Stable Diffusion 2.1

```
print("Hello!")
```

```
Hello!
```

Install dependencies (~1 min.)

```
#@title

!pip install --upgrade git+https://github.com/huggingface/diffusers.git
!pip install --upgrade git+https://github.com/huggingface/transformers/
!pip install accelerate==0.12.0
!pip install scipy
!pip install ftfy
!pip install gradio -q
```

```
Requirement already satisfied: filelock in /usr/local/lib/python3.8/dist-packages (from transformers==4.27.0.dev0) (3.9.0)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.8/dist-packages (from transformers==4.27.0.dev0) (6.0)
Collecting tokenizers!=0.11.3,<0.14,>=0.11.1
  Downloading tokenizers-0.13.2-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (7.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.6/7.6 MB 63.6 MB/s eta 0:00:00
Requirement already satisfied: requests in /usr/local/lib/python3.8/dist-packages (from transformers==4.27.0.dev0) (2.25.1)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.8/dist-packages (from huggingface-hub<1.0,>=0.11.
Requirement already satisfied: chardet<5,>=3.0.2 in /usr/local/lib/python3.8/dist-packages (from requests->transformers==4.27.0.dev0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.8/dist-packages (from requests->transformers==4.27.0.dev0
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.8/dist-packages (from requests->transformers==4.27.0.dev0) (2.1
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.8/dist-packages (from requests->transformers==4.27.0.d
Building wheels for collected packages: transformers
  Building wheel for transformers (pyproject.toml) ... done
  Created wheel for transformers: filename=transformers-4.27.0.dev0-py3-none-any.whl size=6465467 sha256=27ac887810f28c7ff2f2346e6a47
  Stored in directory: /tmp/pip-ephem-wheel-cache-zznb2ex0/wheels/e5/f5/cc/ad6c154eb1e3a67cc966c2e68f008d19272cd57e0e8375155b
Successfully built transformers
Installing collected packages: tokenizers, transformers
Successfully installed tokenizers-0.13.2 transformers-4.27.0.dev0
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting accelerate==0.12.0
  Downloading accelerate-0.12.0-py3-none-any.whl (143 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 144.0/144.0 KB 9.7 MB/s eta 0:00:00
Requirement already satisfied: psutil in /usr/local/lib/python3.8/dist-packages (from accelerate==0.12.0) (5.4.8)
Requirement already satisfied: torch>=1.4.0 in /usr/local/lib/python3.8/dist-packages (from accelerate==0.12.0) (1.13.1+cu116)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.8/dist-packages (from accelerate==0.12.0) (6.0)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.8/dist-packages (from accelerate==0.12.0) (23.0)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.8/dist-packages (from accelerate==0.12.0) (1.21.6)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.8/dist-packages (from torch>=1.4.0->accelerate==0.12.0) (4
Installing collected packages: accelerate
Successfully installed accelerate-0.12.0
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: scipy in /usr/local/lib/python3.8/dist-packages (1.7.3)
Requirement already satisfied: numpy<1.23.0,>=1.16.5 in /usr/local/lib/python3.8/dist-packages (from scipy) (1.21.6)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting ftfy
  Downloading ftfy-6.1.1-py3-none-any.whl (53 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 53.1/53.1 KB 4.4 MB/s eta 0:00:00
Requirement already satisfied: wcwidth>=0.2.5 in /usr/local/lib/python3.8/dist-packages (from ftfy) (0.2.6)
Installing collected packages: ftfy
Successfully installed ftfy-6.1.1
