Faster Sampling: Training from Scratch

Lecturer: Qiang Sun Lecture 10

Contents

1 Faster Sampling: Distillation and Learning from Scratch

These notes cover a family of **training-based acceleration** methods that aim to reduce sampling from many solver steps to *one* or *a handful* of neural-network evaluations.

In Lecture 9 we studied **distillation**: start from a pretrained diffusion model and train a smaller/cheaper model, or a different model class, to match the teacher's sampling behavior. In this lecture we push further and ask:

Question 1 (Learning fast generators from scratch)

Can we train a fast sampler *without* a pretrained diffusion teacher, by directly learning the flow map from data?

The central object will be **probability-flow ODE (PF-ODE)** of a diffusion model, and its associated **flow maps**. We will see three closely related model families:

1. **Consistency Models (CM):** learn the *special* flow map from any noisy time to the data time;
2. **Consistency Trajectory Models (CTM):** learn the *general* flow map between any two times;
3. **Mean Flow (MF):** learn an *averaged drift* whose one-step update approximates the flow map.

All three can be trained via distillation *or* from scratch, and all inherit a **consistency** (semigroup) structure.

1.1 Recap: PF-ODE and flow maps

We use a standard diffusion-model convention where $t \in [0, 1]$ indexes the *noise level*:

- $t = 0$: clean data, $x_0 \sim p_{\text{data}}$;
- $t = 1$: pure noise, $x_1 \sim p_{\text{init}}$ (often a Gaussian).

A score-based diffusion model defines a forward SDE

$$\mathrm{d}x_t = f_t(x_t)\,\mathrm{d}t + g_t\,\mathrm{d}W_t, \qquad x_0 \sim p_{\text{data}}, \qquad t \in [0, 1], \tag{1.1}$$

whose marginals p_t are the noisy distributions. The corresponding probability-flow ODE has the same marginals and reads

$$\frac{\mathrm{d}x_t}{\mathrm{d}t} = f_t(x_t) - \frac{1}{2} g_t^2 \nabla_x \log p_t(x_t) =: v^*(x_t, t). \tag{1.2}$$

We write v^* for the *oracle* PF-ODE drift.

Flow maps. For any $s \geq t$, let $\Psi_{s\to t} : \mathbb{R}^d \to \mathbb{R}^d$ denote the exact flow map of (1.2):

$$x_t = \Psi_{s\to t}(x_s) \qquad \Longleftrightarrow \qquad \dot{x}_u = v^*(x_u, u), \quad u \in [t, s], \quad x_s \text{ given}. \tag{1.3}$$

The flow maps satisfy the **semigroup property**

$$\Psi_{u\to t} \circ \Psi_{s\to u} = \Psi_{s\to t}, \qquad \text{for all} \quad s \geq u \geq t. \tag{1.4}$$

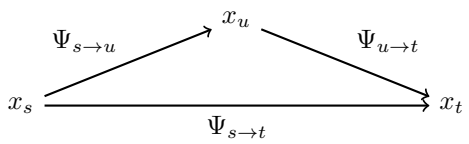

Figure 1: Semigroup property of PF-ODE flow maps: composing two sub-flows equals the full flow.

Gaussian perturbation paths and an alternative expression for v^*. Many diffusion models admit a Gaussian forward kernel

$$x_t \mid x_0 \sim \mathcal{N}\big(\alpha_t x_0,\ \beta_t^2 I_d\big), \qquad \beta_0 = 0,\ \alpha_0 = 1. \tag{1.5}$$

Equivalently, one may sample by

$$x_t = \alpha_t x_0 + \beta_t \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, I_d). \tag{1.6}$$

Differentiating (1.6) in t gives the (conditional) "velocity" $\dot\alpha_t x_0 + \dot\beta_t \varepsilon$. Taking conditional expectation yields an identity frequently used in flow-map models:

$$v^*(x_t, t) = \mathbb{E}\big[\dot\alpha_t x_0 + \dot\beta_t \varepsilon \ \big|\ x_t\big]. \tag{1.7}$$

The above expression is equivalent to (1.2) but does not explicitly involve the score $\nabla_x \log p_t$.

1.2 Flow-map learning as a unifying training objective

A generic **flow-map model** is a neural network $G_\theta(x_s, s, t)$ trained to approximate $\Psi_{s\to t}(x_s)$ for $s \geq t$. The most direct supervised objective would be

$$\mathcal{L}_{\text{oracle-FM}}(\theta) \triangleq \mathbb{E}_{(s,t)}\, \mathbb{E}_{x_s\sim p_s}\Big[\, w(s,t)\, \|G_\theta(x_s, s, t) - \Psi_{s\to t}(x_s)\|_2^2\Big], \tag{1.8}$$

with a weighting function w and a sampling scheme for (s, t). The difficulty is that $\Psi_{s\to t}$ is not available in closed form. Training-based acceleration replaces the oracle map with a *surrogate*:

- **Distillation:** approximate $\Psi_{s\to t}$ using a pretrained diffusion teacher and a numerical solver.
- **From scratch:** build a surrogate directly from the Gaussian path (1.6) and/or from the semigroup property (1.4).

A particularly important special case is $t = 0$: the map from any noise level s directly to (approximately) clean data. This is where **consistency models** enter.

1.3 Consistency models

Special flow map. Define the **consistency function**

$$f^*(\cdot, s) \triangleq \Psi_{s\to 0}(\cdot), \tag{1.9}$$

i.e., $f^*(x_s, s)$ is the deterministic PF-ODE solution at time 0 obtained by starting from x_s at time s. The consistency function naturally satisfies the boundary condition $f^*(x, 0) = x$ for all x, since $\Psi_{0\to 0}$ is the identity map. As the special flow map from any noise level to the data level, f^* is a natural target for learning a fast sampler. This boundary condition also motivates convenient parameterizations for consistency models (Lu and Song, 2024). A learned consistency model is a network $f_\theta(x, s) \approx f^*(x, s)$.

Consistency property. By the semigroup property (1.4), for any $s \geq u \geq 0$,

$$f^*(x_s, s) = \Psi_{u\to 0}(\Psi_{s\to u}(x_s)) = f^*(\Psi_{s\to u}(x_s), u). \tag{1.10}$$

This says that the oracle consistency function is invariant along the PF-ODE trajectory.

