Embodied Vision

Generative Modeling

Tsung-Wei Ke

Spring 2026

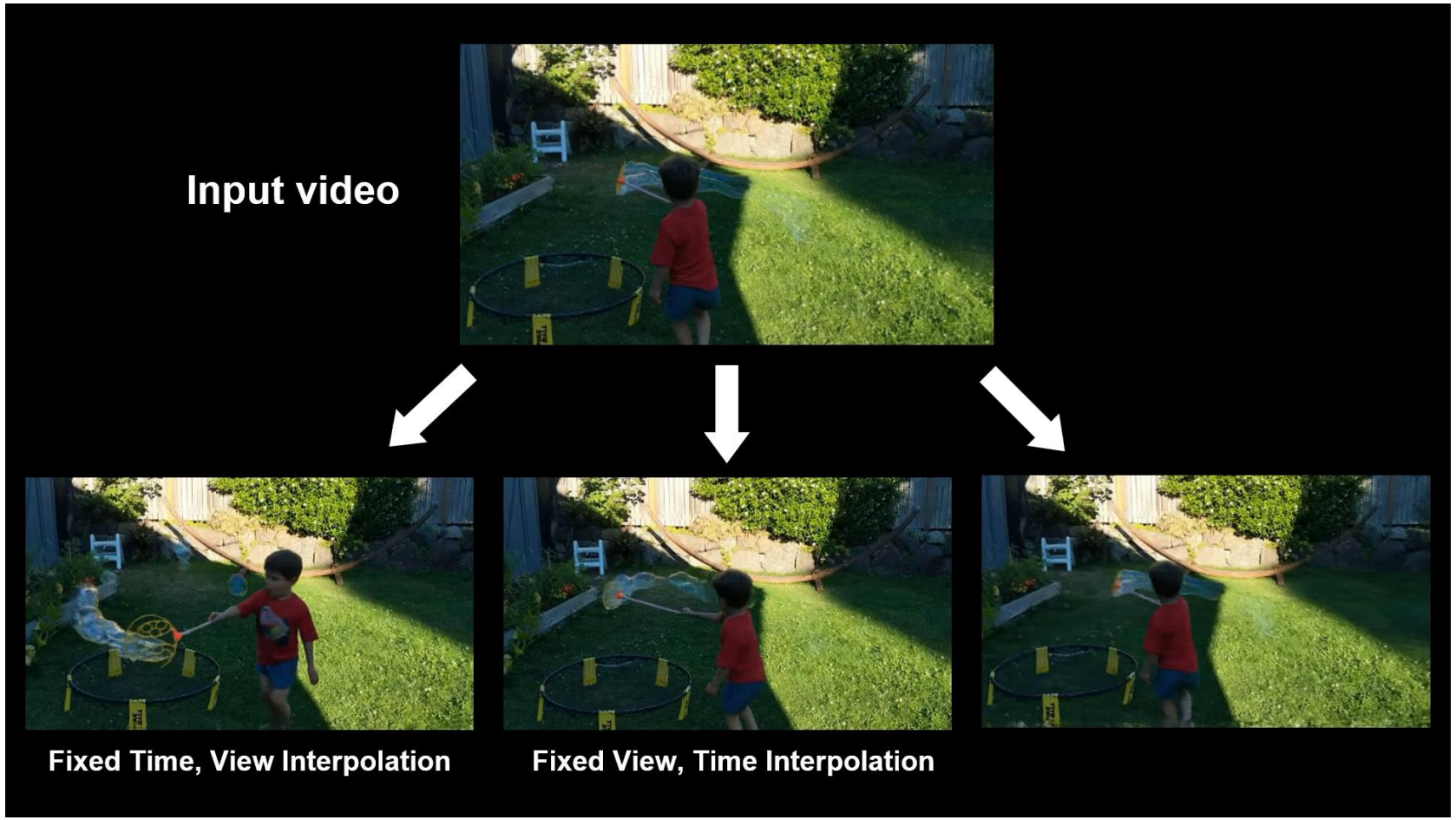
Disclaimer

- This lecture borrows contents heavily from
  - Deep Generative Models by Kaiming He at MIT

Content

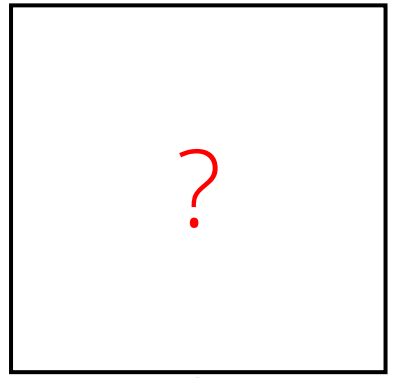
- Generative Modeling
  - Autoregressive Models
  - Variational AutoEncoders
  - Denoising Diffusion Probabilistic Models
  - Flow Matching Models
- 4D Generative Modeling
  - 3D Generation with Multi-view Video Generation
  - 4D Generation with Multi-view Video Generation
  - 3D Geometry Generation

Previously, We Model the 4D World by Observing Monocular Videos



Neural Scene Flow Fields for Space Time View Synthesis of Dynamic Scenes. Li et al.

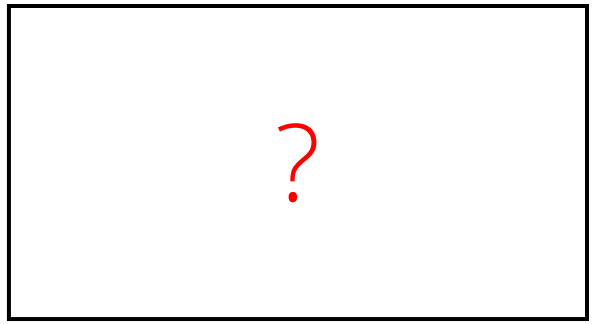
We Inferred the Past Motion of an Observed Object… How About the Future Motion of the Object?



past

now

future



Stereo4D Learning How Things Move in 3D from Internet Stereo Videos . Jin et al.

Unfortunately, the Future is Not Unique. We May Have Multiple Plausible Futures

past

now

future

Nosedive

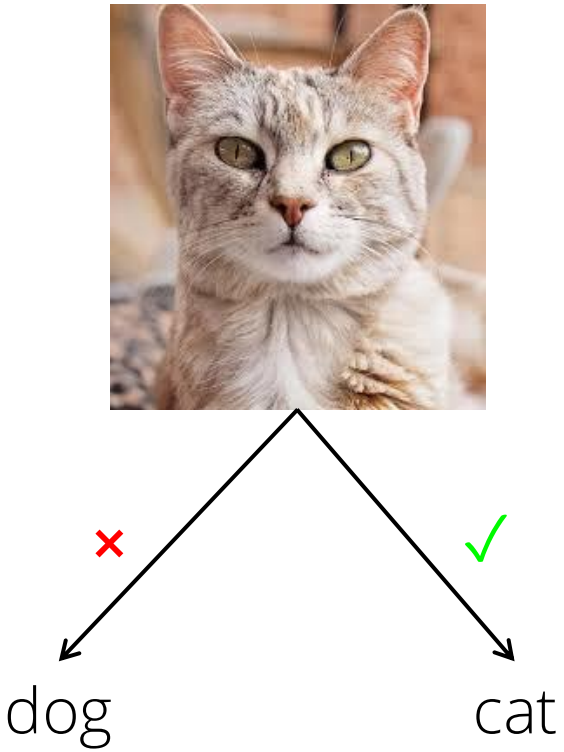
Faceplant

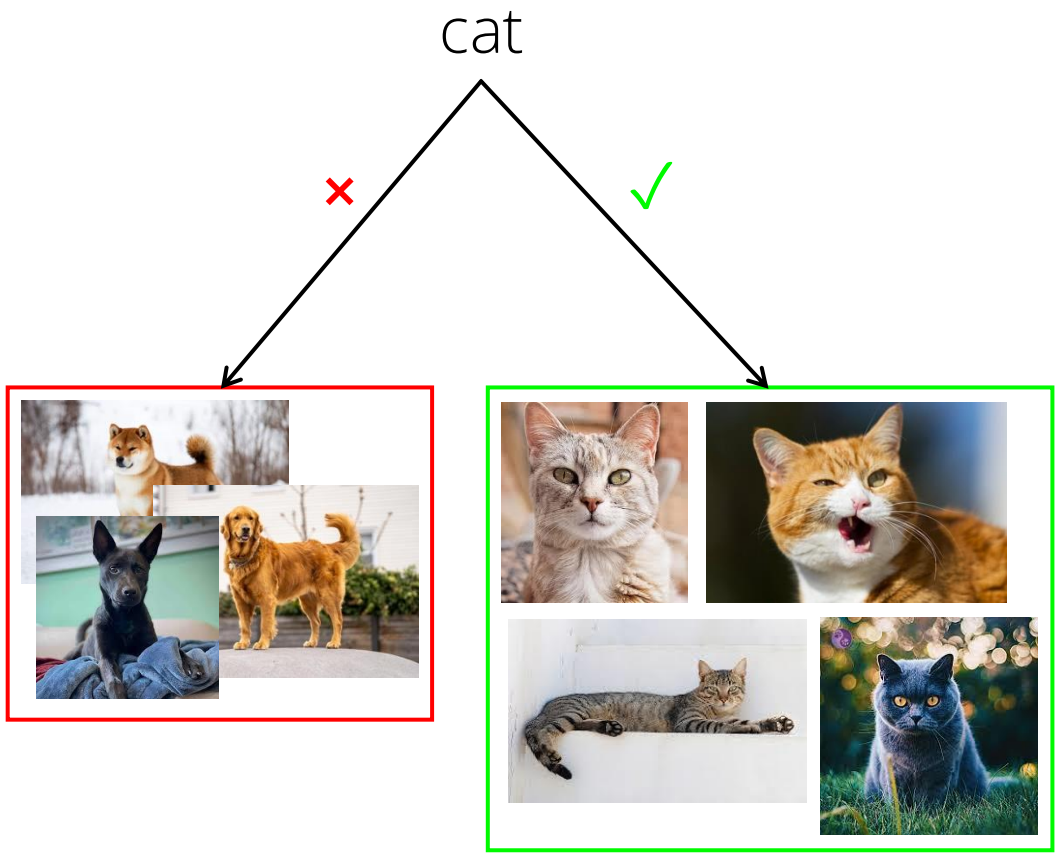
Prat fall

Squat

Some Futures are “More Plausible” than Others

An Analogy of Image Classification and Generation

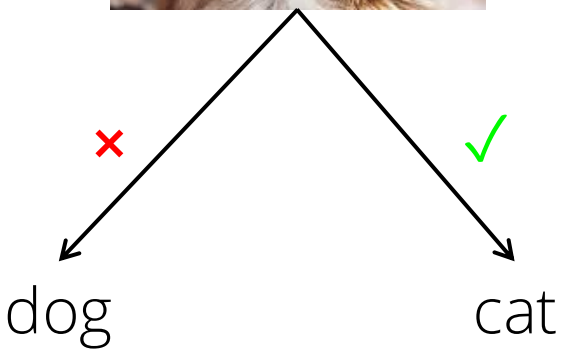




An Analogy of Image Classification and Generation

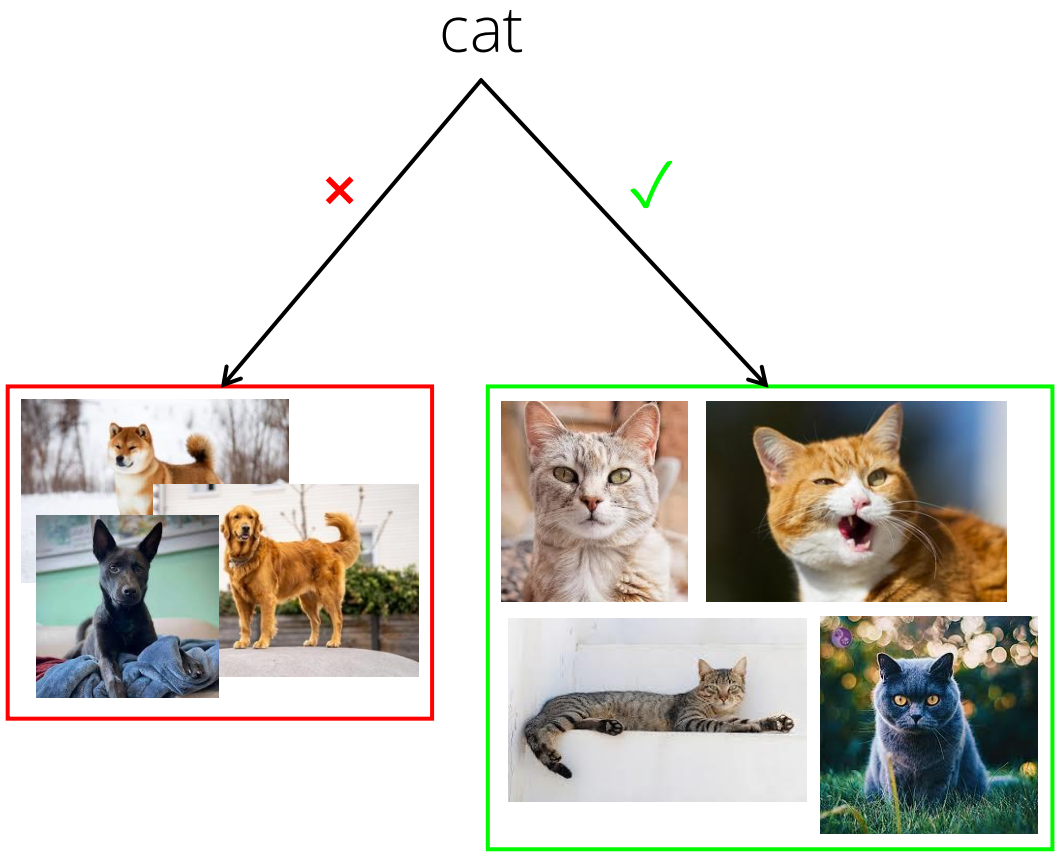
Discriminative

- Sample “x” → Label “y”
- One desired output

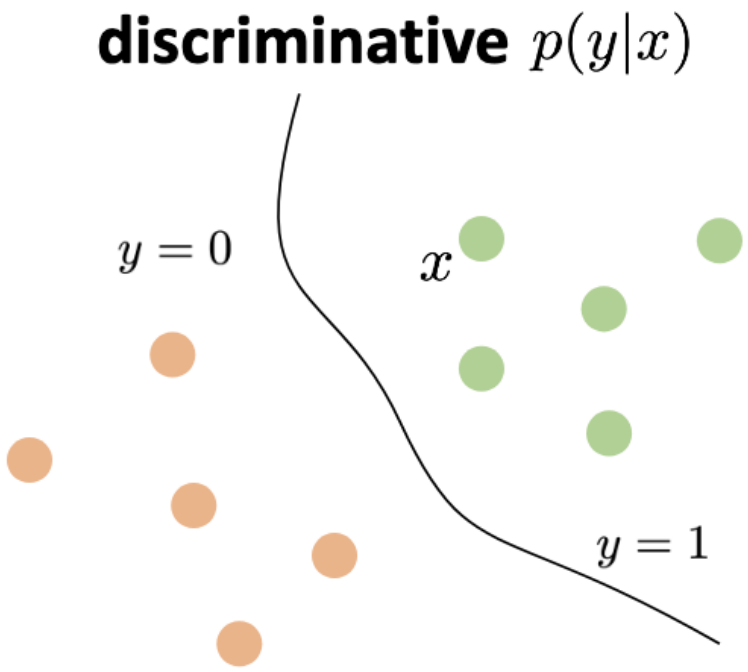


Generative

- Label “y” → Sample “x”
- Multiple possible outputs



Discriminative vs. Generative Models



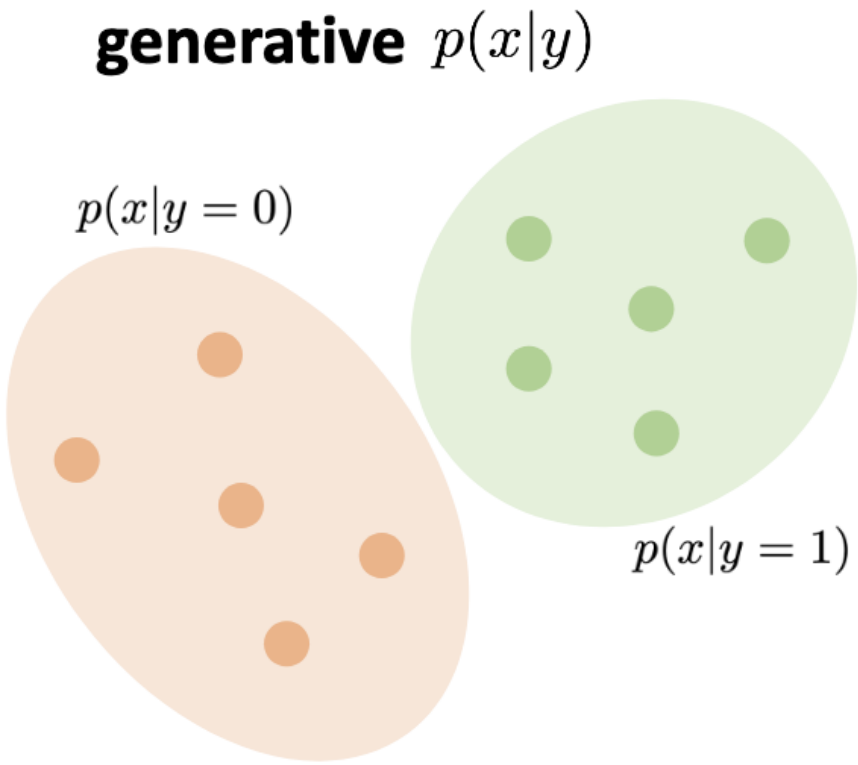


Image source: Kaiming He

Discriminative vs. Generative Models

- Discriminative models to be generative: Baye's rule

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

Prior distribution of label "y" (e.g. uniform distribution)

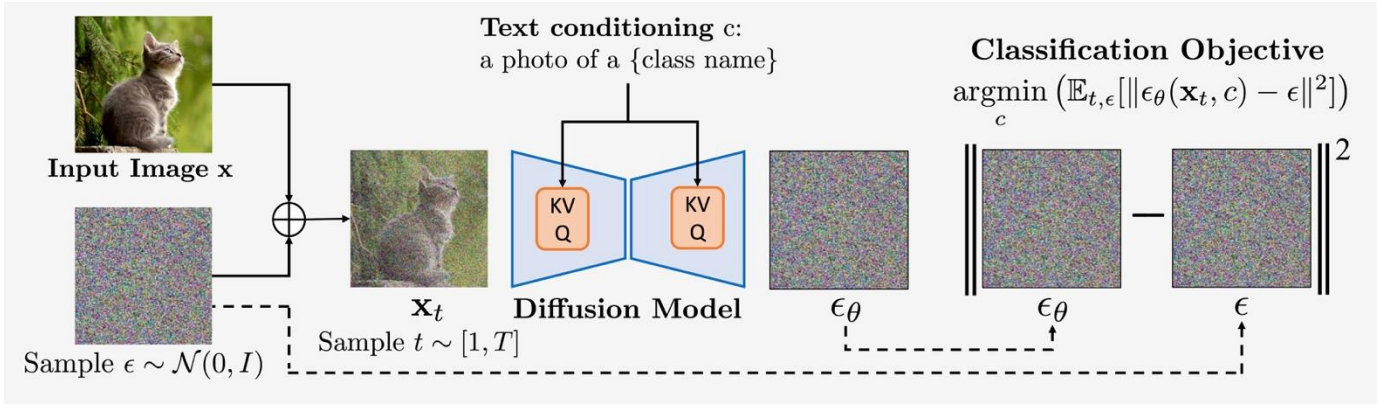
Prior distribution of sample "x" (e.g. constant for every "x")

- Generative models to be discriminative:

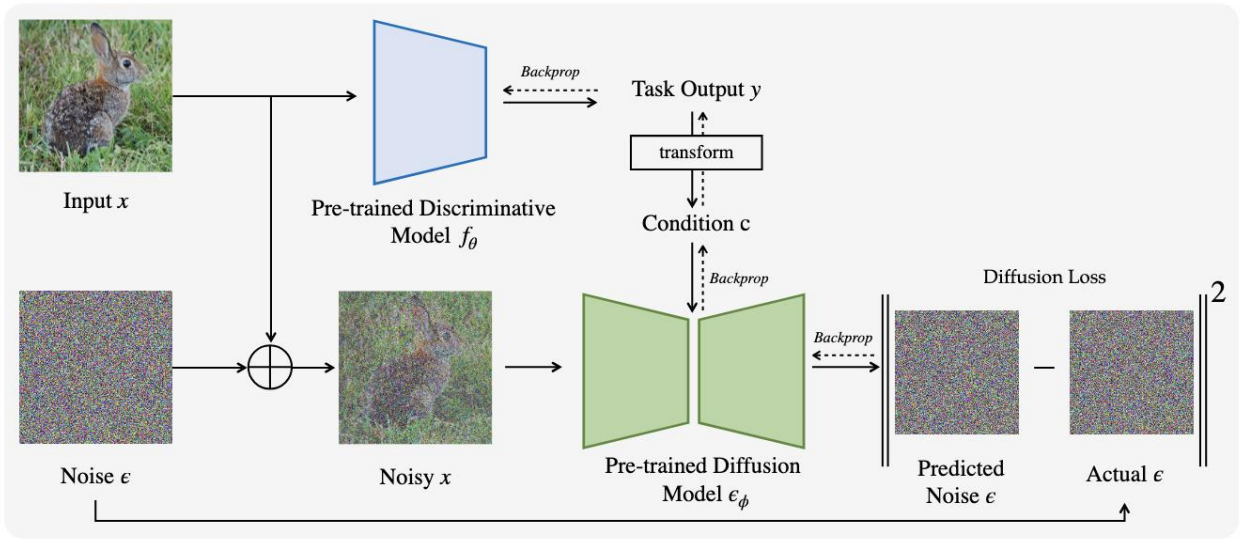
$$p(x|y) = p(y|x)\frac{p(x)}{p(y)}$$

Slide adapted from Kaiming He

Generative Models are Implicitly Discriminative Models

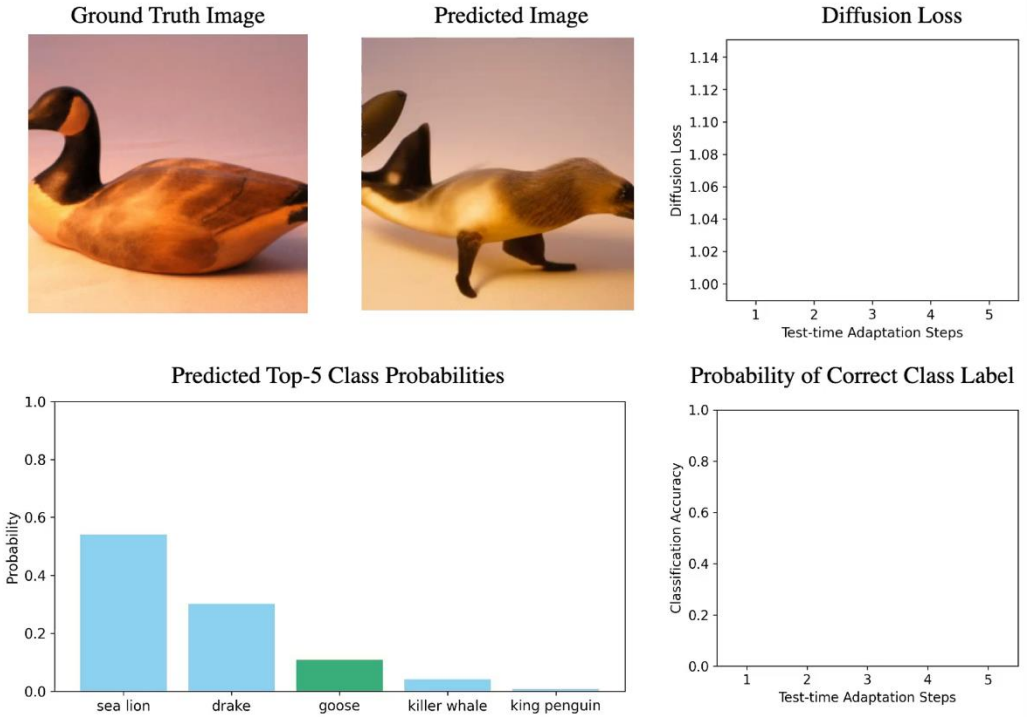


Your Diffusion Model is Secretly a Zero-Shot Classifier. Li et al.



Diffusion-TTA: Test-time Adaptation of Discriminative Models via Generative Feedback. Prabhudesai et al.

Improve pre-trained image classifiers by minimizing diffusion loss at test-time



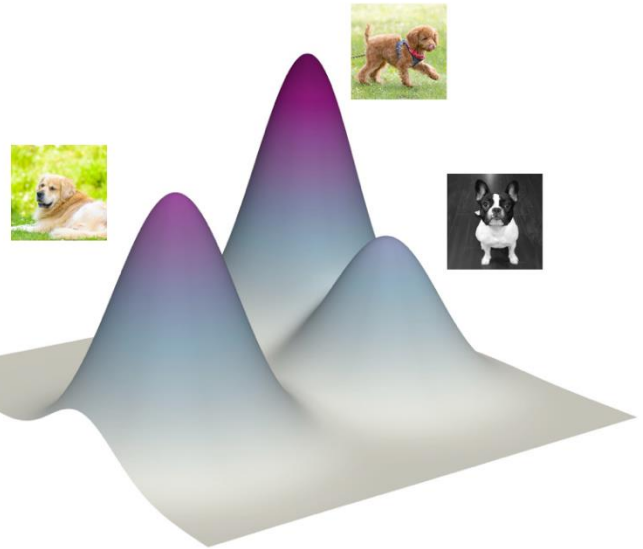
Generative models w/ probabilistic modeling

data

Slide credit Kaiming He

Generative models w/ probabilistic modeling

data



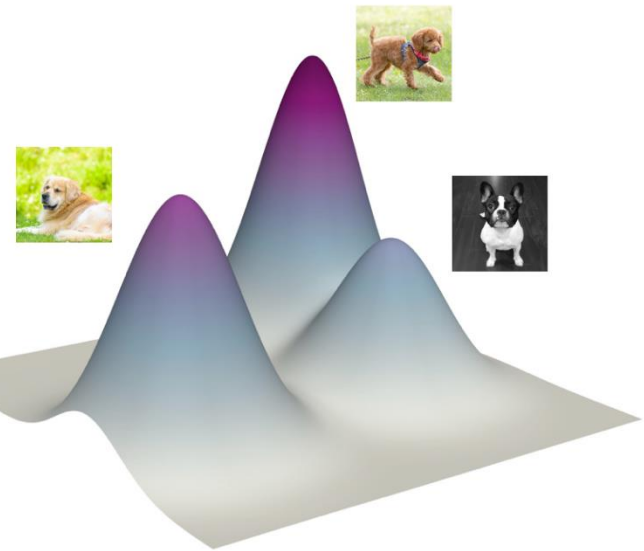
distribution of data

- Capture the probabilistic distribution of training with the model

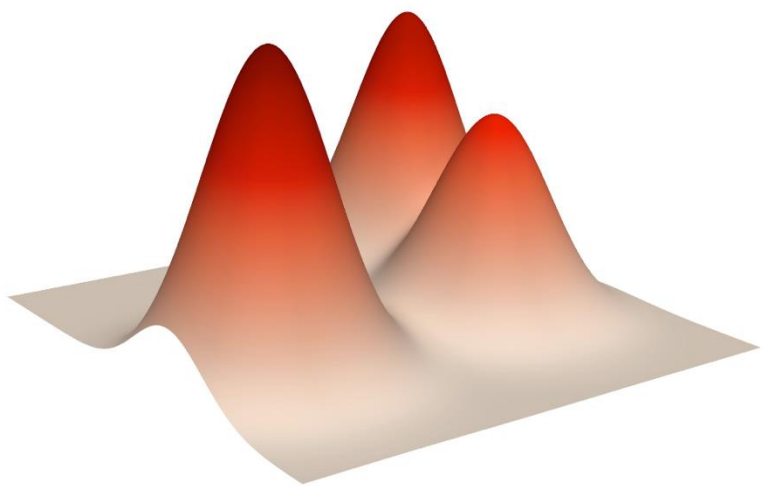
Slide credit Kaiming He

Generative models w/ probabilistic modeling

data



distribution of data



estimated distribution of data

- Optimize a loss function

$$\mathcal{L}\left(\quad , \quad \right)$$

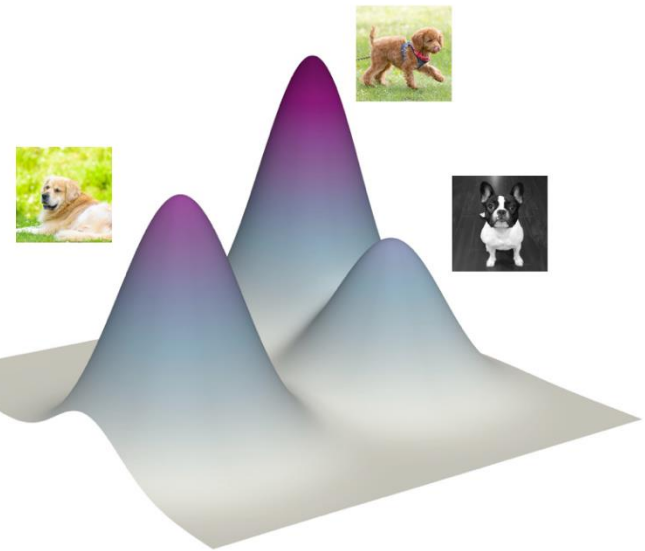
Slide credit Kaiming He

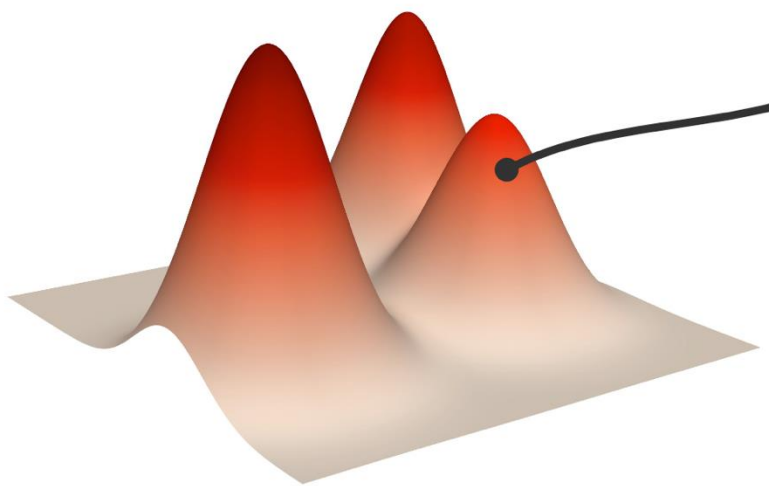
Generative models w/ probabilistic modeling

data

distribution of data

estimated distribution of data





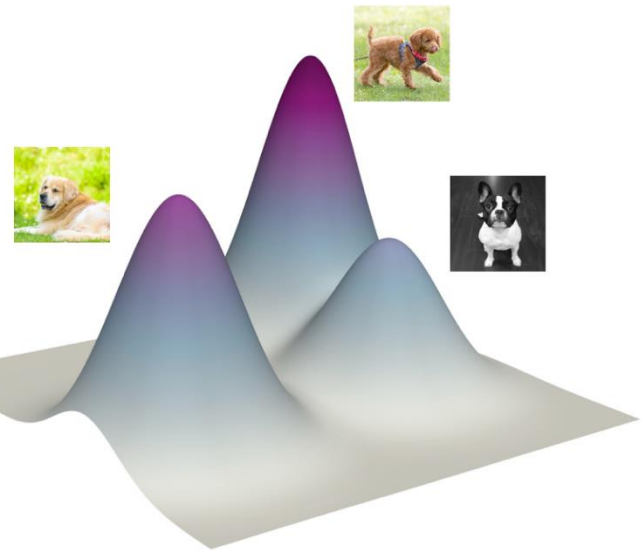
sample new “data”

$$x' \sim p(x|y)$$

Slide credit Kaiming He

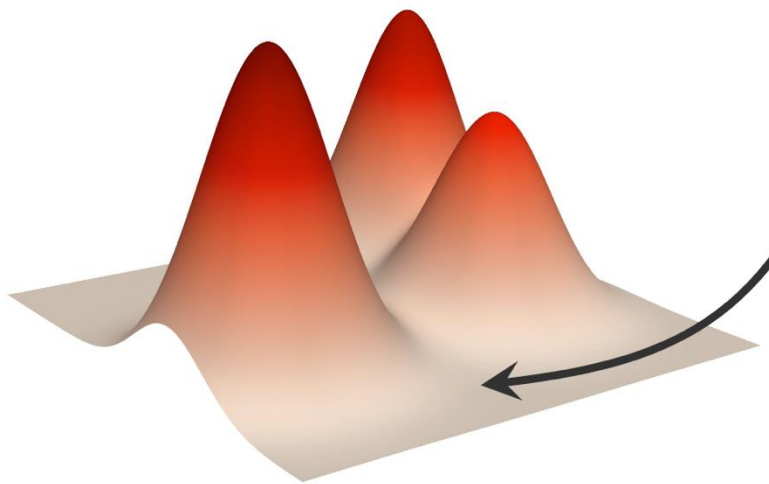
Generative models w/ probabilistic modeling

