

Flow Matching for 2D Data Generation

From Intuition to Implementation

Arthur Courselle, Baptiste Villeneuve, Eugénie Beauvillain
Flavien Geoffray, Lucas Duport, Lucas Juanico

SCIA 2026

January 23, 2026

Outline

- 1 Introduction & Baseline
- 2 Flows Normalisants Continus
- 3 Théorie du Flow Matching
- 4 Implémentation: Optimal Transport & Sampling
- 5 Implémentation: Code & Architecture
- 6 Résultats & Comparaison

Introduction: Flow Matching Context

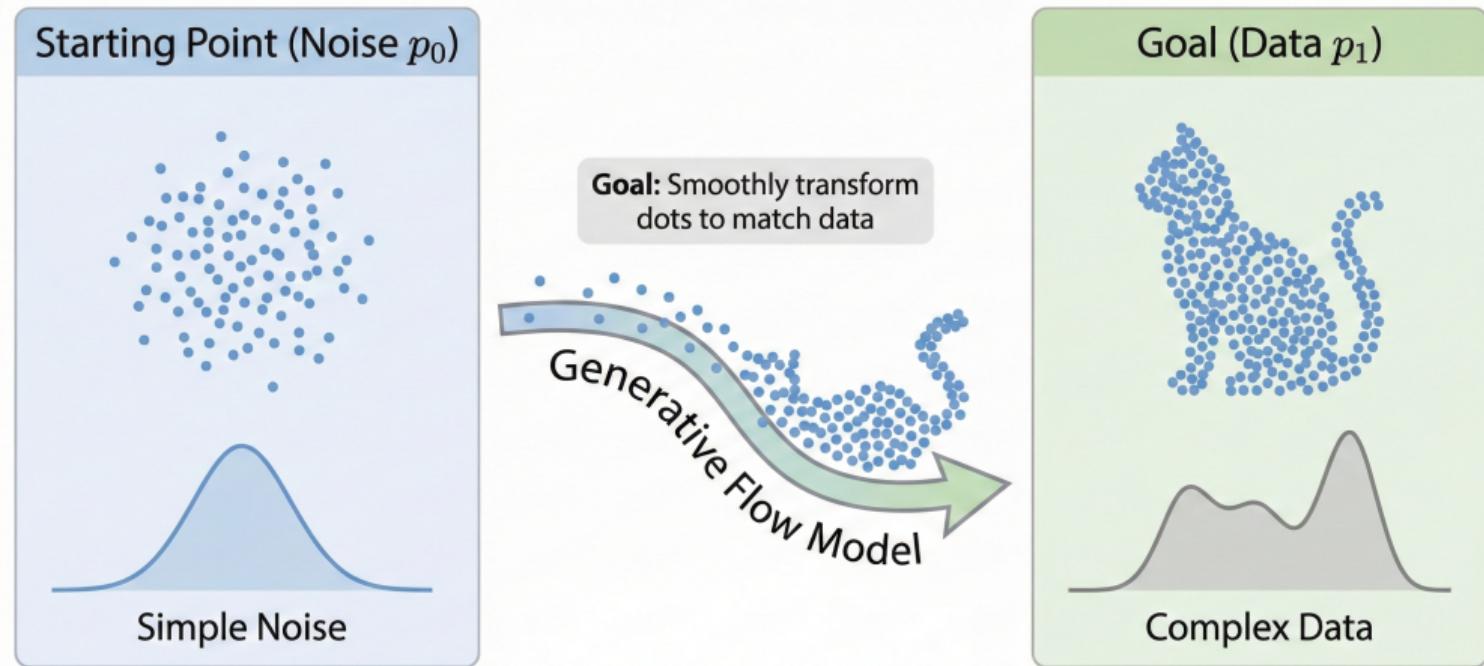


Figure 1: Flow Matching simply means learning a smooth path from simple noise to complex data.

