Score Matching

1. High-level summary (one-paragraph)

Score matching is a technique for estimating the *score function* $\nabla_x \log p(x)$ of a data distribution without requiring its normalizing constant. In score-based generative modeling (diffusion models), we train a time-conditional neural network $s_\theta(x,t)$ to approximate $\nabla_x \log p_t(x)$ or distributions p_t obtained by progressively adding Gaussian noise to data. After learning these conditional scores, generation is performed by integrating a reverse-time stochastic differential equation (SDE) or an equivalent deterministic ODE (probability-flow ODE) that uses the learned scores to denoise samples from Gaussian noise back to the data manifold. Practical implementations commonly use denoising score matching (DSM) and an equivalent ε-prediction parameterization (as in DDPM), and rely on sampler design (predictor–corrector, DDIM, distillation) to trade off quality and sampling speed.

---

2. Notation and problem setup

- $x \in \mathbb{R}^d$ denotes data. True data density is $p_{\text{data}}(x)$ (unknown up to normalization).
- Score function: $s_p(x) = \nabla_x \log p(x) \in \mathbb{R}^d$.
- Parametric score model: $s_\theta(x)$ or time-conditional $s_\theta(x,t)$.
- Forward noising (continuous viewpoint): an SDE
$$dx = f(x,t)dt + g(t)dW_t, \qquad t \in [0,T],$$
with initial marginal $p_0 = p_{\text{data}}$. For many constructions p_T is approximately Gaussian.
- Discrete DDPM viewpoint: finite-step Markov chain with Gaussian transitions $q(x_t|x_{t-1}) = \mathcal{N}\left(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I\right)$.

---

3. Hyvärinen score matching — rigorous derivation

We want to estimate $s_p(x)$ by minimizing
$$J(\theta) = \frac{1}{2}\mathbb{E}_p\left[\left\|s_\theta(x) - s_p(x)\right\|^2\right] = \frac{1}{2}\mathbb{E}_p[\|s_\theta(x)\|^2] - \mathbb{E}_p\left[s_\theta(x)^\top s_p(x)\right] + \text{const.}$$
The problematic term $\mathbb{E}_p\left[{s_\theta}^\top s_p\right]$ contains $s_p(x) = \nabla_x \log p$. Use integration by parts to remove s_p. For clarity, assume p has support $\mathbb{R}^d$ and boundary terms vanish (sufficient decay at infinity). For scalar functions $\mu(x)$ and vector field $v(x)$:

$$\int u(x)\mathrm{div}v(x)dx = -\int \nabla u(x)^\top v(x)dx \qquad \text{(if boundary term zero).}$$

Take $u(x) = p(x)$ and $v(x) = s_\theta(x)$. Then

$$\int p(x)\mathrm{div}s_\theta(x)dx = -\int \nabla p(x)^\top s_\theta(x)dx.$$

But $\nabla p = p\nabla \log p = ps_p$. Hence

$$\mathbb{E}_p[s_\theta{}^\top s_p] = \int s_\theta{}^\top s_p p dx = -\int p\mathrm{div}s_\theta dx = -\mathbb{E}_p[\mathrm{div}s_p].$$

Substitute back:

$$J(\theta) = \frac{1}{2}\mathbb{E}_p[\|s_\theta\|^2] + \mathbb{E}_p[\mathrm{div}s_\theta] + \mathrm{const.}$$

Thus minimizing $\tilde{J}(\theta) = \frac{1}{2}\mathbb{E}_p\left[\frac{1}{2}\|s_\theta\|^2 + \mathrm{div}s_\theta\right]$ yields the same estimator.. **Key:** $\tilde{J}$ requires only evaluating s_θ and its divergence (trace of Jacobian), not the unknown s_p itself.

Practical caveat: computing $\mathrm{div}s_\theta = \mathrm{tr}(\nabla_x s_\theta)$ is expensive in high dimensions (requires full Jacobian trace). That motivates denoising formulations that avoid explicit divergence computation.

---

4. Denoising Score Matching (DSM) — motivation & derivation

Consider corrupting data $x_0 \sim p_{\mathrm{data}}$ with Gaussian noise: $x = x_0 + \sigma\epsilon, \epsilon \sim \mathcal{N}(0, I)$.

