Score Matching and Flow Matching

Handout Notes - Generative and Deep Learning (GDL)

Davide Bacciu - University of Pisa

Notation. Observed random vectors are denoted by $\mathbf{X}$ and values by $\mathbf{x} \in \mathbb{R}^d$. In the diffusion/score part, time is denoted by $t \in [0,1]$ or by discrete steps $t \in \{1,\dots,T\}$, with $\mathbf{z}_t$ the noisy variable at time t. In the flow-matching part, time is denoted by $\tau \in [0,1]$, where $\tau = 0$ corresponds to the source distribution and $\tau = 1$ to the target data distribution. A score field is written

$$s_t(\mathbf{z}) = \nabla_{\mathbf{z}} \log p_t(\mathbf{z}).$$

A velocity field is written $v_\theta(\mathbf{z}, \tau)$, and a ground-truth conditional velocity by $u_\tau(\mathbf{z}_\tau \mid \cdot)$. A dataset is denoted by $\mathcal{D}$ with size $N = |\mathcal{D}|$. Gaussian noise variables are denoted by $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, I)$.

1 A unifying perspective

Modern generative modeling can be described through different mathematical languages: denoising, scores, stochastic differential equations, ordinary differential equations, and transport fields. At first sight these may look like separate paradigms. In fact, they are tightly connected.

A useful big-picture view is:

$$\text{diffusion} \implies \text{scores} \implies \text{probability flow ODE} \implies \text{flow matching}.$$

The common theme is that generation is viewed not as a single jump from latent space to data space, but as a *path* through state space. What changes from one formulation to another is how that path is parameterized and what vector field is learned along it.

Diffusion models describe a stochastic noising process and learn how to reverse it. Score-based models reinterpret denoising as learning the gradient of log-density. Probability flow ODEs show that the same family of intermediate distributions can often be reproduced by a deterministic flow. Flow matching goes one step further and learns the transport velocity directly, without deriving it indirectly from a diffusion objective.

2 From diffusion to score fields

Recall the kernelized forward perturbation used in diffusion models:

$$\mathbf{z}_t = \sqrt{\bar{\alpha}_t}\, \mathbf{z}_0 + \sqrt{1 - \bar{\alpha}_t}\, \boldsymbol{\epsilon}_t, \qquad \boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, I),$$

where $\mathbf{z}_0 = \mathbf{x}$ and $\bar{\alpha}_t$ is the cumulative signal-retention coefficient. Equivalently,

$$q(\mathbf{z}_t \mid \mathbf{z}_0) = \mathcal{N}\big(\sqrt{\bar{\alpha}_t}\, \mathbf{z}_0,\ (1 - \bar{\alpha}_t) I\big).$$

A standard diffusion network is trained to predict the noise that was added:

$$\mathbb{E}\big[\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\mathbf{z}_t, t)\|_2^2\big].$$

This is usually presented as "predict the Gaussian corruption". A more revealing interpretation is that the network is learning a *time-dependent vector field* over the data space.

The bridge is provided by Tweedie's formula.

3 Tweedie's formula and denoising

Suppose that, conditionally on a latent clean point, a noisy variable is Gaussian:

$$\mathbf{Z}_t \mid \mathbf{Z}_0 = \mathbf{z}_0 \sim \mathcal{N}\big(\sqrt{\bar{\alpha}_t}\,\mathbf{z}_0,\ (1-\bar{\alpha}_t)I\big).$$

Tweedie's formula gives the posterior mean of the clean signal from the noisy observation:

$$\mathbb{E}\big[\sqrt{\bar{\alpha}_t}\mathbf{Z}_0 \mid \mathbf{Z}_t = \mathbf{z}_t\big] = \mathbf{z}_t + (1-\bar{\alpha}_t)\nabla_{\mathbf{z}_t}\log p_t(\mathbf{z}_t),$$

where $p_t(\mathbf{z})$ is the marginal density of the noisy variable at time t.

