From Zero to Generative

Learning Generative Modelling from scratch

Carolina Cuesta-Lazaro

Flatiron Institute
Institute for Advanced Studies

Art: "The art of painting" by Johannes Vermeer

Probabilistic ML has made ***high dimensional inference tractable***

$$p(\text{World}|\text{Prompt})$$

["Genie 2: A large-scale foundation model"
Parker-Holder et al (2024)]

Probabilistic ML has made ***high dimensional inference tractable***

$$p(\text{World}|\text{Prompt})$$

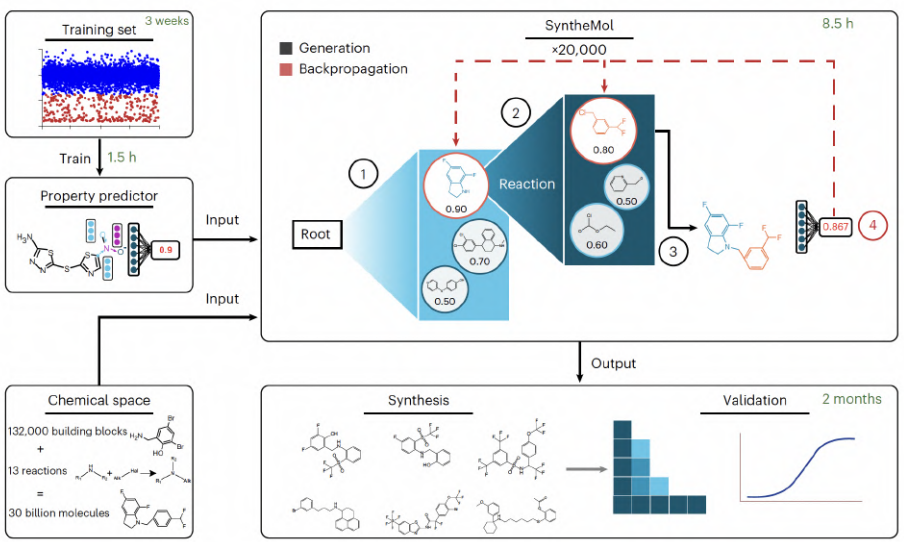
["Genie 2: A large-scale foundation model"
Parker-Holder et al (2024)]

["Genie 3: A new frontier for world models"
Parker-Holder et al (2025)]

Probabilistic ML has made ***high dimensional inference tractable***

$$p(\text{World}|\text{Prompt})$$

$$p(\text{Drug}|\text{Properties})$$



["Genie 2: A large-scale foundation model"
Parker-Holder et al (2024)]

["Genie 3: A new frontier for world models"
Parker-Holder et al (2025)]

["Generative AI for designing and validating easily synthesizable and structurally novel antibiotics"
Swanson et al]

Parti-350M

A portrait photo of a kangaroo wearing an orange hoodie and blue sunglasses standing on the grass in front of the Sydney Opera House holding a sign on the chest that says Welcome Friends!

https://parti.research.google

Parti-350M

Parti-750M

A portrait photo of a kangaroo wearing an orange hoodie and blue sunglasses standing on the grass in front of the Sydney Opera House holding a sign on the chest that says Welcome Friends!

https://parti.research.google

Parti-350M **Parti-750M** **Parti-3B**

A portrait photo of a kangaroo wearing an orange hoodie and blue sunglasses standing on the grass in front of the Sydney Opera House holding a sign on the chest that says Welcome Friends!

https://parti.research.google

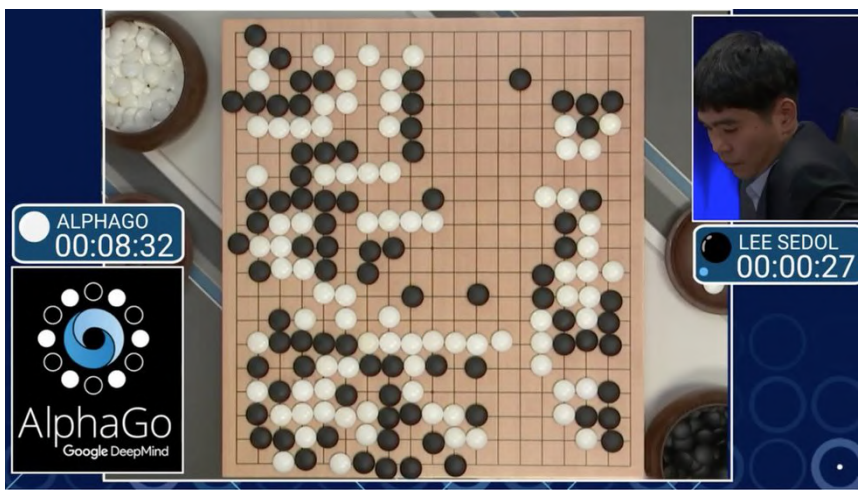
Parti-350M	Parti-750M	Parti-3B	Parti-20B
			

A portrait photo of a kangaroo wearing an orange hoodie and blue sunglasses standing on the grass in front of the Sydney Opera House holding a sign on the chest that says Welcome Friends!

https://parti.research.google

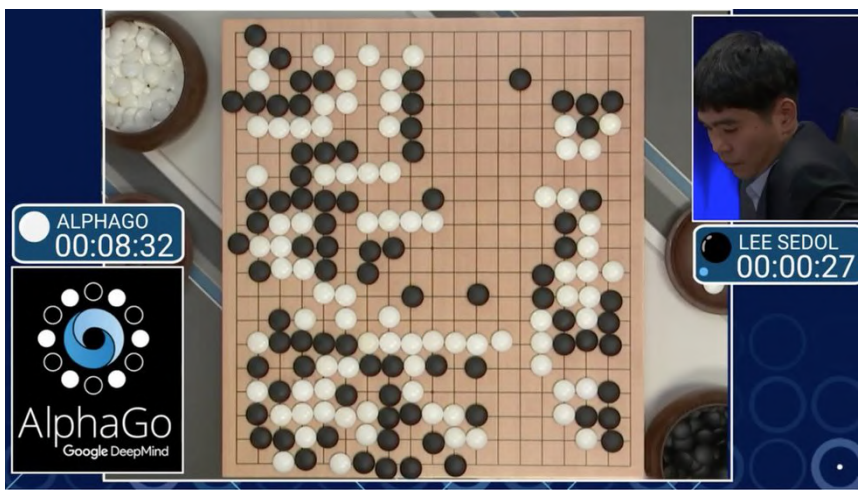
Artificial General Intelligence?

BEFORE

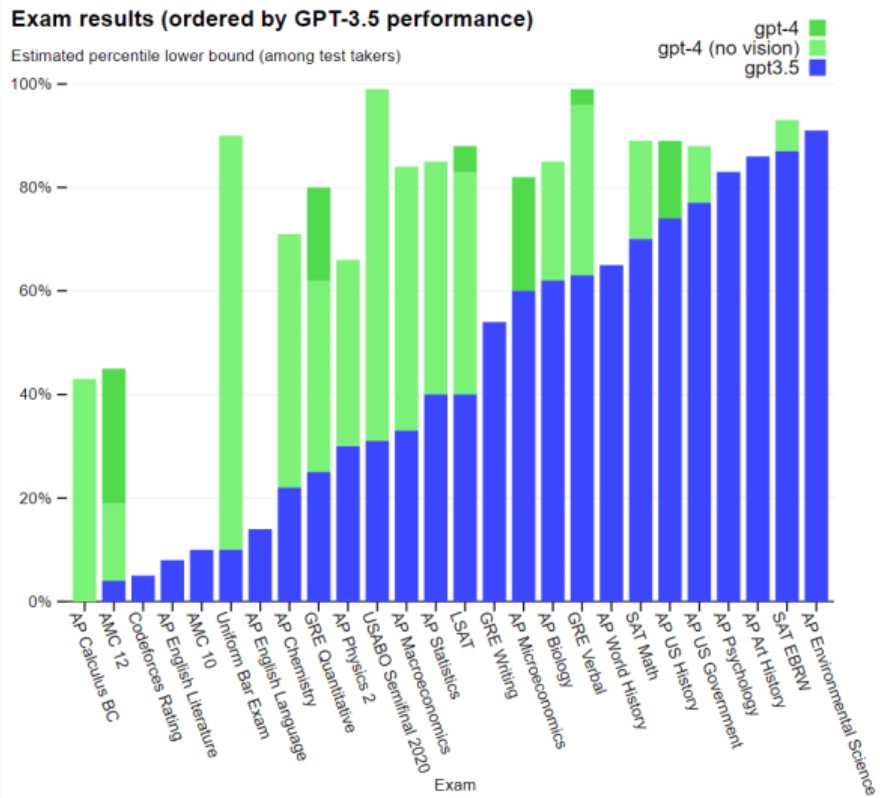


Artificial General Intelligence?

BEFORE

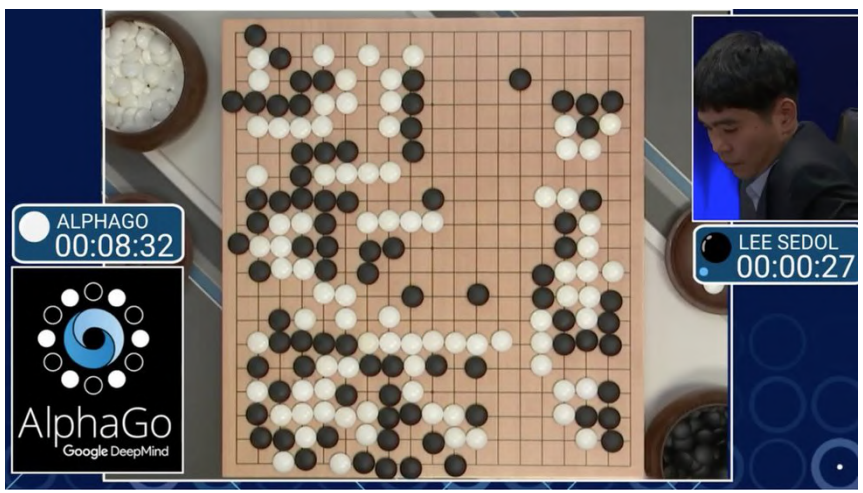


AFTER

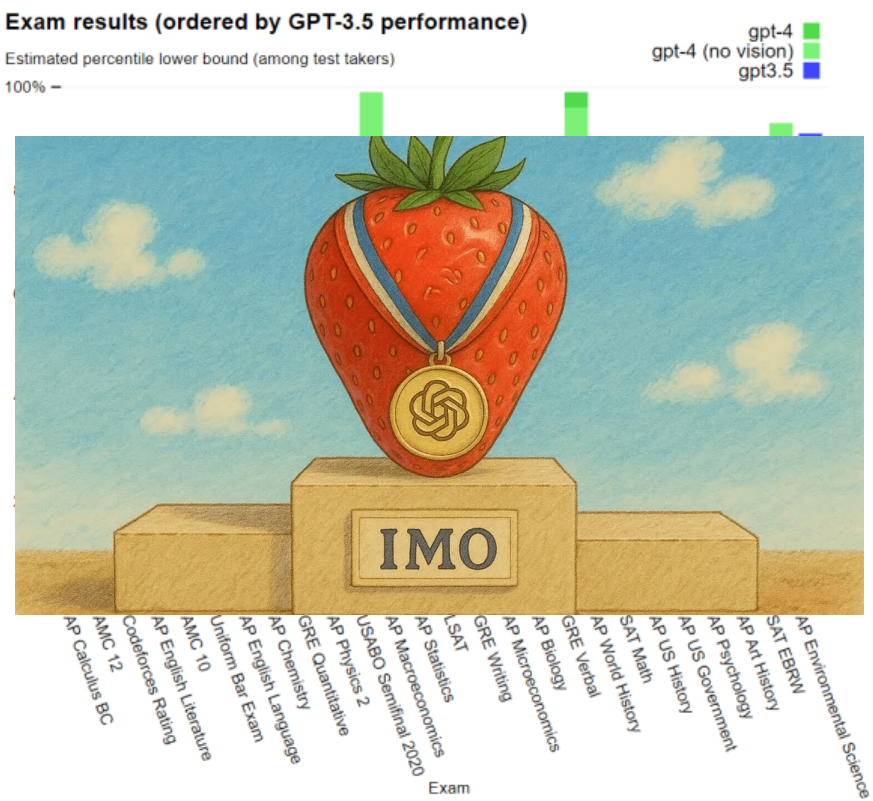


Artificial General Intelligence?

BEFORE



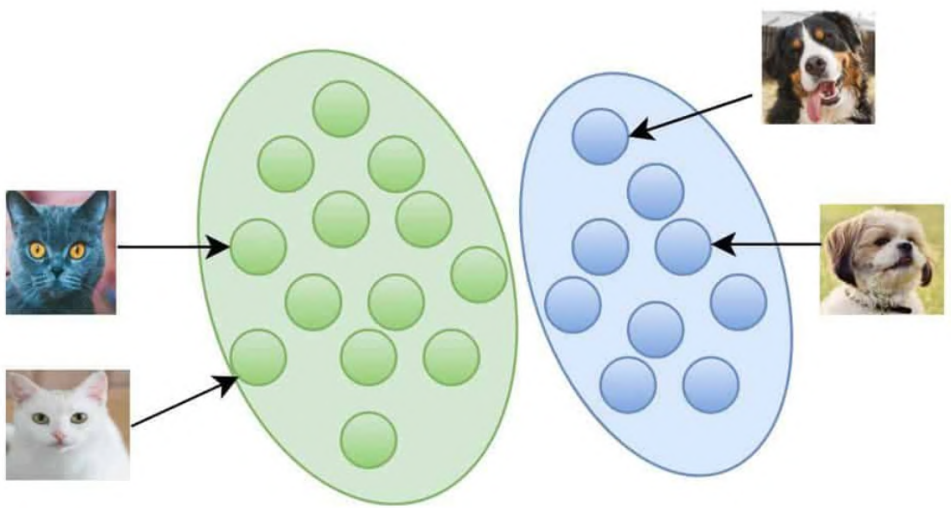
AFTER



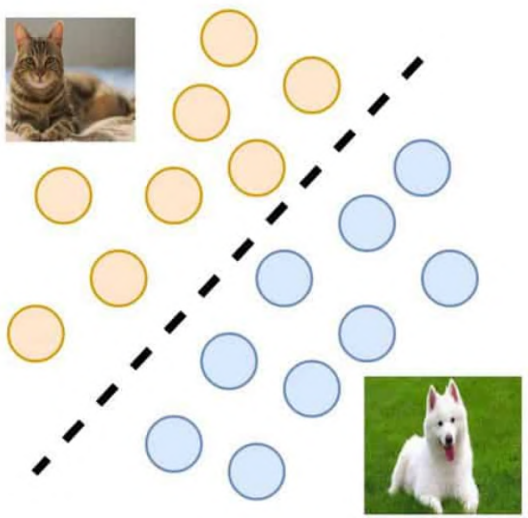
Generation vs Discrimination

$p(x)$

$p(y|x)$



Generative



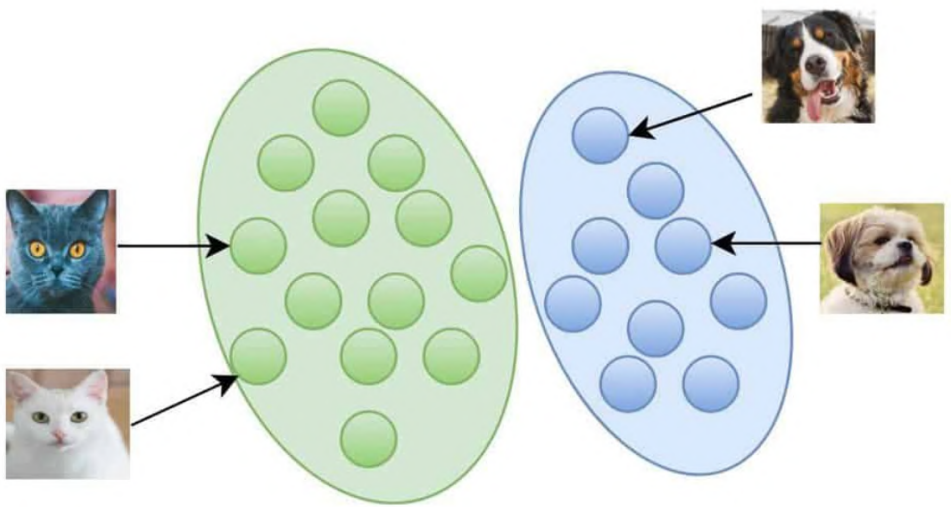
Discriminative

$p(x|y)$

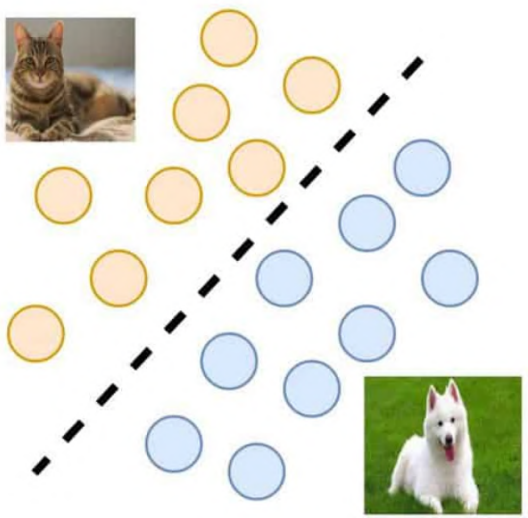
Generation vs Discrimination

$p(x)$

$p(y|x)$



Generative



Discriminative

$p(x|y)$

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}$$

Generative Models

$$p_\phi(x)$$

Generative Models

Data

$$p_\phi(x)$$

Generative Models

Data

$$p_\phi(x)$$

A PDF that we can optimize

Generative Models

Maximize the likelihood of the data

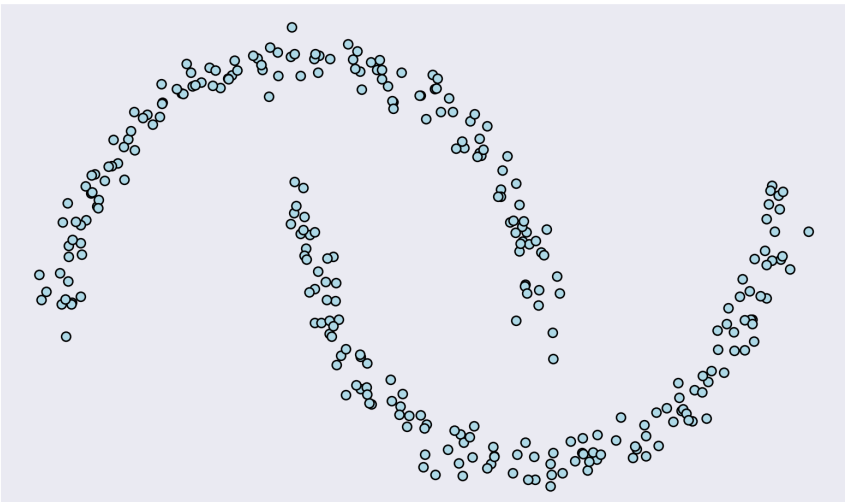
Data

$$p_\phi(x)$$

A PDF that we can optimize

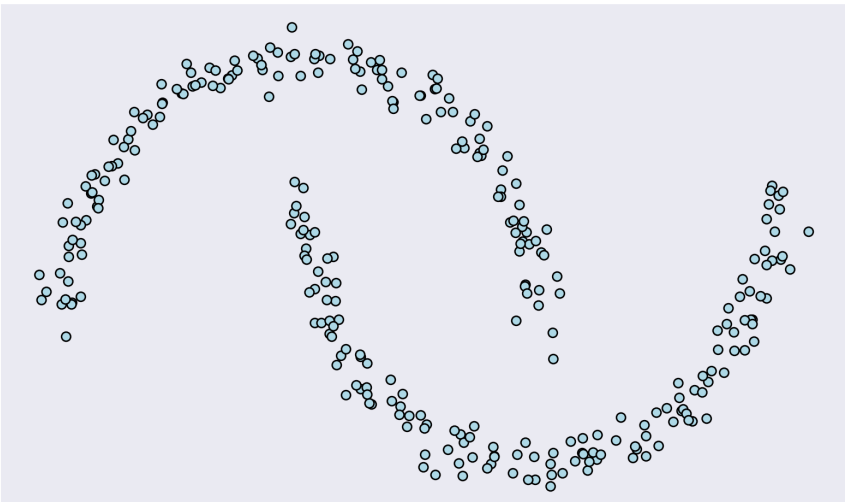
Generative Models 101

Training Samples x_{train}

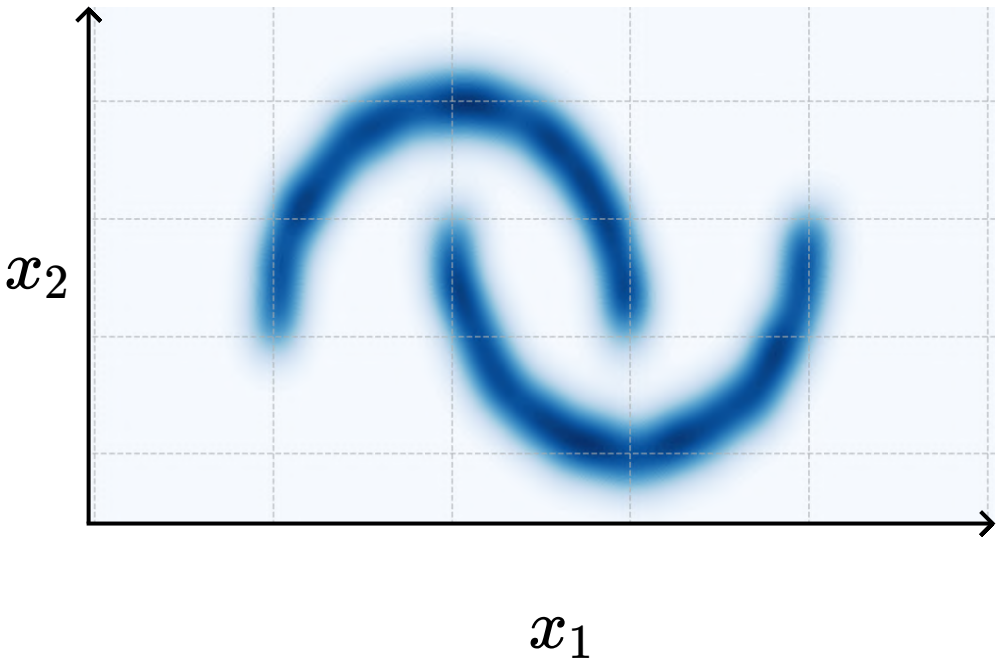


Generative Models 101

Training Samples x_{train}

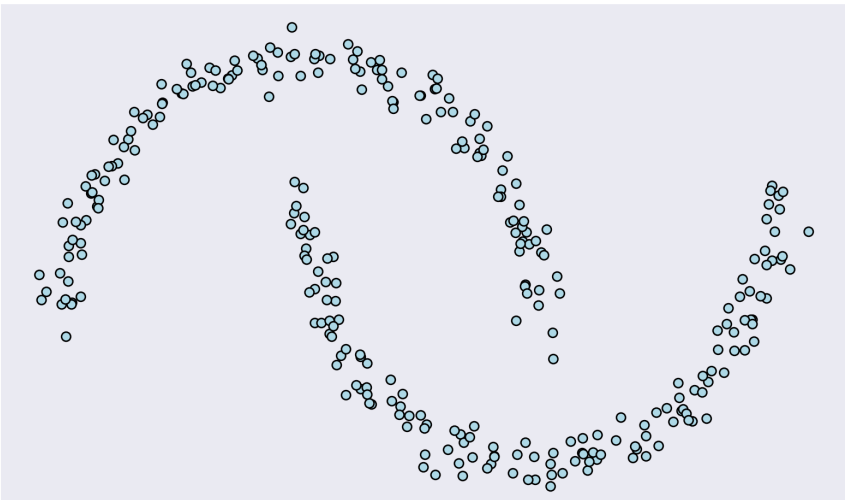


Parametric Model $p_\phi(x)$

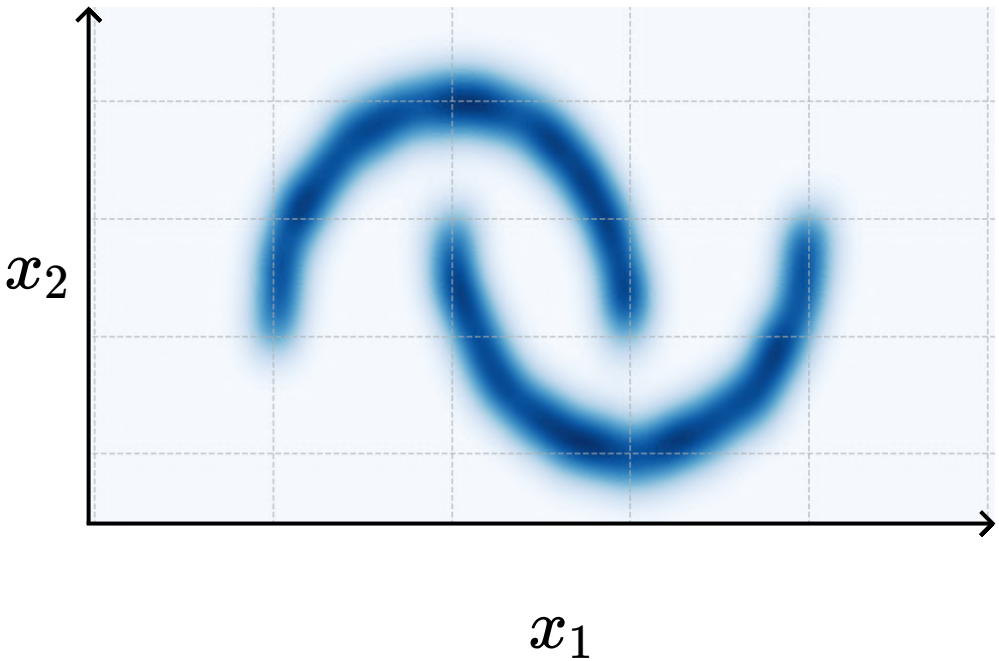


Generative Models 101

Training Samples x_{train}



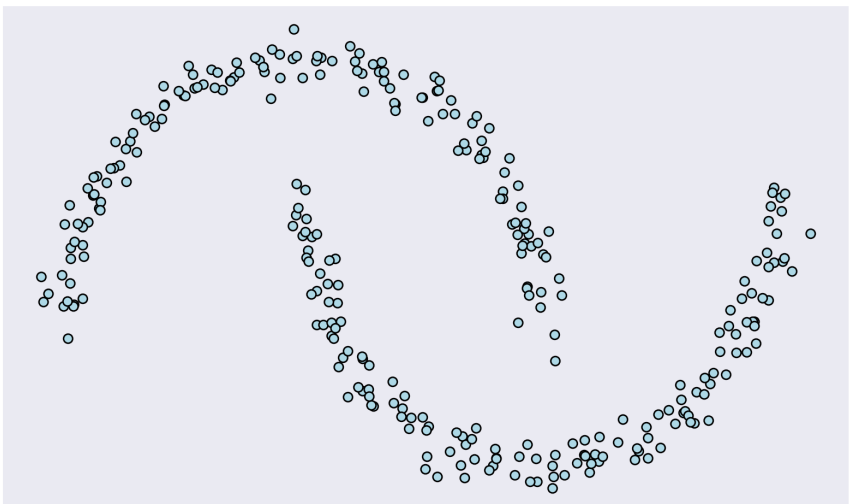
Parametric Model $p_\phi(x)$



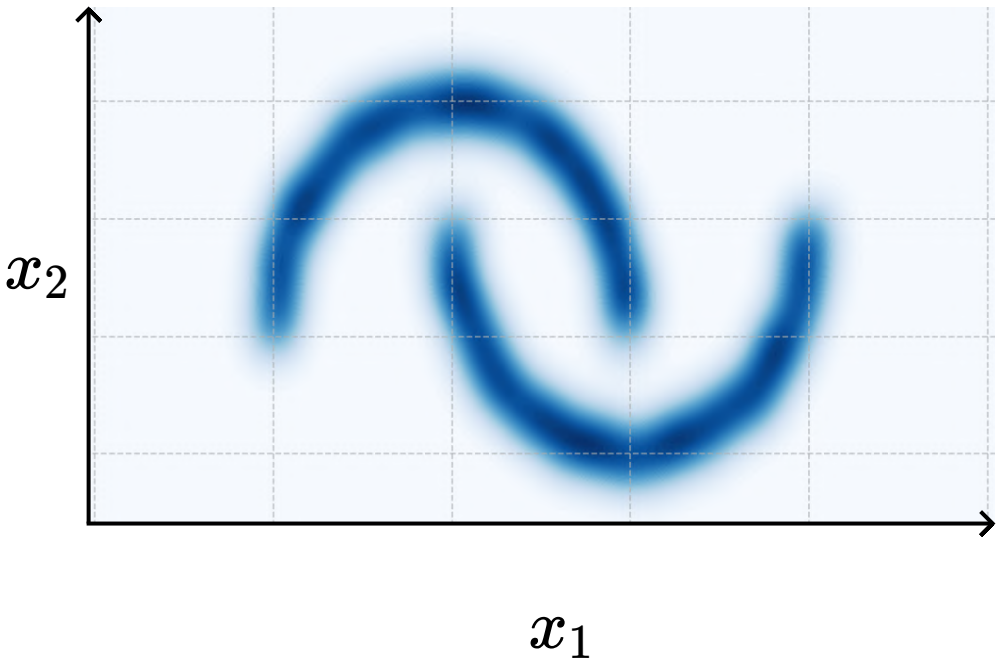
Maximize the likelihood of the training samples

