Ghibli-Style Image Generation

313512039 楊翔王 313512058 邱文揚

109950006 董少鈞 410707001 李彥璋

Outline

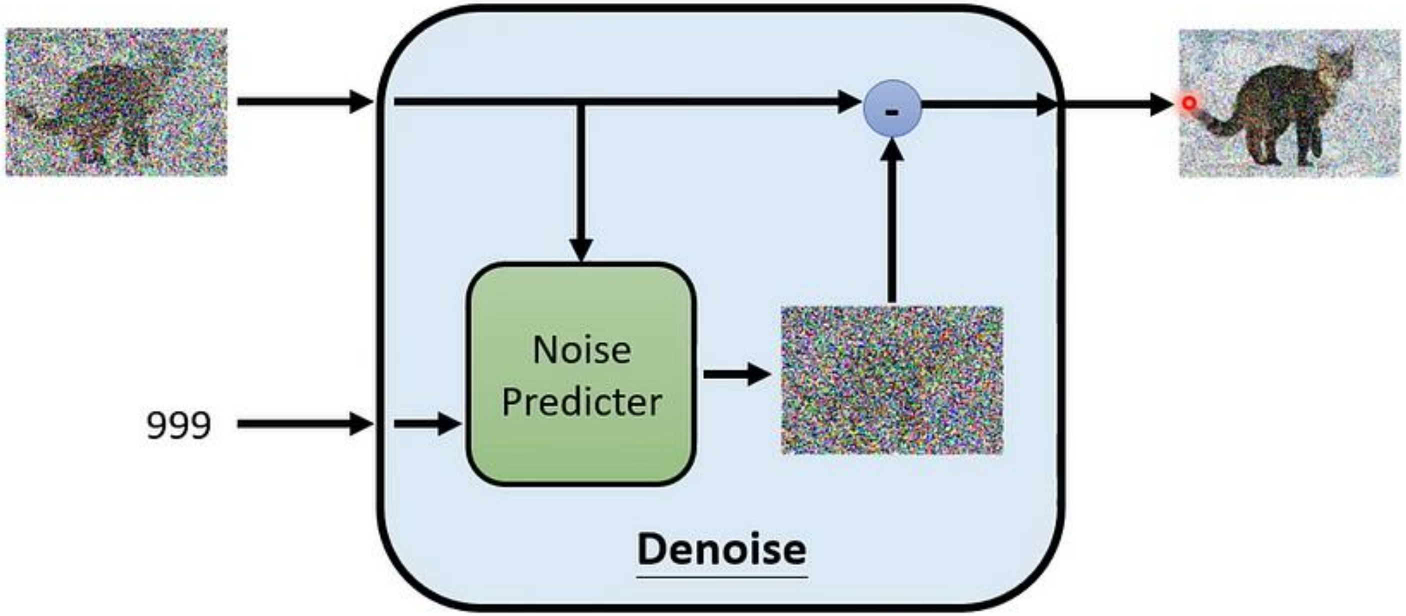
- Stable Diffusion Model
- LoRA (Low-Rank Adaptor)
- DDPM/DDIM/CFG
- Implementations and Results

Outline

- Stable Diffusion Model
- LoRA (Low-Rank Adaptor)
- DDPM/DDIM/CFG
- Implementations and Results

What is stable diffusion model

- Text-To-Image Generative
- Training：Picture → Noise
- Generation：Noise → Picture



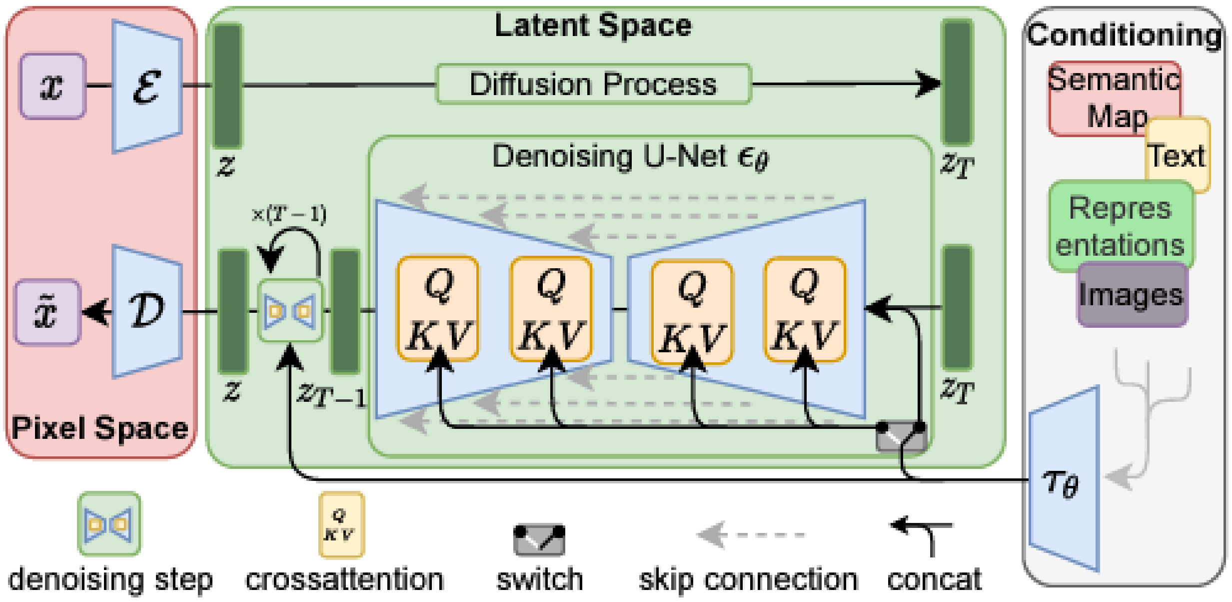
Why Do We Need Stable Diffusion?

