# What we are learning & Why

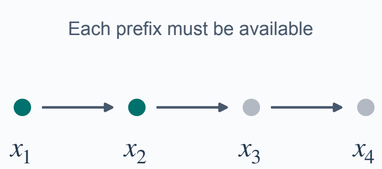


Generating samples requires a sampler—and a tractable objective that makes its distribution match the data.

Evaluate: $x \mapsto p_\theta(x)$ Sample: $\hat{X} \sim P_\theta$

A model can provide both capabilities. Density evaluation alone does not specify an efficient sampling procedure.

## Autoregressive

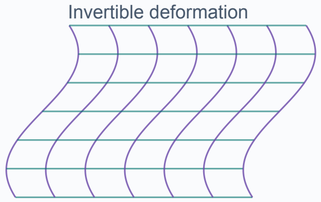



$$p_\theta(x) = \prod_i p_\theta(x_i \mid x_{<i})$$

Observed prefixes supply conditional supervision.

Generated prefixes must exist before subsequent draws.

## Discrete normalizing flow

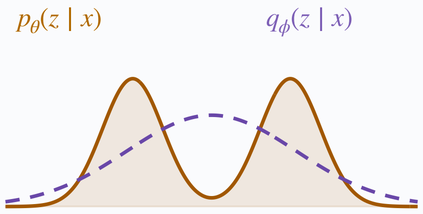



$$p_\theta(x) = p_0(z) / |\det DT(z)|$$
$$x = T(z)$$

Change of variables connects sampling with likelihood.

Cheap inversion and determinants constrain the map architecture.

## Variational autoencoder




$$\log p_\theta(x) = \mathrm{ELBO}(x) + \mathrm{KL}\left(q_\phi(z \mid x) \| p_\theta(z \mid x)\right)$$

Latent inference makes a likelihood bound tractable.

A restricted posterior leaves a model-dependent bound gap.

## Generative adversarial net

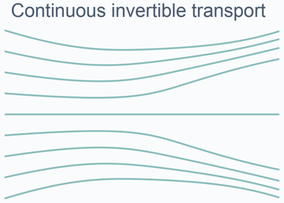



$$\min_G \max_D \mathscr{L}_{\mathrm{adv}}(G, D)$$

A learned discriminator supplies a distribution-level signal.

Coupled optimization can impair stability and mode coverage.

## Continuous normalizing flow




$$Y_1 = \phi_1^\theta(Z), \quad Z \sim P_0$$

Regular dynamics provide an invertible generative map.

Likelihood training requires ODE solves and divergence.

## Our lens: CNF + Flow Matching

Selection criteria: joint-state updates, a flexible velocity network, non-adversarial training, and no variational posterior bound gap.

CNF avoids coordinate-wise generation and tractable-Jacobian layers.
FM replaces likelihood ODE solves and divergence with local regression.

## Why the combination meets these criteria

$$\mathscr{L}_{\mathrm{CFM}} = \mathscr{L}_{\mathrm{FM}} + C, \quad \nabla_\theta C = 0$$

No adversary or variational posterior is required.
Field approximation and sampling integration remain.

$Z \sim P_0$, $X \sim P_{\mathrm{data}}$, $R_t$: training interpolation, $Y_t$: ODE state, $S_t$: SDE state

$x \in \mathbb{R}^d$, $t \in [0, 1]$; regular flows and densities assumed. The CFM identity requires finite second moments and valid differentiation under the expectation.

Sources: Neural ODE (2018); FFJORD (2018); Flow Matching (2022). Illustrations are schematic except CNF trajectories.

# Learning and sampling: microscopic and macroscopic views

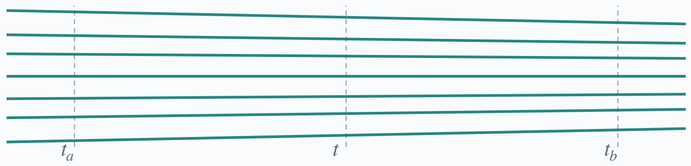


Two views of generation: individual samples and the whole distribution.

| | Lagrangian view: samples and trajectories | Eulerian view: fields and distributions |
|---|---|---|
| **LEARNING**<br>Construct tractable conditional labels. | Each sampled pair provides a state and a conditional label.<br>FM : $r_t = (1-t)z + tx, \quad u = x - z$<br>Diffusion : $r_s = \alpha_s x + \sigma_s z, \quad \nabla_r \log p_s(r \mid X = x)\big\|_{r=r_s} = -z/\sigma_s$ | Regression integrates out the hidden training pair.<br>$v_t^*(r) = \mathbb{E}[U \mid R_t = r]$<br>$\ell_s(r) = \nabla_r \log p_s(r) = -\mathbb{E}[Z \mid R_s = r]/\sigma_s$ |
| **SAMPLING**<br>Execute the learned dynamics. | Local rules define an executable sampler.<br>ODE: $dy_t = v_t(y_t)\,dt$<br>SDE: $dS_t = b_t(S_t)\,dt + g(t)\,dW_t$ | Probability conservation connects motion to density.<br>$\partial_t p_t = -\nabla \cdot (p_t v_t)$<br>$\partial_t p_t = -\nabla \cdot (p_t b_t) + \frac{1}{2} g(t)^2 \Delta p_t$ |

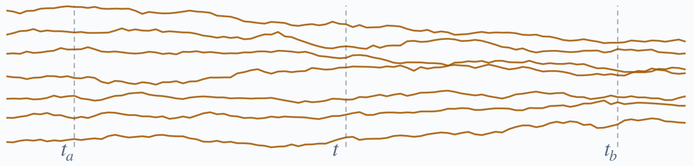
## Different trajectories can share the same time marginals.

