ICML 2026 Tutorial

Diffusion and Flow Matching

From Memorization to Generalization & Beyond

Quentin Bertrand & Mathurin Massias

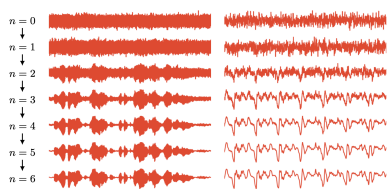
https://memorization-generalization.github.io

Inria | ENS Lyon | Laboratoire Hubert Curien | Mila Affiliated Member | CIFAR Global Scholar

Diffusion and Flow matching are everywhere

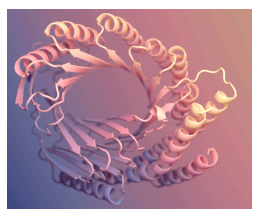
- Diffusion introduced in 2015[1], took off in 2020[2,3]
- Flow matching introduced in 2023[4,5,6]
- Current SOTA techniques for generation of images, audio, videos, proteins...

Stable Diffusion



WaveGrad

Sora



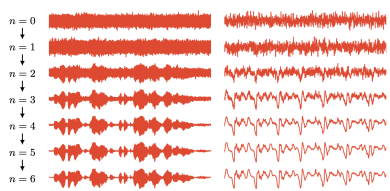
RF Diffusion

[1] J. Sohl-Dickstein et al., **Deep unsupervised learning using nonequilibrium thermodynamics,** In: ICML, 2015.
[2] J. Ho et al., **Denoising diffusion probabilistic models,** In: NeurIPS, 2020.
[3] Y. Song et al., **Score-based generative modeling through stochastic differential equations,** In: ICLR, 2021.
[4] Y. Lipman et al., **Flow matching for generative modeling,** In: ICLR, 2023.
[5] M. Albergo et al., **Building Normalizing Flows with Stochastic Interpolants,** In: ICLR, 2023.
[6] X. Liu et al., **Flow straight and fast: Learning to generate and transfer data with rectified flow,** In: ICLR, 2023.

Diffusion and Flow matching are everywhere

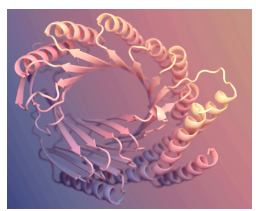
- Diffusion introduced in 2015[1], took off in 2020[2,3]
- Flow matching introduced in 2023[4,5,6]
- Current SOTA techniques for generation of images, audio, videos, proteins...
- And even for text now

Stable Diffusion



WaveGrad

Sora



RF Diffusion

Diffusion Gemma

[1] J. Sohl-Dickstein et al., **Deep unsupervised learning using nonequilibrium thermodynamics,** In: ICML, 2015.
[2] J. Ho et al., **Denoising diffusion probabilistic models,** In: NeurIPS, 2020.
[3] Y. Song et al., **Score-based generative modeling through stochastic differential equations,** In: ICLR, 2021.
[4] Y. Lipman et al., **Flow matching for generative modeling,** In: ICLR, 2023.
[5] M. Albergo et al., **Building Normalizing Flows with Stochastic Interpolants,** In: ICLR, 2023.
[6] X. Liu et al., **Flow straight and fast: Learning to generate and transfer data with rectified flow,** In: ICLR, 2023.

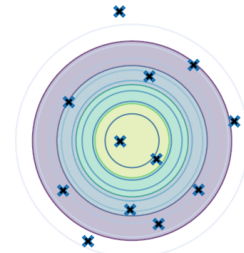
These models have stellar performances but…

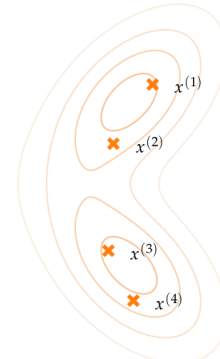
They are mostly black boxes with many puzzles:

These models have stellar performances but…

They are mostly black boxes with many puzzles:

- Optimal models should only generate training data?[7]



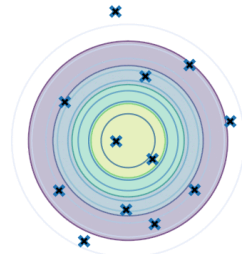


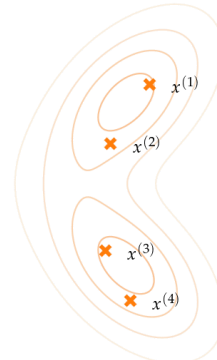
[7] X. Gu et al., **On memorization in diffusion models,** In: TMLR, 2023.
[8] Z. Kadkhodaie et al., **Generalization in Diffusion Models Arises from Geometry-Adaptive Harmonic Representations,** In: ICLR, 2024.
[9] M. Kamb et al., **An Analytic Theory of Creativity in Convolutional Diffusion Models,** In: ICML, 2025.

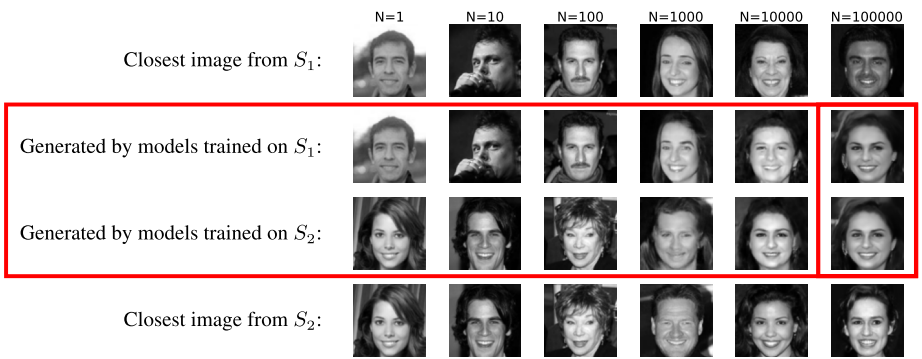
These models have stellar performances but…

They are mostly black boxes with many puzzles:

- Optimal models should only generate training data?[7]
- Models trained on different datasets generate the same images?[8]





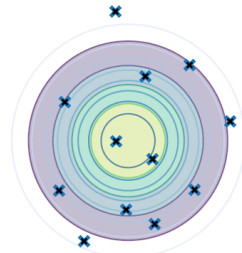


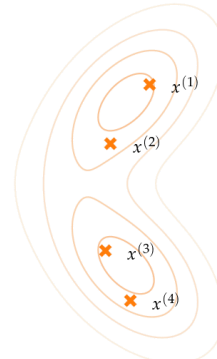
[7] X. Gu et al., **On memorization in diffusion models,** In: TMLR, 2023.
[8] Z. Kadkhodaie et al., **Generalization in Diffusion Models Arises from Geometry-Adaptive Harmonic Representations,** In: ICLR, 2024.
[9] M. Kamb et al., **An Analytic Theory of Creativity in Convolutional Diffusion Models,** In: ICML, 2025.

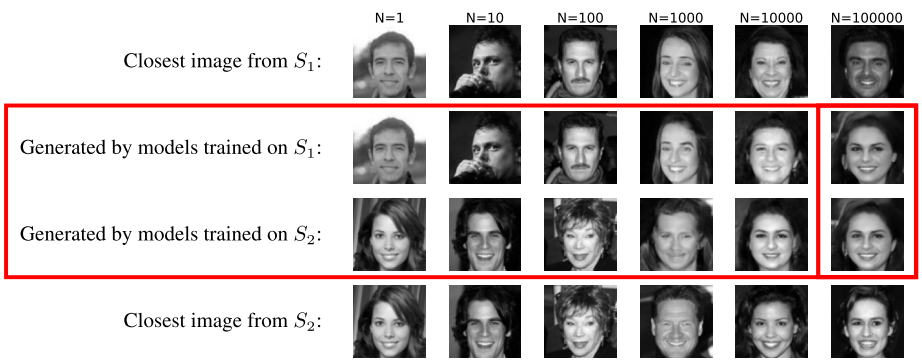
These models have stellar performances but…

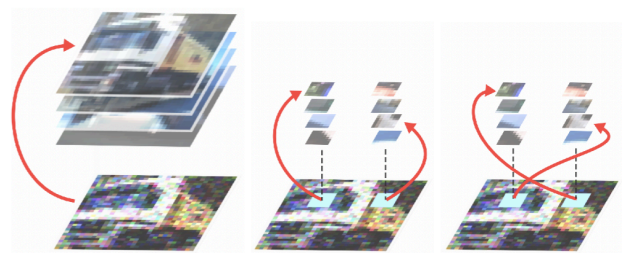
They are mostly black boxes with many puzzles:

- Optimal models should only generate training data?[7]
- Models trained on different datasets generate the same images?[8]
- Generated images are patchworks of training images?[9]









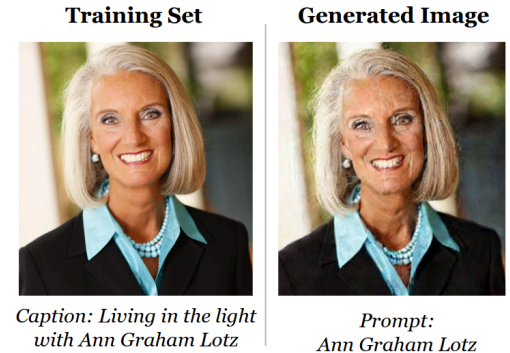
[7] X. Gu et al., **On memorization in diffusion models,** In: TMLR, 2023.
[8] Z. Kadkhodaie et al., **Generalization in Diffusion Models Arises from Geometry-Adaptive Harmonic Representations,** In: ICLR, 2024.
[9] M. Kamb et al., **An Analytic Theory of Creativity in Convolutional Diffusion Models,** In: ICML, 2025.

Tutorial goal

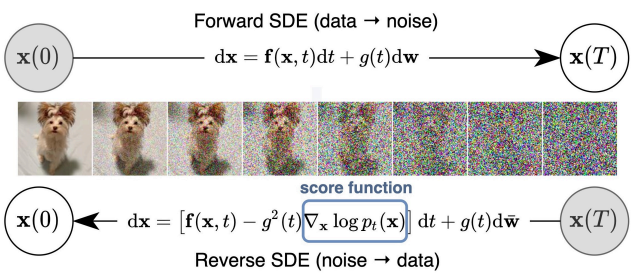
When, how, and why do Diffusion & Flow matching create new data?

Plan

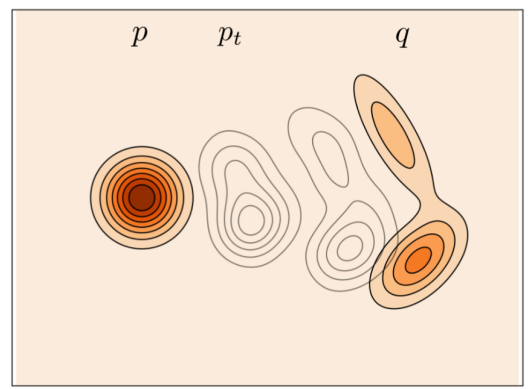
- Introduction to Flow matching and Diffusion: why should they memorize? (40 min)
- Discussion #1 (10 min)



Extracting Training Data from Diffusion Models Carlini et al. (2023)



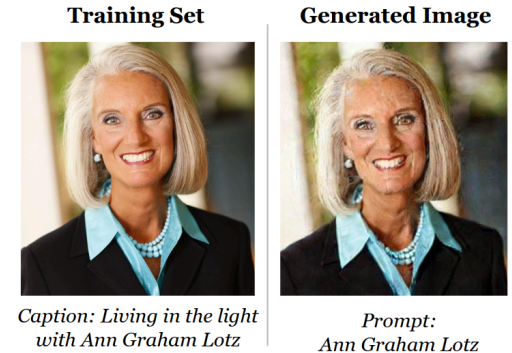
https://yang-song.net/blog/2021/score/



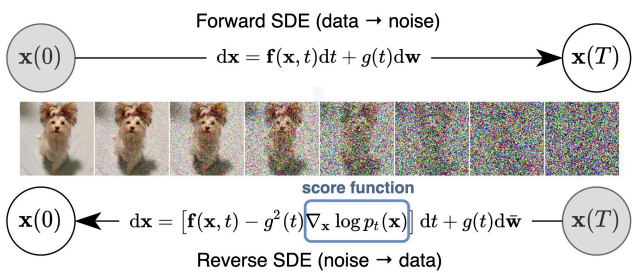
Flow Matching Guide and Code, Lipman et al. (2025)

Plan

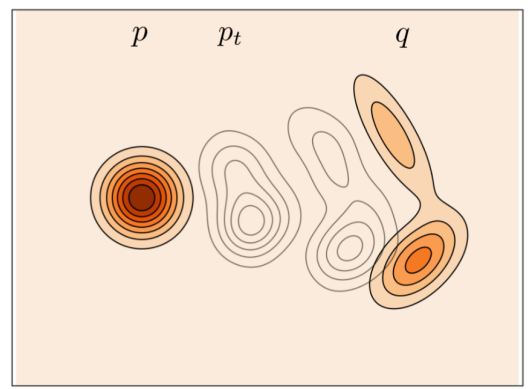
- Introduction to Flow matching and Diffusion: why should they memorize? (40 min)
- Discussion #1 (10 min)
- Toward understanding the generalization of Flow matching and Diffusion (30 min)
- Discussion #2 (10 min)



