



Computer Science &
Artificial Intelligence
Laboratory

An Introduction to Flow Matching and Diffusion Models

From differential equations to image, video and language generators

Johannes Herter

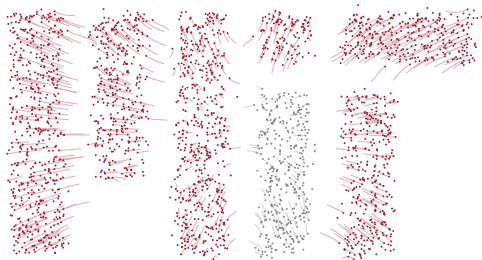
MIT CSAIL · 6.S184 · February 2026



The one idea of this lecture

Creating noise from data is easy; creating data from noise is generative modeling. [Song et al.]

- Data are vectors $z \in \mathbb{R}^d$; generating means **sampling** $z \sim p_{\text{data}}$.
- We will build a differential equation that carries $X_0 \sim \mathcal{N}(0, I_d)$ to $X_1 \sim p_{\text{data}}$.
- Its vector field is learned by plain **regression**, never simulating the equation in training.



$t = 1.00$

2600 Gaussian samples carried along the exact marginal flow of Section 3 onto the MIT logo, a mixture of 400 small Gaussians in its seven bars; each sample takes the colour of the bar it will reach.

1. **Generation as sampling**: what we are asked to do.
2. **Flow and diffusion models**: ODEs, SDEs, and how to simulate them.
3. **Flow matching**: probability paths, the marginalisation trick, the continuity equation, the CFM loss.
4. **Score matching and SDE sampling**: scores, denoisers, Fokker–Planck, Langevin.
5. **Guidance**: conditioning on a prompt; classifier-free guidance.
6. **Large-scale generators**: time embeddings, diffusion transformers, U-Nets, latent spaces and VAEs.
7. **Discrete diffusion**: CTMCs, the discrete marginalisation trick, masked language models.

Background. Probability and ODEs. Every figure is computed exactly on a toy mixture: it shows the theory, not a training run.

Toolbox: random vectors and conditioning (Appendix A)



Densities

$\mathbb{P}(X \in A) = \int_A p_X$. Gaussian (97):

$$\mathcal{N}(x; \mu, \Sigma) = \frac{\exp\left(-\frac{1}{2}(x - \mu)^\top \Sigma^{-1}(x - \mu)\right)}{\sqrt{(2\pi)^d \det \Sigma}}.$$

$$\begin{array}{ll} \text{Joint } p_{X,Y}, & \text{marginal } p_X(x) \\ \int p_{X,Y}(x, y) dy, & \text{conditional } p_{X|Y}(x|y) \end{array} = \frac{p_{X,Y}(x, y)}{p_Y(y)} \quad (101).$$

Bayes (102): $p_{Y|X} = \frac{p_{X|Y}p_Y}{p_X}$, the posterior of every marginalisation trick.

Conditional expectation

$g^\star = \operatorname{argmin}_g \mathbb{E}\|X - g(Y)\|^2$ is $g^\star(y) = \mathbb{E}[X|Y = y] = \int x p_{X|Y}(x|y) dx$ (103, 104): the **best least-squares predictor**. $\mathbb{E}[X|Y] := g^\star(Y)$ is a random vector (105).

Tower property (106): $\mathbb{E}[\mathbb{E}[X|Y]] = \mathbb{E}X$.

Here: Thms. 12, 22 are “least squares learns the conditional mean”.

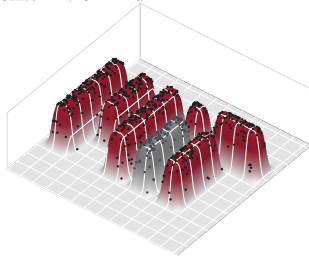
1 *Generative modeling as sampling*

Objects are vectors, generation is sampling

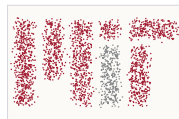
Key ideas 1–4

1. Objects are vectors $z \in \mathbb{R}^d$ (images, videos, molecules).
2. Generating z means sampling $z \sim p_{\text{data}}$.
3. We only have a **dataset** $z_1, \dots, z_N \sim p_{\text{data}}$.
4. Guided: sample $z \sim p_{\text{data}}(\cdot|y)$ for a prompt y .

$p_{\text{data}}(z)$ on $\mathbb{R}^2$ (height = density)



the dataset $z_1, \dots, z_{2400} \sim p_{\text{data}}$



the same draws, seen from above

generating = drawing a new
 $z \sim p_{\text{data}}$, knowing only samples

Running example: p_{data} shaped like the MIT logo. Each draw $z_n \sim p_{\text{data}}$ lands on the density surface (left) and in the dataset seen from above (right); the model must produce *new* draws.

2 *Flow and diffusion models*

A **vector field** $u : \mathbb{R}^d \times [0, 1] \rightarrow \mathbb{R}^d$,
 $(x, t) \mapsto u_t(x)$, defines the ODE

$$\frac{d}{dt} X_t = u_t(X_t), \quad X_0 = x_0. \quad (1)$$

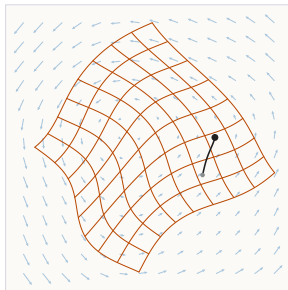
Its **flow** ψ solves

$$\frac{d}{dt} \psi_t(x_0) = u_t(\psi_t(x_0)), \quad \psi_0(x_0) = x_0. \quad (2)$$

Theorem 3 (existence and uniqueness)

$u \in C^1$ with bounded derivative $\Rightarrow$
(2) has a unique solution, and each ψ_t
is a diffeomorphism.

ψ_t (orange grid) and u_t (blue), $t = 1.00$



$$\frac{d}{dt} \psi_t(x_0) = u_t(\psi_t(x_0)), \quad \psi_0(x_0) = x_0$$

vector field $\rightarrow$ ODE $\rightarrow$ flow

black: one trajectory $X_t = \psi_t(x_0)$

each ψ_t is a diffeomorphism:
the grid bends, never tears or folds

Example 4 and simulating ODEs

Example 4. $u_t(x) = -\theta x$:

$\psi_t(x_0) = e^{-\theta t} x_0$, since

$$\frac{d}{dt} e^{-\theta t} x_0 = -\theta e^{-\theta t} x_0 = u_t(\psi_t(x_0)).$$

In general ψ_t is not explicit; we simulate.

Euler and Heun, $h = 1/n$

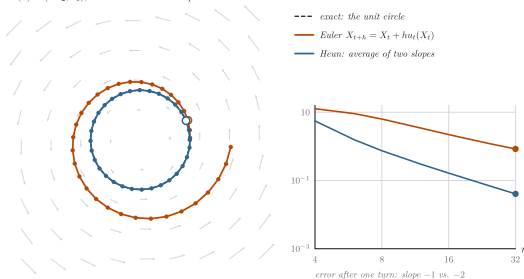
Euler: $X_{t+h} = X_t + h u_t(X_t)$.

Heun: $X'_{t+h} = X_t + h u_t(X_t)$,

$$X_{t+h} = X_t + \frac{h}{2} (u_t(X_t) + u_{t+h}(X'_{t+h})).$$

Euler has error $O(h)$, Heun $O(h^2)$.

$u(x) = (-x_2, x_1)$, one turn in $n = 32$ steps



The rotation field: Euler spirals outward, Heun almost closes the circle.

An ODE is deterministic; randomness enters through the start. A **flow model**:

$$X_0 \sim p_{\text{init}} = \mathcal{N}(0, I_d), \quad \frac{d}{dt} X_t = u_t^\theta(X_t),$$

with a neural network u^θ . **Goal:**

$$X_1 = \psi_1^\theta(X_0) \sim p_{\text{data}}.$$

Note. The network parametrises the *vector field*, not the flow: sampling means simulating the ODE.

Algorithm 1 (sampling a flow model)

```
h = 1/n, X0 ~ pinit, t = 0  
for i = 1, ..., n:  
  Xt+h = Xt + h utθ(Xt), t ← t + h  
return X1
```

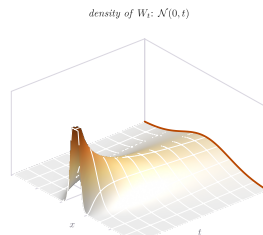
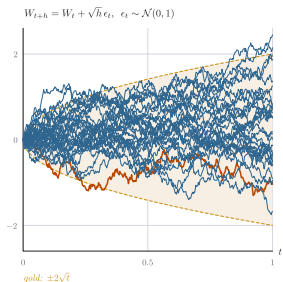
Brownian motion W

$W_0 = 0$, continuous paths, and

1. $W_t - W_s \sim \mathcal{N}(0, (t - s)I_d)$ for $s < t$;
2. increments over disjoint intervals are independent.

Simulation: $W_{t+h} = W_t + \sqrt{h} \epsilon_t$,
 $\epsilon_t \sim \mathcal{N}(0, I_d)$.

The scaling $\sqrt{h}$, not h : variance adds linearly in time.



Left: 30 paths, one highlighted. Right: the density of W_t spreading like $\mathcal{N}(0, t)$.

An ODE without derivatives: $X_{t+h} = X_t + h u_t(X_t) + h R_t(h)$, $R_t(h) \rightarrow 0$. Add a Brownian increment:

$$X_{t+h} = X_t + \underbrace{h u_t(X_t)}_{\text{deterministic}} + \underbrace{\sigma_t(W_{t+h} - W_t)}_{\text{stochastic}} + \underbrace{h R_t(h)}_{\text{error}}, \quad dX_t = u_t(X_t) dt + \sigma_t dW_t. \quad (6, 7)$$

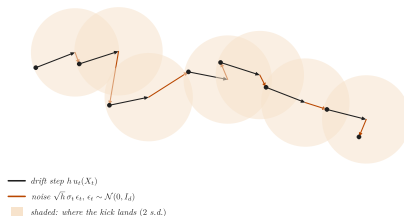
Theorem 5

$u \in C^1$ with bounded derivative, σ continuous $\Rightarrow$ (7) has a unique solution process.

