Faster Sampling: Distillation

Lecture 09

Qiang Sun

Outline

Motivation and Viewpoints

Why distillation?

Sampling from flow and diffusion models is expensive because generation is *iterative*.

- Strong training-free ODE solvers often still need about $10-20$ network evaluations.
- Naive diffusion discretizations are much slower; original DDPM sampling used roughly 1000 steps.
- **Distillation** shifts compute to training time: a slow teacher supervises a fast student.

Key Idea (Inference-time promise)

Use the teacher only during training, then sample with the student in *few steps* — ideally even *one step*.

The reconstruction / x_0 perspective

Under the linear-Gaussian noising model,

$$x_t \mid x_0 \sim \mathcal{N}(\alpha_t x_0,\, \beta_t^2 I_d), \qquad 0 \le t \le 1,$$

a diffusion model can be viewed as a family of denoisers.

Key Idea (Tweedie + reconstruction)

$$x_0^*(x,t) = \mathbb{E}[X_0 \mid X_t = x] = \frac{1}{\alpha_t}\Big(x + \beta_t^2\, \nabla_x \log p_t(x)\Big).$$

If a model predicts noise $\varepsilon_\theta(x,t)$, then

$$x_{0,\theta}(x,t) = \frac{x - \beta_t \varepsilon_\theta(x,t)}{\alpha_t}.$$

So distillation can be interpreted as learning to recover x_0 — or equivalently the score / noise — in *many fewer steps*.

Two distillation paradigms

Distribution-level

Match the student's output distribution to the teacher / data distribution.

$$\min_{\theta} \; D\left(p_0^{\theta} \,\|\, p_0^{\phi^\star}\right).$$

Examples: DMD, VSD, SiD.

Flow-map-level

Learn coarse transport maps between time marginals.

$$G_\theta(\cdot, s, t) \approx \Psi_{s \to t}.$$

Examples: KD, progressive distillation (PD).

Chronology: **KD/PD** appeared earlier; **distribution-level** methods became prominent later.

A unifying flow-map view

The probability-flow ODE is

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

and its oracle flow map is

$$\Psi_{s\to t}(x_s) = x_s + \int_s^t v^\star(x_\tau, \tau)\,\mathrm{d}\tau.$$

In practice $\Psi_{s\to t}$ is unknown, so we replace it with a teacher-induced transition map

$$\mathrm{Solver}_{s\to t}(x_s; \phi^\star).$$

- Ideal generation: draw $x_1 \sim p_{\text{init}}$ and apply $\Psi_{1\to 0}$.
- Distillation asks whether teacher transitions can supervise a faster student map.
- This is the common lens behind both KD and PD.

KD and the oracle objective

Knowledge distillation (KD): regress a one-step student to a long teacher rollout,

$$\mathcal{L}_{\mathrm{KD}}(\theta) = \mathbb{E}_{x_1 \sim p_{\mathrm{init}}}\Big[\| G_\theta(x_1) - \mathrm{Solver}^{\circ}_{1\to 0}(x_1; \phi^\star)\|_2^2\Big].$$

A more general ideal objective is

$$\mathcal{L}_{\mathrm{oracle}}(\theta) = \mathbb{E}_{(s,t)\sim\pi}\mathbb{E}_{X_s\sim p_s}\Big[w(s,t)\, d\big(G_\theta(X_s, s, t), \Psi_{s\to t}(X_s)\big)\Big].$$

Key Idea (Unification)

- **KD**: put all mass of π on $(s,t) = (1,0)$.
- **PD**: use many *local* pairs (s,t) and match short teacher fragments.

Distribution-Level Distillation

Distribution-level setup and the VSD objective

Let the student generator be $\hat{x}_0 = G_\theta(z)$ with $z \sim p_{\text{init}}$, so $\hat{x}_0 \sim p_0^\theta$.
Using the same Gaussian noising process as the teacher,

$$\hat{x}_t = \alpha_t G_\theta(z) + \beta_t \varepsilon, \qquad z \sim p_{\text{init}},\ \varepsilon \sim \mathcal{N}(0, I_d), \qquad \hat{x}_t \sim p_t^\theta.$$

Variational score distillation (VSD)

$$\mathcal{L}_{\text{VSD}}(\theta) = \mathbb{E}_{t\sim p(t)}\Big[\omega(t)\, D_{\text{KL}}\big(p_t^\theta \,\|\, p_t\big)\Big].$$

Matching *noisy marginals across time* forces the student distribution to align with the teacher distribution.

