Image: Synthetic ImageNet samples the diffusion model by Dhariwal and Nichol

COMP547

DEEP UNSUPERVISED LEARNING

Lecture #9 – Denoising Diffusion Models

KOÇ UNIVERSITY

Aykut Erdem // Koç University // Spring 2025

Previously on COMP547

- Motivation
- Original GAN (Goodfellow et al, 2014)
- Evaluation: Parzen, Inception, Fréchet
- Theory of GANs
- GAN Progression
- Conditional GANs, Cycle-Consistent Adversarial Networks
- Applications

Lecture overview

- Motivation
- Energy-based models
- Score-based Models
- Denoising Diffusion Models
- Deeper Dive into Diffusion Models

Disclaimer: Much of the material and slides for this lecture were borrowed from
—Stefano Ermon and Aditya Grover's Stanford CS236 class
—Yang Song and Stefano Ermon's talk titled "Generative Modeling by Estimating Gradients of the Data Distribution"
—Jascha Sohl-Dickstein's talk "Deep Unsupervised Learning using Nonequilibrium Thermodynamics"
—Sangwoo Mo's talk titled "Introduction to Diffusion Models"
—Pieter Abbeel, Wilson Yan, Kevin Frans, Philipp Wu's Berkeley CS294-158 class

Lecture overview

- Motivation
- Energy-based models
- Score-based Models
- Denoising Diffusion Models
- Deeper Dive into Diffusion Mo

Inspired by their work, we further investigated the relationship between diffusion models and score-based generative models in an ICLR 2021 paper [20]. We found that the sampling method of diffusion probabilistic models can be integrated with annealed Langevin dynamics of score-based models to create a unified and more powerful sampler (the Predictor-Corrector sampler). By generalizing the number of noise scales to infinity, we further proved that score-based generative models and diffusion probabilistic models can both be viewed as discretizations to stochastic differential equations determined by score functions. This work bridges both score-based generative modeling and diffusion probabilistic modeling into a unified framework.

Collectively, these latest developments seem to indicate that both score-based generative modeling with multiple noise perturbations and diffusion probabilistic models are different perspectives of the same model family, much like how wave mechanics and matrix mechanics are equivalent formulations of quantum mechanics in the history of physics [5]. The perspective of score matching and score-based models allows one to calculate log-likelihoods exactly, solve inverse problems naturally, and is directly connected to energy-based models, Schrödinger bridges and optimal transport [47]. The perspective of diffusion models is naturally connected to VAEs, lossy compression, and can be directly incorporated with variational probabilistic inference. This blog post focuses on the first perspective, but I highly recommend interested readers to learn about the alternative perspective of diffusion models as well (see a great blog by Lilian Weng).

https://yang-song.github.io/blog/2021/score/

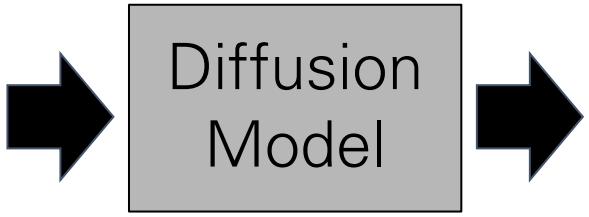
Disclaimer: Much of the material and slides fo
—Stefano Ermon and Aditya Grover's Stanford CS236
—Yang Song and Stefano Ermon's talk titled "Generati
—Jascha Sohl-Dickstein's talk "Deep Unsupervised Le
—Sangwoo Mo's talk titled "Introduction to Diffusion Models"
—Pieter Abbeel, Wilson Yan, Kevin Frans, Philipp Wu's Berkeley CS294-158 class

Lecture overview

- **Motivation**
- Energy-based models
- Score-based Models
- Denoising Diffusion Models
- Deeper Dive into Diffusion Models

Motivation: Diffusion Models

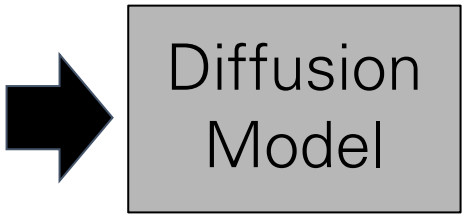
a masterful oil painting a persian exotic cat discovering their astounding crypto losses while checking their phone



OpenAI Dall-E2

Motivation: Diffusion Models

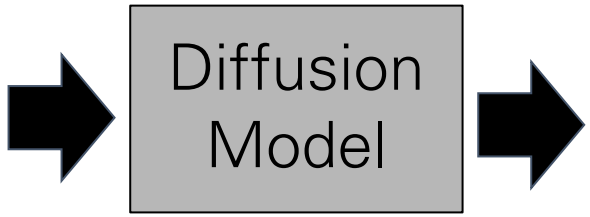
Beautiful, snowy Tokyo city is bustling. The camera moves through the bustling city street, following several people enjoying the beautiful snowy weather and shopping at nearby stalls. Gorgeous sakura petals are flying through the wind along with snowflakes.



OpenAI Sora, Bill Peeves* Tim Brooks*, et al, 2024

Motivation: Diffusion Models

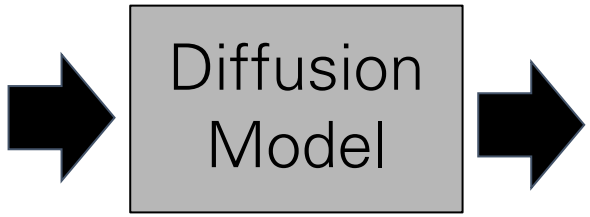
Reflections in the window of a train traveling through the Tokyo suburbs.

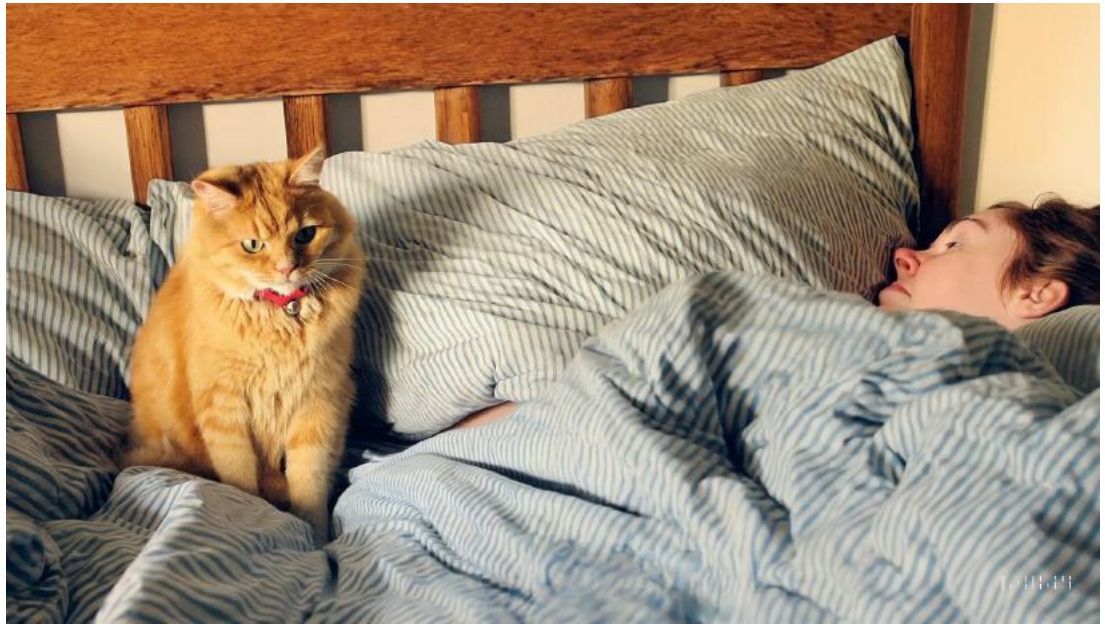


OpenAI Sora, Bill Peeves* Tim Brooks*, et al, 2024

Motivation: Diffusion Models

A cat waking up its sleeping owner demanding breakfast. The owner tries to ignore the cat, but the cat tries new tactics and finally the owner pulls out a secret stash of treats from under the pillow to hold the cat off a little longer.

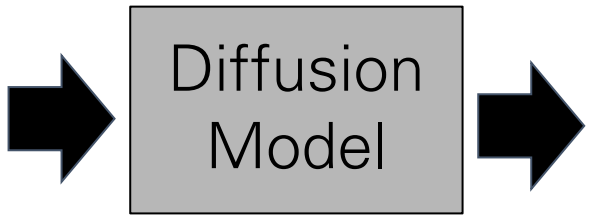




OpenAI Sora, Bill Peeves* Tim Brooks*, et al, 2024

Motivation: Diffusion Models

Five gray wolf pups frolicking and chasing each other around a remote gravel road, surrounded by grass. The pups run and leap, chasing each other, and nipping at each other, playing. More Prompt: Five gray wolf pups frolicking and chasing each other around a remote gravel road, surrounded by grass. The pups run and leap, chasing each other, and nipping at each other, playing.



OpenAI Sora, Bill Peeves* Tim Brooks*, et al, 2024

Lecture overview

- Motivation
- **Energy-based models**
  - **Parametrizing probability distributions**
  - Energy-based generative modeling
  - Ising Model, Product of Experts, Restricted Boltzmann machine, Deep Boltzman Machines
  - Training and sampling from EBMs
- Score-based Models
- Denoising Diffusion Models
- Deeper Dive into Diffusion Models

Parameterizing probability distributions

- Probability distributions $p(x)$ are a key building block in generative modeling.
  1. Non-negative: $p(x) \geq 0$
  2. Sum-to-one: $\sum_x p(x) = 1$ (or $\int p(x)dx = 1$ for continuous variables)

- Coming up with a non-negative function $p_\theta(\mathbf{x})$ is not hard. Given any function $f_\theta(\mathbf{x})$, we can choose
  - $g_\theta(\mathbf{x}) = f_\theta(\mathbf{x})^2$
  - $g_\theta(\mathbf{x}) = \exp(f_\theta(\mathbf{x}))$
  - $g_\theta(\mathbf{x}) = |f_\theta(\mathbf{x})|$
  - $g_\theta(\mathbf{x}) = \log(1 + \exp(f_\theta(\mathbf{x})))$

Parameterizing probability distributions

- Probability distributions $p(x)$ are a key building block in generative modeling.
  1. Non-negative: $p(x) \geq 0$
  2. Sum-to-one: $\sum_x p(x) = 1$ (or $\int p(x)dx = 1$ for continuous variables)

     Sum-to-one is key!
- Total "volume" is fixed: increasing $p(x_{train})$ guarantees that x_{train} becomes relatively more likely (compared to the rest)
- Problem:
  - $g_\theta(\mathbf{x}) \geq 0$ is easy, but $g_\theta(\mathbf{x})$ might not sum-to-one.
  - $\sum_x g_\theta(\mathbf{x}) = Z(\theta) \neq 1$ in general, so $g_\theta(\mathbf{x})$ is not a valid probability mass function or density

Parameterizing probability distributions

- **Problem:** $g_\theta(\mathbf{x}) \geq 0$ is easy, but $g_\theta(\mathbf{x})$ might not be normalized
- **Solution:** $p_\theta(\mathbf{x}) = \frac{1}{\text{Volume}(g_\theta)} g_\theta(\mathbf{x}) = \frac{1}{\int g_\theta(\mathbf{x})\mathrm{d}\mathbf{x}} g_\theta(\mathbf{x})$

  Then by definition $\int p_\theta(\mathbf{x})\mathrm{d}\mathbf{x} = 1$
- **Example:** Choose $g_\theta(\mathbf{x})$ so that the volume is **analytically** as a function of θ.
  1. $g_{(\mu,\sigma)}(x) = e^{-\frac{(x-\mu)^2}{2\sigma^2}}$. **Volume** is: $\int e^{-\frac{x-\mu}{2\sigma^2}} dx = \sqrt{2\pi\sigma^2}$ → **Gaussian**
  2. $g_\lambda(x) = e^{-\lambda x}$. **Volume** is: $\int_0^{+\infty} e^{-\lambda x} dx = \frac{1}{\lambda}$ → **Exponential**
  3. $g_\theta(x) = h(x)\exp\{\theta \cdot T(x)\}$. **Volume** is $\exp\{A(\theta)\}$, where $A(\theta) = \log \int h(x)\exp\{\theta \cdot T(x)\}\mathrm{d}\mathbf{x}$ → **Exponential family**
     - Normal, Poisson, exponential, Bernoulli
     - beta, gamma, Dirichlet, Wishart, etc.
- Function forms $g_\theta(\mathbf{x})$ need to allow **analytical** integration. Despite being restrictive, they are very useful as building blocks for more complex distributions.

Lecture overview

- Motivation
- **Energy-based models**
  - Parametrizing probability distributions
  - **Energy-based generative modeling**
  - Ising Model, Product of Experts, Restricted Boltzmann machine, Deep Boltzman Machines
  - Training and sampling from EBMs
- Score-based Models
- Denoising Diffusion Models
- Deeper Dive into Diffusion Models

Likelihood based learning

- **Problem:** $g_\theta(\mathbf{x}) \geq 0$ is easy, but $g_\theta(\mathbf{x})$ might not be normalized
- **Solution:** $p_\theta(\mathbf{x}) = \frac{1}{\text{Volume}(g_\theta)} g_\theta(\mathbf{x}) = \frac{1}{\int g_\theta(\mathbf{x}) \mathrm{d}\mathbf{x}} g_\theta(\mathbf{x})$

  Typically, choose $g_\theta(\mathbf{x})$ so that we know the volume analytically. More complex models can be obtained by combining these building blocks.

  1. **Autoregressive:** Products of normalized objects $p_\theta(\mathbf{x}) p_{\theta'(\mathbf{x})}(\mathbf{y})$

     $$\int_{\mathbf{x}} \int_{\mathbf{y}} p_\theta(\mathbf{x}) p_{\theta'(\mathbf{x})}(\mathbf{y}) \mathrm{d}\mathbf{x} \mathrm{d}\mathbf{y} = \int_{\mathbf{x}} p_\theta(\mathbf{x}) \underbrace{\int_{\mathbf{y}} p_{\theta'(\mathbf{x})}(\mathbf{y}) \mathrm{d}\mathbf{y}}_{=1} \mathrm{d}\mathbf{x} = \int_{\mathbf{x}} p_\theta(\mathbf{x}) \mathrm{d}\mathbf{x} = 1$$

  2. **Latent variables:** Mixtures of normalized objects $\alpha p_\theta(\mathbf{x}) + (1-\alpha) p_{\theta'}(\mathbf{x})$

     $$\int_{\mathbf{x}} \alpha p_\theta(\mathbf{x}) + (1-\alpha) p_{\theta'}(\mathbf{x}) \mathrm{d}\mathbf{x} = \alpha + (1-\alpha) = 1$$

Likelihood based learning

- **Problem:** $g_\theta(\mathbf{x}) \geq 0$ is easy, but $g_\theta(\mathbf{x})$ might not be normalized
- **Solution:** $p_\theta(\mathbf{x}) = \frac{1}{\text{Volume}(g_\theta)} g_\theta(\mathbf{x}) = \frac{1}{\int g_\theta(\mathbf{x}) \mathrm{d}\mathbf{x}} g_\theta(\mathbf{x})$

  Typically, choose $g_\theta(\mathbf{x})$ so that we know the volume analytically. More complex models can be obtained by combining these building blocks.

  1. **Autoregressive:** Products of normalized objects $p_\theta(\mathbf{x}) p_{\theta'(\mathbf{x})}(\mathbf{y})$

     $$\int_{\mathbf{x}} \int_{\mathbf{y}} p_\theta(\mathbf{x}) p_{\theta'(\mathbf{x})}(\mathbf{y}) \mathrm{d}\mathbf{x} \mathrm{d}\mathbf{y} = \int_{\mathbf{x}} p_\theta(\mathbf{x}) \underbrace{\int_{\mathbf{y}} p_{\theta'(\mathbf{x})}(\mathbf{y}) \mathrm{d}\mathbf{y}}_{=1} \mathrm{d}\mathbf{x} = \int_{\mathbf{x}} p_\theta(\mathbf{x}) \mathrm{d}\mathbf{x} = 1$$

  2. **Latent variables:** Mixtures of normalized objects $\alpha p_\theta(\mathbf{x}) + (1-\alpha) p_{\theta'}(\mathbf{x})$

     $$\int_{\mathbf{x}} \alpha p_\theta(\mathbf{x}) + (1-\alpha) p_{\theta'}(\mathbf{x}) \mathrm{d}\mathbf{x} = \alpha + (1-\alpha) = 1$$

How about using models where the "volume"/normalization constant of $g_\theta(\mathbf{x})$ is not easy to compute analytically?

Energy-based model

$$p_\theta(\mathbf{x}) = \frac{1}{\int \exp(f_\theta(\mathbf{x}))d\mathbf{x}} \exp\left(f_\theta(\mathbf{x})\right) = \frac{1}{Z(\theta)} \exp\left(f_\theta(\mathbf{x})\right)$$

The output of f_θ is a scalar value between $-\infty$ and ∞.

- The volume/normalization constant

$$Z(\theta) = \int \exp\left(f_\theta(\mathbf{x})\right) d\mathbf{x}$$

is also called the **partition** function. Why exponential (and not e.g. $f_\theta(\mathbf{x})^2$)?

1. Want to capture very large variations in probability. log-probability is the natural scale we want to work with. Otherwise need highly non-smooth f_θ.
2. Exponential families. Many common distributions can be written in this form.
3. These distributions arise under fairly general assumptions in statistical physics (maximum entropy, second law of thermodynamics).
    - $-f_\theta(\mathbf{x})$ is called the **energy**, hence the name.
    - Intuitively, configurations x with low energy (high $f_\theta(x)$) are more likely.

Energy-based model

$$p_\theta(\mathbf{x}) = \frac{1}{\int \exp(f_\theta(\mathbf{x}))d\mathbf{x}} \exp\left(f_\theta(\mathbf{x})\right) = \frac{1}{Z(\theta)} \exp\left(f_\theta(\mathbf{x})\right)$$

- Pros:
  - extreme flexibility: can use pretty much any function $f_\theta(\mathbf{x})$ you want
- Cons:
  - Sampling from $p_\theta(\mathbf{x})$ is hard
  - Evaluating and optimizing likelihood $p_\theta(\mathbf{x})$ is hard (learning is hard)
  - No feature learning (but can add latent variables)
- **Curse of dimensionality:** The fundamental issue is that computing $Z(\theta)$ numerically (when no analytic solution is available) scales exponentially in the number of dimensions of x.
- Nevertheless, **some tasks do not require knowing** $Z(\theta)$

Applications of Energy-based models

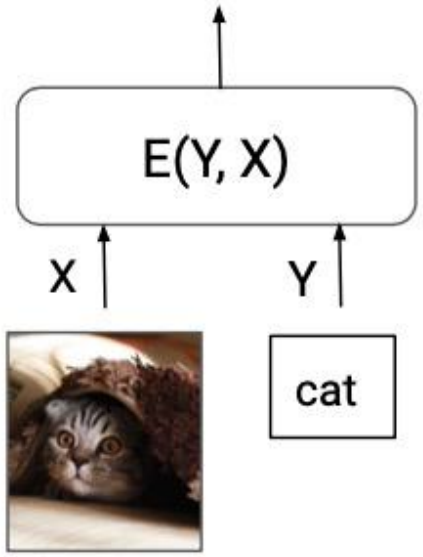
$$p_\theta(\mathbf{x}) = \frac{1}{\int \exp(f_\theta(\mathbf{x}))d\mathbf{x}} \exp\left(f_\theta(\mathbf{x})\right) = \frac{1}{Z(\theta)} \exp\left(f_\theta(\mathbf{x})\right)$$

- Given x, x′ evaluating $p_\theta(\mathbf{x})$ or $p_\theta\left(\mathbf{x}'\right)$ requires $Z(\theta)$.
- However, their ratio

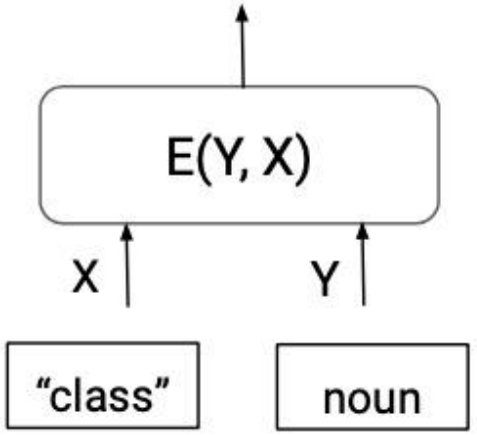
$$\frac{p_\theta(\mathbf{x})}{p_\theta(\mathbf{x}')} = \exp\left(f_\theta(\mathbf{x}) - f_\theta\left(\mathbf{x}'\right)\right)$$

  does not involve $Z(\theta)$.
- This means we can easily check which one is more likely. Applications:
  - Anomaly detection
  - Denoising

Applications of Energy-based models



object recognition



sequence labeling

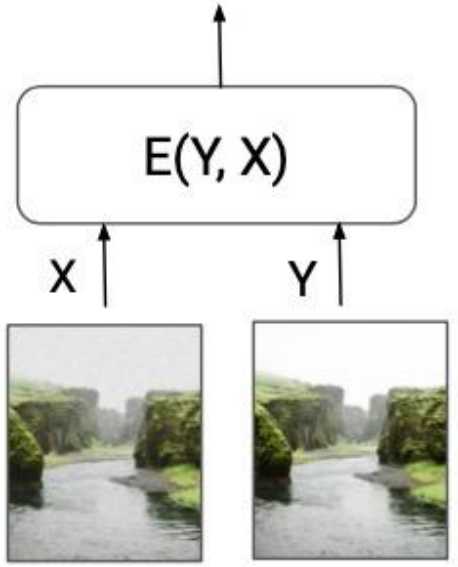


image restoration

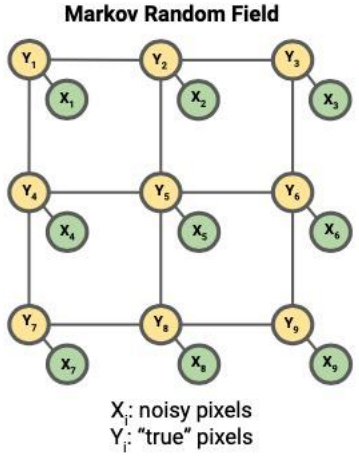
- Given a trained model, many applications require relative comparisons. Hence $Z(\theta)$ is not needed.

Lecture overview

- Motivation
- **Energy-based models**
  - Parametrizing probability distributions
  - Energy-based generative modeling
  - **Ising Model**, Product of Experts, Restricted Boltzmann machine, Deep Boltzman Machines
  - Training and sampling from EBMs
- Score-based Models
- Denoising Diffusion Models
- Deeper Dive into Diffusion Models

Ising Model

- There is a true image y ∈ {0, 1}$^{3\times3}$, and a corrupted image x ∈ {0, 1}$^{3\times3}$. We know **x**, and want to somehow recover **y**.



- We model the joint probability distribution p(**y**, **x**) as

$$p(\mathbf{y}, \mathbf{x}) = \frac{1}{Z} \exp\left(\sum_i \psi_i\left(x_i, y_i\right) + \sum_{(i,j)\in E} \psi_{ij}\left(y_i, y_j\right)\right)$$

  - $\psi_i\left(x_i, y_i\right)$: the i-th corrupted pixel depends on the i-th original pixel
  - $\psi_{ij}\left(y_i, y_j\right)$: neighboring pixels tend to have the same value

- How did the original image **y** look like? Solution: maximize p(**y**|**x**). Or equivalently, maximize p(**y**, **x**).

Lecture overview

- Motivation
- **Energy-based models**
  - Parametrizing probability distributions
  - Energy-based generative modeling
  - Ising Model, **Product of Experts**, Restricted Boltzmann machine, Deep Boltzman Machines
  - Training and sampling from EBMs
- Score-Based Models
- Denoising Diffusion Models
- Deeper Dive into Diffusion Models

Product of Experts

- Suppose you have trained several models $q_{\theta_1}(\mathbf{x}), r_{\theta_2}(\mathbf{x}), t_{\theta_3}(\mathbf{x})$. They can be different models (PixelCNN, Flow, etc.)
- Each one is like an expert that can be used to score how likely an input $\mathbf{x}$ is.
- Assuming the experts make their judgments indpendently, it is tempting to ensemble them as

$$p_{\theta_1}(\mathbf{x})q_{\theta_2}(\mathbf{x})r_{\theta_3}(\mathbf{x})$$

- To get a valid probability distribution, we need to normalize

$$p_{\theta_1,\theta_2,\theta_3}(\mathbf{x}) = \frac{1}{Z(\theta_1,\theta_2,\theta_3)} q_{\theta_1}(\mathbf{x})r_{\theta_2}(\mathbf{x})t_{\theta_3}(\mathbf{x})$$

- Note: similar to an AND operation (e.g., probability is zero as long as one model gives zero probability), unlike mixture models which behave more like OR

Product of Experts

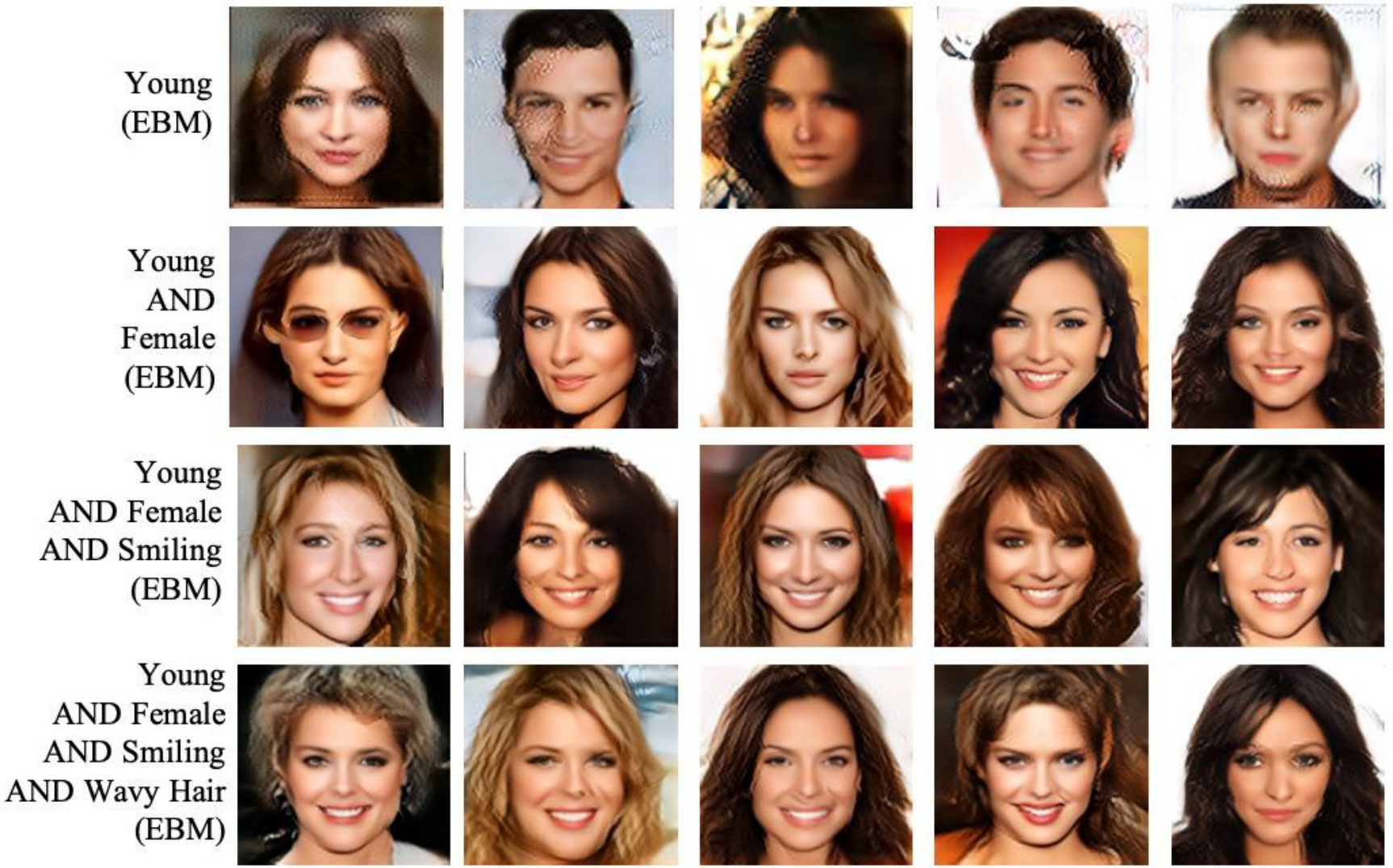


Image source: Du et al., 2020.

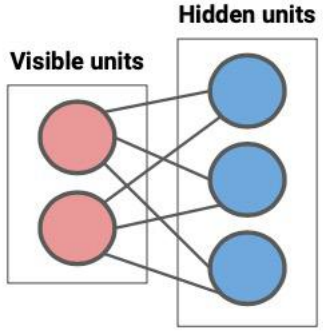
Lecture overview

- Motivation
- **Energy-based models**
  - Parametrizing probability distributions
  - Energy-based generative modeling
  - Ising Model, Product of Experts, **Restricted Boltzmann machine**, Deep Boltzman Machines
  - Training and sampling from EBMs
- Score-based Models
- Denoising Diffusion Models
- Deeper Dive into Diffusion Models

Restricted Boltzmann machine (RBM)

- RBM: energy-based model with latent variables
- Two types of variables:
  - $\mathbf{x} \in \{0, 1\}^n$ are visible variables (e.g., pixel values)
  - $\mathbf{z} \in \{0, 1\}^m$ are latent ones
- The joint distribution is

$$p_{W,b,c}(\mathbf{x}, \mathbf{z}) = \frac{1}{Z} \exp\left(\mathbf{x}^T W \mathbf{z} + b\mathbf{x} + c\mathbf{z}\right) = \frac{1}{Z} \exp\left(\sum^n \sum^m x_i z_j w_{ij} + b\mathbf{x} + c\mathbf{z}\right)$$



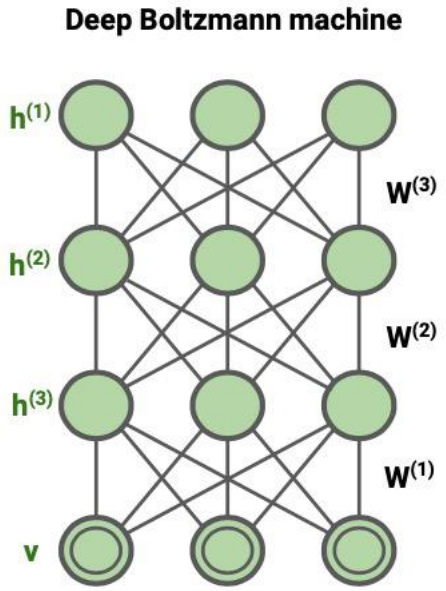
- Restricted because there are no visible-visible and hidden-hidden connections, i.e., x_i x_j or z_i z_j terms in the objective

Lecture overview

- Motivation
- **Energy-based models**
  - Parametrizing probability distributions
  - Energy-based generative modeling
  - Ising Model, Product of Experts, Restricted Boltzmann machine, **Deep Boltzman Machines**
  - Training and sampling from EBMs
- Score-based Models
- Denoising Diffusion Models
- Deeper Dive into Diffusion Models

Deep Boltzmann Machines

- Stacked RBMs are one of the first deep generative models:



- Bottom layer variables v are pixel values. Layers above (**h**) represent "higher-level" features (corners, edges, etc.).
- Early deep neural networks for supervised learning had to be pre-trained like this to make them work.

Deep Boltzmann Machines: Samples

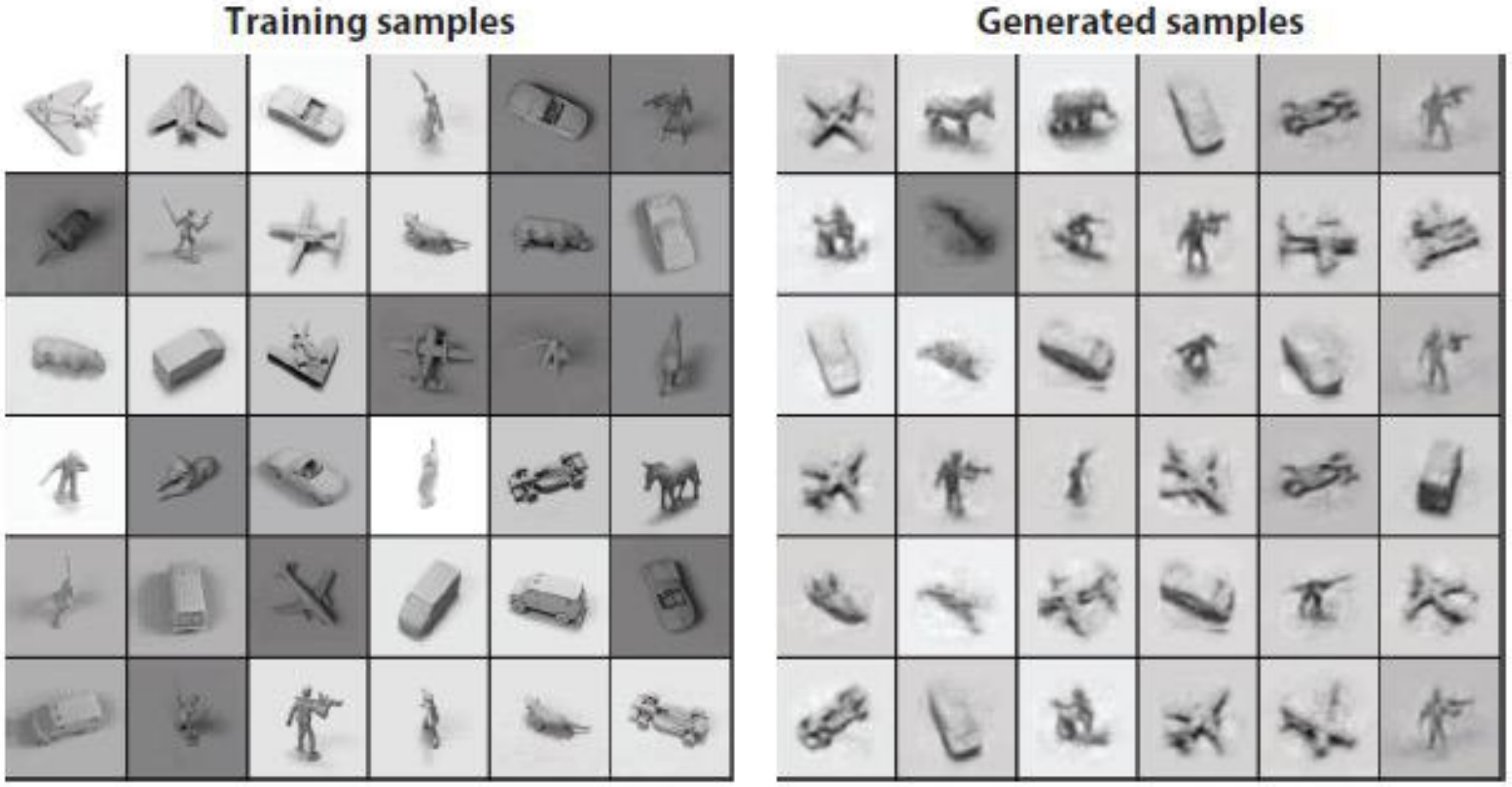


Image source: Salakhutdinov and Hinton, 2009.

Lecture overview

- Motivation
- **Energy-based models**
  - Parametrizing probability distributions
  - Energy-based generative modeling
  - Ising Model, Product of Experts, Restricted Boltzmann machine, Deep Boltzman Machines
  - **Training and sampling from EBMs**
- Score-based Models
- Denoising Diffusion Models
- Deeper Dive into Diffusion Models

Energy-based model

$$p_\theta(\mathbf{x}) = \frac{1}{\int \exp(f_\theta(\mathbf{x}))d\mathbf{x}} \exp\left(f_\theta(\mathbf{x})\right) = \frac{1}{Z(\theta)} \exp\left(f_\theta(\mathbf{x})\right)$$

- Pros:
  - can plug in pretty much any function $f_\theta(\mathbf{x})$ you want
- Cons:
  - Sampling is hard
  - Evaluating likelihood (learning) is hard
  - No feature learning

- **Curse of dimensionality:** The fundamental issue is that computing $Z(\theta)$ numerically (when no analytic solution is available) scales exponentially in the number of dimensions of **x**

Computing the normalization constant is hard

- As an example, the RBM joint distribution is

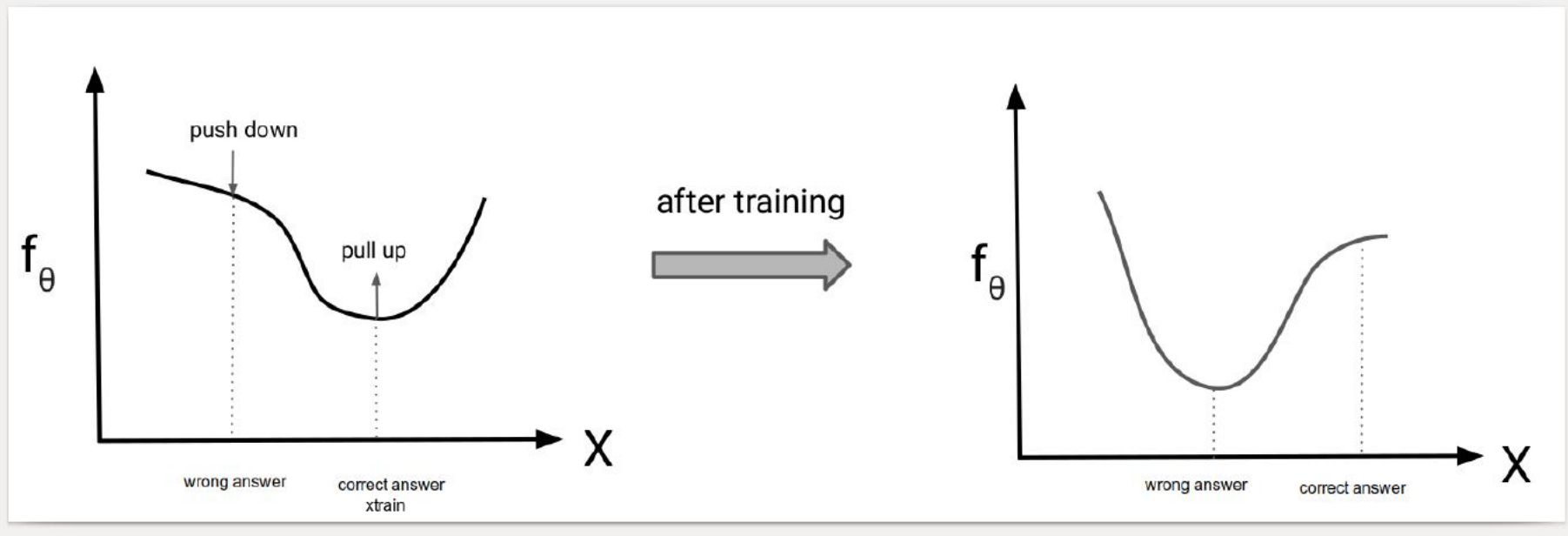
$$p_{W,b,c}(\mathbf{x}, \mathbf{z}) = \frac{1}{Z} \exp\left(\mathbf{x}^T W \mathbf{z} + b\mathbf{x} + c\mathbf{z}\right)$$

  where
  - $\mathbf{x} \in \{0, 1\}^n$ are visible variables (e.g., pixel values)
  - $\mathbf{z} \in \{0, 1\}^m$ are latent ones
- The normalization constant (the "volume") is

$$Z(W, b, c) = \sum_{\mathbf{x} \in \{0,1\}^n} \sum_{\mathbf{z} \in \{0,1\}^m} \exp\left(\mathbf{x}^T W \mathbf{z} + b\mathbf{x} + c\mathbf{z}\right)$$

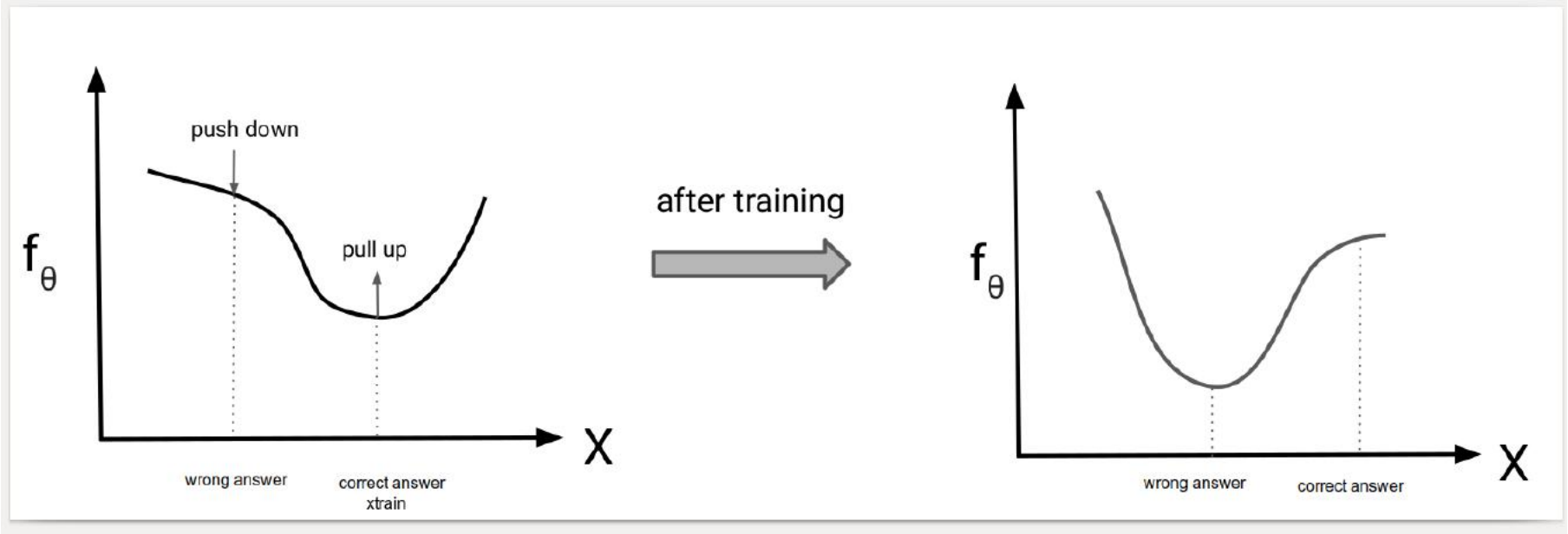
- **Note:** it is a well-defined function of the parameters W, b, c, but no simple closed-form. Takes time exponential in n, m to compute. This means that evaluating the objective function $p_{W,b,c}(\mathbf{x}, \mathbf{z})$ for likelihood-based learning is hard.
- **Observation:** Optimizing the likelihood $p_{W\,b,c}(\mathbf{x}, \mathbf{z})$ is difficult, but optimizing the un-normalized probability $\exp\left(\mathbf{x}^T W \mathbf{z} + b\mathbf{x} + c\mathbf{z}\right)$ (wrt trainable parameters W, b, c) is easy.

Training intuition



- **Goal**: $\text{maximize } \frac{\exp\{f_\theta(\mathbf{x}_{\text{train}})\}}{Z(\theta)}$. Increase numerator, decrease denominator.
- **Intuition**: because the model is not normalized, increasing the un-normalized log-probability $f_\theta(\mathbf{x}_{\text{train}})$ by changing θ does **not** guarantee that $\mathbf{x}_{\text{train}}$ becomes relatively more likely (compared to the rest).
- We also need to take into account the effect on other "wrong points" and try to "push them down" to also make $Z(\theta)$ small.

Constrastive Divergence



- **Goal**: $\text{maximize } \frac{\exp\{f_\theta(\mathbf{x}_{\text{train}})\}}{Z(\theta)}$
- **Idea**: Instead of evaluating.$Z(\theta)$ exactly, use a Monte Carlo estimate.
- Contrastive divergence algorithm: sample $x_{\text{sample}} \sim p_\theta$ take step on $\nabla_\theta \left(f_\theta\left(x_{\text{train}}\right) - f_\theta\left(x_{\text{sample}}\right)\right)$. Make training data more likely than typical sample from the model.

Contrastive Divergence

- Maximize log-likelihood: $\max_\theta f_\theta\left(x_{\text{train}}\right) - \log Z(\theta)$
- Gradient of log-likelihood:

$$
\begin{aligned}
& \nabla_\theta f_\theta\left(x_{\text{train}}\right) - \nabla_\theta \log Z(\theta) \\
= \quad & \nabla_\theta f_\theta\left(x_{\text{train}}\right) - \frac{\nabla_\theta Z(\theta)}{Z(\theta)} \\
= \quad & \nabla_\theta f_\theta\left(x_{\text{train}}\right) - \frac{1}{Z(\theta)} \int \nabla_\theta \exp\left\{f_\theta(x)\right\} \mathrm{d}x \\
= \quad & \nabla_\theta f_\theta\left(x_{\text{train}}\right) - \frac{1}{Z(\theta)} \int \exp\left\{f_\theta(x)\right\} \nabla_\theta f_\theta(x) \mathrm{d}x \\
= \quad & \nabla_\theta f_\theta\left(x_{\text{train}}\right) - \int \frac{\exp\{f_\theta(x)\}}{Z(\theta)} \nabla_\theta f_\theta(x) \mathrm{d}x \\
= \quad & \nabla_\theta f_\theta\left(x_{\text{train}}\right) - E_{x_{\text{sample}}}\left[\nabla_\theta f_\theta\left(x_{\text{sample}}\right)\right] \\
\approx \quad & \nabla_\theta f_\theta\left(x_{\text{train}}\right) - \nabla_\theta f_\theta\left(x_{\text{sample}}\right),
\end{aligned}
$$

where $x_{\text{sample}} \sim \exp\left\{f_\theta\left(x_{\text{sample}}\right)\right\} / Z(\theta)$

- How to sample?