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 14.2/14.2 MB 90.6 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 92.7 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.9/56.9 KB 9.0 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 71.5/71.5 KB 10.6 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 107.0/107.0 KB 14.4 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.2/56.2 KB 5.5 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 140.7/140.7 KB 15.1 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 84.5/84.5 KB 12.9 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 50.5/50.5 KB 7.1 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 65.8/65.8 KB 10.9 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 69.6/69.6 KB 10.4 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 58.3/58.3 KB 9.4 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 80.6/80.6 KB 13.3 MB/s eta 0:00:00
  Building wheel for ffmpy (setup.py) ... done
  Building wheel for python-multipart (setup.py) ... done
```

Run the app

```
#@title

from diffusers import StableDiffusionPipeline, StableDiffusionImg2ImgPipeline, StableDiffusionUpscalePipeline, DiffusionPipeline, StableDiffu
import gradio as gr
import torch
from PIL import Image
import random

state = None
current_steps = 25
attn_slicing_enabled = True
mem_eff_attn_enabled = False

# model_id = 'stabilityai/stable-diffusion-2'
model_id = 'stabilityai/stable-diffusion-2-1'

scheduler = DPMSolverMultistepScheduler.from_pretrained(model_id, subfolder="scheduler")

pipe = StableDiffusionPipeline.from_pretrained(
      model_id,
      revision="fp16",
      torch_dtype=torch.float16 if torch.cuda.is_available() else torch.float32,
      scheduler=scheduler
    ).to("cuda")
pipe.enable_attention_slicing()
if mem_eff_attn_enabled:
  pipe.enable_xformers_memory_efficient_attention()

pipe_i2i = None
pipe_upscale = None
pipe_inpaint = None
pipe_depth2img = None


modes = {
    'txt2img': 'Text to Image',
    'img2img': 'Image to Image',
    'inpaint': 'Inpainting',
    'upscale4x': 'Upscale 4x',
    'depth2img': 'Depth to Image'
}
current_mode = modes['txt2img']

def error_str(error, title="Error"):
    return f"""#### {title}
            {error}"""  if error else ""

def update_state(new_state):
  global state
  state = new_state

def update_state_info(old_state):
  if state and state != old_state:
    return gr.update(value=state)

def set_mem_optimizations(pipe):
    if attn_slicing_enabled:
      pipe.enable_attention_slicing()
    else:
      pipe.disable_attention_slicing()

    if mem_eff_attn_enabled:
      pipe.enable_xformers_memory_efficient_attention()
    # else:
    #   pipe.disable_xformers_memory_efficient_attention()

def get_i2i_pipe(scheduler):

    update_state("Loading image to image model...")

    pipe = StableDiffusionImg2ImgPipeline.from_pretrained(
      model_id,
```

```
        revision="fp16",
        torch_dtype=torch.float16 if torch.cuda.is_available() else torch.float32,
        scheduler=scheduler,
        safety_checker=None,
        feature_extractor=None
      )
      set_mem_optimizations(pipe)
      pipe.to("cuda")
      return pipe

  def get_inpaint_pipe():

    update_state("Loading inpainting model...")

    pipe = DiffusionPipeline.from_pretrained(
        "stabilityai/stable-diffusion-2-inpainting",
        revision="fp16",
        torch_dtype=torch.float16 if torch.cuda.is_available() else torch.float32,
        # scheduler=scheduler # TODO currently setting scheduler here messes up the end result. A bug in Diffusers🧨
      ).to("cuda")
    pipe.scheduler = DPMSolverMultistepScheduler.from_config(pipe.scheduler.config)
    # pipe.enable_attention_slicing()
    # pipe.enable_xformers_memory_efficient_attention()
    set_mem_optimizations(pipe)
    return pipe

  def get_upscale_pipe(scheduler):

      update_state("Loading upscale model...")

      pipe = StableDiffusionUpscalePipeline.from_pretrained(
        "stabilityai/stable-diffusion-x4-upscaler",
        revision="fp16",
        torch_dtype=torch.float16 if torch.cuda.is_available() else torch.float32,
        # scheduler=scheduler
      )
      # pipe.scheduler = DPMSolverMultistepScheduler.from_config(pipe.scheduler.config)
      set_mem_optimizations(pipe)
      pipe.to("cuda")
      return pipe

  def get_depth2img_pipe():

      update_state("Loading depth to image model...")

