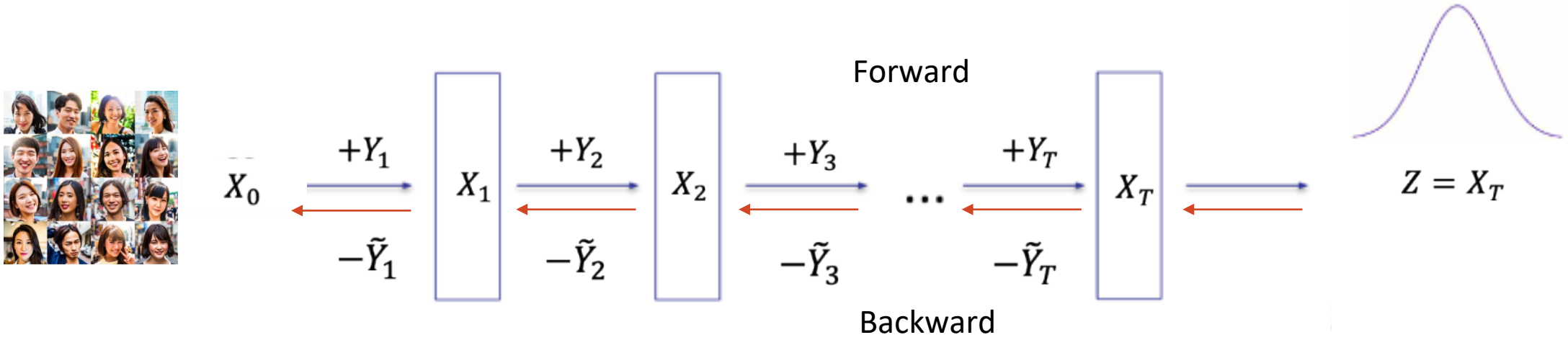
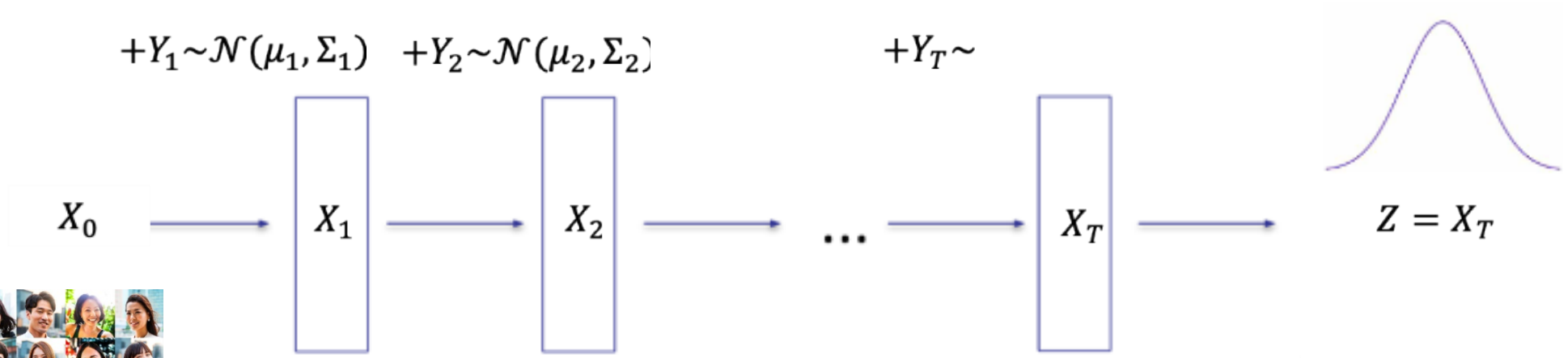
# Forward: Diffusion Process

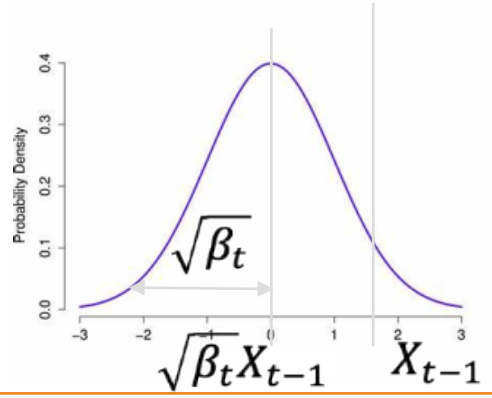



https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

# Forward: Diffusion Process




$$\mu_t = (\sqrt{\beta_t} - 1)X_{t-1},$$
$$\Sigma_t = \beta_t I$$
$$\beta_t \in (0,1)$$




https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Jonathan Ho, Ajay Jain, and Pieter Abbeel. “Denoising diffusion probabilistic models”. In: Advances in Neural Information Processing Systems 33 (2020), pp. 6840–6851.

# Forward: Central Limit Theorem

$+Y_1 \sim \mathcal{N}(\mu_1, \Sigma_1)$ $+Y_2 \sim \mathcal{N}(\mu_2, \Sigma_2)$ $+Y_T \sim$

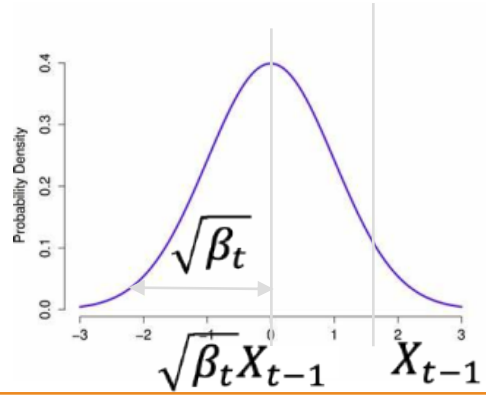
$X_0 \rightarrow X_1 \rightarrow X_2 \rightarrow \cdots \rightarrow X_T \rightarrow$

$$Z = X_T = X_0 + Y_1 + \cdots + Y_T$$

$$\sim \mathcal{N}\left(\sqrt{\theta}x_0, (1-\theta)I\right)$$

Diffusion Diffusion Diffusion

$$\mu_t = (\sqrt{\beta_t} - 1)X_{t-1},$$
$$\Sigma_t = \beta_t I$$
$$\beta_t \in (0,1)$$




$$\Pr(Z|x_0) = \mathcal{N}\left(\sqrt{\theta}x_0, (1-\theta)I\right)$$

$$\theta = \prod_{\tau=1}^{T}(1-\beta_\tau)$$

https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Jonathan Ho, Ajay Jain, and Pieter Abbeel. “Denoising diffusion probabilistic models”. In: Advances in Neural Information Processing Systems 33 (2020), pp. 6840–6851.

# Forward: Central Limit Theorem

$+Y_1 \sim \mathcal{N}(\mu_1, \Sigma_1)$ $+Y_2 \sim \mathcal{N}(\mu_2, \Sigma_2)$ $+Y_T \sim$

