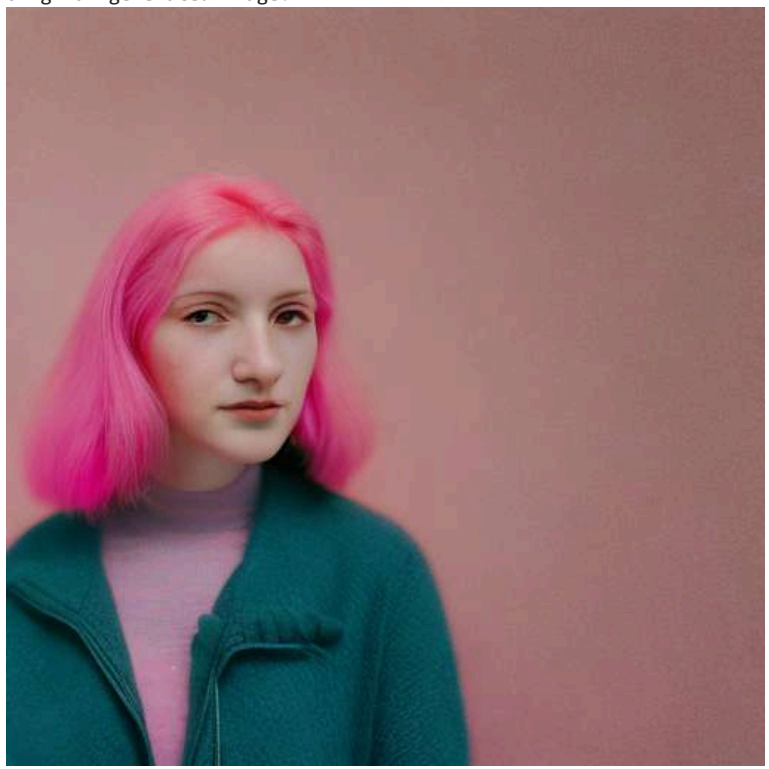
```
!pip install diffusers
!pip install xformers
```

installing dependencies

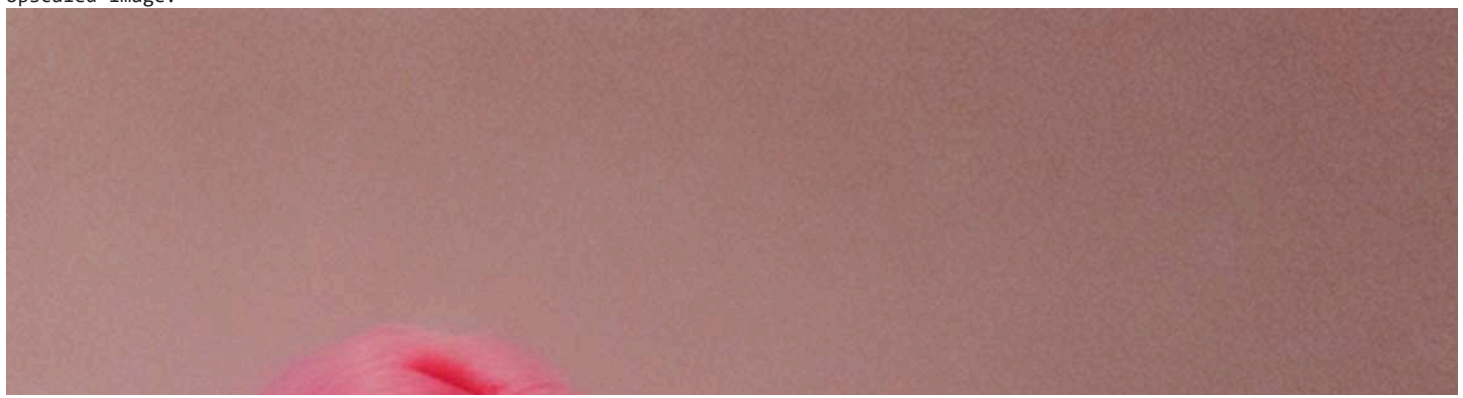
```
import torch
from torch import autocast
from diffusers import StableDiffusionPipeline, DDIMScheduler
from PIL import Image
import requests
from io import BytesIO
from IPython.display import display

# Initialize Stable Diffusion Pipeline
model_id = "wavymulder/Analog-Diffusion"
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 7
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])
```

```
2024-03-16 21:07:57.118614: E external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:9261] Unable to register cuDNN factory: At
2024-03-16 21:07:57.118677: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:607] Unable to register cuFFT factory: Att
2024-03-16 21:07:57.120556: E external/local_xla/xla/stream_executor/cuda/cuda_blas.cc:1515] Unable to register cuBLAS factory:
vae/diffusion_pytorch_model.safetensors not found
Fetching 13 files:   0%|          | 0/13 [00:00<?, ?it/s]
diffusion_pytorch_model.bin:   0%|          | 0.00/3.44G [00:00<?, ?B/s]
Loading pipeline components...:   0%|          | 0/6 [00:00<?, ?it/s]
You have disabled the safety checker for <class 'diffusers.pipelines.stable_diffusion.pipeline_stable_diffusion.StableDiffusionP
  0%|          | 0/24 [00:00<?, ?it/s]
Original generated image:
```



```
Upscaled image:
```



