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

```
  Downloading nvidia_cudnn_cu12-8.9.2.26-py3-none-manylinux1_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cublas-cu12==12.1.3.1 (from torch==2.2.1->xformers)
  Downloading nvidia_cublas_cu12-12.1.3.1-py3-none-manylinux1_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cufft-cu12==11.0.2.54 (from torch==2.2.1->xformers)
  Downloading nvidia_cufft_cu12-11.0.2.54-py3-none-manylinux1_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-curand-cu12==10.3.2.106 (from torch==2.2.1->xformers)
  Downloading nvidia_curand_cu12-10.3.2.106-py3-none-manylinux1_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cusolver-cu12==11.4.5.107 (from torch==2.2.1->xformers)
```

```
  Attempting uninstall: torch
    Found existing installation: torch 2.1.2
    Uninstalling torch-2.1.2:
      Successfully uninstalled torch-2.1.2
Successfully installed nvidia-cublas-cu12-12.1.3.1 nvidia-cuda-cupti-cu12-12.1.105 nvidia-cuda-nvrtc-cu12-12.1.105 nvidia-cuda-runtime-cu12-12.1.105 nvidia-cudnn-cu12-8.9.2
```

install all packages required

```
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 = "SG161222/Realistic_Vision_V6.0_B1_noVAE"
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:")
```

```
    print("Original generated image:")
    display(images[0])

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

```
t/diffusion_pytorch_model.safetensors not found
ding pipeline components...:   0%|          | 0/6 [00:00<?, ?it/s]
 have disabled the safety checker for <class 'diffusers.pipelines.stable_diffusion.pipeline_stable_diffusion.StableDiffusionPipeline'> by passing `safety_checker=None`. Ensure t
%|          | 0/24 [00:00<?, ?it/s]
ginal generated image:
```

```
caled 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 93%.
2. no distortion present in 512 pixel.
3. able to understand dress based prompts effectively.

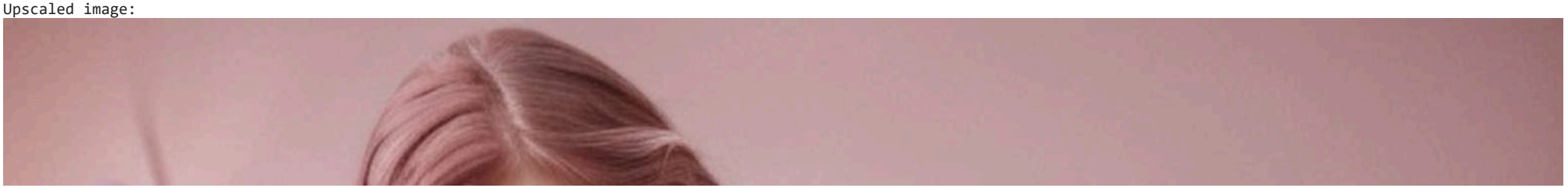
**disadvantages or limitations:**

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:
```



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 90%.
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 = 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:
```

**advantages:**

1. better than dreamart for 1024 X 1024 base image and processing time is little high than dreamartai.
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 = 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:
```

**advantages:**

1. able to create 2048 X 2048 base image

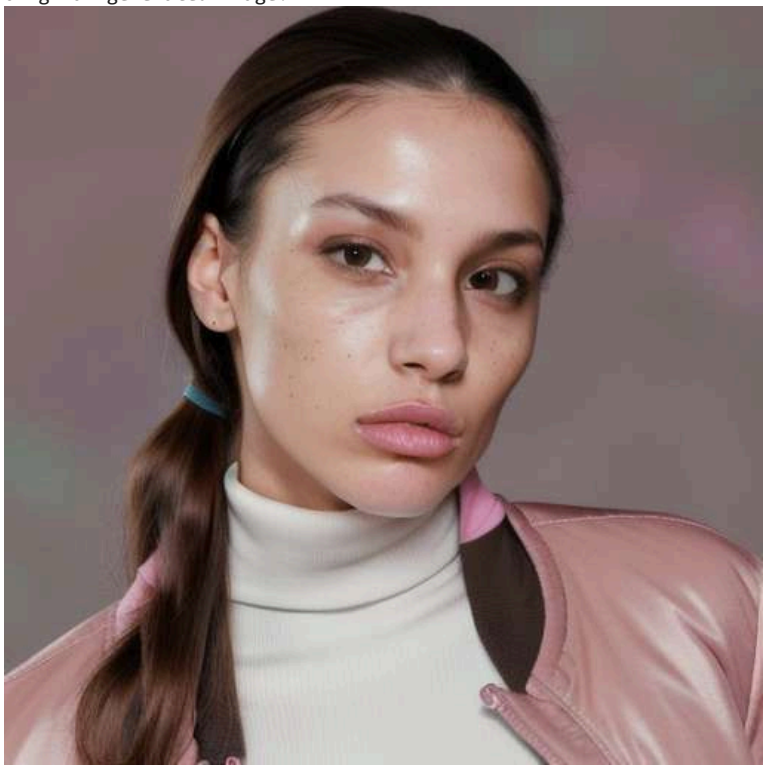
**disadvantages or limitations:**

1. less efficency in eyes
2. processing time is too high.
3. ditortion
4. multiple images of person in same image

**OTHER PROMPTS 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])
```

```
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:
```