Solving for the posterior mean estimate of the clean sample gives

$$\hat{\mathbf{z}}_0 = \frac{\mathbf{z}_t + (1-\bar{\alpha}_t)\nabla_{\mathbf{z}_t}\log p_t(\mathbf{z}_t)}{\sqrt{\bar{\alpha}_t}}.$$

This already shows something profound: denoising is controlled by the score

$$\nabla_{\mathbf{z}_t}\log p_t(\mathbf{z}_t).$$

Worked example — From diffusion denoising to the score formula

From the diffusion kernel we also have

$$\mathbf{z}_t = \sqrt{\bar{\alpha}_t}\mathbf{z}_0 + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}_t,$$

so rearranging gives

$$\mathbf{z}_0 = \frac{\mathbf{z}_t - \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}_t}{\sqrt{\bar{\alpha}_t}}.$$

On the other hand, Tweedie's formula yields

$$\mathbf{z}_0 = \frac{\mathbf{z}_t + (1-\bar{\alpha}_t)\nabla_{\mathbf{z}_t}\log p_t(\mathbf{z}_t)}{\sqrt{\bar{\alpha}_t}}.$$

Equating the two expressions and multiplying by $\sqrt{\bar{\alpha}_t}$ gives

$$\mathbf{z}_t - \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}_t = \mathbf{z}_t + (1-\bar{\alpha}_t)\nabla_{\mathbf{z}_t}\log p_t(\mathbf{z}_t).$$

Therefore

$$\nabla_{\mathbf{z}_t}\log p_t(\mathbf{z}_t) = -\frac{\boldsymbol{\epsilon}_t}{\sqrt{1-\bar{\alpha}_t}}.$$

So the added noise is proportional to the negative score. Predicting the noise is therefore equivalent to learning the score field up to a time-dependent scaling.

This is the first key unification result:

> *diffusion models do not merely subtract noise; they learn a family of score fields over progressively noisier distributions.*

4 Why score functions are attractive

For a density of the exponential form

$$p_\theta(\mathbf{z}) = \frac{1}{Z_\theta}\exp\big(-f_\theta(\mathbf{z})\big),$$

where f_θ is an energy function and Z_θ is the partition function, the score is

$$\nabla_{\mathbf{z}} \log p_\theta(\mathbf{z}) = -\nabla_{\mathbf{z}} f_\theta(\mathbf{z}).$$

The partition function disappears because it is constant with respect to $\mathbf{z}$. This is extremely useful: the score can be tractable even when the normalized density itself is not.

This is one reason score-based learning is powerful. It turns density learning into vector-field learning.

Geometrically, the score points in the direction of steepest increase of log-density. So if we move in the direction of the score, we move toward higher-density regions. Generation can therefore be seen as following a learned vector field toward the data manifold.

5 Langevin dynamics as score-based sampling

If a score model $s_\theta(\mathbf{z}) \approx \nabla_{\mathbf{z}} \log p(\mathbf{z})$ is available, one can sample approximately from p using Langevin dynamics:

$$\mathbf{z}_{k+1} = \mathbf{z}_k + c_1\, s_\theta(\mathbf{z}_k) + c_2\, \boldsymbol{\epsilon}_k, \qquad \boldsymbol{\epsilon}_k \sim \mathcal{N}(\mathbf{0}, I).$$

The drift term moves samples toward higher-density regions, while the Gaussian perturbation prevents collapse to a single mode and preserves diversity.

This already connects score learning to generative modeling without explicit likelihood training.

6 Score matching

The ideal score-matching objective would be

$$\mathcal{L}_{\mathrm{SM}}(\theta) = \mathbb{E}_{\mathbf{z}\sim p}\big[\|s_\theta(\mathbf{z}) - \nabla_{\mathbf{z}} \log p(\mathbf{z})\|_2^2\big].$$

The obstacle is obvious: the ground-truth score $\nabla_{\mathbf{z}} \log p(\mathbf{z})$ is precisely what we do not know.

The classical workaround is to define a computable surrogate. The most practically relevant form for modern generative models is *denoising score matching* (DSM).

7 Denoising score matching

Let $\tilde{\mathbf{z}}$ be a corrupted version of $\mathbf{z}$:

$$\tilde{\mathbf{z}} = \mathbf{z} + \sigma\boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, I).$$

Then

$$p(\tilde{\mathbf{z}} \mid \mathbf{z}) = \mathcal{N}(\mathbf{z}, \sigma^2 I).$$

The DSM objective regresses the score network onto the score of the corruption kernel:

$$\mathcal{L}_{\mathrm{DSM}}(\theta) = \mathbb{E}_{\mathbf{z},\tilde{\mathbf{z}}}\left[\|s_\theta(\tilde{\mathbf{z}}) - \nabla_{\tilde{\mathbf{z}}} \log p(\tilde{\mathbf{z}} \mid \mathbf{z})\|_2^2\right].$$