Sampling from energy-based models

$$p_\theta(\mathbf{x}) = \frac{1}{\int \exp(f_\theta(\mathbf{x}))d\mathbf{x}} \exp\left(f_\theta(\mathbf{x})\right) = \frac{1}{Z(\theta)} \exp\left(f_\theta(\mathbf{x})\right)$$

- No direct way to sample like in autoregressive or flow models. Main issue: cannot easily compute how likely each possible sample is
- However, we can easily compare two samples x,x′.
- Use an iterative approach called Markov Chain Monte Carlo:
  1. Initialize x^0 randomly, t = 0
  2. Let $x' = x^t + \text{noise}$
     - If $f_\theta(x') > f_\theta(x^t)$, let $x^{t+1} = x'$
     - Else let $x^{t+1} = x'$ with probability $\exp(f_\theta(x') - f_\theta(x^t))$
  3. Goto step 2
- Works in theory, but can take a very long time to converge

Sampling from energy-based models

- For any continuous distribution $p_\theta(\mathbf{x})$, suppose we can compute its gradient (the **score function**) $\nabla_{\mathbf{x}} \log p_\theta(\mathbf{x})$
- Let $\pi(x)$ be a prior distribution that is easy to sample from.
- Langevin MCMC.
  - $\mathbf{x}^0 \sim \pi(\mathbf{x})$
  - Repeat $\mathbf{x}^{t+1} \sim \mathbf{x}^t + \epsilon \nabla_{\mathbf{x}} \log p_\theta\left(\mathbf{x}^t\right) + \sqrt{2\epsilon}\mathbf{z}^t$ for $t = 0, 1, 2, \cdots, T-1$, where $\mathbf{z}^t \sim \mathcal{N}(0, I)$.
  - If $\varepsilon \to 0$ and $T \to \infty$, we have $\mathbf{x}_T \sim p_\theta(\mathbf{x})$.
- Note that for energy-based models

$$\nabla_{\mathbf{x}} \log p_\theta(\mathbf{x}) = \nabla_{\mathbf{x}} f_\theta(\mathbf{x}) - \underbrace{\nabla_{\mathbf{x}} \log Z(\theta)}_{=0}$$

$$= \nabla_{\mathbf{x}} f_\theta(\mathbf{x})$$

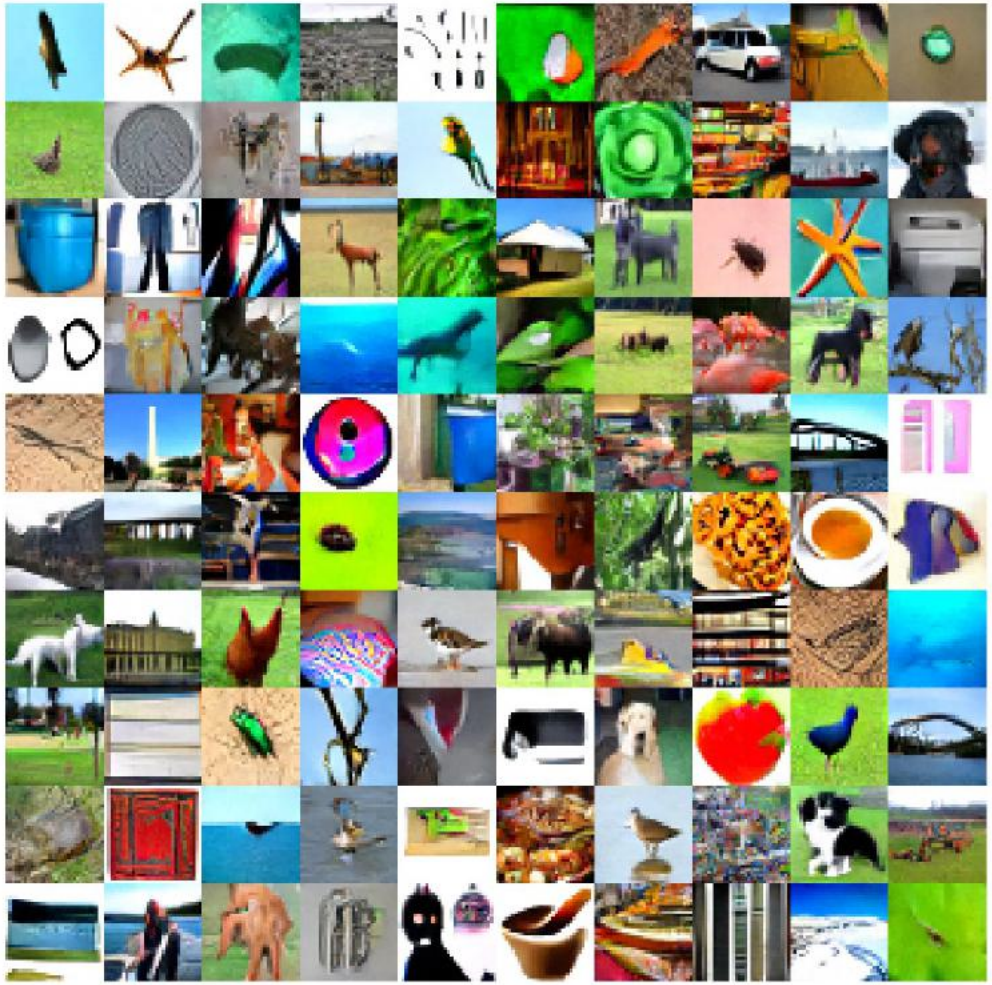
Modern energy-based models

Langevin sampling

Face samples

Image source: Nijkamp et al. 2019

Modern energy-based models



ImageNet samples

Image source: Du et al. 2019

Lecture overview

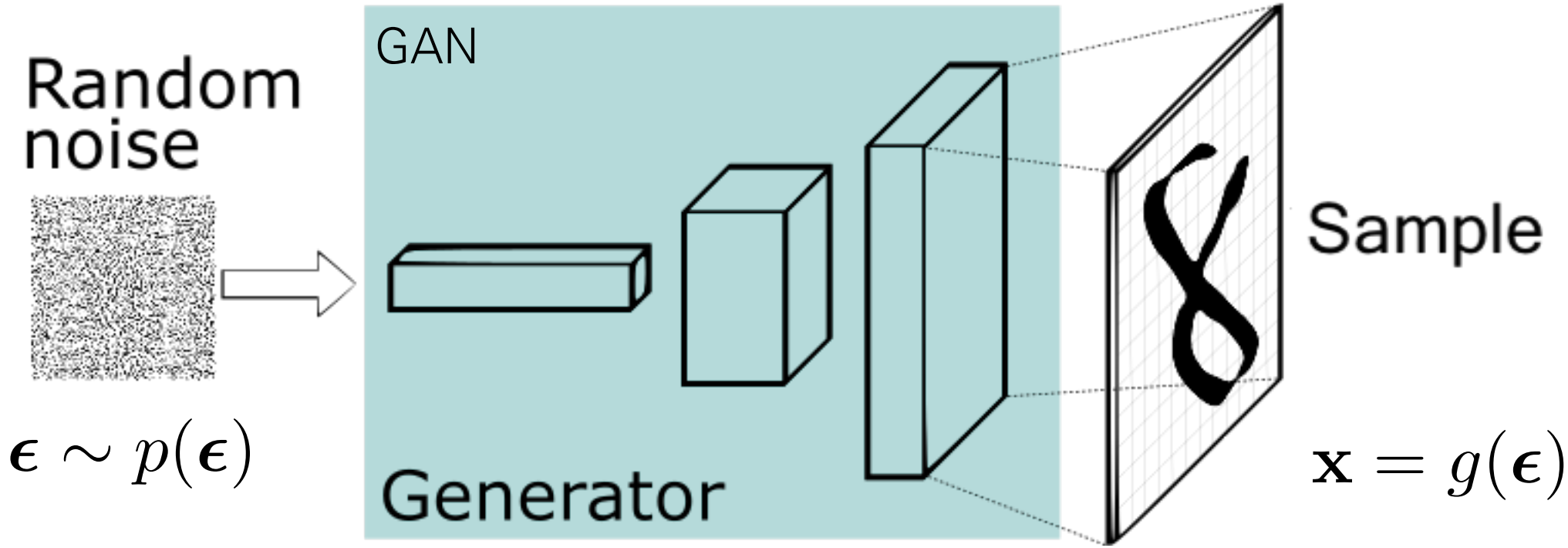
- Energy-based models
- **Score-based Models**
  - Probability distributions and Score functions
  - Score-based Generative Models
  - Sampling from a Score-based Model
  - Latent Score-based Generative Models
- Denoising Diffusion Models
- Deeper Dive into Diffusion Models

Lecture overview

- Energy-based models
- **Score-based Models**
  - **Probability Distribution and Score functions**
  - Score-based Generative Models
  - Sampling from a Score-based Model
  - Latent Score-based Generative Models
- Denoising Diffusion Models
- Deeper Dive into Diffusion Models

Representations of Probability Distributions

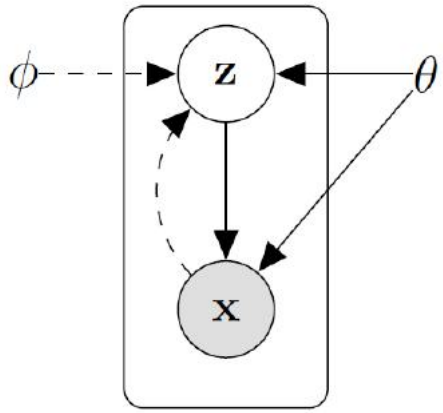
Implicit models: directly represent the sampling process



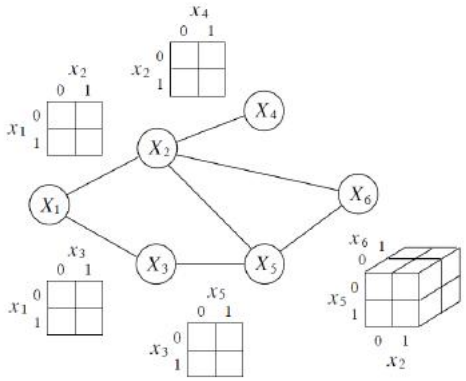
- **Cons:** hard to train, no likelihood, no principled model comparisons

Representations of Probability Distributions

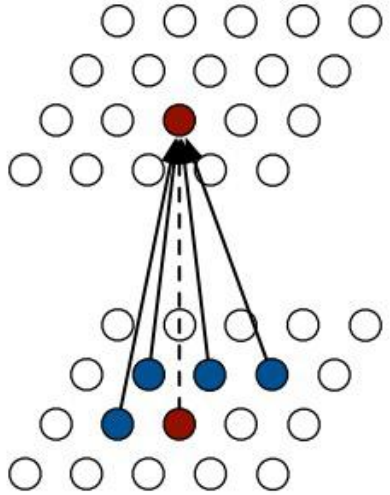
Explicit models: represent a probability density/mass function $p(\mathbf{x})$



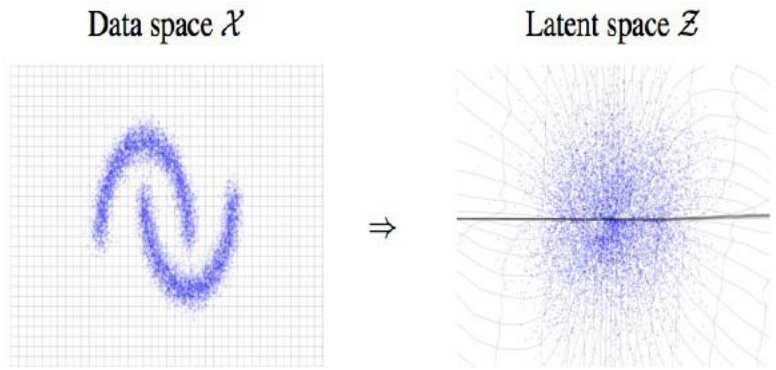
Bayesian networks (e.g., VAEs)



MRF



Autoregressive models



Flow models

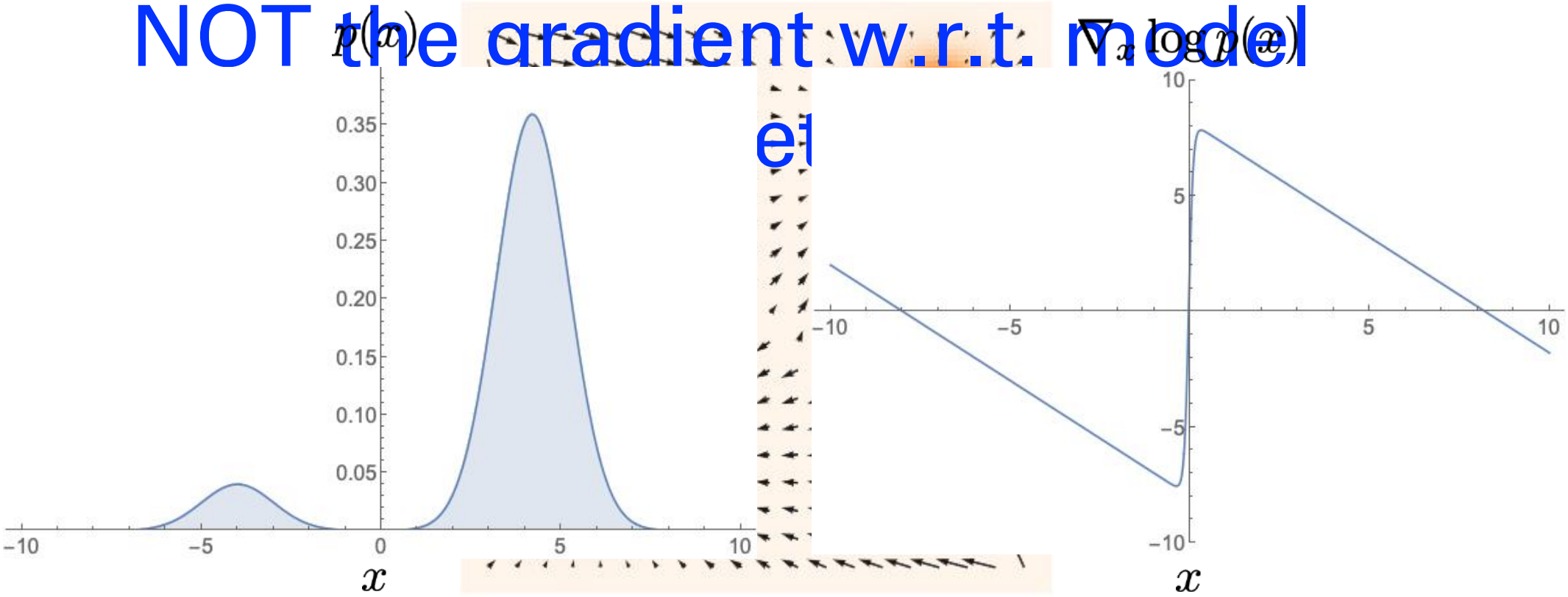
- **Cons**: need to be normalized → balance expressivity and tractability

Representation of Probability Distributions

Alternative: The gradient of a probability density wrt the input dimensions

$$\nabla_{\mathbf{x}} \log p(\mathbf{x})$$ Score

NOT the gradient w.r.t. model



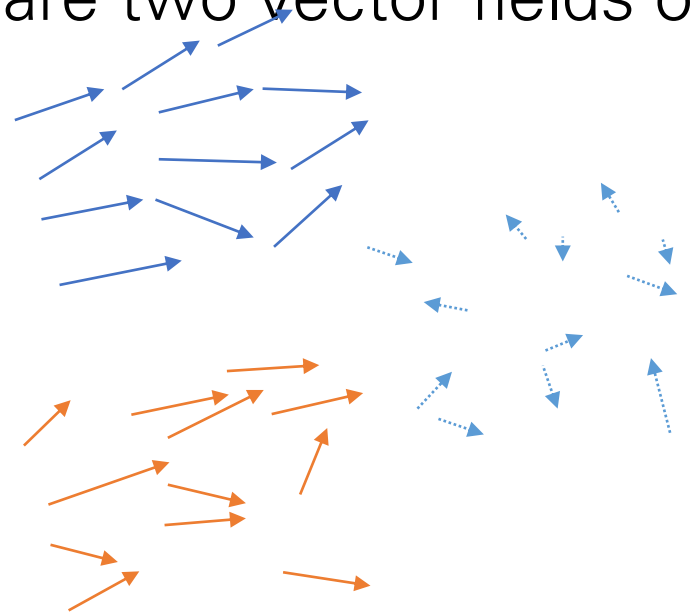
(pdf and score)

Score Estimation

- **Given:** i.i.d. samples $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\} \overset{\text{i.i.d.}}{\sim} p_{\text{data}}(\mathbf{x})$
- **Task:** Estimating the score $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$
- **Score Model:** A trainable vector-valued function $s_\theta(\mathbf{x}) : \mathbb{R}^D \to \mathbb{R}^D$
- **Objective:** How to compare two vector fields of scores?

$\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$

$s_\theta(\mathbf{x})$



Average Euclidean distance over the space

Score Estimation

- **Given:** i.i.d. samples $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\} \overset{\text{i.i.d.}}{\sim} p_{\text{data}}(\mathbf{x})$
- **Task:** Estimating the score $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$
- **Score Model:** A trainable vector-valued function $s_\theta(\mathbf{x}) : \mathbb{R}^D \to \mathbb{R}^D$
- **Objective:** How to compare two vector fields of scores?

$$\frac{1}{2}\mathbb{E}_{p_{\text{data}}}[\|\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) - s_\theta(\mathbf{x})\|_2^2]$$

?

(Fisher divergence)

- Integration by parts

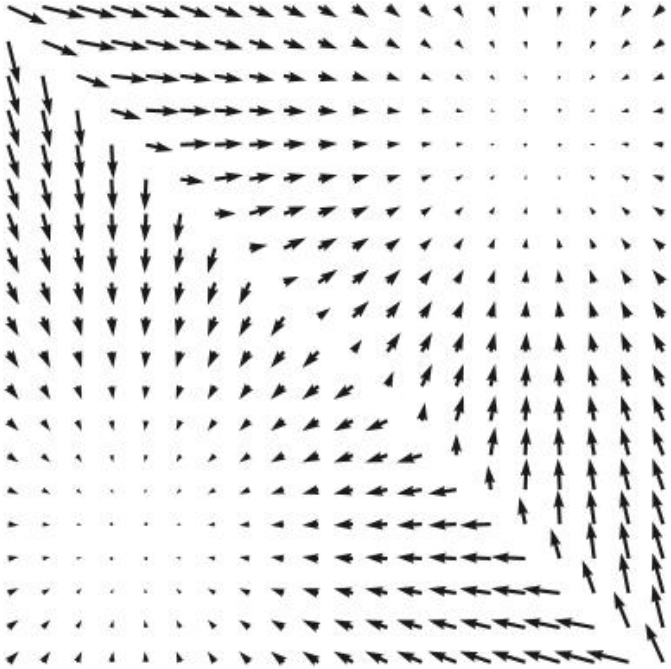
$$\mathbb{E}_{p_{\text{data}}}\left[\frac{1}{2}\|s_\theta(\mathbf{x})\|_2^2 + \text{trace}(\nabla_{\mathbf{x}} s_\theta(\mathbf{x}))\right]$$

$$\approx \frac{1}{N}\sum_{i=1}^{N}\left[\frac{1}{2}\|s_\theta(\mathbf{x}_i)\|_2^2 + \text{trace}(\nabla_{\mathbf{x}} s_\theta(\mathbf{x}_i))\right]$$

Score Matching

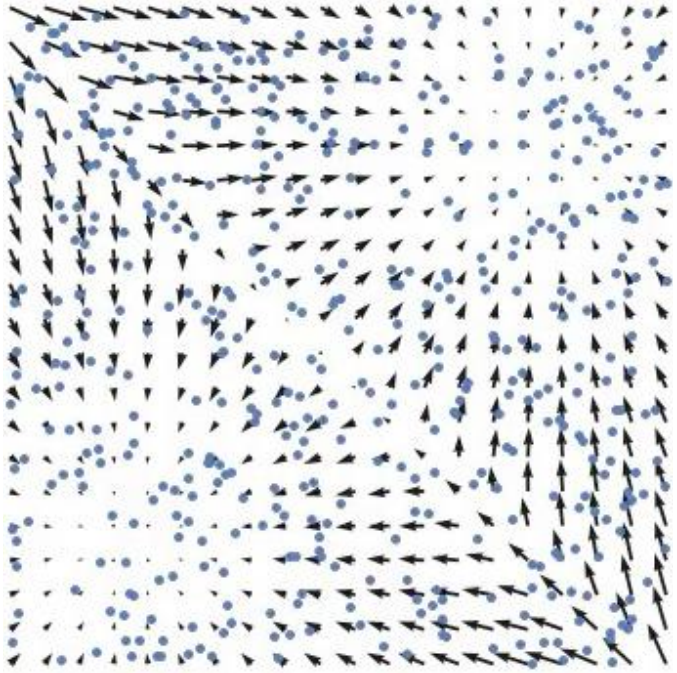
Hyvärinen (2005)

From Scores to Samples: Langevin Dynamics



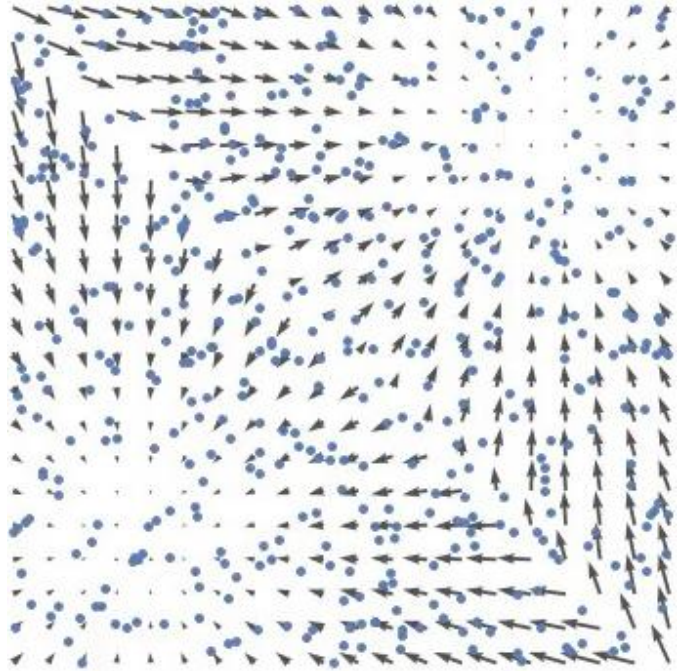
Scores

$$s_\theta(\mathbf{x})$$



Follow the scores

$$\tilde{\mathbf{x}}_{t+1} \leftarrow \tilde{\mathbf{x}}_t + \frac{\epsilon}{2} s_\theta(\tilde{\mathbf{x}}_t)$$



Follow noisy scores: Langevin dynamics

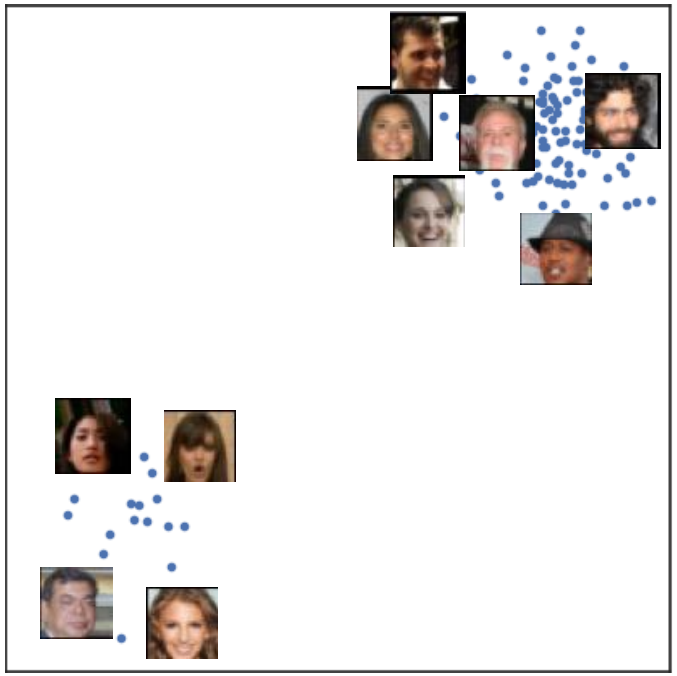
$$\mathbf{z}_t \sim \mathcal{N}(0, I)$$

$$\tilde{\mathbf{x}}_{t+1} \leftarrow \tilde{\mathbf{x}}_t + \frac{\epsilon}{2} s_\theta(\tilde{\mathbf{x}}_t) + \sqrt{\epsilon}\mathbf{z}_t$$

Lecture overview

- Energy-based models
- **Score-based Models**
  - Probability Distribution and Score functions
  - **Score-based Generative Models**
  - Sampling from a Score-based Model
  - Latent Score-based Generative Models
- Denoising Diffusion Models
- Deeper Dive into Diffusion Models

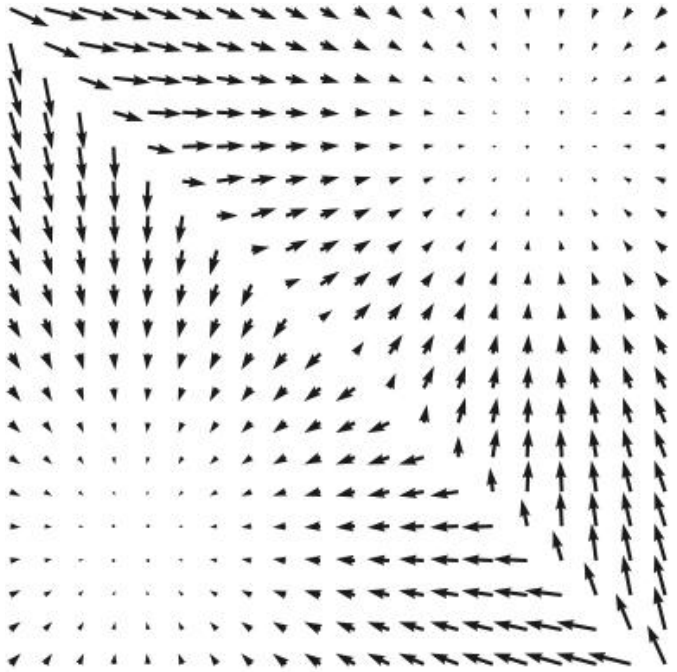
Score-Based Generative Modeling



Data samples

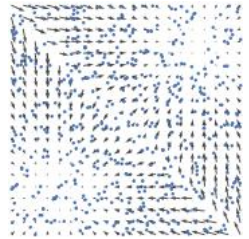
$$\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\} \overset{\text{i.i.d.}}{\sim} p_{\text{data}}(\mathbf{x})$$

Score Matching



Scores

$$s_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$$

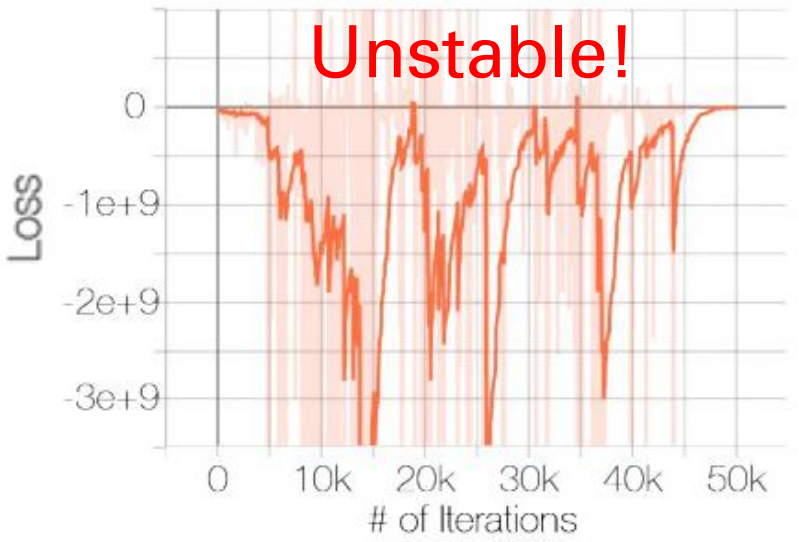


Langevin dynamics

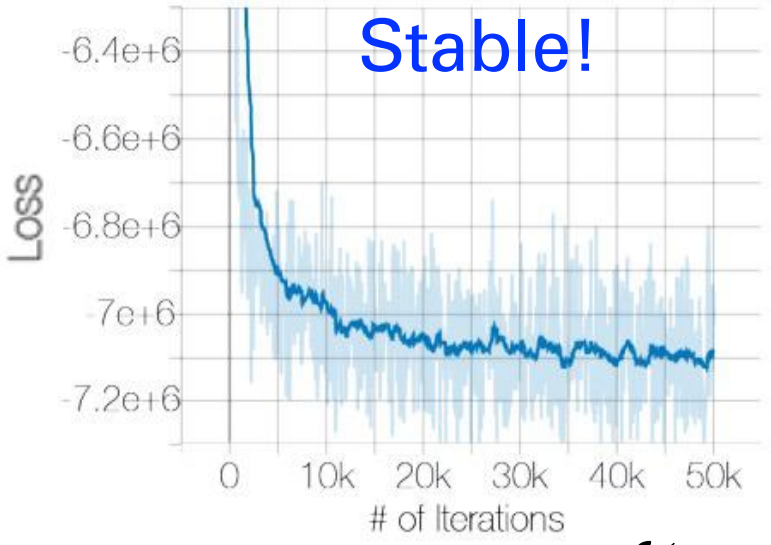
New samples

Adding Noise to Data for Well-Defined Scores

- Scores can be undefined when
  - The support of data distribution is on a low-dimensional manifold
  - The data distribution is discrete
- Solution: adding noise

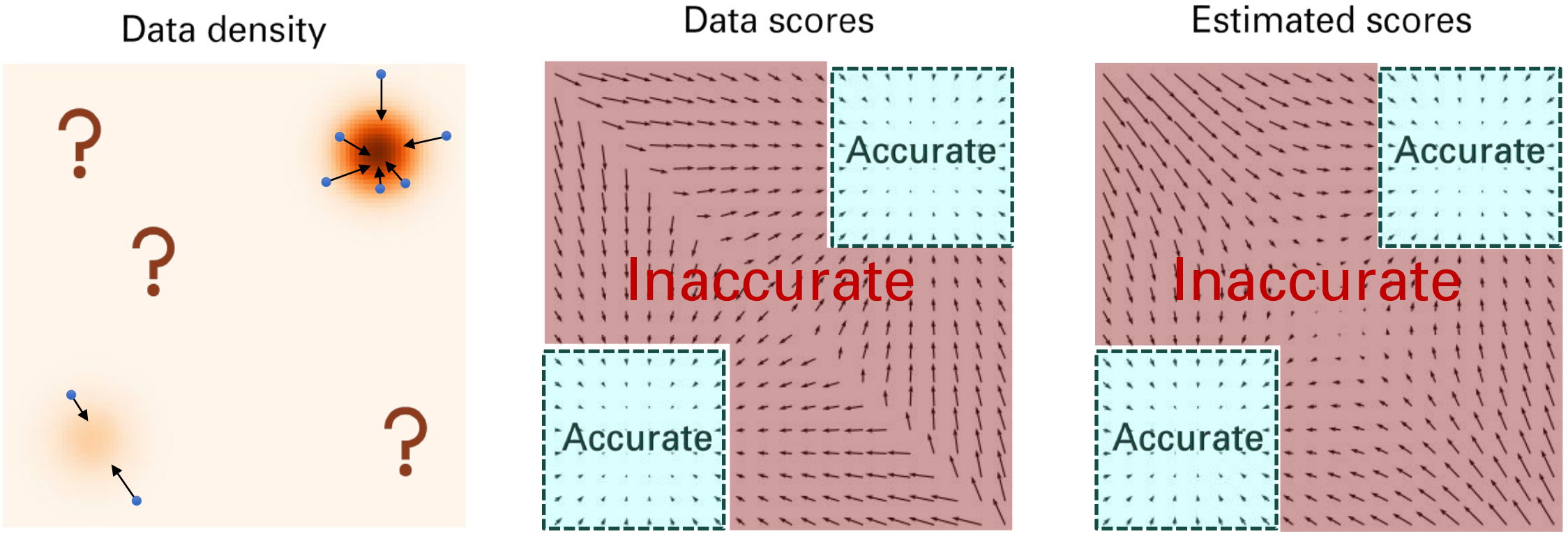


Data unperturbed



Data perturbed with $\mathcal{N}(0; 0.0001)$

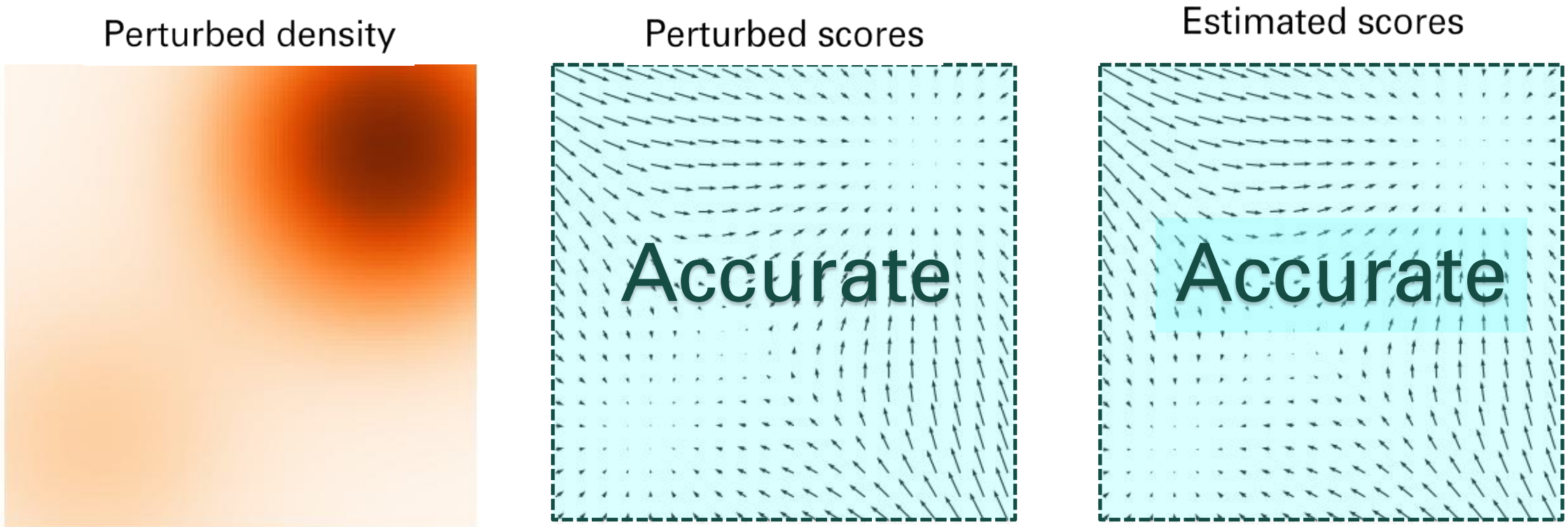
Challenge in Low Data Density Regions



$$\frac{1}{2}\mathbb{E}_{p_{\text{data}}}[\|\nabla_{\mathbf{x}}\log p_{\text{data}}(\mathbf{x}) - s_\theta(\mathbf{x})\|_2^2] \approx \frac{1}{2N}\sum_{i=1}^{N}\|\nabla_{\mathbf{x}}\log p_{\text{data}}(\mathbf{x}_i) - s_\theta(\mathbf{x}_i)\|_2^2$$

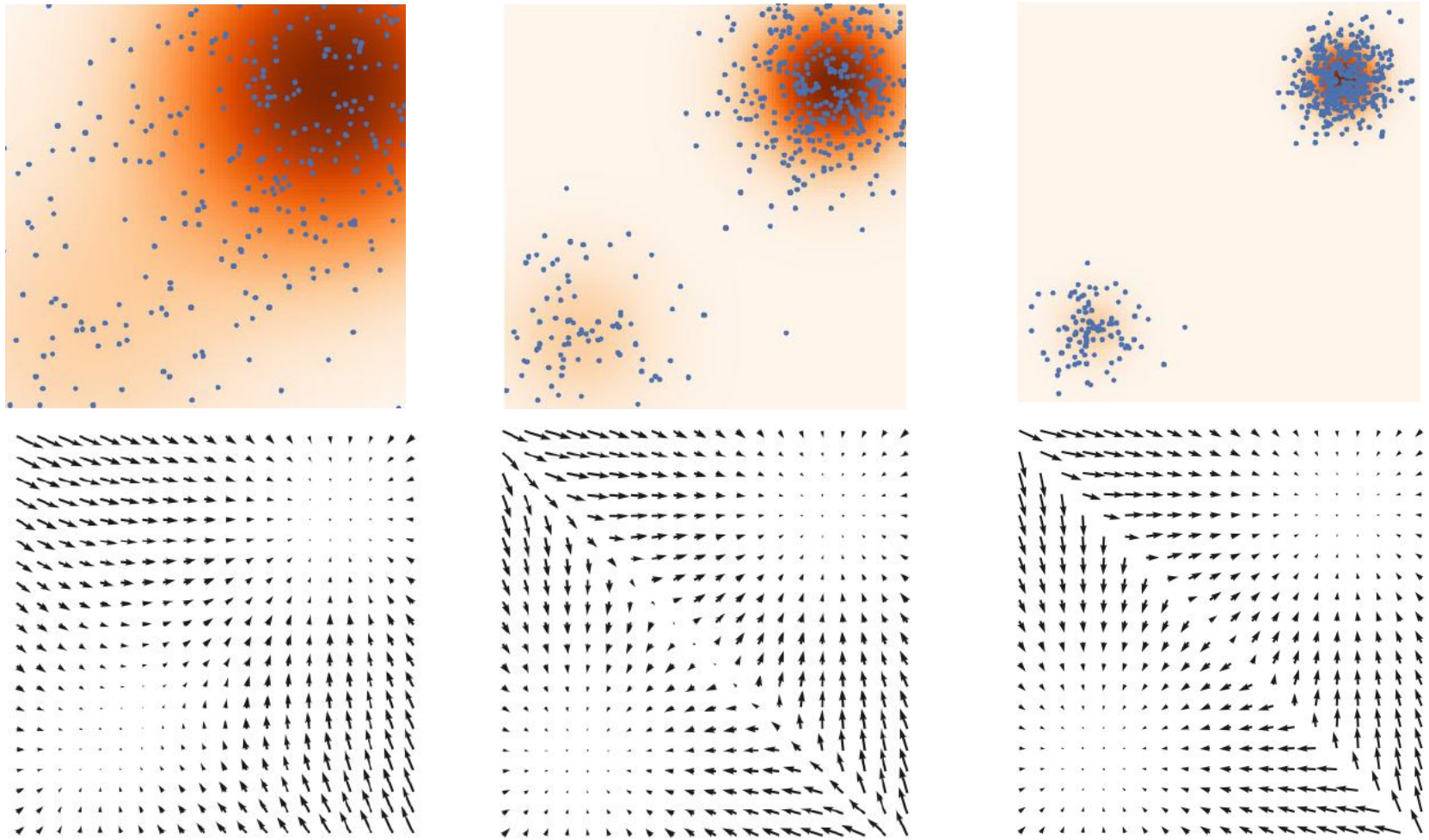
Adding Noise to Data for Better Score Estimation

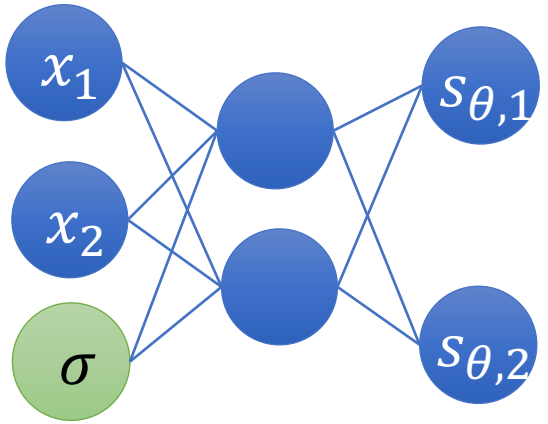
- Random noise provides samples in low data density regions.



Joint Score Estimation via Noise Conditional Score Networks

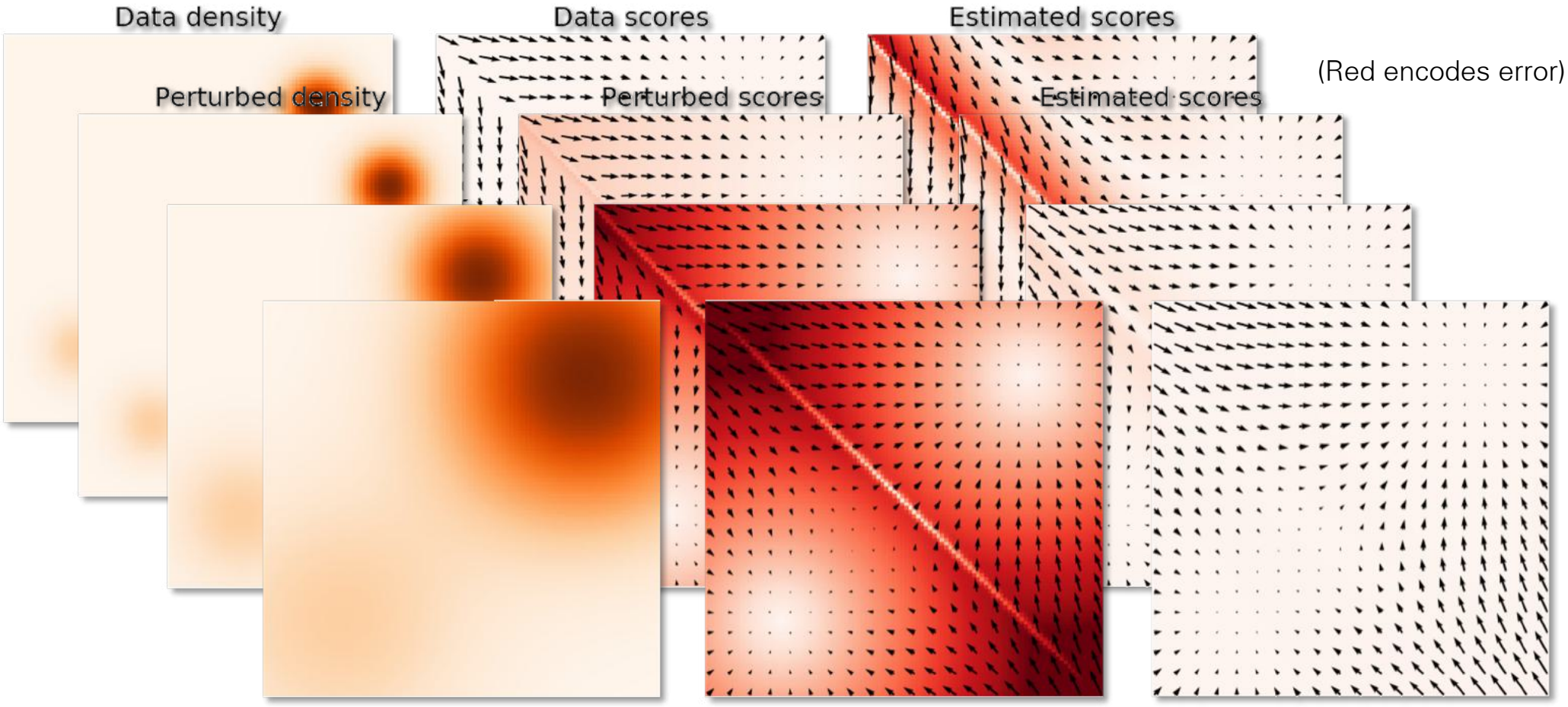
σ_1 σ_2 σ_3





Noise Conditional Score Network (NCSN)

Sample Quality vs. Estimation Accuracy



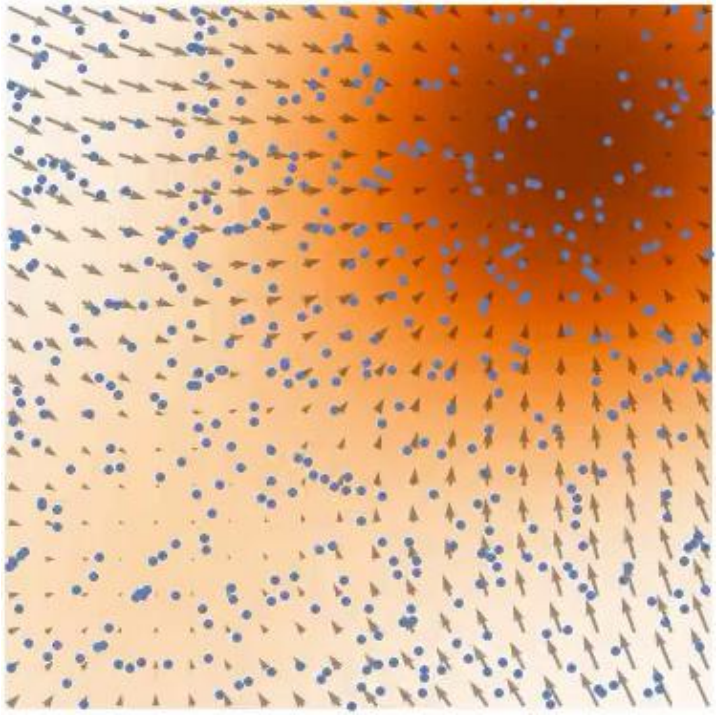
Lecture overview

- Energy-based models
- **Score-based Models**
  - Probability Distribution and Score functions
  - Score-based Generative Models
  - **Sampling from a Score-based Model**
  - Latent Score-based Generative Models
- Denoising Diffusion Models
- Deeper Dive into Diffusion Models

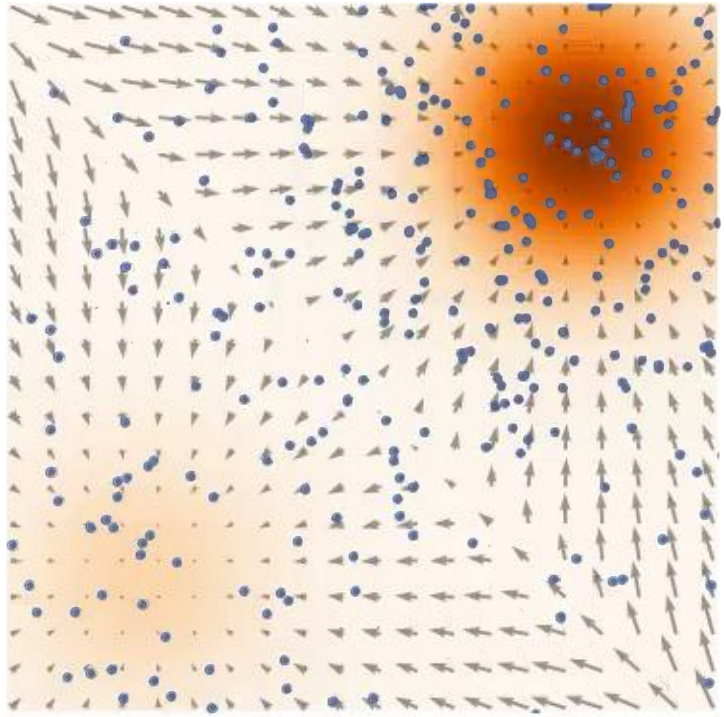
Annealed Langevin Dynamics

Joint Scores to Samples

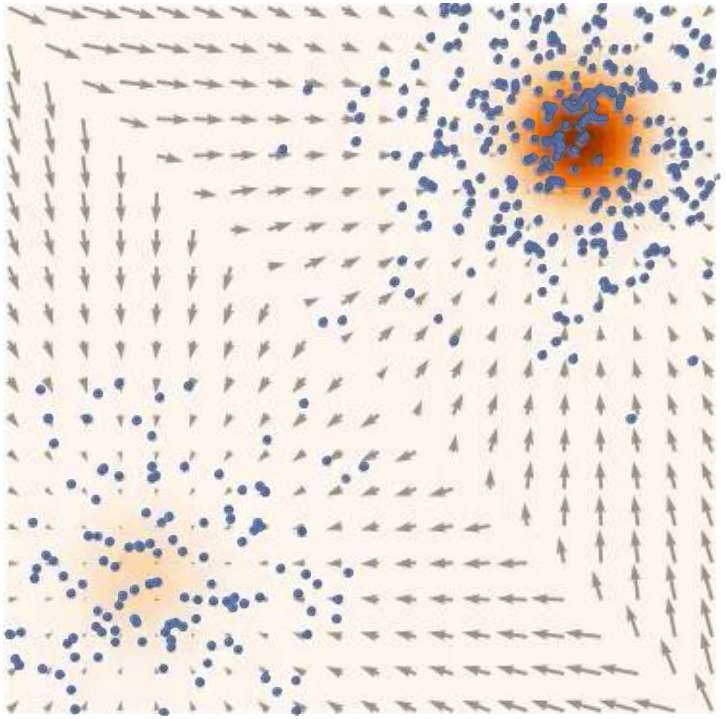
- Sample using $\sigma_1, \sigma_2, \cdots, \sigma_L$ sequentially with Langevin dynamics.
- Anneal down the noise level.
- Samples used as initialization for the next level.



σ_1

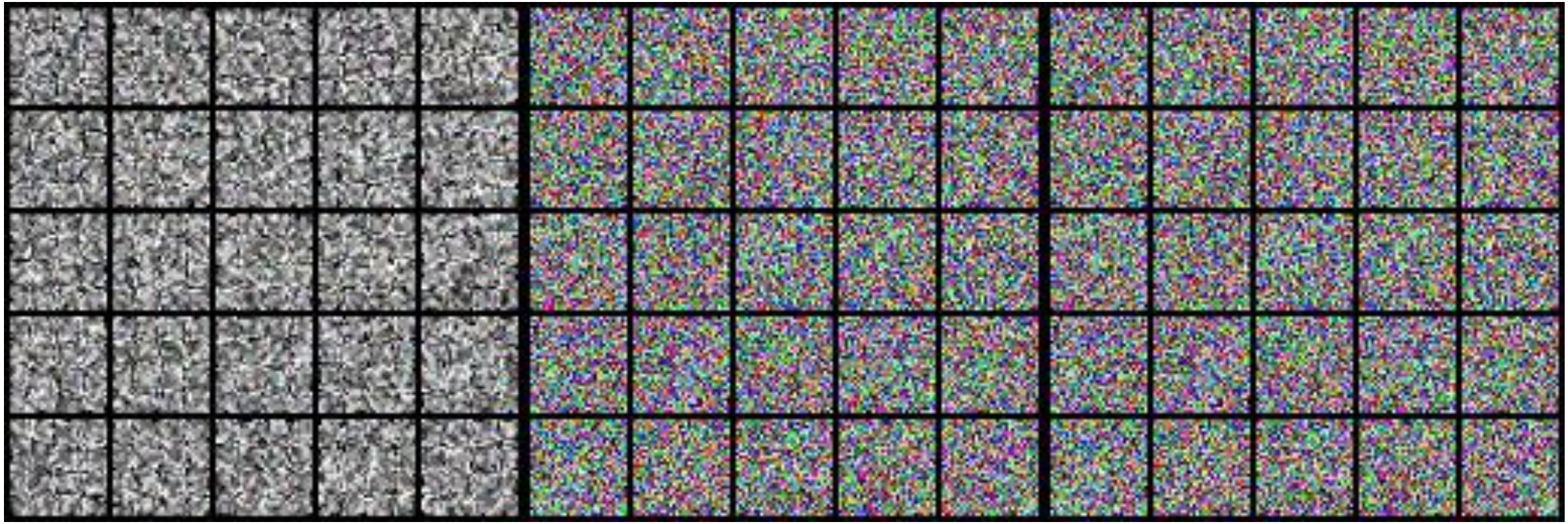


σ_2



σ_3

Experiments: Sampling



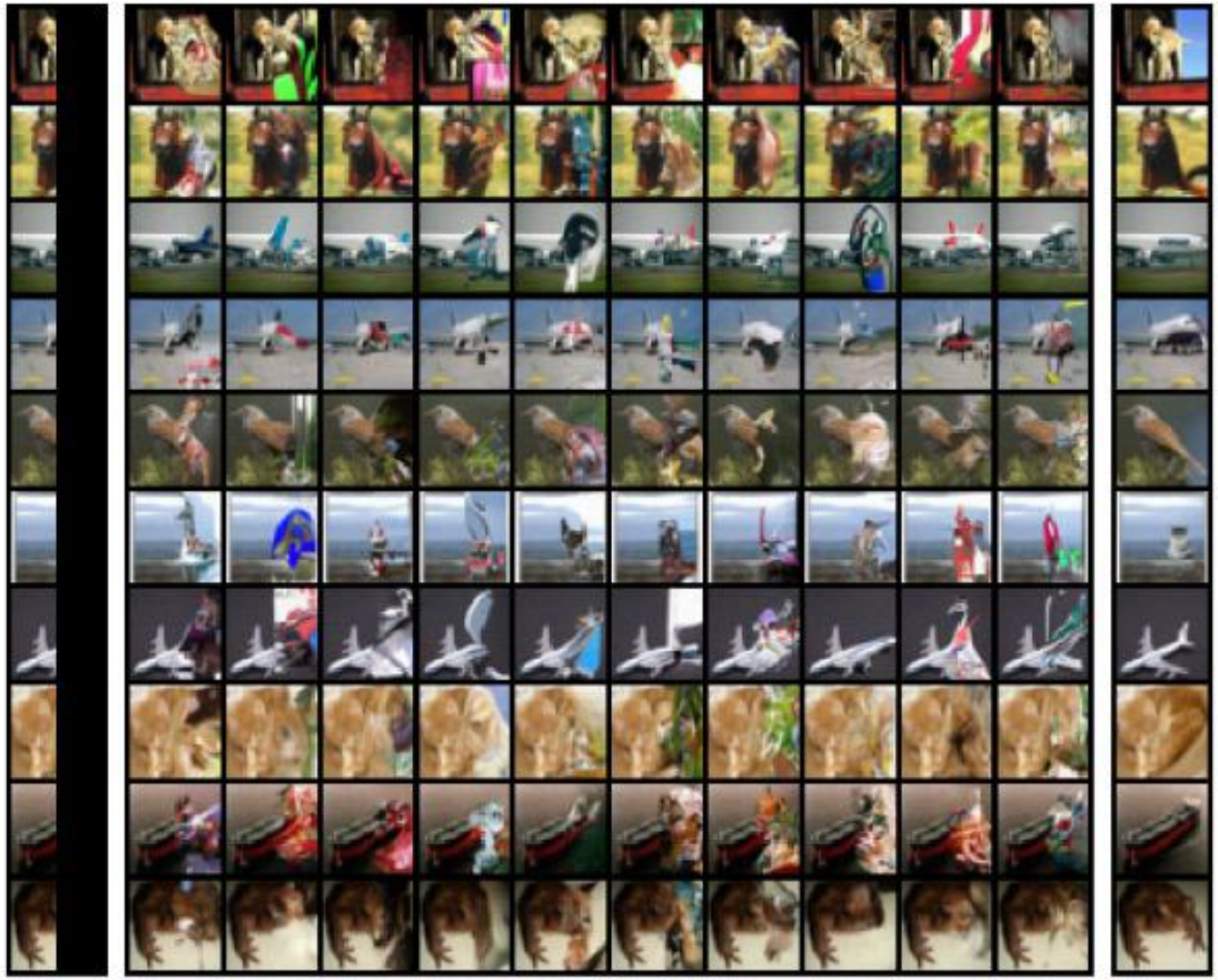
Experiments: Sample Quality

CIFAR-10 Unconditional

Model	Inception Score (higher is better)	FID score (lower is better)
PixelCNN	4.60	65.93
EBM	6.02	40.58
SNGAN	8.22 ± 0.05	21.7
ProgressiveGAN	8.80 ± 0.05	-
NCSN (ours)	**8.87 ± 0.12**	**25.32**

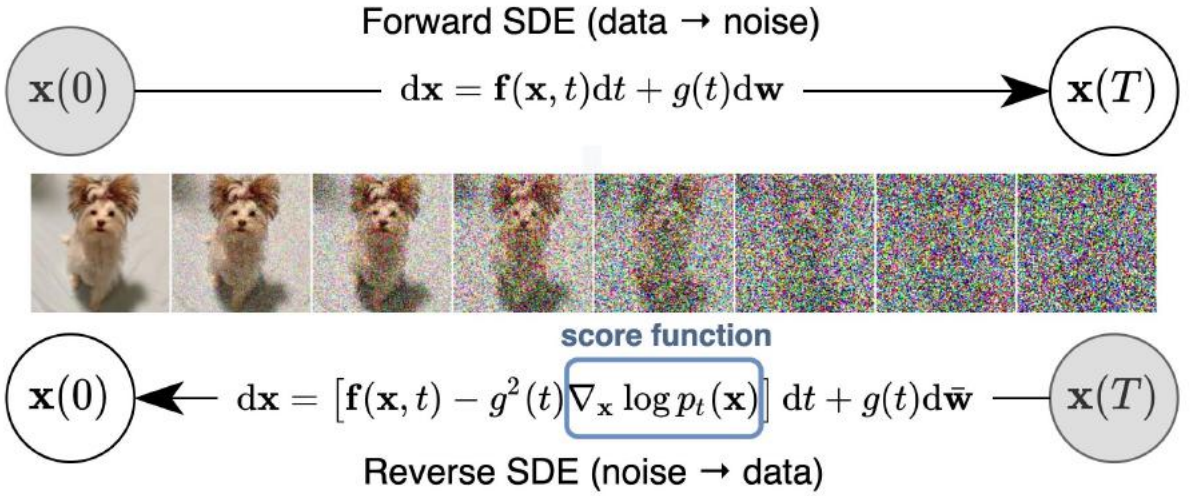
Experiments: Inpainting

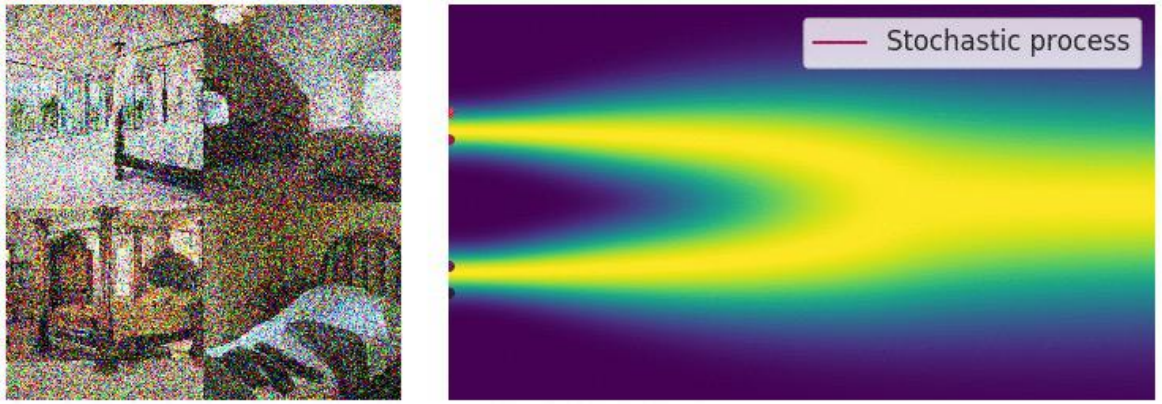
Experiments: Inpainting



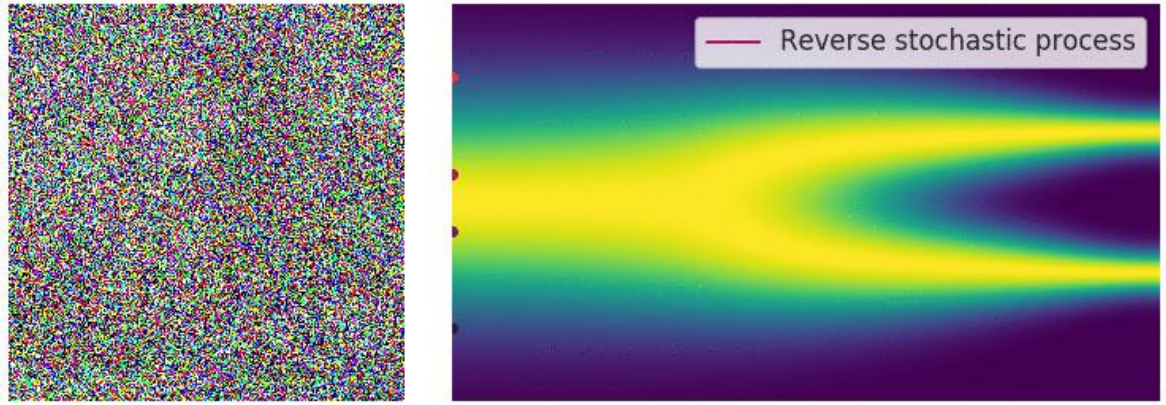
Reversing the SDE for sample generation

- One can generate samples by reversing the perturbation process with annealed Langevin dynamics.
- For infinite noise scales, we can analogously reverse the perturbation process for sample generation by using the reverse SDE.





