GenAI HW10

TA: **Nai-Xuan Ye, Hsi-Che Lin, Hsiang Hsieh**

ntu-gen-ai-2024-spring-ta@googlegroups.com

Deadline: 2024/**06/13** 23:59:59 (UTC+8)

Outline

- Overview
- Task Introduction
  - Fine-tune Stable Diffusion
  - Generate Images
  - Evaluation Metrics
- Grading Policy
- Result Submission

Link

- Training Code

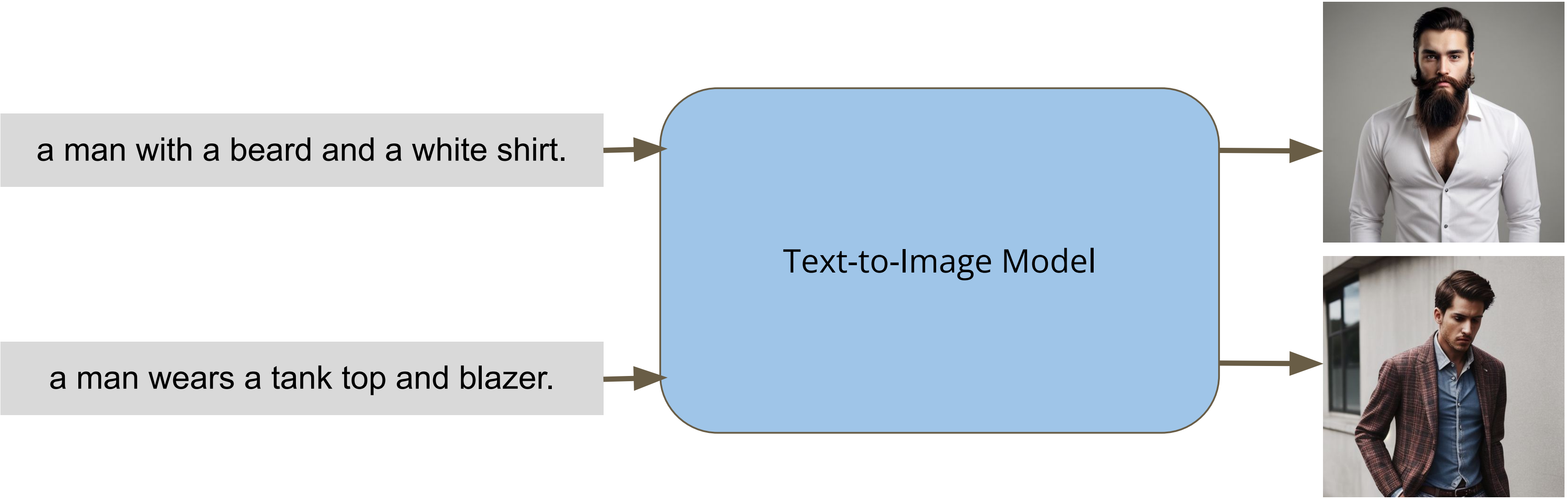
Disclaimer

- The human faces generated using the stable diffusion model for this homework are solely for educational and research purposes. They are generated by artificial intelligence algorithms and may not accurately represent real individuals.
- It is important to acknowledge that the use of AI-generated images may raise ethical considerations, and it is the responsibility of users to use them ethically and responsibly.
- The creators of the stable diffusion model and the developers of the tools utilized in this homework bear no liability for the use or consequences of these generated images.

