Diffusion

Romain Loiseau

Slides mainly taken from a talk from **Yannis Siglidis** (IMAGINE/ENPC)

Diffusion

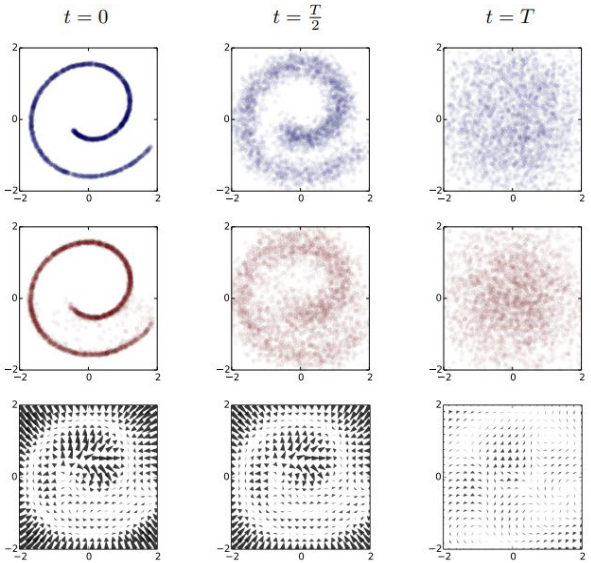
1. What is diffusion and how does it work ?
2. Conditioning diffusion to match a description ?
3. Diffusion for 3D !

I didn’t knew anything about diffusion 10 days ago so **please interrupt me** if:

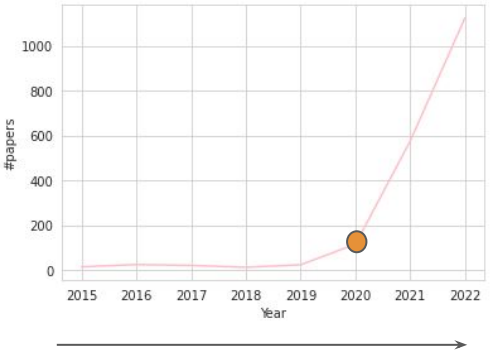
- You don’t understand anything I’m saying and want me to clarify
- You want to add more details you know
- You want me to stop explaining useless math things because it’s too painful
- I’m saying stupid things

Key ideas for each slide in red here

History



Papers on Diffusion



+ 5 yrs

First results mainly p.o.c. (cifar, texture synthesis, etc)

Sohl-Dickstein et al., 2015 -> Nonequilibrium Thermodynamics

"DDPM": Realistic faces, churches, bedrooms, etc.

Ho et al. 2020,

“Diffusion models beat gans on image synthesis.” (SoTA + conditional generation)
Nichol & Dhariwal (2021)

State of the art:

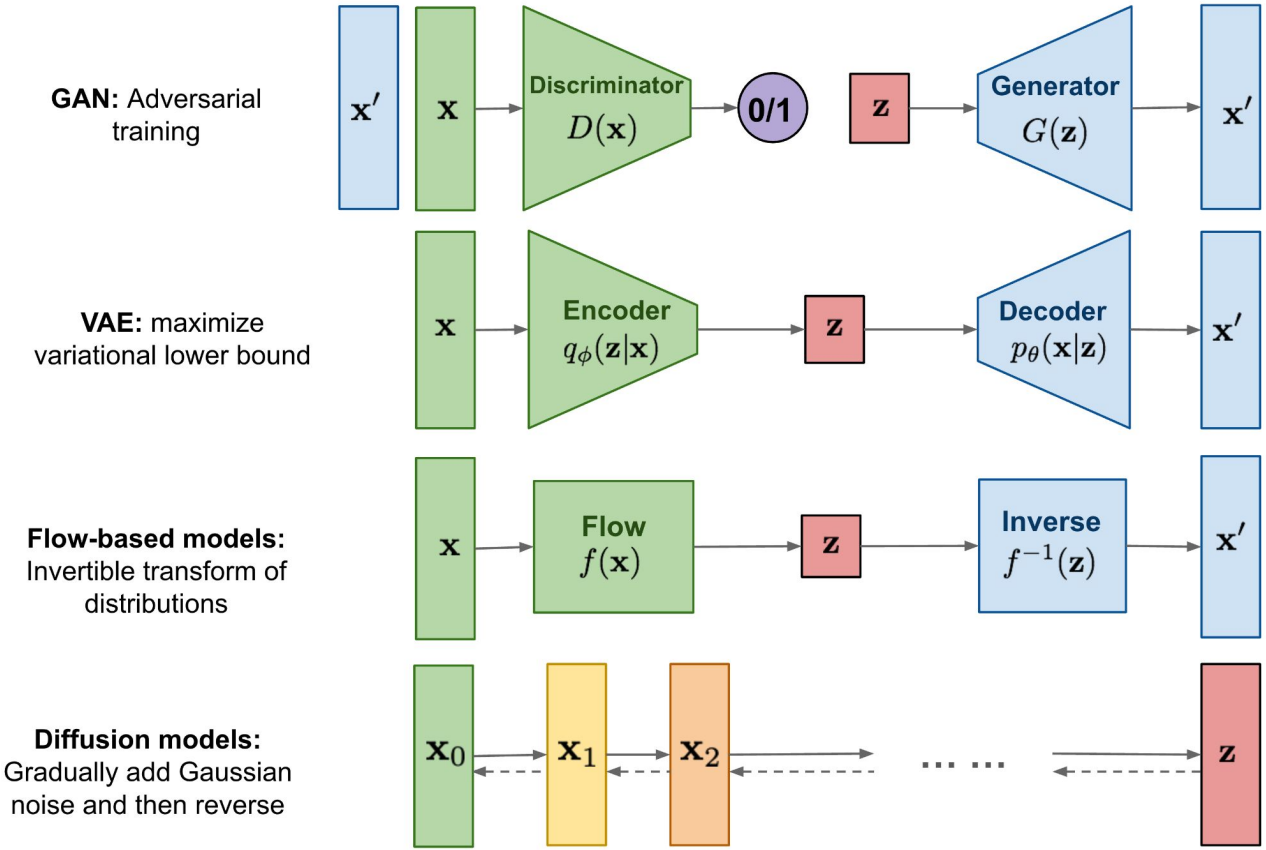
DALL·E 2

Stable Diffusion

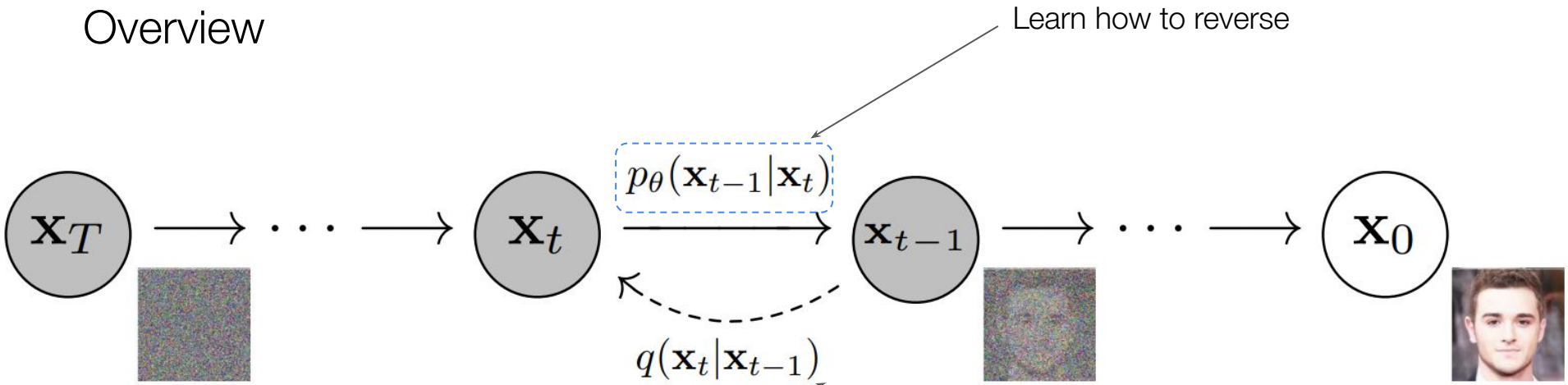
Stable Diffusion is a machine learning, text-to-image model to generate digital images from natural language descriptions. The underlying approach was developed at LMU Munich and then extended by a collaboration of StabilityAI, LMU, and Runway with support from EleutherAI and LAION. Wikipedia

Compositional Generalization!

A generative model.



