Text to Image models

Lecture 18

Slides from: Karsten Kreis Ruiqi Gao Arash Vahdat

Denoising Diffusion-based Generative Modeling: Foundations and Applications

Karsten Kreis

Ruiqi Gao

Arash Vahdat

NVIDIA

CVPR JUNE 19-24 2022 NEW ORLEANS LOUISIANA

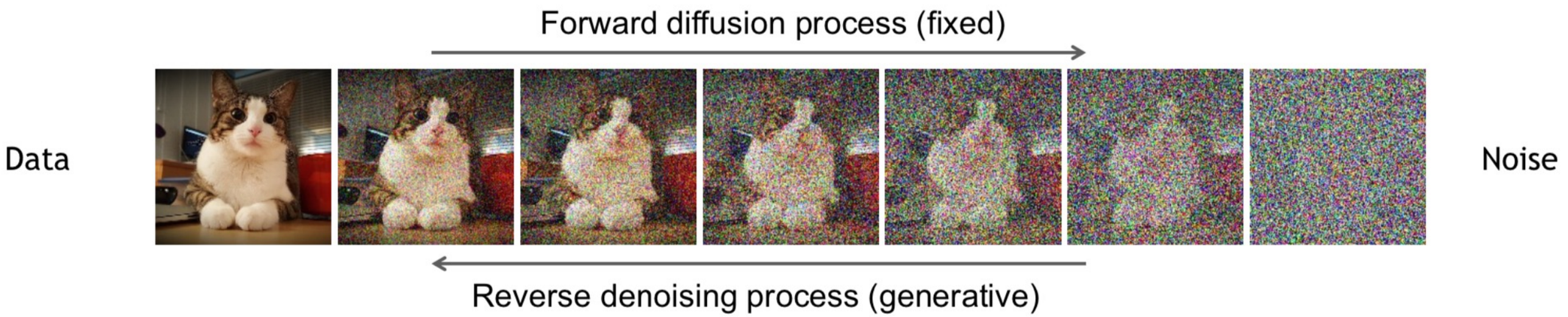
Google AI

Denoising Diffusion Models

Learning to generate by denoising

Denoising diffusion models consist of two processes:

- Forward diffusion process that gradually adds noise to input
- Reverse denoising process that learns to generate data by denoising



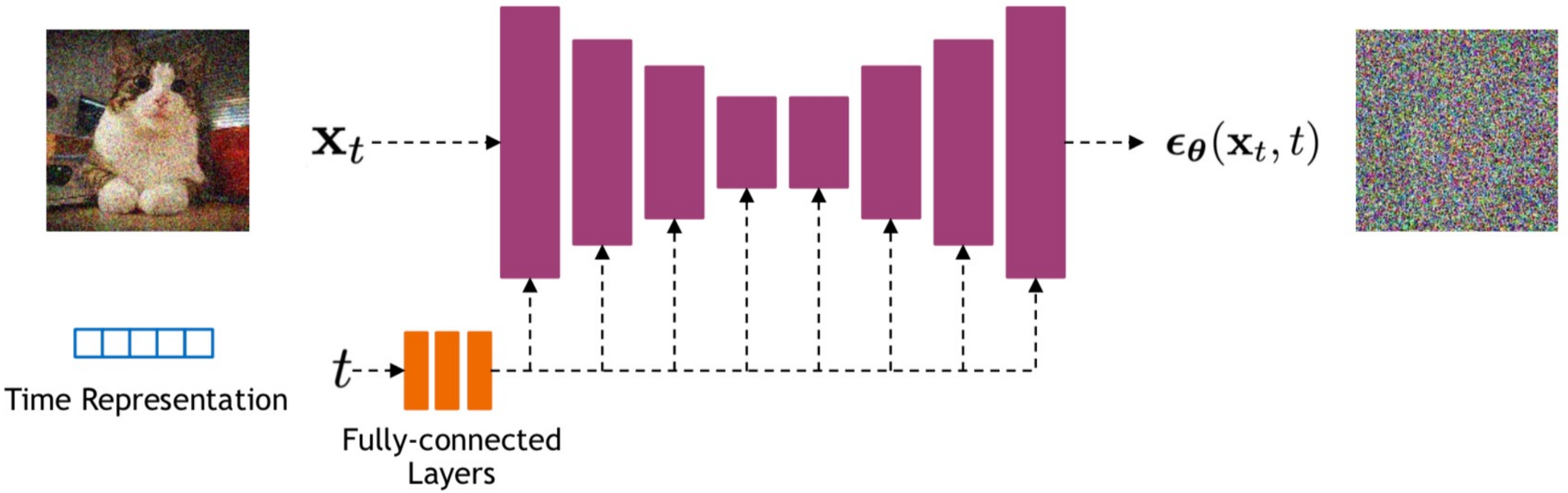
Sohl-Dickstein et al., Deep Unsupervised Learning using Nonequilibrium Thermodynamics, ICML 2015
Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS 2020
Song et al., Score-Based Generative Modeling through Stochastic Differential Equations, ICLR 2021

[Kreis, Gao & Vahdat, CVPR22]

Implementation Considerations

Network Architectures

Diffusion models often use U-Net architectures with ResNet blocks and self-attention layers to represent $\epsilon_\theta(\mathbf{x}_t, t)$



Time representation: sinusoidal positional embeddings or random Fourier features.

Time features are fed to the residual blocks using either simple spatial addition or using adaptive group normalization layers. (see Dharivwal and Nichol NeurIPS 2021)

[Kreis, Gao & Vahdat, CVPR22]

Summary

Training and Sample Generation

Algorithm 1 Training

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

Algorithm 2 Sampling

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

[Kreis, Gao & Vahdat, CVPR22]

Impressive conditional diffusion models

Text-to-image generation

DALL·E 2

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

IMAGEN

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

Ramesh et al., "Hierarchical Text-Conditional Image Generation with CLIP Latents", arXiv 2022.
Saharia et al., "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding", arXiv 2022.

Conditional diffusion models

Include condition as input to reverse process

$$p_\theta(\mathbf{x}_{0:T}|\mathbf{c}) = p(\mathbf{x}_T)\prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{c}), \quad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{c}) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t, \mathbf{c}), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t, \mathbf{c}))$$

$$L_\theta(\mathbf{x}_0|\mathbf{c}) = \mathbb{E}_q\left[L_T(\mathbf{x}_0) + \sum_{t>1} D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{c})) - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1, \mathbf{c})\right].$$

- Scalar/vector conditioning: incorporate scalar embedding or vector into intermediate layers, using either simple spatial addition or adaptive group normalization layers.
- Image conditioning: channel-wise concatenation of the conditional image.
- Text conditioning: cross-attention with intermediate layers.

[Kreis, Gao & Vahdat, CVPR22]

Classifier guidance

Using the gradient of a trained classifier as guidance

Algorithm 1 Classifier guided diffusion sampling, given a diffusion model $(\mu_\theta(x_t), \Sigma_\theta(x_t))$, classifier $p_\phi(y|x_t)$, and gradient scale s.

Input: class label y, gradient scale s
$x_T \leftarrow$ sample from $\mathcal{N}(0, \mathbf{I})$
for all t from T to 1 **do**
  $\mu, \Sigma \leftarrow \mu_\theta(x_t), \Sigma_\theta(x_t)$
  $x_{t-1} \leftarrow$ sample from $\mathcal{N}(\mu + s\Sigma \nabla_{x_t} \log p_\phi(y|x_t), \Sigma)$
end for
return x_0

Score model

Classifier gradient

Main Idea

For class-conditional modeling of $p(\mathbf{x}_t|\mathbf{c})$, train an extra image classifier $p(\mathbf{c}|\mathbf{x}_t)$

Mix its gradient with the score model during sampling

Dhariwal and Nichol, "Diffusion models beat GANs on image synthesis", NeurIPS 2021.

[Kreis, Gao & Vahdat, CVPR22]

Classifier guidance

Using the gradient of a trained classifier as guidance

Algorithm 1 Classifier guided diffusion sampling, given a diffusion model $(\mu_\theta(x_t), \Sigma_\theta(x_t))$, classifier $p_\phi(y|x_t)$, and gradient scale s.

Input: class label y, gradient scale s
$x_T \leftarrow$ sample from $\mathcal{N}(0, \mathbf{I})$
for all t from T to 1 **do**
  $\mu, \Sigma \leftarrow \mu_\theta(x_t), \Sigma_\theta(x_t)$
  $x_{t-1} \leftarrow$ sample from $\mathcal{N}(\mu + s\Sigma \nabla_{x_t} \log p_\phi(y|x_t), \Sigma)$
end for
return x_0

Score model

Classifier gradient

Main Idea

Sample with a modified score: $\nabla_{\mathbf{x}_t}[\log p(\mathbf{x}_t|\mathbf{c}) + \omega \log p(\mathbf{c}|\mathbf{x}_t)]$

Approximate samples from the distribution $\tilde{p}(\mathbf{x}_t|\mathbf{c}) \propto p(\mathbf{x}_t|\mathbf{c})p(\mathbf{c}|\mathbf{x}_t)^\omega$

Dhariwal and Nichol, "Diffusion models beat GANs on image synthesis", NeurIPS 2021.