The Transport Problem

Objective: Generative Modeling

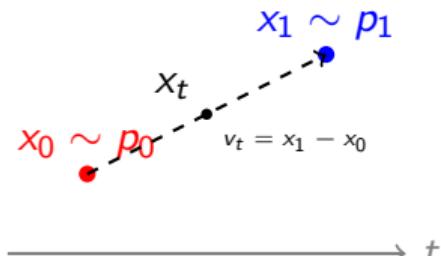
Let p_0 be a source distribution (Gaussian noise) and p_1 an unknown target distribution (data). We seek a **diffeomorphism** $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^d$ such that:

$$\phi_* p_0 = p_1 \quad (\text{Push-forward})$$

Our Approach: Optimal Transport Path

- We define the probability path p_t via linear interpolation.
- Stochastic Process:

$$\mathbf{x}_t = (1 - t)\mathbf{x}_0 + t\mathbf{x}_1$$



Baseline: Normalizing Flows and the Jacobian Bottleneck

The discrete flow of RealNVP is a finished series of steps. The model is defined as an integer number of layers :

$$x_0 \rightarrow x_1 \rightarrow x_2 \rightarrow \dots \rightarrow x_k$$

$$z = f_k(\dots f_2(f_1(x)))$$

Let $f_\theta : \mathcal{X} \rightarrow \mathcal{Z}$ be an invertible generative model. According to the **Change of Variable Theorem**, the log-likelihood is:

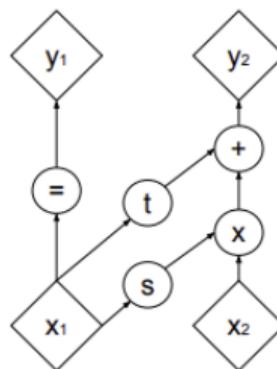
$$\log p_X(x) = \log p_Z(f_\theta(x)) + \log \left| \det \left(\frac{\partial f_\theta(x)}{\partial x} \right) \right|$$

RealNVP: The Affine Coupling Trick

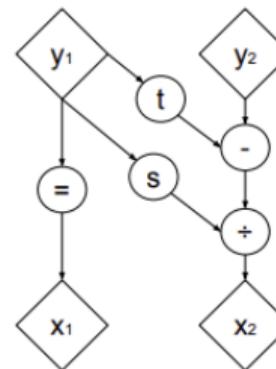
We split the data into two parts (x_a, x_b) .

The Strategy:

$$\begin{cases} y_a = x_a & \text{(Identity)} \\ y_b = x_b \cdot \exp(s(x_a)) + t(x_a) & \text{(Affine Transform)} \end{cases}$$

