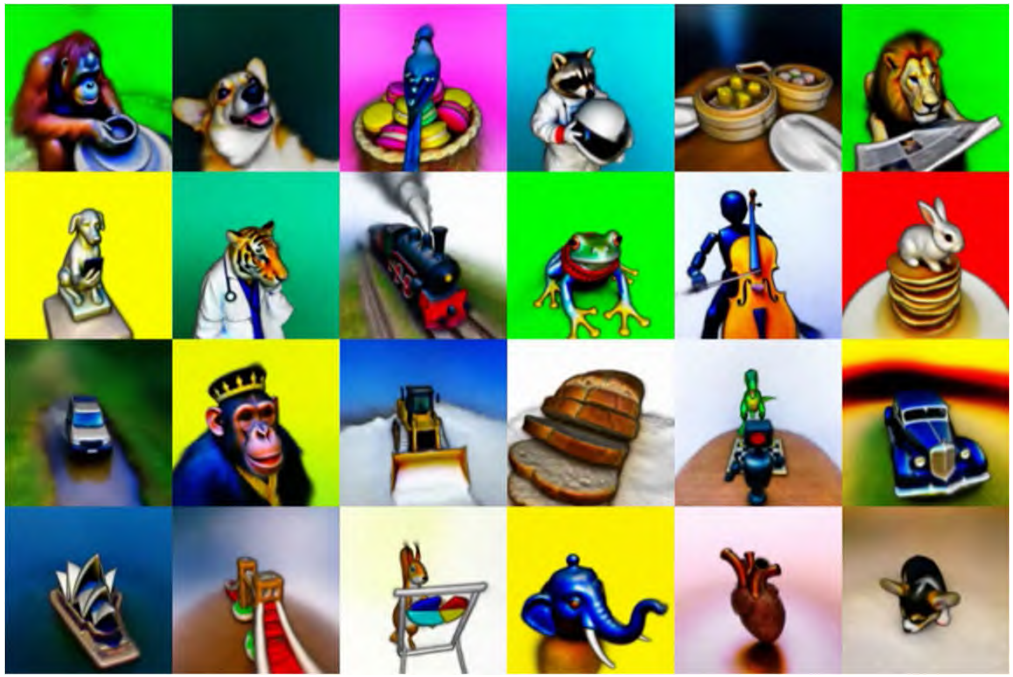
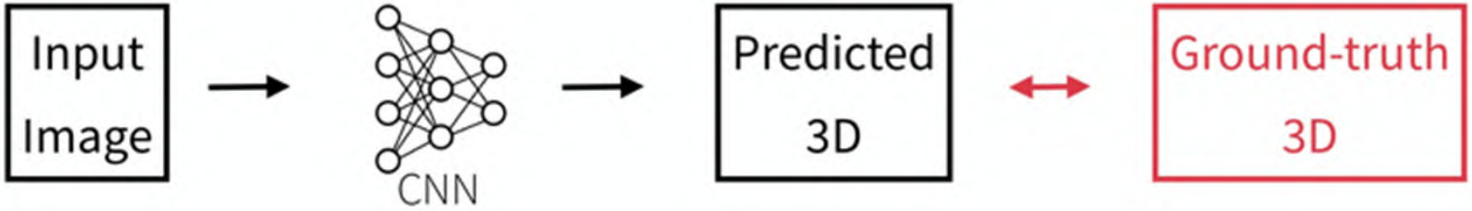
# 3D from 2D Diffusion Guidance

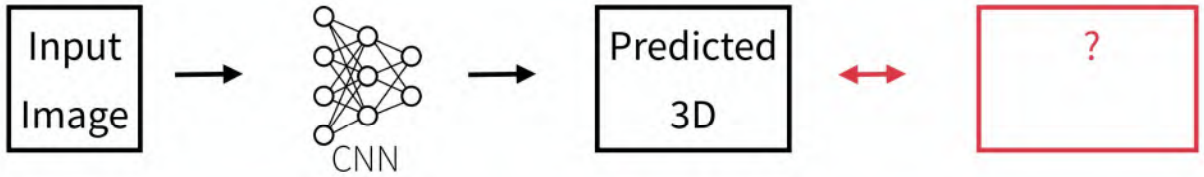
# Review Single-view Learning



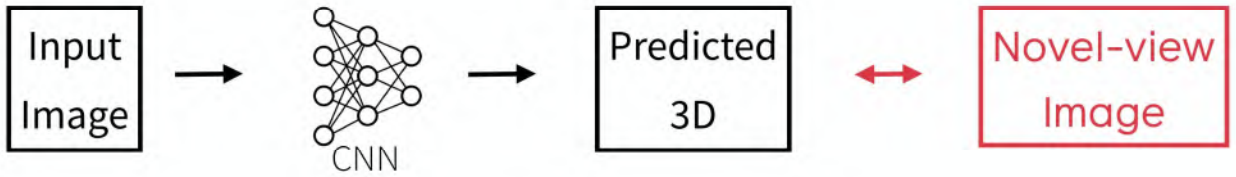

合成数据造型简单

难以完美泛化到真实数据

# Learning without 3D Supervision

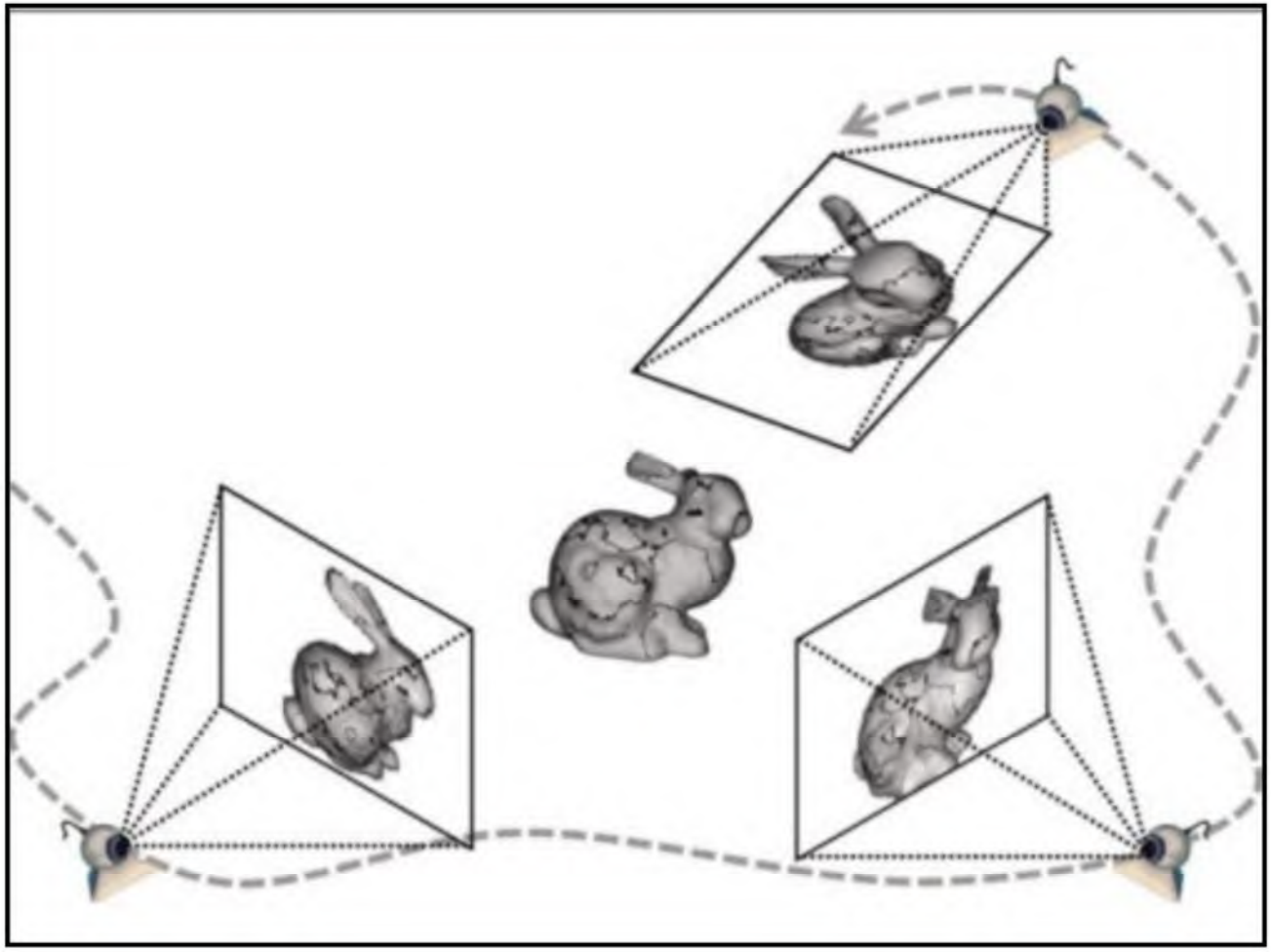



# Learning 3D from Indirect Supervision




Can we learn without 3D Supervision?

# Learning 3D via Multi-view Supervision



# Learning 3D via Multi-view Supervision

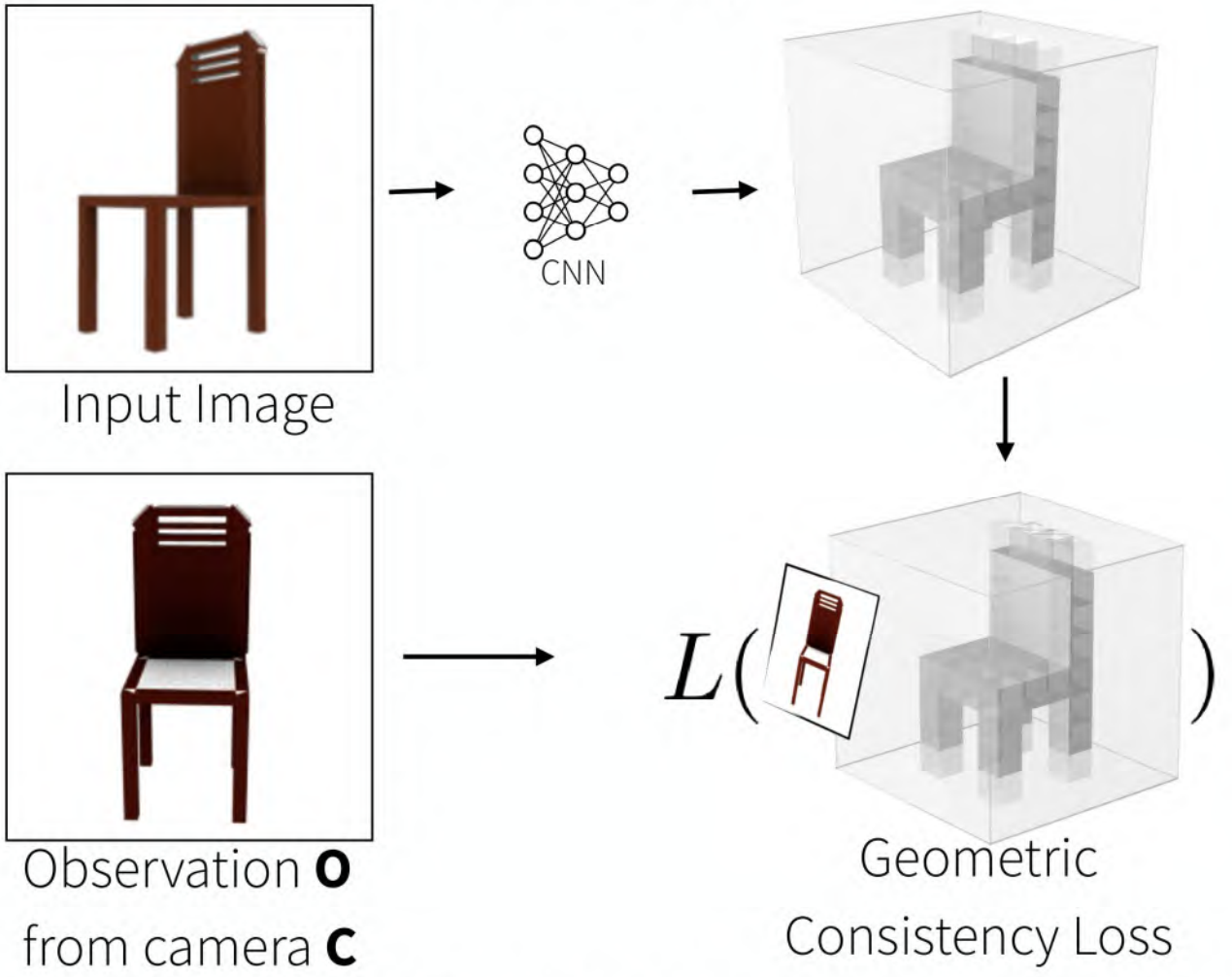



*Image Credits: Multi-view Supervision for Single-view Reconstruction via Differentiable Ray Consistency. Tulsiani et. al.*