An oracle CM objective. A natural oracle objective, though infeasible in practice, is

$$\mathcal{L}_{\text{oracle-CM}}(\theta) \triangleq \mathbb{E}_{s\sim\text{Unif}[0,1]}\, \mathbb{E}_{x_s\sim p_s}\Big[\, \omega(s)\, d\big(f_\theta(x_s, s),\ \Psi_{s\to 0}(x_s)\big)\Big], \tag{1.11}$$

where d is a discrepancy measure, often $\|\cdot\|_2^2$ in pixel space or a perceptual distance, and ω weights different noise levels.

Discrete viewpoint. The original consistency-model and consistency-distillation objectives of Song et al. (2023) work on a finite grid $0 = t_0 < t_1 < \cdots < t_N = 1$ and enforce local consistency between neighboring times. For these notes, we treat that discrete construction only as a finite-difference discretization of the continuous-time formulation below, rather than as a separate model class.

1.3.1 A differential viewpoint

We treat $s \in [0, 1]$ as continuous and aim to learn f_θ for every noise level. Along the PF-ODE trajectory x_s, the oracle consistency function is constant:

$$\frac{\mathrm{d}}{\mathrm{d}s} f^*(x_s, s) = 0. \tag{1.12}$$

Using the chain rule and $\dot{x}_s = v^*(x_s, s)$,

$$\frac{\mathrm{d}}{\mathrm{d}s} f^*(x_s, s) = \partial_s f^*(x_s, s) + \big(\nabla_x f^*(x_s, s)\big)\, v^*(x_s, s) = 0. \tag{1.13}$$

One could try to train by minimizing the residual of (1.13), reminiscent of physics-informed neural networks (PINNs) (Raissi et al., 2019). In practice, however, directly regressing this residual can be unstable.

1.3.2 From local consistency to a tangent objective

A more robust route starts from a small-step local-consistency loss. Fix $\Delta s > 0$ and consider

$$\mathcal{L}_{\text{CM}}^{\Delta s}(\theta, \theta^-) \triangleq \mathbb{E}_s\, \mathbb{E}_{x_s\sim p_s}\Big[\, \omega(s)\, \big\|f_\theta(x_s, s) - f_{\theta^-}(\Psi_{s\to s-\Delta s}(x_s), s - \Delta s)\big\|_2^2\Big]. \tag{1.14}$$

As $\Delta s \to 0$, this loss approaches a first-order tangent objective. On a finite grid, replacing $\Psi_{s\to s-\Delta s}$ by a teacher solver step or by the paired Gaussian corruption path recovers the familiar discrete CM / CD losses.

Proposition 1 (Tangent-limit gradient)

The following convergence result holds:

$$\lim_{\Delta s \to 0} \frac{1}{\Delta s} \nabla_\theta \mathcal{L}_{\mathrm{CM}}^{\Delta s}(\theta, \theta^-) = \nabla_\theta \mathcal{L}_{\mathrm{CM}}^{\infty}(\theta, \theta^-), \tag{1.15}$$

where

$$\mathcal{L}_{\mathrm{CM}}^{\infty}(\theta, \theta^-) \triangleq \mathbb{E}_{s, x_s}\Big[2\omega(s)\, f_\theta(x_s, s)^\top \frac{\mathrm{d}}{\mathrm{d}s} f_{\theta^-}(x_s, s)\Big] \tag{1.16}$$

and the total derivative is

$$\frac{\mathrm{d}}{\mathrm{d}s} f_{\theta^-}(x_s, s) = \partial_s f_{\theta^-}(x_s, s) + \big(\nabla_x f_{\theta^-}(x_s, s)\big)\, v^*(x_s, s). \tag{1.17}$$

The right hand side of (1.16) depends on the **tangent vector** $\frac{\mathrm{d}}{\mathrm{d}s} f_{\theta^-}(x_s, s)$, which itself requires the PF-ODE drift v^*. This is where we choose distillation or from-scratch surrogates.

1.3.3 Continuous-time consistency distillation

With a pretrained diffusion teacher, we can approximate v^* by v_{ϕ^*} (e.g., via a score or noise-prediction network). Then the tangent term becomes

$$\frac{\mathrm{d}}{\mathrm{d}s} f_{\theta^-}(x_s, s) \approx \partial_s f_{\theta^-}(x_s, s) + \big(\nabla_x f_{\theta^-}(x_s, s)\big)\, v_{\phi^*}(x_s, s), \tag{1.18}$$

and substituting into (1.16) yields the continuous-time distillation objective

$$\mathcal{L}_{\mathrm{CD}}^{\infty}(\theta, \theta^-; \phi^*) \triangleq \mathbb{E}_{s, x_s}\Big[2\,\omega(s)\, f_\theta(x_s, s)^\top \Big(\partial_s f_{\theta^-}(x_s, s) + \big(\nabla_x f_{\theta^-}(x_s, s)\big)\, v_{\phi^*}(x_s, s)\Big)\Big]. \tag{1.19}$$

Equivalently, if the teacher provides a score estimate $s_{\phi^*}(x_s, s) \approx \nabla_x \log p_s(x_s)$, then its PF-ODE drift is

$$v_{\phi^*}(x_s, s) = f_s(x_s) - \frac{1}{2} g_s^2\, s_{\phi^*}(x_s, s), \tag{1.20}$$

so (1.19) can be viewed as using the teacher-induced tangent direction along the PF-ODE.

In practice, training proceeds by sampling $x_0 \sim p_{\mathrm{data}}$, a noise level s, and a corrupted point $x_s \sim p_s$, evaluating the teacher drift $v_{\phi^*}(x_s, s)$, and then forming the stop-gradient tangent target

$$\tau_{\theta^-}(x_s, s) \triangleq \partial_s f_{\theta^-}(x_s, s) + \big(\nabla_x f_{\theta^-}(x_s, s)\big)\, v_{\phi^*}(x_s, s). \tag{1.21}$$

The online parameters θ are updated by stochastic gradient descent on the batch objective

$$\widehat{\mathcal{L}}_{\mathrm{CD}}^{\infty}(\theta) = \frac{1}{B} \sum_{b=1}^{B} 2\,\omega(s_b)\, f_\theta(x_{s_b}, s_b)^\top \,\mathrm{sg}\big(\tau_{\theta^-}(x_{s_b}, s_b)\big), \tag{1.22}$$

where sg$(\cdot)$ denotes stop-gradient, and the target parameters are updated by EMA, e.g.

$$\theta^- \leftarrow \tau\,\theta^- + (1-\tau)\,\theta. \tag{1.23}$$

The main computational primitive is the Jacobian–vector product $\big(\nabla_x f_{\theta^-}(x_s,s)\big)\,v_{\phi^*}(x_s,s)$, which is typically computed directly by autodiff rather than by forming the full Jacobian.

