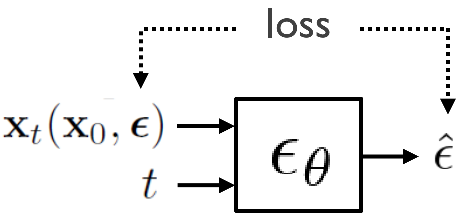


Overview of Modern ML Applications: Diffusion Models

Learning objectives:

- Be able to explain the principle of diffusion models and name common applications.
- Be able to follow the key derivation steps of diffusion models.

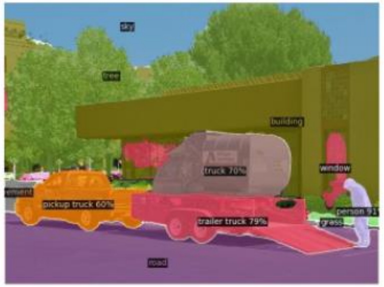
Acknowledgment: This slide deck was adapted from this CVPR 2023 tutorial by Song, Meng, and Vahdat. [Video recording]

Applications for Generative Models

Art & Design

Content Generation

Representation Learning

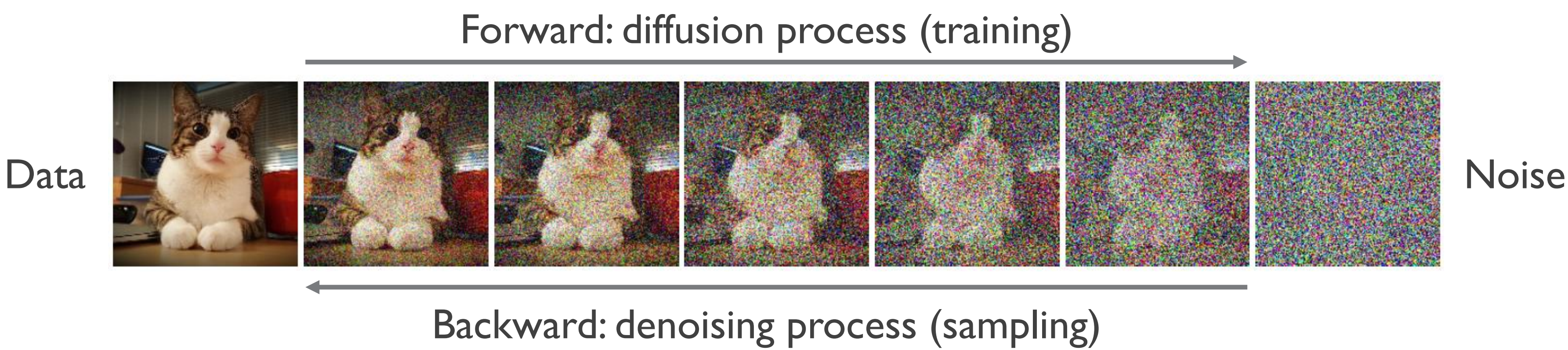


Entertainment

[Playground Demo]

Diffusion Model: Basic Idea

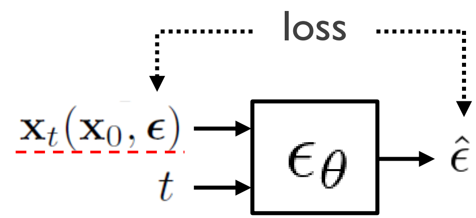
- Goal: Learning to generate by denoising.
- Diffusion model contains two processes:
  - Forward diffusion: Gradually adds noise and learns a denoising net.
  - Backward denoising: Reconstruct data via learned denoising net.



Diffusion Model: Algorithmic Perspective

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
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



Forward (training)

Data x_0

Noise

$t = 1$ $t = T$

Backward (sampling)

