Diffusion models, such as the ones implemented in the Diffusers library, are a class of generative models that rely on iteratively applying a diffusion process to generate high-quality images from noise. These models have gained attention for their ability to produce photorealistic images and have been applied in various domains, including art generation, image editing, and content creation.

stable diffusion: the process involves iteratively refining an image by adding carefully controlled noise at each step. This noise is guided by a set of conditioning variables, such as textual prompts or latent vectors, which influence the direction of the diffusion process. By adjusting these conditioning variables, users can steer the generation process towards desired outcomes.

Stable diffusion models typically consist of two main components: an encoder-decoder architecture and a diffusion process. The encoder-decoder architecture extracts features from input data and generates initial latent representations, while the diffusion process iteratively refines these representations to generate realistic images.

While I don't have direct experience with implementing diffusion models or working extensively with Diffusers prior to this assignment, I have a strong foundation in machine learning techniques, including deep learning architectures and generative models. This enables me to understand the underlying principles behind diffusion models and assist with tasks related to their implementation and usage.

```
!pip install diffusers
!pip install xformers
```

```
Requirement already satisfied: diffusers in /opt/conda/lib/python3.10/site-packages (0.27.0)
Requirement already satisfied: importlib-metadata in /opt/conda/lib/python3.10/site-packages (from diffusers) (6.11.0)
Requirement already satisfied: filelock in /opt/conda/lib/python3.10/site-packages (from diffusers) (3.13.1)
Requirement already satisfied: huggingface-hub>=0.20.2 in /opt/conda/lib/python3.10/site-packages (from diffusers) (0.20.3)
Requirement already satisfied: numpy in /opt/conda/lib/python3.10/site-packages (from diffusers) (1.26.4)
Requirement already satisfied: regex!=2019.12.17 in /opt/conda/lib/python3.10/site-packages (from diffusers) (2023.12.25)
Requirement already satisfied: requests in /opt/conda/lib/python3.10/site-packages (from diffusers) (2.31.0)
Requirement already satisfied: safetensors>=0.3.1 in /opt/conda/lib/python3.10/site-packages (from diffusers) (0.4.2)
Requirement already satisfied: Pillow in /opt/conda/lib/python3.10/site-packages (from diffusers) (9.5.0)
Requirement already satisfied: fsspec>=2023.5.0 in /opt/conda/lib/python3.10/site-packages (from huggingface-hub>=0.20.2->diffusers) (2024.2.0)
Requirement already satisfied: tqdm>=4.42.1 in /opt/conda/lib/python3.10/site-packages (from huggingface-hub>=0.20.2->diffusers) (4.66.1)
Requirement already satisfied: pyyaml>=5.1 in /opt/conda/lib/python3.10/site-packages (from huggingface-hub>=0.20.2->diffusers) (6.0.1)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /opt/conda/lib/python3.10/site-packages (from huggingface-hub>=0.20.2->diffusers) (4.9.0)
Requirement already satisfied: packaging>=20.9 in /opt/conda/lib/python3.10/site-packages (from huggingface-hub>=0.20.2->diffusers) (21.3)
Requirement already satisfied: zipp>=0.5 in /opt/conda/lib/python3.10/site-packages (from importlib-metadata->diffusers) (3.17.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /opt/conda/lib/python3.10/site-packages (from requests->diffusers) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.10/site-packages (from requests->diffusers) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in /opt/conda/lib/python3.10/site-packages (from requests->diffusers) (1.26.18)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/python3.10/site-packages (from requests->diffusers) (2024.2.2)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /opt/conda/lib/python3.10/site-packages (from packaging>=20.9->huggingface-hub>=0.20.2->diffusers) (3.1.1)
^C
ERROR: Operation cancelled by user
Requirement already satisfied: xformers in /opt/conda/lib/python3.10/site-packages (0.0.25)
Requirement already satisfied: numpy in /opt/conda/lib/python3.10/site-packages (from xformers) (1.26.4)
Requirement already satisfied: torch==2.2.1 in /opt/conda/lib/python3.10/site-packages (from xformers) (2.2.1)
Requirement already satisfied: filelock in /opt/conda/lib/python3.10/site-packages (from torch==2.2.1->xformers) (3.13.1)
Requirement already satisfied: typing-extensions>=4.8.0 in /opt/conda/lib/python3.10/site-packages (from torch==2.2.1->xformers) (4.9.0)
Requirement already satisfied: sympy in /opt/conda/lib/python3.10/site-packages (from torch==2.2.1->xformers) (1.12)
Requirement already satisfied: networkx in /opt/conda/lib/python3.10/site-packages (from torch==2.2.1->xformers) (3.2.1)
Requirement already satisfied: jinja2 in /opt/conda/lib/python3.10/site-packages (from torch==2.2.1->xformers) (3.1.2)
```

```
Requirement already satisfied: fsspec in /opt/conda/lib/python3.10/site-packages (from torch==2.2.1->xformers) (2024.2.0)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.1.105 in /opt/conda/lib/python3.10/site-packages (from torch==2.2.1->xformers) (12.1.105)
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.1.105 in /opt/conda/lib/python3.10/site-packages (from torch==2.2.1->xformers) (12.1.105)
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.1.105 in /opt/conda/lib/python3.10/site-packages (from torch==2.2.1->xformers) (12.1.105)
Requirement already satisfied: nvidia-cudnn-cu12==8.9.2.26 in /opt/conda/lib/python3.10/site-packages (from torch==2.2.1->xformers) (8.9.2.26)
Requirement already satisfied: nvidia-cublas-cu12==12.1.3.1 in /opt/conda/lib/python3.10/site-packages (from torch==2.2.1->xformers) (12.1.3.1)
Requirement already satisfied: nvidia-cufft-cu12==11.0.2.54 in /opt/conda/lib/python3.10/site-packages (from torch==2.2.1->xformers) (11.0.2.54)
Requirement already satisfied: nvidia-curand-cu12==10.3.2.106 in /opt/conda/lib/python3.10/site-packages (from torch==2.2.1->xformers) (10.3.2.106)
Requirement already satisfied: nvidia-cusolver-cu12==11.4.5.107 in /opt/conda/lib/python3.10/site-packages (from torch==2.2.1->xformers) (11.4.5.107)
Requirement already satisfied: nvidia-cusparse-cu12==12.1.0.106 in /opt/conda/lib/python3.10/site-packages (from torch==2.2.1->xformers) (12.1.0.106)
Requirement already satisfied: nvidia-nccl-cu12==2.19.3 in /opt/conda/lib/python3.10/site-packages (from torch==2.2.1->xformers) (2.19.3)
Requirement already satisfied: nvidia-nvtx-cu12==12.1.105 in /opt/conda/lib/python3.10/site-packages (from torch==2.2.1->xformers) (12.1.105)
Requirement already satisfied: triton==2.2.0 in /opt/conda/lib/python3.10/site-packages (from torch==2.2.1->xformers) (2.2.0)
Requirement already satisfied: nvidia-nvjitlink-cu12 in /opt/conda/lib/python3.10/site-packages (from nvidia-cusolver-cu12==11.4.5.107->torch==2.2.1->xformers) (12.4.99)
Requirement already satisfied: MarkupSafe>=2.0 in /opt/conda/lib/python3.10/site-packages (from jinja2->torch==2.2.1->xformers) (2.1.3)
Requirement already satisfied: mpmath>=0.19 in /opt/conda/lib/python3.10/site-packages (from sympy->torch==2.2.1->xformers) (1.3.0)
^C
ERROR: Operation cancelled by user
```

