A. Extended Technical Discussion

We continue our technical discussion in Secs. 4 and 5, dividing this section into three parts.

A.1. More on the Prediction Objective

Optimality of training with Eq. (8). Recall that we have already established that the loss value of Eq. (8) equals $\mathbb{E}_{\boldsymbol{z},\delta}\|\partial_\delta \boldsymbol{f}_\theta(\boldsymbol{z},\delta) - \boldsymbol{u}(\boldsymbol{f}_\theta(\boldsymbol{z},\delta), 1-\delta)\|^2$. Thus, at optimality, we have $\partial_\delta \boldsymbol{f}_\theta(\boldsymbol{z},\delta) = \boldsymbol{u}(\boldsymbol{f}_\theta(\boldsymbol{z},\delta), 1-\delta)$ for all $\boldsymbol{z} \sim \pi$ and $\delta \in [0,1]$. Furthermore, we note that the parametric relationship $\boldsymbol{f}_\theta(\boldsymbol{z},\delta) = \boldsymbol{z} + \delta \boldsymbol{F}_\theta(\boldsymbol{z},\delta)$, which satisfies $\boldsymbol{f}_\theta(\boldsymbol{z},0) = \boldsymbol{z}$ for all $\boldsymbol{z} \sim \pi$ by design. We show that, assuming standard regularity conditions on $\boldsymbol{u}$, we ensure $\boldsymbol{\phi}_{\boldsymbol{u}}$ is uniquely defined via an initial value problem by the Picard-Lindelöf theorem [8], thus providing a well-defined criterion for $\boldsymbol{f}_\theta$. This is a direct result of the fundamental theory of ODE.

Proposition A.1. *Let $\boldsymbol{\phi}_{\boldsymbol{u}}(\boldsymbol{x}_t, t, s)$ be defined as in Eq. (3), and we assume that there exists some $L > 0$ for all $\boldsymbol{y}, \boldsymbol{y}' \in \mathbb{R}^d$ and $r \in [0,1]$, $\|\boldsymbol{u}(\boldsymbol{y}, r) - \boldsymbol{u}(\boldsymbol{y}', r)\| \le L\|\boldsymbol{y} - \boldsymbol{y}'\|$. We further assume that $\boldsymbol{f}_\theta(\boldsymbol{z},\delta)$ is continuously differentiable for $\delta \in [0,1]$. Then, we have $\boldsymbol{f}_\theta(\boldsymbol{z},\delta) = \boldsymbol{\phi}_{\boldsymbol{u}}(\boldsymbol{z}, 1, 1-\delta)$ for all $\boldsymbol{z} \sim \pi, \delta \in [0,1]$, if and only if, (i) $\boldsymbol{f}_\theta(\boldsymbol{z},0) = \boldsymbol{z}$ for all $\boldsymbol{z} \sim \pi$, and (ii) $\partial_\delta \boldsymbol{f}_\theta(\boldsymbol{z},\delta) = \boldsymbol{u}(\boldsymbol{f}_\theta(\boldsymbol{z},\delta), 1-\delta)$ for all $\boldsymbol{z} \sim \pi, \delta \in [0,1]$.*

Proof. By assuming $\boldsymbol{u}(\boldsymbol{y}, r)$ continuous in r and Lipschitz continuous in $\boldsymbol{y}$, we could apply the Picard-Lindelöf theorem [8] to guarantee the existence and uniqueness of its solution $\boldsymbol{\phi}_{\boldsymbol{u}}$, which is

$$\boldsymbol{\phi}_{\boldsymbol{u}}(\boldsymbol{x}_t, t, s) = \boldsymbol{x}_t + \int_t^s -\boldsymbol{u}(\boldsymbol{x}(\tau), \tau) d\tau.$$

$\Longrightarrow$:

(i). Since the equality $\boldsymbol{f}_\theta(\boldsymbol{z},\delta) = \boldsymbol{\phi}_{\boldsymbol{u}}(\boldsymbol{z}, 1, 1-\delta)$ holds for all $\delta \in [0,1]$, it must hold for $\delta = 0$. We know that $\boldsymbol{\phi}_{\boldsymbol{u}}(\boldsymbol{z}, 1, 1) = \boldsymbol{z}$. Thus, for any $\boldsymbol{z} \sim \pi$,

$$\boldsymbol{f}_\theta(\boldsymbol{z}, 0) = \boldsymbol{\phi}_{\boldsymbol{u}}(\boldsymbol{z}, 1, 1) = \boldsymbol{z}.$$

