Flow Matching: A Step-by-Step Tutorial

Please read the DDIM tutorial before continuing... background and notation follow from it.

1 Where We Are Coming From

Suppose our data are drawn from some unknown target distribution

$$x_0 \sim p.$$

For images, x_0 could be an image. For atomistic generation, it could represent an atomic configuration. We do not generally know an analytic expression for p, but we have samples from it in the form of a training dataset.

Our goal is to construct a generative model that allows us to produce new samples from p.

In DDPM, the basic strategy was to define a forward corruption process

$$\text{data} \longrightarrow \text{progressively noisier data} \longrightarrow \text{Gaussian noise},$$

and then learn how to reverse this process:

$$\text{Gaussian noise} \longrightarrow \text{data}.$$

DDIM showed us that the generative trajectory does not actually have to be stochastic. Instead, we can construct a deterministic trajectory that transports samples from the noise distribution into the data distribution.

This observation brings us very close to flow matching.

2 Forget Diffusion for a Moment

Flow matching asks us to temporarily set aside the Gaussian forward process, reverse diffusion, and the specific probabilistic structure of DDPM.

Suppose instead that we simply have two distributions:

$$q = \text{easy source distribution}$$

and

$$p = \text{target/data distribution}.$$

A common example is

$$q = \mathcal{N}(0, I).$$

We know how to sample

$$x_1 \sim q,$$

and our dataset gives us samples

$$x_0 \sim p.$$

The time convention used here matches the diffusion tutorial:

$$t = 1$$

corresponds to the source distribution, while

$$t = 0$$

corresponds to the target distribution.

Our generative-modeling problem can therefore be written as

$$q \longrightarrow p.$$

The central question is:

> Can we learn a rule for continuously moving particles distributed according to q so that, by the end of the motion, they are distributed according to p?

3 Start with a Single Pair of Points

Before transporting entire distributions, consider two points:

$$x_1 \qquad \text{and} \qquad x_0.$$

We want to construct a path that begins at x_1 when $t = 1$ and ends at x_0 when $t = 0$.

There are infinitely many possible paths. The simplest possible choice is a straight line:

$$\boxed{x_t = tx_1 + (1-t)x_0} \qquad t \in [0, 1].$$

Checking the endpoints,
at $t = 1$,

$$x_t = x_1,$$

and at $t = 0$,

$$x_t = x_0.$$

At $t = 1/2$,

$$x_t = \frac{1}{2}x_1 + \frac{1}{2}x_0,$$

so the particle is halfway between the two points.

This is a simple example of a *pointwise flow*.

4 A Numerical Example

Consider a one-dimensional example:

$$x_1 = 8, \qquad x_0 = 2.$$

Then

$$x_t = 8t + 2(1 - t) = 2 + 6t.$$

The path is

t	x_t
1.00	8
0.75	6.5
0.50	5
0.25	3.5
0.00	2

Because generation proceeds from $t = 1$ toward $t = 0$, the particle moves as

$$8 \longrightarrow 6.5 \longrightarrow 5 \longrightarrow 3.5 \longrightarrow 2.$$

At this stage there is nothing probabilistic about the construction. We have simply defined a path between two known points.

5 Describing the Path Using Velocity

Instead of specifying the whole trajectory x_t, we can specify the velocity field that generates it.

Starting with

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

differentiate with respect to t:

$$\frac{dx_t}{dt} = x_1 - x_0.$$

The convention used in the tutorial defines the flow ODE as

$$\boxed{\frac{dx_t}{dt} = -v_t(x_t)}.$$

Therefore,

$$-v_t(x_t) = x_1 - x_0,$$

or equivalently,

$$\boxed{v_t^{[x_1, x_0]}(x_t) = x_0 - x_1.}$$

For the numerical example,

$$v = 2 - 8 = -6.$$

This velocity is constant because the pointwise path is a straight line traversed at constant speed.

6 What Is a Flow?

A flow is a time-dependent vector field

$$v = \{v_t\}_{t\in[0,1]}.$$

The best mental model is a velocity field in a fluid or gas.

