Task 2: LoRA Adaptation for Stable Diffusion Using Docker

Table of Contents

LoRA adaptation for a Stable Diffusion model

Files and folders:

<table>
<tr><th colspan="2">Files</th><th>Description</th></tr>
<tr><td colspan="2">config.yaml</td><td>Configuration file for base_model and loras</td></tr>
<tr><td colspan="2">Dockerfile</td><td>To build and run docker image</td></tr>
<tr><td colspan="2">main.py</td><td>main pipeline</td></tr>
<tr><td>landscape.ipynb</td><td rowspan="2">Note: Code in these 2 notebooks are exactly the same (explore the same single and multiple LoRAs)

The only difference is the text prompt being used</td><td>Demo jupyter notebook for landscape prompt: “A beautiful sky”</td></tr>
<tr><td>single_object.ipynb</td><td>Demo jupyter notebook for single object prompt: “A green pokemon with blue eyes’</td></tr>
<tr><td colspan="2">Evaluation_img_generation.ipynb
(please refer to this notebook for interpretation of inception score)</td><td>Evaluation strategy (Inception Score) jupyter notebook
To evaluate quality, diversity and consistency of generated images based on mean and standard deviation of inception score</td></tr>
</table>

Consolidation of Findings/Insights from notebooks

from **landscape.ipynb** and **single_object.ipynb** which explores Single LoRa and multiple LoRA integration with Stable Diffusion model (copied from the respective notebooks to this document for easier reference)

Findings/Insights From landscape.ipynb

Findings from Single LoRA Image Generation Results

- In general, generated image results are better (more closely follow the LoRA) when the specific LoRA tag is included in the text prompt.
  - Hence, the use of LoRAs WITH the respective LoRA tags is explored
  - LoRA tags can be found on the respective LoRA model download page
- if text is present in the generated image, it is highly probable that the text does not make sense.

Findings from Multiple LoRAs Image Generation Results

Results from merging multiple LoRAs tend to be beautiful, aesthetically-pleasing for a

landscape text prompt.
This could be due to
- a wider variety of stylistic renditions can be applied on landscape subject matter (better than single object form)
- fortunate suitability of num_inference_steps and guidance scale
  - optimal num_inference_steps and guidance scale can be determined empirically (trial and error)

Findings/Insights From single_object.ipynb

Findings from Single LoRA Image Generation Results
- In general, generated image results are better (more closely follow the LoRA) when the specific LoRA tag is included in the text prompt.
    - Hence, the following section which applies Multiple LoRAs to the Stable Diffusion model will only explore the use of LoRAs WITH the respective LoRA tags.

- fuse_loRA method's lora_scale parameter value >0.5 yields generally better results than values <=0.5
- For LoRA types such as concept LoRA (jellyfish forest), which are not very suitable for the text prompt subject matter "a green pokemon with blue eyes" is an object, results are generally poor

Findings from Multiple LoRAs Image Generation Results
Results from merging multiple LoRAs tend to be unpredictable/bizarre for an object text prompt.
This could be due to
- uncomplementary LoRAs styles/concepts/objects/etc.
- suitability of num_inference_steps and guidance scale
  - optimal num_inference_steps and guidance scale can be determined empirically (trial and error)

Folders	Description
base_model - sd.py	Stable diffusion + lora integration model class
generated_images < subfolders named by text prompt> <files named by LoRA tag>	Images generated from running the main.py main pipeline
LoRAs - basepixel-20.safetensors - easter-fusion-v2.safetensors	Lora model weights downloaded

- jellyfish-forest.safetensors - MoXinStyle.safetensors - wanostyle_2_offset.safetensors	

