Lab5 - MaskGIT for Image Inpainting

2024 Summer

詹雨婷

Important Date

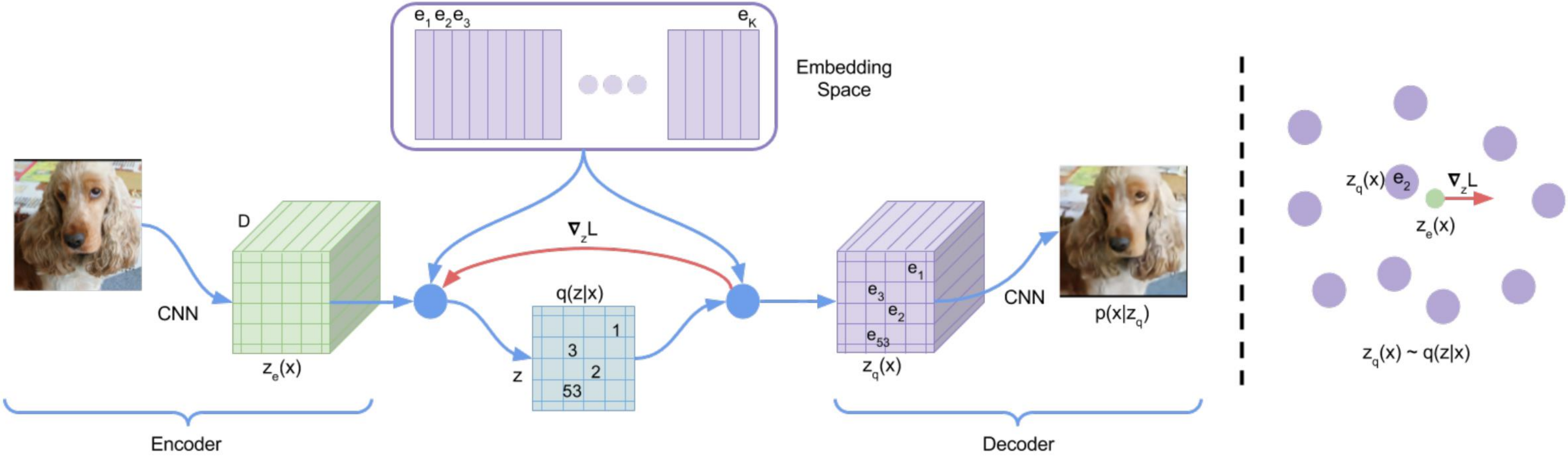
	LAB1 Back-Propagation	LAB2 CNN	LAB3 CNN	LAB4 VAE	LAB5 MaskGIT	LAB6 Generative Models
Announce	7/9 (Tabc)	7/16 (Tabc)	7/23 (Tabc)	7/30 (Tabc)	8/6 (Tabc)	8/13 (Tabc)
DEMO	7/16 (Tabc)	7/23 (Tabc)	7/30 (Tabc)	8/13 (Tabc)	**No demo**	No demo

Submission

- Score: 50% experiment score + 50% report
- If the zip file name or the report spec have format error, you will be punished (-5)
- Submission Deadline: **8/23 (Fri) 11:59 a.m.**
- Turn in: a. Experiment Report (.pdf) b. Source code
- Notice : zip all files in one file and name it like「DL_LAB5_YourStudentID_name.zip」, ex: [DL_LAB5_312581028_詹雨婷.zip」

