Training Flow and Diffusion Models

Lecture 05

Qiang Sun

Reminder: Flow and Diffusion Models

Flow Model

Initialize: $X_0 \sim p_{\text{init}},$

ODE: $\mathrm{d}X_t = u_t^\theta(X_t)\mathrm{d}t$

E.g. Gaussian

Neural network vector field

Diffusion coeff.

Diffusion Model

Initialize: $X_0 \sim p_{\text{init}},$

SDE: $\mathrm{d}X_t = u_t^\theta(X_t)\mathrm{d}t + \sigma_t \mathrm{d}W_t$

To get samples, simulate ODE/SDE from t=0 to t=1 and return X_1

Conditional Prob. Path, Vector Field, and Score

	Notation	Key property	Gaussian example
Conditional Probability Path	$p_t(\cdot\|z)$	Interpolates p_{init} and a data point z	$\mathcal{N}(\alpha_t z, \beta_t^2 I_d)$
Conditional Vector Field	$u_t^{\text{target}}(x\|z)$	ODE follows conditional path	$\left(\dot{\alpha}_t - \frac{\dot{\beta}_t}{\beta_t}\alpha_t\right) z + \frac{\dot{\beta}_t}{\beta_t} x$
Conditional Score Function	$\nabla \log p_t(x\|z)$	Gradient of log-likelihood	$-\frac{x - \alpha_t z}{\beta_t^2}$

Marginal Prob. Path, Vector Field, and Score

	Notation	Key property	Formula
Marginal Probability Path	p_t	Interpolates p_{init} and p_{data}	$\int p_t(x\|z)p_{\text{data}}(z)\mathrm{d}z$
Marginal Vector Field	$u_t^{\text{target}}(x)$	ODE follows marginal path	$\int u_t^{\text{target}}(x\|z)\frac{p_t(x\|z)p_{\text{data}}(z)}{p_t(x)}\mathrm{d}z$
Marginal Score Function	$\nabla \log p_t(x)$	Can be used to convert ODE target to SDE	$\int \nabla \log p_t(x\|z)\frac{p_t(x\|z)p_{\text{data}}(z)}{p_t(x)}\mathrm{d}z$

Flow Matching

Flow model (ODE)

Consider the flow model

$$X_0 \sim p_{\text{init}}, \qquad \mathrm{d}X_t = u_t^\theta(X_t)\,\mathrm{d}t, \qquad t \in [0,1].$$

We want parameters θ so that

$$u_t^\theta \approx u_t^{\text{target}} \quad \Longrightarrow \quad X_1 \sim p_{\text{data}}.$$

Flow matching loss

Let $\mathrm{Unif} = \mathrm{Unif}[0,1]$. Define the *flow matching loss*

$$\begin{aligned}\mathcal{L}_{\mathrm{FM}}(\theta) &= \mathbb{E}_{t\sim\mathrm{Unif},\, x\sim p_t}\Big[\|u_t^\theta(x) - u_t^{\mathrm{target}}(x)\|^2\Big] \\ &= \mathbb{E}_{t\sim\mathrm{Unif},\, z\sim p_{\mathrm{data}},\, x\sim p_t(\cdot|z)}\Big[\|u_t^\theta(x) - u_t^{\mathrm{target}}(x)\|^2\Big].\end{aligned}$$

Intuition:

- sample $t \in [0,1]$,
- sample $z \sim p_{\mathrm{data}}$,
- sample $x \sim p_t(\cdot \mid z)$,
- regress $u_t^\theta(x)$ toward the (marginal) target $u_t^{\mathrm{target}}(x)$.

Intractability of the marginal target

We know the marginalization identity

$$u_t^{\text{target}}(x) = \int u_t^{\text{target}}(x \mid z) \, \frac{p_t(x \mid z) \, p_{\text{data}}(z)}{p_t(x)} \, \mathrm{d}z,$$

but the integral is typically intractable.

