Recent Trends in Machine Learning: A Large-scale Perspective

A Short Introduction to Multi-modal AI Models (Part 2)

Saehoon Kim @ Kakaobrain

Outline of This Course

CLIP
Encoder-only
05/04

DALL-E
Decoder-only
05/11

DALL-E 2
Enc-Dec
05/18

Outline of This Course

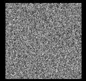
- Contrastive Learning
- Autoregressive Model
- DALL-E 2 Enc-Dec

Autoregressive Models

Image Generation through GAN

Goodfellow et al. Generative Adversarial Nets. NeurIPS 2014.

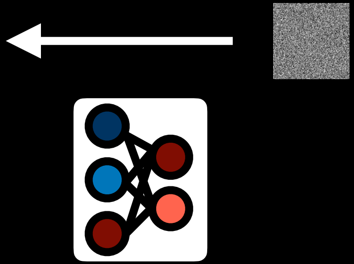
Image Generation through GAN



$$p(\mathbf{z}) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

Goodfellow et al. Generative Adversarial Nets. NeurIPS 2014.

Image Generation through GAN



$$p(\mathbf{z}) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

Goodfellow et al. Generative Adversarial Nets. NeurIPS 2014.

Autoregressive Image Generation

Definition [edit]

The notation $AR(p)$ indicates an autoregressive model of order *p*. The AR(*p*) model is defined as

$$X_t = c + \sum_{i=1}^{p} \varphi_i X_{t-i} + \varepsilon_t$$

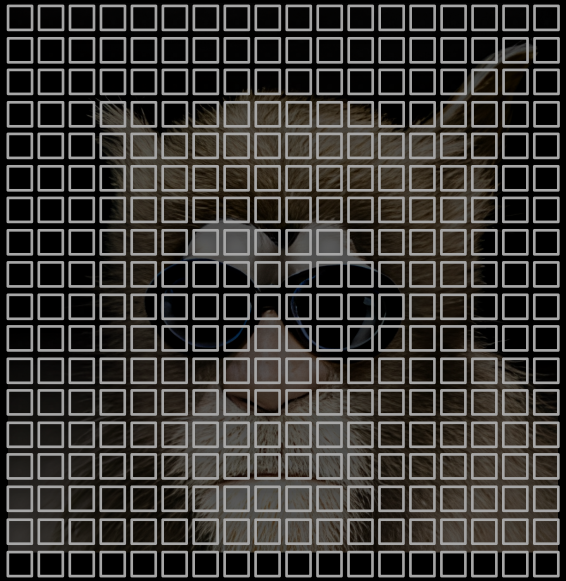
where $\varphi_1, \ldots, \varphi_p$ are the *parameters* of the model, c is a constant, and ε_t is white noise. This can be equivalently written using the backshift operator *B* as

$$X_t = c + \sum_{i=1}^{p} \varphi_i B^i X_t + \varepsilon_t$$

Autoregressive Image Generation

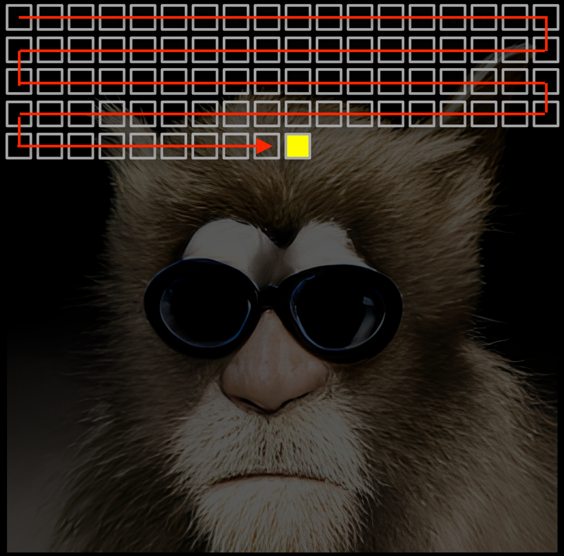
Oord et al. Pixel Recurrent Neural Networks. ICML 2016.

Autoregressive Image Generation



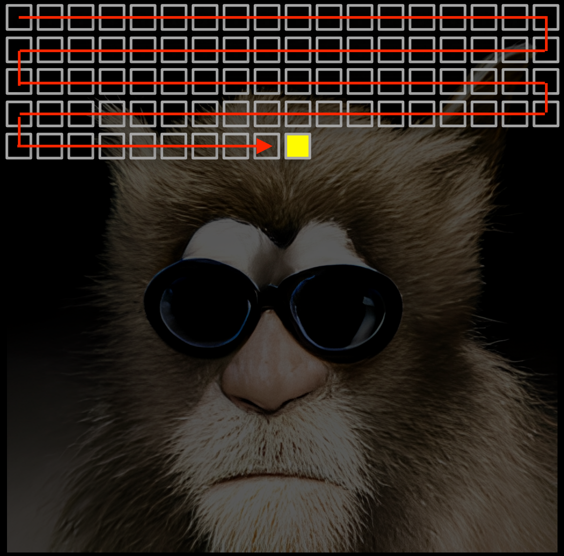
Oord et al. Pixel Recurrent Neural Networks. ICML 2016.

Autoregressive Image Generation



Oord et al. Pixel Recurrent Neural Networks. ICML 2016.

Autoregressive Image Generation

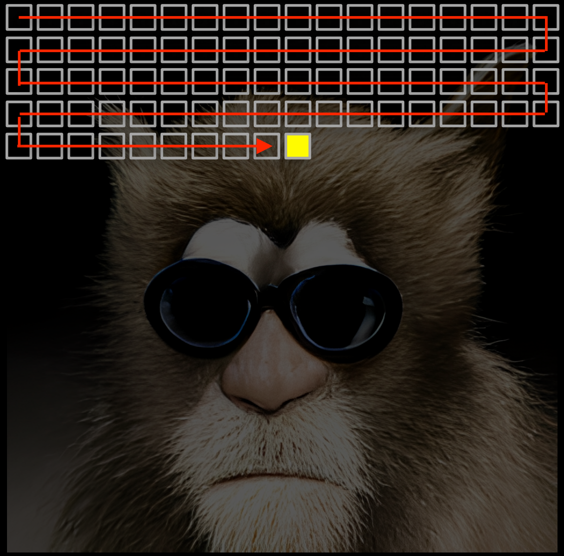


$$p_\theta(x_1, \boxed{x_2}, \cdots, x_N)$$

A single pixel

Oord et al. Pixel Recurrent Neural Networks. ICML 2016.

Autoregressive Image Generation

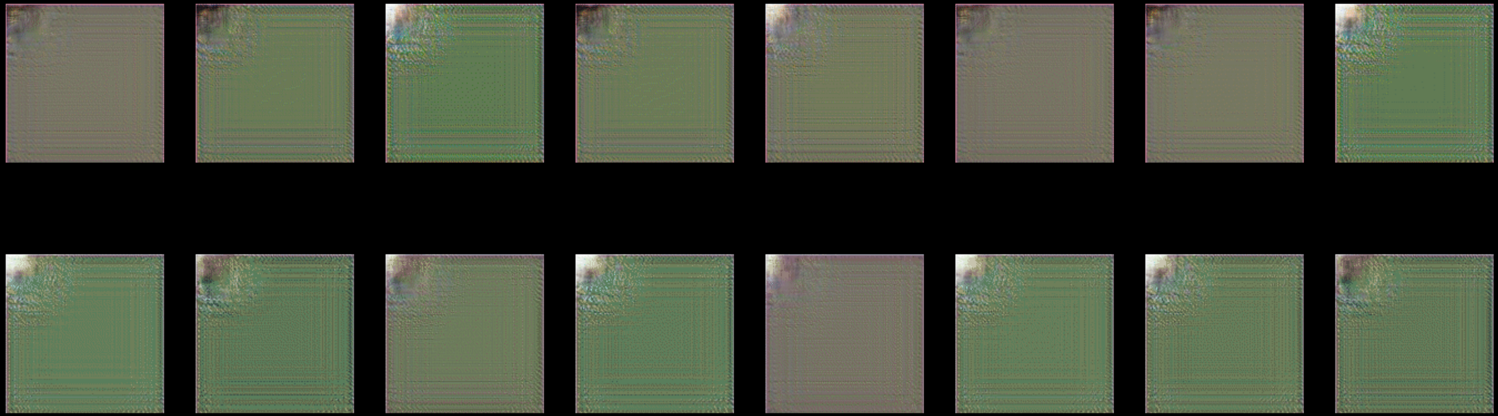


$$p_\theta(x_1, \boxed{x_2}, \cdots, x_N) = \prod_{n=1}^{N} p_\theta(x_n | x_{<n})$$

A single pixel

Oord et al. Pixel Recurrent Neural Networks. ICML 2016.

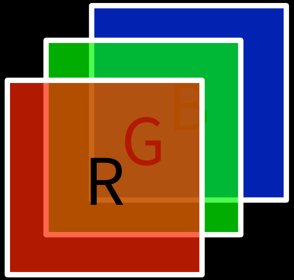
Autoregressive Image Generation



PixelCNN

Oord et al. Pixel Recurrent Neural Networks. ICML 2016.

PixelCNN

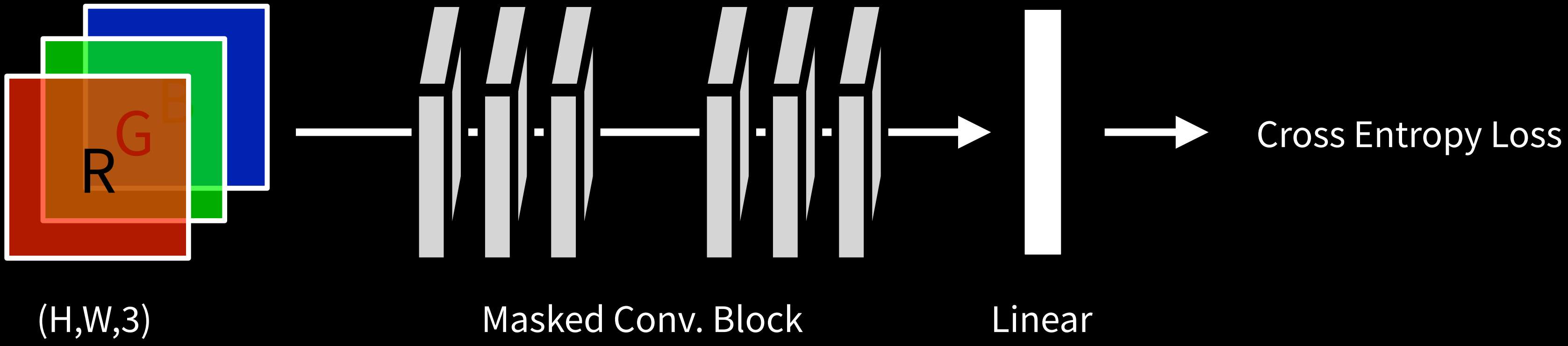


(H,W,3)

Color intensity treated as a **categorical variable**

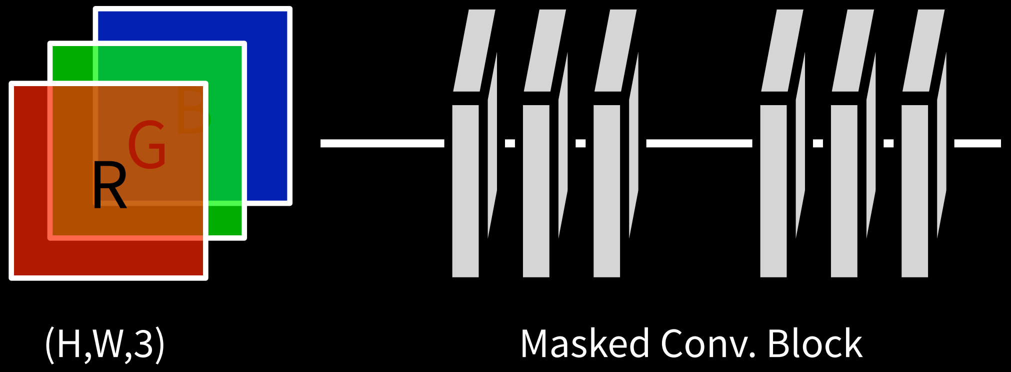
Oord et al. Pixel Recurrent Neural Networks. ICML 2016.

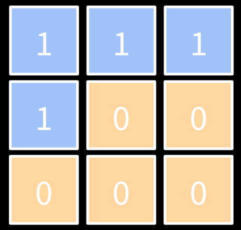
PixelCNN



Oord et al. Pixel Recurrent Neural Networks. ICML 2016.

PixelCNN

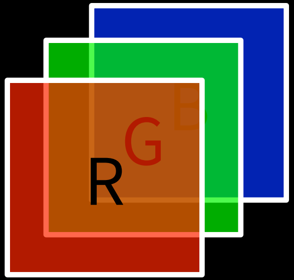




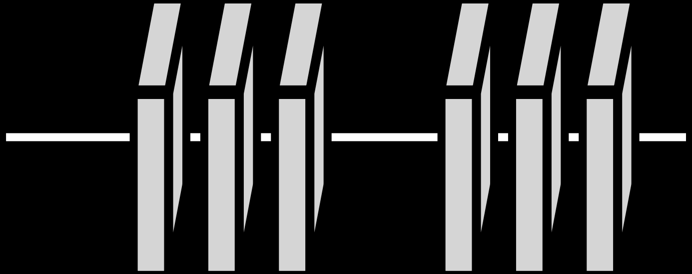
Masked conv. filter

Oord et al. Pixel Recurrent Neural Networks. ICML 2016.

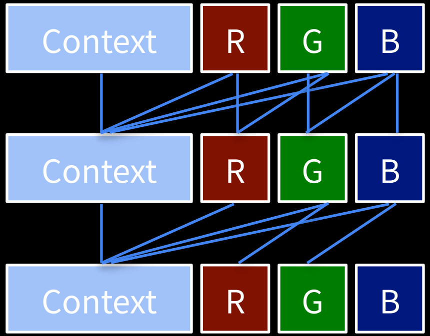
PixelCNN



(H,W,3)



Masked Conv. Block



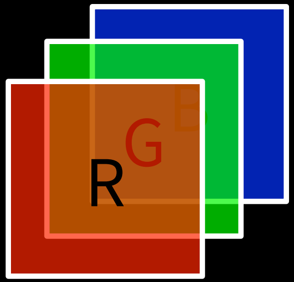
Mask for color channels

Oord et al. Pixel Recurrent Neural Networks. ICML 2016.

PixelCNN

Oord et al. Pixel Recurrent Neural Networks. ICML 2016.

