生成式 AI 技術與應用

Generative AI Technology and Applications

生成 AI テクノロジーとアプリケーション

Tz-heng, Fu
National Taipei University of Technology, IXD

Diffusion Model Fine-tuning

- LoRA Theory
- Inference with LoRA Model
- Discussion

Model Fine-Tuning for SD

- What is Model Fine-Tuning for Stable Diffusion?
  - → Fine-tuning involves adapting a pre-trained Stable Diffusion model by adjusting its parameters. This process allows the model to better capture and generate specific ***concepts*** - such as particular objects or artistic styles—making it more effective for specialized applications.
  - → Key Fine-Tuning Methods (by Date):
    - 2022.08.02: Textual Inversion
    - 2022.08.25: DreamBooth
    - 2022.11.21: DreamArtist
    - 2022.12.07: LoRA
  - → You can find plenty of different models on civitai.com.

LoRA

- Low-Rank Adaptation (LoRA), introduced by Edward Hu in 2021 during his PhD at Mila, Quebec, is a technique designed to improve domain adaptation by reducing the dimensionality of the feature space. This method enhances the model's ability to generalize across different downstream tasks in various domains.
- Domain adaptation is the process of transferring a model trained on one domain (the source domain) to a different domain (the target domain), where the data distributions and feature representations may differ. The goal of LoRA is to minimize the domain gap by capturing the shared structure and information between the source and target domains.
- LoRA achieves this by introducing low-rank matrices into the model's architecture. These matrices approximate the updates required for domain adaptation, reducing the rank (or dimensionality) of the feature space. By using low-rank approximations, LoRA effectively reduces the complexity of the adaptation process, allowing the model to better align with the target domain's feature representation.
- This approach enables the model to preserve important shared features while minimizing unnecessary complexity, which ultimately enhances performance on the target domain. Additionally, LoRA can be seen as a computationally efficient technique, as it avoids the need to update the entire model, focusing instead on learning smaller, task-specific adaptations.

- LoRA: Low-Rank Adaptation of Large Language Models [*]

Why LoRA?

- For large language models (LLMs), it becomes increasingly impractical to fine-tune **ALL** the parameters after just a few months of training. For example, GPT-3-175B has 175 billion parameters, making both fine-tuning and model deployment extremely challenging. In response, Microsoft introduced Low-Rank Adaptation (LoRA), which freezes the weights of the pre-trained model and injects trainable low-rank decomposed matrices into each layer of the Transformer architecture. This significantly reduces the number of trainable parameters for downstream tasks, decreases the checkpoint size, and avoids additional inference time overhead.
- Taking the GPT-3-175B model as an example, under training with the Adam optimizer, LoRA reduces the number of trainable parameters to 1/10,000 and reduces GPU memory usage to 1/3.

LoRA

- Pre-trained Weights (W): LoRA starts with a pre-trained model, which has been trained on large-scale datasets. These weights are fixed during the fine-tuning process to avoid excessive computational cost.
- Low-Rank Decomposition:
  - → Decomposing Parameters: LoRA introduces two matrices, A and B, that are trainable. A and B are low-rank decompositions that approximate the updates needed for the model's adaptation.
  - → Matrix A: This matrix is initialized with a random distribution (typically Gaussian), capturing the essence of the adaptation.
  - → Matrix B: This matrix is set to zero initially, and during training, it captures the necessary transformations required by the task.

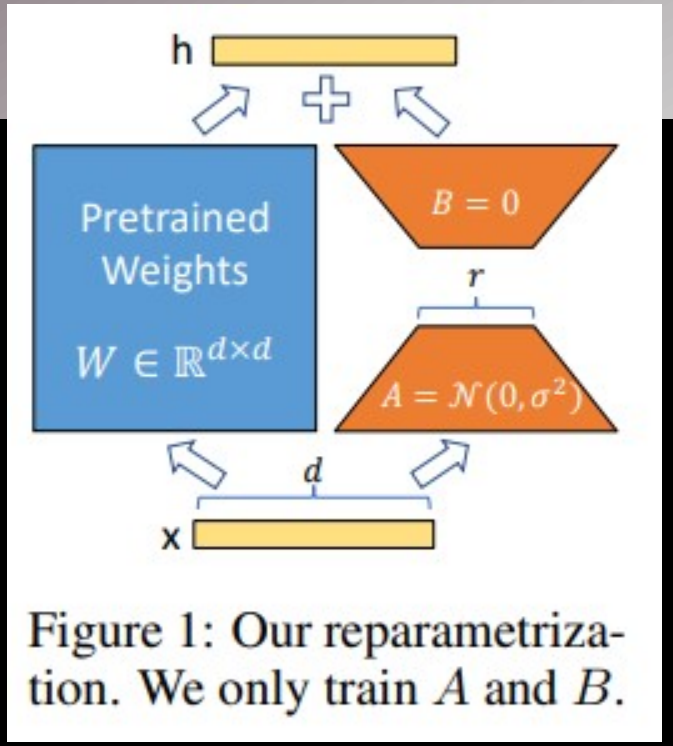


Figure 1: Our reparametrization. We only train A and B.

Replace 1 Huge Matrix with 2 Smaller Matrices

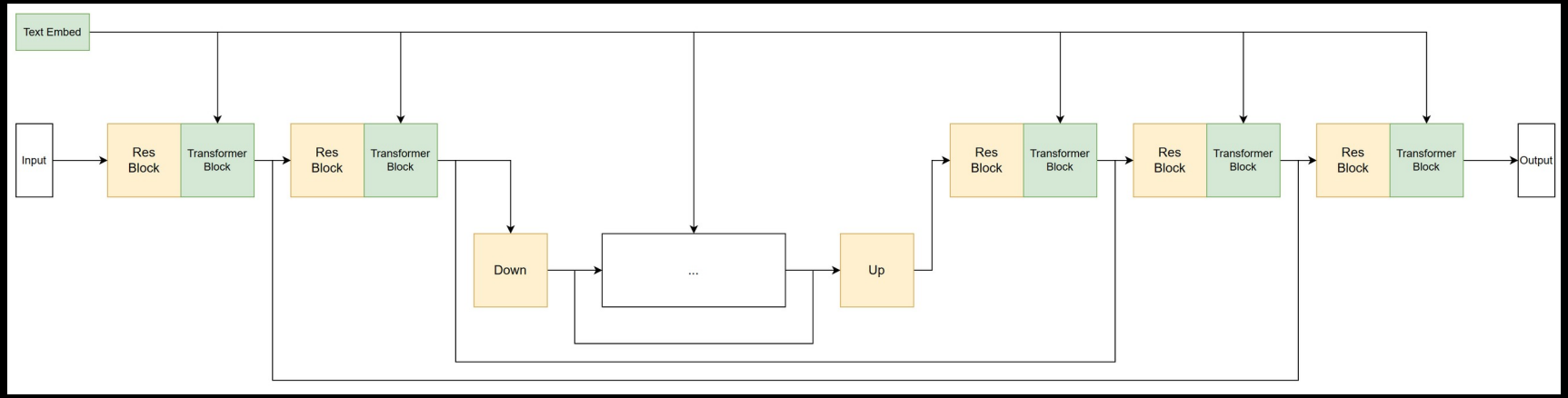
$$\Delta W = BA$$

$$h = W_0 x$$

$$h = W_0 x + \Delta W x = W_0 x + BAx$$

LoRA Family – LyCORIS Package

- LyCORIS: A package of LoRA / LoCON / LoHA / LoKR / DyLoRA / (IA)^3 and similar fine-tuning solutions.[*]
  - → LoRA for SD trains the text encoder and attention layers in UNET, the green part.
  - → LoCon: LoRA for Convolutional Layer. Use the same idea from LoRA on convolutional layers in UNET, the yellow part
  - → LoHA: inspired by FedPara: Low-Rank Hadamard Product for Communication-Efficient Federated Learning. It is originally designed for federated learning. Further lowering down dimension (half to conventional LoRA).



