Text-to-Image + Adapter 실습 과제

소프트웨어학부

20215214

최재원

1. TODO1 이미지 생성 결과 및 코드

```
1. from diffusers import DiffusionPipeline
2. import torch
3.
4. # Base Model 설정
5. #model_id = "emilianJR/XXMix_9realistic"
6. #model_id = "stablediffusionapi/lyrielv16"
7. model_id = "sinkinai/majicMIX-realistic-v5" # 각 모델 비교
8.
9. pipe = DiffusionPipeline.from_pretrained(model_id, torch_dtype=torch.float16)
# Diffusion pipeline 에 모델 장착
10. pipe = pipe.to("cuda")
# GPU 사용
11.
# 프롬프트 작성
12. prompt = "A stylish and confident woman with long, wavy hair, wearing an elegant dress and
minimalistic jewelry. She has a serene expression and is standing in a graceful pose. The
background is softly blurred, enhancing her presence in the image with warm and gentle lighting
that highlights her features."
# 이미지 생성 시 negative_prompt 를 사용하여 원하지 않거나 불 필요한 것들을 프롬프트로 입력
14. image = pipe(prompt, negative_prompt="lowres, bad anatomy, worst quality, low
quality").images[0]
# 이미지 출력
16. image
```

이미지 분석

- sinkinai/majicMIX-realistic-v5 모델을 사용하여 입력한 프롬프트인 "A stylish and confident woman with long, wavy hair, wearing an elegant dress and minimalistic jewelry. She has a serene expression and is standing in a graceful pose. The background is softly blurred, enhancing her presence in the image with warm and gentle lighting that highlights her features."에서 웨이브 된 머리, 스타일리쉬하고 큰 키 등 입력한대로 잘 들어간 모습을 보여준다.

2. 각 Adapter 별 이미지 및 코드 분석

2-1. IP-Adapter & ControlNet

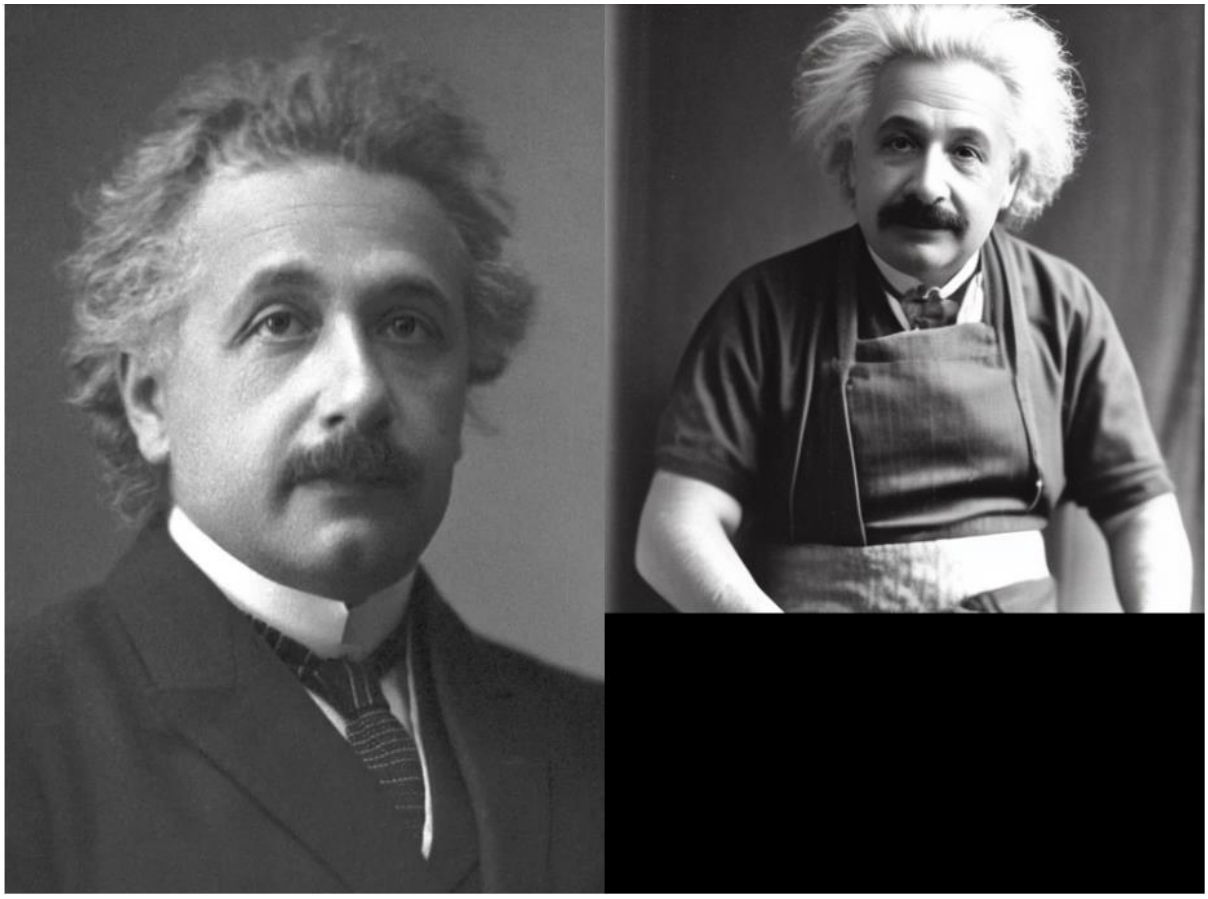
```
# TODO : IP-Adapter 와 ControlNet 연결하기
from diffusers import AutoPipelineForText2Image
import torch
from diffusers.utils import load_image

# Base Model 설정
model_id = "runwayml/stable-diffusion-v1-5"

# 파이프라인에 모델 입력
pipeline = AutoPipelineForText2Image.from_pretrained(model_id, torch_dtype=torch.float16).to("cuda")

# 파이프라인에 IP-Adapter 로드 및 스케일 설정
pipeline.load_ip_adapter("h94/IP-Adapter", subfolder="models", weight_name="ip-adapter-full-face_sd15.bin")
pipeline.set_ip_adapter_scale(0.5)

# 이미지 로드
origin_image = load_image("https://huggingface.co/datasets/huggingface/documentation-images/resolve/main/diffusers/ip_adapter_einstein_base.png")

# generator 선언
generator = torch.Generator(device="cpu").manual_seed(26)

# 파이프라인을 통해 프롬프트와 이미지를 넣어 새로운 이미지 생성
origin_image = pipeline(
    prompt="A photo of Einstein as a chef, wearing an apron, cooking in a French restaurant",
    ip_adapter_image=origin_image,
    negative_prompt="lowres, bad anatomy, worst quality, low quality",
    num_inference_steps=50,
    generator=generator,
).images[0]

from diffusers.utils import load_image, make_image_grid
from PIL import Image
import cv2
import numpy as np

