Diffusion Models

2022
Multicampus

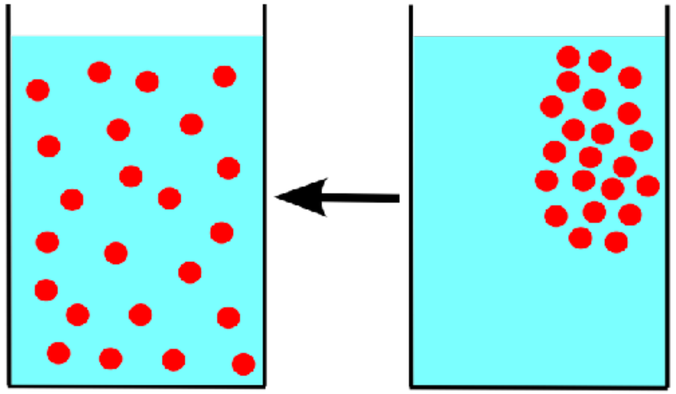
Soochul Park
Vivestudios, Research Scientist

v.1.0

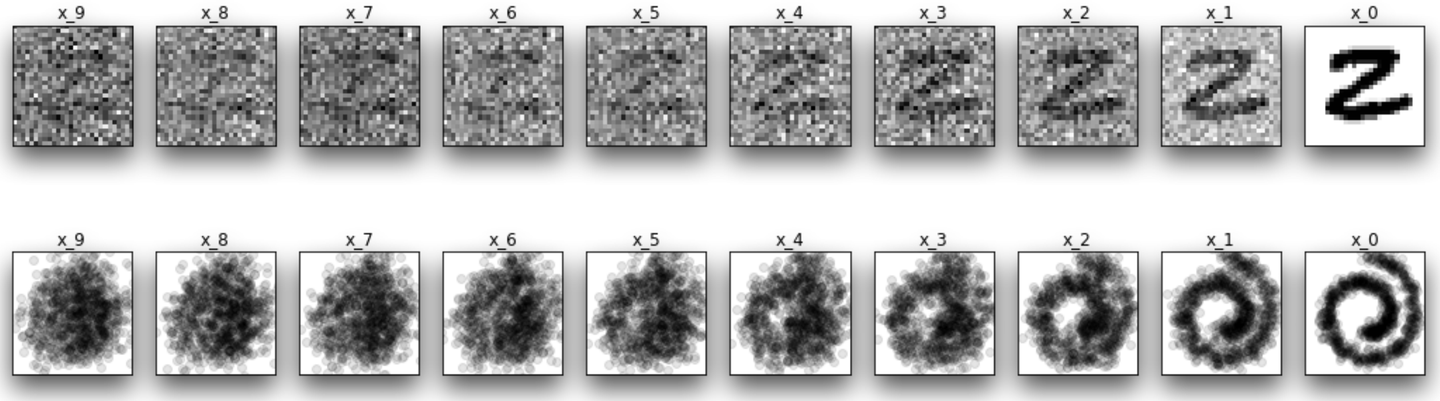
Diffusion Model
DDPM

Diffusion Model

- Sohl-Dickstein, Jascha의 논문 Deep unsupervised learning using nonequilibrium thermodynamics에서 제안



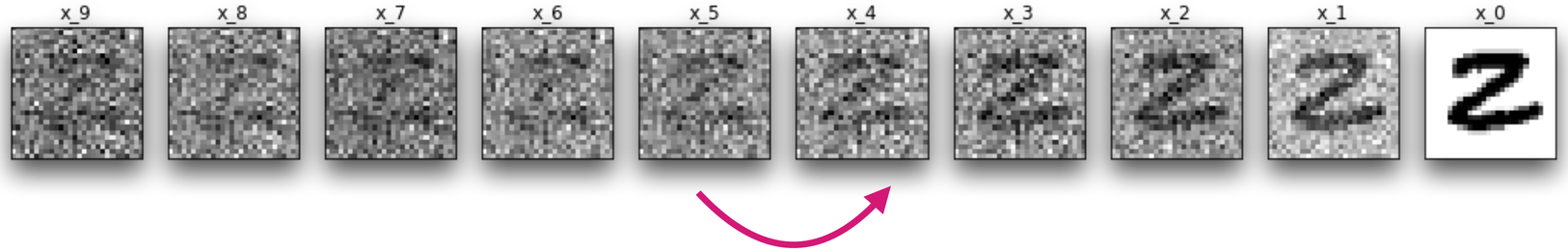
Diffusion
https://en.wikipedia.org/wiki/Diffusion
(Flipped)



Sohl-Dickstein, Jascha, et al. "Deep unsupervised learning using nonequilibrium thermodynamics." *International Conference on Machine Learning*. PMLR, 2015.

Diffusion Model

Forward Process $q\left(\mathbf{x}_{1:T} \mid \mathbf{x}_0\right) := \prod_{t=1}^{T} q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right), q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{x}_0\right) := \mathcal{N}\left(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}\right)$



Posterior $q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right) = \mathcal{N}\left(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t, \mathbf{x}_0\right), \tilde{\beta}_t \mathbf{I}\right)$

where $\tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t, \mathbf{x}_0\right) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}\left(1-\bar{\alpha}_{t-1}\right)}{1-\bar{\alpha}_t}\mathbf{x}_t$ and $\tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$

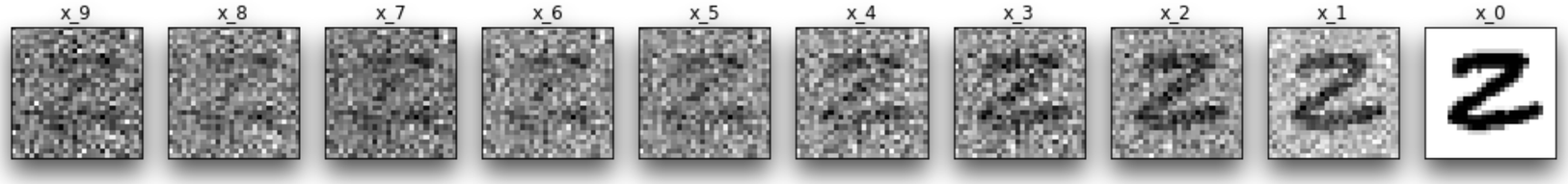
Backward Process (Neural Networks) $p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right) := \mathcal{N}\left(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta\left(\mathbf{x}_t, t\right), \boldsymbol{\Sigma}_\theta\left(\mathbf{x}_t, t\right)\right)$

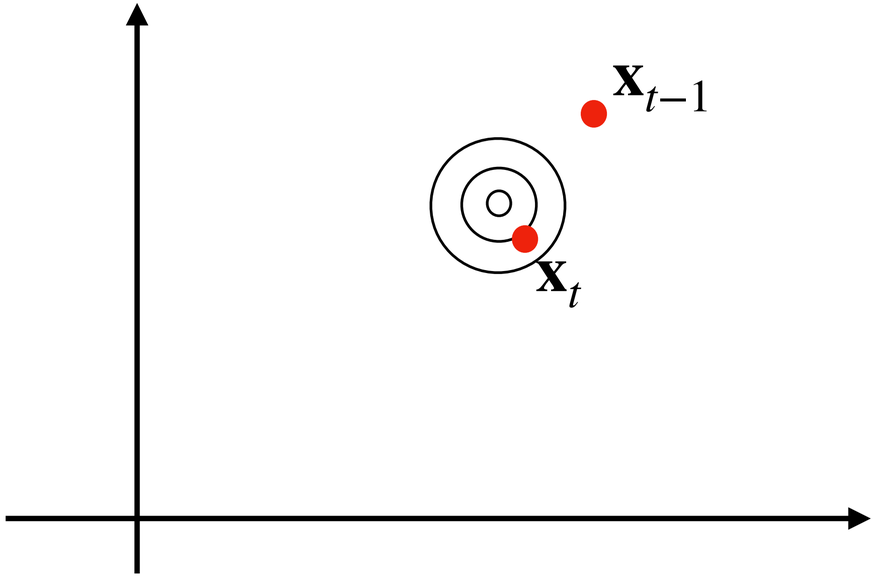
Loss Function $D_{\mathrm{KL}}\left(q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right) \| p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)\right)$

Sohl-Dickstein, Jascha, et al. "Deep unsupervised learning using nonequilibrium thermodynamics." *International Conference on Machine Learning*. PMLR, 2015.