[Kreis, Gao & Vahdat, CVPR22]

Classifier-free guidance

Get guidance by Bayes' rule on conditional diffusion models

- Instead of training an additional classifier, get an "implicit classifier" by jointly training a conditional and unconditional diffusion model:

$$p(\mathbf{c}|\mathbf{x}_t) \propto p(\mathbf{x}_t|\mathbf{c})/p(\mathbf{x}_t)$$

Conditional model

Unconditional model

- The modified score with this implicit classifier is:

$$\nabla_{\mathbf{x}_t}[(1+\omega)\log p(\mathbf{x}_t|\mathbf{c}) - \omega \log p(\mathbf{x}_t)]$$

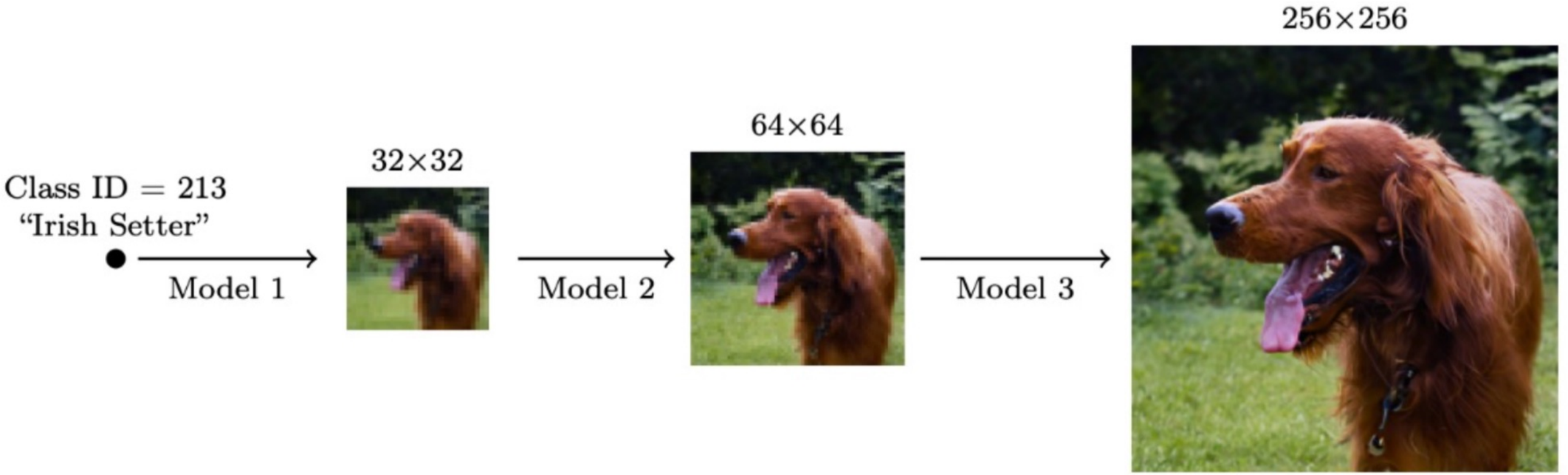
- In practice, we can jointly train the conditional and unconditional diffusion models by randomly dropping the condition.

Ho & Salimans, "Classifier-Free Diffusion Guidance", 2021.

[Kreis, Gao & Vahdat, CVPR22]

Cascaded generation

Pipeline



Cascaded Diffusion Models outperform Big-GAN FID and IS and VQ-VAE2 in Classification Accuracy Score.

Ho et al., "Cascaded Diffusion Models for High Fidelity Image Generation", 2021.

[Kreis, Gao & Vahdat, CVPR22]

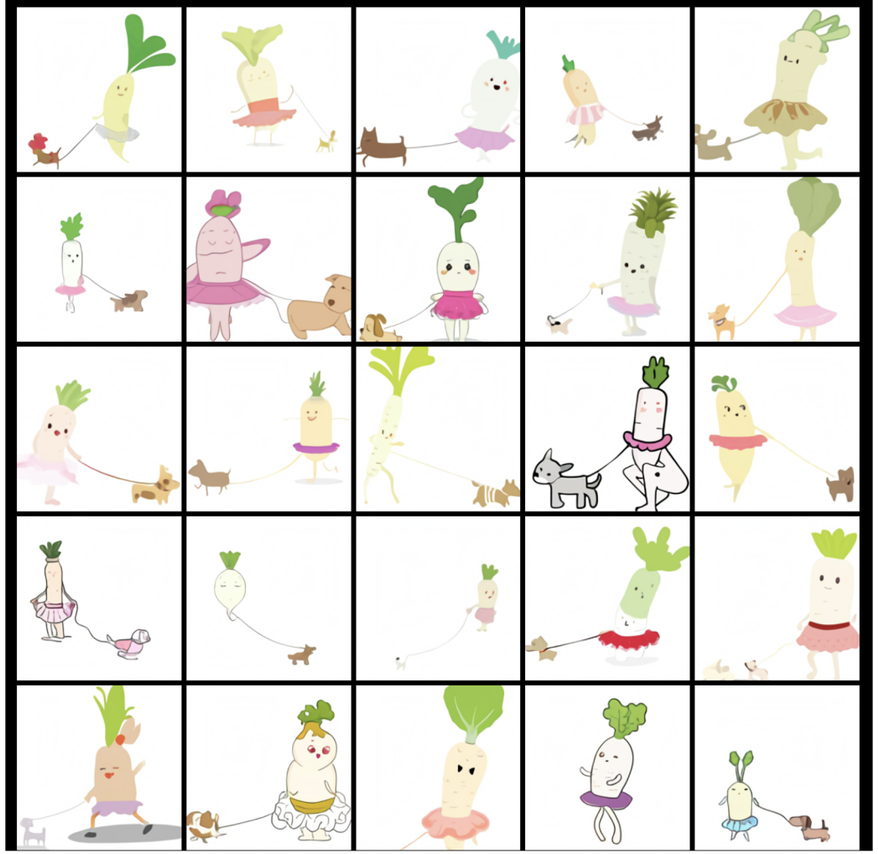
DALL·E 2

Aditya Ramesh, Prafulla Dhariwal,
Alex Nichol, Casey Chu, Mark Chen

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

DALL·E 1

an illustration of a baby daikon radish in a tutu walking a dog



a store front that has the word 'openai' written on it. a store front that has the word 'openai' written on it. a store front that has the word 'openai' written on it. openai store front.

an armchair in the shape of an avocado. an armchair imitating an avocado.

“a propaganda poster depicting a cat dressed as French Emperor Napoleon holding a piece of cheese”

[Slide: A. Ramesh]

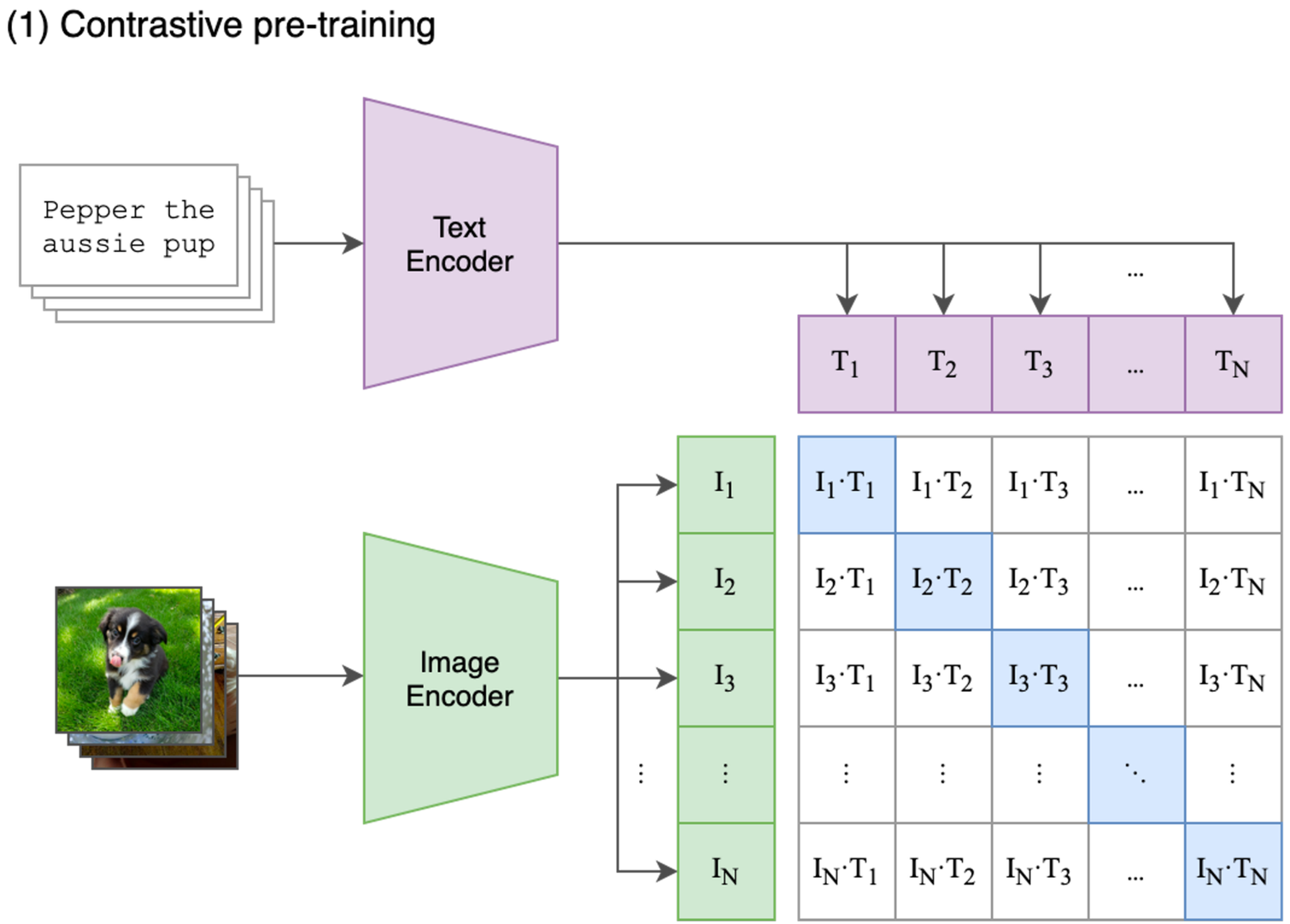
[Slide: A. Ramesh]