Perturbing data to noise with a continuous-time stochastic process.



Generating data from noise by reversing the perturbation procedure.

1024 x 1024 samples on FFHQ dataset

256 x 256 samples on LSUN Bedroom

Sample Quality

Table 2: NLLs and FIDs (ODE) on CIFAR-10.

Model	NLL Test ↓	FID ↓
RealNVP (Dinh et al., 2016)	3.49	-
iResNet (Behrmann et al., 2019)	3.45	-
Glow (Kingma & Dhariwal, 2018)	3.35	-
MintNet (Song et al., 2019b)	3.32	-
Residual Flow (Chen et al., 2019)	3.28	46.37
FFJORD (Grathwohl et al., 2018)	3.40	-
Flow++ (Ho et al., 2019)	3.29	-
DDPM (L) (Ho et al., 2020)	$\leqslant 3.70^*$	13.51
DDPM (L_{simple}) (Ho et al., 2020)	$\leqslant 3.75^*$	3.17
DDPM	3.28	3.37
DDPM cont. (VP)	3.21	3.69
DDPM cont. (sub-VP)	3.05	3.56
DDPM++ cont. (VP)	3.16	3.93
DDPM++ cont. (sub-VP)	3.02	3.16
DDPM++ cont. (deep, VP)	3.13	3.08
DDPM++ cont. (deep, sub-VP)	**2.99**	**2.92**

Table 3: CIFAR-10 sample quality.

Model	FID↓	IS↑
Conditional		
BigGAN (Brock et al., 2018)	14.73	9.22
StyleGAN2-ADA (Karras et al., 2020a)	**2.42**	**10.14**
Unconditional		
StyleGAN2-ADA (Karras et al., 2020a)	2.92	9.83
NCSN (Song & Ermon, 2019)	25.32	$8.87 \pm .12$
NCSNv2 (Song & Ermon, 2020)	10.87	$8.40 \pm .07$
DDPM (Ho et al., 2020)	3.17	$9.46 \pm .11$
DDPM++	2.78	9.64
DDPM++ cont. (VP)	2.55	9.58
DDPM++ cont. (sub-VP)	2.61	9.56
DDPM++ cont. (deep, VP)	2.41	9.68
DDPM++ cont. (deep, sub-VP)	2.41	9.57
NCSN++	2.45	9.73
NCSN++ cont. (VE)	2.38	9.83
NCSN++ cont. (deep, VE)	**2.20**	**9.89**

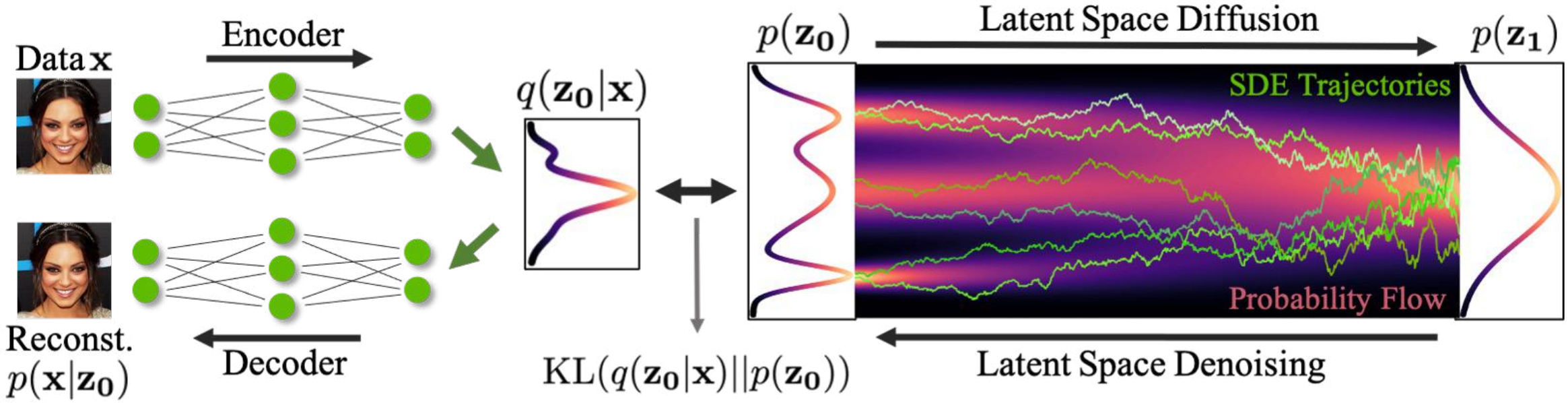
Lecture overview

- Energy-based models
- **Score-based Models**
  - Probability Distribution and Score functions
  - Score-based Generative Models
  - Sampling from a Score-based Model
  - **Latent Score-based Generative Models**
- Denoising Diffusion Models
- Deeper Dive into Diffusion Models

Latent Score-based Generative Models

- Score-based generative models (SGMs) are applied directly in data space and often require 1000s of network evaluations for sampling.

- **Idea:** Can we train SGMs in a latent space?



Latent Score-based Generative Models

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\phi}, \boldsymbol{\theta}, \boldsymbol{\psi}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})}[-\log p_{\boldsymbol{\psi}}(\mathbf{x} \mid \mathbf{z}_0)] + \mathrm{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}_0 \mid \mathbf{x}) \| p_{\boldsymbol{\theta}}(\mathbf{z}_0))$$

$$= \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})}\left[-\log p_{\boldsymbol{\psi}}(\mathbf{x}|\mathbf{z}_0)\right]}_{\text{reconstruction term}} + \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})}\left[\log q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})\right]}_{\text{negative encoder entropy}} + \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x})}\left[-\log p_{\boldsymbol{\theta}}(\mathbf{z}_0)\right]}_{\text{cross entropy}}$$

- A simple expression for the cross-entropy term in the variational loss
- A parameterization of the latent space score function, which mixes a Normal distribution with a learnable SGM.

Also employs a SDE-based variance reduction importance sampling schemes to stably train deep LSGMs.

Latent Score-based Generative Models

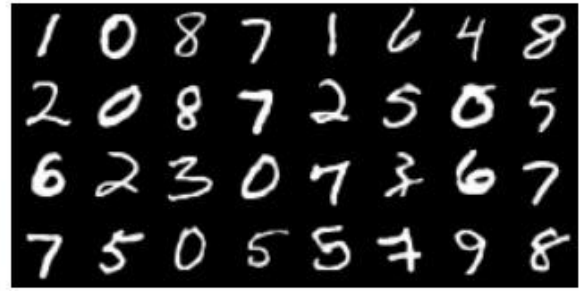
The evolution of the latent variables under the reverse-time generative process by feeding latent variables from different stages along the process to the decoder to map them back to image space

Latent SGM (LSGM) Samples

(a) CIFAR-10

(b) CelebA-HQ-256

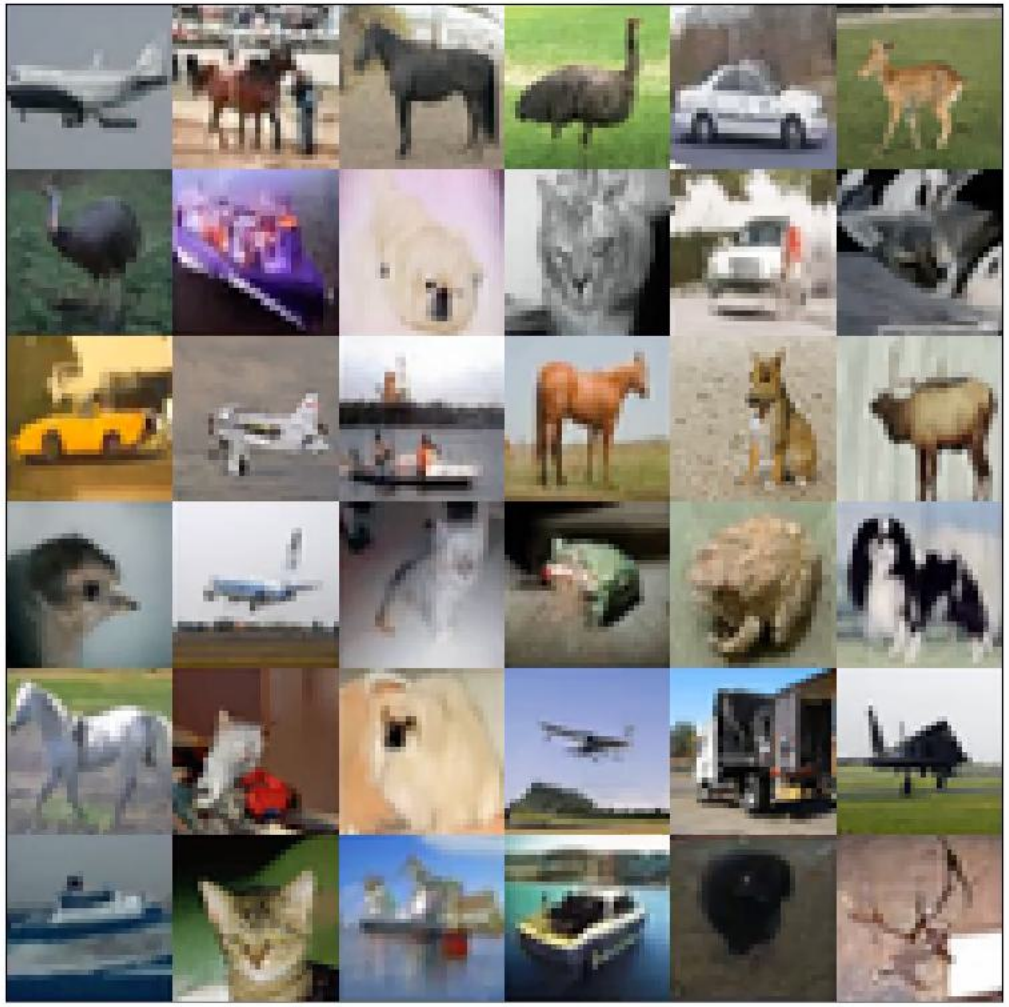
(c) OMNIGLOT



(d) MNIST

Traversing in the latent space of LSGM.

CIFAR-10



CelebA-HQ-256

LSGM Sample Quality

Table 2: Generative performance on CIFAR-10.

	Method	NLL↓	FID↓
Ours	LSGM (FID)	≤3.43	**2.10**
	LSGM (NLL)	≤**2.87**	6.89
	LSGM (balanced)	≤2.95	2.17
	VAE Backbone	2.96	43.18
VAEs	VDVAE [21]	**2.87**	-
	NVAE [20]	2.91	23.49
	VAEBM [76]	-	12.19
	NCP-VAE [56]	-	24.08
	BIVA [48]	3.08	-
	DC-VAE [77]	-	17.90
Score	NCSN [3]	-	25.32
	Rec. Likelihood [40]	3.18	9.36
	DSM-ALS [39]	3.65	-
	DDPM [1]	3.75	3.17
	Improved DDPM [26]	2.94	11.47
	SDE (DDPM++) [2]	2.99	2.92
	SDE (NCSN++) [2]	-	2.20
Flows	VFlow [19]	2.98	-
	ANF [18]	3.05	-
Aut. Reg.	DistAug aug [78]	2.53	42.90
	Sp. Transformers [79]	2.80	-
	δ-VAE [80]	2.83	-
	PixelSNAIL [81]	2.85	-
	PixelCNN++ [82]	2.92	-
GANs	AutoGAN [83]	-	12.42
	StyleGAN2-ADA [84]	-	2.92

Table 3: Generative results on CelebA-HQ-256.

	Method	NLL↓	FID↓
Ours	LSGM	≤**0.70**	**7.22**
	VAE Backbone	**0.70**	30.87
VAEs	NVAE [20]	**0.70**	29.76
	VAEBM [76]	-	20.38
	NCP-VAE [56]	-	24.79
	DC-VAE [77]	-	15.80
Score	SDE [2]	-	**7.23**
Flows	GLOW [85]	1.03	68.93
Aut. Reg.	SPN [86]	0.61	-
GANs	Adv. LAE [87]	-	19.21
	VQ-GAN [64]	-	10.70
	PGGAN [88]	-	8.03

- LSGM obtains the SOTA FID score of 2.10 on CIFAR-10, outperforming previous GANs and SGMs.
- On CelebA-HQ-256, it is on a par with previous SGMs while being 50x to 600x faster in sampling.

Conclusion

- Score-based generative modeling
  - **No need to be normalized / invertible**
    - Flexible architecture choices
  - **No minimax optimization**
    - stable training
    - a natural measurement of training progress / model comparison

- **Adding noise** and **annealing the noise levels** are critical

- Better or comparable sample quality to GANs.

Related Work

- Generative Stochastic Networks (Bengio et al. (2013), Alain et al. (2016))
  - Sampling starts **close to data points.**
  - Need **MCMC during training** with walkback.

- Nonequilibrium Thermodynamics (Sohl-Dickstein et al. (2015)), Infusion Training (Bordes et al. (2017)), Variational Walkback (Goyal et al. (2017))
  - **Likelihood**-based training.
  - Need **MCMC during training.**

Experiments: Nearest Neighbors

Future Directions

- How to apply score-based generative modeling to discrete data?
- Theoretical guidance on how to choose noise levels?
- Better architecture for higher resolution image generation?
- Improved score estimation?

Lecture overview

- Energy-based models
- Score-based Models
- **Denoising Diffusion Models**
  - Overview
  - Denoising Diffusion Probabilistic Models
  - Diffusion Models Beat GANs on Image Synthesis; Classifier Guidance
  - Classifier-Free Diffusion Guidance
  - GLIDE: Text-Guided Diffusion Models
  - Dall-E: 1, 2, 3
  - Imagen
- Deeper Dive into Diffusion Models

Lecture overview

- Motivation
- Energy-based models
- Score-based Models
- **Denoising Diffusion Models**
  - **Overview**
  - Denoising Diffusion Probabilistic Models
  - Diffusion Models Beat GANs on Image Synthesis; Classifier Guidance
  - Classifier-Free Diffusion Guidance
  - GLIDE: Text-Guided Diffusion Models
  - Dall-E: 1, 2, 3
  - Imagen
- Deeper Dive into Diffusion Models

Observation 1: Diffusion Destroys Structure

- Dye density represents probability density
- **Goal:** Learn structure probability density
- **Observation:** Diffusion destroys structure

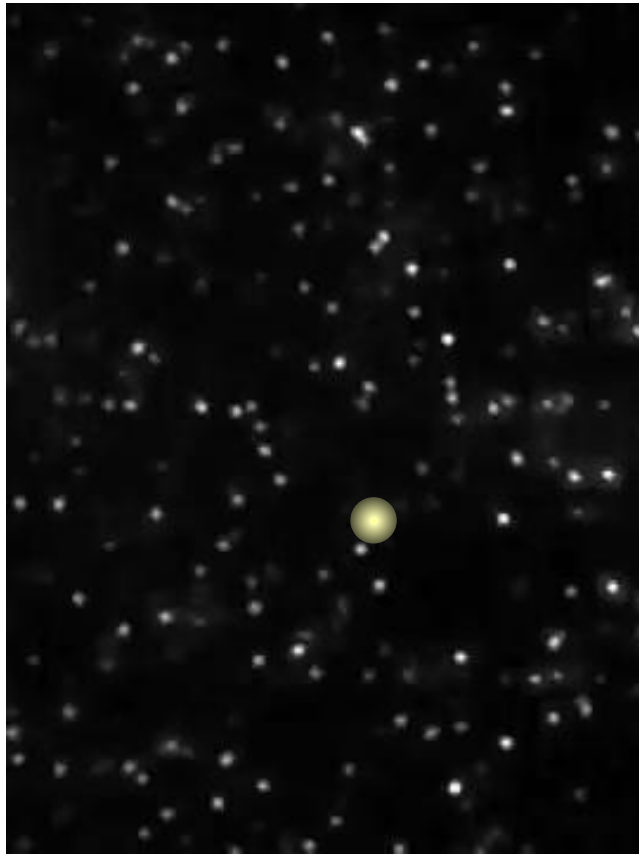
Data distribution → Uniform distribution

Idea: Recover Structure by Reversing Time

- What if we could reverse time?
- Recover data distribution by starting from uniform distribution and running dynamics backwards

Data distribution ← Uniform distribution

Observation 2: Microscopic Diffusion is Time Reversible



Nanoparticles in water

- Microscopic view
- Brownian motion
- Position updates are small Gaussians
  - Both forwards and backwards in time

Video credit: Rutger Saly

Overview of Diffusion Probabilistic Models

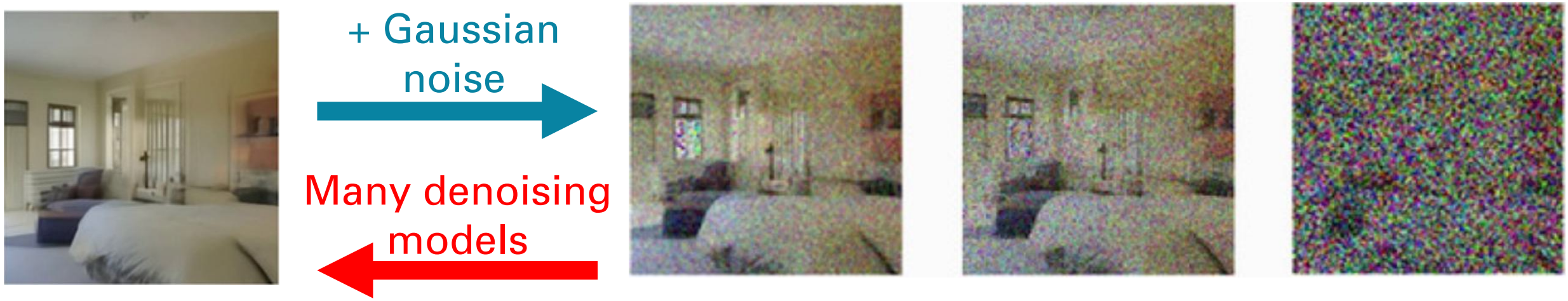
+ Gaussian noise

Denoising model

Clean

Noisy

Diffusion Probabilistic Models



Overview of Diffusion Probabilistic Models

- Destroy all structure in data distribution using diffusion process
- Learn reversal of diffusion process
  - Estimate function for mean and covariance of each step in the reverse diffusion process (binomial rate for binary data)

Lecture overview

- Motivation
- Energy-based models
- Score-based Models
- **Denoising Diffusion Models**
  - Overview
  - **Denoising Diffusion Probabilistic Models**
  - Diffusion Models Beat GANs on Image Synthesis; Classifier Guidance
  - Classifier-Free Diffusion Guidance
  - GLIDE: Text-Guided Diffusion Models
  - Dall-E: 1, 2, 3
  - Imagen
- Deeper Dive into Diffusion Models

State of Affairs of this class + field anno 2021

- Autoregressive models
  - MADE, PixelRNN/CNN, Gated PixelCNN, PixelSNAIL
- Flow models
  - Autoregressive Flows, NICE, RealNVP, Glow, Flow++
- Latent Variable Models
  - VAE, IWAE, VQ-VAE
- GANs

GANs gave the most realistic images, but have lots of bells and whistles + struggle with truly covering what's in data distribution

→ Diffusion Models

Denoising Diffusion Probabilistic Models

Denoising Diffusion Probabilistic Models

Jonathan Ho
UC Berkeley

Ajay Jain
UC Berkeley

Pieter Abbeel
UC Berkeley

[Ho, Jain, Abbeel, 2020]

Denoising Diffusion Probabilistic Models

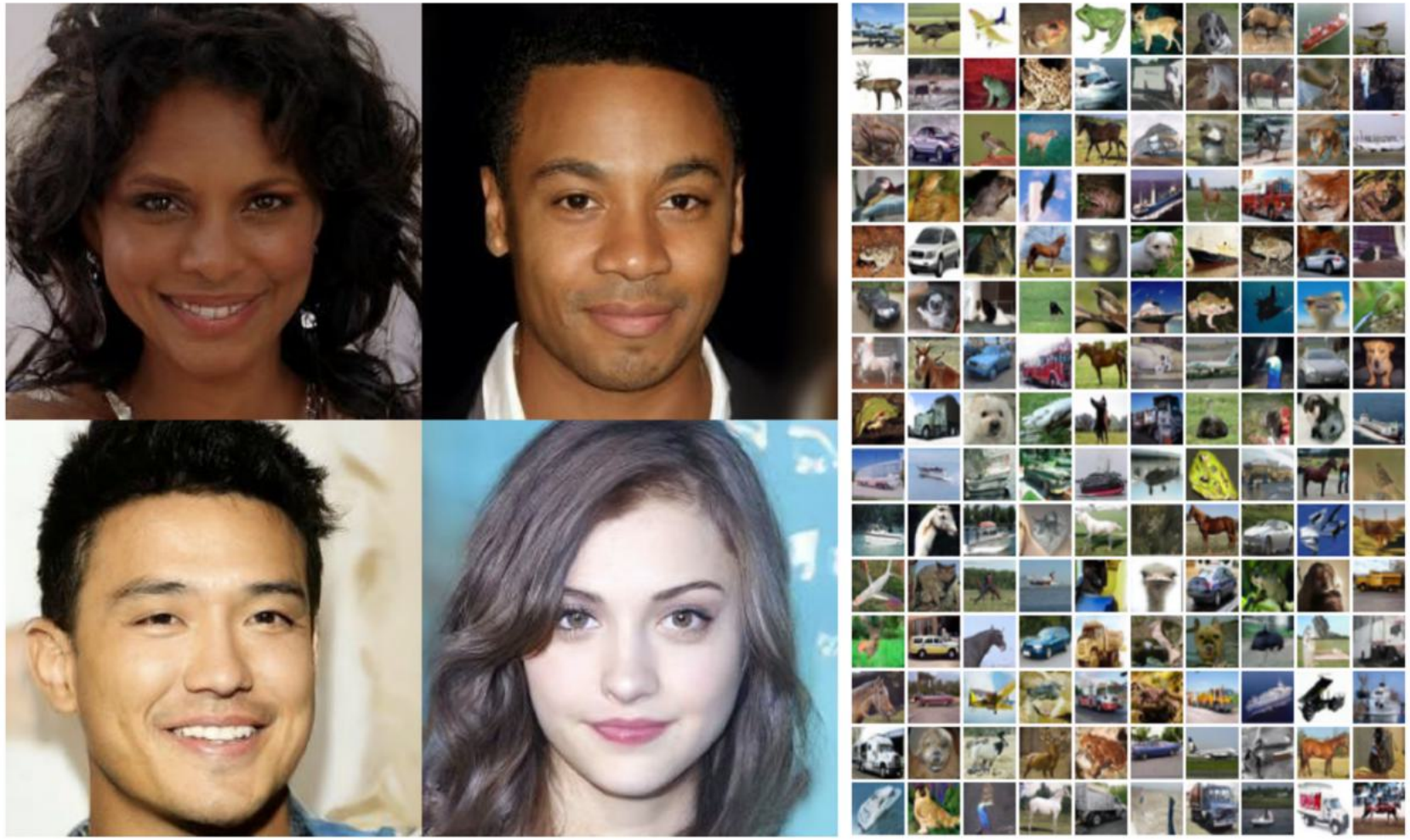
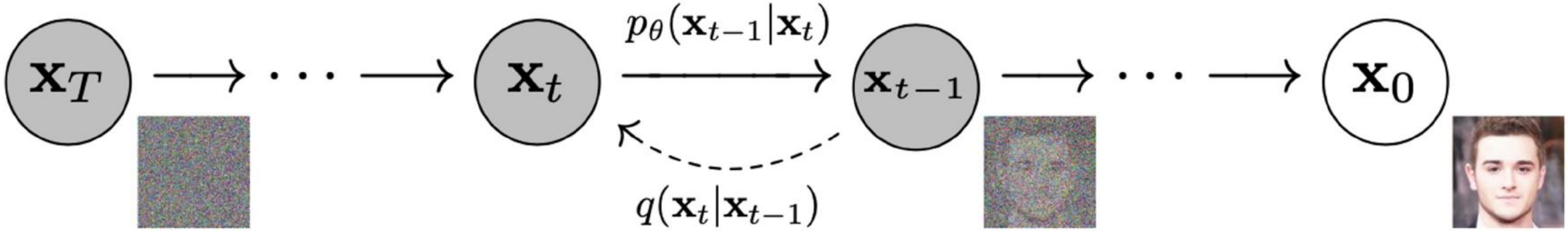


Figure 1: Generated samples on CelebA-HQ 256 × 256 (left) and unconditional CIFAR10 (right)

[Ho, Jain, Abbeel, 2020]

Denoising Diffusion Probabilistic Models

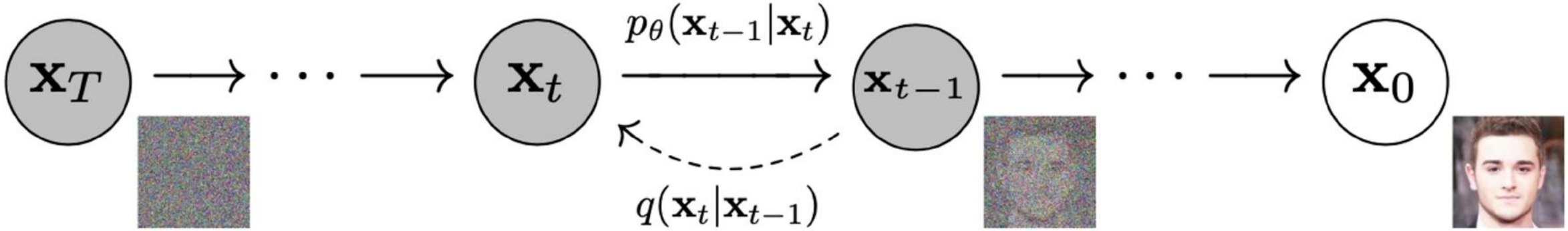


$$p_\theta(\mathbf{x}_{0:T}) := p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t), \qquad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)) \quad (1)$$

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) := \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}), \qquad q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}) \quad (2)$$

[Ho, Jain, Abbeel, 2020]

Denoising Diffusion Probabilistic Models



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

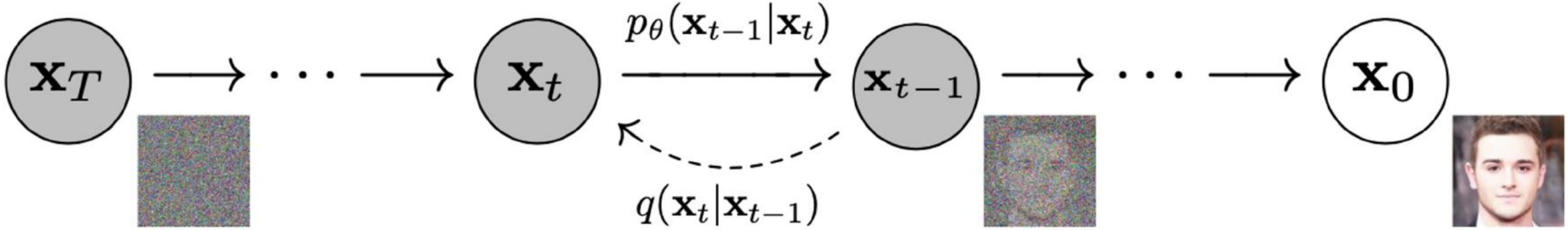
Algorithm 2 Sampling

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]

Denoising Diffusion Probabilistic Models



Training is performed by optimizing the usual variational bound on negative log likelihood:

$$\mathbb{E}\left[-\log p_\theta(\mathbf{x}_0)\right] \leq \mathbb{E}_q\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] = \mathbb{E}_q\left[-\log p(\mathbf{x}_T) - \sum_{t\geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})}\right] =: L \quad (3)$$

Variance reduction version:

$$\mathbb{E}_q\Big[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p(\mathbf{x}_T))}_{L_T} + \sum_{t>1} \underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{-\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0}\Big] \quad (5)$$

[Ho, Jain, Abbeel, 2020]

Architecture

- Backbone of PixelCNN++ , which is a U-Net based on a Wide ResNet.
- Replaced weight normalization with group normalization to make implementation simpler.
- The 32×32 models use four feature map resolutions (32×32 to 4×4), and the 256×256 models use six.
- All models have two convolutional residual blocks per resolution level and self-attention blocks at the 16×16 resolution between the convolutional blocks.
- Diffusion time t is specified by adding the Transformer sinusoidal position embedding into each residual block.
- The CIFAR10 model has 35.7 million parameters
- The LSUN and CelebA-HQ models have 114 million parameters. Also trained a larger variant of the LSUN Bedroom model with approximately 256 million parameters by increasing filter count.

[Ho, Jain, Abbeel, 2020]

Denoising Diffusion Probabilistic Models

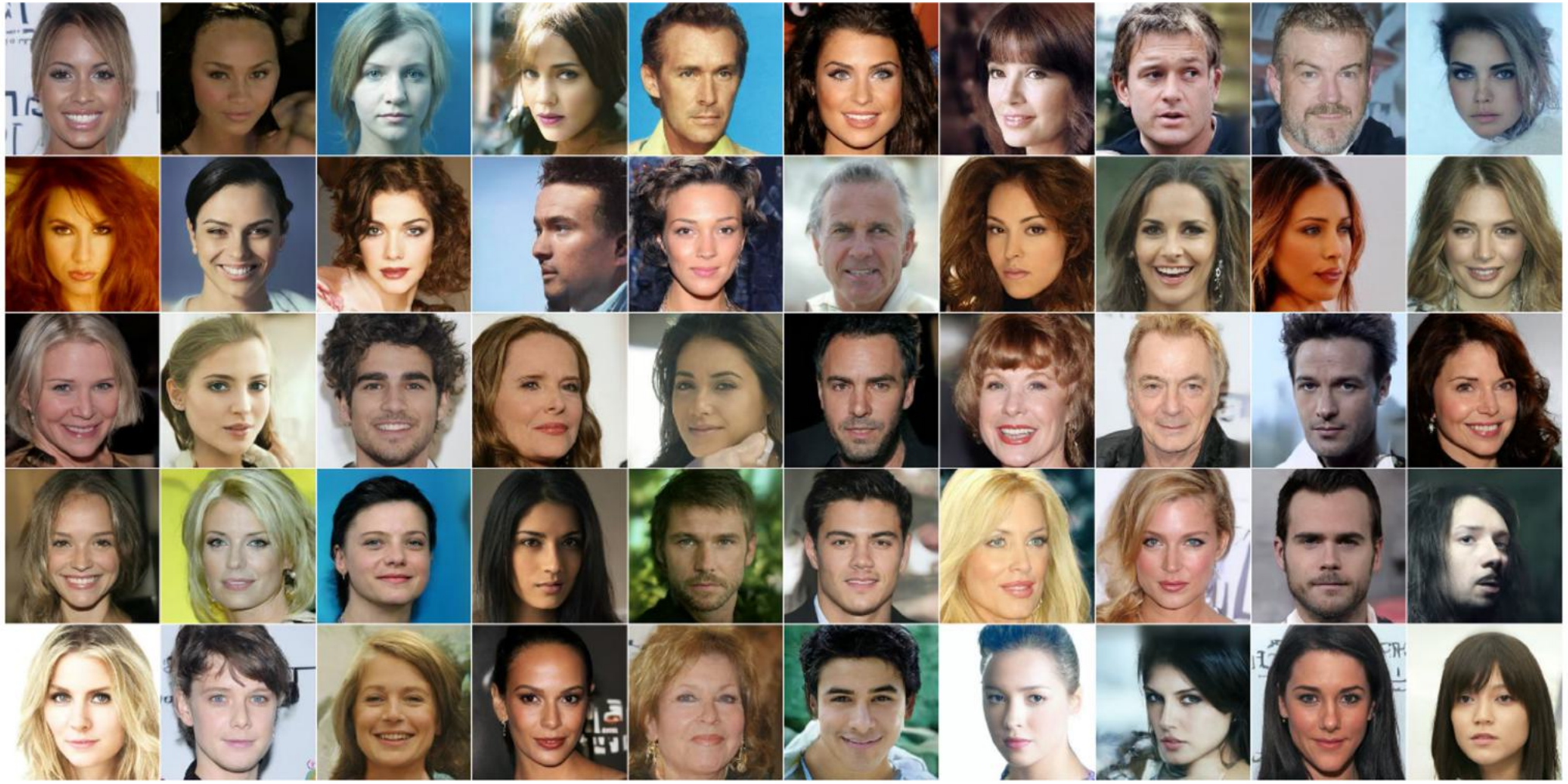


Figure 11: CelebA-HQ 256 × 256 generated samples

[Ho, Jain, Abbeel, 2020]

Denoising Diffusion Probabilistic Models

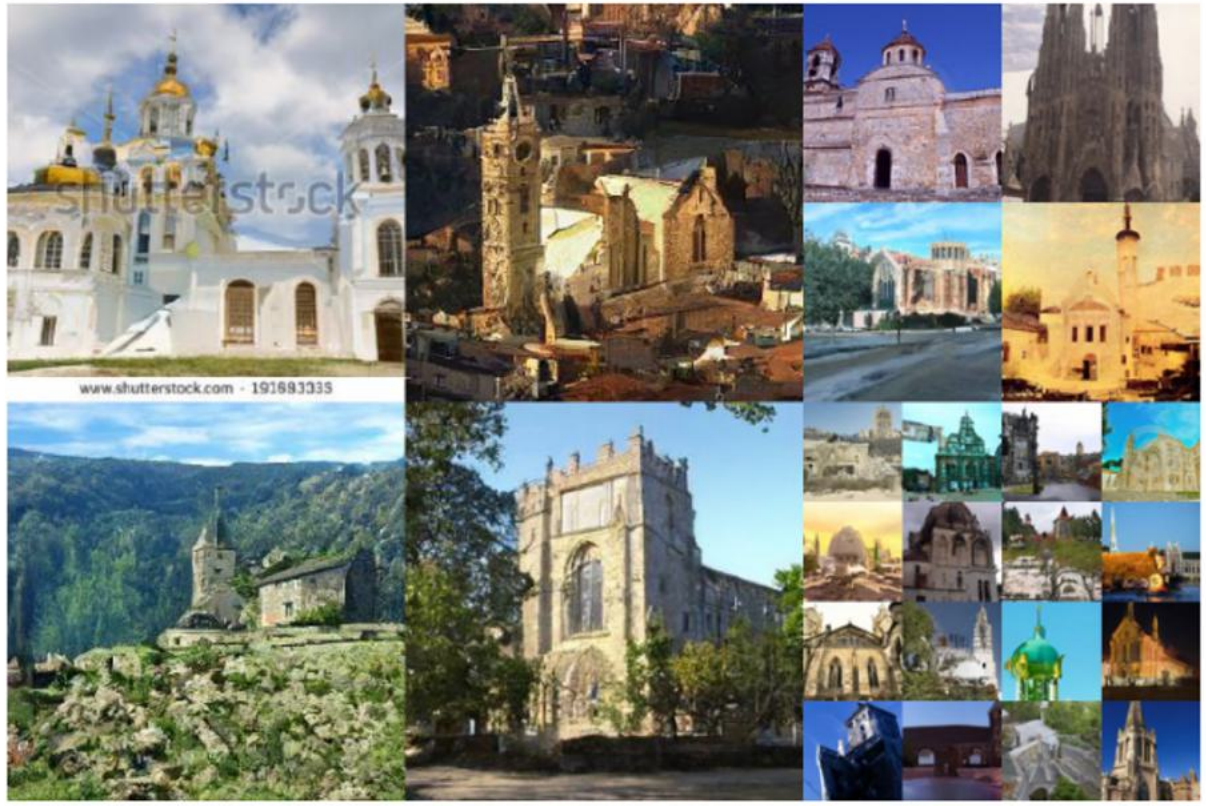


Figure 3: LSUN Church samples. FID=7.89

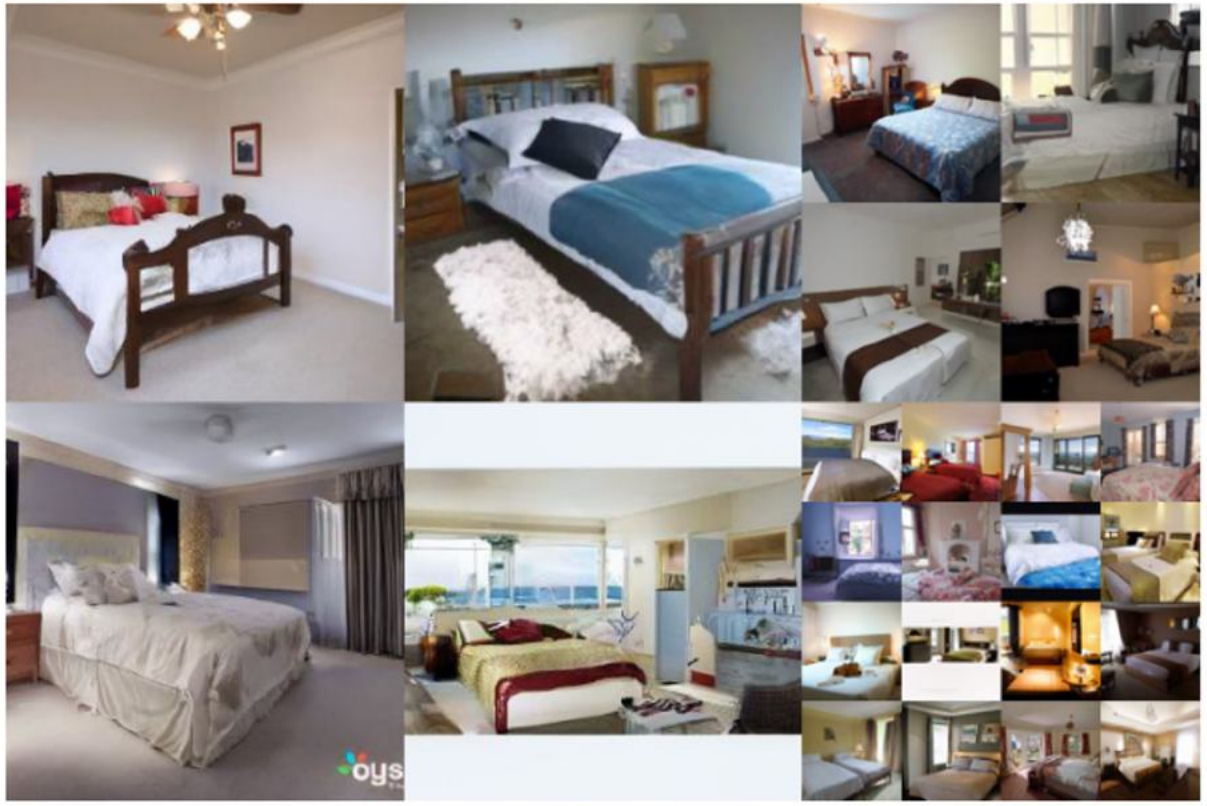


Figure 4: LSUN Bedroom samples. FID=4.90

[Ho, Jain, Abbeel, 2020]

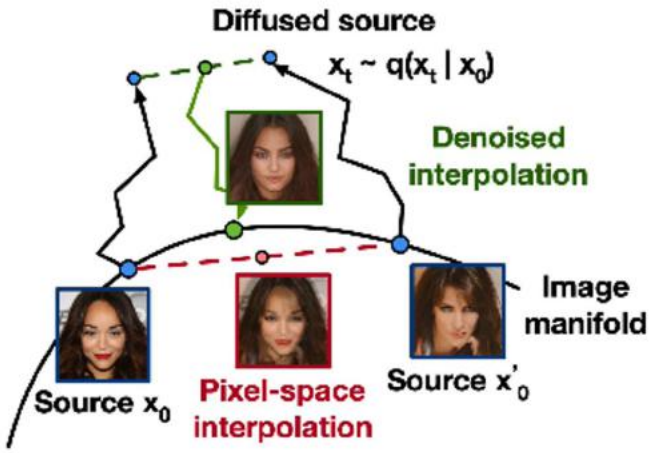
Denoising Diffusion Probabilistic Models

Table 1: CIFAR10 results. NLL measured in bits/dim.

Model	IS	FID	NLL Test (Train)
Conditional			
EBM [11]	8.30	37.9	
JEM [17]	8.76	38.4	
BigGAN [3]	9.22	14.73	
StyleGAN2 + ADA (v1) [29]	**10.06**	**2.67**	
Unconditional			
Diffusion (original) [53]			≤ 5.40
Gated PixelCNN [59]	4.60	65.93	3.03 (2.90)
Sparse Transformer [7]			**2.80**
PixelIQN [43]	5.29	49.46	
EBM [11]	6.78	38.2	
NCSNv2 [56]		31.75	
NCSN [55]	8.87 ± 0.12	25.32	
SNGAN [39]	8.22 ± 0.05	21.7	
SNGAN-DDLS [4]	9.09 ± 0.10	15.42	
StyleGAN2 + ADA (v1) [29]	$\mathbf{9.74} \pm 0.05$	3.26	
Ours (L, fixed isotropic $\mathbf{\Sigma}$)	7.67 ± 0.13	13.51	≤ 3.70 (3.69)
Ours (L_{simple})	9.46 ± 0.11	**3.17**	≤ 3.75 (3.72)

[Ho, Jain, Abbeel, 2020]

Denoising Diffusion Probabilistic Models



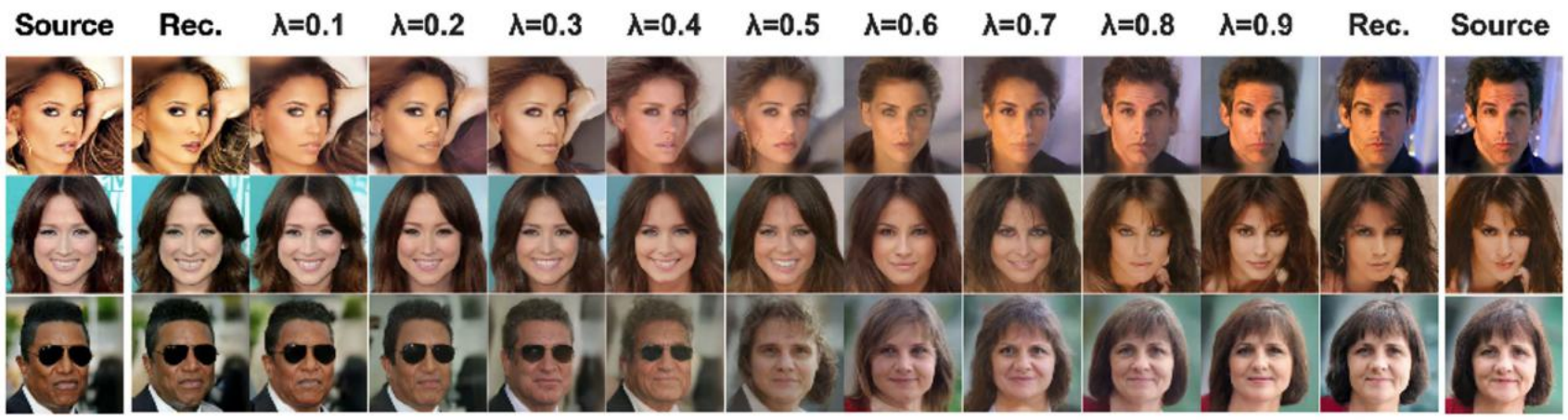


Figure 8: Interpolations of CelebA-HQ 256x256 images with 500 timesteps of diffusion.

[Ho, Jain, Abbeel, 2020]

Denoising Diffusion Probabilistic Models

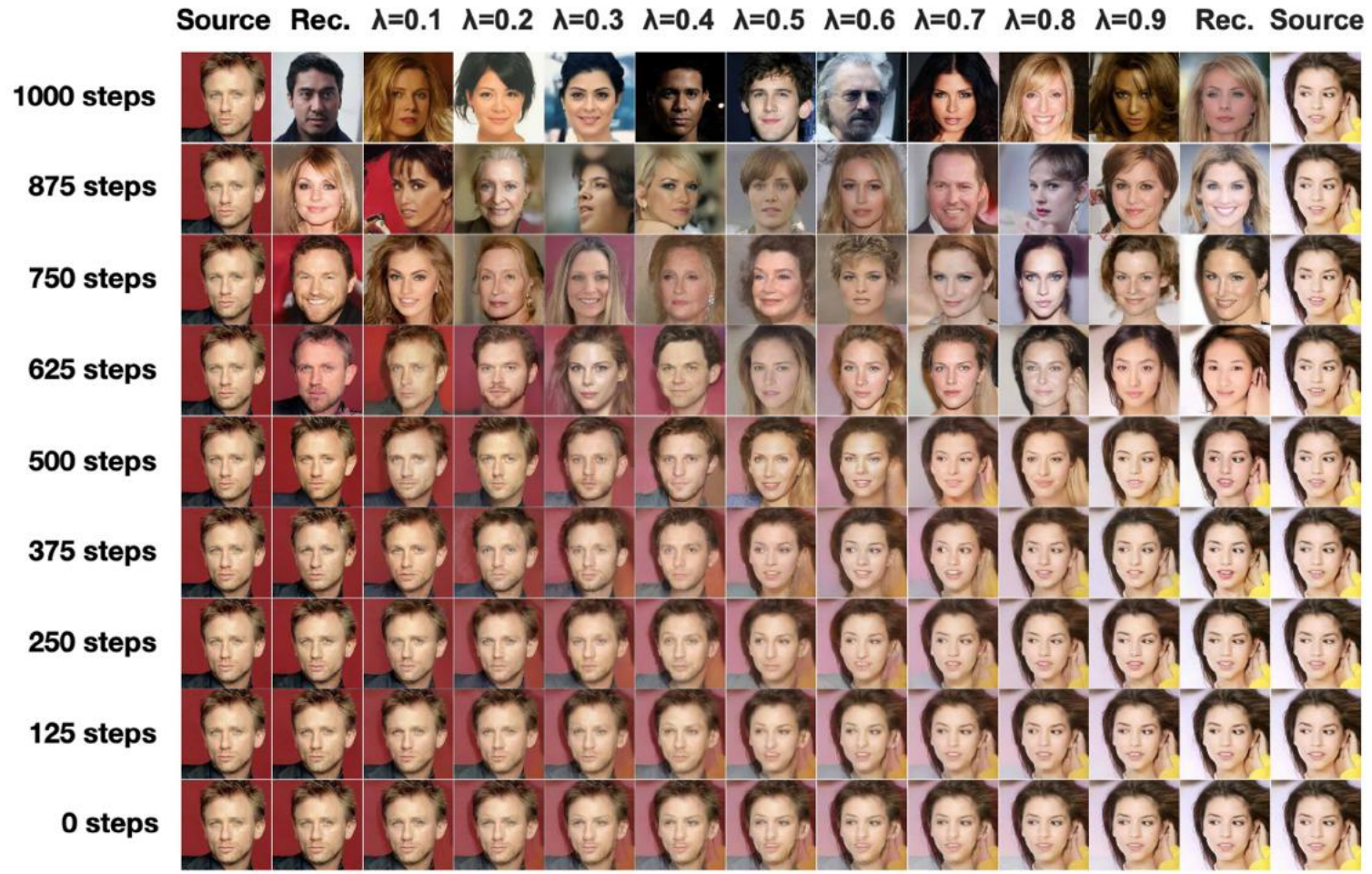


Figure 9: Coarse-to-fine interpolations that vary the number of diffusion steps prior to latent mixing.

[Ho, Jain, Abbeel, 2020]

Denoising Diffusion Probabilistic Models

Connection to autoregressive decoding Note that the variational bound (5) can be rewritten as:

$$L = D_{\mathrm{KL}}(q(\mathbf{x}_T) \parallel p(\mathbf{x}_T)) + \mathbb{E}_q \left[\sum_{t \geq 1} D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) \right] + H(\mathbf{x}_0) \tag{16}$$

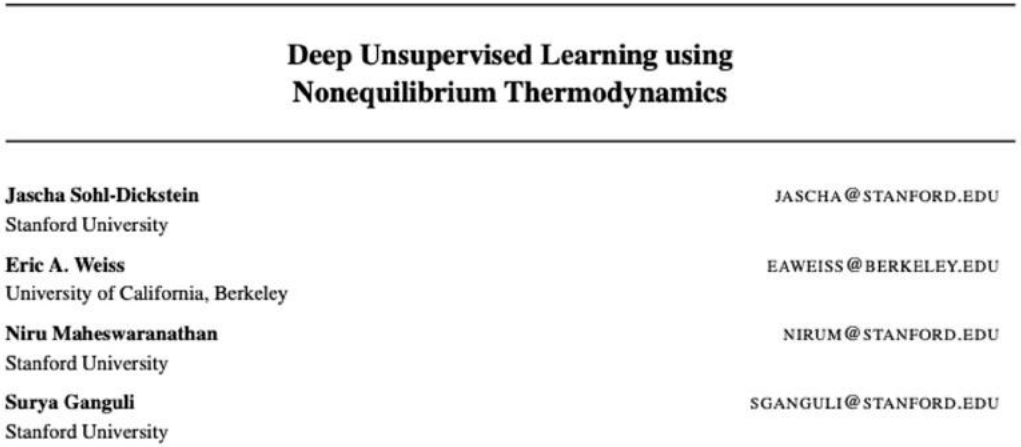
(See Appendix A for a derivation.) Now consider setting the diffusion process length T to the dimensionality of the data, defining the forward process so that $q(\mathbf{x}_t|\mathbf{x}_0)$ places all probability mass on $\mathbf{x}_0$ with the first t coordinates masked out (i.e. $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ masks out the t^{th} coordinate), setting $p(\mathbf{x}_T)$ to place all mass on a blank image, and, for the sake of argument, taking $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ to be a fully expressive conditional distribution. With these choices, $D_{\mathrm{KL}}(q(\mathbf{x}_T) \parallel p(\mathbf{x}_T)) = 0$, and minimizing $D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))$ trains p_θ to copy coordinates $t+1, \ldots, T$ unchanged and to predict the t^{th} coordinate given $t+1, \ldots, T$. Thus, training p_θ with this particular diffusion is training an autoregressive model.

We can therefore interpret the Gaussian diffusion model (2) as a kind of autoregressive model with a generalized bit ordering that cannot be expressed by reordering data coordinates. Prior work has shown that such reorderings introduce inductive biases that have an impact on sample quality [38], so we speculate that the Gaussian diffusion serves a similar purpose, perhaps to greater effect since Gaussian noise might be more natural to add to images compared to masking noise. Moreover, the Gaussian diffusion length is not restricted to equal the data dimension; for instance, we use $T = 1000$, which is less than the dimension of the $32 \times 32 \times 3$ or $256 \times 256 \times 3$ images in our experiments. Gaussian diffusions can be made shorter for fast sampling or longer for model expressiveness.