Overview

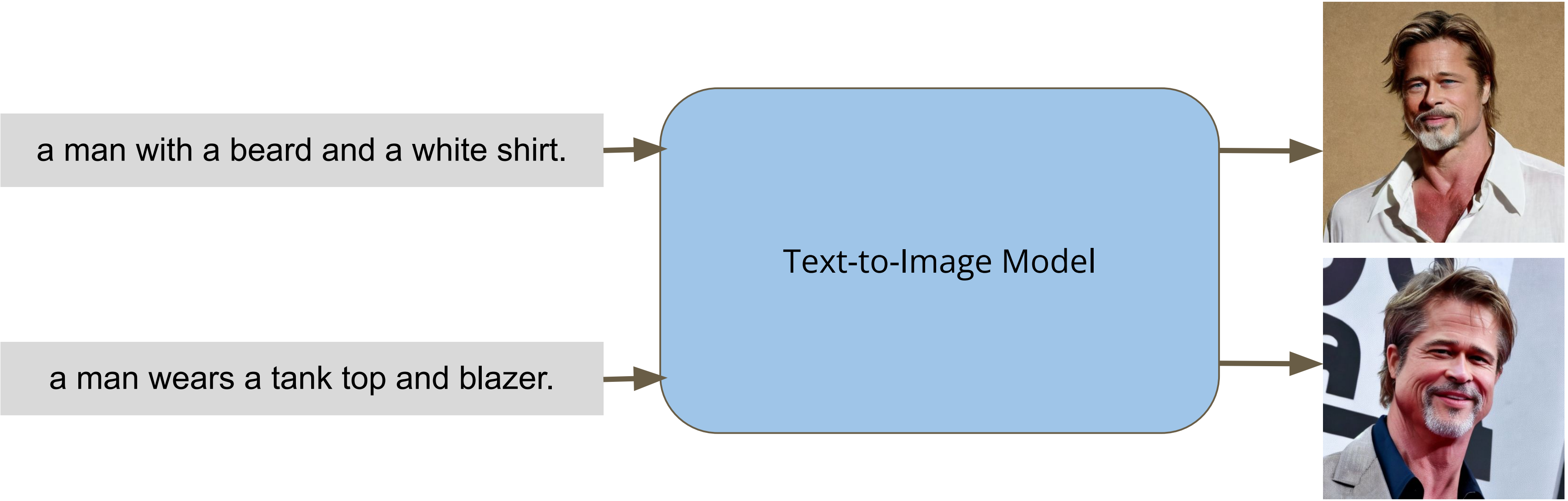
Text-to-Image Model

- Text-to-image model can generate images that matches input description



Personalization

- Publicly available text-to-image models may not meet everyone's need. E.g., a model generating images of a specific person consistently



Achieve Personalization Using LoRA

- LoRA is a technique to make small changes to a trained model (ref)

Model that generate good images

Text-to-Image Model

a man with a beard and a white shirt.

Add LoRA modules to model
(model gets additional capacity)
Then fine-tune it.

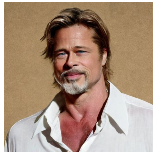
Text-to-Image Model

LoRA

fine-tune

Data

Model that generate good images
& do personalization



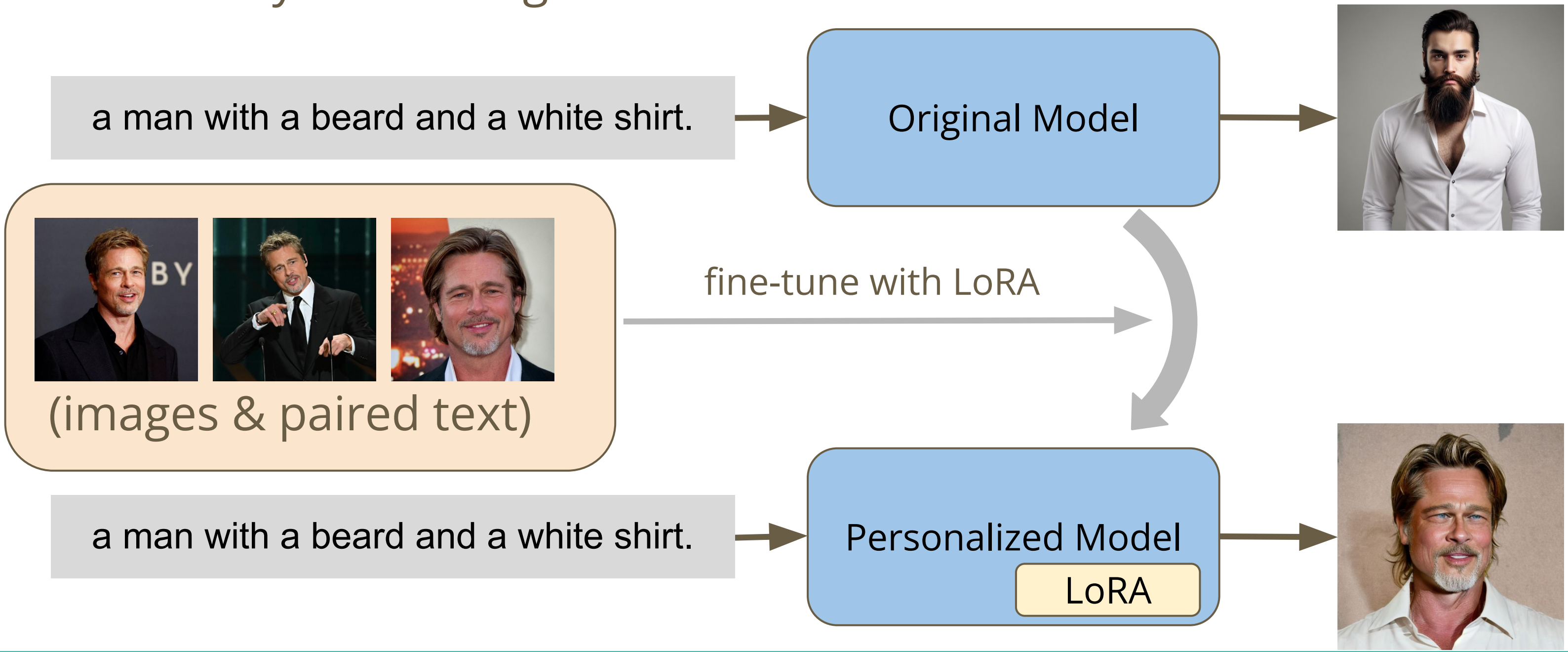
Text-to-Image Model

LoRA

a man with a beard and a white shirt.

Goal of This Homework

- You will learn how to make Stable Diffusion generate consistent facial results by fine-tuning it with identical faces.