Building blocks for DALL·E 2

1. CLIP
2. Diffusion

[Slide: A. Ramesh]

CLIP (Radford et al., 2021)



[Slide: A. Ramesh]

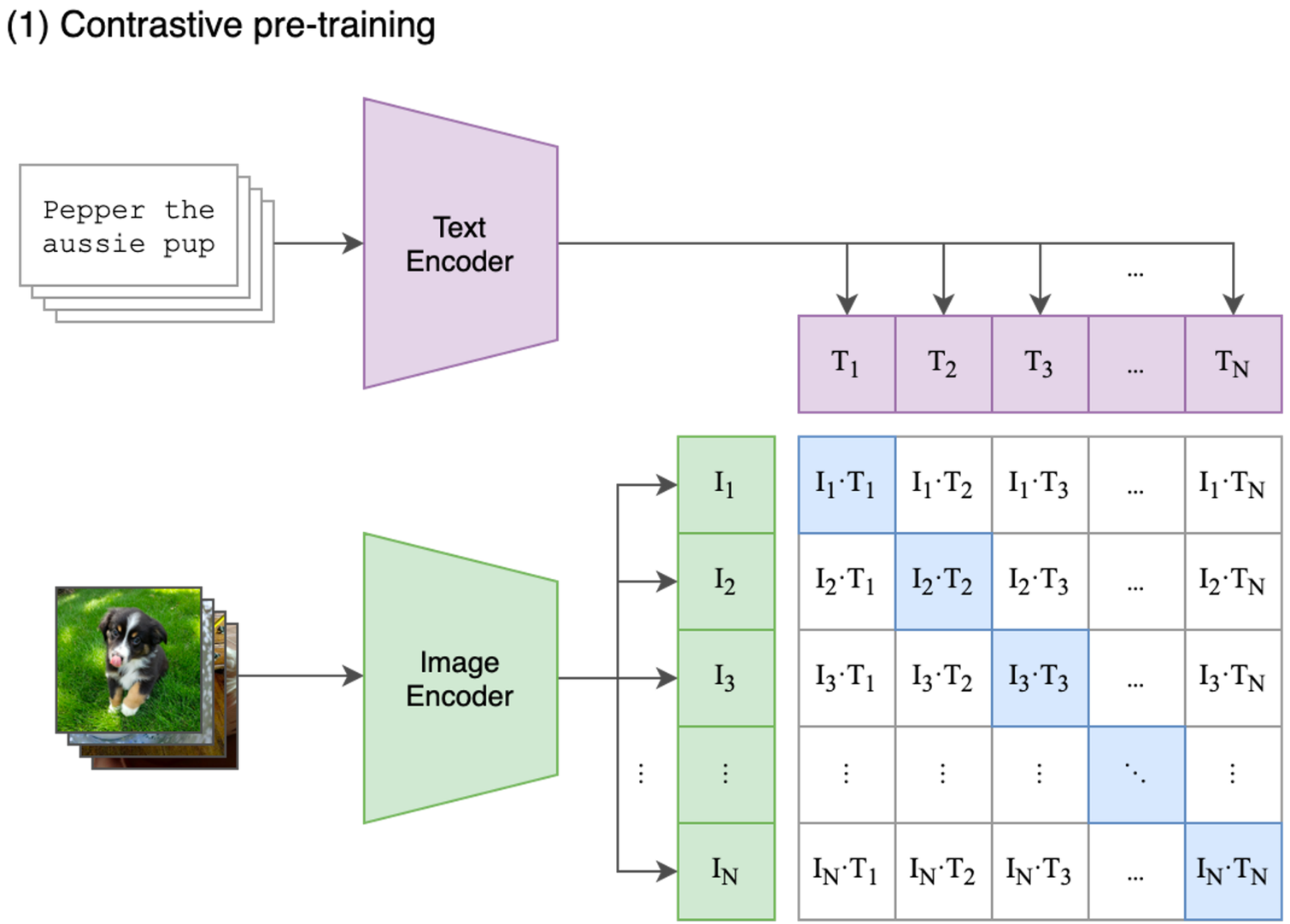
CLIP (Radford et al., 2021)

```
# 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
```

Figure 3. Numpy-like pseudocode for the core of an implementation of CLIP.

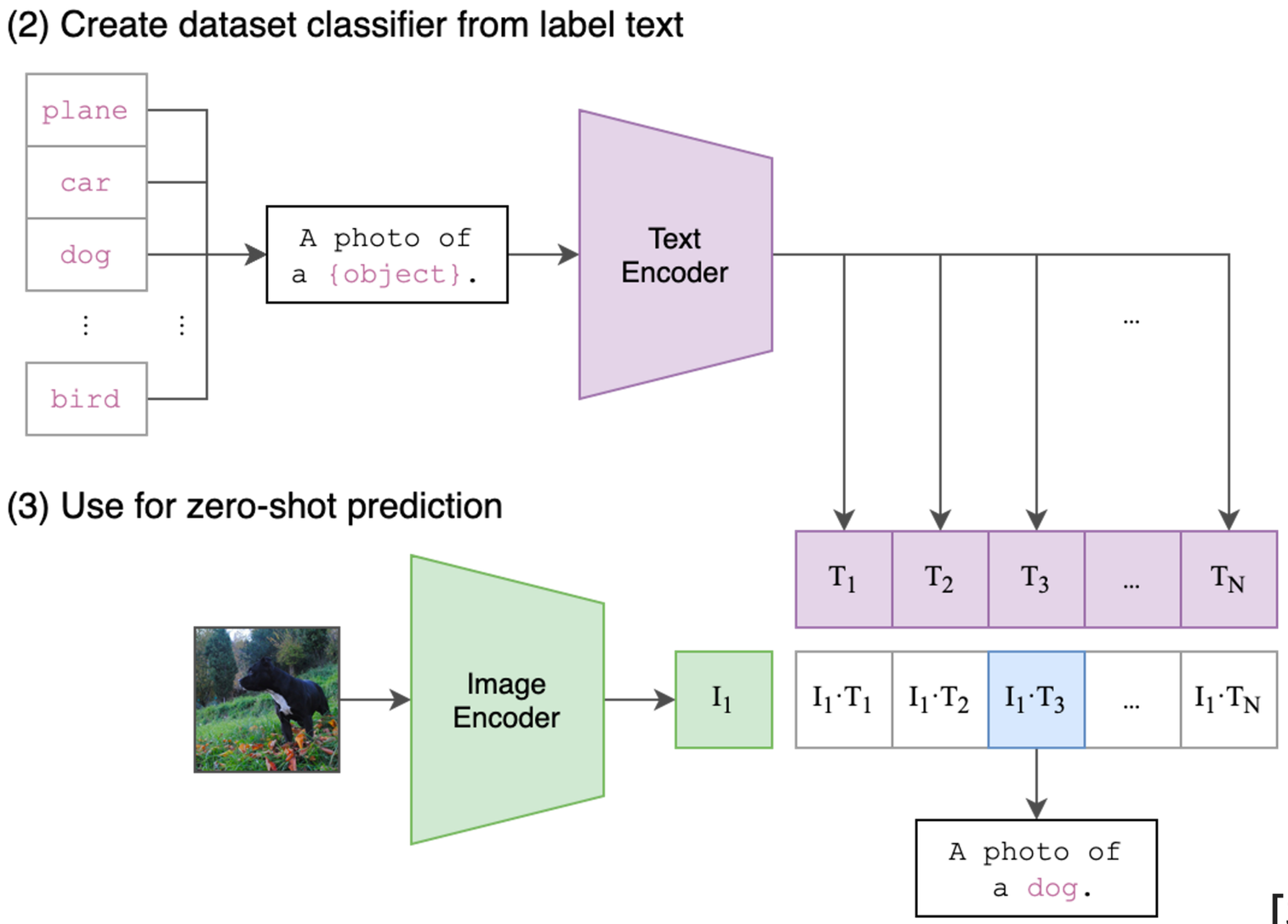
[Slide: A. Ramesh]

CLIP (Radford et al., 2021)