[Ho, Jain, Abbeel, 2020]

Key Inspiration Works:

[Sohl-Dickstein et al, ICML 2015]



Deep Unsupervised Learning using Nonequilibrium Thermodynamics

Jascha Sohl-Dickstein JASCHA@STANFORD.EDU
Stanford University

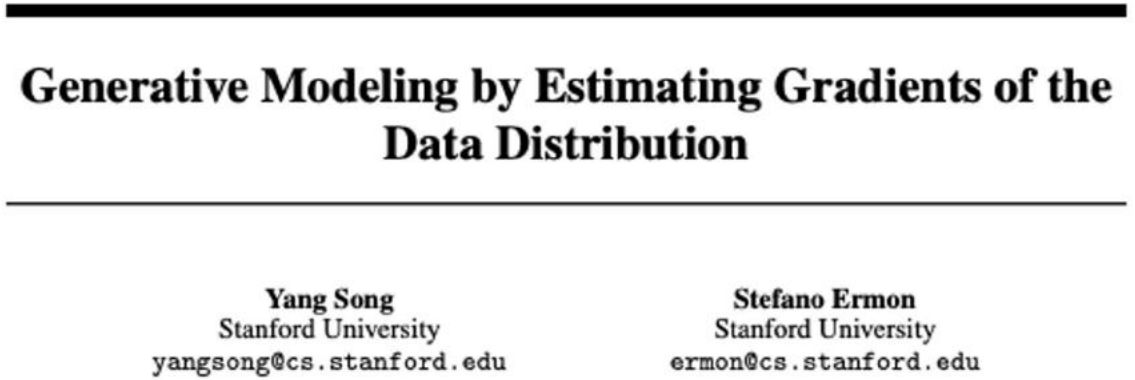
Eric A. Weiss EAWEISS@BERKELEY.EDU
University of California, Berkeley

Niru Maheswaranathan NIRUM@STANFORD.EDU
Stanford University

Surya Ganguli SGANGULI@STANFORD.EDU
Stanford University

- Introduced Diffusion Models

[Song & Ermon, NeurIPS 2019]



Generative Modeling by Estimating Gradients of the Data Distribution

Yang Song
Stanford University
yangsong@cs.stanford.edu

Stefano Ermon
Stanford University
ermon@cs.stanford.edu

- Annealed Langevin Dynamics for sampling against energy model with score function (grad log p(x))
- Very promising sample generation

Lecture overview

- Motivation
- Energy-based models
- Score-based Models
- **Denoising Diffusion Models**
  - Overview
  - Denoising Diffusion Probabilistic Models
  - **Diffusion Models Beat GANs on Image Synthesis; Classifier Guidance**
  - Classifier-Free Diffusion Guidance
  - GLIDE: Text-Guided Diffusion Models
  - Dall-E: 1, 2, 3
  - Imagen
- Deeper Dive into Diffusion Models

Diffusion Models Beat GANs on Image Synthesis

Diffusion Models Beat GANs on Image Synthesis

Prafulla Dhariwal*
OpenAI
prafulla@openai.com

Alex Nichol*
OpenAI
alex@openai.com

Abstract

We show that diffusion models can achieve image sample quality superior to the current state-of-the-art generative models. We achieve this on unconditional image synthesis by finding a better architecture through a series of ablations. For conditional image synthesis, we further improve sample quality with classifier guidance: a simple, compute-efficient method for trading off diversity for fidelity using gradients from a classifier. We achieve an FID of 2.97 on ImageNet 128×128, 4.59 on ImageNet 256×256, and 7.72 on ImageNet 512×512, and we match BigGAN-deep even with as few as 25 forward passes per sample, all while maintaining better coverage of the distribution. Finally, we find that classifier guidance combines well with upsampling diffusion models, further improving FID to 3.94 on ImageNet 256×256 and 3.85 on ImageNet 512×512. We release our code at https://github.com/openai/guided-diffusion.

GANs [19] currently hold the state-of-the-art on most image generation tasks [5, 68, 28] as measured by sample quality metrics such as FID [23], Inception Score [54] and Precision [32]. However, some of these metrics do not fully capture diversity, and it has been shown that GANs capture less diversity than state-of-the-art likelihood-based models [51, 43, 42]. Furthermore, GANs are often difficult to train, collapsing without carefully selected hyperparameters and regularizers [5, 41, 4].

[Dhariwal & Nichol, 2021]

Diffusion Models Beat GANs on Image Synthesis

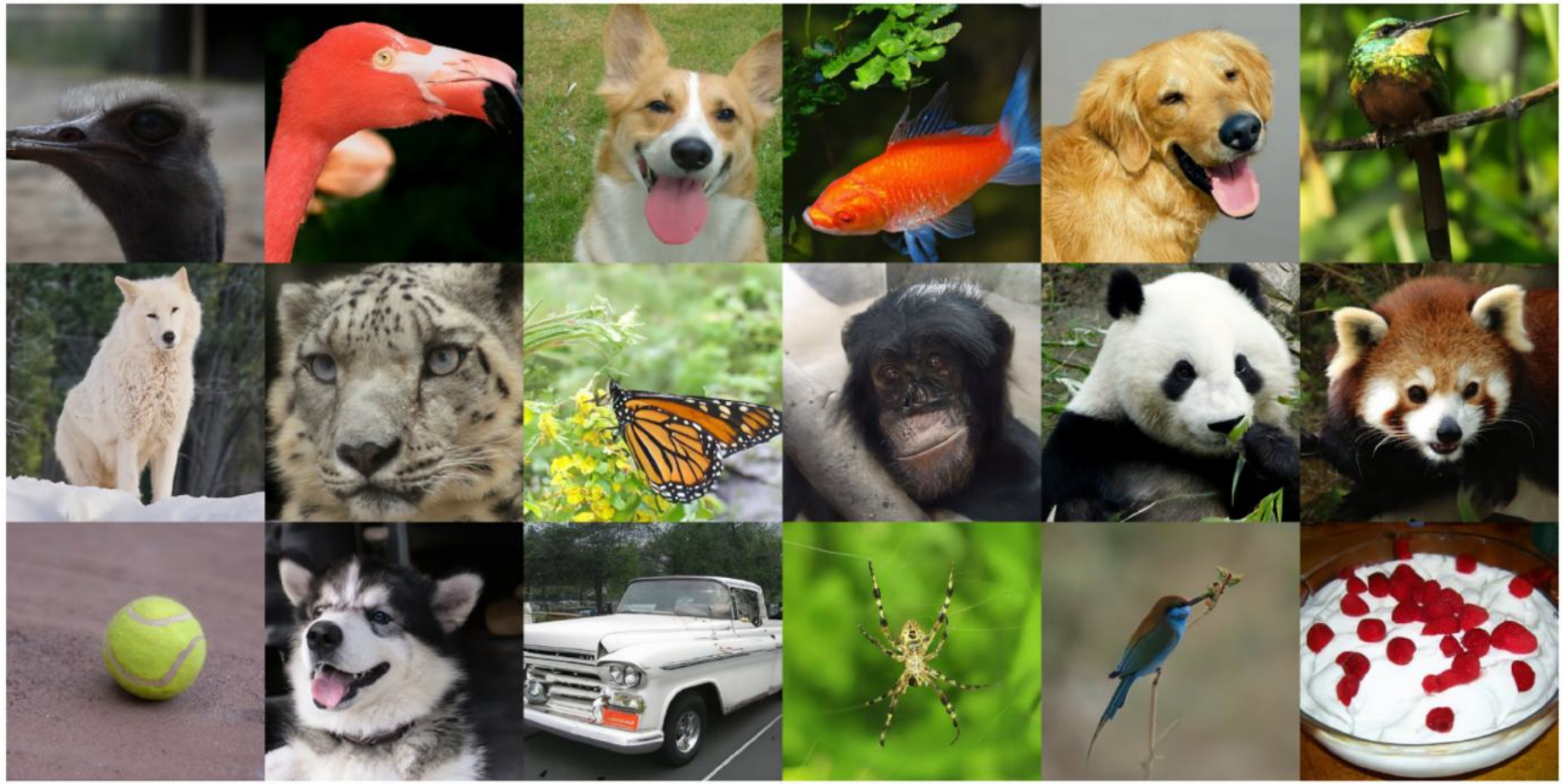


Figure 1: Selected samples from our best ImageNet 512×512 model (FID 3.85)

[Dhariwal & Nichol, 2021]

Algorithmic Improvements

Learned variance

Nichol and Dhariwal [43] find that fixing the variance $\Sigma_\theta(x_t, t)$ to a constant as done in Ho et al. [25] is sub-optimal for sampling with fewer diffusion steps, and propose to parameterize $\Sigma_\theta(x_t, t)$ as a neural network whose output v is interpolated as:

$$\Sigma_\theta(x_t, t) = \exp(v \log \beta_t + (1 - v) \log \tilde{\beta}_t) \tag{1}$$

Deterministic Sampler

Song et al. [57] propose DDIM, which formulates an alternative non-Markovian noising process that has the same forward marginals as DDPM, but allows producing different reverse samplers by changing the variance of the reverse noise. By setting this noise to 0, they provide a way to turn any model $\epsilon_\theta(x_t, t)$ into a deterministic mapping from latents to images, and find that this provides an alternative way to sample with fewer steps. We adopt this sampling approach when using fewer than 50 sampling steps, since Nichol and Dhariwal [43] found it to be beneficial in this regime.

Architecture Improvements

- Increasing depth versus width, holding model size relatively constant.
- Increasing the number of attention heads.
- Using attention at 32×32, 16×16, and 8×8 resolutions rather than only at 16×16.
- Using the BigGAN [5] residual block for upsampling and downsampling the activations, following [60].
- Rescaling residual connections with $\frac{1}{\sqrt{2}}$, following [60, 27, 28].

Classifier Guidance

Key idea: add an additional term in the sampling process to direct towards the desired class

Train a classifier $p_\phi(y|x_t, t)$ on noisy images x_t

Use gradients $\nabla_{x_t} \log p_\phi(y|x_t, t)$ to guide towards a class label

$$x_{t-1} = w_0 x_t + w_1 \epsilon_\theta(x_t, t) + w_3 \nabla_{x_t} \log p_\phi(y|x_t, t) \qquad \epsilon_\theta \approx \sigma \nabla_{x_t} \log p_\theta(x_t)$$

Similar Image Quality + Better Diversity than BigGAN

Figure 6: Samples from BigGAN-deep with truncation 1.0 (FID 6.95, left) vs samples from our diffusion model with guidance (FID 4.59, middle) and samples from the training set (right).

Lecture overview

- Motivation
- Energy-based models
- Score-based Models
- **Denoising Diffusion Models**
  - Overview
  - Denoising Diffusion Probabilistic Models
  - Diffusion Models Beat GANs on Image Synthesis; Classifier Guidance
  - **Classifier-Free Diffusion Guidance**
  - GLIDE: Text-Guided Diffusion Models
  - Dall-E: 1, 2, 3
  - Imagen
- Deeper Dive into Diffusion Models

CLASSIFIER-FREE DIFFUSION GUIDANCE

Jonathan Ho & Tim Salimans
Google Research, Brain team
{jonathanho,salimans}@google.com

ABSTRACT

Classifier guidance is a recently introduced method to trade off mode coverage and sample fidelity in conditional diffusion models post training, in the same spirit as low temperature sampling or truncation in other types of generative models. Classifier guidance combines the score estimate of a diffusion model with the gradient of an image classifier and thereby requires training an image classifier separate from the diffusion model. It also raises the question of whether guidance can be performed without a classifier. We show that guidance can be indeed performed by a pure generative model without such a classifier: in what we call classifier-free guidance, we jointly train a conditional and an unconditional diffusion model, and we combine the resulting conditional and unconditional score estimates to attain a trade-off between sample quality and diversity similar to that obtained using classifier guidance.

[Ho & Salimans, 2022]

Classifier-Free Guidance

- Motivation
  - Classifier-guidance enabled “low temperature” / “sharper” samples
  - BUT: Can we achieve this without an additional classifier to train, purely in the generative model itself?
- Key Idea, consider the extra term in classifier free guidance sampling + apply Bayes rule:

$$
\begin{aligned}
& +w\nabla_x \log p(y|x) && \text{<= classifier-guidance} \\
= \quad & +w\nabla_x \log \frac{p(x|y)p(y)}{p(x)} && \\
= \quad & +w\nabla_x \log p(x|y) + w\nabla_x p(y) - w\nabla_x \log p(x) && \\
= \quad & +w\nabla_x \log p(x|y) + 0 - w\nabla_x \log p(x) && \text{<= classifier-\textbf{free} guidance} \\
= \quad & w\epsilon_\theta(x, y) - w\epsilon_\theta(x) && \text{<= in DDPM notation}
\end{aligned}
$$

No need for classifier, just need to train both conditional and unconditional generator

Sampling with Classifier-Free Guidance

Algorithm 2 Conditional sampling with classifier-free guidance

Require: w: guidance strength
Require: $\mathbf{c}$: conditioning information for conditional sampling
Require: $\lambda_1, \ldots, \lambda_T$: increasing log SNR sequence with $\lambda_1 = \lambda_{\min}, \lambda_T = \lambda_{\max}$

1: $\mathbf{z}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = 1, \ldots, T$ **do**
▷ Form the classifier-free guided score at log SNR λ_t
3: $\tilde{\boldsymbol{\epsilon}}_t = (1 + w)\boldsymbol{\epsilon}_\theta(\mathbf{z}_t, \mathbf{c}) - w\boldsymbol{\epsilon}_\theta(\mathbf{z}_t)$
▷ Sampling step (could be replaced by another sampler, e.g. DDIM)
4: $\tilde{\mathbf{x}}_t = (\mathbf{z}_t - \sigma_{\lambda_t}\tilde{\boldsymbol{\epsilon}}_t)/\alpha_{\lambda_t}$
5: $\mathbf{z}_{t+1} \sim \mathcal{N}(\tilde{\boldsymbol{\mu}}_{\lambda_{t+1}|\lambda_t}(\mathbf{z}_t, \tilde{\mathbf{x}}_t), (\tilde{\sigma}^2_{\lambda_{t+1}|\lambda_t})^{1-v}(\sigma^2_{\lambda_t|\lambda_{t+1}})^v)$ if $t < T$ else $\mathbf{z}_{t+1} = \tilde{\mathbf{x}}_t$
6: **end for**
7: **return** $\mathbf{z}_{T+1}$

[Ho & Salimans, 2022]

Training with Classifier-Free Guidance

Algorithm 1 Joint training a diffusion model with classifier-free guidance

Require: p_{uncond}: probability of unconditional training

1: **repeat**
2: $(\mathbf{x}, \mathbf{c}) \sim p(\mathbf{x}, \mathbf{c})$ ▷ Sample data with conditioning from the dataset
3: $\mathbf{c} \leftarrow \varnothing$ with probability p_{uncond} ▷ Randomly discard conditioning to train unconditionally
4: $\lambda \sim p(\lambda)$ ▷ Sample log SNR value
5: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
6: $\mathbf{z}_\lambda = \alpha_\lambda \mathbf{x} + \sigma_\lambda \boldsymbol{\epsilon}$ ▷ Corrupt data to the sampled log SNR value
7: Take gradient step on $\nabla_\theta \left\| \boldsymbol{\epsilon}_\theta(\mathbf{z}_\lambda, \mathbf{c}) - \boldsymbol{\epsilon} \right\|^2$ ▷ Optimization of denoising model
8: **until** converged

[Ho & Salimans, 2022]

Effect of Classifier-Free Guidance

Figure 2: The effect of guidance on a mixture of three Gaussians, each mixture component representing data conditioned on a class. The leftmost plot is the non-guided marginal density. Left to right are densities of mixtures of normalized guided conditionals with increasing guidance strength.

[Ho & Salimans, 2022]

Effect of Classifier-Free Guidance

Figure 1: Classifier-free guidance on the malamute class for a 64x64 ImageNet diffusion model. Left to right: increasing amounts of classifier-free guidance, starting from non-guided samples on the left.

[Ho & Salimans, 2022]

Classifier-Free Guidance

Figure 3: Classifier-free guidance on 128x128 ImageNet. Left: non-guided samples, right: classifier-free guided samples with $w = 3.0$. Interestingly, strongly guided samples such as these display saturated colors. See Fig. 8 for more.

[Ho & Salimans, 2022]

FID / IS over training run

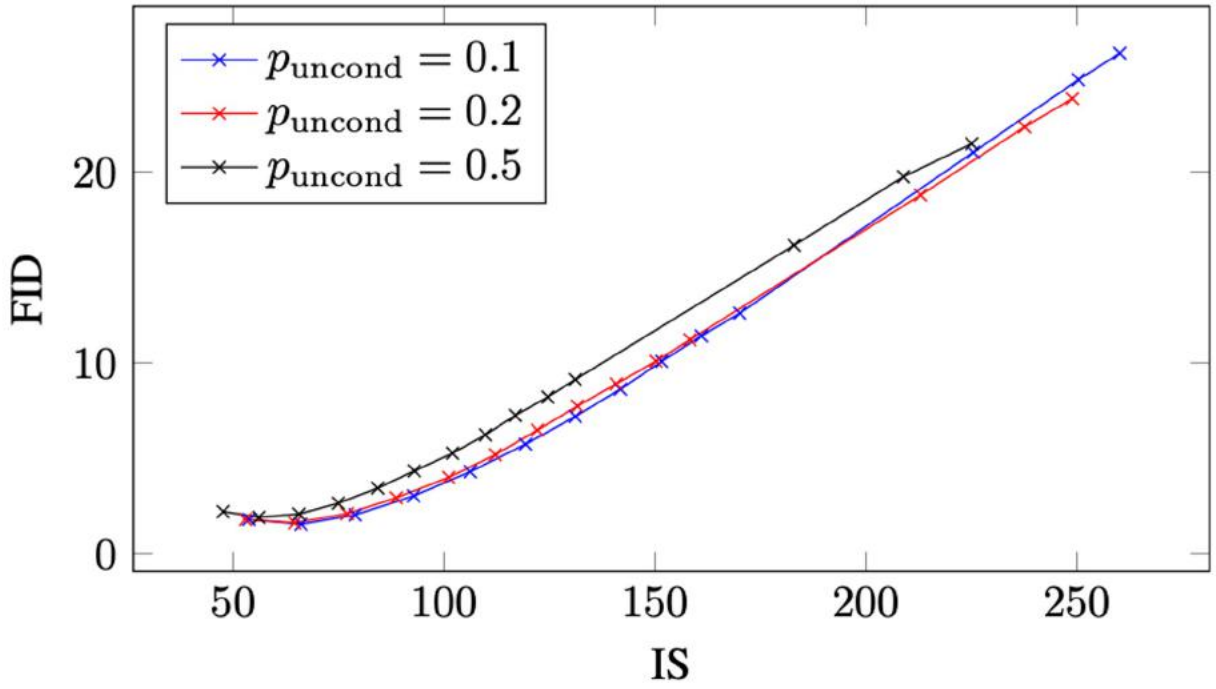


Figure 4: IS/FID curves over guidance strengths for ImageNet 64x64 models. Each curve represents a model with unconditional training probability p_{uncond}. Accompanies Table 1.

IS keeps getting better – i.e. more and more "realistic" member of a class
FID briefly improves, then keeps getting worse – i.e. lose diversity

[Ho & Salimans, 2022]

Classifier-Free Guidance

- Achieves guidance in a very simple way;
  - One-line change of code during training (randomly drop out conditioning)
  - During sampling just mix conditional and unconditional score estimates

Lecture overview

- Motivation
- Energy-based models
- Score-based Models
- **Denoising Diffusion Models**
  - Overview
  - Denoising Diffusion Probabilistic Models
  - Diffusion Models Beat GANs on Image Synthesis; Classifier Guidance
  - Classifier-Free Diffusion Guidance
  - **GLIDE: Text-Guided Diffusion Models**
  - Dall-E: 1, 2, 3
  - Imagen
- Deeper Dive into Diffusion Models

GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models

Alex Nichol * Prafulla Dhariwal * Aditya Ramesh * Pranav Shyam Pamela Mishkin Bob McGrew
Ilya Sutskever Mark Chen

[Nichol*, Dhariwal*, Ramesh* et al, 2022]

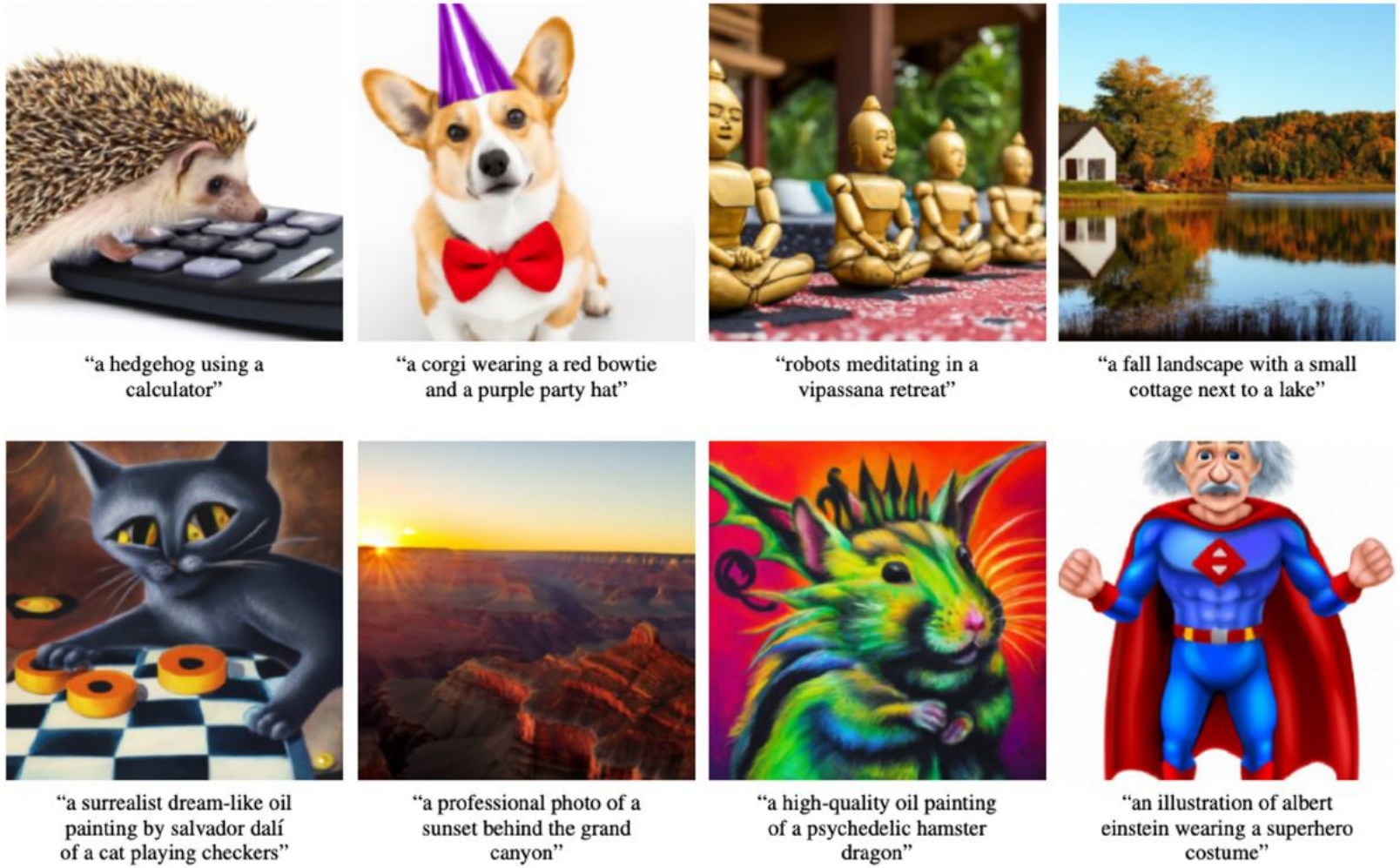
GLIDE

Main Idea:

- 3.5 Billion parameter text-conditional diffusion model for 64x64
- 1.5 Billion parameter text-conditional diffusion model for super-resolution to 256x256
- Classifier-Free Diffusion Guidance with caption conditioning (rather than class conditioning)
- Also tried CLIP-Guided Diffusion but that didn't work as well

[Nichol*, Dhariwal*, Ramesh* et al, 2022]

GLIDE



[Nichol*, Dhariwal*, Ramesh* et al, 2022]

GLIDE

Figure 2. Text-conditional image inpainting examples from GLIDE. The green region is erased, and the model fills it in conditioned on the given prompt. Our model is able to match the style and lighting of the surrounding context to produce a realistic completion.

[Nichol*, Dhariwal*, Ramesh* et al, 2022]

GLIDE

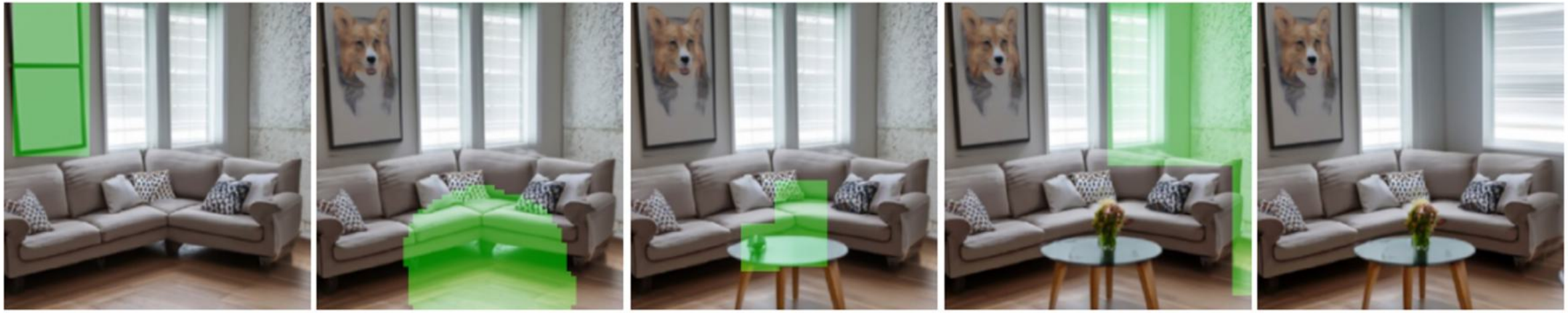


Figure 3. Iteratively creating a complex scene using GLIDE. First, we generate an image for the prompt "a cozy living room", then use the shown inpainting masks and follow-up text prompts to add a painting to the wall, a coffee table, and a vase of flowers on the coffee table, and finally to move the wall up to the couch.

[Nichol*, Dhariwal*, Ramesh* et al, 2022]

GLIDE

- Classifier-Free Diffusion Guidance with caption conditioning (rather than class conditioning)

$$\hat{\epsilon}_\theta(x_t|c) = \epsilon_\theta(x_t|\emptyset) + s \cdot (\epsilon_\theta(x_t|c) - \epsilon_\theta(x_t|\emptyset))$$

- Also tried CLIP-Guided Diffusion but that didn't work as well

$$\hat{\mu}_\theta(x_t|c) = \mu_\theta(x_t|c) + s \cdot \Sigma_\theta(x_t|c)\nabla_{x_t}\left(f(x_t) \cdot g(c)\right)$$

[Nichol*, Dhariwal*, Ramesh* et al, 2022]

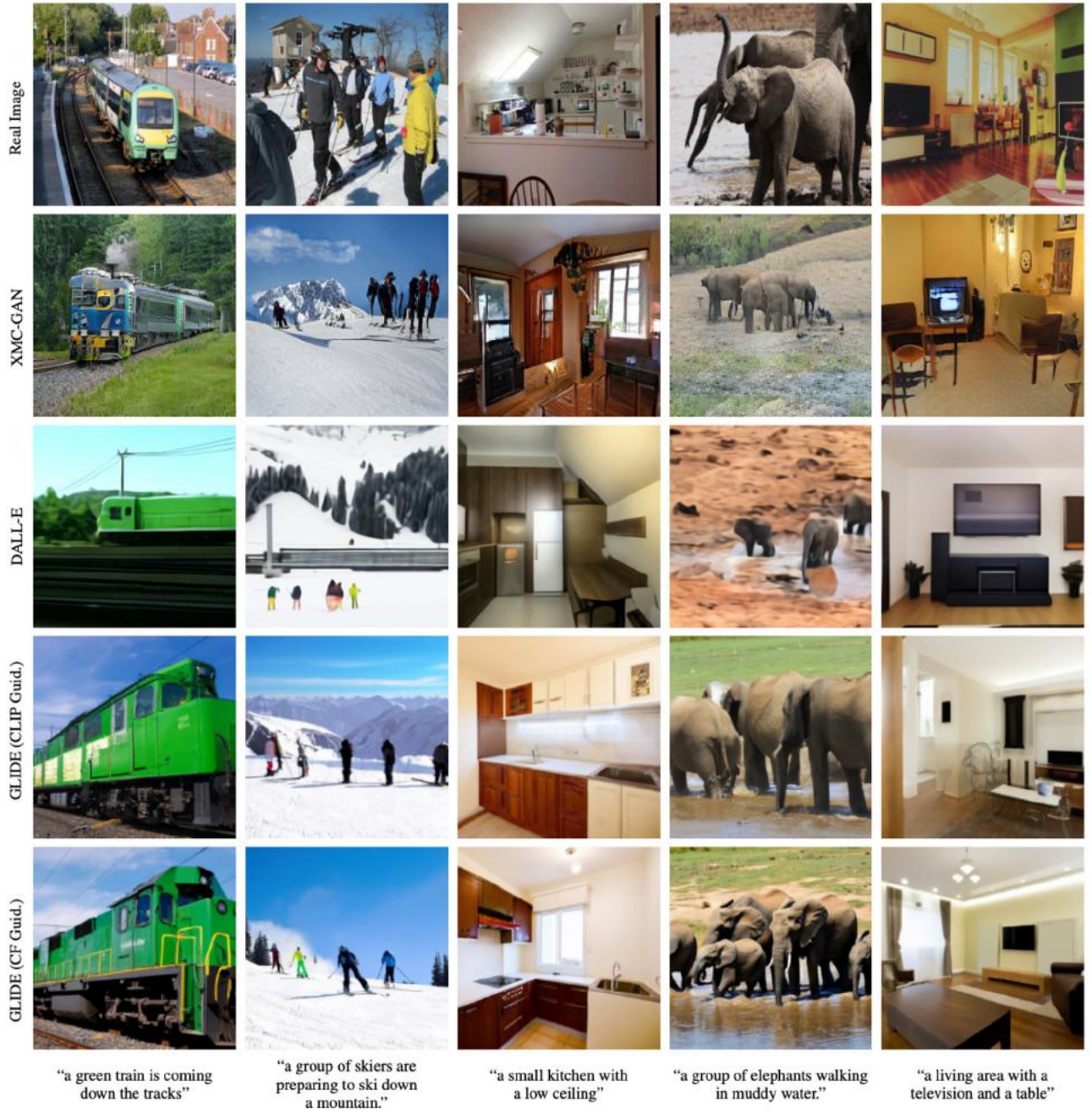


Figure 5. Random image samples on MS-COCO prompts. For XMC-GAN, we take samples from Zhang et al. (2021). For DALL-E, we generate samples at temperature 0.85 and select the best of 256 using CLIP reranking. For GLIDE, we use CLIP guidance with scale 2.0 and classifier-free guidance with scale 3.0. We do not perform any CLIP reranking or cherry-picking for GLIDE.

[Nichol*, Dhariwal*, Ramesh* et al, 2022]

Lecture overview

- Motivation
- Energy-based models
- Score-based Models
- **Denoising Diffusion Models**
  - Overview
  - Denoising Diffusion Probabilistic Models
  - Diffusion Models Beat GANs on Image Synthesis; Classifier Guidance
  - Classifier-Free Diffusion Guidance
  - GLIDE: Text-Guided Diffusion Models
  - **Dall-E: 1, 2, 3**
  - Imagen
- Deeper Dive into Diffusion Models

Dall-E1

Zero-Shot Text-to-Image Generation

Aditya Ramesh [1] Mikhail Pavlov [1] Gabriel Goh [1] Scott Gray [1]
Chelsea Voss [1] Alec Radford [1] Mark Chen [1] Ilya Sutskever [1]

NOTE: Dall-E1 does not involve a diffusion model, but later Dall-E models do

Ramesh, Aditya, et al. "Zero-shot text-to-image generation." International Conference on Machine Learning. PMLR, 2021.

Dall-E1

Key Idea:

- Transformer with text tokens and image tokens
- Image tokens from discrete variational autoencoder, k=8192
  - ~VQ-VAE with Gumbel-Softmax hence trainable as VAE, some small architecture changes
- 250 million text image pairs
- Sample generation with post-processing by CLIP re-ranking of samples

Dall-E1

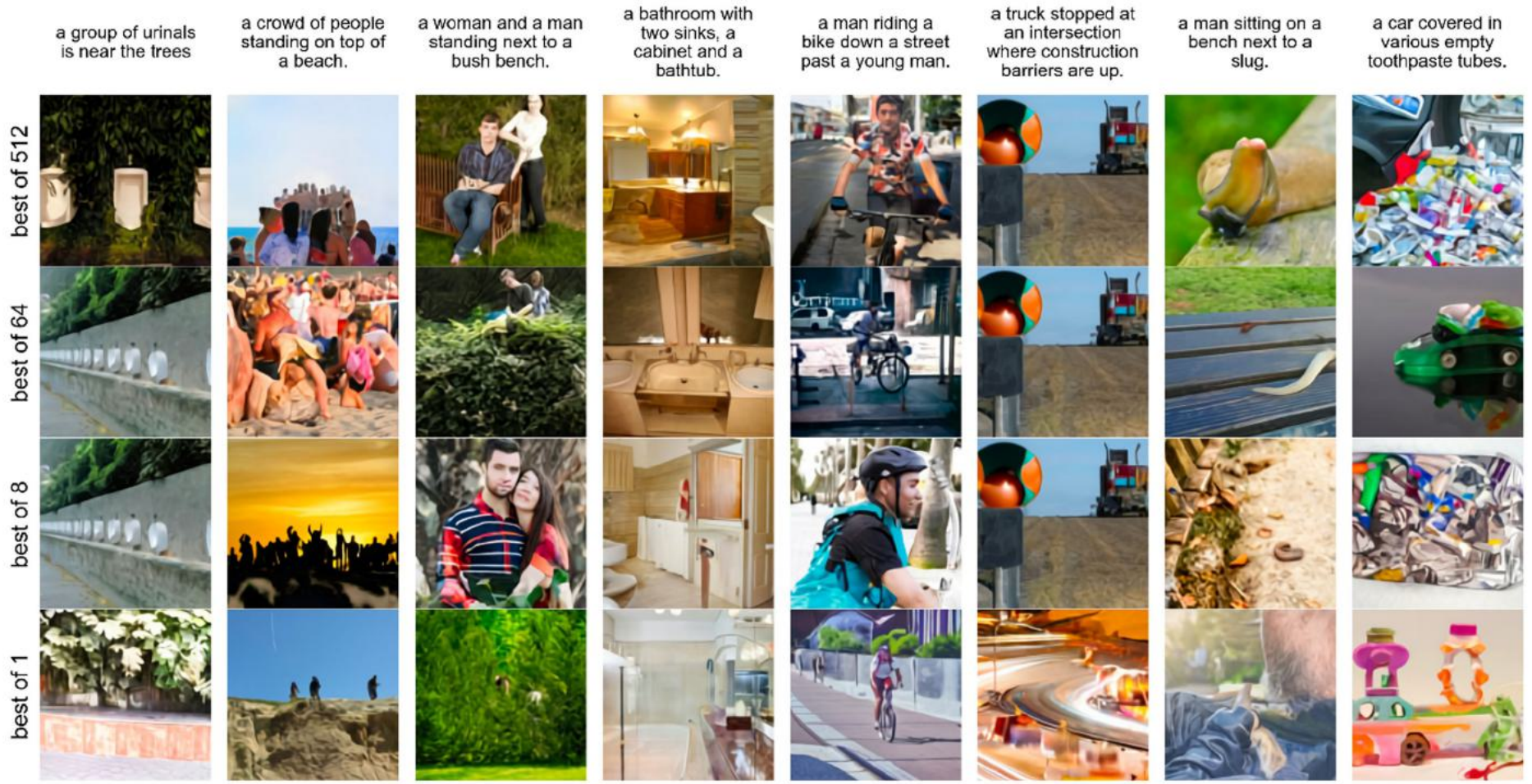


Figure 6. Effect of increasing the number of images for the contrastive reranking procedure on MS-COCO captions.

Findings

- Generalization ability to compose many concepts at high level of abstraction (including unusual combinations)

(a) a tapir made of accordion. a tapir with the texture of an accordion.

(b) an illustration of a baby hedgehog in a christmas sweater walking a dog

(c) a neon sign that reads "backprop". a neon sign that reads "backprop". backprop neon sign

(d) the exact same cat on the top as a sketch on the bottom

Dall-E2 / unCLIP

Hierarchical Text-Conditional Image Generation with CLIP Latents

Aditya Ramesh*
OpenAI
aramesh@openai.com

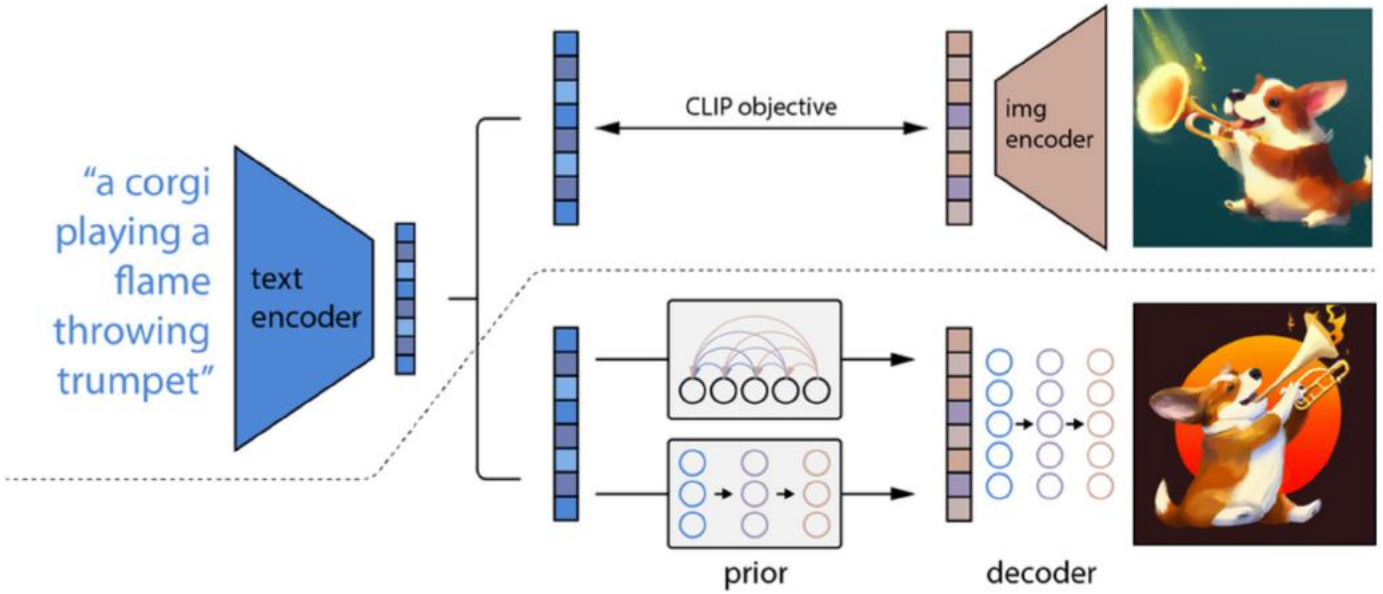
Prafulla Dhariwal*
OpenAI
prafulla@openai.com

Alex Nichol*
OpenAI
alex@openai.com

Casey Chu*
OpenAI
casey@openai.com

Mark Chen
OpenAI
mark@openai.com

Dall-E2 / unCLIP



First 64x64
Then upsample 256x256
Then upsample 1024x1024
No conditioning in upsamplers

Figure 2: A high-level overview of unCLIP. Above the dotted line, we depict the CLIP training process, through which we learn a joint representation space for text and images. Below the dotted line, we depict our text-to-image generation process: a CLIP text embedding is first fed to an autoregressive or diffusion prior to produce an image embedding, and then this embedding is used to condition a diffusion decoder which produces a final image. Note that the CLIP model is frozen during training of the prior and decoder.

Decoder = ~ GLIDE but conditioned on CLIP image embedding space (+text)
Enable classifier-free guidance by setting CLIP embeddings = 0 (10%) and dropping text caption 50% of the time

Dall-E2 / unCLIP: Samples

vibrant portrait painting of Salvador Dalí with a robotic half face

a shiba inu wearing a beret and black turtleneck

a close up of a handpalm with leaves growing from it

Encode with CLIP then Decode/Sample

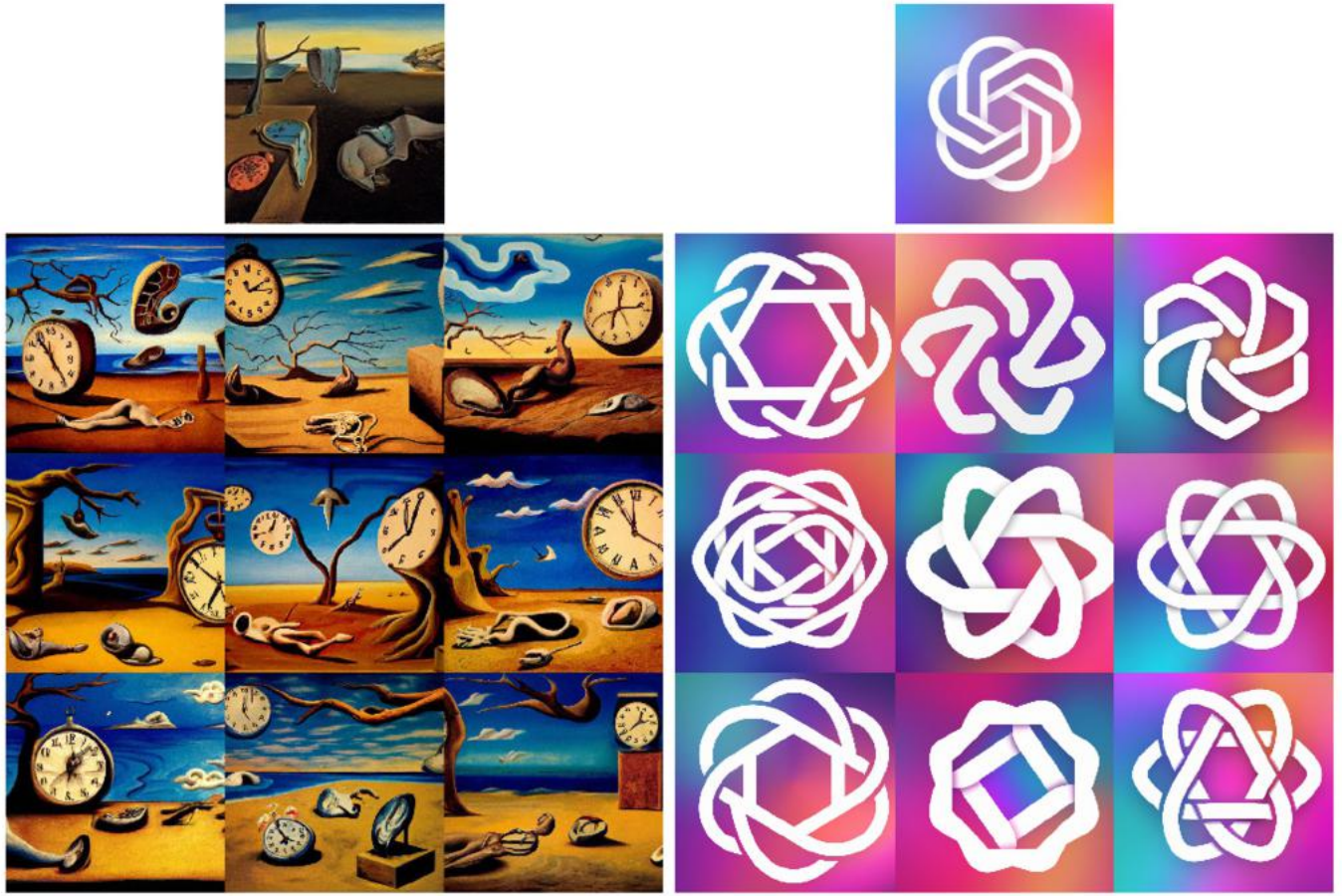
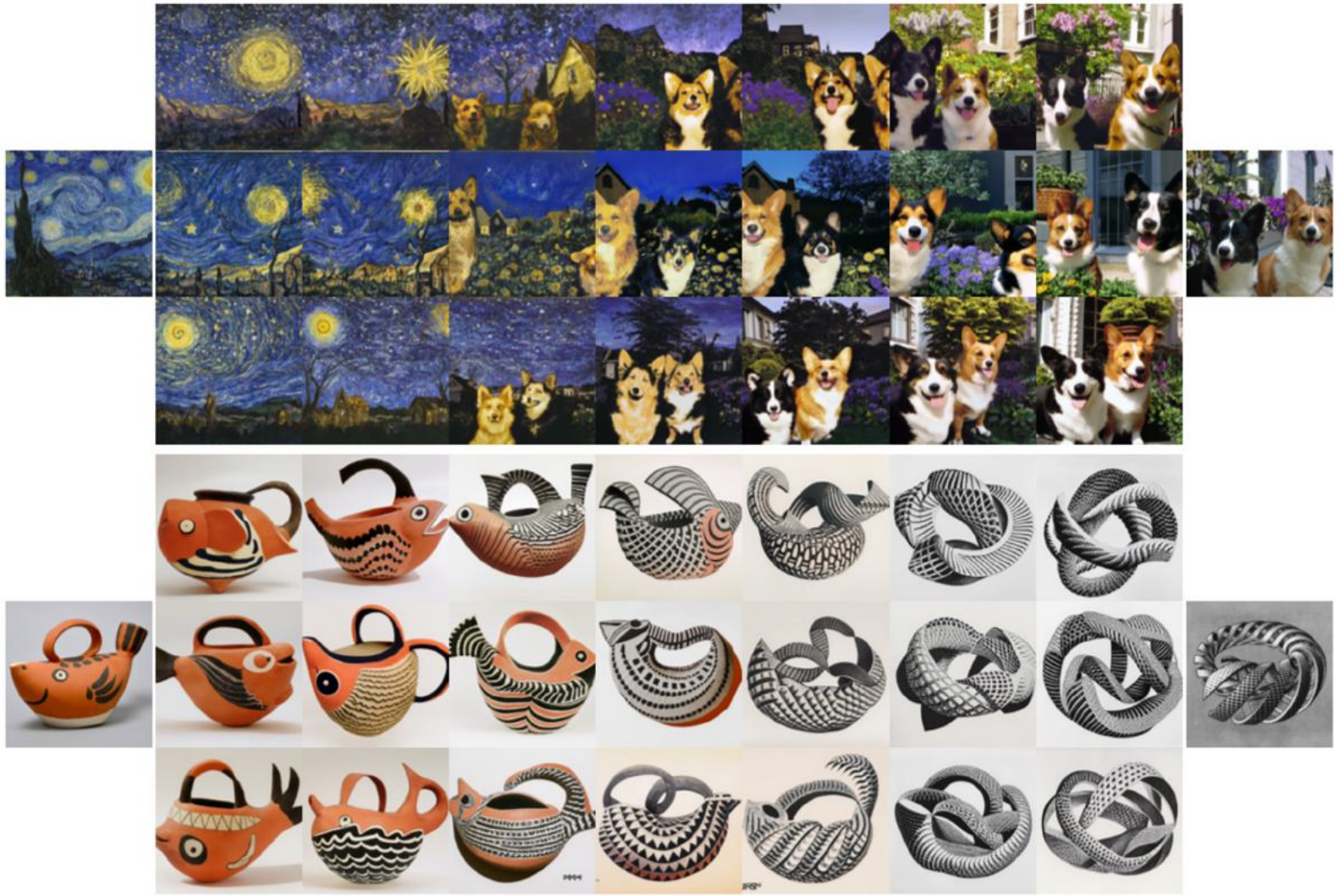


Figure 3: Variations of an input image by encoding with CLIP and then decoding with a diffusion model. The variations preserve both semantic information like presence of a clock in the painting and the overlapping strokes in the logo, as well as stylistic elements like the surrealism in the painting and the color gradients in the logo, while varying the non-essential details.

Interpolation in CLIP image embeddings



Dall-E3

Improving Image Generation with Better Captions

James Betker[*†] jbetker@openai.com

Gabriel Goh[*†] ggoh@openai.com

Li Jing[*†] lijing@openai.com

Tim Brooks[†]

Jianfeng Wang[‡] **Linjie Li**[‡] **Long Ouyang**[†] **Juntang Zhuang**[†] **Joyce Lee**[†] **Yufei Guo**[†]

Wesam Manassra[†] **Prafulla Dhariwal**[†] **Casey Chu**[†] **Yunxin Jiao**[†]

Aditya Ramesh[*†] aramesh@openai.com

[Betker, Goh, Jing, et al, Sep 2023]

Dall-E3

Key Hypothesis:

- Existing text-to-image models struggle to follow detailed image descriptions and often ignore words or confuse the meaning of prompts. We hypothesize that this issue stems from noisy and inaccurate image captions in the training dataset.

Proposed Method:

- train a bespoke image captioner and use it to recaption the training dataset
- at sample time: use GPT-4 to “upsample” the prompt

Underlying Model:

- ~ stable diffusion VAE: 256x256 -> 32x32 latents + diffusion (not VAE!) decoder; might have super-resolution diffusion or might not (details undisclosed)

Results:

- Even better :)

Weaknesses:

- Spatial awareness
- Text rendering not as good as hoped for (possibly because of tokenizer)

In a fantastical setting, a highly detailed furry humanoid skunk with piercing eyes confidently poses in a medium shot, wearing an animal hide jacket. The artist has masterfully rendered the character in digital art, capturing the intricate details of fur and clothing texture.

A illustration from a graphic novel. A bustling city street under the shine of a full moon. The sidewalks bustling with pedestrians enjoying the nightlife. At the corner stall, a young woman with fiery red hair, dressed in a signature velvet cloak, is haggling with the grumpy old vendor. the grumpy vendor, a tall, sophisticated man is wearing a sharp suit, sports a noteworthy moustache is animatedly conversing on his steampunk telephone.

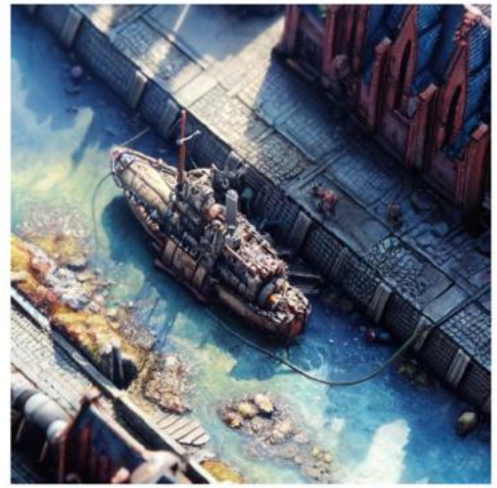
Ancient pages filled with sketches and writings of fantasy beasts, monsters, and plants sprawl across an old, weathered journal. The faded dark green ink tells tales of magical adventures, while the high-resolution drawings detail each creature's intricate characteristics. Sunlight peeks through a nearby window, illuminating the pages and revealing their timeworn charm.

A vibrant 1960s-style poster depicting interplanetary migration, with a retro rocket ship blasting off from earth towards a distant, colorful planet. Bold typography announces "Join the galactic adventure!" with smaller text underneath reading "Explore new worlds, build a brighter future." The background features a swirling galaxy of stars and constellations.

A mischievous ferret with a playful grin squeezes itself into a large glass jar, surrounded by colorful candy. The jar sits on a wooden table in a cozy kitchen, and warm sunlight filters through a nearby window.

A bird scaring a scarecrow.

Paying for a quarter-sized pizza with a pizza-sized quarter.



A smafml vessef epropoeilled on watvewr by ors, sauls, or han engie.

A large, vibrant bird with an impressive wingspan swoops down from the sky, letting out a piercing call as it approaches a weathered scarecrow in a sunlit field. The scarecrow, dressed in tattered clothing and a straw hat, appears to tremble, almost as if it's coming to life in fear of the approaching bird.

A person is standing at a pizza counter, holding a gigantic quarter the size of a pizza. The cashier, wide-eyed with astonishment, hands over a tiny, quarter-sized pizza in return. The background features various pizza toppings and other customers, all of them equally amazed by the unusual transaction.

