Visual Prompting via Image Inpainting

Yue Zhao, Chengcan Gao, Weiye Gao

Abstract

We analyze and implement an image restoration algorithm：Visual Prompting via Image Inpainting train based on datasets CIFAR10，model-based algorithm MAE-VQGAN,And compare the performance of the algorithm in the downstream task (foreground segmentation, colorization, reasoning,segmentation),colorization evaluation is based on ImageNet, foreground segmentation and segmentation evaluation is based on Pascal,We have three members in our team,**Yue Zhao** responsible for writing papers and collecting data sources，**Chengcan Gao** is responsible for running Visual Prompting via Image Inpainting code and experimental results，**Weiye Gao** take the responsibility for presentation.

1.Introduction

features learned via self-supervision are not “ready for use” – they typically need to be adapted for agiven downstream task by fine-tuning on some labeled dataset. In this paper we take a step toward this goal by demonstrating that large-capacity image inpainting models, when trained on the right data, can be surprisingly effective tools for visual prompting.

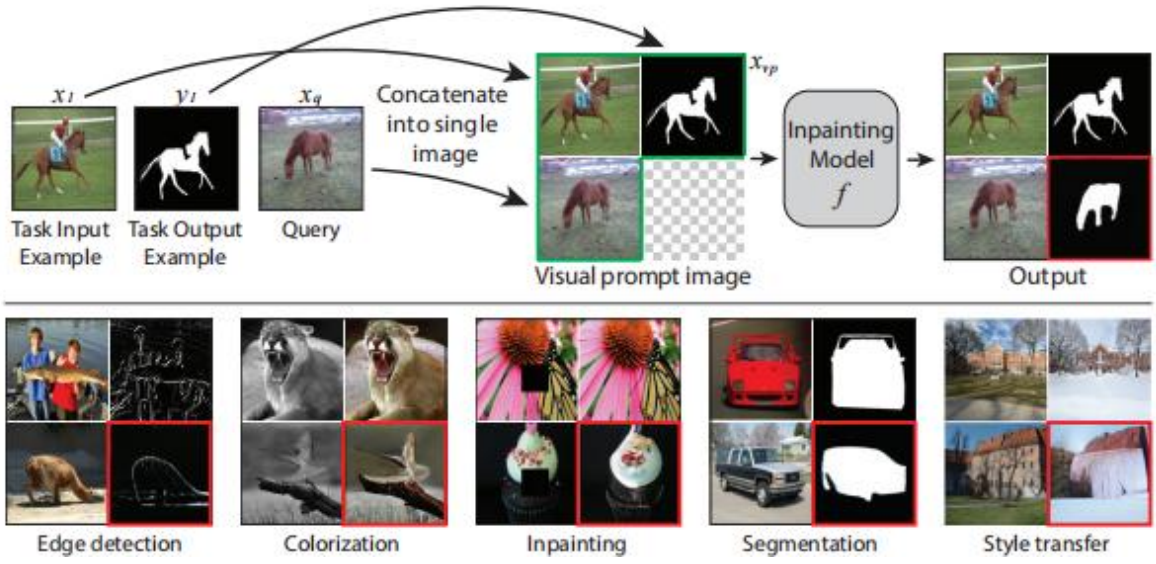

Figure 1:Visual prompting via Image Inpainting

Visual prompting via Image Inpainting. Top: Prompting Image Inpainting Models. Given input output example(s) (x1, y1) and image query x_q, we construct a grid-like single image called a visual prompt x_{vp}. The visual prompt is composed of the desired task example(s) and a new query image (all in green). The inpainting model goal is then to predict the masked region (red) such that it is consistent with the example(s).Bottom: an inpainting model can solve this

way various computer vision tasks, given that it was trained on the right data. The model predictions are annotated in red.

Below we focus on data sources, algorithmic principles, and the results of downstream tasks and final runs.

2.The Computer Vision Figures Dataset

2.1CIFAR10

The dataset consists of 60,000 32 *32 color images with 10 categories. 6000 images per category.
We split the set of train set to 50000 and test as 10000.

This data set is used for pretrain， see [prep_cifar.py] for the code.

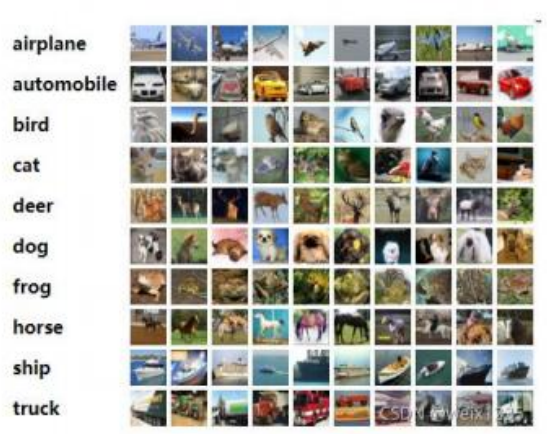

Figure 2: CIFAR10 Dataset

```
# Specifies the path to the local CIFAR-10 data set
cifar_10_data_dir = 'D:/python code/visual_prompting-main/figures_dataset/cifar-10-batches-py'
(train_images, train_labels), (test_images, test_labels) = load_cifar_10_data(cifar_10_data_dir)
# 标准化数据
# train_images = train_images / 255.0
# test_images = test_images / 255.0
```

2.2ImageNet

ImageNet has data for classification, localization, and evaluation of detection tasks.

Similar to categorical data, there are 1000 categories for location tasks. The accuracy rate is calculated based on the top five test results.

All images have at least one border. The detection problem for 200 targets has 470,000 images, with an average of 1. 1 targets per image.

