Carnegie Mellon University

Lecture 5: Flow Matching

Yutong (Kelly) He

10-799 Diffusion & Flow Matching, Jan 22nd, 2026

Many figures derived from Yang Song's https://yang-song.net/blog/2021/score/

Modal aws

Quiz time!

10 minutes

Closed-book

Pen & Paper

💯

If you don't want to stay for the lecture, feel free to leave after submitting your quiz!

Q1. Write out the expression of score function in terms of x and p [1 pts]

Q2. ***(True/False)*** [1 pts]

Score-based models and DDPM are completely unrelated approaches to generative modeling.

◯ True ◯ False

Q3. ***(True/False)*** [1 pts]

Adding noise to data helps score matching work better in low-density regions.

◯ True ◯ False

Q4. Langevin dynamics generates samples by [1 pts]

◯ Directly using chain rule
◯ Iteratively following the score with added noise
◯ Training a discriminator network
◯ Maximizing the likelihood

Q5. Select ALL that are benefits of score matching over maximum likelihood *(Select all that apply)* [1 pts]

☐ Can train models defined by unnormalized probability densities
☐ Guarantees faster training
☐ Guarantees faster sampling
☐ Only requires to predict the gradient of the log density, not the density itself
☐ Always produces better samples

Carnegie Mellon University

Housekeeping Announcements

- Homework 1 is out! https://kellyyutonghe.github.io/10799S26/homework/
  - Q6 (Alternative Parameterization) is now an optional extra credit question!
  - Due date: 1/24 Sat, Late Due date: 1/26 Mon
  - Training models takes time! Start early!
- Office Hours are announced:
  - Kelly is hosting OH
    - In-person: Wednesdays 1:00 PM - 2:00 PM, Gates 8th Floor common area near the printer
    - Virtual: Fridays 11:00 AM - 12:00 PM, Discord
  - Krish is hosting OH Tuesdays 4:00 PM - 5:00 PM, Gates 8th Floor common area near the printer
    - Extra OH this week: Friday 3 – 4 PM same location

Carnegie Mellon University

Diffusion’s way to turn noise into data

Forward process (adding noise)

Data x_0 $t = 0$ | x_1 $t = 1$ | x_2 $t = 2$ | x_3 $t = 3$ | Noise x_4 $t = 4 = T$

Reserve process (denoising)

Carnegie Mellon University

Cat stolen from Chieh-Hsin (Jesse) Lai

Score-based model's way to turn noise into data

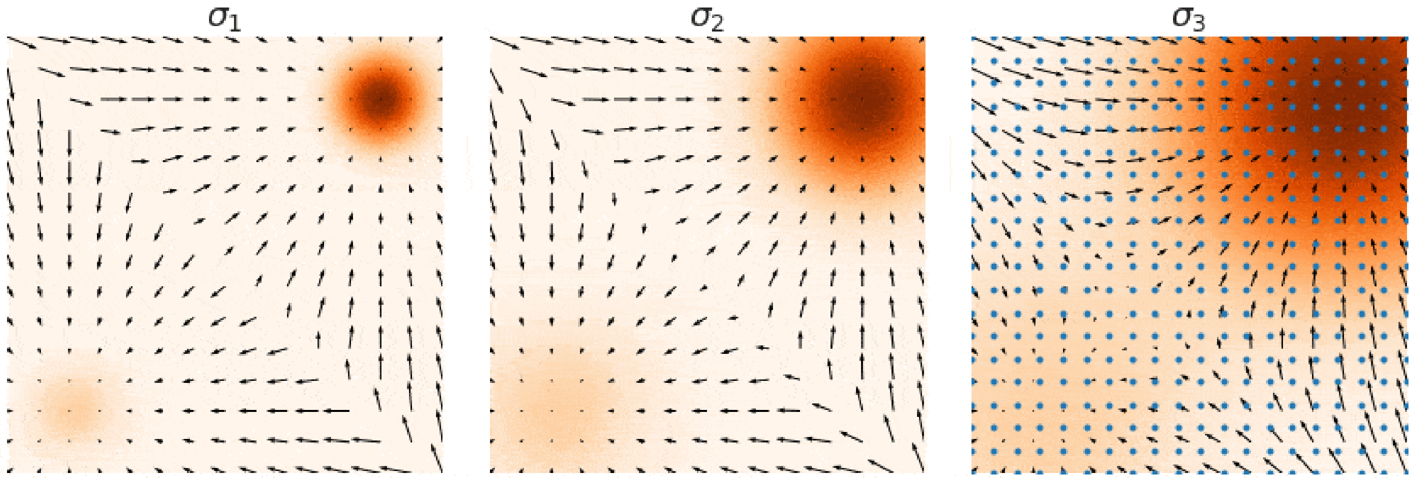
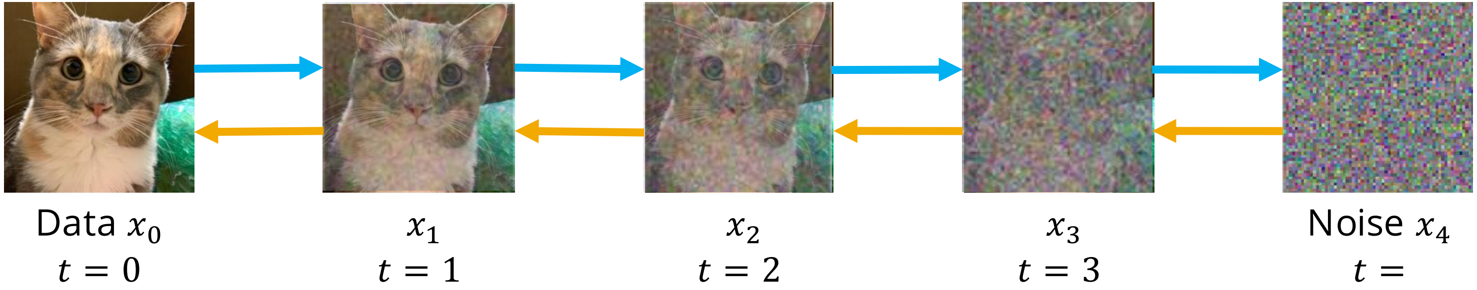


Figure from Yang Song https://yang-song.net/blog/2021/score/

Hold up, wait a minute, doesn't this look familiar?

Diffusion

(DDPM)

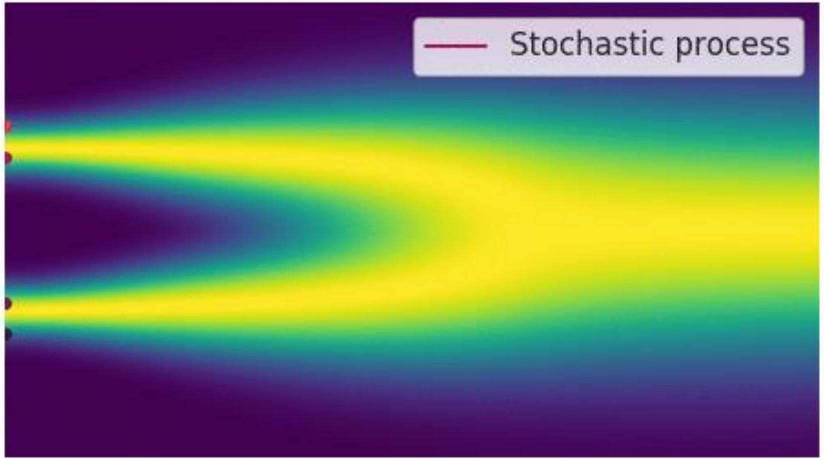


Score-based model

(NCSN)

When the number of noise scales goes to infinity

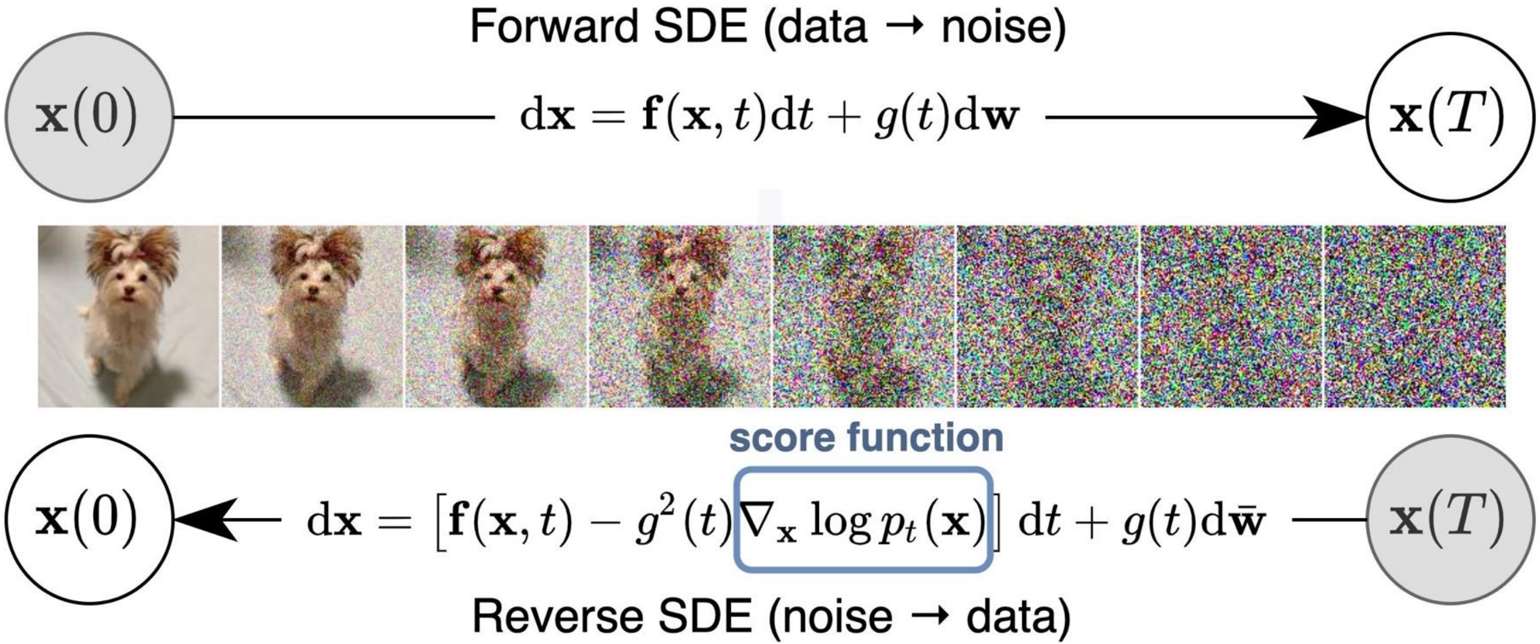
It becomes a **continuous-time stochastic process**, many of which can be solved by **stochastic differential equations** (SDEs)