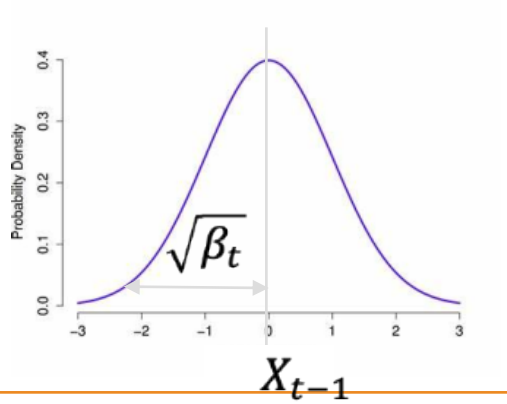
$X_0 \rightarrow X_1 \rightarrow X_2 \rightarrow \cdots \rightarrow X_T \rightarrow$

$$Z = X_T = X_0 + Y_1 + \cdots + Y_T$$

$$\sim \mathcal{N}\left(\sqrt{\theta} X_0, (1-\theta) I\right)$$

Diffusion Diffusion Diffusion

$$\mu_t = (\sqrt{\beta_t} - 1) X_{t-1},$$
$$\Sigma_t = \beta_t I$$
$$\beta_t \in (0,1)$$




$$\theta = \prod_{\tau=1}^{T} (1 - \beta_\tau)$$

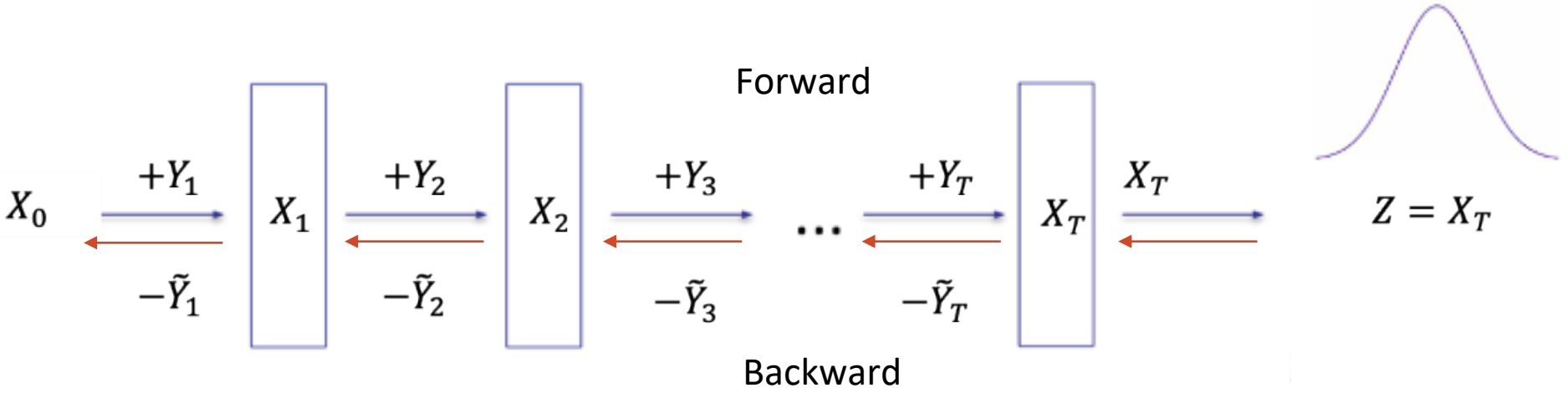
Usually,

$$\beta_1 < \beta_2 < \cdots < \beta_T$$

https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Jonathan Ho, Ajay Jain, and Pieter Abbeel. “Denoising diffusion probabilistic models”. In: Advances in Neural Information Processing Systems 33 (2020), pp. 6840–6851.

# Backward: Reverse Diffusion Models




$$\tilde{Y}_t \sim \mathcal{N}\left(X_{t-1}; \tilde{\mu}(x_t, t), \Sigma(x_t, t)\right)$$

$$\tilde{\mu}(x_t, t) = \left(1 - \frac{1}{\sqrt{1-\beta_t}}\right) x_t + \frac{\beta_t}{\sqrt{(1-\beta_t)(1-\theta_t)}} \epsilon_t$$

$$\theta_t = \prod_{\tau=1}^{t} (1-\beta_\tau)$$

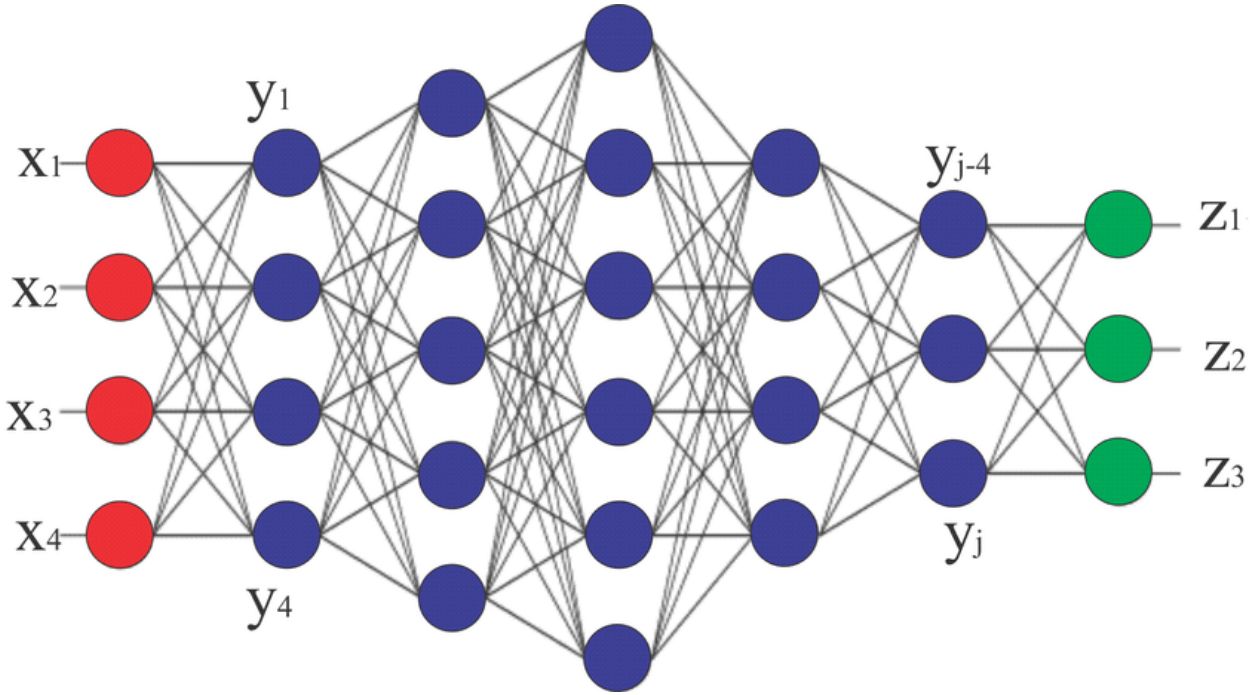
$$\Sigma_\theta(x_t, t) = \frac{1-\theta_{t-1}}{1-\theta_t} \cdot \beta_t \cdot I$$

$$\epsilon_t \sim \mathcal{N}(0, I)$$

https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Jonathan Ho, Ajay Jain, and Pieter Abbeel. “Denoising diffusion probabilistic models”. In: Advances in Neural Information Processing Systems 33 (2020), pp. 6840–6851.

# Backward: Reverse Diffusion Models




$$\tilde{Y}_t \sim \mathcal{N}\left(X_{t-1}; \tilde{\mu}(x_t, t), \Sigma(x_t, t)\right)$$

$$\tilde{Y}_t^W \sim \mathcal{N}\left(X_{t-1}; \tilde{\mu}_W(x_t, t), \Sigma_W(x_t, t)\right)$$

https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Jonathan Ho, Ajay Jain, and Pieter Abbeel. “Denoising diffusion probabilistic models”. In: Advances in Neural Information Processing Systems 33 (2020), pp. 6840–6851.

# Backward: Reverse Diffusion Models

$$\tilde{Y}_t \sim \mathcal{N}\left(X_{t-1}; \tilde{\mu}(x_t, t), \Sigma(x_t, t)\right) \quad \Leftarrow \quad \tilde{Y}_t^W \sim \mathcal{N}\left(X_{t-1};\ \tilde{\mu}_W(x_t, t), \Sigma_W(x_t, t)\right)$$

$$\tilde{\mu}(x_t, t) = \left(1 - \frac{1}{\sqrt{1-\beta_t}}\right) x_t + \frac{\beta_t}{\sqrt{(1-\beta_t)(1-\theta_t)}} \epsilon_t \quad \Leftarrow \quad \tilde{\mu}_W(x_t, t) = \left(1 - \frac{1}{\sqrt{1-\beta_t}}\right) x_t + \frac{\beta_t}{\sqrt{(1-\beta_t)(1-\theta_t)}} \epsilon_t^W(x_t, t)$$

$$\Sigma(x_t, t) \quad \Leftarrow \quad \Sigma_W(x_t, t)$$

https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Jonathan Ho, Ajay Jain, and Pieter Abbeel. “Denoising diffusion probabilistic models”. In: Advances in Neural Information Processing Systems 33 (2020), pp. 6840–6851.