- Text-to-Image Generation
- Boosts Creative and Design Workflows
- Cuts Costs & Replaces Stock Images
- Highly Controllable AI Tool
- Open Source and Privacy-Friendly
- Driving Innovation and Accessibility

How Stable Diffusion Works

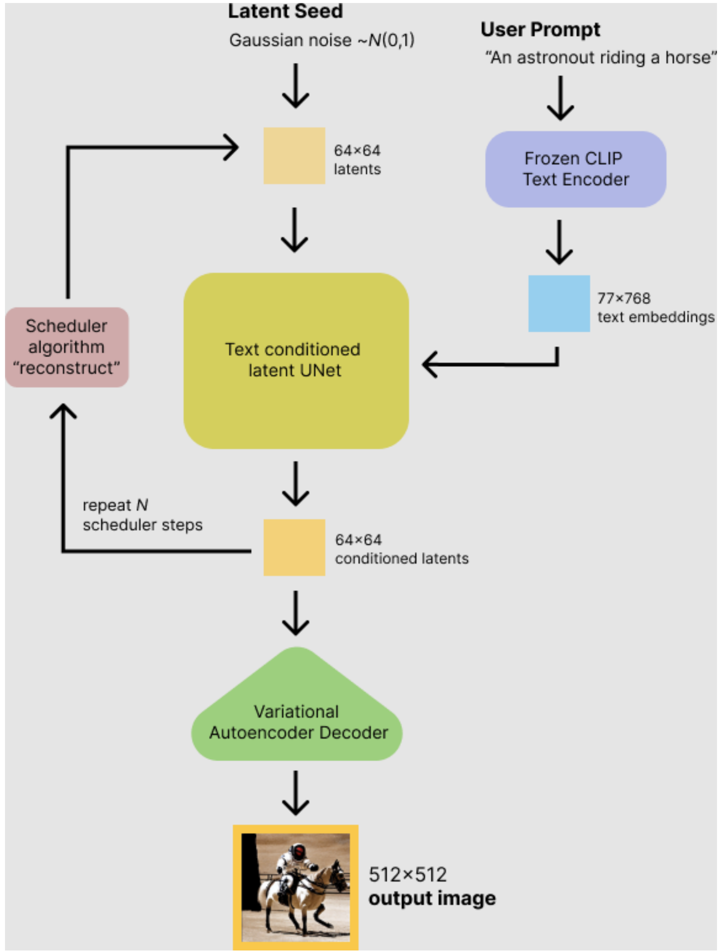
This is divided into three main sections

- Pixel Space
- Laten Space
- conditioning



Stable Diffusion Training Process

- Step 1: Generate Latent Seed (Noise)
- Step 2: Encode the Text Prompt
- Step 3: Conditional Denoising via U-Net
- Step 4: Scheduler Algorithm
- Step 5: Decode Latents into Image
- Step 6: Output Image

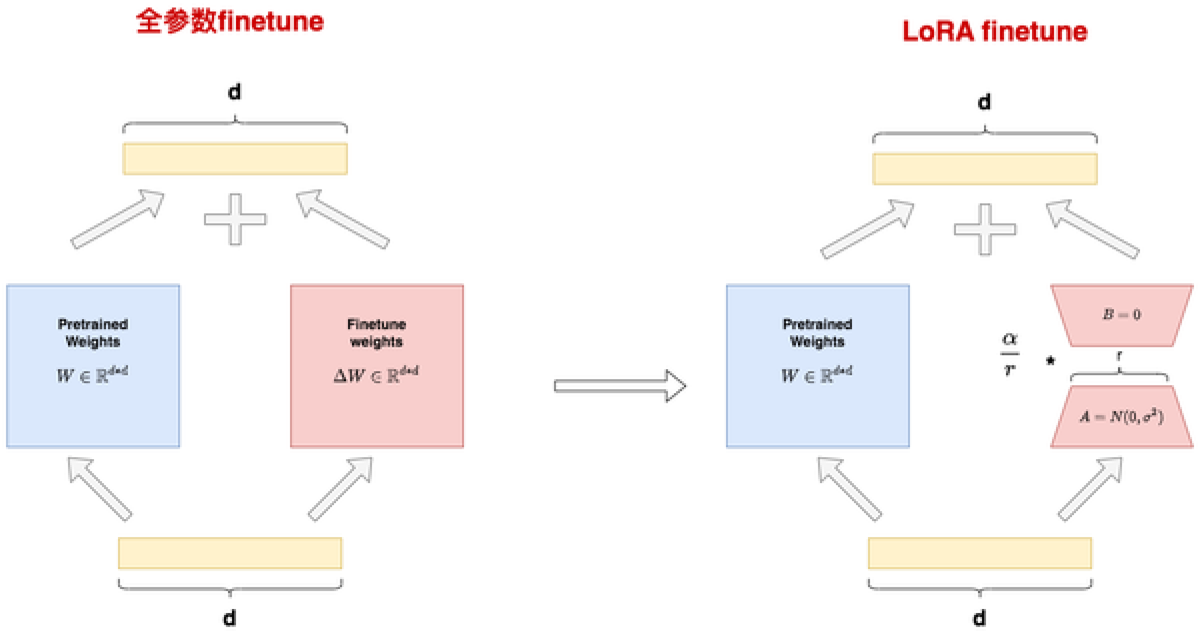


Outline

- Stable Diffusion Model
- LoRA (Low-Rank Adaptor)
- DDPM/DDIM/CFG
- Implementations and Results

What is LoRA?

- LoRA = Low-Rank Adaptation
- Efficient fine-tuning technique
- Injects trainable low-rank matrices into frozen layers
- Saves compute and storage



Why Choose LoRA?

Problems with Full Fine-Tuning

- Expensive and inefficient
- High compute cost
- Long training time
- Requires updating all model parameters

Advantages of LoRA

•Fine-tunes only a few parameters
•Keeps base model frozen
•Lower compute and memory cost
•Maintains original model performance
•Modular and reusable adapters

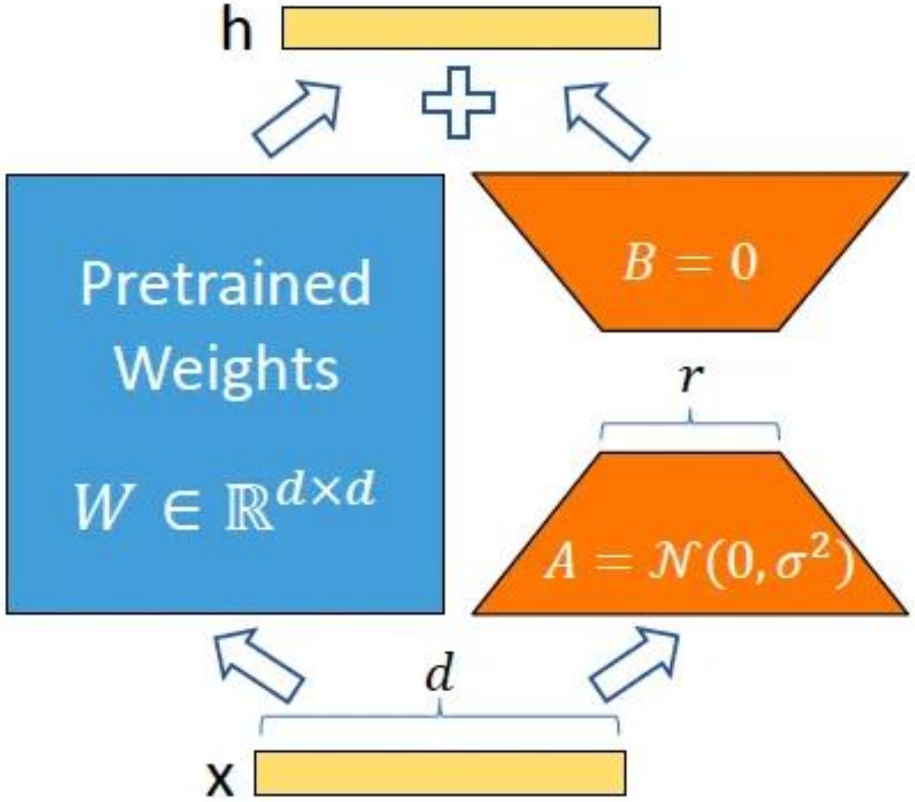
How LoRA Works

•Targets linear layers (e.g. attention, FFN)