Define the tractable *conditional flow matching loss*

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t \sim \text{Unif},\, z \sim p_{\text{data}},\, x \sim p_t(\cdot \mid z)} \left[\left\| u_t^\theta(x) - u_t^{\text{target}}(x \mid z) \right\|^2 \right].$$

Algorithm 3 Flow Matching Training Procedure (General)

Require: A dataset of samples $z \sim p_{\text{data}}$, neural network u_t^θ

1: **for** each mini-batch of data **do**
2: Sample a data example z from the dataset.
3: Sample a random time $t \sim \text{Unif}_{[0,1]}$.
4: Sample $x \sim p_t(\cdot|z)$
5: Compute loss

$$\mathcal{L}(\theta) = \|u_t^\theta(x) - u_t^{\text{target}}(x|z)\|^2$$

6: Update the model parameters θ via gradient descent on $\mathcal{L}(\theta)$
7: **end for**

Marginal vs. conditional flow matching

Theorem (Marginal vs. conditional flow matching)

The marginal flow matching loss equals the conditional flow matching loss up to a constant:

$$\mathcal{L}_{\mathrm{FM}}(\theta) = \mathcal{L}_{\mathrm{CFM}}(\theta) + C,$$

where C is independent of θ. In particular,

$$\nabla_\theta \mathcal{L}_{\mathrm{FM}}(\theta) = \nabla_\theta \mathcal{L}_{\mathrm{CFM}}(\theta).$$

Hence, minimizing $\mathcal{L}_{\mathrm{CFM}}$ is equivalent to minimizing $\mathcal{L}_{\mathrm{FM}}$.

Proof sketch (why the gradients match)

- Expand $\|a-b\|^2 = \|a\|^2 - 2a^\top b + \|b\|^2$.
- Terms involving $\|u_t^{\text{target}}\|^2$ do not depend on $\theta \Rightarrow$ constants.
- The cross-term uses the marginalization identity for $u_t^{\text{target}}(x)$ and swaps integrals:

$$\mathbb{E}_{x\sim p_t}\big[u_t^\theta(x)^\top u_t^{\text{target}}(x)\big] = \mathbb{E}_{z\sim p_{\text{data}},\, x\sim p_t(\cdot|z)}\big[u_t^\theta(x)^\top u_t^{\text{target}}(x \mid z)\big].$$

- This yields $\mathcal{L}_{\text{FM}}(\theta) = \mathcal{L}_{\text{CFM}}(\theta) + C$.

After training: sampling

After training u_t^θ, generate by simulating the learned ODE (e.g. Euler):

$$\mathrm{d}X_t = u_t^\theta(X_t)\,\mathrm{d}t, \qquad X_0 \sim p_{\text{init}} \quad \Rightarrow \quad X_1 \approx p_{\text{data}}.$$

This pipeline is *flow matching* (Lipman et al., 2022; Liu et al., 2022; Lipman et al., 2024; Albergo et al., 2025).

Gaussian conditional probability paths

Consider Gaussian probability paths

$$p_t(\cdot \mid z) = \mathcal{N}(\alpha_t z, \beta_t^2 I_d),$$

sampled by

$$\varepsilon \sim \mathcal{N}(0, I_d) \quad \Rightarrow \quad x_t = \alpha_t z + \beta_t \varepsilon \sim p_t(\cdot \mid z).$$

Conditional target vector field:

$$u_t^{\text{target}}(x \mid z) = \left(\dot{\alpha}_t - \frac{\dot{\beta}_t}{\beta_t} \alpha_t \right) z + \frac{\dot{\beta}_t}{\beta_t} x.$$

Gaussian path: conditional flow matching loss

Plugging in the Gaussian path gives

$$\mathcal{L}_{\mathrm{CFM}}(\theta) = \mathbb{E}_{t\sim\mathrm{Unif},\, z\sim p_{\mathrm{data}},\, \varepsilon\sim\mathcal{N}(0,I_d)} \left[\left\| u_t^\theta(\alpha_t z + \beta_t \varepsilon) - (\dot{\alpha}_t z + \dot{\beta}_t \varepsilon) \right\|^2 \right].$$

CondOT special case: $\alpha_t = t$, $\beta_t = 1 - t \Rightarrow$

$$\mathcal{L}_{\mathrm{CFM}}(\theta) = \mathbb{E}_{t,z,\varepsilon} \left[\left\| u_t^\theta(tz + (1-t)\varepsilon) - (z - \varepsilon) \right\|^2 \right].$$