Specific task(s) within game creation where this adapted model could be applied:
Given that Output of a StableDiffusion + LoRA model is: 2D image(s)
1. **Facilitate Asset Creation:** - **Convert Text into 2D images ⟵ this project would be useful for this step** - Convert 2D images into 3D mesh/asset/voxel art: for reward tokens (e.g. easter eggs but in particular design/ theme) [To be explored in future] Methods: - 2D image of single subject into 3D models with NeRF https://github.com/junshutang/Make-It-3D - 2D image to 3D models: https://github.com/harry7557558/img23d - 2D image to 3D mesh: https://github.com/divyanshu-talwar/ShakaLakaBoomBoom - 2D pixel art to 3D voxel art: https://github.com/Orama-Interactive/VoxeloramaExtension
2. Concept ideation/brainstorming for visual themes for story concepts/ideas

Stable Diffusion model and LoRA integration process

Overview: Main process of generating the image:

1. Select and load pretrained Stable Diffusion model
2. Download/Select LoRA model: Load LoRA weights (safetensors file) into stable diffusion model
3. Generate the image by inputting text prompt into pipeline → image(s) generated

```
from diffusers import AutoPipelineForText2Image
import torch

pipeline = AutoPipelineForText2Image.from_pretrained(
    "runwayml/stable-diffusion-v1-5", torch_dtype=torch.float16, use_safetensors=True
).to("cuda")
prompt = "Astronaut in a jungle, cold color palette, muted colors, detailed, 8k"

image = pipeline(prompt, num_inference_steps=25).images[0]
```

Extended detailed process of generating the image:

(includes method on how to fuse multiple LoRAs into the model and how to unload LoRA weights)

1. **Select and load pretrained Stable Diffusion model**

```
# Initialize the pipeline with Stable Diffusion model
model_id = "runwayml/stable-diffusion-v1-5"
pipeline = AutoPipelineForText2Image.from_pretrained(model_id,
torch_dtype=torch.float16).to("cuda")
```

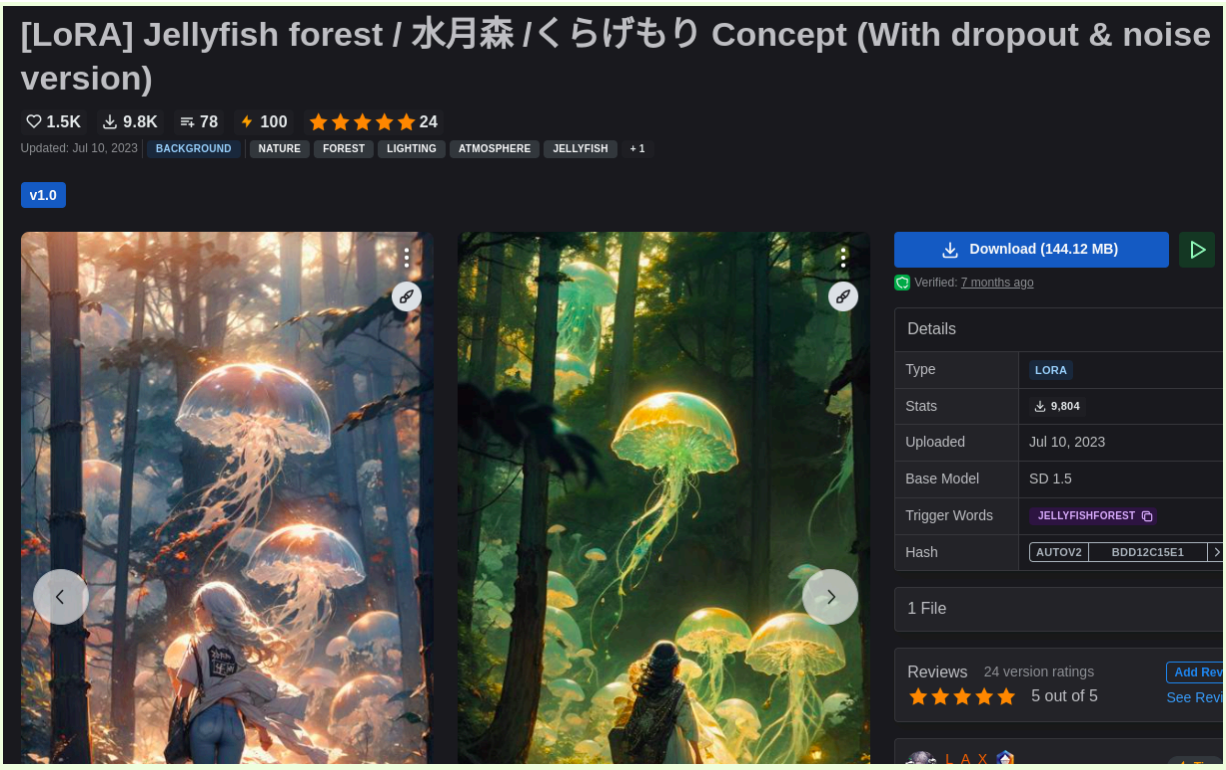
https://huggingface.co/docs/diffusers/en/api/pipelines/auto_pipeline