PixelCNN++

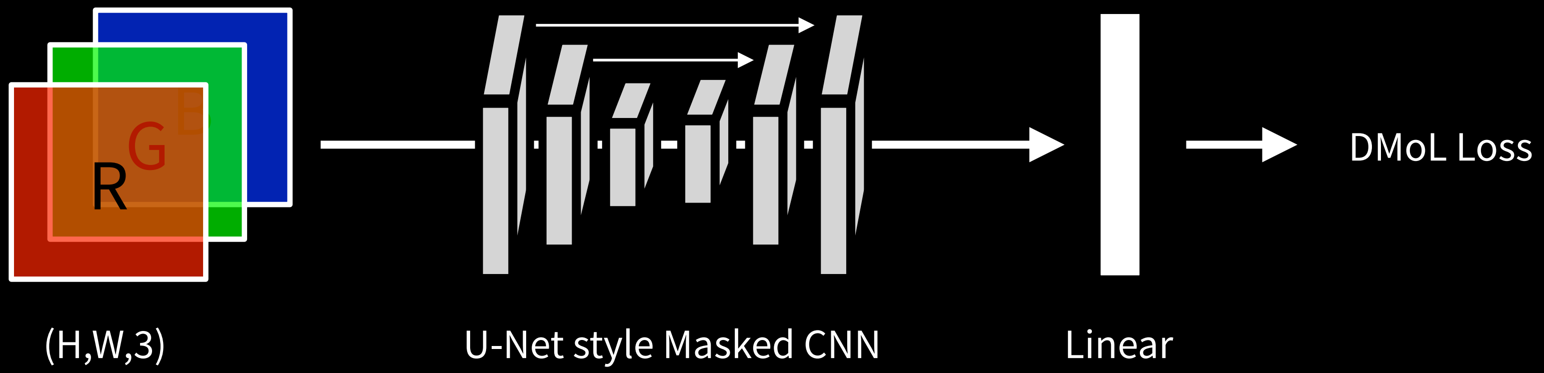


(H,W,3)

Color intensity treated as an **ordinal variable**

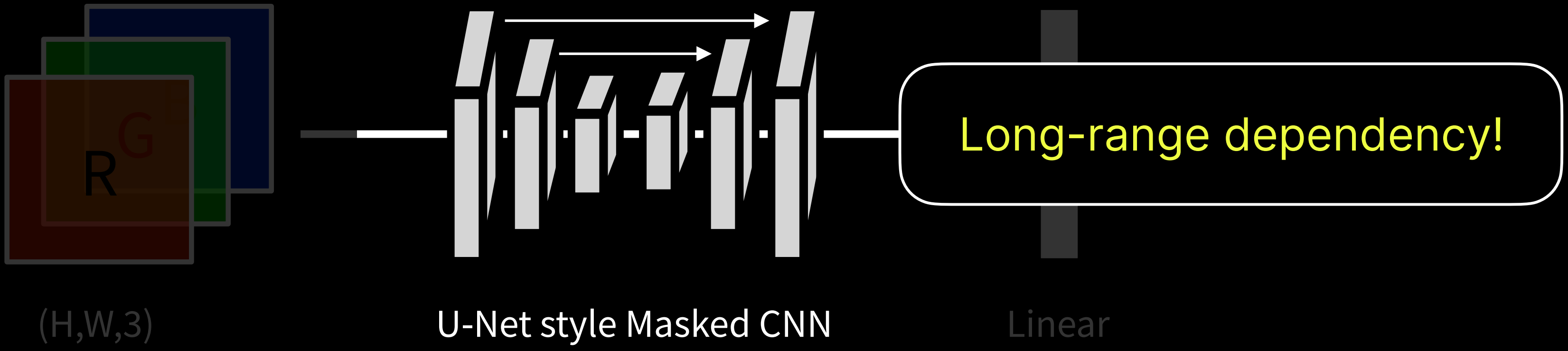
Salimans et al. PixelCNN++: A Pixel CNN Implementation with DMoL and Other Modifications, ICLR 2017.

PixelCNN++



Oord et al. Pixel Recurrent Neural Networks. ICML 2016.

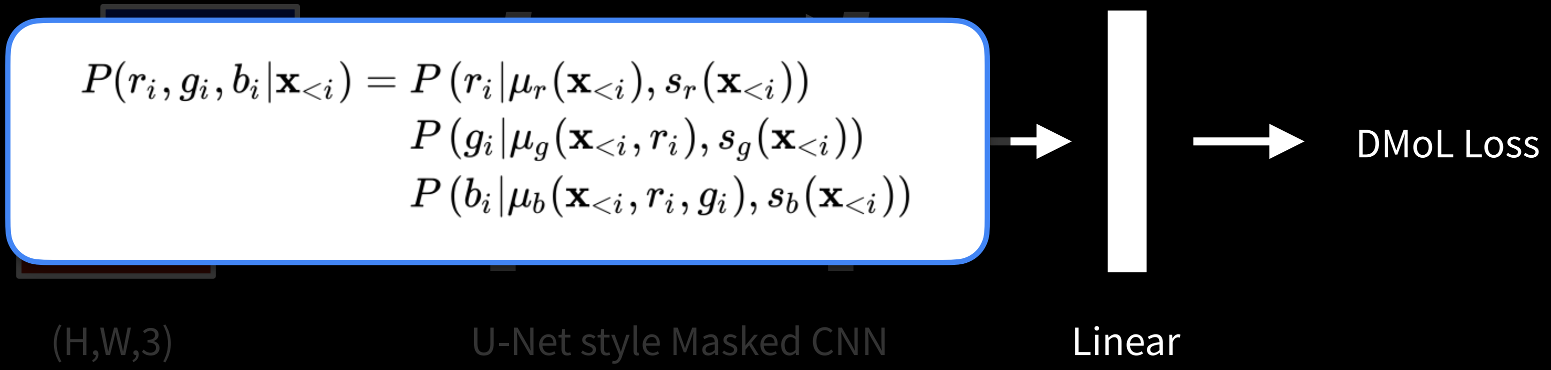
PixelCNN++



Oord et al. Pixel Recurrent Neural Networks. ICML 2016.

PixelCNN++

$$
\begin{aligned}
P(r_i, g_i, b_i | \mathbf{x}_{<i}) = & \; P\left(r_i | \mu_r(\mathbf{x}_{<i}), s_r(\mathbf{x}_{<i})\right) \\
& \; P\left(g_i | \mu_g(\mathbf{x}_{<i}, r_i), s_g(\mathbf{x}_{<i})\right) \\
& \; P\left(b_i | \mu_b(\mathbf{x}_{<i}, r_i, g_i), s_b(\mathbf{x}_{<i})\right)
\end{aligned}
$$

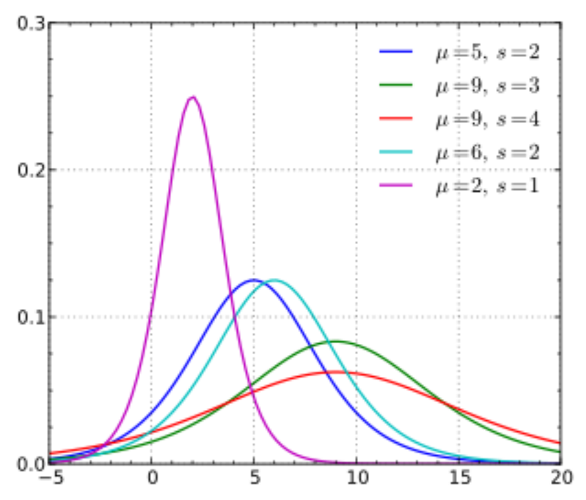


Oord et al. Pixel Recurrent Neural Networks. ICML 2016.

Discretized Mixture of Logistic Loss

"Assume there is a latent color intensity v with a continuous distribution, rounded to its nearest 8-bit representation to give the observed x"

$$v \sim \sum_{i=1}^{K} \pi_i \operatorname{logistic}(\mu_i, s_i)$$



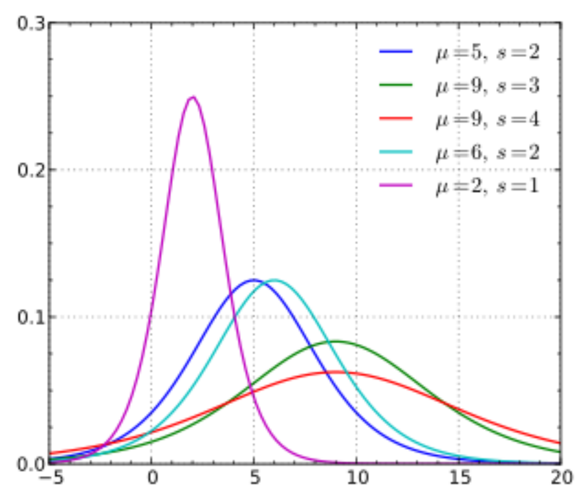
Oord et al. Pixel Recurrent Neural Networks. ICML 2016.

Discretized Mixture of Logistic Loss

"Assume there is a latent color intensity v with a continuous distribution, rounded to its nearest 8-bit representation to give the observed x"

$$\text{CDF-logistic} = \frac{1}{1 + \exp(-(x - \mu)/s)}$$

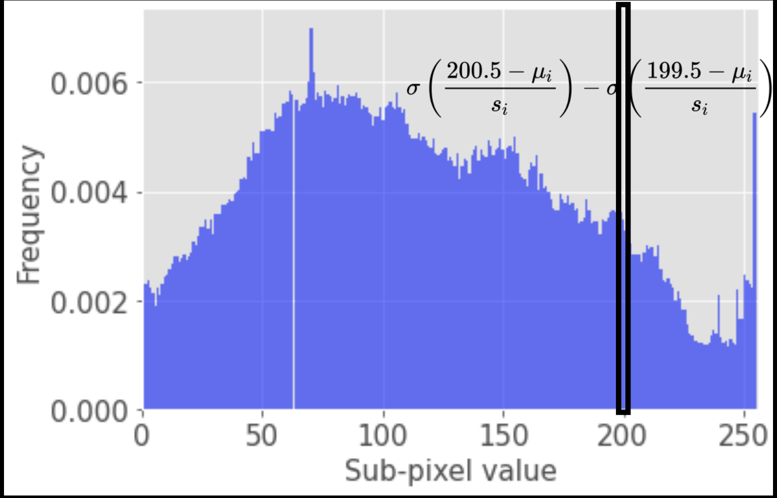
$$\triangleq \sigma((x - \mu)/s)$$



Oord et al. Pixel Recurrent Neural Networks. ICML 2016.

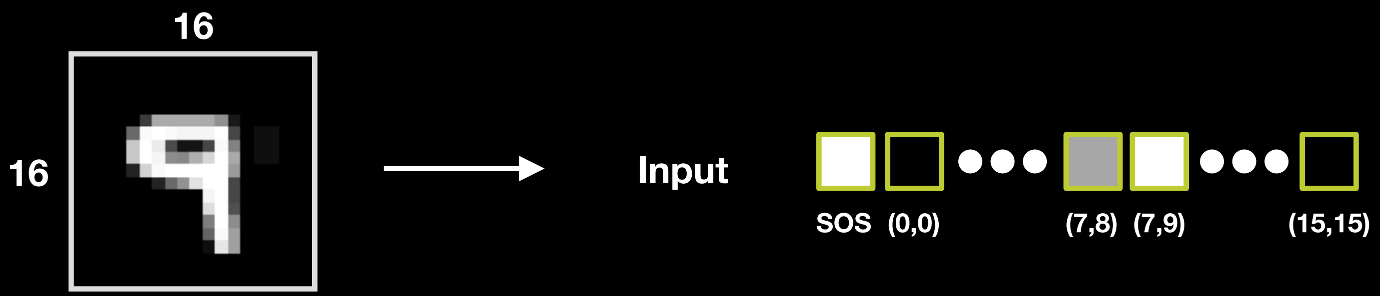
Discretized Mixture of Logistic Loss

"Assume there is a latent color intensity v with a continuous distribution, rounded to its nearest 8-bit representation to give the observed x"



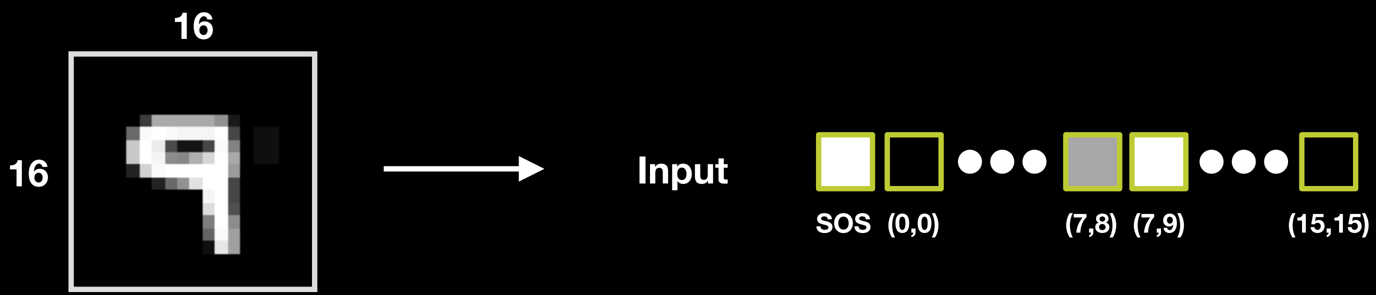
Oord et al. Pixel Recurrent Neural Networks. ICML 2016.

Image Transformer



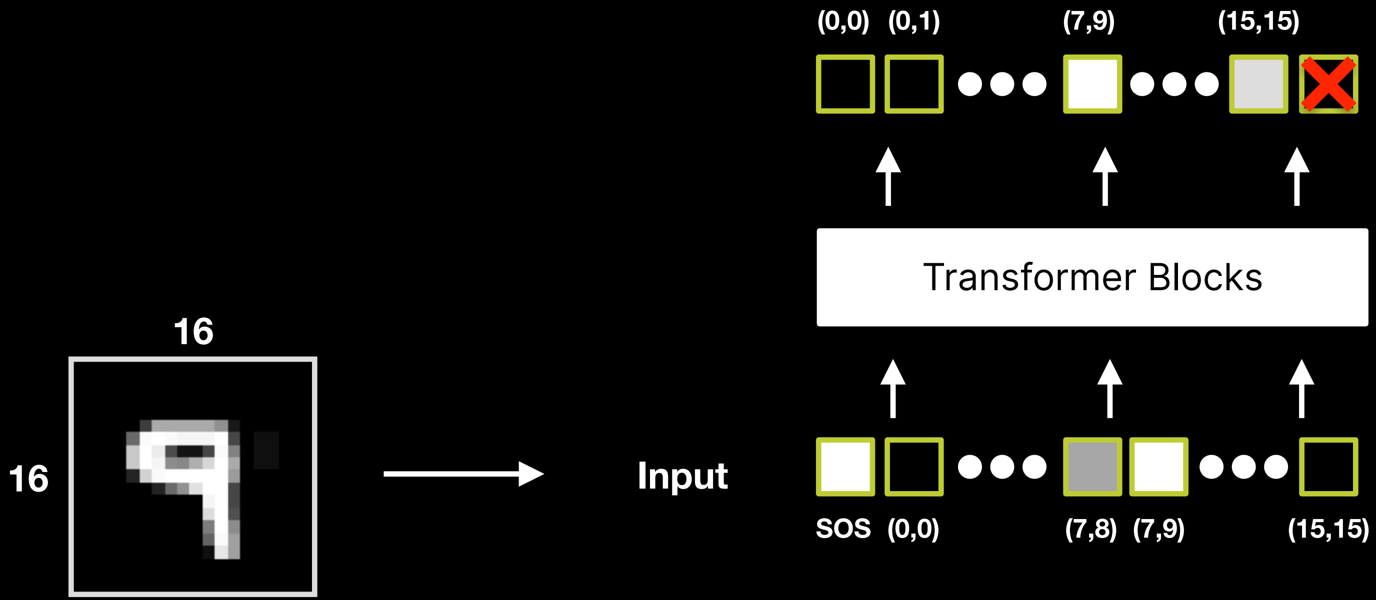
Parma et al. Image Transformer, ICML’18.