### Probability-flow ODE




A local time window: smooth deterministic trajectories.

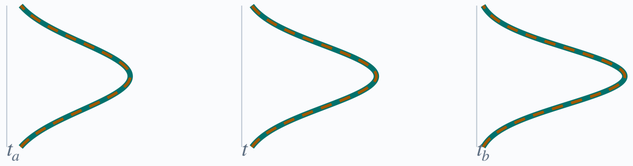
### SDE




Each curve is one realization: $s_t = S_t(\omega)$.

### The same density at each time

Teal: probability-flow ODE / dashed gold: SDE




Each cross-section has the same probability density.

$$v_t(r) = b_t(r) - \frac{1}{2} g(t)^2 \nabla_r \log p_t(r) \quad \Longrightarrow \quad p_t^{\mathrm{ODE}}(r) = p_t^{\mathrm{SDE}}(r)$$

Same ODE + same initial point: same trajectory.
Matched ODE / SDE: same marginals.

Assume x in R^d, matched laws at the window entrance, exact fields, positive smooth densities and well-posed dynamics; g is scalar and state-independent.

FM: t runs from noise to data. Diffusion: s runs from data to noise (sampling reverses s). Z is standard Gaussian and independent of X; σs > 0.

t = 0.00



Training image

Data points sampled from color-weighted pixels.

Flow Matching learns a velocity field from sampled data points.
ODE sampling transports Gaussian noise into new samples.

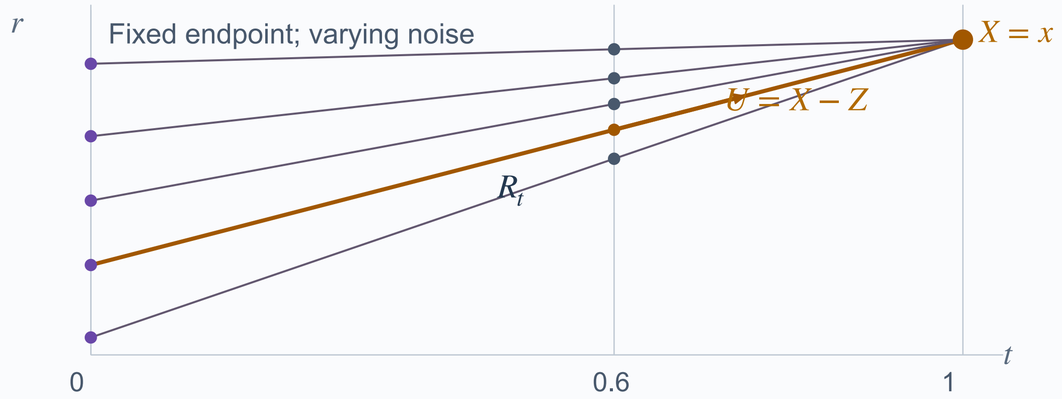
RTX 5090 · 500,000 simulated / 200,000 displayed · RK4 · h = 0.005 · seed 42

# Constructing tractable supervision from data samples



We design a conditional construction with computable states and velocities; its mixture retains the structure of the data.

## A chosen path supplies the missing velocity labels

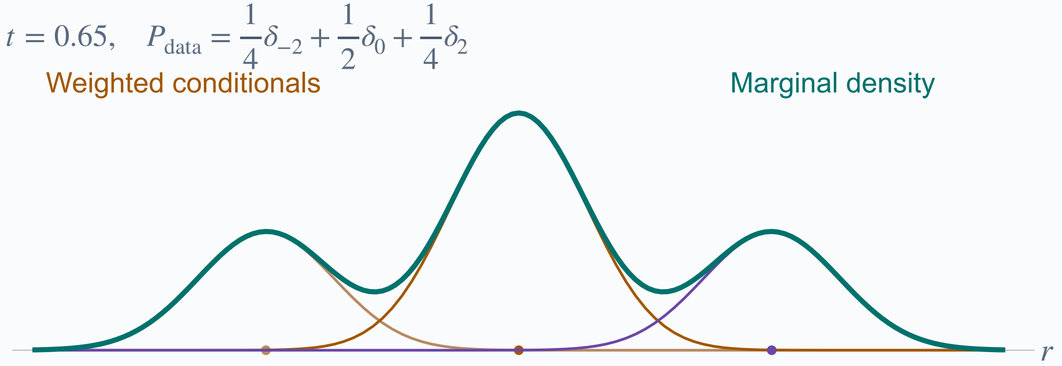



$$R_t = (1-t)Z + tX, \qquad \dot{R}_t = X - Z$$

$$u_t(r \mid x) = \frac{x - r}{1 - t}, \quad u_t(R_t \mid X) = X - Z$$

Labels are designed, not observed trajectories.