# Loss Function

$$L = -\mathbb{E}_{X_0} \log p_\theta(x_0) = \mathbb{E}_{X_0} \log \left( \int p_\theta(x_{0:T}) dx_{1:T} \right) \leq \mathbb{E}_{X_{0:T}} \left( \sum_{t=2}^{T} L_t + L_0 + C \right)$$

$$L_0 = -\log \mathcal{N}\left(X_0; x_1 - \tilde{\mu}_W(x_1, 1), \Sigma_W(x_1, 1)\right)$$

$$L_t = D_{KL}\left(\mathcal{N}\left(X_{t-1}; x_t - \tilde{\mu}(x_t, t), \Sigma(x_t, t)\right) || \mathcal{N}\left(X_{t-1}; x_t - \tilde{\mu}_W(x_t, t), \Sigma_W(x_t, t)\right)\right)$$
$$= \mathbb{E}_{X_0, \epsilon}\left[\frac{\beta_t^2}{2(1-\beta_t)(1-\theta_t)\|\Sigma_W\|_2^2} \left\|\epsilon_t - \epsilon_W\left(\sqrt{\theta_t} x_0 + \sqrt{1-\theta_t}\epsilon_t, t\right)\right\|^2\right]$$
$$\approx \mathbb{E}_{X_0, \epsilon}\left[\left\|\epsilon_t - \epsilon_W\left(\sqrt{\theta_t} x_0 + \sqrt{1-\theta_t}\epsilon_t, t\right)\right\|^2\right]$$

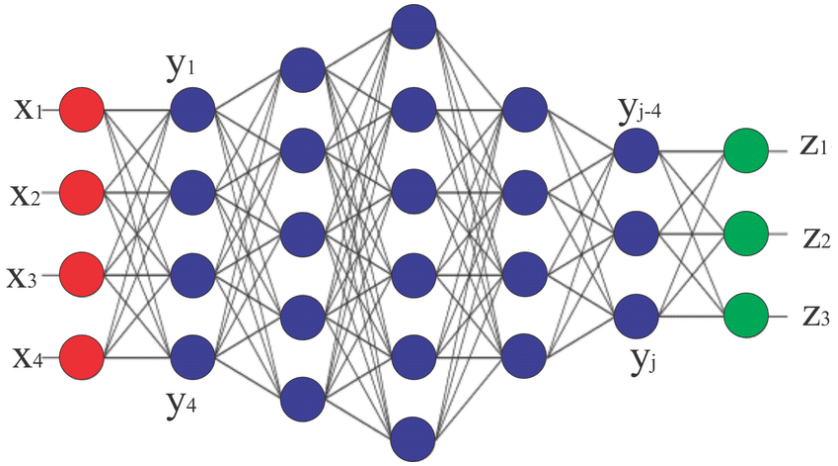
$$L \approx \mathbb{E}_{t\sim[1,T], X_0, \epsilon}\left[\left\|\epsilon_t - \epsilon_W\left(\sqrt{\theta_t} x_0 + \sqrt{1-\theta_t}\epsilon_t, t\right)\right\|^2\right]$$

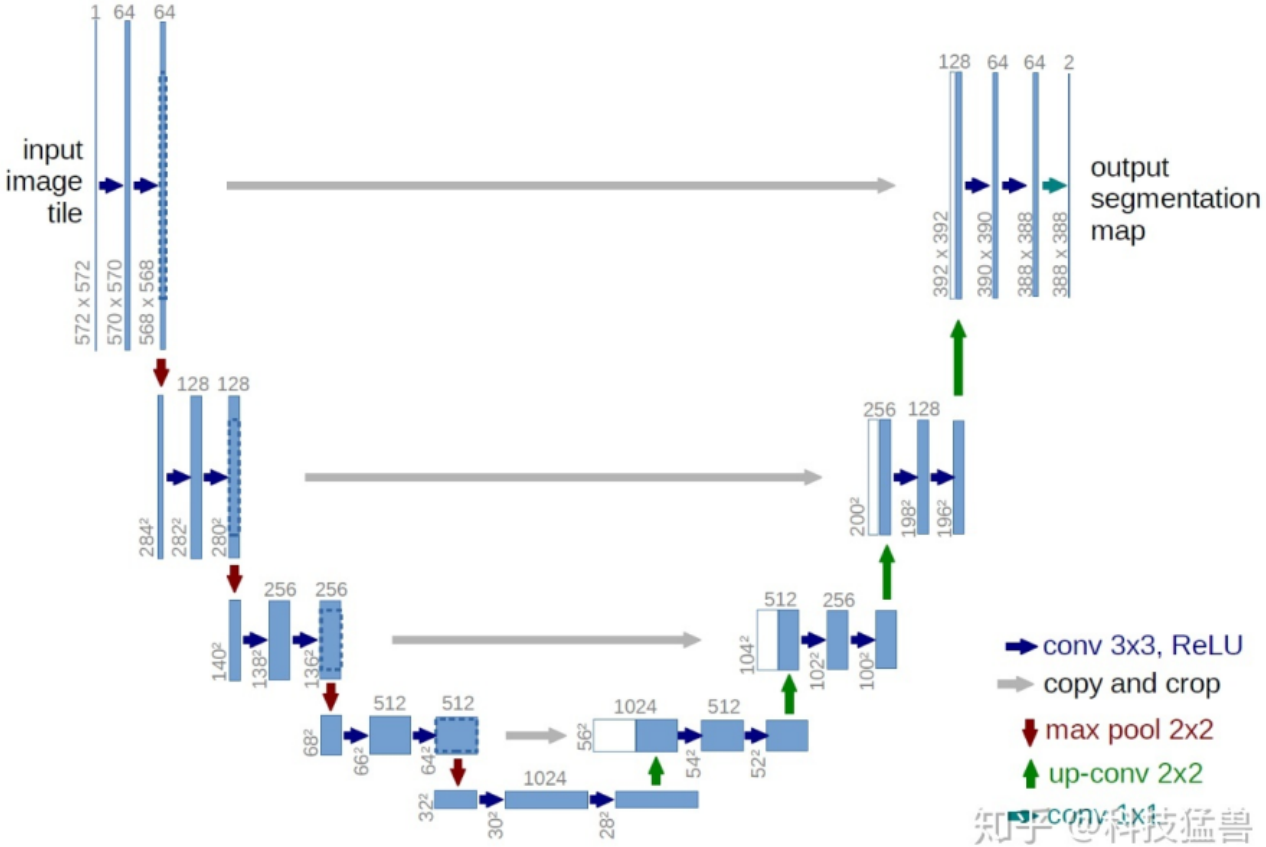
https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Jonathan Ho, Ajay Jain, and Pieter Abbeel. “Denoising diffusion probabilistic models”. In: Advances in Neural Information Processing Systems 33 (2020), pp. 6840–6851.

# U-Net

$\epsilon_W$




知乎 @科技猛兽

https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Jonathan Ho, Ajay Jain, and Pieter Abbeel. “Denoising diffusion probabilistic models”. In: Advances in Neural Information Processing Systems 33 (2020), pp. 6840–6851.