Image Transformer



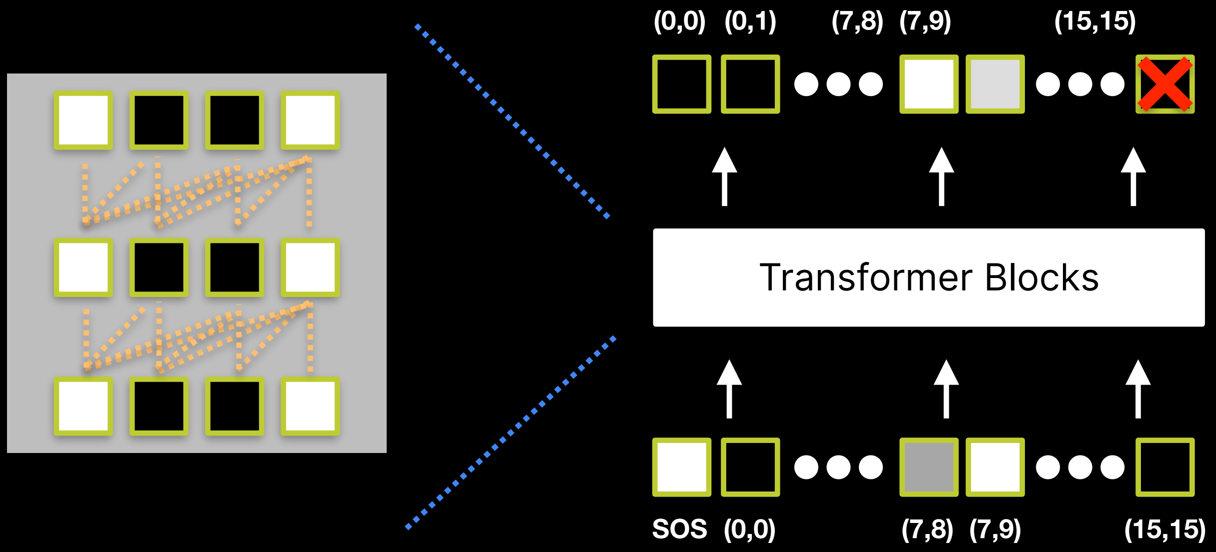
Parma et al. Image Transformer, ICML'18.

Image Transformer



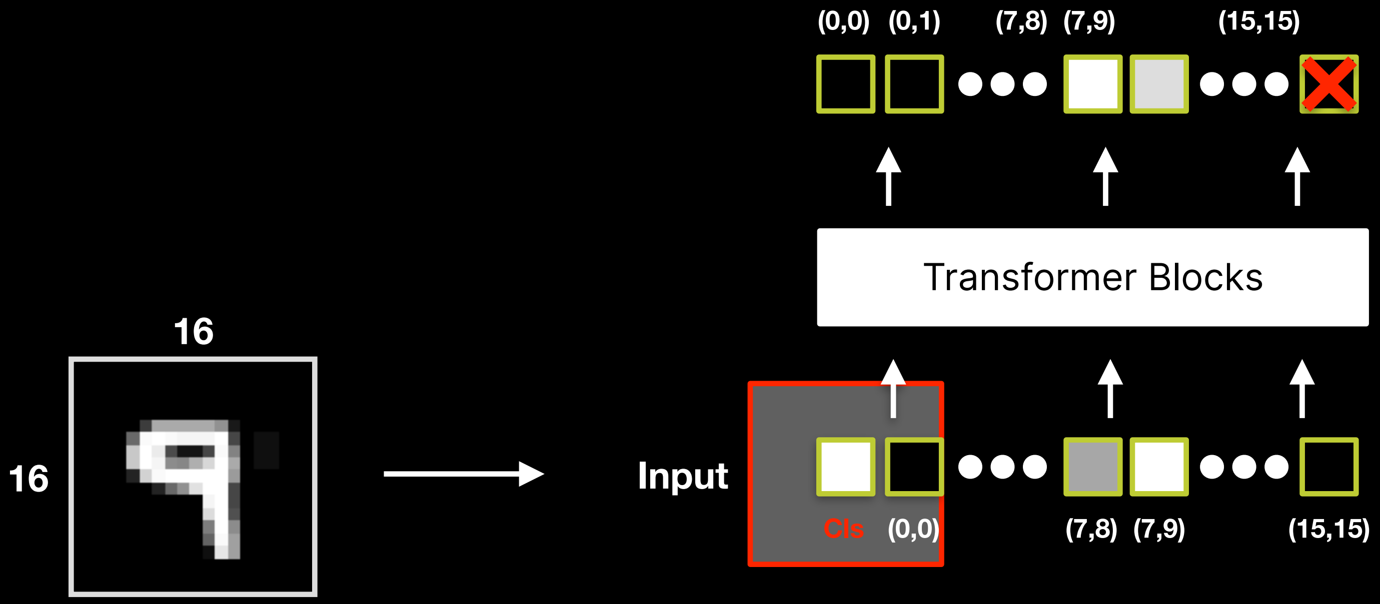
Parma et al. Image Transformer, ICML'18.

Image Transformer



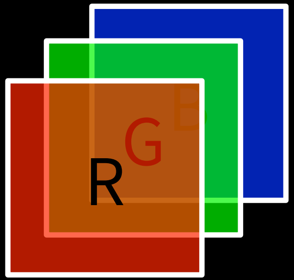
Parma et al. Image Transformer, ICML'18.

Image Transformer (class-conditional)



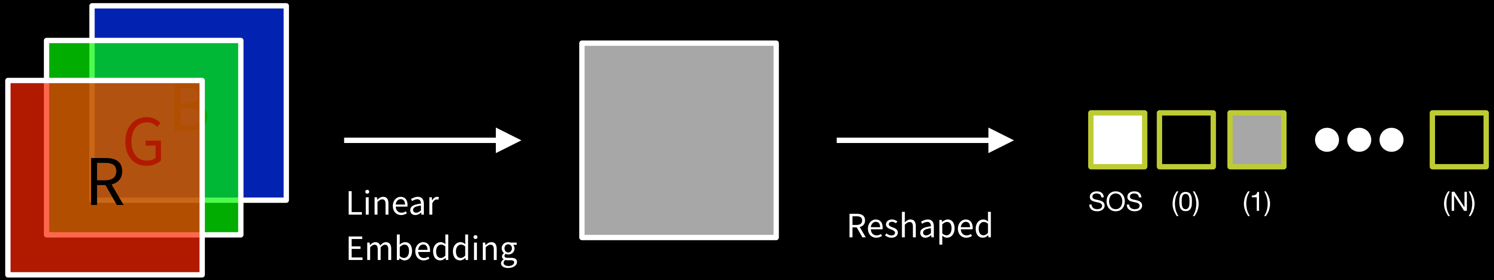
Parma et al. Image Transformer, ICML'18.

Image Transformer (RGB)



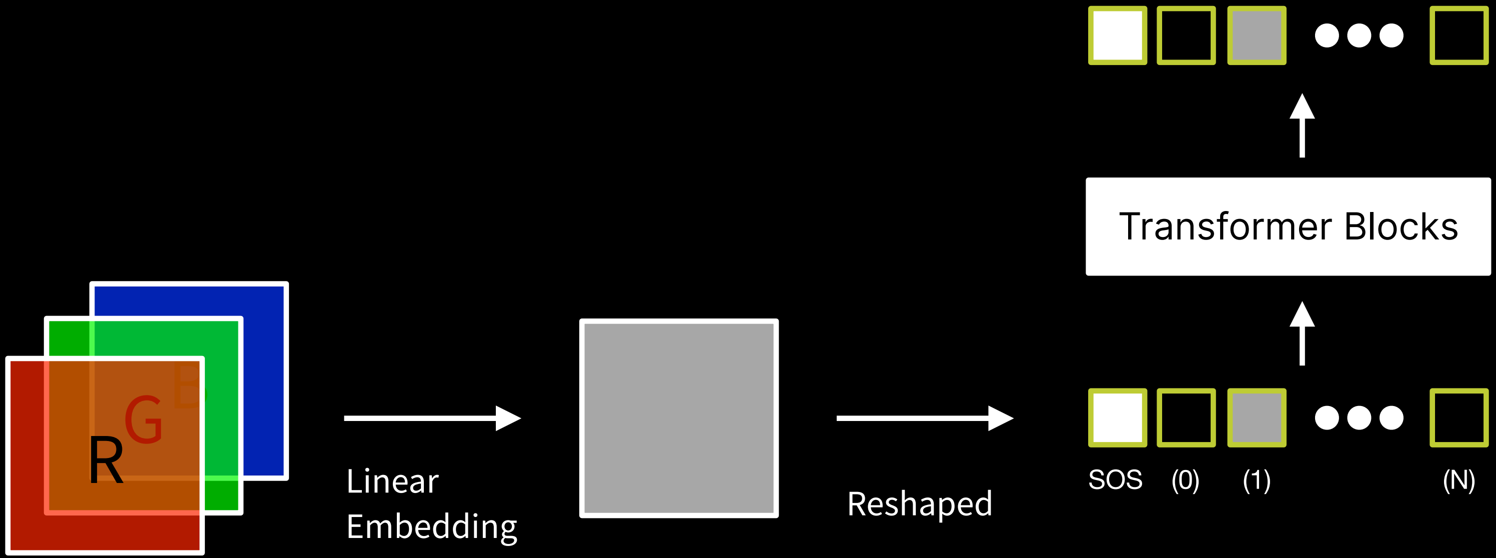
Parma et al. Image Transformer, ICML'18.

Image Transformer (RGB)



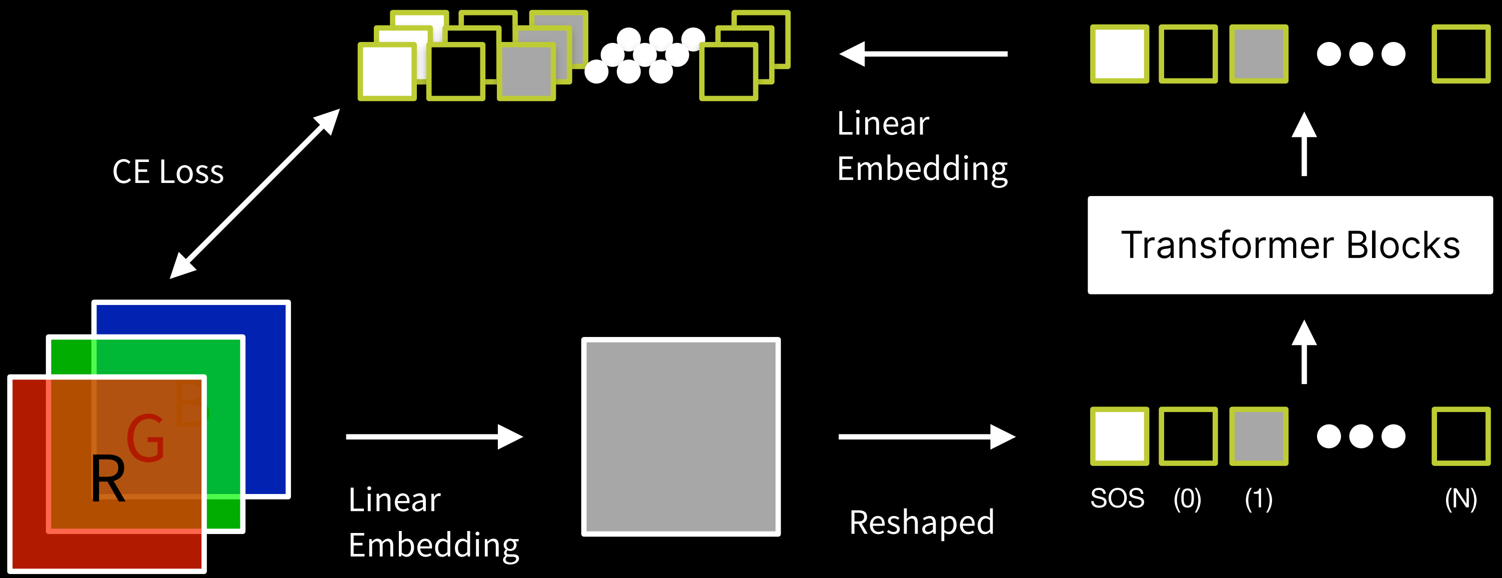
Parma et al. Image Transformer, ICML'18.

Image Transformer (RGB)



Parma et al. Image Transformer, ICML'18.

Image Transformer (RGB)



Parma et al. Image Transformer, ICML'18.

Pixel-level AR Generation

256

256

$$P(x_{1,1}, x_{1,2}, \ldots, x_{256,256}) = ?$$

Pixel-level AR Generation

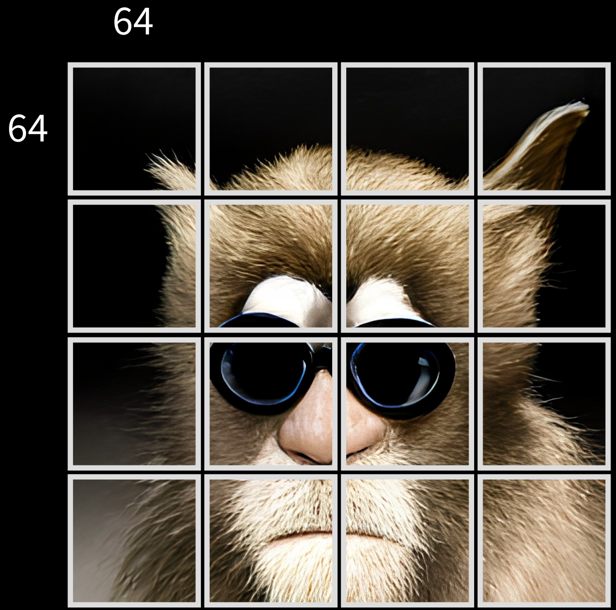
256

256

$$P(x_{1,1}, x_{1,2}, \ldots, x_{256,256}) = ?$$

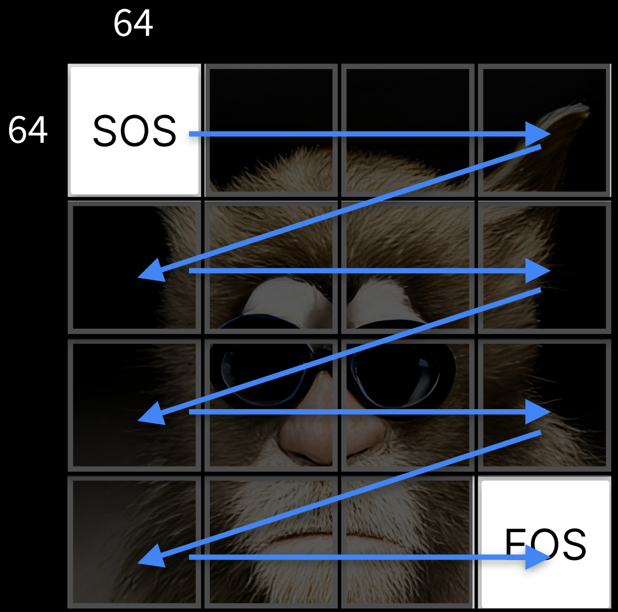
Sequence length = 65K !?