Carnegie Mellon University

Figure from Yang Song https://yang-song.net/blog/2021/score/

Score SDE: Reverse Process w/ infinite noise scales



Brian D.O. Anderson. “Reverse-time diffusion equation models”. Stochastic Processes and their Applications 1982.

Song et al. “Score-Based Generative Modeling through Stochastic Differential Equations”. ICLR 2021. https://openreview.net/pdf?id=PxTIG12RRHS

Carnegie Mellon University

Is there an even simpler way to do the same thing?

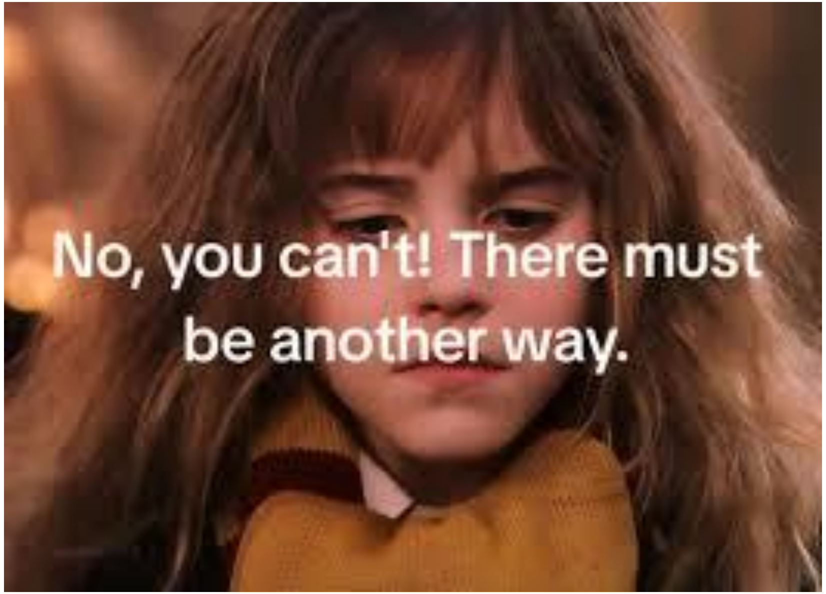


Image from Harry Potter

Let’s say we are given a data point, what would be the simplest way to construct a trajectory from noise to this data point?

How about let’s just do linear interpolation?

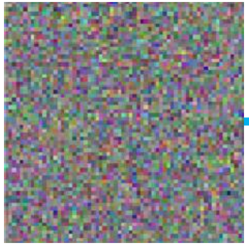
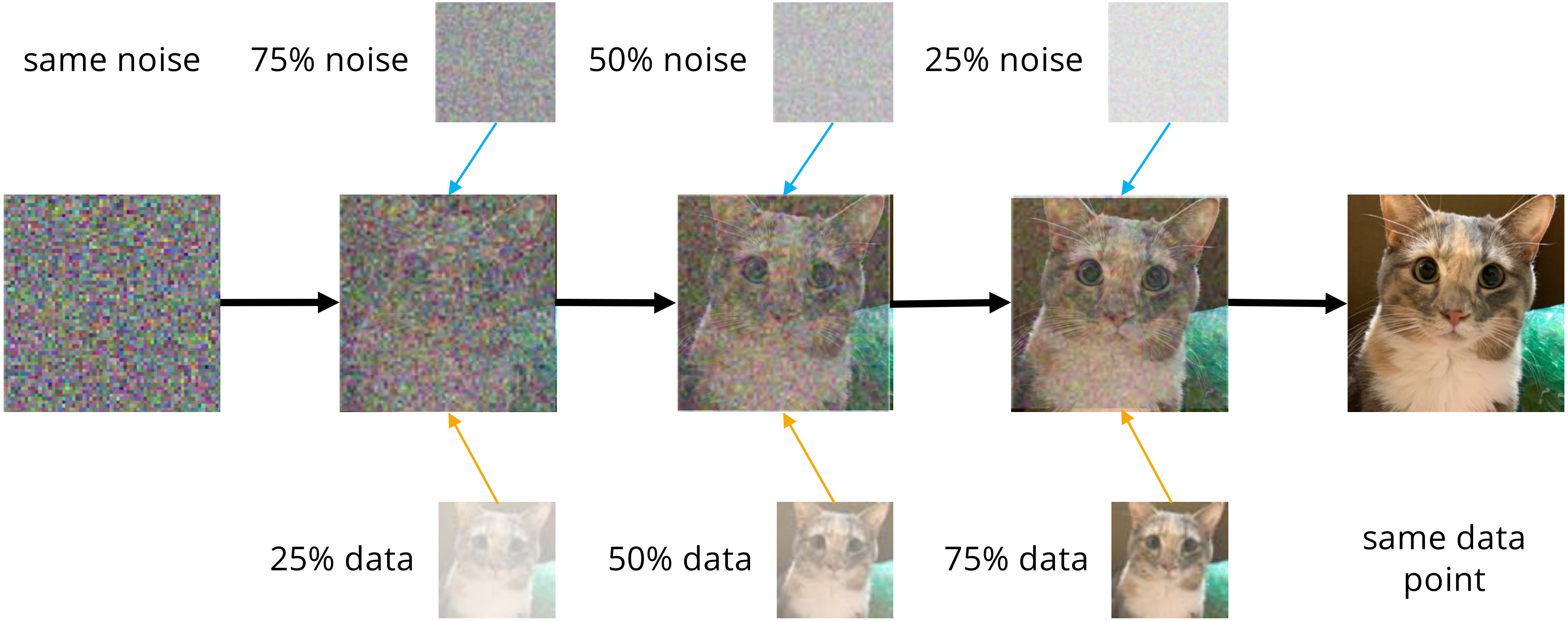


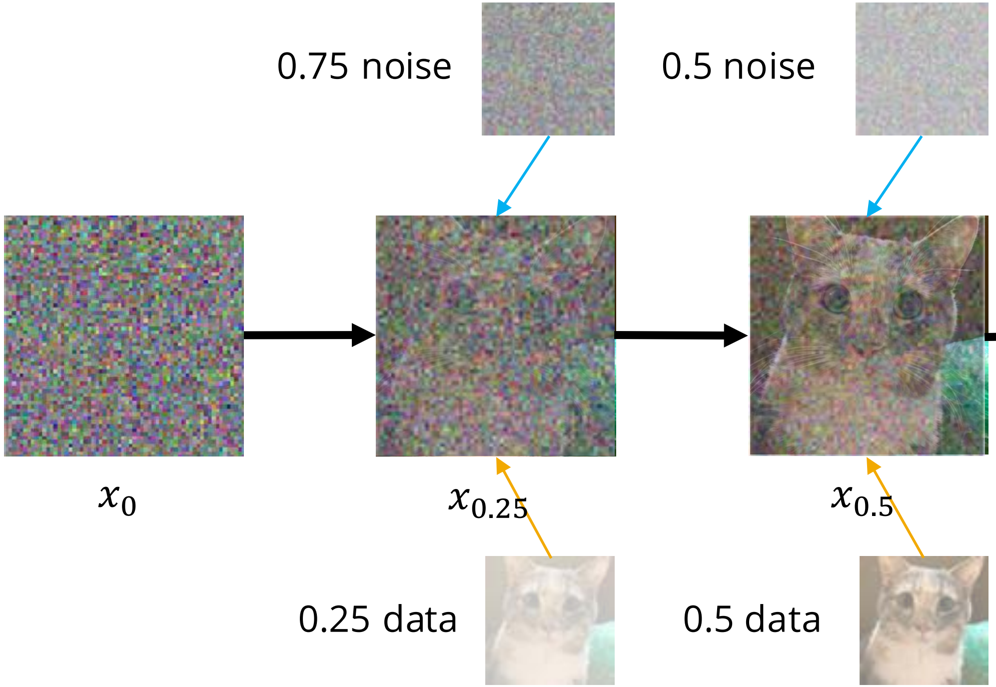
Image from PPAP

How about let's just do linear interpolation?