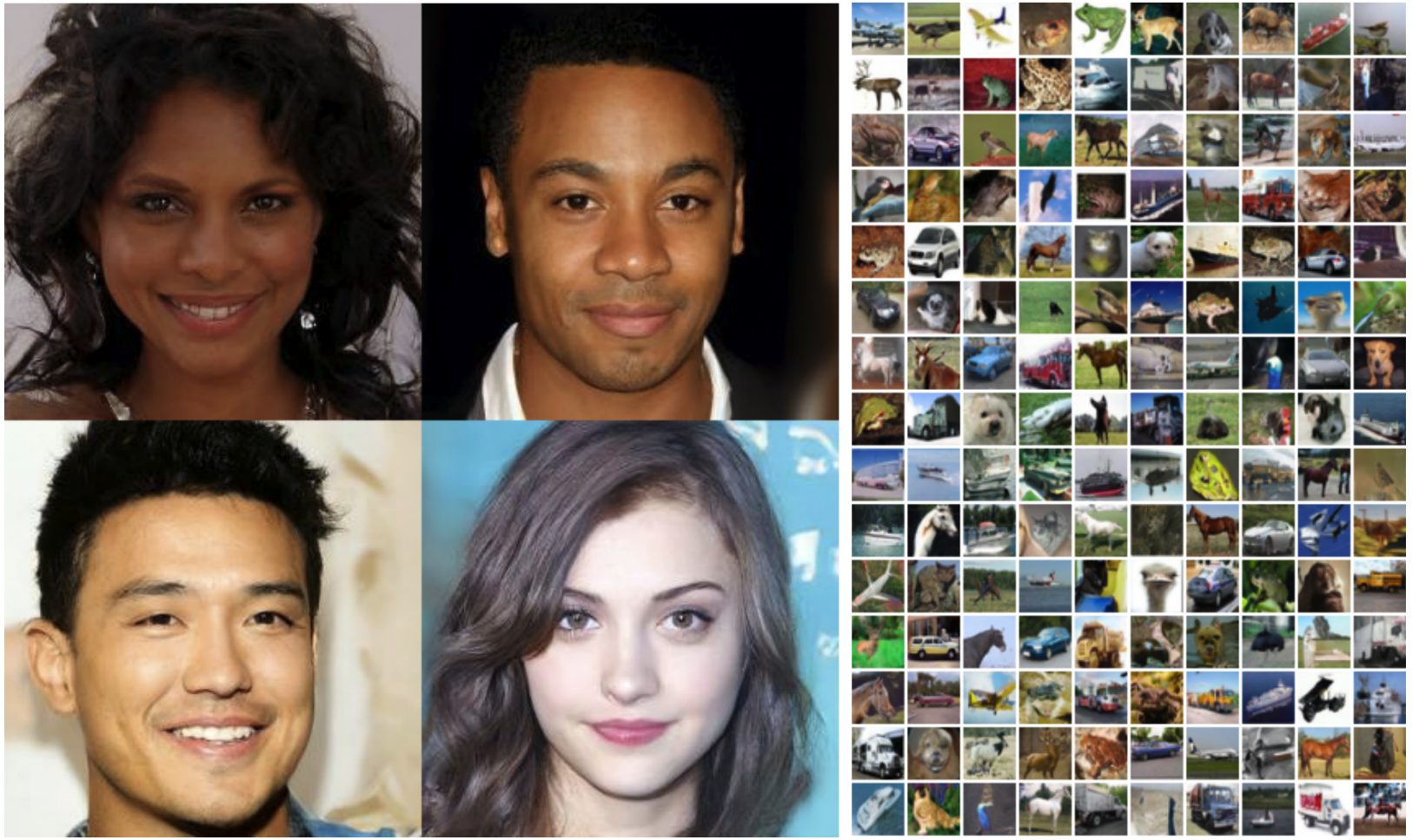
# Algorithms

| | **Algorithm 1 Training** | | **Algorithm 2 Sampling** |
|---|---|---|---|
| 1 | repeat | 1 | $x_T \sim \mathcal{N}(0, I)$ |
| 2 | $x_0 \sim P_{data}(X_0)$ | 2 | for t=T,…,1 do |
| 3 | $t \sim \text{Uniform}(\{1, \dots, T\})$ | 3 | $z \sim \mathcal{N}(0, I)$ if t>1, else z=0 |
| 4 | $\epsilon \sim \mathcal{N}(0, I)$ | 4 | $x_{t-1} = \frac{1}{\sqrt{1-\beta_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\theta_t}}\epsilon_W(x_t, t)\right) + \beta_t z$ |
| 5 | Take gradient descent step on | 5 | end for |
| | $\nabla_W \left\| \epsilon_t - \epsilon_W\left(\sqrt{\theta_t}x_0 + \sqrt{1-\theta_t}\epsilon_t, t\right)\right\|^2$ | 6 | return $x_0$ |
| 6 | until converged | | |

https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Jonathan Ho, Ajay Jain, and Pieter Abbeel. “Denoising diffusion probabilistic models”. In: Advances in Neural Information Processing Systems 33 (2020), pp. 6840–6851.

# Results



https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Jonathan Ho, Ajay Jain, and Pieter Abbeel. “Denoising diffusion probabilistic models”. In: Advances in Neural Information Processing Systems 33 (2020), pp. 6840–6851.

OpenAI Research Product Developers Safety Company

# DALL·E 2

DALL·E 2 is an AI system that can create realistic images and art from a description in natural language.

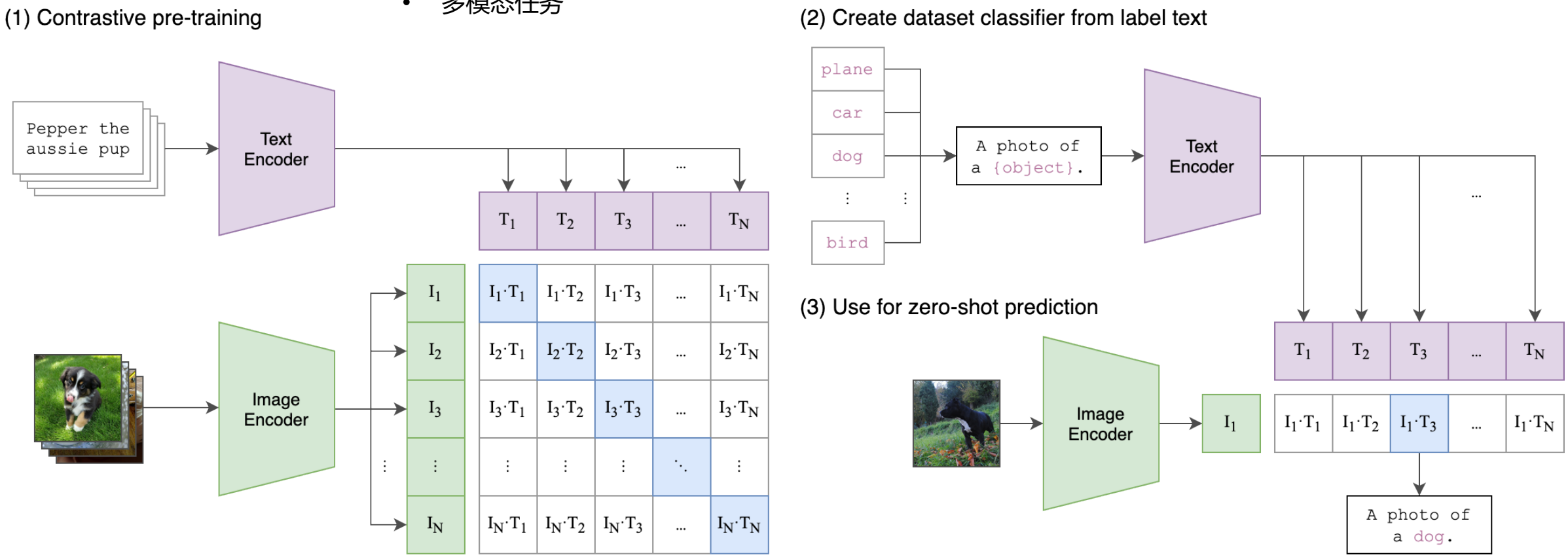
Try DALL·E ↗ Follow on Instagram ↗

An astronaut riding a horse in photorealistic style.

