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

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 a the standard normal distribution $p(x) = \mathcal{N}(x|0, I)$, and $p_1(x) \approx q(x)$.

Flow Matching (FM) objective

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

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 $\boxed{\mathrm{p}_t(x) = \int p_t(x|x_1)q(x_1)dx_1}$.

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

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

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

$$u_t(x)p_t(x) = \int u_t(x|x_1)p_t(x|x_1)q(x_1)\, dx_1 \tag{4}$$

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

Conditional Flow Matching

$$
\begin{aligned}
\mathcal{L}_{\text{FM}}(\theta) &= \mathbb{E}_{t\sim\mathcal{U}[0,1],x\sim p_t(x)}\|v_t(x) - u_t(x)\|^2 && (6)\\
&= \mathbb{E}_{t\sim\mathcal{U}[0,1],x\sim p_t(x)}\left[\|v_t(x)\|^2 - 2.v_t(x).u_t(x) + \|u_t(x)\|^2\right] && (7)\\
& && (8)
\end{aligned}
$$

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

$$
\begin{aligned}
\mathbb{E}_{t\sim\mathcal{U}[0,1],x\sim p_t(x)}\left[\|2.v_t(x).u_t(x)\right] &= 2\int v_t(x).u_t(x).p_t(x)dx && (9)\\
&= 2\int v_t(x).\frac{\int u_t(x|x_1)p_t(x|x_1)q(x_1)dx_1}{p_t(x)}.p_t(x)dx && (10)\\
&= 2\int\int v_t(x).u_t(x|x_1)p_t(x|x_1)q(x_1)dx_1dx && (11)\\
&= 2.\mathbb{E}_{x\sim p_t(x|x_1),x_1\sim q(x_1)}\left[\ v_t(x).u_t(x|x_1)\right] && (12)\\
& && (13)
\end{aligned}
$$

Conditional Flow Matching

$$
\begin{aligned}
\mathcal{L}_{\text{FM}}(\theta) &= \mathbb{E}_{t\sim\mathcal{U}[0,1],x\sim p_t(x)}\left[\|v_t(x)\|^2 - 2.v_t(x).u_t(x) + \|u_t(x)\|^2\right] && (14)\\
&= \mathbb{E}_{x\sim p_t(x|x_1),x_1\sim q(x_1)}\left[\|v_t(x)\|^2 - 2.v_t(x).u_t(x|x_1) + \|u_t(x|x_1)\|^2 + \|u_t(x)\|^2 - \|u_t(x|x_1)\|^2\right] && (15)\\
&= \mathbb{E}_{x\sim p_t(x|x_1),x_1\sim q(x_1)}\left[\|v_t(x) - u_t(x|x_1)\|^2 + \|u_t(x)\|^2 - \|u_t(x|x_1)\|^2\right] && (16)\\
& && (17)
\end{aligned}
$$

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

Conditional Flow Matching (CFM) objective

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

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 :

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

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

$$= \boxed{x_1 - x_0} \tag{21}$$

$$\tag{22}$$

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

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

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

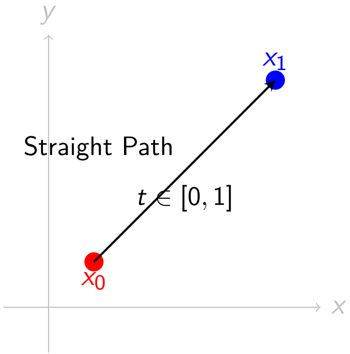
Defining the Probability Path

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:

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

The Key Advantage

The target velocity is **constant**:

$$u_t = x_1 - x_0$$

$\Rightarrow$ 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

Thank You!

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