

Grooming Diffusion

- ◆ Advanced diffusion
-

NAVER AI Lab

김준호

<https://github.com/taki0112>



NAVER AI LAB



오전

오후

Basic Diffusion

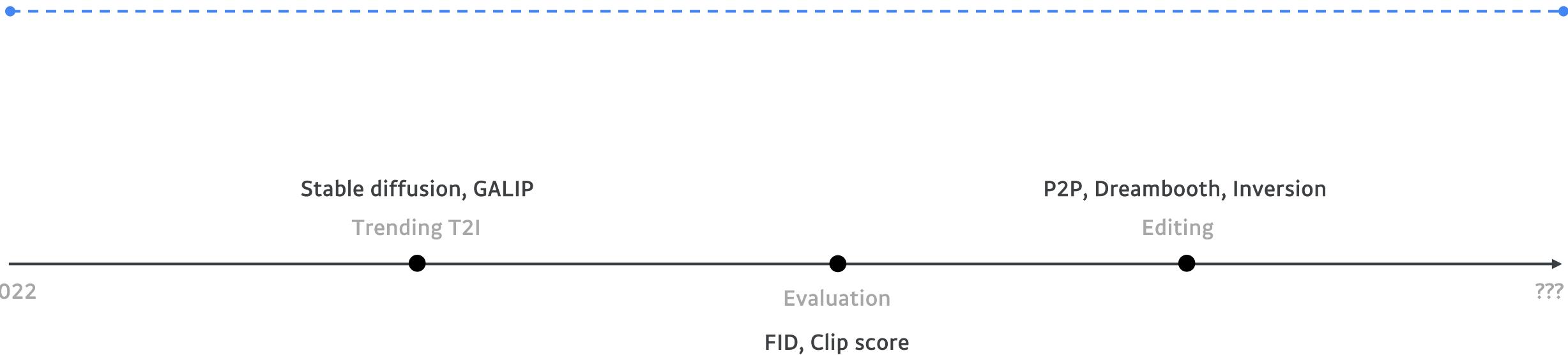
기본
: Scientist

Advanced Diffusion

심화
: Scientist, Engineer

Hands-on Diffusion

구현
: Scientist, Engineer





license Apache-2.0 release v0.7.2 Contributor Covenant 2.0

😊 Diffusers provides pretrained diffusion models across multiple modalities, such as vision and audio, and serves as a modular toolbox for inference and training of diffusion models.

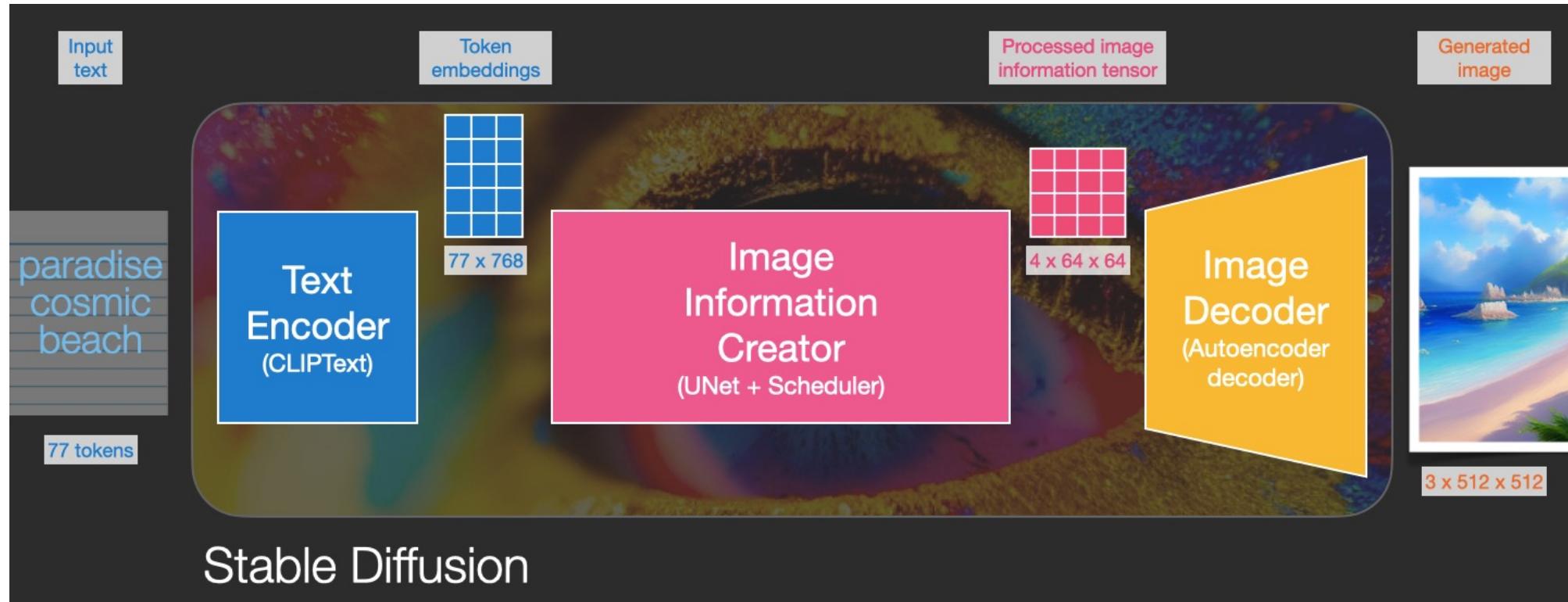
More precisely, 😊 Diffusers offers:

- State-of-the-art diffusion pipelines that can be run in inference with just a couple of lines of code (see [src/diffusers/pipelines](#)). Check [this overview](#) to see all supported pipelines and their corresponding official papers.
- Various noise schedulers that can be used interchangeably for the preferred speed vs. quality trade-off in inference (see [src/diffusers/schedulers](#)).
- Multiple types of models, such as UNet, can be used as building blocks in an end-to-end diffusion system (see [src/diffusers/models](#)).
- Training examples to show how to train the most popular diffusion model tasks (see [examples](#), e.g. [unconditional-image-generation](#)).



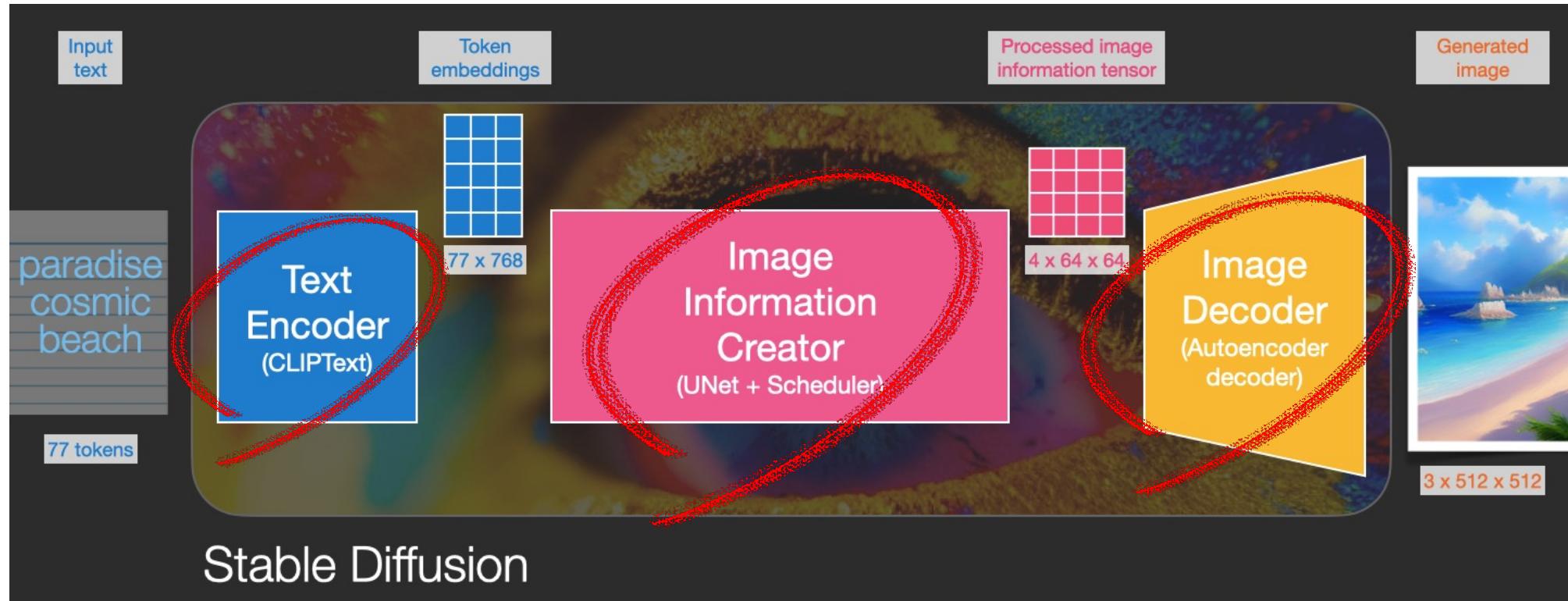
Stable Diffusion

Components



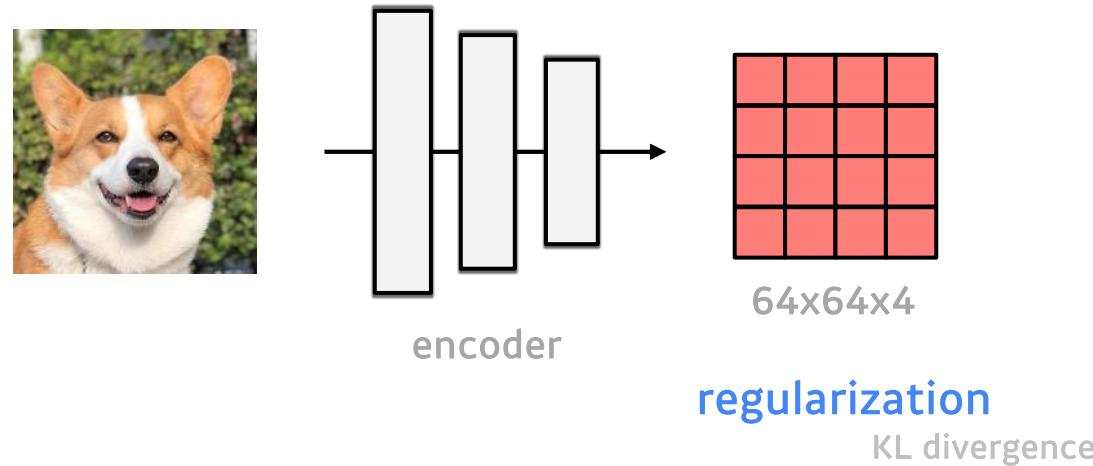
Stable Diffusion

Components



Stable Diffusion

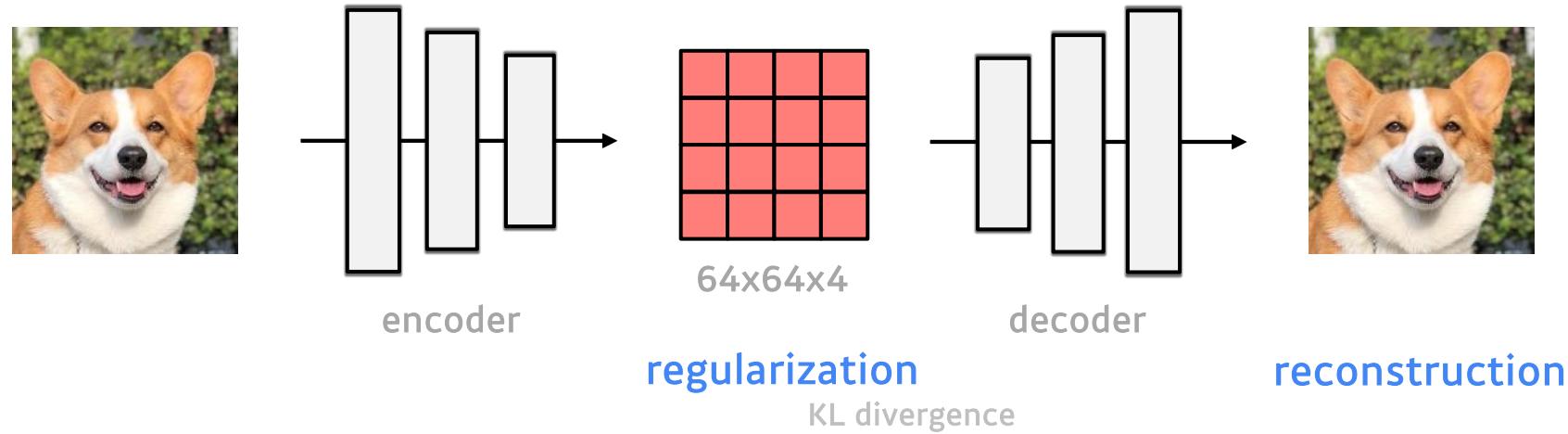




Stable Diffusion

| Autoencoder

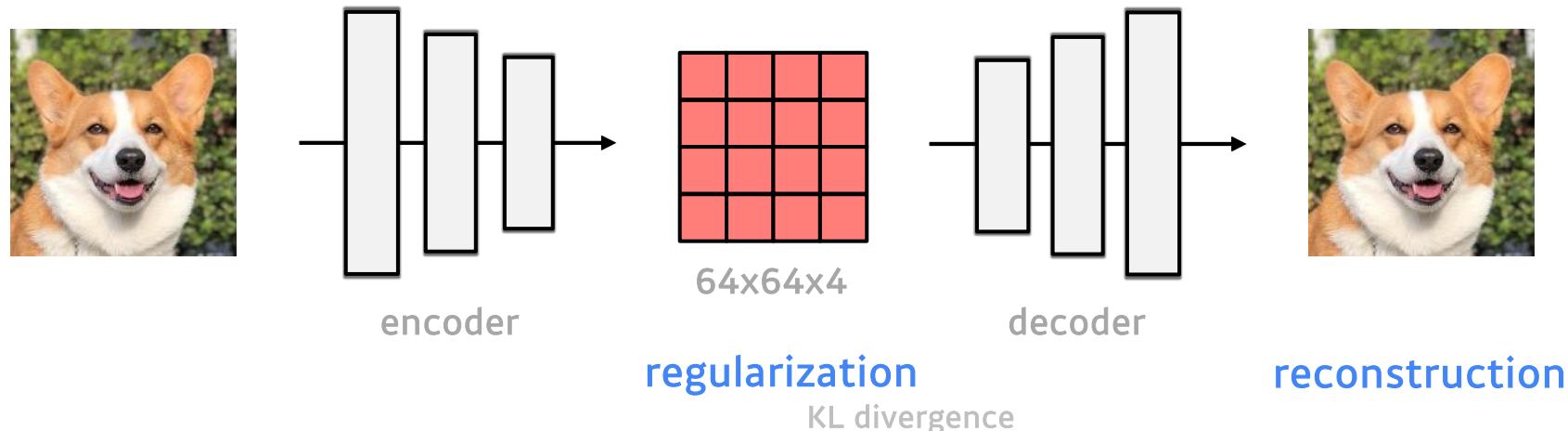
 NAVER AI LAB



Stable Diffusion

| Autoencoder

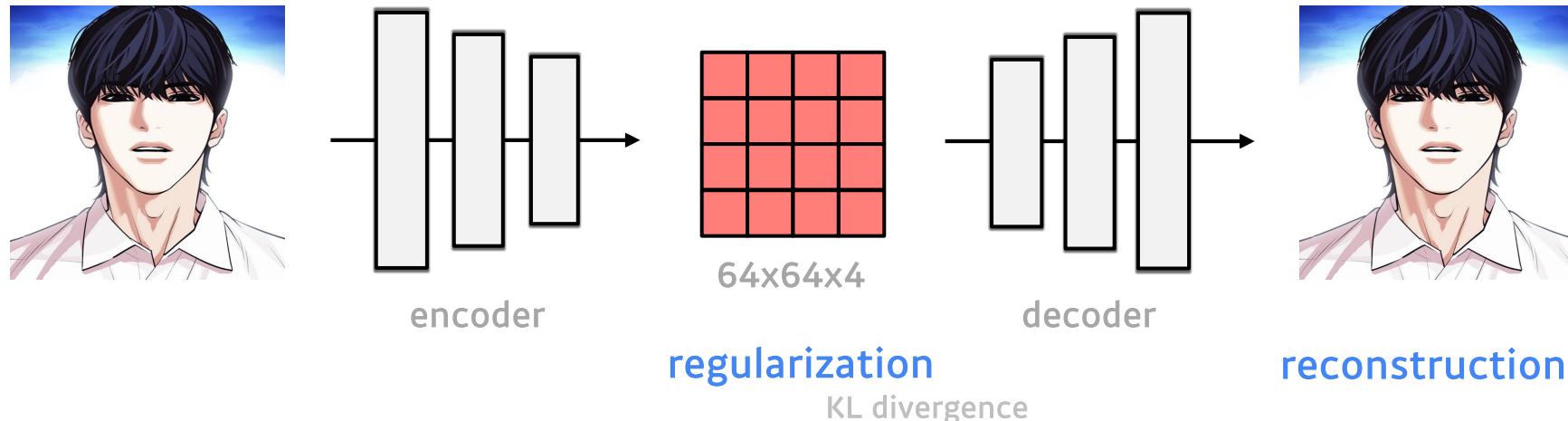
 NAVER AI LAB



Stable Diffusion

Autoencoder

 NAVER AI LAB



“The photo of
a welsh corgi”



text encoder

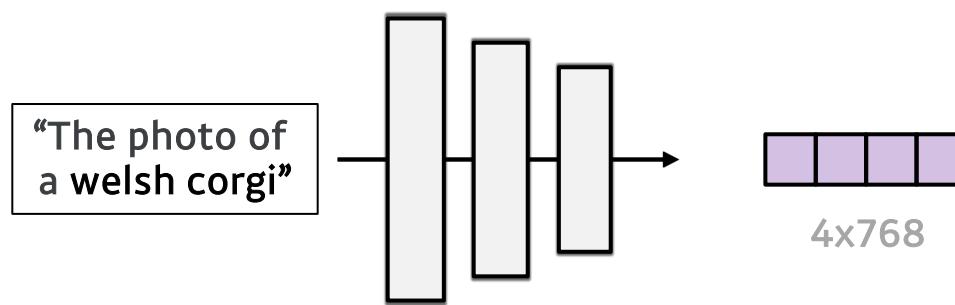
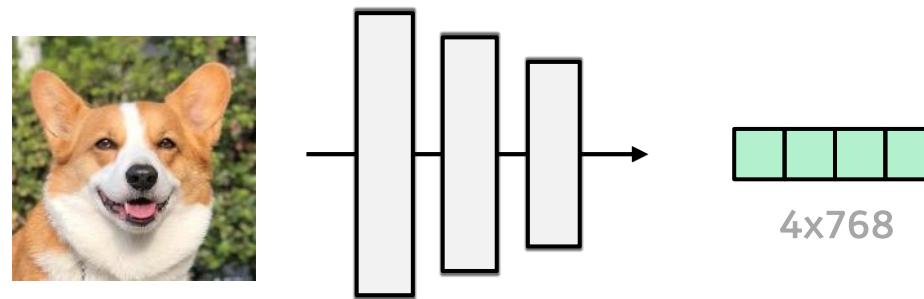


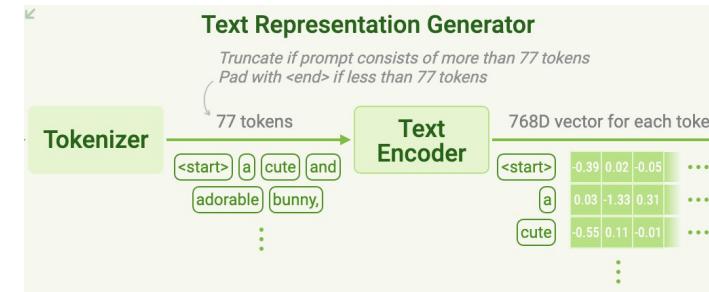
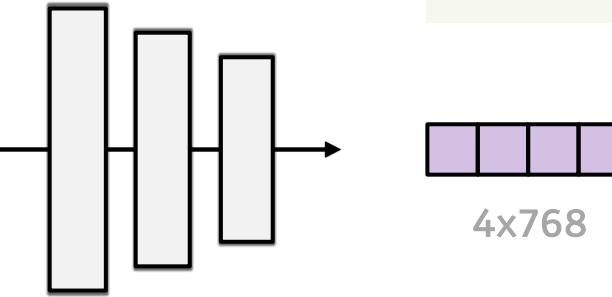
image encoder



Stable Diffusion | CLIP

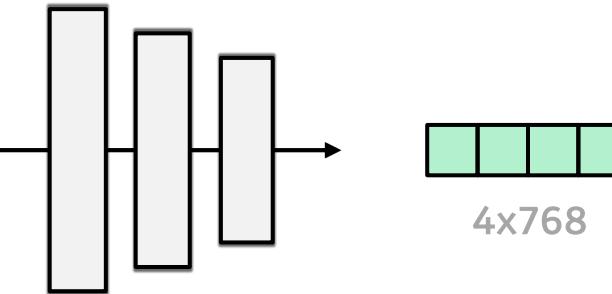
text encoder

“The photo of
a welsh corgi”