A small vessel, propelled on water by oars, sails, or an engine, floats gracefully on a serene lake. the sun casts a warm glow on the water, reflecting the vibrant colors of the sky as birds fly overhead.

Figure 6 – Effect of using "upsampled" drawbench captions to create samples with DALL-E 3. Original drawbench captions on top, upsampled captions on bottom. Images are best of 4 for each caption.

Lecture overview

- Motivation
- Energy-based models
- Score-based Models
- **Denoising Diffusion Models**
  - Overview
  - Denoising Diffusion Probabilistic Models
  - Diffusion Models Beat GANs on Image Synthesis; Classifier Guidance
  - Classifier-Free Diffusion Guidance
  - GLIDE: Text-Guided Diffusion Models
  - Dall-E: 1, 2, 3
  - **Imagen**
- Deeper Dive into Diffusion Models

Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding

**Chitwan Saharia*, William Chan*, Saurabh Saxena†, Lala Li†, Jay Whang†,
Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan,
S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans,
Jonathan Ho†, David J Fleet†, Mohammad Norouzi***

{sahariac,williamchan,mnorouzi}@google.com
{srbs,lala,jwhang,jonathanho,davidfleet}@google.com

Google Research, Brain Team
Toronto, Ontario, Canada

Imagen achieves a new state-of-the-art FID score of 7.27 on the COCO dataset, without ever training on COCO, and human raters find Imagen samples to be on par with the COCO data itself in image-text alignment. To assess text-to-image models in greater depth, we introduce DrawBench, a comprehensive and challenging benchmark for text-to-image models. With DrawBench, we compare Imagen with recent methods including VQ-GAN+CLIP, Latent Diffusion Models, GLIDE and DALL-E 2, and find that human raters prefer Imagen over other models in side-by-side comparisons, both in terms of sample quality and image-text alignment. See imagen.research.google for an overview of the results.

Imagen Architecture

2.5M training steps for each model

2B

600M

400M

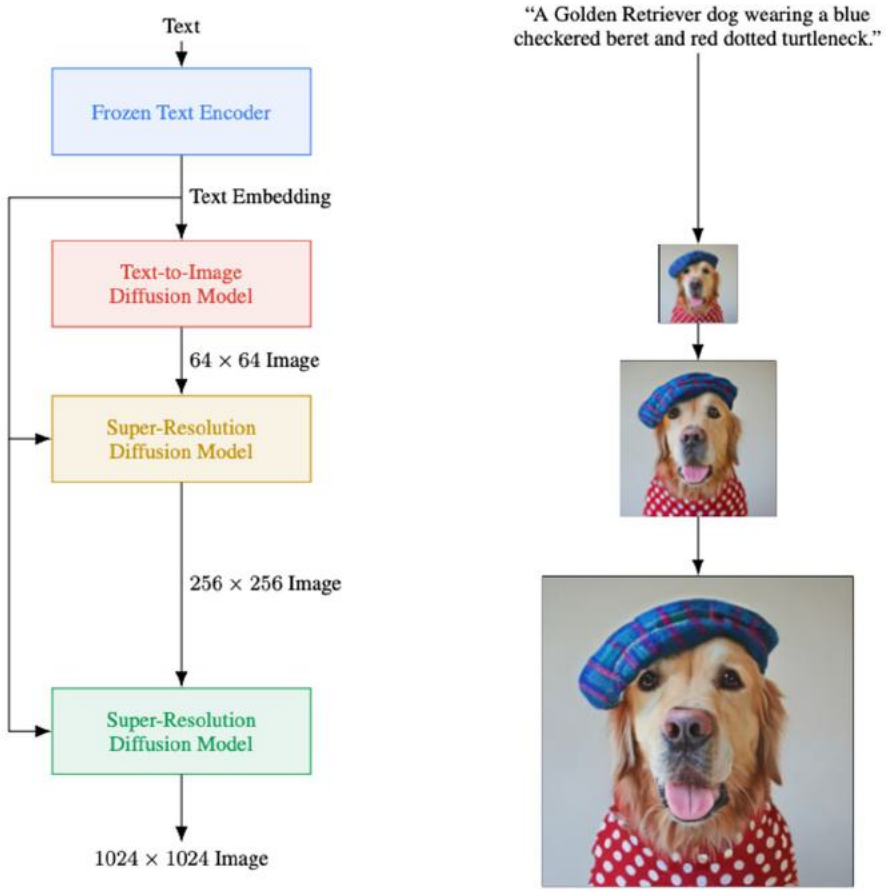


Figure A.4: Visualization of Imagen. Imagen uses a frozen text encoder to encode the input text into text embeddings. A conditional diffusion model maps the text embedding into a 64×64 image. Imagen further utilizes text-conditional super-resolution diffusion models to upsample the image, first $64 \times 64 \rightarrow 256 \times 256$, and then $256 \times 256 \rightarrow 1024 \times 1024$.

[Saharia, Chan, et al, May 2022]

Imagen Key Observations / Ideas

- LLMs trained on text-only (e.g. T5) very effective at encoding text for image synthesis
- Increasing LLM size boost sample fidelity and image-text alignment more than increasing image diffusion model size
- Efficient U-Net: more model parameters at low res

Sprouts in the shape of text ‘Imagen’ coming out of a fairytale book.

A photo of a Shiba Inu dog with a backpack riding a bike. It is wearing sunglasses and a beach hat.

A high contrast portrait of a very happy fuzzy panda dressed as a chef in a high end kitchen making dough. There is a painting of flowers on the wall behind him.

Teddy bears swimming at the Olympics 400m Butterfly event.

A cute corgi lives in a house made out of sushi.

A cute sloth holding a small treasure chest. A bright golden glow is coming from the chest.

New York Skyline with Hello World written with fireworks on the sky.

Lecture overview

- Motivation
- Energy-based models
- Score-based Models
- Denoising Diffusion Models
- **Deeper Dive into Diffusion Models**
  - **DDIM Sampling**
  - Prediction Space / Loss: x, eps, v
  - Issues with Common Noise Schedules
  - Architectures: UNet, latent space / stable diffusion, hierarchical generation, transformers
  - Image Editing / Video Editing
  - Using scores in diffusion models: text-to-3D, text-to-SVG, Video Adapter
  - Faster Sampling
  - Diffusion as Pre-Training
  - Other Fields: visuomotor control, materials discovery

DDIM Sampler

Consider the noising forward process $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon$

Let $\alpha_t = \sqrt{\bar{\alpha}_t}, \sigma_t = \sqrt{1 - \bar{\alpha}_t}$

Then $x_t = \alpha_t x_0 + \sigma_t \epsilon$

Song, Jiaming, Chenlin Meng, and Stefano Ermon. "Denoising diffusion implicit models." arXiv preprint arXiv:2010.02502 (2020).

DDIM Sampler

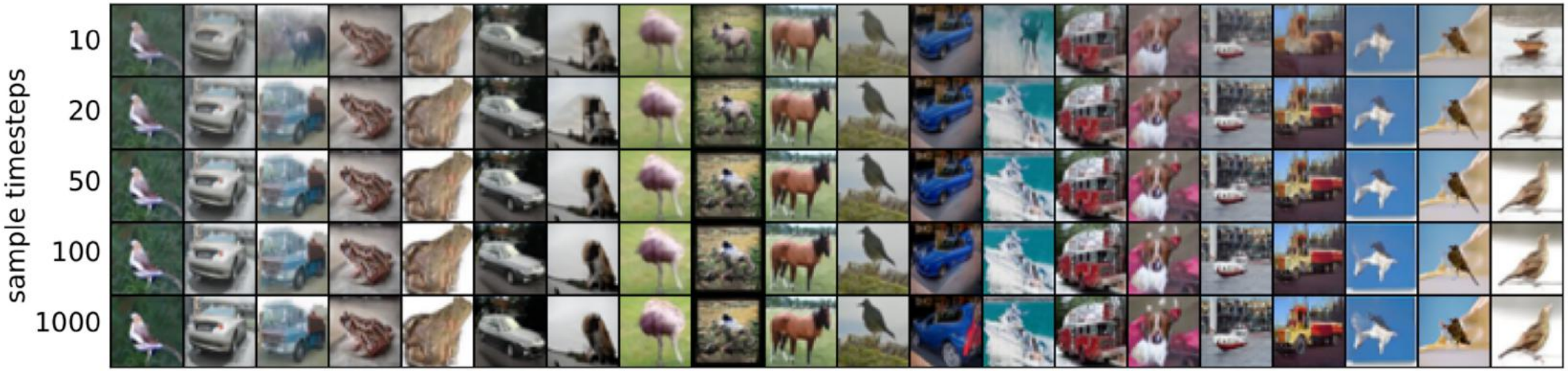
Deterministic DDIM update:

$$x_0 = \frac{x_t - \sigma_t \epsilon}{\alpha_t}$$

$$x_{t-1} = \alpha_{t-1} \underbrace{\left(\frac{x_t - \sigma_t \hat{\epsilon}(x_t)}{\alpha_t} \right)}_{\text{“ predicted } \boldsymbol{x}_0\text{”}} + \underbrace{\sigma_{t-1} \hat{\epsilon}(x_t)}_{\text{“direction pointing to } \boldsymbol{x}_t\text{”}}$$

DDIM Sampler

Can choose a different step size during inference than training



Most models (e.g. SD) can generate good samples even with 50 steps

Lecture overview

- Motivation
- Energy-based models
- Score-based Models
- Denoising Diffusion Models
- **Deeper Dive into Diffusion Models**
  - DDIM Sampling
  - **Prediction Space / Loss: x, eps, v**
  - Issues with Common Noise Schedules
  - Architectures: UNet, latent space / stable diffusion, hierarchical generation, transformers
  - Image Editing / Video Editing
  - Using scores in diffusion models: text-to-3D, text-to-SVG, Video Adapter
  - Faster Sampling
  - Diffusion as Pre-Training
  - Other Fields: visuomotor control, materials discovery

Different Reparameterizations

$$x_t = \alpha_t x_0 + \sigma_t \epsilon$$

Epsilon - space (DDPM)

$$\mathcal{L}(x_0, t) = \|\epsilon - \hat{\epsilon}_\theta(\alpha_t x_0 + \sigma_t \epsilon)\|_2^2$$

Salimans, Tim, and Jonathan Ho. "Progressive distillation for fast sampling of diffusion models." arXiv preprint arXiv:2202.00512 (2022).

Different Reparameterizations

$$x_t = \alpha_t x_0 + \sigma_t \epsilon$$

x - space

$$\mathcal{L}(x_0, t) = \|x_0 - \hat{x}_\theta(\alpha_t x_0 + \sigma_t \epsilon)\|_2^2$$

Different Reparameterizations

Relationship between x-space and epsilon-space

$$\mathcal{L}(x_0, t) = \|x_0 - \hat{x}_\theta(x_t)\|_2^2$$
$$= \left\| \frac{x_t - \sigma_t \epsilon}{\alpha_t} - \frac{x_t - \sigma_t \hat{\epsilon}_\theta(x_t)}{\alpha_t} \right\|_2^2$$
$$= \frac{\sigma_t^2}{\alpha_t^2} \|\epsilon - \hat{\epsilon}_\theta(x_t)\|_2^2$$

$$x_t = \alpha_t x_0 + \sigma_t \epsilon$$

$$x_0 = \frac{x_t - \sigma_t \epsilon}{\alpha_t}$$

$$\frac{\alpha_t^2}{\sigma_t^2} \|x_0 - \hat{x}_\theta(x_t)\|_2^2 = \|\epsilon - \hat{\epsilon}_\theta(x_t)\|_2^2$$

Different Reparameterizations

How about something else?

Note that DDPM defines: $\alpha_t = \sqrt{\bar{\alpha}_t}, \sigma_t = \sqrt{1 - \bar{\alpha}_t}$

(α_t, σ_t) defines vectors on the unit circle.

Instead of DDIM by t, can we do by angle $\phi = \arctan(\sigma_t / \alpha_t)$

Different Reparameterizations

We can define the forward process as:

$$x_\phi = \cos(\phi)x_0 + \sin(\phi)\epsilon$$

Then, define the "velocity" as:

$$v_\phi = \frac{dx_\phi}{d\phi} = \cos(\phi)\epsilon - \sin(\epsilon)x_0$$

DDIM update rule is then:

$$x_{\phi_t - \delta} = \cos(\delta)z_{\phi_t} - \sin(\delta)\hat{v}_\theta(x_{\phi_t})$$

Different Reparameterizations

$$x_t = \alpha_t x_0 + \sigma_t \epsilon$$

v - space

$$\mathcal{L}(x_0, t) = \|v - \hat{v}(\alpha_t x_0 + \sigma_t \epsilon)\|_2^2$$

where

$$v = \alpha_t \epsilon - \sigma_t x_0$$

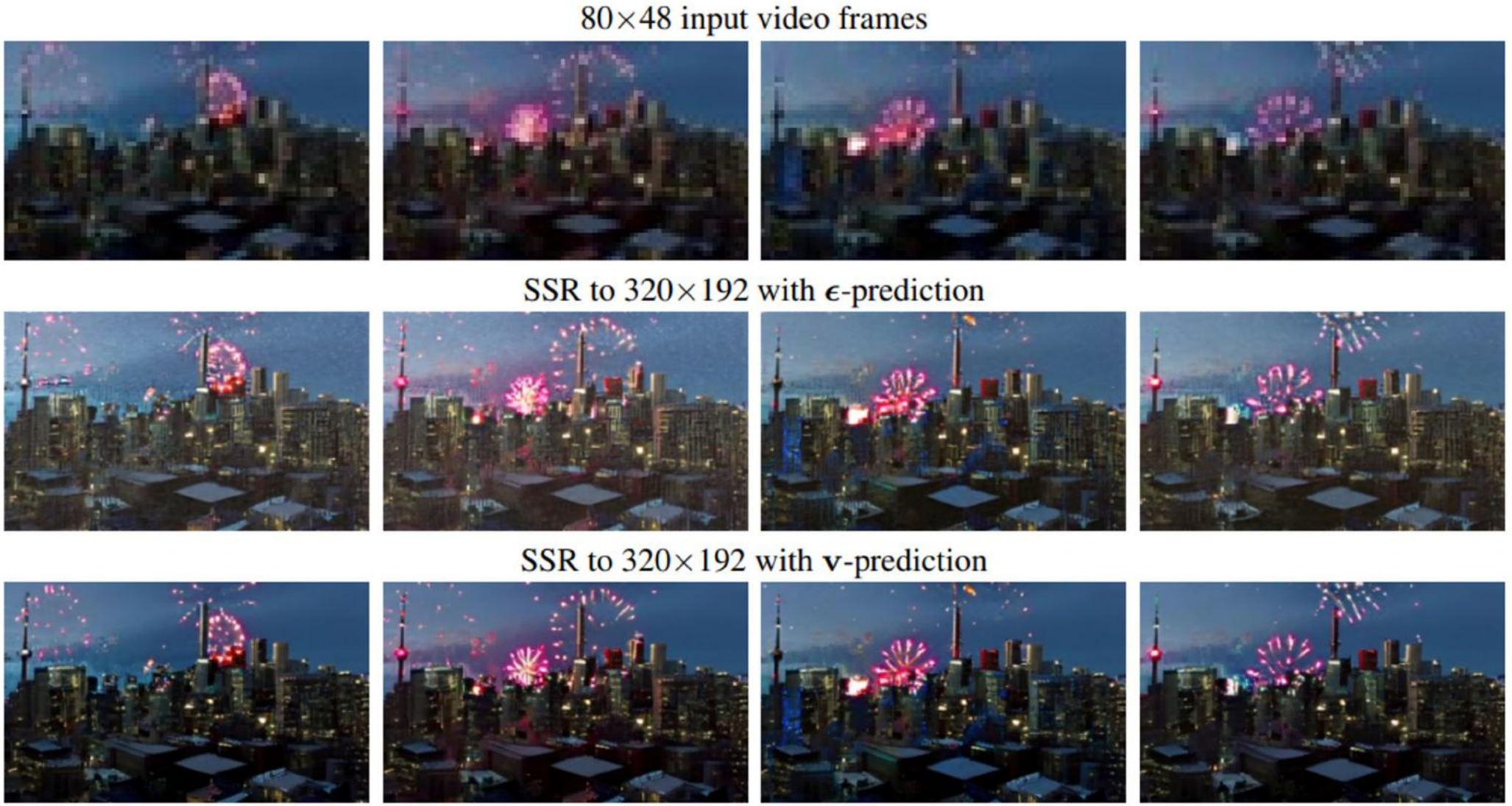
Different Reparameterizations

Relationship between x-space and v-space $x = \alpha_t x_t - \sigma_t v$

$$\begin{aligned}\mathcal{L}(x_0, t) &= \|x_0 - \hat{x}_\theta(x_t)\|_2^2 \\ &= \|(\alpha_t x_t - \sigma_t v) - (\alpha_t x_t - \sigma_t \hat{v}_\theta(x_t)\|_2^2 \\ &= \sigma_t^2 \|v - \hat{v}_\theta(x_t)\|_2^2\end{aligned}$$

$$\left(1 + \frac{\alpha_t^2}{\sigma_t^2}\right) \|x_0 - \hat{x}_\theta(x_t)\|_2^2 = \|v - \hat{v}_\theta(x_t)\|_2^2$$

Benefits of v-space

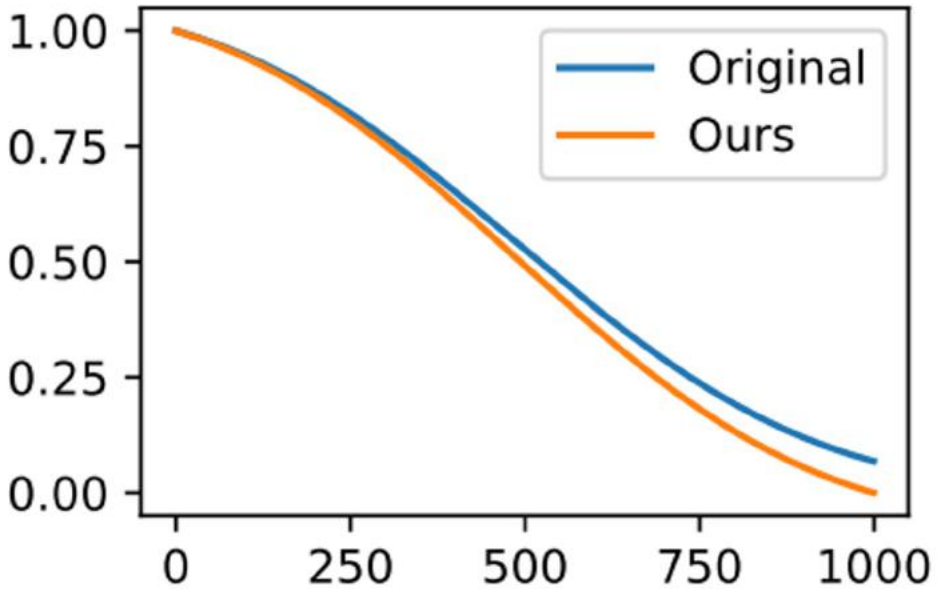


Lecture overview

- Motivation
- Energy-based models
- Score-based Models
- Denoising Diffusion Models
- **Deeper Dive into Diffusion Models**
  - DDIM Sampling
  - Prediction Space / Loss: x, eps, v
  - **Issues with Common Noise Schedules**
  - Architectures: UNet, latent space / stable diffusion, hierarchical generation, transformers
  - Image Editing / Video Editing
  - Using scores in diffusion models: text-to-3D, text-to-SVG, Video Adapter
  - Faster Sampling
  - Diffusion as Pre-Training
  - Other Fields: visuomotor control, materials discovery

Issues with Common Noise Schedules

Most diffusion noise schedules do not end at 0 for t = T



(b) $\sqrt{\bar{\alpha}_t}$

$$\alpha_t = \sqrt{\bar{\alpha}_t}, \sigma_t = \sqrt{1 - \bar{\alpha}_t}$$

$$x_t = \alpha_t x_0 + \sigma_t \epsilon$$

Lin, Shanchuan, et al. "Common diffusion noise schedules and sample steps are flawed." Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. 2024.

Issues with Common Noise Schedules

DDPM uses epsilon-space for loss. What happens when t = T?

If t = T, then $\alpha_t = 0$ and $\sigma_t = 1$ (pure gaussian noise)

$$\mathcal{L}(x_0, t) = \|\epsilon - \hat{\epsilon}_\theta(0 * x_0 + 1 * \epsilon)\|_2^2$$

The objective becomes a trivial task (learn the identity function)

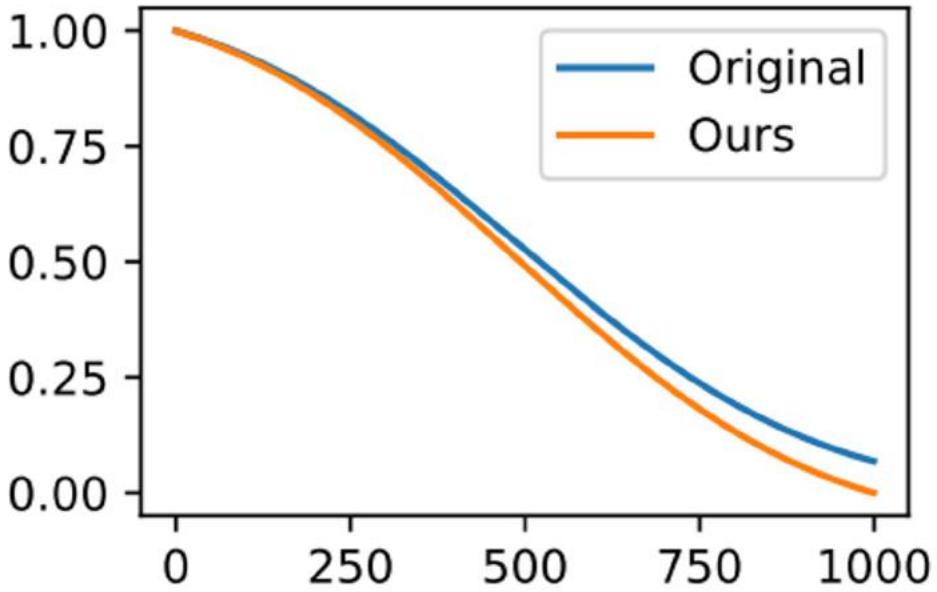
Issues with Common Noise Schedules

Why is this an issue?

- Leaks global information about x_0 during training since it is not fully noised
- During inference, we sample pure noise. The model will try to interpret signal where there is none.

Zero-Terminal SNR

We can enforce 0 terminal SNR by adjusting the noise schedule



(b) $\sqrt{\bar{\alpha}_t}$

$$\alpha_t = \sqrt{\bar{\alpha}_t}, \sigma_t = \sqrt{1 - \bar{\alpha}_t}$$

$$x_t = \alpha_t x_0 + \sigma_t \epsilon$$

Zero-Terminal SNR

But we can't use epsilon-space loss - v space to the rescue!

At t = T, v-space loss is

$$\mathcal{L}(x_0, t) = \|0 * \epsilon - 1 * x_0 - \hat{v}_\theta(0 * x_0 + 1 * \epsilon)\|_2^2 = \|x_0 - \hat{v}_\theta(\epsilon)\|_2^2$$

Process:

- Re-scale noise schedule to enforce zero-terminal SNR
- Finetune diffusion model with v-space loss on new noise schedule

Zero-Terminal SNR

SD

SD + Zero SNR

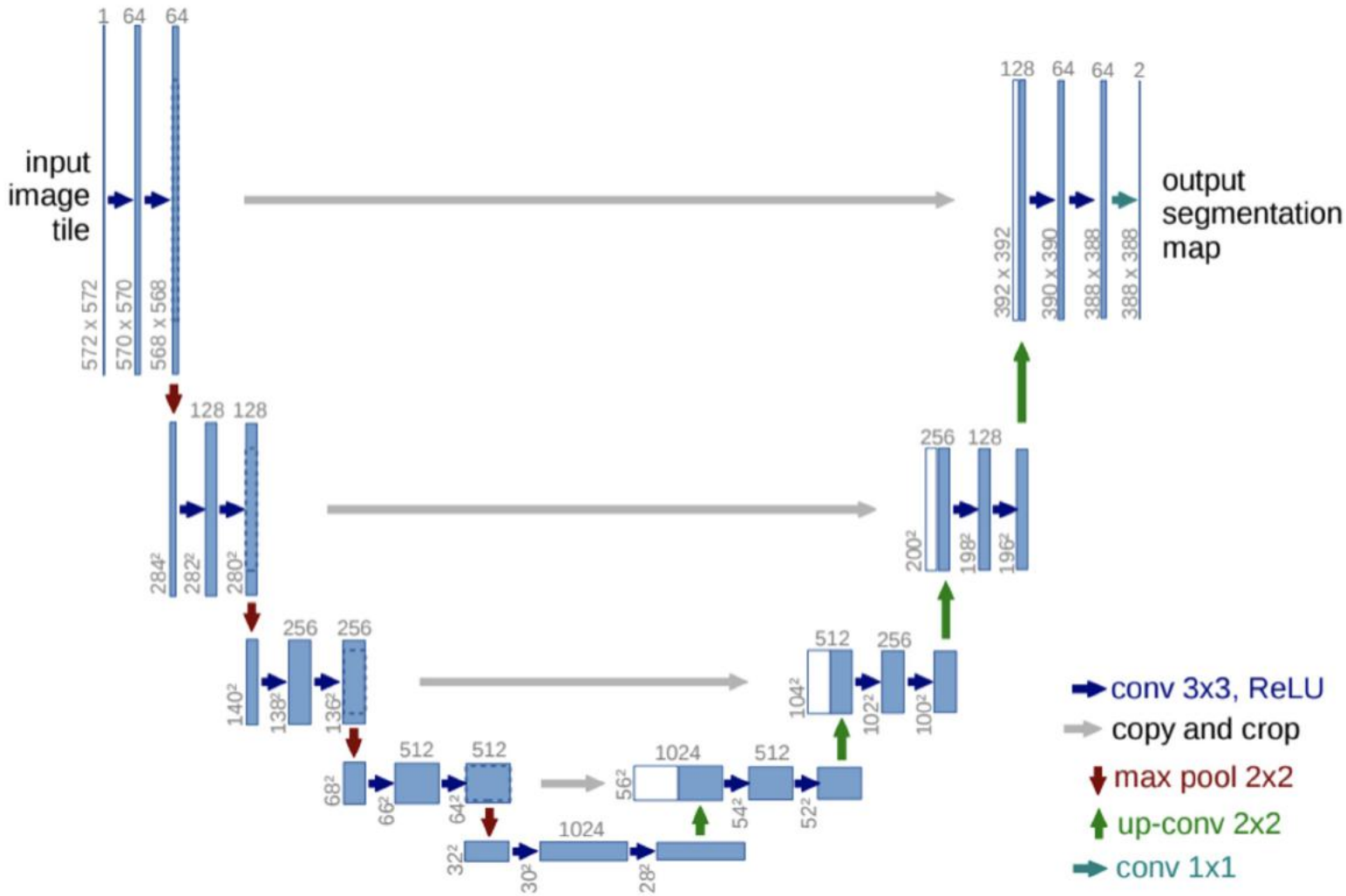
(e) A bald eagle against a white background

Zero-Terminal SNR

Lecture overview

- Motivation
- Energy-based models
- Score-based Models
- Denoising Diffusion Models
- **Deeper Dive into Diffusion Models**
  - DDIM Sampling
  - Prediction Space / Loss: x, eps, v
  - Issues with Common Noise Schedules
  - **Architectures: UNet, latent space / stable diffusion, hierarchical generation, transformers**
  - Image Editing / Video Editing
  - Using scores in diffusion models: text-to-3D, text-to-SVG, Video Adapter
  - Faster Sampling
  - Diffusion as Pre-Training
  - Other Fields: visuomotor control, materials discovery

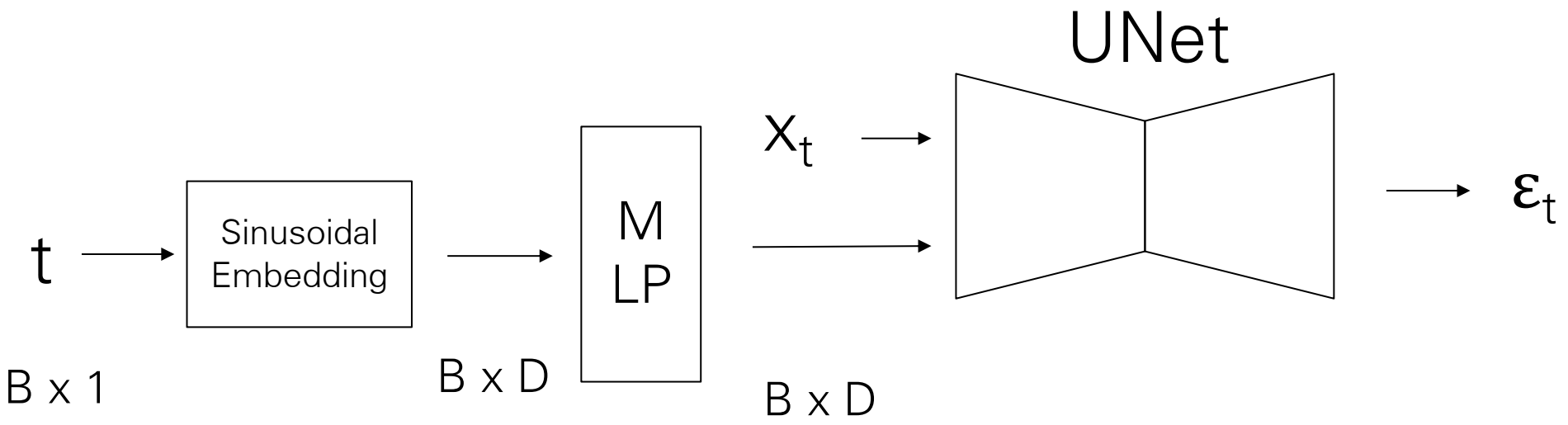
UNet - Base Model



Additionally add some self-attention layers at lower resolutions

UNet - Conditioning

Condition on diffusion timestep t



UNet - Conditioning

Shapes: h (H x W x D), t_{emb} (1 x 1 x D)
Method 1: Simple addition

$$h = h + t_{emb}$$

Method 2: Modulation parameters

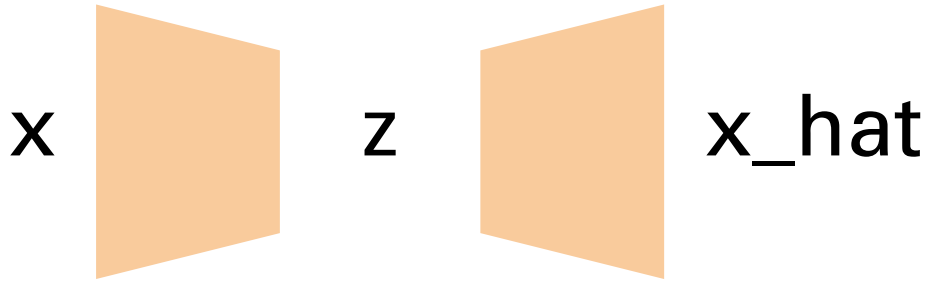
$$h = \text{GroupNorm}(h) * (1 + f_{scale}(t_{emb})) + f_{shift}(t_{emb})$$

Latent Diffusion Models

Pixels can still be expensive.

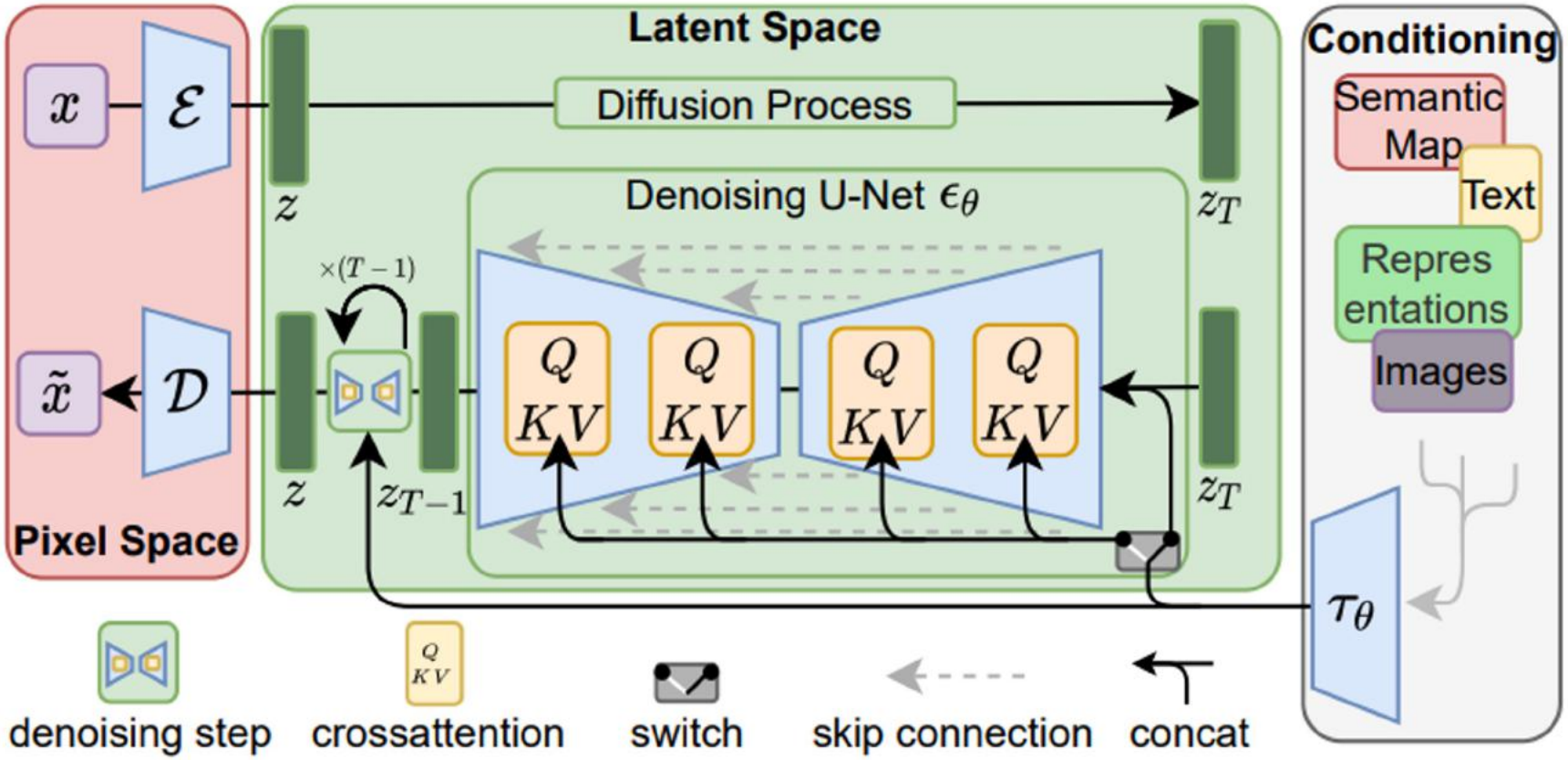
Can we learn a latent space like we did for AR models (VQGAN)?

Learn a VAE: very low weighting on KL loss (1e-6)



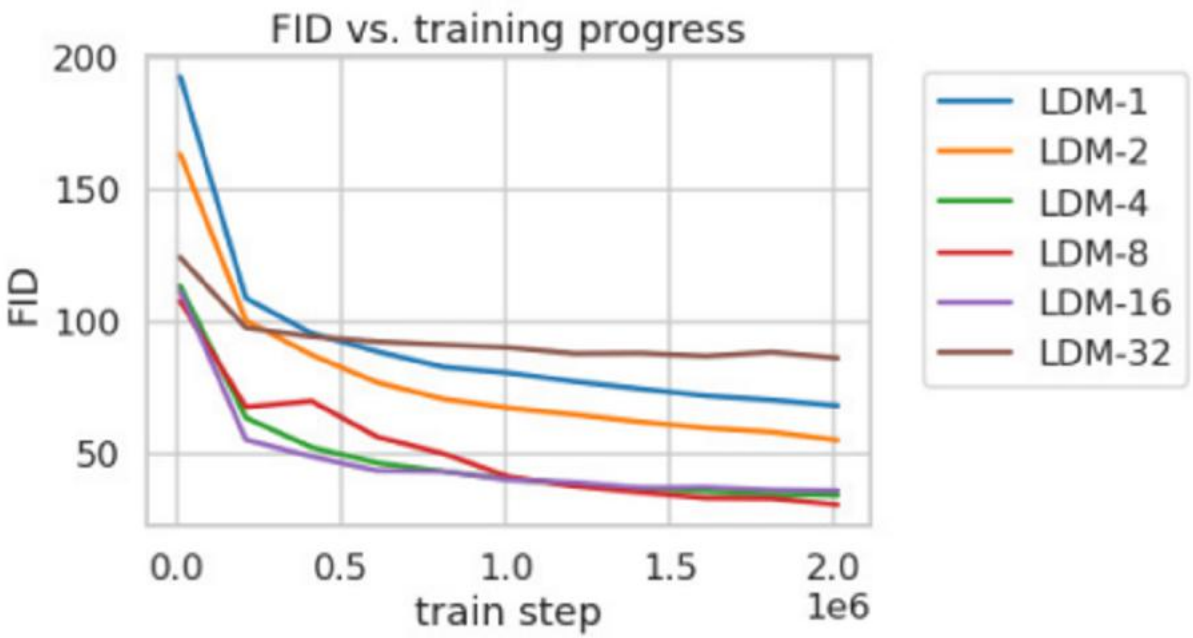
Rombach, Robin, et al. "High-resolution image synthesis with latent diffusion models." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022.

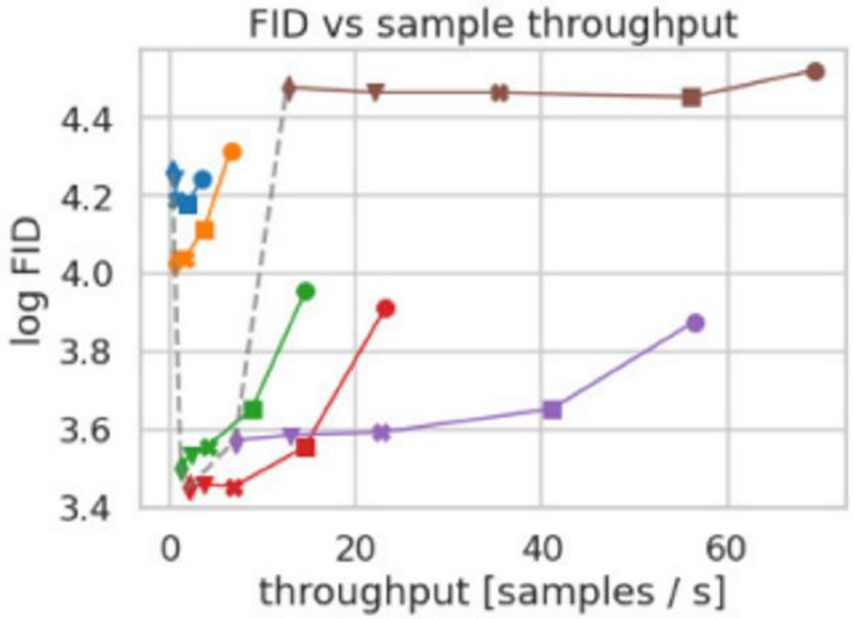
Latent Diffusion Model



Latent Diffusion Model

Speed - Quality Trade-off





Stable Diffusion

- VAE that downsamples images by 8x spatially (256 -> 32, 512 > 64)
- UNet architecture with self-attention layers
- Text conditioning with cross attention on text features
  - Text features using CLIP text encoder

Stable Video Diffusion

- Initialize from Stable Diffusion
- Add temporal layers (e.g. 1D Conv, temporal attention)
- Finetune on video data

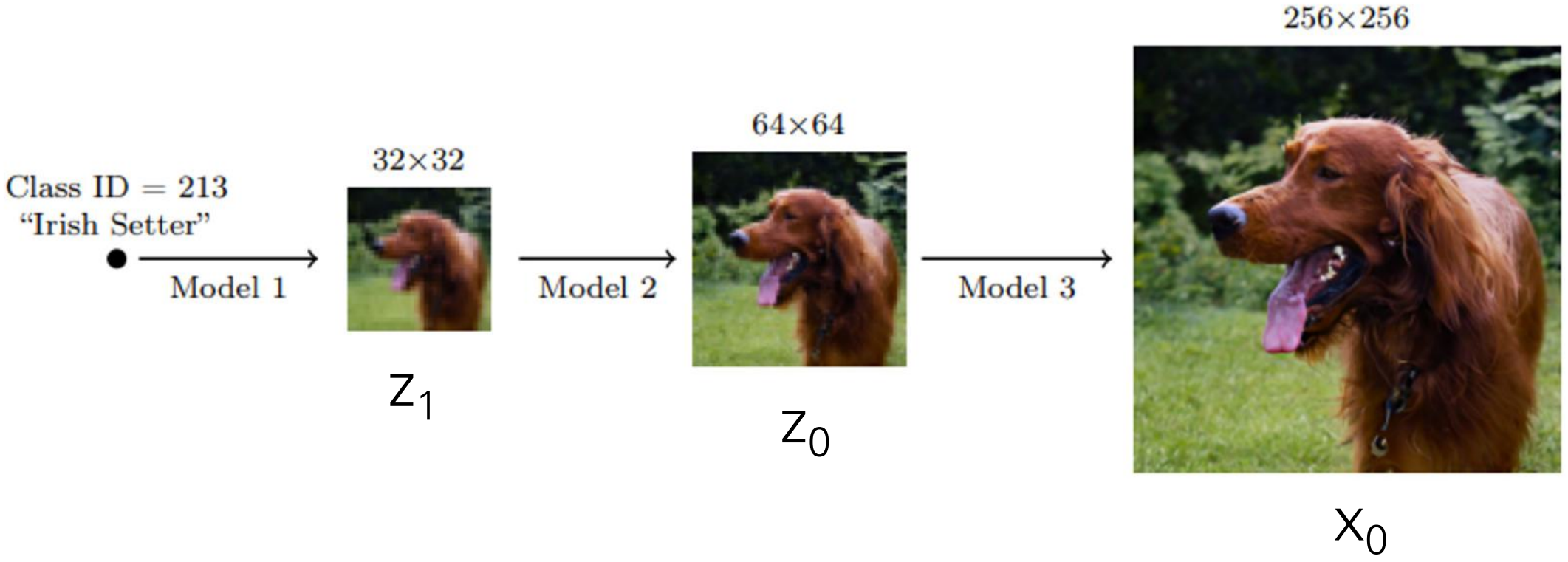
Data pipeline:

- Scrape **unlabeled** videos from the internet
- Use image + video captioners to generate synthetic text labels
- Apply aggressive data filtering using several heuristics (clip score, optical flow score, aesthetic scores, OCR, etc.

Blattmann, Andreas, et al. "Stable video diffusion: Scaling latent video diffusion models to large datasets." arXiv preprint arXiv:2311.15127 (2023).

Stable Video Diffusion

Cascaded Diffusion Models



$$p_\theta(x_0) = \int p_\theta(z_1) p_\theta(z_0 \mid z_1) p_\theta(x_0 \mid z_0) dz$$

Ho, Jonathan, et al. "Cascaded diffusion models for high fidelity image generation." The Journal of Machine Learning Research 23.1 (2022): 2249-2281.

Cascaded Diffusion Models

- Train 1 unconditional model, and the rest are low-resolution conditional super-resolution models
  - Condition on low-res image -> bilinear / bicubic upsample -> concatenate to noised input
- Models can be trained independently
- Hyperparameters can be tuned to be resolution specific

Cascaded Diffusion Models

Conditioning Augmentation: Augment the low-res input z

- Method 1: Blurring
  - Gaussian blurring filter
  - Good for models at 128, 256 resolution
- Method 2: Non-truncated Conditioning Augmentation
  - Train the super-resolution $p(x|z_0)$ model conditioned on samples from the prior $p(z_0)$ with added corruption from the forward process $q(z_s|z_0)$.
  - S (corruption / noise strength) is a hyperparameter
  - Good for < 128 resolution

Cascaded Diffusion Models

(a) 16×16 base

(b) 16×16→64×64 super-resolution, $s = 0$

Cascaded Diffusion Models

Large improvements in using conditioning augmentation

Conditioning	FID vs train	FID vs validation	IS
No cascading	2.35	2.91	52.72 ± 1.15
$16\times16 \rightarrow 64\times64$ cascading			
$s = 0$	6.02	5.84	35.59 ± 1.19
$s = 101$	3.41	3.67	44.72 ± 1.12
$s = 1001$	**2.13**	**2.79**	$\mathbf{54.47 \pm 1.05}$

Imagen Video

Ho, Jonathan, et al. "Imagen video: High definition video generation with diffusion models." arXiv preprint arXiv:2210.02303 (2022).

Imagen Video

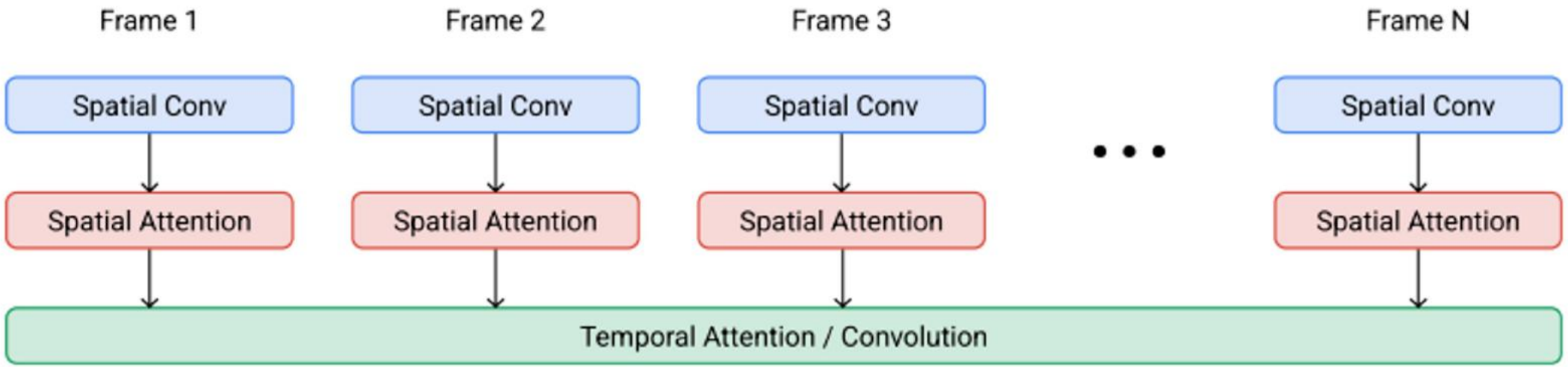


Imagen Video

Key aspects

- Cascaded structure allows better scalability on more limited compute
- Conditioning augmentation for super-res models
- Video-image joint training
- Large, high-quality dataset (14M text-video, 500M text-image)

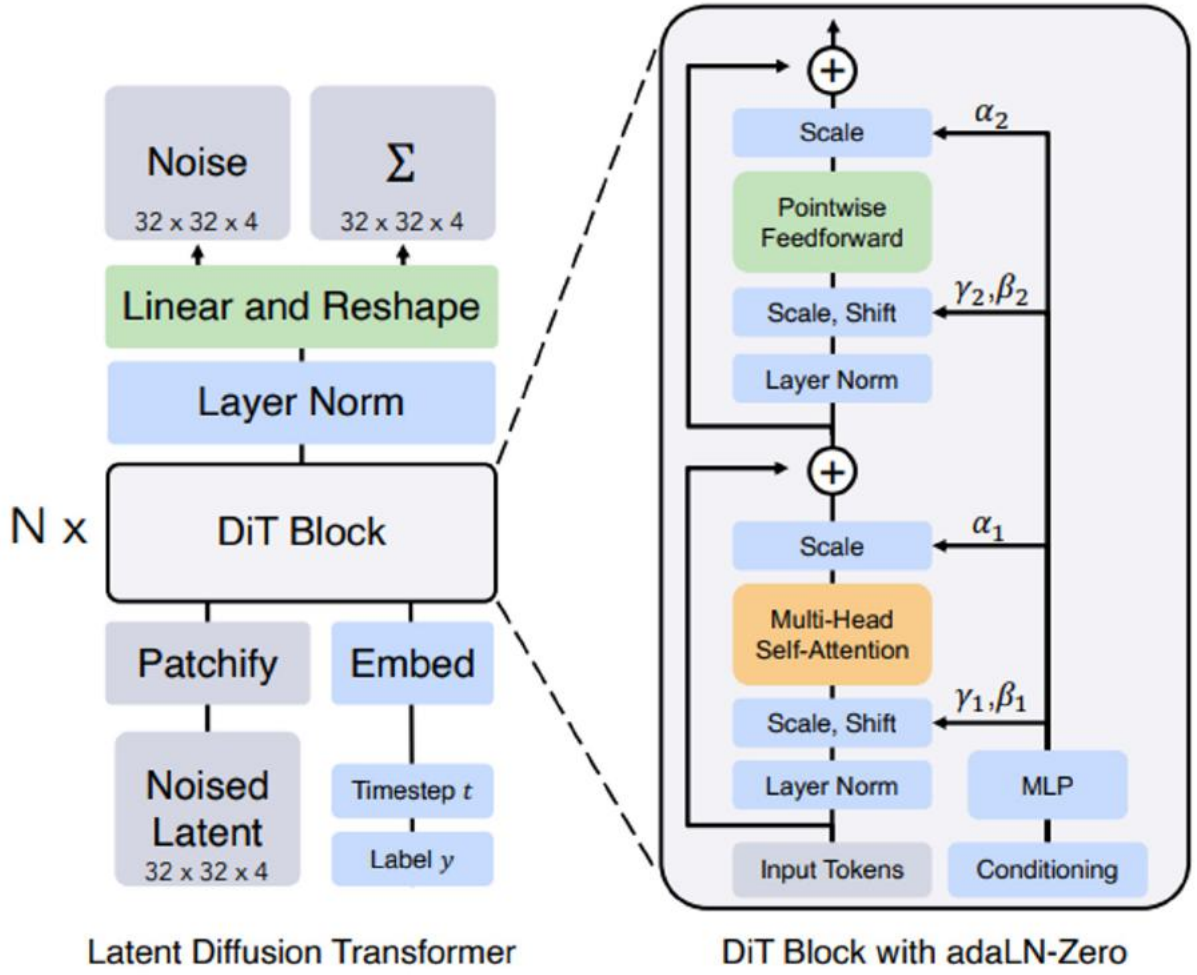
Imagen Video

A teddy bear running in New York City

Imagen Video

Coffee pouring into a cup

Diffusion Transformers



Peebles, William, and Saining Xie. "Scalable diffusion models with transformers." Proceedings of the IEEE/CVF International Conference on Computer Vision. 2023.

WALT

- Learn a 3D CNN autoencoder
  - Downsamples over space and time
  - First frame is encoded as an image (no temporal downsampling) to support joint image-video representations
- Transformer diffusion model on latent space
  - Space-time factorized attention for better efficiency
  - Zero Terminal SNR
  - Different variant of AdaLN block
  - Latent self-conditioning [paper]
- 3B params

Gupta, Agrim, et al. "Photorealistic video generation with diffusion models." arXiv preprint arXiv:2312.06662 (2023).