•Adds low-rank matrices A and B:
W′=W+A·B

•Trains only A and B

•Original weights W are frozen

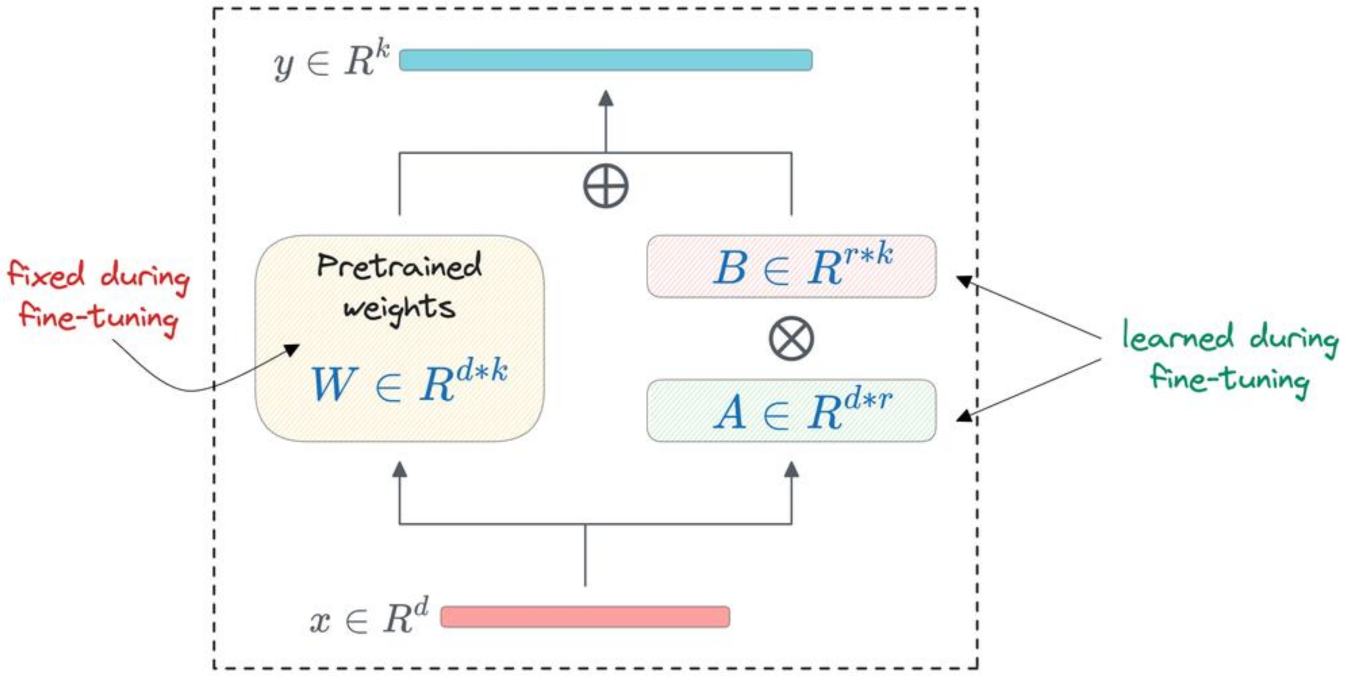


Where is LoRA Used?

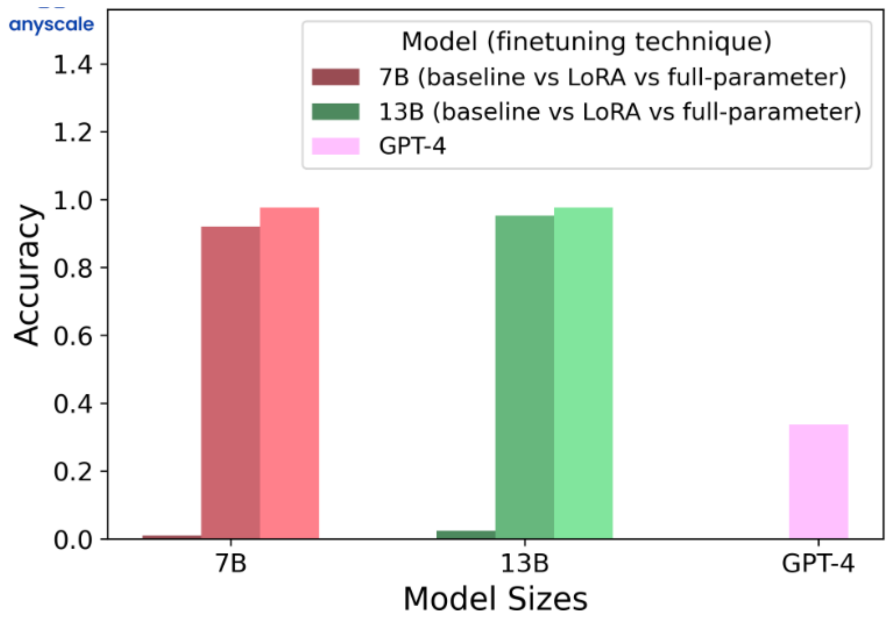
- Large Language Models: GPT, LLaMA, BERT
- Image Generation: Stable Diffusion
- Multi-task adaptation
- Plug-and-play style modules

LoRA Training Process

- •Step 1: Freeze original weights
- •Step 2: Insert low-rank matrices
- •Step 3: Train A and B only
- •Step 4: Merge updates during inference



Experimental Results



- LoRA tested on multiple NLP benchmarks
- Comparable or better than full fine-tuning
- Great tradeoff between performance and efficiency

Model	Average Score	Additional Param.	Training Time (Hour/epoch)
LLaMA-13B + LoRA(2M)	0.648	28M	8
LLaMA-7B + LoRA(4M)	0.624	17.9M	11
LLaMA-7B + LoRA(2M)	0.609	17.9M	7
LLaMA-7B + LoRA(0.6M)	0.589	17.9M	5
LLaMA-7B + FT(2M)	0.710	-	31
LLaMA-7B + LoRA(4M)	0.686	-	17
LLaMA-7B + FT(2M) + LoRA(math_0.25M)	0.729	17.9M	3
LLaMA-7B + FT(2M) + FT(math_0.25M)	0.738	-	6