Then learning the transformations along this trajectory is also easy

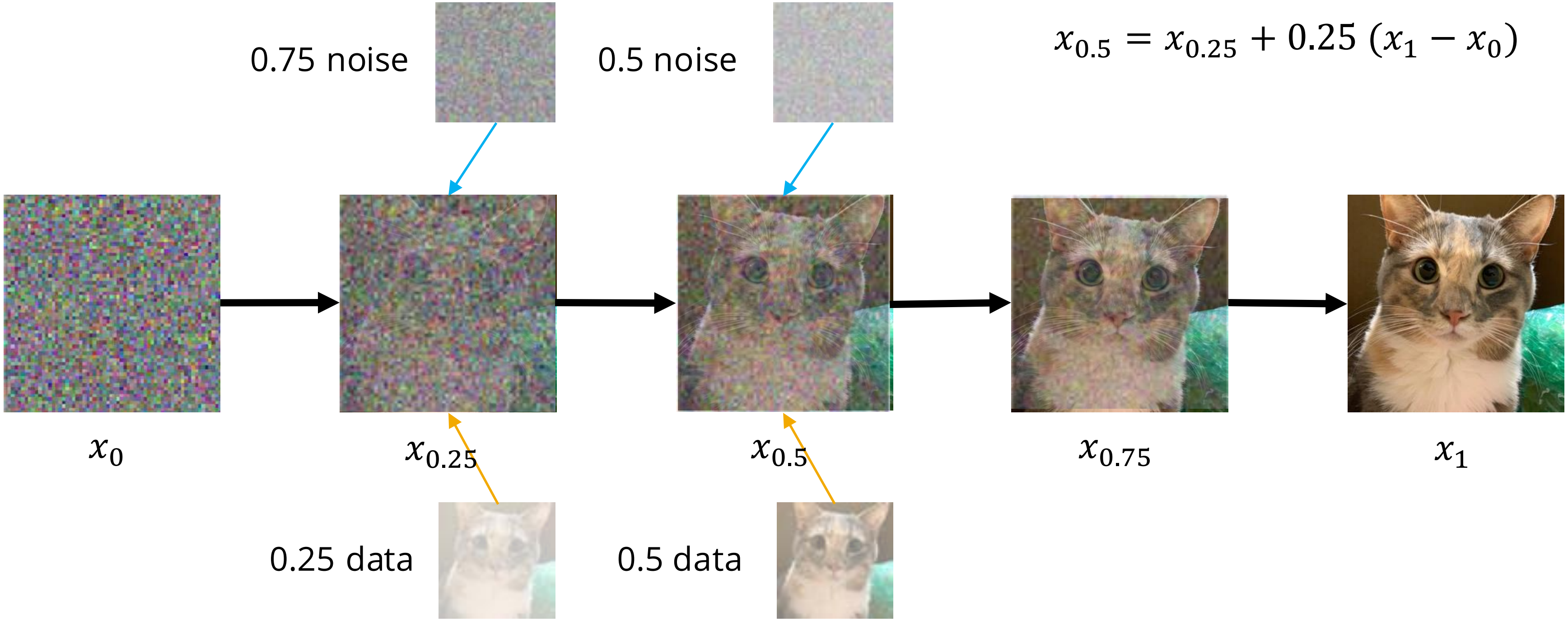
$$x_{0.5} = x_{0.25} + 0.25\ data - 0.25\ noise$$



Carnegie Mellon University

Then learning the transformations along this trajectory is also easy

$x_{0.5} = x_{0.25} + 0.25\,(x_1 - x_0)$



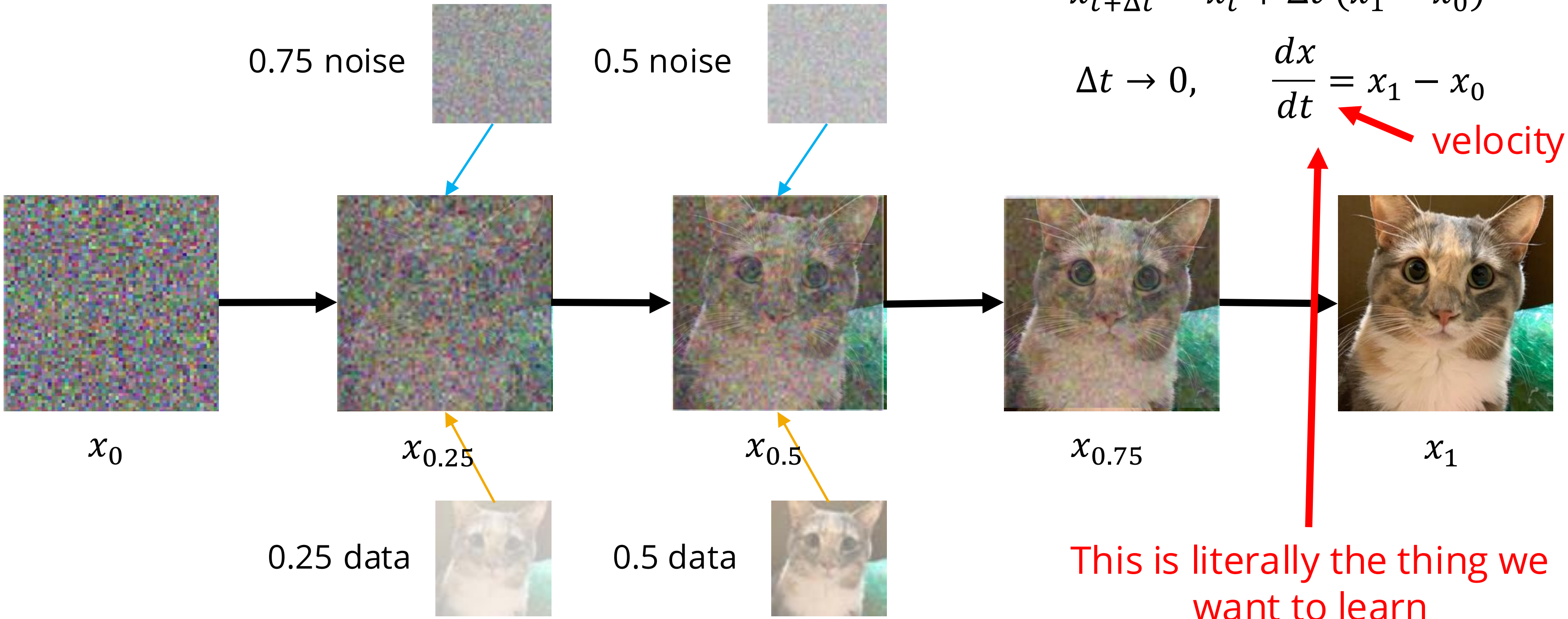
Carnegie Mellon University

Then learning the transformations along this trajectory is also easy

$$x_{t+\Delta t} = x_t + \Delta t\,(x_1 - x_0)$$

$$\Delta t \to 0, \qquad \frac{dx}{dt} = x_1 - x_0$$

velocity



This is literally the thing we want to learn

Learning to transform noise ”straight” into data

Training:

1. Sample noise $x_0 \sim N(0, I)$
2. Sample data $x_0 \sim p_{data}$
3. Uniformly sample time step $t \sim U(0,1)$
4. Compute noisy sample $x_t = tx_1 + (1-t)x_0$
5. Compute velocity $v = x_1 - x_0$
6. Learn to predict the velocity

$$L(\theta) = E[||v_\theta(x_t, t) - v||\hat{}2]$$

Sampling:

Using step size Δt, starting from $t = 0$

1. Sample noise $x_0 \sim N(0, I)$
2. While $t < 1$, do
   1) $\Delta x = v_\theta(x_t, t)\Delta t$
   2) $x_{t+\Delta t} = x_t + \Delta x$
   3) $t = t + \Delta t$
3. Output x_1

Now you have flow matching!

Why is this a proper probabilistic generative model?

To understand this, we need to go back in time
(pun intended)

Carnegie Mellon University

GIF from Star Trek

Continuous normalizing flows

A CNF is a generative model that transports data from an initial distribution (denoted as p_0) to a target distribution (denoted as p_1) by integrating an ODE.

$$\frac{dx_t}{dt} = v(x_t, t)$$

velocity

$$x_t = x_0 + \int_0^t v(x_\tau, \tau) d\tau$$

sampling

Chen et al. Neural Ordinary Differential Equations. NeurIPS 2018. https://arxiv.org/abs/1806.07366

Basically this but with generalized velocities

$$\Delta t \to 0, \qquad \frac{dx}{dt} = v(x_t, t)$$

velocity

x_0 $\quad x_{0.25}$ $\quad x_{0.5}$ $\quad x_{0.75}$ $\quad x_1$

Sampling (Numerically solving of the ODE):

Using step size Δt, starting from $t = 0$

1. Sample noise $x_0 \sim N(0, I)$
2. While $t < 1$, do: $\Delta x = v_\theta(x_t, t)\Delta t, x_{t+\Delta t} = x_t + \Delta x, t = t + \Delta t$
3. Output x_1

Carnegie Mellon University

Think of it as the wind flow transports water vapor (humidity) from the west coast to the east coast