# Clip

- 数据：4亿个 图片文本对
- 可以做zero-shot推理
- 需要预训练
- 多模态任务

预训练用句子做的，如果后面句子只有一个单词需要判断




http://arxiv.org/abs/2103.00020

# DALL-E

- 现有对比学习方法（如CLIP）得到的特征是非常稳健的，可以同时捕获语义信息和风格特征；
- 用这个特征来做图像生成；
- DALLE是一个two-stage的模型：prior和decoder

## 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

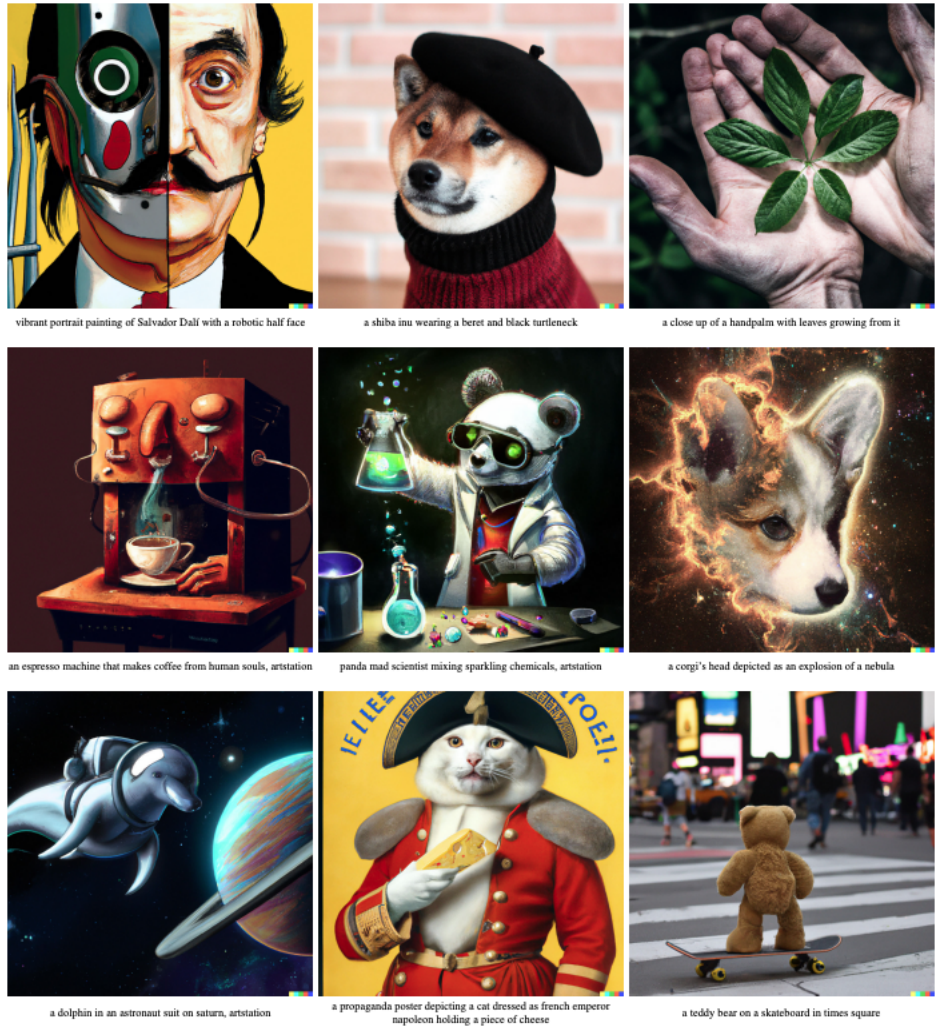
**Casey Chu*** OpenAI casey@openai.com

**Mark Chen** OpenAI mark@openai.com



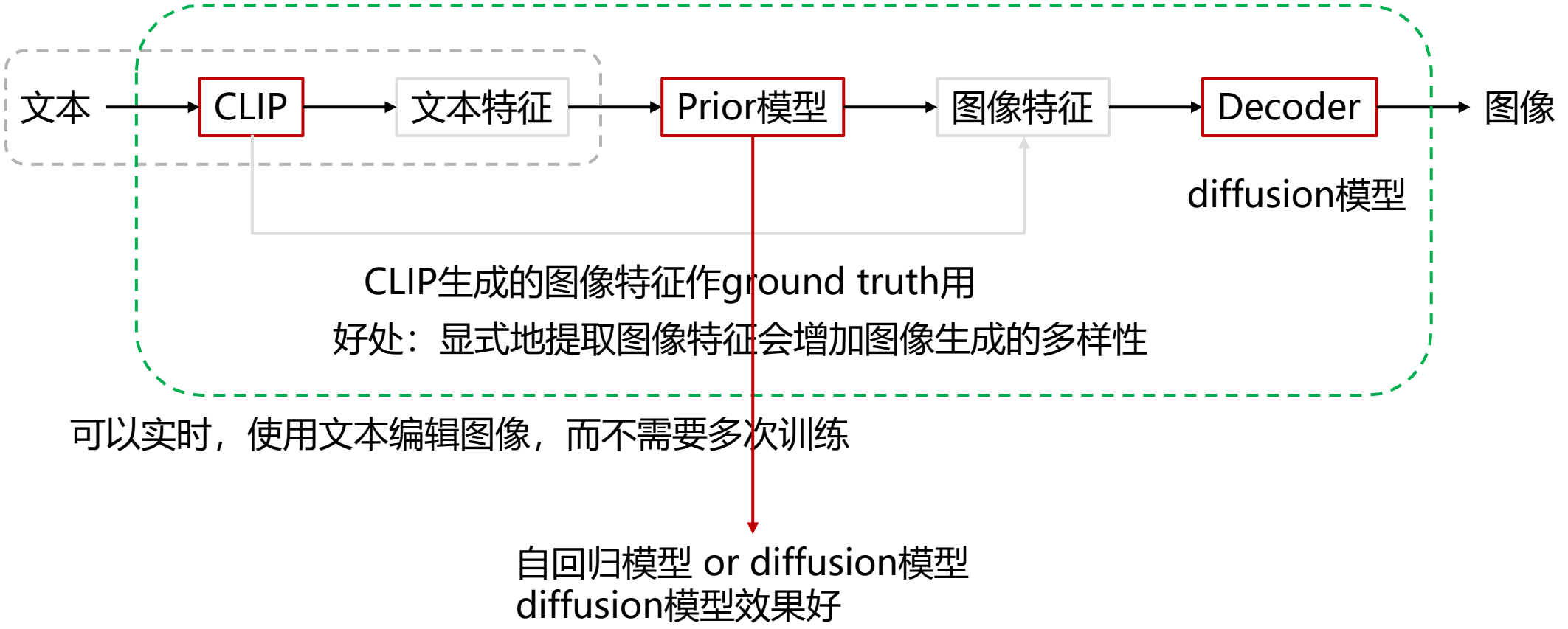
### Abstract

Contrastive models like CLIP have been shown to learn robust representations of images that capture both semantics and style. To leverage these representations for image generation, we propose a two-stage model: a prior that generates a CLIP image embedding given a text caption, and a decoder that generates an image conditioned on the image embedding. We show that explicitly generating image representations improves image diversity with minimal loss in photorealism and caption similarity. Our decoders conditioned on image representations can also produce variations of an image that preserve both its semantics and style, while varying the non-essential details absent from the image representation. Moreover, the joint embedding space of CLIP enables language-guided image manipulations in a zero-shot fashion. We use diffusion models for the decoder and experiment with both autoregressive and diffusion models for the prior, finding that the latter are computationally more efficient and produce higher-quality samples.