Extracting Training Data from Diffusion Models Carlini et al. (2023)



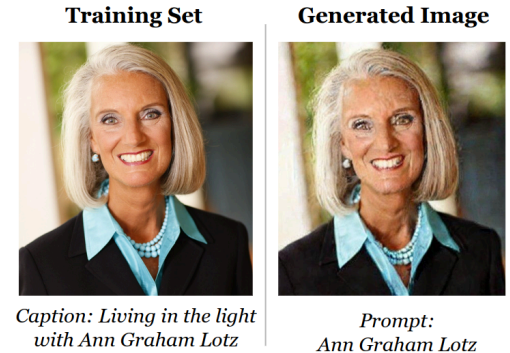
https://yang-song.net/blog/2021/score/



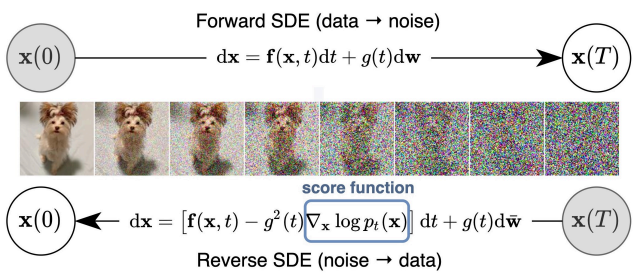
Flow Matching Guide and Code, Lipman et al. (2025)

Plan

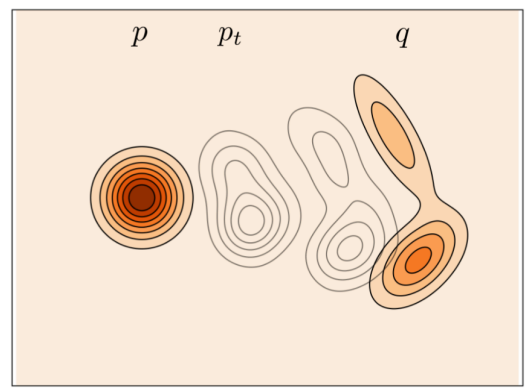
- Introduction to Flow matching and Diffusion: why should they memorize? (40 min)
- Discussion #1 (10 min)
- Toward understanding the generalization of Flow matching and Diffusion (30 min)
- Discussion #2 (10 min)
- Beyond generalization (30 min)
- Current open problems (15 min)
- Discussion #3 (15 min)



Extracting Training Data from Diffusion Models Carlini et al. (2023)



https://yang-song.net/blog/2021/score/

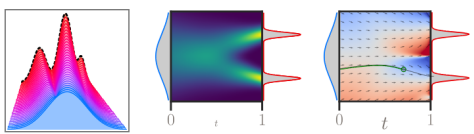


Flow Matching Guide and Code, Lipman et al. (2025)

Who we are

- Researchers at Inria (France)
- Blog post[10] and contributions in memorization[11,12]

A Visual Dive into Conditional Flow Matching



On the Closed-Form of Flow Matching: Generalization Does Not Arise from Target Stochasticity

Quentin Bertrand[15]*; Anne Gagneux[2]*; Mathurin Massias[3]*; Rémi Emonet[14]*

TRAINING FLOW MATCHING: THE ROLE OF WEIGHTING AND PARAMETERIZATION

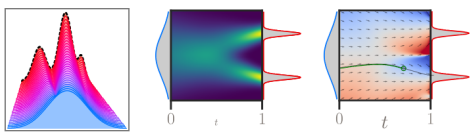
Anne Gagneux*[1], Ségolène Martin*[23], Rémi Gribonval[3] & Mathurin Massias[3]

[10] A. Gagneux et al., **A Visual Dive into Conditional Flow Matching,** In: ICLR Blogposts, 2025.
[11] Q. Bertrand et al., **On the Closed-Form of Flow Matching: Generalization Does Not Arise from Target Stochasticity,** In: NeurIPS, 2025.
[12] A. Gagneux et al., **Training flow matching: The role of weighting and parameterization,** In: ICLR Delta workshop, 2026.

Who we are

- Researchers at Inria (France)
- Blog post[10] and contributions in memorization[11,12]

A Visual Dive into Conditional Flow Matching



On the Closed-Form of Flow Matching: Generalization Does Not Arise from Target Stochasticity

Quentin Bertrand[15]*; Anne Gagneux[2]*; Mathurin Massias[3]*; Rémi Emonet[14]*

TRAINING FLOW MATCHING: THE ROLE OF WEIGHTING AND PARAMETERIZATION

Anne Gagneux*[1], Ségolène Martin*[23], Rémi Gribonval[3] & Mathurin Massias[3]

- Many thanks to Rémi Emonet, Ségolène Martin, Anne Gagneux, Georges Le Bellier and Eloi Tanguy who contributed to the material





[10] A. Gagneux et al., **A Visual Dive into Conditional Flow Matching,** In: ICLR Blogposts, 2025.
[11] Q. Bertrand et al., **On the Closed-Form of Flow Matching: Generalization Does Not Arise from Target Stochasticity,** In: NeurIPS, 2025.
[12] A. Gagneux et al., **Training flow matching: The role of weighting and parameterization,** In: ICLR Delta workshop, 2026.

A few disclaimers

A few disclaimers

- We sometimes trade rigor for pedagogy; there will be approximations

A few disclaimers

- We sometimes trade rigor for pedagogy; there will be approximations
- For completeness we include some bonus slides that we will not cover live

A few disclaimers

- We sometimes trade rigor for pedagogy; there will be approximations
- For completeness we include some bonus slides that we will not cover live
- We limit ourselves to unconditional image generation

A few disclaimers

- We sometimes trade rigor for pedagogy; there will be approximations
- For completeness we include some bonus slides that we will not cover live
- We limit ourselves to unconditional image generation
- We could not include all the literature on the topic[1]

[1] If you feel we should have included something, please send us an email!

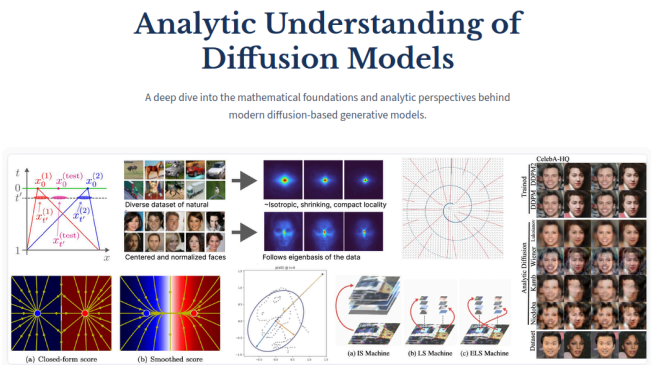
A few disclaimers

- We sometimes trade rigor for pedagogy; there will be approximations
- For completeness we include some bonus slides that we will not cover live
- We limit ourselves to unconditional image generation
- We could not include all the literature on the topic[1]
- Presentation available at https://memorization-generalization.github.io

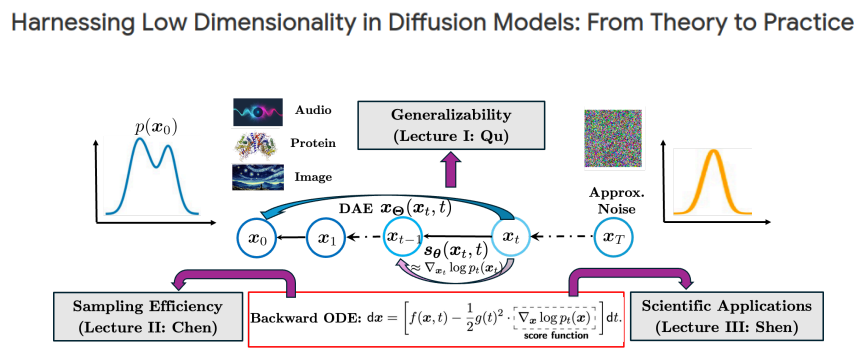
[1] If you feel we should have included something, please send us an email!

A few disclaimers

- We sometimes trade rigor for pedagogy; there will be approximations
- For completeness we include some bonus slides that we will not cover live
- We limit ourselves to unconditional image generation
- We could not include all the literature on the topic[1]
- Presentation available at https://memorization-generalization.github.io
- Other tutorials!



https://analytic-diffusion.github.io/ CVPR 2026



https://icml-2025-tutorial.github.io/ ICML 2025

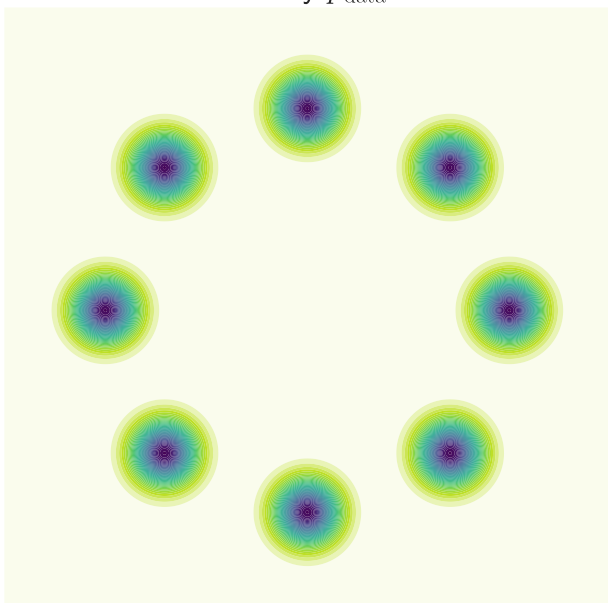
[1] If you feel we should have included something, please send us an email!

Introduction to
Flow matching and Diffusion

A step back: what is generative modeling?

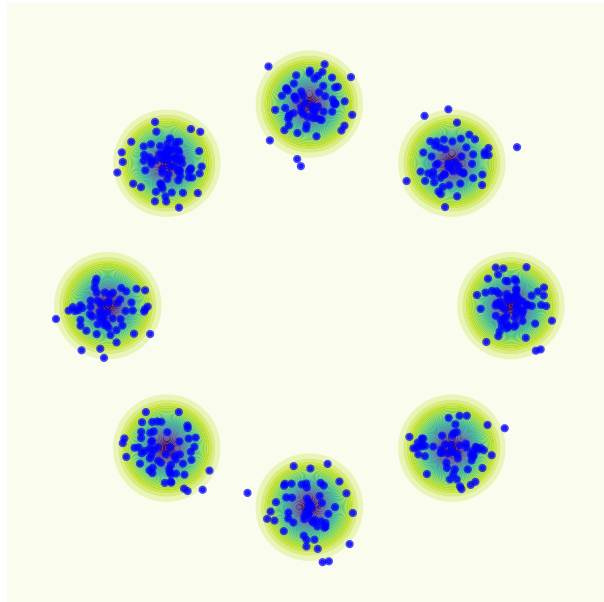
A step back: what is generative modeling?

density p_{data}



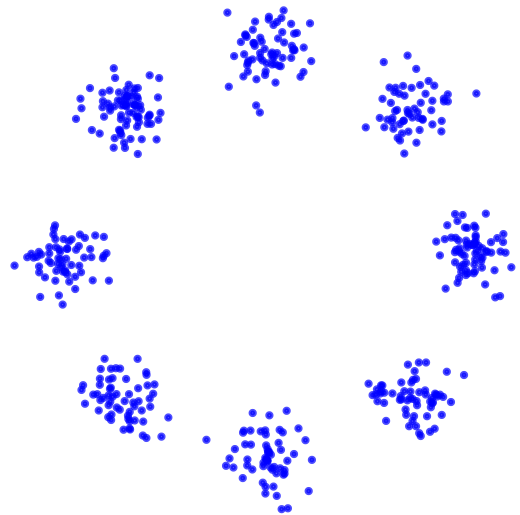
A step back: what is generative modeling?

density p_{data} + samples $x^{(1)}, \ldots, x^{(n)} \sim p_{\text{data}}$



A step back: what is generative modeling?

samples $x^{(1)}, \ldots, x^{(n)} \sim p_{\text{data}}$

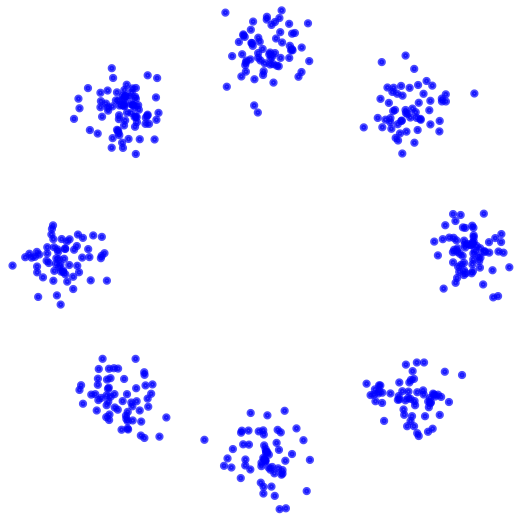


A step back: what is generative modeling?