Figure 3: ImageNet Dataset

2.3Pascal

This data set is used for foreground segmentation and segmentation evaluation， see main pretrain.py for the code.

voc 2007 contains a training set (5011 images), a test set (4952 images), a total of 9963 images, containing a total of 20 classes. The subfolders in the main folder are annotation_cache, ImageSets, JPEGImage, results, SegmentationClass, and SegmentationObject.。

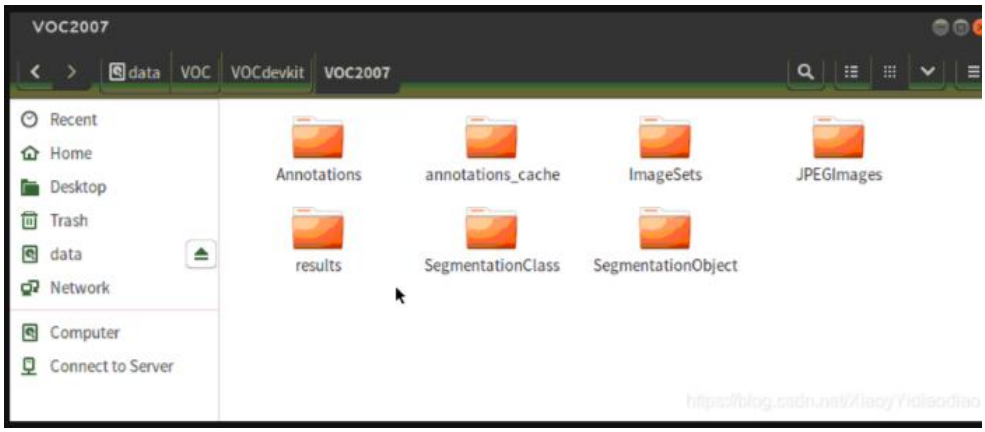

Figure 4: Pascal Dataset

- Annotations

The Annotation folder stores an xml annotation file that corresponds to the name in the JPEGImage folder. The contents of the images in the JPEGImage folder mainly include the following fields: Name of the image, width, height, name of the target of interest, attitude of the target of interest, whether the target of interest is truncated, whether the target of interest is difficult to train, position information of the target of interest: top-left coordinates and lower-right coordinates.

- annotation_cache

The annotation_cache folder caches the information of the files in the Annotation folder and saves them in pkl files.

- ImageSets

The following subfolders are: Layout, Main, Segmentation; txt file is stored, txt file each line contains a JPEGImage folder image name, the Main folder contains training, test picture collection, some txt file will add 1 or -1 at the end to indicate positive and negative samples.

- JPGEImages

The JPGEImages folder contains jpg original files.

- SegmentationClass &SegmentationObject

The SegmentationClass folder holds images divided by Class. The SegmentationObject folder stores images that are segmented by Object.

- results

Store the result of each graph in each class.

For example, det_test_aeroplane.txt is used to detect the aeroplane class. In the txt file, the first line indicates the confidence and coordinates of the aeroplane class in the second graph.

3.Visual Prompting via Image Inpainting

We turn to describe how to perform visual prompting using Image Inpainting models. we describe our proposed inpainting model, which is a combination of MAE and VQGAN. We then proceed to discuss visual prompting and propose different ways to create visual prompts .Finally, we describe the dataset we collected for training our model in Section 2 The training process is illustrated in Section 3.2.

The algorithms for image restoration are BEIT, VQGAN, MAE, and MAE-VQGAN.

1.BEIT

2.MAE

3.MAE-VQGAN

3.1 MAE-VQGAN

MAE-VQGAN assigns probabilities to visual tokens through the softmax layer. During the training process, the true visual token is obtained by mapping the image to the visual token index using the VQGAN encoder. The cross-entropy loss is used to train the model.

VQGAN is an autoregressive transformer model used for inpainting and image generation.

Visual tokens are predicted sequentially, line-by-line, and the model is trained using cross-entropy loss. The VQGAN model codebook is used to encode visual tokens, and it is trained beforehand using perceptual loss and GAN loss We train it on ImageNet and our Figures dataset,following hyperparams in , and use a pretrained codebook with a vocabulary of size |V | = 1024.

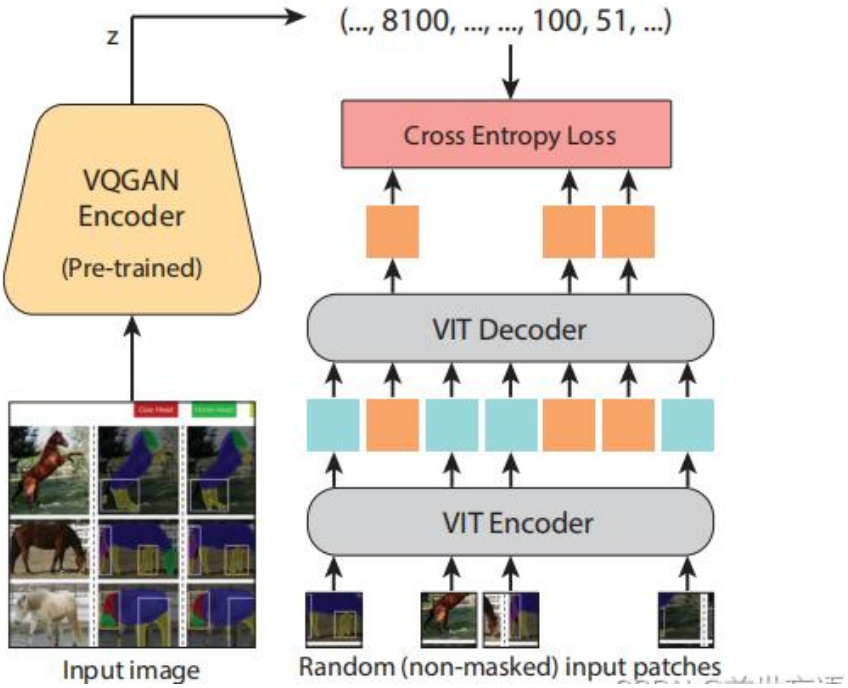

Figure 5: MAE-VQGAN

Figure 6: MAE-VQGAN Architecture. During training, an input image is patchified, masked and fed into an MAE . For each masked token, the decoder outputs a distribution over a pretrained VQGAN codebook. The model is trained using cross entropy loss.

- Inpainting using MAE-VQGAN

Given an input image x $\in R^{H*W*3}$and a binary mask m $\in \{0, 1\}^{H\times W}$, the goal of an in painting function f is to synthesize a new image y $\in R^{H\times W\times 3}$, with the masked regions filled:y = f(x, m),To implement f with a neural network, it is necessary to consider design choices like the network architecture, how to train it, and whether it outputs a distribution over possible completions or pixels.

We propose the MAE-VQGAN model, which combines ideas from MAE and VQGAN .Following the design of MAE, the model is based on ViT and it is trained via masked auto-encoding by randomly masking image patches and then applying f to reconstruct the image from the non-masked parts. MAE-VQGAN models the distribution $P_\theta\theta(z^i|x, m)$, where zi $\in$ V is a visual token from a VQGAN vocabulary V that corresponds to the ith ViT patch. For simplicity,we use a fixed ImageNet pretrained VQGAN codebook.2 Unlike MAE which directly predicts pixels,MAE-VQGAN assigns probabilities to visual tokens via a softmax layer, which is better suited for capturing ambiguities. During training, we obtain ground truth visual tokens by mapping the image to visual tokens indices using the VQGAN encoder. The model is trained using cross entropy loss.