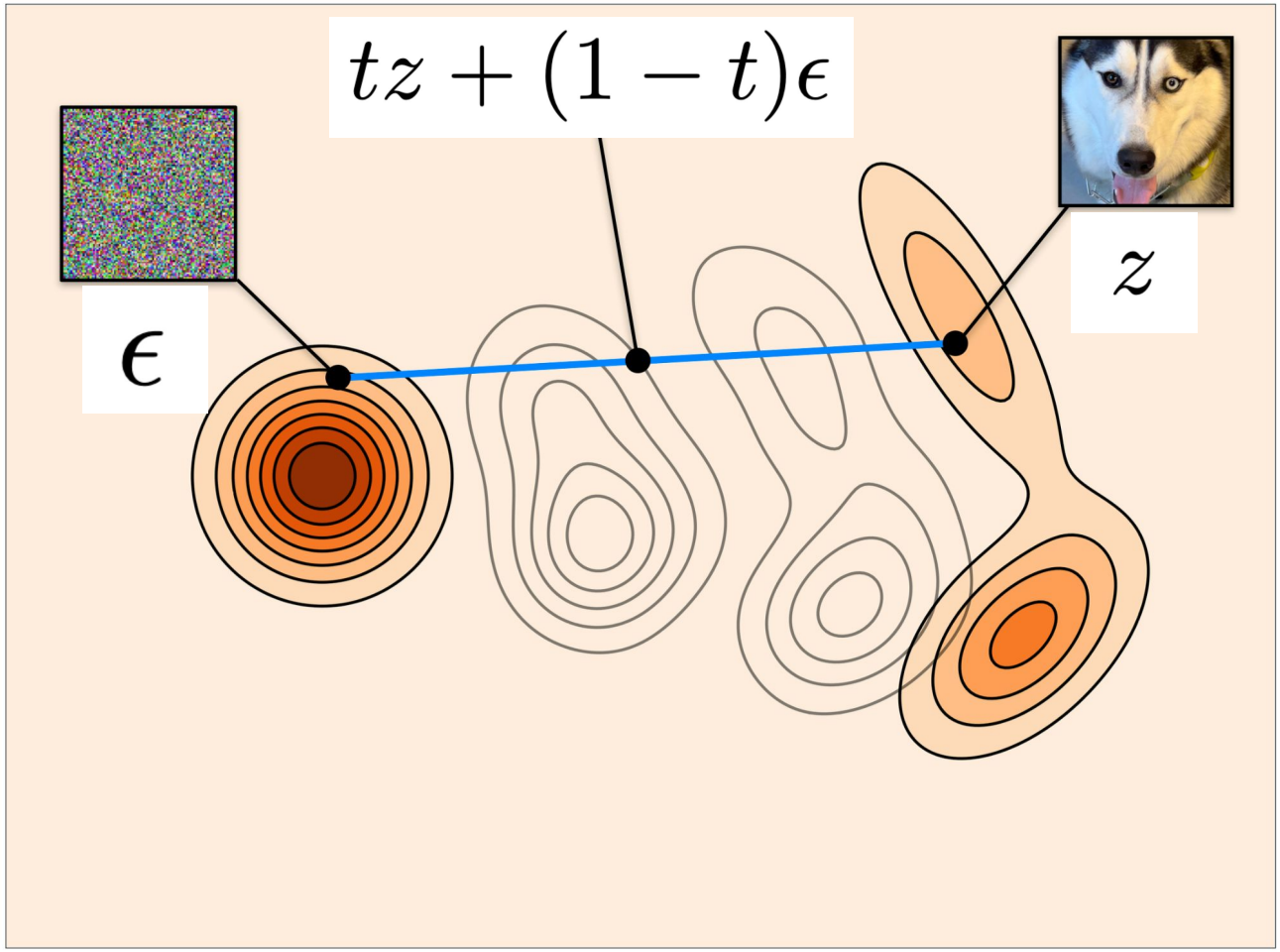

Figure credit: Yaron Lipman

Algorithm: Flow matching training (CondOT Gaussian)

Algorithm 1: Flow Matching Training (Gaussian CondOT path $p_t(x \mid z) = \mathcal{N}(tz, (1-t)^2 I_d)$)

Require: Dataset samples $z \sim p_{\text{data}}$, neural network u_t^θ

for *each mini-batch* **do**

- Sample $z \sim p_{\text{data}}$
- Sample $t \sim \text{Unif}[0, 1]$
- Sample $\varepsilon \sim \mathcal{N}(0, I_d)$
- Set $x \leftarrow tz + (1-t)\varepsilon$ ▷ `general:` $x \sim p_t(\cdot \mid z)$
- Compute $\mathcal{L}(\theta) \leftarrow \|u_t^\theta(x) - (z - \varepsilon)\|^2$ ▷ `general:` $\|u_t^\theta(x) - u_t^{\text{target}}(x \mid z)\|^2$
- Update θ by gradient descent on $\mathcal{L}(\theta)$

Score Matching

Extending to SDEs

Extend the target ODE to an SDE that preserves the same marginals:

$$dX_t = \left[u_t^{\text{target}}(X_t) + \frac{\sigma_t^2}{2} \nabla \log p_t(X_t) \right] dt + \sigma_t \, dW_t, \qquad X_0 \sim p_{\text{init}}.$$

Then $X_t \sim p_t$ for all $t \in [0, 1]$.

