Supplementary Material - A Complete Recipe for Diffusion Generative Models

Kushagra Pandey
Department of Computer Science
University of California, Irvine
pandeyk1@uci.edu

Stephan Mandt
Department of Computer Science
University of California, Irvine
mandt@uci.edu

A A Complete Recipe for SGMs

A.1 Proof of Theorems

A.1.1 Proof of Stationarity

Given a positive semi-definite diffusion matrix $\boldsymbol{D}(\mathbf{z})$ and a skew-symmetric matrix $\boldsymbol{Q}(\mathbf{z})$, we can parameterize the drift $f(\mathbf{z})$ for a stochastic process: $d\mathbf{z} = \boldsymbol{f}(\mathbf{z})dt + \sqrt{2\boldsymbol{D}(\mathbf{z})}d\mathbf{w}_t$ as follows:

$$\boldsymbol{f}(\mathbf{z}) = -(\boldsymbol{D}(\mathbf{z}) + \boldsymbol{Q}(\mathbf{z}))\nabla H + \tau(\mathbf{z}), \qquad \tau_i(\mathbf{z}) = \sum_j \frac{\partial}{\partial \mathbf{z}_j}\left(\boldsymbol{D}_{ij}(\mathbf{z}) + \boldsymbol{Q}_{ij}(\mathbf{z})\right) \tag{30}$$

Theorem 2.1 then states that the distribution $p_s(z) \propto \exp\left(-H(\mathbf{z})\right)$ will be the stationary distribution for the stochastic process as defined above.

Proof. Using the Fokker-Planck formulation for the stochastic dynamics, we have:

$$\frac{\partial p_t(\mathbf{z})}{\partial t} = -\sum_i \frac{\partial}{\partial z_i}\left(\boldsymbol{f}_i(\mathbf{z})p_t(\mathbf{z})\right) + \sum_{i,j} \frac{\partial^2}{\partial \mathbf{z}_i \mathbf{z}_j}\left(\boldsymbol{D}_{ij}(\mathbf{z})p_t(\mathbf{z})\right) \tag{31}$$

Furthermore, we have:

$$\boldsymbol{f}_i(\mathbf{z}) = \tau_i(\mathbf{z}) - \sum_j \left(\boldsymbol{D}_{ij}(\mathbf{z}) + \boldsymbol{Q}_{ij}(\mathbf{z})\right)\nabla H_j(\mathbf{z}) \tag{32}$$

Therefore,

$$\sum_i \frac{\partial}{\partial \mathbf{z}_i}\left(\boldsymbol{f}_i(\mathbf{z})p_t(\mathbf{z})\right) = \sum_i \frac{\partial}{\partial z_i}\left[\tau_i(\mathbf{z})p_t(\mathbf{z}) - \sum_j \left(\boldsymbol{D}_{ij}(\mathbf{z}) + \boldsymbol{Q}_{ij}(\mathbf{z})\right)\nabla H_j(\mathbf{z})p_t(\mathbf{z})\right] \tag{33}$$

$$= \sum_i \frac{\partial}{\partial \mathbf{z}_i}\left[\sum_j \frac{\partial}{\partial \mathbf{z}_j}\left(\boldsymbol{D}_{ij}(\mathbf{z}) + \boldsymbol{Q}_{ij}(\mathbf{z})\right)p_t(\mathbf{z}) - \sum_j \left(\boldsymbol{D}_{ij}(\mathbf{z}) + \boldsymbol{Q}_{ij}(\mathbf{z})\right)\nabla H_j(\mathbf{z})p_t(\mathbf{z})\right] \tag{34}$$

$$= \sum_{i,j} \frac{\partial}{\partial \mathbf{z}_i}\left[\frac{\partial}{\partial \mathbf{z}_j}\left(\boldsymbol{D}_{ij}(\mathbf{z}) + \boldsymbol{Q}_{ij}(\mathbf{z})\right)p_t(\mathbf{z})\right] - \sum_{i,j} \frac{\partial}{\partial \mathbf{z}_i}\left[\left(\boldsymbol{D}_{ij}(\mathbf{z}) + \boldsymbol{Q}_{ij}(\mathbf{z})\right)\nabla H_j(\mathbf{z})p_t(\mathbf{z})\right] \tag{35}$$

$$= \sum_{i,j} \frac{\partial}{\partial \mathbf{z}_i}\left[\frac{\partial}{\partial \mathbf{z}_j}\left(\boldsymbol{D}_{ij}(\mathbf{z}) + \boldsymbol{Q}_{ij}(\mathbf{z})\right)p_t(\mathbf{z})\right] - \underbrace{\sum_{i,j} \frac{\partial}{\partial \mathbf{z}_i}\left[\left(\boldsymbol{D}_{ij}(\mathbf{z}) + \boldsymbol{Q}_{ij}(\mathbf{z})\right)\nabla H_j(\mathbf{z})p_t(\mathbf{z})\right]}_{=F(\mathbf{z})} \tag{36}$$

Substituting the above result in the Fokker-Planck formulation in Eqn. 31, we have:

$$\frac{\partial p_t(\mathbf{z})}{\partial t} = F(\mathbf{z}) - \sum_{i,j} \frac{\partial}{\partial \mathbf{z}_i}\left[\frac{\partial}{\partial \mathbf{z}_j}\left(\boldsymbol{D}_{ij}(\mathbf{z}) + \boldsymbol{Q}_{ij}(\mathbf{z})\right) p_t(\mathbf{z}) - \frac{\partial}{\partial \mathbf{z}_j}\left(\boldsymbol{D}_{ij}(\mathbf{z}) p_t(\mathbf{z})\right)\right] \tag{37}$$

$$= F(\mathbf{z}) - \sum_{i,j} \frac{\partial}{\partial \mathbf{z}_i}\left[\frac{\partial}{\partial \mathbf{z}_j}\left(\boldsymbol{Q}_{ij}(\mathbf{z})\right) p_t(\mathbf{z}) - \boldsymbol{D}_{ij}(\mathbf{z})\frac{\partial}{\partial \mathbf{z}_j}\left(p_t(\mathbf{z})\right)\right] \tag{38}$$

$$= F(\mathbf{z}) - \sum_{i,j} \frac{\partial}{\partial \mathbf{z}_i}\left[\frac{\partial}{\partial \mathbf{z}_j}\left(\boldsymbol{Q}_{ij}(\mathbf{z}) p_t(\mathbf{z})\right) - \left(\boldsymbol{D}_{ij}(\mathbf{z}) + \boldsymbol{Q}_{ij}(\mathbf{z})\right)\frac{\partial}{\partial \mathbf{z}_j}\left(p_t(\mathbf{z})\right)\right] \tag{39}$$

$$= F(\mathbf{z}) + \underbrace{\sum_{i,j} \frac{\partial}{\partial \mathbf{z}_i}\left[\left(\boldsymbol{D}_{ij}(\mathbf{z}) + \boldsymbol{Q}_{ij}(\mathbf{z})\right)\frac{\partial p_t(\mathbf{z})}{\partial \mathbf{z}_j}\right]}_{=G(\mathbf{z})} - \sum_{i,j} \frac{\partial^2}{\partial \mathbf{z}_i \partial \mathbf{z}_j}\left(\boldsymbol{Q}_{ij}(\mathbf{z}) p_t(\mathbf{z})\right) \tag{40}$$

$$= F(\mathbf{z}) + G(\mathbf{z}) - \sum_{i,j} \frac{\partial^2}{\partial \mathbf{z}_i \partial \mathbf{z}_j}\left(\boldsymbol{Q}_{ij}(\mathbf{z}) p_t(\mathbf{z})\right) \tag{41}$$

Since $\boldsymbol{Q}(\mathbf{z})$ is a skew-symmetric matrix, $\sum_{i,j} \frac{\partial^2}{\partial \mathbf{z}_i \partial \mathbf{z}_j}\left(\boldsymbol{Q}_{ij}(\mathbf{z}) p_t(\mathbf{z})\right) = 0$. Therefore,

$$\frac{\partial p_t(\mathbf{z})}{\partial t} = F(\mathbf{z}) + G(\mathbf{z}) \tag{42}$$

$$= \sum_{i,j} \frac{\partial}{\partial \mathbf{z}_i}\left[\left(\boldsymbol{D}_{ij}(\mathbf{z}) + \boldsymbol{Q}_{ij}(\mathbf{z})\right)\left(\nabla H_j(\mathbf{z}) p_t(\mathbf{z}) + \frac{\partial p_t(\mathbf{z})}{\partial \mathbf{z}_j}\right)\right] \tag{43}$$

$$= \sum_{i} \frac{\partial}{\partial \mathbf{z}_i}\left[\left(\boldsymbol{D}_{i}(\mathbf{z}) + \boldsymbol{Q}_{i}(\mathbf{z})\right)\left(\nabla H(\mathbf{z}) p_t(\mathbf{z}) + \frac{\partial p_t(\mathbf{z})}{\partial \mathbf{z}}\right)\right] \tag{44}$$

$$= \nabla \cdot \left[\left(\boldsymbol{D}_{i}(\mathbf{z}) + \boldsymbol{Q}_{i}(\mathbf{z})\right)\left(\nabla H(\mathbf{z}) p_t(\mathbf{z}) + \frac{\partial p_t(\mathbf{z})}{\partial \mathbf{z}}\right)\right] \tag{45}$$

Therefore, we have the following parameterization for the Fokker-Planck formulation for the defined stochastic dynamics:

$$\frac{\partial p_t(\mathbf{z})}{\partial t} = \nabla \cdot \left[\left(\boldsymbol{D}_{i}(\mathbf{z}) + \boldsymbol{Q}_{i}(\mathbf{z})\right)\left(\nabla H(\mathbf{z}) p_t(\mathbf{z}) + \frac{\partial p_t(\mathbf{z})}{\partial \mathbf{z}}\right)\right] \tag{46}$$

Substituting $p_s(\mathbf{z}) \propto \exp\left(-H(\mathbf{z})\right)$ in the above result implies $\frac{\partial p_t(\mathbf{z})}{\partial t} = 0$. This implies that $p_s(\mathbf{z}) \propto \exp\left(-H(\mathbf{z})\right)$ is the form of the stationary distribution for the drift parameterization $\boldsymbol{f}(\mathbf{z}) = -(\boldsymbol{D}(\mathbf{z}) + \boldsymbol{Q}(\mathbf{z}))\nabla H + \tau(\mathbf{z})$. An alternative version of this proof can be found in Ma et al. [1] □

A.1.2 Proof of Completeness

We now state the proof for Theorem 2.2 which states that for every stochastic dynamics $d\mathbf{z} = \boldsymbol{f}(\mathbf{z})dt + \sqrt{2\boldsymbol{D}(\mathbf{z})}d\mathbf{w}_t$ with the desired stationary distribution $p_s(\mathbf{z}) \propto \exp\left(-H(\mathbf{z})\right)$, there exists a positive semi-definite $\boldsymbol{D}(\mathbf{z})$ and a skew-symmetric $\boldsymbol{Q}(\mathbf{z})$ such that $\boldsymbol{f}(\mathbf{z}) = -(\boldsymbol{D}(\mathbf{z}) + \boldsymbol{Q}(\mathbf{z}))\nabla H + \boldsymbol{\tau}(\mathbf{z})$ holds. We directly include the proof from Ma et al. [1] for completeness.

Proof. We have the following result:

$$\boldsymbol{f}_i(\mathbf{z}) p_s(\mathbf{z}) = \boldsymbol{\tau}_i(\mathbf{z}) p_s(\mathbf{z}) - \sum_j \left(\boldsymbol{D}_{ij}(\mathbf{z}) + \boldsymbol{Q}_{ij}(\mathbf{z})\right)\nabla H_j(\mathbf{z}) p_s(\mathbf{z}) \tag{47}$$

$$= \sum_j \frac{\partial}{\partial \mathbf{z}_j}\left(\boldsymbol{D}_{ij}(\mathbf{z}) + \boldsymbol{Q}_{ij}(\mathbf{z})\right) p_s(\mathbf{z}) - \sum_j \left(\boldsymbol{D}_{ij}(\mathbf{z}) + \boldsymbol{Q}_{ij}(\mathbf{z})\right)\nabla H_j(\mathbf{z}) p_s(\mathbf{z}) \tag{48}$$

$$= \sum_j \frac{\partial}{\partial \mathbf{z}_j}\left[\left(\boldsymbol{D}_{ij}(\mathbf{z}) + \boldsymbol{Q}_{ij}(\mathbf{z})\right) p_s(\mathbf{z})\right] \tag{49}$$

which implies,

$$\sum_j \frac{\partial}{\partial \mathbf{z}_j} \left(\boldsymbol{Q}_{ij}(\mathbf{z}) p_s(\mathbf{z})\right) = \boldsymbol{f}_i(\mathbf{z}) p_s(\mathbf{z}) - \sum_j \frac{\partial}{\partial \mathbf{z}_j} \left(\boldsymbol{D}_{ij}(\mathbf{z}) p_s(\mathbf{z})\right) \tag{50}$$

Furthermore, from the Fokker-Planck formalism,

$$\frac{\partial p_t(\mathbf{z})}{\partial t} = -\sum_i \frac{\partial}{\partial z_i} \left(\boldsymbol{f}_i(\mathbf{z}) p_t(\mathbf{z})\right) + \sum_{i,j} \frac{\partial^2}{\partial \mathbf{z}_i \mathbf{z}_j} \left(\boldsymbol{D}_{ij}(\mathbf{z}) p_t(\mathbf{z})\right) \tag{51}$$

$$= -\sum_i \frac{\partial}{\partial z_i} \left[\boldsymbol{f}_i(\mathbf{z}) p_t(\mathbf{z}) - \sum_j \frac{\partial}{\partial \mathbf{z}_j} \left(\boldsymbol{D}_{ij}(\mathbf{z}) p_t(\mathbf{z})\right)\right] \tag{52}$$

For $p_t(\mathbf{z}) = p_s(\mathbf{z})$, we have,

$$\sum_i \frac{\partial}{\partial z_i} \left[\boldsymbol{f}_i(\mathbf{z}) p_t(\mathbf{z}) - \sum_j \frac{\partial}{\partial \mathbf{z}_j} \left(\boldsymbol{D}_{ij}(\mathbf{z}) p_t(\mathbf{z})\right)\right] = 0 \tag{53}$$

Denoting the Fourier transform of $\boldsymbol{Q}(\mathbf{z}) p_s(\mathbf{z})$ as $\hat{\boldsymbol{Q}}(\boldsymbol{k})$ and the Fourier transform of $\boldsymbol{f}_i(\mathbf{z}) p_t(\mathbf{z}) - \sum_j \frac{\partial}{\partial \mathbf{z}_j} \left(\boldsymbol{D}_{ij}(\mathbf{z}) p_t(\mathbf{z})\right)$ by $\hat{\boldsymbol{F}}(\boldsymbol{k})$, then from Eqns. 50 and 53 we have the following equations in the Fourier-space:

$$2\pi i \hat{\boldsymbol{Q}} \boldsymbol{k} = \hat{\boldsymbol{F}} \tag{54}$$

$$\boldsymbol{k}^T \hat{\boldsymbol{F}} = 0 \tag{55}$$

Therefore, it implies that the matrix $\hat{\boldsymbol{Q}}$ is a projection matrix from $\boldsymbol{k}$ to the span of $\hat{\boldsymbol{F}}$. Consequently, the matrix $\hat{\boldsymbol{Q}}$ can be constructed as: $\hat{\boldsymbol{Q}} = (2\pi i)^{-1} \frac{\hat{\boldsymbol{F}} \boldsymbol{k}^T}{\boldsymbol{k}^T \boldsymbol{k}} - (2\pi i)^{-1} \frac{\boldsymbol{k} \hat{\boldsymbol{F}}^T}{\boldsymbol{k}^T \boldsymbol{k}}$. This construction also shows that $\hat{\boldsymbol{Q}}$ is skew-symmetric. Moreover, the skew-symmetric $\boldsymbol{Q}$ can be obtained by computing the Inverse-Fourier transform of $(p_s(\mathbf{z}))^{-1} \hat{\boldsymbol{Q}}$ $\square$

A.2 Existing SGMs parameterized using the SGM recipe

In this section, we provide examples of SGMs that can be cast under the recipe proposed in Section 2.2. It is worth noting that under the completeness framework proposed in Section 2.2, given a positive semi-definite diffusion matrix $\boldsymbol{D}(\mathbf{z})$, a skew-symmetric curl matrix $\boldsymbol{Q}(\mathbf{z})$ and the Hamiltonian $H(\mathbf{z})$ corresponding to a specified target distribution $p_s(\mathbf{z})$, the forward process SDE can be parameterized in terms of the target distribution as follows:

$$H(\mathbf{z}) = U(\mathbf{x}) + \frac{\mathbf{m}^T M^{-1} \mathbf{m}}{2}, \quad \nabla H(\mathbf{z}) = \begin{pmatrix} \nabla U(\mathbf{x}) \\ M^{-1} \mathbf{m} \end{pmatrix} \tag{56}$$

$$\boldsymbol{f}(\mathbf{z}) = -(\boldsymbol{D}(\mathbf{z}) + \boldsymbol{Q}(\mathbf{z})) \begin{pmatrix} \nabla U(\mathbf{x}) \\ M^{-1} \mathbf{m} \end{pmatrix} + \tau(\mathbf{z}) \tag{57}$$

$$d\mathbf{z} = \boldsymbol{f}(\mathbf{z}) dt + \sqrt{2\boldsymbol{D}(\mathbf{z})} d\mathbf{w}_t \tag{58}$$

We now recast several existing SGMs under this framework:

A.2.1 Non-augmented SGMs

For SGMs with a non-augmented form, we assume auxiliary variables $\mathbf{m}_t = 0$ with the equilibrium distribution given by $p_s(\mathbf{z}) = \mathcal{N}(\mathbf{0_d}, \mathbf{I_d})$. The Hamiltonian and its gradient can then be specified as follows:

$$H(\mathbf{z}) = \frac{\mathbf{x}^T \mathbf{x}}{2}, \qquad \nabla H(\mathbf{z}) = \mathbf{x} \tag{59}$$

For the choice of $\boldsymbol{D}_{\text{VP}}(\mathbf{z}) = \frac{\beta_t}{2} \boldsymbol{I}_d$ and $\boldsymbol{Q}_{\text{VP}}(\mathbf{z}) = \mathbf{0}_d$, the drift for the forward SDE defined in Eqn. 57 reduces to the following form:

$$d\mathbf{x} = -\frac{\beta_t}{2} \mathbf{x} dt + \sqrt{\beta_t} d\mathbf{w}_t \tag{60}$$

where β_t is a time-dependent constant. The forward SDE in Eqn. 60 is the same as the VP-SDE proposed in [2]. From our recipe, the stationary distribution for the VPSDE should be $p_s(\mathbf{x}) = \mathcal{N}(\mathbf{0}_d, \boldsymbol{I}_d)$. Indeed the perturbation kernel for the VP-SDE is specified as follows:

$$p(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_0 e^{-\frac{1}{2}\int_0^t \beta(s)ds}, (1 - e^{-\int_0^t \beta(s)ds})^2 \boldsymbol{I}_d) \tag{61}$$

which converges to a standard Gaussian distribution as $t \to \infty$. This example suggests that the proposed recipe can be used to establish the validity of the convergence of a forward process with a specified stationary distribution $p_s(\mathbf{z})$ without deriving the perturbation kernel or relying on physical intuition. Interestingly, the Variance-Exploding (VE) SDE [2] is one example that cannot be cast in our framework. This would mean that it will not asymptotically converge to the standard Gaussian distribution at equilibrium. Indeed, this can be confirmed from the analytical form of the perturbation kernel of the VE-SDE given by:

$$p(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_0, [\sigma^2(t) - \sigma^2(0)]\boldsymbol{I}_d) \tag{62}$$

As $t \to \infty$, the variance of the perturbation kernel of the VE-SDE grows unbounded and therefore does not converge to the equilibrium distribution $\mathcal{N}(\mathbf{0}_d, \boldsymbol{I}_d)$. This should not be surprising since the VE-SDE, for the specified hamiltonian, could not be recast in the completeness framework, to begin with.