The conditional density $q(x|x_0) = \mathcal{N}(x; x_0, \sigma^2 I)$ has known score

$$\nabla_x \log q(x|x_0) = -\frac{x - x_0}{\sigma^2} = -\frac{\epsilon}{\sigma}.$$

Vincent (2011) showed that minimizing the expected squared error between model $s_\theta(x)$ and conditional score $\nabla_x \log q(x|x_0)$

$$\mathbb{E}_{x_0,\epsilon}[\|s_\theta(x_0 + \sigma\epsilon) - \nabla_x \log q(x_0 + \sigma\epsilon|x_0)\|^2]$$

is (under some conditions and small σ limits) equivalent to Hyvärinen's objective for estimating the true score of p. Intuitively, DSM replaces the unknown score of the corrupted marginal $p_\sigma(x) = \int q(x|x_0)\, p_{\mathrm{data}}(x_0)dx_0$ with the *known* conditional score $\nabla_x \log q(x|x_0)$ as a training target; averaging over x_0 gives a valid objective to learn $\nabla_x \log p_\sigma(x)$.

In diffusion models we extend DSM to many noise levels $\{\sigma_t\}$ or continuous t: we train $s_\theta(x, t)$ to predict conditional score at each t:

$$L(\theta) = \mathbb{E}_{t\sim p(t), x_0, \epsilon}\left[\lambda(t)\left\|s_\theta(x_t, t) + \frac{\epsilon}{\sigma_t}\right\|^2\right], \qquad x_t = x_0 + \sigma_t\epsilon.$$

Here $\lambda(t)$ is a weight function.

---

5. Continuous-time SDE view & reverse-time SDE

A unifying continuous-time formalism (Song & Ermon, 2019; 2020) uses forward SDEs parameterized so that marginal p_t evolves from p_0 to an easy-to-sample p_T (often Gaussian). Example SDEs:

- Variance Exploding (VE): $dx = \sqrt{\frac{d[\sigma_t{}^2]}{dt}} dW_t$ (pure diffusion).
- Variance Preserving (VP): $dx = -\frac{1}{2}\beta(t)xdt + \sqrt{\beta(t)}dW_t$ (analogue of

  discrete DDPM).

Given forward SDE $dx = f(x,t)dt + g(t)dW_t$ with known f, g, marginal p_t satisfy Fokker–Planck equation. Anderson (1982) gives the reverse-time SDE:

$$dx = [f(x,t) - g(t)^2\nabla_x \log p_t(x)]dt + g(t)d\bar{W}_t, \quad t \text{ decreasing,}$$

where $d\bar{W}_t$ is reverse-time Brownian motion. Replacing $\nabla_x \log p_t(x)$ by learned $s_\theta(x,t)$ yields a practical sampler. Numerical integration (Euler–Maruyama) from $t = T$ down to 0 produces samples approximating p_0.

---

6. Probability-flow ODE (deterministic equivalent)

There exists an ODE with the same marginals as the SDE (Song et al.). The probability-flow ODE is

$$\frac{dx}{dt} = f(x,t) - \frac{1}{2}g(t)^2\nabla_x \log p_t(x).$$

This ODE is deterministic and maps $p_T \to p_0$. Its advantages:

- Deterministic trajectories (no stochastic noise), and
- Enables exact log-density tracking via instantaneous change-of-variables:

$$\frac{d}{dt}\log p_t\big(x(t)\big) = -\mathrm{div}\left(f(x,t) - \frac{1}{2}g(t)^2\nabla_x \log p_t(x)\right),$$

  allowing (with numerical integrator/adjoint) approximation of $\log p_0(x)$

for

  likelihood evaluation.

Practically, both reverse SDE samplers and ODE samplers use the same learned scores; ODE samplers produce deterministic samples (used when likelihood evaluation is desired).

---

7. Equivalence to ε-prediction (DDPM parameterization)

DDPM (Ho et al., 2020) frames training as predicting the added Gaussian noise ϵ in the finite-step discrete forward process. For Gaussian corruption $x_t = \alpha_t x_0 + \sigma_t \epsilon$, predicting ϵ with $\epsilon_\theta(x_t, t)$ is algebraically equivalent to predicting score:

$$s_\theta(x_t, t) \approx \nabla_{x_t} \log p_t(x_t) \propto -\frac{1}{\sigma_t}\epsilon_\theta(x_t, t).$$

Thus minimizing $\mathbb{E}\|\epsilon_\theta(x_t, t) - \epsilon\|^2$ corresponds (up to a scaling and weighting) to minimizing the DSM loss. Empirically, ϵ- prediction is more stable, avoids explicit Jacobian/divergence computation, and is standard in engineering implementations.