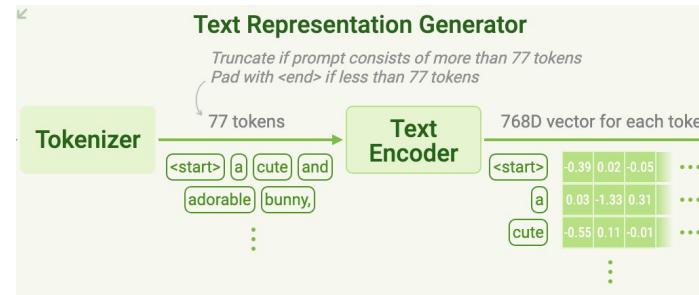
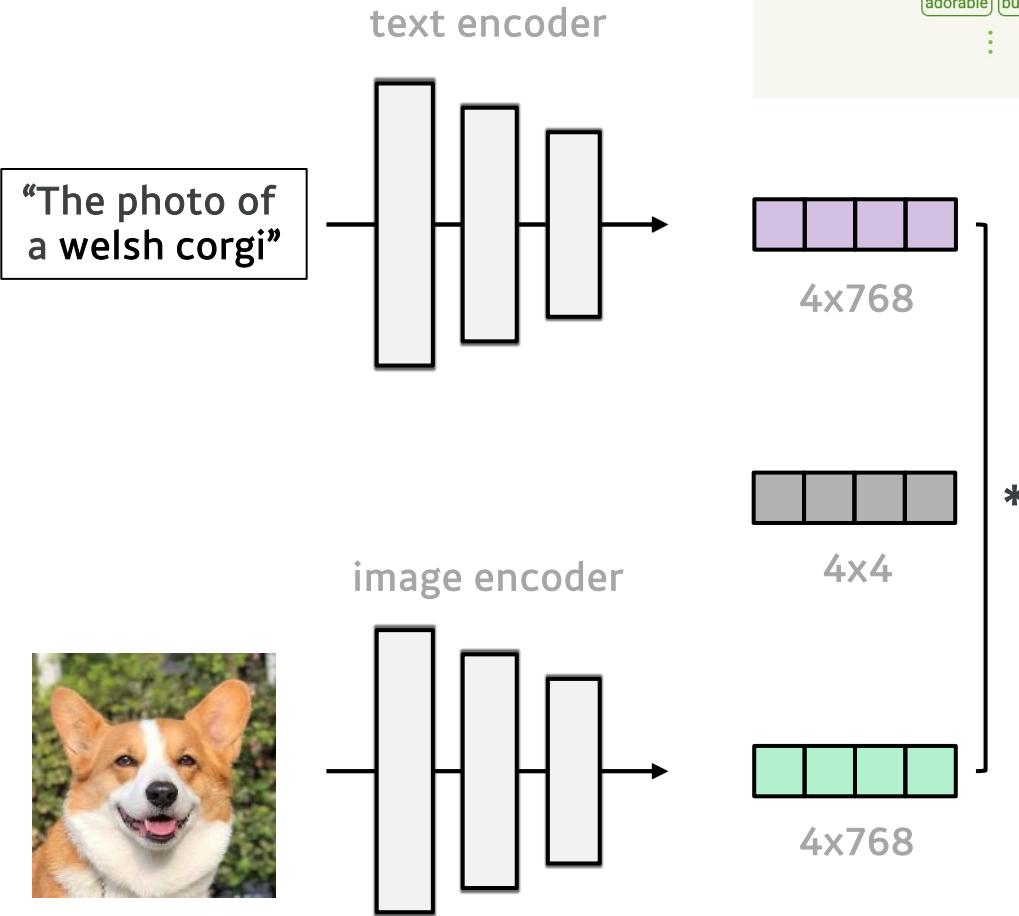
4x768

image encoder

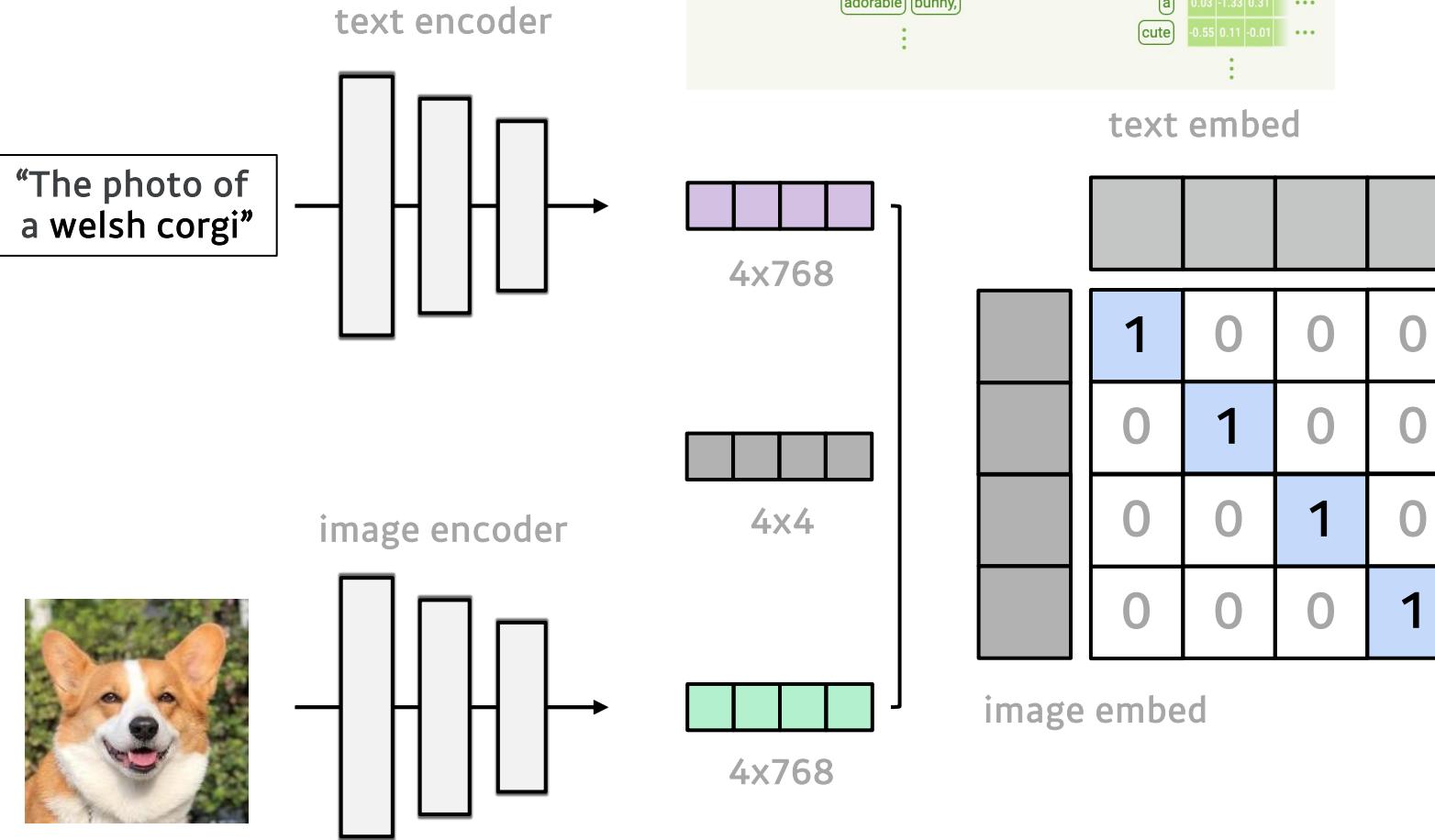


4x768

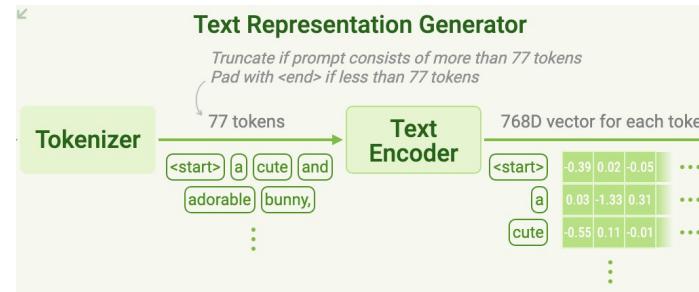
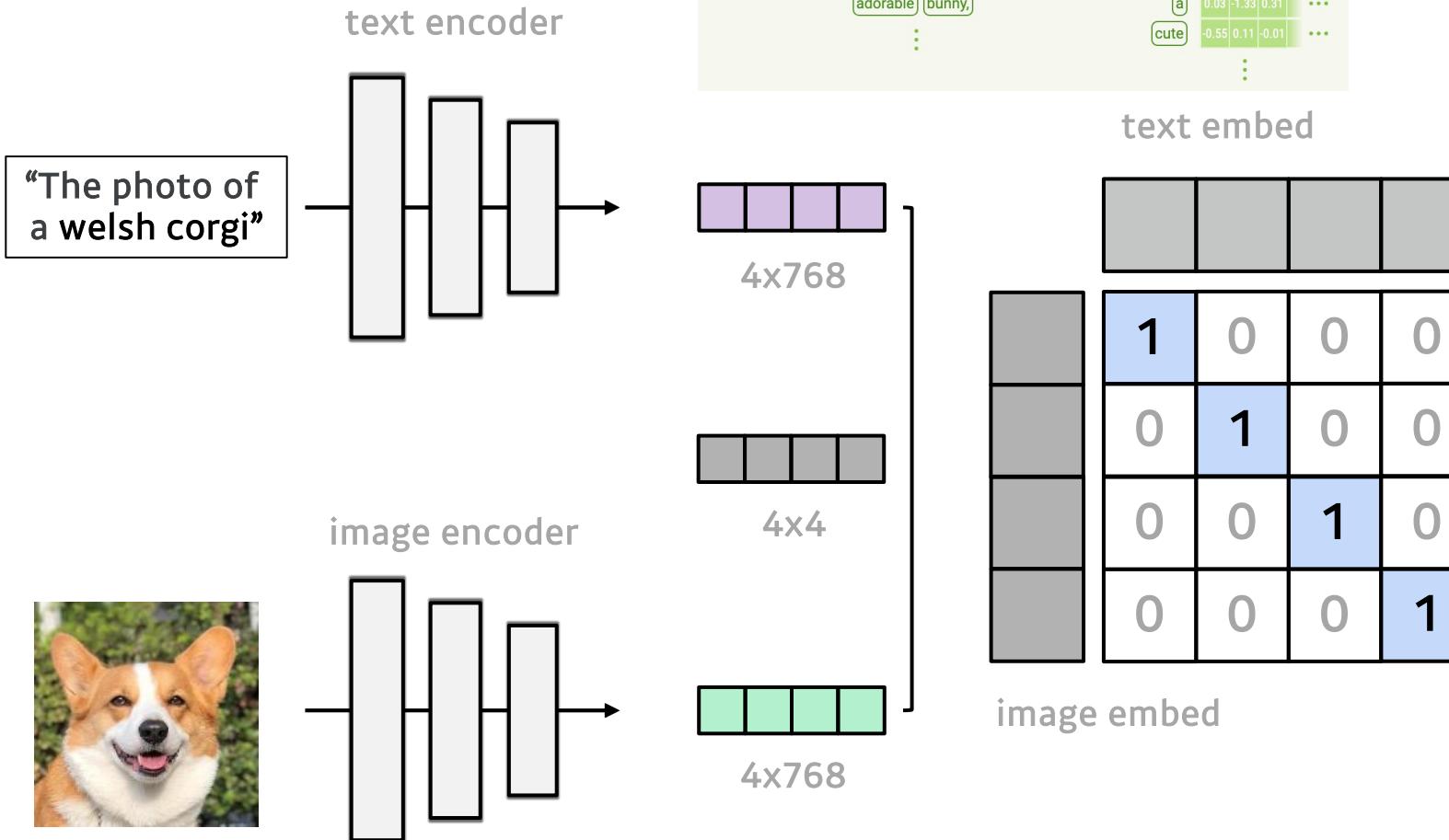
Stable Diffusion | CLIP



Stable Diffusion | CLIP



Stable Diffusion | CLIP



LAION-5B, 84.8TB

2B-en

```
import torch
from PIL import Image
import open_clip

model, _, preprocess = open_clip.create_model_and_transforms('ViT-B-32-quickgelu', pretrained='laion5b_v1.4')
tokenizer = open_clip.get_tokenizer('ViT-B-32-quickgelu')

image = preprocess(Image.open("CLIP.png")).unsqueeze(0)
text = tokenizer(["a diagram", "a dog", "a cat"])

with torch.no_grad(), torch.cuda.amp.autocast():
    image_features = model.encode_image(image)
    text_features = model.encode_text(text)
    image_features /= image_features.norm(dim=-1, keepdim=True)
    text_features /= text_features.norm(dim=-1, keepdim=True)

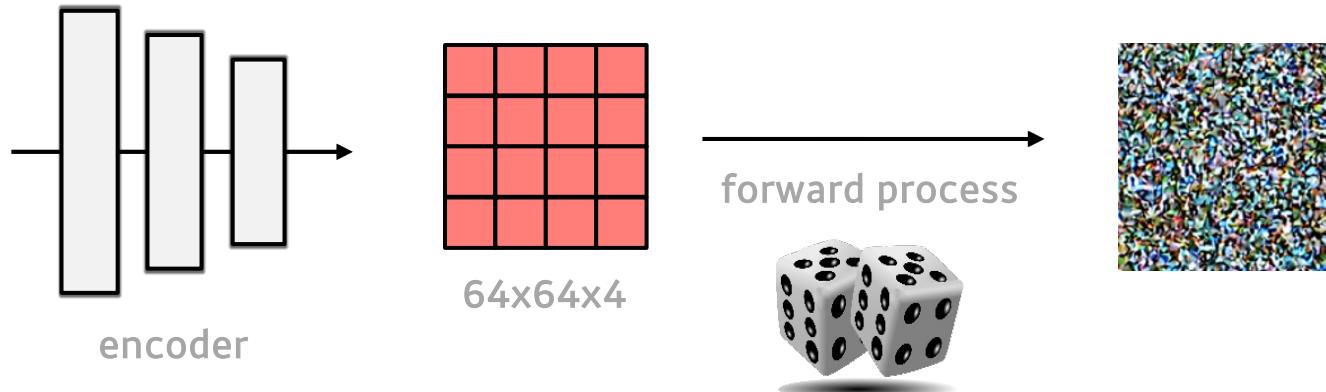
    text_probs = (100.0 * image_features @ text_features.T).softmax(dim=-1)

print("Label probs:", text_probs) # prints: [[1., 0., 0.]]
```

github.com/mlfoundations/open_clip

Stable Diffusion

| Training

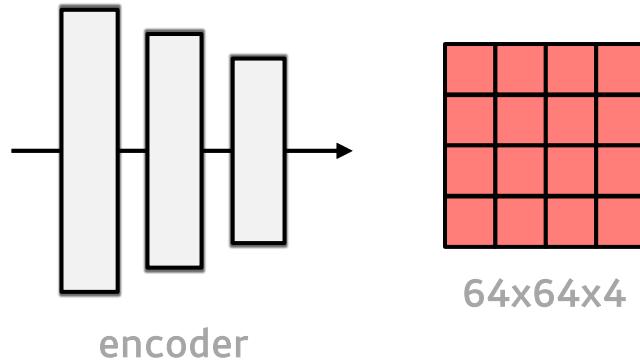


step 1

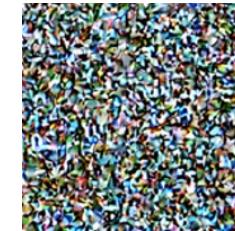
Stable Diffusion

| Training

 NAVER AI LAB

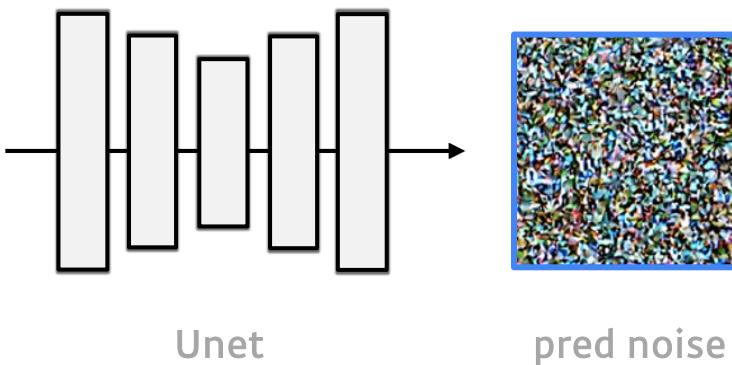


forward process



step 1

time embed

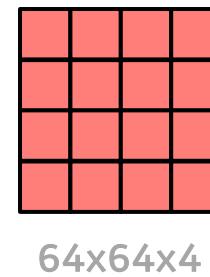
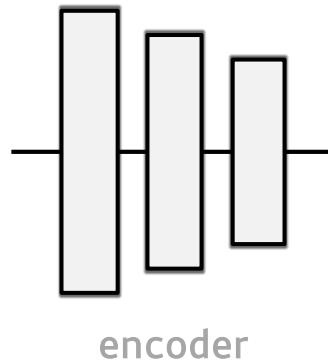


pred noise

text embed
77x768

Stable Diffusion

| Training



forward process

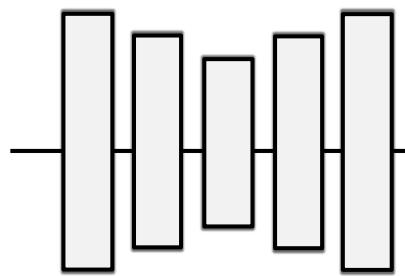


step 1

time embed



text embed
77x768



pred noise

L2 loss

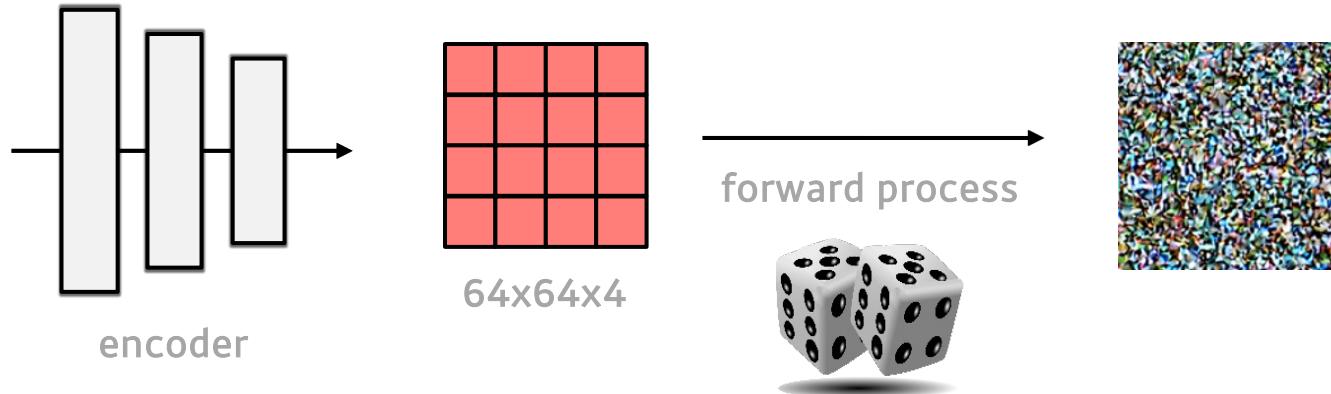


-
GT noise

Stable Diffusion

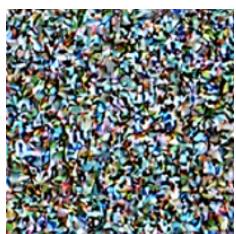
| Training

 NAVER AI LAB

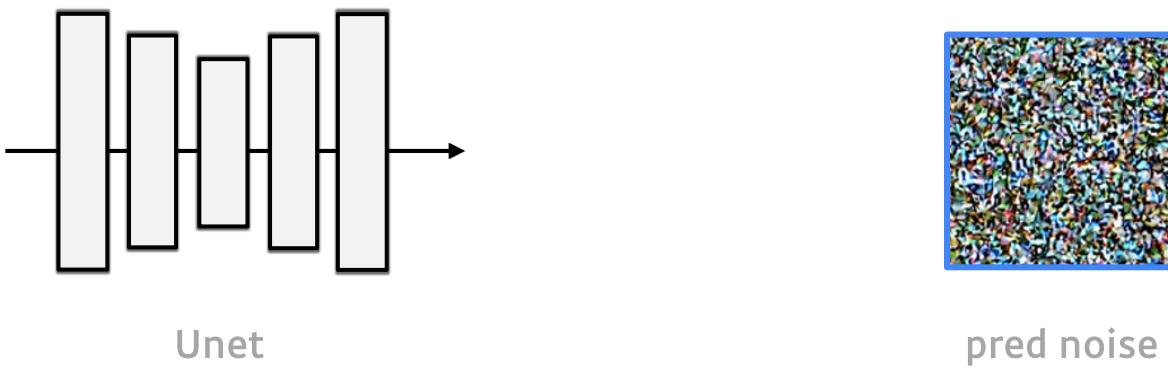


step 1

time embed



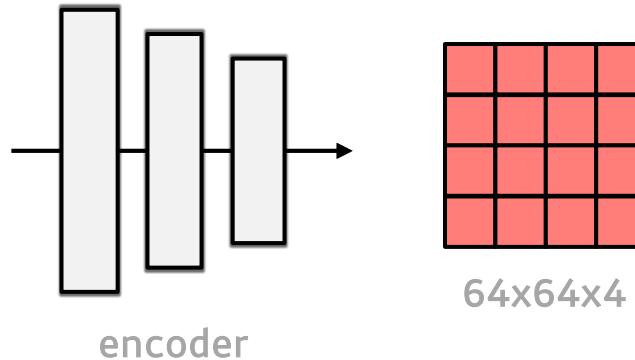
text embed
77x768



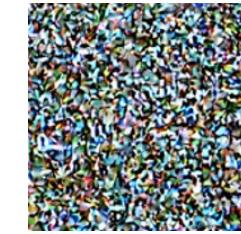
pred noise

Stable Diffusion

Training

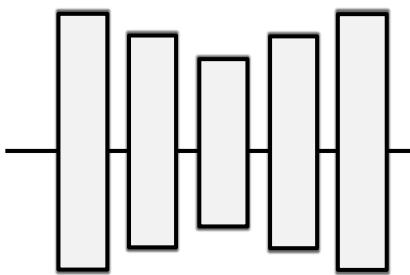
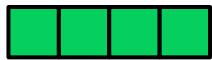


forward process



step 1

time embed



input latent

pred noise

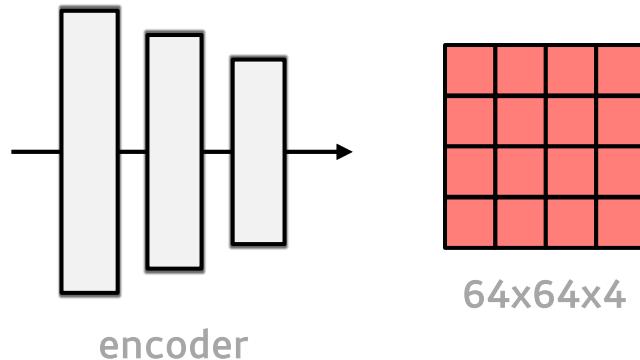
64x64x4

text embed
77x768

Unet

Stable Diffusion

Training

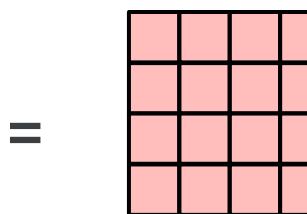
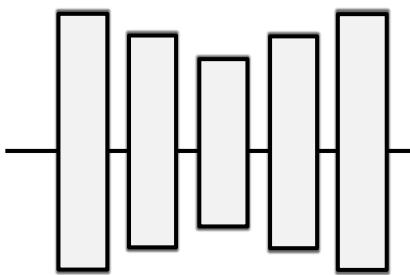
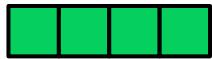


forward process



step 4

time embed

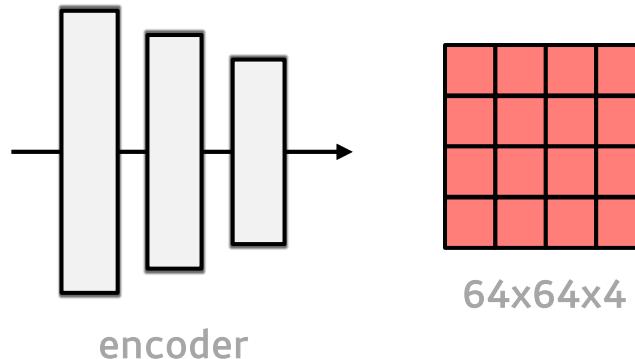


64x64x4

text embed
77x768

Stable Diffusion

Training

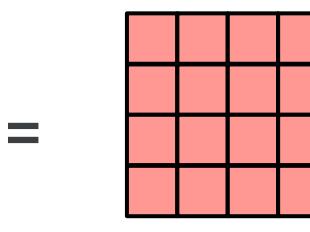
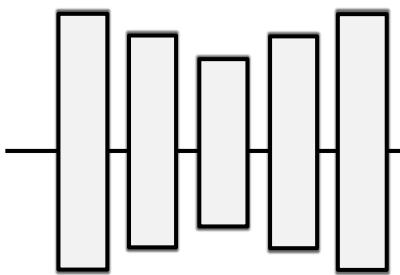


forward process



step 10

time embed



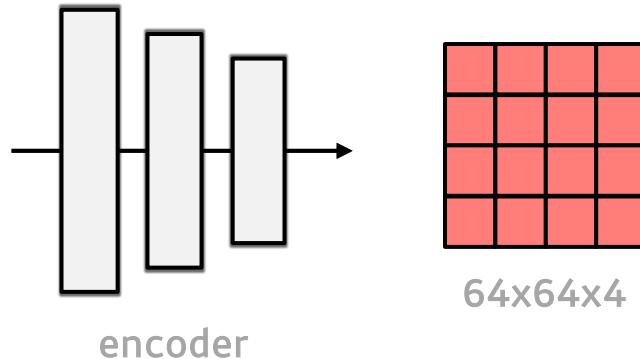
64x64x4

text embed
77x768

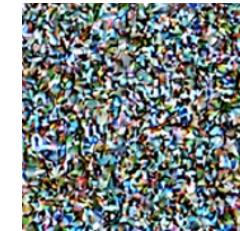
Stable Diffusion

| Training

 NAVER AI LAB

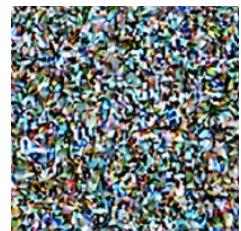
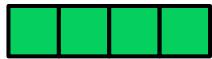