Generative Models 101

Training Samples x_{train}



Parametric Model $p_\phi(x)$

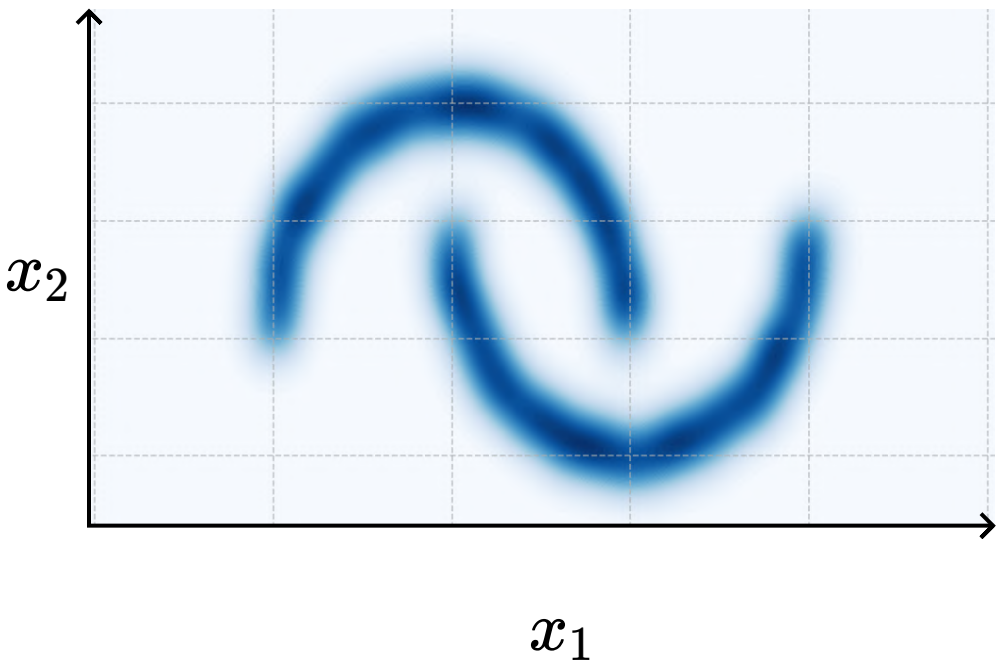


Maximize the likelihood of the training samples

$$\hat{\phi} = \arg\max \left[\log p_\phi(x_{\text{train}})\right]$$

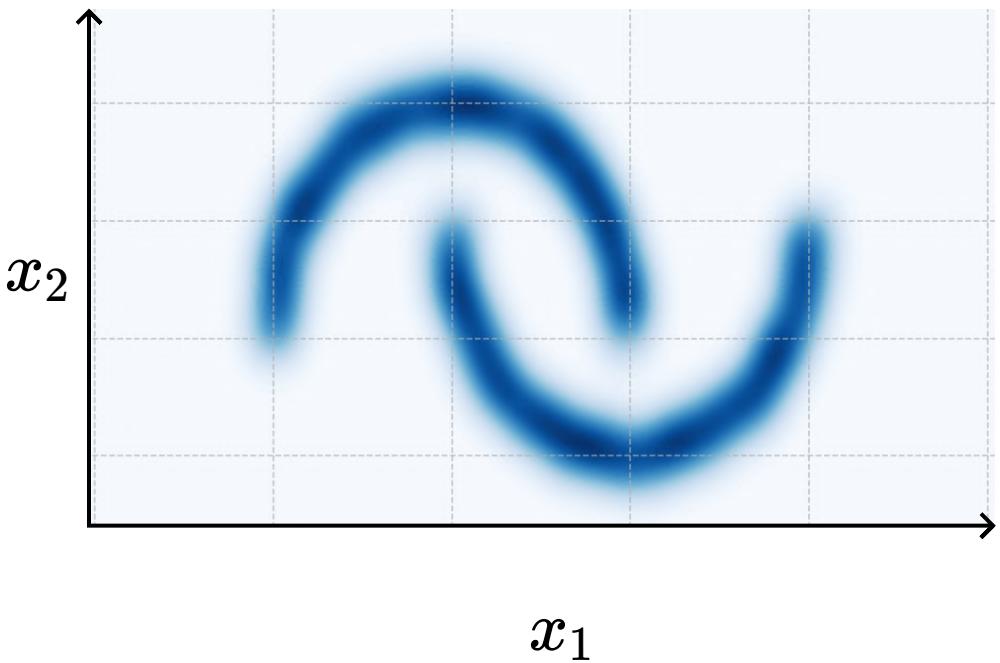
Generative Models: Simulate and Analyze

Trained Model $p_\phi(x)$

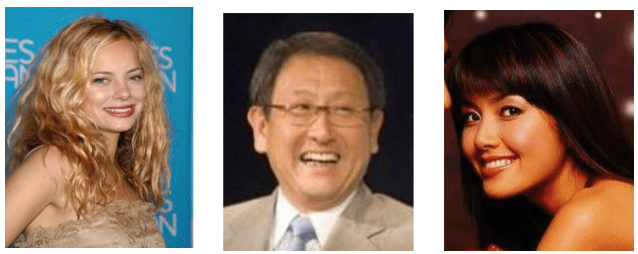


Generative Models: Simulate and Analyze

Trained Model $p_\phi(x)$

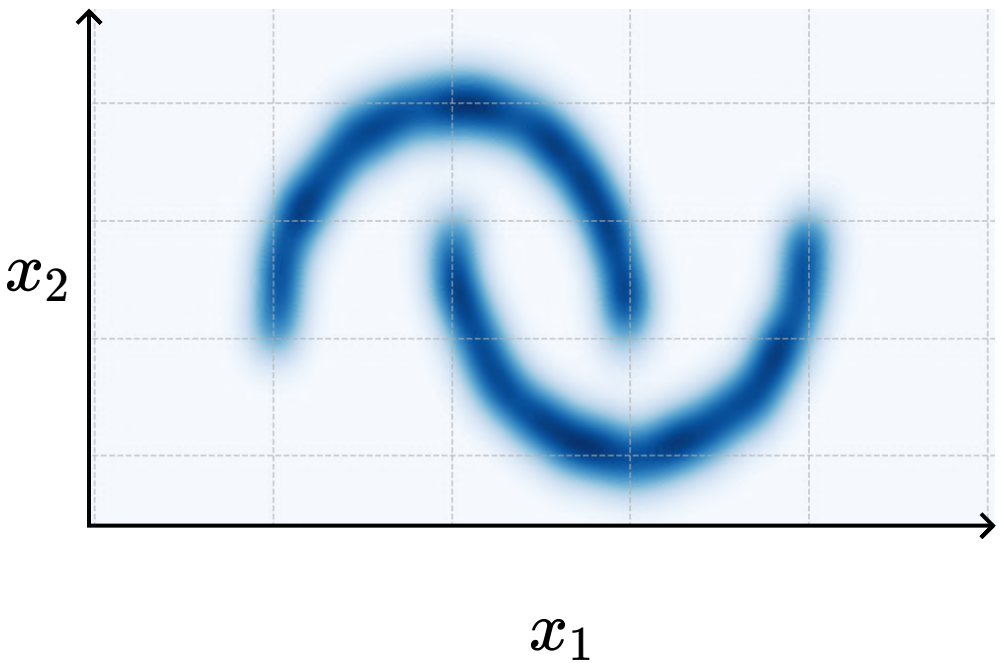


Generate Novel Samples

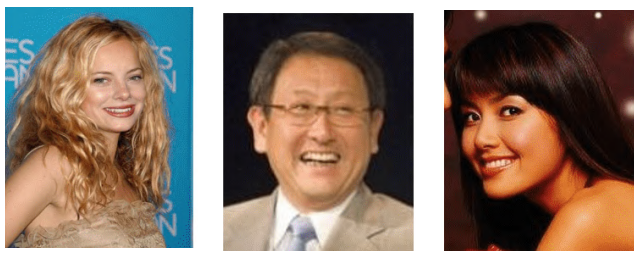


Generative Models: Simulate and Analyze

Trained Model $p_\phi(x)$

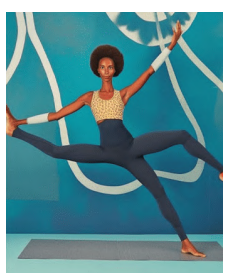


Generate Novel Samples



Evaluate probabilities

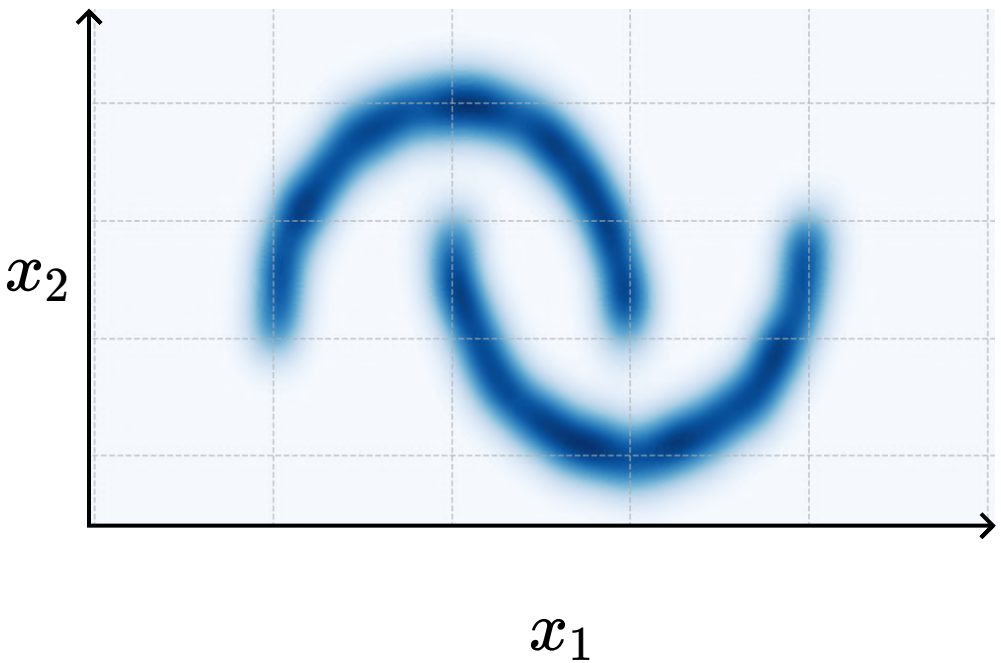
High Probability



Low Probability

Generative Models: Simulate and Analyze

Trained Model $p_\phi(x)$



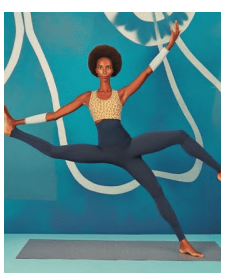
Generate Novel Samples

✅ *Generative Model*

✅ *Simulator*

Evaluate probabilities

High Probability



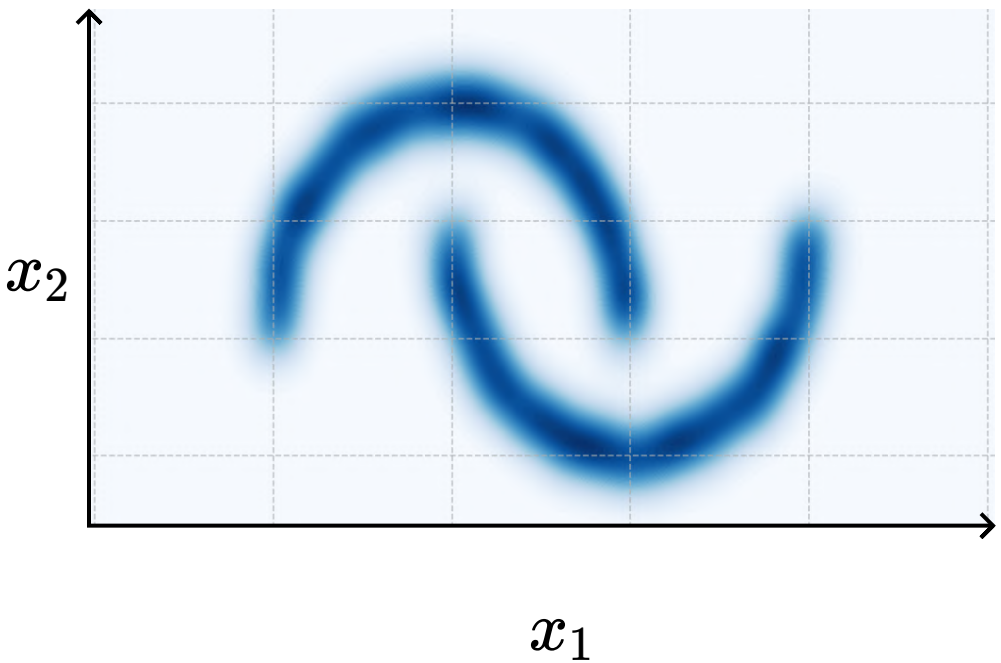
Low Probability

✅ *Generative Model*

❌ *Simulator*

Generative Models: Simulate and Analyze

Trained Model $p_\phi(x)$



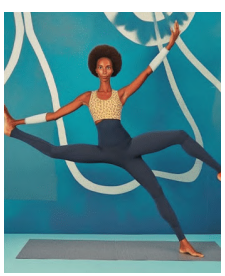
Fast emulators

Generate Novel Samples

- *Generative Model*
- *Simulator*

Evaluate probabilities

High Probability

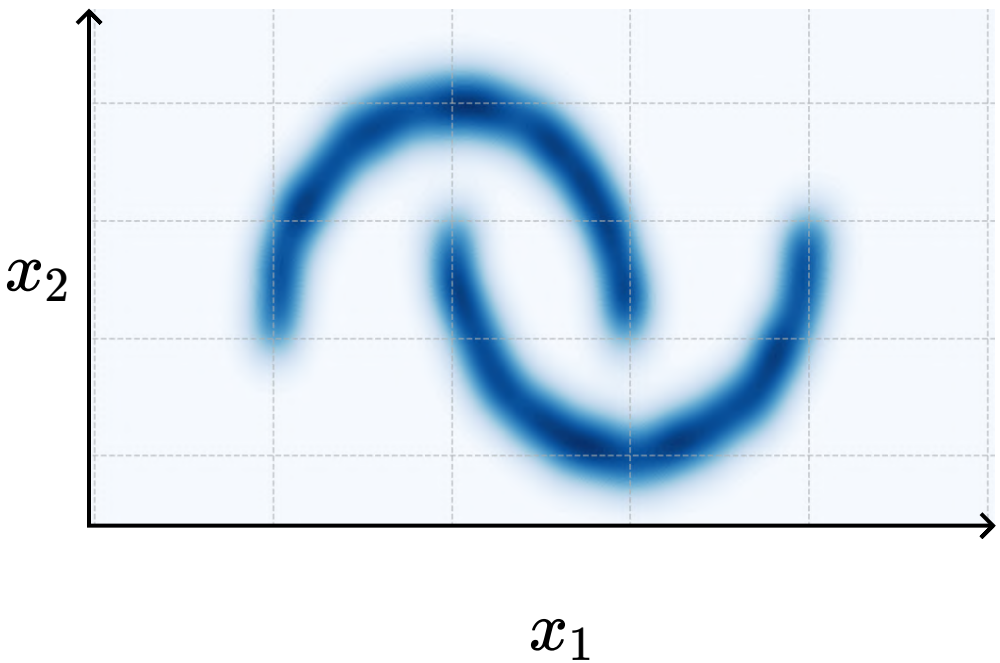


Low Probability

- *Generative Model*
- *Simulator*

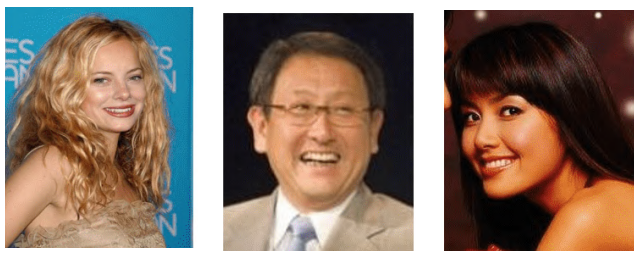
Generative Models: Simulate and Analyze

Trained Model $p_\phi(x)$



Fast emulators

Generate Novel Samples



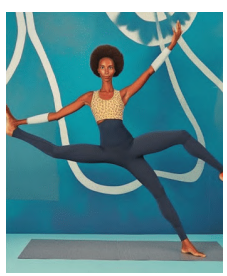
✅ *Generative Model*

✅ *Simulator*

Testing Theories

Evaluate probabilities

High Probability



Low Probability

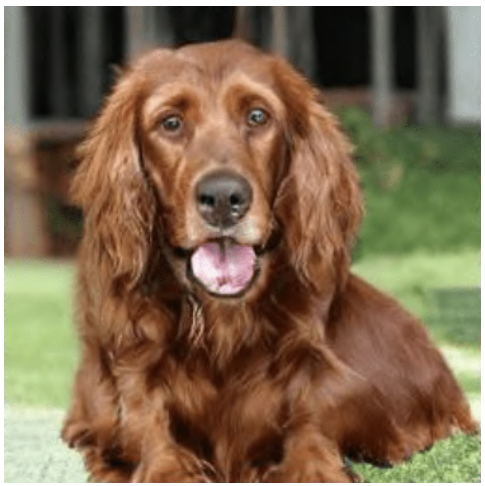
✅ *Generative Model*

❌ *Simulator*

Contrastive Learning

A folk music band of anthropomorphic autumn leaves playing bluegrass instrument

BigGAN

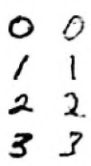


VAEs

GANS

2006 2014 2017 2019 2022 2023

Deep Belief Networks



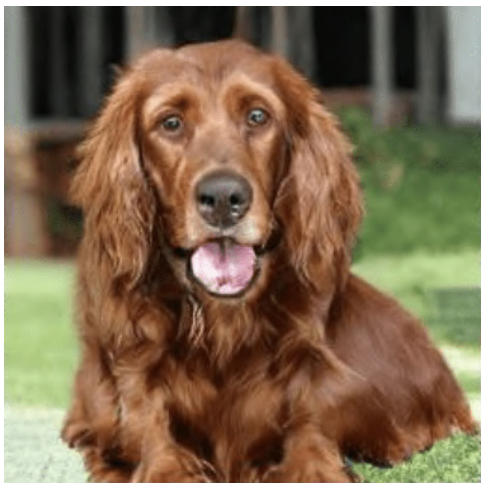
Normalising Flows

Diffusion Models

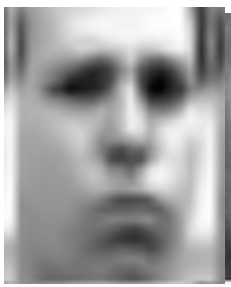
Contrastive Learning

A folk music band of anthropomorphic autumn leaves playing bluegrass instrument

BigGAN



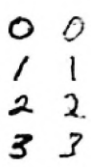
VAEs



GANS

2006 2014 2017 2019 2022 2023

Deep Belief Networks

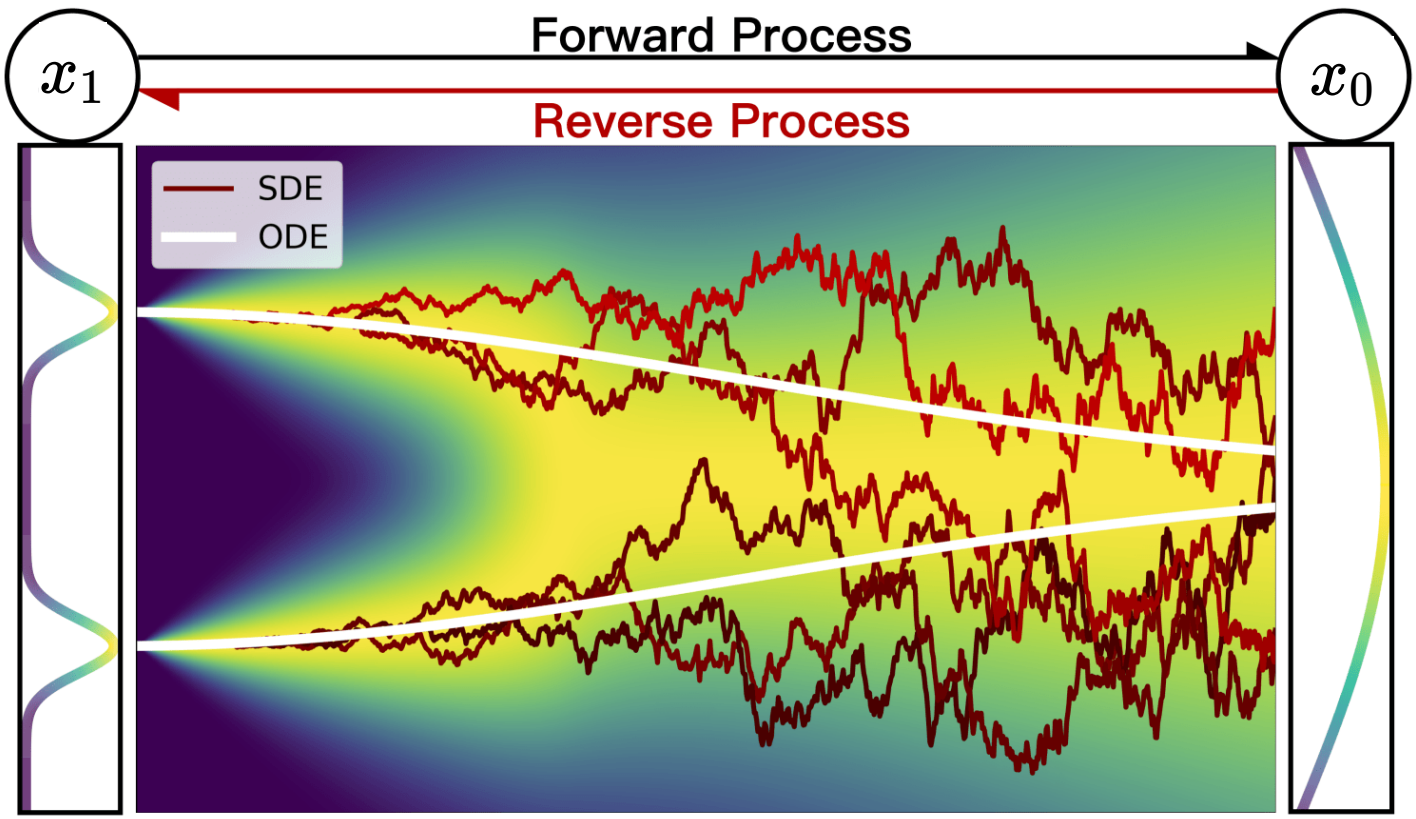


Normalising Flows

Diffusion Models

Bridging two distributions

"Creating noise from data is easy;
creating data from noise is generative modeling."
Yang Song



Data

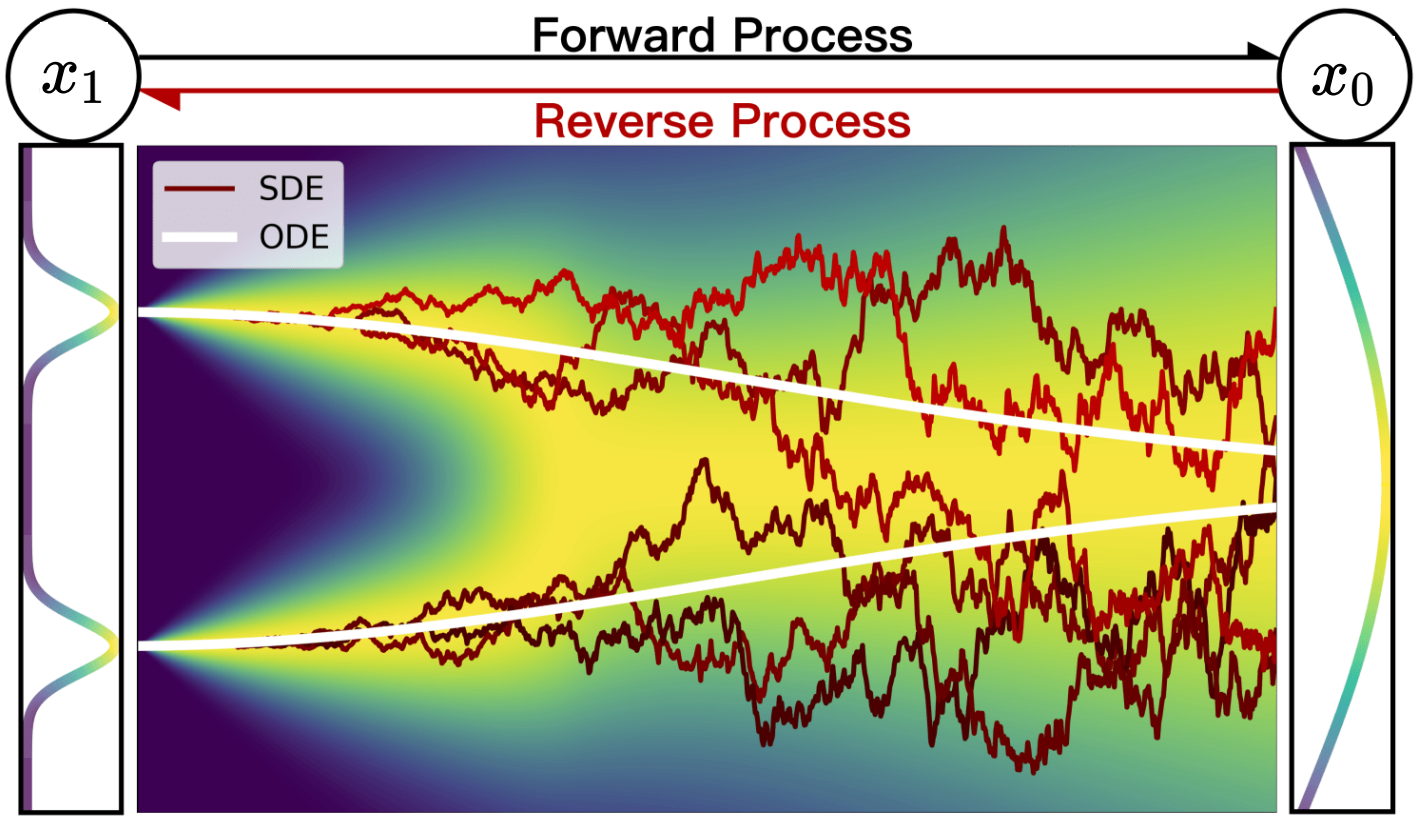
Base

Bridging two distributions

is p(x0) restricted?

Diffusion: p(x0) is Gaussian

Normalising flows: p(x0) can be evaluated



Data

Base

Bridging two distributions

How is the bridge constrained?

Normalizing flows: Reverse = Forward inverse

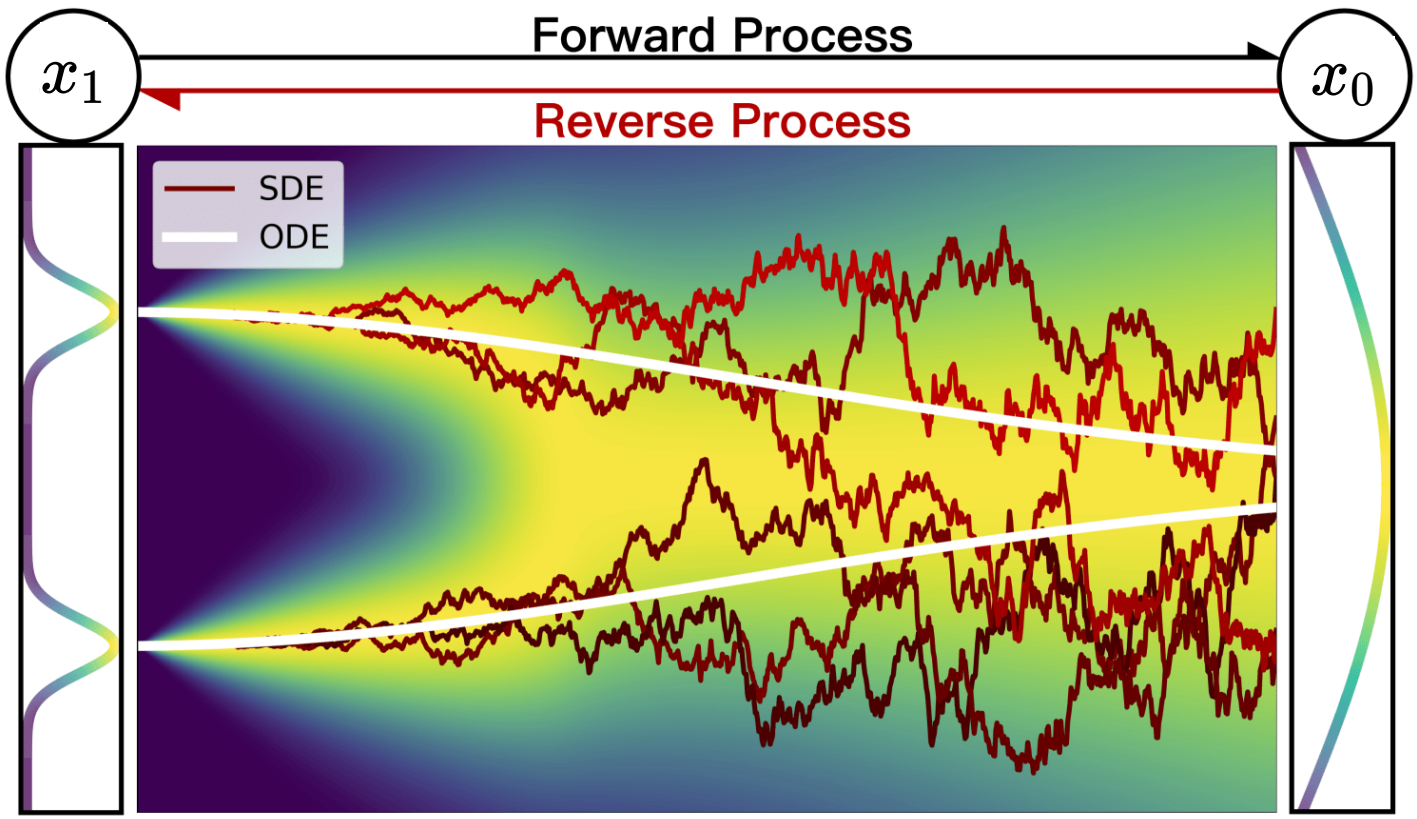
Diffusion: Forward = Gaussian noising

Flow Matching: Forward = Interpolant

is p(x0) restricted?

Diffusion: p(x0) is Gaussian

Normalising flows: p(x0) can be evaluated



Data

Base

Bridging two distributions

How is the bridge constrained?

Normalizing flows: Reverse = Forward inverse

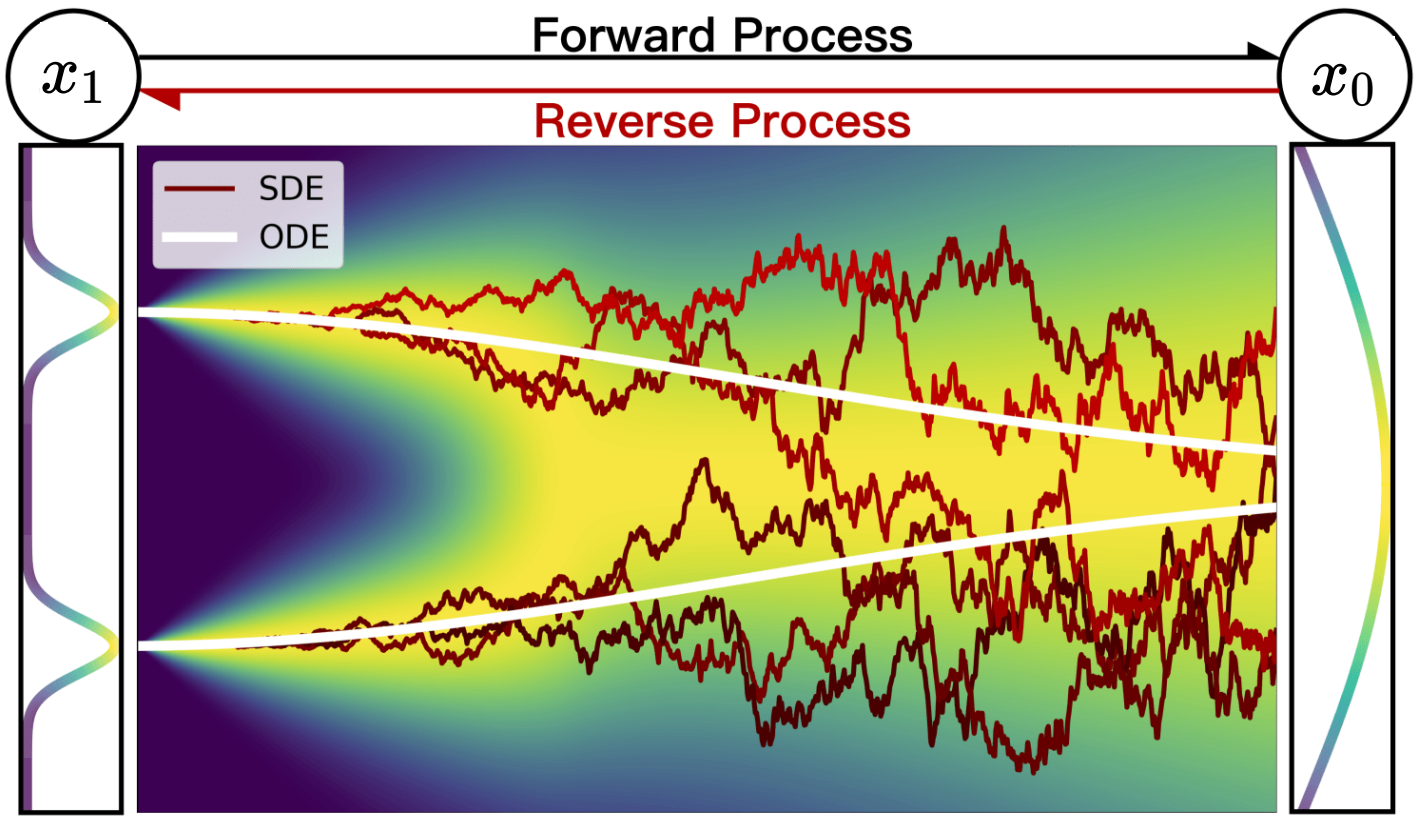
Diffusion: Forward = Gaussian noising

Flow Matching: Forward = Interpolant

is p(x0) restricted?

Diffusion: p(x0) is Gaussian

Normalising flows: p(x0) can be evaluated



Data

Base

Is bridge stochastic (SDE) or deterministic (ODE)?

Diffusion: Stochastic (SDE) *Normalising flows: Deterministic (ODE)*

Bridging two distributions

How is the bridge constrained?

Normalizing flows: Reverse = Forward inverse

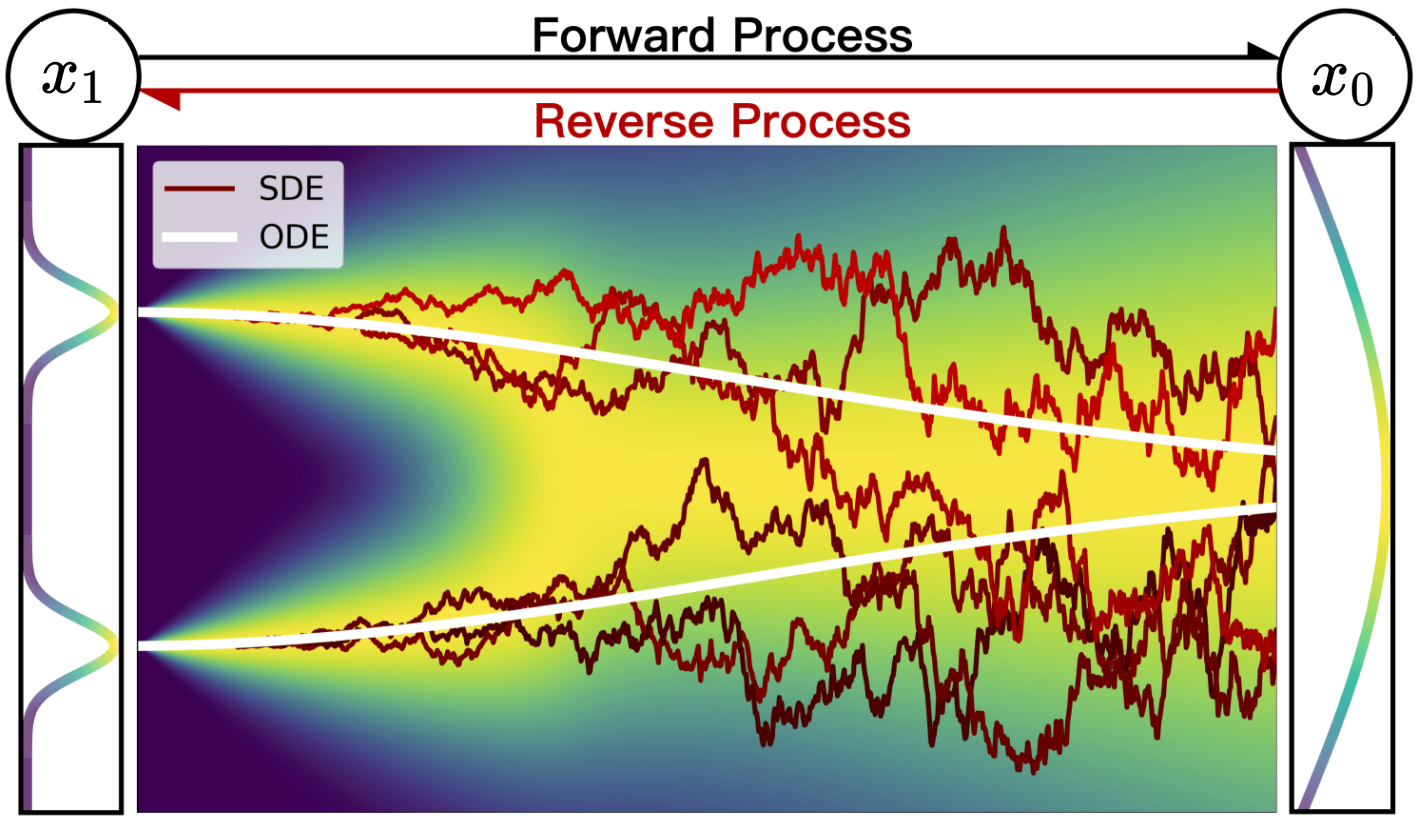
Diffusion: Forward = Gaussian noising

Flow Matching: Forward = Interpolant

is p(x0) restricted?

Diffusion: p(x0) is Gaussian

Normalising flows: p(x0) can be evaluated



Data

Base

(Exact likelihood evaluation)

Is bridge stochastic (SDE) or deterministic (ODE)?