Diffusion Model - Forward Process

Forward Process $q\left(\mathbf{x}_{1:T} \mid \mathbf{x}_0\right) := \prod_{t=1}^{T} q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right), q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{x}_0\right) := \mathcal{N}\left(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}\right)$





Distribution of $\mathbf{x}_t$ at an arbitrary timestep t in closed form

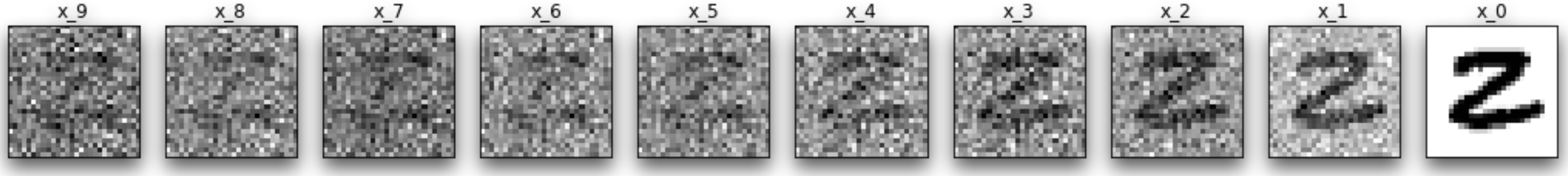
$q\left(\mathbf{x}_t \mid \mathbf{x}_0\right) = \mathcal{N}\left(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, \left(1-\bar{\alpha}_t\right)\mathbf{I}\right)$, where $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$

식 유도는Lil'Log 참고
https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

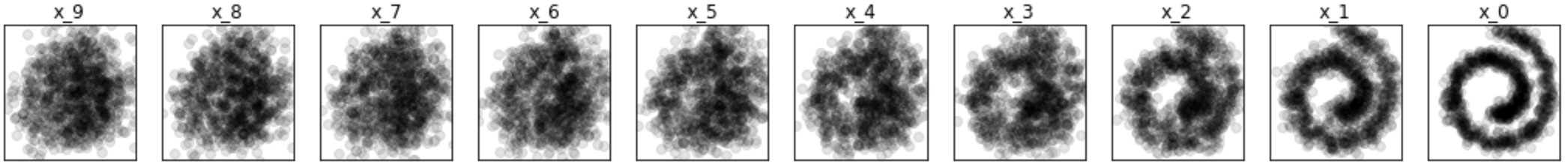
Sohl-Dickstein, Jascha, et al. "Deep unsupervised learning using nonequilibrium thermodynamics." *International Conference on Machine Learning*. PMLR, 2015.

Diffusion Model - Forward Process

MNIST single data, $\beta = 0.2$, $T = 10$



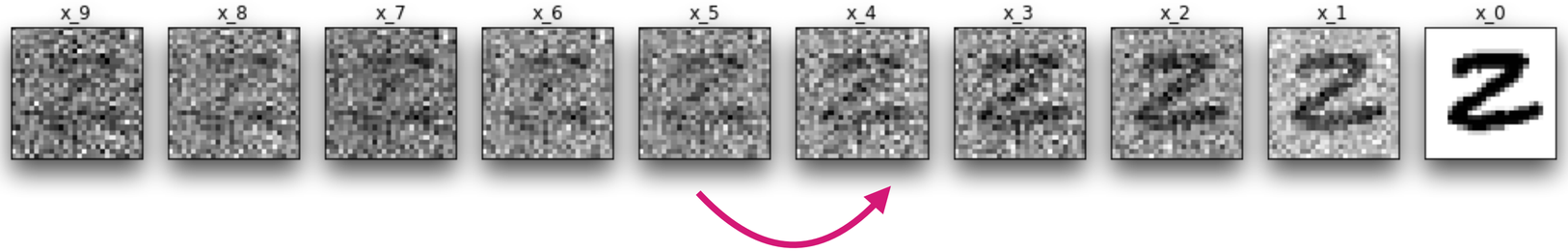
Swiss roll dataset, $\beta = 0.05$, $T = 10$



Sohl-Dickstein, Jascha, et al. "Deep unsupervised learning using nonequilibrium thermodynamics." *International Conference on Machine Learning*. PMLR, 2015.

Diffusion Model - Posterior

Forward Process $q\left(\mathbf{x}_{1:T} \mid \mathbf{x}_0\right) := \prod_{t=1}^{T} q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right), q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{x}_0\right) := \mathcal{N}\left(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}\right)$



Posterior $q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right) = \mathcal{N}\left(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t, \mathbf{x}_0\right), \tilde{\beta}_t \mathbf{I}\right)$

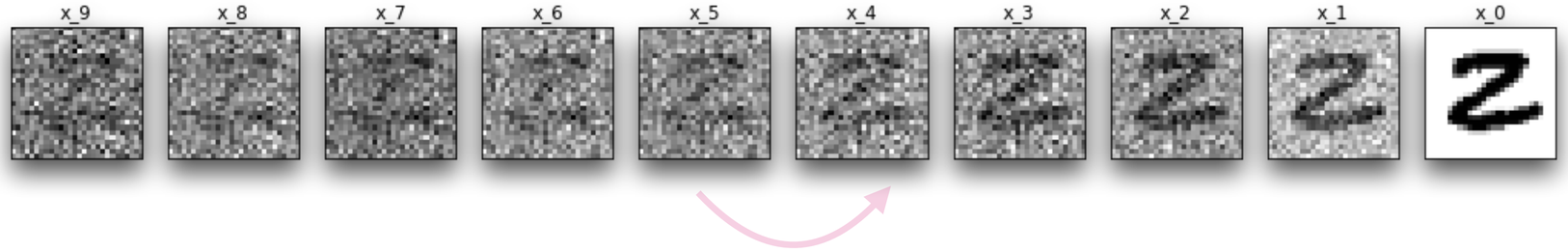
where $\tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t, \mathbf{x}_0\right) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}\left(1-\bar{\alpha}_{t-1}\right)}{1-\bar{\alpha}_t}\mathbf{x}_t$ and $\tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$

$$q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) = \frac{q(\mathbf{x}_{t-1} \mid \mathbf{x}_0) q(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{x}_0)}{q(\mathbf{x}_t \mid \mathbf{x}_0)}$$ by Bayes' Rule

Sohl-Dickstein, Jascha, et al. "Deep unsupervised learning using nonequilibrium thermodynamics." *International Conference on Machine Learning*. PMLR, 2015.

Diffusion Model - Backward Process

Forward Process $q\left(\mathbf{x}_{1:T} \mid \mathbf{x}_0\right) := \prod_{t=1}^{T} q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right), q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{x}_0\right) := \mathcal{N}\left(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}\right)$

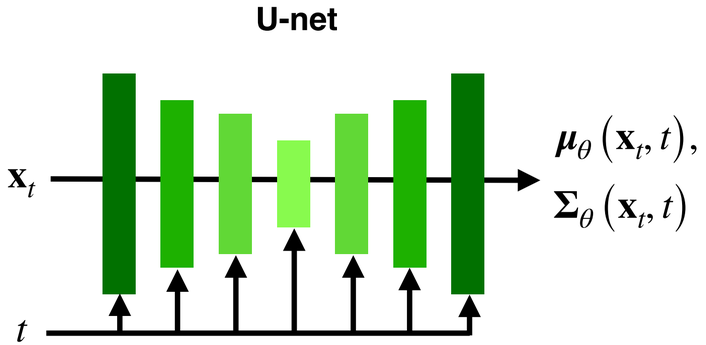


Posterior $q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right) = \mathcal{N}\left(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t, \mathbf{x}_0\right), \tilde{\beta}_t \mathbf{I}\right)$

where $\tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t, \mathbf{x}_0\right) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}\left(1-\bar{\alpha}_{t-1}\right)}{1-\bar{\alpha}_t}\mathbf{x}_t$ and $\tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$

Backward Process (Neural Networks) $p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right) := \mathcal{N}\left(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta\left(\mathbf{x}_t, t\right), \boldsymbol{\Sigma}_\theta\left(\mathbf{x}_t, t\right)\right)$

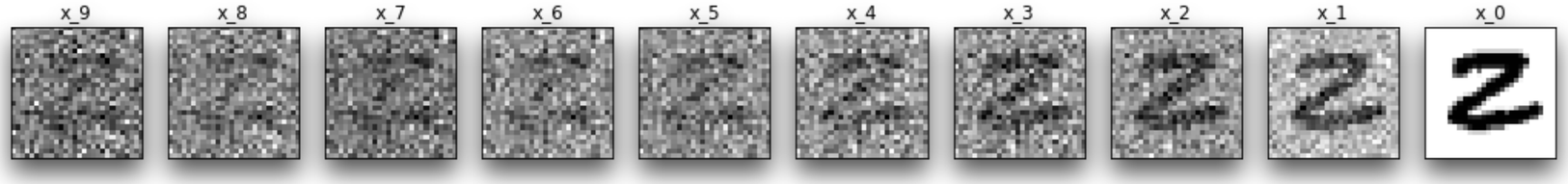
Loss Function $D_{\mathrm{KL}}\left(q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right) \| p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)\right)$



Sohl-Dickstein, Jascha, et al. "Deep unsupervised learning using nonequilibrium thermodynamics." *International Conference on Machine Learning*. PMLR, 2015.

Diffusion Model - Loss Function

Forward Process $q\left(\mathbf{x}_{1:T} \mid \mathbf{x}_0\right) := \prod_{t=1}^{T} q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right), q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{x}_0\right) := \mathcal{N}\left(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}\right)$

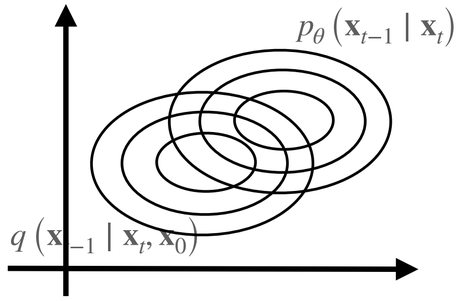


Posterior $q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right) = \mathcal{N}\left(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t, \mathbf{x}_0\right), \tilde{\beta}_t \mathbf{I}\right)$

where $\tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t, \mathbf{x}_0\right) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}\left(1-\bar{\alpha}_{t-1}\right)}{1-\bar{\alpha}_t}\mathbf{x}_t$ and $\tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$

Backward Process (Neural Networks) $p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right) := \mathcal{N}\left(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta\left(\mathbf{x}_t, t\right), \boldsymbol{\Sigma}_\theta\left(\mathbf{x}_t, t\right)\right)$

Loss Function $D_{\mathrm{KL}}\left(q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right) \| p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)\right)$



Sohl-Dickstein, Jascha, et al. "Deep unsupervised learning using nonequilibrium thermodynamics." *International Conference on Machine Learning*. PMLR, 2015.