- Navigating Text-To-Image Customization: From LyCORIS Fine-Tuning to Model Evaluation [*]

Benefits of LoRA

- Pre-trained models can be shared and reused as a foundation, while newly trained LoRA modules serve as plug-and-play components to extend new downstream tasks.
- LoRA reduces the number of trainable parameters, thereby improving training efficiency.
- It uses a bypass injection approach to merge with the original pre-trained model. Compared to directly fine-tuning the pre-trained model, there is no additional latency during inference.
- LoRA remains orthogonal to most priors, meaning it can be integrated with them.

Train Once, Use Multiple Times!

The same LoRA can be adapted to different SD models. It's the best to train LoRA on standard SD v1.5 then apply to other v1.5 variant models.

- Navigating Text-To-Image Customization: From LyCORIS Fine-Tuning to Model Evaluation [*]

Diffusion Model Fine-tuning

- LoRA Theory
- Preparations for Training
- Discussion

Two Different Dreambooth Approach

- Dedicated Model:
  - → Just maintain and focus on the target concept. Don't care too much about most of the other unrelated concepts => this cause damages!
  - → Turn Off Regularization by setting its strength to 0.
  - → Train on shorter steps. Do not train too deep
    - This damages the prior so do not train too deep. Otherwise the model will be burnt (alias/glitchy noise or over bright).
- Generative Model:
  - → Not only to train the features of target concept but also preserve/repair a linked prior class concept => repair the damages!
  - → Tunr On Regularization by setting it strength to reasonable standard (1 ~ 0.1).
  - → Set a reasonable class token.
  - → Prepare for class images (these are also called prior knowledge). You may generate them by SD v1.5.
  - → Train on larger steps.

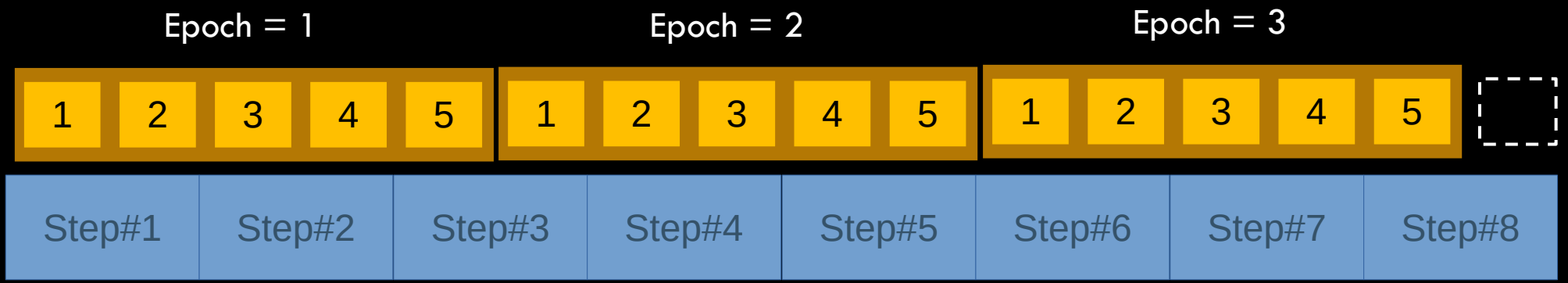
Dataset

- Collect total 4 (or more) images of a character. Each has a size of 512 x 512.
  - → You may create these 4 images AI or handpaint.
  - → If you just have one image, you can scale, rotate or mirror the image to make it four.
- Folder names:
  - → Instance images folders: <instance name>
    - Different token name may have different quality!! Ref words [*]
      - shs, sts, scs, cpc, coc, cic, msm, usu, ici, lvl, cic, dii, muk, ori, hru, rik, koo, yos, wny
  - → Regularization images (class images) folders: <class name>
    - Class images are for prior preservation loss.

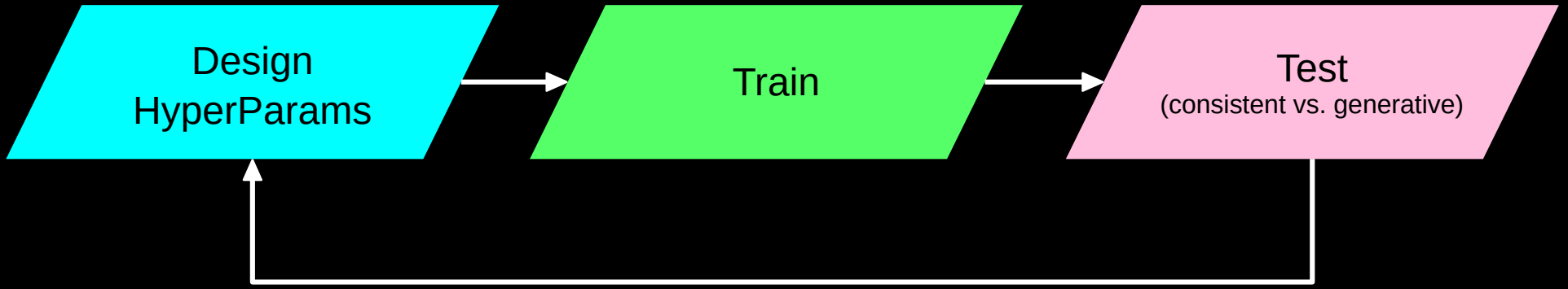
```
Dataset
|
├──instance
|   |
|   └── wpg
|       |
|       ├── image1.jpg
|       ├── image2.jpg
|       ├── image3.png
|       └── image4.png
|
|
└──regularization
    |
    └── woman
        |
        ├── reg1.jpg
        └── reg2.jpg
```

Hyperparameters

- Epoch, Batch Size, Step and sample
  - → The following shows a dataset of 5 image trained for 3 Epochs.
    - The gives a 15 training samples in total.
  - → The batch size is set to 2. So it needs 8 steps to train these 15 training samples.
- GPU processes one iteration in one step. A 8-step training means GPU calls train() function for 8 times. And each time it calls train(), it measures loss and do back propagation for once.
  - → A larger batch size requires more VRAM to store the input, hidden layer and output tensors values.
  - → Size of model weights are independent to batch size.