可训练参数较小

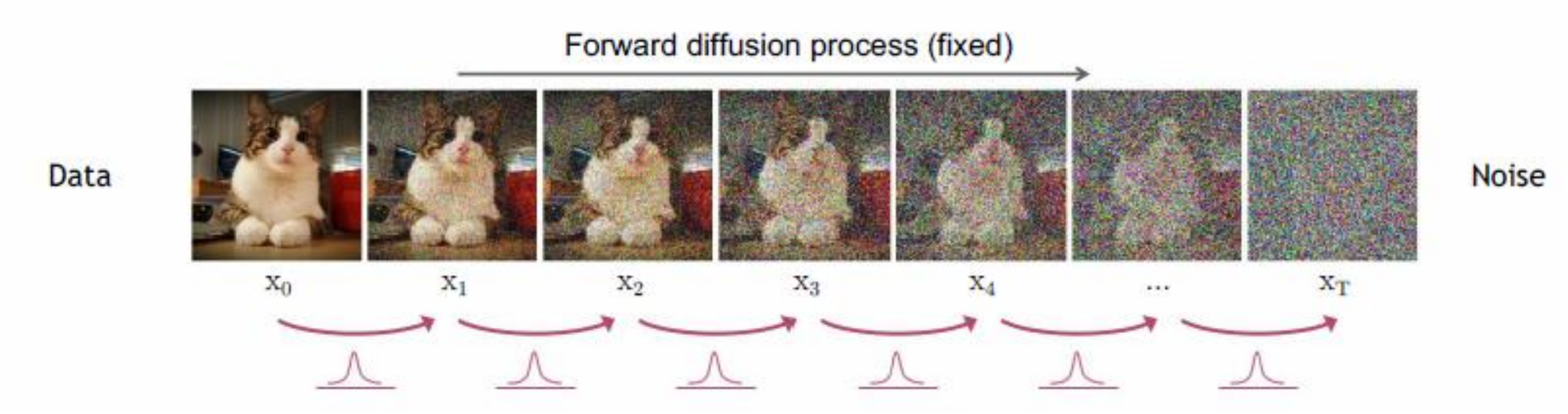
训练时长不对等

Outline

- Stable Diffusion Model
- LoRA (Low-Rank Adaptor)
- DDPM/DDIM/CFG
- Implementations and Results

DDPM (Denoising Diffusion Probabilistic Models)

Forward:

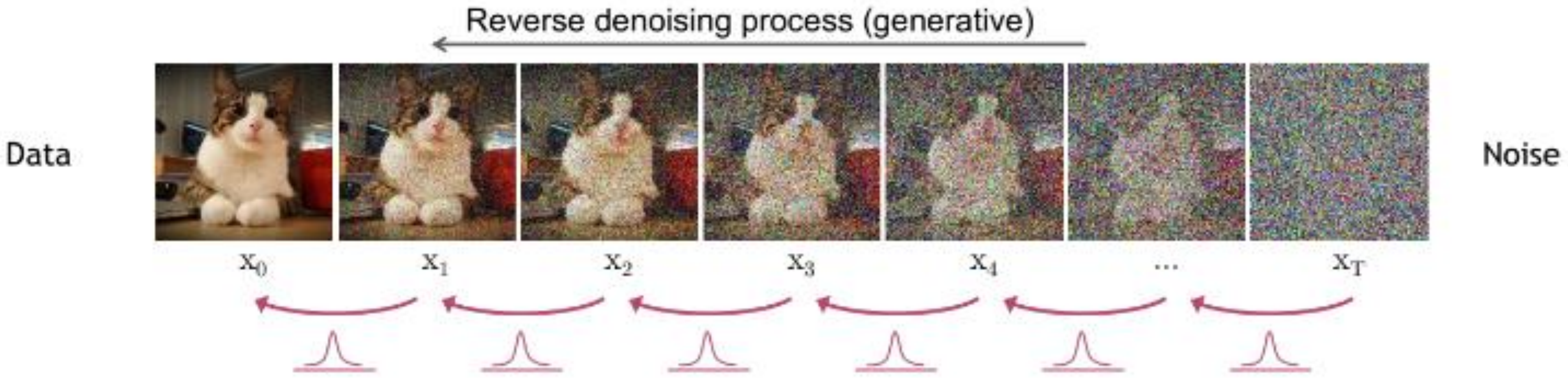


$$q(x_t | x_{t-1}) = \mathcal{N}(x_t \, ; \, \sqrt{1-\beta_t} x_{t-1} \, , \, \beta_t I)$$

$$\Rightarrow x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t} \epsilon, \quad \epsilon \sim \mathcal{N}(0,1), \, \bar{\alpha}_t = \prod_{s=1}^{t} (1-\beta_s)$$

DDPM (Denoising Diffusion Probabilistic Models)

Backward:



$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(x_{t-1} \; ; \; \mu_\theta(x_t, t) \; , \boxed{\Sigma_\theta(x_t, t)})$$

Fixed

$$\Rightarrow x_{t-1} = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boxed{\epsilon_\theta(x_t, t)}\right) + \sigma_t z, \quad z \sim \mathcal{N}(0,1)$$

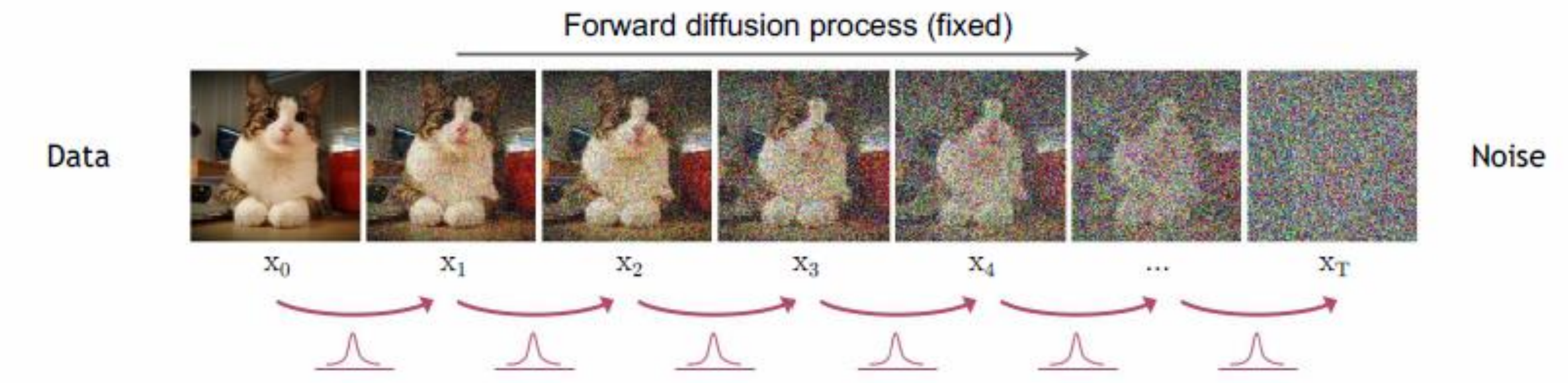
What we are training

DDIM (Denoising Diffusion Implicit Models)