Diffusion: Stochastic (SDE) *Normalising flows: Deterministic (ODE)*

Flow: Change of Variables

$$X \sim \mathcal{N}(\mu, \sigma)$$

Transformation (flow):

$$Y = g(X) = aX + b$$

How is Y distributed?

Flow: Change of Variables

$$X \sim \mathcal{N}(\mu, \sigma)$$

Transformation (flow):

$$Y = g(X) = aX + b$$

How is Y distributed?

$$Y \sim N(a\mu + b, a^2\sigma^2)$$

Flow: Change of Variables

$$X \sim \mathcal{N}(\mu, \sigma)$$

Transformation (flow):

$$Y = g(X) = aX + b$$

How is Y distributed?

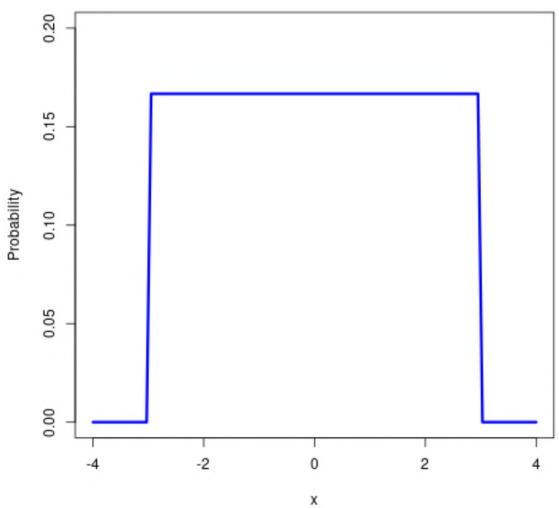
$$Y \sim N(a\mu + b, a^2\sigma^2)$$

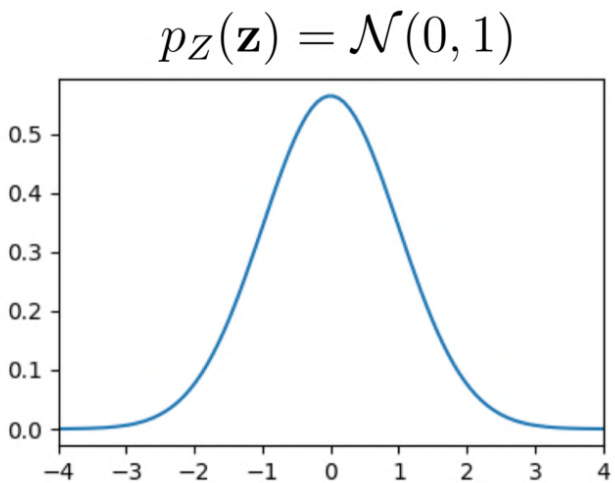
$$p_Y(y) = p_X(g^{-1}(y)) \left| \frac{dg^{-1}(y)}{dy} \right|$$

Box-Muller transform

Normalizing flows in 1934

$$\text{Uniform}(0,1) \rightarrow U_1, U_2$$

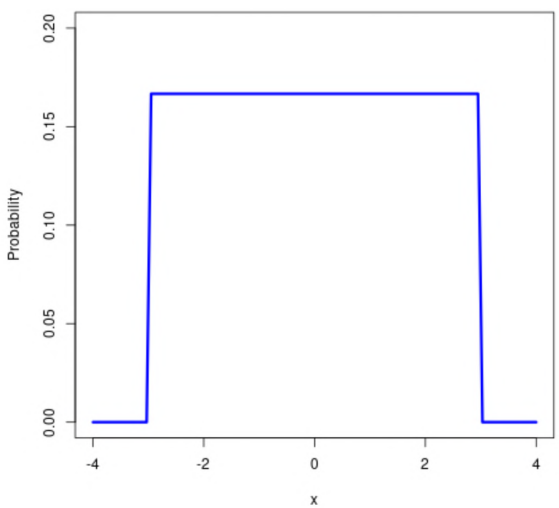


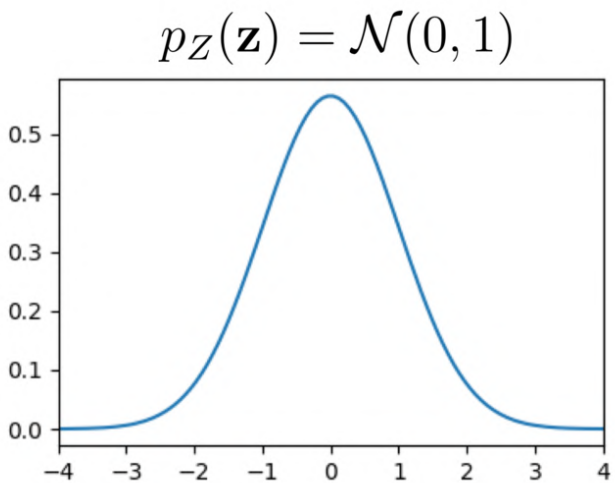


Box-Muller transform

Normalizing flows in 1934

$$\text{Uniform}(0,1) \rightarrow U_1, U_2$$





$$Z_0 = \sqrt{-2\ln U_1}\cos(2\pi U_2)$$

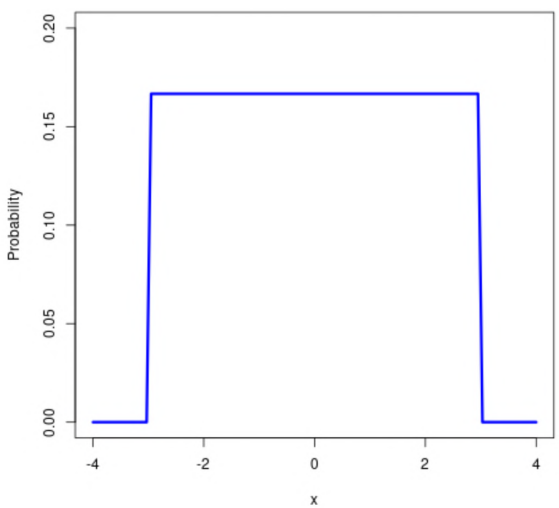
$$Z_1 = \sqrt{-2\ln U_1}\sin(2\pi U_2)$$

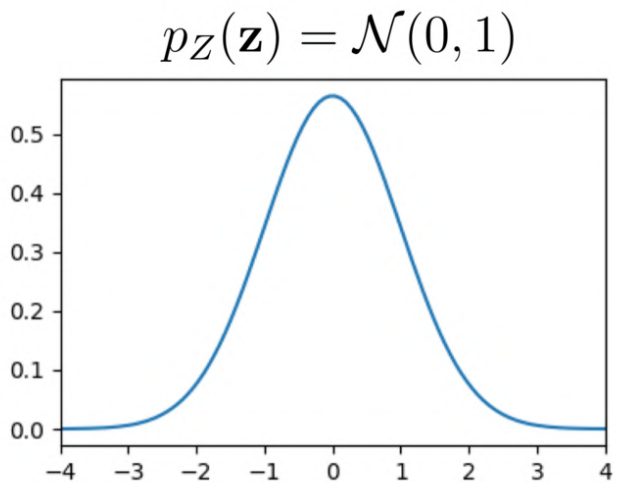
Box-Muller transform

Normalizing flows in 1934

$$\text{Uniform}(0,1) \rightarrow U_1, U_2$$

$$Z_0, Z_2 \leftarrow N(0,1)$$



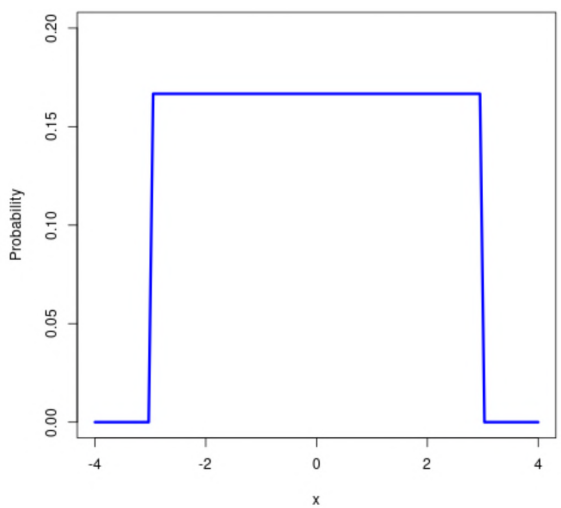


$$Z_0 = \sqrt{-2\ln U_1}\cos(2\pi U_2)$$

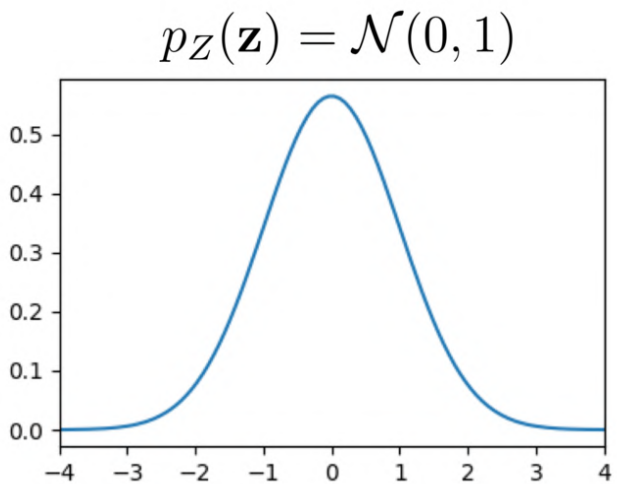
$$Z_1 = \sqrt{-2\ln U_1}\sin(2\pi U_2)$$

Tutorial 1

$$\text{Uniform}(0,1) \rightarrow U_1, U_2$$



$$Z_0, Z_2 \leftarrow N(0,1)$$

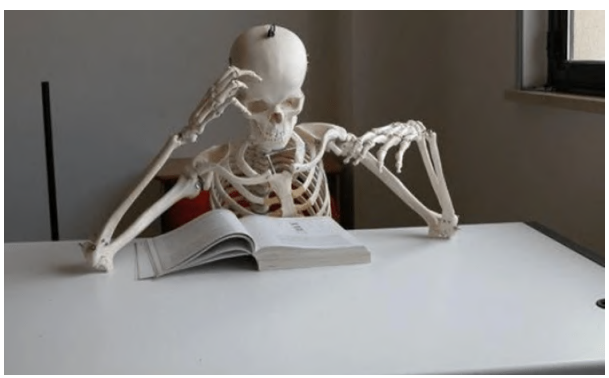


https://tinyurl.com/heidelbergschool

On tutorials

Different people learn differently

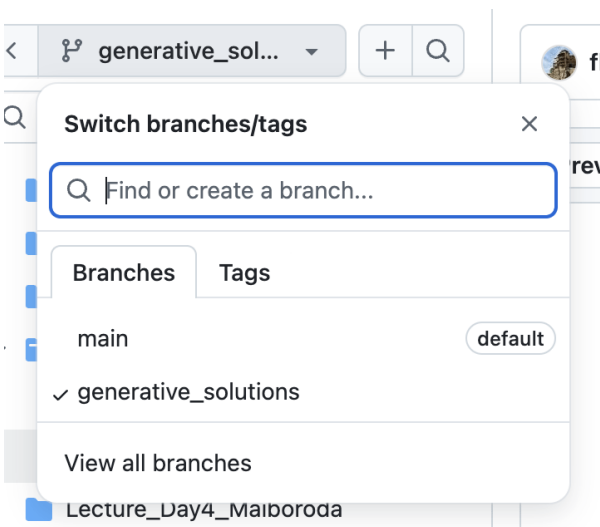
Take the data generator and start from scratch



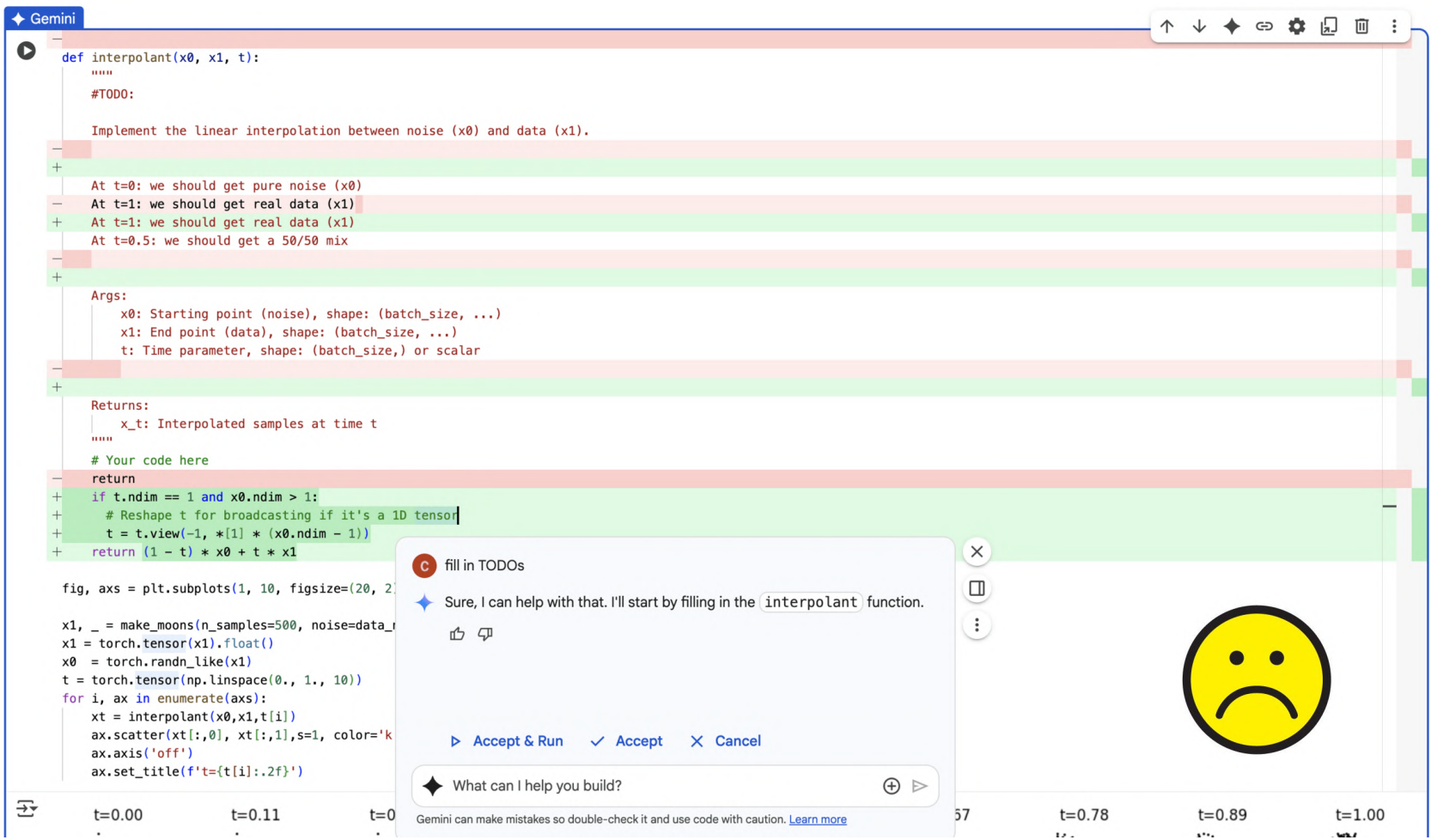
Fill in the gaps

```
In [ ]:
class FlowMatching:
    def __init__(self, velocity_network, device='cuda'):
        self.velocity_net = velocity_network
        self.device = device

    def sample_time(self, batch_size):
        """Sample random time steps between 0 and 1."""
        return torch.rand(batch_size, device=self.device)

    def interpolate(self, x0, x1, t):
        """
        Linear interpolation between noise (x0) and data (x1).
        At t=0: return x0 (noise)
        At t=1: return x1 (data)
        """
        # TODO: fill in the interpolant again

        # Your code here

        return

    def compute_target_velocity(self, x0, x1, t):
        """
        #TODO
        Compute the target velocity field that the neural network should learn.

        Think about this: if we have x_t = (1-t)*x0 + t*x1,
        what is dx_t/dt (the derivative with respect to time)?
        """
        # Your code here
        return # What is dx_t/dt?
```

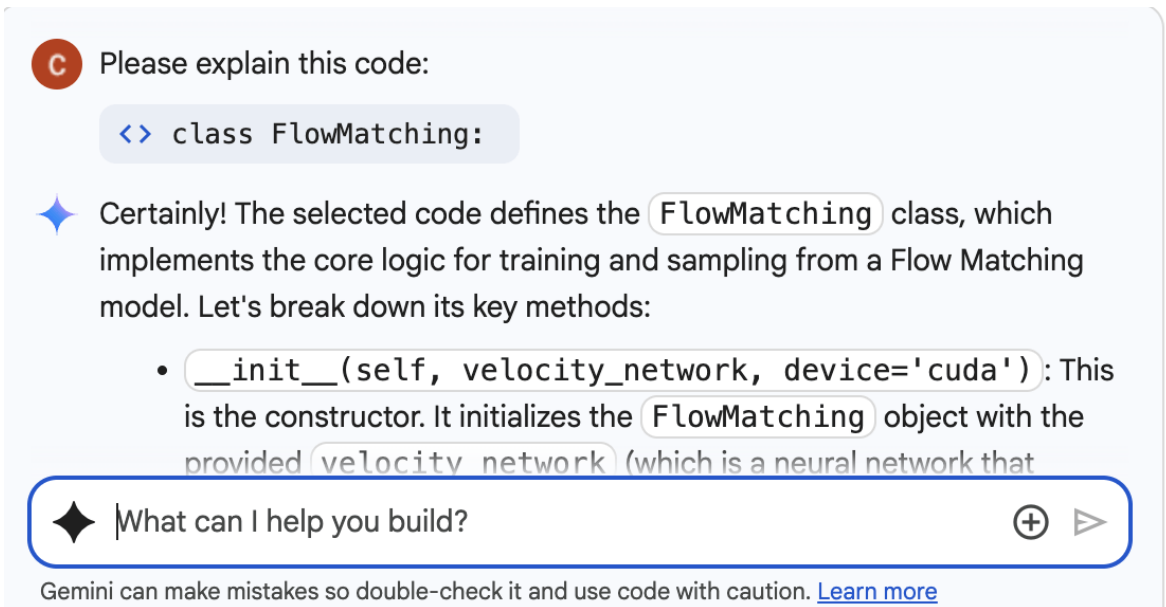
Start from a working solution and play around



On tutorials

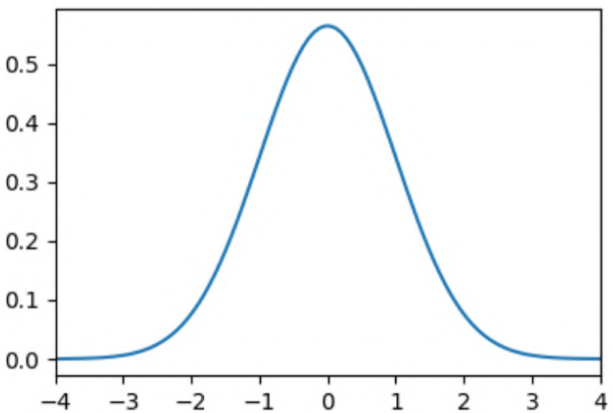


On tutorials



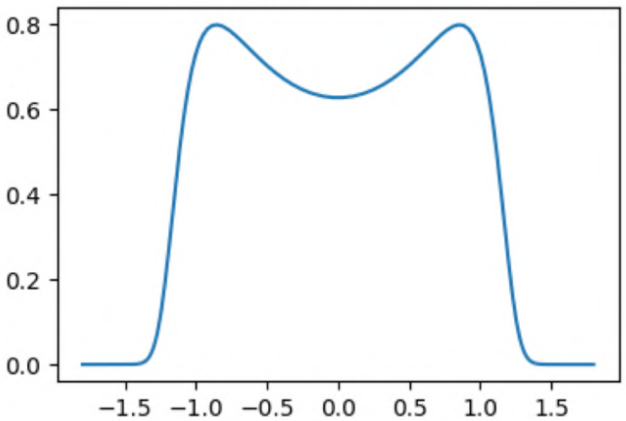
Normalizing flows

$$p_Z(\mathbf{z}) = \mathcal{N}(0, 1)$$



Base distribution

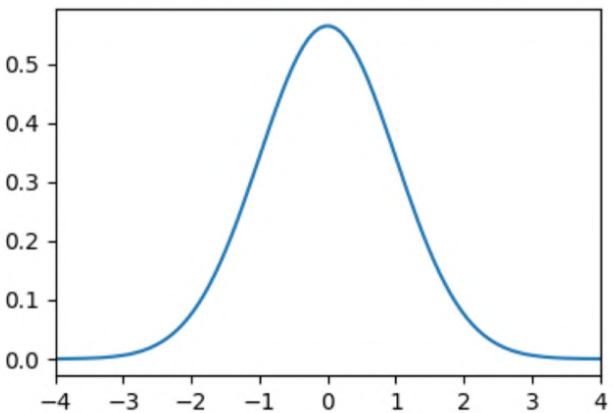
$$p_X(\mathbf{x})$$



Target distribution

Normalizing flows

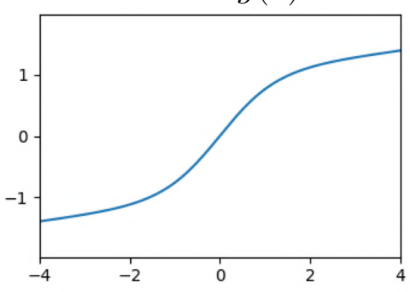
$p_Z(\mathbf{z}) = \mathcal{N}(0, 1)$



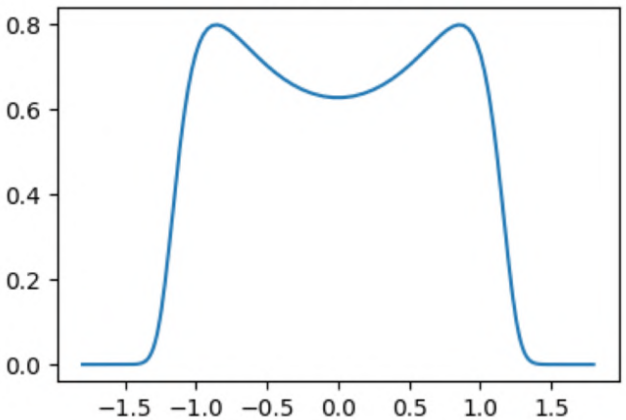
Base distribution

Invertible transformation

$\mathbf{x} = g(\mathbf{z})$



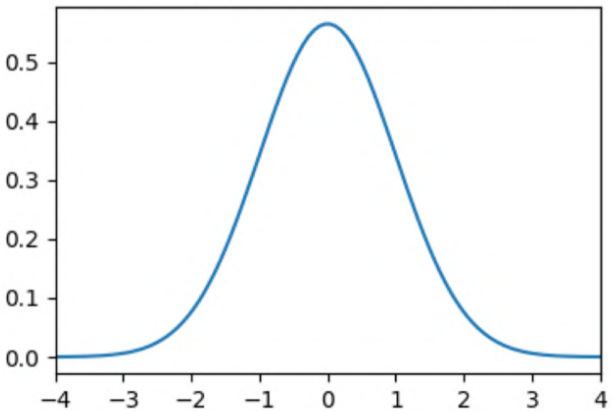
$p_X(\mathbf{x})$



Target distribution

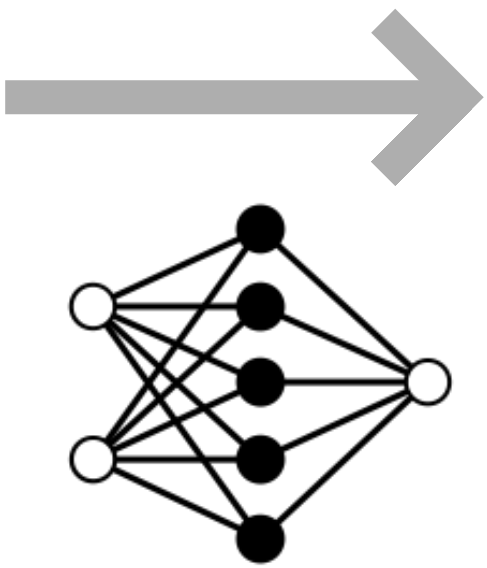
Normalizing flows

$p_Z(\mathbf{z}) = \mathcal{N}(0, 1)$

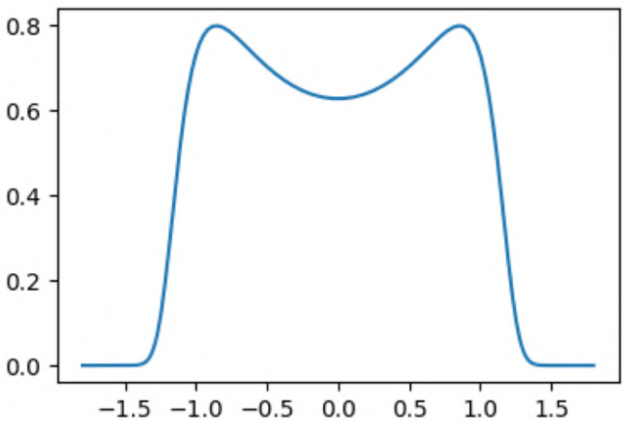


Base distribution

Invertible transformation



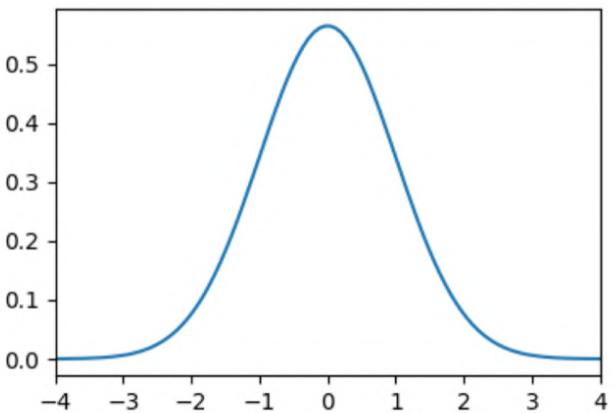
$p_X(\mathbf{x})$



Target distribution

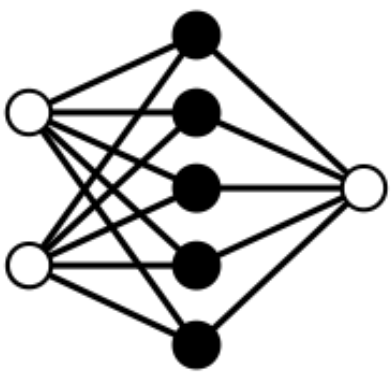
Normalizing flows

$p_Z(\mathbf{z}) = \mathcal{N}(0, 1)$

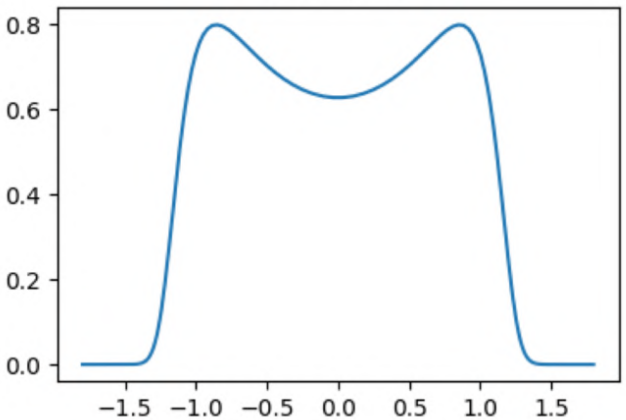


Base distribution

Invertible transformation



$p_X(\mathbf{x})$

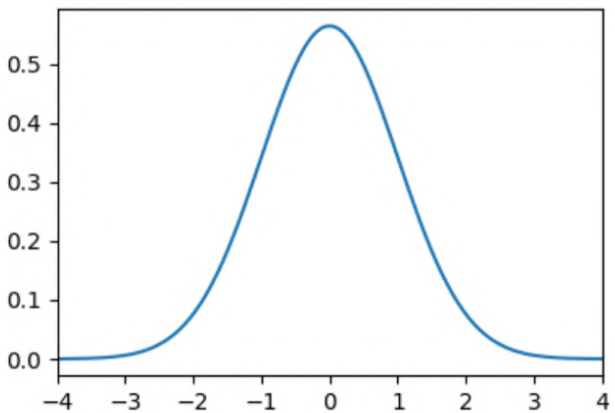


Target distribution

$z \sim p_Z(z)$

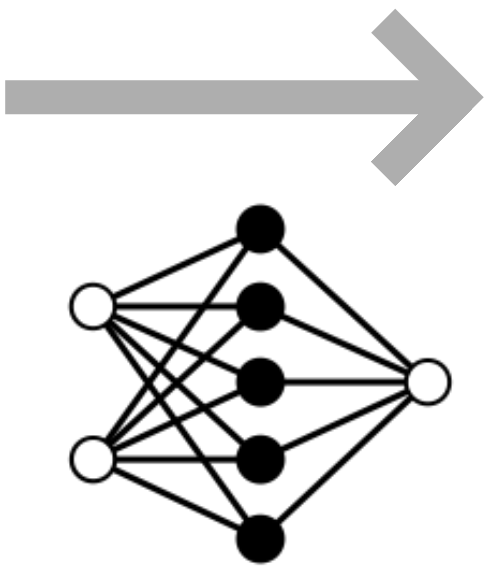
Normalizing flows

$p_Z(\mathbf{z}) = \mathcal{N}(0, 1)$

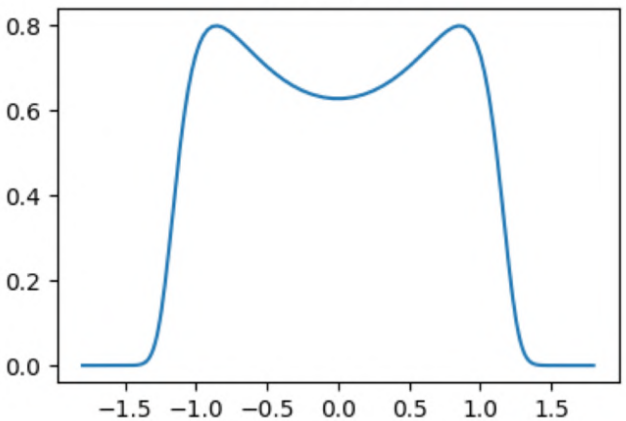


Base distribution

Invertible transformation



$p_X(\mathbf{x})$

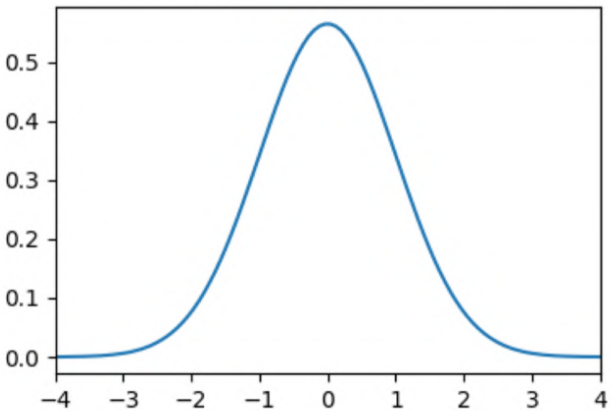


Target distribution

- $z \sim p_Z(z)$
- $p_Z(z)$

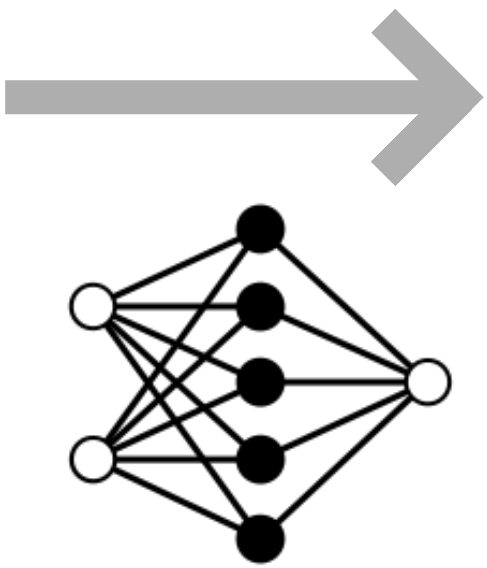
Normalizing flows

$p_Z(\mathbf{z}) = \mathcal{N}(0, 1)$

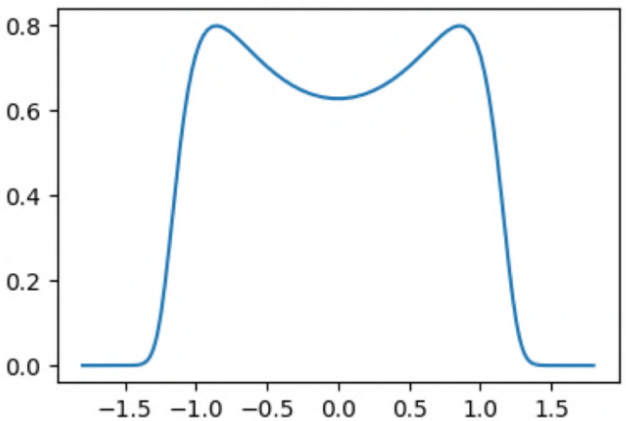


Base distribution

Invertible transformation



$p_X(\mathbf{x})$

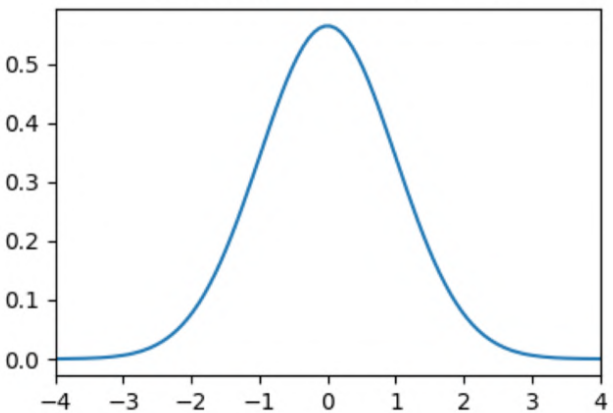


Target distribution

- $z \sim p_Z(z)$
- $p_Z(z)$
- $Z \sim \mathcal{N}(0, 1) \rightarrow g(z) \rightarrow X$

