Generative AI with Stochastic Differential Equations

An introduction to flow and diffusion models

MIT IAP 2025 | Jan 21, 2025

Peter Holderrieth and Ezra Erives

Sponsor: Tommi Jaakkola

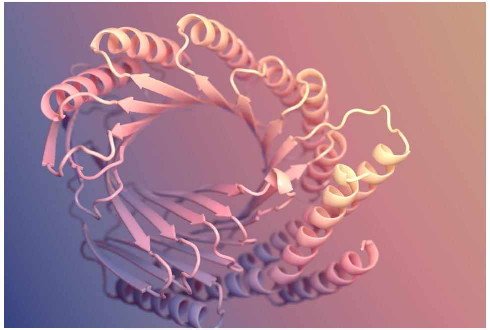
Flow and Diffusion Models: state-of-the-art models for generating images, videos, proteins!

Stable Diffusion

DALEE

OpenAI Sora

Meta MovieGen



AlphaFold3

RFDiffusion

Most SOTA generative AI models for images/videos/proteins/robotics: Diffusion and Flow Models

Section 1:

From Generation to Sampling

Goal: Formalize what it means to “generate” something.

We represent images/videos/protein as vectors

Images:

- Height H and Width W
- 3 color channels (RBG)

$$z \in \mathbb{R}^{H \times W \times 3}$$

Videos:

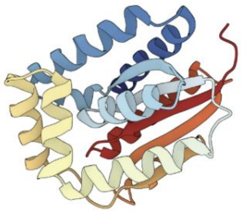
- T time frames
- Each frame is image

$$z \in \mathbb{R}^{T \times H \times W \times 3}$$

Molecular structures:

- N atoms
- Each atom has 3 coordinate

$$z \in \mathbb{R}^{N \times 3}$$

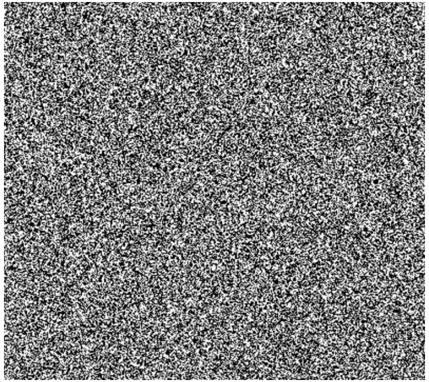


We represent the objects we want to generate as vectors:

$$z \in \mathbb{R}^d$$

What does it mean to successfully generate something?

Prompt: “A picture of a dog”

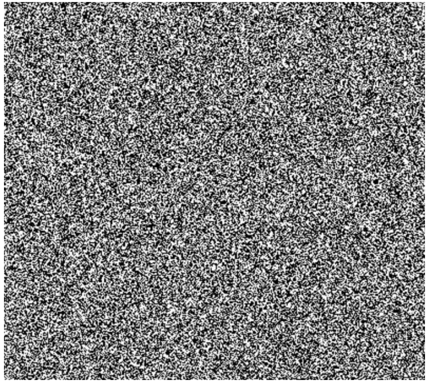


Useless < Bad < Wrong animal < Great!

These are subjective statements - Can we formalize this?

Data Distribution: How "likely" are we to find this picture in the internet?

Prompt: "A picture of a dog"



Impossible <

Rare <

Unlikely <

Very likely

How good an image is ~= How likely it is under the data distribution

Generation as sampling from the data distribution

Data distribution: Distribution of objects that we want to generate: $\boxed{p_{\text{data}}}$

Probability density:

$$\boxed{\begin{aligned} p_{\text{data}} : \mathbb{R}^d &\to \mathbb{R}_{\geq 0}, \\ z &\mapsto p_{\text{data}}(z) \end{aligned}}$$

Note: We don't know the probability density!

Generation means sampling the data distribution:

$$\boxed{z \sim p_{\text{data}}} \Rightarrow z =$$

A Dataset consists of samples from the data distribution

Training requires datasets: To train our algorithms, we need a **dataset**.

Examples:

- Images: Publicly available images from the internet
- Videos: YouTube
- Protein structures: Scientific data (e.g. Protein Data Bank)

A dataset consists of a finite number of samples from the data distribution:

$$z_1, \ldots, z_N \sim p_{\text{data}}$$

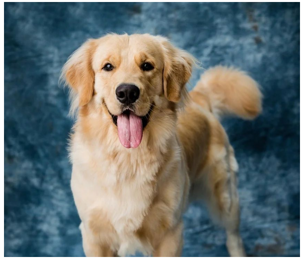
Conditional Generation allows us to condition on prompts

Data distribution $\boxed{p_{\text{data}}}$

Fixed prompt

"Dog"





Conditional data distribution $\boxed{p_{\text{data}}(\cdot|y)}$

Condition variable: **y**

y="Dog" y="Cat" y="Landscape"



Conditional generation means sampling the conditional data distribution:

$$z \sim p_{\text{data}}(\cdot|y)$$

We will first focus on unconditional generation and then learn how to translate an unconditional model to a conditional one.

Generative Models generate samples from data distribution

Initial distribution: $\boxed{p_{\text{init}}}$ ***Default:*** $\boxed{p_{\text{init}} = \mathcal{N}(0, I_d)}$

A generative model converts samples from a initial distribution (e.g. Gaussian) into samples from the data distribution:

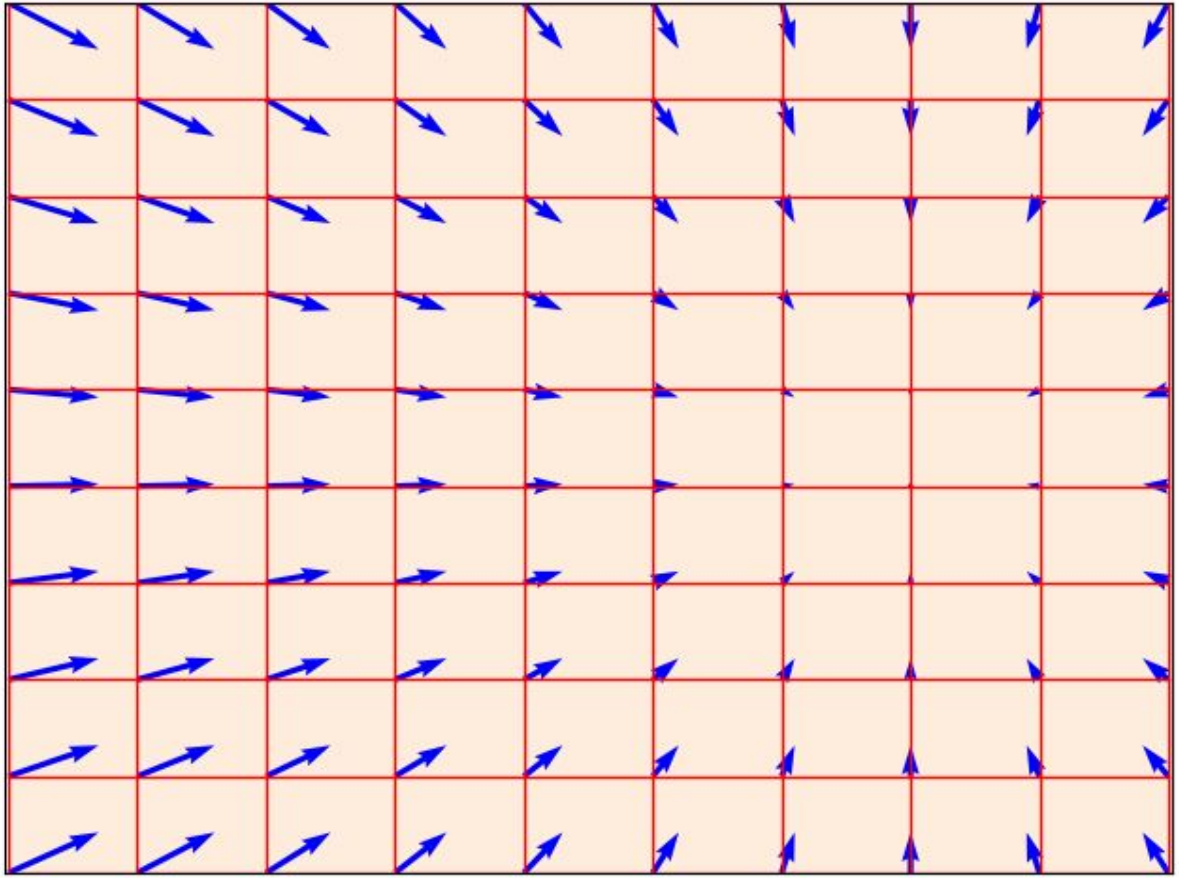
$\boxed{x \sim p_{\text{init}}}$ ⇒ Generative Model ⇒ $\boxed{z \sim p_{\text{data}}}$

Section 2:

Flow and Diffusion Models

Goal: Understand differential equations and how we can build generative models with them.

Flow - Example



Existence and Uniqueness Theorem ODEs

Theorem (Picard–Lindelöf theorem): If the vector field $u_t(x)$ is continuously differentiable with bounded derivatives, then a *unique* solution to the ODE

$$X_0 = x_0, \qquad \frac{\mathrm{d}}{\mathrm{d}t} X_t = u_t(X_t)$$

exists. In other words, a flow map exists. More generally, this is true if the vector field is **Lipschitz**.

Key takeaway: In the cases of practical interest for machine learning, **unique solutions to ODE/flows exist**.

Math class: Construct solutions via Picard-Iteration

Example: Linear ODE

Simple vector field:

$$u_t(x) = -\theta x \qquad (\theta > 0)$$

Claim: Flow is given by

$$\psi_t(x_0) = \exp\left(-\theta t\right) x_0$$

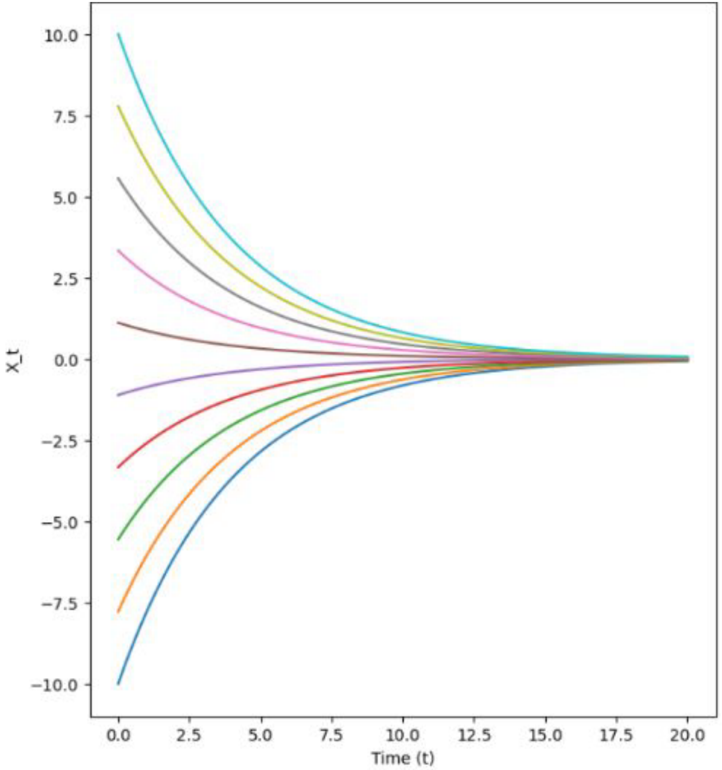
Proof:

1. Initial condition:

$$\psi_t(x_0) = \exp(0) x_0 = x_0$$

2. ODE:

$$\frac{\mathrm{d}}{\mathrm{d}t}\psi_t(x_0) = \frac{\mathrm{d}}{\mathrm{d}t}\left(\exp\left(-\theta t\right) x_0\right) = -\theta \exp\left(-\theta t\right) x_0 = -\theta \psi_t(x_0) = u_t(\psi_t(x_0))$$