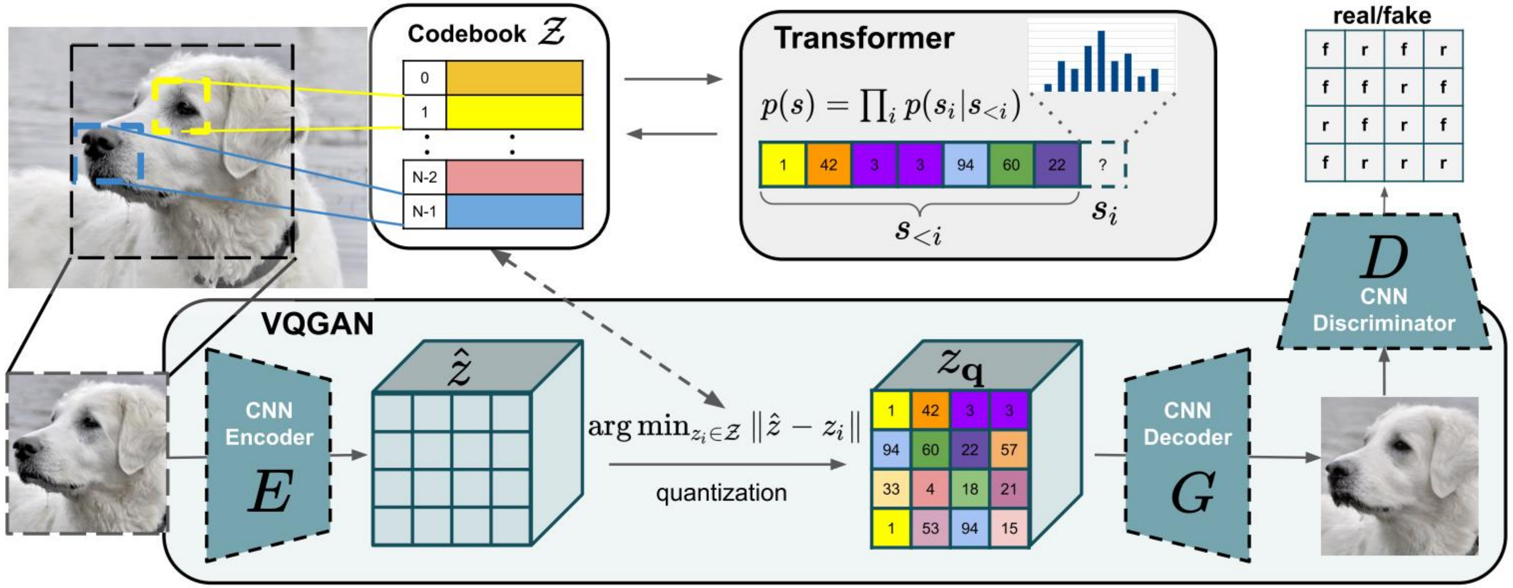
Introduction

VQ-VAE (prior work)



$$q(z=k|x) = \begin{cases} 1 & \text{for } \mathrm{k} = \mathrm{argmin}_j \|z_e(x) - e_j\|_2, \\ 0 & \text{otherwise} \end{cases}$$

VQ-GAN (prior work)



$$\mathcal{L}_{\mathrm{VQ}}(E, G, \mathcal{Z}) = \boxed{\|x - \hat{x}\|^2} + \|\mathrm{sg}[E(x)] - z_{\mathbf{q}}\|_2^2 + \|\mathrm{sg}[z_{\mathbf{q}}] - E(x)\|_2^2.$$