A.2.2 Augmented SGMs

For SGMs with an augmented state-space (data state space $\mathbf{x}_t$ + auxiliary variables $\mathbf{m}_t$), we assume the equilibrium distribution $p_s(\mathbf{z}) = \mathcal{N}(\mathbf{0_d}, \mathbf{I_d})\mathcal{N}(\mathbf{0_d}, \mathbf{MI_d})$. The Hamiltonian and its gradient can then be specified as follows:

$$H(\mathbf{z}) = \frac{\mathbf{x}^T\mathbf{x}}{2} + \frac{\mathbf{m}^T M^{-1}\mathbf{m}}{2}, \qquad \nabla H(\mathbf{z}) = \begin{pmatrix} \mathbf{x} \\ M^{-1}\mathbf{m} \end{pmatrix} \tag{63}$$

For this choice of H, the forward SDE representative of PSLD can be obtained by choosing the following $\boldsymbol{D}$ and $\boldsymbol{Q}$ matrices:

$$\boldsymbol{D}_{\text{PSLD}} = \frac{\beta}{2}\left(\begin{pmatrix} \Gamma & 0 \\ 0 & M\gamma \end{pmatrix} \otimes \boldsymbol{I}_d\right) \qquad \boldsymbol{Q}_{\text{PSLD}} = \frac{\beta}{2}\left(\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \otimes \boldsymbol{I}_d\right) \tag{64}$$

Similarly, the forward SDE representative of CLD [3] can be obtained by choosing:

$$\boldsymbol{D}_{\text{CLD}} = \beta\left(\begin{pmatrix} 0 & 0 \\ 0 & \Gamma \end{pmatrix} \otimes \boldsymbol{I}_d\right) \qquad \boldsymbol{Q}_{\text{CLD}} = \beta\left(\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \otimes \boldsymbol{I}_d\right) \tag{65}$$

Since both PSLD and CLD can be shown to converge asymptotically to $p_s(\mathbf{z})$ from the analytical form of their perturbation kernels $p(\mathbf{z}_t|\mathbf{z}_0)$, the result from our completeness framework is valid. More importantly, given a forward process for an SGM, our recipe can be used to validate if the SGM converges to a specified equilibrium distribution without the need for analytically determining the perturbation kernel (which is usually non-trivial).

B Phase Space Langevin Diffusion

In this section, we elaborate on several aspects of PSLD, which were discussed briefly in the main text. Moreover, we work with the following form of the forward process for PSLD:

$$\begin{pmatrix} d\mathbf{x}_t \\ d\mathbf{m}_t \end{pmatrix} = \left(\frac{\beta_t}{2}\begin{pmatrix} -\Gamma & M^{-1} \\ -1 & -\nu \end{pmatrix} \otimes \boldsymbol{I}_d\right)\begin{pmatrix} \mathbf{x}_t \\ \mathbf{m}_t \end{pmatrix} dt + \left(\begin{pmatrix} \sqrt{\Gamma\beta_t} & 0 \\ 0 & \sqrt{M\nu\beta_t} \end{pmatrix} \otimes \boldsymbol{I}_d\right) d\mathbf{w}_t, \tag{66}$$

It is worth noting that the form of the forward process defined in Eqn. 66 is more general than Eqn. 12 in the sense that we consider a time-dependent β_t here for our discussions. We can then reason about the forward SDE in Eqn. 12 by fixing β_t to a time-independent quantity β for all subsequent analyses.

B.1 Critical Damping in PSLD

Assuming $\beta_t = 1$ and $\mathbf{x}_t, \mathbf{m}_t \in \mathbb{R}$ for simplicity, the equations of motion for the deterministic dynamics can be specified as follows:

$$\frac{dx_t}{dt} = -\Gamma x_t + M^{-1} m_t \tag{67}$$

$$\frac{dm_t}{dt} = -x_t - \nu m_t \tag{68}$$

From Eqn. 67, we have:

$$m_t = M \left(\frac{dx_t}{dt} + \Gamma x_t \right) \tag{69}$$

Furthermore, taking the derivative of both sides in Eqn. 67, we have:

$$\frac{d^2 x_t}{dt^2} = -\Gamma \frac{dx_t}{dt} + M^{-1} \frac{dm_t}{dt} \tag{70}$$

$$= -\Gamma \frac{dx_t}{dt} + M^{-1} \left[-x_t - \nu m_t \right] \tag{71}$$

$$= -\Gamma \frac{dx_t}{dt} + M^{-1} \left[-x_t - \nu M \left(\frac{dx_t}{dt} + \Gamma x_t \right) \right] \tag{72}$$

$$= -\Gamma \frac{dx_t}{dt} - M^{-1} x_t - \nu \left(\frac{dx_t}{dt} + \Gamma x_t \right) \tag{73}$$

$$= -\Gamma \frac{dx_t}{dt} - M^{-1} x_t - \nu \frac{dx_t}{dt} - \Gamma \nu x_t \tag{74}$$

$$= -(\Gamma + \nu) \frac{dx_t}{dt} - M^{-1} x_t - \Gamma \nu x_t \tag{75}$$

We, therefore, have the following dynamical equation in terms of the position:

$$\frac{d^2 x_t}{dt^2} + (\Gamma + \nu) \frac{dx_t}{dt} + (M^{-1} + \Gamma \nu) x_t = 0 \tag{76}$$

Assuming the exponential *ansatz* $\mathbf{x}_t = \exp(-\lambda t)$ and plugging into the above ODE, we have the following result:

$$\exp(-\lambda t) \left[\lambda^2 - (\Gamma + \nu) \lambda + (M^{-1} + \Gamma \nu) \right] = 0 \tag{77}$$

which implies,

$$\lambda^2 - (\Gamma + \nu) \lambda + (M^{-1} + \Gamma \nu) = 0 \tag{78}$$

$$\lambda = \frac{(\Gamma + \nu) \pm \sqrt{(\Gamma + \nu)^2 - 4M^{-1} - 4\Gamma \nu}}{2} \tag{79}$$

$$\lambda = \frac{(\Gamma + \nu) \pm \sqrt{(\Gamma - \nu)^2 - 4M^{-1}}}{2} \tag{80}$$

Corresponding to the value of ν, Γ and M, we can now have the following damping conditions:

(i) $(\Gamma - \nu)^2 < 4M^{-1}$ corresponds to *Underdamped dynamics*

(ii) $(\Gamma - \nu)^2 = 4M^{-1}$ corresponds to *Critical damping*

(iii) $(\Gamma - \nu)^2 > 4M^{-1}$ corresponds to *Overdamped dynamics*

Moreover when $\Gamma = 0$ and $\bar{\nu} = M\nu$, we get: $\bar{\nu}^2 = 4M$ which is the critical damping condition proposed in Dockhorn et al. [3]. Therefore, similar to Dockhorn et al. [3], we work in the Critical Damping regime specified by the condition $(\Gamma - \nu)^2 = 4M^{-1}$

B.2 PSLD Training

B.2.1 Overall Training Framework in PSLD

Following the derivation in Dockhorn et al. [3], the maximum likelihood training formulation for score matching can be specified as follows. Let p_0, q_0 be two densities with corresponding marginal densities p_t and q_t (for forward diffusion using PSLD defined in Eqn. 66) at time t. As shown in Song et al. [4], the KL-Divergence between p_0 and q_0 can then be expressed as a mixture of score-matching losses over multiple time scales as follows:

$$\begin{aligned} D_{\mathrm{KL}}(p_0 \parallel q_0) &= D_{\mathrm{KL}}(p_0 \parallel q_0) - D_{\mathrm{KL}}(p_T \parallel q_T) + D_{\mathrm{KL}}(p_T \parallel q_T) \\ &= -\int_0^T \frac{\partial D_{\mathrm{KL}}(p_t \parallel q_t)}{\partial t} dt + D_{\mathrm{KL}}(p_T \parallel q_T) \end{aligned} \tag{81}$$

Following the derivation from Song et al. [4], the Fokker-Planck equation describing the time evolution of the probability density function of the SDE in Eqn. 66 can be expressed as follows:

$$\begin{aligned} \frac{\partial p_t(\mathbf{z}_t)}{\partial t} &= \nabla_{\mathbf{z}_t} \cdot \left[\tfrac{1}{2}\left(G(t)G(t)^\top \otimes \boldsymbol{I}_d\right) \nabla_{\mathbf{z}_t} p_t(\mathbf{z}_t) - p_t(\mathbf{z}_t)(f(t)\otimes \boldsymbol{I}_d)\mathbf{z}_t\right] \\ &= \nabla_{\mathbf{z}_t} \cdot \left[\boldsymbol{h}_p(\mathbf{z}_t, t) p_t(\mathbf{z}_t)\right] \end{aligned} \tag{82}$$

where

$$\boldsymbol{h}_p(\mathbf{z}_t, t) \coloneqq \tfrac{1}{2}\left(G(t)G(t)^\top \otimes \boldsymbol{I}_d\right) \nabla_{\mathbf{z}_t} \log p_t(\mathbf{z}_t) - (f(t)\otimes \boldsymbol{I}_d)\mathbf{z}_t \tag{83}$$

$$f(t) = \left(\frac{\beta_t}{2}\begin{pmatrix} -\Gamma & M^{-1} \\ -1 & -\nu \end{pmatrix}\right) \tag{84}$$

$$G(t) = \begin{pmatrix} \sqrt{\Gamma\beta_t} & 0 \\ 0 & \sqrt{M\nu\beta_t} \end{pmatrix} \tag{85}$$

Further assuming that $\log p_t(\mathbf{z}_t)$ and $\log q_t(\mathbf{z}_t)$ are smooth functions with at most polynomial growth at infinity, we have

$$\lim_{\mathbf{z}_t \to \infty} \boldsymbol{h}_p(\mathbf{z}_t, t) p_t(\mathbf{z}_t) = \lim_{\mathbf{z}_t \to \infty} \boldsymbol{h}_q(\mathbf{z}_t, t) q_t(\mathbf{z}_t) = 0. \tag{86}$$

Using the above fact, we can compute the time-derivative of the Kullback–Leibler divergence between p_t and q_t as

$$\begin{aligned} \frac{\partial D_{\mathrm{KL}}(p_t \parallel q_t)}{\partial t} &= \frac{\partial}{\partial t}\int p_t(\mathbf{z}_t)\log\frac{p_t(\mathbf{z}_t)}{q_t(\mathbf{z}_t)}\, d\mathbf{z}_t \\ &= \int \frac{\partial p_t(\mathbf{z}_t)}{\partial t}\log\frac{p_t(\mathbf{z}_t)}{q_t(\mathbf{z}_t)}\, d\mathbf{z}_t - \int \frac{p_t(\mathbf{z}_t)}{q_t(\mathbf{z}_t)}\frac{\partial q_t(\mathbf{z}_t)}{\partial t} d\mathbf{z}_t \\ &= -\int p_t(\mathbf{z}_t)\Big[\boldsymbol{h}_p(\mathbf{z}_t, t) - \boldsymbol{h}_q(\mathbf{z}_t, t)\Big]^\top \Big[\nabla_{\mathbf{z}_t}\log p_t(\mathbf{z}_t) - \nabla_{\mathbf{z}_t}\log q_t(\mathbf{z}_t)\Big]\, d\mathbf{z}_t \\ &= -\frac{1}{2}\int p_t(\mathbf{z}_t)\Big[\nabla_{\mathbf{z}_t}\log p_t(\mathbf{z}_t) - \nabla_{\mathbf{z}_t}\log q_t(\mathbf{z}_t)\Big]^\top \Big(G(t)G(t)^\top \otimes \boldsymbol{I}_d\Big) \\ &\qquad \Big[\nabla_{\mathbf{u}_t}\log p_t(\mathbf{z}_t) - \nabla_{\mathbf{z}_t}\log q_t(\mathbf{z}_t)\Big]\, d\mathbf{z}_t \\ &= -\frac{1}{2}\int p_t(\mathbf{z}_t)\Big[\Gamma\beta_t\|\nabla_{\mathbf{x}_t}\log p_t(\mathbf{z}_t) - \nabla_{\mathbf{x}_t}\log q_t(\mathbf{z}_t)\|_2^2 + \\ &\qquad M\nu\beta_t\|\nabla_{\mathbf{m}_t}\log p_t(\mathbf{z}_t) - \nabla_{\mathbf{m}_t}\log q_t(\mathbf{z}_t)\|_2^2\Big]\, d\mathbf{z}_t \end{aligned} \tag{87}$$

Assuming our generative prior $p(x_T)$ matches the equilibrium state of the forward process closely i.e. $D_{\mathrm{KL}}(p_T \parallel q_T) \approx 0$ and substituting the result in Eqn. 87 in Eqn. 81, we get the following score-matching objective corresponding to the maximum-likelihood objective in Eqn. 81 as follows:

$$\begin{aligned} D_{\mathrm{KL}}(p_0 \parallel q_0) = \frac{1}{2}\mathbb{E}_{t\sim\mathcal{U}(0,T)}\mathbb{E}_{\mathbf{z}_t\sim p_t(\mathbf{z}_t)}\Big[&\underbrace{\Gamma\beta_t\big\|\nabla_{\mathbf{x}_t}\log p_t(\mathbf{z}_t) - \nabla_{\mathbf{x}_t}\log q_t(\mathbf{z}_t)\big\|_2^2}_{\text{Data-Space}} + \\ &\underbrace{M\nu\beta_t\big\|\nabla_{\mathbf{m}_t}\log p_t(\mathbf{z}_t) - \nabla_{\mathbf{m}_t}\log q_t(\mathbf{z}_t)\big\|_2^2}_{\text{Momentum-Space}}\Big] \end{aligned} \tag{88}$$

In general, the above score-matching loss can be re-formulated using arbitrary loss weightings $\lambda_1(t)$ and $\lambda_2(t)$ as follows:

$$D_{\mathrm{KL}}(p_0 \parallel q_0) = \frac{1}{2}\mathbb{E}_{t\sim\mathcal{U}(0,T)}\mathbb{E}_{\mathbf{z}_t\sim p_t(\mathbf{z}_t)}\Big[\lambda_1(t)\big\|\nabla_{\mathbf{x}_t}\log p_t(\mathbf{z}_t) - \nabla_{\mathbf{x}_t}\log q_t(\mathbf{z}_t)\big\|_2^2 + \lambda_2(t)\big\|\nabla_{\mathbf{m}_t}\log p_t(\mathbf{z}_t) - \nabla_{\mathbf{m}_t}\log q_t(\mathbf{z}_t)\big\|_2^2\Big] \tag{89}$$

Choosing the same weighting for both loss components i.e. $\lambda_1(t) = \lambda_2(t) = \lambda(t)$, the score-matching objective in Eqn. 88 can be simplified as follows:

$$\mathcal{L}_{\mathrm{SM}} = \frac{1}{2}\mathbb{E}_{t\sim\mathcal{U}(0,T)}\mathbb{E}_{\mathbf{z}_t\sim p_t(\mathbf{z}_t)}\Big[\lambda(t)\big\|\nabla_{\mathbf{z}_t}\log p_t(\mathbf{z}_t) - \nabla_{\mathbf{z}_t}\log q_t(\mathbf{z}_t)\big\|_2^2\Big] \tag{90}$$

Approximating the score $\nabla_{\mathbf{z}_t}\log q_t(\mathbf{z}_t)$ using a parametric estimator $\boldsymbol{s}_\theta(\mathbf{z}_t, t)$ and following Vincent [5], it can be shown that the $\mathcal{L}_{\mathrm{SM}}$ objective is equivalent to the following Denoising Score Matching (DSM) objective:

$$\mathcal{L}_{\mathrm{DSM}} = \frac{1}{2}\mathbb{E}_{t\sim\mathcal{U}(0,T)}\mathbb{E}_{\mathbf{z}_0\sim p(\mathbf{z}_0)}\mathbb{E}_{\mathbf{z}_t\sim p_t(\mathbf{z}_t|\mathbf{z}_0)}\Big[\lambda(t)\big\|\nabla_{\mathbf{z}_t}\log p_t(\mathbf{z}_t|\mathbf{z}_0) - \boldsymbol{s}_\theta(\mathbf{z}_t, t)\big\|_2^2\Big] \tag{91}$$

Moreover, Dockhorn et al. [3] propose to use the following objective a.k.a. Hybrid Score Matching (HSM), which is equivalent to the DSM objective (upto a constant independent of θ):

$$\mathcal{L}_{\mathrm{HSM}} = \frac{1}{2}\mathbb{E}_{t\sim\mathcal{U}(0,T)}\mathbb{E}_{\mathbf{x}_0\sim p(\mathbf{x}_0)}\mathbb{E}_{\mathbf{z}_t\sim p_t(\mathbf{z}_t|\mathbf{x}_0)}\Big[\lambda(t)\big\|\nabla_{\mathbf{z}_t}\log p_t(\mathbf{z}_t|\mathbf{x}_0) - \boldsymbol{s}_\theta(\mathbf{z}_t, t)\big\|_2^2\Big] \tag{92}$$

The perturbation kernels $p(\mathbf{z}_t|\mathbf{z}_0)$ and $p(\mathbf{z}_t|\mathbf{x}_0)$ can be computed analytically for an SDE with affine drift (See Appendix B.3 for the exact analytical forms of the perturbation kernel for PSLD). Following Dockhorn et al. [3], we use the Hybrid Score Matching (HSM) objective throughout this work. We next discuss the computation of the analytical form of the target score $\nabla_{\mathbf{z}_t}\log p_t(\mathbf{z}_t|\mathbf{x}_0)$ (or $\nabla_{\mathbf{z}_t}\log p_t(\mathbf{z}_t|\mathbf{z}_0)$ for DSM) and the parameterization of our score network $\boldsymbol{s}_\theta(\mathbf{z}_t, t)$.