Problem setting:

- Access to n samples $x^{(1)}, \ldots, x^{(n)}$
- $\underbrace{x^{(1)}, \ldots, x^{(n)}}_{\text{known}} \sim \underbrace{p_{\text{data}}}_{\text{unknown}}$

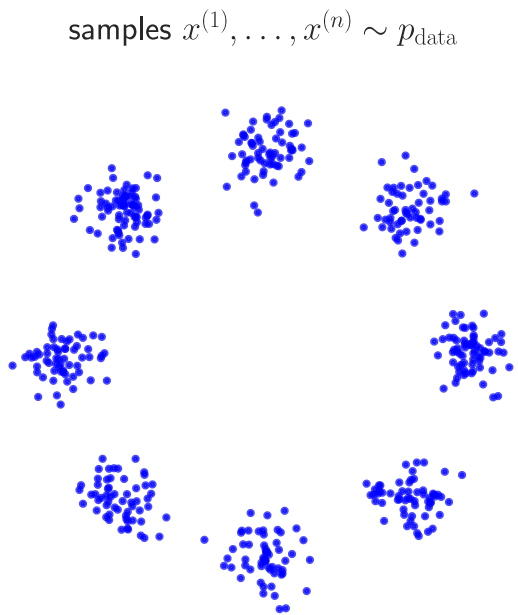
samples $x^{(1)}, \ldots, x^{(n)} \sim p_{\text{data}}$



A step back: what is generative modeling?

Problem setting:

- Access to n samples $x^{(1)}, \ldots, x^{(n)}$
- $\underbrace{x^{(1)}, \ldots, x^{(n)}}_{\text{known}} \sim \underbrace{p_{\text{data}}}_{\text{unknown}}$



Goal:

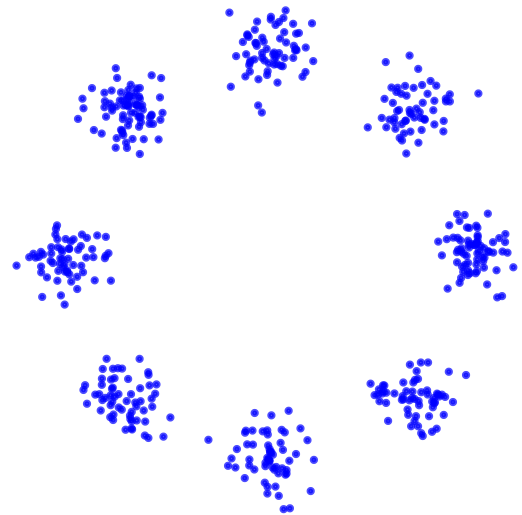
- Draw new samples $\underbrace{x^{(1)}_{\text{new}}, x^{(2)}_{\text{new}}, \ldots}_{\text{goal}} \sim \underbrace{p_{\text{data}}}_{\text{unknown}}$

A step back: what is generative modeling?

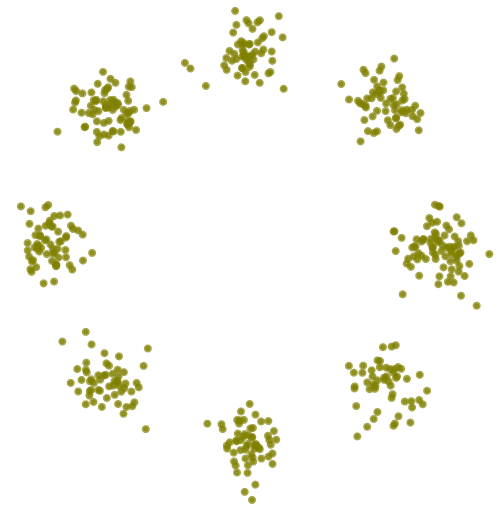
Problem setting:

- Access to n samples $x^{(1)}, \ldots, x^{(n)}$
- $\underbrace{x^{(1)}, \ldots, x^{(n)}}_{\text{known}} \sim \underbrace{p_{\text{data}}}_{\text{unknown}}$

samples $x^{(1)}, \ldots, x^{(n)} \sim p_{\text{data}}$



Goal: generate samples $x^{(1)}_{\text{new}}, x^{(2)}_{\text{new}}, \ldots \sim p_{\text{data}}$



Goal:

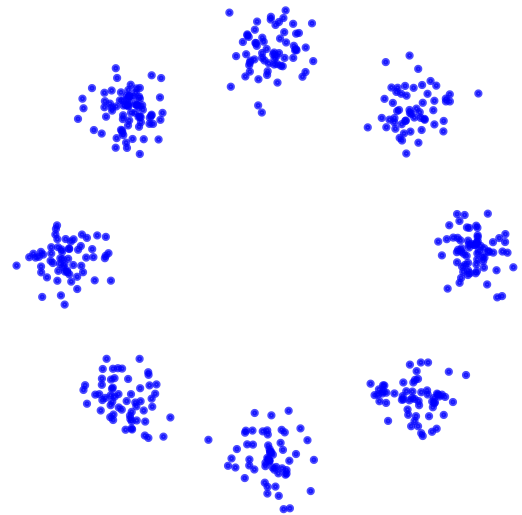
- Draw new samples $\underbrace{x^{(1)}_{\text{new}}, x^{(2)}_{\text{new}}, \ldots}_{\text{goal}} \sim \underbrace{p_{\text{data}}}_{\text{unknown}}$

A step back: what is generative modeling?

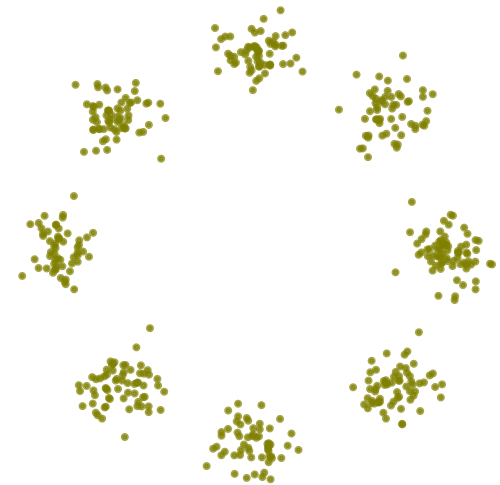
Problem setting:

- Access to n samples $x^{(1)}, \ldots, x^{(n)}$
- $\underbrace{x^{(1)}, \ldots, x^{(n)}}_{\text{known}} \sim \underbrace{p_{\text{data}}}_{\text{unknown}}$

samples $x^{(1)}, \ldots, x^{(n)} \sim p_{\text{data}}$



Goal: generate samples $x^{(1)}_{\text{new}}, x^{(2)}_{\text{new}}, \ldots \sim p_{\text{data}}$



Goal:

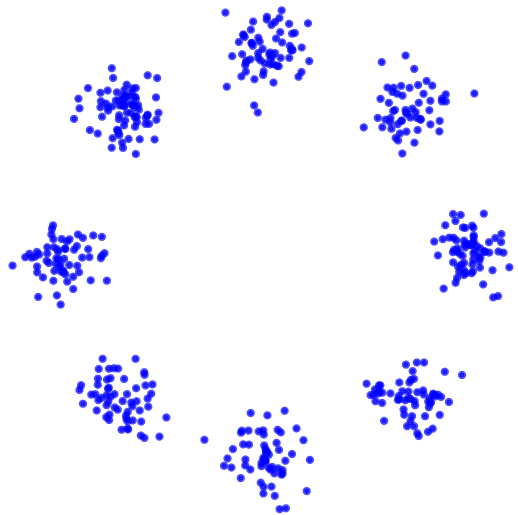
- Draw new samples $\underbrace{x^{(1)}_{\text{new}}, x^{(2)}_{\text{new}}, \ldots}_{\text{goal}} \sim \underbrace{p_{\text{data}}}_{\text{unknown}}$

A step back: what is generative modeling?

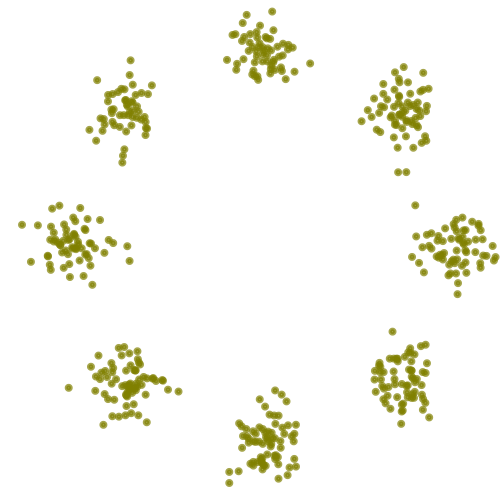
Problem setting:

- Access to n samples $x^{(1)}, \ldots, x^{(n)}$
- $\underbrace{x^{(1)}, \ldots, x^{(n)}}_{\text{known}} \sim \underbrace{p_{\text{data}}}_{\text{unknown}}$

samples $x^{(1)}, \ldots, x^{(n)} \sim p_{\text{data}}$



Goal: generate samples $x^{(1)}_{\text{new}}, x^{(2)}_{\text{new}}, \ldots \sim p_{\text{data}}$



Goal:

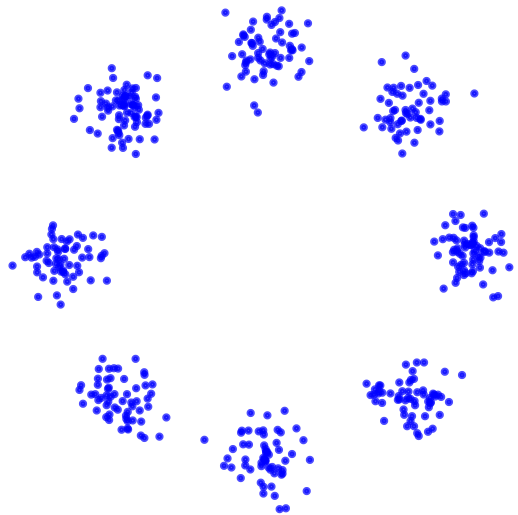
- Draw new samples $\underbrace{x^{(1)}_{\text{new}}, x^{(2)}_{\text{new}}, \ldots}_{\text{goal}} \sim \underbrace{p_{\text{data}}}_{\text{unknown}}$

A step back: what is generative modeling?

Problem setting:

- Access to n samples $x^{(1)}, \ldots, x^{(n)}$
- $\underbrace{x^{(1)}, \ldots, x^{(n)}}_{\text{known}} \sim \underbrace{p_{\text{data}}}_{\text{unknown}}$

samples $x^{(1)}, \ldots, x^{(n)} \sim p_{\text{data}}$



Goal: generate samples $x^{(1)}_{\text{new}}, x^{(2)}_{\text{new}}, \ldots \sim p_{\text{data}}$

Goal:

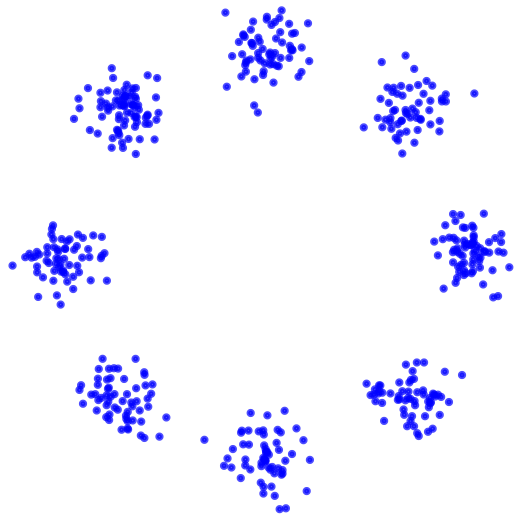
- Draw new samples $\underbrace{x^{(1)}_{\text{new}}, x^{(2)}_{\text{new}}, \ldots}_{\text{goal}} \sim \underbrace{p_{\text{data}}}_{\text{unknown}}$

A step back: what is generative modeling?

