Introduction to Flow Matching

23 August 2024

Flow matching (FM) is a recent generative modelling paradigm which enables scalable training of Continuous Normalising Flows (CNFs). Flow matching models learn a mapping between source and target distributions.

Learning Objectives

1. Define generative modelling within a probabilistic framework.
2. Explain the formulation and training of (finite and continuous) Normalizing Flows.
3. State the advantages of using flow matching and conditional flow matching objectives in training.
4. Understand Gaussian probability paths and optimal transport as simple variants of FM.

Generative Modelling

Assume we have data samples $y_1, y_2, \ldots y_3$ from a distribution of interest $q(y)$ whose density is unknown. Generative models use these samples to learn a probabilistic model approximating q - the *target distribution*. This enables efficient generation of new samples from (an approximation of) q, prediction of the likelihood of future events (density estimation) or inference of latent variables which allow for dimensionality reduction of data.

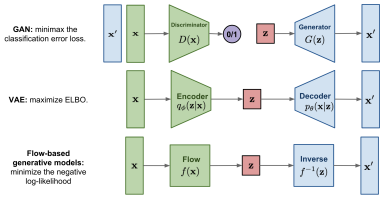


Figure 1: Comparison of three categories of generative models..

Flows as Generative Models

One popular choice of generative models are *flows*. Flow based models start with an arbitrary source distribution $p(x)$ where $x \in \mathbb{R}^d$ and draw a sample from it $x_0 \sim p$. Their goal is to learn a map Φ such that $\Phi(x_0) \sim q$.

(Finite) Normalizing Flows

The key idea underlying Normalizing Flows is to transform a simple distribution to a more expressive distribution which approximates the data using a composition of successive bijective (invertible) transformations. Flowing through a chain of transformations, variables are repeatedly substituted for new variables according to the *change of variable theorem* to obtain a probability distribution for the final target variable.

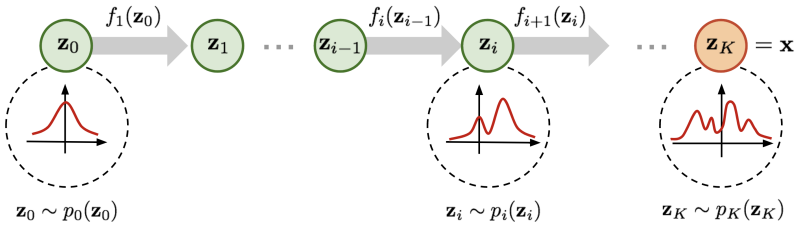


Figure 2: Illustration of a normalizing flow model, transforming a simple distribution p_0 to a complex one p_k step by step.

Change of Variable Theorem

Given a random variable z and its known probability density function $z \sim \pi(x)$, we would like to construct a new random variable using a 1-1 mapping function $x = f(z)$. The function f is invertible, hence $z = f^{-1}(x)$. The unknown probability density function of the new variable $p(x)$ can be inferred by the change of variable theorem. For $z \in \mathbb{R}^1$ and $x \in \mathbb{R}^1$ the change of variable theorem is derived as follows:

$$\int p(x)dx = \int \pi(z)dz = 1$$
$$p(x) = \pi(z)\left|\frac{dz}{dx}\right|$$
$$= \pi(f^{-1}(x))\left|\frac{df^{-1}}{dx}\right|$$

The multivariate version has a similar format:

$$\mathbf{z} \sim \pi(\mathbf{z}), \mathbf{x} = f(\mathbf{z}), \mathbf{z} = f^{-1}(\mathbf{x})$$
$$p(\mathbf{x}) = \pi(\mathbf{z})\left|\det \frac{d\mathbf{z}}{d\mathbf{x}}\right|$$
$$= \pi(f^{-1}(\mathbf{x}))\left|\det \frac{df^{-1}}{d\mathbf{x}}\right|$$

where $\det \frac{d\mathbf{z}}{d\mathbf{x}}$ is the determinant of the Jacobian of function f.