**advantages:**

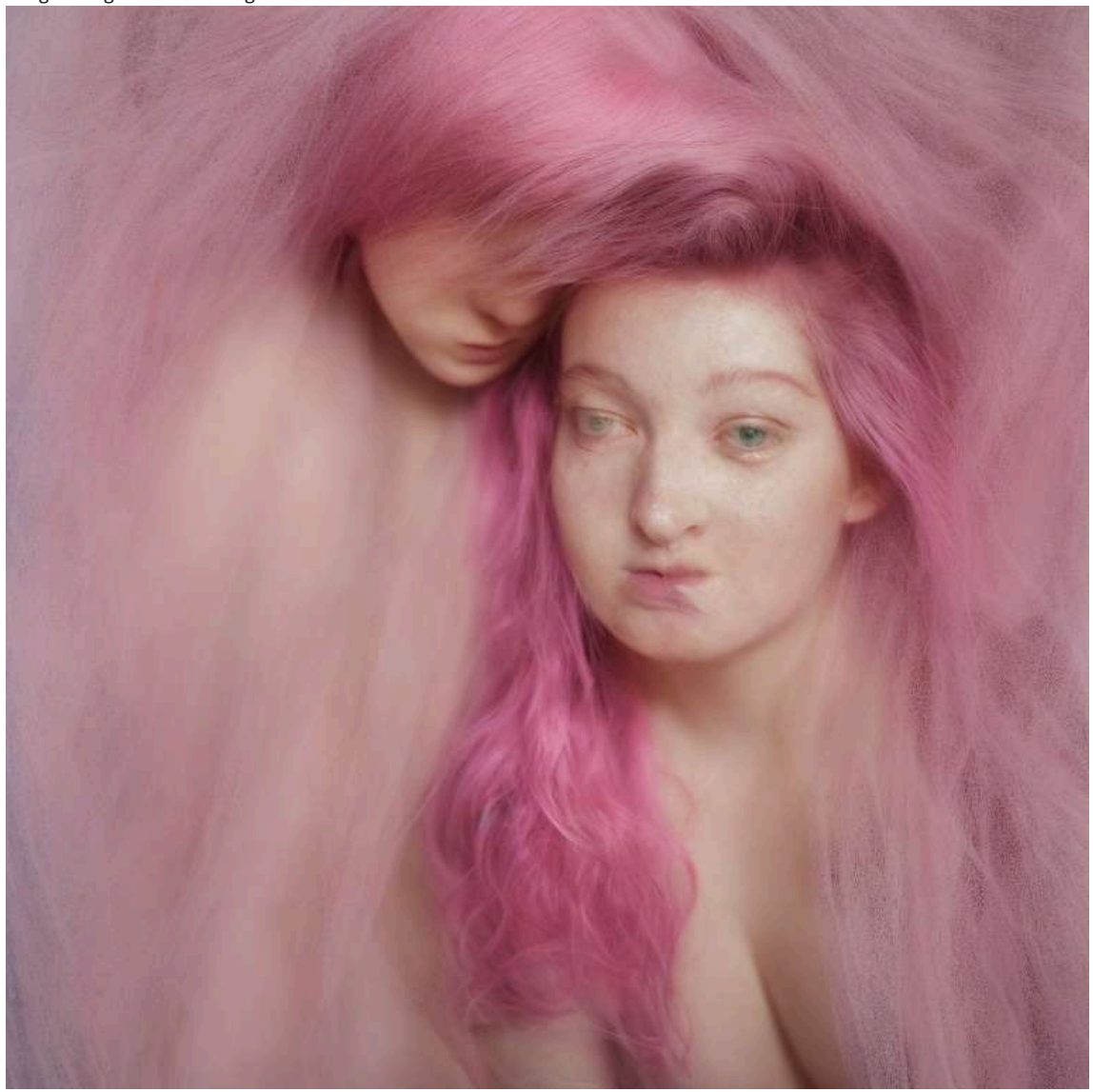
1. good for 512 X 512 base image and processing time is also less and provide efficency of 70%.
2. no distortion present in 512 pixel.
3. able to understand dress based prompts effectively.