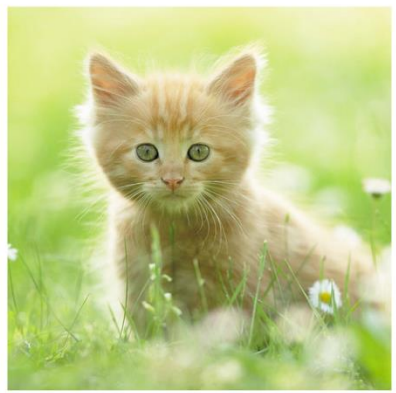
data



distribution of data

estimated distribution of data





estimate prob density

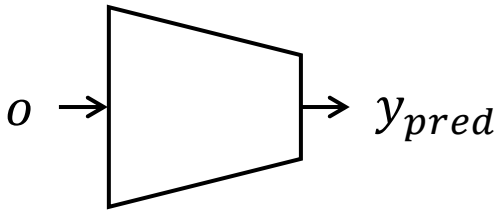
$p(x|y)$ = ?

Slide credit Kaiming He

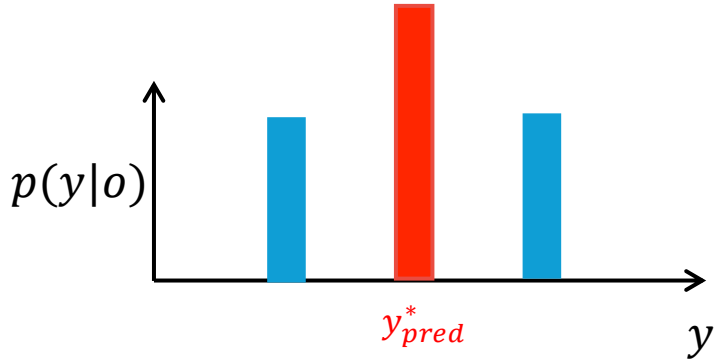
Probabilistic Models Differ From Regression Models

Deterministic

- predicts the same output given the same input

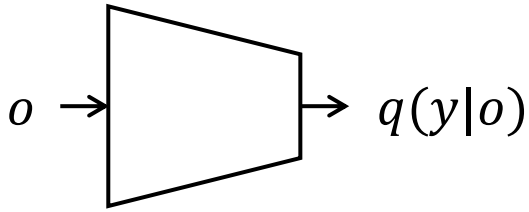


$$y^*_{pred} = \underset{y_{pred}}{\text{argmin}} \frac{1}{N} \Sigma^N_{n=1} |y_{pred} - y_n|^2$$

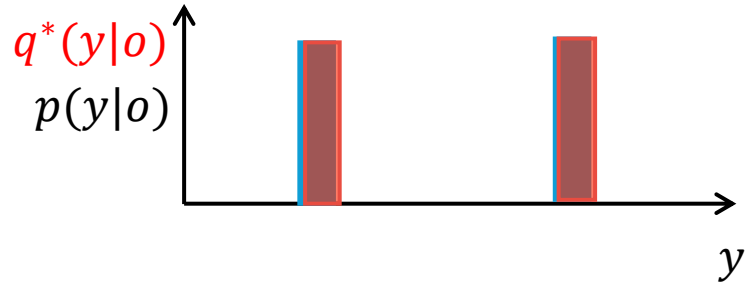


Probabilistic

- predicts the same likelihood of all output given the same input



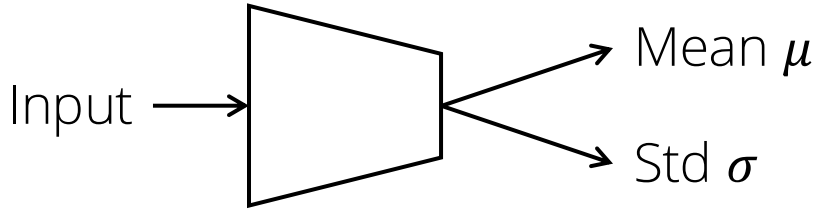
$$q^*(y|o) = \underset{q^*}{\text{argmin}}\, D_{KL}(q(y|o), p(y|o))$$

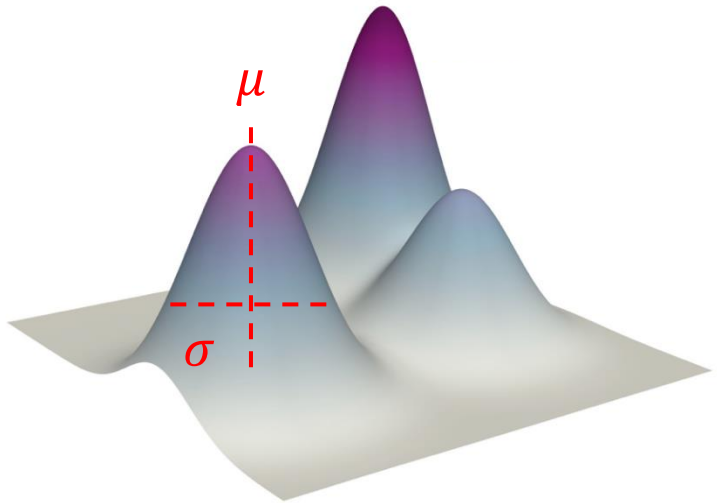


How to Model Probabilistic Distribution?

Gaussian

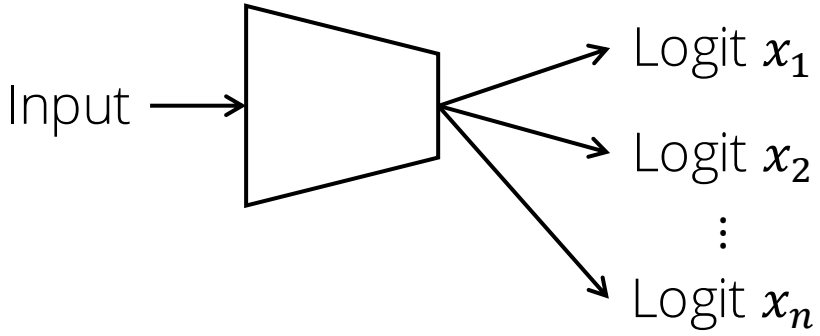
- Predict the mean and standard deviation to model the probability of sample "x"
- VAE, Diffusion models etc

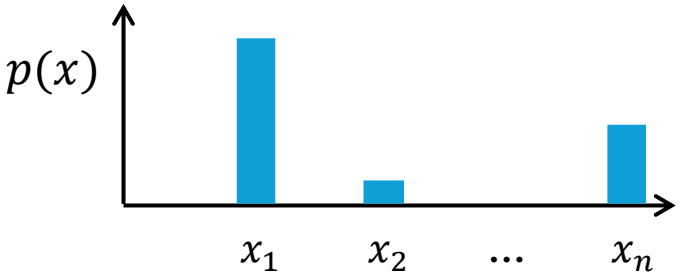




Categorical

- Predict the categorical probability of sample "x"
- Most LLMs etc





Content

- Generative Modeling
  - ➢Autoregressive Models
  - ➢Variational AutoEncoders
  - ➢Denoising Diffusion Probabilistic Models
  - ➢Flow Matching Models
- 4D Generative Modeling
  - ➢3D Generation with Multi-view Video Generation
  - ➢4D Generation with Multi-view Video Generation
  - ➢3D Geometry Generation

What are Common Generative Models?

Generation Can be Considered as Modeling Joint Distribution

Once | upon | a | time | ...

- Text generation:

$$\mathbf{p(word1 = Once, word2 = upon)}$$

- Image generation:

$$\mathbf{p(pixel1 = red, pixel2 = green)}$$

- Joint distribution of data:

$$\mathbf{p(x1 = X1, x2 = X2)}$$

  - ➢ How to model this?

Generation Can be Considered as Modeling Joint Distribution

Once | upon | a | time | ...

- Text generation:

$$\mathbf{p(word1 = Once, word2 = upon)}$$

- Image generation:

$$\mathbf{p(pixel1 = red, pixel2 = green)}$$

- Joint distribution of data:

$$\mathbf{p(x1 = X1, x2 = X2)}$$

  - Modeling as independent distributions

$$\mathbf{p(x1, x2) \not\rightarrow p(x1)p(x2)}$$

  - Modeling as conditional distributions

$$\mathbf{p(x1, x2) \rightarrow p(x1)p(x2|x1)}$$

Conditional Distribution Modeling

Chain rule:

Any joint distribution can be written as a product of conditionals

$$p(A, B, C) = p(A)p(B \mid A)p(C \mid A, B)$$

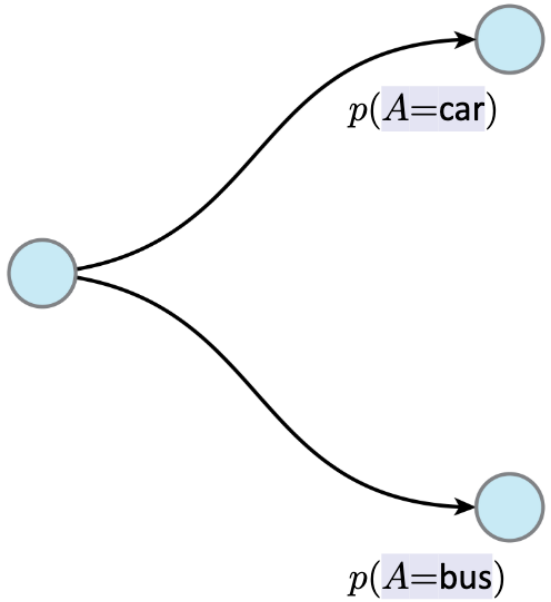
Slide credit Kaiming He

Conditional Distribution Modeling

Chain rule:

Any joint distribution can be written as a product of conditionals

$$p(A, B, C) = p(A)p(B \mid A)p(C \mid A, B)$$



*example: binary variables

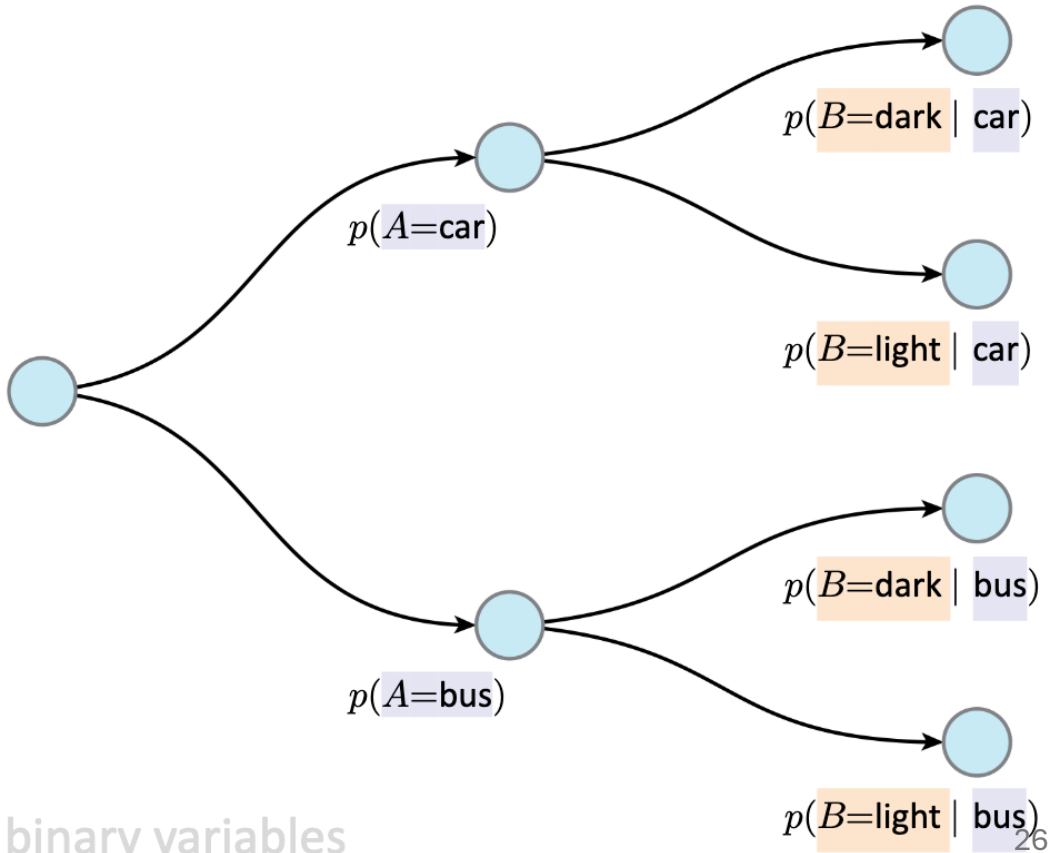
Slide credit Kaiming He

Conditional Distribution Modeling

Chain rule:

Any joint distribution can be written as a product of conditionals

$$p(A, B, C) = p(A)p(B \mid A)p(C \mid A, B)$$



*example: binary variables

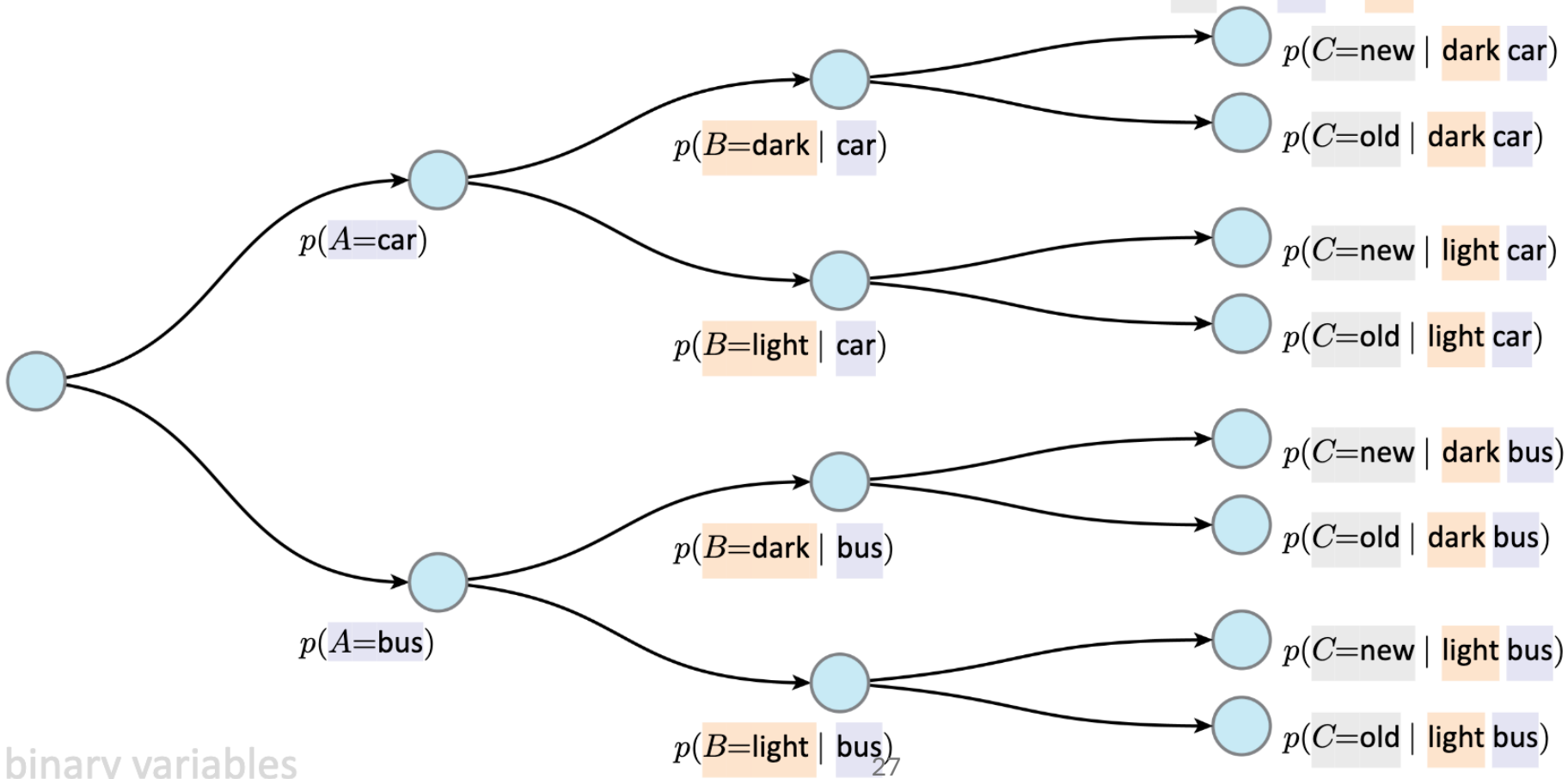
Slide credit Kaiming He

Conditional Distribution Modeling

Chain rule:

Any joint distribution can be written as a product of conditionals

$$p(A, B, C) = p(A)p(B \mid A)p(C \mid A, B)$$



*example: binary variables

Slide credit Kaiming He

Conditional Distribution Modeling

Chain rule:

Any joint distribution can be written as a product of conditionals

in any <u>order</u>:

$$
\begin{aligned}
p(A, B, C) &= p(A)p(B \mid A)p(C \mid A, B) \\
&= p(A)p(C \mid A)p(B \mid A, C) \\
&= p(B)p(A \mid B)p(C \mid A, B) \\
&= p(B)p(C \mid B)p(A \mid B, C) \\
&= p(C)p(A \mid C)p(B \mid A, C) \\
&= p(C)p(B \mid C)p(A \mid B, C)
\end{aligned}
$$

* This is a foundation of Masked Autoregressive (MAR) models:
Li, et al. Autoregressive Image Generation without Vector Quantization, 2024

Slide credit Kaiming He

Conditional Distribution Modeling

Chain rule:

Any joint distribution can be written as a product of conditionals

in any <u>partition</u>:

$$
\begin{aligned}
p(A, B, C, D) &= p(A, B)p(C, D \mid A, B) \\
&= p(C, D)p(A, B \mid C, D) \\
&= p(A, B, C)p(D \mid A, B, C) \\
&= \dots
\end{aligned}
$$

* This is a foundation of Masked Autoregressive (MAR) models:
Li, et al. Autoregressive Image Generation without Vector Quantization, 2024

Slide credit Kaiming He

Categorize Generation Models by the Way of Modeling Conditional Distributions

- Condition on data partitions: Autoregressive models

$$\mathrm{p}(\mathrm{x}_1, \mathrm{x}_2, \cdots, \mathrm{x}_\mathrm{n}) = \mathrm{p}(\mathrm{x}_1)\mathrm{p}(\mathrm{x}_2|\mathrm{x}_1) \cdots \mathrm{p}(\mathrm{x}_\mathrm{n}|\mathrm{x}_1, \mathrm{x}_2, \cdots, \mathrm{x}_{\mathrm{n}-1})$$

- Condition on latent variables: VAE, Diffusion models

$$\mathrm{p}(\mathrm{x}_1, \mathrm{x}_2, \cdots, \mathrm{x}_\mathrm{n}) = \mathrm{p}(\boldsymbol{z})\mathrm{p}(\mathrm{x}_1, \mathrm{x}_2, \cdots, \mathrm{x}_\mathrm{n}|\boldsymbol{z})$$

with $\mathrm{p}(\boldsymbol{z}) = \mathrm{p}(\mathrm{z}_1)\mathrm{p}(\mathrm{z}_2|\mathrm{z}_1) \cdots \mathrm{p}(\mathrm{z}_\mathrm{n}|\mathrm{z}_1, \mathrm{z}_2, \cdots, \mathrm{z}_{\mathrm{n}-1})$

Content

- Generative Modeling
  - ➢Autoregressive Models
  - ➢Variational AutoEncoders
  - ➢Denoising Diffusion Probabilistic Models
  - ➢Flow Matching Models
- 4D Generative Modeling
  - ➢3D Generation with Multi-view Video Generation
  - ➢4D Generation with Multi-view Video Generation
  - ➢3D Geometry Generation

AutoRegressive Models

- Condition on data partitions:

$$p(x_1, x_2, \cdots, x_n) = p(x_1)p(x_2|x_1) \cdots p(x_n|x_1, x_2, \cdots, x_{n-1})$$

AutoRegressive Models

- Condition on data partitions:

$$\mathrm{p}(\mathrm{x}_1, \mathrm{x}_2, \cdots, \mathrm{x}_\mathrm{n}) = \mathrm{p}(\mathrm{x}_1)\mathrm{p}(\mathrm{x}_2|\mathrm{x}_1) \cdots \mathrm{p}(\mathrm{x}_\mathrm{n}|\mathrm{x}_1, \mathrm{x}_2, \cdots, \mathrm{x}_{\mathrm{n}-1})$$

$$= p_\theta(\mathrm{x}_1)p_\theta(\mathrm{x}_2|\mathrm{x}_1) \cdots p_\theta(\mathrm{x}_\mathrm{n}|\mathrm{x}_1, \mathrm{x}_2, \cdots, \mathrm{x}_{\mathrm{n}-1})$$

An inductive bias that shares models among different mapping

AutoRegressive Models

- Condition on data partitions:

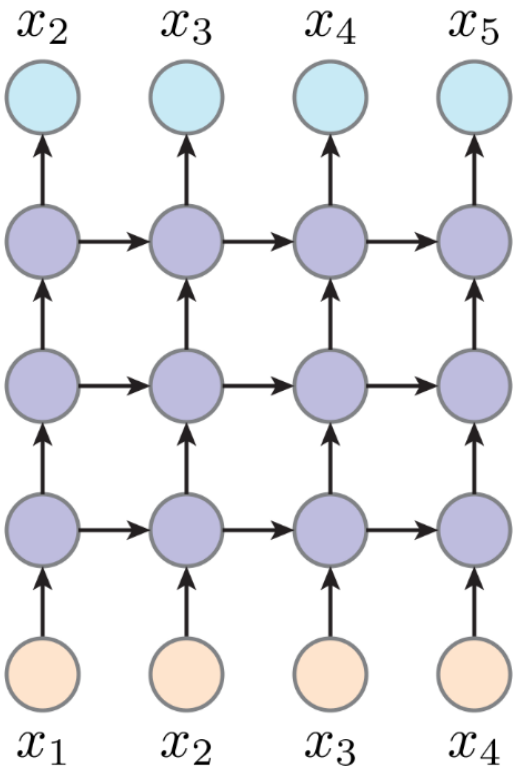
$$\mathrm{p}(\mathrm{x}_1, \mathrm{x}_2, \cdots, \mathrm{x}_\mathrm{n}) = \mathrm{p}(\mathrm{x}_1)\mathrm{p}(\mathrm{x}_2|\mathrm{x}_1) \cdots \mathrm{p}(\mathrm{x}_\mathrm{n}|\mathrm{x}_1, \mathrm{x}_2, \cdots, \mathrm{x}_{\mathrm{n}-1})$$

$$= p_\theta(\mathrm{x}_1)p_\theta(\mathrm{x}_2|\mathrm{x}_1) \cdots p_\theta(\mathrm{x}_\mathrm{n}|\mathrm{x}_1, \mathrm{x}_2, \cdots, \mathrm{x}_{\mathrm{n}-1})$$

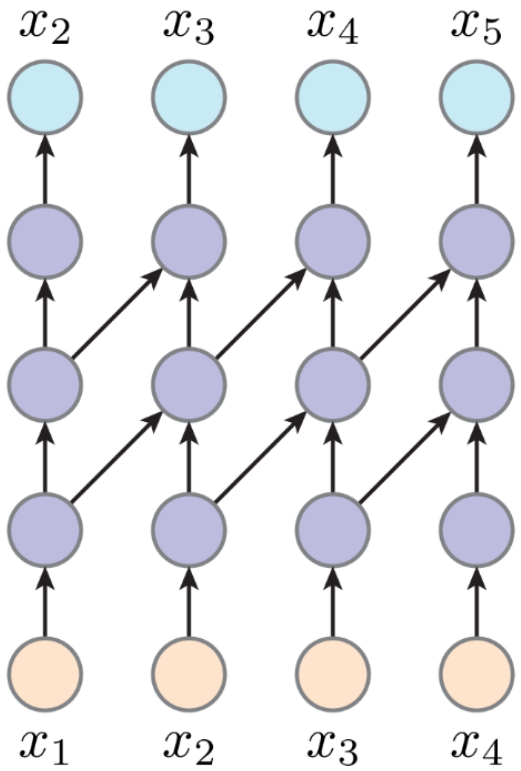
An inductive bias that shares models among different mapping

What kind of model architectures can handle these different mappings?