At every time t and location x, the vector

$$v_t(x)$$

tells a particle which direction to move.

Once a starting point and a velocity field are known, the particle trajectory is determined by the ODE

$$\frac{dx_t}{dt} = -v_t(x_t).$$

A discrete approximation is

$$x_{t-\Delta t} \approx x_t + v_t(x_t)\Delta t.$$

Thus,

$$\boxed{\text{velocity field} + \text{starting point} \Longrightarrow \text{trajectory.}}$$

7 From Moving One Point to Moving a Distribution

Suppose we draw many particles independently from

$$x_1^{(i)} \sim q.$$

Now apply the same velocity field $v_t(x)$ to every particle.

Each individual particle follows a trajectory, but collectively the entire distribution of particles changes over time.

Let this intermediate distribution be denoted by

$$p_t.$$

Initially,

$$p_1 = q.$$

We want the flow to satisfy

$$p_0 = p.$$

Therefore the desired transport is

$$\boxed{q \xrightarrow{\text{flow}} p.}$$

If we knew such a velocity field, generation would be straightforward:

1. Sample
$$x_1 \sim q.$$
2. Integrate the flow ODE from $t = 1$ to $t = 0$.
3. Return the resulting x_0.

If the learned flow is correct, these final points should be distributed according to the target distribution p.

8 How Should Source Points Be Paired with Target Points?

Suppose the dataset contains target points

$$x_0^{(1)}, x_0^{(2)}, \ldots$$

and the source distribution gives us samples

$$x_1^{(1)}, x_1^{(2)}, \ldots.$$

We need some rule for deciding which source point corresponds to which target point. This is described by a *coupling*

$$\Pi_{q,p}.$$

A coupling is a joint distribution over pairs

$$(x_1, x_0)$$

whose marginals are

$$x_1 \sim q$$

and

$$x_0 \sim p.$$

The simplest possible choice is the *independent coupling*:

$$\boxed{x_1 \sim q, \qquad x_0 \sim p, \qquad x_1 \perp x_0.}$$

That is, we sample one point from the source distribution and one point from the target distribution independently, and declare them to be a pair for the purpose of constructing a training trajectory.

9 Constructing Pointwise Training Trajectories

For every sampled pair

$$(x_1, x_0),$$

we can define the linear path

$$\boxed{x_t = tx_1 + (1-t)x_0}$$

with pointwise velocity

$$\boxed{v_t^{[x_1,x_0]} = x_0 - x_1.}$$

This means we can manufacture supervised training data. Sample

$$x_0 \sim p,$$

sample

$$x_1 \sim q,$$

and sample

$$t \sim \text{Uniform}(0,1).$$

Then compute

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

and

$$v_{\text{target}} = x_0 - x_1.$$

This gives a supervised-learning example

$$(x_t, t) \longrightarrow v_{\text{target}}.$$

At first glance, this seems to solve the learning problem. However, there is an important complication.

10 Different Pointwise Trajectories Can Disagree

Suppose two independently sampled source-target pairs produce two different straight-line trajectories.

Both trajectories may pass through the same intermediate location x_t at the same time t, while having different velocities:

$$v_t^A(x_t) \neq v_t^B(x_t).$$

Our neural network receives only

$$(x_t, t).$$

It does not know which original pair

$$(x_1, x_0)$$

generated that point.

Therefore it cannot output both pointwise velocities.

The model must learn one effective velocity for that location and time.

This motivates the *marginal velocity field*.

11 The Marginal Velocity Field

Given that a particle is observed at location x_t at time t, consider all possible source-target pairs that could have produced it.

Each such pair implies a pointwise velocity

$$v_t^{[x_1,x_0]}(x_t).$$

The effective velocity is their conditional expectation:

$$\boxed{v_t^\star(x_t) = \mathbb{E}\left[v_t^{[x_1,x_0]}(x_t) \mid x_t\right].}$$

For linear pointwise flows,

$$v_t^{[x_1,x_0]} = x_0 - x_1,$$

so

$$\boxed{v_t^\star(x_t) = \mathbb{E}\left[x_0 - x_1 \mid x_t\right].}$$

This is the velocity field we ultimately want the neural network to learn.