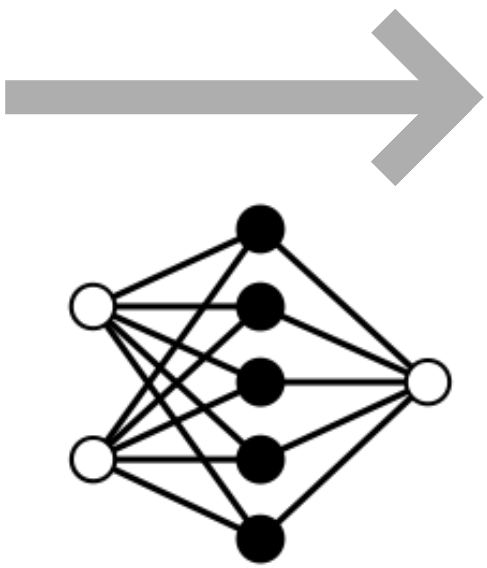
Normalizing flows

$p_Z(\mathbf{z}) = \mathcal{N}(0, 1)$

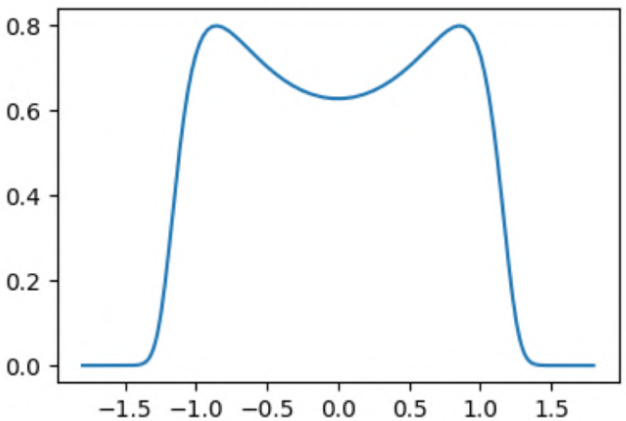


Base distribution

Invertible transformation



$p_X(\mathbf{x})$



Target distribution

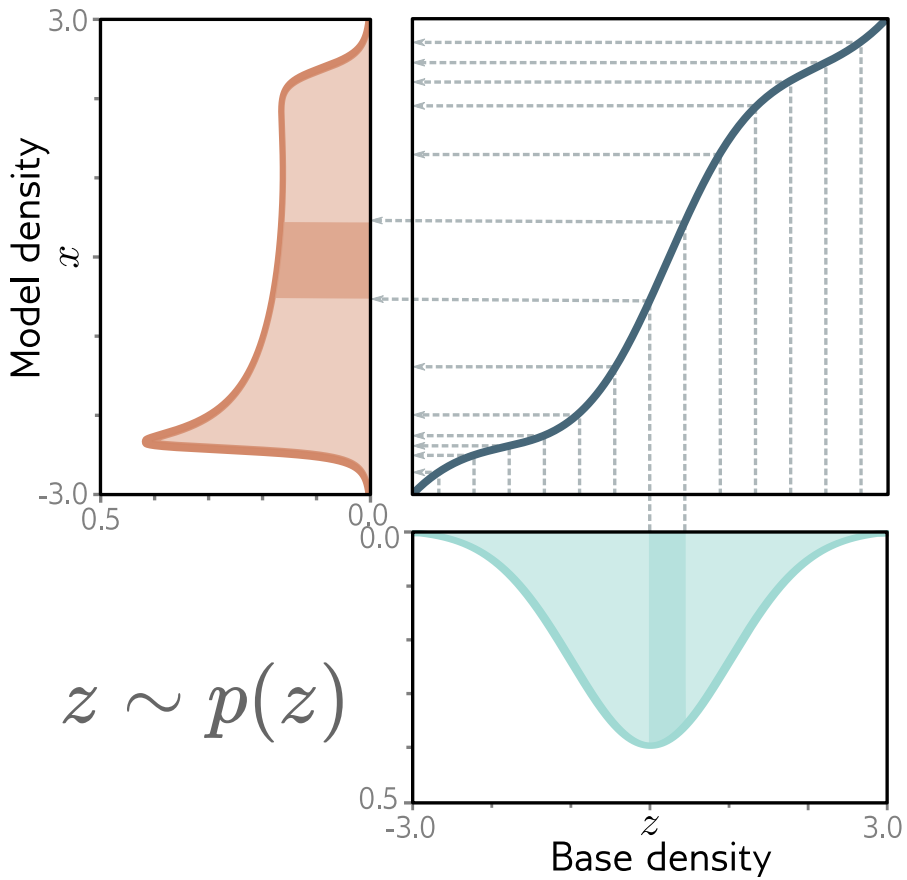
- $z \sim p_Z(z)$
- $p_Z(z)$
- $Z \sim \mathcal{N}(0,1) \rightarrow g(z) \rightarrow X$
- $p_X(x) = p_Z(z) \left| \frac{dz}{dx} \right|$

Normalizing flows

Bijective

$$x = f(z)$$

$$x \sim p(x)$$



$$z \sim p(z)$$

[Image Credit: "Understanding Deep Learning" Simon J.D. Prince]

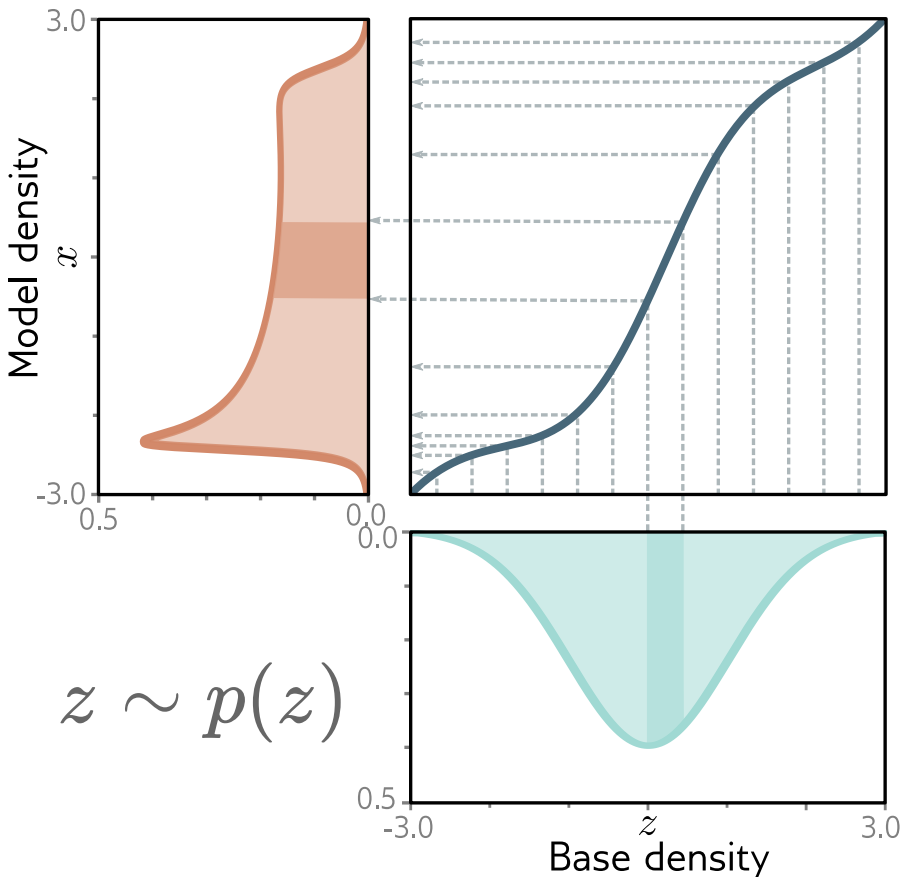
Normalizing flows

Bijective

$$x = f(z)$$

Sample

$$x \sim p(x)$$



$$z \sim p(z)$$

[Image Credit: "Understanding Deep Learning" Simon J.D. Prince]

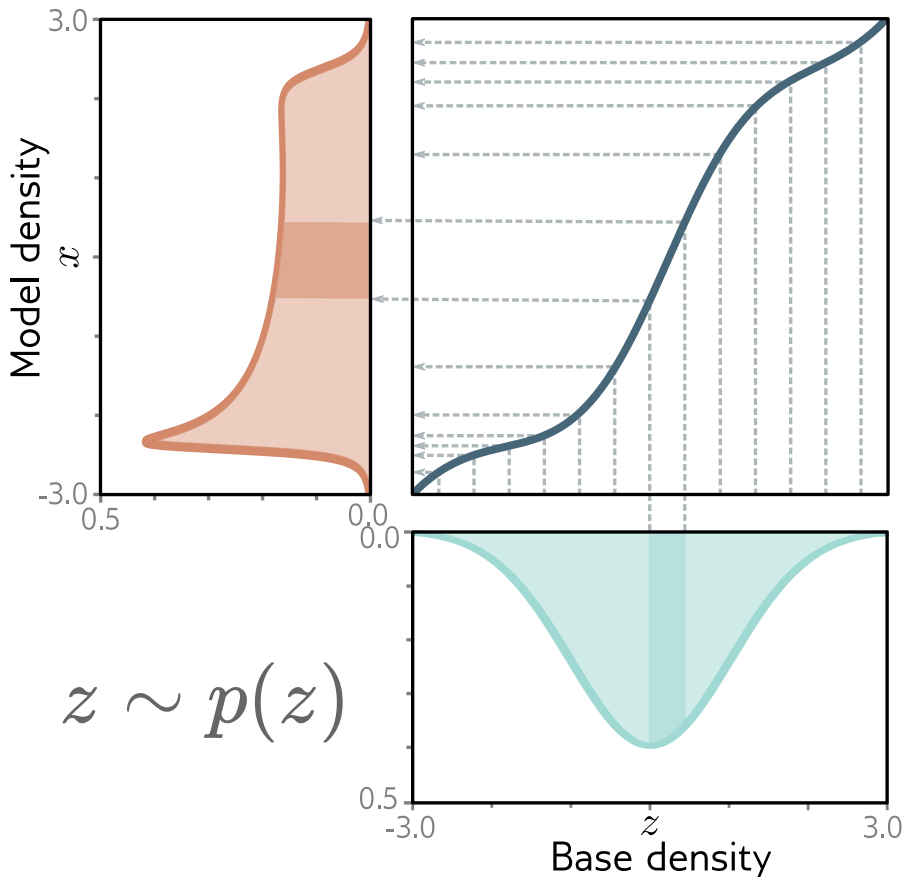
Normalizing flows

Bijective

$$x = f(z)$$

Sample

$$x \sim p(x)$$



$$z \sim p(z)$$

[Image Credit: "Understanding Deep Learning" Simon J.D. Prince]

Probability mass conserved locally

$$p(x) = p(z = f^{-1}(x)) \left| \det J_{f^{-1}}(x) \right|$$

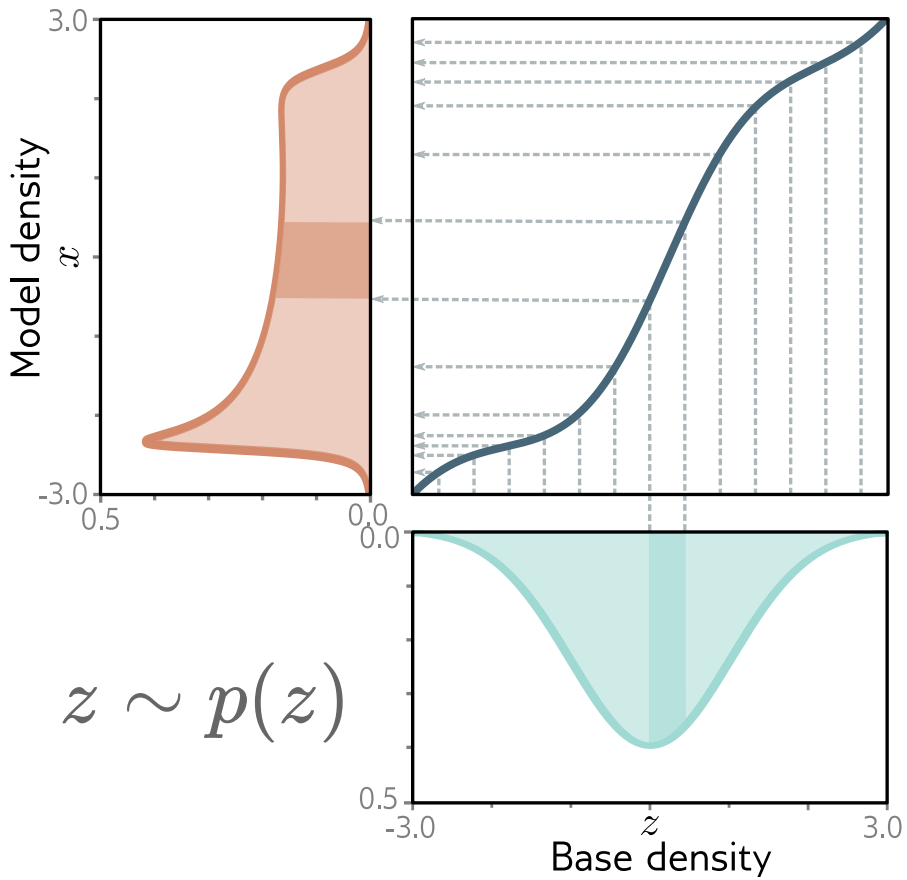
Normalizing flows

Bijective

$$x = f(z)$$

Sample

$$x \sim p(x)$$



$$z \sim p(z)$$

[Image Credit: "Understanding Deep Learning" Simon J.D. Prince]

Probability mass conserved locally

$$p(x) = p(z = f^{-1}(x)) \left| \det J_{f^{-1}}(x) \right|$$

Evaluate probabilities

$$z_0 \sim p(z) \qquad z_k = f_k(z_{k-1})$$

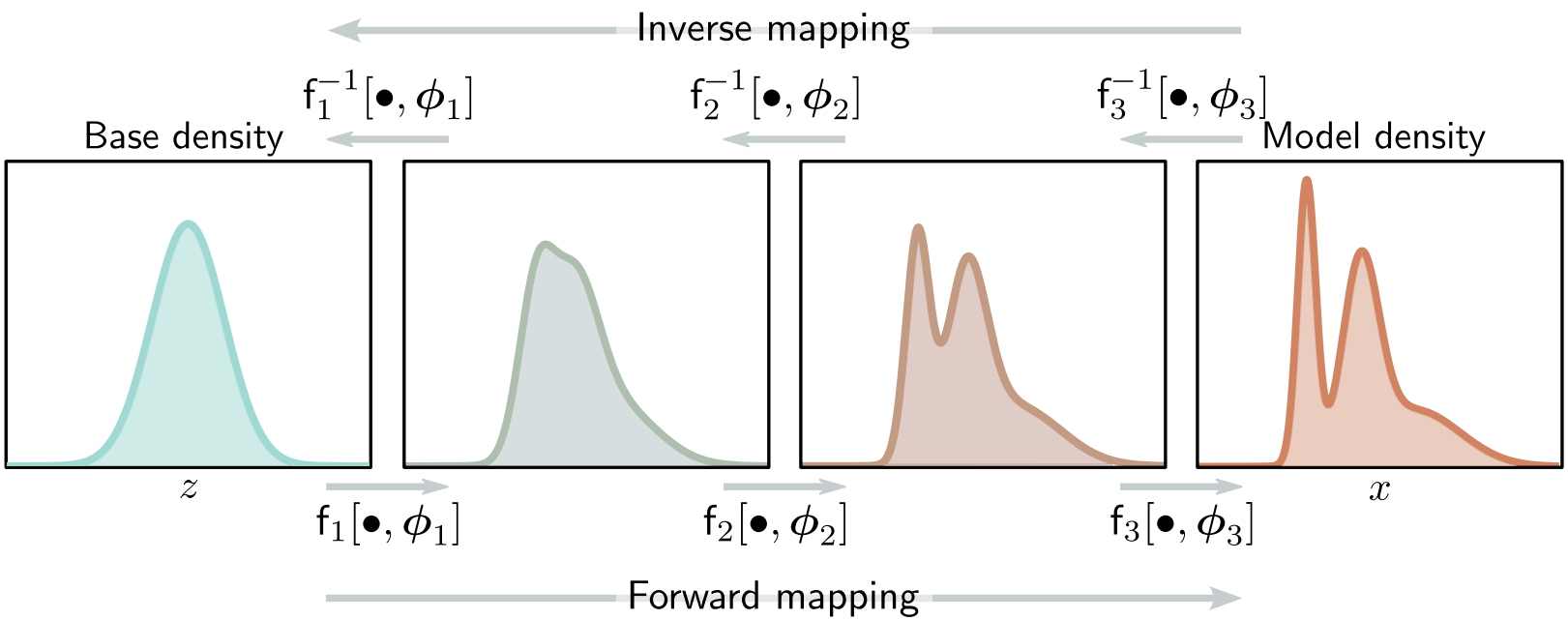
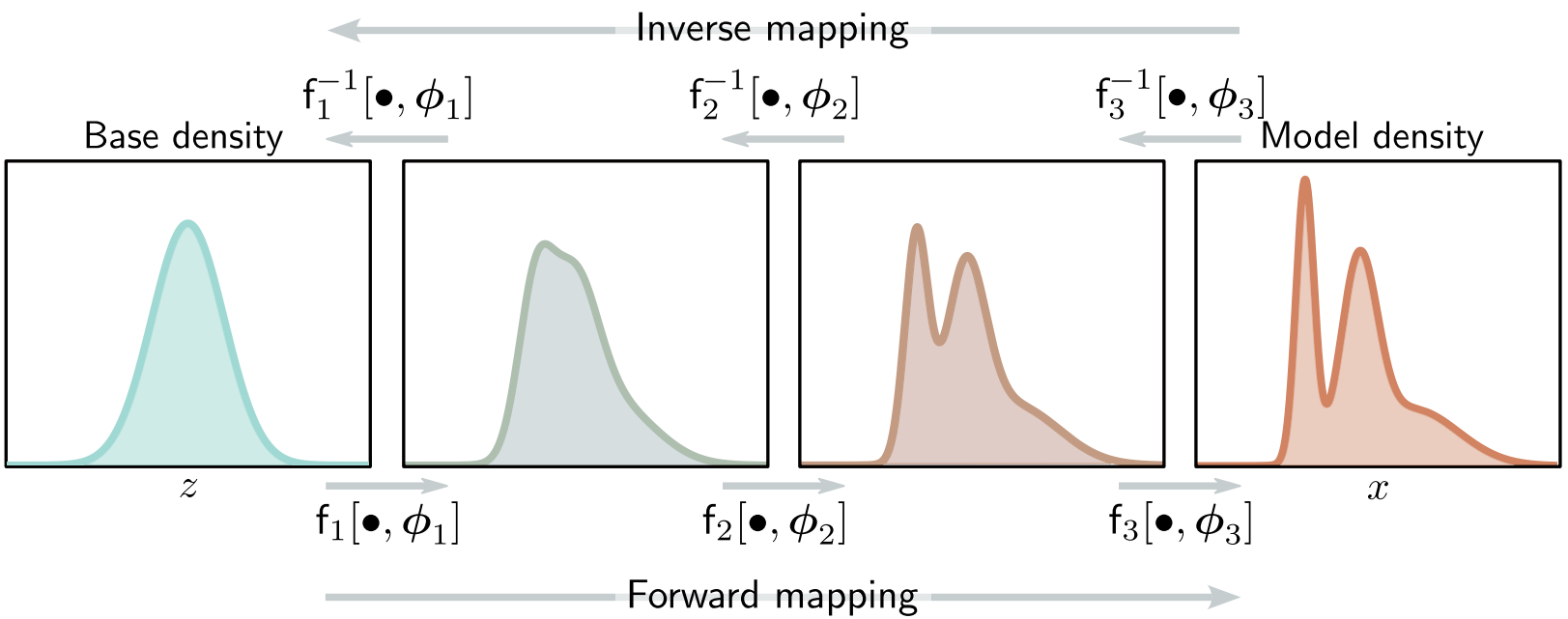


Image Credit: "Understanding Deep Learning" Simon J.D. Prince

$$z_0 \sim p(z) \qquad z_k = f_k(z_{k-1})$$

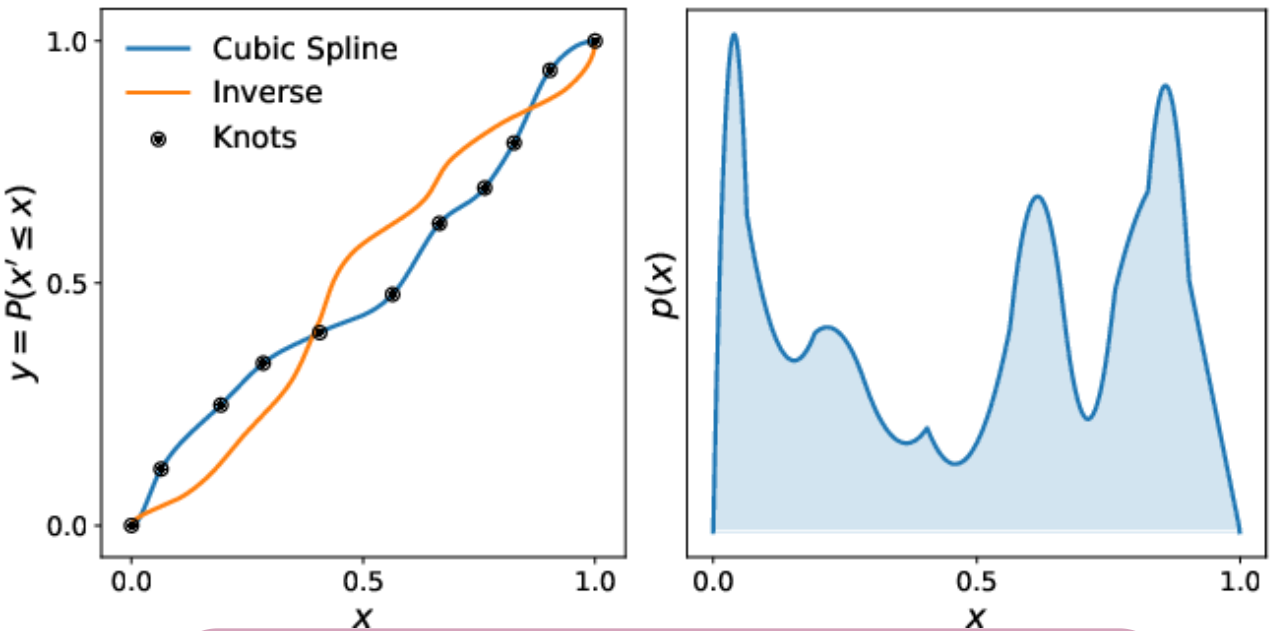


$$\log p(x) = \log p(z_0) - \sum_{k=1}^{K} \log \left| \det J_{f_k}(z_{k-1}) \right|$$

Image Credit: "Understanding Deep Learning" Simon J.D. Prince

Invertible functions aren't that common!