**disadvantages or limitations:**

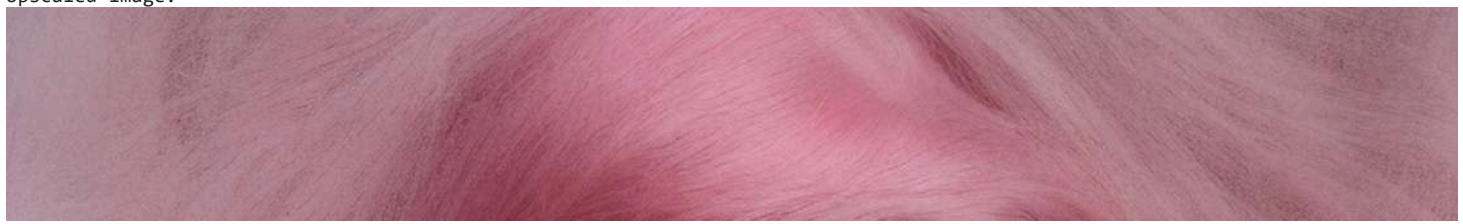
1. less efficency in eyes

```
# 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])
```

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



```
Upscaled image:
```



**advantages:**

1. processing time is less and provide efficency of 70%..

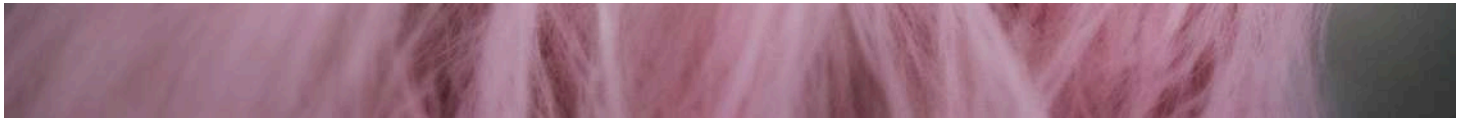
**disadvantages or limitations:**

1. less efficency in eyes
2. distortion present in 768 pixel.
