Flow Matching for Generative Modeling

Pramook Khungurn

July 24, 2024

This note was written as I read the "Flow Matching for Generative Modeling" paper by Lipman et al. [7]. In this document, we use both standard notations for partial derivatives and my notations for partial derivatives [5]. The latter is used when we need want to avoid all ambiguities.

1 Background

1.1 Flow and Neural ODE

- A data item is denoted by $x = (x^1, x^2, \ldots, x^d) \in \mathbb{R}^d$.
- **Definition 1.** *A* **time-dependent vector field** *is a function* $v : [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$.
- For a time dependent function $f : [0,1] \times \mathbb{R}^d \to R$ for some range set R, we may write $f(t,x)$ as $f_t(x)$ to emphasize time dependence. Moreover, we can refer to $f_t : \mathbb{R}^d \to R$ as a function in its own right.
  - With this, for a time-dependent vector field v, we may write v_t to mean the vector field obtained from v when restricted to a given time.
- **Definition 2.** *A vector function* $f : \mathbb{R}^d \to \mathbb{R}^d$ *is called a* **diffeomorphism** *if* f *is differentiable and bijective.*
  - If f is a diffeomorphism, then so is f^{-1}.
  - If f is a diffeomorphism, then, for all $x \in \mathbb{R}^d$, its Jacobian matrix $\nabla f(x)$ (my notation [5]) is invertible. This means that $\det \nabla f(x) \neq 0$.
- **Definition 3.** *A* **flow** *is a time-dependent vector field* $\phi : [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$ *such that*
  - ϕ *is differentiable.*
  - $\phi_0(x) = x$ *for all* x*, and*
  - ϕ_t*, when viewed as a function of signature* $\mathbb{R}^d \to \mathbb{R}^d$*, is a diffeomorphism for all* t*.*
- We note that we call such a function a flow because, for any x, the function $\Phi_x : t \mapsto \phi_t(x)$ traces a path of a particle that starts at x and moves through space. So, ϕ indicates how particles "flow" through space.
- If ϕ is a flow, we can interpret $\phi_t(x)$ as the position at time t of the particle that starts at x.
- **Definition 4.** *Let* v *be a time-dependent vector field and* ϕ *be a flow. We say that* v **generates** ϕ *if*

$$\frac{\partial}{\partial t}\phi_t(x) = v_t(\phi_t(x)). \tag{1}$$

  In other words, v_t acts as a velocity field that governs the direction and the speed that a particle at each position move.

- Note that we can rewrite Equation (1) using my notation [5] as:
$$\nabla_1 \phi_t(x) = v_t(\phi_t(x)). \tag{2}$$
- We note that, for any time-dependent vector field v, it generates a flow ϕ that is obtained by solving the differential equation $\partial\phi_t(x)/\partial t = v_t(\phi_t(x))$ with the initial condition $\phi_0(x) = x$.
- Chen et al. proposed the **neural ordinary differential equation** model [2]. The idea is to model the vector field v with a neural network $v_t(x;\theta)$. The vector field generates a flow ϕ_t. The goal is to make ϕ_1 have properties that we want.
  - If you want a refresher on neural ODE, then read my previous note on it [4].
  - Chen et al.'s paper proposes a way to train $v_t(x;\theta)$. However, this training method involves integrating the vector field in each iteration, which means that optimization takes a long time. This is still incredible because the gradient of this process can be computed kind of easily.
- The flow matching paper is here to offer another way to train a neural ODE without integrating the vector field. However, the flow matching algorithm is specialized to the task of generative modeling, so its scope is narrower than the neural ODE paper.

1.2 Generative Modeling as Transformation of Probability Distribution

- At a high level, generative modeling is about transforming a noise distribution p_{noise} to a distribution of data items p_{data}.
- Perhaps the easiest form of transforming one probability distribution to another is the following process.
  - Sample a data item from the starting probability distribution.
  - Transform the data item in some way.
  - Return the transformed result to the usesr.

  This process has a name.

  Definition 5. *Let $p : \mathbb{R}^d \to \mathbb{R}^+ \cup \{0\}$ be a probability distribution. Let $f : \mathbb{R}^d \to \mathbb{R}^d$ be a diffeomorphism. A* **push-forward** *or a* **change of variable** *of p according to f is the probability distribution q of elements y created through the following process:*
  - *Sample $x \sim p$.*
  - *Compute $y = f(x)$.*

  Notationally, we write $q = [f]_*p$.

- **Lemma 6.** *Let $q = [f]_*p$. It follows that*
$$q(y) = [f]_*p(y) = p(f^{-1}(y))|\det \nabla f^{-1}(y)|,$$
  or
$$q(f(x)) = [f]_*p(f(x)) = \frac{p(x)}{|\det \nabla f(x)|}.$$

  Here, ∇f is a notation for the derivative (the Jacobian matrix) of f according to my notation [5]. A non-rigourous proof of this lemma can be found in my note on probability density under transformation [6]. It is analogous to the change of variable formula in multi-variable calculus.

- **Definition 7.** *A* **probability density path** *is a function* $p : [0,1] \times \mathbb{R}^d \to \mathbb{R}^+ \cup \{0\}$ *such that each* $p(t, \cdot)$ *is a probabilty density function on* $\mathbb{R}^d$*. In other words, it holds that*

$$\int p(t,x)\,\mathrm{d}x = 1$$

for all $t \in [0,1]$*.*

- A flow ϕ_t can be used to transform one probability to another in a gradual sort of way.

  Lemma 8. *Let* $p_0 : \mathbb{R}^d \to \mathbb{R}^+ \cup \{0\}$ *be a probability distribution and* ϕ *be a flow. The* **push-forward** *or the* **change of variable** *of* p_0 *according to* ϕ *is a probability path* $p : [0,1] \times \mathbb{R}^d \to \mathbb{R}^+ \cup \{0\}$ *that is the result of the following process:*

  – *Sample* $x_0 \sim p_0$*.*
  – *Apply the flow to get* $x_t = \phi_t(x)$*.*
  – *Let* p_t *be the distribution of* x_t*.*

  It follows that $p_t = [\phi_t]_* p_0$*. In other words,*

$$p_t(x') = p_0(\phi_t^{-1}(x'))\,|\det \nabla \phi_t^{-1}(x')| \tag{3}$$

  for all $x' \in \mathbb{R}^d$*.*

  So, instead of getting just one probability distribution from p_0, we get an infinite number of distributions.