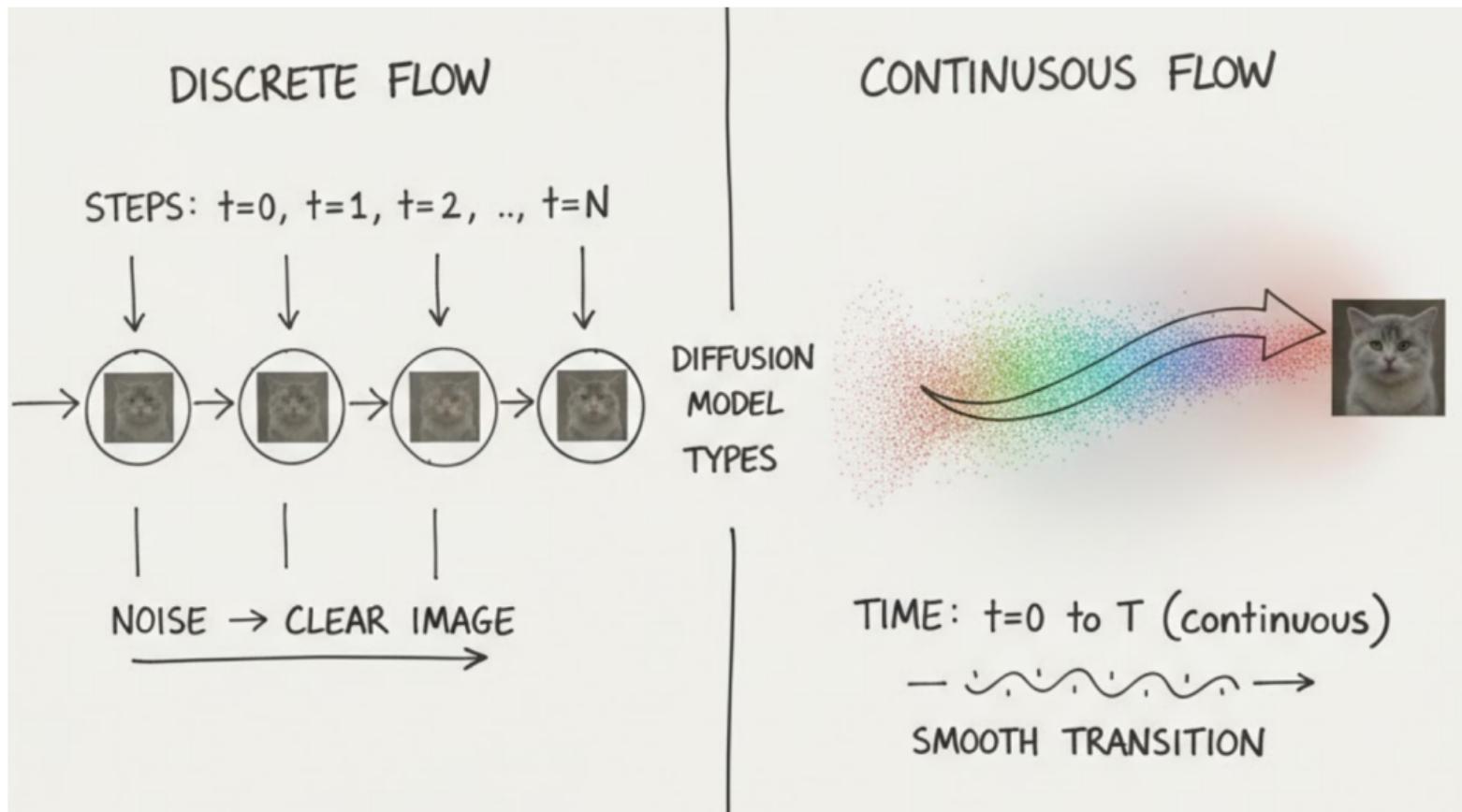


(a) Forward propagation



(b) Inverse propagation

Flow Discret vs Flow Continu



The Probability Flow ODE

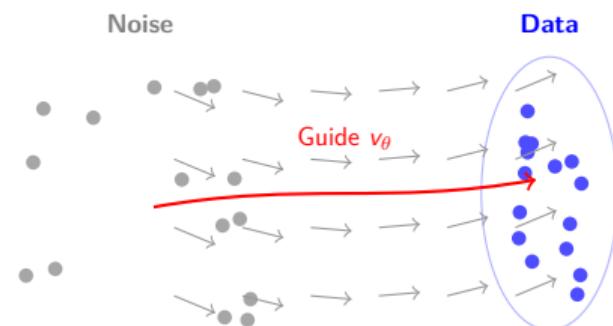
Ordinary Differential Equation (ODE)

$$\frac{d}{dt} \phi_t(x) = v_t(\phi_t(x))$$

- $\phi_t(x)$: The probability path (flow) from noise to data.
- $v_t(\cdot)$: The velocity field defined at every point (x, t) .

Flow Matching Intuition

- Learning a **conditional velocity field**
 $v_t(x)$.
- **Objective:** Push noise towards the data structure.
- **Architectural Freedom:** The network v_θ no longer needs to be invertible!
 - We can use a simple MLP or a U-Net for example.



Flow Matching

Let $x_1 \sim q(x_1)$ from which we only have access to data samples.

Let p_t be a probability path such that $p_0 = p$ is the standard normal distribution $p(x) = \mathcal{N}(x|0, I)$, and $p_1(x) \approx q(x)$.