      pipe = StableDiffusionDepth2ImgPipeline.from_pretrained(
       "stabilityai/stable-diffusion-2-depth",
        revision="fp16",
        torch_dtype=torch.float16 if torch.cuda.is_available() else torch.float32,
        # scheduler=scheduler
      )
      pipe.scheduler = DPMSolverMultistepScheduler.from_config(pipe.scheduler.config)
      set_mem_optimizations(pipe)
      pipe.to("cuda")
      return pipe

  def switch_attention_slicing(attn_slicing):
      global attn_slicing_enabled
      attn_slicing_enabled = attn_slicing

  def switch_mem_eff_attn(mem_eff_attn):
      global mem_eff_attn_enabled
      mem_eff_attn_enabled = mem_eff_attn

  def pipe_callback(step: int, timestep: int, latents: torch.FloatTensor):
      update_state(f"{step}/{current_steps} steps")#\nTime left, sec: {timestep/100:.0f}")

  def inference(inf_mode, prompt, n_images, guidance, steps, width=768, height=768, seed=0, img=None, strength=0.5, neg_prompt=""):

    update_state(" ")

    global current_mode
    if inf_mode != current_mode:
      pipe.to("cuda" if inf_mode == modes['txt2img'] else "cpu")

      if pipe_i2i is not None:
        pipe_i2i.to("cuda" if inf_mode == modes['img2img'] else "cpu")
```

```
      if pipe_inpaint is not None:
        pipe_inpaint.to("cuda" if inf_mode == modes['inpaint'] else "cpu")

      if pipe_upscale is not None:
        pipe_upscale.to("cuda" if inf_mode == modes['upscale4x'] else "cpu")

      if pipe_depth2img is not None:
        pipe_depth2img.to("cuda" if inf_mode == modes['depth2img'] else "cpu")

      current_mode = inf_mode

    if seed == 0:
      seed = random.randint(0, 2147483647)
      print("seed value : " + str(seed))

    generator = torch.Generator('cuda').manual_seed(seed)
    prompt = prompt

    try:

      if inf_mode == modes['txt2img']:
        return txt_to_img(prompt, n_images, neg_prompt, guidance, steps, width, height, generator, seed), gr.update(visible=False, value=None)

      elif inf_mode == modes['img2img']:
        if img is None:
          return None, gr.update(visible=True, value=error_str("Image is required for Image to Image mode"))

        return img_to_img(prompt, n_images, neg_prompt, img, strength, guidance, steps, width, height, generator, seed), gr.update(visible=Fals

      elif inf_mode == modes['inpaint']:
        if img is None:
          return None, gr.update(visible=True, value=error_str("Image is required for Inpainting mode"))

        return inpaint(prompt, n_images, neg_prompt, img, guidance, steps, width, height, generator, seed), gr.update(visible=False, value=None

      elif inf_mode == modes['upscale4x']:
        if img is None:
          return None, gr.update(visible=True, value=error_str("Image is required for Upscale mode"))

        return upscale(prompt, n_images, neg_prompt, img, guidance, steps, generator), gr.update(visible=False, value=None)

      elif inf_mode == modes['depth2img']:
        if img is None:
          return None, gr.update(visible=True, value=error_str("Image is required for Depth to Image mode"))

        return depth2img(prompt, n_images, neg_prompt, img, guidance, steps, generator, seed), gr.update(visible=False, value=None)

    except Exception as e:
      return None, gr.update(visible=True, value=error_str(e))

  def txt_to_img(prompt, n_images, neg_prompt, guidance, steps, width, height, generator, seed):

      result = pipe(
        prompt,
        num_images_per_prompt = n_images,
        negative_prompt = neg_prompt,
        num_inference_steps = int(steps),
        guidance_scale = guidance,
        width = width,
        height = height,
        generator = generator,
        callback=pipe_callback).images

      update_state(f"Done. Seed: {seed}")