KL gives a score-difference gradient

Proposition (VSD gradient)

For $\hat{x}_t = \alpha_t G_\theta(z) + \beta_t \varepsilon$,

$$\nabla_\theta \mathcal{L}_{\mathrm{VSD}}(\theta) = \mathbb{E}_{t,z,\varepsilon}\Big[\omega(t)\,\alpha_t\,\big(\nabla_x \log p_t^\theta(\hat{x}_t) - \nabla_x \log p_t(\hat{x}_t)\big)^\top \nabla_\theta G_\theta(z)\Big].$$

- **Teacher score** $\nabla_x \log p_t(x)$: provided by a pretrained diffusion model.
- **Student score** $\nabla_x \log p_t^\theta(x)$: depends on the current generator and must be estimated.

Key Idea (Why KL is special)

KL yields a **clean score-difference signal**. Generic f-divergences or Renyi divergences usually introduce unknown density-ratio terms.

Bilevel training: estimate the student score

Because $\hat{x}_t \mid \hat{x}_0$ is Gaussian,

$$\nabla_x \log p_t(\hat{x}_t \mid \hat{x}_0) = -\frac{\hat{x}_t - \alpha_t \hat{x}_0}{\beta_t^2}.$$

This lets us train an auxiliary score network $s_\zeta(x, t)$ on student-generated samples via denoising score matching:

$$\mathcal{L}_{\mathrm{DSM}}(\zeta; \theta) = \mathbb{E}_{t,z,\varepsilon}\left[\left\| s_\zeta(\hat{x}_t, t) + \tfrac{\hat{x}_t - \alpha_t \hat{x}_0}{\beta_t^2} \right\|_2^2\right].$$

Then update the generator using the approximate gradient

$$\nabla_\theta \mathcal{L}_{\mathrm{VSD}}(\theta) \approx \mathbb{E}_{t,z,\varepsilon}\Big[\omega(t)\, \alpha_t \left(s_\zeta(\hat{x}_t, t) - s_{\mathrm{teacher}}(\hat{x}_t, t)\right)^\top \nabla_\theta G_\theta(z)\Big].$$

So training alternates between **score estimation** (update ζ) and **generator update** (update θ).

VSD training recipe and why it works

Practical loop

1. Sample t, latent z, and Gaussian noise ε.
2. Form $\hat{x}_0 = G_\theta(z)$ and $\hat{x}_t = \alpha_t \hat{x}_0 + \beta_t \varepsilon$.
3. Update s_ζ with DSM on student samples.
4. Update G_θ using the score-difference signal.

Key Idea (Why this works)

If student and teacher scores match over time, then $p_t^\theta \approx p_t$. Because Gaussian noising is injective, this propagates back to $p_0^\theta \approx p_0$.

Beyond KL and an important 3D-generation application

Beyond KL. Replacing D_{KL} by a generic f-divergence or Renyi divergence usually leads to expectations weighted by unknown density ratios p_t^θ / p_t. In practice this often means a critic / discriminator or explicit ratio estimation.

VSD for 3D generation. Let θ parametrize a 3D scene and let a differentiable renderer produce

$$\hat{x}_0 = \mathcal{R}(\theta), \qquad \hat{x}_t = \alpha_t \mathcal{R}(\theta) + \beta_t \varepsilon.$$

A score-alignment objective is

$$\mathcal{L}_{\mathrm{VSD}}^{\mathrm{3D}}(\theta) = \frac{1}{2}\mathbb{E}_{t,\varepsilon}\Big[\omega(t)\|s_\zeta(\hat{x}_t, t) - s_{\mathrm{teacher}}(\hat{x}_t, t \mid c)\|_2^2\Big].$$

Using a stop-gradient surrogate gives the practical update

$$\nabla_\theta \tilde{\mathcal{L}}_{\mathrm{VSD}}^{\mathrm{3D}}(\theta) = \mathbb{E}\Big[\omega(t)\alpha_t \big(s_\zeta - s_{\mathrm{teacher}}\big)^\top \nabla_\theta \mathcal{R}(\theta)\Big].$$

Setting $s_\zeta \equiv 0$ recovers the familiar **SDS** gradient used in text-to-3D pipelines.