- → Recommendations for training a Dedicated Model:
  - Turn off Regularizations by set strengh to 0.
  - Number of Training Steps closed to 100~200.
  - A balanced Batch Size x Learning Rate is around 1e-3
    - Larger batch size can stablize the training. But how large it can be depends on your VRAM size.
- → Trade-off between consistency and generativeness
  - The deeper you train, the more consistency and less generativeness you'll get.
  - To go deeper:
    - Larger Steps
    - Smaller Learning Rate

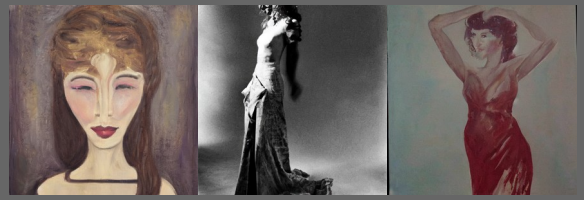


Overwhelming Tokens

- Some tokens in SD v1.5 show an incredible memory to some classic images.
- Before that, let's exam for some current instance tokens in SD v1.5
  - → " ", the empty token
  - → Girl
  - → Woman
  - → Anime
  - → Mona lisa
  - → Bloodborne

girl

woman



anime

Overfitted Concepts

Mona Lisa

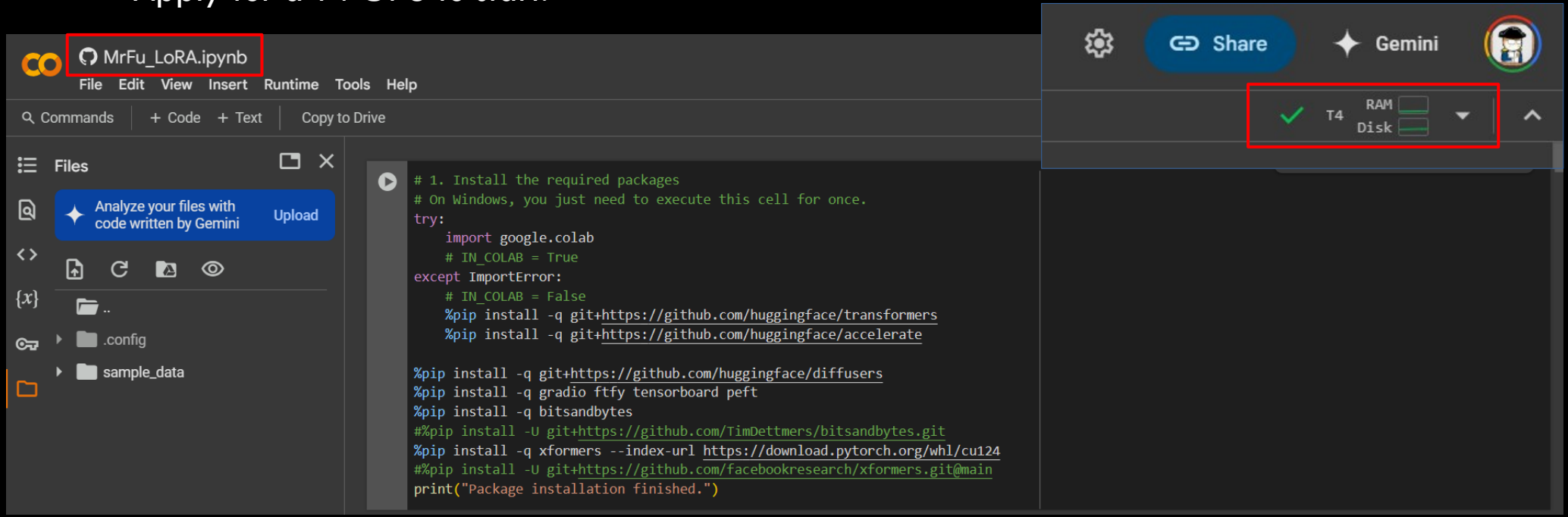
Bloodborne

Empty token

Underfit Concepts

Mr. Fu's LoRA WebUI

- Open the following Github link in Google Colab:
  - → https://github.com/jomo0825/MrFuGenerativeAI/blob/main/LoRA/MrFu_LoRA.ipynb
  - → The LoRA model can be saved to your Google Drive. It's much smaller then a Dreambooth model.
  - → Apply for a T4 GPU to start.

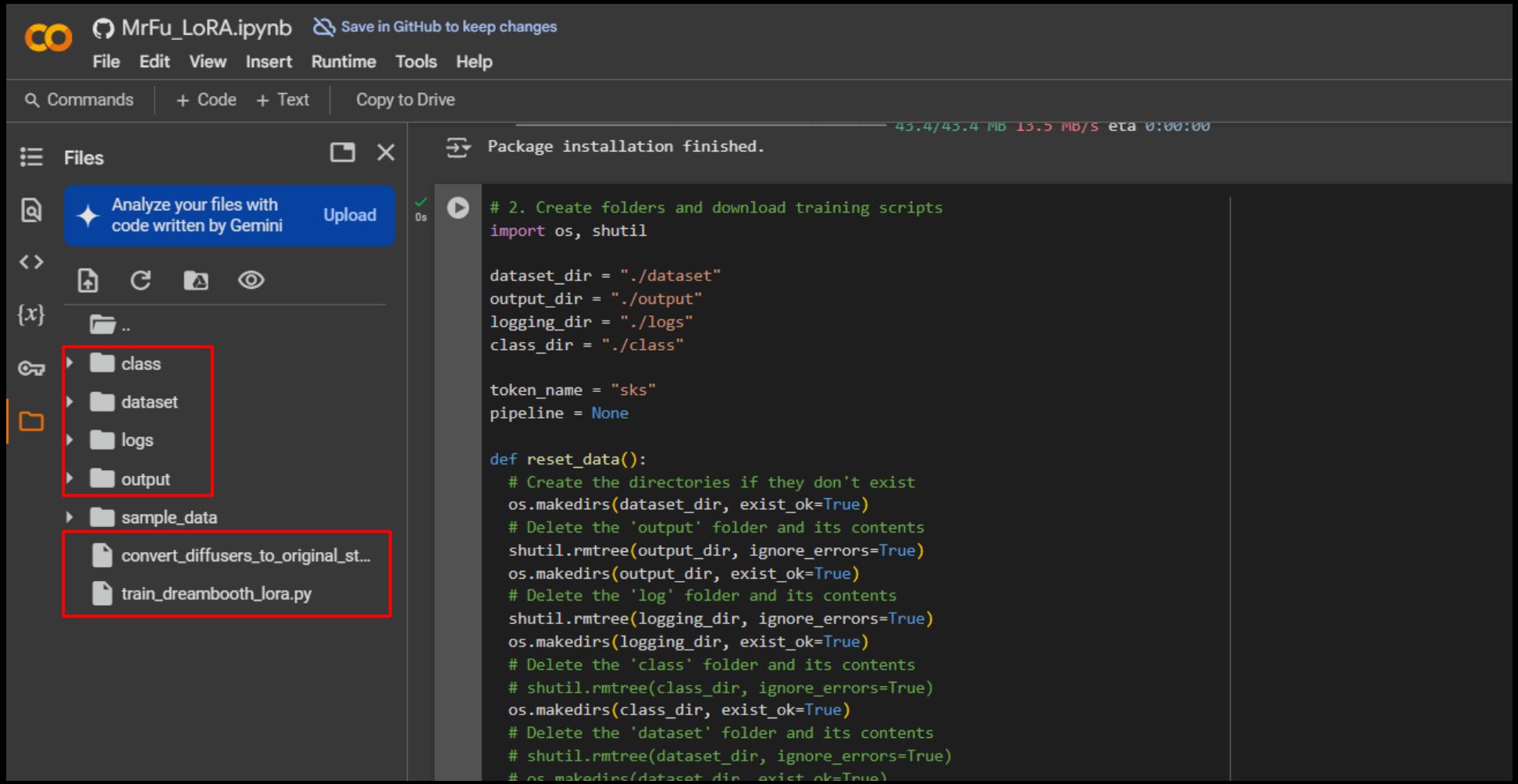


- Listen to the teach's explaination for the whole jupyter notebook.
- Key points:
  - → 1. Install the required packages
    - Wait about 3 minutes until you see "Package installation finished".
    - We can ignore the pytorch version imcompability error.

```
# Install the required packages
# On Windows, you just need to execute this cell for once.
try:
    import google.colab
    # IN_COLAB = True
except ImportError:
    # IN_COLAB = False
    %pip install -q git+https://github.com/huggingface/transformers
    %pip install -q git+https://github.com/huggingface/accelerate

%pip install -q git+https://github.com/huggingface/diffusers
%pip install -q gradio ftfy tensorboard
%pip install -q bitsandbytes
#%pip install -U git+https://github.com/TimDettmers/bitsandbytes.git
%pip install -q xformers --index-url https://download.pytorch.org/whl/cu124
#%pip install -U git+https://github.com/facebookresearch/xformers.git@main
print("Package installation finished.")
```