# Learning 3D via Multi-view Supervision

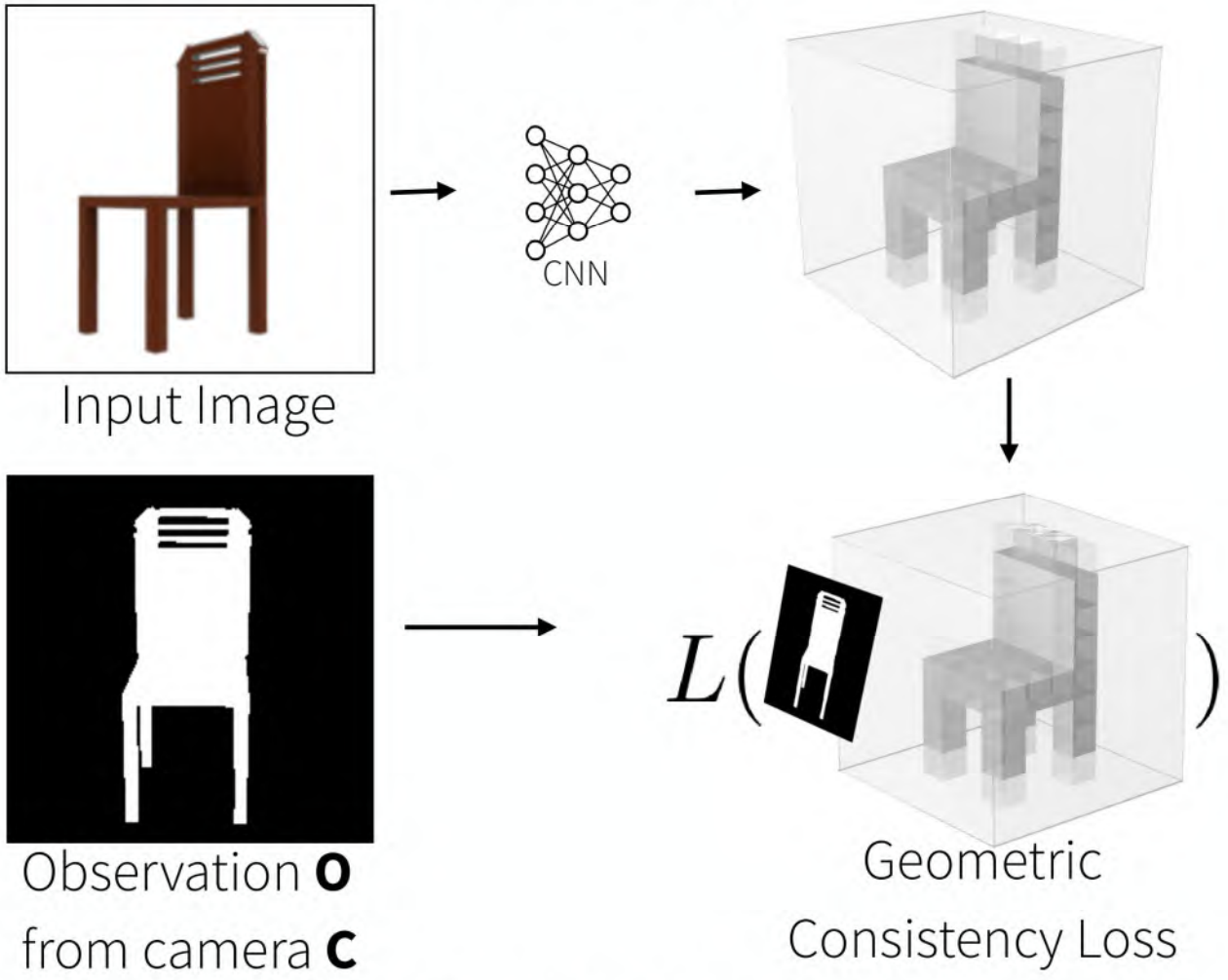
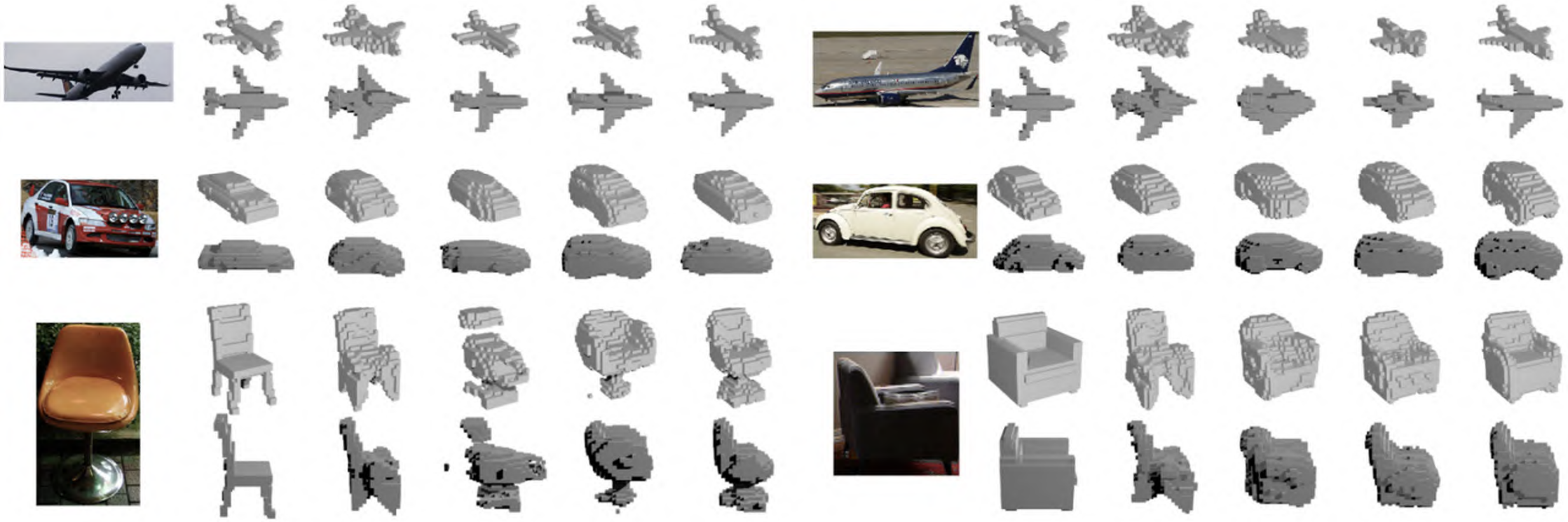


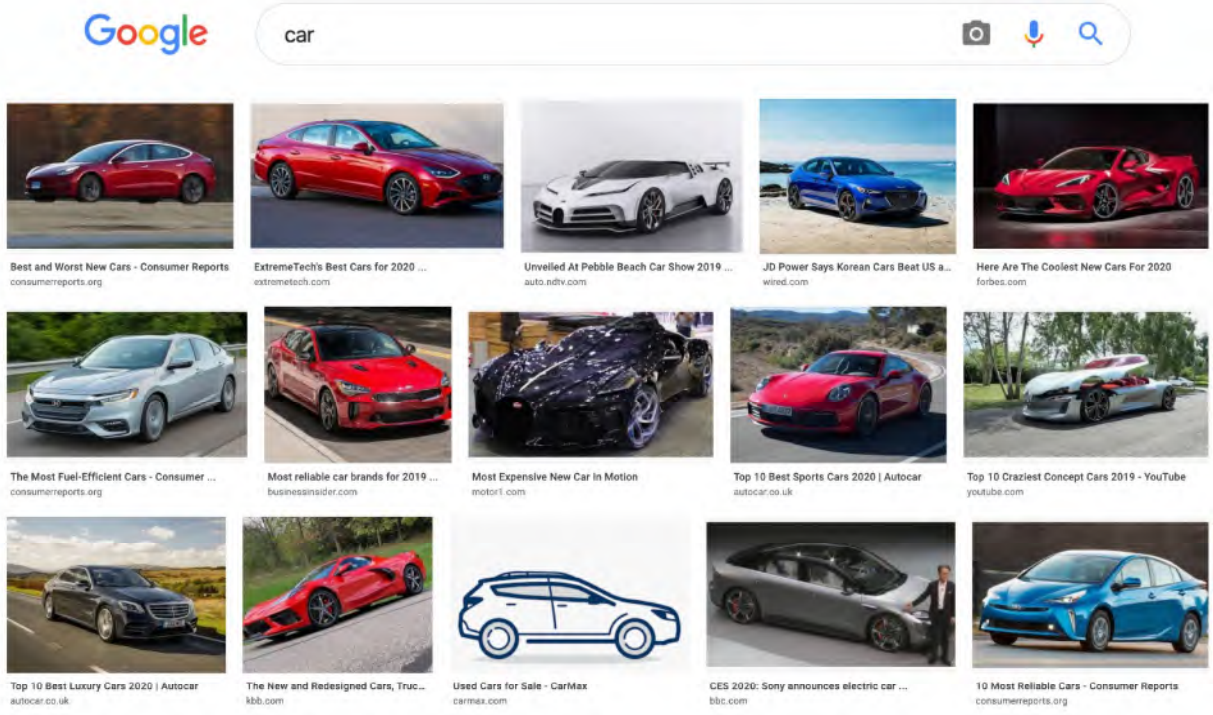

*Image Credits: Multi-view Supervision for Single-view Reconstruction via Differentiable Ray Consistency. Tulsiani et. al.*