Diffusion Model - Output Samples

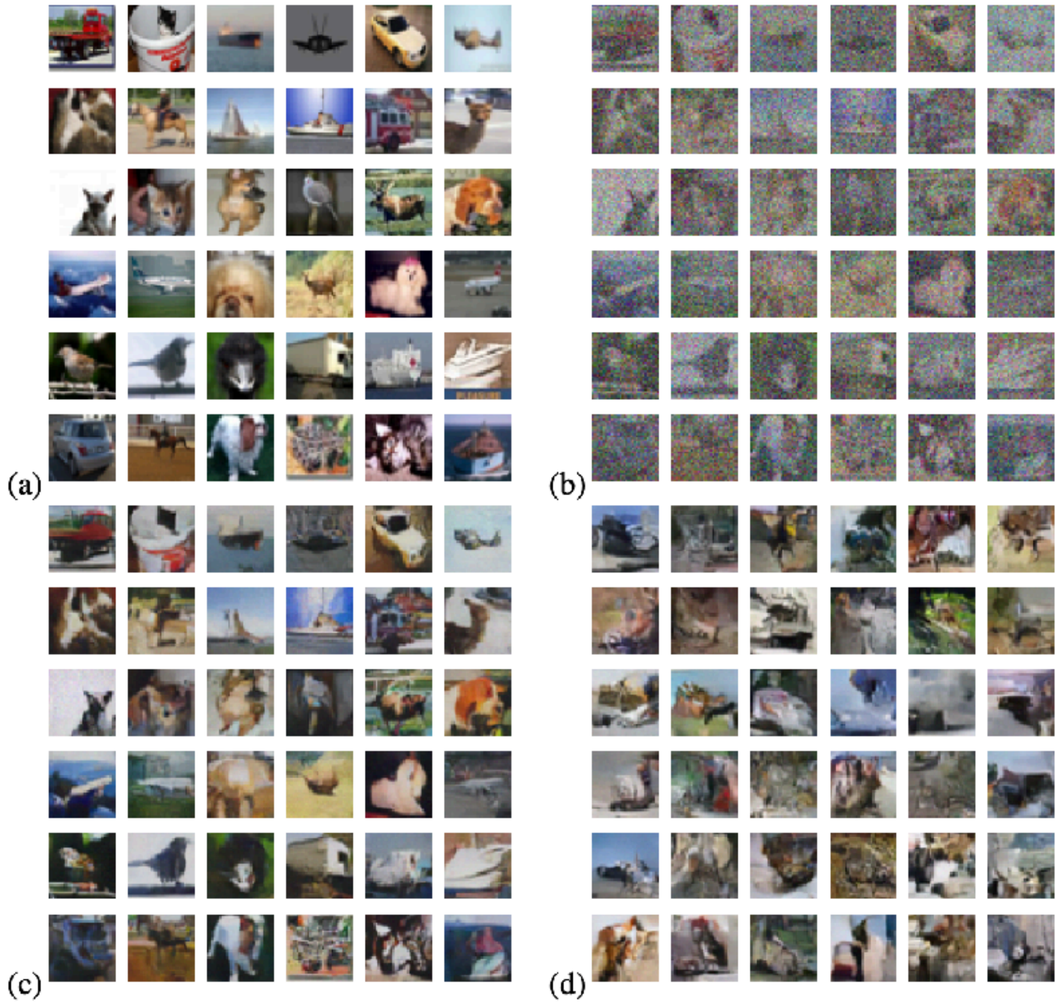


Figure 3. The proposed framework trained on the CIFAR-10 (Krizhevsky & Hinton, 2009) dataset. *(a)* Example holdout data (similar to training data). *(b)* Holdout data corrupted with Gaussian noise of variance 1 (SNR = 1). *(c)* Denoised images, generated by sampling from the posterior distribution over denoised images conditioned on the images in (b). *(d)* Samples generated by the diffusion model.

Sohl-Dickstein, Jascha, et al. "Deep unsupervised learning using nonequilibrium thermodynamics." *International Conference on Machine Learning*. PMLR, 2015.

DDPM (Denoising Diffusion Probabilistic Models)

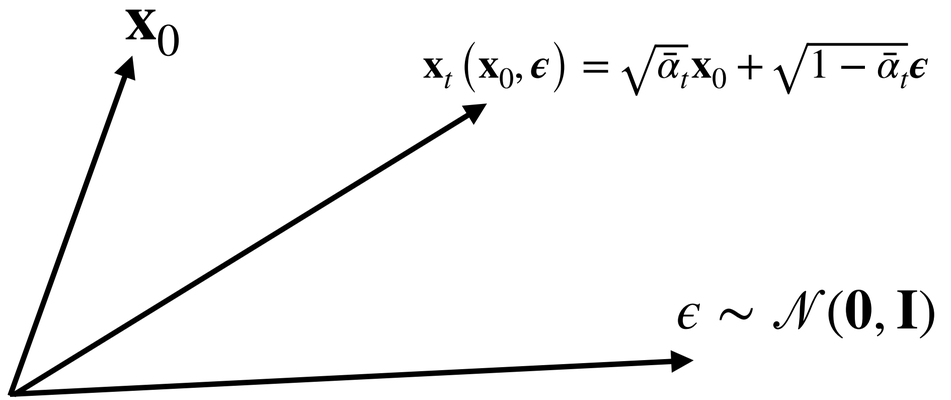
- Jonathan Ho의 논문 Denoising diffusion probabilistic models에서 제안

Posterior $q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right) = \mathcal{N}\left(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t, \mathbf{x}_0\right), \tilde{\beta}_t \mathbf{I}\right)$

Distribution of $\mathbf{x}_t$ at an arbitrary timestep t in closed form

$q\left(\mathbf{x}_t \mid \mathbf{x}_0\right) = \mathcal{N}\left(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, \left(1-\bar{\alpha}_t\right)\mathbf{I}\right)$, where $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$

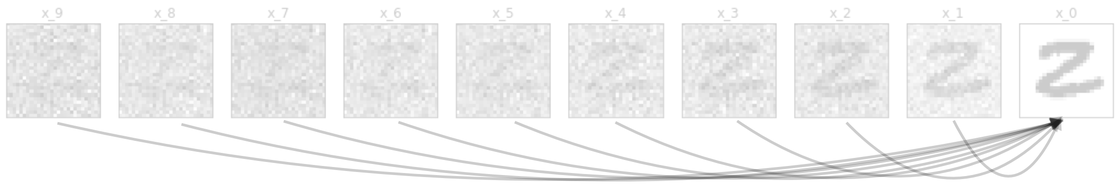
Loss Function $L_{\text{simple}}(\theta) := \mathbb{E}_{t,\mathbf{x}_0,\epsilon}\left[\left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t\right)\right\|^2\right]$



$\mathbf{x}_t$ is a linear combination of $\mathbf{x}_0$ and ϵ

```
106     def p_loss(self, model, x_0, t, noise=None):
107         if noise is None:
108             noise = torch.randn_like(x_0)          generate ϵ
109
110         x_noise = self.q_sample(x_0, t, noise)     sample x_t
111         x_recon = model(x_noise, t)                predict ϵ
112
113         return F.mse_loss(x_recon, noise)
```

https://github.com/rosinality/denoising-diffusion-pytorch/blob/master/diffusion.py



Predict ϵ (or $\mathbf{x}_0$) at each step

Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." *Advances in Neural Information Processing Systems* 33 (2020): 6840-6851.

DDPM (Denoising Diffusion Probabilistic Models)

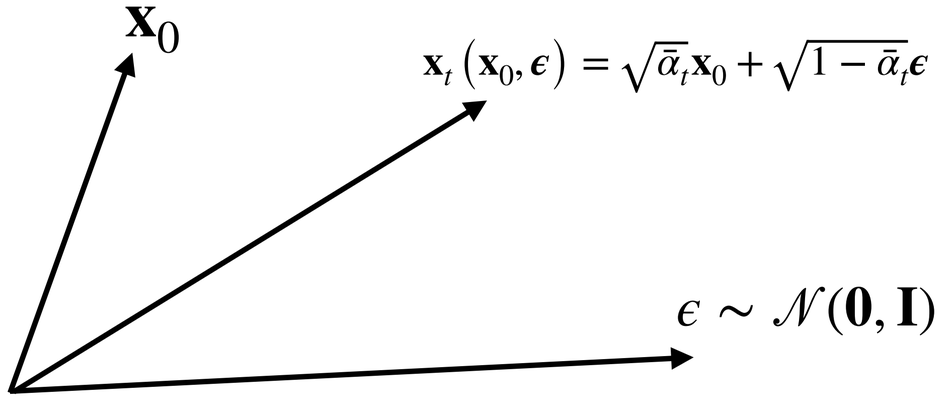
- Jonathan Ho의 논문 Denoising diffusion probabilistic models에서 제안

Posterior $q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right) = \mathcal{N}\left(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t, \mathbf{x}_0\right), \tilde{\beta}_t \mathbf{I}\right)$

Distribution of $\mathbf{x}_t$ at an arbitrary timestep t in closed form

$q\left(\mathbf{x}_t \mid \mathbf{x}_0\right) = \mathcal{N}\left(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, \left(1-\bar{\alpha}_t\right)\mathbf{I}\right)$, where $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$

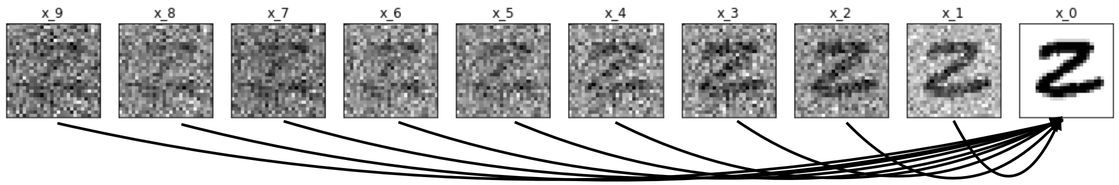
Loss Function $L_{\text{simple}}(\theta) := \mathbb{E}_{t,\mathbf{x}_0,\epsilon}\left[\left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t\right)\right\|^2\right]$



$\mathbf{x}_t$ is a linear combination of $\mathbf{x}_0$ and ϵ

```
106     def p_loss(self, model, x_0, t, noise=None):
107         if noise is None:
108             noise = torch.randn_like(x_0)          generate ϵ
109
110         x_noise = self.q_sample(x_0, t, noise)     sample x_t
111         x_recon = model(x_noise, t)                predict ϵ
112
113         return F.mse_loss(x_recon, noise)
```

https://github.com/rosinality/denoising-diffusion-pytorch/blob/master/diffusion.py



Predict ϵ (or $\mathbf{x}_0$) at each step

Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." *Advances in Neural Information Processing Systems* 33 (2020): 6840-6851.