WALT

A unicorn walking,
slow cinematic motion

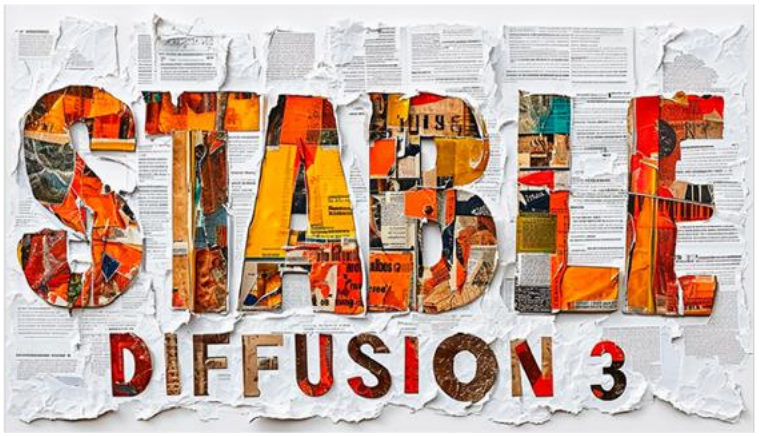
An astronaut feeding ducks
on a sunny afternoon

Sora

- Based on a scaled up diffusion transformer
  - Learn a VAE -> encode -> patchify -> flatten
  - Allows for more flexible learning of variable resolutions and video length
  - (Maybe?) trained with random conditioning to support video prediction and infilling

Stable Diffusion 3

- 8B parameters - largest open-source text-image model to date
- Diffusion transformer architecture
- Uses Flow Matching [paper]

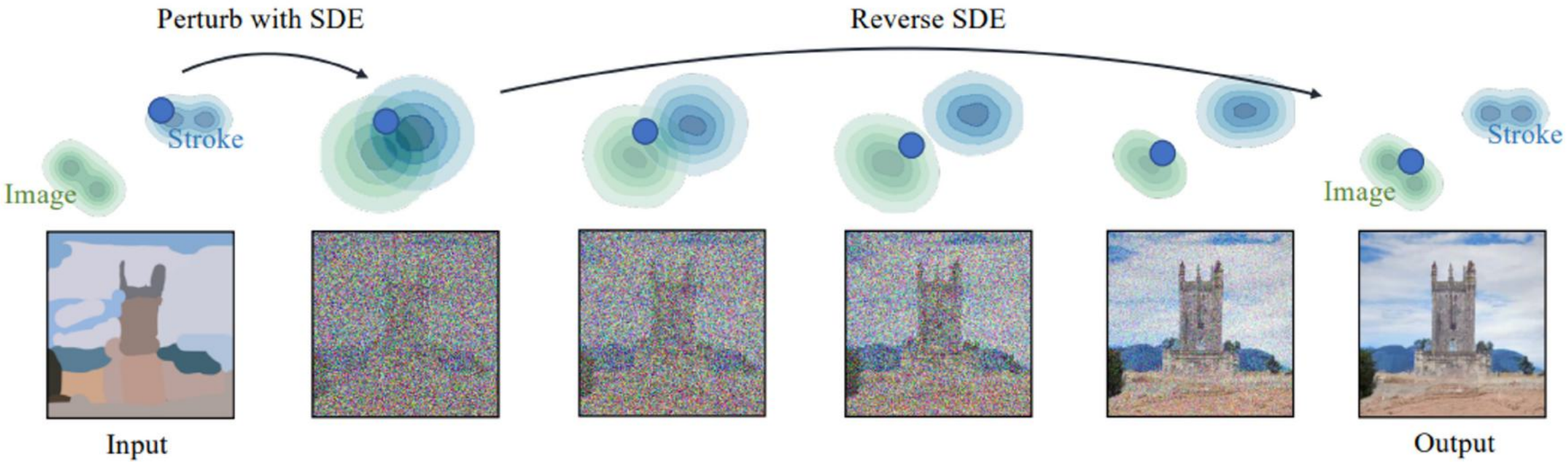


Lecture overview

- Motivation
- Energy-based models
- Score-based Models
- Denoising Diffusion Models
- **Deeper Dive into Diffusion Models**
  - DDIM Sampling
  - Prediction Space / Loss: x, eps, v
  - Issues with Common Noise Schedules
  - Architectures: UNet, latent space / stable diffusion, hierarchical generation, transformers
  - **Image Editing / Video Editing**
  - Using scores in diffusion models: text-to-3D, text-to-SVG, Video Adapter
  - Faster Sampling
  - Diffusion as Pre-Training
  - Other Fields: visuomotor control, materials discovery

SDEdit

Goal: High-level, editing of semantic image features while retaining global structure



Meng, Chenlin, et al. "Sdedit: Guided image synthesis and editing with stochastic differential equations." arXiv preprint arXiv:2108.01073 (2021).

SDEdit

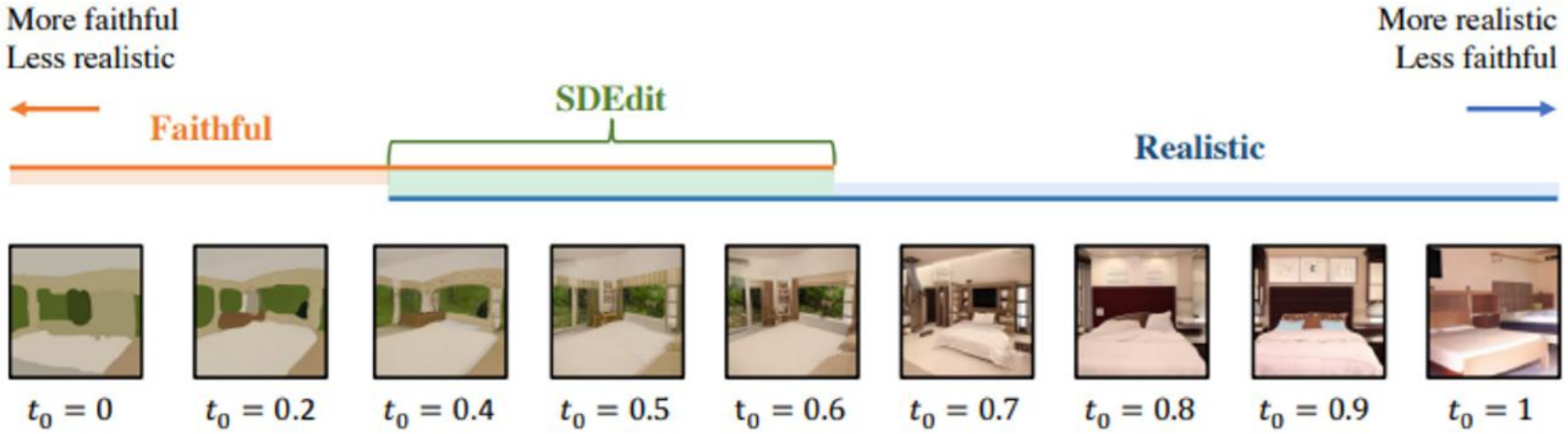
Given some diffusion timestep t, image x_0, edit caption c:

1)Apply the forward process to t: $q(x_t \mid x_0)$
2)Apply the reverse process to 0: $p(x_{t-1} \mid x_t, c)$ (t times)

SDEdit

T is a hyperparameter

- Trade-off between faithfulness to original image, and alignment with target edit



Using SDEdit

Source Video

Made of wooden blocks

Using SDEdit

Source Video

Make of colorful toy bricks

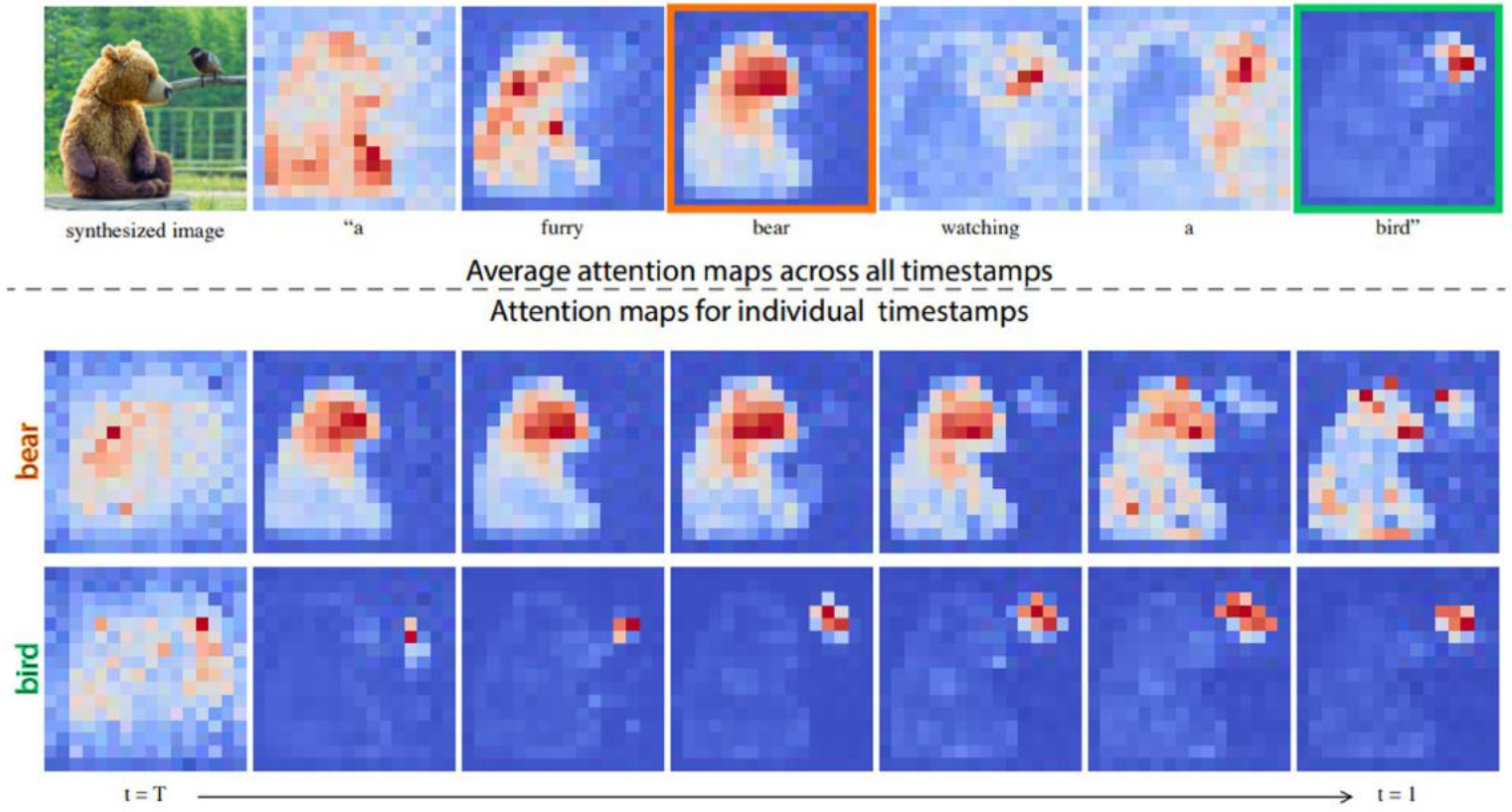
Prompt-to-Prompt (P2P)

Goal: Retain the fine-grained structure but change the content of an image

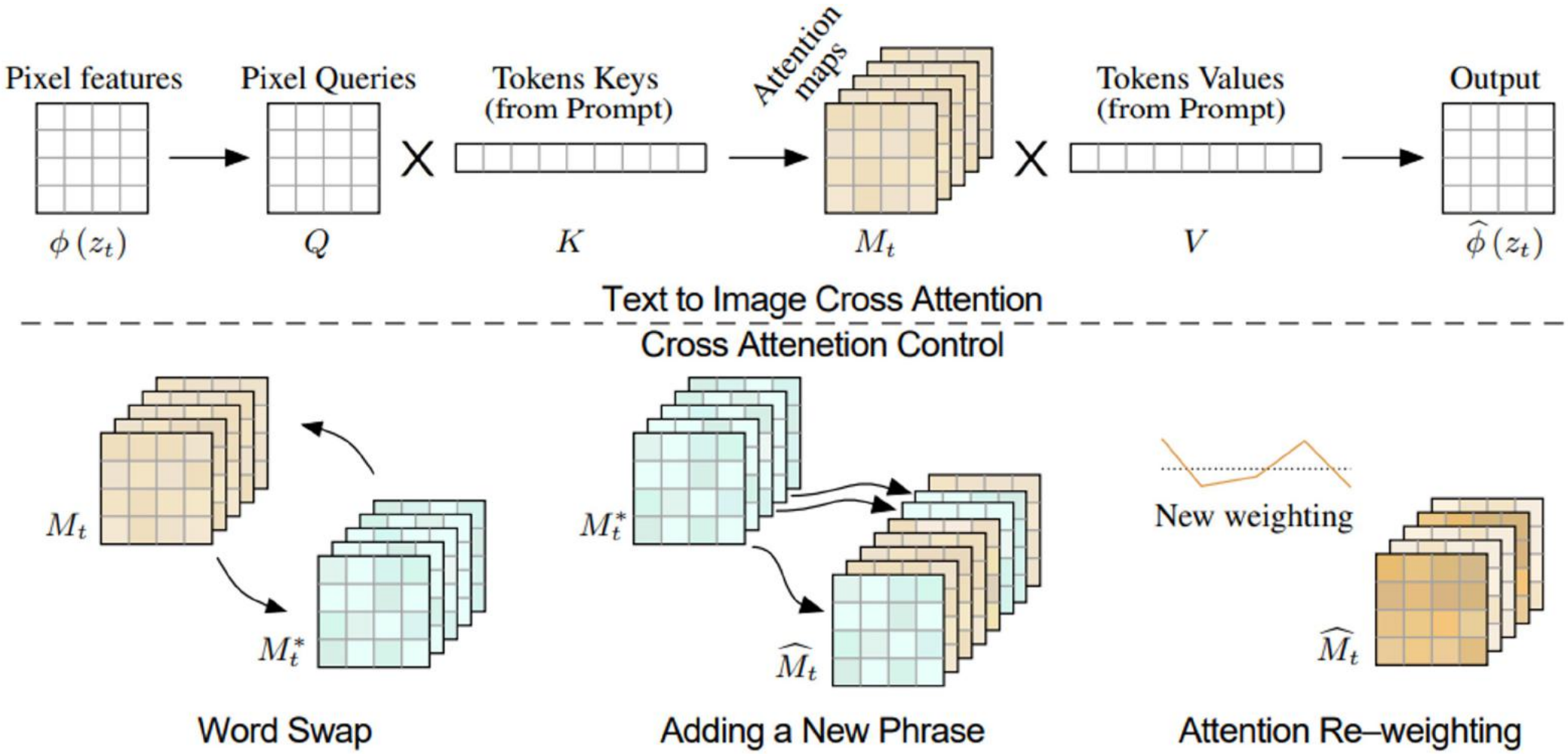
Hertz, Amir, et al. "Prompt-to-prompt image editing with cross attention control." arXiv preprint arXiv:2208.01626 (2022).

Prompt-to-Prompt (P2P)

Key Idea: Attention maps in UNets provide structure information

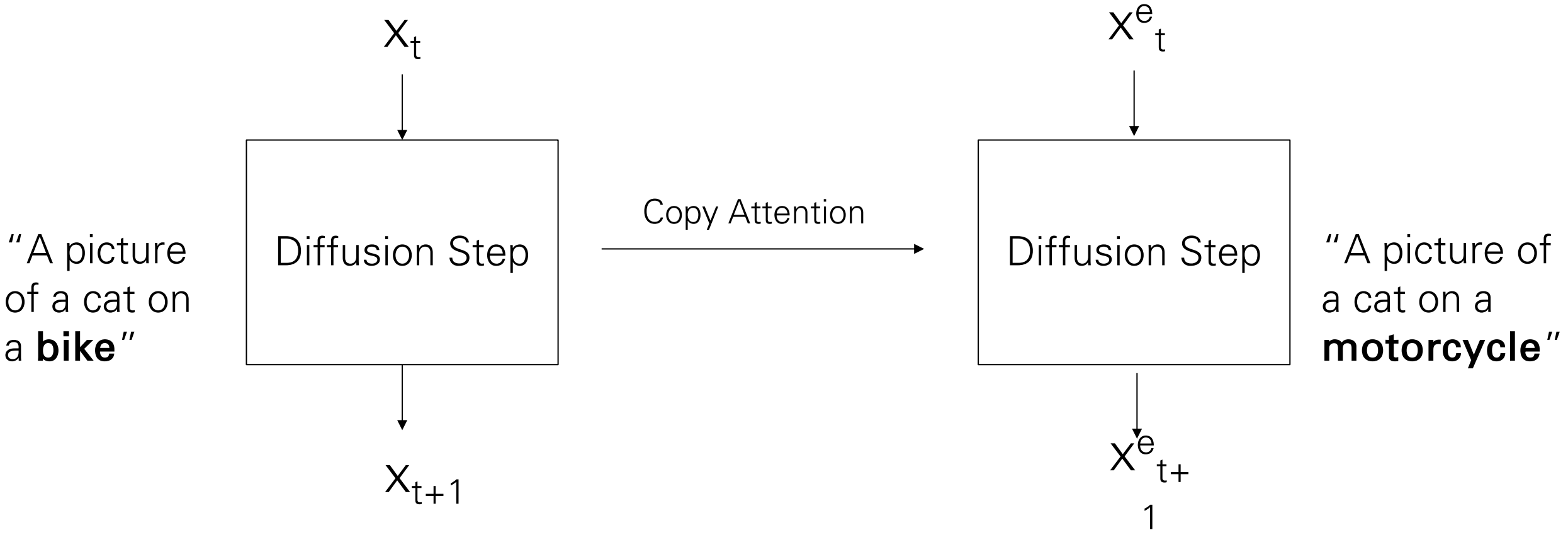


Prompt-to-Prompt (P2P)

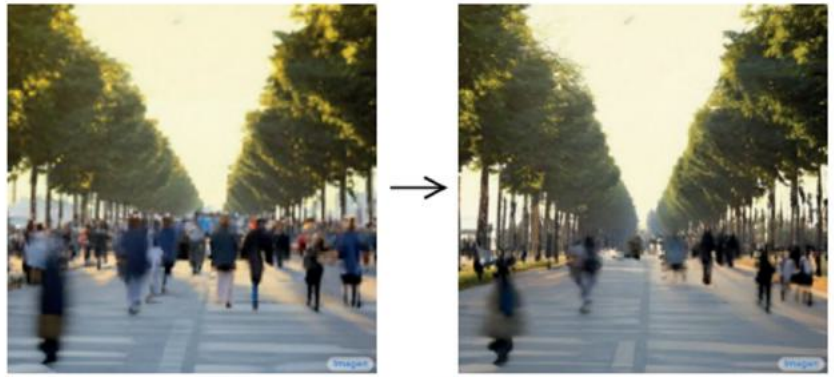


Sampling Process

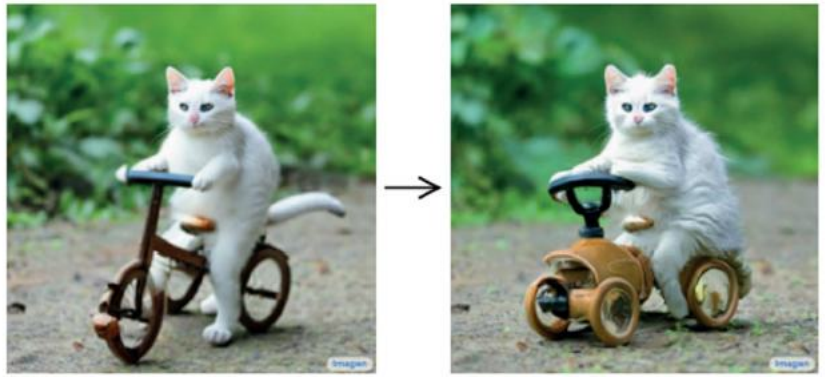
Sample with original and edited prompt in parallel, and copy attention maps



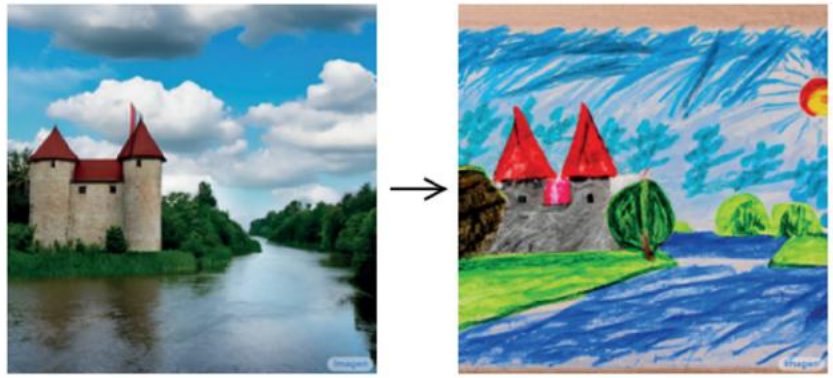
Prompt-to-Prompt (P2P)



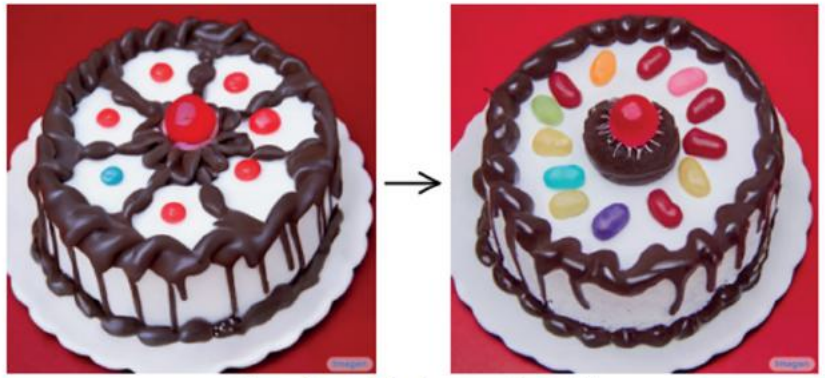
"The boulevards are crowded today."



"Photo of a cat riding on a bicycle."

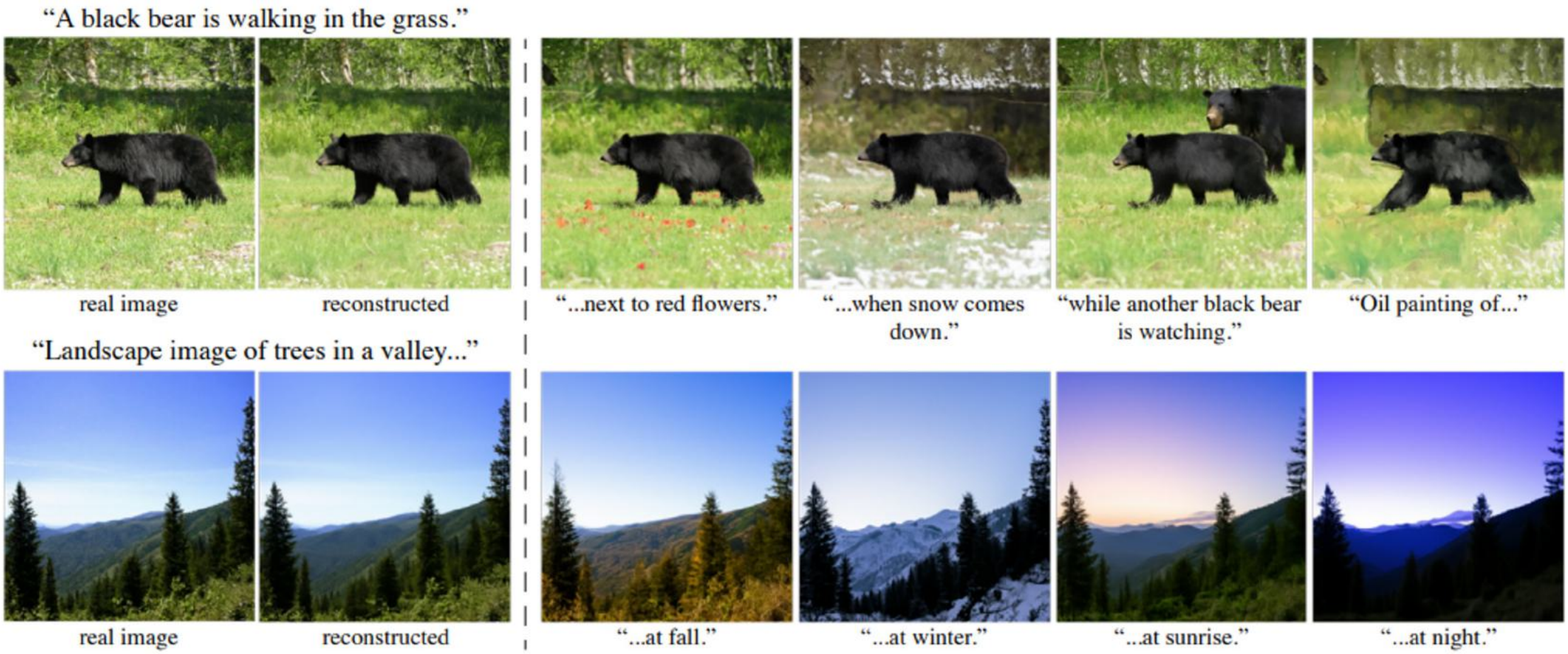


"Children drawing of a castle next to a river."



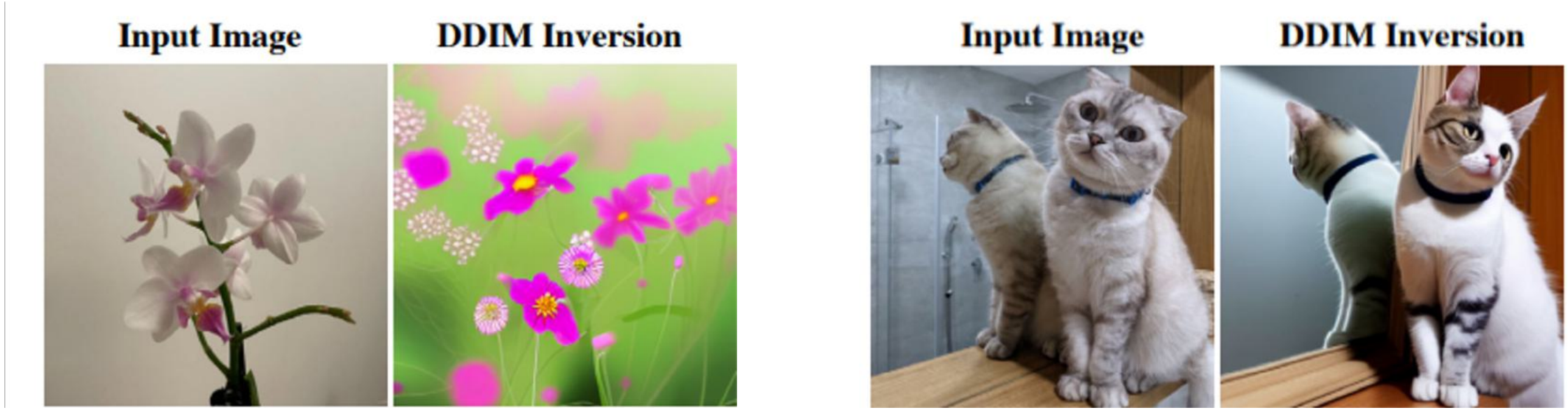
"a cake with decorations."

Real Image Editing



Failure Cases

Depends on DDIM reconstruction quality - especially bad with CFG



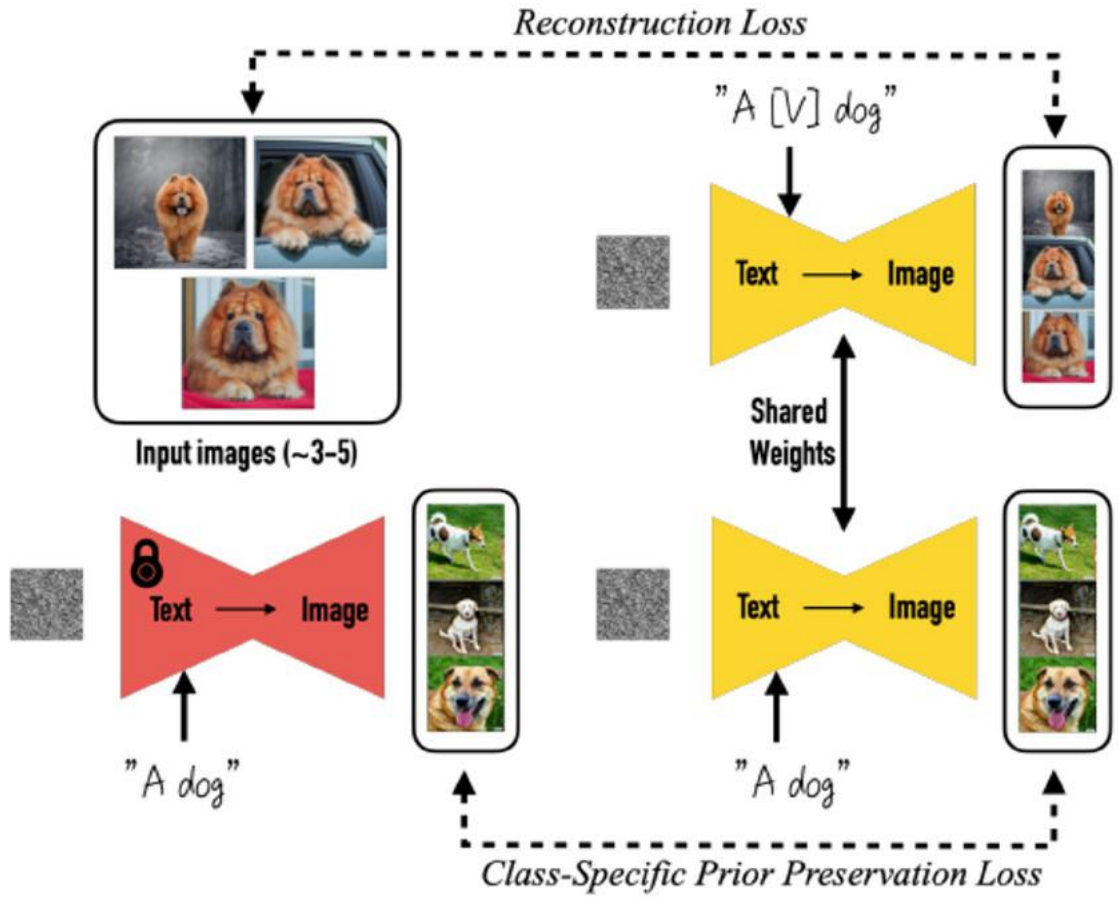
DreamBooth

Goal: Subject-driven image generation given 3-5 images of a specific subject

Ruiz, Nataniel, et al. "Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2023.

DreamBooth

$$\mathbb{E}_{\mathbf{x},\mathbf{c},\boldsymbol{\epsilon},\boldsymbol{\epsilon}',t}[w_t \|\hat{\mathbf{x}}_\theta(\alpha_t \mathbf{x} + \sigma_t \boldsymbol{\epsilon}, \mathbf{c}) - \mathbf{x}\|_2^2 + \lambda w_{t'} \|\hat{\mathbf{x}}_\theta(\alpha_{t'} \mathbf{x}_{\text{pr}} + \sigma_{t'} \boldsymbol{\epsilon}', \mathbf{c}_{\text{pr}}) - \mathbf{x}_{\text{pr}}\|_2^2],$$



DreamBooth


Input images
A [V] backpack in the Grand Canyon
A wet [V] backpack in water
A [V] backpack in Boston
A [V] backpack with the night sky
Input images
A [V] teapot floating in milk
A transparent [V] teapot with milk inside
A [V] teapot pouring tea
A [V] teapot floating in the sea

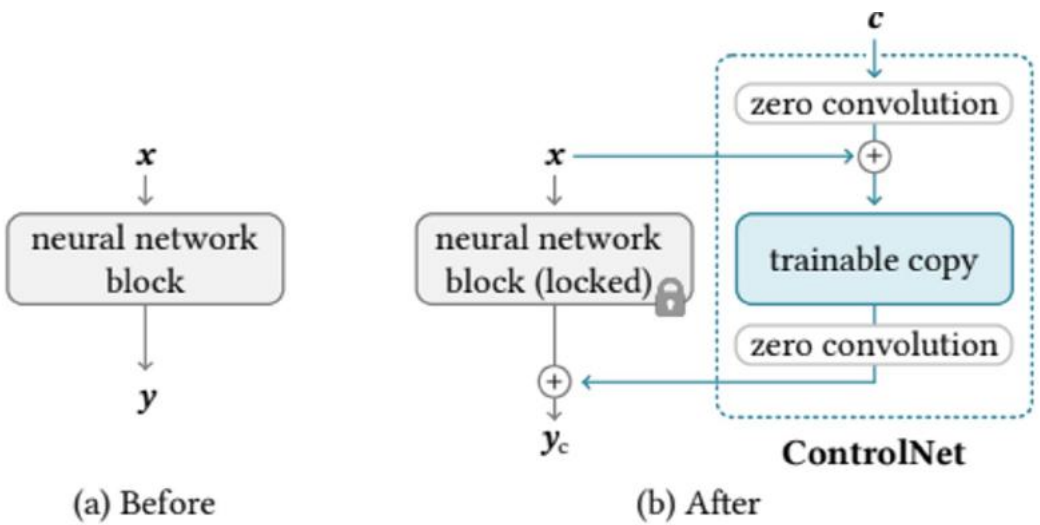
ControlNet

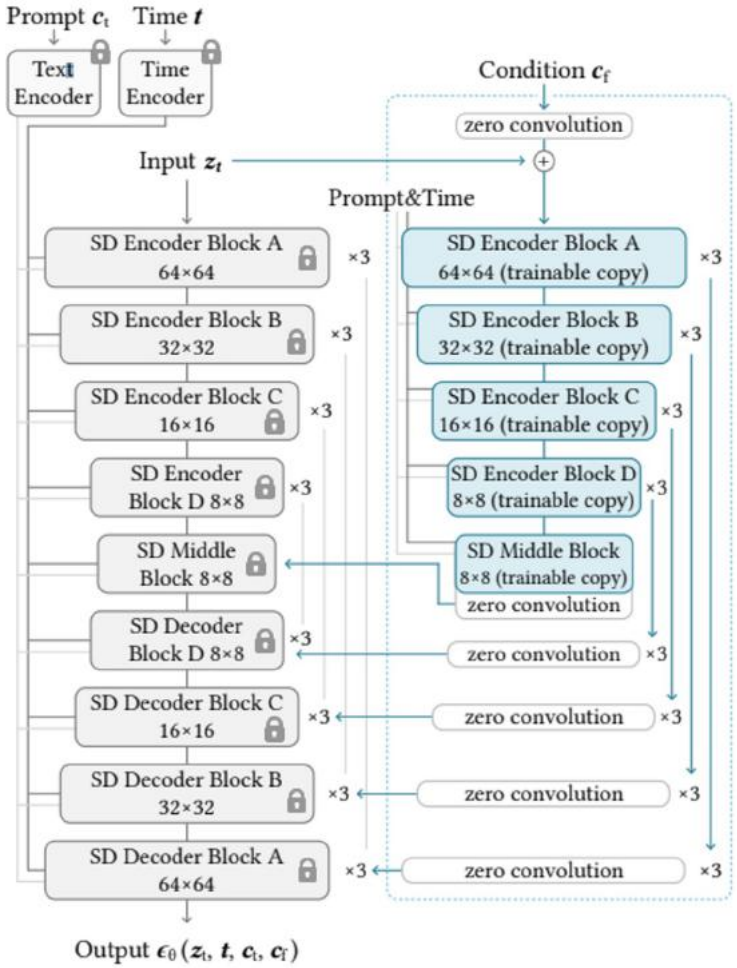
Goal: Finetune a diffusion model to include extra conditioning

- Canny edge map
- Depth
- Human Pose
- Lower Resolution Image
- Segmentation
- Sketch

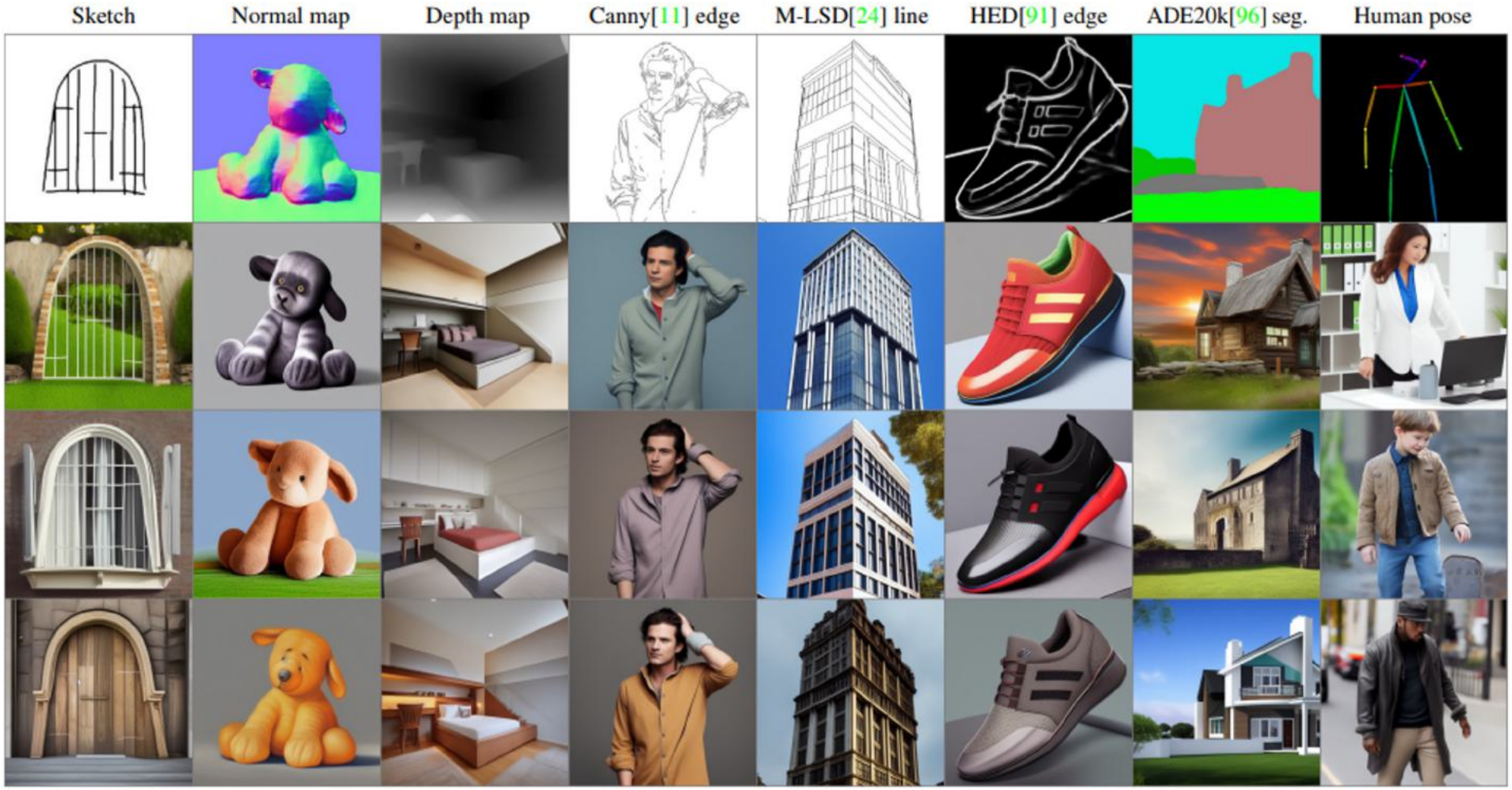
Zhang, Lvmin, Anyi Rao, and Maneesh Agrawala. "Adding conditional control to text-to-image diffusion models." Proceedings of the IEEE/CVF International Conference on Computer Vision. 2023.

ControlNet


x
neural network block
y
(a) Before
c
zero convolution
x
neural network block (locked)
trainable copy
zero convolution
y_c
ControlNet
(b) After


Prompt c_t
Time t
Text Encoder
Time Encoder
Condition c_f
zero convolution
Input z_t
Prompt&Time
SD Encoder Block A 64×64
SD Encoder Block A 64×64 (trainable copy)
SD Encoder Block B 32×32
SD Encoder Block B 32×32 (trainable copy)
SD Encoder Block C 16×16
SD Encoder Block C 16×16 (trainable copy)
SD Encoder Block D 8×8
SD Encoder Block D 8×8 (trainable copy)
SD Middle Block 8×8
SD Middle Block 8×8 (trainable copy)
zero convolution
SD Decoder Block D 8×8
zero convolution
SD Decoder Block C 16×16
zero convolution
SD Decoder Block B 32×32
zero convolution
SD Decoder Block A 64×64
zero convolution
×3
Output $\epsilon_\theta(z_t, t, c_t, c_f)$

ControlNet


Sketch
Normal map
Depth map
Canny[11] edge
M-LSD[24] line
HED[91] edge
ADE20k[96] seg.
Human pose

Tune-a-Video

Goal: Edit videos using a text-image model

1)Start with a pretrained text-image model (e.g. SD)

Wu, Jay Zhangjie, et al. "Tune-a-video: One-shot tuning of image diffusion models for text-to-video generation." Proceedings of the IEEE/CVF International Conference on Computer Vision. 2023.

Tune-a-Video

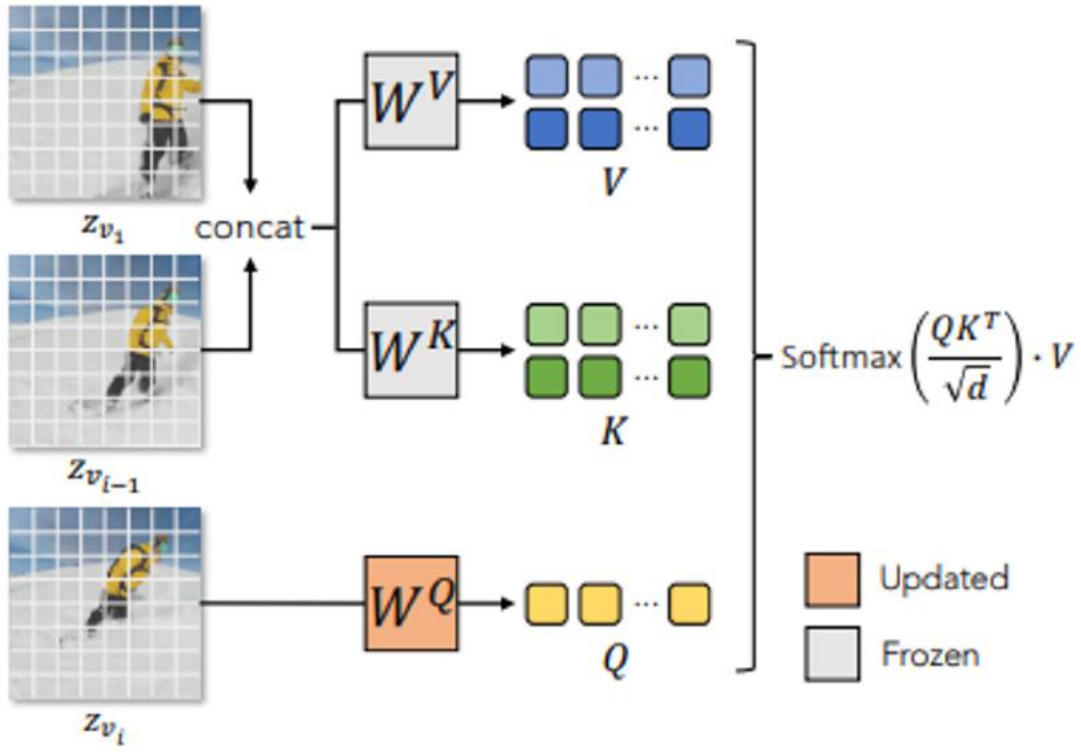
2) Inflate the network to process videos

- k x k 2D convs become 1 x k x k 3D convs
- Self-attention for each frame additional attends to past 2 frames

Tune-a-Video

3) Finetune the model on the standard diffusion loss

- Finetune **on a single video**
- Freeze all parameters except **self-attention query projection**



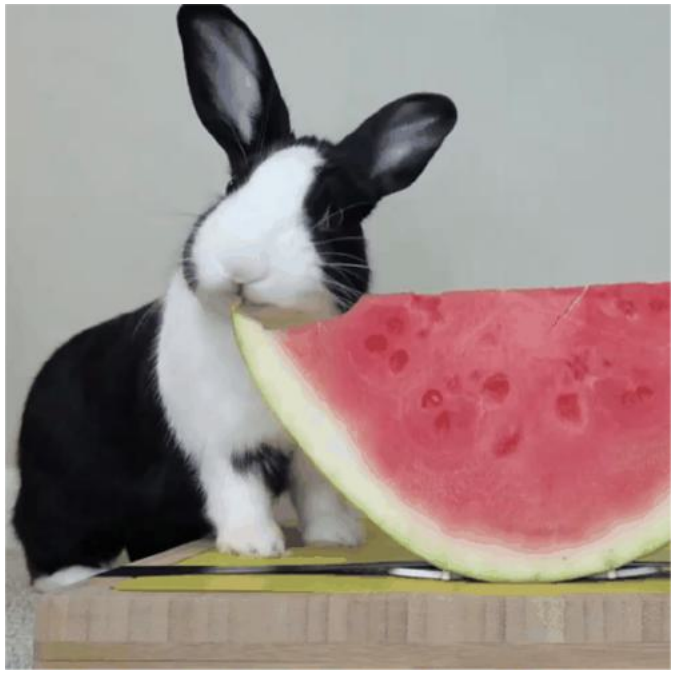
Tune-a-Video

Inference

- Compute the DDIM inversion of the source video
- Run diffusion sampling from the DDIM inversion with the target edit caption

Tune-a-Video

A rabbit is eating a watermelon on a table



A rabbit is on the table

A cat with sunglasses is eating a watermelon on a beach

Dreamix

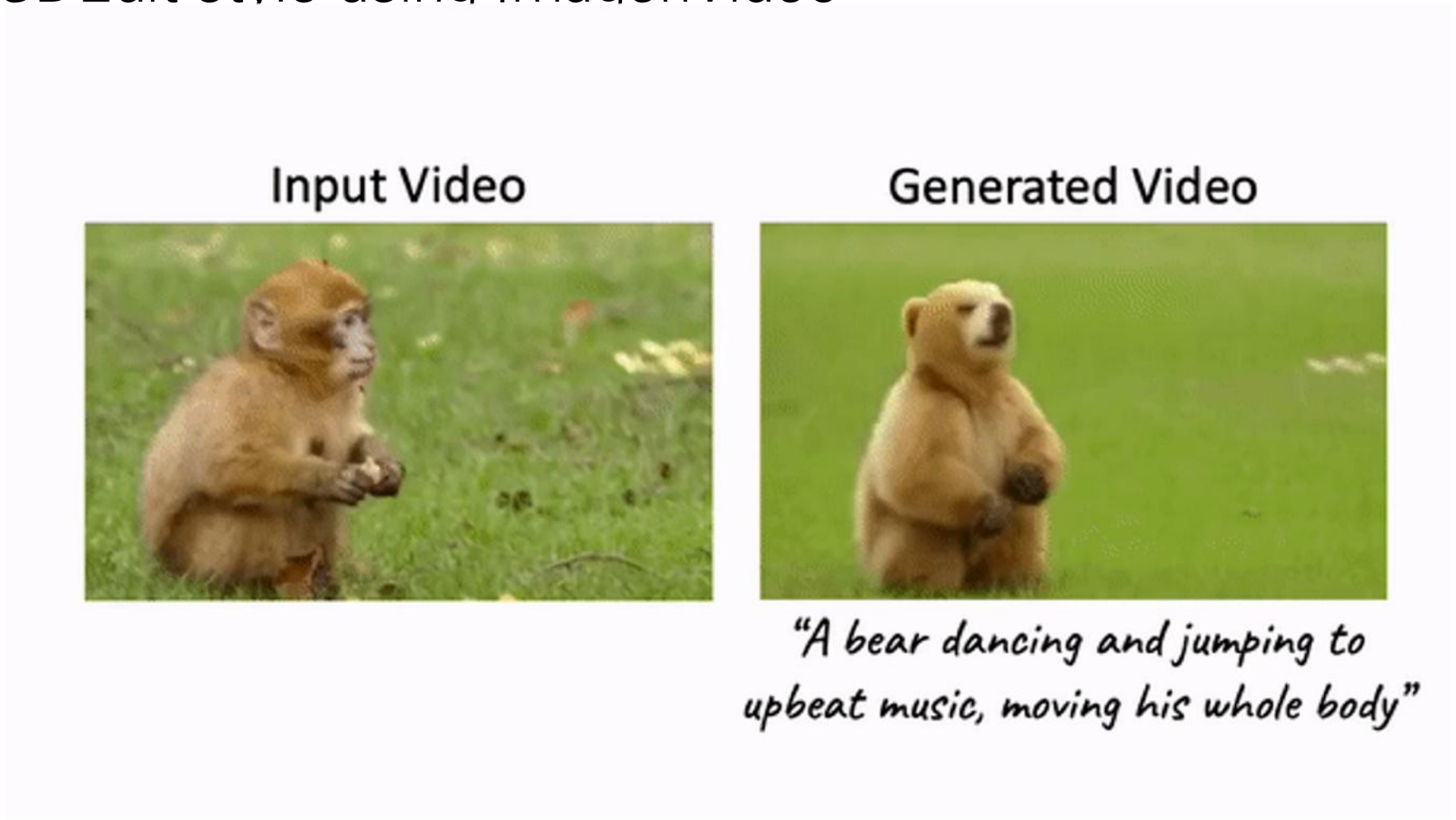
How about more general video editing?

Use a pretrained text-video model instead of text-image

Molad, Eyal, et al. "Dreamix: Video diffusion models are general video editors." arXiv preprint arXiv:2302.01329 (2023).