For Gaussian corruption,

$$\nabla_{\tilde{\mathbf{z}}} \log p(\tilde{\mathbf{z}} \mid \mathbf{z}) = -\frac{\tilde{\mathbf{z}} - \mathbf{z}}{\sigma^2} = -\frac{\boldsymbol{\epsilon}}{\sigma}.$$

So DSM becomes

$$\mathcal{L}_{\mathrm{DSM}}(\theta) = \mathbb{E}\left[\left\|s_\theta(\tilde{\mathbf{z}}) + \frac{\boldsymbol{\epsilon}}{\sigma}\right\|_2^2\right].$$

Worked example — Why DSM looks like diffusion noise prediction

With
$$\tilde{\mathbf{z}} = \mathbf{z} + \sigma\boldsymbol{\epsilon},$$
the Gaussian corruption density is
$$p(\tilde{\mathbf{z}} \mid \mathbf{z}) = (2\pi\sigma^2)^{-d/2} \exp\left(-\frac{1}{2\sigma^2}\|\tilde{\mathbf{z}} - \mathbf{z}\|_2^2\right).$$
Taking the gradient with respect to $\tilde{\mathbf{z}}$,
$$\nabla_{\tilde{\mathbf{z}}} \log p(\tilde{\mathbf{z}} \mid \mathbf{z}) = -\frac{1}{\sigma^2}(\tilde{\mathbf{z}} - \mathbf{z}) = -\frac{1}{\sigma^2}(\sigma\boldsymbol{\epsilon}) = -\frac{\boldsymbol{\epsilon}}{\sigma}.$$
Therefore
$$\mathcal{L}_{\mathrm{DSM}}(\theta) = \mathbb{E}\left[\left\|s_\theta(\tilde{\mathbf{z}}) + \frac{\boldsymbol{\epsilon}}{\sigma}\right\|_2^2\right].$$
This has exactly the same flavor as diffusion training: construct a corrupted input, then regress a vector target derived from the perturbing noise.

This is the second key unification result:

the usual diffusion loss is a reparameterized denoising score-matching loss.

8 Continuous-time diffusion as an SDE

A discrete collection of noise scales can be refined into a continuous-time stochastic process. In continuous time, diffusion is described by a stochastic differential equation (SDE):
$$d\mathbf{z} = f(\mathbf{z}, t)\,dt + g(t)\,d\mathbf{w},$$
where:

- $f(\mathbf{z}, t)$ is the drift,
- $g(t)$ is the diffusion coefficient,
- $d\mathbf{w}$ is Brownian motion noise.

The forward noising process is now a continuous stochastic perturbation from data at $t = 0$ to a simple base distribution at $t = 1$. At each time t, the process induces a marginal density $p_t(\mathbf{z})$.

This continuous-time view is powerful because it unifies discrete diffusion schedules, multiscale score matching, and stochastic process theory.

9 The reverse-time SDE

Once the forward SDE is defined, one can derive the reverse-time denoising SDE:
$$d\mathbf{z} = \left[f(\mathbf{z}, t) - g(t)^2 \nabla_{\mathbf{z}} \log p_t(\mathbf{z})\right] dt + g(t)\,d\mathbf{w}.$$
The learned score function appears explicitly. This equation states that reverse diffusion can be obtained by correcting the forward drift with a score-dependent term.

Conceptually, the forward process spreads probability mass outward, while the reverse process pulls it back inward according to the local score geometry of the noisy marginals.

10 Probability flow ODE

The reverse SDE is still stochastic because of the Brownian term. A remarkable fact is that there exists a deterministic ODE that generates the same family of marginals $p_t(\mathbf{z})$:

$$\frac{d\mathbf{z}}{dt} = f(\mathbf{z}, t) - \frac{1}{2}g(t)^2 \nabla_{\mathbf{z}} \log p_t(\mathbf{z}).$$

This is called the *probability flow ODE*.

It does not reproduce the same stochastic trajectories as the reverse SDE. What it reproduces is the same *sequence of distributions*. So the SDE and the ODE are two different dynamical descriptions of the same marginal transport.

This is the bridge from score-based diffusion to continuous normalizing flows.

Worked example — Why the probability flow ODE matters conceptually

The reverse SDE defines a stochastic denoising process from a simple base distribution to the data distribution. The probability flow ODE replaces this noisy evolution by a deterministic vector field while preserving the same marginals. Therefore:

- sampling can be viewed either as stochastic denoising or deterministic transport;
- latent inversion becomes possible by integrating the ODE in the opposite direction;
- a diffusion model implicitly defines a bijective map between base samples and data samples, at the level of the ODE flow.