Let $\widehat{Z} = (\widehat{z1}, ..., \widehat{zk})$ be the ordered set of predicted visual tokens. To obtain $\widehat{zi}$, we use argmax:$\widehat{zi} = \underset{z^i}{\text{argmax}}$ zip$^\theta$(z^i|x, m)， Then, to decode the visual tokens to pixels, we apply VQGAN decoder to $\widehat{Z}$ to obtain y.To prompt an inpainting model, we construct a visual prompt, a grid-like image composed of task input-output example(s), and a new query image. The model then has to inpaint the rest of the image such that it is consistent with the task defined in the examples .

- Prompting Inpainting Models

Let S = $\{(xi, yi)\}_{i=1}^{n}$ be the set of input-output examples where x^i is an image and y^i is a function of x^i (e.g y^i is a

segmentation mask). We assume n is small (one or few examples). Then, given S and a new input query x^q, the goal is to predict the corresponding label y^q. we need to define a function g that maps the examples set S and query image x^q to a new image and a mask:

$$[x_{vp}, m] = g(S, x_q)$$

The image x_{vp} is the visual prompt and the mask m defines the masked region f has to predict. For a given task, there might exist multiple implementations of g that can be considered. The goal of the inpainting model is to reason about the visual prompt x_{vp}, and output a plausible completion without performing any additional training:

$$y_{vp} = f(x_{vp}, m)$$

To obtain y_q, we just take the part of y_{vp} corresponding to the mask m.

3.2 CODE

Specific source code see

VQGAN:visual_prompting-main123\visual_prompting-main\vqgan.py

models_mae：visual_prompting-main123\visual_prompting-main\models_mae.py

4.Downstream Vision Tasks

- ✓ **Image classification.** For image classification tasks, we directly employ a simple linear classifier as the task layer. Specifically, we use average pooling to aggregate the representations, and feed the global to a softmax classifier. The category probabilities are computed as softmax, We maximize the likelihood of labeled data by updating the parameters of MAE-VQGAN and the softmax classifier.
- ✓ **Foreground Segmentation**. The goal is to binary-segment the query image to Foreground and Background. The example is an image and corresponding binary segmentation mask. The query is a new image, and the goal is to complete a corresponding segmentation mask. We use the Pascal-5i dataset, which is comprised of 4 different image splits where every split contains between 346 and 725 images and associated segmentation masks. For each class, the data contains a few image-mask pairs, together with held-out image queries. For every image query, we choose one random example pair. To evaluate, every pixel in the completed image is first mapped to the nearest Foreground or Background color. Finally, we report the mean IOU (mIOU) metric.
- ✓ **single object detection.**Similarly to Foreground Segmentation, the goal here is to binary-segment the object that appears in the query image. However, this task is more challenging than Foreground Segmentation because the example mask is obtained from a bounding box which is more coarse than a segmentation mask. We use the Pascal VOC 2012 dataset using images and their associated detection boxes. For simplicity, we use Pascal annotations to include only images with a single object and filter out trivial images that have an object covering more than 50% of the image. We randomly select an example pair and image query of the same object class and repeat the process with 4 different random seeds. For evaluation, we follow a similar process as in Foreground Segmentation to obtain a binary segmentation mask. Then we keep the connected component with the largest area using morphological operations and draw a bounding box around it. We report the results.
- ✓ **Colorization.**The goal is to map a gray-scale image to a color image. The example pair is a grayscaled image and the corresponding color image.

5.Experiments and Results

To study visual prompting, we pretrain different models on the Figures dataset CIFAR10 ,then quantitatively evaluate the models using different prompts on simple downstream computer vision tasks

1.**Visual Prompt.** Given one example pair and a query image, we structure the prompt in the same fashion for all tasks. reasoning algorithm here limits that color can only be a 2*3 matrix, that is, white and target color. But read this code: np.reshape(target,(-1,3)) reshape the input image target from an M*N*3 image matrix to a K*3 matrix, where K = M*N, and each row represents the RGB value of a pixel. np.unique(... axis=0) means to find the only row in the K*3 matrix, that is, if there are duplicate colors, only one will be retained. So the final output size of color should be X*3, X depends on the different colors in the target image.

2.**Computer vision tasks**. We evaluate the inpainting models on standard image to image tasks like Foreground Segmentation, Single Object Detection and Colorization.

3.**Results.** We include quantitative results , and qualitative completion results .Training on the Figures dataset improves the results for MAE-VQGAN most models in all the downstream tasks.

5.1single object detection

The evaluation results are in the folder: visual_hue-main \evaluate\result\segmentation

Namespace(model='mae_vit_large_patch16',output_dir='C:/Users/70241/PycharmProjects/visual_prompting-mai

n/evaluate/result/segmentation', device='cuda',
base_dir='C:/Users/70241/PycharmProjects/visual_prompting-main/figures_dataset', seed=0, t=[0, 0, 0],
task='segmentation', ckpt='C:/Users/70241/PycharmProjects/visual_prompting-main/output_dir/checkpoint-1000.pth',
dataset_type='**pascal_det**', split=0, purple=0, flip=0)

A total of 558 images are produced, and the accuracy is

{'iou': 0.22901295841043023, 'color_blind_iou': 0.22901295841043023, 'accuracy': 0.8700871252725434}

log - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

Namespace(model='mae_vit_large_patch16', output_dir='C:/Users/70241/PycharmProjects/visual_prompting-main/evaluate/res

