Python编程与人工智能实践

Sprouts in the shape of text 'Imagen' coming out of a fairytale book.

A photo of a Shiba Inu dog with a backpack riding a bike. It is wearing sunglasses and a beach hat.

应用篇：AI绘图

Stable Diffusion （稳态扩散模型）

于泓
鲁东大学
信息与电气工程学院

2022.10.9

AI绘图（文本-图像生成）

文献：“Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding ”

Sprouts in the shape of text ‘Imagen’ coming out of a fairytale book.


A photo of a Shiba Inu dog with a backpack riding a bike. It is wearing sunglasses and a beach hat.

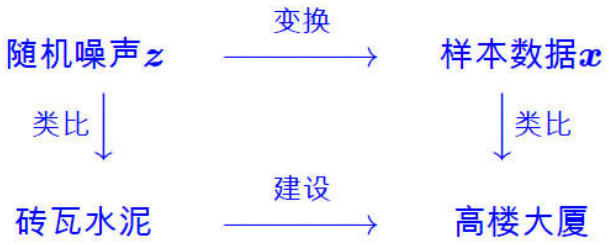

A high contrast portrait of a very happy fuzzy panda dressed as a chef in a high end kitchen making dough. There is a painting of flowers on the wall behind him


Teddy bears swimming at the Olympics 400m Butterfly event.


A cute corgi lives in a house made out of sushi.


A cute sloth holding a small treasure chest. A bright golden glow is coming from the chest.

基础模型（DDPM Denoising Diffusion Probabilistic Model）



普通的生成模型： GAN，VAE
直接利用随机噪声生成图像样本

$x=x_0 \rightarrow x_1 \rightarrow x_2 \rightarrow \cdots\cdots \rightarrow x_{T-1} \rightarrow x_T = z$

DDPM 逐步分解，(**加噪声**)

$z=x_T \rightarrow x_{T-1} \rightarrow \cdots \rightarrow x_2 \rightarrow x_1 \rightarrow x_0 = x$

再逐步重构 （**去噪声**）

$X=X_0 \rightarrow X_1 \rightarrow X_2 \rightarrow \cdots\cdots \rightarrow X_{T-1} \rightarrow X_T = Z$

$Z=X_T \rightarrow X_{T-1} \rightarrow \cdots \rightarrow X_2 \rightarrow X_1 \rightarrow X_0 = X$

训练：构造神经网络，学习加入的噪声

$$\frac{\beta_t^2}{\alpha_t^2}\left\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)\right\|^2$$

损失权重　　神经网络 UNET　　噪声强度

加噪声的公式：x_{t-1}递推 x_t

$$\boldsymbol{x}_t = \alpha_t \boldsymbol{x}_{t-1} + \beta_t \boldsymbol{\epsilon}_t, \quad \boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$$

$$\alpha_t^2 + \beta_t^2 = 1$$

随着t的增加不断变大

直接从x_0递推x_t

$$\boldsymbol{x}_t = \underbrace{(\alpha_t \cdots \alpha_1)}_{\text{记为}\bar{\alpha}_t}\boldsymbol{x}_0 + \underbrace{\sqrt{1-(\alpha_t \cdots \alpha_1)^2}}_{\text{记为}\bar{\beta}_t}\bar{\boldsymbol{\epsilon}}_t, \quad \bar{\boldsymbol{\epsilon}}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$$

Algorithm 1 Training

1: **repeat**
2: $\mathrm{x}_0 \sim q(\mathrm{x}_0)$
3: $t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathrm{I})$
5: Take gradient descent step on
$\nabla_\theta \left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathrm{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2$
6: **until** converged

重构过程:

$Z=X_T \rightarrow X_{T-1} \rightarrow \dots \rightarrow X_2 \rightarrow X_1 \rightarrow X_0 = X$

$$\boldsymbol{x}_t = \alpha_t \boldsymbol{x}_{t-1} + \beta_t \boldsymbol{\epsilon}_t,$$

$$\boldsymbol{x}_{t-1} = \frac{1}{\alpha_t}(\boldsymbol{x}_t - \beta_t \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)) \quad \text{直接重构}$$