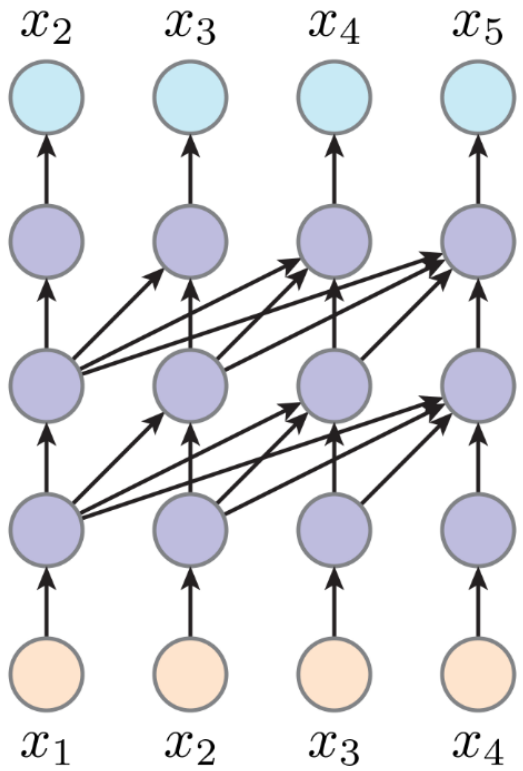
Common Architectures for Autoregression



RNN



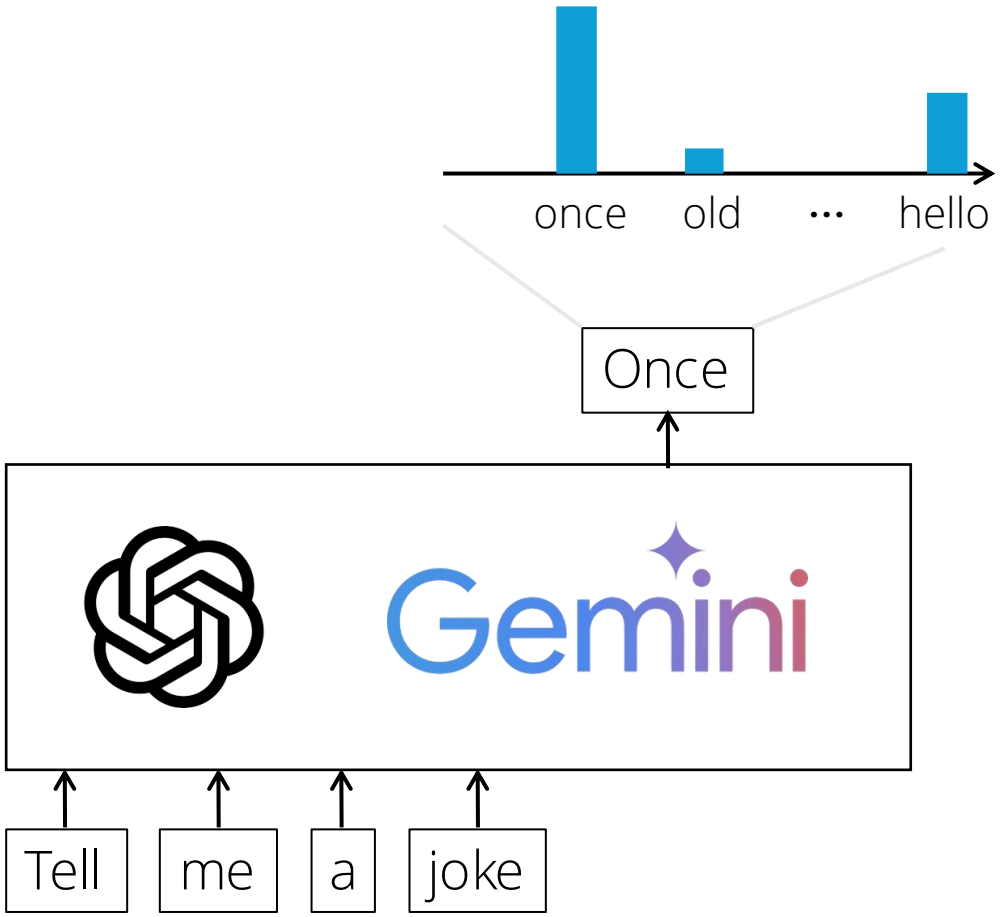
CNN

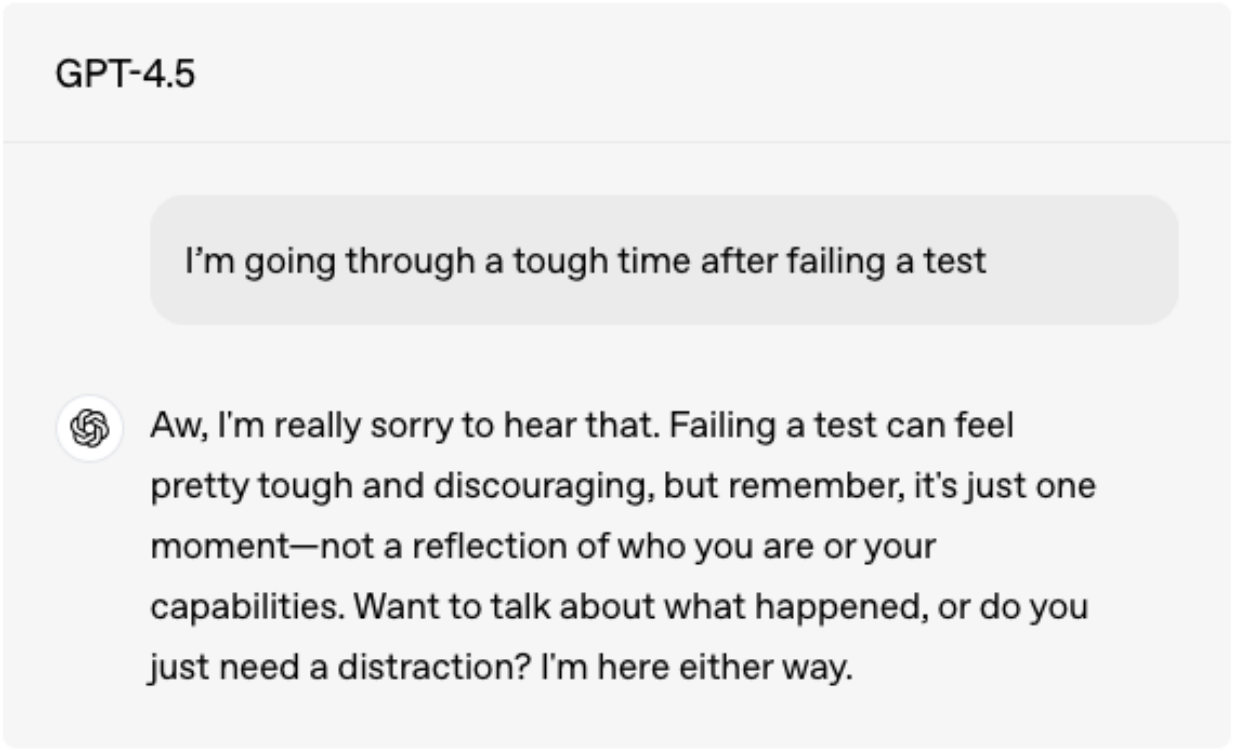


Attention

Slide credit Kaiming He

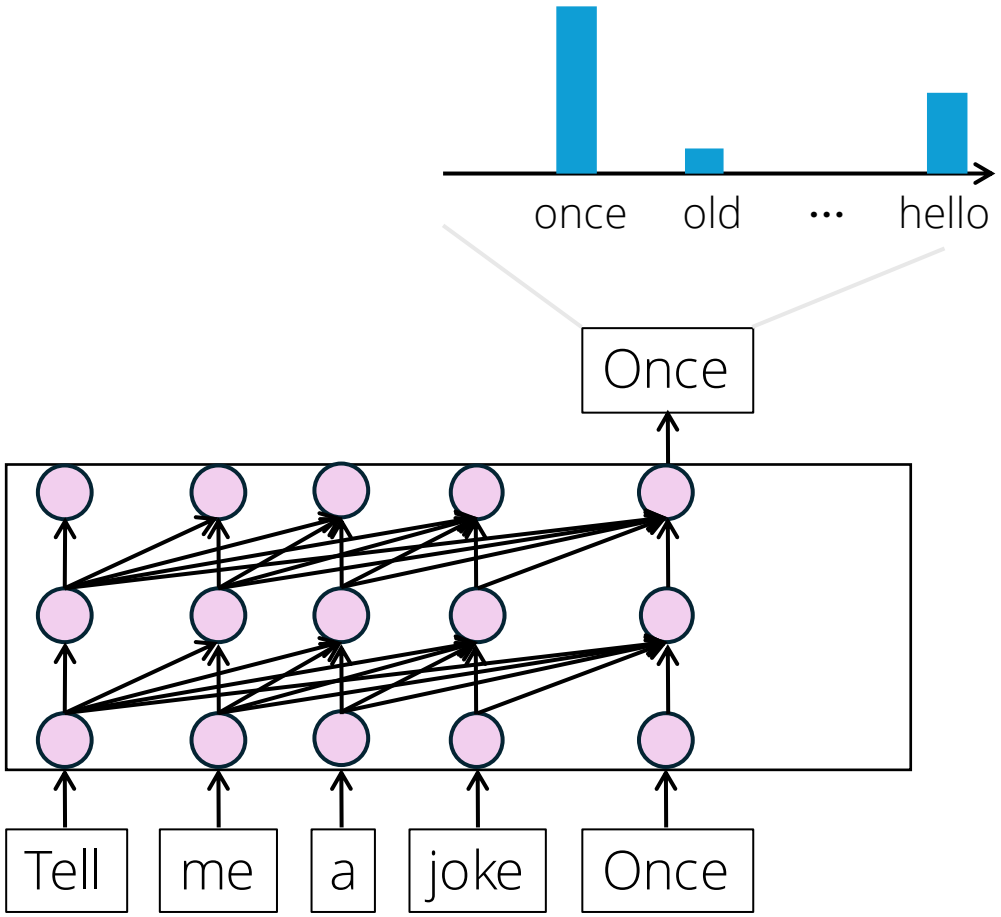
Discrete Autoregressive Transformer are Categorical Probabilistic Generative Models





https://openai.com/index/introducing-gpt-4-5/

Discrete Autoregressive Transformer are Categorical Probabilistic Generative Models



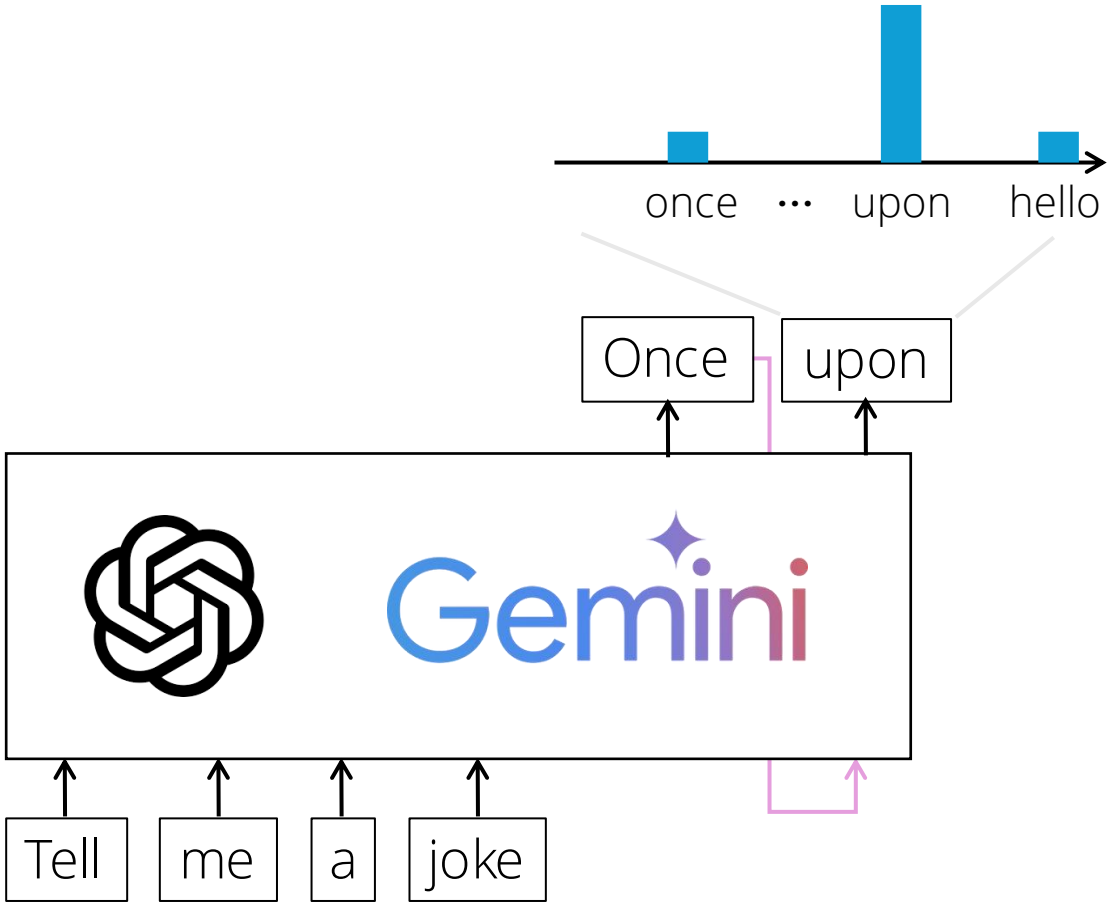
GPT-4.5

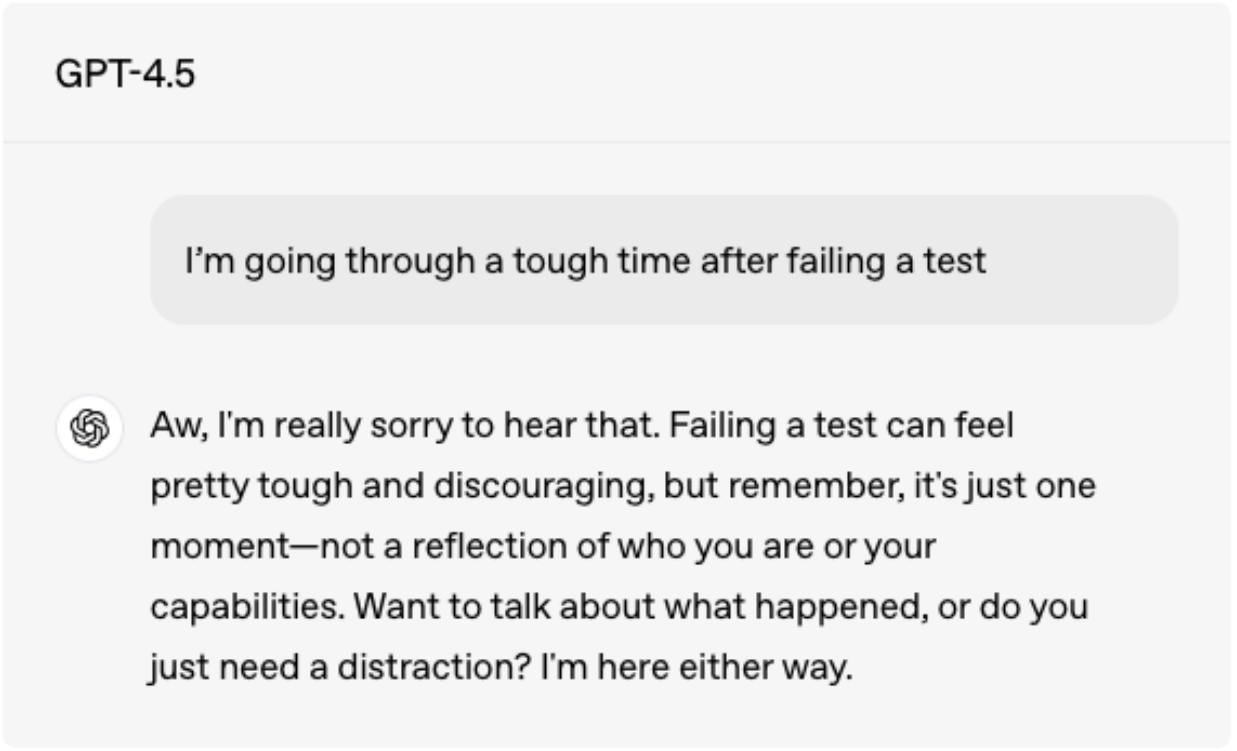
I'm going through a tough time after failing a test

Aw, I'm really sorry to hear that. Failing a test can feel pretty tough and discouraging, but remember, it's just one moment—not a reflection of who you are or your capabilities. Want to talk about what happened, or do you just need a distraction? I'm here either way.

https://openai.com/index/introducing-gpt-4-5/

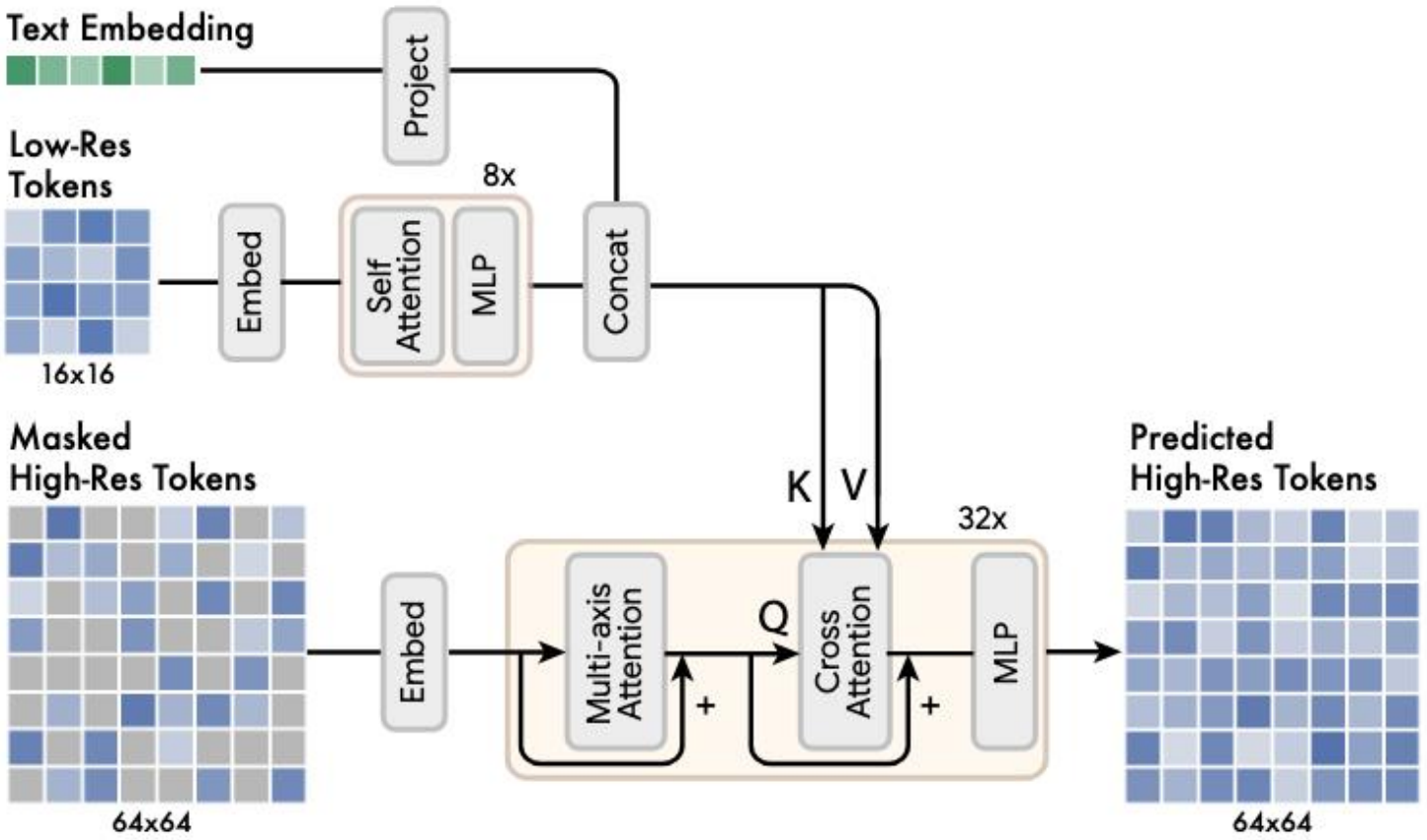
Discrete Autoregressive Transformer are Categorical Probabilistic Generative Models





https://openai.com/index/introducing-gpt-4-5/

Discrete Autoregressive Transformer Can Also Generate Images / Videos



Muse: Text-To-Image Generation via Masked Generative Transformers. Chang et al.

Content

- Generative Modeling
  - ➢Autoregressive Models
  - ➢Variational AutoEncoders
  - ➢Denoising Diffusion Probabilistic Models
  - ➢Flow Matching Models
- 4D Generative Modeling
  - ➢3D Generation with Multi-view Video Generation
  - ➢4D Generation with Multi-view Video Generation
  - ➢3D Geometry Generation

Another Way of Modeling Conditional Distributions

- Condition on latent variables:

$$\mathrm{p}(\mathrm{x}_1, \mathrm{x}_2, \cdots, \mathrm{x}_\mathrm{n}) = \mathrm{p}(\boldsymbol{z})\mathrm{p}(\mathrm{x}_1, \mathrm{x}_2, \cdots, \mathrm{x}_\mathrm{n}|\boldsymbol{z})$$

You can select a random distribution for variable $\mathbf{z}$, such as Gaussian.

In fact, Gaussian prior results in minimal variance if the data distribution is unknown [TODO; citation]

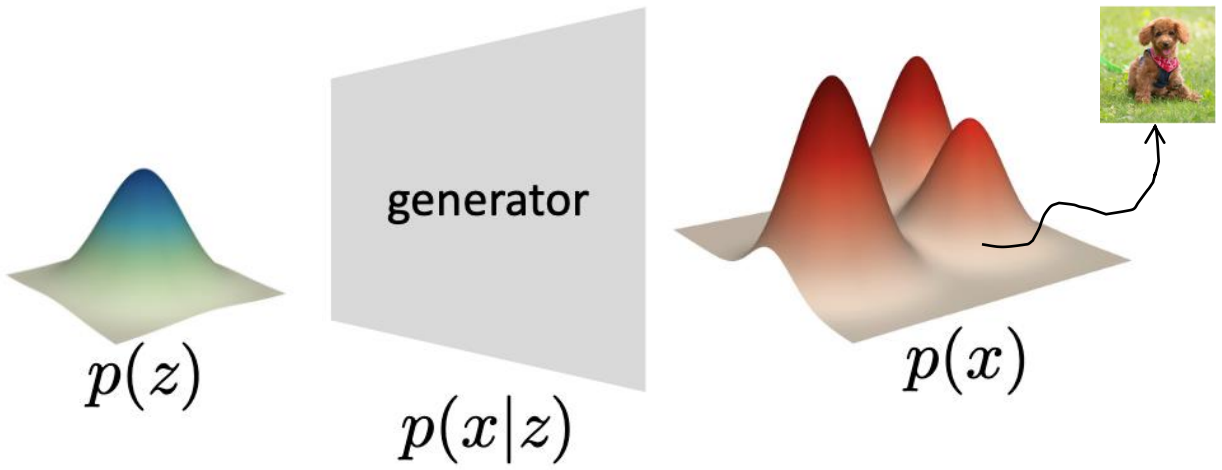


Image source: Kaiming He

In Fact, This Views Generation as Mapping a Prior Distribution $\mathbf{p(z)}$ to the Data Distribution $\mathbf{p(x)}$

How to model the mapping differentiates generative models, such as VAE, DDPM, Flow Matching Model, and GAN...

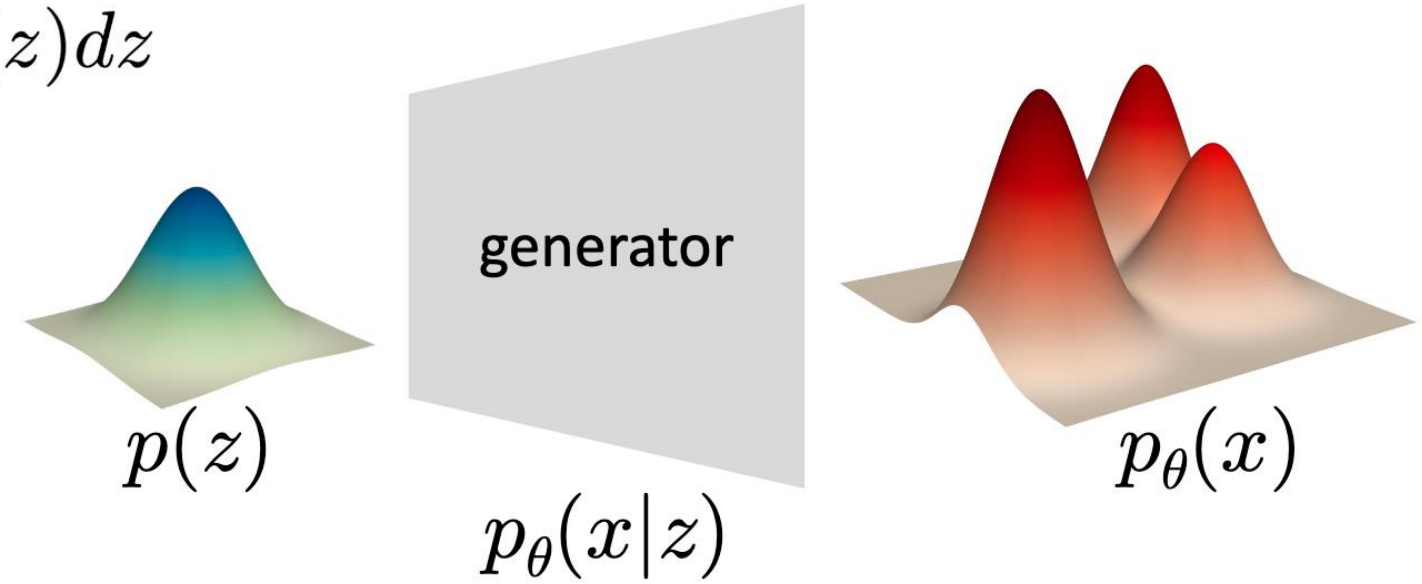
Video source: https://mlg.eng.cam.ac.uk/blog/2024/01/20/flow-matching.html

VAE as Latent Variable Models

We want to maximize $\mathbb{E}_{x \sim p_{data}} \log p_\theta(x)$

with $p_\theta(x)$ **represented as:**

$$p_\theta(x) = \int_z p_\theta(x|z)p(z)dz$$



VAE as Latent Variable Models

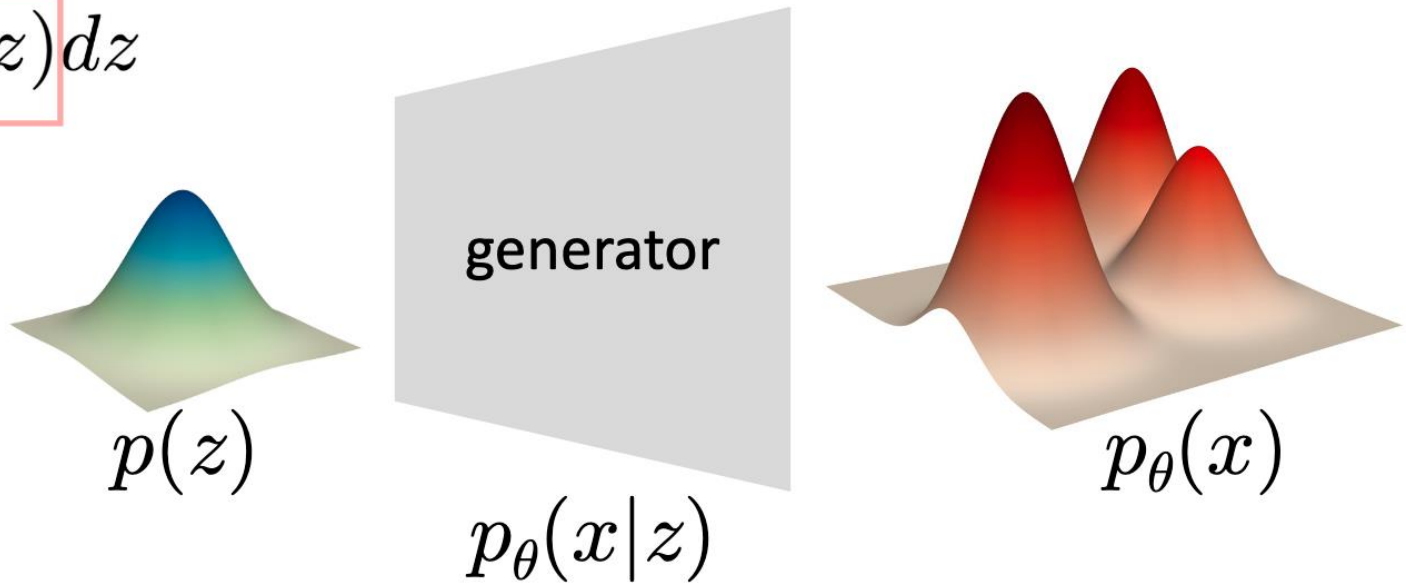
We want to maximize $\mathbb{E}_{x \sim p_{data}} \log p_\theta(x)$

with $p_\theta(x)$ represented as:

$$p_\theta(x) = \int_z p_\theta(x|z) p(z) dz$$

Two sets of unknowns:

- We need to optimize for θ
- We can't control "**true**" $p(z)$



Idea: introduce a "controllable" distribution $q(z)$

Slide credit Kaiming He

Latent Variable Models

$$\log p_\theta(x)$$

Rewrite log likelihood by latent z

- for any distribution $q(z)$
- Bayes' rule

$$= \int_z q(z) \log p_\theta(x) dz$$

$$= \int_z q(z) \log \Big(\frac{p_\theta(x|z) p_\theta(z)}{p_\theta(z|x)} \Big) dz$$

$$= \int_z q(z) \log \Big(\frac{p_\theta(x|z) p_\theta(z)}{p_\theta(z|x)} \frac{q(z)}{q(z)} \Big) dz$$

$$= \int_z q(z) \Big(\log p_\theta(x|z) + \log \frac{p_\theta(z)}{q(z)} + \log \frac{q(z)}{p_\theta(z|x)} \Big) dz$$

$$= \mathbb{E}_{z \sim q(z)} \Big[\log p_\theta(x|z) \Big] - \mathcal{D}_{\mathrm{KL}} \Big(q(z) || p_\theta(z) \Big) + \mathcal{D}_{\mathrm{KL}} \Big(q(z) || p_\theta(z|x) \Big)$$

Slide credit Kaiming He

Latent Variable Models

$$\log p_\theta(x)$$

$$= \int_z q(z) \log p_\theta(x) dz$$

$$= \int_z q(z) \log \left(\frac{p_\theta(x|z) p_\theta(z)}{p_\theta(z|x)} \right) dz$$

$$= \int_z q(z) \log \left(\frac{p_\theta(x|z) p_\theta(z)}{p_\theta(z|x)} \frac{q(z)}{q(z)} \right) dz$$

$$= \int_z q(z) \left(\log p_\theta(x|z) + \log \frac{p_\theta(z)}{q(z)} + \log \frac{q(z)}{p_\theta(z|x)} \right) dz$$

$$= \mathbb{E}_{z \sim q(z)} \Big[\log p_\theta(x|z) \Big] - \mathcal{D}_{\mathrm{KL}} \Big(q(z) || p_\theta(z) \Big) + \mathcal{D}_{\mathrm{KL}} \Big(q(z) || p_\theta(z|x) \Big)$$

Rewrite log likelihood by latent z

- for any distribution $q(z)$
- Bayes' rule
- just algebra
- just algebra

Slide credit Kaiming He

Latent Variable Models

$$\log p_\theta(x)$$

$$= \int_z q(z) \log p_\theta(x) dz$$

$$= \int_z q(z) \log \left(\frac{p_\theta(x|z) p_\theta(z)}{p_\theta(z|x)} \right) dz$$

$$= \int_z q(z) \log \left(\frac{p_\theta(x|z) p_\theta(z)}{p_\theta(z|x)} \frac{q(z)}{q(z)} \right) dz$$

$$= \int_z q(z) \left(\log p_\theta(x|z) + \log \frac{p_\theta(z)}{q(z)} + \log \frac{q(z)}{p_\theta(z|x)} \right) dz$$

$$= \mathbb{E}_{z \sim q(z)} \Big[\log p_\theta(x|z) \Big] - \mathcal{D}_{\mathrm{KL}} \Big(q(z) || p_\theta(z) \Big) + \mathcal{D}_{\mathrm{KL}} \Big(q(z) || p_\theta(z|x) \Big)$$

Rewrite log likelihood by latent z

- for any distribution $q(z)$
- Bayes' rule
- just algebra
- just algebra

Slide credit Kaiming He

Latent Variable Models

$$\log p_\theta(x)$$

$$= \int_z q(z) \log p_\theta(x) dz$$

$$= \int_z q(z) \log \left(\frac{p_\theta(x|z) p_\theta(z)}{p_\theta(z|x)} \right) dz$$

$$= \int_z q(z) \log \left(\frac{p_\theta(x|z) p_\theta(z)}{p_\theta(z|x)} \frac{q(z)}{q(z)} \right) dz$$

$$= \int_z q(z) \left(\log p_\theta(x|z) + \log \frac{p_\theta(z)}{q(z)} + \log \frac{q(z)}{p_\theta(z|x)} \right) dz$$

$$= \mathbb{E}_{z \sim q(z)} \Big[\log p_\theta(x|z) \Big] - \mathcal{D}_{\mathrm{KL}} \Big(q(z) || p_\theta(z) \Big) + \mathcal{D}_{\mathrm{KL}} \Big(q(z) || p_\theta(z|x) \Big)$$

Rewrite log likelihood by latent z

- for any distribution $q(z)$
- Bayes' rule
- just algebra
- just algebra

Slide credit Kaiming He

Latent Variable Models

$$\log p_\theta(x)$$

Rewrite log likelihood by latent z

$$= \int_z q(z) \log p_\theta(x) dz$$

- for any distribution $q(z)$

$$= \int_z q(z) \log \left(\frac{p_\theta(x|z) p_\theta(z)}{p_\theta(z|x)} \right) dz$$

- Bayes' rule

$$= \int_z q(z) \log \left(\frac{p_\theta(x|z) p_\theta(z)}{p_\theta(z|x)} \frac{q(z)}{q(z)} \right) dz$$

- just algebra

$$= \int_z q(z) \left(\log p_\theta(x|z) + \log \frac{p_\theta(z)}{q(z)} + \log \frac{q(z)}{p_\theta(z|x)} \right) dz$$

- just algebra

$$= \mathbb{E}_{z \sim q(z)} \Big[\log p_\theta(x|z) \Big] - \mathcal{D}_{\mathrm{KL}} \Big(q(z) || p_\theta(z) \Big) + \mathcal{D}_{\mathrm{KL}} \Big(q(z) || p_\theta(z|x) \Big)$$

Slide credit Kaiming He

Latent Variable Models

$$
\begin{aligned}
& \log p_\theta(x) \\
= & \int_z q(z) \log p_\theta(x) dz \\
= & \int_z q(z) \log \left(\frac{p_\theta(x|z) p_\theta(z)}{p_\theta(z|x)} \right) dz \\
= & \int_z q(z) \log \left(\frac{p_\theta(x|z) p_\theta(z)}{p_\theta(z|x)} \frac{q(z)}{q(z)} \right) dz \\
= & \int_z q(z) \left(\log p_\theta(x|z) + \log \frac{p_\theta(z)}{q(z)} + \log \frac{q(z)}{p_\theta(z|x)} \right) dz \\
= & \mathbb{E}_{z \sim q(z)} \Big[\log p_\theta(x|z) \Big] - \mathcal{D}_{\mathrm{KL}} \Big(q(z) || p_\theta(z) \Big) + \mathcal{D}_{\mathrm{KL}} \Big(q(z) || p_\theta(z|x) \Big)
\end{aligned}
$$

Rewrite log likelihood by latent z

- for any distribution $q(z)$
- Bayes' rule
- just algebra
- just algebra

Slide credit Kaiming He

Latent Variable Models

intractable $\log p_\theta(x)$

Rewrite log likelihood by latent z

- for any distribution $q(z)$
- Bayes' rule

$$= \int_z q(z) \log p_\theta(x) dz$$

$$= \int_z q(z) \log \left(\frac{p_\theta(x|z) p_\theta(z)}{p_\theta(z|x)} \right) dz$$

$$= \int_z q(z) \log \left(\frac{p_\theta(x|z) p_\theta(z)}{p_\theta(z|x)} \frac{q(z)}{q(z)} \right) dz$$

$$= \int_z q(z) \left(\log p_\theta(x|z) + \log \frac{p_\theta(z)}{q(z)} + \log \frac{q(z)}{p_\theta(z|x)} \right) dz$$

$$= \mathbb{E}_{z \sim q(z)} \left[\log p_\theta(x|z) \right] - \mathcal{D}_{\mathrm{KL}} \left(q(z) || p_\theta(z) \right) + \mathcal{D}_{\mathrm{KL}} \left(q(z) || p_\theta(z|x) \right)$$

tractable tractable intractable

Slide credit Kaiming He

Latent Variable Models

intractable $\log p_\theta(x)$ $- \mathcal{D}_{\mathrm{KL}}\left(q(z)||p_\theta(z|x)\right)$ intractable

$= \mathbb{E}_{z\sim q(z)}\left[\log p_\theta(x|z)\right]$ $- \mathcal{D}_{\mathrm{KL}}\left(q(z)||p_\theta(z)\right)$

tractable tractable

Slide credit Kaiming He

Latent Variable Models

intractable $\log p_\theta(x) - \mathcal{D}_{\mathrm{KL}}\left(q(z)||p_\theta(z|x)\right)$ intractable

$= \mathbb{E}_{z\sim q(z)}\left[\log p_\theta(x|z)\right] - \mathcal{D}_{\mathrm{KL}}\left(q(z)||p_\theta(z)\right)$

tractable tractable

- This is called Evidence Lower Bound (ELBO)
- Lower bound of $\log p_\theta(x)$

- This equation holds for <u>any</u> distribution $q(z)$

Slide credit Kaiming He

Latent Variable Models

$$\underbrace{\mathbb{E}_{z\sim q(z)}\Big[\log p_\theta(x|z)\Big]}_{q_\phi(z|x)} \underbrace{- \mathcal{D}_{\mathrm{KL}}\Big(q(z)||p_\theta(z)\Big)}_{q_\phi(z|x)}$$

tractable tractable

- This is called Evidence Lower Bound (ELBO)
- Lower bound of $\log p_\theta(x)$
- This equation holds for <u>any</u> distribution $q(z)$
- Parameterize $q(z)$ by $q_\phi(z|x)$

Slide credit Kaiming He

Latent Variable Models

$$\underbrace{\mathbb{E}_{z\sim q(z)}\left[\log p_\theta(x|z)\right]}_{\text{tractable}} \underbrace{- \mathcal{D}_{\mathrm{KL}}\left(q(z)||p_\theta(z)\right)}_{\text{tractable}}$$

$q_\phi(z|x)$ $\quad$ $q_\phi(z|x)$ $\quad$ $p(z)$

- This is called Evidence Lower Bound (ELBO)
- Lower bound of $\log p_\theta(x)$

- This equation holds for <u>any</u> distribution $q(z)$

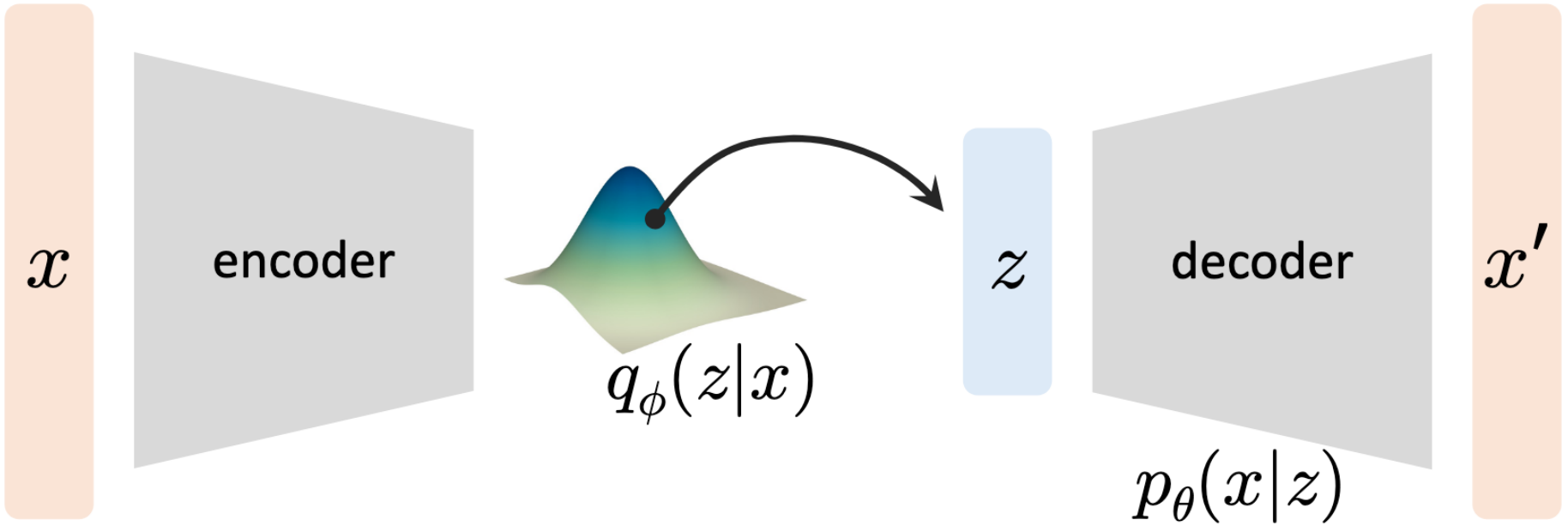
- Parameterize $q(z)$ by $q_\phi(z|x)$
- let $p_\theta(z)$ be a simple known prior $p(z)$

Slide credit Kaiming He

Variational Autoencoder

Maximize ELBO ⇒ minimize an objective:

$$\mathcal{L}_{\theta,\phi}(x) = -\mathbb{E}_{z\sim q_\phi(z|x)}\Big[\log p_\theta(x|z)\Big] + \mathcal{D}_{\mathrm{KL}}\Big(q_\phi(z|x)||p(z)\Big)$$

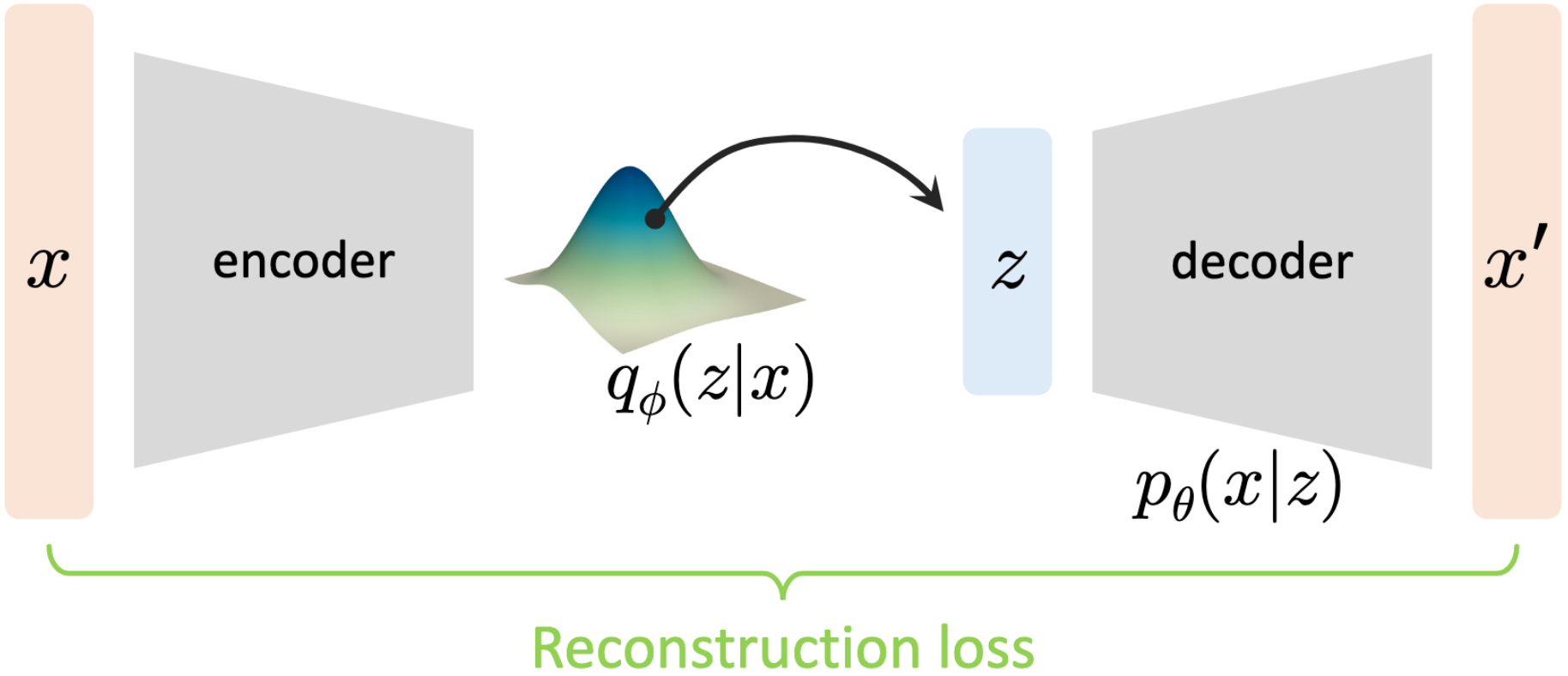


Slide credit Kaiming He

Variational Autoencoder

Maximize ELBO ⇒ minimize an objective:

$$\mathcal{L}_{\theta,\phi}(x) = -\mathbb{E}_{z\sim q_\phi(z|x)}\Big[\log p_\theta(x|z)\Big] + \mathcal{D}_{\mathrm{KL}}\Big(q_\phi(z|x)||p(z)\Big)$$

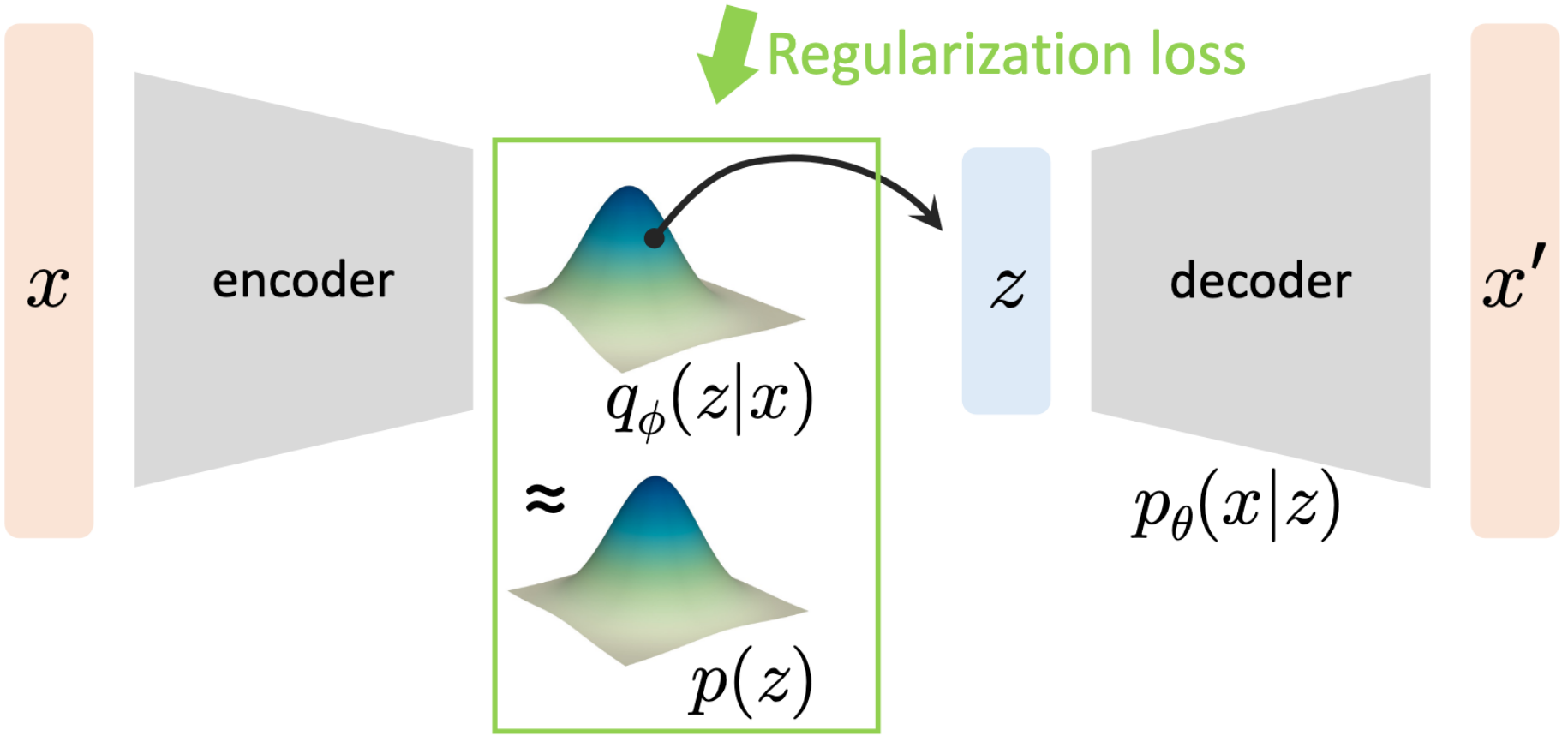


Slide credit Kaiming He

Variational Autoencoder

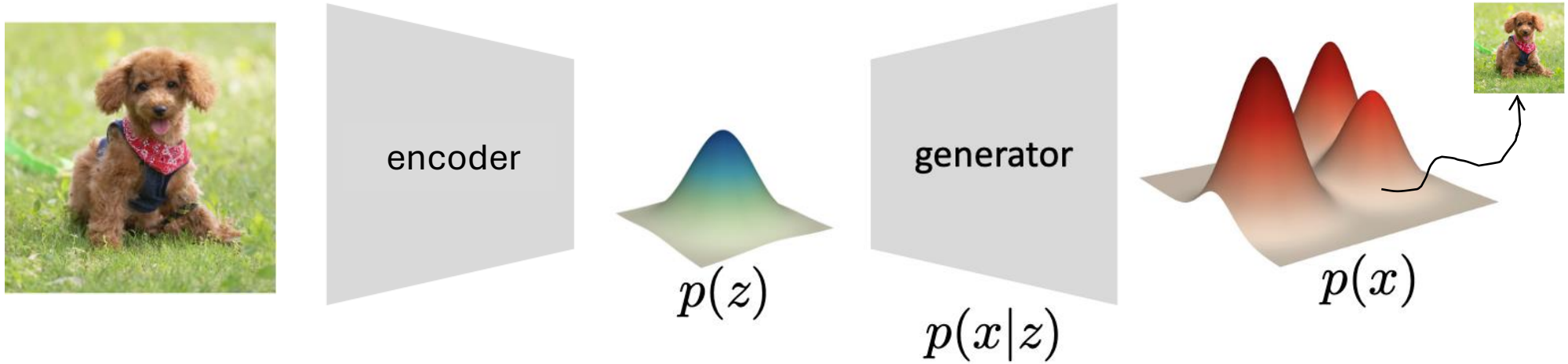
Maximize ELBO ⇒ minimize an objective:

$$\mathcal{L}_{\theta,\phi}(x) = -\mathbb{E}_{z\sim q_\phi(z|x)}\Big[\log p_\theta(x|z)\Big] + \mathcal{D}_{\mathrm{KL}}\Big(q_\phi(z|x)||p(z)\Big)$$



Slide credit Kaiming He

Training:



Testing:

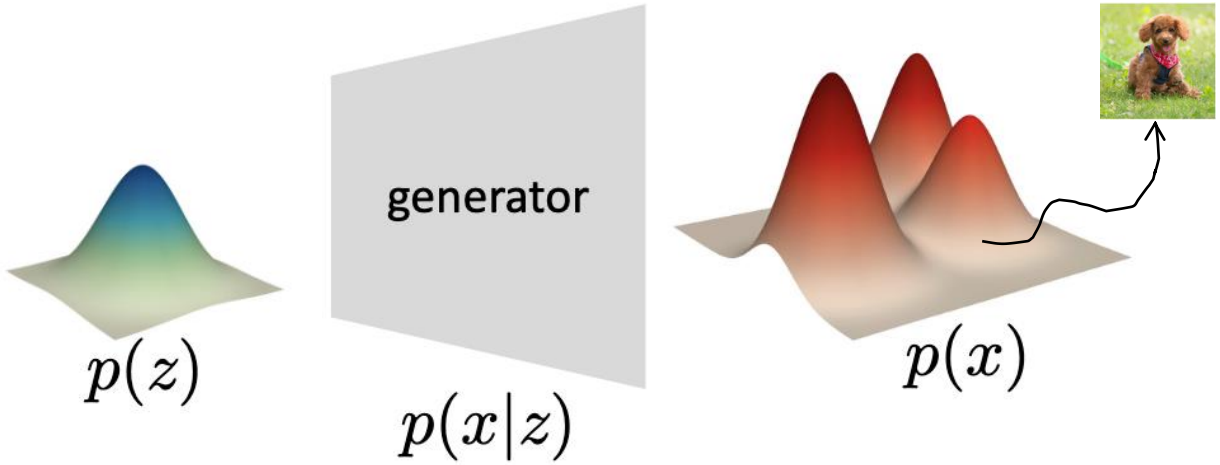
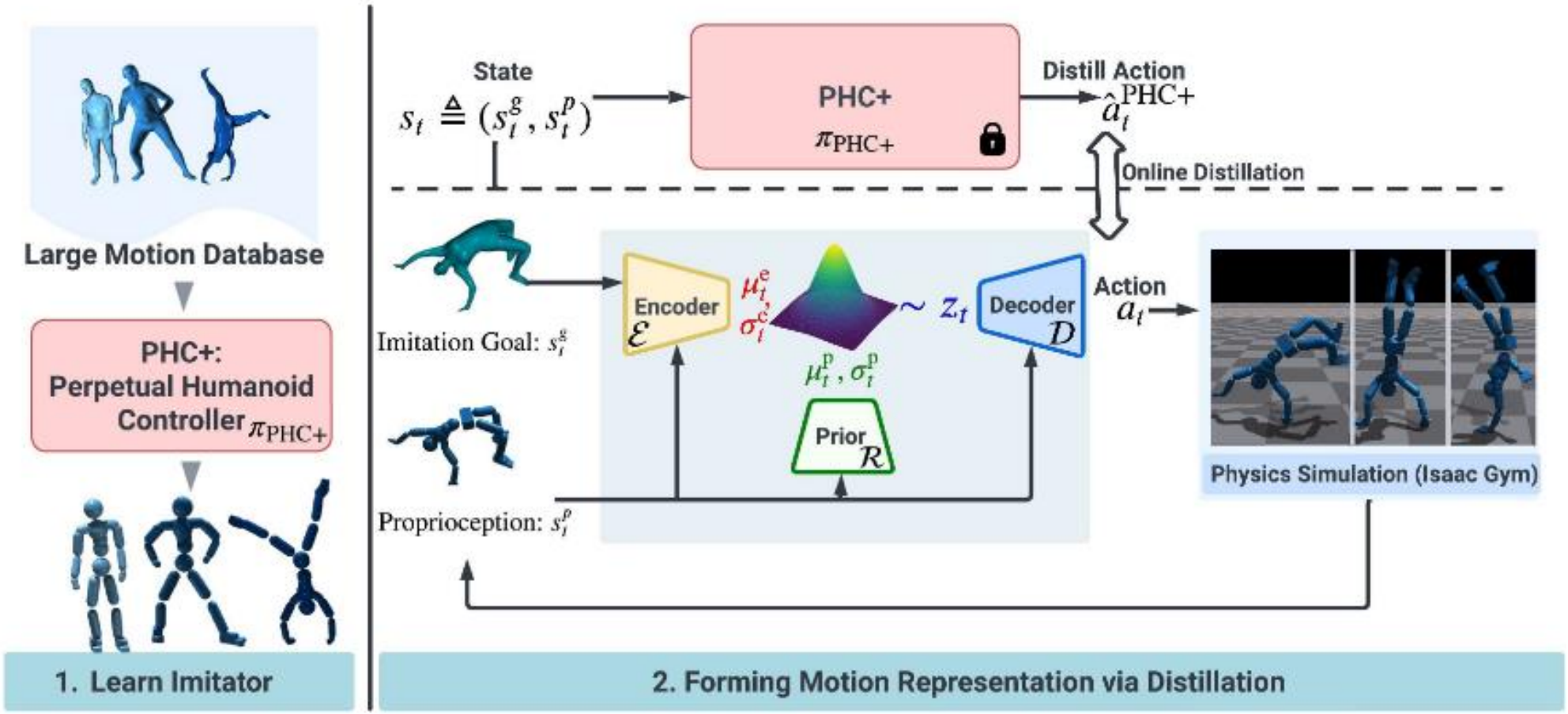


Image source: Kaiming He

Why Do We Care About VAE? A Common Technique to Compress Data as Priors

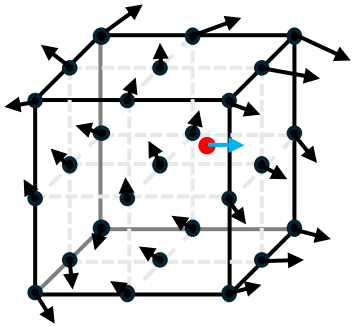


Universal Humanoid Motion Representations for Physics-Based Control. Luo et al.

Why Do We Care About VAE? A Common Technique to Compress Data as Priors

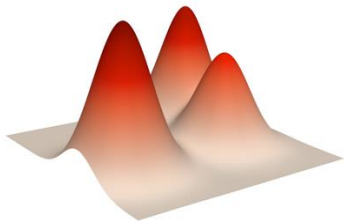
Static Object

Animated Object

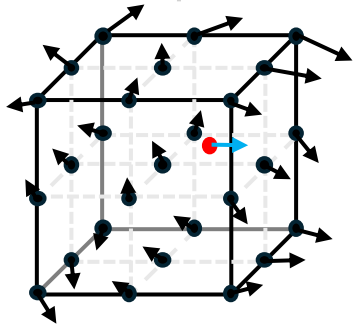


Motion Field

Encoder



Decoder



Motion Field

Video generated by Gaussian Variation Field Diffusion for High-fidelity Video-to-4D Synthesis.