DDPM - Output Samples

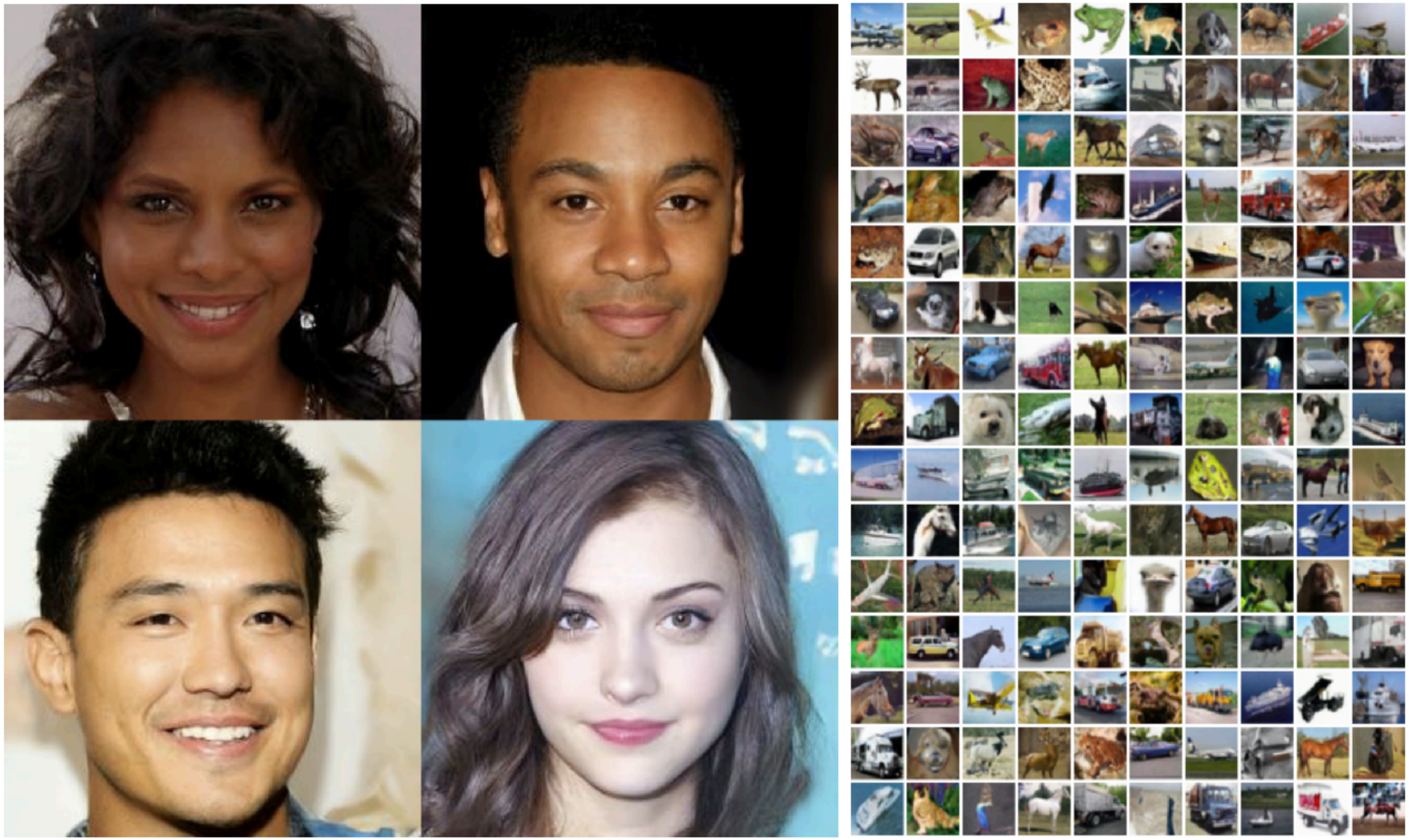


Figure 1: Generated samples on CelebA-HQ 256 × 256 (left) and unconditional CIFAR10 (right)

Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." *Advances in Neural Information Processing Systems* 33 (2020): 6840-6851.

DDPM - Output Samples

https://github.com/yang-song/score_sde_pytorch/

Song, Yang, et al. "Score-based generative modeling through stochastic differential equations." *arXiv preprint arXiv:2011.13456* (2020).

Diffusion vs. ...

"We present a novel way to define probabilistic models that allows:

1. **extreme flexibility in model structure**,
2. exact sampling,
3. easy multiplication with other distributions, e.g. in order to compute a posterior, and
4. **the model log likelihood, and the probability of individual states, to be cheaply evaluated.**"

Sohl-Dickstein, Jascha, et al. "Deep unsupervised learning using nonequilibrium thermodynamics." *International Conference on Machine Learning*. PMLR, 2015.

"We emphasize that our objective Eq. (6) requires **no adversarial training, no surrogate losses, and nosampling from the score network during training** (e.g., unlike contrastive divergence). Also, it **does not require sθ(x, σ) to have special architectures** in order to be tractable."

Song, Yang, and Stefano Ermon. "Generative modeling by estimating gradients of the data distribution." *Advances in Neural Information Processing Systems* 32 (2019).

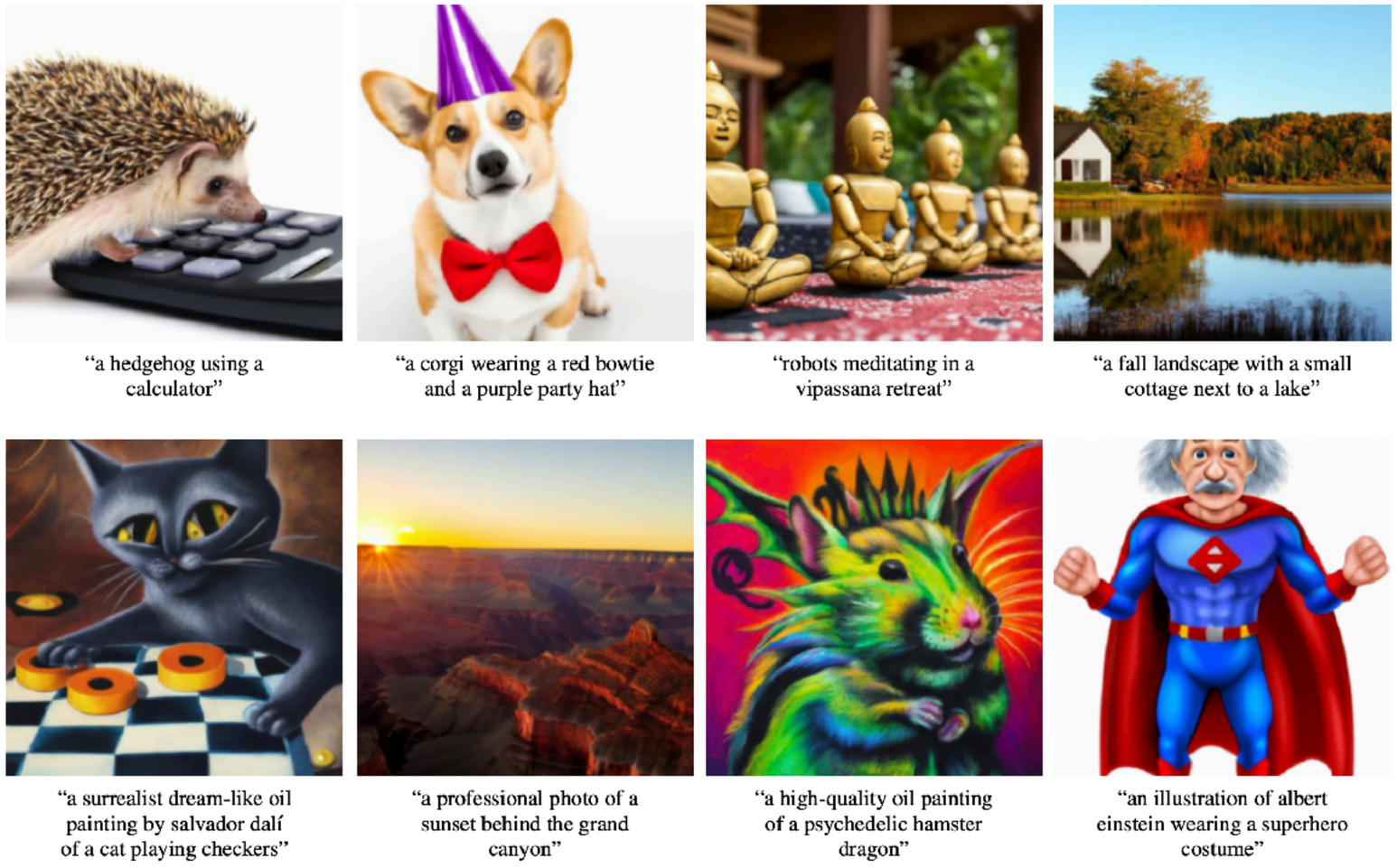
Diffusion vs. ...