forward process



step 50

time embed



text embed
77x768

Unet

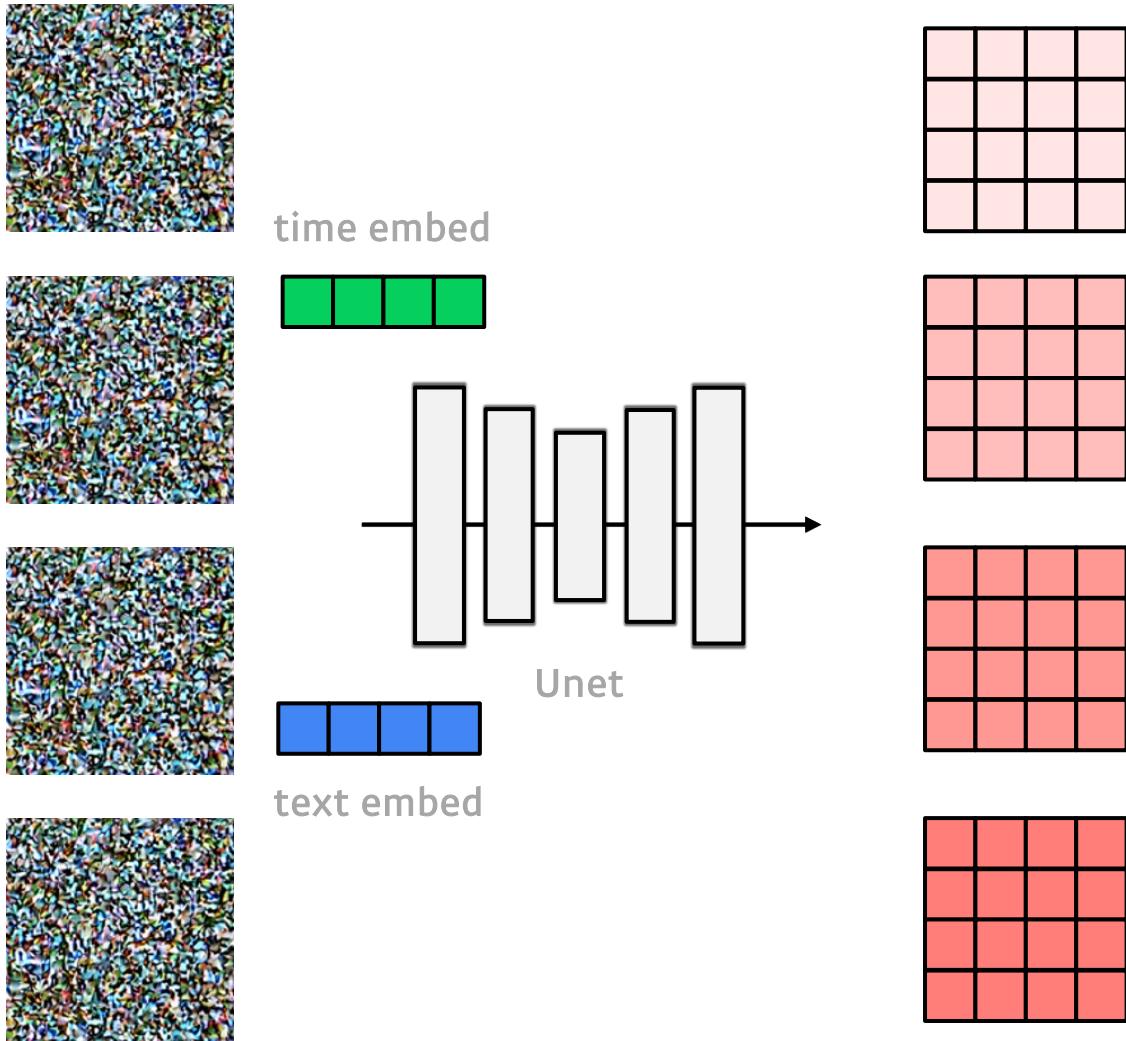
input latent

pred noise

64x64x4

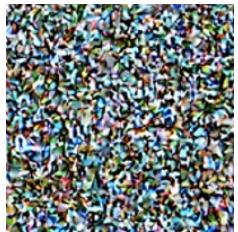
Stable Diffusion

| Training results

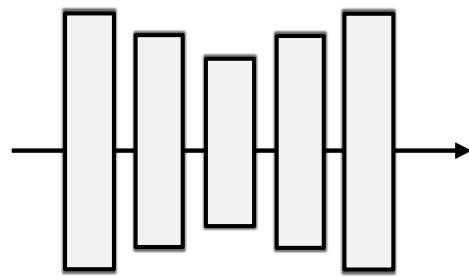
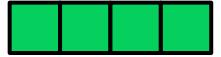
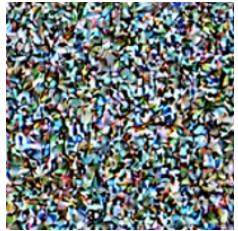


Stable Diffusion

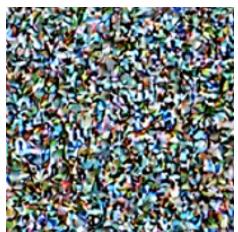
| Training results



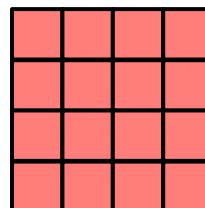
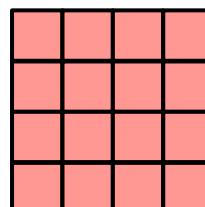
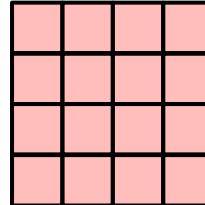
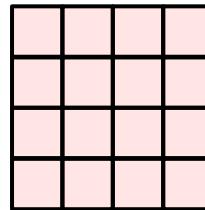
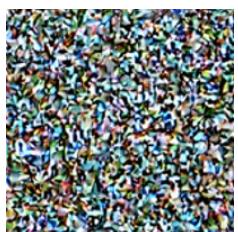
time embed



Unet



text embed



step 1



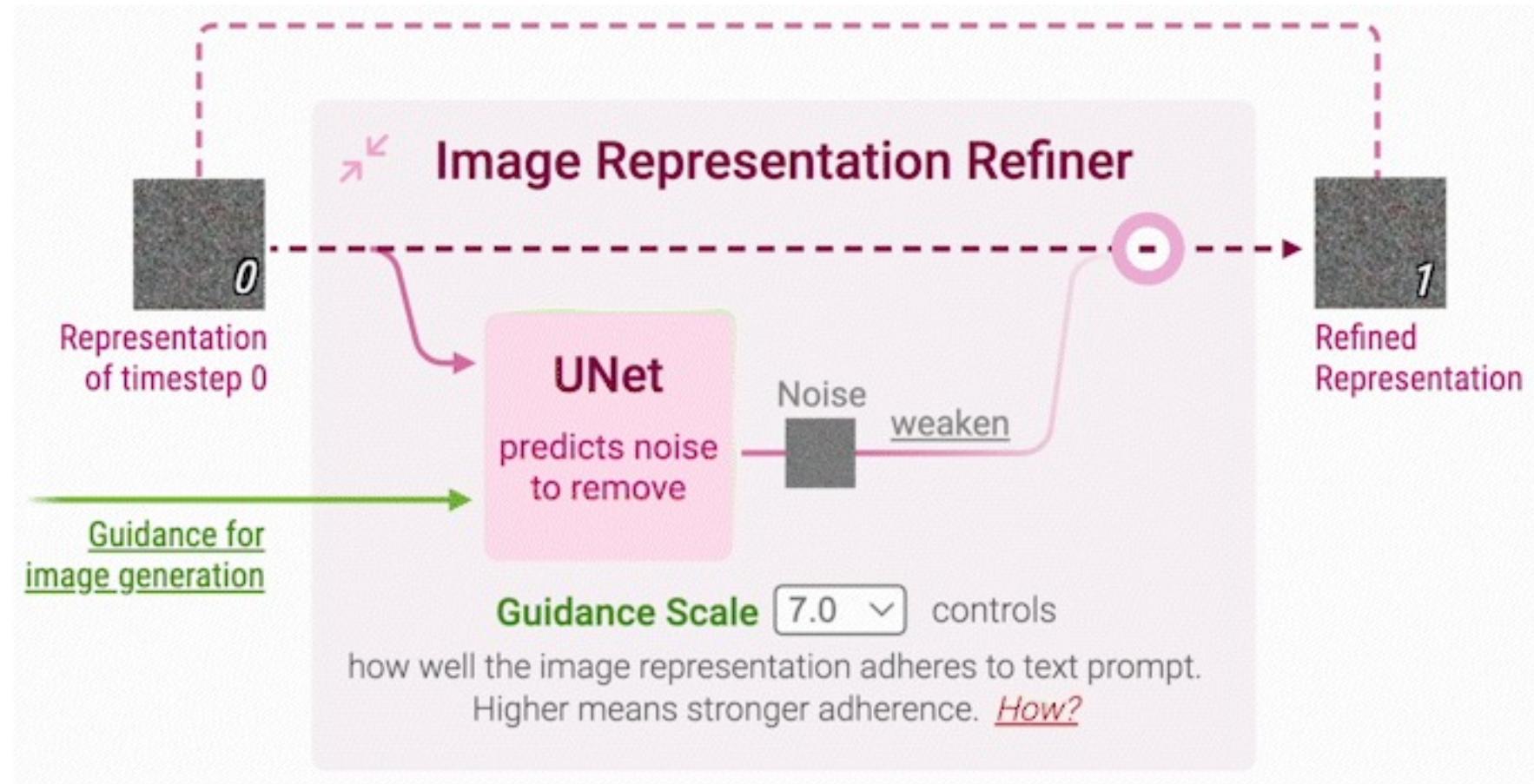
step 4



step 10

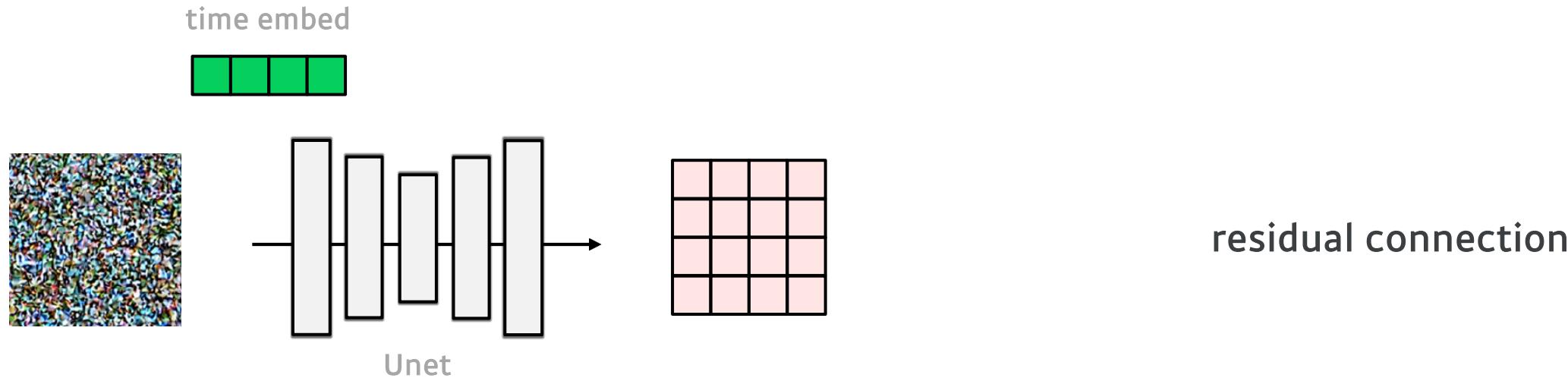


step 50

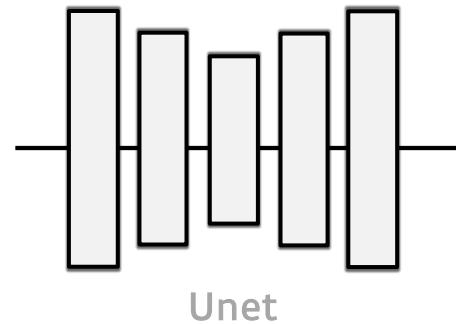


Stable Diffusion

Conditioning

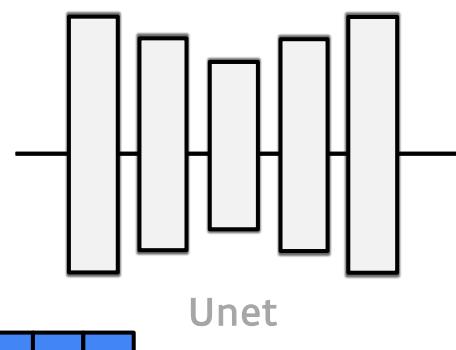


time embed



residual connection

time embed

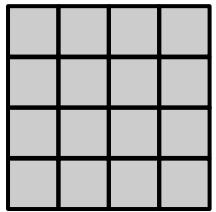


cross attention

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$
$$Q = (N, d_k), K = (N, d_k), V = (N, d_v)$$

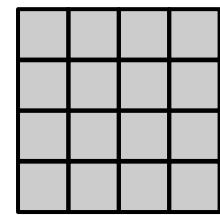
Stable Diffusion

Self-attention vs Cross-attention

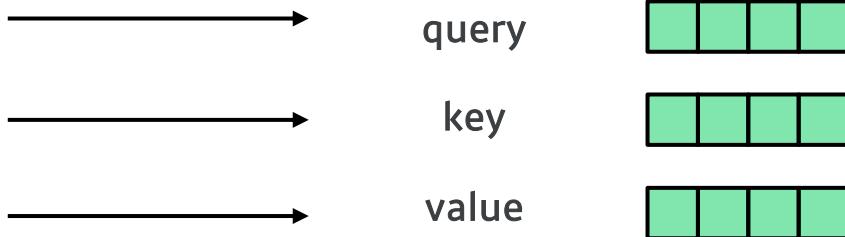


feature map

Stable Diffusion | Self-attention vs Cross-attention



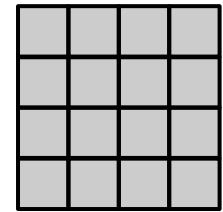
feature map



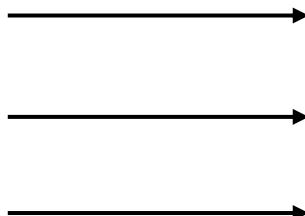
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$
$$Q = (N, d_k), K = (N, d_k), V = (N, d_v)$$

[N, seq_len, dim]

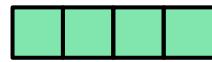
Stable Diffusion | Self-attention vs Cross-attention



feature map



query



key

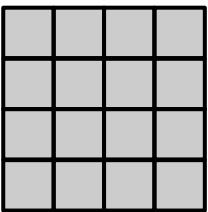


value



$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$
$$Q = (N, d_k), K = (N, d_k), V = (N, d_v)$$

[N, seq_len, dim]



feature map



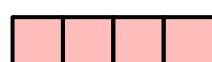
query



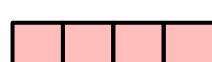
text embed



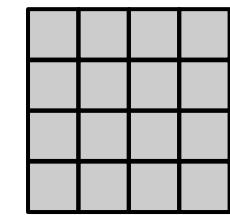
key



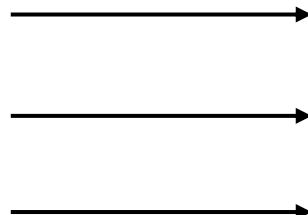
value



Stable Diffusion | Self-attention vs Cross-attention



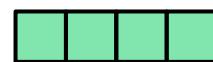
feature map



query



key

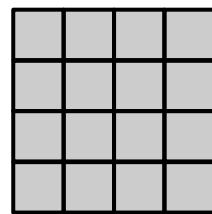


value



$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$
$$Q = (N, d_k), K = (N, d_k), V = (N, d_v)$$

[N, seq_len, dim]



feature map



query



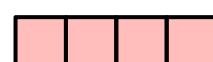
관계성
QK



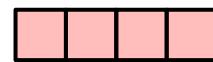
text embed



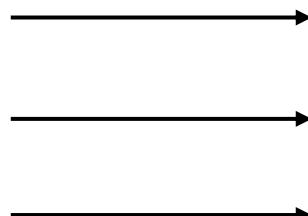
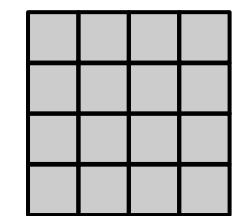
key



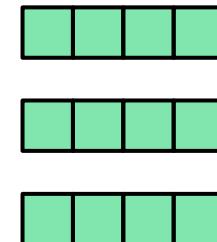
value



Stable Diffusion | Self-attention vs Cross-attention



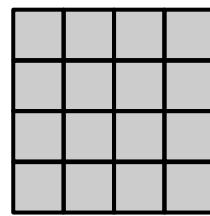
query
key
value



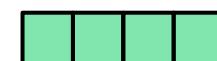
feature map

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$
$$Q = (N, d_k), K = (N, d_k), V = (N, d_v)$$

[N, seq_len, dim]



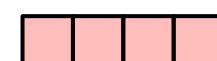
query



feature map



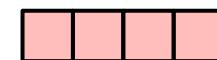
key



text embed



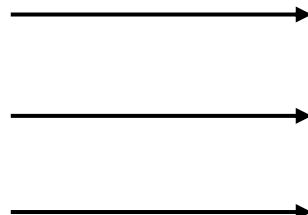
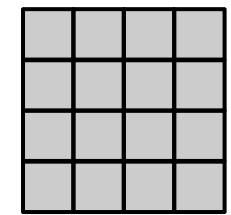
value



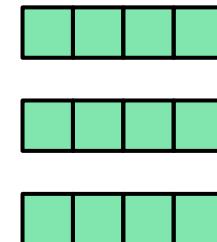
관계성
QK

가중치
QK * V

Stable Diffusion | Self-attention vs Cross-attention



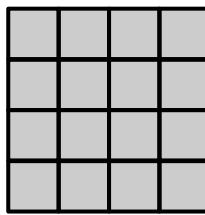
query
key
value



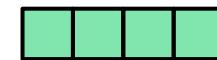
feature map

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$
$$Q = (N, d_k), K = (N, d_k), V = (N, d_v)$$

[N, seq_len, dim]



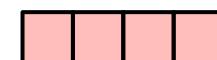
query



feature map

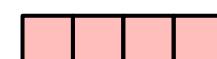


key



text embed

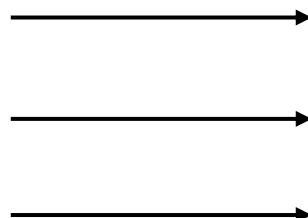
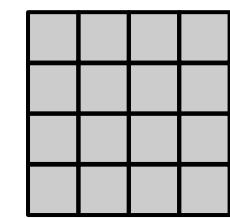
value



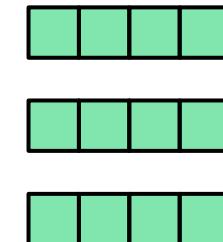
관계성
QK

가중치
QK * V
특정 단어 강조 가능

Stable Diffusion | Self-attention vs Cross-attention



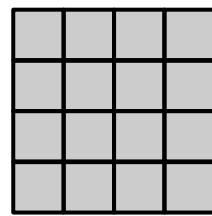
query
key
value



feature map

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$
$$Q = (N, d_k), K = (N, d_k), V = (N, d_v)$$

[N, seq_len, dim]



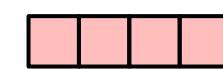
query



feature map



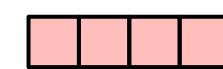
key



text embed



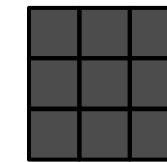
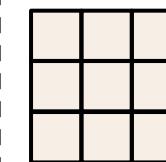
value



관계성
QK

가중치
 $QK * V$
특정 단어 강조 가능

mask matrix 활용 가능



QK

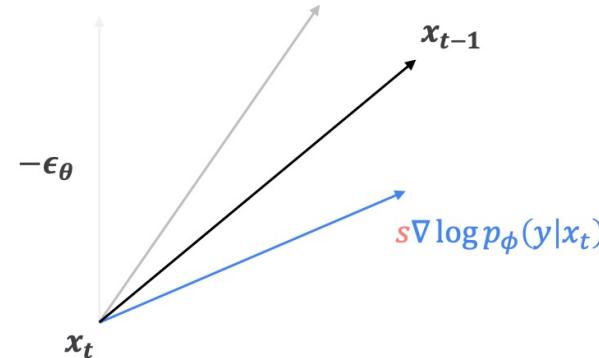
mask

Stable Diffusion | Classifier free guidance

$$x_{t-1} \leftarrow N(\mu + s * \Sigma \nabla \log p_\phi(y|x_t), \Sigma)$$



Figure 3: Samples from an unconditional diffusion model with classifier guidance to condition on the class "Pembroke Welsh corgi". Using classifier scale 1.0 (left; FID: 33.0) does not produce convincing samples in this class, whereas classifier scale 10.0 (right; FID: 12.0) produces much more class-consistent images.