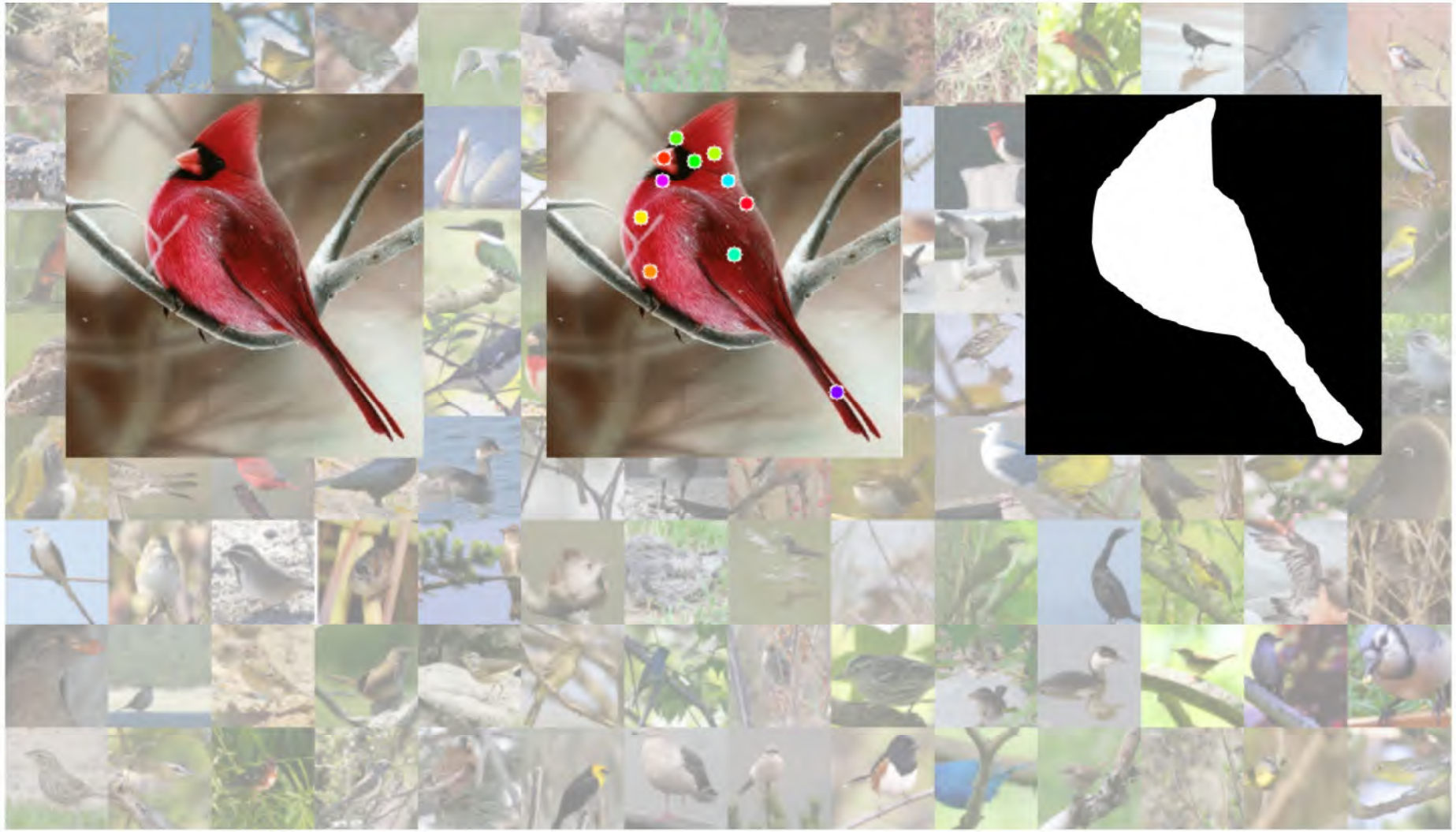
（大量）多视角数据也很难获取

能够大量获取的还是单视角数据

# Learning 3D from Image Collections



# Learning 3D from Image Collections



# Learning 3D from Image Collections

CNN

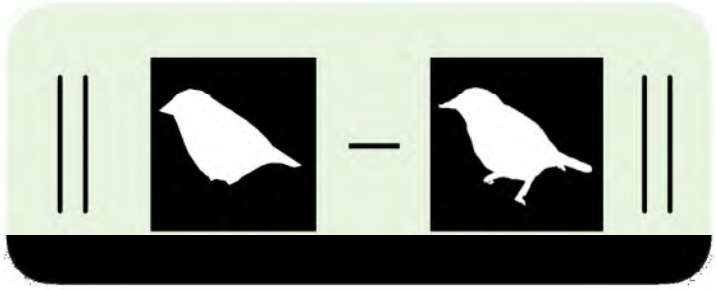
=

Learned Mean Shape

+

Shape Deformation

Camera

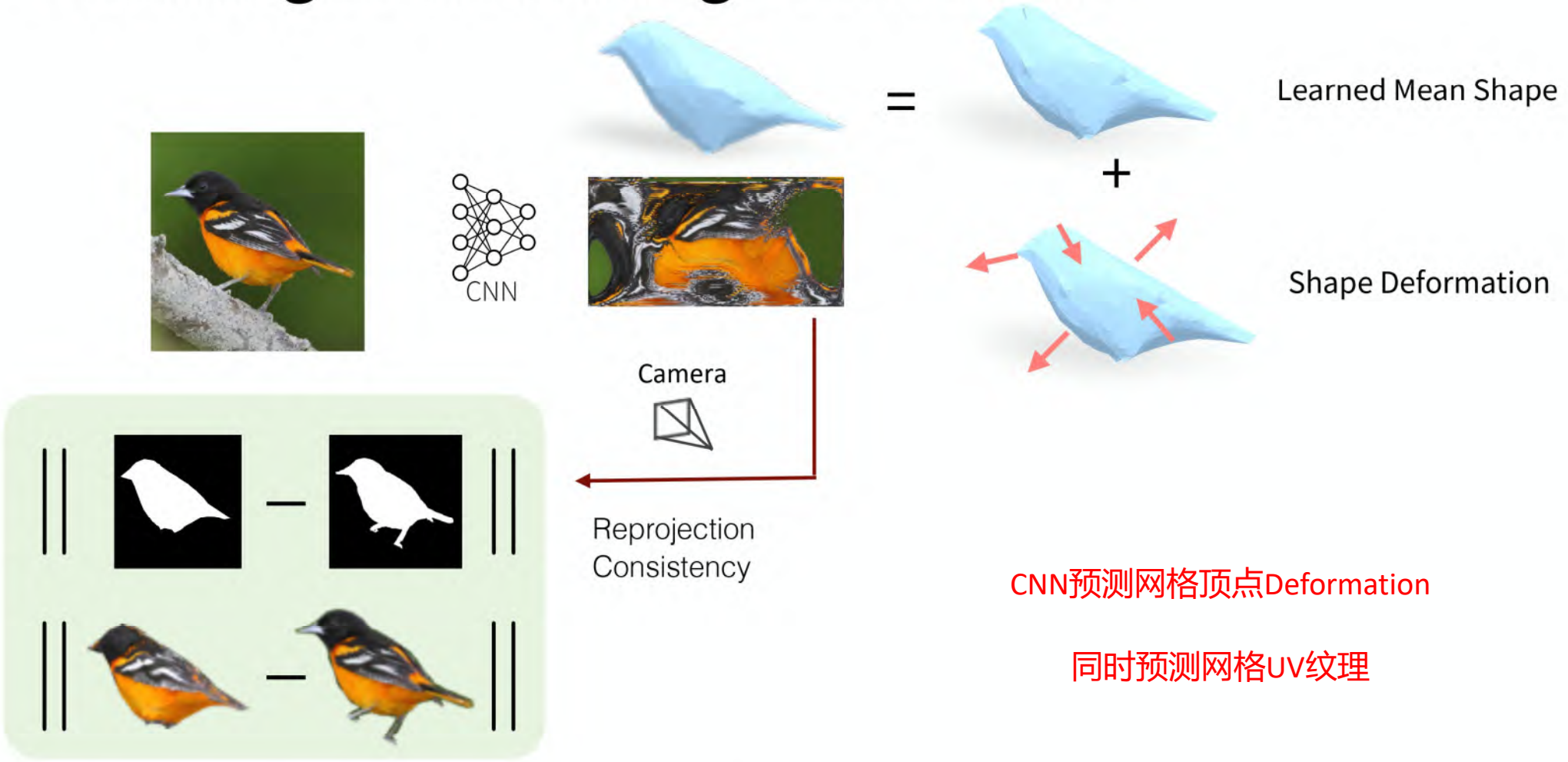


Reprojection
Consistency

CNN预测网格顶点Deformation

*Image Credits:* Learning Category-Specific Mesh Reconstruction from Image Collections. *Kanazawa* et. al.

# Learning 3D from Image Collections




CNN预测网格顶点Deformation

同时预测网格UV纹理

*Image Credits:* Learning Category-Specific Mesh Reconstruction from Image Collections. *Kanazawa* et. al.

# Learning 3D from Image Collections

*Image Credits:* Learning Category-Specific Mesh Reconstruction from Image Collections. *Kanazawa* et. al.

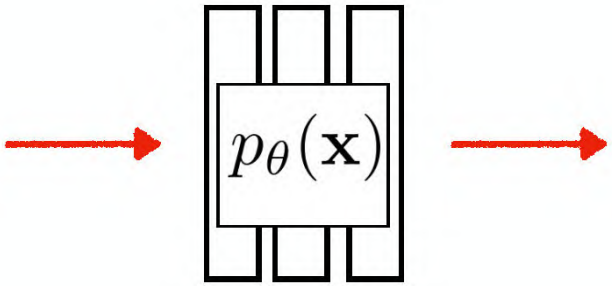
3D from single-view still far from realistic

But 2D generative models could be very realistic

Can we generate 3D from 2D generative models?

# Review Generative Models

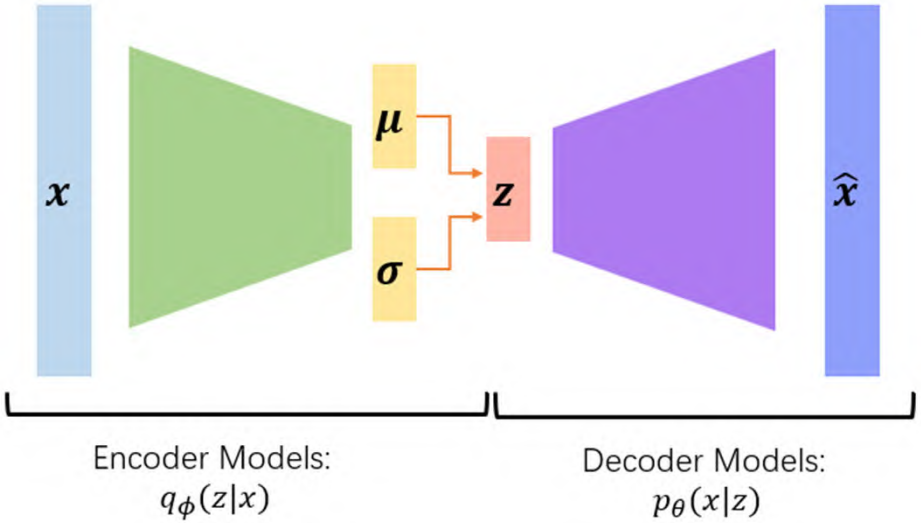
$\mathcal{D}$

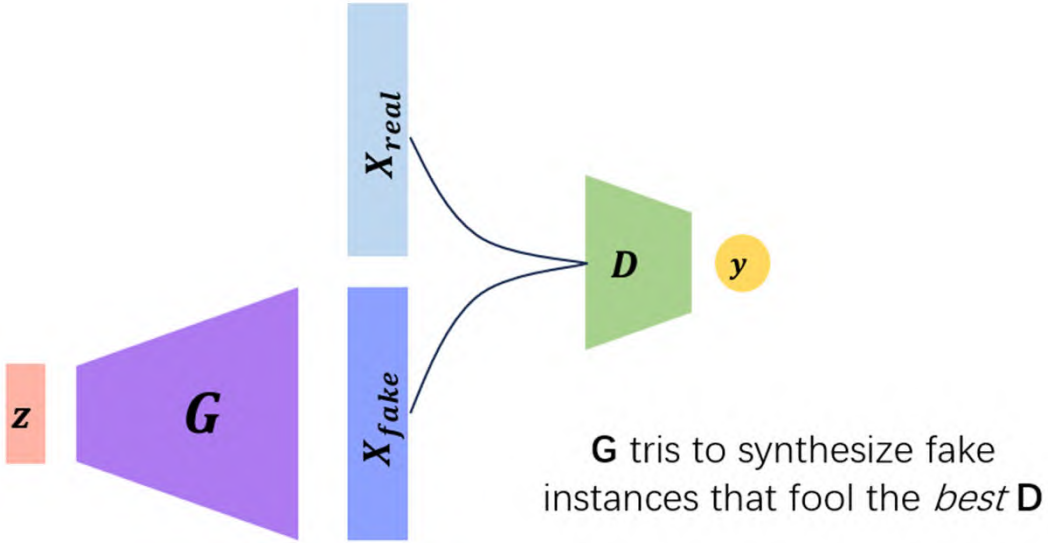



$\mathbf{x} \sim p_\theta(\cdot)$

**Goal:** Given a dataset of samples of a variable of interest (e.g. images), learn a (parametrized) distribution that allows drawing new samples

# Review Generative Models




VAEs and GANs
are not the best for image generation now

# Diffusion Models — The best for now

a space elevator, cinematic scifi art

A cheeseburger with juicy beef patties and melted cheese sits on top of a toilet that looks like a throne and stands in the middle of the royal chamber.

a hole in the floor of my bathroom with small gremlins living in it

a small office made out of car parts

This dreamlike digital art captures a vibrant, kaleidoscopic bird in a lush rainforest.

human life depicted entirely out of fractals

an origami pig on fire in the middle of a dark room with a pentagram on the floor

Stable Diffusion 3

Beautiful, snowy Tokyo city is bustling. The camera moves through the bustling city street, following several people enjoying the beautiful snowy weather and shopping at nearby stalls. Gorgeous sakura petals are flying through the wind along with snowflakes.

Sora

# Why called “Diffusion” Model

为什么diffusion model被称作“Diffusion” Model

“Diffusion Model”（扩散模型）这个名称来源于其核心的生成机制，模仿了物理学中的“扩散”过程。具体来说，这个模型通过一个逐步添加噪声的正向扩散过程（Forward Diffusion Process），将数据从结构化分布逐渐“扩散”成纯噪声；然后，通过一个逆扩散过程（Reverse Diffusion Process），逐步将噪声还原为数据分布。以下是详细的解释：

# Why called “Diffusion” Model

### 1. 正向扩散过程（Forward Diffusion Process）

- 在训练过程中，扩散模型将数据（如图像）逐步加入高斯噪声，直到完全变成纯噪声。
- 这个过程类似于热扩散或粒子扩散——系统逐步趋于无序状态。
- 数学上，这通常用一个马尔科夫链建模，表示数据分布如何一步步被破坏：

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$$

其中，$\beta_t$ 控制每一步加入的噪声强度。

### 2. 逆扩散过程（Reverse Diffusion Process）

- 在生成过程中，扩散模型学会反向操作，将纯噪声逐步还原成结构化数据。
- 模型通过学习数据分布的时间条件概率 $p_\theta(x_{t-1}|x_t)$ 实现这个逆过程。
- 与物理扩散中的“逆扩散”类似，这一过程逐步减少无序性，回归原始结构化分布。

# Why called “Diffusion” Model

### 3. 为什么叫“扩散”?

这种逐步破坏和还原数据的过程，与物理意义上的扩散过程十分相似：

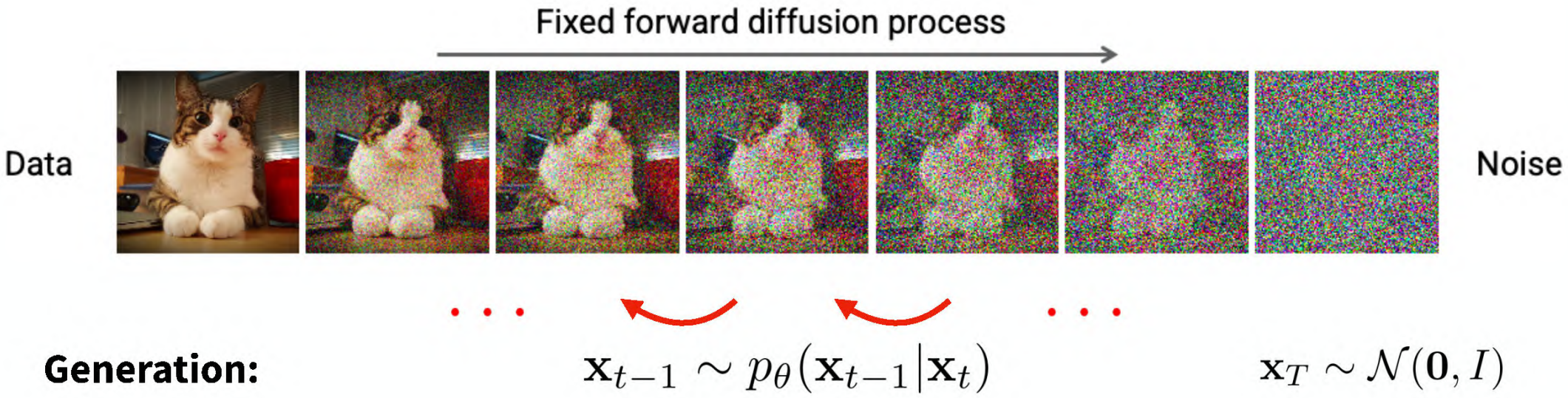
- **正向扩散**：数据逐渐丧失其特定结构，变得随机且无序（类似热力学中的熵增）。
- **逆扩散**：从无序到有序，逐步恢复结构化分布。

这个过程的名称形象地描述了模型的工作机制，因此被称为“扩散模型”。

### 补充：扩散模型的物理灵感

扩散模型与物理中的**布朗运动**和**热扩散**相关联，特别是其正向扩散的随机性和逆扩散的去噪过程。本质上，扩散模型可以看作是对概率分布的动态建模，借助扩散方程的数学性质，模拟数据分布的演化。

