Aim of the project

Recreate one or more images provided to match exact proportions, composition, and resolution as closely as possible using any tool, software, or generative AI platforms.

Getting the proportions, composition, and resolution of the given images

```
!pip install opencv-python numpy pillow matplotlib

```

```
Requirement already satisfied: opencv-python in /usr/local/lib/python3.11/dist-packages (4.10.0.84)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (1.26.4)
Requirement already satisfied: pillow in /usr/local/lib/python3.11/dist-packages (11.1.0)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (3.10.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (4.55.7)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.4.8)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (24.2)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (3.2.1)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.7->matplotlib) (1.17.0)
```

```
from PIL import Image

# Function to get image details
def get_image_details(image_path):
    """Extracts resolution, aspect ratio, mode, and number of channels from an image."""
    img = Image.open(image_path)

    width, height = img.size
    proportions = round(width / height, 2)  # Aspect ratio
    mode = img.mode
    channels = len(img.getbands())  # Number of channels (e.g., RGB = 3)

    return {
        "Resolution": f"{width} x {height}",
        "Aspect Ratio": proportions,
        "Mode": mode,
        "Channels": channels
    }


```

For the given image 1

```
# Example usage with an image
image_path = "/content/Given_pic_1.jpg"  # Update with your actual path
details_given_image_1 = get_image_details(image_path)

# Print the stored details
details_given_image_1
```

```
{'Resolution': '624 x 807', 'Aspect Ratio': 0.77, 'Mode': 'RGB', 'Channels': 3}
```

For the given image 2

```
# Example usage with an image
image_path = "/content/Given_pic_2.jpg"  # Update with your actual path
details_given_image_2 = get_image_details(image_path)

# Print the stored details
print(details_given_image_2)
```

```
{'Resolution': '624 x 624', 'Aspect Ratio': 1.0, 'Mode': 'RGB', 'Channels': 3}
```

For the given image 3

```
# Example usage with an image
```

```
image_path = "/content/Given_pic_3.jpg"  # Update with your actual path
details_given_image_3 = get_image_details(image_path)

# Print the stored details
print(details_given_image_3)
```

```
{'Resolution': '624 x 624', 'Aspect Ratio': 1.0, 'Mode': 'RGB', 'Channels': 3}
```

COT 1: Image to text and then text to image

We used BLIP to generate the image description

```
from transformers import BlipProcessor, BlipForConditionalGeneration
from PIL import Image

# Load BLIP processor and model
processor = BlipProcessor.from_pretrained("Salesforce/blip-image-captioning-base")
model = BlipForConditionalGeneration.from_pretrained("Salesforce/blip-image-captioning-base")
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as s
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
preprocessor_config.json: 100%  287/287 [00:00<00:00, 25.4kB/s]
tokenizer_config.json: 100%  506/506 [00:00<00:00, 50.6kB/s]
vocab.txt: 100%  232k/232k [00:00<00:00, 1.76MB/s]
tokenizer.json: 100%  711k/711k [00:00<00:00, 3.52MB/s]
special_tokens_map.json: 100%  125/125 [00:00<00:00, 10.3kB/s]
config.json: 100%  4.56k/4.56k [00:00<00:00, 430kB/s]
pytorch_model.bin: 100%  990M/990M [00:04<00:00, 234MB/s]
```

BLIP desciption for given image 1

```
# Load the image
image_path = "/content/Given_pic_1.jpg"
image = Image.open(image_path)

# Generate a caption
inputs = processor(images=image, return_tensors="pt")
out = model.generate(**inputs)

# Decode and print the caption
caption_1 = processor.decode(out[0], skip_special_tokens=True)
print("Generated Caption for given image 1: ", caption_1)

```

```
Generated Caption for given image 1:  a painting of a woman in a blue bikini
```

BLIP desciption for given image 2

```
# Load the image
image_path = "/content/Given_pic_2.jpg"
image = Image.open(image_path)

# Generate a caption
inputs = processor(images=image, return_tensors="pt")
out = model.generate(**inputs)

# Decode and print the caption
caption_2 = processor.decode(out[0], skip_special_tokens=True)
print("Generated Caption for given image 1: ", caption_2)

```

```
Generated Caption for given image 1:  a painting of a man with a face full of blood
```

BLIP desciption for given image 3

```
# Load the image
image_path = "/content/Given_pic_3.jpg"
image = Image.open(image_path)

# Generate a caption
inputs = processor(images=image, return_tensors="pt")
out = model.generate(**inputs)

# Decode and print the caption
caption_3 = processor.decode(out[0], skip_special_tokens=True)
print("Generated Caption for given image 1: ", caption_3)

```

```
Generated Caption for given image 1:  a painting of a woman with a black cat
```

Conclusion

This COT was dropped as an acceptable level of image description was not generated.

COT 2: Image to image

```
!python --version
!nvcc --version
!pip install nvcc4jupyter
%load_ext nvcc4jupyter
```

```
Python 3.11.11
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2024 NVIDIA Corporation
Built on Thu_Jun__6_02:18:23_PDT_2024
Cuda compilation tools, release 12.5, V12.5.82
Build cuda_12.5.r12.5/compiler.34385749_0
Collecting nvcc4jupyter
  Downloading nvcc4jupyter-1.2.1-py3-none-any.whl.metadata (5.1 kB)
Downloading nvcc4jupyter-1.2.1-py3-none-any.whl (10 kB)
Installing collected packages: nvcc4jupyter
Successfully installed nvcc4jupyter-1.2.1
Detected platform "Colab". Running its setup...
Source files will be saved in "/tmp/tmpb_n64q4i".
```

Using diffusers from Hugging Face.