- Perceptual loss replace L2 loss

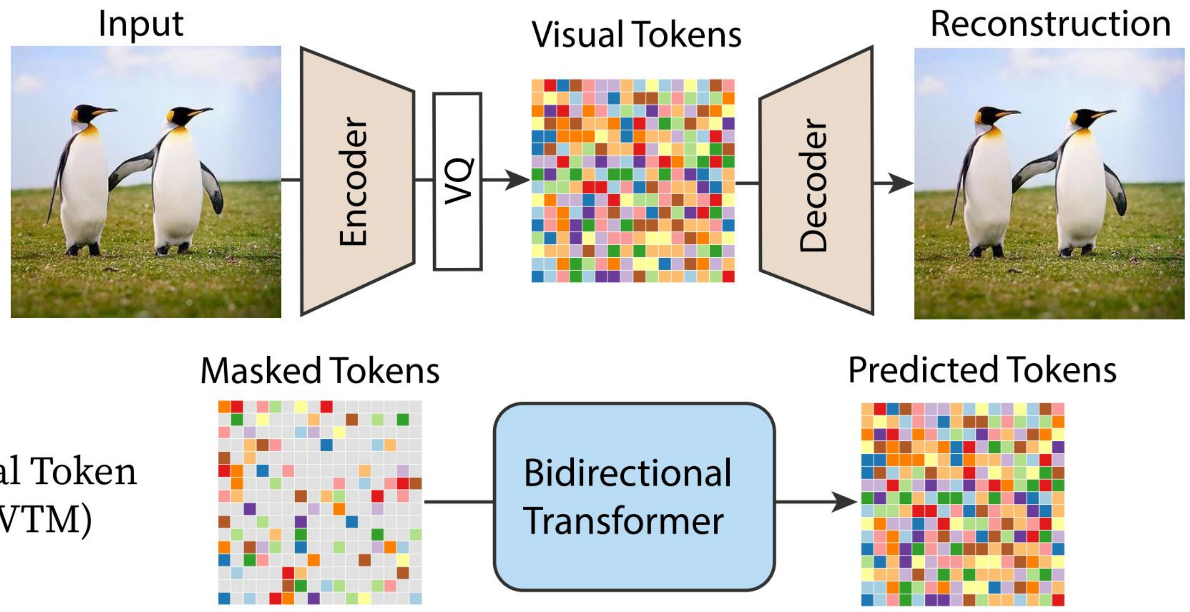
$$\mathcal{L}_{\mathrm{GAN}}(\{E, G, \mathcal{Z}\}, D) = [\log D(x) + \log(1 - D(\hat{x}))]$$

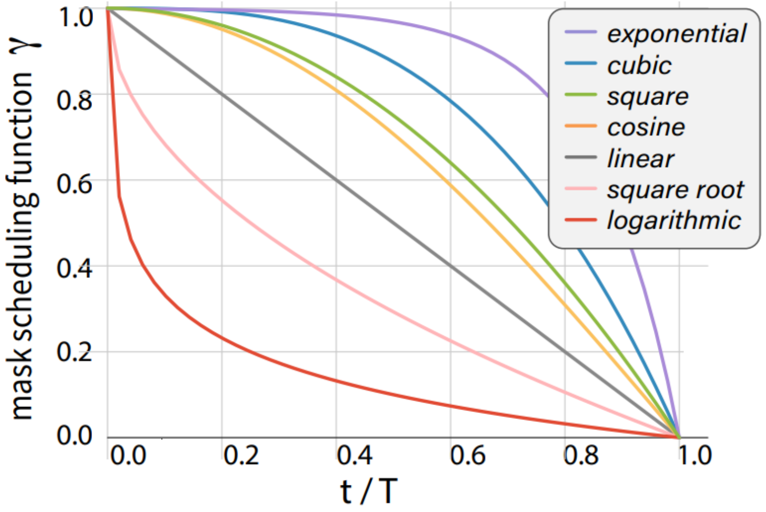
- Transformer (AR model) prior ancestral sampling z

MaskGIT Pipeline Overview

STAGE1 Tokenization

STAGE2 Masked Visual Token Modeling (MVTM)