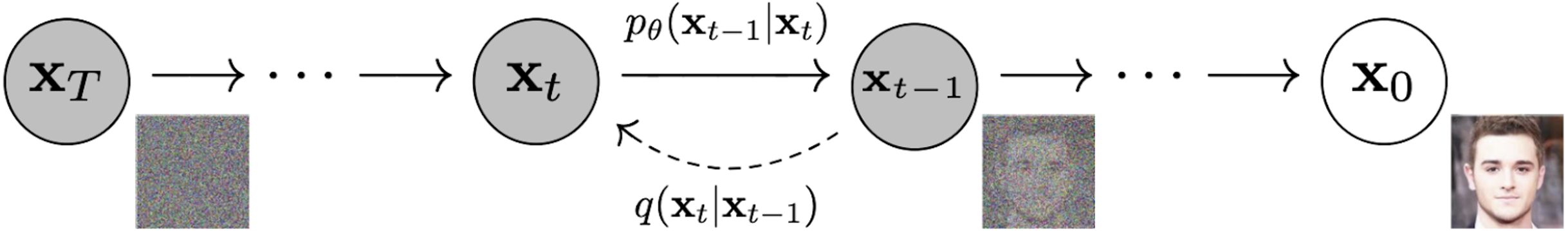
[Slide: A. Ramesh]

CLIP (Radford et al., 2021)



[Slide: A. Ramesh]

Diffusion (Sohl-Dickstein et al., 2016; Ho et al., 2020)



[Slide: A. Ramesh]

Diffusion (Sohl-Dickstein et al., 2016; Ho et al., 2020)

Algorithm 1 Training

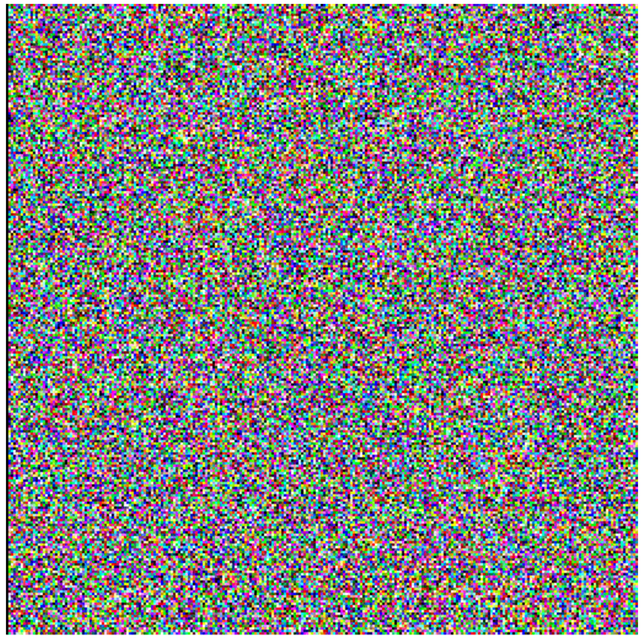
1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0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

Algorithm 2 Sampling

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

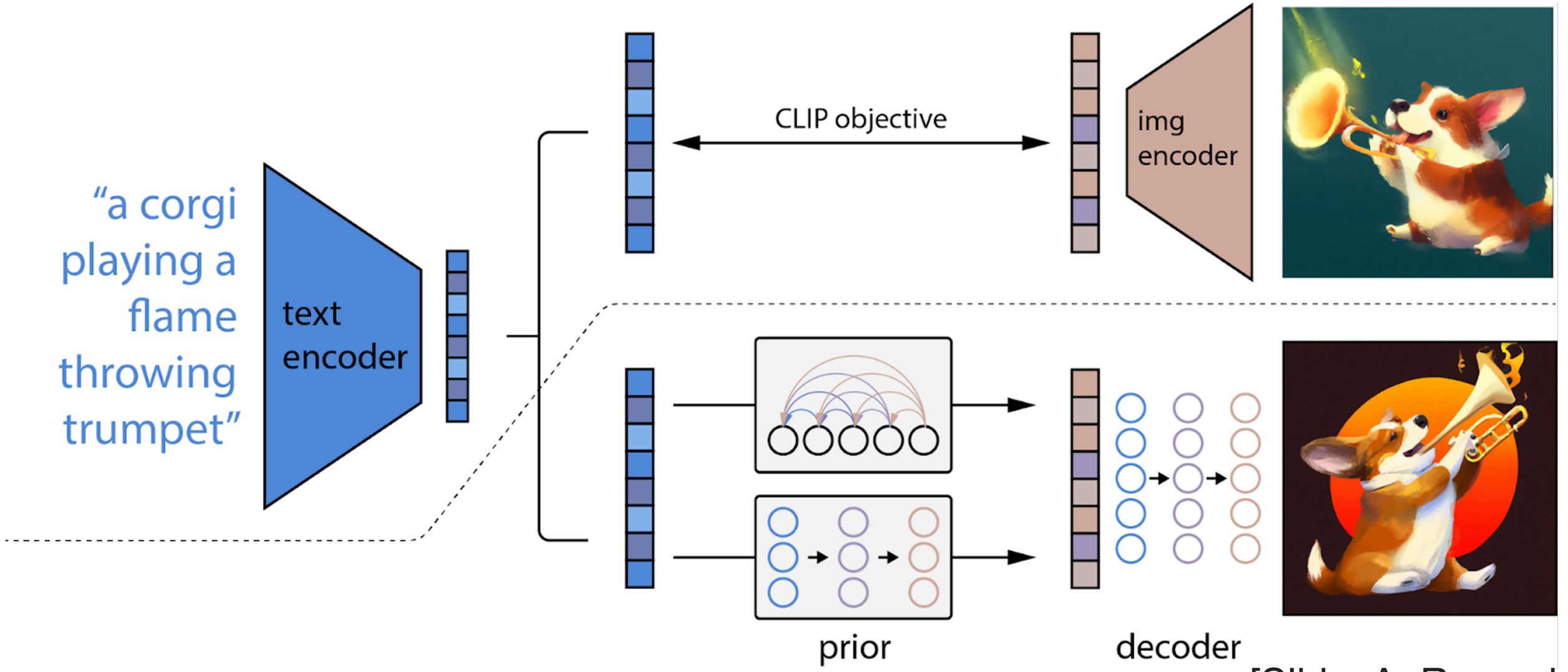
[Slide: A. Ramesh]

Diffusion (Sohl-Dickstein et al., 2016; Ho et al., 2020)



[Slide: A. Ramesh]

The unCLIP stack



[Slide: A. Ramesh]

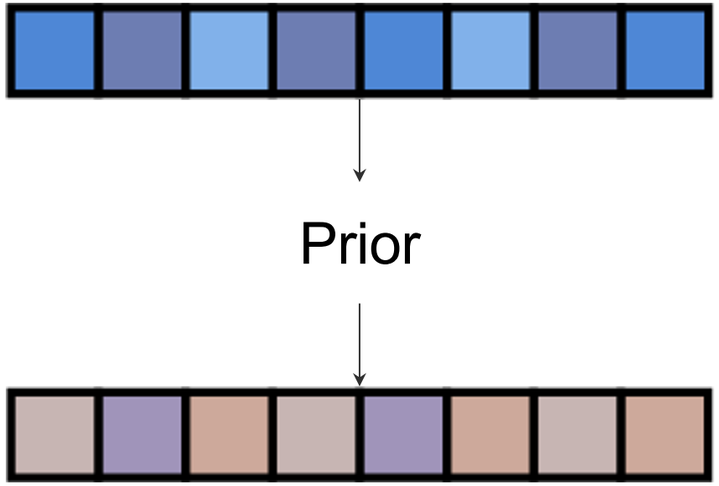
Generating an image from a caption

1. Encode the caption with the CLIP text encoder to get the text representation *z_t*

[Slide: A. Ramesh]

Generating an image from a caption

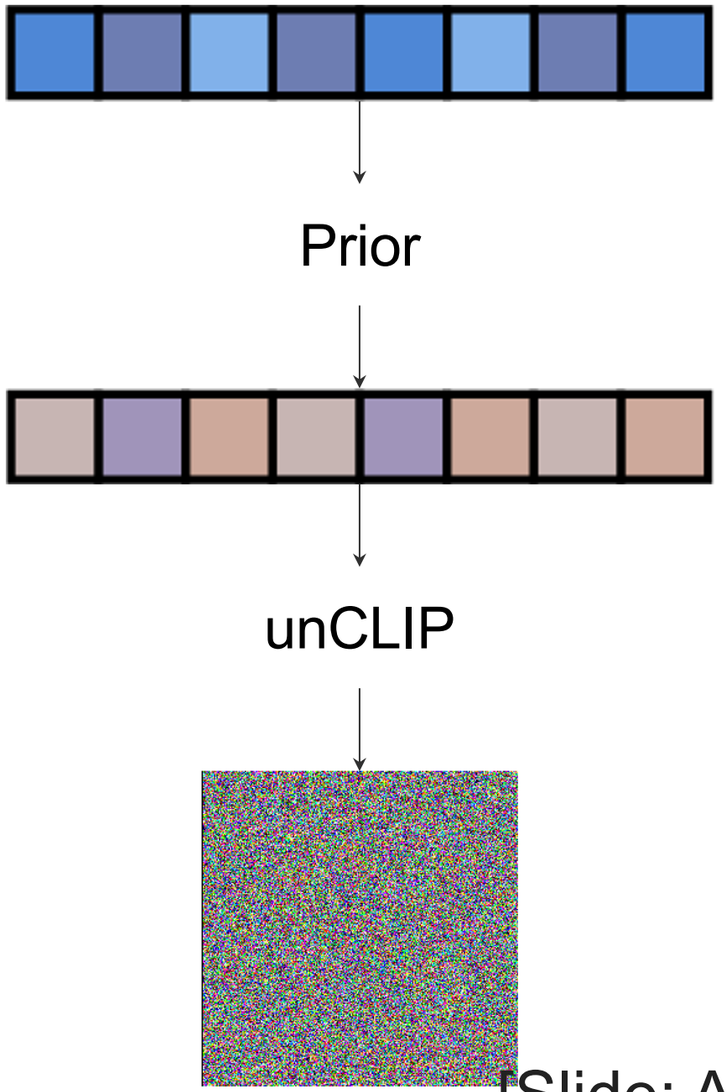
1. Encode the caption with the CLIP text encoder to get the text representation *z_t*
2. Use the prior to sample a CLIP image representation *z_i* given the caption and *z_t*