1.3.4 Continuous-time consistency training from scratch

Without a teacher, a simple surrogate for $v^*(x_s,s)$ comes from the Gaussian path (1.6):

$$v^*(x_s,s) = \mathbb{E}\big[\dot\alpha_s x_0 + \dot\beta_s \varepsilon \mid x_s\big] \approx \dot\alpha_s x_0 + \dot\beta_s \varepsilon, \tag{1.24}$$

where (x_0,ε) are the same pair used to construct x_s and the approximation is due to the one-point velocity estimator. This leads to the one-point tangent approximation

$$\frac{\mathrm{d}}{\mathrm{d}s} f_{\theta^-}(x_s,s) \approx \partial_s f_{\theta^-}(x_s,s) + \big(\nabla_x f_{\theta^-}(x_s,s)\big)\,(\dot\alpha_s x_0 + \dot\beta_s \varepsilon). \tag{1.25}$$

Example 1 (Simplified continuous-time consistency model)

Directly training with tangent objectives can be numerically delicate. Recent work has proposed stabilizations and parameterizations that make continuous-time training practical and scalable (Lu and Song, 2024; Geng et al., 2024). Below is one concrete instantiation by Lu and Song (2024), referred to as **sCM** (simplified consistency model).

Trigonometric schedule. Let the dataset have standard deviation σ_d, which can be estimated empirically. Use the forward corruption

$$x_s = \cos(s)\,x_0 + \sin(s)\,z, \qquad z \sim \mathcal{N}(0,\sigma_d^2 I_d), \qquad s \in [0,\tfrac{\pi}{2}], \tag{1.26}$$

which interpolates between data ($s=0$) and noise ($s=\frac{\pi}{2}$).

EDM-style parameterization. Following the EDM design space (Karras et al., 2022), parameterize the consistency function as

$$f_\theta(x,s) = \cos(s)\,x \;-\; \sin(s)\,\sigma_d\,F_\theta\Big(\frac{x}{\sigma_d},\; s\Big), \tag{1.27}$$

which enforces $f_\theta(x,0)=x$ automatically.

Stabilizations. Training with tangent targets involves derivatives of the network with respect to time and input. Empirical stabilizations include:

- **Tangent normalization/clipping:** rescale the tangent vector to have bounded norm.
- **Tangent warm-up:** gradually introduce the time-derivative component over early iterations.
- **Careful time embedding:** choose embeddings so that time-derivatives are well-behaved.

Karras et al. (2022) also motivate **adaptive loss weighting** to equalize gradient magnitudes across noise levels.

Algorithm 1: Training a continuous-time consistency model (sCM-style)

Require: data p_{data} with empirical std σ_d; model F_θ and EMA copy F_{θ^-}; learning rate η; stabilization constants c and warm-up horizon H

Initialize θ (optionally from a pretrained network), set $\theta^- \leftarrow \theta$

for *iter* $= 1, 2, \ldots$ **do**

- Sample $x_0 \sim p_{\text{data}}$, $z \sim \mathcal{N}(0, \sigma_d^2 I_d)$
- Sample $s \in [0, \pi/2]$ from a chosen noise-level distribution (e.g., log-normal in $\tan(s)$)
- $x_s \leftarrow \cos(s)\, x_0 + \sin(s)\, z$
- Build a tangent target $w(s, x_s)$ using either `// (i) teacher drift` v_{ϕ^*} `(distillation) or (ii) one-point drift (from scratch)` Apply warm-up and normalization: $w \leftarrow \text{NormalizeWarmup}(w; \texttt{iter}, c, H)$
- Compute $f_\theta(x_s, s)$ via (1.27) and a corresponding $f_{\theta^-}(x_s, s)$
- Update θ by SGD on a stabilized tangent loss (possibly with adaptive weighting)
- Update EMA parameters $\theta^- \leftarrow \text{EMA}(\theta^-, \theta)$

Sampling distribution over noise levels. A commonly used choice is to sample the "noise scale" approximately log-uniformly. Under the trigonometric parameterization, this corresponds to sampling $\tan(s)$ from a log-normal family (Karras et al., 2022).

A training template. Algorithm 1 summarizes a practical training loop (distillation or from scratch), written in a way that matches the notation above.

1.4 General flow map: Consistency Trajectory Models (CTM)

Consistency models learn only $\Psi_{s\to 0}$. For multi-step generation, inverse problems, or likelihood evaluation, it can be valuable to have *all* transitions $\Psi_{s\to t}$ for $t \le s$. This motivates **Consistency Trajectory Models (CTM)** (Kim et al., 2024).

1.4.1 A convenient parameterization of the flow map

Start from the PF-ODE integral representation

$$\Psi_{s\to t}(x_s) = x_s + \int_s^t v^*(x_u, u)\, \mathrm{d}u. \tag{1.28}$$

Rearrange it as a convex combination between the current point x_s and a residual term:

$$\Psi_{s\to t}(x_s) = \frac{t}{s}\, x_s + \frac{s-t}{s} \underbrace{\left[x_s + \frac{s}{s-t}\int_s^t v^*(x_u, u)\, \mathrm{d}u\right]}_{\triangleq\, g^*(x_s, s, t)}. \tag{1.29}$$

This suggests the CTM parameterization

$$G_\theta(x_s, s, t) \triangleq \frac{t}{s}\, x_s + \frac{s-t}{s}\, g_\theta(x_s, s, t), \tag{1.30}$$

where g_θ learns g^*. The boundary condition is automatic: $G_\theta(x_s, s, s) = x_s$.