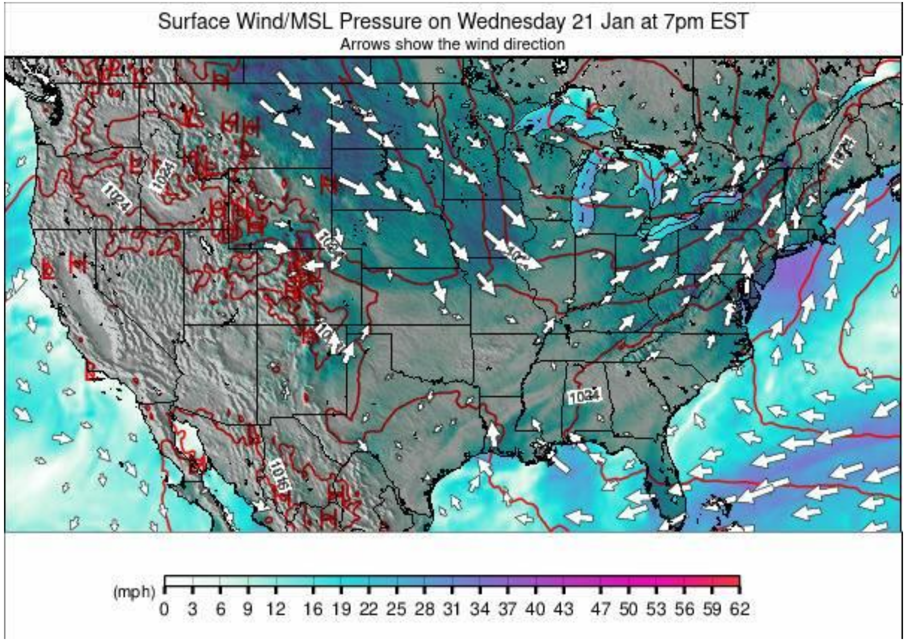
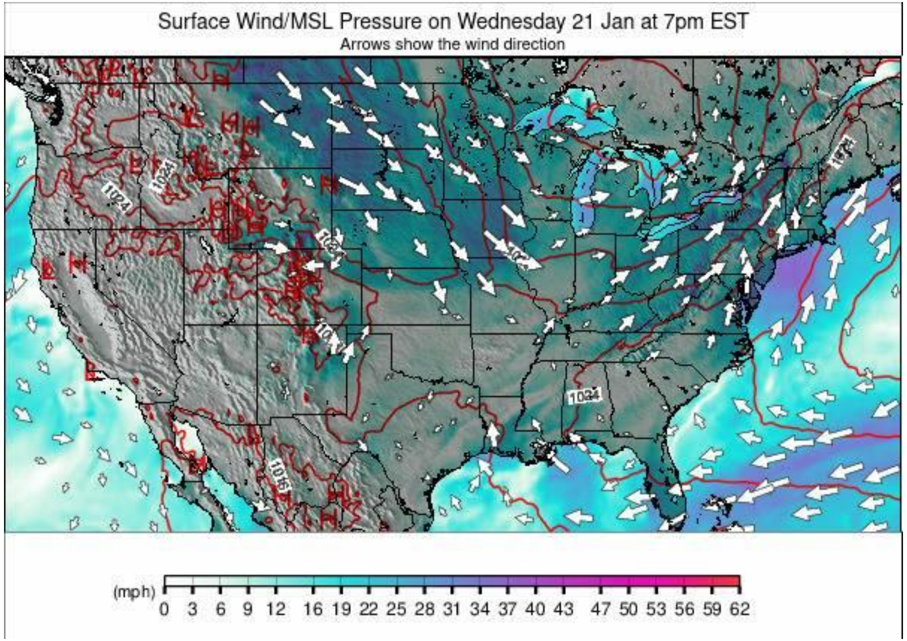


Image from https://www.snow-forecast.com/maps/static/usa/6/wind

Carnegie Mellon University

Why is this a normalizing flow

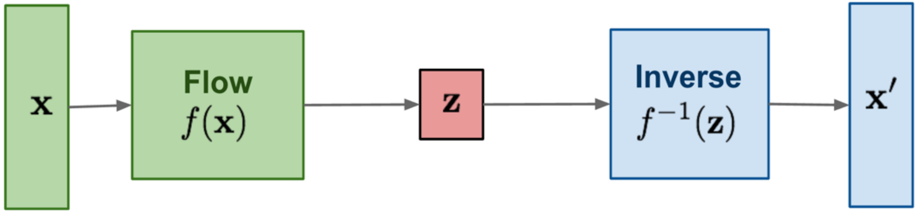


The streams never cross!

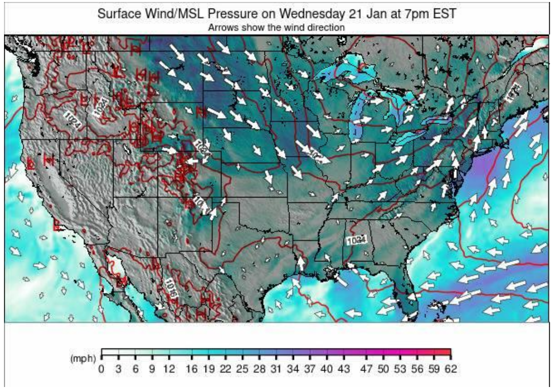
Carnegie Mellon University

Why is this a normalizing flow

Normalizing flows:



Because the streams never cross, following the ODE flow is an **invertible** transformation!



Carnegie Mellon University

How does this connect to probability

In order for a CNF to model transports between probability distributions, we need the following assumptions:

- **Conservation of mass:** No new mass and mass does not disappear

  => Probability always adds up to 1

- **Continuity equation:** Not only that the mass is conserved, it also does not teleport

  => Probability can only move/change continuously

Probability flux & divergence

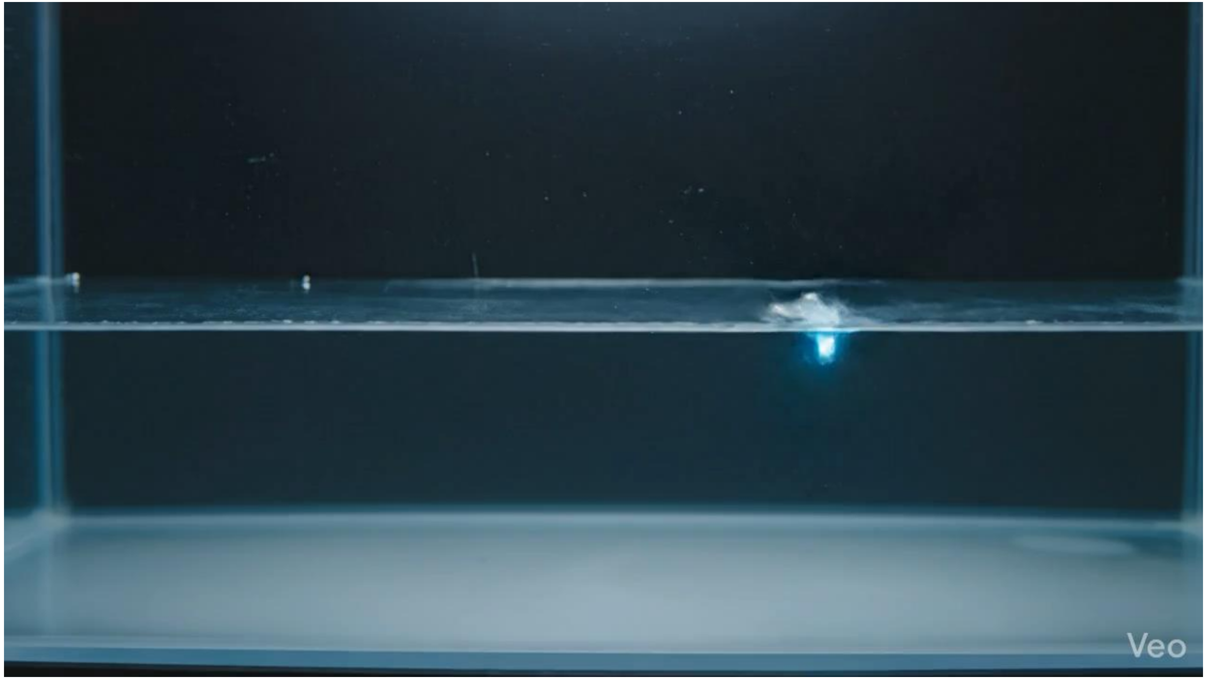
Flux: the amount of flow per unit time through a unit space

=> Probability flux = velocity x density

where and how fast it flows

how much probability it flows

Probability flux & divergence



Video generation by Veo

Probability flux & divergence

Flux: the amount of flow per unit time through a unit space

=> Probability flux = velocity x density

where and how fast it flows

how much

probability it flows

The two assumptions can be formally written in math in this way:

$$\frac{\partial p_t}{\partial t} = -\mathrm{div}\big(p_t(x_t)v(x_t,t)\big) = \sum_d \frac{\partial v(x_t,t)}{\partial x_t^{(d)}}$$