	Auto-Regressive	VAE	Flow	GAN	Diffusion
Tractability	**Good**	**Good**	**Good**	Not Good Likelihood can't be evaluated	**Good**
Flexibility	Not Good Causal structure Fixed distribution	Not Good Dimension reduction Fixed distribution	Not Good Invertible structure Fixed distribution	**Good**	**Good**

GLIDE
CLIP
DALL·E 2

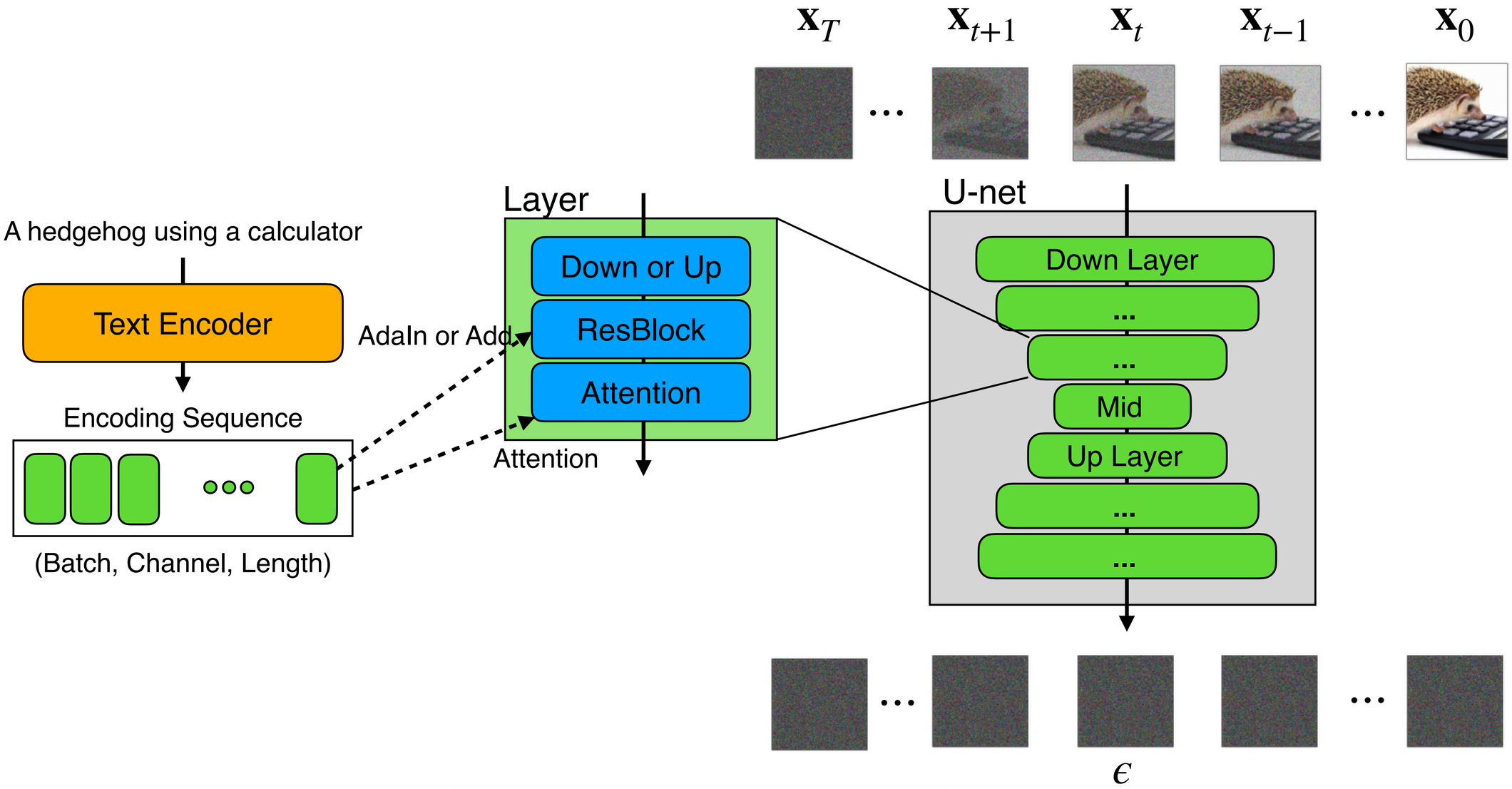
GLIDE - Output Samples

- OpenAI에서 논문 Glide: Towards photorealistic image generation and editing with text-guided diffusion models을 통해 제안



Nichol, Alex, et al. "Glide: Towards photorealistic image generation and editing with text-guided diffusion models." *arXiv preprint arXiv:2112.10741* (2021).

GLIDE - Overall Architecture

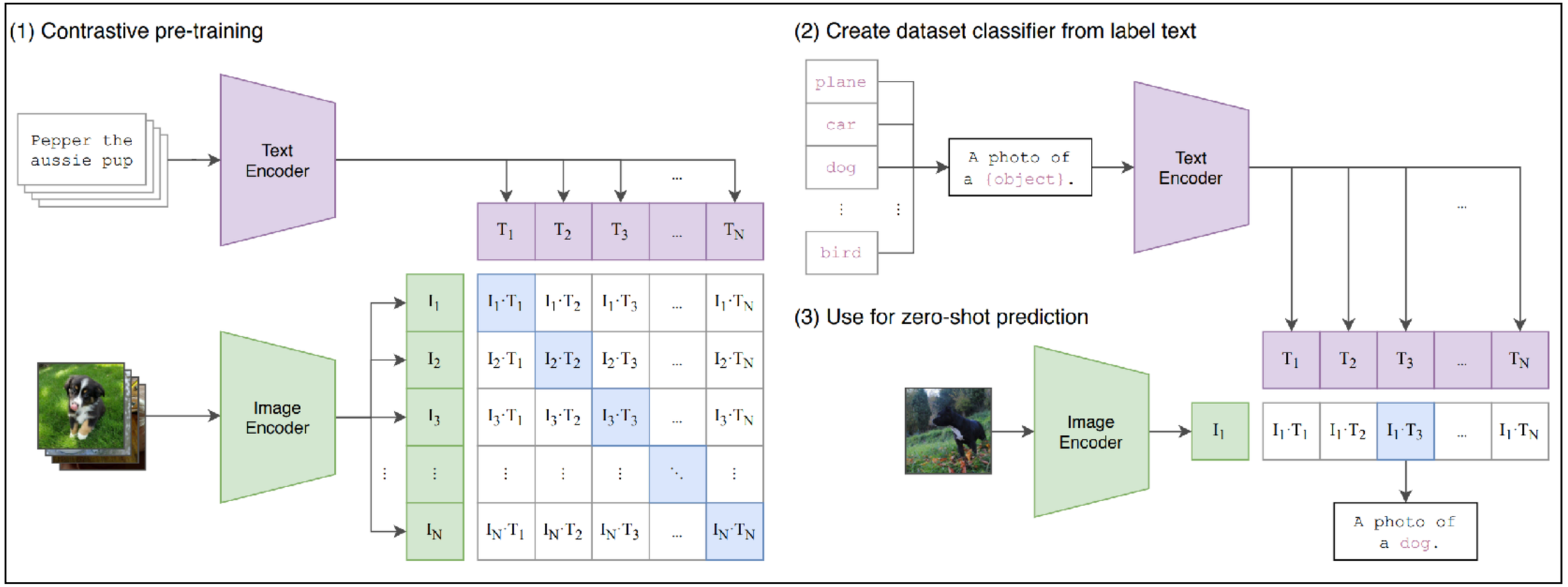


GLIDE source : https://github.com/openai/glide-text2im

Nichol, Alex, et al. "Glide: Towards photorealistic image generation and editing with text-guided diffusion models." *arXiv preprint arXiv:2112.10741* (2021).

CLIP (Contrastive Language-Image Pre-training)