Numerical ODE simulation - Euler method

Algorithm 1 Simulating an ODE with the Euler method

Require: Vector field u_t, initial condition x_0, number of steps n

1: Set $t = 0$
2: Set step size $h = \frac{1}{n}$
3: Set $X_0 = x_0$
4: **for** $i = 1, \ldots, n-1$ **do**
5: $\quad X_{t+h} = X_t + h u_t(X_t)$ *Small step into direction of vector field*
6: $\quad$ Update $t \leftarrow t + h$
7: **end for**
8: **return** $X_0, X_h, X_{2h}, \ldots, X_1$ *Return trajectory*

Toy example

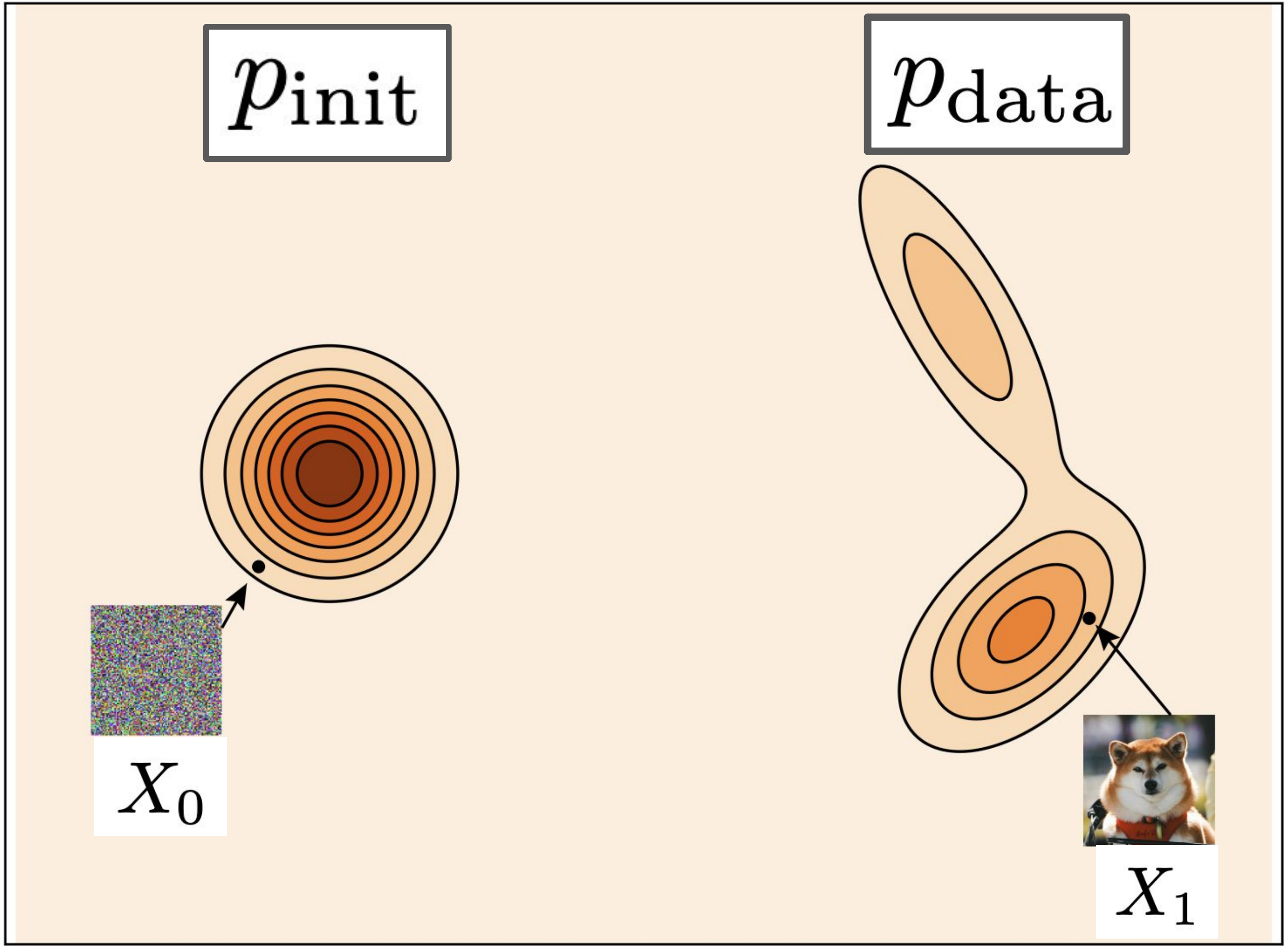


Figure credit: Yaron Lipman

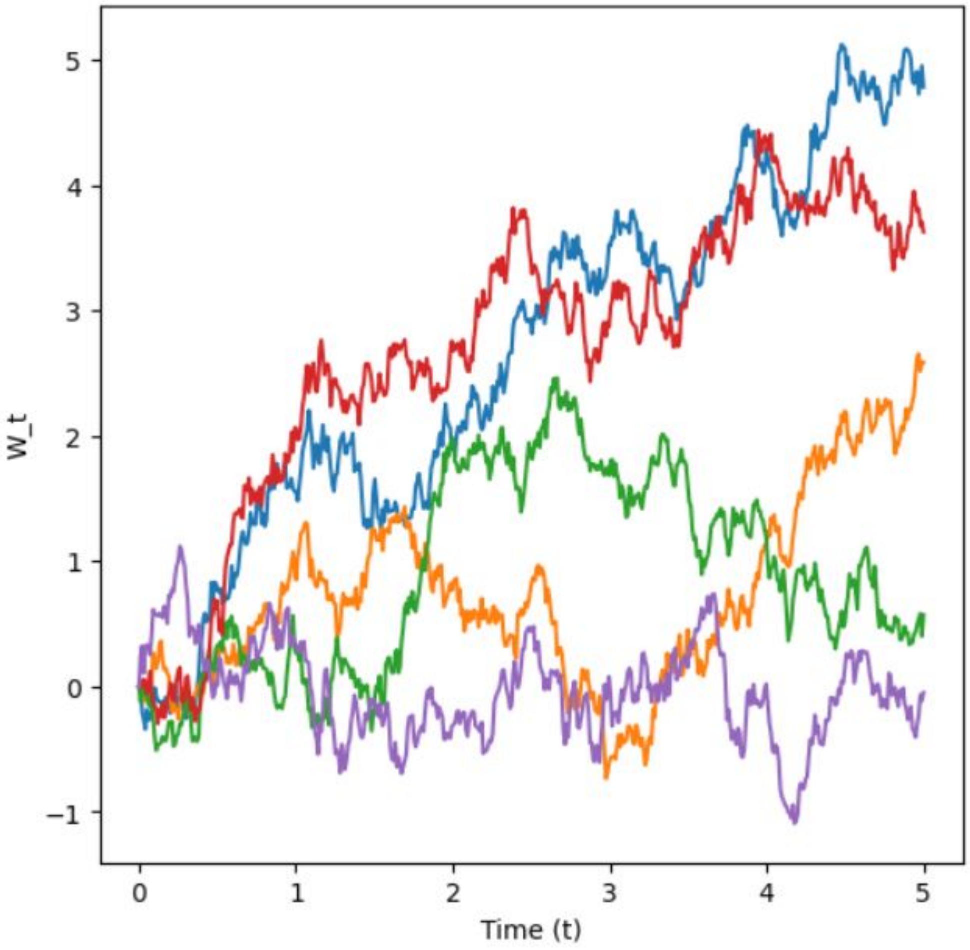
How to generate objects with a Flow Model

