Tutorial on Diffusion Models

Tutorial on ~~Diffusion~~ Conditional Flow Matching

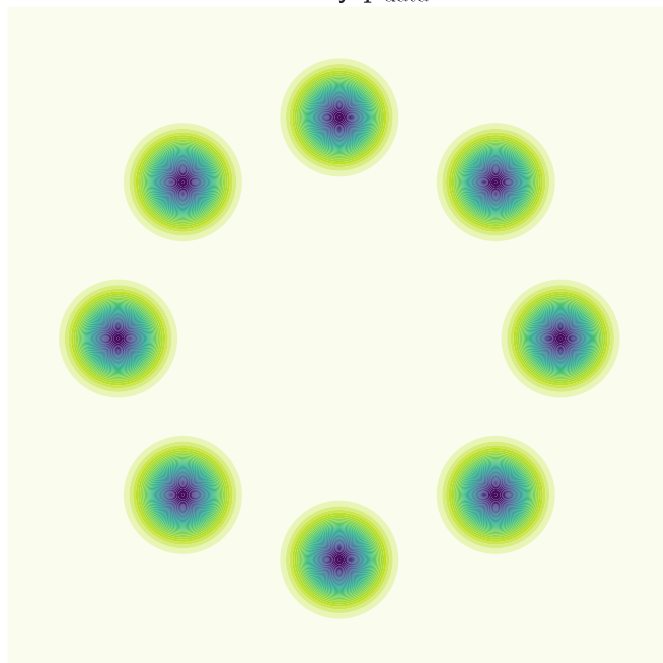
Quentin Bertrand
QB3.github.io
Inria | Laboratoire Hubert Curien | Mila Affiliated Member | CIFAR Global Scholar

anr agence nationale de la recherche
NSERC CRSNG
IVADO

What is Generative Modeling? A 2D-Example

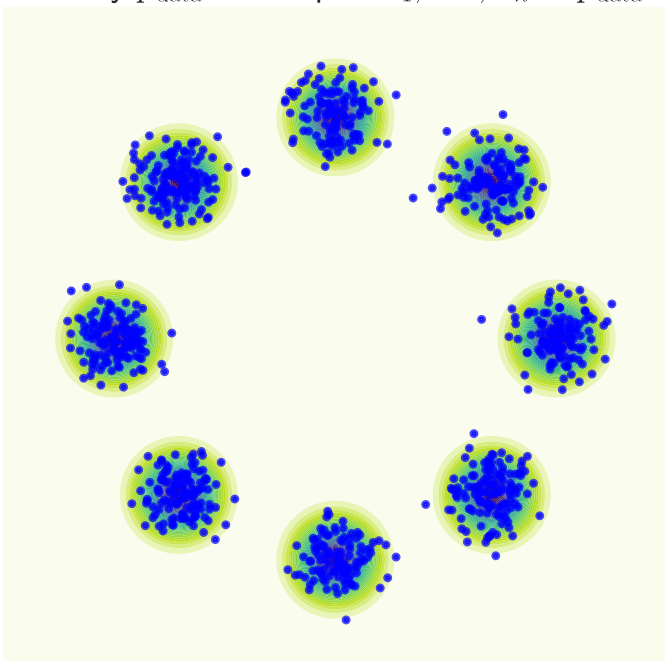
What is Generative Modeling? A 2D-Example

Density p_{data}



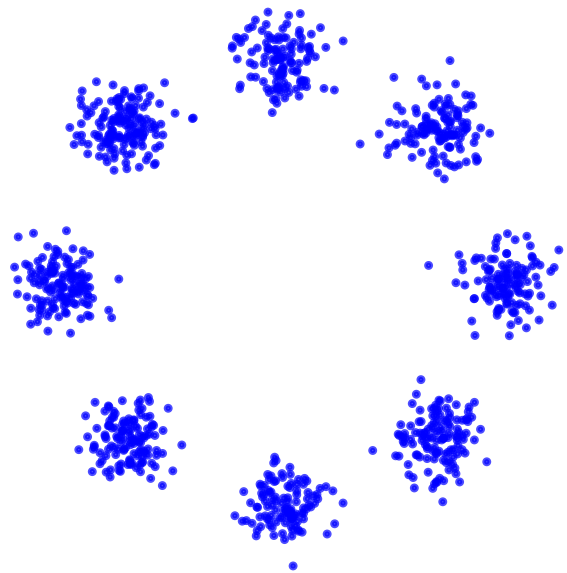
What is Generative Modeling? A 2D-Example

Density p_{data} + Samples $x_1, \ldots, x_n \sim p_{\text{data}}$



What is Generative Modeling? A 2D-Example

Samples $x_1, \ldots, x_n \sim p_{\text{data}}$

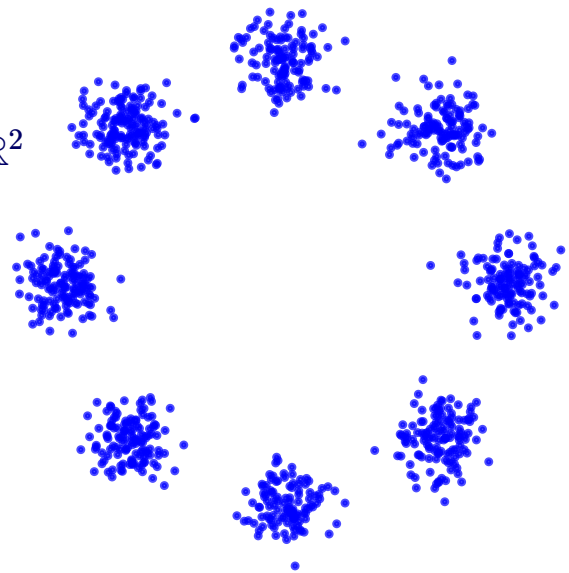


What is Generative Modeling? A 2D-Example

Samples $x_1, \ldots, x_n \sim p_{\text{data}}$

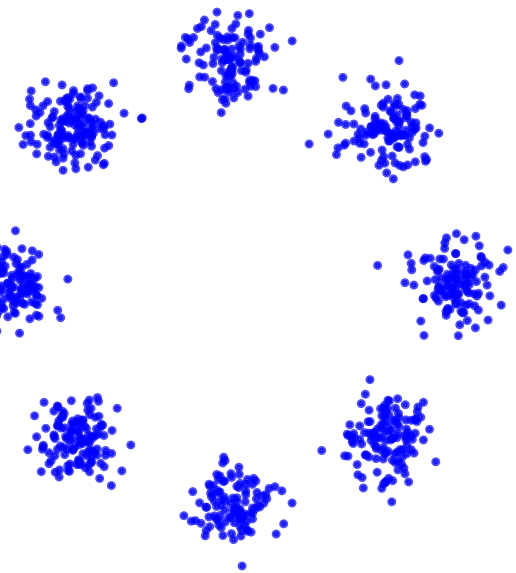
Problem Setting:

- Access to n samples $x_1, \ldots, x_n \in \mathbb{R}^2$
- $\underbrace{x_1, \ldots, x_n}_{\text{known}} \sim \underbrace{p_{\text{data}}}_{\text{unknown}}$



What is Generative Modeling? A 2D-Example

Samples $x_1, \ldots, x_n \sim p_{\text{data}}$



Problem Setting:

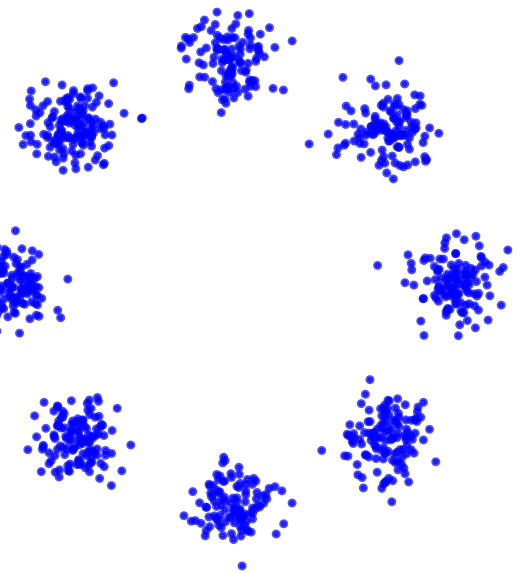
- Access to n samples $x_1, \ldots, x_n \in \mathbb{R}^2$
- $\underbrace{x_1, \ldots, x_n}_{\text{known}} \sim \underbrace{p_{\text{data}}}_{\text{unknown}}$

Goal:

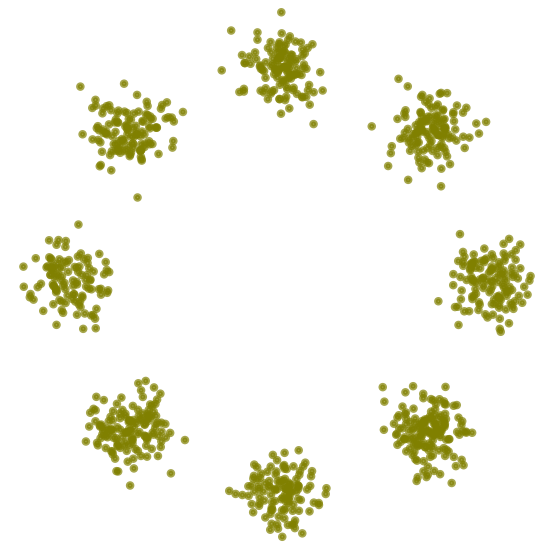
- Draw new samples $\underbrace{x_1^{\text{new}}, \ldots, x_n^{\text{new}}}_{\text{Goal}} \sim \underbrace{p_{\text{data}}}_{\text{unknown}}$

What is Generative Modeling? A 2D-Example

Samples $x_1, \ldots, x_n \sim p_{\text{data}}$



Goal: Generate Samples $x_1^{\text{new}}, \ldots, x_n^{\text{new}} \sim p_{\text{data}}$



Problem Setting:

- Access to n samples $x_1, \ldots, x_n \in \mathbb{R}^2$
- $\underbrace{x_1, \ldots, x_n}_{\text{known}} \sim \underbrace{p_{\text{data}}}_{\text{unknown}}$

Goal:

- Draw new samples $\underbrace{x_1^{\text{new}}, \ldots, x_n^{\text{new}}}_{\text{Goal}} \sim \underbrace{p_{\text{data}}}_{\text{unknown}}$

What is Generative Modeling? A 2D-Example

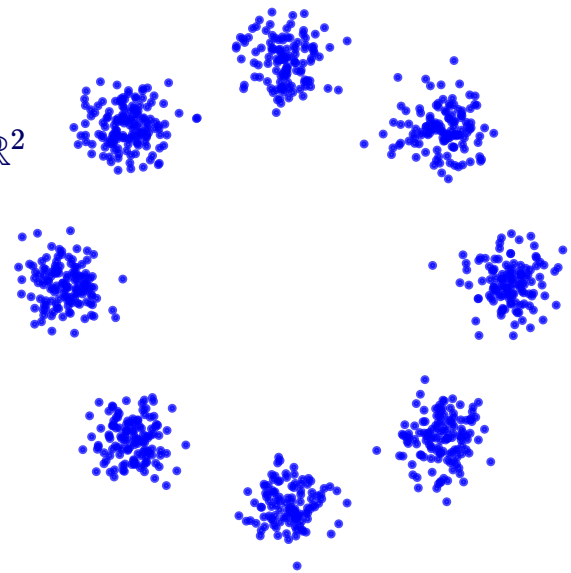
Problem Setting:

- Access to n samples $x_1, \ldots, x_n \in \mathbb{R}^2$
- $\underbrace{x_1, \ldots, x_n}_{\text{known}} \sim \underbrace{p_{\text{data}}}_{\text{unknown}}$

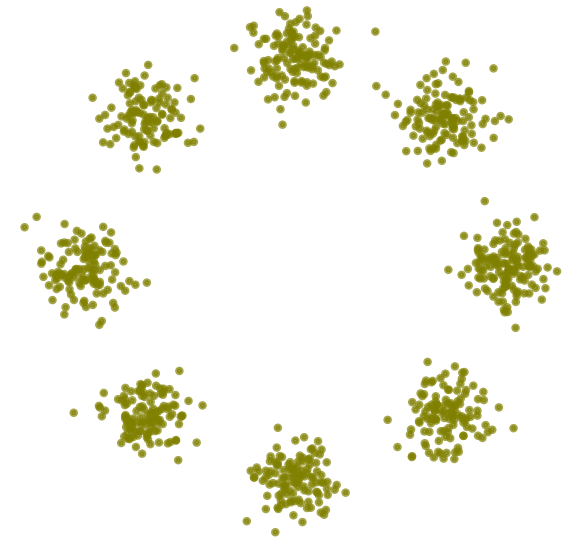
Goal:

- Draw new samples $\underbrace{x_1^{\text{new}}, \ldots, x_n^{\text{new}}}_{\text{Goal}} \sim \underbrace{p_{\text{data}}}_{\text{unknown}}$