(ii). Given that $\boldsymbol{f}_\theta(\boldsymbol{z},\delta) = \boldsymbol{\phi}_{\boldsymbol{u}}(\boldsymbol{z}, 1, 1-\delta)$ for all $\delta \in [0,1]$, and both $\boldsymbol{f}_\theta(\boldsymbol{z},\delta)$ and $\boldsymbol{\phi}_{\boldsymbol{u}}(\boldsymbol{z}, 1, 1-\delta)$ are continuously differentiable with respect to δ on $[0,1]$, their derivatives with respect to δ must be equal. That is, for all $\boldsymbol{z} \sim \pi, \delta \in [0,1]$,

$$\partial_\delta \boldsymbol{f}_\theta(\boldsymbol{z},\delta) = \frac{d}{d\delta}\boldsymbol{\phi}_{\boldsymbol{u}}(\boldsymbol{z}, 1, 1-\delta) = \boldsymbol{u}(\boldsymbol{\phi}_{\boldsymbol{u}}(\boldsymbol{z}, 1, 1-\delta), 1-\delta) = \boldsymbol{u}(\boldsymbol{f}_\theta(\boldsymbol{z},\delta), 1-\delta).$$

$\Longleftarrow$:

Since $\boldsymbol{f}_\theta(\boldsymbol{z}, 0) = \boldsymbol{z}$ for all $\boldsymbol{z} \sim \pi$, and $\partial_\delta \boldsymbol{f}_\theta(\boldsymbol{z},\delta) = \boldsymbol{u}(\boldsymbol{f}_\theta(\boldsymbol{z},\delta), 1-\delta)$ for all $\boldsymbol{z} \sim \pi, \delta \in [0,1]$, we know that $\boldsymbol{f}_\theta(\boldsymbol{z},\delta)$ is a solution to the initial value problem on the interval $\delta \in [0,1]$:

$$\frac{d}{d\delta}\boldsymbol{f}_\theta(\boldsymbol{z},\delta) = \boldsymbol{u}(\boldsymbol{f}_\theta(\boldsymbol{z},\delta), 1-\delta), \quad \boldsymbol{f}_\theta(\boldsymbol{z}, 0) = \boldsymbol{z}.$$

By definition, $\boldsymbol{\phi}_{\boldsymbol{u}}(\boldsymbol{z}, 1, 1-\delta)$ is also a solution to the same IVP on $[0,1]$. Thus, we arrive at the equality between $\boldsymbol{f}_\theta(\boldsymbol{z},\delta)$ and $\boldsymbol{\phi}_{\boldsymbol{u}}(\boldsymbol{z}, 1, 1-\delta)$.

□

Discrete-time objective. In Sec. 4.1, we skipped over the development of a more flexible discrete-time training objective for Eq. (8), which does not require efficient JVP implementations. First, recall that our default continuous-time objective is

$$\mathbb{E}_{\boldsymbol{z},\delta}\left\| \boldsymbol{F}_\theta(\boldsymbol{z},\delta) + \mathrm{sg}\Big(\delta \partial_\delta \boldsymbol{F}_\theta(\boldsymbol{z},\delta) - \boldsymbol{u}\left(\boldsymbol{f}_\theta(\boldsymbol{z},\delta), 1-\delta\right)\Big) \right\|^2. \tag{13}$$

Specifically, we can discretize the time horizon and approximate the partial derivative $\partial_\delta \boldsymbol{F}_\theta$ using finite differences. Following the common practice in the sampling procedure of flow models [22], we divide the time horizon into N intervals with $N+1$ boundary points: $1 = t_1 > t_2 > \cdots > t_{N+1} = 0$. We note that the generating velocity at point $\boldsymbol{f}_\theta(\boldsymbol{z}, 1-t_i)$, or equivalently $\boldsymbol{f}_\theta(\boldsymbol{z}, t_1 - t_i)$ where $1 \le i \le N$, can be approximated by $\frac{1}{t_i - t_{i+1}}(\boldsymbol{f}_\theta(\boldsymbol{z}, t_1 - t_{i+1}) - \boldsymbol{f}_\theta(\boldsymbol{z}, t_1 - t_i))$. Short-handing $t_a - t_b$ to $\delta_{a,b}$, we have the following discrete-time objective:

$$\mathbb{E}_{\boldsymbol{z},i}\left\| \boldsymbol{F}_\theta(\boldsymbol{z}, \delta_{1,i+1}) + \mathrm{sg}\left(\delta_{1,i} \cdot \frac{\boldsymbol{F}_\theta(\boldsymbol{z}, \delta_{1,i+1}) - \boldsymbol{F}_\theta(\boldsymbol{z}, \delta_{1,i})}{\delta_{i,i+1}} - \boldsymbol{u}\left(\boldsymbol{f}_\theta(\boldsymbol{z}, \delta_{1,i}), t_i\right)\right) \right\|^2, \tag{14}$$

