MaskGIT: Masked Generative Image Transformer

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, William T. Freeman
Google

Presenters: Andrea Dunn Beltran, Rodrigo Meza

MaskGIT

Image Synthesis and Manipulation Tasks

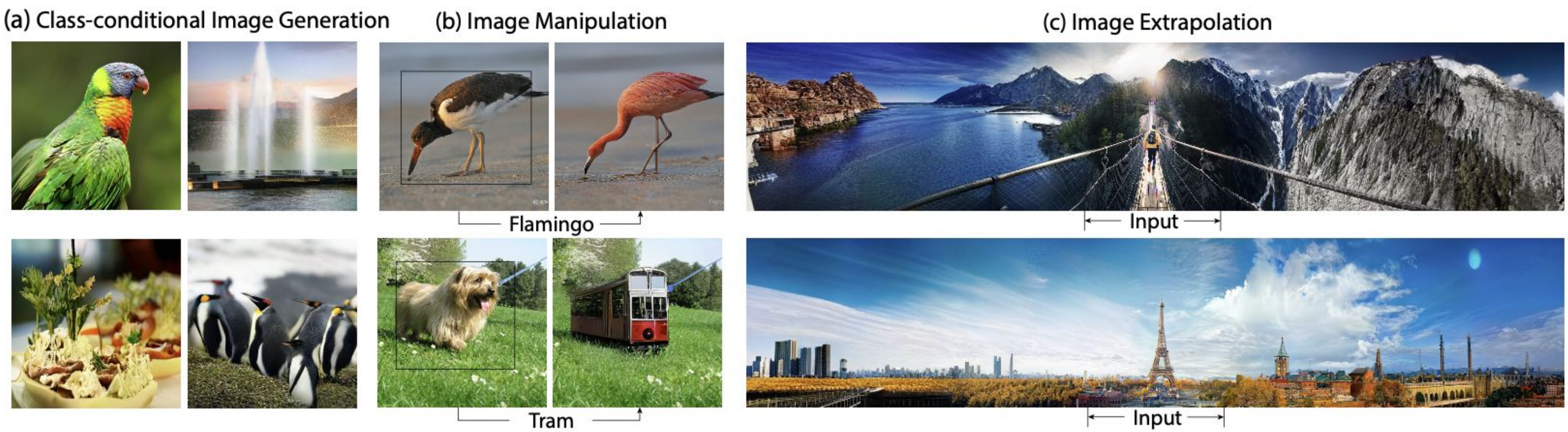


Image Synthesis

Impainting

Input ——— MaskGIT (Our Samples) ———

Input ——— MaskGIT's Samples ———

https://masked-generative-image-transformer.github.io/

Image Synthesis

Outpainting

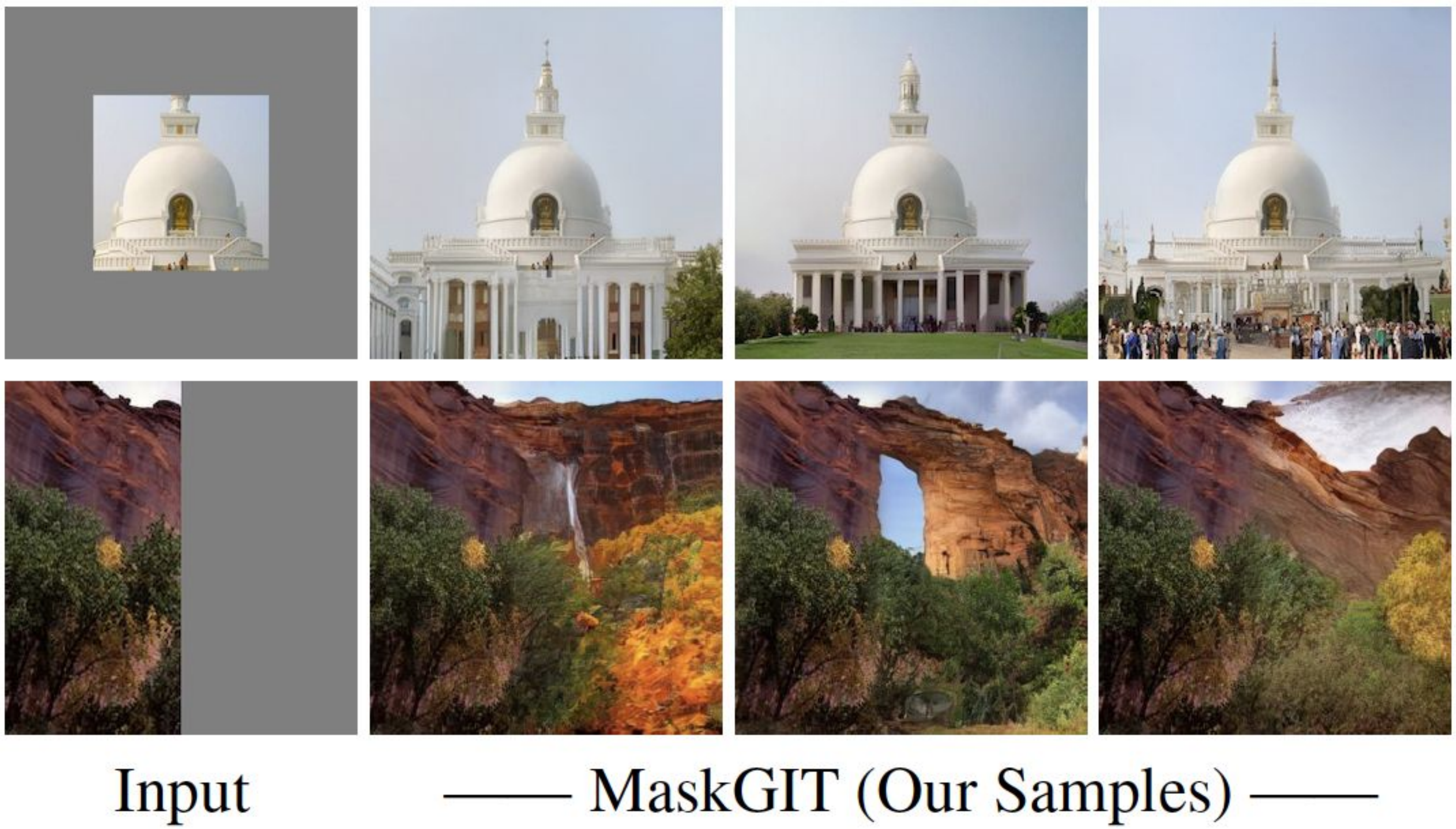


Image Synthesis

Horizontal Image Extrapolation

Image Synthesis

Class-conditional Image editing

Class-conditional Image Editing by MaskGIT

Image Synthesis

Class-conditional Image editing

Input Image

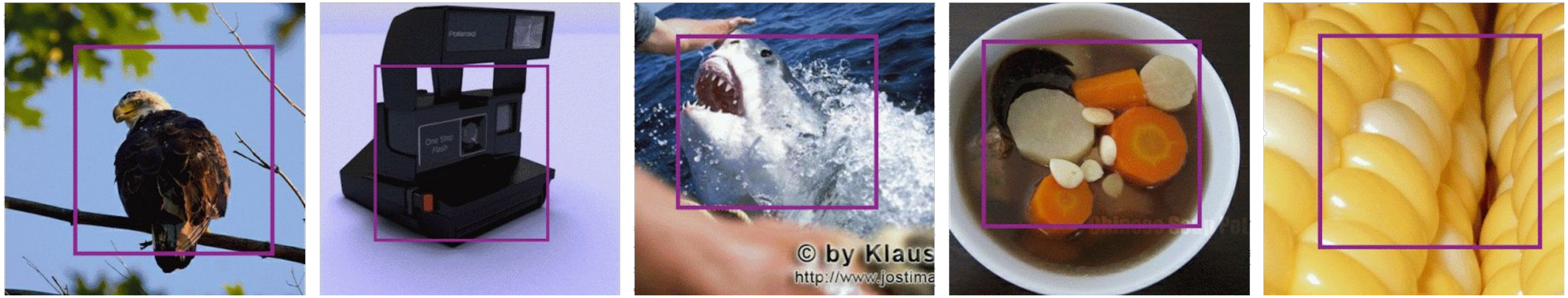


Image Synthesis with Transformers

Two Stage Image Generation

- **Stage 1:** Tokenization, image quantization to sequence of discrete tokens
- **Stage 2:** Autoregressive model is learned to generate image tokens sequentially
- **Issues with raster scan:** Images are NOT sequential