Patch-level AR Generation



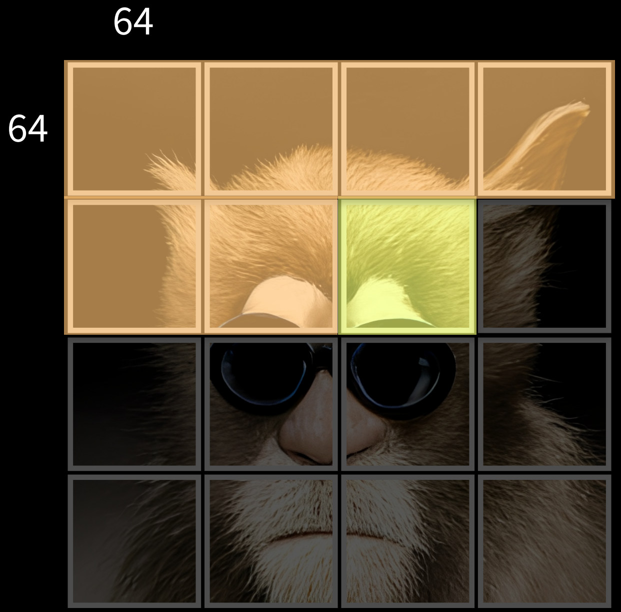
$$P(x_1, x_1, \ldots, x_{16})$$

Patch-level AR Generation



$$P(x_1, x_1, \ldots, x_{16})$$

Patch-level AR Generation



$$P(x_1, x_1, \ldots, x_{16}) = \prod_m P(x_m \mid x_{<m})$$

VQ(Vector Quantization)-VAE

Original Image

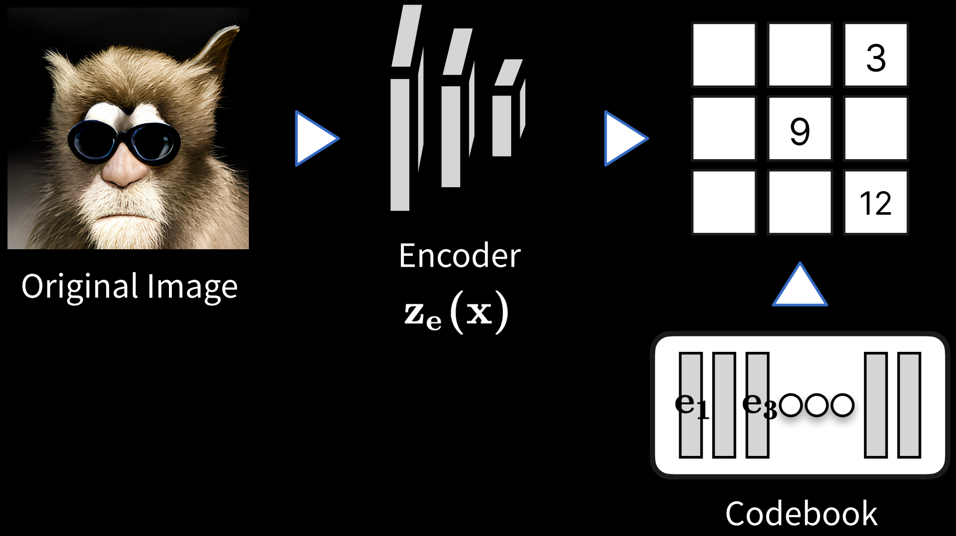
Oord et al. Neural Discrete Representation Learning. NeurIPS 2017.

VQ(Vector Quantization)-VAE

Original Image

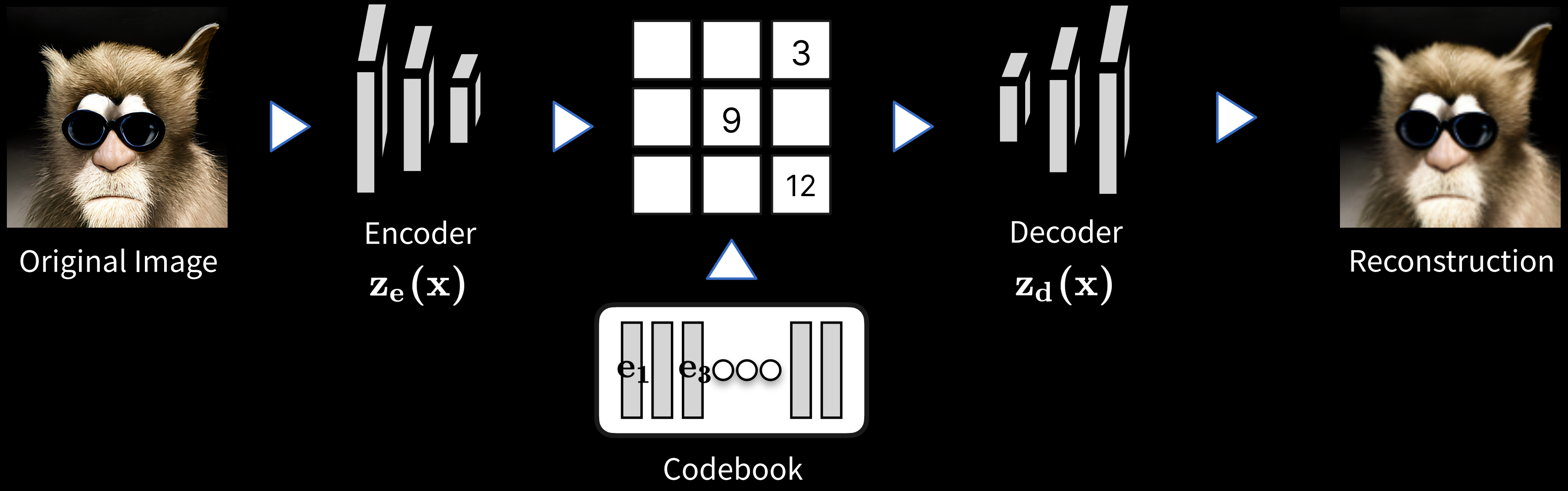
Oord et al. Neural Discrete Representation Learning. NeurIPS 2017.

VQ(Vector Quantization)-VAE



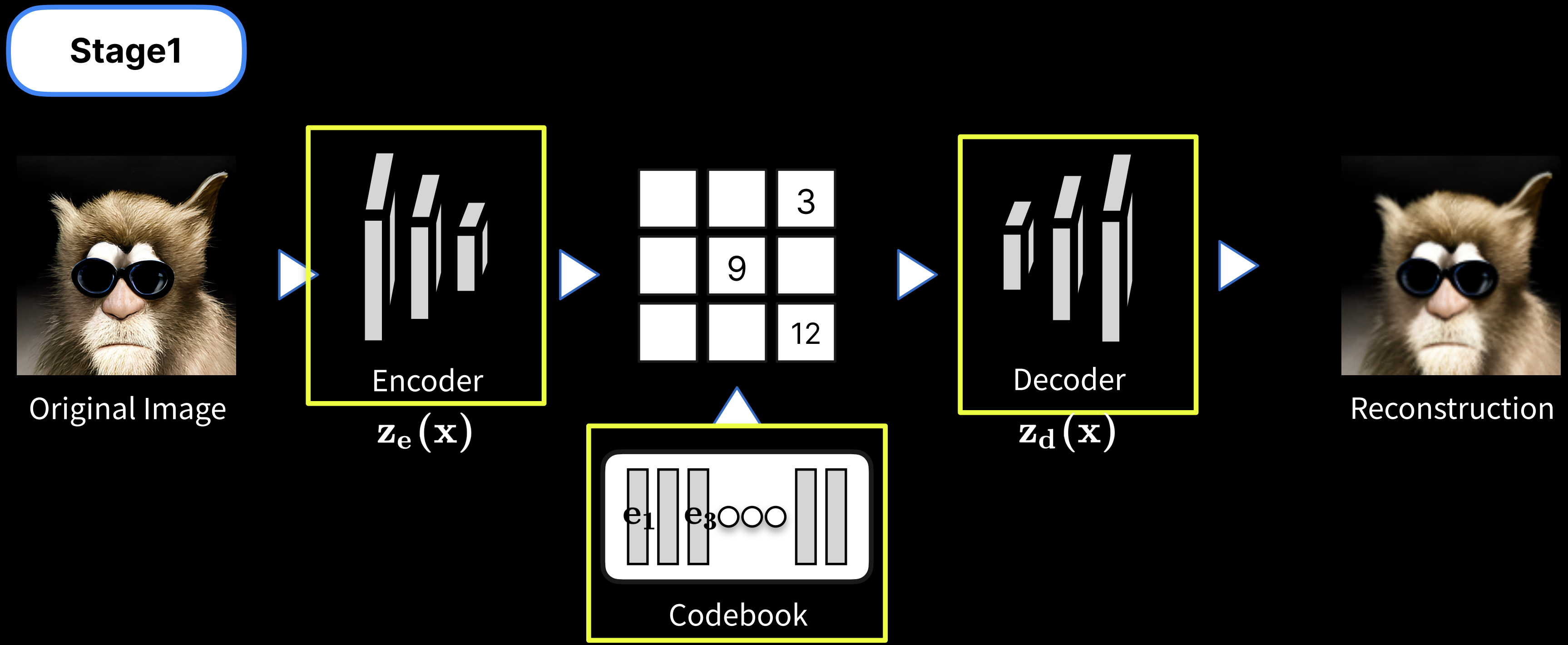
Oord et al. Neural Discrete Representation Learning. NeurIPS 2017.

VQ(Vector Quantization)-VAE



Oord et al. Neural Discrete Representation Learning. NeurIPS 2017.

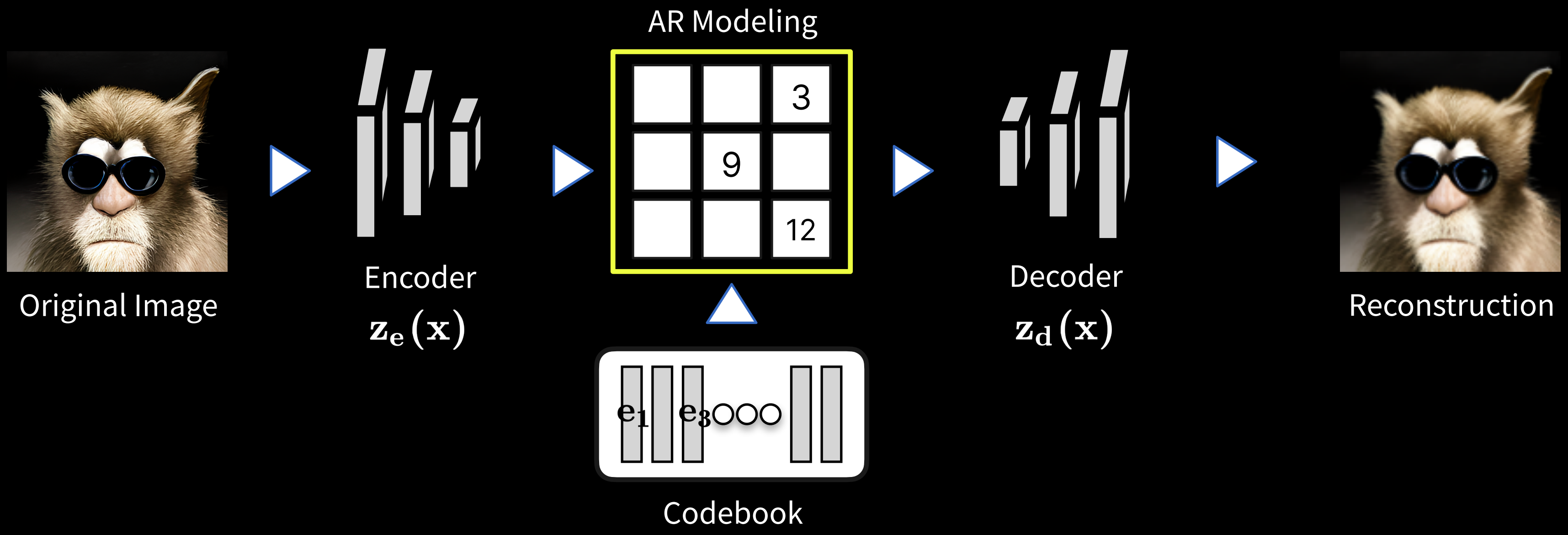
VQ(Vector Quantization)-VAE



Oord et al. Neural Discrete Representation Learning. NeurIPS 2017.

VQ(Vector Quantization)-VAE

Stage2



Oord et al. Neural Discrete Representation Learning. NeurIPS 2017.

VQ-VAE Formulation

$$\mathcal{L} = \log p(\mathbf{x}|\mathbf{z}_d(\mathbf{e})) + \beta\|\mathbf{z}_e(\mathbf{x}) - \mathrm{sg}[\mathbf{e}]\|_2^2$$

$$+ \|\mathrm{sg}[\mathbf{z}_e(\mathbf{x})] - \mathbf{e}\|_2^2$$

Oord et al. Neural Discrete Representation Learning. NeurIPS 2017.

VQ-VAE Formulation

$$\mathcal{L} = \underbrace{\log p(\mathbf{x}|\mathbf{z}_d(\mathbf{e}))}_{\text{Reconstruction loss}} + \beta\|\mathbf{z}_e(\mathbf{x}) - \mathrm{sg}[\mathbf{e}]\|_2^2$$

$$+ \|\mathrm{sg}[\mathbf{z}_e(\mathbf{x})] - \mathbf{e}\|_2^2$$

Oord et al. Neural Discrete Representation Learning. NeurIPS 2017.

VQ-VAE Formulation

$$\mathcal{L} = \underbrace{\log p(\mathbf{x}|\mathbf{z}_d(\mathbf{e}))}_{\text{Reconstruction loss}} + \underbrace{\beta\|\mathbf{z}_e(\mathbf{x}) - \mathrm{sg}[\mathbf{e}]\|_2^2}_{\text{Commitment loss}}$$

$$+ \|\mathrm{sg}[\mathbf{z}_e(\mathbf{x})] - \mathbf{e}\|_2^2$$

Oord et al. Neural Discrete Representation Learning. NeurIPS 2017.

VQ-VAE Formulation

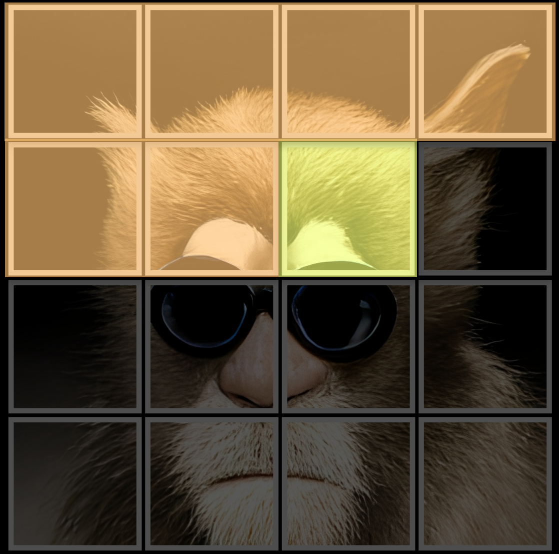
$$\mathcal{L} = \underbrace{\log p(\mathbf{x}|\mathbf{z}_d(\mathbf{e}))}_{\text{Reconstruction loss}} + \underbrace{\beta \|\mathbf{z}_e(\mathbf{x}) - \mathrm{sg}[\mathbf{e}]\|_2^2}_{\text{Commitment loss}}$$

$$+ \underbrace{\|\mathrm{sg}[\mathbf{z}_e(\mathbf{x})] - \mathbf{e}\|_2^2}_{\text{Codebook loss}}$$

Oord et al. Neural Discrete Representation Learning. NeurIPS 2017.