12 Why Averaging Velocities Makes Sense

Suppose that near some point x_t,

- 70% of possible trajectories have velocity $+2$,
- 30% have velocity -1.

Then the average velocity is

$$0.7(2) + 0.3(-1) = 1.1.$$

Although individual particles may follow different trajectories, the distribution as a whole has a net local transport velocity.

This is the distinction between

$$\boxed{\text{pointwise flow} \neq \text{marginal flow.}}$$

A pointwise flow describes how one specifically paired source point moves to one specifically paired target point.

The marginal flow describes the effective evolution of the probability distribution as a whole.

13 The Apparent Training Problem

The quantity we want is

$$v_t^\star(x_t) = \mathbb{E}\left[v_t^{[x_1,x_0]} \mid x_t\right].$$

But directly computing this expectation would require evaluating the conditional distribution over all source-target pairs compatible with x_t.

For linear flows, this means reasoning about quantities like

$$p(x_0, x_1 \mid x_t),$$

which is generally difficult.

Fortunately, flow matching avoids calculating this conditional expectation explicitly.

14 The Regression Trick

We use the standard squared-error regression identity.

Suppose we train a function $f(x)$ to predict a random variable y using

$$\mathcal{L} = \mathbb{E}\left[\|f(x) - y\|^2\right].$$

The optimal predictor is

$$\boxed{f^\star(x) = \mathbb{E}[y \mid x].}$$

Flow matching applies this identity directly.

Train a neural network

$$v_\theta(x_t, t)$$

with the loss

$$\boxed{\mathcal{L}(\theta) = \mathbb{E}\left[\left\|v_\theta(x_t,t) - v_t^{[x_1,x_0]}(x_t)\right\|^2\right].}$$

The optimal network satisfies

$$v_\theta^\star(x_t, t) = \mathbb{E}\left[v_t^{[x_1,x_0]}(x_t) \mid x_t\right].$$

But this is exactly the marginal velocity field:

$$\boxed{v_\theta^\star(x_t, t) = v_t^\star(x_t).}$$

Therefore ordinary L_2 regression performs the conditional averaging for us.

15 Why This Step Is So Important

We never explicitly compute

$$\mathbb{E}\left[v_t^{[x_1,x_0]} \mid x_t\right].$$

Instead, during training we repeatedly show the network individual pointwise velocities.

If multiple trajectories pass through similar (x_t, t) values with different velocities, the network cannot distinguish which specific trajectory produced each sample.

Under squared-error regression, the optimal output is therefore the conditional mean.

So the network implicitly turns many pointwise flows into one marginal flow.

16 Linear Flow Matching Training

For the linear pointwise flow,

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

and

$$v_t^{[x_1,x_0]} = x_0 - x_1.$$

One training iteration is therefore:

1. Sample a target point
$$x_0 \sim p.$$
2. Sample a source point
$$x_1 \sim q.$$
3. Sample a time
$$t \sim \text{Uniform}(0, 1).$$
4. Construct the intermediate point
$$x_t = tx_1 + (1-t)x_0.$$
5. Construct the velocity target
$$v_{\text{target}} = x_0 - x_1.$$
6. Predict
$$v_{\text{pred}} = v_\theta(x_t, t).$$

7. Minimize

$$\boxed{\mathcal{L} = \|v_\theta(x_t, t) - (x_0 - x_1)\|^2 .}$$

A notable feature of this construction is that we do not need to simulate the complete flow during training.

Because the linear path has a closed form,

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

we can jump directly to any randomly chosen time t.

This is analogous to the DDPM shortcut that allows us to sample x_t directly from x_0 without simulating every previous noising step.