[Slide: A. Ramesh]

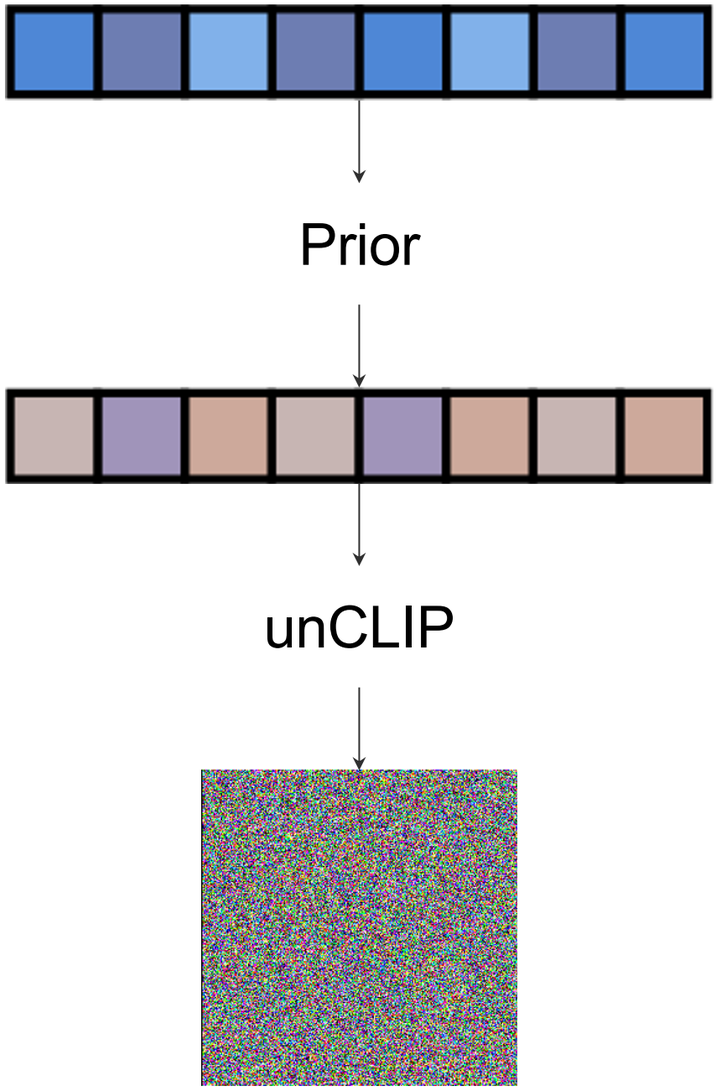
Generating an image from a caption

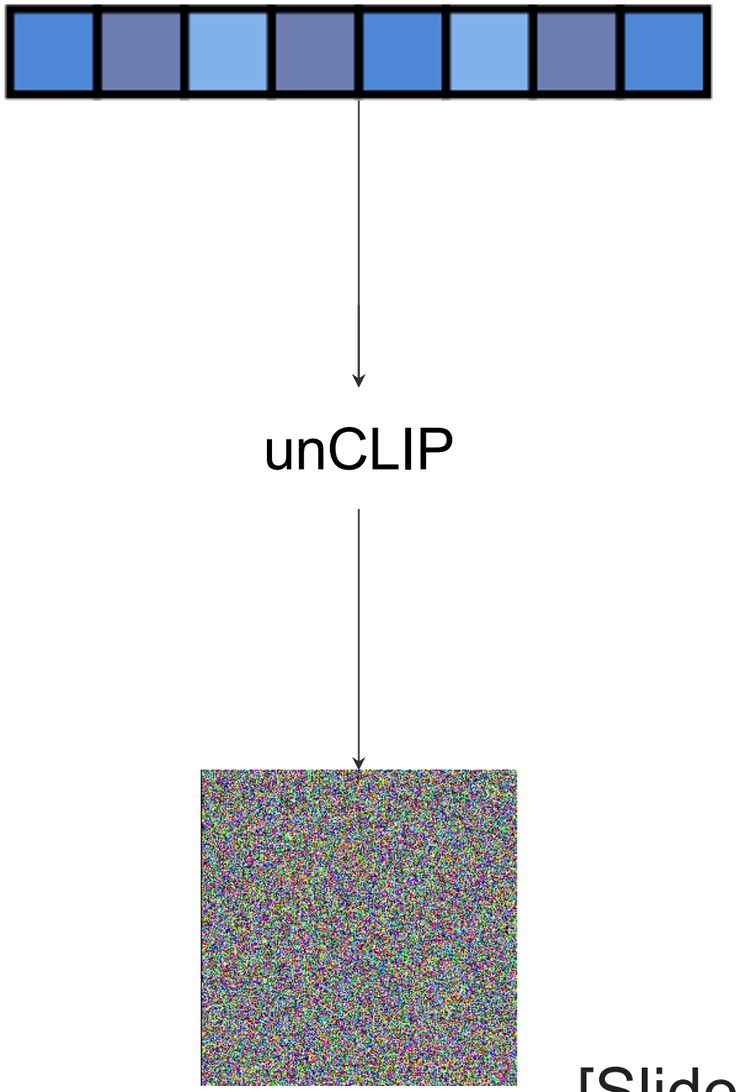
1. Encode the caption with the CLIP text encoder to get the text representation *z_t*
2. Use the prior to sample a CLIP image representation *z_i* given the caption and *z_t*
3. Use the unCLIP diffusion model to generate an image given the caption and *z_i*



[Slide: A. Ramesh]

Why do we need a prior?





[Slide: A. Ramesh]

Training compute for unCLIP paper models

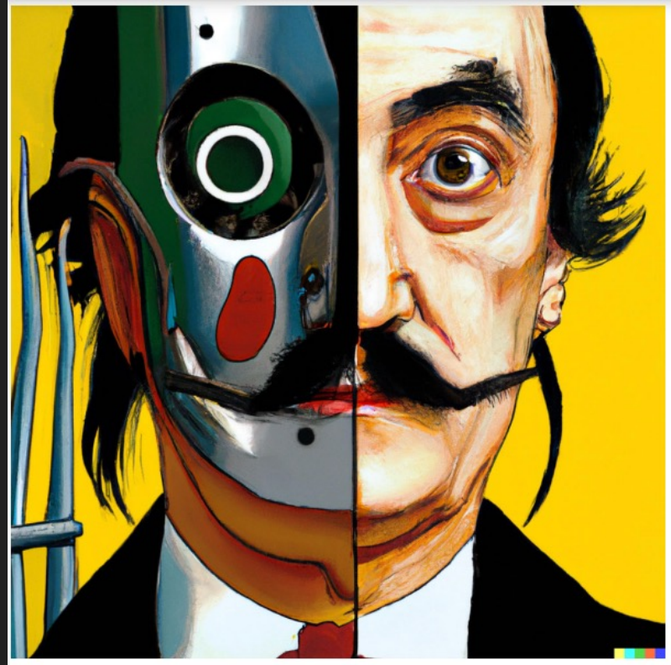
- CLIP model uses ViT-H/16 image encoder and 1024W / 24L text encoder.
- AR prior uses 1664W / 64L decoder with 2048W / 24L text encoder (~GPT-2 scale)
  - Also try a diffusion prior
- Decoder: unCLIP diffusion model ~3.5B ADMNet (improved UNet). Generates at 64x64 resolution.
- Upsamplers use ADMNets without attention
  - 64→256 upsampler is ~700M params
  - 256→1024 upsampler is ~300M params
- More details given in Appendix C of paper.

[Slide: A. Ramesh]

Dataset sizes for unCLIP paper models

- CLIP model is trained on a 50-50 mix of the datasets described in the CLIP and DALL-E 1 papers.
- All other models are trained on the DALL·E 1 dataset only.
- 650M {text,image text} pairs

[Slide: A. Ramesh]



vibrant portrait painting of Salvador Dalí with a robotic half face

a shiba inu wearing a beret and black turtleneck

a close up of a handpalm with leaves growing from it

an espresso machine that makes coffee from human souls, artstation

panda mad scientist mixing sparkling chemicals, artstation

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

[Slide: A. Ramesh]

Why do we need a prior?