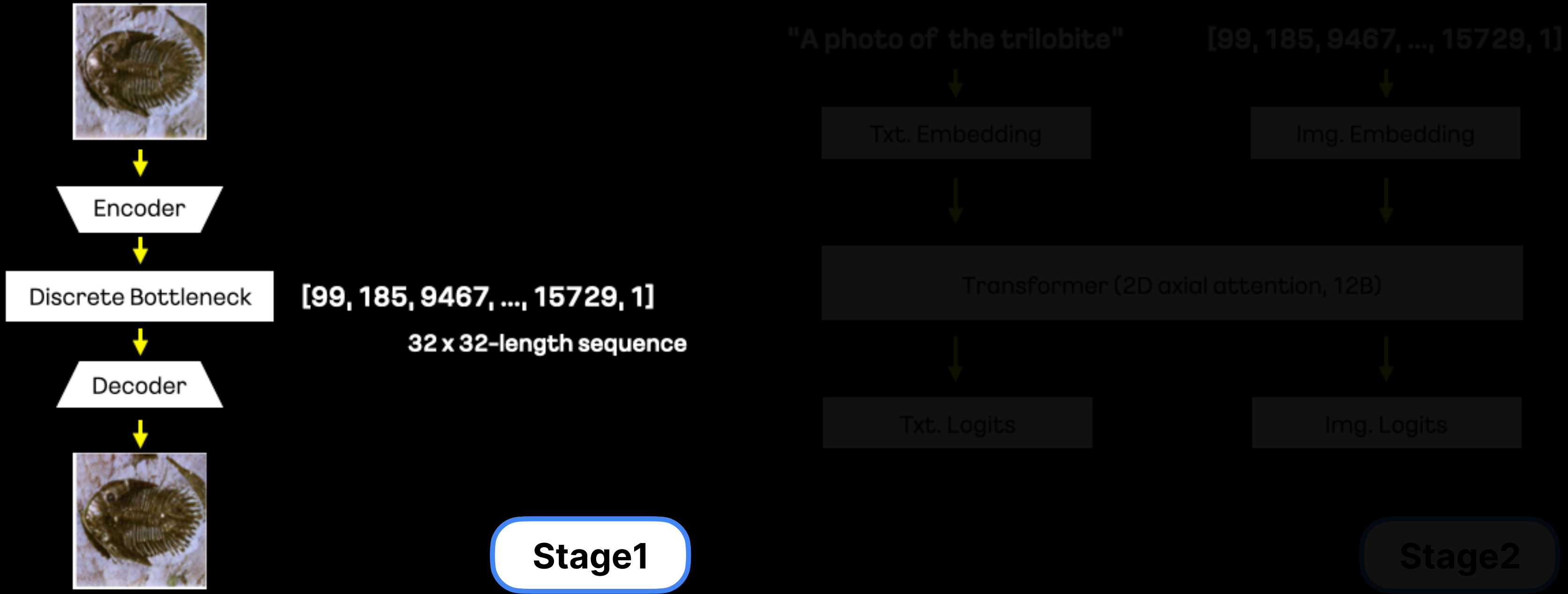
DALL-E: Text-to-Image AR Generation

"A painting of a monkey with sunglasses"



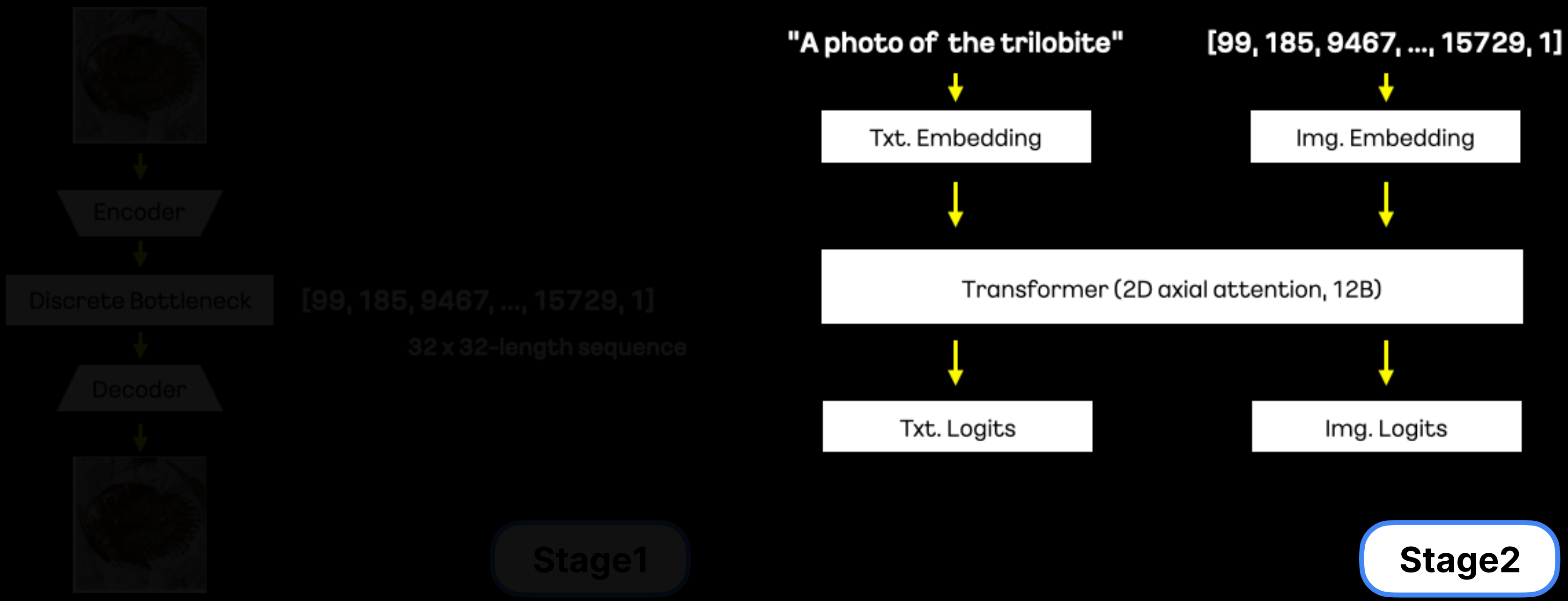
$$P(x_{txt}, x_1, x_1, \ldots, x_{16}) = \prod_m P(x_m \,|\, x_{<m}, x_{txt})$$

DALL-E (Model)



Ramesh et al. Zero-shot Text-to-Image Generation. ICML 2021

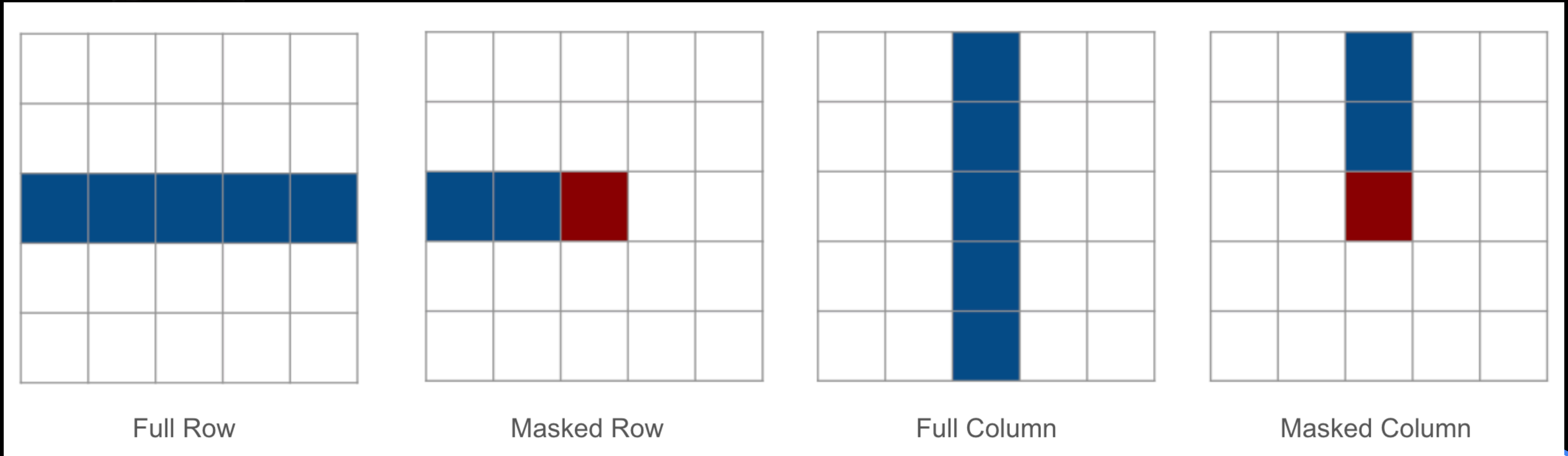
DALL-E (Model)



Ramesh et al. Zero-shot Text-to-Image Generation. ICML 2021

DALL-E (Model)

"A photo of the trilobite"   [99, 185, 9467, ..., 15729, 1]

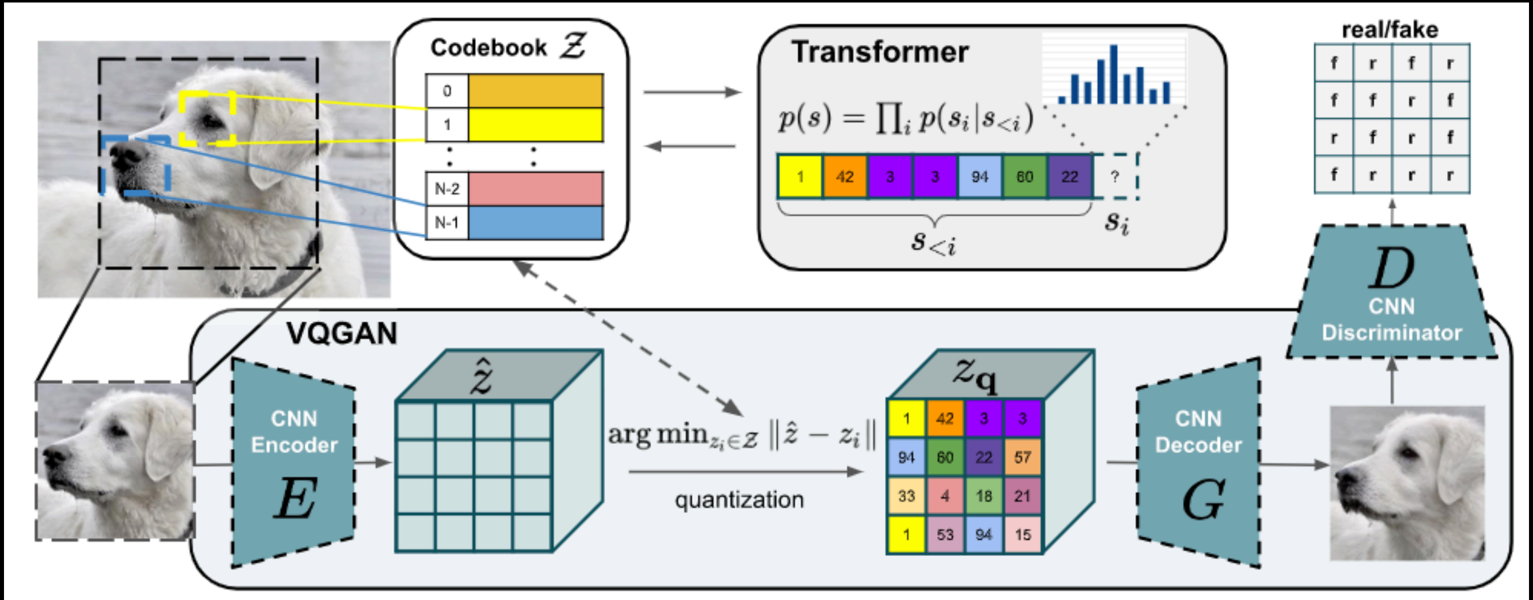


Stage1

Stage2

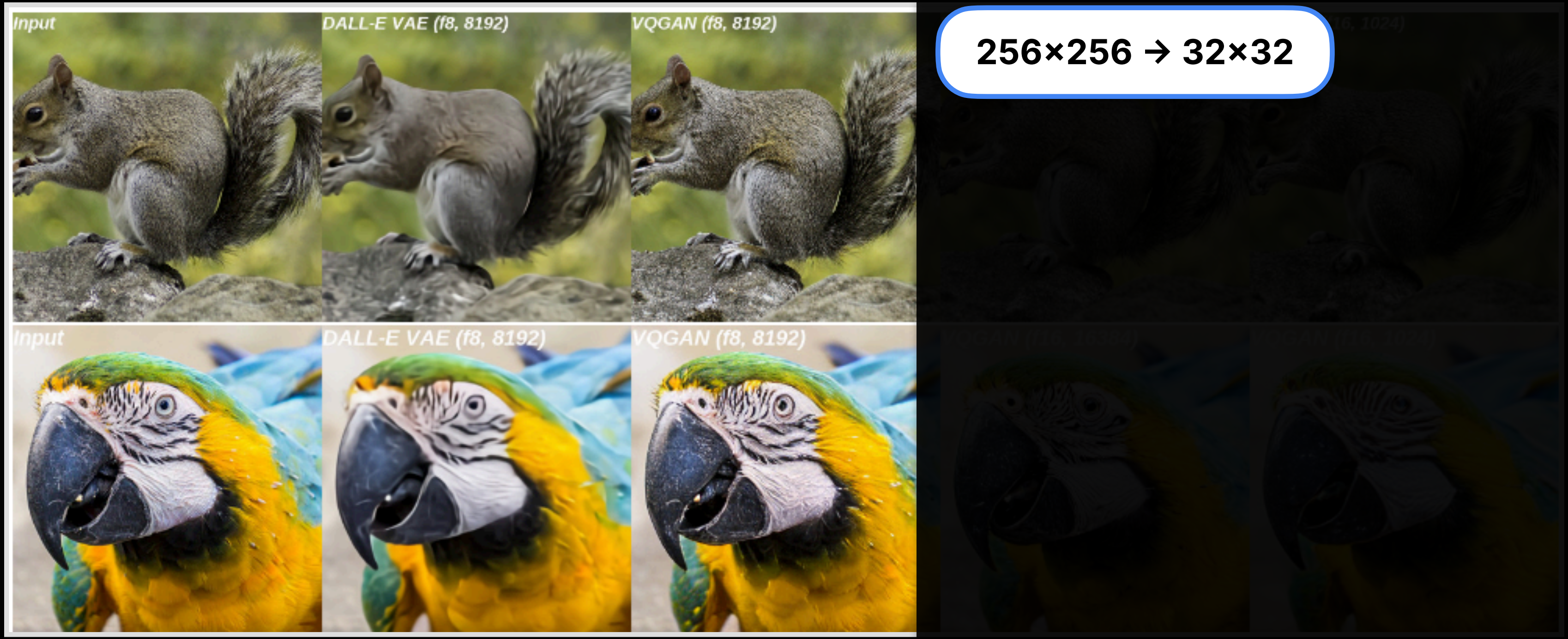
Ramesh et al. Zero-shot Text-to-Image Generation. ICML 2021

VQ-GAN



Esser et al. Taming Transformers for High-Resolution Image Synthesis. CVPR 2021.

VQ-GAN



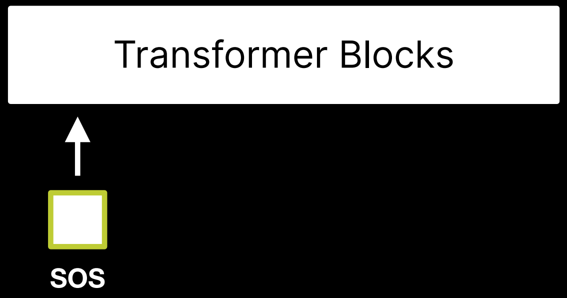
256×256 → 32×32

Esser et al. Taming Transformers for High-Resolution Image Synthesis. CVPR 2021.

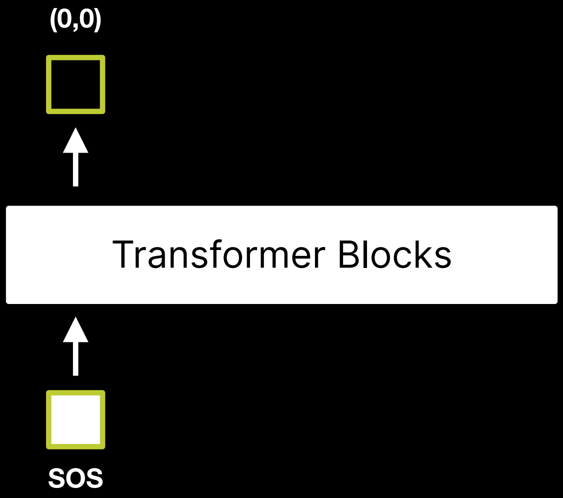
VQ-GAN

Esser et al. Taming Transformers for High-Resolution Image Synthesis. CVPR 2021.

