Lab 9: Stable diffusion customizing

CHI YEONG HEO[1],
[1]School of Computer Science and Engineering, Pusan National University, Busan 46241 Republic of Korea

1. Introduction

This report provides an analysis of Stable Diffusion [2] and fine-tuning process of it with Low-rank Adaptation (LoRA) [1] method.

2. Stable Diffusion

2.1. Overview

Stable Diffusion is a state-of-the-art generative model based on the diffusion process. Unlike traditional generative adversarial networks (GANs), it employs a denoising approach where the model learns to reverse a noise-adding process applied to data. This method allows for high-quality image generation from latent representations, making it both computationally efficient and versatile.

2.2. Latent Diffusion Model (LDM)

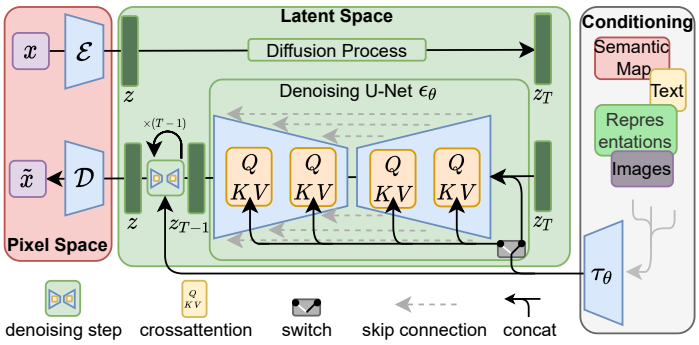

Figure 1: Model architecture of LDM.

The Latent Diffusion Model (LDM) forms the backbone of the Stable Diffusion architecture, enabling efficient image synthesis through operations in a compressed latent space. This design contrasts with traditional diffusion models, which operate directly in the high-dimensional pixel space, leading to increased computational demands.

Core Principles of Diffusion Models. Diffusion models aim to learn the data distribution $p(x)$ by progressively denoising a Gaussian noise variable. The training process involves reversing a fixed Markov chain of length T, with the denoising process modeled as a sequence of autoencoders. The objective is simplified to minimizing the difference between the model's prediction and the added noise:

$$\mathcal{L}_{\text{simple}} = \mathbb{E}_{x,\varepsilon\sim\mathcal{N}(0,1),t}\left[\|\varepsilon - \varepsilon_\theta(x_t,t)\|_2^2\right],$$

where t is uniformly sampled, x_t is a noisy version of the input x, and ε represents the Gaussian noise.

Shifting to Latent Space. LDMs leverage a low-dimensional latent space derived from a perceptual compression model comprising an encoder-decoder pair. The encoder $\mathcal{E}$ maps input images to a compact latent representation, while the decoder $\mathcal{D}$ reconstructs images from this latent space. By operating in this latent space, LDMs focus on semantically meaningful data and avoid high-frequency noise, achieving computational efficiency without sacrificing quality.

Generative Modeling in Latent Space. Unlike autoregressive models that depend on attention-based transformers in discrete latent spaces, LDMs utilize inductive biases specific to image data. The UNet backbone, primarily built with 2D convolutional layers, serves as the neural network $\varepsilon_\theta(z_t,t)$ for learning the denoising process in the latent space. The loss function adapts to this latent setting:

$$\mathcal{L}_{\text{LDM}} = \mathbb{E}_{\mathcal{E}(x),\varepsilon\sim\mathcal{N}(0,1),t}\left[\|\varepsilon - \varepsilon_\theta(z_t,t)\|_2^2\right],$$

where z_t is the latent representation at step t.

Advantages of Latent Diffusion. Operating in latent space provides several benefits:

- *Dimensionality Reduction:* Lower-dimensional representations lead to significant computational savings.
- *Semantic Focus:* The compression process abstracts away imperceptible details, allowing the model to prioritize perceptually relevant features.
- *Efficient Sampling:* A single forward pass through the decoder $\mathcal{D}$ translates samples from $p(z)$ back to image space.

The combination of perceptual compression and diffusion in the latent space makes LDMs a powerful framework for efficient, high-quality image synthesis. This foundation supports further enhancements, such as conditioning mechanisms for guided image generation, which are explored in subsequent sections.