Algorithm 1 Sampling from a Flow Model with Euler method

Require: Neural network vector field u_t^θ, number of steps n

1: Set $t = 0$
2: Set step size $h = \frac{1}{n}$
3: Draw a sample $X_0 \sim p_{\text{init}}$ *Random initialization!*
4: **for** $i = 1, \ldots, n-1$ **do**
5: $\quad X_{t+h} = X_t + h u_t^\theta(X_t)$
6: $\quad$ Update $t \leftarrow t + h$
7: **end for**
8: **return** X_1 *Return final point*

Brownian Motion



Existence and Uniqueness Theorem SDEs

Theorem: If the vector field $u_t(x)$ is continuously differentiable with bounded derivatives and the diffusion coeff. is continuous, then a *unique* solution to the SDE

$$X_0 = x_0, \quad \mathrm{d}X_t = u_t(X_t)\mathrm{d}t + \sigma_t \mathrm{d}W_t$$

exists. More generally, this is true if the vector field is **Lipschitz**.

Key takeaway: In the cases of practical interest for machine learning, **unique solutions to SDE**.

Stochastic calculus class: Construct solutions via stochastic integrals and Ito-Riemann sums

Numerical SDE simulation (Euler-Maruyama method)

Algorithm 2 Sampling from a SDE (Euler-Maruyama method)

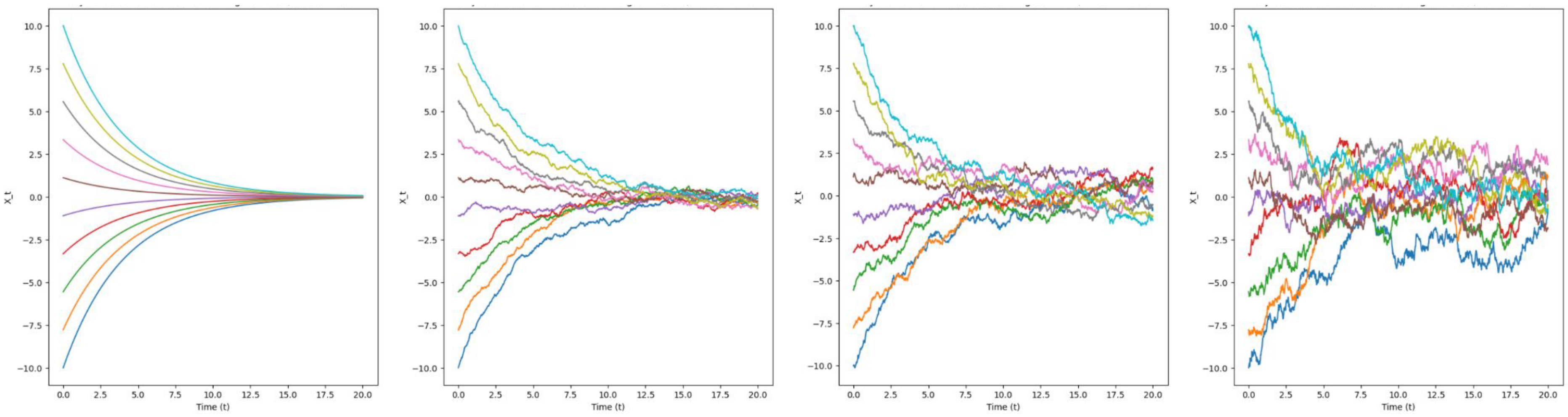
Require: Vector field u_t, number of steps n, diffusion coefficient σ_t

Set $t = 0$
Set step size $h = \frac{1}{n}$
Set $X_0 = x_0$
for $i = 1, \ldots, n-1$ **do**
    Draw a sample $\epsilon \sim \mathcal{N}(0, I_d)$
    $X_{t+h} = X_t + hu_t(X_t) + \sigma_t \sqrt{h}\epsilon$
    Update $t \leftarrow t + h$
end for
return $X_0, X_h, X_{2h}, X_{3h}, \ldots, X_1$

Add additional noise with var=h scaled by diffusion coefficient σ_t

Ornstein-Uhlenbeck Process

$$\mathrm{d}X_t = -\theta X_t \mathrm{d}t + \sigma dW_t$$



Increasing diffusion coefficient σ

Algorithm 2 Sampling from a Diffusion Model (Euler-Maruyama method)

Require: Neural network u_t^θ, number of steps n, diffusion coefficient σ_t

Set $t = 0$

Set step size $h = \frac{1}{n}$

Draw a sample $X_0 \sim p_{\text{init}}$

for $i = 1, \ldots, n-1$ **do**

    Draw a sample $\epsilon \sim \mathcal{N}(0, I_d)$

    $X_{t+h} = X_t + h u_t^\theta(X_t) + \sigma_t \sqrt{h}\epsilon$

    Update $t \leftarrow t + h$

end for

return X_1

Reminder: Flow and Diffusion Models

Flow Model

Initialize:

$$X_0 \sim p_{\text{init}},$$

ODE:

$$\mathrm{d}X_t = u_t^\theta(X_t)\mathrm{d}t$$

E.g. Gaussian

Neural network vector field

Diffusion coeff.

