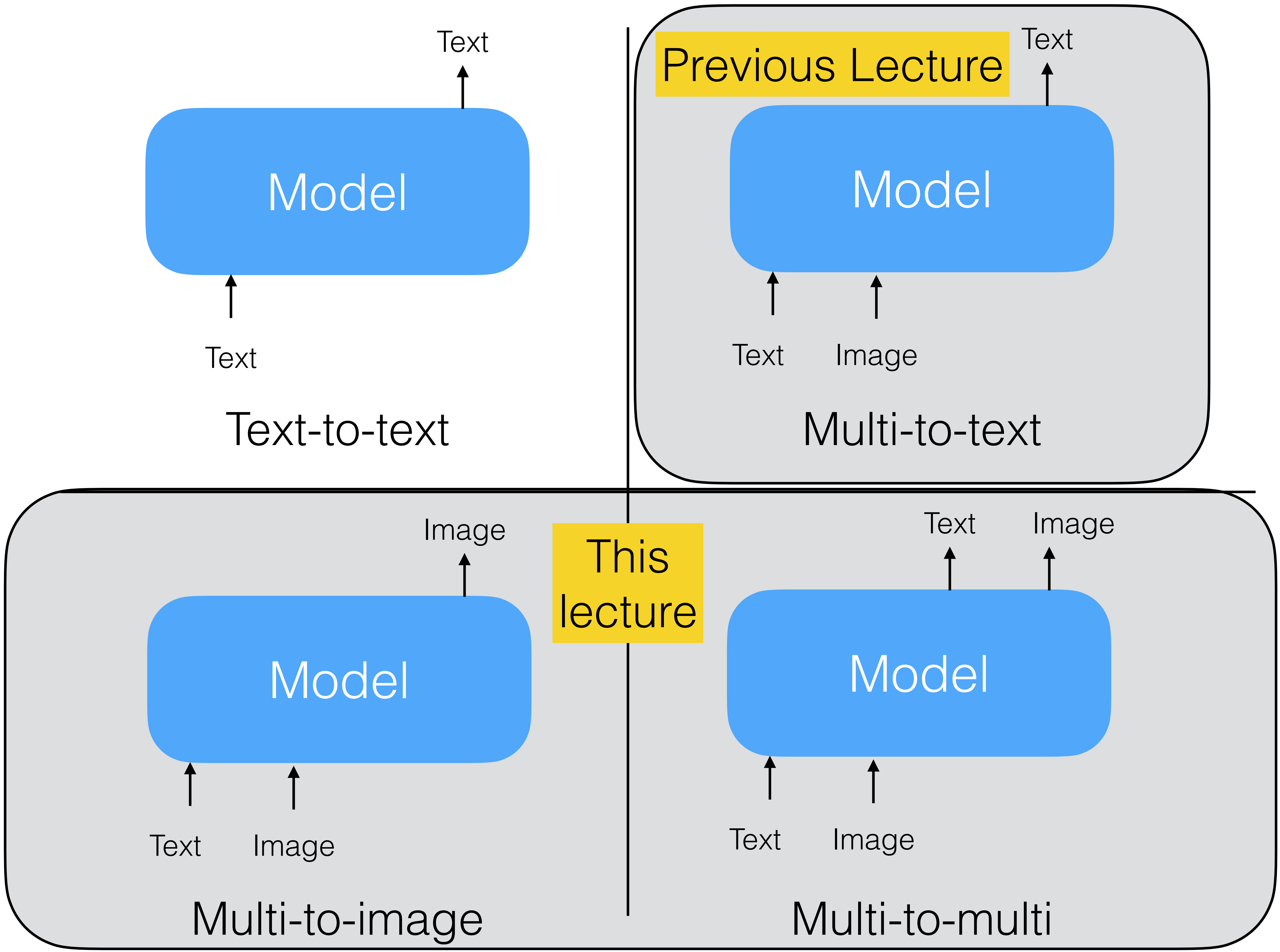
CS11-711 Advanced NLP

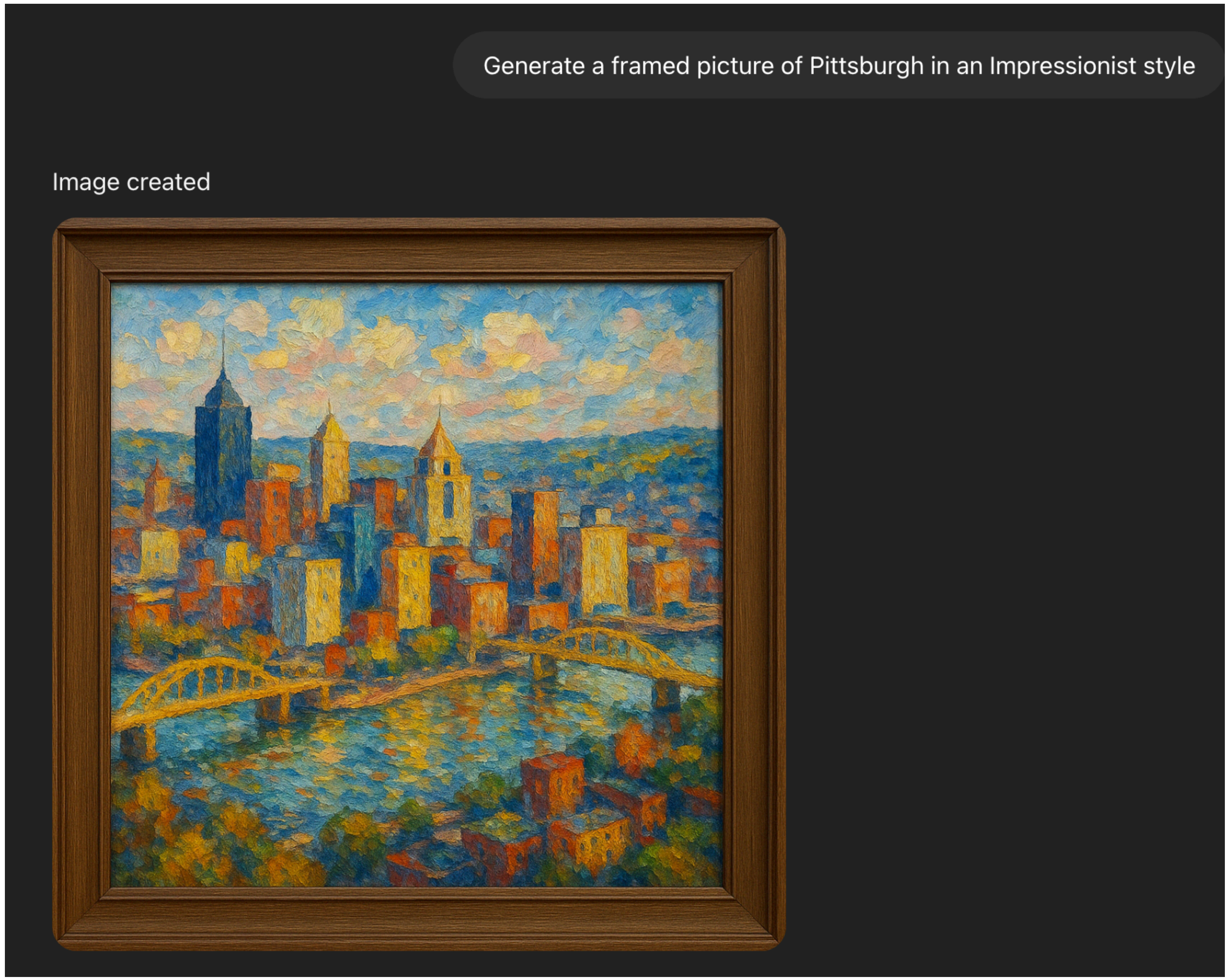
Multimodal Modeling II

Sean Welleck

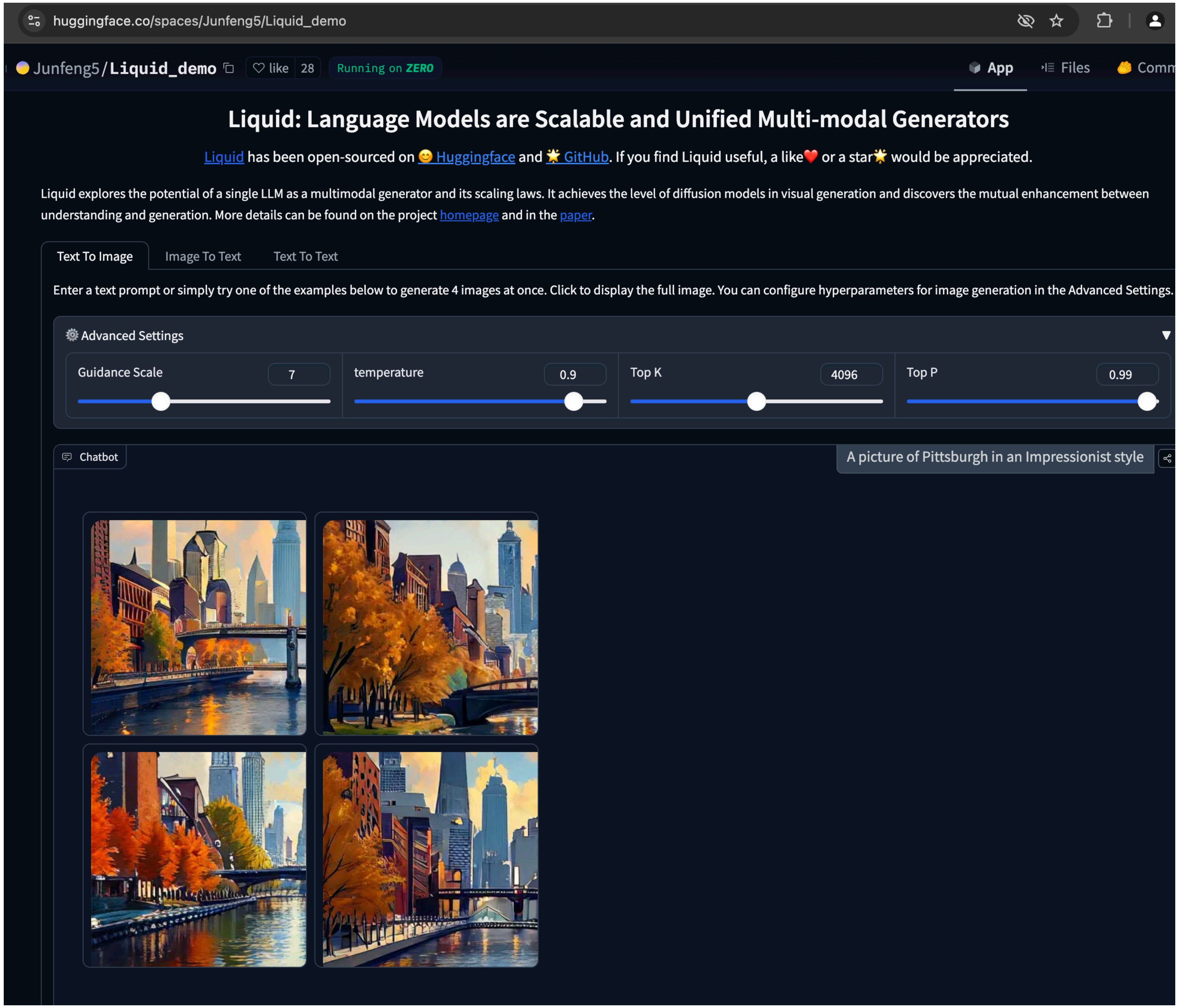
Carnegie Mellon University
Language Technologies Institute

https://cmu-l3.github.io/anlp-spring2025/


Text
Model
Text
Text-to-text
Previous Lecture
Text
Model
Text
Image
Multi-to-text
Image
Model
Text
Image
Multi-to-image
This lecture
Text
Image
Model
Text
Image
Multi-to-multi

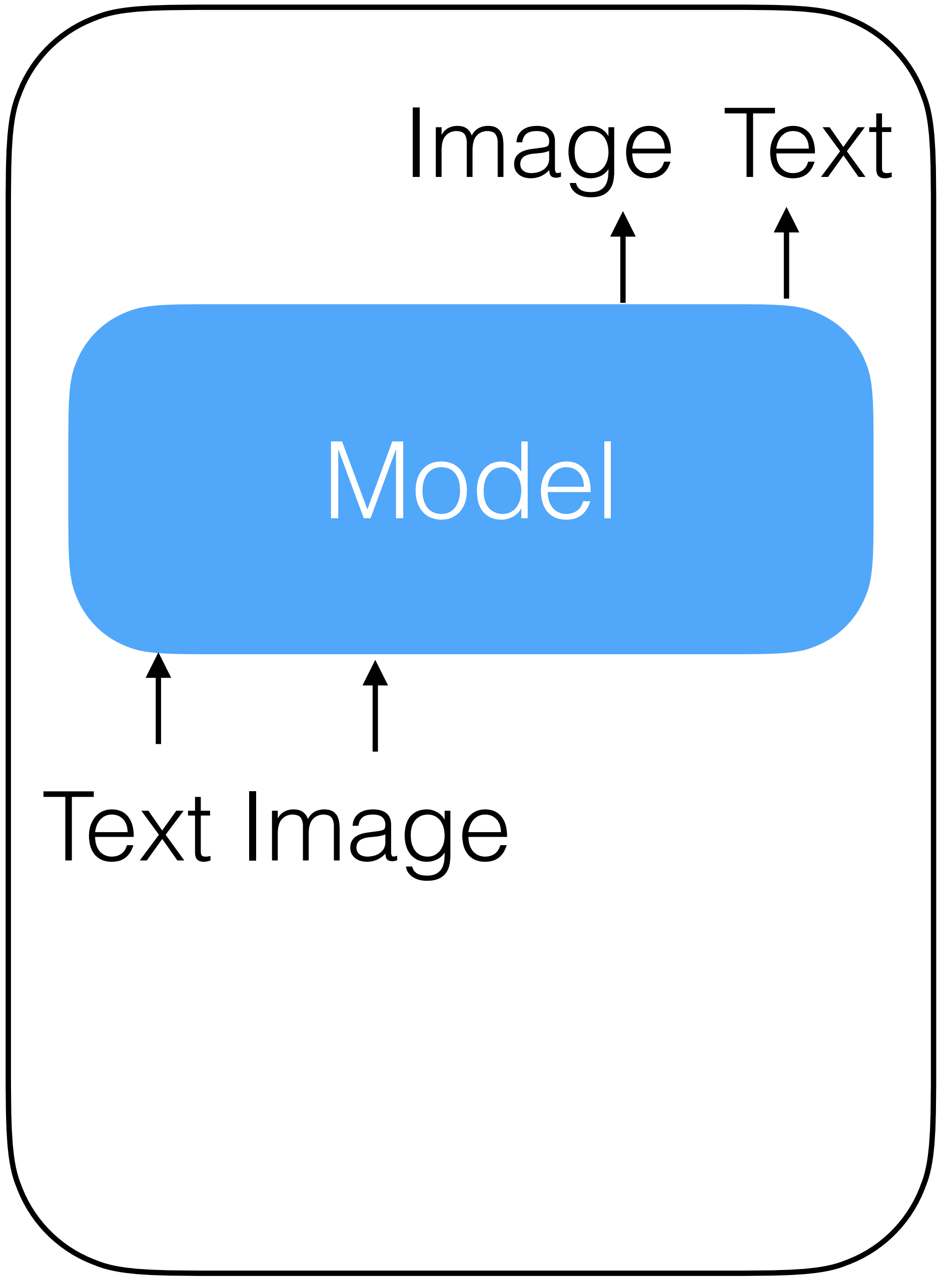


Example: ChatGPT 4o



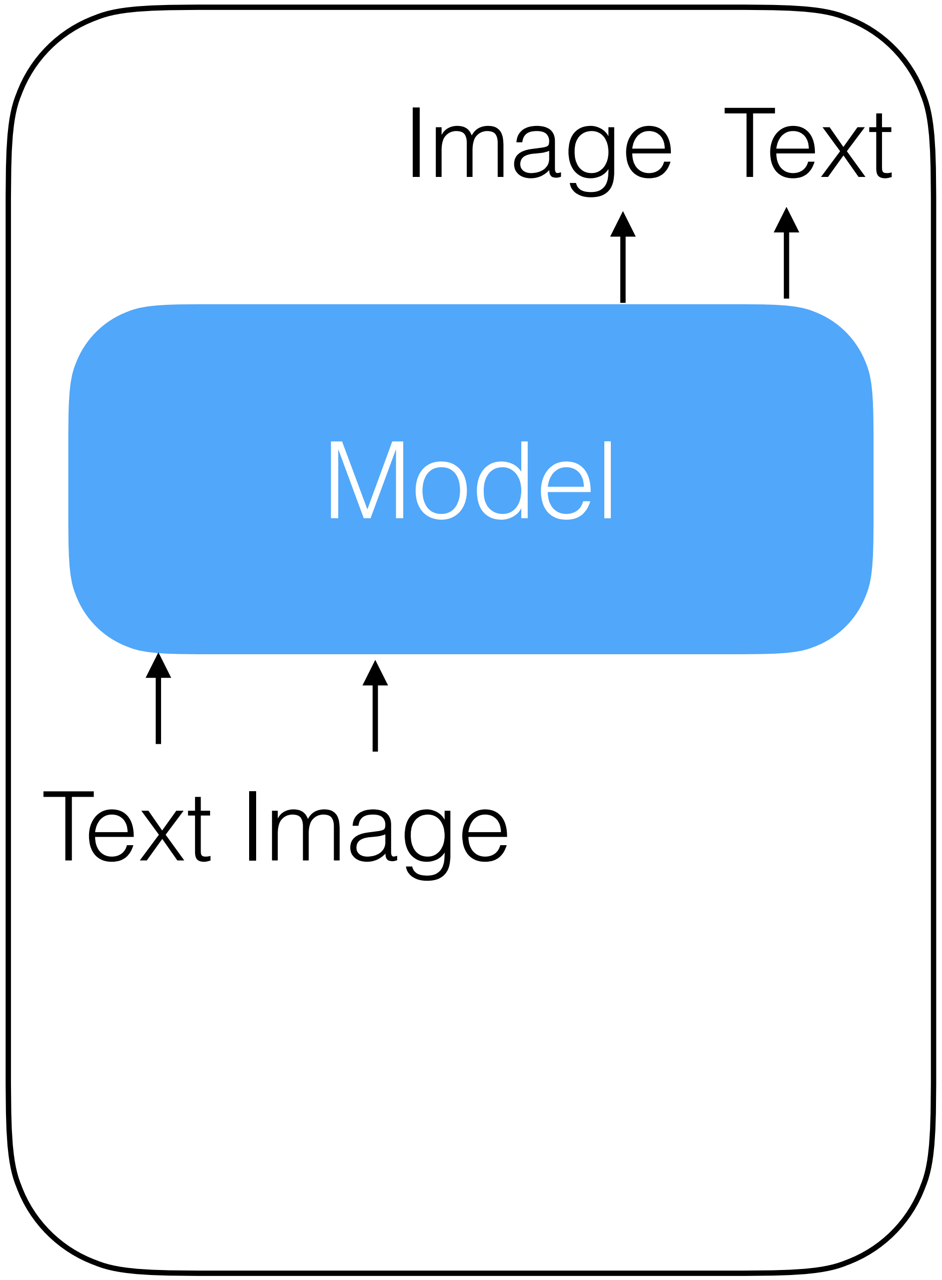
Example: Liquid

Today's lecture



Today's lecture

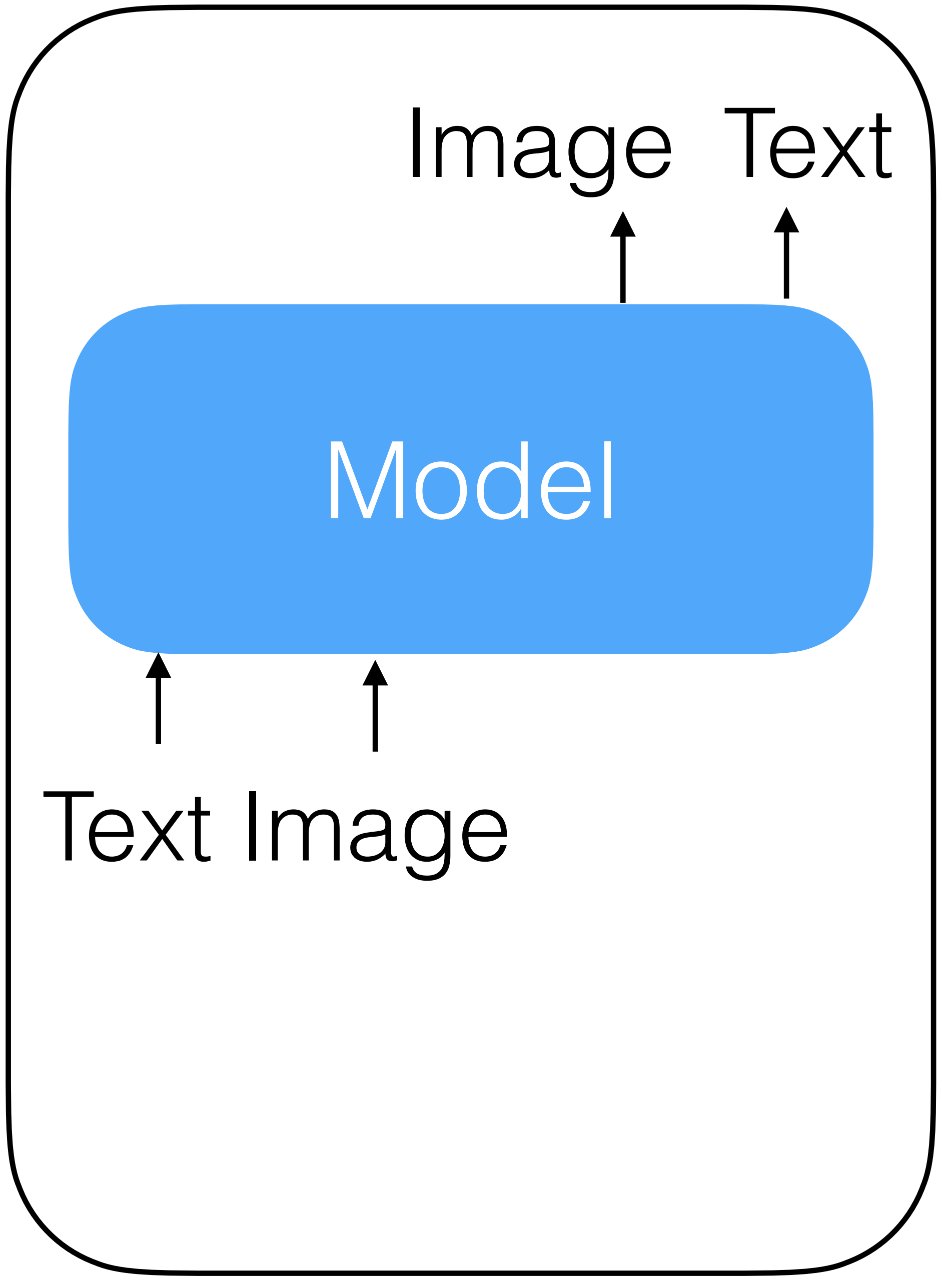
- Basic generative modeling paradigms



Today's lecture

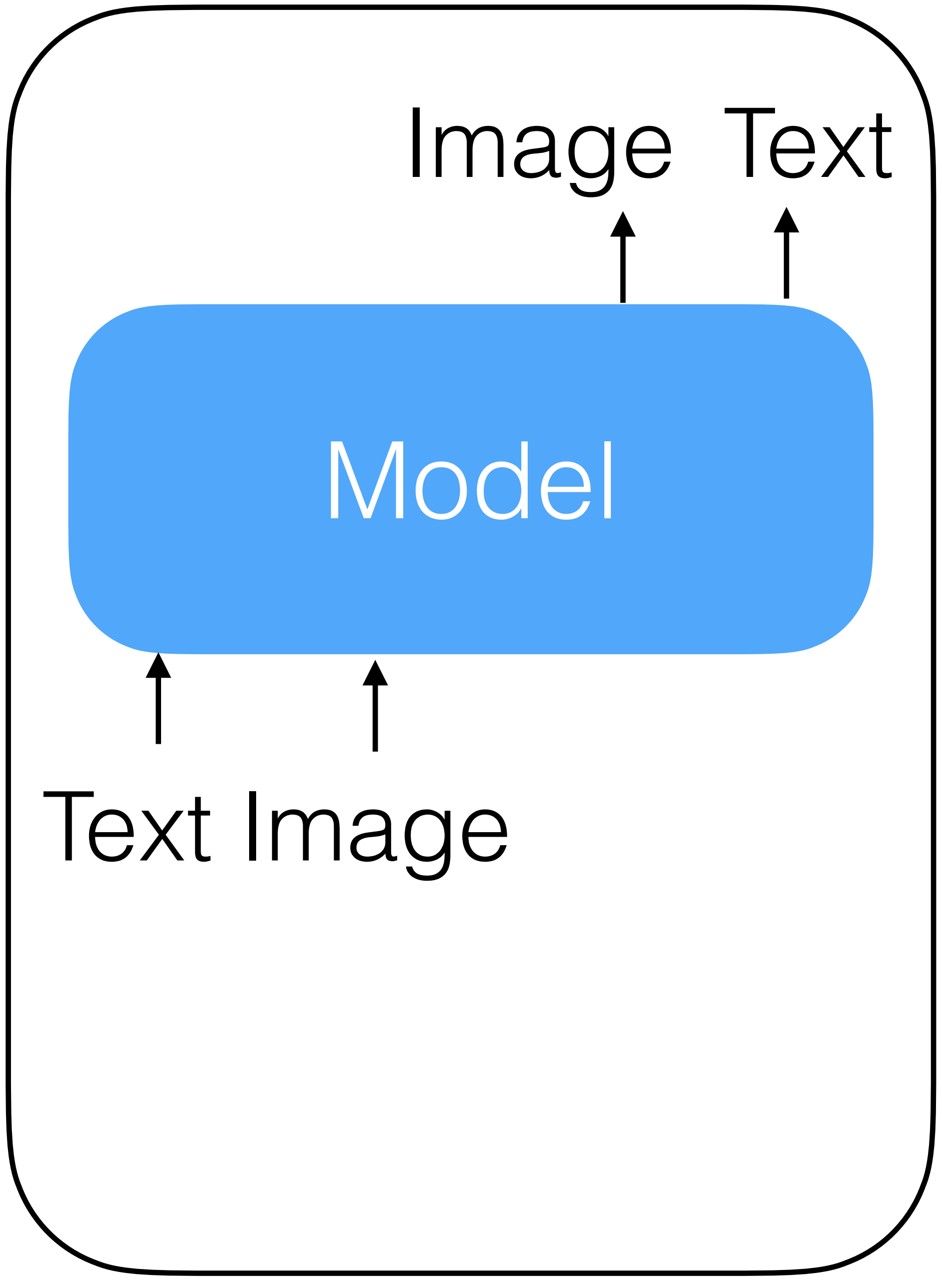
- Basic generative modeling paradigms

- Autoregressive modeling of pixels



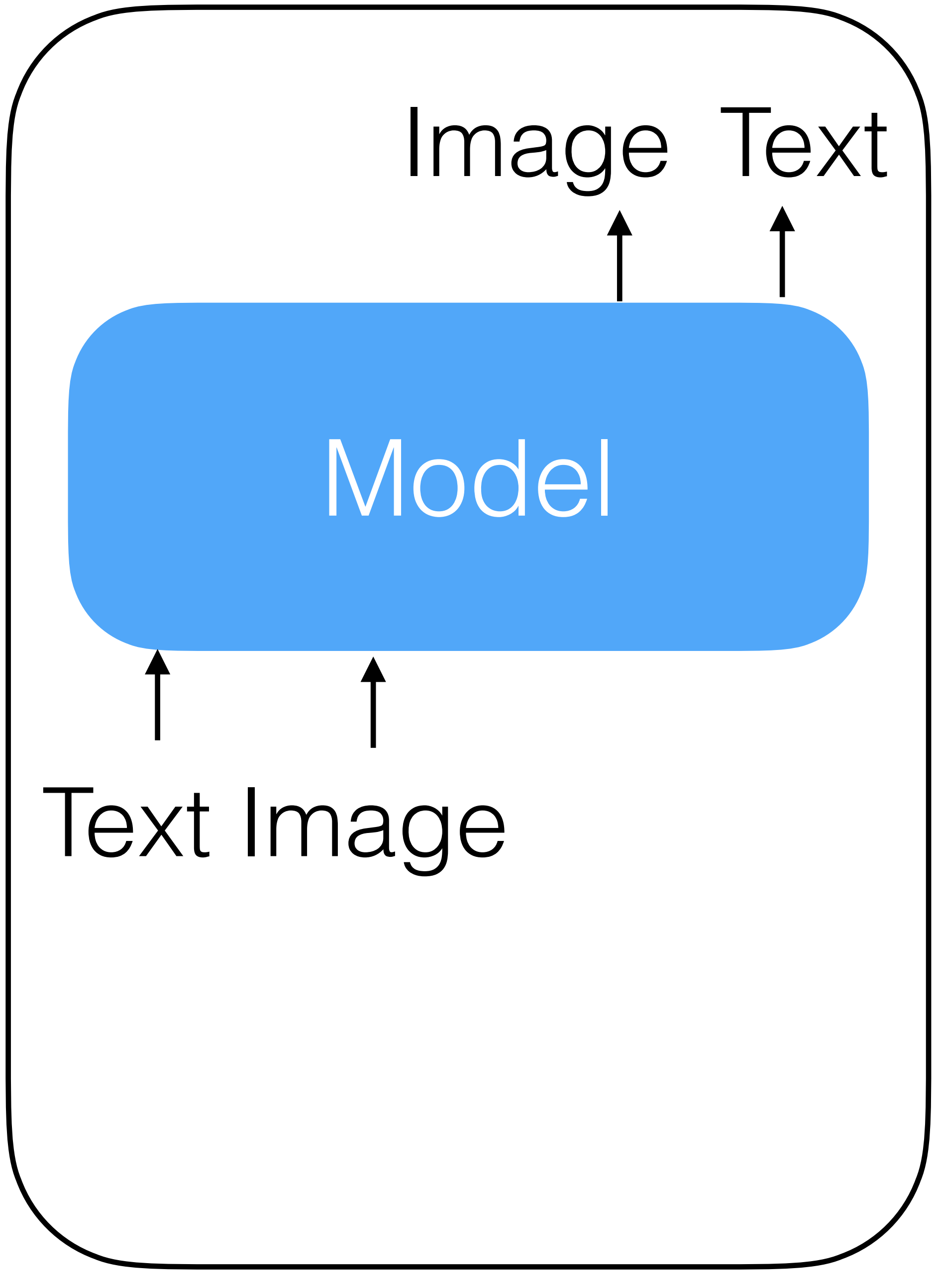
Today's lecture

- Basic generative modeling paradigms
- Autoregressive modeling of pixels
- Autoregressive modeling of "image tokens"