# Denoising Diffusion Models: A Brief Introduction

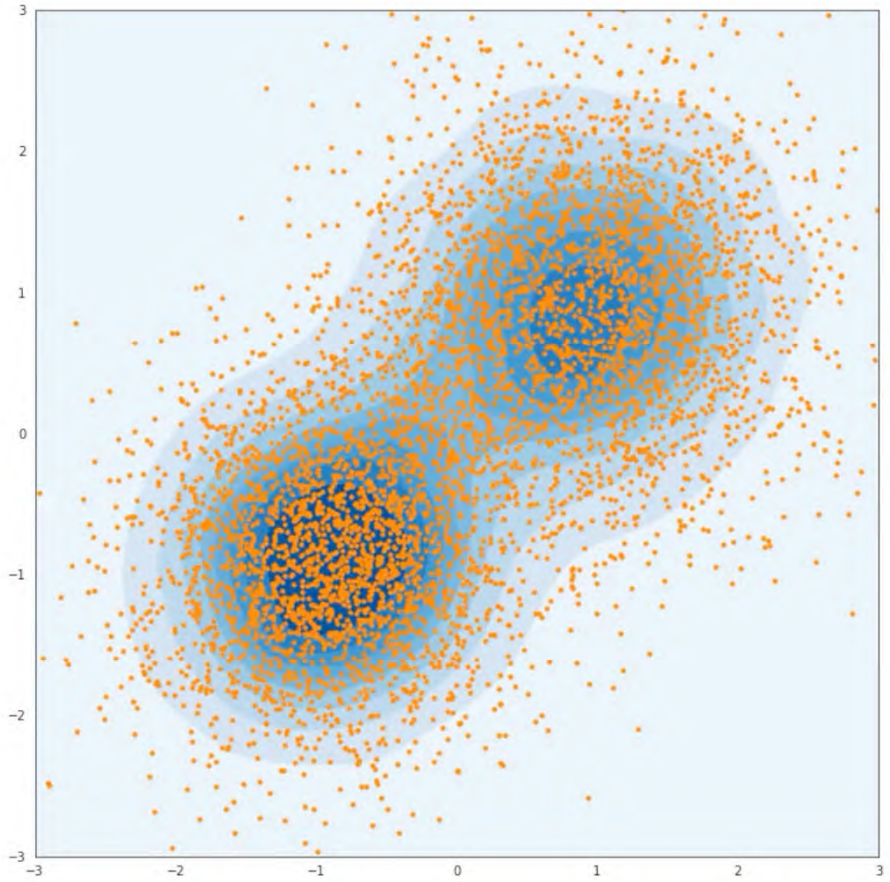



Denoising diffusion models learn to reverse a fixed forward diffusion process

Tutorial: https://cvpr2022-tutorial-diffusion-models.github.io/

# Denoising Diffusion Models: Forward Diffusion

$\mathbf{x}_0 \sim p_{\texttt{data}}$

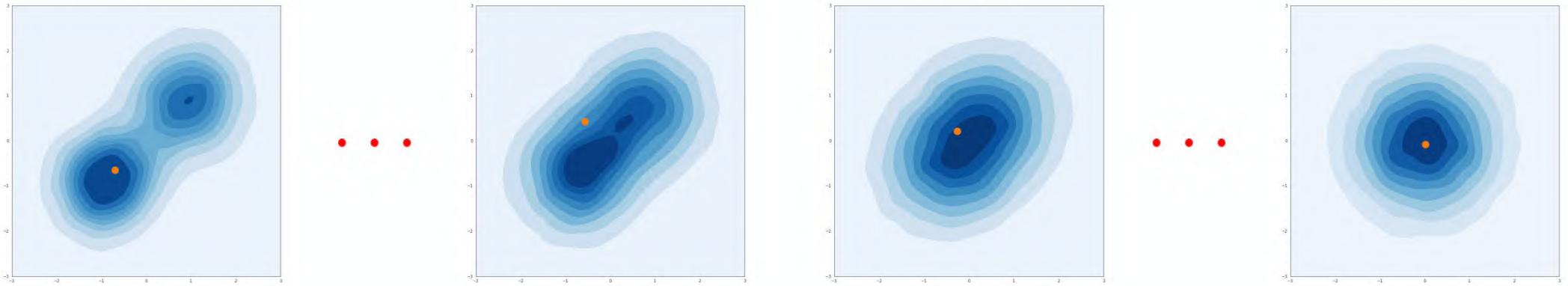



$\mathbf{x}_t \sim \mathcal{N}(\sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$

# Denoising Diffusion Models: Forward Diffusion

A schedule (defined via $\{\beta_t\}$) to add noise to data samples s.t. $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

$\mathbf{x}_0 \sim p_{\text{data}}$ $\quad$ $\mathbf{x}_t \sim \mathcal{N}(\sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$ $\quad$ $\mathbf{x}_T$



# Prove that $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

$$\mathbf{x}_t \sim \mathcal{N}(\sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I})$$

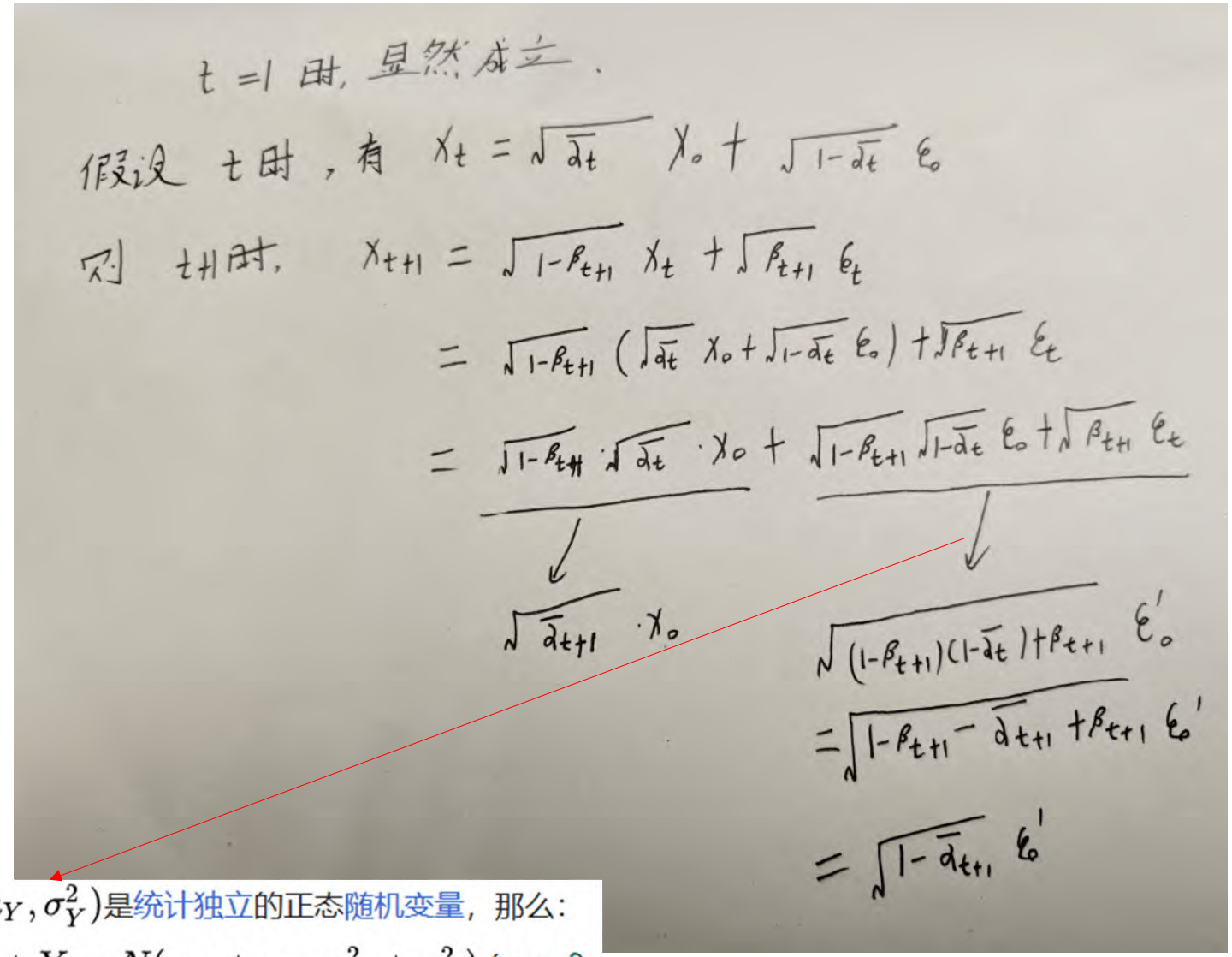
If we define $\alpha_t = 1 - \beta_t$, $\bar{\alpha}_t = \prod_{s=0}^{t} \alpha_s$ where $\boldsymbol{\epsilon}_0, ..., \epsilon_{t-2}, \epsilon_{t-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, one can use the reparameterization trick in a recursive manner to prove that:

$$\begin{aligned}
\mathbf{x}_t &= \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\boldsymbol{\epsilon}_{t-1} \\
&= \sqrt{\alpha_t}\mathbf{x}_{t-2} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{t-2} \\
&= \dots \\
&= \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_0
\end{aligned}$$

$$\mathbf{x}_t \sim \mathcal{N}(\sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I})$$

$$\begin{aligned}
\mathbf{x}_t &= \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\boldsymbol{\epsilon}_{t-1} \\
&= \sqrt{\alpha_t}\mathbf{x}_{t-2} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{t-2} \\
&= \dots \\
&= \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_0
\end{aligned}$$

$$\alpha_t = 1 - \beta_t, \quad \bar{\alpha}_t = \prod_{s=0}^{t} \alpha_s$$




如果$X \sim N(\mu_X, \sigma_X^2)$与$Y \sim N(\mu_Y, \sigma_Y^2)$是统计独立的正态随机变量，那么：

- 它们的和也满足正态分布$U = X + Y \sim N(\mu_X + \mu_Y, \sigma_X^2 + \sigma_Y^2)$ (proof).

When t -> ∞ $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

$$\mathbf{x}_t \sim \mathcal{N}(\sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I})$$

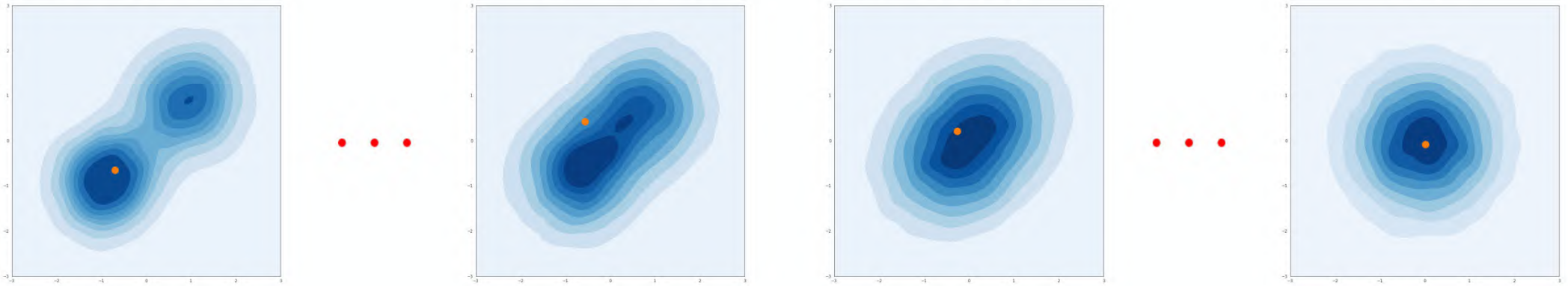
If we define $\alpha_t = 1 - \beta_t$, $\bar{\alpha}_t = \prod_{s=0}^{t} \alpha_s$ where $\boldsymbol{\epsilon}_0, ..., \epsilon_{t-2}, \epsilon_{t-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, one can use the reparameterization trick in a recursive manner to prove that:

$$\begin{aligned}
\mathbf{x}_t &= \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\boldsymbol{\epsilon}_{t-1} \\
&= \sqrt{\alpha_t}\mathbf{x}_{t-2} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{t-2} \\
&= \dots \\
&= \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_0
\end{aligned}$$

# Denoising Diffusion Models: Forward Diffusion

A schedule (defined via $\{\beta_t\}$) to add noise to data samples s.t. $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

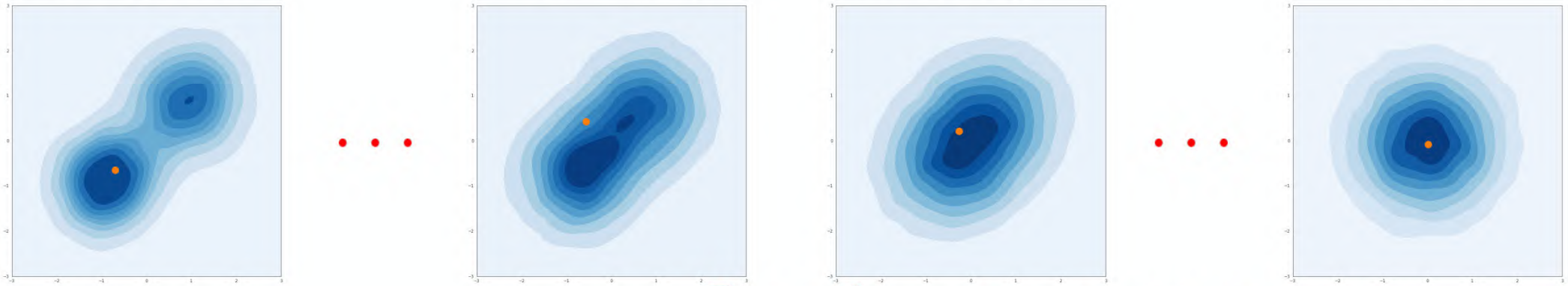
$\mathbf{x}_0 \sim p_{\text{data}}$ $\quad$ $\mathbf{x}_t \sim \mathcal{N}(\sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$ $\quad$ $\mathbf{x}_T$



# Denoising Diffusion Models: Training

A schedule (defined via $\{\beta_t\}$) to add noise to data samples s.t. $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

$\mathbf{x}_0 \sim p_{\text{data}}$ $\quad$ $\mathbf{x}_t \sim \mathcal{N}(\sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$ $\quad$ $\mathbf{x}_T$