Scheduled Parallel Decoding

Comparison against Sequential Decoding

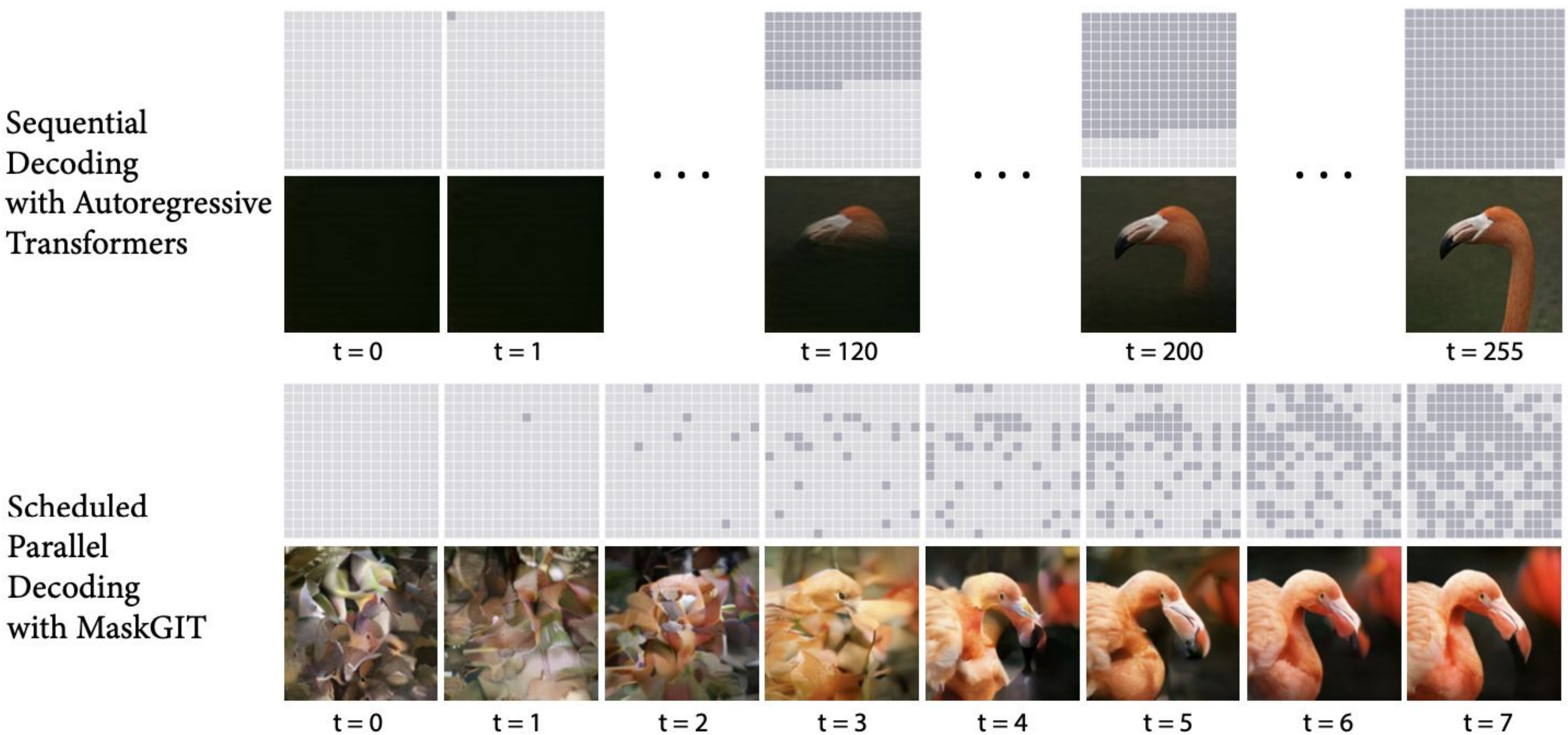


Image Synthesis

Related Work

- **Generative Adversarial Networks** (GANs)
- **Variational Auto-encoders** (VAEs)

- Transformer Based Image Synthesis
  - **VQVAE** introduces vector quantization to the **VAE** method using a 2 stage approach
  - **VQGAN** improves on **VQVAE** and combines vector quantization with adversarial and perceptual loss

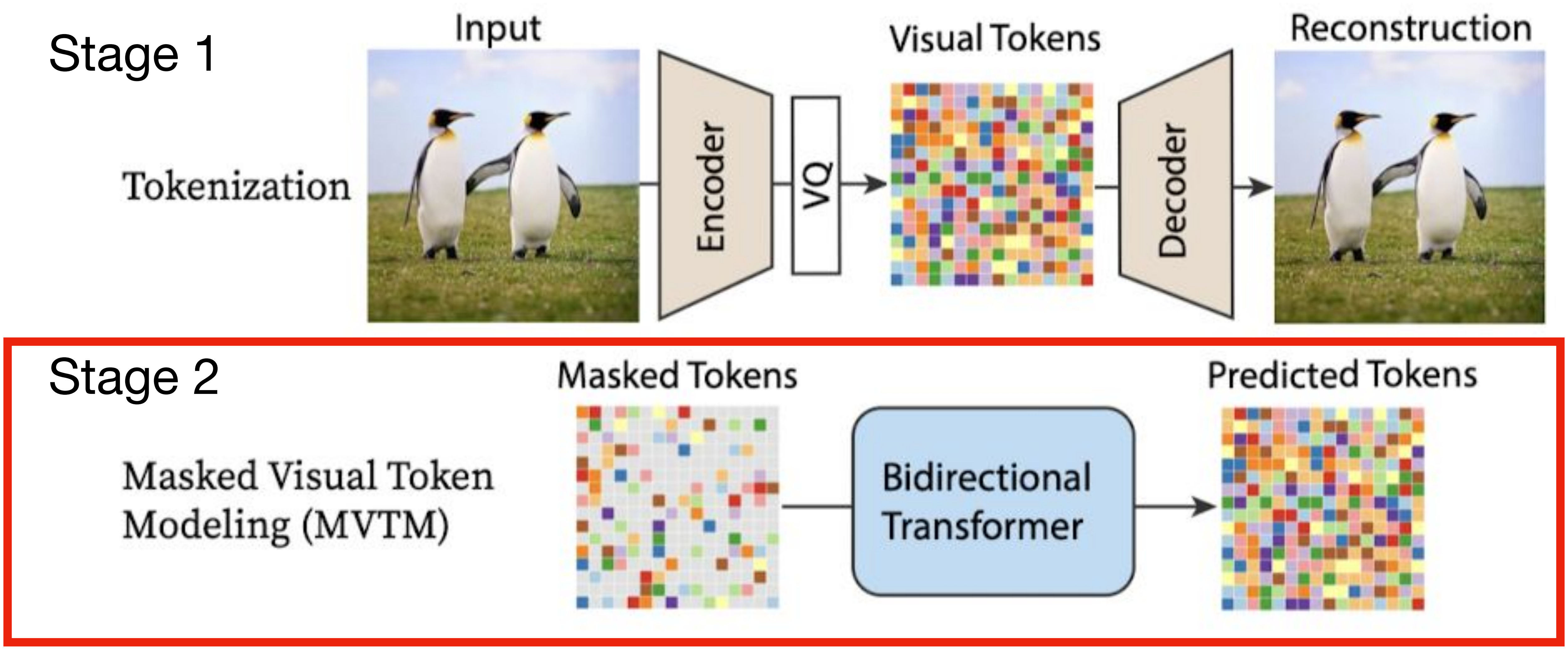
Masked Modeling with Bi-directional Transformers