B.2.2 Analytical Score Computation and Parameterization

In cases when the perturbation kernels are multivariate Gaussian distributions of the form $\mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$, the target score $\nabla_{\mathbf{z}_t}\log p_t(\mathbf{z}_t|\mathbf{x}_0)$ can be computed analytically as follows:

$$\nabla_{\mathbf{z}_t}\log p(\mathbf{z}_t|\mathbf{x}_0) = -\boldsymbol{\Sigma}_t^{-1}(\mathbf{z}_t - \boldsymbol{\mu}_t) \tag{93}$$

$$= -\boldsymbol{L}_t^{-T}\boldsymbol{L}_t^{-1}(\boldsymbol{L}_t\boldsymbol{\epsilon}) = -\boldsymbol{L}_t^{-T}\boldsymbol{\epsilon} \tag{94}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}_{2d}, \boldsymbol{I}_{2d})$ and $\boldsymbol{\Sigma}_t = \boldsymbol{L}_t\boldsymbol{L}_t^T$ is the Cholesky decomposition. Moreover, for $\boldsymbol{\Sigma}_t = \left(\begin{pmatrix}\Sigma_t^{xx} & \Sigma_t^{xm}\\ \Sigma_t^{xm} & \Sigma_t^{mm}\end{pmatrix}\otimes\boldsymbol{I}_d\right)$ (as is the case in PSLD), the Cholesky decomposition can be computed analytically as follows:

$$\boldsymbol{L}_t = \left(\begin{pmatrix}L_t^{xx} & 0\\ L_t^{xm} & L_t^{mm}\end{pmatrix}\otimes\boldsymbol{I}_d\right) \tag{95}$$

$$L_t = \begin{pmatrix}L_t^{xx} & 0\\ L_t^{xm} & L_t^{mm}\end{pmatrix} = \begin{pmatrix}\sqrt{\Sigma_t^{xx}} & 0\\ \frac{\Sigma_t^{xm}}{\sqrt{\Sigma_t^{xx}}} & \sqrt{\frac{\Sigma_t^{xx}\Sigma_t^{mm}-(\Sigma_t^{xm})^2}{\Sigma_t^{xx}}}\end{pmatrix} \tag{96}$$

Consequently,

$$\begin{aligned}\boldsymbol{L}_t^{-T} &= L_t^{-T}\otimes\boldsymbol{I}_d\\ &= \begin{pmatrix}\sqrt{\Sigma_t^{xx}} & \frac{\Sigma_t^{xm}}{\sqrt{\Sigma_t^{xx}}}\\ 0 & \sqrt{\frac{\Sigma_t^{xx}\Sigma_t^{mm}-(\Sigma_t^{xm})^2}{\Sigma_t^{xx}}}\end{pmatrix}^{-1}\otimes\boldsymbol{I}_d\\ &= \begin{pmatrix}\frac{1}{\sqrt{\Sigma_t^{xx}}} & \frac{-\Sigma_t^{xm}}{\sqrt{\Sigma_t^{xx}}\sqrt{\Sigma_t^{xx}\Sigma_t^{mm}-(\Sigma_t^{xm})^2}}\\ 0 & \sqrt{\frac{\Sigma_t^{xx}}{\Sigma_t^{xx}\Sigma_t^{mm}-(\Sigma_t^{xm})^2}}\end{pmatrix}\otimes\boldsymbol{I}_d.\end{aligned} \tag{97}$$

Plugging the analytical form of $\boldsymbol{L}_t^{-T}$ into Eqn. 94, we get the following analytical form of the target score $\nabla_{\mathbf{z}_t} \log p_t(\mathbf{z}_t|\mathbf{x}_0)$:

$$\nabla_{\mathbf{z}_t} \log p_t(\mathbf{z}_t|\mathbf{x}_0) = -\boldsymbol{L}_t^{-T}\boldsymbol{\epsilon} \tag{98}$$

$$= -\left(\begin{pmatrix} l_t^{xx} & l_t^{xm} \\ 0 & l_t^{mm} \end{pmatrix} \otimes \boldsymbol{I}_d\right)\begin{pmatrix} \boldsymbol{\epsilon}_x \\ \boldsymbol{\epsilon}_m \end{pmatrix} \tag{99}$$

$$= -\begin{pmatrix} l_t^{xx}\boldsymbol{\epsilon}_{\boldsymbol{x}} + l_t^{xm}\boldsymbol{\epsilon}_{\boldsymbol{m}} \\ l_t^{mm}\boldsymbol{\epsilon}_m \end{pmatrix} \tag{100}$$

where $l_t^{xx} = \frac{1}{\sqrt{\Sigma_t^{xx}}}$, $l_t^{xm} = \frac{-\Sigma_t^{xm}}{\sqrt{\Sigma_t^{xx}}\sqrt{\Sigma_t^{xx}\Sigma_t^{mm} - (\Sigma_t^{xm})^2}}$ and $l_t^{mm} = \sqrt{\frac{\Sigma_t^{xx}}{\Sigma_t^{xx}\Sigma_t^{mm} - (\Sigma_t^{xm})^2}}$. While one can directly model the score as defined in Eqn. 100, we instead parameterize the score network as $\boldsymbol{s}_\theta(\mathbf{z}_t, t) = -\boldsymbol{L}_t^{-T}\boldsymbol{\epsilon}_\theta(\mathbf{z}_t, t)$.

B.2.3 Putting it all together

Plugging the analytical form of $\nabla_{\mathbf{z}_t} \log p_t(\mathbf{z}_t|\mathbf{x}_0)$ and our score network parameterization $\boldsymbol{s}_\theta(\mathbf{z}_t, t) = -\boldsymbol{L}_t^{-T}\boldsymbol{\epsilon}_\theta(\mathbf{z}_t, t)$ in the HSM objective in Eqn. 92, we get the following objective:

$$\mathcal{L}_{\text{HSM}} = \frac{1}{2}\mathbb{E}_{t\sim\mathcal{U}(0,T)}\mathbb{E}_{\mathbf{x}_0\sim p(\mathbf{x}_0)}\mathbb{E}_{\mathbf{z}_t\sim p_t(\mathbf{z}_t|\mathbf{x}_0)}\Big[\lambda(t)\big\|\nabla_{\mathbf{z}_t}\log p_t(\mathbf{z}_t|\mathbf{x}_0) - \boldsymbol{s}_\theta(\mathbf{z}_t, t)\big\|_2^2\Big] \tag{101}$$

$$= \frac{1}{2}\mathbb{E}_{t\sim\mathcal{U}(0,T)}\mathbb{E}_{\mathbf{x}_0\sim p(\mathbf{x}_0)}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0}_{2d}, \boldsymbol{I}_{2d})}\Big[\lambda(t)\big\|\boldsymbol{L}_t^{-T}\boldsymbol{\epsilon} - \boldsymbol{L}_t^{-T}\boldsymbol{\epsilon}_\theta(\boldsymbol{\mu}_t + \boldsymbol{L}_t\boldsymbol{\epsilon}, t)\big\|_2^2\Big] \tag{102}$$

$$\leq \frac{1}{2}\mathbb{E}_{t\sim\mathcal{U}(0,T)}\mathbb{E}_{\mathbf{x}_0\sim p(\mathbf{x}_0)}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0}_{2d}, \boldsymbol{I}_{2d})}\Big[\lambda(t)\big\|\boldsymbol{L}_t^{-T}\big\|_2^2\big\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\boldsymbol{\mu}_t + \boldsymbol{L}_t\boldsymbol{\epsilon}, t)\big\|_2^2\Big] \tag{103}$$

It is worth noting that since our original HSM objective is upper bounded by the objective in Eqn. 103, minimizing the latter also minimizes $\mathcal{L}_{\text{HSM}}$. Since we optimize for sample quality, following prior work [2, 3], we choose $\lambda(t) = \frac{1}{\|\boldsymbol{L}_t^{-T}\|_2^2}$ to cancel the weighting induced by $\|\boldsymbol{L}_t^{-T}\|_2^2$. Our final training objective reduces to the following *noise-prediction* formulation:

$$\mathcal{L}(\theta) = \frac{1}{2}\mathbb{E}_{t\sim\mathcal{U}(0,T)}\mathbb{E}_{\mathbf{x}_0\sim p(\mathbf{x}_0)}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0}_{2d}, \boldsymbol{I}_{2d})}\Big[\big\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\boldsymbol{\mu}_t + \boldsymbol{L}_t\boldsymbol{\epsilon}, t)\big\|_2^2\Big] \tag{104}$$

It is worth noting that in our training setup, we need to predict the full $2d$-dimensional $\boldsymbol{\epsilon}$. This is in contrast to the training setup in CLD, where we only need to predict the last-d components i.e. $\boldsymbol{\epsilon}_{d:2d}$ of the noise vector $\boldsymbol{\epsilon}$. This difference in training arises due to different formulations of the diffusion coefficient in PSLD and CLD. Indeed, setting $\Gamma = 0$ in Eqn. 88 would result in a similar training objective as in CLD.

B.3 Perturbation Kernel in PSLD

We now present the analytical form of the perturbation kernels $p(\mathbf{z}_t|\mathbf{z}_0)$ and $p(\mathbf{z}_t|\mathbf{x}_0)$ for PSLD, which are required for training using DSM or HSM respectively.

B.3.1 Mean and Variance of $p(\mathbf{z}_t|\mathbf{z}_0)$

Since the drift and the diffusion coefficients in Eqn. 66 are affine, the perturbation kernel $p(\mathbf{z}_t|\mathbf{z}_0)$ will be a multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$. Following Särkkä and Solin [6], $\boldsymbol{\mu}_t$ and $\boldsymbol{\Sigma}_t$, evolve as the following ODEs:

$$\frac{d\boldsymbol{\mu}_t}{dt} = \boldsymbol{F}(t)\boldsymbol{\mu}_t \tag{105}$$

$$\frac{d\boldsymbol{\Sigma}_t}{dt} = \boldsymbol{F}(t)\boldsymbol{\Sigma}_t + \boldsymbol{\Sigma}_t\boldsymbol{F}^T(t) + \boldsymbol{G}(t)\boldsymbol{G}(t)^T \tag{106}$$

where $\boldsymbol{F}(t) = \left(\frac{\beta_t}{2}\begin{pmatrix} -\Gamma & M^{-1} \\ -1 & -\nu \end{pmatrix} \otimes \boldsymbol{I}_d\right)$ and $\boldsymbol{G}(t) = \left(\begin{pmatrix} \sqrt{\Gamma\beta_t} & 0 \\ 0 & \sqrt{M\nu\beta_t} \end{pmatrix} \otimes \boldsymbol{I}_d\right)$ for PSLD. Under critical damping i.e. $M^{-1} = \frac{(\Gamma-\nu)^2}{4}$, solving the ODEs for the mean and variance yields the following form of $p(\mathbf{z}_t|\mathbf{z}_0) = \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$:

$$\boldsymbol{\mu}_t = \begin{pmatrix} \boldsymbol{\mu}_t^x \\ \boldsymbol{\mu}_t^m \end{pmatrix} = \begin{pmatrix} A_1\mathcal{B}(t)\boldsymbol{x}_0 + A_2\mathcal{B}(t)\boldsymbol{m}_0 + \boldsymbol{x}_0 \\ C_1\mathcal{B}(t)\boldsymbol{x}_0 + C_2\mathcal{B}(t)\boldsymbol{m}_0 + \boldsymbol{m}_0 \end{pmatrix} e^{-(\frac{\nu+\Gamma}{4})\mathcal{B}(t)} \tag{107}$$

where $\mathcal{B}(t) = \int_0^t \beta(s)ds$ and coefficients:

$$A_1 = \frac{\nu - \Gamma}{4} \qquad A_2 = \frac{(\Gamma-\nu)^2}{8} \tag{108}$$

$$C_1 = \frac{-1}{2} \qquad C_2 = \frac{\Gamma - \nu}{4} \tag{109}$$

The variance Σ_t for the perturbation kernel $p(\mathbf{z}_t)\mathbf{z}_0$ is given by:

$$\boldsymbol{\Sigma}_t = \left(\begin{pmatrix} \Sigma_t^{xx} & \Sigma_t^{xm} \\ \Sigma_t^{xm} & \Sigma_t^{mm} \end{pmatrix} e^{-(\frac{\Gamma+\nu}{2})\mathcal{B}(t)} \right) \otimes \boldsymbol{I}_d \tag{110}$$

where,

$$\Sigma_t^{xx} = A_1\mathcal{B}^2(t)\Sigma_0^{xx} + A_2\mathcal{B}^2(t)\Sigma_0^{mm} + A_3\mathcal{B}(t)\Sigma_0^{xx} + A_4\mathcal{B}^2(t) + A_5\mathcal{B}(t) + (e^{2\lambda\mathcal{B}(t)} - 1) + \Sigma_0^{xx} \tag{111}$$

$$\Sigma_t^{xm} = C_1\mathcal{B}^2(t)\Sigma_0^{xx} + C_2\mathcal{B}^2(t)\Sigma_0^{mm} + C_3\mathcal{B}(t)\Sigma_0^{xx} + C_4\mathcal{B}(t)\Sigma_0^{mm} + C_5\mathcal{B}^2(t) \tag{112}$$

$$\Sigma_t^{mm} = D_1\mathcal{B}^2(t)\Sigma_0^{xx} + D_2\mathcal{B}^2(t)\Sigma_0^{mm} + D_3\mathcal{B}(t)\Sigma_0^{mm} + D_4\mathcal{B}^2(t) + D_5\mathcal{B}(t) + M(e^{2\lambda\mathcal{B}(t)} - 1) + \Sigma_0^{mm} \tag{113}$$

where $\boldsymbol{\Sigma}_0 = \begin{pmatrix} \Sigma_0^{xx} & 0 \\ 0 & \Sigma_0^{mm} \end{pmatrix}$, $\mathcal{B}(t) = \int_0^t \beta(s)ds$ and coefficients:

$$A_1 = \frac{M^{-1}}{4} \qquad A_2 = \frac{M^{-2}}{4} \qquad A_3 = \frac{\nu-\Gamma}{2} \qquad A_4 = \frac{-M^{-1}}{2} \qquad A_5 = \frac{\Gamma-\nu}{2} \tag{114}$$

$$C_1 = \frac{\Gamma-\nu}{8} \qquad C_2 = \frac{(\Gamma-\nu)^3}{32} \qquad C_3 = \frac{-1}{2} \qquad C_4 = \frac{M^{-1}}{2} \qquad C_5 = \frac{\nu-\Gamma}{4} \tag{115}$$

$$D_1 = \frac{1}{4} \qquad D_2 = \frac{M^{-1}}{4} \qquad D_3 = \frac{\Gamma-\nu}{2} \qquad D_4 = \frac{-1}{2} \qquad D_5 = \frac{M(\nu-\Gamma)}{2} \tag{116}$$

It is worth noting that, when $\Gamma = 0$, $\bar{\nu} = M\nu$ and $\bar{\beta}(t) = \frac{\beta(t)}{2}$ such that $\mathcal{B}(t) = 2\bar{\mathcal{B}}(t)$ where $\bar{\mathcal{B}}(t) = \int_0^t \bar{\beta}(s)ds$, we have the following form of the mean μ_t:

$$\mu_t = \begin{pmatrix} 2\bar{\nu}^{-2}\bar{\mathcal{B}}(t)\boldsymbol{x}_0 + 4\bar{\nu}^{-2}\bar{\mathcal{B}}(t)\boldsymbol{m}_0 + \boldsymbol{x}_0 \\ -\bar{\mathcal{B}}(t)\boldsymbol{x}_0 - 2\bar{\nu}^{-1}\bar{\mathcal{B}}(t)\boldsymbol{m}_0 + \boldsymbol{m}_0 \end{pmatrix} e^{-2\bar{\nu}^{-1}\bar{\mathcal{B}}(t)} \tag{117}$$

The expression for μ_t in Eqn. 117 is exactly the same as the mean of the perturbation kernel for CLD (Refer to Appendix B.1 in Dockhorn et al. [3]). A similar analysis holds for the variance Σ_t which provides more insight into CLD being a special case of PSLD. Similar to CLD, at $t = 0$, we have $\boldsymbol{\mu}_0 = \begin{pmatrix} \mathbf{x}_0 \\ \mathbf{m}_0 \end{pmatrix}$, where $\mathbf{x}_0 \sim p(\mathbf{x}_0)$ (a.k.a the data generating distribution) and $\mathbf{m}_0 \sim \mathcal{N}(\mathbf{0}_d, M\gamma\boldsymbol{I}_d)$, where γ is a scalar hyperparameter. Similarly, for DSM training, both Σ_0^{xx} and Σ_0^{mm} can be set to 0 (since $\mathbf{z}_t = [\mathbf{x}_t, \mathbf{m}_t]^T$ is a sample based estimate).

B.3.2 Mean and Variance of $p(\mathbf{z}_t|\mathbf{x}_0)$

Since the data generating distribution for m_0 and the DSM perturbation kernel $p(\mathbf{z}_t|\mathbf{z}_0)$ are multivariate Gaussians, we can marginalize out the initial momentum variables $\mathbf{m}_0$ from $p(\mathbf{z}_t|\mathbf{z}_0)$ to obtain the perturbation kernel for HSM as $p(\mathbf{z}_t|\mathbf{x}_0) = \int p(\mathbf{z}_t|\mathbf{x}_0, \mathbf{m}_0)p(\mathbf{m}_0)d\mathbf{m}_0$. Consequently, the perturbation kernel $p(\mathbf{z}_t|\mathbf{x}_0)$ can be obtained by setting $m_0 = \mathbf{0_d}$ and $\Sigma_0^{mm} = M\gamma$ in the expressions of $\boldsymbol{\mu}_t$ and $\boldsymbol{\Sigma}_t$ for $p(\mathbf{z}_t|\mathbf{z}_0)$.

B.3.3 Convergence

As $t \to \infty$, the mean μ_t converges to $\mathbf{0}_{2d}$ since the multiplicative term $e^{-(\frac{\nu+\Gamma}{4})\mathcal{B}(t)}$ goes to 0. Similarly, the covariance for the perturbation kernel converges to the following case:

$$\Sigma_e^{xx} = \lim_{t\to\infty} \Sigma_t^{xx} e^{-(\frac{\Gamma+\nu}{2})\mathcal{B}(t)} = 1 \tag{118}$$

$$\Sigma_e^{xm} = \lim_{t\to\infty} \Sigma_t^{xm} e^{-(\frac{\Gamma+\nu}{2})\mathcal{B}(t)} = 0 \tag{119}$$

$$\Sigma_e^{mm} = \lim_{t\to\infty} \Sigma_t^{mm} e^{-(\frac{\Gamma+\nu}{2})\mathcal{B}(t)} = M \tag{120}$$

Therefore, the perturbation kernel converges to the following steady-state distribution $p_{\text{EQ}}(\mathbf{z}) = \mathcal{N}(\mathbf{x}; \mathbf{0}_d, \boldsymbol{I}_d)\mathcal{N}(\mathbf{m}; \mathbf{0}_d, M\boldsymbol{I}_d)$. It is worth noting that this is the exact equilibrium distribution that we specified in our SGM recipe to construct the forward process for PSLD.