Only caption, no CLIP image embedding

Caption + substitute CLIP text embedding for image embedding

Caption + generate CLIP image embedding from prior

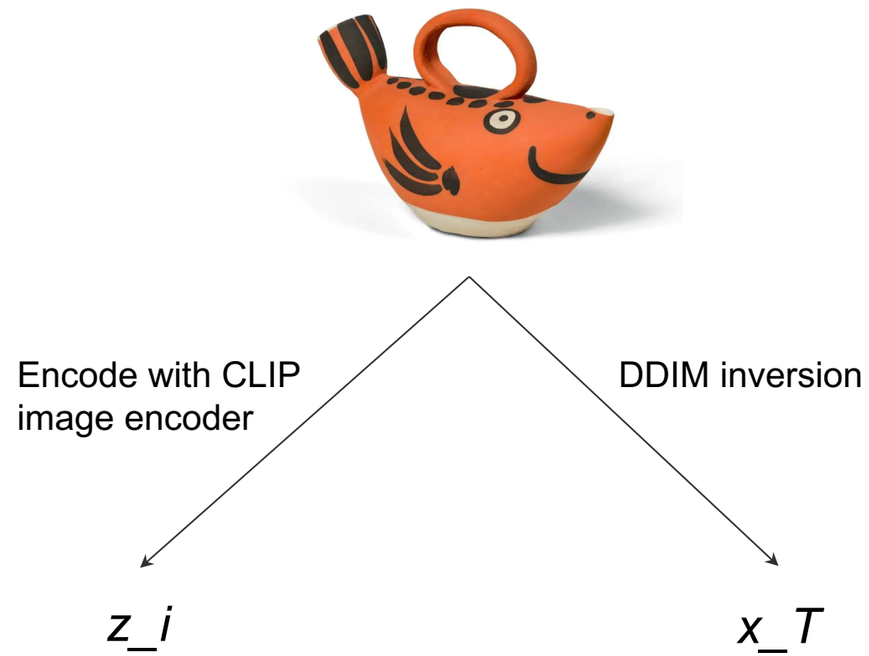
Why use CLIP?

1. unCLIP decoder can inherit knowledge of the world and aesthetic styles from CLIP
2. Enables language-guided image manipulations to be applied to images
3. Better quality-diversity tradeoff than generating images from scratch
4. Allows us to invert CLIP representations to see what is happening when it makes “stupid mistakes”

[Slide: A. Ramesh]

Bipartite latent representation

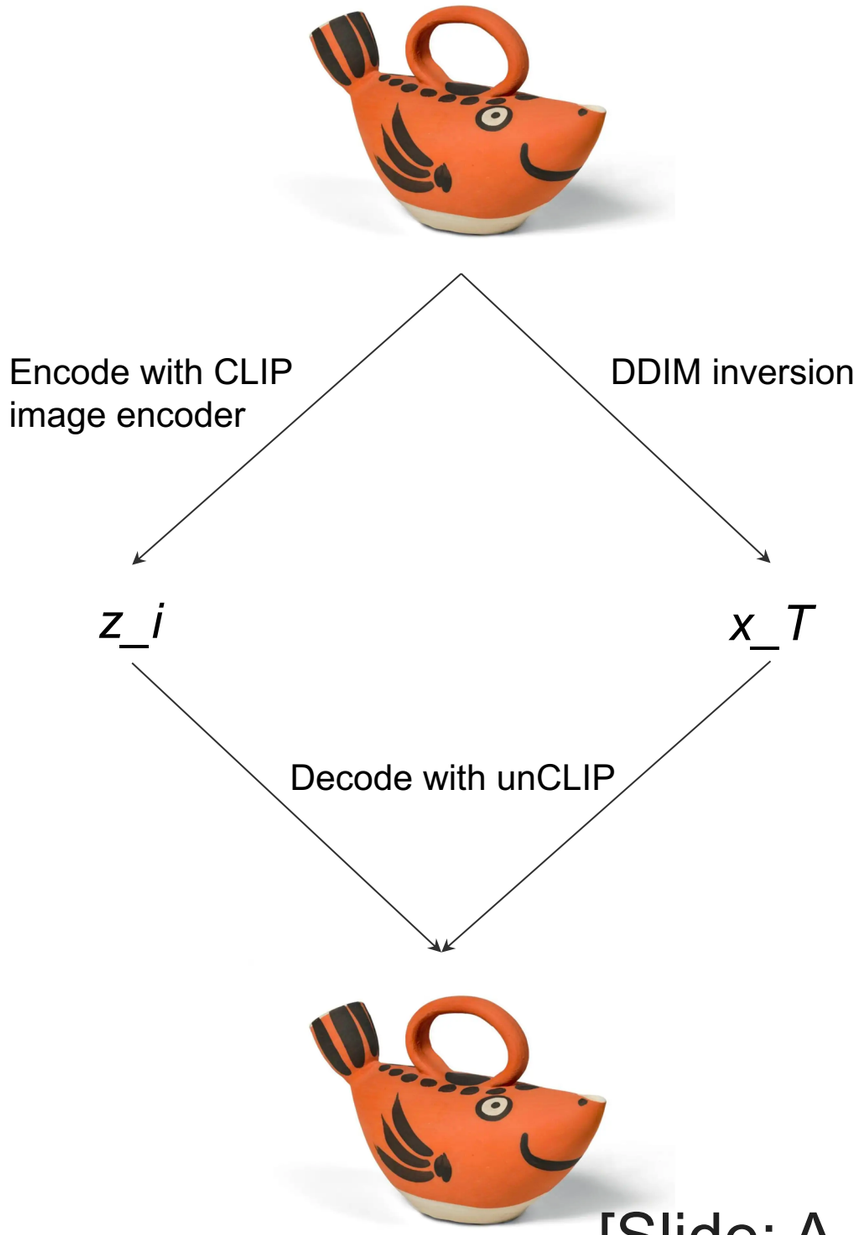
- We can encode a given image *x* into a bipartite representation (*z_i*, *x_T*), where *z_i* represents everything about the image that was recognized by CLIP, and *x_T* encodes all of the residual variation.
- DDIM inversion is described in “Diffusion Models Beat GANs on Image Synthesis” (Dhariwal and Nichol 2021), Appendix F



[Slide: A. Ramesh]

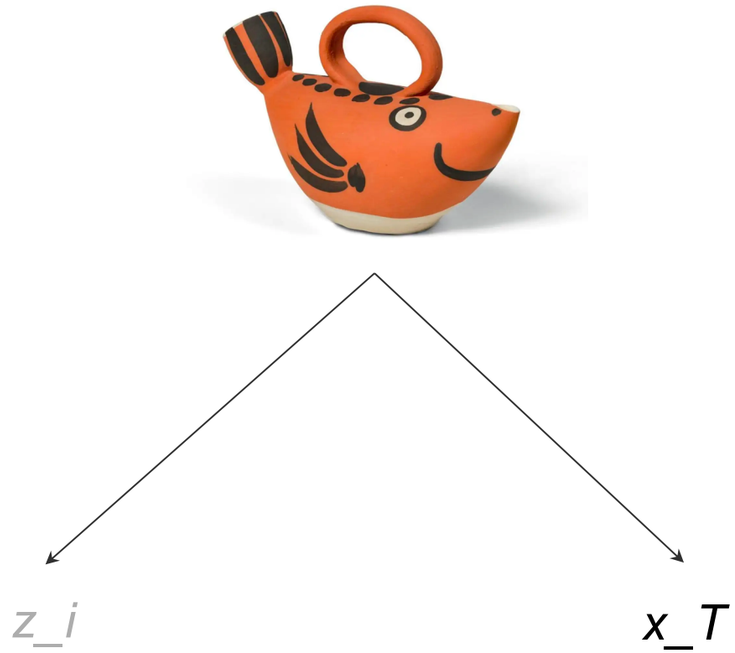
Bipartite latent representation

- Given (*z_i*, *x_T*), the unCLIP decoder can almost perfectly reconstruct *x*.
- This bipartite representation enables three kinds of image manipulations.