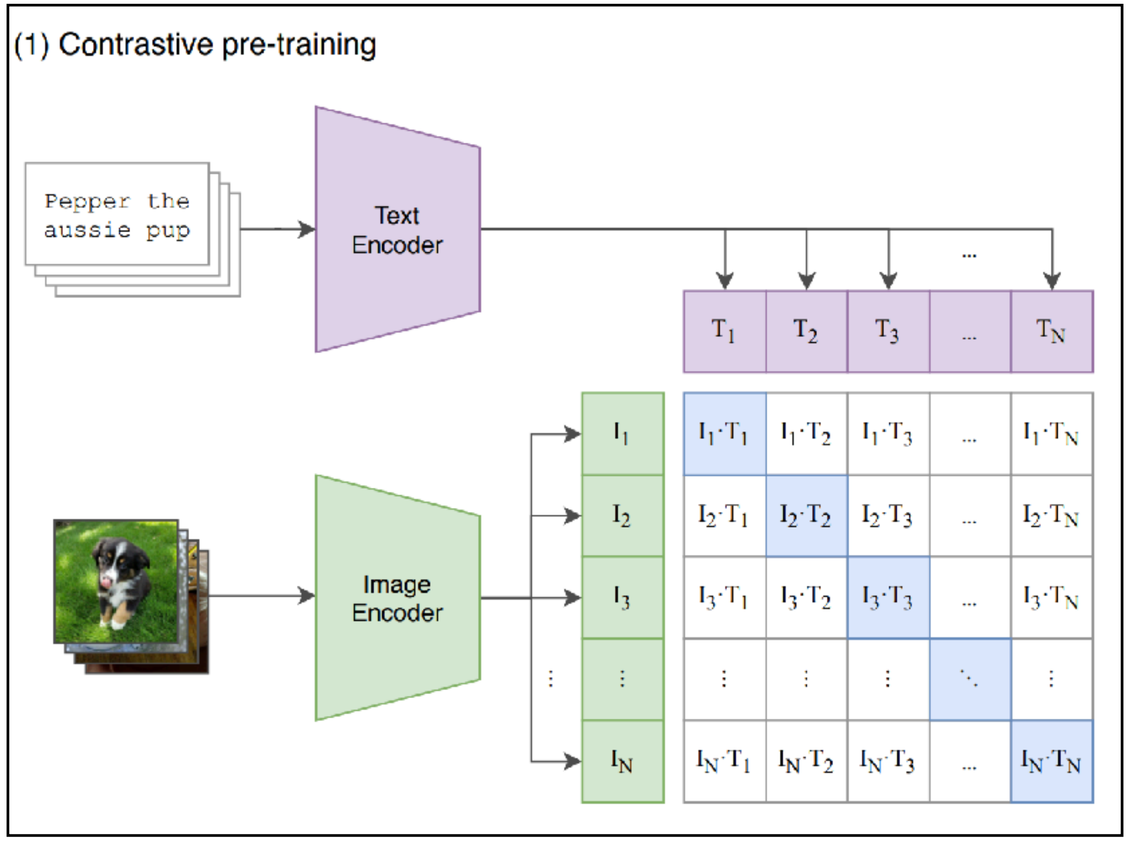
- (image, text) 쌍의 데이터로 self-supervised learning을 통해 text/image encoder를 학습



Radford, Alec, et al. "Learning transferable visual models from natural language supervision." *International Conference on Machine Learning*. PMLR, 2021.

CLIP (Contrastive Language-Image Pre-training)

- (image, text) 쌍의 데이터로 self-supervised learning을 통해 text/image encoder를 학습



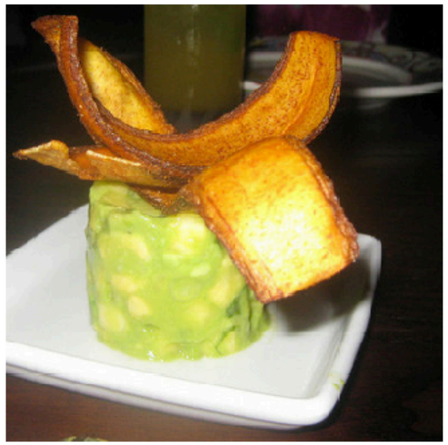

```
# image_encoder - ResNet or Vision Transformer
# text_encoder  - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l]       - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t             - learned temperature parameter

# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T)  #[n, d_t]

# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)

# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss   = (loss_i + loss_t)/2
```

Radford, Alec, et al. "Learning transferable visual models from natural language supervision." *International Conference on Machine Learning*. PMLR, 2021.

CLIP (Contrastive Language-Image Pre-training)

FOOD101

guacamole (90.1%) Ranked 1 out of 101 labels



✓ a photo of **guacamole**, a type of food.

× a photo of **ceviche**, a type of food.

× a photo of **edamame**, a type of food.

× a photo of **tuna tartare**, a type of food.

× a photo of **hummus**, a type of food.

SUN397

television studio (90.2%) Ranked 1 out of 397

✓ a photo of a **television studio**.

× a photo of a **podium indoor**.

× a photo of a **conference room**.

× a photo of a **lecture room**.

× a photo of a **control room**.

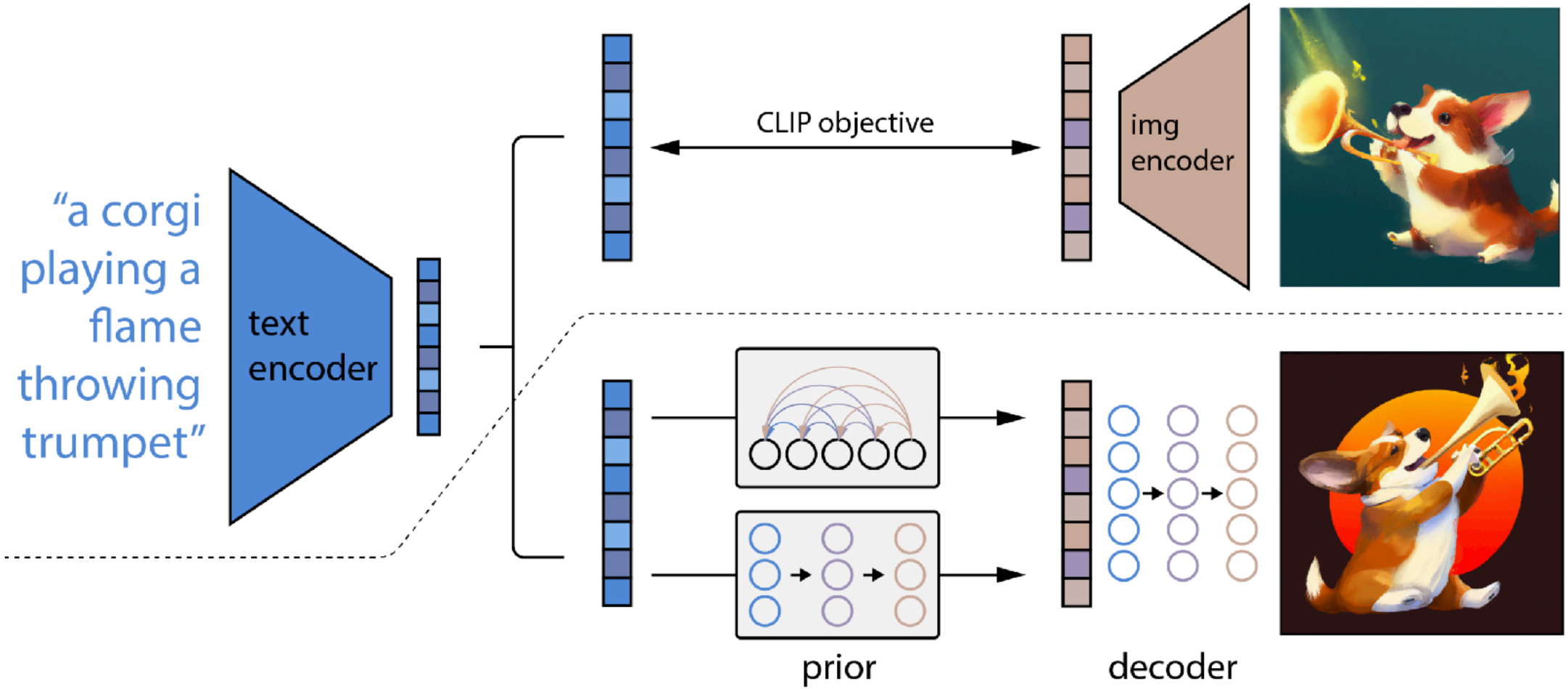
https://openai.com/blog/clip/

DALL·E 2 - Output Samples

- OpenAI에서 Hierarchical text-conditional image generation with clip latents 논문을 통해 제안

Ramesh, Aditya, et al. "Hierarchical text-conditional image generation with clip latents." *arXiv preprint arXiv:2204.06125* (2022).

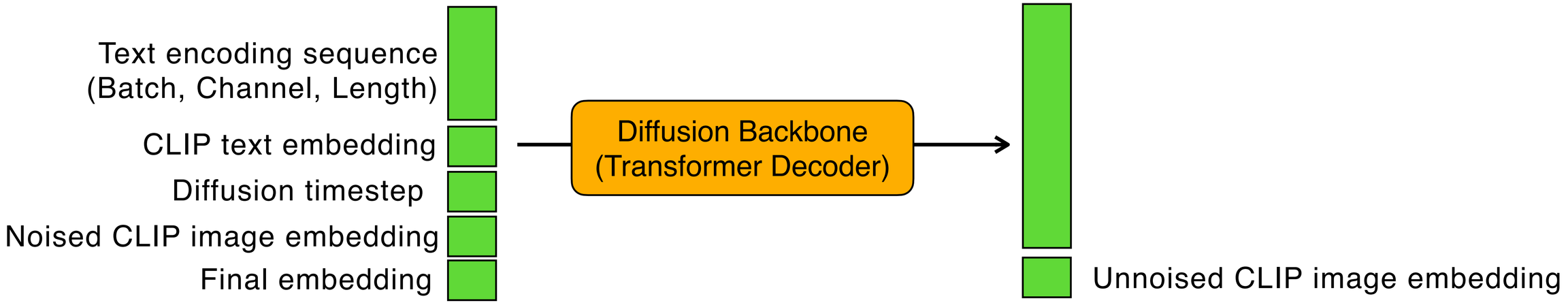
DALL·E 2 - Overall Architecture



Ramesh, Aditya, et al. "Hierarchical text-conditional image generation with clip latents." *arXiv preprint arXiv:2204.06125* (2022).

DALL·E 2 - Prior

- GLIDE text embedding과 CLIP text embedding으로부터 CLIP image embedding을 생성



Ramesh, Aditya, et al. "Hierarchical text-conditional image generation with clip latents." *arXiv preprint arXiv:2204.06125* (2022).