Flow Matching (FM) objective

$$\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{t \sim \mathcal{U}[0,1], x \sim p_t(x)} \|v_t(x) - u_t(x)\|^2 \quad (1)$$

θ : the learnable parameters of v_t .

Conditional Flow Matching

Let $p_t(x|x_1)$ such that $p_0(x|x_1) = p(x)$ and $p_1(x|x_1) = \mathcal{N}(x|x_1, \sigma^2 I)$.

The marginal probability $p_t(x)$ is given by
$$p_t(x) = \int p_t(x|x_1)q(x_1)dx_1.$$

To find $u_t(x)$, we derive $p_t(x)$ with respect to t and apply the continuity equation $(\frac{\partial p_t(x)}{\partial t} + \nabla \cdot [u_t(x)p_t(x)] = 0)$:

$$\frac{\partial p_t(x)}{\partial t} = \int \frac{\partial p_t(x|x_1)}{\partial t} q(x_1) dx_1 \quad (2)$$

$$= -\nabla \cdot \left[\int u_t(x|x_1) p_t(x|x_1) q(x_1) dx_1 \right] \quad (3)$$

$$u_t(x)p_t(x) = \int u_t(x|x_1) p_t(x|x_1) q(x_1) dx_1 \quad (4)$$

$$u_t(x) = \int u_t(x|x_1) \frac{p_t(x|x_1) q(x_1)}{p_t(x)} dx_1 \quad (5)$$

Conditional Flow Matching

$$\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{t \sim \mathcal{U}[0,1], x \sim p_t(x)} \|v_t(x) - u_t(x)\|^2 \quad (6)$$

$$= \mathbb{E}_{t \sim \mathcal{U}[0,1], x \sim p_t(x)} [\|v_t(x)\|^2 - 2 \cdot v_t(x) \cdot u_t(x) + \|u_t(x)\|^2] \quad (7)$$

(8)

We focus on the term $\mathbb{E}_{t \sim \mathcal{U}[0,1], x \sim p_t(x)} [\|2 \cdot v_t(x) \cdot u_t(x)\|]$:

$$\mathbb{E}_{t \sim \mathcal{U}[0,1], x \sim p_t(x)} [\|2 \cdot v_t(x) \cdot u_t(x)\|] = 2 \int v_t(x) \cdot u_t(x) \cdot p_t(x) dx \quad (9)$$

$$= 2 \int v_t(x) \cdot \frac{\int u_t(x|x_1) p_t(x|x_1) q(x_1) dx_1}{p_t(x)} \cdot p_t(x) dx \quad (10)$$

$$= 2 \int \int v_t(x) \cdot u_t(x|x_1) p_t(x|x_1) q(x_1) dx_1 dx \quad (11)$$

$$= 2 \cdot \mathbb{E}_{x \sim p_t(x|x_1), x_1 \sim q(x_1)} [v_t(x) \cdot u_t(x|x_1)] \quad (12)$$

Conditional Flow Matching

$$\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{t \sim \mathcal{U}[0,1], x \sim p_t(x)} [\|v_t(x)\|^2 - 2 \cdot v_t(x) \cdot u_t(x) + \|u_t(x)\|^2] \quad (14)$$

$$= \mathbb{E}_{x \sim p_t(x|x_1), x_1 \sim q(x_1)} [\|v_t(x)\|^2 - 2 \cdot v_t(x) \cdot u_t(x|x_1) + \|u_t(x|x_1)\|^2 + \|u_t(x)\|^2 - \|u_t(x|x_1)\|^2] \quad (15)$$

$$= \mathbb{E}_{x \sim p_t(x|x_1), x_1 \sim q(x_1)} [\|v_t(x) - u_t(x|x_1)\|^2 + \|u_t(x)\|^2 - \|u_t(x|x_1)\|^2] \quad (16)$$

(17)

Since $u_t()$ does not have any impact on the weights, we can rewrite the FM objective as :

Conditional Flow Matching (CFM) objective

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{x \sim p_t(x|x_1), x_1 \sim q(x_1)} [\|v_t(x) - u_t(x|x_1)\|^2] \quad (18)$$

Optimal Transport

We consider the flow : $\psi_t(x_0) = \sigma_t(x_1)x_0 + \mu_t(x_1)$ with $u_t(x_1) = tx_1$ and $\sigma_t(x_1) = 1 - t$.

