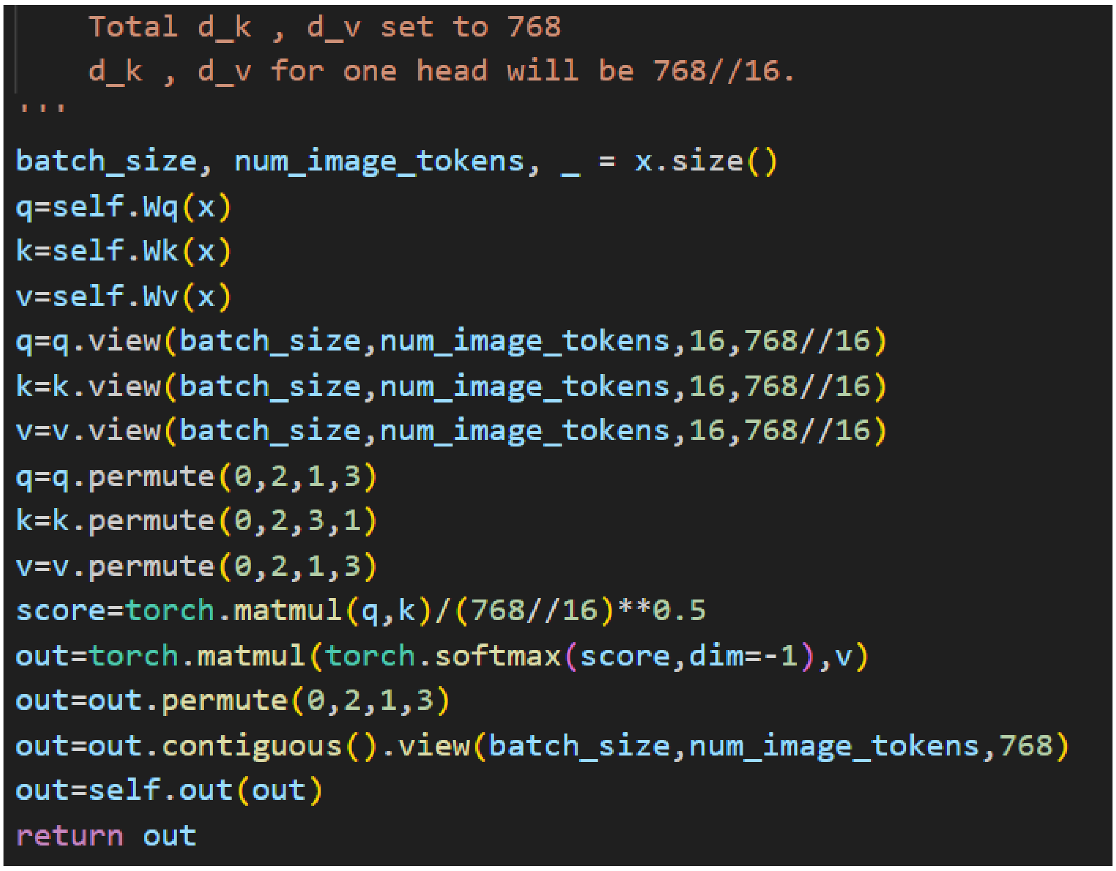
DLP hw5

1. Introduction (5%)

   Train a VQVAE to predict the masked picture. The model use extract the feature from the input image, use codebook to match the feature vector into discreate domain, and use multi-head self-attention to produce the image. The framework to produce the feature vector is already given and cannot be changed so I only train the transformer.

2. Implementation Details (45%)

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



```
    Total d_k , d_v set to 768
    d_k , d_v for one head will be 768//16.
'''
batch_size, num_image_tokens, _ = x.size()
q=self.Wq(x)
k=self.Wk(x)
v=self.Wv(x)
q=q.view(batch_size,num_image_tokens,16,768//16)
k=k.view(batch_size,num_image_tokens,16,768//16)
v=v.view(batch_size,num_image_tokens,16,768//16)
q=q.permute(0,2,1,3)
k=k.permute(0,2,3,1)
v=v.permute(0,2,1,3)
score=torch.matmul(q,k)/(768//16)**0.5
out=torch.matmul(torch.softmax(score,dim=-1),v)
out=out.permute(0,2,1,3)
out=out.contiguous().view(batch_size,num_image_tokens,768)
out=self.out(out)
return out
```

Use linear layer to generate query, key, and value vector. Split all the vector into 16 head and only calculate them in the same head in the following process. The score is calculated by product of query and key. The output of the head is product of score and value. Finally, concat all the output of head and go through the linear layer.

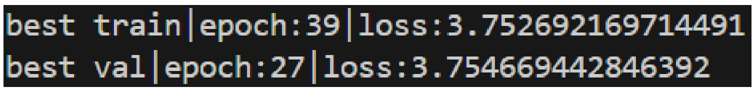
B. The details of your stage2 training (MVTM, forward, loss)

I use Adam as optimizer and cross-entropy error as loss function to train.