[Slide: A. Ramesh]

1. Variations: fix *z_i* and vary *x_T*



[Slide: A. Ramesh]

[Slide: A. Ramesh]

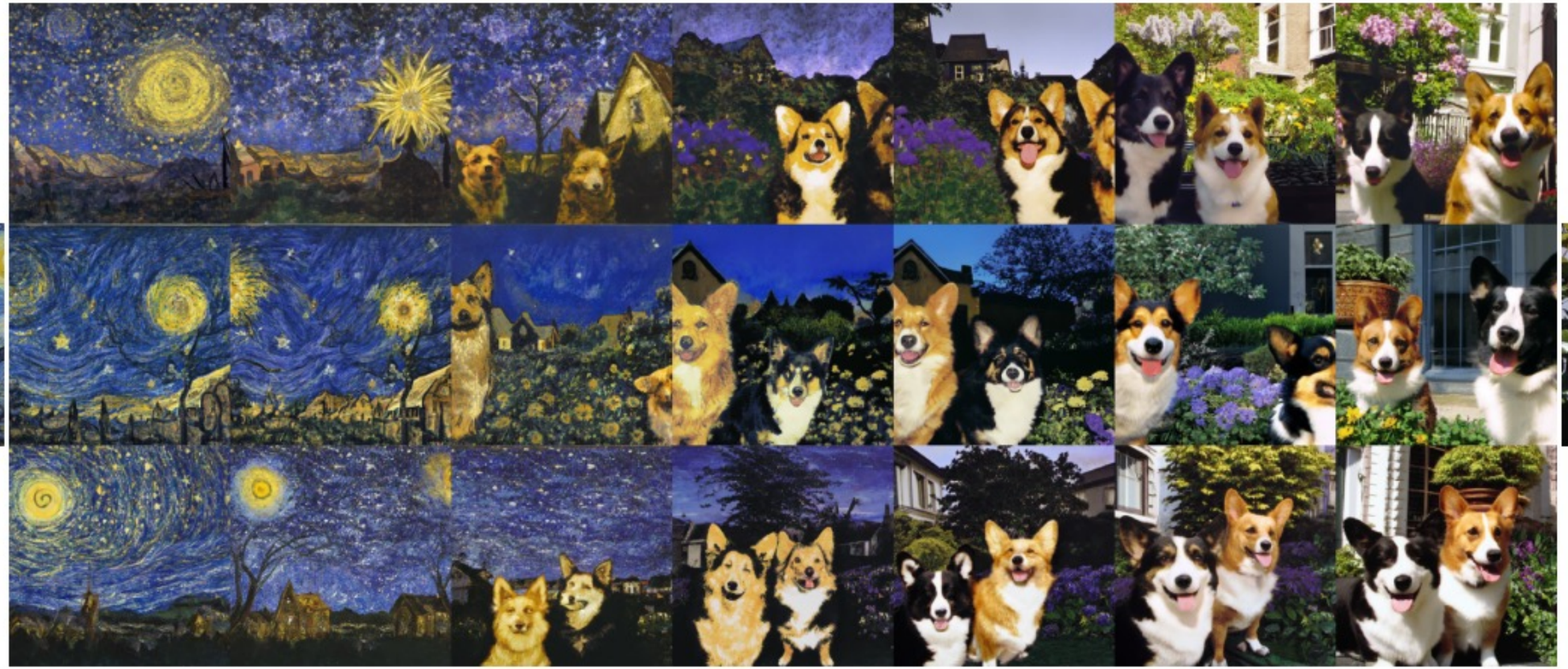
[Slide: A. Ramesh]

[Slide: A. Ramesh]

2. Interpolation

[Slide: A. Ramesh]

[Slide: A. Ramesh]



3. Text Diffs

- CLIP learns a joint embedding space over images and text
- Can we manipulate images using word2vec-style arithmetic with caption embeddings?

[Slide: A. Ramesh]

3. Text Diffs

- Encoded initial caption (e.g. “a photo of a cat”): *z_t0*
- Encoded final caption (e.g. “a photo of a super saiyan cat”): *z_t1*
- Text diff: *z_d = (z_t1 - z_t0) / ||z_t1 - z_t0||*

[Slide: A. Ramesh]

3. Text Diffs

z_i

x_T

slerp(*z_i, z_d, w*)

Typically, *w* is a value in [0.25, 0.50]

[Slide: A. Ramesh]

3. Text Diffs

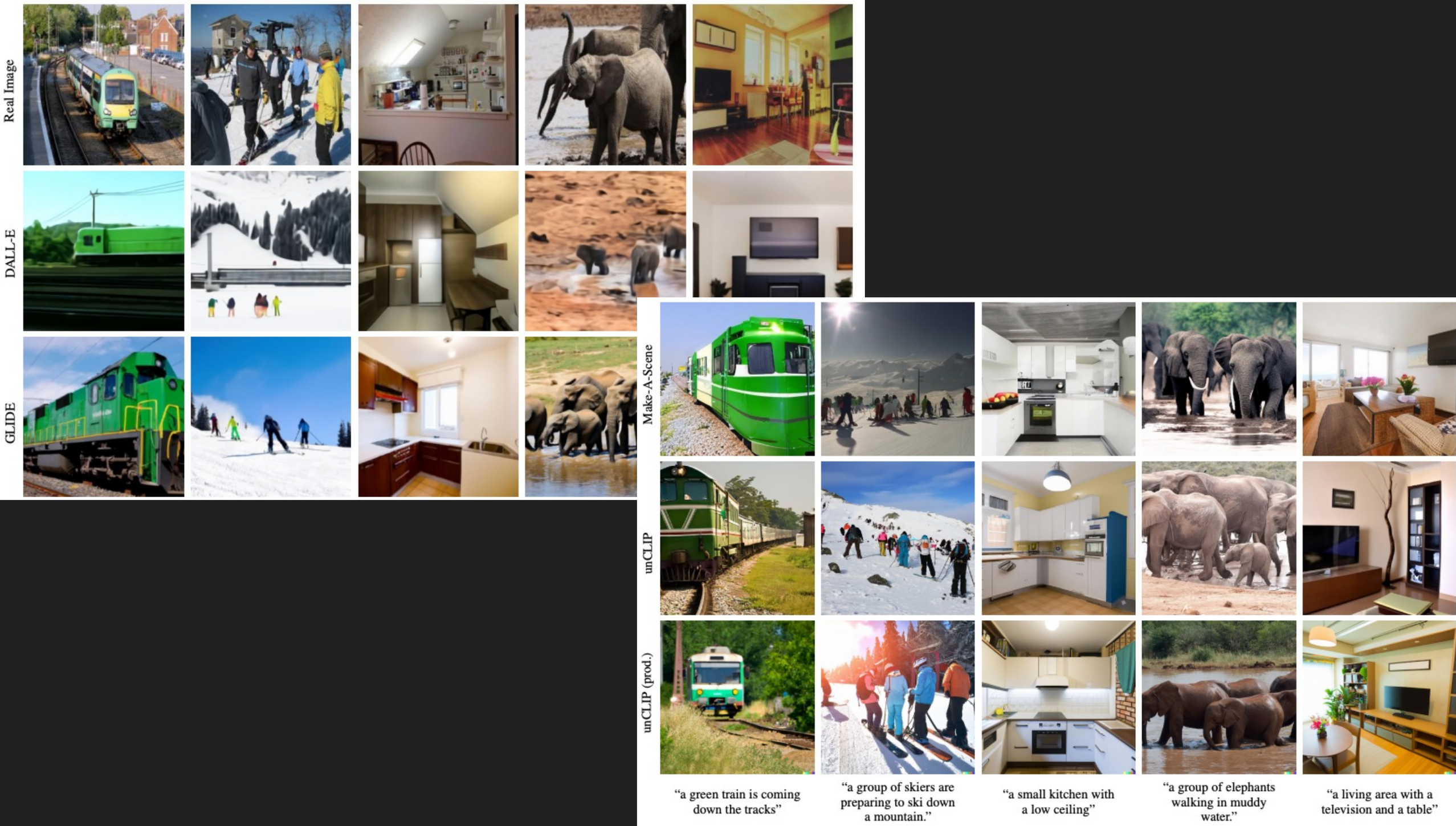
a photo of a cat → an anime drawing of a super saiyan cat, artstation

a photo of a victorian house → a photo of a modern house

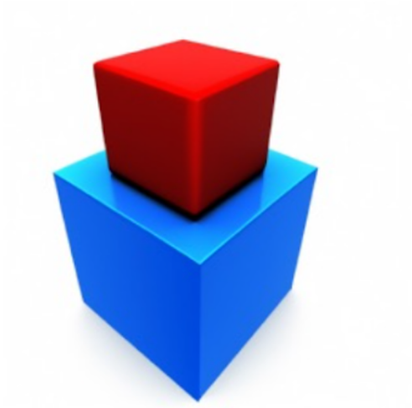
a photo of an adult lion → a photo of lion cub

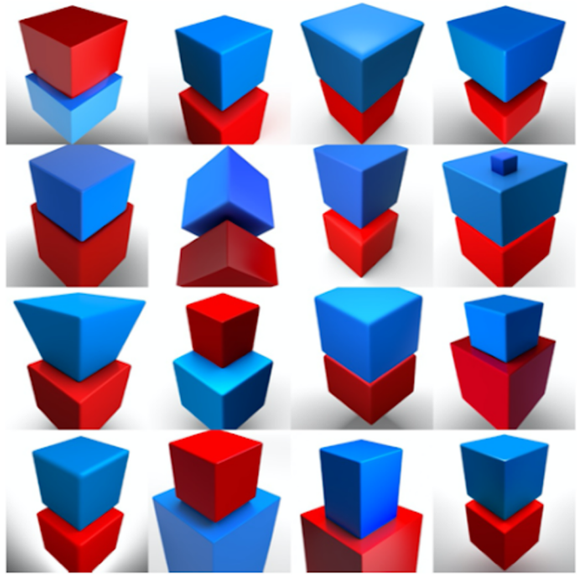
a photo of a landscape in winter → a photo of a landscape in fall

[Slide: A. Ramesh]


Real Image
DALL-E
GLIDE
Make-A-Scene
unCLIP
unCLIP (prod.)
“a green train is coming down the tracks”
“a group of skiers are preparing to ski down a mountain.”
“a small kitchen with a low ceiling”
“a group of elephants walking in muddy water.”
“a living area with a television and a table”

Limitations





[Slide: A. Ramesh]

Figure 18: Random samples from unCLIP for prompt “Vibrant portrait painting of Salvador Dali with a robotic half face”

References

- DALL•E 2
- Diffusion: 1, 2
- Improvements to diffusion: 1, 2
- GLIDE
- DALL•E 1

[Slide: A. Ramesh]

Imagen

Google Research, Brain team

Key modeling components:

- Cascaded diffusion models.
- Classifier-free guidance and dynamic thresholding.
- Large pretrained language models as text encoders.