Score network and losses

Approximate the marginal score $\nabla \log p_t$ with a score network

$$s_t^\theta : \mathbb{R}^d \times [0,1] \to \mathbb{R}^d.$$

Define:

$$\mathcal{L}_{\mathrm{SM}}(\theta) = \mathbb{E}_{t\sim \mathrm{Unif},\, z\sim p_{\mathrm{data}},\, x\sim p_t(\cdot|z)} \left[\left\| s_t^\theta(x) - \nabla \log p_t(x) \right\|^2 \right],$$

$$\mathcal{L}_{\mathrm{CSM}}(\theta) = \mathbb{E}_{t\sim \mathrm{Unif},\, z\sim p_{\mathrm{data}},\, x\sim p_t(\cdot|z)} \left[\left\| s_t^\theta(x) - \nabla \log p_t(x \mid z) \right\|^2 \right].$$

Marginal vs. conditional score matching

Theorem (Marginal vs. conditional score matching)

$$\mathcal{L}_{\mathrm{SM}}(\theta) = \mathcal{L}_{\mathrm{CSM}}(\theta) + C,$$

where C is independent of θ. Hence

$$\nabla_\theta \mathcal{L}_{\mathrm{SM}}(\theta) = \nabla_\theta \mathcal{L}_{\mathrm{CSM}}(\theta),$$

and at the minimizer $s_t^{\theta^\star} = \nabla \log p_t$.

Sampling after score training

After training, for any diffusion coefficient $\sigma_t \geq 0$, sample via

$$X_0 \sim p_{\text{init}}, \qquad \mathrm{d}X_t = \left(u_t^\theta(X_t) + \frac{\sigma_t^2}{2} s_t^\theta(X_t) \right) \mathrm{d}t + \sigma_t \, \mathrm{d}W_t.$$

In practice, σ_t trades off:

- numerical simulation error (larger noise can help),
- training/model error (too much noise can blur).

Remark: denoising diffusion models

"Denoising diffusion models" correspond to Gaussian probability paths

$$p_t(\cdot \mid z) = \mathcal{N}(\alpha_t z, \beta_t^2 I_d).$$

Early papers often use an inverted time convention (noise increases with t); align formulas by a time reversal / rescaling.

Example: denoising score matching (Gaussian paths)

For $p_t(\cdot \mid z) = \mathcal{N}(\alpha_t z, \beta_t^2 I_d)$,

$$\nabla \log p_t(x \mid z) = -\frac{x - \alpha_t z}{\beta_t^2}.$$

With $x_t = \alpha_t z + \beta_t \varepsilon$, $\varepsilon \sim \mathcal{N}(0, I_d)$,

$$\mathcal{L}_{\mathrm{CSM}}(\theta) = \mathbb{E}_{t,z,\varepsilon}\left[\left\| s_t^\theta(\alpha_t z + \beta_t \varepsilon) + \frac{\varepsilon}{\beta_t} \right\|^2\right].$$

DDPM trick (Ho et al., 2020): drop $1/\beta_t^2$ and reparameterize

$$-\beta_t s_t^\theta(x) = \varepsilon_t^\theta(x), \quad \mathcal{L}_{\mathrm{DDPM}}(\theta) = \mathbb{E}\big[\|\varepsilon_t^\theta(\alpha_t z + \beta_t \varepsilon) - \varepsilon\|^2\big].$$

Algorithm: Score matching training (Gaussian path)

Algorithm 2: Score Matching Training (Gaussian path)

Require: Dataset samples $z \sim p_{\text{data}}$; score net s_t^θ or noise predictor ε_t^θ

for *each mini-batch* **do**

- Sample $z \sim p_{\text{data}}$
- Sample $t \sim \mathrm{Unif}[0, 1]$
- Sample $\varepsilon \sim \mathcal{N}(0, I_d)$
- Set $x_t \leftarrow \alpha_t z + \beta_t \varepsilon$ ▷ `general:` $x_t \sim p_t(\cdot \mid z)$
- Compute $\mathcal{L}(\theta) \leftarrow \left\| s_t^\theta(x_t) + \varepsilon/\beta_t \right\|^2$ ▷ `general:` $\|s_t^\theta - \nabla \log p_t(\cdot \mid z)\|^2$
- ▷ `Optional DDPM parameterization`
- ▷ $\mathcal{L}(\theta) \leftarrow \|\varepsilon_t^\theta(x_t) - \varepsilon\|^2$
- Update θ by gradient descent on $\mathcal{L}(\theta)$