Splines



Issues NFs: **Lack of flexibility**

- Invertible functions
- Tractable Jacobians

Masked Autoregressive Flows

$$p(x) = \prod_i p(x_i \mid x_{1:i-1})$$

Masked Autoregressive Flows

$$p(x) = \prod_i p(x_i \mid x_{1:i-1})$$

$$p(x_i \mid x_{1:i-1}) = \mathcal{N}(x_i \mid \mu_i, (\exp \alpha_i)^2)$$

Masked Autoregressive Flows

$$p(x) = \prod_i p(x_i \mid x_{1:i-1})$$

Neural Network

$$\mu_i, \alpha_i = f_{\phi_i}(x_{1:i-1})$$

$$p(x_i \mid x_{1:i-1}) = \mathcal{N}(x_i \mid \mu_i, (\exp \alpha_i)^2)$$

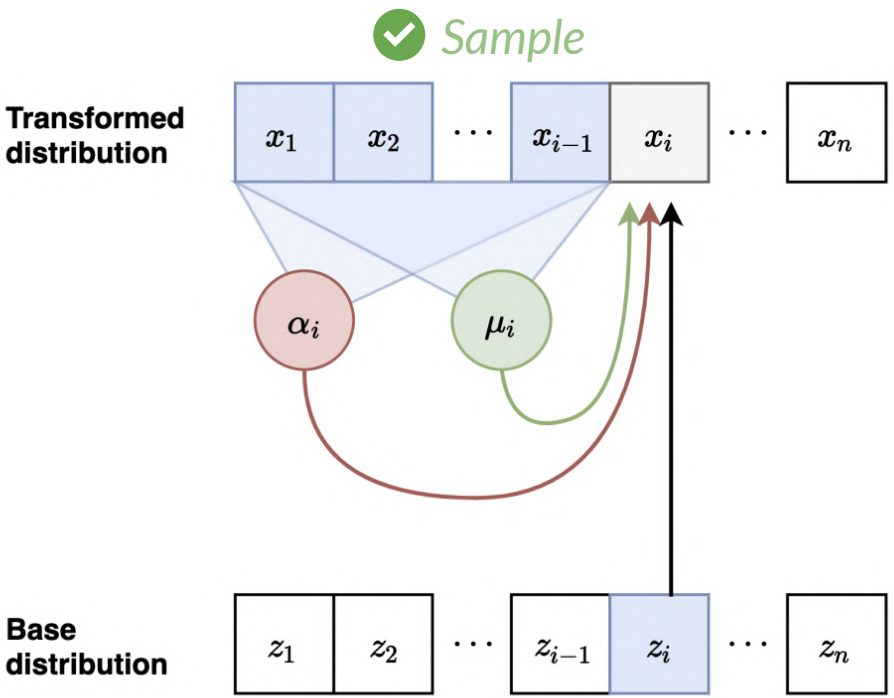
Masked Autoregressive Flows

$$p(x) = \prod_i p(x_i \mid x_{1:i-1})$$

Neural Network

$$\mu_i, \alpha_i = f_{\phi_i}(x_{1:i-1})$$

$$p(x_i \mid x_{1:i-1}) = \mathcal{N}(x_i \mid \mu_i, (\exp \alpha_i)^2)$$



$$x_i = z_i \exp(\alpha_i) + \mu_i$$

Masked Autoregressive Flows

$$p(x) = \prod_i p(x_i \mid x_{1:i-1})$$

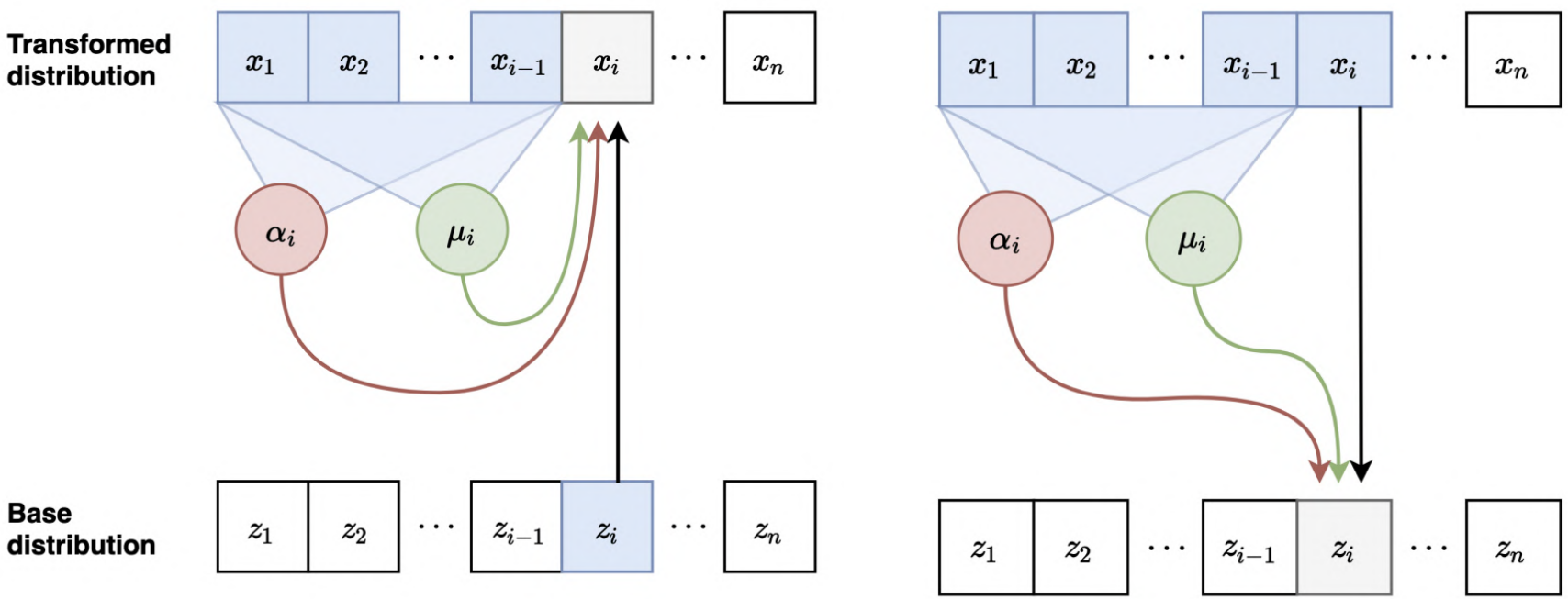
Neural Network

$$\mu_i, \alpha_i = f_{\phi_i}(x_{1:i-1})$$

$$p(x_i \mid x_{1:i-1}) = \mathcal{N}(x_i \mid \mu_i, (\exp \alpha_i)^2)$$

Sample

Evaluate probabilities

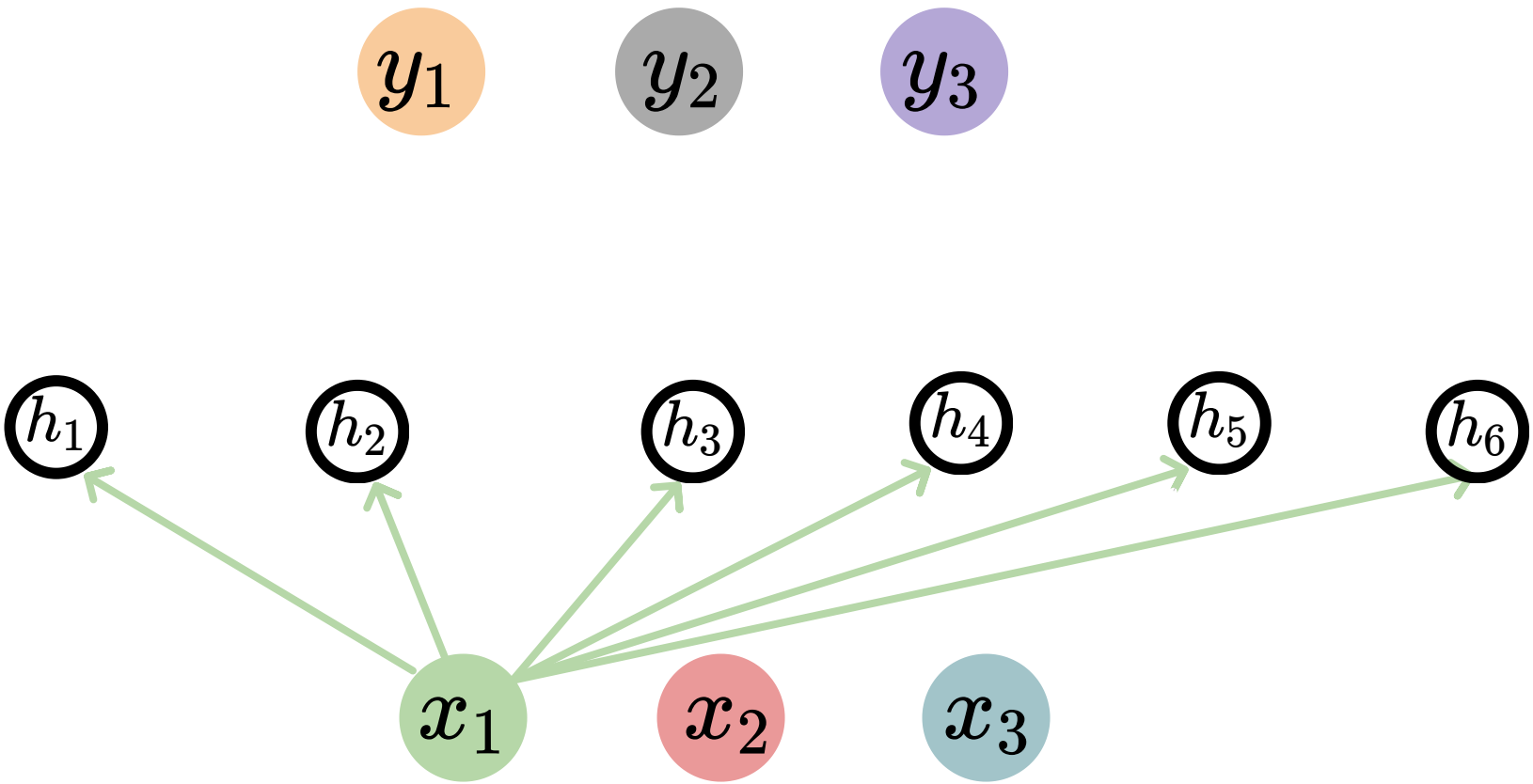


$$x_i = z_i \exp(\alpha_i) + \mu_i$$

$$z_i = (x_i - \mu_i) \exp(-\alpha_i)$$

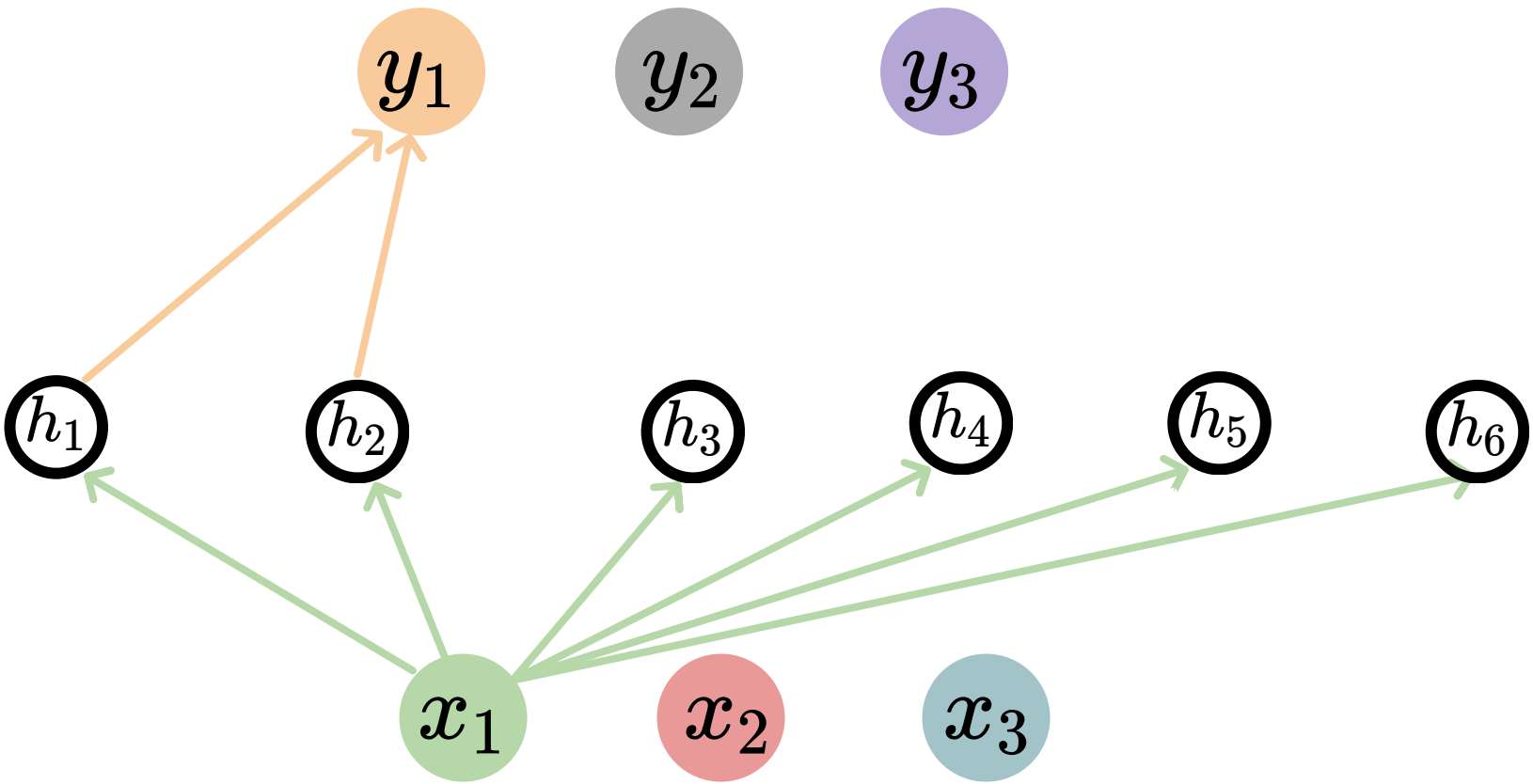
Masked MLP

$$y_i = \mu_i, \alpha_i = f_{\phi_i}(x_{1:i-1})$$



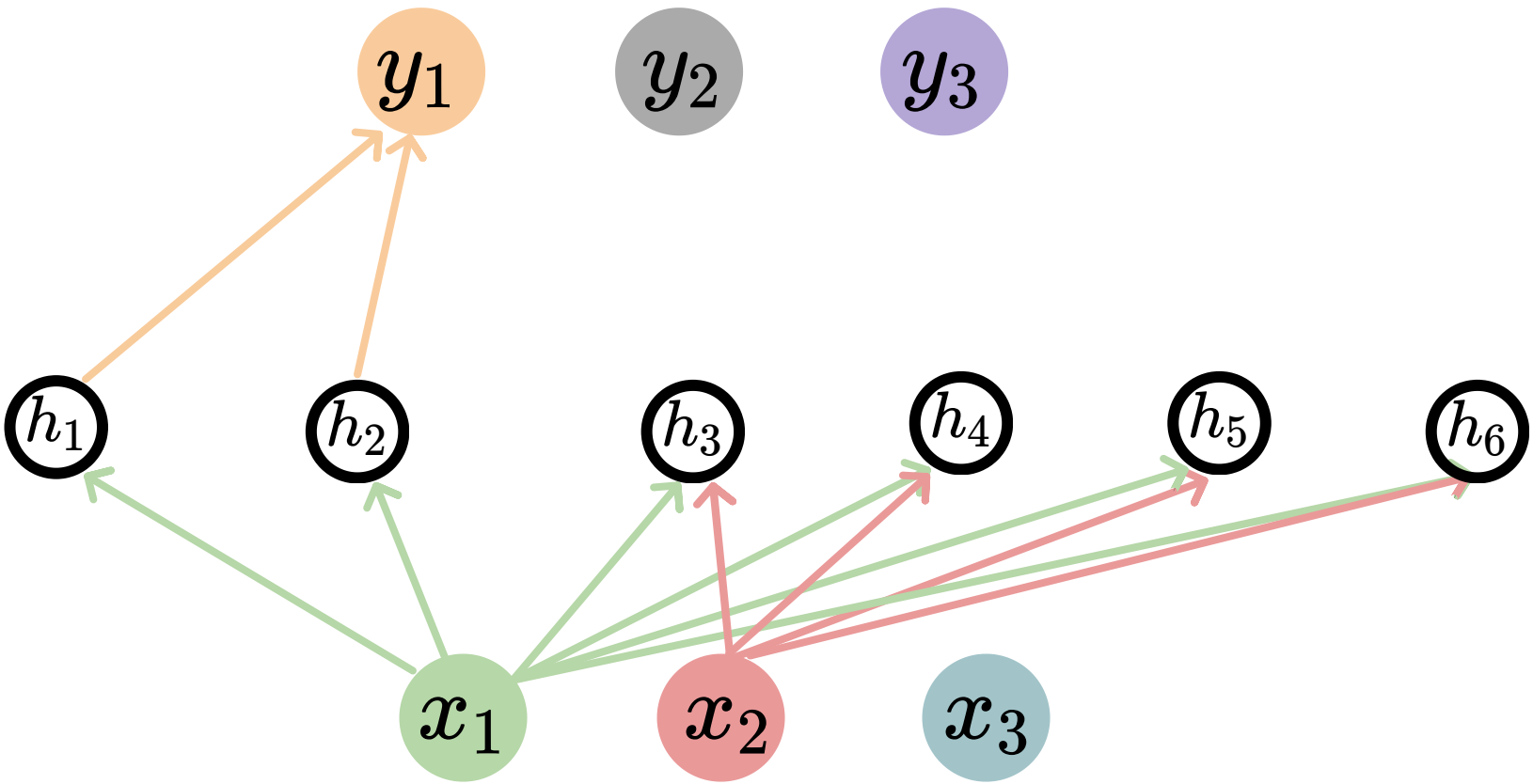
Masked MLP

$$y_i = \mu_i, \alpha_i = f_{\phi_i}(x_{1:i-1})$$



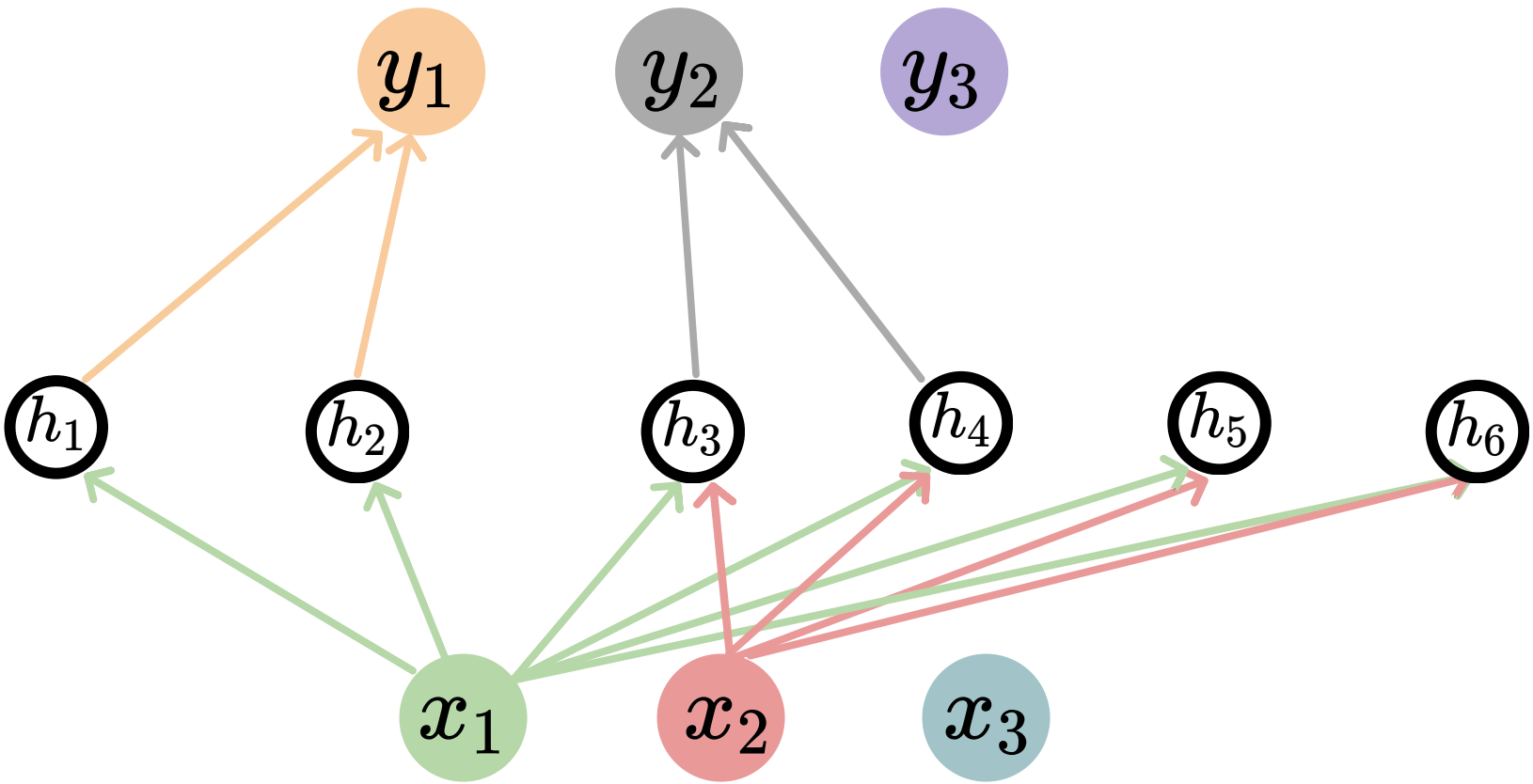
Masked MLP

$$y_i = \mu_i, \alpha_i = f_{\phi_i}(x_{1:i-1})$$



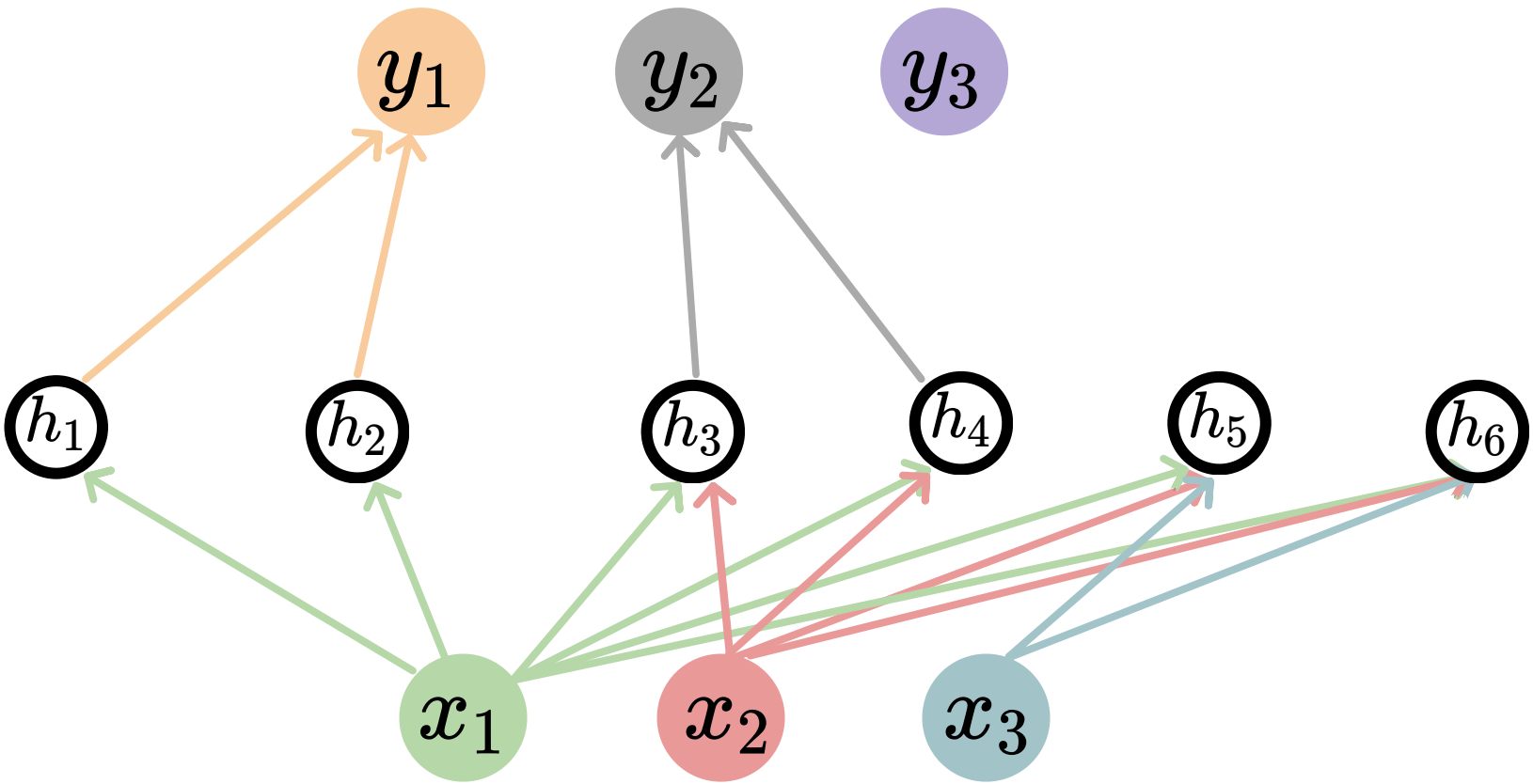
Masked MLP

$$y_i = \mu_i, \alpha_i = f_{\phi_i}(x_{1:i-1})$$



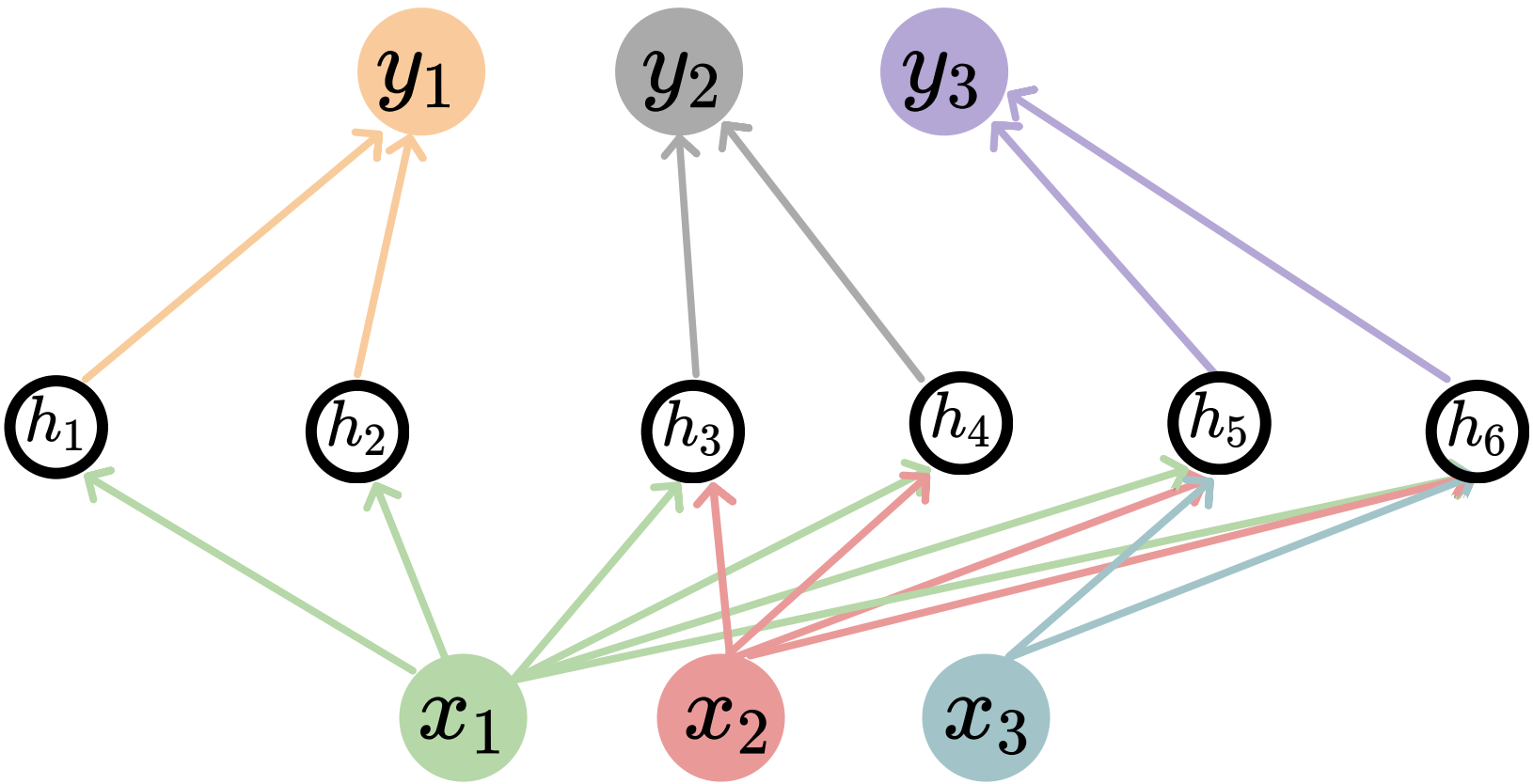
Masked MLP

$$y_i = \mu_i, \alpha_i = f_{\phi_i}(x_{1:i-1})$$



Masked MLP

$$y_i = \mu_i, \alpha_i = f_{\phi_i}(x_{1:i-1})$$



$$\log p(x) = \log p(z_0) - \sum_{k=1}^{K} \log \left| \det J_{f_k}(z_{k-1}) \right|$$

Computational Complexity

$$\mathcal{O}(N^3)$$

$$\log p(x) = \log p(z_0) - \sum_{k=1}^{K} \log \left| \det J_{f_k}(z_{k-1}) \right|$$

Computational Complexity

$$\mathcal{O}(N^3)$$

Autoregressive = Triangular matrix

$$\frac{\partial u_k^{(d)}}{\partial u_{k-1}^{(>d)}} == 0 \implies \begin{pmatrix} \frac{\partial u_k^{(1)}}{\partial u_{k-1}^{(1)}} & 0 & \cdots & 0 \\ \frac{\partial u_k^{(2)}}{\partial u_{k-1}^{(1)}} & \frac{\partial u_k^{(2)}}{\partial u_{k-1}^{(2)}} & \cdots & 0 \\ \vdots & \cdots & \cdots & \vdots \\ \frac{\partial u_k^{(D)}}{\partial u_{k-1}^{(1)}} & \cdots & \cdots & \frac{\partial u_k^{(D)}}{\partial u_{k-1}^{(D)}} \end{pmatrix}$$

$$\log p(x) = \log p(z_0) - \sum_{k=1}^{K} \log \left| \det J_{f_k}(z_{k-1}) \right|$$

Computational Complexity

$$\mathcal{O}(N^3)$$

Autoregressive = Triangular matrix

$\mathcal{O}(N)$

$$\frac{\partial u_k^{(d)}}{\partial u_{k-1}^{(>d)}} == 0 \implies \begin{pmatrix} \frac{\partial u_k^{(1)}}{\partial u_{k-1}^{(1)}} & 0 & \cdots & 0 \\ \frac{\partial u_k^{(2)}}{\partial u_{k-1}^{(1)}} & \frac{\partial u_k^{(2)}}{\partial u_{k-1}^{(2)}} & \cdots & 0 \\ \vdots & \cdots & \cdots & \vdots \\ \frac{\partial u_k^{(D)}}{\partial u_{k-1}^{(1)}} & \cdots & \cdots & \frac{\partial u_k^{(D)}}{\partial u_{k-1}^{(D)}} \end{pmatrix}$$

Simulation-based Inference

Parameters

θ

Predict

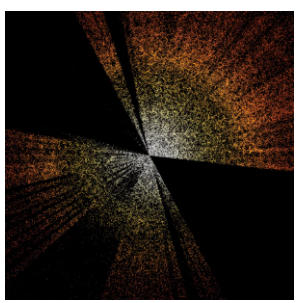
Forward Model

Observable

x

$\Omega_m, w_0, w_a, f_{\rm NL} \cdots$

Dark matter | Dark energy | Inflation



Simulation-based Inference

Parameters

$p(\theta|x)$

Predict

Forward Model

Inverse mapping

Infer

Observable

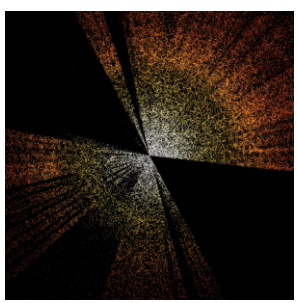
x

$\Omega_m, w_0, w_a, f_{\rm NL} \cdots$

Dark matter

Dark energy

Inflation



Simulation-based Inference

Parameters

$p(\theta|x)$

Normalizing flow

Predict

Forward Model

Inverse mapping

Infer

Observable

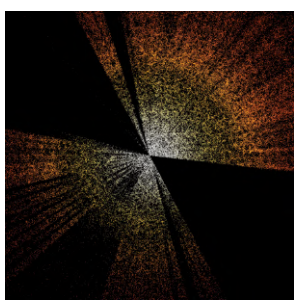
x

$\Omega_m, w_0, w_a, f_{\mathrm{NL}} \cdots$

Dark matter

Dark energy

Inflation



In continuous time

$$\frac{dx_t}{dt} = v_t^{\phi}(x_t)$$

[Image Credit: "Understanding Deep Learning" Simon J.D. Prince]

In continuous time

$$\frac{dx_t}{dt} = v_t^{\phi}(x_t)$$

$$x_1 = x_0 + \int_0^1 v_t^{\phi}(x_t)dt$$

$$x_0 = x_1 + \int_1^0 v_t^{\phi}(x_t)dt$$

[Image Credit: "Understanding Deep Learning" Simon J.D. Prince]

In continuous time

$$\frac{dx_t}{dt} = v_t^{\phi}(x_t)$$

$$x_1 = x_0 + \int_0^1 v_t^{\phi}(x_t)dt$$

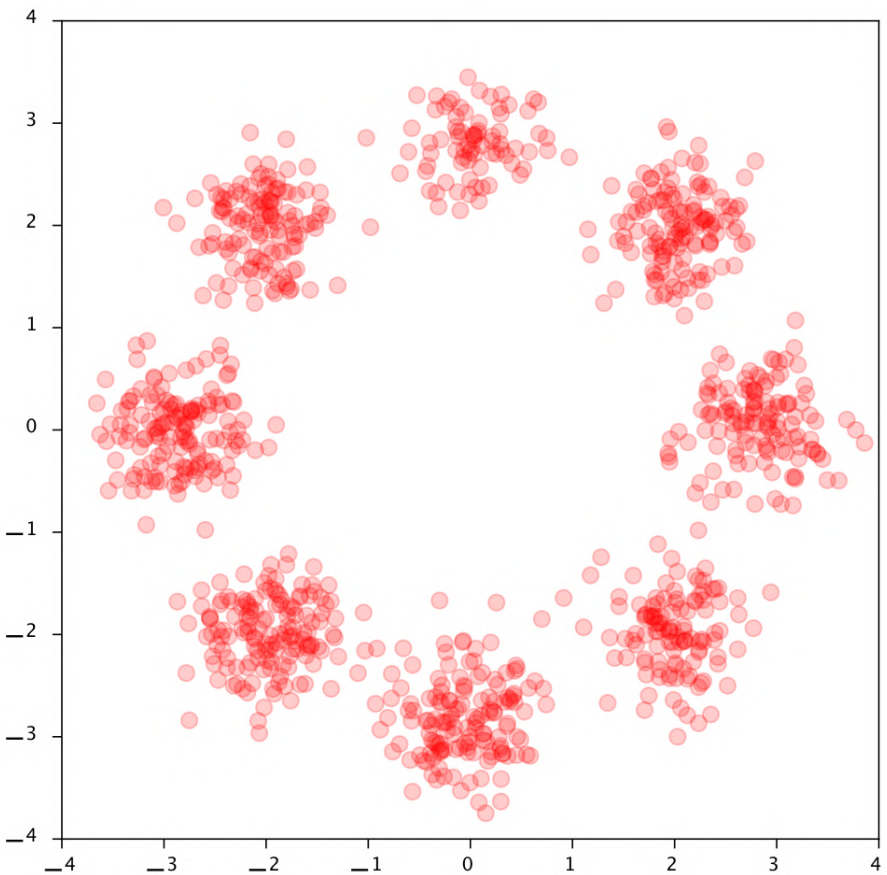
$$x_0 = x_1 + \int_1^0 v_t^{\phi}(x_t)dt$$

$$\frac{dp(x_t)}{dt} = -\nabla\left(v_t^{\phi}(x_t)p(x_t)\right)$$

Continuity Equation

[Image Credit: "Understanding Deep Learning" Simon J.D. Prince]

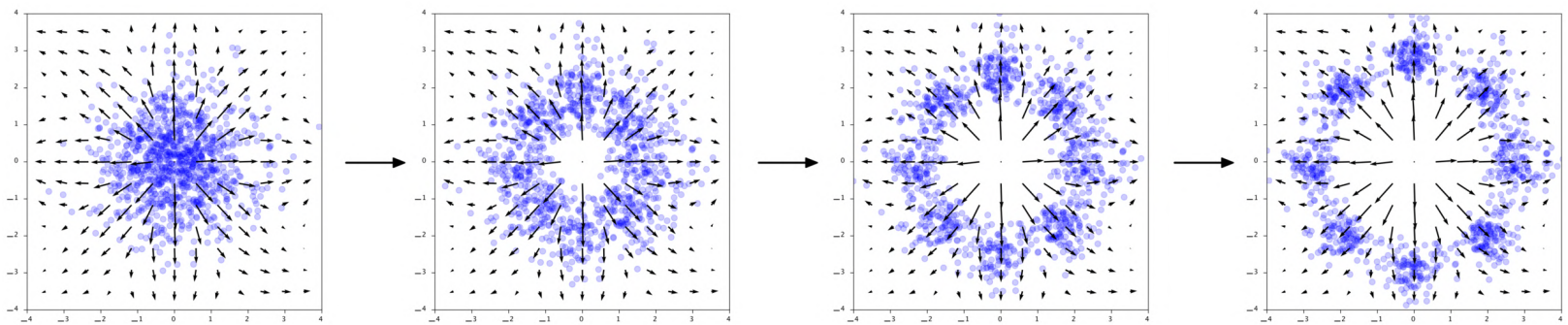
$$x_1 = x_0 + \int_0^1 v_\theta(x(t), t)dt$$



Chen et al. (2018), Grathwohl et al. (2018)

$$x_1 = x_0 + \int_0^1 v_\theta(x(t), t)dt$$

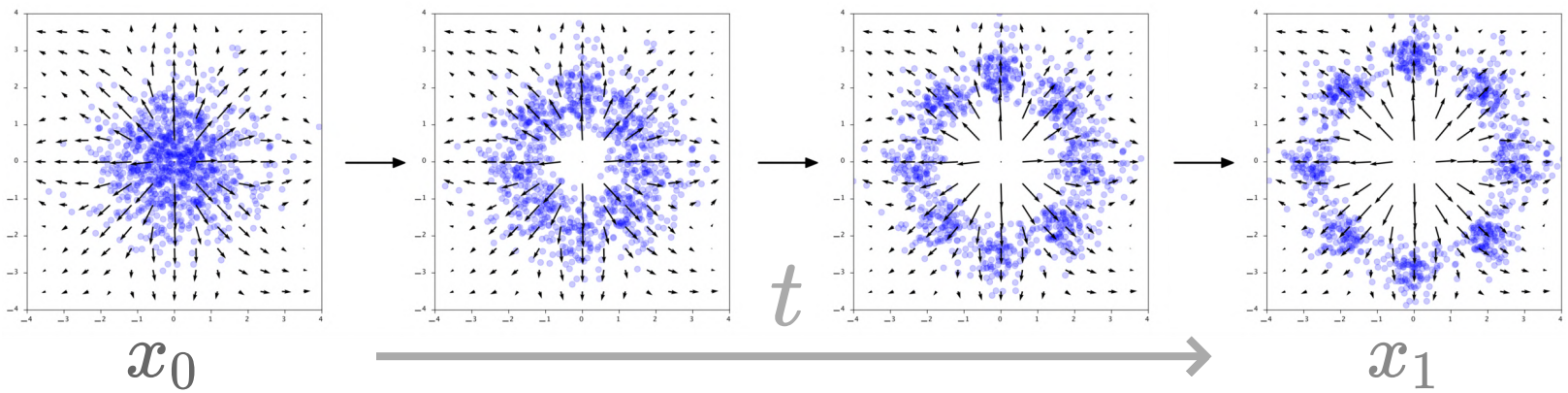
Generate



Chen et al. (2018), Grathwohl et al. (2018)

$$x_1 = x_0 + \int_0^1 v_\theta(x(t), t)dt$$

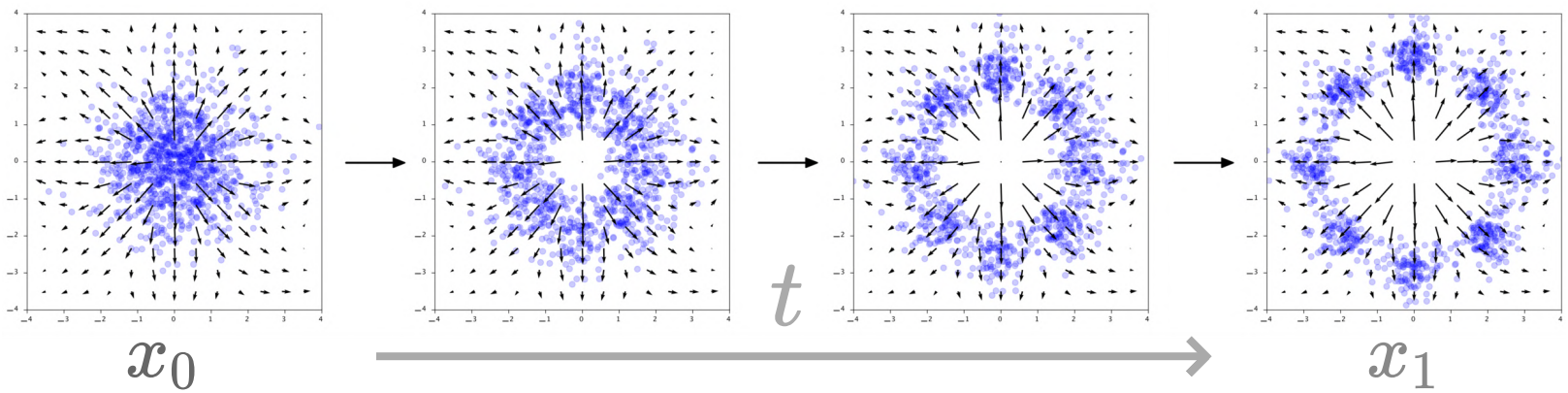
Generate



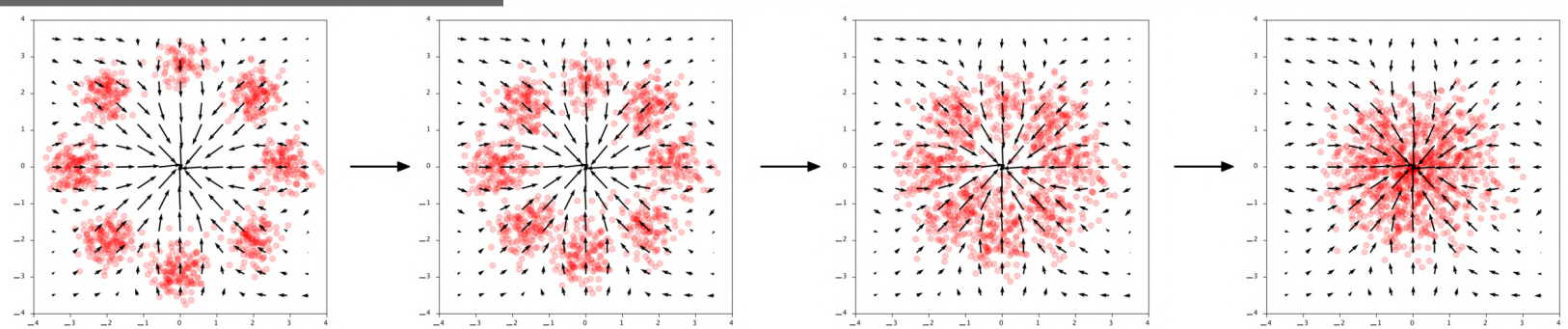
Chen et al. (2018), Grathwohl et al. (2018)

$$x_1 = x_0 + \int_0^1 v_\theta(x(t), t)dt$$

Generate

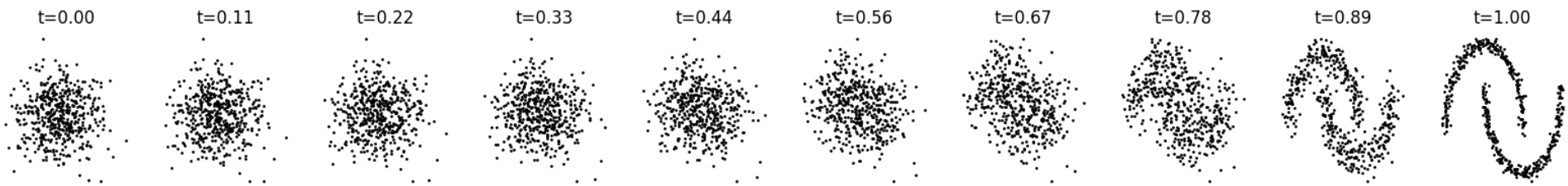


Evaluate Probability



Chen et al. (2018), Grathwohl et al. (2018)

Tutorial 2: Continuous time normalizing flows



x_0

Gaussian

x_1

Two Moons

$$\log p_X(x) = \log p_Z(z) + \int_0^1 \mathrm{Tr} J_v(x(t)) dt$$

$$\log p_X(x) = \log p_Z(z) + \int_0^1 \mathrm{Tr} J_v(x(t))dt$$

Loss requires solving an ODE!

$$\log p_X(x) = \log p_Z(z) + \int_0^1 \mathrm{Tr} J_v(x(t))dt$$

Loss requires solving an ODE!