Diffusion Model

Initialize:

$$X_0 \sim p_{\text{init}},$$

SDE:

$$\mathrm{d}X_t = u_t^\theta(X_t)\mathrm{d}t + \sigma_t \mathrm{d}W_t$$

To get samples, simulate ODE/SDE from t=0 to t=1 and return X_1

Next step: Training the model

Without training, the model produces "non-sense"→ **We need to train** u_t^θ

Training = Finding parameters θ **such that**

$$X_0 \sim p_{\text{init}}, \quad \mathrm{d}X_t = u_t^\theta(X_t)\mathrm{d}t \quad \xrightarrow{\text{Implies}} \quad X_1 \sim p_{\text{data}}$$

Start with initial distribution

Follow along the vector field

The distribution of the final point = data distribution

Goal of lecture 2 (today) and lecture 3 (tomorrow):

Derive training algorithm

Today's goal: Derive a Training Target

- Typically, we train the model by minimizing a **mean squared error**:

$$\boxed{L(\theta) = \|u_t^\theta(x) - \underbrace{u_t^{\text{target}}}_{\text{Training target}}(x)\|^2}$$

- In regression or classification, the training target is the label.
- Here: No label :(→ We have to **derive a training target**

Today: Derive a formula for the training target: $u_t^{\text{target}}(x)$

Tomorrow: Training algorithm using $u_t^{\text{target}}(x)$

Section 2:

Constructing a Training Target

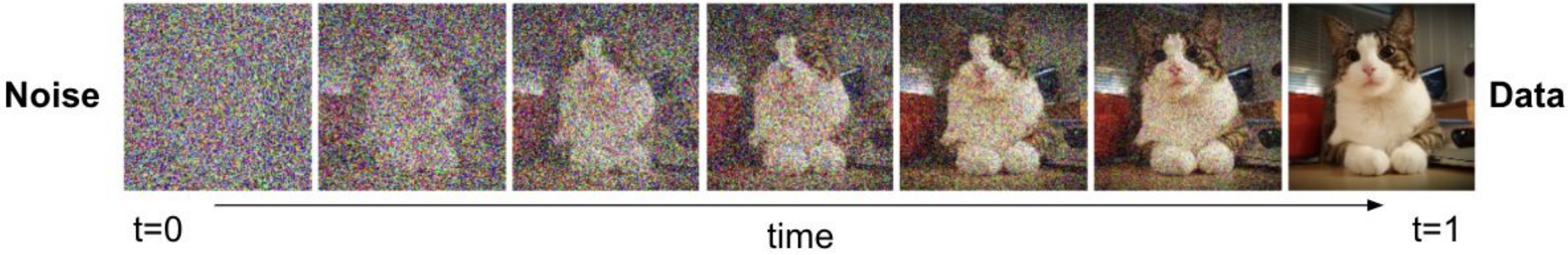
Goal: Derive a formula for a training target for training our models

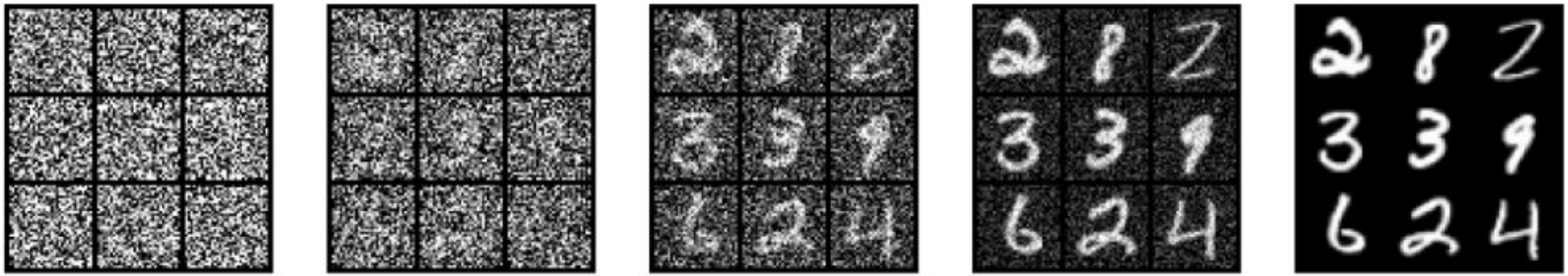
Key terminology:

“Conditional” = “Per single data point”

“Marginal” = “Across distribution of data points”