      return result

  def img_to_img(prompt, n_images, neg_prompt, img, strength, guidance, steps, width, height, generator, seed):

      global pipe_i2i
      if pipe_i2i is None:
        pipe_i2i = get_i2i_pipe(scheduler)

      img = img['image']
      ratio = min(height / img.height, width / img.width)
      img = img.resize((int(img.width * ratio), int(img.height * ratio)), Image.LANCZOS)
      result = pipe_i2i(
```

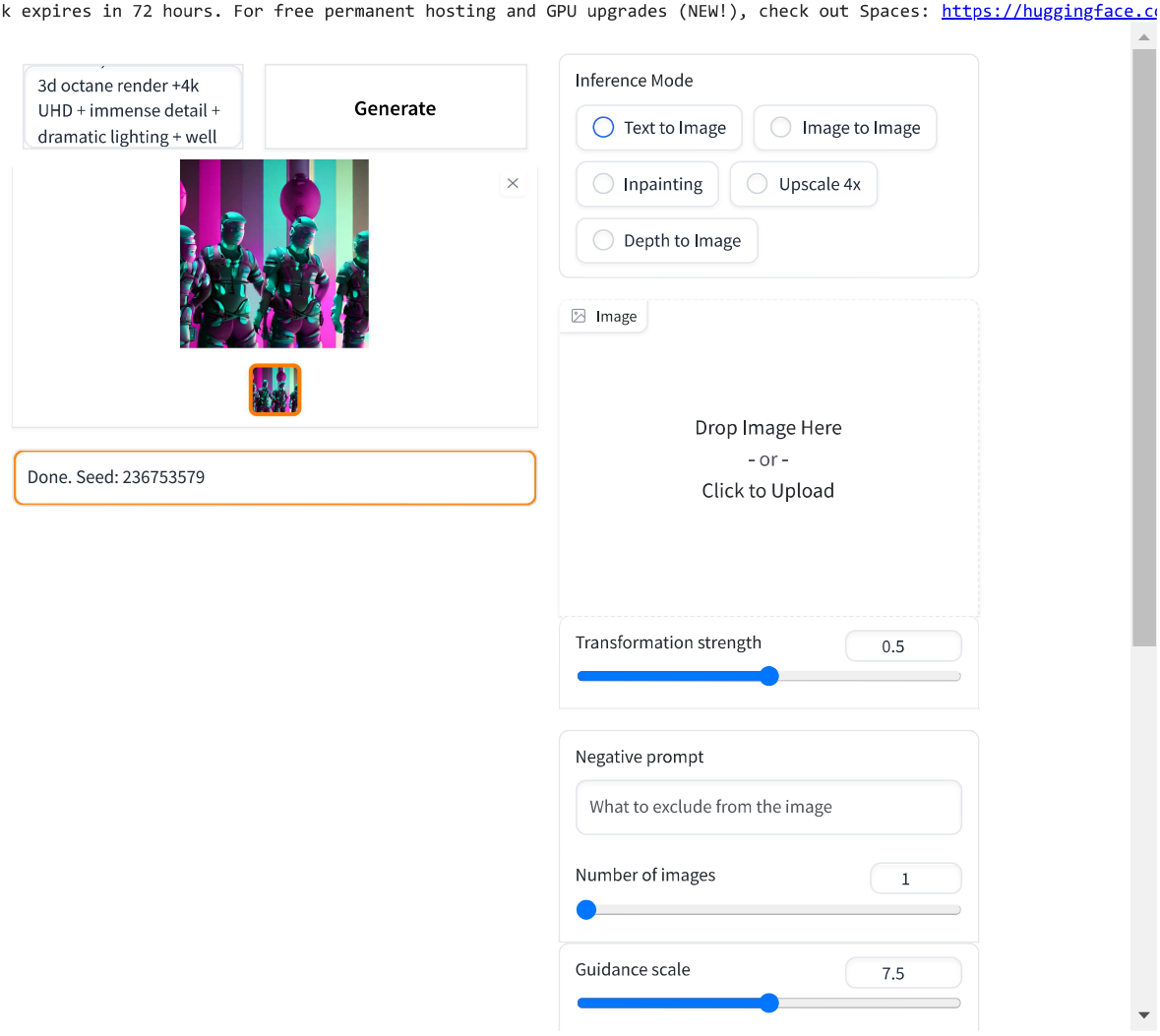
```
        prompt,
        num_images_per_prompt = n_images,
        negative_prompt = neg_prompt,
        image = img,
        num_inference_steps = int(steps),
        strength = strength,
        guidance_scale = guidance,
        # width = width,
        # height = height,
        generator = generator,
        callback=pipe_callback).images

    update_state(f"Done. Seed: {seed}")

    return result

# TODO Currently supports only 512x512 images
def inpaint(prompt, n_images, neg_prompt, img, guidance, steps, width, height, generator, seed):

    global pipe_inpaint
    if pipe_inpaint is None:
      pipe_inpaint = get_inpaint_pipe()

    inp_img = img['image']
    mask = img['mask']
    inp_img = square_padding(inp_img)
    mask = square_padding(mask)

    # # ratio = min(height / inp_img.height, width / inp_img.width)
    # ratio = min(512 / inp_img.height, 512 / inp_img.width)
    # inp_img = inp_img.resize((int(inp_img.width * ratio), int(inp_img.height * ratio)), Image.LANCZOS)
    # mask = mask.resize((int(mask.width * ratio), int(mask.height * ratio)), Image.LANCZOS)

    inp_img = inp_img.resize((512, 512))
    mask = mask.resize((512, 512))

    result = pipe_inpaint(
      prompt,
      image = inp_img,
      mask_image = mask,
      num_images_per_prompt = n_images,
      negative_prompt = neg_prompt,
      num_inference_steps = int(steps),
      guidance_scale = guidance,
      # width = width,
      # height = height,
      generator = generator,
      callback=pipe_callback).images

    update_state(f"Done. Seed: {seed}")

    return result

def depth2img(prompt, n_images, neg_prompt, img, guidance, steps, generator, seed):

    global pipe_depth2img
    if pipe_depth2img is None:
      pipe_depth2img = get_depth2img_pipe()

    img = img['image']
    result = pipe_depth2img(
      prompt,
      num_images_per_prompt = n_images,
      negative_prompt = neg_prompt,
      image = img,
      num_inference_steps = int(steps),
      guidance_scale = guidance,
      # width = width,
      # height = height,
      generator = generator,
      callback=pipe_callback).images

    update_state(f"Done. Seed: {seed}")

    return result

def square_padding(img):
    width, height = img.size
```

```
        width, height = img.size
        if width == height:
            return img
        new_size = max(width, height)
        new_img = Image.new('RGB', (new_size, new_size), (0, 0, 0, 255))
        new_img.paste(img, ((new_size - width) // 2, (new_size - height) // 2))
        return new_img

    def upscale(prompt, n_images, neg_prompt, img, guidance, steps, generator):

        global pipe_upscale
        if pipe_upscale is None:
          pipe_upscale = get_upscale_pipe(scheduler)

        img = img['image']
        return upscale_tiling(prompt, neg_prompt, img, guidance, steps, generator)


    def upscale_tiling(prompt, neg_prompt, img, guidance, steps, generator):

        width, height = img.size

        # calculate the padding needed to make the image dimensions a multiple of 128
        padding_x = 128 - (width % 128) if width % 128 != 0 else 0
        padding_y = 128 - (height % 128) if height % 128 != 0 else 0

        # create a white image of the right size to be used as padding