We differentiate :

$$\frac{\partial}{\partial t} \psi_t(x_0) = \frac{\partial}{\partial t} ((1-t)x_0 + tx_1) \quad (19)$$

$$= \frac{\partial}{\partial t} (x_0 - tx_0 + tx_1) \quad (20)$$

$$= \boxed{x_1 - x_0} \quad (21)$$

$$(22)$$

Optimal Transport

We recall : $\frac{\partial}{\partial t} \psi_t(x_0) = u(\psi_t(x_0)|x_1)$

We can now rewrite the objective as :

Optimal Transport objective

$$\mathcal{L}_{\text{OT}}(\theta) = \mathbb{E}_{x_0 \sim p_0(x_0), x_1 \sim q(x_1)} \left[\|v_t(\psi_t(x_0)) - (x_1 - x_0)\|^2 \right] \quad (23)$$

We then just need to train a neural network $v_\theta(x, t)$ to approximate the target vector field.

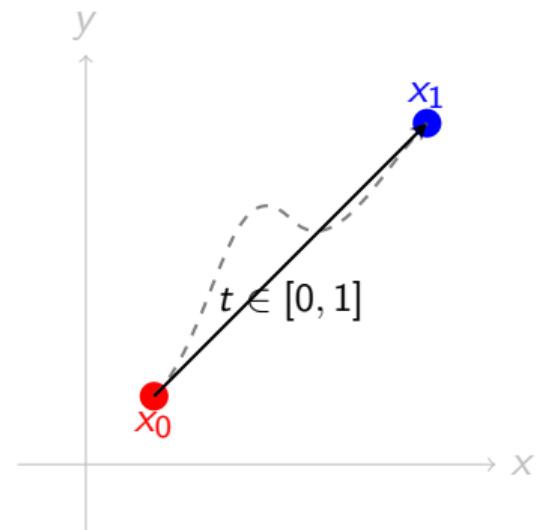
Defining the Probability Path

The Goal: We need to define a path for a particle to move from Noise (x_0) to Data (x_1).

Optimal Transport We choose the simplest geometric path: the **straight line**.

Interpolation Formula

$$x_t = (1 - t)x_0 + tx_1$$



The Regression Target

The neural network needs to learn the **velocity field** $v_\theta(x, t)$ that generates this path.

Since the path is a straight line, the velocity is the time derivative:

$$u_t(x|x_1) = \frac{d}{dt}x_t = \frac{d}{dt}((1-t)x_0 + tx_1)$$

The Key Advantage

The target velocity is **constant**:

$$u_t = x_1 - x_0$$

⇒ The network learns a stable, constant direction for each pair, which is easier than learning a curved diffusion path.

Sampling with Euler Method

Inference: Starting from noise x_0 , we follow the learned velocity field.

We use the **Euler Integration**:

$$x_{t+dt} = x_t + v_\theta(x_t, t) \cdot dt$$

Why it works well?

- The learned trajectories are nearly straight (Optimal Transport).
- Low discretization error.
- Fast generation ($N = 10$ to 20 steps).

Algorithm: Sampling

- ① $x \leftarrow \text{SampleNoise}()$
- ② $dt \leftarrow 1/N$
- ③ **For** $i = 0$ to $N - 1$:
 - $v \leftarrow \text{Model}(x, t)$
 - $x \leftarrow x + v \cdot dt$
 - $t \leftarrow t + dt$
- ④ **Return** x

Flow Matching Training Loop

Algorithm

- ① Sample a minibatch $x_1 \sim p_{\text{data}}$
- ② Sample noise $x_0 \sim \mathcal{N}(0, I)$ and time $t \sim \mathcal{U}[0, 1]$
- ③ Interpolate:

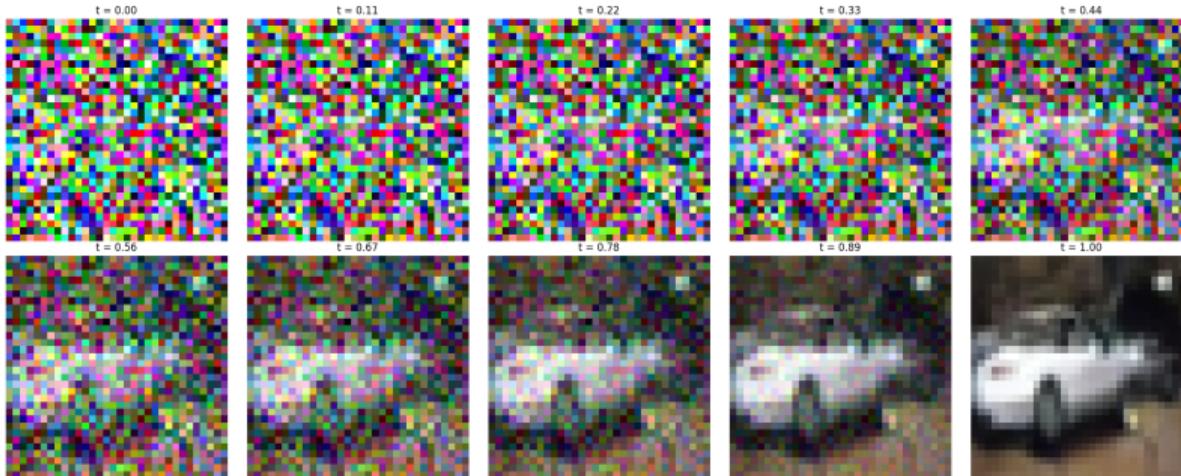
$$x_t = (1 - t)x_0 + tx_1$$

- ④ Compute target vector field:

$$u_t = x_1 - x_0$$

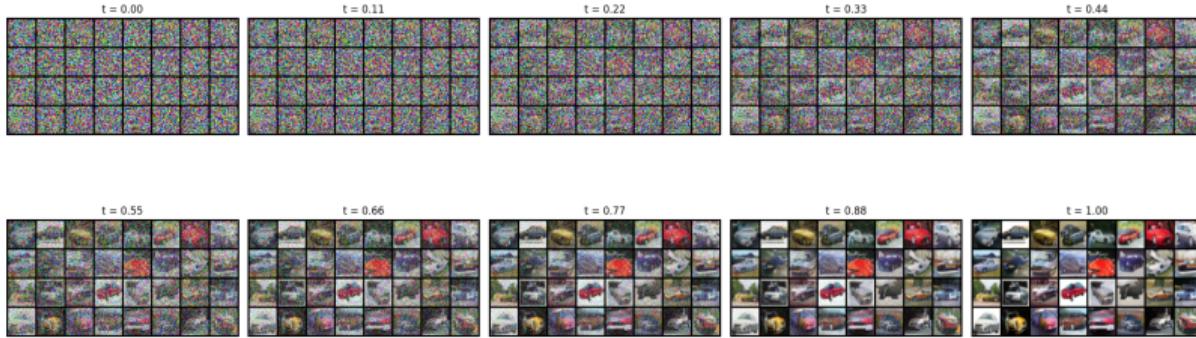
- ⑤ Predict vector field $\hat{u}_t = v_\theta(x_t, t)$
- ⑥ Calculate $MSE(\hat{u}_t, u_t)$
- ⑦ Backpropagate and optimize model