Goal of This Homework

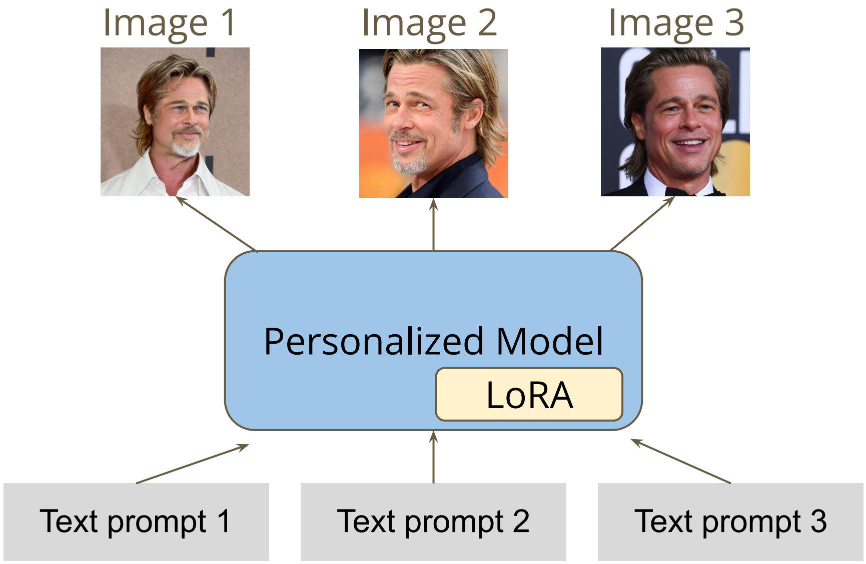
- Given enough training time, Stable Diffusion will start to produce images of the same individual.

Training Step

a man with a beard and a white shirt.

Evaluation

- We will give you some text prompts
- You need to submit images generated from the prompts
- The score will evaluated based on
  - Are the images similar to training data
  - Are the images and text matched
  - Do the images contain human faces



Task Introduction

Task Introduction

1. Fine-tune Stable Diffusion — Using LoRA technique
2. Generate Images — Using provided validation prompts
3. Evaluate Images — Evaluated by GhostFaceNet and CLIP

TODOs

- Try out different hyperparameters to fine-tune your own model. (**Using default setting will result in at most 8 points**)
- The training time will take about 1 hours. Each account is only able to run the code twice a day. You might have to try different hyperparameters multiple time. **Please start to work on this assignment as early as possible. We won't extend the deadline given the long training time**.
- Please ensure your **Google Drive has a minimum of 4GB** of available space to accommodate 2 stable diffusion checkpoints.
- We provide 25 different prompts to describe clothing. You have to ensure that your model can generate images with same faces as training dataset based on these prompts.

Step 1. Fine-tune Stable Diffusion

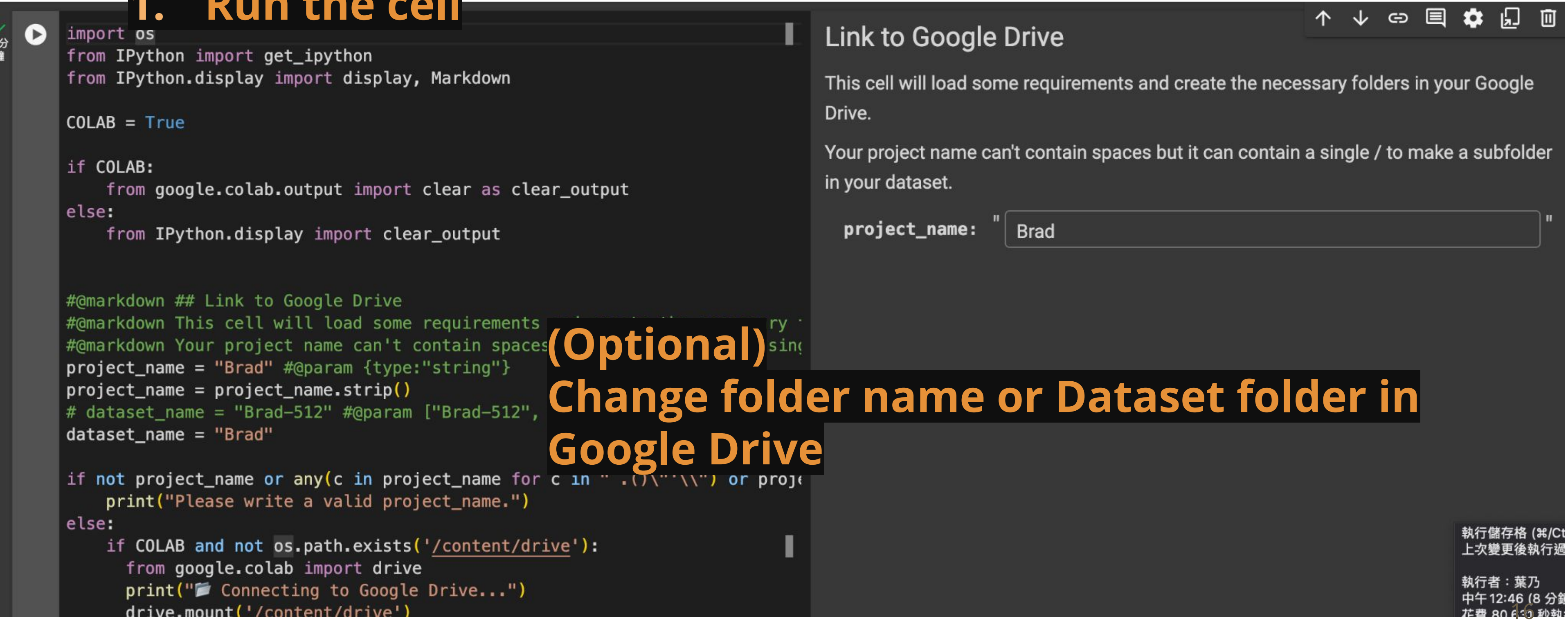
Fine-tune a Stable Diffusion LoRA Model

- Choose a model based on Stable Diffusion from Hugging Face and use LoRA technique to fine-tune Stable Diffusion.