Related Work

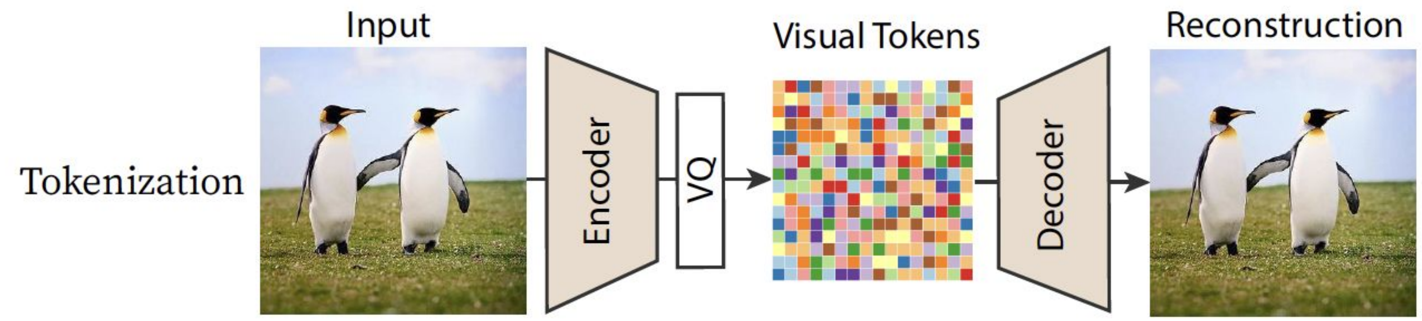
- **Masked Language Modeling** (MLM)
  - Introduced by BERT
  - Allows the masked tokens to be predicted using context from both directions
- Difficult to perform autoregressive decoding using bi-directional attentions

Pipeline Overview

Method

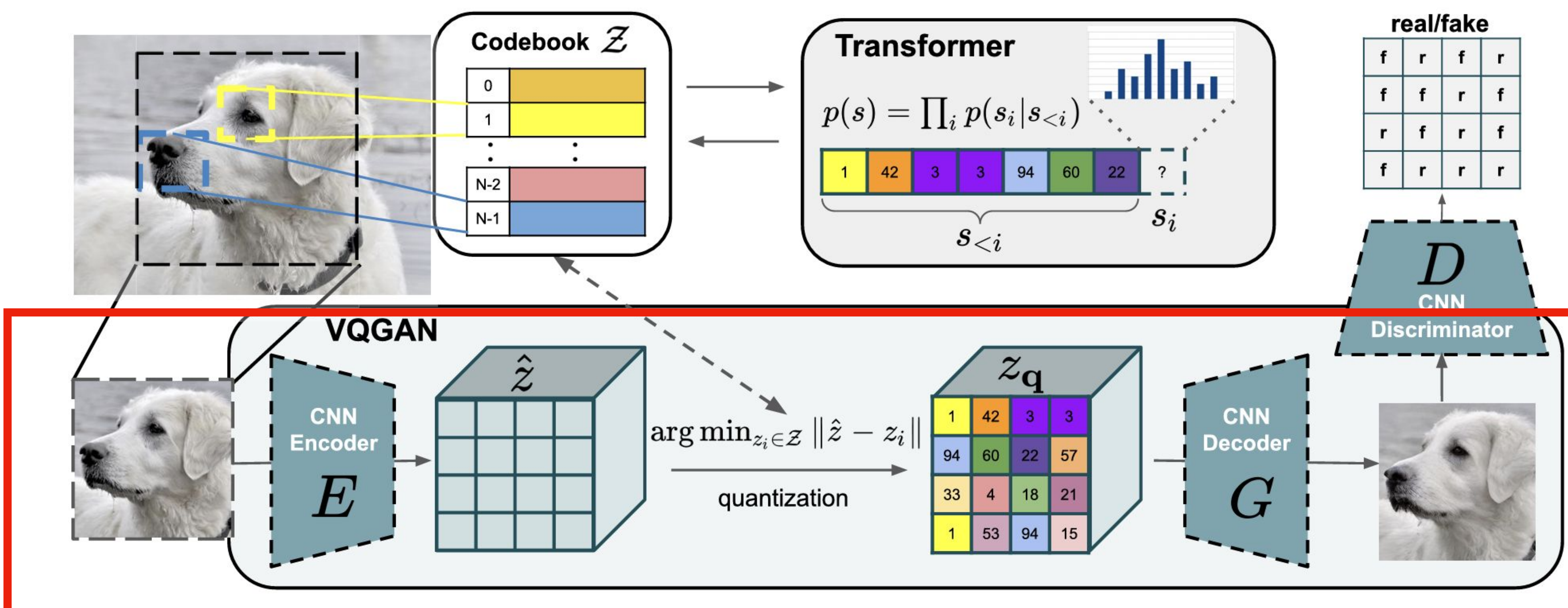


Tokenization



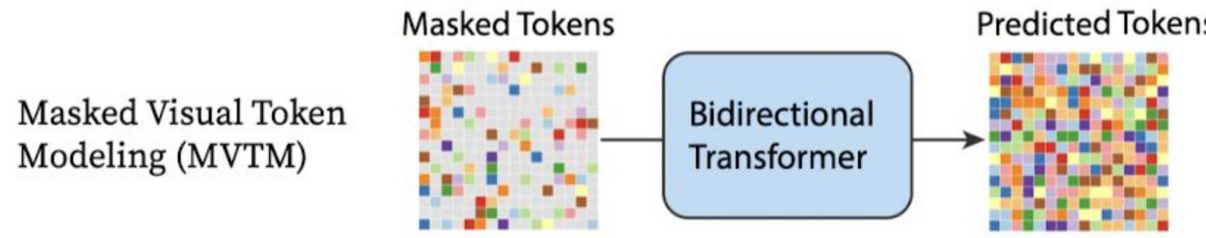
Method: Stage 1

- The paper uses the tokenization method introduced in the VQGAN method.
- This allows them to solely focus on improving Stage 2.



MVTM in Training

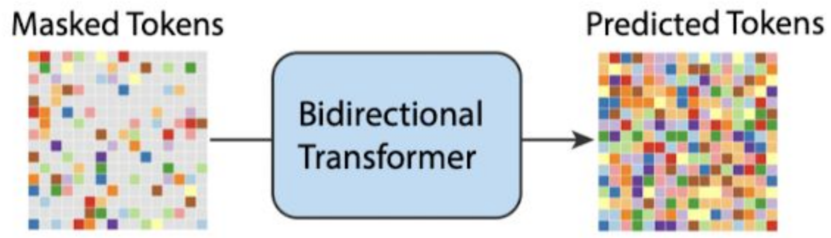
Method: Stage 2



- $\mathbf{Y} = [y_i]_{i=1}^{N}$: latent tokens obtained by inputting the image to the VQ-encoder
- N : length of the reshaped token matrix
- $\mathbf{M} = [m_i]_{i=1}^{N}$: corresponding binary mask.
- y_i is replaced with a special $\texttt{[MASK]}$ token if $m_i = 1$, otherwise left intact.

MVTM in Training

Masked Visual Token Modeling (MVTM)



Method: Stage 2

- $\gamma(r) \in (0, 1]$: Mask scheduling function

  1. Sample a ratio from 0 to 1

  2. Uniformly select $\lceil \gamma(r) \cdot N \rceil$ tokens in $\mathbf{Y}$ to place masks.

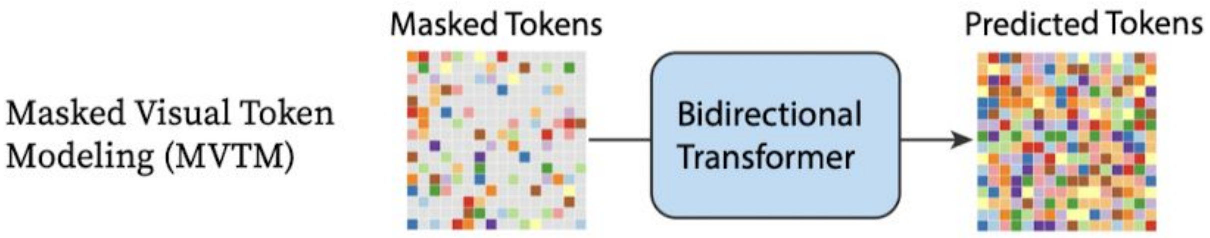
- $Y_{\overline{\mathbf{M}}}$: the result after applying mask $\mathbf{M}$ to $\mathbf{Y}$.