Based on the task, the `AutoPipeline` class automatically retrieves the relevant pipeline given the name or path to the pretrained weights with the `from_pretrained()` method.

`AutoPipeline` supports text-to-image, image-to-image, and inpainting for the following diffusion models:

- Stable Diffusion
- ControlNet
- Stable Diffusion XL (SDXL)
- DeepFloyd IF
- Kandinsky 2.1
- Kandinsky 2.2

2. **Download/Select LoRA model: Load LoRA weights (safetensors file) into stable diffusion model**
   https://huggingface.co/docs/diffusers/en/using-diffusers/loading_adapters#LoRA



Make sure **trigger word(s)** for the LoRA are included in the text prompt: easter; (highlighted in purple)

3. **input text prompt and other parameters into pipeline → image(s) generated**

```
# Define parameters for image generation
text_prompt = "a green pokemon with blue eyes"
seed = 3 # for reproducibility
num_imgs = 1
h, w = 512, 512
num_inference_steps = 15
guidance_scale = 7.5
```

4. **Optional: set the lora_scale parameter in lora_fuse to vary the impact of the LoRA to be applied.**
    a. default = 1 means full effect of LoRA weights is applied;
    b. range = [0,1];
    c. 0 means LoRA is not applied to the layers in stable diffusion model

```
pipeline.load_lora_weights(weight, weight_name=weight.split("/")[-1])
pipeline.fuse_lora(lora_scale=scale)
```

5. **Generate the image(s)**

 # to be implemented: If NSFW image(s) generated, regenerate new image(s)

```
image = pipeline(prompt + tag, generator=generator, num_inference_steps=num_inference_steps,
                 guidance_scale=guidance_scale, height=h, width=w,
                 num_images_per_prompt=num_imgs).images[0]
```

— currently, pipeline has 1 LoRA model's weights loaded —

To unload current LoRA weights and reload different LoRA weights:	**To fuse currently loaded LoRA weights with a different LoRA's weights**

`pipeline.unfuse_lora()`
`pipeline.unload_lora_weights()`
 	
`pipeline.load_lora_weights(weight_safetensor_filepath, weight_name=weight_name)`
 	
`pipeline.fuse_lora(lora_scale=scale)`
	`pipeline.load_lora_weights(weight_safetensor_filepath, weight_name=weight_name)`
	
	`pipeline.fuse_lora(lora_scale=scale)`

Pipeline still has 1 LoRA model's weights loaded and fused	Pipeline has 2 LoRA model's weights loaded and fused

LoRA Integration Results

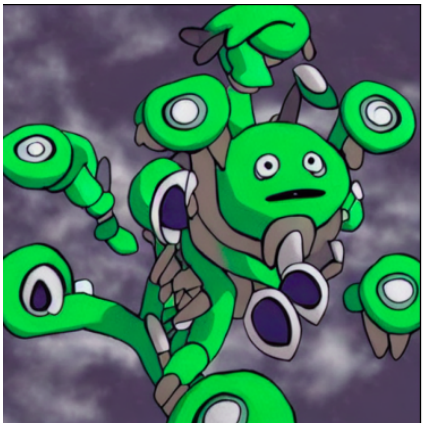
Note: recommended to have the specified LoRA Tag word(s) included in the text prompt for better results:

- Tag word(s) are typically stated on the LoRA model download page

text_prompt = "a green pokemon with blue eyes"

Stable Diffusion only:

Single LoRA integration results

Object LoRA: easter egg - Tag word(s): easter 	**Concept LoRA**: jellyfish forest - Tag word(s): jellyfish forest 
Style LoRA: Wanostyle (One Piece Style) - Tag word(s): wanostyle 	**Style LoRA**: Pixel art; - Tag word(s): basepixel 

Multiple LoRA integration results

Text prompt: “A beautiful sky”

Easter + basepixel	Easter + jellyfish forest

	
Wanostyle + basepixel 	

Parameter Adjustments: StableDifusionPipeline

https://huggingface.co/docs/diffusers/v0.3.0/en/api/pipelines/stable_diffusion AutoPipelineText2Image > automatically chooses StableDiffusionPipeline

diffusers.StableDiffusionPipeline :

The load_lora_weights() method loads LoRA weights into both the UNet and text encoder. It is the preferred way for loading LoRAs because it can handle cases where:

- the LoRA weights don't have separate identifiers for the UNet and text encoder
- the LoRA weights have separate identifiers for the UNet and text encoder

Pipeline output:

- **images** (List[PIL.Image.Image] or np.ndarray) — List of denoised PIL images of length batch_size or numpy array of shape (batch_size, height, width, num_channels). PIL images or numpy array present the denoised images of the diffusion pipeline.
- **nsfw_content_detected** (List[bool]) — List of flags denoting whether the corresponding generated image likely represents “not-safe-for-work” (nsfw) content.

StableDifusionPipeline

All Parameters:

Parameters

- **prompt** (`str` or `List[str]`) — The prompt or prompts to guide the image generation.
- **height** (`int`, *optional*, defaults to 512) — The height in pixels of the generated image.
- **width** (`int`, *optional*, defaults to 512) — The width in pixels of the generated image.
- **num_inference_steps** (`int`, *optional*, defaults to 50) — The number of denoising steps. More denoising steps usually lead to a higher quality image at the expense of slower inference.
- **guidance_scale** (`float`, *optional*, defaults to 7.5) — Guidance scale as defined in Classifier-Free Diffusion Guidance. `guidance_scale` is defined as `w` of equation 2. of Imagen Paper. Guidance scale is enabled by setting `guidance_scale > 1`. Higher guidance scale encourages to generate images that are closely linked to the text `prompt`, usually at the expense of lower image quality.
- **eta** (`float`, *optional*, defaults to 0.0) — Corresponds to parameter eta (η) in the DDIM paper: https://arxiv.org/abs/2010.02502. Only applies to schedulers.DDIMScheduler, will be ignored for others.
- **generator** (`torch.Generator`, *optional*) — A torch generator to make generation deterministic.
- **latents** (`torch.FloatTensor`, *optional*) — Pre-generated noisy latents, sampled from a Gaussian distribution, to be used as inputs for image generation. Can be used to tweak the same generation with different prompts. If not provided, a latents tensor will ge generated by sampling using the supplied random `generator`.
- **output_type** (`str`, *optional*, defaults to `"pil"`) — The output format of the generate image. Choose between PIL: `PIL.Image.Image` or `nd.array`.
- **return_dict** (`bool`, *optional*, defaults to `True`) — Whether or not to return a StableDiffusionPipelineOutput instead of a plain tuple.

Key Parameters

1. Input text prompt
2. num_inference_steps
3. guidance_scale
4. num_images_per_prompt (for scalability)

5. generator for reproducibility
   https://huggingface.co/docs/diffusers/v0.13.0/en/using-diffusers/reusing_seeds

Note on num_inference_steps and guidance scale:
In practice, finding the optimal balance between `num_inference_steps` and `guidance_scale` requires experimentation:

StableDiffusionPipeline has **nsfw_content_detected output**
→ can be used to minimize the risk of generating unwanted or harmful images when generating content that is intended for public or sensitive use.

Further details available in Appendix.

