\mathbf{u}_k)\mathbf{I}$$

$$M_{bb} = \mathbf{I}$$

986 We denote the overlap between the mean and principal components as $m_k = \mu^T \mathbf{u}_k$,
 987 Then we can represent the dynamic matrix M as a tensor product of a dense symmetric matrix Q
 988 with identity \mathbf{I}_N . More specifically, the dense matrix Q is a diagonal matrix plus the outer product of
 989 a vector. So it's real symmetric and diagonalizable.

$$M = \begin{bmatrix} \sigma^2 + \lambda_1 + m_1^2 & m_1 m_2 & \cdot & m_1 \\ m_1 m_2 & \sigma^2 + \lambda_2 + m_2^2 & \cdot & m_2 \\ \cdots & \cdots & \cdots & \cdot \\ m_1 & m_2 & \cdot & 1 \end{bmatrix} \otimes \mathbf{I}_N \quad (161)$$

$$= Q \otimes \mathbf{I}_N \quad (162)$$

$$Q := \begin{bmatrix} \sigma^2 + \lambda_1 + m_1^2 & m_1 m_2 & \cdot & m_1 \\ m_1 m_2 & \sigma^2 + \lambda_2 + m_2^2 & \cdot & m_2 \\ \cdots & \cdots & \cdots & \cdot \\ m_1 & m_2 & \cdot & 1 \end{bmatrix} \quad (163)$$

$$= D + qq^T \quad (164)$$

$$q := [m_1, m_2, \dots, 1]^T \quad (165)$$

$$D := \text{diag}(\sigma^2 + \lambda_1, \sigma^2 + \lambda_2, \sigma^2 + \lambda_3, \dots, \sigma^2 + \lambda_N, 0) \quad (166)$$

990 Note the inverse of Q is analytical, but the general eigendecomposition of it is not. Since the dynamic
 991 matrix M is real symmetric, the dynamics will still be separable along each eigenmode of M and
 992 converge w.r.t. it's own eigenvalue. The eigen decomposition of M can be obtained by numerical
 993 analysis and eigenvectors from Bunch–Nielsen–Sorensen formula [53, 55] . Generally speaking,
 994 since the mean of dataset μ usually lie in the directions of higher eigenvalues, the dynamics of \mathbf{b} and
 995 \mathbf{v}_k in the top eigenspace will be entangled with each other.

996 As a take home message, the overlap of μ and spectrum of Gaussian will induce some complex
 997 dynamics of bias and weight matrix along these modes. The full dynamics of \mathbf{W} , \mathbf{b} is still linear and
 998 solvable, but since the dynamic matrix is a tensor product of a Diagonal + low rank with identity, a
 999 closed form solution is generally harder, we can still obtain numerical solution of the dynamics easily.

1000 D.2.1 Special case: low dimensional interaction of mean and variance learning

1001 To gain intuition into how the mean and variance learning happens, consider the 1d distribution case,
 1002 which share the same math as the multi dimensional case where the mean overlaps with only one
 1003 eigenmode

$$\nabla_b \mathcal{L} = 2(b - \mu + \mathbf{W}\mu) \quad (167)$$

$$= 2b - 2(1 - w)\mu \quad (168)$$

$$= 2\mu w + 2b - 2\mu \quad (169)$$

$$\nabla_{\mathbf{W}} \mathcal{L} = -2\Sigma + 2\mathbf{W}(\sigma^2 \mathbf{I} + \Sigma) + 2(b - \mu + \mathbf{W}\mu)\mu^T \quad (170)$$

$$= -2\lambda - 2(1 - w)\mu^2 + 2w(\sigma^2 + \lambda) + 2\mu b \quad (171)$$

$$= 2(\mu^2 + \sigma^2 + \lambda)w + 2\mu b - 2(\lambda + \mu^2) \quad (172)$$

1004 Write down the dynamic equation as matrix equation

$$\frac{d}{d\tau} \begin{bmatrix} w \\ b \end{bmatrix} = -2\eta \left(\begin{bmatrix} \mu^2 + \sigma^2 + \lambda & \mu \\ \mu & 1 \end{bmatrix} \begin{bmatrix} w \\ b \end{bmatrix} - \begin{bmatrix} \lambda + \mu^2 \\ \mu \end{bmatrix} \right)$$

1005 Eigen equation reads

$$\det(A - \gamma \mathbf{I}) = (1 - \gamma)(\mu^2 + \sigma^2 + \lambda - \gamma) - \mu^2 \quad (173)$$

$$= \gamma^2 - (\lambda + \mu^2 + \sigma^2 + 1)\gamma + \lambda + \sigma^2 \quad (174)$$

1006 Generally for 2x2 matrices,

$$\begin{bmatrix} a & b \\ b & c \end{bmatrix}$$

1007 their eigenvalues are

$$1008 \quad \lambda_{1,2} = \frac{1}{2} \left(\pm \sqrt{(a-c)^2 + 4b^2} + a + c \right)$$

$$\mathbf{u}_{12} = \begin{bmatrix} \frac{\pm \sqrt{(a-c)^2 + 4b^2} + a - c}{2b} \\ 1 \end{bmatrix}$$

1009 In our case, $a = \mu^2 + \sigma^2 + \lambda$, $b = \mu$, $c = 1$.

$$\mathbf{u}_{12} = \begin{bmatrix} \frac{\pm \sqrt{(\mu^2 + \sigma^2 + \lambda - 1)^2 + 4\mu^2} + \mu^2 + \sigma^2 + \lambda - 1}{2\mu} \\ 1 \end{bmatrix}$$

1010 The faster learning dimension will be λ_1, \mathbf{u}_1 , where w and b will move in the same direction.

1011 **Key observation** is that the w and b 's dynamics depend on σ ,

- 1012 • for larger σ , w will converge faster, while b will slowly meandering, w moving with b ,
- 1013 following entrainment.

$$w^*(b) = \frac{\lambda + \mu^2 - \mu b}{\lambda + \mu^2 + \sigma^2}$$

- 1014 • for smaller σ , b will converge faster, comparable or entrained by w

$$b^*(w) = (1 - w)\mu$$

1015 D.3 Sampling ODE and Generated Distribution

1016 For simplicity consider the zero-mean case, where

$$\mathbf{W}(\tau; \sigma) = \mathbf{W}^* + \sum_k (\mathbf{W}(0; \sigma) \mathbf{u}_k - \frac{\lambda_k}{(\sigma^2 + \lambda_k)} \mathbf{u}_k) \mathbf{u}_k^T e^{-2\eta(\sigma^2 + \lambda_k)\tau} \quad (175)$$

$$= \sum_k \frac{\lambda_k}{(\sigma^2 + \lambda_k)} \mathbf{u}_k \mathbf{u}_k^T (1 - e^{-2\eta(\sigma^2 + \lambda_k)\tau}) + \mathbf{W}(0; \sigma) \sum_k \mathbf{u}_k \mathbf{u}_k^T e^{-2\eta(\sigma^2 + \lambda_k)\tau} \quad (176)$$

$$\mathbf{b}(\tau; \sigma) = \mathbf{b}(0) \exp(-2\eta\tau) \quad (177)$$

1017 To let it decompose mode by mode in the sampling ODE, we assume aligned initialization
 1018 $\mathbf{u}_k^T \mathbf{W}(0; \sigma) \mathbf{u}_m = 0$ when $n \neq m$.

1019 Then

$$\frac{d}{d\sigma} \mathbf{u}_k^T \mathbf{x} = -\frac{1}{\sigma} \left(\left(-\frac{\sigma^2}{(\sigma^2 + \lambda_k)} + (\mathbf{u}_k^T \mathbf{W}(0; \sigma) \mathbf{u}_k - \frac{\lambda_k}{(\sigma^2 + \lambda_k)}) e^{-2\eta(\sigma^2 + \lambda_k)\tau} \right) \mathbf{u}_k^T \mathbf{x} + \mathbf{u}_k^T \mathbf{b}(0; \sigma) e^{-2\eta\tau} \right)$$

1020 Let the initialization along \mathbf{u}_k be $\mathbf{u}_k^T \mathbf{W}(0; \sigma) \mathbf{u}_k = q_k$, then

$$\frac{d}{d\sigma} \mathbf{u}_k^T \mathbf{x} = -\frac{1}{\sigma} \left(\left(-\frac{\sigma^2}{(\sigma^2 + \lambda_k)} + (q_k - \frac{\lambda_k}{(\sigma^2 + \lambda_k)}) e^{-2\eta(\sigma^2 + \lambda_k)\tau} \right) \mathbf{u}_k^T \mathbf{x} + \mathbf{u}_k^T \mathbf{b}(0; \sigma) e^{-2\eta\tau} \right)$$

1021 Using some integration results

$$\int d\sigma \frac{1}{\sigma(\sigma^2 + \lambda_k)} \sigma^2 = \frac{1}{2} \log(\lambda_k + \sigma^2) + C$$

$$\int d\sigma \frac{1}{\sigma} e^{-2\eta(\sigma^2 + \lambda_k)\tau} = -\frac{1}{2} e^{-2\eta\lambda_k\tau} \text{Ei}(-2\eta\tau\sigma^2) + C$$

$$\int d\sigma \frac{\lambda_k}{\sigma(\sigma^2 + \lambda_k)} e^{-2\eta(\sigma^2 + \lambda_k)\tau} = \frac{1}{2} (\text{Ei}(-2\eta\tau\sigma^2) e^{-2\eta\tau\lambda_k} - \text{Ei}(-2\eta\tau(\sigma^2 + \lambda_k))) + C$$

1022 Integrating this ODE, we get

$$c_k(\sigma) = C \exp \left(\frac{1}{2} \log (\lambda_k + \sigma^2) + \frac{1}{2} (\text{Ei} (-2\eta\tau\sigma^2) e^{-2\eta\tau\lambda_k} - \text{Ei} (-2\eta\tau(\sigma^2 + \lambda_k))) \right) + \quad (178)$$

$$- q_k \frac{1}{2} \text{Ei} (-2\eta\tau\sigma^2) e^{-2\eta\lambda_k\tau} \quad (179)$$

$$= C \sqrt{\lambda_k + \sigma^2} \exp \left(\frac{1}{2} ((1 - q_k) \text{Ei} (-2\eta\tau\sigma^2) e^{-2\eta\tau\lambda_k} - \text{Ei} (-2\eta\tau(\sigma^2 + \lambda_k))) \right) \quad (180)$$

1023 Solution of sampling dynamics ODE

$$\mathbf{x}(\sigma_0) = \sum_k \mathbf{u}_k \frac{c_k(\sigma_0)}{c_k(\sigma_T)} (\mathbf{u}_k^T \mathbf{x}(\sigma_T)) \quad (181)$$

$$= \sum_k \frac{c_k(\sigma_0)}{c_k(\sigma_T)} \mathbf{u}_k \mathbf{u}_k^T \mathbf{x}(\sigma_T) \quad (182)$$

1024 The variance of generated distribution reads

$$\tilde{\Sigma}^\tau = \sigma_T^2 \sum_k \left(\frac{c_k(\sigma_0)}{c_k(\sigma_T)} \right)^2 \mathbf{u}_k \mathbf{u}_k^T \quad (183)$$

1025 where the variance along eigenvector \mathbf{u}_k reads

$$\tilde{\lambda}_k^\tau = \sigma_T^2 \left(\frac{c_k(\sigma_0)}{c_k(\sigma_T)} \right)^2 \quad (184)$$

$$= \sigma_T^2 \frac{\lambda_k + \sigma_0^2 \exp \left((1 - q_k) \text{Ei} (-2\eta\tau\sigma_0^2) e^{-2\eta\tau\lambda_k} - \text{Ei} (-2\eta\tau(\sigma_0^2 + \lambda_k)) \right)}{\lambda_k + \sigma_T^2 \exp \left((1 - q_k) \text{Ei} (-2\eta\tau\sigma_T^2) e^{-2\eta\tau\lambda_k} - \text{Ei} (-2\eta\tau(\sigma_T^2 + \lambda_k)) \right)} \quad (185)$$

1026 E Detailed Derivations for Two-Layer Symmetric Parameterization

1027 Here we outline the main derivation steps for the two-layer symmetric case:

$$\mathbf{D}(\mathbf{x}) = P P^T \mathbf{x} + \mathbf{b}. \quad (186)$$

1028 E.1 Symmetric parametrization zero mean gradient dynamics

1029 Note that if the weight matrix \mathbf{W} has internal structure, i.e. parametrized by θ , we can easily derive
1030 the gradient flow of those parameters using chain rule.

1031 Let $\mathbf{W} = \mathbf{W}(\theta)$,

$$\nabla_{\theta} \mathcal{L} = \sum_{ij} \left(\frac{\partial \mathcal{L}}{\partial \mathbf{W}} \right)_{ij} \frac{\partial \mathbf{W}_{ij}}{\partial \theta}$$

1032 Here, when $\mathbf{W} = P P^T$, via chain rule, we can derive its gradient, which depends on the symmetrized
1033 gradient of \mathbf{W}

$$\nabla_P \mathcal{L} = (\nabla_{\mathbf{W}} \mathcal{L}) P + (\nabla_{\mathbf{W}} \mathcal{L})^T P \quad (187)$$

$$= \left[\nabla_{\mathbf{W}} \mathcal{L} + (\nabla_{\mathbf{W}} \mathcal{L})^T \right] P \quad (188)$$

1034 where

$$\nabla_{\mathbf{b}} \mathcal{L} = 2(\mathbf{b} - (\mathbf{I} - \mathbf{W})\mu) \quad (189)$$

$$\nabla_{\mathbf{W}} \mathcal{L} = -2\Sigma + 2\mathbf{W}(\sigma^2 \mathbf{I} + \Sigma) + [2\mathbf{W}\mu\mu^T + 2(\mathbf{b} - \mu)\mu^T] \quad (190)$$

1035 Expand the full gradient (non-zero mean case), then we have

$$\nabla_P \mathcal{L} = \left[-4\Sigma + 2\mathbf{W}(\sigma^2 \mathbf{I} + \Sigma) + [2\mathbf{W}\mu\mu^T + 2(\mathbf{b} - \mu)\mu^T] \right] P \quad (191)$$

$$+ 2(\sigma^2 \mathbf{I} + \Sigma) \mathbf{W}^T + [2\mu\mu^T \mathbf{W}^T + 2\mu(\mathbf{b} - \mu)^T] \right] P \quad (192)$$

$$= -4\Sigma P + 2P P^T (\sigma^2 \mathbf{I} + \Sigma) P + 2(\sigma^2 \mathbf{I} + \Sigma) P P^T P \quad (193)$$

$$+ [2P P^T \mu\mu^T + 2(\mathbf{b} - \mu)\mu^T] P + [2\mu\mu^T P P^T + 2\mu(\mathbf{b} - \mu)^T] P \quad (194)$$

1036 In the zero-mean case this simplifies to

$$\nabla_P \mathcal{L}_{\mu=0} = -4\Sigma P + 2P P^T (\sigma^2 \mathbf{I} + \Sigma) P + 2(\sigma^2 \mathbf{I} + \Sigma) P P^T P \quad (195)$$

1037 Consider representing the gradient on eigenbases, let $\mathbf{u}_k^T P = q_k^T$.

$$\mathbf{u}_k^T \nabla_P \mathcal{L}_{\mu=0} = -4\mathbf{u}_k^T \Sigma P + 2\mathbf{u}_k^T P P^T (\sigma^2 \mathbf{I} + \Sigma) P + 2\mathbf{u}_k^T (\sigma^2 \mathbf{I} + \Sigma) P P^T P \quad (196)$$

$$= -4\lambda_k \mathbf{u}_k^T P + 2\mathbf{u}_k^T P \sum_m (\sigma^2 + \lambda_m) P^T \mathbf{u}_m \mathbf{u}_m^T P + 2(\sigma^2 + \lambda_k) \mathbf{u}_k^T P P^T P \quad (197)$$

$$= -4\lambda_k q_k^T + 2q_k^T \sum_m (\sigma^2 + \lambda_m) q_m q_m^T + 2(\sigma^2 + \lambda_k) q_k^T \sum_m q_m q_m^T \quad (198)$$

$$= -4\lambda_k q_k^T + 2 \sum_m (2\sigma^2 + \lambda_m + \lambda_k) (q_k^T q_m) q_m^T \quad (199)$$

$$\nabla_{q_k^T} \mathcal{L}_{\mu=0} = -4\lambda_k q_k^T + 2 \sum_m (2\sigma^2 + \lambda_m + \lambda_k) (q_k^T q_m) q_m^T \quad (200)$$

$$\nabla_{q_k} \mathcal{L}_{\mu=0} = -4\lambda_k q_k + 2 \sum_m (2\sigma^2 + \lambda_m + \lambda_k) (q_k^T q_m) q_m \quad (201)$$

1038 **Fixed points analysis** A stationary solution at which the gradient vanishes is

$$(q_k^T q_m) = 0 \text{ if } k \neq m \quad (202)$$

$$(q_k^T q_k) = \frac{\lambda_k}{\lambda_k + \sigma^2} \text{ or } 0 \quad (203)$$

1039 Note, this is different from the one-layer case where there is no saddle points, here we got a bunch of
1040 zero solution as saddle points.

1041 **Dynamics of the overlap** Note the dynamics of the overlap

$$\frac{d}{d\tau}(q_k^T q_m) = q_k^T \frac{d}{d\tau} q_m + q_m^T \frac{d}{d\tau} q_k \quad (204)$$

$$= -\eta[q_k^T \nabla_{q_m} \mathcal{L}_{\mu=0} + q_m^T \nabla_{q_k} \mathcal{L}_{\mu=0}] \quad (205)$$

$$= -\eta[-4\lambda_k(q_m^T q_k) + 2 \sum_n (2\sigma^2 + \lambda_n + \lambda_k)(q_k^T q_n) q_m^T q_n] \quad (206)$$

$$- 4\lambda_m(q_k^T q_m) + 2 \sum_n (2\sigma^2 + \lambda_n + \lambda_m)(q_m^T q_n) q_k^T q_n] \quad (207)$$

$$= -\eta[-4(\lambda_k + \lambda_m)(q_m^T q_k) + 2 \sum_n (4\sigma^2 + 2\lambda_n + \lambda_m + \lambda_k)(q_k^T q_n)(q_m^T q_n)] \quad (208)$$

$$= 4\eta[(\lambda_k + \lambda_m)(q_m^T q_k) - \sum_n (2\sigma^2 + \lambda_n + \frac{\lambda_m + \lambda_k}{2})(q_k^T q_n)(q_m^T q_n)] \quad (209)$$

1042 This shows that when all the overlaps are initialized as zero, they will stay at zero, i.e. when weights
1043 are initialized to be aligned to the eigenbasis, it will stay aligned.

1044 **E.1.1 Simplifying assumption: orthogonal initialization** $q_k^T q_m = 0$

1045 Consider the simple case where each $q_k^T q_m = 0$, $k \neq m$ at network initialization. i.e. each q are
1046 orthogonal to each other. Then, it's easy to show that $\frac{d}{d\tau}(q_k^T q_m) = 0$ at the start and throughout
1047 training. Thus, we know orthogonally initialized modes will evolve independently.

1048 Note this assumption can also be written as

$$q_k^T q_m = \mathbf{u}_k^T P P^T \mathbf{u}_m = \mathbf{u}_k^T \mathbf{W} \mathbf{u}_m = 0 \quad \forall k \neq m$$

1049 which means, the eigenvectors of Σ are still orthogonal w.r.t. matrix \mathbf{W} , i.e. the matrix \mathbf{W} shares
1050 eigenbases with the data covariance Σ .

1051 In such case,

$$\nabla_{q_k} \mathcal{L}_{\mu=0} = -4\lambda_k q_k + 2 \sum_m (2\sigma^2 + \lambda_m + \lambda_k)(q_k^T q_m) q_m \quad (210)$$

$$(ortho) = -4\lambda_k q_k + 2(2\sigma^2 + 2\lambda_k)(q_k^T q_k) q_k \quad (211)$$

$$= -4\lambda_k q_k + 4(\sigma^2 + \lambda_k)(q_k^T q_k) q_k \quad (212)$$

1052 The dynamics read

$$\frac{dq_k}{d\tau} = -\eta \nabla_{q_k} \mathcal{L}_{\mu=0} \quad (213)$$

$$= -\eta(-4\lambda_k q_k + 4(\sigma^2 + \lambda_k)(q_k^T q_k) q_k) \quad (214)$$

$$= 4\eta(\lambda_k - (\sigma^2 + \lambda_k)(q_k^T q_k)) q_k \quad (215)$$

1053 since the right hand side is aligned with q_k so it can only move by scaling the initial value.