Content

- Generative Modeling
  - ➢Autoregressive Models
  - ➢Variational AutoEncoders
  - ➢Denoising Diffusion Probabilistic Models
  - ➢Flow Matching Models
- 4D Generative Modeling
  - ➢3D Generation with Multi-view Video Generation
  - ➢4D Generation with Multi-view Video Generation
  - ➢3D Geometry Generation

Another Way of Modeling Conditional Distributions

- Condition on latent variables:

$$p(x_1, x_2, \cdots, x_n) = p(\boldsymbol{z})p(x_1, x_2, \cdots, x_n|\boldsymbol{z})$$

$$\text{with } p(\boldsymbol{z}) = p(z_1)p(z_2|z_1)\cdots p(z_n|z_1, z_2, \cdots, z_{n-1})$$

Another Way of Modeling Conditional Distributions

- Condition on latent variables:

$$\mathrm{p}(\mathrm{x}_1, \mathrm{x}_2, \cdots, \mathrm{x}_\mathrm{n}) = \mathrm{p}(\boldsymbol{z})\mathrm{p}(\mathrm{x}_1, \mathrm{x}_2, \cdots, \mathrm{x}_\mathrm{n}|\boldsymbol{z})$$

$$\text{with } \mathrm{p}(\boldsymbol{z}) = \mathrm{p}(\mathrm{z}_1)\mathrm{p}(\mathrm{z}_2|\mathrm{z}_1)\cdots\mathrm{p}(\mathrm{z}_\mathrm{n}|\mathrm{z}_1, \mathrm{z}_2, \cdots, \mathrm{z}_{\mathrm{n}-1})$$

$$p(\mathbf{x}_{0:T}) = p(\mathbf{x}_T)p(\mathbf{x}_{T-1} \mid \mathbf{x}_T)\ldots p(\mathbf{x}_1 \mid \mathbf{x}_2)p(\mathbf{x}_0 \mid \mathbf{x}_1)$$

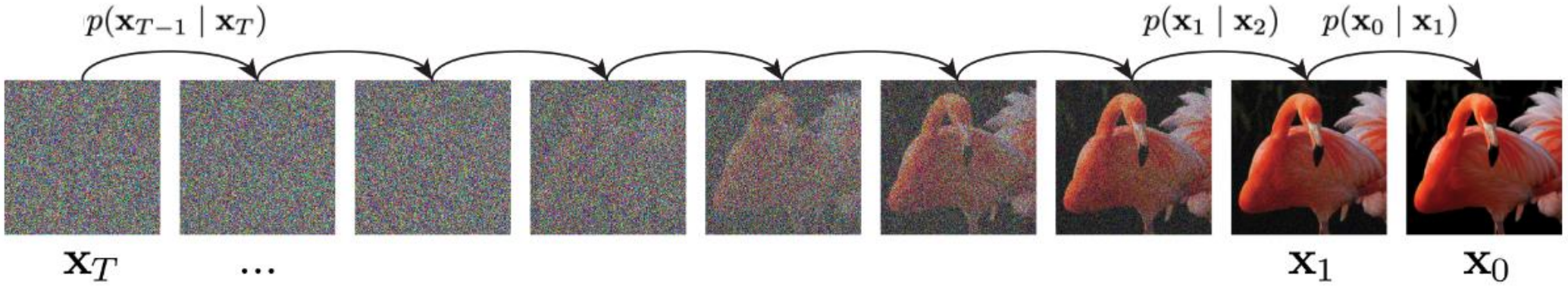


Image source: Kaiming He

Modeling Transitions of Two Distributions

$$p(\mathbf{x}_{0:T}) = p(\mathbf{x}_T)p(\mathbf{x}_{T-1} \mid \mathbf{x}_T)...p(\mathbf{x}_1 \mid \mathbf{x}_2)p(\mathbf{x}_0 \mid \mathbf{x}_1)$$

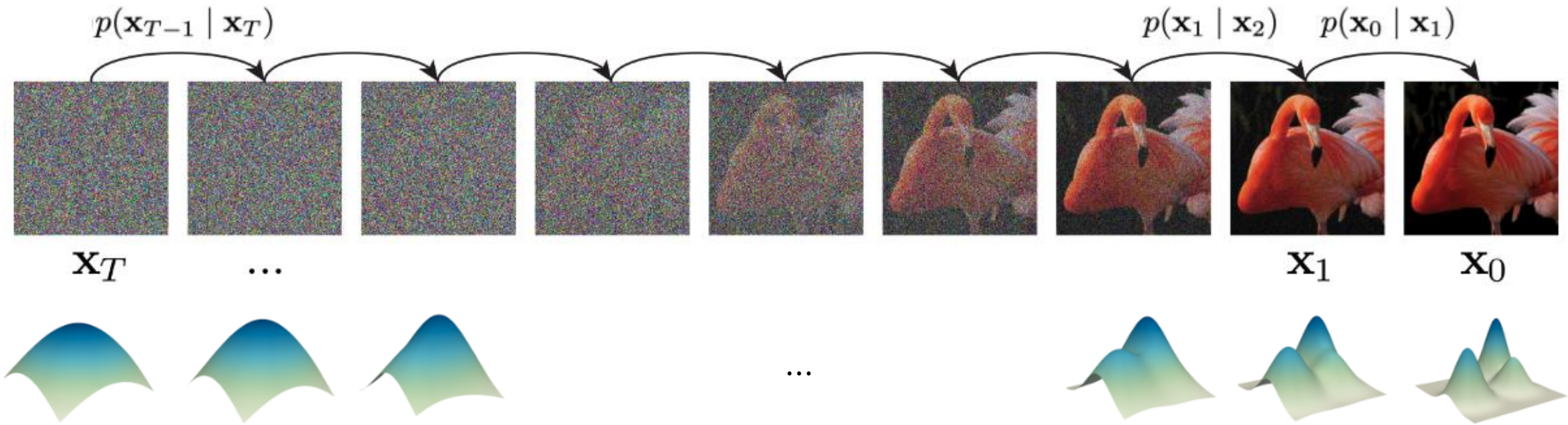


Image source: Kaiming He

Model How Data is Transported Among the Prior Distribution $\mathbf{p(z)}$ and the Target Distribution $\mathbf{p(x)}$

Video source: https://mlg.eng.cam.ac.uk/blog/2024/01/20/flow-matching.html

Generation: Map a Prior Distribution $\mathbf{p(z)}$ to the Data Distribution $\mathbf{p(x)}$

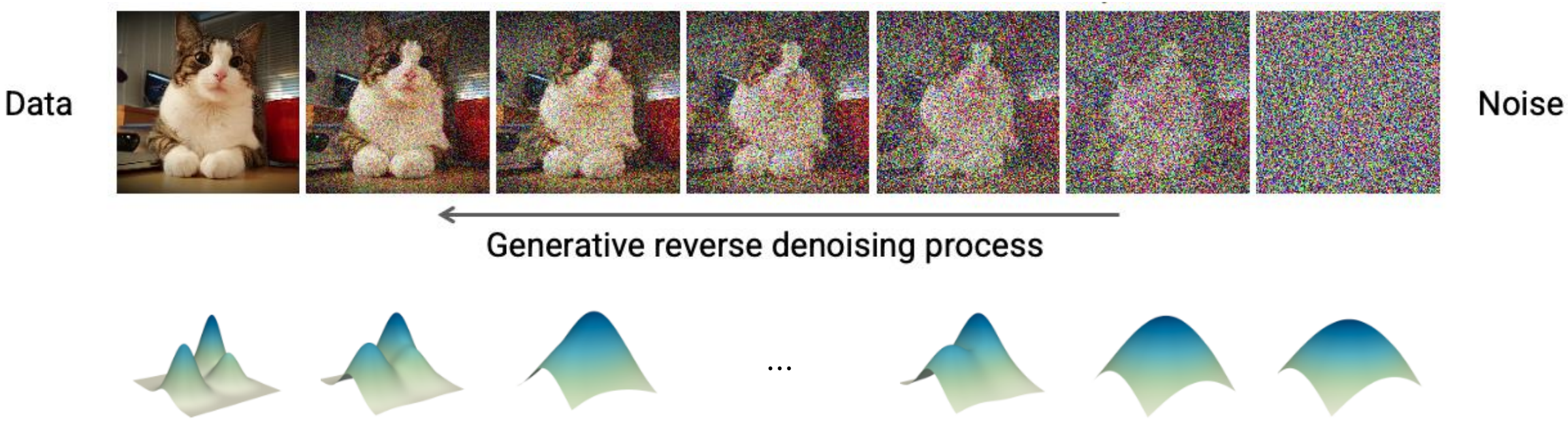


Image source: https://cvpr2023-tutorial-diffusion-models.github.io/

Basic Idea: First, Determine How the Data Distribution $\mathbf{p(x)}$ is Mapped to a Prior Distribution $\mathbf{p(z)}$

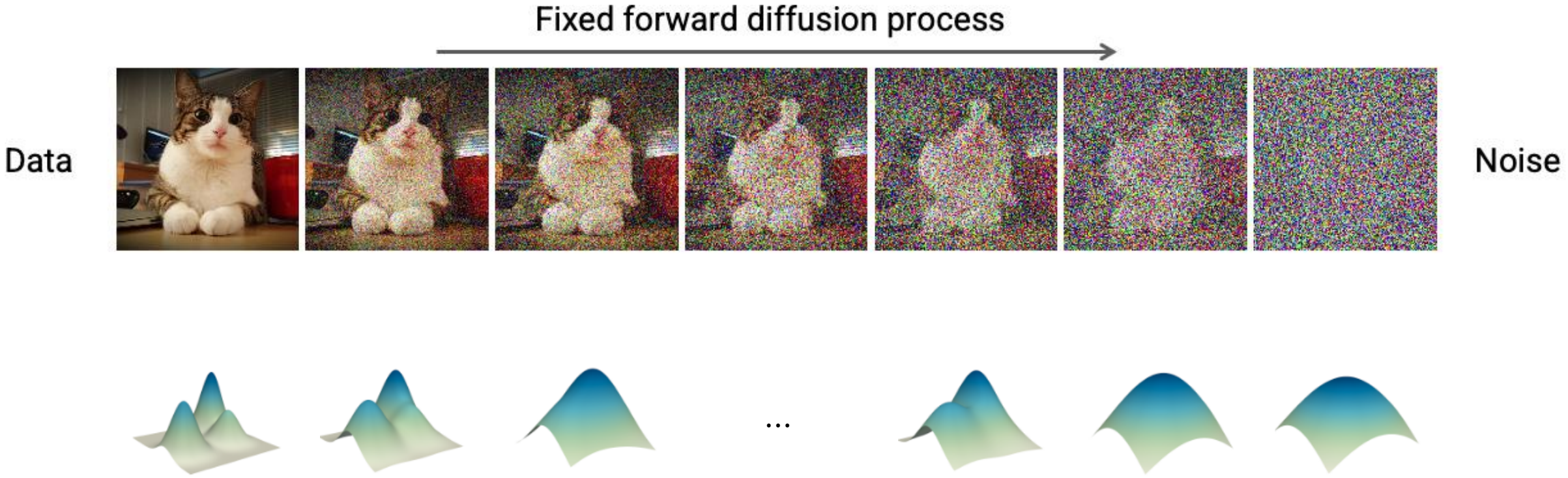


Image source: https://cvpr2023-tutorial-diffusion-models.github.io/

Basic Idea: Then, Learn to Reverse the Mapping

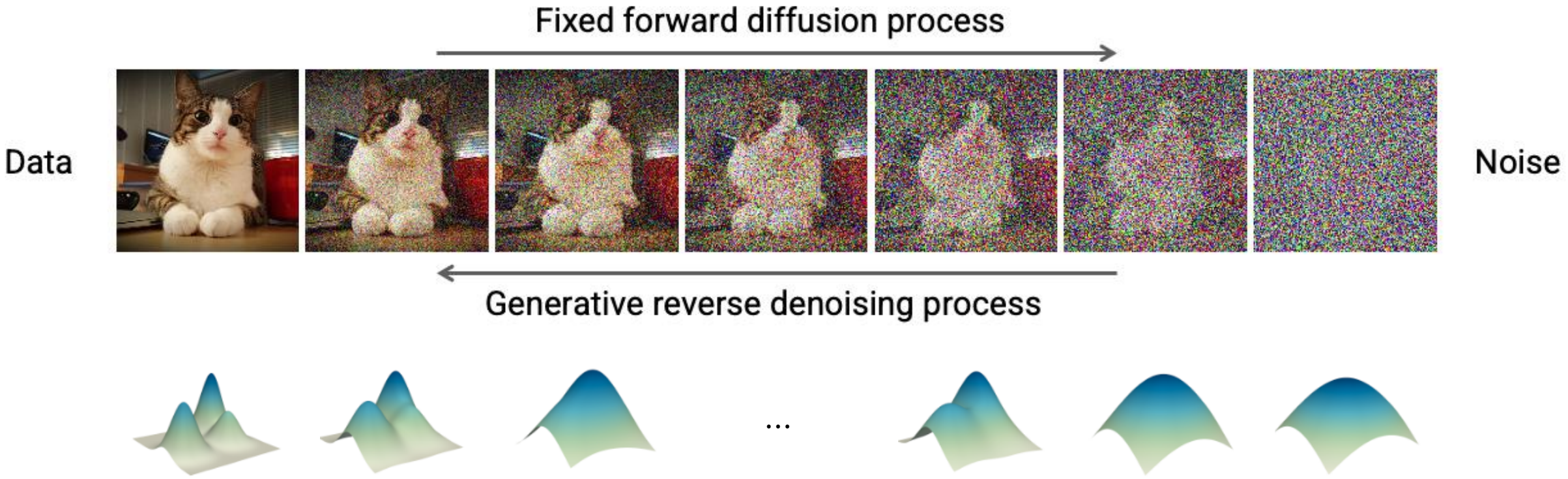
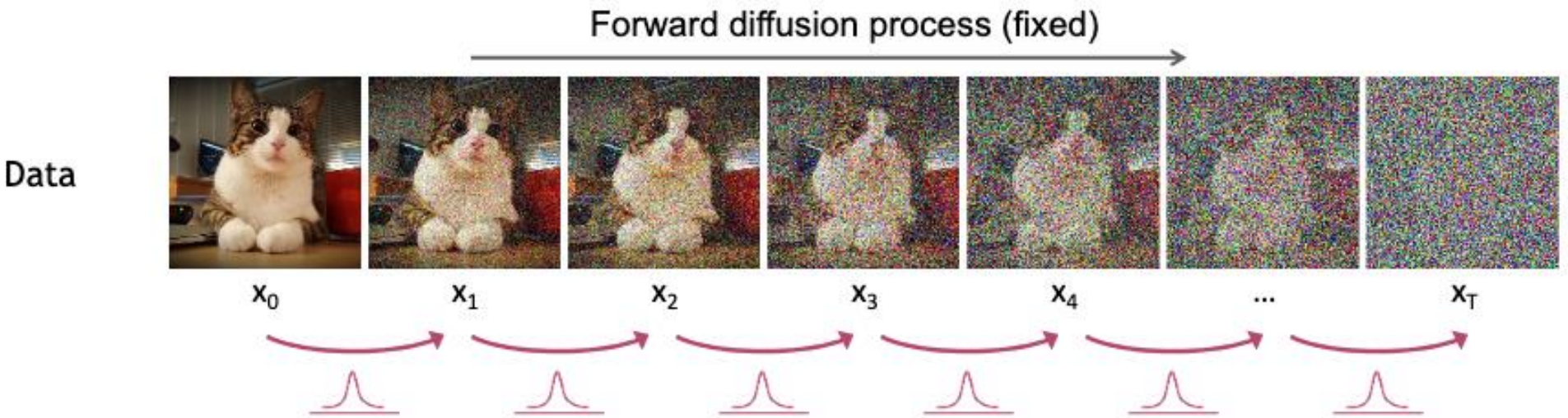


Image source: https://cvpr2023-tutorial-diffusion-models.github.io/

Denoising Diffusion Probabilistic Models Specify the Forward Mapping with Gaussian Diffusion Process

The formal definition of the forward process in T steps:



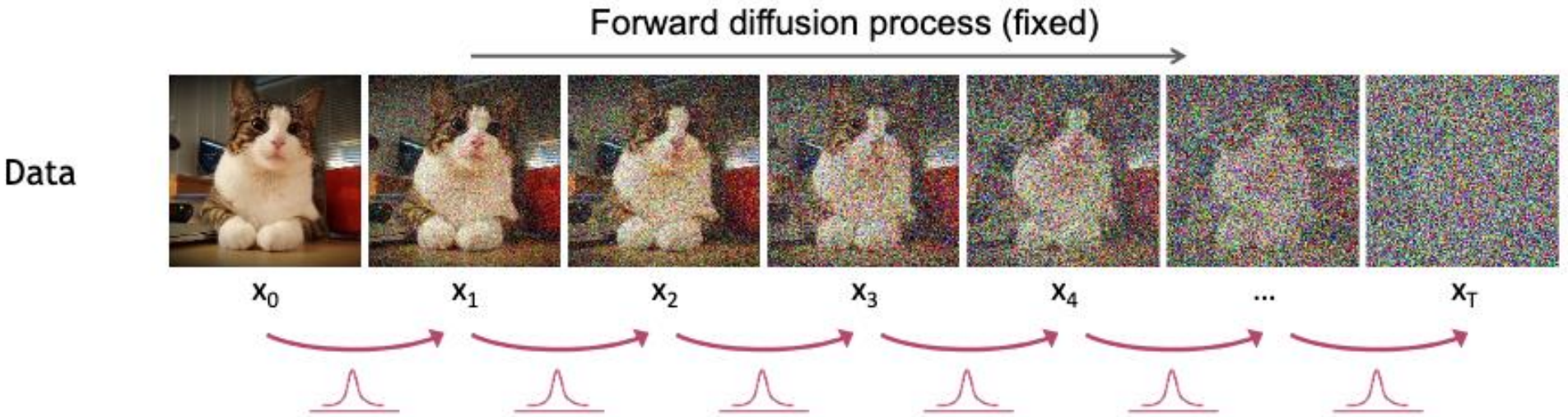
$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \quad \Rightarrow \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \qquad \text{(joint)}$$

mean variance

Slide credit: https://cvpr2023-tutorial-diffusion-models.github.io/

Denoising Diffusion Probabilistic Models Specify the Forward Mapping with Gaussian Diffusion Process

The formal definition of the forward process in T steps:

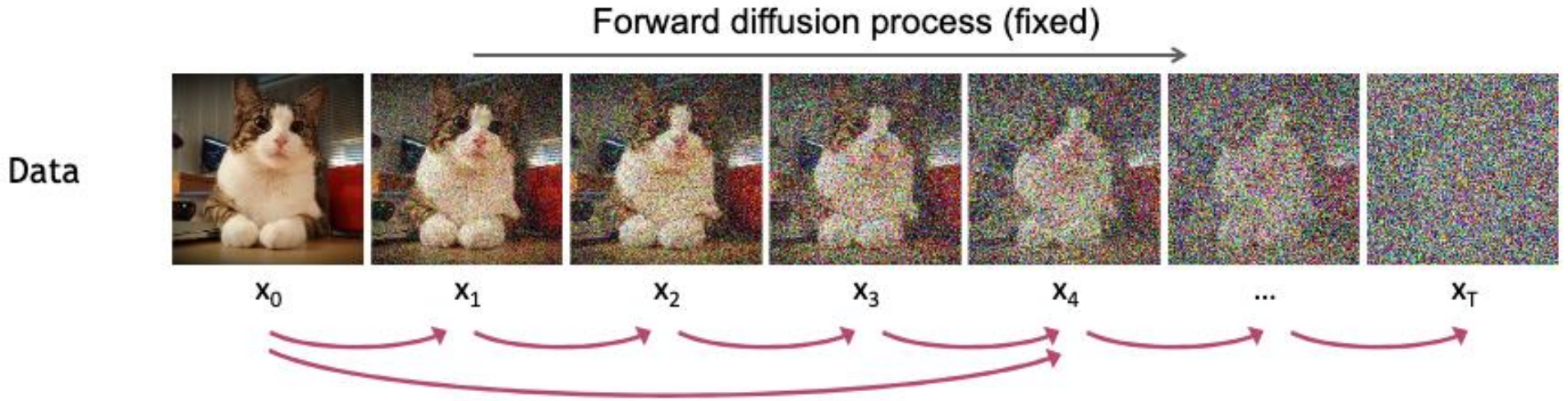


$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \quad \Rightarrow \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \qquad \text{(joint)}$$

schedule mean variance

Slide credit: https://cvpr2023-tutorial-diffusion-models.github.io/

Denoising Diffusion Probabilistic Models Specify the Forward Mapping with Gaussian Diffusion Process



Define $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$ ➡ $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}))$ (Diffusion Kernel)

For sampling: $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\,\epsilon$ where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

β_t values schedule (i.e., the noise schedule) is designed such that $\bar{\alpha}_T \to 0$ and $q(\mathbf{x}_T|\mathbf{x}_0) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I}))$

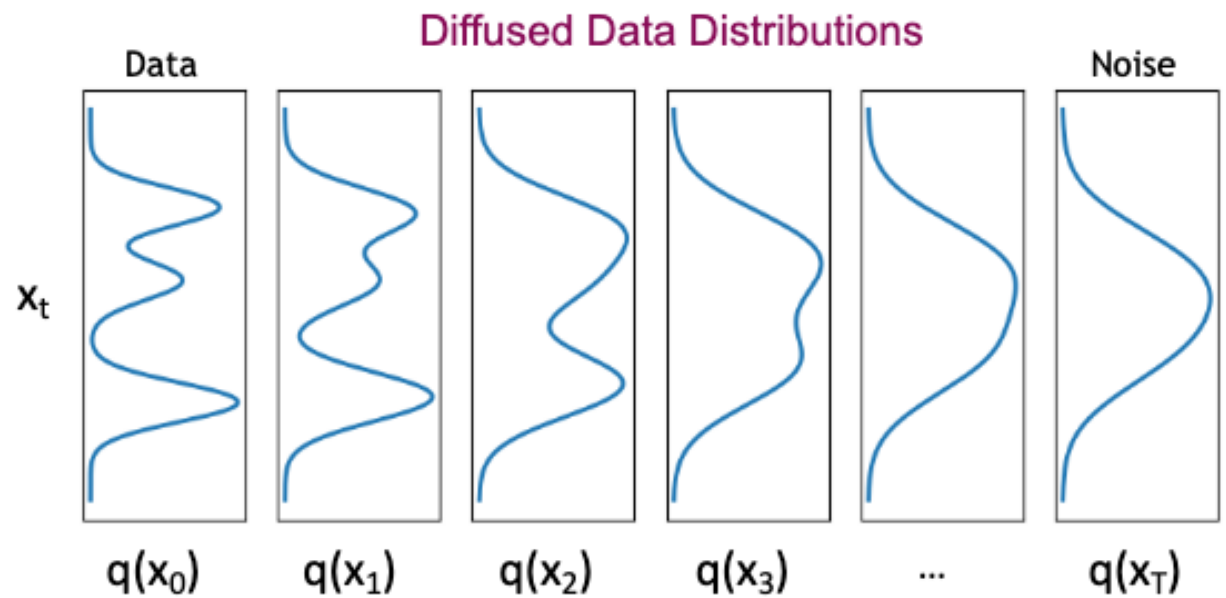
Slide credit: https://cvpr2023-tutorial-diffusion-models.github.io/

What happens to a distribution in the forward diffusion?

So far, we discussed the diffusion kernel $q(\mathbf{x}_t|\mathbf{x}_0)$ but what about $q(\mathbf{x}_t)$?

$$\underbrace{q(\mathbf{x}_t)}_{\text{Diffused data dist.}} = \int \underbrace{q(\mathbf{x}_0, \mathbf{x}_t)}_{\text{Joint dist.}} d\mathbf{x}_0 = \int \underbrace{q(\mathbf{x}_0)}_{\text{Input data dist.}} \underbrace{q(\mathbf{x}_t|\mathbf{x}_0)}_{\text{Diffusion kernel}} d\mathbf{x}_0$$

The diffusion kernel is Gaussian convolution.



We can sample $\mathbf{x}_t \sim q(\mathbf{x}_t)$ by first sampling $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ and then sampling $\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x}_0)$ (i.e., ancestral sampling).

Slide credit: https://cvpr2023-tutorial-diffusion-models.github.io/

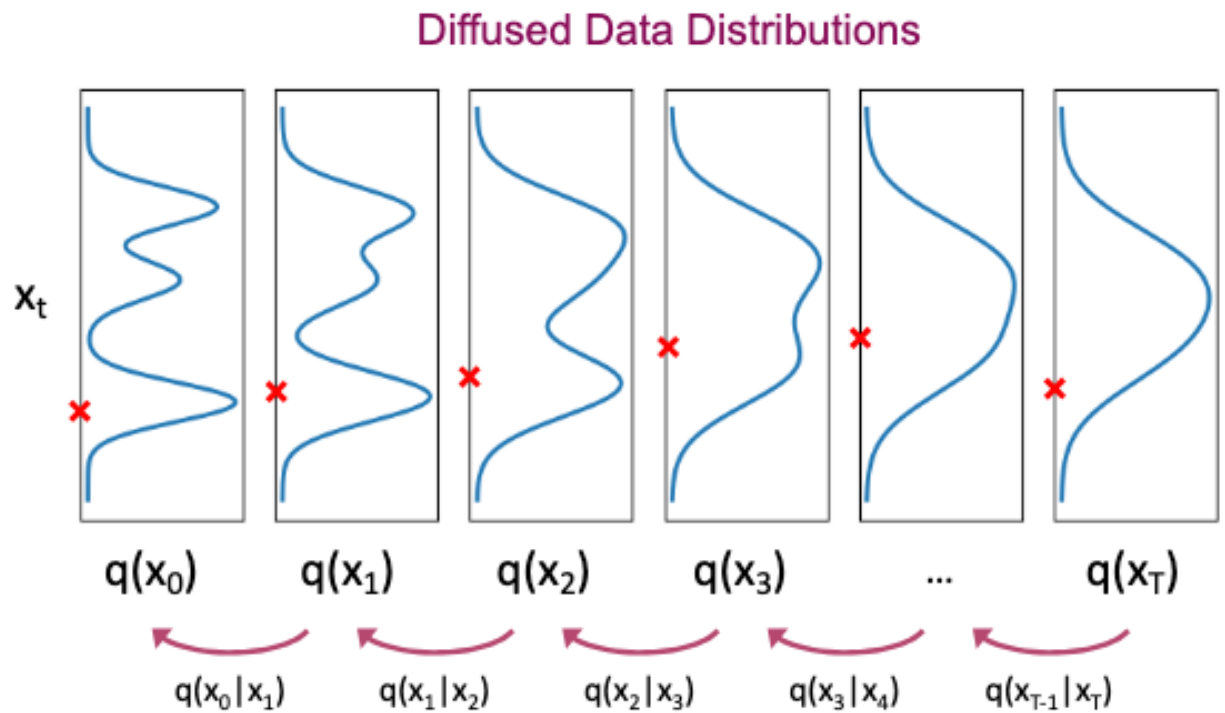
Generative Learning by Denoising

Recall, that the diffusion parameters are designed such that $q(\mathbf{x}_T) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$

Generation:

Sample $\mathbf{x}_T \sim \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$

Iteratively sample $\mathbf{x}_{t-1} \sim \underbrace{q(\mathbf{x}_{t-1}|\mathbf{x}_t)}_{\text{True Denoising Dist.}}$

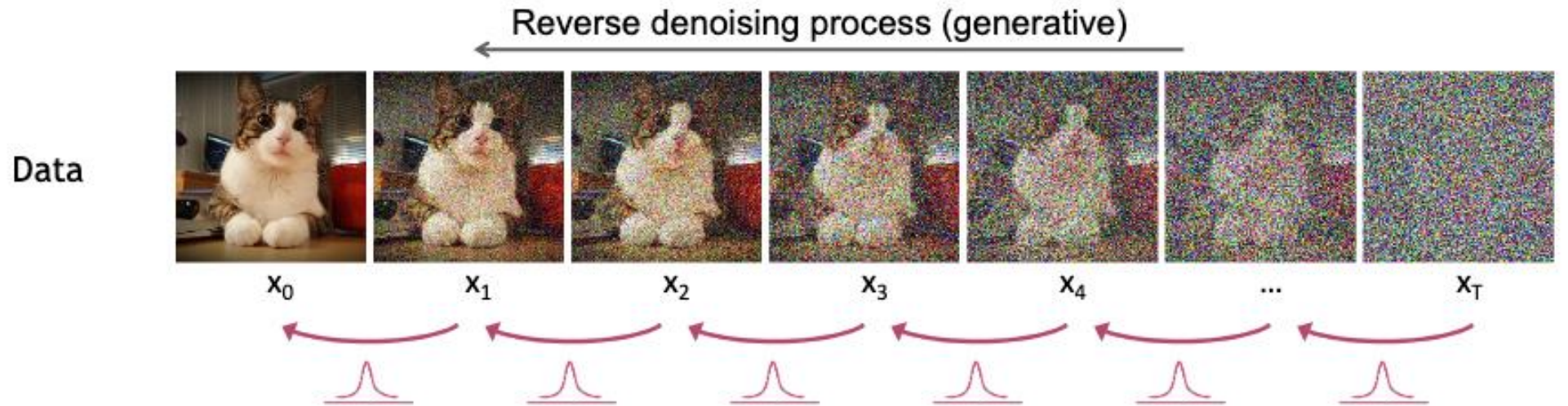


Can we approximate $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$? Yes, we can use a Normal distribution if β_t is small in each forward diffusion step.