Training Normalizing Flows

Considering a single transformation step of the normalizing flow in Fig. 2 (from $\mathbf{z}_{i-1} \sim p_{i-1}$ to $\mathbf{z}_i \sim p_i$), the change of variable theorem

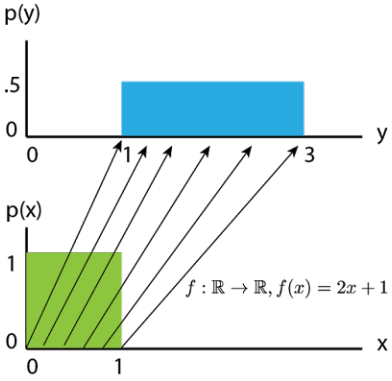


Figure 3: Intuition for the change of variables rule operator by examining linear transformations of 1D random variables. Let $X \sim Uniform(0,1)$ and $Y = f(X) = 2X + 1$. Y is a affine transformation of the underlying "source distribution" X. The green square represents the shaded probability mass on $\mathbb{R}$ for both $p(x)$ and $p(y)$. Since probability mass must integrate to 1 for any distribution, the act of scaling the domain by 2 means we must divide the probability density by 2 everywhere i.e. $p(x)dx = p(y)dy = 1 \implies p(y) = p(x)|\frac{dx}{dy}|$.

can be applied as follows:

$$\begin{aligned}
\mathbf{z}_{i-1} &\sim p_{i-1}(\mathbf{z}_{i-1}) \\
\mathbf{z}_i = f_i(\mathbf{z}_{i-1}) \implies \mathbf{z}_{i-1} &= f_i^{-1}(\mathbf{z}_i) \\
p_i(\mathbf{z}_i) &= p_{i-1}(f_i^{-1}(\mathbf{z}_i)) \left| \det \frac{df_i^{-1}}{d\mathbf{z}_i} \right| \\
&= p_{i-1}(\mathbf{z}_{i-1}) \left| \det \left(\frac{df_i}{d\mathbf{z}_{i-1}} \right)^{-1} \right| \\
&= p_{i-1}(\mathbf{z}_{i-1}) \left| \det(\frac{df_i}{d\mathbf{z}_{i-1}} \right|^{-1} \\
log(p_i(\mathbf{z}_i)) &= \log p_{i-1}(\mathbf{z}_{i-1}) - \log \left| \det \frac{df_i}{d\mathbf{z}_{i-1}} \right|
\end{aligned}$$

Chaining together multiple such transformations to obtain the output distribution from the input to the normalizing flow model we obtain:

$$\mathbf{x} = \mathbf{z}_K = f_K \circ f_{K-1} \circ \cdots \circ f_1(\mathbf{z}_0)$$

The corresponding log-likelihood of observing a training data point under the learnt distribution is:

$$\begin{aligned}
\log p(\mathbf{x}) = \log p_K(\mathbf{z}_K) &= \log p_{K-1}(\mathbf{z}_{K-1}) - \log \left| \det \frac{df_K}{d\mathbf{z}_{K-1}} \right| \\
&= \log p_{K-2}(\mathbf{z}_{K-2}) - \log \left| \det \frac{df_{K-1}}{d\mathbf{z}_{K-2}} \right| - \log \left| \det \frac{df_K}{d\mathbf{z}_{K-1}} \right| \\
&= \dots \\
&= \log p_0(\mathbf{z}_0) - \sum_{i=1}^{K} \log \left| \det \frac{df_i}{d\mathbf{z}_{i-1}} \right|
\end{aligned}$$

The negative log-likelihood over the training dataset $\mathcal{D}$ is given by:

$$\mathcal{L}(\mathcal{D}) = -\frac{1}{|\mathcal{D}|} \sum_{\mathbf{x} \in \mathcal{D}} \log p(\mathbf{x})$$

Continuous Normalizing Flows (CNFs)

Consider letting the number of layers (transformations) tend to infinity ($T \to \infty$) in a deep finite normalizing flow

$$x_T = \Phi_T \circ \Phi_{T-1} \circ \cdots \circ \Phi_1(x_0)$$

The result in a *continuous normalizing flows* where

$$x_T = \int_0^T u_t(\Phi_t(x_0))\, dt \tag{1}$$

Notice Eq. (1) is a Neural ODE if the vector field u_t is parameterized as a neural network.
The definition of CNFs involves the following objects:

Probability density path The time dependent probability density is given by the *probability density path* $p : [0,1] \times \mathbb{R}^d \to \mathbb{R}_{>0}$ where $\int p_t(x)dx = 1$.

Flow A time-dependent diffeomorphic map (roughly a differentiable map with a differentiable inverse) $\Phi : [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$.