Euler-Maruyama:

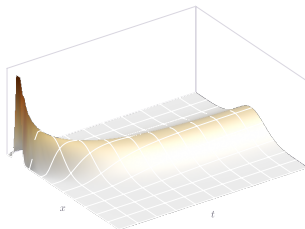
$$X_{t+h} = X_t + h u_t(X_t) + \sqrt{h} \sigma_t \epsilon_t \quad (9).$$

No flow map any more: X_0 does not determine X_t . Every ODE is the SDE with $\sigma = 0$.



Example 6: the Ornstein–Uhlenbeck process

density from $X_0 = 2.4$, $\theta = 1$, $\sigma = 1$

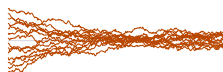


$\rightarrow \mathcal{N}(0, \frac{\sigma^2}{2\theta})$ as $t \rightarrow \infty$

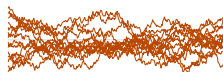
$\sigma = 0.0$ (a flow)



$\sigma = 0.6$



$\sigma = 1.2$



$$dX_t = -\theta X_t dt + \sigma dW_t$$

$dX_t = -\theta X_t dt + \sigma dW_t$: the drift pulls to 0, the noise spreads. Left: the exact density from $X_0 = 2.4$, $\mathcal{N}(x_0 e^{-\theta t}, \frac{\sigma^2}{2\theta}(1 - e^{-2\theta t}))$, converging to $\mathcal{N}(0, \frac{\sigma^2}{2\theta})$. Right: paths for three σ ; $\sigma = 0$ is the flow of Example 4.

Summary 7 (diffusion model)

Network $u^\theta : \mathbb{R}^d \times [0, 1] \rightarrow \mathbb{R}^d$, fixed $\sigma : [0, 1] \rightarrow [0, \infty)$:

$$X_0 \sim p_{\text{init}}, \quad dX_t = u_t^\theta(X_t) dt + \sigma_t dW_t,$$

goal $X_1 \sim p_{\text{data}}$. $\sigma \equiv 0$: a flow model.

Algorithm 2 (Euler–Maruyama sampling)

$h = 1/n$, $X_0 \sim p_{\text{init}}$, $t = 0$

for $i = 1, \dots, n$:

$\epsilon \sim \mathcal{N}(0, I_d)$

$X_{t+h} = X_t + h u_t^\theta(X_t) + \sigma_t \sqrt{h} \epsilon$

$t \leftarrow t + h$

return X_1

Open question: how to *train* u^θ so that $X_1 \sim p_{\text{data}}$? That is the rest of the lecture.

3 *Flow matching*

Step 1: a probability path

Fix what X_t should look like for $0 < t < 1$. A **conditional probability path**: distributions $p_t(\cdot|z)$ with

$$p_0(\cdot|z) = p_{\text{init}}, \quad p_1(\cdot|z) = \delta_z \quad (11)$$

and its **marginal path** $z \sim p_{\text{data}}, x \sim p_t(\cdot|z) \Rightarrow x \sim p_t, p_t(x) = \int p_t(x|z)p_{\text{data}}(z) dz$ (12, 13), so $p_0 = p_{\text{init}}, p_1 = p_{\text{data}}$ (14). We can *sample* p_t , but not evaluate its density.

Example 8 (Gaussian path)

Schedulers α_t, β_t , monotone, C^1 , $\alpha_0 = \beta_1 = 0, \alpha_1 = \beta_0 = 1$:

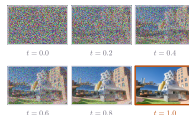
$$p_t(\cdot|z) = \mathcal{N}(\alpha_t z, \beta_t^2 I_d), \quad x = \alpha_t z + \beta_t \epsilon.$$

CondOT: $\alpha_t = t, \beta_t = 1 - t$.

$$x_t = tz + (1-t)\epsilon, \quad t = 1.00$$



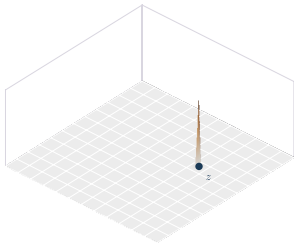
samples of $p_t(\cdot|z)$



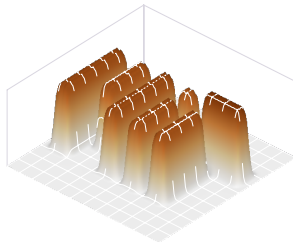
z : the Stata Center (CSAIL),
 $d = 192 \times 124 \times 3$; every pixel gets
independent noise, one schedule
 $\alpha_t = t, \beta_t = 1 - t$

Conditional and marginal paths

$$\text{conditional } p_t(x|z) = \mathcal{N}(tz, (1-t)^2 I)$$



$$\text{marginal } p_t(x) = \int p_t(x|z)p_{\text{data}}(z)dz$$



$t = 0.98$: almost data

(each surface scaled to its maximum)

Left: $p_t(\cdot|z)$ for one data point, a Gaussian shrinking onto z . Right: the marginal p_t for the logo, a mixture of all such Gaussians, which sharpens into the seven bars as $t \rightarrow 1$.

Step 2: a conditional vector field (Example 10)

Want $u_t(\cdot|z)$ whose ODE from $X_0 \sim p_{\text{init}}$ gives $X_t \sim p_t(\cdot|z)$ (17).

1

A flow that works. $\psi_t(x|z) := \alpha_t z + \beta_t x$ pushes $X_0 \sim \mathcal{N}(0, I_d)$ to $\mathcal{N}(\alpha_t z, \beta_t^2 I_d)$ (21).

2

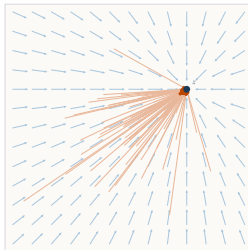
Differentiate. By (2): $\dot{\alpha}_t z + \dot{\beta}_t x = u_t(\alpha_t z + \beta_t x|z)$ for all x .

3

Substitute $x \mapsto \frac{x - \alpha_t z}{\beta_t}$:

$$u_t(x|z) = \left(\dot{\alpha}_t - \frac{\dot{\beta}_t}{\beta_t} \alpha_t \right) z + \frac{\dot{\beta}_t}{\beta_t} x. \quad (20) \blacksquare$$

$t = 0.96$: particles $X_t = tz + (1-t)X_0$



$\psi_t(x|z) = \alpha_t z + \beta_t x$ pushes $\mathcal{N}(0, I)$ to $\mathcal{N}(\alpha_t z, \beta_t^2 I)$

$\frac{d}{dt} \psi_t = \dot{\alpha}_t z + \dot{\beta}_t x$, substitute $x \mapsto \frac{x - \alpha_t z}{\beta_t}$:

$$u_t(x|z) = \left(\dot{\alpha}_t - \frac{\dot{\beta}_t}{\beta_t} \alpha_t \right) z + \frac{\dot{\beta}_t}{\beta_t} x$$

CondOT ($\alpha_t = t, \beta_t = 1 - t$): $u_t(x|z) = \frac{z - x}{1 - t}$

gold: the $1\sigma, 2\sigma$ circles of $p_t(\cdot|z)$
straight lines, constant speed

Theorem 9: the marginalisation trick

Theorem 9

If $u_t(\cdot|z)$ is a conditional vector field (17), the **marginal vector field**

$$u_t(x) = \int u_t(x|z) \frac{p_t(x|z)p_{\text{data}}(z)}{p_t(x)} dz \quad (18)$$

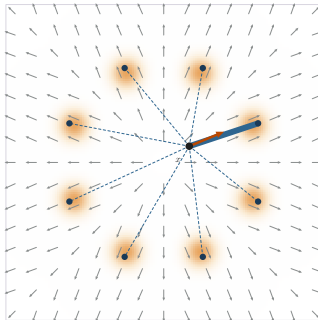
follows the marginal path: $X_0 \sim p_{\text{init}}, \frac{d}{dt} X_t = u_t(X_t) \Rightarrow X_t \sim p_t$, in particular $X_1 \sim p_{\text{data}}$ (19).

Reading. $\frac{p_t(x|z)p_{\text{data}}(z)}{p_t(x)}$ is the **posterior** of z given x (Bayes). So $u_t(x)$ averages the directions towards each data point, weighted by how plausibly x came from it:
 $u_t(x) = \mathbb{E}[u_t(x|z) \mid x_t = x].$

Why it is not obvious. Averaging vector fields does not average their flows. The proof needs a statement about densities, not trajectories: the continuity equation.

The marginal field as a posterior average

$t = 0.95$: p_t (shade), u_t (grey)

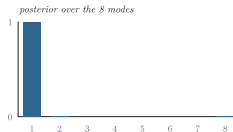


$$u_t(x) = \int u_t(x|z) \underbrace{\frac{p_t(x|z)p_{\text{data}}(z)}{p_t(x)}}_{\text{posterior of } z \text{ given } x} dz$$

blue: directions to each data mode,

thickness = posterior weight

orange: their average $u_t(x)$



Eight data modes. At a fixed x the blue lines point to every mode, thick where the posterior is large; their weighted average is the orange $u_t(x)$. Early on the posterior is spread out and u_t points to the mean; late it commits to one mode.