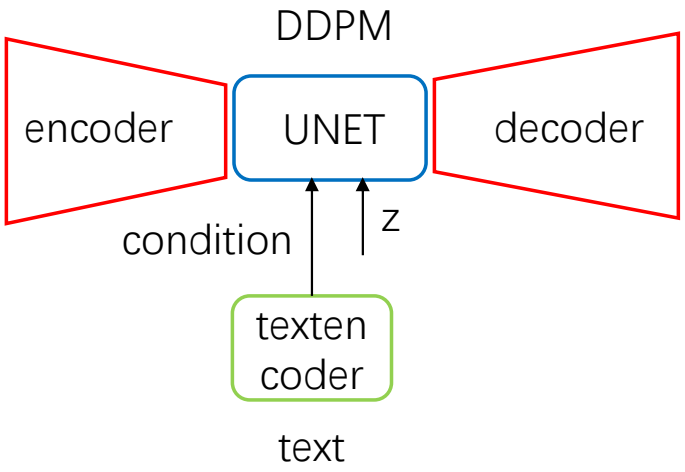
$$\boldsymbol{x}_{t-1} = \frac{1}{\alpha_t}(\boldsymbol{x}_t - \beta_t \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)) + \sigma_t \boldsymbol{z}, \quad \boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$$

与 β_t 同步

控制主要方向

实际应用中需要加入一些随机性

Stable Diffusion 模型结构



区别 （1）引入了encoder 和decoder
对输入的图形进行降维，在潜空间（latents）上进行 DDPM

(2) 在噪声估计时引入了文本作为条件c

噪声估计

$$\tilde{\epsilon}_\theta(\mathbf{z}_t, \mathbf{c}) = w\epsilon_\theta(\mathbf{z}_t, \mathbf{c}) + (1-w)\epsilon_\theta(\mathbf{z}_t).$$

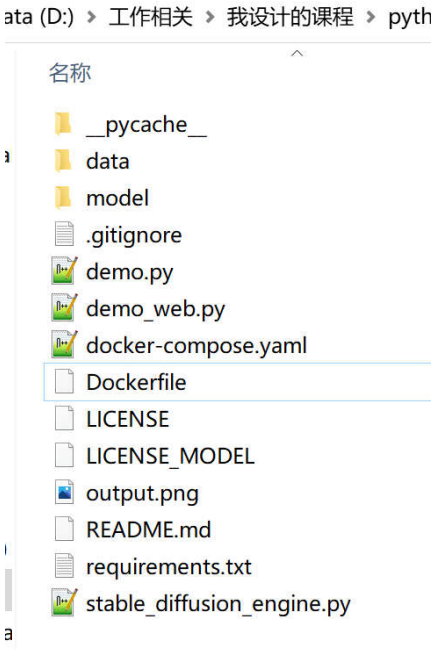
本地部署 (openvino版)

代码： https://github.com/bes-dev/stable_diffusion.openvino

下载代码：

安装必要的包

pip install –r requirements.txt



下载模型：

https://huggingface.co/bes-dev/stable-diffusion-v1-4-openvino/tree/main

代码:

```
# -- coding: utf-8 --`
import argparse
import os
# engine
from stable_diffusion_engine import StableDiffusionEngine
# scheduler
from diffusers import LMSDiscreteScheduler, PNDMScheduler
# utils
import cv2
import numpy as np
import time
```

信号重构

$$x_{t-1} = \frac{1}{\alpha_t}(x_t - \beta_t \epsilon_\theta(x_t, t)) + \sigma_t z, \quad z \sim \mathcal{N}(0,$$

β_t

生成

```
def main(args):
    if args.seed is not None:
        np.random.seed(args.seed)
    scheduler = LMSDiscreteScheduler(
            beta_start=args.beta_start,
            beta_end=args.beta_end,
            beta_schedule=args.beta_schedule,
            tensor_format="np")

    engine = StableDiffusionEngine(
        model = args.model,
        scheduler = scheduler,
        tokenizer = args.tokenizer
    )
    image = engine(
        prompt = args.prompt,
        num_inference_steps = args.num_inference_steps,
        guidance_scale = args.guidance_scale,
        eta = args.eta
    )
    cv2.imwrite(args.output, image)
```

随机种子

$$x_t = \alpha_t x_{t-1} + \beta_t \varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$$