## Simple conditionals form a complex marginal




$$p_t(r \mid x) = \mathcal{N}(r; tx, (1-t)^2 I)$$

$$p_t(r) = \int p_t(r \mid x)\, P_{\text{data}}(dx)$$

The network receives state and time; endpoint identities are used only to construct supervision.

$$\mathscr{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t,X,Z}\left\| v_\theta(R_t, t) - (X - Z) \right\|^2$$

No marginal-density evaluation.
No model ODE solve during training.

## Why can one predicted velocity represent the many endpoint pairs compatible with the same state?

Assume X, Z ∈ $R^d$; Z ~ N(0,I), independent of data X; t ~ U(0,1); finite second moments. Gaussian formulas use t < 1.

At t → 1, the conditional law tends to $\delta_x$. The chart is an exact one-dimensional mixture. Source: Flow Matching, §§3–4.

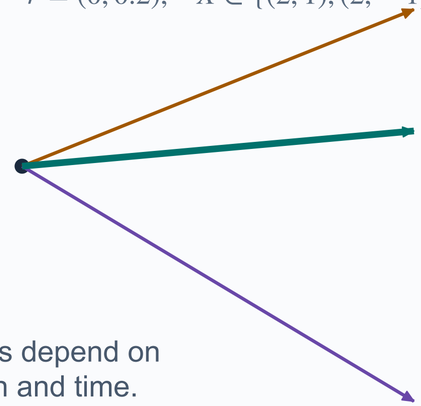
# Why conditional regression preserves marginal probability transport



Train micro, predict macro. Conditional velocity labels identify the marginal field that transports the distribution.

## Learning: recover the local posterior mean

$t = 0.5, \quad r = (0, 0.2), \quad X \in \{(2, 1), (2, -1)\}$



$u_+ = (4, 1.6), \quad w_+ = 0.690$

$v_t^*(r) = (4, 0.360)$

$u_- = (4, -2.4), \quad w_- = 0.310$

Weights depend on position and time.

$$v_t^*(r) = \mathbb{E}[U \mid R_t = r], \qquad U = X - Z$$

$$P(X \in dx \mid R_t = r) = \frac{p_t(r \mid x) P_{\text{data}}(dx)}{p_t(r)}$$

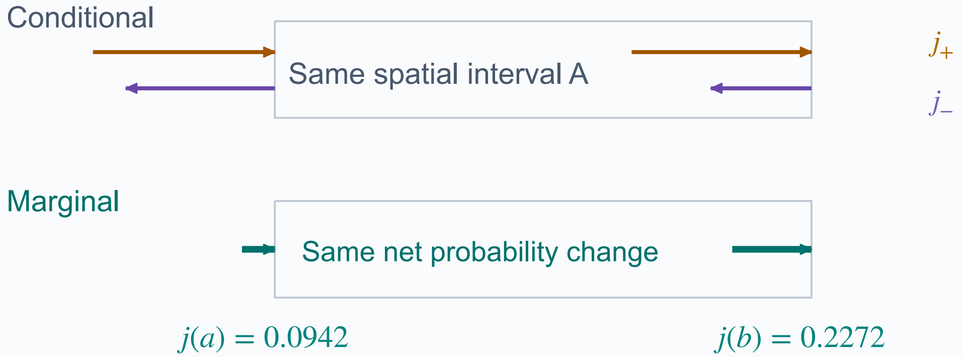
$$\mathscr{L}_{\text{CFM}} = \mathscr{L}_{\text{FM}} + \mathbb{E}\left\| U - v_t^*(R_t) \right\|^2$$

$$\mathscr{L}_{\text{FM}} = \mathbb{E}\left\| v_\theta(R_t, t) - v_t^*(R_t) \right\|^2$$

CFM loss can remain nonzero when the marginal field is exact.