1054 The fixed-point solution is $q_k = 0$ or when $q_k^T q_k = \frac{\lambda_k}{\sigma^2 + \lambda_k}$. Given the arbitrariness of q_k itself, we
1055 track the dynamics of its squared norm. The learning dynamics of $q_k^T q_k$ reads

$$q_k^T \frac{dq_k}{d\tau} = 4\eta(\lambda_k - (\sigma^2 + \lambda_k)(q_k^T q_k))(q_k^T q_k) \quad (216)$$

$$\frac{1}{2} \frac{d(q_k^T q_k)}{d\tau} = 4\eta(\lambda_k - (\sigma^2 + \lambda_k)(q_k^T q_k))(q_k^T q_k) \quad (217)$$

$$\frac{d(f)}{d\tau} = 8\eta(\lambda_k - (\sigma^2 + \lambda_k)f)f \quad (218)$$

1056 Fortunately, this ODE has close-form solution.

$$(q_k^T q_k)(\tau) = \|q_k(\tau)\|^2 = \frac{a}{1/K e^{-at} + b} \quad (219)$$

$$= \frac{8\eta\lambda_k}{1/K e^{-8\eta\lambda_k\tau} + 8\eta(\sigma^2 + \lambda_k)} \quad (220)$$

$$= \frac{8\eta\lambda_k}{(\frac{8\eta\lambda_k}{\|q_k(0)\|^2} - 8\eta(\sigma^2 + \lambda_k))e^{-8\eta\lambda_k\tau} + 8\eta(\sigma^2 + \lambda_k)} \quad (221)$$

$$= \frac{\lambda_k}{(\frac{\lambda_k}{\|q_k(0)\|^2} - (\sigma^2 + \lambda_k))e^{-8\eta\lambda_k\tau} + (\sigma^2 + \lambda_k)} \quad (222)$$

$$= \frac{\lambda_k}{\sigma^2 + \lambda_k} \left(\frac{1}{(\frac{1}{\|q_k(0)\|^2} \frac{\lambda_k}{\sigma^2 + \lambda_k} - 1)e^{-8\eta\lambda_k\tau} + 1} \right) \quad (223)$$

$$(q_k^T q_m)(\tau) = 0 \quad \text{if } k \neq m \quad (224)$$

1057 This gives rise to the full vector solution

$$q_k(\tau) = \sqrt{\|q_k(\tau)\|^2} \frac{q_k(0)}{\|q_k(0)\|} \quad (225)$$

$$= \sqrt{\frac{\lambda_k}{\sigma^2 + \lambda_k}} \left(\frac{1}{\sqrt{(\frac{1}{\|q_k(0)\|^2} \frac{\lambda_k}{\sigma^2 + \lambda_k} - 1)e^{-8\eta\lambda_k\tau} + 1}} \right) \frac{q_k(0)}{\|q_k(0)\|} \quad (226)$$

$$= \sqrt{\frac{\lambda_k}{\sigma^2 + \lambda_k}} \left(\frac{1}{\sqrt{(\frac{\lambda_k}{\sigma^2 + \lambda_k} - \|q_k(0)\|^2)e^{-8\eta\lambda_k\tau} + \|q_k(0)\|^2}} \right) q_k(0) \quad (227)$$

$$= \sqrt{\frac{\lambda_k}{\sigma^2 + \lambda_k}} \left(\frac{1}{\sqrt{\frac{\lambda_k}{\sigma^2 + \lambda_k} e^{-8\eta\lambda_k\tau} + (1 - e^{-8\eta\lambda_k\tau})\|q_k(0)\|^2}} \right) q_k(0) \quad (228)$$

1058 **Learning Dynamics of score estimator** Now, consider the whole estimator, $\mathbf{W} = PP^T$. Recall

1059 $\mathbf{u}_k^T P = q_k^T$

$$P = \sum_k \mathbf{u}_k \mathbf{u}_k^T P \quad (229)$$

$$= \sum_k \mathbf{u}_k q_k^T \quad (230)$$

1060

$$PP^T = \left(\sum_k \mathbf{u}_k q_k^T \right) \left(\sum_m q_m \mathbf{u}_m^T \right) \quad (231)$$

$$= \sum_k \sum_m \mathbf{u}_k (q_k^T q_m) \mathbf{u}_m^T \quad (232)$$

1061 Note that under our assumption, the $(q_k^T q_m)(\tau) = 0$ if $k \neq m$, $q_k^T q_m$ is diagonal. So

$$PP^T = \sum_k \mathbf{u}_k (q_k^T q_k) \mathbf{u}_k^T \quad (233)$$

$$= \sum_k \|q_k(\tau)\|^2 \mathbf{u}_k \mathbf{u}_k^T \quad (234)$$

1062 Then we can rewrite

$$\|q_k(\tau)\|^2 = (q_k^T q_k)(\tau) = \frac{\lambda_k}{\sigma^2 + \lambda_k} \left(\frac{1}{(\frac{1}{\|q_k(0)\|^2} \frac{\lambda_k}{\sigma^2 + \lambda_k} - 1)e^{-8\eta\lambda_k\tau} + 1} \right) \quad (235)$$

1063

$$\mathbf{W}(\tau) = PP^T(\tau) = \sum_k \|q_k(\tau)\|^2 \mathbf{u}_k \mathbf{u}_k^T \quad (236)$$

$$= \sum_k \frac{\lambda_k}{\sigma^2 + \lambda_k} \mathbf{u}_k \mathbf{u}_k^T \left(\frac{1}{(\frac{1}{\|q_k(0)\|^2} \frac{\lambda_k}{\sigma^2 + \lambda_k} - 1)e^{-8\eta\lambda_k\tau} + 1} \right) \quad (237)$$

Score estimation error dynamics

$$E_s = \mathbb{E}_{\mathbf{x}} \|\mathbf{s}(\mathbf{x}) - \mathbf{s}^*(\mathbf{x})\|^2 = \frac{1}{\sigma^4} \left[\|\mathbf{b} - \mathbf{b}^*\|^2 + 2(\mathbf{b} - \mathbf{b}^*)^T (\mathbf{W} - \mathbf{W}^*) \mu + \text{Tr}[(\mathbf{W} - \mathbf{W}^*)^T (\mathbf{W} - \mathbf{W}^*) (\mu \mu^T + \Sigma + \sigma^2 \mathbf{I})] \right] \quad (238)$$

$$\begin{aligned} E_s^{(\mu=0)} &= \frac{1}{\sigma^4} \left[\|\mathbf{b} - \mathbf{b}^*\|^2 + \text{Tr}[(\mathbf{W} - \mathbf{W}^*)^T (\mathbf{W} - \mathbf{W}^*) (\Sigma + \sigma^2 \mathbf{I})] \right] \quad (239) \\ &= \frac{1}{\sigma^4} \left[\|\mathbf{b} - \mathbf{b}^*\|^2 + \text{Tr}[(\mathbf{W} - \mathbf{W}^*)^T (\mathbf{W} - \mathbf{W}^*) \sum_k (\lambda_k + \sigma^2) \mathbf{u}_k \mathbf{u}_k^T] \right] \quad (240) \end{aligned}$$

1064 Thus

$$E_s^{(\mu=0, \text{cov term})} = \text{Tr}[(\mathbf{W} - \mathbf{W}^*)^T (\mathbf{W} - \mathbf{W}^*) \sum_k (\lambda_k + \sigma^2) \mathbf{u}_k \mathbf{u}_k^T] \quad (241)$$

$$= \text{Tr} \left[\sum_k (\lambda_k + \sigma^2) (\mathbf{W} - \mathbf{W}^*) \mathbf{u}_k \mathbf{u}_k^T (\mathbf{W} - \mathbf{W}^*)^T \right] \quad (242)$$

$$= \text{Tr} \left[\sum_k (\lambda_k + \sigma^2) \left(P q_k - \frac{\lambda_k}{\lambda_k + \sigma^2} \mathbf{u}_k \right) \left(P q_k - \frac{\lambda_k}{\lambda_k + \sigma^2} \mathbf{u}_k \right)^T \right] \quad (243)$$

$$= \text{Tr} \left[\sum_k (\lambda_k + \sigma^2) \left(P q_k - \frac{\lambda_k}{\lambda_k + \sigma^2} \mathbf{u}_k \right)^T \left(P q_k - \frac{\lambda_k}{\lambda_k + \sigma^2} \mathbf{u}_k \right) \right] \quad (244)$$

$$= \sum_k (\lambda_k + \sigma^2) \left(q_k^T P^T P q_k - 2 \frac{\lambda_k}{\lambda_k + \sigma^2} \mathbf{u}_k^T P q_k + \left(\frac{\lambda_k}{\lambda_k + \sigma^2} \right)^2 \mathbf{u}_k^T \mathbf{u}_k \right) \quad (245)$$

$$= \sum_k (\lambda_k + \sigma^2) \left((q_k^T q_k)^2 - 2 \frac{\lambda_k}{\lambda_k + \sigma^2} q_k^T q_k + \left(\frac{\lambda_k}{\lambda_k + \sigma^2} \right)^2 \right) \quad (246)$$

$$= \sum_k (\lambda_k + \sigma^2) \left((q_k^T q_k) - \frac{\lambda_k}{\lambda_k + \sigma^2} \right)^2 \quad (247)$$

$$E_s^{(\mu=0, \text{bias term})} = (b^0)^2 \exp(-4\eta\tau)$$

1065 The dynamics can be applied

$$(q_k^T q_k)(\tau) = \frac{\lambda_k}{\sigma^2 + \lambda_k} \left(\frac{1}{\left(\frac{1}{\|q_k(0)\|^2} \frac{\lambda_k}{\sigma^2 + \lambda_k} - 1 \right) e^{-8\eta\lambda_k\tau} + 1} \right)$$

1066

$$E_s^{(\mu=0, \text{cov term})} = \sum_k (\lambda_k + \sigma^2) \left(\frac{\lambda_k}{\sigma^2 + \lambda_k} \left(\frac{1}{\left(\frac{1}{\|q_k(0)\|^2} \frac{\lambda_k}{\sigma^2 + \lambda_k} - 1 \right) e^{-8\eta\lambda_k\tau} + 1} \right) - \frac{\lambda_k}{\lambda_k + \sigma^2} \right)^2 \quad (248)$$

$$= \sum_k (\lambda_k + \sigma^2) \left(\frac{\lambda_k}{\sigma^2 + \lambda_k} \right)^2 \left(\left(\frac{1}{\left(\frac{1}{\|q_k(0)\|^2} \frac{\lambda_k}{\sigma^2 + \lambda_k} - 1 \right) e^{-8\eta\lambda_k\tau} + 1} \right) - 1 \right)^2 \quad (249)$$

$$= \sum_k \frac{\lambda_k^2}{\lambda_k + \sigma^2} \left(\frac{\left(\frac{1}{\|q_k(0)\|^2} \frac{\lambda_k}{\sigma^2 + \lambda_k} - 1 \right) e^{-8\eta\lambda_k\tau}}{\left(\frac{1}{\|q_k(0)\|^2} \frac{\lambda_k}{\sigma^2 + \lambda_k} - 1 \right) e^{-8\eta\lambda_k\tau} + 1} \right)^2 \quad (250)$$

$$= \sum_k \frac{\lambda_k^2}{\lambda_k + \sigma^2} \left(\frac{\left(\frac{\lambda_k}{\sigma^2 + \lambda_k} - \|q_k(0)\|^2 \right) e^{-8\eta\lambda_k\tau}}{\left(\frac{\lambda_k}{\sigma^2 + \lambda_k} - \|q_k(0)\|^2 \right) e^{-8\eta\lambda_k\tau} + \|q_k(0)\|^2} \right)^2 \quad (251)$$

1067 The full score estimation loss is

$$E_s^{(\mu=0)} = (b^0)^2 \exp(-4\eta\tau) + \sum_k \frac{\lambda_k^2}{\lambda_k + \sigma^2} \left(\frac{(\frac{\lambda_k}{\sigma^2 + \lambda_k} - \|q_k(0)\|^2) e^{-8\eta\lambda_k\tau}}{(\frac{\lambda_k}{\sigma^2 + \lambda_k} - \|q_k(0)\|^2) e^{-8\eta\lambda_k\tau} + \|q_k(0)\|^2} \right)^2 \quad (252)$$

1068 **E.1.2 Beyond Aligned Initialization : qualitative analysis of off diagonal dynamics**

1069 Next we can write down the dynamics of the non-diagonal part of the weight, i.e. overlaps between
1070 q_k and q_m

$$\frac{d}{d\tau}(q_k^T q_m) \quad (253)$$

$$= q_k^T \frac{d}{d\tau} q_m + q_m^T \frac{d}{d\tau} q_k \quad (254)$$

$$= -\eta[-4(\lambda_k + \lambda_m)(q_m^T q_k) + 2 \sum_n (4\sigma^2 + 2\lambda_n + \lambda_m + \lambda_k)(q_k^T q_n)(q_m^T q_n)] \quad (255)$$

$$= 4\eta[(\lambda_k + \lambda_m)(q_m^T q_k) - \sum_n (2\sigma^2 + \lambda_n + \frac{\lambda_m + \lambda_k}{2})(q_k^T q_n)(q_m^T q_n)] \quad (256)$$

$$= 4\eta \left[(\lambda_k + \lambda_m)(q_m^T q_k) - (2\sigma^2 + \lambda_k + \frac{\lambda_m + \lambda_k}{2})(q_k^T q_k)(q_m^T q_k) \right. \quad (257)$$

$$\left. - (2\sigma^2 + \lambda_m + \frac{\lambda_m + \lambda_k}{2})(q_k^T q_m)(q_m^T q_m) - \sum_{n \neq k, m} (2\sigma^2 + \lambda_n + \frac{\lambda_m + \lambda_k}{2})(q_k^T q_n)(q_m^T q_n) \right]$$

$$= 4\eta \left[\left(\lambda_k + \lambda_m - (2\sigma^2 + \frac{\lambda_m + 3\lambda_k}{2})\|q_k\|^2 - (2\sigma^2 + \frac{3\lambda_m + \lambda_k}{2})\|q_m\|^2 \right) (q_m^T q_k) \right. \quad (258)$$

$$\left. - \sum_{n \neq k, m} (2\sigma^2 + \lambda_n + \frac{\lambda_m + \lambda_k}{2})(q_k^T q_n)(q_m^T q_n) \right]$$

$$= 4\eta \left[\left(\lambda_k(1 - \|q_k\|^2) + \lambda_m(1 - \|q_m\|^2) - (2\sigma^2 + \frac{\lambda_m + \lambda_k}{2})(\|q_k\|^2 + \|q_m\|^2) \right) (q_m^T q_k) \right. \quad (259)$$

$$\left. - \sum_{n \neq k, m} (2\sigma^2 + \lambda_n + \frac{\lambda_m + \lambda_k}{2})(q_k^T q_n)(q_m^T q_n) \right]$$

1071 As a reference, recall the dynamics of diagonal term,

$$\frac{d(q_k^T q_k)}{d\tau} = 8\eta(\lambda_k - (\sigma^2 + \lambda_k)(q_k^T q_k))(q_k^T q_k)$$

1072 Assume the diagonal term is not stuck at zero $q_k^T q_k \neq 0$, then we know it will asymptotically go to
1073 $\|q_k(\infty)\|^2 \approx \frac{\lambda_k}{\sigma^2 + \lambda_k}$ following sigmoidal dynamics.

1074 For overlap between $q_k^T q_m$, without loss of generality, assume $\lambda_k > \lambda_m$. Then we have three dynamic
1075 phases, 1) neither mode has emerged; 2) mode k has emerged, while m has not; 3) both modes have
1076 emerged.

1077 **Phase 1: neither mode has emerged** When neither mode has emerged, assume $\|q_k(\tau)\|^2 \approx 0$,
1078 then the overlap will grow exponentially.

$$\frac{d}{d\tau}(q_k^T q_m) \approx 4\eta \left(\lambda_k + \lambda_m \right) (q_m^T q_k)$$

1079 Note given $\lambda_k > \lambda_m$, the diagonal term has a higher increasing speed than the non-diagonal overlap
1080 term $8\eta(\lambda_k - (\sigma^2 + \lambda_k)(q_k^T q_k)) > 4\eta(\lambda_k + \lambda_m)$.

1081 **Phase 2: one mode has emerged, the other has not** When one mode has converged $\|q_k(\tau)\|^2 \approx$
 1082 $\frac{\lambda_k}{\sigma^2 + \lambda_k}$, but the other has not ($\lambda_k > \lambda_m$)

$$\frac{d}{d\tau}(q_k^T q_m) \approx 4\eta \left(\lambda_k + \lambda_m - (2\sigma^2 + \frac{\lambda_m + 3\lambda_k}{2}) \frac{\lambda_k}{\sigma^2 + \lambda_k} \right) (q_m^T q_k) \quad (260)$$

$$= 4\eta \left(\lambda_k + \lambda_m - (2\sigma^2 + 2\lambda_k + \frac{\lambda_m - \lambda_k}{2}) \frac{\lambda_k}{\sigma^2 + \lambda_k} \right) (q_m^T q_k) \quad (261)$$

$$= 4\eta \left(\lambda_k + \lambda_m - 2\lambda_k - (\frac{\lambda_m - \lambda_k}{2}) \frac{\lambda_k}{\sigma^2 + \lambda_k} \right) (q_m^T q_k) \quad (262)$$

$$= 4\eta \left(\lambda_m - \lambda_k - (\frac{\lambda_m - \lambda_k}{2}) \frac{\lambda_k}{\sigma^2 + \lambda_k} \right) (q_m^T q_k) \quad (263)$$

$$= 4\eta(\lambda_m - \lambda_k) \left(1 - \frac{1}{2} \frac{\lambda_k}{\sigma^2 + \lambda_k} \right) (q_m^T q_k) \quad (264)$$

1083 Since $\frac{\lambda_k}{\sigma^2 + \lambda_k} < 1$, we know $1 - \frac{1}{2} \frac{\lambda_k}{\sigma^2 + \lambda_k} > 0$. Thus, the dynamic coefficient $(\lambda_m - \lambda_k)(1 -$
 1084 $\frac{1}{2} \frac{\lambda_k}{\sigma^2 + \lambda_k}) < 0$, so the overlap will follow an exponential decay dynamics. Further, given the same λ_k ,
 1085 the decay speed of $q_k^T q_m$ is proportional to the difference of the eigenvalues $\lambda_m - \lambda_k$. So, the larger
 1086 the difference, the faster the decay.

1087 **Phase 3: both modes have emerged** When both modes have emerged $\|q_k(\tau)\|^2 \approx \frac{\lambda_k}{\sigma^2 + \lambda_k}$,
 1088 $\|q_m(\tau)\|^2 \approx \frac{\lambda_m}{\sigma^2 + \lambda_m}$,

$$\frac{d}{d\tau}(q_k^T q_m) \approx 4\eta \left(\lambda_k + \lambda_m - (2\sigma^2 + \frac{\lambda_m + 3\lambda_k}{2}) \frac{\lambda_k}{\sigma^2 + \lambda_k} - (2\sigma^2 + \frac{3\lambda_m + \lambda_k}{2}) \frac{\lambda_m}{\sigma^2 + \lambda_m} \right) (q_m^T q_k) \quad (265)$$

$$= 4\eta \left(-\lambda_k - \lambda_m + (\frac{\sigma^2}{2}) (\frac{(\lambda_m - \lambda_k)^2}{(\sigma^2 + \lambda_m)(\sigma^2 + \lambda_k)}) \right) (q_m^T q_k) \quad (266)$$

$$= -4\eta \frac{(\lambda_k + \lambda_m + 2\sigma^2) (2\lambda_m \lambda_k + \sigma^2 (\lambda_k + \lambda_m))}{2(\lambda_k + \sigma^2)(\lambda_m + \sigma^2)} (q_m^T q_k) \quad (267)$$

1089 The dynamic coefficient is negative, so the overlap decays even more rapidly.

1090 **Summary: qualitative dynamics of off diagonal elements** $q_k^T q_m$

- 1091 • The off-diagonal term will exponentially rising after the rise of the larger variance dimension,
- 1092 and before the rise of smaller variance dimension;
- 1093 • after one has rise it will decay to zero;
- 1094 • after both has rise it will decay faster.