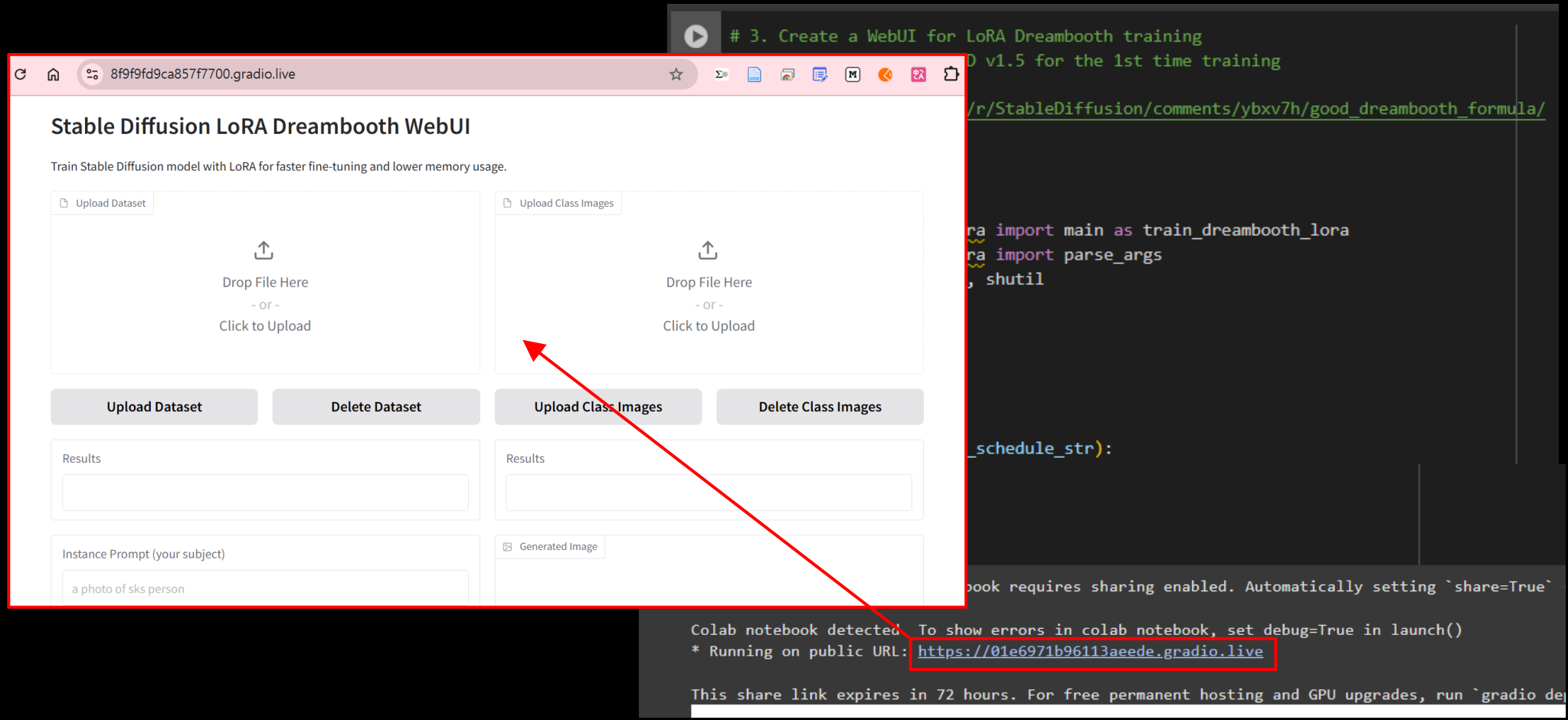
```
rement already satisfied: sympy==1.13.1 in /us
rement already satisfied: mpmath<1.4,>=1.1.0 i
rement already satisfied: MarkupSafe>=2.0 in /
oading https://download.pytorch.org/whl/cu124/
                                        43.4/43
Installing collected packages: xformers
Successfully installed xformers-0.0.29.post3
Package installation finished.
```