## Transport: preserve the total probability flux

Exact 1D slice: equal endpoint masses at X = ±1; t = 0.5




$$p_t(r) v_t^*(r) = \int p_t(r \mid x) u_t(r \mid x) \, P_{\text{data}}(dx)$$

$$\partial_t p_t = -\nabla \cdot (p_t v_t^*)$$

$$\frac{d}{dt} P(R_t \in A) = j(a) - j(b) = -0.1329$$

Averaging preserves flux, not endpoint pairings.

$$\dot{Y}_t = v_t^*(Y_t), \quad Y_0 \sim P_0 \implies \mathscr{L}(Y_t) = \mathscr{L}(R_t)$$

$$V = v_\theta(r, t), \quad M = \mathbb{E}[U \mid R_t = r]$$

$$\mathbb{E}[\langle V - M, M - U \rangle \mid R_t = r] = \langle V - M, \underbrace{M - \mathbb{E}[U \mid R_t = r]}_{0} \rangle = 0.$$

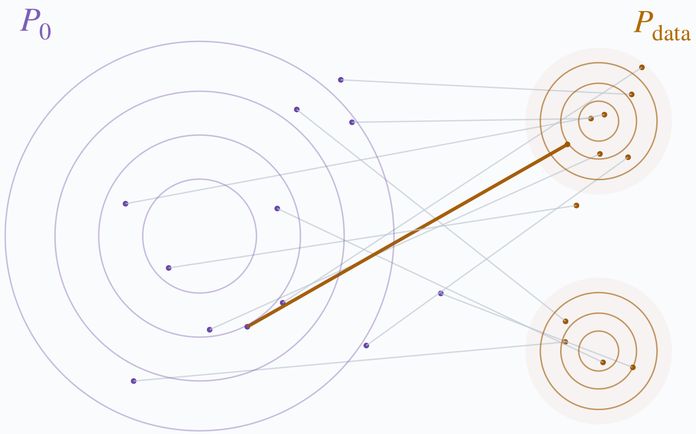
Sampling supplies the weights, regression learns the mean, averaging preserves flux, and the continuity equation determines the distribution's evolution.
Train micro, predict macro.

# Why straight training paths can produce curved sampling trajectories



The regression target is a local conditional average. It preserves probability flow, not the original endpoint pairings.

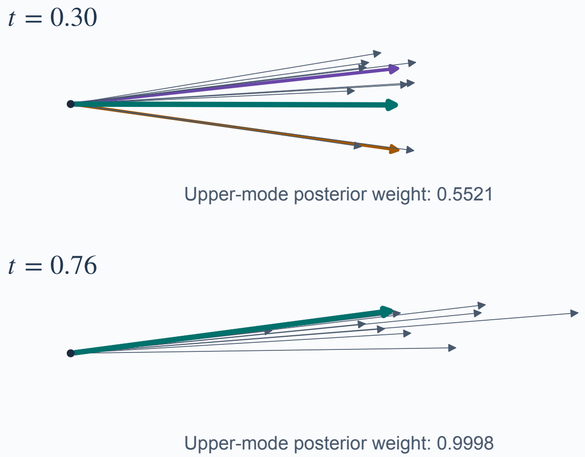
## 1 Specify conditional paths




$$R_t = (1-t)Z + tX, \quad U = X - Z$$

Each paired path has a constant velocity.
The network receives state and time, not the endpoint pair.

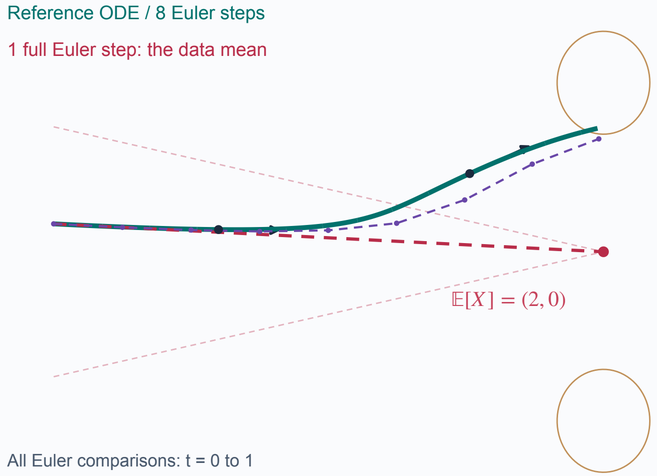
## 2 Learn the marginal velocity




$$v_t^*(x) = \mathbb{E}[U \mid R_t = x]$$

Posterior weights change with position and time.
The local mean can change along the resulting trajectory.

## 3 Integrate the marginal field




$$\dot{Y}_t = v_t^*(Y_t), \qquad Y_0 \sim P_0$$

$$z + v_0^*(z) = \mathbb{E}[X \mid Z = z] = \mathbb{E}[X]$$

Independent-pair endpoint MSE collapses to the data mean.

## One full Euler step is an endpoint regressor

$$z + v_0^*(z) = \mathbb{E}[X \mid Z = z] = \mathbb{E}[X] = (2,0)$$

$$T(z) = \mathbb{E}[X] \implies T_{\#}P_0 = \delta_{(2,0)}$$

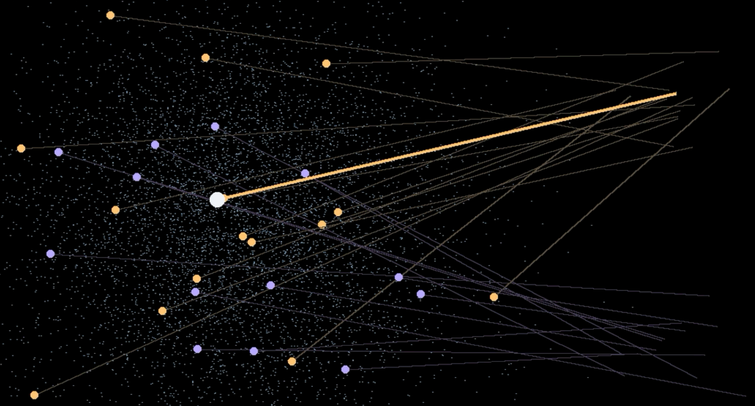
Independent endpoint regression averages both modes.
Every initial state receives the same endpoint.