1095 Thus, it will follow non monotonic rise and fall dynamics.

1096 E.2 Sampling ODE and Generated Distribution

1097 Here, for tractability purposes, we will mainly consider the zero-mean and aligned initialization case.

1098 Using the solutions from above,

$$\mathbf{W}(\tau; \sigma) = PP^T(\tau) = \sum_k \|q_k(\tau)\|^2 \mathbf{u}_k \mathbf{u}_k^T \quad (268)$$

$$= \sum_k \frac{\lambda_k}{\sigma^2 + \lambda_k} \mathbf{u}_k \mathbf{u}_k^T \left(\left(\frac{1}{\|q_k(0; \sigma)\|^2} \frac{\lambda_k}{\sigma^2 + \lambda_k} - 1 \right) e^{-8\eta\tau\lambda_k} + 1 \right)^{-1} \quad (269)$$

1099 with the weight initialization

$$\mathbf{W}(0; \sigma) = \sum_k \|q_k(0; \sigma)\|^2 \mathbf{u}_k \mathbf{u}_k^T$$

1100 Note here we assume the initialization $\|q_k(0; \sigma)\|^2$ is the same for all σ level, with no σ dependency.
 1101 $\|q_k(0; \sigma)\|^2 = \|q_k(0)\|^2 = Q_k$

1102 Per our previous convention the key factors are

$$\psi_k(\sigma; \tau) = \frac{\lambda_k}{\sigma^2 + \lambda_k} \frac{1}{\left(\frac{1}{Q_k} \frac{\lambda_k}{\sigma^2 + \lambda_k} - 1\right) e^{-8\eta\tau\lambda_k} + 1} \quad (270)$$

1103 and the integration reads

$$\Phi_k(\sigma) = \exp\left(-\int^\sigma \frac{\psi_k(\sigma'; \tau) - 1}{\sigma'} d\sigma'\right)$$

$$\frac{d}{d\sigma} \mathbf{u}_k^T \mathbf{x} = -\frac{1}{\sigma} \left(\psi_k(\sigma; \tau) - 1 \right) \mathbf{u}_k^T \mathbf{x} \quad (271)$$

$$= -\frac{1}{\sigma} \left(\frac{\lambda_k}{\sigma^2 + \lambda_k} \frac{1}{\left(\frac{1}{Q_k} \frac{\lambda_k}{\sigma^2 + \lambda_k} - 1\right) e^{-8\eta\tau\lambda_k} + 1} - 1 \right) \mathbf{u}_k^T \mathbf{x} \quad (272)$$

1104 Note that the integrand is just a fraction function of σ^2 which can be integrated analytically.

$$\int d\sigma - \frac{1}{\sigma} \left(\frac{\lambda_k}{\sigma^2 + \lambda_k} \frac{1}{\left(\frac{1}{Q_k} \frac{\lambda_k}{\sigma^2 + \lambda_k} - 1\right) e^{-8\eta\tau\lambda_k} + 1} - 1 \right) \quad (273)$$

$$= \frac{Q_k e^{8\eta\tau\lambda_k} \log(\lambda_k + Q_k (e^{8\eta\tau\lambda_k} - 1) (\lambda_k + \sigma^2)) - 2Q_k \log(\sigma) + 2 \log(\sigma)}{2(e^{8\eta\tau\lambda_k} - 1) Q_k + 2} \quad (274)$$

$$= \frac{Q_k e^{8\eta\tau\lambda_k} \log(\lambda_k + Q_k (e^{8\eta\tau\lambda_k} - 1) (\lambda_k + \sigma^2)) + 2(1 - Q_k) \log(\sigma)}{2(e^{8\eta\tau\lambda_k} - 1) Q_k + 2} \quad (275)$$

$$= \frac{Q_k \log(\lambda_k + Q_k (e^{8\eta\tau\lambda_k} - 1) (\lambda_k + \sigma^2)) + 2(1 - Q_k) \log(\sigma) e^{-8\eta\tau\lambda_k}}{2(1 - e^{-8\eta\tau\lambda_k}) Q_k + 2e^{-8\eta\tau\lambda_k}} \quad (276)$$

$$= \frac{Q_k \log[e^{8\eta\tau\lambda_k} (\lambda_k e^{-8\eta\tau\lambda_k} + Q_k (1 - e^{-8\eta\tau\lambda_k}) (\lambda_k + \sigma^2))] + 2(1 - Q_k) \log(\sigma) e^{-8\eta\tau\lambda_k}}{2Q_k + 2(1 - Q_k) e^{-8\eta\tau\lambda_k}} \quad (277)$$

$$= \frac{Q_k \log(\lambda_k e^{-8\eta\tau\lambda_k} + Q_k (1 - e^{-8\eta\tau\lambda_k}) (\lambda_k + \sigma^2)) + 8\eta\tau\lambda_k Q_k + 2(1 - Q_k) \log(\sigma) e^{-8\eta\tau\lambda_k}}{2Q_k + 2(1 - Q_k) e^{-8\eta\tau\lambda_k}} \quad (278)$$

1105 Similarly, the general solution to the sampling ODE is

$$\begin{aligned} c_k(\sigma) &= C \exp\left(\frac{Q_k \log(\lambda_k e^{-8\eta\tau\lambda_k} + Q_k (1 - e^{-8\eta\tau\lambda_k}) (\lambda_k + \sigma^2)) + 8\eta\tau\lambda_k Q_k + 2(1 - Q_k) \log(\sigma) e^{-8\eta\tau\lambda_k}}{2Q_k + 2(1 - Q_k) e^{-8\eta\tau\lambda_k}}\right) \\ &= C \left[\lambda_k e^{-8\eta\tau\lambda_k} + Q_k (1 - e^{-8\eta\tau\lambda_k}) (\lambda_k + \sigma^2) \right]^{\frac{Q_k}{2Q_k + 2(1 - Q_k) e^{-8\eta\tau\lambda_k}}} \\ &\quad \exp\left(\frac{8\eta\tau\lambda_k Q_k}{2Q_k + 2(1 - Q_k) e^{-8\eta\tau\lambda_k}}\right) \exp\left(\frac{(1 - Q_k) e^{-8\eta\tau\lambda_k} \log(\sigma)}{Q_k + (1 - Q_k) e^{-8\eta\tau\lambda_k}}\right) \end{aligned} \quad (279)$$

1106 The scaling ratio is

$$\frac{c_k(\sigma_0)}{c_k(\sigma_T)} = \left[\frac{\lambda_k e^{-8\eta\tau\lambda_k} + Q_k (1 - e^{-8\eta\tau\lambda_k}) (\lambda_k + \sigma_0^2)}{\lambda_k e^{-8\eta\tau\lambda_k} + Q_k (1 - e^{-8\eta\tau\lambda_k}) (\lambda_k + \sigma_T^2)} \right]^{\frac{Q_k}{2Q_k + 2(1 - Q_k) e^{-8\eta\tau\lambda_k}}} \quad (280)$$

$$\begin{aligned} &\exp\left(\frac{(1 - Q_k) e^{-8\eta\tau\lambda_k} (\log(\sigma_0) - \log(\sigma_T))}{Q_k + (1 - Q_k) e^{-8\eta\tau\lambda_k}}\right) \\ &= \left[\frac{\lambda_k e^{-8\eta\tau\lambda_k} + Q_k (1 - e^{-8\eta\tau\lambda_k}) (\lambda_k + \sigma_0^2)}{\lambda_k e^{-8\eta\tau\lambda_k} + Q_k (1 - e^{-8\eta\tau\lambda_k}) (\lambda_k + \sigma_T^2)} \right]^{\frac{Q_k}{2Q_k + 2(1 - Q_k) e^{-8\eta\tau\lambda_k}}} \end{aligned} \quad (281)$$

$$\left(\frac{\sigma_0}{\sigma_T} \right)^{\frac{(1 - Q_k) e^{-8\eta\tau\lambda_k}}{Q_k + (1 - Q_k) e^{-8\eta\tau\lambda_k}}} \quad (282)$$

$$\Phi_k(\sigma) = \left[\lambda_k e^{-8\eta\tau\lambda_k} + Q_k (1 - e^{-8\eta\tau\lambda_k}) (\lambda_k + \sigma^2) \right]^{\frac{Q_k}{2Q_k + 2(1-Q_k)e^{-8\eta\tau\lambda_k}}} (\sigma)^{\frac{(1-Q_k)e^{-8\eta\tau\lambda_k}}{Q_k + (1-Q_k)e^{-8\eta\tau\lambda_k}}}$$

1107 **Learning Asymptotics: early and late training** At late training stage $\eta\tau \rightarrow \infty$,

$$\lim_{\eta\tau \rightarrow \infty} \frac{c_k(\sigma_0)}{c_k(\sigma_T)} = \left[\frac{Q_k (\lambda_k + \sigma_0^2)}{Q_k (\lambda_k + \sigma_T^2)} \right]^{\frac{Q_k}{2Q_k}} \exp\left(\frac{0}{Q_k}\right) \quad (283)$$

$$= \sqrt{\frac{(\lambda_k + \sigma_0^2)}{(\lambda_k + \sigma_T^2)}} \quad (284)$$

1108 At early training stage $\eta\tau \rightarrow 0$,

$$\lim_{\eta\tau \rightarrow 0} \frac{c_k(\sigma_0)}{c_k(\sigma_T)} = \left[\frac{\lambda_k 1 + Q_k (1 - 1) (\lambda_k + \sigma_0^2)}{\lambda_k 1 + Q_k (1 - 1) (\lambda_k + \sigma_T^2)} \right]^{\frac{Q_k}{2Q_k + 2(1-Q_k)}} \left(\frac{\sigma_0}{\sigma_T} \right)^{\frac{(1-Q_k)}{Q_k + (1-Q_k)}} \quad (285)$$

$$= \left[1 \right]^{\frac{Q_k}{2Q_k + 2(1-Q_k)}} \left(\frac{\sigma_0}{\sigma_T} \right)^{1-Q_k} \quad (286)$$

$$= \left(\frac{\sigma_0}{\sigma_T} \right)^{1-Q_k} \quad (287)$$

1109 **Evolution of distribution** Following the derivation above,

$$\Sigma[\mathbf{x}(\sigma_0)] = \sum_k (\sigma_T \frac{c_k(\sigma_0)}{c_k(\sigma_T)})^2 \mathbf{u}_k \mathbf{u}_k^T$$

1110 The generated variance along eigenvector \mathbf{u}_k is

$$\begin{aligned} \tilde{\lambda}_k(\tau) &= (\sigma_T \frac{c_k(\sigma_0)}{c_k(\sigma_T)})^2 \\ &= \sigma_T^2 \left[\frac{\lambda_k e^{-8\eta\tau\lambda_k} + Q_k (1 - e^{-8\eta\tau\lambda_k}) (\lambda_k + \sigma_0^2)}{\lambda_k e^{-8\eta\tau\lambda_k} + Q_k (1 - e^{-8\eta\tau\lambda_k}) (\lambda_k + \sigma_T^2)} \right]^{\frac{Q_k}{Q_k + (1-Q_k)e^{-8\eta\tau\lambda_k}}} \left(\frac{\sigma_0}{\sigma_T} \right)^{\frac{2(1-Q_k)e^{-8\eta\tau\lambda_k}}{Q_k + (1-Q_k)e^{-8\eta\tau\lambda_k}}} \end{aligned} \quad (288)$$

1111 **F Detailed Derivations for Deep Linear network**

1112 Consider a general deep linear network. $\mathbf{W} = \prod_i \mathbf{W}_i = \mathbf{W}_L \mathbf{W}_{L-1} \dots \mathbf{W}_1$

1113 **Gradient structure** Using the chain rule

$$\nabla_y \mathcal{L} = \left(\frac{\partial \mathcal{L}}{\partial \mathbf{W}} \right)_{ij} \frac{\partial \mathbf{W}_{ij}}{\partial y}$$

1114 we have gradient to matrix \mathbf{W}_l

$$\frac{\partial \mathbf{W}_{mn}}{\partial \mathbf{W}_{l,ij}} = (\mathbf{W}_L \dots \mathbf{W}_{l+1})_{mi} (\mathbf{W}_{l-1} \dots \mathbf{W}_1)_{jn}$$

1115 In vector notation,

$$\begin{aligned} [\nabla_{\mathbf{W}_l} \mathcal{L}]_{ij} &= \left(\frac{\partial \mathcal{L}}{\partial \mathbf{W}} \right)_{mn} \frac{\partial \mathbf{W}_{mn}}{\partial \mathbf{W}_{l,ij}} \\ &= (\mathbf{W}_L \dots \mathbf{W}_{l+1})_{mi} \left(\frac{\partial \mathcal{L}}{\partial \mathbf{W}} \right)_{mn} (\mathbf{W}_{l-1} \dots \mathbf{W}_1)_{jn} \\ \nabla_{\mathbf{W}_l} \mathcal{L} &= (\mathbf{W}_L \dots \mathbf{W}_{l+1})^T \left(\frac{\partial \mathcal{L}}{\partial \mathbf{W}} \right) (\mathbf{W}_{l-1} \dots \mathbf{W}_1)^T \end{aligned}$$

1116 where $\nabla_{\mathbf{W}} \mathcal{L}$ can be found in (5)

$$\begin{aligned} \nabla_b \mathcal{L} &= 2(b - (I - \mathbf{W})\mu) \\ \nabla_{\mathbf{W}} \mathcal{L} &= -2\Sigma + 2\mathbf{W}(\sigma^2 I + \Sigma) + [2\mathbf{W}\mu\mu^T + 2(b - \mu)\mu^T] \end{aligned}$$

1117 **F.1 Aligned assumption**

1118 This gradient structure could potentially be substantially simplified if the weight matrices are aligned
1119 at the initialization.

1120 Consider the singular value decomposition of each weight matrix \mathbf{W}_l

$$\mathbf{W}_l = U_l \Lambda_l V_l^T$$

1121 and for each neighboring layers the singular mode are aligned .

$$V_l^T U_{l-1} = I$$

1122 or equivalently $U_{l-1} = V_l, \forall l \in [2, L]$

1123 **Gradient structure under aligned assumption** Then the product of weight matrix

$$\mathbf{W}_L \dots \mathbf{W}_{l+1} = U_L \prod_{k=l+1}^L \Lambda_k V_{l+1}^T$$

$$\mathbf{W}_{l-1} \dots \mathbf{W}_1 = U_{l-1} \prod_{k=1}^{l-1} \Lambda_k V_1^T$$

$$\mathbf{W}_L \dots \mathbf{W}_1 = U_L \prod_{k=1}^L \Lambda_k V_1^T$$

1126 Then the evolution of hidden weights read

$$\begin{aligned} \nabla_{\mathbf{W}_l} \mathcal{L} &= (U_L \prod_{k=l+1}^L \Lambda_k V_{l+1}^T)^T \nabla_{\mathbf{W}} \mathcal{L} (U_{l-1} \prod_{k=1}^{l-1} \Lambda_k V_1^T)^T \\ &= V_{l+1} \prod_{k=l+1}^L \Lambda_k U_L^T \nabla_{\mathbf{W}} \mathcal{L} V_1 \prod_{k=1}^{l-1} \Lambda_k U_{l-1}^T \\ &= U_l \left[\prod_{k=l+1}^L \Lambda_k U_L^T \nabla_{\mathbf{W}} \mathcal{L} V_1 \prod_{k=1}^{l-1} \Lambda_k \right] V_l^T \end{aligned}$$

1127 Bring in the gradient to the loss (5), assuming centered data $\mu = 0$,

$$\nabla_{\mathbf{W}} \mathcal{L}_{\mu=0} = -2\Sigma + 2\mathbf{W}(\sigma^2 I + \Sigma)$$

1128 Then

$$\begin{aligned} \nabla_{\mathbf{W}_l} \mathcal{L} &= U_l \left[\prod_{k=l+1}^L \Lambda_k U_L^T (-2\Sigma + 2\mathbf{W}(\sigma^2 I + \Sigma)) V_1 \prod_{k=1}^{l-1} \Lambda_k \right] V_l^T \\ &= U_l \left[\prod_{k=l+1}^L \Lambda_k U_L^T (-2\Sigma + 2U_L \prod_{k=1}^L \Lambda_k V_1^T (\sigma^2 I + \Sigma)) V_1 \prod_{k=1}^{l-1} \Lambda_k \right] V_l^T \\ &= -2U_l \prod_{k=l+1}^L \Lambda_k U_L^T \Sigma V_1 \prod_{k=1}^{l-1} \Lambda_k V_l^T + 2\sigma^2 U_l \prod_{k=l+1}^L \Lambda_k \prod_{k=1}^L \Lambda_k \prod_{k=1}^{l-1} \Lambda_k V_l^T \\ &\quad + 2U_l \prod_{k=l+1}^L \Lambda_k \prod_{k=1}^L \Lambda_k V_1^T \Sigma V_1 \prod_{k=1}^{l-1} \Lambda_k V_l^T \end{aligned}$$

1129 Simplifying assumption, if $U_L = V_1$ and they exactly match the eigen basis of Σ , U , then

$$V_1^T \Sigma V_1 = \Lambda, \quad U_L^T \Sigma V_1 = \Lambda$$

1130

$$\begin{aligned} \nabla_{\mathbf{W}_l} \mathcal{L} &= -2U_l \Lambda \prod_{k=l+1}^L \Lambda_k \prod_{k=1}^{l-1} \Lambda_k V_l^T + 2\sigma^2 U_l \prod_{k=l+1}^L \Lambda_k \prod_{k=1}^L \Lambda_k \prod_{k=1}^{l-1} \Lambda_k V_l^T + 2U_l \prod_{k=l+1}^L \Lambda_k \prod_{k=1}^L \Lambda_k \Lambda \prod_{k=1}^{l-1} \Lambda_k V_l^T \\ &= 2U_l [-\Lambda + \sigma^2 \prod_{k=1}^L \Lambda_k + \Lambda \prod_{k=1}^L \Lambda_k] \prod_{k=1, k \neq l}^L \Lambda_k V_l^T \\ &= 2U_l [-\Lambda + (\sigma^2 + \Lambda) \Lambda_l \prod_{k=1, k \neq l}^L \Lambda_k] \prod_{k=1, k \neq l}^L \Lambda_k V_l^T \end{aligned}$$

1131 Consider the singular value of each weights

$$\begin{aligned} \nabla_{\Lambda_l} \mathcal{L} &= U_l^T \nabla_{\mathbf{W}_l} \mathcal{L} V_l \\ &= [-\Lambda + (\sigma^2 + \Lambda) \Lambda_l \prod_{k=1, k \neq l}^L \Lambda_k] \prod_{k=1, k \neq l}^L \Lambda_k \end{aligned}$$

1132 The weight dynamics have an surprisingly simple mode-by-mode form. For the m -th mode, the
1133 gradient is

$$\nabla_{\Lambda_{l,m}} \mathcal{L} = [-\lambda_m + (\sigma^2 + \lambda_m) \Lambda_{l,m} \prod_{k=1, k \neq l}^L \Lambda_{k,m}] \prod_{k=1, k \neq l}^L \Lambda_{k,m}$$

1134 To simplify notation, we define $\Xi_{l,m} := \prod_{k=1, k \neq l}^L \Lambda_{k,m}$. Then the gradient flow dynamics reads,

$$\begin{aligned} \frac{d}{d\tau} \Lambda_{l,m} &= -\eta [-\lambda_m + (\sigma^2 + \lambda_m) \Lambda_{l,m} \prod_{k=1, k \neq l}^L \Lambda_{k,m}] \prod_{k=1, k \neq l}^L \Lambda_{k,m} \\ &= -\eta [-\lambda_m + (\sigma^2 + \lambda_m) \Lambda_{l,m} \Xi_{l,m}] \Xi_{l,m} \end{aligned}$$

1135 From the first glance, this is a linear system for $\Lambda_{l,m}$, with fixed point at $\lambda_m / (\sigma^2 + \lambda_m) / \Xi_{l,m}$. But
1136 the effect of other layers $\Xi_{l,m}$ is also dynamics, forming a coupled nonlinear system. $\Xi_{l,m}$ will
1137 affects both its time constant and convergence level.

1138 F.2 Two layer linear network

1139 We start by considering two layer case of this equation. Under the aligned initialization assumption,

$$\mathbf{W}_1 = U_1 \Lambda_1 V_1^T, \quad \mathbf{W}_2 = U_2 \Lambda_2 V_2^T \quad (289)$$

$$U_1 = V_2, \quad U_2 = V_1 = U \quad (290)$$

1140 So the main degree of freedom comes from Λ_1, Λ_2 , and we discuss the two cases whether they equal
1141 each other.

1142 **F.2.1 Special case: homogeneous initialization $\Lambda_1 = \Lambda_2$ (symmetric two layer case)**

1143 If $\Lambda_1 = \Lambda_2$, then

$$\mathbf{W}_1 = U_1 \Lambda_1 V_1^T = V_2 \Lambda_2 U_2^T = \mathbf{W}_2^T$$

1144 We are reduced to the symmetric two layer case $\mathbf{W} = \mathbf{W}_1^T \mathbf{W}_1$ as we discussed above E. Due to
1145 weight sharing, the gradients to each layer are accumulated.

$$\begin{aligned}\nabla_{\Lambda_1} \mathcal{L} &= 2[-\Lambda + (\sigma^2 + \Lambda) \Lambda_1 \Lambda_2] \Lambda_2 \\ &= 2[-\Lambda + (\sigma^2 + \Lambda) \Lambda_1^2] \Lambda_1\end{aligned}$$

1146

$$\begin{aligned}\frac{d}{d\tau} \Lambda_1 &= -2\eta[-\Lambda + (\sigma^2 + \Lambda) \Lambda_1^2] \Lambda_1 \\ \frac{d}{d\tau} (\Lambda_1)^2 &= -4\eta[-\Lambda + (\sigma^2 + \Lambda) \Lambda_1^2] \Lambda_1^2\end{aligned}$$

1147 This recovers the dynamics we got in last section 5.1.

1148 **F.2.2 General Case: general initialization (general two layer $\mathbf{W} = PQ$)**

1149 Generally, if the initialization is not homogeneous, $\Lambda_1 \neq \Lambda_2$, the matrices are not the same $\mathbf{W} = PQ$,
1150 we have the general two layer parametrization.