Today's lecture

- Basic generative modeling paradigms
- Autoregressive modeling of pixels
- Autoregressive modeling of "image tokens"
- Other paradigms

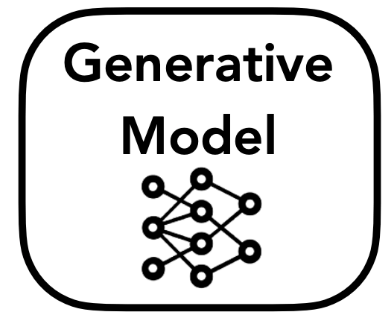


Generative modeling paradigms

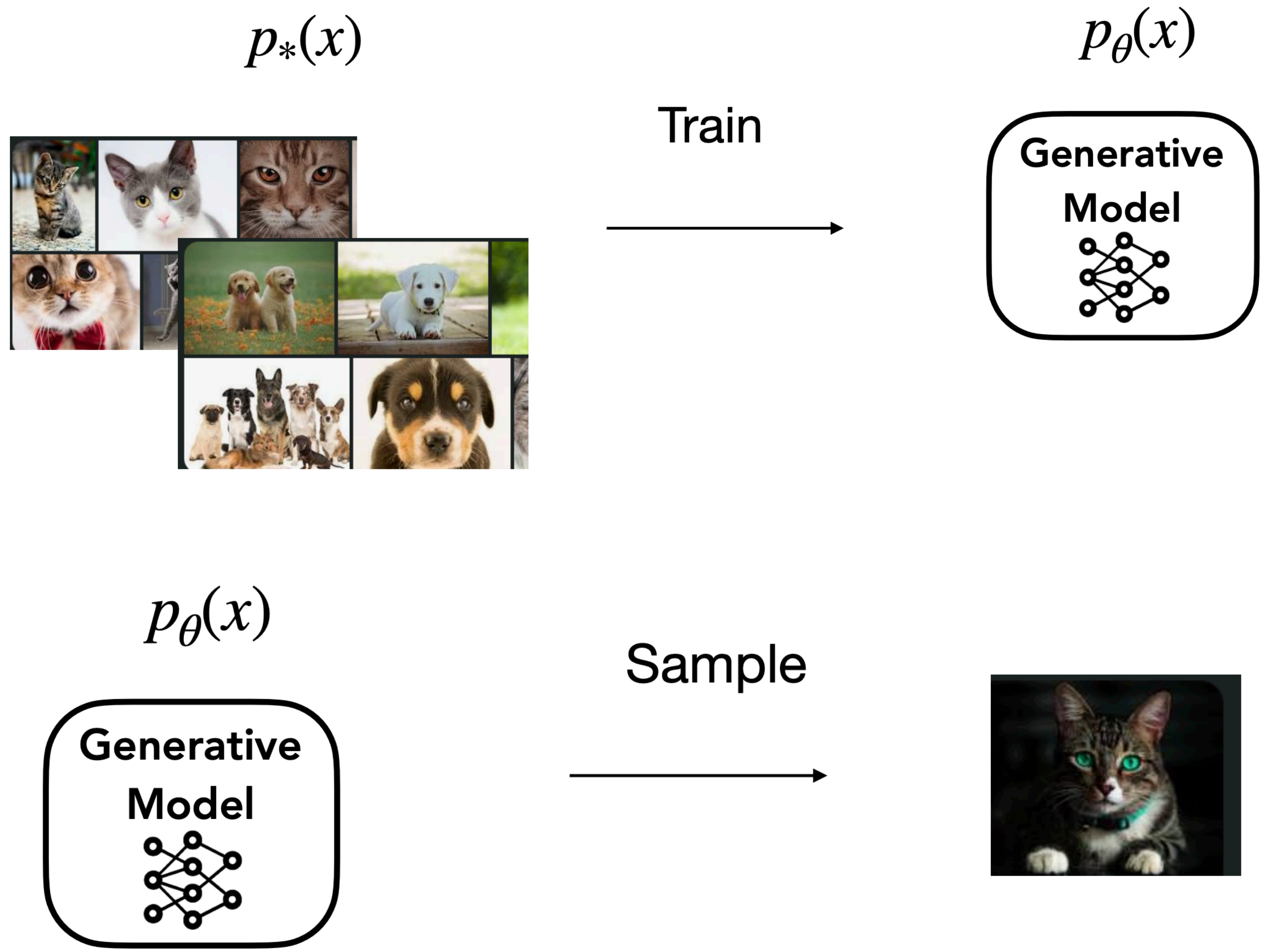
$p_*(x)$

Train

$p_\theta(x)$



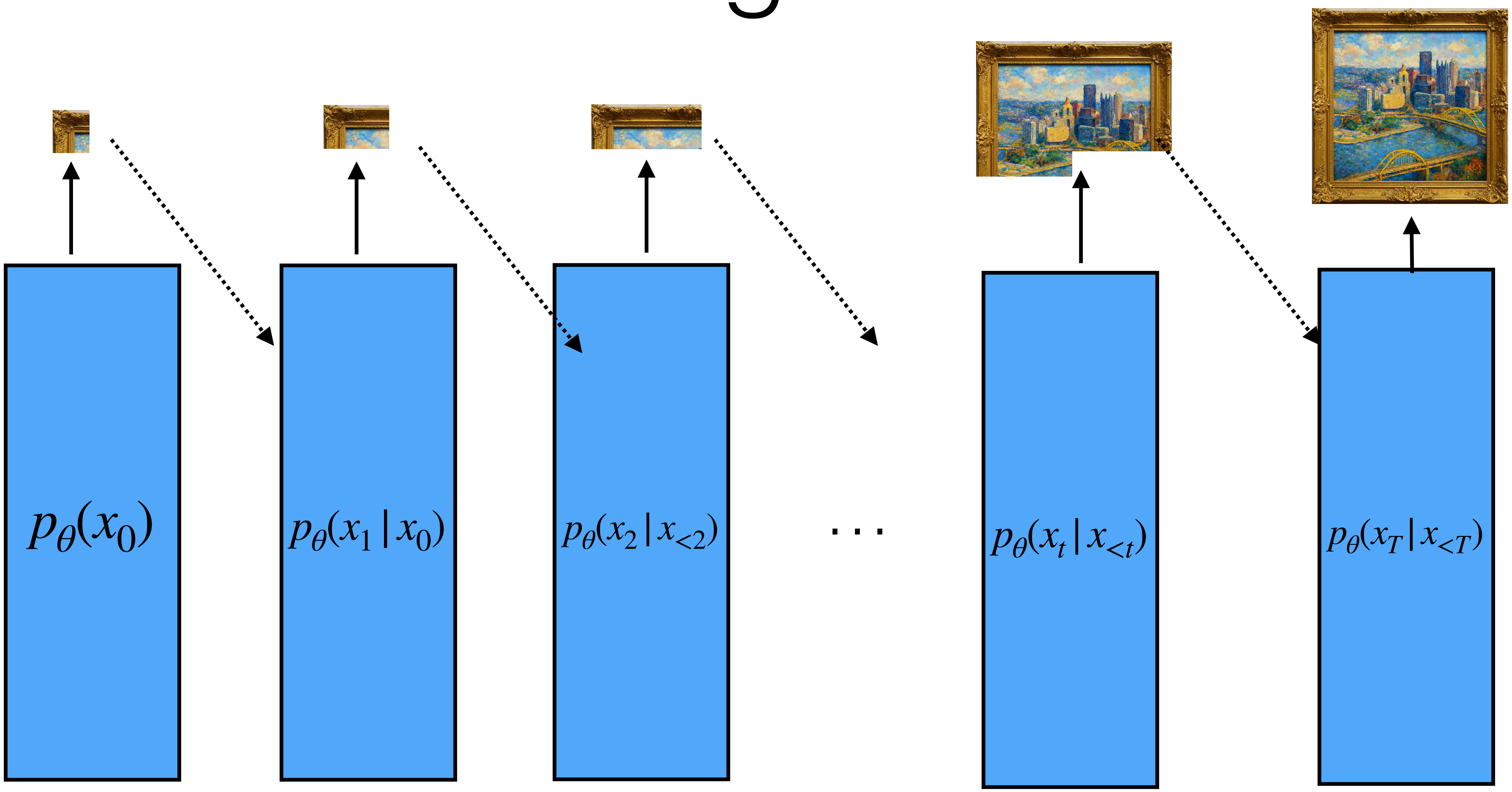
Generative modeling paradigms



Generative modeling paradigms

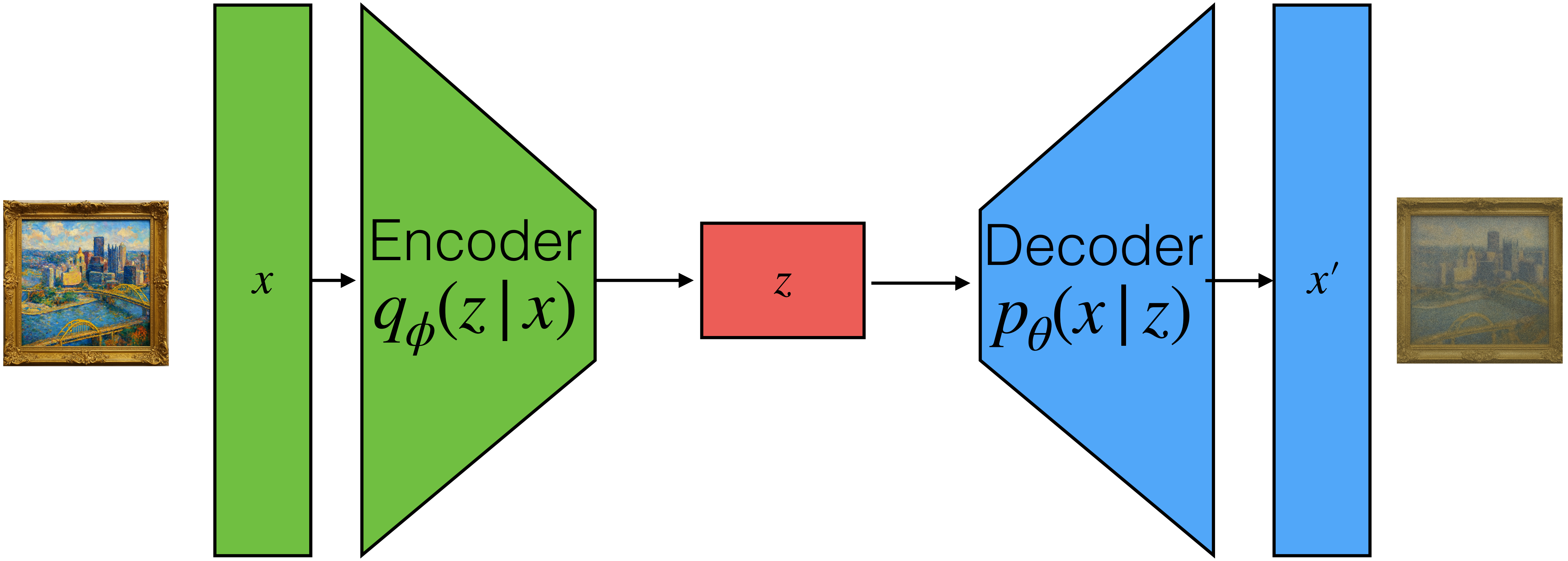
- Autoregressive
- Variational auto-encoder (VAE)
- Generative adversarial networks (GAN)
- Diffusion models

Autoregressive



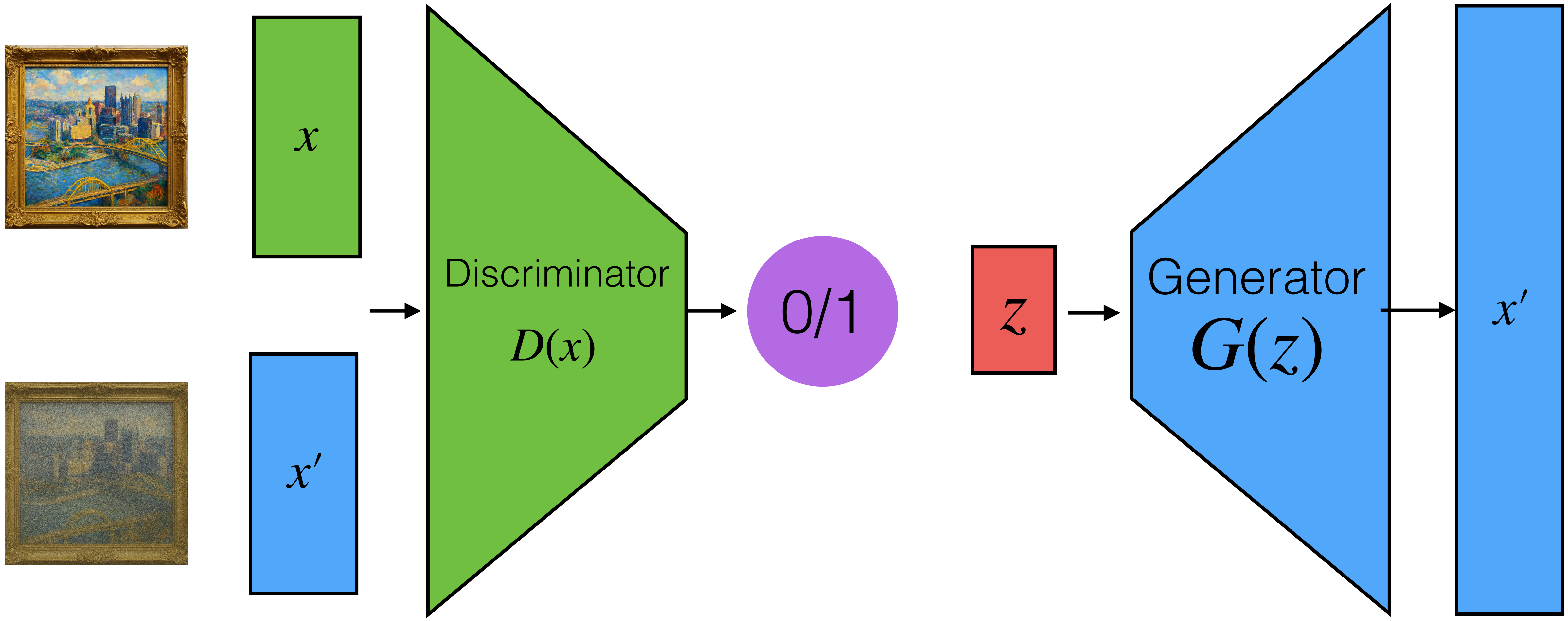
- Generate one dimension (e.g., token, pixel) at a time given the previous ones

Variational Auto-encoders (VAE)



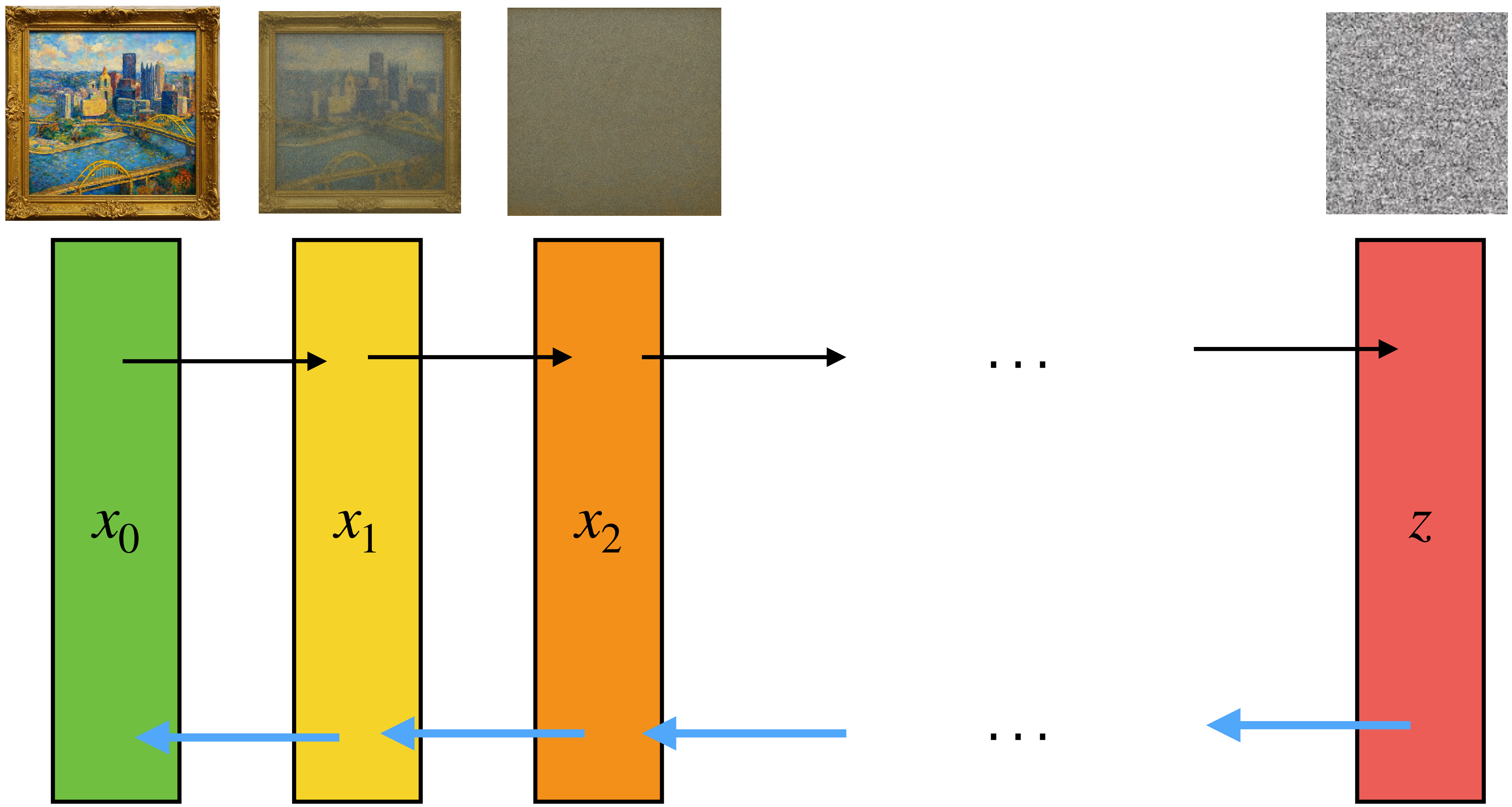
- Learn to reconstruct the image by compressing it into a vector and uncompressing it

Generative adversarial networks (GAN)



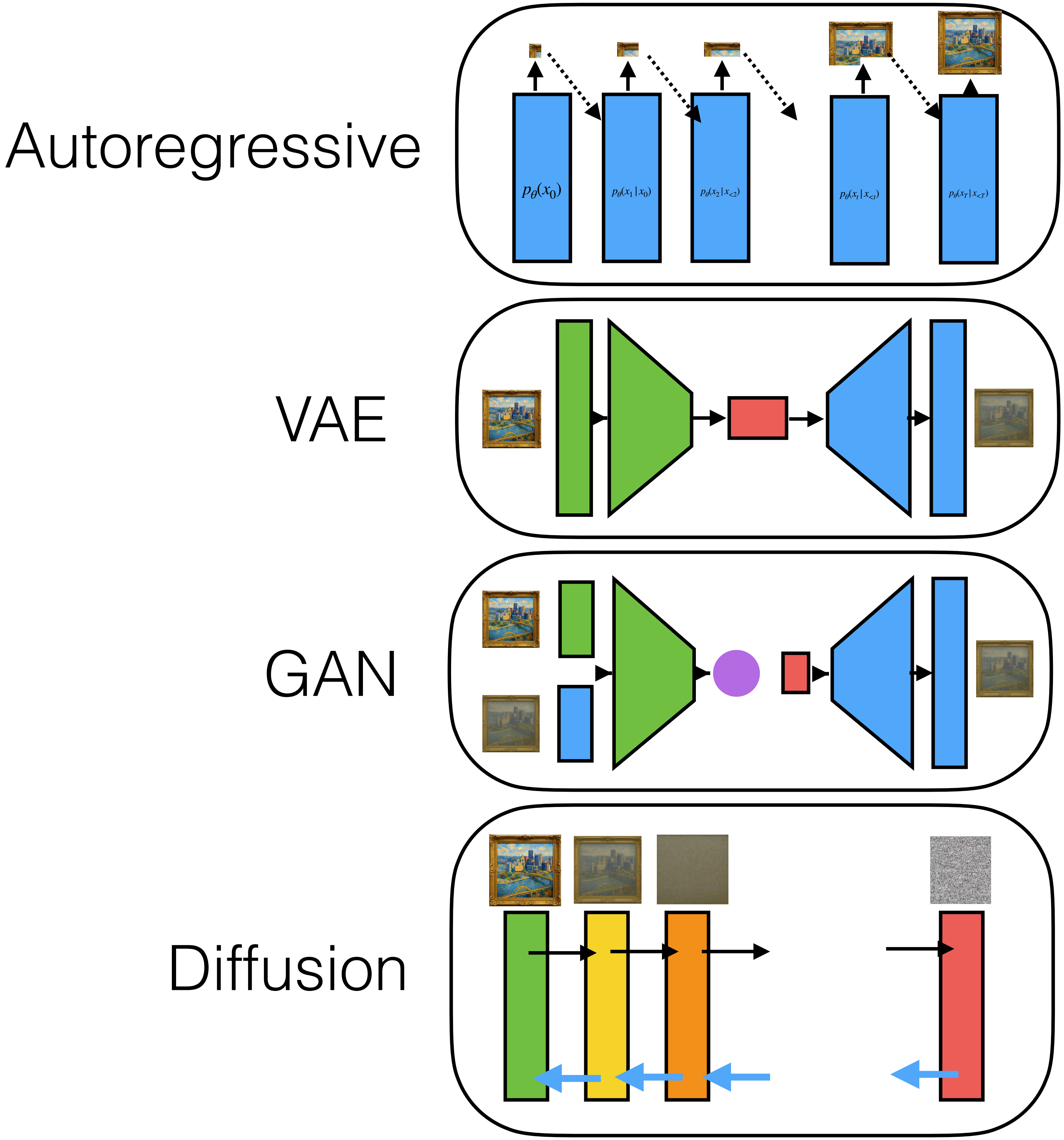
- Learn to generate an image by fooling a discriminator that detects whether an image is fake