- Before fine-tuning, try to understand and adjust hyperparameters of LoRA so that your model can have better performance in the end.

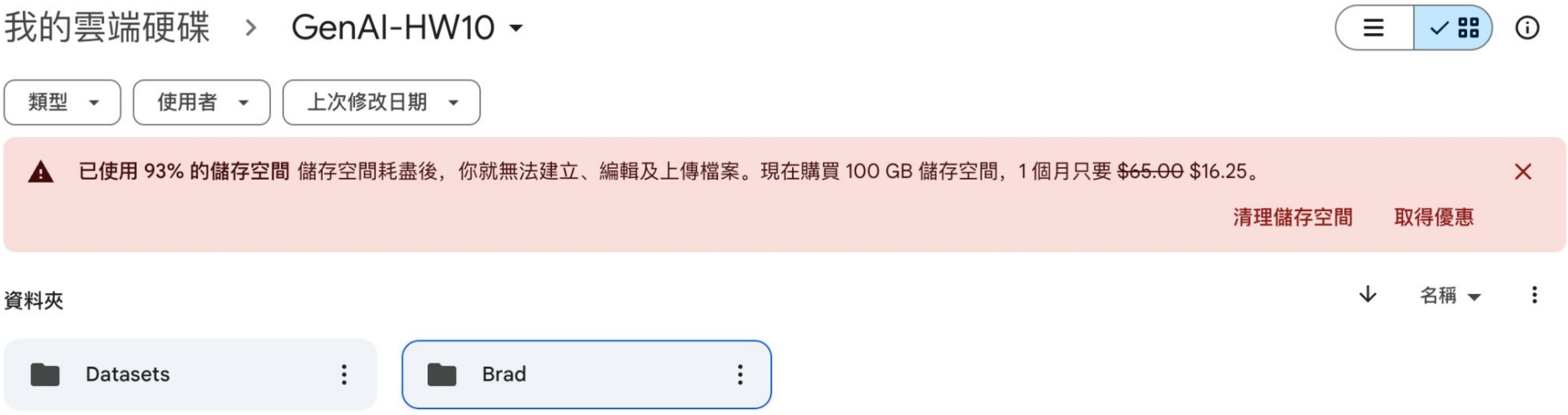
- Fine-tuning will take about **30 minutes**

1. Link to Google Drive



1. Link to Google Drive

- We will build GenAI-HW10 folders in your Google drive
- Make sure you have at least **4GB** space on the Google drive



2. Import required packages for fine-tuning

Import necessary packages

It is recommmended NOT to change codes in this cell.

```
[ ]  import argparse
     import logging
     import math
     import os
     import random
     import glob
     import shutil
     from pathlib import Path
     import numpy as np
     import torch
     import torch.nn.functional as F
     import torch.utils.checkpoint
     import transformers
     from PIL import Image
     from torchvision import transforms
     from torchvision.utils import save_image
     from tqdm.auto import tqdm
     from peft import LoraConfig
     from peft.utils import get_peft_model_state_dict
     from transformers import AutoProcessor, AutoModel, CLIPTextModel, CLIPTokenizer

     import diffusers
     from diffusers import AutoencoderKL, DDPMScheduler, DiffusionPipeline, StableDiffusionPipeline, UNet2DConditionModel
```

3. Adjust the hyperparameters

```
# Do not change the following parameters, or the process may crashed due to
output_folder = os.path.join(project_dir, "logs") # 存放model checkpoints跟
seed = 1126 # random seed
train_batch_size = 2 # training batch size
resolution = 512 # Image size
weight_dtype = torch.bfloat16 #
snr_gamma = 5
#####

#@markdown ## Important parameters for fine-tuning Stable Diffusion
pretrained_model_name_or_path = "stablediffusionapi/cyberrealistic-41"
lora_rank = 32
lora_alpha = 16
#@markdown ### ▶ Learning Rate
#@markdown The learning rate is the most important for your results. If you
#@markdown The text encoder helps your Lora learn concepts slightly better.
learning_rate = 1e-4 #@param {type:"number"}
unet_learning_rate = learning_rate
text_encoder_learning_rate = learning_rate
lr_scheduler_name = "cosine_with_restarts" # 設定學習率的排程
lr_warmup_steps = 100 # 設定緩慢更新的步數
#@markdown ### ▶ Steps
#@markdown Choose your training step and the number of generated images per
max_train_steps = 200 #@param {type:"slider", min:200, max:2000, step:100}
validation_prompt = "validation_prompt.txt"
validation_prompt_path = os.path.join(prompts_folder, validation_prompt)
validation_prompt_num = 3 #@param {type:"slider", min:1, max:5, step:1}
```

Important parameters for fine-tuning Stable Diffusion

▶ Learning Rate

The learning rate is the most important for your results. If you want to train slower with lots of images, or if your dim and alpha are high, move the unet to 2e-4 or lower.