1151 Under aligned assumption F.1, the weight learning dynamics are reduced to that of Λ_1, Λ_2 . For
1152 simplicity of notation let $f_k = \Lambda_{1,k}, g_k = \Lambda_{2,k}$. We have

$$\begin{aligned}\nabla_{f_k} \mathcal{L} &= -\lambda_k g_k + (\sigma^2 + \lambda_k) g_k^2 f_k \\ \nabla_{g_k} \mathcal{L} &= -\lambda_k f_k + (\sigma^2 + \lambda_k) f_k^2 g_k\end{aligned}$$

1153 which is a two dimensional coupled system. Lets focus on the dynamics along one eigenmode k . and
1154 drop the subscript k from f_k, g_k . Let the constants $A = \eta \lambda_k, B = \eta(\sigma^2 + \lambda_k)$, we have

$$\begin{aligned}\frac{d}{d\tau} f &= Ag - Bg^2 f \\ \frac{d}{d\tau} g &= Af - Bf^2 g\end{aligned}$$

1155 There are three important variables $f_k g_k$ and $f_k^2 + g_k^2$ and $f^2 - g^2$, This system can be represented in
1156 the following way, which exposes its conserved quantity

$$\begin{aligned}\frac{d}{d\tau} (fg) &= f \frac{d}{d\tau} g + g \frac{d}{d\tau} f \\ &= Af^2 - Bf^3 g + Ag^2 - Bg^3 f \\ &= (f^2 + g^2) (A - Bfg) \\ \frac{d}{d\tau} (f^2 + g^2) &= 2f \frac{d}{d\tau} f + 2g \frac{d}{d\tau} g \\ &= 2(Afg - Bg^2 f^2) + 2(Afg - Bf^2 g^2) \\ &= 2fg (A - Bfg) \\ \frac{d}{d\tau} (f^2 - g^2) &= 0\end{aligned}$$

1157 Thus the (f, g) pair can only flow along hyperbolic lines or diagonal lines where $f^2 - g^2 = \text{const.}$
1158 We can leverage this fact to write down the overall solution.

1159 Notice that

$$(f^2 + g^2)^2 - (f^2 - g^2)^2 = 4f^2 g^2$$

1160 So we can represent

$$\begin{aligned}f^2 + g^2 &= \sqrt{4f^2 g^2 + (f^2 - g^2)^2} \\ &= \sqrt{4f^2 g^2 + C^2}\end{aligned}$$

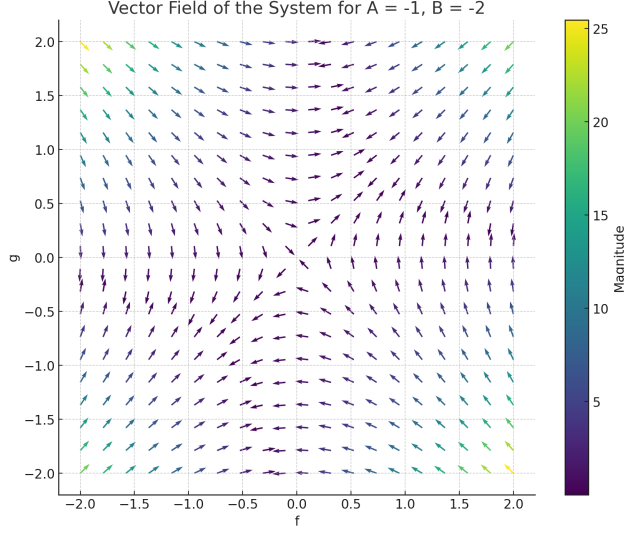


Figure 22: **Phase diagram for the simplified 2d dynamic system.** above $\frac{d}{d\tau}f = Ag - Bg^2f$, $\frac{d}{d\tau}g = Af - Bf^2g$ for $A = 1, B = 2$ we can see the manifold of stable attractors in 1 and 3rd quadrangle. $fg = A/B$, and the conserved quantity along the hyperbolic lines.

1161 Now the equation for fg becomes closed

$$\frac{d}{d\tau}(fg) = \sqrt{4f^2g^2 + C^2}(A - Bfg)$$

1162 Let $h = fg$

$$\frac{d}{d\tau}h = \sqrt{4h^2 + C^2}(A - Bh)$$

1163 Note that for this self contained equation, unless $C = 0$, it shall have only one attractive fixed point
1164 which is $h^* = A/B$. Thus asymptotically, it will always converge to the correct value.

1165 When we face the weight tying case where $C = 0$, the equation becomes

$$\frac{d}{d\tau}h = 2|h|(A - Bh)$$

1166 it will have another fixed point at $h = 0$, which made it impossible to converge to some fixed point
1167 A/B , if the initialization $h(0)$ has opposite sign.

1168 E.3 General deep linear network

1169 **Weight tying assumption / initialization** Note, when there is weight tying, it can be regarded as
1170 special case. The key gradient equation is

$$\nabla_{\Lambda_l} \mathcal{L} = [-\Lambda + (\sigma^2 + \Lambda)\Lambda_l \prod_{k=1, k \neq l}^L \Lambda_k] \prod_{k=1, k \neq l}^L \Lambda_k$$

1171 Let's assume all Λ_l are equal at initialization¹. Then

$$\nabla_{\Lambda_l} \mathcal{L} = [-\Lambda + (\sigma^2 + \Lambda)\Lambda_l^L] \Lambda_l^{L-1}$$

1172 Consider a single element at mode k , $a_k^{(l)} = \Lambda_{l,k}$, then we have

$$\frac{\partial \mathcal{L}}{\partial a_k^{(l)}} = [-\lambda_k + (\sigma^2 + \lambda_k) (a_k^{(l)})^L] (a_k^{(l)})^{L-1}$$

¹ Λ_l not to be confused with Λ which is the spectrum of data

1173 Further consider the aggregated effect $c_k = \prod_l a_k^{(l)}$, assume weight tying, then gradient flow of this
 1174 variable reads

$$\begin{aligned}\frac{\partial c_k}{\partial \tau} &= L(a_k^{(l)})^{L-1} \frac{\partial a_k^{(l)}}{\partial \tau} \\ &= -\eta L(a_k^{(l)})^{L-1} \frac{\partial \mathcal{L}}{\partial a_k^{(l)}} \\ &= -\eta L [-\lambda_k + (\sigma^2 + \lambda_k) (a_k^{(l)})^L] (a_k^{(l)})^{2L-2} \\ &= -\eta L [-\lambda_k + (\sigma^2 + \lambda_k) c_k] c_k^{\frac{2L-2}{L}} \\ &= -\eta L [-\lambda_k + (\sigma^2 + \lambda_k) c_k] c_k^{2-\frac{2}{L}}\end{aligned}$$

1175 We arrived at the dynamics of the weights along eigenmode, note this form also encompasses the
 1176 solution of one layer and two layer symmetric linear model by setting $L = 1, 2$

$$\frac{\partial c_k}{\partial \tau} = \eta L [\lambda_k - (\sigma^2 + \lambda_k) c_k] c_k^{2-\frac{2}{L}}$$

1177 For general L , there is no closed-form solution, one only have implicit solution to c_k involving the
 1178 hyper geometric function ${}_2F_1$,

$$\left(\frac{A c(\tau)}{B}\right)^{2/L} {}_2F_1\left(1, \frac{2}{L}; 1 + \frac{2}{L}; \frac{A c(\tau)}{B}\right) = \left(\frac{A c(0)}{B}\right)^{2/L} {}_2F_1\left(1, \frac{2}{L}; 1 + \frac{2}{L}; \frac{A c(0)}{B}\right) e^{-\eta L B \tau}.$$

1179

$$\begin{aligned}A &= (\sigma^2 + \lambda_k) \\ B &= \lambda_k\end{aligned}$$

1180 **Infinite depth limit** The limit case is $L \rightarrow \infty$, then the dynamics read

$$\begin{aligned}\frac{\partial c_k}{\partial \tau} &= -\eta L [-\lambda_k + (\sigma^2 + \lambda_k) c_k] c_k^{2-\frac{2}{L}} \\ (L \rightarrow \infty) &= \eta L [\lambda_k - (\sigma^2 + \lambda_k) c_k] c_k^2\end{aligned}$$

1181 Then we have

$$\begin{aligned}\frac{dc_k}{[\lambda_k - (\sigma^2 + \lambda_k) c_k] c_k^2} &= \eta L d\tau \\ \frac{\sigma^2 + \lambda_k}{\lambda_k^2} \ln \frac{c_k}{\lambda_k - (\sigma^2 + \lambda_k) c_k} - \frac{1}{\lambda_k c_k} &= \eta L \tau + C.\end{aligned}$$

1182 Setting the initial values as $c_k^{(0)}$ we have the implicit solution of c_k

$$\begin{aligned}\frac{\sigma^2 + \lambda_k}{\lambda_k^2} \ln \frac{c_k}{\lambda_k - (\sigma^2 + \lambda_k) c_k} - \frac{1}{\lambda_k c_k} &= \eta L \tau + \frac{\sigma^2 + \lambda_k}{\lambda_k^2} \ln \frac{c_k^{(0)}}{\lambda_k - (\sigma^2 + \lambda_k) c_k^{(0)}} - \frac{1}{\lambda_k c_k^{(0)}} \\ \tau &= \frac{1}{\eta L} \left[\frac{\sigma^2 + \lambda_k}{\lambda_k^2} \ln \frac{c_k (\lambda_k - (\sigma^2 + \lambda_k) c_k^{(0)})}{c_k^{(0)} (\lambda_k - (\sigma^2 + \lambda_k) c_k)} - \frac{1}{\lambda_k c_k} + \frac{1}{\lambda_k c_k^{(0)}} \right]\end{aligned}$$

1184 F.3.1 Deep Residual network

1185 If the network is parametrized with a residual connection at each layer, i.e. $\mathbf{W} = \prod_i (\mathbf{I} + \mathbf{W}_i) =$
 1186 $(\mathbf{I} + \mathbf{W}_L)(\mathbf{I} + \mathbf{W}_{L-1}) \dots (\mathbf{I} + \mathbf{W}_1)$, Then it's a linear shift in parametrization, using the same derivation
 1187 and notation, assume aligned initialization, and homogeneous initialization, we can see

$$\frac{\partial \mathcal{L}}{\partial a_k^{(l)}} = \left[-\lambda_k + (\sigma^2 + \lambda_k) (1 + a_k^{(l)})^L \right] (1 + a_k^{(l)})^{L-1}$$

1188 Let the overall effective weight as $c_k = (1 + a_k^{(l)})^L$

$$\frac{\partial c_k}{\partial a_k^{(l)}} = L (1 + a_k^{(l)})^{L-1}$$

1189 The dynamics of effective weight is

$$\begin{aligned} \frac{\partial c_k}{\partial \tau} &= L (1 + a_k^{(l)})^{L-1} \frac{\partial a_k^{(l)}}{\partial \tau} \\ &= -\eta L (1 + a_k^{(l)})^{L-1} \nabla_{a_k^{(l)}} \mathcal{L} \\ &= -\eta L (1 + a_k^{(l)})^{L-1} \left[-\lambda_k + (\sigma^2 + \lambda_k) (1 + a_k^{(l)})^L \right] (1 + a_k^{(l)})^{L-1} \\ &= -\eta L \left[-\lambda_k + (\sigma^2 + \lambda_k) (1 + a_k^{(l)})^L \right] (1 + a_k^{(l)})^{2L-2} \\ &= -\eta L \left[-\lambda_k + (\sigma^2 + \lambda_k) c_k \right] c_k^{2-\frac{2}{L}} \end{aligned}$$

1190 We recovers the same dynamics as the normal deep linear network.

$$\frac{\partial c_k}{\partial \tau} = \eta L [\lambda_k - (\sigma^2 + \lambda_k) c_k] c_k^{2-\frac{2}{L}}$$

1191 Here the initialization is $c_k(0) = (1 + a_k^{(l)})^L$ Thus in linear network, residual connection does not
1192 change the learning dynamics, but just potentially changed the initialization of weights.

1193 G Detailed Derivations for Linear Convolutional Network

1194 In this section, we study a linear convolutional denoiser. For simplicity, let's consider the space is
 1195 1d, with a 1d convolutional filter \mathbf{w} . For simplicity, we ignore the bias term and assume zero-mean
 1196 $\mu = 0$. The convolutional denoiser is defined as

$$\mathbf{D}(\mathbf{x}; \sigma) = \mathbf{w}_\sigma * \mathbf{x} \quad (291)$$

1197 This convolution $*$ can be equivalently represented as matrix multiplication, where the weight matrix
 1198 \mathbf{W}_σ is a Toeplitz or Circulants (when circular boundary condition).

$$\mathbf{D}(\mathbf{x}; \sigma) = \mathbf{W}_\sigma \mathbf{x}$$

1199 In this case, the score learning problem becomes a *circulant or Toeplitz regression* problem, which
 1200 has been studied before [56, 57, 58]. This problem is similar to finding the best convolutional or
 1201 deconvolutional kernel to a 1d sequence.

1202 G.1 General set up

1203 **Cyclic weight matrix and spectral representation** Let's consider the circular convolution case,
 1204 where we ignore the boundary effect, and assume the matrix $\mathbf{W}(\sigma)$ is cyclic at any noise scale. Since
 1205 it's cyclic, they can be diagonalized by discrete Fourier transform (DFT).

1206 Let's diagonalize the weight matrix with the DFT matrix,

$$\mathbf{W}(\sigma) = F\Gamma(\sigma)F^*$$

1207 specifically the DFT matrix is defined as

$$F_{mk} = \frac{1}{\sqrt{N}} \exp\left(-2\pi i \frac{mk}{N}\right), \quad m, k = 0, 1, 2, \dots, N-1$$

1208 G.2 General analysis of sampling dynamics

1209 Note that, if we assume weight matrix at every noise scale is circulant, then we always can solve
 1210 the sampling dynamics on the Fourier basis, mode by mode. At its core, this is because all circulant
 1211 matrices commute.

$$\begin{aligned} \frac{d}{d\sigma} \mathbf{x} &= -\frac{F\Gamma(\sigma)F^* \mathbf{x} - \mathbf{x}}{\sigma} \\ F^* \frac{d}{d\sigma} \mathbf{x} &= -\frac{\Gamma(\sigma)F^* \mathbf{x} - F^* \mathbf{x}}{\sigma} \\ &= -\frac{\Gamma(\sigma) - I}{\sigma} F^* \mathbf{x} \end{aligned}$$

1212 We can perform integration mode by mode. Let $c_k = (F^* \mathbf{x})_k$, $\mathbf{x} = F\mathbf{c}$

$$\frac{d}{d\sigma} c_k = -\frac{\gamma_k(\sigma) - 1}{\sigma} c_k$$

1213 Thus, if we know the Fourier parametrization of weights $\Gamma(\sigma)$, we can integrate the sampling of \mathbf{x} in
 1214 closed form. We will need to solve these Fourier mode $\Gamma(\sigma)$ of the weights during training dynamics.
 1215 Integrating the ODE, we get

$$\Phi_k(\sigma) = \exp\left(\int -\frac{\gamma_k(\lambda) - 1}{\lambda} d\lambda\right)$$

1216

$$c_k(\sigma_0) = \frac{\Phi_k(\sigma_0)}{\Phi_k(\sigma_T)} c_k(\sigma_T) \quad (292)$$

1217 Then the generated variance will be

$$\tilde{\lambda}_k = \sigma_T^2 \left(\frac{\Phi_k(\sigma_0)}{\Phi_k(\sigma_T)} \right)^2 \quad (293)$$

$$\tilde{\Sigma} = F \text{diag}(\tilde{\lambda}_k) F^* \quad (294)$$

1218 Because of this we can easily prove Proposition 5.2

1219 *Proof.* Generated distributions from linear denoisers are Gaussian distribution, with covariance $\tilde{\Sigma}$.
 1220 Since the generated covariance is diagonalized by discrete Fourier transform F , $\tilde{\Sigma}$ is circulant i.e.
 1221 translation invariant. Given that the generated \mathbf{x} is Gaussian distributed, \mathbf{x} conform to a stationary
 1222 Gaussian process. While the integration function $\Phi_k(\sigma)$ determines the covariance kernel of this
 1223 Gaussian process. \square

1224 G.3 Full width linear convolutional network

1225 G.3.1 Training dynamics of full width linear convolutional network

1226 **Gradient structure in spectral basis** Using the Fourier representation of the circulant weight

$$\mathbf{W}(\sigma) = F\mathbf{\Gamma}(\sigma)F^*$$

1227 we can derive the gradient to the Fourier parameters $\mathbf{\Gamma}(\sigma)$ which is a diagonal matrix.

1228 Recall that

$$\begin{aligned}\nabla_{\mathbf{b}}\mathcal{L} &= 2\mathbf{b} \\ \nabla_{\mathbf{W}}\mathcal{L} &= -2\Sigma + 2\mathbf{W}(\sigma^2 I + \Sigma)\end{aligned}$$

1229 Then the gradient to the Fourier parametrization reads

$$\begin{aligned}\langle \nabla_{\mathbf{W}}\mathcal{L}, d\mathbf{W} \rangle &= \langle \nabla_{\mathbf{W}}\mathcal{L}, Fd\mathbf{\Gamma}(\sigma)F^* \rangle \\ &= \langle F^{-1}\nabla_{\mathbf{W}}\mathcal{L}F^{-1*}, d\mathbf{\Gamma}(\sigma) \rangle \\ &= \langle F^*\nabla_{\mathbf{W}}\mathcal{L}F, d\mathbf{\Gamma}(\sigma) \rangle\end{aligned}$$

1230 Thus

$$\begin{aligned}\nabla_{\mathbf{\Gamma}}\mathcal{L} &= F^*\nabla_{\mathbf{W}}\mathcal{L}F \\ &= -2F^*\Sigma F + 2F^*\mathbf{W}(\sigma^2 I + \Sigma)F \\ &= -2F^*\Sigma F + 2F^*F\mathbf{\Gamma}F^*(\sigma^2 I + \Sigma)F \\ &= -2F^*\Sigma F + 2\mathbf{\Gamma}(\sigma^2 I + F^*\Sigma F)\end{aligned}$$

1231 Under circular convolution assumption, $\mathbf{\Gamma}$ is a diagonal matrix, $\mathbf{\Gamma}_{ij} = \delta_{ij}\gamma_i$ then

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \gamma_i} &= [\nabla_{\mathbf{\Gamma}}\mathcal{L}]_{ii} \\ &= -2[F^*\Sigma F]_{ii} + 2[\mathbf{\Gamma}(\sigma^2 I + F^*\Sigma F)]_{ii} \\ &= -2[F^*\Sigma F]_{ii} + 2[\delta_{ij}\gamma_i(\sigma^2 I + F^*\Sigma F)]_{ii} \\ &= -2[F^*\Sigma F]_{ii} + 2\gamma_i(\sigma^2 + [F^*\Sigma F]_{ii})\end{aligned}$$

1232 The key entity is the $F^*\Sigma F$ matrix, let's define it as $\tilde{\Sigma}$

1233 Interpretation of $F^*\Sigma F$

- 1234 • We can regard it as covariance matrix in the spectral basis, i.e. covariance matrix of the
 1235 complex variable $F^*\bar{\mathbf{x}} \in \mathbb{C}^N$.

$$\begin{aligned}\Sigma &= \text{cov}(\mathbf{x}) = \mathbb{E}[\bar{\mathbf{x}}\mathbf{x}^T] \\ F^*\Sigma F &= F^*\mathbb{E}[\bar{\mathbf{x}}\mathbf{x}^T]F \\ &= \mathbb{E}[F^*\bar{\mathbf{x}}\mathbf{x}^T F] \\ &= \mathbb{E}[F^*\bar{\mathbf{x}}\mathbf{x}^T F^T] \\ &= \mathbb{E}[(F^*\bar{\mathbf{x}})(F^*\mathbf{x})^\dagger] \\ &= \text{cov}(F^*\mathbf{x})\end{aligned}$$

1236 • Diagonal values tell us about the power spectrum of the data.

$$1237 [F^* \Sigma F]_{ii} = \text{Var}([F^* \tilde{\mathbf{x}}]_i)$$

$$\begin{aligned} F_{jk} &= \frac{1}{\sqrt{N}} e^{-2\pi i jk/N} \\ [F^* \Sigma F]_{jk} &= \sum_{mn} F_{jm}^* \Sigma_{mn} F_{nk} \\ &= \frac{1}{N} \sum_{mn} \Sigma_{mn} e^{+2\pi i jm/N} e^{-2\pi i nk/N} \\ &= \frac{1}{N} \sum_{mn} \Sigma_{mn} e^{2\pi i \frac{jm-kn}{N}} \\ [F^* \Sigma F]_{jj} &= \frac{1}{N} \sum_{mn} \Sigma_{mn} e^{2\pi i \frac{j(m-n)}{N}} \end{aligned}$$