Diffusion models

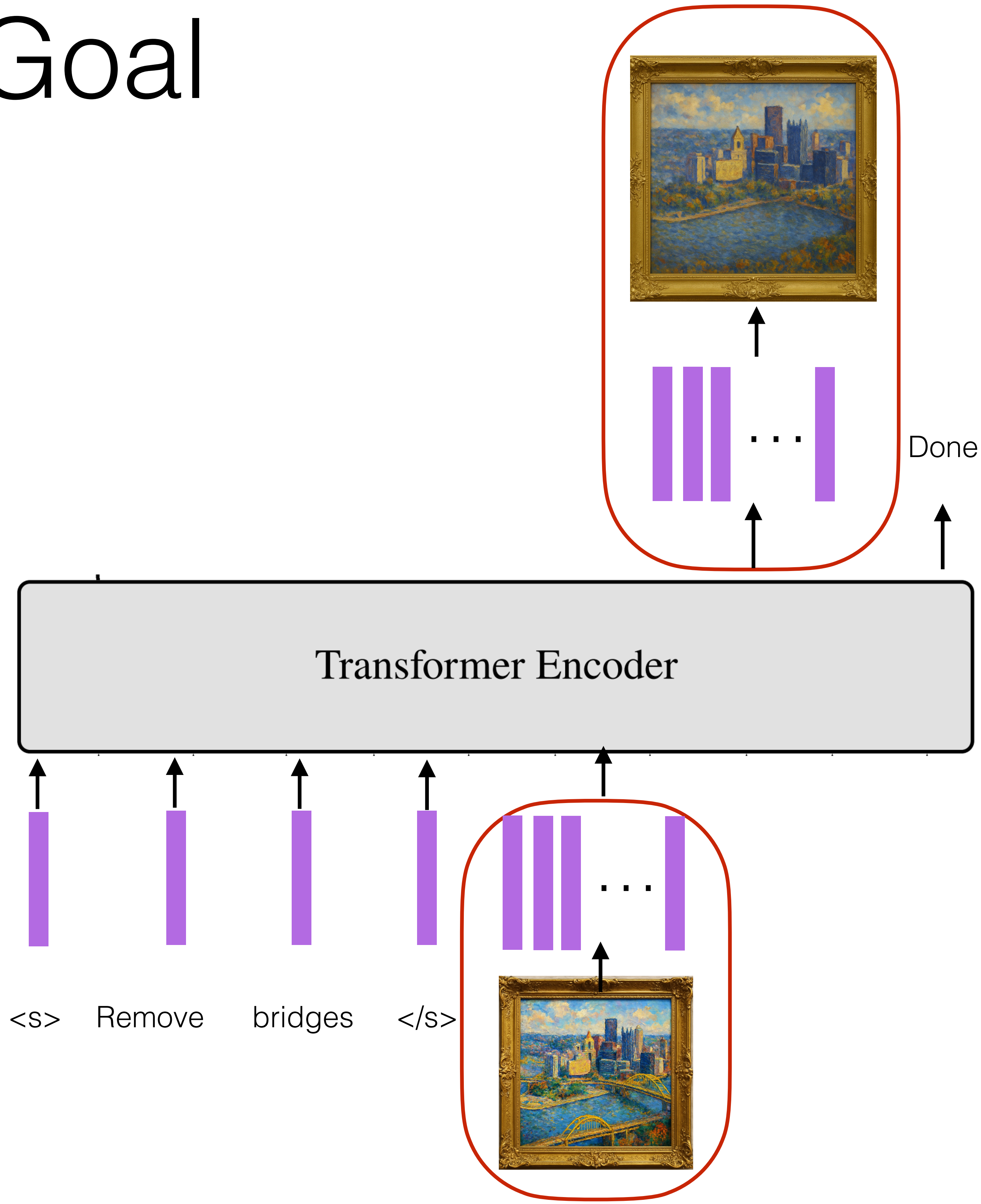


- Gradually add noise to an image, then learn to de-noise

Generative modeling paradigms

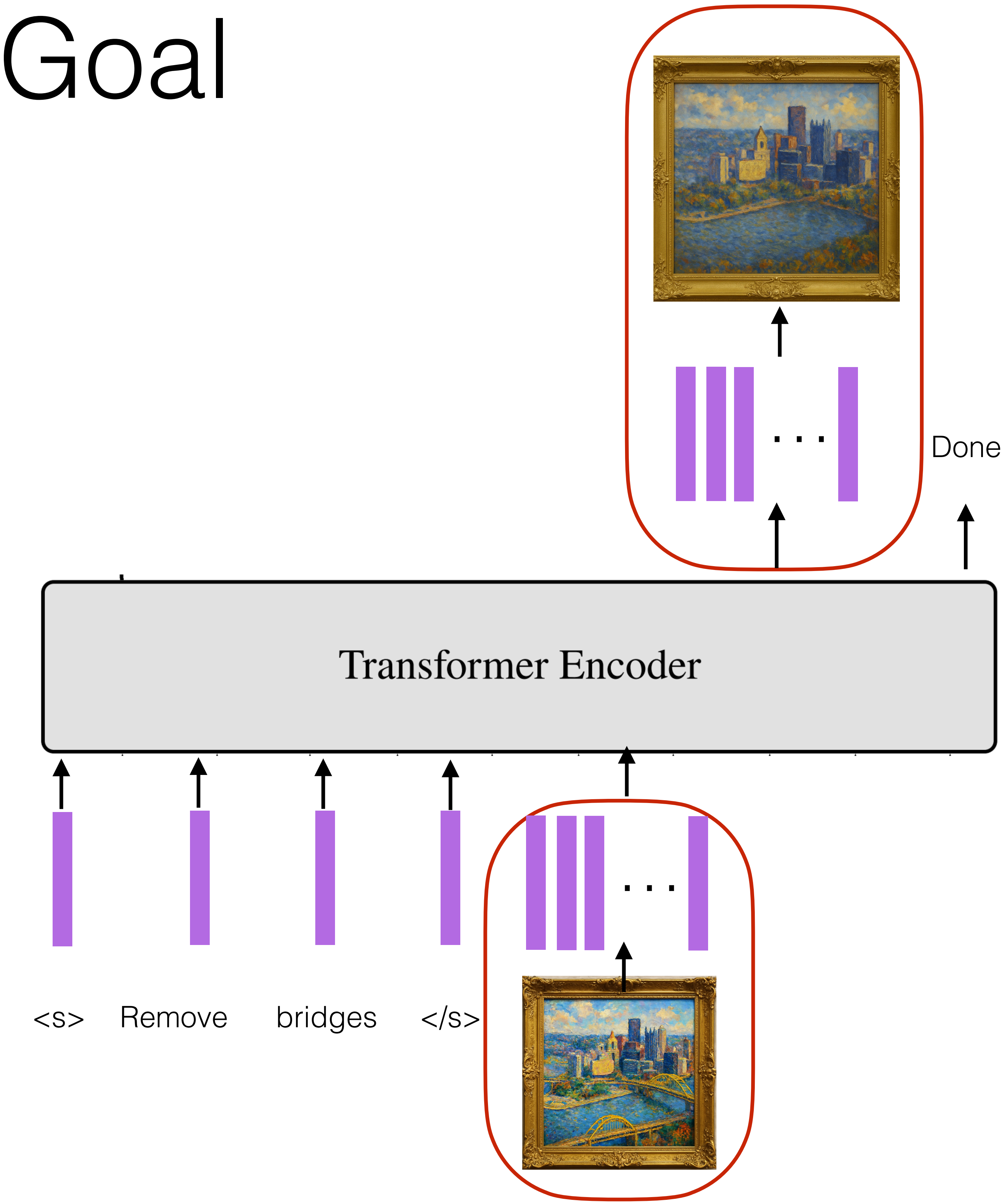


Goal



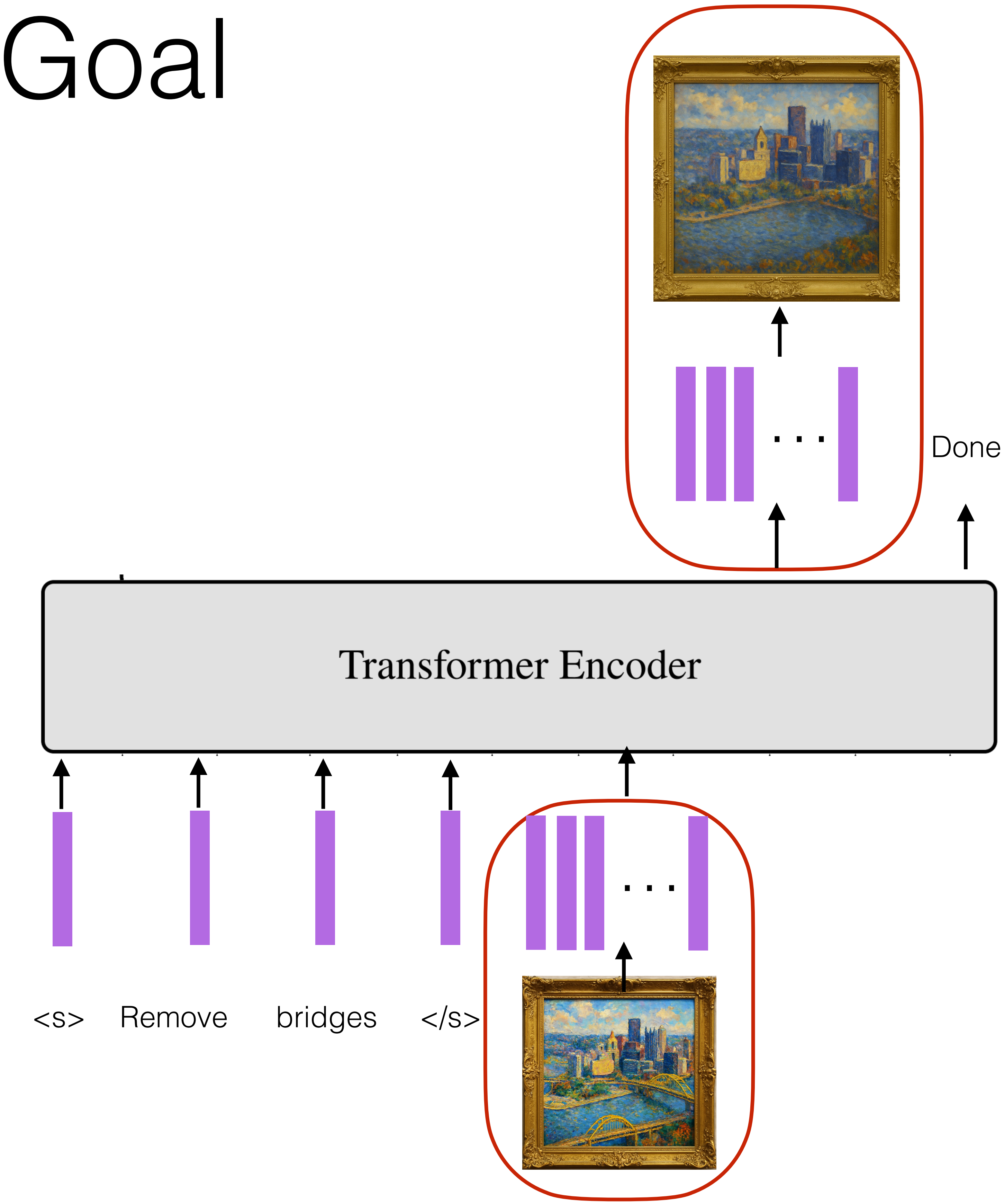
Goal

- Image tokenizer



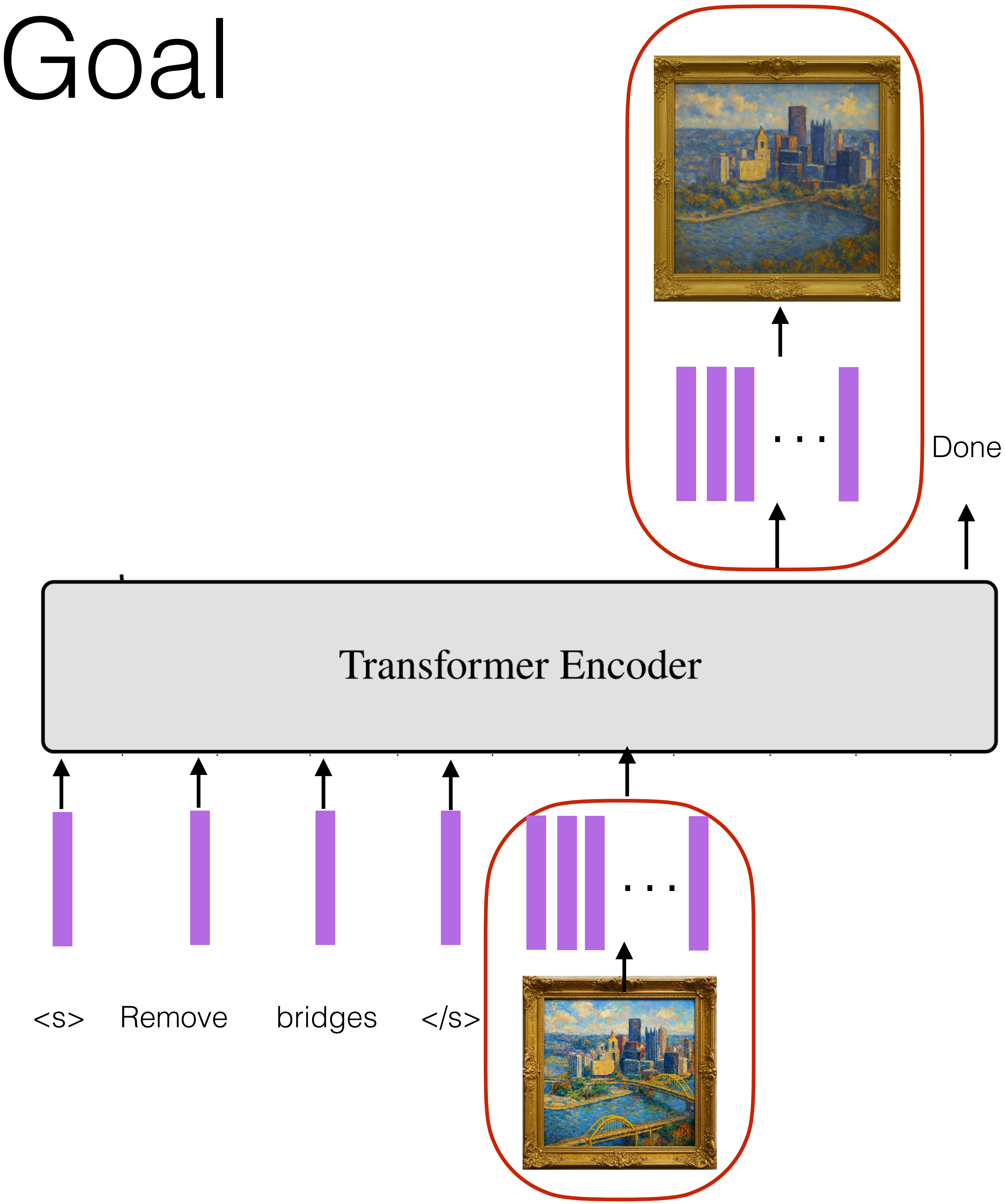
Goal

- Image tokenizer
- Image “de-tokenizer”



Goal

- Image tokenizer
- Image "de-tokenizer"
- Single Transformer for text and image



Roadmap

Roadmap

- **Attempt 1**: use a standard auto-regressive model
  - Tokenizer: individual pixels
  - De-tokenizer: generate individual pixels, no de-tokenizer needed

Roadmap

- **Attempt 1**: use a standard auto-regressive model
  - Tokenizer: individual pixels
  - De-tokenizer: generate individual pixels, no de-tokenizer needed
- **Attempt 2**: learn a discrete tokenizer / de-tokenizer
  - Vector-quantized VAE (VQ-VAE)
  - VQ-GAN

Auto-regressive model

Auto-regressive model

- **Basic idea**: treat an image as a sequence of pixels
  - Learn a language model over the pixel sequences

Auto-regressive model

- **Basic idea**: treat an image as a sequence of pixels
    - Learn a language model over the pixel sequences
- $x_{\text{img}} \in \mathbb{R}^{H \times W \times C} \rightarrow x_1, \ldots, x_T$
    - $x_t \in \{1,2,\ldots,256\}$

Auto-regressive model

- **Basic idea**: treat an image as a sequence of pixels
  - Learn a language model over the pixel sequences
- $x_{\text{img}} \in \mathbb{R}^{H \times W \times C} \to x_1, \ldots, x_T$
  - $x_t \in \{1,2,\ldots,256\}$
- Given a dataset of sequences:
  - $\mathscr{L}_{MLE} = \sum_{t=1}^{T} -\log p_\theta(x_t \mid x_{<t})$

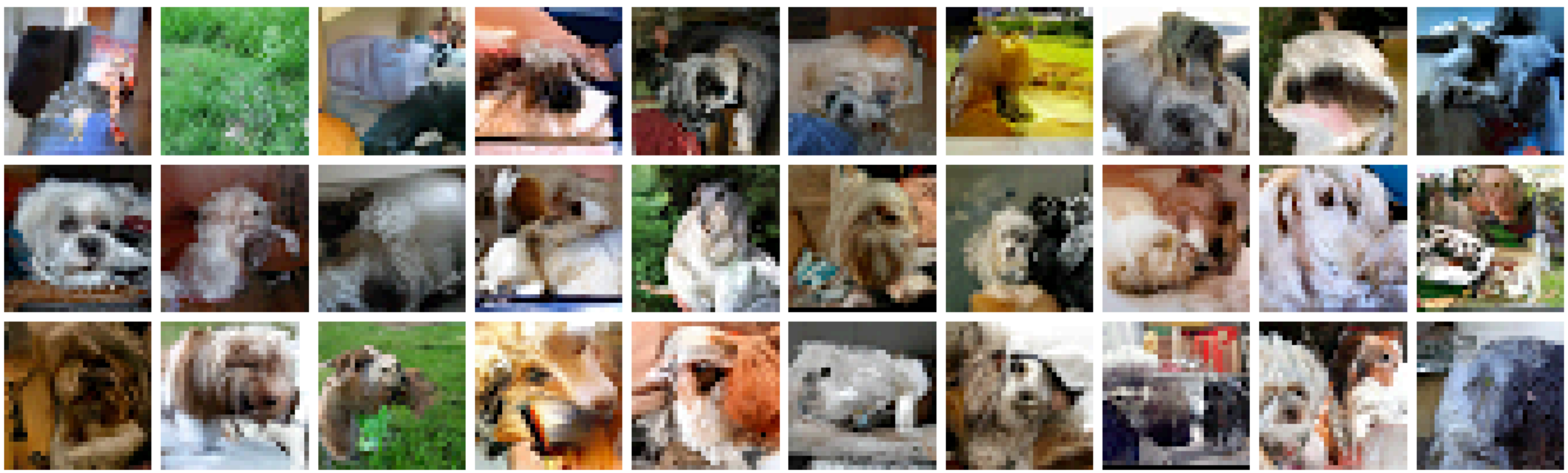
Example: PixelRNN

[van den Oord 2016]

Figure 1. Image completions sampled from a PixelRNN.

Example: PixelCNN

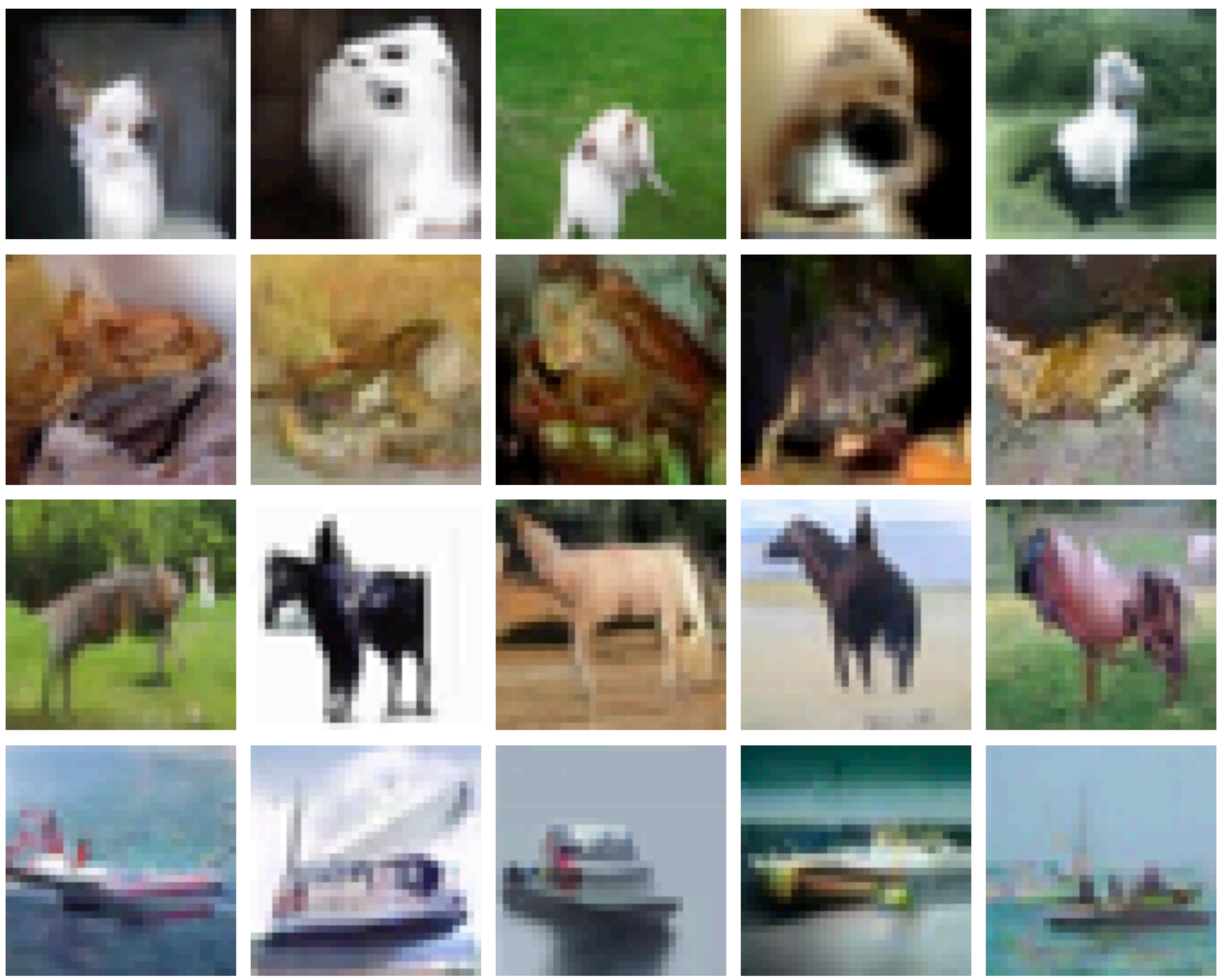
[van den Oord 2016]



Lhasa Apso (dog)

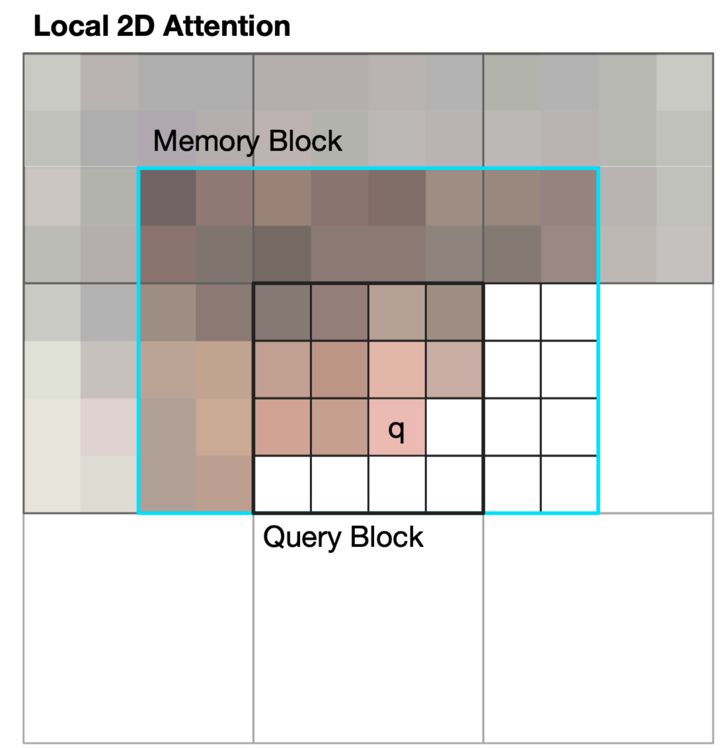
Example: Image Transformer

[Parmar et al 2018]



Auto-regressive model

- **Key challenge**: sequence length
  - 1024 x 1024 image with 3 channels (RGB)
    - 3 million tokens



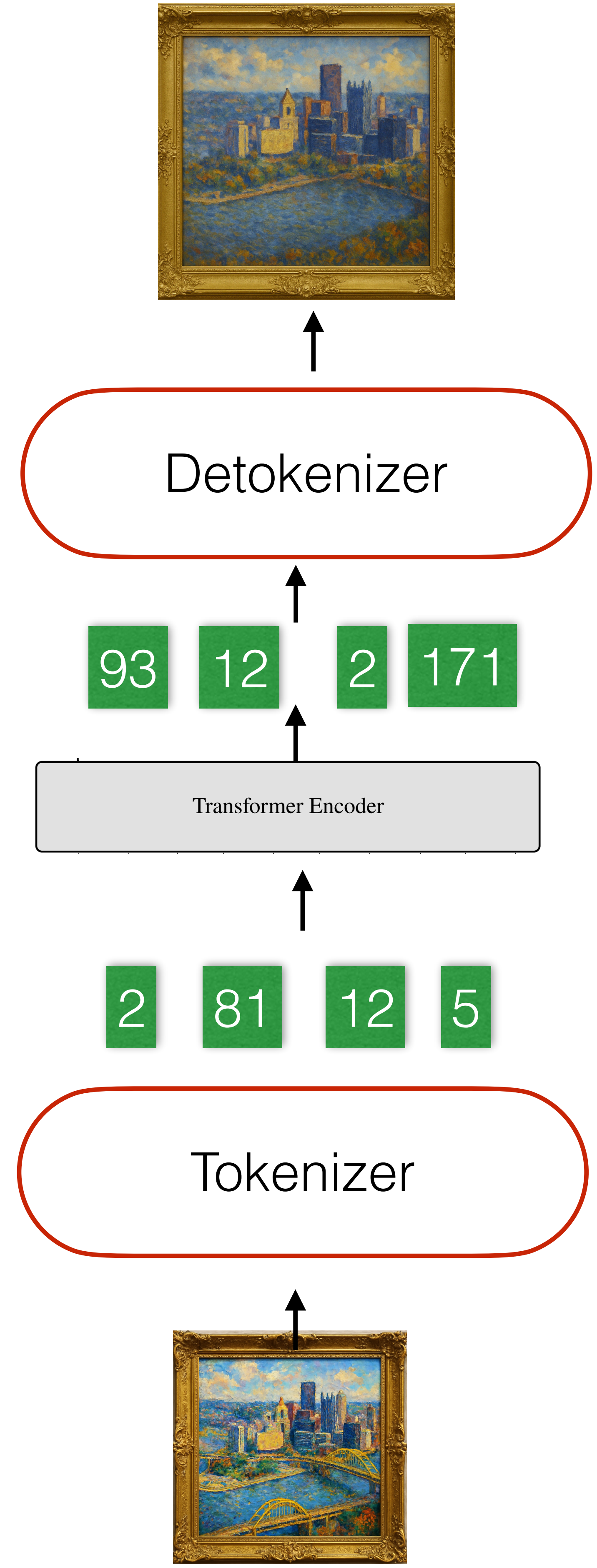
Typically involved alternative memory-efficient attention patterns

Roadmap

- Attempt 1: use a standard auto-regressive model
- **Attempt 2**: learn a discrete tokenizer / de-tokenizer
  - Vector-quantized VAE (VQ-VAE)
  - VQ-GAN

Idea

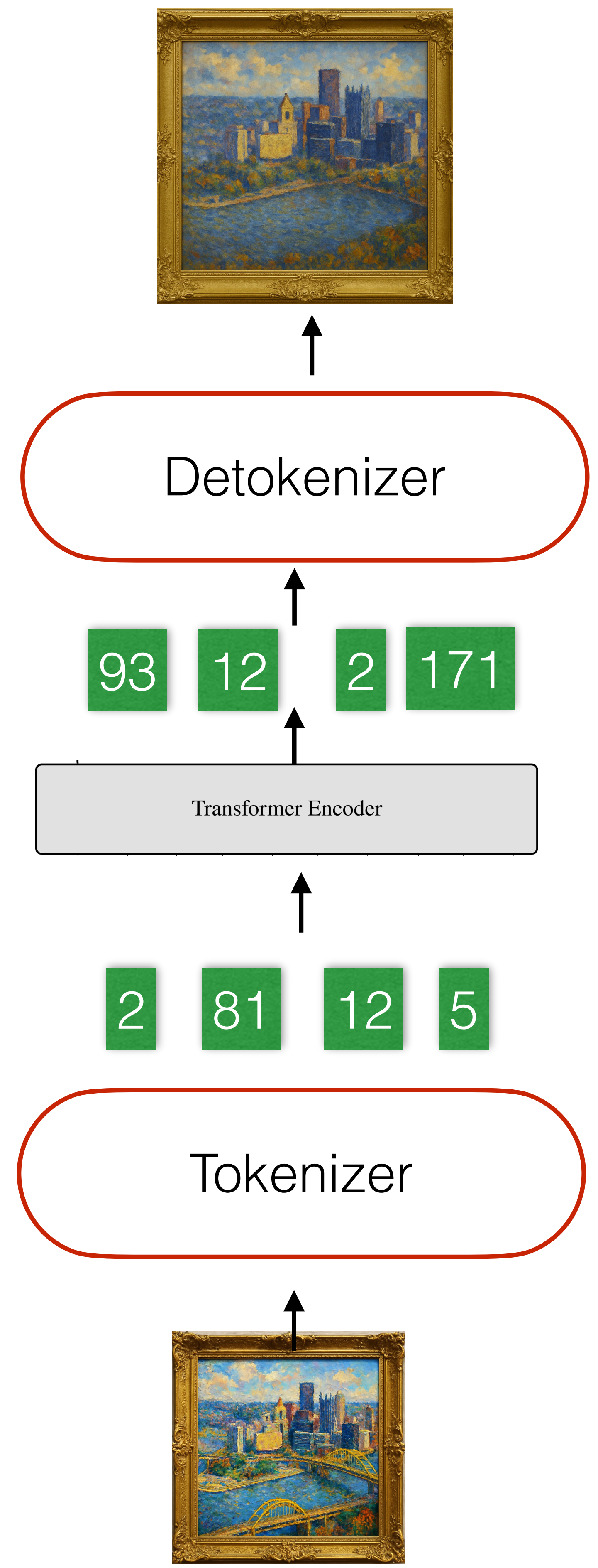
- Learn a tokenizer that compactly represents an image as a sequence of discrete tokens.



Idea

- Learn a tokenizer that compactly represents an image as a sequence of discrete tokens.

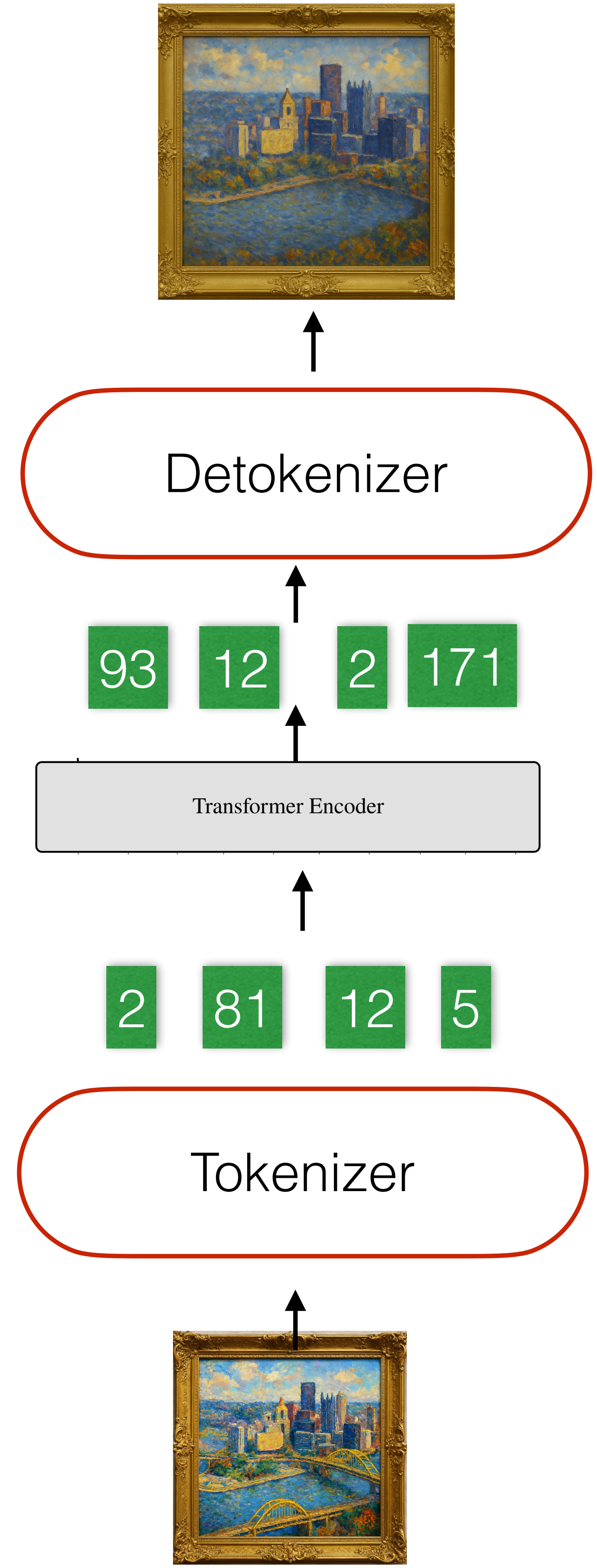
- Example:



Idea

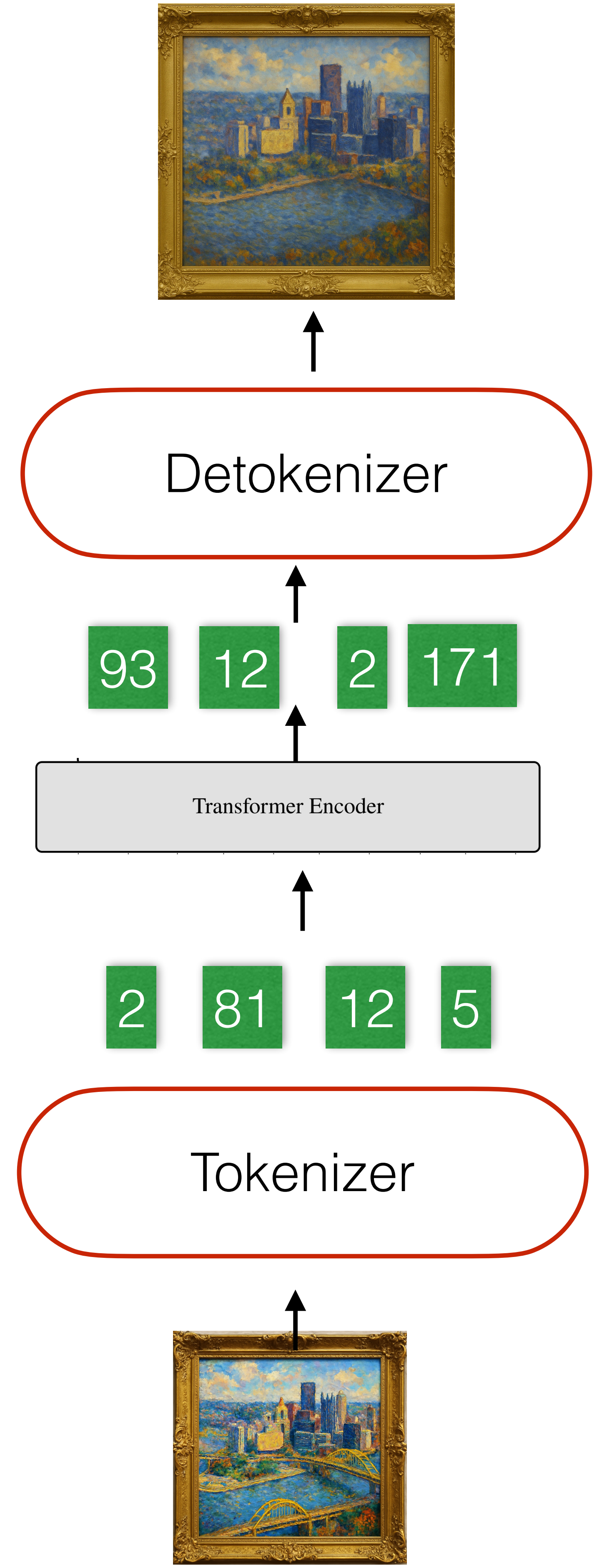
- Learn a tokenizer that compactly represents an image as a sequence of discrete tokens.