Overview



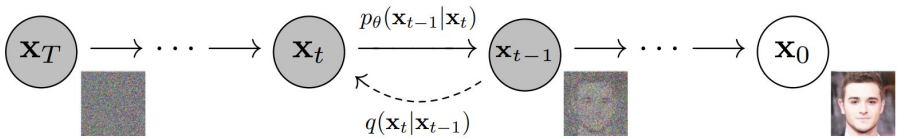
$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

$\beta_T = 0.02$ — Fixed linear schedule → $\beta_1 = 10^{-4}$

Ho et al. 2020

The Markov chain diffusion process of generating a sample

Backward



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

$$x_1 \leftarrow \sqrt{1-\beta_1}x_0 + \sqrt{\beta_1}z, \;\; z \sim \mathcal{N}(0,1)$$

$$x_2 \leftarrow \sqrt{1-\beta_2}x_1 + \beta_2 z = \sqrt{1-\beta_2}(\sqrt{1-\beta_1}x_0 + \sqrt{\beta_1}z') + \sqrt{\beta_2}z, \;\; z' \sim \mathcal{N}(0,1)$$

$$x_2 \leftarrow \sqrt{(1-\beta_2)(1-\beta_1)}x_0 + \sqrt{1-\beta_2}\sqrt{\beta_1}z' + \sqrt{\beta_2}z$$

$$x_2 \leftarrow \sqrt{(1-\beta_2)(1-\beta_1)}x_0 + \mathcal{N}(0, (1-\beta_2)\beta_1) + \mathcal{N}(0, \beta_2)$$

$$x_2 \leftarrow \sqrt{(1-\beta_2)(1-\beta_1)}x_0 + \mathcal{N}(0, 1-(1-\beta_2)(1-\beta_1))$$

$$q(x_2 \mid x_0) = \mathcal{N}(x_2; \sqrt{(1-\beta_2)(1-\beta_1)}x_0, 1-(1-\beta_2)(1-\beta_1))$$

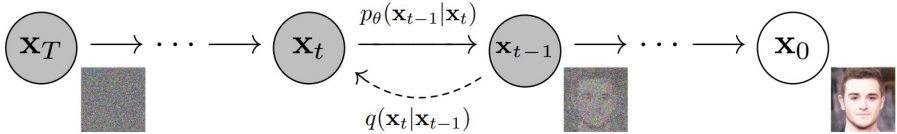
$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \xrightarrow[\bar{\alpha}_t = \prod_{i=1}^{T} \alpha_i]{\alpha_t = 1-\beta_t} \boxed{q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})}$$

Sohl-Dickstein et al., 2015

Step Independent Computation is key for the training to be tractable

Forward

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$$



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

$$\tilde{\boldsymbol{\mu}}_t \equiv \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\mathbf{z}_t\right) \qquad \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\cdot\beta_t$$

Close form transition kernel conditioned on $\mathbf{x}_0$

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

$$\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\mathbf{z}_\theta(\mathbf{x}_t, t)\right)$$

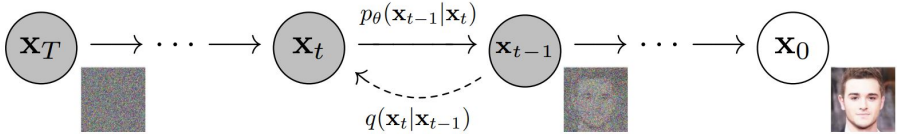
Original: $\boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t) = \sigma_t^2\mathbf{I}$

Model

Sohl-Dickstein et al., 2015

Ask the model to denoise -> estimate the mean and variance of the applied noise

Loss Derivation



$$q\left(\mathbf{x_{0:T}}\right) = q\left(\mathbf{x}_0\right) \prod_{t=1}^{T} q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right)$$

Forward Trajectory

Markov Property

$$p_\theta\left(\mathbf{x_{0:T}}\right) = p_\theta\left(\mathbf{x}_T\right) \prod_{t=1}^{T} p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)$$

Reverse Trajectory

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

Variational Lower Bound Loss

$$L_{\text{VLB}} = L_T + L_{T-1} + \cdots + L_0$$

where $L_T = D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_T))$ Constant - can be ignored

$L_t = D_{\text{KL}}(q(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1}))$ for $1 \leq t \leq T-1$

Actual Loss

$L_0 = -\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)$ Entropy

Expanding VLB

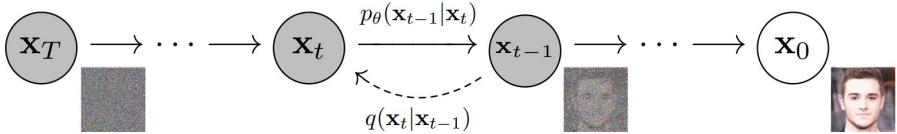
$$D_{KL}(q||p) = E_q[\log \tfrac{p}{q}]$$