0 {'iou': 0.3224299, 'color_blind_iou': 0.3224299, 'accuracy': 0.8587776966155345}
1 {'iou': 0.0, 'color_blind_iou': 0.0, 'accuracy': 0.9709439168898628}
2 {'iou': 0.0, 'color_blind_iou': 0.0, 'accuracy': 0.970375781186592}
3 {'iou': 0.053301513, 'color_blind_iou': 0.053301513, 'accuracy': 0.9034169304439574}
4 {'iou': 0.33212176, 'color_blind_iou': 0.33212176, 'accuracy': 0.880691502313124}
5 {'iou': 0.3110266, 'color_blind_iou': 0.3110266, 'accuracy': 0.9264670075480886}
6 {'iou': 0.479067, 'color_blind_iou': 0.479067, 'accuracy': 0.9293076860644428}
7 {'iou': 0.2125775, 'color_blind_iou': 0.2125775, 'accuracy': 0.855693531369207}
8 {'iou': 0.28712872, 'color_blind_iou': 0.28712872, 'accuracy': 0.8772826880934989}
9 {'iou': 0.056099396, 'color_blind_iou': 0.056099396, 'accuracy': 0.38957876795714635}
10 {'iou': 0.67887324, 'color_blind_iou': 0.67887324, 'accuracy': 0.9537375213050888}
11 {'iou': 0.6182026, 'color_blind_iou': 0.6182026, 'accuracy': 0.9458647837026215}
12 {'iou': 0.5263827, 'color_blind_iou': 0.5263827, 'accuracy': 0.8790682574466359}
13 {'iou': 0.5667646, 'color_blind_iou': 0.5667646, 'accuracy': 0.892297703108514}
14 {'iou': 0.0, 'color_blind_iou': 0.0, 'accuracy': 0.9556854151448746}
15 {'iou': 0.0, 'color_blind_iou': 0.0, 'accuracy': 0.9785731677623569}
16 {'iou': 0.060231023, 'color_blind_iou': 0.060231023, 'accuracy': 0.9075562048535022}
17 {'iou': 0.0, 'color_blind_iou': 0.0, 'accuracy': 0.9844980115250386}
18 {'iou': 0.184375, 'color_blind_iou': 0.184375, 'accuracy': 0.6822498173849525}
19 {'iou': 0.034648187, 'color_blind_iou': 0.034648187, 'accuracy': 0.559045532018505}
20 {'iou': 0.07153502, 'color_blind_iou': 0.07153502, 'accuracy': 0.6966155344533723}
21 {'iou': 0.04442388, 'color_blind_iou': 0.04442388, 'accuracy': 0.497199902605308}
22 {'iou': 0.0, 'color_blind_iou': 0.0, 'accuracy': 0.8358087817547277}
23 {'iou': 0.10163112, 'color_blind_iou': 0.10163112, 'accuracy': 0.7094391688986283}
24 {'iou': 0.0, 'color_blind_iou': 0.0, 'accuracy': 0.8421394367340314}
25 {'iou': 0.67797554, 'color_blind_iou': 0.67797554, 'accuracy': 0.9530070611151692}
26 {'iou': 0.08614782, 'color_blind_iou': 0.08614782, 'accuracy': 0.848470091713335}
27 {'iou': 0.63098484, 'color_blind_iou': 0.63098484, 'accuracy': 0.9230581933284636}
28 {'iou': 0.48670885, 'color_blind_iou': 0.48670885, 'accuracy': 0.9341774206639072}
29 {'iou': 0.0, 'color_blind_iou': 0.0, 'accuracy': 0.9260611963314666}

Figure 6: ACC

Results：

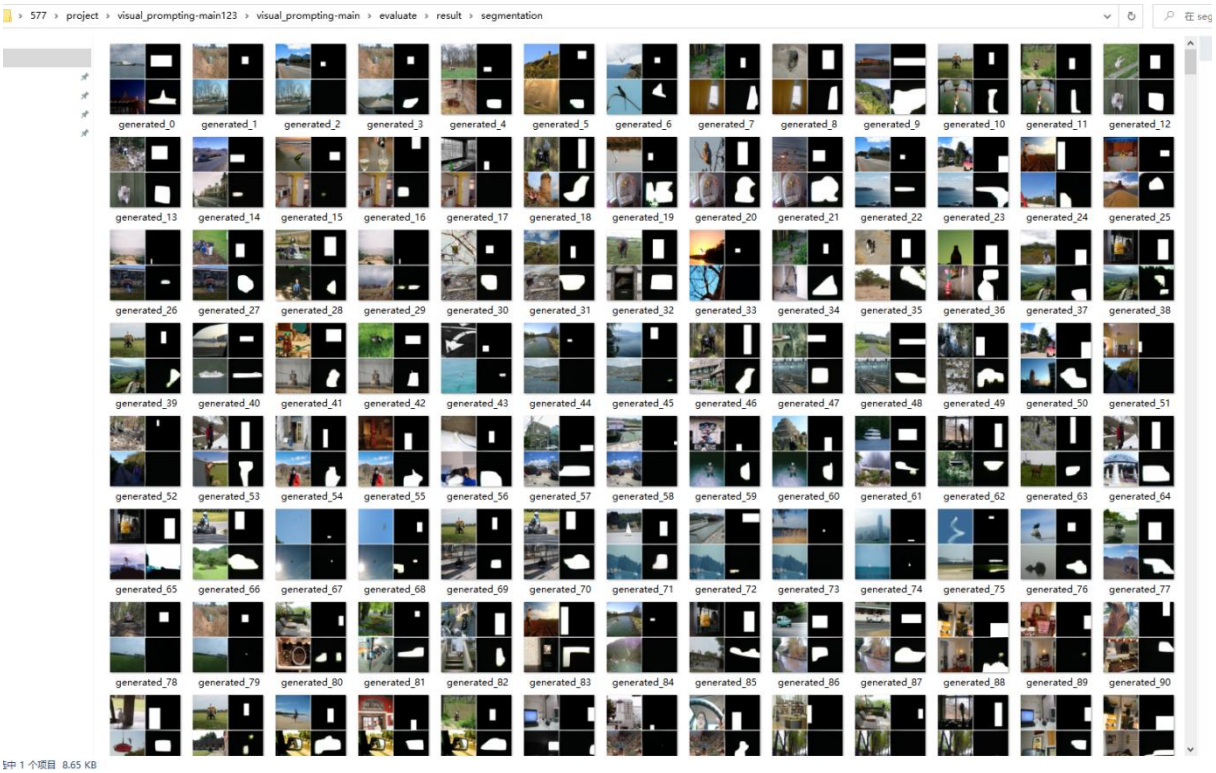

Figure 7: Results

5.2Foreground Segmentation

Namespace(model='mae_vit_large_patch16',output_dir='C:/Users/70241/PycharmProjects/visual_prompting-main/evaluate/result/ForegroundSegm', device='cuda',

base_dir='C:/Users/70241/PycharmProjects/visual_prompting-main/figures_dataset', seed=0, t=[0, 0, 0], task='segmentation', ckpt='C:/Users/70241/PycharmProjects/visual_prompting-main/output_dir/checkpoint-1000.pth', dataset_type='**pascal**', split=0, purple=0, flip=0)