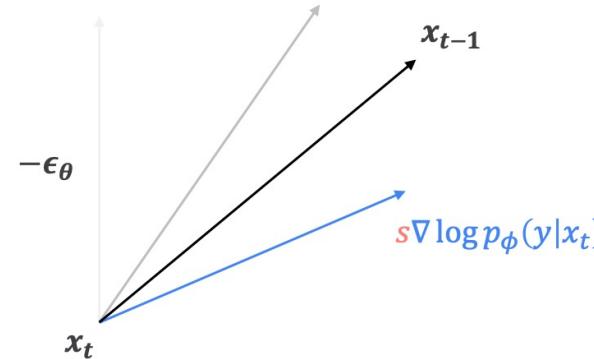
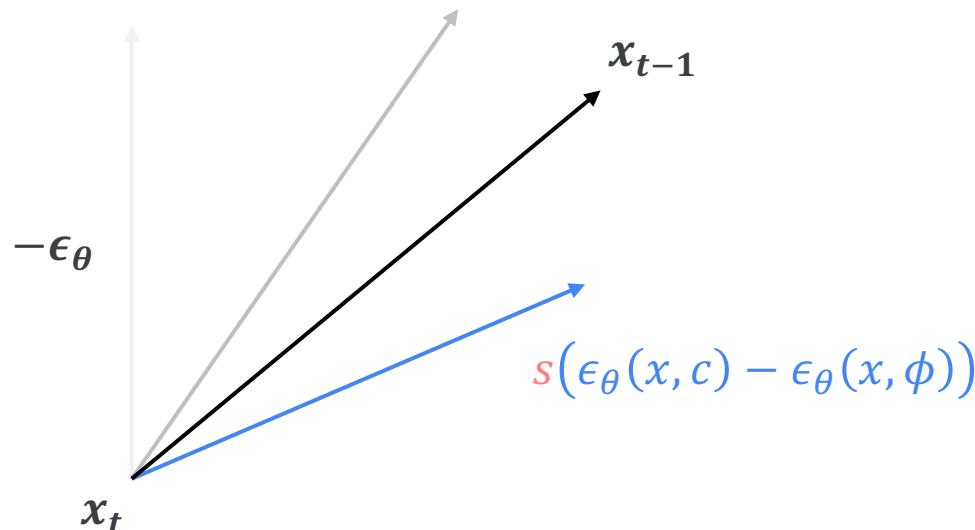
Stable Diffusion | Classifier free guidance

$$x_{t-1} \leftarrow N(\mu + s * \Sigma \nabla \log p_\phi(y|x_t), \Sigma)$$



Figure 3: Samples from an unconditional diffusion model with classifier guidance to condition on the class "Pembroke Welsh corgi". Using classifier scale 1.0 (left; FID: 33.0) does not produce convincing samples in this class, whereas classifier scale 10.0 (right; FID: 12.0) produces much more class-consistent images.

$$\tilde{\epsilon}_\theta(x, c) = \epsilon_\theta(x, \phi) + s * (\epsilon_\theta(x, c) - \epsilon_\theta(x, \phi))$$



Stable Diffusion

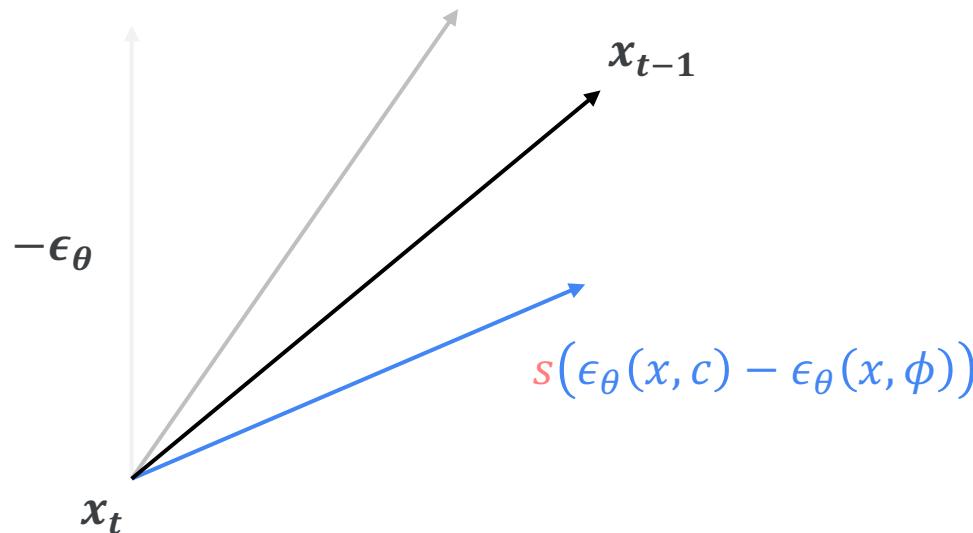
Classifier free guidance

$$x_{t-1} \leftarrow N(\mu + s * \Sigma \nabla \log p_\phi(y|x_t), \Sigma)$$



Figure 3: Samples from an unconditional diffusion model with classifier guidance to condition on the class "Pembroke Welsh corgi". Using classifier scale 1.0 (left; FID: 33.0) does not produce convincing samples in this class, whereas classifier scale 10.0 (right; FID: 12.0) produces much more class-consistent images.

$$\tilde{\epsilon}_\theta(x, c) = \epsilon_\theta(x, \phi) + s * (\epsilon_\theta(x, c) - \epsilon_\theta(x, \phi))$$



Stable Diffusion

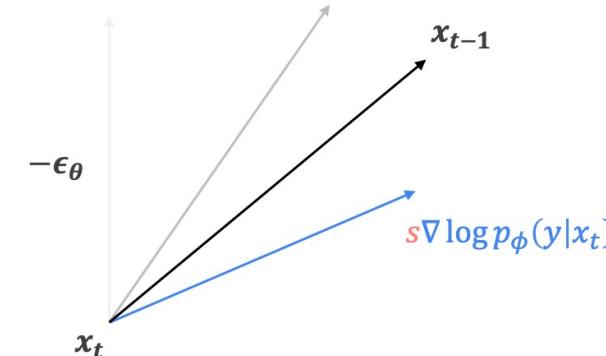
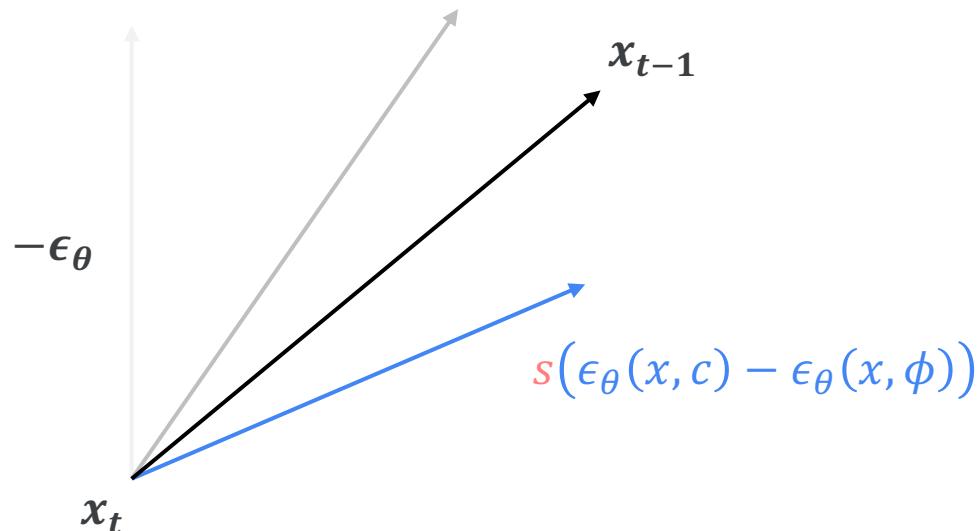
Classifier free guidance

$$x_{t-1} \leftarrow N(\mu + s * \Sigma \nabla \log p_\phi(y|x_t), \Sigma)$$



Figure 3: Samples from an unconditional diffusion model with classifier guidance to condition on the class "Pembroke Welsh corgi". Using classifier scale 1.0 (left; FID: 33.0) does not produce convincing samples in this class, whereas classifier scale 10.0 (right; FID: 12.0) produces much more class-consistent images.

$$\tilde{\epsilon}_\theta(x, c) = \epsilon_\theta(x, \phi) + s * (\epsilon_\theta(x, c) - \epsilon_\theta(x, \phi))$$



```
# perform guidance
if do_classifier_free_guidance:
    noise_pred_uncond, noise_pred_text = noise_pred.chunk(2)
    noise_pred = noise_pred_uncond + guidance_scale * (noise_pred_text - noise_pred_uncond)
```

Stable Diffusion

Application



A collage of images created by Stable Diffusion. **Image Credits:** Daniel Jeffries



스페이스 오로라 극장, 제이스 앤런

Stable Diffusion | Application



스페이스 오로라 극장, 제이스 앤런



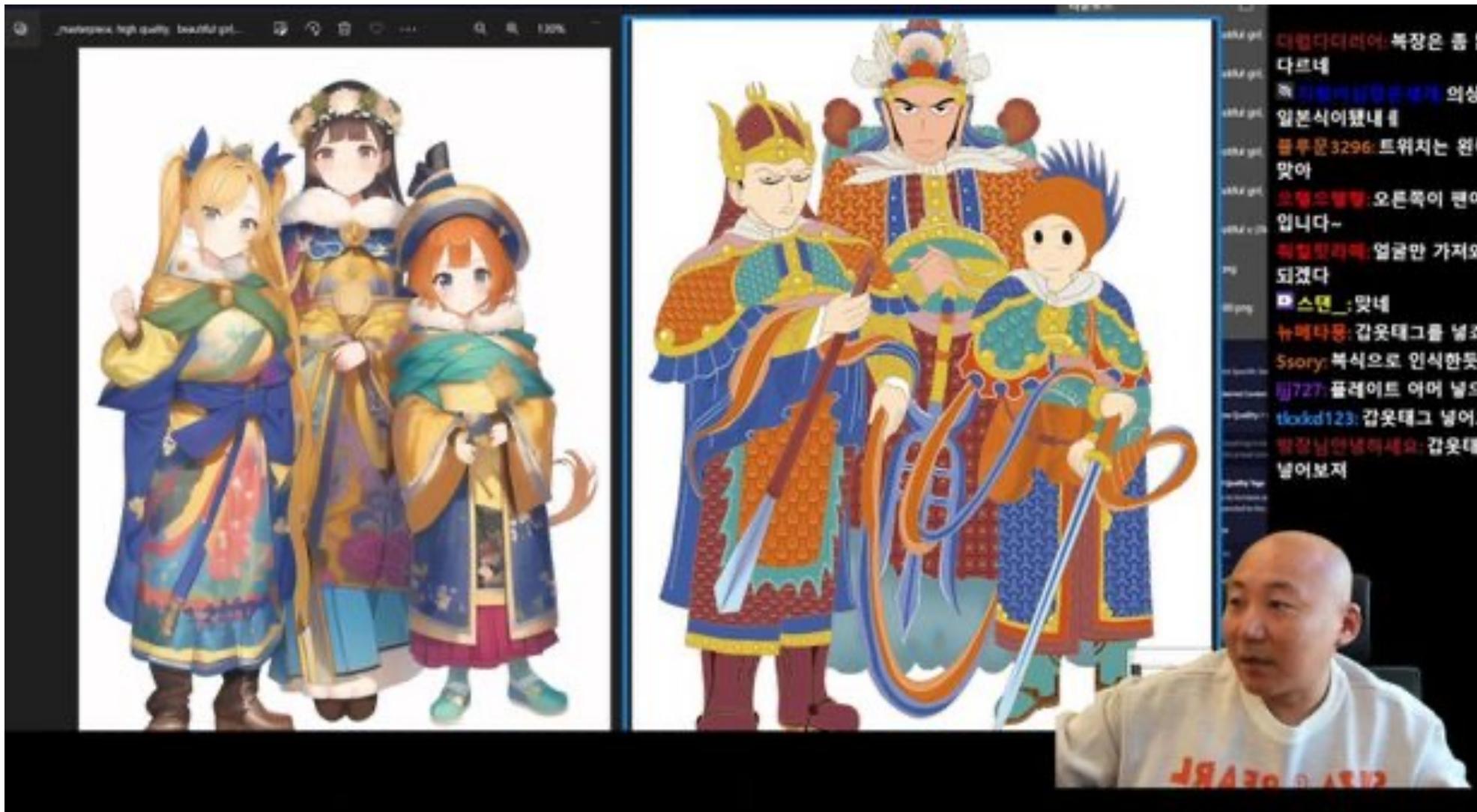
Stable Diffusion

| Image inpainting

 NAVER AI LAB



Stable Diffusion | Image inpainting



Stable Diffusion | Application

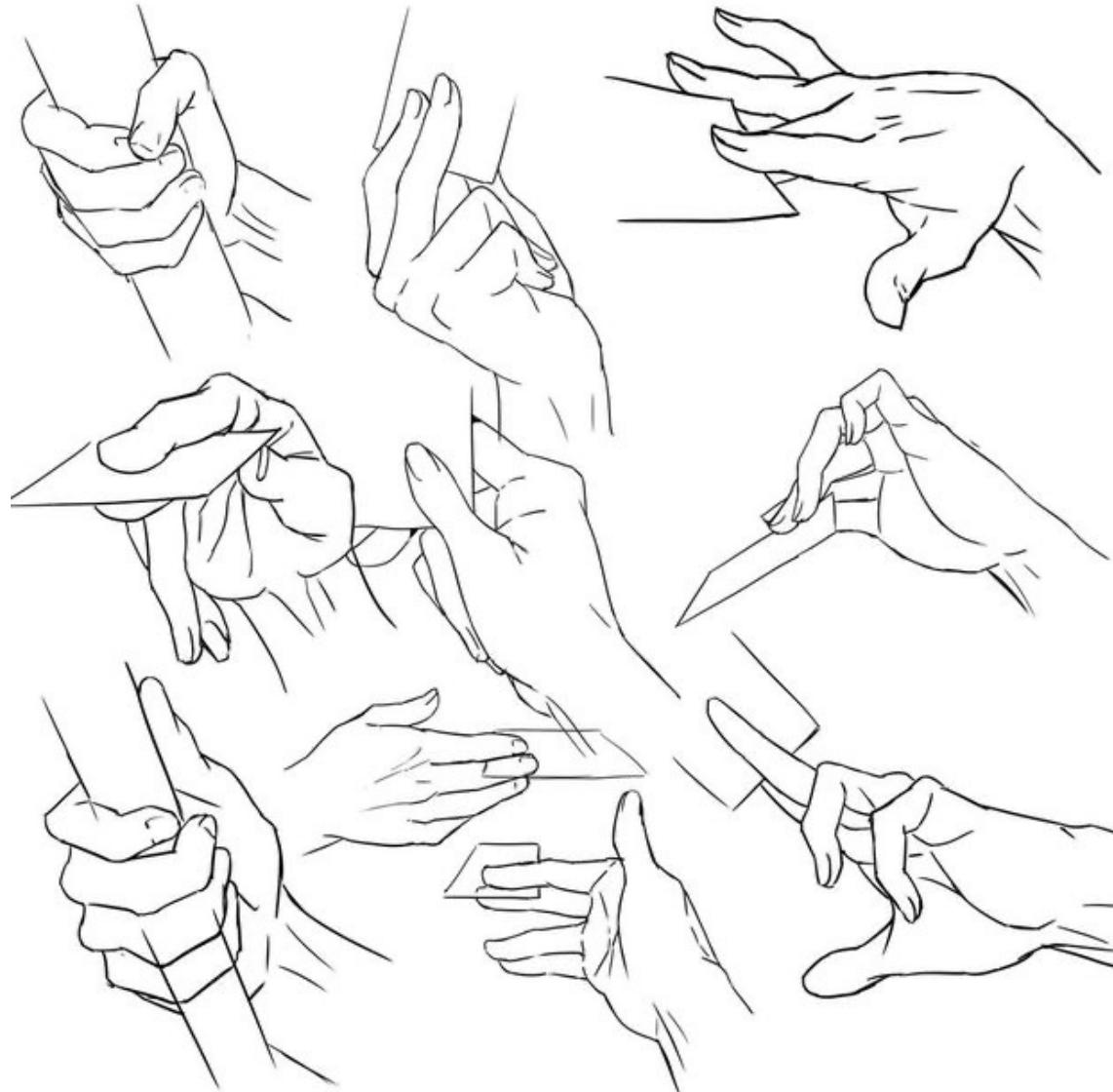
 NAVER AI LAB



Stable Diffusion

Application

 NAVER AI LAB



Stable Diffusion

Application

 NAVER AI LAB





Stable Diffusion | Application

전체 ALL	주 기술 PRIMARY	보조 기술 SECONDARY	방어 DEFENSIVE	사냥 HUNTING	장치 DEVICES	궁술 ARCHERY
레벨 아이콘 설명 편성 영상 보기						
굶주린 화살 [8] 주 기술 생성: 증오 4 마력이 깃든 화살을 발사합니다. 화살은 대상을 추적하여 무기 공격력의 15%만큼 피해를 주고, 35% 확률로 여러 적을 관통합니다. 활 필요						
관통 화살 들판 화살 파편 사격 어귀 화살 출날리는 뱃조..						
준비중						
속박탄 [1] 주 기술 생성: 증오 4 화살에 어둠의 기운을 불어넣어 대상에게 무기 공격력의 200% 만큼 피해를 주고, 최대 2마리 적의 이동 속도를 2초 동안 60% 감소시킵니다. 속박탄이 적에게 적중하면 속박 상태인 모든 적의 감속 효과가 초기화됩니다. 활 필요						
글비 여기 전깃줄 무거운 짐 정의 구현 현상금 사냥꾼						
준비중						
울가미 주 기술 생성: 증오 4 적을 휘감는 울가미를 발사합니다. 울가미는 1초 뒤에 폭발하여 대상에게 무기 공격력의 160%만큼 화염 피해를 주고 대상으로부터 14미터 내에 있는 모든 적들에게 무기 공격력의 110%만큼 화염 피해를 줍니다.						
강화 폭탄 천둥구 빙결의 일격 입에 쓴 약 파멸의 도래						
준비중						
회피 사격 [2] 주 기술 생성: 증오 4 다수의 화살을 날려 주 대상에게 무기 공격력의 200%만큼 피해를 주고 추가적으로 적 두 마리에게 무기 공격력의 100%만큼 피해를 줍니다. 적이 정면에 가까이 있으면 5미터 뒤로 공중제비를 넘습니다. 활 필요						
단련 작별 선물 엄호 사격 침중 과도 전류						
준비중						
수류탄 주 기술 생성: 증오 4 통통 뛰는 수류탄을 던집니다. 수류탄은 폭발하여 무기 공격력의 160%만큼 화염 피해를 줍니다.						
땅장이 집수 수류탄 상비 수류탄 섬광 수류탄 냉기 수류탄						
준비중						

전체 ALL	주 기술 PRIMARY	보조 기술 SECONDARY	방어 DEFENSIVE	위력 FORCE	창조 CONJURATION	통달 MASTERY
레벨 아이콘 설명 편성 영상 보기						
힘이파동 [2] 위력 소모: 비전력 25 순수한 마력 파동을 일으켜 근처의 적에게 무기 공격력의 390% 만큼 비전 피해를 줍니다.						
충격의 파동 쇠약의 힘 비전 조율 전화 진동 열기의 파도						
준비중						
마력 돌개바람 위력 소모: 비전력 35 순수한 마력을 이용해 소용돌이를 일으킵니다. 소용돌이는 경로에 있는 모든 대상에게 6초에 걸쳐 무기 공격력의 1525%만큼 비전 피해를 줍니다.						
하니바람 질풍 광풍 사나운 바람 폭풍 주최자						
준비중						
히드라 [2] 위력 소모: 비전력 15 머리가 여럿 달린 히드라는 소환합니다. 히드라는 15초 동안 유지되며, 적에게 불덩이를 쏘아 공격할 때마다 무기 공격력의 165% 만큼 화염 피해를 줍니다. 히드라는 동시에 하나만 소환해둘 수 있습니다.						
비전 히드라 번개 히드라 불길 히드라 서리 히드라 거대 히드라						
준비중						
운석 낙하 위력 소모: 비전력 40 하늘에서 거대한 운석을 떨어뜨립니다. 충돌 지점에 있는 모든 적에게 무기 공격력의 740%만큼 화염 피해를 줍니다. 운석이 떨어진 땅은 녹아내리며 3초에 걸쳐 무기 공격력의 235%만큼 화염 피해를 줍니다.						
우렛소리 별의 약속 혜성 운석 소나기 강렬한 충돌						
준비중						
눈보라 [2] 위력 소모: 비전력 40 목표 지역에 얼음 조각을 쏟아부어 반경 12미터 범위에 6초에 걸쳐 공격 범위 내의 모든 적에게 무기 공격력의 1075%만큼 냉기 피해를 줍니다. 동일한 시전자가 같은 위치에 여러 번 주문을 시전해도 피해가 중첩되지는 않습니다.						
번개 폭풍 매얼음 폭설 종말 출기찬 눈보라						
준비중						

GALIP

- **Loss**
 - Adversarial loss
 - w/ GP
- **Image-Text align**
 - Affine transformation
 - No cross-attention
- **Architecture**
 - Traditional GANs
- **Contribution**
 - 224x224, 8x3090 GPUs, 3days (CC-12M)

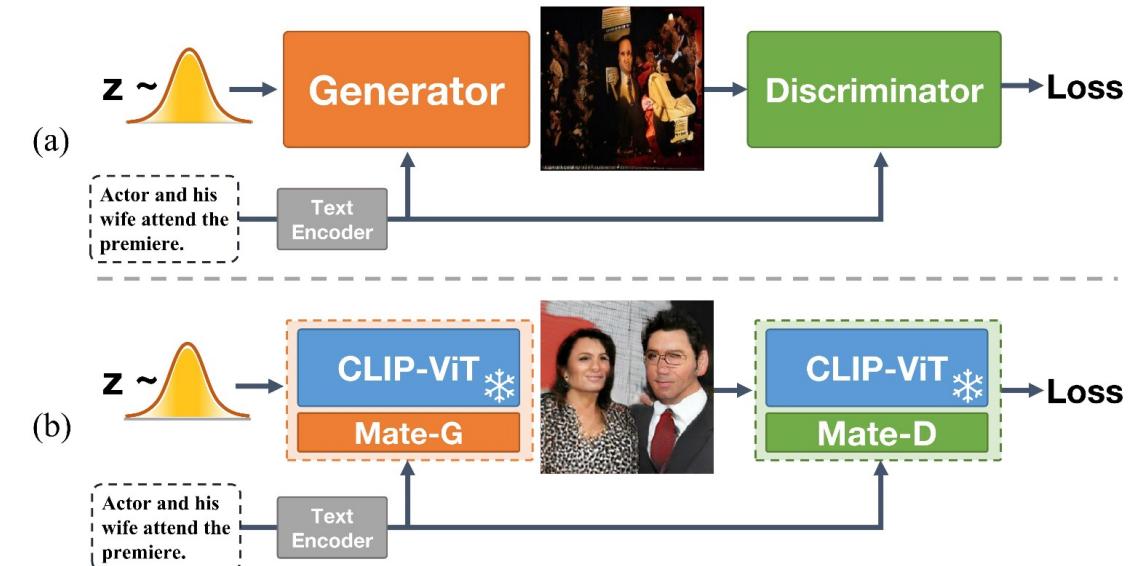
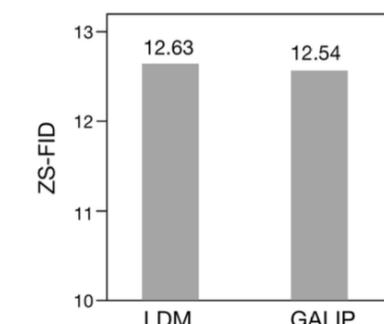
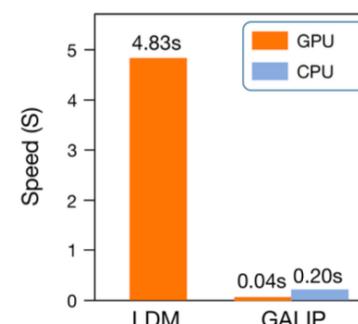
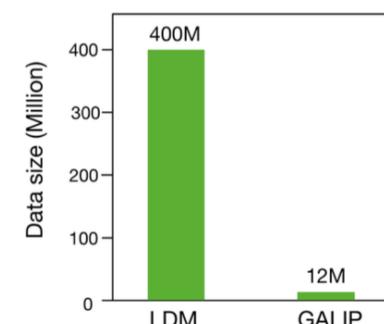
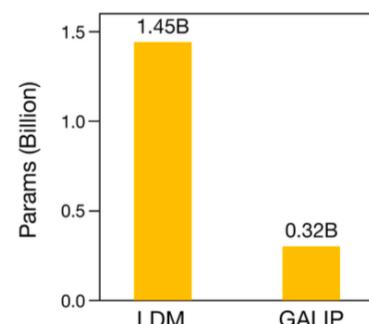


Table 1. The results of FID and CLIPSIM (CS) compared with the state-of-the-art methods on the test set of CUB and COCO.

Model	CUB		COCO	
	FID ↓	CS ↑	FID ↓	CS ↑
DM-GAN [57]	16.09	-	32.64	-
XMC-GAN [53]	-	-	9.30	-
DAE-GAN [37]	15.19	-	28.12	-
DF-GAN [42]	14.81	0.2920	19.32	0.2972
LAFITE [56]	14.58	0.3125	8.21	0.3335
VQ-Diffusion [11]	10.32	-	13.86	-
GALIP (Ours)	10.08	0.3164	5.85	0.3338

Table 2. We compare the performance of large pretrained autoregressive models (AR), diffusion models (DF), and GANs under zero-shot setting on the COCO test dataset.