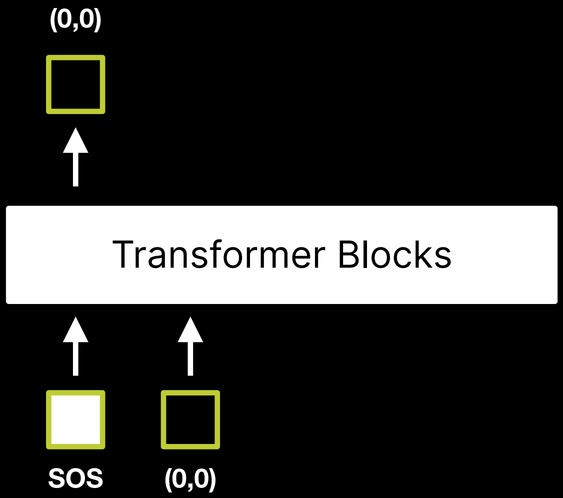
Naive Sampling



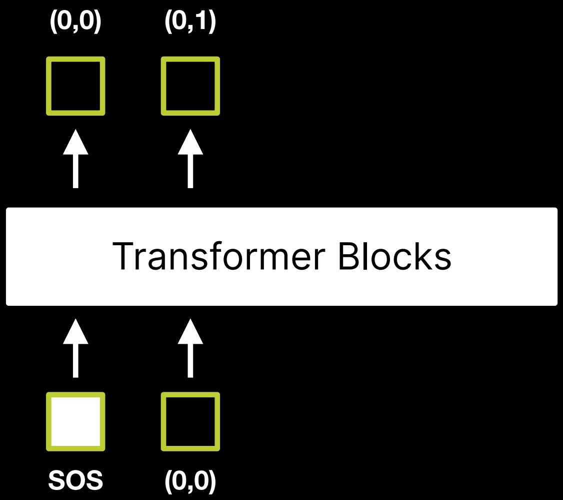
Naive Sampling



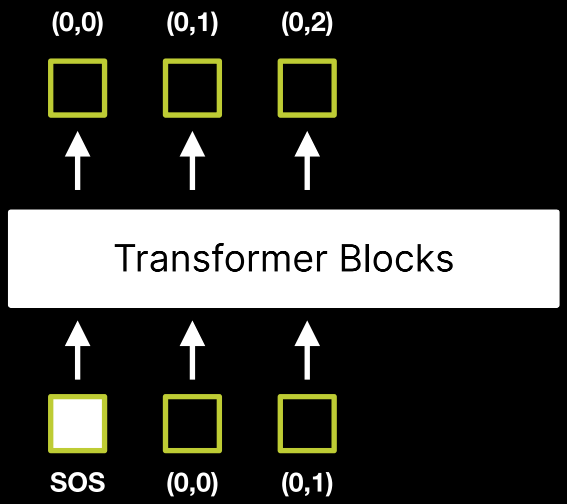
Naive Sampling



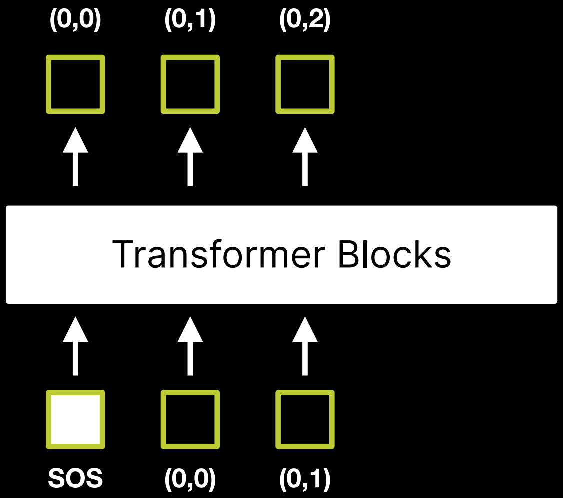
Naive Sampling



Naive Sampling

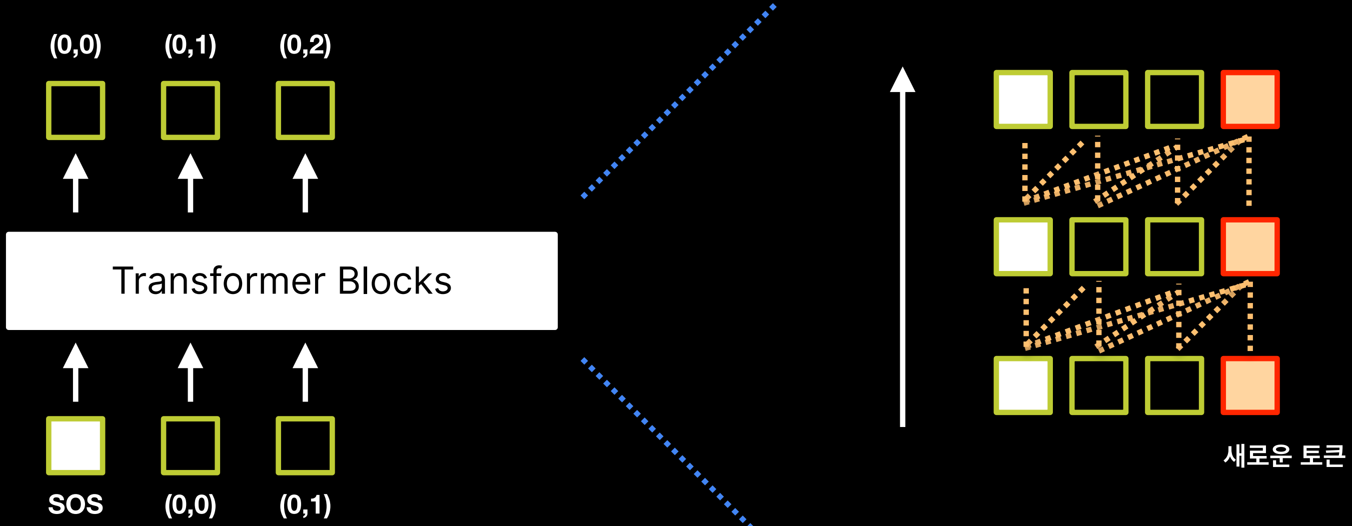


Naive Sampling

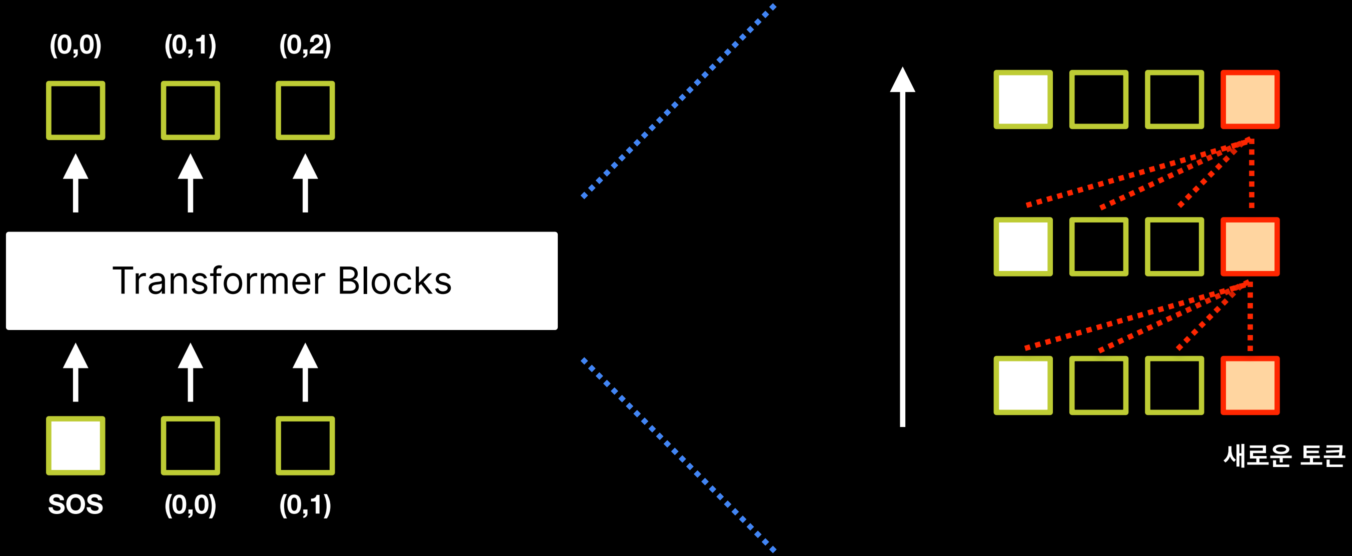


Need to re-compute hidden representations between previous selected tokens!

Fast Sampling - Caching



Fast Sampling - Caching



Advanced Topics

minDALL-E (publicly available)

1.3B text-to-image autoregressive generation model trained on 14M pairs

README.md

minDALL-E on Conceptual Captions

`minDALL-E`, naming after minGPT, is a 1.3B text-to-image generation model trained on 14 million image-text pairs for non-commercial purposes.

```
PyTorch == 1.8.0
CUDA >= 10.1
```

- Other packages

```
pip install -r requirements.txt
```

Model Checkpoint

- Model structure (two-stage autoregressive model)
  - Stage1: Unlike the original DALL-E [1], we replace Discrete VAE with VQGAN [2] to generate high-quality samples effectively. We slightly fine-tune vqgan_imagenet_f16_16384, provided by the official VQGAN repository, on FFHQ [3] as well as ImageNet.
  - Stage2: We train our 1.3B transformer from scratch on 14 million image-text pairs from CC3M [4] and CC12M [5]. For the more detailed model spec, please see configs/dalle-1.3B.yaml.

Sampling

- Given a text prompt, the code snippet below generates candidate images and re-ranks them using OpenAI's CLIP [6].
- This has been tested under a single V100 of 32GB memory. In the case of using GPUs with limited memory, please lower down num_candidates to avoid OOM.

```
from matplotlib import pyplot as plt
import clip
from dalle.models import Dalle
from dalle.utils.utils import set_seed, clip_score
```

```
images = model.sampling(prompt=prompt,
                        top_k=256, # It is recommended that top_k
                        top_p=None,
                        softmax_temperature=1.0,
                        num_candidates=96,
                        device=device).cpu().numpy()
images = np.transpose(images, (0, 2, 3, 1))

# CLIP Re-ranking
model_clip, preprocess_clip = clip.load("ViT-B/32", device=device)
model_clip.to(device=device)
rank = clip_score(prompt=prompt,
                  images=images,
                  model_clip=model_clip,
                  preprocess_clip=preprocess_clip,
                  device=device)

# Plot images
images = images[rank]
plt.imshow(images[0])
plt.show()
```

Quantitative Results

- We have validated `minDALL-E` on the CC3M validation set (in-distribution evaluation) and MS-COCO (zero-shot evaluation).
- For CC3M, we measure the cosine similarity between image and text representations from the pretrained CLIP model (ViT-B/32), referred to as CLIP-score.
- For MS-COCO, we compute FID between 30K generated and real samples from MS-COCO 2017, where we randomly choose 30K captions from COCO as in DALL-E. We select the best out of 32 candidates by CLIP re-ranking.

DALL-E [1]	-	27.5
minDALL-E	**0.26**	**14.7**

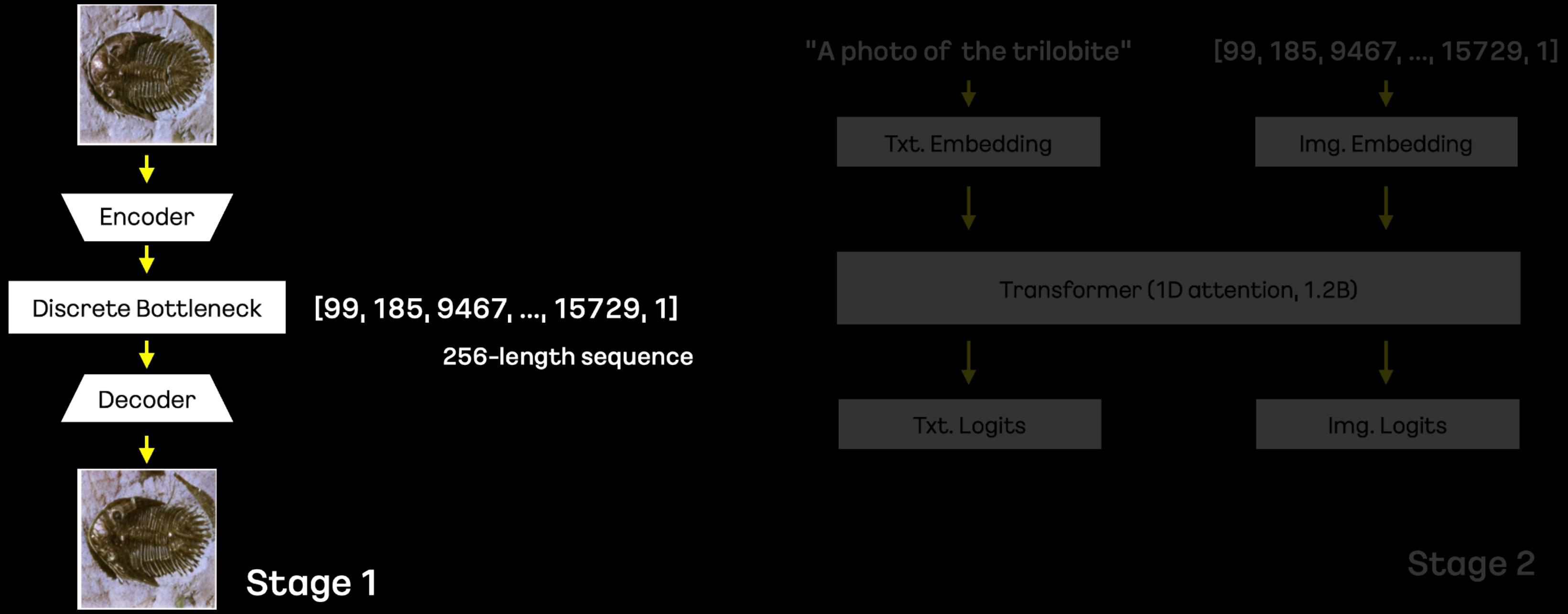
Transfer Learning Examples

- `minDALL-E`, which is pre-trained on noisy text supervisions, could be transferable to class-conditional and unconditional generation tasks. To validate this, we simply fine-tune it on ImageNet over 8 epochs in the case of class-conditional generation and unconditional generation.
- The commands below fine-tune the pretrained DALL-E. It takes about 36 hours on 8 V100 GPUs.

```
# unconditinoal image generation for imagenet (256x256)
python examples/transfer_learning_ex.py -d=configs/transfer-imagen
                                        -u=[MODEL_CKPT]
                                        -r=[RESULT_PATH]
                                        --n-gpus=[NUM_GPUS]
```