Velocity field Flows choose to represent the map Φ in terms of a *time dependent velocity field* $u_t(x)$:

$$u : [0,1] \times x \in \mathbb{R}^d \to \mathbb{R}^d$$

The velocity field $u_t(x)$ defines the flow in terms of a ordinary differential equation (ODE):

$$\frac{d}{dt}\Phi_t(x_0) = u_t(\Phi_t(x_0)) \tag{2}$$

$$\Phi_0(x_0) = x_0 \tag{3}$$

Continuity equation The time dependent probability density p_t is linked to the vector field u_t by the continuity equation:

$$\frac{\partial p}{\partial t} = -\nabla \cdot (p_t u_t) \tag{4}$$

The flow Φ_t reshapes a simple prior density p_0 to a more complicated one, p_t, via the push-forward equation:

$$p_t = [\Phi_t]_* p_0 \tag{5}$$

where the push forward (or change of variable) operator $*$ is defined by:

$$[\Phi_t]_* p_0(x) = p_0(\Phi_t^{-1}(x)) \cdot det \left[\frac{\partial \Phi_t^{-1}}{\partial x}(x)\right] \tag{6}$$

The probability density is linked to the vector field by the continuity eq

The probability density at each time step can be calculated using the change of variables as discussed above for finite NFs:

$$\log p_T = \log p_0(x_0) - \sum_{t=1}^{T} \log \left|\det \frac{d\Phi_t}{dx_{t-1}}\right| \tag{7}$$

An advantage of CNFs is that we can tractably estimate the log-determinant of the Jacobian using Jacobi's identity.

$$\log p_T = \log p_0(x_0) - \int_{t=0}^{t=1} Tr(\frac{\partial v}{\partial x_t})\, dt \tag{8}$$

However, this still involves an ODE solve and backpropagation through the solver at each training iteration to find the change in log probability along the path which makes training difficult and slow.

Flow Matching

Flow Matching Objective

Flow Matching proposes a more scalable loss for training flows.

Let x denote a random variable distributed according to some unknown data distribution $q(x)$. We assume we only have access to data samples from $q(x)$ but have no access to the density function itself. Furthermore, we let p_t be a probability path such that $p_0 = p$ is a simple distribution, e.g the standard normal distribution $p(x) = N(x|0, I)$, and let p_1 be approximately equal in distribution to q. The Flow Matching objective is then designed to match this target probability path, which will allows us to flow from p_0 to p_1

Given a target probability density path $p_t(x)$ and a corresponding vector field $u_t(x)$, which generates $p_t(x)$, the Flow Matching objective is defined as

$$\mathcal{L}_{FM}(\theta) = \mathbb{E}_{t\sim[0,1],x\sim p_t(x)} \|v_{\theta,t}(x) - u_t(x)\|^2, \tag{9}$$

where θ denotes the learnable parameters of the CNF vector field $v_{\theta,t}$, $t \sim U[0,1]$ (uniform distribution), and $x \sim p_t(x)$. The FM loss regresses the vector field u_t with a neural network $v_{\theta,t}$. Upon reaching zero loss, the learned CNF model will generate $p_t(x)$.

- There are many choices of probability paths p_t that can satisfy $p_1(x) \approx q(x)$, how do we know which one to use?
- Generally, we don't have access to a closed form u_t that generates the desired p_t. What u_t should be used?

Constructing p_t and u_t

Flow Matching constructs both p_t and u_t using probability paths and vector fields that are only defined *per sample*, and an appropriate method of aggregation provides the desired p_t and u_t (between the full source and target distributions).

A simple way to construct a target probability path is via a mixture of simpler probability paths: Given a particular data sample x^1 we denote by $p_t(x|x^1)$ a *conditional probability path* such that it satisfies $p_0(x|x^1) = p(x)$ at time $t = 0$. We design $p_1(x|x^1)$ at $t = 1$ to be a distribution concentrated around $x = x^1$, e.g. $p_1(x|x^1) = N(x|x^1, \sigma^2 I)$,

a normal distribution with x^1 mean and a sufficiently small standard deviation $\sigma > 0$.

Marginalizing the conditional probability paths over $q(x^1)$ gives rise to *the marginal probability path*

$$p_t(x) = \int p_t(x|x^1)q(x^1)dx^1, \tag{10}$$

where in particular at time $t = 1$, the marginal probability p_1 is a mixture distribution that closely approximates the data distribution q,

$$p_1(x) = \int p_1(x|x^1)q(x^1)dx^1 \approx q(x). \tag{11}$$

Similarly, the *marginal vector field* can also be defined by "marginalizing" over the conditional vector fields.

$$u_t(x) = \int u_t(x|x^1)\frac{p_t(x|x^1)q(x^1)}{p_t(x)}dx^1, \tag{12}$$

where $u_t(\cdot|x^1) : \mathbb{R}^d \to \mathbb{R}^d$ is a conditional vector field that generates $p_t(\cdot|x^1)$.

The proof for this is under Theorem 1 in Appendix A.