Guided Diffusion Sampling
Classifier-free Diffusion Guidance
CLIP Guidance

Guided Diffusion Sampling

- Diffusion Models Beat GANs on Image Synthesis 논문에서 제안

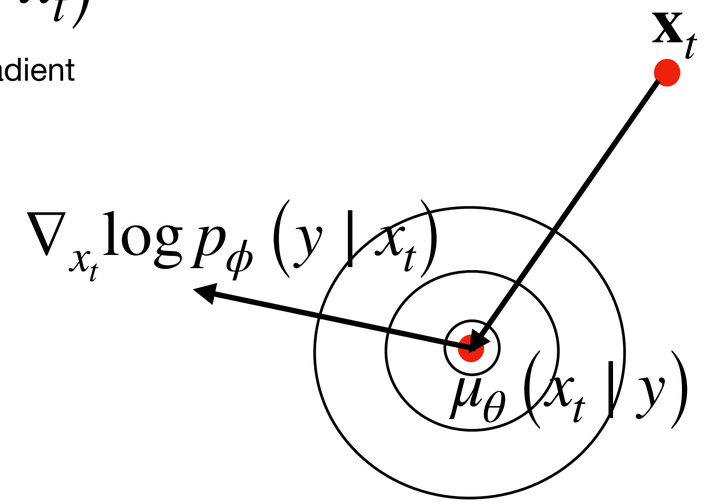
- Diffusion 모델 외 추가적인 image classifier를 학습시키고 sampling 과정에서 classifier로부터 gradient를 받아 sampling에 도움을 줌

Posterior $q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right) = \mathcal{N}\left(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t, \mathbf{x}_0\right), \tilde{\beta}_t \mathbf{I}\right)$

Nichol, Alex, et al.에서 재인용

$$\hat{\mu}_\theta\left(x_t \mid y\right) = \mu_\theta\left(x_t \mid y\right) + s \cdot \Sigma_\theta\left(x_t \mid y\right) \nabla_{x_t} \log p_\phi\left(y \mid x_t\right)$$

- $\hat{\mu}_\theta\left(x_t \mid y\right)$: 수정된 mean
- $\mu_\theta\left(x_t \mid y\right)$: 기존 mean
- s: guidance scale
- $\Sigma_\theta\left(x_t \mid y\right)$: 기존 covariance
- $\nabla_{x_t} \log p_\phi\left(y \mid x_t\right)$: classifier로부터 전해진 gradient



Dhariwal, Prafulla, and Alexander Nichol. "Diffusion models beat gans on image synthesis." *Advances in Neural Information Processing Systems* 34 (2021): 8780-8794.

Nichol, Alex, et al. "Glide: Towards photorealistic image generation and editing with text-guided diffusion models." arXiv preprint arXiv:2112.10741 (2021).

Guided Diffusion Sampling

Figure 13: Samples from our best 512×512 model (FID: 3.85). Classes are 1: goldfish, 279: arctic fox, 323: monarch butterfly, 386: african elephant, 130: flamingo, 852: tennis ball.

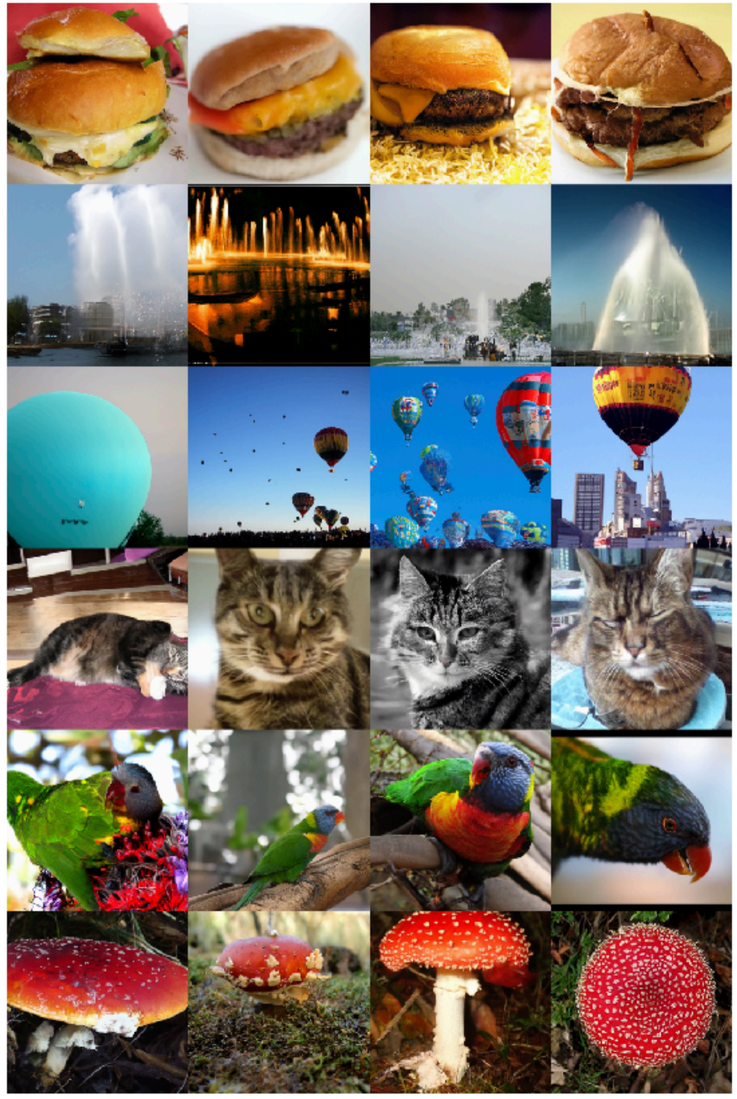


Figure 14: Samples from our best 512×512 model (FID: 3.85). Classes are 933: cheeseburger, 562: fountain, 417: balloon, 281: tabby cat, 90: lorikeet, 992: agaric.

Figure 15: Difficult class samples from our best 512×512 model (FID: 3.85). Classes are 432: bassoon, 468: cab, 424: barbershop, 444: bicycle-built-for-two, 981: ballplayer, 550: espresso maker.

Dhariwal, Prafulla, and Alexander Nichol. "Diffusion models beat gans on image synthesis." *Advances in Neural Information Processing Systems* 34 (2021): 8780-8794.

Classifier-free guidance

- Ho, Jonathan과 Tim Salimans의 논문 Classifier-free diffusion guidance에서 제안

- 추가적인 classifier를 트레이닝할 필요없이 diffusion 모델만 가지고 guided sampling을 가능하게 만듦

Nichol, Alex, et al.에서 재인용

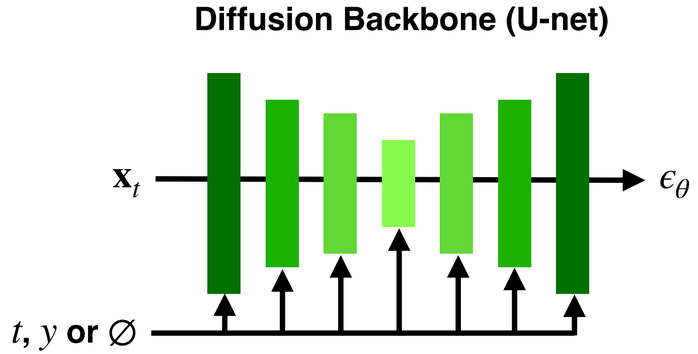
$$\hat{\epsilon}_\theta\left(x_t \mid y\right) = \epsilon_\theta\left(x_t \mid y\right) + s \cdot \left(\epsilon_\theta\left(x_t \mid y\right) - \epsilon_\theta\left(x_t \mid \varnothing\right)\right)$$

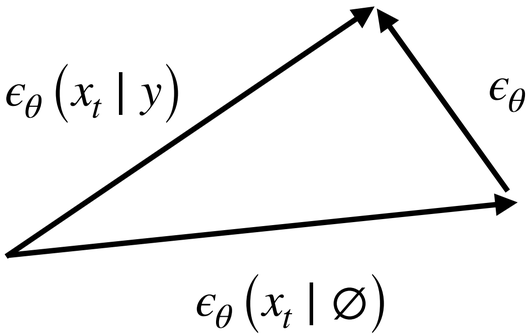
수정된 score

conditional predicted score

guidance scale

unconditional predicted score



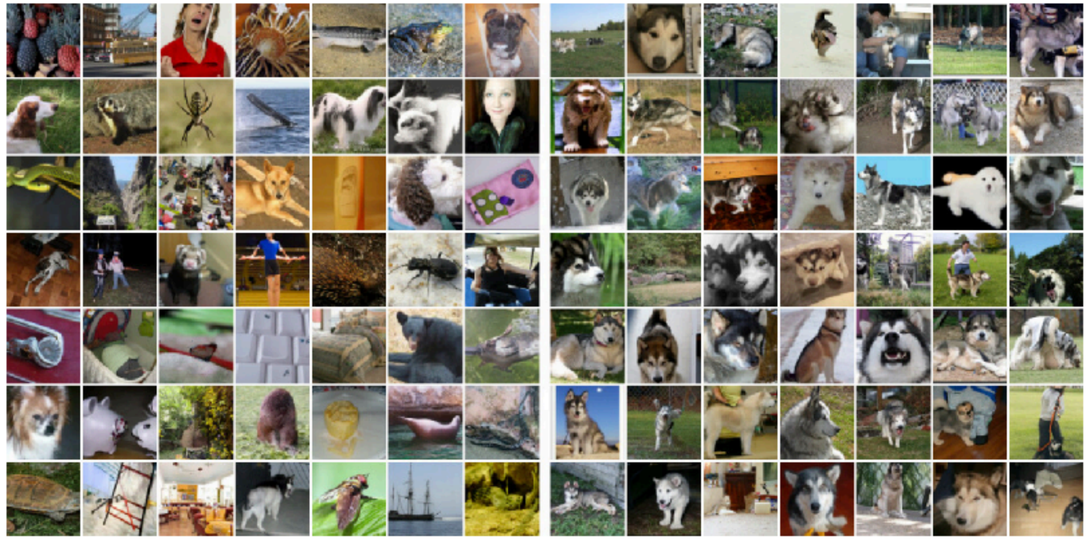


$$\epsilon_\theta\left(x_t \mid y\right) - \epsilon_\theta\left(x_t \mid \varnothing\right) \approx -\sigma_t \nabla_{x_t} \log p^i\left(y \mid x_t\right)$$