Qualitative Results: CIFAR-10

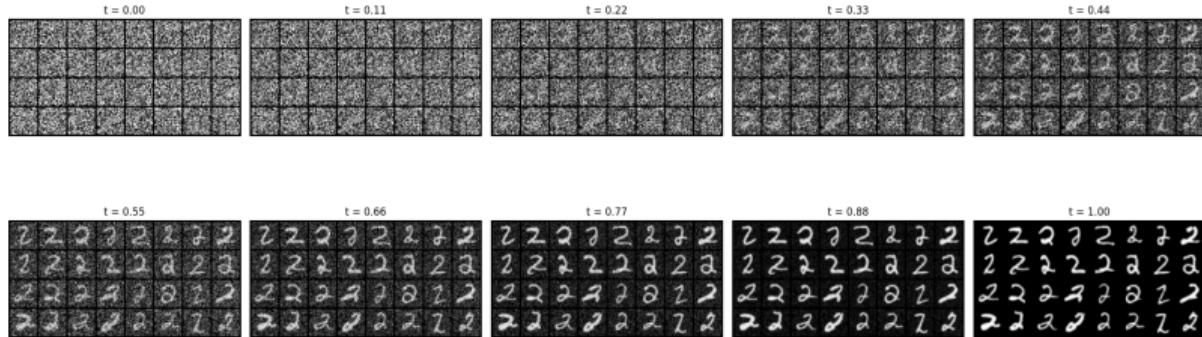


- Scaling to high-dimensional data using a **U-Net** backbone.
- The model generates sharp samples by following the learned velocity field.

Qualitative Results



Qualitative Results



Comparison

RealNVP (Baseline)

- **Inference:** Very Fast ($O(K)$ layers)
- **Training:** More Complex
 - NLL
 - Invertibility

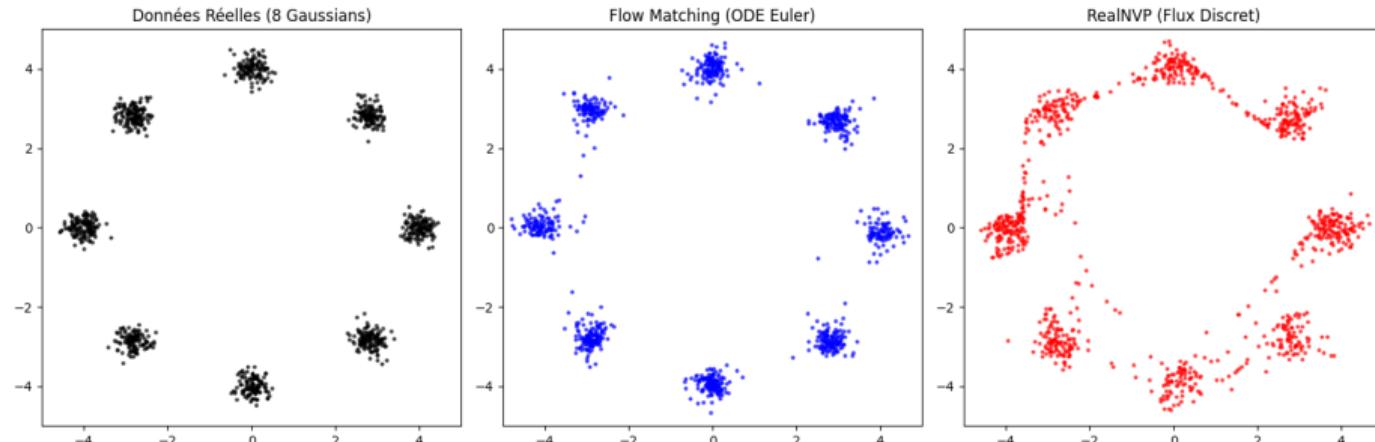
Flow Matching (Our Approach)

- **Inference:** Slower (ODE Solver)
- **Training:** Simple & Stable
 - Faster Convergence

Performance Summary

| Criteria | RealNVP | Flow Matching |
|-----------------------|---------------|------------------|
| Inference Speed | Instantaneous | Slow |
| Training Complexity | High | Low (Regression) |
| Convergence Stability | Less stable | Very stable |

Table: Summary table of advantages and disadvantages.



Thank You!

Code available at: <https://github.com/ArthurCourselle/FlowMatching>