Conditional Flow Matching

Due to the intractable integrals in the definitions of the marginal probability path and vector field, it is still intractable to compute u_t, and consequently the original Flow Matching objective $\mathcal{L}_{FM}$. Instead, a simpler objective which surprisingly will result in the same optima as the original objective - the *Conditional Flow Matching* (CFM) objective is used:

$$\mathcal{L}_{CFM}(\theta) = \mathbb{E}_{t,q(x_1),p_t(x|x_1)}\left\|v_{\theta,t}(x) - u_t(x|x^1)\right\|^2 \tag{13}$$

where $t \sim U[0,1]$, $x_1 \sim q(x_1)$, and now $x \sim p_t(x|x_1)$. Unlike the FM objective, the CFM objective allows sampling of unbiased estimates as long as we can efficiently sample from $p_t(x|x_1)$ and compute $u_t(x|x_1)$, both of which can be easily done as they are defined on a per-sample basis.
Optimizing the CFM objective is equivalent (in expectation) to optimizing the FM objective:

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

Completing the square in both losses we get:

$$\|v_\theta(t,x) - u_t(x \mid x^1)\|^2 = \|v_\theta(t,x)\|^2 + \|u_t(x \mid x^1)\|^2 - 2\langle v_\theta(t,x), u_t(x \mid x^1)\rangle \tag{14}$$

and

$$\|v_\theta(t,x) - u_t(x)\|^2 = \|u_\theta(t,x)\|^2 + \|u_t(x)\|^2 - 2\langle v_\theta(t,x), u_t(x)\rangle \quad (15)$$

Taking the expectation over the last inner product term:

$$\mathbb{E}_{x\sim p_t}\ \langle u_\theta(t,x), u_t(x)\rangle = \int \langle u_\theta(t,x), \int u_t(x|x^1)\frac{p_t(x \mid x^1)q(x^1)}{p_t(x)}dx^1\rangle p_t(x)dx \quad (16)$$

$$= \int \langle u_\theta(t,x), \int u_t(x \mid x^1)p_t(x \mid x^1)q(x^1)dx^1\rangle dx \quad (17)$$

$$= \int\int \langle u_\theta(t,x), u_t(x \mid x^1)\rangle p_t(x \mid x^1)q(x^1)dx^1 dx \quad (18)$$

$$= \mathbb{E}_{q_1(x^1)p(x|x^1)}\ \langle u_\theta(t,x), u_t(x \mid x^1)\rangle \quad (19)$$

This allows us to train a CNF to generate the marginal probability path p_t - which in particular, approximates the unknown data distribution q at t=1 - without ever needing access to either the marginal probability path or the marginal vector field.

In summary, the conditional flow matching algorithm is implemented as follows:

Algorithm 1 Conditional Flow Matching

Input: dataset q, noise p
Initialize v^θ
while not converged **do**
  $t \sim U([0,1])$
  $x_1 \sim q(x_1)$
  $x_0 \sim p(x_0)$
  $x_t \sim \Phi_t(x_0|x_1)$
  Gradient step with $\nabla_\theta \|v_t^\theta(x_t) - \partial_t \Phi_t\|^2$
end while
return v_θ

Gaussian Conditional Probability Paths

The Conditional Flow Matching objective works with any choice of conditional probability path and conditional vector fields. For simplicity, consider the construction of $p_t(x|x^1)$ and $u_t(x|x^1)$ for a general family of Gaussian conditional probability paths:

$$p_t(x|x^1) = N(x \mid \mu_t(x^1), \sigma_t(x^1)^2 I) \quad (20)$$

where $\mu : [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$ is the time-dependent mean of the Gaussian distribution, while $\sigma : [0,1] \times \mathbb{R}^d \to \mathbb{R}^d_{>0}$ describes a time-dependent scalar standard deviation (std).

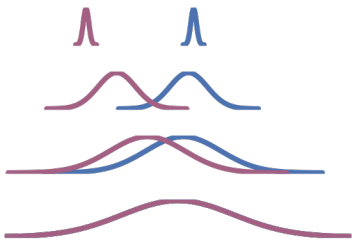


Figure 4: Gaussian probability paths with $\mu_0(x^1) = 0$ and $\sigma_0(x^1) = 1$ i.e. a standard Gaussian noise distribution at $t = 0$ and $\mu_1(x^1) = x^1$ and $\sigma_1(x^1) = \sigma_{min}$ at $t = 1$.

Set $\mu_0(x^1) = 0$ and $\sigma_0(x^1) = 1$, so that all conditional probability paths converge to the same standard Gaussian noise distribution at $t = 0$, $p(x) = N(x|0, I)$. Set $\mu_1(x^1) = x^1$ and $\sigma_1(x^1) = \sigma_{\text{min}}$, which is set sufficiently small so that $p^1(x|x^1)$ is a concentrated Gaussian distribution centered at x^1.