噪声权重的范围

神经网络模型

β_t

文本转特征

文本

重构步数

文本指导力

定义text-to-image 引擎

$$\tilde{\epsilon}_\theta(\mathbf{z}_t, \mathbf{c}) = w\epsilon_\theta(\mathbf{z}_t, \mathbf{c}) + (1 - w)\epsilon_\theta(\mathbf{z}_t).$$

```python
if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    # pipeline configure
    parser.add_argument("--model", type=str, default="bes-dev/stable-diffusion-v1-4-openvino", help="model name")
    # randomizer params
    parser.add_argument("--seed", type=int, default=None, help="random seed for generating consistent images per pr
    # scheduler params
    parser.add_argument("--beta-start", type=float, default=0.00085, help="LMSDiscreteScheduler::beta_start")
    parser.add_argument("--beta-end", type=float, default=0.012, help="LMSDiscreteScheduler::beta_end")
    parser.add_argument("--beta-schedule", type=str, default="scaled_linear", help="LMSDiscreteScheduler::beta_sche
    # diffusion params
    parser.add_argument("--num-inference-steps", type=int, default=32, help="num inference steps")
    parser.add_argument("--guidance-scale", type=float, default=7, help="guidance scale")
    parser.add_argument("--eta", type=float, default=0.0, help="eta")
    # tokenizer
    parser.add_argument("--tokenizer", type=str, default="openai/clip-vit-large-patch14", help="tokenizer")
    # prompt
    parser.add_argument("--prompt", type=str, default="ultra nekopara fantastically detailed reflecting eyes modern
    # output name
    parser.add_argument("--output", type=str, default="output.png", help="output image name")
    args = parser.parse_args()
    main(args)

    img = cv2.imread('output.png')
    cv2.imshow("win",img)
    cv2.waitKey(0)
```

stable_diffusion_engine.py


```python
from openvino.runtime import Core
# tokenizer
from transformers import CLIPTokenizer
# utils
from tqdm import tqdm
from diffusers import LMSDiscreteScheduler,
import os

def result(var):
    return next(iter(var.values()))


class StableDiffusionEngine:
    def __init__(
            self,
            scheduler,
            model="bes-dev/stable-diffusion-v1-4-openvino",
            tokenizer="openai/clip-vit-large-patch14",
            device="CPU"
    ):
        self.tokenizer = CLIPTokenizer.from_pretrained(tokenizer)
        self.scheduler = scheduler
        # models
        self.core = Core()
        # text features
        self._text_encoder = self.core.read_model(
              os.path.join("model","text_encoder.xml"),
              os.path.join("model","text_encoder.bin")
        )
        self.text_encoder = self.core.compile_model(self._text_encoder, device)
        # diffusion
        self._unet = self.core.read_model(
            os.path.join("model","unet.xml"),
            os.path.join("model","unet.bin")

        )
        self.unet = self.core.compile_model(self._unet, device)
        self.latent_shape = tuple(self._unet.inputs[0].shape)[1:]
        # decoder
        self._vae_decoder = self.core.read_model(
            os.path.join("model","vae_decoder.xml"),
            os.path.join("model","vae_decoder.bin")

        )
        self.vae_decoder = self.core.compile_model(self._vae_decoder, device)
```

初始化 加载模型

提取文本特征

```
def __call__(
        self,
        prompt,
        num_inference_steps = 32,
        guidance_scale = 7.5,
        eta = 0.0
):
    # extract condition
    tokens = self.tokenizer(
        prompt,
        padding="max_length",
        max_length=self.tokenizer.model_max_length,
        truncation=True
    ).input_ids
    text_embeddings = result(
        self.text_encoder.infer_new_request({"tokens": np.array([tokens])})
    )

    # do classifier free guidance
    if guidance_scale > 1.0:
        tokens_uncond = self.tokenizer(
            "",
            padding="max_length",
            max_length=self.tokenizer.model_max_length,
            truncation=True
        ).input_ids
        uncond_embeddings = result(
            self.text_encoder.infer_new_request({"tokens": np.array([tokens_uncond])})
        )
        text_embeddings = np.concatenate((uncond_embeddings, text_embeddings), axis=0)
```