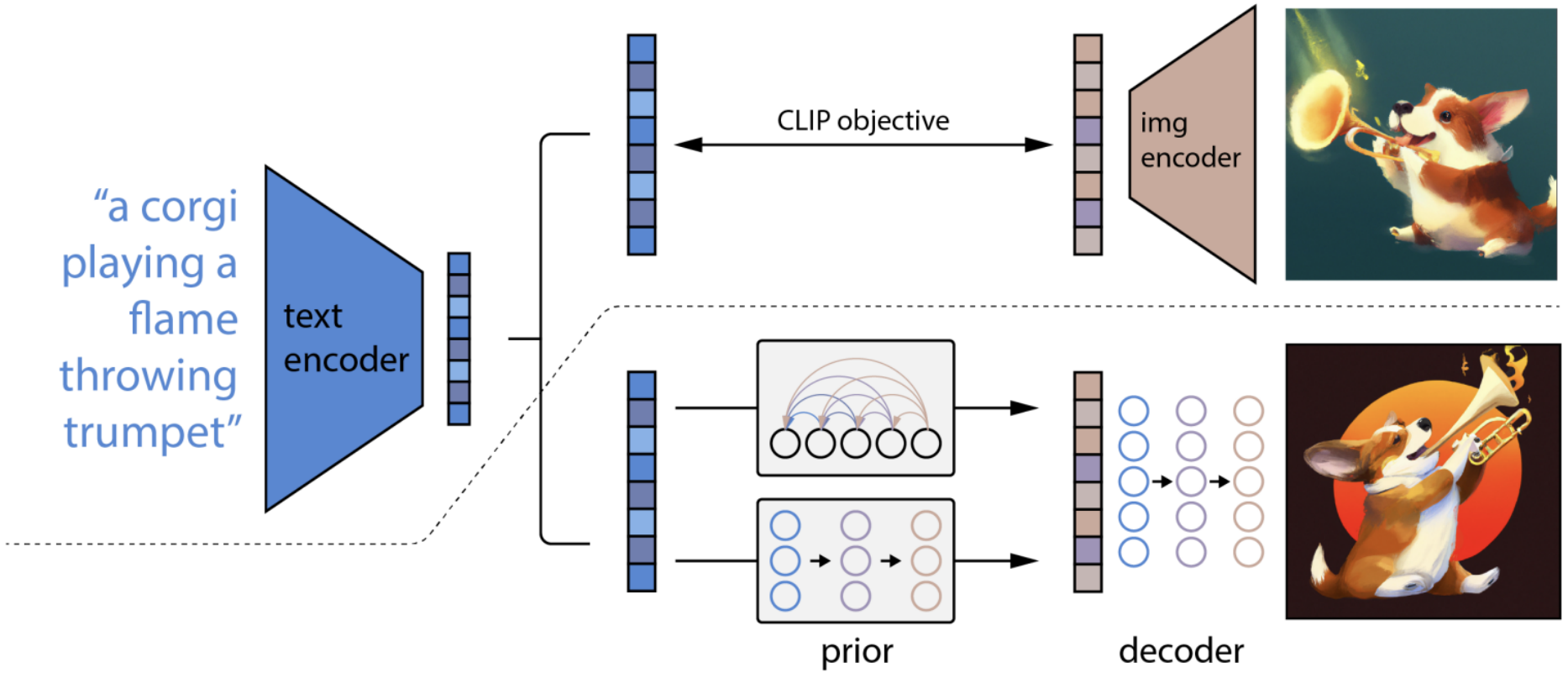

https://arxiv.org/pdf/2204.06125.pdf

# Clip



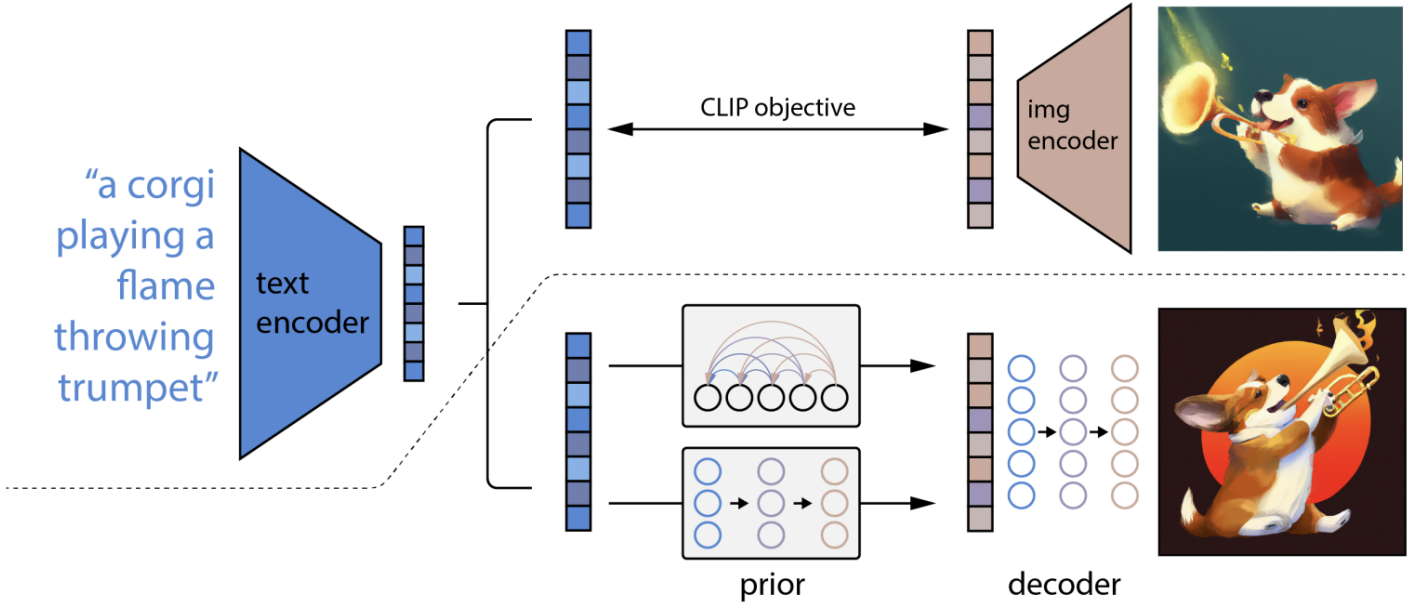

https://arxiv.org/pdf/2204.06125.pdf