Algorithm 2 Sampling

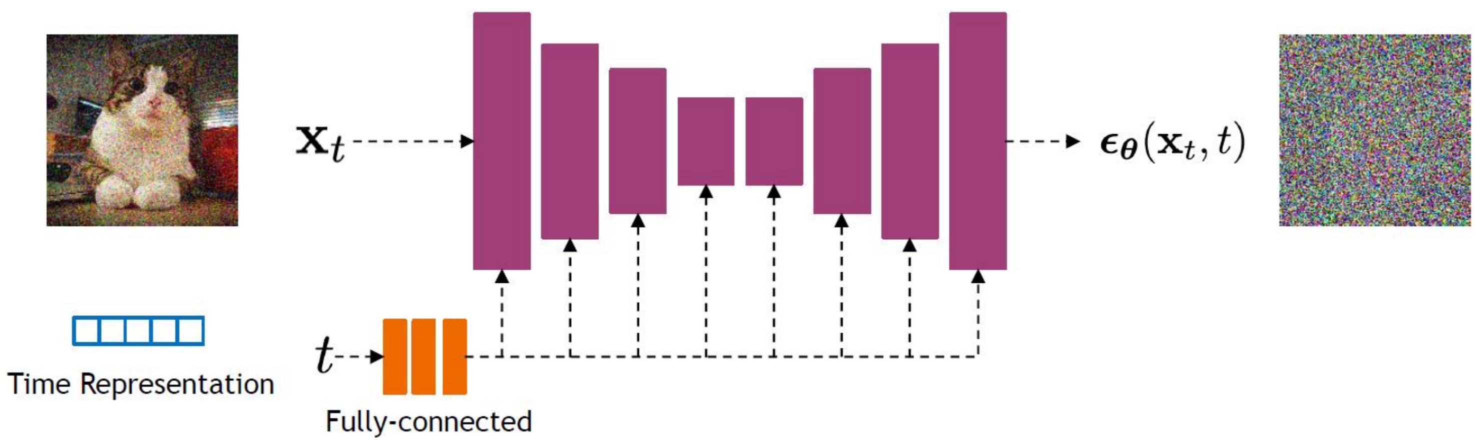
1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \ldots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t,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

DDPM, 5,000 citations as of Dec. 2023:

Ho, J., Jain, A., and Abbeel, P, "Denoising diffusion probabilistic models," NeurIPS, 2020.

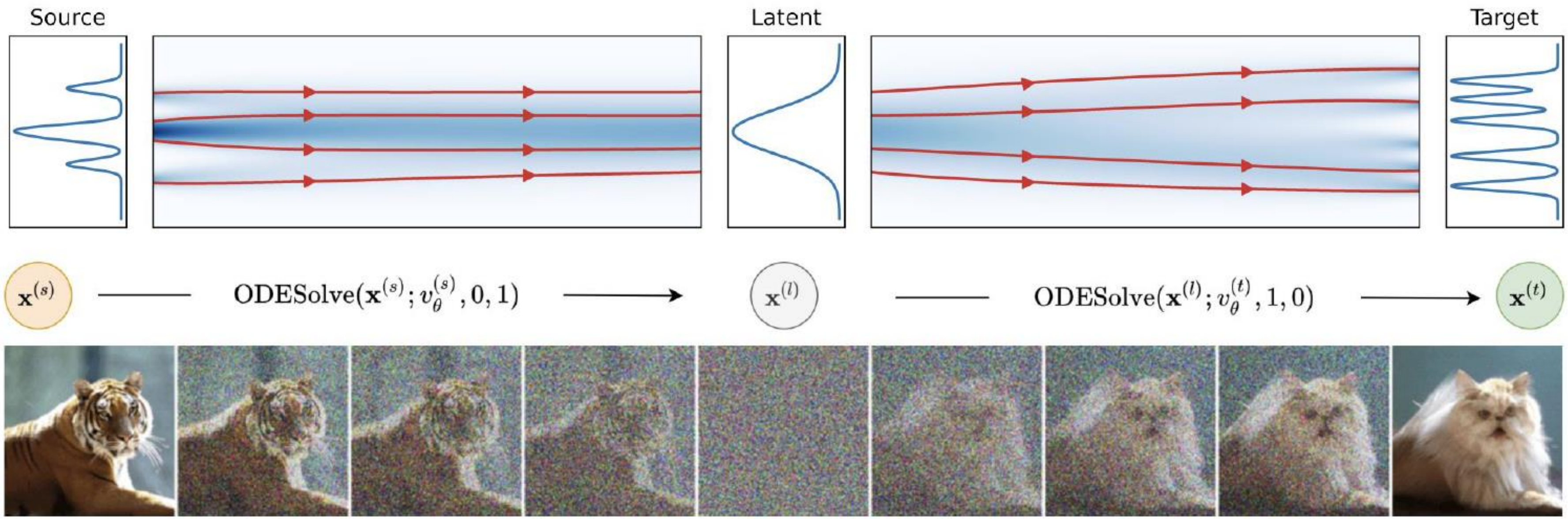
Diffusion Model: Implementation Details

- Diffusion models often use U-Net with ResNet blocks and self-attention layers.
- Time representation: Sinusoidal positional embeddings or random Fourier features.
- Time is fed to the residual blocks using (i) simple spatial addition or (ii) adaptive group normalization layers (Dharivwal & Nichol, 2021).