Sohl-Dickstein et al., 2015

Train it using a single loss at time t

Loss

$$D_{KL}(q\|p) = E_q[\log \frac{p}{q}]$$



$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$$

$$\tilde{\boldsymbol{\mu}}_t \equiv \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\mathbf{z}_t\right)$$

Transition Kernel

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

$$\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\mathbf{z}_\theta(\mathbf{x}_t, t)\right)$$

Model

$$L_t = \mathbb{E}_{\mathbf{x}_0,\mathbf{z}}\left[\frac{1}{2\|\boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)\|_2^2}\|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t, t)\|^2\right]$$

$$L_t = \mathbb{E}_{\mathbf{x}_0,\mathbf{z}}\left[\frac{\beta_t^2}{2\alpha_t(1-\bar{\alpha}_t)\|\boldsymbol{\Sigma}_\theta\|_2^2}\|\mathbf{z}_t - \mathbf{z}_\theta(\mathbf{x}_t, t)\|^2\right]$$

Denoising

Step Independent Computation

$$L_t = \mathbb{E}_{\mathbf{x}_0,\mathbf{z}}\left[\frac{\beta_t^2}{2\alpha_t(1-\bar{\alpha}_t)\|\boldsymbol{\Sigma}_\theta\|_2^2}\|\mathbf{z}_t - \mathbf{z}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\mathbf{z}_t, t)\|^2\right]$$

Ignored by Ho et al. (2020) and denoted L_{simple}

Loss

Sohl-Dickstein et al., 2015

Ask the model to denoise

Training

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

Ho et al. (2020)

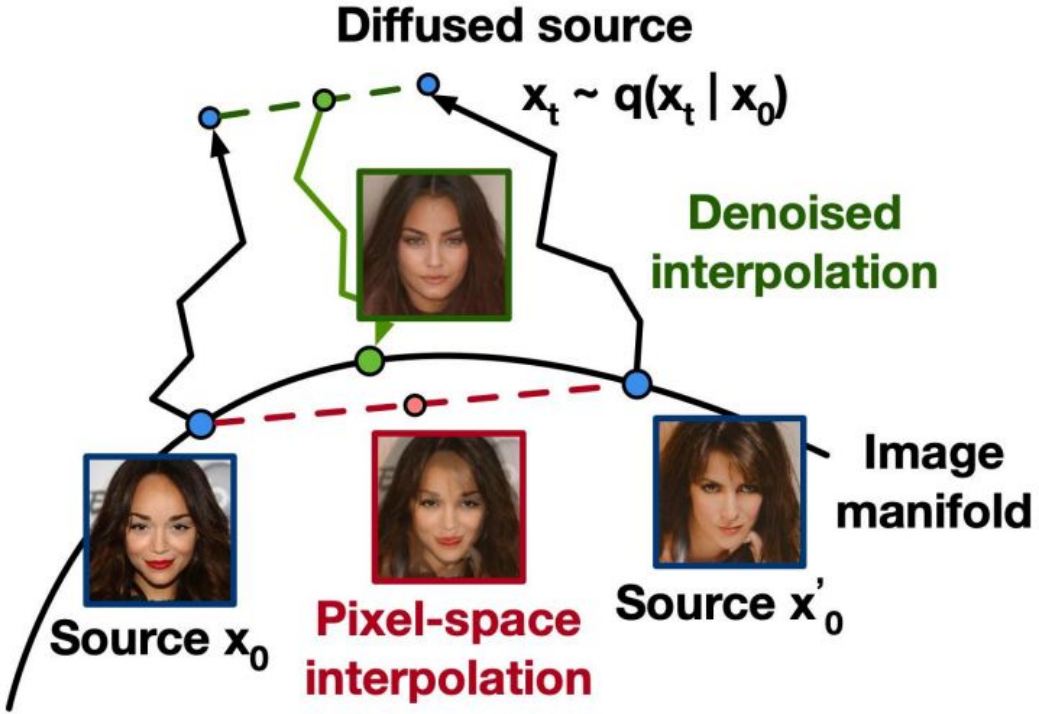
Sampling

Algorithm 2 Sampling $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$

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

Ho et al. (2020)

Interpolation



$\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ must be kept the same across all steps.