Local averaging preserves probability flux,
while endpoint pairings are lost (p6).

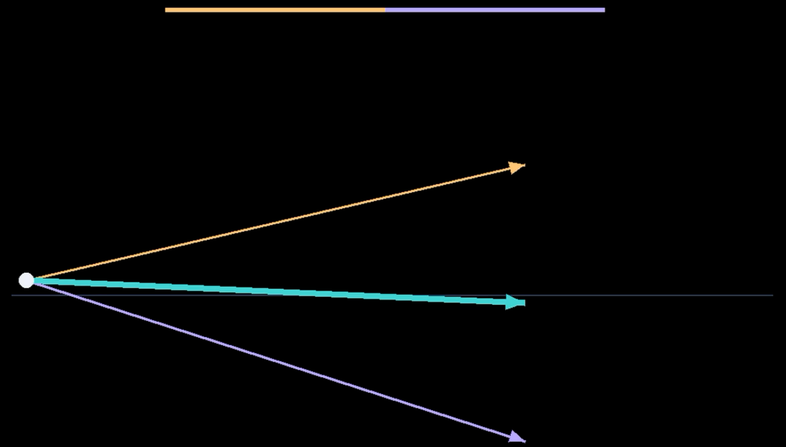
$$Z \sim \mathcal{N}((-2,0), I), \quad X \sim \frac{1}{2}\mathcal{N}((2,1.15), 0.09I) + \frac{1}{2}\mathcal{N}((2,-1.15), 0.09I), \quad Z \perp X.$$

Posterior weights are evaluated along the reference trajectory at the white points (t = 0.30, 0.76). RK4 reference: 2,000 steps. Euler interval: [0,1]. Endpoint-MSE statement assumes independent pairing.

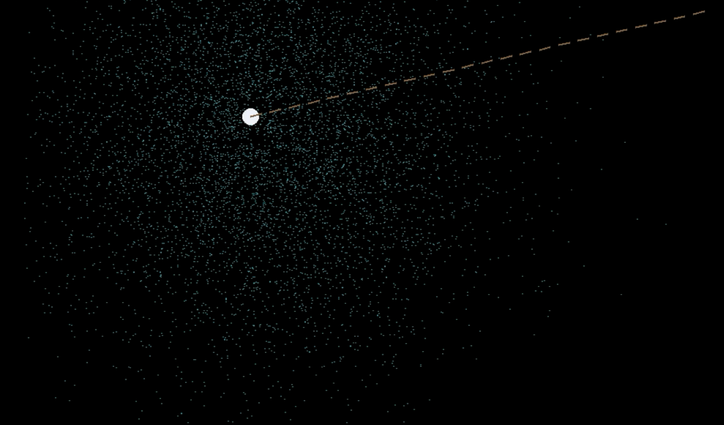
The collapse result assumes independent Z and X, finite second moments, a linear path and an exact population regressor. It does not rule out learned one-step generators.



Conditional paths



Mean velocity



Sampling trajectories

RTX 5090 · Exact mixture field · RK4 / 2,000 steps · 200,000 particles / 7,000 shown

t = 0.00

Diffusion’s marginal score supports ODE/SDE sampling; can Flow Matching’s velocity also support SDE sampling?

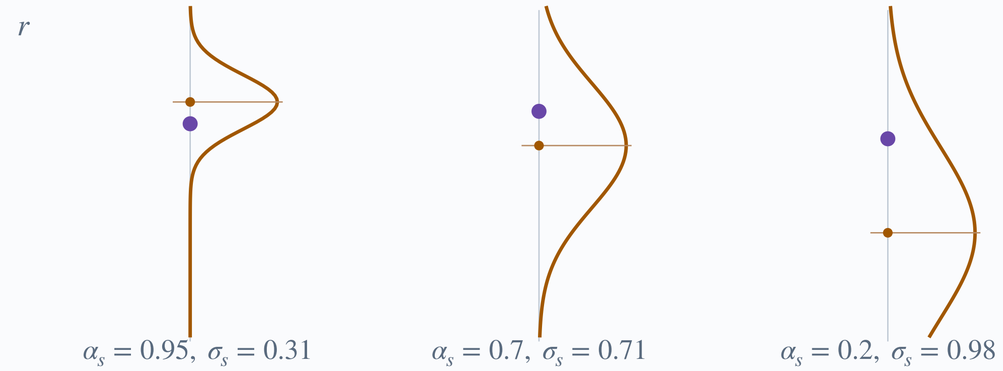
# Why conditional labels learn the marginal score



Gaussian noise supplies a known score label. Squared regression learns its posterior average, exactly as in Flow Matching.