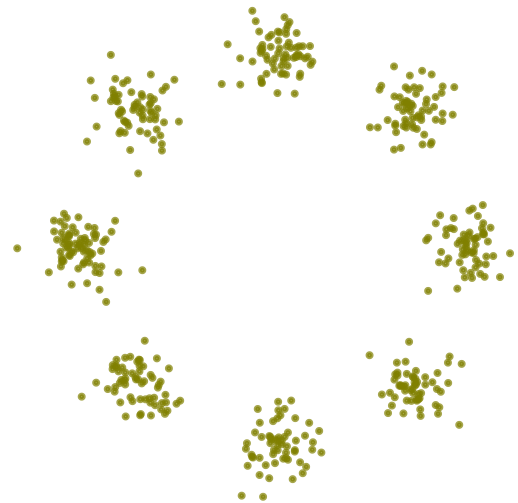
Problem setting:

- Access to n samples $x^{(1)}, \ldots, x^{(n)}$
- $\underbrace{x^{(1)}, \ldots, x^{(n)}}_{\text{known}} \sim \underbrace{p_{\text{data}}}_{\text{unknown}}$

samples $x^{(1)}, \ldots, x^{(n)} \sim p_{\text{data}}$



Goal: generate samples $x^{(1)}_{\text{new}}, x^{(2)}_{\text{new}}, \ldots \sim p_{\text{data}}$



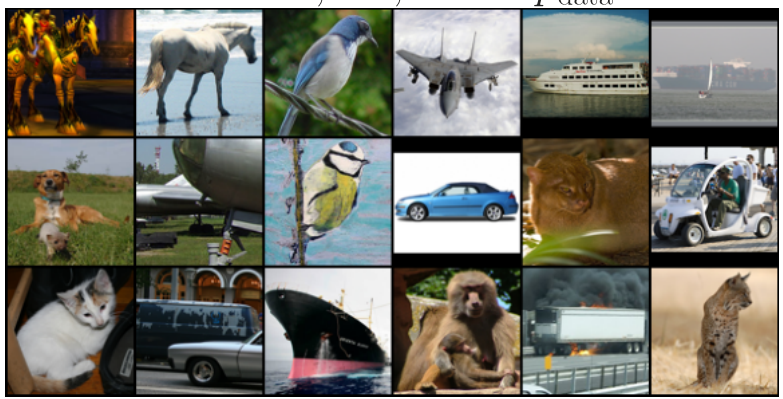
Goal:

- Draw new samples $\underbrace{x^{(1)}_{\text{new}}, x^{(2)}_{\text{new}}, \ldots}_{\text{goal}} \sim \underbrace{p_{\text{data}}}_{\text{unknown}}$

Generative modeling: an image example

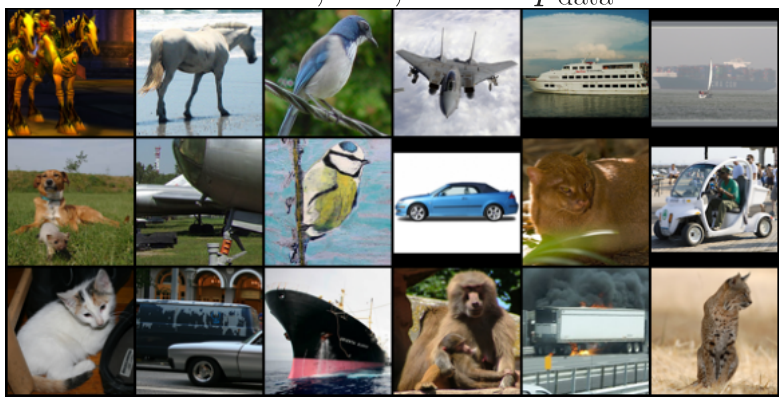
Generative modeling: an image example

data $x^{(1)}, \ldots, x^{(n)} \sim p_{\text{data}}$



Generative modeling: an image example

data $x^{(1)}, \ldots, x^{(n)} \sim p_{\text{data}}$

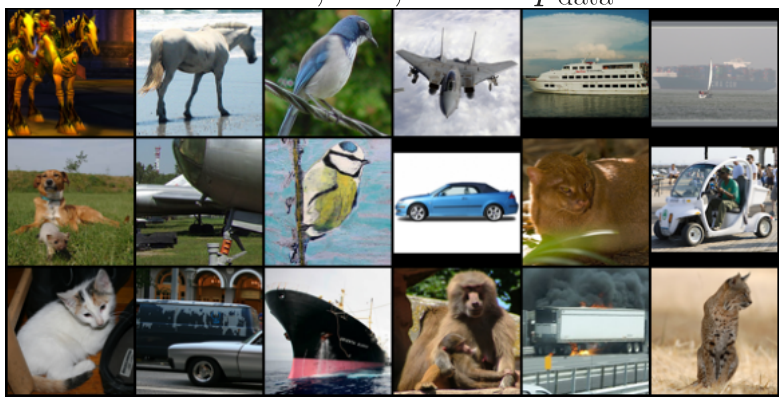


Data:

- Access to samples $x^{(1)}, \ldots, x^{(n)} \in \mathbb{R}^d$
- Drawn from p_{data} : $\underbrace{x^{(1)}, \ldots, x^{(n)}}_{\text{known}} \sim \underbrace{p_{\text{data}}}_{\text{unknown}}$
- Could also be audio, texts, proteins, etc

Generative modeling: an image example

data $x^{(1)}, \ldots, x^{(n)} \sim p_{\text{data}}$



Data:

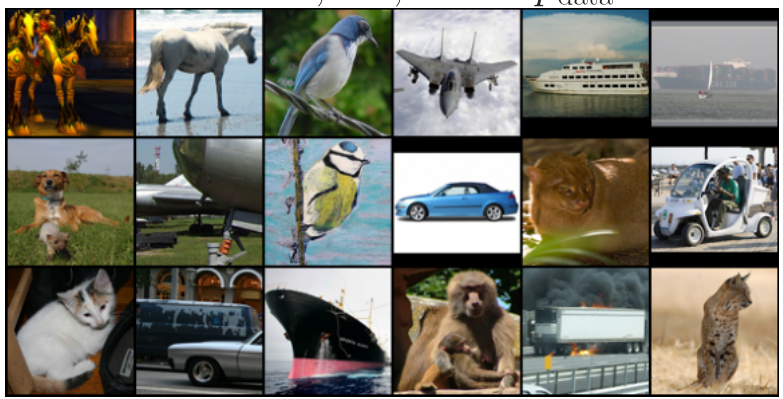
- Access to samples $x^{(1)}, \ldots, x^{(n)} \in \mathbb{R}^d$
- Drawn from p_{data} : $\underbrace{x^{(1)}, \ldots, x^{(n)}}_{\text{known}} \sim \underbrace{p_{\text{data}}}_{\text{unknown}}$
- Could also be audio, texts, proteins, etc

Goal:

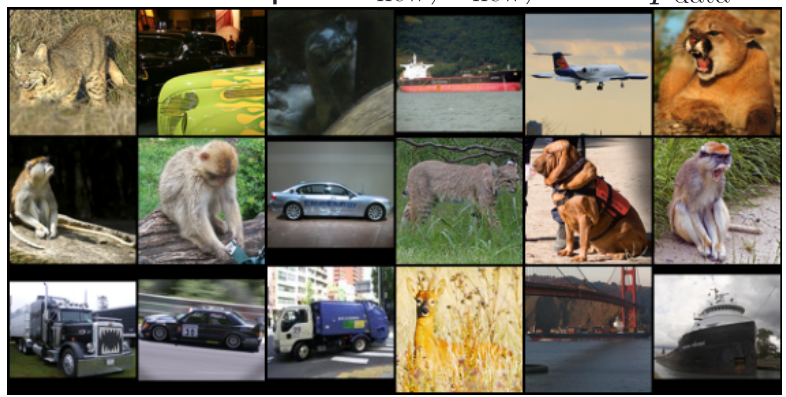
- Create new samples $\underbrace{x^{(1)}_{\text{new}}, x^{(2)}_{\text{new}}, \ldots}_{\text{goal}} \sim \underbrace{p_{\text{data}}}_{\text{unknown}}$

Generative modeling: an image example

data $x^{(1)}, \ldots, x^{(n)} \sim p_{\text{data}}$



new samples $x_{\text{new}}^{(1)}, x_{\text{new}}^{(2)}, \ldots \sim p_{\text{data}}$



Data:

- Access to samples $x^{(1)}, \ldots, x^{(n)} \in \mathbb{R}^d$
- Drawn from p_{data} : $\underbrace{x^{(1)}, \ldots, x^{(n)}}_{\text{known}} \sim \underbrace{p_{\text{data}}}_{\text{unknown}}$
- Could also be audio, texts, proteins, etc

Goal:

- Create new samples $\underbrace{x_{\text{new}}^{(1)}, x_{\text{new}}^{(2)}, \ldots}_{\text{goal}} \sim \underbrace{p_{\text{data}}}_{\text{unknown}}$

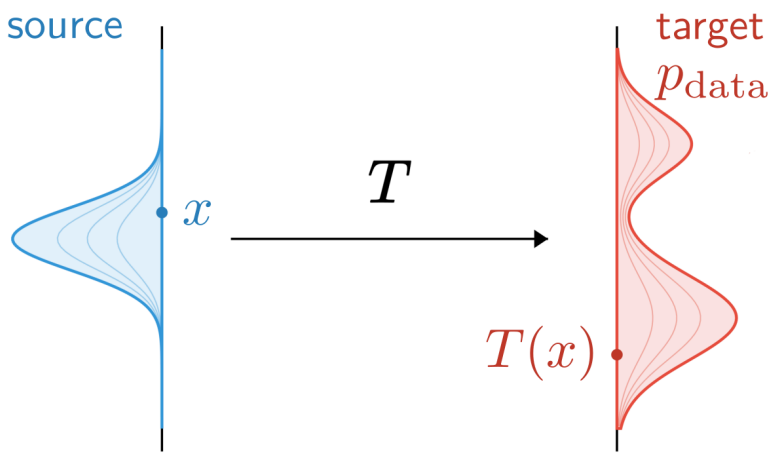
Diffusion and Flow matching: the big picture

Common idea:

Diffusion and Flow matching: the big picture

Common idea:

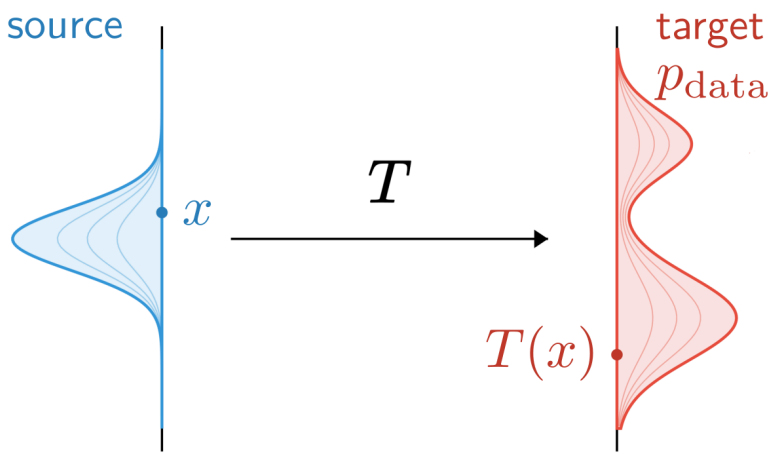
- Learn mapping T sending samples from source distribution to samples from target distribution p_{data}

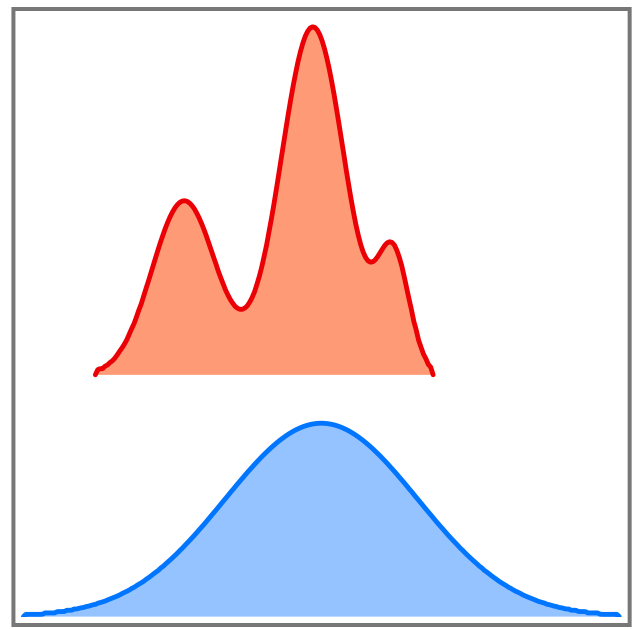


Diffusion and Flow matching: the big picture

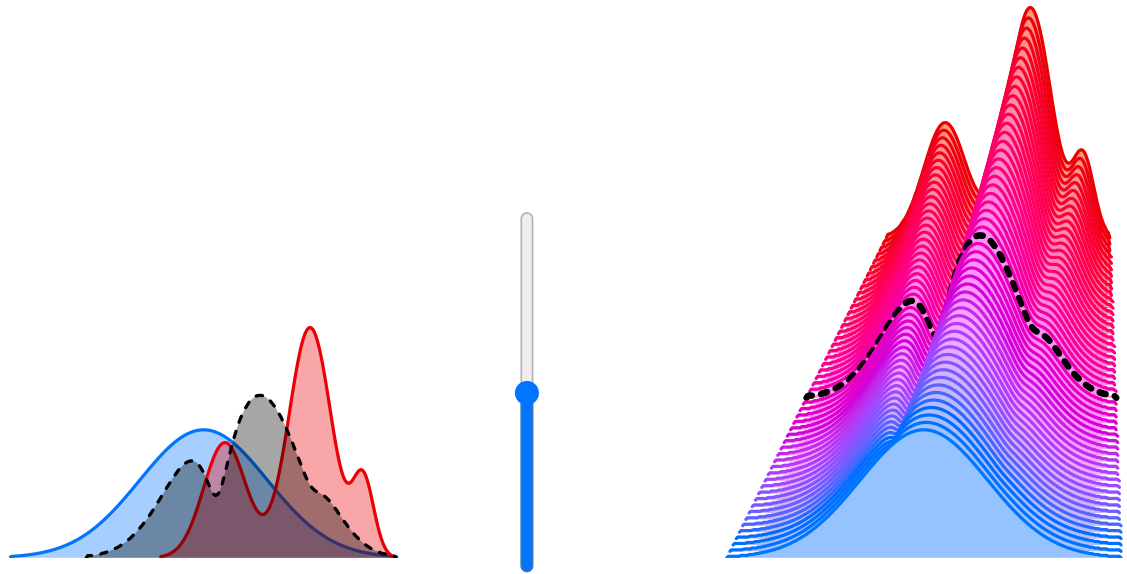
Common idea:

- Learn mapping T sending samples from source distribution to samples from target distribution p_{data}
- Rely on a continuous interpolation between the source and target distributions





Play with paths from to source and target distributions!