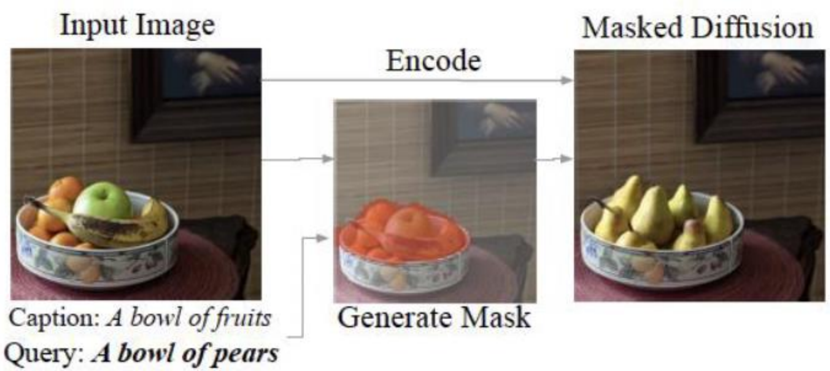
Dhariwal, P., and Nichol, A, “Diffusion models beat GANs on image synthesis,” NeurIPS, 2021.

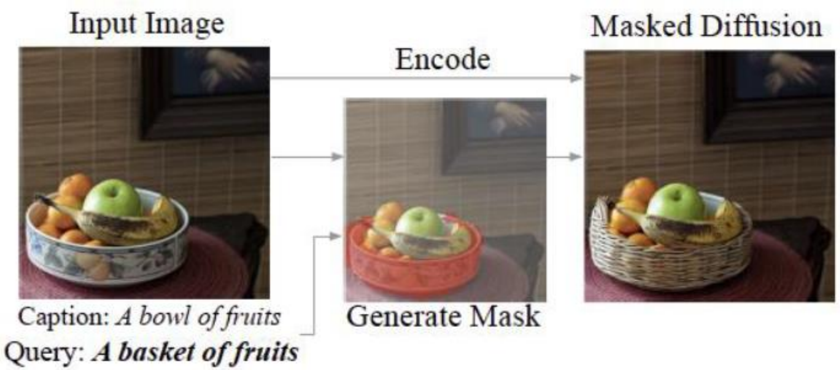
Ex: Style Transfer



Su, X., Song, J., Meng, C. and Ermon, S., "Dual diffusion implicit bridges for image to image translation," ICLR, 2023.

Ex: Semantic Editing with Mask Guidance (DiffEdit)