which approximates Eq. (8) as $\delta_{i,i+1} \to 0$. Just like Eq. (13) equals to $\mathbb{E}_{\boldsymbol{z},\delta}\|\partial_\delta \boldsymbol{f}_\theta(\boldsymbol{z},\delta) - \boldsymbol{u}(\boldsymbol{f}_\theta(\boldsymbol{z},\delta), 1-\delta)\|^2$, we note that the loss value of Eq. (14) is the same as $\mathbb{E}_{\boldsymbol{z},i}\left\|\left(\boldsymbol{f}_\theta(\boldsymbol{z},\delta_{1,i+1}) - \boldsymbol{f}_\theta(\boldsymbol{z},\delta_{1,i})\right)/\delta_{i,i+1} - \boldsymbol{u}\left(\boldsymbol{f}_\theta(\boldsymbol{z},\delta_{1,i}), t_i\right)\right\|^2$, where $\left(\boldsymbol{f}_\theta(\boldsymbol{z},\delta_{1,i+1}) - \boldsymbol{f}_\theta(\boldsymbol{z},\delta_{1,i})\right)/\delta_{i,i+1}$ is the model's generating velocity, approximating $\partial_\delta \boldsymbol{f}_\theta(\boldsymbol{z},\delta)$. If we use $\boldsymbol{v}_{\mathrm{G}}$ to further simplify the notations, we arrive at the following optimization gradients in Eq. (9):

$$\nabla_\theta \, \mathbb{E}_{\boldsymbol{z},\delta}\left[\boldsymbol{F}_\theta(\boldsymbol{z},\delta)^\top \mathrm{sg}\Big(\underbrace{\boldsymbol{v}_{\mathrm{G}}\big(\boldsymbol{f}_\theta(\boldsymbol{z},\delta), 1-\delta\big) - \boldsymbol{u}\big(\boldsymbol{f}_\theta(\boldsymbol{z},\delta), 1-\delta\big)}_{\Delta_{\boldsymbol{v}_{\mathrm{G}},\boldsymbol{u}}(\boldsymbol{f}_\theta(\boldsymbol{z},\delta),1-\delta)}\Big)\right], \tag{15}$$

Error analysis. With a discrete-time objective, we essentially train the student to trace a numerically integrated trajectory by an ODE solver. More concretely, the loss function inherently mimics a specific solver, whose error rate depends on the technique deployed. As written, Eq. (14) can be understood as using an Euler method [74, 78] for solving the flow, specified at time steps $t_1, t_2, \ldots, t_N$. Similar to existing fast-solver literature, we could readily adapt the objective using higher-order finite differences to mimic more precise, higher-order solvers [28, 46]. The classical theory from numerical analysis [80] indicates that a local truncation error at t_i bounded by $\mathcal{O}((\delta_{i,i+1})^{p+1})$ leads to the global error rate is $\mathcal{O}((\delta_{\max})^p)$, where $\delta_{\max} \coloneqq \max_i \delta_{i,i+1} = \max_i(t_i - t_{i+1})$ is the maximum step size. This component is the discretization error, which is entirely independent of the network's approximation error (the inevitable training inaccuracy discussed in Sec. 4.1). Together, these two errors represent the total deviation of the student's trajectory from the true teacher flow defined by $\boldsymbol{u}$. The total error can also be shown [5] to directly control the Wasserstein distance between the generative distribution of the student and the teacher.

Number of discretization steps. The above discussion suggests that one should take as small a step as possible, $(t_i - t_{i+1}) \to 0$, and as precise a solver as possible, for the discretization error to be minimal. This means that we should use the continuous-time setup if allowed, and for the discrete-time setting, we should choose to use a large number of discretization steps N. However, this is not the trend we observe in practice. Surprisingly, larger N after a certain point actually degrades the performance. We attribute this behavior to the accumulated approximation error (discussed in Sec. 4.1) made by the model. Since $\boldsymbol{u}$ is evaluated at the model's prediction, the goodness of the trajectory implicitly depends on the quality of $\boldsymbol{f}_\theta$ itself, and such a problem exacerbates as N becomes larger. We further note that incorporating higher-order solvers in $\boldsymbol{u}$ like the Heun method [28] is helpful, as it significantly reduces the discretization error. Fortunately, in Fig. 9, we observe that while the model is sensitive to the number of discretization steps and solver choices when training with Eq. (9) alone, the corrective signal in Eq. (11) effectively renders such a decision unimportant, making our final proposal robust against N and the precision of solvers.