NB: Diffusion only supports Gaussian source distribution

Flow matching

Flow matching

- Convention: go from time $t = 0$ (source) to time $t = 1$ (target)

Flow matching

- Convention: go from time $t = 0$ (source) to time $t = 1$ (target)
- The source distribution is p_0 ; we'll take $p_0 = \mathcal{N}(0, \mathrm{Id}_d)$

Flow matching

- Convention: go from time $t = 0$ (source) to time $t = 1$ (target)
- The source distribution is p_0 ; we'll take $p_0 = \mathcal{N}(0, \mathrm{Id}_d)$
- Generate samples by solving an Ordinary Differential Equation (ODE):

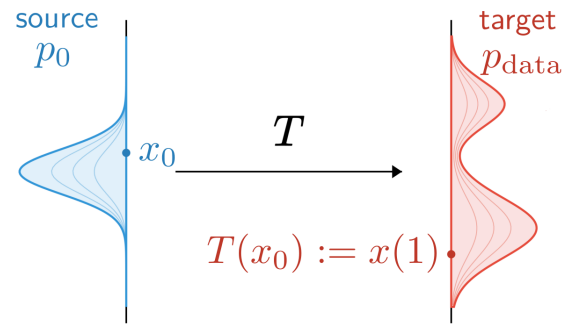
$$\begin{cases} x(0) = x_0 \sim p_0 \\ \partial_t x(t) = u(x(t), t) \quad \forall t \in [0, 1] \end{cases}$$

Flow matching

- Convention: go from time $t = 0$ (source) to time $t = 1$ (target)
- The source distribution is p_0 ; we'll take $p_0 = \mathcal{N}(0, \mathrm{Id}_d)$
- Generate samples by solving an Ordinary Differential Equation (ODE):

$$\begin{cases} x(0) = x_0 \sim p_0 \\ \partial_t x(t) = u(x(t), t) \quad \forall t \in [0, 1] \end{cases}$$

- The generated sample is defined as the solution at time 1, $x(1) \in \mathbb{R}^d$

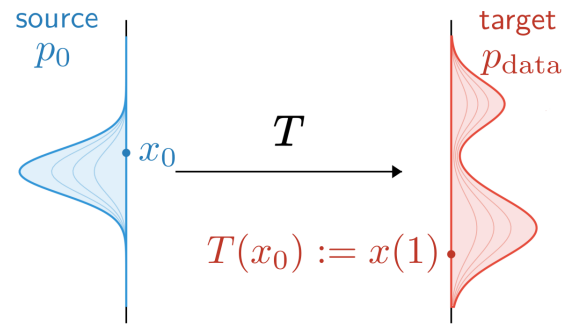


Flow matching

- Convention: go from time $t = 0$ (source) to time $t = 1$ (target)
- The source distribution is p_0 ; we'll take $p_0 = \mathcal{N}(0, \mathrm{Id}_d)$
- Generate samples by solving an Ordinary Differential Equation (ODE):

$$\begin{cases} x(0) = x_0 \sim p_0 \\ \partial_t x(t) = u(x(t), t) \quad \forall t \in [0, 1] \end{cases}$$

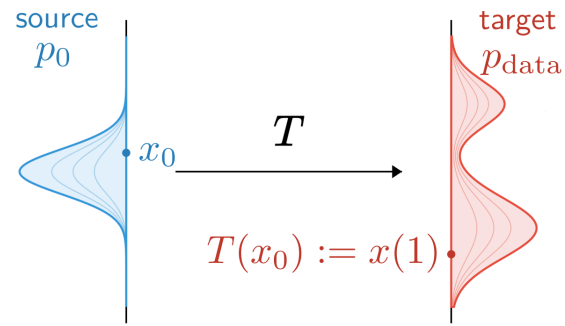
- The generated sample is defined as the solution at time 1, $x(1) \in \mathbb{R}^d$
- Key object: **velocity field** $u : \mathbb{R}^d \times [0, 1] \to \mathbb{R}^d$



Flow matching

- Convention: go from time $t = 0$ (source) to time $t = 1$ (target)
- The source distribution is p_0 ; we'll take $p_0 = \mathcal{N}(0, \mathrm{Id}_d)$
- Generate samples by solving an Ordinary Differential Equation (ODE):

$$\begin{cases} x(0) = x_0 \sim p_0 \\ \partial_t x(t) = u(x(t), t) \quad \forall t \in [0, 1] \end{cases}$$



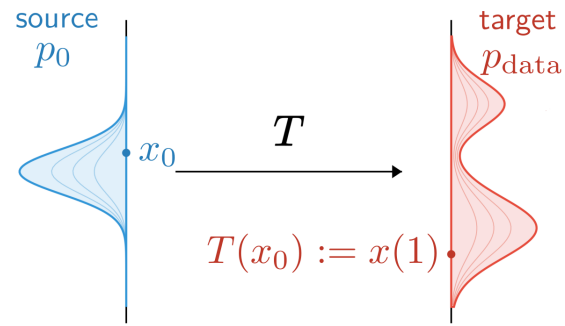
- The generated sample is defined as the solution at time 1, $x(1) \in \mathbb{R}^d$
- Key object: **velocity field** $u : \mathbb{R}^d \times [0, 1] \to \mathbb{R}^d$
- If velocity $u(x, t)$ is good, $x_0 \sim p_0 \implies x(1) \sim p_{\text{data}}$

  $x(1)$ is random because x_0 is

Flow matching

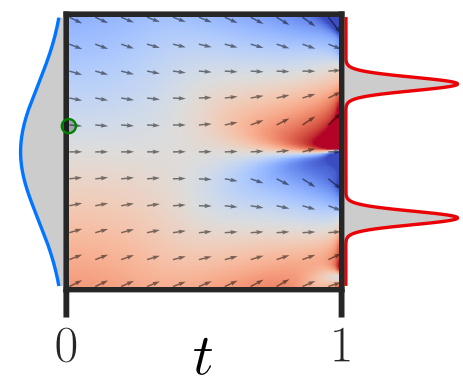
- Convention: go from time $t = 0$ (source) to time $t = 1$ (target)
- The source distribution is p_0 ; we'll take $p_0 = \mathcal{N}(0, \mathrm{Id}_d)$
- Generate samples by solving an Ordinary Differential Equation (ODE):

$$\begin{cases} x(0) = x_0 \sim p_0 \\ \partial_t x(t) = u(x(t), t) \quad \forall t \in [0, 1] \end{cases}$$



- The generated sample is defined as the solution at time 1, $x(1) \in \mathbb{R}^d$
- Key object: **velocity field** $u : \mathbb{R}^d \times [0, 1] \to \mathbb{R}^d$
- If velocity $u(x, t)$ is good, $x_0 \sim p_0 \implies x(1) \sim p_{\text{data}}$

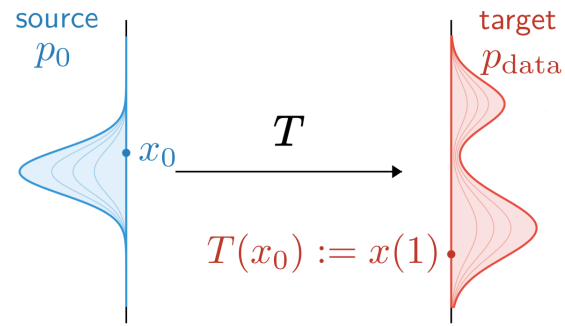
  $x(1)$ is random because x_0 is



Flow matching

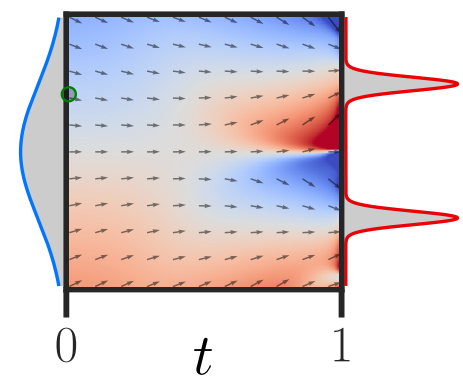
- Convention: go from time $t = 0$ (source) to time $t = 1$ (target)
- The source distribution is p_0 ; we'll take $p_0 = \mathcal{N}(0, \mathrm{Id}_d)$
- Generate samples by solving an Ordinary Differential Equation (ODE):

$$\begin{cases} x(0) = x_0 \sim p_0 \\ \partial_t x(t) = u(x(t), t) \quad \forall t \in [0, 1] \end{cases}$$



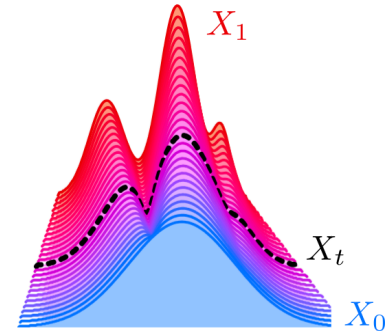
- The generated sample is defined as the solution at time 1, $x(1) \in \mathbb{R}^d$
- Key object: **velocity field** $u : \mathbb{R}^d \times [0, 1] \to \mathbb{R}^d$
- If velocity $u(x, t)$ is good, $x_0 \sim p_0 \implies x(1) \sim p_{\text{data}}$

  $x(1)$ is random because x_0 is



But how to find a *good* velocity $u(x, t)$?

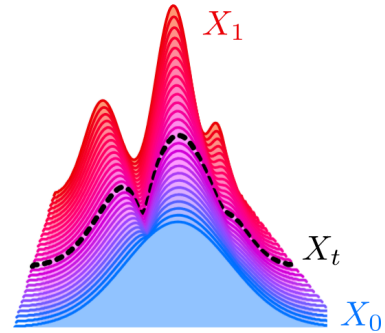
How to find a good velocity?



Setup: define three random variables

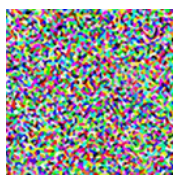
- $X_0 \sim p_0$ source distribution: we will take $p_0 = \mathcal{N}(0, \mathrm{Id}_d)$
- $X_1 \sim p_{\mathrm{data}}$ target distribution: clean image
- $X_t := (1-t)X_0 + tX_1$ interpolation: noised image

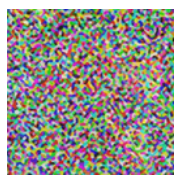
How to find a good velocity?

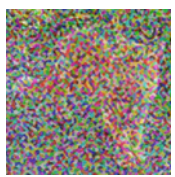


Setup: define three random variables

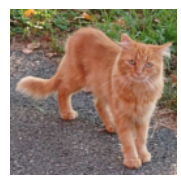
- $X_0 \sim p_0$ source distribution: we will take $p_0 = \mathcal{N}(0, \mathrm{Id}_d)$
- $X_1 \sim p_{\mathrm{data}}$ target distribution: clean image
- $X_t := (1-t)X_0 + tX_1$ interpolation: noised image


$X_{t=0}$

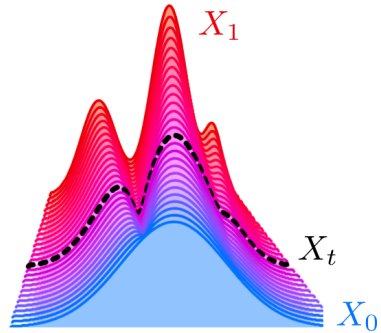

$X_{t=0.25}$


$X_{t=0.5}$


$X_{t=0.75}$

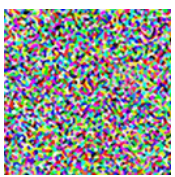

$X_{t=1}$

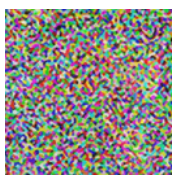
How to find a good velocity?