```

```
image = np.array(origin_image)

# 상한, 하한 설정
low_threshold = 100
high_threshold = 200

# canny 이미지 생성
image = cv2.Canny(image, low_threshold, high_threshold)
image = image[:, :, None]
image = np.concatenate([image, image, image], axis=2)
canny_image = Image.fromarray(image)

from diffusers import StableDiffusionControlNetPipeline, ControlNetModel,
UniPCMultistepScheduler
import torch

# 새로운 파이프라인 생성
controlnet = ControlNetModel.from_pretrained("lllyasviel/sd-controlnet-canny",
torch_dtype=torch.float16, use_safetensors=True)
pipe = StableDiffusionControlNetPipeline.from_pretrained(
    "runwayml/stable-diffusion-v1-5", controlnet=controlnet, torch_dtype=torch.float16,
use_safetensors=True
)

pipe.scheduler = UniPCMultistepScheduler.from_config(pipe.scheduler.config)
pipe.enable_model_cpu_offload()

# canny 이미지에 프롬프트로 생성하여 채워넣기
output = pipe(
    "the einstein", image=canny_image
).images[0]

# 원본 이미지, canny 이미지, 생성 이미지 출력
make_image_grid([origin_image, canny_image, output], rows=1, cols=3)

```

- IP-Adapter로 생성한 이미지 / IP-Adapter로 생성한 이미지를 Canny 이미지로 변환 / Canny 이미지에 프롬프트로 채워 넣은 이미지

2-2. IP-Adapter & LoRA

```
from diffusers import AutoPipelineForText2Image
import torch
from diffusers.utils import load_image, make_image_grid
from diffusers import LCMScheduler

# Base Model 및 lora Adapter 설정
model_id = "runwayml/stable-diffusion-v1-5"
adapter_id = "latent-consistency/lcm-lora-sdv1-5"

# AutoPipelineForText2Image 객체 초기화
pipeline = AutoPipelineForText2Image.from_pretrained(model_id, torch_dtype=torch.float16, variant="fp16")

# IP-Adapter 설정
pipeline.load_ip_adapter("h94/IP-Adapter", subfolder="models", weight_name="ip-adapter-full-face_sd15.bin")
pipeline.set_ip_adapter_scale(0.5)

# set scheduler
pipeline.scheduler = LCMScheduler.from_config(pipeline.scheduler.config)
pipeline.to("cuda")

# lolar 가중치 로드
pipeline.load_lora_weights(adapter_id)
pipeline.fuse_lora()

# origin_image 로드
origin_image = load_image("https://huggingface.co/datasets/huggingface/documentation-images/resolve/main/diffusers/ip_adapter_einstein_base.png")

generator = torch.Generator(device="cpu").manual_seed(26)

# 첫 번째 이미지 생성
origin_result = pipeline(
    prompt="A photo of Einstein as a chef, wearing an apron, cooking in a French restaurant",
    ip_adapter_image=origin_image,
    #negative_prompt="lowres, bad anatomy, worst quality, low quality",
    num_inference_steps=4,
    guidance_scale=1,
    #generator=generator,
)

make_image_grid([origin_image, origin_result.images[0]], rows=1, cols=2)
```