Goal: Reduce the number of sampling steps

DDPM (Denoising Diffusion Probabilistic Models)

Forward:



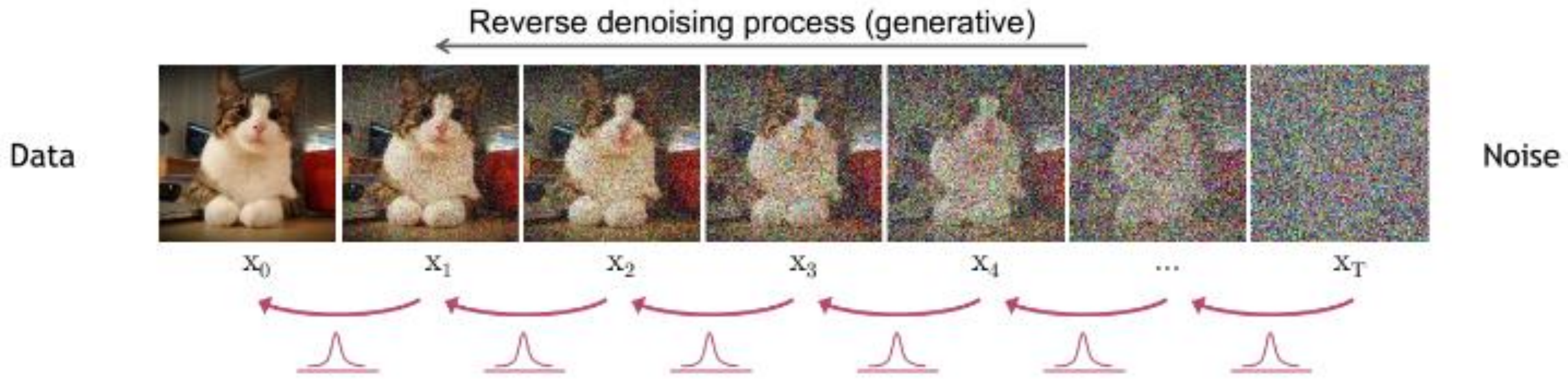
$$q(x_t | x_{t-1}) = \mathcal{N}(x_t \, ; \, \sqrt{1-\beta_t} x_{t-1} \, , \, \beta_t I)$$

$$\boxed{\Rightarrow x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, \quad \epsilon \sim \mathcal{N}(0,1), \, \bar{\alpha}_t = \prod_{s=1}^{t} (1-\beta_s)}$$

Recall this part, we do have a closed-form solution

DDIM (Denoising Diffusion Implicit Models)

Backward:

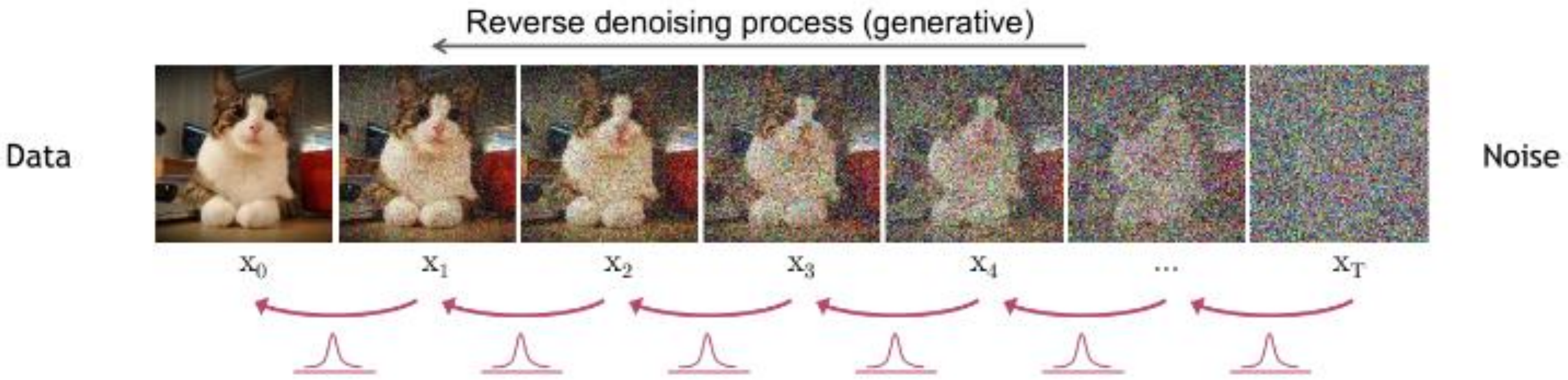


$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$$

$$\Rightarrow x_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}(x_t - \sqrt{1-\bar{\alpha}_t}\epsilon)$$

DDIM (Denoising Diffusion Implicit Models)

Backward:



$$\begin{cases} x_{t-k} = \sqrt{\bar{\alpha}_{t-k}}\, x_0 + \sqrt{1-\bar{\alpha}_{t-k}}\,\epsilon_\theta \\ x_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}(x_t - \sqrt{1-\bar{\alpha}_t}\,\epsilon_\theta(x_t, t)) \end{cases} \Rightarrow x_{t-k} = \sqrt{\bar{\alpha}_{t-k}}\left(\frac{x_t - \sqrt{1-\bar{\alpha}_t}\,\boxed{\epsilon_\theta(x_t, t)}}{\sqrt{\alpha_t}}\right) + \sqrt{1-\bar{\alpha}_{t-k}}\,\boxed{\epsilon_\theta(x_t, t)}$$

What we are training

CFG (Classifier-Free Guidance)

Goal: Improve how well the diffusion model aligns with the prompt

Traditionally, an external classifier is required to guide the generation process.

With CFG, we no longer need an additional classifier.

CFG (Classifier-Free Guidance)

Approach: During training, the model is exposed to both conditional and unconditional inputs. At inference time, predictions from both are combined to guide the generation path.

CFG (Classifier-Free Guidance)

$(x_t, t, c = prompt)$

$(x_t, t, c = \varnothing)$

Noise Prediction

$\epsilon(x_t, t, c = prompt)$

$\epsilon(x_t, t, c = \varnothing)$

$$\epsilon_{final} = (1 + w)\epsilon_{cond} - w\epsilon_{uncond}$$

Outline

- Stable Diffusion Model
- LoRA (Low-Rank Adaptor)
- DDPM/DDIM/CFG
- Implementations and Results

Dataset

Real to Ghibli Image Dataset

▲ 32 | <> Code

Data Card | Code (5) | Discussion (0) | Suggestions (0)

About Dataset

📌 Real to Ghibli Image Dataset (5K High-Quality Images)

📖 Overview

The **Real to Ghibli Image Dataset** is a high-quality collection of **5,000 images** designed for **AI-driven style transfer and artistic transformations**. This dataset is ideal for training **GANs, CycleGAN, diffusion models, and other deep learning applications** in image-to-image translation.