There exist an infinite number of vector fields that generate any particular probability path. The simplest vector field corresponding to a canonical transformation for Gaussian distributions (conditioned on x^1) is used:

$$\psi_t(x) = \sigma_t(x^1)x + \mu_t(x^1). \tag{21}$$

When x is distributed as a standard Gaussian, $\psi_t(x)$ is the affine transformation that maps to a normally-distributed random variable with mean $\mu_t(x^1)$ and std $\sigma_t(x^1)$ i.e. ψ_t pushes the noise distribution $p_0(x|x^1) = p(x)$ to $p_t(x|x^1)$:

$$[\psi_t]_* p(x) = p_t(x|x^1). \tag{22}$$

This flow then provides a vector field that generates the conditional probability path:

$$\frac{d}{dt}\psi_t(x) = u_t(\psi_t(x)|x^1). \tag{23}$$

Reparameterizing $p_t(x|x^1)$ in terms of just x_0 and plugging Eq. (23) in the CFM loss:

$$L_{CFM}(\theta) = \mathbb{E}_{t,q(x_1),p(x_0)}\|v_t(\psi_t(x_0)) - \frac{d}{dt}\psi_t(x_0)\|^2. \tag{24}$$

Since ψ_t is a simple (invertible) affine map we can use Eq. (23) to solve for u_t in a closed form. Let $f' = \frac{d}{dt}f$, for a time-dependent function f. Let $p_t(x|x_1)$ be a Gaussian probability path as in Eq. (20), and ψ_t its corresponding flow map as in Eq. (21). Then, the unique vector field that defines ψ_t has the form (derivation in Appendix A, Theorem 3):

$$u_t(x|x^1) = \frac{\sigma_t'(x^1)}{\sigma_t(x^1)}(x - \mu_t(x^1)) + \mu_t'(x^1) \tag{25}$$

Consequently, $u_t(x|x^1)$ generates the Gaussian path $p_t(x|x^1)$.

Optimal Transport Conditional Probability Paths [1]

Optimal transport involves defining the mean and the standard deviation to change linearly in time:

$$\mu_t(x) = tx_1, \text{ and } \sigma_t(x) = 1 - (1 - \sigma_{min})t. \tag{26}$$

According to Eq. (25) this path is generated by the vector field

$$u_t(x|x_1) = \frac{x_1 - (1 - \sigma_{min})x}{1 - (1 - \sigma_{min})t}, \tag{27}$$

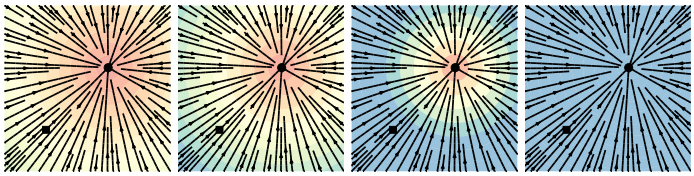
[1] Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport, 2024. URL https://arxiv.org/abs/2302.00482

The conditional flow that corresponds to $u_t(x|x_1)$ is

$$\psi_t(x) = (1 - (1 - \sigma_{min})t)x + tx_1 \tag{28}$$

and in this case, the CFM loss (see Eq. (13), Eq. (24)) takes the form:

$$L_{CFM}(\theta) = E_{t,q(x_1),p(x_0)}\|v_t(\psi_t(x_0)) - (x_1 - (1 - \sigma_{\min})x_0)\|^2. \tag{29}$$



$t = 0.0$ $t = 1/3$ $t = 2/3$ $t = 1.0$

Figure 9: The OT path's conditional vector field has constant direction in time (the blue color denotes larger magnitude while red color denotes smaller magnitude).

Appendix (reproduced from [2])

[2] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023. URL https://arxiv.org/abs/2210.02747