A total of 1000 images are produced, and the accuracy is

{'iou': 0.2750272090546491, 'color_blind_iou': 0.2750272090546491, 'accuracy': 0.8513287882477079}

```
0   {'iou': 0.65627295, 'color_blind_iou': 0.65627295, 'accuracy': 0.923950978005032}
1   {'iou': 0.2633904, 'color_blind_iou': 0.2633904, 'accuracy': 0.8917295674052431}
2   {'iou': 0.0011587485, 'color_blind_iou': 0.0011587485, 'accuracy': 0.7901144387630874}
3   {'iou': 0.0, 'color_blind_iou': 0.0, 'accuracy': 0.8537456375294213}
4   {'iou': 0.0, 'color_blind_iou': 0.0, 'accuracy': 0.8496063631198766}
5   {'iou': 0.0, 'color_blind_iou': 0.0, 'accuracy': 0.6140735329924519}
6   {'iou': 0.030837005, 'color_blind_iou': 0.030837005, 'accuracy': 0.9464329194058924}
7   {'iou': 0.060761113, 'color_blind_iou': 0.060761113, 'accuracy': 0.7616264913562211}
8   {'iou': 0.5519031, 'color_blind_iou': 0.5519031, 'accuracy': 0.9369369369369369}
9   {'iou': 0.028169014, 'color_blind_iou': 0.028169014, 'accuracy': 0.9495982468955442}
10  {'iou': 0.04689655, 'color_blind_iou': 0.04689655, 'accuracy': 0.8878337797256716}
11  {'iou': 0.0, 'color_blind_iou': 0.0, 'accuracy': 0.936612879636393}
12  {'iou': 0.5718194, 'color_blind_iou': 0.5718194, 'accuracy': 0.8729810891973054}
13  {'iou': 0.046430487, 'color_blind_iou': 0.046430487, 'accuracy': 0.7116305494683873}
14  {'iou': 0.7580916, 'color_blind_iou': 0.7580916, 'accuracy': 0.9065822579336092}
15  {'iou': 0.14211288, 'color_blind_iou': 0.14211288, 'accuracy': 0.7594351107864621}
16  {'iou': 0.057806324, 'color_blind_iou': 0.057806324, 'accuracy': 0.8452236019803587}
17  {'iou': 0.6090037, 'color_blind_iou': 0.6090037, 'accuracy': 0.9238698157617077}
18  {'iou': 0.49263874, 'color_blind_iou': 0.49263874, 'accuracy': 0.8545572599626654}
19  {'iou': 0.0, 'color_blind_iou': 0.0, 'accuracy': 0.31912994075156237}
20  {'iou': 0.64719677, 'color_blind_iou': 0.64719677, 'accuracy': 0.8656764872981089}
21  {'iou': 0.68917775, 'color_blind_iou': 0.68917775, 'accuracy': 0.910721532343154}
22  {'iou': 0.034183398, 'color_blind_iou': 0.034183398, 'accuracy': 0.7110624137651165}
23  {'iou': 0.20867209, 'color_blind_iou': 0.20867209, 'accuracy': 0.9763006249492736}
24  {'iou': 0.049483012, 'color_blind_iou': 0.049483012, 'accuracy': 0.8955441928414901}
25  {'iou': 0.5921916, 'color_blind_iou': 0.5921916, 'accuracy': 0.899115331547764}
26  {'iou': 0.16157989, 'color_blind_iou': 0.16157989, 'accuracy': 0.9241944647350053}
27  {'iou': 0.054607507, 'color_blind_iou': 0.054607507, 'accuracy': 0.9775180585991396}
28  {'iou': 0.070584476, 'color_blind_iou': 0.070584476, 'accuracy': 0.76381787192598}
29  {'iou': 0.47593096, 'color_blind_iou': 0.47593096, 'accuracy': 0.906338771203636}
```

Figure 8: ACC

Results:

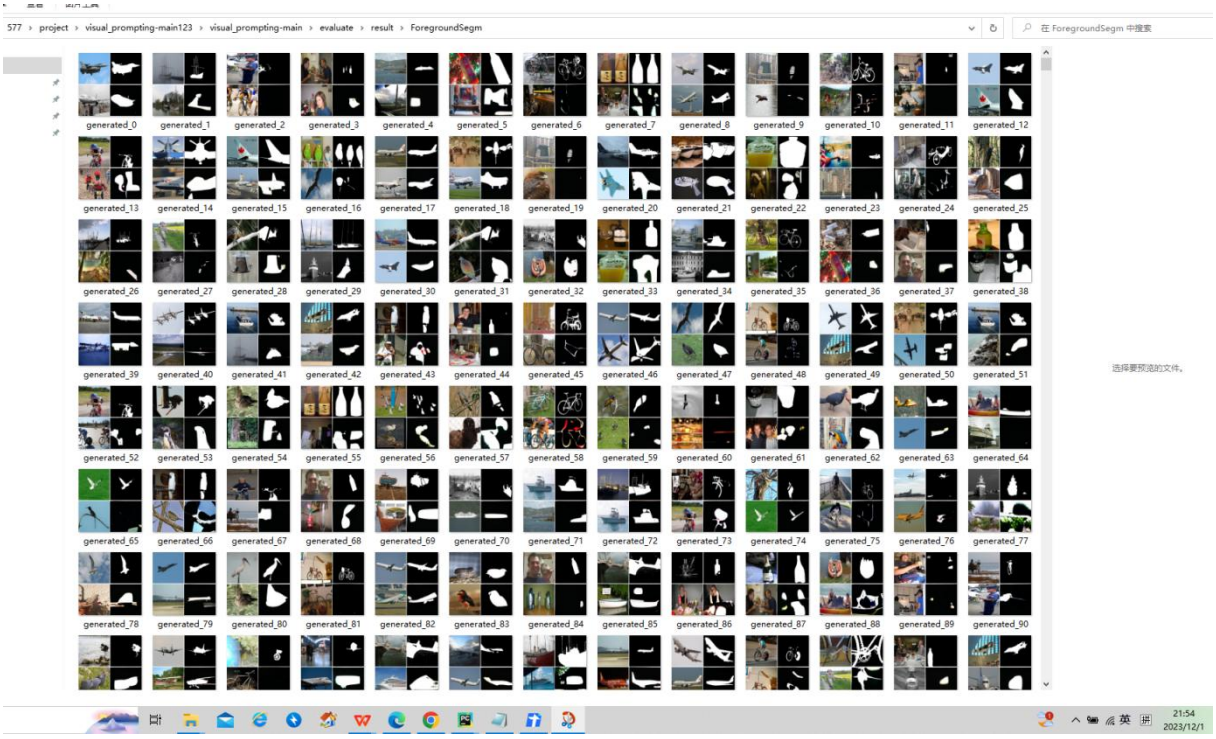

Figure 9: Results

5.3Colorization.