Gaussian conversion: score $\leftrightarrow$ vector field

Conversion formula for Gaussian probability paths

Proposition (Conversion formula for Gaussian probability paths)

For $p_t(\cdot \mid z) = \mathcal{N}(\alpha_t z, \beta_t^2 I_d)$,

$$u_t^{\text{target}}(x \mid z) = \left(\frac{\beta_t^2 \dot{\alpha}_t}{\alpha_t} - \beta_t \dot{\beta}_t \right) \nabla \log p_t(x \mid z) + \frac{\dot{\alpha}_t}{\alpha_t} x,$$

$$u_t^{\text{target}}(x) = \left(\frac{\beta_t^2 \dot{\alpha}_t}{\alpha_t} - \beta_t \dot{\beta}_t \right) \nabla \log p_t(x) + \frac{\dot{\alpha}_t}{\alpha_t} x.$$

The marginal formula corresponds to the *probability flow ODE* viewpoint.

Consequence: parameterize one from the other

We can convert a learned score network into a vector field:

$$u_t^\theta(x) = \left(\frac{\beta_t^2 \dot{\alpha}_t}{\alpha_t} - \beta_t \dot{\beta}_t \right) s_t^\theta(x) + \frac{\dot{\alpha}_t}{\alpha_t} x.$$

Conversely (for $t \in [0, 1)$, the denominator is nonzero),

$$s_t^\theta(x) = \frac{\alpha_t u_t^\theta(x) - \dot{\alpha}_t x}{\beta_t^2 \dot{\alpha}_t - \alpha_t \dot{\beta}_t \beta_t}.$$

Takeaway: for Gaussian probability paths, you do *not* need to train both u_t^θ and s_t^θ separately.