- The formula in Equation (3) is not that great because there is an issue about arity. We said earlier that ϕ_t can be viewed as a function of signature $\mathbb{R}^d \to \mathbb{R}^d$. However, ϕ itself has signature $[0,1] \times \mathbb{R}^d \to \mathbb{R}^d$, which means that it is function that maps a $(d+1)$-dimensional space to a d-dimensional space. So, we can treat it in the same way as a function of signature $\mathbb{R}^{d+1} \to \mathbb{R}^d$. In other words, we can say that ϕ takes $d+1$ inputs. We can then divide the $d+1$ inputs into two blocks.

  – The first block is the first argument alone. Using Python slice notation, it is "1 : 2." Using my "chapter" notation, it can be abbreviated as §1.
  – The second block is the rest of the arguments. Using Python slice notation, it is "2 : $d+2$." Using my "chapter" notation, it can be abbreviate as §2.

  Hence, using my notation for partial derivatives, we can rewrite the equation as:

$$p_t(x') = p_0(\phi_t^{-1}(x'))\,|\det \nabla_{\S 2} \phi_t^{-1}(x')|. \tag{4}$$

- **Definition 9.** *If* p_t *is a probability path that is a push-forward of* p_0 *according to flow* ϕ_t*, and* v_t *generates* ϕ_t*, we say that* v_t **generates** p_t*.*

- The neural ODE framework can be used to do generative modeling in the following way. We train a neural network to model a vector field $v_t(x;\theta)$ such that it generates a flow ϕ_t so that the push-forward of a noise distribution $p_0 = p_{\text{noise}}$ results in a probability path p_t such that $p_1 = p_{\text{data}}$.

  A neural ODE used in this way is called a **continous normalizing flow** model.

2 Flow Matching

2.1 Flow Matching Objective

- We want to use the above framework to transform a simple noise distribution $p_0 = p_{\text{noise}}$ to a data distribution $p_1 = p_{\text{data}}$.
  - p_{noise} is typically a Gaussian distribution $p_{\text{noise}} = \mathcal{N}(0, I)$.
  - As in most ML settings, we do not have access to the density function p_{data}, but we only have samples from the distribution.
- Suppose we know a probability path p_t and a time-dependent vector field u_t that has the following property:
  - p_0 is the desired noise distribution, and p_1 is the desired data distribution.
  - u_t generates p_t.

  Suppose again that we want to model u_t with a neural network $v_t(x; \theta)$. Then, we may do it my minimizing the **flow matching objective**:

$$\mathcal{L}_{\text{FM}}(\theta) = E_{\substack{t \sim \mathcal{U}([0,1]),\\ x \sim p_t(x)}} \big[\| u_t(x) - v_t(x; \theta) \|^2 \big]. \tag{5}$$

- The flow maching objective is usable if we know p_t and u_t before hand. However, in our settings, we do not know anything about u_t, and we only know $p_0 = p_{\text{noise}}$ and $p_1 = p_{\text{data}}$ but nothing in between.

2.2 Special Case: Single Item Dataset

- One of the difficulty we are facing right now is that p_{data} can be quite complicated and that we only have access to its samples, not a function that can evaluate the density of sample from the distribution.
- So, let's start with a special case where the distribution can generate exactly one data item. Let us call this item x_{data}.
  - You know where this is going. We will later approximate p_{data} as a mixture of the distributions of individual samples. So, stay tuned and work with this special case first.
- The distribution p_{data} is given by $p_{\text{data}} = \delta(x_{\text{data}})$ where δ is the Dirac delta function.
- We want to derive a vector field u_t that generates a probability distribution p_t so that (1) $p_0 = p_{\text{noise}} = \mathcal{N}(0, I)$ and $p_1 = p_{\text{data}} = \delta(x_{\text{data}})$.
- Unfortunately, I don't think there is a finite-time process that can turn a Gaussian distribution into a delta distribution. So, we will settle for an approximation. We instead require that

$$p_1 = \mathcal{N}(x_{\text{data}}, \sigma_{\min}^2 I)$$

  where $\sigma_{\min}$ is a small positive constant.
- Since u_t and p_t we shall derive is specific to x_{data}, we may write them as "conditional" vector field and probability density, using the notation $u_t(\cdot|x_{\text{data}})$ and $p_t(\cdot|x_{\text{data}})$.
  - Of course, this will be used later when we approximate of p_{data} as a mixture of single item distributions.
- In other words, we want to find a time-dependent vector field $u_t(\cdot|x_{\text{data}})$ that generates a probability path $p_t(\cdot|x_{\text{data}})$ such that

- $p_0(x|x_{\text{data}}) = \mathcal{N}(x; 0, I)$, and
- $p_1(x|x_{\text{data}}) = \mathcal{N}(x; x_{\text{data}}, \sigma_{\min}^2 I)$.

- We let $p_t(\cdot|x_{\text{data}})$ take the form
$$p_t(x|x_{\text{data}}) = \mathcal{N}(x; \mu_t(x_{\text{data}}), \sigma_t(x_{\text{data}})^2 I) \tag{6}$$
where $\mu : [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$ and $\sigma : [0,1] \times \mathbb{R} \to \mathbb{R}^+$. We will specify these two functions later, but there are many choises of them.

- To satsify the requirement on p_0, it must be the case that
  - $\mu_0(x_{\text{data}}) = 0$ for all x_{data}, and
  - $\sigma_0(x_{\text{data}}) = 1$ for all x_{data}.

  Morever, to satisfy the requirment on p_1, it must be the case that
  - $\mu_1(x_{\text{data}}) = x_{\text{data}}$ for all x_{data}, and
  - $\sigma_1(x_{\text{data}}) = \sigma_{\min}$ for all x_{data}.