17 A Complete Numerical Training Example

Suppose

$$x_0 = 3$$

and

$$x_1 = -5.$$

Sample

$$t = 0.25.$$

Then

$$x_t = 0.25(-5) + 0.75(3).$$

Therefore,

$$x_t = -1.25 + 2.25 = 1.$$

The target velocity is

$$v_{\text{target}} = x_0 - x_1 = 3 - (-5) = 8.$$

The neural network receives

$$(x_t, t) = (1, 0.25)$$

and is trained toward the target

$$8.$$

If the network predicts

$$v_\theta(1, 0.25) = 7.5,$$

then the squared-error loss is

$$(7.5 - 8)^2 = 0.25.$$

We then backpropagate this loss through the neural network.

18 Generation

After training, we no longer know the target endpoint x_0.

We begin only with

$$x_1 \sim q.$$

We then integrate the learned velocity field from $t = 1$ to $t = 0$.

Using a simple Euler discretization,

$$\boxed{x_{t-\Delta t} = x_t + v_\theta(x_t, t)\Delta t.}$$

For example,

$$x_{0.99} = x_1 + 0.01\, v_\theta(x_1, 1),$$

followed by

$$x_{0.98} = x_{0.99} + 0.01\, v_\theta(x_{0.99}, 0.99),$$

and so on until $t = 0$.

The final point is taken as the generated sample.

19 Training Versus Generation

The distinction between training and generation is important.

During training, we know both endpoints:

$$x_1 \qquad \text{and} \qquad x_0.$$

Therefore we can compute

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

and

$$x_0 - x_1.$$

During generation, however, we know only the source point

$$x_1.$$

The target point x_0 is unknown. Finding it is the whole purpose of generation.

Therefore we cannot directly use

$$x_0 - x_1.$$

Instead, we repeatedly evaluate the learned marginal velocity field

$$v_\theta(x_t, t)$$

and follow it numerically.

This is conceptually similar to diffusion training, where the clean sample is available during training but not during generation.

20 Why Generated Paths Need Not Be Straight

The pointwise training paths may all be straight:

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

However, the neural network does not learn one particular pointwise path.
It learns

$$v_t^\star(x) = \mathbb{E}\left[x_0 - x_1 \mid x_t = x\right].$$

This averaged field can vary with both space and time.
Therefore, a generated sample following

$$\frac{dx_t}{dt} = -v_t^\star(x_t)$$

can trace out a curved trajectory.

Thus,

straight pointwise training paths $\not\Rightarrow$ straight generated paths.

21 Where Did Diffusion Go?

DDPM begins by defining an explicit forward corruption process:

$$x_0 \to x_t \to x_T.$$

Flow matching does not require us to begin with such a process.
Instead, we choose:

1. a source distribution q,
2. a target distribution p,
3. a coupling between samples from q and p,
4. a pointwise path between paired samples.

These choices define a transport problem.
We then learn the marginal velocity field that performs the corresponding transport.
In that sense, flow matching is more general than diffusion.

22 The Three Main Modeling Choices

A flow-matching model is determined by three basic ingredients.

22.1 Source Distribution

Choose an easy-to-sample distribution

$$q.$$

A common choice is

$$q = \mathcal{N}(0, I),$$

but Gaussianity is not required.

22.2 Coupling

Choose a coupling

$$\Pi_{q,p}$$

between the source distribution q and target distribution p.

The simplest choice is independent sampling:

$$x_1 \sim q, \qquad x_0 \sim p.$$

22.3 Pointwise Flow

Choose a path connecting each paired source and target sample.

The simplest choice is linear interpolation:

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

These three ingredients jointly determine the marginal flow that the neural network is trained to approximate.

23 DDIM as a Special Case

Flow matching provides a useful reinterpretation of DDIM.

Rather than thinking of DDIM only as an algorithm derived from diffusion, we can think of it as one particular deterministic transport.

In the variance-exploding convention used in the source tutorial, the DDIM pointwise trajectory can be written as

$$x_t = x_0 + (x_1 - x_0)\sqrt{t}.$$

Compare this with linear flow matching:

$$x_t = x_0 + (x_1 - x_0)t.$$

The difference is largely a reparameterization of time.

Linear flow:

$$x_t = x_0 + (x_1 - x_0)t.$$

DDIM-like flow:

$$x_t = x_0 + (x_1 - x_0)\sqrt{t}.$$

The source tutorial therefore interprets variance-exploding DDIM as a special case of flow matching with a particular coupling and time parameterization.