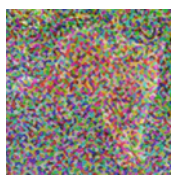


Setup: define three random variables

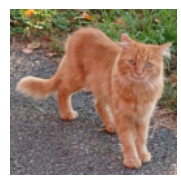
- $X_0 \sim p_0$ source distribution: we will take $p_0 = \mathcal{N}(0, \mathrm{Id}_d)$
- $X_1 \sim p_{\text{data}}$ target distribution: clean image
- $X_t := (1-t)X_0 + tX_1$ interpolation: noised image


$X_{t=0}$


$X_{t=0.25}$


$X_{t=0.5}$


$X_{t=0.75}$


$X_{t=1}$

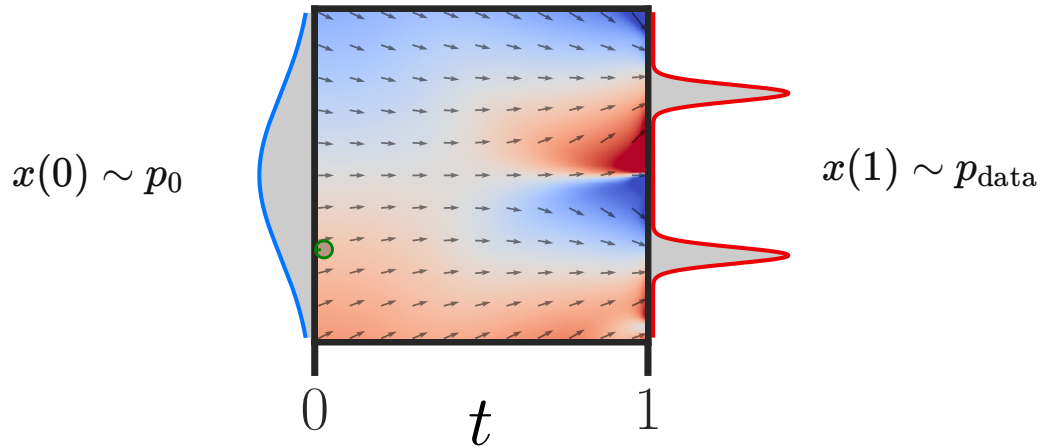
The FM magic[5,13,14] : $u(x,t) := \mathbb{E}[X_1 - X_0 | X_t = x]$ is a *good* velocity:

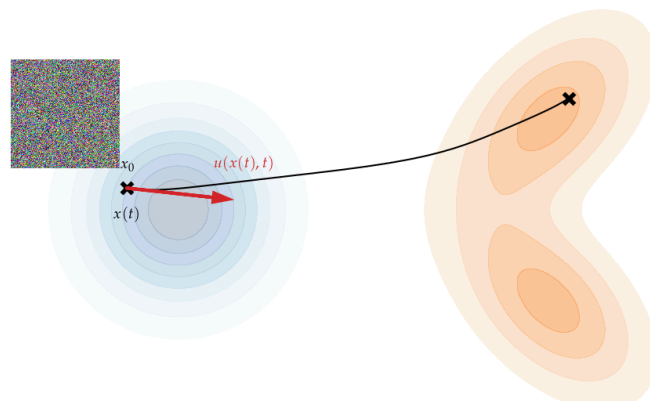
$$x(0) \sim p_0 \implies x(1) \sim p_{\text{data}}$$

[5] M. Albergo et al., **Building Normalizing Flows with Stochastic Interpolants,** In: ICLR, 2023.

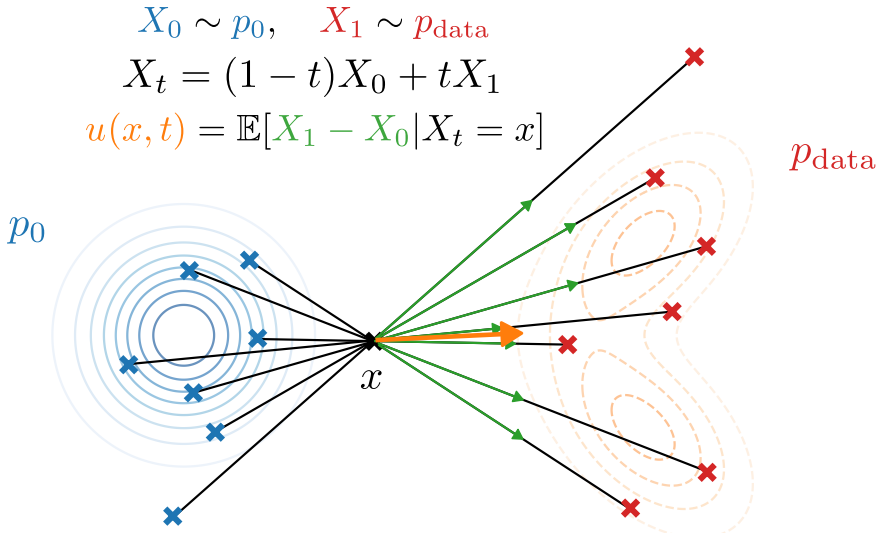
[13] Y. Lipman et al., **Flow Matching for Generative Modeling,** In: ICLR, 2023.

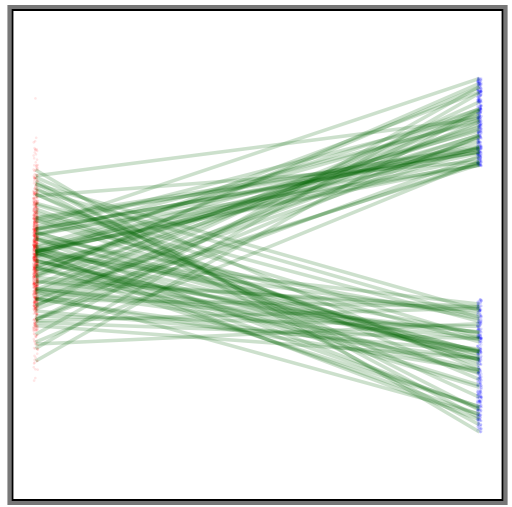
[14] X. Liu et al., **Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow,** In: ICLR, 2023.


$x(0) \sim p_0$
$x(1) \sim p_{\text{data}}$
0
t
1


x_0
$u(x(t), t)$
$x(t)$

Why the conditional expectation is "good"





More in our blog post: **A Visual dive into conditional flow matching**[10]: https://dl.heeere.com/cfm/

[10] A. Gagneux et al., **A Visual Dive into Conditional Flow Matching,** In: ICLR Blogposts, 2025.

How to train a Flow matching model?

$$\min_{u_\theta} \mathbb{E}_{\substack{x_0 \sim p_0 \\ x_1 \sim p_{\text{data}} \\ t \sim \text{unif}([0,1])}} \| u_\theta(\underbrace{(1-t)x_0 + t x_1}_{=x_t}, t) - (x_1 - x_0) \|^2$$

Train practically with:

$$\begin{aligned} x_0 &\sim p_0 = \mathcal{N}(0, \text{Id}_d) \\ x_1 &\sim p_{\text{data}} \\ t &\sim \text{Uniform}([0,1]) \\ x_t &= (1-t)x_0 + t x_1 \end{aligned}$$

SGD step on θ with loss $\| u_\theta(x_t, t) - (x_1 - x_0) \|^2$

How to train a Flow matching model?

$$\min_{u_\theta} \mathbb{E}_{\substack{x_0 \sim p_0 \\ x_1 \sim p_{\text{data}} \\ t \sim \text{unif}([0,1])}} \| u_\theta(\underbrace{(1-t)x_0 + t x_1}_{=x_t}, t) - (x_1 - x_0) \|^2$$

Train practically with:

$$\begin{aligned} x_0 &\sim p_0 = \mathcal{N}(0, \text{Id}_d) \\ x_1 &\sim p_{\text{data}} \\ t &\sim \text{Uniform}([0,1]) \\ x_t &= (1-t)x_0 + t x_1 \end{aligned}$$

SGD step on θ with loss $\| u_\theta(x_t, t) - (x_1 - x_0) \|^2$

How to train a Flow matching model?

$$\min_{u_\theta} \mathbb{E}_{\substack{x_0 \sim p_0 \\ x_1 \sim p_{\text{data}} \\ t \sim \text{unif}([0,1])}} \| u_\theta(\underbrace{(1-t)x_0 + t x_1}_{=x_t}, t) - (x_1 - x_0) \|^2$$

Train practically with:

$$\begin{aligned} x_0 &\sim p_0 = \mathcal{N}(0, \text{Id}_d) \\ x_1 &\sim p_{\text{data}} \\ t &\sim \text{Uniform}([0,1]) \\ x_t &= (1-t)x_0 + t x_1 \end{aligned}$$

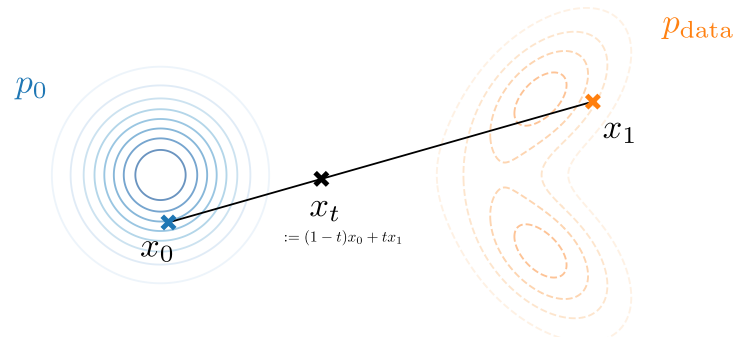
SGD step on θ with loss $\| u_\theta(x_t, t) - (x_1 - x_0) \|^2$

How to train a Flow matching model?

$$\min_{u_\theta} \mathbb{E}_{\substack{x_0 \sim p_0 \\ x_1 \sim p_{\text{data}} \\ t \sim \text{unif}([0,1])}} \| u_\theta(\underbrace{(1-t)x_0 + t x_1}_{=x_t}, t) - (x_1 - x_0) \|^2$$

Train practically with:

$$\begin{aligned} x_0 &\sim p_0 = \mathcal{N}(0, \text{Id}_d) \\ x_1 &\sim p_{\text{data}} \\ t &\sim \text{Uniform}([0,1]) \\ x_t &= (1-t)x_0 + t x_1 \end{aligned}$$



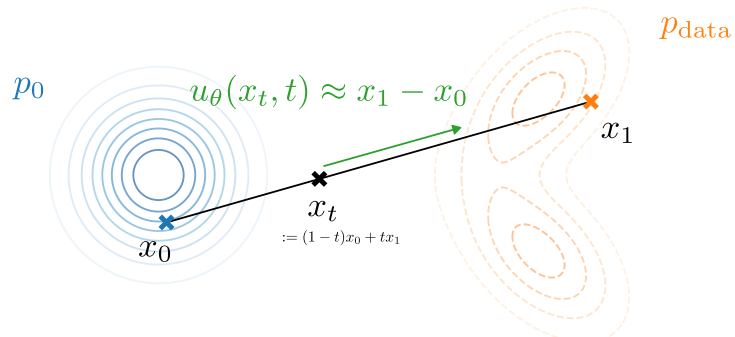
SGD step on θ with loss $\| u_\theta(x_t, t) - (x_1 - x_0) \|^2$

How to train a Flow matching model?

$$\min_{u_\theta} \mathbb{E}_{\substack{x_0 \sim p_0 \\ x_1 \sim p_{\text{data}} \\ t \sim \text{unif}([0,1])}} \| u_\theta(\underbrace{(1-t)x_0 + t x_1}_{=x_t}, t) - (x_1 - x_0) \|^2$$

Train practically with:

$$\begin{aligned} x_0 &\sim p_0 = \mathcal{N}(0, \text{Id}_d) \\ x_1 &\sim p_{\text{data}} \\ t &\sim \text{Uniform}([0,1]) \\ x_t &= (1-t)x_0 + t x_1 \end{aligned}$$



SGD step on θ with loss $\| u_\theta(x_t, t) - (x_1 - x_0) \|^2$

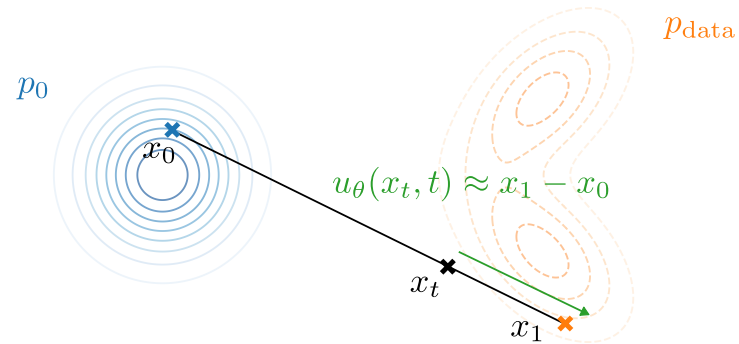
How to train a Flow matching model?