Samples $x_1, \ldots, x_n \sim p_{\text{data}}$

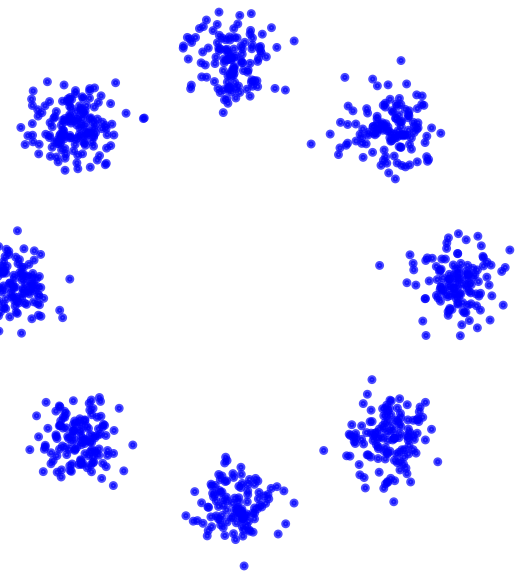


Goal: Generate Samples $x_1^{\text{new}}, \ldots, x_n^{\text{new}} \sim p_{\text{data}}$

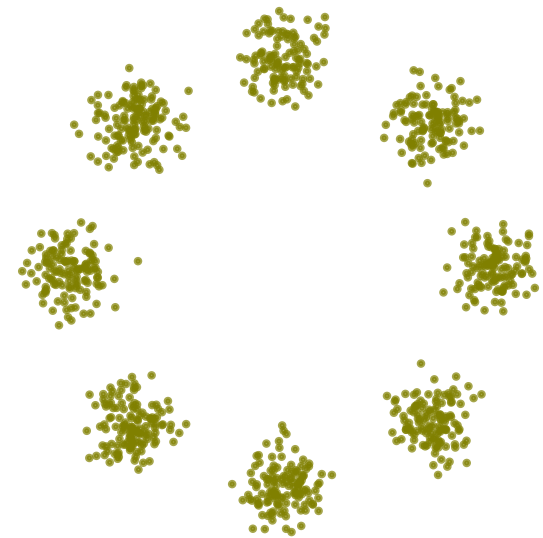


What is Generative Modeling? A 2D-Example

Samples $x_1, \ldots, x_n \sim p_{\text{data}}$



Goal: Generate Samples $x_1^{\text{new}}, \ldots, x_n^{\text{new}} \sim p_{\text{data}}$



Problem Setting:

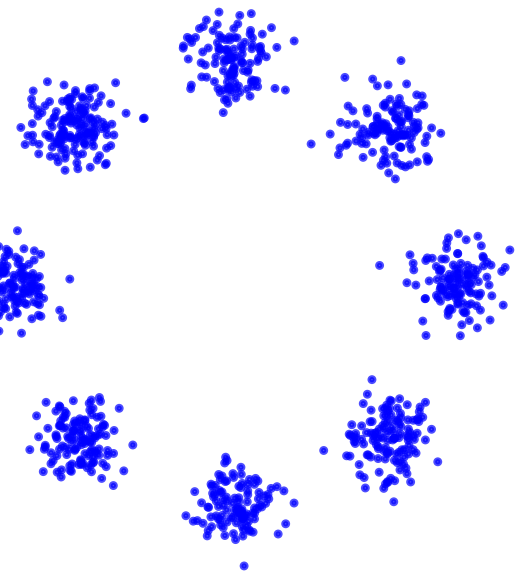
- Access to n samples $x_1, \ldots, x_n \in \mathbb{R}^2$
- $\underbrace{x_1, \ldots, x_n}_{\text{known}} \sim \underbrace{p_{\text{data}}}_{\text{unknown}}$

Goal:

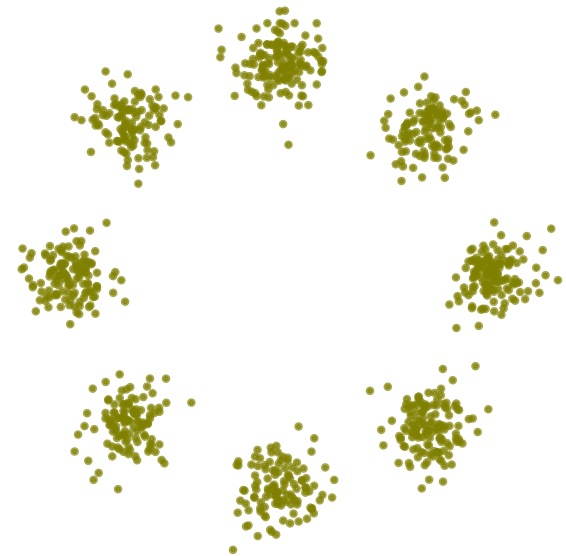
- Draw new samples $\underbrace{x_1^{\text{new}}, \ldots, x_n^{\text{new}}}_{\text{Goal}} \sim \underbrace{p_{\text{data}}}_{\text{unknown}}$

What is Generative Modeling? A 2D-Example

Samples $x_1, \ldots, x_n \sim p_{\text{data}}$



Goal: Generate Samples $x_1^{\text{new}}, \ldots, x_n^{\text{new}} \sim p_{\text{data}}$



Problem Setting:

- Access to n samples $x_1, \ldots, x_n \in \mathbb{R}^2$
- $\underbrace{x_1, \ldots, x_n}_{\text{known}} \sim \underbrace{p_{\text{data}}}_{\text{unknown}}$

Goal:

- Draw new samples $\underbrace{x_1^{\text{new}}, \ldots, x_n^{\text{new}}}_{\text{Goal}} \sim \underbrace{p_{\text{data}}}_{\text{unknown}}$

What is Generative Modeling? A 2D-Example

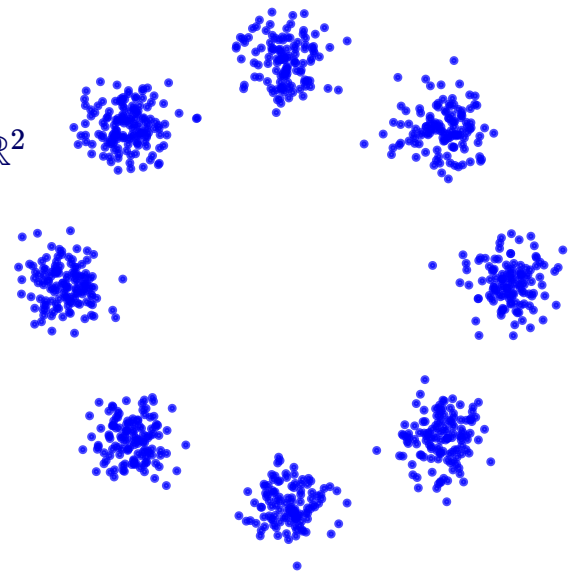
Problem Setting:

- Access to n samples $x_1, \ldots, x_n \in \mathbb{R}^2$
- $\underbrace{x_1, \ldots, x_n}_{\text{known}} \sim \underbrace{p_{\text{data}}}_{\text{unknown}}$

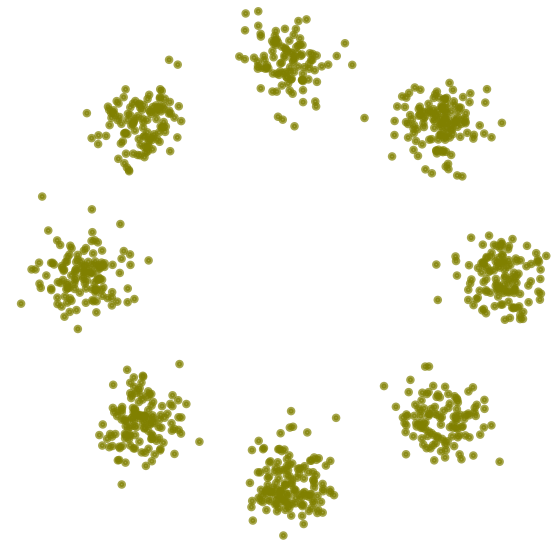
Goal:

- Draw new samples $\underbrace{x_1^{\text{new}}, \ldots, x_n^{\text{new}}}_{\text{Goal}} \sim \underbrace{p_{\text{data}}}_{\text{unknown}}$

Samples $x_1, \ldots, x_n \sim p_{\text{data}}$



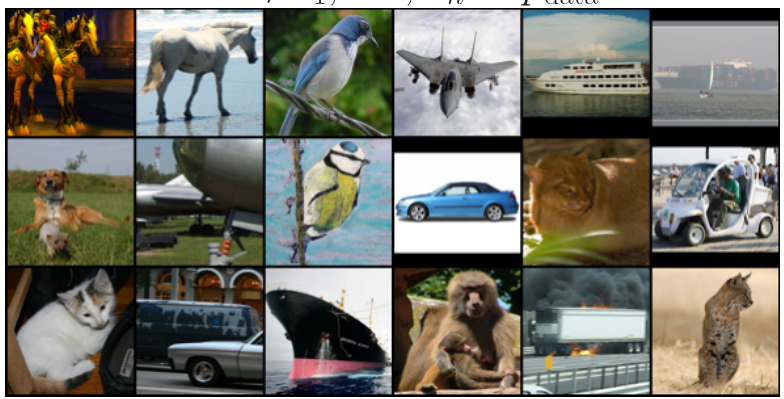
Goal: Generate Samples $x_1^{\text{new}}, \ldots, x_n^{\text{new}} \sim p_{\text{data}}$



What is Generative Modeling? An Image Example

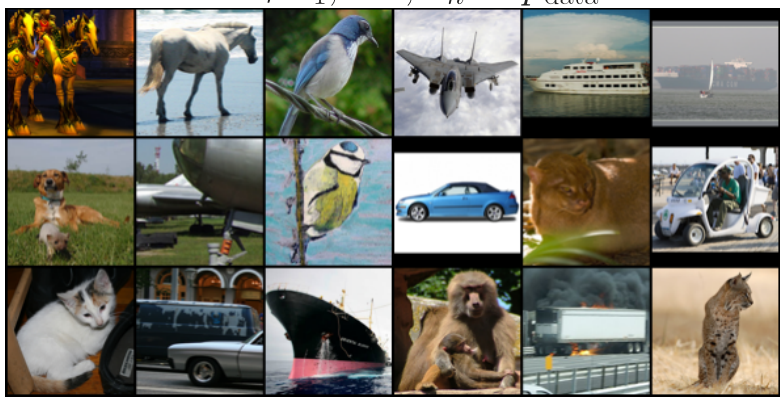
What is Generative Modeling? An Image Example

Data, $x_1, \ldots, x_n \sim p_{\text{data}}$



What is Generative Modeling? An Image Example

Data, $x_1, \ldots, x_n \sim p_{\text{data}}$

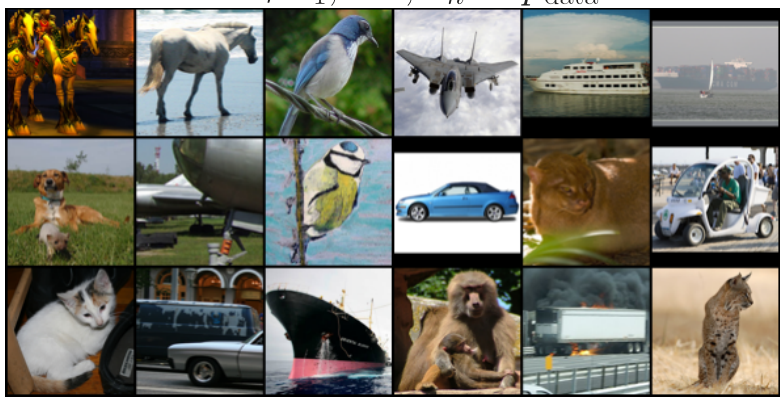


Data:

- Access to samples $x_1, \ldots, x_n$
- Drawn from p_{data} , $\underbrace{x_1, \ldots, x_n}_{\text{known}} \sim \underbrace{p_{\text{data}}}_{\text{unknown}}$
- e.g., set of images, text, proteins, etc

What is Generative Modeling? An Image Example

Data, $x_1, \ldots, x_n \sim p_{\text{data}}$



Data:

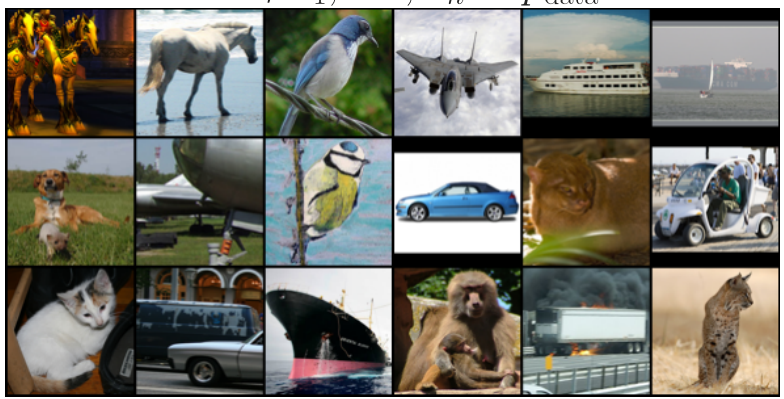
- Access to samples $x_1, \ldots, x_n$
- Drawn from p_{data} , $\underbrace{x_1, \ldots, x_n}_{\text{known}} \sim \underbrace{p_{\text{data}}}_{\text{unknown}}$
- e.g., set of images, text, proteins, etc