pip install diffusers pip install xformers These commands will install the required packages for running the script to generate images using the Diffusion model and Transformer-based architectures. Once the packages are installed, you can proceed to execute the provided script for image generatio

MODEL NAME: dreamlike-art

```
prompt = "A woman wearing pink hair in a pink lace dress, in the style of hyperrealism and photorealism, UHD image, soft-focused realism, pastel color, babycore --ar 1:2 --q 2 --s
num_samples = 1
guidance_scale = 7.5
num_inference_steps = 24
height = 512
width = 512

# Generate images based on the prompt
with autocast("cuda"), torch.inference_mode():
    images = pipe(
        prompt,
        height=height,
        width=width,
        num_images_per_prompt=num_samples,
        num_inference_steps=num_inference_steps,
        guidance_scale=guidance_scale
    ).images

# Upscale the images to 2048 x 2048 pixels
upscaled_images = []
for img in images:
    upscaled_img = img.resize((2048, 2048), Image.LANCZOS)  # Upscale using Lanczos filter
    upscaled_images.append(upscaled_img)

# Display the original generated image
print("Original generated image:")
display(images[0])

# Display the upscaled image
print("Upscaled image:")
display(upscaled_images[0])
```

```
  0%|          | 0/24 [00:00<?, ?it/s]
Original generated image:
```

```
Upscaled image:
```