Model	Type	Param [B]	Data size [M]	ZS-FID ↓
DALL-E [33]	AR	12	250	27.5
Cogview [6]	AR	4	30	27.1
Cogview2 [7]	AR	6	30	24.0
Parti-350M [51]	AR	0.35	>800	14.10
Parti-20B [51]	AR	20	>800	7.23
GLIDE [27]	DF	5	250	12.24
LDM [35]	DF	1.45	400	12.63
DALL-E 2 [32]	DF	6.5	250	10.39
Imagen [38]	DF	7.9	860	7.27
eDiff-I [1]	DF	9.1	1000	6.95
LAFITE [56]	GAN	0.15+0.08	3	26.94
GALIP (CC3M)	GAN	0.24+0.08	3	16.12
GALIP (CC12M)	GAN	0.24+0.08	12	12.54



Figure 7. Text-to-Image samples from GALIP (CC12M) and Latent Diffusion (LAION-400M) [35, 36]. We sample 16 images from each given text description, and randomly select one as the final generation result

A corgi's head depicted as an explosion of a nebula



A lion teacher wearing a suit is in front of a blackboard.



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



A robot is riding a horse under the blue and cloudy sky.



Figure 9. Illustration of failure cases. It is still hard for current GALIP to synthesize some imaginary images. Enlarging the model size and training data may improve image quality.

Evaluation

Univariate Normal Fréchet Distance =

$$(\mu_X - \mu_Y)^2 + (\sigma_X - \sigma_Y)^2$$

Multivariate Normal Fréchet Distance =

$$\|\mu_X - \mu_Y\|^2 + \text{Tr} \left(\Sigma_X + \Sigma_Y - 2\sqrt{\Sigma_X \Sigma_Y} \right)$$

공분산 행렬의 대각성분은 분산 !

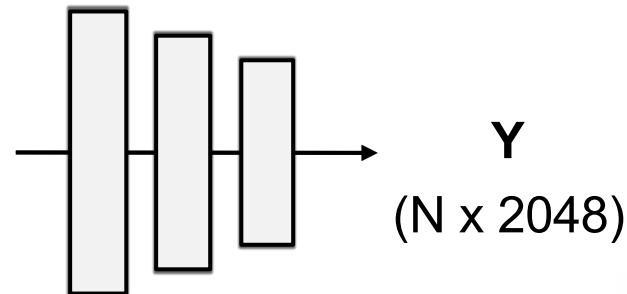
Real samples



Fake samples



Inception v3

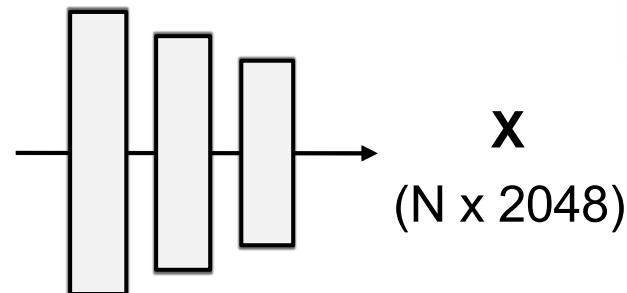


FID =

$$\|\mu_X - \mu_Y\|^2 + \text{Tr} \left(\Sigma_X + \Sigma_Y - 2\sqrt{\Sigma_X \Sigma_Y} \right)$$

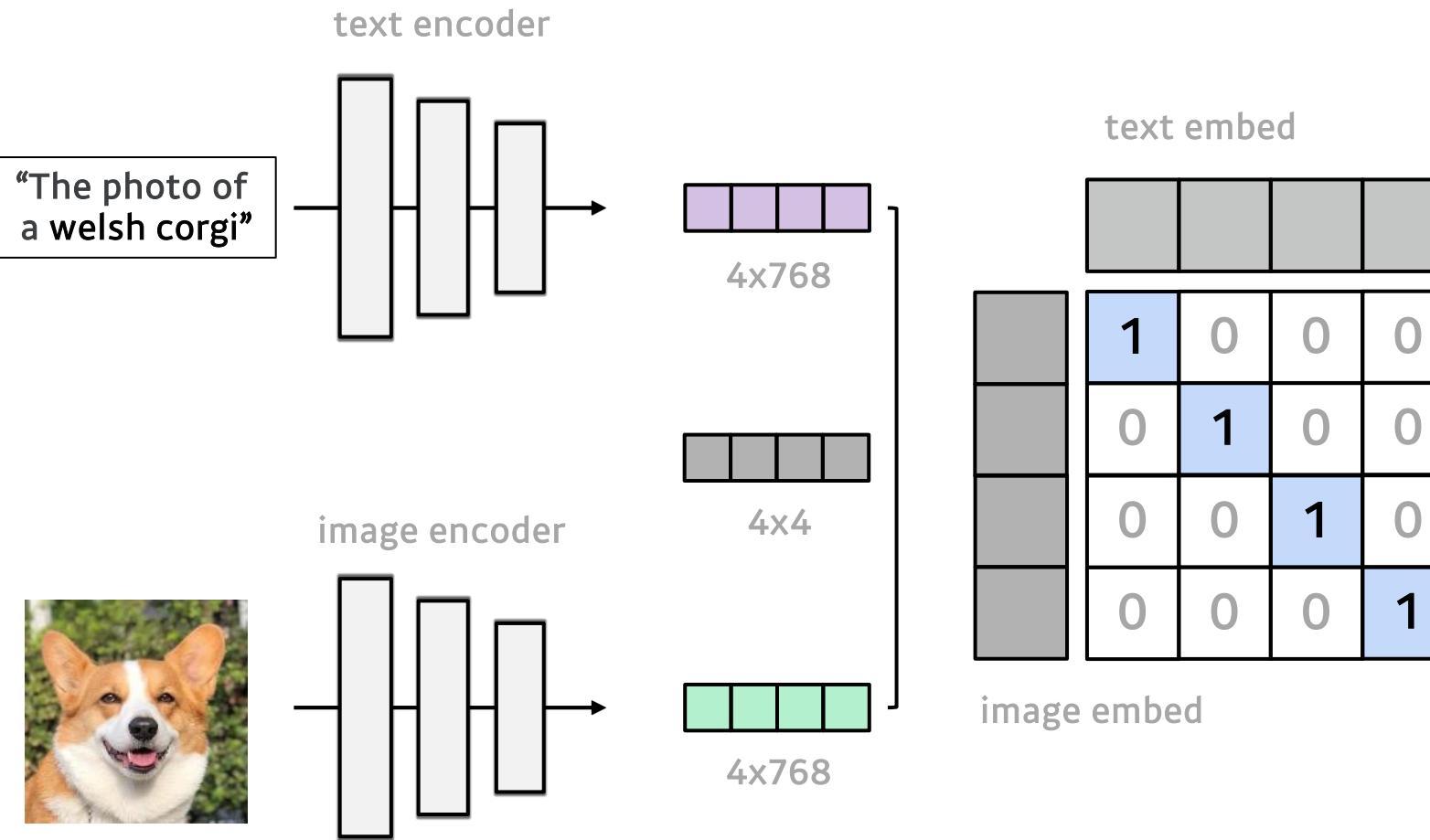
Real and fake embeddings are two
multivariate normal distributions

Inception v3

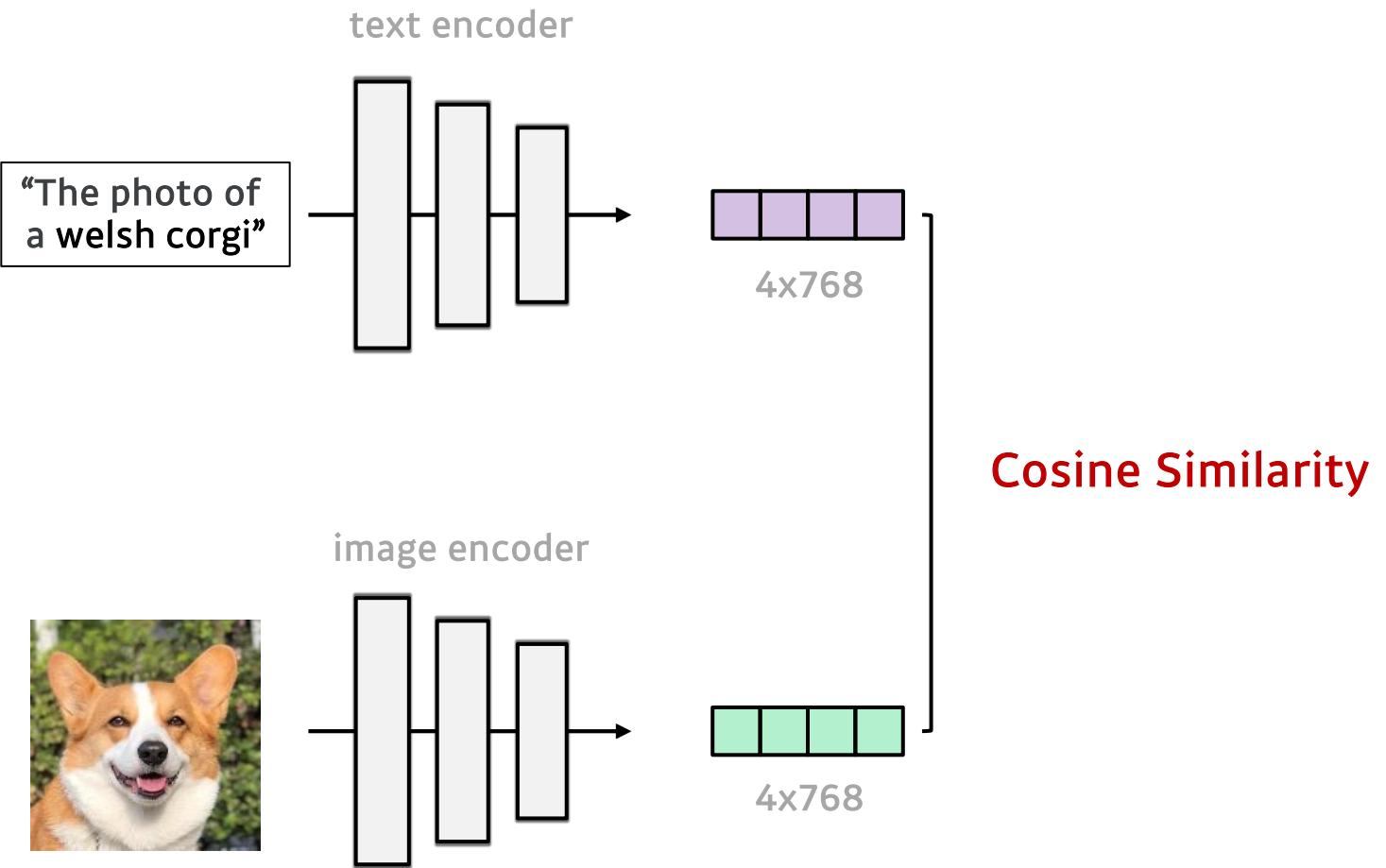


Evaluation

Clip Score



Contrastive Loss



* 1: 관계가 깊다.

* 0: 관계가 없다.

* -1: 관계가 반대다.



 openai/clip-vit-base-patch16
Zero-Shot Image Classification • Updated Oct 4, 2022 • 809k • 31

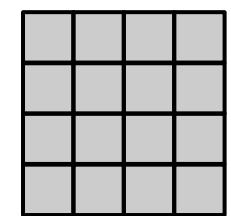
 openai/clip-vit-base-patch32
Zero-Shot Image Classification • Updated Oct 4, 2022 • 4.18M • 204

 openai/clip-vit-large-patch14-336
Zero-Shot Image Classification • Updated Oct 4, 2022 • 381k • 39

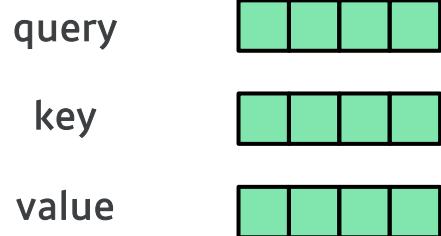
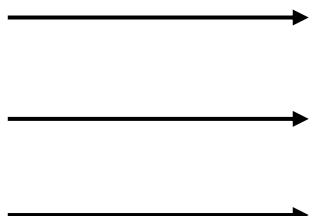
 openai/clip-vit-large-patch14
Zero-Shot Image Classification • Updated Oct 4, 2022 • 15.7M • 490

Editing with SD

Editing method | Attention control



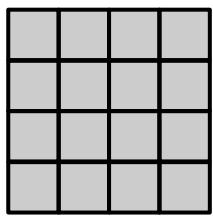
feature map



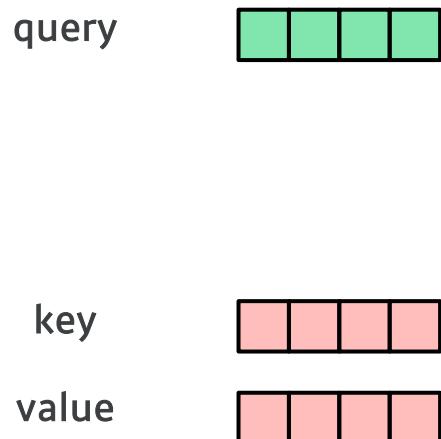
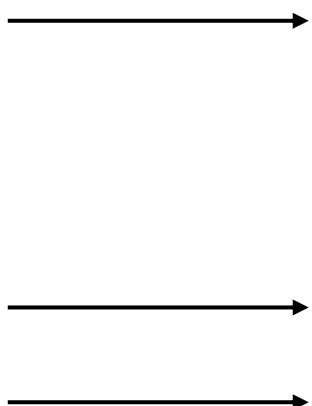
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$Q = (N, d_k), K = (N, d_k), V = (N, d_v)$$

[N, seq_len, dim]



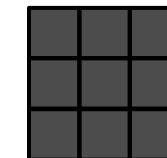
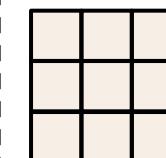
feature map



관계성
QK

가중치
QK * V
특정 단어 강조 가능

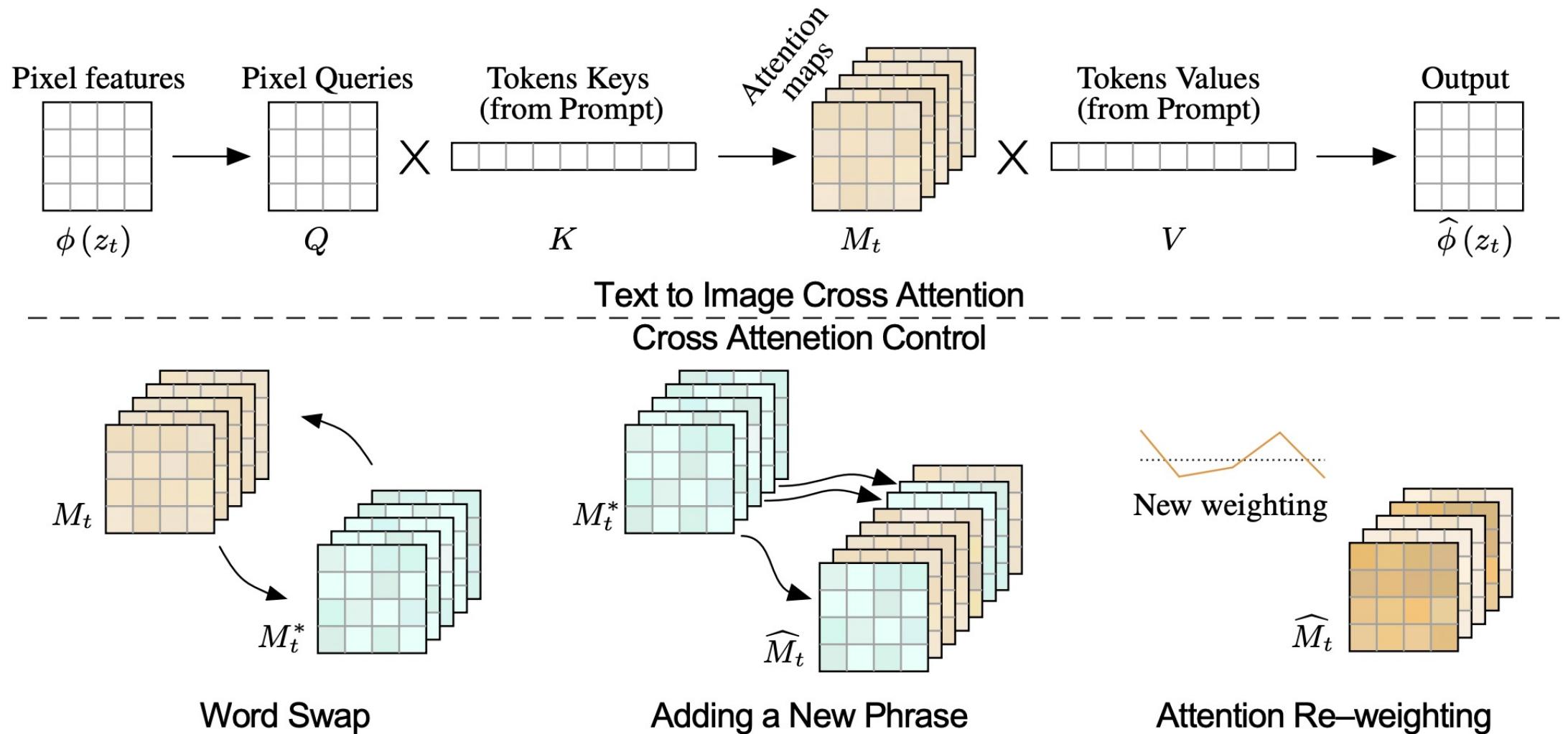
mask matrix 활용 가능

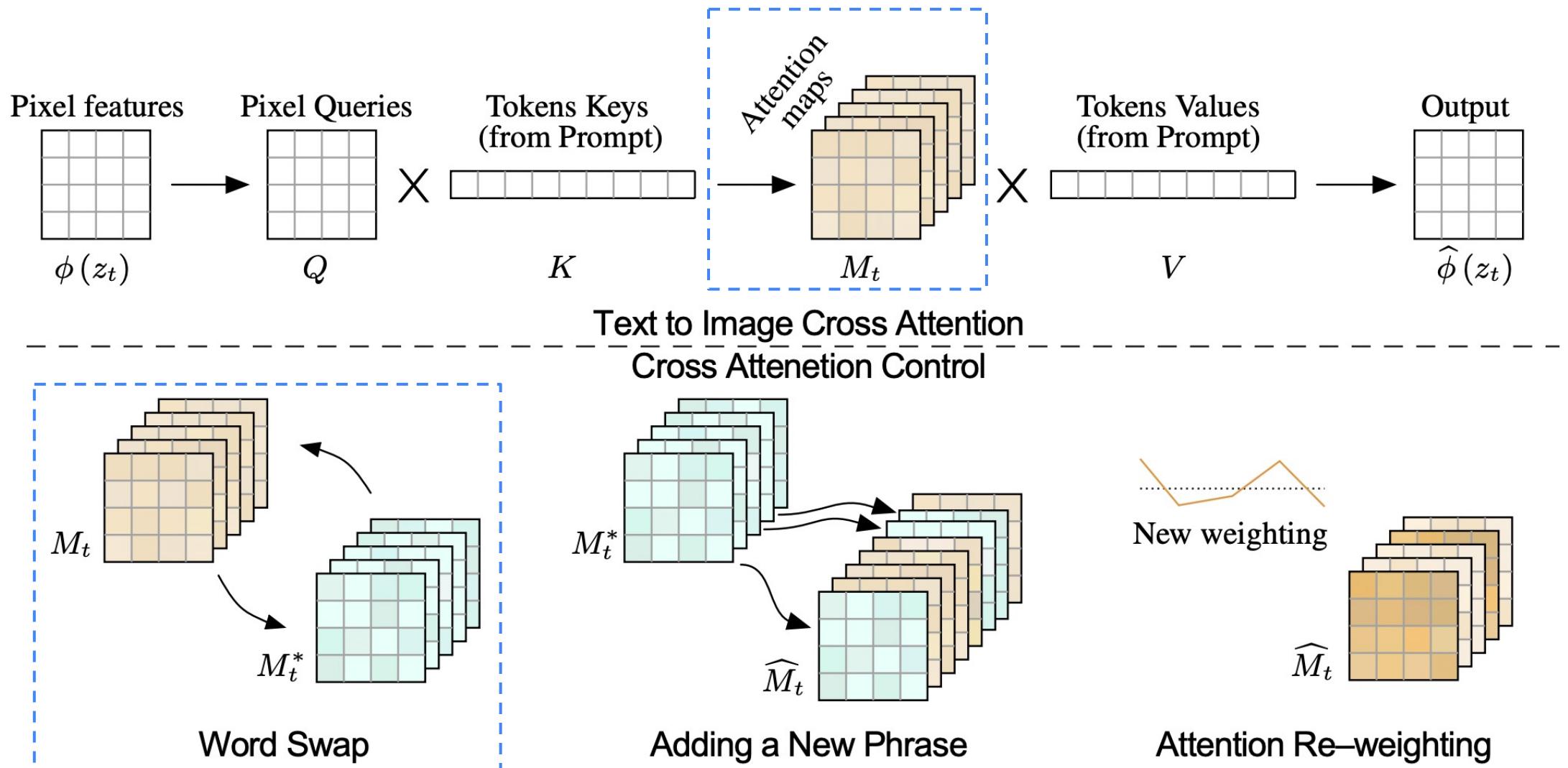


QK

mask

원하는 단어 무시 가능

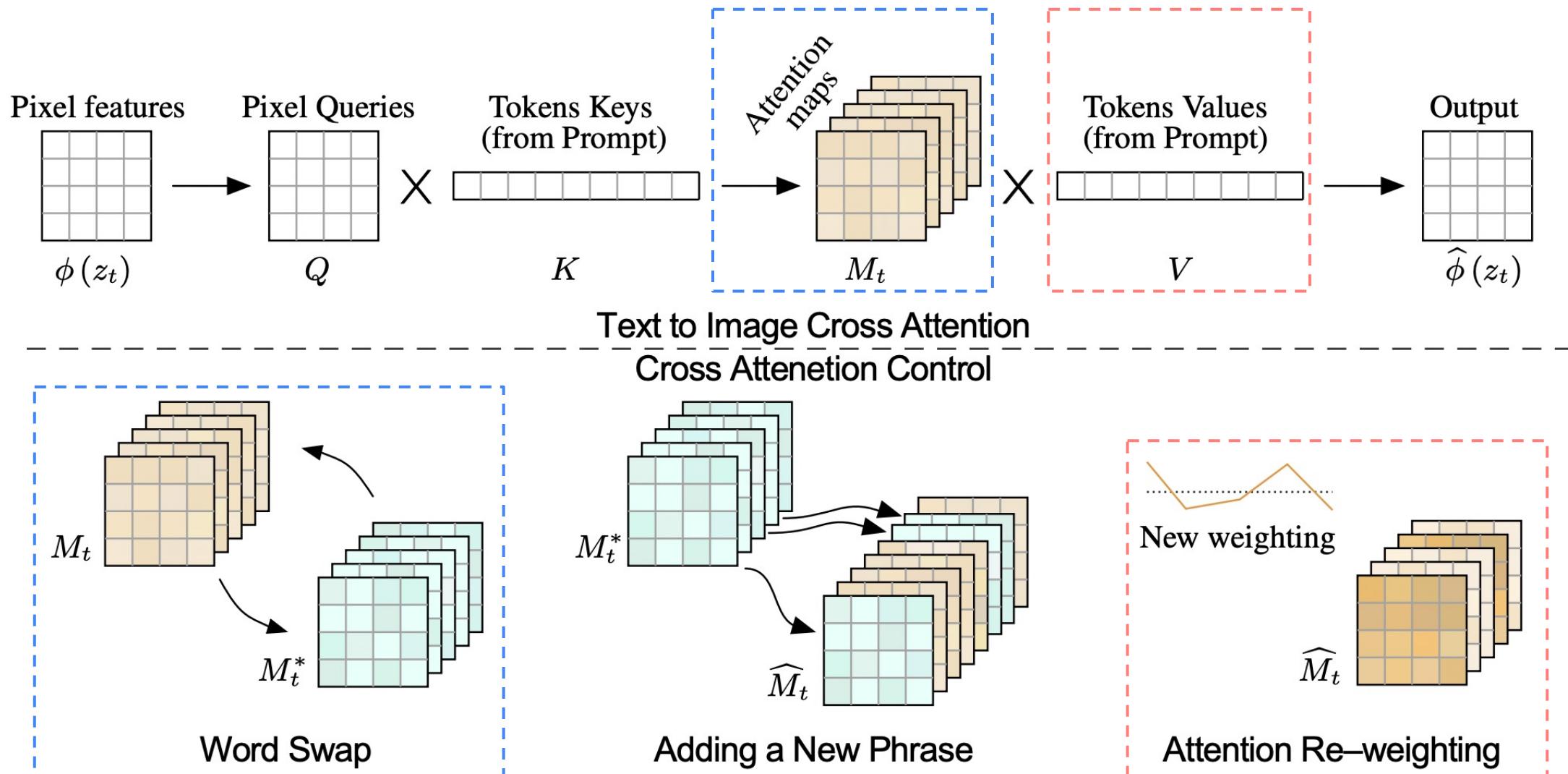




Editing method

Attention control

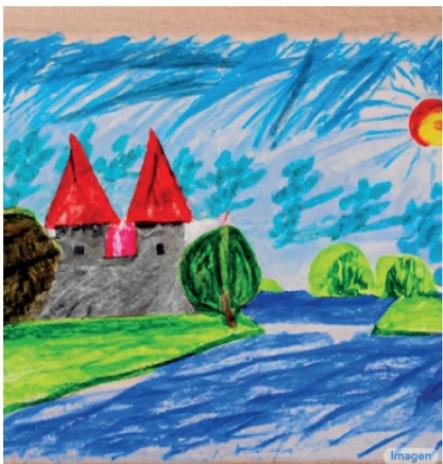
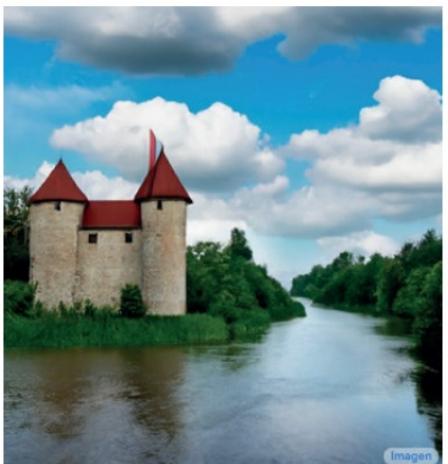
NAVER AI LAB



Prompt-to-Prompt | Results



“Photo of a cat riding on a ~~bicycle~~
car.”