Goal:

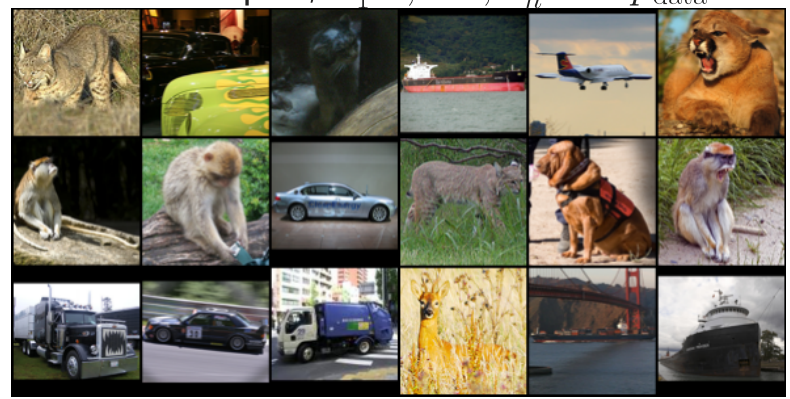
- Create new samples $\underbrace{x_1^{\text{new}}, \ldots, x_n^{\text{new}}}_{\text{Goal}} \sim \underbrace{p_{\text{data}}}_{\text{unknown}}$

What is Generative Modeling? An Image Example

Data, $x_1, \ldots, x_n \sim p_{\text{data}}$



New Samples, $x_1^{\text{new}}, \ldots, x_n^{\text{new}} \sim p_{\text{data}}$



Data:

- Access to samples $x_1, \ldots, x_n$
- Drawn from p_{data}, $\underbrace{x_1, \ldots, x_n}_{\text{known}} \sim \underbrace{p_{\text{data}}}_{\text{unknown}}$
- e.g., set of images, text, proteins, etc

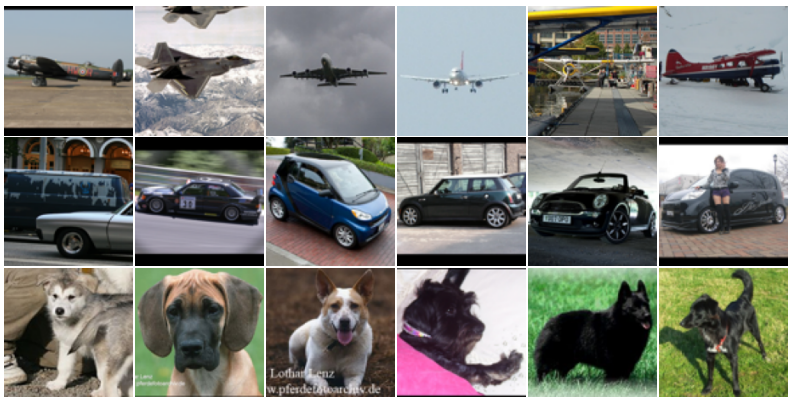
Goal:

- Create new samples $\underbrace{x_1^{\text{new}}, \ldots, x_n^{\text{new}}}_{\text{Goal}} \sim \underbrace{p_{\text{data}}}_{\text{unknown}}$

Class-Conditional Generative Modeling

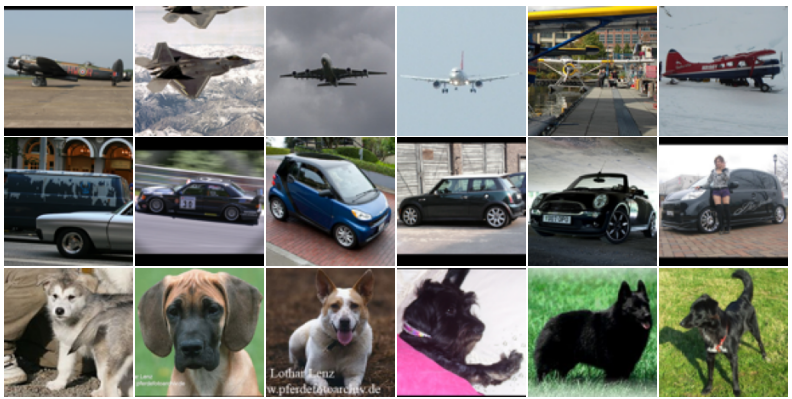
Class-Conditional Generative Modeling

Data: $(x_i, y_i), x_i \sim p_{\text{data}}(\cdot|y_i)$



Class-Conditional Generative Modeling

Data: $(x_i, y_i), x_i \sim p_{\text{data}}(\cdot|y_i)$

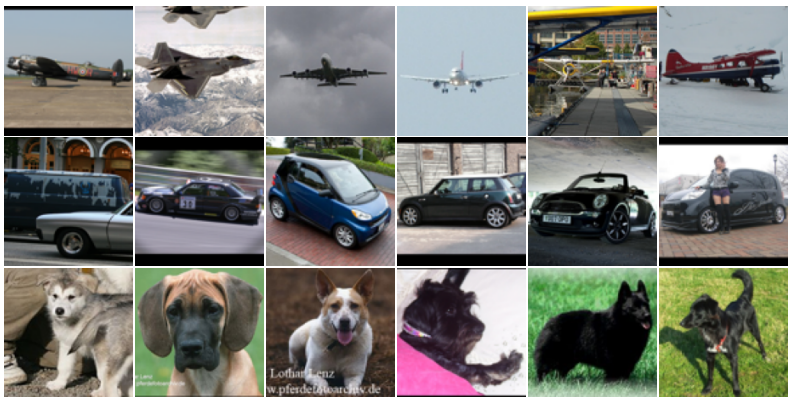


Data:

- Access to samples $(\underbrace{x_1}_{\text{images}}, \underbrace{y_1}_{\text{label}}), \ldots, (x_n, y_n)$
- Drawn from p_{data}, $\underbrace{x_i}_{\text{known}} \sim \underbrace{p_{\text{data}}(\cdot|y_i)}_{\text{unknown}}$
- e.g., set of labelled images

Class-Conditional Generative Modeling

Data: $(x_i, y_i), x_i \sim p_{\text{data}}(\cdot|y_i)$



Data:

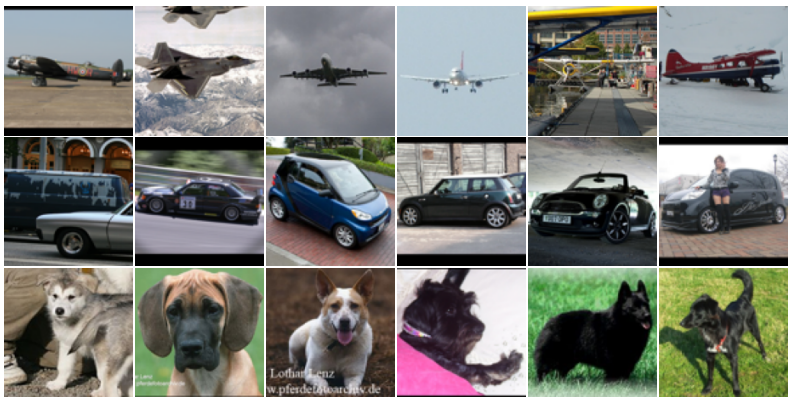
- Access to samples $(\underbrace{x_1}_{\text{images}}, \underbrace{y_1}_{\text{label}}), \ldots, (x_n, y_n)$
- Drawn from p_{data}, $\underbrace{x_i}_{\text{known}} \sim \underbrace{p_{\text{data}}(\cdot|y_i)}_{\text{unknown}}$
- e.g., set of labelled images

Goal:

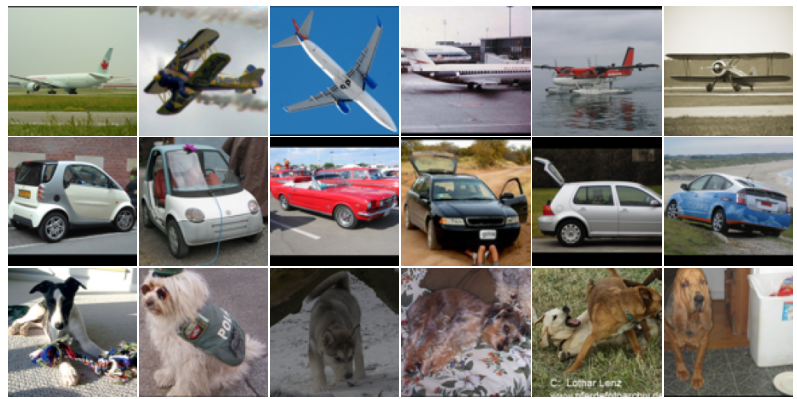
- Create new samples $\underbrace{x_1^{\text{new}}, \ldots, x_n^{\text{new}}}_{\text{Goal}} \sim \underbrace{p_{\text{data}}(\cdot|y_i)}_{\text{unknown}}$

Class-Conditional Generative Modeling

Data: $(x_i, y_i), x_i \sim p_{\text{data}}(\cdot|y_i)$



Goal: $x_i^{\text{new}} \sim p_{\text{data}}(\cdot|y_i)$



Data:

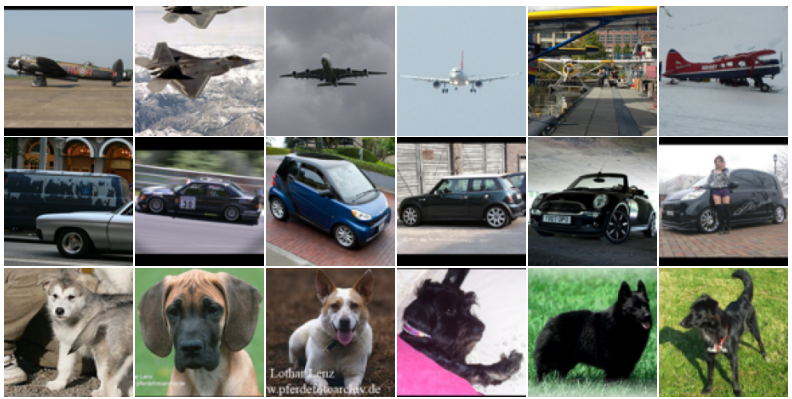
- Access to samples $(\underbrace{x_1}_{\text{images}}, \underbrace{y_1}_{\text{label}}), \ldots, (x_n, y_n)$
- Drawn from p_{data}, $\underbrace{x_i}_{\text{known}} \sim \underbrace{p_{\text{data}}(\cdot|y_i)}_{\text{unknown}}$
- e.g., set of labelled images

Goal:

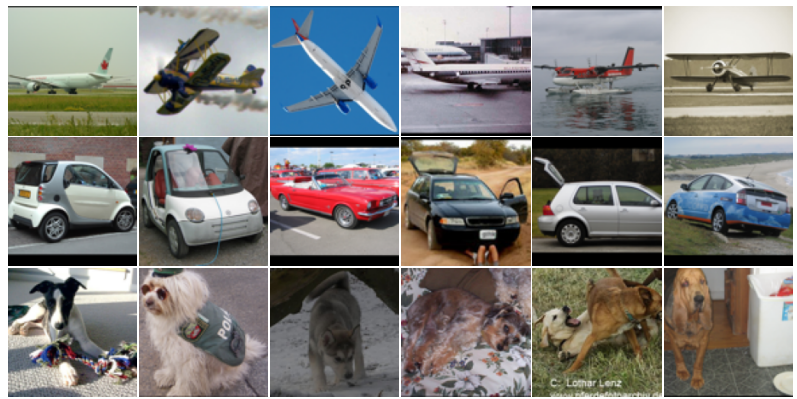
- Create new samples $\underbrace{x_1^{\text{new}}, \ldots, x_n^{\text{new}}}_{\text{Goal}} \sim \underbrace{p_{\text{data}}(\cdot|y_i)}_{\text{unknown}}$

Class-Conditional Generative Modeling

Data: $(x_i, y_i), x_i \sim p_{\text{data}}(\cdot|y_i)$



Goal: $x_i^{\text{new}} \sim p_{\text{data}}(\cdot|y_i)$



Data:

- Access to samples $(\underbrace{x_1}_{\text{images}}, \underbrace{y_1}_{\text{label}}), \ldots, (x_n, y_n)$
- Drawn from p_{data}, $\underbrace{x_i}_{\text{known}} \sim \underbrace{p_{\text{data}}(\cdot|y_i)}_{\text{unknown}}$
- e.g., set of labelled images

Goal:

- Create new samples $\underbrace{x_1^{\text{new}}, \ldots, x_n^{\text{new}}}_{\text{Goal}} \sim \underbrace{p_{\text{data}}(\cdot|y_i)}_{\text{unknown}}$

Comment:

- **Generative learning**: given a class y_i, generate an image $x_i^{new} \sim p_{\text{data}}(\cdot|y_i)$

Text-to-Image Generative Modeling

MALICE team logo

An avocado chair

A house made of sushi

Text-to-Image Generative Modeling

MALICE team logo

An avocado chair

A house made of sushi

Data:

- Access to samples $(\underbrace{x_1}_{\text{image}}, \underbrace{t_1}_{\text{text}}), \ldots, (x_n, t_n)$
- Drawn from an unknown distribution $\underbrace{(x_i, t_i)}_{\text{known}} \sim$

$\underbrace{p_{\text{data}}(x, t)}_{\text{unknown}}$

Text-to-Image Generative Modeling

MALICE team logo

An avocado chair

A house made of sushi

Data:

- Access to samples $(\underbrace{x_1}_{\text{image}}, \underbrace{t_1}_{\text{text}}), \ldots, (x_n, t_n)$
- Drawn from an unknown distribution $\underbrace{(x_i, t_i)}_{\text{known}} \sim \underbrace{p_{\text{data}}(x, t)}_{\text{unknown}}$

Goal:

- Generate images conditioned on **new** text
- Sample: $\underbrace{x^{\text{new}}}_{\text{image}} \sim p_\theta(x \mid \underbrace{t}_{\text{unseen at training}})$

Text-to-Image Generative Modeling

MALICE team logo

An avocado chair

A house made of sushi

Data:

- Access to samples $(\underbrace{x_1}_{\text{image}}, \underbrace{t_1}_{\text{text}}), \ldots, (x_n, t_n)$
- Drawn from an unknown distribution $\underbrace{(x_i, t_i)}_{\text{known}} \sim$

$$\underbrace{p_{\text{data}}(x, t)}_{\text{unknown}}$$

Goal:

- Generate images conditioned on **new** text
- Sample: $\underbrace{x^{\text{new}}}_{\text{image}} \sim p_\theta(x \mid \underbrace{t}_{\text{unseen at training}})$

Comment: text is harder

- Discrete
- Generalization to unseen prompts

A Few Comments

A Few Comments

Text-to-Text Models

- Conditional text generation w.r.t. the previous text sequence

A Few Comments

Text-to-Text Models

- Conditional text generation w.r.t. the previous text sequence

"From Noise to Structure"

A Few Comments

Text-to-Text Models

- Conditional text generation w.r.t. the previous text sequence

"From Noise to Structure"

- Just samples, no density
  - Too hard to sample from the set of natural images
  - Easy to sample from standard Gaussian noise
  - "Rebranding of sampling"

A Few Comments

Text-to-Text Models

- Conditional text generation w.r.t. the previous text sequence

"From Noise to Structure"

- Just samples, no density
  - Too hard to sample from the set of natural images
  - Easy to sample from standard Gaussian noise
  - "Rebranding of sampling"

Evaluation?

- Quality of the generated images?
  - No cross-validation
  - Very challenging for text-to-text models
  - Often use other models to assess quality (e.g. GPT)

Zoo of Generative Models [1]

[1] F. Caetano, GenerativeZoo: A Model Zoo for Generative Models, In: GitHub repository, 2024.
[2] D. Kingma et al., Auto-encoding variational bayes, In: ICLR, 2014.
[3] C. Doersch, Tutorial on variational autoencoders, In: arXiv preprint arXiv:1606.05908, 2016.
[4] I. Goodfellow et al., Generative adversarial nets, In: NeurIPS, 2014.
[5] J. Zhu et al., Unpaired image-to-image translation using cycle-consistent adversarial networks, In: ICCV, 2017.
[6] I. Gulrajani et al., Improved training of Wasserstein gans, In: NeurIPS, 2017.
[7] A. Van Den Oord et al., Pixel recurrent neural networks, In: International conference on machine learning, 2016.
[8] A. Van Den Oord et al., Wavenet: A generative model for raw audio, In: arXiv preprint arXiv:1609.03499, 2016.
[9] J. Sohl-Dickstein et al., Deep unsupervised learning using nonequilibrium thermodynamics, In: ICML, 2015.
[10] J. Ho et al., Denoising diffusion probabilistic models, In: Advances in neural information processing systems, 2020.
[11] R. Gao et al., Diffusion Meets Flow Matching: Two Sides of the Same Coin, In: , 2024.
[12] Y. Lipman et al., Flow matching for generative modeling, In: ICLR, 2023.
[13] X. Liu et al., Flow straight and fast: Learning to generate and transfer data with rectified flow, In: ICLR, 2023.
[14] A. Gagneux et al., A Visual Dive into Conditional Flow Matching, In: ICLR Blogposts, 2025.

Zoo of Generative Models [1]

- Variational Autoencoders (2014) [2] [3]

[1] F. Caetano, GenerativeZoo: A Model Zoo for Generative Models, In: GitHub repository, 2024.
[2] D. Kingma et al., Auto-encoding variational bayes, In: ICLR, 2014.
[3] C. Doersch, Tutorial on variational autoencoders, In: arXiv preprint arXiv:1606.05908, 2016.
[4] I. Goodfellow et al., Generative adversarial nets, In: NeurIPS, 2014.
[5] J. Zhu et al., Unpaired image-to-image translation using cycle-consistent adversarial networks, In: ICCV, 2017.
[6] I. Gulrajani et al., Improved training of Wasserstein gans, In: NeurIPS, 2017.
[7] A. Van Den Oord et al., Pixel recurrent neural networks, In: International conference on machine learning, 2016.
[8] A. Van Den Oord et al., Wavenet: A generative model for raw audio, In: arXiv preprint arXiv:1609.03499, 2016.
[9] J. Sohl-Dickstein et al., Deep unsupervised learning using nonequilibrium thermodynamics, In: ICML, 2015.
[10] J. Ho et al., Denoising diffusion probabilistic models, In: Advances in neural information processing systems, 2020.
[11] R. Gao et al., Diffusion Meets Flow Matching: Two Sides of the Same Coin, In: , 2024.
[12] Y. Lipman et al., Flow matching for generative modeling, In: ICLR, 2023.
[13] X. Liu et al., Flow straight and fast: Learning to generate and transfer data with rectified flow, In: ICLR, 2023.
[14] A. Gagneux et al., A Visual Dive into Conditional Flow Matching, In: ICLR Blogposts, 2025.

Zoo of Generative Models [1]

- Variational Autoencoders (2014) [2] [3]
- Generative Adversarial Networks (GANs) [4] (2014)
  - Cycle GANs [5]
  - Wasserstein GANs [6]

[1] F. Caetano, GenerativeZoo: A Model Zoo for Generative Models, In: GitHub repository, 2024.
[2] D. Kingma et al., Auto-encoding variational bayes, In: ICLR, 2014.
[3] C. Doersch, Tutorial on variational autoencoders, In: arXiv preprint arXiv:1606.05908, 2016.
[4] I. Goodfellow et al., Generative adversarial nets, In: NeurIPS, 2014.
[5] J. Zhu et al., Unpaired image-to-image translation using cycle-consistent adversarial networks, In: ICCV, 2017.
[6] I. Gulrajani et al., Improved training of Wasserstein gans, In: NeurIPS, 2017.
[7] A. Van Den Oord et al., Pixel recurrent neural networks, In: International conference on machine learning, 2016.
[8] A. Van Den Oord et al., Wavenet: A generative model for raw audio, In: arXiv preprint arXiv:1609.03499, 2016.
[9] J. Sohl-Dickstein et al., Deep unsupervised learning using nonequilibrium thermodynamics, In: ICML, 2015.
[10] J. Ho et al., Denoising diffusion probabilistic models, In: Advances in neural information processing systems, 2020.
[11] R. Gao et al., Diffusion Meets Flow Matching: Two Sides of the Same Coin, In: , 2024.
[12] Y. Lipman et al., Flow matching for generative modeling, In: ICLR, 2023.
[13] X. Liu et al., Flow straight and fast: Learning to generate and transfer data with rectified flow, In: ICLR, 2023.
[14] A. Gagneux et al., A Visual Dive into Conditional Flow Matching, In: ICLR Blogposts, 2025.

Zoo of Generative Models [1]

- Variational Autoencoders (2014) [2] [3]
- Generative Adversarial Networks (GANs) [4] (2014)
  - Cycle GANs [5]
  - Wasserstein GANs [6]
- Autoregressive Models (2016)
  - Pixel RNNs [7]
  - Wavenets [8]

[1] F. Caetano, GenerativeZoo: A Model Zoo for Generative Models, In: GitHub repository, 2024.
[2] D. Kingma et al., Auto-encoding variational bayes, In: ICLR, 2014.
[3] C. Doersch, Tutorial on variational autoencoders, In: arXiv preprint arXiv:1606.05908, 2016.
[4] I. Goodfellow et al., Generative adversarial nets, In: NeurIPS, 2014.
[5] J. Zhu et al., Unpaired image-to-image translation using cycle-consistent adversarial networks, In: ICCV, 2017.
[6] I. Gulrajani et al., Improved training of Wasserstein gans, In: NeurIPS, 2017.
[7] A. Van Den Oord et al., Pixel recurrent neural networks, In: International conference on machine learning, 2016.
[8] A. Van Den Oord et al., Wavenet: A generative model for raw audio, In: arXiv preprint arXiv:1609.03499, 2016.
[9] J. Sohl-Dickstein et al., Deep unsupervised learning using nonequilibrium thermodynamics, In: ICML, 2015.
[10] J. Ho et al., Denoising diffusion probabilistic models, In: Advances in neural information processing systems, 2020.
[11] R. Gao et al., Diffusion Meets Flow Matching: Two Sides of the Same Coin, In: , 2024.
[12] Y. Lipman et al., Flow matching for generative modeling, In: ICLR, 2023.
[13] X. Liu et al., Flow straight and fast: Learning to generate and transfer data with rectified flow, In: ICLR, 2023.
[14] A. Gagneux et al., A Visual Dive into Conditional Flow Matching, In: ICLR Blogposts, 2025.

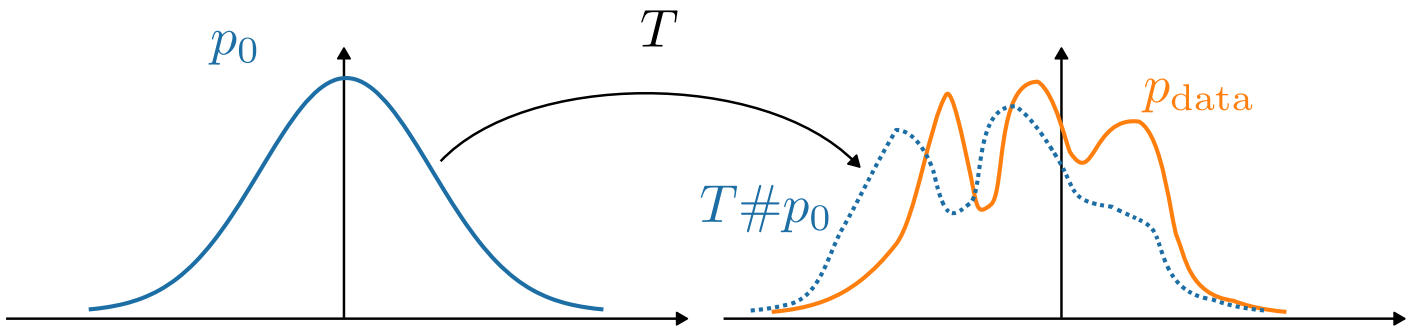
Zoo of Generative Models [1]

- Variational Autoencoders (2014) [2] [3]
- Generative Adversarial Networks (GANs) [4] (2014)
  - Cycle GANs [5]
  - Wasserstein GANs [6]
- Autoregressive Models (2016)
  - Pixel RNNs [7]
  - Wavenets [8]
- Diffusion (2020) [9] [10] [11] / Flow Matching Models (2023) [12] [13] [14]

[1] F. Caetano, GenerativeZoo: A Model Zoo for Generative Models, In: GitHub repository, 2024.
[2] D. Kingma et al., Auto-encoding variational bayes, In: ICLR, 2014.
[3] C. Doersch, Tutorial on variational autoencoders, In: arXiv preprint arXiv:1606.05908, 2016.
[4] I. Goodfellow et al., Generative adversarial nets, In: NeurIPS, 2014.
[5] J. Zhu et al., Unpaired image-to-image translation using cycle-consistent adversarial networks, In: ICCV, 2017.
[6] I. Gulrajani et al., Improved training of Wasserstein gans, In: NeurIPS, 2017.
[7] A. Van Den Oord et al., Pixel recurrent neural networks, In: International conference on machine learning, 2016.
[8] A. Van Den Oord et al., Wavenet: A generative model for raw audio, In: arXiv preprint arXiv:1609.03499, 2016.
[9] J. Sohl-Dickstein et al., Deep unsupervised learning using nonequilibrium thermodynamics, In: ICML, 2015.
[10] J. Ho et al., Denoising diffusion probabilistic models, In: Advances in neural information processing systems, 2020.
[11] R. Gao et al., Diffusion Meets Flow Matching: Two Sides of the Same Coin, In: , 2024.
[12] Y. Lipman et al., Flow matching for generative modeling, In: ICLR, 2023.
[13] X. Liu et al., Flow straight and fast: Learning to generate and transfer data with rectified flow, In: ICLR, 2023.
[14] A. Gagneux et al., A Visual Dive into Conditional Flow Matching, In: ICLR Blogposts, 2025.