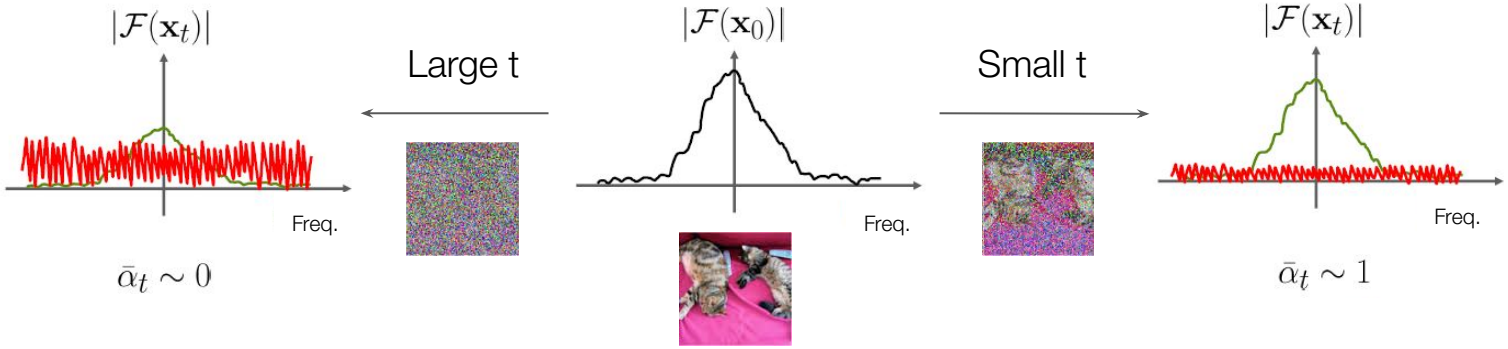
Ho et al. (2020)

Why does it work?

Fourier Transform

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

$$\mathcal{F}(\mathbf{x}_t) = \sqrt{\bar{\alpha}_t}\mathcal{F}(\mathbf{x}_0) + \sqrt{(1-\bar{\alpha}_t)}\mathcal{F}(\epsilon)$$



Low frequency dominance of image responses ⟶ high frequency information is perturbed faster.

You all know everything about how diffusion works know

Let’s dive into some fun stuffs now

Conditioning the Model

An astronaut riding a horse in a photorealistic style

Conditional Model: $\tilde{\boldsymbol{\epsilon}}_\theta\left(\mathbf{z}_t, t, \mathbf{c}\right)$

E.g. The network can be a UNET and ***c*** *can be a BERT embedding.*

Sample with: $\tilde{\boldsymbol{\epsilon}}_\theta\left(\mathbf{z}_t, t, \mathbf{c}\right) = (1+w)\underbrace{\boldsymbol{\epsilon}_\theta\left(\mathbf{z}_t, t, \mathbf{c}\right)}_{\text{Conditional}} - w\underbrace{\boldsymbol{\epsilon}_\theta\left(\mathbf{z}_t, t, \mathbf{c}=\mathbf{0}\right)}_{\text{Unconditional}}$

good balance between FID (distinguish between synthetic and generated images) and IS (quality and diversity)

Ho & Salimans (2021)

Classifier Free Guidance

Proof

Bayes formula

$$\nabla_{\mathbf{x}_t} \log p(y|\mathbf{x}_t) = \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|y) - \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t)$$
$$= -\frac{1}{\sqrt{1-\bar{\alpha}_t}} \Big(\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, y) - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \Big)$$

gradient of the log likelihood of an auxiliary classifier model pθ(y|x_t)

$$\bar{\boldsymbol{\epsilon}}_\theta(\mathbf{x}_t, t, y) = \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, y) - \sqrt{1-\bar{\alpha}_t}\, w \nabla_{\mathbf{x}_t} \log p(y|\mathbf{x}_t)$$
$$= \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, y) + w\big(\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, y) - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\big)$$
$$= (w+1)\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, y) - w\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$$

Latent Space Diffusion

x

Stable Diffusion

$$L_{LDM} = \mathbb{E}_{\mathcal{E}(x),\epsilon\sim\mathcal{N}(0,1),t}\left[\|\epsilon - \epsilon_\theta(z_t, t)\|_2^2\right]$$