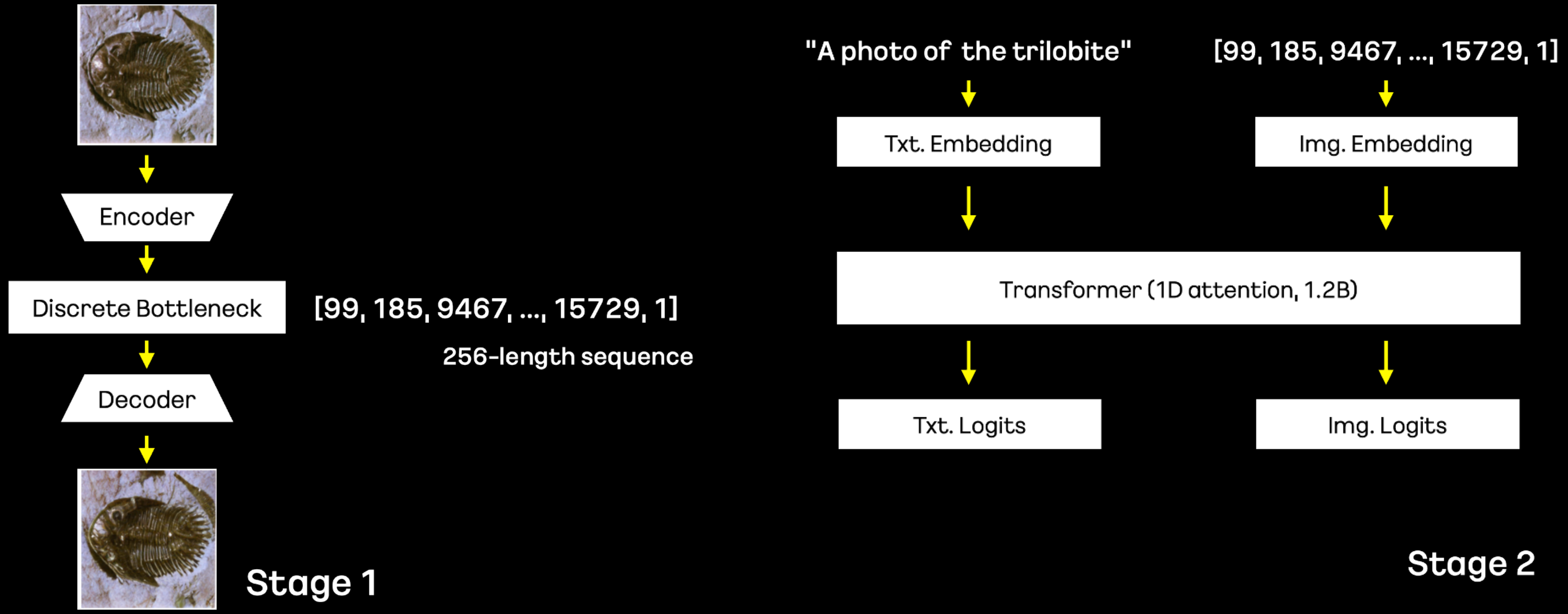
https://github.com/kakaobrain/minDALL-E

minDALL-E = VQGAN + Transformer 1D



Esser et al. Taming Transformers for High-Resolution Image Synthesis. CVPR 2021.

minDALL-E = VQGAN + Transformer 1D

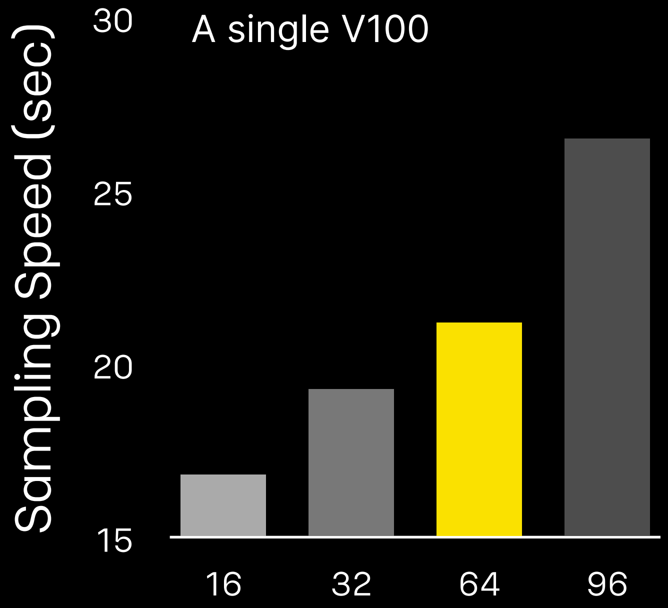


Esser et al. Taming Transformers for High-Resolution Image Synthesis. CVPR 2021.

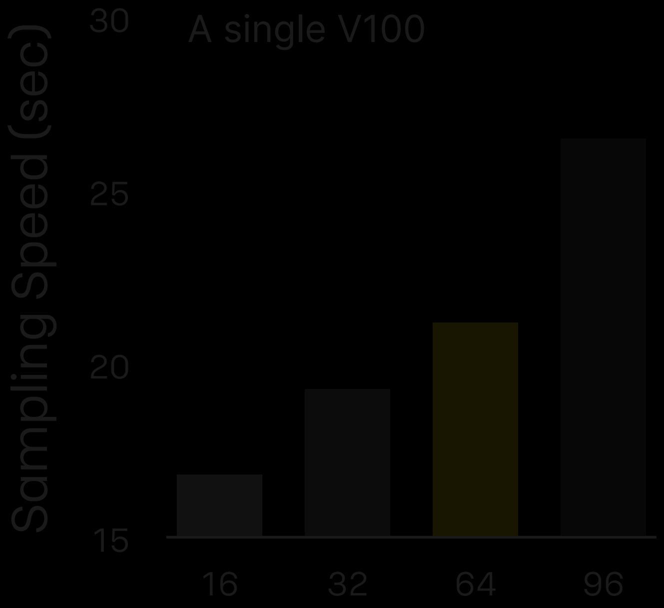
Quantitative Results

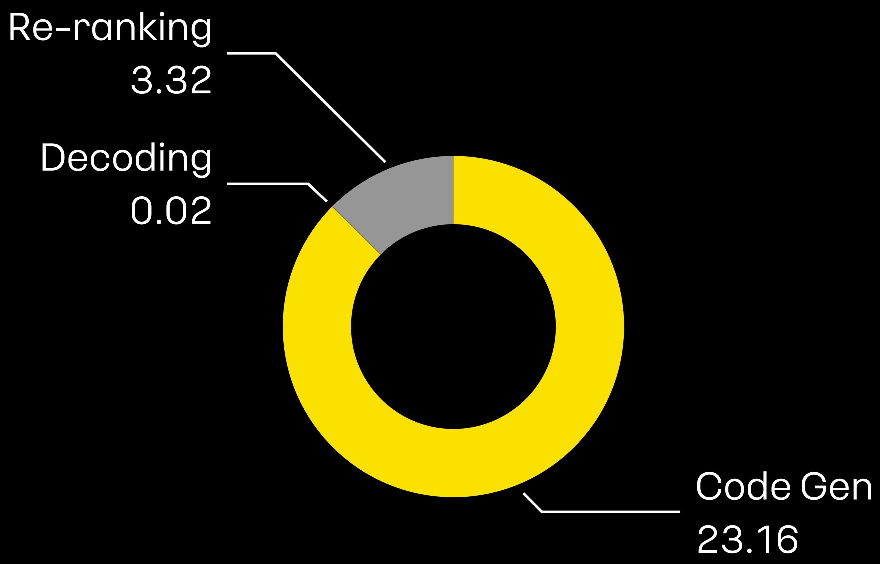
Model	CC3M Validation	COCO Validation	
	CLIP Score	FID-30K	FID-30K (re-ranking)
VQ-GAN	0.20	-	-
ImageBART	0.23	-	-
DALL-E	-	34.5	27.5
minDALL-E	0.26	19.6	14.7

Sampling Time



Sampling Time





Characteristic w.r.t. Hyper-parameters

A painting of a cherry blossom tree

Top-K = 256, Temp=0.5

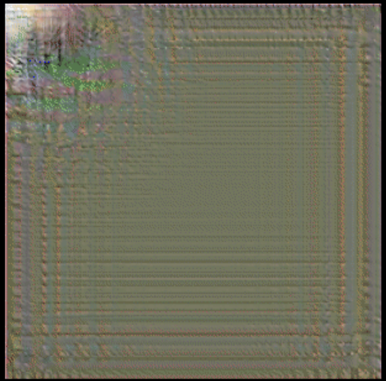
Top-K = 256, Temp=1.0

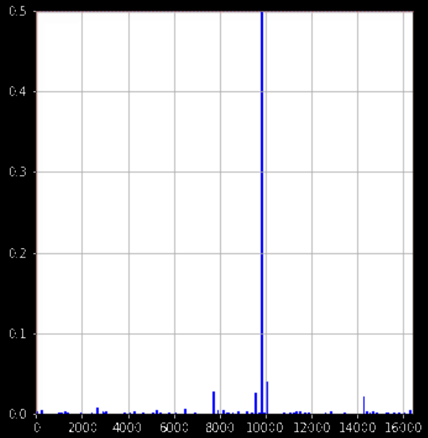
Top-K = 256, Temp=5.0

Characteristic w.r.t. Hyper-parameters

A painting of a cherry blossom tree

Top-K = 256, Temp=0.5

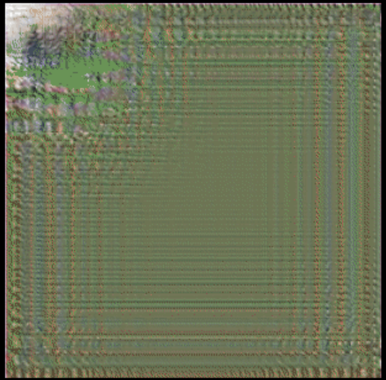


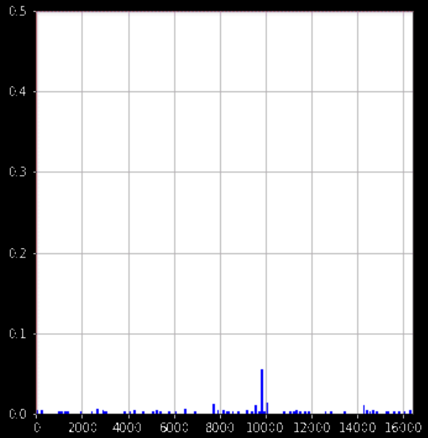


Characteristic w.r.t. Hyper-parameters

A painting of a cherry blossom tree

Top-K = 256, Temp=1.0

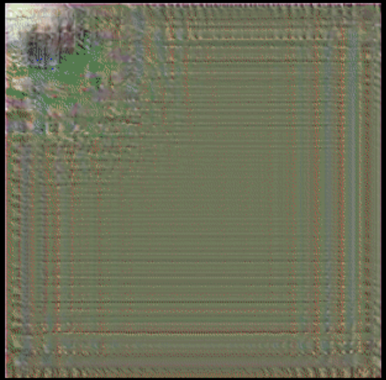


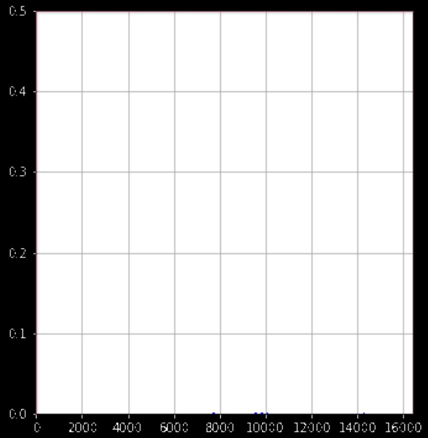


Characteristic w.r.t. Hyper-parameters

A painting of a cherry blossom tree

Top-K = 256, Temp=5.0





Our Research

Sampling/
Training
Speed-up

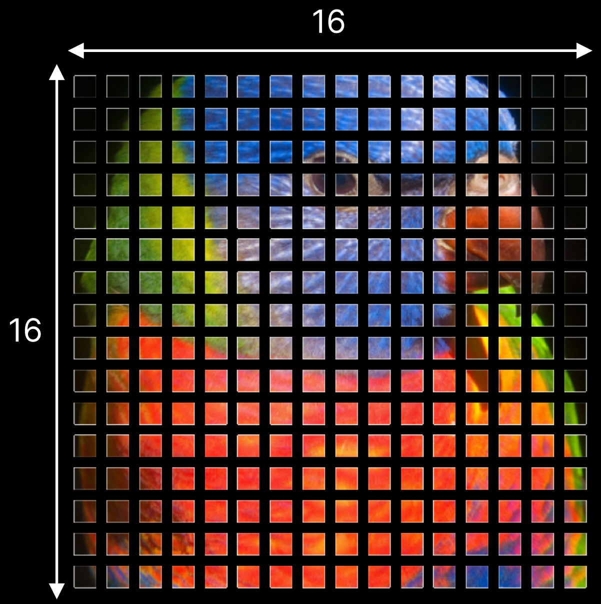
Fine-grained
Sampling

Our Research

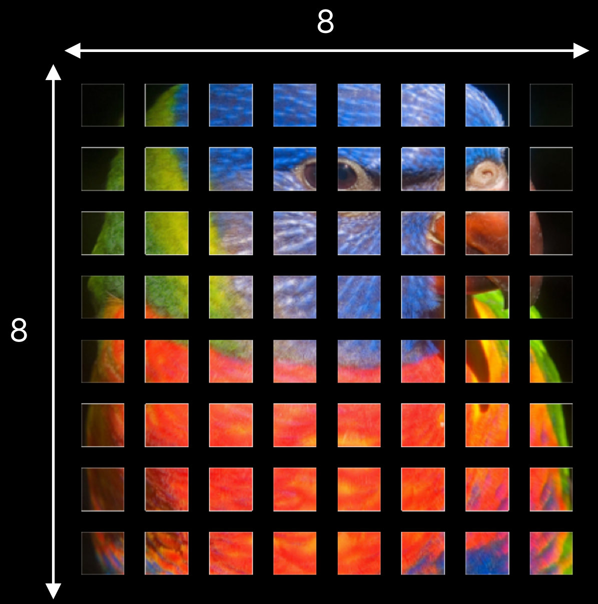
Sampling/
Training
Speed-up

Fine-grained
Sampling

Why training/sampling slow?

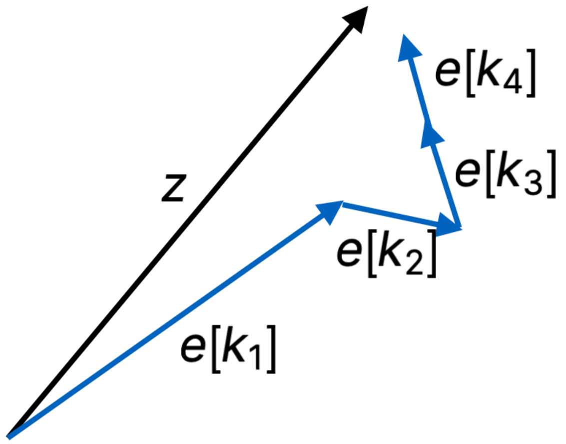


Why training/sampling slow?



Residual-Quantized VAE (RQ-VAE)

Coarse-to-fine reconstruction by residual quantization



$$\mathcal{RQ} : z \mapsto (k_1, k_2, k_3, k_4)$$

$$z \approx e[k_1] + e[k_2] + e[k_3] + e[k_4]$$

Lee et al. Autoregressive Image Generation using Residual Quantization, CVPR'22.