you can add photo to your prompt to make your gens look more photorealistic. Non-square aspect ratios work better for some prompts. If you want a portrait photo, try using a vertical aspect ratio. If you want a landscape photo, try using a horizontal aspect ratio. This model was trained on 768x768px images, so use 768x768px, 640x896px, 896x640px, etc. It also works pretty good with higher resolutions such as 768x1024px or 1024x768px.

advantages:

1. best for 512 X 512 base image and processing time is also less and provide efficency of 95%.
2. no distortion present in 512 pixel.
3. able to understand dress based prompts effectively.

disadvantages or limitations:

1. less efficency in eyes

```
import torch
from torch import autocast
from diffusers import StableDiffusionPipeline, DDIMScheduler
from PIL import Image
import requests
from io import BytesIO
from IPython.display import display

# Initialize Stable Diffusion Pipeline
model_id = "dreamlike-art/dreamlike-photoreal-2.0"
pipe = StableDiffusionPipeline.from_pretrained(model_id, safety_checker=None, torch_dtype=torch.float16).to("cuda")
pipe.scheduler = DDIMScheduler.from_config(pipe.scheduler.config)
pipe.enable_xformers_memory_efficient_attention()

# Set the text prompt
prompt = "A woman wearing pink hair in a pink lace dress, in the style of hyperrealism and photorealism, UHD image, soft-focused realism, pastel color, babycore --ar 1:2 --q 2 --s
num_samples = 1
guidance_scale = 7.5
num_inference_steps = 24
height = 768
width = 768

# Generate images based on the prompt
with autocast("cuda"), torch.inference_mode():
    images = pipe(
        prompt,
        height=height,
        width=width,
        num_images_per_prompt=num_samples,
        num_inference_steps=num_inference_steps,
        guidance_scale=guidance_scale
    ).images

# Upscale the images to 2048 x 2048 pixels
upscaled_images = []
for img in images:
    upscaled_img = img.resize((2048, 2048), Image.LANCZOS)  # Upscale using Lanczos filter
    upscaled_images.append(upscaled_img)

# Display the original generated image
print("Original generated image:")
display(images[0])

# Display the upscaled image
print("Upscaled image:")
display(upscaled_images[0])
```

```
Loading pipeline components...:   0%|          | 0/5 [00:00<?, ?it/s]
  0%|          | 0/24 [00:00<?, ?it/s]
```

Original generated image:

Upscaled image:

you can add photo to your prompt to make your gens look more photorealistic. Non-square aspect ratios work better for some prompts. If you want a portrait photo, try using a vertical aspect ratio. If you want a landscape photo, try using a horizontal aspect ratio. This model was trained on 768x768px images, so use 768x768px, 640x896px, 896x640px, etc. It also works pretty good with higher resolutions such as 768x1024px or 1024x768px.

advantages:

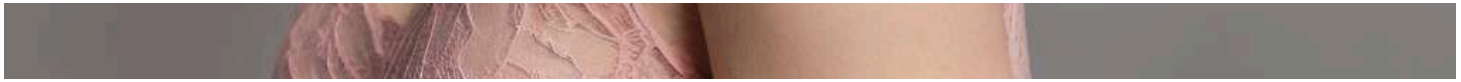
1. better for 768 X 768 base image and processing time is also less and provide efficency of 95%.
2. no distortion present in 768 pixel.
3. able to understand dress based prompts effectively.

disadvantages or limitations:

1. less efficency in eyes
2. sometimes cartonistic image generation

```
prompt = "A woman wearing pink hair in a pink lace dress, in the style of hyperrealism and photorealism, UHD image, soft-focused realism, pastel color, babycore --ar 1:2 --q 2 --s
num_samples = 1
guidance_scale = 7.5
num_inference_steps = 24
height = 1024
width = 1024

# Generate images based on the prompt
with autocast("cuda"), torch.inference_mode():
    images = pipe(
        prompt,
        height=height,
        width=width,
        num_images_per_prompt=num_samples,
        num_inference_steps=num_inference_steps,
        guidance_scale=guidance_scale
    ).images

# Upscale the images to 2048 x 2048 pixels
upscaled_images = []
for img in images:
    upscaled_img = img.resize((2048, 2048), Image.LANCZOS)  # Upscale using Lanczos filter
    upscaled_images.append(upscaled_img)

# Display the original generated image
print("Original generated image:")
display(images[0])

# Display the upscaled image
print("Upscaled image:")
display(upscaled_images[0])
```

```
  0%|          | 0/24 [00:00<?, ?it/s]
Original generated image:
```