Normalizing Flows
a.k.a., Iterative Refinement

Normalizing Flows



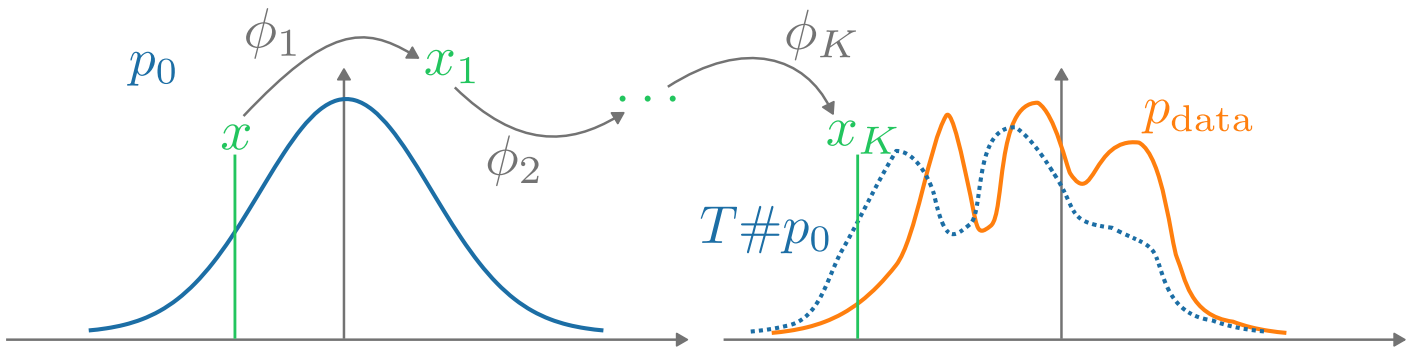
Definition (push-forward): if $x \sim p_0$ then $T(x) \sim T_{\#}p_0$

Normalizing flow (intuition):

- denoting $p_{gen} = T_{\#}p_0$
- locally, if T compresses the space by a factor 42, then $p_{gen}(T(x)) = 42 \cdot p_0(x)$
- formally, change of variable, $p_{gen}(T(x)) = |det(J_{T^{-1}}(x))| \cdot p_0(x)$ **(determinant of the jacobian of T^{-1})**

Principle: parametrize and learn T … so that its inverse exists (and has an easy jacobian det).

Normalizing Flows, with Composed Functions



Learn a deep T, i.e.,

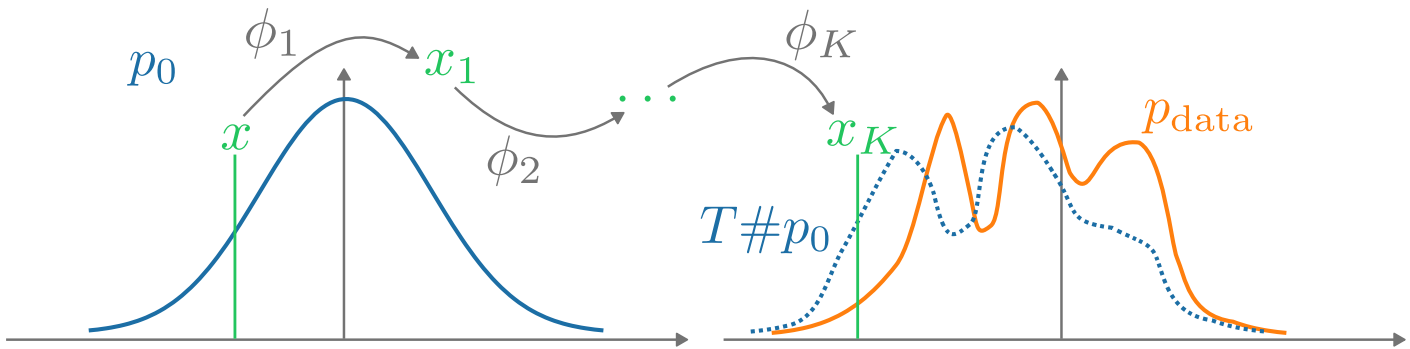
$$T = \phi_1 \circ \phi_2 \circ ... \circ \phi_K$$

Chain rule of change of variable,

$$|det(J_{T^{-1}}(x))| = \prod_k |det(J_{\phi_k^{-1}}(x))|$$

Principle: compose invertible blocks (with easy jacobian det)

Normalizing Flows, with Composed Functions



Learn a deep T , i.e.,

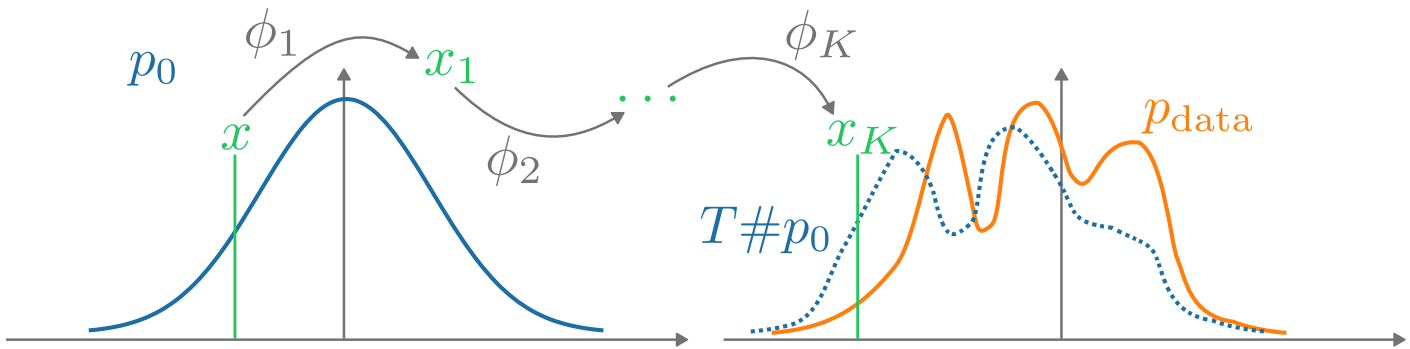
$$T = \phi_1 \circ \phi_2 \circ ... \circ \phi_K$$

Chain rule of change of variable,

$$|det(J_{T^{-1}}(x))| = \prod_k |det(J_{\phi_k^{-1}}(x))|$$

Principle: compose invertible blocks (with easy jacobian det)

Normalizing Flows, with Composed Functions



Learn a deep T, i.e.,

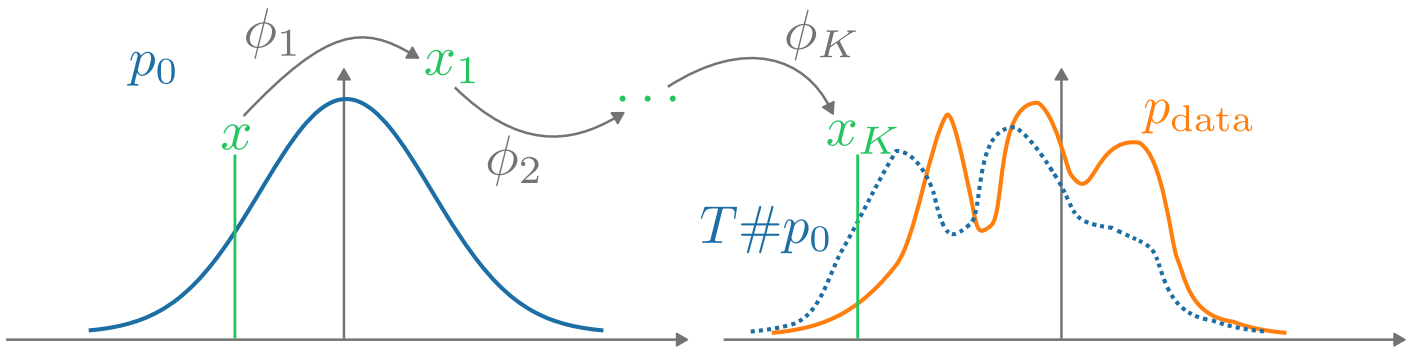
$$T = \phi_1 \circ \phi_2 \circ ... \circ \phi_K$$

Chain rule of change of variable,

$$|det(J_{T^{-1}}(x))| = \prod_k |det(J_{\phi_k^{-1}}(x))|$$

Principle: compose invertible blocks (with easy jacobian det)

Normalizing Flows, with Composed Functions



Learn a deep T, i.e.,

$$T = \phi_1 \circ \phi_2 \circ ... \circ \phi_K$$

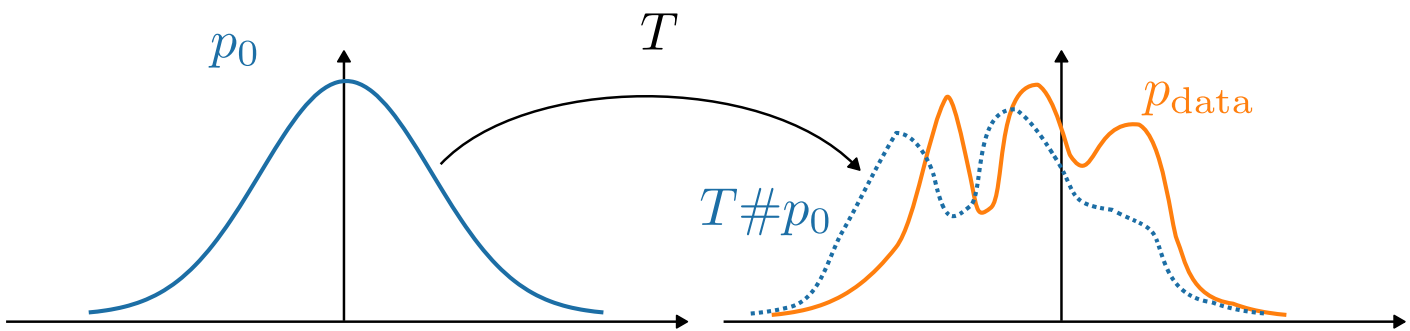
Chain rule of change of variable,

$$|det(J_{T^{-1}}(x))| = \prod_k |det(J_{\phi_k^{-1}}(x))|$$

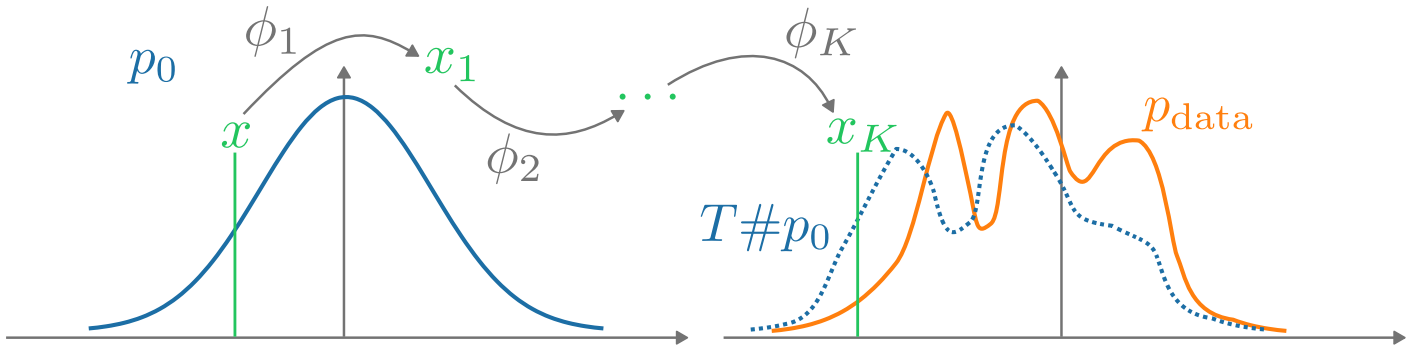
Principle: compose invertible blocks (with easy jacobian det)