- 오리지널 이미지 / IP-Adapter와 LoRA를 연결하여 생성한 이미지

2-3. ControlNet & LoRA

```
# TODO : ControlNet 과 LoRA 연결하기
from diffusers import AutoPipelineForText2Image
import torch
from diffusers.utils import load_image, make_image_grid
from diffusers import LCMScheduler

# Base Model 및 Adapter 설정
model_id = "runwayml/stable-diffusion-v1-5"
adapter_id = "latent-consistency/lcm-lora-sdv1-5"

# AutoPipelineForText2Image 객체 초기화
pipeline = AutoPipelineForText2Image.from_pretrained(model_id, torch_dtype=torch.float16, variant="fp16")

# set scheduler
pipeline.scheduler = LCMScheduler.from_config(pipeline.scheduler.config)
pipeline.to("cuda")

# LoRA 가중치 로드
pipeline.load_lora_weights(adapter_id)
pipeline.fuse_lora()

# 첫 번째 이미지 생성
```

```
origin_result = pipeline(
    prompt="masterpiece, best quality, 1girl, dress, wavy hair, lying, bare shoulders, blonde, jewelry, frills, (((intricate, pattern, psychedelic, surreal, abstract, swirl)))",
    num_inference_steps=4,
    guidance_scale=1
).images[0]

from PIL import Image
import cv2
import numpy as np

image = np.array(origin_result)

# 한계점 설정
low_threshold = 100
high_threshold = 200

# Canny 이미지 생성
image = cv2.Canny(image, low_threshold, high_threshold)
image = image[:, :, None]
image = np.concatenate([image, image, image], axis=2)
canny_image = Image.fromarray(image)

from diffusers import StableDiffusionControlNetPipeline, ControlNetModel, UniPCMultistepScheduler
import torch

# 새 파이프라인 생성
controlnet = ControlNetModel.from_pretrained("lllyasviel/sd-controlnet-canny", torch_dtype=torch.float16, use_safetensors=True)
pipe = StableDiffusionControlNetPipeline.from_pretrained(
    "runwayml/stable-diffusion-v1-5", controlnet=controlnet, torch_dtype=torch.float16, use_safetensors=True
)

# set scheduler
pipe.scheduler = UniPCMultistepScheduler.from_config(pipe.scheduler.config)
pipe.enable_model_cpu_offload()

# Canny 이미지에 프롬프트로 채워넣기
output = pipe(
    "masterpiece, best quality, 1girl", image=canny_image
).images[0]

# 원본 이미지, canny 이미지, 생성 이미지 출력
make_image_grid([origin_result, canny_image, output], rows=1, cols=3)
```

- LoRA를 사용하여 만든 이미지 / LoRA를 사용하여 만든 이미지를 Canny 이미지로 변환 / Canny 이미지에 프롬프트로 채워 넣은 이미지

2-4. lora_dreambooth_dog_example Adapter

```
# TODO : 하나 이상의 새로운 Adapter 연결하기
import torch
from diffusers import StableDiffusionPipeline

# Hugging Face 인증 토큰
auth_token = "YOUR_HUGGING_FACE_AUTH_TOKEN"

# 모델 ID와 어댑터 ID 설정
model_id = "runwayml/stable-diffusion-v1-5"
adapter_id = "patrickvonplaten/lora_dreambooth_dog_example"

# Stable Diffusion Pipeline 불러오기
pipeline = StableDiffusionPipeline.from_pretrained(model_id, torch_dtype=torch.float16, use_auth_token=auth_token)
pipeline = pipeline.to("cuda")

# LoRA 어댑터 로드
pipeline.load_lora_weights(adapter_id, use_auth_token=auth_token)
pipeline.fuse_lora()

# 텍스트 프롬프트 설정
prompt = "A photo of a dog in a park, wearing sunglasses and a hat"

# 이미지 생성
generator = torch.Generator(device="cuda").manual_seed(42)
output = pipeline(prompt=prompt, num_inference_steps=50, generator=generator).images[0]

output
```

- lora_dreambooth_dog_example로 생성한 이미지

2-5. Adapter 사용 결과 분석

IP-Adapter : 이미지와 프롬프트를 함께 입력하여 생성하므로 베이스 이미지에 따라 생성되므로, 원하는 결과물을 유도해낼 수 있다.

ControlNet : 이미지를 다양한 텍스트 프롬프트를 변경하지 않고 특정 이미지를 생성 가능하므로 간편하게 이미지를 생성 할 수 있다.

LoRA : 훈련 가능한 파라미터의 수를 크게 줄여주어 이미지 생성 속도를 높여준다.

Adapter를 사용하지 않은 경우와의 차이점 및 특징

- Adapter를 사용하지 않았을 경우에는 프롬프트를 세세하게 설정해야 해서 긴 프롬프트를 작성해야 하는 불편함이 있고, 이미지 생성 속도면에서도 느리지만, Adapter를 사용하였을 경우에는 원하는 Adapter를 사용하여 목표하는 출력물로 유도 가능하다.