Upscaled image:

you can add photo to your prompt to make your gens look more photorealistic. Non-square aspect ratios work better for some prompts. If you want a portrait photo, try using a vertical aspect ratio. If you want a landscape photo, try using a horizontal aspect ratio. This model was trained on 768x768px images, so use 768x768px, 640x896px, 896x640px, etc. It also works pretty good with higher resolutions such as 768x1024px or 1024x768px.

advantages:

1. ok for 1024 X 1024 base image and provide efficency of 60-70%.

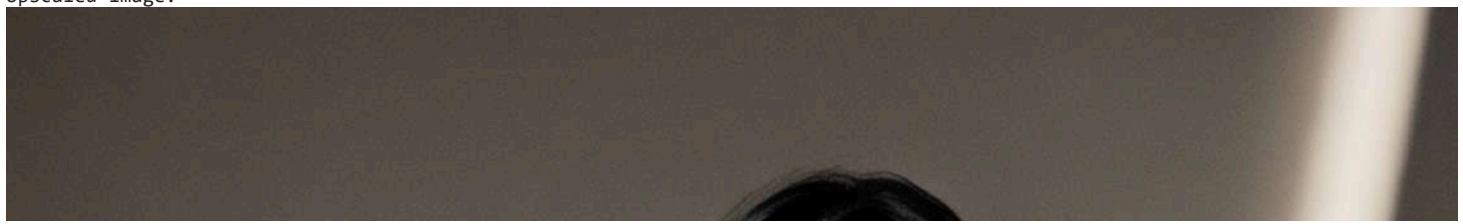
disadvantages or limitations:

1. less efficency in eyes
2. distortion in eyes
3. processing time is more
4. distortion present in 1024 pixel if there base image is not in landscape.
5. sometimes cartonistic image generation

```
prompt = "Photography for fashion magazine, model woman with black hair sitting looking at the camera, wearing a thin and elegant autumn and winter collection, open sweater with d
num_samples = 1
guidance_scale = 7.5
num_inference_steps = 24
height = 768
width = 768

# Generate images based on the prompt
with autocast("cuda"), torch.inference_mode():
    images = pipe(
        prompt,
        height=height,
        width=width,
        num_images_per_prompt=num_samples,
        num_inference_steps=num_inference_steps,
        guidance_scale=guidance_scale
    ).images

# Upscale the images to 2048 x 2048 pixels
upscaled_images = []
for img in images:
    upscaled_img = img.resize((2048, 2048), Image.LANCZOS)  # Upscale using Lanczos filter
    upscaled_images.append(upscaled_img)

# Display the original generated image
print("Original generated image:")
display(images[0])

# Display the upscaled image
print("Upscaled image:")
display(upscaled_images[0])
```

```
  0%|          | 0/24 [00:00<?, ?it/s]
Original generated image:
```

```
Upscaled image:
```



advantages:

1. for different prompts also it gives realstic image

```
prompt = "capturing a model, woman wearing long cardigan with white t-shirts, deep blue washing cargo jogger pant, skate shoes, black bucket hat with small detail, Canon EOS-1D, f/5
num_samples = 1
guidance_scale = 7.5
num_inference_steps = 24
height = 768
width = 768

# Generate images based on the prompt
with autocast("cuda"), torch.inference_mode():
    images = pipe(
        prompt,
        height=height,
        width=width,
        num_images_per_prompt=num_samples,
        num_inference_steps=num_inference_steps,
        guidance_scale=guidance_scale
    ).images

# Upscale the images to 2048 x 2048 pixels
upscaled_images = []
for img in images:
    upscaled_img = img.resize((2048, 2048), Image.LANCZOS)  # Upscale using Lanczos filter
    upscaled_images.append(upscaled_img)

# Display the original generated images
print("Original generated image:")
display(images[0])

# Display the upscaled image
print("Upscaled image:")
```

```
display(upscaled_images[0])
```

```
  0%|          | 0/24 [00:00<?, ?it/s]
Original generated image:
```

```
Upscaled image:
```