- Now that we have specified the form of $p_t(\cdot|x_{\text{data}})$, it is now time to figure out the vector field $u_t(\cdot|x_{\text{data}})$ that generates it.

- We do so by first specifying a flow ψ_t such that $p_t(\cdot|x_{\text{data}}) = [\psi_t]_* p_0(\cdot|x_{\text{data}})$. Then, we can define u_t according to the equation
$$\nabla_1 \psi_t(x) = u_t(\psi_t(x)|x_{\text{data}}).$$
In other words,
$$u_t(x'|x_{\text{data}}) = \nabla_1 \psi_t(\psi_t^{-1}(x')). \tag{7}$$

- Now, let's specify ψ_t. We use a very simple flow:
$$\psi_t(x) = \sigma_t(x_{\text{data}})x + \mu_t(x_{\text{data}})$$
Let's do some sanity check.
  - At $t = 0$, we have that $\phi_0(x) = x$. So, ϕ_t satisfies the initial condition. The distribution p_0 is the distribution of x, which is $\mathcal{N}(0, I)$ as required.
  - At $t = 1$, we have that $\phi_1(x) = \sigma_{\min} x + x_{\text{data}}$. Because $x \sim \mathcal{N}(0, I)$, we have that $p_1 \sim \mathcal{N}(x_{\text{data}}, \sigma_{\min}^2 I)$ as required too.
  - At other values of t, we have that $\phi_t(x) = \sigma_t(x_{\text{data}})x + \mu_t(x_{\text{data}})$. Again, because $x \sim \mathcal{N}(0, I)$, we have that $\phi_t(x) \sim \mathcal{N}(\mu_t x_{\text{data}}, \sigma_t(x_{\text{data}})^2 I)$ as required again.

- Let's derive $u_t(x'|x_{\text{data}})$.
  - First, we need to derive $\psi_t^{-1}(x')$. Let $x = \psi_t^{-1}(x')$. We have that
  $$x' = \psi_t(x) = \sigma_t(x_{\text{data}})x + \mu_t(x_{\text{data}}).$$
  So,
  $$x = \frac{x' - \mu_t(x_{\text{data}})}{\sigma_t(x_{\text{data}})}.$$
  In other words,
  $$\psi_t^{-1}(x') = \frac{x' - \mu_t(x_{\text{data}})}{\sigma_t(x_{\text{data}})}. \tag{8}$$

– Second, we need to derive the time-derivative $\nabla_1 \psi_t(x)$. This is also simple:

$$\begin{aligned}\nabla_1 \psi_t(x) &= \frac{\partial}{\partial t}\big[\sigma_t(x_{\text{data}})x + \mu_t(x_{\text{data}})\big] \\ &= \bigg(\frac{\partial}{\partial t}\sigma_t(x_{\text{data}})\bigg)x + \frac{\partial}{\partial t}\mu_t(x_{\text{data}}) \\ &= x\nabla_1\sigma_t(x_{\text{data}}) + \nabla_1\mu_t(x_{\text{data}}). \end{aligned} \tag{9}$$

- Substituting (8) and (9) into (7), we have that

$$\begin{aligned} u_t(x'|x_{\text{data}}) &= \bigg(\frac{x' - \mu_t(x_{\text{data}})}{\sigma_t(x_{\text{data}})}\bigg)\nabla_1\sigma_t(x_{\text{data}}) + \nabla_1\mu_t(x_{\text{data}}) \\ &= \frac{\nabla_1\sigma_t(x_{\text{data}})}{\sigma_t(x_{\text{data}})}(x' - \mu_t(x_{\text{data}})) + \nabla_1\mu_t(x_{\text{data}}). \end{aligned}$$

- **Theorem 10.** *Suppose that we are given the following functions.*
  - *Let $\sigma : [0,1] \times \mathbb{R}^d \to \mathbb{R}^+$ be a differentable function such that $\sigma_0(x) = 1$ and $\sigma_1(x) = \sigma_{\min}$ for all x.*
  - *Let $\mu : [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$ be a differentiable time-dependent vector field such that $\mu_0(x) = 0$ and $\mu_1(x) = x$ for all x.*

  Then, the vector field

$$u_t(x|x_{\text{data}}) = \frac{\nabla_1\sigma_t(x_{\text{data}})}{\sigma_t(x_{\text{data}})}(x - \mu_t(x_{\text{data}})) + \nabla_1\mu_t(x_{\text{data}})$$

  generates the flow

$$\phi_t(x) = \sigma_t(x_{\text{data}})x + \mu_t(x_{\text{data}})$$

  and a probablity path $p_t(\cdot|x_{\text{data}})$ such that that

$$p_t(\cdot|x_{\text{data}}) \sim \mathcal{N}(\mu_t(x_{\text{data}}), \sigma_t(x_{\text{data}})^2 I)$$

  for all t. In particular, we have that

  1. *$p_0(\cdot|x_{\text{data}}) \sim \mathcal{N}(0, I)$, and*
  2. *$p_1(\cdot|x_{\text{data}}) \sim \mathcal{N}(x_{\text{data}}, \sigma^2_{\min} I)$.*

  So, $u_t(\cdot|x_{\text{data}})$ transforms a Gaussian noise distribution into an approximation of a single data distribution that only contains x_{data}.

2.3 From single item distribution to multi-item distribution

- Now, we get back to the case where p_{data} is not a distrbution that output only a single item.
- Using the law of total proability, we can define the **marginal probability path** as

$$p_t(x) = \int p_t(x|x_1)p_{\text{data}}(x_{\text{data}})\,\mathrm{d}x_{\text{data}}. \tag{10}$$

- It follows that $p_0 = \mathcal{N}(0, I)$, and

$$p_1 = p_{\text{data}} * \mathcal{N}(0, \sigma^2_{\min} I) \approx p_{\text{data}}.$$

  where $*$ is the convolution operation. So, we can use p_1 in place of p_{data} in many cases.