B.4 PSLD Sampling

The reverse SDE analogous to the forward SDE defined in Eqn. 66 can be formulated as follows [2]:

$$d\bar{\mathbf{z}}_t = \bar{\boldsymbol{f}}(\bar{\mathbf{z}}_t)dt + \boldsymbol{G}(T-t)d\bar{\mathbf{w}}_t \tag{121}$$

$$\bar{\boldsymbol{f}}(\bar{\mathbf{z}}_t) = \frac{\beta_t}{2}\begin{pmatrix}\Gamma\bar{\mathbf{x}}_t - M^{-1}\bar{\mathbf{m}}_t + 2\Gamma \boldsymbol{s}_\theta(\bar{\mathbf{z}}_t, T-t)|_{0:d} \\ \bar{\mathbf{x}}_t + \nu\bar{\mathbf{m}}_t + 2M\nu \boldsymbol{s}_\theta(\bar{\mathbf{z}}_t, T-t)|_{d:2d}\end{pmatrix}, \qquad \boldsymbol{G}(T-t) = \begin{pmatrix}\sqrt{\Gamma\beta_t} & 0 \\ 0 & \sqrt{M\nu\beta_t}\end{pmatrix} \otimes \boldsymbol{I}_d \tag{122}$$

where $\bar{\mathbf{z}}_t = \mathbf{z}_{T-t}$, $\bar{\mathbf{x}}_t = \mathbf{x}_{T-t}$, $\bar{\mathbf{m}}_t = \mathbf{m}_{T-t}$. Given an estimate of the score $\boldsymbol{s_\theta}(\mathbf{z}_t, T-t)$, one can simulate the above SDE to generate data from noise. Given $\bar{\mathbf{z}}_0 = \left(\bar{\mathbf{x}}_0, \bar{\mathbf{m}}_0\right)^T \sim p_{\text{EQ}}(\mathbf{z}) = \mathcal{N}(\mathbf{x}; \mathbf{0}_d, \boldsymbol{I}_d)\mathcal{N}(\mathbf{m}; \mathbf{0}_d, M\boldsymbol{I}_d)$, we now discuss update steps for different samplers in context of PSLD.

B.4.1 Euler-Maruyama (EM) Sampler

The EM update step for the reverse SDE corresponding to PSLD are as follows:

$$\begin{pmatrix}\bar{\mathbf{x}}_{t'} \\ \bar{\mathbf{m}}_{t'}\end{pmatrix} = \begin{pmatrix}\bar{\mathbf{x}}_t \\ \bar{\mathbf{m}}_t\end{pmatrix} + \frac{\beta_t\delta t}{2}\begin{pmatrix}\Gamma\bar{\mathbf{x}}_t - M^{-1}\bar{\mathbf{m}}_t + 2\Gamma \boldsymbol{s}_\theta(\bar{\mathbf{z}}_t, T-t)|_{0:d} \\ \bar{\mathbf{x}}_t + \nu\bar{\mathbf{m}}_t + 2M\nu \boldsymbol{s}_\theta(\bar{\mathbf{z}}_t, T-t)|_{d:2d}\end{pmatrix} + \begin{pmatrix}\sqrt{\Gamma\beta_t\delta t}\boldsymbol{\epsilon}_{t'}^x \\ \sqrt{M\nu\beta_t\delta t}\boldsymbol{\epsilon}_{t'}^m\end{pmatrix} \tag{123}$$

where $\boldsymbol{\epsilon}_t = [\boldsymbol{\epsilon}_t^x, \boldsymbol{\epsilon}_t^m]^T \sim \mathcal{N}(\mathbf{0}_{2d}, \boldsymbol{I}_{2d})$ and $t' = t + \delta t$ where δt is the step size for a single update.

B.4.2 Symmetric Splitting CLD Sampler (SSCS)

Inspired by the application of splitting-based integrators in molecular dynamics [7], Dockhorn et al. [3] proposed the SSCS sampler with the following symmetric splitting scheme:

$$\begin{pmatrix}d\bar{\mathbf{x}}_t \\ d\bar{\mathbf{m}}_t\end{pmatrix} = \underbrace{\frac{\beta_t}{2}\begin{pmatrix}-\Gamma\bar{\mathbf{x}}_t - M^{-1}\bar{\mathbf{m}}_t \\ \bar{\mathbf{x}}_t - \nu\bar{\mathbf{m}}_t\end{pmatrix}dt + \boldsymbol{G}(T-t)d\bar{\mathbf{w}}_t}_{A} + \underbrace{\beta_t\begin{pmatrix}\Gamma\bar{\mathbf{x}}_t + \Gamma \boldsymbol{s}_\theta(\bar{\mathbf{z}}_t, T-t)|_{0:d} \\ \nu\bar{\mathbf{m}}_t + M\nu \boldsymbol{s}_\theta(\bar{\mathbf{z}}_t, T-t)|_{d:2d}\end{pmatrix}dt}_{S} \tag{124}$$

Dockhorn et al. [3] then approximate the flow map for the original SDE by the application of the following symmetric splitting schedule [8, 9]:

$$e^{t(\mathcal{L}_A+\mathcal{L}_S)} \approx \left[e^{\frac{\delta t}{2}\mathcal{L}_A^*}e^{\delta t\mathcal{L}_S^*}e^{\frac{\delta t}{2}\mathcal{L}_A^*}\right]^N + \mathcal{O}(N\delta t^3) \tag{125}$$

where $N = \frac{t}{\delta t}$. The solution $\bar{\mathbf{z}}_t$ for the reverse SDE at any time t can then be obtained by the application of the flow map approximation of $e^{t(\mathcal{L}_A+\mathcal{L}_S)}$ to $\bar{\mathbf{z}}_0$. Since we use the same splitting formulation as Dockhorn et al. [3], the modified SSCS sampler for PSLD is still a first-order integrator sampler (Also see Appendix D in Dockhorn et al. [3] for more analysis of the SSCS sampler as proposed for CLD). However, since the form of the analytical splitting component in Eqn. 124 is different from CLD (due to a non-zero Γ), we next discuss the solution for this analytical form.

Analytical splitting-term: We have the following analytical splitting term:

$$\begin{pmatrix} d\bar{\mathbf{x}}_t \\ d\bar{\mathbf{m}}_t \end{pmatrix} = \frac{\beta_t}{2}\begin{pmatrix} -\Gamma\bar{\mathbf{x}}_t - M^{-1}\bar{\mathbf{m}}_t \\ \bar{\mathbf{x}}_t - \nu\bar{\mathbf{m}}_t \end{pmatrix} dt + \left(\begin{pmatrix} \sqrt{\Gamma\beta_t} & 0 \\ 0 & \sqrt{M\nu\beta_t} \end{pmatrix} \otimes \boldsymbol{I}_d\right) d\bar{\mathbf{w}}_t \tag{126}$$

The solution for this analytical SDE is similar to the derivation of the perturbation kernel in Appendix B.3. However, there are two key differences. Firstly, we need to integrate between time-intervals $(t, t+\delta t)$ as opposed to from $(0, t)$ for the perturbation kernel. Secondly, since we are sampling, we set the initial covariances Σ_{xx}^t and Σ_{mm}^t to zero. The analytical solution for the SDE in Eqn. 126 can then be specified as follows:

$$\bar{\mathbf{z}}_t \sim \mathcal{N}(\bar{\boldsymbol{\mu}}_t, \bar{\boldsymbol{\Sigma}}_t) \tag{127}$$

where

$$\boldsymbol{\mu}(\bar{\mathbf{x}}_t, \bar{\mathbf{m}}_t, t, t') = \begin{pmatrix} A_1\mathcal{B}(t,t')\bar{\boldsymbol{x}}_t + A_2\mathcal{B}(t,t')\bar{\boldsymbol{m}}_t + \bar{\boldsymbol{x}}_t \\ C_1\mathcal{B}(t,t')\bar{\boldsymbol{x}}_t + C_2\mathcal{B}(t,t')\bar{\boldsymbol{m}}_t + \bar{\boldsymbol{m}}_t \end{pmatrix} e^{-(\frac{\nu+\Gamma}{4})\mathcal{B}(t,t')} \tag{128}$$

$$A_1 = \frac{\nu - \Gamma}{4} \qquad A_2 = \frac{-(\Gamma-\nu)^2}{8} \tag{129}$$

$$C_1 = \frac{1}{2} \qquad C_2 = \frac{\Gamma - \nu}{4} \tag{130}$$

The solution for the covariance is given by the following expression:

$$\boldsymbol{\Sigma}(t,t') = \left(\begin{pmatrix} \Sigma_{t,t'}^{xx} & \Sigma_{t,t'}^{xm} \\ \Sigma_{t,t'}^{xm} & \Sigma_{t,t'}^{mm} \end{pmatrix} e^{-(\frac{\Gamma+\nu}{2})\mathcal{B}(t,t')}\right) \otimes \boldsymbol{I}_d \tag{131}$$

where,

$$\Sigma_t^{xx} = -\frac{(\Gamma-\nu)^2}{8}\mathcal{B}^2(t,t') + \frac{(\Gamma-\nu)}{2}\mathcal{B}(t,t') + (e^{-(\frac{\Gamma+\nu}{2})\mathcal{B}(t,t')} - 1) \tag{132}$$

$$\Sigma_t^{xm} = \frac{(\Gamma-\nu)}{4}\mathcal{B}^2(t,t') \tag{133}$$

$$\Sigma_t^{mm} = -\frac{1}{2}\mathcal{B}^2(t,t') + \frac{M(\Gamma-\nu)}{2}\mathcal{B}(t,t') + M(e^{-(\frac{\Gamma+\nu}{2})\mathcal{B}(t,t')} - 1) \tag{134}$$

where $\mathcal{B}(t,t') = -\int_t^{t'} \beta(s)ds$ and $t' = t + \delta t$. Indeed, setting $\Gamma = 0$ recovers the original SSCS algorithm proposed in Dockhorn et al. [3]. Therefore, given $\bar{\mathbf{z}}_t = \left(\bar{\mathbf{x}}_t, \bar{\mathbf{m}}_t\right)^T$, the flow map update for the analytical splitting term $e^{\frac{\delta t}{2}\mathcal{L}_A^*}$ is given by:

$$\bar{\mathbf{z}}_{t'} \sim \mathcal{N}(\boldsymbol{\mu}(\bar{\mathbf{x}}_t, \bar{\mathbf{m}}_t, t, t'), \boldsymbol{\Sigma}(t,t')) \tag{135}$$

where $t' = t + \frac{\delta t}{2}$.

Score-based splitting term: The flow map update for the score-based splitting term $e^{\delta t \mathcal{L}_S^*}$ is given by an Euler update as follows:

$$\begin{pmatrix} \bar{\mathbf{x}}_{t'} \\ \bar{\mathbf{m}}_{t'} \end{pmatrix} = \begin{pmatrix} \bar{\mathbf{x}}_t \\ \bar{\mathbf{m}}_t \end{pmatrix} + \delta t\beta_t \begin{pmatrix} \Gamma\bar{\mathbf{x}}_t + \Gamma\boldsymbol{s}_\theta(\bar{\mathbf{z}}_t, T-t)|_{0:d} \\ \nu\bar{\mathbf{m}}_t + M\nu\boldsymbol{s}_\theta(\bar{\mathbf{z}}_t, T-t)|_{d:2d} \end{pmatrix} \tag{136}$$

Combining the two splitting terms together, a more generic form of the SSCS algorithm can be specified as follows:

Algorithm 1 *Modified SSCS* (Terms in blue indicate differences from the SSCS sampler proposed in Dockhorn et al. [3])

```
Input: Trajectory length T, Score function $\boldsymbol{s_\theta}(\mathbf{z}_t, T-t)$, PSLD parameters $\Gamma$, $\nu$, $\beta_t$, $M = \frac{(\Gamma-\nu)^2}{4}$, number
of sampling steps $N$, step sizes $\{\delta t_n \geq 0\}_{n=0}^{N-1}$ spanning the interval (0, $T-\epsilon$).
Output: $\bar{\mathbf{z}}_T$ = ($\bar{\mathbf{x}}_T$, $\bar{\mathbf{m}}_T$)

$\bar{\mathbf{x}}_0 \sim \mathcal{N}(\mathbf{0}_d, \boldsymbol{I}_d)$, $\bar{\mathbf{m}}_0 \sim \mathcal{N}(\mathbf{0}_d, M\boldsymbol{I}_d)$, $\bar{\mathbf{z}}_0 = (\bar{\mathbf{x}}_0, \bar{\mathbf{m}}_0)$     ▷ Draw initial prior samples from $p_{\mathrm{EQ}}(\mathbf{u})$
$t = 0$     ▷ Initialize time
for $n = 0$ to $N-1$ do
    $\bar{\mathbf{z}}_{n+\frac{1}{2}} \sim \mathcal{N}(\boldsymbol{\mu}(\bar{\mathbf{x}}_n, \bar{\mathbf{m}}_n, t, t+\frac{\delta t_n}{2}), \boldsymbol{\Sigma}(t, t+\frac{\delta t_n}{2}))$     ▷ First half-step: Apply $\exp\{\frac{\delta t_n}{2}\hat{\mathcal{L}}_A^*\}$
    $\bar{\mathbf{z}}_{n+\frac{1}{2}} \leftarrow \bar{\mathbf{z}}_{n+\frac{1}{2}} + \delta t_n \beta_t \begin{pmatrix} \Gamma\bar{\mathbf{x}}_{n+\frac{1}{2}} + \Gamma\boldsymbol{s}_\theta(\bar{\mathbf{z}}_{n+\frac{1}{2}}, T-t)|_{0:d} \\ \nu\bar{\mathbf{m}}_{n+\frac{1}{2}} + M\nu\boldsymbol{s}_\theta(\bar{\mathbf{z}}_{n+\frac{1}{2}}, T-t)|_{d:2d} \end{pmatrix}$     ▷ Full step: Apply $\exp\{\delta t_n \hat{\mathcal{L}}_S^*\}$
    $\bar{\mathbf{z}}_{n+1} \sim \mathcal{N}(\boldsymbol{\mu}(\bar{\mathbf{x}}_{n+\frac{1}{2}}, \bar{\mathbf{m}}_{n+\frac{1}{2}}, t, t+\frac{\delta t_n}{2}), \boldsymbol{\Sigma}(t, t+\frac{\delta t_n}{2}))$     ▷ Second half-step: Apply $\exp\{\frac{\delta t_n}{2}\hat{\mathcal{L}}_A^*\}$
    $t \leftarrow t + \delta t_n$     ▷ Update time
end for
$\bar{\mathbf{z}}_N \leftarrow \bar{\mathbf{z}}_N + \epsilon\frac{\beta_t}{2} \begin{pmatrix} \Gamma\bar{\mathbf{x}}_{n+1} - M^{-1}\bar{\mathbf{m}}_{n+1} + 2\Gamma\boldsymbol{s}_\theta(\bar{\mathbf{z}}_{n+1}, \epsilon)|_{0:d} \\ \bar{\mathbf{x}}_{n+1} + \nu\bar{\mathbf{m}}_{n+1} + 2M\nu\boldsymbol{s}_\theta(\bar{\mathbf{z}}_{n+1}, \epsilon)|_{d:2d} \end{pmatrix}$     ▷ Denoising
$(\bar{\mathbf{x}}_N, \bar{\mathbf{m}}_N) = \bar{\mathbf{z}}_N$     ▷ Extract output data and momentum samples
```

B.4.3 Probability Flow ODE

Following Song et al. [2], the probability flow ODE for PSLD can be specified as follows:

$$d\bar{\mathbf{z}}_t = \bar{\boldsymbol{f}}(\bar{\mathbf{z}}_t)dt \tag{137}$$

$$\bar{\boldsymbol{f}}(\bar{\mathbf{z}}_t) = \frac{\beta_t}{2}\begin{pmatrix} \Gamma\bar{\mathbf{x}}_t - M^{-1}\bar{\mathbf{m}}_t + \Gamma\boldsymbol{s}_\theta(\bar{\mathbf{z}}_t, T-t)|_{0:d} \\ \bar{\mathbf{x}}_t + \nu\bar{\mathbf{m}}_t + M\nu\boldsymbol{s}_\theta(\bar{\mathbf{z}}_t, T-t)|_{d:2d} \end{pmatrix} \tag{138}$$

The Probability-Flow ODE can be solved using any fixed/adaptive step-size black-box ODE solvers like RK45 [10]

C Implementation Details

C.1 Datasets and Preprocessing

We use CIFAR-10 [11] (50k images) and CelebA-64 ($\approx$ 200k images) [12] datasets for both quantitative and qualitative analysis. We use the AFHQv2 [13] dataset ($\approx$ 14k images) only for qualitative analysis. Unless specified otherwise, we always use the CelebA dataset at 64x64 resolution and the AFHQv2 dataset at 128 x 128 resolution. During training, all datasets are preprocessed to a numerical range of [-1, 1]. Following prior work, we use random horizontal flips to train all models (ablation and SOTA) across datasets as a data augmentation strategy.

C.2 Score Network Architecture

Table 7 illustrates our score model architectures for different datasets. Our network architectures are largely based on the design of the DDPM++/NCSN++ score networks introduced in Song et al. [2]. Apart from minor design choices, the DDPM++/NCSN++ score-network architectures are primarily based on the U-Net [14] model. We further highlight several key aspects of our score network architectures across different datasets as follows:

CIFAR-10: We use a smaller version (39M) of the DDPM++ architecture (with the number of residual blocks per resolution set to two) for ablation studies (for both VP-SDE and PSLD) while we use the NCSN++ architecture [2] for training larger models used for SOTA comparisons. Moreover, when training larger models, like Karras et al. [15] we remove the layers at 4x4 resolution and re-distribute capacity to the layers at the 16x16 resolution. This results in model sizes of 55M/97M parameters corresponding to four and eight residual blocks per resolution, respectively, with channel multipliers [2,2,2]. Moreover, when training larger models (55M/97M), we slightly increase the dropout rate from 0.1 to 0.15. We observed that these changes improved performance slightly while reducing model sizes and enabling faster training.