The text encoder helps your Lora learn concepts slightly better. It is recommended to make it half or a fifth of the unet. If you're training a style you can even set it to 0.

learning_rate: 1e-4

▶ Steps

Choose your training step and the number of generated images per each validaion

max_train_steps: 200

validation_prompt_num: 3

validation_step_ratio: 0.4

3. Adjust the hyperparameters

- learning_rate (**recommended**): learning rate of model
  - Increase this make model focus more on our training data than text prompt (Face distance↓(**good**) and CLIP score↓(**bad**))
- max_train_steps: total training step
- validation_promt_num: the number of validation images
- validation_step_ratio: ratio between validation step and max_train_steps

3. Adjust the hyperparameters

- lora_rank: the dimension of LoRA model
- lora_alpha (recommended): weight of LoRA model
- learning_rate (recommended): learning rate of model
- $\text{lora\_rank} \uparrow \Rightarrow \text{Face Distance} \downarrow \text{CLIP Score} \downarrow$
- $\text{learning\_rate} \uparrow \Rightarrow \text{Face Distance} \downarrow \text{CLIP Score} \downarrow$
- max_train_steps: total training step
- validation_promt_num: the number of validation images
- validation_step_ratio: the ratio between validation step and max_train_steps

4. Prepare Dataset, LoRA model, and Optimizer

Prepare Dataset, Lora model, and Optimizer

Declare everything needed for Stable Diffusion fine-tuning.

```
tokenizer, noise_scheduler, unet, vae, text_encoder = prepare_lora_model(pretrained_model_name_or_path, lora_rank, lora_alpha)
optimizer                                           = prepare_optimizer(unet, text_encoder, unet_learning_rate, text_encoder_learning_rate)
lr_scheduler = get_scheduler(
    lr_scheduler,
    optimizer=optimizer,
    num_warmup_steps=lr_warmup_steps,
    num_training_steps=max_train_steps,
    num_cycles=3
)

dataset = Text2ImageDataset(
    images_folder=images_folder,
    captions_folder=captions_folder,
    transform=train_transform,
    tokenizer=tokenizer,
)
def collate_fn(examples):
    pixel_values = []
    input_ids = []
    for tensor, input_id in examples:
        pixel_values.append(tensor)
        input_ids.append(input_id)
    pixel_values = torch.stack(pixel_values, dim=0).float()
    input_ids = torch.stack(input_ids, dim=0)
    return {"pixel_values": pixel_values, "input_ids": input_ids}
train_dataloader = torch.utils.data.DataLoader(
    dataset,
    shuffle=True,
    collate_fn=collate_fn,
    batch_size=train_batch_size,
    num_workers=8,
)
print("Preparation Finished!")
```

5. Start Fine-tuning

Start Fine-tuning

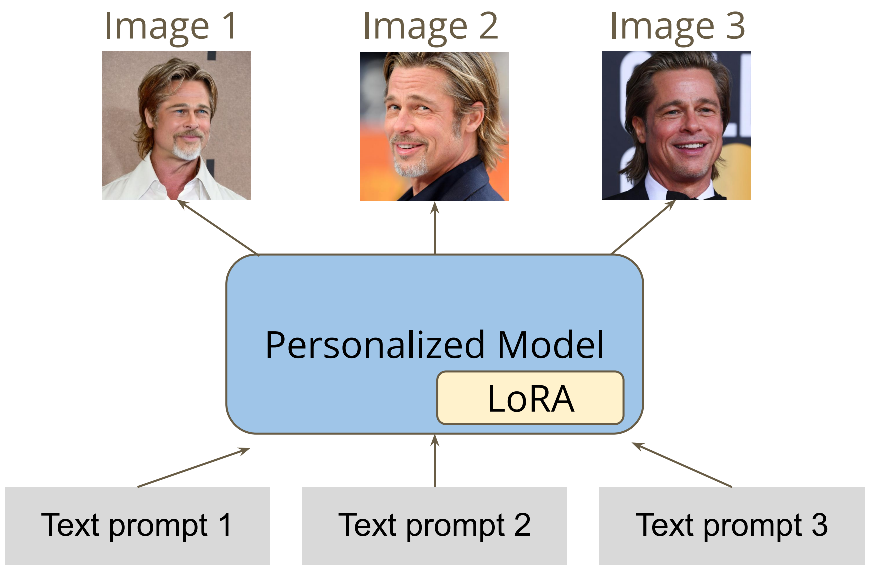
This cell takes 25 minutes to run in the default setting, but it may vary depending on the condition of Colab and `max_train_step`.

```
os.environ["TOKENIZERS_PARALLELISM"] = "false"
torch.backends.cuda.enable_mem_efficient_sdp(False)
torch.backends.cuda.enable_flash_sdp(False)
progress_bar = tqdm(
    range(0, max_train_steps),
    initial=0,
    desc="Steps",
)
global_step = 0
num_epochs = math.ceil(max_train_steps / len(train_dataloader))
validation_step = int(max_train_steps * validation_step_ratio)
best_face_score = float("inf")
for epoch in range(num_epochs):
    unet.train()
    text_encoder.train()
    for step, batch in enumerate(train_dataloader):
        if global_step >= max_train_steps:
            break
        latents = vae.encode(batch["pixel_values"].to(DEVICE, dtype=weight_dtype)).latent_dist.sample()
        latents = latents * vae.config.scaling_factor
        # Sample noise that we'll add to the latents
        noise = torch.randn_like(latents)
```