3. not able to understand dress based prompts effectively.
4. less efficency in eyes

```
# 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 = 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:

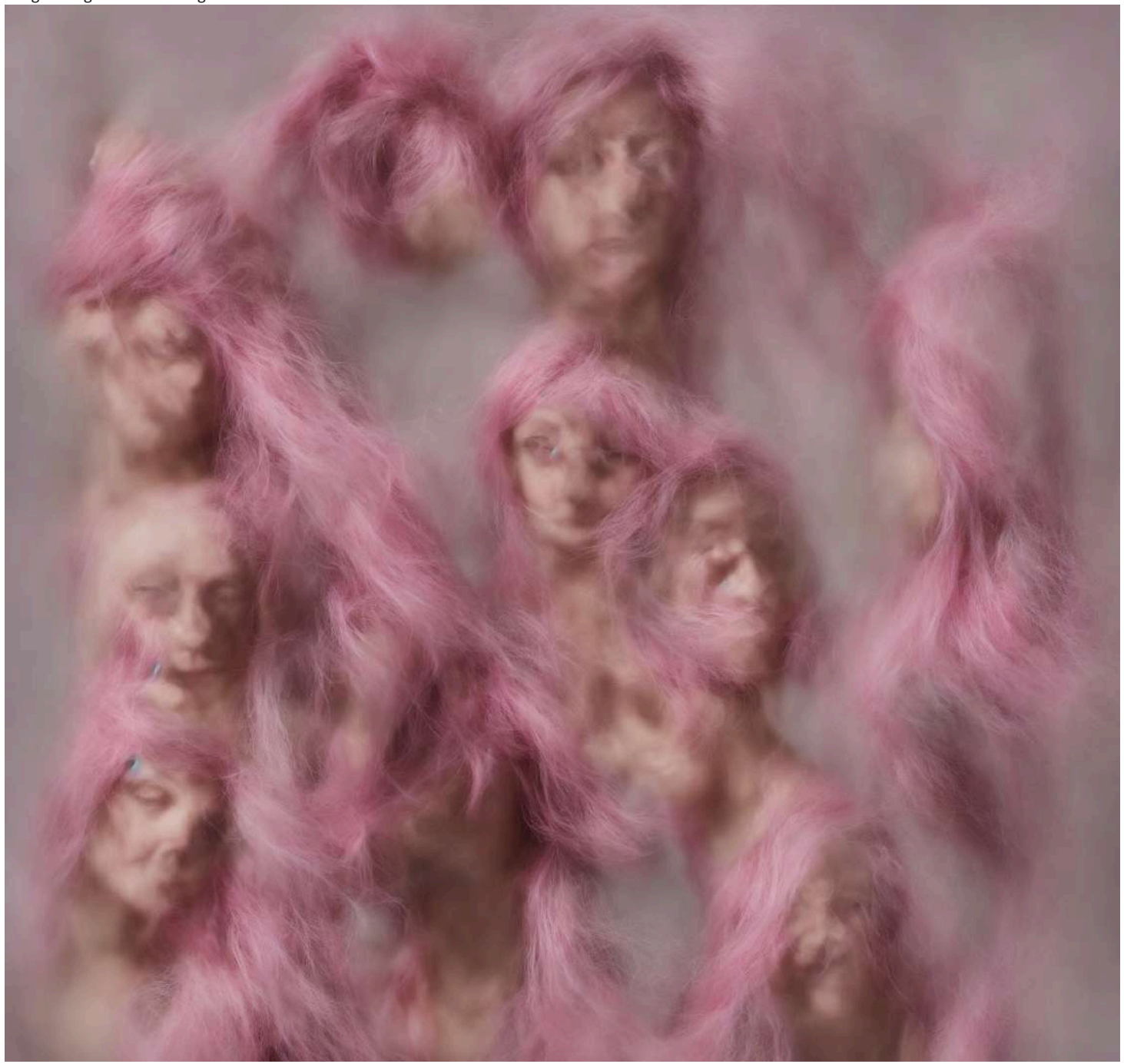


**disadvantages or limitations:**

1. less efficency in eyes
2. distortion present in base 1024 pixel.
3. not able to understand dress properly
4. processing time is high and provide very less efficency.