Continuous Normalizing Flows

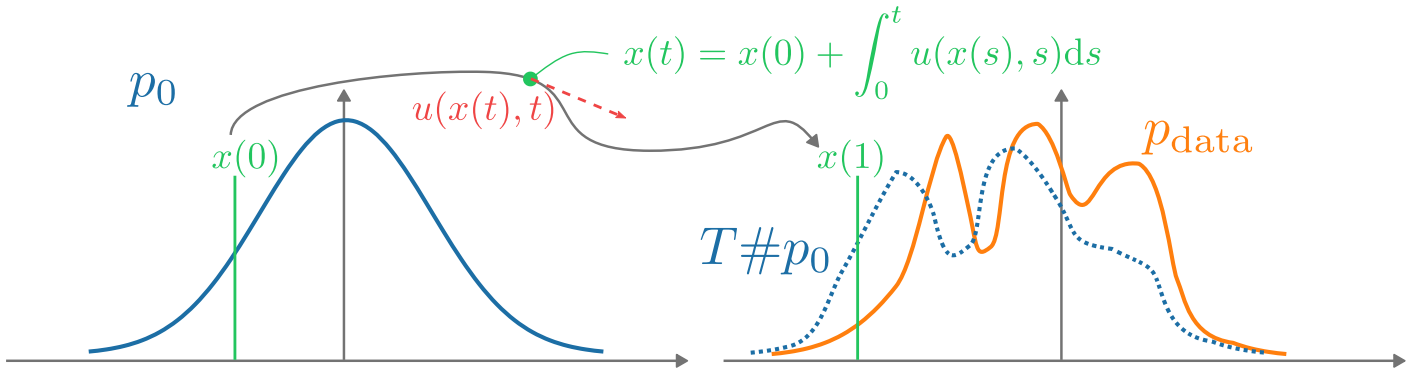
Continuous Normalizing Flows



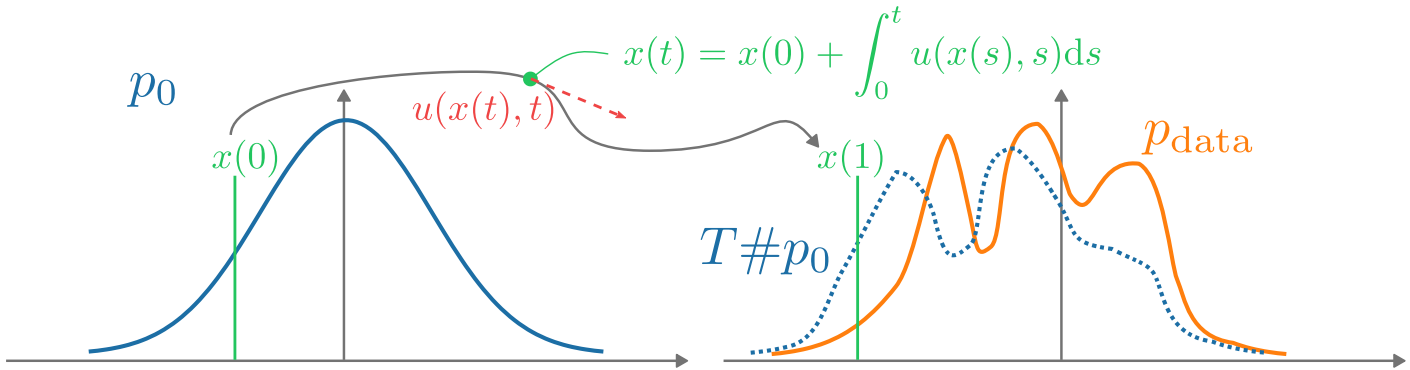
Continuous Normalizing Flows



Continuous Normalizing Flows



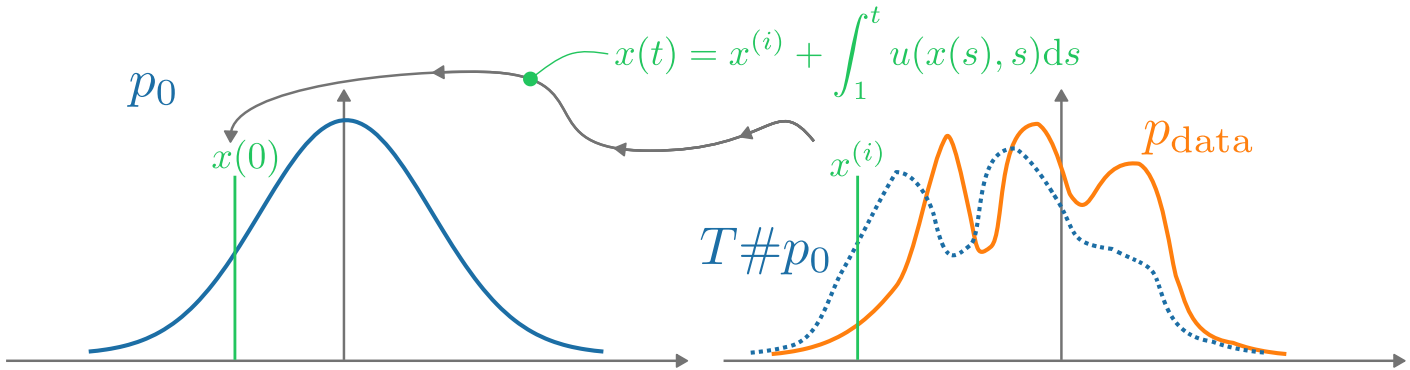
Continuous Normalizing Flows



Pushing to the limit

- Infinitely many infinitely-small steps
- Making depth continuous $k \mapsto t$
- Replacing $\phi_k(x)$ by $u(x, t)$, or $u_t(x)$

Continuous Normalizing Flows



Pushing to the limit

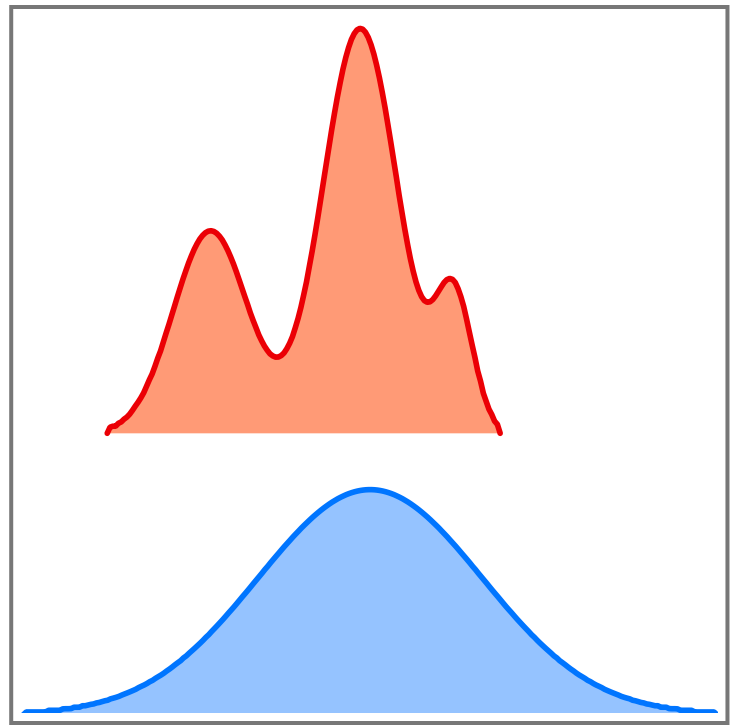
- Infinitely many infinitely-small steps
- Making depth continuous $k \mapsto t$
- Replacing $\phi_k(x)$ by $u(x,t)$, or $u_t(x)$

Continuous Normalizing Flow

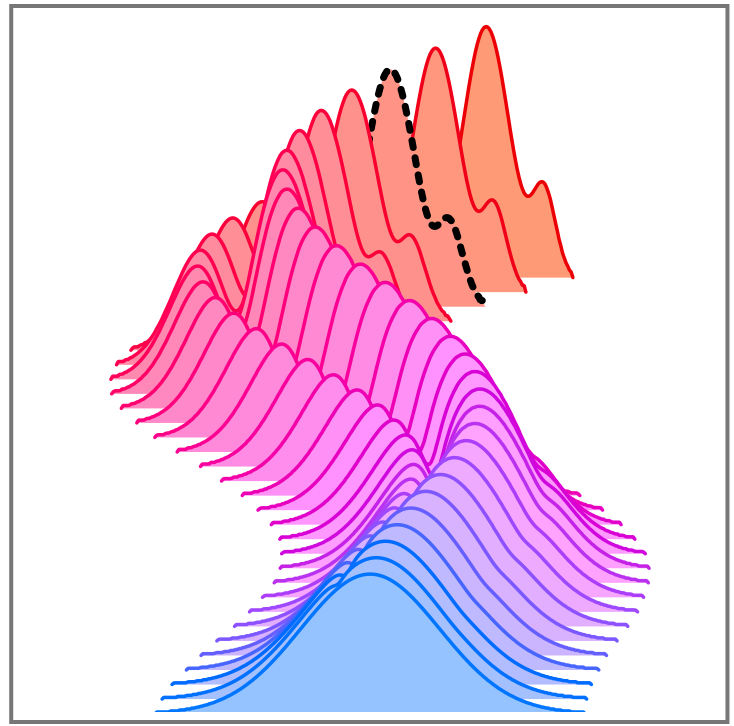
- easier: less constraints on u than ϕ

$$\min_\theta \mathrm{Dist}(T_\theta \# p_0, p_{\text{data}})$$

Continuous Normalizing Flows: Visual Summary



Continuous Normalizing Flows: "limitation"

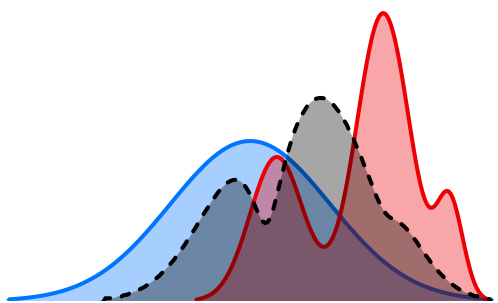


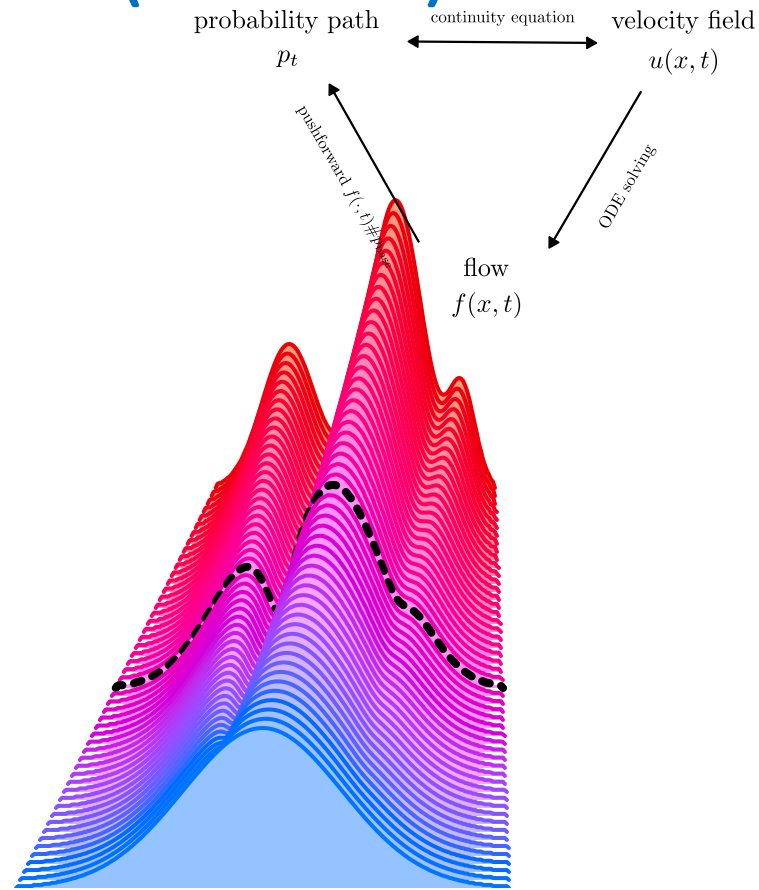
The flow is unspecified!
(there is an infinity of equally good solutions)

Probability paths, velocity fields (and flows)

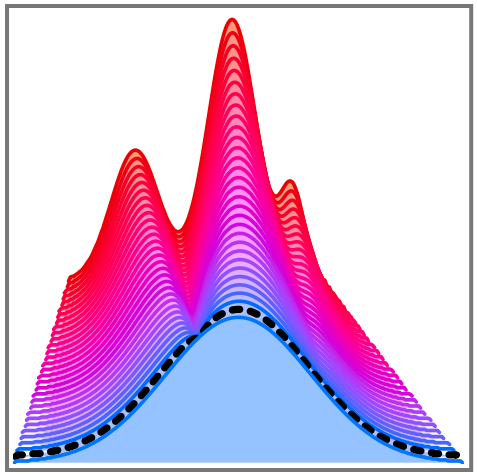
Continuity Equation / Transport Equation

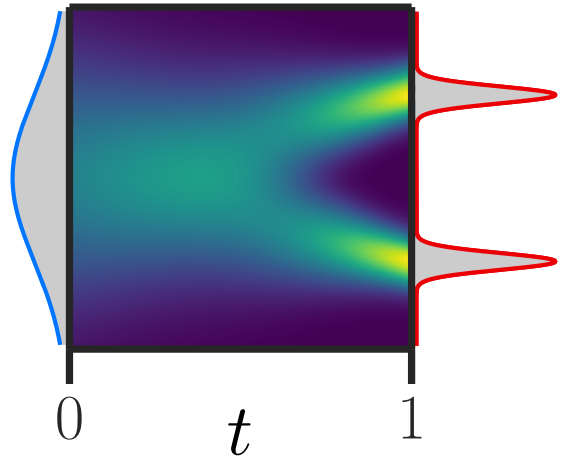
$$\partial_t p_t + \nabla \cdot u_t p_t = 0$$

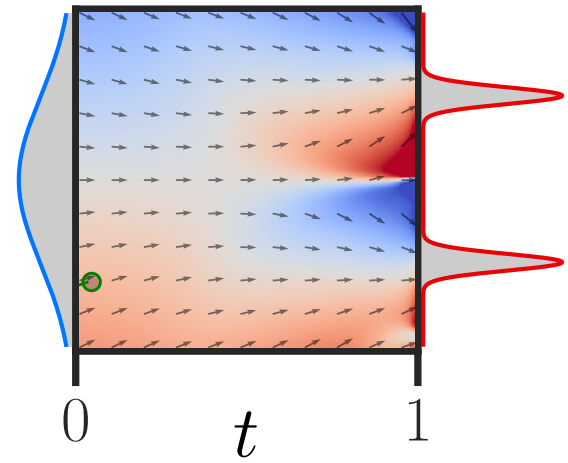





Conditional Flow Matching (CFM)







probability path
p_t

continuity equation

velocity field
$u(x,t)$

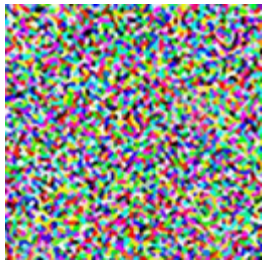
pushforward $f(\cdot,t)\#p_{\text{base}}$

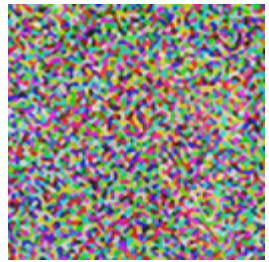
ODE solving

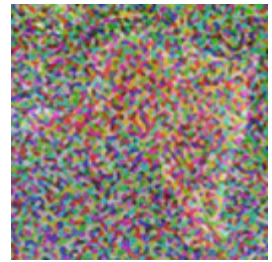
flow
$f(x,t)$

How to Learn a Good Velocity u_θ ?