This is the crucial link between score-based generation and flow-based transport.

11 From probability flow to flow matching

The probability flow ODE has the form

$$\frac{d\mathbf{z}_t}{dt} = f(\mathbf{z}_t, t) - \frac{1}{2}g(t)^2 \nabla_{\mathbf{z}} \log p_t(\mathbf{z}_t).$$

This defines a *velocity field*. Instead of deriving this field through a noising process and a score model, one may ask a more direct question:

> *Why not learn the transport velocity itself?*

This idea leads to flow matching. We introduce a time-dependent path of distributions $(p_\tau)_{\tau\in[0,1]}$ connecting:

$$p_0 \quad \text{(source / prior)} \qquad \text{to} \qquad p_1 \quad \text{(target / data)}.$$

Then we learn a neural velocity field

$$v_\theta(\mathbf{z}_\tau, \tau)$$

that transports samples along this path according to the ODE

$$\frac{d\mathbf{z}_\tau}{d\tau} = v_\theta(\mathbf{z}_\tau, \tau).$$

This is a continuous normalizing-flow viewpoint, but trained through direct velocity regression rather than likelihood maximization.

12 Flow matching setup

A flow-matching method requires two ingredients:

- a probability path $(p_\tau)_{\tau\in[0,1]}$ connecting source and target;
- a regression target for the velocity field that generates that path.

The important difference from diffusion is conceptual:

- in diffusion, the path is largely dictated by a carefully designed noising process;
- in flow matching, the path can be chosen more freely.

So flow matching keeps the dynamic-transport viewpoint of continuous diffusion, but removes the need to derive training from likelihood or reverse-time SDE theory.

13 Rectified flow: the simplest path

The simplest probability path is linear interpolation between a source sample $\mathbf{z}_0 \sim p_0$ and a data sample $\mathbf{x} \sim p_1$:

$$\mathbf{z}_\tau = (1-\tau)\mathbf{z}_0 + \tau\mathbf{x}.$$

Differentiating with respect to τ gives the conditional velocity

$$u_\tau(\mathbf{z}_\tau \mid \mathbf{x}, \mathbf{z}_0) = \frac{d\mathbf{z}_\tau}{d\tau} = \mathbf{x} - \mathbf{z}_0.$$

So each sample pair defines a straight path with constant velocity.

This construction is called rectified flow. It is attractive because the regression target is extremely simple.

Worked example — Training target for rectified flow

Given a source sample $\mathbf{z}_0$ and a data sample $\mathbf{x}$, define

$$\mathbf{z}_\tau = (1-\tau)\mathbf{z}_0 + \tau\mathbf{x}.$$

Differentiate:

$$\frac{d\mathbf{z}_\tau}{d\tau} = -\mathbf{z}_0 + \mathbf{x} = \mathbf{x} - \mathbf{z}_0.$$

Therefore the neural velocity model is trained on pairs

$$(\mathbf{z}_\tau, \tau) \mapsto \mathbf{x} - \mathbf{z}_0.$$

No score, reverse kernel, or ELBO derivation is needed.

14 Gaussian probability paths

A broader and very useful family of paths is

$$\mathbf{z}_\tau = \alpha_\tau\mathbf{x} + \sigma_\tau\boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, I),$$

with boundary conditions

$$\alpha_0 = 0, \quad \sigma_0 = 1, \qquad \alpha_1 = 1, \quad \sigma_1 = 0.$$

At $\tau = 0$ this gives pure Gaussian noise, and at $\tau = 1$ it gives the data point.

Differentiating yields

$$u_\tau(\mathbf{z}_\tau \mid \mathbf{x}, \boldsymbol{\epsilon}) = \frac{d\mathbf{z}_\tau}{d\tau} = \dot{\alpha}_\tau\mathbf{x} + \dot{\sigma}_\tau\boldsymbol{\epsilon}.$$

So the target velocity is again directly computable.

This family is important because diffusion-style noising paths are special cases of Gaussian probability paths. Hence Gaussian flow matching and diffusion can represent the same underlying generative model under different parameterizations.

15 Conditional flow matching

Ideally, one would regress the marginal velocity field $u_\tau(\mathbf{z})$ that transports the full path of marginals. But this quantity is not directly observable from data. The key trick is to regress an *easy conditional velocity* instead.