$$\tilde{\epsilon}_\theta(\mathbf{z}_t, \mathbf{c}) = w\epsilon_\theta(\mathbf{z}_t, \mathbf{c}) + (1-w)\epsilon_\theta(\mathbf{z}_t).$$

```
# set timesteps
accepts_offset = "offset" in set(inspect.signature(self.scheduler.set_timesteps).parameters.keys())
extra_set_kwargs = {}
offset = 0
if accepts_offset:
    offset = 1
    extra_set_kwargs["offset"] = 1

self.scheduler.set_timesteps(num_inference_steps, **extra_set_kwargs)


latents = np.random.randn(*self.latent_shape)
init_timestep = num_inference_steps


# if we use LMSDiscreteScheduler, let's make sure latents are mulitplied by sigmas
if isinstance(self.scheduler, LMSDiscreteScheduler):
    latents = latents * self.scheduler.sigmas[0]

# prepare extra kwargs for the scheduler step, since not all schedulers have the same signature
# eta (η) is only used with the DDIMScheduler, it will be ignored for other schedulers.
# eta corresponds to η in DDIM paper: https://arxiv.org/abs/2010.02502
# and should be between [0, 1]
accepts_eta = "eta" in set(inspect.signature(self.scheduler.step).parameters.keys())
extra_step_kwargs = {}
if accepts_eta:
    extra_step_kwargs["eta"] = eta
```

β_t 相关的一些初始化

```
t_start = max(num_inference_steps - init_timestep + offset, 0)
for i, t in tqdm(enumerate(self.scheduler.timesteps[t_start:])):
    # expand the latents if we are doing classifier free guidance
    latent_model_input = np.stack([latents, latents], 0) if guidance_scale > 1.0 else latents[None]
    if isinstance(self.scheduler, LMSDiscreteScheduler):
        sigma = self.scheduler.sigmas[i]
        latent_model_input = latent_model_input / ((sigma**2 + 1) ** 0.5)

    # predict the noise residual
    noise_pred = result(self.unet.infer_new_request({
        "latent_model_input": latent_model_input,
        "t": t,
        "encoder_hidden_states": text_embeddings
    }))

    # perform guidance
    if guidance_scale > 1.0:
        noise_pred = noise_pred[0] + guidance_scale * (noise_pred[1] - noise_pred[0])

    # compute the previous noisy sample x_t -> x_t-1
    if isinstance(self.scheduler, LMSDiscreteScheduler):
        latents = self.scheduler.step(noise_pred, i, latents, **extra_step_kwargs)["prev_sample"]
    else:
        latents = self.scheduler.step(noise_pred, t, latents, **extra_step_kwargs)["prev_sample"]

image = result(self.vae_decoder.infer_new_request({
    "latents": np.expand_dims(latents, 0)
}))
```

$$\tilde{\epsilon}_\theta(\mathbf{z}_t, \mathbf{c}) = w\epsilon_\theta(\mathbf{z}_t, \mathbf{c}) + (1-w)\epsilon_\theta(\mathbf{z}_t).$$

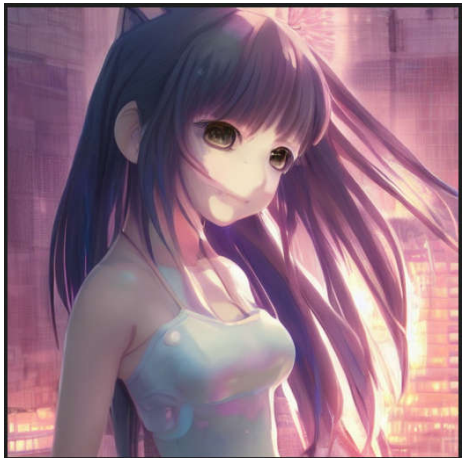
噪声估计

一步重构

解码生成图片

一些结果

ultra nekopara fantastically detailed reflecting eyes modern anime style art cute detailed ears cat girl neko dress portrait shinkai makoto vibrant Studio ghibli kyoto animation hideaki anno Sakimichan Stanley Artgerm Lau Rossdraws James Jean Marc Simonetti elegant highly detailed digital painting artstation pixiv cyberpunk



a hyperdetailed matte painting of a german romantic tree emerging from an oceanographic landscape, magic realism painting, trending on artstation