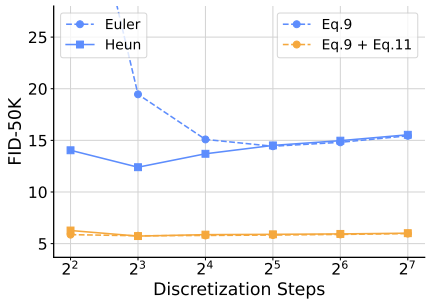

Figure 9. **Effects of discretization steps and solver type.** Our correction objective in Eq. (11) makes the model robust against both.

Sampling of δ. Another important aspect of the training algorithm is the time sampling of δ. On one hand, the correctness of the model propagates from small jumps $\delta = 0$ to large jumps $\delta = 1$. On the other hand, the high-frequency features of the images only emerge at a lower noise level (large δ). To balance these two notions, we investigate a mixture of a logit-normal distribution [11, 28] and a fixed value of $\delta = 0$ (effectively a dropout on δ). Results are presented in Tab. 3. Note that for the discrete-time setting, we apply an additional step of the floor operation with respect to our predefined set of discrete time steps.

% of $\delta = 0$	δ sampling	FID ↓
10%	LogitNormal(0, 1)	12.40
0%	LogitNormal(0, 1)	47.09
30%	LogitNormal(0, 1)	13.19
50%	LogitNormal(0, 1)	14.30
10%	LogitNormal(-0.8, 1)	13.78
10%	LogitNormal(-0.4, 1)	12.98
10%	LogitNormal(0, 1.2)	12.60
0%	LogitNormal(-0.8, 1.6)	13.43
0%	LogitNormal(-0.4, 1.6)	12.98
0%	LogitNormal(0, 1.6)	12.59

Table 3. **Ablation of δ sampling.** A mixture of a logit-normal and a fixed δ=0 works well.

Confident region warmup. We observe that naively optimizing with Eq. (9) brings about instability early on in training, mainly with the JVP approach. We hypothesize that it is because $\boldsymbol{u}$ is evaluated at a state predicted by $\boldsymbol{f}_\theta$, which can be out-of-distribution for $\boldsymbol{u}$ at initialization. We propose to add noise to the predicted state before feeding it to $\boldsymbol{u}$. Specifically, we replace $\boldsymbol{u}(\boldsymbol{f}_\theta(\boldsymbol{z},\delta), 1-\delta)$, with $\boldsymbol{u}(\boldsymbol{I}_{t_c|1-\delta}(\boldsymbol{f}_\theta(\boldsymbol{z},\delta), \boldsymbol{n}), t_c)$, where $\boldsymbol{I}_{t_c|t}$ is the generalized interpolating function, a transition kernel that takes samples from the noise level of t to t_c.

For the linear interpolation scheme [40, 43, 50], we have $\boldsymbol{I}_{t_c|t}(\boldsymbol{x}_t, \boldsymbol{n}) = \frac{1-t_c}{1-t}\boldsymbol{x}_t + \sqrt{t_c^2 - \frac{(1-t_c)^2}{(1-t)^2}t^2}\boldsymbol{n}$, assuming $t_c > t$; if $t_c \leq t$, it simply does nothing and returns $\boldsymbol{x}_t$. At the start of training, we linearly decrease t_c from 1 to 0 over a short warmup of 10K steps. Intuitively, this procedure ensures that $\boldsymbol{u}$ always operates in a region where it is confident. After the inclusion of this warmup period, we do not find any instability of training $\boldsymbol{f}_\theta$ with a reasonable sampling distribution of δ.

A.2. More on the Correction Objective

From minimizing IKL to aligning the noising velocities. We know from prior works [48, 98] that minimizing Eq. (10) w.r.t. θ gives us the following gradient:

$$\mathbb{E}_{r,\boldsymbol{x}_r}\left[\left(\nabla_\theta \boldsymbol{x}_r\right)^\top \left(\nabla_{\boldsymbol{x}_r}\log q_r(\boldsymbol{x}_r) - \nabla_{\boldsymbol{x}_r}\log p_r(\boldsymbol{x}_r)\right)\right], \tag{16}$$

where $\nabla_{\boldsymbol{x}_r}\log q_r(\boldsymbol{x}_r)$ and $\nabla_{\boldsymbol{x}_r}\log p_r(\boldsymbol{x}_r)$ are the score functions [25, 77] of q_r and p_r respectively. It has also been shown that there exists a direct bijective translation between the score functions and the marginal velocities [50]. Specifically, for the linear interpolation scheme [40, 43, 50] and our definition of the conditional velocity, we have:

$$\begin{aligned}\boldsymbol{u}(\boldsymbol{x}_r, r) &= \frac{r}{1-r}\nabla_{\boldsymbol{x}_r}\log p_r(\boldsymbol{x}_r) + \frac{1}{1-r}\boldsymbol{x}_r \\ \nabla_{\boldsymbol{x}_r}\log p_r(\boldsymbol{x}_r) &= \frac{1-r}{r}\boldsymbol{u}(\boldsymbol{x}_r, r) - \frac{1}{r}\boldsymbol{x}_r.\end{aligned}$$

Additionally, recall that the student predicts the data sample as $\boldsymbol{f}_\theta(\boldsymbol{z}, 1)$. With average velocity parameterization, the intermediate state is thus $\boldsymbol{x}_r = \boldsymbol{I}_r(\boldsymbol{f}_\theta(\boldsymbol{z},1), \boldsymbol{n}) = (1-r)(\boldsymbol{z} + \boldsymbol{F}_\theta(\boldsymbol{z},1)) + r\boldsymbol{n}$. Thus, we could rewrite Eq. (16) as

$$\mathbb{E}_{r,\boldsymbol{x}_r}\left[\frac{(1-r)^2}{r}\left(\nabla_\theta \boldsymbol{F}_\theta(\boldsymbol{z},1)\right)^\top \left(\boldsymbol{v}_{\mathrm{N}}\big(\boldsymbol{I}_r(\boldsymbol{f}_\theta(\boldsymbol{z},1),\boldsymbol{n}), r\big) - \boldsymbol{u}\big(\boldsymbol{I}_r(\boldsymbol{f}_\theta(\boldsymbol{z},1),\boldsymbol{n}), r\big)\right)\right],$$

where $\boldsymbol{v}_{\mathrm{N}}$ is the marginal noising velocity induced by the student's generated distribution q, similar to $\boldsymbol{u}$ with p. Dropping the weighting $\frac{(1-r)^2}{r}$, as it does not change the optimal solution and provides us with easier-to-control gradients, we arrive at Eq. (11). Note that a different interpolation scheme does not change our investigation.

$$\nabla_\theta\, \mathbb{E}_{\boldsymbol{z},\boldsymbol{n},r}\left[\boldsymbol{F}_\theta(\boldsymbol{z},1)^\top \mathrm{sg}\Big(\underbrace{\boldsymbol{v}_{\mathrm{N}}\big(\boldsymbol{I}_r(\boldsymbol{f}_\theta(\boldsymbol{z},1),\boldsymbol{n}), r\big) - \boldsymbol{u}\big(\boldsymbol{I}_r(\boldsymbol{f}_\theta(\boldsymbol{z},1),\boldsymbol{n}), r\big)}_{\Delta_{\boldsymbol{v}_{\mathrm{N}},\boldsymbol{u}}(\boldsymbol{I}_r(\boldsymbol{f}_\theta(\boldsymbol{z},1),\boldsymbol{n}),r)}\Big)\right]. \tag{17}$$

Learning rate. Our correction objective defined in Eq. (11) assumes access to $\boldsymbol{v}_{\mathrm{N}}$, which is the velocity of the noising flow starting from the generated distribution, and we approximate it by training an auxiliary model $\boldsymbol{g}_\psi$ concurrently with Eq. (2). The quality of the optimization signal $\Delta_{\boldsymbol{v}_{\mathrm{N}},\boldsymbol{u}}$ depends on the quality of this approximation. Empirically in Tab. 4, we confirm this intuition and observe that the algorithm benefits from having a larger learning rate on $\boldsymbol{g}_\psi$ compared to $\boldsymbol{f}_\theta$. We note that it is also possible to adopt a two time-scale update rule [19, 97], where we train multiple iterations on $\boldsymbol{g}_\psi$ before updating $\boldsymbol{f}_\theta$, but we do not explore this option given the overhead. In a related vein, we find that while training with only the prediction objective in Eq. (9) permits a wide range of learning rates, incorporating the correction objective in Eq. (11) prefers a smaller one on $\boldsymbol{f}_\theta$ (adopted 3×10^{-5} compared to 1×10^{-4} in SiT).

learning rate for $\boldsymbol{g}_\psi$	FID ↓
3×10^{-5}	8.28
6×10^{-5}	5.77
8×10^{-5}	5.72
1×10^{-4}	5.63

Table 4. **Ablation of $\boldsymbol{g}_\psi$'s lr.** A higher lr compared to $\boldsymbol{f}_\theta$'s one (3×10^{-5}) is better.