## Known conditionals make supervision tractable

Fix X = 2. Draw a training state directly at any noise level.

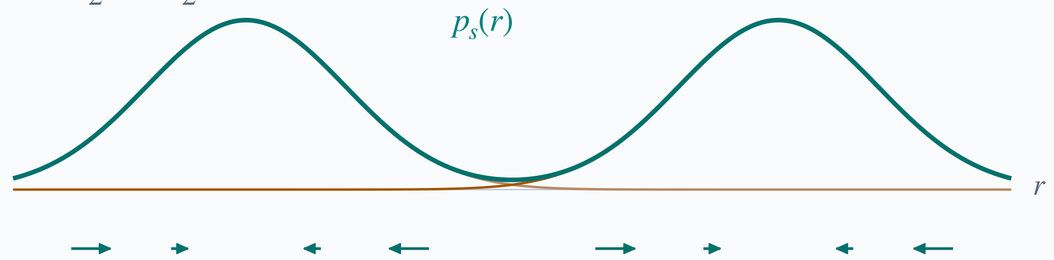



$$R_s = \alpha_s X + \sigma_s Z, \qquad Z \sim \mathcal{N}(0, I)$$

$$C_s := \nabla_r \log p_s(r \mid X)\Big|_{r=R_s} = -\frac{Z}{\sigma_s}$$

## Posterior averaging recovers the unknown score

$$P_{\text{data}} = \frac{1}{2}\delta_{-2} + \frac{1}{2}\delta_{2}, \quad \alpha_s = 0.8, \quad \sigma_s = 0.6$$




Bayesian posterior weights: the same averaging structure as P06.

$$P(X \in dx \mid R_s = r) = \frac{p_s(r \mid x) P_{\text{data}}(dx)}{p_s(r)}$$

$$\ell_s(r) := \nabla_r \log p_s(r) = \mathbb{E}[C_s \mid R_s = r]$$

## The same loss identity as in Flow Matching

$$\underbrace{\mathbb{E}\|\ell_\theta(R_s, s) - C_s\|^2}_{\mathcal{L}_{\text{conditional}}} = \underbrace{\mathbb{E}\|\ell_\theta(R_s, s) - \ell_s(R_s)\|^2}_{\mathcal{L}_{\text{marginal}}} + K$$

K is independent of the network parameters. Known labels optimize the marginal objective.

X, Z ∈ $R^d$; s ∈ (0,1] runs from data to noise. $R_s$ is a training draw. X and Z are independent; $\sigma_s > 0$. Losses shown at fixed s.

Assume finite moments and valid differentiation under the mixture integral. Gaussian kernels are a chosen family, not a property of every SDE. Sources: DDPM; Song et al. (2021).

# The score determines probability flux and sampling dynamics



One learned score supports reverse SDE and probability-flow ODE sampling, without retraining.

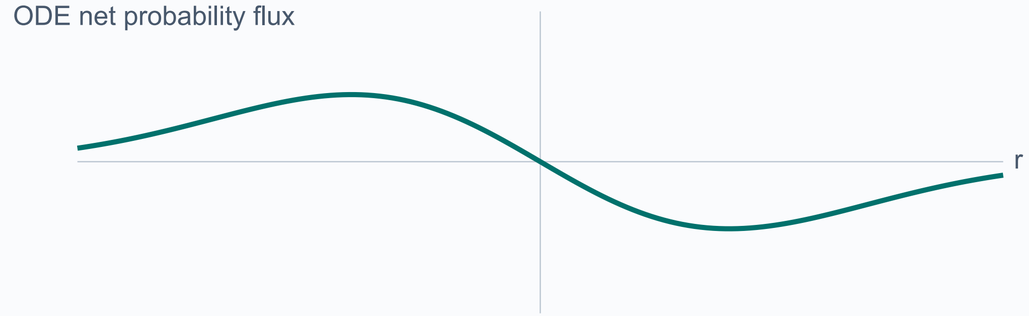
Fokker–Planck flux: forward time s runs from data to noise. The chosen schedule fixes drift f and noise strength g.

$$J_s(r) = p_s(r)\left[f(r,s) - \frac{1}{2}g(s)^2 \ell_s(r)\right] = p_s(r)v^{\mathrm{PF}}(r,s)$$

## The same continuity equation as in Flow Matching

$$\partial_s p_s(r) + \nabla_r \cdot \left[p_s(r)v^{\mathrm{PF}}(r,s)\right] = 0$$

$v^{Pf}$: probability-flow velocity (PF = Probability Flow).
Computed from the score and known schedule; no extra network.