Theorem 11: the continuity equation

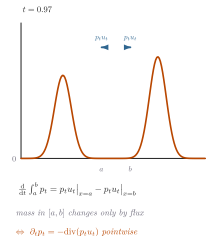
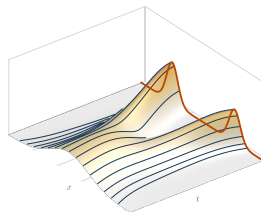
Theorem 11

For the ODE of u_t with $X_0 \sim p_0$: $X_t \sim p_t$ for all t iff

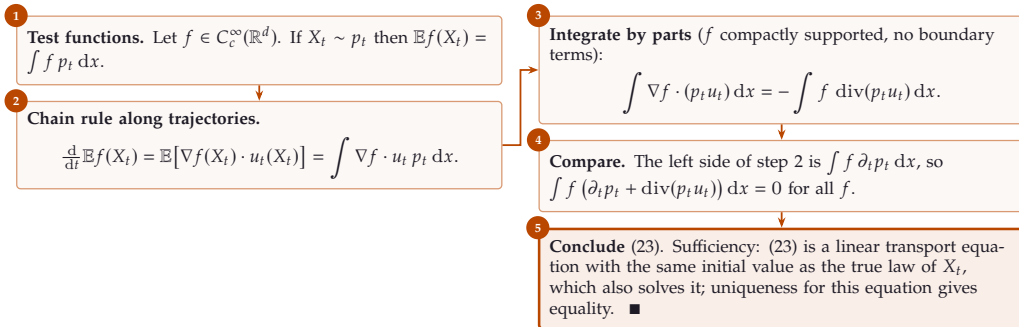
$$\partial_t p_t(x) = -\operatorname{div}(p_t u_t)(x). \quad (23)$$

$\operatorname{div} v = \sum_i \partial_{x_i} v^i$ (22). **Intuition:** $p_t u_t$ is a probability *flux*; $-\operatorname{div}$ of a flux is net inflow; mass is conserved, so the density changes only by inflow.

$p_t(x)$ and the trajectories (navy)

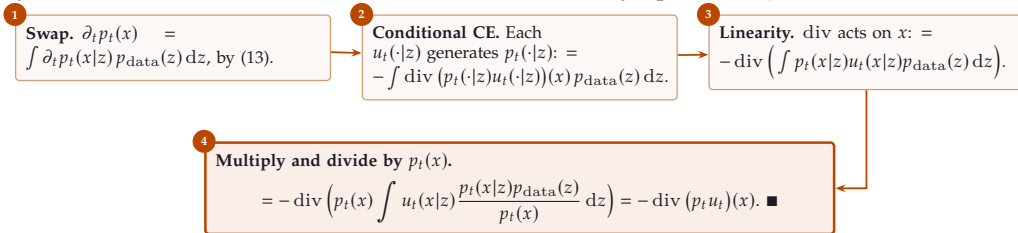


Proof of Theorem 11 (necessity)



Proof of Theorem 9

By Theorem 11 it suffices that u_t of (18) satisfies the continuity equation for p_t :

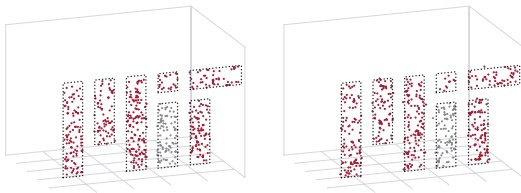


The point: the continuity equation is *linear* in the pair $(p_t, p_t u_t)$; mixing densities and fluxes with the same weights preserves it. Flows themselves do not mix linearly.

The marginal flow in space-time: ODE and SDE

probability-flow ODE $dX_t = u_t dt$

SDE $dX_t = \left(u_t + \frac{\sigma_t^2}{2} \nabla \log p_t\right) dt + \sigma_t dW_t$



$t = 1$: both reach p_{data}

Same noise, probability-flow ODE (left) and SDE of Thm. 17 (right, $\sigma_t = 0.9\sqrt{1-t}$). The ODE is deterministic: X_0 fixes the bar, so colours are set at $t = 0$ and paths never cross. SDE paths wander; the bar is decided only at the end (colour fades in). Both reach p_{data} .

Step 3: learning the marginal field

Flow matching loss (24, 25): $\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{t \sim \text{Unif}, x \sim p_t} \|u_t^\theta(x) - u_t(x)\|^2$. But $u_t(x)$ is an intractable integral.

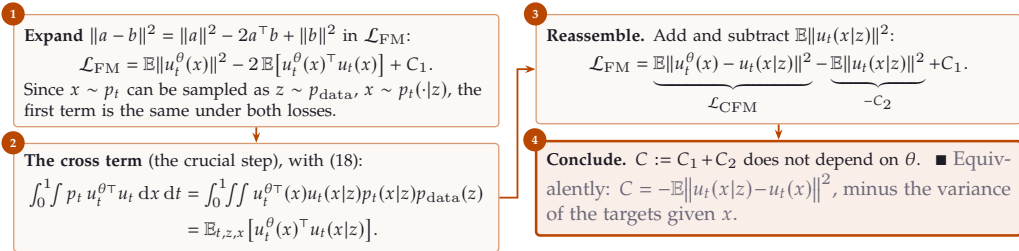
Conditional flow matching loss (26): $\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t \sim \text{Unif}, z \sim p_{\text{data}}, x \sim p_t(\cdot|z)} \|u_t^\theta(x) - u_t(x|z)\|^2$: fully tractable.

Theorem 12

$\mathcal{L}_{\text{FM}}(\theta) = \mathcal{L}_{\text{CFM}}(\theta) + C$ with C independent of θ . Hence $\nabla_\theta \mathcal{L}_{\text{FM}} = \nabla_\theta \mathcal{L}_{\text{CFM}}$, and the minimiser of $\mathcal{L}_{\text{CFM}}$ is $u_t^{\theta^*} = u_t$ (for an expressive enough model).

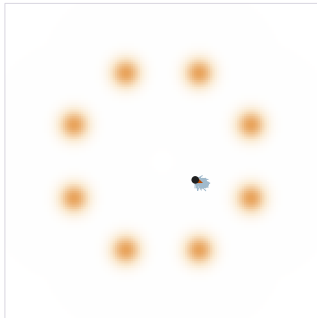
In one line. $u_t(x) = \mathbb{E}[u_t(x|z) \mid x]$: regressing on noisy targets learns their conditional mean. This is the bias–variance identity of least squares.

Proof of Theorem 12



Noisy targets, exact mean

$t = 0.94$, fixed x



blue: CFM targets $u_t(x|z)$, $z \sim p(z|x, t)$

orange: their mean, the FM target $u_t(x)$

$u_t(x) = \mathbb{E}[u_t(x|z) \mid x]$ so for every u :

$$\mathbb{E}\|u - u_t(x|z)\|^2 = \|u - u_t(x)\|^2 + \mathbb{E}\|u_t(x|z) - u_t(x)\|^2$$

the second term does not depend on θ :

$$\mathcal{L}_{\text{CFM}} = \mathcal{L}_{\text{FM}} + C$$

regressing on noisy labels learns their mean

as $t \rightarrow 1$ the posterior concentrates

At a fixed (x, t) , each training example contributes a target $u_t(x|z)$ with z drawn from the posterior; their spread is the constant of Theorem 12, their mean is the marginal field. As $t \rightarrow 1$ the posterior concentrates and the targets agree.

Algorithm 3: flow matching training

Algorithm 3 (CondOT path $p_t(x|z) = \mathcal{N}(tz, (1-t)^2 I_d)$)

Require: dataset $z \sim p_{\text{data}}$, network u_t^θ
for each mini-batch **do**
 $z \sim \text{dataset}, t \sim \text{Unif}[0, 1], \epsilon \sim \mathcal{N}(0, I_d)$
 $x = tz + (1-t)\epsilon$ general: $x \sim p_t(\cdot|z)$
 $\mathcal{L}(\theta) = \|u_t^\theta(x) - (z - \epsilon)\|^2$ general:
 $\|u_t^\theta(x) - u_t(x|z)\|^2$
 $\theta \leftarrow \text{grad_update}(\mathcal{L}(\theta))$
end for

Three striking features.

1. **Simulation-free:** no ODE is rolled out in training, which makes training cheap.
2. **Plain regression** on an explicit target: close to supervised learning.
3. **Hardly simpler possible:** sample, interpolate, regress.

After training, sample with Algorithm 1 on $dX_t = u_t^\theta(X_t)dt, X_0 \sim p_{\text{init}}$ (27). This whole pipeline is *flow matching* [25, 27, 1, 26].

Example 13: flow matching for Gaussian paths

With $x = \alpha_t z + \beta_t \epsilon$ in (20): $u_t(x|z) = \dot{\alpha}_t z + \dot{\beta}_t \epsilon$, so

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t,z,\epsilon} \|u_t^\theta(\alpha_t z + \beta_t \epsilon) - (\dot{\alpha}_t z + \dot{\beta}_t \epsilon)\|^2. \quad (31)$$

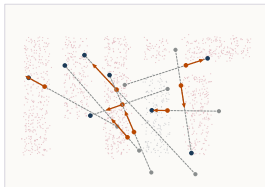
CondOT: target $z - \epsilon$:

$$\mathcal{L}_{\text{CFM}} = \mathbb{E} \|u_t^\theta(tz + (1-t)\epsilon) - (z - \epsilon)\|^2.$$

Used by Stable Diffusion 3, Meta Movie Gen and many others.