**approximated as a gaussian**

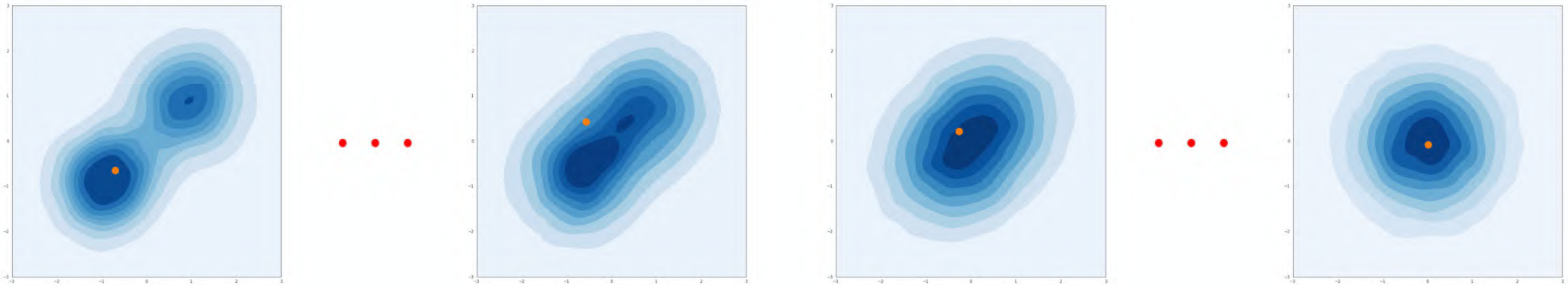
We need to learn: $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \equiv \mathcal{N}(\mu_\theta(\mathbf{x}_t, t), \sigma_t^2\mathbf{I})$

**diffusion model trained to predict the mean**

# Denoising Diffusion Models: Training

A schedule (defined via $\{\beta_t\}$) to add noise to data samples s.t. $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

$\mathbf{x}_0 \sim p_{\text{data}}$ $\quad$ $\mathbf{x}_t \sim \mathcal{N}(\sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I})$ $\quad$ $\mathbf{x}_T$



We need to learn: $\mu_\theta(\mathbf{x}_t, t)$

(in expectation under the forward denoising process, what is the mean $\mathbf{x}_{t-1}$ given $\mathbf{x}_t$)

# Denoising Diffusion Models: Training

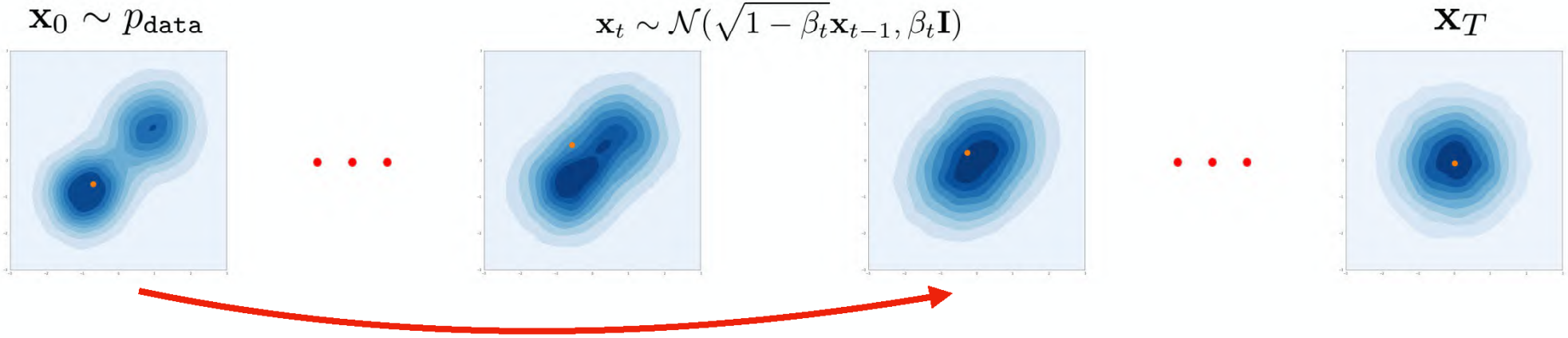
$\mathbf{x}_0 \sim p_{\text{data}}$ $\mathbf{x}_t \sim \mathcal{N}(\sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I})$ $\mathbf{x}_T$

$$\mathbb{E}_{\mathbf{x}_0 \sim \mathcal{D}}\ \mathbb{E}_t\ \mathbb{E}_{\mathbf{x}_1 \sim q(\mathbf{x}_1|\mathbf{x}_0)} \cdots \mathbb{E}_{\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x}_{t-1})} \left\| \mu_\theta(\mathbf{x}_t, t) - \mathbf{x}_{t-1} \right\|^2$$

(a very high-variance implementation due to multiple sampling steps)

Instead, we can directly sample $\mathbf{x}_t$ given $\mathbf{x}_0$!

# Denoising Diffusion Models: Training



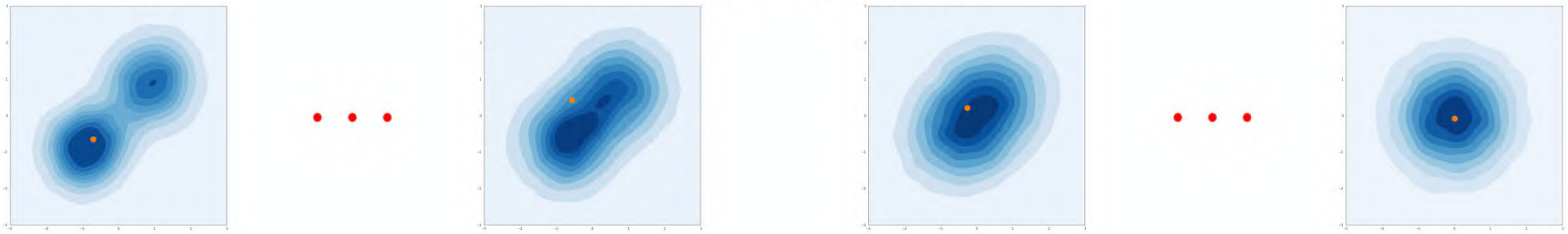

Instead, we can directly sample $\mathbf{x}_t$ given $\mathbf{x}_0$!

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

$$\mathbf{x}_t \sim \mathcal{N}(\sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}) \qquad \bar{\alpha}_t = \Pi_{t' \leq t}(1-\beta_{t'})$$

# Denoising Diffusion Models: Training

$\mathbf{x}_0 \sim p_{\text{data}}$ $\mathbf{x}_t \sim \mathcal{N}(\sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$ $\mathbf{x}_T$



$$\mathbb{E}_{\mathbf{x}_0 \sim \mathcal{D}}\ \mathbb{E}_t\ \mathbb{E}_{\epsilon \sim \mathcal{N}(\mathbf{0},\mathbf{I})} \left\| \mu_\theta(\mathbf{x}_t, t) - \mu_q(\mathbf{x}_{t-1}|\mathbf{x}_0, \mathbf{x}_t) \right\|^2$$

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

$$\mu_q(\mathbf{x}_{t-1}|\mathbf{x}_0, \mathbf{x}_t) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t$$

**(target mean can be analytically derived given the the forward diffusion process)**

# Interchangeability of Prediction Targets: $\mu_q, \epsilon, \mathbf{x}_0$

Suppose we learn a model $\mu_\theta(\mathbf{x}_t, t)$ to predict $\mu_q$ as defined earlier

Given some $(\mathbf{x}_t, t)$, say this model predicts $\hat{\mu} = \mu_\theta(\mathbf{x}_t, t)$

$$\hat{\mathbf{x}}_0 = \frac{1-\bar{\alpha}_t}{\sqrt{\bar{\alpha}_{t-1}}\beta_t}\hat{\mu} - \mathbf{x}_t\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_t)}{\sqrt{\bar{\alpha}_{t-1}}\beta_t}$$

$$\hat{\epsilon} = \frac{1}{\sqrt{(1-\bar{\alpha}_t)}}\mathbf{x}_t - \frac{\sqrt{\bar{\alpha}_t}}{\sqrt{(1-\bar{\alpha}_t)}}\hat{\mathbf{x}}_0$$

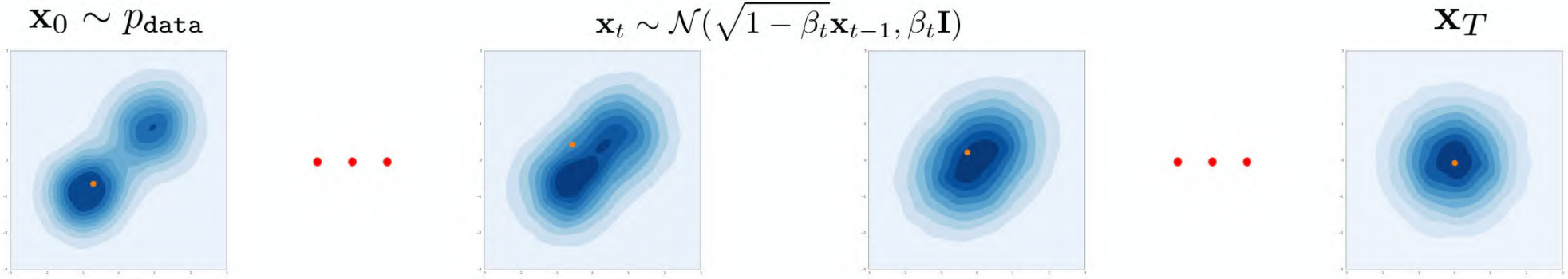
**Given $(\mathbf{x}_t, t)$, a prediction for $\mu_q$ also implies a prediction for $\epsilon$ or $\mathbf{x}_0$ (and vice-versa)**

**We can train models to predict any of these as they are inter-convertible!**

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

$$\mu_q(\mathbf{x}_{t-1}|\mathbf{x}_0, \mathbf{x}_t) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t$$

# Denoising Diffusion Models: Training




$$\mathbb{E}_{\mathbf{x}_0 \sim \mathcal{D}}\ \mathbb{E}_t\ \mathbb{E}_{\epsilon \sim \mathcal{N}(\mathbf{0},\mathbf{I})} \left\| \epsilon_\theta(\mathbf{x}_t, t) - \epsilon \right\|^2$$

$$\approx -\log p_\theta(\mathbf{x}_0 | \mathbf{c})$$

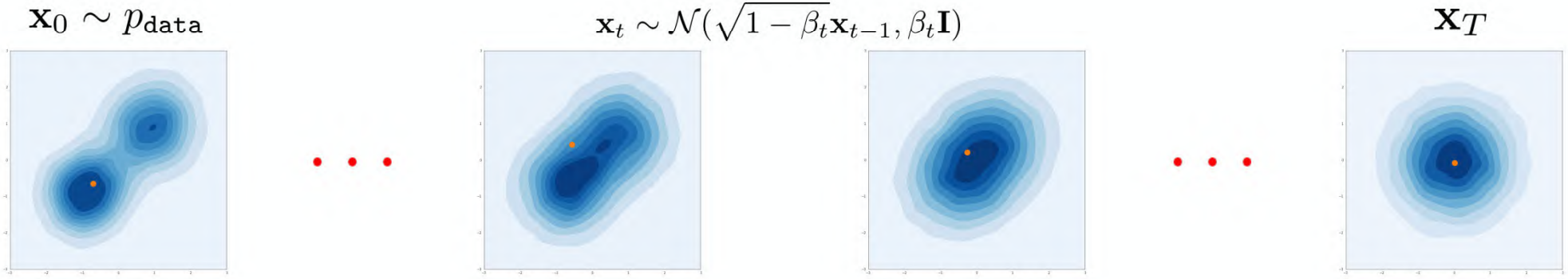
(*not exactly. Need to add time-dependent weights in above error to be a lower bound — see DDPM paper for equations)

**tl;dr — Training a diffusion model = training a network to predict the sampled noise**

**Minimizing this error can be interpreted as maximizing log-likelihood of training data**

Denoising Diffusion Probabilistic Models. *Ho et. al.*

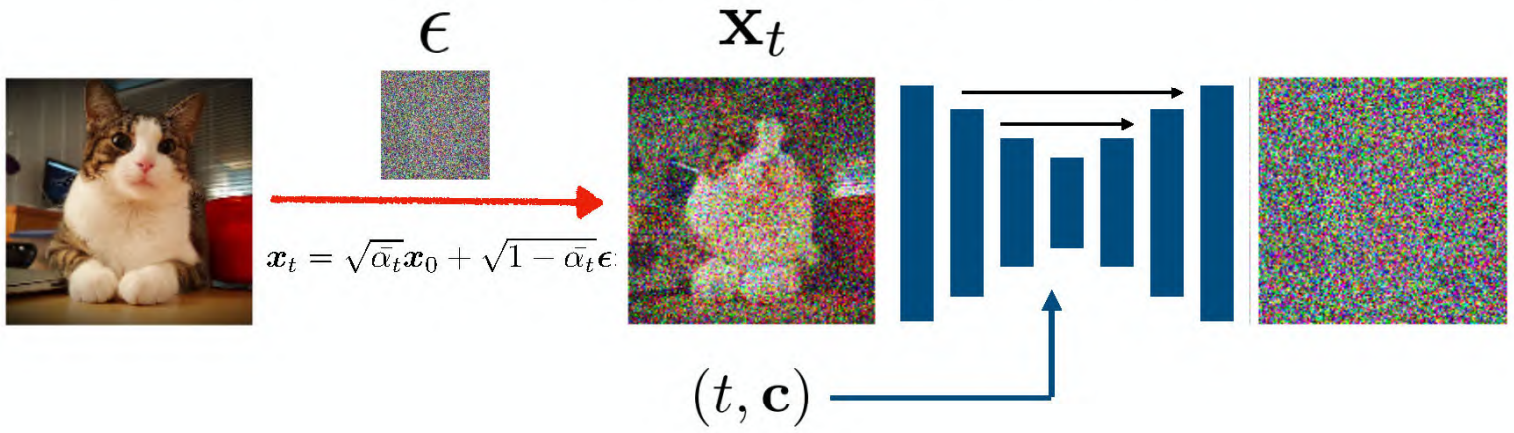
# Denoising Diffusion Models: Training




$$\mathbb{E}_{\mathbf{x}_0,\mathbf{c}\sim\mathcal{D}}\,\mathbb{E}_t\,\mathbb{E}_{\epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left\|\epsilon_\theta(\mathbf{x}_t, t, \mathbf{c}) - \epsilon\right\|^2$$