        padding_img = Image.new('RGB', (padding_x, padding_y), color=(255, 255, 255, 0))

        # paste the padding image onto the original image to add the padding
        img.paste(padding_img, (width, height))

        # update the image dimensions to include the padding
        width += padding_x
        height += padding_y

        if width > 128 or height > 128:

            num_tiles_x = int(width / 128)
            num_tiles_y = int(height / 128)

            upscaled_img = Image.new('RGB', (img.size[0] * 4, img.size[1] * 4))
            for x in range(num_tiles_x):
                for y in range(num_tiles_y):
                    update_state(f"Upscaling tile {x * num_tiles_y + y + 1}/{num_tiles_x * num_tiles_y}")
                    tile = img.crop((x * 128, y * 128, (x + 1) * 128, (y + 1) * 128))

                    upscaled_tile = pipe_upscale(
                        prompt="",
                        image=tile,
                        num_inference_steps=steps,
                        guidance_scale=guidance,
                        # negative_prompt = neg_prompt,
                        generator=generator,
                    ).images[0]

                    upscaled_img.paste(upscaled_tile, (x * upscaled_tile.size[0], y * upscaled_tile.size[1]))

            return [upscaled_img]
        else:
            return pipe_upscale(
                prompt=prompt,
                image=img,
                num_inference_steps=steps,
                guidance_scale=guidance,
                negative_prompt = neg_prompt,
                generator=generator,
            ).images



    def on_mode_change(mode):
      return gr.update(visible = mode in (modes['img2img'], modes['inpaint'], modes['upscale4x'], modes['depth2img'])), \
             gr.update(visible = mode == modes['inpaint']), \
             gr.update(visible = mode == modes['upscale4x']), \
             gr.update(visible = mode == modes['img2img'])

    def on_steps_change(steps):
```

```
    global current_steps
    current_steps = steps

with gr.Blocks() as demo:

    with gr.Row():

        with gr.Column(scale=70):
          with gr.Group():
              with gr.Row():
                prompt = gr.Textbox(label="Prompt", show_label=False, max_lines=2,placeholder=f"Enter prompt").style(container=False)
                generate = gr.Button(value="Generate").style(rounded=(False, True, True, False))

              gallery = gr.Gallery(label="Generated images", show_label=False).style(grid=[2], height="auto")
          state_info = gr.Textbox(label="State", show_label=False, max_lines=2).style(container=False)
          error_output = gr.Markdown(visible=False)

        with gr.Column(scale=30):
          inf_mode = gr.Radio(label="Inference Mode", choices=list(modes.values()), value=modes['txt2img'])

          with gr.Group(visible=False) as i2i_options:
            image = gr.Image(label="Image", height=128, type="pil", tool='sketch')
            inpaint_info = gr.Markdown("Inpainting resizes and pads images to 512x512", visible=False)
            upscale_info = gr.Markdown("""Best for small images (128x128 or smaller).<br>
                                          Bigger images will be sliced into 128x128 tiles which will be upscaled individually.<br>
                                          This is done to avoid running out of GPU memory.""", visible=False)
            strength = gr.Slider(label="Transformation strength", minimum=0, maximum=1, step=0.01, value=0.5)

          with gr.Group():
            neg_prompt = gr.Textbox(label="Negative prompt", placeholder="What to exclude from the image")

            n_images = gr.Slider(label="Number of images", value=1, minimum=1, maximum=4, step=1)
            with gr.Row():