$$\min_{u_\theta} \mathbb{E}_{\substack{x_0 \sim p_0 \\ x_1 \sim p_{\text{data}} \\ t \sim \text{unif}([0,1])}} \| u_\theta(\underbrace{(1-t)x_0 + t x_1}_{=x_t}, t) - (x_1 - x_0) \|^2$$

Train practically with:

$$\begin{aligned} x_0 &\sim p_0 = \mathcal{N}(0, \text{Id}_d) \\ x_1 &\sim p_{\text{data}} \\ t &\sim \text{Uniform}([0,1]) \\ x_t &= (1-t)x_0 + t x_1 \end{aligned}$$

SGD step on θ with loss $\| u_\theta(x_t, t) - (x_1 - x_0) \|^2$

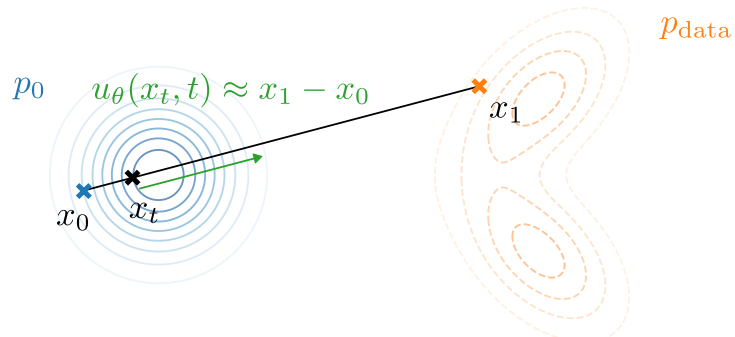


How to train a Flow matching model?

$$\min_{u_\theta} \mathbb{E}_{\substack{x_0 \sim p_0 \\ x_1 \sim p_{\text{data}} \\ t \sim \text{unif}([0,1])}} \| u_\theta(\underbrace{(1-t)x_0 + t x_1}_{=x_t}, t) - (x_1 - x_0) \|^2$$

Train practically with:

$$\begin{aligned} x_0 &\sim p_0 = \mathcal{N}(0, \text{Id}_d) \\ x_1 &\sim p_{\text{data}} \\ t &\sim \text{Uniform}([0,1]) \\ x_t &= (1-t)x_0 + t x_1 \end{aligned}$$



SGD step on θ with loss $\| u_\theta(x_t, t) - (x_1 - x_0) \|^2$

How to train a Flow matching model?

$$\min_{u_\theta} \mathbb{E}_{\substack{x_0 \sim p_0 \\ x_1 \sim p_{\text{data}} \\ t \sim \text{unif}([0,1])}} \| u_\theta(\underbrace{(1-t)x_0 + t x_1}_{=x_t}, t) - (x_1 - x_0) \|^2$$

Train practically with:

$$\begin{aligned} x_0 &\sim p_0 = \mathcal{N}(0, \text{Id}_d) \\ x_1 &\sim p_{\text{data}} \\ t &\sim \text{Uniform}([0,1]) \\ x_t &= (1-t)x_0 + t x_1 \end{aligned}$$

SGD step on θ with loss $\| u_\theta(x_t, t) - (x_1 - x_0) \|^2$

few days of training and you're done

How to train a Flow matching model?

$$\min_{u_\theta} \mathbb{E}_{\substack{x_0 \sim p_0 \\ x_1 \sim p_{\text{data}} \\ t \sim \text{unif}([0,1])}} \| u_\theta(\underbrace{(1-t)x_0 + t x_1}_{=x_t}, t) - (x_1 - x_0) \|^2$$

Train practically with:

$$\begin{aligned} x_0 &\sim p_0 = \mathcal{N}(0, \text{Id}_d) \\ x_1 &\sim p_{\text{data}} \\ t &\sim \text{Uniform}([0,1]) \\ x_t &= (1-t)x_0 + t x_1 \end{aligned}$$

SGD step on θ with loss $\| u_\theta(x_t, t) - (x_1 - x_0) \|^2$

few days of training and you're done

the minimizer is $u(x,t) = \mathbb{E}[X_1 - X_0 | X_t = x]$

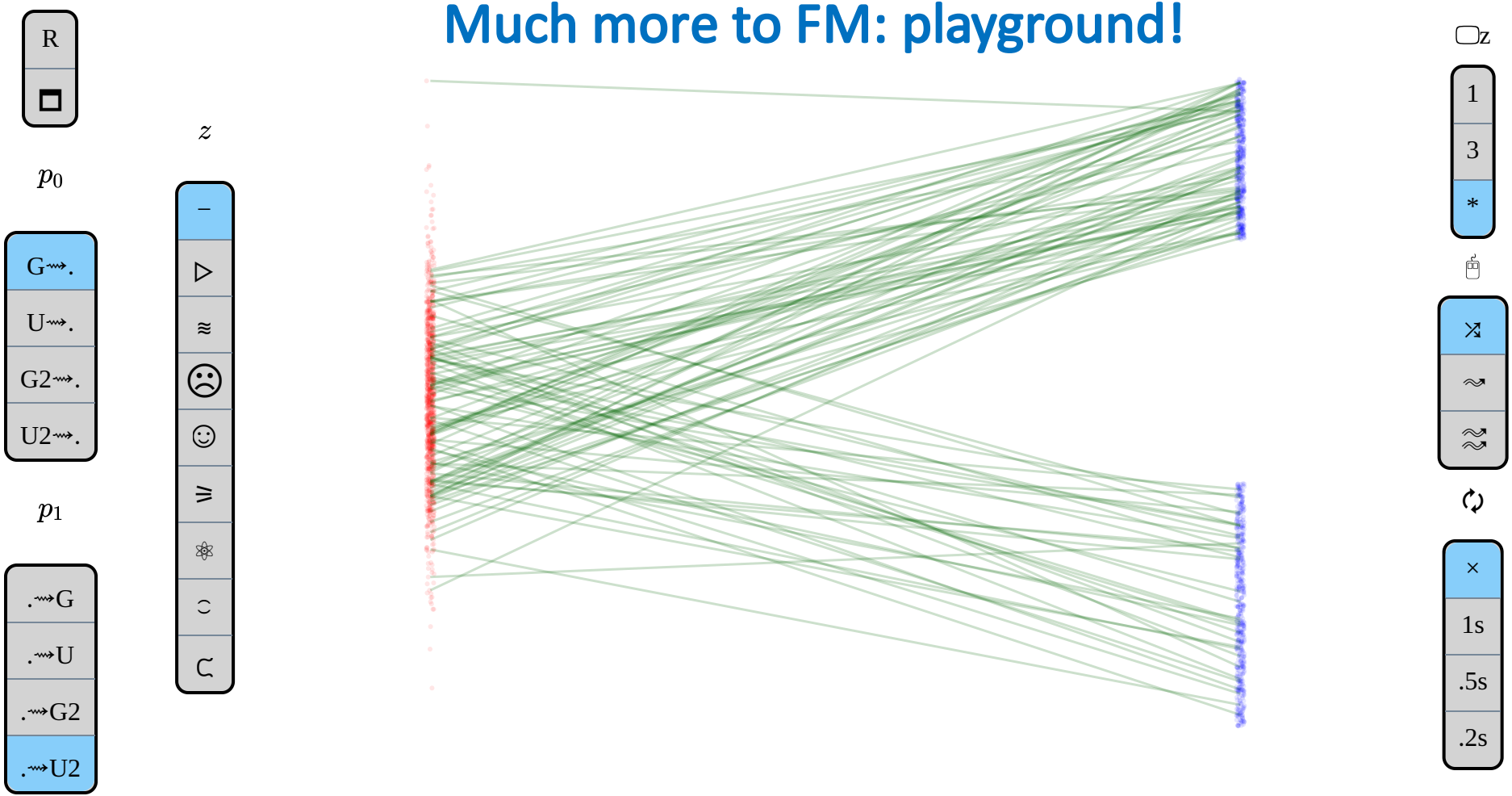
the minimizer (over all measurable functions)

Bonus: much more to FM[15,16]

- Source distribution p_0 can be arbitrary
- Time sampling need not be uniform
- In training, x_0 and x_1 need not be sampled independently[17,18]
- X_t need not equal $(1-t)X_0 + tX_1$
- In practice guidance/conditioning is used a lot

[15] E. Pierret et al., **Flow Matching for Applied Mathematicians,** In: , 2026.
[16] Y. Lipman et al., **Flow matching guide and code,** In: arXiv preprint arXiv:2412.06264, 2024.
[17] A. Pooladian et al., **Multisample flow matching: Straightening flows with minibatch couplings,** In: ICML, 2023.
[18] A. Tong et al., **Improving and generalizing flow-based generative models with minibatch optimal transport,** In: TMLR, 2023.

Much more to FM: playground!



Diffusion

Only thing to know: it's equivalent to Flow matching[19]!

Diffusion Meets Flow Matching: Two Sides of the Same Coin

https://diffusionflow.github.io/

[19] R. Gao et al., **Diffusion Meets Flow Matching: Two Sides of the Same Coin,** In: ICLR Blogposts, 2025.

Diffusion

Diffusion

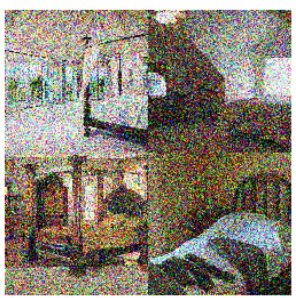
- Convention differs from Flow matching: $t = 0$ corresponds to clean data

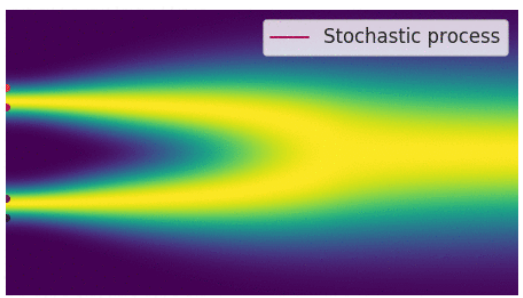
Diffusion

- Convention differs from Flow matching: $t = 0$ corresponds to clean data
- Similar idea: progressively corrupt image by adding Gaussian noise

Diffusion

- Convention differs from Flow matching: $t = 0$ corresponds to clean data
- Similar idea: progressively corrupt image by adding Gaussian noise
- Trajectories are stochastic

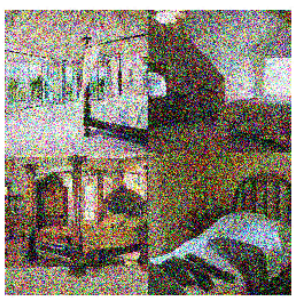


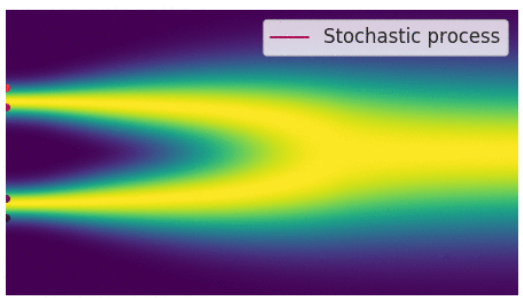


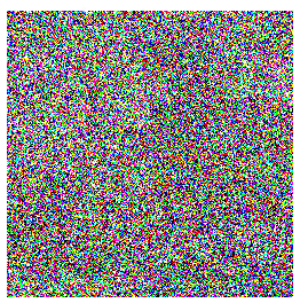
https://yang-song.net/blog/2021/score/

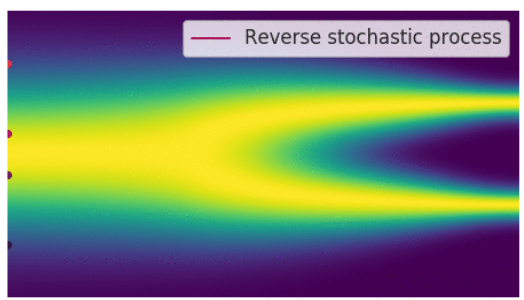
Diffusion

- Convention differs from Flow matching: $t = 0$ corresponds to clean data
- Similar idea: progressively corrupt image by adding Gaussian noise
- Trajectories are stochastic
- One learns to reverse the noising process