---

8. Sampling algorithms (practical recipes)

8.1 Basic reverse SDE (Euler–Maruyama)

Initialize $x_T \sim p_T$ (Gaussian). For t decreasing in steps:

$$x_{t-\Delta t} = x_t + [f(x_t, t) - g(t)^2 s_\theta(x_t, t)]\Delta t + g(t)\sqrt{\Delta t} z, \qquad z \sim \mathcal{N}(0, I).$$

Repeat until $t \approx 0$.

8.2 Predictor–Corrector (PC) Sampler

Alternate:

- Predictor: one Euler–Maruyama or ODE step using learned score (predict next state).
- Corrector: apply Langevin dynamics (stochastic gradient MCMC) using the current score as gradient to refine distributional fit:

$$x \leftarrow x + \alpha s_\theta(x_t, t) + \sqrt{2\alpha} z.$$

The corrector reduces discretization bias and improves sample quality at cost of extra score evaluations.

8.3 DDIM (deterministic non-Markov sampler)

DDIM (Song et al., 2020 variant) derives deterministic sampling updates in discrete time that are consistent with marginal transitions; permits fewer sampling steps (sublinear in T) with reasonable quality. DDIM can be interpreted as discretizing the probability-flow ODE.

8.4 Acceleration & Distillation

- **Distillation:** train a smaller/faster model to mimic many-step sampler (e.g., progressive distillation) to reduce steps.
- **Higher-order solvers:** for ODE sampler, Runge–Kutta / adaptive solvers can

reduce error (but require more eval per step).

- **Step schedule:** non-uniform time grids concentrated where score changes rapidly.

---

9. Likelihood, log-probability estimation

The probability-flow ODE yields an invertible mapping between x_T and x_0. Using instantaneous change-of-variables:

$$\log p_0(x_0) = \log p_T(x_T) - \int_0^T \operatorname{div}\left(f(x(t),t) - \frac{1}{2}g(t)^2 s_\theta(x(t),t)\right) dt,$$

where $x(t)$ is ODE trajectory from x_0 to x_T. The divergence term can be estimated with Hutchinson trace estimators (random vector probe) to avoid explicit Jacobians. This allows approximate log-likelihood evaluation for model comparison (albeit at extra computational cost).

---

10. Architectural & training choices

- **Network:** U-Net with residual blocks + attention is common for images. For conditional generation, condition on labels/embeddings.
- **Time embedding:** encode scalar t using sinusoidal embeddings or small MLP (to modulate layers).
- **Loss weighting $\boldsymbol{\lambda(t)}$:** affects where model focuses; common choices include $\lambda(t) = 1$ or $\lambda(t) = 1/{\sigma_t}^2$ depending on parameterization.
- **Normalization & stability:** predict ϵ\epsilonϵ instead of score; use gradient clipping, Adam optimizer, EMA of parameters for sampling.
- **Batching noise levels:** sample t per-example uniformly or according to importance distribution.
- **Data preprocessing:** scale images to $[-1, 1]$ or $[0, 1]$; consider variance-stabilizing transforms for likelihood work.

---

11. Variants & related methods

- **Sliced score matching:** projects high-dimensional score estimation to 1D slices to reduce cost.
- **Stein discrepancy / kernelized score matching:** alternate objectives using Stein identities.
- **Score-Matching + Flow hybrids:** combine normalizing flows with score-based training for improved likelihoods or invertibility.

- **Classifier guidance / classifier-free guidance:** at sampling time, bias the score with gradients from an auxiliary classifier (or conditional model) to improve conditional sample fidelity.

---

12. Common pitfalls & practical tips

- **Divergence computation is expensive** — prefer DSM or ε-prediction.
- **Edge behavior & boundary terms:** Hyvärinen derivation assumes vanishing boundary terms; ensure data or parameterization respects this (or use DSM).
- **Small $\boldsymbol{\sigma}$ instability:** as $\sigma_t \to 0$, targets become large; use careful weighting or separate handling of low-noise regime.
- **Sampling steps vs. quality tradeoff:** fewer steps reduce cost but may introduce bias; use PC or DDIM and/or distill for speedups.
- **Memory / compute:** U-Net + long training + many sampling steps => heavy GPU cost. EMA weights and mixed precision help.

---

13. Intuition recap (non-technical)

- The score $\nabla_x \log p(x)$ tells us which way to move in data space to increase probability.
- We learn this vector field at many noise scales so we know how to denoise from coarse to fine.
- Generation is simply following these vector fields backward: start with noise and iteratively step in directions that increase data-likelihood.