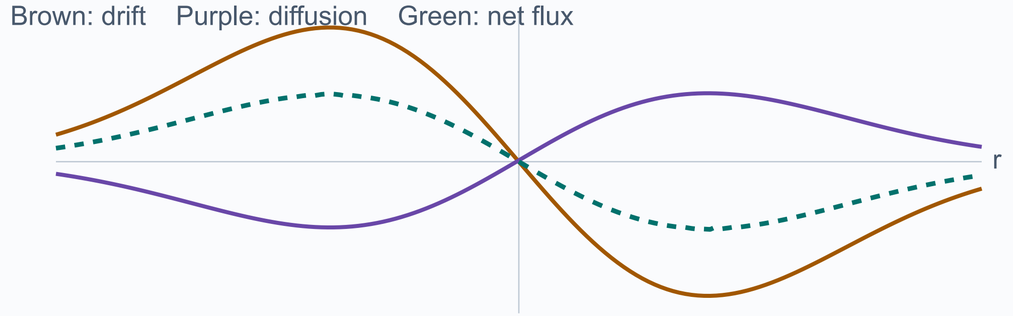

ODE: the drift alone carries the net probability flux.

## Generation reverses time: τ = 1 − s

$$b_{\mathrm{ODE}}(r,\tau) = -f(r,1-\tau) + \frac{g(1-\tau)^2}{2}\ell_{1-\tau}(r)$$

$$b_{\mathrm{SDE}}(r,\tau) = -f(r,1-\tau) + g(1-\tau)^2\ell_{1-\tau}(r)$$

The SDE adds noise; its drift compensates for diffusion.

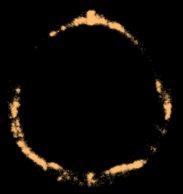



SDE: drift flux + diffusion flux = the same net flux.

With the exact score, matching initial distributions, and well-posed, unique evolution, both samplers have the same marginal distribution at every time, although their trajectories differ.
In practice, learning and numerical errors make this agreement approximate.

$$J_{\mathrm{ODE}}(r,\tau) = J_{\mathrm{SDE}}(r,\tau)$$

$r \in R^d$; g depends only on time; compatible noising schedule. Exact score, smooth densities, boundary decay, matched initial laws and uniqueness. Song et al. (2021).



Conditional target

w = 0

w = 1

w = 4

RTX 5090 · RK4 / 200 steps · 100,000 particles per run / 30,000 shown

t = 0.00

# Conditioning changes the target distribution; guidance changes the field



The condition selects which data remain plausible. Guidance amplifies that selection and can change the distribution produced by sampling.

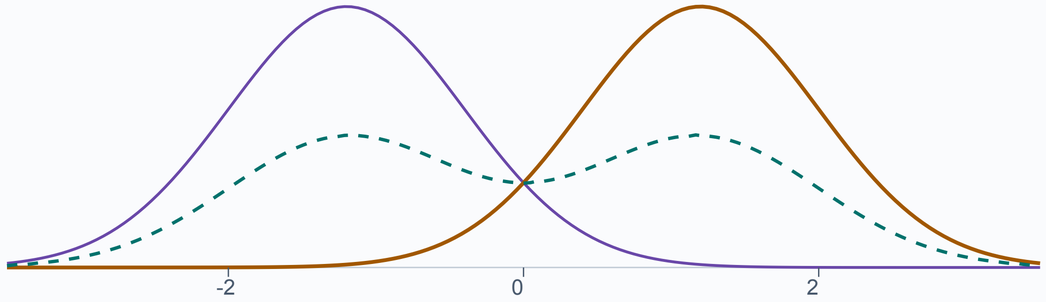
## Conditioning restricts the posterior average

$$(X, C) \sim P_{\text{data}}, \quad Z \perp (X, C), \quad R_t = (1 - t)Z + tX$$

$$v_t^*(r, c) = \mathbb{E}[X - Z \mid R_t = r, C = c]$$

The network keeps c during sampling and averages over endpoint uncertainty within that condition.

At one time t: equal class priors, means ±1.2, SD 0.8.




Purple: c = −  Gold: c = +  Dashed teal: unconditional

$$\Pr(C = + \mid R_t = 0) = \frac{1}{2}, \quad \Pr(C = + \mid R_t = 1.2) \approx 0.989$$

The condition resolves class ambiguity, while many possible samples can still remain within that class.

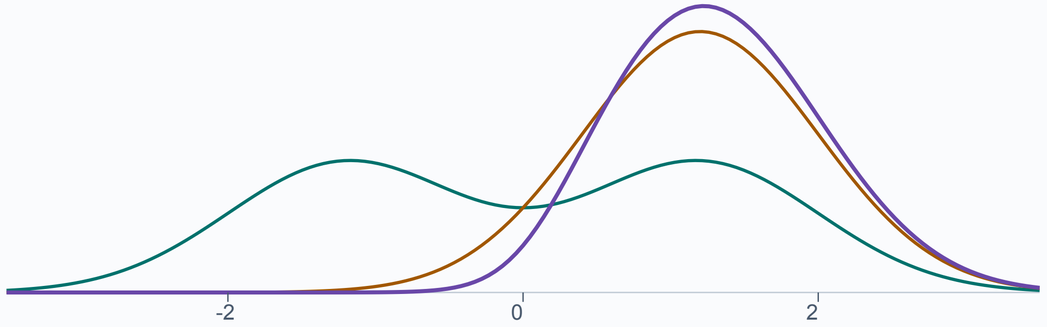
## Classifier-free guidance extrapolates the prediction