→ 2. Create folders and download training scripts



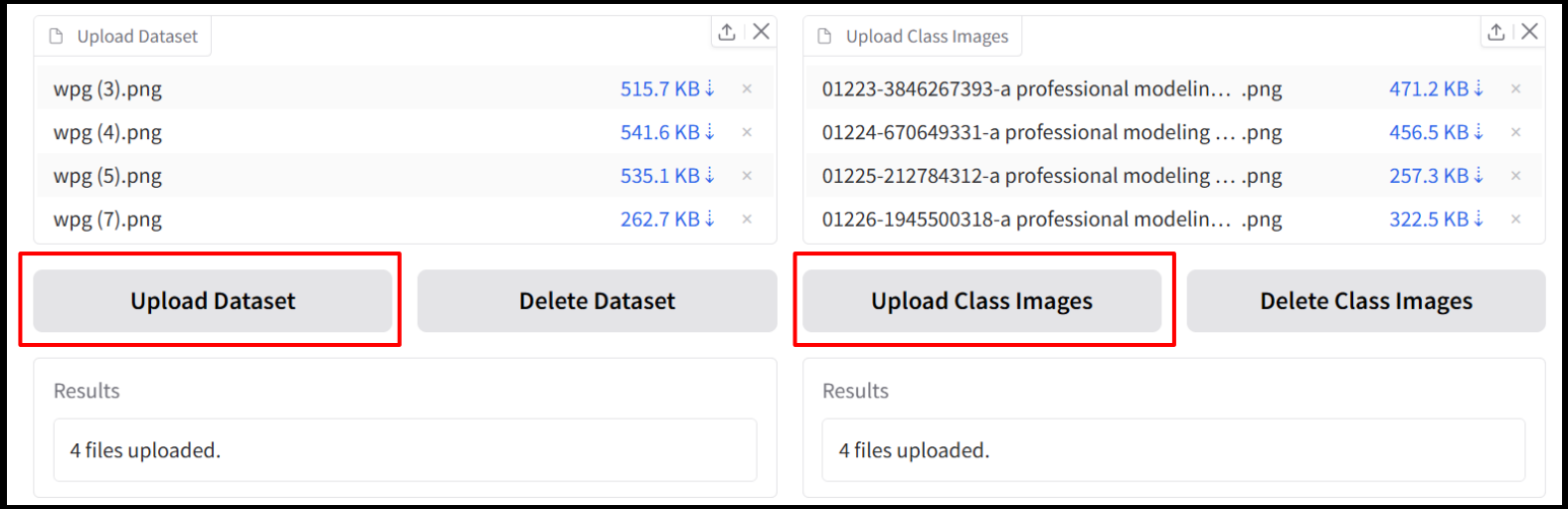
→ 3. Create a WebUI for training

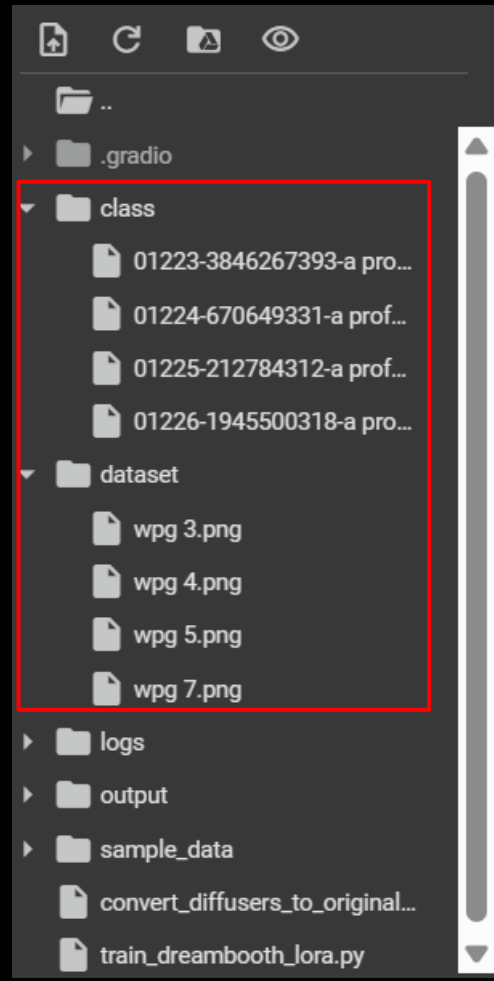
- Click the gradio.live link to open the WebUI in browser.



→ 3. Upload images

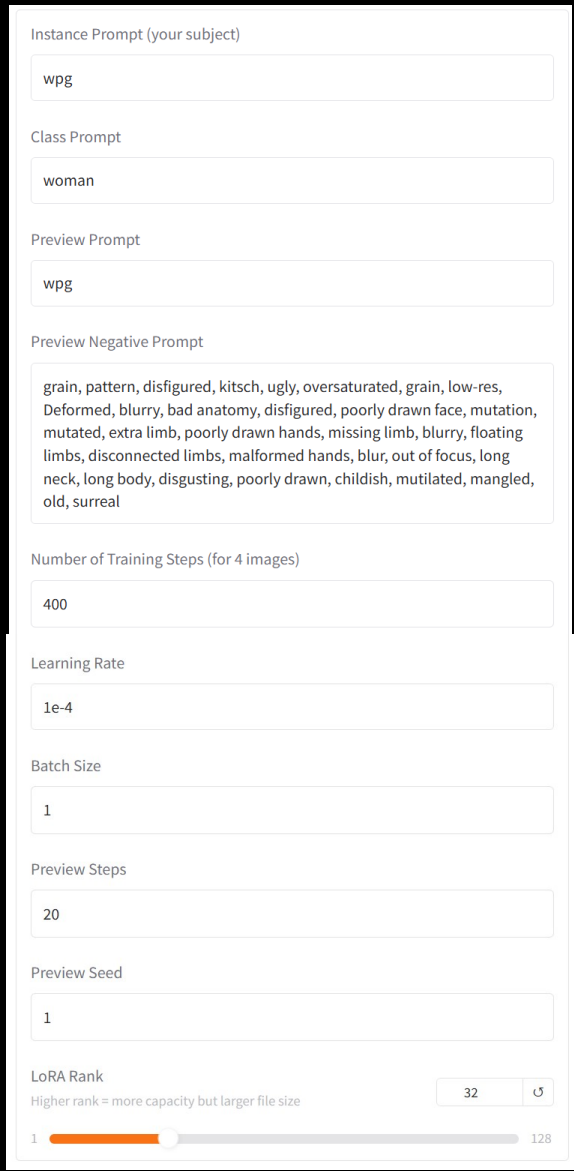
- Upload dataset images and class images
  - Select files
  - Click Upload Dataset or Upload Class Images
- The Class images should have similar account to dataset images.





→ Training hyperparameters

- Instance Prompt: the 1-vector token name for the training target. Please write down this token name for later image generation.
- Class Prompt: this should be used to train prior perservation loss. Currently this is not available.
- Preview Positive Prompt: the prompt used for preview.
- Preview Negative Prompt: the negative prompt used for preview.
- Number of Training Steps: how many times is GPU going to iterate the training process. Note that for each step, you may send multiple samples to the network when Batch Size is greater than 1.
- Learning Rate: the intensity for back propagation. Keep this value as default (0.0001)
- Batch Size: the number of samples to train in each step. If you are using Colab, you can set this to 2 to speed up (also half the number of steps).
- Preview Steps: how often do you like to log current preview image.
- Preview Seed: the seed for network to inference.
- LoRA Rank: the rand of LoRA. The bigger the detailed but it causes a larger LoRA model size.



→ 4. Start Training

- Press Start Training to train your LoRA.
- Cancel at any time if you want to stop.
- It starts to generate your LoRA model right after you cancel. Wait for about 30 seconds.
- The preview images are in output/images folder
- Click Reset Data to clear all uploaded data.

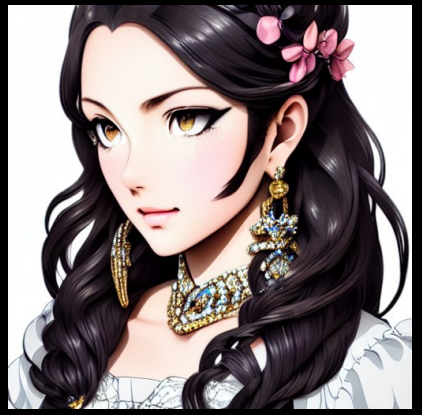
Number of Training Steps: 600
Learning Rate: 1e-4
Batch size: 1

→ Option#1: Test your LoRA model

- Run this cell to inference with your LoRA model.
- Edit the preview_prompt 1st, the {token name} is what you provided in the WebUI before. This token name MUST BE e put at the 1st word position as a trigger word.
- You may edit the negative prompt to get excellent quality.

```
# Option#1: Test the LoRA model
from diffusers import StableDiffusionPipeline, DPMSolverMultistepSchedule
import torch

preview_prompt = f"{token_name} "
negative_prompt = "grain, noise"

if pipeline is None:
  pipeline = StableDiffusionPipeline.from_pretrained(
      "stable-diffusion-v1-5/stable-diffusion-v1-5",
      torch_dtype=torch.float16,
  ).to("cuda")

pipeline.enable_xformers_memory_efficient_attention()

pipeline.load_lora_weights(output_dir, weight_name="pytorch_lora_weights.safetensors")

scheduler_args = {}
```



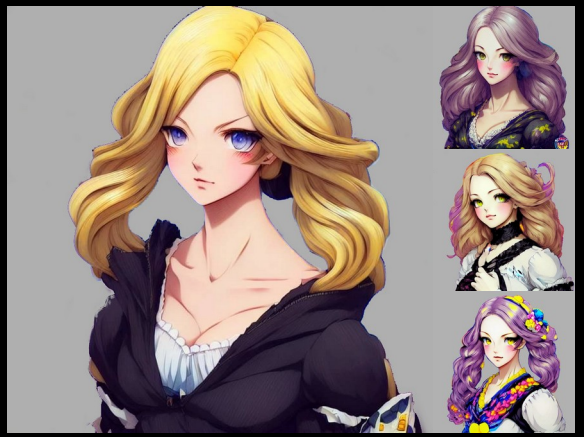
→ Option#2: Upload to your Google Drive

- If you are using Google Colab, run this cell to create a folder called LoRA in your root and rename, upload your LoRA safetensors model file.
- Put this safetensors to your A1111 model/LoRA folder to use in WebUI.