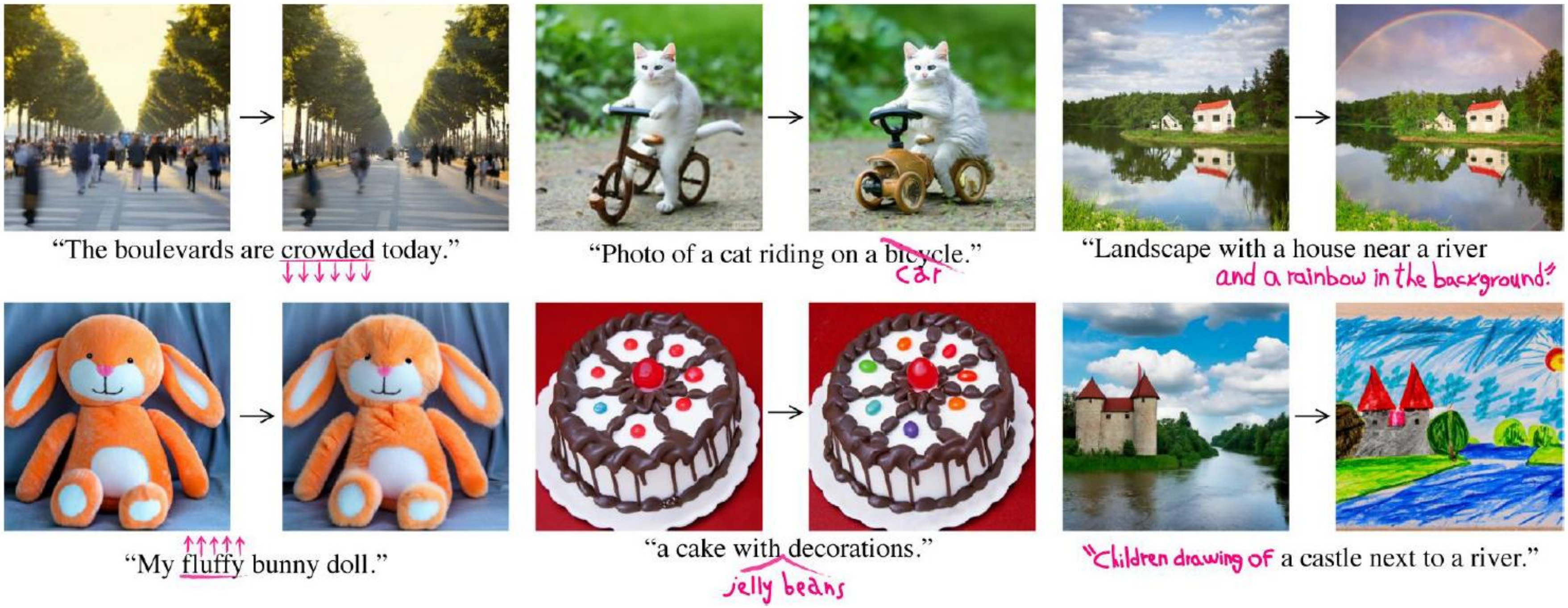




User-provided mask not needed: Model generates a mask based on caption & query.

Couairon, G., Verbeek, J., Schwenk, H., and Cord, M., “DiffEdit: Diffusion-based semantic image editing with mask guidance,” ICLR, 2023.