2.3. Training Process

The training of Stable Diffusion involves a multi-stage procedure that combines VAEs and denoising score matching. Initially, the model learns to progressively add noise to the data, simulating the diffusion process. In the reverse stage, it is trained to denoise and recover the original data. The training aims to minimize the difference between the noisy version of the data and the model's denoising output, enabling the model to capture complex data distributions. This process is carried out in a latent space, allowing for efficient computation without compromising quality. Specifically, the model learns to predict the noise at each step of the process, gradually refining its ability to synthesize data that aligns with the original distribution.

2.4. Applications

Stable Diffusion excels in various generative tasks, particularly in high-quality image generation from text descriptions. Its ability to generate realistic images from textual prompts has made it a powerful tool for creative industries, such as art and design. Additionally, it is well-suited for applications in style transfer, image inpainting, and photo enhancement, where users can guide the generation process to meet specific requirements. The model's flexibility allows it to be fine-tuned for domain-specific tasks, such as medical image synthesis or fashion design, making it highly adaptable across different industries. Its efficiency and scalability also make it suitable for integration into larger generative workflows, enabling real-time applications in areas like virtual reality, gaming, and content creation.

3. Low-rank Adaptation (LoRA)

Low-rank Adaptation (LoRA) is a parameter-efficient technique for fine-tuning large-scale models like Stable Diffusion. Instead of updating all parameters during the customization process, LoRA injects trainable low-rank matrices into the model's architecture. These matrices adjust the weights of pre-trained layers without requiring a full retraining, resulting in significant computational savings.

The key idea behind LoRA is to decompose the weight updates into two low-rank components. By doing so, it reduces the number of trainable parameters, making fine-tuning feasible even with limited hardware resources. LoRA is especially advantageous in scenarios where the target domain requires subtle adjustments to the pre-trained model's behavior.

In the context of Stable Diffusion, LoRA can be applied to specific layers or modules to adapt the model for domain-specific tasks. This ensures that the core generative capabilities of the model remain intact while tailoring its outputs to meet specialized requirements. Additionally, LoRA's efficiency makes it an ideal choice for rapid prototyping and experimentation in generative tasks.

4. Implementation Details

The implementation of Stable Diffusion customization involves three main stages: dataset preparation, sub-model definition, and fine-tuning using LoRA. Each stage is detailed below to outline the methodology and relevant implementation components.

4.1. Dataset Preparation

To train the model on custom data, a dataset of "8-bit illustrations" is created. The preparation process consists of several steps:

Image Collection: Images related to the query "8-bit illustration" are collected using the `bing-image-downloader` library. The collected images are stored in a dedicated folder (`custom_dataset_folder_name`) with standardized filenames for consistency.

Caption Generation: Each image is captioned using a pre-trained image captioning model, specifically `Salesforce/blip-image-captioning-base`. The model processes images through a `BlipProcessor` and generates textual descriptions. These captions are then collated into a metadata file in CSV format with two columns: *file_name* and *caption*.

Integration with Hugging Face Datasets: The custom dataset is structured for training by converting it into a format compatible with the `datasets` library from Hugging Face. This allows seamless interaction with downstream training workflows for Stable Diffusion customization.

4.2. Model Components and Initialization

The customization process revolves around the Stable Diffusion architecture, which comprises several sub-models detailed below:

Pre-trained Base Model: The backbone of the system is the pre-trained model `runwayml/stable-diffusion-v1-5`. This model serves as the foundation for all fine-tuning processes.

Core Components: The sub-models defining the diffusion process include:

- **Noise Scheduler:** Initialized using `DDPMScheduler`, this component manages the diffusion timesteps and noise magnitudes for training.
- **Variational Autoencoder (VAE):** The `stabilityai/sd-vae-ft-mse` VAE is used to encode and decode images into latent space during training and inference.
- **U-Net Denoiser:** The `UNet2DConditionModel` serves as the primary denoising architecture, operating in the latent space to refine image representations.

Prompt Encoding: Prompts are processed using dedicated components:

- A tokenizer is loaded using `AutoTokenizer`.
- A text encoder is initialized using `CLIPTextModel` to generate prompt embeddings from tokenized text.
- Supporting utility functions, such as `tokenize_prompt` and `encode_prompt`, are implemented to handle prompt preprocessing and embedding generation.

Parameter Freezing: The VAE, text encoder, and U-Net are pretrained and frozen during the fine-tuning process by setting `requires_grad_ = False`. This ensures that only LoRA weights are updated.