Namespace(model='mae_vit_large_patch16', output_dir='C:/Users/70241/PycharmProjects/visual_prompting-main/evaluate/result/', data_path='C:/Users/70241/PycharmProjects/visual_prompting-main/figures_dataset/**imagenet**', device='cuda', seed=0,

tta_option=0, ckpt='C:/Users/70241/PycharmProjects/visual_prompting-main/output_dir/checkpoint-1000.pth', autoregressive=False)

MSE is a commonly used performance metric for regression problems, which measures the square of the average error between the predicted value and the true value. The smaller the MSE, the higher the accuracy of the prediction model.

A total of 1000 images are produced, and the mse is all{'mse': 0.6642630747678648}

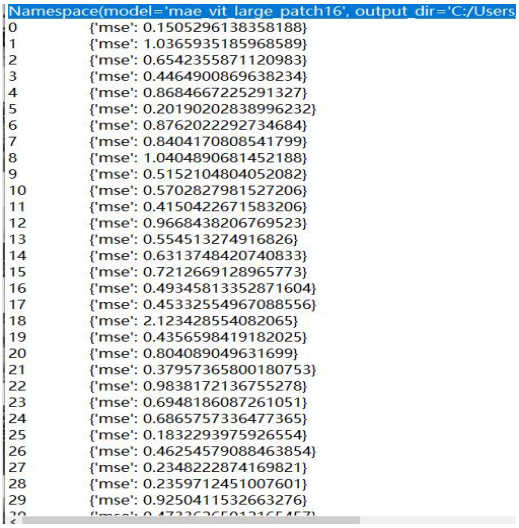

Figure 10: mse

Results:

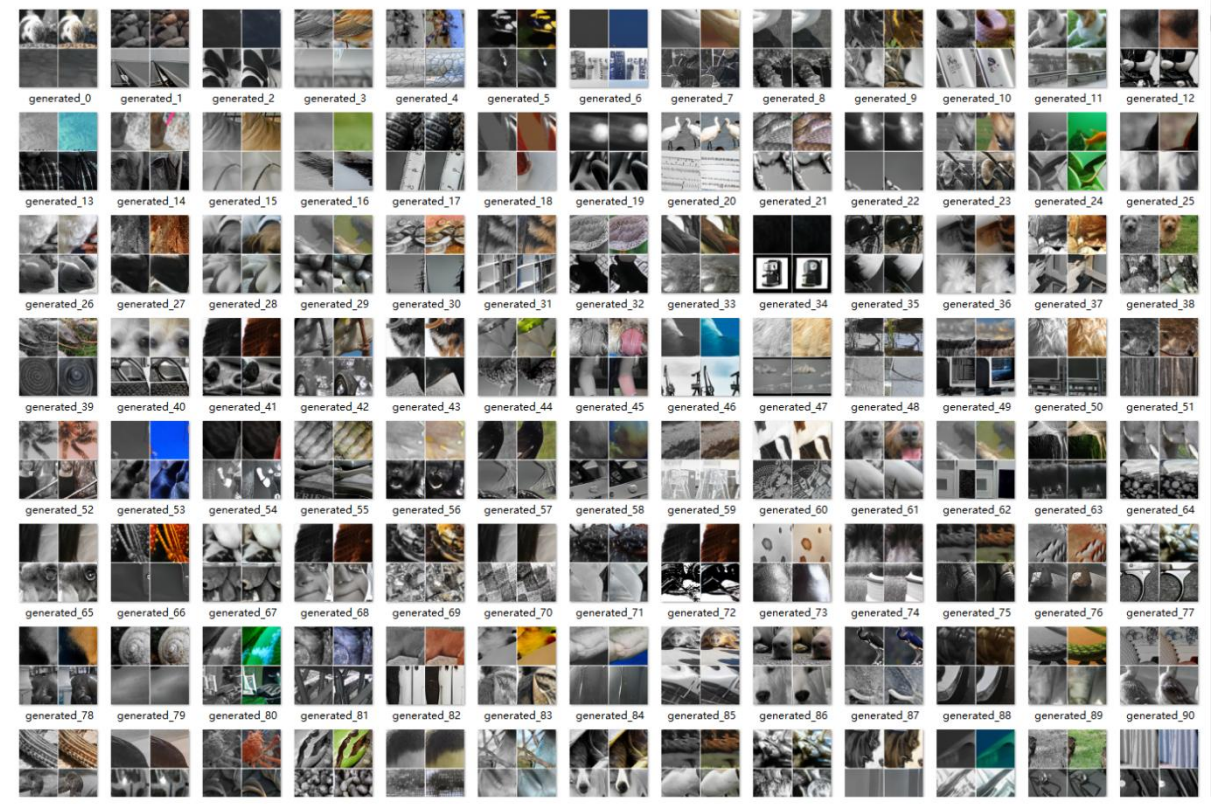

Figure 11: Results

5.4reasoning

Namespace(model='mae_vit_large_patch16',

output_dir='C:/Users/70241/PycharmProjects/visual_prompting-main/evaluate/result/reasoning/', device='cuda', seed=0, tta_option=0, ckpt='C:/Users/70241/PycharmProjects/visual_prompting-main/output_dir/checkpoint-1000.pth', dataset_type='color')

Color change prediction Results:

Each example pair contains an image of a circle appearing in the same location, with the color changed from green to blue. Given a new image query, the goal is to predict the corresponding image with the circle colored in blue.