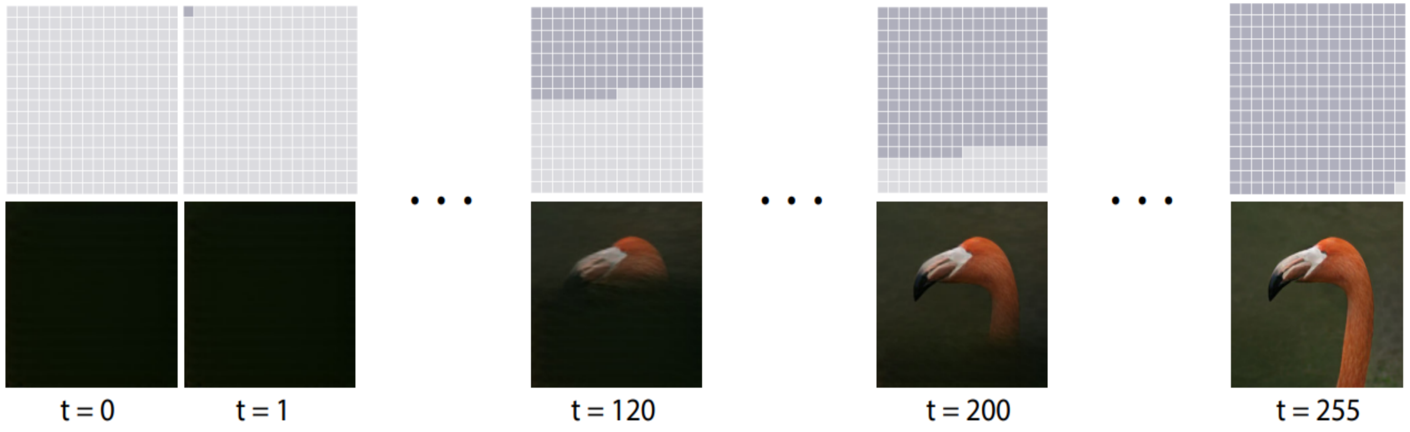
- Transformer (BERT) prior ancestral sampling z
- MVTM in Training $\gamma(r) \in (0,1]$ $\mathcal{L}_{\text{mask}} = -\mathop{\mathbb{E}}_{\mathbf{Y}\in\mathcal{D}}\left[\sum_{\forall i\in[1,N], m_i=1} \log p(y_i|Y_{\overline{\mathbf{M}}})\right]$
- Iterative Decoding