The moon is high in the sky,National Day,Happy Valentine's Day,oil painting,the style of Monet

stable_diffusion.openvino （bug 修改）

stable_diffusion_engine.py

```
184             t = float(t)
185             # predict the noise residual
186             noise_pred = result(self.unet.infer_new_request({
187                 "latent_model_input": latent_model_input,
188                 "t": t,
189                 "encoder_hidden_states": text_embeddings
190             }))
```

需要找寻自己的Python安装位置

D:\python\python38\lib\site-packages\diffusers\schedulers\scheduling_pndm.py

230行

alpha_prod_t = self.alphas_cumprod[int(timestep) + 1 - self._offset]

370行

timesteps = torch.from_numpy(timesteps)
# timesteps = timesteps.to(self.alphas_cumprod.device)
timesteps = timesteps.to('cpu')

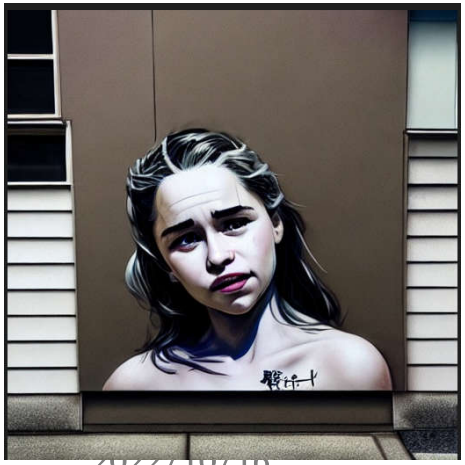
Windows系统需要修改
Linux系统不用

Img2img （图转图）

输入

输出

Street-art painting of Emilia Clarke in style of Banksy, photorealism



Street-art painting of Emilia Clarke, photorealism,portrait shinkai makoto vibrant Studio ghibli kyoto animation hideaki anno Sakimichan Stanley Artgerm Lau Rossdraws James Jean Marc Simonetti elegant highly detailed digital painting artstation pixiv cyberpunk

sketch effect,Street-art painting of Emilia Clarke"

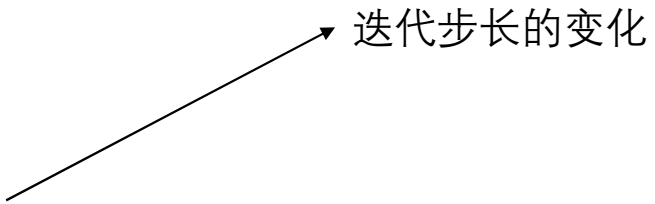
```python
def _preprocess_image(self, image):
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    h, w = image.shape[1:]
    if h != self.init_image_shape[0] and w != self.init_image_shape[1]:
        image = cv2.resize(
            image,
            (self.init_image_shape[1], self.init_image_shape[0]),
            interpolation=cv2.INTER_LANCZOS4
        )
    # normalize
    image = image.astype(np.float32) / 255.0
    image = 2.0 * image - 1.0
    # to batch
    image = image[None].transpose(0, 3, 1, 2)
    return image
```

图像初始化

```python
def _encode_image(self, init_image):
    moments = result(self.vae_encoder.infer_new_request({
        "init_image": self._preprocess_image(init_image)
    }))
    mean, logvar = np.split(moments, 2, axis=1)
    std = np.exp(logvar * 0.5)
    latent = (mean + std * np.random.randn(*mean.shape)) * 0.18215
    return latent
```

图像编码，变为64*64

```
# initialize latent latent
if init_image is None:
    latents = np.random.randn(*self.latent_shape)
    init_timestep = num_inference_steps
else:
    init_latents = self._encode_image(init_image)
    init_timestep = int(num_inference_steps * strength) + offset
    init_timestep = min(init_timestep, num_inference_steps)
    timesteps = np.array([[self.scheduler.timesteps[-init_timestep]]]).astype(np.long)
    noise = np.random.randn(*self.latent_shape)
    latents = self.scheduler.add_noise(init_latents, noise, timesteps)[0]
```