Dreamix

Can do SDEdit-style using ImagenVideo

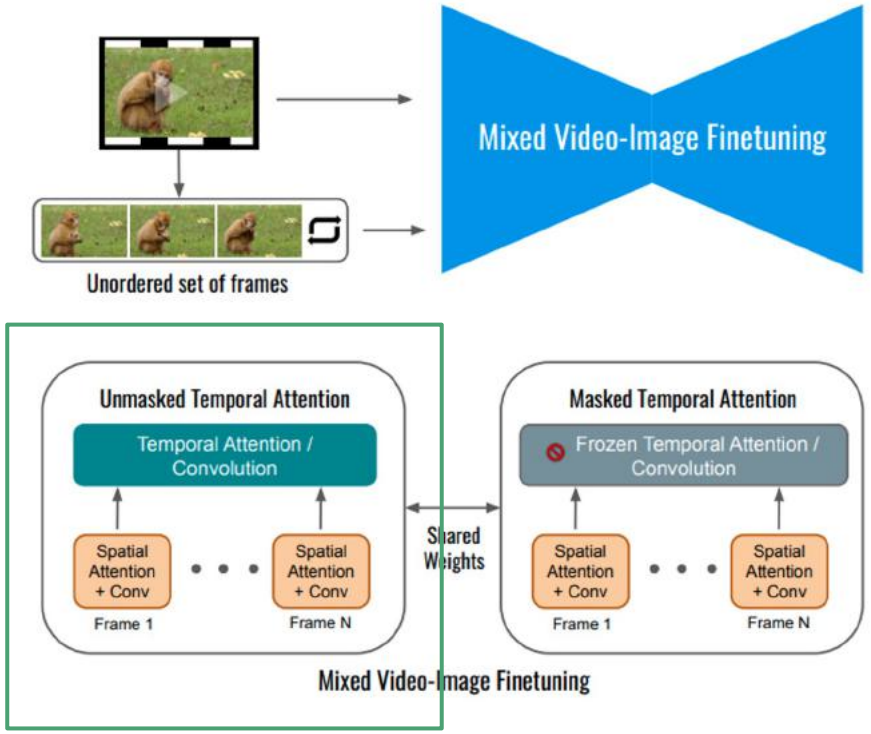


Dreamix

Mixed Video-Image Finetuning (Video)

- Reconstruct the original video **v** conditioned on noised version $\mathbf{z_s}$ and rare token **t***

$$\mathcal{L}_\theta^{vid}(v) = \mathbb{E}_{\epsilon \sim N(0,\mathbf{I}), s \in \mathcal{U}(0,1)} \| D_{\theta'}(z_s, s, t^*, c) - v \|^2$$



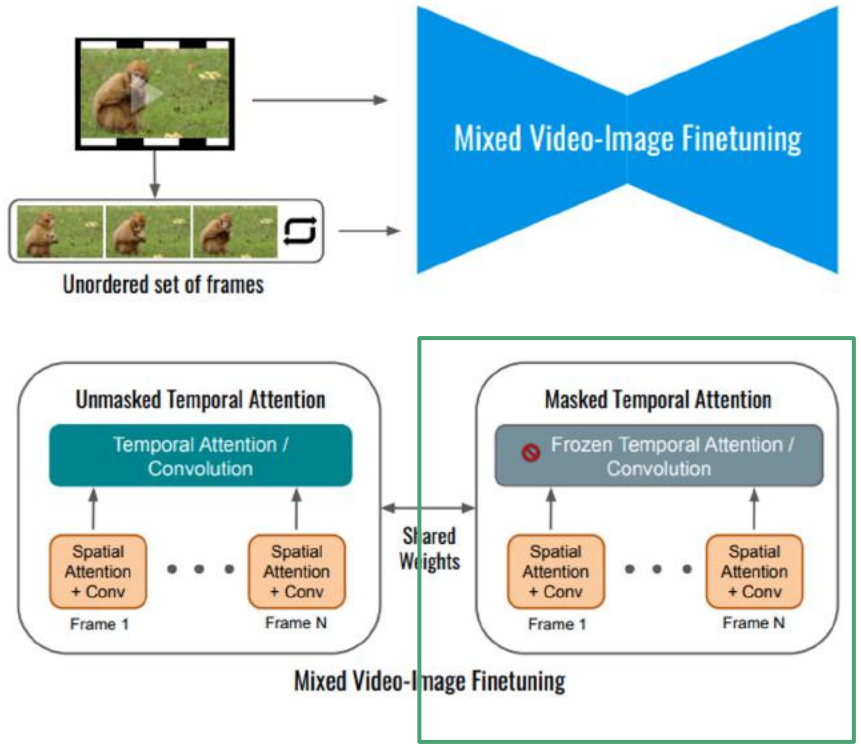
Dreamix

Mixed Video-Image Finetuning (Image)

- Reconstruct the original video **v** conditioned on noised version $\mathbf{z_s}$ and rare token **t***

Mask out any temporal operations for only frame-based denoising

$$\mathcal{L}_\theta^{frame}(u) = \mathbb{E}_{\epsilon\sim N(0,\mathbf{I}),s\in\mathcal{U}(0,1)}\|D_{\theta'}^a(z_s, s, t^*, c) - u\|^2$$

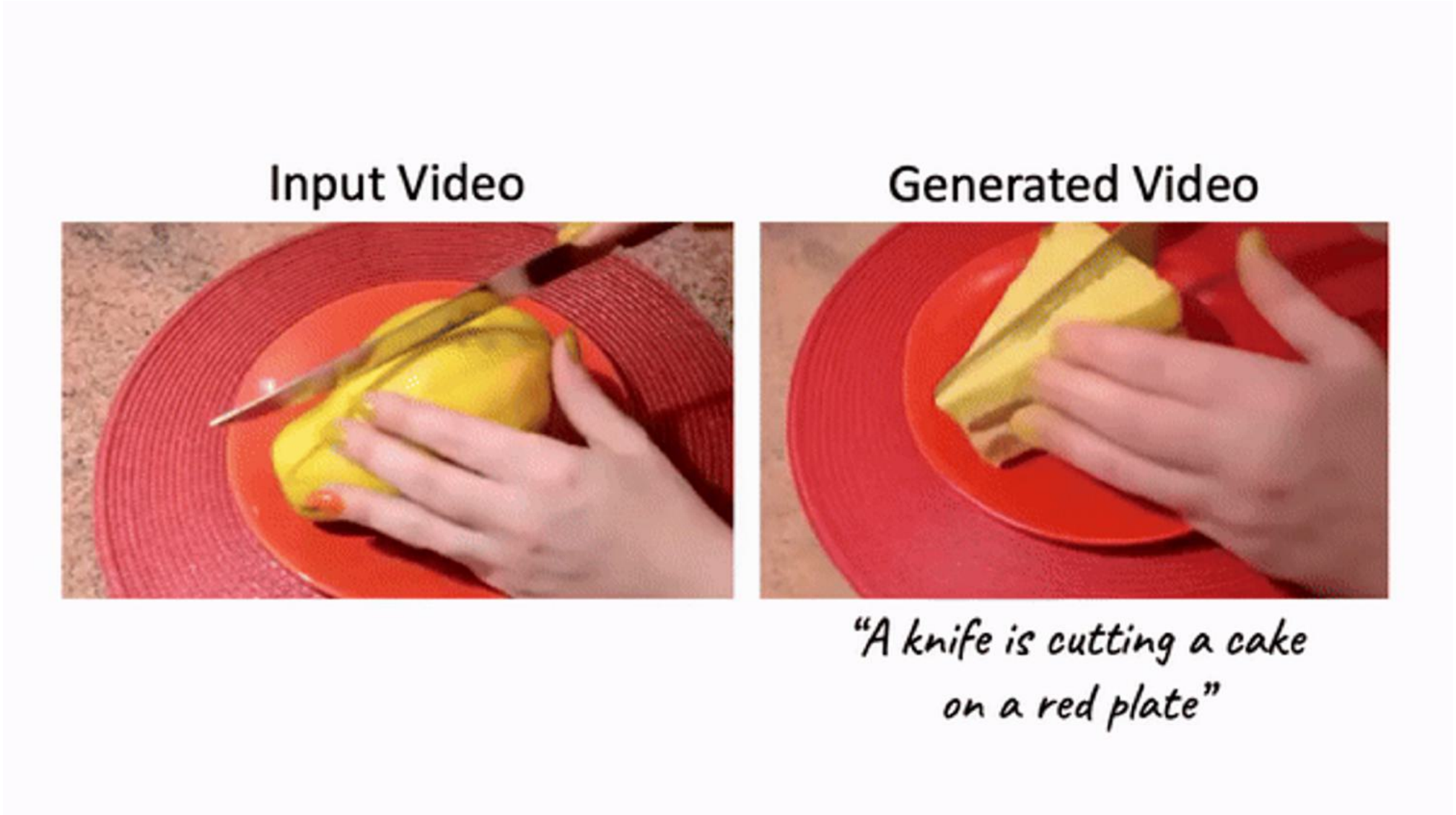


Dreamix

Final Fine-tuning Loss

$$\theta = arg \min_{\theta'} \alpha \mathcal{L}_{\theta'}^{vid}(v) + (1-\alpha)\mathcal{L}_{\theta'}^{frame}(u)$$

Dreamix


Input Video
Generated Video
"A knife is cutting a cake on a red plate"

Dreamix

Input Images

Generated Video

"A toy fireman is lifting weights"

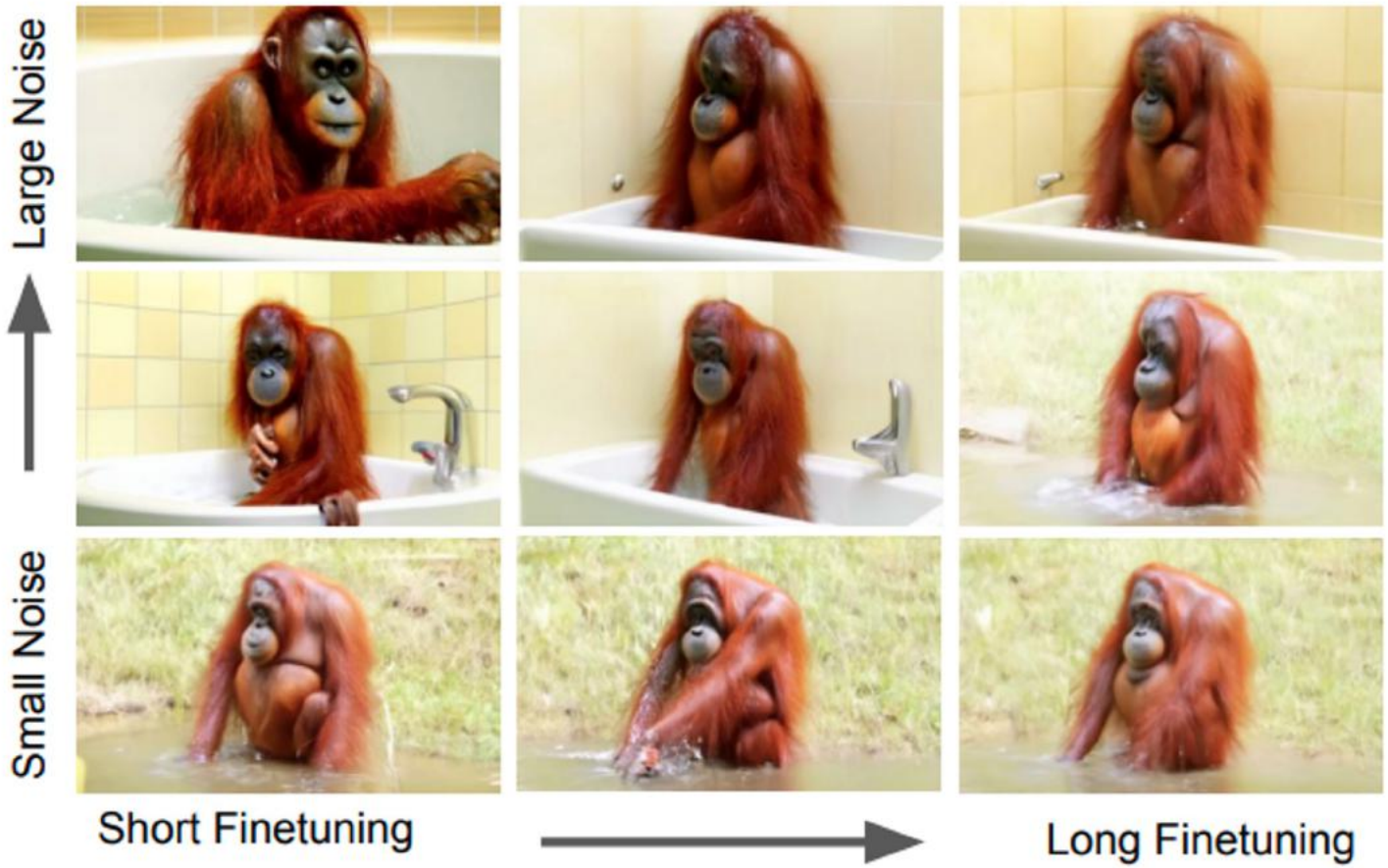
Dreamix

Input Video

Generated Video

"An orangutan with an orange hair waving hello next to a pond"

Dreamix

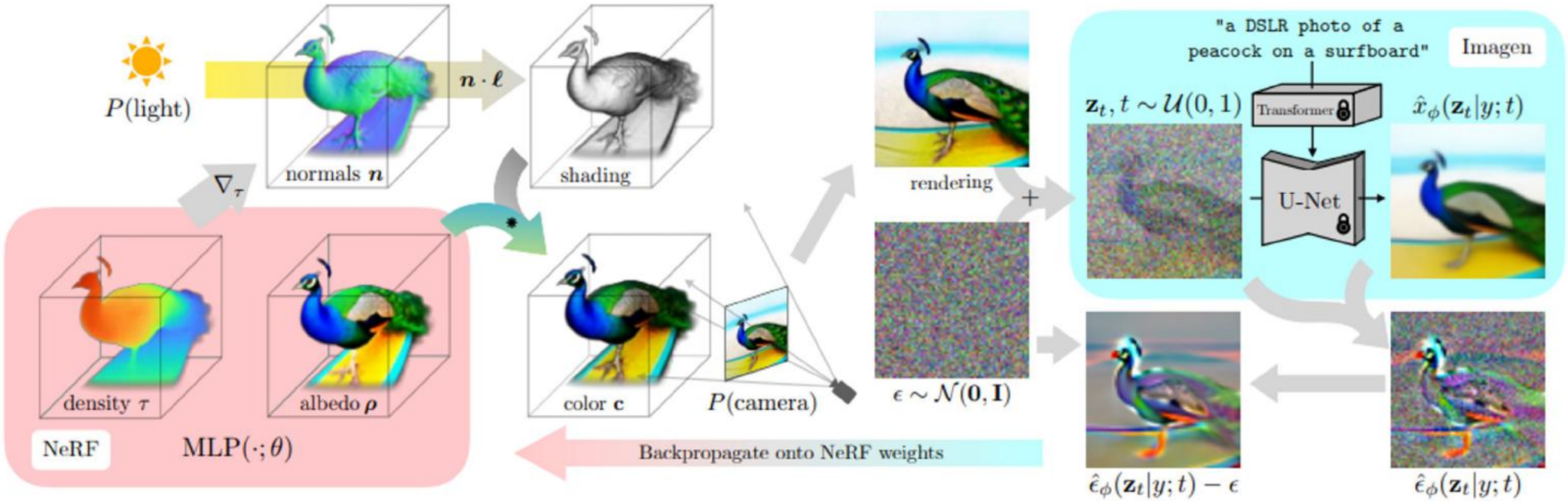


Lecture overview

- Motivation
- Energy-based models
- Score-based Models
- Denoising Diffusion Models
- **Deeper Dive into Diffusion Models**
  - DDIM Sampling
  - Prediction Space / Loss: x, eps, v
  - Issues with Common Noise Schedules
  - Architectures: UNet, latent space / stable diffusion, hierarchical generation, transformers
  - Image Editing / Video Editing
  - **Using scores in diffusion models: text-to-3D, text-to-SVG, Video Adapter**
  - Faster Sampling
  - Diffusion as Pre-Training
  - Other Fields: visuomotor control, materials discovery

DreamFusion

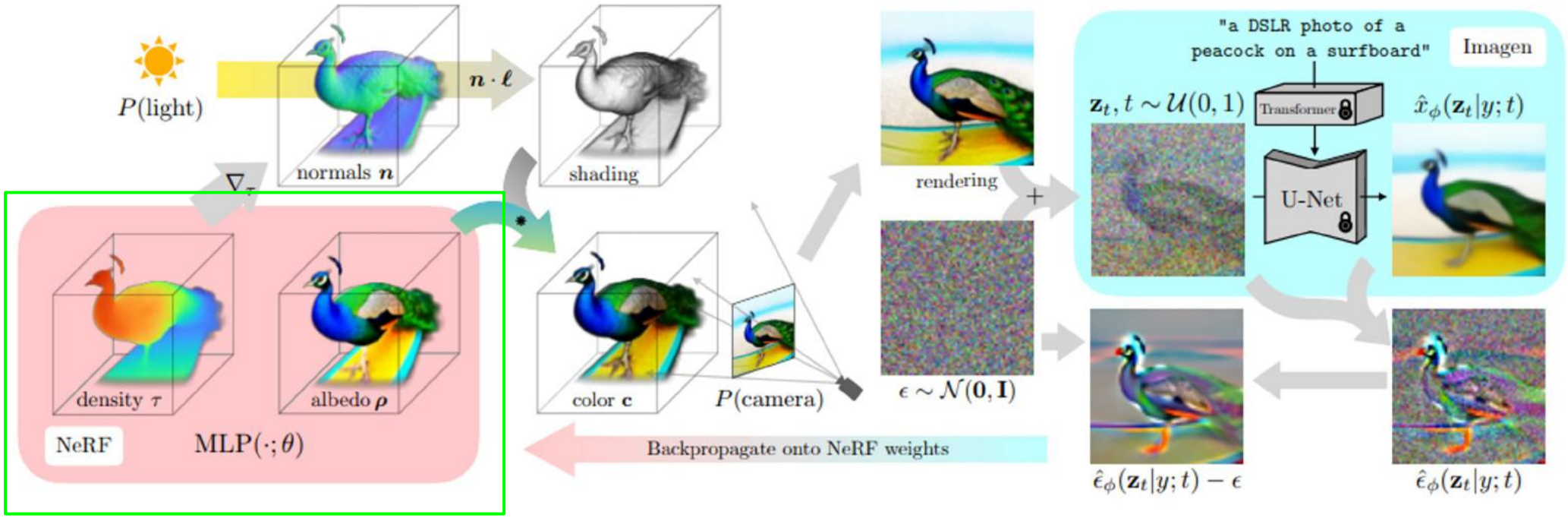
How can you use a pretrained text-image model to generate 3D objects?



Poole, Ben, et al. "Dreamfusion: Text-to-3d using 2d diffusion." arXiv preprint arXiv:2209.14988 (2022).

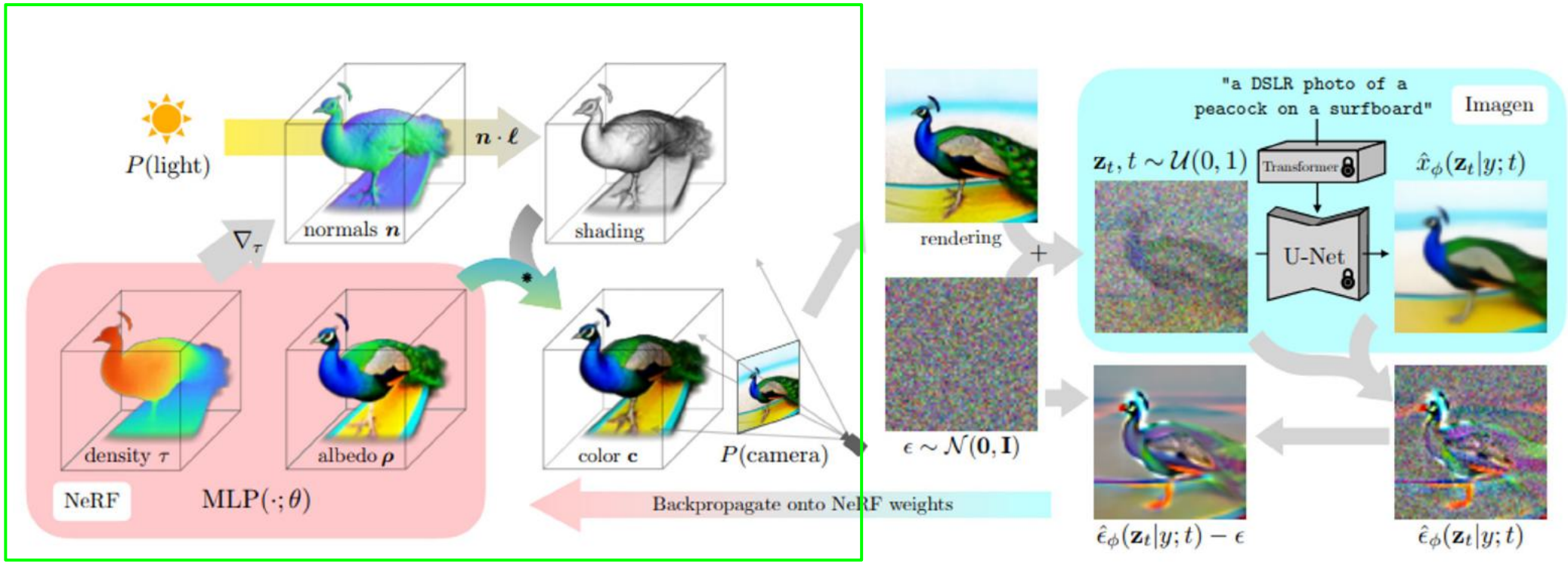
DreamFusion

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



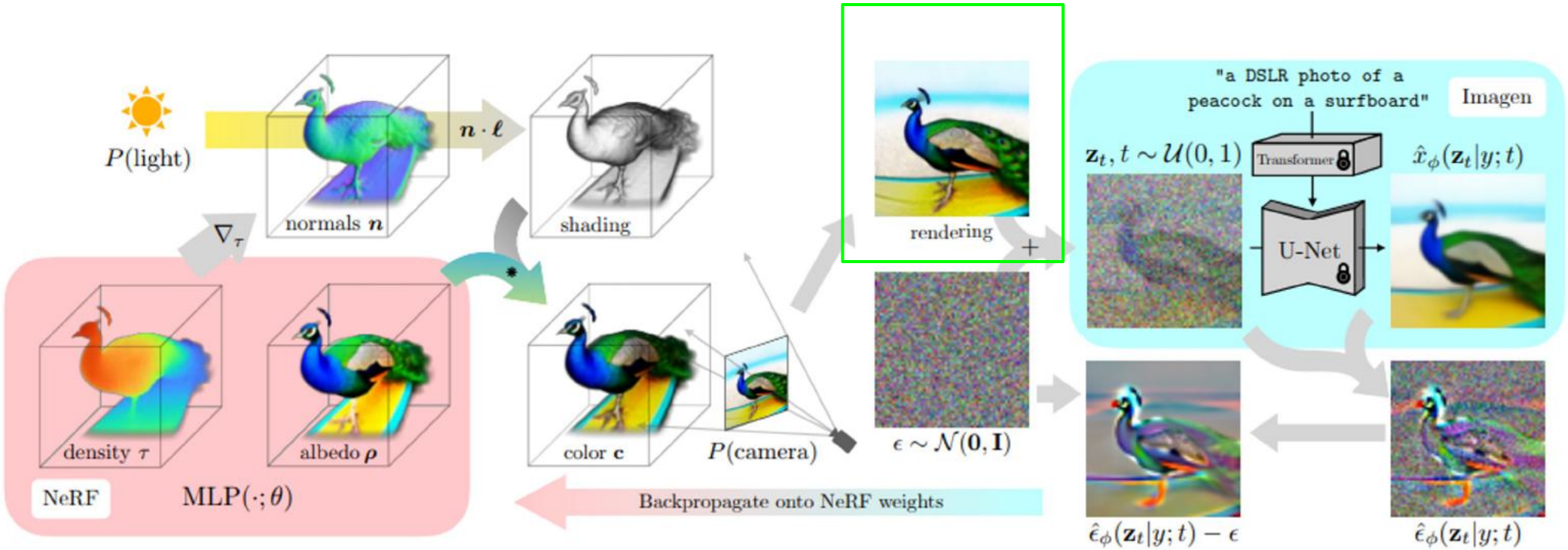
DreamFusion

Randomly sample camera angle and lighting



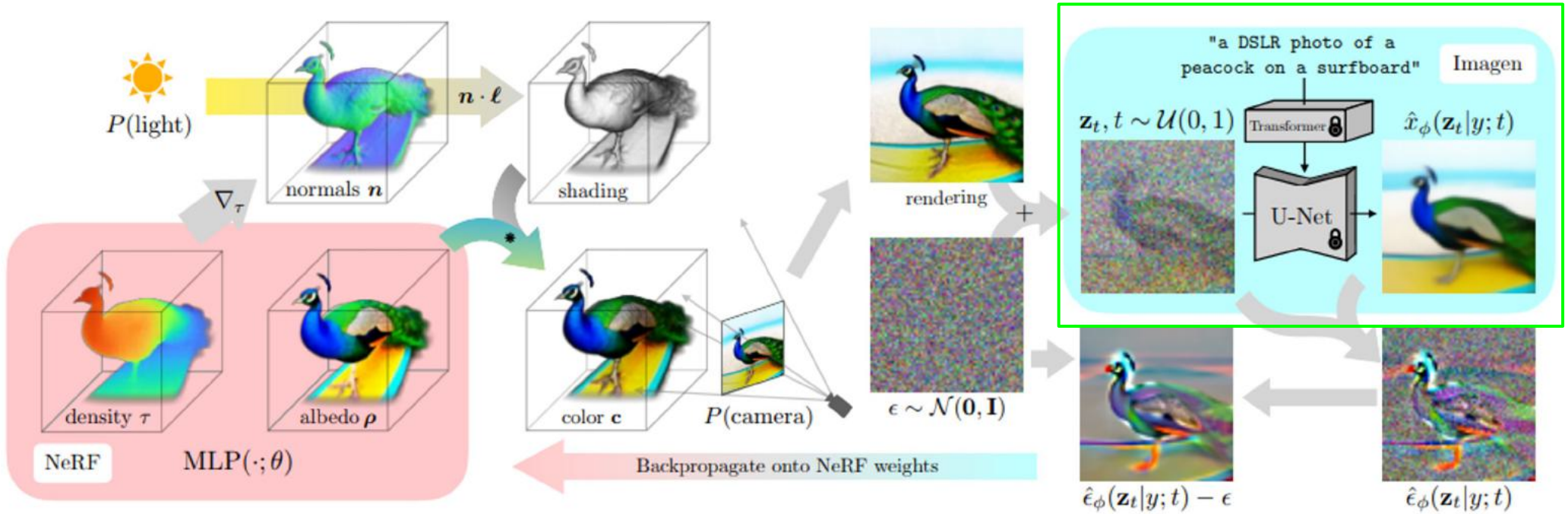
DreamFusion

Render the image (64 x 64)



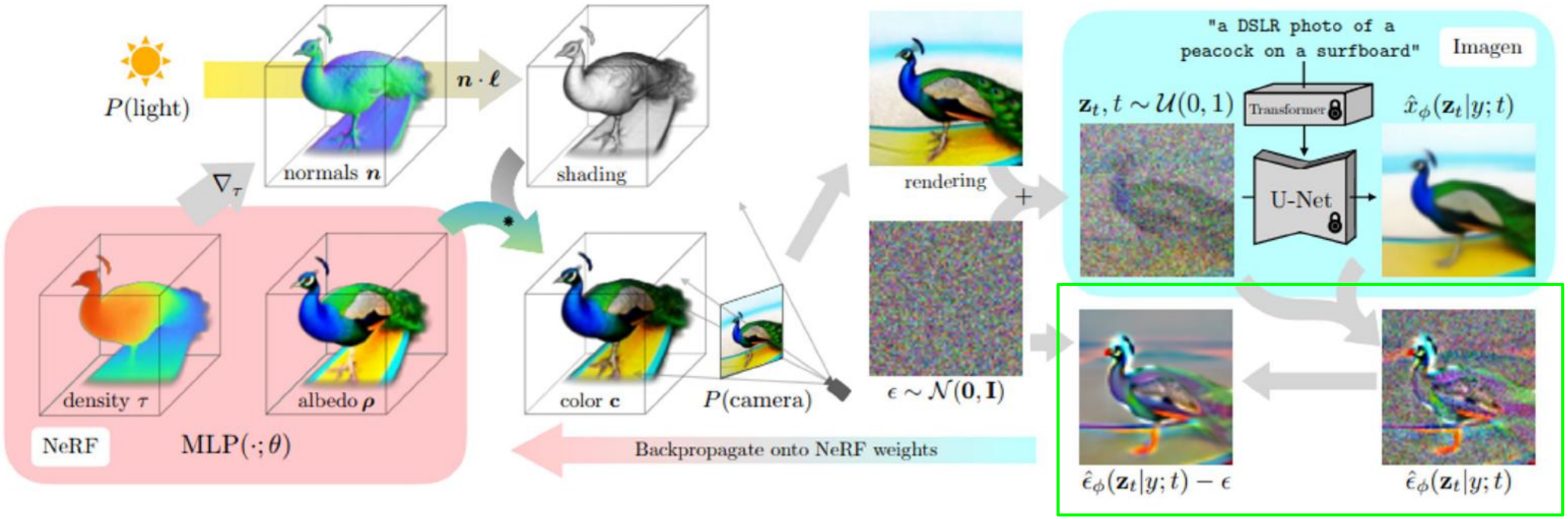
DreamFusion

Noise the image and feed it through the diffusion loss 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

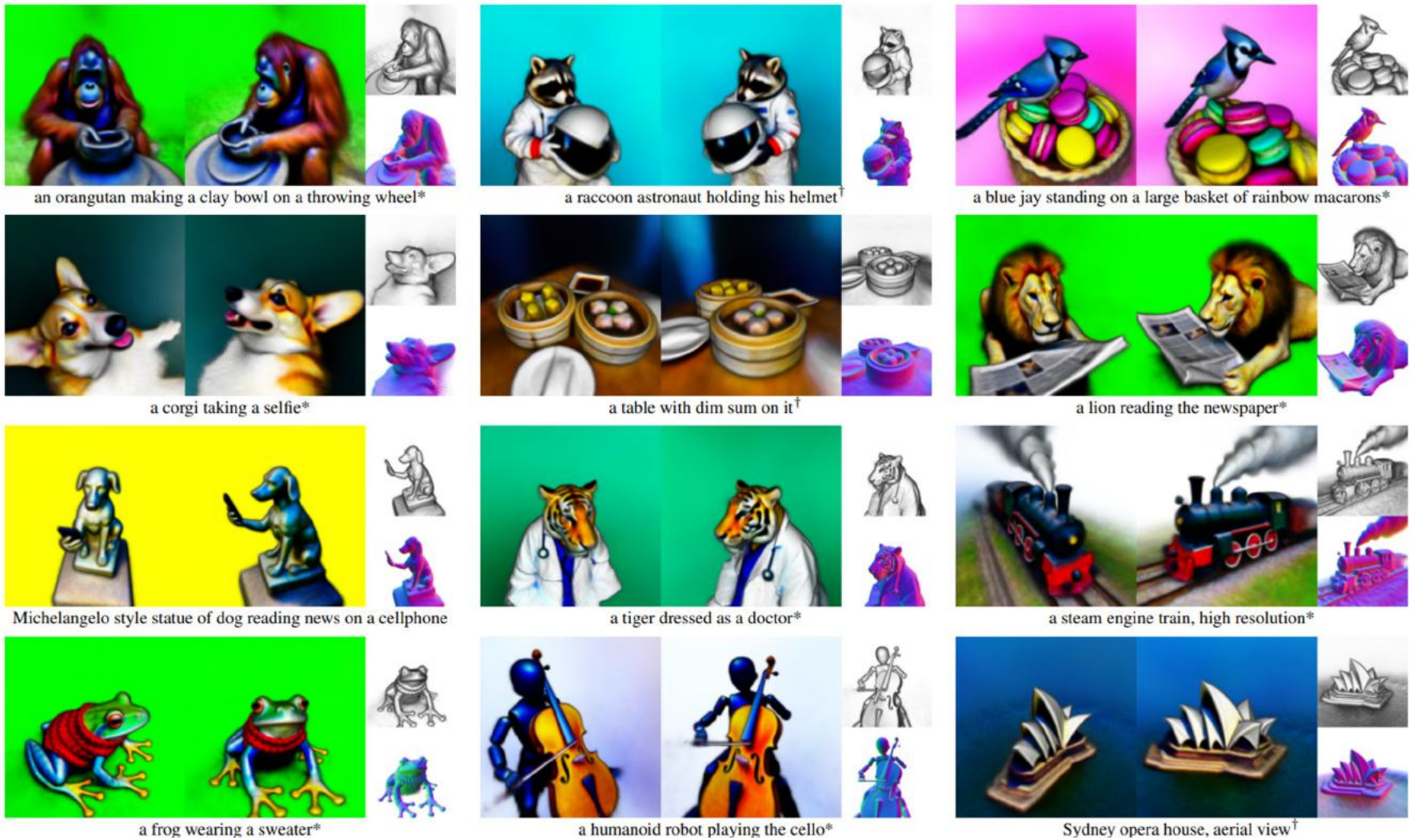
It works better to just remove the U-Net jacobian

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

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

There is a theoretical motivation (see Appendix in paper)

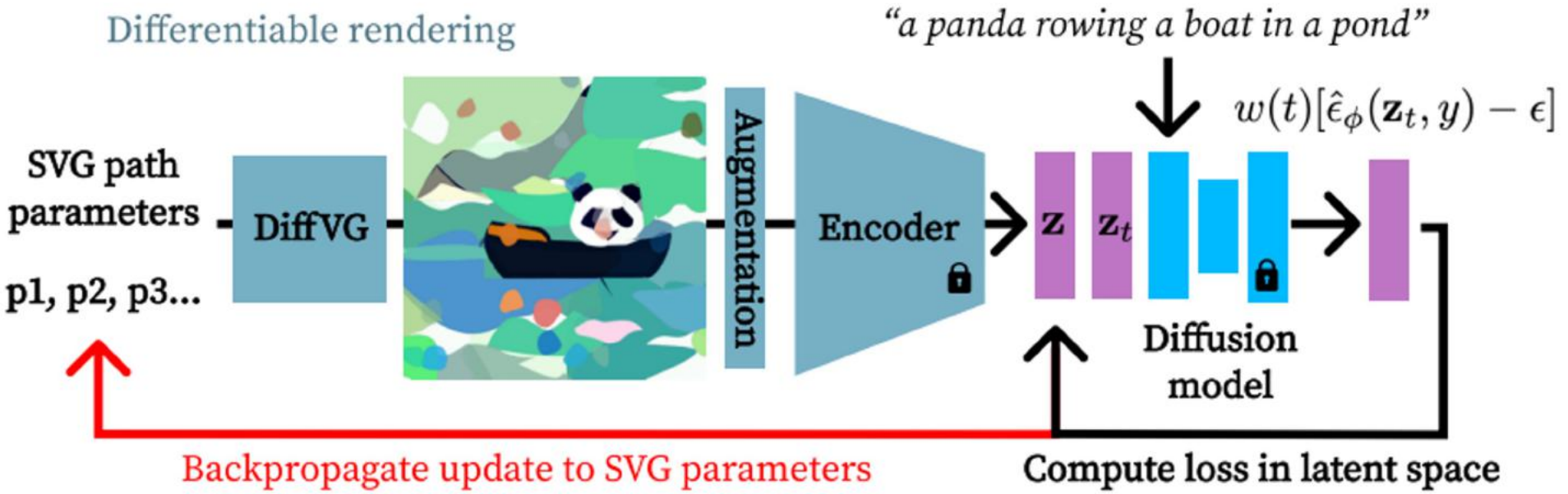
DreamFusion



Text-to-SVG

Using SDS loss is not limited to text-to-3D

- SDS provides a general way to backprop / learn through any differential "rendering" pipeline



Jain, Ajay, Amber Xie, and Pieter Abbeel. "Vectorfusion: Text-to-svg by abstracting pixel-based diffusion models." Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2023.

Text-to-SVG

a train*

an owl standing on a wire*

Underwater Submarine*

a boat*

A photo of a Ming Dynasty vase on a leather topped table.*

A smiling sloth wearing a leather jacket, a cowboy hat and a kilt.*

a tuba with red flowers protruding from its bell*

a blue poison dart frog sitting on a water lily*

VideoAdapter

DDPMs as Energy-Based Models (EBMs)

$$p_\theta(\tau) \propto e^{-E_\theta(\tau)}$$

Since the denoising function (epsilon) models the score, we have

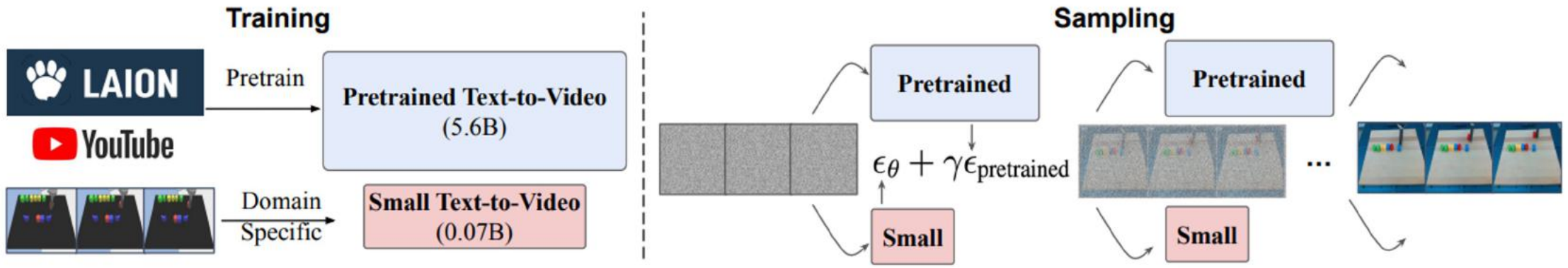
$$\epsilon_\theta(\tau^t, t) = \nabla_\tau E_\theta(\tau^t)$$

Yang, Mengjiao, et al. "Probabilistic Adaptation of Text-to-Video Models." arXiv preprint arXiv:2306.01872 (2023).

VideoAdapter

Learn a distribution as a product of a pretrained (large) model and domain specific (small) model

$$\underbrace{p_{\text{product}}(\tau|\text{text})}_{\text{Product Distribution}} \propto \underbrace{p_{\text{pretrained}}(\tau|\text{text})}_{\text{Pretrained Prior}} \underbrace{p_\theta(\tau|\text{text})}_{\text{Video Model}}.$$



VideoAdapter

Under the product assumption, we can use the EBM interpretation to combine the scores of the diffusion models

$$p_{product}(\tau|\text{text}) \propto e^{-E_1(\tau)} e^{-E_2(\tau)} = e^{-(E_1(\tau)+E_2(\tau))}$$

$$\nabla_\tau E_\theta(\tau) = -(\nabla E_1(\tau) + \nabla E_2(\tau))$$

VideoAdapter

Algorithm 1 Sampling algorithm of Video Adapter

Input: Pretrained Text-to-Video Model $\epsilon_{\text{pretrained}}(\tau, t|\text{text})$, Inverse Temperature α, Prior strength γ.
Initialize sample $\tau_T \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
for $t = T, \ldots, 1$ **do**
  $\tilde{\epsilon}_{\text{text}} \leftarrow \epsilon_\theta(\tau_t, t|\text{text}) + \gamma\epsilon_{\text{pretrained}}(\tau_t, t|\text{text})$ // compute score using text-conditioned prior
  $\epsilon \leftarrow \epsilon_\theta(\tau_t, t)$ // compute unconditional score
  $\tilde{\epsilon}_{\text{cfg}} \leftarrow \epsilon + \alpha(\tilde{\epsilon}_{\text{text}} - \epsilon)$ // compute classifier guidance score score
  $\tau_{t-1} = \texttt{ddpm\_sample}(\tau_t, \tilde{\epsilon}_{\text{cfg}})$ // run diffusion sampling (can use other samplers)
end for

VideoAdapter

	Bridge			Ego4D		
Model	FVD ↓	FID ↓	Param (B)↓	FVD ↓	IS ↑	Param (B) ↓
Small (S)	186.8	38.8	0.07	228.3	2.28	0.07
Small (S) + Pretrained	**177.4**	**37.6**	0.07	156.3	2.82	0.07
Small (L)	152.5	30.1	0.14	65.1	3.31	2.8
Small (L) + Pretrained	**148.1**	**29.5**	0.14	**52.5**	**3.53**	2.8
Pretrained	350.1	42.6	5.6	91.7	3.12	5.6

VideoAdapter

Can adapt to different domains (robot trajectories, ego-centric camera)

Lecture overview

- Motivation
- Energy-based models
- Score-based Models
- Denoising Diffusion Models
- **Deeper Dive into Diffusion Models**
  - DDIM Sampling
  - Prediction Space / Loss: x, eps, v
  - Issues with Common Noise Schedules
  - Architectures: UNet, latent space / stable diffusion, hierarchical generation, transformers
  - Image Editing / Video Editing
  - Using scores in diffusion models: text-to-3D, text-to-SVG, Video Adapter
  - **Faster Sampling**
  - Diffusion as Pre-Training
  - Other Fields: visuomotor control, materials discovery

Progressive Distillation

PROGRESSIVE DISTILLATION FOR FAST SAMPLING OF DIFFUSION MODELS

Tim Salimans & Jonathan Ho
Google Research, Brain team
{salimans,jonathanho}@google.com

Progressive Distillation

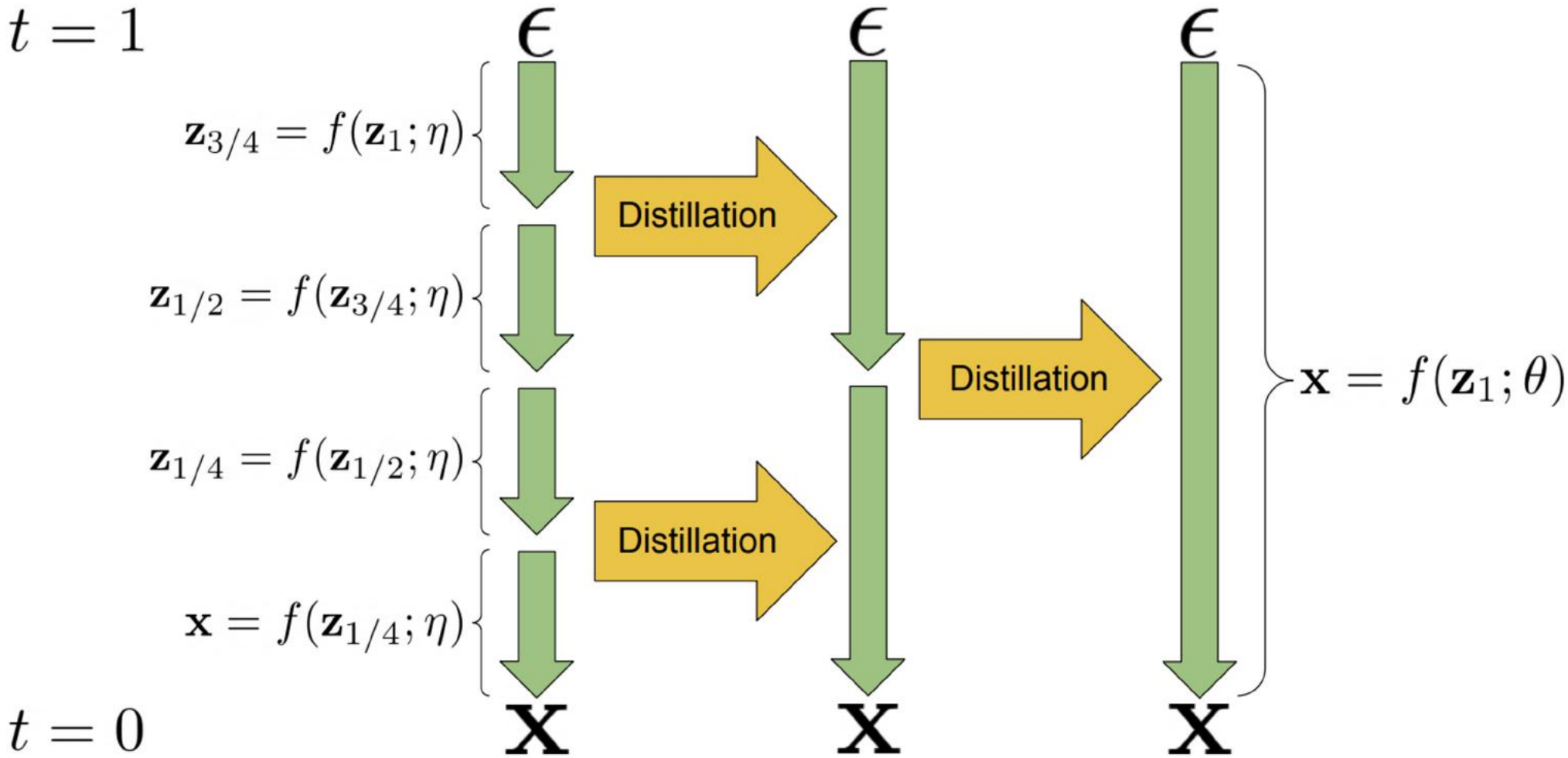
- **Problem:** Diffusion models take many steps to sample.
  - DDPM, Diffusion Beats Gans: 1000 timesteps.
  - Large N good for training, but slow to evaluate.

- **Insight:** Distill an N-step diffusion model into N/2 steps.
  - Then repeat this progressively -- N/4 steps, N/8, etc.

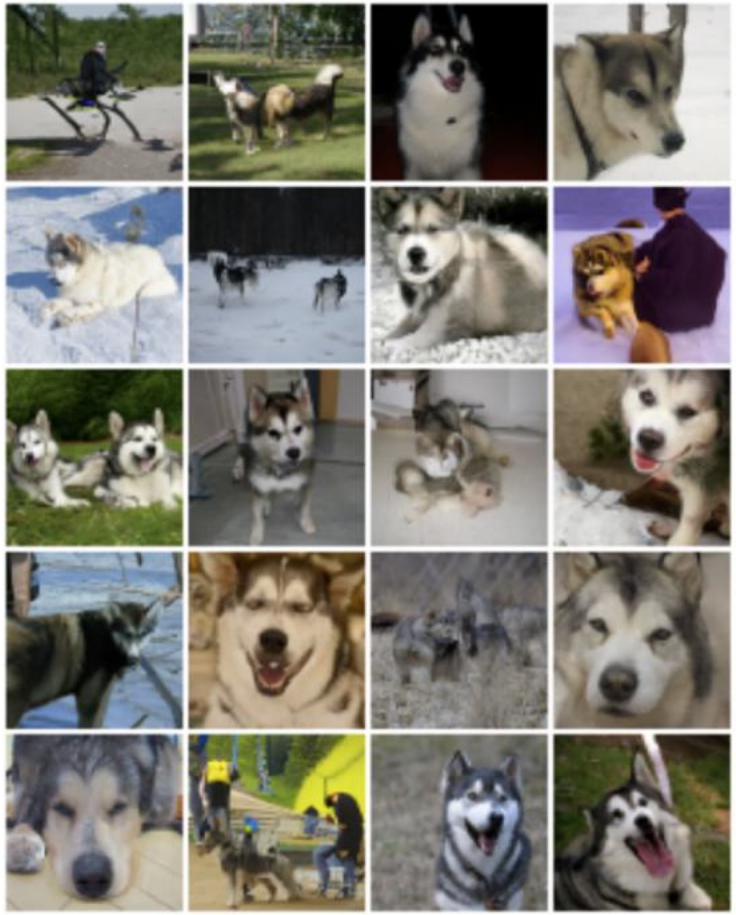
Progressive Distillation

- **Standard Diffusion**
  - Sample x, t, eps. Add eps to x.
  - Update model to predict **original x**.
- **Progressive Distillation**
  - Sample x, t, eps. Add eps to x.
  - Run teacher model twice to get x'.
  - Update model to predict x'.
- Why?
  - p(x_original | x_noise) is a noisier distribution with many answers, whereas p(x' | x_noise) is more determined.

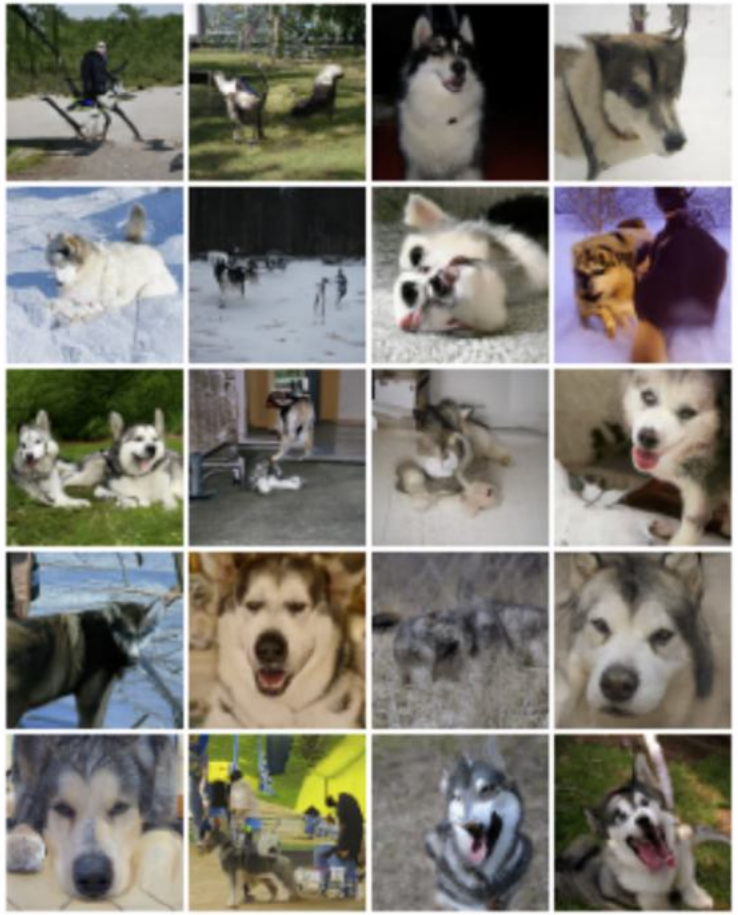
Progressive Distillation



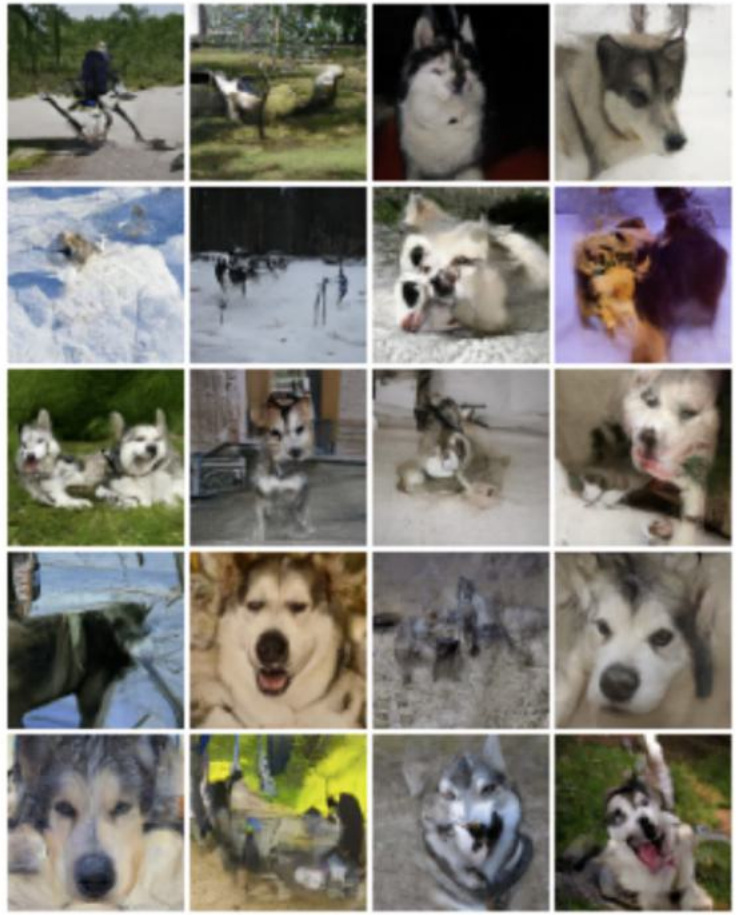
Progressive Distillation



(a) 256 sampling steps



(b) 4 sampling steps



(c) 1 sampling step

Denoising Diffusion GAN

- We need many diffusion steps because of Gaussian assumption.
  - The forward process is a Gaussian process.
  - True backward process is a Gaussian, **only for small step sizes.**
  - For large steps, not Gaussian.
    - p(x_0) is multimodal, a result of many unimodal (gaussian) steps.

Xiao, Zhisheng, Karsten Kreis, and Arash Vahdat. "Tackling the generative learning trilemma with denoising diffusion gans." arXiv preprint arXiv:2112.07804 (2021).

Denoising Diffusion GAN

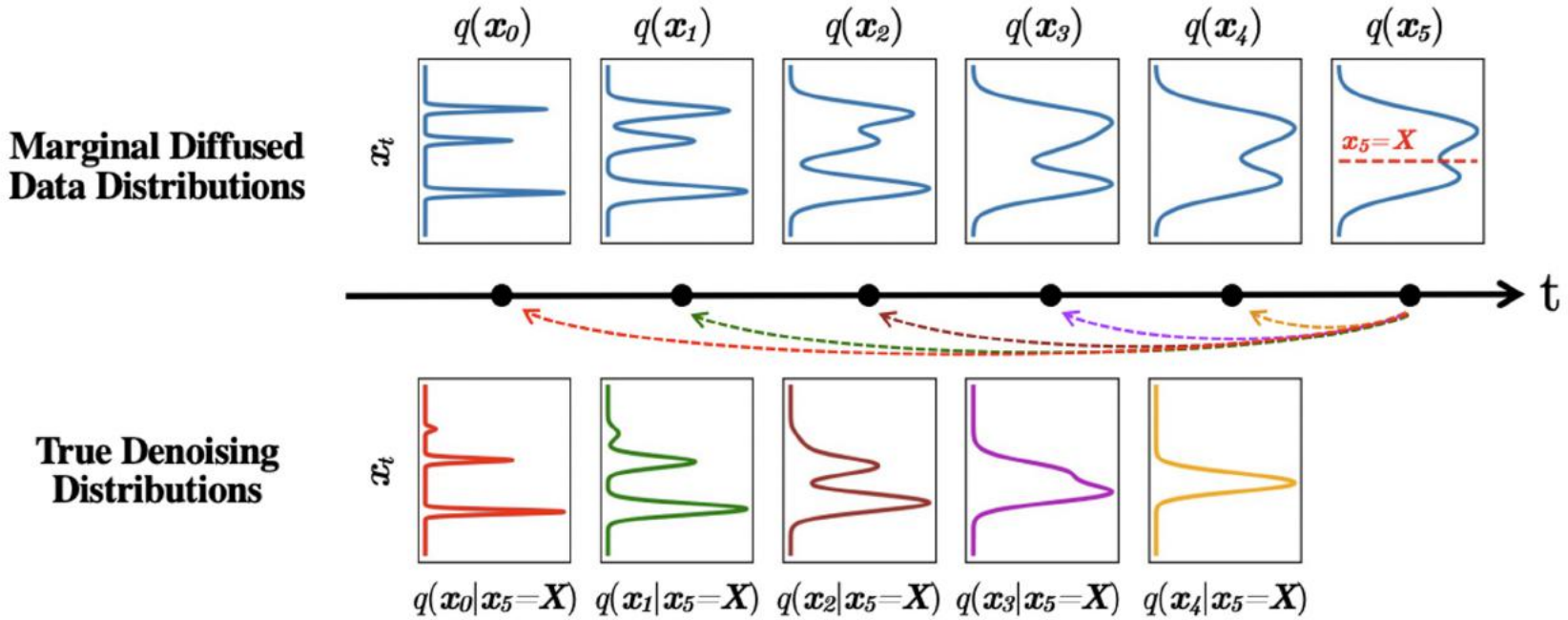


Figure 2: **Top**: The evolution of 1D data distribution $q(\mathbf{x}_0)$ through the diffusion process. **Bottom**:, The visualization of the true denoising distribution for varying step sizes conditioned on a fixed $\mathbf{x}_5$. The true denoising distribution for a small step size (i.e., $q(\mathbf{x}_4|\mathbf{x}_5 = X)$) is close to a Gaussian distribution. However, it becomes more complex and multimodal as the step size increases.