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

Iterative Decoding

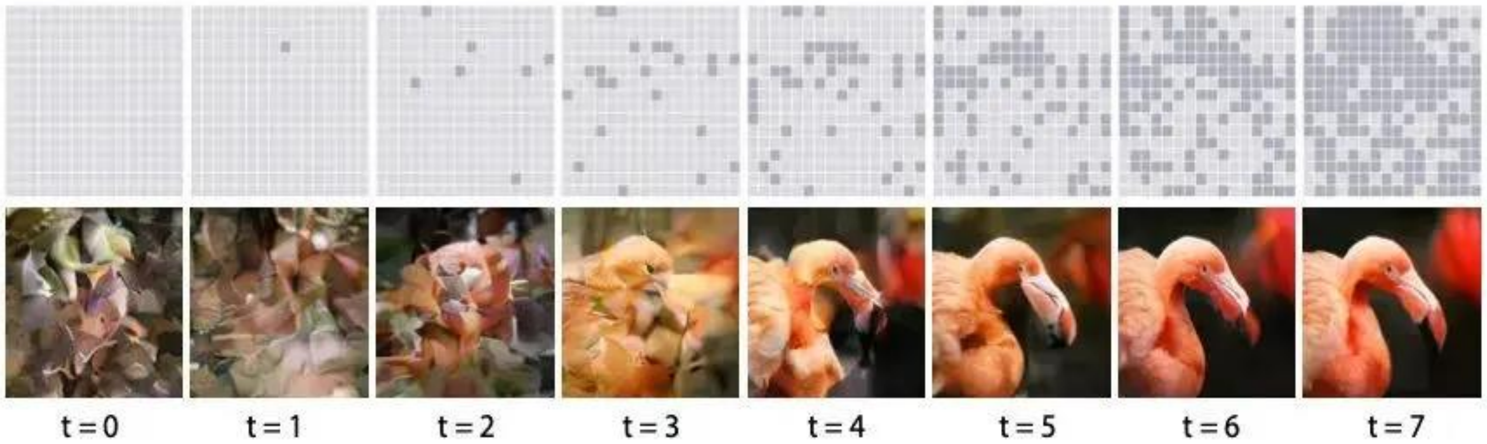
VQGAN

Sequential Decoding with Autoregressive Transformers



MaskGIT

Scheduled Parallel Decoding with MaskGIT

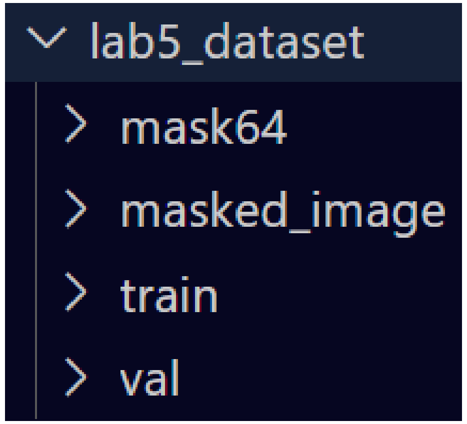


Lab Details

Lab Objective

- Focus on implementing MaskGIT for the inpainting task
- During testing, images contain gray regions indicating missing information, which we aim to restore using MaskGIT.
- The key practical emphasis of this lab lies in three main areas:
  - Multi-head attention
  - Transformer training
  - Inference inpainting

Dataset



a. Training dataset

image: 12000 png files **(./lab5_dataset/train)**

b. Validation dataset

image: 3000 png files **(./lab5_dataset/val)**

c. Testing dataset

masked image: 747 png files **(./lab5_dataset/masked_image)**

mask: 747 png files **(./lab5_dataset/mask64)**

Reference: https://www.kaggle.com/datasets/spandan2/cats-faces-64x64-for-generative-models

VQGAN Stage1 Pretrained Weight

- **You can't modify any model structure or retrain stage1.**

- models
  - __pycache__
  - Transformer
  - VQGAN
    - __pycache__
    - checkpoints
      - VQGAN.pt

- VQGAN
  - __pycache__
  - checkpoints
  - config
    - Discriminator.yml
    - VQGAN.yml

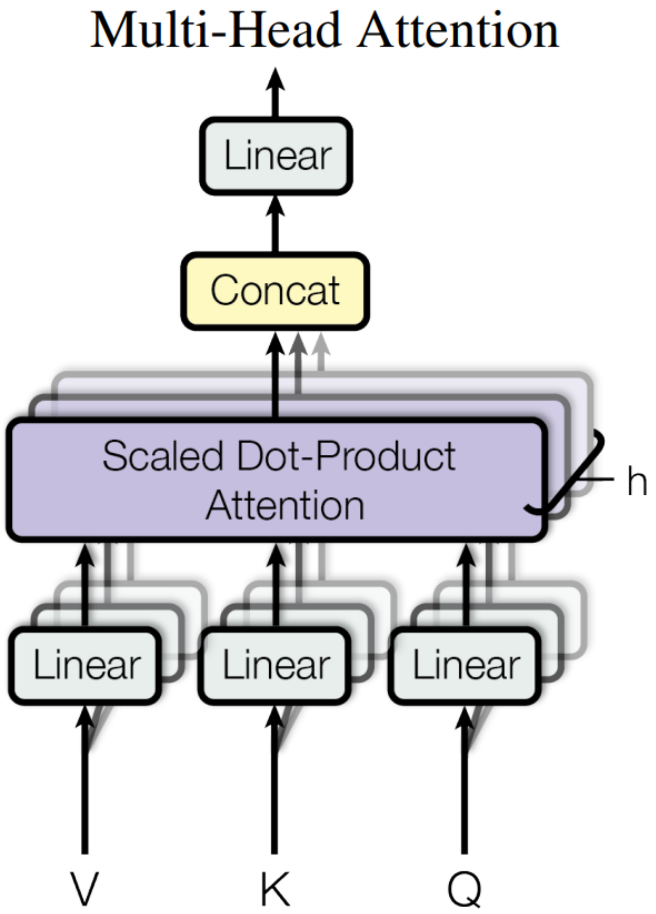
```yaml
---
MODEL_NAME: VQ_GAN

model_param:
  image_channels: 3
  enc_channels: [128, 128, 128, 256, 256, 512]
  dec_channels: [128, 128, 256, 256, 512]
  latent_dim: 256
  img_resolution: 64
  latent_resolution: 16
  num_codebook_vectors: 1024
  beta: 0.25
```