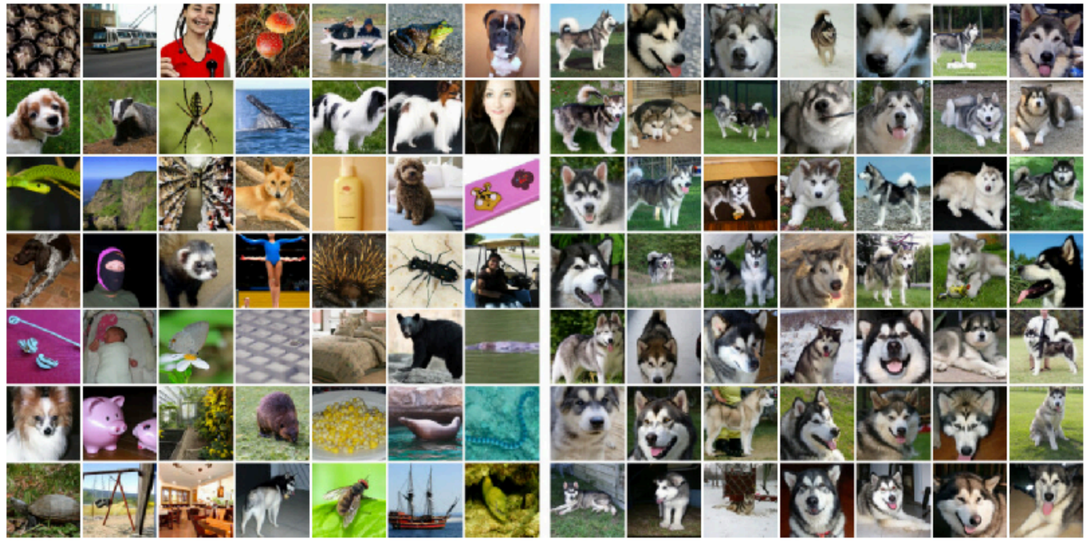
Ho, Jonathan, and Tim Salimans. "Classifier-free diffusion guidance." NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications. 2021.

Nichol, Alex, et al. "Glide: Towards photorealistic image generation and editing with text-guided diffusion models." arXiv preprint arXiv:2112.10741 (2021).

Classifier-free guidance



(a) Non-guided conditional sampling: FID=1.80, IS=53.71



(b) Classifier-free guidance with $w = 1.0$: FID=12.6, IS=170.1

Ho, Jonathan, and Tim Salimans. "Classifier-free diffusion guidance." NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications. 2021.

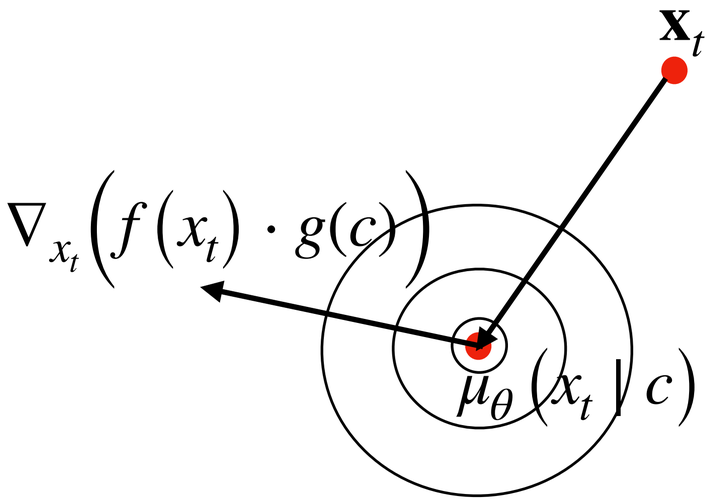
CLIP Guidance

- Classifier를 이용한 guidance 방법과 비슷하게 CLIP 모델을 이용하여 sampling 단계에서 도움을 줌

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

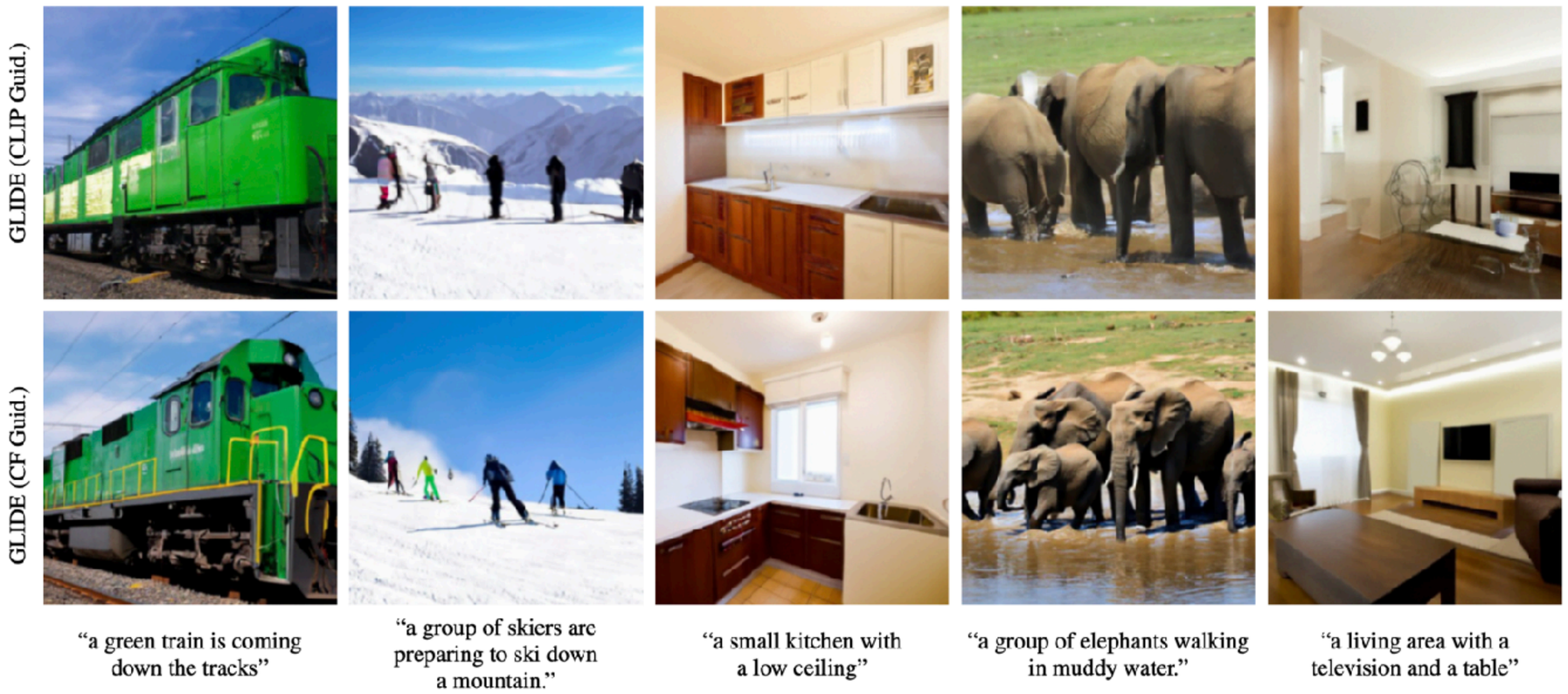
CLIP image encoding

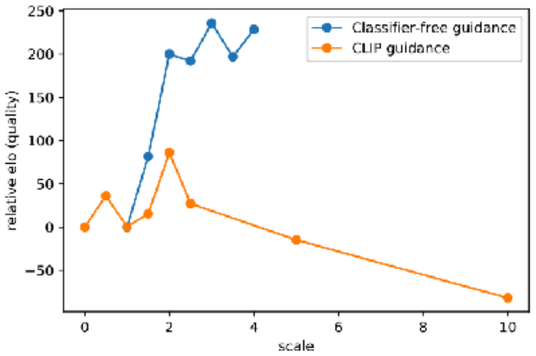
CLIP text encoding



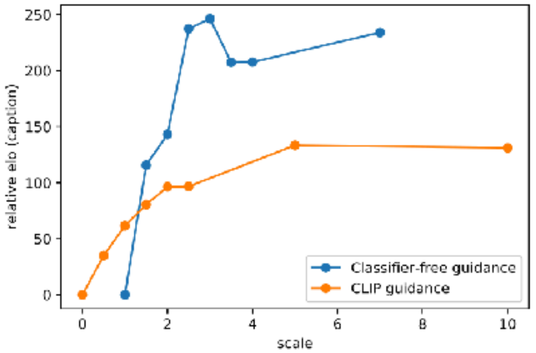
Nichol, Alex, et al. "Glide: Towards photorealistic image generation and editing with text-guided diffusion models." arXiv preprint arXiv:2112.10741 (2021).

Classfier-free Guidance vs. CLIP Guidance in GLIDE





(a) Photorealism



(b) Caption Similarity

Nichol, Alex, et al. "Glide: Towards photorealistic image generation and editing with text-guided diffusion models." arXiv preprint arXiv:2112.10741 (2021).