Divergence: how much probability that outflows

from a given point per unit time in every direction

Instantaneous change of variables (Chen et al. 2018)

$$\frac{\partial p_t}{\partial t} = -\mathrm{div}\,(p_t(x_t)v(x_t,t))$$

$$\frac{1}{p_t(x_t)}\frac{\partial p_t}{\partial t} = -\frac{1}{p_t(x_t)}\mathrm{div}(p_t(x_t)v(x_t,t))$$

$$\frac{\partial \log p_t}{\partial t} = -\frac{1}{p_t(x_t)}(< \nabla_{x_t} p_t, v > + p_t \mathrm{div}(v(x_t,t)))$$

$$= -(< \nabla_{x_t} \log p_t\,, v > + \mathrm{div}(v(x_t,t)))$$

$$= -(< \nabla_{x_t} \log p_t\,, \partial_t x_t > + \mathrm{div}(v(x_t,t)))$$

$$\frac{d \log p_t}{dt} = \frac{\partial \log p_t}{\partial t} + < \nabla_{x_t} \log p_t\,, \partial_t x_t >$$

$$\frac{d \log p_t}{dt} = -\mathrm{div}(v(x_t,t))$$

Carnegie Mellon University

Chen et al. Neural Ordinary Differential Equations. NeurIPS 2018. https://arxiv.org/abs/1806.07366

How CNF based models calculate likelihood

$$\frac{dx_t}{dt} = v(x_t, t)$$

$$x_t = x_0 + \int_0^t v(x_\tau, \tau) d\tau$$

$$\frac{d \log p_t}{dt} = -\mathrm{div}\big(v(x_t, t)\big)$$

$$\log p_t\,(x_t) = \log p_0(\,x_0) - \int_0^t \mathrm{div}\big(v(x_\tau, \tau)\big) d\tau$$

How to train your CNF models

$$\frac{dx_t}{dt} = v(x_t, t)$$

$$x_t = x_0 + \int_0^t v(x_\tau, \tau) d\tau$$

$$\frac{d \log p_t}{dt} = -\mathrm{div}\big(v(x_t, t)\big)$$

$$\log p_t\,(x_t) = \log p_0(\,x_0) - \int_0^t \mathrm{div}\big(v(x_\tau, \tau)\big) d\tau$$

Attempt 1: Maximum likelihood

$$\frac{dx_t}{dt} = v_\theta(x_t, t)$$

$$x_t = x_0 + \int_0^t v_\theta(x_\tau, \tau) d\tau$$

$$\frac{d \log p_t}{dt} = -\mathrm{div}\big(v_\theta(x_t, t)\big)$$

$$\log p_t\,(x_t; \theta) = \log p_0(\,x_0) - \int_0^t \mathrm{div}\big(v_\theta(x_\tau, \tau)\big) d\tau$$

$$=> \mathrm{argmax}_\theta \log p_1\,(x_1; \theta)$$

Need numerical integration at training time

$$\log p_1\,(x_1; \theta) = \log p_0(\,x_0) - \int_0^1 div\big(v_\theta(x_\tau, \tau)\big) d\tau$$

Carnegie Mellon University

Attempt 2: Flow matching

$$\frac{dx_t}{dt} = v_\theta(x_t, t)$$

$$x_t = x_0 + \int_0^t v_\theta(x_\tau, \tau) d\tau$$

$$\frac{d \log p_t}{dt} = -\mathrm{div}\big(v_\theta(x_t, t)\big)$$

$$\log p_t\,(x_t; \theta) = \log p_0(\,x_0) - \int_0^t \mathrm{div}\big(v_\theta(x_\tau, \tau)\big) d\tau$$

Both depend on the same velocity field

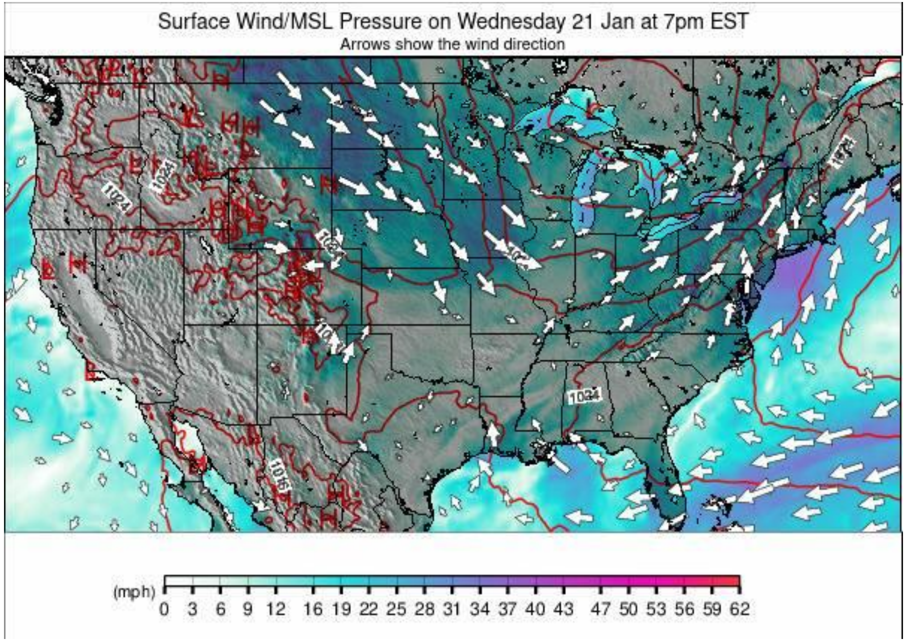
=> Just need to make sure v_θ match with the ground truth velocity

=> $||v_\theta(x_t, t) - u(x_t, t)||^2$

Ground truth velocity

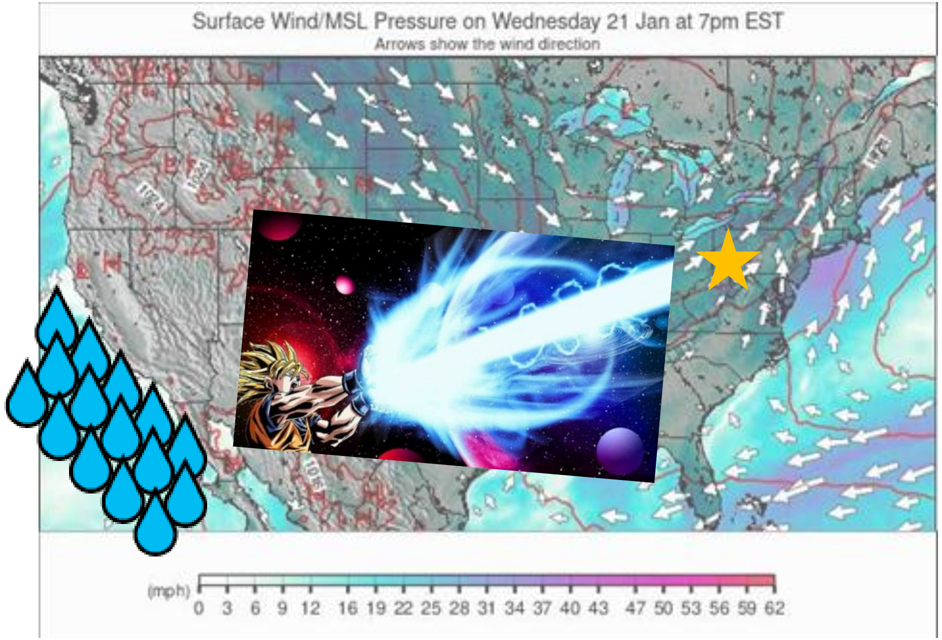
Carnegie Mellon University

But we don’t have the ground truth velocity



Carnegie Mellon University

What if we fix a point to transport



Carnegie Mellon University

Poseidon generation by nano banana, Goku Kamehameha from https://www.reddit.com/r/dbz/comments/2ggs8j/goku_kamehameha_wallpaper/

Conditional probability path => Marginal probability path

Given a data point x_1, it's usually to define a **conditional velocity field** $u_t(x_t|x_1)$

Then we call the trajectory of the probability distribution generated along the way the **conditional probability path** $p_t(x_t|x_1)$

Here the conditional probability path starts from the prior $p_0(x|x_1) = p_0(x)$, and always end up at x_1 or a small Gaussian concentrated around x_1, i.e. $p_1(x|x_1) = \delta(x_1)$, or $p_1(x|x_1) = N(x_1, \sigma^2 I)$ with small σ

Then then **marginal probability path** is

$$p_t(x_t) = \int p_t(x_t|x_1) p_{data}(x_1) dx_1$$

$$p_1(x) = \int p_1(x|x_1) p_{data}(x_1) dx_1 \approx p_{data}(x)$$

Carnegie Mellon University

Lipman et al. Flow Matching for Generative Modeling. ICLR 2023. https://arxiv.org/pdf/2210.02747

Conditional velocity => Marginal velocity

With a conditional velocity $u_t(x_t|x_1)$, we can also define a **marginal velocity**

$$u_t(x_t) = \int u_t(x_t|x_1) \frac{p_t(x_t|x_1)p_{data}(x_1)}{p_t(x_t)} dx_1$$

$\frac{p_t(x_t|x_1)p_{data}(x_1)}{p_t(x_t)}$: Pseudo "Bayes theorem"

- $p_t(x_t|x_1)$: how likely is current intermediate sample along the conditional probability path
- $p_{data}(x_1)$: how likely is the data point that defines the conditional probability path
- $p_t(x_t)$: how likely is the current intermediate sample in general (normalization)

Conditional velocity => Marginal velocity

With a conditional velocity $u_t(x_t|x_1)$, we can also define a **marginal velocity**

$$u_t(x_t) = \int u_t(x_t|x_1) \frac{p_t(x_t|x_1)p_{data}(x_1)}{p_t(x_t)} dx_1$$

$\frac{p_t(x_t|x_1)p_{data}(x_1)}{p_t(x_t)}$: Pseudo "Bayes theorem", sort of $p_t(x_1|x_t)$

$$u_t(x_t) = E_{x_1 \sim p_t(x_1|x_t)}[u_t(x_t|x_1)]$$

Intuitively, it's basically the average conditional velocity at location x_t time t, weighted by how likely the data point is for the current location and time