Simulation-free. Training never simulates an ODE. Sampling does.

batch 4: 8 pairs, each with its own random t



for each batch:

$$z \sim p_{\text{data}}, \epsilon \sim \mathcal{N}(0, I), t \sim \text{Unif}[0, 1]$$

$$x = tz + (1-t)\epsilon \text{ (orange dot)}$$

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

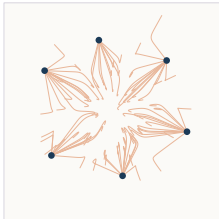
$$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(\theta)$$

each arrow is a regression label for u_t^θ at x ;

no ODE is simulated in training

Two warnings

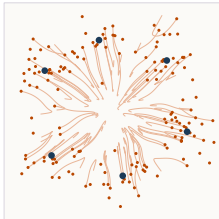
$p_{\text{data}} = \frac{1}{N} \sum_i \delta_{z_i}$ (the dataset itself)



every sample is a training point

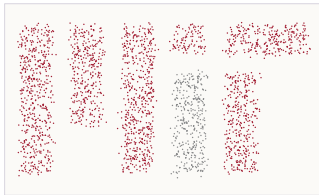
a smoothed p_{data} (N blobs)

$t = 1.00$



new samples between the data

Euler with $n = 100$ steps on the exact u_t



data: the MIT logo as a mixture
of 400 small Gaussians

few steps: the curved
trajectories are cut short

$n = 1$: $u_0(x) = \mathbb{E}[z] - x$, so
every sample lands on $\mathbb{E}[z]$

many steps: the logo appears

The exact minimiser memorises. For $p_{\text{data}} = \frac{1}{N} \sum \delta_{z_i}$ the optimal u_t sends every sample to a training point. Generalisation comes from the network *not* fitting u_t exactly.

Sampling has a cost. Euler on the exact field of the logo: with $n = 1$, $u_0(x) = \mathbb{E}z - x$ lands every sample on $\mathbb{E}z$; the curved trajectories need many steps.

Summary 14: flow matching

1. Choose a conditional path $p_t(x|z)$ with $p_0(\cdot|z) = p_{\text{init}}, p_1(\cdot|z) = \delta_z$.
2. Find $u_t(x|z)$ generating it (by hand, from a flow $\psi_t(x|z)$).
3. The marginal $u_t(x) = \int u_t(x|z) \frac{p_t(x|z)p_{\text{data}}(z)}{p_t(x)} dz$ generates p_t , hence $X_1 \sim p_{\text{data}}$ (Thm. 9, via Thm. 11).
4. Learn it by regressing on the conditional field: $\mathcal{L}_{\text{CFM}}$ (Thm. 12).

Gaussian paths

$$p_t(x|z) = \mathcal{N}(\alpha_t z, \beta_t^2 I_d), \quad u_t(x|z) = \left(\dot{\alpha}_t - \frac{\dot{\beta}_t}{\beta_t} \alpha_t \right) z + \frac{\dot{\beta}_t}{\beta_t} x, \quad \mathcal{L}_{\text{CFM}} = \mathbb{E} \|u_t^\theta(\alpha_t z + \beta_t \epsilon) - (\dot{\alpha}_t z + \dot{\beta}_t \epsilon)\|^2.$$

4

Score functions and score matching

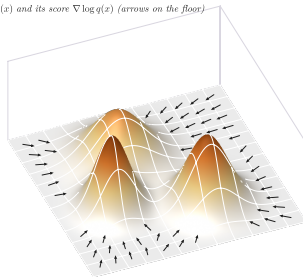
The **score** of a density q : $\nabla \log q(x)$.

Conditional score $\nabla \log p_t(x|z)$, marginal score $\nabla \log p_t(x)$, and the same averaging as (18):

$$\nabla \log p_t(x) = \int \nabla \log p_t(x|z) \frac{p_t(x|z)p_{\text{data}}(z)}{p_t(x)} dz$$

(38). **Proof** (39): $\nabla \log p_t = \frac{\nabla p_t}{p_t}$, differentiate (13) under the integral and use $\nabla p_t(x|z) = p_t(x|z)\nabla \log p_t(x|z)$. ■

$q(x)$ and its score $\nabla \log q(x)$ (arrows on the floor)



$\nabla \log q = \frac{\nabla q}{q}$: steepest ascent
of the log-likelihood

long arrows in the tails,
zero at every mode and saddle

no normalising constant needed:

$$\nabla \log(cq) = \nabla \log q$$

Proposition 1: scores are vector fields

Example 15. Gaussian path: $\nabla \log p_t(x|z) = -\frac{x - \alpha_t z}{\beta_t^2}$ (40), linear in x and z , like $u_t(x|z)$.

Proposition 1

For $p_t(\cdot|z) = \mathcal{N}(\alpha_t z, \beta_t^2 I_d)$, with $a_t = \beta_t^2 \frac{\dot{\alpha}_t}{\alpha_t} - \dot{\beta}_t \beta_t$ and $b_t = \frac{\dot{\alpha}_t}{\alpha_t}$:

$$u_t(x|z) = a_t \nabla \log p_t(x|z) + b_t x, \quad u_t(x) = a_t \nabla \log p_t(x) + b_t x. \quad (41, 42)$$

1

Conditional. Write z -coefficient and x -coefficient of (20) through $\frac{\alpha_t z - x}{\beta_t^2}$:

$$\left(\dot{\alpha}_t - \frac{\dot{\beta}_t \alpha_t}{\beta_t} \right) z + \frac{\dot{\beta}_t}{\beta_t} x = a_t \frac{\alpha_t z - x}{\beta_t^2} + b_t x.$$

2

Marginal. Average both sides against the posterior: the left gives $u_t(x)$ by (18), the right $a_t \nabla \log p_t(x) + b_t x$ by (38), as the posterior integrates to 1. ■

Remark 16: the denoiser and Tweedie's formula

Marginally, u_t and the score are affine in the **posterior mean**

$$D_t(x) = \mathbb{E}[z \mid x_t = x],$$

the **denoiser** (43). Averaging (40):

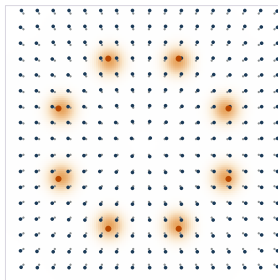
$$\nabla \log p_t(x) = \frac{\alpha_t D_t(x) - x}{\beta_t^2}$$

(Tweedie), and

$$u_t = \dot{\alpha}_t D_t + \dot{\beta}_t \frac{x - \alpha_t D_t}{\beta_t}.$$

Hence *denoising* diffusion models.

$t = 0.97$: grey $x \mapsto$ navy $D_t(x) = \mathbb{E}[z \mid x_t = x]$



the denoiser: best guess of the clean z

small t : all noise, every guess is $\mathbb{E}[z]$

large t : snaps to the nearest mode

Tweedie: the score is the denoiser

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

so u_t , $\nabla \log p_t$, D_t , ϵ_t are

affine reparametrisations of each other

Theorem 17: the SDE extension trick

Theorem 17

For *any* diffusion coefficient $\sigma_t \geq 0$,

$$X_0 \sim p_{\text{init}}, \quad dX_t = \left[u_t(X_t) + \frac{\sigma_t^2}{2} \nabla \log p_t(X_t) \right] dt + \sigma_t dW_t \quad \implies \quad X_t \sim p_t \quad (0 \leq t \leq 1). \quad (44, 45)$$

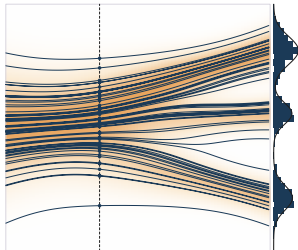
Example 18 (Gaussian paths). Everything through the score:

$$dX_t = \left[\left(a_t + \frac{\sigma_t^2}{2} \right) \nabla \log p_t(X_t) + b_t X_t \right] dt + \sigma_t dW_t.$$

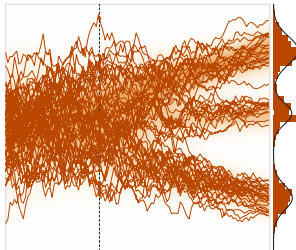
Striking: σ_t can be chosen *after* training. In theory every σ_t is exact; in practice training and discretisation errors make some σ_t better, found empirically.

Same marginals, different paths

ODE: $dX_t = u_t dt$



SDE: $dX_t = [u_t + \frac{\sigma_t^2}{2} \nabla \log p_t] dt + \sigma_t dW_t$



$$\sigma_t = 2.5\sqrt{1-t}$$

same initial noise; shade: p_t (x -marginal); right of each panel: histogram at $t=1$ vs. p_{data}

Three data modes on a line; ODE (left) and SDE (right) from the same noise, for growing σ . Individual paths change completely, the distribution at every t does not: both endpoint histograms match p_{data} .

Theorem 19: the Fokker–Planck equation

Theorem 19

For $X_0 \sim p_{\text{init}}$, $dX_t = u_t(X_t)dt + \sigma_t dW_t$: $X_t \sim p_t$ for all t iff

$$\partial_t p_t(x) = -\operatorname{div}(p_t u_t)(x) + \frac{\sigma_t^2}{2} \Delta p_t(x), \quad \Delta = \operatorname{div} \nabla. \quad (49)$$

1

One Euler–Maruyama step $\delta = hu + \sigma\sqrt{h}\epsilon$; Taylor for a test function f :
 $f(X + \delta) = f + \nabla f \cdot \delta + \frac{1}{2} \delta^\top \nabla^2 f \delta + O(|\delta|^3)$.

2

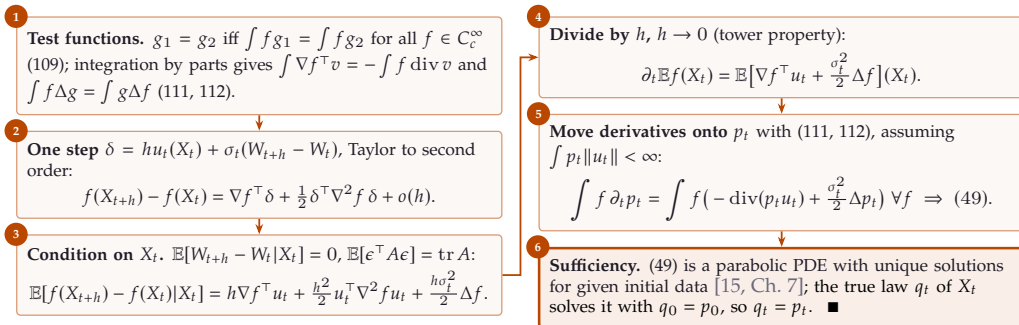
Expectations. $\mathbb{E}\delta = hu$, $\mathbb{E}\delta\delta^\top = \sigma^2 h I_d + O(h^2)$: $\frac{d}{dt} \mathbb{E}f(X_t) = \mathbb{E}[\nabla f \cdot u + \frac{\sigma^2}{2} \Delta f]$.