Slide credit: https://cvpr2023-tutorial-diffusion-models.github.io/

We Directly Have Reverse Mappings of Gaussian Diffusion Process

Formal definition of forward and reverse processes in T steps:



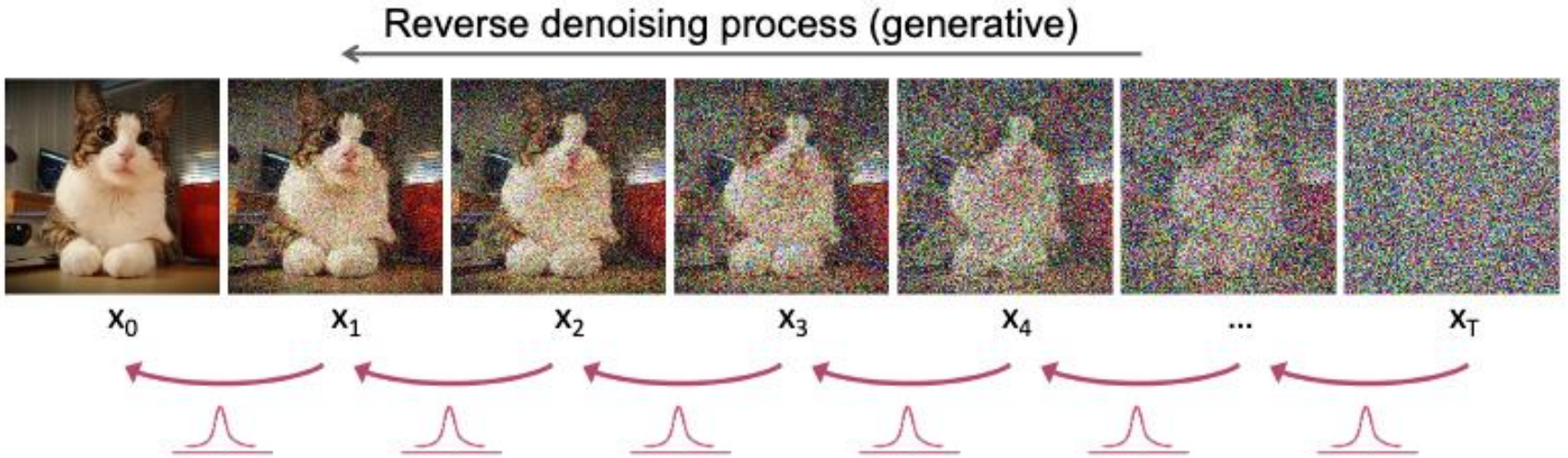
$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I}),$$

$$\text{where} \quad \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t \quad \text{and} \quad \tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

Slide credit: https://cvpr2023-tutorial-diffusion-models.github.io/

Learn a Model to Reverse Mappings of Gaussian Diffusion Process

Formal definition of forward and reverse processes in T steps:



$$p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$$
$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2\mathbf{I})$$

Trainable network
(U-net, Denoising Autoencoder)

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$$

Slide credit: https://cvpr2023-tutorial-diffusion-models.github.io/

Learning Denoising Model

Variational upper bound

For training, we can form variational upper bound that is commonly used for training variational autoencoders:

$$\mathbb{E}_{q(\mathbf{x}_0)}\left[-\log p_\theta(\mathbf{x}_0)\right] \leq \mathbb{E}_{q(\mathbf{x}_0)q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] =: L$$

Recall that $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\,\epsilon$. Ho et al. NeurIPS 2020 parameterized the mean of denoising model via:

$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{1-\beta_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\,\epsilon_\theta(\mathbf{x}_t, t)\right)$$

Using a few simple arithmetic operations, we can write down the variational objective as:

$$L = \mathbb{E}_{\mathbf{x}_0\sim q(\mathbf{x}_0), t\sim\mathcal{U}\{1,T\}, \epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\lambda_t ||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon, t)||^2\right]$$

Ho et al. NeurIPS 2020 observe that simply setting λ_t to 1 for all *t* works best in practice.

Summary

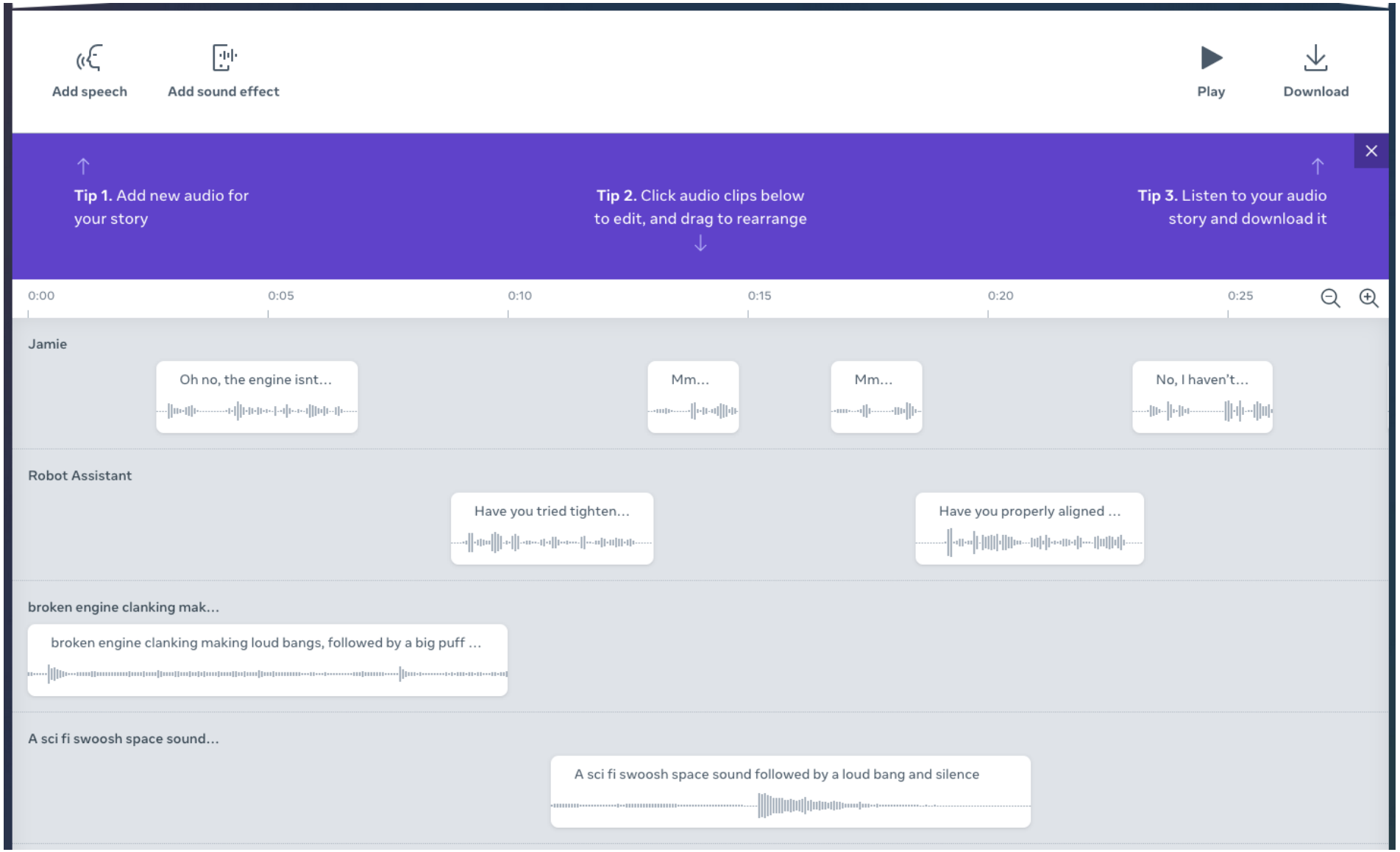
Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

Diffusion Models are Awesome



Meta: AudioBox

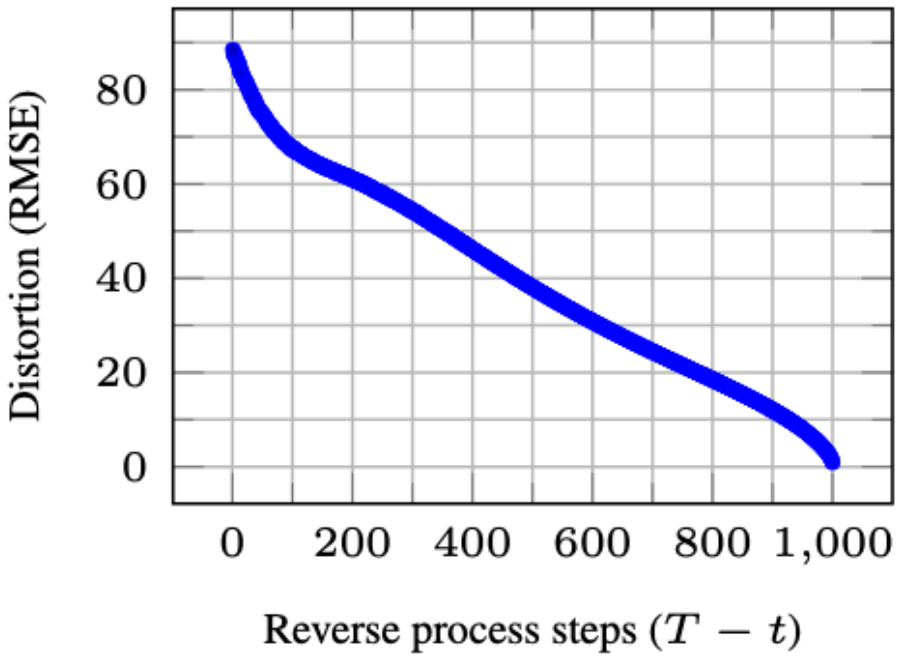
Diffusion Models are Awesome

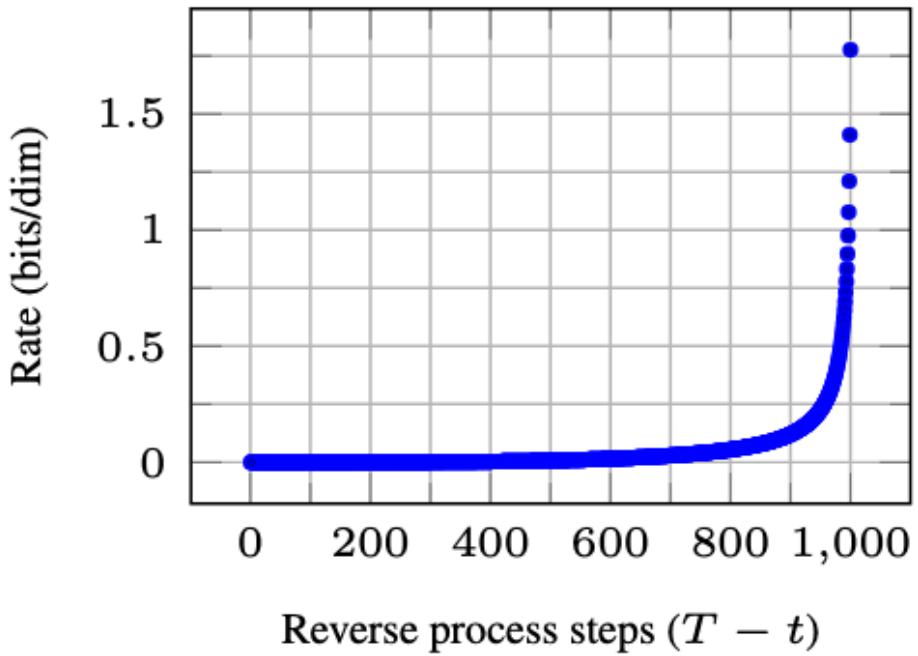
Jason Allen: Theatre Dopera Spatial

Diffusion Models are Awesome

OpenAI: Sora

Diffusion Models Are Slow…



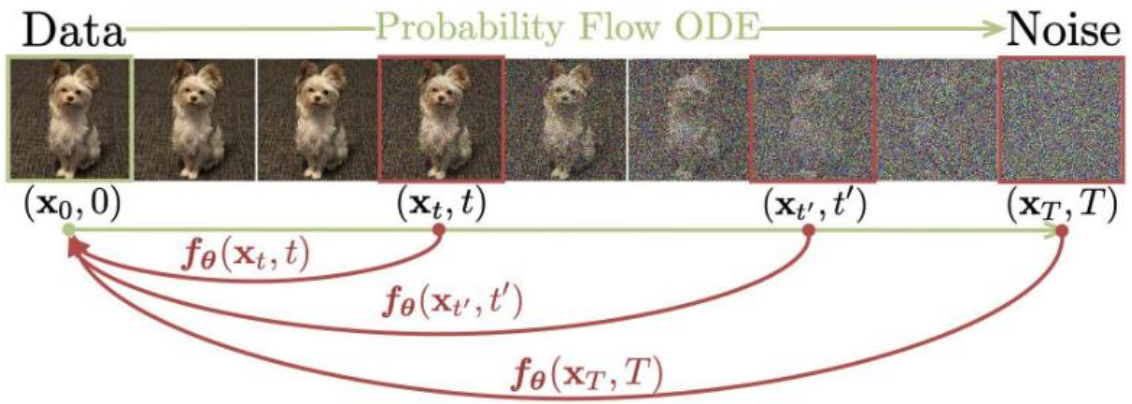


DDPM often requires 1000 samplings steps to generate high-quality data….

Many Techniques Introduced to Accelerate DDPM

Consistency Models

Yang Song [1] **Prafulla Dhariwal** [1] **Mark Chen** [1] **Ilya Sutskever** [1]



DENOISING DIFFUSION IMPLICIT MODELS

Jiaming Song, Chenlin Meng & Stefano Ermon
Stanford University
{tsong,chenlin,ermon}@cs.stanford.edu

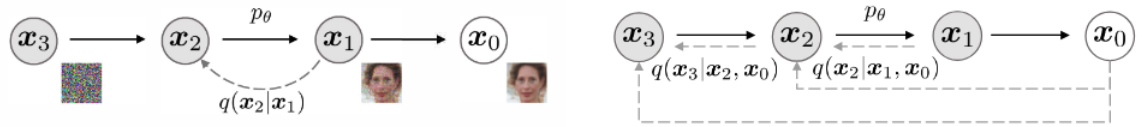
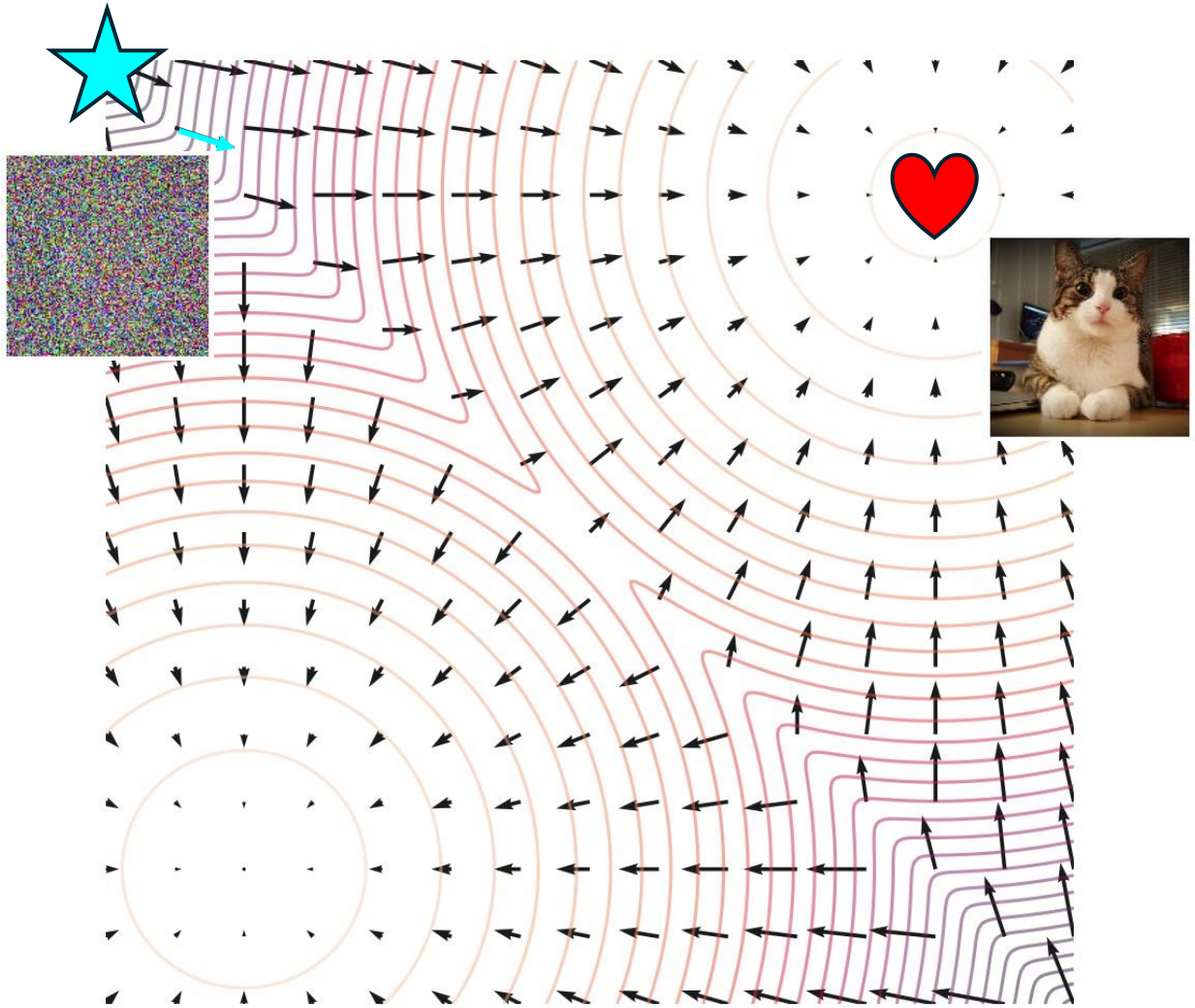


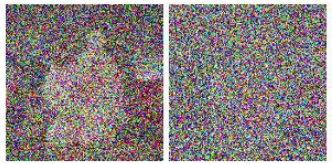
Figure 1: Graphical models for diffusion (left) and non-Markovian (right) inference models.

Content

- Generative Modeling
  - ➢Autoregressive Models
  - ➢Variational AutoEncoders
  - ➢Denoising Diffusion Probabilistic Models
  - ➢Flow Matching Models
- 4D Generative Modeling
  - ➢3D Generation with Multi-view Video Generation
  - ➢4D Generation with Multi-view Video Generation
  - ➢3D Geometry Generation

A Visual Diagram of Distributional Transportation

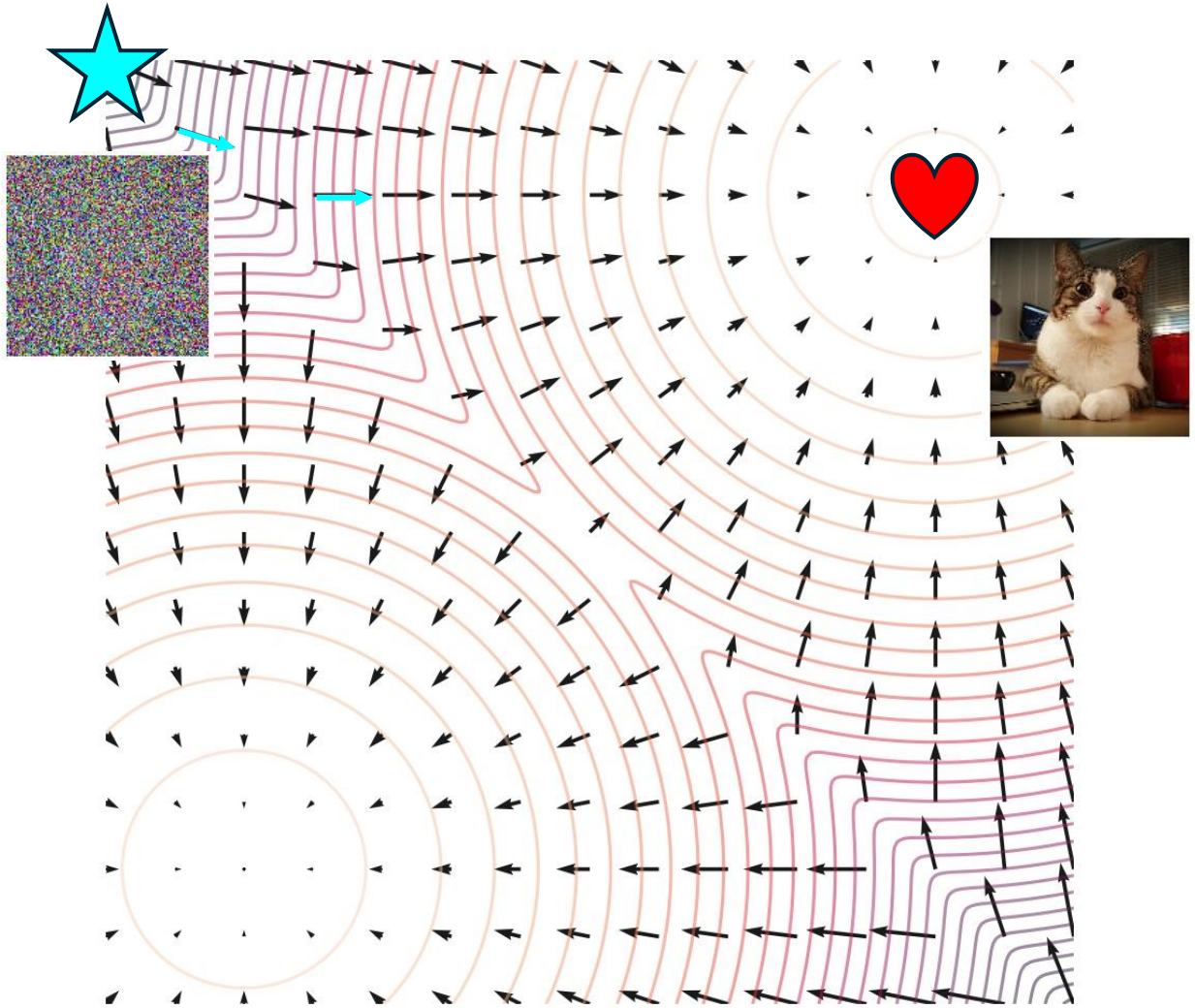




Generative reverse denoising process

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

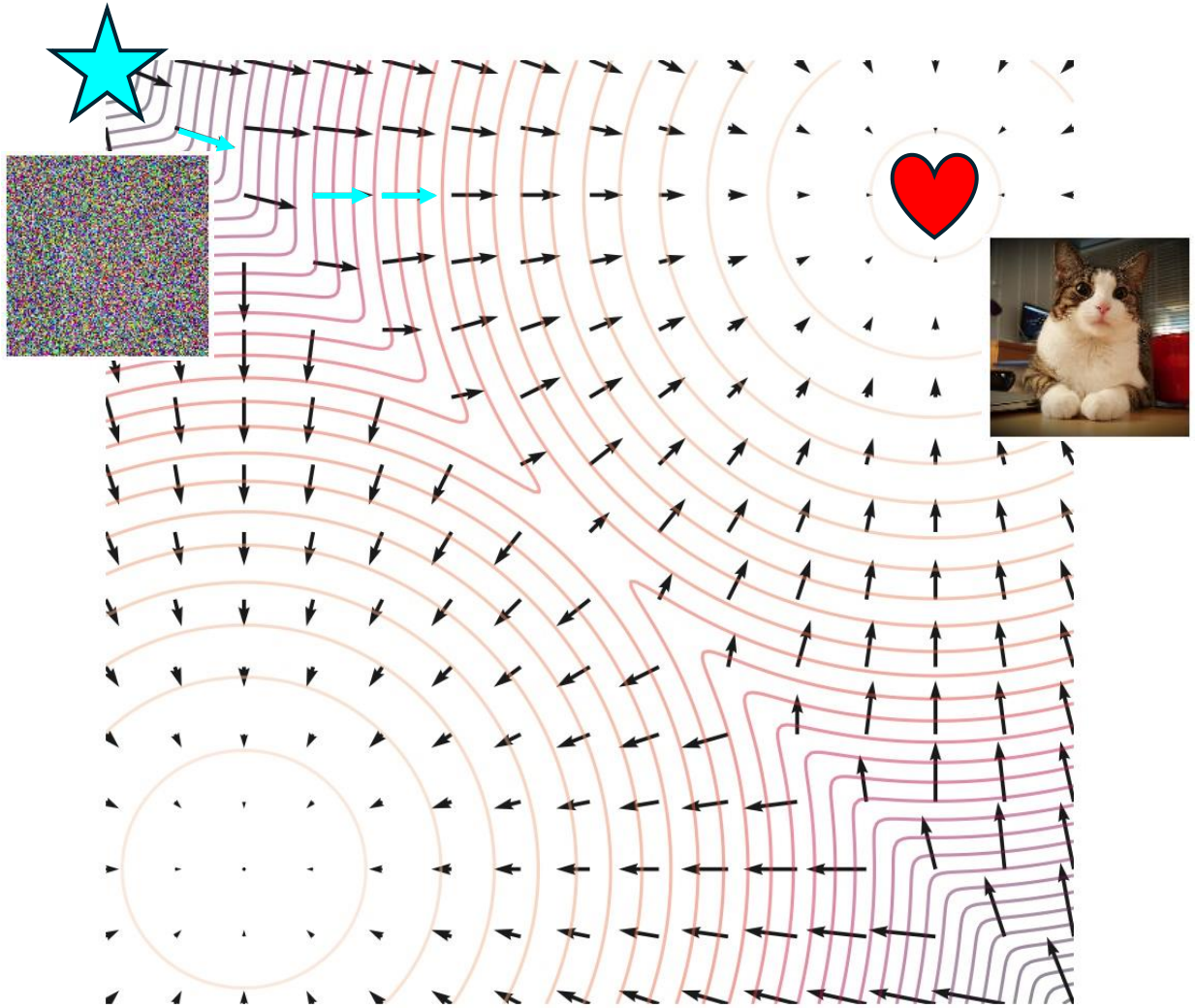
A Visual Diagram of Distributional Transportation