The most popular image-to-image models are **Stable Diffusion v1.5, Stable Diffusion XL (SDXL), and Kandinsky 2.2**.

Prompt

The prompt was kept simple to recreate the given image and fine tuning was done on the following parameters.

Strength

This parameter determines how much the model should alter the initial image when generating the final result. A higher value will result in more significant changes to the image, whereas a lower value will keep the original image closer to its initial state.

The value was kept low as we wanted the replica of the given image.

guidance_scale

This parameter controls the strength of the guidance from the prompt during the image generation process. A higher value forces the model to adhere more strictly to the prompt, making the generated image more faithful to the description, but it may reduce creativity or diversity in the result. A lower value allows for more freedom in the generation, which may lead to more unexpected or diverse results.

The value was kept high as we wanted the replica of the given image.

```
!pip install transformers accelerate torch

!pip install git+https://github.com/huggingface/diffusers.git
```

Show hidden output

kandinsky-2-2 pipeline

```
import torch
from diffusers import AutoPipelineForImage2Image
from diffusers.utils import make_image_grid, load_image
```

```
4
5 pipeline_kandinsky = AutoPipelineForImage2Image.from_pretrained(
6     "kandinsky-community/kandinsky-2-2-decoder", torch_dtype=torch.float16, use_safetensors=True
7 )
8 pipeline_kandinsky.enable_model_cpu_offload()
9
```

```
The cache for model files in Transformers v4.22.0 has been updated. Migrating your old cache. This is a one-time only operation. You
     0/0 [00:00<?, ?it/s]
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as s
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
model_index.json: 100%  250/250 [00:00<00:00, 850B/s]
Fetching 7 files: 100%  7/7 [00:24<00:00, 4.23s/it]
movq/config.json: 100%  660/660 [00:00<00:00, 24.7kB/s]
scheduler/scheduler_config.json: 100%  317/317 [00:00<00:00, 7.68kB/s]
unet/config.json: 100%  1.67k/1.67k [00:00<00:00, 68.1kB/s]
README.md: 100%  7.34k/7.34k [00:00<00:00, 140kB/s]
diffusion_pytorch_model.safetensors: 100%  271M/271M [00:01<00:00, 187MB/s]
diffusion_pytorch_model.safetensors: 100%  5.01G/5.01G [00:23<00:00, 235MB/s]
Loading pipeline components...: 100%  3/3 [00:01<00:00, 3.69it/s]
model_index.json: 100%  501/501 [00:00<00:00, 43.1kB/s]
Fetching 13 files: 100%  13/13 [00:40<00:00, 5.50s/it]
diffusion_pytorch_model.safetensors: 100%  4.10G/4.10G [00:39<00:00, 236MB/s]
model.safetensors: 100%  2.78G/2.78G [00:35<00:00, 59.3MB/s]
prior/config.json: 100%  252/252 [00:00<00:00, 5.98kB/s]
image_processor/preprocessor_config.json: 100%  315/315 [00:00<00:00, 8.55kB/s]
text_encoder/config.json: 100%  2.02k/2.02k [00:00<00:00, 31.5kB/s]
scheduler/scheduler_config.json: 100%  229/229 [00:00<00:00, 4.24kB/s]
model.safetensors: 100%  3.69G/3.69G [00:38<00:00, 240MB/s]
image_encoder/config.json: 100%  2.01k/2.01k [00:00<00:00, 109kB/s]
tokenizer/merges.txt: 100%  525k/525k [00:00<00:00, 6.29MB/s]
tokenizer/vocab.json: 100%  862k/862k [00:00<00:00, 4.31MB/s]
tokenizer/special_tokens_map.json: 100%  389/389 [00:00<00:00, 29.6kB/s]
tokenizer/tokenizer_config.json: 100%  904/904 [00:00<00:00, 50.4kB/s]
Loading pipeline components...: 100%  6/6 [00:01<00:00, 2.34it/s]
```

Stable Diffusion v1.5 pipeline

```
1 pipeline_sd = AutoPipelineForImage2Image.from_pretrained(
2     "stable-diffusion-v1-5/stable-diffusion-v1-5", torch_dtype=torch.float16, variant="fp16",safety_checker=None, use_safetensors=Tru
3 )
4 pipeline_sd.enable_model_cpu_offload()
5
```

```
model_index.json: 100% 541/541 [00:00<00:00, 49.4kB/s]
Fetching 13 files: 100% 13/13 [00:10<00:00, 1.08s/it]
tokenizer/tokenizer_config.json: 100% 806/806 [00:00<00:00, 17.9kB/s]
tokenizer/special_tokens_map.json: 100% 472/472 [00:00<00:00, 14.5kB/s]
model.fp16.safetensors: 100% 246M/246M [00:01<00:00, 185MB/s]
(…)ature_extractor/preprocessor_config.json: 100% 342/342 [00:00<00:00, 5.36kB/s]
tokenizer/merges.txt: 100% 525k/525k [00:00<00:00, 8.06MB/s]
scheduler/scheduler_config.json: 100% 308/308 [00:00<00:00, 6.22kB/s]
tokenizer/vocab.json: 100% 1.06M/1.06M [00:00<00:00, 5.22MB/s]
diffusion_pytorch_model.fp16.safetensors: 100% 167M/167M [00:01<00:00, 139MB/s]
diffusion_pytorch_model.fp16.safetensors: 100% 1.72G/1.72G [00:09<00:00, 187MB/s]
vae/config.json: 100% 547/547 [00:00<00:00, 20.7kB/s]
unet/config.json: 100% 743/743 [00:00<00:00, 31.9kB/s]
text_encoder/config.json: 100% 617/617 [00:00<00:00, 51.0kB/s]
Loading pipeline components...: 100% 6/6 [00:00<00:00, 5.22it/s]
You have disabled the safety checker for <class 'diffusers.pipelines.stable_diffusion.pipeline_stable_diffusion_img2img.StableDiffus
```