	CIFAR-10		CelebA-64		AFHQv2
Hyperparameter	SOTA	Ablation	SOTA	Ablation	Qualitative
Base channels	128	128	128	128	128
Channel multiplier	[2,2,2]	[1,2,2,2]	[1,2,2,2]	[1,1,2,2,2]	[1,2,2,2,3]
# Residual blocks	4,8	2	4	4	2
Non-Linearity	Swish	Swish	Swish	Swish	Swish
Attention resolution	[16]	[16]	[16]	[16]	[16]
# Attention heads	1	1	1	1	1
Dropout	0.15	0.1	0.1	0.1	0.2
Finite Impulse Response (FIR) [16]	True	False	True	False	False
FIR kernel	[1,3,3,1]	N/A	[1,3,31]	N/A	N/A
Progressive Input	Residual	None	Residual	None	None
Progressive Combine	Sum	Sum	Sum	Sum	Sum
Embedding type	Fourier	Positional	Fourier	Positional	Positional
Sigma scaling	False	False	False	False	False
Model size	55M/97M	39M	62M	66M	68M

Table 7: Score Network hyperparameters for PSLD.

CelebA-64: Similar to CIFAR-10, we use a DDPM++ score model architecture for ablation experiments. while we use a NCSN++ architecture for SOTA comparisons. Moreover, we remove the 4x4 layers from our ablation model for SOTA analysis and increase the channel multiplier for the 32x32 layers from 1 to 2. The dropout rate is set to 0.1 due to a larger dataset size for CelebA-64. This setting results in a model size of approximately 66M for the ablation experiments and 62M for SOTA comparisons.

AFHQv2: We use the original DDPM++ architecture for training our AFHQv2 model. Additionally, we increase the dropout rate to 0.2, given a relatively smaller dataset size. This setting results in a model size of approximately 68M parameters for qualitative analysis.

C.3 SDE Parameters

PSLD: For PSLD (including the CLD baseline), unless specified otherwise, we set the mass parameter $M^{-1} = 4$ and $\beta = 8$. The choice of these parameters is motivated by empirical results presented in Dockhorn et al. [3]. We add a stabilizing numerical epsilon value of $1e^{-9}$ in the diagonal entries of the Cholesky decomposition of $\mathbf{\Sigma}_t$ when sampling the perturbed data-point $\mathbf{z}_t \sim \mathcal{N}(\boldsymbol{\mu}_t, \mathbf{\Sigma}_t)$ during training. The data-generating distribution is set to $p_0(\mathbf{z}) = \mathcal{N}(\mathbf{0}_d, \boldsymbol{I}_d)\mathcal{N}(\mathbf{0}_d, M\gamma\boldsymbol{I}_d)$ where $\gamma = 0.04$. For SOTA analysis, we experiment with $\Gamma \in \{0.01, 0.02\}$ for CIFAR-10 and $\Gamma = 0.005$ for the CelebA-64 datasets. We chose these values of Γ and ν based on the best-performing (in terms of FID) ablation models for these datasets (See Table 4 in the main text). Lastly, for training our AFHQv2 model for qualitative analysis, we set $\Gamma = 0.01$. All the other SDE parameters remain the same.

VP-SDE: For our VP-SDE baseline, following Song et al. [2], we set $\beta_{\text{min}} = 0.1$ and $\beta_{\text{max}} = 20.0$

C.4 Training

Table 8 summarizes the different training hyperparameters across datasets and evaluation settings (ablation and SOTA). Additionally, we use the Hybrid Score Matching (HSM) objective (See Appendix B.2.1) for all augmented state-space models (PSLD and CLD); for the VP-SDE baseline, we use the Denoising Score Matching (DSM) objective. Throughout this work, we optimize for sample quality and thus use the *epsilon-prediction* loss during training (See Appendix B.2.1).

C.5 Evaluation

SDE Sampling: As is common in prior works [2, 3], we use the integration interval $(1e^{-3}, 1.0)$ for solving the reverse SDE/ODE for sample generation. Unless specified otherwise, we use 1000 sampling steps when using numerical SDE solvers. When using numerical black-box ODE solvers, we use the RK-45 [10] solver with the same absolute and relative tolerance levels. We use the ODE

	CIFAR-10		CelebA-64		AFHQv2
	SOTA	Ablation	SOTA	Ablation	Qualitative
Random Seed	0	0	0	0	0
# iterations	800k	800k	800k	320k	400k
Optimizer	Adam	Adam	Adam	Adam	Adam
Grad Clip. cutoff	1.0	1.0	1.0	1.0	1.0
Learning rate (LR)	2e-4	2e-4	2e-4	2e-4	1e-4
LR Warmup steps	5000	5000	5000	5000	5000
FP16	False	False	False	False	False
EMA Rate	0.9999	0.9999	0.9999	0.9999	0.9999
Effective Batch size	128	128	128	128	64
# GPUs	8	4	8	4	8
Train eps cutoff	1e-5	1e-5	1e-5	1e-5	1e-5

Table 8: Training hyperparameters for PSLD

		CIFAR-10	AFHQ-v2
Training	Random Seed	0	0
	# iterations	200k	70k
	Optimizer	Adam	Adam
	LR	2e-4	2e-4
	Warmup steps	5000	5000
	FP16	False	False
	Batch size	256	64
	# GPUs	4	4
	Train eps cutoff	1e-5	1e-5
SDE	M^{-1}	4.0	4.0
	Γ	0.01	0.01
	ν	4.01	4.01
	β	8.0	8.0

Table 9: Classifier Training Hyperparameters

	CIFAR-10	AFHQ-v2
Base channels	128	128
Num. classes	10	3
Channel multiplier	[1,2,3,4]	[1,2,3,4]
# Residual blocks	4	4
Non-Linearity	Swish	Swish
Attention resolution	[16, 8]	[16, 8]
# Attention heads	1	1
Dropout	0.1	0.1
FIR [16]	False	False
Progressive Input	None	None
Progressive Combine	Sum	Sum
Embedding type	Positional	Positional
Sigma scaling	False	False
Model size	56.7M	57.8M

Table 10: Classifier Network Hyperparameters

solver at different tolerance levels for ablations studies (Table 6 in the main text) and tolerance levels of $1e^{-5}$ and $1e^{-4}$ for reporting SOTA results on CIFAR-10. Similarly, we use a tolerance level of $1e^{-5}$ for reporting ODE solver performance on CelebA-64. We use the `odeint` function from the `torchdiffeq`[17] package when using the black-box ODE solver for sampling.

Timestep Selection during Sampling: We use *Uniform* (US) and *Quadratic* (QS) striding for timestep discretization in this work. In uniform striding, given an NFE budget N, we discretize the integration interval (ϵ, T) into N equidistant parts, which are then used for score function evaluations. In quadratic striding [3, 18], the evaluation timepoints are given by:

$$\tau_i = \left(\frac{i}{N}\right)^2 \quad \forall i \in [0, N) \tag{139}$$

This ensures more number of score function evaluations in the lower timestep regime (i.e. t, which is close to the data). This kind of timestep selection is particularly useful when the NFE budget is limited (See Table 5).

Last-Step Denoising: Similar to prior works [2, 3, 19], we perform a single denoising EM step (without noise injection) at the very last step of our sampling routine for both SDE and ODE solvers. Formally, we perform the following update:

$$\begin{pmatrix}\mathbf{x}_0 \\ \mathbf{m}_0\end{pmatrix} = \begin{pmatrix}\mathbf{x}_\epsilon \\ \mathbf{m}_\epsilon\end{pmatrix} + \frac{\beta_t \epsilon}{2}\begin{pmatrix}\Gamma \mathbf{x}_\epsilon - M^{-1}\mathbf{m}_\epsilon + 2\Gamma \boldsymbol{s}_\theta(\mathbf{z}_\epsilon, \epsilon)|_{0:d} \\ \mathbf{x}_\epsilon + \nu \mathbf{m}_\epsilon + 2M\nu \boldsymbol{s}_\theta(\mathbf{z}_\epsilon, \epsilon)|_{d:2d}\end{pmatrix} \tag{140}$$

where $\epsilon = 1e^{-3}$. Such a denoising step has been found useful in removing additional noise, thereby improving FID scores [19].

Γ	ν	$M^{-1} = \frac{(\Gamma-\nu)^2}{4}$	FID@50k $\downarrow$ (EM-QS)
0	4	4	3.64
0.005	4.005	4	3.42
0.01	4.01	4	3.15
0.02	4.02	4	3.26
0.25	4.25	4	4.99
4	0	4	11.43
8	4	4	14.15

Table 11: Extended results for impact of the choice of Γ on sample quality for CIFAR-10. FID (lower is better) reported on 50k samples.

Evaluation Metrics: For most ablation experiments involving the analysis of speed vs sample quality trade-offs between different models, we report the FID [20] score on 10k samples for computational convenience. For SOTA comparisons, we report FID for 50k samples for both CIFAR-10 and CelebA-64 datasets. When reporting extended SOTA results for CIFAR-10, we also report the Inception Score (IS) [21] metric. We use the `torch-fidelity`[22] package for computing all FID and IS scores reported in this work. When reporting average NFE (number of function evaluations) in Table 6, we average the NFE values over a batch size of 16 samples for 10k samples in aggregate and take a ceiling of the resulting value.

C.6 Classifier Architecture and Training

For class conditional synthesis (Appendix D.4), we append the downsampling part of the UNet architecture with a classification head and use the resulting model as our classifier architecture. Table 10 shows different hyperparameters of our classifier model architecture. For classifier training, we set $\Gamma = 0.01$ for the AFHQv2 and the CIFAR-10 datasets. The remaining SDE parameters remain unchanged from our previous setting. Table 9 lists different hyperparameters for classifier training.

D Additional Results

D.1 Impact of Γ and ν on PSLD Sample Quality

Table 11 shows the impact of varying Γ and ν (with a fixed M^{-1}) on the PSLD sample quality using the EM-sampler with quadratic striding (EM-QS) for the CIFAR-10 dataset. Extending Table 4, we additionally present FID scores for $\Gamma \in \{4.0, 8.0\}$ in Table 11. As we increase the value of Γ to 4.0 or 8.0, the FID scores further increase to 11.43 and 14.15, respectively, confirming our observations in Section 4.2. Figure 6 further illustrates the qualitative impact of increasing Γ on CIFAR-10 sample quality. For the setting with $\Gamma = 8.0$, using EM with uniform striding (EM-US) introduces evident noise artifacts in the generated samples. However, such artifacts are less pronounced when using EM with quadratic striding ((EM-QS)) instead. This suggests potential denoising problems for low timestep indices during sampling as quadratic striding focuses more score network evaluations in the low timestep regime, which might lead to lesser artifacts. This is similar to our observations in Section 4.2 (See Figure 7 for more qualitative results on CelebA-64). We now provide a formal justification for these observations.

Given an input sample $\bar{\mathbf{z}}_t = (\bar{\mathbf{x}}_t, \bar{\mathbf{m}}_t)$ at time t, consider the following update rule for the EM-sampler for PSLD with a uniform spacing interval of δt between successive steps:

$$\begin{pmatrix}\bar{\mathbf{x}}_{t'}\\ \bar{\mathbf{m}}_{t'}\end{pmatrix} = \begin{pmatrix}\bar{\mathbf{x}}_{t}\\ \bar{\mathbf{m}}_{t}\end{pmatrix} + \frac{\beta\delta t}{2}\begin{pmatrix}\Gamma\bar{\mathbf{x}}_t - M^{-1}\bar{\mathbf{m}}_t + 2\Gamma \boldsymbol{s}_\theta(\bar{\mathbf{z}}_t, T-t)|_{0:d}\\ \bar{\mathbf{x}}_t + \nu\bar{\mathbf{m}}_t + 2M\nu \boldsymbol{s}_\theta(\bar{\mathbf{z}}_t, T-t)|_{d:2d}\end{pmatrix} + \begin{pmatrix}\sqrt{\Gamma\beta\delta t}\boldsymbol{\epsilon}^x_{t'}\\ \sqrt{M\nu\beta\delta t}\boldsymbol{\epsilon}^m_{t'}\end{pmatrix} \tag{141}$$

To simplify notation, let us denote $\bar{\beta} = \frac{\beta\delta t}{2}$, $\boldsymbol{s}_\theta^x(\bar{\mathbf{z}}_t) = \boldsymbol{s}_\theta(\bar{\mathbf{z}}_t, T-t)|_{0:d}$ and $\boldsymbol{s}_\theta^m(\bar{\mathbf{z}}_t) = \boldsymbol{s}_\theta(\bar{\mathbf{z}}_t, T-t)|_{d:2d}$. Therefore, for the next timestep t', we have:

$$\bar{\mathbf{x}}_{t'} = \bar{\mathbf{x}}_t + \bar{\beta}\Big(\Gamma\bar{\mathbf{x}}_t - M^{-1}\bar{\mathbf{m}}_t + 2\Gamma\boldsymbol{s}_\theta^x(\bar{\mathbf{z}}_t)\Big) + \sqrt{\Gamma\beta\delta t}\boldsymbol{\epsilon}_{t'}^x \tag{142}$$

$$\bar{\mathbf{m}}_{t'} = \bar{\mathbf{m}}_t + \bar{\beta}\Big(\bar{\mathbf{x}}_t + \nu\bar{\mathbf{m}}_t + 2M\nu\boldsymbol{s}_\theta^m(\bar{\mathbf{z}}_t)\Big) + \sqrt{M\nu\beta\delta t}\boldsymbol{\epsilon}_{t'}^m \tag{143}$$

Similarly for the next consecutive time-step $t^{''}$, we have the following EM-update rule:

$$\bar{\mathbf{x}}_{t''} = \bar{\mathbf{x}}_{t'} + \bar{\beta}\Big(\Gamma\bar{\mathbf{x}}_{t'} - M^{-1}\bar{\mathbf{m}}_{t'} + 2\Gamma\boldsymbol{s}_\theta^x(\bar{\mathbf{z}}_{t'})\Big) + \sqrt{\Gamma\beta\delta t}\boldsymbol{\epsilon}_{t''}^x \tag{144}$$

Substituting the update expressions for $\bar{\mathbf{x}}_{t'}$ and $\bar{\mathbf{m}}_{t'}$ from Eqns. 142-143 in the update rule for $\bar{\mathbf{x}}_{t''}$, we have the following modified update rule for $\bar{\mathbf{x}}_{t''}$:

$$\bar{\mathbf{x}}_{t''} = \boldsymbol{f}(\bar{\mathbf{x}}_t, \bar{\mathbf{m}}_t) + \hat{\boldsymbol{s}}_\theta + \boldsymbol{\eta} \tag{145}$$

where $\boldsymbol{f}$ is a function of $(\bar{\mathbf{x}}_t, \bar{\mathbf{m}}_t)$, $\boldsymbol{\eta}$ is the aggregate stochastic noise. More importantly, the score term $\hat{\boldsymbol{s}}_\theta$ is given as follows:

$$\hat{\boldsymbol{s}}_\theta = 2\bar{\beta}\Gamma\boldsymbol{s}_\theta^x(\bar{\mathbf{z}}_t) + 2\bar{\beta}^2\Big[\Gamma^2\boldsymbol{s}_\theta^x(\bar{\mathbf{z}}_t) - \nu\boldsymbol{s}_\theta^m(\bar{\mathbf{z}}_t)\Big] + 2\Gamma\bar{\beta}\boldsymbol{s}_\theta^x(\bar{\mathbf{z}}_{t'}) \tag{146}$$

$$= 2\bar{\beta}\Gamma\boldsymbol{s}_\theta^x(\bar{\mathbf{z}}_t) + 2\bar{\beta}^2\Big[\begin{pmatrix}\boldsymbol{s}_\theta^x(\bar{\mathbf{z}}_t)\\ \boldsymbol{s}_\theta^m(\bar{\mathbf{z}}_t)\end{pmatrix}^T\begin{pmatrix}\Gamma^2\\ -\nu\end{pmatrix}\Big] + 2\Gamma\bar{\beta}\boldsymbol{s}_\theta^x(\bar{\mathbf{z}}_{t'}) \tag{147}$$

$$= 2\bar{\beta}\Gamma\boldsymbol{s}_\theta^x(\bar{\mathbf{z}}_t) + 2\bar{\beta}^2\Big[\boldsymbol{s}_\theta(\bar{\mathbf{z}}_t)^T\begin{pmatrix}\Gamma^2\\ -\nu\end{pmatrix}\Big] + 2\Gamma\bar{\beta}\boldsymbol{s}_\theta^x(\bar{\mathbf{z}}_{t'}) \tag{148}$$

In this work, we parameterize the score $\boldsymbol{s}_\theta(\bar{\mathbf{z}}_t) = -\boldsymbol{L}_t^{-T}\boldsymbol{\epsilon}_\theta(\bar{\mathbf{z}}_t)$ where $\boldsymbol{L}_t$ is Cholesky factorization matrix of the covariance matrix $\boldsymbol{\Sigma}_t$ of the perturbation kernel at time t. Substituting this parameterization in Eqn. 148, we get,

$$\hat{\boldsymbol{s}}_\theta = 2\bar{\beta}\Gamma\boldsymbol{s}_\theta^x(\bar{\mathbf{z}}_t) + 2\bar{\beta}^2\Big[-\boldsymbol{\epsilon}_\theta^T(\bar{\mathbf{z}}_t)\boldsymbol{L}_t^{-1}\begin{pmatrix}\Gamma^2\\ -\nu\end{pmatrix}\Big] + 2\Gamma\bar{\beta}\boldsymbol{s}_\theta^x(\bar{\mathbf{z}}_{t'}) \tag{149}$$

$$= 2\bar{\beta}\Gamma\boldsymbol{s}_\theta^x(\bar{\mathbf{z}}_t) - 2\bar{\beta}^2\Big[\Gamma^2(l_t^{xx}\boldsymbol{\epsilon}_\theta^x(\bar{\mathbf{z}}_t) + l_t^{xm}\boldsymbol{\epsilon}_\theta^m(\bar{\mathbf{z}}_t)) - \nu l_t^{mm}\boldsymbol{\epsilon}_\theta^m(\bar{\mathbf{z}}_t)\Big] + 2\Gamma\bar{\beta}\boldsymbol{s}_\theta^x(\bar{\mathbf{z}}_{t'}) \tag{150}$$

$$= 2\bar{\beta}\Gamma\boldsymbol{s}_\theta^x(\bar{\mathbf{z}}_t) - 2\bar{\beta}^2\Big[\underbrace{\Gamma^2 l_t^{xx}}_{=\lambda_1}\boldsymbol{\epsilon}_\theta^x(\bar{\mathbf{z}}_t) + \underbrace{(\Gamma^2 l_t^{xm} - \nu l_t^{mm})}_{=\lambda_2}\boldsymbol{\epsilon}_\theta^m(\bar{\mathbf{z}}_t)\Big] + 2\Gamma\bar{\beta}\boldsymbol{s}_\theta^x(\bar{\mathbf{z}}_{t'}) \tag{151}$$

where,

$$l_t^{xx} = \frac{1}{\sqrt{\Sigma_t^{xx}}} \tag{152}$$

$$l_t^{xm} = \frac{-\Sigma_t^{xm}}{\sqrt{\Sigma_t^{xx}}\sqrt{\Sigma_t^{xx}\Sigma_t^{mm} - (\Sigma_t^{xm})^2}} \tag{153}$$

$$l_t^{mm} = \sqrt{\frac{\Sigma_t^{xx}}{\Sigma_t^{xx}\Sigma_t^{mm} - (\Sigma_t^{xm})^2}} \tag{154}$$

Assuming the input $\bar{\mathbf{z}}_t$ is a sample from the underlying flow map of the reverse SDE (a very strong assumption), the score term $\hat{\boldsymbol{s}}_\theta$ in Eqn. 144 is the primary source of introducing errors (since the neural network-based score prediction will be offset by some error from the true underlying score). Without loss of generality, we further assume that the update timepoints $t, t'$and$\,t''$ lie in the low timestep regime. Furthermore, for notational convenience, we denote $\lambda_1 = \Gamma^2 l_t^{xx}$ and $\lambda_2 = (\Gamma^2 l_t^{xm} - \nu l_t^{mm})$ as the scaling factors for the second term in Eqn. 151. Thus, the error introduced due to the neural network predictors $\boldsymbol{\epsilon}_\theta^x(\bar{\mathbf{z}}_t)$ and $\boldsymbol{\epsilon}_\theta^m(\bar{\mathbf{z}}_t)$ will be scaled by λ_1 and λ_2 respectively. Therefore, for achieving lower sampler discretization errors, it might be desirable to have low magnitudes of λ_1 and λ_2. We now qualitatively analyze the magnitude of these coefficients for different ranges of values of Γ and ν.