Probability Paths: The Path from Noise to Data





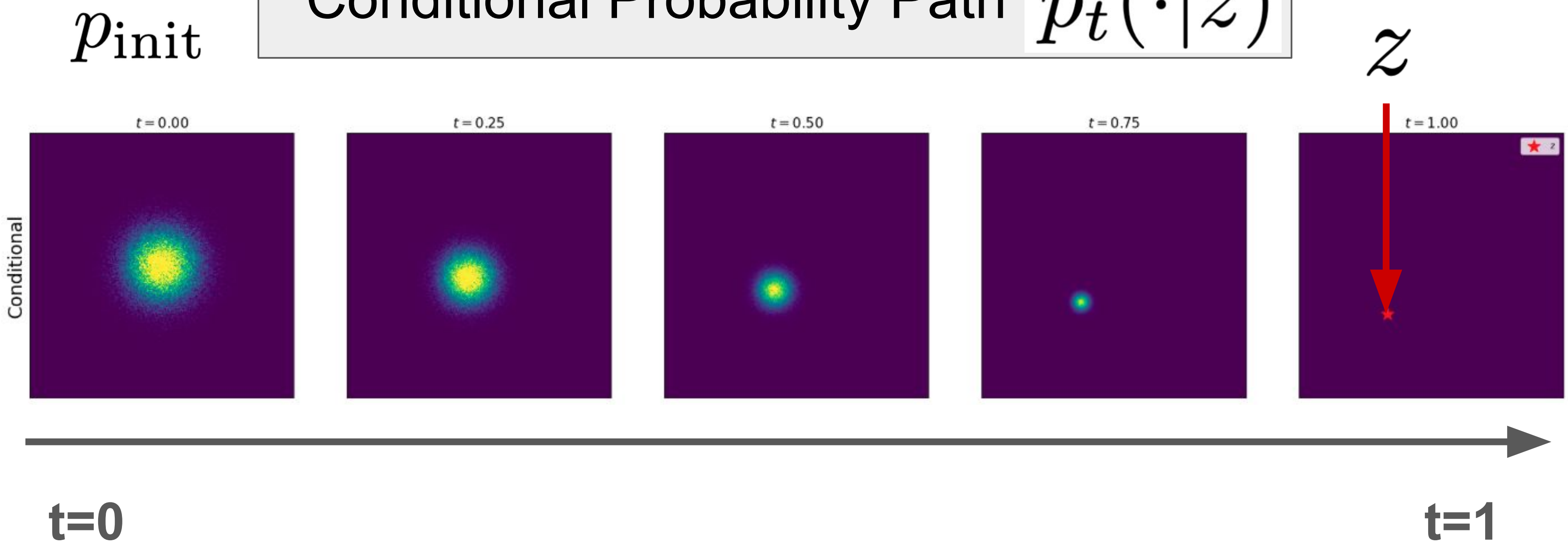
Conditional Prob. Path

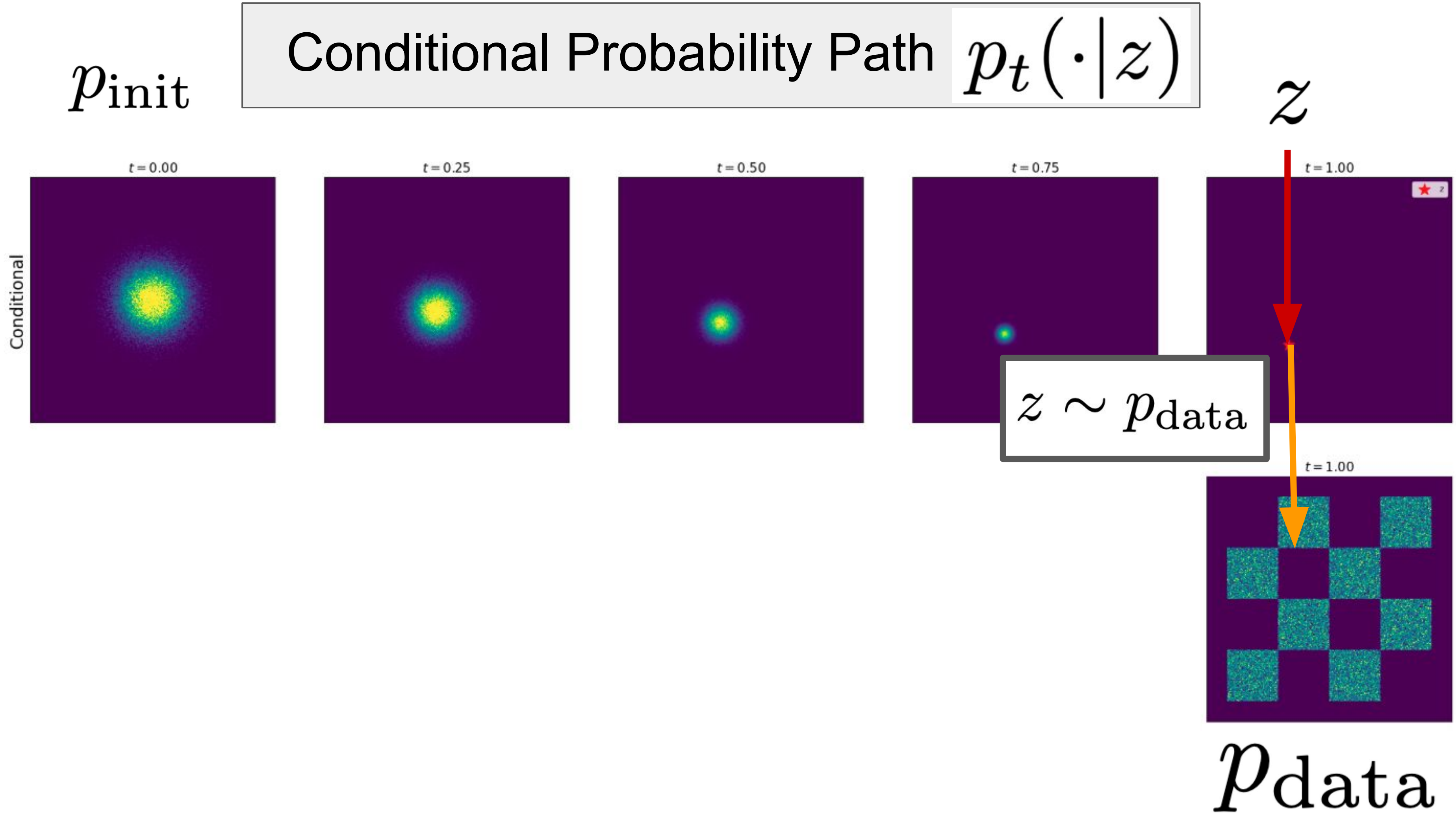
	Notation	Key property	Gaussian example
Conditional Probability Path	$p_t(\cdot\|z)$	Interpolates p_{init} and a data point z	$\mathcal{N}(\alpha_t z, \beta_t^2 I_d)$
Conditional Vector Field			
Conditional Score Function			