Diffusion, Flow matching, Interpolants...
All ways to avoid this ***at training time***

Conditional Flow matching

["Flow Matching for Generative Modeling" Lipman et al]

["Stochastic Interpolants: A Unifying framework for Flows and Diffusions" Albergo et al]

Conditional Flow matching

Assume a conditional vector field (known at training time)

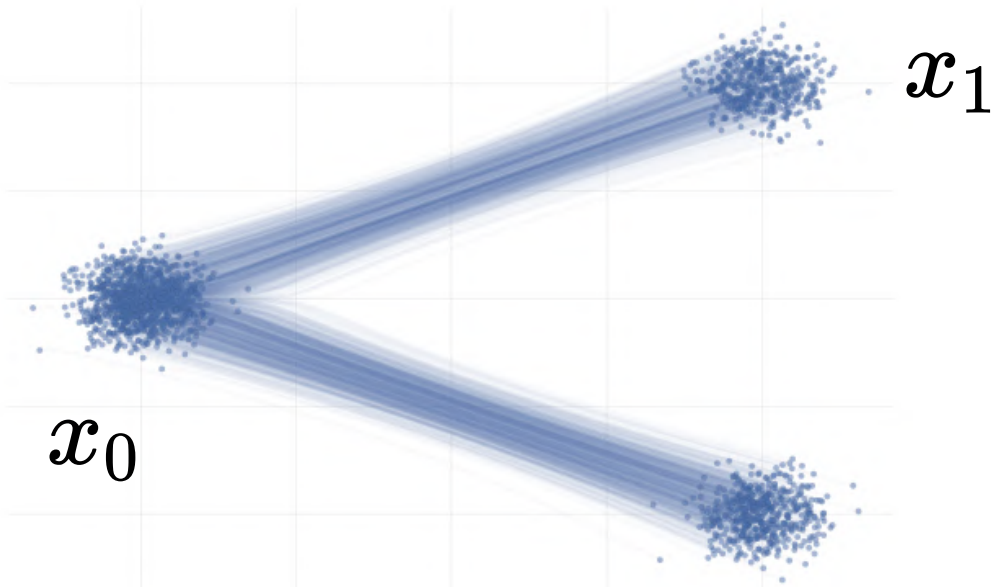
$$x_t = (1-t)x_0 + tx_1$$

["Flow Matching for Generative Modeling" Lipman et al]
["Stochastic Interpolants: A Unifying framework for Flows and Diffusions" Albergo et al]

Conditional Flow matching

Assume a conditional vector field (known at training time)

$$x_t = (1-t)x_0 + tx_1$$



["Flow Matching for Generative Modeling" Lipman et al]
["Stochastic Interpolants: A Unifying framework for Flows and Diffusions" Albergo et al]

Conditional Flow matching

Assume a conditional vector field (known at training time)

$$x_t = (1-t)x_0 + tx_1$$

$$p_t(x) = \int p_t(x|x_1)q(x_1)\,dx_1$$

$$u_t^{\text{target}}(x) = \int u_t(x|x_1)\frac{p_t(x|x_1)p_1(x_1)}{p_t(x)}\,dx_1$$

["Flow Matching for Generative Modeling" Lipman et al]

["Stochastic Interpolants: A Unifying framework for Flows and Diffusions" Albergo et al]

Conditional Flow matching

Assume a conditional vector field (known at training time)

$$p_t(x) = \int p_t(x|x_1) q(x_1)\, dx_1$$

$$x_t = (1-t)x_0 + t x_1$$

$$u_t^{\text{target}}(x) = \int u_t(x|x_1) \frac{p_t(x|x_1) p_1(x_1)}{p_t(x)}\, dx_1$$

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{t\sim U, x\sim p_t}\left[\| u_t^{\phi}(x) - u_t^{\text{target}}(x) \|^2 \right]$$

["Flow Matching for Generative Modeling" Lipman et al]

["Stochastic Interpolants: A Unifying framework for Flows and Diffusions" Albergo et al]

Conditional Flow matching

Assume a conditional vector field (known at training time)

$$x_t = (1-t)x_0 + tx_1$$

Intractable

$$p_t(x) = \int p_t(x|x_1)q(x_1)\,dx_1$$

$$u_t^{\text{target}}(x) = \int u_t(x|x_1)\frac{p_t(x|x_1)p_1(x_1)}{p_t(x)}\,dx_1$$

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{t\sim U, x\sim p_t}\left[\|u_t^{\phi}(x) - u_t^{\text{target}}(x)\|^2\right]$$

["Flow Matching for Generative Modeling" Lipman et al]

["Stochastic Interpolants: A Unifying framework for Flows and Diffusions" Albergo et al]

Conditional Flow matching

Assume a conditional vector field (known at training time)

$$x_t = (1-t)x_0 + tx_1$$

Intractable

$$p_t(x) = \int p_t(x|x_1)q(x_1)\,dx_1$$

$$u_t^{\text{target}}(x) = \int u_t(x|x_1)\frac{p_t(x|x_1)p_1(x_1)}{p_t(x)}\,dx_1$$

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{t\sim U, x\sim p_t}\left[\|u_t^{\phi}(x) - u_t^{\text{target}}(x)\|^2\right]$$

The loss that we can compute

$$\mathcal{L}_{\text{CFM}} = \mathbb{E}_{t\sim U, x_1\sim p_1, x_t\sim p_t}\left[\|u_t^{\phi}(x_t) - u_t^{\text{target}}(x|x_1)\|^2\right]$$

["Flow Matching for Generative Modeling" Lipman et al]

["Stochastic Interpolants: A Unifying framework for Flows and Diffusions" Albergo et al]

Conditional Flow matching

Assume a conditional vector field (known at training time)

$$x_t = (1-t)x_0 + t x_1$$

Intractable

$$p_t(x) = \int p_t(x|x_1) q(x_1)\, dx_1$$

$$u_t^{\text{target}}(x) = \int u_t(x|x_1) \frac{p_t(x|x_1) p_1(x_1)}{p_t(x)}\, dx_1$$

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{t\sim U, x\sim p_t}\left[\|u_t^{\phi}(x) - u_t^{\text{target}}(x)\|^2\right]$$

The loss that we can compute

$$\mathcal{L}_{\text{CFM}} = \mathbb{E}_{t\sim U, x_1\sim p_1, x_t\sim p_t}\left[\|u_t^{\phi}(x_t) - u_t^{\text{target}}(x|x_1)\|^2\right]$$

The gradients of the losses are the same!

$$\nabla_\phi \mathcal{L}_{\text{CFM}} = \nabla_\phi \mathcal{L}_{\text{FM}}$$

["Flow Matching for Generative Modeling" Lipman et al]

["Stochastic Interpolants: A Unifying framework for Flows and Diffusions" Albergo et al]

Flow Matching

$$\frac{dz_t}{dt} = u_t^{\phi}(z_t)$$

[Image Credit: "Understanding Deep Learning" Simon J.D. Prince]

Flow Matching

$$\frac{dz_t}{dt} = u_t^{\phi}(z_t)$$

$$x = z_0 + \int_0^1 u_t^{\phi}(z_t)dt$$

[Image Credit: "Understanding Deep Learning" Simon J.D. Prince]

Flow Matching

$$\frac{dz_t}{dt} = u_t^{\phi}(z_t)$$

$$x = z_0 + \int_0^1 u_t^{\phi}(z_t)dt$$

$$\frac{dp(z_t)}{dt} = -\nabla\left(u_t^{\phi}(z_t)p(z_t)\right)$$

Continuity equation

[Image Credit: "Understanding Deep Learning" Simon J.D. Prince]

Flow Matching

$$\frac{dz_t}{dt} = u_t^{\phi}(z_t)$$

✅ *Sample*

$$x = z_0 + \int_0^1 u_t^{\phi}(z_t)dt$$

✅ *Evaluate probabilities*

$$\frac{dp(z_t)}{dt} = -\nabla\left(u_t^{\phi}(z_t)p(z_t)\right)$$

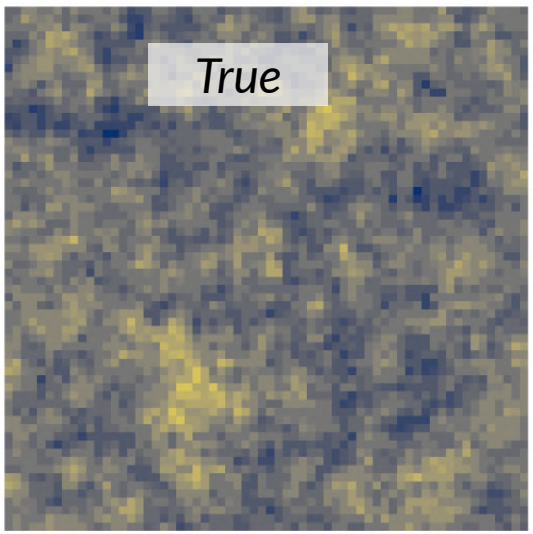
Continuity equation

[Image Credit: "Understanding Deep Learning" Simon J.D. Prince]

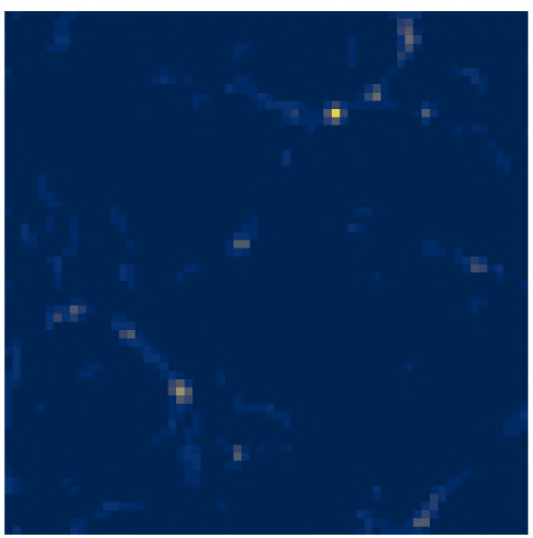
Stochastic Interpolants

$$p(\delta_{\rm ICs}, \theta | \delta_{\rm Obs})$$

$\delta_{\rm ICs}$



$\delta_{\rm Obs}$



"Joint cosmological parameter inference and initial condition reconstruction with Stochastic Interpolants"

Cuesta-Lazaro, Bayer, Albergo et al

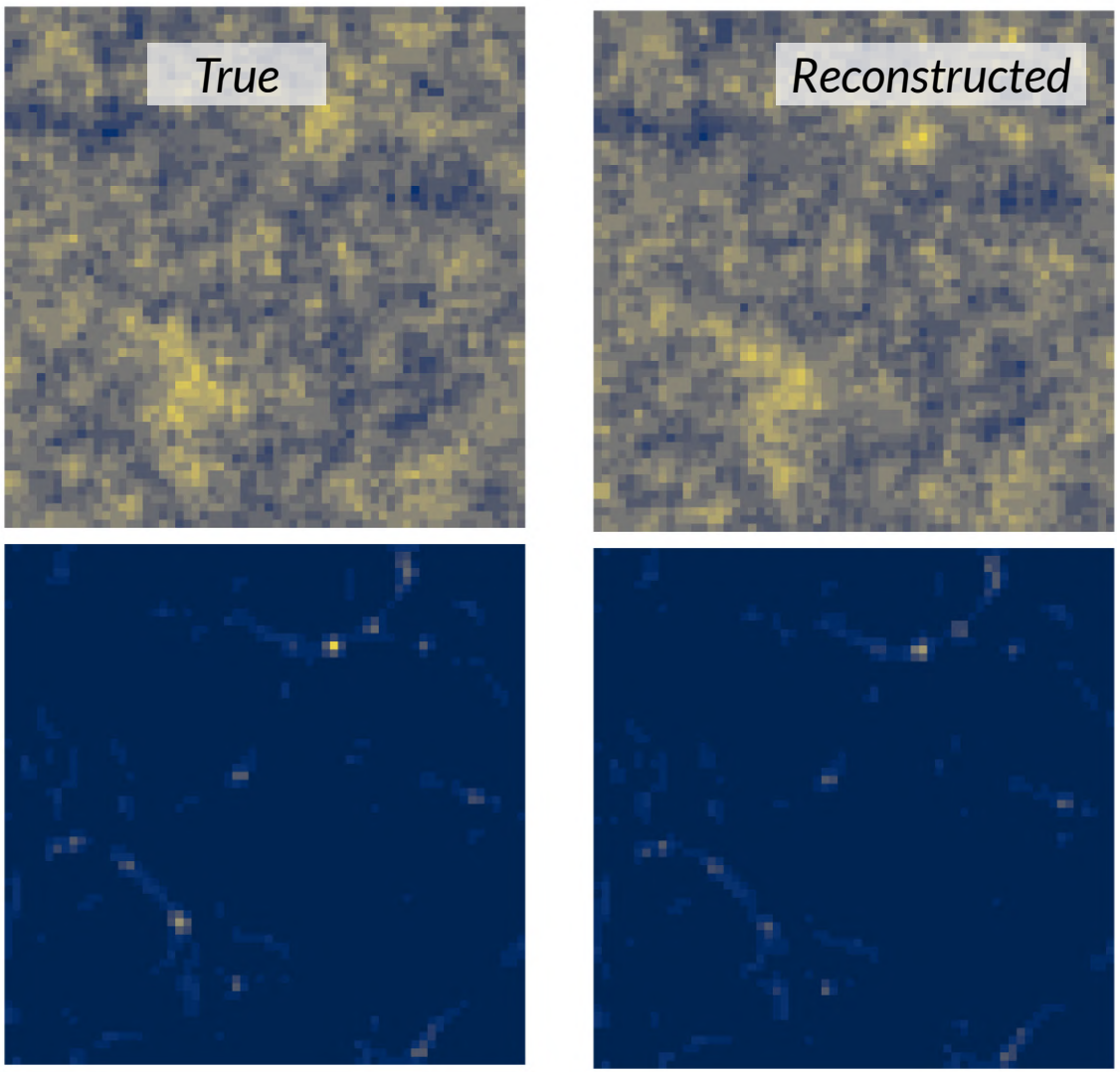
NeurIPs ML4PS 2024 Spotlight talk

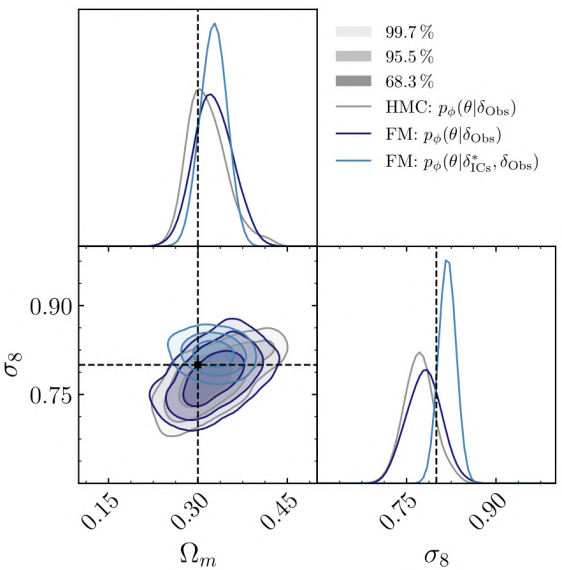
Stochastic Interpolants

$$p(\delta_{\mathrm{ICs}}, \theta | \delta_{\mathrm{Obs}})$$

δ_{ICs}

δ_{Obs}





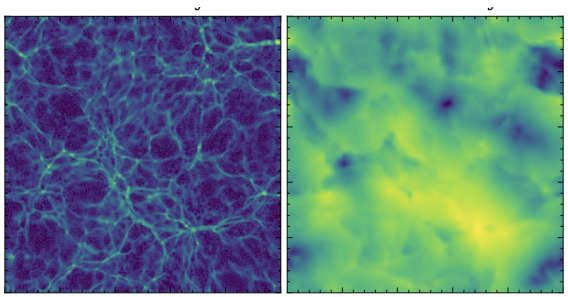
"Joint cosmological parameter inference and initial condition reconstruction with Stochastic Interpolants"

Cuesta-Lazaro, Bayer, Albergo et al

NeurIPs ML4PS 2024 Spotlight talk

Hydro Simulations at scale

Particle Mesh for Gravity



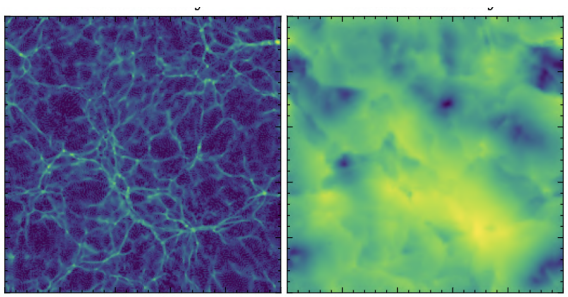
CAMELS Volumes

$25h^{-1}\mathrm{Mpc}$

["BaryonBridge: Interpolants models for fast hydrodynamical simulations" Horowitz, Cuesta-Lazaro, Yehia ML4Astro workshop 2025]

Hydro Simulations at scale

Particle Mesh for Gravity



CAMELS Volumes

$25h^{-1}\mathrm{Mpc}$

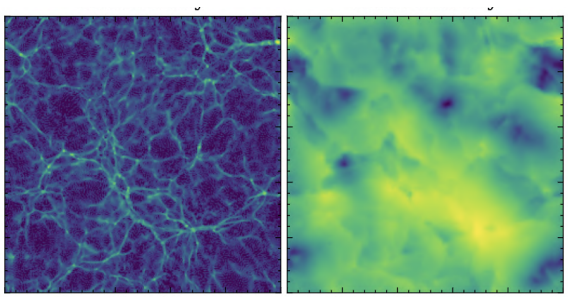
$+\mathcal{C}, \mathcal{A}$

1000 boxes with varying
cosmology and ***feedback*** models

["BaryonBridge: Interpolants models for fast hydrodynamical simulations" Horowitz, Cuesta-Lazaro, Yehia ML4Astro workshop 2025]

Hydro Simulations at scale

Particle Mesh for Gravity



CAMELS Volumes

$25h^{-1}\mathrm{Mpc}$

$$p(\mathrm{Baryons}|\mathrm{DM}, \mathcal{C}, \mathcal{A})$$

$$\longrightarrow$$

$$+\mathcal{C}, \mathcal{A}$$

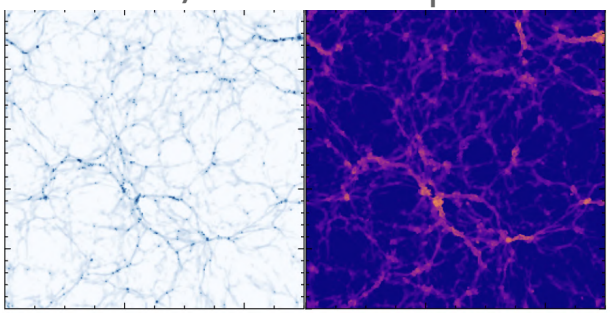
1000 boxes with varying
cosmology and ***feedback*** models

["BaryonBridge: Interpolants models for fast hydrodynamical simulations" Horowitz, Cuesta-Lazaro, Yehia ML4Astro workshop 2025]

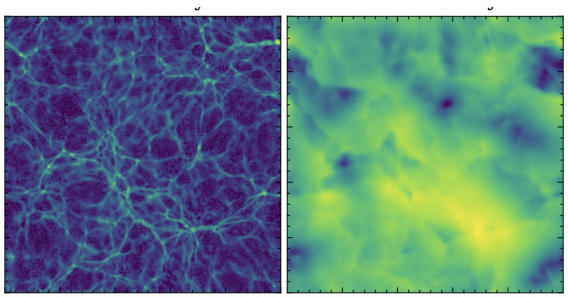
Hydro Simulations at scale

Gas Properties

Density Temperature



Particle Mesh for Gravity



CAMELS Volumes

$25h^{-1}\mathrm{Mpc}$

$$p(\mathrm{Baryons}|\mathrm{DM}, \mathcal{C}, \mathcal{A})$$

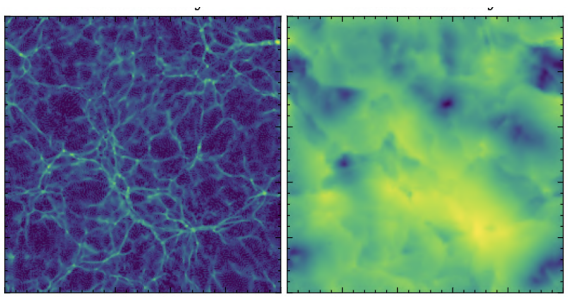
$$+\mathcal{C}, \mathcal{A}$$

1000 boxes with varying
cosmology and ***feedback*** models

["BaryonBridge: Interpolants models for fast hydrodynamical simulations" Horowitz, Cuesta-Lazaro, Yehia ML4Astro workshop 2025]

Hydro Simulations at scale

Particle Mesh for Gravity



CAMELS Volumes

$25h^{-1}\mathrm{Mpc}$

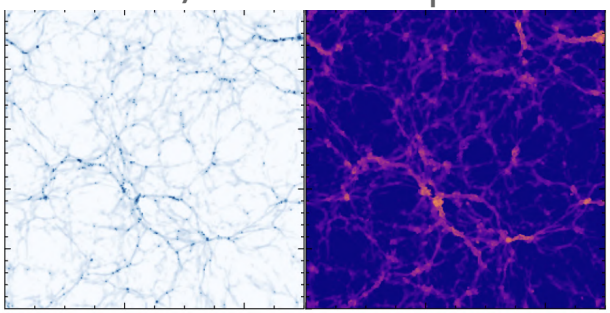
$$p(\mathrm{Baryons}|\mathrm{DM}, \mathcal{C}, \mathcal{A})$$

$$+\mathcal{C}, \mathcal{A}$$

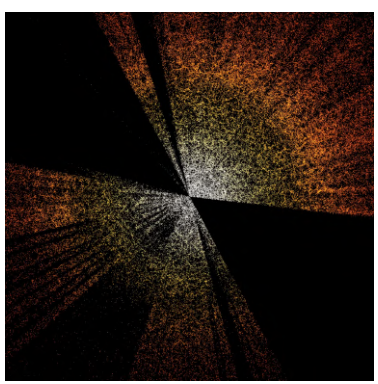
1000 boxes with varying
cosmology and ***feedback*** models

Gas Properties

Density Temperature



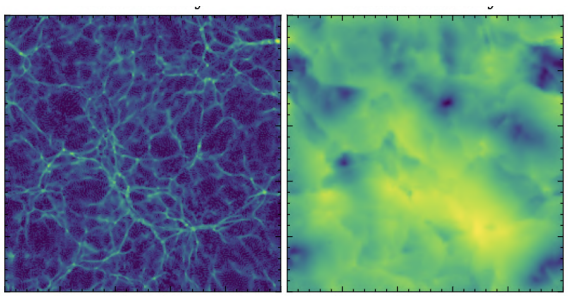
Galaxy Distribution



["BaryonBridge: Interpolants models for fast hydrodynamical simulations" Horowitz, Cuesta-Lazaro, Yehia ML4Astro workshop 2025]

Hydro Simulations at scale

Particle Mesh for Gravity



CAMELS Volumes

$25h^{-1}\mathrm{Mpc}$

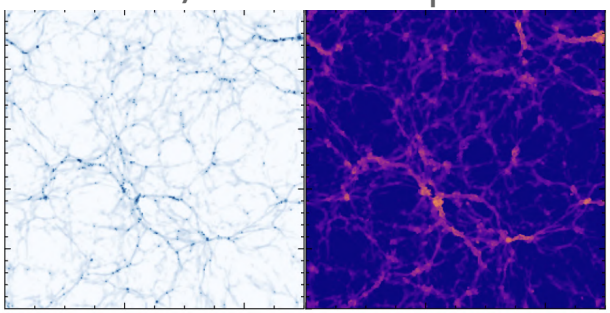
$p(\mathrm{Baryons}|\mathrm{DM}, \mathcal{C}, \mathcal{A})$

$+\mathcal{C}, \mathcal{A}$

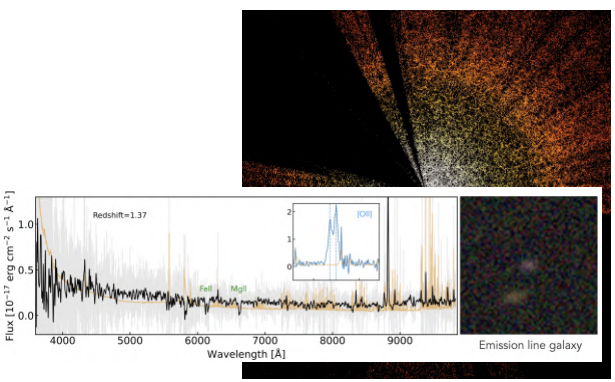
1000 boxes with varying
cosmology and ***feedback*** models

Gas Properties

Density Temperature



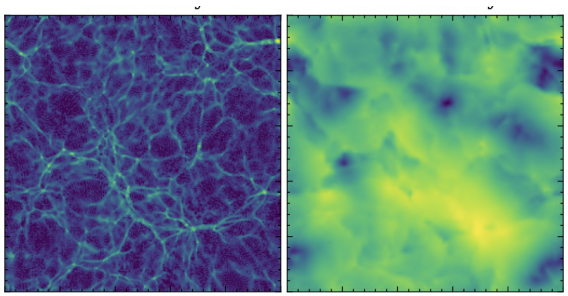
Galaxy Distribution



["BaryonBridge: Interpolants models for fast hydrodynamical simulations" Horowitz, Cuesta-Lazaro, Yehia ML4Astro workshop 2025]

Hydro Simulations at scale

Particle Mesh for Gravity



CAMELS Volumes

$25h^{-1}\mathrm{Mpc}$

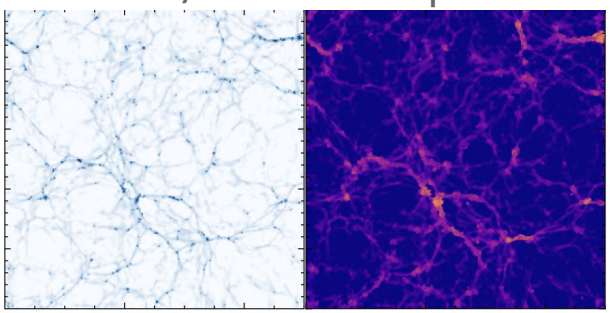
$$p(\mathrm{Baryons}|\mathrm{DM}, \mathcal{C}, \mathcal{A})$$

$$+\mathcal{C}, \mathcal{A}$$

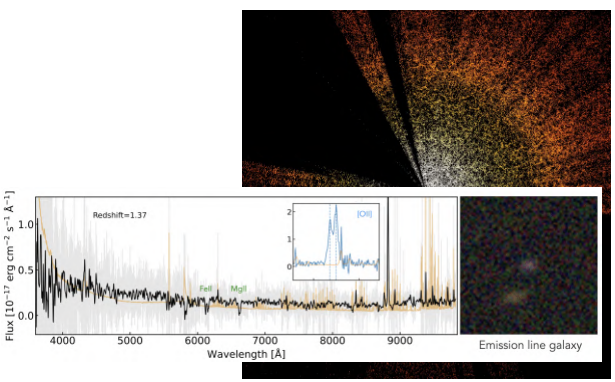
1000 boxes with varying
cosmology and ***feedback*** models

Gas Properties

Density Temperature



Galaxy Distribution



Current model optimised for Lyman Alpha forest

7 GPU minutes for a 50 Mpc simulation

130 million CPU core hours for TNG50

["BaryonBridge: Interpolants models for fast hydrodynamical simulations" Horowitz, Cuesta-Lazaro, Yehia ML4Astro workshop 2025]

Find straighter paths with OT

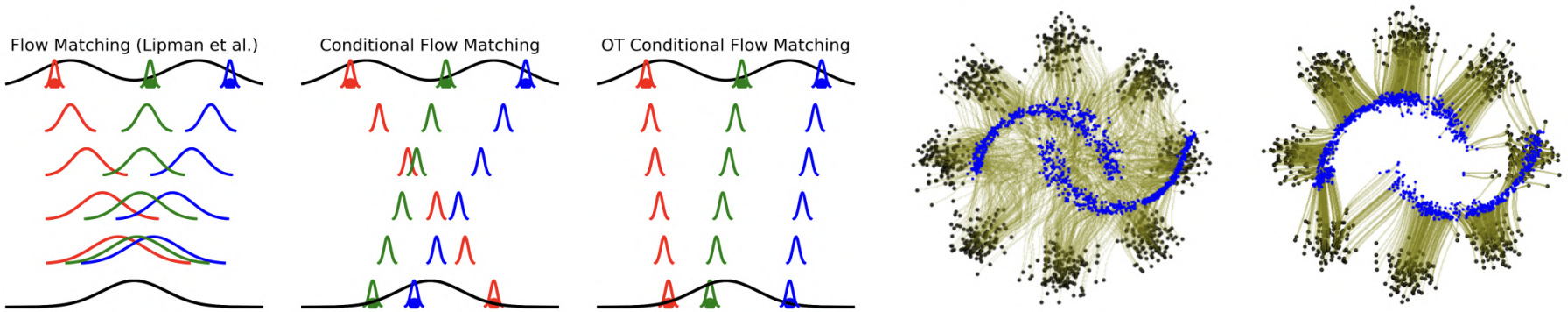
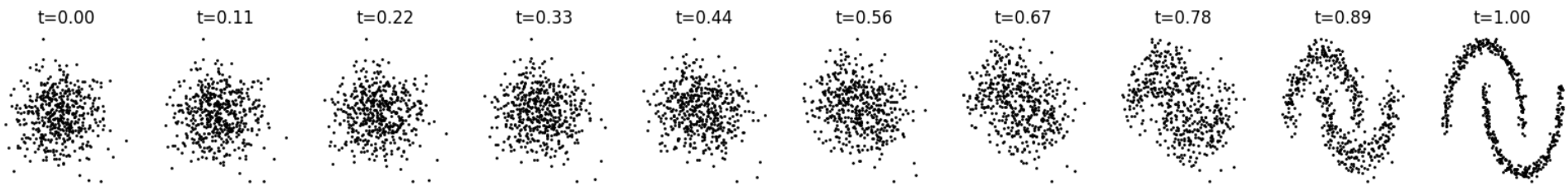


Figure 1: **Left:** Conditional flows from FM (Lipman et al., 2023), I-CFM (§3.2.2), and OT-CFM (§3.2.3). **Right:** Learned flows (green) from moons (blue) to 8gaussians (black) using I-CFM (centre-right) and OT-CFM (far right).