If $q_\tau(\cdot \mid \mathbf{x})$ is the chosen conditional path distribution, then conditional flow matching uses

$$\mathcal{L}_{\mathrm{CFM}}(\theta) = \mathbb{E}_{\tau,\mathbf{x},\mathbf{z}_\tau}\left[\|v_\theta(\mathbf{z}_\tau, \tau) - u_\tau(\mathbf{z}_\tau \mid \mathbf{x}, \cdot)\|_2^2\right].$$

In practice:

1. sample a data point $\mathbf{x}$;
2. sample a source variable, such as $\mathbf{z}_0$ or $\boldsymbol{\epsilon}$;
3. choose a random time τ;
4. construct the interpolated or perturbed point $\mathbf{z}_\tau$;
5. regress the network output onto the known conditional velocity target.

16 Why conditional flow matching works

The crucial theoretical fact is that the marginal velocity field is the conditional expectation of the conditional targets:

$$u_\tau(\mathbf{z}) = \mathbb{E}[u_\tau(\mathbf{z}_\tau \mid \mathbf{x}, \cdot) \mid \mathbf{z}_\tau = \mathbf{z}].$$

So although the marginal velocity is not directly observed, the conditional targets average to the correct transport field. This is why regressing the easy conditional target gives the correct gradient signal for the desired marginal flow.

This role is analogous to what happens in diffusion and denoising score matching: we train on tractable conditional quantities that average to the correct marginal object.

17 Training flow matching versus diffusion

The training recipes of diffusion and flow matching look surprisingly similar in code, even though their conceptual derivations differ.

Diffusion-style recipe

1. sample $\mathbf{x} \sim p_1$;
2. sample time t;
3. sample Gaussian noise $\boldsymbol{\epsilon}$;
4. construct a noisy point
$$\mathbf{z}_t = \alpha_t \mathbf{x} + \sqrt{1 - \alpha_t}\,\boldsymbol{\epsilon};$$
5. predict noise or score.

Flow-matching recipe

1. sample $\mathbf{x} \sim p_1$;
2. sample source point $\mathbf{z}_0 \sim p_0$ or noise $\boldsymbol{\epsilon}$;
3. sample time τ;
4. construct an interpolated point

$$\mathbf{z}_\tau = (1-\tau)\mathbf{z}_0 + \tau\mathbf{x} \quad \text{or} \quad \mathbf{z}_\tau = \alpha_\tau \mathbf{x} + \sigma_\tau \boldsymbol{\epsilon};$$

5. predict velocity.

So architecturally these families are often very close. What changes is the head and the target:

- predict noise $\boldsymbol{\epsilon}$ in diffusion,
- predict score s in score matching,
- predict velocity v in flow matching.

18 Sampling in flow matching

After training, generation is performed by solving the ODE

$$\frac{d\mathbf{z}_\tau}{d\tau} = v_\theta(\mathbf{z}_\tau, \tau), \qquad \mathbf{z}_0 \sim p_0.$$

This is an ordinary Cauchy problem and can be integrated numerically by any ODE solver: Euler, midpoint, Runge–Kutta, and so on.

The simplest first-order Euler step is

$$\mathbf{z}_{\tau+\Delta\tau} = \mathbf{z}_\tau + \Delta\tau\, v_\theta(\mathbf{z}_\tau, \tau).$$

A key practical advantage is that sampling is deterministic once the initial source sample is chosen. Moreover, if the learned velocity field is smooth, sampling may require significantly fewer integration steps than classical diffusion requires denoising steps.

Worked example — Euler sampling for flow matching

Suppose the source distribution is

$$\mathbf{z}_0 \sim \mathcal{N}(\mathbf{0}, I),$$

and the learned velocity field is $v_\theta(\mathbf{z}, \tau)$. Choose a step size $\Delta\tau = 1/K$ and define $\tau_k = k/K$. Starting from $\mathbf{z}_{\tau_0} = \mathbf{z}_0$, iterate

$$\mathbf{z}_{\tau_{k+1}} = \mathbf{z}_{\tau_k} + \Delta\tau\, v_\theta(\mathbf{z}_{\tau_k}, \tau_k), \qquad k = 0, \ldots, K-1.$$

Then $\mathbf{z}_{\tau_K}$ is the generated sample in data space. The whole trajectory is deterministic conditioned on the initial latent draw.

19 Practical architectures

In modern practice, the architectural gap between diffusion and flow matching is much smaller than the conceptual gap in their derivations.

Common backbones include:

- U-Nets for image-space generation,
- Transformer variants in latent or patch space,

- sinusoidal or learned time embeddings,
- cross-attention for conditional generation.

So the major distinction is not usually the network body. It is the target vector field that the network is trained to predict.