**FACE AGING GENERATOR USING GAN**

import pandas as pd

import torch

from diffusers import StableDiffusionPipeline

import os

import shutil

import random

from diffusers import LMSDiscreteScheduler

The cache for model files in Transformers v4.22.0 has been updated. Migrating your old cache. This is a one-time only operation.

!mkdir images

model\_id = "CompVis/stable-diffusion-v1-4"

pipe = StableDiffusionPipeline.from\_pretrained(model\_id,

local\_files\_only = False,

torch\_dtype=torch.float16)

pipe = pipe.to("cuda")

model\_index.json: 100%

541/541 [00:00<00:00, 32.5kB/s]

Fetching 16 files: 100%

16/16 [00:18<00:00, 1.32s/it]

tokenizer/merges.txt: 100%

525k/525k [00:00<00:00, 3.92MB/s]

(…)ature\_extractor/preprocessor\_config.json: 100%

342/342 [00:00<00:00, 12.5kB/s]

model.safetensors: 100%

492M/492M [00:02<00:00, 214MB/s]

safety\_checker/config.json: 100%

4.56k/4.56k [00:00<00:00, 49.7kB/s]

text\_encoder/config.json: 100%

592/592 [00:00<00:00, 6.18kB/s]

scheduler/scheduler\_config.json: 100%

313/313 [00:00<00:00, 3.45kB/s]

(…)kpoints/scheduler\_config-checkpoint.json: 100%

209/209 [00:00<00:00, 3.71kB/s]

model.safetensors: 100%

1.22G/1.22G [00:05<00:00, 240MB/s]

diffusion\_pytorch\_model.safetensors: 100%

3.44G/3.44G [00:17<00:00, 283MB/s]

unet/config.json: 100%

743/743 [00:00<00:00, 11.2kB/s]

tokenizer/tokenizer\_config.json: 100%

806/806 [00:00<00:00, 16.9kB/s]

vae/config.json: 100%

551/551 [00:00<00:00, 13.6kB/s]

tokenizer/special\_tokens\_map.json: 100%

472/472 [00:00<00:00, 11.3kB/s]

tokenizer/vocab.json: 100%

1.06M/1.06M [00:00<00:00, 5.80MB/s]

diffusion\_pytorch\_model.safetensors: 100%

335M/335M [00:02<00:00, 155MB/s]

Loading pipeline components...: 100%

7/7 [00:03<00:00, 2.24it/s]

`text\_config\_dict` is provided which will be used to initialize `CLIPTextConfig`. The value `text\_config["id2label"]` will be overriden.

`text\_config\_dict` is provided which will be used to initialize `CLIPTextConfig`. The value `text\_config["bos\_token\_id"]` will be overriden.

`text\_config\_dict` is provided which will be used to initialize `CLIPTextConfig`. The value `text\_config["eos\_token\_id"]` will be overriden.

pipe.set\_progress\_bar\_config(leave=False)

pipe.set\_progress\_bar\_config(disable=True)

import os

import random

import matplotlib.pyplot as plt

from PIL import Image as pilImage

def show\_images(uid):

folder\_path = 'images'

image\_files = [f for f **in** os.listdir(folder\_path) if f.startswith((f'**{**uid**}**'))]

*#print(image\_files)*

random\_images = image\_files *# Select 16 random images*

random\_images.sort()

thumbnails = []

for image\_file **in** random\_images:

image\_path = os.path.join(folder\_path, image\_file)

image = pilImage.open(image\_path)

image.thumbnail((140, 140)) *# Decrease the thumbnail size to fit in the grid*

thumbnails.append(image)

*# Calculate the size of the collage*

thumbnail\_size = thumbnails[0].size

spacing = 10 *# Adjust the spacing between images*

collage\_size = (thumbnail\_size[0] \* 7 + spacing \* 3, thumbnail\_size[1] \* 1 + spacing \* 2)

*# Create a new blank image for the collage*

collage = pilImage.new('RGB', collage\_size, (255, 255, 255)) *# Set white background for the collage*

*# Paste the thumbnails onto the collage in a grid with spacing*

x\_offset = spacing

y\_offset = spacing

for i, thumbnail **in** enumerate(thumbnails):

row = i // 6

col = i % 6

paste\_x = col \* (thumbnail\_size[0] + spacing) + spacing

paste\_y = row \* (thumbnail\_size[1] + spacing) + spacing

collage.paste(thumbnail, (paste\_x, paste\_y))

*# Display the collage*

plt.figure(figsize=(7, 1)) *# Adjust the figure size to accommodate the collage*

plt.imshow(collage)

plt.axis('off') *# Remove the axis*

plt.show()

The code below should run for about 6 hours on GPU and generate 480 sets of images

from IPython.display import Image as dispImage

import uuid

from PIL import Image

from PIL.PngImagePlugin import PngInfo

import random

for index **in** range(496,496\*2):

gender = random.choice(["man","woman"])

continent = random.choice(["Asia","North America","South America","Australia","Antartica","Europe","Africa"])

uid= str(uuid.uuid4())

print(uid)

seed = index

for age **in** range(20,80,10):

generator = torch.Generator("cuda").manual\_seed(seed)

prompt =f"Closeup portrait of a beautiful, **{**age**}** year old **{**gender**}** from **{**continent**}**"

image = pipe(prompt,

negative\_prompt="ugly,other objects,deformed",

generator = generator

,guidance\_scale=6.5,num\_inference\_steps=32).images[0]

filename = f"images/**{**uid**}**\_**{**age**}**\_**{**gender**}**\_**{**continent**}**\_**{**seed**}**.png"

image.save(filename)

image = Image.open(filename)

metadata = PngInfo()

metadata.add\_text('prompt', prompt)

metadata.add\_text('model', model\_id)

metadata.add\_text('seed',str(seed))

image.save(filename, pnginfo=metadata)

*#print(prompt)*

if index%10==0:

show\_images(uid)

print("completed ", index)

!zip -rq output.zip images/

!rm -rf images/\*

**OUTPUT**



completed



completed



completed



completed



completed

completed



completed