Run the Code:

Refer to **config.yaml** for the
- list of base_models (currently only 1): runwayml/stable-diffusion-v1-5
- List of LoRA model names: easter_egg, basepixel, jellyfish_forest, wanostyle, moxinstyle

Set up your virtual environment, then run 'pip install -r requirements.txt'
Then run the following code examples or modify them to suit your preferences. Note: there are no required arguments, all arguments have been set with default values to facilitate "python3 main.py"

Examples:

```
Python3 main.py --text-prompt "An underwater adventure"  --lora-name "jellyfish_forest" --fuse-lora-scale 0.9 --height 1024 --width 640 --num-inference-steps 17 --guidance-scale 8.4 --seed-num 13 --num-imgs 3
```
This example defines all possible available arguments (does not use any default arguments)

```
python3 main.py --text-prompt "a girl flying a kite" --lora-name "moxinstyle" --fuse-lora-scale 0.9 --height 1024 --width 640 --num-inference-steps 14 --guidance-scale 9.1 --seed-num 0 --num-imgs 3
```

```
python3 main.py --text-prompt "a girl flying a kite" --lora-name "moxinstyle" --fuse-lora-scale 0.9 --height 1024 --width 640 --num-inference-steps 17 --guidance-scale 6.7 --seed-num 13 --num-imgs 3
```

```
python3 main.py --text-prompt "a vast grassy meadow" --lora-name "jellyfish_forest"
```

--fuse-lora-scale 0.9 --height 1024 --width 640 --num-inference-steps 17 --guidance-scale 8.4 --seed-num 13 --num-imgs 3

Docker Containerization:

Create Dockerfile in project
Build docker container and image: # build the docker image: name of image is to be **stable-diffusion-lora** docker build -t **stable-diffusion-lora** .
To run docker container: (Note: ensure: 1. you have exited the virtual environment before running docker container, and 2. You are in the directory which contains the generated_images folder if it already exists) For GPU: docker run --gpus all -p 8080:8080 **stable-diffusion-lora**
To save docker image: docker save **stable-diffusion-lora**:latest > stable-diffusion-lora.tar
To load docker image: docker load < stable-diffusion-lora.tar
To use docker image to run the docker container/model (detailed): (Note: ensure: 3. you have exited the virtual environment before running docker container, and 4. You are in the directory which contains the generated_images folder if it already exists) #Run docker container with custom arguments: docker run --gpus all -v $(pwd)/generated_images:/app/generated_images stable-diffusion-lora:latest --text-prompt "An underwater adventure" --lora-name "jellyfish_forest" --fuse-lora-scale 0.9 --height 1024 --width 640 --num-inference-steps 17 --guidance-scale 8.4 --seed-num 13 --num-imgs 3 Assuming /app is the working directory specified in the Dockerfile This example defines all possible available arguments (does not use any default arguments)

Evaluation Strategy for the Adapted Model

Evaluating images generated by a pre-trained Stable Diffusion model loaded with LoRA weights involves several aspects of both **qualitative** and **quantitative** evaluations

Aim: to ensure that the adaptations through LoRA:

1. improve upon specific capabilities or qualities of the original model
2. without significantly compromising the overall performance.

1. Qualitative Evaluation

- Via **Human Evaluation:**
- Conduct surveys/studies where human evaluators rate the images on various criteria like aesthetic appeal, relevance to the prompt, and overall impression, in qualitative (not quantitative terms; discussed below).

Note: the scores by each individual evaluator should be normalized
→ to guard against absolute 'optimism'/'pessimism' about the generated art, and to ensure the evaluations are on the same relative scale.

a. **Visual Quality:**

- **Fidelity:** Assess the realism and clarity of the generated images. Check for artifacts, blurriness, or inconsistencies within the images.
- **Diversity:** Evaluate if the model can generate a diverse set of outputs given the same prompt or different variations of a prompt.
- **Alignment with Prompts:** Determine how well the images align with the given prompts.
  → includes checking the accuracy of depicted subjects, themes, and details specified in the prompts.