4.3. LoRA-Based Fine-Tuning

Fine-tuning is conducted using the LoRA technique, which introduces trainable low-rank weight matrices within the attention layers of the U-Net. The key details of this process are listed below:

LoRA Configuration: The LoRA layers are added to U-Net's attention modules, specifically to the `to_k`, `to_q`, `to_v`, and `to_out.0` layers. The configuration includes:

- Rank: $r = 4$
- Scaling Factor: `lora_alpha` $= 4$
- Initialization: Gaussian distribution-based initialization.

Training Configuration: Training is performed with the following parameters:

- Learning rate: `learning_rate` $=$ 1e-4
- Batch size: `train_batch_size` $= 2$
- Resolution: 1024×1024
- Maximum training steps: `max_train_steps` $= 10{,}000$
- Checkpointing frequency: 5000 steps.

Data Augmentation and Preprocessing: Training images undergo several augmentations using the `torchvision.transforms` module:

- Resizing to 1024×1024 resolution.
- Random horizontal flips ($p = 0.5$).
- Center cropping and pixel normalization.

Text captions for the training dataset are tokenized into input embeddings using the tokenizer and encoder configurations described earlier.

Optimizer and Scheduler: AdamW is used as the optimizer with a weight decay of 1e-2, and a cosine learning rate scheduler controls the learning rate throughout training. Gradient clipping is applied to stabilize training.

Accelerator Integration: The `Accelerator` API from Hugging Face is employed for distributed training and mixed precision. It simplifies multi-GPU/TPU environments and efficiently manages gradient synchronization across devices.

LoRA Weight Management: Custom saving and loading mechanisms handle LoRA weights exclusively, ensuring modularity. This functionality supports resuming training from checkpoints or directly applying LoRA weights for model inference.

5. Results

The fine-tuned Stable Diffusion model is loaded using the base model `runwayml/stable-diffusion-v1-5`, enhanced with LoRA weights for improved inference. The process involves:

Loading the base model and the LoRA weights using the `StableDiffusionPipeline` and the `load_lora_weights` method.

Generating images from textual prompts. For example, the prompt "8-bit illustration of a city view" generates three variations of the city view.

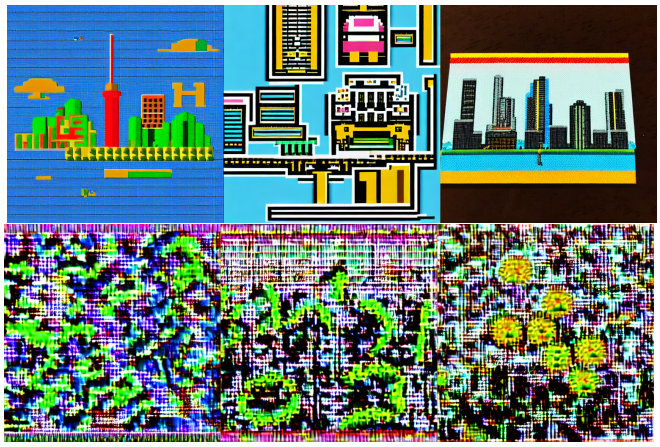

Figure 2: Inference results from the pretrained model (top) and the fine-tuned model (bottom). The fine-tuned model's outputs are less recognizable, while the pretrained model without fine-tuned LoRA weights produces more coherent and reasonable images.

6. Conclusion

This implementation of Stable Diffusion customization demonstrates the efficacy of using the LoRA-based fine-tuning approach for improving the model's performance on a specific task. By leveraging a pre-trained Stable Diffusion model and integrating custom dataset preparation, sub-model configuration, and LoRA fine-tuning, we successfully adapted the model to generate high-quality images based on user-defined prompts.

The dataset preparation process, which included image collection, caption generation, and integration with Hugging Face Datasets, was essential for ensuring that the model could learn from domain-specific data. The careful configuration of sub-models such as VAE, U-Net denoiser, and text encoder provided a solid foundation for model customization.

The LoRA technique, which allows for efficient fine-tuning by updating low-rank weight matrices within the U-Net's attention layers, proved to be a valuable tool for enhancing model performance without requiring full retraining. This approach, combined with rigorous training configurations, data augmentation, and optimizer settings, contributed to the model's ability to generate visually coherent and creative outputs from textual prompts.

Overall, this implementation highlights the flexibility and power of the Stable Diffusion model, as well as the potential of LoRA-based fine-tuning for customizing generative models to suit specific tasks. Future work could explore further optimization of the fine-tuning process, including experimenting with different dataset types, hyperparameters, and model architectures to achieve even more refined results.

[1] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

[2] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022. URL https://arxiv.org/abs/2112.10752.