- Objective is to minimize the negative log likelihood of the masked tokens:

$$\mathcal{L}_{\text{mask}} = - \mathop{\mathbb{E}}_{\mathbf{Y} \in \mathcal{D}} \left[\sum_{\forall i \in [1,N], m_i = 1} \log p(y_i | Y_{\overline{\mathbf{M}}}) \right]$$

MVTM in Training

Method: Stage 2



$$\mathcal{L}_{\text{mask}} = - \mathop{\mathbb{E}}_{\mathbf{Y} \in \mathcal{D}} \Big[\sum_{\forall i \in [1,N], m_i = 1} \log p(y_i | Y_{\overline{\mathbf{M}}}) \Big]$$

- We feed the masked $Y_{\overline{\mathbf{M}}}$ into multi-layer bidirectional transformer to predict the probabilities $P(y_i | Y_{\overline{\mathbf{M}}})$ for each masked token.
- Negative log-likelihood is computed as the cross-entropy between the ground truth one-hot token and predicted token..

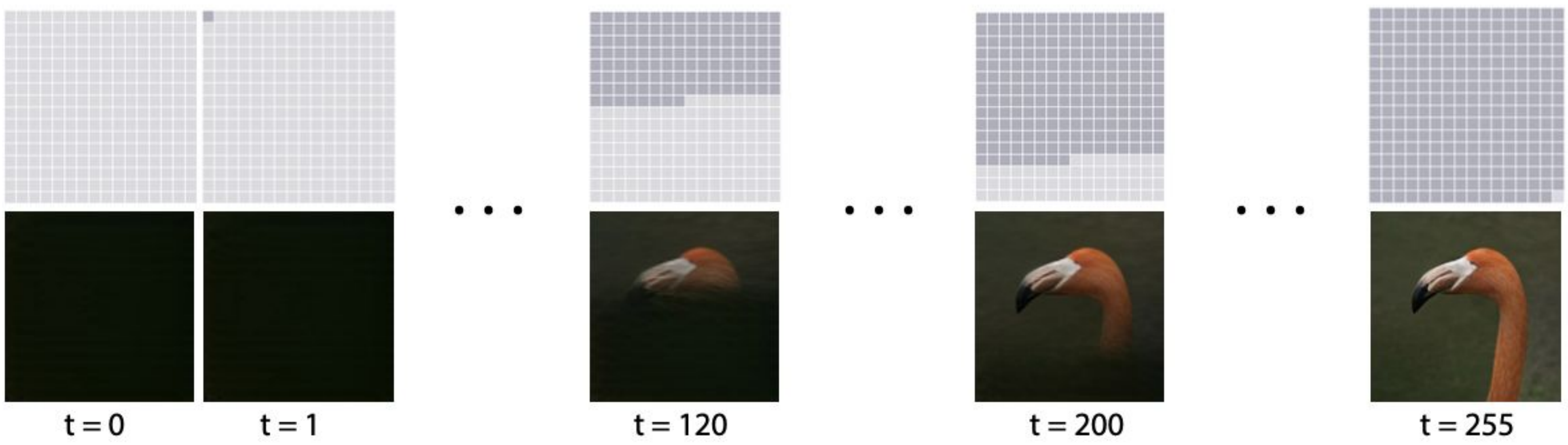
Main difference to autoregressive modeling is the conditional dependency in MVTM has two directions, which allows it to get richer tokens by attending all tokens in the image.

Iterative Decoding

Autoregressive Decoding

- Tokens are generated sequentially based on previous output.
- Not parallelizable and very slow for images due to image token length.

Sequential Decoding with Autoregressive Transformers



Iterative Decoding

Method

- Novel method where all image tokens are created simultaneously in parallel.
  - Feasible due to the bi-directional self-attention of MTVM.

For iteration *t,* we do:

1. **Predict:** Given the tokens $Y_{\mathrm{M}}^{(t)}$, we predict the probabilities, $p^{(t)} \in \mathbb{R}^{N \times K}$, for all masked locations in parallel.

2. **Sample:** At each masked location *i,* we sample the tokens and predict the probability to use as a confidence score. Leave unmasked tokens with a confidence score of 1.0

3. **Mask Schedule**

4. **Mask**

Iterative Decoding

Method

1. **Predict**

2. **Sample**

3. **Mask Schedule:** Compute the number of tokens to mask according to the mask scheduling function γ by $n = \lceil \gamma(\frac{t}{T}) N \rceil$, where *N* is the input length and *T* the total number of iterations.

4. **Mask:** We obtain $Y_{\mathrm{M}}^{(t+1)}$ by masking *n* tokens in $Y_{\mathrm{M}}^{(t)}$. The mask for iteration *(t+1)* is:

$$m_i^{(t+1)} = \begin{cases} 1, & \text{if } c_i < \text{sorted}_j(c_j)[n]. \\ 0, & \text{otherwise.} \end{cases},$$

where c_i is the confidence score for the *i*-th token.

Iterative Decoding

Method

- The decoding algorithm synthesizes an image in T steps.
- At each iteration, it predicts all tokens simultaneously, keeping only the most confident ones.
- Remaining tokens are masked out and re-predicted in the next iteration.
- Mask ratio decreases until all tokens are generated within T iterations.