Step 3. Generate Images

Generate Images

- Use TA's validation prompts and your fine-tuned model to generate images.
- Testing will take at least **15 minutes**



Generate Images

Testing

The fine-tuning process is done. We then want to test our model.

We will first load the fine-tuned model for checkpoint we saved and calculate the face similarity, CLIP score, and the number of faceless images.

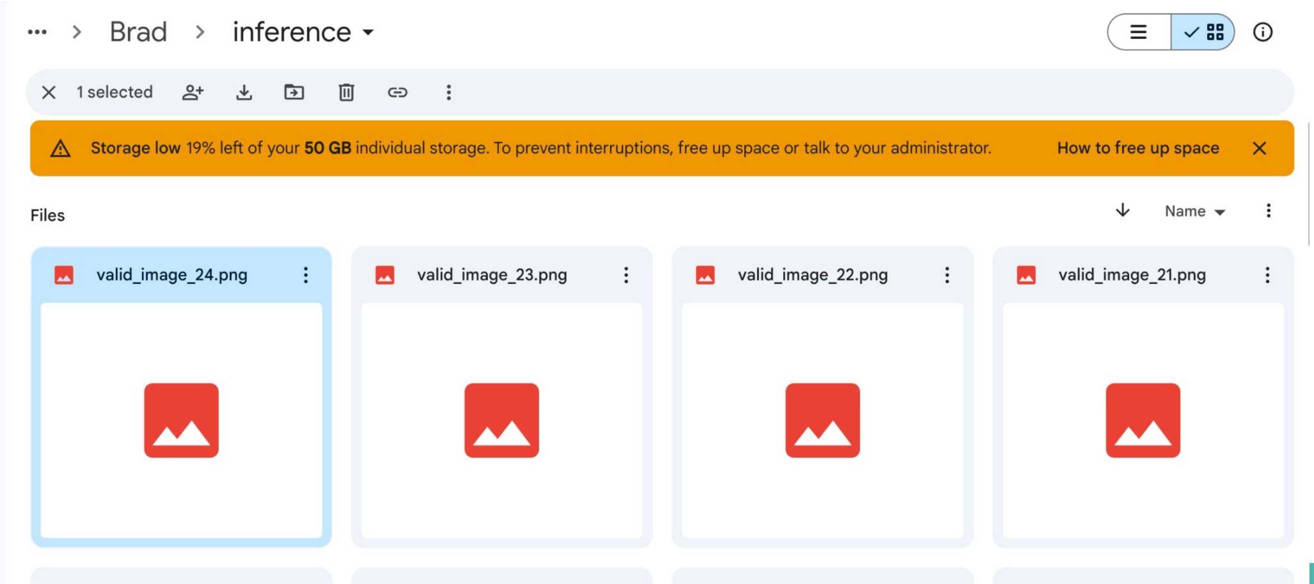
```
checkpoint_path = os.path.join(output_folder, f"checkpoint-best") # 設定使用哪個checkpoint inference
unet_path = os.path.join(checkpoint_path, "unet.pt")
text_encoder_path = os.path.join(checkpoint_path, "text_encoder.pt")
inference_path = os.path.join(project_dir, "inference")
os.makedirs(inference_path, exist_ok=True)
train_image_paths = []
for ext in IMAGE_EXTENSIONS:
    train_image_paths.extend(glob.glob(f"{images_folder}/*{ext}"))
train_image_paths = sorted(train_image_paths)
train_emb = torch.tensor([DeepFace.represent(img_path, detector_backend="ssd", model_name="GhostFaceNet", enforce_detection=False)[0]['embedding'] for img_path in train_image_paths])

face_score, clip_score, mis = evaluate(
    pretrained_model_name_or_path=pretrained_model_name_or_path,
    weight_dtype=weight_dtype,
    seed=seed,
    unet_path=unet_path,
    text_encoder_path=text_encoder_path,
    validation_prompt=validation_prompt,
    output_folder=inference_path,
    train_emb=train_emb,
)
print("Face Similarity Score:", face_score, "CLIP Score:", clip_score, "Faceless Images:", mis)
```

Generate Images

```
You have disabled the safety checker for <class 'diffusers.pipelines.stable_diffusion.pipeline
Generating validaion pictures ......
100%  30/30 [01:43<00:00, 3.34s/it]
100%  30/30 [01:44<00:00, 3.35s/it]
100%  30/30 [01:43<00:00, 3.35s/it]
100%  30/30 [01:44<00:00, 3.36s/it]
100%  30/30 [01:43<00:00, 3.35s/it]
100%  30/30 [01:44<00:00, 3.36s/it]
100%  30/30 [01:43<00:00, 3.36s/it]
100%  30/30 [01:44<00:00, 3.35s/it]
100%  30/30 [01:44<00:00, 3.37s/it]
100%  30/30 [01:43<00:00, 3.36s/it]
100%  30/30 [01:44<00:00, 3.36s/it]
100%  30/30 [01:44<00:00, 3.36s/it]
100%  30/30 [00:52<00:00, 1.70s/it]
Calculating validaion score ......
100%  50/50 [00:36<00:00, 1.49it/s]
Face Similarity Score: 1.1781010627746582 CLIP Score: 29.211345138549806 Faceless Images: 0
```

Generate Images

- The generated images will be saved in Google Cloud <project_name>/inference/
- Please zip images in this folder and submit it to NTU Cool.