              guidance = gr.Slider(label="Guidance scale", value=7.5, maximum=15)
              steps = gr.Slider(label="Steps", value=current_steps, minimum=2, maximum=100, step=1)

            with gr.Row():
              width = gr.Slider(label="Width", value=768, minimum=64, maximum=1024, step=8)
              height = gr.Slider(label="Height", value=768, minimum=64, maximum=1024, step=8)

            seed = gr.Slider(0, 2147483647, label='Seed (0 = random)', value=0, step=1)
            with gr.Accordion("Memory optimization"):
              attn_slicing = gr.Checkbox(label="Attention slicing (a bit slower, but uses less memory)", value=attn_slicing_enabled)
              # mem_eff_attn = gr.Checkbox(label="Memory efficient attention (xformers)", value=mem_eff_attn_enabled)

    inf_mode.change(on_mode_change, inputs=[inf_mode], outputs=[i2i_options, inpaint_info, upscale_info, strength], queue=False)
    steps.change(on_steps_change, inputs=[steps], outputs=[], queue=False)
    attn_slicing.change(lambda x: switch_attention_slicing(x), inputs=[attn_slicing], queue=False)
    # mem_eff_attn.change(lambda x: switch_mem_eff_attn(x), inputs=[mem_eff_attn], queue=False)

    inputs = [inf_mode, prompt, n_images, guidance, steps, width, height, seed, image, strength, neg_prompt]
    outputs = [gallery, error_output]
    prompt.submit(inference, inputs=inputs, outputs=outputs)
    generate.click(inference, inputs=inputs, outputs=outputs)

    demo.load(update_state_info, inputs=state_info, outputs=state_info, every=0.5, show_progress=False)

demo.queue()
demo.launch(debug=True, share=True, height=768)
```

...

```
Downloading (…)cheduler_config.json: 100%                 345/345 [00:00<00:00, 11.7kB/s]
Downloading (…)p16/model_index.json: 100%                 517/517 [00:00<00:00, 8.73kB/s]
Fetching 12 files: 100%                                   12/12 [00:21<00:00, 2.19s/it]
Downloading (…)"pytorch_model.bin";: 100%                 681M/681M [00:08<00:00, 99.6MB/s]
Downloading (…)_encoder/config.json: 100%                 628/628 [00:00<00:00, 4.48kB/s]
Downloading (…)okenizer_config.json: 100%                 819/819 [00:00<00:00, 6.07kB/s]
Downloading (…)cheduler_config.json: 100%                 351/351 [00:00<00:00, 1.56kB/s]
Downloading (…)cial_tokens_map.json: 100%                 460/460 [00:00<00:00, 2.21kB/s]
Downloading (…)tokenizer/merges.txt: 100%                 525k/525k [00:00<00:00, 920kB/s]
Downloading (…)75a/unet/config.json: 100%                 999/999 [00:00<00:00, 3.90kB/s]
Downloading (…)tokenizer/vocab.json: 100%                 1.06M/1.06M [00:00<00:00, 1.61MB/s]
Downloading (…)_pytorch_model.bin";: 100%                 1.73G/1.73G [00:19<00:00, 96.0MB/s]
Downloading (…)_pytorch_model.bin";: 100%                 167M/167M [00:02<00:00, 60.2MB/s]
Downloading (…)d75a/vae/config.json: 100%                 612/612 [00:00<00:00, 4.93kB/s]
/usr/local/lib/python3.8/dist-packages/gradio/components.py:122: UserWarning: 'rounded' styling is no longer supported. To round
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/gradio/deprecation.py:40: UserWarning: `height` is deprecated in `Interface()`, please use
  warnings.warn(value)
Colab notebook detected. This cell will run indefinitely so that you can see errors and logs. To turn off, set debug=False in lau
Running on public URL: https://f9edbb8f-60ae-4c63.gradio.live

This share link expires in 72 hours. For free permanent hosting and GPU upgrades (NEW!), check out Spaces: https://huggingface.c
```




```
seed value : 1815538119
100%                                      25/25 [00:24<00:00, 1.18it/s]
seed value : 236753579
100%                                      25/25 [00:21<00:00, 1.16it/s]
```

Executing (48s) Cell › launch() › block_thread()