Additional note on sampling of r in Eq. (11). Recall that our understanding of Eq. (11) with a PDE perspective through the continuity equation in Sec. 5.1 leads to our design of sampling r more in the higher noise levels. We would like to make a brief note that, similar to Eq. (9), if $\boldsymbol{F}_\theta$ is further parameterized, we should replace the first term in Eq. (11) with the actual network output. Additionally, one needs to ensure $\Delta_{\boldsymbol{v}_{\mathrm{N}},\boldsymbol{u}}$ across different r values are roughly on the same scale first, so that their actual contributions can be precisely controlled via the sampling of r. Concretely, for our case of the linear interpolation and velocity parameterization [40, 43, 50], $\Delta_{\boldsymbol{v}_{\mathrm{N}},\boldsymbol{u}}$ is really just the difference between the direct outputs of two neural networks, which does not require further manipulations, assuming the network outputs are of consistent scale. In comparison, another

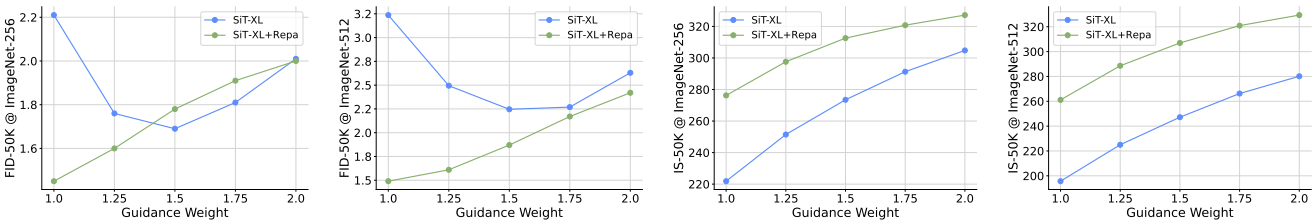

Figure 10. Performances across different guidance strength parameter γ in terms of FID (↓) and Inception Score (↑).

commonly used setup is the EDM parameterization [28], whose velocity at $(\boldsymbol{x}_r = \boldsymbol{f}_\theta(\boldsymbol{z}, 1) + r\boldsymbol{n}, r), \boldsymbol{n} \sim \pi$ is of the form $\frac{c_{\text{skip}}(r)-1}{r}\boldsymbol{x}_r + \frac{c_{\text{out}}(r)}{r}\boldsymbol{G}$, where $\boldsymbol{G}$ is the actual network (the auxiliary or teacher model). Notice that the outputs of $\boldsymbol{G}$ are by design of constant norm across r, so the magnitude of $\Delta_{\boldsymbol{v}_{\text{N}},\boldsymbol{u}}$ is proportional to $\frac{c_{\text{out}}(r)}{r}$. In Yin et al. [98], an additional weighting of r is applied, which makes the overall gradient contributions from Eq. (11) proportional to $c_{\text{out}}(r)$. Substituting in the actual terms used, we have $\frac{0.5r}{\sqrt{0.5^2+r^2}}$, which heavily downplays the effect of small r (lower noise levels).

A.3. More on How to Combine Both

Additional schedule on α. We set $\alpha = 0$ for the first 10K steps (recall that the prediction objective has a warmup of 10K steps), and follow it with a linear warmup of another 10K steps before α settles at a reference value α_{ref}. While the model performance is shown to be robust across a wide range of α in Fig. 6, we further notice a general trend that larger α learns faster in the beginning of training, and smaller α converges better as the training continues. Hence, for our REPA distillation tasks, we try out a simple inverse square root decay [30, 33], and arrive at the following schedule on α:

$$\alpha = \alpha_{\text{ref}} \frac{\text{clip}\left(\frac{n - T_{\text{delay}}}{T_{\text{warmup}}}, 0, 1\right)}{\sqrt{\max(n/T_{\text{decay}}, 1)}},$$

where n is the current training iteration, $T_{\text{delay}} = T_{\text{warmup}} = 10K$, and T_{decay} is the hyperparameter that controls the decay rate with ∞ indicating no decay applied (α stays at the constant value α_{ref} after warmup). We also would like to clarify that, considering the adopted 75-25 split, an α value of 0.3 really means that the gradient contribution of the correction objective is around 10% of that of the prediction objective.