Step 4. Evaluate Images

Evaluation Metrics - Face Distance Score

- For each generated face, we find its average distance to training data and average this distance across all generated faces.
- Distances between faces determined by a neural network, GhostFaceNet
- Since we want a model that generates a specific person, less average distance is better.

$$F(D_G, D_T) = \frac{1}{\|D_G\|\|D_T\|} \sum_{d_G \in D_G} \sum_{d_t \in D_T} f(d_G, d_T)$$

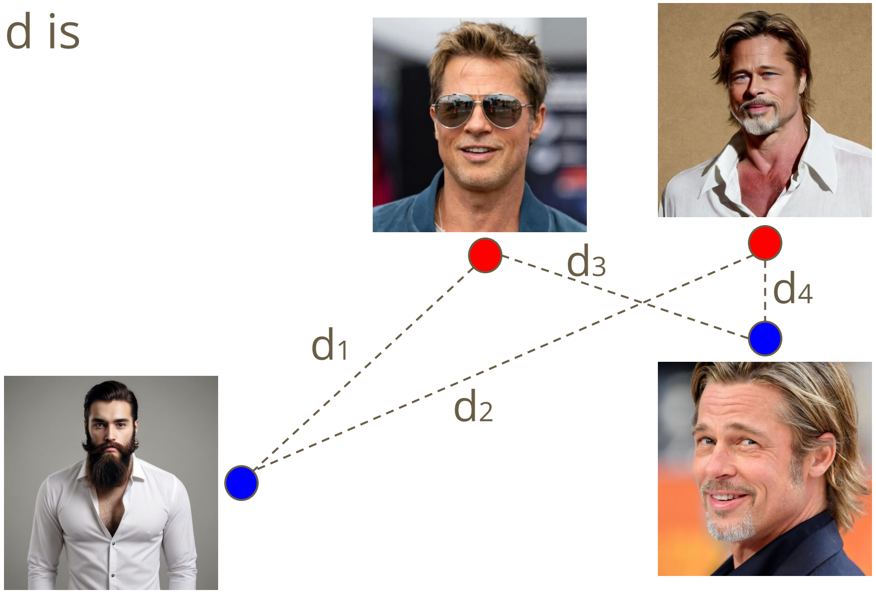
where D_G is generated faces, D_T is training data, $f(*)$ is GhostFaceNet

Paper reference: GhostFaceNets

Evaluation Metrics - Face Distance Score

- d_1, d_2, d_3, d_4 are computed by GhostFaceNet
- Face distance score is the average of d_1, d_2, d_3, d_4
- The more similar the smaller d is (**smaller is better**)

● : Training Dataset(D_T)
● : Submitted Images(D_G)



Evaluation Metrics - Face Distance Score

- However, merely calculating the face similarity is not ideal since it can easily be hacked by submitting 25 pictures of same person.

Training Dataset

Submitted Validation Images

Evaluation Metrics - Face Distance Score

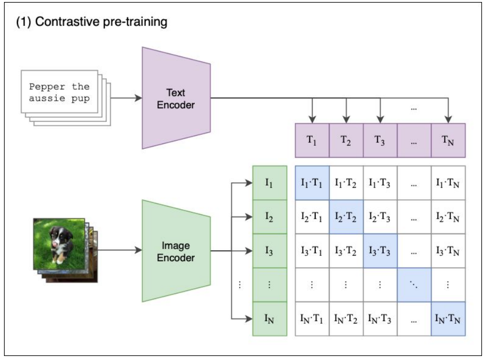
- Or just submit same picture 25 times.