Appendix A: Theorem Proofs

Theorem 1: *Given vector fields $u_t(x|x_1)$ that generate conditional probability paths $p_t(x|x_1)$, for any distribution $q(x_1)$, the marginal vector field u_t in equation 30 generates the marginal probability path p_t in equation 31, i.e., u_t and p_t satisfy the continuity equation (equation 34).*

$$u_t(x) = \int u_t(x|x_1) \frac{p_t(x|x_1)q(x_1)}{p_t(x)} dx_1, \tag{30}$$

Proof. To verify this, we check that p_t and u_t satisfy the continuity equation (equation 34):

$$\begin{aligned}\frac{d}{dt}p_t(x) &= \int \Big(\frac{d}{dt}p_t(x|x_1)\Big)q(x_1)dx_1 = -\int \mathrm{div}\Big(u_t(x|x_1)p_t(x|x_1)\Big)q(x_1)dx_1 \\ &= -\mathrm{div}\Big(\int u_t(x|x_1)p_t(x|x_1)q(x_1)dx_1\Big) = -\mathrm{div}\Big(u_t(x)p_t(x)\Big),\end{aligned}$$

where in the second equality we used the fact that $u_t(\cdot|x_1)$ generates $p_t(\cdot|x_1)$, in the last equality we used equation 30. Furthermore, the first and third equalities are justified by assuming the integrands satisfy the regularity conditions of the Leibniz Rule (for exchanging integration and differentiation).

Theorem 2: *Assuming that $p_t(x) > 0$ for all $x \in \mathbb{R}^d$ and $t \in [0,1]$, then, up to a constant independent of θ, $\mathcal{L}_{CFM}$ and $\mathcal{L}_{FM}$ are equal. Hence, $\nabla_\theta \mathcal{L}_{FM}(\theta) = \nabla_\theta \mathcal{L}_{CFM}(\theta)$.*

Proof. To ensure existence of all integrals and to allow the changing of integration order (by Fubini's Theorem) in the following we assume that $q(x)$ and $p_t(x|x_1)$ are decreasing to zero at a sufficient speed as $\|x\| \to \infty$, and that $u_t, v_t, \nabla_\theta v_t$ are bounded.

First, using the standard bilinearity of the 2-norm we have that

$$\begin{aligned}\|v_t(x) - u_t(x)\|^2 &= \|v_t(x)\|^2 - 2\langle v_t(x), u_t(x)\rangle + \|u_t(x)\|^2 \\ \|v_t(x) - u_t(x|x_1)\|^2 &= \|v_t(x)\|^2 - 2\langle v_t(x), u_t(x|x_1)\rangle + \|u_t(x|x_1)\|^2\end{aligned}$$

Next, remember that u_t is independent of θ and note that

$$\begin{aligned}\mathbb{E}_{p_t(x)}\|v_t(x)\|^2 &= \int \|v_t(x)\|^2 p_t(x)dx = \int \|v_t(x)\|^2 p_t(x|x_1)q(x_1)dx_1dx \\ &= \mathbb{E}_{q(x_1),p_t(x|x_1)}\|v_t(x)\|^2,\end{aligned}$$

where in the second equality we use equation 31, and in the third equality we change the order of integration.

$$p_t(x) = \int p_t(x|x_1)q(x_1)dx_1, \tag{31}$$

Next,

$$\begin{aligned}\mathbb{E}_{p_t(x)}\left\langle v_t(x), u_t(x)\right\rangle &= \int \left\langle v_t(x), \frac{\int u_t(x|x_1)p_t(x|x_1)q(x_1)dx_1}{p_t(x)}\right\rangle p_t(x)dx \\ &= \int \left\langle v_t(x), \int u_t(x|x_1)p_t(x|x_1)q(x_1)dx_1\right\rangle dx \\ &= \int \left\langle v_t(x), u_t(x|x_1)\right\rangle p_t(x|x_1)q(x_1)dx_1dx \\ &= \mathbb{E}_{q(x_1),p_t(x|x_1)}\left\langle v_t(x), u_t(x|x_1)\right\rangle,\end{aligned}$$

where in the last equality we change again the order of integration.

Theorem 3: *Let $p_t(x|x_1)$ be a Gaussian probability path as in Eq.* (20)*, and ψ_t its corresponding flow map as in Eq.* (21)*. Then, the unique vector field that defines ψ_t has the form:*

$$u_t(x|x_1) = \frac{\sigma_t'(x_1)}{\sigma_t(x_1)}\left(x - \mu_t(x_1)\right) + \mu_t'(x_1). \tag{32}$$

Consequently, $u_t(x|x_1)$ generates the Gaussian path $p_t(x|x_1)$.

Proof. For notational simplicity let $w_t(x) = u_t(x|x_1)$. Now consider Eq. (1):

$$\frac{d}{dt}\psi_t(x) = w_t(\psi_t(x)).$$

Since ψ_t is invertible (as $\sigma_t(x_1) > 0$) we let $x = \psi^{-1}(y)$ and get

$$\psi_t'(\psi^{-1}(y)) = w_t(y), \tag{33}$$

where we used the apostrophe notation for the derivative to emphasis that ψ_t' is evaluated at $\psi^{-1}(y)$. Now, inverting $\psi_t(x)$ provides

$$\psi_t^{-1}(y) = \frac{y - \mu_t(x_1)}{\sigma_t(x_1)}.$$

Differentiating ψ_t with respect to t gives

$$\psi_t'(x) = \sigma_t'(x_1)x + \mu_t'(x_1).$$

Plugging these last two equations in equation 33 we get

$$w_t(y) = \frac{\sigma_t'(x_1)}{\sigma_t(x_1)}\left(y - \mu_t(x_1)\right) + \mu_t'(x_1)$$

as required.