Stable Diffusion XL (SDXL) pipeline

```
pipeline_sdxl = AutoPipelineForImage2Image.from_pretrained(
    "stabilityai/stable-diffusion-xl-refiner-1.0", torch_dtype=torch.float16, variant="fp16", use_safetensors=True
)
pipeline_sdxl.enable_model_cpu_offload()
```

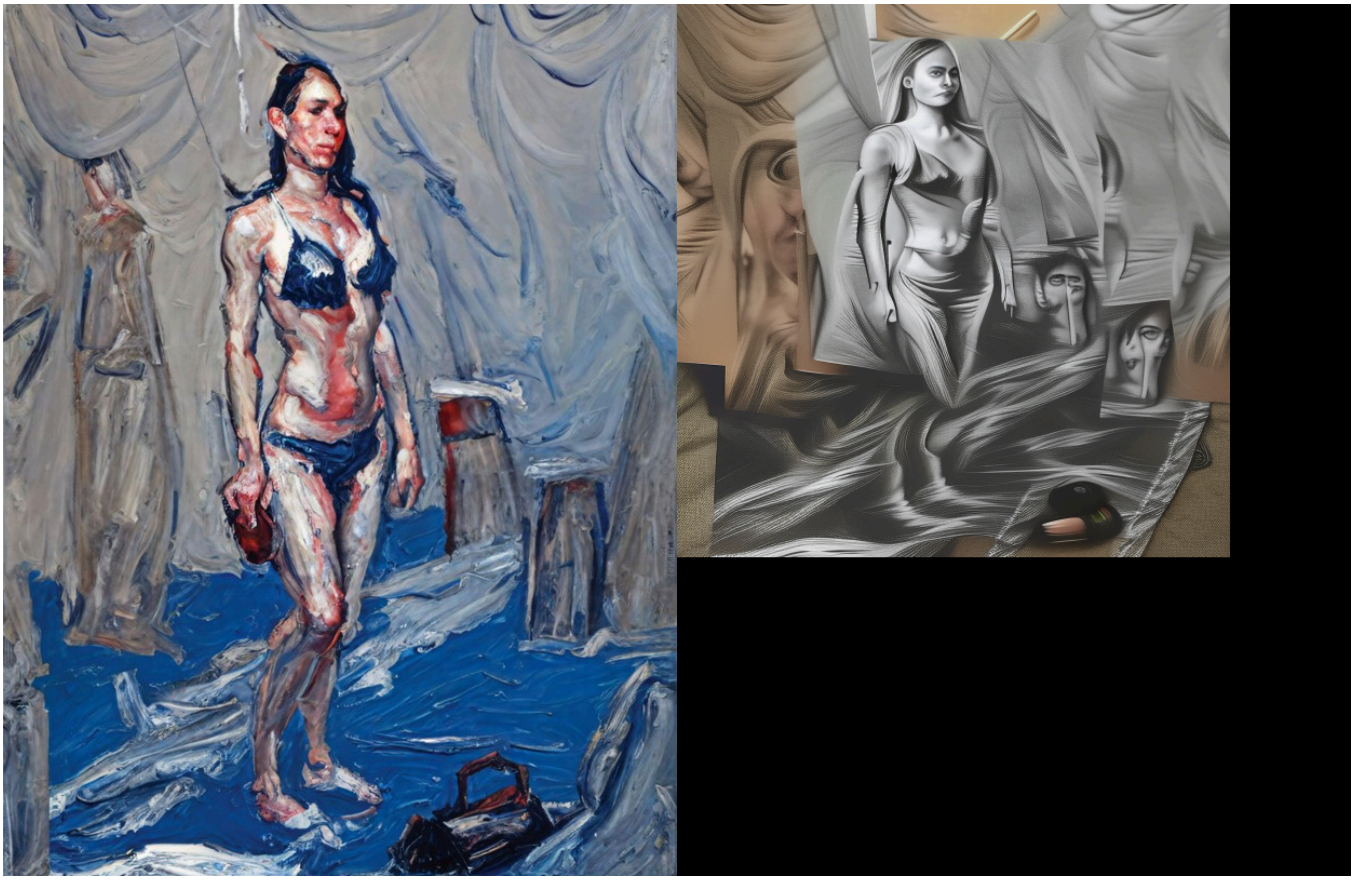
```
model_index.json: 100% 612/612 [00:00<00:00, 55.4kB/s]
Fetching 13 files: 100% 13/13 [00:29<00:00, 3.16s/it]
scheduler/scheduler_config.json: 100% 479/479 [00:00<00:00, 20.8kB/s]
text_encoder_2/config.json: 100% 575/575 [00:00<00:00, 45.5kB/s]
model.fp16.safetensors: 100% 1.39G/1.39G [00:09<00:00, 127MB/s]
tokenizer_2/vocab.json: 100% 1.06M/1.06M [00:00<00:00, 5.40MB/s]
unet/config.json: 100% 1.71k/1.71k [00:00<00:00, 36.4kB/s]
tokenizer_2/tokenizer_config.json: 100% 725/725 [00:00<00:00, 19.5kB/s]
tokenizer_2/merges.txt: 100% 525k/525k [00:00<00:00, 4.02MB/s]
tokenizer_2/special_tokens_map.json: 100% 460/460 [00:00<00:00, 14.8kB/s]
diffusion_pytorch_model.fp16.safetensors: 100% 4.52G/4.52G [00:29<00:00, 173MB/s]
diffusion_pytorch_model.fp16.safetensors: 100% 167M/167M [00:00<00:00, 244MB/s]
vae/config.json: 100% 642/642 [00:00<00:00, 50.6kB/s]
diffusion_pytorch_model.fp16.safetensors: 100% 167M/167M [00:01<00:00, 183MB/s]
Loading pipeline components...: 100% 5/5 [00:00<00:00, 5.85it/s]
```