Training Dataset

Submitted Validation Images

Evaluation Metrics - CLIP Score

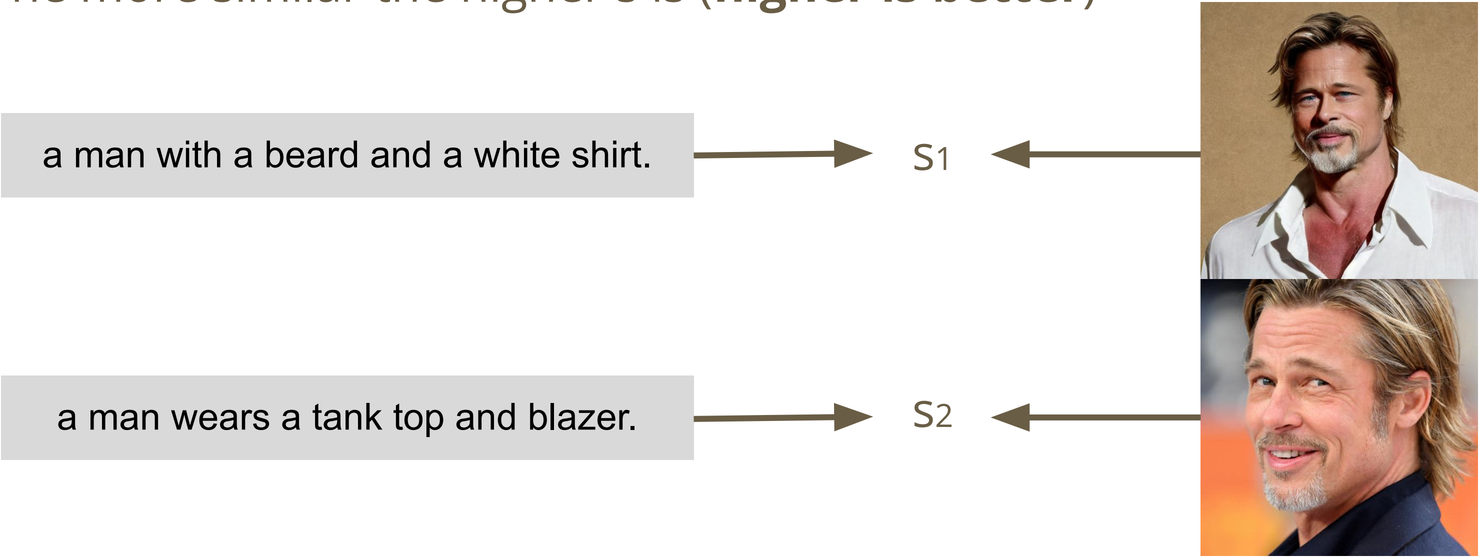
- To avoid such cases, we will calculate CLIP score to ensure that prompts are truly related to image.
- CLIP score can measure the similarity between text and image, where higher scores indicates greater relevance between the two modalities.
- By doing so, unless you can find images of the same person with 25 different clothes mentioned in validation prompts, or you'll have to fine-tune your own text-to-image model.



Paper reference: CLIP

Evaluation Metrics - CLIP Score

- s_1, s_2 are computed by CLIP
- CLIP is the average of s_1, s_2
- The more similar the higher s is (**higher is better**)



Evaluation Metrics - The number of Faceless Images

- Since it is inevitable that sometimes Stable Diffusion generates images that are related to input text but faceless.
- To avoid rerunning generation process, we tolerate at most 5 submitted validation images without faces.

a man dressed casually in a hoodie and jeans

Grading Policy

- You have to use validation prompt with same gender as your training dataset to generate images.
- Scoring:
  - Face distance:
    - $\leq 1.2 \rightarrow$ 10 pts
    - $\leq 1.22 \rightarrow$ 8 pts
    - $\leq 1.25 \rightarrow$ 6 pts
    - $\leq 1.28 \rightarrow$ 4 pts
    - $\leq 1.3 \rightarrow$ 2 pts
  - CLIP Score > 22, or you'll get 0 pts
  - The number of faceless images ≤ 5, or you'll get 0 pts
- Default setting in sample code can obtain at least 8 pts

Submission & Deadline

- Submit your homework to **NTU Cool**
- Please submit <student-id>.zip (start with capital) to **NTU Cool**. The directory after unzipping should be like:
    - B09902008
        - valid_image_0.png
        - valid_image_1.png
        - ...
        - valid_image_24.png
- 2024/**06/13** 23:59:59 (UTC+8)
- No late submission is allowed

Grading Release Date

- The grading of the homework will be released by 2024/**06/21** 23:59:59 (UTC+8)

需要提早送出成績的同學

- 本課程學期成績預計於 6/30送出
- 若有同學因為要畢業需要提早拿到學期成績，請在 **6/12 23:59:59 (UTC+8) 之前**寄信跟助教說明需要提早送出成績的原因，**我們會在 6/17 送出你的成績**。寄信說明如下：
  - 助教信箱：ntu-gen-ai-2024-spring-ta@googlegroups.com
  - 信件標題：[GenAI 2024 Spring **提早送成績**]
  - An email with the wrong title will be moved to trash automatically
  - 寄給助教的信件應該包含以下資訊：
    - 姓名
    - 學號
    - 你需要提早送出成績的理由
- 需要提早送成績的同學，我們會在 **6/16 公佈你的HW10成績**。若同學覺得HW10成績有問題，需要在 **6/17 前反應**，成績送出後不會再受理修改成績的要求

If You Have Any Questions

- NTU Cool **HW10** 作業討論區
  - 如果同學的問題不涉及作業答案或隱私，請**一律使用**NTU Cool 討論區
  - 助教們會優先回答NTU Cool討論區上的問題
- Email: ntu-gen-ai-2024-spring-ta@googlegroups.com
  - Title should start with [GenAI 2024 Spring **HW10**]
  - Email with the wrong title will be moved to trash automatically
- TA Hours
  - Time: 5/31 (Fri.) 16:30 ~ 17:20, 6/7 (Fri.) **14:20~16:10**
  - Location: 綜合大講堂