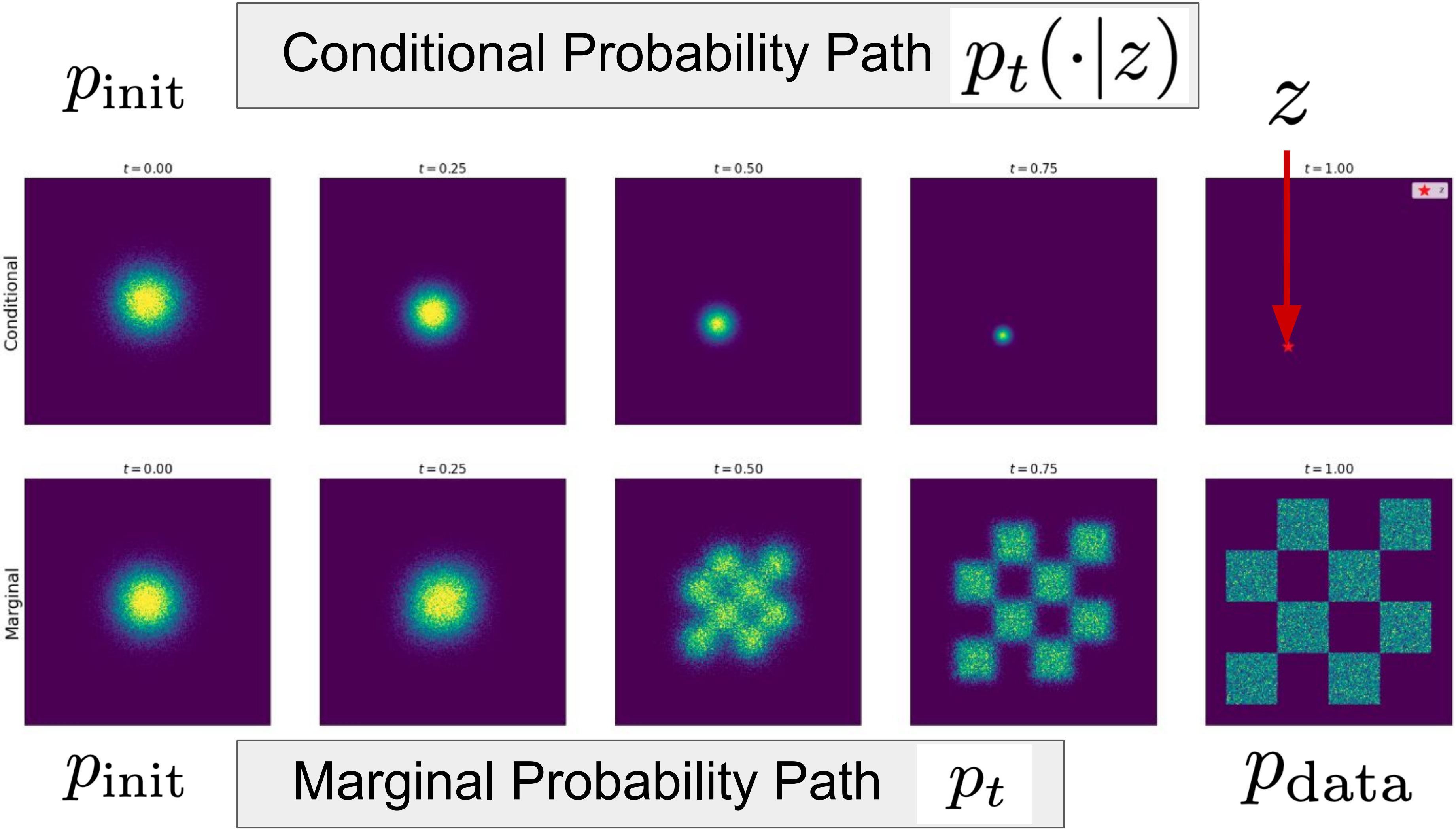
Marginal Prob. Path

	Notation	Key property	Formula
Marginal Probability Path	p_t	Interpolates p_{init} and p_{data}	$\int p_t(x\|z)p_{\mathrm{data}}(z)\mathrm{d}z$
Marginal Vector Field			
Marginal Score Function			

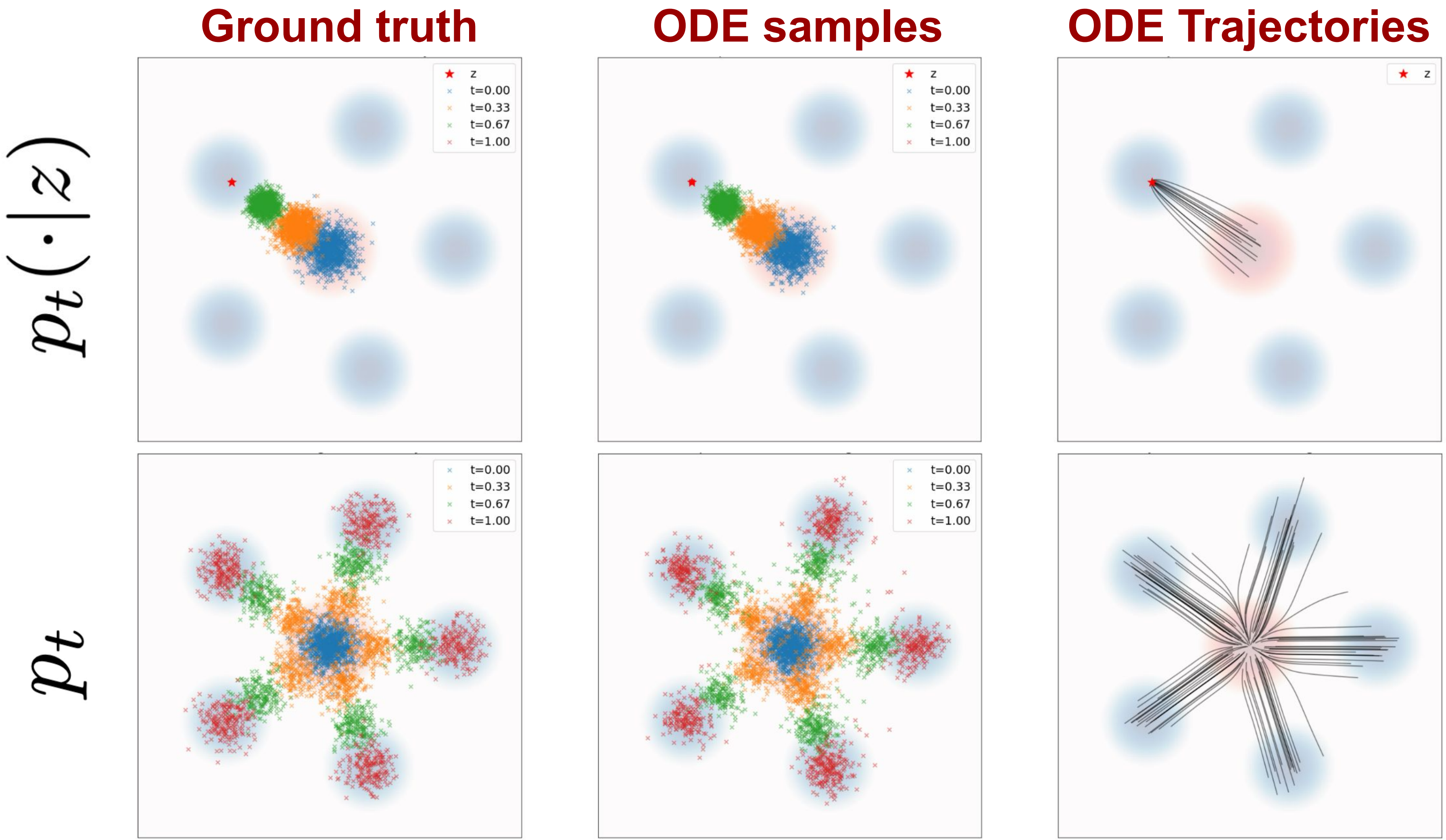
Conditional Probability Path $p_t(\cdot|z)$