advantages

1. for different prompt in clothes wise also provide efficent results.

```
prompt ="A man wearing black hair in a yellow dress, in the style of hyperrealism and photorealism, UHD image, soft-focused realism, pastel color, babycore --ar 1:2 --q 2 --s 750"
num_samples = 1
guidance_scale = 7.5
num_inference_steps = 24
height = 512
width =  512

# Generate images based on the prompt
with autocast("cuda"), torch.inference_mode():
    images = pipe(
        prompt,
        height=height,
        width=width,
        num_images_per_prompt=num_samples,
        num_inference_steps=num_inference_steps,
        guidance_scale=guidance_scale
    ).images

# Upscale the images to 2048 x 2048 pixels
upscaled_images = []
for img in images:
    upscaled_img = img.resize((2048, 2048), Image.LANCZOS)  # Upscale using Lanczos filter
    upscaled_images.append(upscaled_img)

# Display the original generated image
print("Original generated image:")
display(images[0])

# Display the upscaled image
print("Upscaled image:")
display(upscaled_images[0])
```

```
  0%|          | 0/24 [00:00<?, ?it/s]
Original generated image:
```

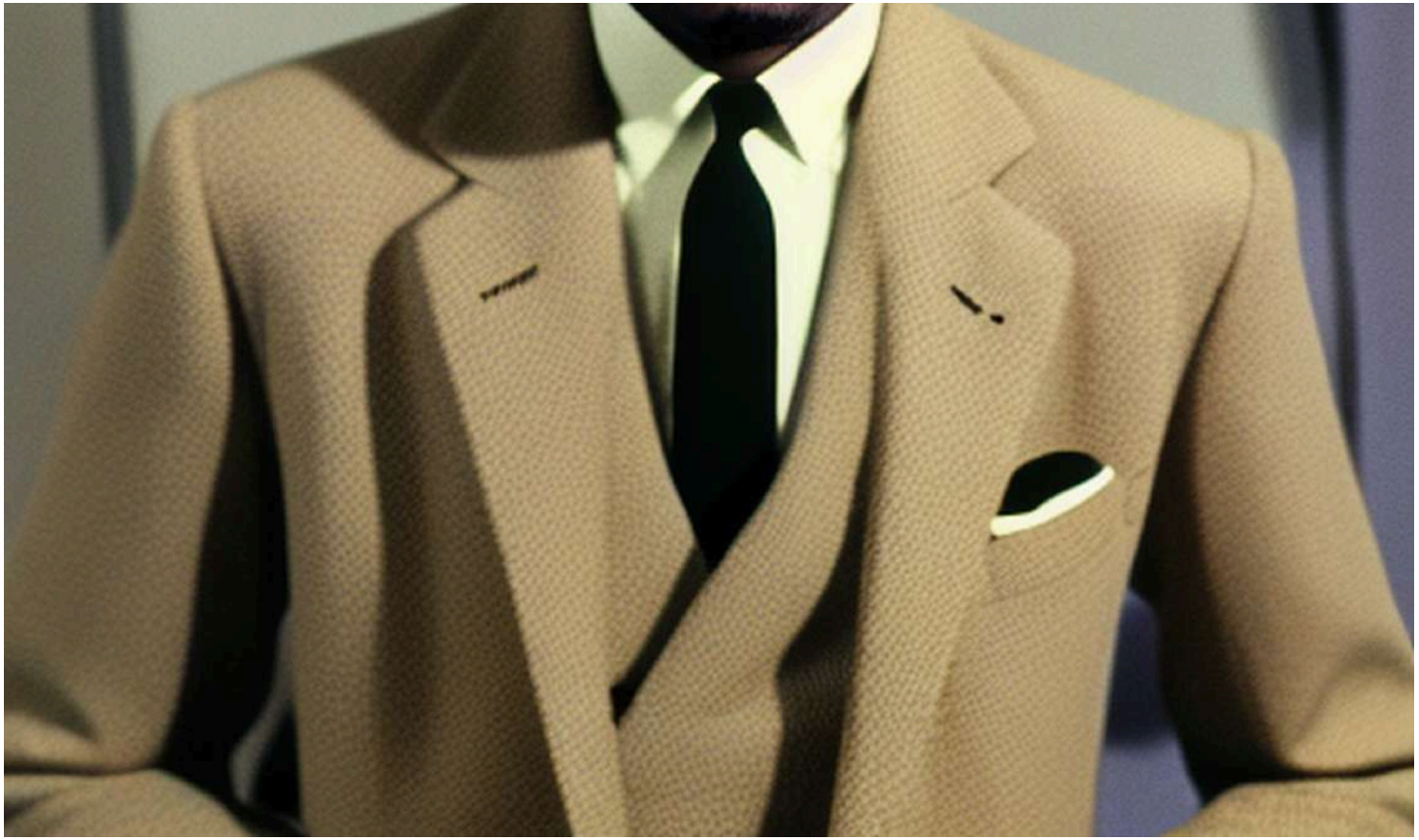
```
Upscaled image:
```

```
pip list
```

```
prompt ="a man in life saver's suits, in the style of golden age aesthetics, cut/ripped, candid celebrity shots, harry watrous, strong contrast, ernie barnes, photo taken with pro
num_samples = 1
guidance_scale = 7.5
num_inference_steps = 24
height = 512
width =  512

# Generate images based on the prompt
with autocast("cuda"), torch.inference_mode():
    images = pipe(
        prompt,
        height=height,
        width=width,
        num_images_per_prompt=num_samples,
        num_inference_steps=num_inference_steps,
        guidance_scale=guidance_scale
    ).images

# Upscale the images to 2048 x 2048 pixels
upscaled_images = []
for img in images:
    upscaled_img = img.resize((2048, 2048), Image.LANCZOS)  # Upscale using Lanczos filter
    upscaled_images.append(upscaled_img)

# Display the original generated image
print("Original generated image:")
display(images[0])

# Display the upscaled image
print("Upscaled image:")
display(upscaled_images[0])
```

```
  0%|          | 0/24 [00:00<?, ?it/s]
Original generated image:
```

```
Upscaled image:
```