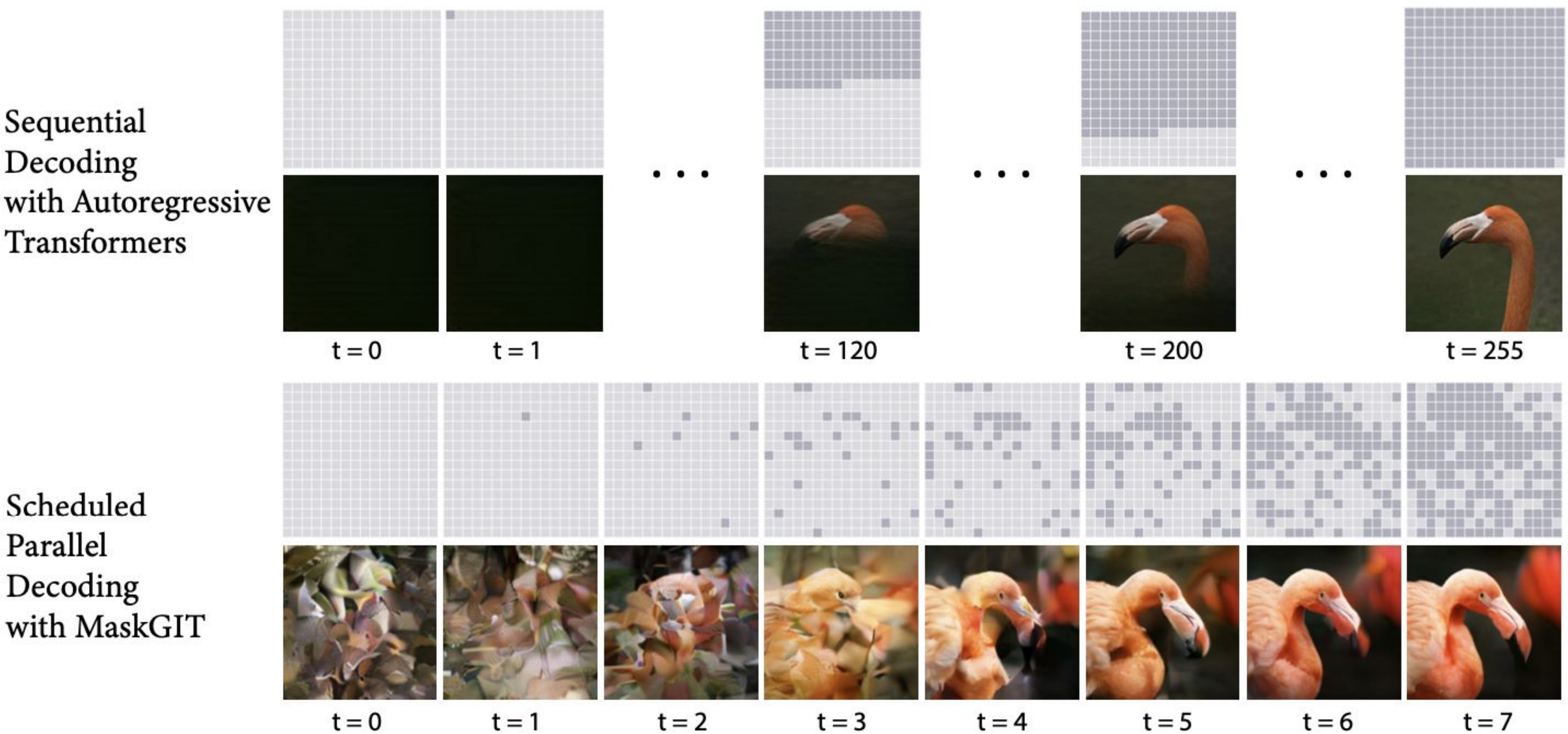
Iterative Decoding

Method

- The decoding algorithm synthesizes an image in T steps.
- At each iteration, it predicts all tokens simultaneously, keeping only the most confident ones.
- Remaining tokens are masked out and re-predicted in the next iteration.
- Mask ratio decreases until all tokens are generated within T iterations.

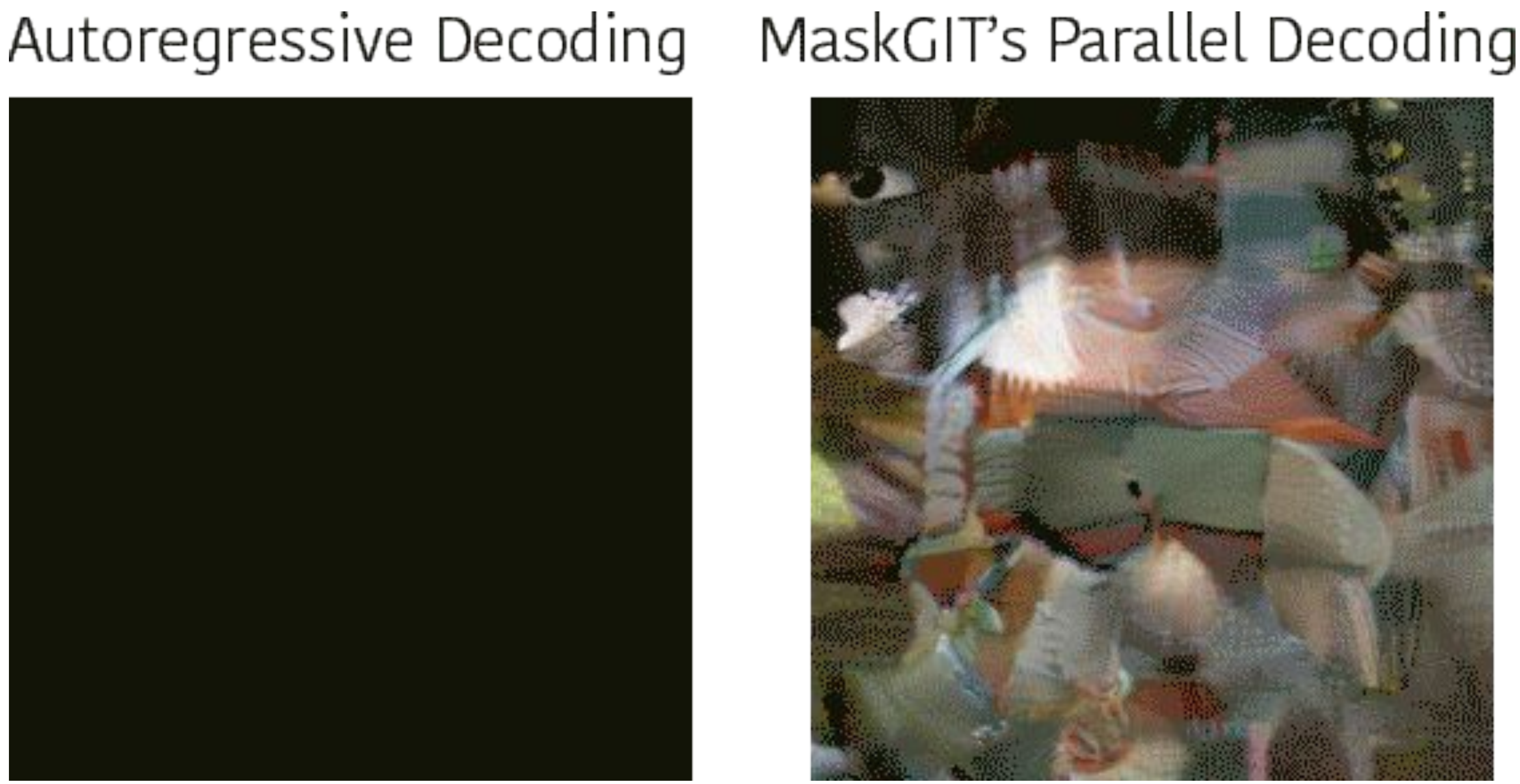
Iterative Decoding

Method



Scheduled Parallel Decoding

Comparison against Sequential Decoding



Masking Design

Method

- $\gamma(\cdot)$ mask scheduling function that computes the mask ratio for the latent tokens
  - Used both in training and inference
    - During inference time, it takes $0/T, 1/T, \cdots, (T-1)/T$ as input and indicates the process in decoding.
    - In training, we sample a ratio r in $[0, 1)$ to simulate various decoding scenarios.

Masking Design

Method

- We need to find a γ that:

  1. $\gamma(r)$ needs to be a continuous function bounded by 0 and 1 for $r \in [0, 1]$.

  2. $\gamma(r)$ should be (monotonically) decreasing with respect to r, and $\gamma(0) \rightarrow 1$ and $\gamma(1) \rightarrow 0$ must hold true.

     - Ensures convergence

Masking Design

Method

- We try 3 different functions for γ :
  - **Linear:** straightforward solution, same amount of tokens each time.
  - **Concave:** less-to-more process
    - Start with most masked tokens, then decrease.
    - Only need to make a few correct predictions to feel confident.
    - Mask ratio drops sharply towards the end, making model have to make a lot more correct predictions.
    - i.e. cosine, square, cubic, and exponential

Masking Design

Method

- We try 3 different functions for γ :
  - **Linear**
  - **Concave**
  - **Convex:** more-to-less process.
    - Model needs to finalize the vast majority of tokens within the first couple of interactions.
    - i.e. square root and logarithmic.

Experimental Setup

Experiments

- **For each Dataset:**
  - single autoencoder, decoder, and codebook with 1024 tokens on cropped 256x256
  - Autoencoder and codebook can be reused to synthesize 512x512 images
- **Transformer Model:**
  - 24 layers, 8 attention heads, 768 embedding dimensions and 3072 hidden dimensions
  - positional embedding, LayerNorm, and truncated normal initialization
- **Data Augmentation:** RandomResizeAndCrop
- **Training:**
  - 4x4 TPU devices
  - ImageNet models: 300 epochs. Places2: 200 epochs

Class-conditional Image Synthesis

Diversity

BigGAN-deep (FID=6.95) MaskGIT (FID=6.18) Training Set

Class-conditional Image Synthesis

Quality

Model	FID ↓	IS ↑	Prec ↑	Rec ↑	# params	# steps	CAS ×100 ↑	
							Top-1 (76.6)	Top-5 (93.1)
ImageNet 256×256								
DCTransformer [32] ▫	36.51	n/a	0.36	**0.67**	738M	>1024		
BigGAN-deep [4]	6.95	**198.2**	**0.87**	0.28	160M	1	43.99	67.89
Improved DDPM [33]▫	12.26	n/a	0.70	0.62	280M	250		
ADM [12]▫	10.94	101.0	0.69	0.63	554M	250		
VQVAE-2 [37]▫	31.11	~45	0.36	0.57	13.5B†	5120	54.83	77.59
VQGAN [15]▫	15.78	78.3	n/a	n/a	1.4B	256		
VQGAN*	18.65	80.4	0.78	0.26	227M	256	53.10	76.18
MaskGIT (Ours)	**6.18**	182.1	0.80	0.51	227M	*8*	**63.14**	**84.45**
ImageNet 512×512								
BigGAN-deep [4]	8.43	**232.5**	**0.88**	0.29	160M	1	44.02	68.22
ADM [12]▫	23.24	58.06	0.73	**0.60**	559M	250		
VQGAN*	26.52	66.8	0.73	0.31	227M	1024	51.29	74.24
MaskGIT (Ours)	**7.32**	156.0	0.78	0.50	227M	*12*	**63.43**	**84.79**