- Hyper params for experiment#A
  - → Dataset: wpg (no BG)
  - → Number of Training Steps: 100
  - → Learning Rate: 1e-3
  - → Batch Size: 1; Acc Steps: 1
  - → LoRA Rank: 32
- This settings looks for a quick training under large LR.

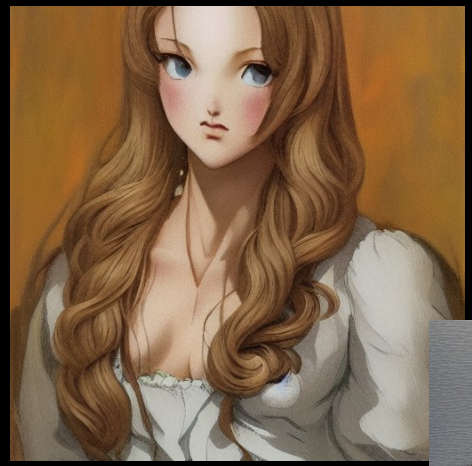
The result is so so, need to cut out more noise.

wpg



Dataset

- Empty token " " and girl are contaminated!




girl




Empty token

- The generativeness of girl and wpg

girl

girl/wpg wear a hat, eating ramen.

wpg

- The generativeness of girl and wpg

girl

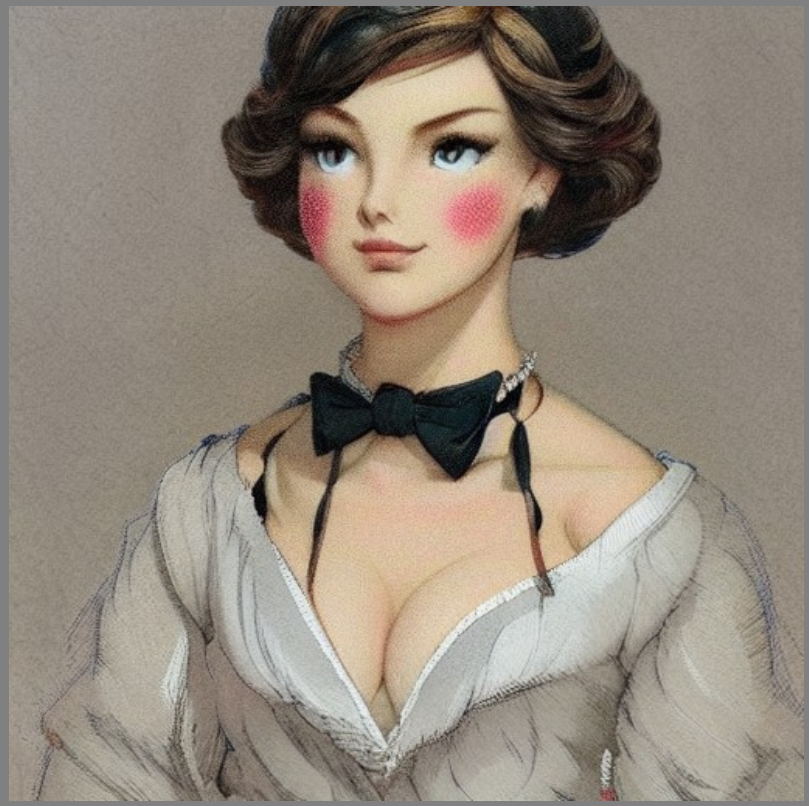
girl/wpg wear a hat, happy laugh, short hair

wpg

- The generativeness of girl and wpg

girl

girl/wpg, a bowtie, dress evening gown



wpg





- Short summary:
  - → Girl has more generativeness then wpg.
    It has a higher hit rate to create a matched image.
    - You may try "a sculpture of a girl/wpg".
    - The safty checker may send you a blackout for nudity.
  - → Even the girl is contaminated, the quality looks good
    - This means the dynamics for training wpg is enough.
      - Maybe it is because the variations between each samples makes the optimization wandering in local area.
      - Let's smooth out the loss a little bit by setting the Batch Size larger to be 2.
        - If your GPU is not capable of doing so, you may double the steps to 200 while set the accumulative steps to 2.

- Hyper params for experiment#A
  - → Dataset: wpg (no BG)
  - → Number of Training Steps: 100
  - → Learning Rate: 1e-3
  - → Batch Size: 2; Acc Steps: 1
  - → LoRA Rank: 32
- This settings looks for a quick training under large LR.

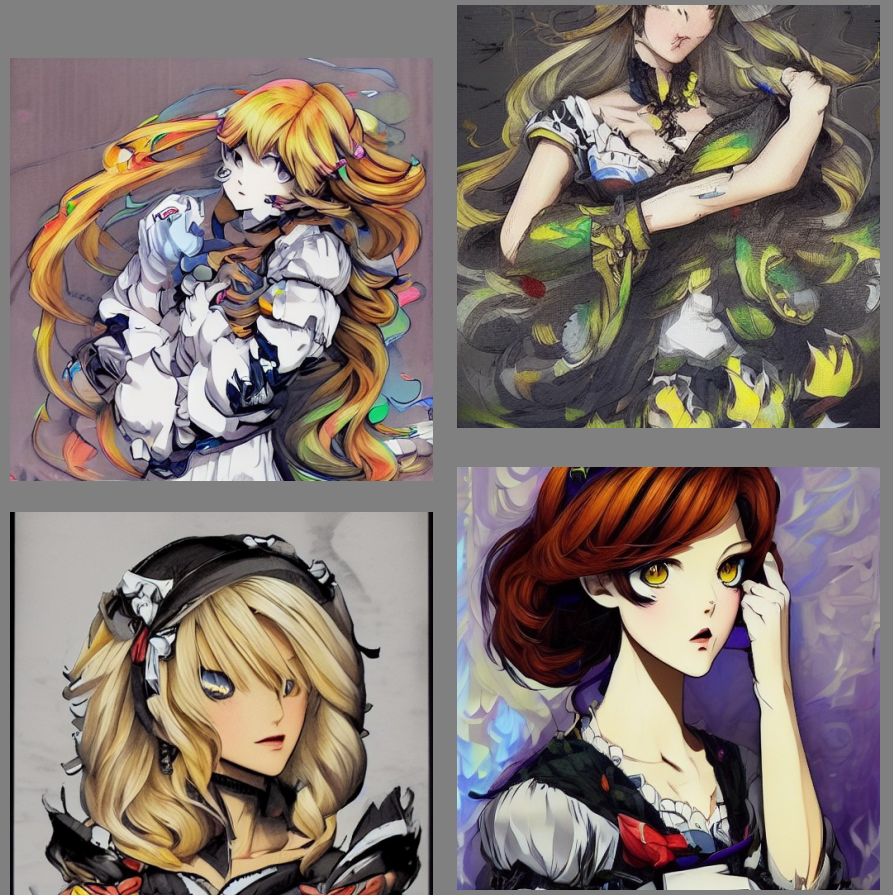
The result is more consistent!

wpg

Dataset

- The strength of girl and empty

girl



empty

- The generativeness of girl and wpg

girl

girl/wpg wear a hat, eating ramen.

wpg

- The generativeness of girl and wpg

girl

girl/wpg wear a hat, happy laugh, short hair

wpg

- The generativeness of girl and wpg

girl

girl/wpg bow, a ribbon, school uniform

wpg

- Short summary:
  - → We are satisfied about the improvement. So how to make it better and better?
  - → Let's try to set Batch Size to 4 and try again.
    - Please be noted that: Since we just have 4 images, a BS larger than 4 is not meaningful.
  - → If your GPU is not having enough VRAM to run under this Batch Size, you may set Accumulate Steps to 4 with Number of Training Steps set to 400.