Multi-Head Self-Attention

- You can't use any functions directly ex. torch.nn.MutiheadAttention
- Multi-Head Attention: total #s of head set to 16.
- Total d_k , d_v set to 768
- d_k , d_v for one head will be 768//16.

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head_1}, \ldots, \mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

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



MaskGIT Stage2 Training

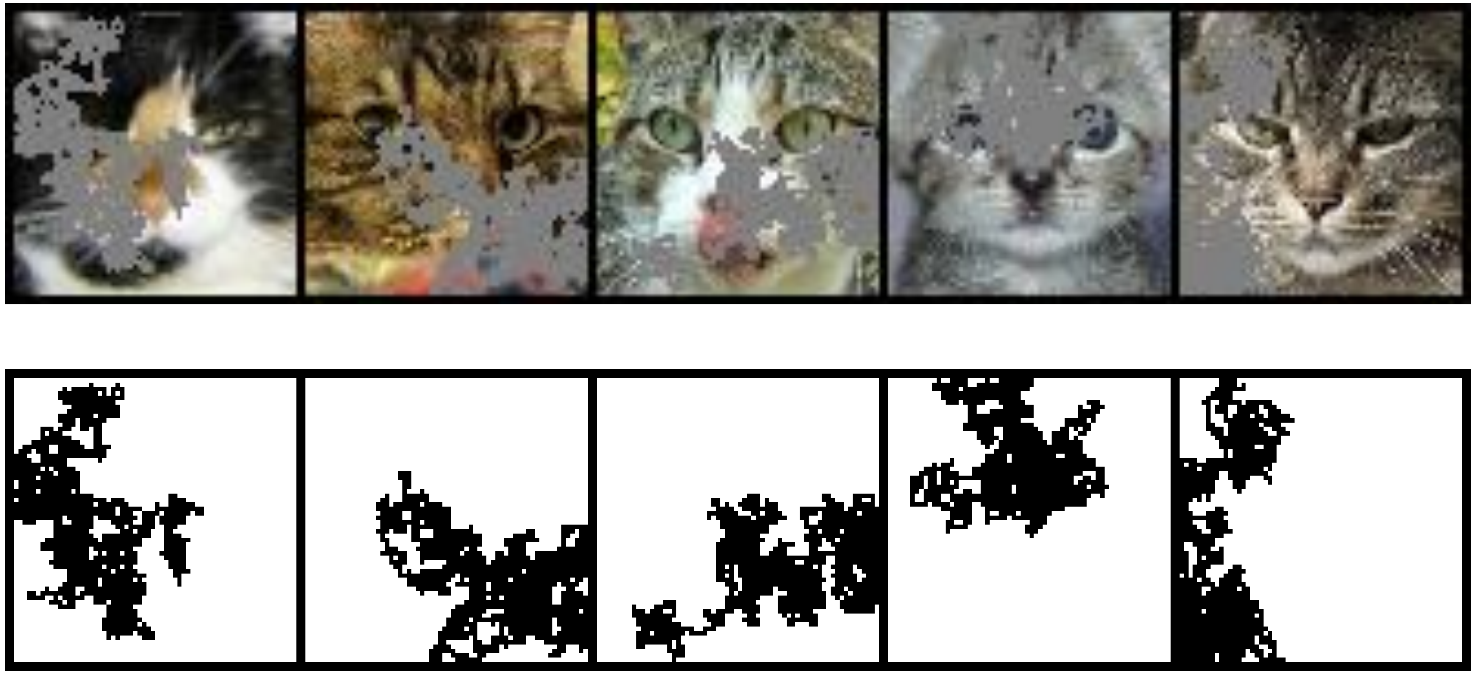
- You can’t modify any model structure.
- Multi-Head Attention: total #s of head set to 16.
- config
  - MaskGit.yml

```yaml
---
MODEL_NAME: MaskGit


model_param:

  VQ_Configs:
    VQ_config_path: models/VQGAN/config/VQGAN.yml
    VQ_CKPT_path: models/VQGAN/checkpoints/VQGAN.pt

  num_image_tokens: 256
  num_codebook_vectors: 1024
  choice_temperature: 4.5
  gamma_type: cosine

  Transformer_param:
    num_image_tokens: 256
    num_codebook_vectors: 1024
    dim: 768
    n_layers: 15
    hidden_dim: 1536
```