Appendix B: The continuity equation

One method of testing if a vector field v_t generates a probability path p_t is the continuity equation. It is a Partial Differential Equation

(PDE) providing a necessary and sufficient condition to ensuring that a vector field v_t generates p_t,

$$\frac{d}{dt}p_t(x) + \text{div}(p_t(x)v_t(x)) = 0, \tag{34}$$

where the divergence operator, div, is defined with respect to the spatial variable $x = (x^1, \ldots, x^d)$, i.e., $\text{div} = \sum_{i=1}^{d} \frac{\partial}{\partial x^i}$.

Appendix C: Computing probabilities of the CNF model

We are given an arbitrary data point $x_1 \in \mathbb{R}^d$ and need to compute the model probability at that point, i.e., $p_1(x_1)$. Below we recap how this can be done covering the basic relevant ODEs, the scaling of the divergence computation, taking into account data transformations (e.g., centering of data), and Bits-Per-Dimension computation.

ODE for computing $p_1(x_1)$ The continuity equation with Eq. (1) lead to the instantaneous change of variable [Chen et al., 2019]:

$$\frac{d}{dt}\log p_t(\phi_t(x)) + \text{div}(v_t(\phi_t(x)) = 0.$$

Integrating $t \in [0, 1]$ gives:

$$\log p_1(\phi_1(x)) - \log p_0(\phi_0(x)) = -\int_0^1 \text{div}(v_t(\phi_t(x)))dt \tag{35}$$

Therefore, the log probability can be computed together with the flow trajectory by solving the ODE:

$$\frac{d}{dt}\begin{bmatrix}\phi_t(x)\\ f(t)\end{bmatrix} = \begin{bmatrix}v_t(\phi_t(x))\\ -\text{div}(v_t(\phi_t(x)))\end{bmatrix} \tag{36}$$

Given initial conditions

$$\begin{bmatrix}\phi_0(x)\\ f(0)\end{bmatrix} = \begin{bmatrix}x_0\\ c\end{bmatrix}. \tag{37}$$

the solution $[\phi_t(x), f(t)]^T$ is uniquely defined (up to some mild conditions on the VF v_t). Denote $x_1 = \phi_1(x)$, and according to equation 35,

$$f(1) = c + \log p_1(x_1) - \log p_0(x_0). \tag{38}$$

Now, we are given an arbitrary x_1 and want to compute $p_1(x_1)$. For this end, we will need to solve equation 36 in reverse. That is,

$$\frac{d}{ds}\begin{bmatrix}\phi_{1-s}(x)\\ f(1-s)\end{bmatrix} = \begin{bmatrix}-v_{1-s}(\phi_{1-s}(x))\\ \text{div}(v_{1-s}(\phi_{1-s}(x)))\end{bmatrix} \tag{39}$$

and we solve this equation for $s \in [0,1]$ with the initial conditions at $s = 0$:

$$\begin{bmatrix} \phi_1(x) \\ f(1) \end{bmatrix} = \begin{bmatrix} x_1 \\ 0 \end{bmatrix}. \tag{40}$$

From uniqueness of ODEs, the solution will be identical to the solution of equation 36 with initial conditions in equation 37 where $c = \log p_0(x_0) - \log p_1(x_1)$. This can be seen from equation 38 and setting $f(1) = 0$. Therefore we get that

$$f(0) = \log p_0(x_0) - \log p_1(x_1)$$

and consequently

$$\log p_1(x_1) = \log p_0(x_0) - f(0). \tag{41}$$

To summarize, to compute $p_1(x_1)$ we first solve the ODE in equation 39 with initial conditions in equation 40, and the compute equation 41.