Rombach, Blattmann, et al. (2022)

y

DALLE-2

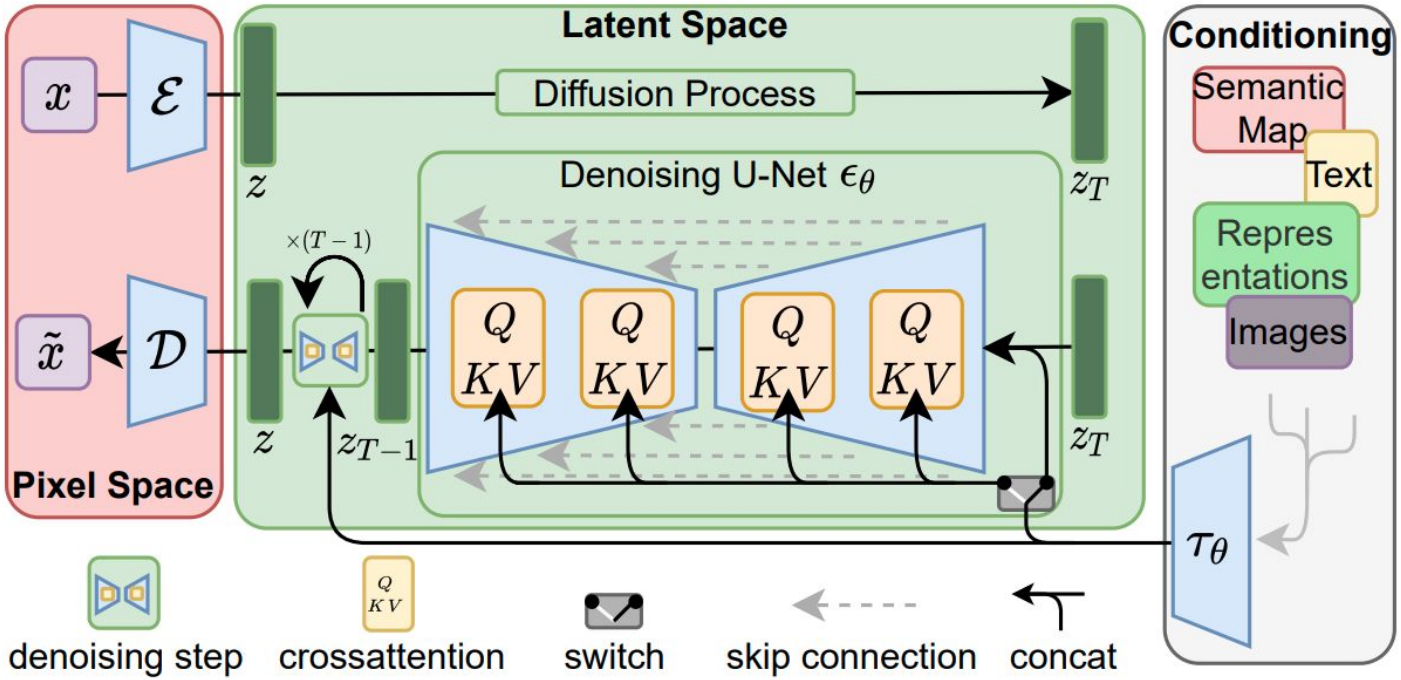
$$L_{\text{prior}} = \mathbb{E}_{t\sim[1,T],z_i^{(t)}\sim q_t}\left[\left\|f_\theta\left(z_i^{(t)}, t, y\right) - z_i\right\|^2\right]$$

Ramesh, et al. (2022)

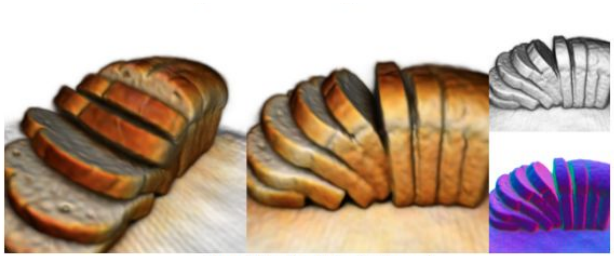
Directly predicts the **denoised**
(instead of the noise)

Latent Space Diffusion

Stable Diffusion, Rombach, Blattmann, et al. (2022)

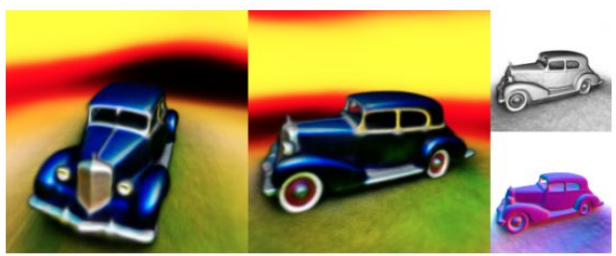


DREAMFUSION, Poole et al. (2022)

