Different techniques of fine-tuning Stable Diffusion

Teodoras Šaulys
Software Engineering, 4th year
Faculty of Mathematics and Informatics
Vilnius, Lithuania
teodoras.saulys@mif.stud.vu.lt

Povilas Slivinskas
Software Engineering, 3rd year
Faculty of Mathematics and Informatics
Vilnius, Lithuania
povilas.slivinskas@mif.stud.vu.lt

***Abstract*—In this paper we analyze the performance of Stable Diffusion, fine-tuned using different techiques. The primary focus of this team project is to thoroughly analyze the performance of Stable Diffusion and assess its ability to be fine-tuned effectively with the same dataset and parameters, particularly in generating images based on textual descriptions. By utilizing pre-trained model on the selected dataset, we compare the results and delve into the field of fine-tuning text-to-image models.**

***Index Terms*—fine-tuning, text-to-image, diffusion models, large language models, weight freezing, LoRA**

I. Introduction

Text-to-image synthesis has gained significant attention in recent years due to its potential applications in various domains such as computer vision, creative arts, and multimedia content generation. The ability to generate realistic images from textual descriptions opens up new possibilities for content creation, storytelling, and assisting individuals with limited artistic skills to visualize their ideas. In this paper, we delve into the realm of fine-tuning text-to-image models, focusing on Stable Diffusion, using pre-made dataset containing images of Pokemons with captions [4]. Our objective is to explore the effectiveness of fine-tuning Stable Diffusion with the same dataset and parameters, but with different fine-tuning techniques. We are aiming to enhance the model's capability to generate visually coherent and accurate images. Through a comprehensive analysis of the performance and results, we aim to provide insights into the potential of fine-tuning text-to-image models and contribute to the advancement of this exciting field.

II. Dataset

The dataset used in this paper [4] consists of 833 image-description pairs of Pokémons. The description for each Pokémon was generated with the Bootstrapping Language-Image Pre-training (BLIP) framework, which means that the descriptions may not be perfect and may have some inaccuracies. However, for our purposes this should not be a big hindrance.

III. Model used

For this paper we used Stable Diffusion, following the guidelines in [3] in order to get it running. The model is based on a type of a diffusion model, which is called Latent Diffusion, which is proposed and explored in [5]. Fig. 1 contains a diagram of the internals of Stable Diffusion. The

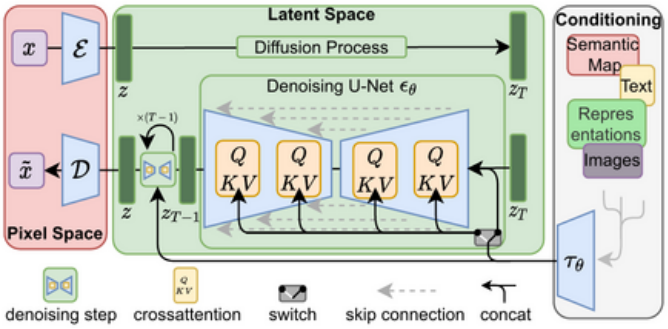

Figure 1 – Inner workings of Stable Diffusion

basic principle of such machine learning models is that they employ a step-by-step denoising of random Gaussian noise to create images. Stable Diffusion consists of three main parts:

- Variational Auto Encoder (VAE)
- U-Net
- Text encoder, such as before mentioned CLIP

The VAE takes care of encoding and decoding of images. In the beginning of image generation, the VAE encodes the image into a low dimensional latent representation, which is fed as input to the U-Net. After denoising the latent representation, the VAE decoder converts this representation back into a image, which becomes the final output. The U-Net also consists of an encoder and decoder, both of which are constructed with ResNet blocks. This part of the model downsamples the image representation for a few steps in order to minimise expensive computations. In figure 3 we can see how this down and upsampling occurs. After downsampling, the decoder upsamples the representation back to the original size with denoising applied. One problem which arises during this process is that we obviously lose information during downsampling. To solve this problem, the downsampling blocks are connected to the upsampling blocks with shortcut connections. Also, cross-attention layers are included in both encoder and decoder of the U-Net, which will become important later in our paper. In fig. 2 we can see the different steps in the noising and de-noising process. In addition, the process of de-noising is executed in a loop for a certain number of times, and the noise from the latent representation is removed gradually for each step.