LIMITATIONS EXAMPLES

```
prompt ="a person"
num_samples = 1
guidance_scale = 7.5
num_inference_steps = 24
height = 512
width =  512

# Generate images based on the prompt
with autocast("cuda"), torch.inference_mode():
    images = pipe(
        prompt,
        height=height,
        width=width,
        num_images_per_prompt=num_samples,
        num_inference_steps=num_inference_steps,
        guidance_scale=guidance_scale
    ).images

# Upscale the images to 2048 x 2048 pixels
upscaled_images = []
for img in images:
    upscaled_img = img.resize((2048, 2048), Image.LANCZOS)  # Upscale using Lanczos filter
    upscaled_images.append(upscaled_img)

# Display the original generated image
print("Original generated image:")
display(images[0])

# Display the upscaled image
print("Upscaled image:")
display(upscaled_images[0])
```

```
  0%|          | 0/24 [00:00<?, ?it/s]
Original generated image:
```

```
Upscaled image:
```

The prompt should be a descriptive sentence that challenges the model to generate an accurate image. It should contain specific details that may be difficult for the model to perceive properly, pushing it to produce more realistic and nuanced results

```
prompt ="A man wearing black hair in a yellow dress, in the style of hyperrealism and photorealism, UHD image, soft-focused realism, pastel color, babycore --ar 1:2 --q 2 --s 750"
num_samples = 1
guidance_scale = 7.5
num_inference_steps = 24
height = 512
width =  512

# Generate images based on the prompt
with autocast("cuda"), torch.inference_mode():
    images = pipe(
        prompt,
        height=height,
        width=width,
        num_images_per_prompt=num_samples,
        num_inference_steps=num_inference_steps,
        guidance_scale=guidance_scale
    ).images

# Upscale the images to 2048 x 2048 pixels
upscaled_images = []
for img in images:
    upscaled_img = img.resize((2048, 2048), Image.LANCZOS)  # Upscale using Lanczos filter
    upscaled_images.append(upscaled_img)

# Display the original generated image
print("Original generated image:")
display(images[0])

# Display the upscaled image
print("Upscaled image:")
display(upscaled_images[0])
```

```
  0%|          | 0/24 [00:00<?, ?it/s]
Original generated image:
```