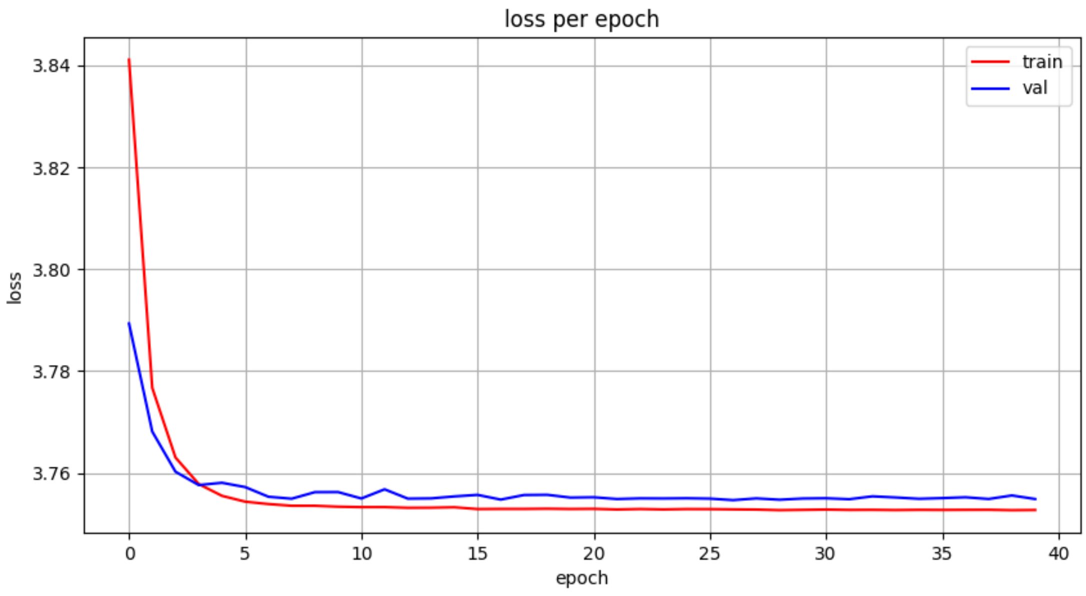
However, I didn’t use scheduler and accumulate gradient.

Every MVTM setting is write in the default of arguments.

The loss is nearly 3.75.



```
best train|epoch:39|loss:3.752692169714491
best val|epoch:27|loss:3.754669442846392
```



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

The inpainting task encode the masked image to feature. Part of the feature will be masked to predict again. The ratio of the masked feature and the total feature is decreased in such ways.

3. Discussion(bonus: 10%)

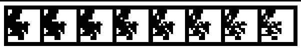
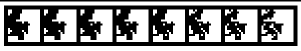
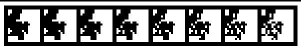
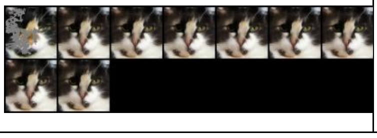
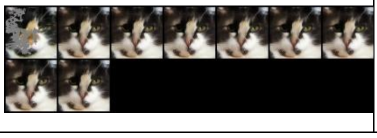
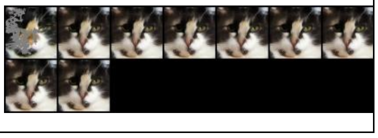
A. Anything you want to share

It seems that the way to schedule the ratio of mask in the inpainting task will not affect the performance. Three different methods(cosine, linear, square) give the same fid score.

Experiment Score

Part1: Prove your code implementation is correct

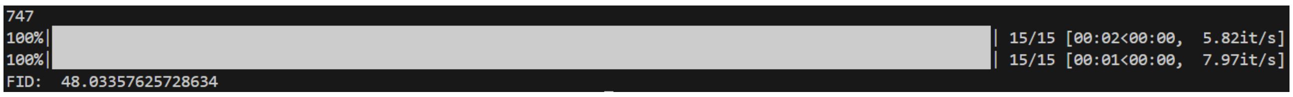
1. Show iterative decoding.

iterative decoding	cosine	linear	square
Mask in latent domain			
Predicted image			

Part2: The Best FID Score (20%)

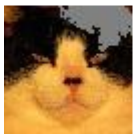
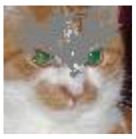
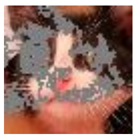
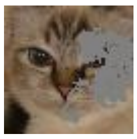
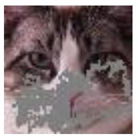
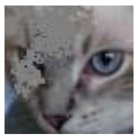
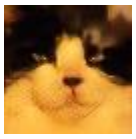
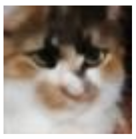
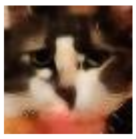
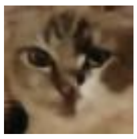
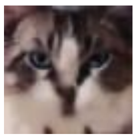
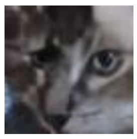
- Screenshot

The best fid is 48



```
747
100%|████████████████████████████████████████| 15/15 [00:02<00:00,  5.82it/s]
100%|████████████████████████████████████████| 15/15 [00:01<00:00,  7.97it/s]
FID:  48.03357625728634
```

- Masked Images v.s MaskGIT Inpainting Results v.s Ground Truth

masked						
predict						

There is no ground truth in inpainting dataset.

- The setting about training strategy, mask scheduling parameters, and so on

training strategy: learning rate = 0.001, epoch = 30, batch-size = 10

mask scheduling: mask-function = cosine, T = 8, t = 8