3

Integrate by parts once and twice:
 $\int f \partial_t p_t = \int f (-\operatorname{div}(p_t u_t) + \frac{\sigma^2}{2} \Delta p_t)$
for all f : (49). ■

Why $\sqrt{h}$ matters: the noise is $O(\sqrt{h})$, so its *second-order* term is $O(h)$ and survives the limit: the Laplacian. $\sigma = 0$ recovers the continuity equation; $u = 0$ the heat equation.

Proof of Theorem 19 (Appendix B)



Proof of Theorem 17

Check Fokker–Planck for the drift $u_t + \frac{\sigma_t^2}{2} \nabla \log p_t$:

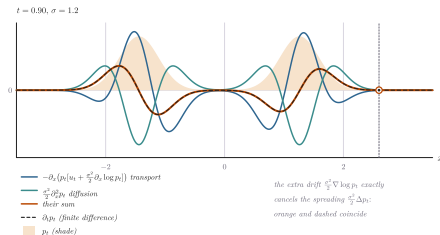
$$\partial_t p_t \stackrel{(i)}{=} -\operatorname{div}(p_t u_t)$$

$$\stackrel{(ii)}{=} -\operatorname{div}(p_t u_t) - \frac{\sigma_t^2}{2} \Delta p_t + \frac{\sigma_t^2}{2} \Delta p_t$$

$$\stackrel{(iii)}{=} -\operatorname{div}(p_t u_t) - \operatorname{div}\left(\frac{\sigma_t^2}{2} \nabla p_t\right) + \frac{\sigma_t^2}{2} \Delta p_t$$

$$\stackrel{(iv)}{=} -\operatorname{div}\left(p_t \left[u_t + \frac{\sigma_t^2}{2} \nabla \log p_t\right]\right) + \frac{\sigma_t^2}{2} \Delta p_t.$$

(i) Thm. 11; (ii) add zero; (iii) $\Delta = \operatorname{div} \nabla$; (iv) $\nabla p_t = p_t \nabla \log p_t$. Thm. 19 concludes. ■



The two terms of (49) for $\sigma = 1.2$ in 1D, computed on the exact p_t : their sum is $\partial_t p_t$.

Remark 20: Langevin dynamics

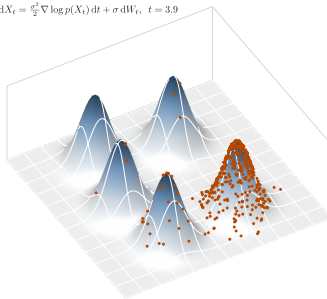
A constant path $p_t = p$ and $u = 0$ in Theorem 17:

$$dX_t = \frac{\sigma_t^2}{2} \nabla \log p(X_t) dt + \sigma_t dW_t. \quad (50)$$

Then p is **stationary**: $X_0 \sim p \Rightarrow X_t \sim p$, and under mild conditions $X_t \rightarrow p$ from any start.

The basis of MCMC and molecular dynamics; with p Gaussian, the Ornstein–Uhlenbeck process.

$$dX_t = \frac{\sigma_t^2}{2} \nabla \log p(X_t) dt + \sigma_t dW_t, \quad t = 3.9$$



particles start in a corner
and settle into p :

p is stationary
(Fokker-Planck with $u = 0$)

the basis of MCMC and
molecular dynamics

Remark 21: stochastic transitions with ODEs

What SDEs buy. With $\sigma_t > 0$ the evolution is random: X_0 does not determine X_t . Many children from one parent, which search, guidance and alignment methods exploit.

GLASS flows [20]: the same stochastic transitions $X_t \mapsto X_{t'}$ can be produced *purely with ODEs* by a sampling trick, keeping the efficiency of deterministic solvers.

In practice. For a fixed trained model and step budget there is an empirically best σ_t [23, 1, 28]: an artefact of training and discretisation error, not of the continuous theory, where every σ_t is exact.

Two kinds of randomness.

- **ODE:** randomness only in X_0 ; deterministic map noise $\rightarrow$ sample.
- **SDE:** fresh randomness at every step; the marginals p_t are the same.

Theorem 22: score matching

Score network $s_t^\theta : \mathbb{R}^d \times [0, 1] \rightarrow \mathbb{R}^d$. As before, a marginal and a conditional loss:

$$\mathcal{L}_{\text{SM}} = \mathbb{E} \|s_t^\theta(x) - \nabla \log p_t(x)\|^2, \quad \mathcal{L}_{\text{CSM}} = \mathbb{E}_{t,z,x \sim p_t(\cdot|z)} \|s_t^\theta(x) - \nabla \log p_t(x|z)\|^2.$$

Theorem 22

$\mathcal{L}_{\text{SM}} = \mathcal{L}_{\text{CSM}} + C$; proof: Thm. 12 verbatim, since (38) has the shape of (18).

Example 23 (Gaussian): with $x = \alpha_t z + \beta_t \epsilon$,

$$\mathcal{L}_{\text{CSM}} = \mathbb{E} \frac{1}{\beta_t^2} \|\beta_t s_t^\theta(\alpha_t z + \beta_t \epsilon) + \epsilon\|^2 :$$

the network predicts the noise. Unstable as $\beta_t \rightarrow 0$.

DDPM [17]: drop $\frac{1}{\beta_t^2}$, set $\epsilon_t^\theta = -\beta_t s_t^\theta$:

$$\mathcal{L}_{\text{DDPM}} = \mathbb{E} \|\epsilon_t^\theta(\alpha_t z + \beta_t \epsilon) - \epsilon\|^2.$$

Same minimiser, reweighted over t .

Algorithm 4: denoising score matching

Algorithm 4 (Gaussian path)

Require: dataset $z \sim p_{\text{data}}$, score network s_t^θ or noise predictor ϵ_t^θ
for each mini-batch **do**
 $z \sim \text{dataset}, t \sim \text{Unif}[0, 1], \epsilon \sim \mathcal{N}(0, I_d)$
 $x_t = \alpha_t z + \beta_t \epsilon$
 $\mathcal{L}(\theta) = \|s_t^\theta(x_t) + \frac{\epsilon}{\beta_t}\|^2$ or $\|\epsilon_t^\theta(x_t) - \epsilon\|^2$
 update θ by gradient descent
end for

Why “denoising”. The target $-\epsilon/\beta_t$ is the scaled noise that corrupted z : the network learns to recognise the noise in x_t .

Why drop $1/\beta_t^2$. As $\beta_t \rightarrow 0$ the score target explodes; the DDPM loss reweights time so all t contribute comparably. The minimiser is unchanged, as each t is minimised separately.

Summary 24: four parametrisations of one object

For Gaussian paths

Each determines the others affinely in x :

- vector field $u_t(x)$;
- score $\nabla \log p_t(x) = \frac{u_t(x) - b_t x}{a_t}$;
- denoiser $D_t(x) = \mathbb{E}[z|x]$;
- noise predictor $\epsilon_t(x) = -\beta_t \nabla \log p_t(x)$.

Sampling with any $\sigma_t \geq 0$:

$$dX_t = \left[\left(1 + \frac{\sigma_t^2}{2a_t} \right) u_t^\theta(X_t) - \frac{\sigma_t^2 b_t}{2a_t} X_t \right] dt + \sigma_t dW_t. \quad (55)$$

Which to learn? Equivalent in theory; they differ in how the loss weights different t and in numerical conditioning near $t = 0, 1$.

5

Guidance: how to condition on a prompt

A **guided** model $u_t^\theta(x|y)$ for a prompt $y \in \mathcal{Y}$ (class label, text). Samples: $X_0 \sim p_{\text{init}}$, $dX_t = u_t^\theta(X_t|y)dt + \sigma_t dW_t$, goal $X_1 \sim p_{\text{data}}(\cdot|y)$.

Nothing changes in the path: fix y , train on $p_{\text{data}}(\cdot|y)$; then average over y :

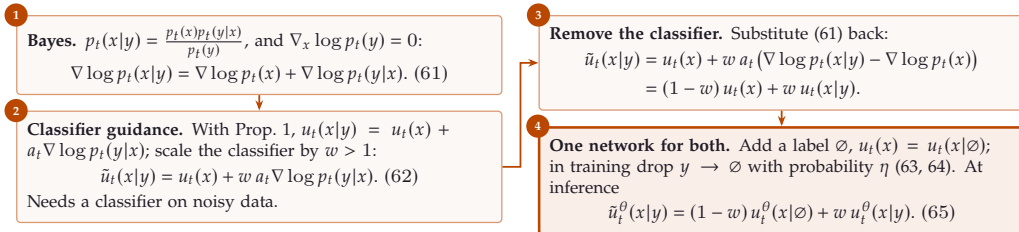
$$\mathcal{L}_{\text{CFM}}^{\text{guided}}(\theta) = \mathbb{E}_{(z,y) \sim p_{\text{data}}, t, x \sim p_t(\cdot|z)} \|u_t^\theta(x|y) - u_t(x|z)\|^2. \quad (58)$$

The dataloader now returns pairs (z, y) .

Terminology. “Conditional” refers to conditioning on z (paths, fields); *guided* refers to the prompt y .

In practice samples fit the prompt too loosely: underfitting and noisy text–image pairs. We need to *amplify* y .

From classifier guidance to classifier-free guidance

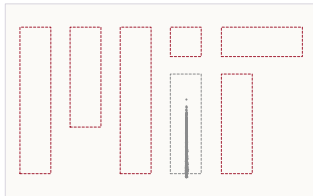


Remark 26. (65) works for any path; exact for $w = 1$, *not* $p_{\text{data}}(\cdot|y)$ for $w \neq 1$.