Given image 1

kandinsky-2-2

```
url = "/content/Given_pic_1.jpg"
init_image = load_image(url)

prompt = "Make an exact copy of the given image with no changes or creativity "

gen_image_kandinsky_1 = pipeline_kandinsky(prompt, image=init_image, strength=0.1, guidance_scale=10.0).images[0]
make_image_grid([init_image, gen_image_kandinsky_1], rows=1, cols=2)
```

```
100%                                                 25/25 [00:01<00:00, 31.17it/s]
/usr/local/lib/python3.11/dist-packages/diffusers/pipelines/kandinsky2_2/pipeline_kandinsky2_2_img2img.py:276: FutureWarning: `callb
  deprecate(
100%                                                 10/10 [00:01<00:00,  8.66it/s]
```



```
# Save the generated image
gen_image_kandinsky_1.save("/content/gen_image_kandinsky_1.jpg")
```

Stable Diffusion v1.5

```
import torch
from diffusers import AutoPipelineForImage2Image
from diffusers.utils import make_image_grid, load_image

url = "/content/Given_pic_1.jpg"
init_image = load_image(url)

prompt = "Make an exact copy of the given image with no changes or creativity "

gen_image_sd_1 = pipeline_sd(prompt, image=init_image, strength=0.1, guidance_scale=10.0).images[0]
make_image_grid([init_image, gen_image_sd_1], rows=1, cols=2)
```

100% 5/5 [00:01<00:00, 4.59it/s]

```
# Save the generated image
gen_image_sd_1.save("/content/gen_image_sd_1.1.jpg")
```

Stable Diffusion XL (SDXL)

```
url = "/content/Given_pic_1.jpg"
init_image = load_image(url)

prompt = "Make an exact copy of the given image with no changes or creativity "

gen_image_sdxl_1 = pipeline_sdxl(prompt, image=init_image, strength=0.1, guidance_scale=10.0).images[0]
make_image_grid([init_image, gen_image_sdxl_1 ], rows=1, cols=2)
```

100% 5/5 [00:03<00:00, 2.41it/s]

```
# Save the generated image
gen_image_sdxl_1.save("/content/gen_image_sdxl_1.1.jpg")
```

Given image 2

kandinsky-2-2

```
import torch
from diffusers import AutoPipelineForImage2Image
from diffusers.utils import make_image_grid, load_image


url = "/content/Given_pic_2.jpg"
init_image = load_image(url)

prompt = "Make an exact copy of the given image with no changes or creativity "

gen_image_kandinsky_2 = pipeline_kandinsky(prompt, image=init_image, strength=0.1, guidance_scale=10.0).images[0]
make_image_grid([init_image, gen_image_kandinsky_2], rows=1, cols=2)
```

```
100%  25/25 [00:01<00:00, 31.60it/s]
100%  10/10 [00:01<00:00,  9.45it/s]
```

```
# Save the generated image
gen_image_kandinsky_2.save("/content/gen_image_kandinsky_2.jpg")
```

Stable Diffusion v1.5

```
import torch
from diffusers import AutoPipelineForImage2Image
from diffusers.utils import make_image_grid, load_image


url = "/content/Given_pic_2.jpg"
init_image = load_image(url)

prompt = "Make an exact copy of the given image with no changes or creativity "

gen_image_sd_2 = pipeline_sd(prompt, image=init_image, strength=0.1, guidance_scale=10.0).images[0]
make_image_grid([init_image, gen_image_sd_2], rows=1, cols=2)
```

100% 5/5 [00:01<00:00, 5.51it/s]

```
# Save the generated image
gen_image_sd_2.save("/content/gen_image_sd_2.jpg")
```

Stable Diffusion XL (SDXL)

```
url = "/content/Given_pic_2.jpg"
init_image = load_image(url)

prompt = "Make an exact copy of the given image with no changes or creativity "

gen_image_sdxl_2 = pipeline_sdxl(prompt, image=init_image, strength=0.1, guidance_scale=10.0).images[0]
make_image_grid([init_image, gen_image_sdxl_2], rows=1, cols=2)
```

100% 5/5 [00:02<00:00, 3.28it/s]