Denoising Diffusion GAN

- Instead, train the reverse distribution with a GAN.

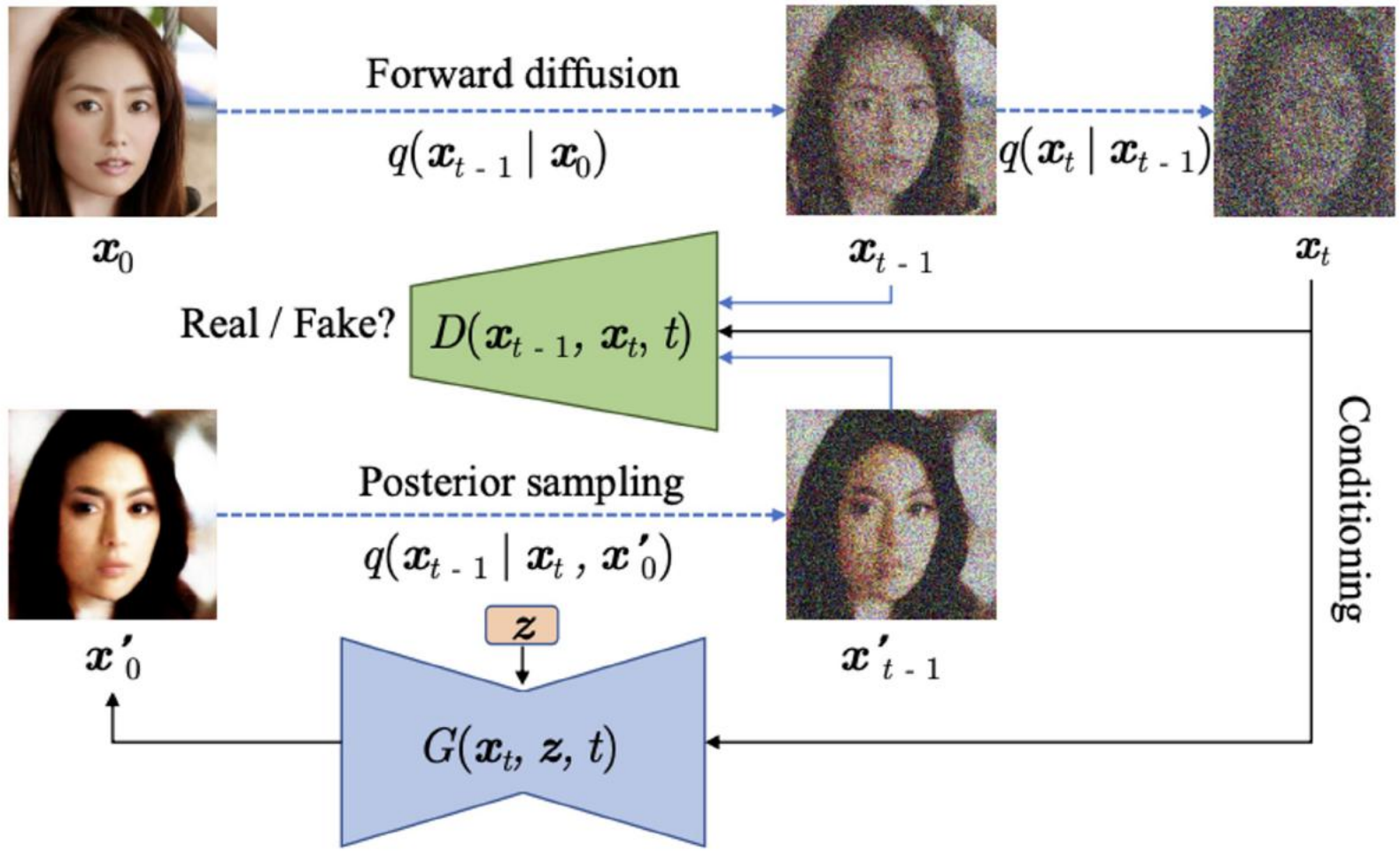
$$\min_{\theta} \sum_{t \geq 1} \mathbb{E}_{q(\mathbf{x}_t)} \left[D_{\text{adv}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t) \| p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) \right],$$

- We can sample from the true reverse distribution by running the forward process on data. Train discriminator:

$$\mathbb{E}_{q(\mathbf{x}_t)q(\mathbf{x}_{t-1}|\mathbf{x}_t)}[-\log(D_\phi(\mathbf{x}_{t-1}, \mathbf{x}_t, t))] = \mathbb{E}_{q(\mathbf{x}_0)q(\mathbf{x}_{t-1}|\mathbf{x}_0)q(\mathbf{x}_t|\mathbf{x}_{t-1})}[-\log(D_\phi(\mathbf{x}_{t-1}, \mathbf{x}_t, t))].$$

- Then, train generator to maximize discriminator objective.

Denoising Diffusion GAN



Denoising Diffusion GAN

(4 Diffusion Steps)

MobileDiffusion

MobileDiffusion: Subsecond Text-to-Image Generation on Mobile Devices

Yang Zhao, Yanwu Xu, Zhisheng Xiao, Tingbo Hou
Google
{yzhaoeric, yanwuxu, zsxiao, tingbo}@google.com

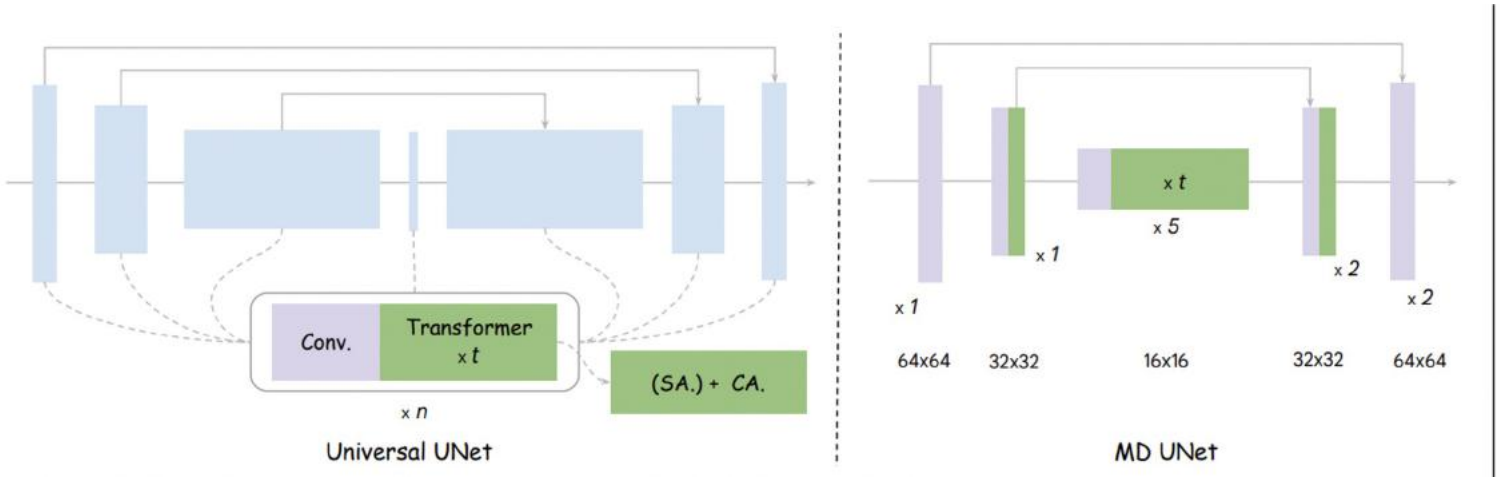
(a) MobileDiffusion, distilled 8 steps

(b) MobileDiffusion, finetuned 1 step

Figure 1. MobileDiffusion text-to-image generation in (a) 8 steps (*cfg-aware* distillation [23]), and (b) 1 step (UFOGen finetuning [57]).

MobileDiffusion

- Many architectural changes can reduce parameter counts.
  - More transformers inside middle of Unet. [26% efficiency gain]
    - Intuition: perform calculation in compressed latent space.
  - Decouple self-attention and cross-attention. [15% efficiency gain]
    - Cross-attention to text is important – include at all layers.
    - Self-attention is expensive at high-resolution – include only in middle.



MobileDiffusion

- Many architectural changes can reduce parameter counts.
  - Share key-value projections. [5% parameter reduction]
    - Set (key = value) for self-attention.
  - Finetune softmax into relu.
    - Attention softmax requires sum over exponents, pointwise relu faster.
    - Can train with softmax, then finetune to work with relu.
  - Reduce feedforward layer width. [10% parameter reduction]
  - Use separable convolutions [Howard et al, 2017] [10% param reduc.]
  - Reduce residual blocks. [15% parameter reduction]

MobileDiffusion

Models	#Channels	#ConvBlocks	#(SA+CA)	#Params(M)	#GFLOPs	Latency(ms)	Model Size (GB)
SD-XL [36]	(320, 640, 1280)	17	31+31	2,300	710	29.5	5.66
SD-1.4/1.5	(320, 640, 1280, 1280)	22	16+16	862	392	21.7	2.07
SnapFusion [23]	(320, 640, 1280, 1280)	18	14+14	848	285	15.0	1.97
MobileDiffusion	(320, 640, 1024)	11	15+18	386	182	9.9	1.04
MobileDiffusion-Lite	(320, 640, 896)	11	12+15	278	153	8.8	0.82

Table 1. Comparison with other recognized latent diffusion models. Latency and GFLOPs, computed with jit per forward step, ar measured for an input latent size of 64×64 on TPU v3. Model size (fp16) includes all, *i.e.*, UNet, text encoder and VAE decoder.

SDXL-Lightning

[Submitted on 21 Feb 2024]

SDXL-Lightning: Progressive Adversarial Diffusion Distillation

Shanchuan Lin, Anran Wang, Xiao Yang

We propose a diffusion distillation method that achieves new state-of-the-art in one-step/few-step 1024px text-to-image generation based on SDXL. Our method combines progressive and adversarial distillation to achieve a balance between quality and mode coverage. In this paper, we discuss the theoretical analysis, discriminator design, model formulation, and training techniques. We open-source our distilled SDXL-Lightning models both as LoRA and full UNet weights.

SDXL-Lightning

- Key Idea: Denoising Diffusion GAN, for **distillation**.

- Distillation of diffusion models fails at small step sizes.
  - It's the same MSE ambiguity problem we covered earlier.
  - Small steps need to model a complicated flow -- Gaussian not enough.
  - Instead -- distill with a GAN.

SDXL-Lightning

- SDXL-Lightning GAN
  - Discriminator conditioned on pairs of (x, x_nstep_denoised).
    - Samples from either teacher network or student.

$$D(x_t, x_{t-ns}, t, t-ns, c)$$

      - Re-use the weights from the teacher encoder for the discriminator.

SDXL-Lightning

- Janus Artifacts
  - Student model still has less capacity than the teacher = mismatch.
  - Relaxing mode-coverage reduces complexity.
    - Finetune on unconditional flows.

$$D(x_t, x_{t-ns}, t, t-ns, c)$$

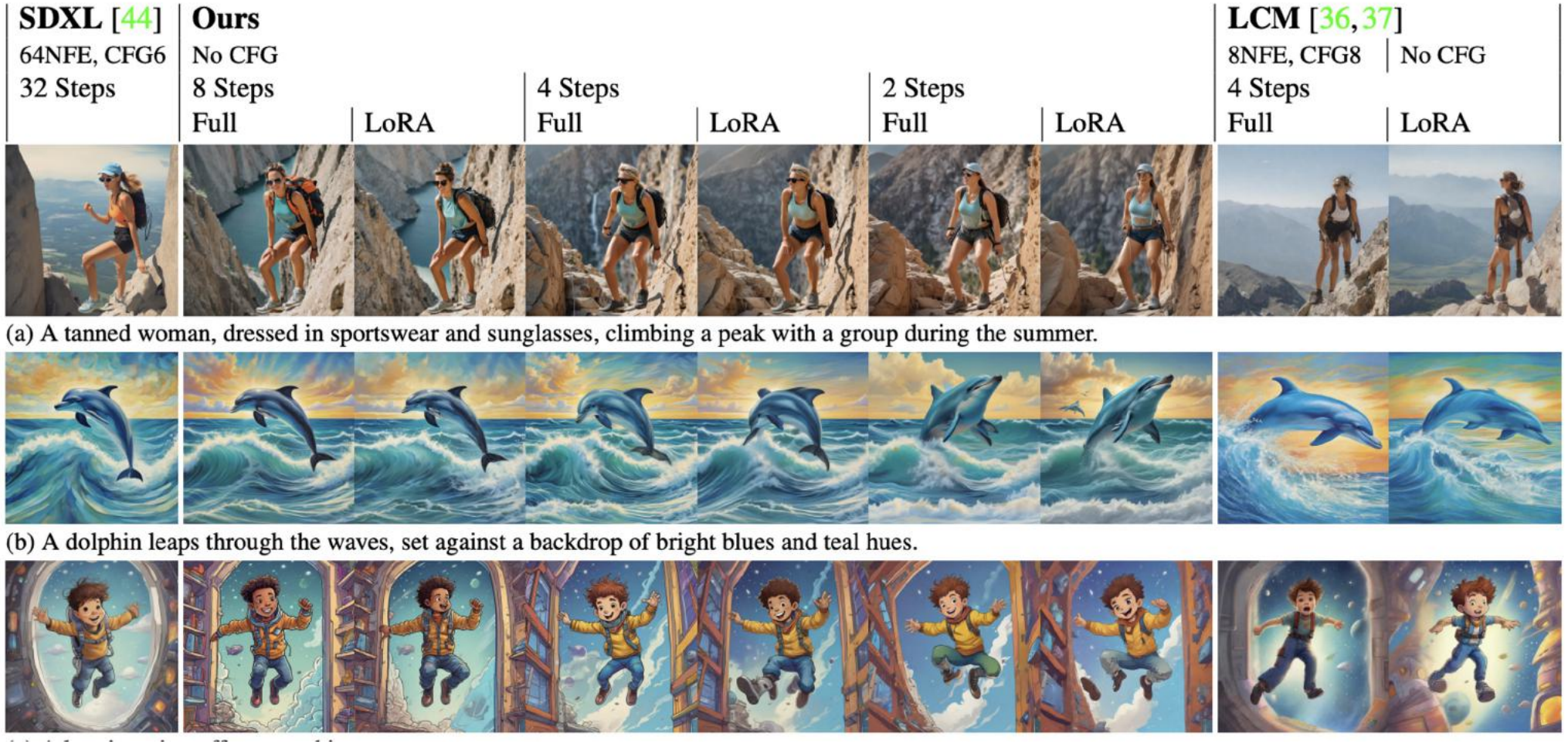
$$D'(x_{t-ns}, t-ns, c)$$

Figure 2. "Janus" artifacts appear when the student network does not have the capacity to match the teacher's sudden changes. This problem can be mitigated by relaxing the mode coverage requirement.

SDXL-Lightning

- Training Details
  - Distill 128 -> 32 with MSE loss.
  - Distill 32,8,4,3,1 with GAN loss.
    - First with conditional, then unconditional, in each stage.
    - Train with LoRA, then further finetune full model.
  - Train at multiple timesteps, even if not necessary for generation.

SDXL-Lightning



Lecture overview

- Motivation
- Energy-based models
- Score-based Models
- Denoising Diffusion Models
- **Deeper Dive into Diffusion Models**
  - DDIM Sampling
  - Prediction Space / Loss: x, eps, v
  - Issues with Common Noise Schedules
  - Architectures: UNet, latent space / stable diffusion, hierarchical generation, transformers
  - Image Editing / Video Editing
  - Using scores in diffusion models: text-to-3D, text-to-SVG, Video Adapter
  - Faster Sampling
  - **Diffusion as Pre-Training**
  - Other Fields: visuomotor control, materials discovery

Diffusion Features

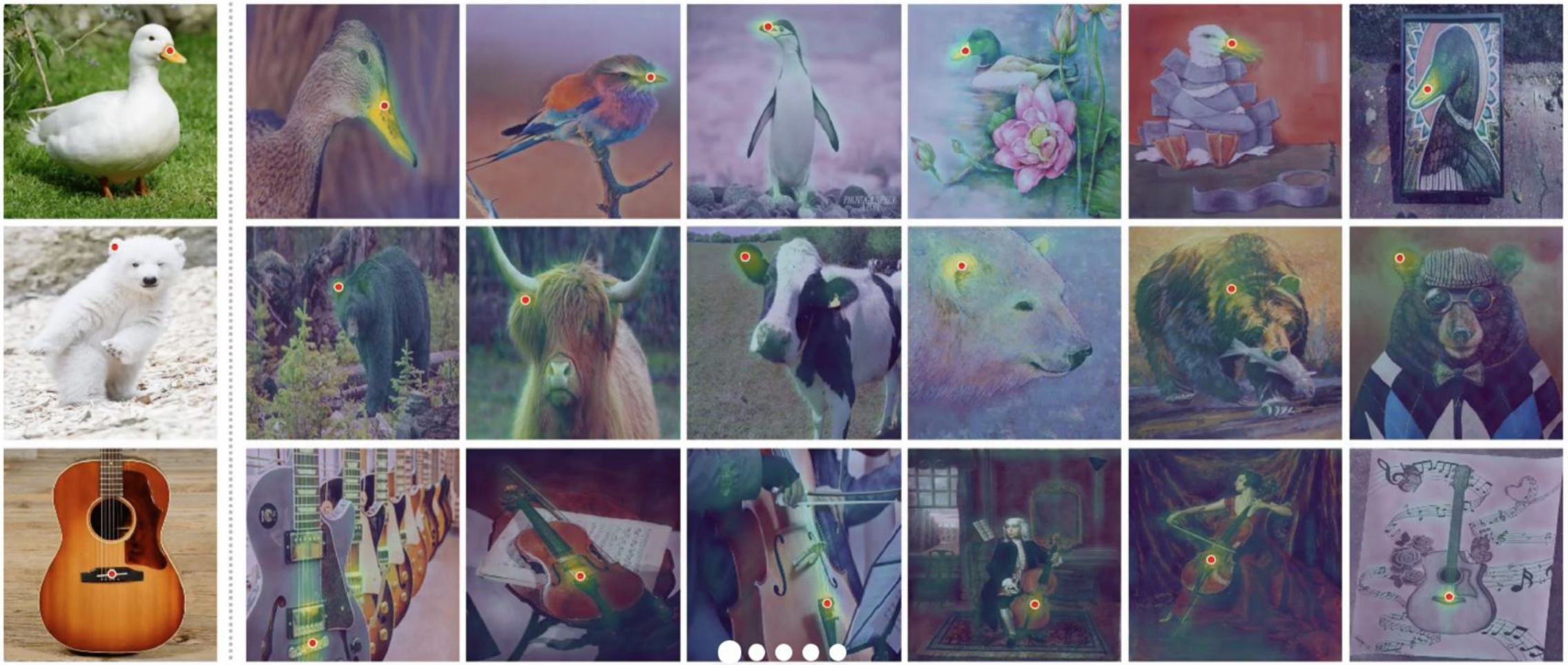
- **Question:** Unsupervised learning is also wanted for representation learning. Can diffusion models be used for this?

- **Insight:** Diffusion model is trained with noisy images as input.
  - To extract features, **add noise** to image, then process.
    - Average over a batch of noise for better accuracy.
  - Correspondences can be located via cosine distance.

  - As T changes, tradeoff between semantics and details.

Tang, Luming, et al. "Emergent correspondence from image diffusion." Advances in Neural Information Processing Systems 36 (2024).

Diffusion Features



Diffusion Features

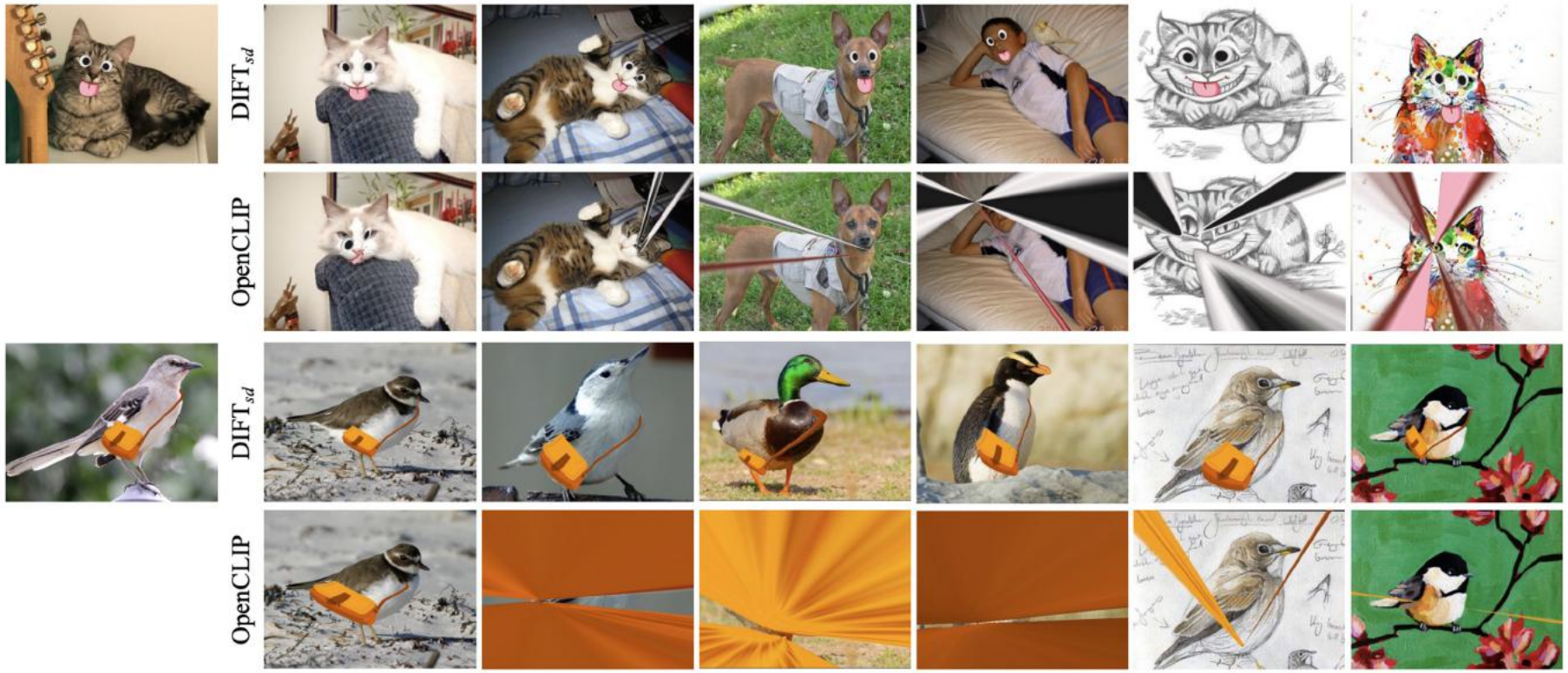


Figure 6: Edit propagation. The first column shows the source image with edits (i.e., the overlaid stickers), and the rest columns are the propagated results on new images from different instances, categories, and domains, respectively. Compared to OpenCLIP, DIFT$_{sd}$ propagates edits much more accurately. More results are in Fig. 20 of Appendix E.

Diffusion Classifier

Your Diffusion Model is Secretly a Zero-Shot Classifier

Alexander C. Li Mihir Prabhudesai Shivam Duggal Ellis Brown Deepak Pathak

Carnegie Mellon University

Diffusion Classifier

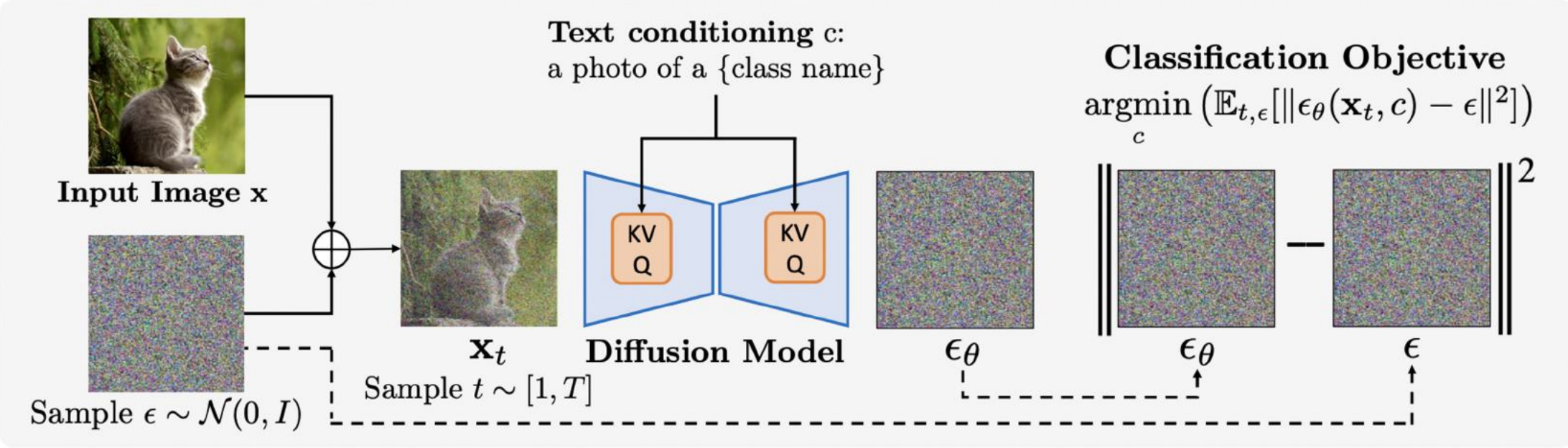
- **Problem:** We want to classify images, or otherwise compute a distance between [image, text] pairs.

- **Insight:** We have an approximate model of p(x|c). Use Bayes' rule:

$$p_\theta(\mathbf{c}_i \mid \mathbf{x}) = \frac{p(\mathbf{c}_i)\, p_\theta(\mathbf{x} \mid \mathbf{c}_i)}{\sum_j p(\mathbf{c}_j)\, p_\theta(\mathbf{x} \mid \mathbf{c}_j)}$$

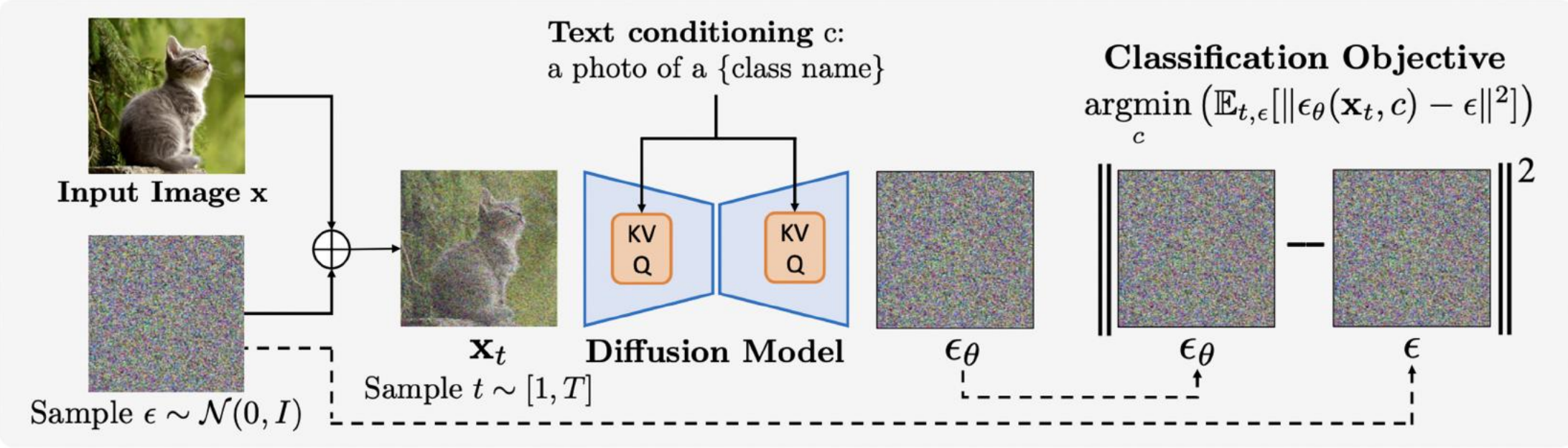
Diffusion Classifier

- [Image, text] distance can be calculated by adding noise, denoising with text-conditioning, and taking L2 distance.



Diffusion Classifier

- [Image, text] distance can be calculated by adding noise, denoising with text-conditioning, and taking L2 distance.



EmerDiff

Question: How do we pixel-level segmentation maps from diffusion models?

- Low-resolution solution: extract feature maps, then use K-means.
  - In a text-conditioned model, use the query vectors as features.
- To upscale: Figure out correspondence from features to pixels.
  - Add a (+,-) constant offset to the feature maps.
  - Then, take the difference in the generated image.

EmerDiff

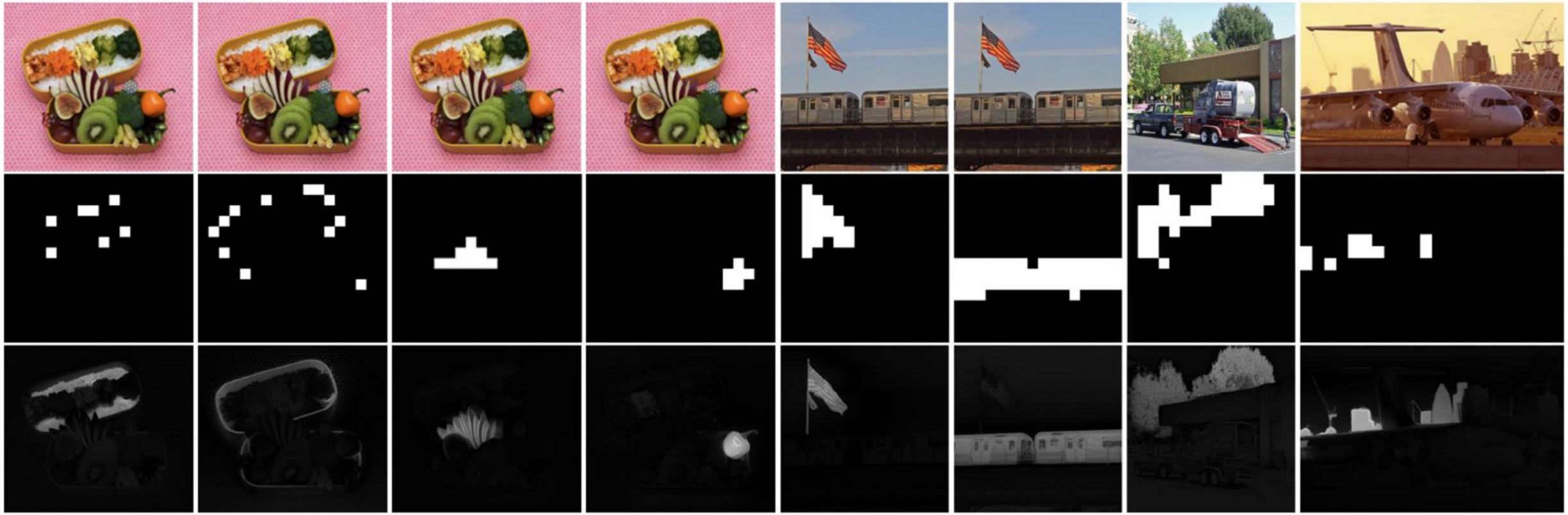
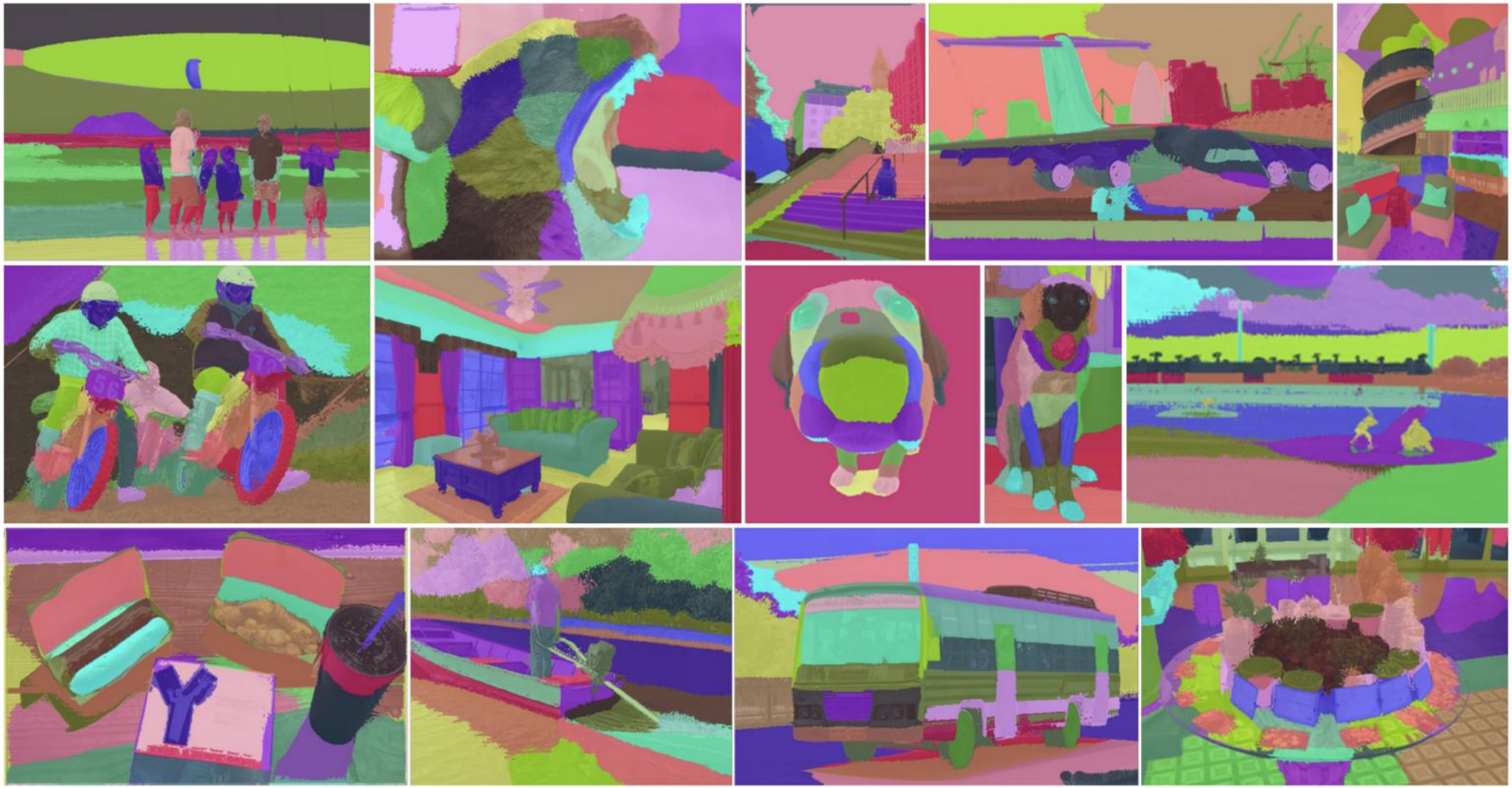


Figure 3: **Visualization of modulated denoising process.** First row: original image. Second row: low-resolution modulation mask $M \in \{0,1\}^{h \times w}$. Third row: obtained difference map $d \in \mathbb{R}^{H \times W}$, where $H/h = W/w = 32$

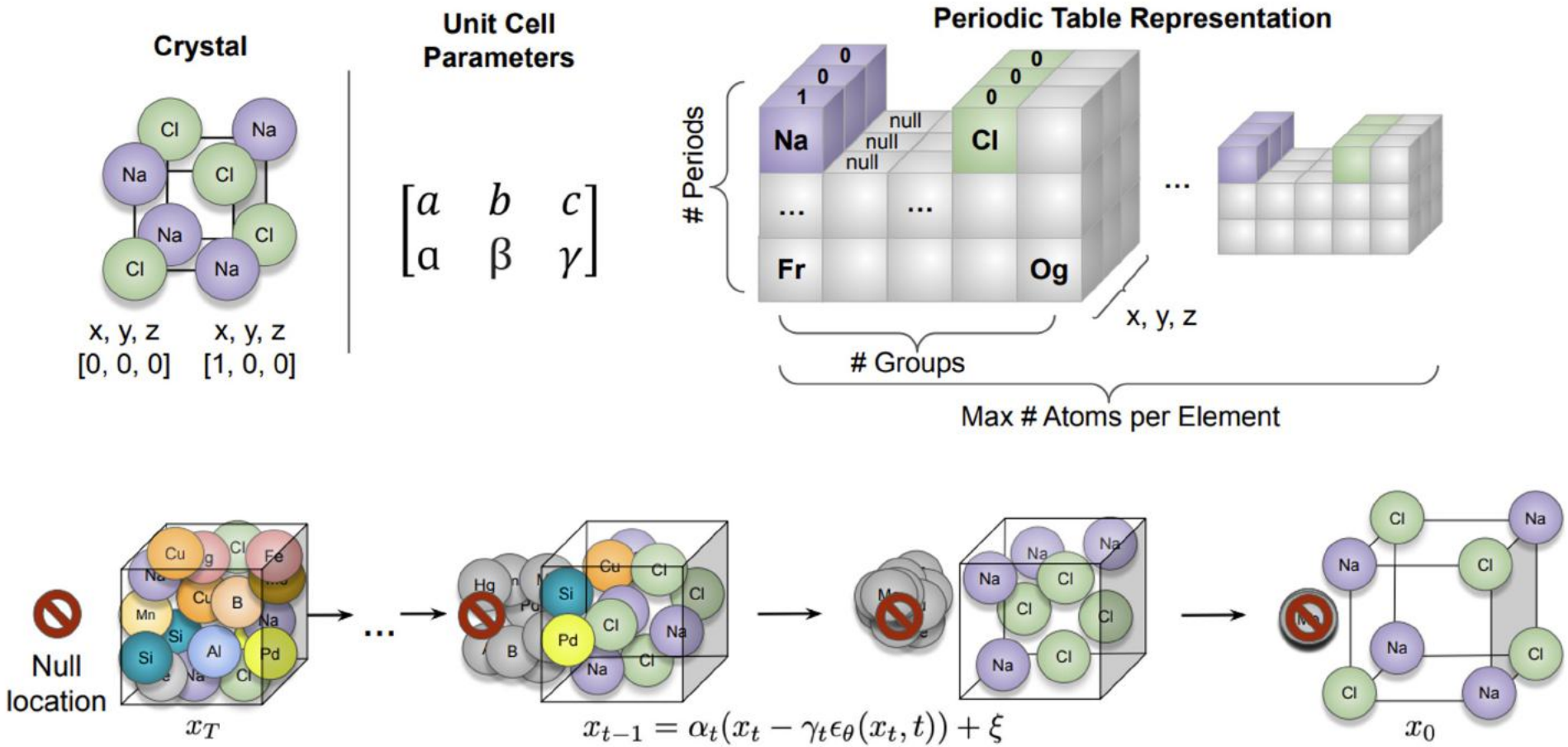
EmerDiff



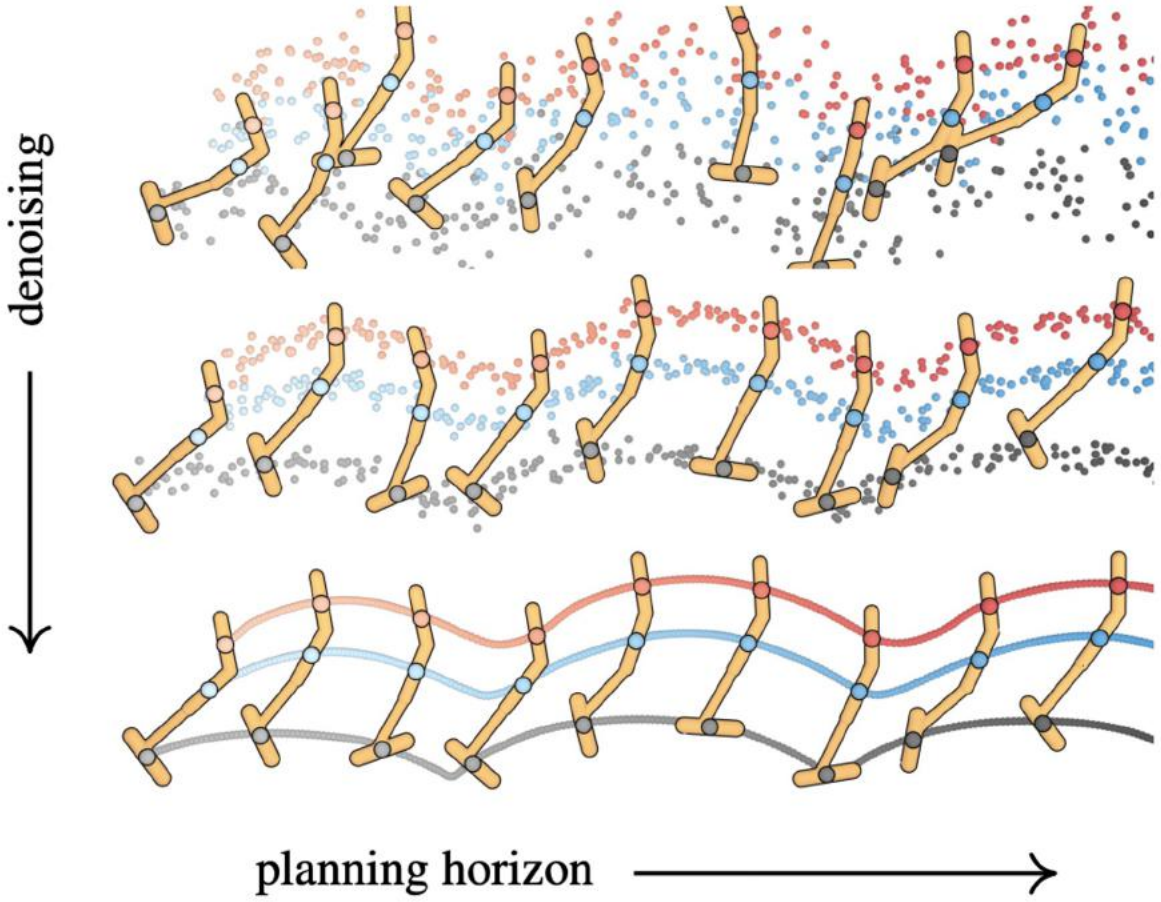
Lecture overview

- Motivation
- Energy-based models
- Score-based Models
- Denoising Diffusion Models
- **Deeper Dive into Diffusion Models**
  - DDIM Sampling
  - Prediction Space / Loss: x, eps, v
  - Issues with Common Noise Schedules
  - Architectures: UNet, latent space / stable diffusion, hierarchical generation, transformers
  - Image Editing / Video Editing
  - Using scores in diffusion models: text-to-3D, text-to-SVG, Video Adapter
  - Faster Sampling
  - Diffusion as Pre-Training
  - **Other Fields: visuomotor control, materials discovery**

UniMat: Scalable Diffusion for Materials Generation



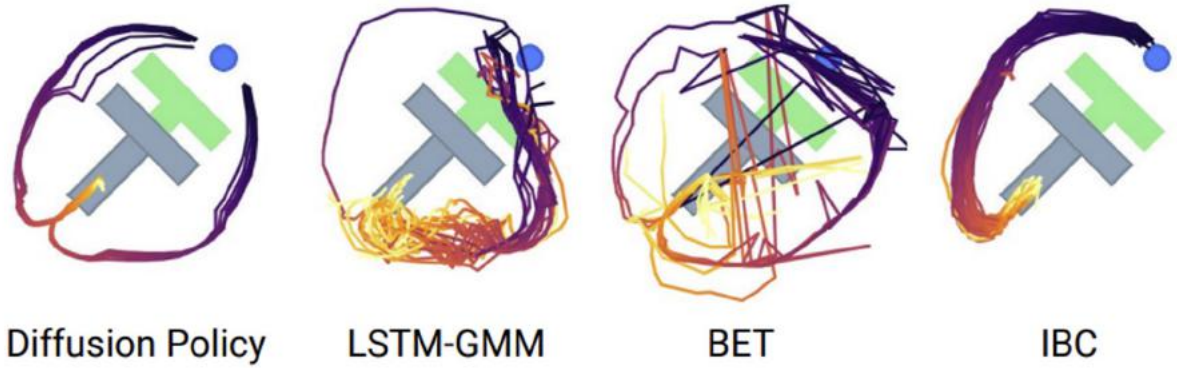
Planning with Diffusion for Flexible Behavior Synthesis



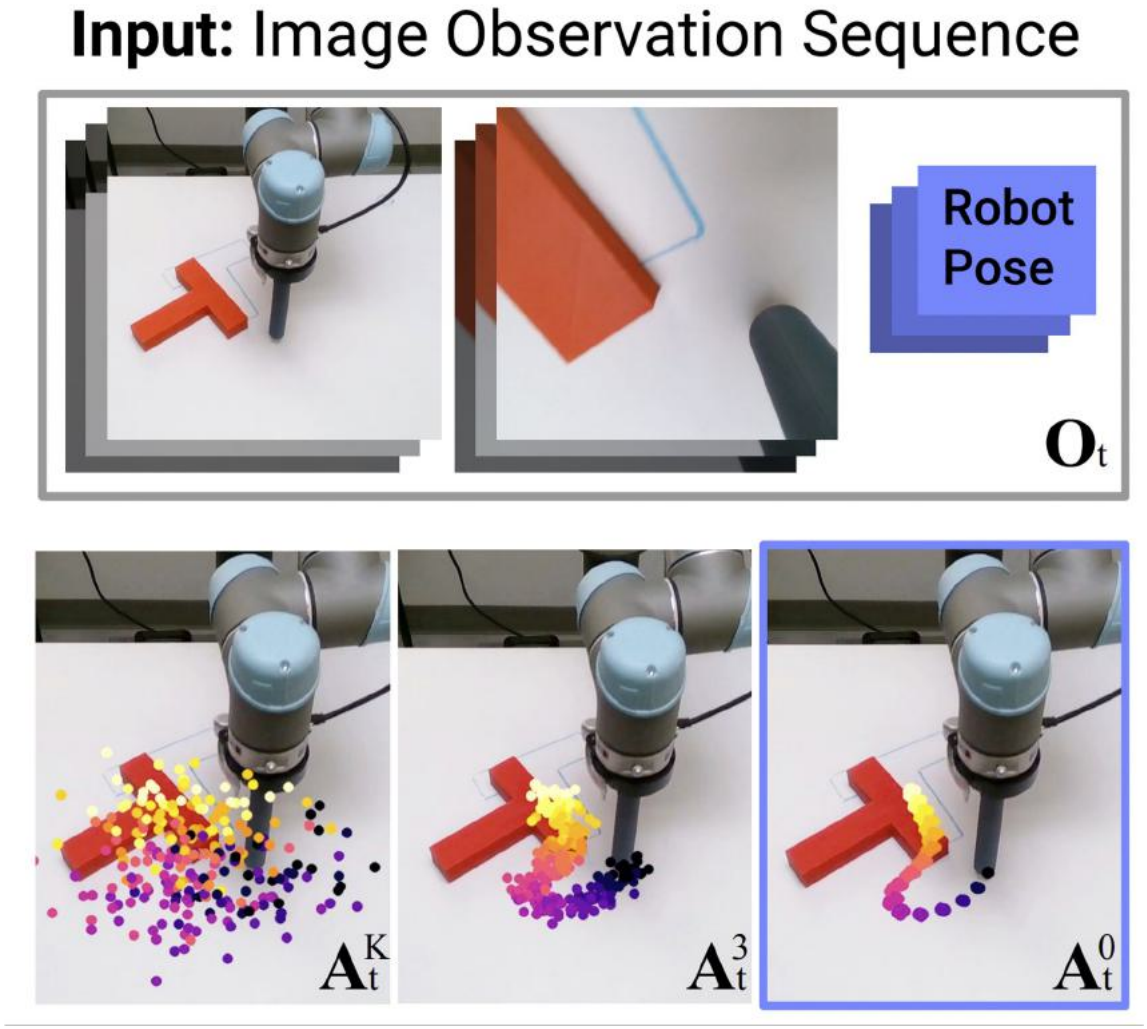
Diffusion Policy

- **Problem:** In robotics, expert trajectories are multimodal -- Gaussian policies fail to capture behavior.

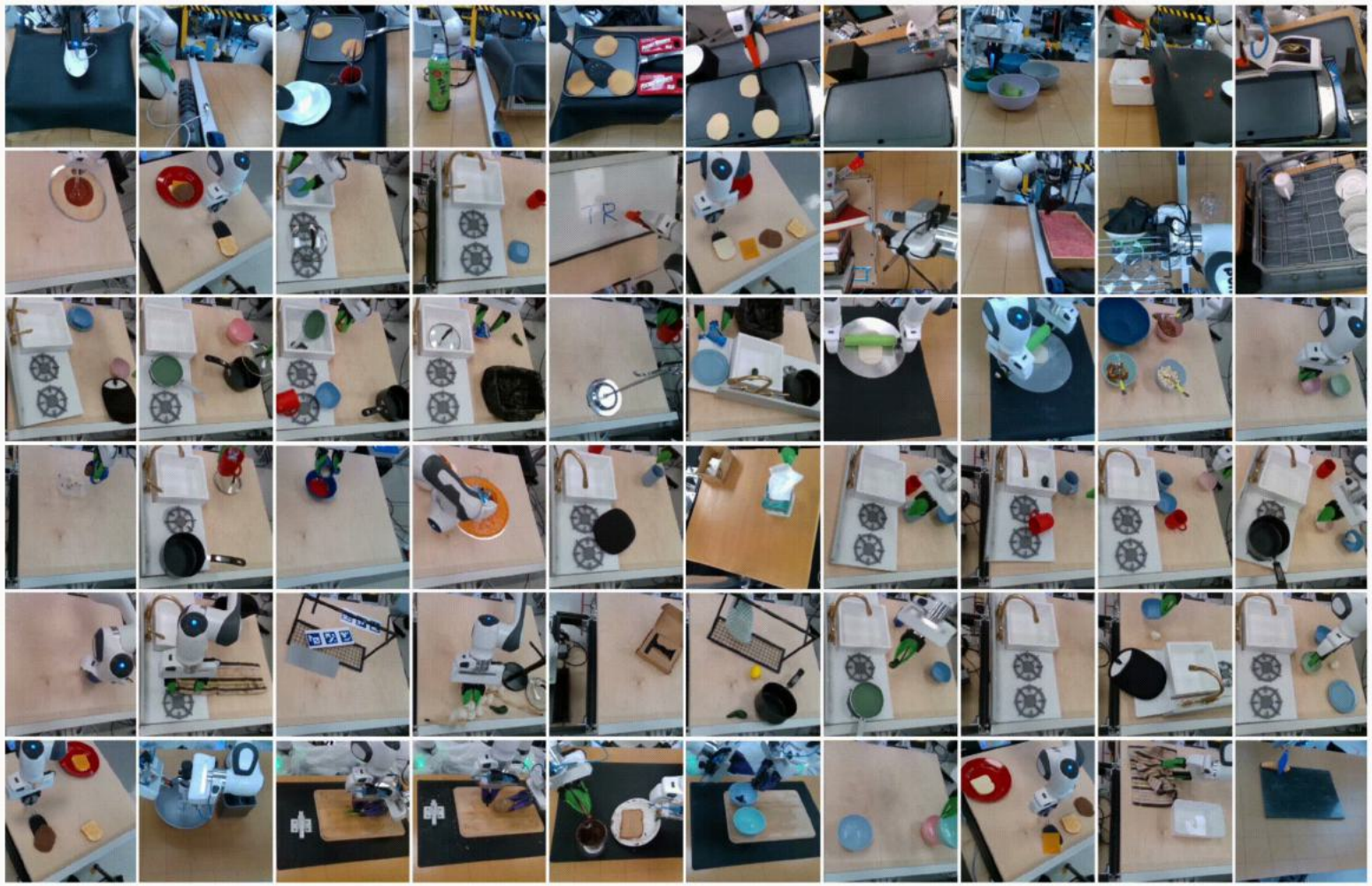
- **Solution:** Use diffusion model over actions, and diffuse over future action trajectories.



Diffusion Policy



Teaching Robots New Behaviors



Next lecture:
Strengths and Weaknesses of Current Models