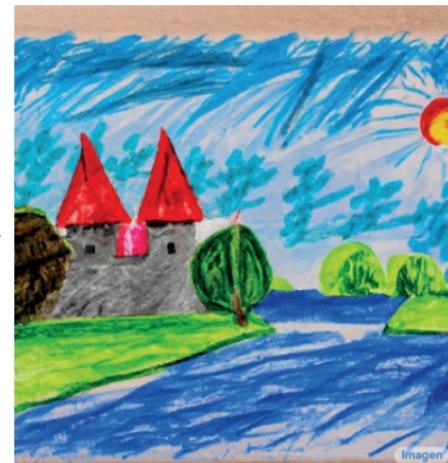
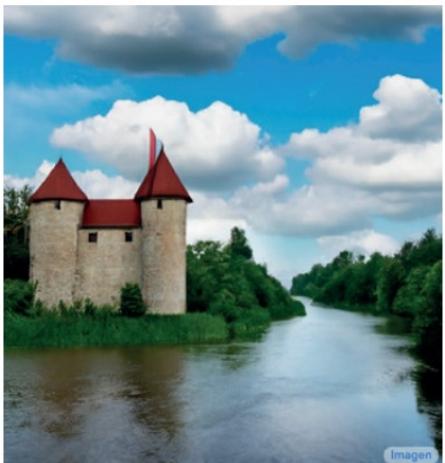
“*Children drawing of* a castle next to a river.”

Prompt-to-Prompt

Results



“Photo of a cat riding on a ~~bicycle~~^{car}.”



“Children drawing of a castle next to a river.”



“A photo of a birthday(~~v~~) cake next to an apple.”



“The picnic is ready under a blossom(~~v~~) tree.”



“A photo of a house on a snowy(~~↑~~) mountain.”



“My fluffy(~~↑~~) bunny doll.”

Editing method

Dreambooth



Input images



in the Acropolis



swimming



sleeping



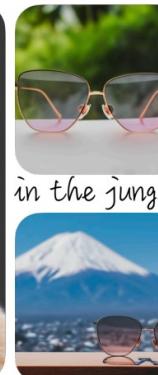
getting a haircut



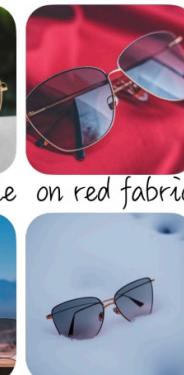
Input images



worn by a bear



in the jungle
at Mt. Fuji



on red fabric
on top of snow



with Eiffel Tower

Editing method

Dreambooth



“A [V] [class]”

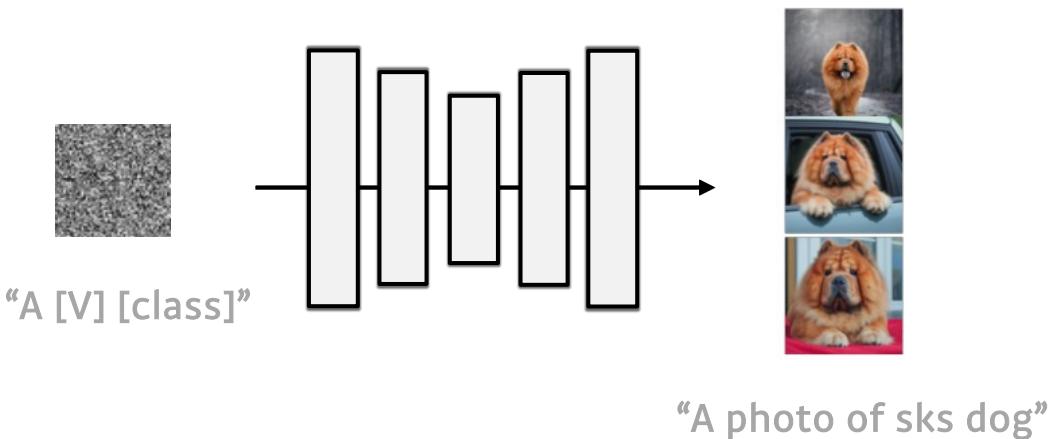


Input image

Editing method

Dreambooth

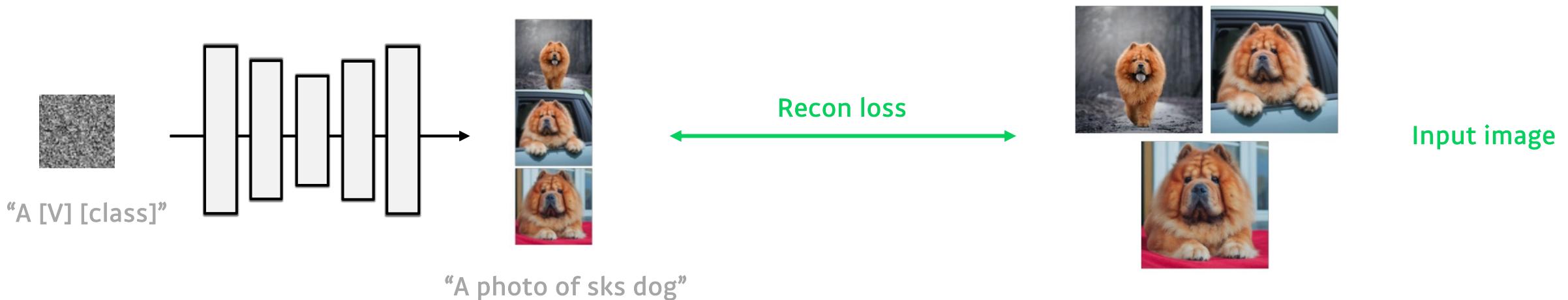
 NAVER AI LAB



Input image

Editing method

Dreambooth



Editing method

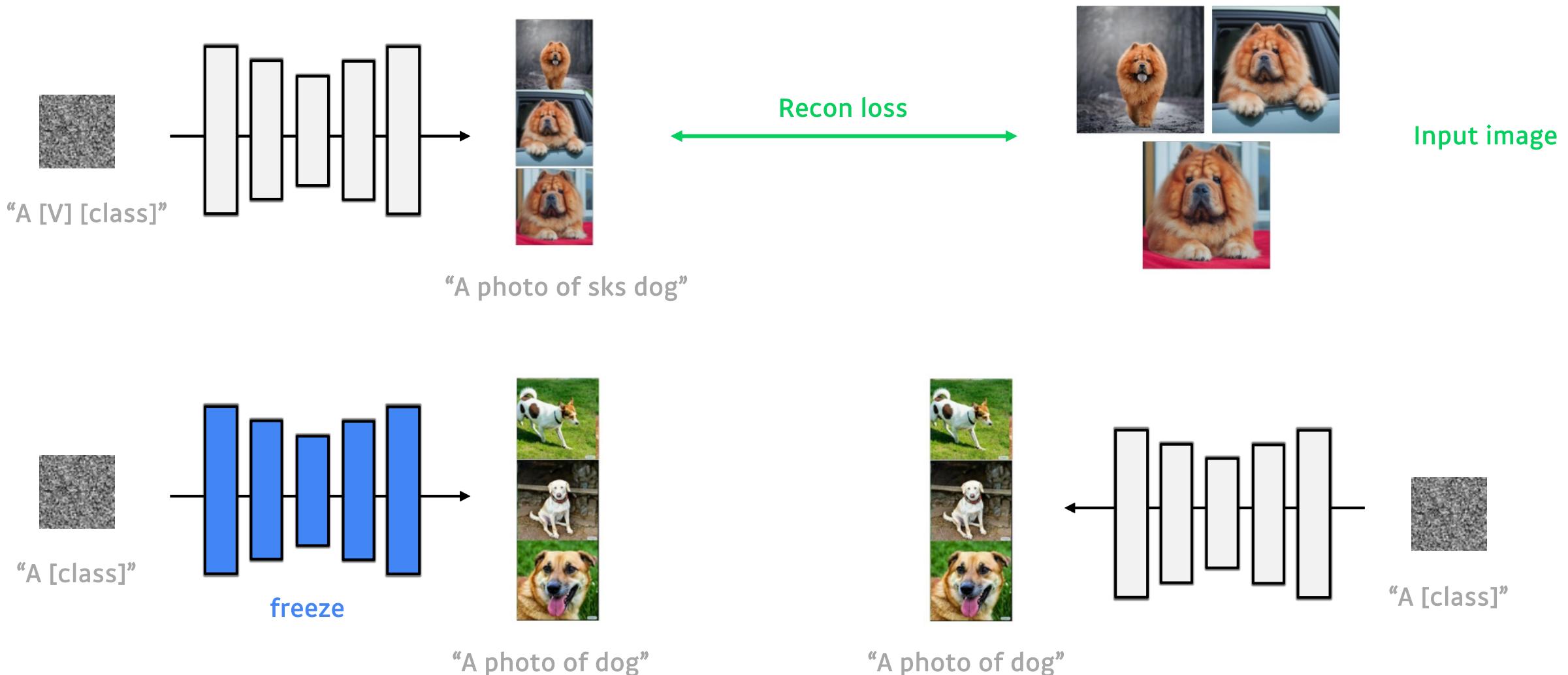
Dreambooth



Editing method

Dreambooth

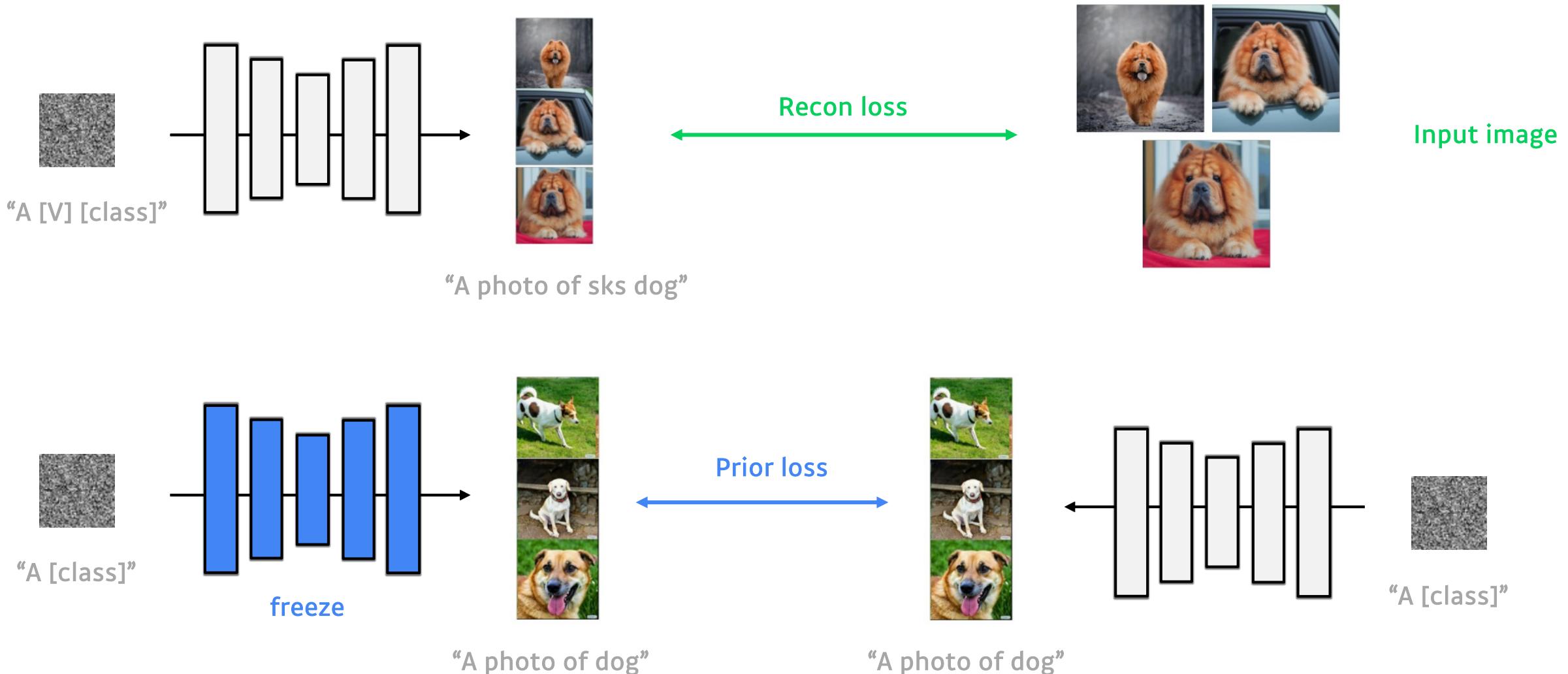
 NAVER AI LAB



Editing method

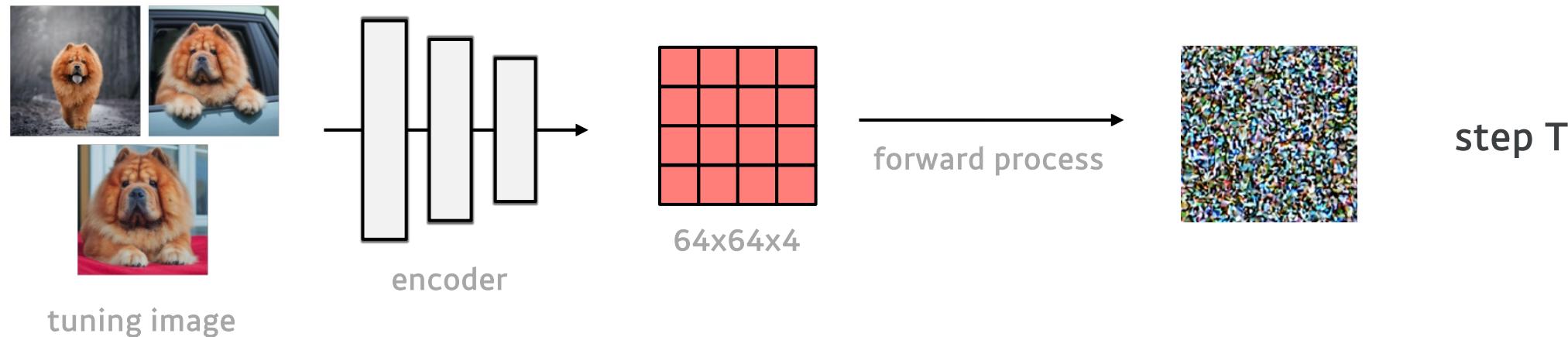
Dreambooth

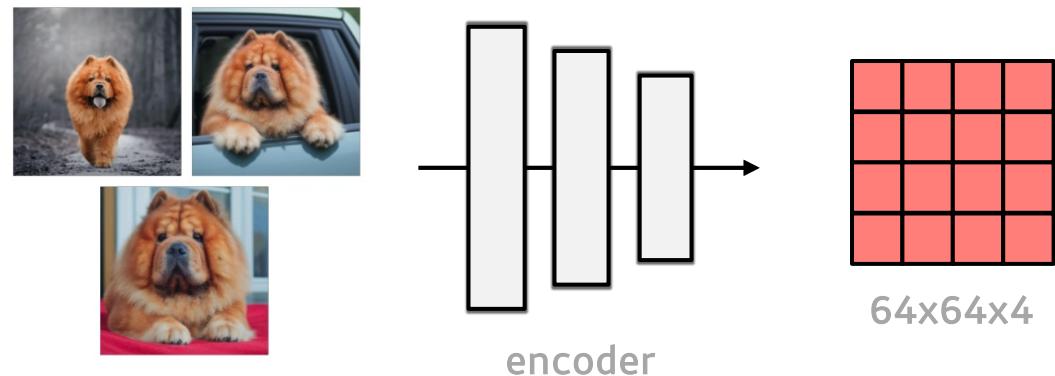
 NAVER AI LAB





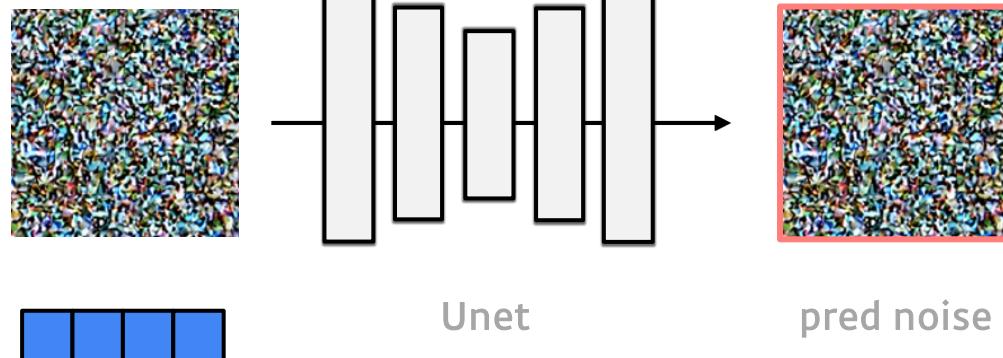
tuning image





tuning image

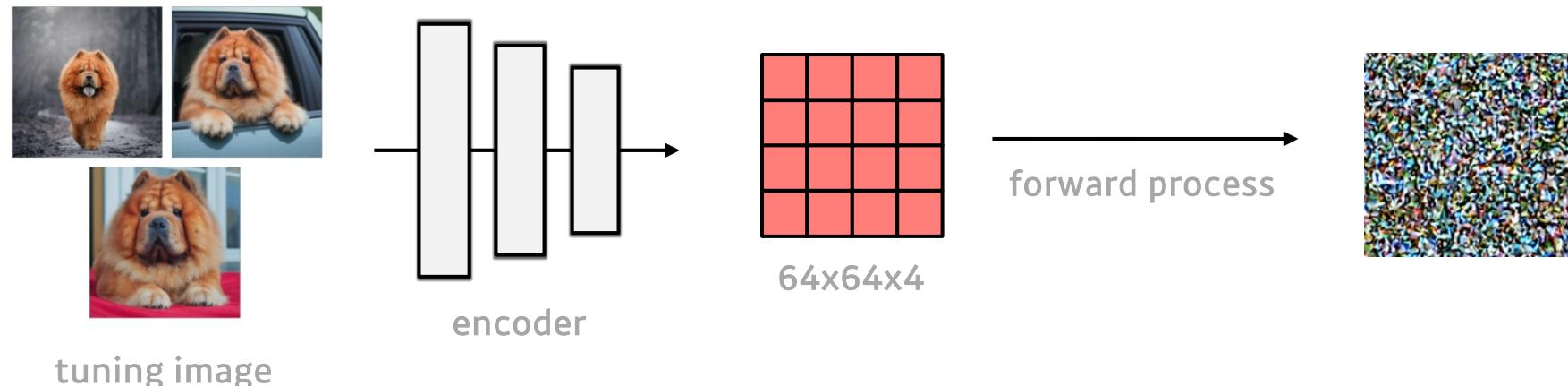
time embed (T)



Unet

pred noise

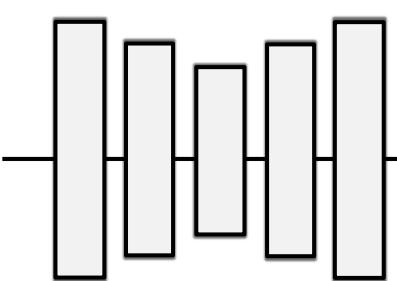
"A photo of sks dog"



time embed (T)



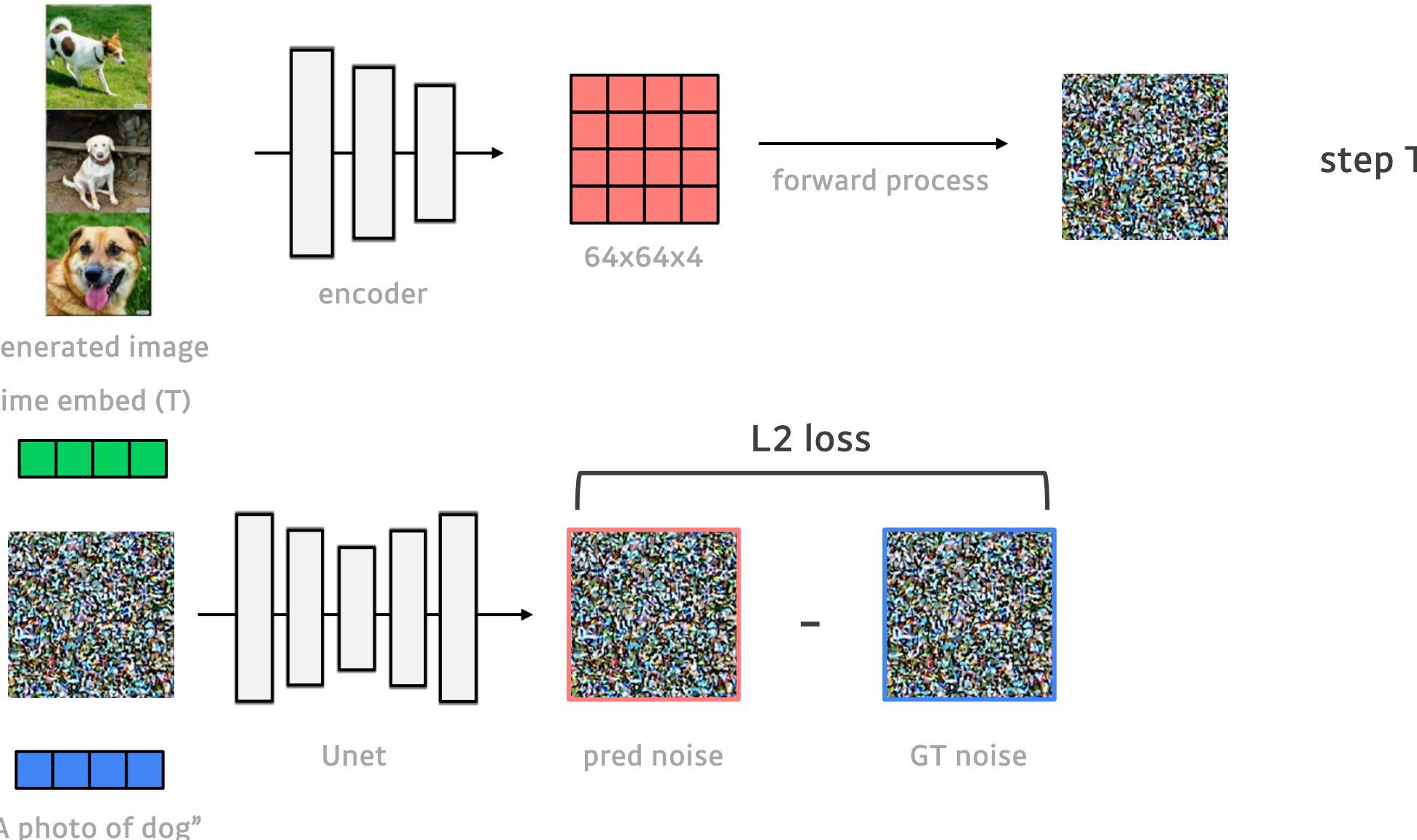
L2 loss



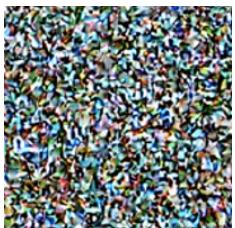
pred noise

GT noise

“A photo of sks dog”

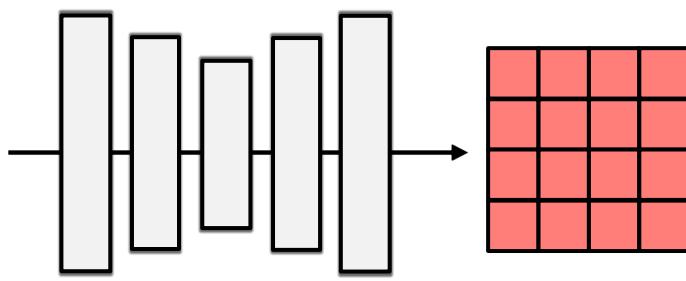


time embed (T)



“A sks dog wearing a [?]”

time embed (T)

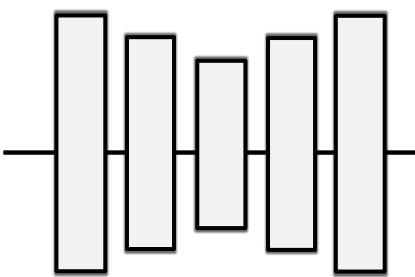


64x64x4

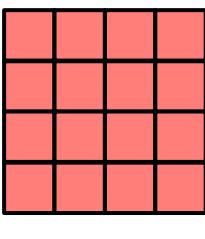


“A sks dog wearing a [?]”