Proposition 2 (Properties of g^*)

Assume that, along the PF-ODE trajectory $u \mapsto x_u$ with initial condition x_s at time s, the map $u \mapsto v^*(x_u, u)$ is continuous at $u = s$. If, in addition, this map is locally Lipschitz at $u = s$, namely

$$\|v^*(x_u, u) - v^*(x_s, s)\| \le C|u - s| \qquad \text{for } u \text{ near } s,$$

then the following hold.

(i) **Recovering diffusion model:**

$$g^*(x_s, s, s) \triangleq \lim_{t \to s} g^*(x_s, s, t) = x_s - s\, v^*(x_s, s). \tag{1.31}$$

(ii) **Integration representation:**

$$g^*(x_s, s, t) = x_s - s\, v^*(x_s, s) + \mathcal{O}(|t - s|). \tag{1.32}$$

1.4.2 Consistency loss for CTM

CTM training again exploits the semigroup property (1.4):

$$\Psi_{s\to t}(x_s) = \Psi_{u\to t}(\Psi_{s\to u}(x_s)), \qquad t \le u \le s. \tag{1.33}$$

A practical approach is **soft consistency matching**: choose $u \sim \mathrm{Unif}[t, s]$ and match a direct student prediction $G_\theta(x_s, s, t)$ to a stop-gradient prediction obtained by (1) moving from s to u, then (2) applying the student from u to t.

CTM with a pretrained diffusion teacher (distillation). If a pretrained diffusion model is available, approximate $\Psi_{s\to u}$ by a solver driven by the teacher,

$$\widetilde{x}_u^{\phi^*} \triangleq \mathrm{Solver}_{s\to u}(x_s; \phi^*). \tag{1.34}$$

Then enforce consistency by composing two predictions and comparing them in a learned feature space:

$$G_\theta(x_s, s, t) \approx G_{\theta^-}\big(\widetilde{x}_u^{\phi^*}, u, t\big), \qquad u \sim \mathrm{Unif}[t, s], \tag{1.35}$$

where θ^- is a stop-gradient copy of θ, typically maintained via exponential moving average. The right side provides a stable target for the online student network on the left to match.

In practice, to reinforce sample quality while aligning trajectories, both predictions are mapped to time 0 by the stop gradient student and compared in a feature space metric d:

$$x_{\mathrm{est}}(x_s, s, t) \triangleq G_{\theta^-}\big(G_\theta(x_s, s, t),\ t,\ 0\big), \tag{1.36}$$

$$x_{\mathrm{tgt}}(x_s, s, u, t) \triangleq G_{\theta^-}\Big(G_{\theta^-}\big(\mathrm{Solver}_{s\to u}(x_s; \phi^*),\ u,\ t\big),\ t,\ 0\Big), \tag{1.37}$$

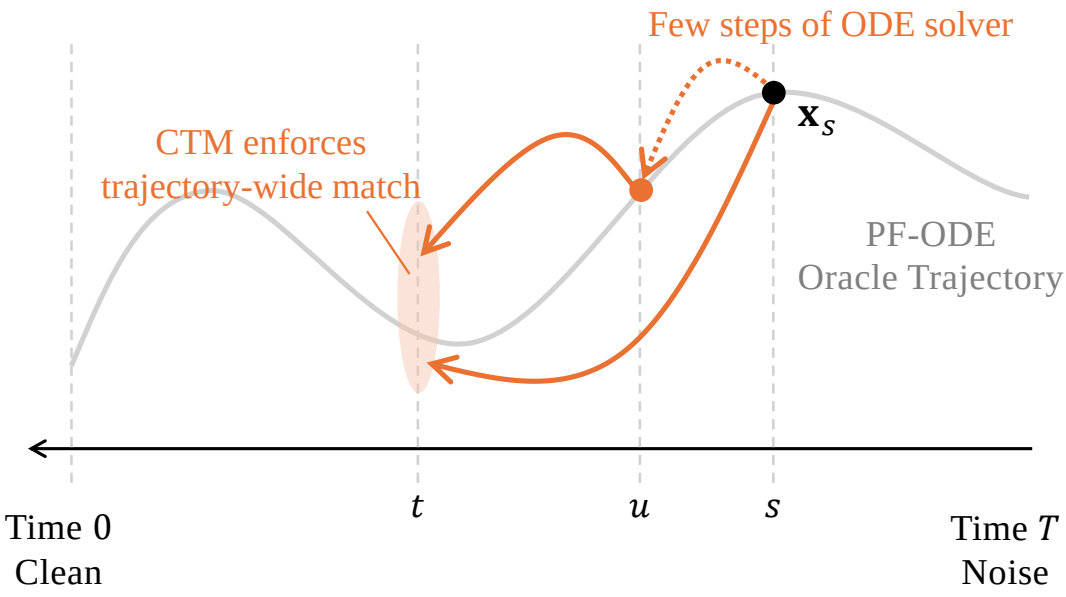

Figure 2: CTM uses the semigroup structure to replace a long-horizon target by a short solver segment followed by a learned jump. The short segment $s \to u$ can come either from a pretrained diffusion teacher or from CTM's self-induced teacher, while the final map $u \to t$ is produced by the student.

The CTM consistency loss is

$$\mathcal{L}_{\text{consist}}(\theta;\phi^*) \triangleq \mathbb{E}_s\, \mathbb{E}_{t\sim\text{Unif}[0,s]}\, \mathbb{E}_{u\sim\text{Unif}[t,s]}\, \mathbb{E}_{x_0,\varepsilon}\Big[\, d\big(x_{\text{est}}(x_s,s,t),\ x_{\text{tgt}}(x_s,s,u,t)\big)\Big], \tag{1.38}$$

where $x_s = \alpha_s x_0 + \beta_s \varepsilon$.