LoRA Fine-Tuning Results on Stable Diffusion

- 90% of data used for LoRA fine-tuning, 10% for FID and loss evaluation
- Results of an un-tuned model (baseline)
- Results of full model fine-tuning
- Fine-tuning results with two different loss functions
  - DDPM & DDIM
- Guidance Scale(1 and 7.5)

Without Fine Tuning

DDPM

A girl standing in a greenhouse,
Studio Ghibli style

DDIM

A panda on the top of the
train ,Studio Ghibli style

DDPM

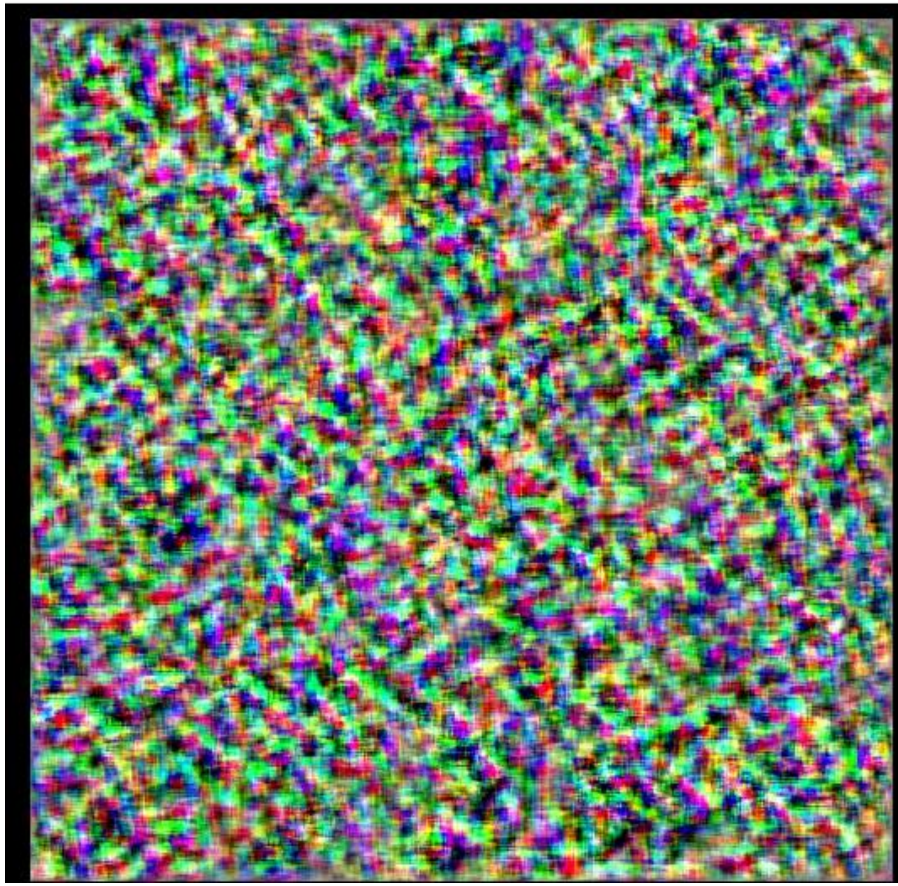
DDIM

Results of Full Model Fine-Tuning

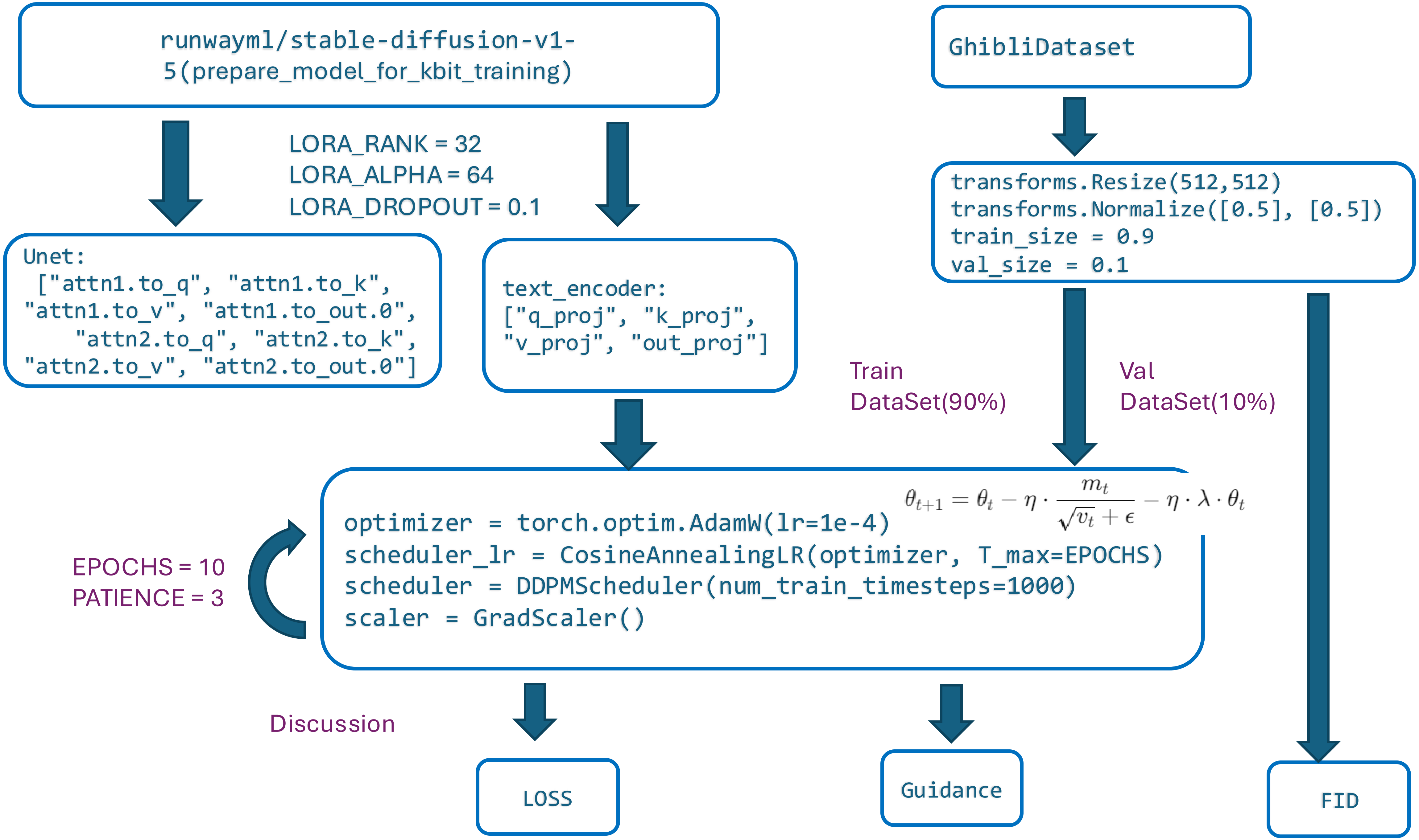
```python
# ✅ 生成圖片
prompt = "a girl standing in a greenhouse, Studio Ghibli style"
image = pipe(prompt, num_inference_steps=30, guidance_scale=7.5).images[0]

# ✅ 顯示與儲存
image.show()
image.save("ghibli_output.png")
```