Note the challenge: realism, clarity, diversity, and alignment with image prompts are susceptible to individual human judgment/perception, which may be inherently inconsistent/unreliable.

b. **Creativity and Novelty:**

- Review the model's ability to generate unique and creative responses to prompts that are not straightforward, assessing its capability to combine concepts in novel ways.

Note the challenge: creativity is subjective and uniqueness is difficult to evaluate unless one has an extremely extensive understanding of all forms of visual art, which is highly unlikely.

2. Quantitative Evaluation

a. **Automated Metrics:**

- Inception Score (IS) (implemented in evaluation_img_generation.ipynb): Measures clarity and diversity of the generated images.
- **Fréchet Inception Distance (FID):** Compares the distribution of generated images to real images from a dataset, assessing both diversity and fidelity.
    - Requires access to real images from the LoRA training set

b. **Stability and Consistency:**

- Test the model's output consistency over repeated trials with the same prompts and slight variations thereof to evaluate stability and robustness.
    - I.e. select a few text prompts to experiment with → keep constant
    - Vary the other input parameters e.g. fuse_lora_scale, guidance_scale, num_inference_steps

3. Comparison with Baselines

- To assess the impact of the LoRA adaptation to the stable diffusion model: Compare the result of the generated images based on the same text prompt using:
    - baseline Stable Diffusion model and
    - The Stable Diffusion model integrated (loaded and fused) with the respective LoRA weights

4. User Studies and Feedback

- Collect and analyze feedback from end-users on the usability, satisfaction, and perceived quality of the generated images on a quantitative scale. This can provide insights that automated metrics cannot capture.

5. Ethical and Bias Evaluation

- Analyze the generated images for potential biases or ethical concerns, especially if the model is being adapted to new domains or datasets.
- Also: to note how often nsfw images are generated for the particular domain/parameter settings; if images are to be regenerated for any nsfw images, would have to account for efficiency/ average increase in computational time required to generate the specified num_imgs number of images

6. Performance and Scalability

LoRA aims to provide efficient adaptation. Hence, assess/evaluate if the computational efficiency and scalability of the model post-adaptation are satisfactory without sacrificing performance unduly.

Evaluating a model comprehensively requires a combination of these strategies to get a holistic understanding of its performance and areas of improvement. It's also important to iterate on the evaluation process itself, as new findings and feedback may necessitate adjustments to the evaluation criteria or methodologies.

Appendix/Miscellaneous:

Part 2: LoRA Adaptation for Stable Diffusion Using Docker

About LoRAs:

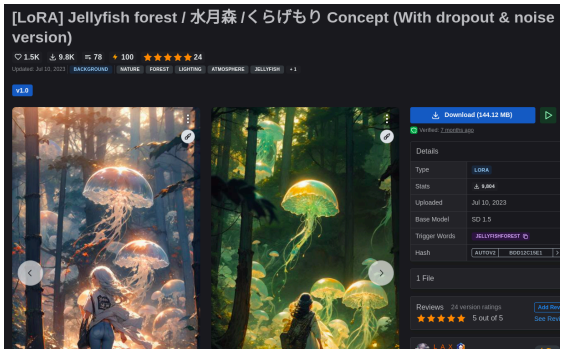
https://nightcafe.studio/blogs/info/stable-diffusion-lora-guide
https://aitoolmall.com/news/what-is-lora-for-stable-diffusion-and-how-to-use-it/
https://softwarekeep.com/help-center/how-to-use-stable-diffusion-lora-models types of LoRAs
Character, **style, concept,** clothing, **object** LoRAs
Difference between style and concept LoRA:

- Concept: idea abstraction (e.g. dreamlike quality)
- Style: particular art style