test.py

```
prompt = "sketch effect,Street-art painting of Emilia Clarke"
# 输出文件的名称
file_output = "out4.jpg"

# 初始化图片路径
init_image = "data/input.png"

# 初始化图片的强度 0-1之间 越小迭代的步数越少，初始化图片的变化也越小
strength = 0.7

# mask图片
maks_image= None

# 是否需要 mask 处理
b_mask = False

# 如需要mask处理又没有mask图像 则从 初始化图片图片中 生成一个
if b_mask and  (maks_image is None) and ( init_image is not None):
    file_mask = gen_pic_mask(init_image)
    maks_image = file_mask


# 模型保存地址
path_model = "model"

# seed 随机种子
seed = None
```

```python
# seed 随机种子
seed = None

# scheduler 参数
beta_start = 0.00085
beta_end = 0.012
beta_schedule = "scaled_linear"

# 扩散参数
num_inference_steps = 60
guidance_scale = 8.5
eta = 0.0

# 文本特征提取器
tokenizer = "openai/clip-vit-large-patch14"

# 设置随机种子
np.random.seed(seed)

# 定义 Scheduler
if init_image is None:
    cheduler = LMSDiscreteScheduler(
        beta_start=beta_start,
        beta_end=beta_end,
        beta_schedule=beta_schedule,
        tensor_format="np"
    )
else:
    scheduler = PNDMScheduler(
        beta_start=beta_start,
        beta_end=beta_end,
        beta_schedule=beta_schedule,
        skip_prk_steps = True,
        tensor_format="np"
    )
```

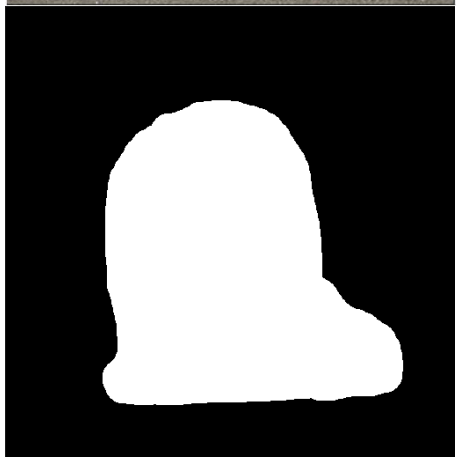
```
# 定义图像推理引擎
print("进行模型加载")
engine = StableDiffusionEngine(
    model = path_model,
    scheduler = scheduler,
    tokenizer = tokenizer
)

# 进行推理
image = engine(
    prompt = prompt,
    init_image = None if init_image is None else cv2.imread(init_image),
    mask = None if maks_image is None else cv2.imread(maks_image, 0),
    strength = strength,
    num_inference_steps = num_inference_steps,
    guidance_scale = guidance_scale,
    eta = eta
)
if init_image is None:
    cv2.imwrite(file_output, image)
else:
    img_input = cv2.imread(init_image)
    h,w,_ = np.shape(img_input)
    img_out = cv2.resize(image,(w,h))
    cv2.imwrite(file_output, img_out)
```

Img2img-mask模式

输入

mask



Street-art painting of Emilia Clarke in style of Banksy, photorealism

sketch effect,Street-art painting of Emilia Clarke"

Street-art painting of Emilia Clarke, photorealism,portrait shinkai makoto vibrant Studio ghibli kyoto animation hideaki anno Sakimichan Stanley Artgerm Lau Rossdraws James Jean Marc Simonetti elegant highly detailed digital painting artstation pixiv cyberpunk