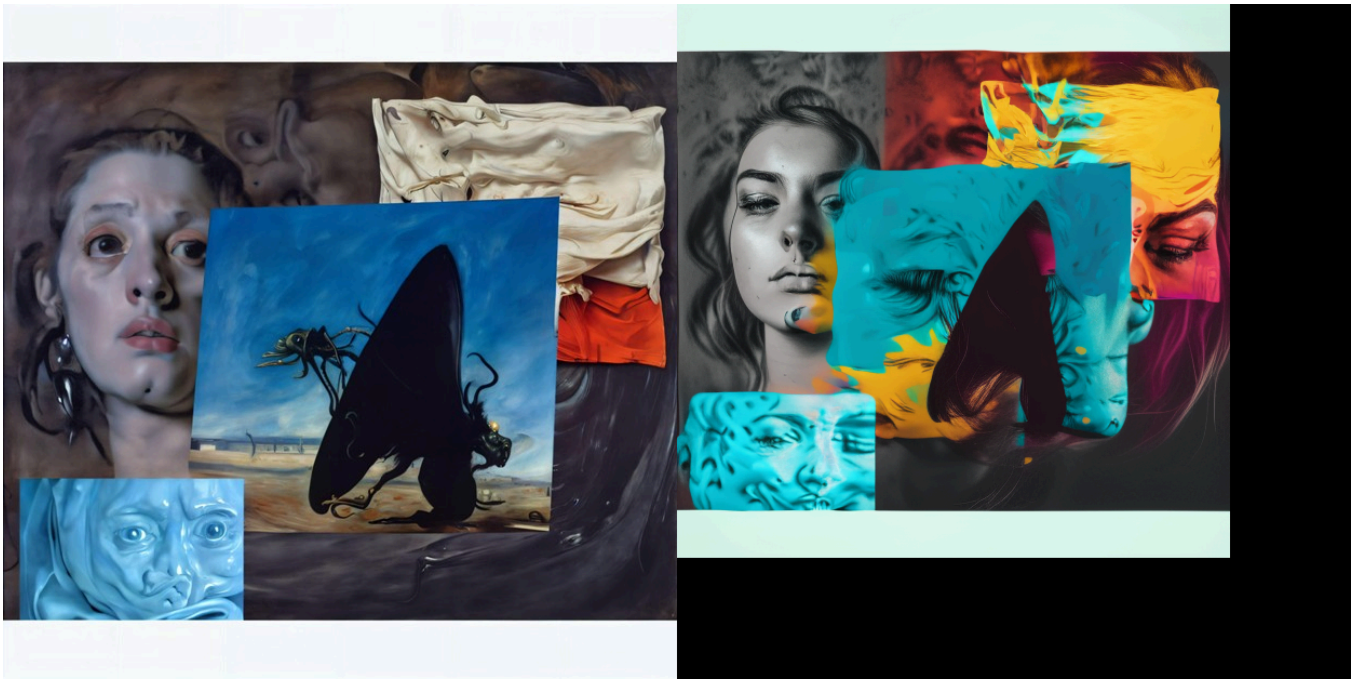
```
# Save the generated image
gen_image_sdxl_2.save("/content/gen_image_sdxl_2.jpg")
```

Given image 3

kandinsky-2-2

```
import torch
from diffusers import AutoPipelineForImage2Image
from diffusers.utils import make_image_grid, load_image


url = "/content/Given_pic_3.jpg"
init_image = load_image(url)

prompt = "Make an exact copy of the given image with no changes or creativity "

gen_image_kandinsky_3 = pipeline_kandinsky(prompt, image=init_image, strength=0.1, guidance_scale=10.0).images[0]
make_image_grid([init_image, gen_image_kandinsky_3], rows=1, cols=2)
```

```
100%  25/25 [00:01<00:00, 30.67it/s]
100%  10/10 [00:01<00:00,  9.37it/s]
```



```
# Save the generated image
gen_image_kandinsky_3.save("/content/gen_image_kandinsky_3.jpg")
```

Stable Diffusion v1.5

```
import torch
from diffusers import AutoPipelineForImage2Image
from diffusers.utils import make_image_grid, load_image

url = "/content/Given_pic_3.jpg"
init_image = load_image(url)

prompt = "Make an exact copy of the given image with no changes or creativity "

gen_image_sd_3 = pipeline_sd(prompt, image=init_image, strength=0.1, guidance_scale=10.0).images[0]
make_image_grid([init_image, gen_image_sd_3], rows=1, cols=2)
```