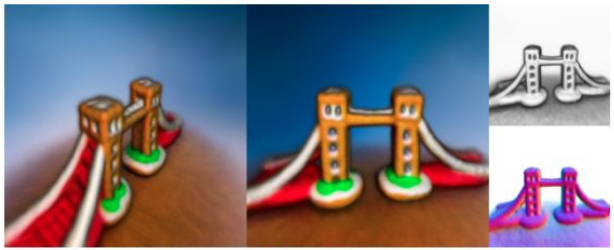


a sliced loaf of fresh bread

a bulldozer clearing away a pile of snow*

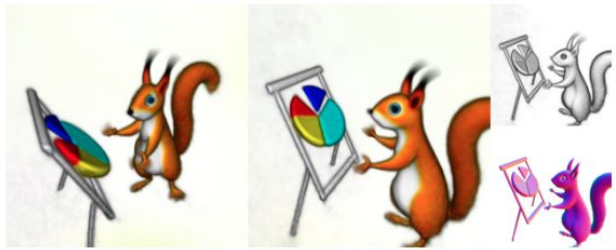


a classic Packard car*



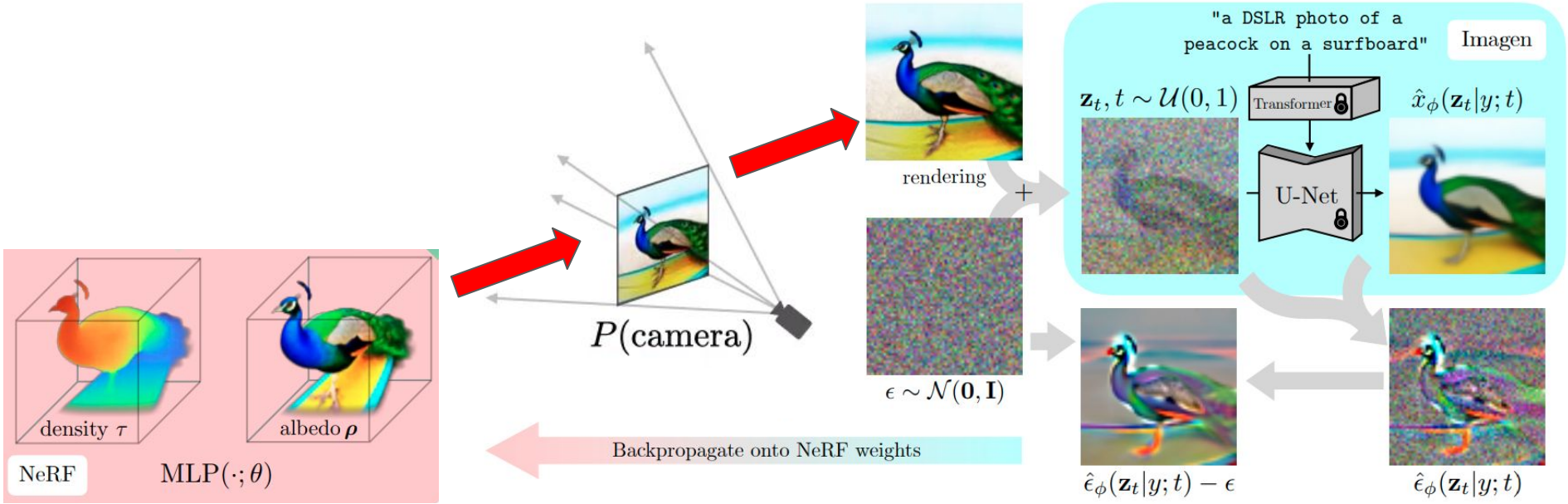
zoomed out view of Tower Bridge made out of gingerbread and candy‡

a robot and dinosaur playing chess, high resolution*



a squirrel gesturing in front of an easel showing colorful pie charts

DREAMFUSION, Poole et al. (2022)