Generative reverse denoising process

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

A Visual Diagram of Distributional Transportation



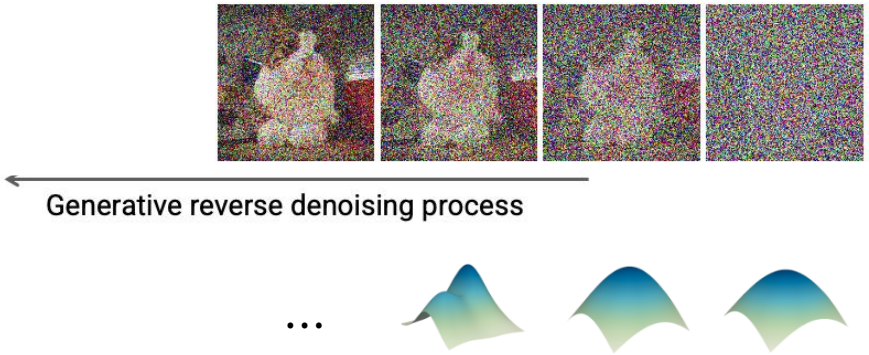
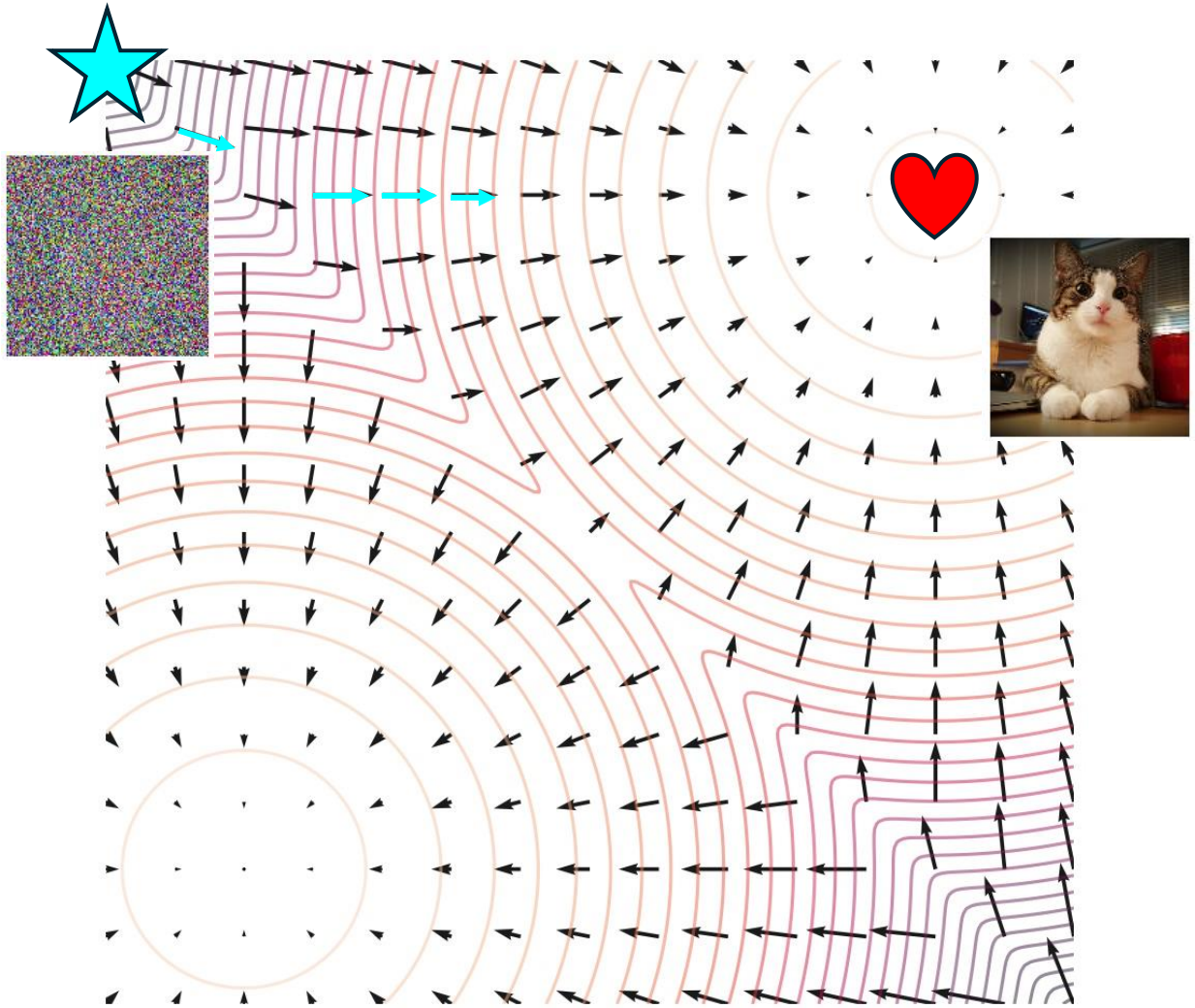


Image credit: https://yang-song.net/blog/2021/score/

A Visual Diagram of Distributional Transportation



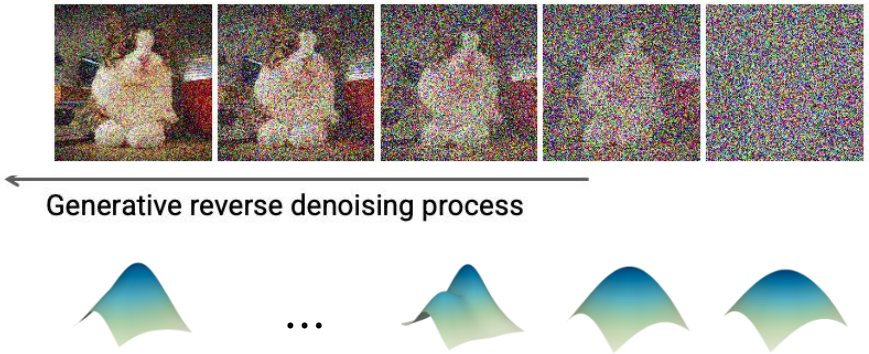
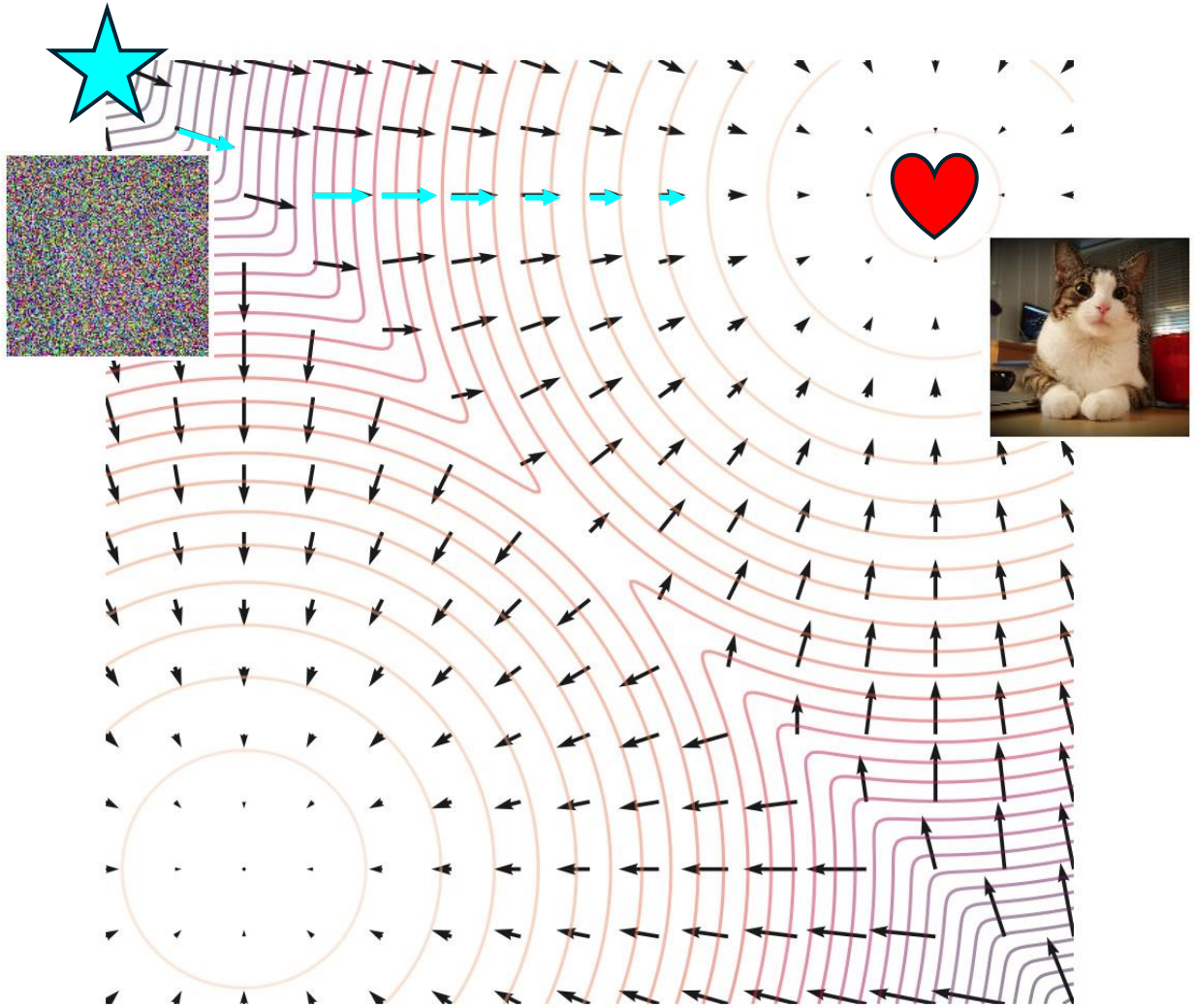


Image credit: https://yang-song.net/blog/2021/score/

A Visual Diagram of Distributional Transportation



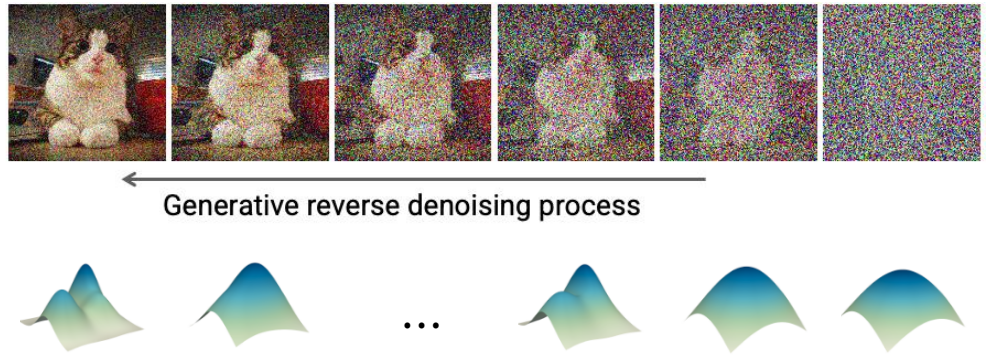
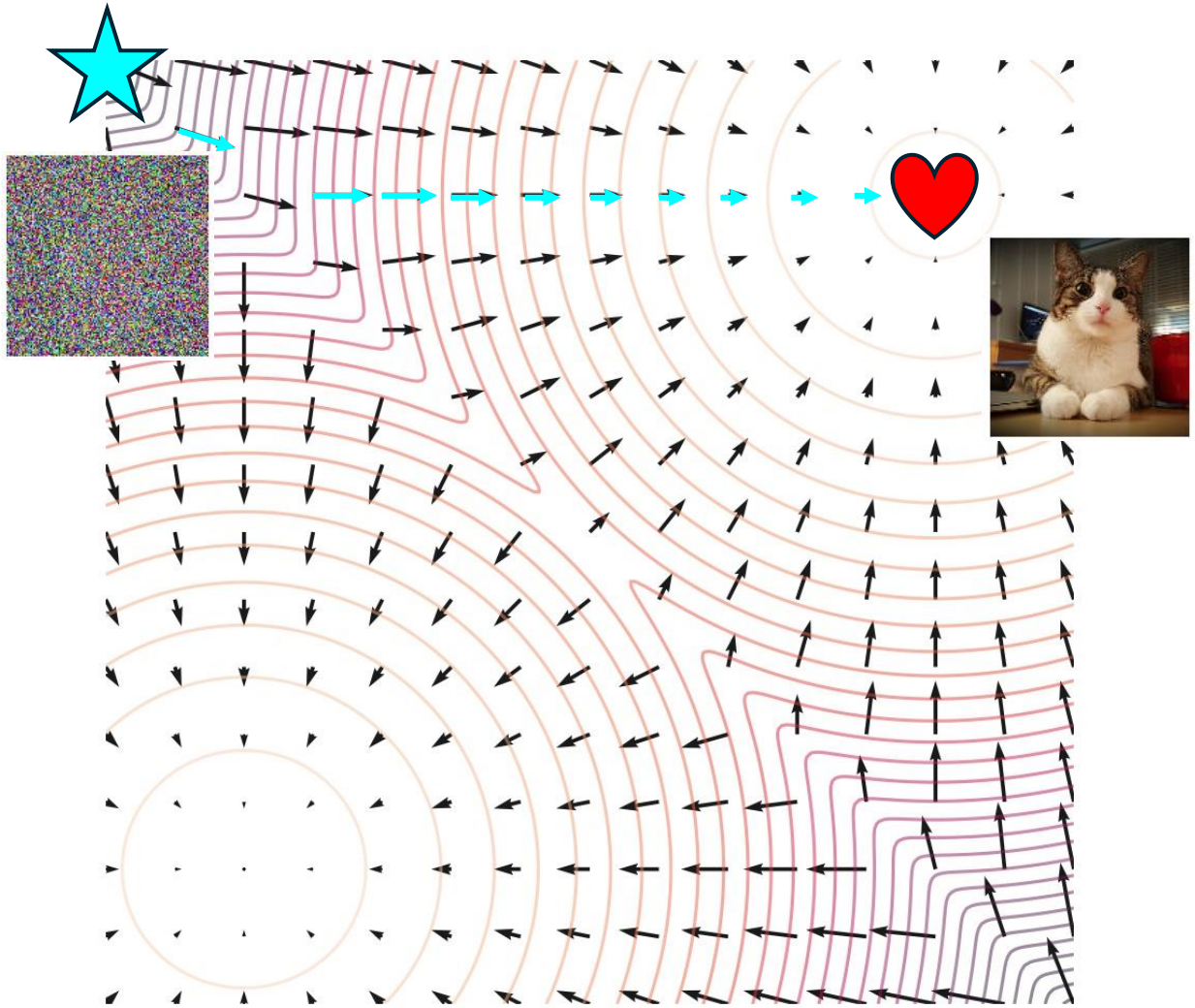


Image credit: https://yang-song.net/blog/2021/score/

A Visual Diagram of Distributional Transportation



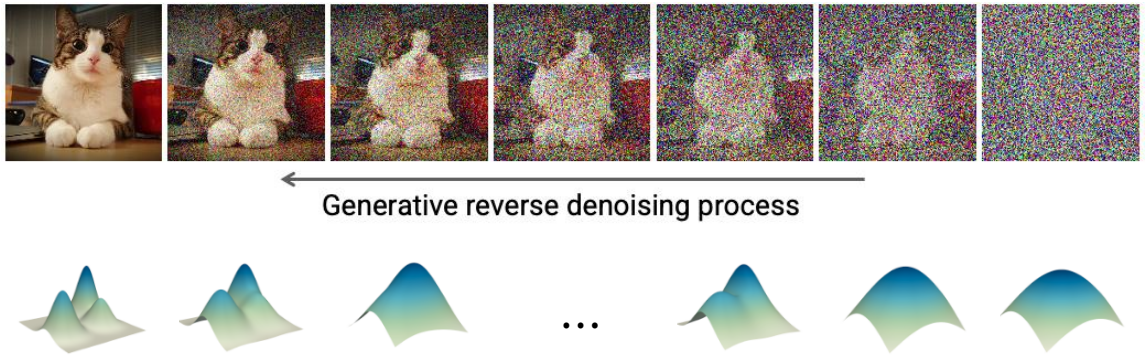


Image credit: https://yang-song.net/blog/2021/score/

A Visual Diagram of Distributional Transportation

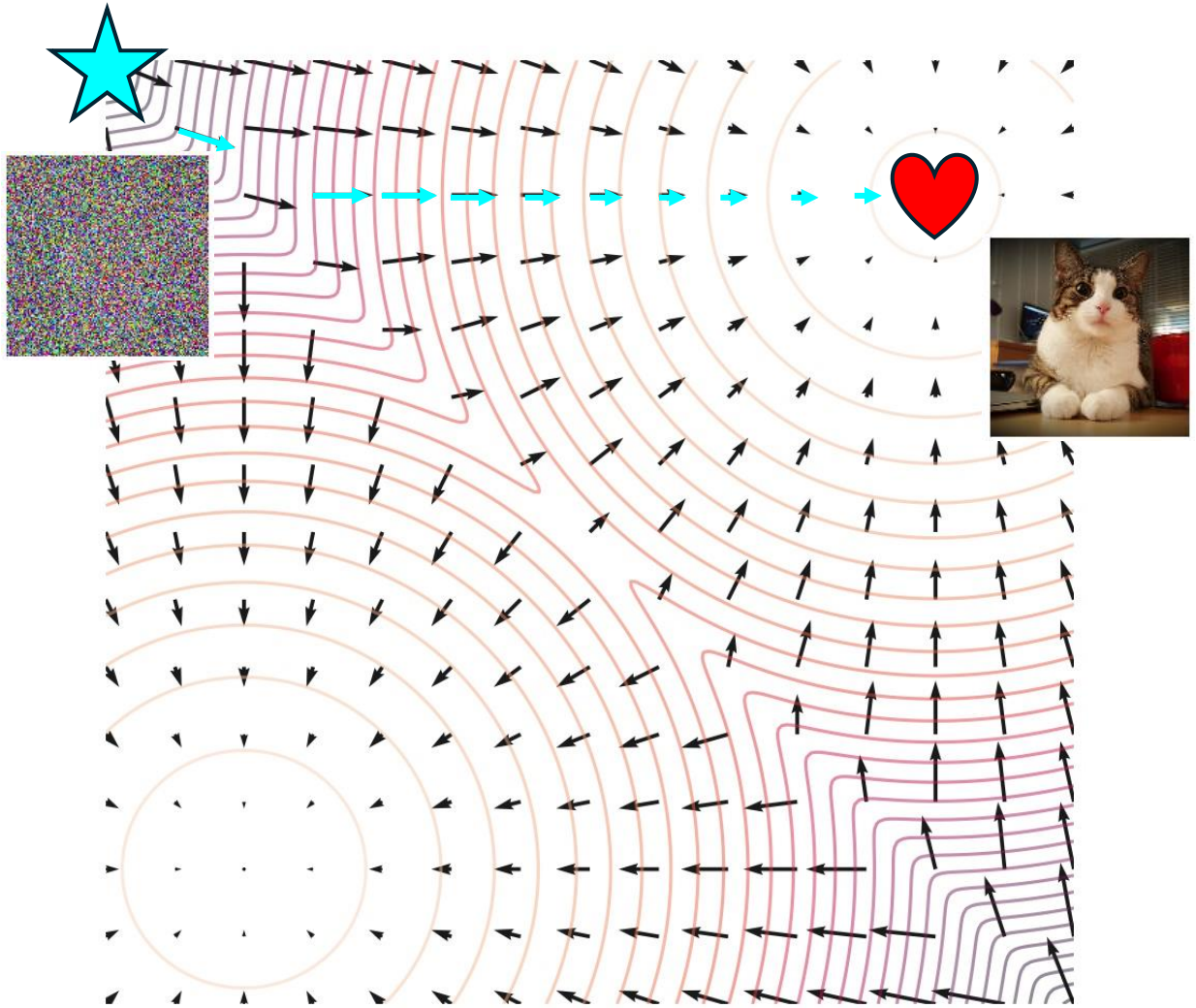
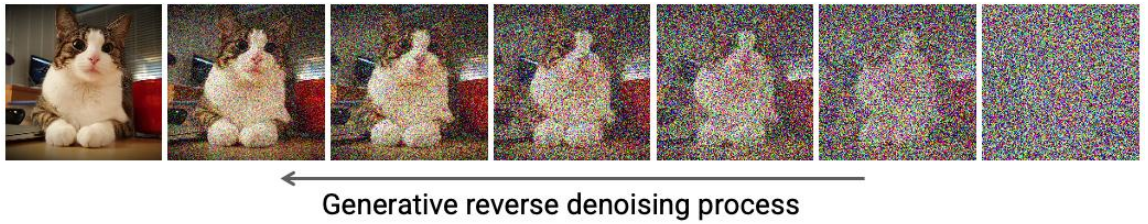


Image credit: https://yang-song.net/blog/2021/score/



- A probabilistic view of generation

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

- A flow view of generation

$$\mathrm{x_0} = \mathrm{x_1} + \int_1^0 \mathrm{v(x_t, t)dt}$$

This is the core idea of flow matching

DDPM vs. Flow Matching

DDPM

- A probabilistic view of generation
- Forward process: Gaussian diffusion process

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) \coloneqq \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

- Backward process: learn a denoising model

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \coloneqq \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

- Sampling steps are discrete: $\mathrm{t} \in [0, \mathrm{T}]$
- Prior distribution is Gaussian

Flow Matching

- A flow view of generation
- Forward process: Any vector field

$$\mathrm{x}_1 = \mathrm{x}_0 - \int_1^0 \mathrm{v}(\mathrm{x}_\mathrm{t}, \mathrm{t})\mathrm{dt}$$

- Backward process: learn a regression model

$$\mathrm{x}_0 = \mathrm{x}_1 + \int_1^0 \mathrm{u}_\theta(\mathrm{x}_\mathrm{t}, \mathrm{t})\mathrm{dt}$$

- Sampling steps are continuous: $\mathrm{t} \sim \mathrm{Uniform}(0, 1)$
- Prior distribution is not restricted to Gaussian

DDPM vs. Flow Matching

Flow matching often assign $\mathbf{t = 0}$ to the prior and $\mathbf{t = 1}$ to the data distribution

DDPM

- A probabilistic view of generation
- Forward process: Gaussian diffusion process

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

- Backward process: learn a denoising model

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

- Sampling steps are discrete: $\mathbf{t} \in [0, \mathbf{T}]$
- Prior distribution is Gaussian

Flow Matching

- A flow view of generation
- Forward process: Any vector field

$$\mathbf{x_0} = \mathbf{x_1} - \int_0^1 \mathbf{v}(\mathbf{x_t}, \mathbf{t})\mathbf{dt}$$

- Backward process: learn a regression model

$$\mathbf{x_1} = \mathbf{x_0} + \int_0^1 \mathbf{u_\theta}(\mathbf{x_t}, \mathbf{t})\mathbf{dt}$$

- Sampling steps are continuous: $\mathbf{t \sim Uniform(0, 1)}$
- Prior distribution is not restricted to Gaussian

A More General Formulation of Generation: Specify Any Vector Field as the Transportation Map

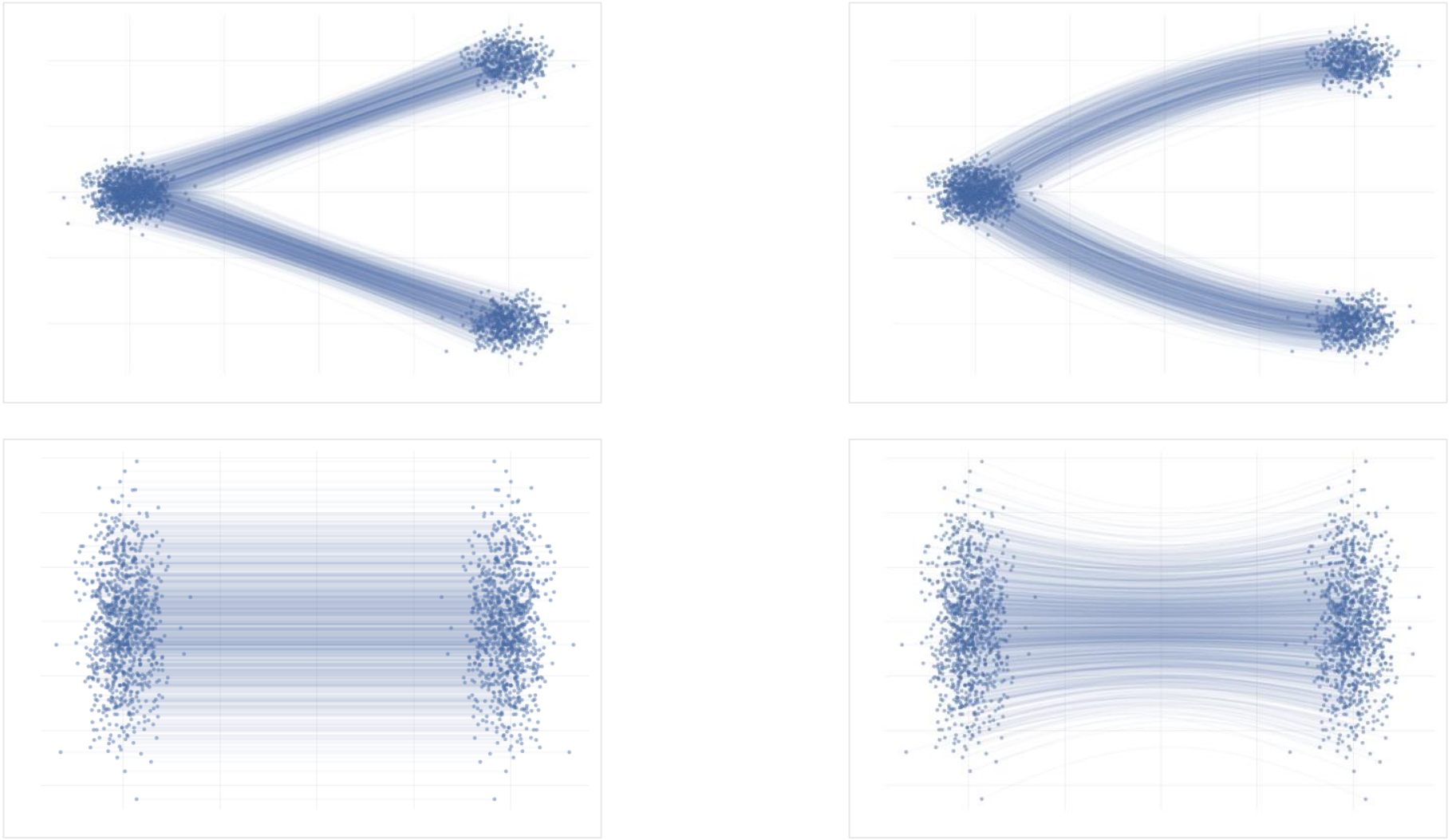
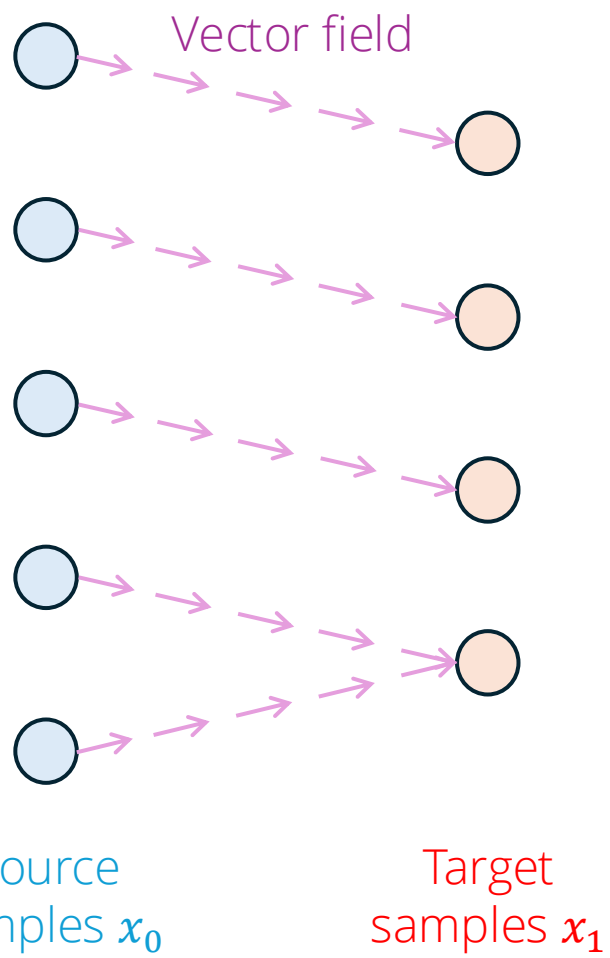


Image credit: https://mlg.eng.cam.ac.uk/blog/2024/01/20/flow-matching.html

However, Specification of the Vector Field is Intractable…

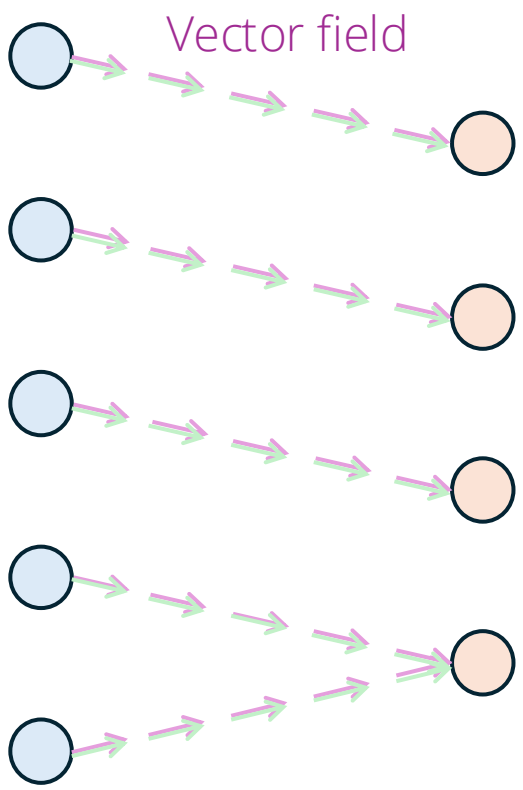


- A vector field $u_t(x)$ transports source samples to target samples
- The objective of Flow Matching: regresses the vector field with a neural network $\mathbf{v_\theta(t, x)}$

$$L_{FM} = \mathbb{E}[\, ||u_t(x) - \mathbf{v_\theta(t, x)}||^2 \,]$$

- However, $u_t(x)$ is often intractable to obtain
  - ➢ The vector field pairs every data with every samples drawn from the prior distribution

Fortunately, the Vector Field can be Composed of a Mixture of Conditional Vector Field



- A vector field $u_t(x)$ transports source samples to target samples
- The objective of Flow Matching: regresses the vector field with a neural network $\mathbf{v_\theta(t, x)}$

$$L_{FM} = \mathbb{E}[\, ||u_t(x) - \mathbf{v_\theta(t, x)}||^2 \,]$$

- Vector field $u_t(x)$ can be composed of a mixture of conditional vector field $u_t(x|x_1)$