- Example:
  - Vocabulary: 8,000 token ids



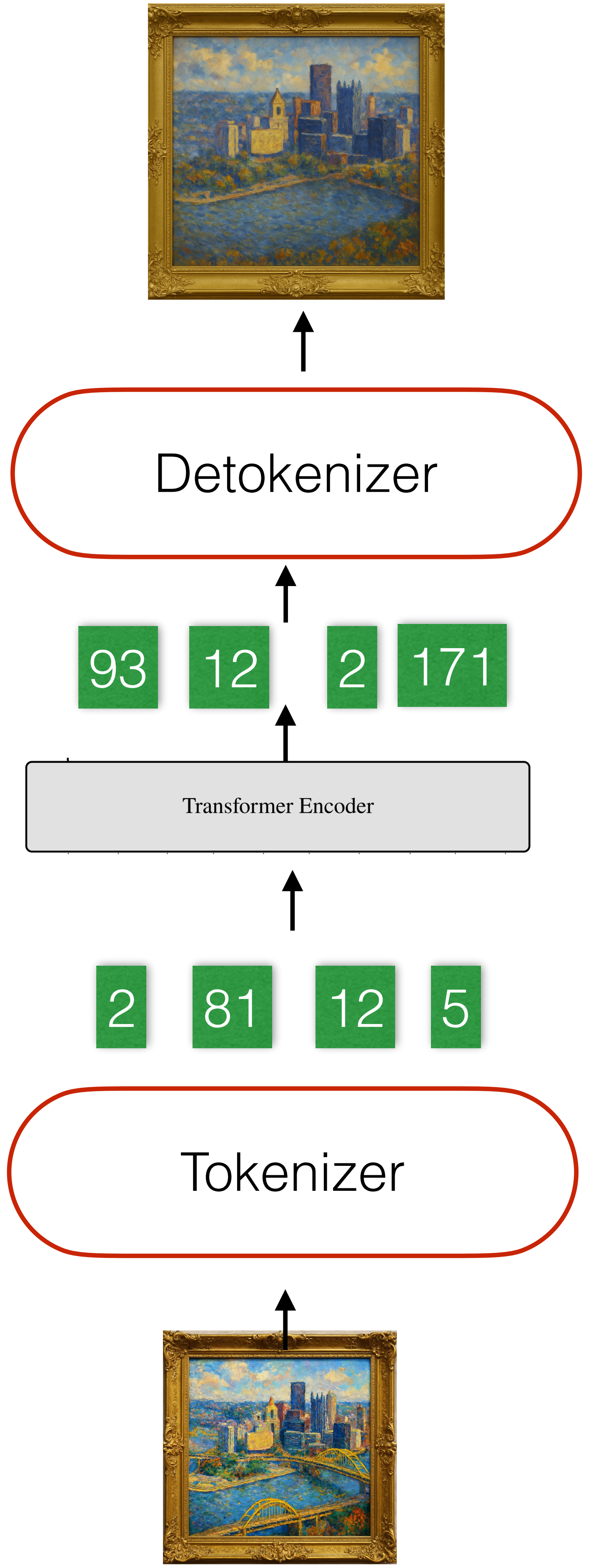
Idea

- Learn a tokenizer that compactly represents an image as a sequence of discrete tokens.
- Example:
    - Vocabulary: 8,000 token ids
    - Image: sequence of 1024 tokens

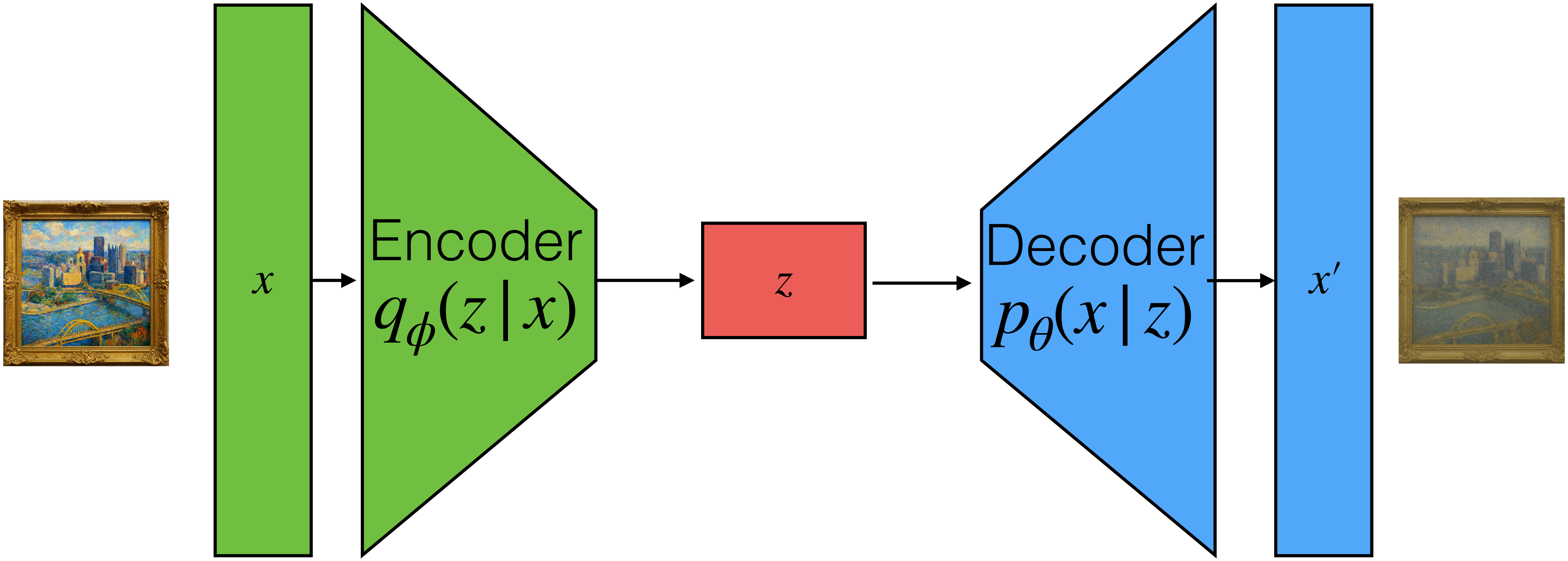


Idea

- Learn a tokenizer that compactly represents an image as a sequence of discrete tokens.
- Example:
    - Vocabulary: 8,000 token ids
    - Image: sequence of 1024 tokens
- Approach: VQ-VAE

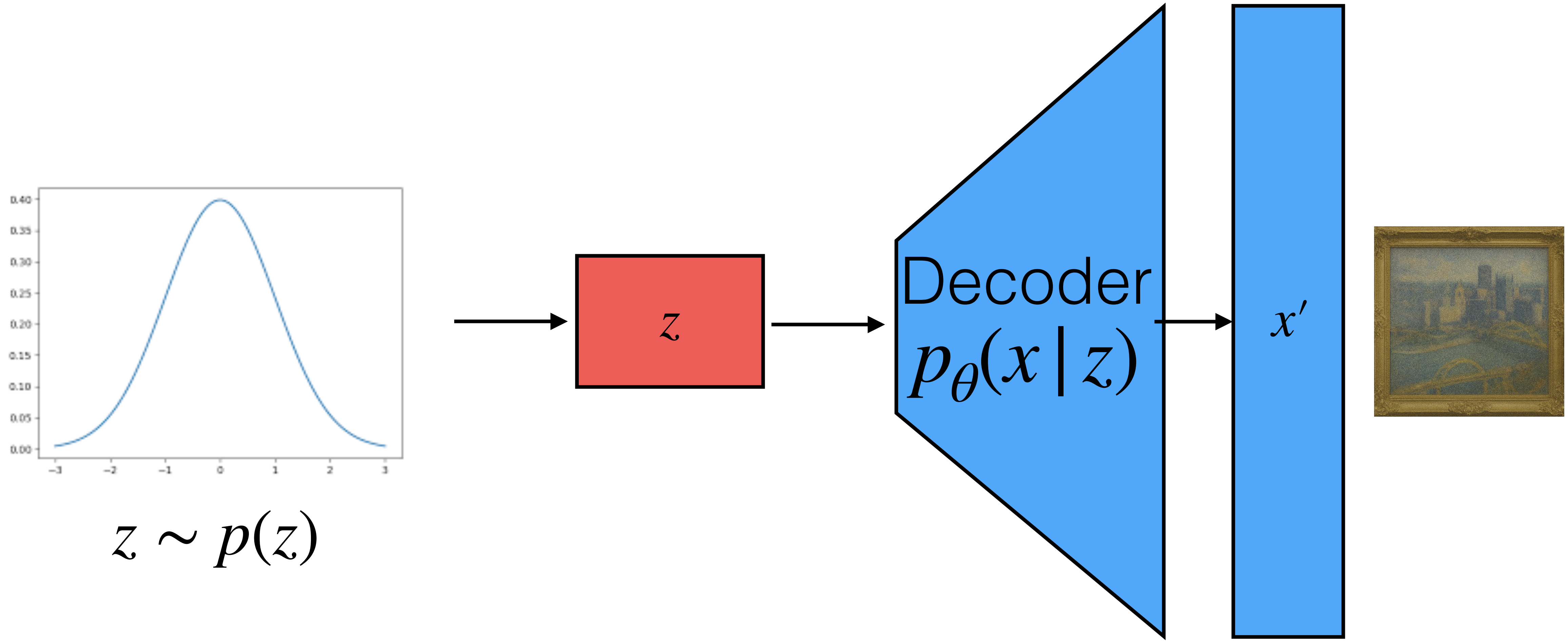


VAE



- Learn to reconstruct the image by compressing it into a vector and uncompressing it

VAE



- After training, generate by sampling a latent vector from the prior and passing it to the decoder

Standard VAE

[Kingma & Welling 2013]

Standard VAE

[Kingma & Welling 2013]

- **Goal**: learn to compress data (like images) into a low-dimensional representation z, and then reconstruct or generate new data with it

Standard VAE

[Kingma & Welling 2013]

- **Goal**: learn to compress data (like images) into a low-dimensional representation z, and then reconstruct or generate new data with it

- Assume that data is generated as follows:

Standard VAE

[Kingma & Welling 2013]

- **Goal**: learn to compress data (like images) into a low-dimensional representation z, and then reconstruct or generate new data with it

- Assume that data is generated as follows:

  - First sample a latent variable $z \sim p(z)$

Standard VAE

[Kingma & Welling 2013]

- **Goal**: learn to compress data (like images) into a low-dimensional representation z, and then reconstruct or generate new data with it
- Assume that data is generated as follows:
    - First sample a latent variable $z \sim p(z)$
    - Turn it into data x with a decoder, $p_\theta(x \mid z)$

Standard VAE

[Kingma & Welling 2013]

- **Goal**: learn to compress data (like images) into a low-dimensional representation z, and then reconstruct or generate new data with it
- Assume that data is generated as follows:
  - First sample a latent variable $z \sim p(z)$
  - Turn it into data x with a decoder, $p_\theta(x \mid z)$
- The encoder maps data x into a latent variable, $q_\phi(z \mid x)$

Standard VAE

[Kingma & Welling 2013]

Standard VAE

[Kingma & Welling 2013]

- Train by balancing two goals:

Standard VAE

[Kingma & Welling 2013]

- Train by balancing two goals:
    - **Reconstruct the input well**:

Standard VAE

[Kingma & Welling 2013]

- Train by balancing two goals:
  - **Reconstruct the input well**:

Reconstruction loss: $-\log p(x \mid z)$

Standard VAE

[Kingma & Welling 2013]

- Train by balancing two goals:
  - **Reconstruct the input well**:

    Reconstruction loss: $-\log p(x \mid z)$

  - **Keep z values well behaved**:

Standard VAE

[Kingma & Welling 2013]

- Train by balancing two goals:
  - **Reconstruct the input well**:

    Reconstruction loss: $-\log p(x \mid z)$

  - **Keep z values well behaved**:

    Regularization: $D_{KL}\left(q(z \mid x) \| \mathcal{N}(0, I)\right)$

Standard VAE

[Kingma & Welling 2013]

- Train by balancing two goals:
  - **Reconstruct the input well**:

    Reconstruction loss: $-\log p(x \mid z)$

  - **Keep z values well behaved**:

    Regularization: $D_{KL}\left(q(z \mid x) \| \mathcal{N}(0,I)\right)$

Makes the encoder's output distribution close to a standard Gaussian

Standard VAE

[Kingma & Welling 2013]

- Final loss:

Standard VAE

[Kingma & Welling 2013]

- Final loss:

$$\mathscr{L}_{VAE}(x) = -\mathbb{E}_{q(z|x)}\left[\log p(x\,|\,z)\right] + D_{KL}(q(z\,|\,x)\|\mathscr{N}(0,I))$$

Standard VAE

[Kingma & Welling 2013]

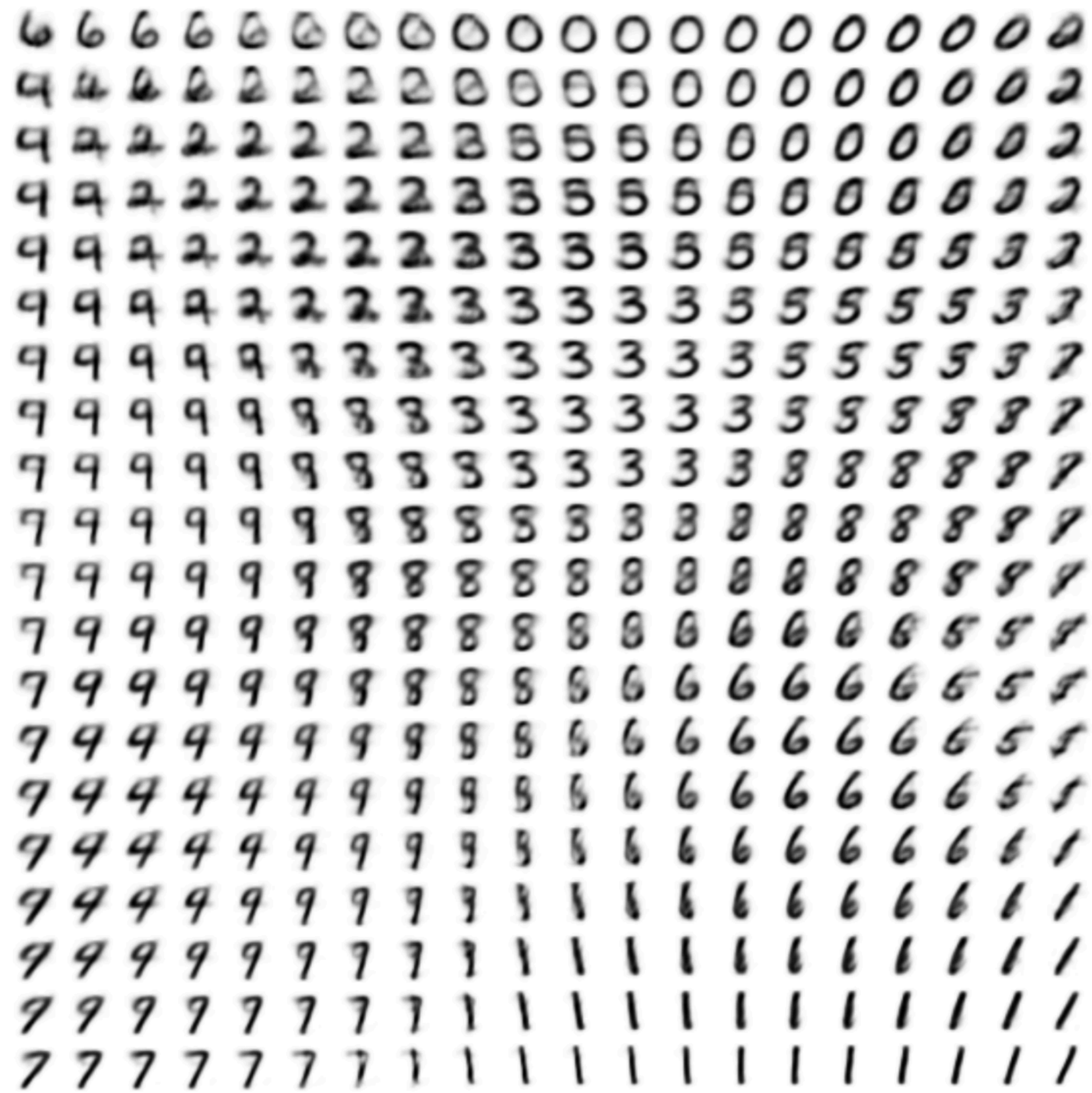
- Final loss:

$$\mathscr{L}_{VAE}(x) = -\,\mathbb{E}_{q(z|x)}\left[\log p(x\,|\,z)\right] + D_{KL}(q(z\,|\,x)\|\mathcal{N}(0,I))$$

- Principled formulation and derivation based on variational inference: see [Kingma & Welling 2013]

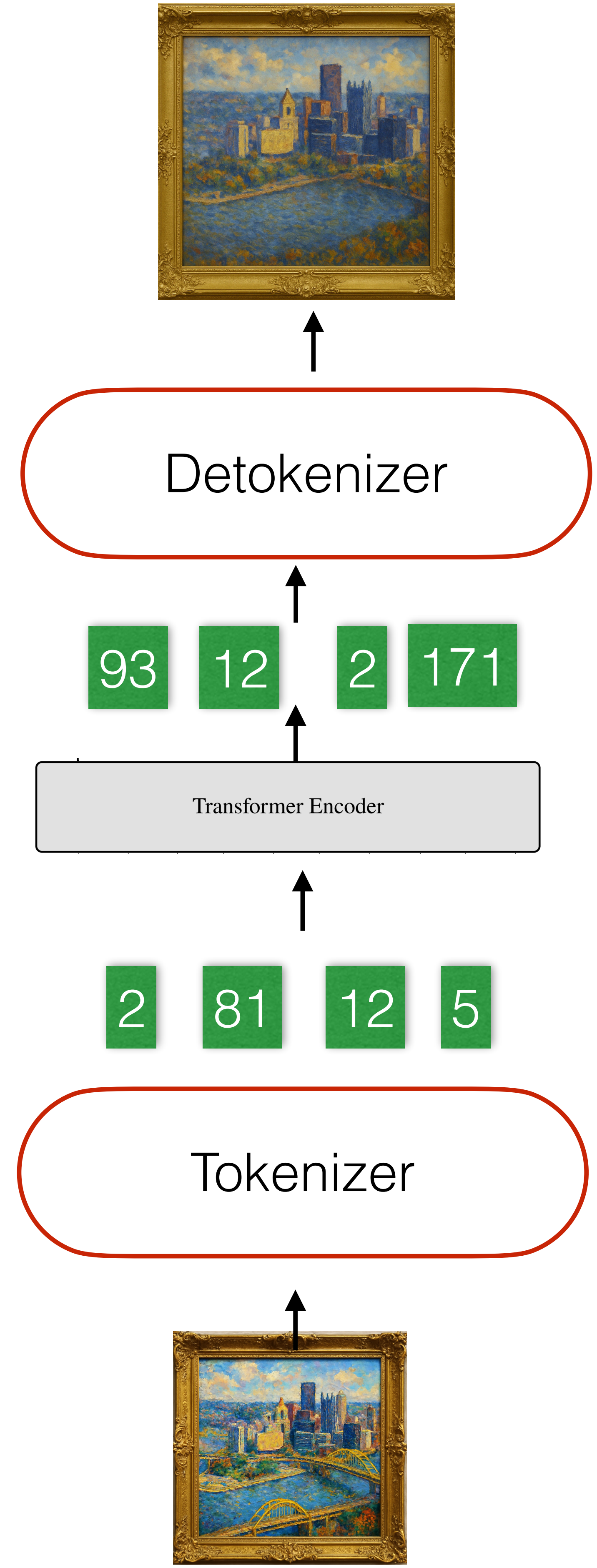
Standard VAE

(a) Learned Frey Face manifold



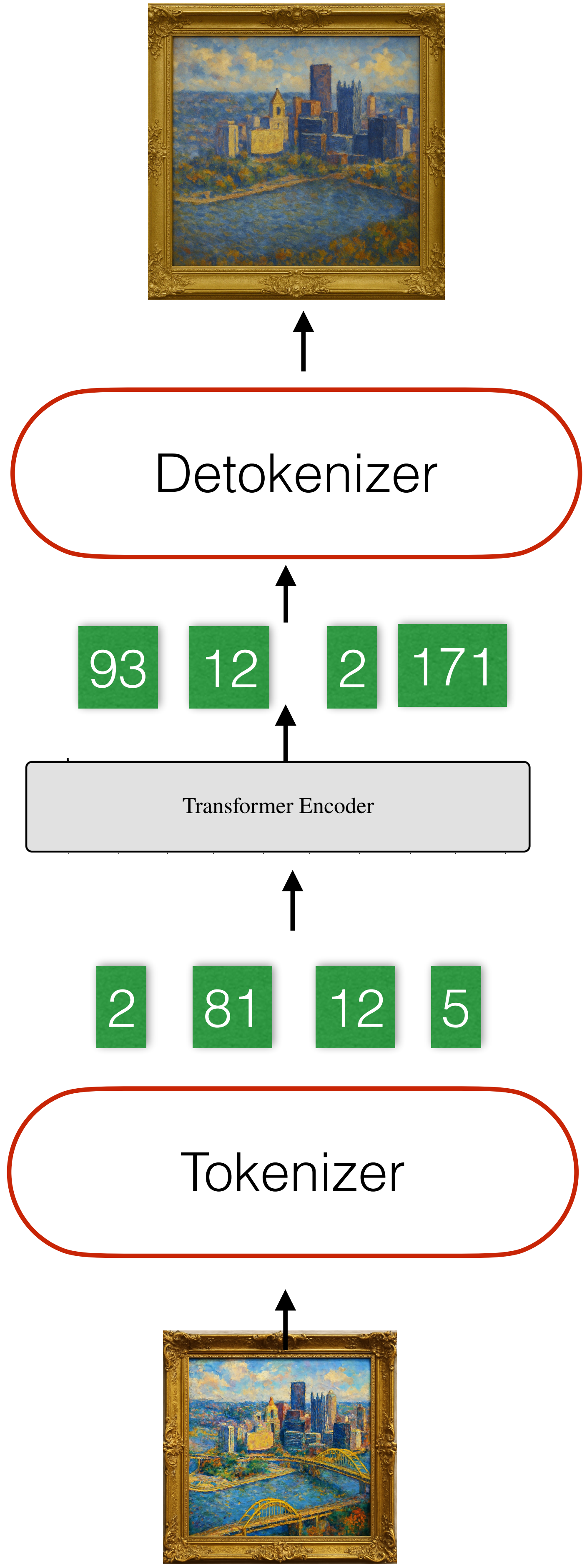
(b) Learned MNIST manifold

From continuous to discrete



From continuous to discrete

- Standard VAEs give us a way to encode and decode images using a continuous vector



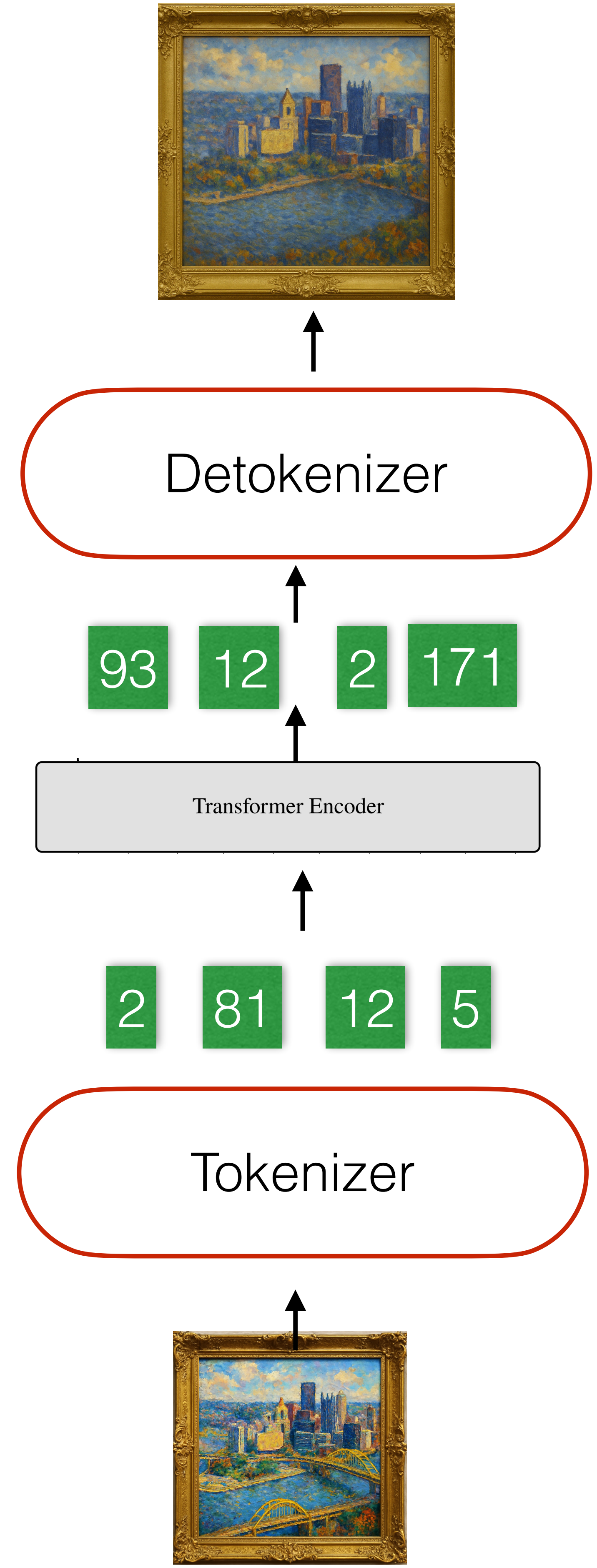
From continuous to discrete

- Standard VAEs give us a way to encode and decode images using a continuous vector

- Idea: make a *discrete* VAE to let us:

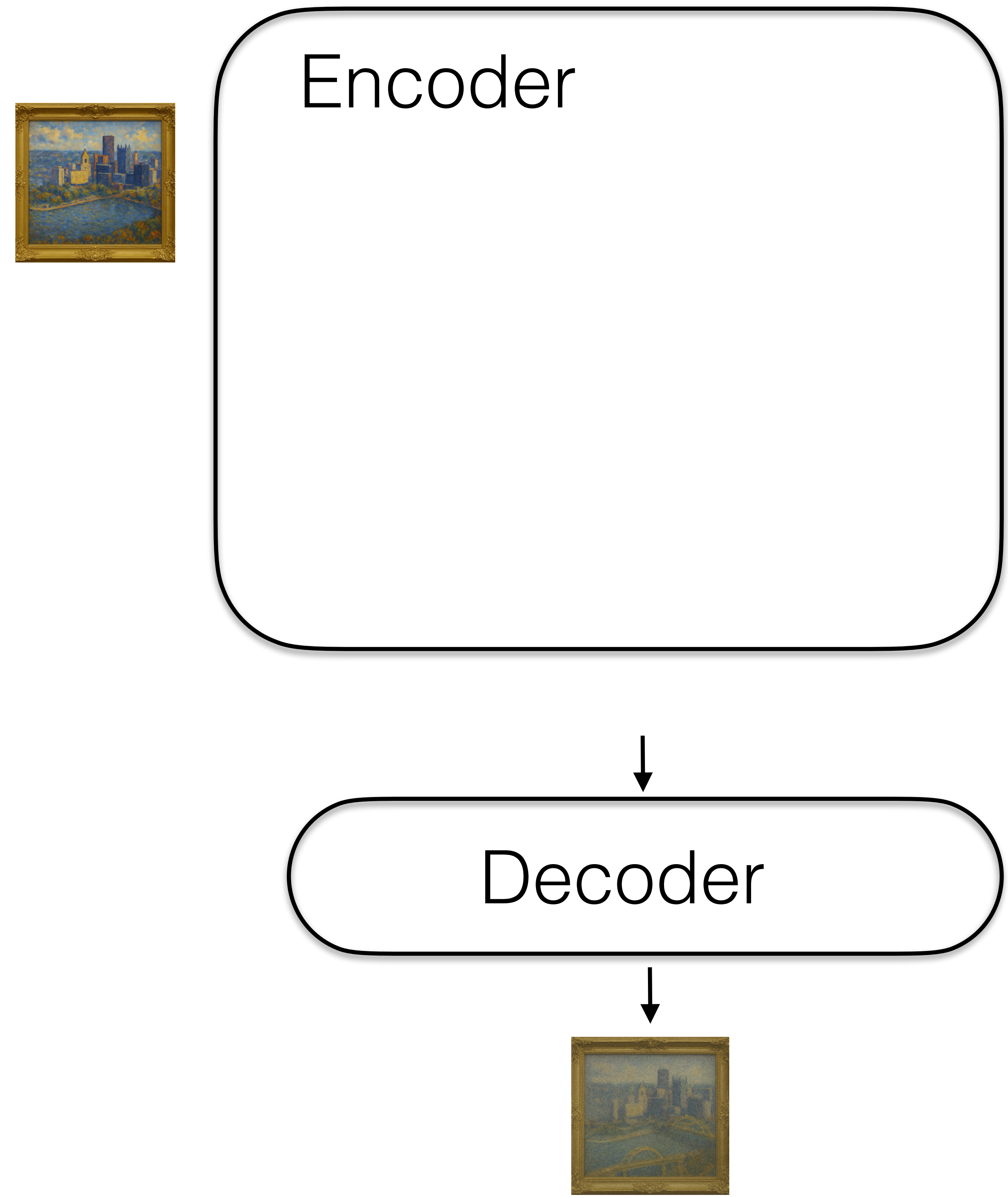
    - encode an image as a sequence of *discrete tokens*