- Hyper params for experiment#A
  - → Dataset: wpg (no BG)
  - → Number of Training Steps: 100
  - → Learning Rate: 1e-3
  - → Batch Size: 4; Acc Steps: 1
  - → LoRA Rank: 32
- This settings looks for a quick training under large LR.

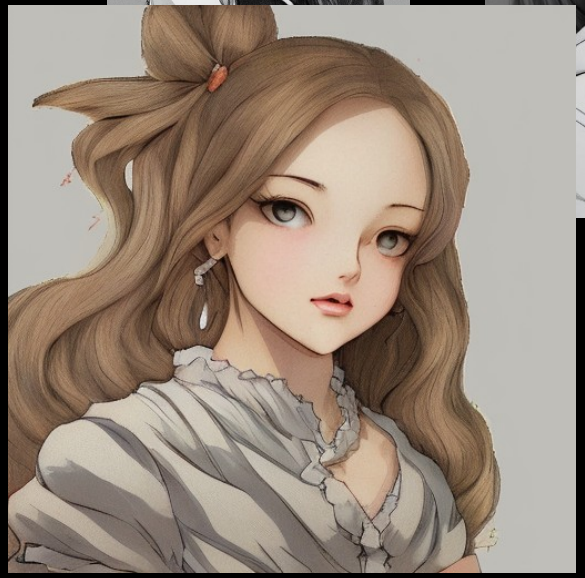
The result is more consistent!

wpg

Dataset

- The strength of girl and empty






girl




empty

- The generativeness of girl and wpg

girl

girl/wpg wear a hat, eating ramen.

wpg

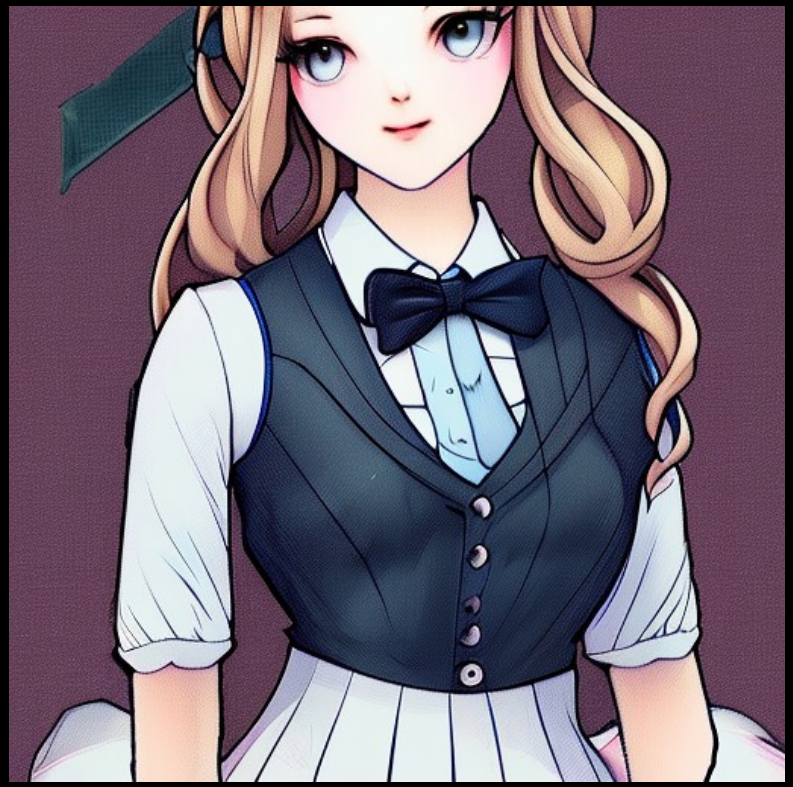
- The generativeness of girl and wpg

girl

girl/wpg wear a hat, happy laugh, short hair

wpg

- The generativeness of girl and wpg



girl

girl/wpg bow, a ribbon, school uniform

wpg

- Short summary:
  - → Photo style for character concept is damaged!
    - Now it is almost **NOT possible** to generate "a photo of girl" or "a magazine cover of a girl". All plain image medium are polluted by wpg's illustration style.
      - But on the contrary, you can still draw "a photo of car" or "photo of building".
    - If you generate "a photo of cat", you have chance to get a drawing of cat girl.
    - Guess what are the rest two pieces of below images? The answer is at the cornder.






Ans: A photo of banana.

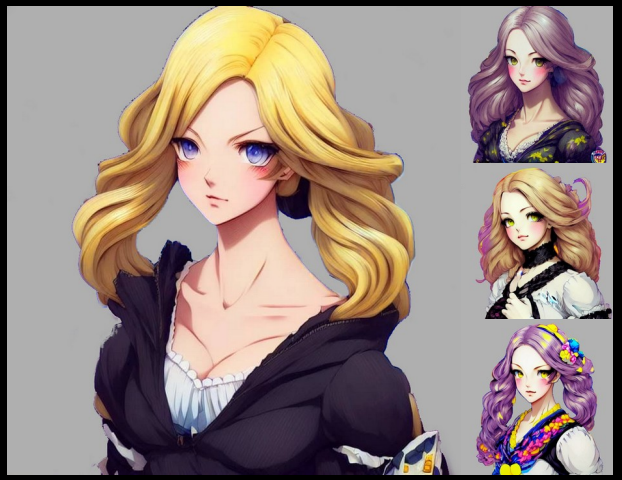
Diffusion Model Fine-tuning

- LoRA
- Inference with LoRA Model
- Discussion

- Hyper params for experiment#A
  - → Dataset: wpg (no BG)
  - → Regularizer Strength: 0 (no Reg)
  - → Number of Training Steps: 400
  - → Learning Rate: 1e-3
  - → Batch Size: 2; Acc Steps: 1
  - → LoRA Rank: 32
- This settings looks for a deeper training

Very consistency! But...

wpg



Dataset

- The strength of girl and empty

girl

Empty token

- The generativeness of girl and wpg

girl

girl/wpg wear a hat, eating ramen.

wpg

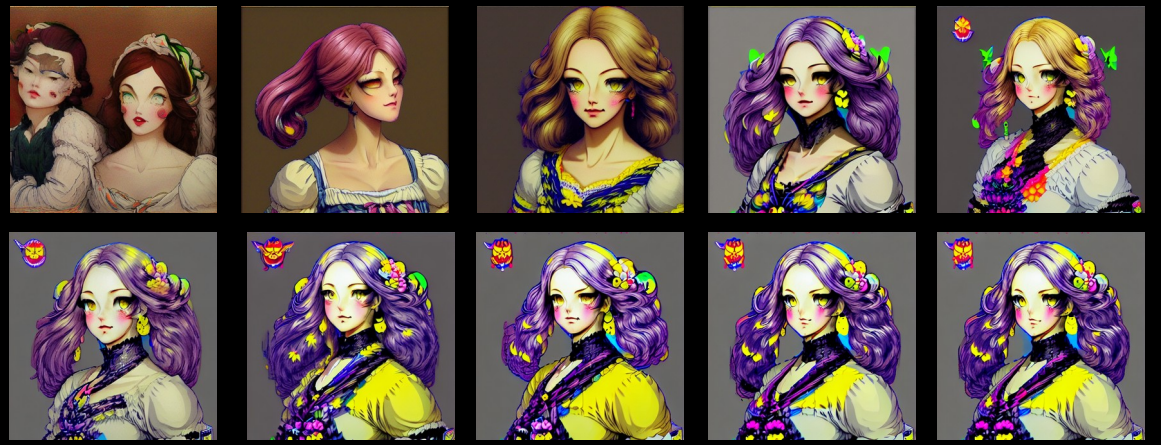
No generativeness of wpg anymore! Mid strength generativeness for girl.