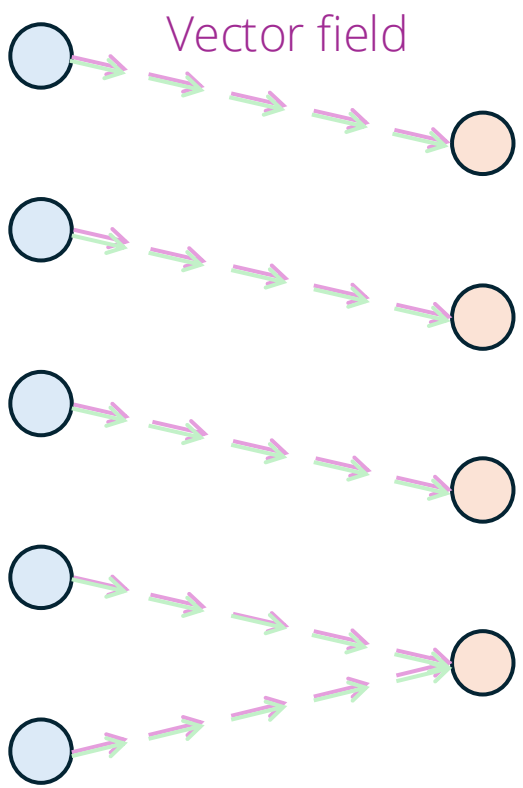
$$u_t(x) = \mathbb{E}_{p(x_1)}\left[\frac{u_t(x|x_1)\mathrm{p}_t(x|x_1)}{\mathrm{p}_t(x)}\right]$$

$u_t(x|x_1)$: the conditional flow between sample x and data x_1

$\mathrm{p}_t(x|x_1)$: the joint distribution of sample x and data x_1

$u_t(x)$: the flow at sample x during time t

Fortunately, the Vector Field can be Composed of a Mixture of Conditional Vector Field



- A vector field $u_t(x)$ transports source samples to target samples
- The objective of Flow Matching: regresses the vector field with a neural network $\mathbf{v_\theta(t, x)}$

$$L_{FM} = \mathbb{E}[\ ||u_t(x) - \mathbf{v_\theta(t, x)}||^2\ ]$$

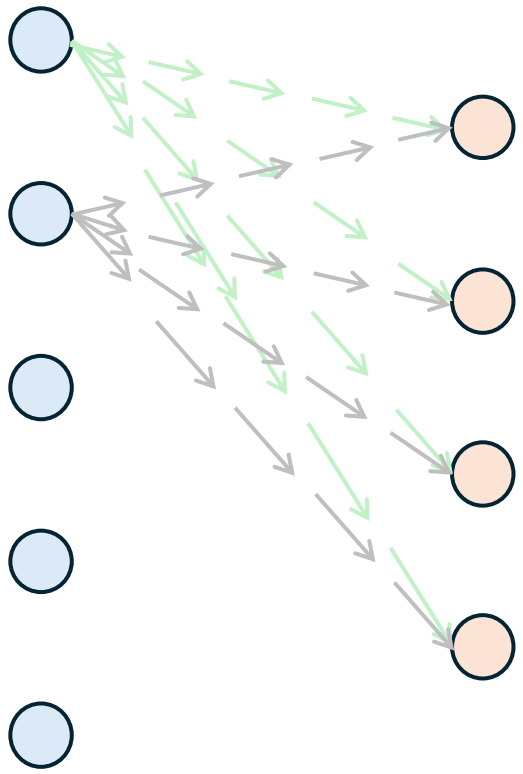
- Vector field $u_t(x)$ can be composed of a mixture of conditional vector field $u_t(x|x_1)$

$$u_t(x) = \mathbb{E}_{p(x_1)}\left[\frac{u_t(x|x_1)\mathrm{p}_t(x|x_1)}{\mathrm{p}_t(x)}\right]$$

- We have:

$$\nabla_\theta \mathbb{E}[\ ||u_t(x) - \mathbf{v_\theta(t, x)}||^2\ ] = \nabla_\theta \mathbb{E}[\ ||u_t(x|x_1) - \mathbf{v_\theta(t, x)}||^2\ ]$$

A Common Practice to Compose the Vector Field



Source samples x_0

Target samples x_1

- In practice, we often
  - Adopt Gaussian $\mathcal{N}(\mathbf{0}, \mathrm{I})$ as the prior distribution
  - Randomly pair source samples and target samples
  - Assign optimal transportation (straight velocity) as the conditional flow for the source-target pair

$$u_t(x) = \mathrm{x}_1 - \mathrm{x}_0$$

$$\mathrm{x}_\mathrm{t} = (1 - \mathrm{t})\mathrm{x}_0 + \mathrm{t}\mathrm{x}_1$$

Flow Matching: Learn to Regress the Vector Field

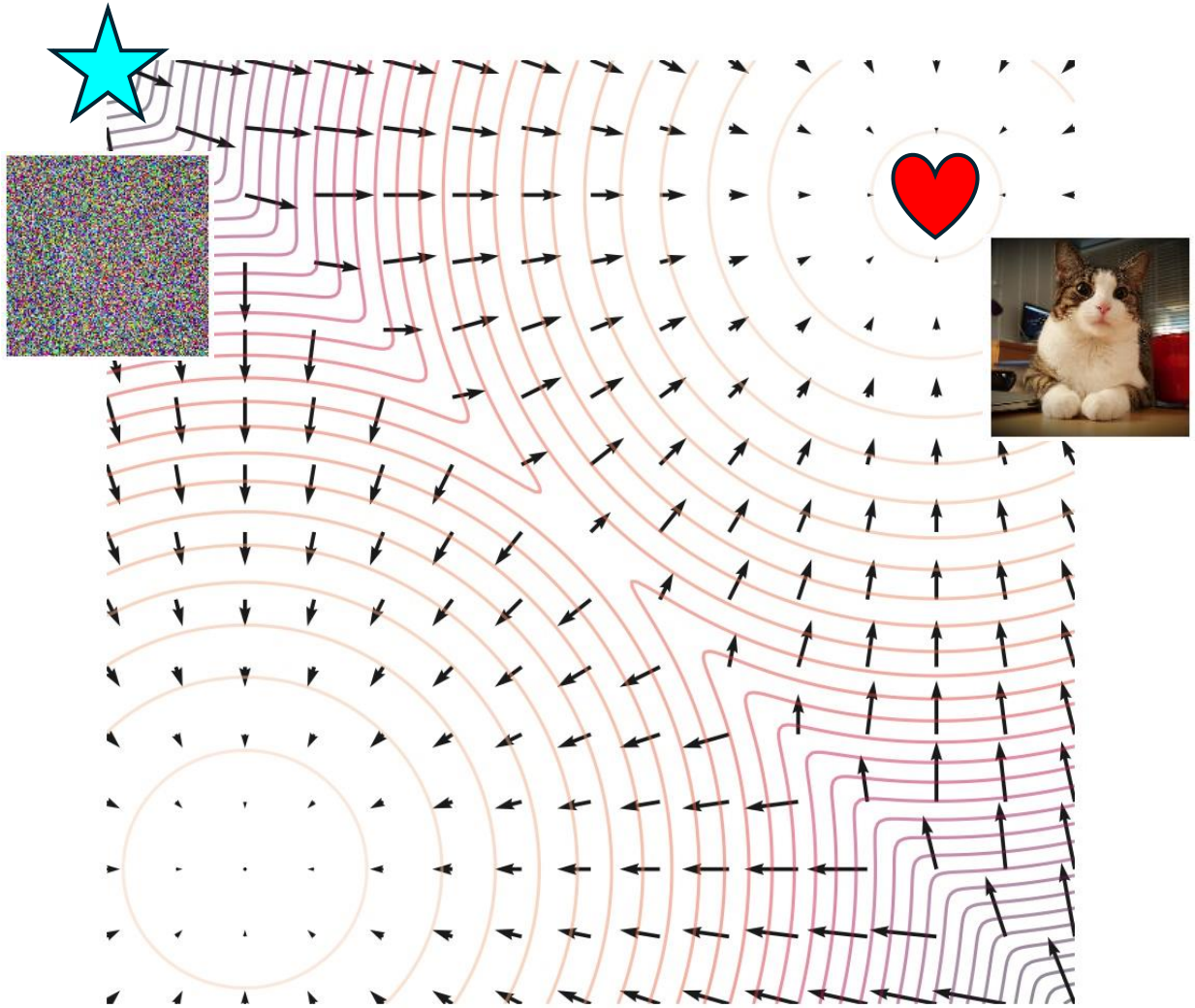
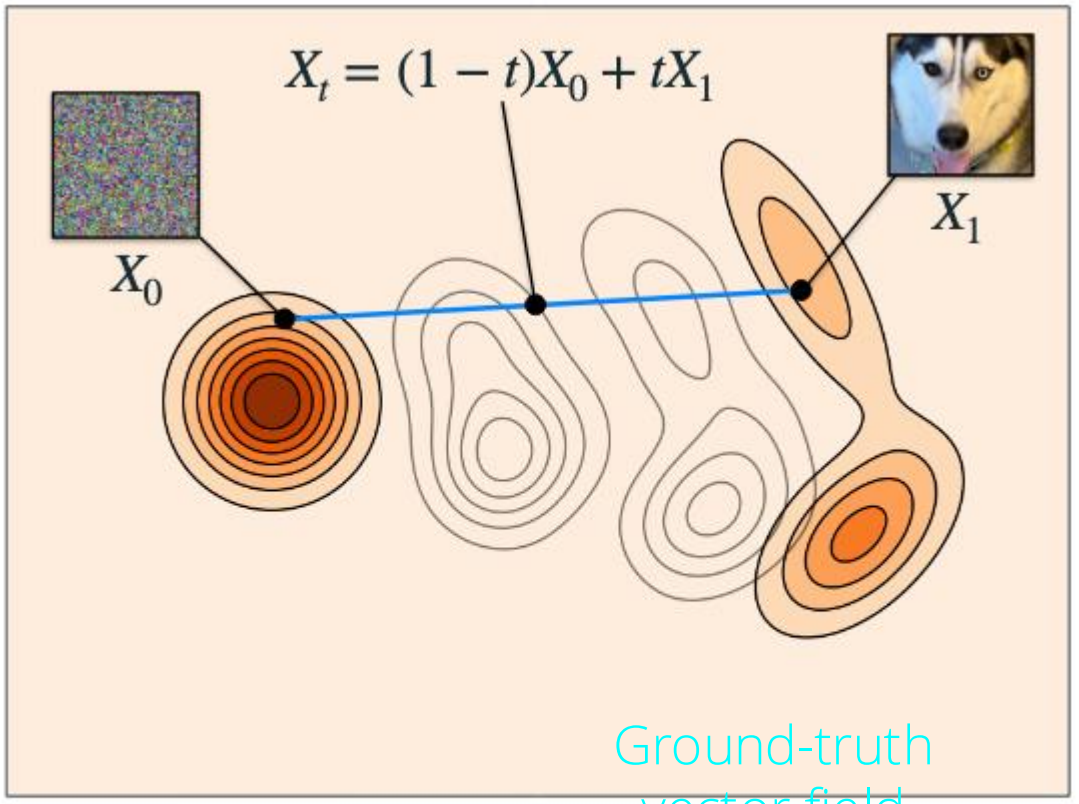


Image credit: https://yang-song.net/blog/2021/score/

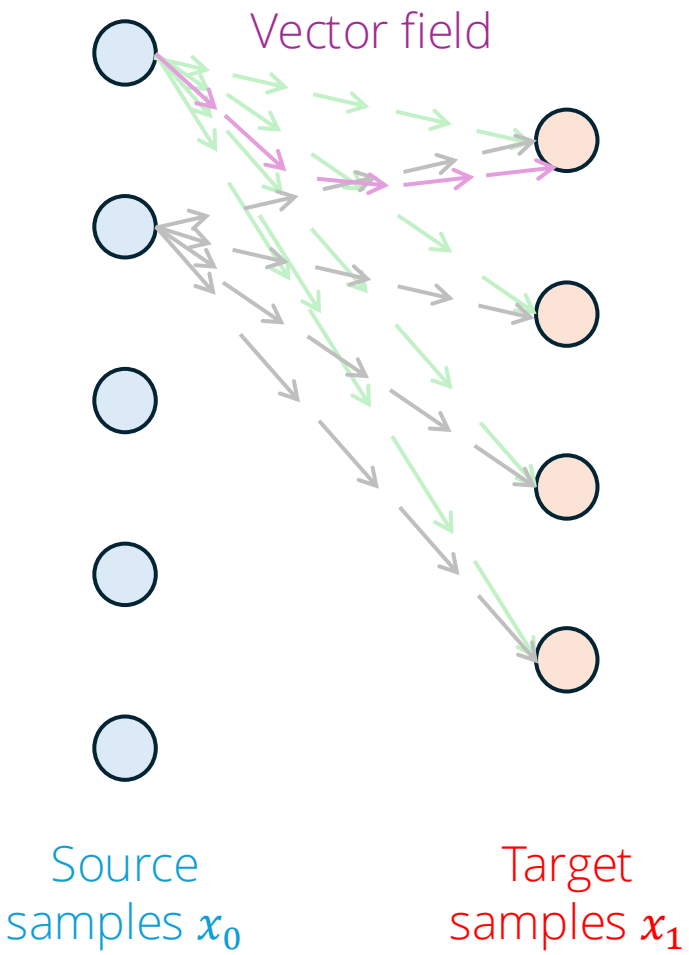


Ground-truth vector field

$$\mathbb{E}_{t,X_0,X_1}\left\| u_t^{\theta}(X_t) - (X_1 - X_0) \right\|^2$$

Estimated vector field

A Common Practice to Compose the Vector Field



- In practice, we often
  - Adopt Gaussian $\mathcal{N}(\mathbf{0}, \mathrm{I})$ as the prior distribution
  - Randomly pair source samples and target samples
  - Assign optimal transportation (straight velocity) as the conditional flow for the source-target pair
- Since vector field $u_t(x)$ is composed of a mixture of conditional vector field $u_t(x|x_1)$, which becomes curved under random coupling

The Target Vector Field vs. Learned Vector Field

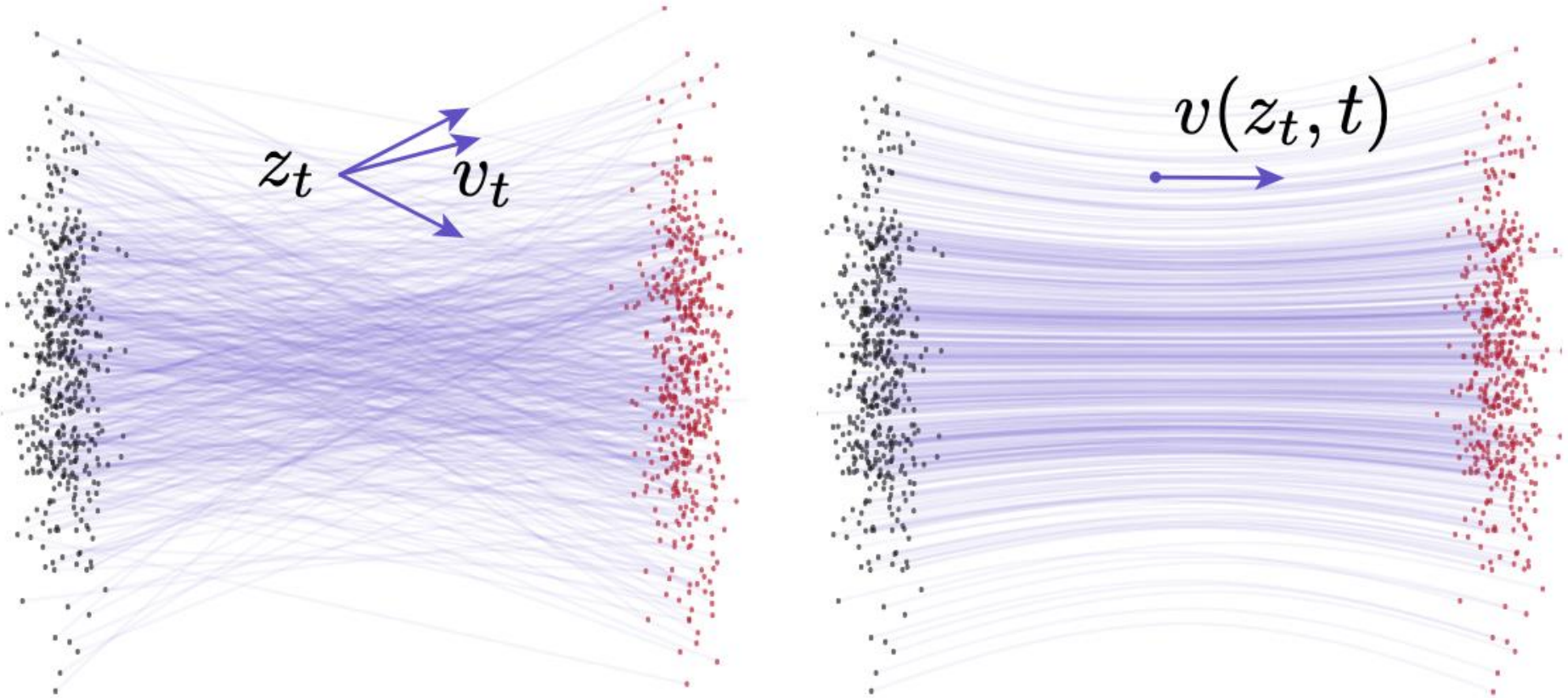


Image credit: Mean Flows for One-step Generative Modeling.

Flow Matching is a General Formulation that Recovers Diffusion Models if the Vector Field is Specified Correctly

Example I: Diffusion conditional VFs. Diffusion models start with data points and gradually add noise until it approximates pure noise. These can be formulated as stochastic processes, which have strict requirements in order to obtain closed form representation at arbitrary times t, resulting in Gaussian conditional probability paths $p_t(x|x_1)$ with specific choices of mean $\mu_t(x_1)$ and std $\sigma_t(x_1)$ (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2020b). For example, the reversed (noise→data) Variance Exploding (VE) path has the form

$$p_t(x) = \mathcal{N}(x|x_1, \sigma_{1-t}^2 I), \tag{16}$$

where σ_t is an increasing function, $\sigma_0 = 0$, and $\sigma_1 \gg 1$. Next, equation 16 provides the choices of $\mu_t(x_1) = x_1$ and $\sigma_t(x_1) = \sigma_{1-t}$. Plugging these into equation 15 of Theorem 3 we get

$$u_t(x|x_1) = -\frac{\sigma'_{1-t}}{\sigma_{1-t}}(x - x_1). \tag{17}$$

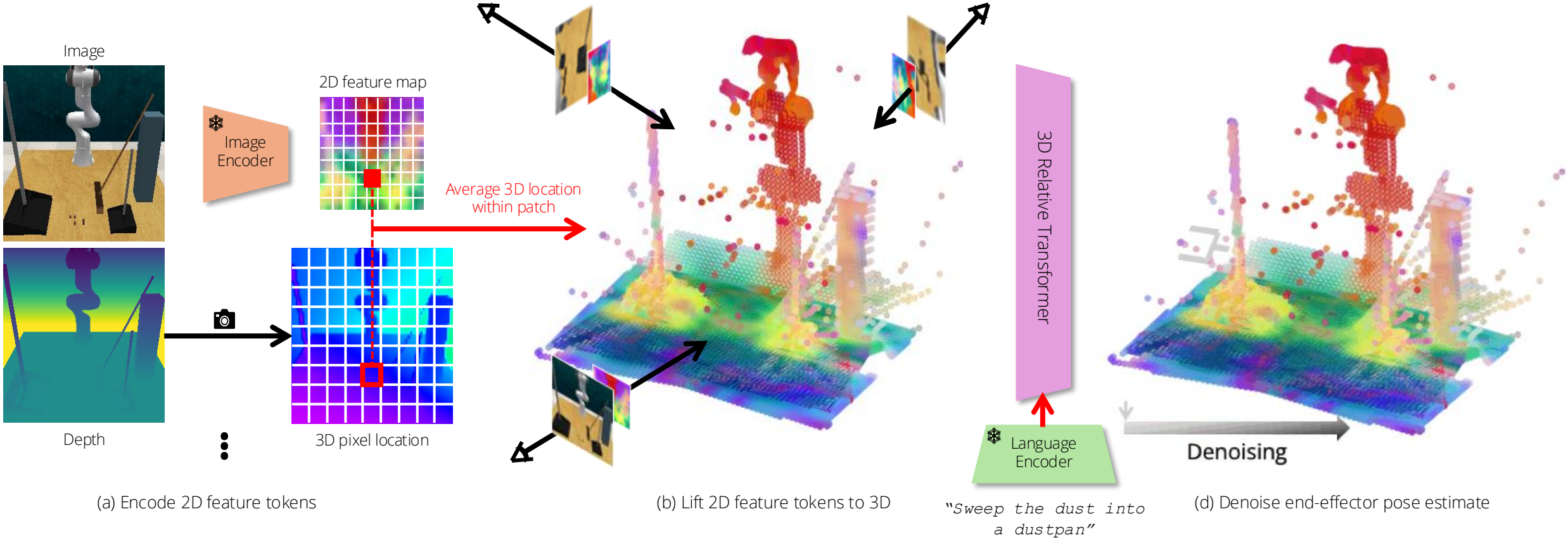
The reversed (noise→data) Variance Preserving (VP) diffusion path has the form

$$p_t(x|x_1) = \mathcal{N}(x \,|\, \alpha_{1-t}x_1, \left(1 - \alpha_{1-t}^2\right) I), \text{ where } \alpha_t = e^{-\frac{1}{2}T(t)}, T(t) = \int_0^t \beta(s)ds, \tag{18}$$

and β is the noise scale function. Equation 18 provides the choices of $\mu_t(x_1) = \alpha_{1-t}x_1$ and $\sigma_t(x_1) = \sqrt{1 - \alpha_{1-t}^2}$. Plugging these into equation 15 of Theorem 3 we get

$$u_t(x|x_1) = \frac{\alpha'_{1-t}}{1 - \alpha_{1-t}^2}\left(\alpha_{1-t}x - x_1\right) = -\frac{T'(1-t)}{2}\left[\frac{e^{-T(1-t)}x - e^{-\frac{1}{2}T(1-t)}x_1}{1 - e^{-T(1-t)}}\right]. \tag{19}$$

Why Flow Matching?



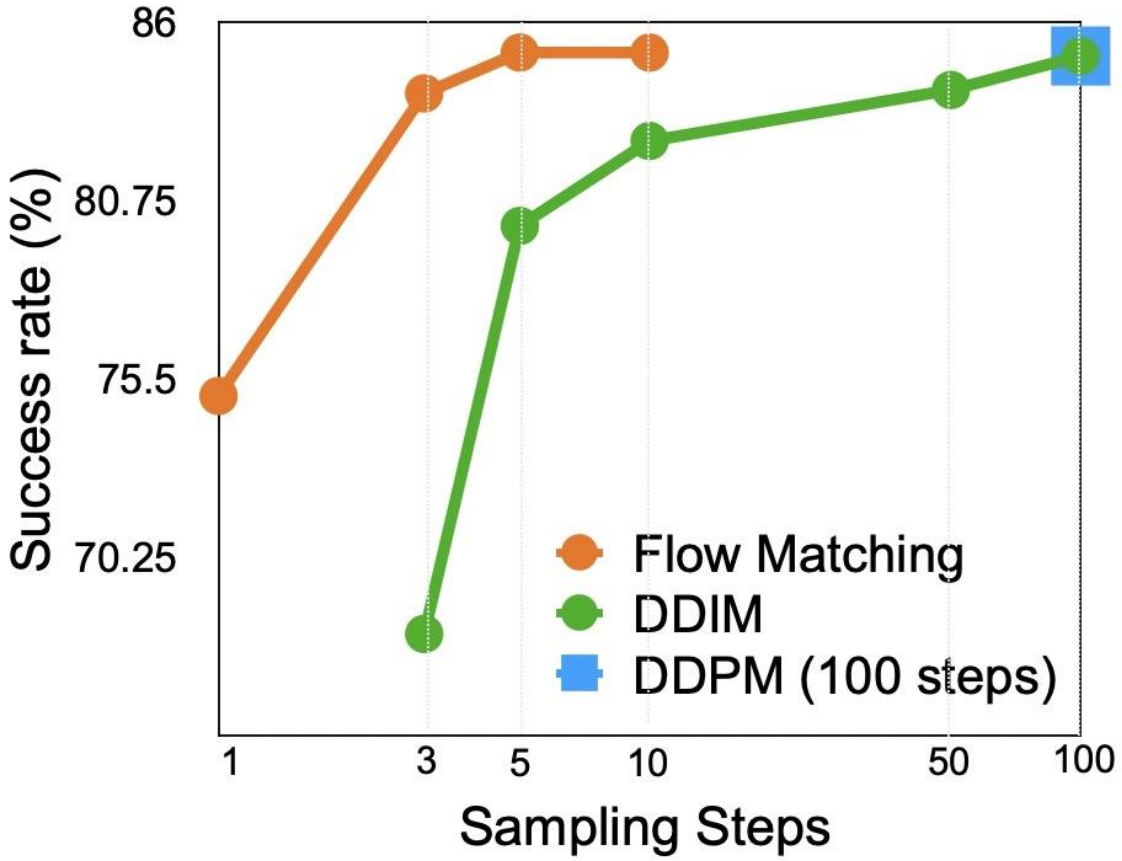
(a) Encode 2D feature tokens

(b) Lift 2D feature tokens to 3D

(c) Encode language instructions

(d) Denoise end-effector pose estimate

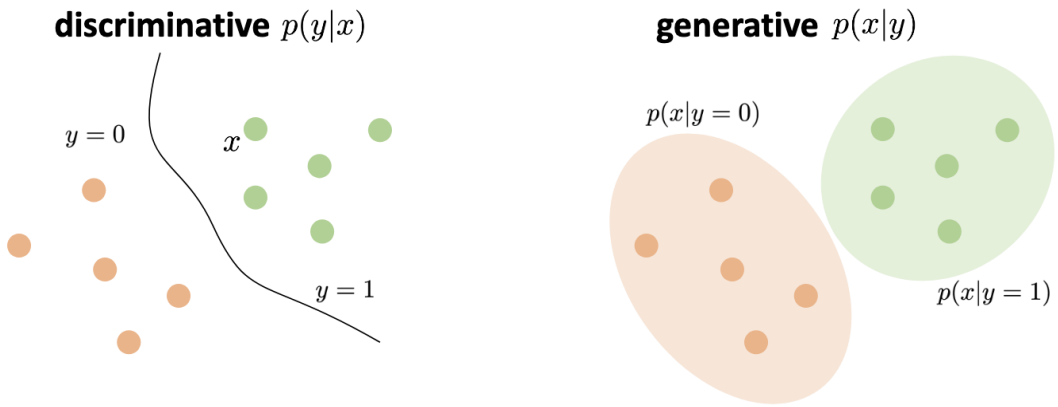
Flow Matching Requires Much Fewer Sampling Steps than DDPM



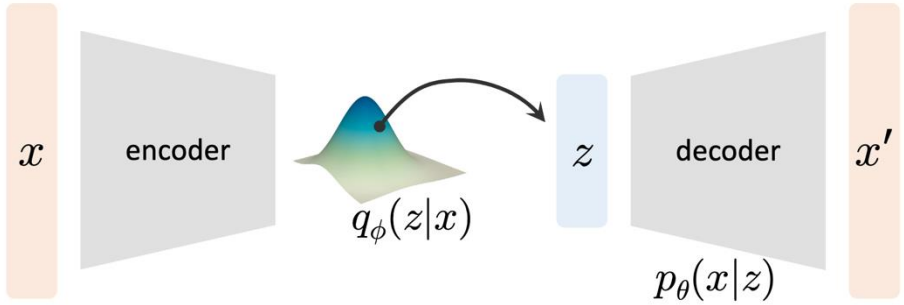
3D FlowMatch Actor: Unified 3D Policy for Single- and Dual-Arm Manipulation. Gkanatsios et al.

Summary

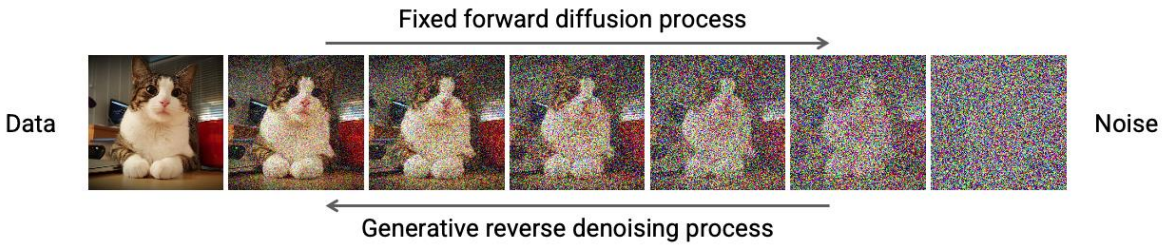
Discriminative Models vs. Generative Models



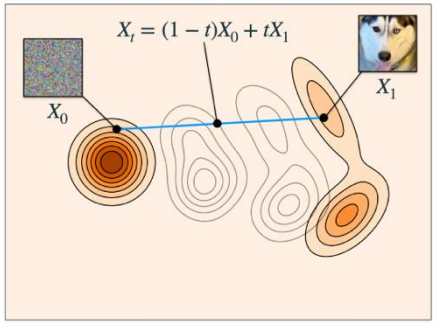
Variational AutoEncoders



Diffusion Models



Flow Matching



$$\mathbb{E}_{t,X_0,X_1}\left\|u_t^\theta(X_t) - (X_1 - X_0)\right\|^2$$

Autoregressive Models