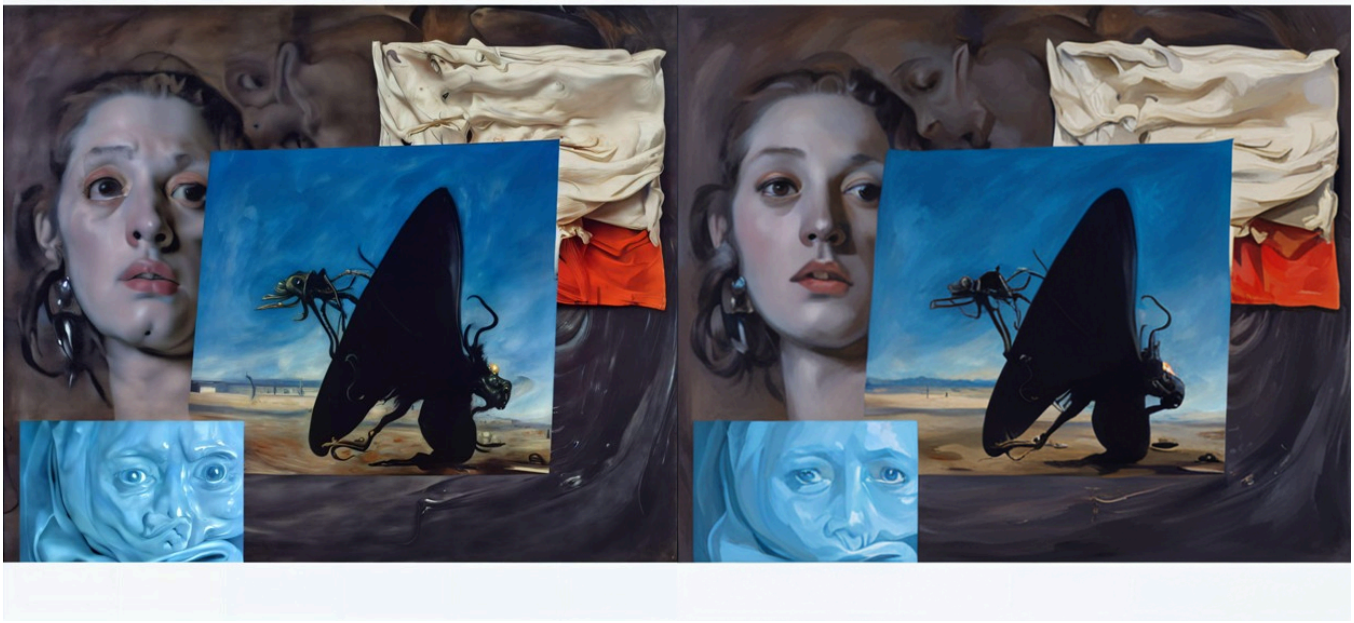
100% 5/5 [00:01<00:00, 5.61it/s]

```
# Save the generated image
gen_image_sd_3.save("/content/gen_image_sd_3.jpg")
```

Stable Diffusion XL (SDXL)

```
url = "/content/Given_pic_3.jpg"
init_image = load_image(url)

prompt = "Make an exact copy of the given image with no changes or creativity "

gen_image_sdxl_3 = pipeline_sdxl(prompt, image=init_image, strength=0.1, guidance_scale=10.0).images[0]
make_image_grid([init_image, gen_image_sdxl_3], rows=1, cols=2)
```

```
100%  5/5 [00:02<00:00,  3.24it/s]
```



```
# Save the generated image
gen_image_sdxl_3.save("/content/gen_image_sdxl_3.jpg")
```

Using instruct-pix2pix

```
import torch
from diffusers import StableDiffusionInstructPix2PixPipeline, EulerAncestralDiscreteScheduler
from PIL import Image

# Load the model and pipeline
model_id = "timbrooks/instruct-pix2pix"
pipe = StableDiffusionInstructPix2PixPipeline.from_pretrained(model_id, torch_dtype=torch.float16, safety_checker=None)
pipe.to("cuda")

# Set up the scheduler
pipe.scheduler = EulerAncestralDiscreteScheduler.from_config(pipe.scheduler.config)
```

```
model_index.json: 100%  616/616 [00:00<00:00, 43.4kB/s]
Fetching 13 files: 100%  13/13 [00:16<00:00,  1.90s/it]
(…)ature_extractor/preprocessor_config.json: 100%  518/518 [00:00<00:00, 28.6kB/s]
tokenizer/merges.txt: 100%  525k/525k [00:00<00:00, 4.61MB/s]
tokenizer/vocab.json: 100%  1.06M/1.06M [00:00<00:00, 7.52MB/s]
tokenizer/tokenizer_config.json: 100%  806/806 [00:00<00:00, 32.7kB/s]
model.safetensors: 100%  492M/492M [00:02<00:00, 208MB/s]
scheduler/scheduler_config.json: 100%  569/569 [00:00<00:00, 21.4kB/s]
tokenizer/special_tokens_map.json: 100%  472/472 [00:00<00:00, 13.6kB/s]
unet/config.json: 100%  1.02k/1.02k [00:00<00:00, 35.8kB/s]
text_encoder/config.json: 100%  617/617 [00:00<00:00, 25.1kB/s]
vae/config.json: 100%  553/553 [00:00<00:00, 45.1kB/s]
diffusion_pytorch_model.safetensors: 100%  3.44G/3.44G [00:16<00:00, 208MB/s]
diffusion_pytorch_model.safetensors: 100%  335M/335M [00:01<00:00, 241MB/s]
Loading pipeline components...: 100%  6/6 [00:01<00:00,  2.49it/s]
```

Given image 1

```
import matplotlib.pyplot as plt
from PIL import Image

# Load the original image
image_path = "/content/Given_pic_1.jpg"  # Update this with your actual path
image = Image.open(image_path)

# Apply the transformation with the instruction
instruction = "make an exact replica of this image"
images_pix2pix_1 = pipe(instruction, image=image).images

# Create a figure with 1 row and 2 columns
fig, axes = plt.subplots(1, 2, figsize=(10, 5))

# Display the original image
axes[0].imshow(image)
axes[0].axis('off')  # Hide axes for better view

# Display the generated image (access the first image in the list)
axes[1].imshow(images_pix2pix_1[0])  # Access the image from the list
axes[1].axis('off')  # Hide axes for better view

# Show the grid of images
plt.tight_layout()
plt.show()