1238 **Gradient flow on spectral parameter** Given the gradient structure,

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \gamma_k} &= -2[F^* \Sigma F]_{kk} + 2\gamma_k (\sigma^2 + [F^* \Sigma F]_{kk}) \\ &= -2\tilde{\Sigma}_{kk} + 2\gamma_k (\sigma^2 + \tilde{\Sigma}_{kk}) \end{aligned}$$

1239 if we *directly perform gradient descent* on the γ_k variable, we have

$$\frac{d}{d\tau} \gamma_k = -\eta \frac{\partial \mathcal{L}}{\partial \gamma_k}$$

1240

$$\begin{aligned} \frac{d}{d\tau} \gamma_k &= -\eta \left[-2\tilde{\Sigma}_{kk} + 2\gamma_k (\sigma^2 + \tilde{\Sigma}_{kk}) \right] \\ &= 2\eta \left[\tilde{\Sigma}_{kk} - \gamma_k (\sigma^2 + \tilde{\Sigma}_{kk}) \right] \end{aligned}$$

1241 **Fixed point** Fixed point of the gradient flow is

$$\gamma_k^* = \frac{\tilde{\Sigma}_{kk}}{\sigma^2 + \tilde{\Sigma}_{kk}}$$

1242 **Learning dynamics of Fourier modes** This is basically a first order dynamics ODE

$$\begin{aligned} \gamma_k(\tau) &= \frac{\tilde{\Sigma}_{kk}}{\sigma^2 + \tilde{\Sigma}_{kk}} + \left(\gamma_k(0) - \frac{\tilde{\Sigma}_{kk}}{\sigma^2 + \tilde{\Sigma}_{kk}} \right) e^{-2\eta(\sigma^2 + \tilde{\Sigma}_{kk})\tau} \\ &= \gamma_k^* + (\gamma_k(0) - \gamma_k^*) e^{-2\eta(\sigma^2 + \tilde{\Sigma}_{kk})\tau} \end{aligned}$$

1243 **Remarks**

- 1244 • This is exactly the same story as the one layer linear case, where $\sigma^2 + \tilde{\Sigma}_{kk}$ determines the
- 1245 convergence speed per mode.
- 1246 • Spectral model $\tilde{\Sigma}_{kk}$ with higher power will converge faster. -> usually the lower spatial
- 1247 frequency modes.
- 1248 • Higher noise level will converge faster

1249 **Learning dynamics of weight entry as rotated Fourier mode dynamics**

1250 **Lemma G.1** (Convolution Spectrum-entry relation). *The relationship between spectral and entry*
 1251 *parametrization of filter weights is*

$$w_\ell = \frac{1}{N} \sum_{k=0}^{N-1} e^{2\pi i \frac{k\ell}{N}} \gamma_k, \quad \ell = 0, \dots, N-1$$

1252 *Proof. Derivation* Using our previous convention

$$\mathbf{W}_{ij} = w_{j-i}$$

1253 Let the vector of the first row in \mathbf{W} be

$$\mathbf{W}_{0k} = w_k$$

1254 Let the vector parameter be

$$\begin{aligned}\mathbf{w} &= [w_0, w_1, \dots, w_{N-1}]^T \\ &= (\mathbf{W}_{0k})^T\end{aligned}$$

1255 Then the connection between γ_k and weights w_l are the following

$$\mathbf{W} = \mathbf{F}\mathbf{\Gamma}\mathbf{F}^*$$

1256

$$\begin{aligned}\mathbf{W}_{jk} &= \sum_{m,n} F_{jm} \Gamma_{mn} F_{nk}^* \\ &= \sum_m \gamma_m F_{jm} F_{mk}^* \\ &= \sum_m \gamma_m F_{jm} F_{mk}^* \\ &= \frac{1}{N} \sum_m \gamma_m e^{-2\pi i jm/N} e^{+2\pi i mk/N} \\ &= \frac{1}{N} \sum_m \gamma_m e^{2\pi i \frac{(k-j)m}{N}}\end{aligned}$$

1257 Thus

$$\begin{aligned}w_k &= \mathbf{W}_{j,j+k} \\ &= \frac{1}{N} \sum_m e^{+2\pi i \frac{km}{N}} \gamma_m \\ &= \frac{1}{\sqrt{N}} \sum_m F_{km}^* \gamma_m\end{aligned}$$

1258

□

1259 In vector notation, we have

$$\begin{aligned}\mathbf{w} &= \frac{1}{\sqrt{N}} \mathbf{F}^* \boldsymbol{\gamma} \\ \boldsymbol{\gamma} &= \sqrt{N} \mathbf{F} \mathbf{w}\end{aligned}$$

1260 Thus, they are related by a \sqrt{N} scaling and a unitary transform.

1261 Similarly, the gradient to $\boldsymbol{\gamma}$ and that of \mathbf{w} has following relation

$$\begin{aligned}\nabla_{\mathbf{w}} \mathcal{L} &= \sqrt{N} \mathbf{F}^{-T} \nabla_{\boldsymbol{\gamma}} \mathcal{L} \\ &= \sqrt{N} \mathbf{F}^* \nabla_{\boldsymbol{\gamma}} \mathcal{L}\end{aligned}$$

1262 *Proof. Derivation*

$$\begin{aligned}&\langle \nabla_{\boldsymbol{\gamma}} \mathcal{L}, d\boldsymbol{\gamma} \rangle \\ &= \langle \nabla_{\boldsymbol{\gamma}} \mathcal{L}, \sqrt{N} \mathbf{F} d\mathbf{w} \rangle \\ &= \langle \sqrt{N} \mathbf{F}^{-T} \nabla_{\boldsymbol{\gamma}} \mathcal{L}, d\mathbf{w} \rangle \\ &= \langle \nabla_{\mathbf{w}} \mathcal{L}, d\mathbf{w} \rangle\end{aligned}$$

1263

□

1264 If we let the optimization parameter be the filters \mathbf{w} , then we have gradient flow in \mathbf{w} space

$$\frac{d}{d\tau}\mathbf{w} = -\eta\nabla_{\mathbf{w}}\mathcal{L}$$

1265 The corresponding gradient flow of γ is

$$\begin{aligned}\frac{d}{d\tau}\mathbf{w} &= -\eta\nabla_{\mathbf{w}}\mathcal{L} \\ \frac{d}{d\tau}\frac{1}{\sqrt{N}}F^*\gamma &= -\eta\sqrt{N}F^*\nabla_{\gamma}\mathcal{L} \\ \frac{d}{d\tau}\gamma &= -\eta N\nabla_{\gamma}\mathcal{L}\end{aligned}$$

1266 So gradient flow in \mathbf{w} space is equivalent to gradient flow in γ , but scaling learning rate by $\eta \rightarrow N\eta$!

1267 Conversely if we treat γ as optimization parameter and use gradient flow, then it's equivalent to scale
1268 learning rate $\eta \rightarrow \eta/N$

$$\begin{aligned}\frac{d}{d\tau}\gamma &= -\eta\nabla_{\gamma}\mathcal{L} \\ \frac{d}{d\tau}\sqrt{N}F\mathbf{w} &= -\eta\frac{1}{\sqrt{N}}F\nabla_{\mathbf{w}}\mathcal{L} \\ \frac{d}{d\tau}\mathbf{w} &= -\frac{\eta}{N}\nabla_{\mathbf{w}}\mathcal{L}\end{aligned}$$

1269 *Remark G.2.* • Thus we can solve gradient flow in any representation, but translating to
1270 solution of the other variable just by rescaling the learning rate.

1271 • This trick only works when the kernel width is N , or the spectral parameter will be con-
1272 strained in a subspace, which will alter its dynamics.

1273 • Direct spectral parametrization without any constraint is basically equivalent to the entry
1274 wise parametrization. It does not provide extra inductive bias.

1275 **Corollary G.3.** For linear circular convolutional denoiser, $\mathbf{D}(\mathbf{x}, \sigma) = \mathbf{w} * \mathbf{x}$, the solution to gradient
1276 flow on filter weight \mathbf{w} is $\mathbf{w}(\tau, \sigma) = \frac{1}{\sqrt{N}}F^*\gamma(\tau, \sigma)$, where the spectral parameter of k -th Fourier
1277 mode evolves as

$$\begin{aligned}\gamma_k(\tau, \sigma) &= \frac{\tilde{\Sigma}_{kk}}{\sigma^2 + \tilde{\Sigma}_{kk}} + \left(\gamma_k(\tau, \sigma) - \frac{\tilde{\Sigma}_{kk}}{\sigma^2 + \tilde{\Sigma}_{kk}}\right)e^{-2N\eta(\sigma^2 + \tilde{\Sigma}_{kk})\tau} \\ &= \gamma_k^*(\sigma) + (\gamma_k(\tau, \sigma) - \gamma_k^*(\sigma))e^{-2N\eta(\sigma^2 + \tilde{\Sigma}_{kk})\tau}\end{aligned}$$

1278 where $\gamma_k^*(\sigma) = \frac{\tilde{\Sigma}_{kk}}{\sigma^2 + \tilde{\Sigma}_{kk}}$, and $\tilde{\Sigma}_{kk}$ is the variance of Fourier mode.

1279 G.3.2 Sampling dynamics of full width linear convolutional network

1280 Let's project the sampling equation on Fourier modes,

$$F^*\frac{d}{d\sigma}\mathbf{x} = -\frac{\Gamma(\sigma) - I}{\sigma}F^*\mathbf{x}$$

1281 Let $c_k = (F^*\mathbf{x})_k$, $\mathbf{x} = F\mathbf{c}$,

$$\frac{d}{d\sigma}c_k = -\frac{\gamma_k(\sigma) - 1}{\sigma}c_k$$

1282 The variance amplification factor is expressed through in the integral,

$$\Phi_k(\sigma) = \exp\left(\int -\frac{\gamma_k(\lambda) - 1}{\lambda}d\lambda\right)$$

1283 **Generated distribution after convergence** For the converged denoiser we have $\gamma_k^* = \frac{\tilde{\Sigma}_{kk}}{\sigma^2 + \tilde{\Sigma}_{kk}}$

$$\begin{aligned}\Phi_k(\sigma) &= \exp \left(\int_{\epsilon}^{\sigma} -\frac{\gamma_k(\lambda) - 1}{\lambda} d\lambda \right) \\ &= \exp \left(\int_{\epsilon}^{\sigma} -\frac{\frac{\tilde{\Sigma}_{kk}}{\lambda^2 + \tilde{\Sigma}_{kk}} - 1}{\lambda} d\lambda \right) \\ &= \exp \left(\int_{\epsilon}^{\sigma} \frac{\lambda}{\lambda^2 + \tilde{\Sigma}_{kk}} d\lambda \right) \\ &= \exp \left(\frac{1}{2} \ln(\lambda^2 + \tilde{\Sigma}_{kk}) \Big|_{\lambda=\epsilon}^{\lambda=\sigma} \right) \\ &= \sqrt{\frac{\sigma^2 + \tilde{\Sigma}_{kk}}{\epsilon^2 + \tilde{\Sigma}_{kk}}}\end{aligned}$$

1284 **Variance of the learned distribution**

$$\tilde{\lambda}_k = \sigma_T^2 \left(\frac{\Phi(\sigma_0)}{\Phi(\sigma_T)} \right)^2 = \sigma_T^2 \frac{\sigma_0^2 + \tilde{\Sigma}_{kk}}{\sigma_T^2 + \tilde{\Sigma}_{kk}} \approx \tilde{\Sigma}_{kk}$$

1285 **The generated covariance will be**

$$\begin{aligned}\tilde{\Sigma} &= F \text{diag}(\tilde{\lambda}_k) F^* \\ &\approx F \text{diag}(\tilde{\Sigma}_{kk}) F^*\end{aligned}$$

1286 *Remark G.4.* • Basically it's the same story as the one-layer linear network, the major differ-
1287 ence is, instead of evolution along the eigenbasis of data covariance (PCs), the parametriza-
1288 tion of convolutional network enforces the learning dynamics to align with the Fourier basis
1289 regardless of the original PC of the data. It treats the data as if it's stationary / translation
1290 invariant.

1291 • The generated data will be independent along each Fourier mode, just like the fully connected
1292 case, the generated data is independent along each eigen-mode.

1293 • This amounts to decorrelate the original covariance in the spectral domain, making it
1294 translation invariance / circulant.

1295 **Generated distribution during sampling** Using the solution to gradient flow of **spectral**
1296 **parametrization**, for each Fourier mode,

$$\gamma_k(\tau; \sigma) = \gamma_k^* + (\gamma_k(0; \sigma) - \gamma_k^*) e^{-2\eta(\sigma^2 + \tilde{\Sigma}_{kk})\tau}$$

1297 We can solve the sampling ODE during training. Assume the initial value of each Fourier mode is the
1298 same across noise scale $\gamma_k(0; \sigma) = \gamma_k(0)$, $\forall \sigma$. We can write down the generated variance through
1299 training time as,²

$$\begin{aligned}\Phi_k(\sigma) &= \exp \left(\int_{\epsilon}^{\sigma} -\frac{\gamma_k(\lambda) - 1}{\lambda} d\lambda \right) \\ &= \exp \left(\int_{\epsilon}^{\sigma} -\frac{\frac{\tilde{\Sigma}_{kk}}{\lambda^2 + \tilde{\Sigma}_{kk}} + (\gamma_k(0) - \frac{\tilde{\Sigma}_{kk}}{\lambda^2 + \tilde{\Sigma}_{kk}}) e^{-2\eta(\lambda^2 + \tilde{\Sigma}_{kk})\tau} - 1}{\lambda} d\lambda \right) \\ &= \sqrt{\tilde{\Sigma}_{kk} + \sigma^2} \exp \left(\frac{1}{2} \left[(1 - \gamma_k(0)) \text{Ei}(-2\eta\tau\sigma^2) e^{-2\eta\tau\tilde{\Sigma}_{kk}} - \text{Ei}(-2\eta\tau(\sigma^2 + \tilde{\Sigma}_{kk})) \right] \right)\end{aligned}$$

1300 Using the solution to gradient flow of **filter weights parametrization**, we effectively amplify the
1301 learning rate $\eta \rightarrow N\eta$ in the expression.

$$\gamma_k(\tau; \sigma) = \gamma_k^* + (\gamma_k(0; \sigma) - \gamma_k^*) e^{-2N\eta(\sigma^2 + \tilde{\Sigma}_{kk})\tau}$$

²One thing to note is that it's highly likely that the initial value of $\gamma_k(0; \sigma)$ can differ for each Fourier mode, i.e. different Fourier modes are initialized at different weight due to the filter initialization.

1302 The generated distribution has variance $\check{\Sigma} = F\check{\Lambda}F^*$, where $\check{\Lambda}_{kk} = \sigma_T^2 \left(\frac{\Phi_k(\sigma_0)}{\Phi_k(\sigma_T)} \right)^2$ controls variance
 1303 along Fourier modes.

$$\Phi_k(\sigma) = \sqrt{\check{\Sigma}_{kk} + \sigma^2} \exp \left(\frac{1}{2} \left[(1 - \gamma_k(0)) \operatorname{Ei}(-2N\eta\tau\sigma^2) e^{-2N\eta\tau\check{\Sigma}_{kk}} - \operatorname{Ei}(-2N\eta\tau(\sigma^2 + \check{\Sigma}_{kk})) \right] \right)$$

1304 **G.4 Local patch linear convolutional network**

1305 Now, let's consider a local filter \mathbf{w} .

1306 **Spectral representation and Spatial locality constraint** The major difference is that if we have
 1307 locality constraint in \mathbf{w} the corresponding constraint in γ will be constrained in a lower dimensional
 1308 space

$$w_\ell = \frac{1}{N} \sum_{k=0}^{N-1} e^{2\pi i \frac{k\ell}{N}} \gamma_k$$

1309

$$\gamma = \sqrt{N} F \mathbf{w}$$

$$\gamma_l = \sum_{k=0}^{N-1} e^{-2\pi i \frac{k\ell}{N}} w_k$$

1310 But if our $w_k = 0, \forall k \geq K$ then the spectral moment just sum over K

$$\gamma_l = \sum_{k=0}^{K-1} e^{-2\pi i \frac{k\ell}{N}} w_k$$

1311 **Specific case: kernel 3 convolution** Consider the case where the kernel size is 3, so only
 1312 $w_1, w_0, w_{-1} \neq 0$ Then the spectrum can only have a DC and a cosine and sine component

$$\begin{aligned} \gamma_l &= \sum_{k \in \{-1, 0, 1\}} e^{-2\pi i \frac{k\ell}{N}} w_k \\ &= w_0 + w_1 e^{-2\pi i \frac{\ell}{N}} + w_{-1} e^{2\pi i \frac{\ell}{N}} \\ &= w_0 + (w_1 + w_{-1}) \cos\left(\frac{2\pi \ell}{N}\right) + i(w_{-1} - w_1) \sin\left(\frac{2\pi \ell}{N}\right) \end{aligned}$$

1313 Basically, the small kernel size mandates that the spectral view of it γ cannot vary very fast! Thus the
 1314 spectral representation have to be very smooth, just sine and cosine waves.

1315 **G.4.1 Training dynamics of patch linear convolutional net**

1316 To study the training dynamics of the linear local convolutional network, it's easier to directly work
 1317 with the entries of the filters.

1318 Consider the Toeplitz weight matrix (general boundary condition),

$$\mathbf{W} = \begin{pmatrix} w_0 & w_1 & w_2 & \cdots & w_{d-1} \\ w_{-1} & w_0 & w_1 & \cdots & w_{d-2} \\ w_{-2} & w_{-1} & w_0 & \cdots & w_{d-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_{-d+1} & w_{-d+2} & w_{-d+3} & \cdots & w_0 \end{pmatrix}.$$

1319 **Shift matrix formulation** Generally, Toeplitz matrix can be written as

$$\mathbf{W} = \sum_{k=0}^{d-1} w_k P^k + \sum_{k=1}^{d-1} w_{-k} (P^T)^k$$

1320 where the (non-circulant) shift matrix is P ,

$$P = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}.$$

1321 This compactly express the weight matrix as a weighted sum of shift matrix power, enable us to write
 1322 down the gradient efficiently.