["Improving and generalizing flow-based generative models with minibatch optimal transport" Tong et al]

Tutorial 3: Flow Matching from scratch



x_0

Gaussian

x_1

Two Moons

Diffusion Models

Diffusion Models

Forward diffusion: Add Gaussian noise (fixed)

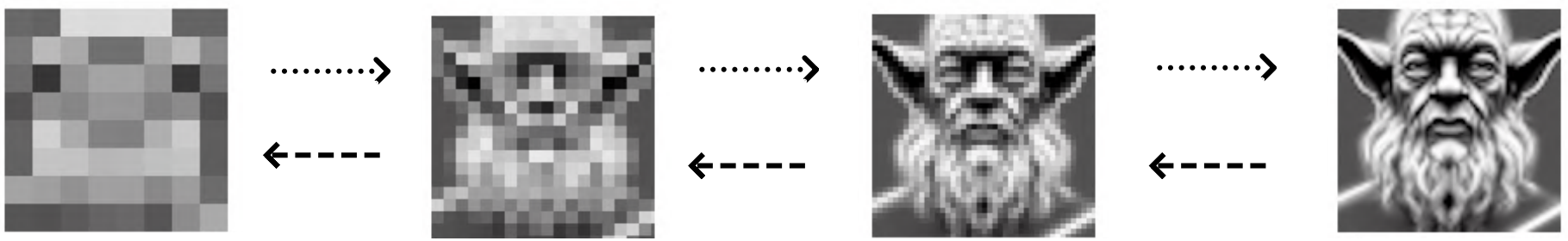
Diffusion Models

Fixed base distribution:

Gaussian

Forward diffusion: Add Gaussian noise (fixed)

Diffusion Models



Fixed base distribution: Gaussian

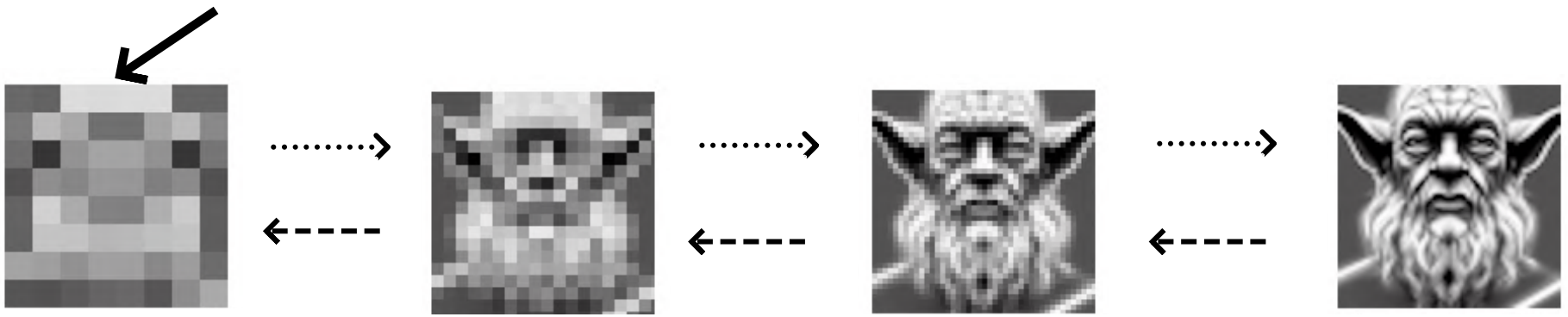
Forward diffusion: Add Gaussian noise (fixed)

Reverse diffusion: Denoise previous step

Diffusion Models

Prompt

A person half Yoda half Gandalf



Fixed base distribution: Gaussian

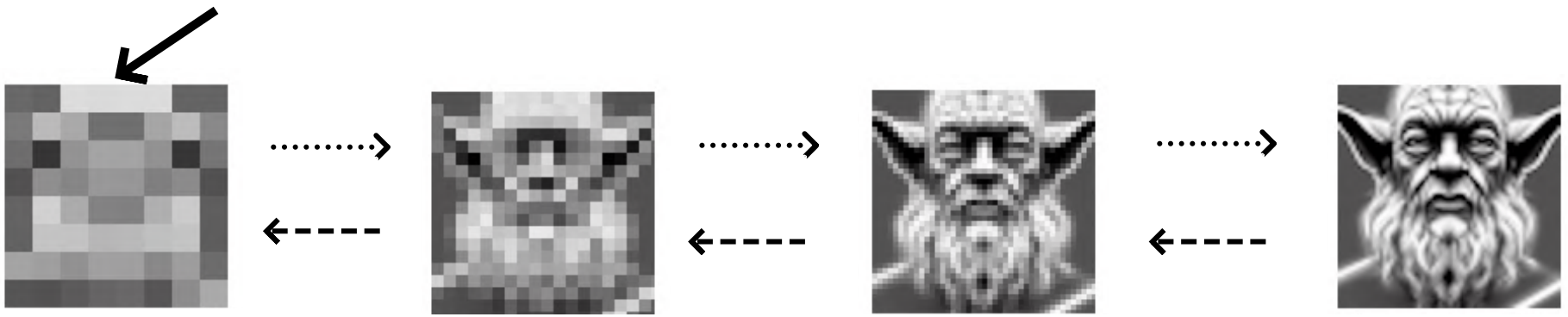
Forward diffusion: Add Gaussian noise (fixed)

Reverse diffusion: Denoise previous step

Diffusion Models

Prompt

A person half Yoda half Gandalf



Fixed base distribution: Gaussian

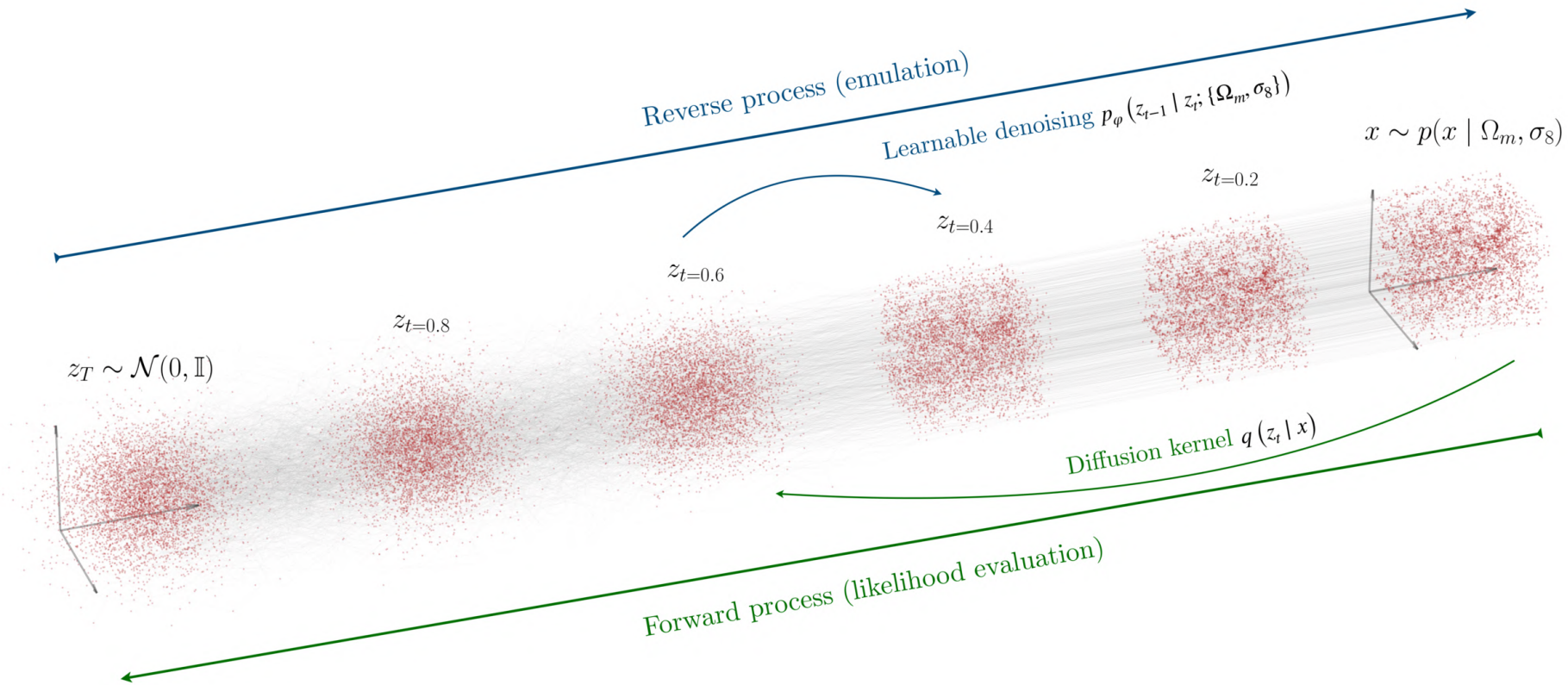
Denoising = Regression

⟵ Forward diffusion: Add Gaussian noise (fixed)

⟶ Reverse diffusion: Denoise previous step

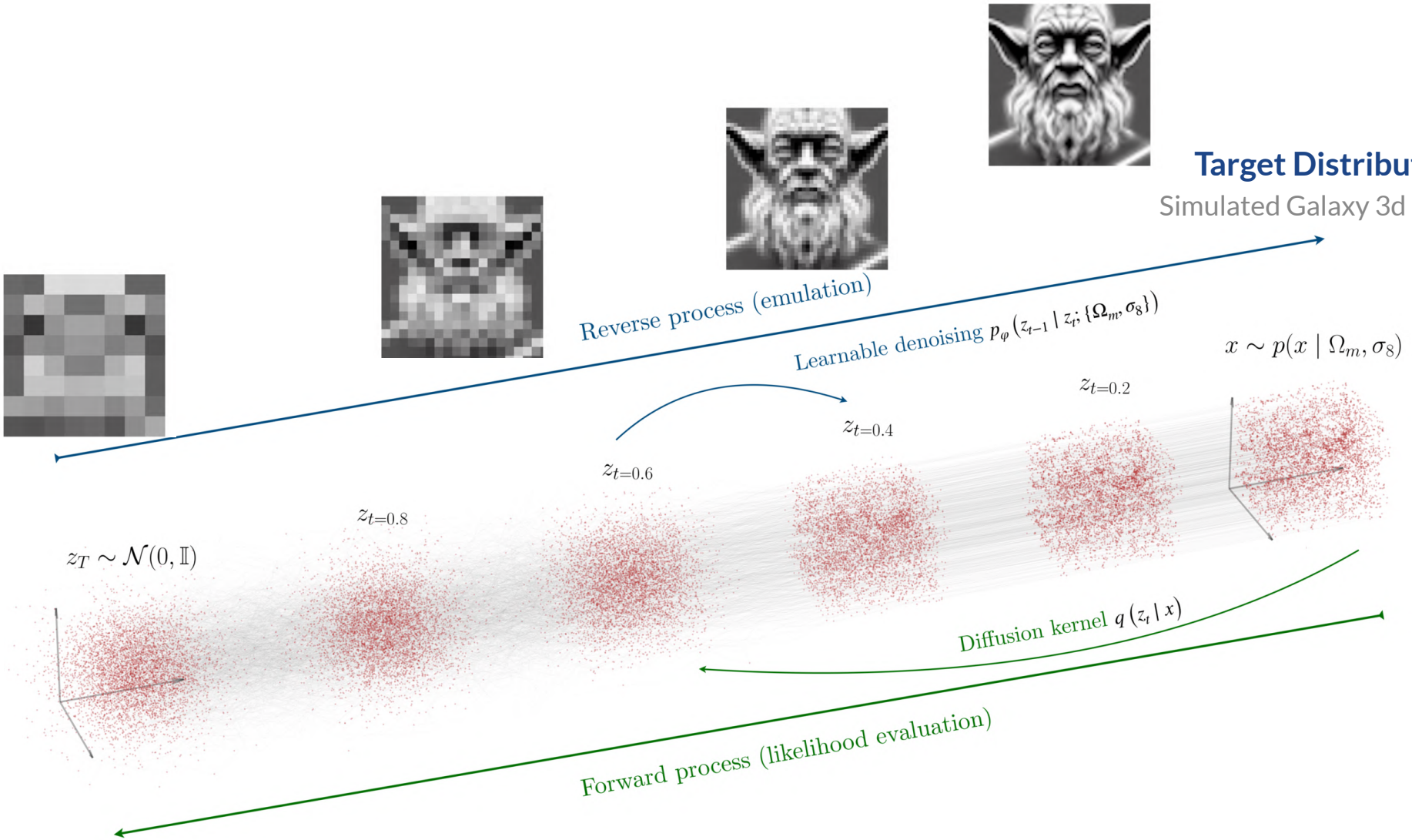
Target Distributio

Simulated Galaxy 3d Ma

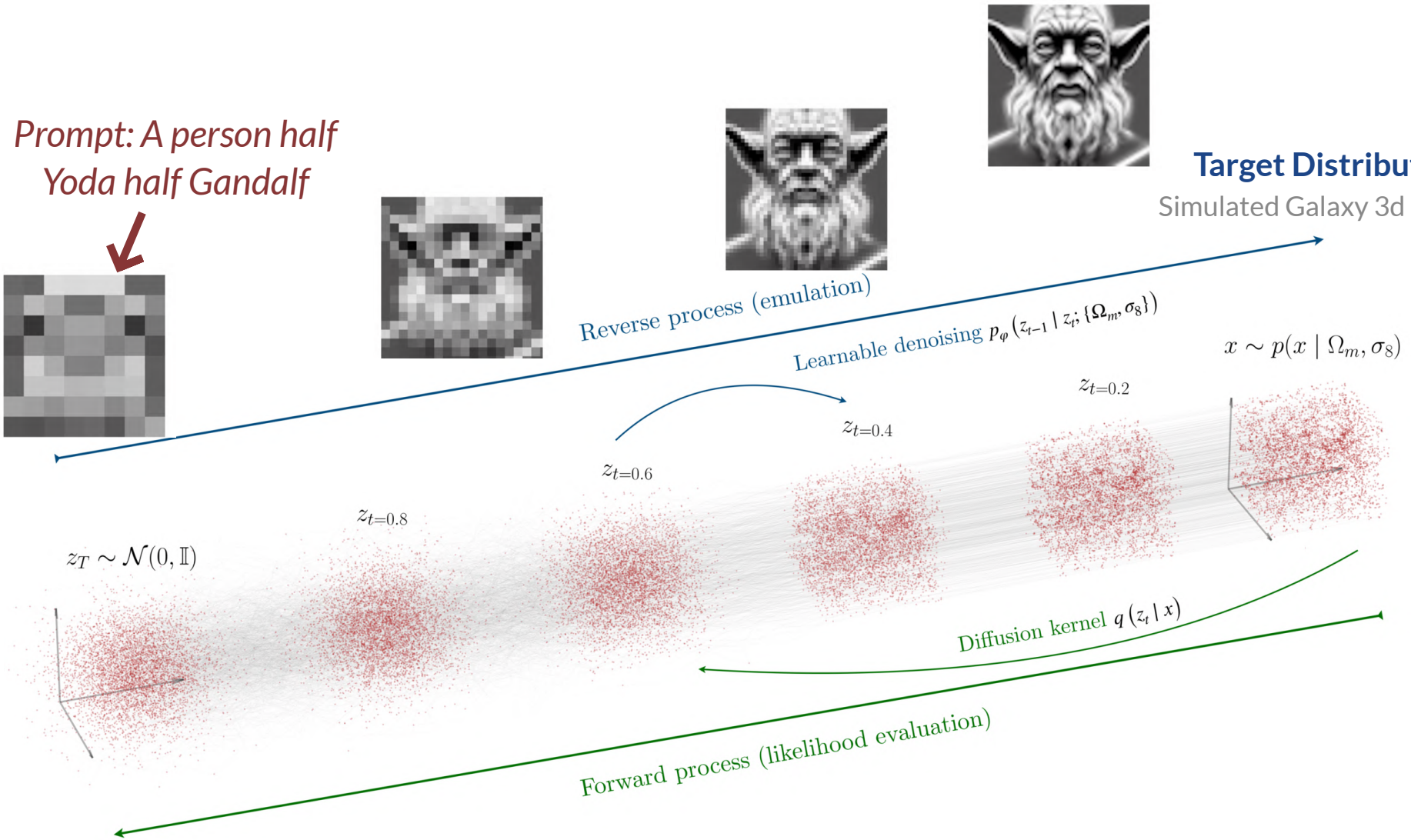


Base Distribution

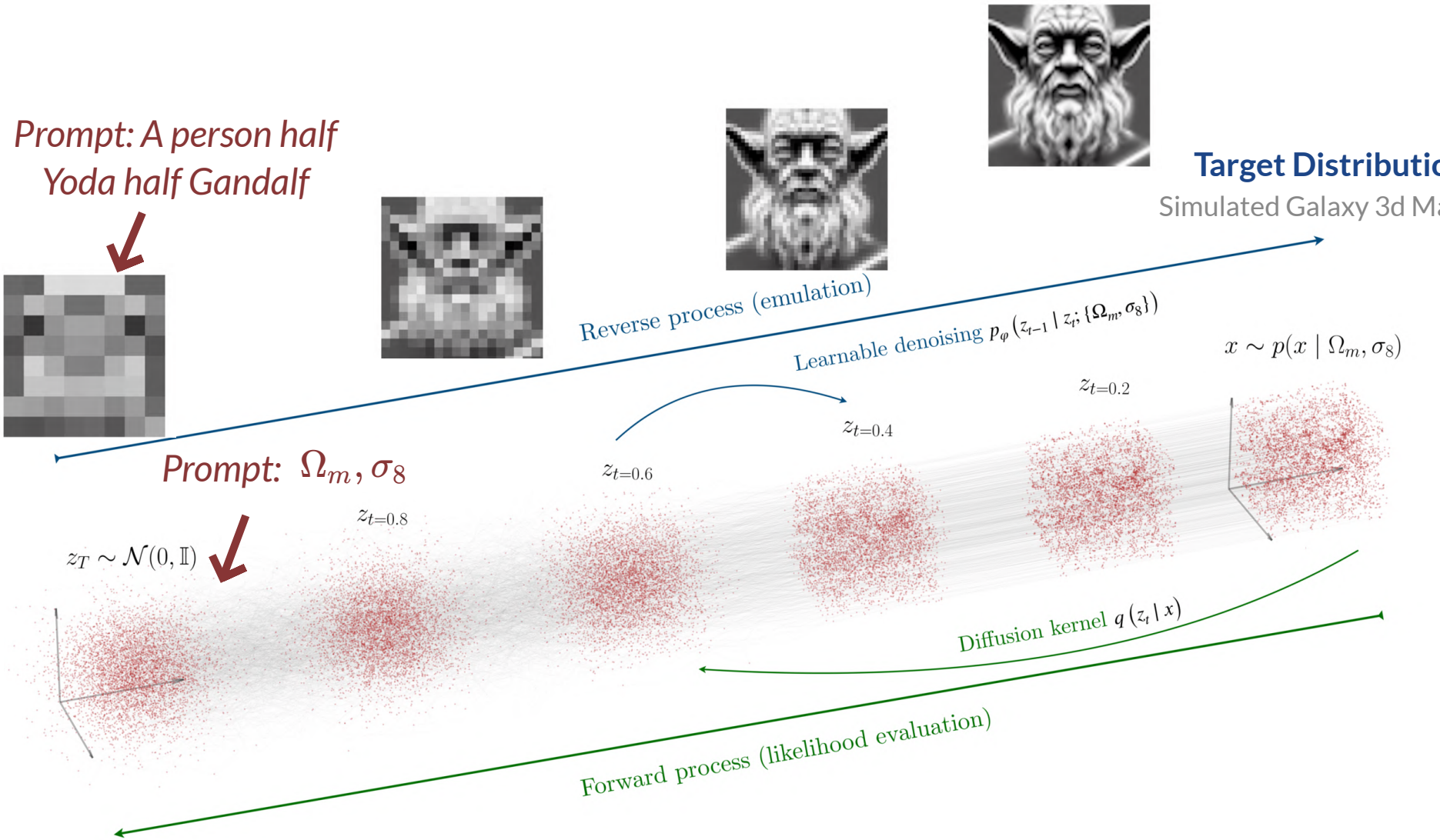
["A point cloud approach to generative modeling for galaxy surveys at the field level"
Cuesta-Lazaro and Mishra-Sharma
International Conference on Machine Learning ICML AI4Astro 2023, Spotlight talk, arXiv:2311.17141]



["A point cloud approach to generative modeling for galaxy surveys at the field level"
Cuesta-Lazaro and Mishra-Sharma
International Conference on Machine Learning ICML AI4Astro 2023, Spotlight talk, arXiv:2311.17141]



["A point cloud approach to generative modeling for galaxy surveys at the field level"
Cuesta-Lazaro and Mishra-Sharma
International Conference on Machine Learning ICML AI4Astro 2023, Spotlight talk, arXiv:2311.17141]

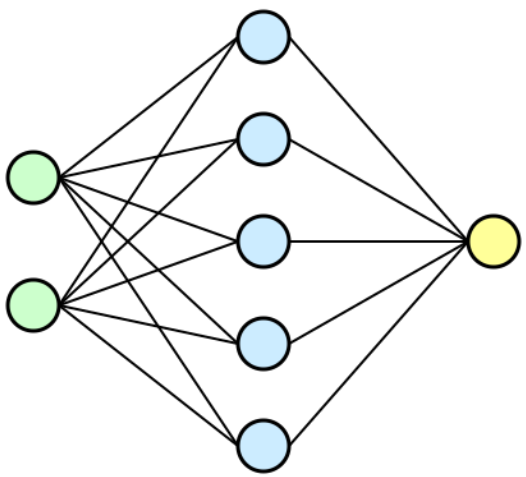


["A point cloud approach to generative modeling for galaxy surveys at the field level"
Cuesta-Lazaro and Mishra-Sharma
International Conference on Machine Learning ICML AI4Astro 2023, Spotlight talk, arXiv:2311.17141]

How good is my generative model?

$\mathrm{R} \sim p(\theta|x)$

$\mathrm{F} \sim \hat{p}(\theta|x)$



Real or Fake?

["A Practical Guide to Sample-based Statistical Distances for Evaluating Generative Models in Science" Bischoff et al 2024 arXiv:2403.12636]

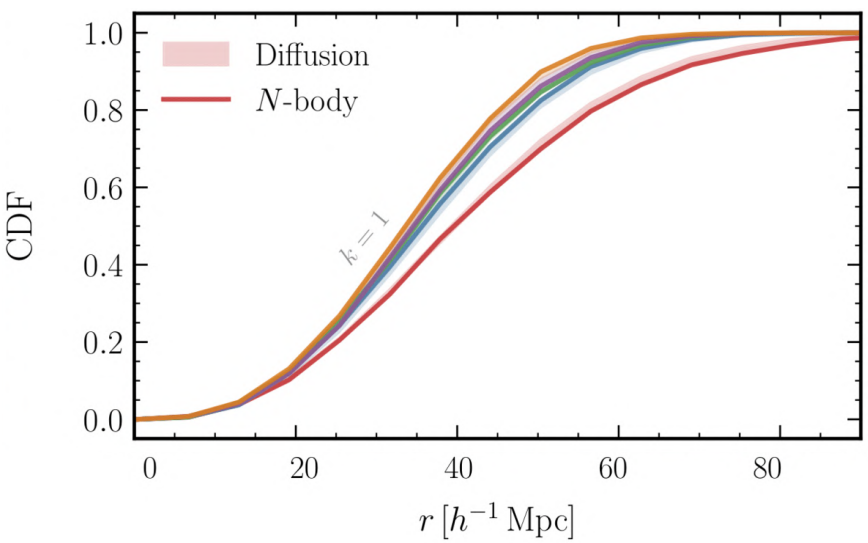
Reproducing summary statistics

Varying cosmological parameters

$\Omega_m = 0.10 \quad \sigma_8 = 0.92$ $\Omega_m = 0.18 \quad \sigma_8 = 0.68$ $\Omega_m = 0.27 \quad \sigma_8 = 0.92$

$\Omega_m = 0.35 \quad \sigma_8 = 0.62$ $\Omega_m = 0.44 \quad \sigma_8 = 0.95$

k Nearest neighbours



Pair separation

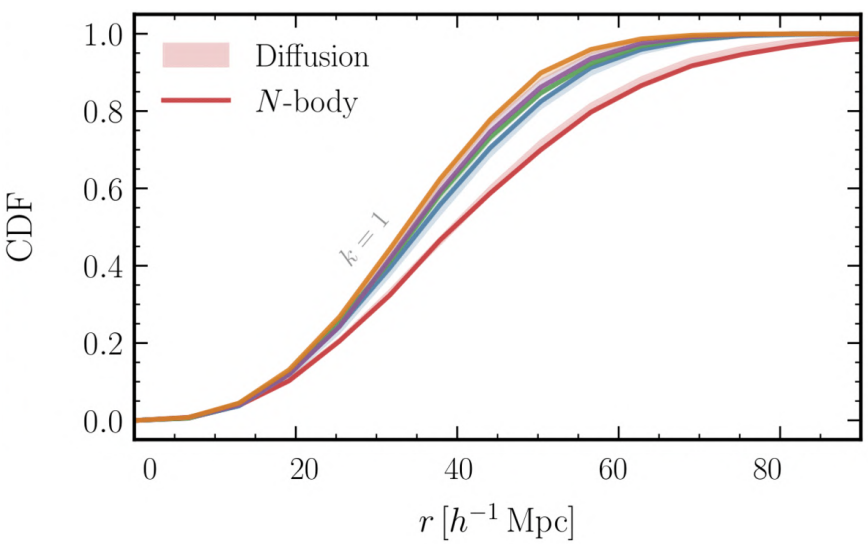
Reproducing summary statistics

Varying cosmological parameters

$\Omega_m = 0.10 \quad \sigma_8 = 0.92$ $\Omega_m = 0.18 \quad \sigma_8 = 0.68$ $\Omega_m = 0.27 \quad \sigma_8 = 0.92$

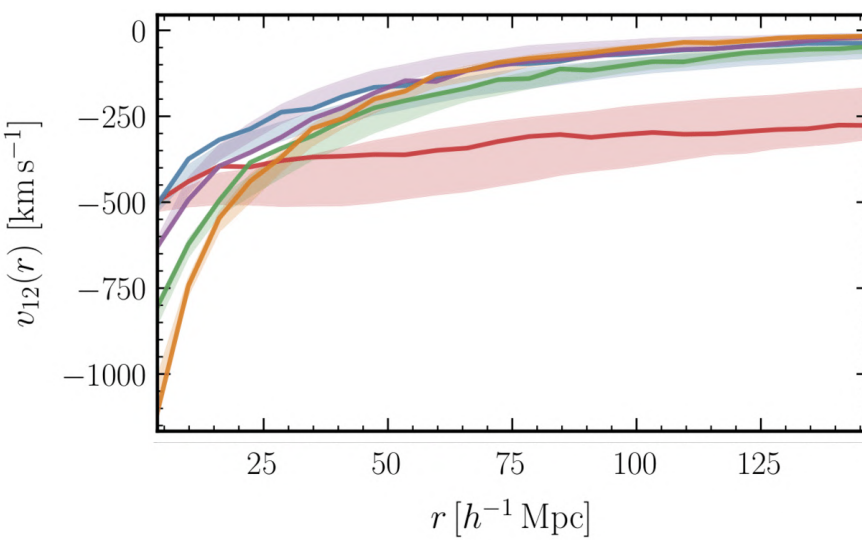
$\Omega_m = 0.35 \quad \sigma_8 = 0.62$ $\Omega_m = 0.44 \quad \sigma_8 = 0.95$

K Nearest neighbours



Pair separation

Mean relative velocity



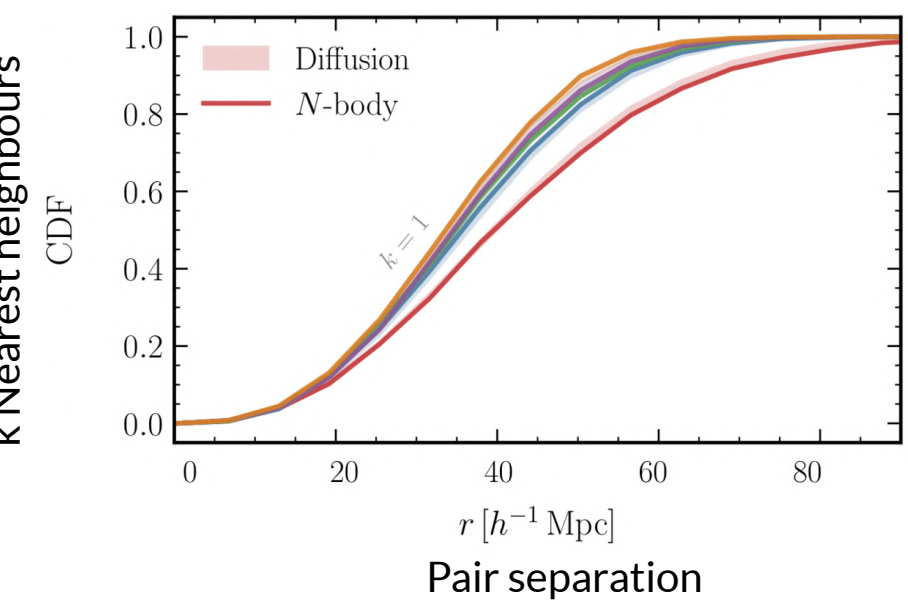
Pair separation

Reproducing summary statistics

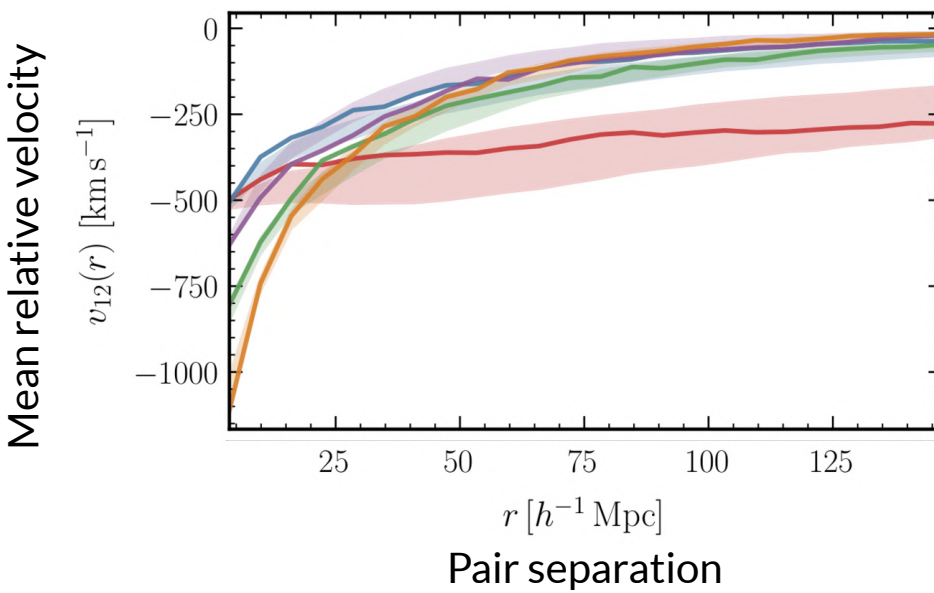
Varying cosmological parameters

$\Omega_m = 0.10 \quad \sigma_8 = 0.92$ $\Omega_m = 0.18 \quad \sigma_8 = 0.68$ $\Omega_m = 0.27 \quad \sigma_8 = 0.92$

$\Omega_m = 0.35 \quad \sigma_8 = 0.62$ $\Omega_m = 0.44 \quad \sigma_8 = 0.95$



Pair separation

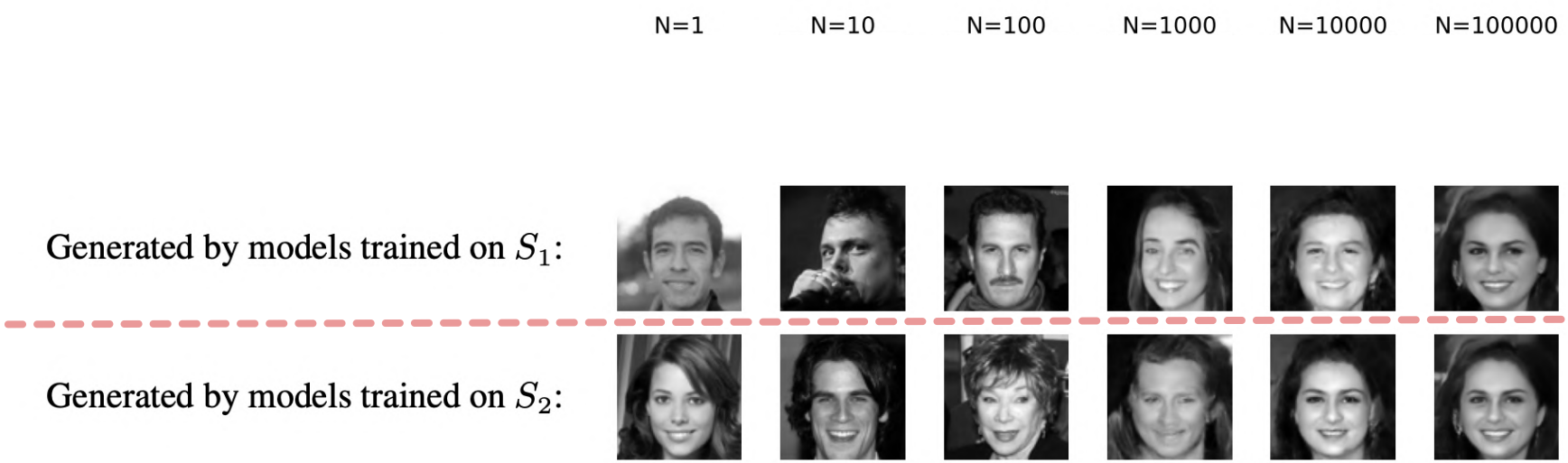


Pair separation

Physics as a testing ground: Well-understood summary statistics enable rigorous validation of generative models

Has my model learned the underlying density?

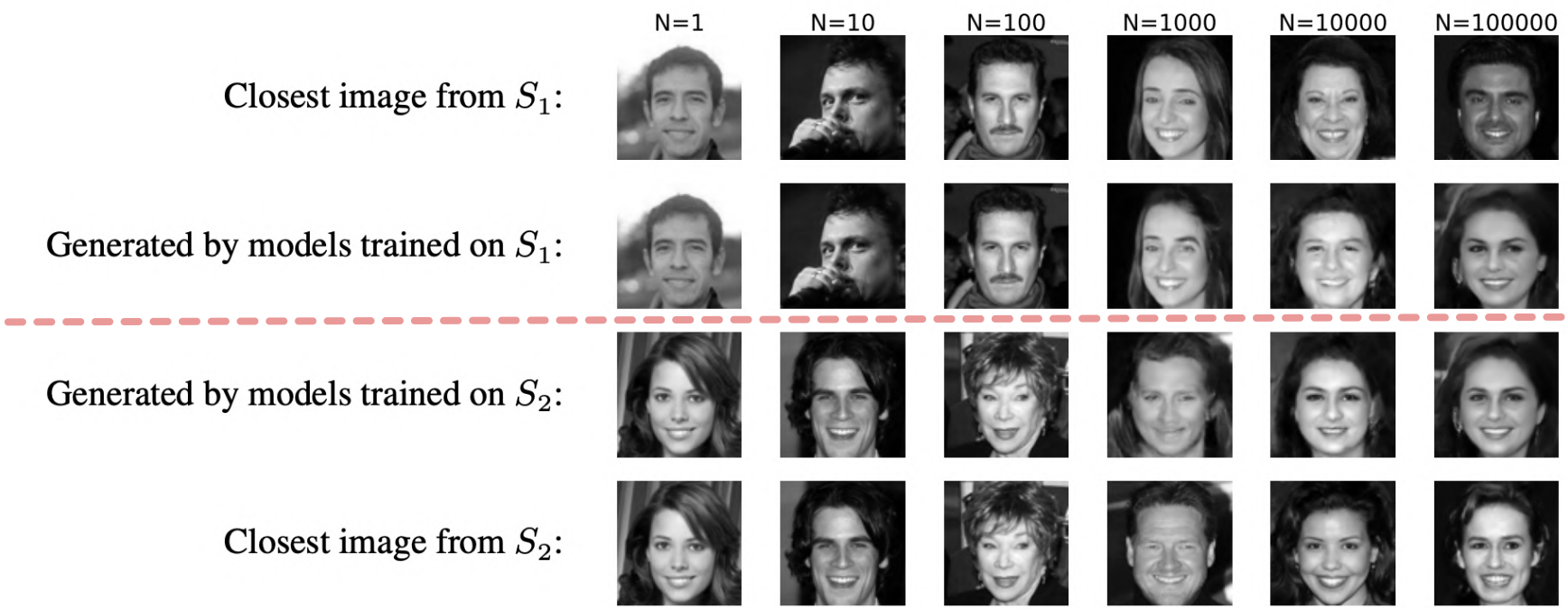
Split training set into non-overlapping S_1, S_2



["Generalization in diffusion models arises from geometry-adaptive harmonic representations" Kadkhodaie et al (2024)]

Has my model learned the underlying density?