```

```
100%  100/100 [00:18<00:00,  5.50it/s]
```

```
# Save the generated image (save the first image in the list)
generated_image = images_pix2pix_1[0]
generated_image.save("/content/images_pix2pix_1.1.jpg")
```

Given image 2

```
import matplotlib.pyplot as plt
from PIL import Image

# Load the original image
image_path = "/content/Given_pic_2.jpg"  # Update this with your actual path
image = Image.open(image_path)

# Apply the transformation with the instruction
instruction = "make an exact replica of this image"
images_pix2pix_2 = pipe(instruction, image=image).images

# Create a figure with 1 row and 2 columns
fig, axes = plt.subplots(1, 2, figsize=(10, 5))

```

```
# Display the original image
axes[0].imshow(image)
axes[0].axis('off')  # Hide axes for better view

# Display the generated image (access the first image in the list)
axes[1].imshow(images_pix2pix_2[0])  # Access the image from the list
axes[1].axis('off')  # Hide axes for better view

# Show the grid of images
plt.tight_layout()
plt.show()

```

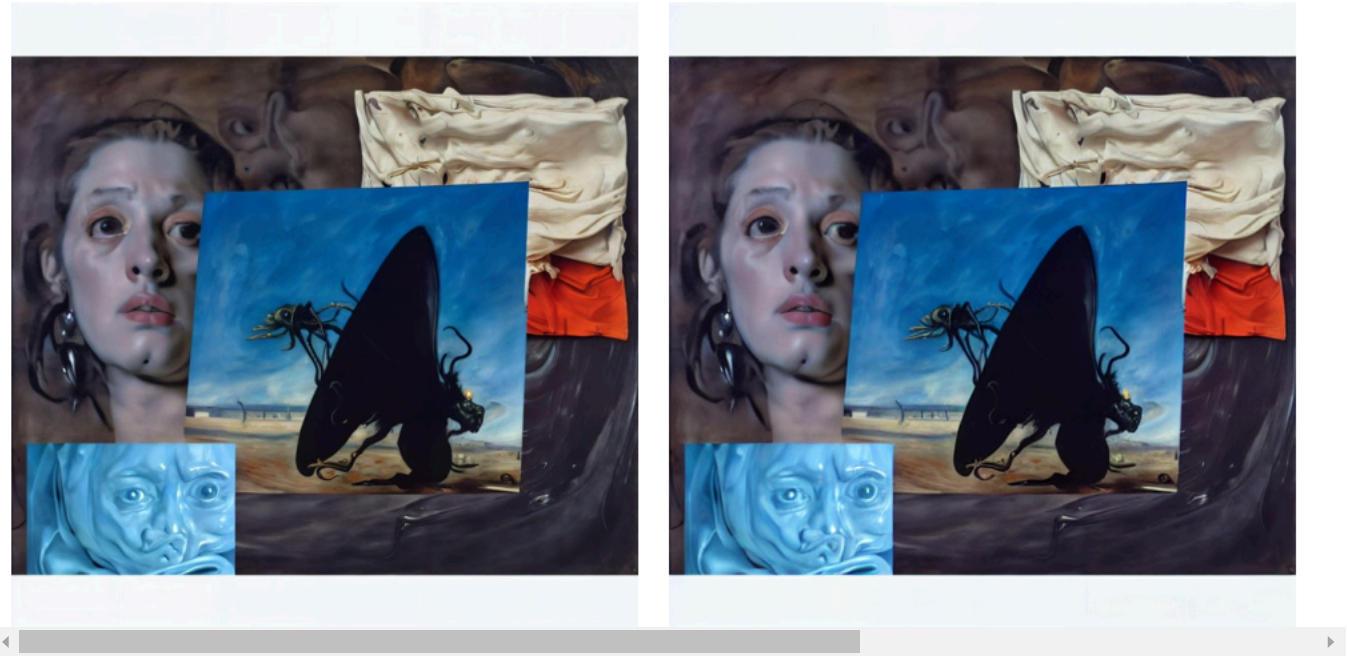
```
100%  100/100 [00:13<00:00,  7.24it/s]
```

```
# Save the generated image (save the first image in the list)
generated_image = images_pix2pix_2[0]
generated_image.save("/content/images_pix2pix_2.jpg")
```

Given image 3

```
import matplotlib.pyplot as plt
from PIL import Image

# Load the original image
image_path = "/content/Given_pic_3.jpg"  # Update this with your actual path
image = Image.open(image_path)

# Apply the transformation with the instruction
instruction = "make an exact replica of this image"
images_pix2pix_3 = pipe(instruction, image=image).images

# Create a figure with 1 row and 2 columns
fig, axes = plt.subplots(1, 2, figsize=(10, 5))

# Display the original image
axes[0].imshow(image)
axes[0].axis('off')  # Hide axes for better view

# Display the generated image (access the first image in the list)
axes[1].imshow(images_pix2pix_3[0])  # Access the image from the list
axes[1].axis('off')  # Hide axes for better view

# Show the grid of images
plt.tight_layout()
plt.show()

```

```
100%  100/100 [00:13<00:00,  7.16it/s]
```



```
# Save the generated image (save the first image in the list)
generated_image = images_pix2pix_3[0]
generated_image.save("/content/images_pix2pix_3.jpg")
```

Even after fine-tuning, the Kandinsky-2-2 model still fails to produce results that are close to the original image. Given this, we have decided to exclude it from further evaluation. We will continue with the Stable Diffusion v1.5 and Stable Diffusion XL (SDXL) models moving forward.