The prompt here is “the grey bar” of the logo. CFG is a heuristic justified by results: nearly every AI-generated image uses $w \geq 4$.

It sharpens towards $p(x) p(y|x)^w$: at large w the samples collapse to the most typical part of the bar, the price of prompt adherence is diversity.

$w = 4.0$



prompt $y =$ “the grey bar”: 100% of samples on it

$$\tilde{u}_t(x|y) = (1 - w) u_t(x|\emptyset) + w u_t(x|y)$$

$w = 0$: the whole logo

$w = 1$: $p_{\text{data}}(\cdot|y)$, the grey bar

$w > 1$: pushed away from $u(\cdot|\emptyset)$:
tighter, more “on-prompt”

equivalently: $\text{sample} \propto p_t(x) p_t(y|x)^w$

$$\nabla \log p_t(x) + w \nabla \log p_t(y|x)$$

dashed: the seven bars

Algorithm 5 (CFG training, Gaussian path)

```
for each mini-batch do  
   $(z, y) \sim p_{\text{data}}, t \sim \text{Unif}, \epsilon \sim \mathcal{N}(0, I_d)$   
   $x = \alpha_t z + \beta_t \epsilon$   
  with probability  $\eta$ :  $y \leftarrow \emptyset$   
   $\mathcal{L} = \|u_t^\theta(x|y) - (\dot{\alpha}_t z + \dot{\beta}_t \epsilon)\|^2$ , update  $\theta$   
end for
```

Summary 27. One network learns both $u_t(x|\emptyset)$ and $u_t(x|y)$ (66, 67). At inference, for a fixed y :

$$dX_t = \tilde{u}_t^\theta(X_t|y)dt,$$

$\tilde{u}^\theta = (1 - w)u^\theta(\cdot|\emptyset) + wu^\theta(\cdot|y)$, $X_0 \sim p_{\text{init}}$. X_1 is *not* distributed as $p_{\text{data}}(\cdot|y)$ for $w > 1$, but adheres better to y .

Remark 28. For diffusion models replace $u_t^\theta(x|y)$ by $\tilde{u}_t^\theta(x|y)$ and sample with the SDE of Section 4.

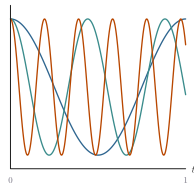
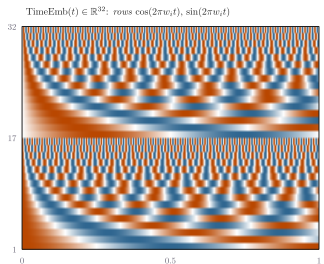
6

*Building large-scale image and
video generators*

Embedding the inputs t and y

The network takes (x, y, t) and returns a vector in $\mathbb{R}^d$.

- **Time:** Fourier features (68, 69), unit norm, frequencies from coarse to fine.
- **Class labels:** a learned embedding vector per class (and one for $\emptyset$).
- **Text:** frozen pretrained encoders: CLIP (one vector per prompt) and/or a transformer (a sequence, $\text{PromptEmb}(y) \in \mathbb{R}^{S \times k}$).

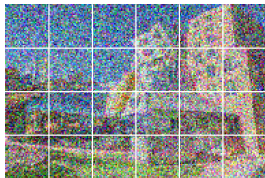


$$w_i = w_{\min} (w_{\max} / w_{\min})^{\frac{i-1}{N-1}}$$

slow rows: coarse time, fast rows: fine;

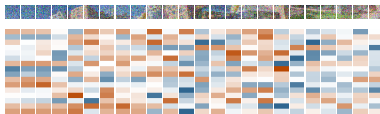
$\|\text{TimeEmb}(t)\| = 1$ for every t

input x_t ($t = 0.6$): the noisy Stata Center



$P = 32: 4 \times 6 = N = 24$ patches

tokens $\text{Patchify}(x_t) \in \mathbb{R}^{N \times CP^2}$



embeddings $\text{Patchify}(x_t)W + \text{pos} \in \mathbb{R}^{N \times d}$, $d = 16$ (blue < 0 < orange)

$\times L$ blocks, then

deparchify



$u_t^\theta(x_t|y) \in \mathbb{R}^{3 \times 128 \times 192}$

same shape as x_t

shown: the exact target $z - \epsilon$

6. output velocity

A DiT on the noisy Stata Center: $x_t \in \mathbb{R}^{C \times H \times W}$ is cut into $N = HW/P^2$ patches, each linearly embedded (columns: the actual 16-dim vectors) with a position code; self-attention lets every token weigh every patch; after L blocks (70), a linear map and depatchify return $u_t^\theta(x_t|y)$ with the shape of x_t .

Remark 29: inside a DiT block

Attention:

$$\text{Attn}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_h}}\right)V,$$

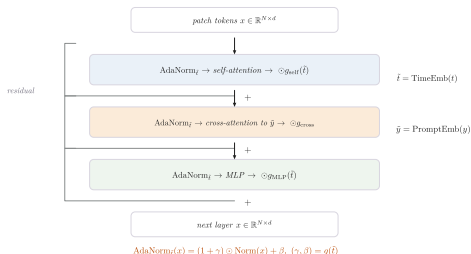
row-wise; multi-head with projections

$W_{Q,K,V}^{(h)}$ and output W_O .

Self-attention: keys from the patches;
cross-attention: keys from the prompt.

Time enters through adaptive normalisation and gates: $(\gamma, \beta) = g(\tilde{t})$.

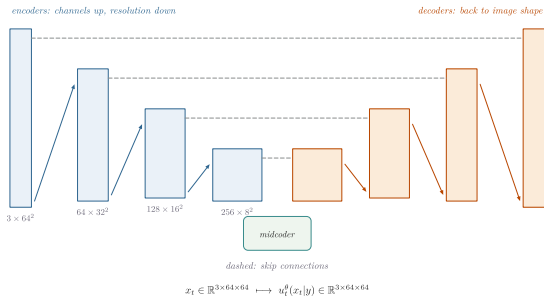
Class-conditional DiTs often drop cross-attention and condition through AdaNorm on t and y .



A convolutional alternative: input and output are images, ideal for $x \mapsto u_t^\theta(x|y)$.

Encoders halve resolution and grow channels, a midcoder processes the latent, decoders return to image shape; skip connections carry detail across.

Used by most early diffusion models [17, 22, 11]; often with attention layers inside.



A 1024×1024 RGB image has $d \approx 3 \cdot 10^6$, and u^θ must output a vector that large. **Compress first.**

Autoencoder: encoder $\mu_\phi : \mathbb{R}^d \rightarrow \mathbb{R}^k$, decoder $\mu_\theta : \mathbb{R}^k \rightarrow \mathbb{R}^d$, $k \ll d$,

$$\mathcal{L}_{\text{Recon}} = \mathbb{E} \|\mu_\theta(\mu_\phi(x)) - x\|^2.$$

Problem: nothing controls p_{latent} ; it may be very hard to learn.

Variational autoencoder: stochastic encoder and decoder (71), $q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \text{diag } \sigma_\phi^2(x))$, $p_\theta(x|z) = \mathcal{N}(\mu_\theta(z), \sigma_\theta^2 I_d)$, and a prior $p_{\text{prior}} = \mathcal{N}(0, I_k)$ that the latents should resemble.

The β -VAE objective (78, 79)

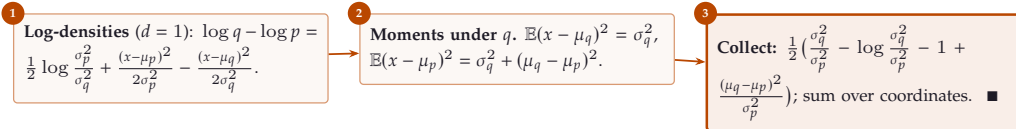
$$\mathcal{L}_{\text{VAE}} = -\mathbb{E}_{x, z \sim q_\phi(\cdot|x)} \log p_\theta(x|z) + \beta \mathbb{E}_x D_{\text{KL}}(q_\phi(\cdot|x) \| p_{\text{prior}}).$$

Example 31: KL between Gaussians

Example 31

$q = \mathcal{N}(\mu_q, \text{diag } \sigma_q^2)$, $p = \mathcal{N}(\mu_p, \text{diag } \sigma_p^2)$:

$$D_{\text{KL}}(q \| p) = \frac{1}{2} \left(K \left(\frac{\sigma_q^2}{\sigma_p^2} \right) + \frac{\|\mu_q - \mu_p\|^2}{\sigma_p^2} \right), \quad K(\alpha) = \sum_i (\alpha_i - \log \alpha_i - 1) \geq 0.$$



With $p = \mathcal{N}(0, I)$: the prior term $\frac{1}{2} (K(\sigma_\phi^2(x)) + \|\mu_\phi(x)\|^2)$ pushes latent means to 0 and variances to 1 (82, 83). **Reparametrisation:** $z = \mu_\phi(x) + \sigma_\phi(x)\epsilon$ moves the randomness out of ϕ , so SGD applies.

Algorithm 6: training a β -VAE, in practice

Algorithm 6 (decoder variance fixed to σ^2)

```
for each mini-batch  $\{x_i\}_{i=1}^B$  do  
   $\epsilon_i \sim \mathcal{N}(0, I_k)$ ,  $z_i = \mu_\phi(x_i) + \sigma_\phi(x_i) \odot \epsilon_i$   
   $\mathcal{L}_{\text{recon}} = \frac{1}{B} \sum_i \frac{1}{2\sigma^2} \|x_i - \mu_\theta(z_i)\|^2$   
   $\mathcal{L}_{\text{KL}} = \frac{1}{B} \sum_i \frac{1}{2} \sum_j (\mu_{ij}^2 + \sigma_{ij}^2 - \log \sigma_{ij}^2 - 1)$   
   $\mathcal{L} = \mathcal{L}_{\text{recon}} + \beta \mathcal{L}_{\text{KL}}$ , update  $(\phi, \theta)$   
end for
```

Practical remarks.