**Can easily be extended to capture conditional distributions $p_\theta(\mathbf{x} \mid \mathbf{c})$**

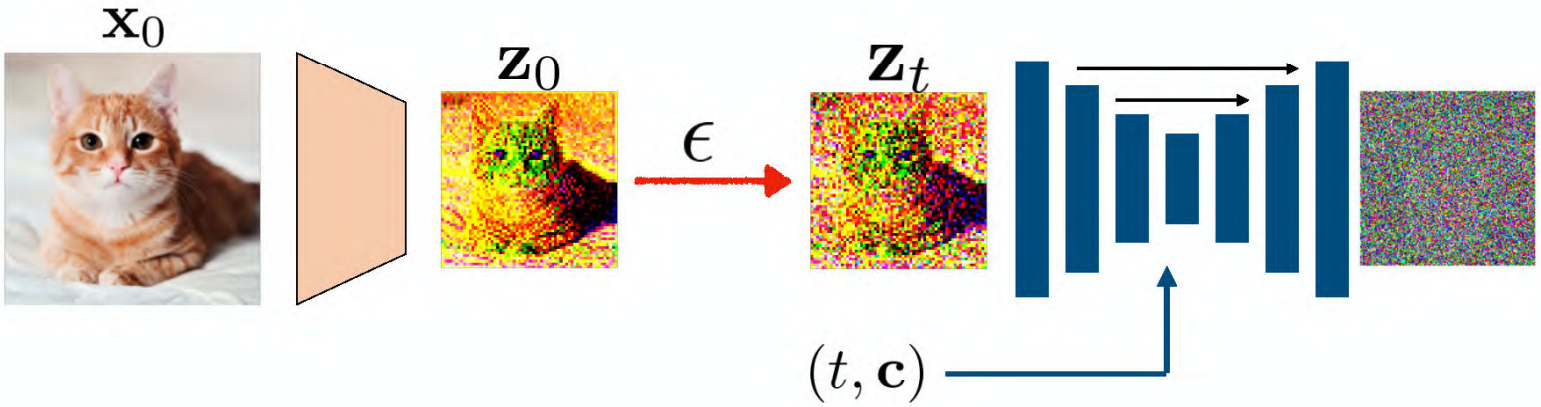
# Text-conditioned Image Diffusion




A U-Net to infer noise given noisy data ($\mathbf{x}_t$), time (t), and text ($\mathbf{c}$)

$$\mathbb{E}_{\mathbf{x}_0,\mathbf{c}\sim\mathcal{D}}\,\mathbb{E}_t\,\mathbb{E}_{\epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left\|\epsilon_\theta(\mathbf{x}_t, t, \mathbf{c}) - \epsilon\right\|^2$$

# Text-conditioned Latent Image Diffusion




Diffusion model learned in latent space (as opposed to pixel space).

Using pre-trained encoder+decoder (similar to VQ-VAE)

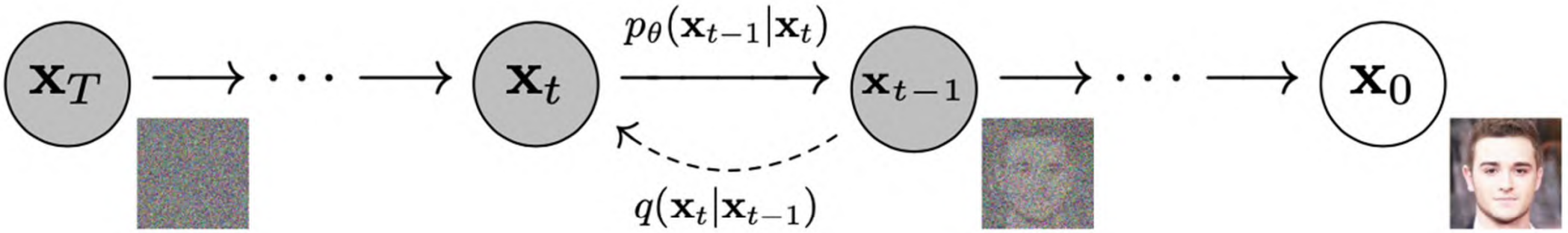
$$\mathbb{E}_{\mathbf{x}_0,\mathbf{c}\sim\mathcal{D}}\,\mathbb{E}_t\,\mathbb{E}_{\epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\|\epsilon_\theta(\mathbf{z}_t,t,\mathbf{c})-\epsilon\|^2$$

"teddy bear riding a skateboard in Times Square"

Stable Diffusion

DALL·E 2

# Algorithm




**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

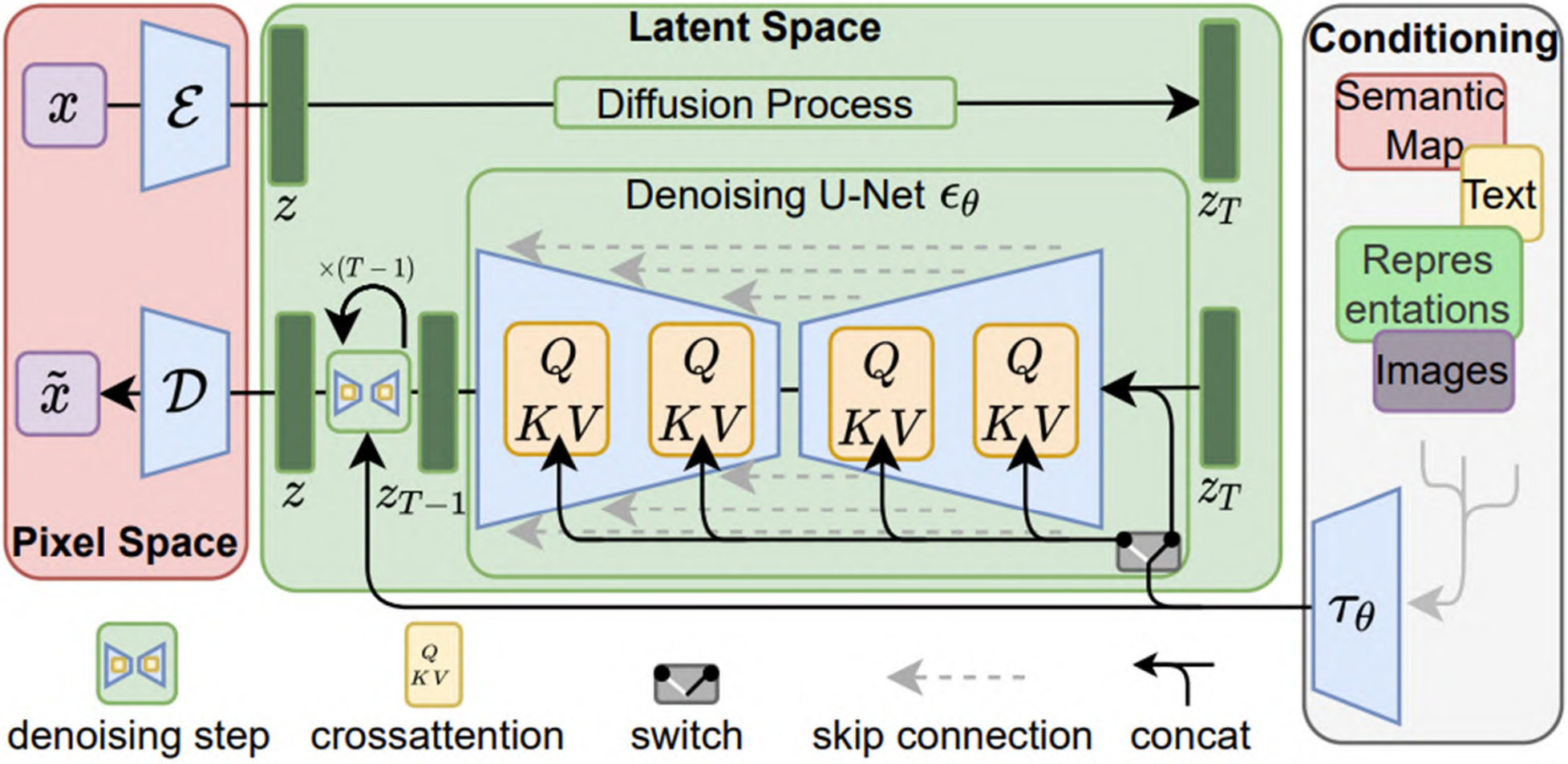
**Algorithm 2** Sampling

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** $\mathbf{x}_0$

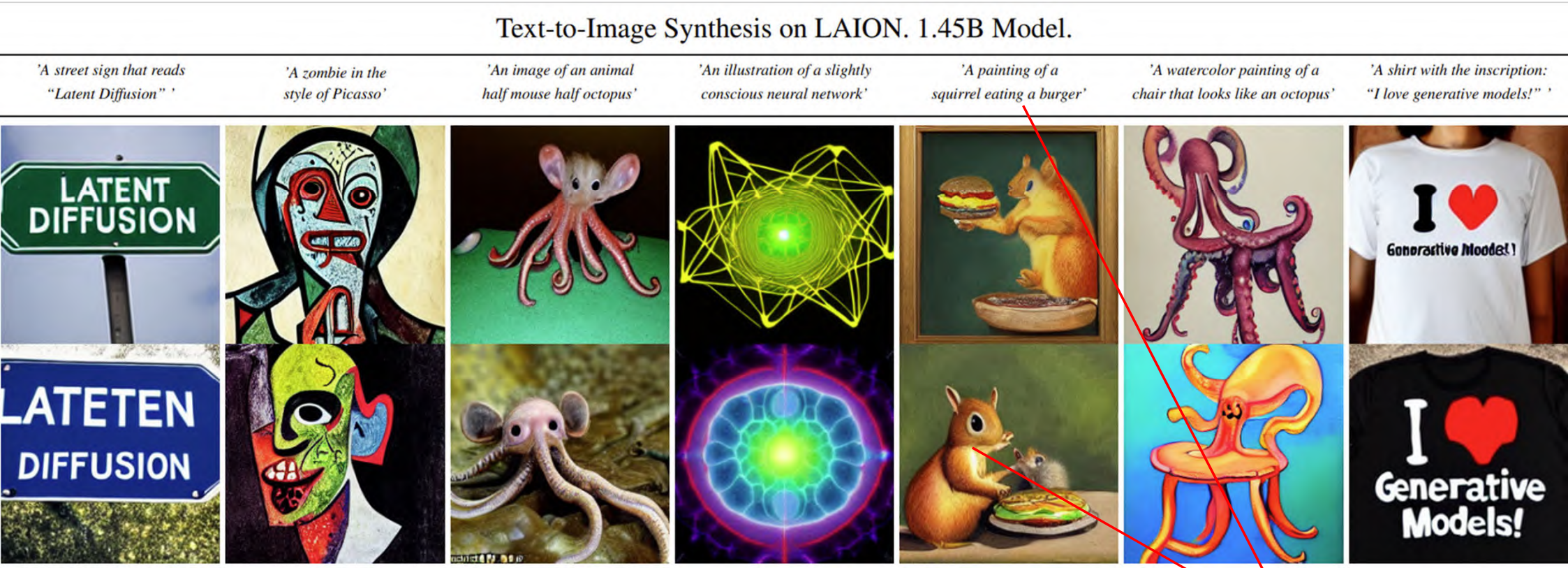
$\epsilon_\theta \approx \sigma \nabla_x \log p_\theta(x)$ (conceptually helpful; but we never explicitly represent p_theta)

[Ho, Jain, Abbeel, 2020]

# Pipeline


Latent Space
Diffusion Process
Denoising U-Net $\epsilon_\theta$
$\times(T-1)$
$x$
$\mathcal{E}$
$z$
$z_T$
$\tilde{x}$
$\mathcal{D}$
$z_{T-1}$
Q
K V
Pixel Space
Conditioning
Semantic Map
Text
Repres entations
Images
$\tau_\theta$
denoising step
crossattention
switch
skip connection
concat

# Result




The text &
The predicted (noise) match

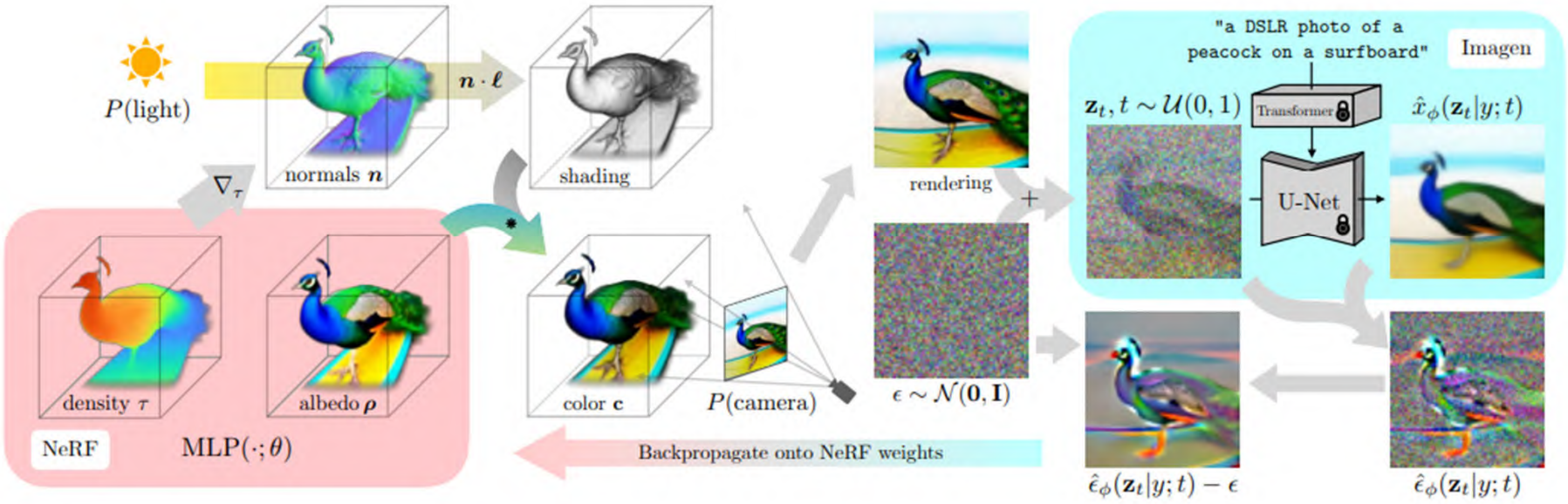
Now we have a good diffusion model that predict the corresponding noise given text.