Mask的自动生成（mediapipe）

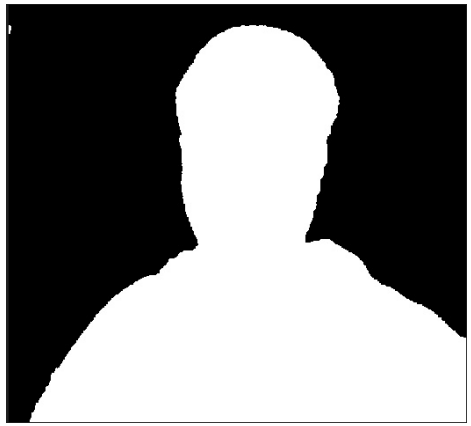
```
os.makedirs("temp",exist_ok=True)
mp_selfie_segmentation = mp.solutions.selfie_segmentation

def gen_pic_mask(file_img_in,BG_COLOR = (0, 0, 0),MASK_COLOR = (255, 255, 255)):

    with mp_selfie_segmentation.SelfieSegmentation(model_selection=0) as selfie_segmentation:
        img_in = cv2.imread(file_img_in)
        image_height, image_width, _ = img_in.shape
        # BGR 转 RGB
        results = selfie_segmentation.process(cv2.cvtColor(img_in, cv2.COLOR_BGR2RGB))

        # 前景背景分离
        condition = np.stack((results.segmentation_mask,) * 3, axis=-1) > 0.1
        # Generate solid color images for showing the output selfie segmentation mask.
        fg_image = np.zeros(img_in.shape, dtype=np.uint8)
        fg_image[:] = MASK_COLOR
        bg_image = np.zeros(img_in.shape, dtype=np.uint8)
        bg_image[:] = BG_COLOR
        out_mask = np.where(condition, fg_image, bg_image)

        file_mask_out= "temp/out_mask.jpg"
        cv2.imwrite(file_mask_out,out_mask)
        return file_mask_out
```



代码部分：

```
def _preprocess_mask(self, mask):
    h, w = mask.shape
    if h != self.init_image_shape[0] and w != self.init_image_shape[1]:
        mask = cv2.resize(
            mask,
            (self.init_image_shape[1], self.init_image_shape[0]),
            interpolation = cv2.INTER_NEAREST
        )
    mask = cv2.resize(
        mask,
        (self.init_image_shape[1] // 8, self.init_image_shape[0] // 8),
        interpolation = cv2.INTER_NEAREST
    )
    mask = mask.astype(np.float32) / 255.0
    mask = np.tile(mask, (4, 1, 1))
    mask = mask[None].transpose(0, 1, 2, 3)
    mask = 1 - mask
    return mask
```

第二次缩放 64*64

保留部分权重为0

```
if init_image is not None and mask is not None:
    mask = self._preprocess_mask(mask)
else:
    mask = None
```

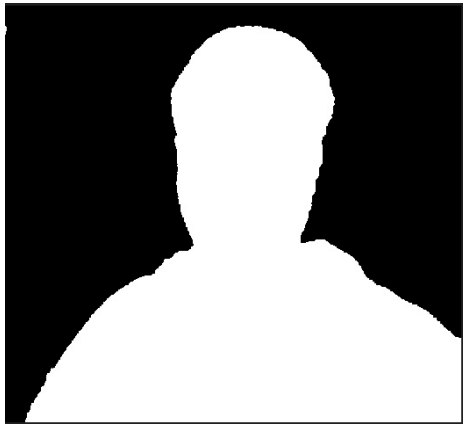
在每次扩散迭代后面添加:

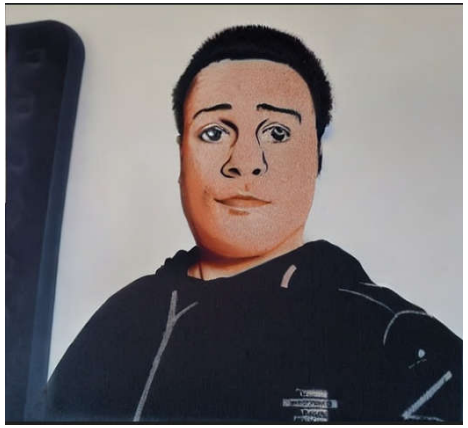
```
# masking for inapinting
if mask is not None:
    t_add = np.array([[t]]).astype(np.long)
    init_latents_proper = self.scheduler.add_noise(init_latents, noise, t_add)
    latents = ((init_latents_proper * mask) + (latents * (1 - mask)))[0]
```

初始输入

生成的

fantastically detailed cute detailed, man,portrait shinkai makoto vibrant Studio ghibli kyoto animation hideaki anno Sakimichan Stanley Artgerm Lau Rossdraws James Jean Marc Simonetti elegant highly detailed digital painting artstation pixiv cyberpunk





sketch effect,a man sit in style of Banksy,fantastically detailed cute detailed,photorealism,