- Our task is now to find a time-dependent vector field u_t that generates p_t. The paper argues that the following **marginal vector field**,

$$u_t(x) = \int u_t(x|x_{\mathrm{data}}) \frac{p_t(x|x_{\mathrm{data}}) p_{\mathrm{data}}(x_{\mathrm{data}})}{p_t(x)} \,\mathrm{d}x_{\mathrm{data}}, \tag{11}$$

  works.

- **Theorem 11.** *The marginal vector field u_t defined in Equation* (11) *generates the marginal probability path p_t in Equation* (10).

  Proof. This proof makes heavy use of the continuity equation (12) and Theorem 16.

  We showed in the last section that the conditional vector field $u_t(\cdot|x_{\mathrm{data}})$ generates the condtional probability path $p_t(\cdot|x_{\mathrm{data}})$. To make derivation easier, we shall write $p_t(x|x_{\mathrm{data}})$ as $p_{|x_{\mathrm{data}}}(t,x)$ and $u_t(x|x_{\mathrm{data}})$ as $u_{|\mathrm{data}}(t,x)$. With this, we have that these two functions satisfy the continuity equation

$$\nabla_1 p_{|x_{\mathrm{data}}}(t,x) + \sum_{i=1}^{d} \nabla_{i+1} (p_{|x_{\mathrm{data}}} u^i_{|x_{\mathrm{data}}})(t,x) = 0.$$

  In other words,

$$\nabla_1 p_{|x_{\mathrm{data}}}(t,x) = -\sum_{i=1}^{d} \nabla_{i+1} (p_{|x_{\mathrm{data}}} u^i_{|x_{\mathrm{data}}})(t,x).$$

  Now, recall the definition of $p(t,x)$.

$$p(t,x) = \int p_{|x_{\mathrm{data}}}(t,x) p_{\mathrm{data}}(x_{\mathrm{data}}) \,\mathrm{d}x_{\mathrm{data}}.$$

  Differentiating both sides with respect to the first argument (t), we have that

$$\begin{aligned}
\nabla_1 p(t,x) &= \int \nabla_1 p_{|x_{\mathrm{data}}}(t,x) p_{\mathrm{data}}(x_{\mathrm{data}}) \,\mathrm{d}x_{\mathrm{data}} \\
&= \int \bigg(-\sum_{i=1}^{d} \nabla_{i+1} (p_{|x_{\mathrm{data}}} u^i_{|x_{\mathrm{data}}})(t,x) \bigg) p_{\mathrm{data}}(x_{\mathrm{data}}) \,\mathrm{d}x_{\mathrm{data}} \\
&= -\sum_{i=1}^{d} \int \nabla_{i+1} (p_{|x_{\mathrm{data}}} u^i_{|x_{\mathrm{data}}})(t,x) p_{\mathrm{data}}(x_{\mathrm{data}}) \,\mathrm{d}x_{\mathrm{data}} \\
&= -\sum_{i=1}^{d} \nabla_{i+1} \bigg(\int p_{|x_{\mathrm{data}}}(t,x) u_{|x_{\mathrm{data}}}(t,x) p_{\mathrm{data}}(x_{\mathrm{data}}) \,\mathrm{d}x_{\mathrm{data}} \bigg) \\
&= -\sum_{i=1}^{d} \nabla_{i+1} \bigg(\int p_t(x|x_{\mathrm{data}}) u_t(x|x_{\mathrm{data}}) p_{\mathrm{data}}(x_{\mathrm{data}}) \,\mathrm{d}x_{\mathrm{data}} \bigg).
\end{aligned}$$

  By Equation (11), we have that

$$u_t(x) p_t(x) = \int p_t(x|x_{\mathrm{data}}) u_t(x|x_{\mathrm{data}}) p_{\mathrm{data}}(x_{\mathrm{data}}) \,\mathrm{d}x_{\mathrm{data}}.$$

As a result,

$$\nabla_1 p(t,x) = -\sum_{i=1}^{d} \nabla_{i+1}\big(u_t(x)p_t(x)\big)$$
$$\nabla_1 p(t,x) = -\sum_{i=1}^{d} \nabla_{i+1}(up)(t,x)$$
$$\nabla_1 p(t,x) + \sum_{i=1}^{d} \nabla_{i+1}(up)(t,x) = 0.$$

This shows that u_t and p_t satisfies the continuity equation, which implites that u_t generates p_t. □

2.4 Conditional Flow Matching

- We have just identified the vector field u_t that we can used in the flow matching loss (5). However, the problem is that u_t is defined as an integral, and we do not want to compute it directly.
- Instead, we optimize the following **conditional flow matching objective** where we sample x_{data} and trying to match $v_t(x;\theta)$ against $u_t(x|x_{\text{data}})$. Here, x is sampled from the conditional distribution $p_t(x|x_{\text{data}})$.

$$\mathcal{L}_{\text{CFM}}(\theta) = E_{\substack{t\sim\mathcal{U}([0,1]),\\ x_{\text{data}}\sim p_{\text{data}},\\ x\sim p_t(x|x_{\text{data}})}}[\|u_t(x|x_{\text{data}}) - v_t(x;\theta)\|^2].$$

- We can sample x_{data} easily because we can sample uniformly from the collections of samples we have at hand.
- We can also sample from $p_t(x|x_{\text{data}})$ easily because $p_t(x|x_{\text{data}}) = \mathcal{N}(x;\mu_t\text{data}(x_{\text{data}}), \sigma_t(x_{\text{data}})^2 I)$.
- The only concern is whether the conditional flow matching objective $\mathcal{L}_{\text{CFM}}(\theta)$ would yield the same θ as $\mathcal{L}_{\text{FM}}(\theta)$ after optimization. The answer is yes.
- **Theorem 12.** *Assuming that $p_t(x) > 0$ for all $x \in \mathbb{R}^d$ and $t \in [0,1]$, then,*

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

where C is a constant independent of θ. As a result,

$$\nabla_\theta \mathcal{L}_{\text{FM}}(\theta) = \nabla_\theta \mathcal{L}_{\text{CFM}}(\theta).$$

Proof. We assume that all functions are well-behaved so that we can say that all integrals exist use the standard trick such as exchanging the order of integration (Fubini's theorem).