But it only generates 2D.

How to do 3d generation from this?

# DreamFusion

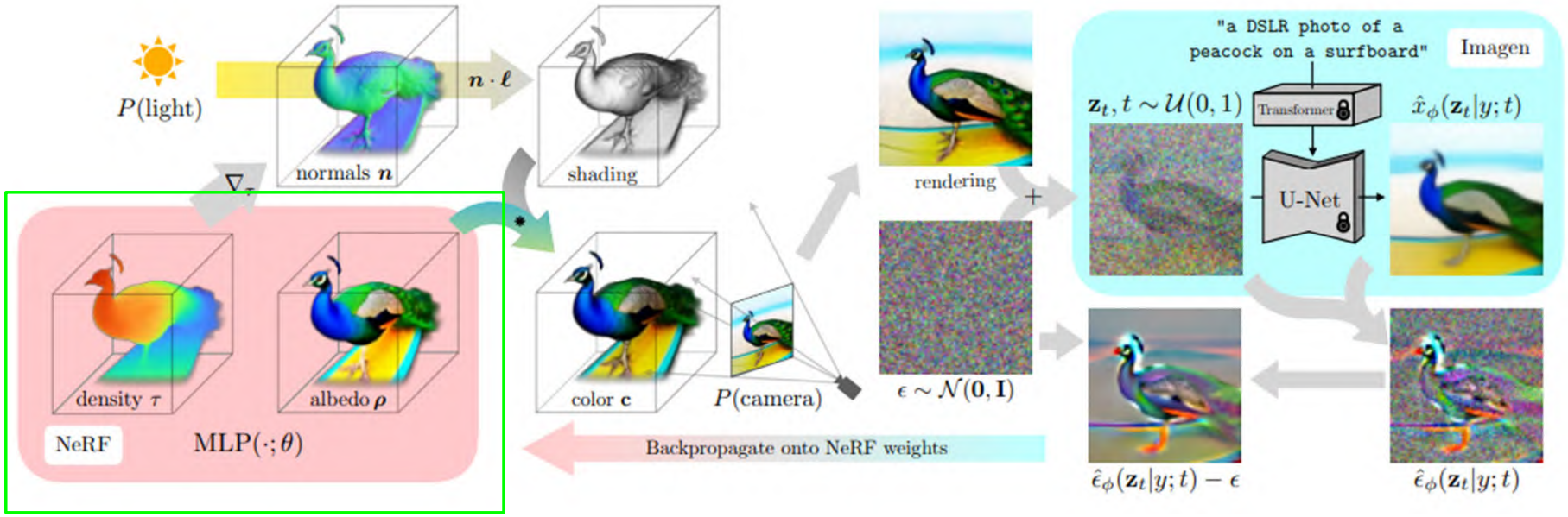
The first pipeline to generate 3D from 2D diffusion Model.




DreamFusion: Text-to-3D using 2D Diffusion. ICLR 2023.

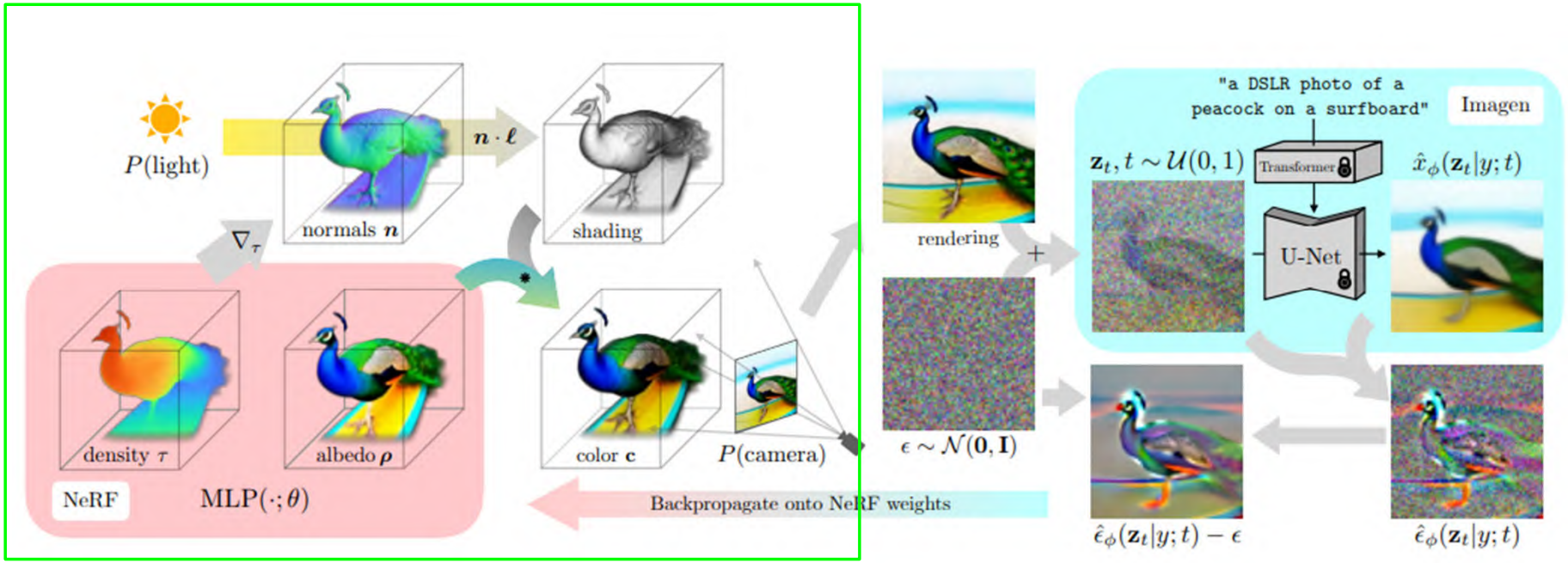
# DreamFusion

Start with a randomly initialized parameterized 3D representation (NeRF)



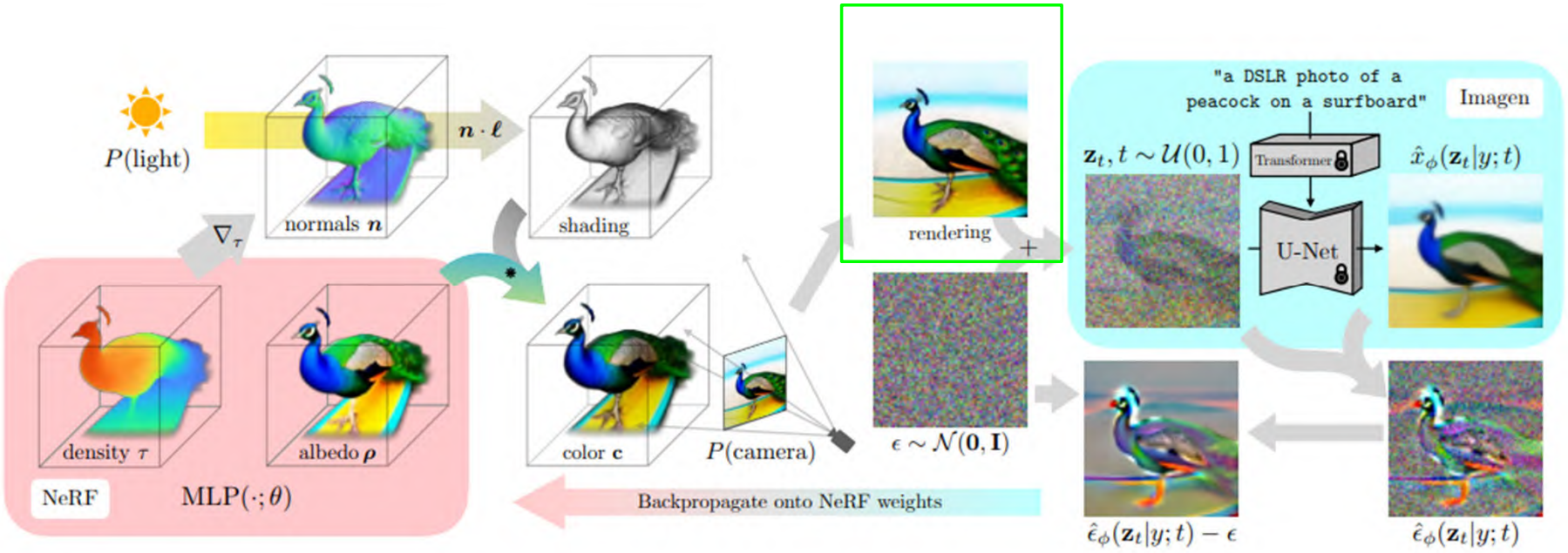
# DreamFusion

Randomly sample camera angle and lighting



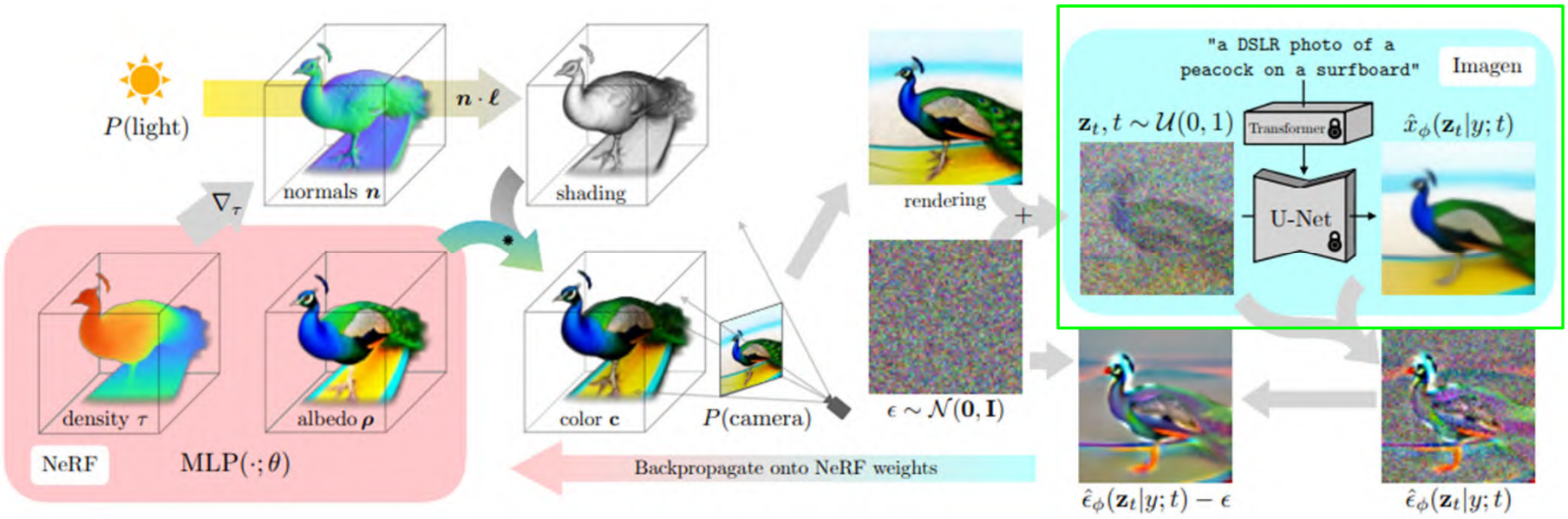
# DreamFusion

Render the image



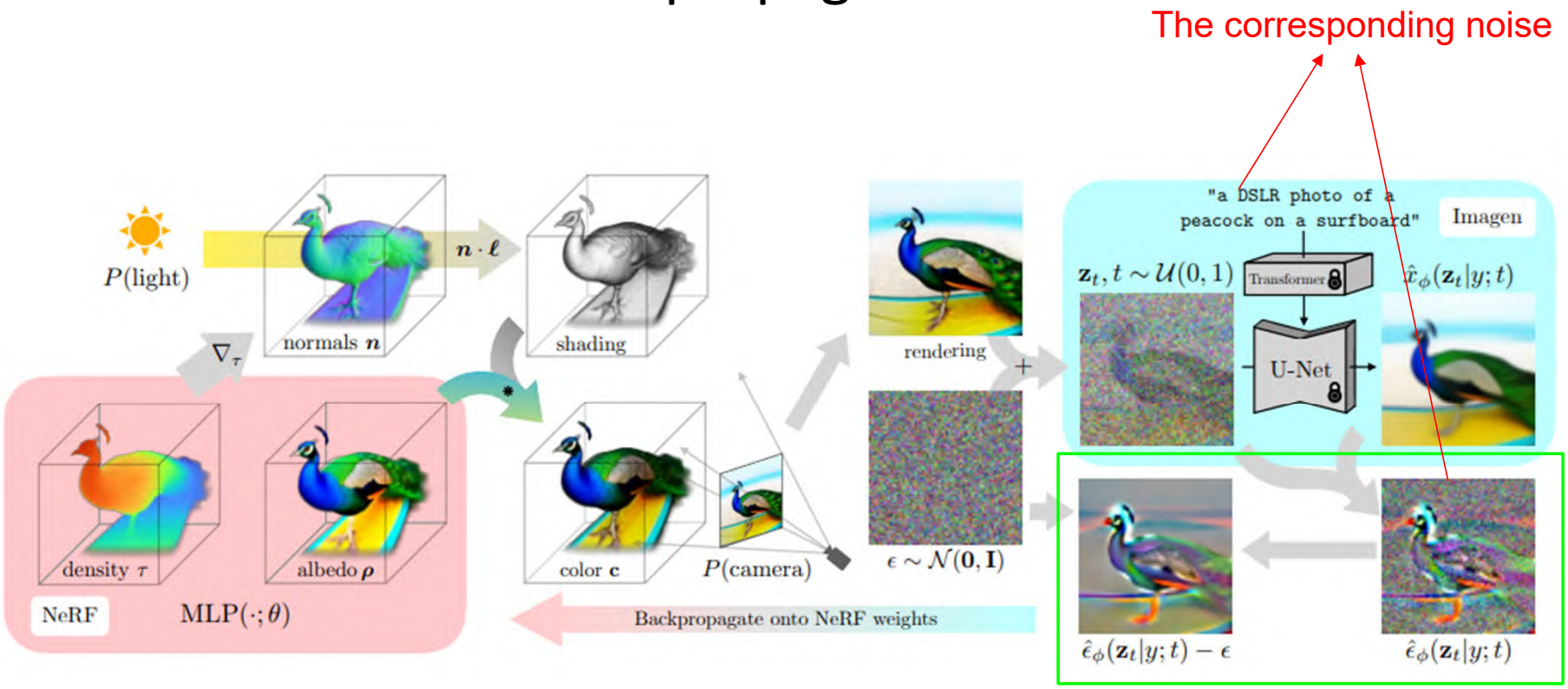
# DreamFusion

Noise the image and feed it through the diffusion model to predict the noise



# DreamFusion

Compute the SDS loss and backpropagate



# DreamFusion

Intuitively, we want the NeRF parameters to produce images that minimize the diffusion loss