Ex: Prompt-to-Prompt Image Editing w/ Cross Attention Control

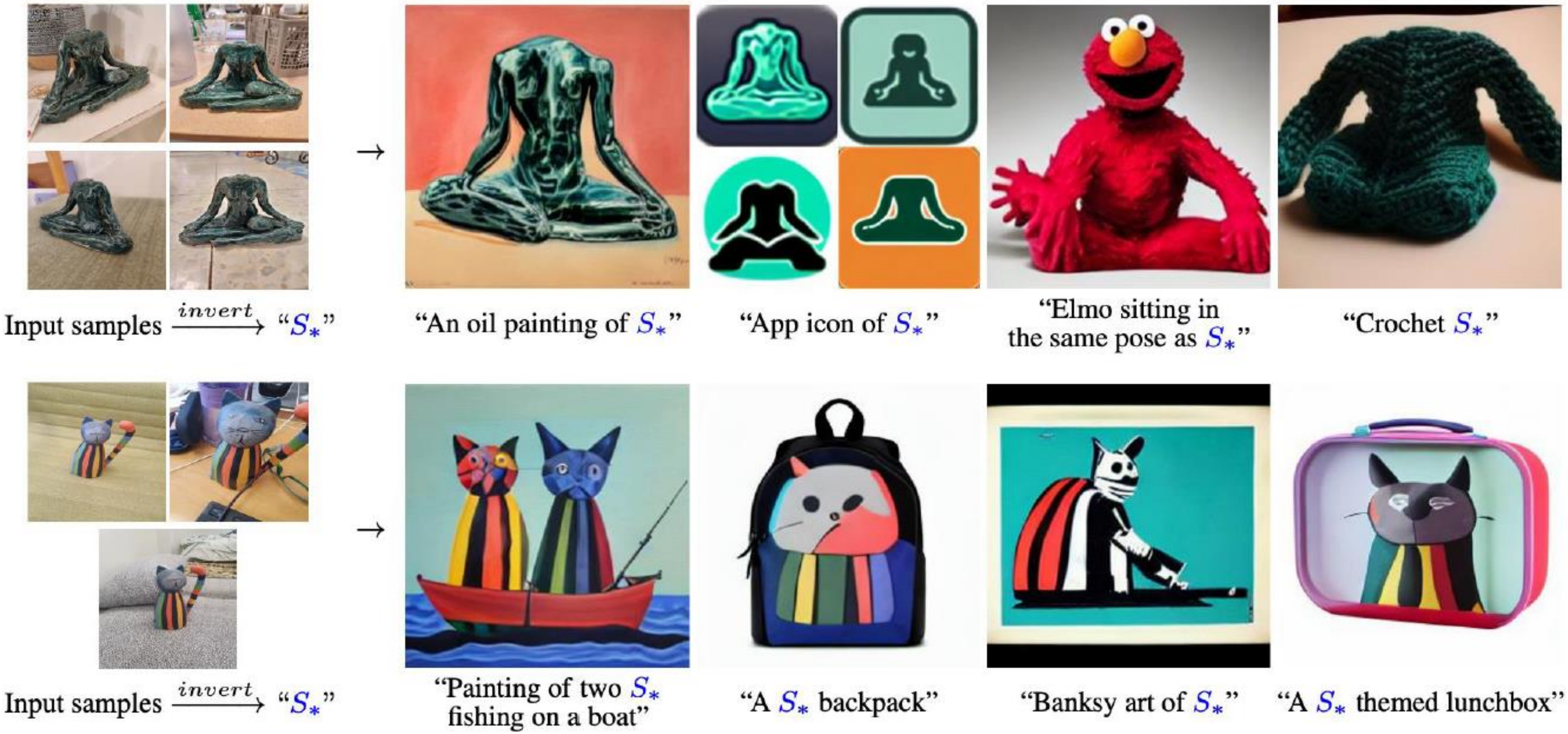


Hertz, A., Mokady, R., Tenenbaum, J., Aberman, K., Pritch, Y., and Cohen-Or, D, "Prompt to prompt image editing with cross attention control," ICLR, 2023.

Ex: Personalization with Diffusion Models

Ruiz, N., Li, Y., Jampani, V., Pritch, Y., Rubinstein, M., and Aberman, K., “DreamBooth: Fine tuning text to image diffusion models for subject driven generation,” CVPR, 2023.

Ex: Optimizing Text Embedding (Textual Inversion)

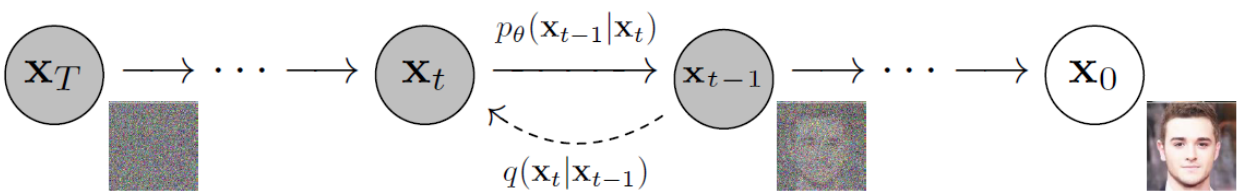


Gal, R., Alaluf, Y., Atzmon, Y., Patashnik, O., Bermano, A. H., Chechik, G., and Cohen-Or, D., "An image is worth one word: Personalizing text-to-image generation using textual inversion," ICLR, 2023.

Mathematical / Probabilistic Formulation

- Raw data model: $q(\mathbf{x}_0)$, where q denotes a PDF
- Diffusion model (parameterized by θ): $p_\theta(\mathbf{x}_0) := \int p_\theta(\mathbf{x}_{0:T})\, d\mathbf{x}_{1:T}$

Note $\mathbf{x}_{0:T} = \mathbf{x}_0, \ldots, \mathbf{x}_T$



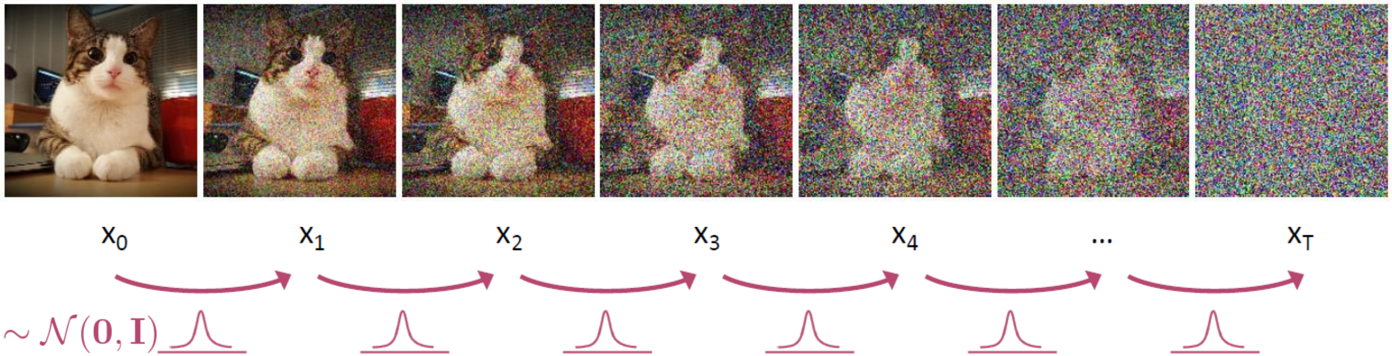
$$p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$$