Additional empirical results on prediction and correction synergy. In addition to presenting the model progress in Fig. 7, we list the final performances of training with only prediction Eq. (9), only correction Eq. (11), and both in Tab. 5. Specifically, all models are distilled from SiT-XL/2 [50], and we compare them in terms of FID [19] and Inception Score [66] at 1.5M iterations (300 epochs). For each method, we select the optimal γ based on the FID performances. Recall from Fig. 7, at this point in training, although Eq. (9) makes progress, its absolute performance still lags far behind the other two configurations because of the significant error accumulation. In contrast, Eq. (11) has already suffered from mode collapse, and its diversity continues to deteriorate.

objective	FID ↓	IS ↑
Eq. (9)	5.78	257.02
Eq. (11)	3.19	258.15
Eqs. (9) and (11)	1.69	273.49

Table 5. **Synergy between Eqs. (9) and (11).** Together, they achieve performance that neither could attain in isolation.

B. Implementation Details

Training. We follow the standard practice and train our models in the latent space of the VAE used in Rombach et al. [62]. The model architecture we use is based on a standard DiT [55] with 2×2 patches. Recall that the standard input to a flow model for ImageNet is $\boldsymbol{f}_\theta(\boldsymbol{x}_t, t, c)$ (c is the class label), and we need to include two additional scalar inputs: jump duration δ and guidance strength γ. That is, during both training and inference, the model follows $\boldsymbol{f}_\theta(\boldsymbol{z}, 1, c, \delta, \gamma)$. Thus, we add a few layers to handle these additional conditions. For both, we follow the standard design for including scalar input. Specifically, we use a 256-dimensional frequency embedding [9, 82] followed by a two-layer SiLU-activated MLP with the same dimensionality as the model's hidden size. We then add all four embeddings from t, c, δ, and γ together (the newly added ones, δ and γ, are initialized at 0) and feed the sum to each block. The same goes for $\boldsymbol{g}_\psi$, where it is modified to take input $\boldsymbol{g}_\psi(\boldsymbol{x}_t, t, c, \gamma)$ as it needs to track different noising flows from different γ of $\boldsymbol{f}_\theta$. No further architecture changes are necessary for stable and

Table 6. Detailed experimental configurations of our main results.

Task				
teacher model	SiT-XL/2 [50]		SiT-XL/2+REPA [100]	
resolution	256×256	512×512	256×256	512×512
General				
iterations (epochs)	1.5M (300)	1M (200)	1.5M (300)	1M (200)
batch size	256			
optimizer	Adam [33]			
optimizer betas	(0.9, 0.99)			
optimizer eps	1×10^{-8}			
weight decay	0.0			
dropout	0.0			
$\boldsymbol{f}_\theta$ learning rate	constant 3×10^{-5}			
$\boldsymbol{g}_\psi$ learning rate	constant 1×10^{-4}			
EMA decay	0.9999			
Network				
params (M)	678			
FLOPs (G)	119	525	119	525
depth	28			
hidden dim	1152			
heads	16			
patch size	2×2			
change from teacher	additional input for δ and γ in $\boldsymbol{f}_\theta$; additional input for γ in $\boldsymbol{g}_\psi$			
Training				
Specific to Eq. (9)				
confident region warmup duration	10K			
δ type	discrete; uniform; N=8			
δ sampling	LogitNormal(0, 1)		LogitNormal(-0.4, 1.2)	
% of $\delta = 0$	10%			
$\boldsymbol{u}$ type	Heun solver [28]			
k	1			
Specific to Eq. (11)				
r sampling	LogitNormal(0.8, 1.6)			
guidance interval	[0, 0.4]	[0, 0.5]	[0, 0.3]	[0, 0.3]
Relevant to both Eqs. (9) *and* (11)				
split between Eqs. (9) and (11)	75% : 25%			
γ range	[1, 2]			
α_{ref}	0.3		0.6	
α_{ref} schedule (T_{delay}, T_{warmup}, T_{decay})	(10K, 10K, ∞)		(10K, 10K, 25K)	

effective training. For each of our reported entries listed in Tab. 2, we present their implementation details in Tab. 6. All model trainings are done with an internal JAX codebase on TPU.

Evaluation. We observe small performance variations between TPU-based FID evaluation and GPU-based FID evaluation (ADM's TensorFlow evaluation suite [9][6]). To ensure a fair comparison with the baseline methods, we convert all of our models into PyTorch, sample all of our models on GPU, and obtain FID scores using the ADM evaluation suite for reporting the final results of our XL-size models in Tabs. 2 and 5 and Fig. 10. Additionally, since our model is trained on a range of guidance strengths $\gamma \sim \mathcal{U}(1, 2)$, we can efficiently sweep for an optimal value during inference. We report the best FID in Tab. 2, and provide the complete performance curves in Fig. 10.