Unbiased estimator to $p_1(x_1)$ Solving equation 39 requires computation of div of VFs in $\mathbb{R}^d$ which is costly. It has been suggested to replace the divergence by the (unbiased) Hutchinson trace estimator,

$$\frac{d}{ds}\begin{bmatrix} \phi_{1-s}(x) \\ \tilde{f}(1-s) \end{bmatrix} = \begin{bmatrix} -v_{1-s}(\phi_{1-s}(x)) \\ z^T D v_{1-s}(\phi_{1-s}(x)) z \end{bmatrix}, \tag{42}$$

where $z \in \mathbb{R}^d$ is a sample from a random variable such that $\mathbb{E} z z^T = I$. Solving the ODE in equation 42 exactly (in practice, with a small controlled error) with initial conditions in equation 40 leads to

$$\begin{aligned}
\mathbb{E}_z\left[\log p_0(x_0) - \tilde{f}(0)\right] &= \log p_0(x_0) - \mathbb{E}_z\left[\tilde{f}(0) - \tilde{f}(1)\right] \\
&= \log p_0(x_0) - \mathbb{E}_z\left[\int_0^1 z^T D v_{1-s}(\phi_{1-s}(x)) z \, ds\right] \\
&= \log p_0(x_0) - \int_0^1 \mathbb{E}_z\left[z^T D v_{1-s}(\phi_{1-s}(x)) z\right] ds \\
&= \log p_0(x_0) - \int_0^1 \operatorname{div}(v_{1-s}(\phi_{1-s}(x))) ds \\
&= \log p_0(x_0) - (f(0) - f(1)) \\
&= \log p_0(x_0) - (\log p_0(x_0) - \log p_1(x_1)) \\
&= \log p_1(x_1),
\end{aligned}$$

where in the third equality we switched order of integration assuming the sufficient condition of Fubini's theorem hold, and in the previous to last equality we used equation 38. Therefore the random variable

$$\log p_0(x_0) - \tilde{f}(0) \tag{43}$$

is an unbiased estimator for $\log p_1(x_1)$. To summarize, for a scalable unbiased estimation of $p_1(x_1)$ we first solve the ODE in equation 42 with initial conditions in equation 40, and then output equation 43.

Transformed data Often, before training our generative model we transform the data, e.g., we scale and/or translate the data. Such a transformation is denoted by $\varphi^{-1} : \mathbb{R}^d \to \mathbb{R}^d$ and our generative model becomes a composition

$$\psi(x) = \varphi \circ \phi(x)$$

where $\phi : \mathbb{R}^d \to \mathbb{R}^d$ is the model we train.

$$p_t = [\phi_t]_* p_0 \tag{44}$$

where the push-forward (or change of variables) operator $*$ is defined by

$$[\phi_t]_* p_0(x) = p_0(\phi_t^{-1}(x)) \det \left[\frac{\partial \phi_t^{-1}}{\partial x}(x) \right]. \tag{45}$$

Given a prior probability p_0 we have that the push forward of this probability under ψ (equation 44 and equation 45) takes the form

$$\begin{aligned} p_1(x) = \psi_* p_0(x) &= p_0(\phi^{-1}(\varphi^{-1}(x))) \det \left[D\phi^{-1}(\varphi^{-1}(x)) \right] \det \left[D\varphi^{-1}(x) \right] \\ &= \left(\phi_* p_0(\varphi^{-1}(x)) \right) \det \left[D\varphi^{-1}(x) \right] \end{aligned}$$

and therefore

$$\log p_1(x) = \log \phi_* p_0(\varphi^{-1}(x)) + \log \det \left[D\varphi^{-1}(x) \right].$$

For images $d = H \times W \times 3$ and we consider a transform ϕ that maps each pixel value from $[-1, 1]$ to $[0, 256]$. Therefore,

$$\varphi(y) = 2^7(y + 1)$$

and

$$\varphi^{-1}(x) = 2^{-7}x - 1$$

For this case we have

$$\log p_1(x) = \log \phi_* p_0(\varphi^{-1}(x)) - 7d \log 2. \tag{46}$$

Bits-Per-Dimension (BPD) computation BPD is defined by

$$\text{BPD} = \mathbb{E}_{x_1} \left[-\frac{\log_2 p_1(x_1)}{d} \right] = \mathbb{E}_{x_1} \left[-\frac{\log p_1(x_1)}{d \log 2} \right] \tag{47}$$

Following equation 46 we get

$$\text{BPD} = -\frac{\log \phi_* p_0(\varphi^{-1}(x))}{d \log 2} + 7$$

and $\log \phi_* p_0(\varphi^{-1}(x))$ is approximated using the unbiased estimator in equation 43 over the transformed data $\varphi^{-1}(x_1)$. Averaging the unbiased estimator on a large test test x_1 provides a good approximation to the test set BPD.

References

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations, 2019. URL https://arxiv.org/abs/1806.07366.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023. URL https://arxiv.org/abs/2210.02747.

Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport, 2024. URL https://arxiv.org/abs/2302.00482.