    - decode an image from a sequence of discrete tokens



VQ-VAE

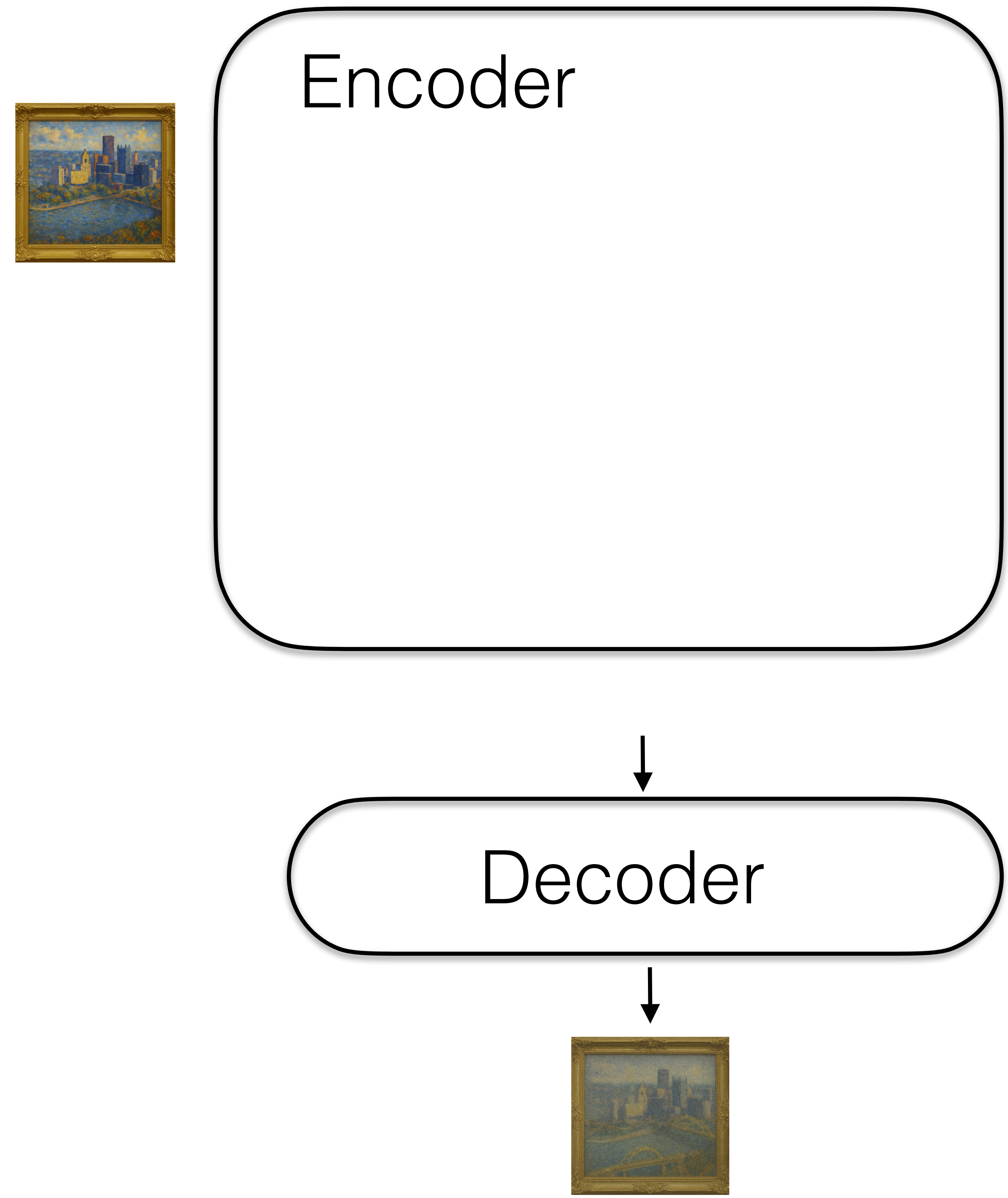
[van den Oord 2017]



VQ-VAE

[van den Oord 2017]

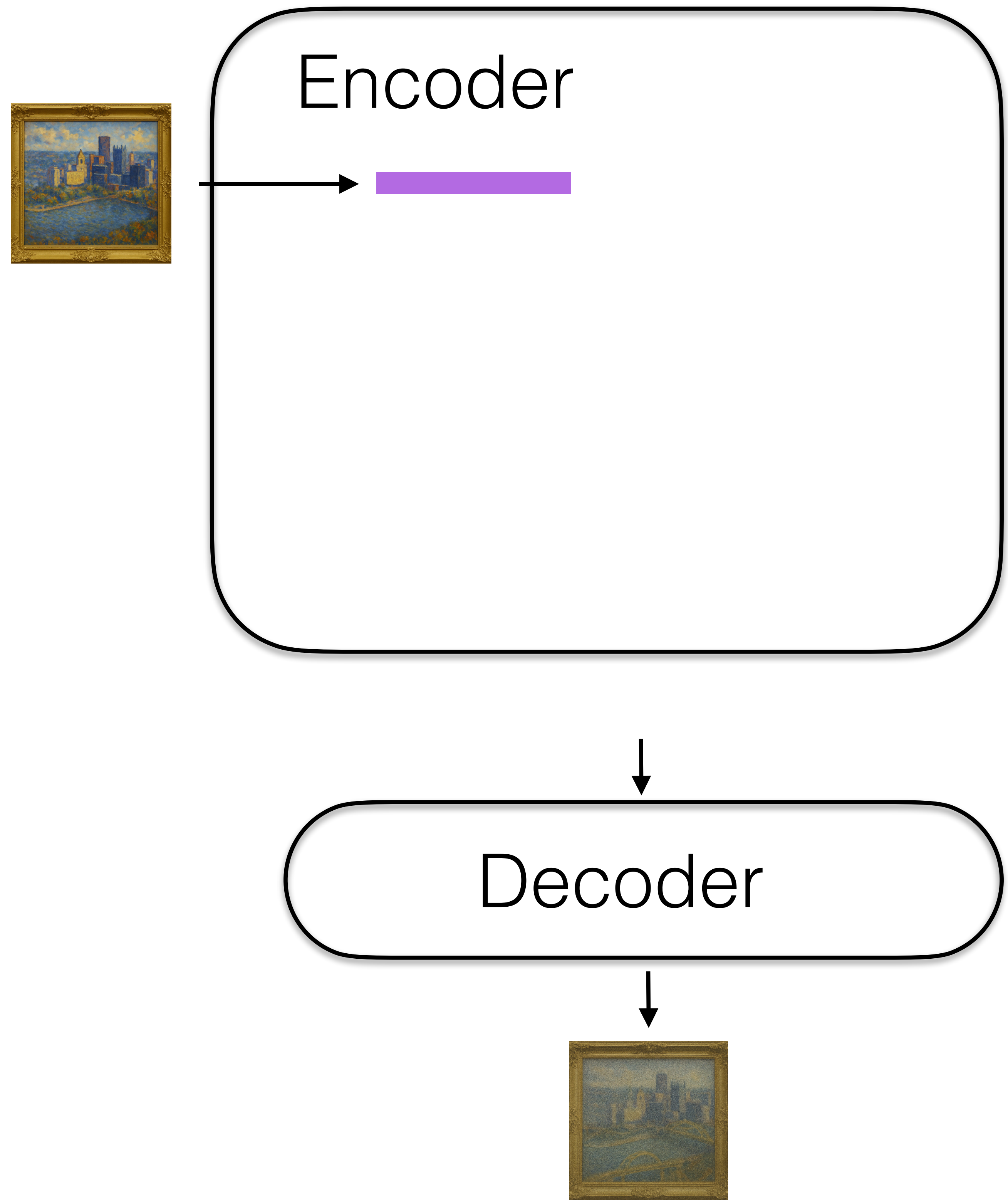
- Encoder: $q(z \mid x)$



VQ-VAE

[van den Oord 2017]

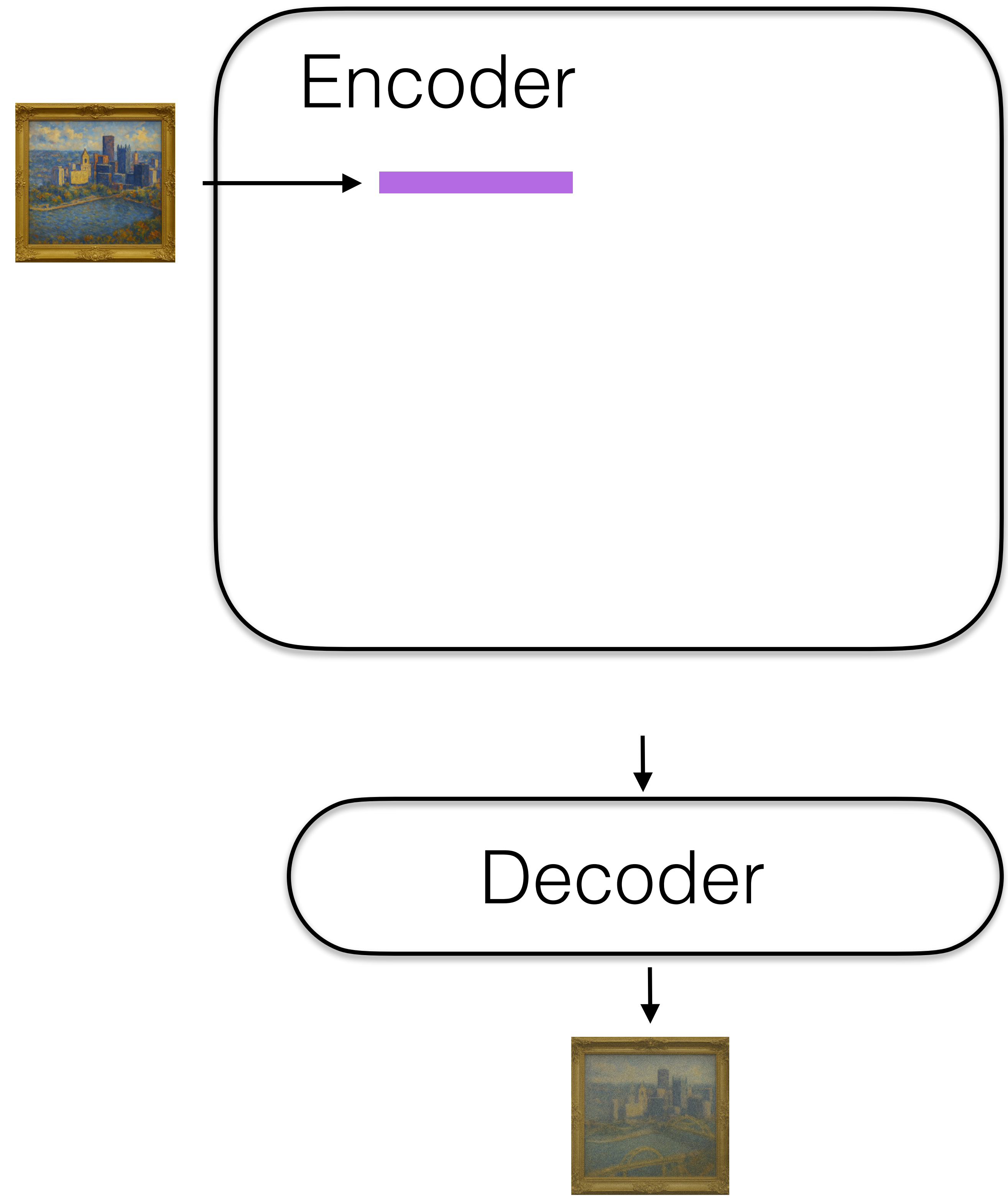
- Encoder: $q(z|x)$
  - $x \to z_e(x) \in \mathbb{R}^d$



VQ-VAE

[van den Oord 2017]

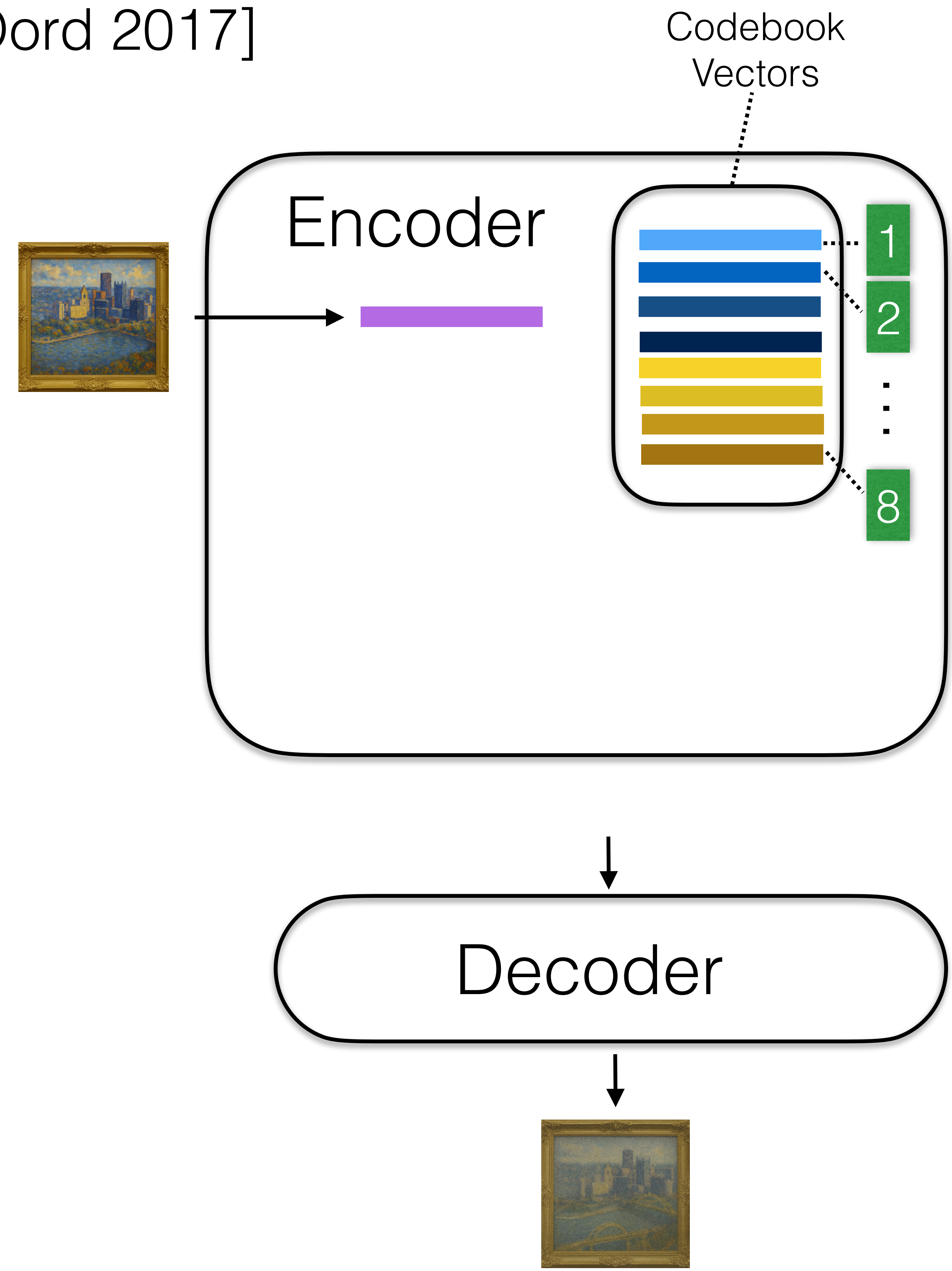
- Encoder: $q(z \mid x)$
  - $x \to z_e(x) \in \mathbb{R}^d$
  - Quantizer: $z_e(x) \to z_q(x)$



VQ-VAE

[van den Oord 2017]

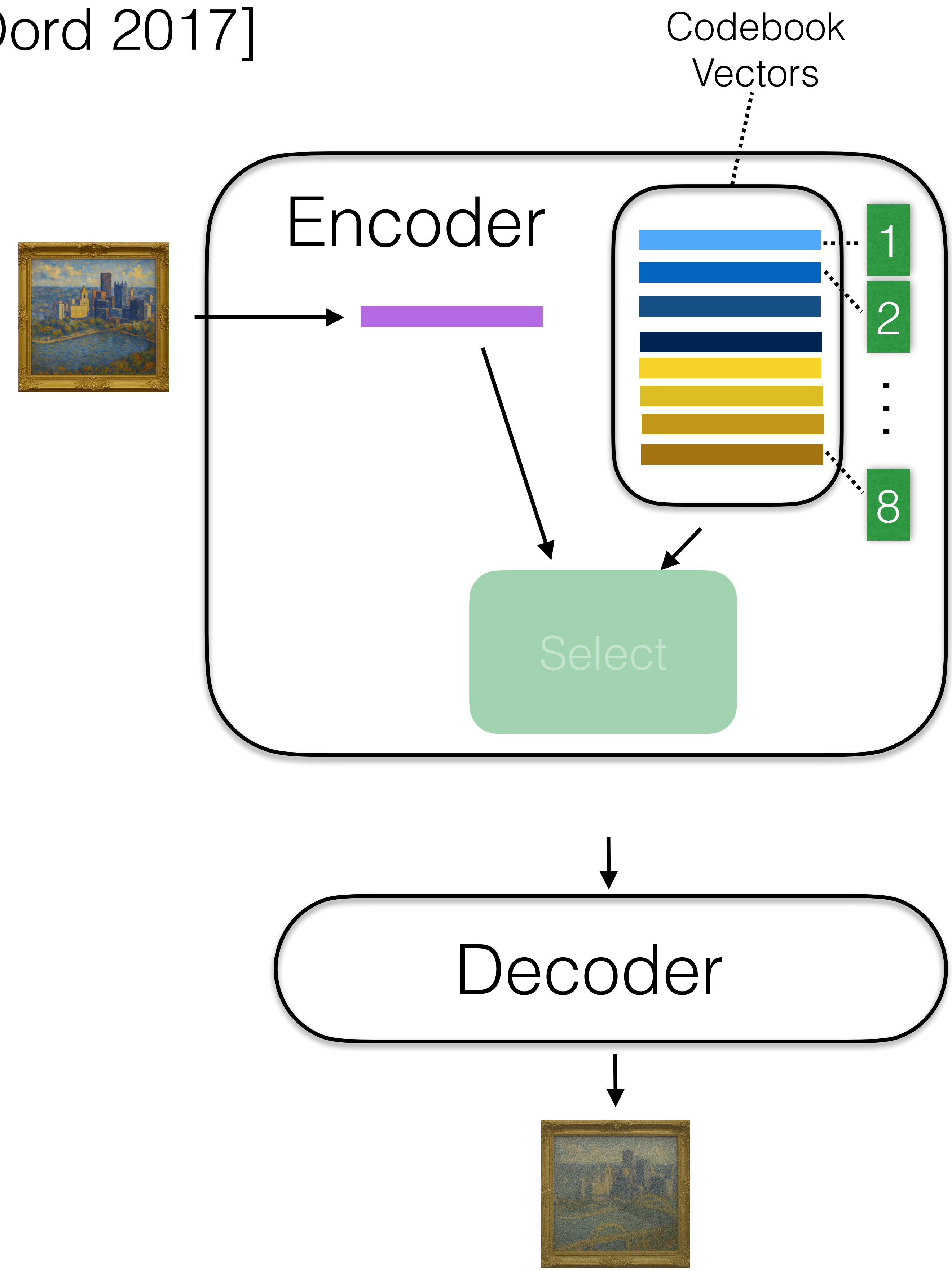
- Encoder: $q(z \mid x)$
  - $x \rightarrow z_e(x) \in \mathbb{R}^d$
  - Quantizer: $z_e(x) \rightarrow z_q(x)$
    - Compare $z_e(x)$ to a set of learned codebook vectors



VQ-VAE

[van den Oord 2017]

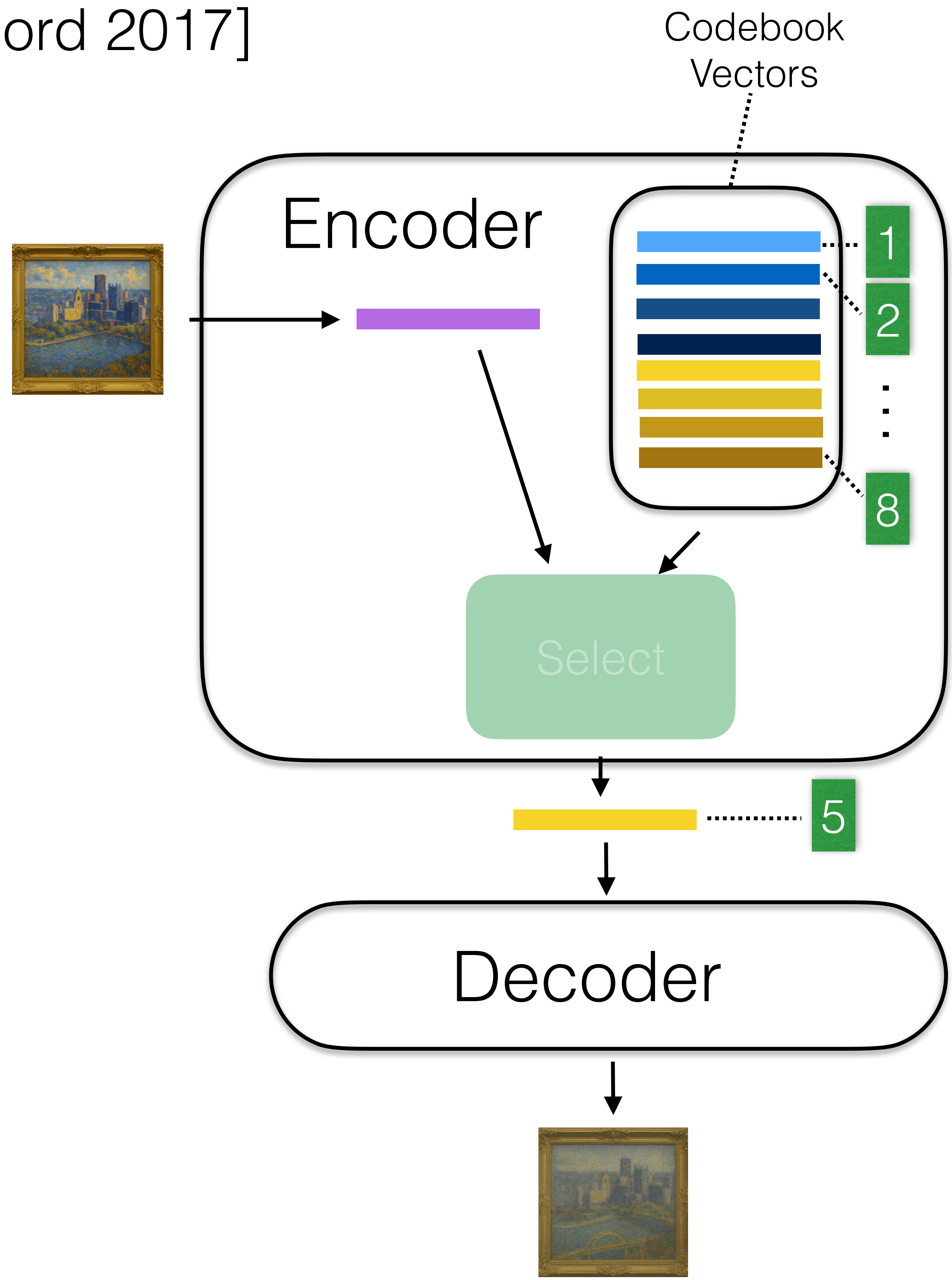
- Encoder: $q(z \mid x)$
  - $x \rightarrow z_e(x) \in \mathbb{R}^d$
  - Quantizer: $z_e(x) \rightarrow z_q(x)$
    - Compare $z_e(x)$ to a set of learned codebook vectors
    - $z_q(x) = e_{k*}$, where $k* = \arg \min_{j \in \{1,\ldots,K\}} \|z_e(x) - e_j\|_2$



VQ-VAE

[van den Oord 2017]

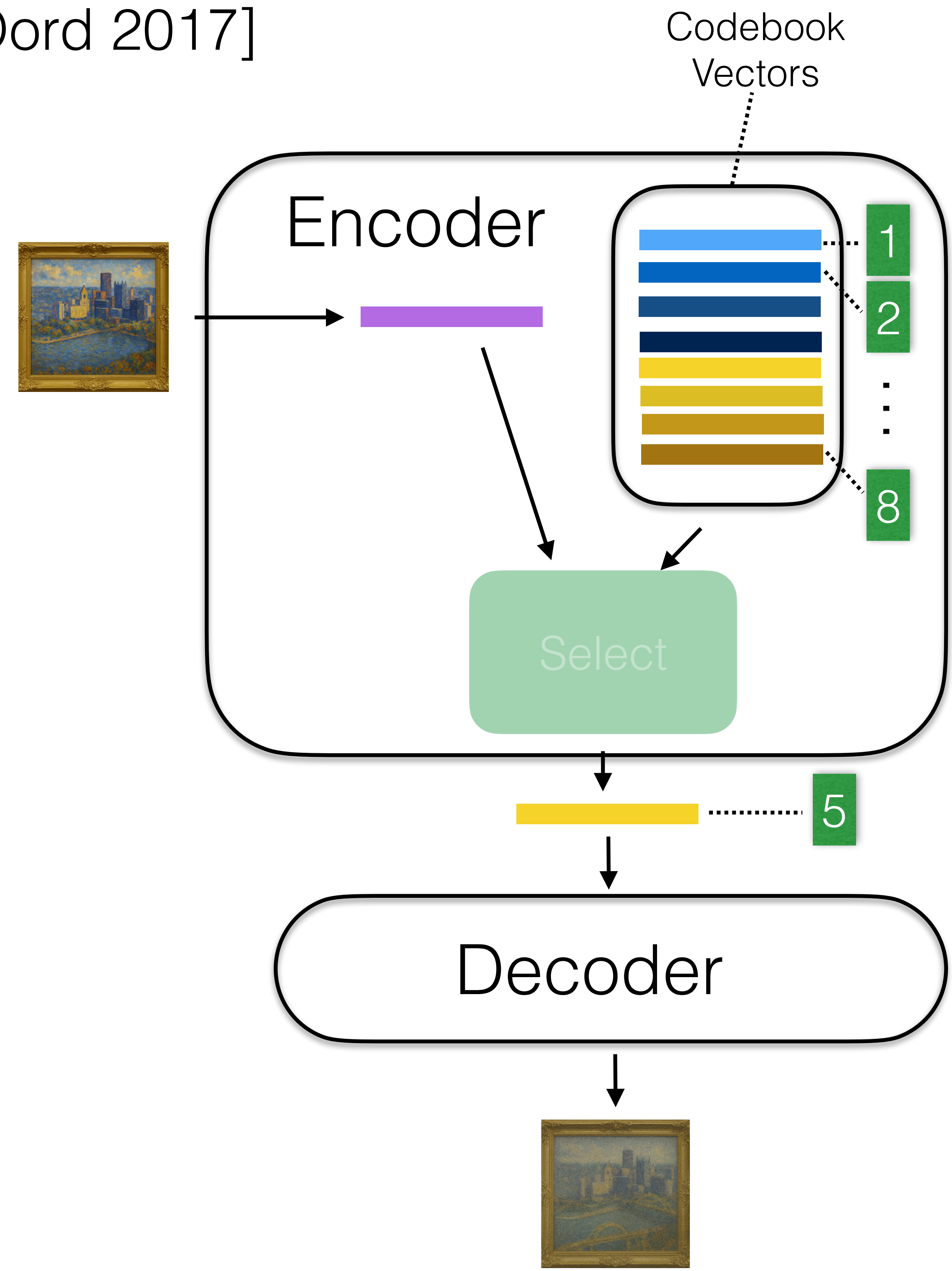
- Encoder: $q(z \mid x)$
  - $x \to z_e(x) \in \mathbb{R}^d$
  - Quantizer: $z_e(x) \to z_q(x)$
    - Compare $z_e(x)$ to a set of learned codebook vectors
    - $z_q(x) = e_{k*}$, where $k* = \arg\min_{j \in \{1,\ldots,K\}} \|z_e(x) - e_j\|_2$
  - $q(z = k \mid x) = \begin{cases} 1 & k = k* \\ 0 & \text{otherwise} \end{cases}$



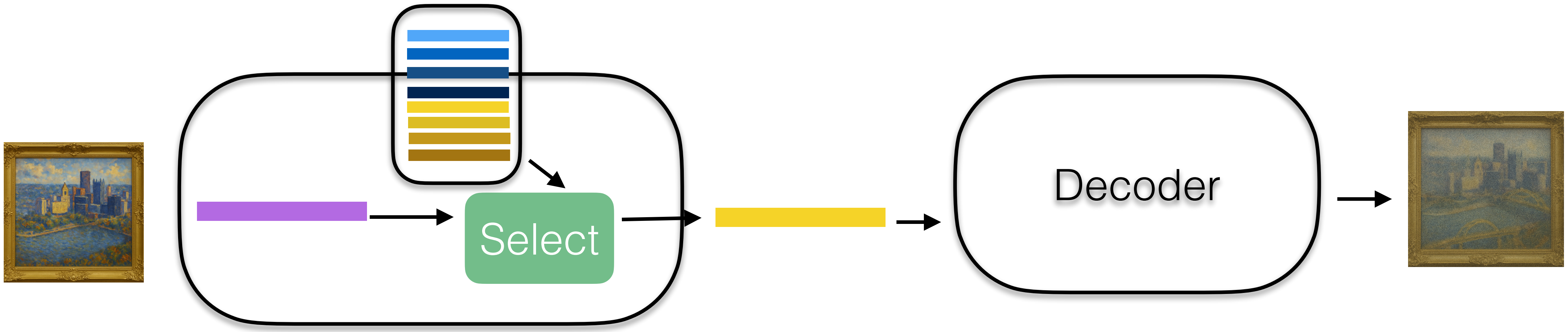
VQ-VAE

[van den Oord 2017]

- Encoder: $q(z \mid x)$
  - $x \rightarrow z_e(x) \in \mathbb{R}^d$
  - Quantizer: $z_e(x) \rightarrow z_q(x)$
    - Compare $z_e(x)$ to a set of learned codebook vectors
    - $z_q(x) = e_{k^*}$, where $k^* = \arg\min_{j \in \{1,\ldots,K\}} \|z_e(x) - e_j\|_2$
  - $q(z = k \mid x) = \begin{cases} 1 & k = k^* \\ 0 & \text{otherwise} \end{cases}$
- Decoder:
  - $p(x \mid z_q(x))$

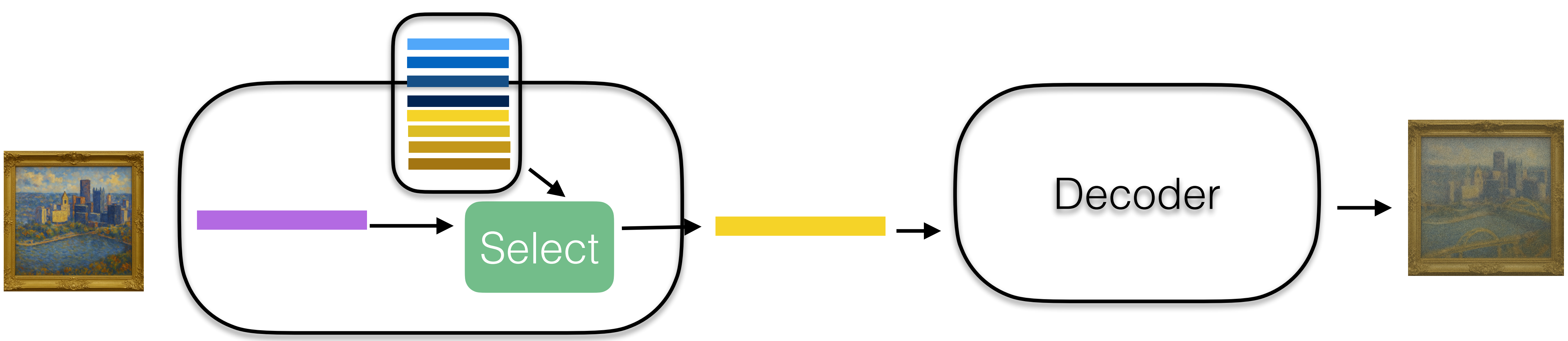


VQ-VAE training



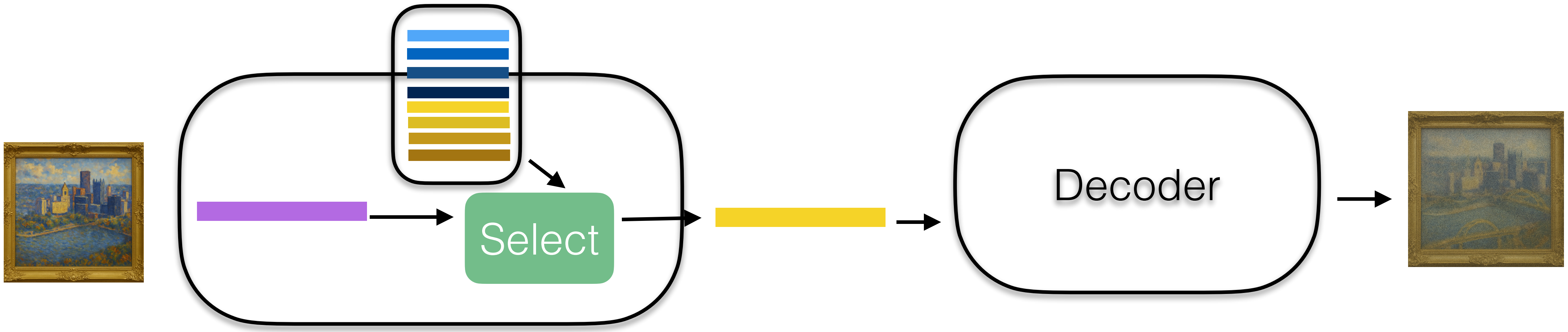
VQ-VAE training

- $\mathscr{L} = \log p(x \,|\, z_q(x)) + \|\text{sg}[z_e(x)] - e\|_2^2 + \beta \|z_e(x) - \text{sg}[e]\|_2^2$