How to Learn a Good Velocity u_θ ?


t=0

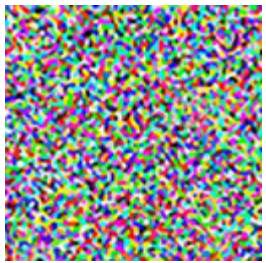

t=0.25

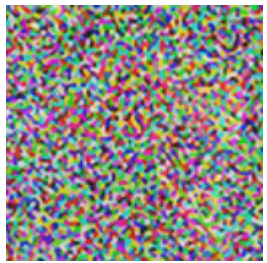

t=0.5

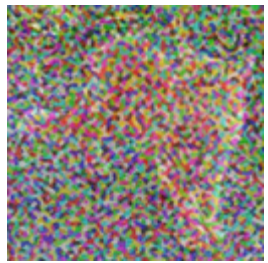

t=0.75


t=1

How to Learn a Good Velocity u_θ ?


t=0


t=0.25


t=0.5

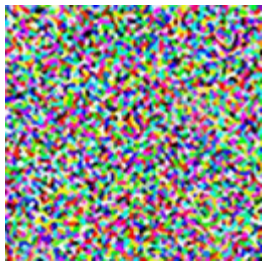

t=0.75

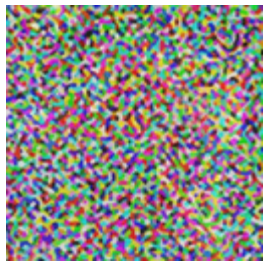

t=1

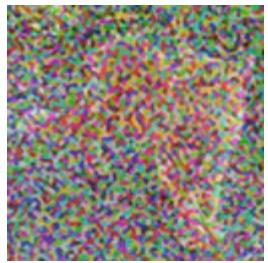
Idea : learn u_θ

$$u_\theta(tx_1 + (1-t)x_0, t) \approx x_1 - x_0$$

How to Learn a Good Velocity u_θ ?


t=0


t=0.25


t=0.5


t=0.75


t=1

Idea : learn u_θ

$$u_\theta(tx_1 + (1-t)x_0, t) \approx x_1 - x_0$$

Diffusion/Conditional Flow Matching [15,16,17]

$$\mathbb{E}_{\substack{x_0 \sim p_0 \\ x_1 \sim p_{\text{data}} \\ t \sim \mathcal{U}([0,1])}} ||u_\theta(tx_1 + (1-t)x_0, t) - (x_1 - x_0)||^2$$

[15] Y. Lipman et al., **Flow Matching for Generative Modeling,** In: ICLR, 2023.
[16] M. Albergo et al., **Building Normalizing Flows with Stochastic Interpolants,** In: ICLR, 2023.
[17] X. Liu et al., **Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow,** In: ICLR, 2023.

Conditional Flow Matching Algorithm

Conditional Flow Matching Algorithm

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

SGD step on θ with loss: $||u_\theta(x, t) - (x_1 - x_0)||_2^2$

Conditional Flow Matching Algorithm

$$
\begin{aligned}
x_0 &\sim \mathcal{N}(0, I) \\
x_1 &\sim p_{\text{data}} \\
t &\sim \text{Uniform}([0, 1]) \\
x_t &= t \cdot x_1 + (1 - t) \cdot x_0 \\
\text{SGD step on } \theta \text{ with loss: } &||u_\theta(x, t) - (x_1 - x_0)||_2^2
\end{aligned}
$$

That's it!

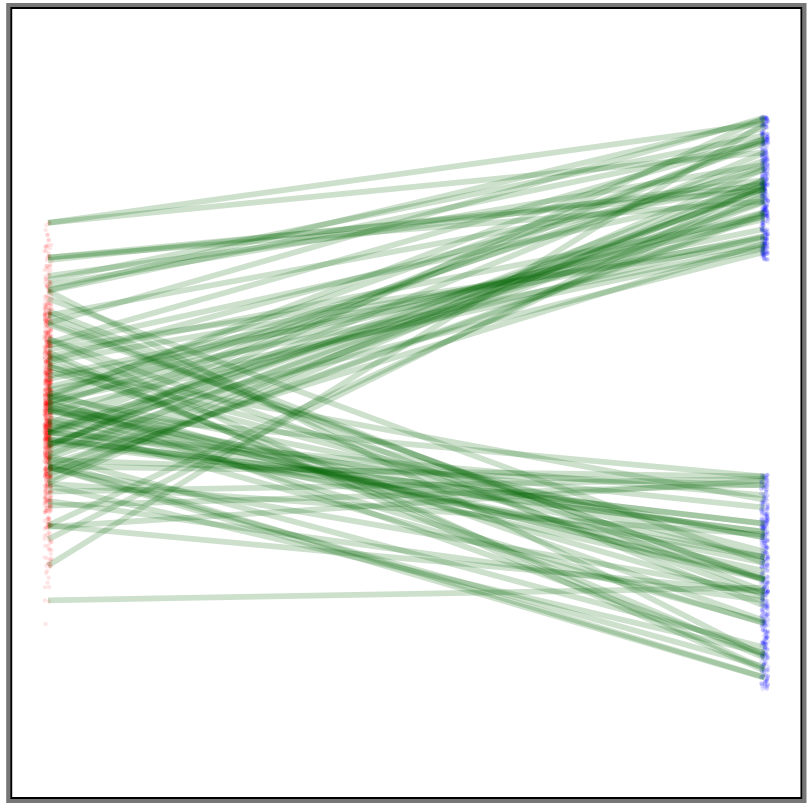
Conditional Flow Matching Algorithm

$$
\begin{aligned}
x_0 &\sim \mathcal{N}(0, I) \\
x_1 &\sim p_{\text{data}} \\
t &\sim \text{Uniform}([0, 1]) \\
x_t &= t \cdot x_1 + (1 - t) \cdot x_0 \\
\text{SGD step on } \theta \text{ with loss: } &||u_\theta(x, t) - (x_1 - x_0)||_2^2
\end{aligned}
$$

That's it!

(up to practical hacks and a few days of training)

CFM: Does it works? the "inversion", path un-mixing



CFM: Design choices

CFM: Design choices

Decide on p_0 , typically $\mathcal{N}(0, I)$

CFM: Design choices

Decide on p_0 , typically $\mathcal{N}(0, I)$

Decide on the conditioning variable (and its distribution), e.g.

CFM: Design choices

Decide on p_0 , typically $\mathcal{N}(0, I)$

Decide on the conditioning variable (and its distribution), e.g.

- z is a pair (x_0, x_1)
- z is a target point x_1
- z is a minibatch of source and target
- z is a pair, constrained by some clusters

CFM: Design choices

Decide on p_0 , typically $\mathcal{N}(0, I)$

Decide on the conditioning variable (and its distribution), e.g.

- z is a pair (x_0, x_1)
- z is a target point x_1
- z is a minibatch of source and target
- z is a pair, constrained by some clusters

Decide on the conditional "flow"

CFM: Design choices

Decide on p_0 , typically $\mathcal{N}(0, I)$

Decide on the conditioning variable (and its distribution), e.g.

- z is a pair (x_0, x_1)
- z is a target point x_1
- z is a minibatch of source and target
- z is a pair, constrained by some clusters

Decide on the conditional "flow"

- conditional probability path $p_t(x|z)$ (or $p(x, t|z)$)
- and associated velocity field $u^{\text{cond}}(x, t)$

CFM: Design choices

Decide on p_0 , typically $\mathcal{N}(0, I)$

Decide on the conditioning variable (and its distribution), e.g.

- z is a pair (x_0, x_1)
- z is a target point x_1
- z is a minibatch of source and target
- z is a pair, constrained by some clusters

Decide on the conditional "flow"

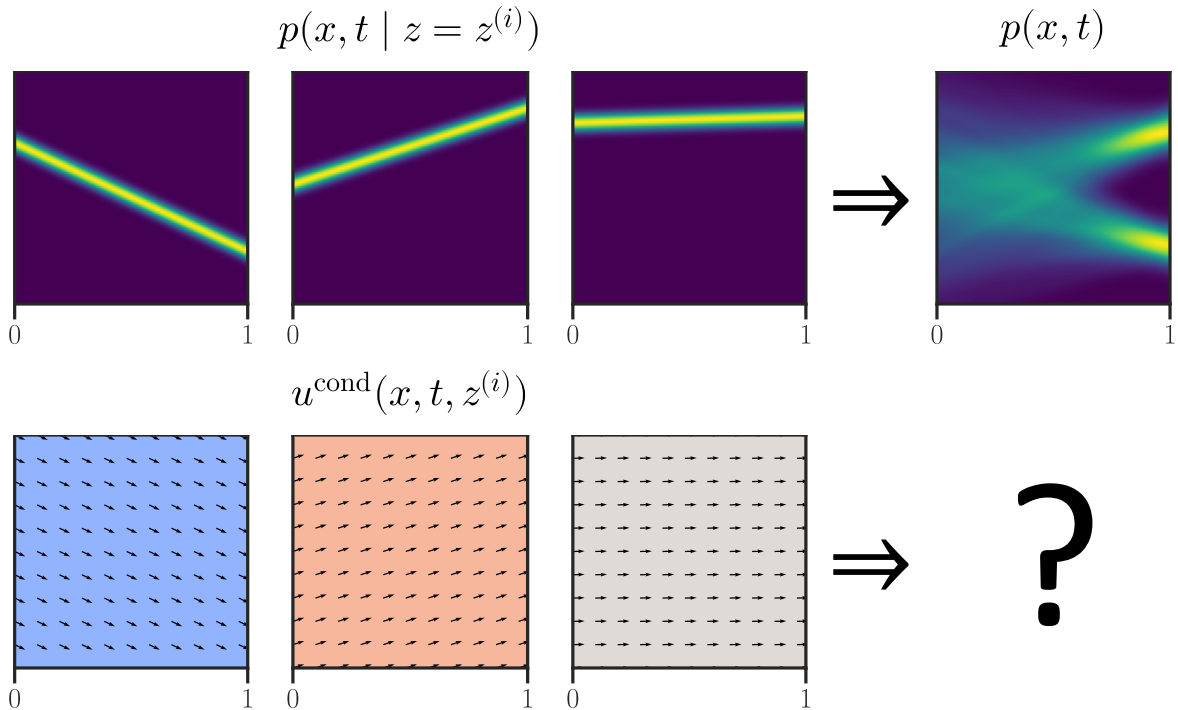
- conditional probability path $p_t(x|z)$ (or $p(x,t|z)$)
- and associated velocity field $u^{\text{cond}}(x,t)$

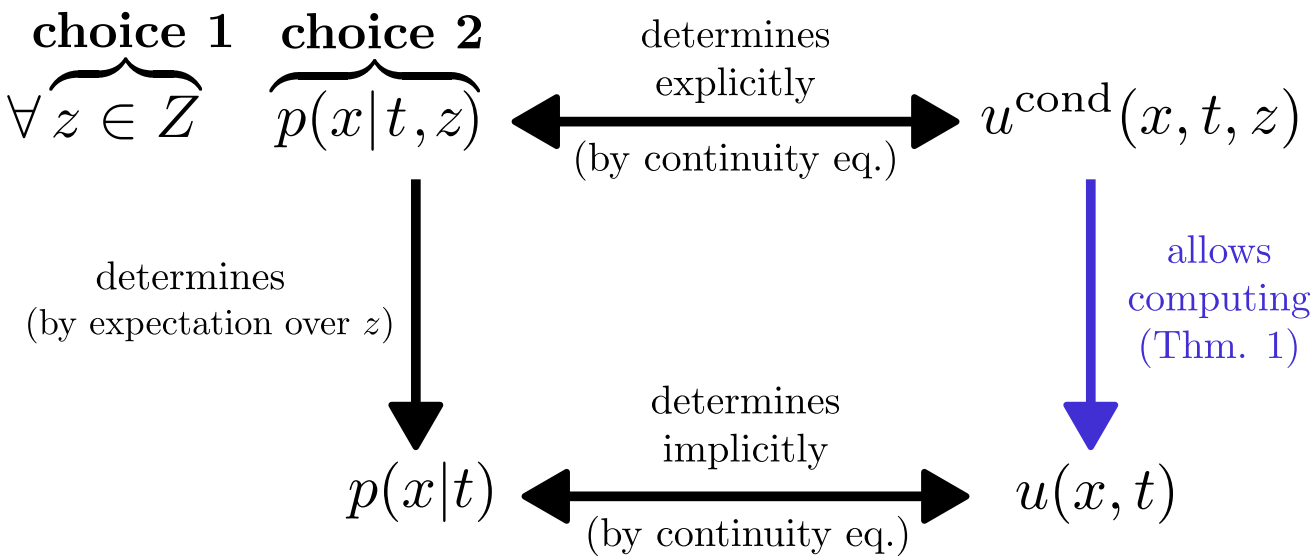
(under marginal constraints, on $p(x,t)$)