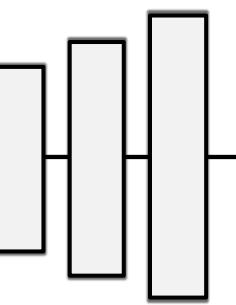
time embed (T)



Unet



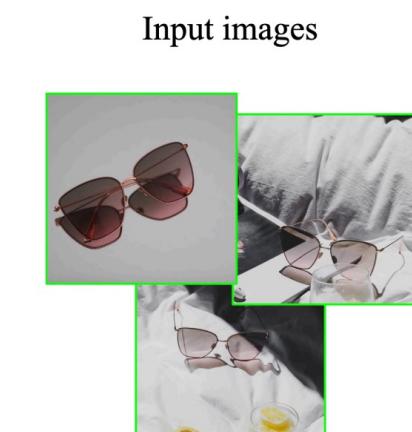
64x64x4



decoder



“A sks dog wearing a [?]”



Fine-tuning

No class noun:
“*A [V]*”



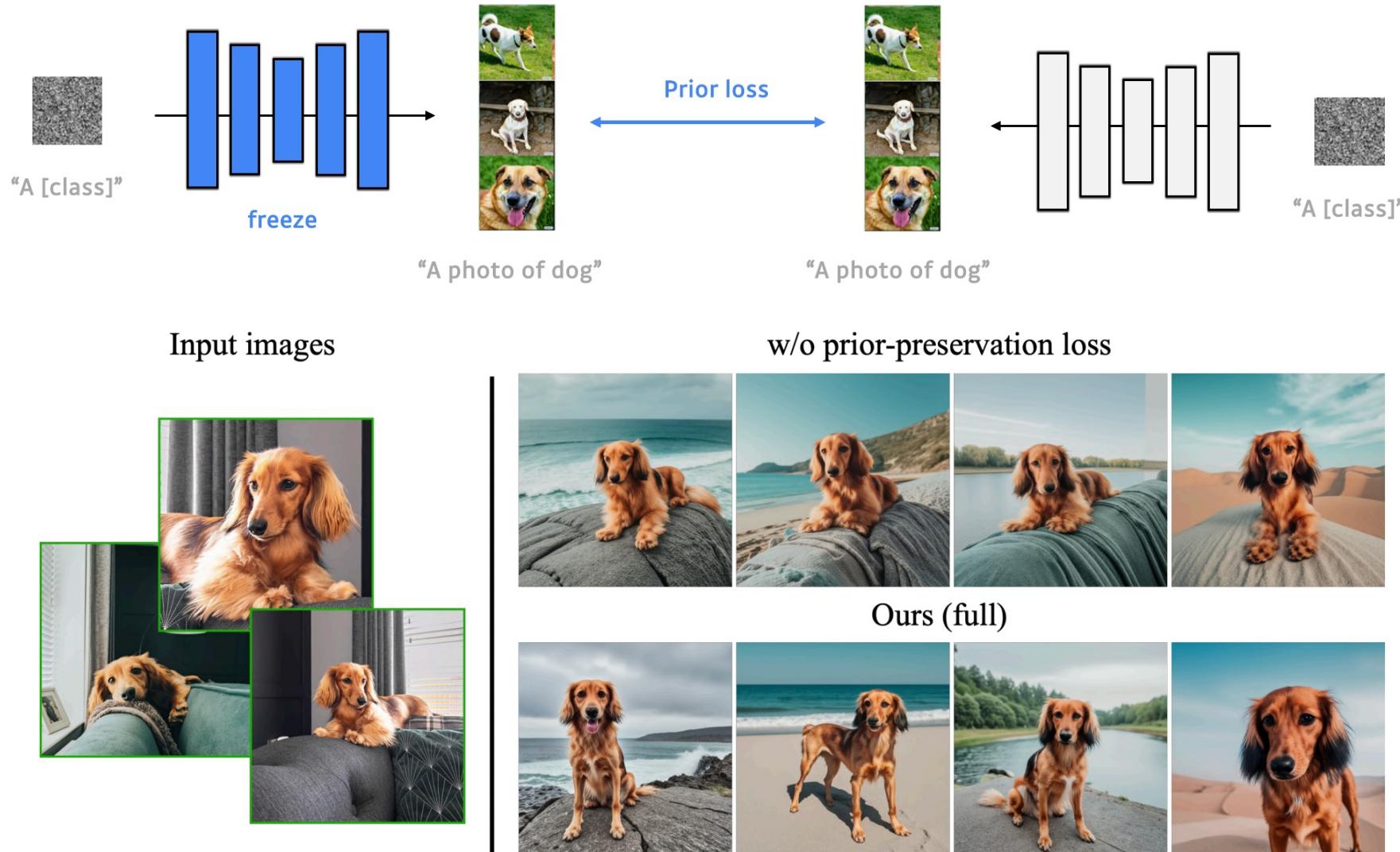
Incorrect class noun:
“*A [V] dog*”



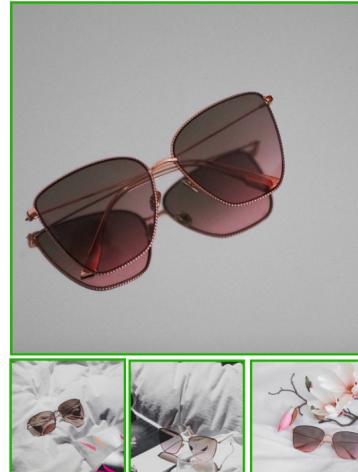
Correct class noun:
“*A [V] sunglasses*”



Inference



Input images



A [V] sunglasses at Mt. Fuji

A [V] sunglasses on top of snow

A [V] sunglasses with Eiffel Tower in the background

Input images



Johannes Vermeer

Pierre-Auguste Renoir

Leonardo da Vinci

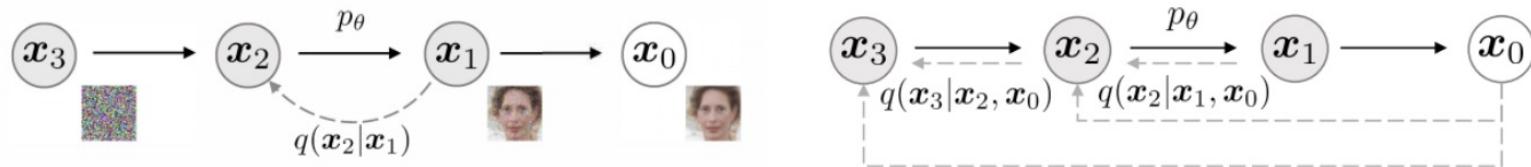


Figure 1: Graphical models for diffusion (left) and non-Markovian (right) inference models.

최근 Trend

DDPM

- $\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, \epsilon \sim \mathcal{N}(0, \mathbf{I})$ (Forward)
 - $\beta_1 = 10^{-4}, \beta_T = 0.02$
 - $\alpha_t := 1 - \beta_t, \bar{\alpha}_t := \prod_{s=1}^t \alpha_s$
- $\epsilon - \epsilon_\theta(\mathbf{x}_t) = \epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon)$ (Loss)
 - ϵ_θ = prediction network
- $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(\mathbf{x}_t, t) \right) + \tilde{\beta}_t \epsilon, \epsilon \sim \mathcal{N}(0, \mathbf{I})$ (Reverse)
 - $\tilde{\beta}_t = \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} \beta_t$
 - $\tilde{\beta}_t = \beta_t$ 로 해도 성능차이 없음

Denoising Diffusion Implicit Models

DDIM

- In paper, DDIM α = DDPM $\bar{\alpha}$
- $\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon$ (Forward)
- $\epsilon - \epsilon_\theta(\mathbf{x}_t) = \epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon)$ (Loss)
 - ϵ_θ = prediction network
- $$\mathbf{x}_{t-1} = \sqrt{\bar{\alpha}_{t-1}} \left(\underbrace{\frac{\mathbf{x}_t - \sqrt{1 - \bar{\alpha}_t} \epsilon_\theta(\mathbf{x}_t)}{\sqrt{\bar{\alpha}_t}}}_{\text{predicted } \mathbf{x}_0 = f_\theta(\mathbf{x}_t)} \right) + \underbrace{\sqrt{1 - \bar{\alpha}_{t-1} - \sigma_t^2} \cdot \epsilon_\theta(\mathbf{x}_t)}_{\text{direction pointing to } \mathbf{x}_t} + \underbrace{\sigma_t \epsilon}_{\text{noise}}$$
 (Reverse)
 - deterministic when $\sigma_t = 0 \rightarrow$ consistency (DDIM)
 - stochastic when $\sigma_t = 1 \rightarrow$ inconsistency (DDPM)

$$x_{t-1} = \hat{\mu}_t(x_t) + \sigma_t \epsilon$$

$$x_{t-1} = \sqrt{\bar{\alpha}_{t-1}} P(f_t(x_t)) + D(f_t(x_t)) + \sigma_t \epsilon$$

$$* P(f_t(x_t)) = \frac{x_t - \sqrt{1 - \bar{\alpha}_t} f_t(x_t)}{\sqrt{\bar{\alpha}_t}}$$

$$* D(f_t(x_t)) = \sqrt{1 - \bar{\alpha}_{t-1} - \sigma_t^2} f_t(x_t)$$

Editing method

DDPM Inversion

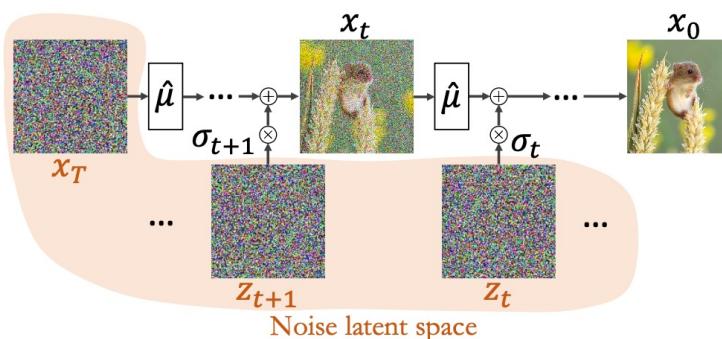
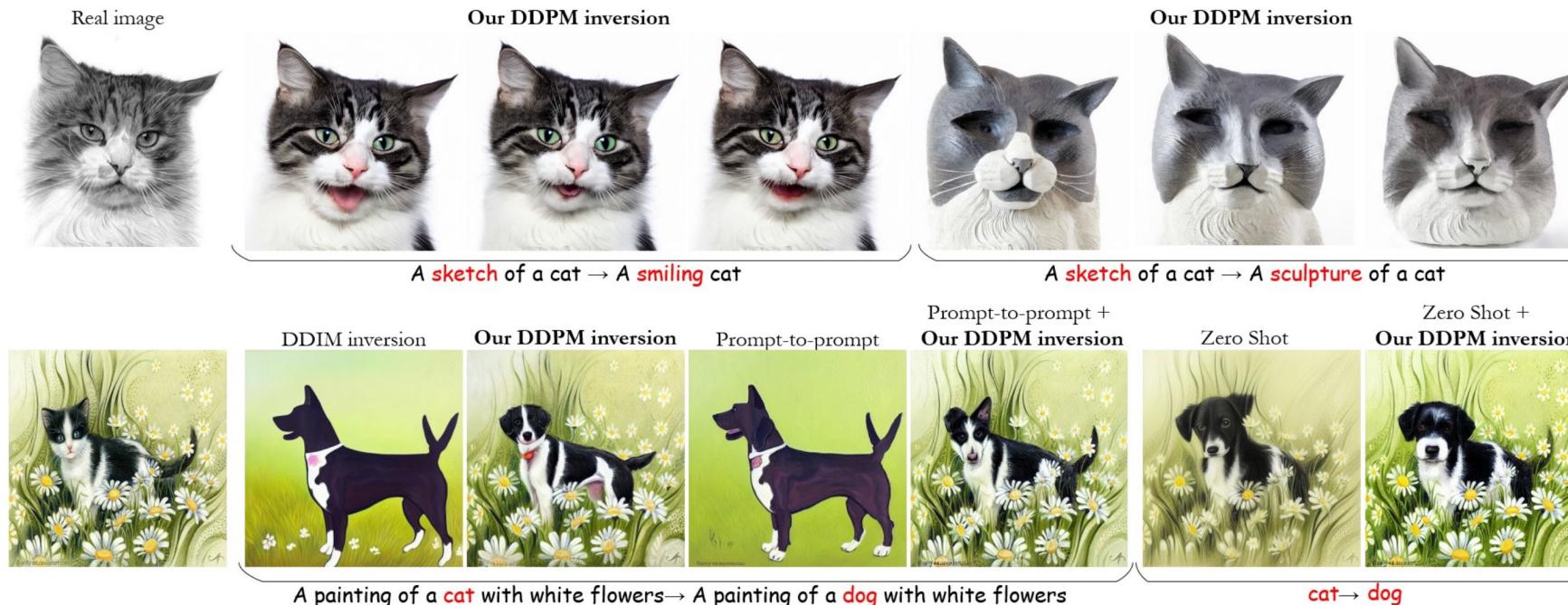
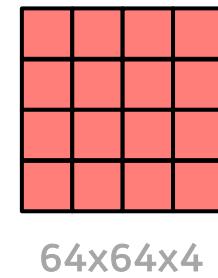
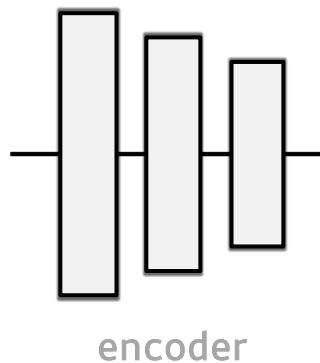


Figure 3: **The DDPM latent noise space.** In DDPM, the generative (reverse) diffusion process synthesizes an image x_0 in T steps, by utilizing $T + 1$ noise maps, $\{x_T, z_T, \dots, z_1\}$. We regard those noise maps as the latent code associated with the generated image.

Editing method

DDPM Inversion



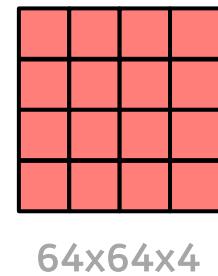
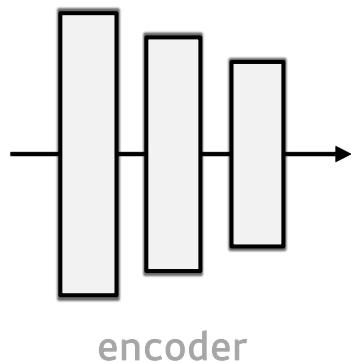
$$x_t = \bar{\alpha}_t x_0 + (1 - \bar{\alpha}_t) \varepsilon_t$$

forward process



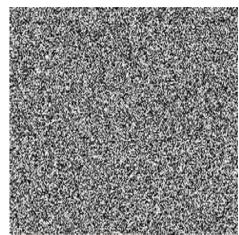
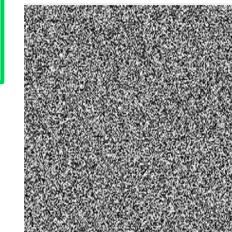
Editing method

DDPM Inversion



$$x_t = \bar{\alpha}_t x_0 + (1 - \bar{\alpha}_t) \varepsilon_t$$

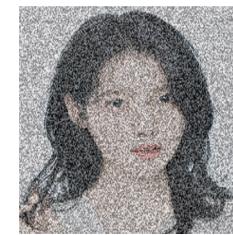
forward process



x_T



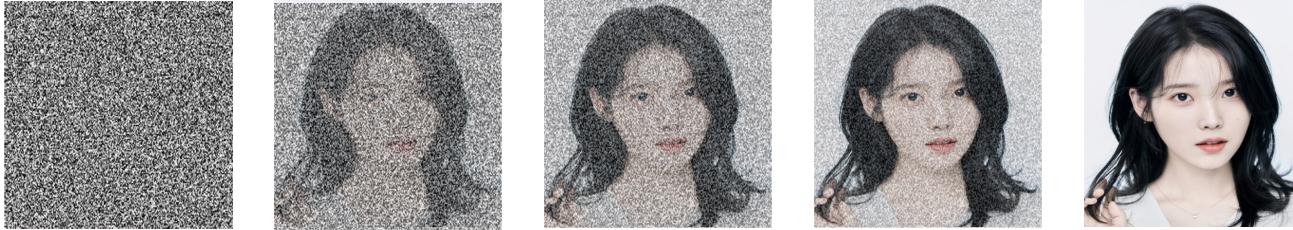
x_t



x_0

Editing method

DDPM Inversion



x_T

x_t

x_0

Reverse process

$$x_{t-1} = \hat{\mu}_t(x_t) + \sigma_t z_t$$

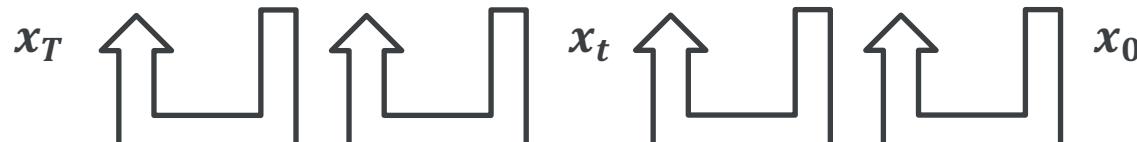
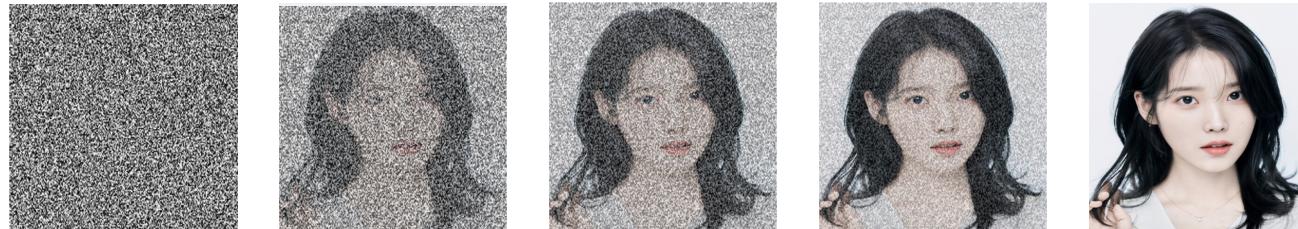
$$\hat{\mu}_t(x_t) = \sqrt{\bar{\alpha}_{t-1}} P(f_t(x_t)) + D(f_t(x_t))$$

$$P(f_t(x_t)) = \frac{x_t - \sqrt{1 - \bar{\alpha}_t} f_t(x_t)}{\sqrt{\bar{\alpha}_t}}$$

$$D(f_t(x_t)) = \sqrt{1 - \bar{\alpha}_{t-1} - \sigma_t^2} f_t(x_t)$$

Editing method

DDPM Inversion



\mathbf{z}_T

\mathbf{z}_t

\mathbf{z}_1

$$z_t \leftarrow \frac{x_{t-1} - \hat{\mu}_t(x_t)}{\sigma_t}$$

Reverse process

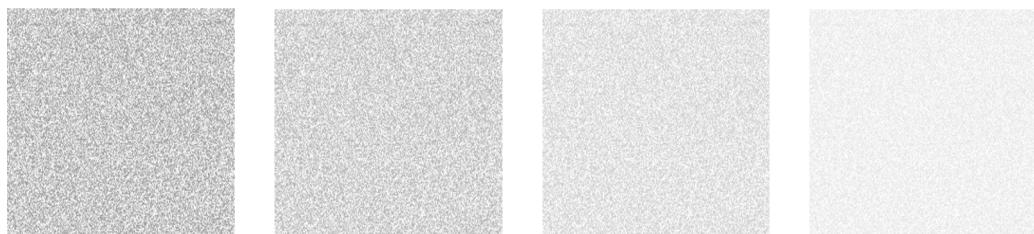
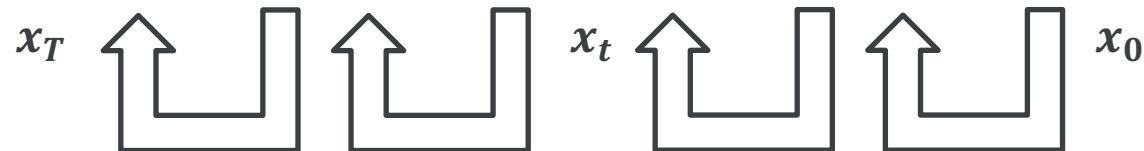
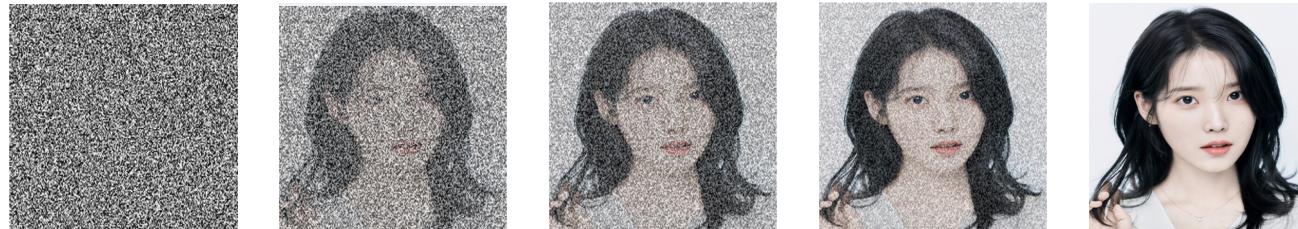
$$x_{t-1} = \hat{\mu}_t(x_t) + \sigma_t z_t$$