[6] https://github.com/openai/guided-diffusion/tree/main/evaluations

C. Related Work

Among the existing distillation approaches, BOOT [16] stands as the most closely related precursor, sharing the distinct operational characteristic of being data-free. However, our works diverge fundamentally in their conceptual positioning and the identified imperative for removing data. BOOT frames the data-free property primarily as a practical/logistical advantage, emphasizing the benefits of bypassing the storage and privacy burdens associated with massive, proprietary training sets. In contrast, we argue that the exclusion of data is not merely a convenience but a theoretical necessity for ensuring distributional fidelity. We elevate the data-free paradigm from a strategy of efficiency to one of correctness, presenting it as the rigorous solution to the identified Teacher-Data Mismatch. Our proposed method also differs significantly from BOOT and its subsequent improvements, which are not necessarily data-free in nature. In particular, Gu et al. [16] focuses on the specific signal-ODE parameterization, which requires a separate loss just to enforce the boundary condition $\boldsymbol{f}_\theta(\boldsymbol{z}, 0) = \boldsymbol{z}$. In comparison, our prediction objective stems from the properties of average velocity [14], which satisfy the boundary condition by design. Tee et al. [81] and the Lagrangian objective in Boffi et al. [5] consider more general ODE formulations. Their optimization involves costly computations of the gradients over the partial derivatives $\partial_\delta \boldsymbol{f}_\theta$, whereas ours does not (the partial derivatives in Eq. (8) are placed inside the stop-gradient operation). This improved training efficiency also originates from the average velocity perspective and our deduced identity in Eq. (7). Furthermore, we introduce an auxiliary correction objective for the accumulated prediction errors, pushing the model performance beyond the current state-of-the-art. In doing so, we believe our work finally completes the picture, validating the data-free paradigm as a robust and promising foundation for the future of generative model acceleration.

Our contribution sits within a much broader body of literature dedicated to accelerating diffusion and flow models. These techniques generally fall into two categories based on their distillation targets. The first category operates at the **trajectory level**, attempting to compress the complex ODE integration into fewer steps by directly mimicking the sampling path or its solution operator [47, 104]. The foundational work of Progressive Distillation [52, 65] further established the viability of this direction through an iterative strategy that progressively halves the required sampling steps. This paradigm was significantly expanded by Consistency Models [3, 45, 76, 79], which enforce a property of self-consistency along the trajectory, allowing the model to map arbitrary intermediate states directly to the data origin. More recent approaches [5, 7, 13, 14, 17, 32, 36, 37, 42–44, 57, 61, 64] have further refined this objective by formulating direct matching conditions between the student's transport map and the teacher's vector field. The second category operates at the **distribution level** [48, 49, 67–69, 89, 94, 95, 97, 98, 107, 108], where the student is trained to match the teacher's marginal distribution directly, often utilizing adversarial or score-based objectives without strictly adhering to the teacher's specific trajectory. We make contributions in both directions by proposing an efficient algorithm for distilling trajectories without data and elucidating additional design spaces for better distribution matching objectives. Together, our proposed predictor-corrector framework can be seen as combining the strengths of the two categories [45, 105], achieving superior quality while maintaining desired diversity, all without reliance on external data.

D. Additional Visual Results

In Figs. 11 to 18, we present additional uncurated samples generated by FreeFlow-XL/2 at 512×512 resolution with only 1-NFE. Again, we emphasize that, during training, we only make use of the teacher model (SiT-XL/2+REPA [100]), without querying any samples from ImageNet.

Figure 11. **Uncurated 512×512 samples by FreeFlow, 1-NFE.** Class label = “robin” (15)

Figure 12. **Uncurated 512×512 samples by FreeFlow, 1-NFE.** Class label = “Siberian husky” (250)

Figure 13. **Uncurated 512×512 samples by FreeFlow, 1-NFE.** Class label = “barn” (425)

Figure 14. **Uncurated 512×512 samples by FreeFlow, 1-NFE.** Class label = “desktop computer” (527)

Figure 15. **Uncurated 512×512 samples by FreeFlow, 1-NFE.** Class label = “dogsled” (537)

Figure 16. **Uncurated 512×512 samples by FreeFlow, 1-NFE.** Class label = “fire truck” (555)

Figure 17. **Uncurated 512×512 samples by FreeFlow, 1-NFE.** Class label = “lakeside” (975)

Figure 18. **Uncurated 512×512 samples by FreeFlow, 1-NFE.** Class label = “volcano” (980)