$\mathbf{x}_T$ is Gaussian distributed
w/ mean $\mathbf{0}$ and covariance $\mathbf{I}$

Note the pipeline is horizontally flipped.

Ho, J., Jain, A., and Abbeel, P, “Denoising diffusion probabilistic models,” NeurIPS, 2020.

Forward Diffusion: Single Step



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

$$\mathbf{x}_1 = \sqrt{\alpha_1}\mathbf{x}_0 + \sqrt{\beta_1}\mathbf{e}_1 \quad \cdots \quad \mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\mathbf{e}_t \quad \cdots \quad \mathbf{x}_T = \sqrt{\alpha_T}\mathbf{x}_{T-1} + \sqrt{\beta_T}\mathbf{e}_T$$

Intuition: Image intensity is scaled down and white Gaussian noise is added to corrupt the image. Variance of the raw image is 1. Variances of the noisy images are maintained to be 1.

$\alpha_t, \beta_t \in (0, 1)$

$\alpha_t + \beta_t = 1$

Alternatively, one-step conditional distribution (1st-order Markov chain) can be written as:

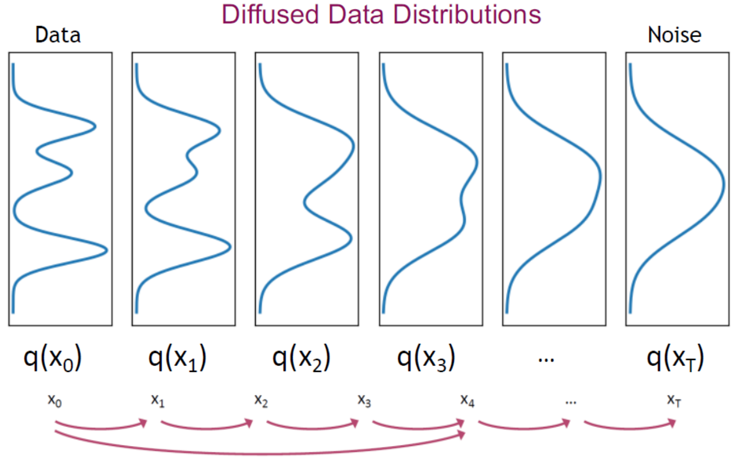
$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

Forward Diffusion: Arbitrary Steps

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\,\epsilon \quad \text{where } \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \quad \bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$$

Validation for $\mathbf{x}_2$:

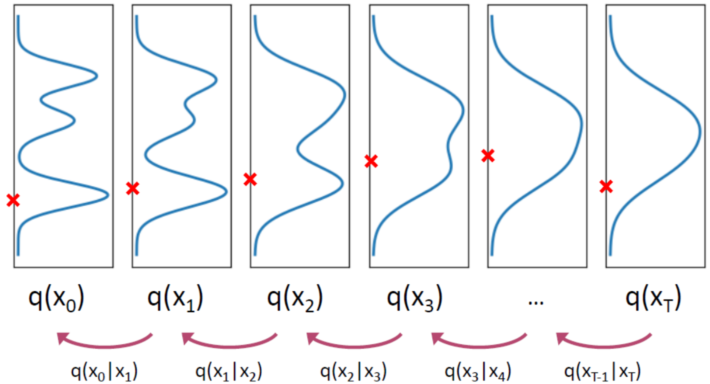
$$\begin{aligned}
\mathbf{x}_2 &= \sqrt{\alpha_2}\mathbf{x}_1 + \sqrt{\beta_2}\mathbf{e}_2 \\
&= \sqrt{\alpha_2}\left(\sqrt{\alpha_1}\mathbf{x}_0 + \sqrt{\beta_1}\mathbf{e}_1\right) + \sqrt{\beta_2}\mathbf{e}_2 \\
&= \sqrt{\alpha_2}\sqrt{\alpha_1}\mathbf{x}_0 + \left(\sqrt{\alpha_2}\sqrt{\beta_1}\mathbf{e}_1 + \sqrt{\beta_2}\mathbf{e}_2\right) \\
&= \sqrt{\bar{\alpha}_2}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_2}\,\mathbf{e}_2'
\end{aligned}$$