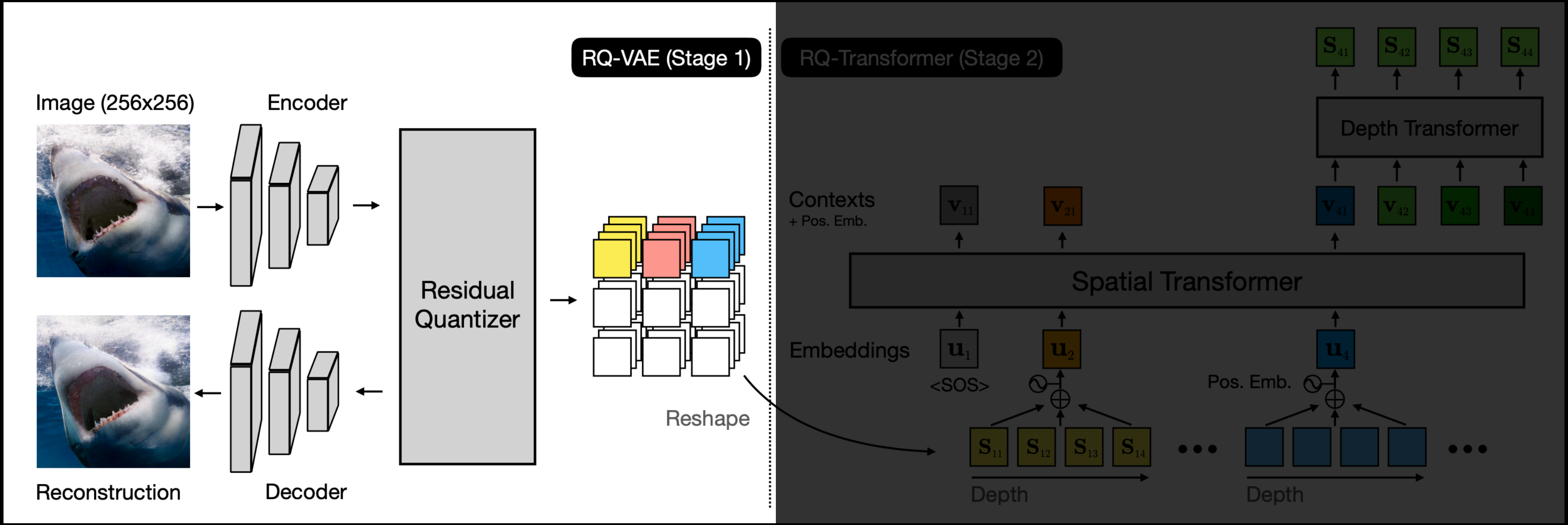
Residual-Quantized VAE (RQ-VAE)

Coarse-to-fine reconstruction by residual quantization

Original Depth 0 Depth 1 Depth 2 Depth 3

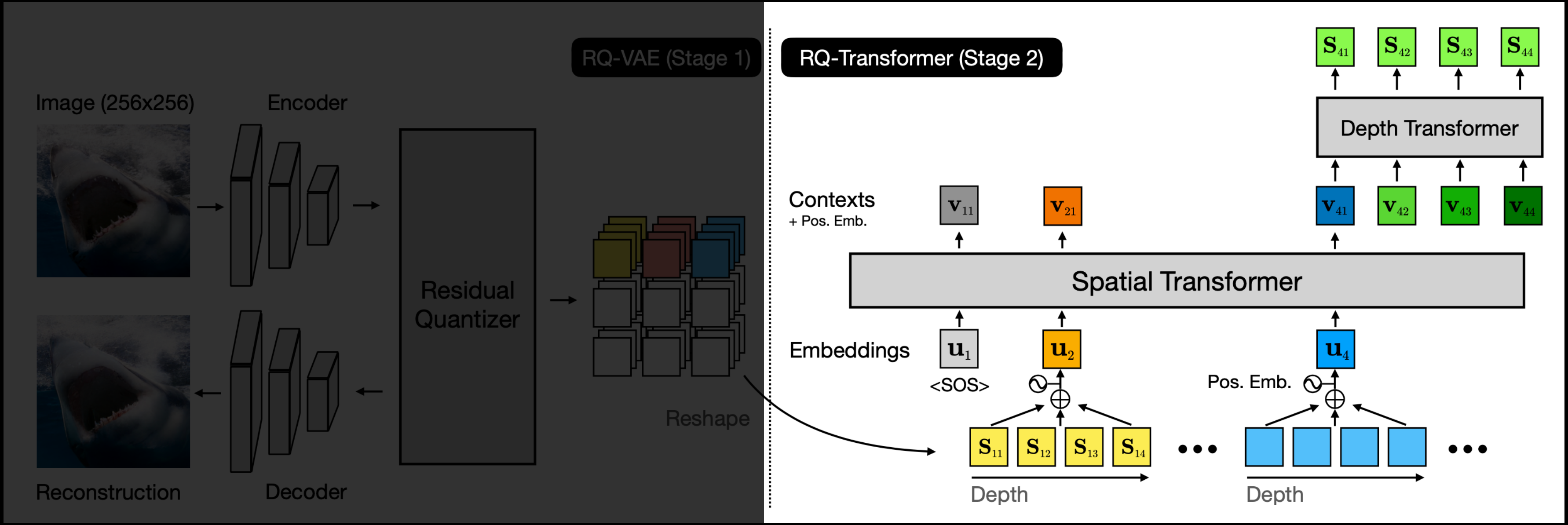
Lee et al. Autoregressive Image Generation using Residual Quantization, CVPR’22.

RQ-VAE & RQ-Transformer



Lee et al. Autoregressive Image Generation using Residual Quantization, CVPR’22.

RQ-VAE & RQ-Transformer



Lee et al. Autoregressive Image Generation using Residual Quantization, CVPR'22.

RQ-Transformer

RQ-Transformer is more efficient than previous AR models in Text-to-Image / class-cond. Image generation task, while performs better than ones

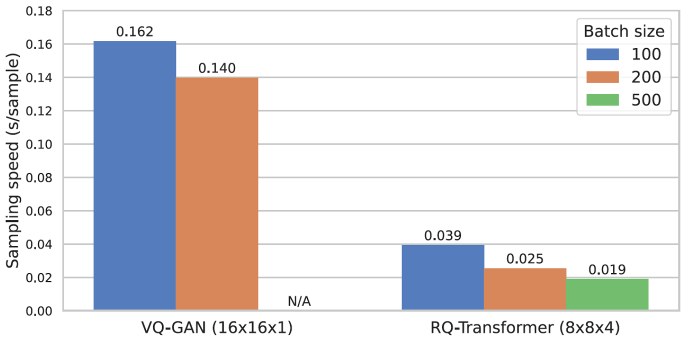


Figure 4. The sampling speed of RQ-Transformer with 1.4B parameters according to batch size and code map shape.

Lee et al. Autoregressive Image Generation using Residual Quantization, CVPR'22.

RQ-Transformer

RQ-Transformer is more efficient than previous AR models in Text-to-Image / class-cond. Image generation task, while performs better than ones

Table 2. Comparison of FIDs and ISs for class-conditioned image generation on ImageNet [9] 256×256. † denotes a model without our stochastic sampling and soft labeling. ‡ denotes the use of rejection sampling with 0.05 acceptance rate.

	Params	FID	IS
ADM [11]	554M	4.59	186.7
ImageBART [13]	3.5B	21.19	61.6
BigGAN [3]	164M	7.53	168.6
BigGAN-deep [3]	112M	6.84	203.6
VQ-VAE2 [39]	13.5B	∼31	∼45
DCT [33]	738M	36.5	n/a
VQ-GAN [14]	1.4B	15.78	74.3
RQ-Transformer†	821M	14.06	95.8±2.1
RQ-Transformer	821M	13.11	104.3±1.5
RQ-Transformer	1.4B	11.56	112.4±1.1
RQ-Transformer‡	1.4B	4.45	326.0±3.5
Validation Data	-	1.62	234.0

Table 3. Comparison of FID and CLIP score [36] on the validation data of CC-3M [43] for text-conditioned image generation.

	Params	FID	CLIP-s
VQ-GAN [14]	600M	28.86	0.20
ImageBART [13]	2.8B	22.61	0.23
RQ-Transformer	654M	12.33	0.26

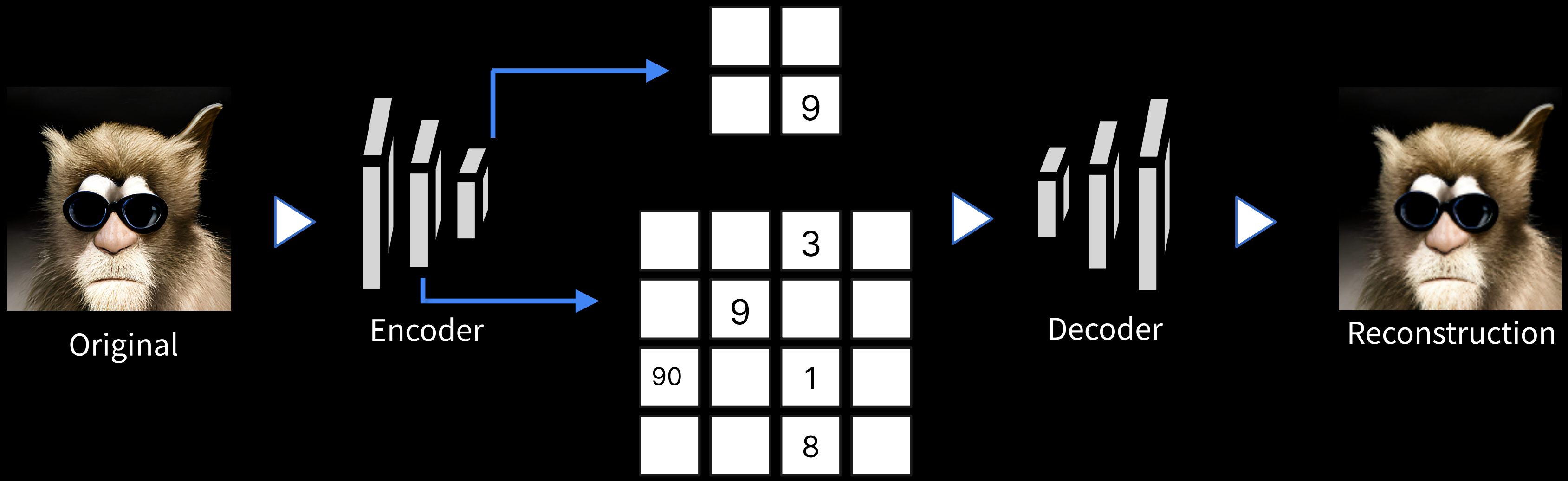
Lee et al. Autoregressive Image Generation using Residual Quantization, CVPR'22.

Our Research

Sampling/
Training
Speed-up

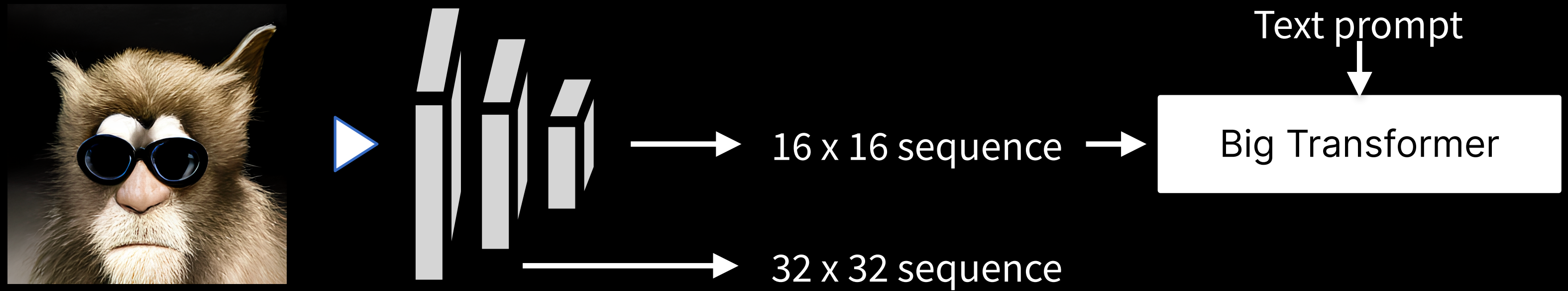
Fine-grained
Sampling

Multi-scale VQ for Enhancement

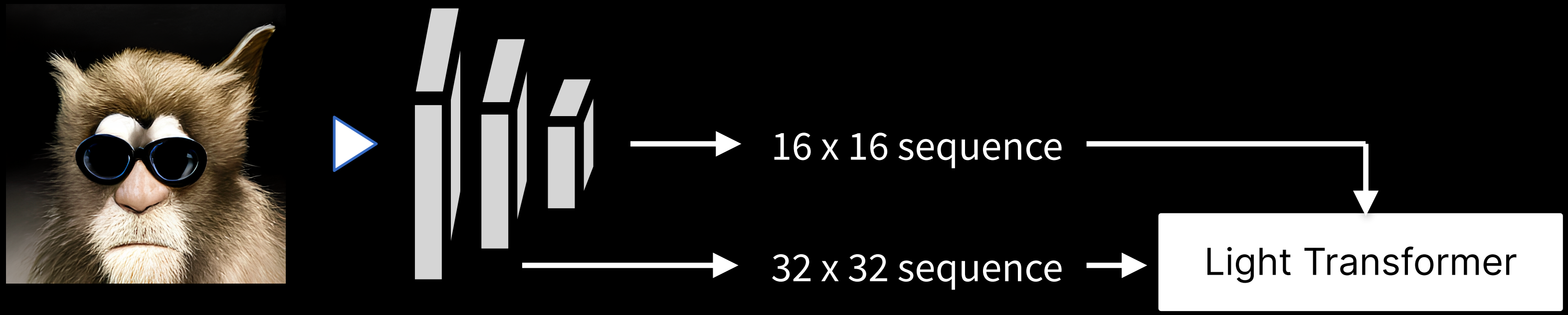


Razavi et al. Generating Diverse High-Fidelity Images with VQ-VAE-2. NeurIPS 2019.

Multi-scale VQ for Enhancement



Multi-scale VQ for Enhancement



Multi-scale VQ for Enhancement

A cartoon character of a pineapple

A painting of a monkey with sunglasses in the frame

minDALL-E

minDALL-E +
multi-scale VQ

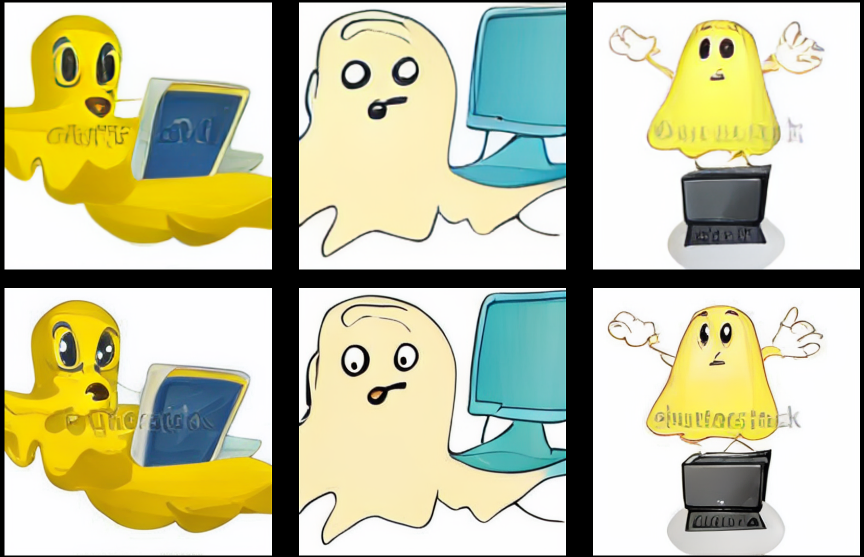
Multi-scale VQ for Enhancement

Café Terrace at Night

An illustration of a yellow ghost with a computer

minDALL-E

minDALL-E + multi-scale VQ



Conclusion

Autoregressive Models / Ours Approaches