We have that

$$\|v_t(x;\theta) - u_t(x)\|^2 = \|v_t(x;\theta)\|^2 - 2\langle v_t(x;\theta), u_t(x)\rangle + \|u_t(x)\|^2$$
$$\|v_t(x;\theta) - u_t(x|x_{\text{data}})\|^2 = \|v_t(x;\theta)\|^2 - 2\langle v_t(x;\theta), u_t(x|x_{\text{data}})\rangle + \|u_t(x|x_{\text{data}})\|^2.$$

So,

$$\mathcal{L}_{\text{FM}}(\theta) = E_{\substack{t\sim\mathcal{U}([0,1]),\\ x\sim p_t(x)}}[\|v_t(x;\theta)\|^2] - 2E_{\substack{t\sim\mathcal{U}([0,1]),\\ x\sim p_t(x)}}[\langle v_t(x;\theta), u_t(x)\rangle] + E_{\substack{t\sim\mathcal{U}([0,1]),\\ x\sim p_t(x)}}[\|u_t(x)\|^2]$$
$$\mathcal{L}_{\text{CFM}}(\theta) = E_{\substack{t\sim\mathcal{U}([0,1]),\\ x_{\text{data}}\sim p_{\text{data}},\\ x\sim p_t(x|x_{\text{data}})}}[\|v_t(x;\theta)\|^2] - 2E_{\substack{t\sim\mathcal{U}([0,1]),\\ x_{\text{data}}\sim p_{\text{data}},\\ x\sim p_t(x|x_{\text{data}})}}[\langle v_t(x;\theta), u_t(x|x_{\text{data}})\rangle] + E_{\substack{t\sim\mathcal{U}([0,1]),\\ x_{\text{data}}\sim p_{\text{data}},\\ x\sim p_t(x|x_{\text{data}})}}[\|u_t(x|x_{\text{data}})\|^2].$$

Note that $u_t(x)$ and $u_t(x|x_{\mathrm{data}})$ does not depend on θ. As a result, we can treat them as constants. In other words,

$$\begin{aligned}
\mathcal{L}_{\mathrm{FM}}(\theta) &= E_{\substack{t\sim\mathcal{U}([0,1]),\\ x\sim p_t(x)}}[\|v_t(x;\theta)\|^2] - 2E_{\substack{t\sim\mathcal{U}([0,1]),\\ x\sim p_t(x)}}[\langle v_t(x;\theta), u_t(x)\rangle] + C_1 \\
\mathcal{L}_{\mathrm{CFM}}(\theta) &= E_{\substack{t\sim\mathcal{U}([0,1]),\\ x_{\mathrm{data}}\sim p_{\mathrm{data}},\\ x\sim p_t(x|x_{\mathrm{data}})}}[\|v_t(x;\theta)\|^2] - 2E_{\substack{t\sim\mathcal{U}([0,1]),\\ x_{\mathrm{data}}\sim p_{\mathrm{data}},\\ x\sim p_t(x|x_{\mathrm{data}})}}[\langle v_t(x;\theta), u_t(x|x_{\mathrm{data}})\rangle] + C_2.
\end{aligned}$$

Next, note that

$$E_{\substack{t\sim\mathcal{U}([0,1]),\\ x\sim p_t(x)}}[\|v_t(x;\theta)\|^2] = E_{\substack{t\sim\mathcal{U}([0,1]),\\ x_{\mathrm{data}}\sim p_{\mathrm{data}},\\ x\sim p_t(x|x_{\mathrm{data}})}}[\|v_t(x;\theta)\|^2]$$

because we still sample x from the same distribution $p_t(x)$. On the LHS, we sample x directly, but, on the RHS, we sample x_{data} before sampling x given x_{data}.

Lastly,

$$\begin{aligned}
&E_{\substack{t\sim\mathcal{U}([0,1]),\\ x\sim p_t(x)}}[\langle v_t(x;\theta), u_t(x)\rangle] \\
&= E_{\substack{t\sim\mathcal{U}([0,1]),\\ x\sim p_t(x)}}\bigg[\bigg\langle v_t(x;\theta), \int u_t(x|x_{\mathrm{data}})\frac{p_t(x|x_{\mathrm{data}})p_{\mathrm{data}}(x_{\mathrm{data}})}{p_t(x)}\,\mathrm{d}x_{\mathrm{data}}\bigg\rangle\bigg] \\
&= E_{\substack{t\sim\mathcal{U}([0,1]),\\ x\sim p_t(x)}}\bigg[\int \langle v_t(x;\theta), u_t(x|x_{\mathrm{data}})\rangle\frac{p_t(x|x_{\mathrm{data}})p_{\mathrm{data}}(x_{\mathrm{data}})}{p_t(x)}\,\mathrm{d}x_{\mathrm{data}}\bigg] \\
&= E_{t\sim\mathcal{U}([0,1])}\bigg[\int p_t(x)\bigg(\int \langle v_t(x;\theta), u_t(x|x_{\mathrm{data}})\rangle\frac{p_t(x|x_{\mathrm{data}})p_{\mathrm{data}}(x_{\mathrm{data}})}{p_t(x)}\,\mathrm{d}x_{\mathrm{data}}\bigg)\mathrm{d}x\bigg] \\
&= E_{t\sim\mathcal{U}([0,1])}\bigg[\int\int \langle v_t(x;\theta), u_t(x|x_{\mathrm{data}})\rangle p_t(x|x_{\mathrm{data}})p_{\mathrm{data}}(x_{\mathrm{data}})\,\mathrm{d}x_{\mathrm{data}}\,\mathrm{d}x\bigg] \\
&= E_{t\sim\mathcal{U}([0,1])}\bigg[\int p_{\mathrm{data}}(x_{\mathrm{data}})\bigg(\int p_t(x|x_{\mathrm{data}})\langle v_t(x;\theta), u_t(x|x_{\mathrm{data}})\rangle\,\mathrm{d}x\bigg)\,\mathrm{d}x_{\mathrm{data}}\bigg] \\
&= E_{\substack{t\sim\mathcal{U}([0,1]),\\ x_{\mathrm{data}}\sim p_{\mathrm{data}},\\ x\sim p_t(x|x_{\mathrm{data}})}}[\langle v_t(x;\theta), u_t(x|x_{\mathrm{data}})\rangle].
\end{aligned}$$

We are done. $\square$

3 Conditional Vector Fields

- We said earlier that there are multiple ways to define μ_t and σ_t in Equation (6). We discuss them in this section.