CTM from scratch (self-induced teacher). Without an external teacher, CTM can leverage the special parameterization from (1.31),

$$g^*(x_\tau,\tau,\tau) = x_\tau - \tau\, v^*(x_\tau,\tau), \qquad \Longrightarrow \qquad v^*(x_\tau,\tau) = \frac{1}{\tau}\Big(x_\tau - g^*(x_\tau,\tau,\tau)\Big). \tag{1.39}$$

We therefore replace the oracle residual map $g^*(\cdot,\tau,\tau)$ by the current EMA estimate $g_{\theta^-}(\cdot,\tau,\tau)$, which defines an empirical drift

$$v_{\theta^-}(x_\tau,\tau) \triangleq \frac{1}{\tau}\Big(x_\tau - g_{\theta^-}(x_\tau,\tau,\tau)\Big).$$

This in turn induces a self-induced ODE for a trajectory $\tau \mapsto x(\tau)$:

$$\frac{\mathrm{d}x(\tau)}{\mathrm{d}\tau} = \frac{x(\tau) - g_{\theta^-}(x(\tau),\tau,\tau)}{\tau}. \tag{1.40}$$

Let $\text{Solver}_{s\to u}(\cdot;\theta^-)$ denote a numerical integrator of (1.40). In principle, we can approximate the oracle flow map by solving this ODE from s to t:

$$G_\theta(x_s,s,t) \approx \text{Solver}_{s\to t}(x_s;\theta^-) \approx \Psi_{s\to t}(x_s). \tag{1.41}$$

As in the distillation case, however, full integration over $[t, s]$ can be costly when s and t are far apart. CTM therefore again uses the semigroup property to obtain a shorter supervision path:

$$G_\theta(x_s, s, t) \approx G_{\theta^-}\big(\mathrm{Solver}_{s\to u}(x_s; \theta^-),\ u,\ t\big), \qquad u \sim \mathrm{Unif}[t, s], \tag{1.42}$$

and use the same feature-space loss (1.38), with ϕ^* replaced by θ^-.

Multi-step sampling and error accumulation. Although CTM is often highlighted for one-step or few-step generation, it can also be run on a grid $1 = \tau_0 > \tau_1 > \cdots > \tau_M = 0$ by repeatedly applying short-horizon maps $G_\theta(\cdot, \tau_n, \tau_{n+1})$. Using CM-like sampling such as the γ-sampling (Kim et al., 2024), injecting a small amount of noise often leads to error accumulation; see Kim et al. (2024) for a more detailed analysis.

CTM Supports Diffusion Inference. Because CTM learns $g_\theta(x_s, s, s)$, it implicitly learns a diffusion-model denoiser / velocity. Thus one may also run standard diffusion ODE/SDE solvers such as DDIM and DPM-Solver for generation, using the CTM as the score/denoiser network.

1.4.3 Auxiliary losses in CTM: Diffusion and GAN losses

(Self-)distillation can underperform the teacher because it optimizes only teacher generated targets, lacking direct supervision from real data. However, CTM training is often improved by adding auxiliary objectives, such as a denoising score matching and an adversarial (GAN) term.

Diffusion-model (velocity) loss. Using (1.31), define a velocity parameterization

$$v_\theta(x_s, s) \triangleq \frac{1}{s}\Big(x_s - g_\theta(x_s, s, s)\Big). \tag{1.43}$$

Under the Gaussian perturbation (1.6), an oracle one-point target is $\dot\alpha_s x_0 + \dot\beta_s \varepsilon$. This yields the auxiliary loss

$$\mathcal{L}_{\mathrm{DM}}(\theta) \triangleq \mathbb{E}_{x_0, \varepsilon, s}\Big[\, w(s)\, \big\| v_\theta(x_s, s) - (\dot\alpha_s x_0 + \dot\beta_s \varepsilon)\big\|_2^2\Big]. \tag{1.44}$$

GAN loss (optional). Let D_ζ be a discriminator that distinguishes real $x_0 \sim p_{\mathrm{data}}$ from generated $x_{\mathrm{est}}(x_s, s, t)$. A typical adversarial term is

$$\mathcal{L}_{\mathrm{GAN}}(\theta, \zeta) \triangleq \mathbb{E}_{x_0 \sim p_{\mathrm{data}}}\big[\log D_\zeta(x_0)\big] + \mathbb{E}_{s,t,x_0,\varepsilon}\Big[\log\big(1 - D_\zeta(x_{\mathrm{est}}(x_s, s, t))\big)\Big], \tag{1.45}$$

where D_ζ is maximized and θ is minimized, following the original GAN formulation (Goodfellow et al., 2014).

Overall CTM objective. A common combined objective is

$$\mathcal{L}_{\mathrm{CTM}}(\theta) = \mathcal{L}_{\mathrm{consist}}(\theta) + \lambda_{\mathrm{DM}}\, \mathcal{L}_{\mathrm{DM}}(\theta) + \lambda_{\mathrm{GAN}}\, \mathcal{L}_{\mathrm{GAN}}(\theta, \zeta), \tag{1.46}$$

with λ's tuned in practice.

1.5 Mean Flow (MF)

Mean Flow (Geng et al., 2025) is another way to learn a general transition $\Psi_{s\to t}$, inspired by flow matching. Assume a PF-ODE drift $v^*(x_u, u)$ and define the **averaged drift**

$$h^*(x_s, s, t) \triangleq \frac{1}{t-s}\int_s^t v^*(x_u, u)\,\mathrm{d}u, \qquad (t < s). \tag{1.47}$$

Then the flow map can be written exactly as

$$\Psi_{s\to t}(x_s) = x_s + (t-s)\,h^*(x_s, s, t). \tag{1.48}$$

1.5.1 Oracle and practical training objectives

An oracle regression objective is

$$\mathcal{L}_{\text{oracle-MF}}(\theta) \triangleq \mathbb{E}_s\,\mathbb{E}_{t\sim\text{Unif}[0,s]}\,\mathbb{E}_{x_s\sim p_s}\Big[\,w(s)\,\|h_\theta(x_s, s, t) - h^*(x_s, s, t)\|_2^2\Big]. \tag{1.49}$$

When $t \to s$, $h^*(x_s, s, t) \to v^*(x_s, s)$, so (1.49) reduces to the usual flow-matching/velocity regression.