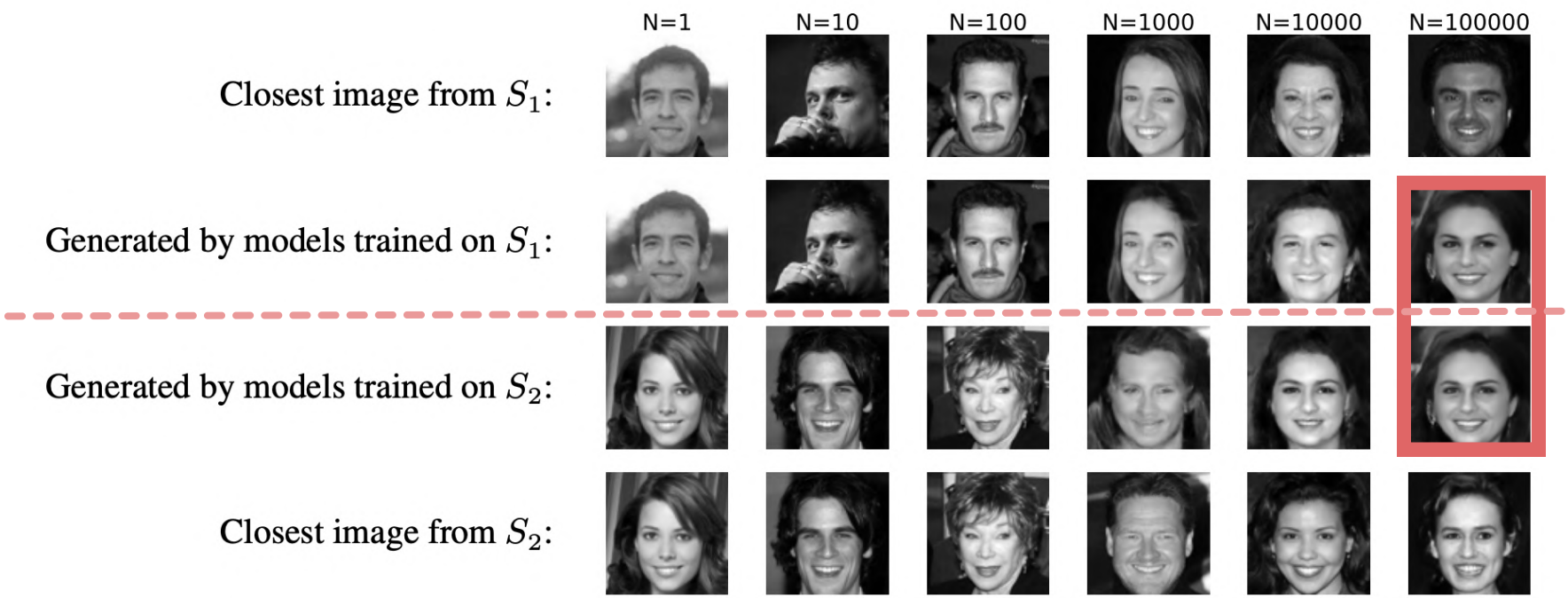
Split training set into non-overlapping S_1, S_2



["Generalization in diffusion models arises from geometry-adaptive harmonic representations" Kadkhodaie et al (2024)]

Has my model learned the underlying density?

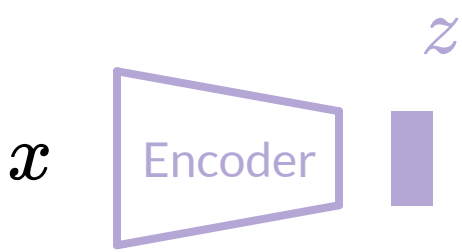
Split training set into non-overlapping S_1, S_2



["Generalization in diffusion models arises from geometry-adaptive harmonic representations" Kadkhodaie et al (2024)]

Generate = Understand?

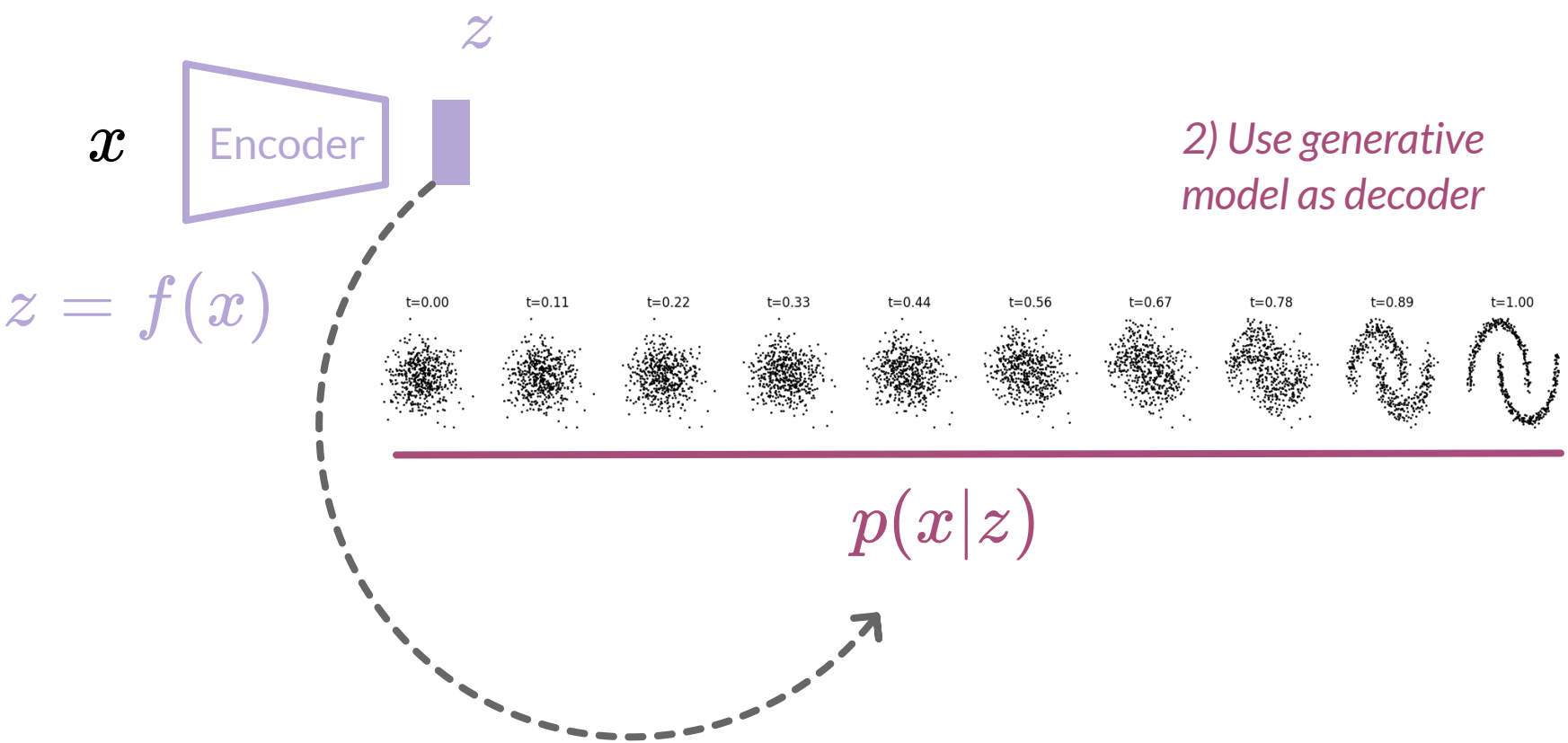
1) Compress into low dimensional latent



$$z = f(x)$$

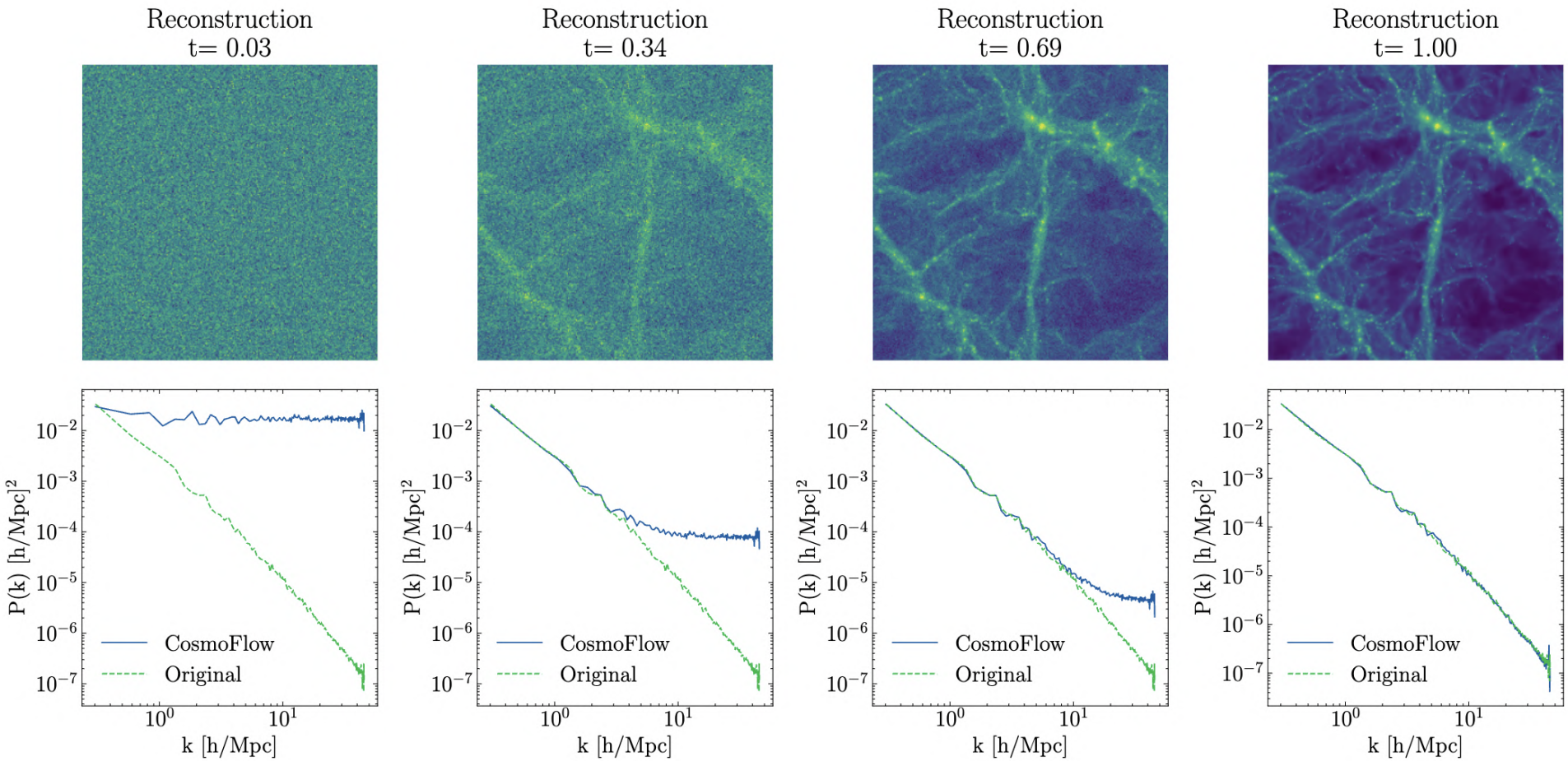
Generate = Understand?

1) Compress into low dimensional latent



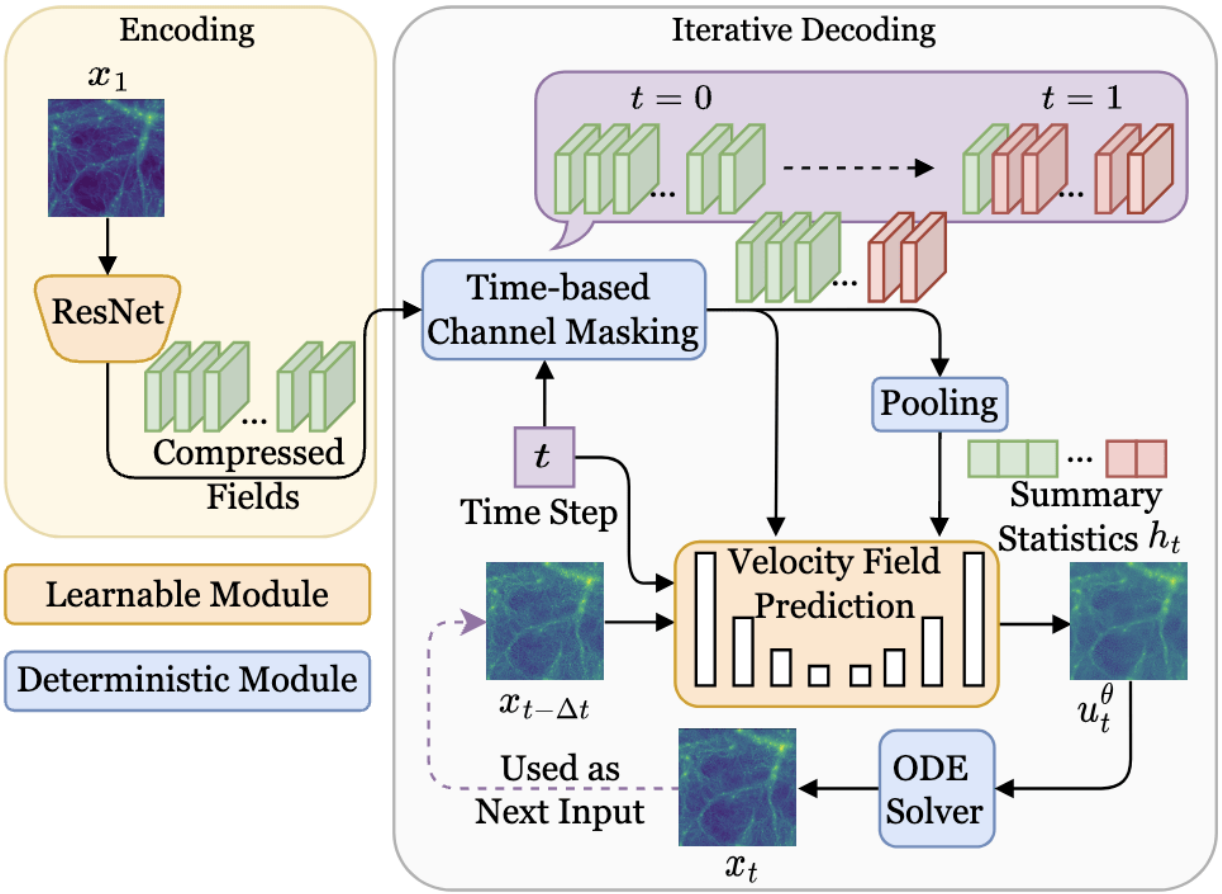
2) Use generative model as decoder

Autoregressive in Frequency



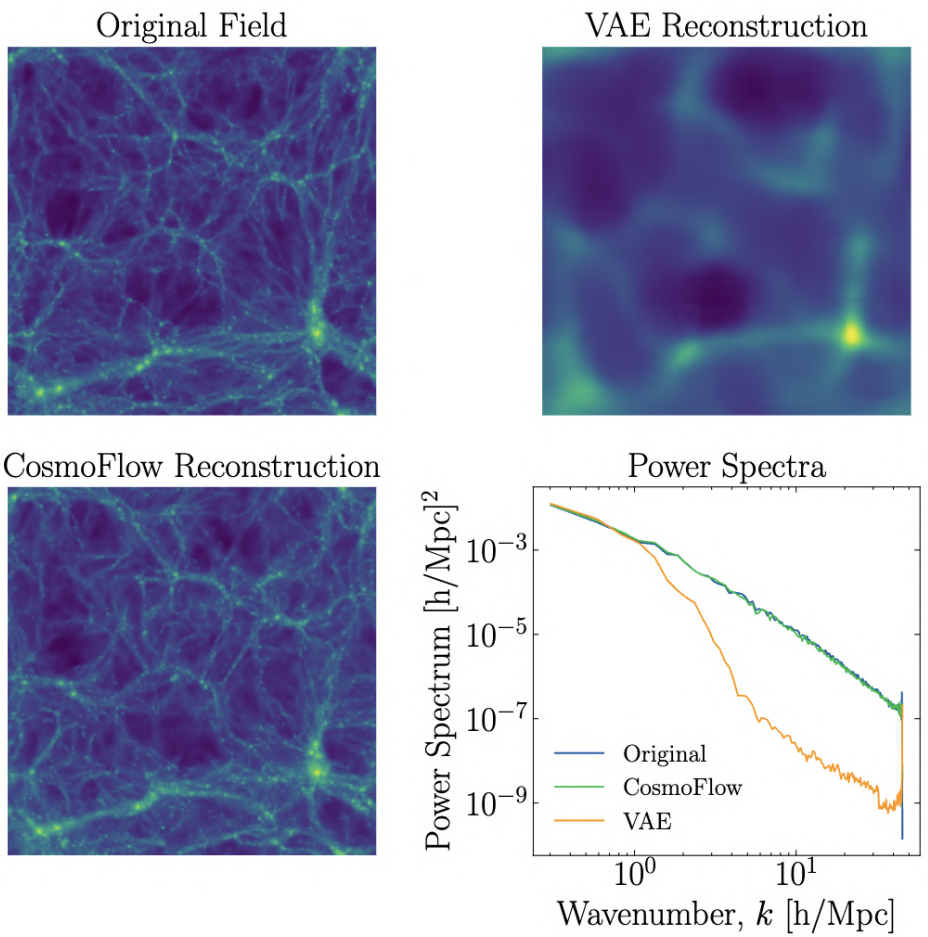
["CosmoFlow: Scale-Aware Representation Learning for Cosmology with Flow Matching" Kannan, Qiu, Cuesta-Lazaro, Jeong (in prep)]

Generate = Understand?



["CosmoFlow: Scale-Aware Representation Learning for Cosmology with Flow Matching" Kannan, Qiu, Cuesta-Lazaro, Jeong (in prep)]

Generate = Understand?



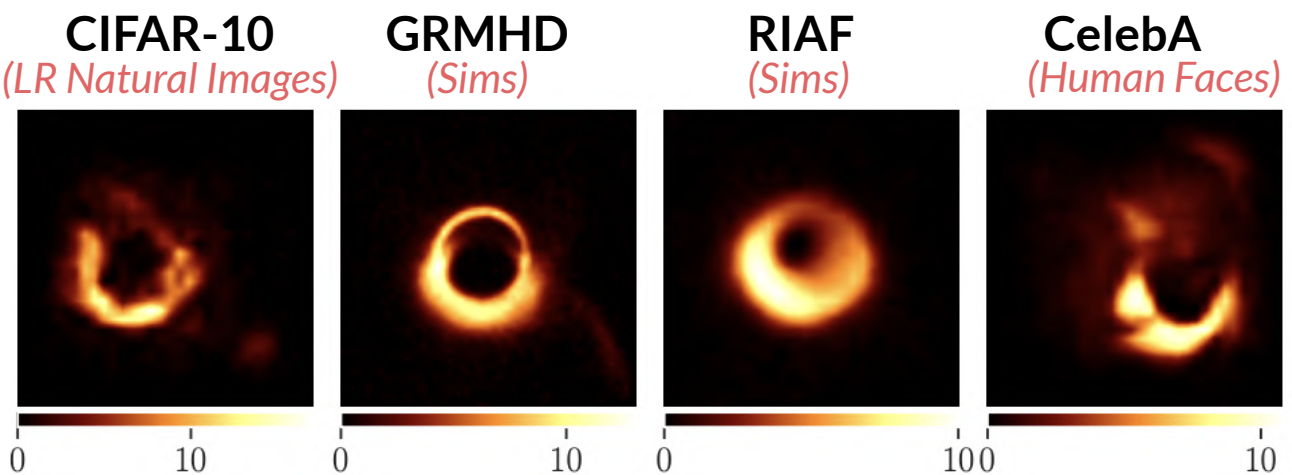
["CosmoFlow: Scale-Aware Representation Learning for Cosmology with Flow Matching" Kannan, Qiu, Cuesta-Lazaro, Jeong (in prep)]

Generative Priors: Plug and Play

$$\log p(x|y) = \log p(y|x) + \log p(x) + \text{constant}$$

Prior

EHT posterior samples with different priors



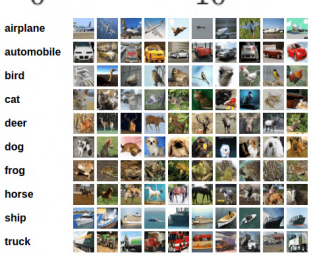
["Event-horizon-scale Imaging of M87* under Different Assumptions via Deep Generative Image Priors" Feng et al]

Generative Priors: Plug and Play

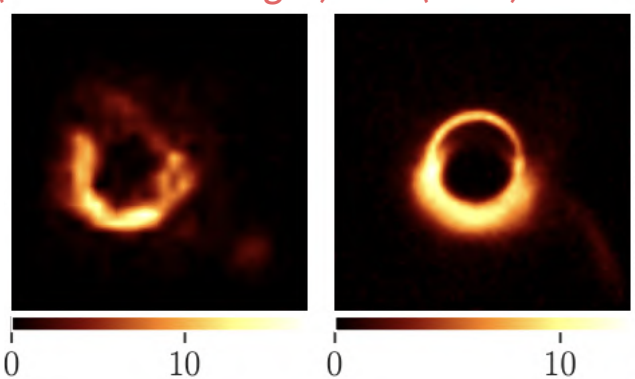
$$\log p(x|y) = \log p(y|x) + \underbrace{\log p(x)}_{\text{Prior}} + \text{constant}$$

EHT posterior samples with different priors

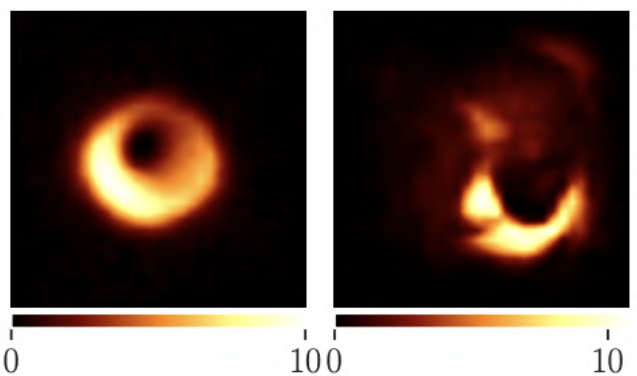
CIFAR-10 *(LR Natural Images)*



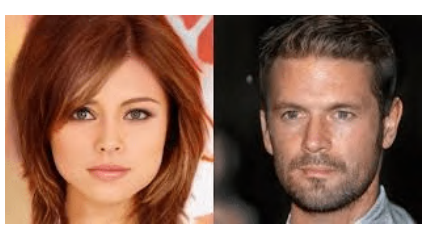
GRMHD *(Sims)*



RIAF *(Sims)*



CelebA *(Human Faces)*



["Event-horizon-scale Imaging of M87* under Different Assumptions via Deep Generative Image Priors" Feng et al]

Generative priors: learn directly?

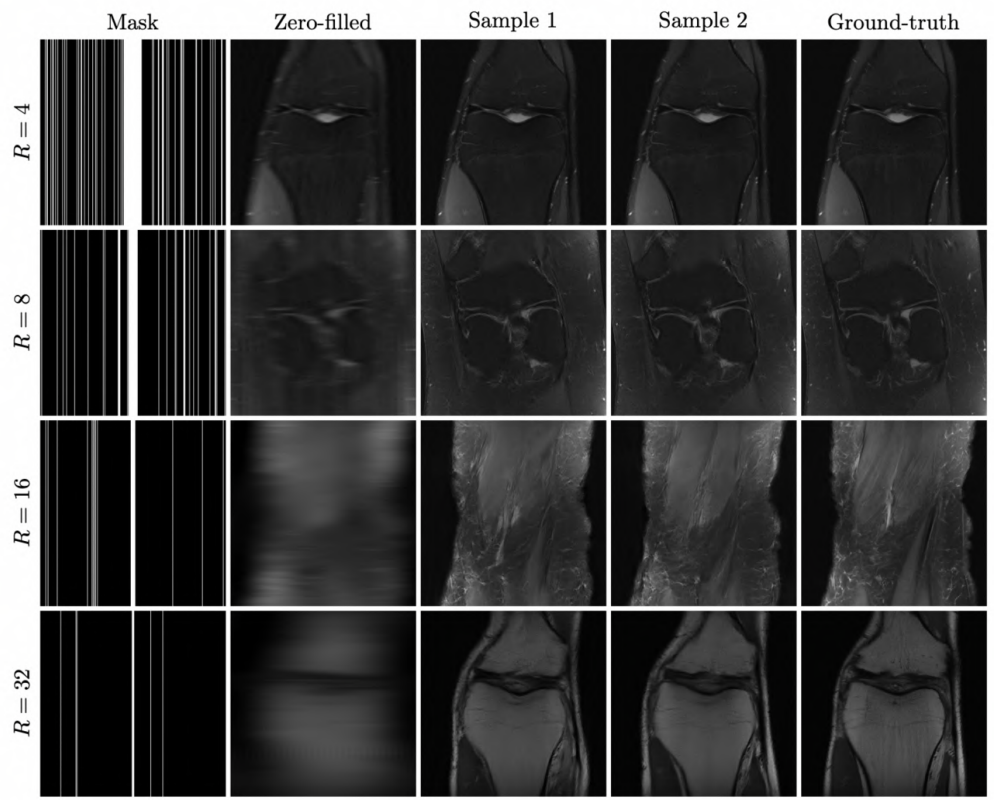


Figure 6. Examples of posterior samples for accelerated MRI using a diffusion prior trained from k-space observations only. Posterior samples are detailed and present plausible variations, while remaining consistent with the observation. We provide the zero-filled inverse, where missing frequencies are set to zero, as baseline.

["Learning Diffusion Priors from Observations by Expectation Maximization"

Rozet et al]

Bridging two distributions

How is the bridge constrained?

Normalizing flows: Reverse = Forward inverse

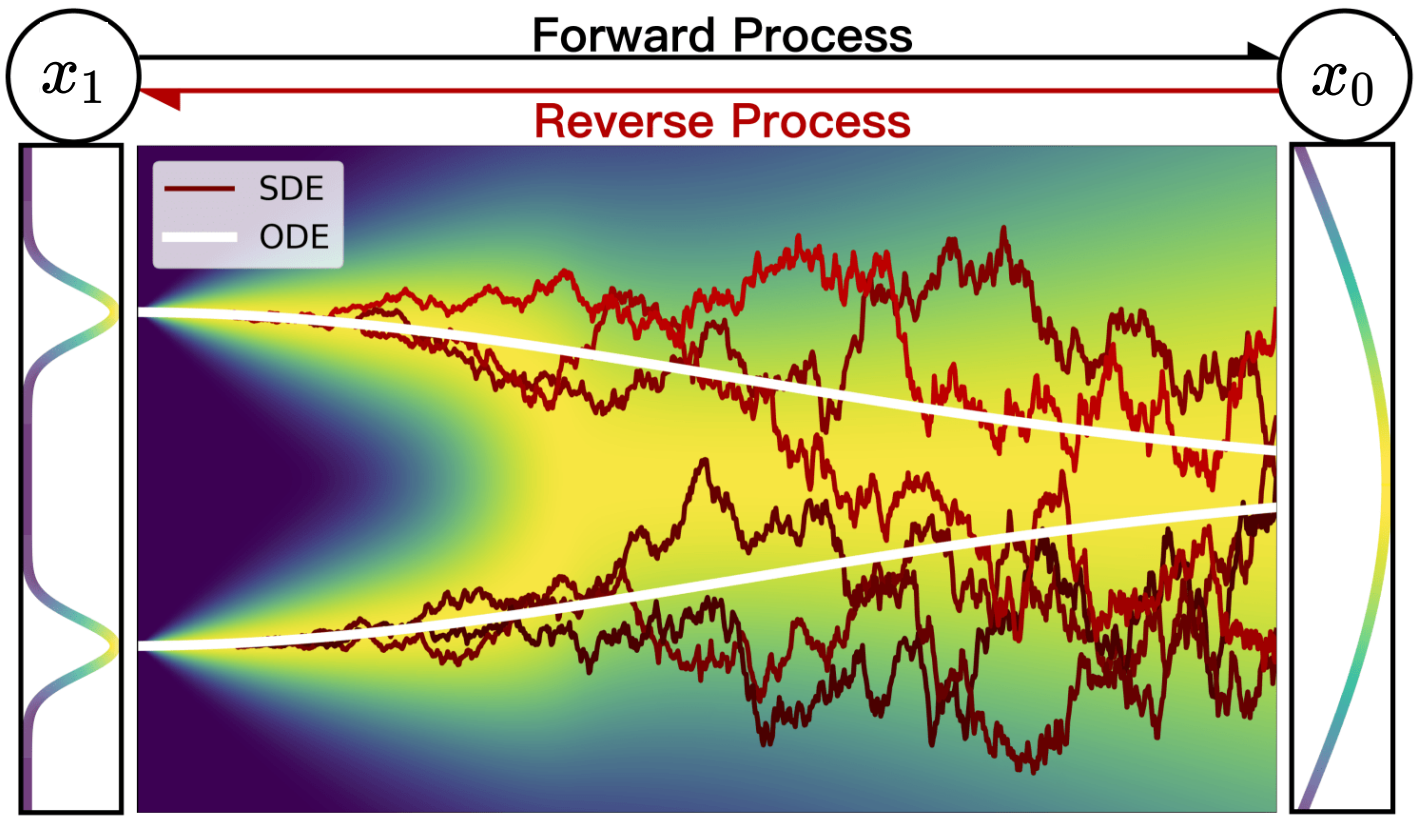
Diffusion: Forward = Gaussian noising

Flow Matching: Forward = Interpolant

is p(x0) restricted?

Diffusion: p(x0) is Gaussian

Normalising flows: p(x0) can be evaluated



Data

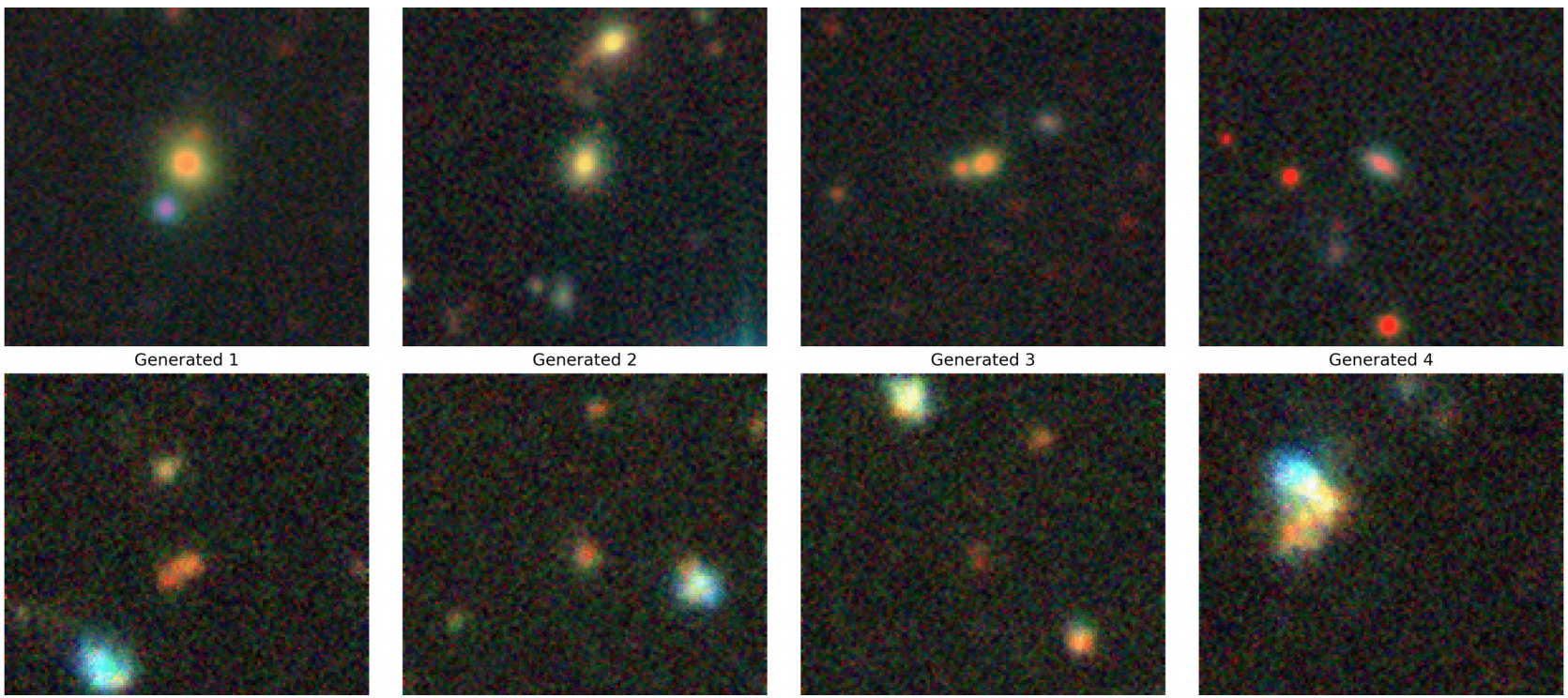
Base

(Exact likelihood evaluation)

Is bridge stochastic (SDE) or deterministic (ODE)?

Diffusion: Stochastic (SDE) *Normalising flows: Deterministic (ODE)*

Tutorial 4: Generating Galaxy Images



References

carolina.clzr@gmail.com

- **Books** by Kevin P. Murphy
  - *Machine learning, a probabilistic perspective*
  - *Probabilistic Machine Learning: advanced topics*
- **ML4Astro workshop** *https://ml4astro.github.io/icml2023/*
- **ProbAI summer school** *https://github.com/probabilisticai/probai-2023*
- **IAIFI Summer school**
  - *https://github.com/iaifi/summer-school-2023*
  - Tutorials *https://github.com/florpi/summer_school_generative*
- **Blogposts**
  - *https://lilianweng.github.io/posts/2018-10-13-flow-models/*
  - *https://sander.ai/2023/07/20/perspectives.html*
  - https://mlg.eng.cam.ac.uk/blog/2024/01/20/flow-matching.html