3.1 Diffusion Conditional Vector Fields

- For a diffusion model, time is the reverse of what we have been using in this note.
  - p_0 is p_{data} or an approximation of it.
  - p_1 is a noise distribution.

  However, to make things simple, we will reverse the time so that it complies with what we have in this note.

- Variance-exploding case.
  - Given x_{data}, we have that $p_t(x|x_{\text{data}}) = \mathcal{N}(x; x_{\text{data}}, \sigma_t^2 I)$ where $\sigma_1 = 0$ and $\sigma_0 \gg 1$.
  - Hence, $\mu_t(x_{\text{data}}) = x_{\text{data}}$, and $\sigma_t(x_{\text{data}})$ does not depend on x_{data}.
  - As a result, the conditional vector field is given by

$$u_t(x|x_{\text{data}}) = \frac{\sigma_t'}{\sigma_t}(x - x_{\text{data}}).$$

- Variance-preserving case.
  - Given x_{data}, we have that $p_t(x|x_{\text{data}}) = \mathcal{N}(x; \alpha_t x_{x_{\text{data}}}, (1-\alpha_t^2)I)$ where $\alpha_t : [0,1] \to \mathbb{R}^+ \cup \{0\}$ is a real function such that $\alpha_0 = 0$ and $\alpha_1 = 1$.
  - In other words, $\mu_t(x_{\text{data}}) = \alpha_t x_{\text{data}}$ and $\sigma_t(x_{\text{data}}) = \sqrt{1-\alpha_t^2}$.
  - The conditonal vector field is given by

$$\begin{aligned} u_t(x|x_{\text{data}}) &= \frac{\{\sqrt{1-\alpha_t^2}\}'}{\sqrt{1-\alpha_t^2}}(x - \alpha_t x_{\text{data}}) + \alpha_t' x_{\text{data}} \\ &= \frac{-\alpha_t \alpha_t'}{1-\alpha_t^2}(x - \alpha_t x_{\text{data}}) + \alpha' x_{\text{data}}. \end{aligned}$$

3.2 Optimal Transport Conditional Vector Field

- We simply make μ_t and σ_t linear functions of time.

$$\begin{aligned} \mu_t(x_{\text{data}}) &= t x_{\text{data}}, \\ \sigma_t(x_{\text{data}}) &= 1 - (1-\sigma_{\min})t. \end{aligned}$$

  This leads to the conditional vector field

$$u_t(x|x_{\text{data}}) = \frac{-(1-\sigma_{min})}{1-(1-\sigma_{\min})t}(x - t x_{\text{data}}) + x_{\text{data}} = \frac{x_{\text{data}} - (1-\sigma_{\min})x}{1-(1-\sigma_{\min})t}.$$

  The conditional flow is given by:

$$\psi_t(x) = \sigma_t(x_{\text{data}})x + \mu_t(x_{\text{data}}) = (1-(1-\sigma_{\min})t)x + t x_{\text{data}}.$$

  The conditional flow matching objective is given by

$$\begin{aligned} \mathcal{L}_{\text{CFM}}(\theta) &= E_{\substack{t\sim\mathcal{U}([0,1]),\\ x_{\text{data}}\sim p_{\text{data}},\\ x\sim p_t(x|x_{\text{data}})}} [\|u_t(x|x_{\text{data}}) - v_t(x;\theta)\|^2] \\ &= E_{\substack{t\sim\mathcal{U}([0,1]),\\ x_{\text{data}}\sim p_{\text{data}},\\ x\sim p_t(x|x_{\text{data}})}} \left[\left\| \frac{x_{\text{data}} - (1-\sigma_{\min})x}{1-(1-\sigma_{\min})t} - v_t(x;\theta) \right\|^2 \right]. \end{aligned}$$

- Let's rewrite the loss so that it becomes simpler. We have that $x \sim p_t(x|x_{\text{data}})$ simply means that $x = (1-(1-\sigma_{\min})t)x_0 + t x_{\text{data}}$ where $x_0 \sim p_0$. So,

$$\begin{aligned} &\mathcal{L}_{\text{CFM}}(\theta) \\ &= E_{\substack{t\sim\mathcal{U}([0,1]),\\ x_{\text{data}}\sim p_{\text{data}},\\ x_0\sim p_0}} \left[\left\| \frac{x_{\text{data}} - (1-\sigma_{\min})((1-(1-\sigma_{\min})t)x_0 + t x_{\text{data}})}{1-(1-\sigma_{\min})t} - v_t((1-(1-\sigma_{\min}t))x_0 + t x_{\text{data}};\theta) \right\|^2 \right] \\ &= E_{\substack{t\sim\mathcal{U}([0,1]),\\ x_{\text{data}}\sim p_{\text{data}},\\ x_0\sim p_0}} \left[\left\| \frac{(1-(1-\sigma_{\min})t)(x_{\text{data}} - (1-\sigma_{\min})x_0)}{1-(1-\sigma_{\min})t} - v_t((1-(1-\sigma_{\min}t))x_0 + t x_{\text{data}};\theta) \right\|^2 \right] \\ &= E_{\substack{t\sim\mathcal{U}([0,1]),\\ x_{\text{data}}\sim p_{\text{data}},\\ x_0\sim p_0}} \left[\left\| x_{\text{data}} - (1-\sigma_{\min})x_0 - v_t((1-(1-\sigma_{\min}t))x_0 + t x_{\text{data}};\theta) \right\|^2 \right]. \end{aligned}$$

- The paper seems to make a great deal with the flow ψ_t is the "optimal transport displacement map" between two Gaussians.
  - Honestly, I found this to be very difficult to parse.
  - The paper cites a paper by McCann [9], but McCann's paper does not contain the word "optimal transport" or the word "displacement map."
- McCann's paper defines something called the "displacement interpolation."