Case-1: Effect of using a non-zero Γ: We first analyze the impact of using a non-zero Γ value on the magnitude of λ_1 and λ_2. Figure 4a illustrates the impact of the choice of Γ and ν on the coefficients

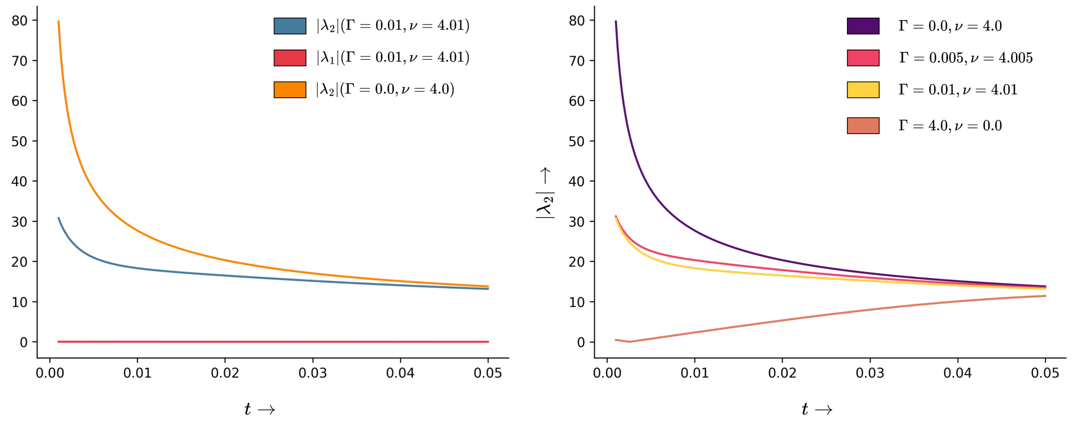

Figure 4: (a) Comparison between $|\lambda_1|$ and $|\lambda_2|$ corresponding to $\Gamma = 0.0$ and $\Gamma = 0.01$ in the low-timestep regime for a fixed $M^{-1} = 4$. (b) Variation of $|\lambda_2|$ for different values of (Γ, ν)

λ_1 and λ_2 in the low-timestep regime. When $\Gamma = 0$, the error in the score term $\hat{\boldsymbol{s}}_\theta$ will be only due to the term $|\lambda_2|\boldsymbol{\epsilon}_\theta^m(\bar{\mathbf{z}}_t)$. As illustrated in Figure 4a, the value of $|\lambda_2|$ (when $\Gamma = 0$) is very high in the low-timestep regime and, therefore might negatively impact the sample quality since any errors in the estimation of $\boldsymbol{\epsilon}_\theta^m(\bar{\mathbf{z}}_t)$ would be scaled by a large factor.

Interestingly, for the setting $\Gamma = 0.01, \nu = 4.01$, the value of $|\lambda_2|$ reduces significantly, thus reducing the error scaling factor. It is worth noting that using a non-zero Γ also simultaneously enables error contribution from other terms in $\hat{\boldsymbol{s}}_\theta$ involving Γ (especially $|\lambda_1|\boldsymbol{\epsilon}_\theta^x(\bar{\mathbf{z}}_t)$). However, as illustrated in Figure 4a, the value of $|\lambda_1|$ is extremely small as compared to $|\lambda_2|$ making the additional error introduced insignificant. Due to this reason, the overall error introduced by the score term $\hat{\boldsymbol{s}}_\theta$ is more when $\Gamma = 0$ as compared to the setting with a (small) non-zero Γ value. This explains why using a small value of Γ yields better sample quality than our CLD baseline (See Table 4)

Case-2: Effect of using a large Γ: Figure 4b illustrates the variation of $|\lambda_2|$ for some more values of Γ. Interestingly for $\Gamma = 4.0$, the value of $|\lambda_2|$ decreases to almost 0 in the low-timestep regime. However, for $\Gamma = 4.0$, the value of $|\lambda_1|$ increases significantly (Figure 5a), therefore, leading to large error scaling factors in the term $|\lambda_1|\boldsymbol{\epsilon}_\theta^x(\bar{\mathbf{z}}_t)$. This finding justifies our observation in Figure 2, where a value of $\Gamma = 0.25$ makes sample quality significantly worse for the CelebA-64 dataset and is unable to recover high-frequency details. Figure 5b further illustrates the variation of $|\lambda_1|$ for different (Γ, ν) pairs in the low-timestep regime.

From the above analysis, it seems that *the choice of* Γ *provides an important trade-off between balancing the errors produced due to the terms* $\boldsymbol{\epsilon}_\theta^x(\bar{\mathbf{z}}_t)$ *and* $\boldsymbol{\epsilon}_\theta^m(\bar{\mathbf{z}}_t)$ in Eqn. 151. Therefore, the choice of Γ is crucial for sample quality in PSLD.

D.2 Additional Speed vs. Sample Quality Comparisons

CIFAR-10: We extend the Speed vs. Sample Quality results in Table 5 to include results for PSLD with $\Gamma = 0.01$ in Table 12

CelebA-64: Similar to our setup for CIFAR-10 (Section 4.3), we benchmark the speed vs quality tradeoffs of PSLD ($\Gamma = 0.005$) against our CLD ablation baseline for the CelebA-64 dataset (See Table 13). Similar to CIFAR-10, PSLD outperforms our CLD baseline across all timesteps. The performance difference is most notable in the low-timestep regime (FID of 6.99 for PSLD with EM-QS vs 10.7 for CLD with SSCS-QS). However, there are two notable differences in our observations when compared to CIFAR-10:

(i) Firstly, quadratic striding works best for the CelebA-64 dataset. This contrasts with CIFAR-10, where a uniform striding schedule works better.

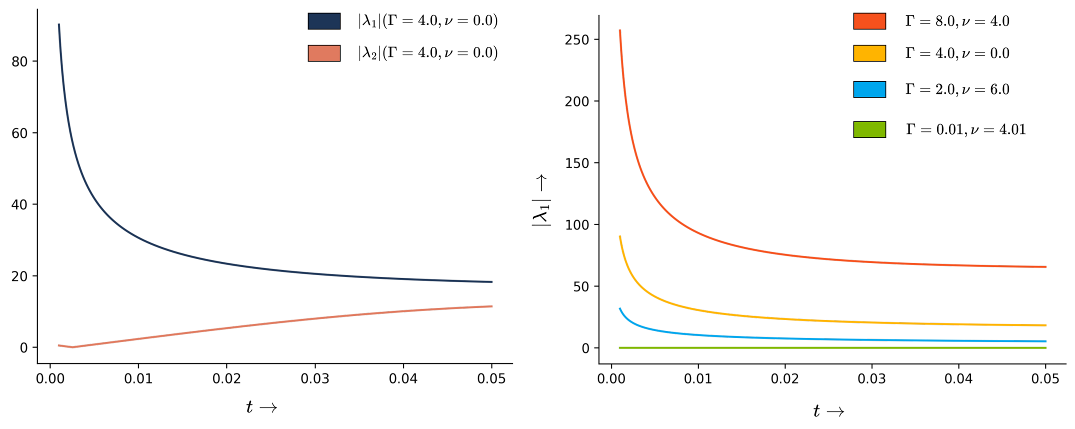

Figure 5: (a) Comparison between $|\lambda_1|$ and $|\lambda_2|$ corresponding to $\Gamma = 4.0$ and $\nu = 0.0$ in the low-timestep regime. (b) Variation of $|\lambda_1|$ for different values of (Γ, ν)

		NFE (FID@10k ↓)				
Sampler	Method	50	100	250	500	1000
EM-QS	CLD	25.01	8.91	5.97	5.61	5.7
	VP-SDE	**17.72**	7.45	5.59	5.51	5.51
	(Ours) PSLD ($\Gamma = 0.01$)	23.96	8.12	5.41	**5.13**	**5.24**
	(Ours) PSLD ($\Gamma = 0.02$)	19.94	**7.33**	**5.26**	5.20	5.28
EM-US	CLD	119.68	45.60	9.08	5.71	5.65
	VP-SDE	**84.54**	41.93	12.61	5.92	5.19
	(Ours) PSLD ($\Gamma = 0.01$)	109.01	40.22	**9.07**	**5.25**	4.95
	(Ours) PSLD ($\Gamma = 0.02$)	100.62	**39.96**	11.26	5.45	**4.82**
SSCS-QS	CLD	21.31	8.37	5.82	5.75	5.69
	(Ours) PSLD ($\Gamma = 0.01$)	18.41	7.42	5.41	**5.28**	5.29
	(Ours) PSLD ($\Gamma = 0.02$)	**16.12**	**7.16**	**5.36**	5.35	**5.27**
SSCS-US	CLD	75.45	24.74	6.09	5.74	5.78
	(Ours) PSLD ($\Gamma = 0.01$)	76.6	21.25	**5.18**	5.10	5.33
	(Ours) PSLD ($\Gamma = 0.02$)	**72.42**	**20.46**	5.19	**4.92**	**5.29**

Table 12: Extended Speed vs. Sample quality comparisons using the SDE setup for CIFAR-10. FID computed for 10k samples. Values in **bold** indicate the best result for that column.

(ii) More interestingly, PSLD achieves the best performance of FID=3.77 at N=250 steps, and sample quality degrades on further increasing the number of steps. This contrasts our results for CIFAR-10, where PSLD achieves the best performance at T=1000.

D.3 Extended SOTA Results

Extended Qualitative Results: We provide qualitative samples from our SOTA CIFAR-10 models using the SDE and ODE setups in Figures 8 and 9 respectively. We provide some additional samples from the AFHQv2 dataset at the 128 x 128 resolution in Figure 10.

Extended Quantitative Results: Table 14 shows the FID and IS scores for all models using the SDE sampling setup. PSLD with $\Gamma = 0.02$ attains the best IS score of 9.74. When using the ODE sampling setup (Table 15), PSLD with $\Gamma = 0.02$ achieves the best IS score of 9.93.

D.4 Conditional Synthesis using PSLD

Class-Conditional Synthesis: As discussed in Section 4.4, given class label information $\mathbf{y}$, an unconditional pre-trained score network $\boldsymbol{s}_\theta(\mathbf{z}_t, t)$ can be used for sampling from the class conditional

		NFE (FID@10k ↓)				
Sampler	Method	50	100	250	500	1000
EM-QS	CLD	73.61	14.78	4.77	4.48	4.59
	(Ours) PSLD ($\Gamma = 0.005$)	**44.36**	**6.99**	**3.77**	**3.92**	**4.17**
EM-US	CLD	122.63	54.67	11.66	4.97	4.6
	(Ours) PSLD ($\Gamma = 0.005$)	**99.05**	**44.06**	**8.9**	**4.42**	**4.37**
SSCS-QS	CLD	44.83	10.7	4.82	4.73	4.74
	(Ours) PSLD ($\Gamma = 0.005$)	**34.3**	**8.13**	**4.16**	**4.11**	**4.09**
SSCS-US	CLD	105.16	45.54	6.75	**4.06**	4.18
	(Ours) PSLD ($\Gamma = 0.005$)	**97.8**	**35.59**	**4.65**	4.08	**4.05**

Table 13: Speed vs. Sample Quality comparison using the SDE setup for CelebA-64. FID computed for 10k samples. Values in **bold** indicate the best result for that column.

Model	Size	NFE	FID ↓	IS ↑
PSLD (Γ=0.01)	55M	1000	2.34	9.57
PSLD (Γ=0.02)	55M	1000	2.3	9.68
PSLD (Γ=0.01, deep)	97M	1000	2.26	9.71
PSLD (Γ=0.02, deep)	97M	1000	**2.21**	**9.74**

Table 14: CIFAR-10 sample quality (SDE). FID (lower is better) and IS (higher is better) were computed on 50k samples.

Model	Size	NFE	FID ↓	IS ↑
PSLD (Γ=0.01)	55M	243	2.41	9.63
PSLD (Γ=0.02)	55M	232	2.4	9.84
PSLD (Γ=0.01, deep)	97M	246	**2.10**	9.79
PSLD (Γ=0.02, deep)	97M	231	2.31	9.91
PSLD (Γ=0.01, deep)	97M	159	2.13	9.76
PSLD (Γ=0.02, deep)	97M	159	2.34	**9.93**

Table 15: CIFAR-10 sample quality (ODE). FID (lower is better) and IS (higher is better) were computed on 50k samples.

distribution $p(\mathbf{z}_t|\mathbf{y})$ in PSLD. More specifically, we need to simulate the following reverse SDE:

$$d\mathbf{z}_t = \left[\boldsymbol{f}(\mathbf{z}_t) - \boldsymbol{G}(t)\boldsymbol{G}(t)^T\nabla_{\mathbf{z}_t}\log p(\mathbf{z}_t|\mathbf{y})\right]dt + \boldsymbol{G}(t)d\mathbf{w}_t \tag{155}$$

The *conditional* score $\nabla_{\mathbf{z}_t}\log p(\mathbf{z}_t|\mathbf{y})$ can be further decomposed as follows:

$$\nabla_{\mathbf{z}_t}\log p(\mathbf{z}_t|\mathbf{y}) = \nabla_{\mathbf{z}_t}\log p(\mathbf{y}|\mathbf{z}_t) + \nabla_{\mathbf{z}_t}\log p(\mathbf{z}_t) \tag{156}$$

$$\approx \underbrace{\nabla_{\mathbf{z}_t}\log p(\mathbf{y}|\mathbf{z}_t)}_{\text{Classifier Gradient}} + \underbrace{\boldsymbol{s}_\theta(\mathbf{z}_t,t)}_{\text{Score}} \tag{157}$$

For practical scenarios, it is common to scale the contribution of the classifier gradient by a factor of $\lambda > 1$. Thus,

$$\nabla_{\mathbf{z}_t}\log p(\mathbf{z}_t|\mathbf{y}) = \lambda\nabla_{\mathbf{z}_t}\log p(\mathbf{y}|\mathbf{z}_t) + \boldsymbol{s}_\theta(\mathbf{z}_t,t) \tag{158}$$

The above technique of approximating the conditional score $\nabla_{\mathbf{z}_t}\log p(\mathbf{z}_t|\mathbf{y})$ is called as *classifier-guidance* [2, 23]. The classifier $p(\mathbf{y}|\mathbf{z}_t)$ is trained by minimizing a time-dependent cross-entropy loss as follows:

$$\mathcal{L}_{\text{clf}}(\phi) = \mathbb{E}_{t\sim\mathcal{U}(0,1)}\mathbb{E}_{\mathbf{x}_0,\mathbf{y}\sim p_{\text{data}(\mathbf{x}_0,\mathbf{y})}}\mathbb{E}_{\mathbf{z}_t\sim p(\mathbf{z}_t|\mathbf{x}_0)}\left[-\sum_k \mathbb{1}(y=y_k)\log C_\phi^k(\mathbf{z}_t,t)\right] \tag{159}$$

where $C_\phi^k(\mathbf{z}_t,t)$ is a time-dependent classifier that takes as input a perturbed data point $\mathbf{z}_t$ and outputs class prediction probabilities. We perform class conditional synthesis for the CIFAR-10 (10 classes) and the AFHQ-v2 datasets. For the AFHQ-v2 dataset, we use the classes *Cats*, *Dogs*, and *Others* from the train split for classifier training (See Appendix C.6 for complete implementation details). We provide additional class conditional samples for CIFAR-10 in Figure 11 and for AFHQ-v2 in Figure 12.