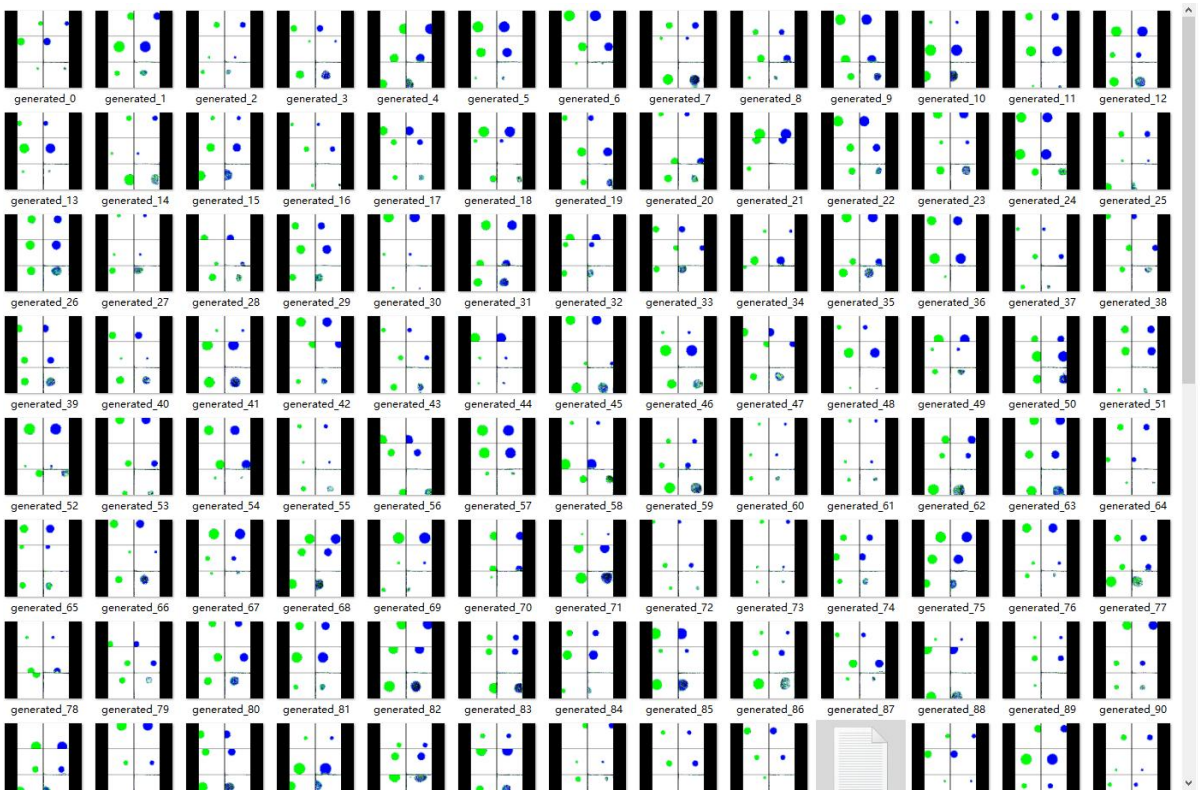

Figure 12: Results

6.summarize

It can be seen from the results of model running that the accuracy of single object detection is higher for the four downstream tasks. Due to limited time, we only run the single code MAE-VQgan at present, instead of BEIT, VQGAN and MAE for comparison among different downstream tasks.

Our code is mainly referred to the paper <Visual Prompting via Image Inpainting>， address: [2209.00647] Visual Prompting via Image Inpainting (arxiv.org), code mainly from: GitHub - amirbar/visual_prompting: Official implementation and data release of the paper "Visual Prompting via Image Inpainting".

Code running environment: python3.11

All you need to do for training is run main_pretrain.py. The dataset is in the folder figures_dataset, where the dataset is the data in its instance

There are a total of four evaluations on github, and the results are evaluated /result， checkpoint is saved in the output_dir folder

The four evaluation codes have been run, using the model pre-trained on github, checkpoint-1000.pth, the result is evaluated /result, log.txt is also in it.

Import package as follows:：

```
import argparse
import datetime
import numpy as np
import os
import time
from pathlib import Path

import torch
import torch.backends.cudnn as cudnn
from torch.utils.tensorboard import SummaryWriter
import torchvision.transforms as transforms
from torchvision import datasets
import timm.optim.optim_factory as optim_factory
import util.misc as misc
from util.misc import NativeScalerWithGradNormCount as NativeScaler
import models_mae
from engine_pretrain import train_one_epoch
```

Adjustable parameter variables are as follows：

```
parser = argparse.ArgumentParser('MAE pre-training', add_help=False)
parser.add_argument('--save_ckpt_freq', default=100, type=int)
parser.add_argument('--batch_size', default=64, type=int,
                    help='Batch size per GPU (effective batch size is batch_size * accum_iter * # gpus')
parser.add_argument('--epochs', default=500, type=int)
parser.add_argument('--accum_iter', default=1, type=int,
                    help='Accumulate gradient iterations (for increasing the effective batch size under memory constraints)')
# Model parameters
parser.add_argument('--model', default='mae_vit_large_patch16', type=str, metavar='MODEL',
                    help='Name of model to train')
parser.add_argument('--input_size', default=224, type=int,
                    help='images input size')
parser.add_argument('--mask_ratio', default=0.75, type=float,
                    help='Masking ratio (percentage of removed patches).')
# Optimizer parameters
parser.add_argument('--weight_decay', type=float, default=0.05,
                    help='weight decay (default: 0.05)')
parser.add_argument('--lr', type=float, default=None, metavar='LR',
                    help='learning rate (absolute lr)')
parser.add_argument('--blr', type=float, default=1e-3, metavar='LR',
                    help='base learning rate: absolute_lr = base_lr * total_batch_size / 256')
parser.add_argument('--min_lr', type=float, default=0., metavar='LR',
                    help='lower lr bound for cyclic schedulers that hit 0')

parser.add_argument('--warmup_epochs', type=int, default=40, metavar='N',
                    help='epochs to warmup LR')

parser.add_argument('--break_after_epoch', type=int, metavar='N', help='break training after X epochs, to tune
hyperparams and avoid messing with training schedule')
```

detailed code：visual_prompting-main

detailed training results：\visual_prompting-main\evaluate\result

7.The main contribution:

The main contributions of the project can be summarized as follows:

Environment Setup: The project recommends using a stable version of Python 3.11 for running the code.

Code Instructions:

Training: The training process involves running the main_pretrain.py file. The dataset is located in the figures_dataset folder. To modify the training data, you can change the path in line 69 of main_pretrain.py. The CIFAR dataset used is CIFAR-10.

Evaluation: There are four evaluation scripts available on GitHub. The evaluation results can be found in the evaluate/result directory, and the checkpoints are saved in the output_dir folder.

Code Modifications:

In util/misc.py, change 'from torch._six import inf' to 'from torch import inf': torch._six is an old usage, and the new version requires importing inf directly from the torch module.