- # How to set the Masked token?

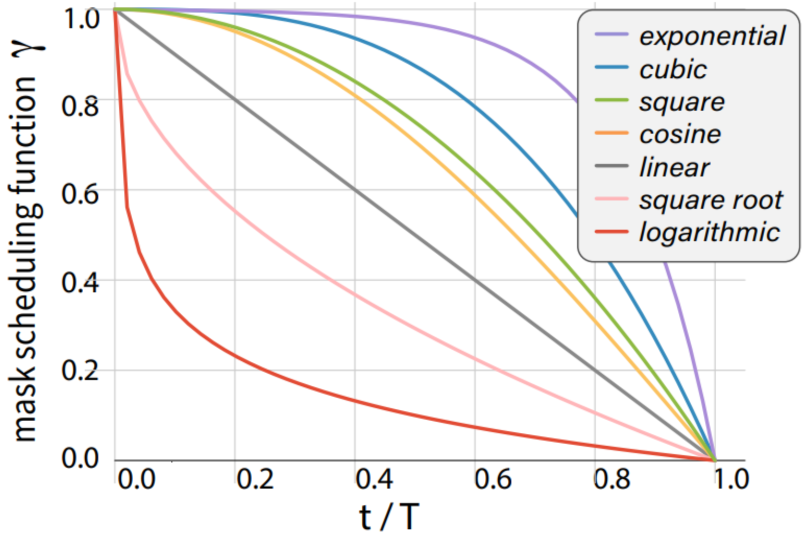
```python
class BidirectionalTransformer(nn.Module):
    def __init__(self, configs):
        super(BidirectionalTransformer, self).__init__()
        self.num_image_tokens = configs['num_image_tokens']
        #mask_token_id:1024
        self.tok_emb = nn.Embedding(configs['num_codebook_vectors'] + 1, configs['dim'])
        self.pos_emb = nn.init.trunc_normal_(nn.Parameter(torch.zeros(configs['num_image_tokens'], configs['dim'])), 0., 0.02)

        self.blocks = nn.Sequential(*[Encoder(configs['dim'], configs['hidden_dim']) for _ in range(configs['n_layers'])])
        self.Token_Prediction = TokenPredictor(configs['dim'])
        self.LN = nn.LayerNorm(configs['dim'], eps=1e-12)
        self.drop = nn.Dropout(p=0.1)

        self.bias = nn.Parameter(torch.zeros(self.num_image_tokens, configs['num_codebook_vectors'] + 1))
        self.apply(weights_init)

    def forward(self, x):
        # Token domain -> Latent domain
        token_embeddings = self.tok_emb(x)

        embed = self.drop(self.LN(token_embeddings + self.pos_emb))
        embed = self.blocks(embed)
        embed = self.Token_Prediction(embed)

        # Latent domain -> Token domain
        logits = torch.matmul(embed, self.tok_emb.weight.T) + self.bias

        return logits
```