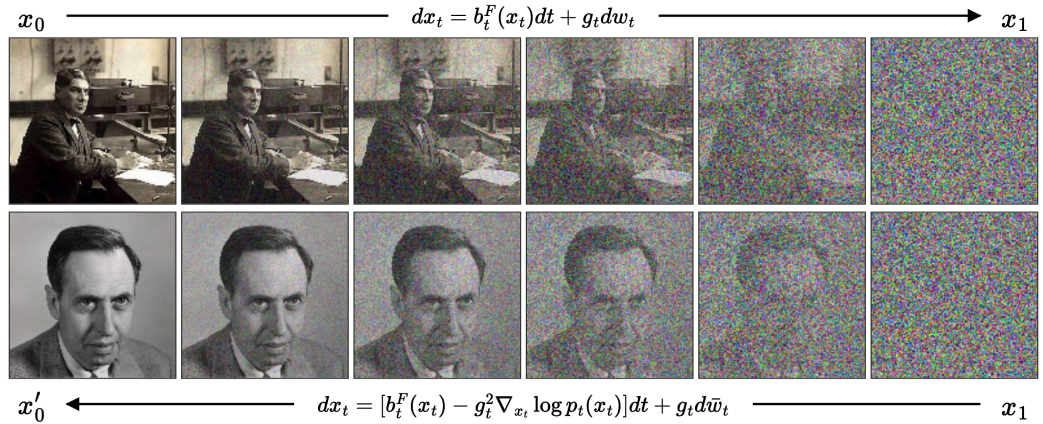




https://yang-song.net/blog/2021/score/

Learning Diffusion

- Instead of learning the velocity u, diffusion learns to reverse the corruption process
- It is driven by a Stochastic Differential Equation (SDE)



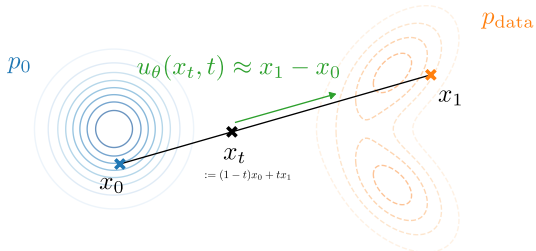
- A key quantity to learn is the *score* $\nabla \log p_t(x_t) \in \mathbb{R}^d$ (p_t is the density of X_t)

  It is theoretically equivalent to learn to predict the score, the added noise or the original image from the corrupted image x_t

- Training procedure is similar to Flow matching: sample clean image, noise it, learn to denoise it

Memorization in Diffusion & Flow Matching

A small caveat in training



Flow matching training:

$$
\begin{aligned}
x_0 &\sim p_0 = \mathcal{N}(0, \mathrm{Id}_d) \\
x_1 &\sim p_{\text{data}} \\
t &\sim \mathrm{Uniform}([0,1]) \\
x_t &= (1-t)x_0 + tx_1
\end{aligned}
$$

SGD step on θ with loss $\|u_\theta(x_t, t) - (x_1 - x_0)\|^2$

Flow matching code

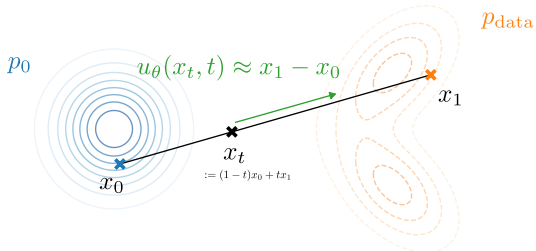
Flow Matching

Yaron Lipman[1], Marton Havas
Ricky T. Q. Chen[1], David Lope
[1]FAIR at Meta, [2]MIT CSAI

```
24  # training
25  flow = Flow()
26  optimizer = torch.optim.Adam(flow.parameters(), 1e-2)
27  loss_fn = nn.MSELoss()
28
29  for _ in range(10000):
30    x_1  = Tensor(make_moons(256, noise=0.05)[0])
31    x_0  = torch.randn_like(x_1)
32    t    = torch.rand(len(x_1), 1)
33    x_t  = (1 - t) * x_0 + t * x_1
34    dx_t = x_1 - x_0
35    optimizer.zero_grad()
36    loss_fn(flow(x_t, t), dx_t).backward()
37    optimizer.step()
38
```

No si
ODE

A small caveat in training



Flow matching training:

$$x_0 \sim p_0 = \mathcal{N}(0, \mathrm{Id}_d)$$
$$x_1 \sim p_{\text{data}}$$
$$t \sim \mathrm{Uniform}([0,1])$$
$$x_t = (1-t)x_0 + tx_1$$

SGD step on θ with loss $\|u_\theta(x_t, t) - (x_1 - x_0)\|^2$

Flow matching code

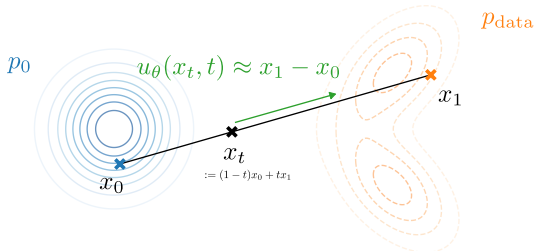
Flow Matching

Yaron Lipman[1], Marton Havas
Ricky T. Q. Chen[1], David Lope
[1]FAIR at Meta, [2]MIT CSAI

```
24  # training
25  flow = Flow()
26  optimizer = torch.optim.Adam(flow.parameters(), 1e-2)
27  loss_fn = nn.MSELoss()
28
29  for _ in range(10000):
30      x_1  = Tensor(make_moons(256, noise=0.05)[0])
31      x_0  = torch.randn_like(x_1)
32      t    = torch.rand(len(x_1), 1)
33      x_t  = (1 - t) * x_0 + t * x_1
34      dx_t = x_1 - x_0
35      optimizer.zero_grad()
36      loss_fn(flow(x_t, t), dx_t).backward()
37      optimizer.step()
38
```

No si
ODE

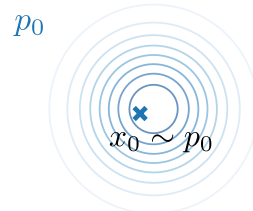
A small caveat in training

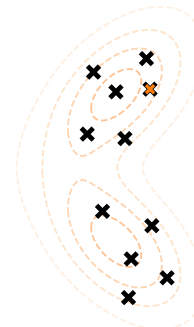


Flow matching training:

$$
\begin{aligned}
x_0 &\sim p_0 = \mathcal{N}(0, \mathrm{Id}_d) \\
x_1 &\sim p_{\text{data}} \\
t &\sim \mathrm{Uniform}([0,1]) \\
x_t &= (1-t)x_0 + tx_1
\end{aligned}
$$

SGD step on θ with loss $\|u_\theta(x_t, t) - (x_1 - x_0)\|^2$





$\hat{p}_{\text{data}} := \frac{1}{n} \sum_{i=1}^{n} \delta_{x^{(i)}}$

$x_1 \in \{x^{(1)}, \ldots, x^{(n)}\}$

only usable loss: $\mathbb{E}_{\substack{x_0 \sim p_0 \\ x_1 \sim \hat{p}_{\text{data}} \\ t \sim \text{unif}([0,1])}} \|u_\theta((1-t)x_0 + tx_1, t) - (x_1 - x_0)\|^2$

Expectation vs reality

$$\text{only usable loss:} \quad \mathbb{E}_{\substack{x_0 \sim p_0 \\ x_1 \sim \hat{p}_{\text{data}} \\ t \sim \text{unif}([0,1])}} \left\| u_\theta((1-t)x_0 + t x_1, t) - (x_1 - x_0) \right\|^2$$

Proposition: when p_{data} is replaced by $\hat{p}_{\text{data}} := \frac{1}{n} \sum_{i=1}^n \delta_{x^{(i)}}$, the FM loss minimizer has a closed-form:

$$\hat{u}^\star(x,t) := \sum_{i=1}^n \lambda_i(x,t) \frac{x^{(i)} - x}{1-t}$$

with $\lambda_i(x,t) = \dfrac{\exp(-\frac{1}{2(1-t)^2}\|x - t x^{(i)}\|^2)}{\sum_{i'=1}^n \exp(-\frac{1}{2(1-t)^2}\|x - t x^{(i')}\|^2)}$

$$\lambda(x,t) = \text{softmax}((-\tfrac{1}{2(1-t)^2}\|x - t x^{(i')}\|^2)_{i'=1,\dots,n}) \in \mathbb{R}^n$$

Expectation vs reality

$$\text{only usable loss:} \quad \mathbb{E}_{\substack{x_0 \sim p_0 \\ x_1 \sim \hat{p}_{\text{data}} \\ t \sim \text{unif}([0,1])}} \left\| u_\theta((1-t)x_0 + t x_1, t) - (x_1 - x_0) \right\|^2$$

Proposition: when p_{data} is replaced by $\hat{p}_{\text{data}} := \frac{1}{n}\sum_{i=1}^n \delta_{x^{(i)}}$, the FM loss minimizer has a closed-form:

$$\hat{u}^\star(x,t) := \sum_{i=1}^n \lambda_i(x,t) \frac{x^{(i)} - x}{1-t}$$

with $\lambda_i(x,t) = \dfrac{\exp(-\frac{1}{2(1-t)^2}\|x - t x^{(i)}\|^2)}{\sum_{i'=1}^n \exp(-\frac{1}{2(1-t)^2}\|x - t x^{(i')}\|^2)}$

$$\lambda(x,t) = \text{softmax}((-\tfrac{1}{2(1-t)^2}\|x - t x^{(i')}\|^2)_{i'=1,\dots,n}) \in \mathbb{R}^n$$

$\hat{u}^\star$ blows up as $t \to 1$ unless $x \approx x^{(i)}$

Expectation vs reality

$$\text{only usable loss:} \quad \mathbb{E}_{\substack{x_0 \sim p_0 \\ x_1 \sim \hat{p}_{\text{data}} \\ t \sim \text{unif}([0,1])}} \left\| u_\theta((1-t)x_0 + t x_1, t) - (x_1 - x_0) \right\|^2$$

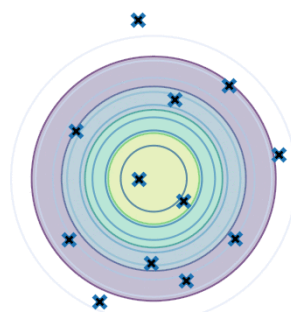
Proposition: when p_{data} is replaced by $\hat{p}_{\text{data}} := \frac{1}{n}\sum_{i=1}^n \delta_{x^{(i)}}$, the FM loss minimizer has a closed-form:

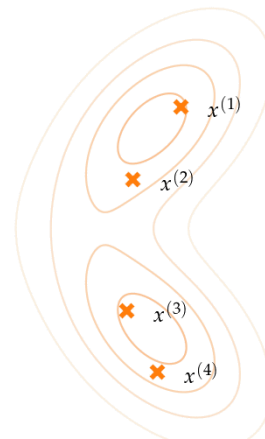
$$\hat{u}^\star(x,t) := \sum_{i=1}^n \lambda_i(x,t) \frac{x^{(i)} - x}{1-t}$$

with $\lambda_i(x,t) = \dfrac{\exp(-\frac{1}{2(1-t)^2}\|x - t x^{(i)}\|^2)}{\sum_{i'=1}^n \exp(-\frac{1}{2(1-t)^2}\|x - t x^{(i')}\|^2)}$

$$\lambda(x,t) = \text{softmax}((-\tfrac{1}{2(1-t)^2}\|x - t x^{(i')}\|^2)_{i'=1,\dots,n}) \in \mathbb{R}^n$$

$\hat{u}^\star$ blows up as $t \to 1$ unless $x \approx x^{(i)}$





it only generates training points!

What about diffusion?

- Training diffusion is a particular instance of Flow matching parametrization
- The ideal diffusion score $s^\star(x,t) = \nabla \log p_t(x)$ is related to the ideal velocity by $s(x,t) = \frac{tu^\star(x,t)-x}{1-t}$
- Closed-form minimizer for velocity $\implies$ closed-form minimizer for score! (which also blows up)

Diffusion suffers from the same curse!

Wrap-up of Part I

Wrap-up of Part I

- Flow matching & Diffusion: powerful generative models

Wrap-up of Part I

- Flow matching & Diffusion: powerful generative models
- Simple training algorithms

Wrap-up of Part I

- Flow matching & Diffusion: powerful generative models
- Simple training algorithms
- Closed-form for the ideal score/velocity

Wrap-up of Part I

- Flow matching & Diffusion: powerful generative models
- Simple training algorithms
- Closed-form for the ideal score/velocity
- Closed-form can only generate training samples

Wrap-up of Part I

- Flow matching & Diffusion: powerful generative models
- Simple training algorithms
- Closed-form for the ideal score/velocity
- Closed-form can only generate training samples
- Properly trained Flow matching and diffusion should only generate training data (like all unregularized generative models)

Wrap-up of Part I

- Flow matching & Diffusion: powerful generative models
- Simple training algorithms
- Closed-form for the ideal score/velocity
- Closed-form can only generate training samples
- Properly trained Flow matching and diffusion should only generate training data (like all unregularized generative models)
- So why do they generate new samples?