1323 **Remark:**

- 1324 • For circulant shift matrix $P^T = P^{-1}$, $P^T P = I$ exactly.
- 1325 • For non circulant shift matrix, approximately $P^T = P^{-1}$ but there will be some issue in the
- 1326 boundary condition.
- 1327 • Trace with it also allow us to elegantly extract entries from a matrix.
- 1328 • Multiplying this shift matrix allows us to shift the matrix

1329 This decomposition of weights allows us to write down the loss and the gradient

1330 For $k \geq 0$

$$\begin{aligned}
 \frac{\partial \mathcal{L}}{\partial w_k} &= \sum_{j-i=k} G_{ij} \\
 &= \sum_{j-i=k} \frac{\partial \mathcal{L}}{\partial \mathbf{W}_{ij}} \\
 &= \langle P^k, G \rangle \\
 &= \text{Tr}[(P^k)^T G] \\
 &= \text{Tr}[(P^k)^T \frac{\partial \mathcal{L}}{\partial \mathbf{W}}]
 \end{aligned}$$

1331 for the other k ,

$$\begin{aligned}
 \frac{\partial \mathcal{L}}{\partial w_{-k}} &= \langle (P^T)^k, G \rangle \\
 &= \text{Tr}[P^k G] \\
 &= \text{Tr}[P^k \frac{\partial \mathcal{L}}{\partial \mathbf{W}}]
 \end{aligned}$$

1332 Using this formulation, we can express the gradient to each.

$$\begin{aligned}
 \frac{\partial \mathcal{L}}{\partial w_k} &= 2\text{Tr}[-(P^k)^T \Sigma + (P^k)^T W(\sigma^2 I + \Sigma)] \\
 &= -2\text{Tr}[(P^k)^T \Sigma] + 2\text{Tr}[(P^k)^T W(\sigma^2 I + \Sigma)] \\
 &= -2\text{Tr}[(P^k)^T \Sigma] + 2\text{Tr}[(P^k)^T \left(\sum_{m=0}^{d-1} w_m P^m + \sum_{m=1}^{d-1} w_{-m} (P^T)^m \right) (\sigma^2 I + \Sigma)] \\
 &= -2\text{Tr}[(P^k)^T \Sigma] + 2\text{Tr} \left[\left(\sum_{m=0}^{d-1} w_m (P^k)^T P^m + \sum_{m=1}^{d-1} w_{-m} (P^T)^{k+m} \right) (\sigma^2 I + \Sigma) \right] \\
 &= -2\text{Tr}[(P^k)^T \Sigma] + 2 \sum_{m=0}^{d-1} w_m \text{Tr}[(P^k)^T P^m (\sigma^2 I + \Sigma)] + 2 \sum_{m=1}^{d-1} w_{-m} \text{Tr}[(P^T)^{k+m} (\sigma^2 I + \Sigma)]
 \end{aligned}$$

1333 Similarly,

$$\begin{aligned}
 \frac{\partial \mathcal{L}}{\partial w_{-k}} &= 2\text{Tr}[-P^k \Sigma + P^k W(\sigma^2 I + \Sigma)] \\
 &= -2\text{Tr}[P^k \Sigma] + 2\text{Tr}[P^k W(\sigma^2 I + \Sigma)] \\
 &= -2\text{Tr}[P^k \Sigma] + 2\text{Tr}[P^k \left(\sum_{m=0}^{d-1} w_m P^m + \sum_{m=1}^{d-1} w_{-m} (P^T)^m \right) (\sigma^2 I + \Sigma)] \\
 &= -2\text{Tr}[P^k \Sigma] + 2\text{Tr} \left[\left(\sum_{m=0}^{d-1} w_m P^{k+m} + \sum_{m=1}^{d-1} w_{-m} P^k (P^T)^m \right) (\sigma^2 I + \Sigma) \right] \\
 &= -2\text{Tr}[P^k \Sigma] + 2 \sum_{m=0}^{d-1} w_m \text{Tr}[P^{k+m} (\sigma^2 I + \Sigma)] + 2 \sum_{m=1}^{d-1} w_{-m} \text{Tr}[P^k (P^T)^m (\sigma^2 I + \Sigma)]
 \end{aligned}$$

1334 We denote the general k-shift operator (non circulant boundary condition)

$$S(k) = \begin{cases} P^k, & k \geq 0, \\ (P^T)^{-k}, & k < 0, \end{cases}$$

1335

$$S(k)^T = (P^k)^T = (P^T)^k = S(-k)$$

1336 Define a special function with two indices

$$\begin{aligned} \mathcal{T}[k, m] &= \text{Tr}[S(m)(\sigma^2 I + \Sigma)S(k)^T] \\ &= \sigma^2 \text{Tr}[S(m)S(k)^T] + \text{Tr}[S(m) \Sigma S(k)^T] \end{aligned}$$

1337 The first term is diagonal, the 2nd term is not. non diagonal terms all come from the 2nd term.

1338 Note that this special function is symmetric, which is understandable, since it will be functioning as
1339 the new covariance matrix.

$$\begin{aligned} \mathcal{T}[m, k] &= \text{Tr}[S(m)(\sigma^2 I + \Sigma)S(k)^T] \\ &= \text{Tr}[S(k)(\sigma^2 I + \Sigma)S(m)^T] \\ &= \mathcal{T}[k, m] \end{aligned}$$

1340 Then the gradient could be written as this simplified equation, of a similar structure as before.

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial w_k} &= -2\text{Tr}[S(k)^T \Sigma] + 2 \sum_{m=1-d}^{d-1} \text{Tr}[S(m)(\sigma^2 I + \Sigma)S(k)^T] w_m \\ &= -2\text{Tr}[S(k)^T \Sigma] + 2 \sum_{m=1-d}^{d-1} \mathcal{T}[k, m] w_m \\ &= -2\text{Tr}[S(k) \Sigma] + 2 \sum_{m=1-d}^{d-1} \mathcal{T}[k, m] w_m \end{aligned}$$

1341 Thus to study the learning dynamics of convolutional linear model, the object that need to be focus on
1342 is $\mathcal{T}[k, m]$, the spectrum of it. Thus, the learning dynamics of w will be first along higher eigenvalues
1343 of \mathcal{T} , and then lower eigen ones.

1344 **Interpretation of T matrix** Note that the $\mathcal{T}[k, m]$ matrix can be regarded as the spatially averaged
1345 version of covariance matrix, esp. by averaging local cross covariances of pixels at distance k, m .

1346 **Optimal solution** The fixed point of the equation is the following linear equation

$$\sum_m \mathcal{T}[k, m] w_m = \text{Tr}[S(k) \Sigma]$$

1347 Let the vector be $R_k = \text{Tr}[S(k) \Sigma]$, $k = \{1-d, \dots, d-1\}$

$$w^* = \mathcal{T}^{-1} R$$

1348 Here we successfully derived the matrix equation for gradient flow for weights w .

1349 **Cyclic case (circular convolution)** Now consider the cyclic shift matrix

$$P = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 1 & 0 & 0 & \cdots & 0 \end{bmatrix}.$$

1350 then $P^T = P^{-1}$ then we have

$$\begin{aligned}\mathcal{T}[k, m] &= \text{Tr}[S(m)(\sigma^2 I + \Sigma)S(k)^T] \\ &= \text{Tr}[P^m(\sigma^2 I + \Sigma)P^{-k}] \\ &= \text{Tr}[P^{m-k}(\sigma^2 I + \Sigma)] \\ &= \sigma^2 \text{Tr}[P^{m-k}] + \text{Tr}[P^{m-k}\Sigma]\end{aligned}$$

1351 Note that

$$\text{Tr}[P^{m-k}] = N\delta_{mk}$$

1352 So

$$\begin{aligned}\mathcal{T}[k, m] &= N\sigma^2\delta_{mk} + \text{Tr}[P^{m-k}\Sigma] \\ &= N\left(\sigma^2\delta_{mk} + \frac{1}{N}\text{Tr}[P^{m-k}\Sigma]\right)\end{aligned}$$

1353 Note that the 2nd term is the shift average version of the covariance matrix

$$\begin{aligned}&\frac{1}{N} \text{Tr}[P^{m-k}\Sigma] \\ &= \frac{1}{N} \sum_{i=1}^N \Sigma_{i, i+(m-k) \pmod{N}}\end{aligned}$$

1354 Let's define a new averaged cross covariance vector

$$\begin{aligned}\chi[k] &= \frac{1}{N} \text{Tr}[P^k \Sigma] \\ &= \frac{1}{N} \sum_{i=1}^N \Sigma_{i, i+k \pmod{N}}\end{aligned}$$

1355 Note $\chi[k] = \chi[-k]$

$$\begin{aligned}R_k &= N\chi[k] \\ \mathcal{T}[k, m] &= N(\sigma^2\delta_{mk} + \chi[m-k])\end{aligned}$$

1356 We want to solve linear system

$$\sum_m \mathcal{T}[k, m]w_m - R_k = 0$$

1357 Note that $\mathcal{T}[k, m]$ is a Toeplitz matrix.

1358 In the circulant case the optimal solution satisfy,

$$\sum_{m=-K}^K (N\sigma^2\delta_{km} + R_{k-m})w_m^* = R_k$$

1359 where R_k is the averaged version of covariance matrix. So again it's a circulant regression with a
1360 Ridge like penalty.

$$R_k = \text{Tr}[P^k \Sigma]$$

1361 If we let $\check{\check{\Sigma}}_{k,m} = \frac{1}{N}R_{k-m} \in \mathbb{R}^{2K+1 \times 2K+1}$ be the Toeplitz patch covariance matrix, then we can
1362 write R_k as the center column of it

$$\sum_{m=-K}^K (N\sigma^2\delta_{km} + N\check{\check{\Sigma}}_{k,m})w_m^* = N\check{\check{\Sigma}}_{k,0}$$

1363 The solution can be written in vector form as

$$\begin{aligned}(N\sigma^2 I + N\check{\check{\Sigma}})w^* &= N\check{\check{\Sigma}}_{:,0} \\ w^* &= (N\sigma^2 I + N\check{\check{\Sigma}})^{-1} N\check{\check{\Sigma}}_{:,0} \\ &= \left((\sigma^2 I + \check{\check{\Sigma}})^{-1} \check{\check{\Sigma}} \right)_{:,0}\end{aligned}$$

1364 Thus the denoiser equals the central column from the Gaussian solution to the patch covariance.

1365 The gradient flow dynamics is

$$\frac{\partial \mathcal{L}}{\partial w_k} = -2R_k + 2 \sum_{m=1-d}^{d-1} (N\sigma^2\delta_{km} + R_{k-m})w_m$$

1366 We can write the flow dynamics in vector form

$$\begin{aligned} \frac{d\mathbf{w}}{d\tau} &= -\eta \frac{\partial \mathcal{L}}{\partial \mathbf{w}} \\ &= 2\eta (N\check{\Sigma}_{:,0} - N(\sigma^2 I + \check{\Sigma})\mathbf{w}) \\ &= 2\eta N (\check{\Sigma}_{:,0} - (\sigma^2 I + \check{\Sigma})\mathbf{w}) \end{aligned}$$

1367 Solution will be

$$\mathbf{w} = \mathbf{w}^* + \exp(-2\eta N(\sigma^2 I + \check{\Sigma})) (\mathbf{w}(0) - \mathbf{w}^*)$$

1368 with $\mathbf{w}^* = \left((\sigma^2 I + \check{\Sigma})^{-1} \check{\Sigma}_{:,0} \right)$.

1369 G.4.2 Sampling dynamics of patch linear convolutional net

1370 Consider a kernel of width $2K + 1$,

$$\begin{aligned} \gamma_l &= \sum_{k \in -K:K} e^{-2\pi i \frac{kl}{N}} w_k \\ &= w_0 + \sum_{k=1}^K w_k e^{-2\pi i \frac{kl}{N}} + w_{-k} e^{2\pi i \frac{kl}{N}} \\ &= w_0 + \sum_{k=1}^K (w_k + w_{-k}) \cos\left(\frac{2\pi kl}{N}\right) + i(w_{-k} - w_k) \sin\left(\frac{2\pi kl}{N}\right) \end{aligned}$$

1371 For symmetric filter weights $w_k = w_{-k}$, we have

$$\begin{aligned} \gamma_l &= w_0 + \sum_{k=1}^K (w_k + w_{-k}) \cos\left(\frac{2\pi kl}{N}\right) \\ &= w_0 + \sum_{k=1}^K w_k \cos\left(\frac{2\pi kl}{N}\right) \end{aligned}$$

1372 During sampling we have

$$\frac{d}{d\sigma} c_k = -\frac{\gamma_k(\sigma) - 1}{\sigma} c_k - b_k(\sigma)$$

1373 The key integral governing the variance is

$$\begin{aligned} \Phi_k(\sigma) &= \exp\left(\int^\sigma -\frac{\gamma_k(\lambda) - 1}{\lambda} d\lambda\right) \\ &= \exp\left(\int^\sigma -\frac{w_0(\lambda) + \sum_{l=1}^K w_l(\lambda) \cos\left(\frac{2\pi lk}{N}\right) - 1}{\lambda} d\lambda\right) \end{aligned}$$

1374 Thus we can see it's integrating these sinusoidal modulations in the frequency domain.

1375 G.5 Appendix: Useful math

Discrete Fourier Transformation (DFT) matrix

$$F_{jk} = \frac{1}{\sqrt{N}} e^{-2\pi i jk/N}$$

1376 Property

1377 Symmetry

$$F = F^T$$

1378 Conjugacy

1379

$$FF^* = I$$

$$\begin{aligned} F_{jk}F_{km}^* &= \frac{1}{N} \sum_k^N e^{-2\pi i jk/N} e^{+2\pi i km/N} \\ &= \frac{1}{N} \sum_k^N e^{-2\pi i \frac{k(j-m)}{N}} \\ &= \delta_{jm} \end{aligned}$$

1380 **Properties of Circulant and DFT** Consider matrices of the following form

$$M = F\Lambda F^*$$

1381 Explicitly, the matrix reads

$$\begin{aligned} M_{jk} &= \sum_m F_{jm} \lambda_m F_{mk}^* \\ &= \sum_m \lambda_m \frac{1}{\sqrt{N}} \exp\left(-2\pi i \frac{jm}{N}\right) \frac{1}{\sqrt{N}} \exp\left(2\pi i \frac{mk}{N}\right) \\ &= \frac{1}{N} \sum_m \lambda_m \exp\left(-2\pi i \frac{jm}{N} + 2\pi i \frac{mk}{N}\right) \\ &= \frac{1}{N} \sum_m \lambda_m \exp\left(-2\pi i \frac{m(j-k)}{N}\right) \end{aligned}$$

1382 Thus we see M_{jk} just depends on $j - k$ thus circulant. Let's define the circulant coefficients

1383 $c_{j-k} := M_{j,k}$.

1384 Then we have for $\Delta = 0, 1, \dots, N-1$

$$c_\Delta = \frac{1}{N} \sum_m \lambda_m \exp\left(-2\pi i \frac{m\Delta}{N}\right)$$

1385 The special case is the “DC”-non-ocillating term, which are the diagonal values in M

$$c_0 = \frac{1}{N} \sum_m \lambda_m$$

1386 the adjacent subdiagonal values in M are

$$\begin{aligned} c_1 &= \frac{1}{N} \sum_m \lambda_m \exp\left(-2\pi i \frac{m}{N}\right) \\ c_{N-1} &= \frac{1}{N} \sum_m \lambda_m \exp\left(-2\pi i \frac{m(N-1)}{N}\right) \\ &= \frac{1}{N} \sum_m \lambda_m \exp\left(+2\pi i \frac{m}{N}\right) \end{aligned}$$

1387 More generally $c_k = c_{N-k}^*$ are complex conjugate, or equal in the real case.

1388 **Properties of the Λ spectrum** Since Σ is real symmetric, the eigenvalues exhibit mirror, i.e. even
1389 symmetry $\lambda_k = \lambda_{N-k}$ for $k = 1, 2, \dots, N-1$
1390 There are one or two standalone eigenvalues, when N is odd, λ_0 is the zero-th frequency, DC
1391 component, which is unpaired.
1392 When N is even, λ_0 (DC component) and $\lambda_{N/2}$ (Nyquist frequency) are both standing alone, which
1393 are both unpaired.

1394 H Detailed derivation of Flow Matching model

1395 Consider the objective of flow matching [28], at a certain t

$$\mathcal{L} = \mathbb{E}_{\mathbf{x}_0 \sim \mathcal{N}(0, \mathbf{I}), \mathbf{x}_1 \sim p_1} \|u(\mathbf{x}_t; t) - (\mathbf{x}_1 - \mathbf{x}_0)\|^2 \quad (295)$$

$$= \mathbb{E}_{\mathbf{x}_0 \sim \mathcal{N}(0, \mathbf{I}), \mathbf{x}_1 \sim p_1} \|u((1-t)\mathbf{x}_0 + t\mathbf{x}_1; t) - (\mathbf{x}_1 - \mathbf{x}_0)\|^2 \quad (296)$$

$$\mathbf{x}_t = (1-t)\mathbf{x}_0 + t\mathbf{x}_1 \quad (297)$$

1396 Given linear function approximator of the velocity field,

$$u(\mathbf{x}; t) = \mathbf{W}_t \mathbf{x} + \mathbf{b}_t \quad (298)$$

1397

$$\mathcal{L} = \mathbb{E}_{\mathbf{x}_0 \sim \mathcal{N}(0, \mathbf{I}), \mathbf{x}_1 \sim p_1} \|\mathbf{W}_t((1-t)\mathbf{x}_0 + t\mathbf{x}_1) + \mathbf{b}_t - (\mathbf{x}_1 - \mathbf{x}_0)\|^2 \quad (299)$$

$$= \mathbb{E}_{\mathbf{x}_0 \sim \mathcal{N}(0, \mathbf{I}), \mathbf{x}_1 \sim p_1} (\mathbf{W}_t((1-t)\mathbf{x}_0 + t\mathbf{x}_1) + \mathbf{b}_t - (\mathbf{x}_1 - \mathbf{x}_0))^T (\mathbf{W}_t((1-t)\mathbf{x}_0 + t\mathbf{x}_1) + \mathbf{b}_t - (\mathbf{x}_1 - \mathbf{x}_0)) \quad (300)$$

$$= \mathbb{E}_{\mathbf{x}_0 \sim \mathcal{N}(0, \mathbf{I}), \mathbf{x}_1 \sim p_1} \text{Tr} \left[((1-t)\mathbf{x}_0 + t\mathbf{x}_1)^T \mathbf{W}_t^T \mathbf{W}_t ((1-t)\mathbf{x}_0 + t\mathbf{x}_1) + \mathbf{b}_t^T \mathbf{b}_t + (\mathbf{x}_1 - \mathbf{x}_0)^T (\mathbf{x}_1 - \mathbf{x}_0) \right. \quad (301)$$

$$\left. - 2(\mathbf{x}_1 - \mathbf{x}_0)^T \mathbf{b}_t - 2(\mathbf{x}_1 - \mathbf{x}_0)^T \mathbf{W}_t((1-t)\mathbf{x}_0 + t\mathbf{x}_1) + 2\mathbf{b}_t^T \mathbf{W}_t((1-t)\mathbf{x}_0 + t\mathbf{x}_1) \right] \quad (302)$$

$$= \mathbb{E}_{\mathbf{x}_0 \sim \mathcal{N}(0, \mathbf{I}), \mathbf{x}_1 \sim p_1} \text{Tr} \left[\mathbf{W}_t^T \mathbf{W}_t ((1-t)\mathbf{x}_0 + t\mathbf{x}_1)((1-t)\mathbf{x}_0 + t\mathbf{x}_1)^T + \mathbf{b}_t^T \mathbf{b}_t + (\mathbf{x}_1 - \mathbf{x}_0)^T (\mathbf{x}_1 - \mathbf{x}_0) \right. \quad (303)$$

$$\left. - 2(\mathbf{x}_1 - \mathbf{x}_0)^T \mathbf{b}_t - 2\mathbf{W}_t((1-t)\mathbf{x}_0 + t\mathbf{x}_1)(\mathbf{x}_1 - \mathbf{x}_0)^T + 2\mathbf{b}_t^T \mathbf{W}_t((1-t)\mathbf{x}_0 + t\mathbf{x}_1) \right] \quad (304)$$

1398 Similar to diffusion case, it will also only depend on mean and covariance of p_1 .

$$\mathbb{E}_{\mathbf{x}_0, \mathbf{x}_1} [(1-t)\mathbf{x}_0 + t\mathbf{x}_1] = t\boldsymbol{\mu} \quad (305)$$

$$\mathbb{E}_{\mathbf{x}_0, \mathbf{x}_1} [\mathbf{x}_1 - \mathbf{x}_0] = \boldsymbol{\mu} \quad (306)$$

$$\mathbb{E}_{\mathbf{x}_0, \mathbf{x}_1} [(\mathbf{x}_1 - \mathbf{x}_0)^T (\mathbf{x}_1 - \mathbf{x}_0)] = \mathbb{E}_{\mathbf{x}_0, \mathbf{x}_1} [\mathbf{x}_1^T \mathbf{x}_1 - 2\mathbf{x}_1^T \mathbf{x}_0 + \mathbf{x}_0^T \mathbf{x}_0] \quad (307)$$

$$= \text{Tr}[\boldsymbol{\Sigma} + \boldsymbol{\mu}\boldsymbol{\mu}^T + \mathbf{I}] \quad (308)$$

$$\mathbb{E}_{\mathbf{x}_0, \mathbf{x}_1} [((1-t)\mathbf{x}_0 + t\mathbf{x}_1)(\mathbf{x}_1 - \mathbf{x}_0)^T] = t(\boldsymbol{\Sigma} + \boldsymbol{\mu}\boldsymbol{\mu}^T) - (1-t)\mathbf{I} \quad (309)$$

$$\mathbb{E}_{\mathbf{x}_0, \mathbf{x}_1} [((1-t)\mathbf{x}_0 + t\mathbf{x}_1)((1-t)\mathbf{x}_0 + t\mathbf{x}_1)^T] = t^2(\boldsymbol{\Sigma} + \boldsymbol{\mu}\boldsymbol{\mu}^T) + (1-t)^2\mathbf{I} \quad (310)$$

1399 Taking full expectation, the average loss reads.

$$\mathcal{L} = \text{Tr} \left[\mathbf{W}_t^T \mathbf{W}_t (t^2(\boldsymbol{\Sigma} + \boldsymbol{\mu}\boldsymbol{\mu}^T) + (1-t)^2\mathbf{I}) + \mathbf{b}_t^T \mathbf{b}_t + (\boldsymbol{\Sigma} + \boldsymbol{\mu}\boldsymbol{\mu}^T + \mathbf{I}) \right. \quad (311)$$

$$\left. - 2\boldsymbol{\mu}^T \mathbf{b}_t - 2\mathbf{W}_t(t(\boldsymbol{\Sigma} + \boldsymbol{\mu}\boldsymbol{\mu}^T) - (1-t)\mathbf{I}) + 2t\mathbf{b}_t^T \mathbf{W}_t \boldsymbol{\mu} \right] \quad (312)$$

1400 The gradient to parameters are

$$\nabla_{\mathbf{b}} \mathcal{L} = 2[\mathbf{b} - \boldsymbol{\mu} + t\mathbf{W}\boldsymbol{\mu}] \quad (313)$$

$$\nabla_{\mathbf{W}} \mathcal{L} = 2 \left[\mathbf{W} (t^2(\boldsymbol{\Sigma} + \boldsymbol{\mu}\boldsymbol{\mu}^T) + (1-t)^2\mathbf{I}) - (t(\boldsymbol{\Sigma} + \boldsymbol{\mu}\boldsymbol{\mu}^T) - (1-t)\mathbf{I}) + t\mathbf{b}\boldsymbol{\mu}^T \right], \quad (314)$$