24 A Useful Mental Model

It is useful to compare the three frameworks conceptually.

DDPM

Learn to reverse a stochastic corruption process:

$$\text{noise} \rightsquigarrow \text{data}.$$

Generation is stochastic.

DDIM

Interpret diffusion as a deterministic transport:

$$\text{noise} \xrightarrow{\text{deterministic trajectory}} \text{data}.$$

Flow Matching

Ask why we should restrict ourselves to the transport implied by diffusion.

Instead, choose convenient transport paths directly:

$$\boxed{\text{source} \xrightarrow{\text{chosen transport}} \text{data}.}$$

Then learn the velocity field that implements the corresponding marginal transport.

25 The Three Levels to Keep Separate

A useful way to organize the framework is to distinguish three different objects.

25.1 Pointwise Path

For one sampled pair

$$(x_1, x_0),$$

choose

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

This path is deliberately chosen by us.

25.2 Pointwise Velocity

The path has velocity

$$v_t^{[x_1,x_0]} = x_0 - x_1.$$

This gives the supervised training target.

25.3 Marginal Velocity

Many different pointwise trajectories may overlap.

Their conditional average defines

$$v_t^\star(x) = \mathbb{E}\left[v_t^{[x_1,x_0]} \mid x_t = x\right].$$

This is the object the neural network ultimately learns and the field that is integrated during generation.

The complete logical chain is therefore

Choose easy pointwise paths
↓
Obtain easy velocity targets
↓
L_2 regression implicitly averages them
↓
Learn marginal velocity field
↓
Integrate learned field
↓
$x_1 \sim q \longrightarrow x_0 \sim p$

26 Comparison with DDPM Training

There is a strong structural analogy between DDPM training and linear flow matching.

In DDPM, one may construct

$$x_t = x_0 + \sigma_t \epsilon$$

and train

$$\epsilon_\theta(x_t, t)$$

against the sampled noise

$$\epsilon.$$

Different combinations of x_0 and ϵ can produce the same x_t, so squared-error regression causes the network to learn an appropriate conditional expectation.

Linear flow matching has the same regression structure.

Construct

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

and train

$$v_\theta(x_t, t)$$

against

$$x_0 - x_1.$$

Again, multiple endpoint pairs can generate the same intermediate point, so the network learns the corresponding conditional average.

A useful comparison is

DDPM	Linear Flow Matching
$x_0 \sim p$	$x_0 \sim p$
$\epsilon \sim \mathcal{N}(0, I)$	$x_1 \sim q$
$x_t = \text{corrupt}(x_0, \epsilon, t)$	$x_t = tx_1 + (1-t)x_0$
target: ϵ	target: $x_0 - x_1$
$\epsilon_\theta(x_t, t)$	$v_\theta(x_t, t)$
L_2 regression	L_2 regression

The predicted quantity changes, but the underlying supervised-regression logic is closely related.

27 Core Ideas to Retain

The main ideas from this introduction to flow matching are:

1. A *flow* is a time-dependent velocity field $v_t(x)$.
2. Integrating that field moves individual particles and therefore transports whole probability distributions.
3. We create simple *pointwise flows* between known source-target pairs.
4. For linear flow matching,
$$x_t = tx_1 + (1-t)x_0$$
and
$$v_t^{[x_1,x_0]} = x_0 - x_1.$$
5. The pointwise velocity is not the final generative velocity field.
6. The desired marginal velocity field is
$$v_t^\star(x_t) = \mathbb{E}\left[v_t^{[x_1,x_0]} \mid x_t\right].$$
7. We do not calculate this conditional expectation explicitly.
8. Squared-error regression against sampled pointwise velocities causes the neural network to learn the conditional expectation automatically.
9. During generation, we sample $x_1 \sim q$ and numerically integrate the learned velocity field from $t = 1$ to $t = 0$.
10. DDIM can be interpreted as a special deterministic flow, while flow matching allows a broader family of transports.