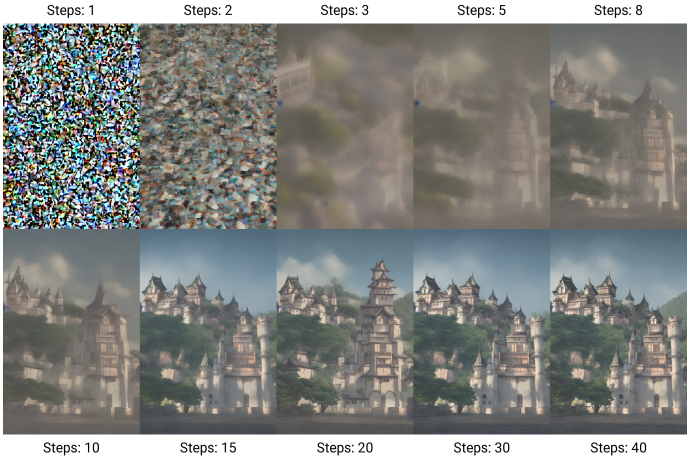

Figure 2 – Noising and de-noising process in Stable Diffusion

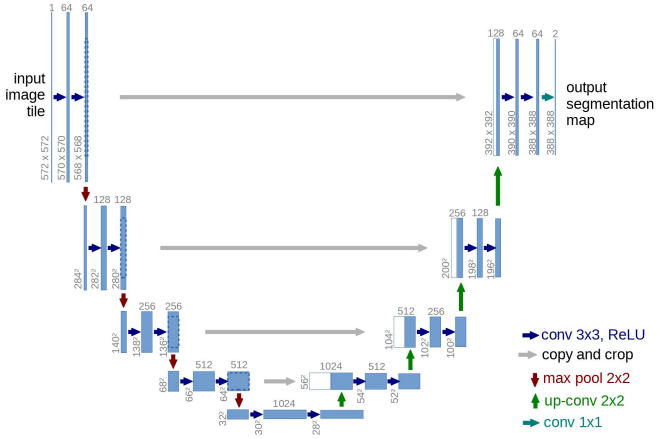

Figure 3 – The U-Net architecture

IV. Methods

A. Classic fine-tuning

Fine-tuning text-to-image models involves the process of adapting pre-trained models, originally trained on a large dataset, to generate visually coherent and accurate images based on textual descriptions. By fine-tuning these models with a specific dataset and parameters, we aim to enhance their ability to generate images that align with the given textual input. Fine-tuning leverages the pre-existing knowledge and learned features of the models, allowing them to specialize and generate more contextually relevant images in response to textual prompts. This process facilitates the transfer of knowledge from the pre-trained models to the target task, enabling improved performance and generating visually compelling images that effectively capture the essence of the provided textual descriptions.

B. Fine-tuning with Low-Rank Adaptation of Large Language Models (LoRA)

Low-Rank adaptation of large language models was proposed by researchers from Microsoft as a viable fine-tuning technique for large language models [2]. The main problem of fine-tuning such models with numerous parameters is that it is often expensive and time-consuming, because all of the parameters are involved during fine-tuning. Instead an alternative is proposed, which consists of freezing the pre-trained weights and injecting trainable rank decomposition matrices into transformer's layers. As a result, instead of training the model's weights, the weights of the injected layers are trained. Originally LoRA was intended for large language models, nevertheless it is applicable to diffusion models, in our specific case Stable Diffusion. In our instance, LoRA can be used in the cross-attention layers, which are involved in mapping image representations with the given prompts. The main advantages of such fine-tuning approach are as follows:

- Significantly faster training
- The resulting weights are much smaller
- Computing resources required for this approach are more modest

Above mentioned points greatly increase availability to fine-tuning Stable Diffusion to a wider population of users which may not have the bleeding-edge hardware. Following the guidelines in [1], we attempted to fine-tune Stable Diffusion in this fashion.

C. Fine-tuning parameters

For fine-tuning both the Stable Diffusion text-to-image model, we employed a consistent set of parameters both for regular and LoRA fine-tuning. The following list provides a brief explanation of each parameter:

- Dataset name: "lambdalabs/pokemon-blip-captions" - a dataset containing Pokemon-related image-caption pairs.
- Resolution: 512 - sets the resolution of the generated images to 512x512 pixels.
- Center_crop: true - Applies center cropping to the input images.
- Random_flip: true - Randomly flips the images horizontally.
- Gradient_accumulation_steps: 4 - accumulates gradients over multiple steps, effectively increasing the effective batch size and - allowing for more stable training.
- Max_train_steps: 100 - limits the maximum number of training steps to 100.
- Learning_rate: 1e-05 - sets the initial learning rate, governing the step size for adjusting the model's parameters during training.
- Number of examples: 833 - Number of images in our dataset.
- Number of epochs: 2 - Complete passes over our dataset
- Total train batch size: 16 - Examples that are processed in parallel during each training iteration
- Device: Cuda
- Mixed precision type: No - We are using single-precision (32-bit) floating-point numbers

The main hyperparameter differences in regular fine-tuning:

Figure 4 – Yoda generated after regular fine-tuning

Figure 5 – Yoda generated after regular fine-tuning

- LR_scheduler: Constant - sets the learning rate scheduler to "constant," indicating a fixed learning rate throughout the training process.
- Use_ema: true - Enables the use of Exponential Moving Average (EMA) during training, which can stabilize the model's performance.