In util/pos_embed.py, in line 56, change 'dtype=np.float' to 'dtype=np.float64': There is a difference in usage between versions, and the new version requires np.float64 as the data type.

In evaluate/reasoning_dataloader.py, in line 59, change 'res_img=torch.tensor(output)[nn_indices]' to 'res_img = outputs.clone().detach()[nn_indices]': This modification creates a completely independent tensor by using .clone().detach() to avoid sharing the data and computation history of the original tensor.

In evaluate/evaluate_reasoning.py, in line 96, the code limits color to be a 2x3 matrix, representing white and target colors. However, after examining the code details:

np.reshape(target,(-1,3)) reshapes the input image target from an MxNx3 image matrix to a Kx3 matrix, where K = M*N, and each row represents the RGB values of a pixel.

np.unique(... , axis=0) finds the unique rows in this Kx3 matrix, meaning if there are duplicate colors, only one will be kept. Therefore, the final output size of color should be Xx3, where X depends on the number of different colors in the target image.

As a result, we did the modification as below:

Directly modify line 96 as follows:

Change 'assert colors.shape[0] == 2, colors' to 'assert colors.shape[0] == 3, colors', and the program will run successfully.

If not modified, an error will be thrown: AssertionError:

[[0 0 255]

[0 255 0]

[255 255 255]]

Additional Information:

The prep_cifar.sh script contains a program for processing the CIFAR dataset, converting it into image format for further use.

To separate the results and facilitate storage, a new evaluate_Foregroundsegm.py file was created for foreground segmentation, while object detection uses the same evaluation script.

8. What We learn from the lesson

In this project, the researchers explored the concept of visual prompting, inspired by the idea of prompting in natural language processing (NLP). The goal was to adapt a pre-trained visual model to novel downstream tasks without task-specific fine-tuning or any model modification.

The researchers investigated the effectiveness of visual prompting by using input-output image examples of a new task at test time, along with a new input image. The objective was to automatically generate the correct output image that is consistent with the given task examples. They found that formulating this problem as a simple image inpainting task, where a hole in a concatenated visual prompt image is filled, yielded surprisingly effective results. This was possible because the inpainting algorithm had been trained on the appropriate data.

To support their approach, the researchers curated a new dataset CIFAR10. They trained masked auto-encoding models on this dataset and applied visual prompting to these pre-trained models. They demonstrated the effectiveness of visual prompting on various downstream tasks, including foreground segmentation, single object detection, colorization, edge detection, and more.

In summary, the project showed that visual prompting, using a simple image inpainting formulation, can effectively adapt pre-trained models to novel tasks without the need for task-specific fine-tuning or model modification. The researchers obtained promising results by leveraging a curated dataset and applying visual prompting to pre-trained masked auto-encoding models.

9. Future work

Based on the given paper abstract, here is a list of potential future work that could be explored:

Extensions to other computer vision tasks: The paper demonstrates the effectiveness of visual prompting on tasks like foreground segmentation, single object detection, colorization, and edge detection. Further exploration can be done to apply visual prompting to a wider range of computer vision tasks, such as image classification, semantic segmentation, or instance segmentation.

Investigating different inpainting algorithms: The paper shows that the performance of visual prompting heavily relies on the quality of the inpainting algorithm. Future work could involve exploring and experimenting with different inpainting algorithms to improve the overall effectiveness of visual prompting.

Data augmentation and generalization: The researchers trained their models on a curated dataset of unlabeled figures from academic papers. Further investigation can be done on augmenting and expanding the dataset to include more diverse and representative images from various sources. This could enhance the generalization capabilities of the visual prompting approach.

Combining visual and textual prompts: Inspired by prompting in NLP, future work could explore the integration of visual and textual prompts to enhance the performance of visual prompting. This could involve leveraging natural language descriptions or captions associated with the images to provide additional context and guidance for the inpainting process.

Investigating transfer learning and domain adaptation: It would be interesting to explore the transferability of pre-trained visual models and investigate techniques for adapting them to novel downstream tasks in different domains. This could involve domain adaptation methods or leveraging pre-trained models from related domains.

Exploring unsupervised or weakly supervised training: The paper focuses on training the models using a curated dataset of unlabeled images. Future work could explore unsupervised or weakly supervised training approaches for visual prompting, reducing the reliance on a large amount of labeled data.

Addressing limitations and failure cases: It is important to analyze and understand the limitations and failure cases of visual prompting. Future work could involve studying scenarios where visual prompting may not perform well and developing techniques to address those limitations, such as handling complex or ambiguous inpainting scenarios.

Real-world applications and user studies: Further exploration can be done to apply visual prompting in real-world applications and evaluate its effectiveness through user studies. This would involve assessing the usability, user satisfaction, and performance of visual prompting in practical scenarios.

These potential future directions can help advance the understanding and application of visual prompting, improving its versatility and effectiveness in adapting pre-trained visual models to novel downstream tasks.

References

[1]our github repository link: https://github.com/GCCCompiler/CS577-Project

[2]通过图像修复的视觉提示_visual prompting via image inpainting_前世忘语的博客-CSDN 博客

[3]https://blog.csdn.net/moxibingdao/article/details/121586379?share_token=510A82CE-4BC9-447F-9A19-CA3AB995D396&tt_from=weixin_moments&utm_source=weixin_moments&utm_medium=toutiao_ios&utm_campaign=client_share&wxshare_count=1

[4]Masked AutoEncoders（MAE) Top-1 准确率 87.8% - AI 社区 (xfyun.cn)

[5]https://github.com/amirbar/visual_prompting

[6]https://arxiv.org/abs/2209.00647

[7]Masked AutoEncoders（MAE) Top-1 准确率 87.8% - 飞桨 AI Studio 星河社区 (baidu.com)

GitHub - ewrfcas/MAE-FAR: Codes of Learning Prior Feature and Attention Enhanced Image Inpainting (ECCV2022)

[8] Chen, M., Radford, A., Child, R., Wu, J., Jun, H., Luan, D., Sutskever, I.: Generative pretraining from pixels. In: International Conference on Machine Learning. pp. 1691–1703. PMLR (2020)

[9] Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. arXiv preprint arXiv:2002.05709 (2020) Chen, X., Fan, H., Girshick, R., He, K.: Improved baselines

[10]PASCAL VOC 2007 数据集的简单介绍_pascal voc2007 数据集-CSDN 博客