Image Inpainting: Following [2], we can partition the input $\mathbf{x}_0$ into known ($\hat{\mathbf{x}}_0$) and unknown ($\bar{\mathbf{x}}_0$) components respectively. We can now define the diffusion for the unknown component in the augmented space as follows:

$$d\bar{\mathbf{z}}_t = \bar{\mathbf{f}}(\mathbf{z}_t)dt + \bar{\boldsymbol{G}}(t)d\mathbf{w}_t \tag{160}$$

where $\bar{\mathbf{f}}(\mathbf{z}_t) = \mathbf{f}(\bar{\mathbf{z}}_t)$ i.e. the drift applied to the missing components of $\mathbf{z}_t$. Similarly, $\bar{\boldsymbol{G}}(t)$ corresponds to the diffusion coefficient applied to the corresponding components of Brownian motion $d\mathbf{w}_t$. The corresponding reverse-SDE (conditioned on the observed signal $\hat{\mathbf{x}}_0$) can be specified as:

$$d\bar{\mathbf{z}}_t = \left[\bar{\mathbf{f}}(\mathbf{z}_t) - \bar{\boldsymbol{G}}(t)\bar{\boldsymbol{G}}^T(t)\nabla_{\bar{\mathbf{z}}_t}\log p(\bar{\mathbf{z}}_t|\hat{\mathbf{x}}_0)\right]dt + \bar{\boldsymbol{G}}(t)d\mathbf{w}_t \tag{161}$$

Following the derivation in [2], it can be shown that:

$$\nabla_{\bar{\mathbf{z}}_t}\log p(\bar{\mathbf{z}}_t|\hat{\mathbf{x}}_0) \approx \nabla_{\bar{\mathbf{z}}_t}\log p(\bar{\mathbf{z}}_t|\hat{\mathbf{z}}_t) \tag{162}$$

$$= \nabla_{\bar{\mathbf{z}}_t}\log p([\bar{\mathbf{z}}_t;\hat{\mathbf{z}}_t]) \tag{163}$$

where $\hat{\mathbf{z}}_t \sim p(\hat{\mathbf{z}}_t|\hat{\mathbf{x}}_0)$ is a noisy augmented state sampled from the perturbation kernel given an observed signal $\hat{\mathbf{x}}_0$. We provide additional imputation results in Figure 13

General Inverse Problems: Similar to imputation, we can utilize PSLD for solving general inverse problems. Given a conditioning signal $\mathbf{y}$, we have,

$$\nabla_{\mathbf{z}_t}\log p_t(\mathbf{z}_t \mid \mathbf{y}) = \nabla_{\mathbf{z}_t}\log\int p_t(\mathbf{z}_t \mid \mathbf{y}_t, \mathbf{y})p(\mathbf{y}_t \mid \mathbf{y})d\mathbf{y}_t, \tag{164}$$

Further assuming that $p(\mathbf{y}_t \mid \mathbf{y})$ is tractable and $p_t(\mathbf{z}_t \mid \mathbf{y}_t, \mathbf{y}) \approx p_t(\mathbf{z}_t \mid \mathbf{y}_t)$, we have

$$\nabla_{\mathbf{z}_t}\log p_t(\mathbf{z}_t \mid \mathbf{y}) \approx \nabla_{\mathbf{z}_t}\log\int p_t(\mathbf{z}_t \mid \mathbf{y}_t)p(\mathbf{y}_t \mid \mathbf{y})d\mathbf{y}_t \tag{165}$$

$$\approx \nabla_{\mathbf{z}_t}\log p_t(\mathbf{z}_t \mid \hat{\mathbf{y}}_t) \tag{166}$$

$$= \nabla_{\mathbf{z}_t}\log p_t(\mathbf{z}_t) + \nabla_{\mathbf{z}_t}\log p_t(\hat{\mathbf{y}}_t \mid \mathbf{z}_t) \tag{167}$$

$$\approx \boldsymbol{s}_{\boldsymbol{\theta}^*}(\mathbf{z}_t, t) + \nabla_{\mathbf{z}_t}\log p_t(\hat{\mathbf{y}}_t \mid \mathbf{z}_t), \tag{168}$$

where $\hat{\mathbf{y}}_t \sim p(\mathbf{y}_t \mid \mathbf{y})$. Thus PSLD can be used for conditional synthesis like previous SGMs [2] while achieving better speed-vs-quality tradeoffs and better overall sample quality. Therefore, PSLD provides an attractive baseline for further developments in SGMs.

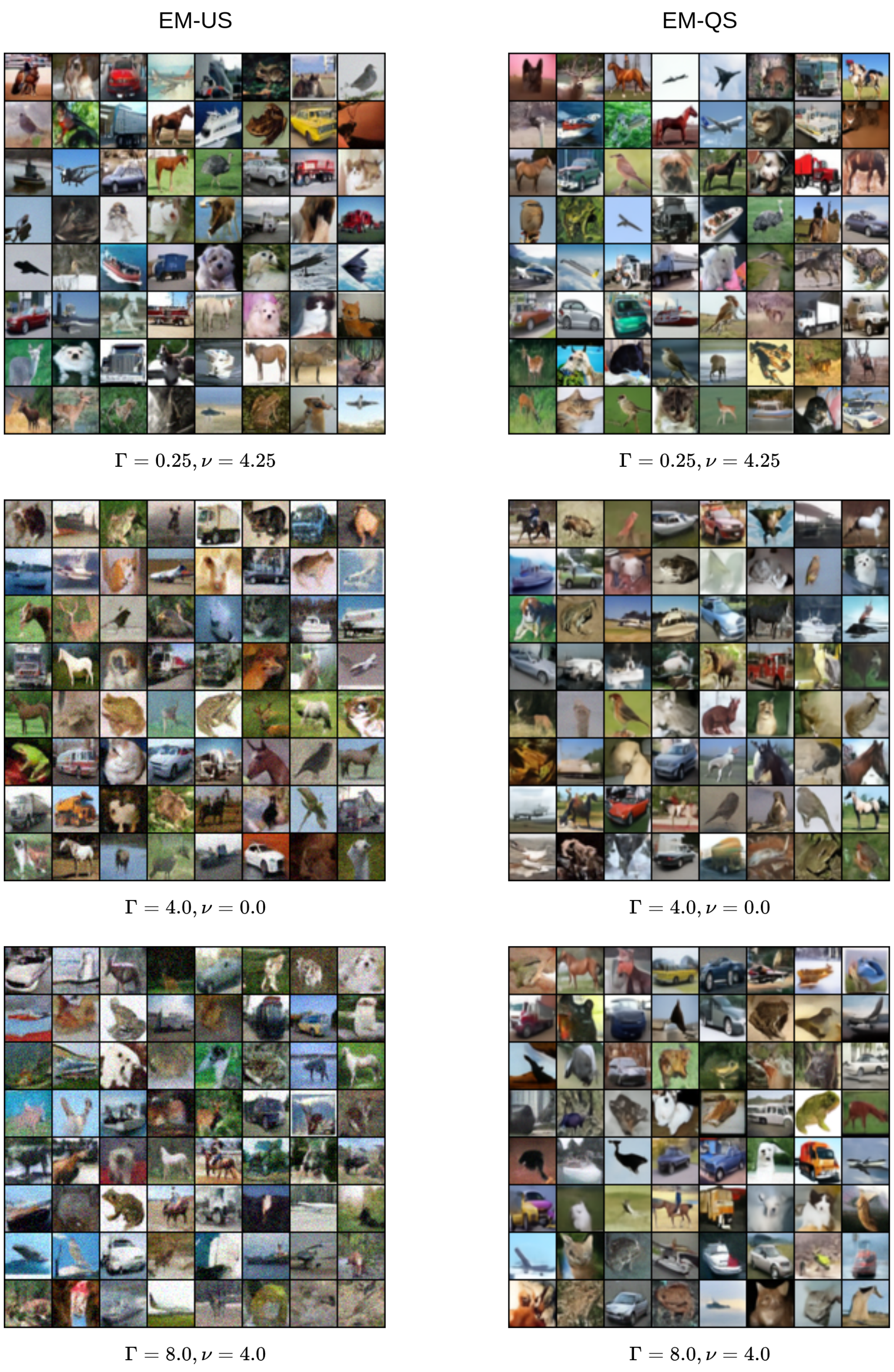
EM-US
EM-QS
$\Gamma = 0.25, \nu = 4.25$
$\Gamma = 0.25, \nu = 4.25$
$\Gamma = 4.0, \nu = 0.0$
$\Gamma = 4.0, \nu = 0.0$
$\Gamma = 8.0, \nu = 4.0$
$\Gamma = 8.0, \nu = 4.0$

Figure 6: Qualitative illustration of the impact of Γ on CIFAR-10 sample quality. Samples get progressively worse when increasing Γ. (Uncurated) Samples in the left and right columns were generated using EM-US and EM-QS samplers.

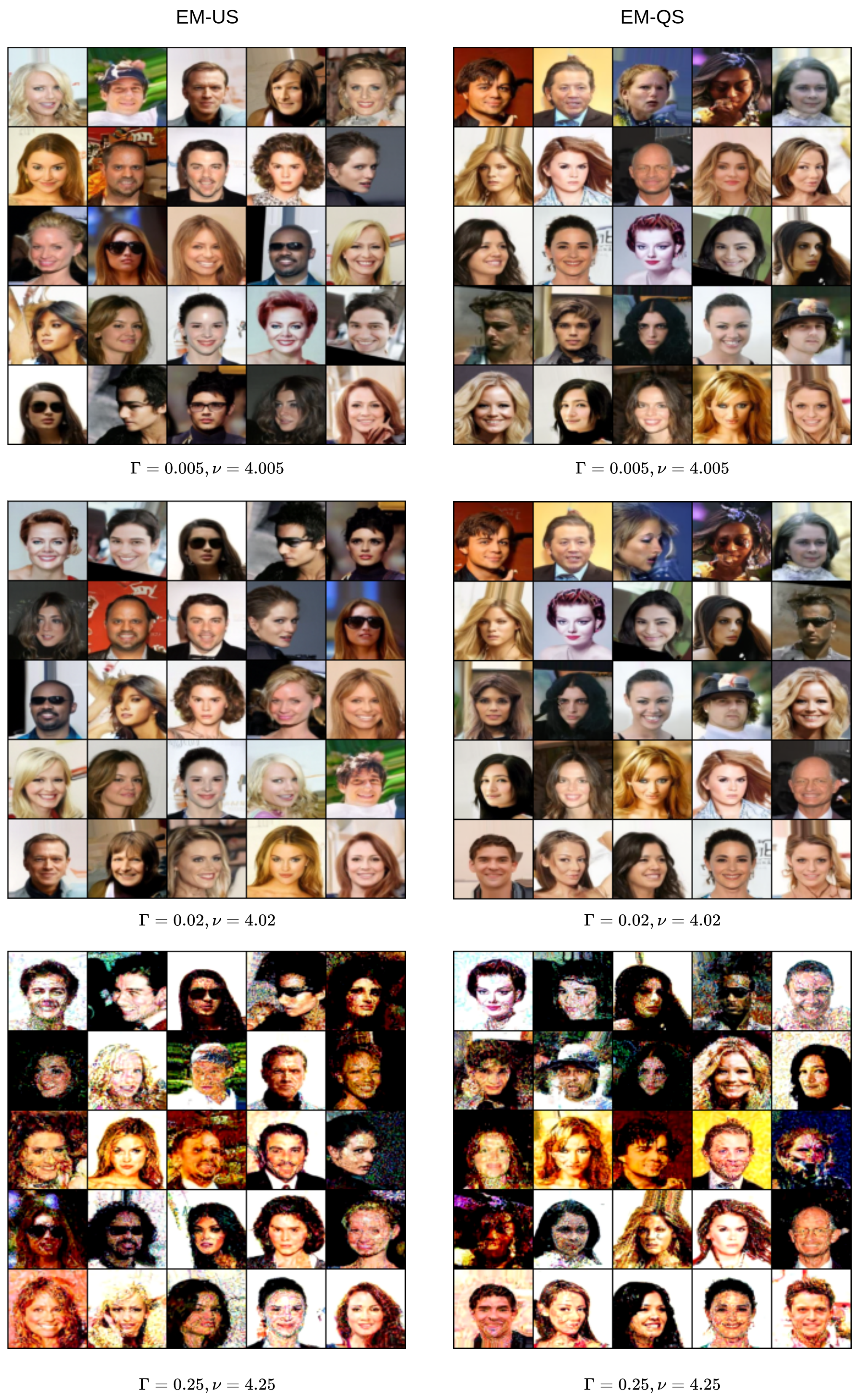
EM-US
EM-QS
$\Gamma = 0.005, \nu = 4.005$
$\Gamma = 0.005, \nu = 4.005$
$\Gamma = 0.02, \nu = 4.02$
$\Gamma = 0.02, \nu = 4.02$
$\Gamma = 0.25, \nu = 4.25$
$\Gamma = 0.25, \nu = 4.25$

Figure 7: Qualitative illustration of the impact of Γ on CelebA-64 sample quality. Samples get progressively worse when increasing Γ. (Uncurated) Samples in the left and right columns were generated using EM-US and EM-QS samplers.

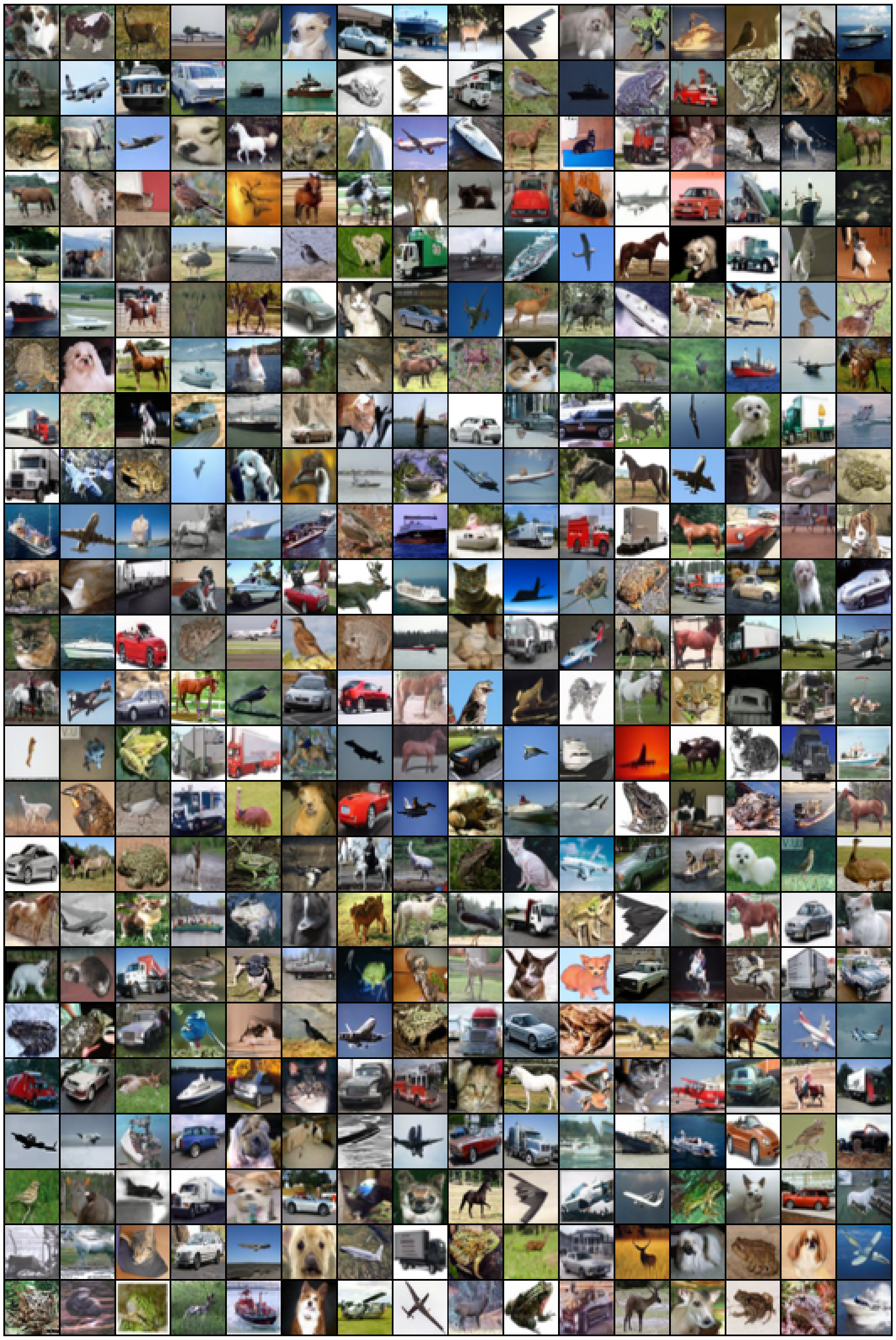

Figure 8: Uncurated samples from our SOTA PSLD ($\Gamma = 0.02, \nu = 4.02$) model using SDE sampling (FID=2.21, NFE=1000)

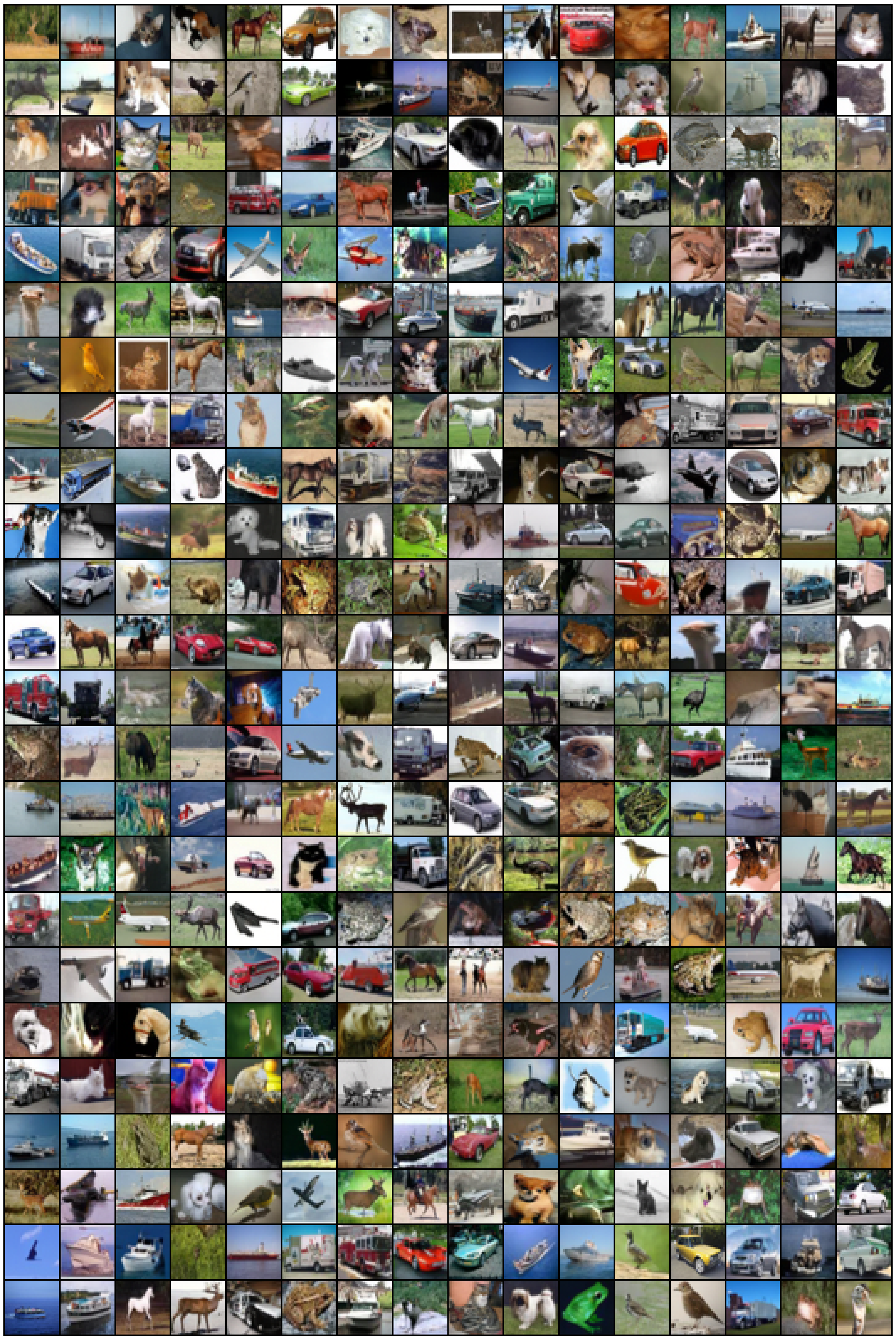
Figure 9: Uncurated samples from our SOTA PSLD ($\Gamma = 0.01, \nu = 4.01$) model using ODE sampling (FID=2.10, NFE=246)