Comparing the learned score vs. the converted score

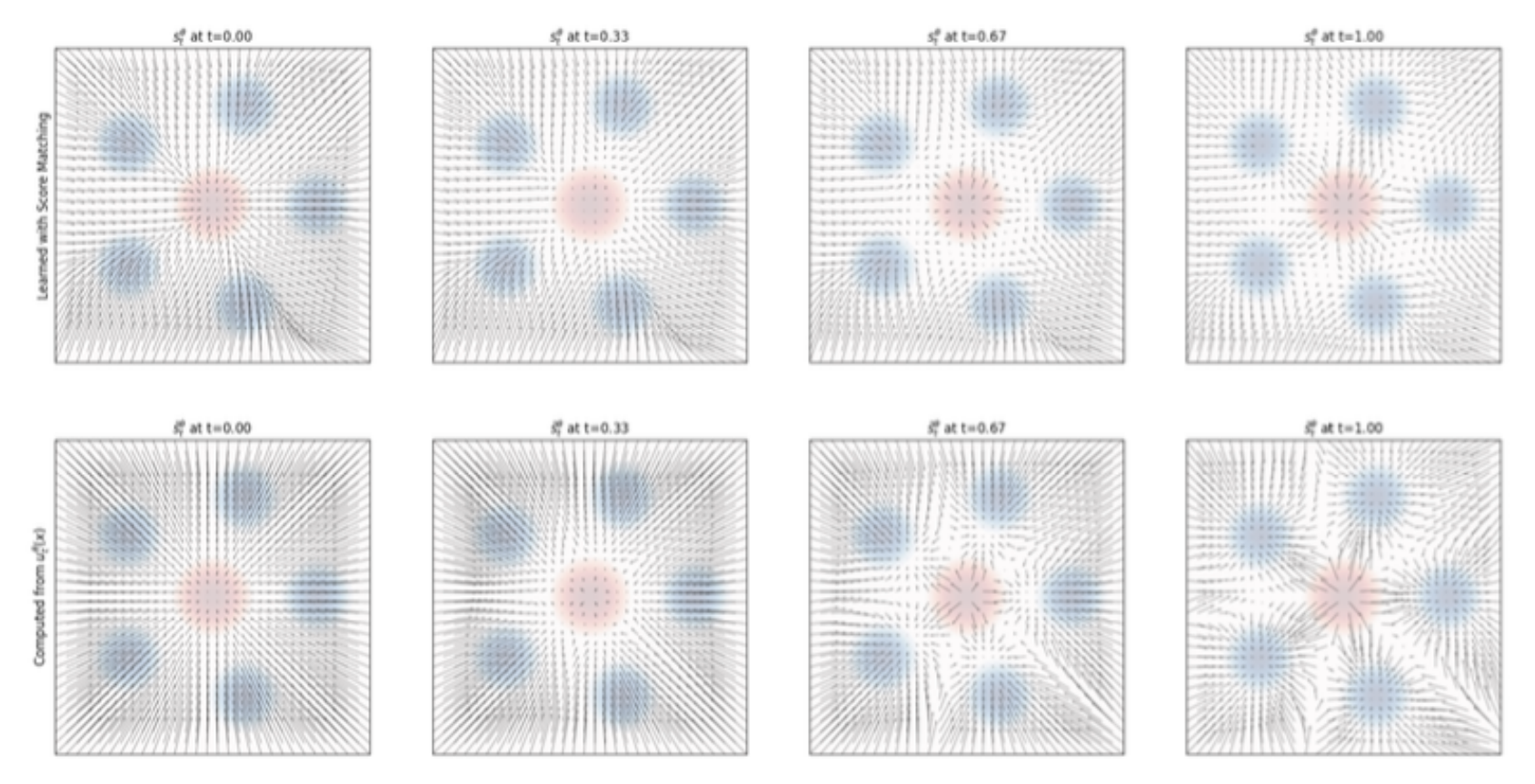

Figure 1: Score learned by score matching vs. score obtained from a trained vector field via the conversion formula.

Sampling with an arbitrary diffusion coefficient

If we have trained a score network s_t^θ, we can sample from

$$X_0 \sim p_{\text{init}}, \quad dX_t = \left[\left(\frac{\beta_t^2 \dot{\alpha}_t}{\alpha_t} - \beta_t \dot{\beta}_t + \frac{\sigma_t^2}{2}\right) s_t^\theta(X_t) + \frac{\dot{\alpha}_t}{\alpha_t} X_t\right] dt + \sigma_t \, dW_t.$$

At perfect training, different σ_t share the same endpoint marginal; in practice, σ_t affects sample quality.

Guide to the diffusion literature

A guide to the diffusion model literature

Discrete time vs. continuous time.

- Early diffusion models: discrete-time Markov chains.
- Later work: continuous-time SDEs $\Rightarrow$ cleaner identities.

"Forward process" vs. probability paths.

- Classical view: noising process from data to Gaussian.
- Training often only needs closed-form $x_t \mid x_0 = z$ (Gaussian paths).

Time reversal vs. PDE derivations.

- Targets can be derived via time reversal or via continuity/Fokker–Planck.
- Sampling may use alternative dynamics (e.g. probability flow ODE).

Summary

Summary: training flow and diffusion models

Flow matching (ODE).

$$\mathcal{L}_{\mathrm{CFM}}(\theta) = \mathbb{E}_{z,t,x\sim p_t(\cdot|z)}\left[\|u_t^\theta(x) - u_t^{\mathrm{target}}(x \mid z)\|^2\right],$$

and generate by simulating $\mathrm{d}X_t = u_t^\theta(X_t)\mathrm{d}t$.

Score matching (SDE).

$$\mathcal{L}_{\mathrm{CSM}}(\theta) = \mathbb{E}_{z,t,x\sim p_t(\cdot|z)}\left[\|s_t^\theta(x) - \nabla \log p_t(x \mid z)\|^2\right],$$

and sample via

$$\mathrm{d}X_t = \left(u_t^\theta(X_t) + \tfrac{\sigma_t^2}{2} s_t^\theta(X_t)\right)\mathrm{d}t + \sigma_t\,\mathrm{d}W_t.$$

Gaussian paths: u_t^θ and s_t^θ convert into each other (Prop. 1).

References

Albergo, M., Boffi, N. M., and Vanden-Eijnden, E. (2025). Stochastic interpolants: A unifying framework for flows and diffusions. *Journal of Machine Learning Research*, 26(209):1–80.

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851.

Lipman, Y., Chen, R. T., Ben-Hamu, H., Nickel, M., and Le, M. (2022). Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*.

References ii

Lipman, Y., Havasi, M., Holderrieth, P., Shaul, N., Le, M., Karrer, B., Chen, R. T., Lopez-Paz, D., Ben-Hamu, H., and Gat, I. (2024). Flow matching guide and code. *arXiv preprint arXiv:2412.06264*.

Liu, X., Gong, C., and Liu, Q. (2022). Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*.