$$\hat{\mu}_t(x_t) = \sqrt{\bar{\alpha}_{t-1}} P(f_t(x_t)) + D(f_t(x_t))$$

$$P(f_t(x_t)) = \frac{x_t - \sqrt{1 - \bar{\alpha}_t} f_t(x_t)}{\sqrt{\bar{\alpha}_t}}$$

$$D(f_t(x_t)) = \sqrt{1 - \bar{\alpha}_{t-1} - \sigma_t^2} f_t(x_t)$$

Editing method



$$z_t \leftarrow \frac{x_{t-1} - \hat{\mu}_t(x_t)}{\sigma_t}$$

Reverse process

$$x_{t-1} = \hat{\mu}_t(x_t) + \sigma_t z_t$$

$$\hat{\mu}_t(x_t) = \sqrt{\bar{\alpha}_{t-1}} P(f_t(x_t)) + D(f_t(x_t))$$

$$P(f_t(x_t)) = \frac{x_t - \sqrt{1 - \bar{\alpha}_t} f_t(x_t)}{\sqrt{\bar{\alpha}_t}}$$

$$D(f_t(x_t)) = \sqrt{1 - \bar{\alpha}_{t-1} - \sigma_t^2} f_t(x_t)$$



x_T

x_t

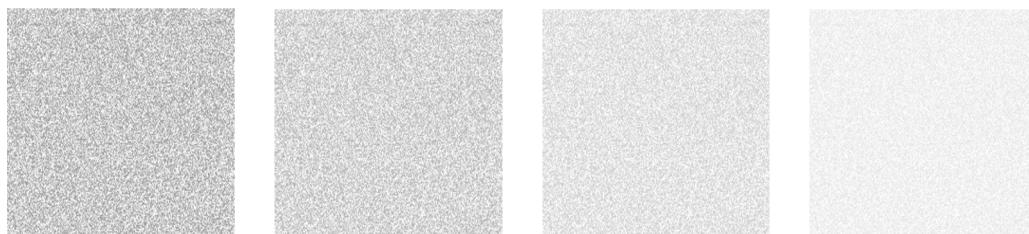
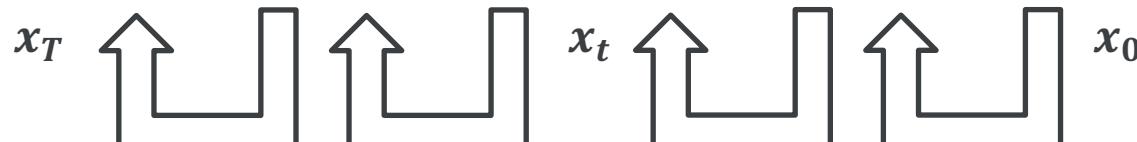
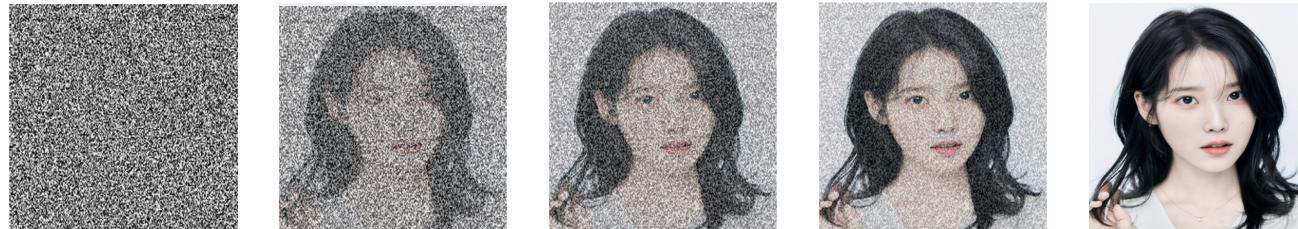
x_0

$$x_{t-1} \leftarrow \hat{\mu}_t(x_t) + \sigma_t z_t$$

x_t 는 x_0 의 semantic, structure를 잘 담아내게 되는다.

Editing method

DDPM Inversion



$$z_t \leftarrow \frac{x_{t-1} - \hat{\mu}_t(x_t)}{\sigma_t}$$

Reverse process

$$x_{t-1} = \hat{\mu}_t(x_t) + \sigma_t z_t$$

$\hat{\mu}_t(x_t) = \sqrt{\bar{\alpha}_{t-1}} P(f_t(x_t)) + D(f_t(x_t))$

$P(f_t(x_t)) = \frac{x_t - \sqrt{1 - \bar{\alpha}_t} f_t(x_t)}{\sqrt{\bar{\alpha}_t}}$

$D(f_t(x_t)) = \sqrt{1 - \bar{\alpha}_{t-1} - \sigma_t^2} f_t(x_t)$



x_T

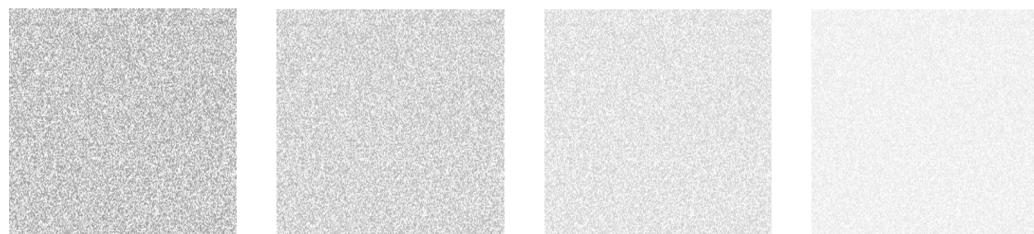
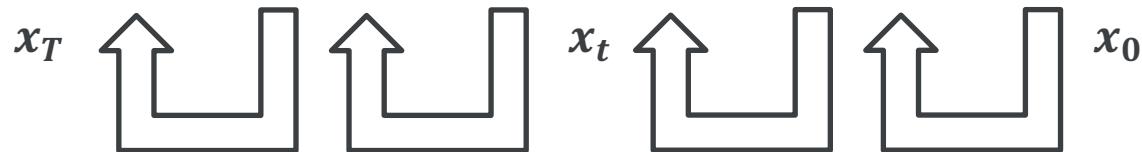
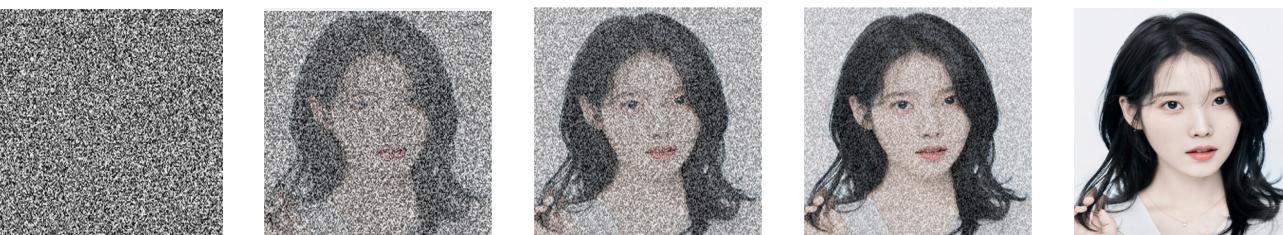
x_t

x_0

$$x_{t-1} \leftarrow \hat{\mu}_t(x_t) + \sigma_t z_t$$

x_t 는 x_0 의 semantic, structure를 잘 담아내게 되다.

Editing method



z_T

z_t

z_1

$$z_t \leftarrow \frac{x_{t-1} - \hat{\mu}_t(x_t)}{\sigma_t}$$

Reverse process
 $x_{t-1} = \hat{\mu}_t(x_t) + \sigma_t z_t$

$$\hat{\mu}_t(x_t) = \sqrt{\bar{\alpha}_{t-1}} P(f_t(x_t)) + D(f_t(x_t))$$

$$P(f_t(x_t)) = \frac{x_t - \sqrt{1 - \bar{\alpha}_t} f_t(x_t)}{\sqrt{\bar{\alpha}_t}}$$

$$D(f_t(x_t)) = \sqrt{1 - \bar{\alpha}_{t-1} - \sigma_t^2} f_t(x_t)$$

$$x_{t+1} = \sqrt{\bar{\alpha}_{t-1}} \left(\frac{x_t - \sqrt{1 - \bar{\alpha}_{t-1}} * f(x_t)}{\sqrt{\bar{\alpha}_{t-1}}} \right) + \sqrt{1 - \bar{\alpha}_t} * f(x_t)$$

이건 ddim inversion

$$x_{t-1} \leftarrow \hat{\mu}_t(x_t) + \sigma_t z_t$$



x_T

x_t

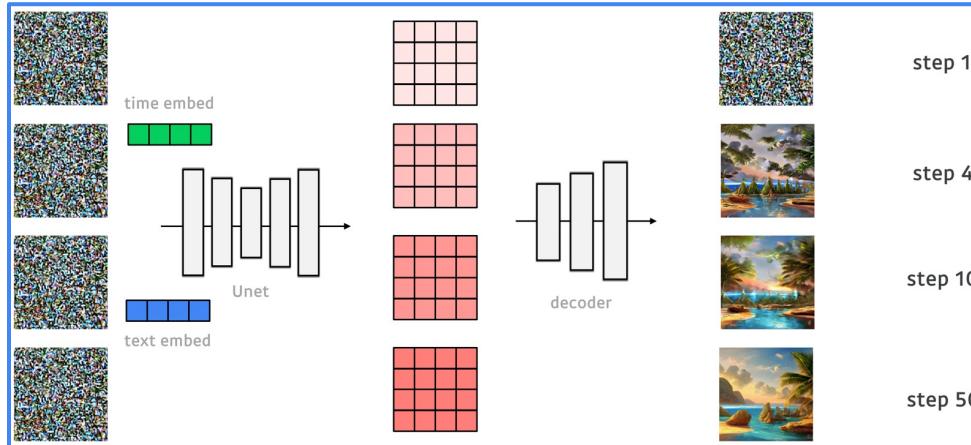
x_0

x_t 는 x_0 의 semantic, structure를 잘 담아내게 되는다.

Summary



Summary



Stable diffusion, GALIP

Trending T2I

P2P, Dreambooth, Inversion

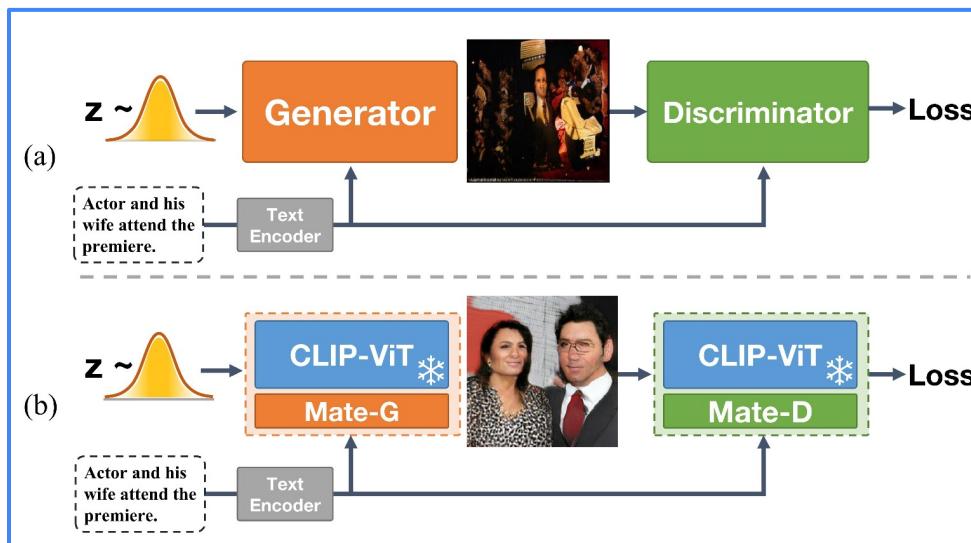
Editing

2022

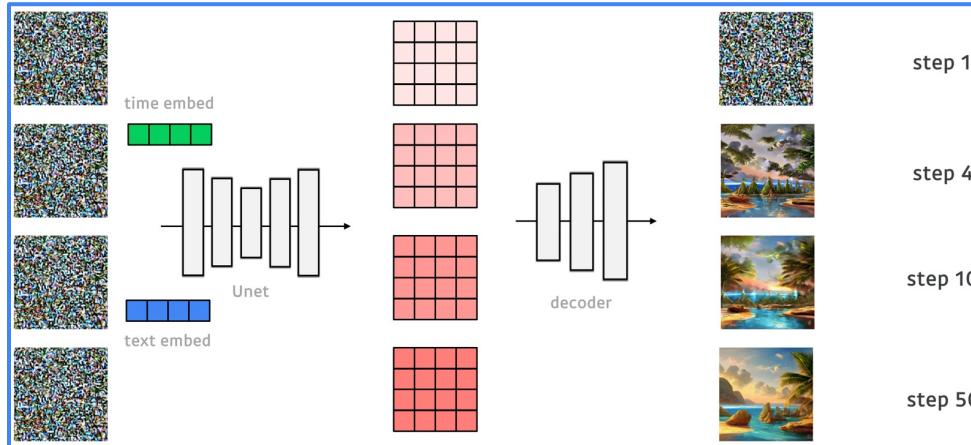
???

Evaluation

FID, Clip score



Summary



Stable diffusion, GALIP

Trending T2I

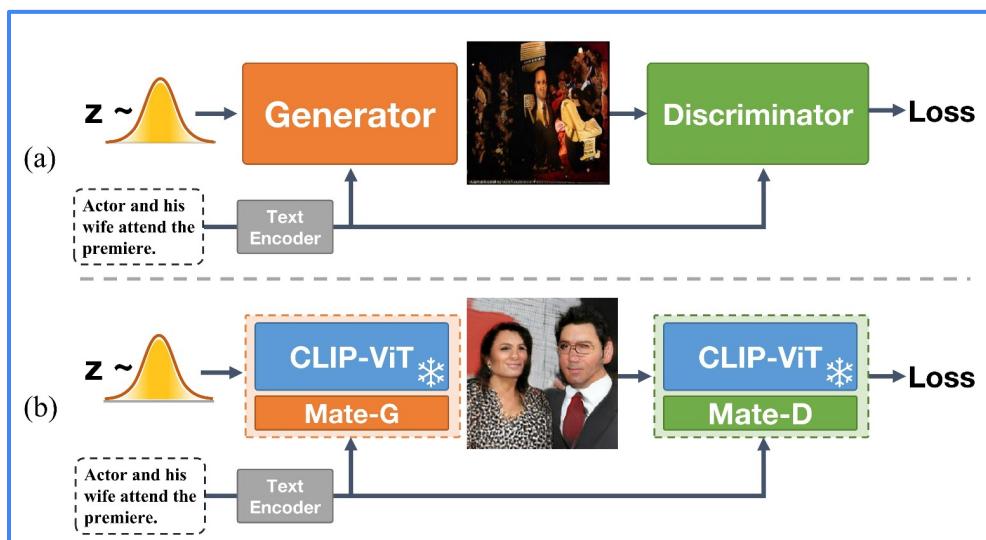
P2P, Dreambooth, Inversion

Editing

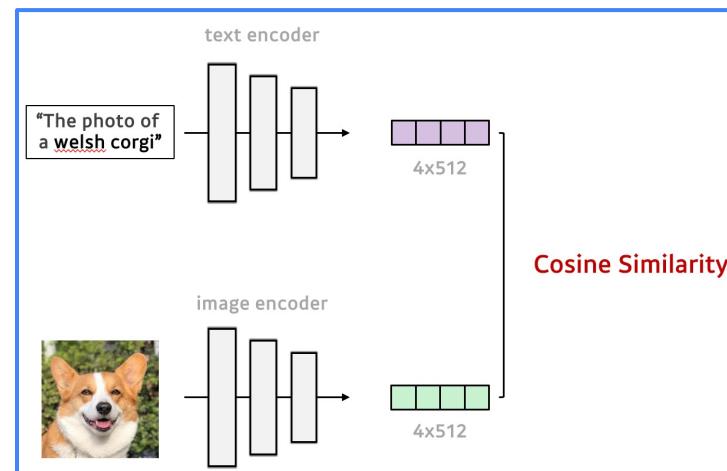
2022

???

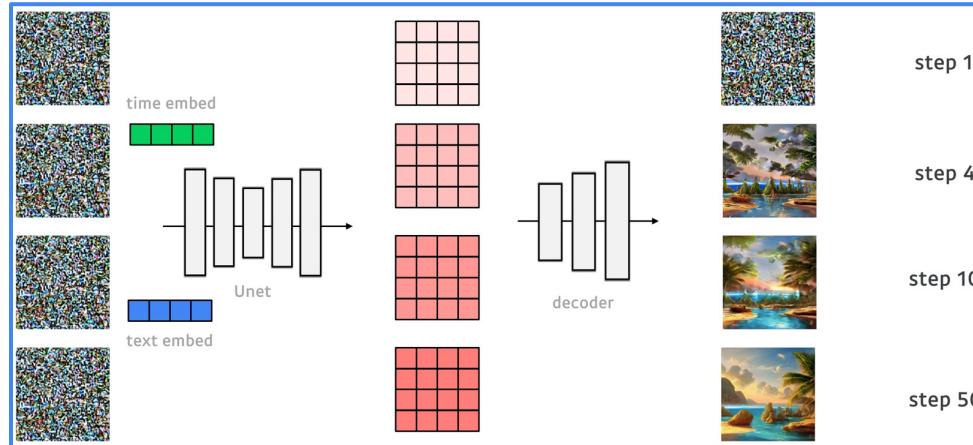
Evaluation



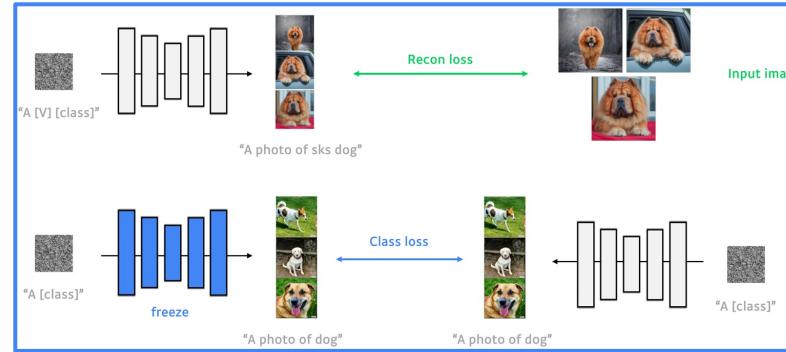
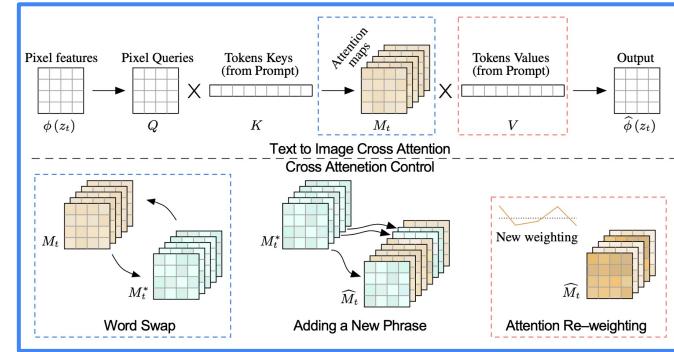
FID, Clip score



Summary



Trending T2I



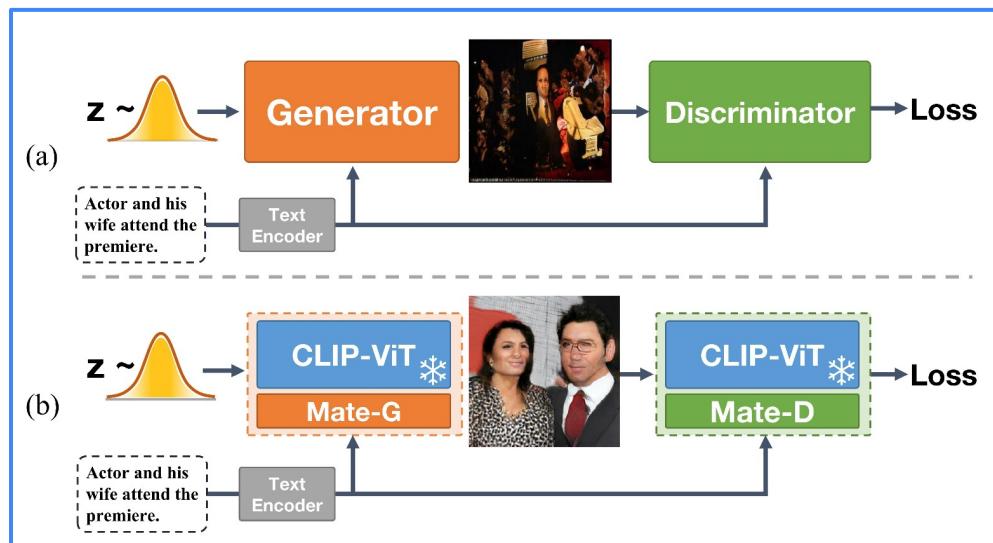
P2P, Dreambooth, Inversion

Editing

2022

???

Evaluation



FID, Clip score

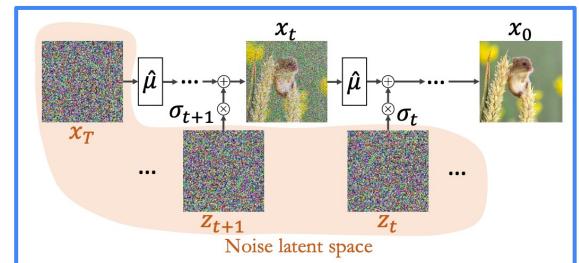
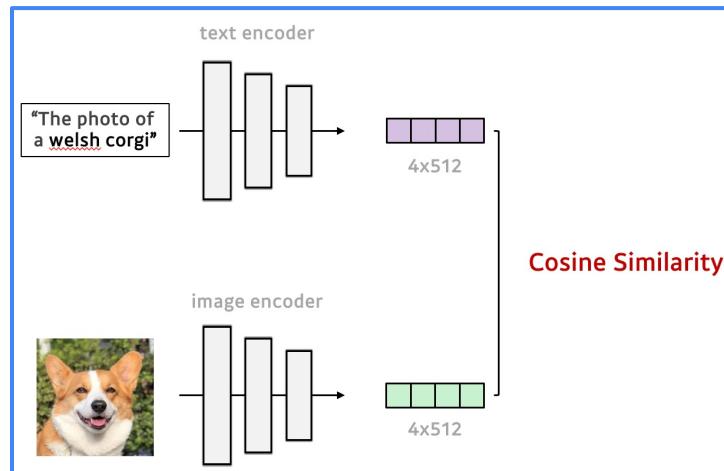


Figure 3: The DDPM latent noise space. In DDPM, the generative (reverse) diffusion process synthesizes an image x_0 in T steps, by utilizing $T + 1$ noise maps, $\{x_T, z_T, \dots, z_1\}$. We regard those noise maps as the latent code associated with the generated image.

Thank you !

jhkim.ai@navercorp.com

Junho Kim