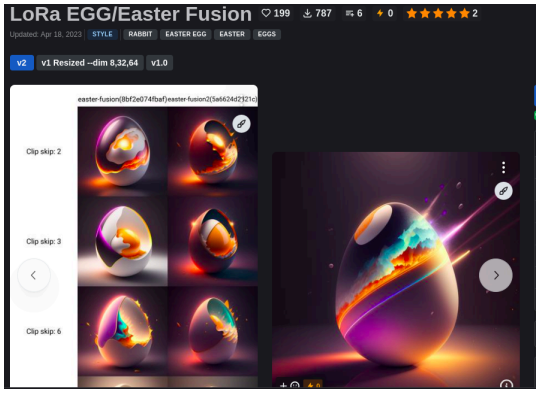
https://aituts.com/stable-diffusion-lora/ how-to-guide to use LoRAs
https://huggingface.co/stabilityai/stable-diffusion-xl-base-0.9/tree/main stable-diffusion model
https://huggingface.co/docs/diffusers/en/using-diffusers/loading_adapters#LoRA integration of stable diffusion with LoRAs; downloaded loras from CIVIT.AI

https://github.com/huggingface/diffusers/issues/5489 sample code for merging AI (stablediffusion-XL); huggingface loras
https://huggingface.co/blog/lora using huggingface loras

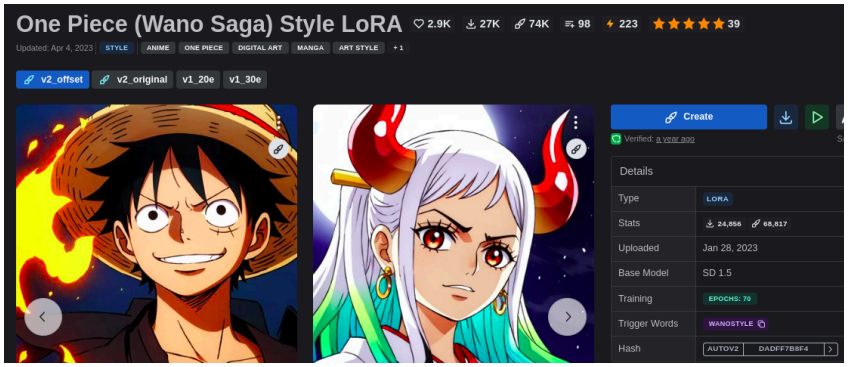
Jellyfish forest lora: https://civitai.com/models/106312?modelVersionId=114163



Easter egg lora: https://civitai.com/models/41893/lora-eggeaster-fusion



https://civitai.com/models/4219/one-piece-wano-saga-style-lora



https://huggingface.co/blog/sdxl_lora_advanced_script#inference
https://github.com/RAHUL-KAD/2d_to_3d_image_generation.git # NeRF; needs images of 360 view of the subject)

Training custom LoRA:

Stable Diffusion Automatic1111 web UI

https://www.appypie.com/blog/how-to-train-lora-with-stable-diffusion-dreambooth
Stable Diffusion Automatic1111 and LoRA https://stable-diffusion-art.com/lora/

Training custom LoRA with Dreambooth

https://www.appypie.com/blog/how-to-train-lora-with-stable-diffusion-dreambooth

Note on num_inference_steps and guidance scale:

num_inference_steps and guidance scale controls the quality, coherence, and alignment of the generated images with the input prompt.

num_inference_steps: number of steps the model takes to refine the generated image. Each step involves the model making adjustments to bring the image closer to what it interprets from the input prompt.

- Range = [1,100]
- Diminishing returns beyond a certain point (ugly images/too ugly images that become classified as NSFW)
- General impact: Higher num_inference_steps
  → more detailed and coherent images because the model has more opportunities to refine its output.
  → increases the computation time linearly.

guidance_scale: controls how closely the generated image should follow the input prompt. higher `guidance_scale` encourages model to prioritize fidelity to the prompt over the diversity of the output.