To avoid direct access to h^*, MF uses a consistency-style identity. Differentiating $(t-s)h^*(x_s, s, t) = \int_s^t v^*(x_u, u)\,\mathrm{d}u$ with respect to s yields

$$h^*(x_s, s, t) = v^*(x_s, s) - (s-t)\,\frac{\mathrm{d}}{\mathrm{d}s}h^*(x_s, s, t), \tag{1.50}$$

where the total derivative is

$$\frac{\mathrm{d}}{\mathrm{d}s}h^*(x_s, s, t) = \partial_s h^* + (\nabla_x h^*)\,v^*(x_s, s). \tag{1.51}$$

Replacing h^* by a stop-gradient copy h_{θ^-} gives the practical MF target

$$h^{\text{tgt}}_{\theta^-}(x_s, s, t) \triangleq v^*(x_s, s) - (s-t)\Big[\big(\nabla_x h_{\theta^-}(x_s, s, t)\big)\,v^*(x_s, s) + \partial_s h_{\theta^-}(x_s, s, t)\Big]. \tag{1.52}$$

Then the MF training loss is

$$\mathcal{L}_{\text{MF}}(\theta) \triangleq \mathbb{E}_{t<s,x_s}\Big[\,w(s)\,\|h_\theta(x_s, s, t) - h^{\text{tgt}}_{\theta^-}(x_s, s, t)\|_2^2\Big]. \tag{1.53}$$

As before, v^* can be approximated either by a pretrained diffusion teacher via distillation or by a one-point estimator from scratch.

1.5.2 Sampling

From (1.48), MF sampling uses

$$x_t \approx x_s + (t-s)\,h_{\theta^*}(x_s, s, t). \tag{1.54}$$

In particular, one-step generation is

$$x_0 \approx x_1 + (0-1)\,h_{\theta^*}(x_1, 1, 0) = x_1 - h_{\theta^*}(x_1, 1, 0). \tag{1.55}$$

Multi-step variants can be constructed analogously to CTM.

1.5.3 Relationship between MF and CTM

CTM and MF both model the same flow map, but with different parameterizations. From (1.29) and (1.47), we have the identity

$$g^*(x_s, s, t) = x_s - s\, h^*(x_s, s, t). \tag{1.56}$$

This suggests the parameterization link

$$g_\theta(x_s, s, t) \approx x_s - s\, h_\theta(x_s, s, t). \tag{1.57}$$

Moreover, substituting (1.57) into the CTM objective yields (up to a schedule-dependent weight) the MF objective; see also Geng et al. (2025) for discussion of this equivalence.

1.6 Drifting models at training time

The recent work *Generative Modeling via Drifting* (Deng et al., 2026) develops a different one-step paradigm, called **drifting models**. The word *drift* there does *not* mean the PF-ODE drift $v^*(x,t)$ from (1.2). Instead, it refers to the *training-time movement* of generated samples as the generator parameters are updated.

Basic idea. Let $f_\theta(\varepsilon)$ be a one-shot generator with input noise $\varepsilon \sim p_{\text{prior}}$, and let

$$x = f_\theta(\varepsilon), \qquad \varepsilon \sim p_{\text{prior}}, \qquad x \sim q_\theta \tag{1.58}$$

define the corresponding model distribution. During (stochastic) gradient descent, the parameters evolve as $\theta_0, \theta_1, \theta_2, \ldots$, and therefore the generated samples $x_i = f_{\theta_i}(\varepsilon)$ and their distributions q_{θ_i} also evolve. Drift modeling makes this training-time evolution the central object of study.

Drifting field. Deng et al. (2026) introduce a field

$$V_{p,q}(x) : \mathbb{R}^d \to \mathbb{R}^d, \tag{1.59}$$

depending on the target distribution p and the current model distribution q, and interprets training as approximately following

$$x_{i+1} = x_i + V_{p,q_i}(x_i), \qquad x_i \sim q_i. \tag{1.60}$$

The desired equilibrium is $q_i = p$, at which point the drift should vanish. A sufficient structural condition is **anti-symmetry**:

$$V_{p,q}(x) = -V_{q,p}(x) \qquad \Longrightarrow \qquad p = q \;\Rightarrow\; V_{p,q}(x) = 0. \tag{1.61}$$

A stop-gradient training objective. The paper turns this fixed-point idea into a loss by matching the current prediction to a frozen "drifted" target (Deng et al., 2026):

$$\mathcal{L}_{\text{drift}}(\theta) \triangleq \mathbb{E}_\varepsilon\Big[\big\| f_\theta(\varepsilon) - \text{sg}\big(f_\theta(\varepsilon) + V_{p,q_\theta}(f_\theta(\varepsilon))\big)\big\|_2^2\Big], \tag{1.62}$$

where $\text{sg}(\cdot)$ denotes stop-gradient. At the level of loss values,

$$\mathcal{L}_{\text{drift}}(\theta) = \mathbb{E}_\varepsilon\big[\|V_{p,q_\theta}(f_\theta(\varepsilon))\|_2^2\big], \tag{1.63}$$

so training seeks a generator whose samples have small training-time drift.

Figure 3: Training-time evolution in drifting models. Starting from different initial model distributions q, optimization progressively moves the generator distribution toward the target data distribution p. This illustrates the central idea that the pushforward distribution is evolved during training rather than through iterative inference-time sampling.

Kernelized attraction–repulsion field. One concrete construction in Deng et al. (2026) uses positive samples $y^+ \sim p$ and negative samples $y^- \sim q$ to define a kernelized attraction–repulsion field inspired by mean shift (Cheng, 1995; Deng et al., 2026):

$$V_{p,q}(x) = V_p^+(x) - V_q^-(x), \tag{1.64}$$

with

$$V_p^+(x) = \frac{1}{Z_p(x)} \mathbb{E}_{y^+ \sim p}\big[k(x, y^+)(y^+ - x)\big], \tag{1.65}$$

$$V_q^-(x) = \frac{1}{Z_q(x)} \mathbb{E}_{y^- \sim q}\big[k(x, y^-)(y^- - x)\big]. \tag{1.66}$$

Thus each generated point is pulled toward nearby data samples and pushed away from nearby generated samples. In practice, this drift can also be computed in a learned feature space rather than directly in pixel space.