Carnegie Mellon University

Marginal velocity generates marginal probability path

$$p_t(x_t) = \int p_t(x_t|x_1)p_{data}(x_1)dx_1$$

$$\frac{\partial p_t}{\partial t} = -\text{div}(p_t v_t)$$

$$u_t(x_t) = \int u_t(x_t|x_1)\frac{p_t(x_t|x_1)p_{data}(x_1)}{p_t(x_t)}dx_1$$

$$\frac{\partial}{\partial t}p_t(x_t) = \frac{\partial}{\partial t}\int p_t(x_t|x_1)p_{data}(x_1)dx_1$$

$$= \int (\frac{\partial}{\partial t}p_t(x_t|x_1))p_{data}(x_1)dx_1$$

$$= \int -\text{div}(p_t(x_t|x_1)u_t(x_t|x_1))p_{data}(x_1)dx_1$$

$$= -\text{div}(\int u_t(x_t|x_1)p_t(x_t|x_1)p_{data}(x_1)dx_1)$$

$$= -\text{div}(u_t(x_t)p_t(x_t))$$

Carnegie Mellon University

Matching conditional velocity <=> Matching marginal velocity

$$L_{FM}(\theta) = E_{t,p_t(x_t)}[||v_\theta(x_t,t) - u_t(x_t)||^2]$$

$$= E_{t,p_t(x_t)}\left[||u_t(x_t)||^2\right] + E_{t,p_t(x_t)}\left[||v_\theta(x_t,t)||^2\right] - 2E_{t,p_t(x_t)}[\langle v_\theta(x_t,t), u_t(x_t)\rangle]$$

$$L_{CFM}(\theta) = E_{t,p_{data}(x_1),p_t(x_t|x_1)}[||v_\theta(x_t,t) - u_t(x_t|x_1)||^2]$$

$$= E_{t,p_{data}(x_1),p_t(x_t|x_1)}\left[||u_t(x_t|x_1)||^2\right] + E_{t,p_{data}(x_1),p_t(x_t|x_1)}\left[||v_\theta(x_t,t)||^2\right]$$

$$- 2E_{t,p_{data}(x_1),p_t(x_t|x_1)}[\langle v_\theta(x_t,t), u_t(x_t|x_1)\rangle]$$

Constant w.r.t. θ

Matching conditional velocity <=> Matching marginal velocity

$$L_{FM}(\theta) = E_{t,p_t(x_t)}\left[||v_\theta(x_t,t)||^2\right] - 2E_{t,p_t(x_t)}[\langle v_\theta(x_t,t), u_t(x_t)\rangle]$$

$$L_{CFM}(\theta) = E_{t,p_{data}(x_1),p_t(x_t|x_1)}\left[||v_\theta(x_t,t)||^2\right] - 2E_{t,p_{data}(x_1),p_t(x_t|x_1)}[\langle v_\theta(x_t,t), u_t(x_t|x_1)\rangle]$$

Matching conditional velocity <=> Matching marginal velocity

$$L_{FM}(\theta) = E_{t,p_t(x_t)}\left[||v_\theta(x_t,t)||^2\right] - 2E_{t,p_t(x_t)}[\langle v_\theta(x_t,t), u_t(x_t)\rangle]$$

$$L_{CFM}(\theta) = E_{t,p_{data}(x_1),p_t(x_t|x_1)}\left[||v_\theta(x_t,t)||^2\right] - 2E_{t,p_{data}(x_1),p_t(x_t|x_1)}[\langle v_\theta(x_t,t), u_t(x_t|x_1)\rangle]$$

$$E_{p_{data}(x_1),p_t(x_t|x_1)}\left[||v_\theta(x_t,t)||^2\right] = \int\int ||v_\theta(x_t,t)||^2\, p_t(x_t|x_1)p_{data}(x_1)dx_t dx_1$$

$$= \int ||v_\theta(x_t,t)||^2 p_t(x_t)dx_t = E_{p_t(x_t)}\left[||v_\theta(x_t,t)||^2\right]$$

Carnegie Mellon University

Matching conditional velocity <=> Matching marginal velocity

$$L_{FM}(\theta) = E_{t,p_t(x_t)}\left[\left|\left|v_\theta(x_t,t)\right|\right|^2\right] - 2E_{t,p_t(x_t)}[\langle v_\theta(x_t,t), u_t(x_t)\rangle]$$

$$L_{CFM}(\theta) = E_{t,p_{data}(x_1),p_t(x_t|x_1)}\left[\left|\left|v_\theta(x_t,t)\right|\right|^2\right] - 2E_{t,p_{data}(x_1),p_t(x_t|x_1)}[\langle v_\theta(x_t,t), u_t(x_t|x_1)\rangle]$$

$$E_{p_t(x_t)}[\langle v_\theta(x_t,t), u_t(x_t)\rangle] = \int \left\langle v_\theta(x_t,t), \int u_t(x_t|x_1)\frac{p_t(x_t|x_1)p_{data}(x_1)}{p_t(x_t)}dx_1\right\rangle p_t(x_t)dx_t$$

$$= \int \left\langle v_\theta(x_t,t), \int u_t(x_t|x_1)p_t(x_t|x_1)p_{data}(x_1)dx_1\right\rangle dx_t$$

$$= \int \langle v_\theta(x_t,t), u_t(x_t|x_1)\rangle p_t(x_t|x_1)p_{data}(x_1)dx_1dx_t$$

$$= E_{p_{data}(x_1),p_t(x_t|x_1)}[\langle v_\theta(x_t,t), u_t(x_t|x_1)\rangle]$$

Carnegie Mellon University

Matching conditional velocity <=> Matching marginal velocity

$$L_{FM}(\theta) = E_{t,p_t(x_t)}\left[||v_\theta(x_t,t)||^2\right] - 2E_{t,p_t(x_t)}[\langle v_\theta(x_t,t), u_t(x_t)\rangle]$$

$$L_{CFM}(\theta) = E_{t,p_{data}(x_1),p_t(x_t|x_1)}\left[||v_\theta(x_t,t)||^2\right] - 2E_{t,p_{data}(x_1),p_t(x_t|x_1)}[\langle v_\theta(x_t,t), u_t(x_t|x_1)\rangle]$$

$$\Rightarrow \text{argmin}_\theta L_{FM}(\theta) = \text{argmin}_\theta L_{CFM}(\theta)$$

$\Rightarrow$ We just need to match the conditional velocity

$$||v_\theta(x_t,t) - u_t(x_t|x_1)||^2 \text{ !!!}$$

Now suppose our conditional probability path is to transform a Gaussian straight to a single point with constant speed

$$p_0 = N(0, I), p_1 = \delta(x_1)$$

$$x_t = tx_1 + (1-t)x_0, \qquad x_0 \sim p_0$$

$$p_t(x_t|x_1) = N(tx_1, (1-t)^2 I)$$

$$\frac{dx_t}{dt} = u(x_t|x_1) = x_1 - x_0$$

$$L_{CFM}(\theta) = E_{t, p_{data}(x_1), p_0(x_0)}[||v_\theta(x_t, t) - (x_1 - x_0)||^2]$$

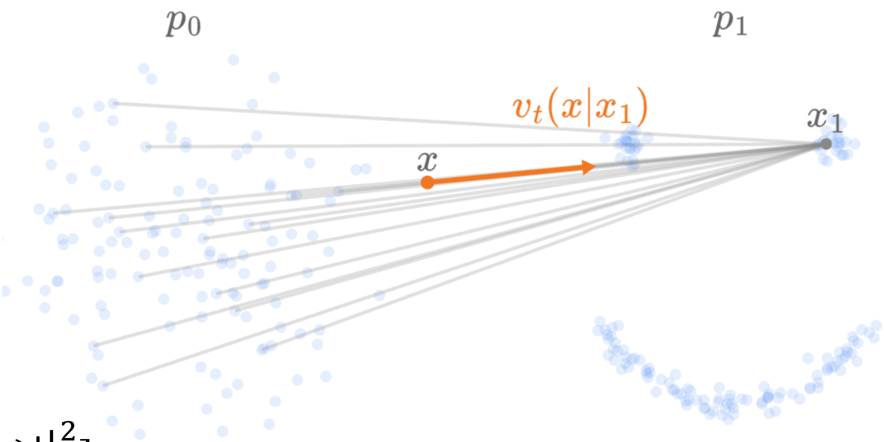


Image from https://alechelbling.com/blog/rectified-flow/

Cond-OT flow matching

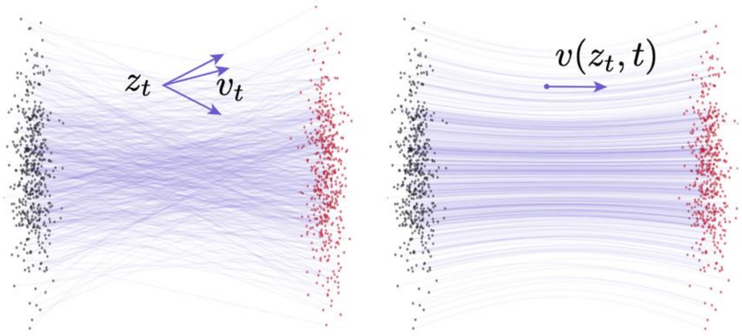


Figure 2: **Velocity fields in Flow Matching** [28]. **Left**: *conditional* flows [28]. A given z_t can arise from different (x, ϵ) pairs, resulting in different conditional velocities v_t. **Right**: *marginal* flows [28], obtained by marginalizing over all possible conditional velocities. The marginal velocity field serves as the underlying ground-truth field for network training. All velocities shown here are essentially ***instantaneous*** velocities. Illustration follows [12]. (*Gray dots: samples from prior; red dots: samples from data.*)

Carnegie Mellon University

Geng et al. "Mean Flows for One-step Generative Modeling". NeurIPS 2025. https://arxiv.org/pdf/2505.13447

Cond-OT flow matching a.k.a **Rectified flow** a.k.a **A special case of stochastic interpolant**

Three different groups of people develop the same algorithm from different theoretical perspective at the same time!