VQ-VAE training

- $\mathscr{L} = \underbrace{\log p(x \mid z_q(x))}_{\text{Reconstruction}} + \|\mathrm{sg}[z_e(x)] - e\|_2^2 + \beta \|z_e(x) - \mathrm{sg}[e]\|_2^2$

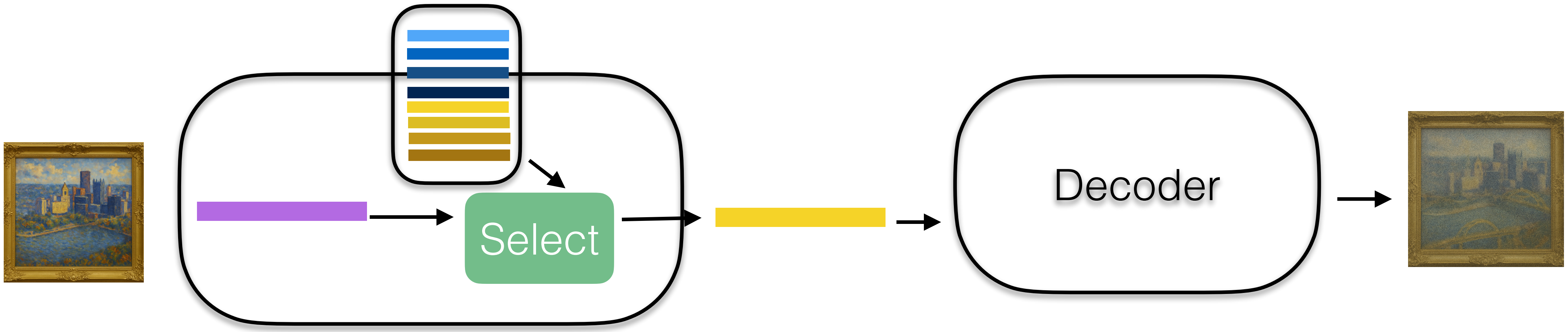


VQ-VAE training

- $\mathscr{L} = \log p(x \,|\, z_q(x)) + \|\mathrm{sg}[z_e(x)] - e\|_2^2 + \beta \|z_e(x) - \mathrm{sg}[e]\|_2^2$

Reconstruction

Update codebook embedding



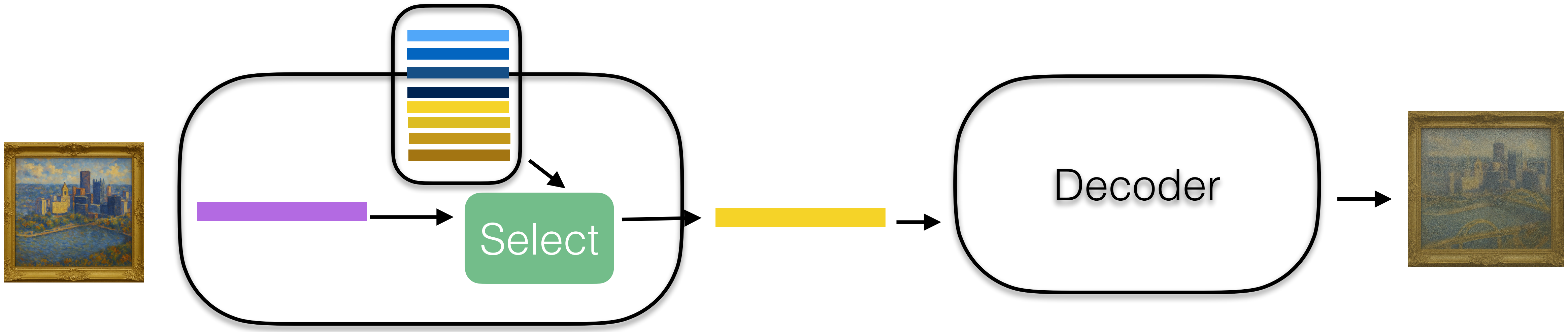
VQ-VAE training

- $\mathscr{L} = \log p(x \mid z_q(x)) + \|\mathrm{sg}[z_e(x)] - e\|_2^2 + \beta \|z_e(x) - \mathrm{sg}[e]\|_2^2$

Reconstruction

Update codebook embedding

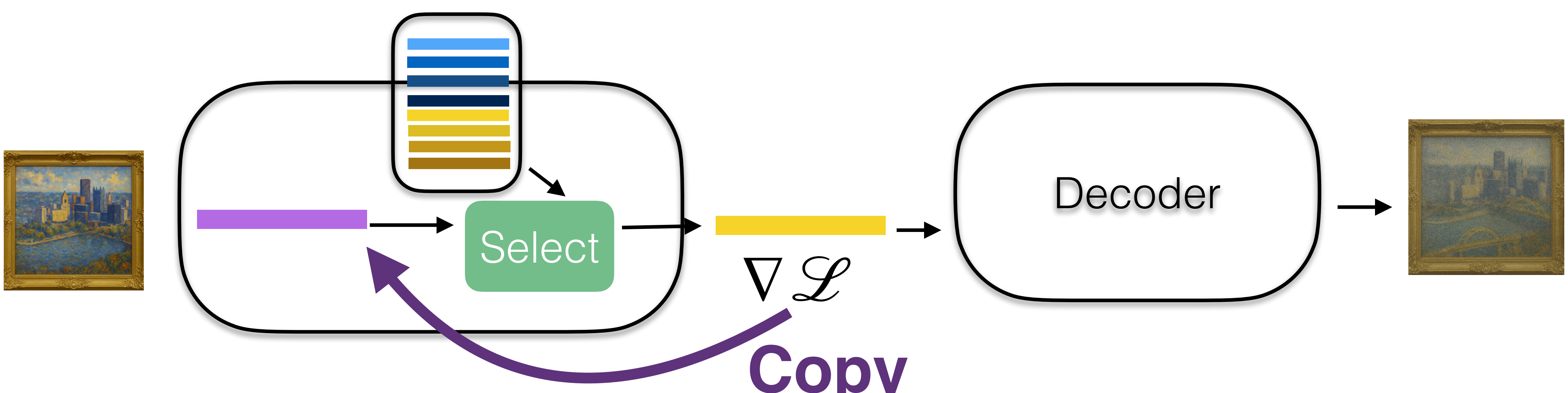
Make encoder “commit to” a codebook embedding



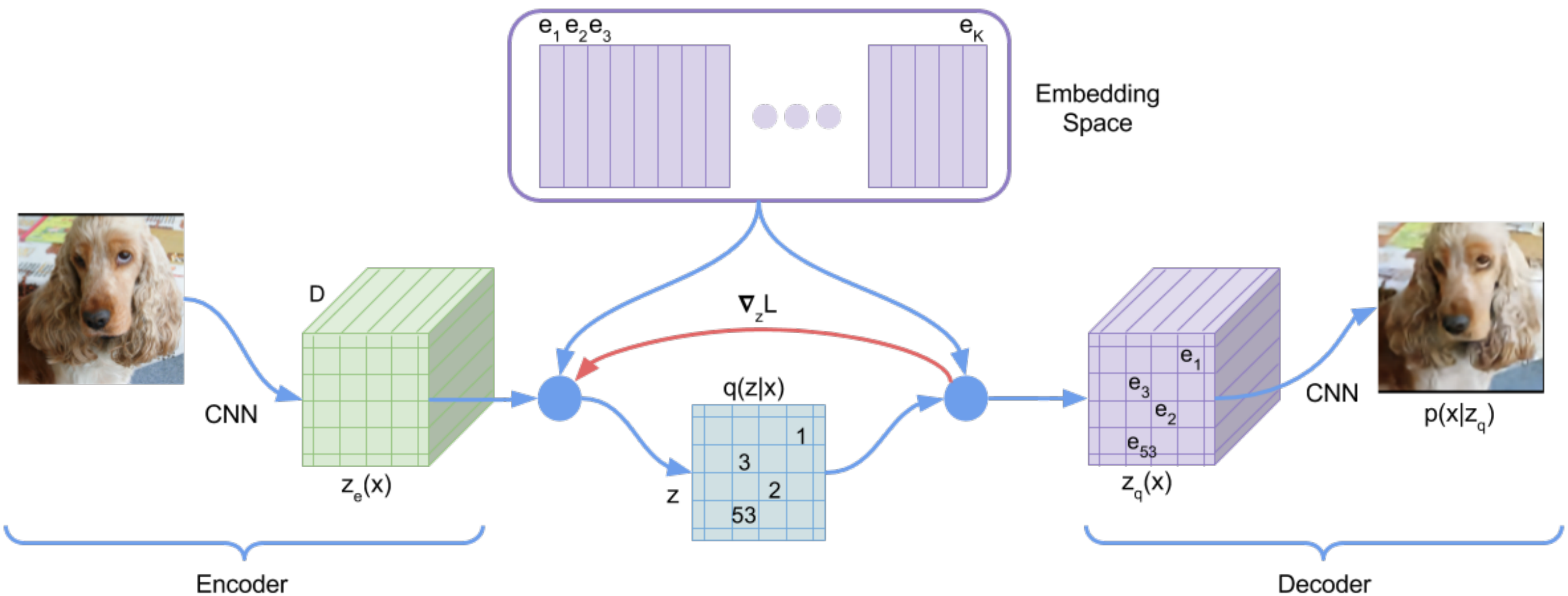
VQ-VAE training

- $\mathscr{L} = \underbrace{\log p(x \mid z_q(x))}_{\text{Reconstruction}} + \underbrace{\|\mathrm{sg}[z_e(x)] - e\|_2^2}_{\text{Update codebook embedding}} + \beta \underbrace{\|z_e(x) - \mathrm{sg}[e]\|_2^2}_{\text{Make encoder “commit to” a codebook embedding}}$

- For reconstruction loss, use a **straight-through gradient estimator** since the codebook lookup is non-differentiable

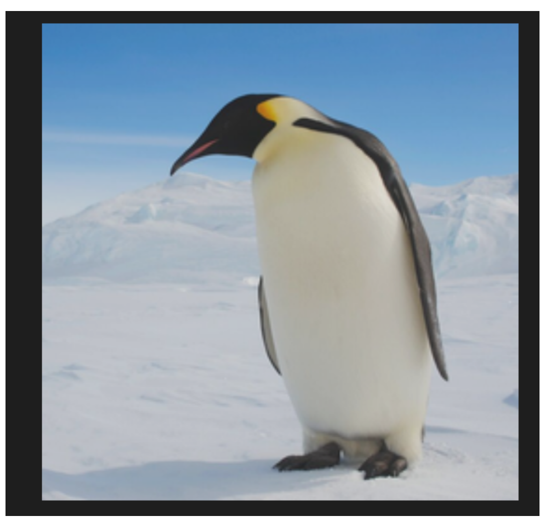


VQ-VAE


e_1 e_2 e_3
e_K
Embedding Space
D
CNN
$z_e(x)$
$\nabla_z L$
$q(z|x)$
1
3
2
53
z
e_1
e_3
e_2
e_{53}
$z_q(x)$
CNN
$p(x|z_q)$
Encoder
Decoder

Example

- DALL-E tokenizer



```
import torch.nn.functional as F

z_logits = enc(x)
z_tokens = torch.argmax(z_logits, axis=1)
z = F.one_hot(z_tokens, num_classes=enc.vocab_size).per
z_tokens, z_tokens.shape
```

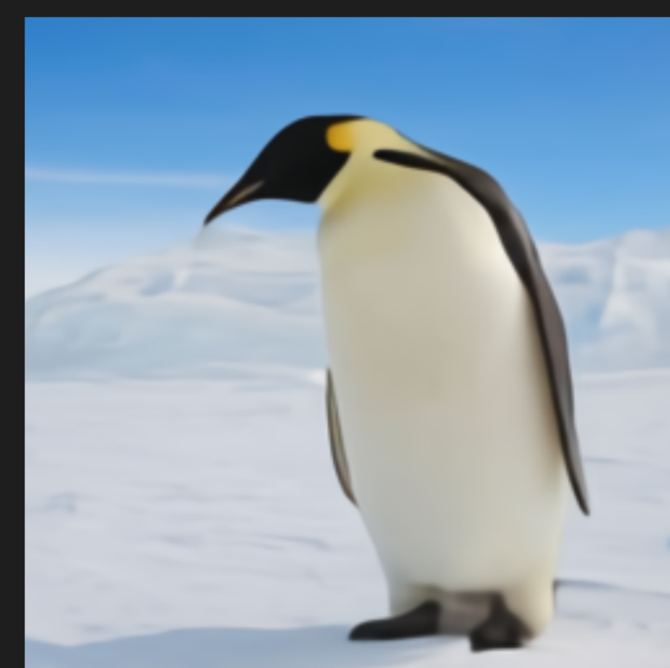
✓ 0.5s

```
(tensor([[[7522,  741,  119,  ...,  629, 4695, 4695],
          [ 782, 7459, 4762,  ..., 6913, 5215, 5887],
          [1580, 4066, 5768,  ..., 5677,  224, 2913],
          ...,
          [3990, 7130, 5047,  ..., 6027, 5770, 1212],
          [4898, 5659, 6296,  ..., 5788, 6476, 6397],
          [2757, 3243, 5504,  ..., 5016, 1144, 7261]]]),
 torch.Size([1, 32, 32]))
```

```
x_stats = dec(z).float()
x_rec = unmap_pixels(torch.sigmoid(x_stats[:, :3]))
x_rec = T.ToPILImage(mode='RGB')(x_rec[0])

display_markdown('Reconstructed image:')
display(x_rec)
```

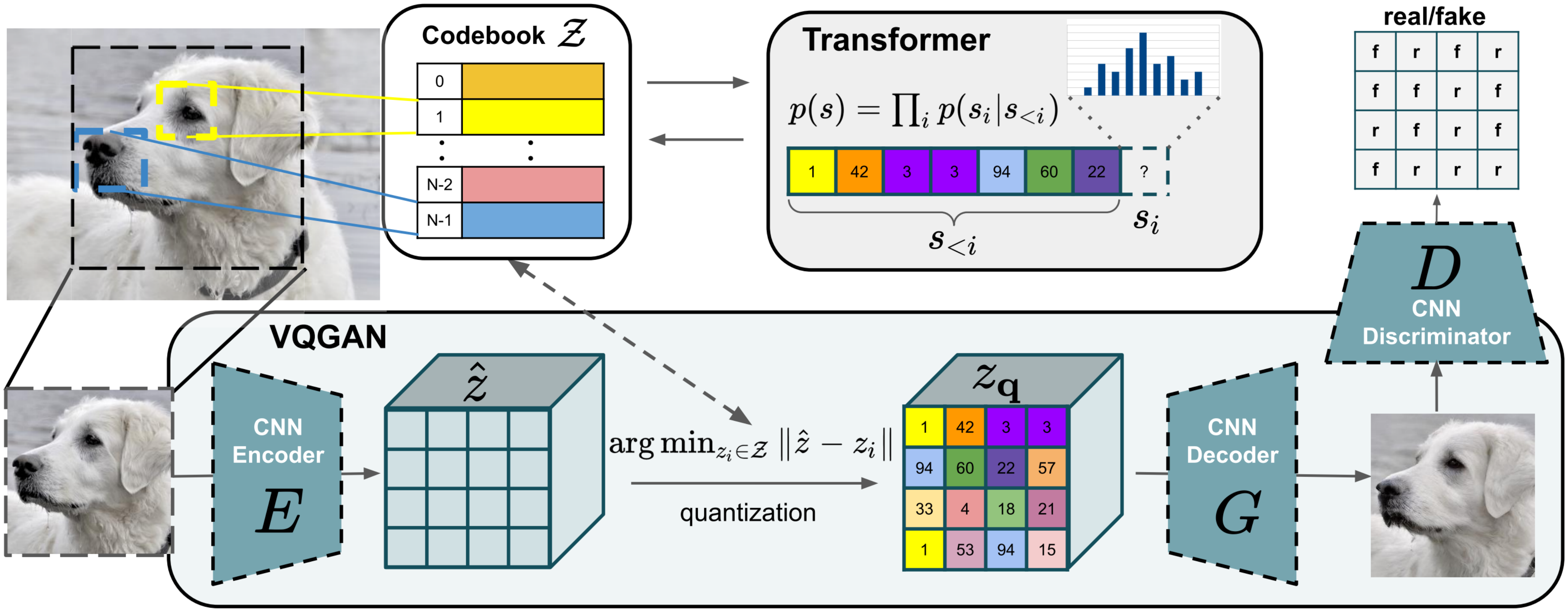
✓ 0.3s



https://github.com/openai/DALL-E

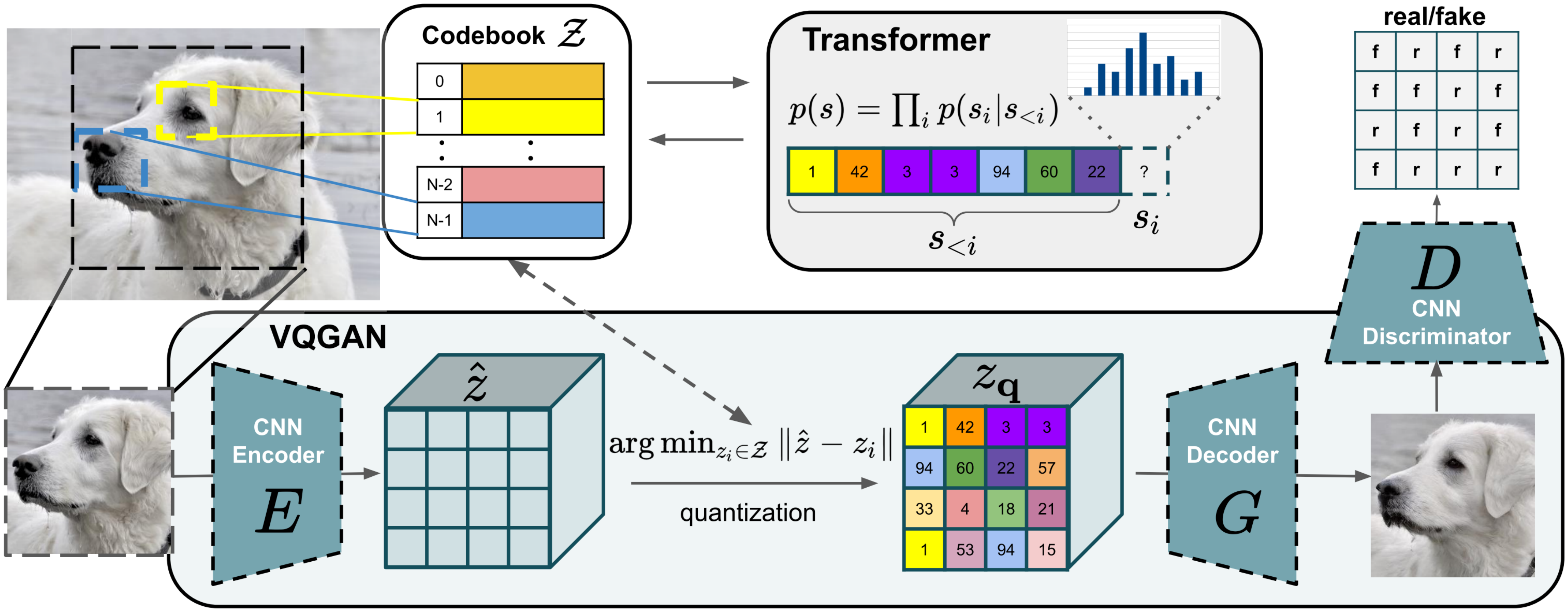
VQ-GAN

[Esser et al 2021]



VQ-GAN

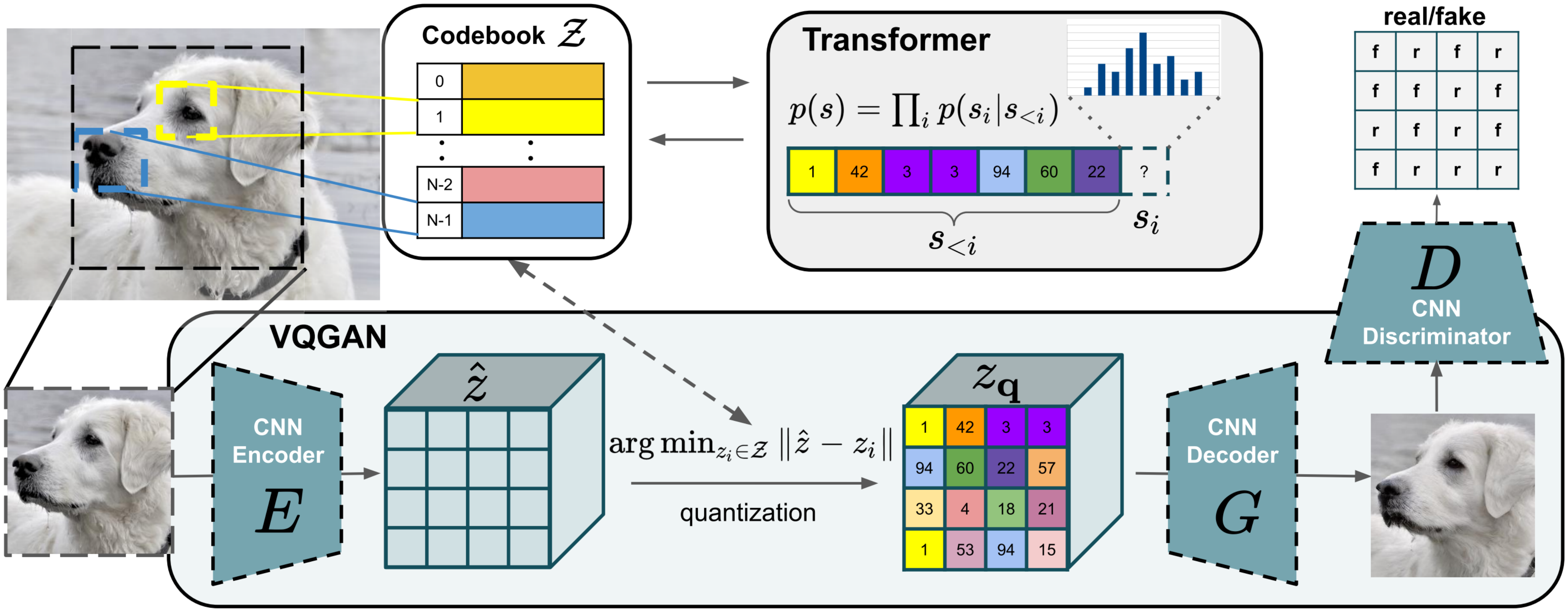
[Esser et al 2021]



- Modify VQ-VAE objective: use a GAN objective and perceptual loss

VQ-GAN

[Esser et al 2021]



- Modify VQ-VAE objective: use a GAN objective and perceptual loss
- Train an autoregressive transformer ("language model") on the resulting discrete tokenized images. Generate images with it.

VQ-GAN

[Esser et al 2021]

Training the encoder/decoder:

VQ-GAN

[Esser et al 2021]

Training the encoder/decoder:

- $\hat{x} = G(\text{select}(E(x))$

VQ-GAN

[Esser et al 2021]

Training the encoder/decoder:

- $\hat{x} = G(\text{select}(E(x))$
  - G: generator

VQ-GAN

[Esser et al 2021]

Training the encoder/decoder:

- $\hat{x} = G(\text{select}(E(x))$
    - G: generator
    - E: encoder

VQ-GAN

[Esser et al 2021]

Training the encoder/decoder:

- $\hat{x} = G(\text{select}(E(x))$
  - G: generator
  - E: encoder
- $\mathscr{L}_{\text{GAN}} = \log D(x) + \log(1 - D(\hat{x}))$

VQ-GAN

[Esser et al 2021]

Training the encoder/decoder:

- $\hat{x} = G(\text{select}(E(x)))$
  - G: generator
  - E: encoder
- $\mathscr{L}_{\text{GAN}} = \log D(x) + \log(1 - D(\hat{x}))$
  - D: discriminator

VQ-GAN

[Esser et al 2021]

Training the transformer language model:

VQ-GAN

[Esser et al 2021]

Training the transformer language model:

- Tokenize lots of images using the tokenizer (encoder) $x_{img} \rightarrow (x_1, x_2, \ldots, x_T)$

VQ-GAN

[Esser et al 2021]

Training the transformer language model:

- Tokenize lots of images using the tokenizer (encoder)
  $x_{img} \rightarrow (x_1, x_2, \ldots, x_T)$

- Train an autoregressive transformer:

VQ-GAN

[Esser et al 2021]

Training the transformer language model:

- Tokenize lots of images using the tokenizer (encoder)
  $x_{img} \rightarrow (x_1, x_2, \ldots, x_T)$

- Train an autoregressive transformer:

- $$\mathscr{L} = \mathbb{E}_{x_{img}}[-\log \prod_{t=1}^{T} p(x_t \,|\, x_{<t})]$$

VQ-GAN

[Esser et al 2021]

Training the transformer language model:

- Tokenize lots of images using the tokenizer (encoder) $x_{img} \to (x_1, x_2, \ldots, x_T)$

- Train an autoregressive transformer:

  - $$\mathscr{L} = \mathbb{E}_{x_{img}}[-\log \prod_{t=1}^{T} p(x_t | x_{<t})]$$

  - Example T: 32 x 32 = 1024

VQ-GAN

[Esser et al 2021]

Training the transformer language model:

- Tokenize lots of images using the tokenizer (encoder) $x_{img} \rightarrow (x_1, x_2, \ldots, x_T)$

- Train an autoregressive transformer:

   - $$\mathscr{L} = \mathbb{E}_{x_{img}}[-\log \prod_{t=1}^{T} p(x_t \mid x_{<t})]$$

   - Example T: 32 x 32 = 1024

      - Much less than modeling pixels (256x256 = 65,536)!

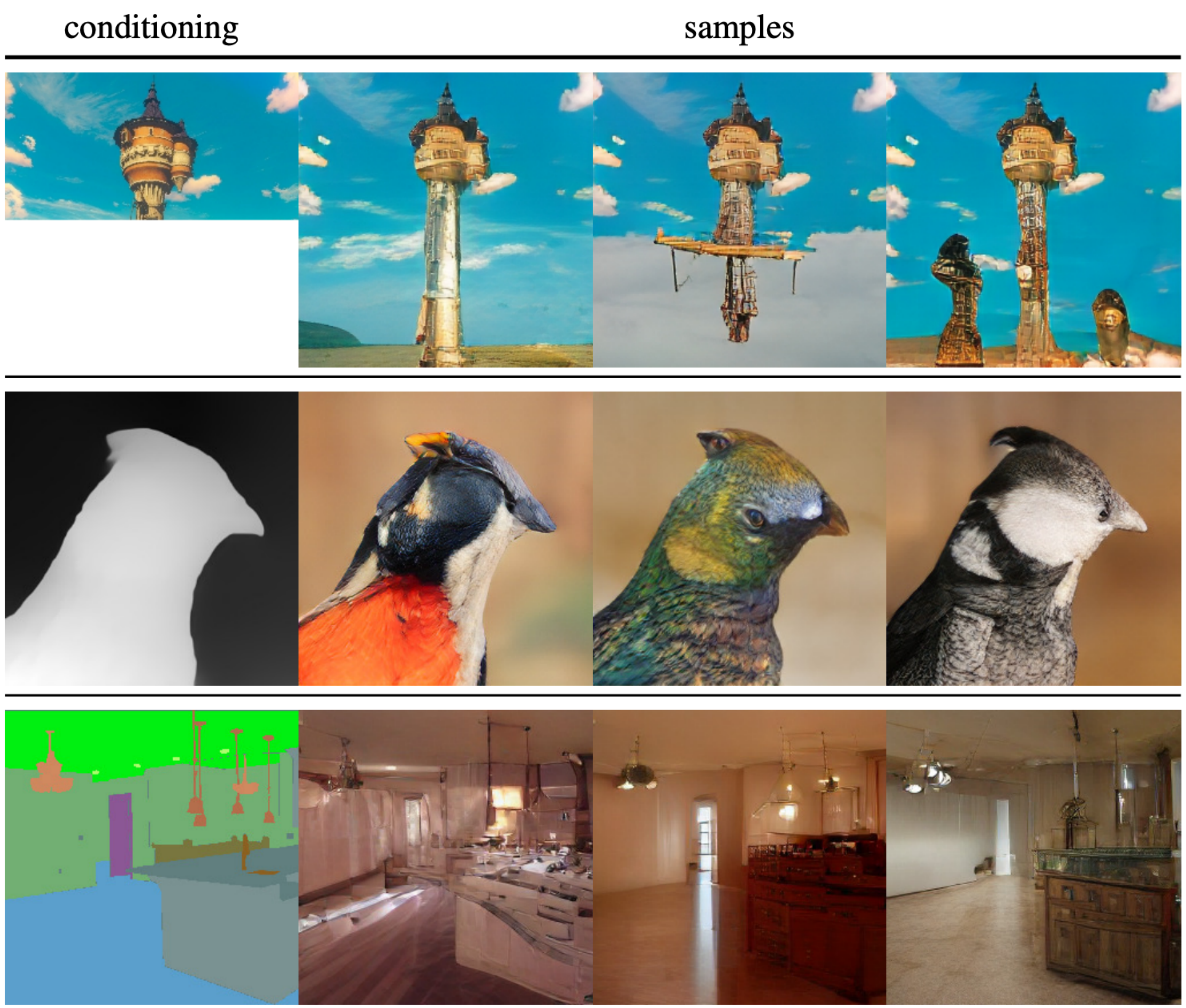
VQ-GAN

[Esser et al 2021]

Figure 8. Samples from our class-conditional ImageNet model trained on 256×256 images.