```
# 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 = 2048
width = 2048

# 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:
```



**disadvantages or limitations:**

1. less efficency in eyes
2. distortion present in 2024 pixel.
3. not able to understand dress properly.
4. worst than any other model
5. can't able to detect face properly

**OTHER PROMPTS EXAMPLES**

```
# Set the text prompt
prompt = "A nike and jaquemus futuristic winter outfit collaboration, made for winter, use pastel tie dye colors, puffy and wool textures, minimal branding but unique design. prov
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])
```

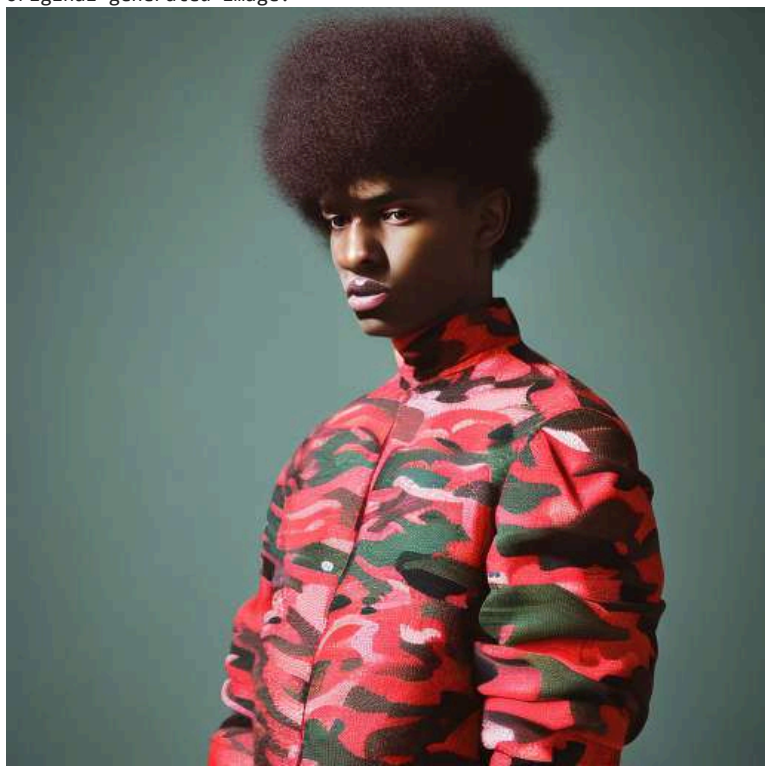
```
  0%|          | 0/24 [00:00<?, ?it/s]
Original generated image:
```

```
Upscaled image:
```

```
# Set the text prompt
prompt = "[contemporary editorial fashion shot] + [black male, striking braided hairstyle, intense expression, medium-dark complexion] + [crimson and green camouflage military-ins
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])
```

```
The following part of your input was truncated because CLIP can only handle sequences up to 77 tokens: ['mood ] -- ar 4 : 5 -- s
  0%|          | 0/24 [00:00<?, ?it/s]
```

Original generated image:



Upscaled image:

```
# Set the text prompt
prompt = "a man dazed editorial Style, Eleven: Eleven – A Sustainable Luxury Collection, eco - friendly textile, natural and gray colors, for the New Era Consumers --ar 4:5"
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])
```

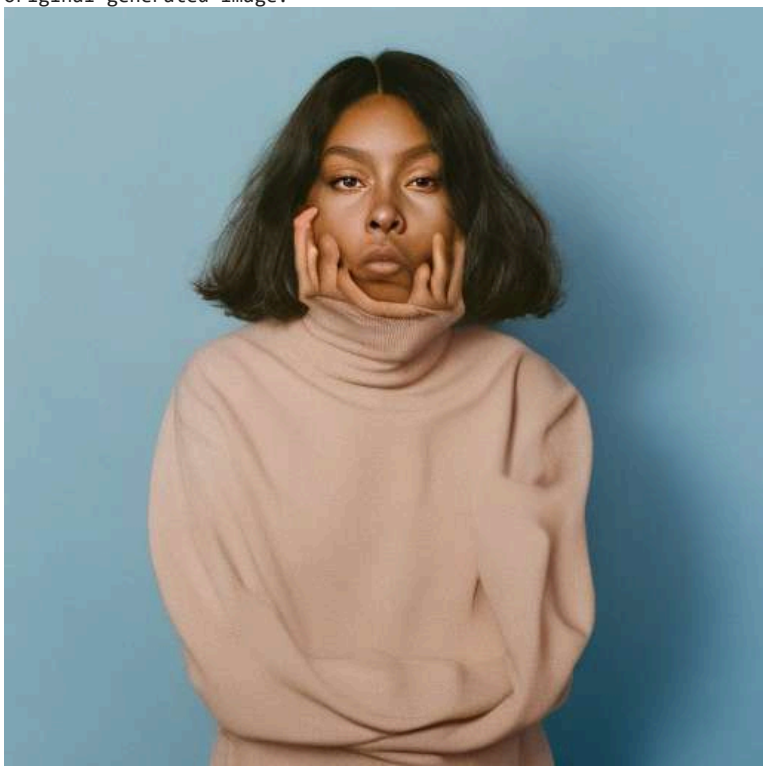
```
  0%|          | 0/24 [00:00<?, ?it/s]
Original generated image:
```