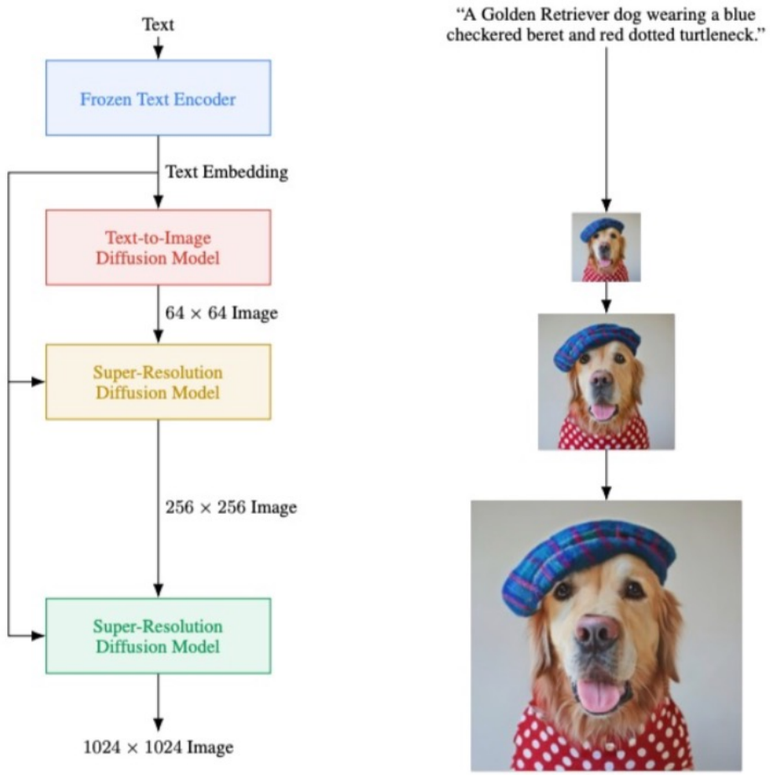


Figure A.4: Visualization of Imagen. Imagen uses a frozen text encoder to encode the input text into text embeddings. A conditional diffusion model maps the text embedding into a 64×64 image. Imagen further utilizes text-conditional super-resolution diffusion models to upsample the image, first $64 \times 64 \rightarrow 256 \times 256$, and then $256 \times 256 \rightarrow 1024 \times 1024$.

Saharia et al., "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding", arXiv 2022.

[Kreis, Gao & Vahdat, CVPR22]

Imagen

Google Research, Brain team

Key modeling components:

- Cascaded diffusion models.
- Classifier-free guidance and dynamic thresholding.
- Large pretrained language models as text encoders.

A brain riding a rocketship heading towards the moon.

- Extreme simple!
  - no latent space, no quantization
- Yet effective!
  - SOTA FID & on-par with the reference images by human raters.

Saharia et al., "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding", arXiv 2022.

[Kreis, Gao & Vahdat, CVPR22]

Imagen

Google Research, Brain team

A photo of a Shiba Inu dog with a backpack riding a bike. It is wearing sunglasses and a beach hat.

Saharia et al., "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding", arXiv 2022.

[Kreis, Gao & Vahdat, CVPR22]

Imagen

Google Research, Brain team

A dragon fruit wearing karate belt in the snow.

Saharia et al., "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding", arXiv 2022.

[Kreis, Gao & Vahdat, CVPR22]

Imagen

Google Research, Brain team

A relaxed garlic with a blindfold reading a newspaper while floating in a pool of tomato soup.

Saharia et al., "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding", arXiv 2022.

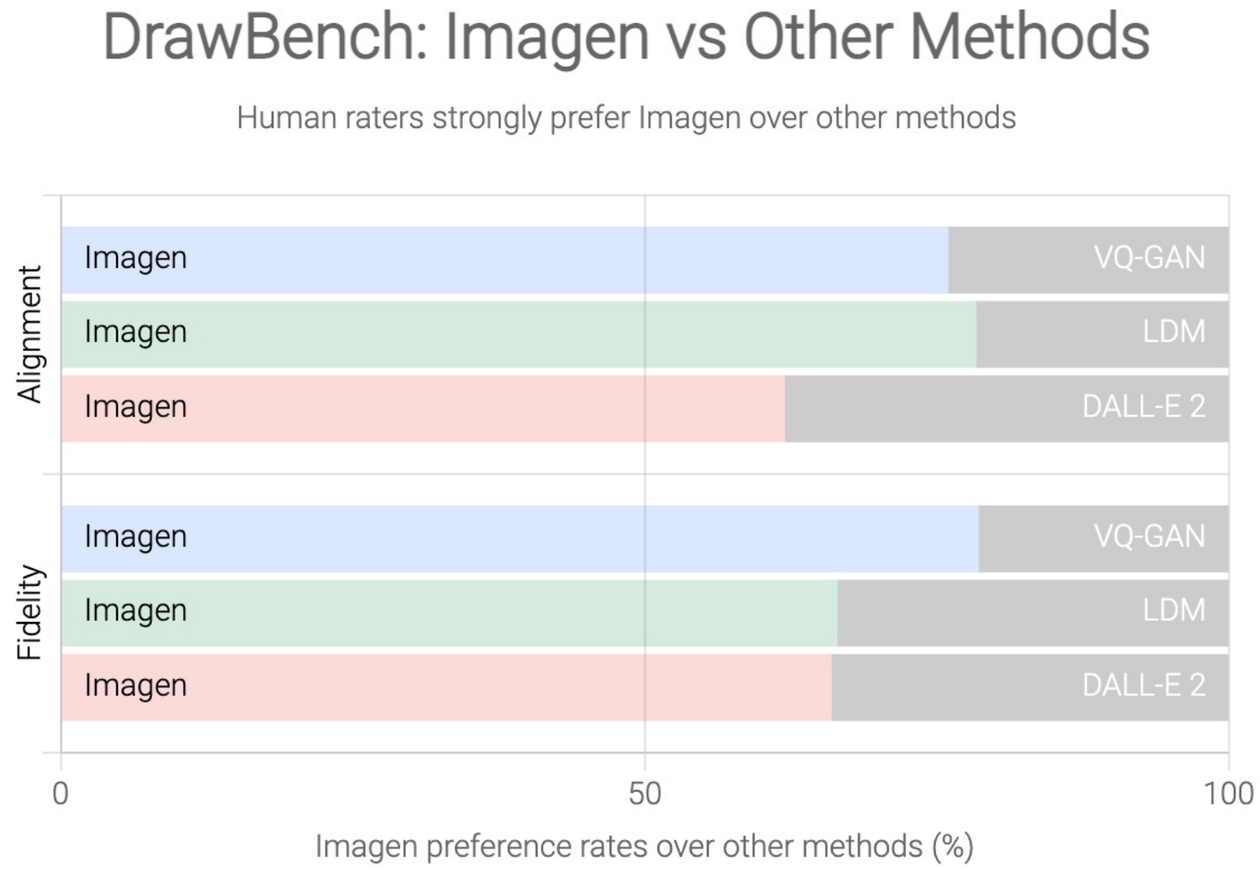
[Kreis, Gao & Vahdat, CVPR22]

ImageGen FID

Model	COCO FID ↓
Trained on COCO	
AttnGAN (Xu et al., 2017)	35.49
DM-GAN (Zhu et al., 2019)	32.64
DF-GAN (Tao et al., 2020)	21.42
DM-GAN + CL (Ye et al., 2021)	20.79
XMC-GAN (Zhang et al., 2021)	9.33
LAFITE (Zhou et al., 2021)	8.12
Make-A-Scene (Gafni et al., 2022)	7.55
Not trained on COCO	
DALL-E (Ramesh et al., 2021)	17.89
GLIDE (Nichol et al., 2021)	12.24
DALL-E 2 (Ramesh et al., 2022)	10.39
Imagen (Our Work)	7.27

Imagen attains a new state-of-the-art COCO FID.

Human Comparison



Imagen

Google Research, Brain team

Key observations:

- Scaling text encoder size is extremely effective, more important than scaling the diffusion model.
- Large pretrained text encoder works better than CLIP embeddings.
- Dynamic thresholding is important, especially for large classifier-free guidance weights.
    - Threshold each image to a certain percentile absolute pixel value, and shrink to [-1, 1].
- Noise conditioning augmentation is important.
- Text conditioning by cross-attention is important.

Saharia et al., "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding", arXiv 2022.

[Kreis, Gao & Vahdat, CVPR22]