  Definition 13. *Let p and p' be two probability distributions on $\mathbb{R}^d$ such that there exists a diffeomorphism φ such that $p' = [\varphi]_* p$. Let $\mathrm{id} : x \mapsto x$ denotes the identity mapping on $\mathbb{R}^d$. The* **displacemenet interpolation** *between p and p' is defined to be the probability path*

  $$p_t := [(1-t)\mathrm{id} + t\varphi]_* p.$$

  In other words, to sample from p_t, one does the following
  - Sample x form p.
  - Compute $x' = \varphi(x)$.
  - Compute $x_t = (1-t)x + tx'$.
  - Return x_t to the user.
- Look at our conditional flow $\psi_t(x) = (1-(1-\sigma_{\mathrm{min}})t)x + tx_{\mathrm{data}}$ again. We note that it transforms a Gaussian distribution $p_0 = \mathcal{N}(0, I)$ to another Gaussian distribution $p_1 = \mathcal{N}(x_{\mathrm{data}}, \sigma_{\mathrm{min}}^2 I)$. Moreover, we can write it as a displacement interpolant between p_0 and p_1 because we can define

  $$\varphi(x) = \sigma_{\mathrm{min}} x + x_{\mathrm{data}}.$$

  It is clear that φ is a diffeomorphism if $\sigma_{\mathrm{min}} \neq 0$. It is also clear that $p_1 = [\varphi]_* p_0$. Lastly,

  $$\psi_t(x) = (1-(1-\sigma_{\mathrm{min}})t)x + tx_{\mathrm{data}} = (1-t)x + t(\sigma_{\mathrm{min}} x + x_{\mathrm{data}}) = (1-t)x + t\varphi(x).$$

  So, indeed, ψ_t is a displacement interpolation between two Gaussians.
- Where does optimal transport come from then?
- First, let us understand what an optimal transport problem is.

  Definition 14. *Given two probability distributions p and p' on $\mathbb{R}^d$, a diffeomorphism $f : \mathbb{R}^d \to \mathbb{R}$ is called an* **optimal transport plan** *from p to p' if the following two conditions are satisfied.*
  - *p' is the push-forward of p according to f. In other word, $p' = [f]_* p$.*
  - *f is a function that minimizes*

    $$E_{x \sim p}[c(x, f(x))] = \int p(x) c(x, f(x)) \,\mathrm{d}x$$

    for some cost function $c : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+ \cup \{0\}$.
- For the cost function $c(x, y) = \|x - y\|^2$, the optimal transport plan exists and is unique for non-pathological p and p'.

  Theorem 15 (Brenier's [1]). *Let the cost function $c(x, y) = \|x - y\|^2$. Let p and p' be well-behaved probability distributions (i.e., have finite moments and do not assign mass to sets with measure zero). Then, there exists a unique optimal transport plan f from p to p'. Moreover, there exists a convex scalar function $F : \mathbb{R}^d \to \mathbb{R}$ such that $\nabla F = f$.*

I'm using the form of the theorem from a Tweet by Gabriel Peyré [10].

- For two Gaussians, our diffeomorphism is $\varphi(x) = \sigma_{\min} x + x_{\mathrm{data}}$. Let

  $$\Phi(x) = \frac{1}{2} \sum_{i=1}^{d} \left(\sqrt{\sigma_{\min}} x^i + \frac{x^i_{\mathrm{data}}}{\sqrt{\sigma_{\min}}} \right)^2.$$

  We have that

  $$\nabla_i \Phi(x) = \sigma_{\min} x^i + x^i_{\mathrm{data}}.$$

  So,

  $$\nabla \Phi(x) = \begin{bmatrix} \nabla_1 \Phi(x) \\ \nabla_2 \Phi(x) \\ \vdots \\ \nabla_d \Phi(x) \end{bmatrix} = \begin{bmatrix} \sigma_{\min} x^1 + x^1_{\mathrm{data}} \\ \sigma_{\min} x^2 + x^2_{\mathrm{data}} \\ \vdots \\ \sigma_{\min} x^d + x^d_{\mathrm{data}} \end{bmatrix} = \sigma_{\min} x + x_{\mathrm{data}} = \varphi(x).$$

  By Brenier's theorem, φ is the unique optimal transport plan that minimizes $\int p(x)\|x - \varphi(x)\|^2 \, \mathrm{d}x$.

- So, when the paper says ϕ_t is the "optimal transport displacement map" between two Gaussians, it means that
  - There exists the unique optimal transport map φ from the Gaussian $p_0 = \mathcal{N}(0, I)$ to another Gaussian $p_1 = \mathcal{N}(x_{\mathrm{data}}, \sigma^2_{\min} I)$.
  - ϕ_t is the displacement interpolation between p_0 and p_1 with respect to φ.

- The conditional flow $\phi_t(x) = (1 - (1 - \sigma_{\min})t)x + t x_{\mathrm{data}}$ has nice properties.
  - These properties are kind of evident by the form of the function itself. Don't be fooled by the fact that ϕ_t is "optimal transport" or anything. I think it's just a marketing gimmick to make it sound more impressive than it is.
  - The first property is that, according to the flow, a particle that starts from x moves in a straight line from x towards x_{data}.
    * This is also true for variance-preserving diffusion conditional flows.
  - The second property is that the velocity at which each particle moves is constant throughout the movement duration. A particle that starts at x always move at velocity $x_{\mathrm{data}} - (1 - \sigma_{\min})x$.
    * This property is not true for diffusion conditional flows in general.