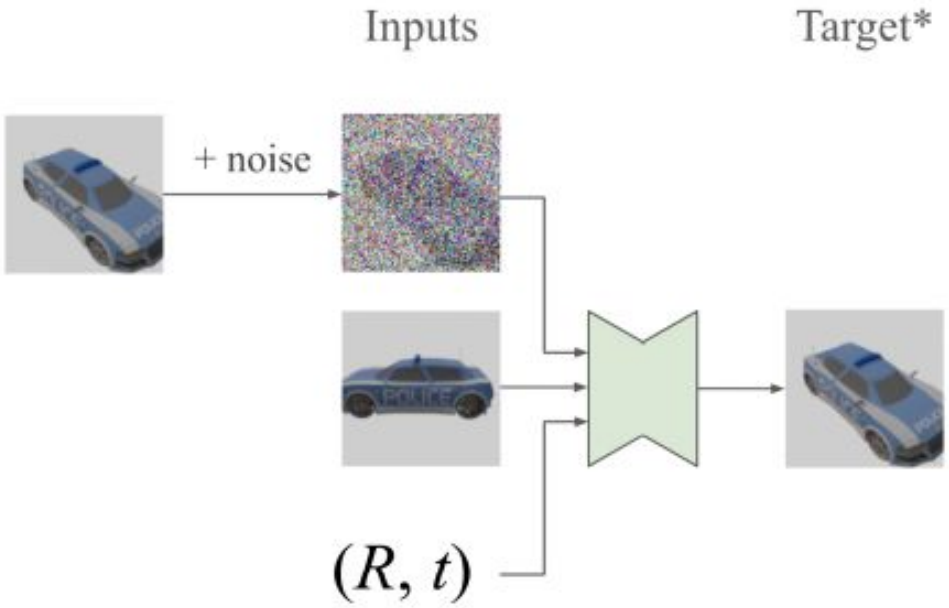


$$\nabla_\theta \mathcal{L}_{\text{SDS}}(\phi, \mathbf{x} = g(\theta)) \triangleq \mathbb{E}_{t,\epsilon}\left[w(t)\left(\hat{\epsilon}_\phi(\mathbf{z}_t; y, t) - \epsilon\right)\frac{\partial \mathbf{x}}{\partial \theta}\right]$$

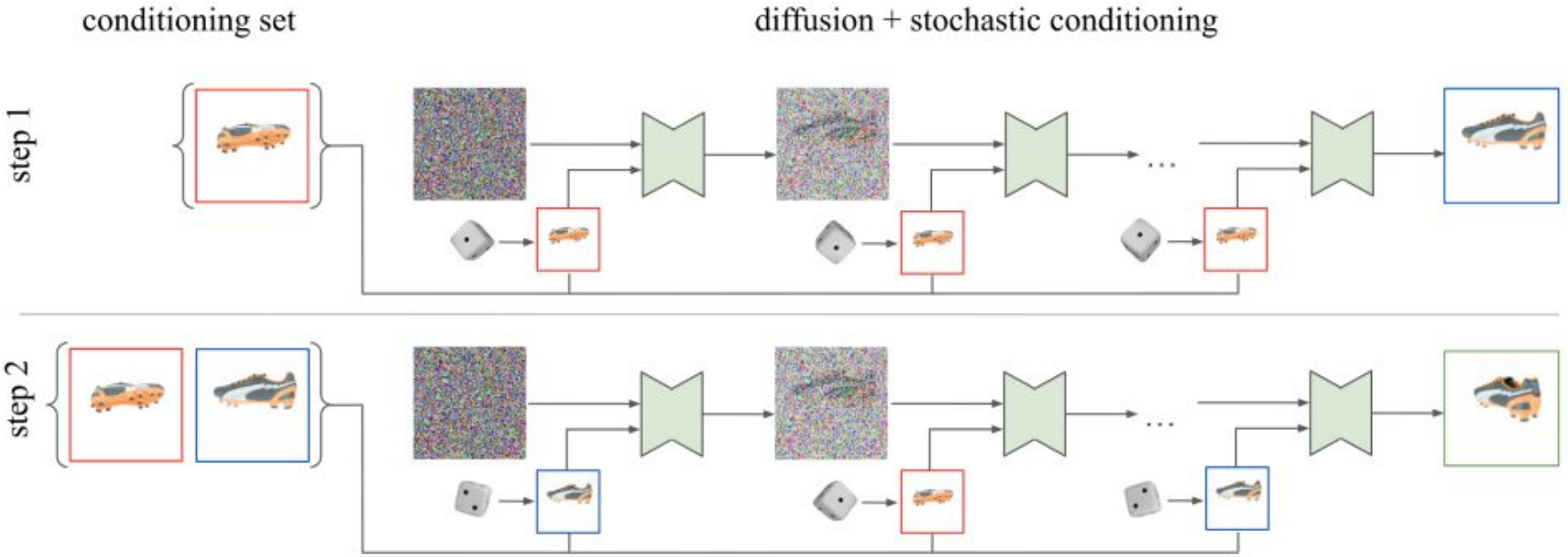
create 3D models that look like good images when rendered from random angles

3D-Diffusion, Watson et al. (2022)

3D-Diffusion, Watson et al. (2022)



3D-Diffusion, Watson et al. (2022)



craiyon

AI model drawing images from any prompt!

A smiley face, with a caption writing "thank you for your attention"