1. β : large β risks **posterior collapse**; KL warm-up from $\beta = 0$. Modern autoencoders use $\beta \ll 1$.
2. Learned decoder variances are delicate; fix σ^2 , making the reconstruction term an MSE.
3. Pixel MSE blurs: add **perceptual** losses.
4. Adversarial terms (VAE-GAN) sharpen, at the cost of stability.

Remark 32: latent diffusion

Recipe. First train the autoencoder. Then train the flow or diffusion model on latents $z \sim q_\phi(z|x)$, $x \sim p_{\text{data}}$. To sample: draw z from the latent model, decode with the mean $x = \mu_\theta(z)$ (sampling the decoder would add noise artefacts).

Why it works. A good autoencoder filters out high-frequency, perceptually irrelevant detail; the generative model spends its capacity on what matters [36]. Nearly all state-of-the-art image and video generators are latent models [36, 48].

The catch. Quality now also depends on how well the autoencoder compresses and decodes.

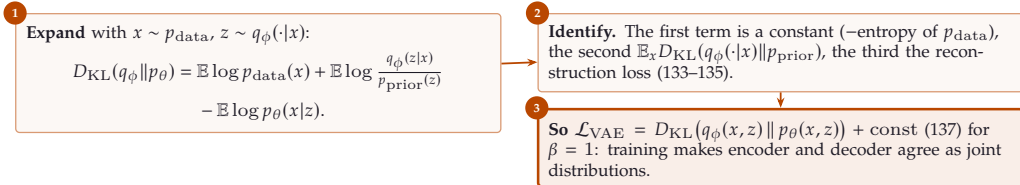
Two models, two jobs.

- the generator transports $\mathcal{N}(0, I)$ to $q_\phi(z) \approx p_{\text{prior}}$;
- the decoder transports $q_\phi(z)$ to p_{data} .

How to split the work is a design choice (Appendix D, below).

Appendix D: the VAE loss is a KL in joint space

Encoder joint $q_\phi(x, z) = p_{\text{data}}(x)q_\phi(z|x)$, decoder joint $p_\theta(x, z) = p_\theta(x|z)p_{\text{prior}}(z)$.



Remark 43 (what is variational). The ideal encoder is the posterior $p_\theta(z|x) = \frac{p_\theta(x|z)p_{\text{prior}}(z)}{p_\theta(x)}$, intractable as $p_\theta(x)$ is unknown; $q_\phi(z|x)$ is its *variational* approximation.

Proposition 3: the chain rule, and the ELBO

Proposition 3 (chain rule)

$D_{\text{KL}}(q(x, z) \| p(x, z)) = D_{\text{KL}}(q(x) \| p(x)) + \mathbb{E}_{x \sim q} D_{\text{KL}}(q(z|x) \| p(z|x));$ hence $D_{\text{KL}}(q(x) \| p(x)) \leq D_{\text{KL}}(q(x, z) \| p(x, z))$ (138).

Proof. $\log \frac{q(x, z)}{p(x, z)} = \log \frac{q(x)}{p(x)} + \log \frac{q(z|x)}{p(z|x)}$; take $\mathbb{E}_{(x, z) \sim q}$. The second term is ≥ 0 by (76). ■

VAEs generate. The x -marginal of q_ϕ is p_{data} , so (139)

$$\mathcal{L}_{\text{VAE}} \geq D_{\text{KL}}(p_{\text{data}} \| p_\theta) + \text{const},$$

$p_\theta(x) = \int p_\theta(x|z) p_{\text{prior}}(z) dz$: an upper bound on the generative error.

ELBO. $\log p_\theta(x) \geq \mathbb{E}_{z \sim q_\phi(\cdot|x)} \log \frac{p_\theta(x|z) p_{\text{prior}}(z)}{q_\phi(z|x)} =: \text{ELBO}(x)$, and

$$\mathcal{L}_{\text{VAE}} = -\mathbb{E}_{x \sim p_{\text{data}}} \text{ELBO}(x; \phi, \theta) + \text{const.} \quad (145)$$

Remark 42. If $q_\phi(x, z) = p_\theta(x, z)$ then $q_\phi(z) = p_{\text{prior}}$ and reconstructions are perfect.

Reconstruction versus generation

Two samplers from one autoencoder: **reconstruction** (encode, decode) and **generation** (latent model, decode).

- small latent: much lost (high rFID) but easy to model;
- large latent: sharp (low rFID) but hard to model: gFID rises.

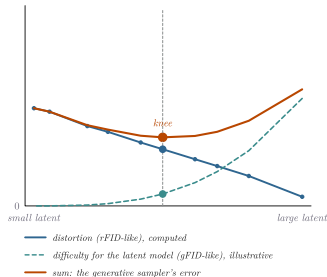
Intuition 44 (division of labour).
The best split sits at the **knee** of the rate–distortion frontier [51, 17, 37].

reconstruction sampler: $x \mapsto z \mapsto \hat{x}$



latent $42 \times 27 \times 3$ ($f = 6$)
decoded: distortion = 0.093

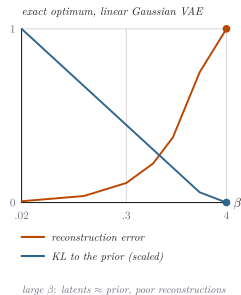
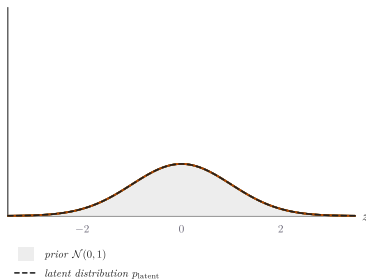
small latents: easy to generate, blurry; large: sharp, hard to model



The Stata Center through a block-average “autoencoder” with downsampling f : distortion is measured, the latent model’s difficulty is a schematic.

The β trade-off, solved exactly

$\beta = 4$: encoders $q_\phi(z|x)$ for $x \in \{-2, \dots, 2\}$



A linear Gaussian VAE on 1D data $x \sim \mathcal{N}(0, 1.6)$, optimised to the exact minimum for each β . Small β : sharp, separated encoders, faithful reconstructions, latents far from the prior. Large β : every $q_\phi(z|x)$ collapses to the prior (posterior collapse). Modern autoencoders use $\beta \ll 1$.

Stable Diffusion 3

- CondOT flow matching (best among the tested alternatives);
- latent space of a pretrained autoencoder;
- MM-DiT: image patches and text tokens attend jointly;
- three text embeddings: CLIP (coarse) and T5-XXL (sequence);
- CFG training; sampling: 50 Euler steps, $w = 2-5$; 8B parameters.

Meta Movie Gen Video

- CondOT flow matching, $\alpha_t = t$, $\beta_t = 1 - t$;
- temporal autoencoder, 8× in time, height and width; temporal tiling for long videos;
- DiT over space-time patches, self- and cross-attention;
- text via UL2, ByT5 (characters) and MetaCLIP; 30B parameters.

7

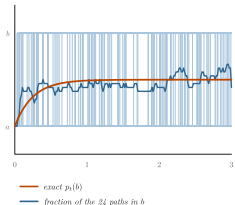
*Discrete diffusion: language
models with diffusion*

State space $\mathcal{S} = \mathcal{V}^d$: sequences of d tokens from a vocabulary $\mathcal{V}$. A Markov process on $\mathcal{S}$ is a **CTMC**; its analogue of a vector field is a **rate matrix**: $Q_t(y|x) \geq 0$ ($y \neq x$), $Q_t(x|x) = -\sum_{y \neq x} Q_t(y|x)$, and $\frac{d}{dh} p_{t+h|t}(y|x)|_{h=0} = Q_t(y|x)$ (87).

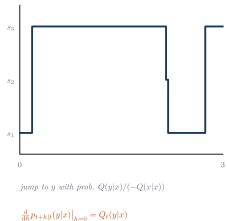
Theorem 33

Every bounded, continuous rate matrix defines a unique CTMC.

$\mathcal{S} = \{a, b\}$, rate $\lambda = 2$: $p_t(b) = \frac{1}{2}(1 - e^{-2\lambda t})$



a path of a 3-state chain: wait $\sim \text{Exp}(-Q(x|x))$,



Example 34 and simulating a CTMC

Example 34. $\mathcal{S} = \{a, b\}$, $Q = \begin{pmatrix} -\lambda & \lambda \\ \lambda & -\lambda \end{pmatrix}$:

$$P(h) = \frac{1}{2} \begin{pmatrix} 1 + e^{-2\lambda h} & 1 - e^{-2\lambda h} \\ 1 - e^{-2\lambda h} & 1 + e^{-2\lambda h} \end{pmatrix} \rightarrow \frac{1}{2} \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}.$$

Check: $P'(0) = \lambda \begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix} = Q$. The memory of the start decays like $e^{-2\lambda h}$.

Euler for CTMCs. By (87),

$p_{t+h|t}(y|x) = \mathbf{1}_{y=x} + hQ_t(y|x) + o(h)$, so

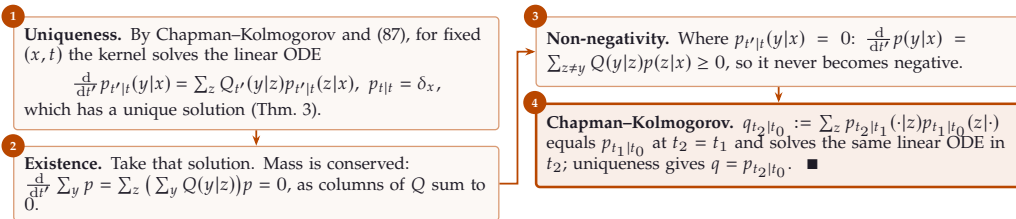
$$X_{t+h} \sim (\mathbf{1}_{y=x} + hQ_t(y|X_t))_{y \in \mathcal{S}} \quad (88)$$

is a valid distribution for small h : a sampleable categorical.