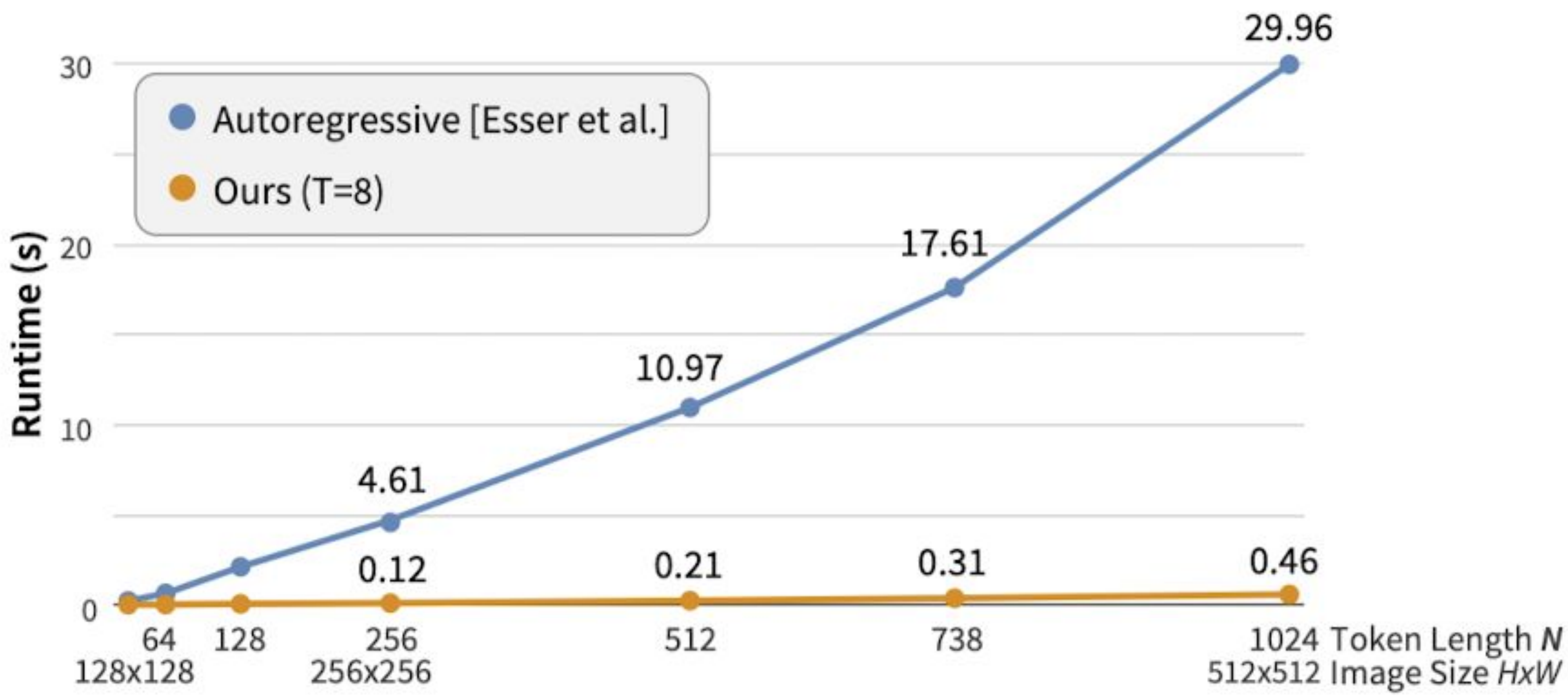
Class-conditional Image Synthesis

Speed

Model	FID ↓	IS ↑	Prec ↑	Rec ↑	# params	# steps	CAS ×100 ↑	
							Top-1 (76.6)	Top-5 (93.1)
ImageNet 256×256								
DCTransformer [32] $^{\square}$	36.51	n/a	0.36	**0.67**	738M	>1024		
BigGAN-deep [4]	6.95	**198.2**	**0.87**	0.28	160M	1	43.99	67.89
Improved DDPM [33]$^{\square}$	12.26	n/a	0.70	0.62	280M	250		
ADM [12]$^{\square}$	10.94	101.0	0.69	0.63	554M	250		
VQVAE-2 [37]$^{\square}$	31.11	~45	0.36	0.57	13.5B†	5120	54.83	77.59
VQGAN [15]$^{\square}$	15.78	78.3	n/a	n/a	1.4B	256		
VQGAN*	18.65	80.4	0.78	0.26	227M	256	53.10	76.18
MaskGIT (Ours)	**6.18**	182.1	0.80	0.51	227M	*8*	**63.14**	**84.45**
ImageNet 512×512								
BigGAN-deep [4]	8.43	**232.5**	**0.88**	0.29	160M	1	44.02	68.22
ADM [12]$^{\square}$	23.24	58.06	0.73	**0.60**	559M	250		
VQGAN*	26.52	66.8	0.73	0.31	227M	1024	51.29	74.24
MaskGIT (Ours)	**7.32**	156.0	0.78	0.50	227M	*12*	**63.43**	**84.79**

Class-conditional Image Synthesis

Speed



Transformer wall-clock runtime comparison

Class-conditional Image Synthesis

Diversity

Model	FID ↓	IS ↑	Prec ↑	Rec ↑	# params	# steps	CAS ×100 ↑	
							Top-1 (76.6)	Top-5 (93.1)
ImageNet 256×256								
DCTransformer [32] ▫	36.51	n/a	0.36	**0.67**	738M	>1024		
BigGAN-deep [4]	6.95	**198.2**	**0.87**	0.28	160M	1	43.99	67.89
Improved DDPM [33]▫	12.26	n/a	0.70	0.62	280M	250		
ADM [12]▫	10.94	101.0	0.69	0.63	554M	250		
VQVAE-2 [37]▫	31.11	∼45	0.36	0.57	13.5B†	5120	54.83	77.59
VQGAN [15]▫	15.78	78.3	n/a	n/a	1.4B	256		
VQGAN*	18.65	80.4	0.78	0.26	227M	256	53.10	76.18
MaskGIT (Ours)	**6.18**	182.1	0.80	0.51	227M	*8*	**63.14**	**84.45**
ImageNet 512×512								
BigGAN-deep [4]	8.43	**232.5**	**0.88**	0.29	160M	1	44.02	68.22
ADM [12]▫	23.24	58.06	0.73	**0.60**	559M	250		
VQGAN*	26.52	66.8	0.73	0.31	227M	1024	51.29	74.24
MaskGIT (Ours)	**7.32**	156.0	0.78	0.50	227M	*12*	**63.43**	**84.79**

Class-conditional Image Synthesis

Diversity

Model	FID ↓	IS ↑	Prec ↑	Rec ↑	# params	# steps	CAS ×100 ↑	
							Top-1 (76.6)	Top-5 (93.1)
ImageNet 256×256								
DCTransformer [32] ▫	36.51	n/a	0.36	**0.67**	738M	>1024		
BigGAN-deep [4]	6.95	**198.2**	**0.87**	0.28	160M	1	43.99	67.89
Improved DDPM [33]▫	12.26	n/a	0.70	0.62	280M	250		
ADM [12]▫	10.94	101.0	0.69	0.63	554M	250		
VQVAE-2 [37]▫	31.11	∼45	0.36	0.57	13.5B†	5120	54.83	77.59
VQGAN [15]▫	15.78	78.3	n/a	n/a	1.4B	256		
VQGAN*	18.65	80.4	0.78	0.26	227M	256	53.10	76.18
MaskGIT (Ours)	**6.18**	182.1	0.80	0.51	227M	*8*	**63.14**	**84.45**
ImageNet 512×512								
BigGAN-deep [4]	8.43	**232.5**	**0.88**	0.29	160M	1	44.02	68.22
ADM [12]▫	23.24	58.06	0.73	**0.60**	559M	250		
VQGAN*	26.52	66.8	0.73	0.31	227M	1024	51.29	74.24
MaskGIT (Ours)	**7.32**	156.0	0.78	0.50	227M	*12*	**63.43**	**84.79**