- Lipman et al. "Flow matching for generative modeling". ICLR 2023. https://arxiv.org/pdf/2210.02747
- Liu & Gong. "Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow". ICLR 2023. https://arxiv.org/pdf/2209.03003
- Albergo & Vanden-Eijnden. "Building Normalizing Flows with Stochastic Interpolants". ICLR 2023. https://arxiv.org/pdf/2209.15571

Hmm but the marginal flows are not the straightest

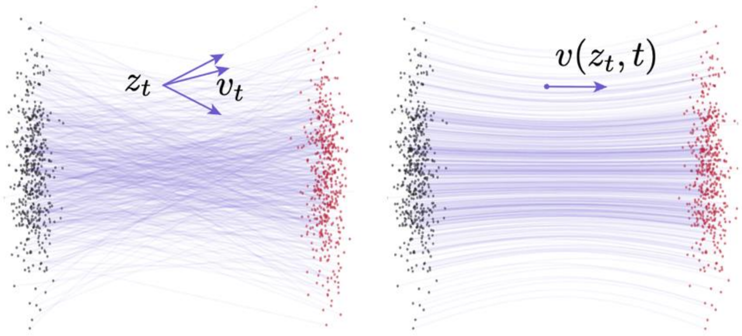
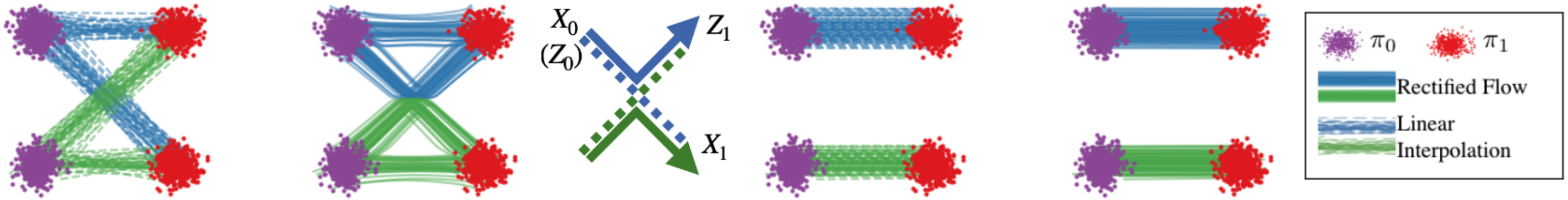


Figure 2: **Velocity fields in Flow Matching** [28]. **Left**: *conditional* flows [28]. A given z_t can arise from different (x, ϵ) pairs, resulting in different conditional velocities v_t. **Right**: *marginal* flows [28], obtained by marginalizing over all possible conditional velocities. The marginal velocity field serves as the underlying ground-truth field for network training. All velocities shown here are essentially ***instantaneous*** velocities. Illustration follows [12]. (*Gray dots: samples from prior; red dots: samples from data.*)

Carnegie Mellon University

Geng et al. "Mean Flows for One-step Generative Modeling". NeurIPS 2025. https://arxiv.org/pdf/2505.13447

Reflow: Flow matching on the flow matched pairs



(a) Linear interpolation $X_t = tX_1 + (1-t)X_0$

(b) Rectified flow Z_t induced by (X_0, X_1)

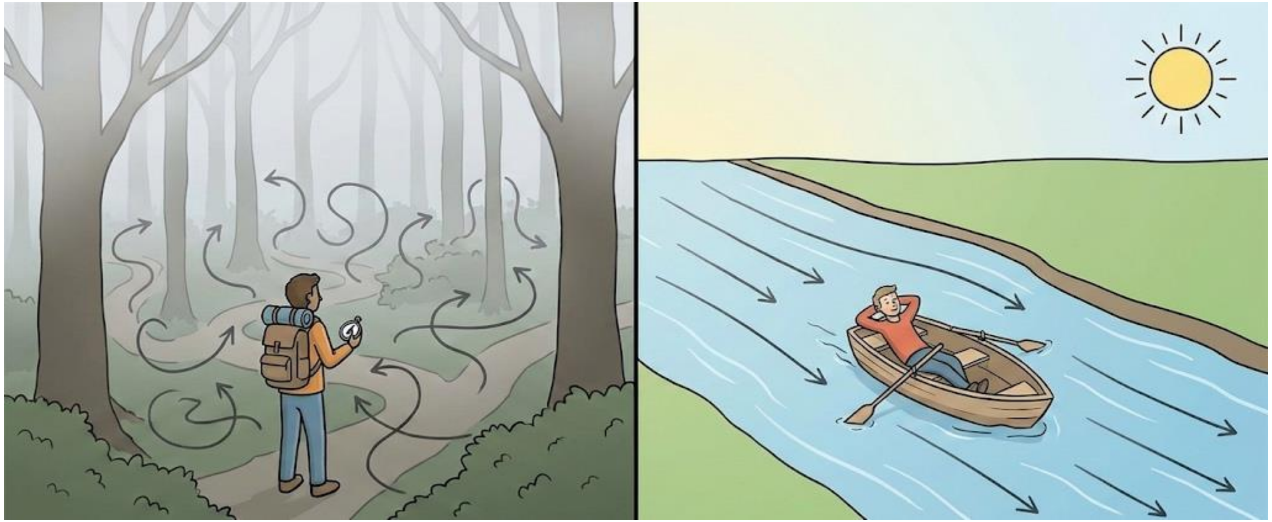
(c) Linear interpolation $Z_t = tZ_1 + (1-t)Z_0$

(d) Rectified flow Z'_t induced by (Z_0, Z_1)

Liu & Gong. "Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow". ICLR 2023. https://arxiv.org/pdf/2209.03003

Diffusion v.s. Flow matching

- Diffusion is like wandering in the woods with a compass
- Flow matching is like sitting on a boat in a river

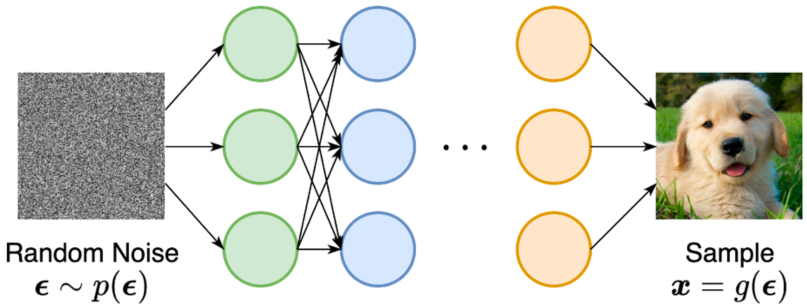


So far we have seen a bunch of generative models…

In general, we can roughly categorize generative models into the following categories

- **Likelihood Based:** Autoregressive models, variational autoencoders (VAE), normalizing flow, energy-based models (EBM), **diffusion models**
- **Likelihood Free:** Generative adversarial networks (GAN), **score-based models,** **flow matching**

Same thing!



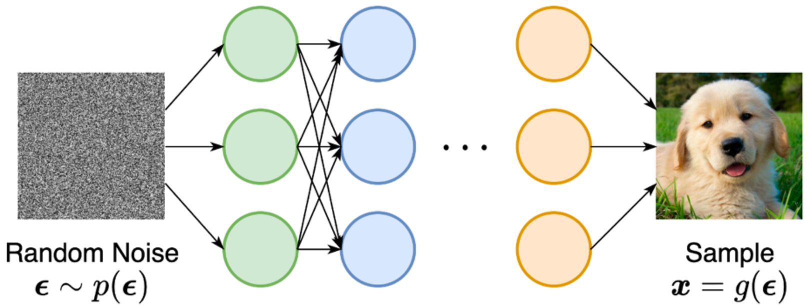
Directly sampling from P(X) is usually hard because they are usually complicated! But **sampling from a simpler distribution** (eg. a Gaussian) is easy!

So far we have seen a bunch of generative models…

In general, we can roughly categorize generative models into the following categories

- **Likelihood Based:** Autoregressive models, variational autoencoders (VAE), normalizing flow, energy-based models (EBM), **diffusion models**
- **Likelihood Free:** Generative adversarial networks (GAN), **score-based models,** **flow matching**

Same thing?



Directly sampling from P(X) is usually hard because they are usually complicated! But **sampling from a simpler distribution** (eg. a Gaussian) is easy!