✓ 3.8s

100% 30/30 [00:03<00:00, 8.97it/s]



```
🚀 Epoch 1/10
100%|          | 2500/2500 [22:05<00:00,  1.89it/s]
✅ Epoch 1 Loss: 1.0126

🚀 Epoch 2/10
100%|          | 2500/2500 [22:00<00:00,  1.89it/s]
✅ Epoch 2 Loss: 1.0000

🚀 Epoch 3/10
100%|          | 2500/2500 [22:00<00:00,  1.89it/s]
✅ Epoch 3 Loss: 0.9613

🚀 Epoch 4/10
100%|          | 2500/2500 [22:03<00:00,  1.89it/s]
✅ Epoch 4 Loss: 1.0017

🚀 Epoch 5/10
100%|          | 2500/2500 [22:02<00:00,  1.89it/s]
✅ Epoch 5 Loss: 1.0114

🚀 Epoch 6/10
100%|          | 2500/2500 [22:05<00:00,  1.89it/s]
✅ Epoch 6 Loss: 0.9946

🚀 Epoch 7/10
100%|          | 2500/2500 [22:05<00:00,  1.89it/s]
✅ Epoch 7 Loss: 0.1415

🚀 Epoch 8/10
100%|          | 2500/2500 [22:02<00:00,  1.89it/s]
✅ Epoch 8 Loss: 0.0022

🚀 Epoch 9/10
100%|          | 2500/2500 [22:06<00:00,  1.89it/s]
✅ Epoch 9 Loss: 1.0152

🚀 Epoch 10/10
100%|          | 2500/2500 [21:59<00:00,  1.90it/s]
✅ Epoch 10 Loss: 0.9994
```


runwayml/stable-diffusion-v1-5(prepare_model_for_kbit_training)
GhibliDataset
LORA_RANK = 32
LORA_ALPHA = 64
LORA_DROPOUT = 0.1
Unet: ["attn1.to_q", "attn1.to_k", "attn1.to_v", "attn1.to_out.0", "attn2.to_q", "attn2.to_k", "attn2.to_v", "attn2.to_out.0"]
text_encoder: ["q_proj", "k_proj", "v_proj", "out_proj"]
transforms.Resize(512,512)
transforms.Normalize([0.5], [0.5])
train_size = 0.9
val_size = 0.1
Train DataSet(90%)
Val DataSet(10%)
$\theta_{t+1} = \theta_t - \eta \cdot \frac{m_t}{\sqrt{v_t} + \epsilon} - \eta \cdot \lambda \cdot \theta_t$
optimizer = torch.optim.AdamW(lr=1e-4)
scheduler_lr = CosineAnnealingLR(optimizer, T_max=EPOCHS)
scheduler = DDPMScheduler(num_train_timesteps=1000)
scaler = GradScaler()
EPOCHS = 10
PATIENCE = 3
Discussion
LOSS
Guidance
FID

Loss Function of Fine Tuning

- Apply on Unet
- Guidance_scale = 1
  - Code:

```
noise_guided  = noise_pred_cond
loss = nn.functional.mse_loss(noise_guided, noise)
```

  - Direct MSE loss between predicted and actual noise.
  - Typically used for standard denoising.

- Guidance_scale = 7.5
  - Code:

```
noise_guided = noise_pred_uncond + guidance_scale * (noise_pred_cond - noise_pred_uncond)
loss = nn.functional.mse_loss(noise_guided, noise)
```

  - Guided noise prediction with conditional information.

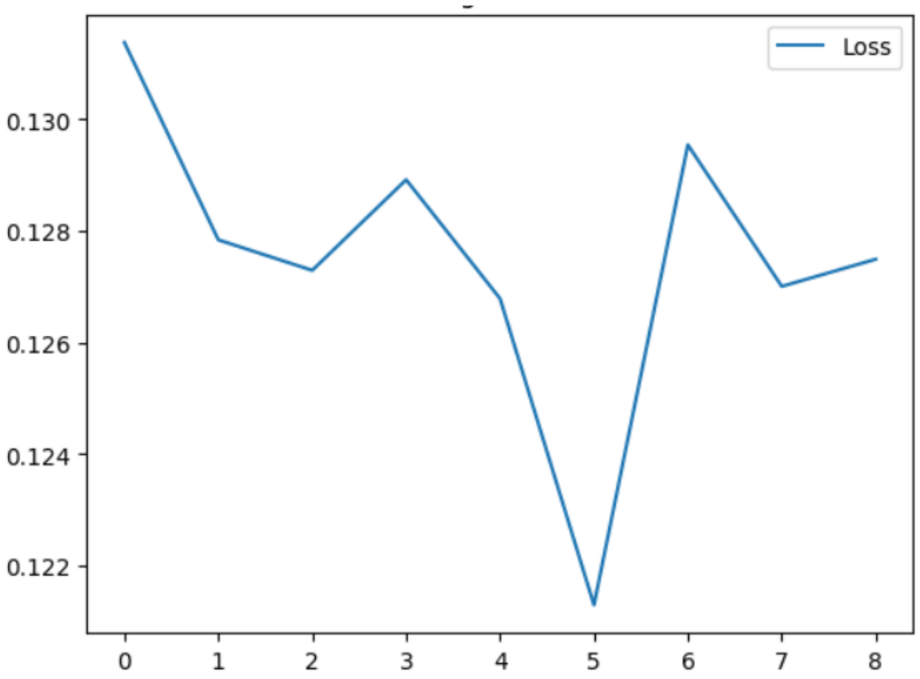
LOSS Anomaly
-Unnoticeable decrease in loss