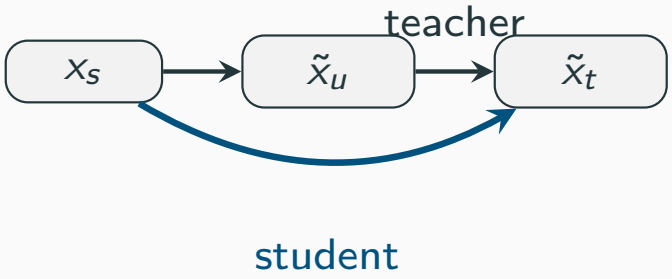
Flow-Map-Level Distillation

Progressive distillation starts from a DDIM map

For deterministic samplers, a DDIM transition from time s to t using an x_0-predictor $\hat{x}_0(\cdot, s)$ is

$$\mathrm{DDIM}_{s\to t}(x_s; \hat{x}_0) = \frac{\beta_t}{\beta_s} x_s + \alpha_s \left(\frac{\alpha_t}{\alpha_s} - \frac{\beta_t}{\beta_s} \right) \hat{x}_0(x_s, s).$$

- PD distills *two teacher DDIM steps into one student step*.
- Fix three times $s > u > t$.
- Teacher produces $\tilde{x}_u$ then $\tilde{x}_t$.
- Student should jump directly from x_s to $\tilde{x}_t$.



Two teacher steps into one student target

Define a pseudo-target $\tilde{x}$ so that a single DDIM step from s to t matches the teacher's two-step output:

$$\tilde{x}_t = \mathrm{DDIM}_{s\to t}(x_s; \tilde{x}) = \frac{\beta_t}{\beta_s} x_s + \alpha_s \Big(\frac{\alpha_t}{\alpha_s} - \frac{\beta_t}{\beta_s} \Big) \tilde{x}.$$

Solving for $\tilde{x}$ yields the closed-form target

$$\tilde{x} = \frac{\beta_s}{\alpha_t \beta_s - \alpha_s \beta_t} \, \tilde{x}_t - \frac{\beta_t}{\alpha_t \beta_s - \alpha_s \beta_t} \, x_s.$$

The student denoiser $f_\theta(x_s, s)$ is then trained to regress to this pseudo-target.

PD objective and training pipeline

On a discrete grid $t_0 = 1 > t_1 > \cdots > t_N = 0$, progressive distillation uses

$$\min_{\theta} \ \mathbb{E}_k \mathbb{E}_{x_{t_k} \sim p_{t_k}} \left[w(\lambda_{t_k}) \, \| f_\theta(x_{t_k}, t_k) - \tilde{x}^{(k)} \|_2^2 \right], \qquad \lambda_t = \log(\alpha_t / \beta_t).$$

Key Idea (Why it is progressive)

1. Train a student so one step matches two teacher steps.
2. Replace the teacher by the student.
3. Keep every other time point and repeat.

Each round halves the number of sampling steps:

$$N \to N/2 \to N/4 \to \cdots$$

PD as local semigroup matching

The oracle flow maps satisfy the semigroup property

$$\Psi_{s\to t} = \Psi_{u\to t} \circ \Psi_{s\to u}.$$

Progressive distillation enforces this *locally* by requiring one long student step to match two short teacher steps:

$$\mathbb{E}_s \mathbb{E}_{x_s\sim p_s} \left\| G_\theta(x_s, s, s-2\Delta s) - \mathrm{Solver}_{s-\Delta s\to s-2\Delta s}\left(\mathrm{Solver}_{s\to s-\Delta s}(x_s)\right) \right\|_2^2.$$

So PD never needs the full teacher rollout as a target — it only matches **short teacher fragments** while recursively coarsening the grid.

Practical extensions of flow-map distillation

- **Other solvers:** the DDIM pseudo-target is analytic because the update is affine in the x_0-prediction. For more general solvers, one can still match transition maps directly.
- **Stochastic samplers:** freeze the random seed per example so the teacher transition becomes deterministic during regression.
- **Classifier-free guidance:** per-step CFG doubles compute. A practical recipe is:
  1. distill guided and unguided predictions into a single network that takes guidance weight ω as input;
  2. then apply PD on that single-network teacher to reduce the number of steps.

Result: a sampler with *one network call per step* and a very small number of steps.

Takeaways

Takeaways

- Distillation is the main **training-based** route to fewer function evaluations.
- **Distribution-level methods** (e.g. VSD) match teacher and student distributions; with KL, the update is a clean *score-difference* signal.
- **Flow-map-level methods** (KD / PD) learn transport maps directly; PD is powerful because it recursively halves the number of sampling steps.
- VSD also supports 3D generation, while PD combines naturally with guidance distillation for practical text-to-image systems.

Big picture

more teacher-guided training $\Longrightarrow$ **much cheaper inference.**

Next: limits of distillation, coarse-grid error accumulation, and possible corrections.