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

$\beta_t \in (0,1)$ ensures that for large T (e.g., $T = 1000$), $\bar{\alpha}_T \to 0$ and $q(\mathbf{x}_T|\mathbf{x}_0) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I}))$

Backward Denoising



Algorithm 2 Sampling

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: $\quad \mathbf{x}_{t-1} = \mu_\theta(\mathbf{x}_t, t) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

Step-by-step reconstruction

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

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

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

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

Use NN as a function approximator

Note: 1. If random variables are jointly Gaussian, then any conditional distribution is also Gaussian.
2. $\mathbf{x}_0$ is not Gaussian, so approximation is needed.

What and How to Train?

- Due to the following relation, may use another network to approximate $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$ instead of $\mu_\theta(\mathbf{x}_t, t)$:

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

- Loss function:

$$\ell=\mathbb{E}_{\mathbf{x}_0 \sim q\left(\mathbf{x}_0\right), t \sim \mathrm{U}[1, T], \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})}\left\|\epsilon-\epsilon_\theta\left(\underbrace{\sqrt{\bar{\alpha}_t} \mathbf{x}_0+\sqrt{1-\bar{\alpha}_t} \epsilon}_{=\mathbf{x}_t}, t\right)\right\|_2^2$$

Forward Diffusion: Forming a Training Data Pair

Step 1: Draw an image $\mathbf{x}_0$ from $q(\cdot)$

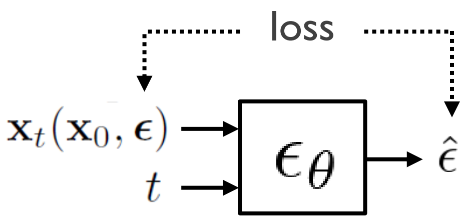
Step 2: Pick a time step t

Step 3: Create a noisy image $\mathbf{x}_t \sim q(\mathbf{x}_t \mid \mathbf{x}_0)$ by fast-forwarding t steps via

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

Algorithm 1 Training

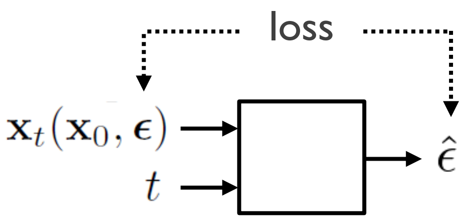
repeat
  $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
  $t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  Take gradient descent step on
    $\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0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



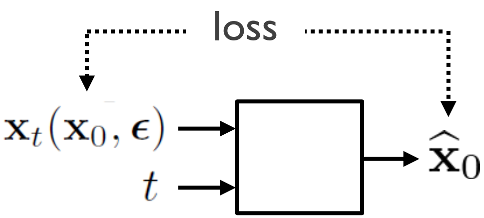
Data: $\left(\mathbf{x}_t(\mathbf{x}_0, \epsilon), t\right)$

Label: ϵ

BTW, OpenAI Uses a Slightly Different Loss



DDPM (Ho et al., 2020)



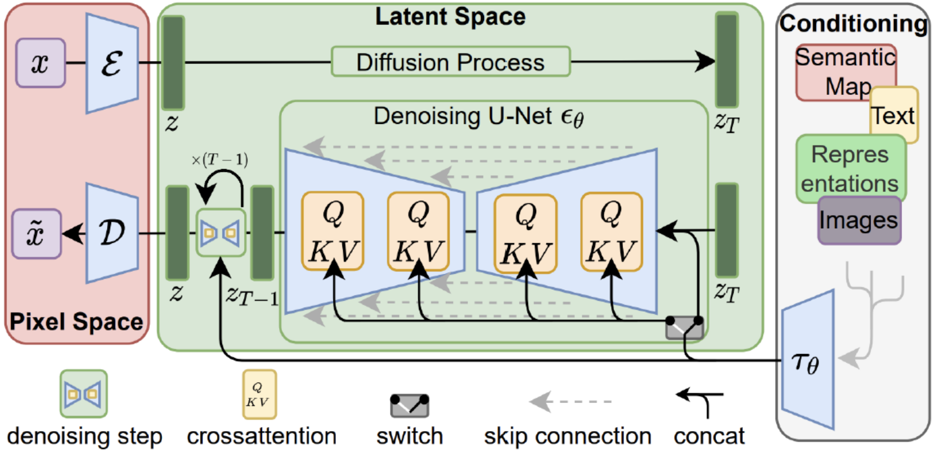
DALL-E 2 (Ramesh et al., 2022)