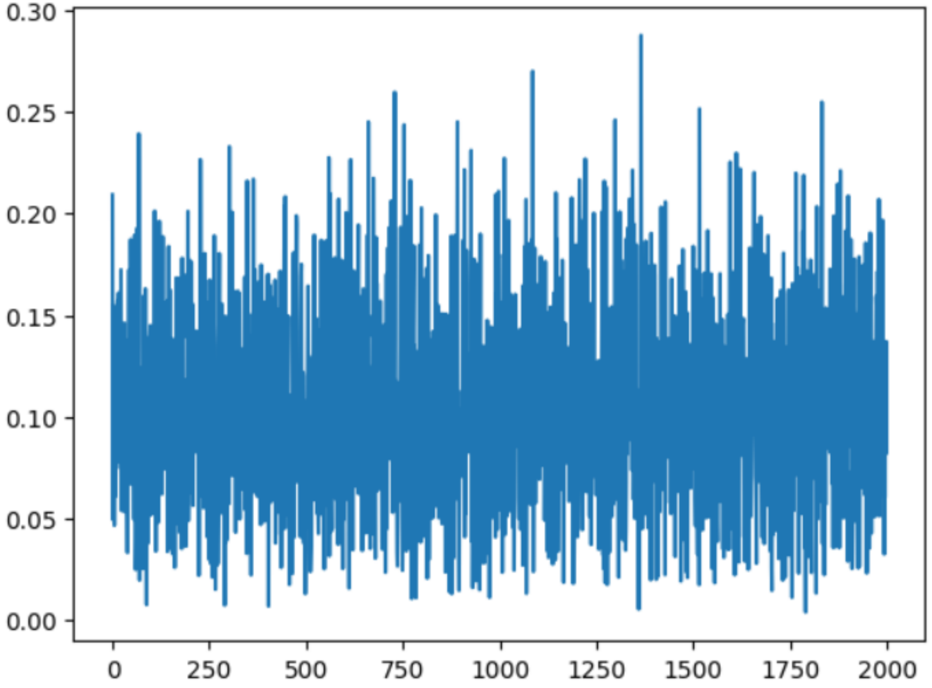
Attempted Approach[But fail]:

- Epoch (10->50)
- RANK(16,32)
- CFG(1 and 7.5)
- Learning Rate(10^{-4} -> 10^{-3})
- Dropout ratio (0 and 0.1)

Our speculation

- The pre-trained model's initial loss is near optimal.
- A small dataset leads to rapid model fitting.

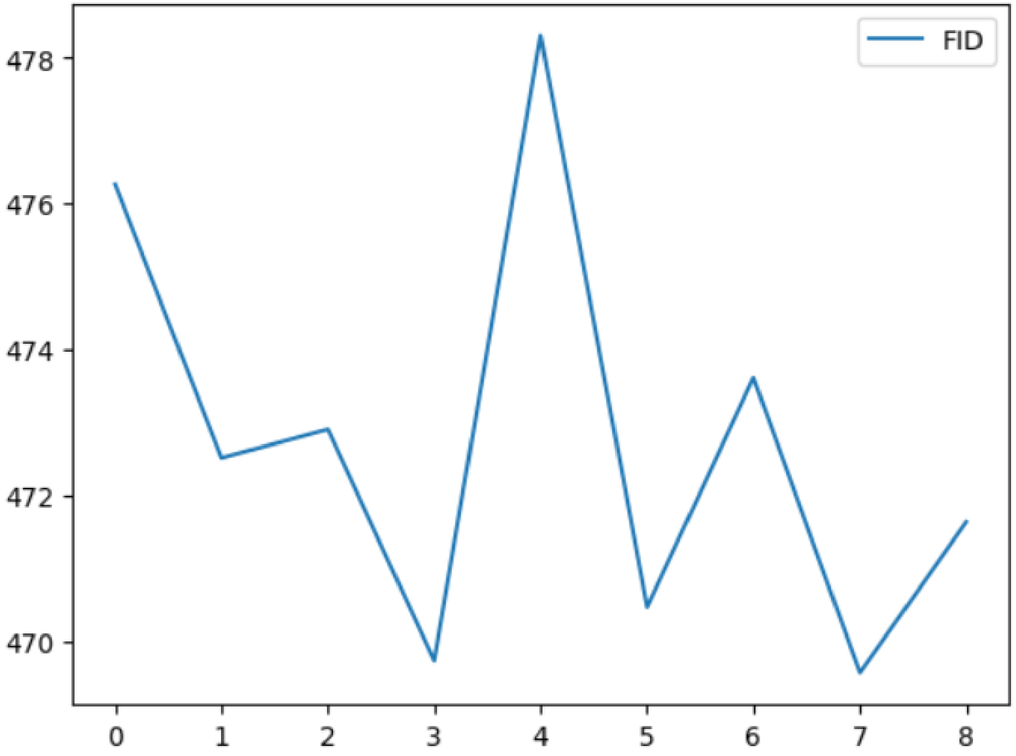




FID

- FID measures the similarity between the distributions of real and generated images.
- It computes the Fréchet distance between the feature distributions of real and generated images, using Inception V3 for feature extraction.

$$FID(x,g) = \left|\mu_x - \mu_g\right|^2 + Tr\left(\Sigma_x + \Sigma_g - 2\sqrt{\Sigma_x \Sigma_g}\right)$$



Fine Tuning Guidance scale = 1 Results

DDPM, guidance scale=1.

DDPM, guidance scale=7.5

DDIM, guidance scale=1.0

DDIM, guidance scale=7.5

Fine Tuning Guidance scale = 1 Results

DDPM, guidance scale=1.0

DDPM, guidance scale=7.5

DDIM, guidance scale=1.0

DDIM, guidance scale=7.5

Fine Tuning Guidance scale = 7.5 Results

DDPM, guidance scale=1.0

DDPM, guidance scale=7.5

DDIM, guidance scale=1.0

DDIM, guidance scale=7.5

Fine Tuning Guidance scale = 7.5 Results

DDPM, guidance scale=1.0

DDPM, guidance scale=7.5

DDIM, guidance scale=1.0

DDIM, guidance scale=7.5

Reference

1. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 10684-10695).
2. https://www.anyscale.com/blog/fine-tuning-llms-lora-or-full-parameter-an-in-depth-analysis-with-llama-2?utm_source=chatgpt.com
3. https://github.com/LianjiaTech/BELLE/issues/277
4. https://zhuanlan.zhihu.com/p/1896478803016009266
5. https://blog.csdn.net/andy20160103/article/details/138357437
6. Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS 2020. https://arxiv.org/abs/2006.11239
7. Song et al., Denoising Diffusion Implicit Models, ICLR 2021. https://arxiv.org/abs/2010.02502
8. Nichol & Dhariwal, GLIDE, arXiv 2021 (Appendix: Classifier-Free Guidance). https://arxiv.org/abs/2112.10741

分工表

姓名	工作
楊翔王	研讀及報告Stable Diffusion Model
邱文揚	引進Stable Diffusion Model 模型程式架構、題目發想、資料集預處理、報告LoRA
董少鈞	引進CFG, DDPM, DDIM、微調訓練過程、尋找資料集、報告DDPM/DDIM/CFG、組長、協調工作
李彥璋	引進LoRA、程式實作、結果分析、微調訓練過程、 提供GPU算力