Class-conditional Image Synthesis

Diversity

Class-conditional Image Editing

Image Editing Applications

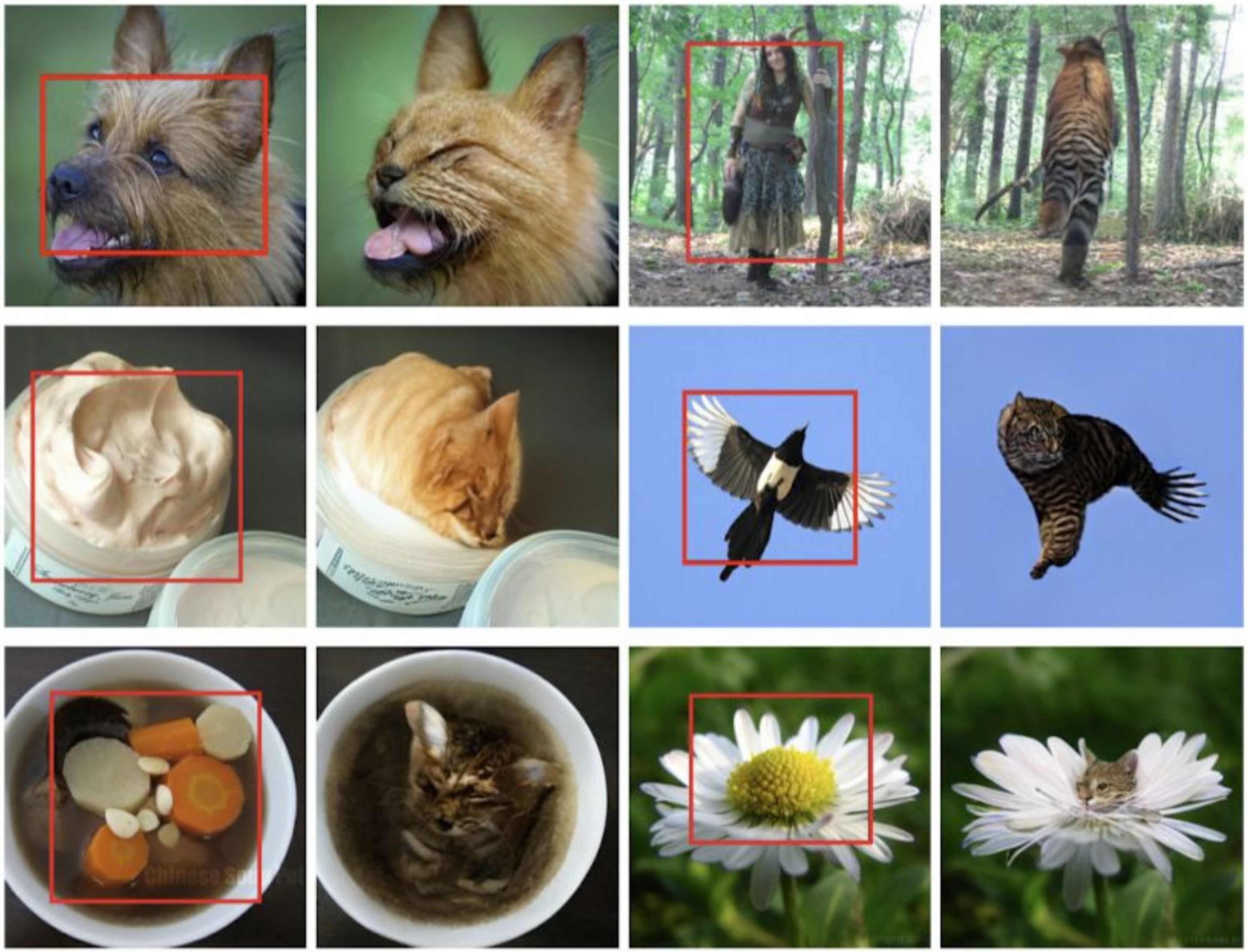
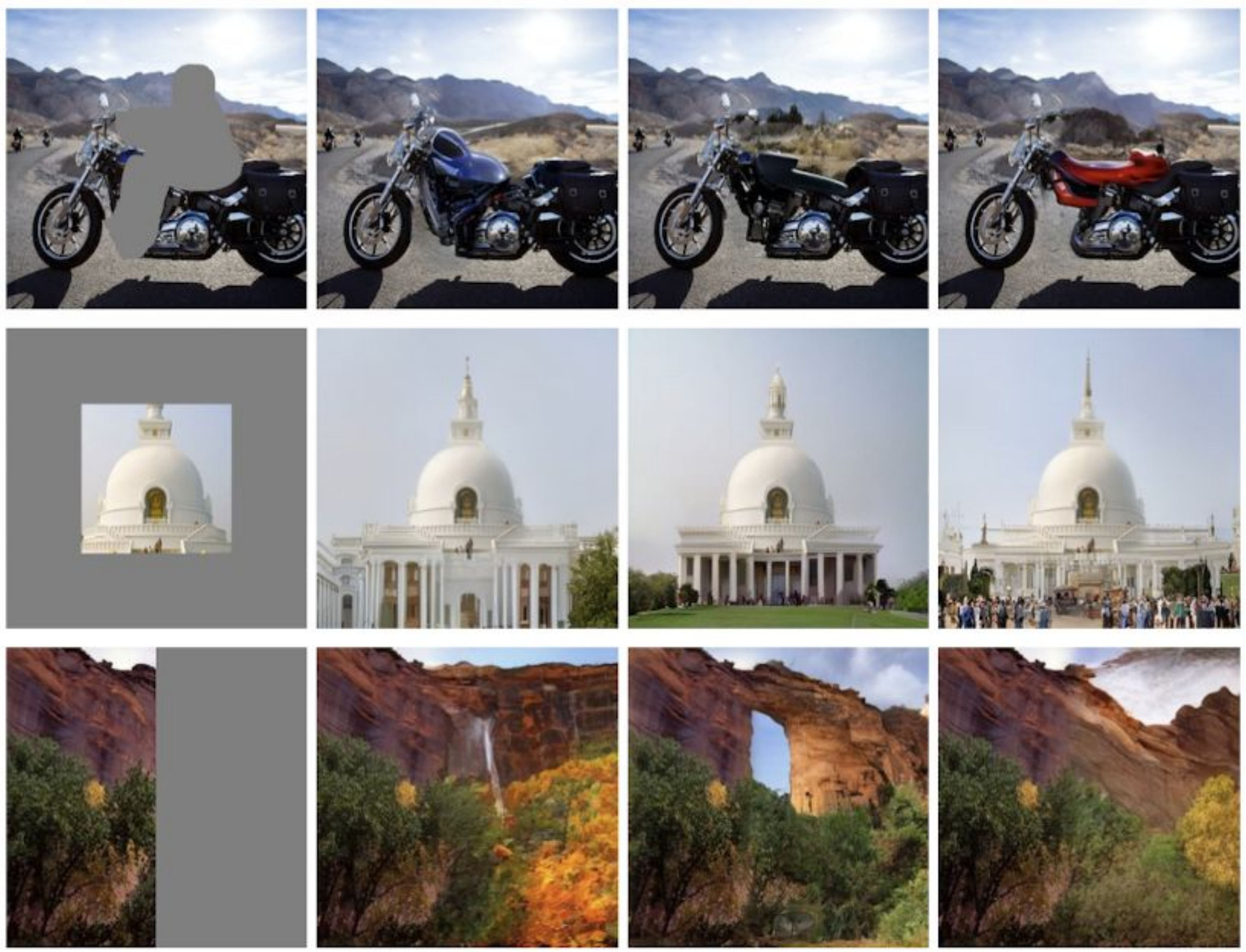