Evaluation of generated images

```
from PIL import Image
from tabulate import tabulate  # To format output as a table

# Function to extract image details
def get_image_details(image, image_name):
    """Extracts resolution, aspect ratio, mode, and number of channels from an image."""
    width, height = image.size
    proportions = round(width / height, 2)  # Aspect ratio
    mode = image.mode
    channels = len(image.getbands())  # Number of channels (e.g., RGB = 3)

    return [image_name, f"{width} x {height}", proportions, mode, channels]

# Assuming 'images_pix2pix_1', 'images_pix2pix_2', etc. are already loaded or defined
# Store details for each image in a list
image_details = [
    get_image_details(images_pix2pix_1[0], "Pix2Pix Image 1"),
    get_image_details(images_pix2pix_2[0], "Pix2Pix Image 2"),
    get_image_details(images_pix2pix_3[0], "Pix2Pix Image 3"),
    get_image_details(gen_image_sd_1, "SD Image 1"),
    get_image_details(gen_image_sdxl_1, "SDXL Image 1"),
    get_image_details(gen_image_sd_2, "SD Image 2"),
    get_image_details(gen_image_sdxl_2, "SDXL Image 2"),
    get_image_details(gen_image_sd_3, "SD Image 3"),
    get_image_details(gen_image_sdxl_3, "SDXL Image 3"),
]

# Print results in table format
headers = ["Image Name", "Resolution (WxH)", "Aspect Ratio", "Mode", "Channels"]
print(tabulate(image_details, headers=headers, tablefmt="grid"))

# Assuming 'details_given_image_1', 'details_given_image_2', 'details_given_image_3'
# are already available (for example from the 'get_image_details' function)

# Print details for given images
print("Details of Given Images:")
print(f"Given Image 1: {details_given_image_1}")
print(f"Given Image 2: {details_given_image_2}")
print(f"Given Image 3: {details_given_image_3}")

```

```
+----------------+--------------------+----------------+--------+------------+
| Image Name     | Resolution (WxH)   |   Aspect Ratio | Mode   |   Channels |
+================+====================+================+========+============+
| Pix2Pix Image 1 | 624 x 800          |           0.78 | RGB    |          3 |
+----------------+--------------------+----------------+--------+------------+
| Pix2Pix Image 2 | 624 x 624          |           1    | RGB    |          3 |
+----------------+--------------------+----------------+--------+------------+
| Pix2Pix Image 3 | 624 x 624          |           1    | RGB    |          3 |
+----------------+--------------------+----------------+--------+------------+
| SD Image 1     | 624 x 800          |           0.78 | RGB    |          3 |
+----------------+--------------------+----------------+--------+------------+
| SDXL Image 1   | 624 x 800          |           0.78 | RGB    |          3 |
+----------------+--------------------+----------------+--------+------------+
| SD Image 2     | 624 x 624          |           1    | RGB    |          3 |
+----------------+--------------------+----------------+--------+------------+
| SDXL Image 2   | 624 x 624          |           1    | RGB    |          3 |
+----------------+--------------------+----------------+--------+------------+
| SD Image 3     | 624 x 624          |           1    | RGB    |          3 |
+----------------+--------------------+----------------+--------+------------+
| SDXL Image 3   | 624 x 624          |           1    | RGB    |          3 |
+----------------+--------------------+----------------+--------+------------+
Details of Given Images:
Given Image 1: {'Resolution': '624 x 807', 'Aspect Ratio': 0.77, 'Mode': 'RGB', 'Channels': 3}
Given Image 2: {'Resolution': '624 x 624', 'Aspect Ratio': 1.0, 'Mode': 'RGB', 'Channels': 3}
Given Image 3: {'Resolution': '624 x 624', 'Aspect Ratio': 1.0, 'Mode': 'RGB', 'Channels': 3}
```

We observe that all the models applied has not given the correct resolution for image 1 only. This could be due to an unintentional shift in the output dimensions due to the randomness of how the model processes the input. We also had not pre-processed the image.

To handle this we pre-process the given_image_1 so that its aspect ratio is 1 and apply the models again and display the result.

Pre-processing of given image 1

```python
from PIL import Image

# Function to pad the image to a square resolution of 624x624
def pad_to_square(image, target_size=(624, 624)):
    width, height = image.size
    max_dim = max(width, height)

    # Create a new white canvas of size target_size
    new_image = Image.new("RGB", target_size, (255, 255, 255))

    # Paste the original image onto the canvas centered
    new_image.paste(image, ((target_size[0] - width) // 2, (target_size[1] - height) // 2))
    return new_image

# Load the original image
image_path = "/content/Given_pic_1.jpg"
image = Image.open(image_path)

# Resize or pad to 624x624
resized_image_1 = pad_to_square(image)

# Save the new image if needed
resized_image_1.save("/content/Given_pic_1_resized.jpg")

# Print the new image details
resized_image_1.show()
print(f"New Resolution: {resized_image_1.size}")  # Should print (624, 624)

```

```
New Resolution: (624, 624)
```

```python
image_path = "/content/Given_pic_1_resized.jpg"
details_resized_image_1 = get_image_details(image_path)

# Print the stored details
details_resized_image_1
```

```
{'Resolution': '624 x 624', 'Aspect Ratio': 1.0, 'Mode': 'RGB', 'Channels': 3}
```

Appylying all models on the resized image 1

Stable Diffusion v1.5