- Hyper params for experiment#A
  - → Dataset: wpg (no BG)
  - → Regularizer Strength: 0.01
  - → Number of Training Steps: 400
  - → Learning Rate: 5e-4
  - → Batch Size: 2; Acc Steps: 1
  - → LoRA Rank: 32
- This settings looks for a deeper training

Let's see whether wpg can eat ramen...

girl

Dataset



- The strength of girl and empty

girl

Empty token

- The generativeness of girl and wpg

girl

girl/wpg wear a hat, eating ramen.

wpg

Now they all have some generativeness

- The generativeness of a "wpg girl"




wpg, a illustration of a girl, wears a hat, eats ramen.

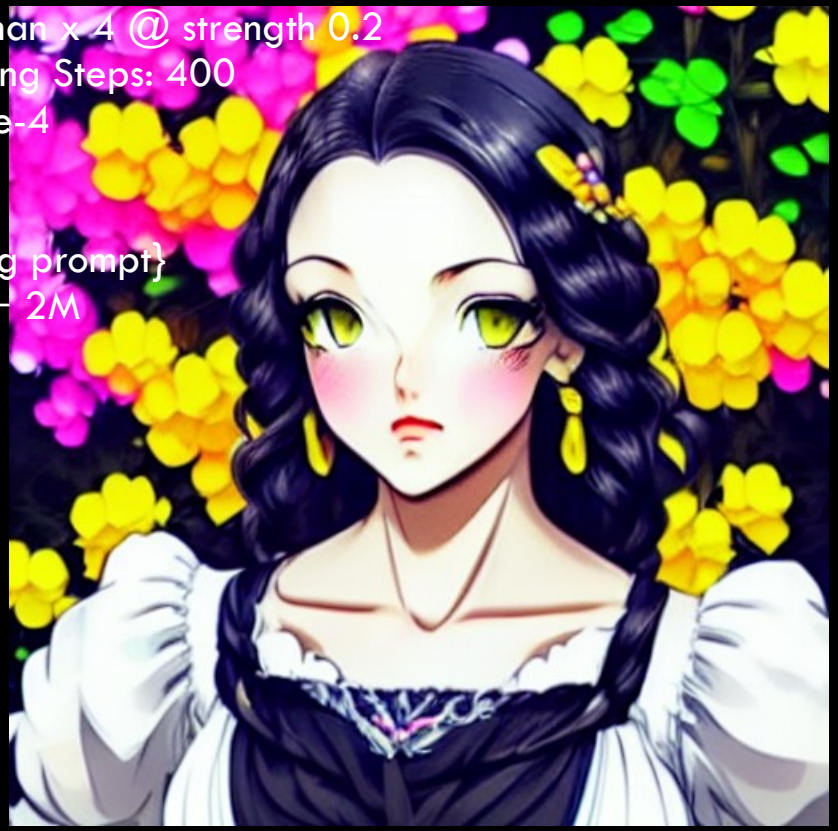
Train deeper to get more precise results.

Diffusion Model Fine-tuning

- LoRA
- Inference with LoRA Model
- Discussion

Results

Regularizer: woman x 4 @ strength 0.2
Number of Training Steps: 400
Learning Rate: 1e-4
Batch size: 1
Prompt: wpg
Neg: {default neg prompt}
Sampler: DPM++ 2M
Steps: 20
CFG: 5

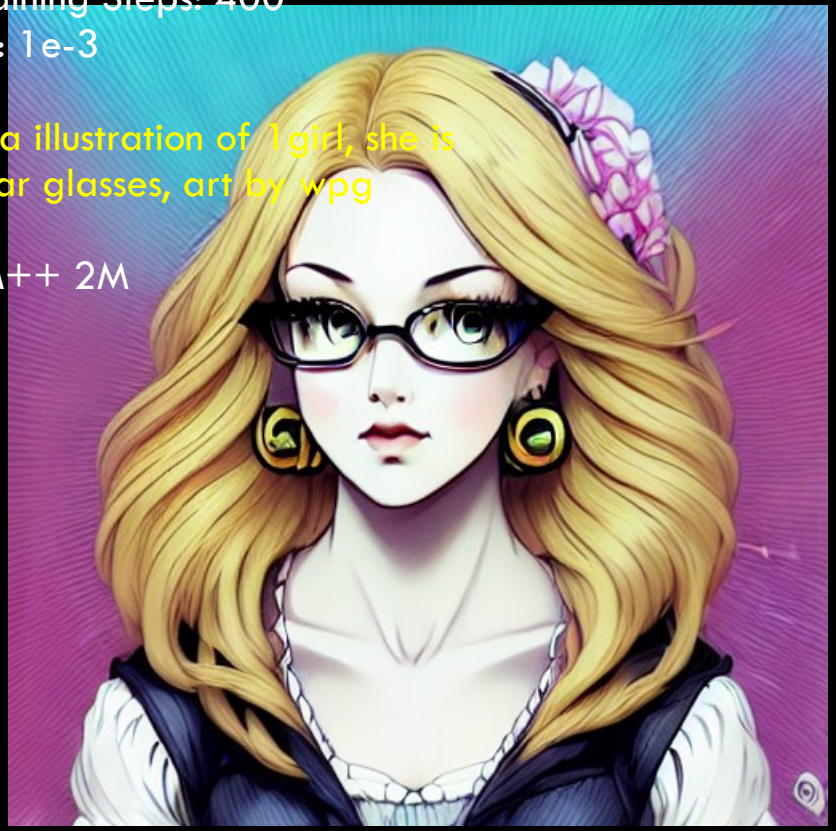


Results

Regularizer: 1girl x 100 @ strenth: 0.2
Number of Training Steps: 400
Learning Rate: 1e-3
Batch size: 4
Prompt: wpg, a illustration of 1girl, wear suit, hat, art by wpg
Neg: {empty}
Sampler: DPM++ 2M
Steps: 20
CFG: 5

Results

Regularizer: 1girl x 100 @ strenth: 0.2
Number of Training Steps: 400
Learning Rate: 1e-3
Batch size: 4
Prompt: wpg, a illustration of 1 girl, she is a teacher, wear glasses, art by wpg
Neg: {empty}
Sampler: DPM++ 2M
Steps: 20
CFG: 5



Results

Regularizer: 1girl x 100 @ strenth: 0.2
Number of Training Steps: 400
Learning Rate: 1e-3
Batch size: 4
Prompt: wpg, a film screenshot of 1girl, reading a book, art by wpg
Neg: {default negative}
Sampler: DPM++ 2M
Steps: 20
CFG: 5

Results: Neutral Objects

[person, character, role]
Regularizer: woman x 4 @ strength 0.2
Number of Training Steps: 800
Learning Rate: 1e-4
Batch size: 4
Prompt: wpg, poster of wpg female character, suit, hat, necklace. art by wpg.
Neg: {default neg prompt}
Sampler: DPM++ 2M
Steps: 20
CFG: 5

Results

-

Number of Training Steps: 400
Learning Rate: 1e-4
Batch size: 2
Prompt: wpg, A cover illustration of a woman, 8k, highly detailed, digital painting, masterpiece, best quality.
Neg: {default neg prompt}

Results

Regularizer: girl x 100
Number of Training Steps: 600
Learning Rate: 1e-4
Batch size: 1
Prompt: girl
Neg: {default neg prompt}
Sampler: DPM++ 2M
Steps: 20
CFG: 5

Q&A