Image Impainting & Outpainting

Image Editing Applications



Input —— MaskGIT (Our Samples) ——

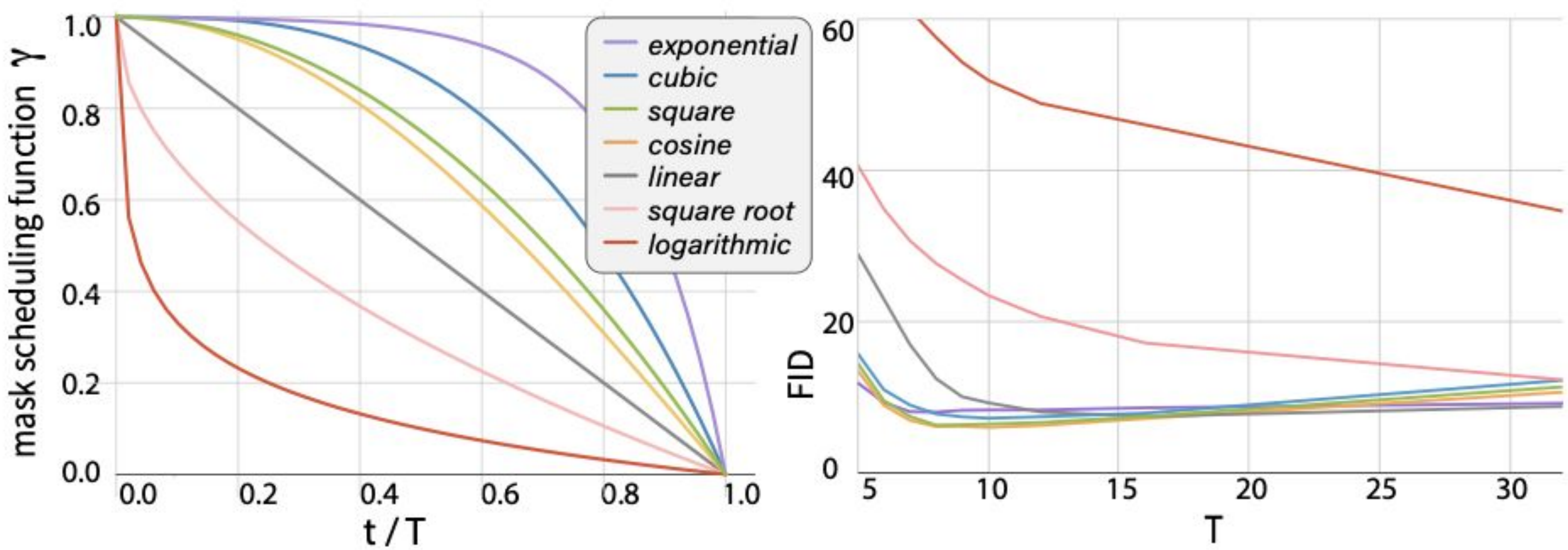
Task	Model	FID ↓	IS ↑
Outpainting	Boundless [43]$^{\square}$	35.02	6.15
Right 50%	In&Out [8]$^{\square}$	23.57	7.18
	InfinityGAN [31]	10.60	5.57
	Boundless [43] TF$^{\blacklozenge}$	7.80	5.99
	MaskGIT (Ours) ^{512}	**6.78**	**11.69**
Inpainting	DeepFill [52]	11.51	22.55
Center 50%×50%	ICT [49]†	13.63	17.70
	HiFill [50]^{512}	16.60	19.93
	CoModGAN [57]^{512}	**7.13**	21.82
	MaskGIT (Ours)^{512}	7.92	**22.95**

Mask Scheduling

Ablation Studies

Ablation Results on the **Mask Scheduling Functions**

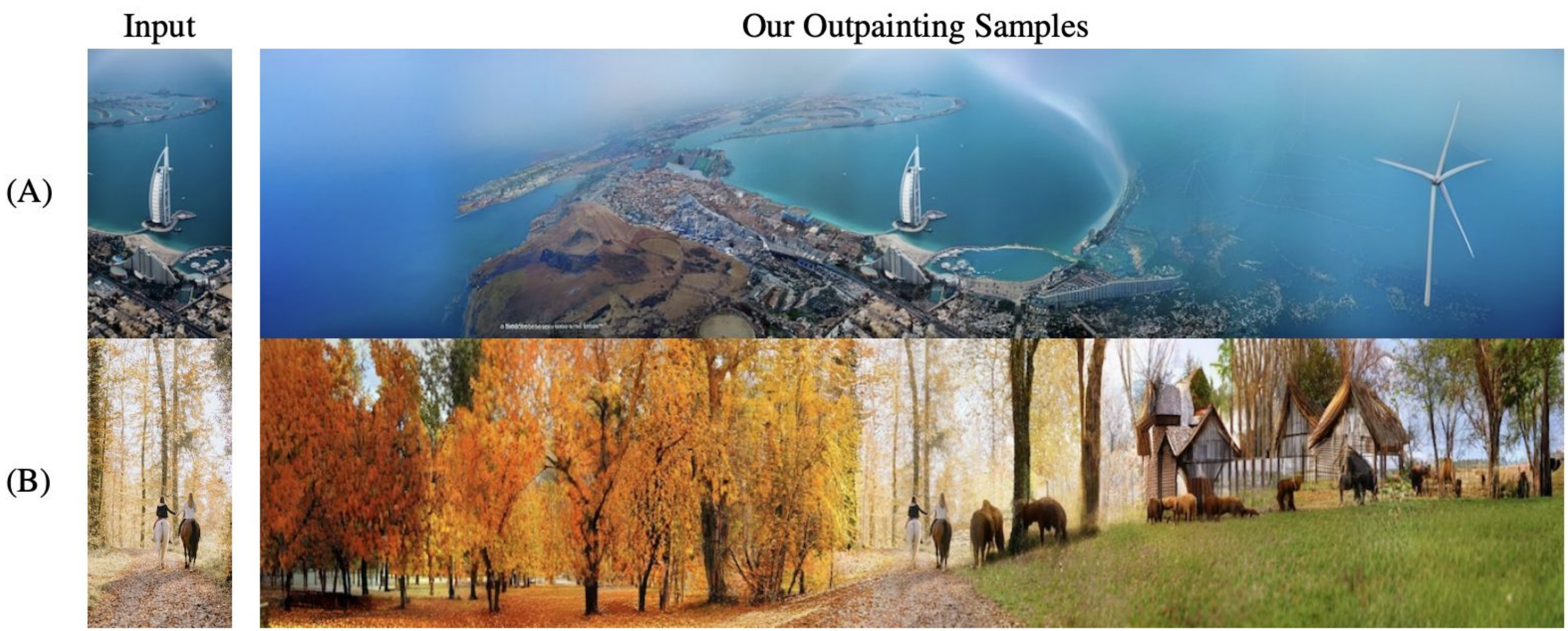
γ	T	FID ↓	IS ↑	NLL
Exponential	8	7.89	156.3	4.83
Cubic	9	7.26	165.2	4.63
Square	10	6.35	179.9	4.38
Cosine	10	**6.06**	**181.5**	4.22
Linear	16	7.51	113.2	3.75
Square Root	32	12.33	99.0	3.34
Logarithmic	60	29.17	47.9	3.08



Choices of **Mask Scheduling Functions** and **Number of Iterations *T***

Limitations and Failure Cases

Semantic and Color Shifts



Limitations and Failure Cases

Outpainting and Inpainting

Limitations and Failure Cases

Outpainting and Inpainting

—Our Class-conditional Samples —

(E)

Summary

Conclusions

- Trained on **Masked Visual Token Modeling** but extendable to various image manipulation tasks
- Significantly **outperforms the SOTA** transformer model on conditional image generation
    - Competitive performance with SOTA GANs
- **Limitations:** Semantic and color shifts; may ignore or modify objects during outpainting and inpainting; oversmoothing on complex structures