CFM: $p(x,t|z)$ (conditional) to $p(x,t)$ is easy

$$p(x,t) = \int p(x,t|z)p(z)dz = E_z\left[p(x,t|z)\right]$$

CFM: $u^{cond}(x, t, z)$ to $u(x, t)$ is less easy




choice 1
choice 2
$\forall z \in Z$
$p(x|t,z)$
determines
explicitly
(by continuity eq.)
$u^{\text{cond}}(x,t,z)$
determines
(by expectation over z)
allows
computing
(Thm. 1)
determines
implicitly
(by continuity eq.)
$p(x|t)$
$u(x,t)$

CFM: Closed form expression (Theorem 1)

$$\forall t, \forall x,$$

$$u(x,t) = E_{z|x,t}[u^{\text{cond}}(x,t,z)]$$

(also written as)

$$\forall\, t, \forall\, x,$$

$$u(x,t) = \int_z u^{\text{cond}}(x,t,z) p(z|x,t)$$

(or Bayes)

$$\forall\, t, \forall\, x,$$

$$u(x,t) = \int_z u^{\text{cond}}(x,t,z) \frac{p(x,t|z)p(z)}{p(x,t)} = E_z\left[\frac{u^{\text{cond}}(x,t,z)p(x,t|z)}{p(x,t)}\right] = E_z\left[\frac{u^{\text{cond}}(x,t,z)p(x,t|z)}{\sum_{z'} p(x,t|z')p(z')}\right]$$

CFM: Closed form expression (Theorem 1)

$$\forall t, \forall x,$$

$$u(x,t) = E_{z|x,t}[u^{\text{cond}}(x,t,z)]$$

(also written as)

$$\forall\, t, \forall\, x,$$

$$u(x,t) = \int_z u^{\text{cond}}(x,t,z) p(z|x,t)$$

(or Bayes)

$$\forall\, t, \forall\, x,$$

$$u(x,t) = \int_z u^{\text{cond}}(x,t,z) \frac{p(x,t|z)p(z)}{p(x,t)} = E_z\left[\frac{u^{\text{cond}}(x,t,z)p(x,t|z)}{p(x,t)}\right] = E_z\left[\frac{u^{\text{cond}}(x,t,z)p(x,t|z)}{\sum_{z'} p(x,t|z')p(z')}\right]$$

CFM: Closed form expression (Theorem 1)

$$\forall t, \forall x,$$

$$u(x,t) = E_{z|x,t}[u^{\text{cond}}(x,t,z)]$$

(also written as)

$$\forall\, t, \forall\, x,$$

$$u(x,t) = \int_z u^{\text{cond}}(x,t,z) p(z|x,t)$$

(or Bayes)

$$\forall\, t, \forall\, x,$$

$$u(x,t) = \int_z u^{\text{cond}}(x,t,z) \frac{p(x,t|z)p(z)}{p(x,t)} = E_z\left[\frac{u^{\text{cond}}(x,t,z)p(x,t|z)}{p(x,t)}\right] = E_z\left[\frac{u^{\text{cond}}(x,t,z)p(x,t|z)}{\sum_{z'} p(x,t|z')p(z')}\right]$$

CFM playground

R

p_0

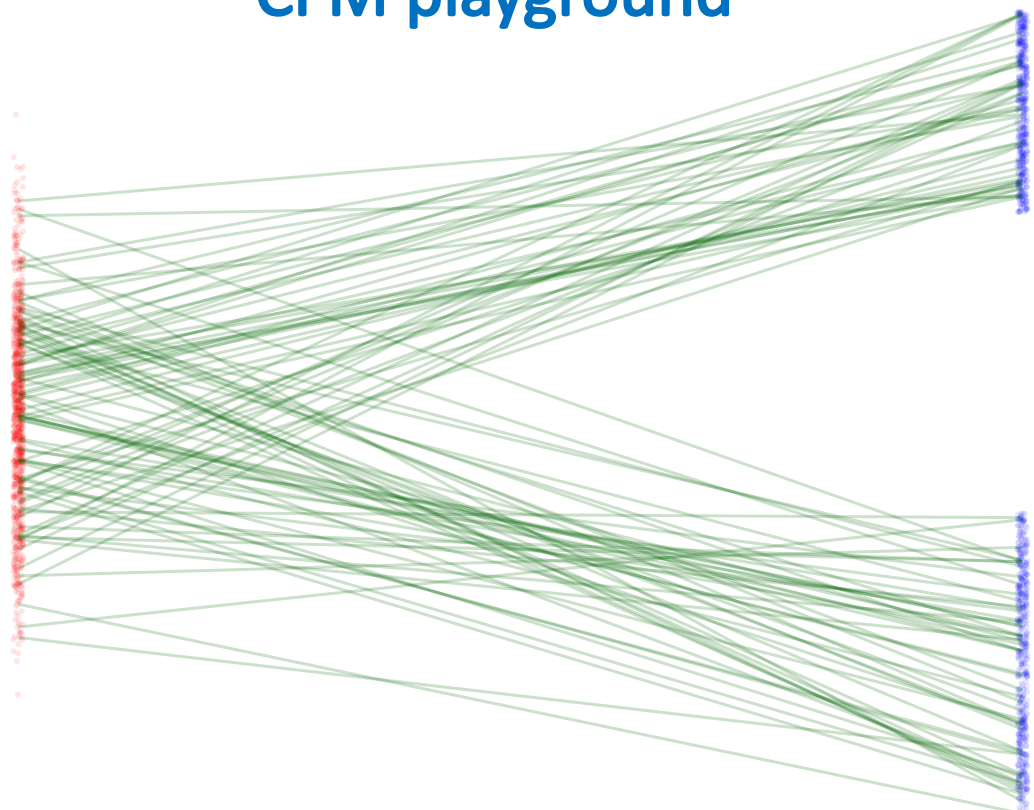
G⇝.
U⇝.
G2⇝.
U2⇝.

p_1

.⇝G
.⇝U
.⇝G2
.⇝U2

z

–
▷
≋
☹
☺
⩵
⚛
⊂
⊂



z

1
3
*

×
1s
.5s
.2s

Diffusion: denoising diffusion probabilistic models (DDPM)

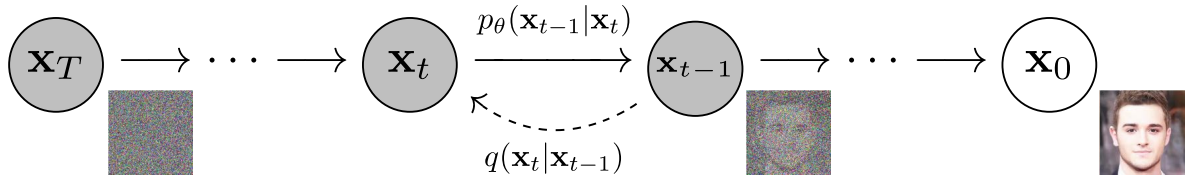


Figure 2: The directed graphical model considered in this work.

Principle

- progressively noise you data
- use that data to learn an infinitesimal denoiser

1. … the latent space dimension is the same as the data space dimension (like CNF, contrary to PCA, GAN, VAE) ↩

Diffusion: denoising diffusion probabilistic models (DDPM)

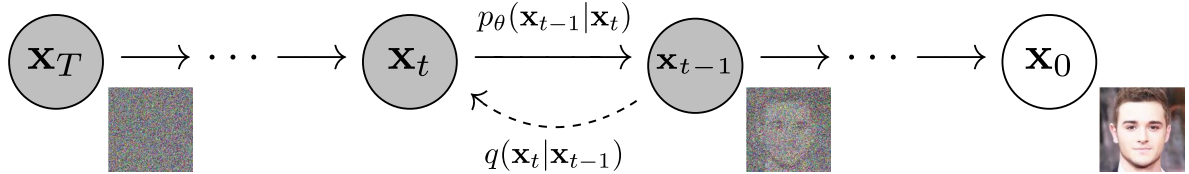


Figure 2: The directed graphical model considered in this work.

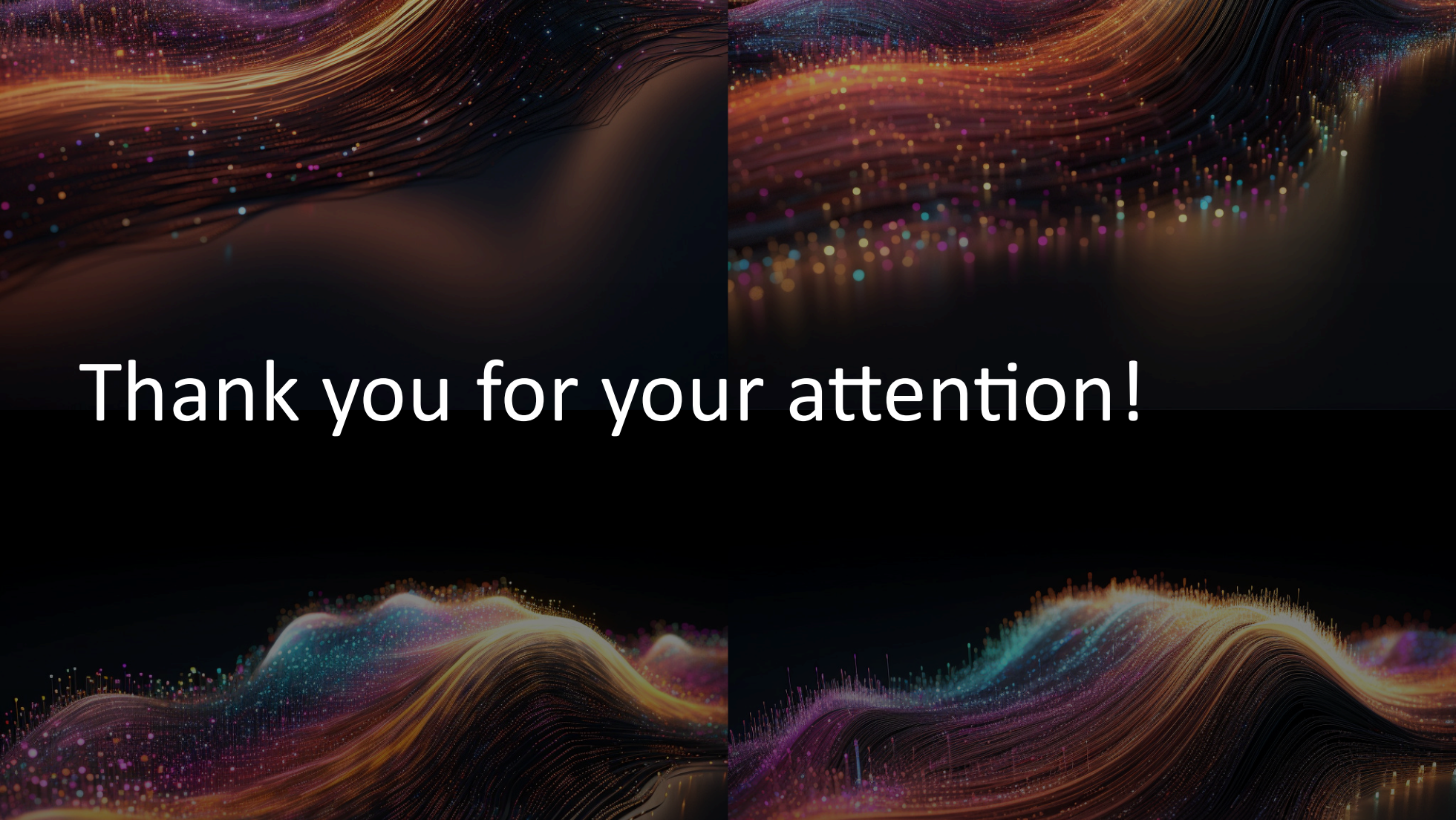
Principle

- progressively noise you data
- use that data to learn an infinitesimal denoiser

Actually

- a VAE with successive latent representations[1]
- learning a velocity field (notation trap: $t \in [\![T, 0]\!]$ instead of $t \in [0, 1]$)
- specifying a unique probability path (but stochastic flow)
- effectively supervising at every step (vs CNF)

1. … the latent space dimension is the same as the data space dimension (like CNF, contrary to PCA, GAN, VAE) ↩


Thank you for your attention!