- The paper claims that "sampling trajectory from diffusion paths can 'overshoot' the final sample, resulting in unnecessary backtracking, whilst the OT paths are guaranteed to stay straight."
  - WTF are the authors talking about?
  - Diffusion conditional paths are straight too. They don't overshoot whatsoever. See Figure 2. Every paths are straight.
  - If they talk about unconditonal sampling paths, then there is no gaurantee that the uncondional paths according to this formulation is going to be straight. See the rectified flow paper [8].

- The paper claims that "An interesting observation is that the OT VF has a constant direction in time, which arguably leads to a simpler regression task."
  - This is plausible, but take it with grain of salt.

- Also, keep in main that "we note that although the conditional flow is optimal, this by no means imply that the marginal VT is an optimal transport solution."
  - The rectified flow paper [8] has more things to say in terms of optimal transport properties of the marginal VT.

4 Experiments

- The paper trains generative models on the following datasets.
  - CIFAR-10 at 32×32.
  - ImageNet at 32×32, 64×64 and 128×128.
- The following generative models were trained:
  - Vanilla DDPM [3]. I believe this is a noise-predicting network with linear-β schedule.
  - Score matching [12]. I believe this is a diffusion model with variance-exploding noise schedule.
  - ScoreFlow [11]. I have not read this paper, so I don't have a good idea what it is about.
  - Flow matching with diffusion conditional path (variance preserving).
  - Flow matching with optimal transport conditional path.
- It seems all the generative models use the ADM architecture because the hyperparameter table is that of the ADM architecture.
- Evaluation metrics.
  - FID
  - NLL = natgative log likelihood
  - NFE = "number of function evaluations for an adaptive solver to reach its prespecified numerical tolerance."
    * Actually, I don't know exactly what this means. This is the first time I see this metric.
    * My guess is that, for a solver like Euler's method, the output converges as you increase the number of steps.
    * This might measure the number of steps N such that the difference between a sample generated with N steps and another sample generated with $N+1$ step is below a certain threshold.
    * This metrics would take a very long time to evaluate.
- In Table 1 of the paper, it seems that FM with OT conditional path beats all other generative models in all metrics for all datsets.
- Paper comments that flow matching models converge much faster than other models on ImageNet 64.
- The paper presents evidence that flow matching models attent higher FID scores with lower NFEs. (Figure 7.)
- Lastly, the paper trains an image super-resolution model with flow matching and compares it to SR3. They found that they beated SR3 on the FID score but not image similarity metrics (PSNR and SSIM).

A Continuity Equation

- The continuity equation is an equation from fluid dynamics that shows up a lot in fields that involves transport pheonomena.
- Here, we start with a probability density function p_0, which tells us how the probability mass is distributed over $\mathbb{R}^d$. Then, we have a vector field v_t that gives us the time-dependent velocity field that governs how the mass at each point should move. The velocity field generates a probability path p_t.

- Note that, because the velocity field just move the mass around, there is no new mass added or no new mass being dropped. It follows that the total probability mass remains constant. It is *conserved*. This means that p_t is a probability distribution for all t. If you integrate it over $\mathbb{R}^d$, you should get 1.

- Given a probability path p_t and a time-dependent velocity field v_t that generates it, the **flux density** $j : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^d$ is defined as:

$$j_t(x) = p_t(x)v_t(x).$$

  It is just the velocity field weighted by the probability density.

- Let $f : \mathbb{R}^d \to \mathbb{R}^d$ be a vector field. The **divergence** of f is a scalar function $\nabla \cdot f : \mathbb{R}^d \to \mathbb{R}$ define by

$$(\nabla \cdot f)(x) = \sum_{i=1}^{d} \nabla_i f^i(x)$$

  where $f^i(x)$ denotes the ith component of $f(x)$.

- The **continuity equation** of p_t and v_t is given by

$$\frac{\partial}{\partial t} p_t(x) + (\nabla \cdot j_t)(x) = 0.$$

  In the above equation, we view j_t with t fixed as a vector field of signature $\mathbb{R}^d \to \mathbb{R}^d$.

- Let's rewrite the continuity equation with my notation and taking into account correct indexing with no ambiguity whatsoever. We have

$$\nabla_1 p(t, x) + \sum_{i=1}^{d} \nabla_{i+1}(pv^i)(t, x) = 0. \tag{12}$$

- In our context, the continuity equation is useful because of the following theorem.

  Theorem 16. *Let $p : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^+ \cup \{0\}$ be a differentiable probability path. Let $v : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^d$ be a differentiable time-dependent vector field. Then, v_t generates p_t if and only if the continuity equation holds.*

- We shall not prove the theorem in this note, but I will be studying it more.

References

[1] Brenier, Y. Polar factorization and monotone rearrangement of vector-valued functions. *Communications on pure and applied mathematics 44*, 4 (1991), 375–417.

[2] Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. Neural ordinary differential equations, 2019.

[3] Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models, 2020.

[4] Khungurn, P. Neural ordinary differential equations. `https://pkhungurn.github.io/notes/notes/ml/neural-ode/neural-ode.pdf`. Accessed: 2024-07-19.

[5] Khungurn, P. Notation for multivariable derivatives. `https://pkhungurn.github.io/notes/notes/math/multivar-deriv-notations/multivar-deriv-notations.pdf`. Accessed: 2024-07-22.

[6] Khungurn, P. Probability under transformation. https://pkhungurn.github.io/notes/notes/gfx/pdf-transform/pdf-transform.pdf. Accessed: 2024-07-22.

[7] Lipman, Y., Chen, R. T. Q., Ben-Hamu, H., Nickel, M., and Le, M. Flow matching for generative modeling, 2023.

[8] Liu, X., Gong, C., and Liu, Q. Flow straight and fast: Learning to generate and transfer data with rectified flow, 2022.

[9] McCann, R. J. A convexity principle for interacting gases. *Advances in Mathematics 128*, 1 (1997), 153–179.

[10] Peyré, G. Tweet on Brenier's theorem. https://twitter.com/gabrielpeyre/status/1609066824238354434. Accessed: 2024-07-24.

[11] Song, Y., Durkan, C., Murray, I., and Ermon, S. Maximum likelihood training of score-based diffusion models, 2021.

[12] Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations, 2021.