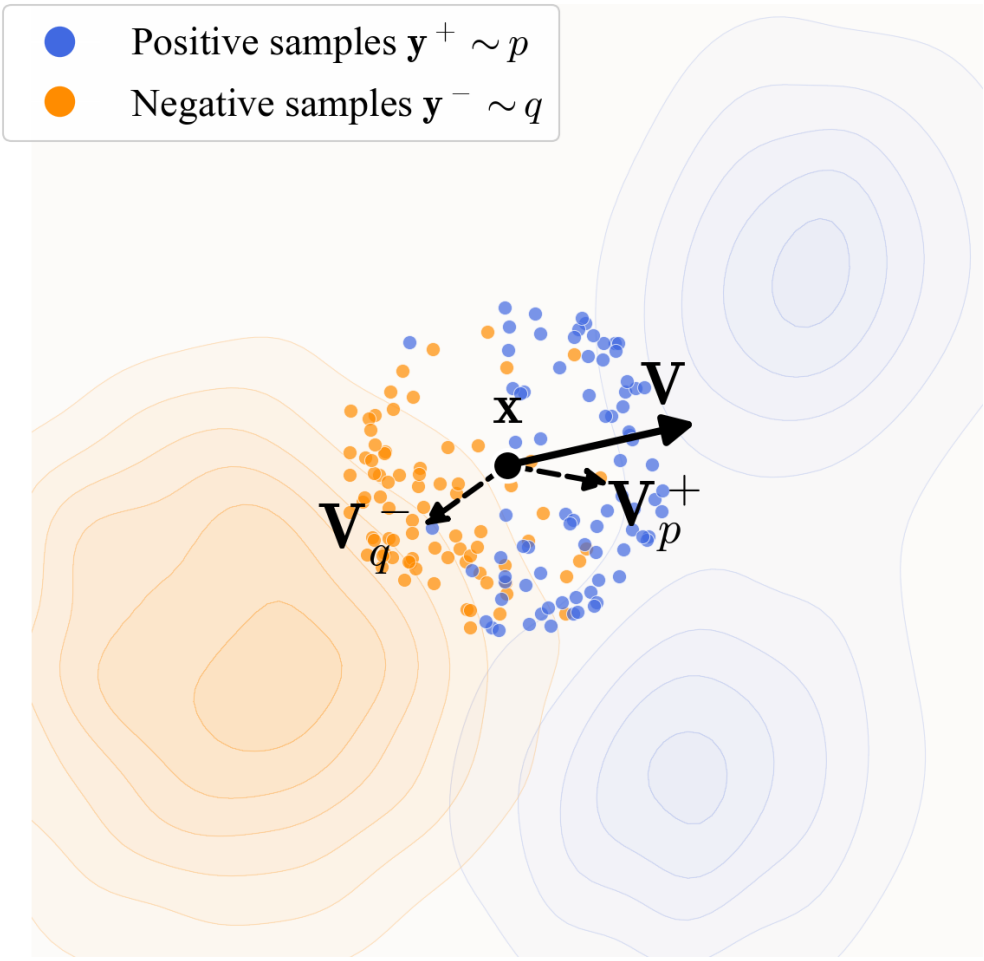

Figure 4: Illustration of a drifting field. A generated point x receives an attractive component from nearby positive samples $y^+ \sim p$ and a repulsive component from nearby negative samples $y^- \sim q$, producing the net update $V = V_p^+ - V_q^-$.

Sampling. Once training is finished, generation is just

$$x = f_{\theta^*}(\varepsilon), \qquad \varepsilon \sim p_{\text{prior}}, \tag{1.67}$$

with no time variable and no ODE/SDE solver at inference time. The iterative process has been shifted entirely into training.

1.7 Takeaways

- **Flow-map viewpoint:** Sampling is evaluating a PF-ODE flow map $\Psi_{s\to t}$. Training-based acceleration learns (approximations to) this map.
- **Continous-time Consistency Models (CM):** learn the special map $\Psi_{s\to 0}$, wheresmall-step consistency leads to tangent objectives; stabilizations such as sCM make training feasible. The original discrete CM / CD constructions can be treated as finite-grid discretizations of the continuous-time CM.

- **CTM and MF:** learn general transitions $\Psi_{s\to t}$. CTM emphasizes semigroup consistency; MF emphasizes averaged drift.
- **Drift modeling:** a related one-step paradigm can instead evolve the generator's pushforward distribution during training via a drifting field $V_{p,q}$. This is conceptually distinct from PF-ODE-based drift modeling, even though both target efficient generation.
- **Distillation vs. from scratch:** the same losses can use either (i) a diffusion teacher-driven solver or (ii) one-point/self-induced surrogates.

References

Cheng, Y. (1995). Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):790–799.

Deng, M., Li, H., Li, T., Du, Y., and He, K. (2026). Generative modeling via drifting. Unpublished manuscript; source files in the source2 folder of this repository.

Geng, Z., Deng, M., Bai, X., Kolter, J. Z., and He, K. (2025). Mean flow: A principled approach to one-step generation from diffusion models.

Geng, Z., Pokle, A., Luo, W., Lin, J., and Kolter, J. Z. (2024). Consistency models made easy. *arXiv preprint arXiv:2406.14548*.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems*.

Karras, T., Aittala, M., Aila, T., Laine, S., and Lehtinen, J. (2022). Elucidating the design space of diffusion-based generative models.

Kim, D., Huang, H., Lai, Y., Liu, B., Kim, H., Leskovec, J., and Ermon, S. (2024). Consistency trajectory models: Learning probability flow ode trajectory of diffusion.

Lu, C. and Song, Y. (2024). Simplifying, stabilizing, and scaling continuous-time consistency models.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707.

Song, Y., Dhariwal, P., Chen, M., and Sutskever, I. (2023). Consistency models. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*.

A Appendix

A.1 Proof of Proposition 1

Proof of Proposition 1. We prove the claim under the standard idealized stop-gradient setting

$$f_{\theta^-}(x,s) = f_\theta(x,s) \quad \text{pointwise},$$

while θ^- is treated as constant when differentiating with respect to θ. More generally, the same argument works if $f_\theta(x,s) - f_{\theta^-}(x,s) = o(\Delta s)$ uniformly as $\Delta s \to 0$.