Conditional Probability Path $p_t(\cdot|z)$
p_{init}
z
$t=0.00$
$t=0.25$
$t=0.50$
$t=0.75$
$t=1.00$
Conditional
$z \sim p_{\text{data}}$
$t=1.00$
p_{data}



Conditional Probability Path $p_t(\cdot|z)$
p_{init}
z
$t=0.00$
$t=0.25$
$t=0.50$
$t=0.75$
$t=1.00$
Conditional
Marginal
p_{init}
Marginal Probability Path p_t
p_{data}



Ground truth
ODE samples
ODE Trajectories
$p_t(\cdot|z)$
p_t
z
t=0.00
t=0.33
t=0.67
t=1.00

Conditional Prob. Path, Vector Field, and Score

	Notation	Key property	Gaussian example
Conditional Probability Path	$p_t(\cdot\|z)$	Interpolates p_{init} and a data point z	$\mathcal{N}(\alpha_t z, \beta_t^2 I_d)$
Conditional Vector Field	$u_t^{\text{target}}(x\|z)$	ODE follows conditional path	$\left(\dot{\alpha}_t - \frac{\dot{\beta}_t}{\beta_t}\alpha_t\right) z + \frac{\dot{\beta}_t}{\beta_t} x$
Conditional Score Function			

Gaussian Conditional Probability Path And Conditional Vector Field

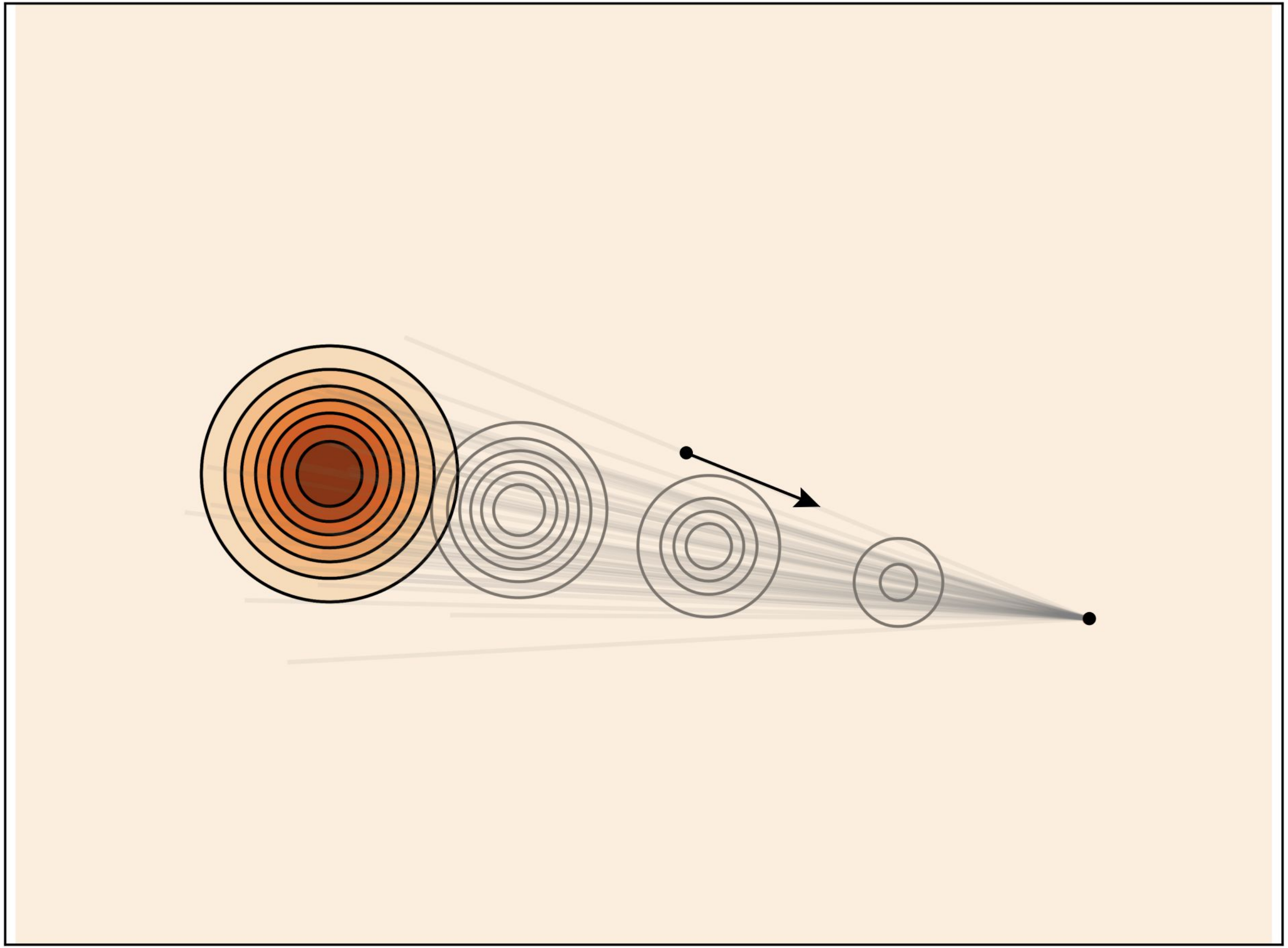


Figure credit: Yaron Lipman