- Range = [1.0, 30.0]
- General Impact: Higher `guidance_scale` values can produce images that are
  - more closely aligned with the prompt's details
  - but can also lead to overfitting to the prompt details, resulting in less realistic or sometimes bizarre outputs.
  - Less diverse/creative results

Why High Values Can Produce Bad or NSFW Results

- **Overfitting to Prompt:** A very high `guidance_scale` can cause the model to overfit to the details of the prompt, including any ambiguities or suggestive content, which might lead to generating NSFW (Not Safe For Work) content even if not explicitly intended.
- **Loss of Diversity:** High `guidance_scale` can also reduce the diversity of the generated images, making them less creative or too literal, which might not always be desirable.
- **Computation Constraints:** Using a high number of `num_inference_steps` increases computation time and might lead to diminishing returns in quality after a certain point. It can also exacerbate any tendencies towards generating unwanted content since the model has more iterations to move in that direction.
- **Sensitivity to Prompt:** The combination of high `num_inference_steps` and `guidance_scale` makes the model more sensitive to the input prompt. If the prompt has any ambiguity or suggestive content, the model is more likely to produce exaggerated or explicit versions of that content.

Alternative method to do inference with Stable Diffusion XL models and LoRAs
https://huggingface.co/docs/diffusers/main/en/tutorials/using_peft_for_inference

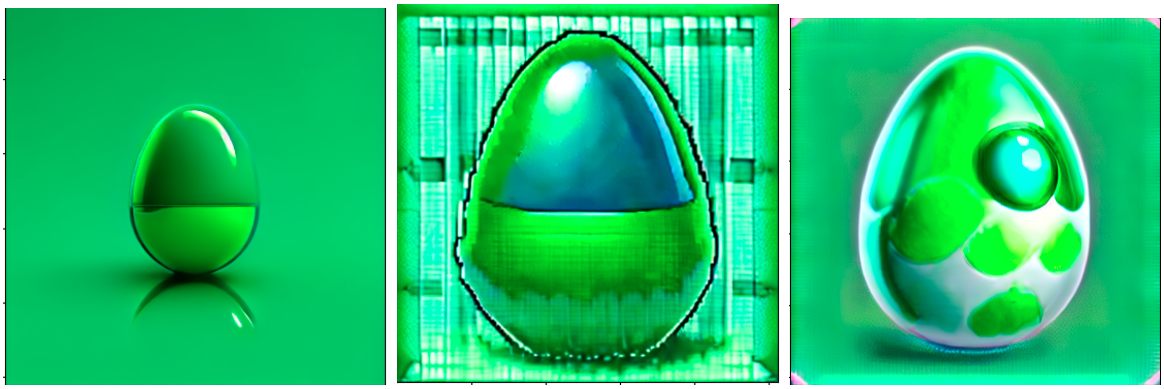
- Run into memory issues when i try to load stable diffusion xl models locally via:

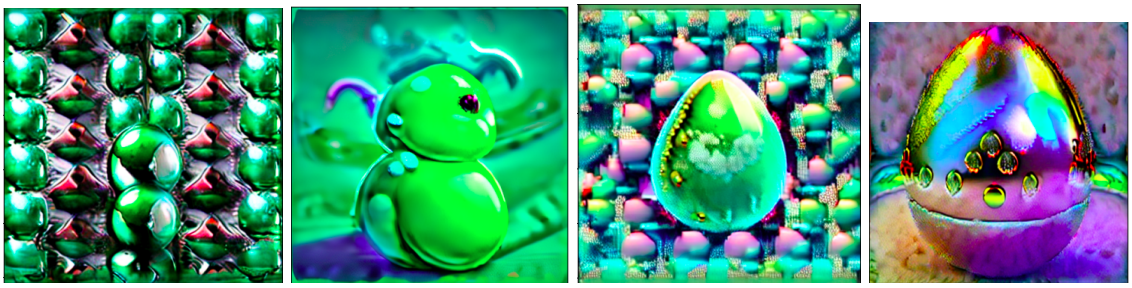
```
from diffusers import DiffusionPipeline
import torch

pipe_id = "stabilityai/stable-diffusion-xl-base-1.0"
pipe = DiffusionPipeline.from_pretrained(pipe_id, torch_dtype=torch.float16).to("cuda")
```

Text-to-3D solution
https://www.alpha3d.io/ ; code not available yet

Miscellaneous Multiple LoRA integration results

Easter Egg LoRA + JellyFish Forest LoRA





Easter + pixel LoRAs