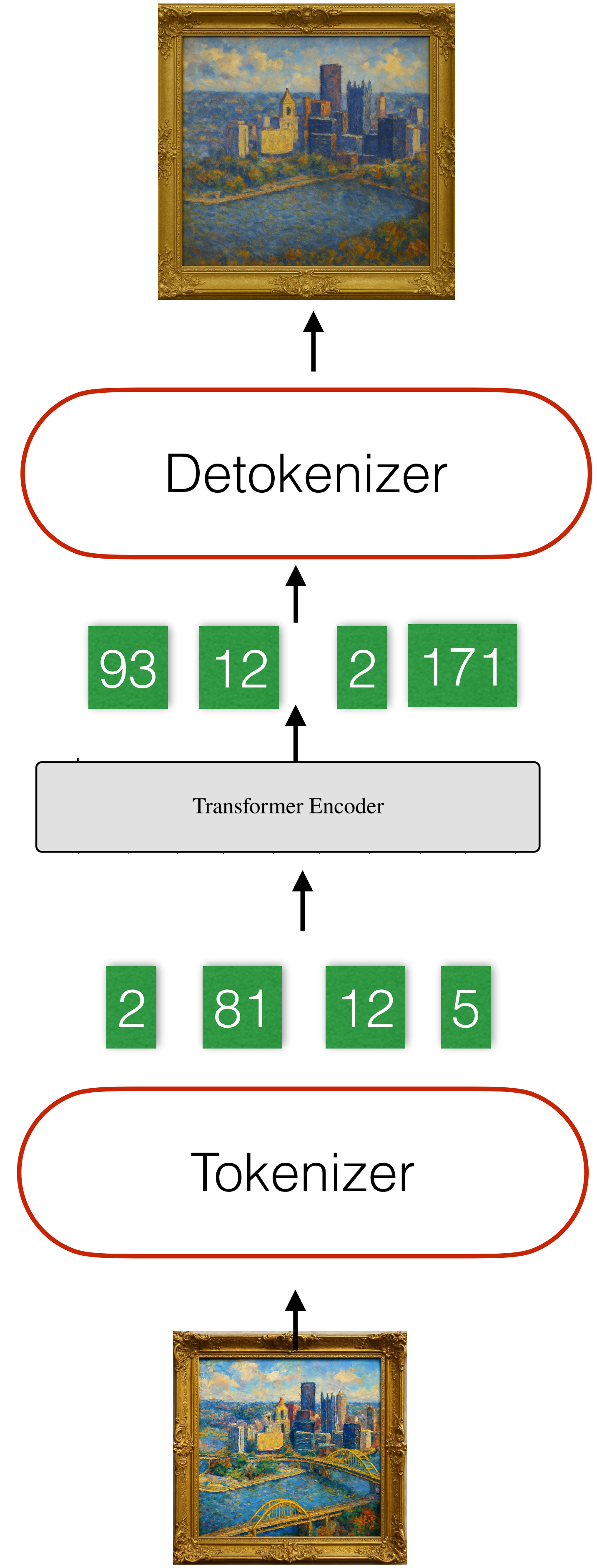
VQ-GAN

[Esser et al 2021]



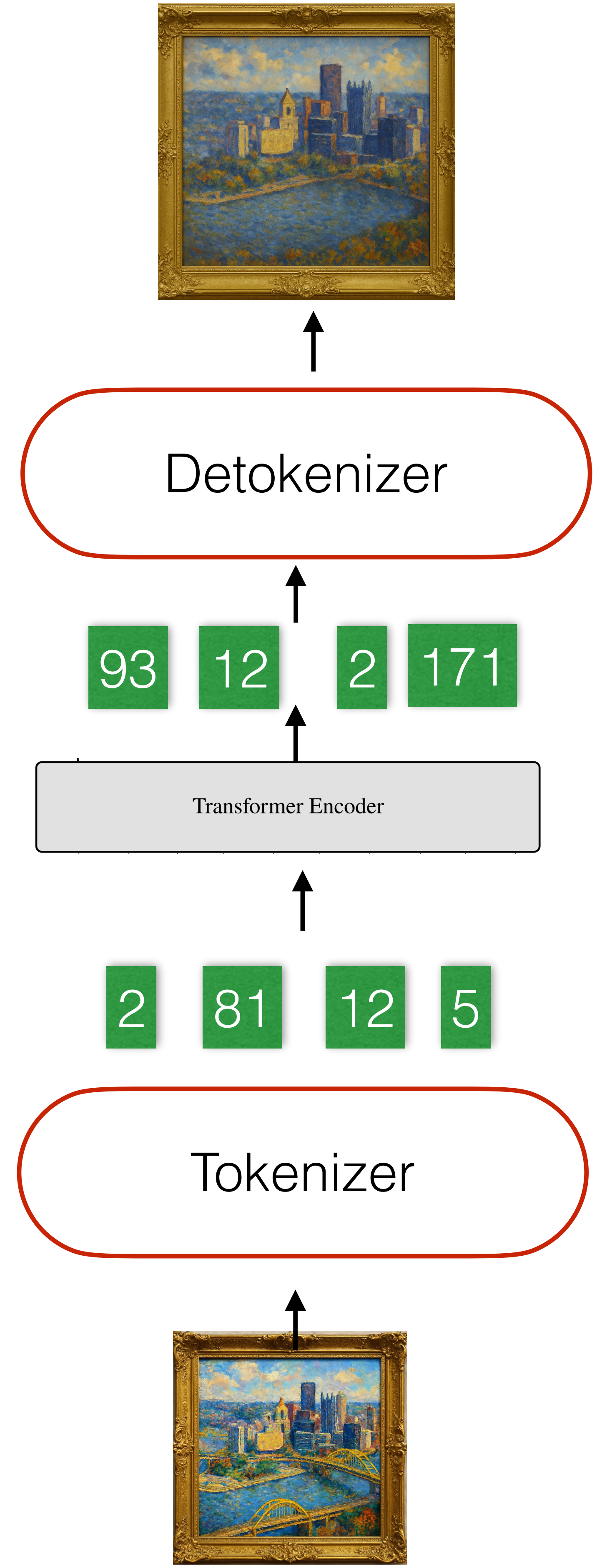
Recap

- Tokenizer: VQ-VAE/VQ-GAN encoder
- De-tokenizer: VQ-VAE/VQ-GAN decoder



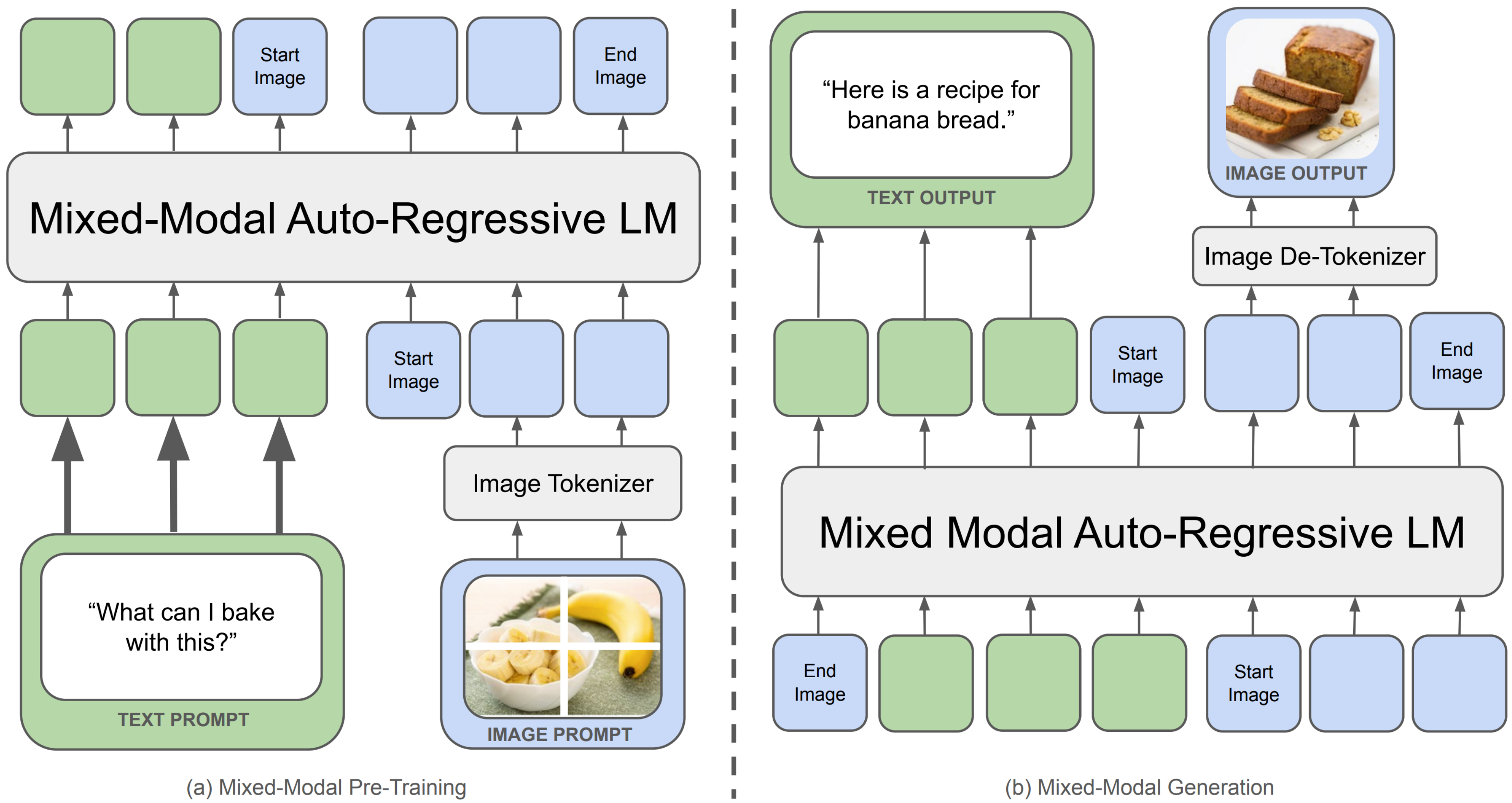
Recap

- Tokenizer: VQ-VAE/VQ-GAN encoder
- De-tokenizer: VQ-VAE/VQ-GAN decoder
- Next:
  - Add the tokens to a language model's vocabulary
  - Train / fine-tune the model on both text and image tokens



Example: Chameleon

[Meta 2024]



- Train a VQ-GAN tokenizer/de-tokenizer
- Then train a language model on 10T tokens

Example: Chameleon

[Meta 2024]

Prompt: **<img>** While chameleons are masters of camouflage in the natural world, let's see how they might fare in a man-made setting.
1. Analyze the colors, patterns, and textures in this image and identify the most challenging spot for a chameleon to blend in.
2. Then generate an image of a chameleon with the optimal pattern to camouflage itself.
3. Then, rate and explain how successful each attempt is on a scale from 1-10.

Response: The image is of a stop sign in a forested area, with some sunlight filtering through the trees. The sign has a red octagonal shape with a white border and the word "STOP" written in white letters. In the background, there are green leaves and trees.

Challenging Spot for a Chameleon: The stop sign would be the most challenging spot for a chameleon to blend in due to its bright red color and bold white letters, which would stand out in the forested environment.

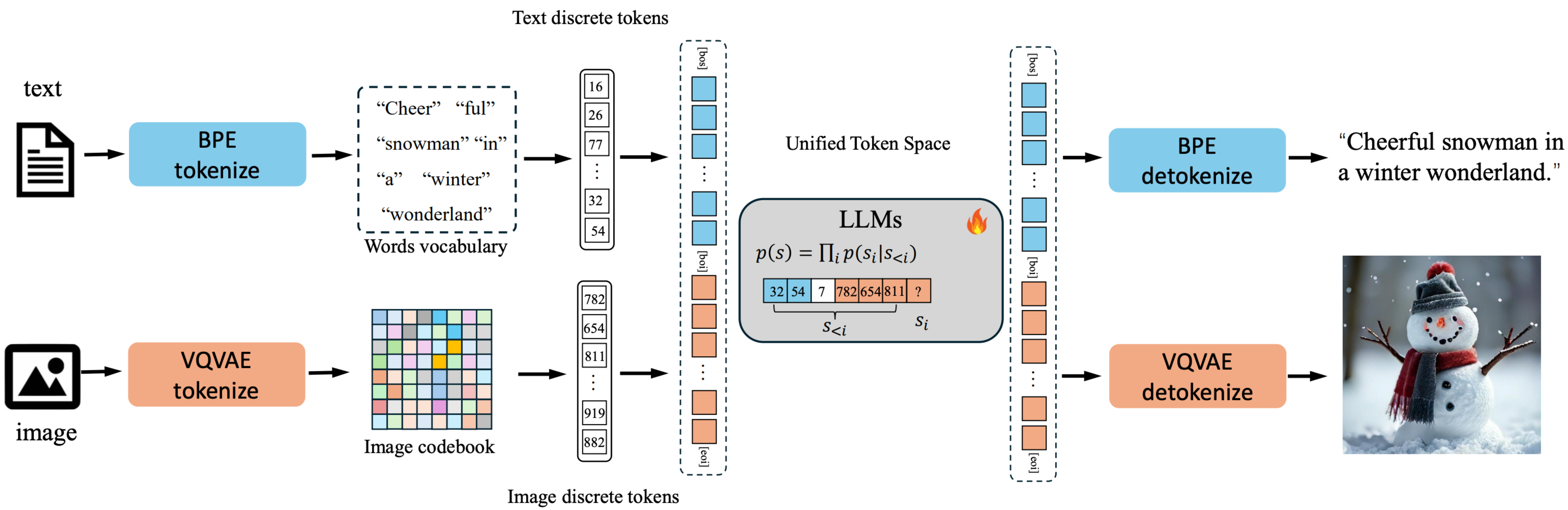
Optimal Pattern for Camouflage: To camouflage itself on the stop sign, a chameleon would need to have a pattern that would break up its outline and make it less conspicuous. A pattern with shades of brown and red would help the chameleon blend in more effectively. Additionally, a mottled or dappled pattern would help to scatter light and further reduce the chameleon's visibility.

<img>

Rating and Explanation: A rating of 2/10 would be given for the chameleon's ability to blend in on the stop sign. While the chameleon's green and brown pattern helps to break up its outline, the bright red color of the stop sign will still make it stand out in the forested environment.

Example: Liquid

[Wu et al 2024]

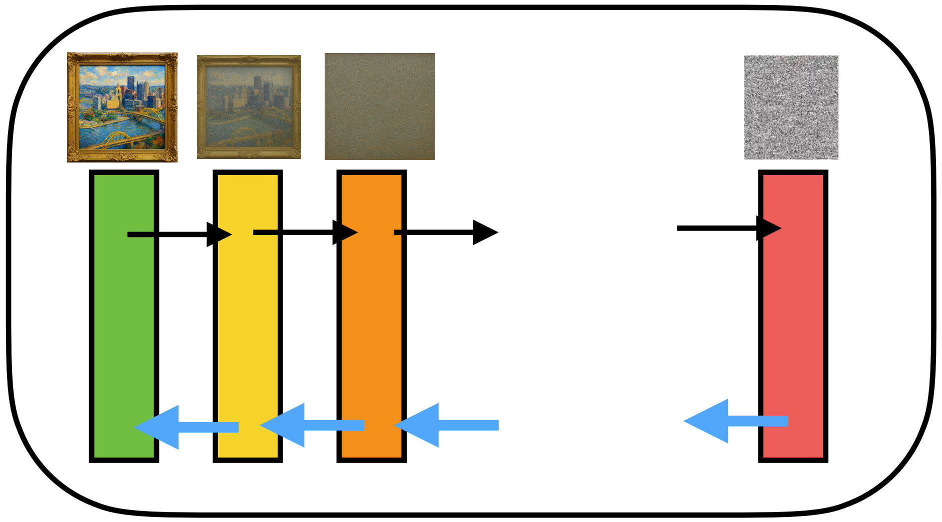


- VQ-GAN tokenizer
- Fine-tune LLMs

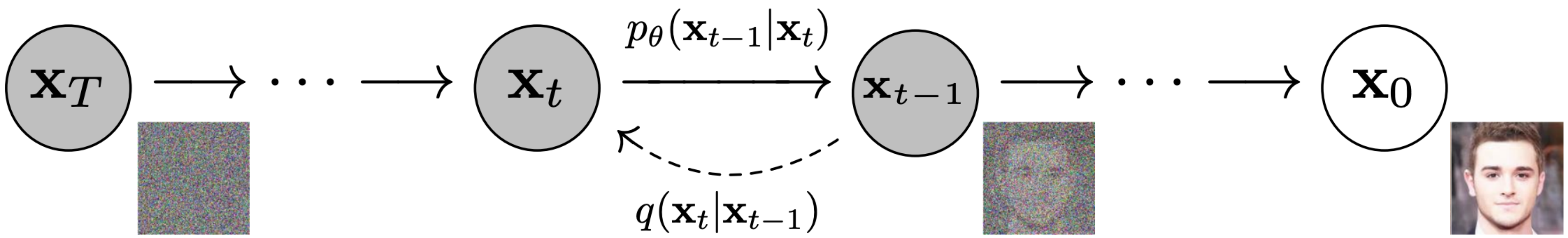
Other methods

- A big missing piece: diffusion!
- Many text-to-image models
- An extremely quick overview of some early references (up to early 2022)

Diffusion



Denoising Diffusion Probabilistic Models [Ho et al 2020]



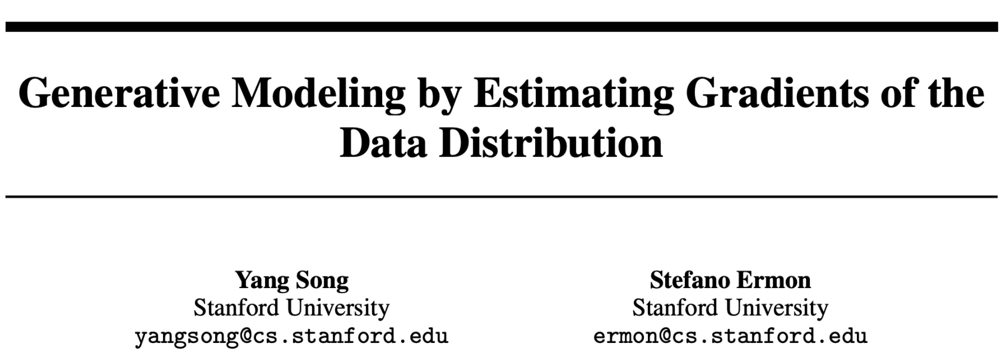
Denoising Diffusion Probabilistic Models

Jonathan Ho
UC Berkeley
jonathanho@berkeley.edu

Ajay Jain
UC Berkeley
ajayj@berkeley.edu

Pieter Abbeel
UC Berkeley
pabbeel@cs.berkeley.edu

Score-based generative models [Song & Ermon 2020]


Generative Modeling by Estimating Gradients of the Data Distribution

Yang Song
Stanford University
yangsong@cs.stanford.edu

Stefano Ermon
Stanford University
ermon@cs.stanford.edu

Diffusion Models Beat GANs [Dhariwal & Nichol 2021]

Diffusion Models Beat GANs on Image Synthesis

Prafulla Dhariwal* OpenAI prafulla@openai.com

Alex Nichol* OpenAI alex@openai.com

Text-Guided Diffusion Models [Nichol et al 2022]

"a hedgehog using a calculator"

"a corgi wearing a red bowtie and a purple party hat"

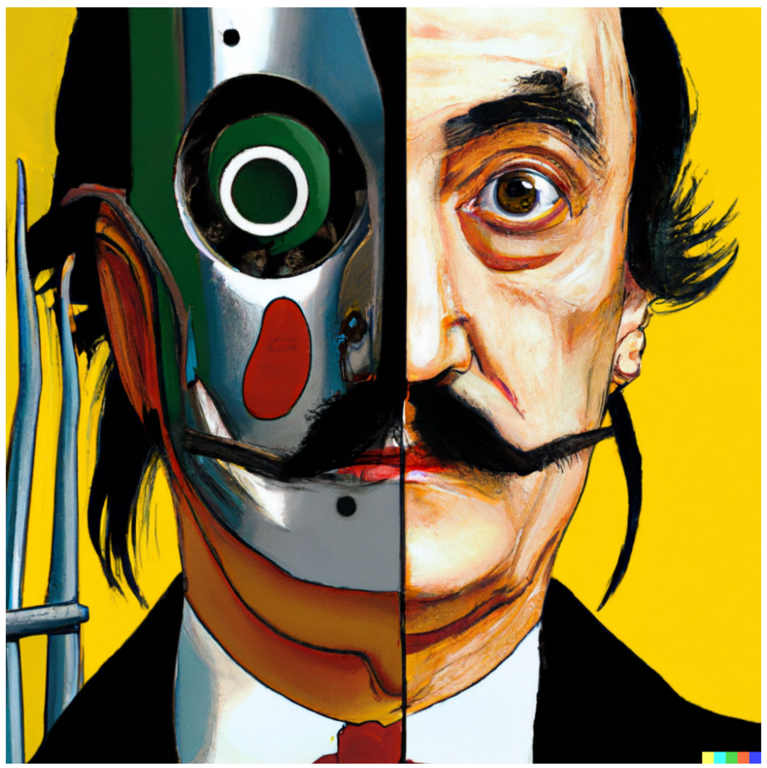
"robots meditating in a vipassana retreat"

"a fall landscape with a small cottage next to a lake"

GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models

Alex Nichol [*] **Prafulla Dhariwal** [*] **Aditya Ramesh** [*] **Pranav Shyam** **Pamela Mishkin** **Bob McGrew** **Ilya Sutskever** **Mark Chen**

DALL-E 2
[Ramesh et al 2022]


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

Hierarchical Text-Conditional Image Generation with CLIP Latents

Aditya Ramesh* OpenAI aramesh@openai.com

Prafulla Dhariwal* OpenAI prafulla@openai.com

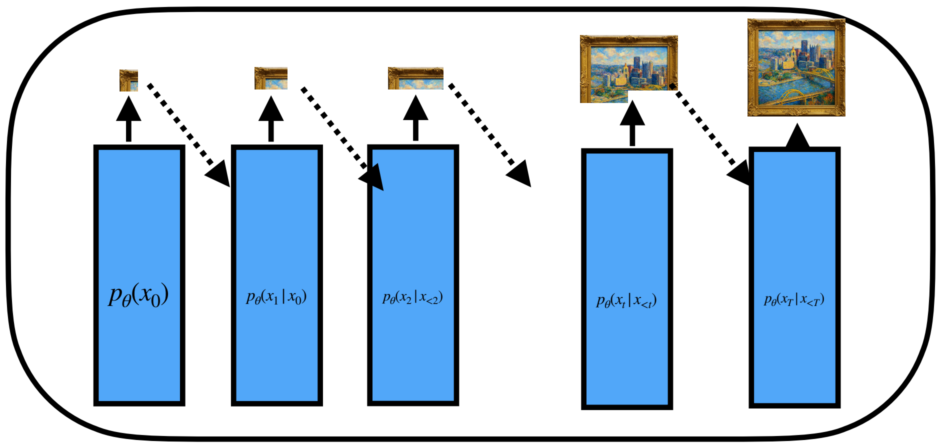
Alex Nichol* OpenAI alex@openai.com

Casey Chu* OpenAI casey@openai.com

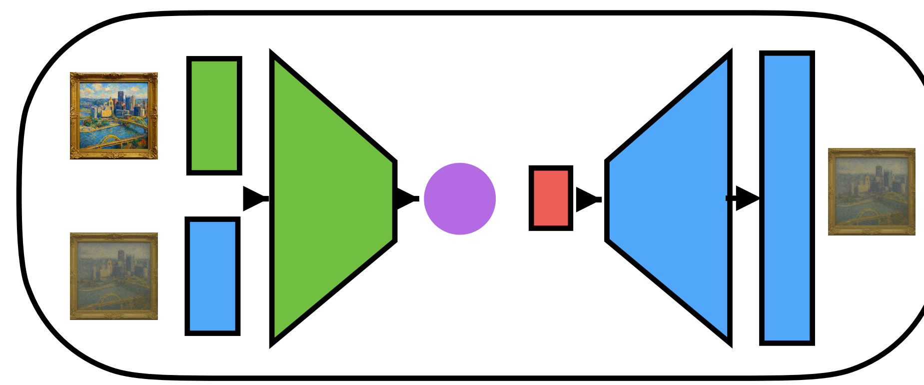
Mark Chen OpenAI mark@openai.com

Recap:

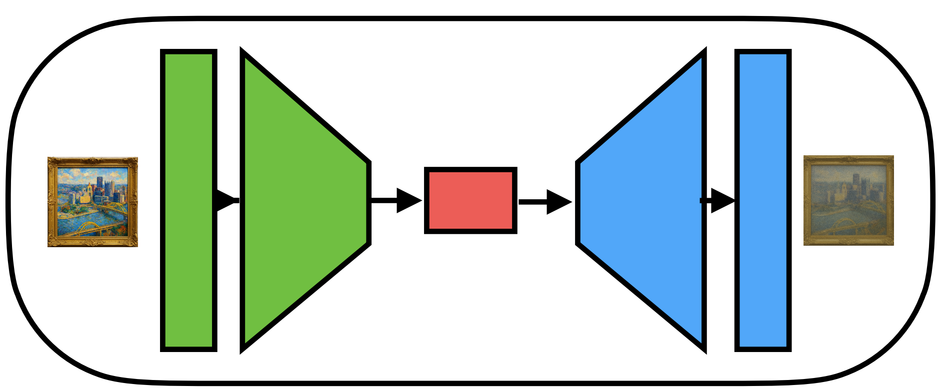
Autoregressive



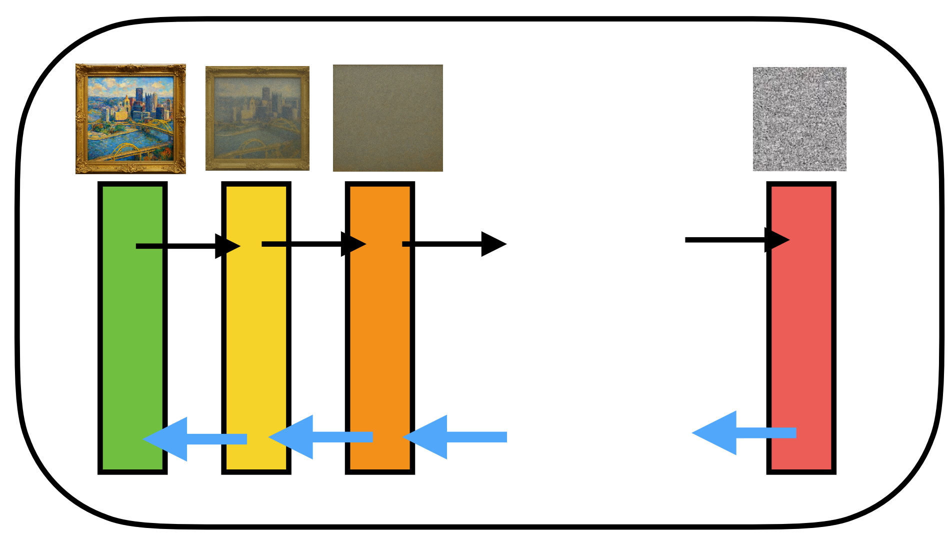
GAN



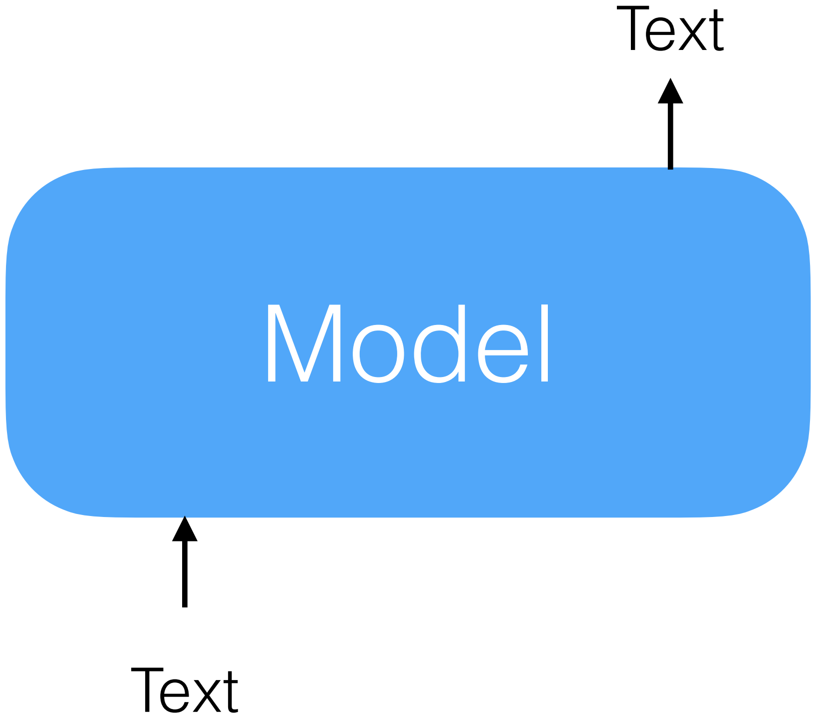
VAE



Diffusion

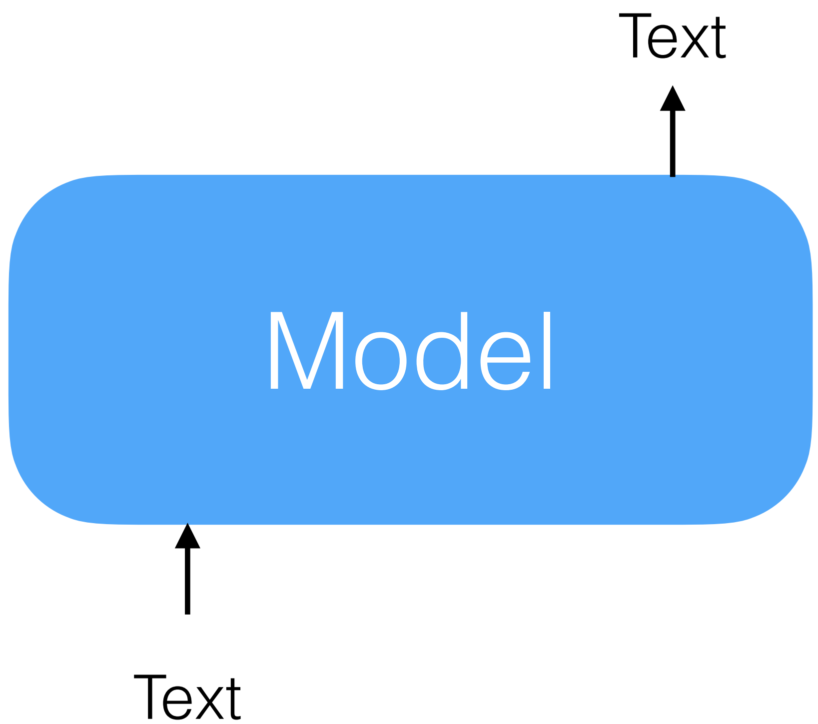


Recap

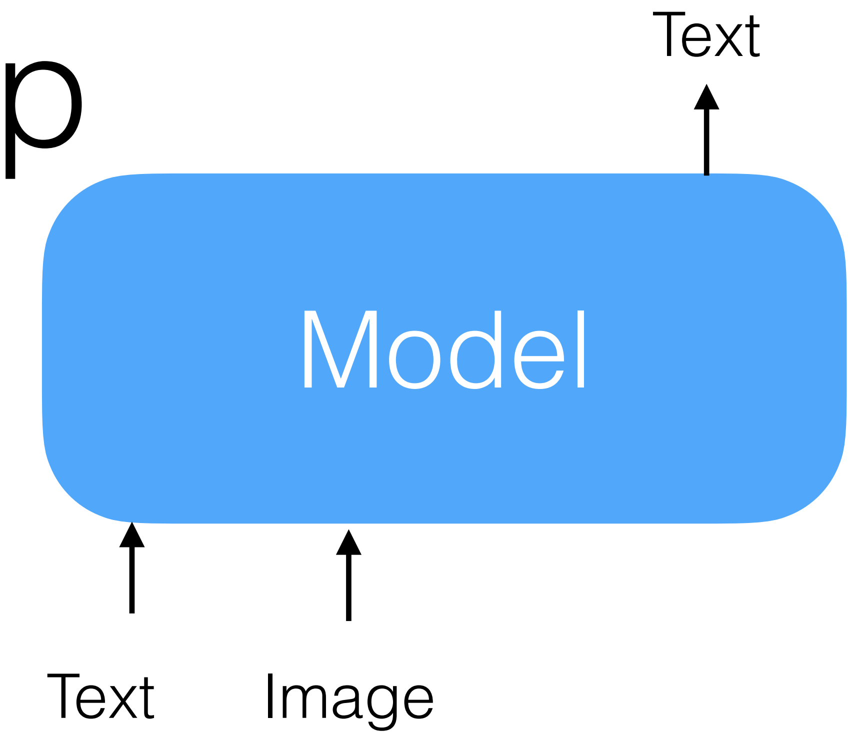


Text-to-text

Recap

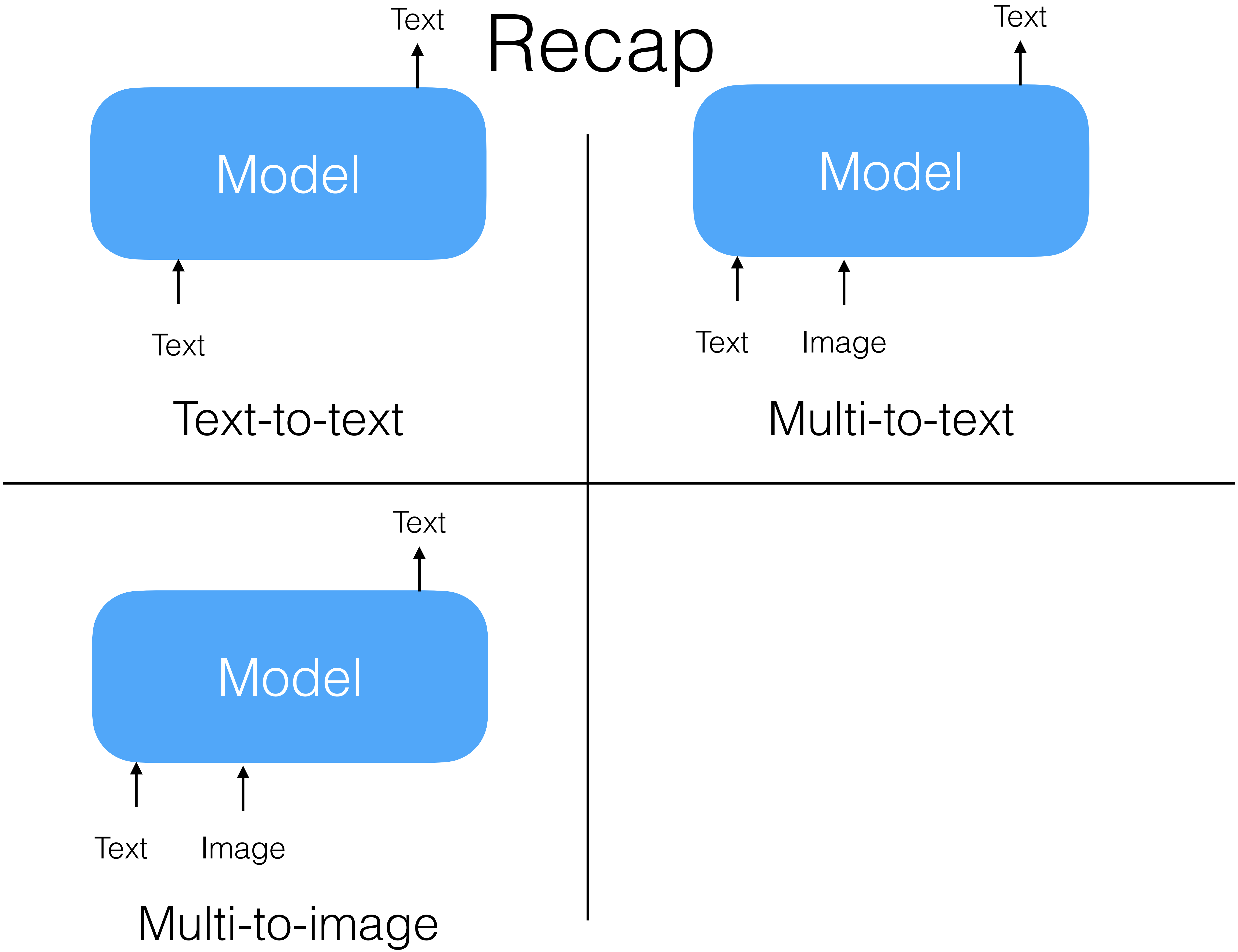


Text-to-text

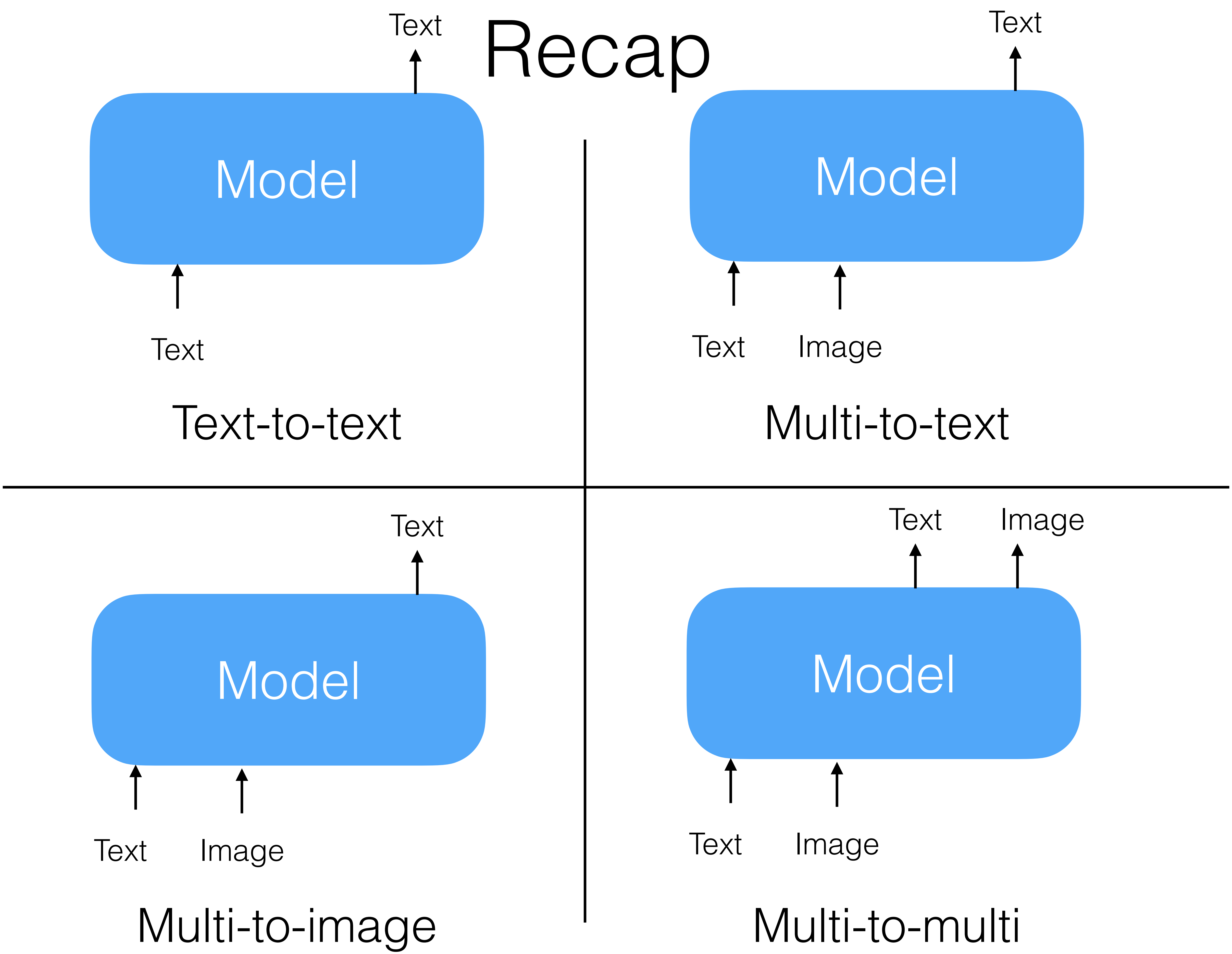


Multi-to-text

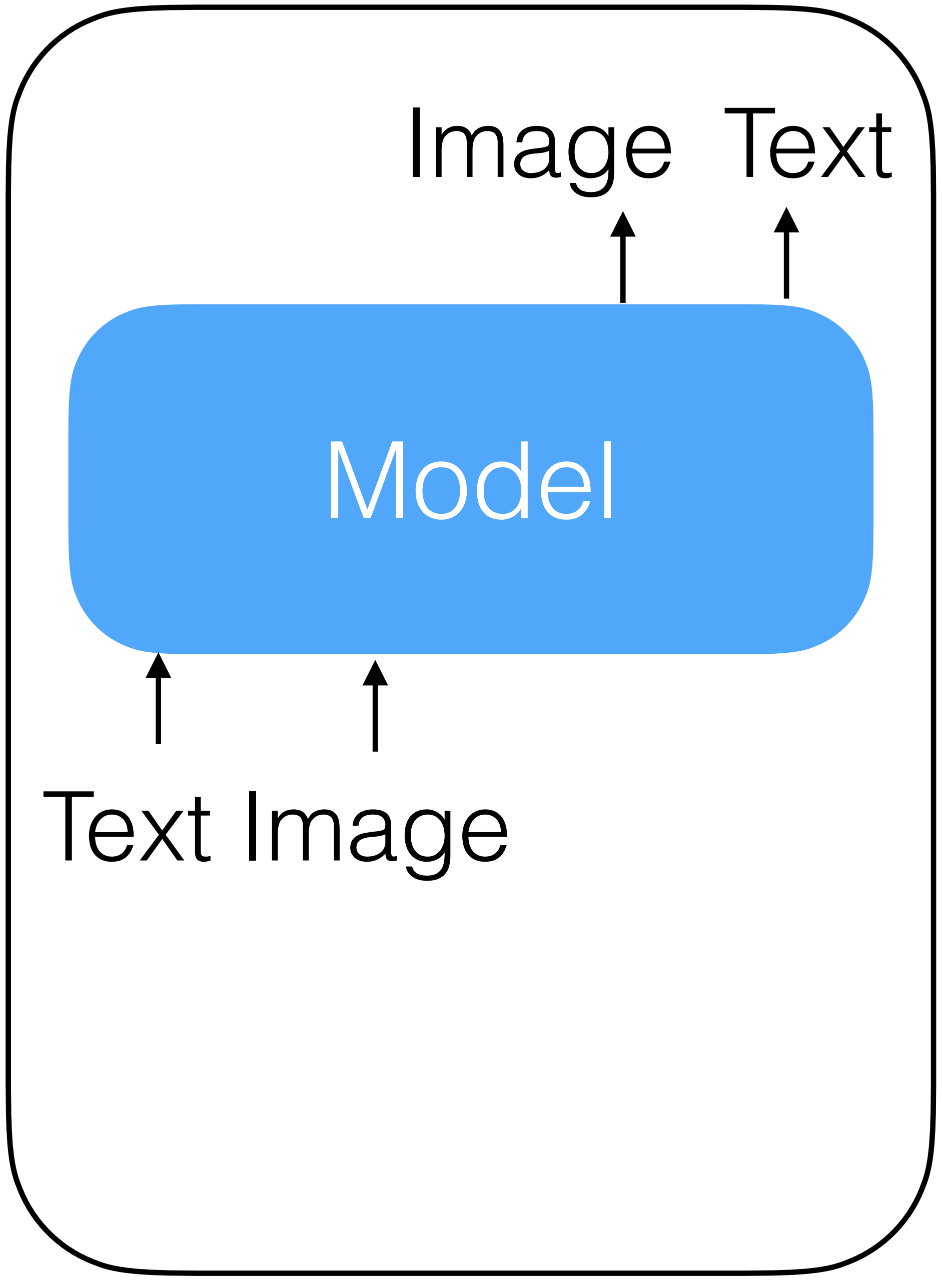
Recap


Text
Model
Text
Text-to-text
Text
Model
Text
Image
Multi-to-text
Text
Model
Text
Image
Multi-to-image

Recap

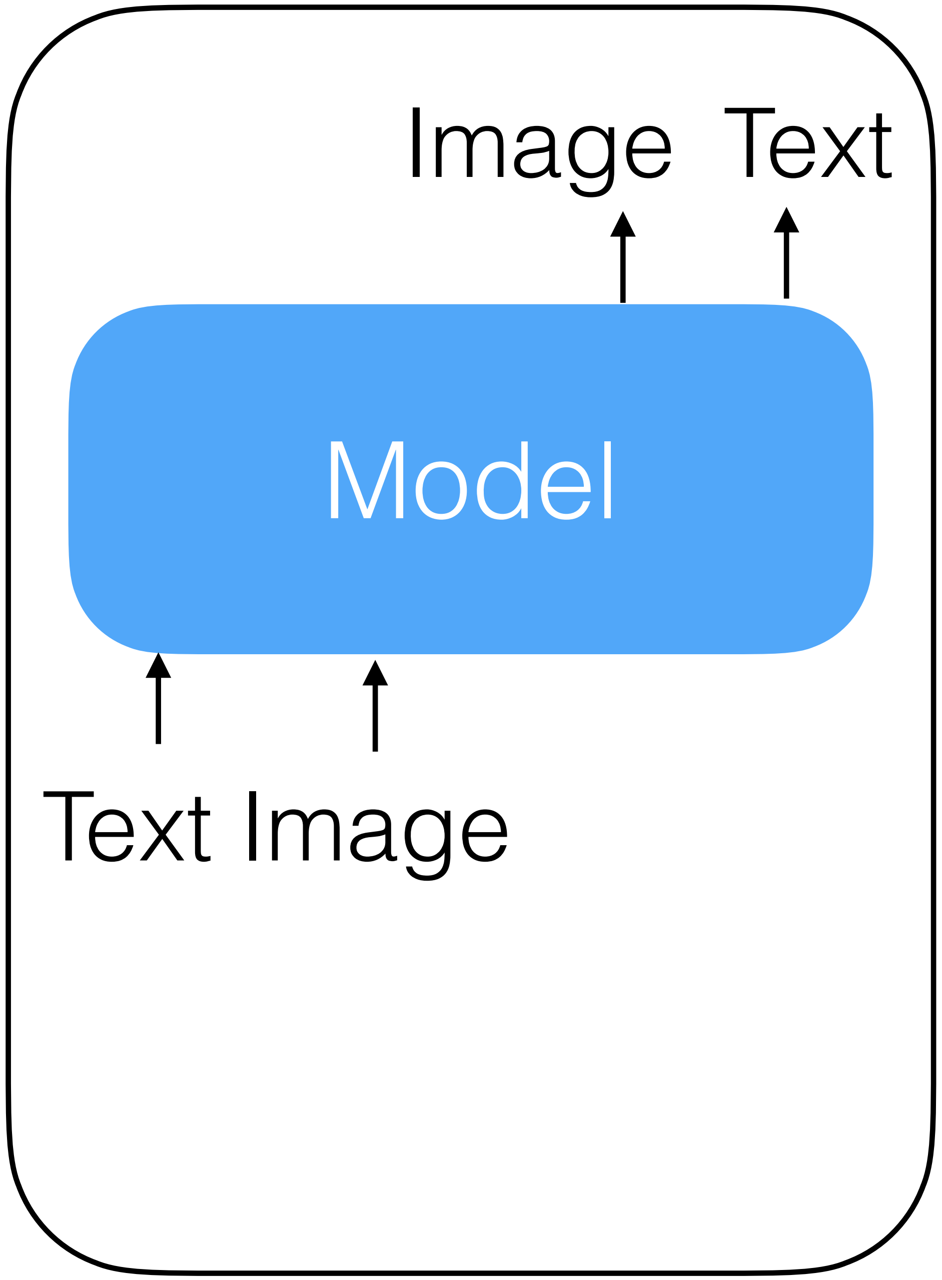

Text
Model
Text
Text-to-text
Text
Model
Text
Image
Multi-to-text
Text
Model
Text
Image
Multi-to-image
Text
Image
Model
Text
Image
Multi-to-multi

Recap



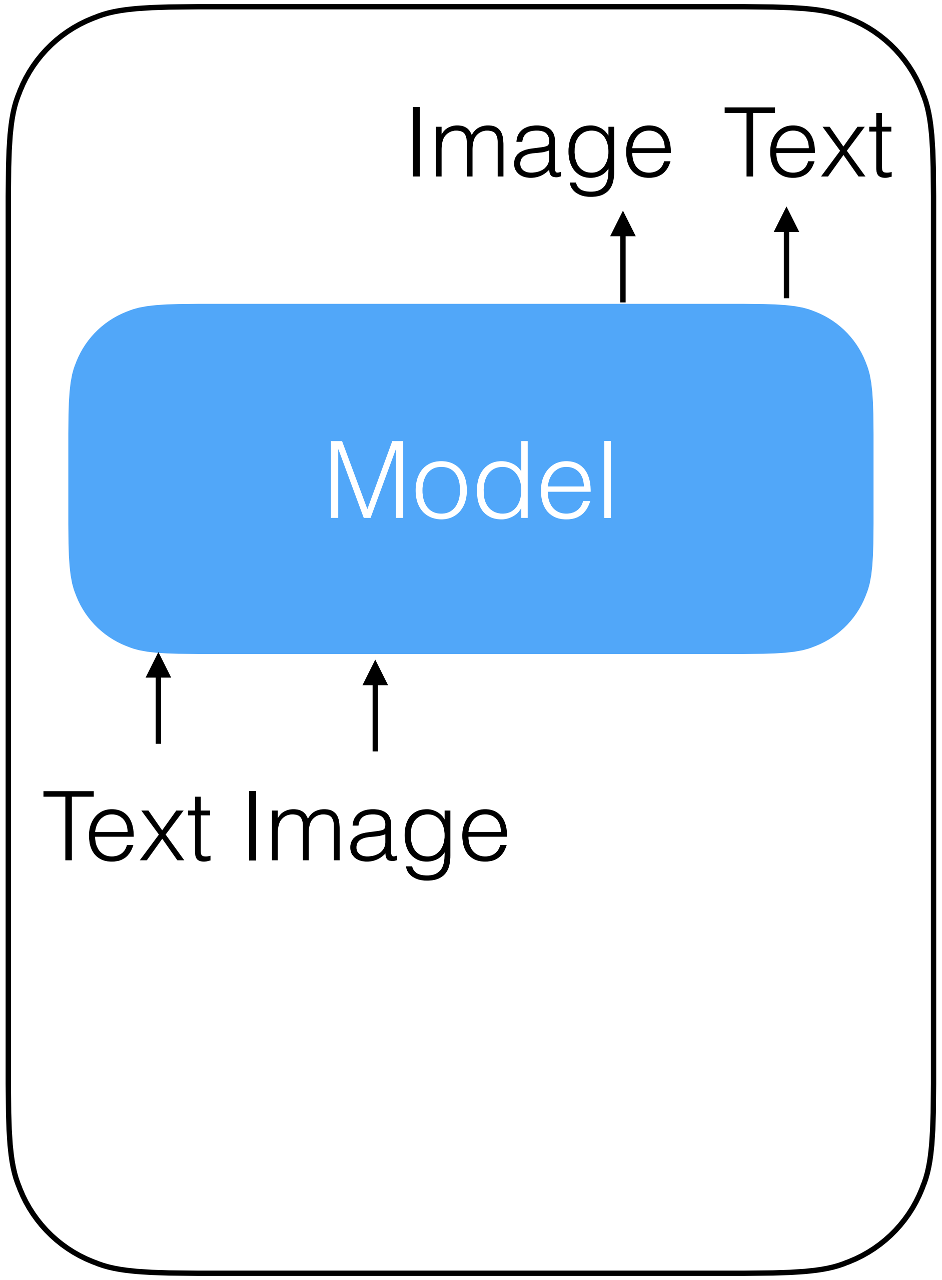
Recap

- Basic generative modeling paradigms



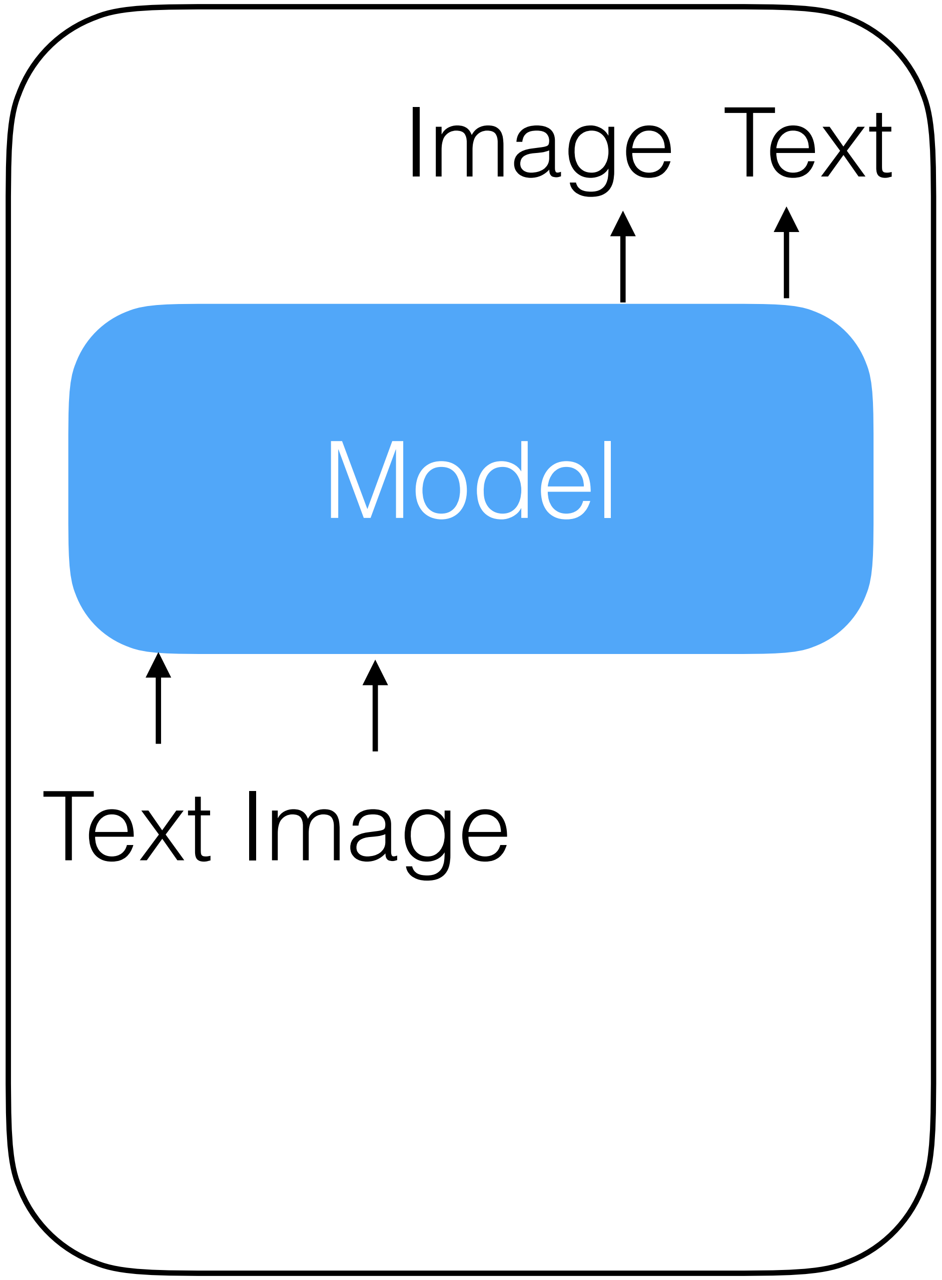
Recap

- Basic generative modeling paradigms
- Autoregressive modeling of pixels



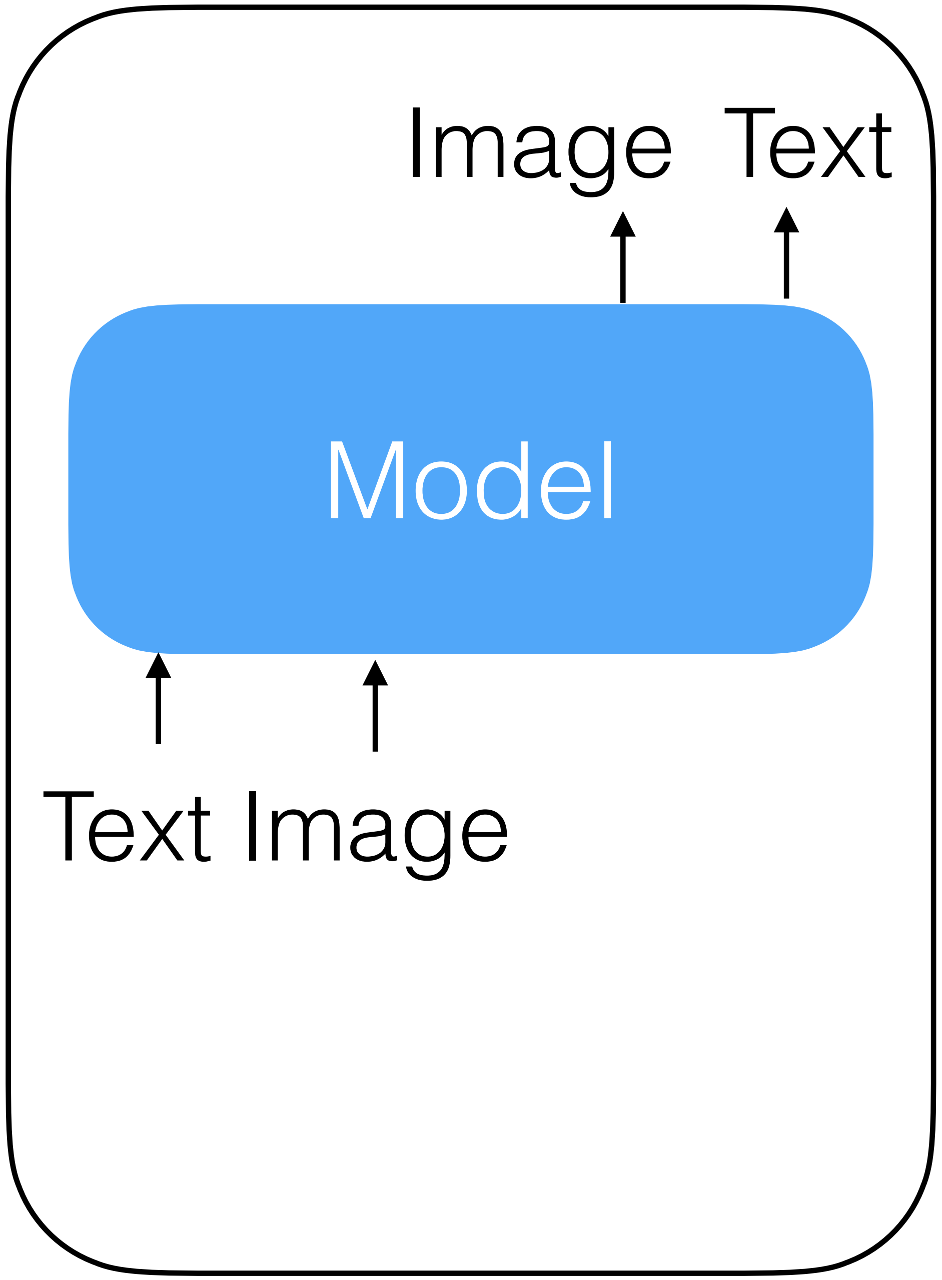
Recap

- Basic generative modeling paradigms
- Autoregressive modeling of pixels
- Autoregressive modeling of “image tokens”



Recap

- Basic generative modeling paradigms
- Autoregressive modeling of pixels
- Autoregressive modeling of “image tokens”
- Quick tour of diffusion references



Thank you!