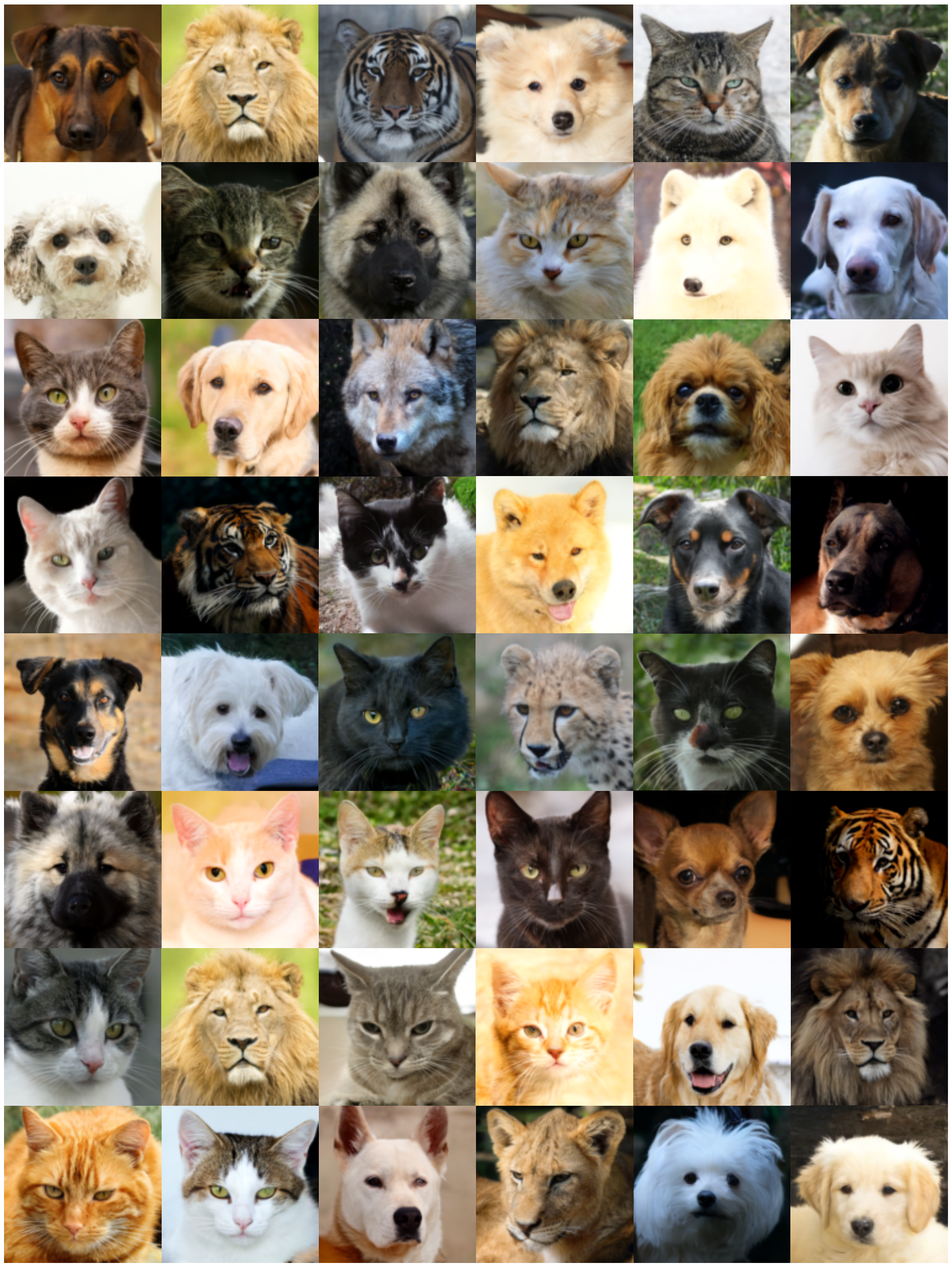

Figure 10: Random unconditional AFHQv2 samples at 128x128 resolution from our PSLD ($\Gamma = 0.01$) model using the EM-QS sampler with N=1000.

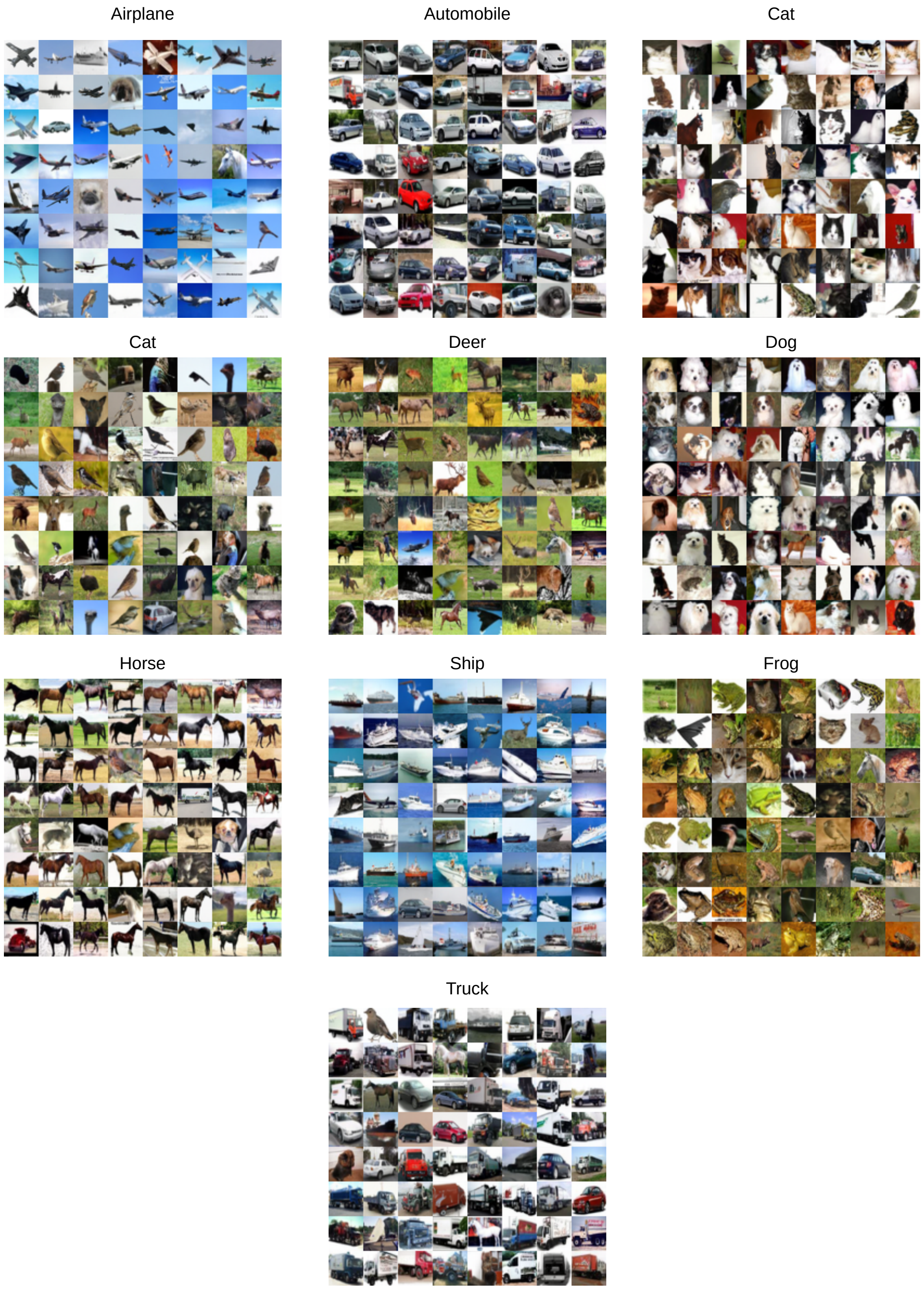

Figure 11: Randomly sampled class conditional results on the CIFAR-10 dataset using the EM-US sampler (N=1000). Guidance weight $\lambda = 5.0$. Using large guidance weight reduces diversity but improves sample quality.

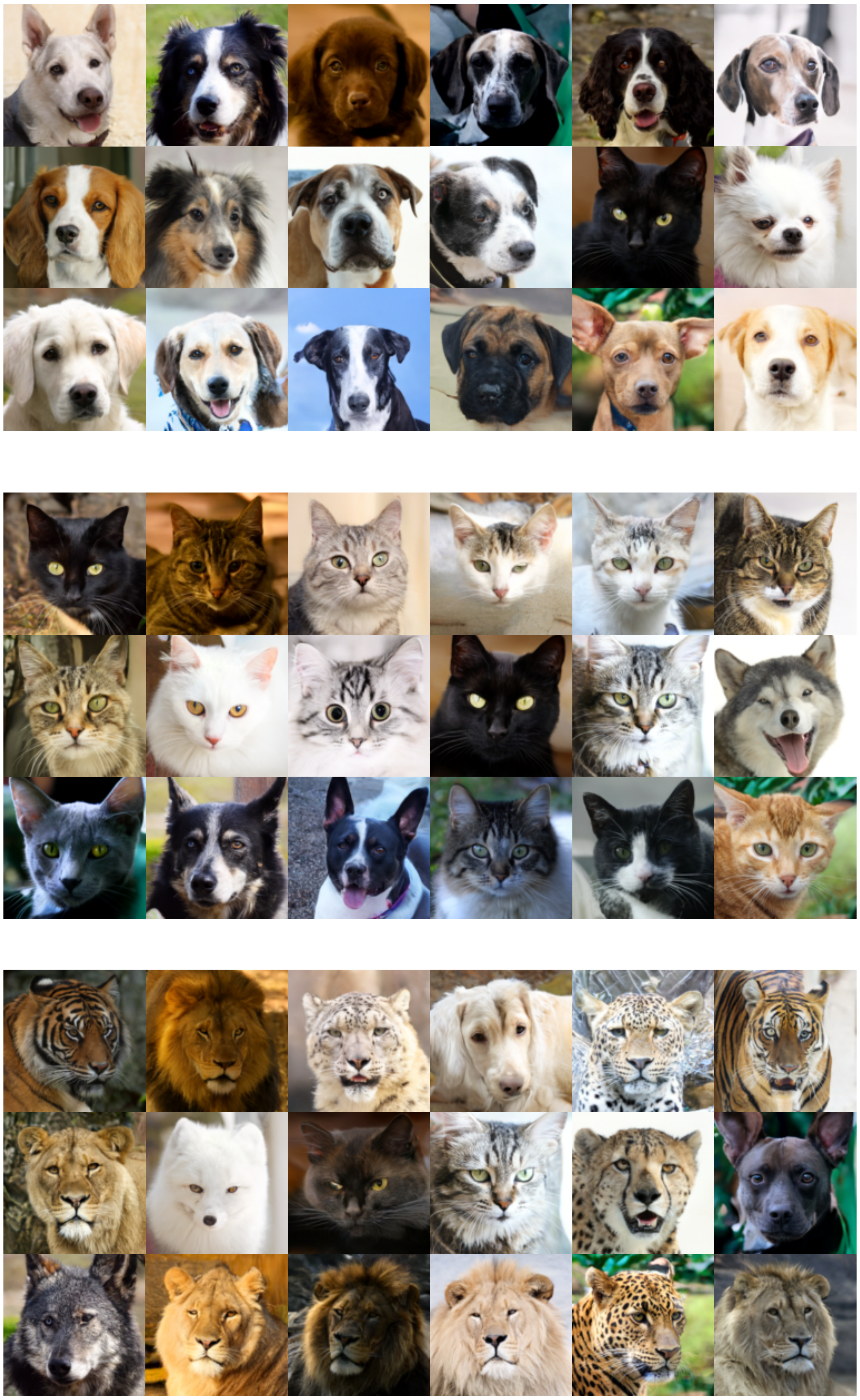
Figure 12: Randomly sampled class conditional results on the AFHQv2 dataset using the EM-US sampler (N=1000). Guidance weight $\lambda = 10.0$. (Top to Bottom) Each of the three rows correspond to *Dog*, *Cats* and *Others*.

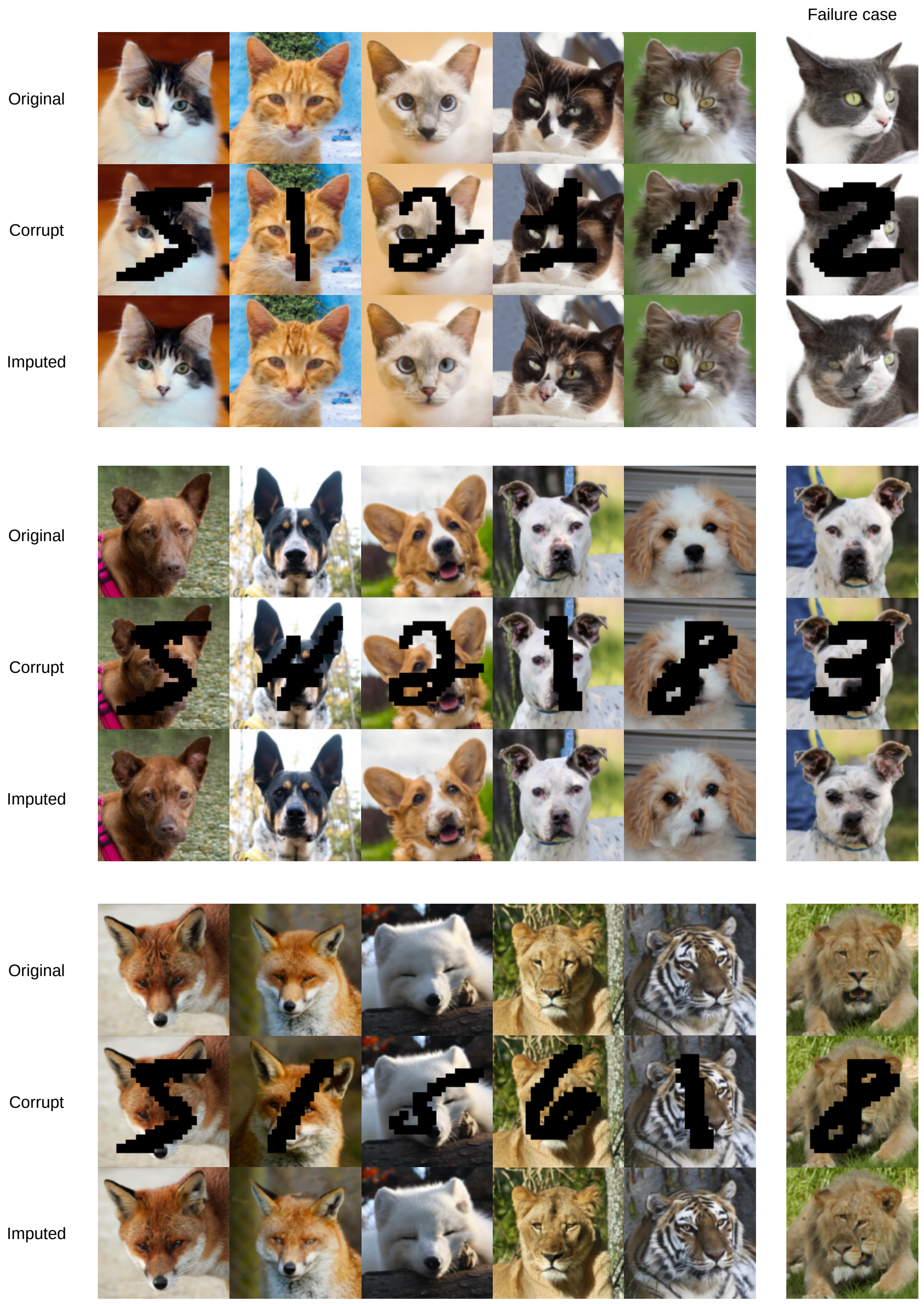

Figure 13: Additional imputation results on the AFHQv2 dataset (test split) using the EM-US sampler (N=1000). The Rightmost column indicates some failure cases.

References

[1] Yi-An Ma, Tianqi Chen, and Emily Fox. A complete recipe for stochastic gradient mcmc. *Advances in neural information processing systems*, 28, 2015.

[2] 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*, .

[3] Tim Dockhorn, Arash Vahdat, and Karsten Kreis. Score-based generative modeling with critically-damped langevin diffusion. In *International Conference on Learning Representations*.

[4] Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum likelihood training of score-based diffusion models. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL `https://openreview.net/forum?id=AklttWFnxS9`.

[5] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7):1661–1674, 2011. doi: 10.1162/NECO_a_00142.

[6] Simo Särkkä and Arno Solin. *Applied stochastic differential equations*, volume 10. Cambridge University Press, 2019.

[7] B. Leimkuhler. *Molecular dynamics : with deterministic and stochastic numerical methods / Ben Leimkuhler, Charles Matthews.* Interdisciplinary applied mathematics, 39. Springer, Cham, 2015. ISBN 3319163744.

[8] H. F. Trotter. On the product of semi-groups of operators. *Proceedings of the American Mathematical Society*, 10(4):545–551, 1959. doi: 10.1090/s0002-9939-1959-0108732-6. URL `https://doi.org/10.1090/s0002-9939-1959-0108732-6`.

[9] Gilbert Strang. On the construction and comparison of difference schemes. *SIAM Journal on Numerical Analysis*, 5(3):506–517, September 1968. doi: 10.1137/0705041. URL `https://doi.org/10.1137/0705041`.

[10] J.R. Dormand and P.J. Prince. A family of embedded runge-kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19–26, 1980. ISSN 0377-0427. doi: https://doi.org/10.1016/0771-050X(80)90013-3. URL `https://www.sciencedirect.com/science/article/pii/0771050X80900133`.

[11] Alex Krizhevsky. Learning multiple layers of features from tiny images. pages 32–33, 2009. URL `https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf`.

[12] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[13] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8188–8197, 2020.

[14] 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.

[15] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems*.

[16] Richard Zhang. Making convolutional networks shift-invariant again. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7324–7334. PMLR, 09–15 Jun 2019. URL `https://proceedings.mlr.press/v97/zhang19a.html`.

[17] Ricky T. Q. Chen. torchdiffeq, 2018. URL `https://github.com/rtqichen/torchdiffeq`.

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

[19] Alexia Jolicoeur-Martineau, Rémi Piché-Taillefer, Ioannis Mitliagkas, and Remi Tachet des Combes. Adversarial score matching and improved sampling for image generation. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=eLfqMl3z3lq`.

[20] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

[21] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

[22] Anton Obukhov, Maximilian Seitzer, Po-Wei Wu, Semen Zhydenko, Jonathan Kyl, and Elvis Yu-Jing Lin. High-fidelity performance metrics for generative models in pytorch, 2020. URL `https://github.com/toshas/torch-fidelity`. Version: 0.3.0, DOI: 10.5281/zenodo.4957738.

[23] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.