1401 **Simplifying case** $\boldsymbol{\mu} = 0$ Note special case $\boldsymbol{\mu} = 0$

$$\nabla_{\mathbf{b}} \mathcal{L} = 2\mathbf{b} \quad (315)$$

$$\nabla_{\mathbf{W}} \mathcal{L} = 2 \left[\mathbf{W} (t^2\boldsymbol{\Sigma} + (1-t)^2\mathbf{I}) - (t\boldsymbol{\Sigma} - (1-t)\mathbf{I}) \right], \quad (316)$$

1402 **Optimal solution** Optimal solution to the full case

$$\mathbf{b}^* = \mu - t\mathbf{W}^*\mu \quad (317)$$

$$\mathbf{W}^* = (t\mathbf{\Sigma} - (1-t)\mathbf{I}) (t^2\mathbf{\Sigma} + (1-t)^2\mathbf{I})^{-1} \quad (318)$$

1403 We can represent it on the eigenbasis of $\mathbf{\Sigma}$, $[u_1, \dots, u_d]$

$$\mathbf{W}^* = \sum_k \frac{t\lambda_k - (1-t)}{t^2\lambda_k + (1-t)^2} \mathbf{u}_k \mathbf{u}_k^T$$

1404 **Asymptotics** Consider the limit $t \rightarrow 0$,

$$\mathbf{W}_{t \rightarrow 0}^* = -\mathbf{I} \quad (319)$$

$$\mathbf{b}_{t \rightarrow 0}^* = \mu \quad (320)$$

1405 Consider the limit $t \rightarrow 1$,

$$\mathbf{W}_{t \rightarrow 1}^* = \mathbf{I} \quad (321)$$

$$\mathbf{b}_{t \rightarrow 1}^* = \mu - \mathbf{W}_{t \rightarrow 1}^*\mu = 0 \quad (322)$$

1406 **H.1 Solution to the flow matching sampling ODE with optimal solution**

1407 Solving the sampling ODE of flow matching integrating from 0 to 1, with the linear vector field

$$\frac{d\mathbf{x}}{dt} = u(\mathbf{x}; t)$$

1408 Under the linear solution case

$$\frac{d\mathbf{x}}{dt} = \mathbf{W}_t^* \mathbf{x} + \mathbf{b}_t^*$$

Simplified zero mean case $\mu = 0$

$$\frac{d\mathbf{x}}{dt} = \sum_k \frac{t\lambda_k - (1-t)}{t^2\lambda_k + (1-t)^2} \mathbf{u}_k \mathbf{u}_k^T \mathbf{x}$$

1409 Solving the flow-matching sampling ODE mode by mode

$$\mathbf{u}_k^T \frac{d\mathbf{x}}{dt} = \frac{t\lambda_k - (1-t)}{t^2\lambda_k + (1-t)^2} \mathbf{u}_k^T \mathbf{x} \quad (323)$$

$$\frac{dc_k(t)}{dt} = \frac{t\lambda_k - (1-t)}{t^2\lambda_k + (1-t)^2} c_k(t) \quad (324)$$

$$\ln c_k(t) = \frac{1}{2} \ln |t^2\lambda_k + (1-t)^2| + C.$$

$$c_k(t) = C \sqrt{t^2\lambda_k + (1-t)^2}$$

$$\frac{c_k(t)}{c_k(0)} = \sqrt{t^2\lambda_k + (1-t)^2}$$

$$\frac{c_k(1)}{c_k(0)} = \sqrt{\lambda_k}$$

1410 This is the correct scaling of the \mathbf{x} , the sampling trajectory of \mathbf{x}_t reads

$$\mathbf{x}_t = \sum_k c_k(t) \mathbf{u}_k \quad (325)$$

$$= \left(\sum_k \sqrt{t^2\lambda_k + (1-t)^2} \mathbf{u}_k \mathbf{u}_k^T \right) \mathbf{x}_0 \quad (326)$$

1411 Thus, at time t the covariance of the sampled points is

$$\mathbb{E}[\mathbf{x}_t \mathbf{x}_t^T] = \sum_k (t^2 \lambda_k + (1-t)^2) \mathbf{u}_k \mathbf{u}_k^T$$

1412

$$\tilde{\lambda}_k = t^2 \lambda_k + (1-t)^2$$

Full case $\mu \neq 0$

$$\frac{d\mathbf{x}}{dt} = \mathbf{W}_t^* \mathbf{x} + \mu - t \mathbf{W}_t^* \mu \quad (327)$$

$$\frac{d\mathbf{x}}{dt} = \mu + \sum_k \frac{t\lambda_k - (1-t)}{t^2\lambda_k + (1-t)^2} \mathbf{u}_k \mathbf{u}_k^T (\mathbf{x} - t\mu) \quad (328)$$

1413 It can also be solved mode by mode. Redefine variable $\mathbf{y}_t = \mathbf{x}_t - t\mu$

$$\frac{d\mathbf{y}_t}{dt} = \frac{d\mathbf{x}_t}{dt} - \mu \quad (329)$$

$$= \sum_k \frac{t\lambda_k - (1-t)}{t^2\lambda_k + (1-t)^2} \mathbf{u}_k \mathbf{u}_k^T \mathbf{y}_t \quad (330)$$

1414 Then each mode can be solved accordingly, $c_k(t) = \mathbf{u}_k^T \mathbf{y}_t$

$$\frac{dc_k(t)}{dt} = \frac{t\lambda_k - (1-t)}{t^2\lambda_k + (1-t)^2} c_k(t)$$

1415 Using the same solution as above, we get the full solution for sampling equation with any μ

$$\mathbf{x}_t = t\mu + \sum_k c_k(t) \mathbf{u}_k \quad (331)$$

$$= t\mu + \sum_k \sqrt{t^2\lambda_k + (1-t)^2} \mathbf{u}_k \mathbf{u}_k^T \mathbf{x}_0 \quad (332)$$

1416 H.2 Learning dynamics of flow matching objective (single layer)

1417 **Simplifying case $\mu = 0$, single layer network** Note the special case $\mu = 0$

$$\nabla_{\mathbf{b}} \mathcal{L} = 2\mathbf{b} \quad (333)$$

$$\nabla_{\mathbf{W}} \mathcal{L} = 2 \left[\mathbf{W} (t^2 \Sigma + (1-t)^2 \mathbf{I}) - (t \Sigma - (1-t) \mathbf{I}) \right] \quad (334)$$

1418

$$\frac{d\mathbf{W}}{d\tau} = -\eta \nabla_{\mathbf{W}} \mathcal{L} \quad (335)$$

$$\frac{d\mathbf{W}}{d\tau} = -2\eta \left[\mathbf{W} (t^2 \Sigma + (1-t)^2 \mathbf{I}) - (t \Sigma - (1-t) \mathbf{I}) \right] \quad (336)$$

1419 Use the eigenbases projection

$$\frac{d\mathbf{W} \mathbf{u}_k}{d\tau} = -2\eta \left[\mathbf{W} (t^2 \Sigma + (1-t)^2 \mathbf{I}) - (t \Sigma - (1-t) \mathbf{I}) \right] \mathbf{u}_k \quad (337)$$

$$= -2\eta \left[\mathbf{W} \mathbf{u}_k (t^2 \lambda_k + (1-t)^2) - (t \lambda_k - (1-t)) \mathbf{u}_k \right] \quad (338)$$

$$= -2\eta (t^2 \lambda_k + (1-t)^2) \left[\mathbf{W} \mathbf{u}_k - \frac{t \lambda_k - (1-t)}{t^2 \lambda_k + (1-t)^2} \mathbf{u}_k \right] \quad (339)$$

$$\mathbf{W}(\tau) \mathbf{u}_k - \frac{t \lambda_k - (1-t)}{t^2 \lambda_k + (1-t)^2} \mathbf{u}_k = A \exp \left(-2\eta \tau (t^2 \lambda_k + (1-t)^2) \right)$$

$$\mathbf{W}(\tau)\mathbf{u}_k = \frac{t\lambda_k - (1-t)}{t^2\lambda_k + (1-t)^2}\mathbf{u}_k + \left(\mathbf{W}(0)\mathbf{u}_k - \frac{t\lambda_k - (1-t)}{t^2\lambda_k + (1-t)^2}\mathbf{u}_k\right) \exp\left(-2\eta\tau(t^2\lambda_k + (1-t)^2)\right)$$

1420 The full solutions of the weight and bias are

$$\begin{aligned} \mathbf{W}(\tau) &= \mathbf{W}^* + \sum_k \left(\mathbf{W}(0)\mathbf{u}_k - \frac{t\lambda_k - (1-t)}{t^2\lambda_k + (1-t)^2}\mathbf{u}_k\right) \mathbf{u}_k^T \exp\left(-2\eta\tau(t^2\lambda_k + (1-t)^2)\right) \\ \mathbf{b}(\tau) &= \mathbf{b}(0) \exp(-2\eta\tau) \end{aligned}$$

1422 It's easy to see this solution has similar structure with that of denoising score matching objective for
1423 diffusion model.

1424 Remarks

- 1425 • The learning dynamics of weight eigenmodes at different time t were visualized in Fig. 7A.
- 1426 • Note that, the convergence speed of each mode is $\exp(-2\eta\tau(t^2\lambda_k + (1-t)^2))$. Similarly,
1427 at the same time t , the higher the data variance λ_k the faster the convergence speed.
- 1428 • At smaller t , all eigenmodes converge at similar speed
- 1429 • At larger $t \sim 1$, the eigenmodes are resolved at distinct speed depend on the eigenvalues.
- 1430 • Note, for each eigenvalue λ_k there is a special time point where the convergence speed is
1431 maximized $t^* = 1/(\lambda_k + 1)$.

1432 H.2.1 Interaction of weight learning and flow sampling

1433 Consider the sampling dynamics of flow matching model,

$$\frac{d\mathbf{x}}{dt} = \mathbf{W}(\tau, t)\mathbf{x} + \mathbf{b}(\tau, t)$$

1434 Assume the weight initialization is aligned, and the same across t

$$\mathbf{W}(0, t) = \sum_k Q_k \mathbf{u}_k \mathbf{u}_k^T$$

1435 then

$$\begin{aligned} \mathbf{W}(\tau, t) &= \mathbf{W}^* + \sum_k \left(\mathbf{W}(0) - \mathbf{W}^*\right) \mathbf{u}_k \mathbf{u}_k^T \exp\left(-2\eta\tau(t^2\lambda_k + (1-t)^2)\right) \\ &= \sum_k \frac{t\lambda_k - (1-t)}{t^2\lambda_k + (1-t)^2} \mathbf{u}_k \mathbf{u}_k^T + \sum_k \left(Q_k - \frac{t\lambda_k - (1-t)}{t^2\lambda_k + (1-t)^2}\right) \mathbf{u}_k \mathbf{u}_k^T \exp\left(-2\eta\tau(t^2\lambda_k + (1-t)^2)\right) \end{aligned} \quad (340)$$

(341)

1436 Ignoring the bias part, consider the weight integration along $c_k(t) = \mathbf{u}_k^T \mathbf{x}(t)$

$$\frac{d}{dt}c_k(t) = \left[\frac{t\lambda_k - (1-t)}{t^2\lambda_k + (1-t)^2} + \left(Q_k - \frac{t\lambda_k - (1-t)}{t^2\lambda_k + (1-t)^2}\right) \exp\left(-2\eta\tau(t^2\lambda_k + (1-t)^2)\right)\right]c_k(t)$$

1437 Integration of the coefficient yields

$$I = \int_0^1 dt \left[\frac{t\lambda_k - (1-t)}{t^2\lambda_k + (1-t)^2} + \left(Q_k - \frac{t\lambda_k - (1-t)}{t^2\lambda_k + (1-t)^2}\right) \exp\left(-2\eta\tau(t^2\lambda_k + (1-t)^2)\right) \right] \quad (342)$$

$$\begin{aligned} &= \frac{\sqrt{2\pi}Q_k e^{-\frac{2\eta\tau\lambda_k}{\lambda_k+1}} \left(\operatorname{erf}\left(\sqrt{2}\sqrt{\frac{\eta\tau}{\lambda_k+1}}\right) + \operatorname{erf}\left(\sqrt{2}\lambda_k\sqrt{\frac{\eta\tau}{\lambda_k+1}}\right) \right)}{4\sqrt{\eta\tau(\lambda_k+1)}} + \\ &\quad \frac{1}{2} (\operatorname{Ei}(-2\eta\tau) - \operatorname{Ei}(-2\eta\tau\lambda_k) + \log(\lambda_k)) \end{aligned} \quad (343)$$

$$\begin{aligned} &= \frac{1}{2} \log(\lambda_k) + \frac{1}{2} (\operatorname{Ei}(-2\eta\tau) - \operatorname{Ei}(-2\eta\tau\lambda_k)) + \frac{1}{2} \sqrt{\frac{\pi}{2\eta\tau(\lambda_k+1)}} Q_k e^{-\frac{2\eta\tau\lambda_k}{\lambda_k+1}} \\ &\quad \left(\operatorname{erf}\left(\sqrt{\frac{2\eta\tau}{\lambda_k+1}}\right) + \operatorname{erf}\left(\lambda_k\sqrt{\frac{2\eta\tau}{\lambda_k+1}}\right) \right) \end{aligned} \quad (344)$$

$$A_k(1; 0) \quad (345)$$

$$= \Phi_k(1) \quad (346)$$

$$= \exp(I) \quad (347)$$

$$= \exp \left(\int_0^1 dt \left[\frac{t\lambda_k - (1-t)}{t^2\lambda_k + (1-t)^2} + \left(Q_k - \frac{t\lambda_k - (1-t)}{t^2\lambda_k + (1-t)^2} \right) \exp \left(-2\eta\tau(t^2\lambda_k + (1-t)^2) \right) \right] \right) \quad (348)$$

$$= \exp \left(\frac{1}{2} \log(\lambda_k) + \frac{1}{2} (\text{Ei}(-2\eta\tau) - \text{Ei}(-2\eta\tau\lambda_k)) + \frac{1}{2} \sqrt{\frac{\pi}{2\eta\tau(\lambda_k + 1)}} Q_k e^{-\frac{2\eta\tau\lambda_k}{\lambda_k + 1}} \right) \quad (349)$$

$$\left(\text{erf} \left(\sqrt{\frac{2\eta\tau}{\lambda_k + 1}} \right) + \text{erf} \left(\lambda_k \sqrt{\frac{2\eta\tau}{\lambda_k + 1}} \right) \right) \quad (350)$$

$$= \sqrt{\lambda_k} \sqrt{\exp \left(\text{Ei}(-2\eta\tau) - \text{Ei}(-2\eta\tau\lambda_k) \right)} \exp \left(\frac{1}{2} \sqrt{\frac{\pi}{2\eta\tau(\lambda_k + 1)}} Q_k e^{-\frac{2\eta\tau\lambda_k}{\lambda_k + 1}} \right) \quad (351)$$

$$\left(\text{erf} \left(\sqrt{\frac{2\eta\tau}{\lambda_k + 1}} \right) + \text{erf} \left(\lambda_k \sqrt{\frac{2\eta\tau}{\lambda_k + 1}} \right) \right) \quad (352)$$

$$\begin{aligned} \frac{\tilde{\lambda}_k}{\lambda_k} &= \frac{\Phi_k(1)^2}{\lambda_k} \\ &= \exp \left(\text{Ei}(-2\eta\tau) - \text{Ei}(-2\eta\tau\lambda_k) \right) \times \\ &\quad \exp \left(\sqrt{\frac{\pi}{2\eta\tau(\lambda_k + 1)}} Q_k e^{-\frac{2\eta\tau\lambda_k}{\lambda_k + 1}} \left(\text{erf} \left(\sqrt{\frac{2\eta\tau}{\lambda_k + 1}} \right) + \text{erf} \left(\lambda_k \sqrt{\frac{2\eta\tau}{\lambda_k + 1}} \right) \right) \right) \end{aligned} \quad (353)$$

1438 **Remarks**

- 1439 • The learning dynamics of generated variance were visualized in Fig. 7B.
- 1440 • The power law relationship between the convergence time τ_k^* of generated variance and the
- 1441 target variance λ_k was shown in Fig. 8. For the harmonic mean criterion, the power law
- 1442 coefficient was also close to -1 .

1443 **H.3 Learning dynamics of flow matching objective (two-layers)**

1444 Let $\mathbf{W} = PP^T$,

$$\nabla_{\mathbf{W}} \mathcal{L} = 2 \left[\mathbf{W}(t^2 \Sigma + (1-t)^2 \mathbf{I}) - (t \Sigma - (1-t) \mathbf{I}) \right]$$

$$\nabla_P \mathcal{L} = (\nabla_{\mathbf{W}} \mathcal{L})P + (\nabla_{\mathbf{W}} \mathcal{L})^T P \quad (354)$$

$$= \left[\nabla_{\mathbf{W}} \mathcal{L} + (\nabla_{\mathbf{W}} \mathcal{L})^T \right] P \quad (355)$$

$$\nabla_P \mathcal{L} = 2 \left[PP^T(t^2 \Sigma + (1-t)^2 \mathbf{I}) - (t \Sigma - (1-t) \mathbf{I}) \right] P \quad (356)$$

$$+ 2 \left[(t^2 \Sigma + (1-t)^2 \mathbf{I}) PP^T - (t \Sigma - (1-t) \mathbf{I}) \right] P \quad (357)$$

$$= 2 \left[-2(t \Sigma - (1-t) \mathbf{I})P + \right. \quad (358)$$

$$\left. PP^T(t^2 \Sigma + (1-t)^2 \mathbf{I})P + (t^2 \Sigma + (1-t)^2 \mathbf{I}) PP^T P \right] \quad (359)$$

$$(360)$$

1445 Similarly, let $\mathbf{u}_k^T P = q_k^T$

$$\mathbf{u}_k^T \nabla_P \mathcal{L} = 2 \left[-2\mathbf{u}_k^T (t\mathbf{\Sigma} - (1-t)\mathbf{I})P + \right. \quad (361)$$

$$\left. \mathbf{u}_k^T P P^T (t^2\mathbf{\Sigma} + (1-t)^2\mathbf{I})P + \mathbf{u}_k^T (t^2\mathbf{\Sigma} + (1-t)^2\mathbf{I})P P^T P \right] \quad (362)$$

$$= 2 \left[-2(t\lambda_k - (1-t))\mathbf{u}_k^T P + q_k^T \sum_m P^T \mathbf{u}_m (t^2\lambda_m + (1-t)^2)\mathbf{u}_m^T P \right. \quad (363)$$

$$\left. + (t^2\lambda_k + (1-t)^2)\mathbf{u}_k^T P \sum_n P^T \mathbf{u}_n \mathbf{u}_n^T P \right] \quad (364)$$

$$= 2 \left[-2(t\lambda_k - (1-t))q_k^T + q_k^T \sum_m q_m (t^2\lambda_m + (1-t)^2)q_m^T + (t^2\lambda_k + (1-t)^2)q_k^T \sum_n q_n q_n^T \right] \quad (365)$$

1446

$$\nabla_{q_k} \mathcal{L} = -4(t\lambda_k - (1-t))q_k + 2 \sum_m (q_k^T q_m) (t^2\lambda_m + (1-t)^2)q_m + 2(t^2\lambda_k + (1-t)^2) \sum_n (q_k^T q_n)q_n \quad (366)$$

$$= -4(t\lambda_k - (1-t))q_k + 2 \sum_m (t^2\lambda_m + (1-t)^2 + t^2\lambda_k + (1-t)^2) (q_k^T q_m)q_m \quad (367)$$

$$= -4(t\lambda_k - (1-t))q_k + 2 \sum_m (t^2\lambda_m + t^2\lambda_k + 2(1-t)^2) (q_k^T q_m) q_m \quad (368)$$

1447 **Simplify assumption: aligned initialization** Assume at initialization $q_k^T q_m \neq 0$

$$\nabla_{q_k} \mathcal{L} = -4(t\lambda_k - (1-t))q_k + 4(t^2\lambda_k + (1-t)^2)(q_k^T q_k)q_k$$

1448 The learning dynamics follows

$$\frac{d}{d\tau} q_k = -\eta \nabla_{q_k} \mathcal{L} \quad (369)$$

$$= 4\eta \left[(t\lambda_k - (1-t)) - (t^2\lambda_k + (1-t)^2)(q_k^T q_k) \right] q_k \quad (370)$$

$$\frac{d}{d\tau} (q_k^T q_k) = 4\eta \left[(t\lambda_k - (1-t)) - (t^2\lambda_k + (1-t)^2)(q_k^T q_k) \right] (q_k^T q_k)$$

1449 With initialization $q_k^T q_k(\tau = 0) = Q_k$, the solution reads

$$A := 4\eta(t\lambda_k - (1-t)) \quad (371)$$

$$B := 4\eta(t^2\lambda_k + (1-t)^2) \quad (372)$$

$$\|q_k\|^2(\tau) = \frac{A}{B} \frac{1}{1 + (\frac{A}{BQ_k} - 1)e^{-A\tau}} \quad (373)$$

$$= \frac{t\lambda_k - (1-t)}{t^2\lambda_k + (1-t)^2} \frac{Q_k}{Q_k + (\frac{t\lambda_k - (1-t)}{t^2\lambda_k + (1-t)^2} - Q_k)e^{-4\eta\tau(t\lambda_k - (1-t))}} \quad (374)$$

$$= Q_k^* \frac{Q_k}{Q_k + (Q_k^* - Q_k)e^{-4\eta\tau(t\lambda_k - (1-t))}} \quad (375)$$

1450 where,

$$Q_k^* = \frac{t\lambda_k - (1-t)}{t^2\lambda_k + (1-t)^2}$$

1451 So

$$\mathbf{W}(\tau; t) = \sum_k \|q_k\|^2(\tau) \mathbf{u}_k \mathbf{u}_k^T \quad (376)$$

$$= \sum_k \frac{Q_k}{Q_k + (Q_k^* - Q_k)e^{-4\eta\tau(t\lambda_k - (1-t))}} Q_k^* \mathbf{u}_k \mathbf{u}_k^T \quad (377)$$

$$= \sum_k \frac{Q_k}{Q_k + (Q_k^* - Q_k)e^{-4\eta\tau(t\lambda_k - (1-t))}} \frac{t\lambda_k - (1-t)}{t^2\lambda_k + (1-t)^2} \mathbf{u}_k \mathbf{u}_k^T \quad (378)$$

1452 Note that the optimal solution is non positive definite. So different from the diffusion case, there are
1453 two scenarios:

- 1454 • When $t > \frac{1}{\lambda_k + 1}$, $A > 0$, $Q^* > 0$. The dynamics is normal, converging to Q^* .
1455 $\lim_{\tau \rightarrow \infty} \|q_k\|^2(\tau) = Q_k^*$.
- 1456 • When $t < \frac{1}{\lambda_k + 1}$, $A < 0$, $Q^* < 0$. In this case, the ideal solution is “non-achievable” by
1457 a two layer network. $\lim_{\tau \rightarrow \infty} e^{-A\tau} \rightarrow \infty$, so $\lim_{\tau \rightarrow \infty} \|q_k\|^2(\tau) = 0$. In other words, 0
1458 becomes a stable fixed points instead of Q^* , and the solution $\|q_k\|^2(\tau)$ will be attracted to
1459 and stuck at 0.