Inference for Image Inpainting Task



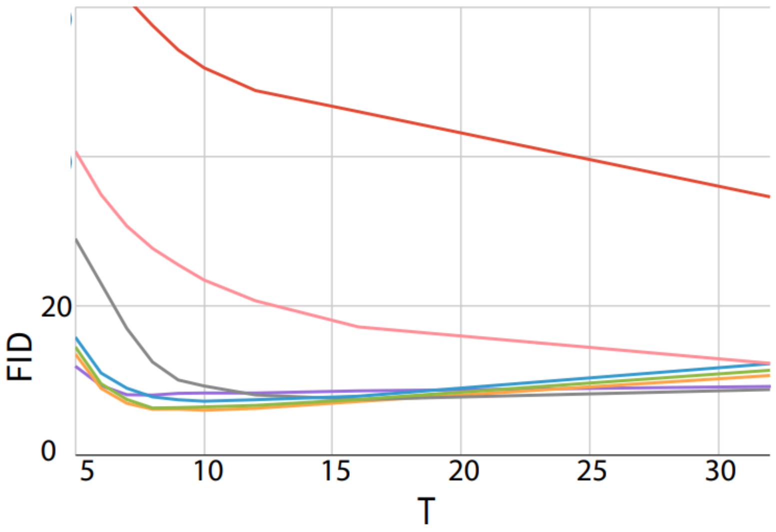
- Tokenize the masked image
- Interpret the **inpainting mask** as the initial mask in iterative decoding

Iterative Decoding



- **Mask Scheduling Functions** $\gamma(\frac{t}{T})$
  - cosine • linear • square
- **Number of iterations** T
  (you can adjust)
- **Sweet spot** t
  (you can adjust)

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



Requirements

1. Download the dataset and pretrained weight of VQGAN (MaksGIT stage1).
2. Implement the Multi-head attention module on your own, if you use any function directly, your score will -15.
3. Train your transformer model (MaskGIT stage2) from scratch.
4. Implement iterative decoding for inpainting task.
5. Compare the FID score with different settings of mask scheduling parameters and visualize the iterative decoding for mask in latent domain or predicted images.

Report Spec (50%)

1. Introduction (5%)

2. Implementation Details (45%)

A. The details of your model (Multi-Head Self-Attention)

(if you directly call any function, you can’t get any score in this part.)

B. The details of your stage2 training (MVTM, forward, loss)

C. The details of your inference for inpainting task (iterative decoding)

3. Discussion(bonus: 10%)

A. Anything you want to share

Experiment Score (50%)
(Prove your code implementation is correct)

- **Experimental results (30%)**
  - show iterative decoding
    - cosine
    - linear
    - square

1.Mask in latent domain

2.Predicted image

Experiment Score (50%)

Average FID	Score
$40 \geq \text{FID}$	20
$45 \geq \text{FID} > 40$	17
$50 \geq \text{FID} > 45$	14
$55 \geq \text{FID} > 50$	11
$60 \geq \text{FID} > 55$	8
$65 \geq \text{FID} > 60$	5
$\text{FID} > 65$	0

- **The Best FID Score(20%)**
  - Screenshot
  - Masked Images v.s MaskGIT Inpainting Results v.s Ground Truth
  - The setting about training strategy, mask scheduling parameters, and so on

```
cd faster-pytorch-fid
python fid_score_gpu.py --predicted-path /path/your_inpainting_results_folder --device cuda:0
```

Masked Images

MaskGIT Inpainting Results

Ground Truth

References

1. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In NeurIPS, 2017. https://arxiv.org/pdf/2202.04200.pdf

2. A. van den Oord, O. Vinyals, et al., “Neural discrete representation learning,” in Advances in Neural Information Processing Systems, pp. 6306–6315, 2017. https://arxiv.org/abs/1711.00937

3. Esser, P., Rombach, R., and Ommer, B.: Taming Transformers for High-Resolution Image Synthesis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12873–12883 (2021) https://arxiv.org/abs/2012.09841

4. Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. Maskgit: Masked generative image transformer. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, June 2022. https://arxiv.org/abs/2202.04200