One-step denoising w/ knowledge of step t

Ho, J., Jain, A., and Abbeel, P, "Denoising diffusion probabilistic models," NeurIPS, 2020.
Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M., "Hierarchical text-conditional image generation with CLIP latents," arXiv:2204.06125, 2022.

Latent/Stable Diffusion

- Idea: Use an encoder to map the input data to an embedding space so that denoising diffusion is done in the latent space.



- Advantages:
  - Embeddings are closer to normal distribution => More correct modeling assumption, simpler denoising, faster synthesis.
  - Latent space => More expressivity and flexibility in design.
  - Tailored Autoencoders => Application to any data type (graphs, text, 3D data, etc.)

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B., “High-resolution image synthesis with latent diffusion models,” CVPR, 2022.

Conditioning the Diffusion Models

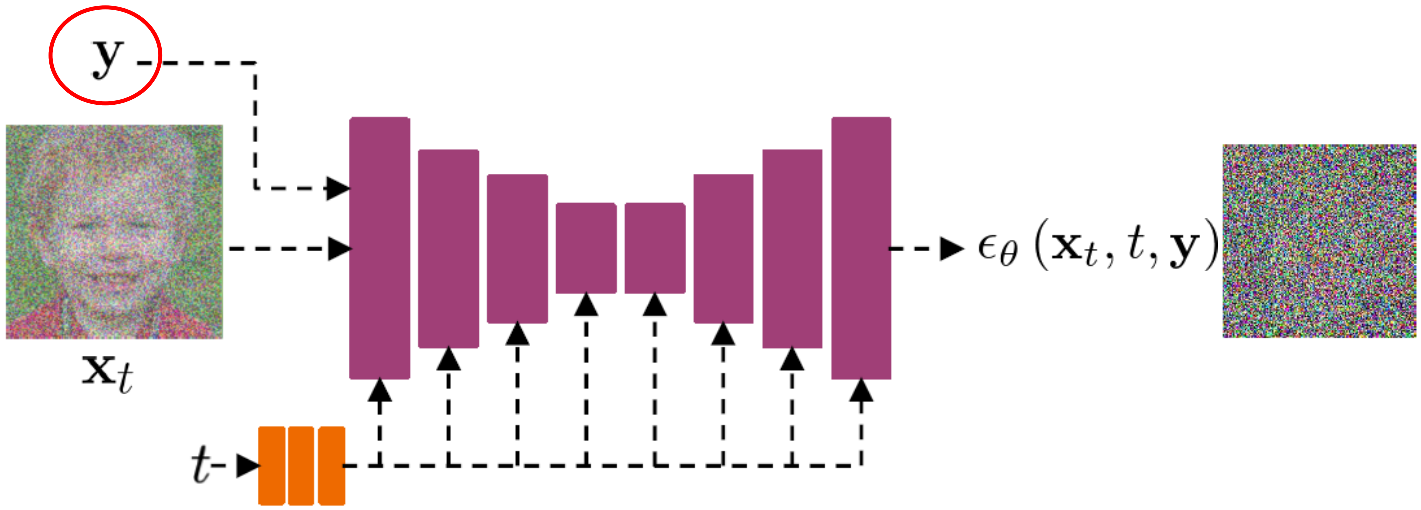
DALL·E 2

"a propaganda poster depicting a cat dressed as french emperor napoleon holding a piece of cheese"

IMAGEN

"A photo of a raccoon wearing an astronaut helmet, looking out of the window at night."

Treating Side Information y as Another Input



$$\ell = \mathbb{E}_{(\mathbf{x}_0,\mathbf{y})\sim q(\mathbf{x}_0,\mathbf{y}),t\sim \mathrm{U}[1,T],\epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})} \left\|\epsilon - \epsilon_\theta\left(\mathbf{x}_t, t, \mathbf{y}\right)\right\|_2^2$$