1460 Thus,

$$\lim_{\tau \rightarrow \infty} \mathbf{W}(\tau; t) = \sum_{k, \text{ where } \lambda_k < \frac{1}{t} - 1} \frac{t\lambda_k - (1-t)}{t^2\lambda_k + (1-t)^2} \mathbf{u}_k \mathbf{u}_k^T$$

1461 Because of this, asymptotically speaking, the symmetric network architecture $P^T P$ will not approx-
1462 imate this vector field very well. Thus, for the purpose of studying the learning dynamics of flow
1463 matching model, some extension beyond the symmetric two-layer linear network is required for a
1464 thorough analysis.

1465 I Detailed Experimental Procedure

1466 I.1 Computational Resources

1467 All experiments were conducted on research cluster. Model training was performed on single A100 /
1468 H100 GPU. MLP training experiments took 20mins-2hrs while CNN based UNet training experiments
1469 took 5-8 hours, using around 20GB RAM.

1470 Evaluations were also done on single A100 / H100 GPU, with heavy covariance computation done
1471 with CUDA and trajectory plotting and fitting on CPU. Covariance computation for generated samples
1472 generally took a few minutes.

1473 I.2 MLP architecture inspired by UNet

1474 We used the following custom architecture inspired by UNet in [26] and [59] paper. The basic block
1475 is the following

```
class UNetMLPBlock(torch.nn.Module):
    def __init__(self,
        in_features, out_features, emb_features, dropout=0, skip_scale=1, eps=1e-5,
        adaptive_scale=True, init=dict(), init_zero=dict(),
    ):
        super().__init__()
        self.in_features = in_features
        self.out_features = out_features
        self.emb_features = emb_features
        self.dropout = dropout
        self.skip_scale = skip_scale
        self.adaptive_scale = adaptive_scale

        self.norm0 = nn.LayerNorm(in_features, eps=eps)
        #GroupNorm(num_channels=in_features, eps=eps)
        self.fc0 = Linear(in_features=in_features, out_features=out_features, **init)
        self.affine = Linear(in_features=emb_features, out_features=out_features*(2
            if adaptive_scale else 1), **init)
        self.norm1 = nn.LayerNorm(out_features, eps=eps)
        #GroupNorm(num_channels=out_features, eps=eps)
        self.fc1 = Linear(in_features=out_features, out_features=out_features,
            **init_zero)

        self.skip = None
        if out_features != in_features:
            self.skip = Linear(in_features=in_features, out_features=out_features,
                **init)

    def forward(self, x, emb):
        orig = x
        x = self.fc0(F.silu(self.norm0(x)))

        params = self.affine(emb).to(x.dtype) # .unsqueeze(1)
        if self.adaptive_scale:
            scale, shift = params.chunk(chunks=2, dim=1)
            x = F.silu(torch.addcmul(shift, self.norm1(x), scale + 1))
        else:
            x = F.silu(self.norm1(x.add_(params)))

        x = self.fc1(F.dropout(x, p=self.dropout, training=self.training))
        x = x.add_(self.skip(orig) if self.skip is not None else orig)
        x = x * self.skip_scale

        return x
```

1476

1477 and the full architecture backbone

```
class UNetBlockStyleMLP_backbone(nn.Module):
    """A time-dependent score-based model."""

    def __init__(self, ndim=2, nlayers=5, nhidden=64, time_embed_dim=64,):
        super().__init__()
        self.embed = GaussianFourierProjection(time_embed_dim, scale=1)
        layers = nn.ModuleList()
        layers.append(UNetMLPBlock(ndim, nhidden, time_embed_dim))
        for _ in range(nlayers-2):
            layers.append(UNetMLPBlock(nhidden, nhidden, time_embed_dim))
        layers.append(nn.Linear(nhidden, ndim))
        self.net = layers

    def forward(self, x, t_enc, cond=None):
        # t_enc : preconditioned version of sigma, usually
        # ln_std_vec = torch.log(std_vec) / 4
        if cond is not None:
            raise NotImplementedError("Conditional training is not implemented")
        t_embed = self.embed(t_enc)
        for layer in self.net[:-1]:
            x = layer(x, t_embed)
        pred = self.net[-1](x)
        return pred
```

1478

```
class EDMPrecondWrapper(nn.Module):
    def __init__(self, model, sigma_data=0.5, sigma_min=0.002, sigma_max=80,
                 rho=7.0):
        super().__init__()
        self.model = model
        self.sigma_data = sigma_data
        self.sigma_min = sigma_min
        self.sigma_max = sigma_max
        self.rho = rho

    def forward(self, X, sigma, cond=None, ):
        sigma[sigma == 0] = self.sigma_min
        ## edm preconditioning for input and output
        ## https://github.com/NVlabs/edm/blob/main/training/networks.py#L632
        # unsqueeze sigma to have same dimension as X (which may have 2-4 dim)
        sigma_vec = sigma.view([-1, ] + [1, ] * (X.ndim - 1))
        c_skip = self.sigma_data ** 2 / (sigma_vec ** 2 + self.sigma_data ** 2)
        c_out = sigma_vec * self.sigma_data / (sigma_vec ** 2 + self.sigma_data **
        2).sqrt()
        c_in = 1 / (self.sigma_data ** 2 + sigma_vec ** 2).sqrt()
        c_noise = sigma.log() / 4
        model_out = self.model(c_in * X, c_noise, cond=cond)
        return c_skip * X + c_out * model_out
```

1479

1480 This architecture can efficiently learn point cloud distributions. More details about the architecture
1481 and training can be found in code supplementary.

1482 I.3 EDM Loss Function

1483 We employ the loss function \mathcal{L}_{EDM} introduced in the Elucidated Diffusion Model (EDM) paper [26],
1484 which is one specific weighting scheme for training diffusion models.

1485 For each data point $\mathbf{x} \in \mathbb{R}^d$, the loss is computed as follows. The noise level for each data point is
1486 sampled from a log-normal distribution with hyperparameters P_{mean} and P_{std} (e.g., $P_{\text{mean}} = -1.2$ and

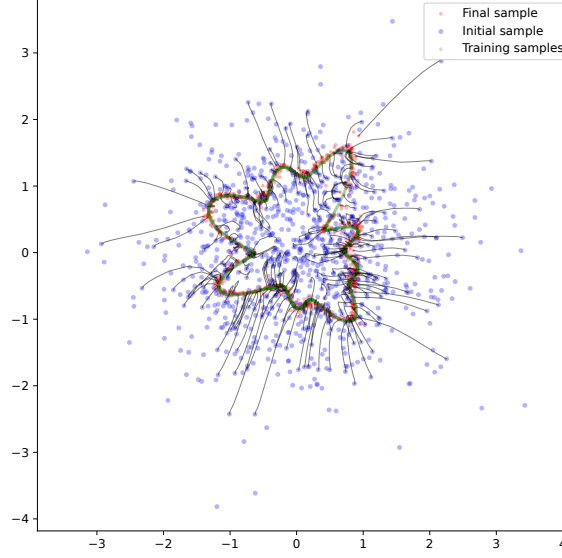


Figure 23: Example of learning to generate low-dimensional manifold with Song UNet-inspired MLP denoiser.

1487 $P_{\text{std}} = 1.2$). Specifically, the noise level σ is sampled via

$$\sigma = \exp(P_{\text{mean}} + P_{\text{std}} \epsilon), \quad \epsilon \sim \mathcal{N}(0, 1).$$

1488 The weighting function per noise scale is defined as:

$$w(\sigma) = \frac{\sigma^2 + \sigma_{\text{data}}^2}{(\sigma \sigma_{\text{data}})^2},$$

1489 with hyperparameter σ_{data} (e.g., $\sigma_{\text{data}} = 0.5$). The noisy input \mathbf{y} is created by the following,

$$\mathbf{y} = \mathbf{x} + \sigma \mathbf{n}, \quad \mathbf{n} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d),$$

1490 Let $D_{\theta}(\mathbf{y}, \sigma, \text{labels})$ denote the output of the denoising network when given the noisy input \mathbf{y} , the
 1491 noise level σ , and optional conditioning labels. The EDM loss per data point can be computed as:

$$\mathcal{L}(\mathbf{x}) = w(\sigma) \|D_{\theta}(\mathbf{x} + \sigma \mathbf{n}, \sigma, \text{labels}) - \mathbf{x}\|^2.$$

1492 Taking expectation over the data points and noise scales, the overall loss reads

$$\mathcal{L}_{EDM} = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \mathbb{E}_{\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)} \mathbb{E}_{\sigma} \left[w(\sigma) \|D_{\theta}(\mathbf{x} + \sigma \mathbf{n}, \sigma, \text{labels}) - \mathbf{x}\|^2 \right] \quad (379)$$

```
class EDMLoss:
    def __init__(self, P_mean=-1.2, P_std=1.2, sigma_data=0.5):
        self.P_mean = P_mean
        self.P_std = P_std
        self.sigma_data = sigma_data

    def __call__(self, net, X, labels=None, ):
        rnd_normal = torch.randn([X.shape[0],] + [1, ] * (X.ndim - 1),
                                device=X.device)
        # unsqueeze to match the ndim of X
        sigma = (rnd_normal * self.P_std + self.P_mean).exp()
        weight = (sigma ** 2 + self.sigma_data ** 2) / (sigma * self.sigma_data) ** 2
        # maybe augment
        n = torch.randn_like(X) * sigma
        D_yn = net(X + n, sigma, cond=labels, )
        loss = weight * ((D_yn - X) ** 2)
        return loss
```

1493

1494 I.4 Experiment 1: Diffusion Learning of High-dimensional Gaussian Data

1495 I.4.1 Data Generation and Covariance Specification

1496 We consider learning a score-based generative model on synthetic data drawn from a high-dimensional
 1497 Gaussian distribution of dimension $d = 128, 256, 512$. Specifically, we first sample a vector of
 1498 variances

$$\sigma^2 = (\sigma_1^2, \sigma_2^2, \dots, \sigma_d^2),$$

1499 where each σ_i^2 is drawn from a log-normal distribution (implemented via
 1500 `torch.exp(torch.randn(. . .))`). We then sort them in descending order and normalize
 1501 these variances to have mean equals 1 to fix the overall scale. Denoting

$$\mathbf{D} = \text{diag}(\sigma_1^2, \dots, \sigma_d^2),$$

1502 we generate a random rotation matrix $\mathbf{R} \in \mathbb{R}^{d \times d}$ by performing a QR decomposition of a matrix of
 1503 i.i.d. Gaussian entries. This allows us to construct the covariance

$$\Sigma = \mathbf{R} \mathbf{D} \mathbf{R}^\top.$$

1504 This rotation matrix \mathbf{R} is the eigenbasis of the true covariance matrix. To obtain training samples
 1505 $\{\mathbf{x}_i\} \subset \mathbb{R}^d$, we draw \mathbf{x}_i from $\mathcal{N}(\mathbf{0}, \Sigma)$. In practice, we generate a total of 10,000 samples and stack
 1506 them as pnts. We compute the empirical covariance of the training set, $\Sigma_{\text{emp}} = \text{Cov}(\text{pnts})$, and
 1507 verify that it is close to the prescribed true covariance Σ .

1508 I.4.2 Network Architecture and Training Setup

1509 We train a multi-layer perceptron (MLP) to approximate the noise conditional score function. The
 1510 base network, implemented as

`model=UNetBlockStyleMLP_backbone(ndim=d, nlayers=5, nhidden=256, time_embed_dim=256)`

1511 maps a data vector $\mathbf{x} \in \mathbb{R}^d$ and a time embedding τ to a vector of the same dimension \mathbb{R}^d . This
 1512 backbone is then wrapped in an EDM-style preconditioner via:

`model_precd = EDMPrecondWrapper(model, $\sigma_{\text{data}} = 0.5$, $\sigma_{\text{min}} = 0.002$, $\sigma_{\text{max}} = 80$, $\rho = 7.0$),`

1513 which standardizes and scales the input according to the EDM framework [26].

1514 We use EDM loss with hyperparameters $P_{\text{mean}} = -1.2$, $P_{\text{std}} = 1.2$, and $\sigma_{\text{data}} = 0.5$. We train the
 1515 model for 5000 steps using mini-batches of size 1024. The Adam optimizer is used with a learning
 1516 rate $1r = 10^{-4}$. Each training step processes a batch of data from pnts, adds noise with randomized
 1517 noise scales, and backpropagates through the EDM loss. The loss values at each training steps are
 1518 recorded.

1519 I.4.3 Sampling and Trajectory Visualization

1520 To visualize the sampling evolution, we sample from the diffusion model using the Heun’s 2nd order
 1521 deterministic sampler, starting from $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$

`edm_sampler(model, \mathbf{z} , num_steps = 20, $\sigma_{\text{min}} = 0.002$, $\sigma_{\text{max}} = 80$, $\rho = 7$).`

1522 We store these samples in `sample_store` to track how sampled distribution evolves over training.

1523 I.4.4 Covariance Evaluation in the True Eigenbasis

1524 To measure how well the trained model captures the true covariance structure, we compute the sample
 1525 covariance from the final generated samples, denoted $\hat{\Sigma}_{\text{sample}}$. We then project $\hat{\Sigma}_{\text{sample}}$ and the true Σ
 1526 into the eigenbasis of Σ . Specifically, letting \mathbf{R} be the rotation used above, we compute

$$\mathbf{R}^\top \hat{\Sigma}_{\text{sample}} \mathbf{R} \quad \text{and} \quad \mathbf{R}^\top \Sigma \mathbf{R}.$$

1527 Since $\Sigma = \mathbf{R} \mathbf{D} \mathbf{R}^\top$ is diagonal in that basis, we then compare the diagonal elements of $\mathbf{R}^\top \hat{\Sigma}_{\text{sample}} \mathbf{R}$
 1528 with $\text{diag}(\mathbf{D})$. As training proceeds, we track the ratio $\text{diag}(\mathbf{R}^\top \hat{\Sigma}_{\text{sample}} \mathbf{R}) / \text{diag}(\mathbf{D})$ to observe
 1529 convergence toward 1 across the spectrum.

1530 All intermediate results, including loss values and sampled trajectories, are stored to disk for later
 1531 analysis.

1532 I.5 Experiment 2: Diffusion Learning of MNIST | MLP

1533 I.5.1 Data Preprocessing

1534 For our second experiment, we apply the same EDM architecture to several natural image datasets:
1535 MNIST, CIFAR, AFHQ32, FFHQ32, FFHQ32-fixword, FFHQ32-randomword. All dataset except
1536 for MNIST are RGB images with 32 resolution, while MNIST is BW images with 28 resolution.
1537 These images were flattened as vectors, (784d for MNIST, 3072 for others) and stacked as pnts
1538 matrix. We normalize these intensities from $[0, 1]$ to $[-1, 1]$ by $\mathbf{x} \mapsto \frac{\mathbf{x}-0.5}{0.5}$. The resulting data
1539 tensor pnts is then transferred to GPU memory for training, and we estimate its empirical covariance
1540 $\Sigma_{\text{emp}} = \text{Cov}(\text{pnts})$ for reference.

1541 I.5.2 Network Architecture and Training Setup

1542 Since the natural dataset is higher dimensional than the synthetic data in the previous experiment, we
1543 use a deeper MLP network: For MNIST:

```
model = UNetBlockStyleMLP_backbone(ndim = 784, nlayers = 8, nhidden = 1024, time_embed_dim = 128).
```

1544 For others

```
model = UNetBlockStyleMLP_backbone(ndim = 3072, nlayers = 8, nhidden = 3072, time_embed_dim = 128).
```

1545 We again wrap this MLP in an EDM preconditioner:

```
model_preced = EDMPrecondWrapper(model,  $\sigma_{\text{data}} = 0.5$ ,  $\sigma_{\text{min}} = 0.002$ ,  $\sigma_{\text{max}} = 80$ ,  $\rho = 7.0$ ).
```

1546 The model is trained using the EDMLoss described in the previous section, with parameters $P_{\text{mean}} =$
1547 -1.2 , $P_{\text{std}} = 1.2$, and $\sigma_{\text{data}} = 0.5$. We set the training hyperparameters to $\text{lr} = 10^{-4}$, $\text{n_steps} =$
1548 100000 , and $\text{batch_size} = 2048$.

1549 I.5.3 Sampling and Analysis

1550 As before, we define a callback function `sampling_callback_fn` that periodically draws i.i.d.
1551 Gaussian noise $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{784})$ and applies the EDM sampler to produce generated samples. These
1552 intermediate samples are stored in `sample_store` for later analysis.

1553 In addition, we assess convergence of the mean of the generated samples by computing

$$\|\mathbb{E}[\mathbf{x}_{\text{out}}] - \mathbb{E}[\text{pnts}]\|^2,$$

1554 and we track how this mean-squared error evolves over training steps. We also examine the sample
1555 covariance $\hat{\Sigma}_{\text{sample}}$ of the final outputs, comparing its diagonal in a given eigenbasis to a target
1556 spectrum (e.g. the diagonal variances of the training data or a reference covariance).

1557 All trajectories and intermediate statistics are saved to disk for further inspection. In particular, we
1558 plot the difference between $\hat{\Sigma}_{\text{sample}}$ and Σ in an eigenbasis to illustrate whether the learned samples
1559 capture the underlying covariance structure of the training data.

1560 I.6 Experiment 3: Diffusion learning of Image Datasets with EDM-style CNN UNet

1561 We used model configuration similar to <https://github.com/NVlabs/edm>, but with simplified
1562 training code more similar to previous experiments.

1563 For the MNIST dataset, we trained a UNet-based CNN (with four blocks, each containing one layer,
1564 no attention, and channel multipliers of 1, 2, 3, and 4) on MNIST for 50,000 steps using a batch size
1565 of 2,048, a learning rate of 10^{-4} , 16 base model channels, and an evaluation sample size of 5,000.

1566 For the CIFAR-10 dataset, we trained a UNet model (with three blocks, each containing one layer,
1567 wide channels of size 128, and attention at resolution 16) for 50,000 steps using a batch size of 512, a
1568 learning rate of 10^{-4} , and an evaluation sample size of 2,000 (evaluated in batches of 1,024) with 20
1569 sampling steps.

1570 For the AFHQ, FFHQ (32 pixels) dataset, we used the same UNet architecture and training setup,
1571 with four blocks, wide channels of size 128, and attention at resolution 8, trained for 50,000 steps

1572 with a batch size of 256 and a learning rate of 1×10^{-4} . Evaluation was conducted on 2,000 samples
1573 in batches of 512.

1574 For the AFHQ, FFHQ (64 pixels) dataset, we trained a UNet model with four blocks (each containing
1575 one layer, wide channels of size 128, and attention at resolution 8) for 250,000 steps using a batch
1576 size of 256, a learning rate of 1×10^{-4} , and an evaluation sample size of 2,000 (evaluated in batches
1577 of 512).