$$v_\theta^{(w)}(r, t, c) = v_\theta(r, t) + w[v_\theta(r, t, c) - v_\theta(r, t)]$$

$$\tilde{p}_t^{(w)}(r \mid c) \propto p_t(r)^{1-w} p_t(r \mid c)^w$$

w = 0: unconditional  w = 1: conditional
w > 1: stronger preference, without an exact-target guarantee.

Illustrative density reweighting, not generated marginals.




Teal: w = 0  Gold: w = 1  Purple: w = 2, c = +

$$\frac{dY_t}{dt} = v_\theta^{(w)}(Y_t, t, c)$$

Guidance changes the sampling velocity.
The reweighted curves are not guaranteed sampling marginals.

Condition dropout supplies the unconditional branch. c is an external label or prompt, distinct from conditioning on the training endpoint. Source: Ho & Salimans (2022).

# Beyond instantaneous velocity: learning maps and representations

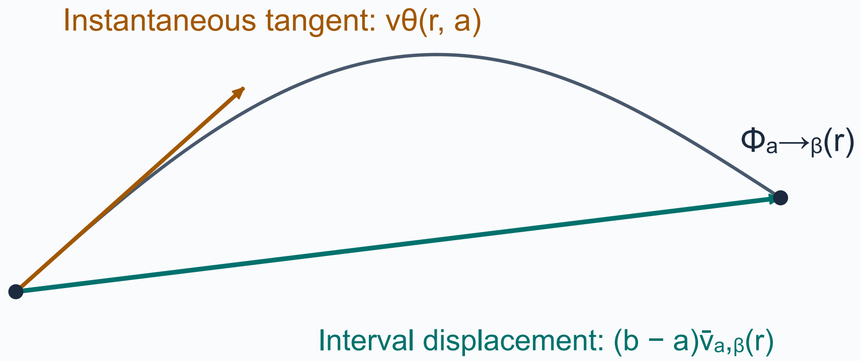


An instantaneous field answers a local question. A finite-interval map must encode what happens throughout the interval.

## Average velocity represents a finite displacement

$$\bar{v}_{a,b}(r) = \frac{\Phi_{a\to b}(r) - r}{b-a} = \frac{1}{b-a}\int_a^b v_\theta(\Phi_{a\to t}(r), t)\,dt$$

$$\Phi_{a\to b}(r) = r + (b-a)\bar{v}_{a,b}(r), \qquad 0 \le a < b \le 1$$



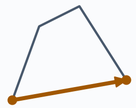

A single evaluation is useful only if training identifies this finite displacement. Taking one large Euler step does not learn it.

$$\partial_b[(b-a)\bar{v}_{a,b}(r)] = v_\theta(\Phi_{a\to b}(r), b)$$

$\bar{v}$ is an interval average, not the conditional training label u.

Φ is the exact flow map of vθ.

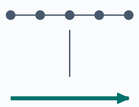
## Methods change different parts of learning



### MeanFlow / iMF

Learn interval-average velocity.

iMF reformulates the loss and exposes guidance controls.



### Distillation / FlowMap-OPD

Distill a teacher into a generator with fewer steps.

FlowMap-OPD separates rollouts from distribution comparisons.



### Drifting

Update the generated distribution during training.

Distribution comparisons replace an inference-time trajectory.



### Self-Flow

Learn representations with a cleaner EMA teacher.

Dual-timestep inputs create an information asymmetry.
This does not establish a one-step sampler.

Yes. A velocity learned through Flow Matching can also support SDE sampling.

For our Gaussian conditional path, velocity determines the marginal score.
Add noise together with the matching score-based drift correction.
The net probability flux—and the ideal time marginals—remain unchanged.

# What we learned about learning a sampler



1 The objective is an executable sampler whose output law approximates the data distribution. Density evaluation alone does not supply an efficient sampling procedure.

2 Conditional constructions turn observed data into tractable regression labels. The population regressor averages the hidden training variables conditional on the current state and time.

3 This conditional average preserves the probability flux of the chosen path. Under well-posed transport, it reproduces the prescribed marginals without preserving the original endpoint pairings.

4 On matched Gaussian paths, velocity and score give convertible descriptions of the same marginal evolution.

Conditional supervision identifies a marginal transport field

$$(R_t, U) \quad \longrightarrow \quad v_t^*(x) = \mathbb{E}[U \mid R_t = x] \quad \longrightarrow \quad \mathscr{L}(Y_t) = P_t$$

Exact population field, matched initial law and well-posed transport. Velocity–score conversion uses a common Gaussian path with nondegenerate coefficients.