CTMC model. A network returns a column $x \mapsto \{Q_t^\theta(y|x)\}_y$: impossible for $|\mathcal{S}| = V^d$.

Factorised models allow only one-token changes: output $Q_t^\theta(v_i, j|x)$, a $d \times V$ table.

Proof of Theorem 33 (Appendix C)



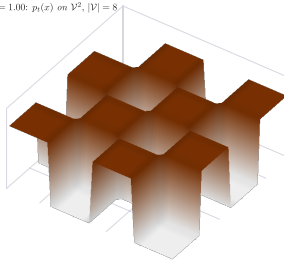
Paths: $p_0(\cdot|z) = p_{\text{init}}, p_1(\cdot|z) = \delta_z$,
 $p_t(x) = \sum_z p_t(x|z)p_{\text{data}}(z)$.

Example 35 (factorised mixture path)

Scheduler κ_t , $\kappa_0 = 0, \kappa_1 = 1$: $p_t(x|z) = \prod_j [(1 - \kappa_t)p_{\text{init}}(x_j) + \kappa_t\delta_{z_j}(x_j)]$: token j is the data token with probability κ_t , else noise.

No mass is transported: one distribution fades in, the other out.

$\kappa_t = 1.00$: $p_t(x)$ on $\mathcal{V}^2$, $|\mathcal{V}| = 8$



each token is the data token with prob. κ_t ,
else drawn from p_{init} :

$$p_t(x|z) = \prod_j [(1 - \kappa_t)p_{\text{init}}(x_j) + \kappa_t\delta_{z_j}(x_j)]$$

no mass moves sideways: one
distribution fades in, the other out

(cross-shaped terms: one token
kept, the other still noise)

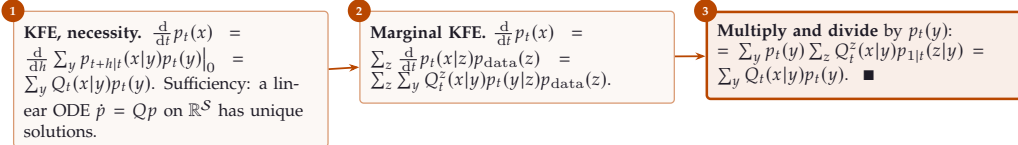
Theorem 36: the discrete marginalisation trick

Theorem 36 and Proposition 2 (Kolmogorov forward equation)

$X_t \sim p_t$ for the CTMC of Q_t iff $\frac{d}{dt} p_t(x) = \sum_y Q_t(x|y) p_t(y)$ (KFE). If each Q_t^z generates $p_t(\cdot|z)$, then

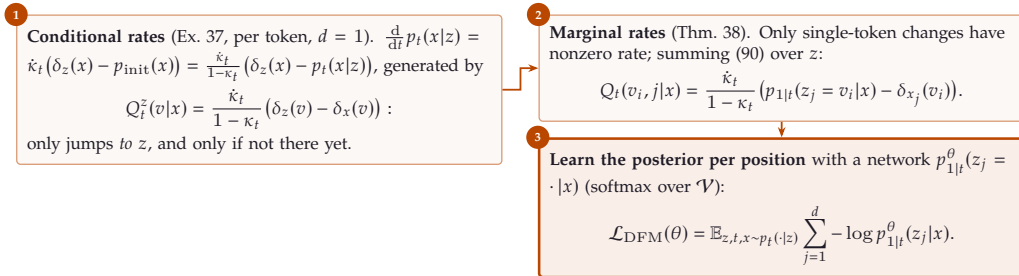
$$Q_t(y|x) = \sum_z Q_t^z(y|x) p_{1|t}(z|x), \quad p_{1|t}(z|x) = \frac{p_t(x|z) p_{\text{data}}(z)}{p_t(x)} \quad (90)$$

generates p_t ; in particular $X_1 \sim p_{\text{data}}$.



The same proof as Theorem 9: the KFE is linear, like the continuity equation.

Example 37 and Theorem 38: training is classification



Continuous flow matching reduced to *regression*; discrete flow matching reduces to *classification*.

Algorithms 7 and 8: discrete diffusion

Algorithm 8 (training)

```
for each iteration do  
   $z \sim p_{\text{data}}, t \sim \text{Unif}[0, 1], \kappa \leftarrow \kappa_t$   
  for  $j = 1..d$ :  $m_j \sim \text{Bern}(\kappa), \xi_j \sim p_{\text{init}}^{(j)}$   
     $x_j \leftarrow m_j z_j + (1 - m_j) \xi_j$   
    logits  $\ell_j = f_{\theta}(x, t)_j, p_{1|t}^{\theta}(\cdot|x)_j = \text{softmax}(\ell_j)$   
     $\mathcal{L} = -\sum_j \log p_{1|t}^{\theta}(z_j|x)_j$ , update  $\theta$   
end for
```

Algorithm 7 (sampling, per-token Euler)

```
 $h = 1/n, X_0 \sim p_{\text{init}}$   
for  $i = 1..n$ , for each token  $j$  in parallel:  
  rates  $Q_t^{\theta}(v, j|X_t) = \frac{\kappa_t}{1-\kappa_t} (p_{1|t}^{\theta}(v|X_t)_j - \delta_{X_t^j}(v))$   
   $X_{t+h}^j \sim (\delta_{X_t^j}(v) + h Q_t^{\theta}(v, j|X_t))_v$   
   $t \leftarrow t + h$   
return  $X_1$ 
```

Per-token updates match the full Euler step to first order.

Example 39: masked diffusion language models

t with $\kappa_t = 1.0$: 10 of 10 tokens revealed

a corgi runs down a snowy hill under the mountains

training: a classifier per position,

$$\mathcal{L} = \mathbb{E} \sum_j -\log p_{1|t}^\theta(z_j|x)$$

sampling: each masked token unmask

at rate $\frac{\kappa_t}{1-\kappa_t}$, to a draw from $p_{1|t}^\theta$

Add a token [mask] and start from the all-masked sequence, $p_{\text{init}} = \delta_{[\text{mask}]^d}$. Each masked token unmask at rate $\frac{\dot{\kappa}_t}{1-\kappa_t}$ to a draw from the learned per-position posterior, in any order and many at a time, unlike left-to-right autoregression.

Remark 40: generator matching

Why did the recipe carry over so seamlessly? It is not about flows or CTMCs: it is a general principle for building generative models from **Markov processes** [19].

1. Choose a conditional path $p_t(\cdot|z)$ interpolating noise and δ_z .
2. Find a conditional **generator** producing it: a vector field, a diffusion, a rate matrix, a jump kernel.
3. Marginalise with the posterior; the forward equation is linear, so the marginal generator produces p_t .
4. Learn it with a conditional loss whose minimiser is the marginal.

Any state space. Manifolds [8, 10], mixed text–image spaces [6], jump processes [19, 7]: the same four steps.

Current discrete diffusion language models follow exactly this recipe at web scale [4].

Discrete vs. continuous time. Early models [41, 42, 17] are discrete-time Markov chains trained on an ELBO (a lower bound). Song et al. [45]: they approximate SDEs; in continuous time the bound is tight (Thms. 12, 22 are equalities). Same losses in the end.

Forward processes vs. paths. Diffusion papers noise data with an SDE $d\bar{X}_t = a_t \bar{X}_t dt + \bar{\sigma}_t d\bar{W}_t$, which gives Gaussian paths with $\alpha_t = e^{\int_0^t a_r dr}$, $\beta_t^2 = \alpha_t^2 \int_0^t \bar{\sigma}_r^2 / \alpha_r^2 dr$, but only reaches noise as $t \rightarrow \infty$, in inverted time.

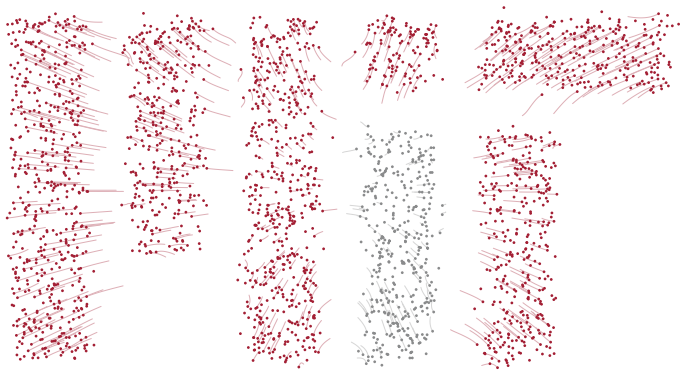
Time reversal vs. Fokker–Planck. The original sampler is the time reversal [2]

$$dX_t = \left[-u_t(X_t) + \sigma_t^2 \nabla \log p_t(X_t) \right] dt + \sigma_t dW_t,$$

one instance of Thm. 17. Only X_1 matters, so any σ_t works; the probability-flow ODE is often better [23, 28].

Flow matching [25] and stochastic interpolants [1]. FM showed flows alone train at scale; SIs include both the flow and the SDE extension used here.

- Generation is sampling; we transport $\mathcal{N}(0, I_d)$ to p_{data} with an ODE or SDE.
- A **probability path** interpolates noise and data; per data point it is explicit.
- The **marginalisation trick**: averaging conditional fields with posterior weights generates the marginal path, because the continuity equation is linear.
- **Conditional flow matching** learns that average by regression, simulation-free.
- Scores, denoisers and noise predictors are reparametrisations; with the score any σ_t gives an exact SDE sampler (Fokker–Planck).
- **Classifier-free guidance**:
 $(1 - w)u(\cdot|\emptyset) + wu(\cdot|y)$.
- At scale: Fourier time features, DiTs or U-Nets, latent VAEs.
- In discrete space the same recipe with rate matrices; training becomes classification.



$t = 1.00$

Thank you.
Johannes Herter