$$\mathcal{L}_{\text{Diff}}(\phi, \mathbf{x}) = \mathbb{E}_{t\sim\mathcal{U}(0,1),\epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[w(t)\|\epsilon_\phi(\alpha_t\mathbf{x} + \sigma_t\epsilon; t) - \epsilon\|_2^2\right]$$

$$\nabla_\theta\mathcal{L}_{\text{Diff}}(\phi, \mathbf{x} = g(\theta)) = \mathbb{E}_{t,\epsilon}\left[w(t)\underbrace{(\hat{\epsilon}_\phi(\mathbf{z}_t; y, t) - \epsilon)}_{\text{Noise Residual}}\underbrace{\frac{\partial\hat{\epsilon}_\phi(\mathbf{z}_t; y, t)}{\mathbf{z}_t}}_{\text{U-Net Jacobian}}\underbrace{\frac{\partial\mathbf{x}}{\partial\theta}}_{\text{Generator Jacobian}}\right]$$

But the objective is generally brittle / expensive (backprop through the UNet)

# DreamFusion

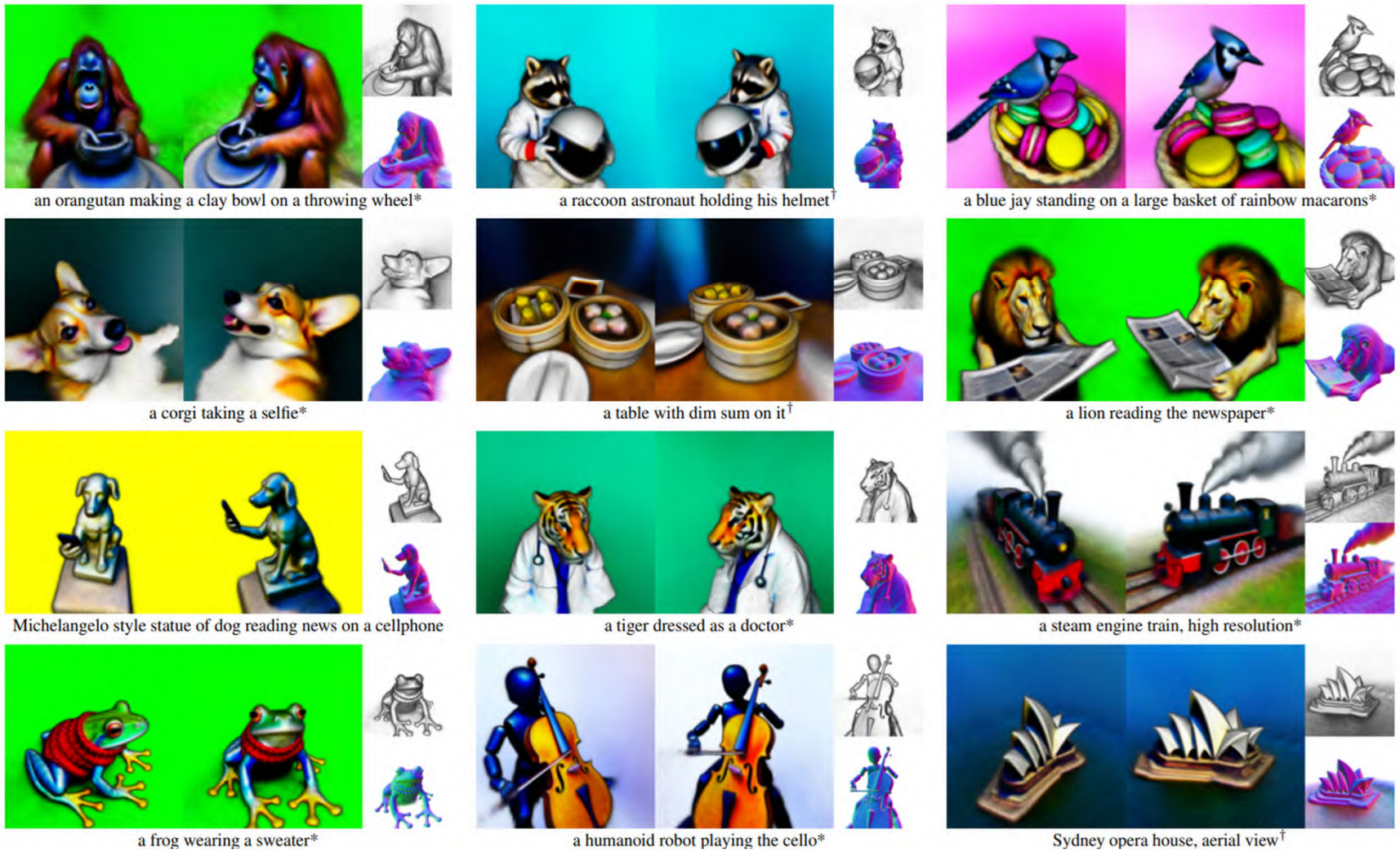
It works better to just remove the U-Net jacobian

$$\nabla_\theta \mathcal{L}_{\text{Diff}}(\phi, \mathbf{x} = g(\theta)) = \mathbb{E}_{t,\epsilon}\left[ w(t) \underbrace{(\hat{\epsilon}_\phi(\mathbf{z}_t; y, t) - \epsilon)}_{\text{Noise Residual}} \underbrace{\frac{\partial \hat{\epsilon}_\phi(\mathbf{z}_t; y, t)}{\mathbf{z}_t}}_{\text{U-Net Jacobian}} \underbrace{\frac{\partial \mathbf{x}}{\partial \theta}}_{\text{Generator Jacobian}} \right]$$

$$\nabla_\theta \mathcal{L}_{\text{SDS}}(\phi, \mathbf{x} = g(\theta)) \triangleq \mathbb{E}_{t,\epsilon}\left[ w(t) (\hat{\epsilon}_\phi(\mathbf{z}_t; y, t) - \epsilon) \frac{\partial \mathbf{x}}{\partial \theta} \right]$$

There is a theoretical motivation (see Appendix in paper)

# DreamFusion



# DreamFusion

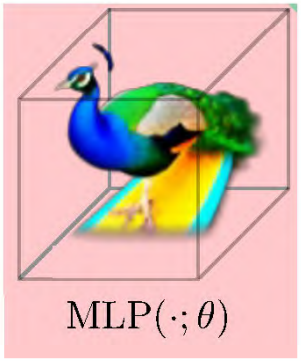
## Generate 3D from text yourself!

a DSLR photo of a squirrel | an intricate wooden carving of a squirrel | a highly detailed metal sculpture of a squirrel

[...] | wearing a kimono | wearing a medieval suit of armor | wearing a purple hoodie | wearing an elegant ballgown

[...] | reading a book | riding a motorcycle | playing the saxophone | chopping vegetables | sitting at a pottery wheel shaping a clay bowl | riding a skateboard | wielding a katana | eating a hamburger | dancing

# Text-to-3D Using Text-to-Image Diffusion



$\text{MLP}(\cdot;\theta)$

$\pi \sim \Pi$

rendering

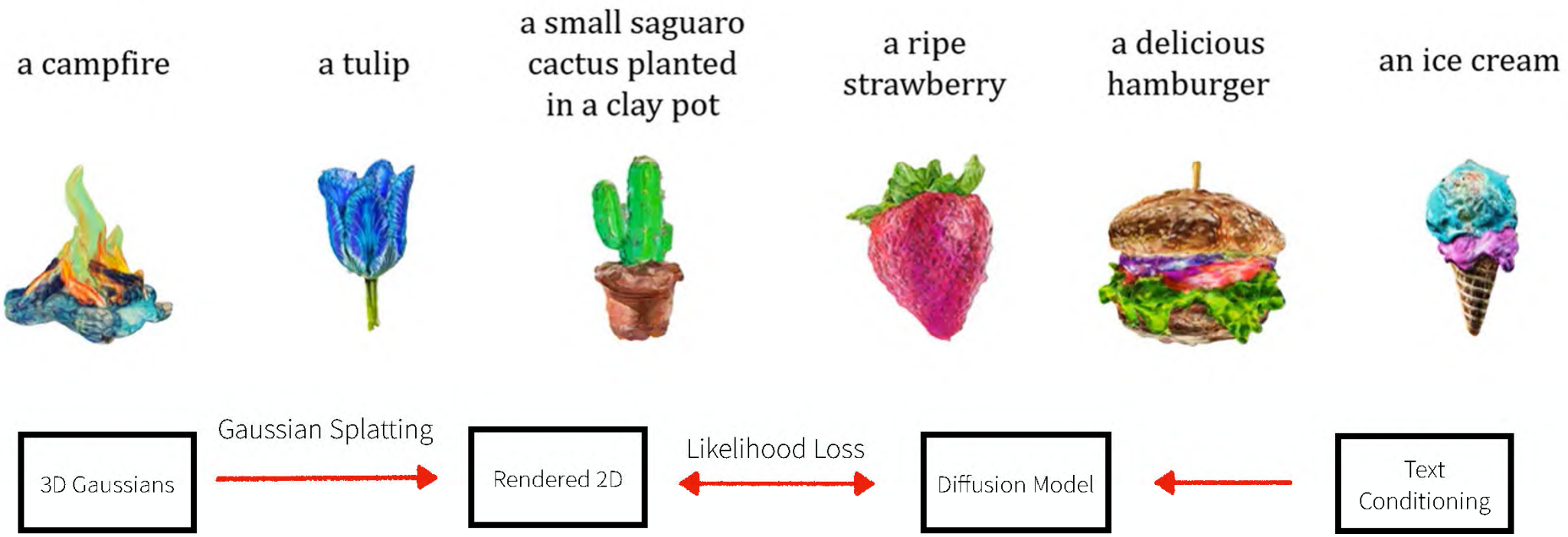
Stable Diffusion

DALL·E 2

"A high quality image of a peacock"

A general framework (not restricted to neural fields)

# Text-to-3D Using Text-to-Image Diffusion




(e.g. ~10x faster optimization using gaussian splatting)

DreamGaussian: Generative Gaussian Splatting for Efficient 3D Content Creation. *Tang et. al.*

*[Submitted on 11 Nov 2024]*

# Edify 3D: Scalable High-Quality 3D Asset Generation

NVIDIA: Maciej Bala, Yin Cui, Yifan Ding, Yunhao Ge, Zekun Hao, Jon Hasselgren, Jacob Munkberg, Stella Shi, Fangyin Wei, Donglai Xiang, Jiashu Xu, Xiaohui Zeng, Qinsheng Z

We introduce Edify 3D, an advanced solution designed for high-quality 3D asset generation. Our meth then used to reconstruct the shape, texture, and PBR materials of the object. Our method can genera

Comments: Project website: this https URL
Subjects: **Computer Vision and Pattern Recognition (cs.CV)**; Artificial Intelligence (cs.AI); Graphics (cs.GR)
Cite as: arXiv:2411.07135 **[cs.CV]**
(or arXiv:2411.07135v1 **[cs.CV]** for this version)
https://doi.org/10.48550/arXiv.2411.07135

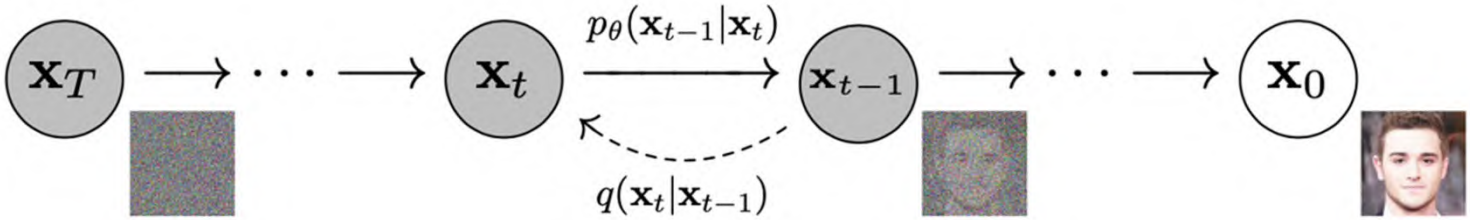
## Submission history

From: Chen-Hsuan Lin [view email]
**[v1]** Mon, 11 Nov 2024 17:07:43 UTC (19,223 KB)

One, two, three

NVIDIA

# Course Summary




Diffusion Models: Predict (corresponding) noise from (text)


Generate 3D from Diffusion: Make the noise corresponding to text by rendering the right object


中国科学技术大学

University of Science and Technology of China

# 谢谢观看!