Carnegie Mellon University

Diffusion path flow matching == diffusion

Cond-OT path:

$p_0 = N(0, I), p_1 = \delta(x_1)$

$x_t = tx_1 + (1-t)x_0, \qquad x_0 \sim p_0$

$p_t(x_t|x_1) = N(tx_1, (1-t)^2 I)$

$$\frac{dx_t}{dt} = u(x_t|x_1) = x_1 - x_0$$

VE diffusion path:

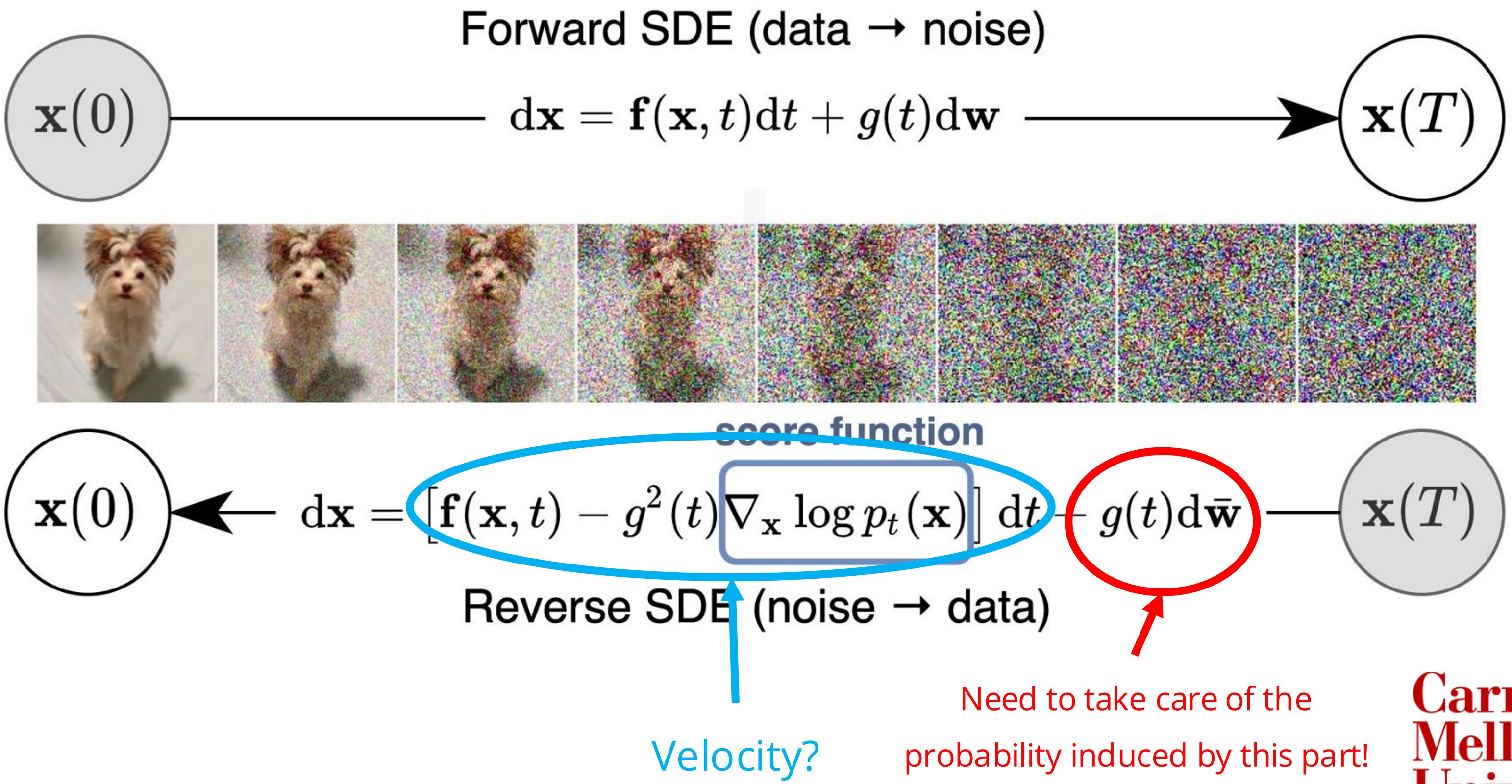
$p_0 = N(0, I), p_1 = \delta(x_1)$

$x_t = x_1 + \sigma_{1-t}\epsilon_t, \qquad \epsilon_t \sim N(0, I)$

$p_t(x_t|x_1) = N(x_1, {\sigma_{1-t}}^2 I)$

$$\frac{dx_t}{dt} = u(x_t|x_1) = -\frac{\sigma'_{1-t}}{\sigma_{1-t}}(x_t - x_1)$$

How about Score SDE => Flow ODE?



Score SDE => Flow ODE via Fokker–Planck PDE

Reverse Score SDE:

$$\mathrm{d}\mathbf{x} = [\mathbf{f}(\mathbf{x}, t) - g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})]\mathrm{d}t + g(t)\mathrm{d}\bar{\mathbf{w}},$$

Forward SDE:

$$dx = f(x,t)dt + g(t)dw$$

Fokker–Planck PDE of the forward SDE:

$$\partial_t p_t(x) = -\mathrm{div}\big(f(x,t)p_t(x)\big) + \frac{1}{2} g(t)^2 \Delta_x p_t(x)$$

Probability flow ODE:

$$\mathrm{d}\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - \frac{1}{2} g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right]\mathrm{d}t,$$

You can sample with this ODE now!

Continuity equation:

$$\partial_t p_t(x) = -\mathrm{div}\big(v_t(x)p_t(x)\big)$$

$$= -\mathrm{div}\big(f(x,t)p_t(x)\big) + \frac{1}{2} g(t)^2 \mathrm{div}(p_t(x)\nabla_x \log p_t(x))$$

$$= -\mathrm{div}\big(f(x,t)p_t(x)\big) + \frac{1}{2} g(t)^2 \mathrm{div}(\nabla_x p_t(x))$$

$$= -\mathrm{div}\big(f(x,t)p_t(x)\big) + \frac{1}{2} g(t)^2 \Delta_x p_t(x)$$

Same PDE for $\partial_t p_t(x)$ <=> Marginals $p_t(x)$ are the same!

Carnegie Mellon University

Density estimation with flow matching

$$\frac{dx_t}{dt} = v(x_t, t)$$

$$\widehat{x_t} = x_1 - \int_t^1 v(\hat{x}_\tau, \tau)d\tau$$

$$\frac{d\log p_t}{dt} = -div\big(v(x_t, t)\big)$$

$$\log p_1(x_1) = \log p_0(\hat{x}_0) - \int_0^1 div\big(v(\hat{x}_\tau, \tau)\big)d\tau$$

Carnegie Mellon University

Density estimation with diffusion Attempt 1: ELBO

$$\log p_\theta(x_0) = \log \int p_\theta(x_{0:T}) dx_{1:T}$$
$$= \log \int q(x_{1:T}|x_0) \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)} dx_{1:T}$$
$$= \log E_{q(x_{1:T}|x_0)}[\frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}]$$
$$\geq E_{q(x_{1:T}|x_0)}[\log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}]$$

=> Just use the loss function as an estimation of the density

$$L_{\text{simple}}(\theta) := \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2\right]$$

Density estimation with diffusion Attempt 2: PF ODE

$$\frac{dx_t}{dt} = v(x_t, t)$$

$$\mathrm{d}\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - \frac{1}{2} g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right] \mathrm{d}t,$$

$$\hat{x}_t = x_1 - \int_t^1 v(\hat{x}_\tau, \tau) d\tau$$

$$\frac{d \log p_t}{dt} = -\mathrm{div}\big(v(x_t, t)\big)$$

$$\log p_1(x_1) = \log p_0(\hat{x}_0) - \int_0^1 \mathrm{div}\big(v(\hat{x}_\tau, \tau)\big) d\tau$$

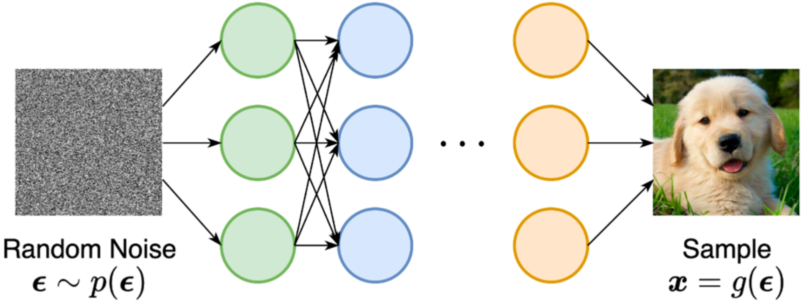
Carnegie Mellon University

So far we have seen a bunch of generative models…

In general, we can roughly categorize generative models into the following categories

- **Likelihood Based:** Autoregressive models, variational autoencoders (VAE), normalizing flow, energy-based models (EBM), **diffusion models**
- **Likelihood Free:** Generative adversarial networks (GAN), **score-based models, flow matching**

Same thing!



Directly sampling from P(X) is usually hard because they are usually complicated! But **sampling from a simpler distribution** (eg. a Gaussian) is easy!

Carnegie Mellon University

We have gone through all the basics! 🎉

Starting from next week, we will be exploring different options to improve diffusion and flow matching models

- The design space of diffusion models (i.e. what knobs can we tune to make the models better)
- How to make generation faster (through training and with no additional training)
- How to make the generation more controllable (through training and with no additional training)