# Framework

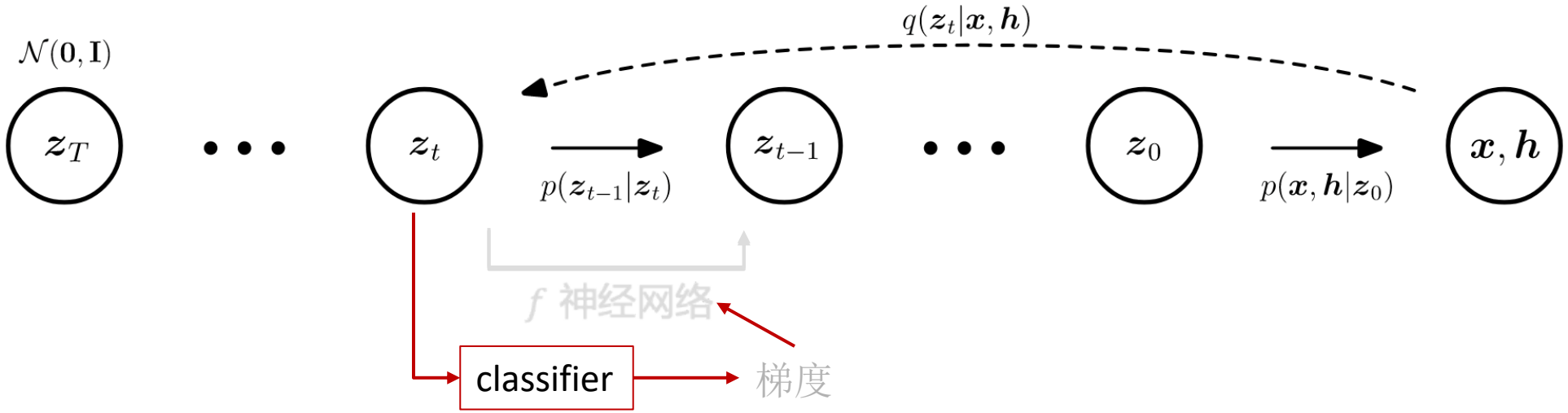



https://arxiv.org/pdf/2204.06125.pdf

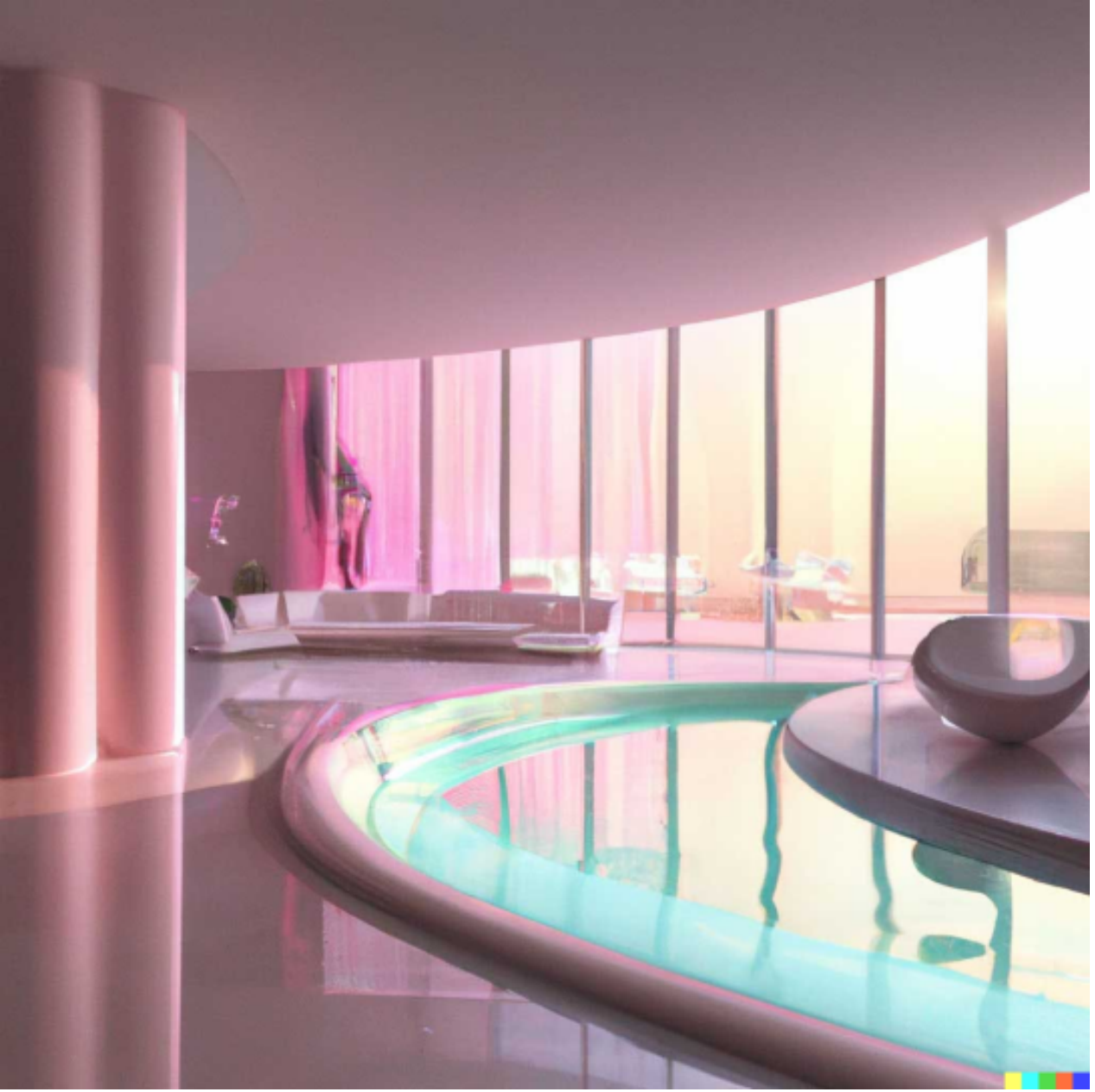
# Framework

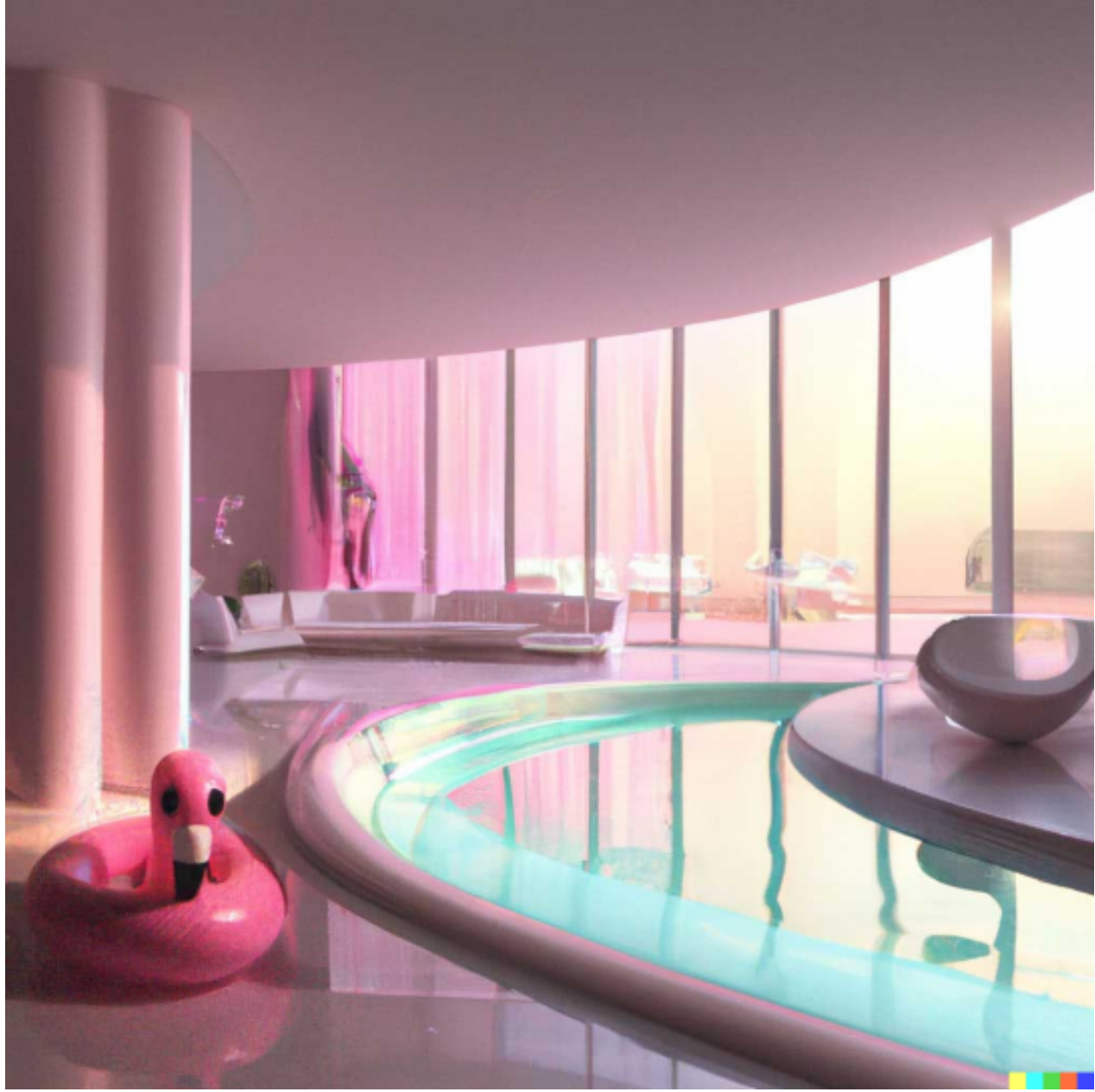



- Classifier guidance:



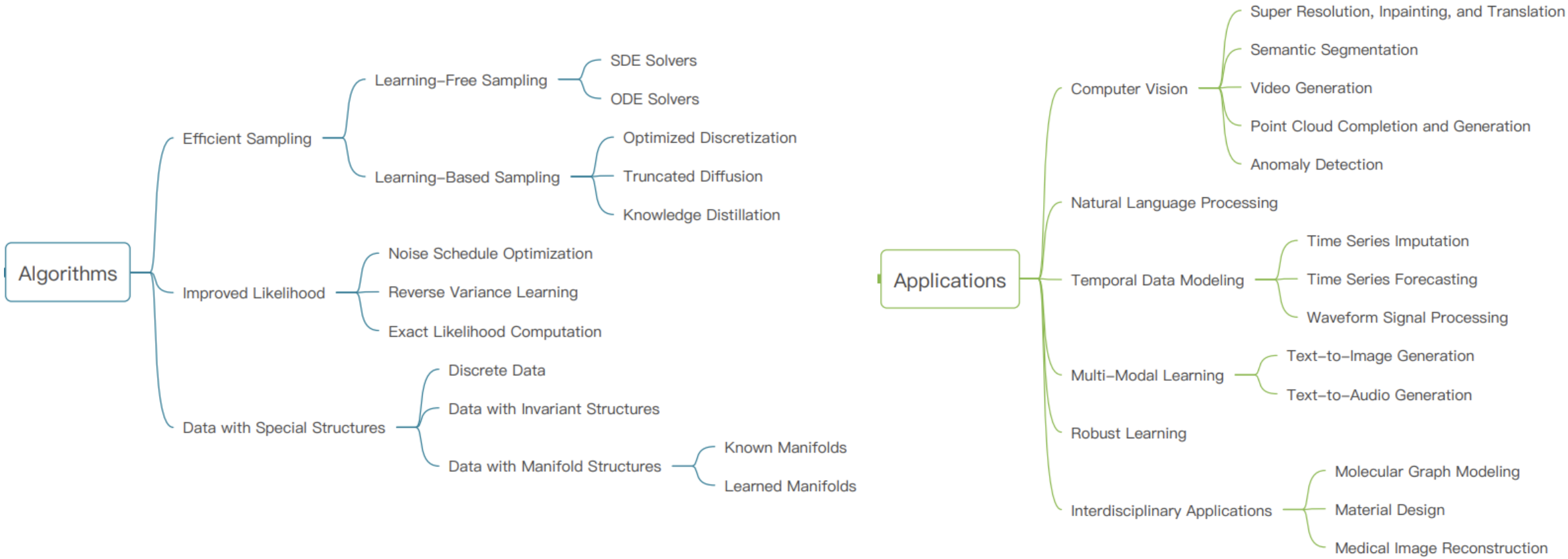

https://arxiv.org/pdf/2204.06125.pdf





Add a flamingo beside the pool.

# More on diffusion models




[1] Cao, H., Tan, C., Gao, Z., Chen, G., Heng, P.-A., & Li, S. Z. (2022). A Survey on Generative Diffusion Model (arXiv:2209.02646). arXiv. http://arxiv.org/abs/2209.02646

[2] Yang, L., Zhang, Z., Song, Y., Hong, S., Xu, R., Zhao, Y., Shao, Y., Zhang, W., Cui, B., & Yang, M.-H. (2022). Diffusion Models: A Comprehensive Survey of Methods and Applications (arXiv:2209.00796). arXiv. http://arxiv.org/abs/2209.00796

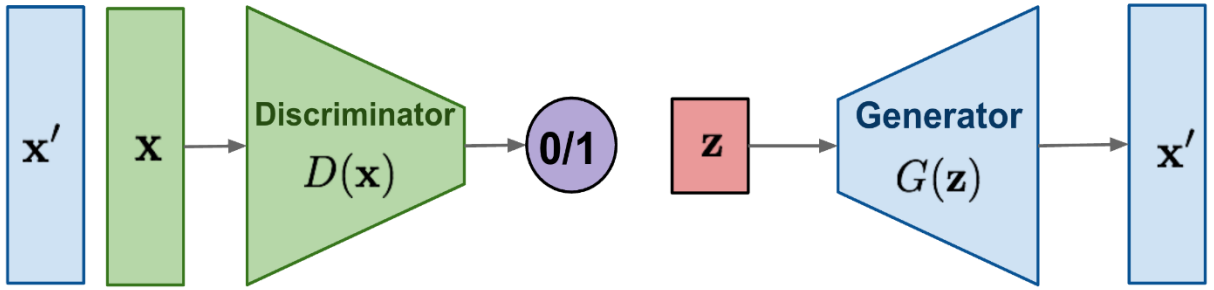
# Normalizing Flow与Diffusion Model的对比

Normalizing flow和Diffusion model都是生成模型，不同之处：

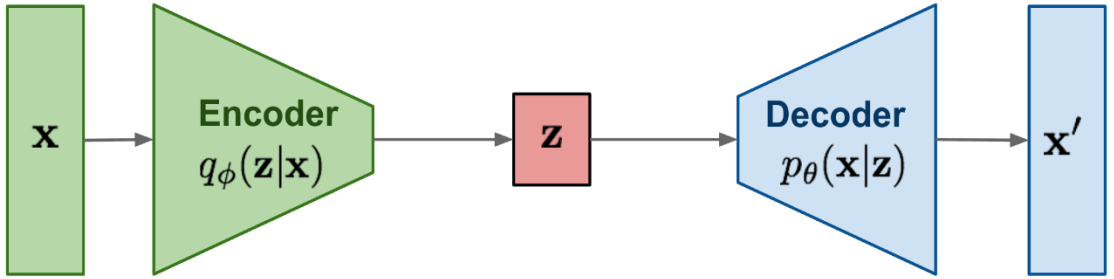
- 变换的顺序：Normalizing flow模型使用一系列的可逆变换来构建生成模型，而Diffusion model则是通过将噪声逐步扩散到数据样本中来生成样本。
- 训练过程：Normalizing flow模型的训练过程可以使用标准的最大似然估计方法进行优化，而Diffusion model则需要使用对数似然差分估计方法进行优化。
- 生成样本：Normalizing flow模型可以直接从潜在空间中生成样本，而Diffusion model则需要使用逆扩散过程来生成样本。
- 生成质量：Normalizing flow模型可以生成高质量的样本，但是在复杂数据集上容易出现模式崩溃的问题；而Diffusion model可以生成良好的样本，且在处理复杂数据集时表现更好。

# Generative Model Zoo

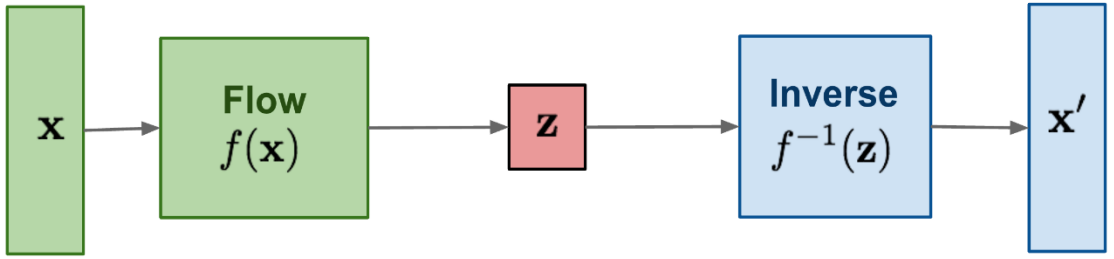
**GAN:** Adversarial training




**VAE:** maximize variational lower bound




**Flow-based models:** Invertible transform of distributions




**Diffusion models:** Gradually add Gaussian noise and then reverse