For brevity, define

$$T_{\Delta s}(x_s, s) \triangleq f_{\theta^-}\big(\Psi_{s\to s-\Delta s}(x_s),\, s - \Delta s\big).$$

Then

$$\mathcal{L}_{\mathrm{CM}}^{\Delta s}(\theta, \theta^-) = \mathbb{E}_{s,x_s}\Big[\omega(s)\, \|f_\theta(x_s,s) - T_{\Delta s}(x_s,s)\|_2^2\Big].$$

Since x_u follows the PF-ODE $\dot{x}_u = v^*(x_u, u)$, for a backward step of size Δs we have

$$\Psi_{s\to s-\Delta s}(x_s) = x_s - \Delta s\, v^*(x_s, s) + o(\Delta s).$$

A first-order Taylor expansion of $f_{\theta^-}(x,s)$ around (x_s, s) gives

$$\begin{aligned} T_{\Delta s}(x_s,s) &= f_{\theta^-}(x_s,s) - \Delta s\, \partial_s f_{\theta^-}(x_s,s) - \Delta s\, \big(\nabla_x f_{\theta^-}(x_s,s)\big)\, v^*(x_s,s) + o(\Delta s) \\ &= f_{\theta^-}(x_s,s) - \Delta s\, \frac{\mathrm{d}}{\mathrm{d}s} f_{\theta^-}(x_s,s) + o(\Delta s), \end{aligned}$$

where

$$\frac{\mathrm{d}}{\mathrm{d}s} f_{\theta^-}(x_s,s) = \partial_s f_{\theta^-}(x_s,s) + \big(\nabla_x f_{\theta^-}(x_s,s)\big)\, v^*(x_s,s).$$

Hence

$$f_\theta(x_s,s) - T_{\Delta s}(x_s,s) = \big(f_\theta(x_s,s) - f_{\theta^-}(x_s,s)\big) + \Delta s\, \frac{\mathrm{d}}{\mathrm{d}s} f_{\theta^-}(x_s,s) + o(\Delta s).$$

Under the stop-gradient-sharing assumption, the first term vanishes, so

$$f_\theta(x_s,s) - T_{\Delta s}(x_s,s) = \Delta s\, \frac{\mathrm{d}}{\mathrm{d}s} f_{\theta^-}(x_s,s) + o(\Delta s).$$

Now differentiate the loss with respect to θ. Since $T_{\Delta s}$ depends only on the stop-gradient parameters θ^-, it is constant with respect to θ, and therefore

$$\nabla_\theta \mathcal{L}_{\mathrm{CM}}^{\Delta s}(\theta,\theta^-) = \mathbb{E}_{s,x_s}\Big[2\,\omega(s)\,\big(\nabla_\theta f_\theta(x_s,s)\big)^\top \big(f_\theta(x_s,s) - T_{\Delta s}(x_s,s)\big)\Big].$$

Substituting the expansion above yields

$$\frac{1}{\Delta s}\nabla_\theta \mathcal{L}_{\mathrm{CM}}^{\Delta s}(\theta,\theta^-) = \mathbb{E}_{s,x_s}\Big[2\,\omega(s)\,\big(\nabla_\theta f_\theta(x_s,s)\big)^\top \frac{\mathrm{d}}{\mathrm{d}s} f_{\theta^-}(x_s,s)\Big] + o(1).$$

Under standard regularity and dominated-convergence assumptions, taking $\Delta s \to 0$ gives

$$\lim_{\Delta s\to 0} \frac{1}{\Delta s}\nabla_\theta \mathcal{L}_{\mathrm{CM}}^{\Delta s}(\theta,\theta^-) = \mathbb{E}_{s,x_s}\Big[2\,\omega(s)\,\big(\nabla_\theta f_\theta(x_s,s)\big)^\top \frac{\mathrm{d}}{\mathrm{d}s} f_{\theta^-}(x_s,s)\Big].$$

Finally, because the total derivative $\frac{\mathrm{d}}{\mathrm{d}s} f_{\theta^-}(x_s,s)$ is independent of θ, the right-hand side is exactly

$$\nabla_\theta \mathbb{E}_{s,x_s}\Big[2\,\omega(s)\, f_\theta(x_s,s)^\top \frac{\mathrm{d}}{\mathrm{d}s} f_{\theta^-}(x_s,s)\Big] = \nabla_\theta \mathcal{L}_{\mathrm{CM}}^{\infty}(\theta,\theta^-).$$

This proves the proposition. □

A.2 Proof of Proposition 2

Proof of Proposition 2. Recall that

$$g^*(x_s, s, t) = x_s + \frac{s}{s-t}\int_s^t v^*(x_u, u)\,\mathrm{d}u = x_s - s\,\frac{1}{t-s}\int_s^t v^*(x_u, u)\,\mathrm{d}u. \tag{A.1}$$

For part (i), since $u \mapsto v^*(x_u, u)$ is continuous at $u = s$, we have

$$\frac{1}{t-s}\int_s^t v^*(x_u, u)\,\mathrm{d}u \to v^*(x_s, s) \qquad \text{as } t \to s.$$

Applying this to (A.1) gives

$$\lim_{t\to s} g^*(x_s, s, t) = x_s - s\,v^*(x_s, s),$$

which proves (1.31).

For part (ii), write

$$\begin{aligned} g^*(x_s, s, t) - \big(x_s - s\,v^*(x_s, s)\big) &= -s\left(\frac{1}{t-s}\int_s^t v^*(x_u, u)\,\mathrm{d}u - v^*(x_s, s)\right) \\ &= -\frac{s}{t-s}\int_s^t \big(v^*(x_u, u) - v^*(x_s, s)\big)\,\mathrm{d}u. \end{aligned}$$

Using the local Lipschitz assumption,

$$\|v^*(x_u, u) - v^*(x_s, s)\| \le C|u - s|,$$

we obtain

$$\begin{aligned} \big\|g^*(x_s, s, t) - \big(x_s - s\,v^*(x_s, s)\big)\big\| &\le \frac{s}{|t-s|}\int_s^t C|u-s|\,\mathrm{d}u \\ &\le \frac{sC}{|t-s|}\cdot\frac{|t-s|^2}{2} \\ &= \frac{sC}{2}|t-s|. \end{aligned}$$

Hence

$$g^*(x_s, s, t) = x_s - s\,v^*(x_s, s) + \mathcal{O}(|t-s|),$$

which proves (1.32). □