The main hyperparameter differences in LoRA fine-tuning:

- LR_scheduler: Cosine - sets the learning rate scheduler to "cosine" - gradually decreases the learning rate in a cosine-like manner over the course of training.

By utilizing these parameters consistently for both models, we aim to ensure a fair comparison and evaluate the effectiveness of fine-tuning in generating images based on textual descriptions from the chosen Pokemon dataset.

V. Results

Evaluating the results of text-to-image generation is inherently subjective, as the perception of image quality and relevance varies among individuals. However, while judging the generated images is subjective, we can objectively compare the results based on the sizes of generated models, the time it takes to train them, and the time it takes to generate images. These objective metrics provide valuable insights into the computational requirements and efficiency of the models, allowing

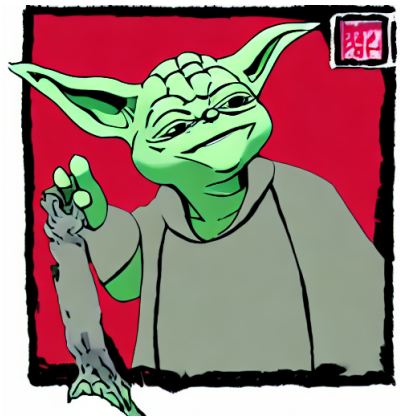

Figure 6 – Yoda generated after regular fine-tuning

Figure 7 – Yoda generated after LoRA fine-tuning

Figure 8 – Yoda generated after LoRA fine-tuning

Figure 9 – Yoda generated after LoRA fine-tuning

Figure 10 – Yoda generated after LoRA fine-tuning

for a comparative analysis of their performance. In the case of Stable Diffusion (SD), the sizes of the weights for the utilized architectures are as follows: UNet with 3.3G and VAE with 320M. These weights indicate the model's complexity and the amount of information it needs to process during training and generation. Additionally, the training process for Stable Diffusion takes approximately 15 minutes (using the parameters described above), while generating images with Stable Diffusion requires around 50 seconds.

In figures 4, 5 and 6 we can see that the after regular fine-tuning these samples have a noticeable comic book style due to the dataset's comic book style. While figures 7, 8 9 and 10, which were generated after LoRA fine-tuning have a more real life style, except for figures 9 and 10. Mainly we can observe 3 different category styles: full-realistic, semi-cartoon, full-cartoon. In case of regular fine-tuning the generated images had a distribution of 20, 30 and 50 percent respectively, while for LoRA the distribution is 20, 60 and 20 respectively. One of the reasons why this is the case can be the fact that during regular fine-tuning all weights are adjusted, while during LoRA the Stable Diffusion model itself is not trained, instead the injected layers are. Other reasons remain to be explored.

On the other hand, for the LoRA approach, the weights have a size of 3.2M, indicating a comparatively smaller model architecture. The training time for LoRA amounts to approximately 8 minutes, and the image generation time is reduced to around 20 seconds. These results provide insights into the computational requirements and efficiency of the models. Stable Diffusion, with its larger weight sizes and longer training time, offers a higher level of complexity and potential for generating detailed and contextually coherent images. However, it comes at the cost of increased training and generation time. Conversely, the LoRA approach, with its smaller weight size and shorter training and generation times, provides a more efficient solution while still delivering satisfactory image generation results. These results showcase the trade-offs between model complexity, training time, and generation speed, highlighting the importance of considering these factors when selecting a text-to-image model for specific applications. Ultimately, the choice of model depends on the desired balance between image quality, computational resources, and real-time generation requirements.

References

[1] Pedro Cuenca and Sayak Paul. "Using LoRA for Efficient Stable Diffusion Fine-Tuning". In: *Hugging Face Blog* (2023). https://huggingface.co/blog/lora.

[2] Edward J. Hu et al. *LoRA: Low-Rank Adaptation of Large Language Models*. 2021. arXiv: 2106.09685 [cs.CL].

[3] Suraj Patil et al. "Stable Diffusion with Diffusers". In: *Hugging Face Blog* (2022). https://huggingface.co/blog/stable_diffusion.

[4] Justin N. M. Pinkney. *Pokemon BLIP captions*. https://huggingface.co/datasets/lambdalabs/pokemon-blip-captions/. 2022.

[5] Robin Rombach et al. *High-Resolution Image Synthesis with Latent Diffusion Models*. 2022. arXiv: 2112.10752 [cs.CV].