```
Upscaled image:
```

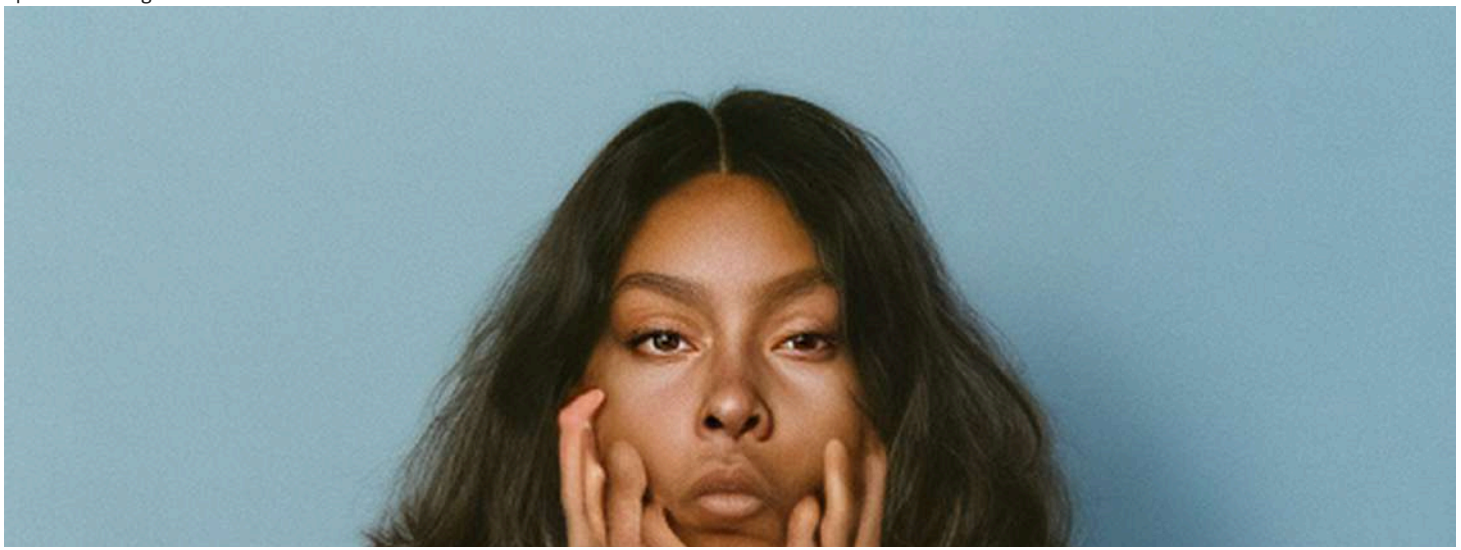
**LIMITATIONS EXAMPLES**

```
prompt = "portrait of a trendy light brown skinned woman, emotive, rebellious facial features, futuristic turtle neck and jacket, iconic album cover, washed colors, studio lightin
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])
```

```
Token indices sequence length is longer than the specified maximum sequence length for this model (79 > 77). Running this sequer
The following part of your input was truncated because CLIP can only handle sequences up to 77 tokens: [': 9']
  0%|          | 0/24 [00:00<?, ?it/s]
Original generated image:
```



```
Upscaled image:
```



```
# Set the text prompt
prompt = "chameleon-inspired streetwear collection by supreme and kith released in soho 2017"
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:
```