**Load LoRAs for inference**

There are many adapters (with LoRAs being the most common type) trained in different styles to achieve different effects. You can even combine multiple adapters to create new and unique images. With the 🤗 [PEFT](https://huggingface.co/docs/peft/index) integration in 🤗 Diffusers, it is really easy to load and manage adapters for inference. In this guide, you’ll learn how to use different adapters with [Stable Diffusion XL (SDXL)](https://huggingface.co/docs/diffusers/api/pipelines/stable_diffusion/stable_diffusion_xl) for inference.

Throughout this guide, you’ll use LoRA as the main adapter technique, so we’ll use the terms LoRA and adapter interchangeably. You should have some familiarity with LoRA, and if you don’t, we welcome you to check out the [LoRA guide](https://huggingface.co/docs/peft/conceptual_guides/lora).

Let’s first install all the required libraries.

Copied

!pip install -q transformers accelerate

!pip install peft

!pip install diffusers

Now, let’s load a pipeline with a SDXL checkpoint:

Copied

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")

Next, load a LoRA checkpoint with the [load\_lora\_weights()](https://huggingface.co/docs/diffusers/v0.25.0/en/api/loaders/lora" \l "diffusers.loaders.StableDiffusionXLLoraLoaderMixin.load_lora_weights) method.

With the 🤗 PEFT integration, you can assign a specific adapter\_name to the checkpoint, which let’s you easily switch between different LoRA checkpoints. Let’s call this adapter "toy".

Copied

pipe.load\_lora\_weights("CiroN2022/toy-face", weight\_name="toy\_face\_sdxl.safetensors", adapter\_name="toy")

And then perform inference:

Copied

prompt = "toy\_face of a hacker with a hoodie"

lora\_scale= 0.9

image = pipe(

prompt, num\_inference\_steps=30, cross\_attention\_kwargs={"scale": lora\_scale}, generator=torch.manual\_seed(0)

).images[0]

image

With the adapter\_name parameter, it is really easy to use another adapter for inference! Load the [nerijs/pixel-art-xl](https://huggingface.co/nerijs/pixel-art-xl) adapter that has been fine-tuned to generate pixel art images, and let’s call it "pixel".

The pipeline automatically sets the first loaded adapter ("toy") as the active adapter. But you can activate the "pixel" adapter with the [set\_adapters()](https://huggingface.co/docs/diffusers/v0.25.0/en/api/loaders/unet" \l "diffusers.loaders.UNet2DConditionLoadersMixin.set_adapters) method as shown below:

Copied

pipe.load\_lora\_weights("nerijs/pixel-art-xl", weight\_name="pixel-art-xl.safetensors", adapter\_name="pixel")

pipe.set\_adapters("pixel")

Let’s now generate an image with the second adapter and check the result:

Copied

prompt = "a hacker with a hoodie, pixel art"

image = pipe(

prompt, num\_inference\_steps=30, cross\_attention\_kwargs={"scale": lora\_scale}, generator=torch.manual\_seed(0)

).images[0]

image

**Combine multiple adapters**

You can also perform multi-adapter inference where you combine different adapter checkpoints for inference.

Once again, use the [set\_adapters()](https://huggingface.co/docs/diffusers/v0.25.0/en/api/loaders/unet" \l "diffusers.loaders.UNet2DConditionLoadersMixin.set_adapters) method to activate two LoRA checkpoints and specify the weight for how the checkpoints should be combined.

Copied

pipe.set\_adapters(["pixel", "toy"], adapter\_weights=[0.5, 1.0])

Now that we have set these two adapters, let’s generate an image from the combined adapters!

LoRA checkpoints in the diffusion community are almost always obtained with [DreamBooth](https://huggingface.co/docs/diffusers/main/en/training/dreambooth). DreamBooth training often relies on “trigger” words in the input text prompts in order for the generation results to look as expected. When you combine multiple LoRA checkpoints, it’s important to ensure the trigger words for the corresponding LoRA checkpoints are present in the input text prompts.

The trigger words for [CiroN2022/toy-face](https://hf.co/CiroN2022/toy-face) and [nerijs/pixel-art-xl](https://hf.co/nerijs/pixel-art-xl) are found in their repositories.

Copied

*# Notice how the prompt is constructed.*

prompt = "toy\_face of a hacker with a hoodie, pixel art"

image = pipe(

prompt, num\_inference\_steps=30, cross\_attention\_kwargs={"scale": 1.0}, generator=torch.manual\_seed(0)

).images[0]

image

Impressive! As you can see, the model was able to generate an image that mixes the characteristics of both adapters.

If you want to go back to using only one adapter, use the [set\_adapters()](https://huggingface.co/docs/diffusers/v0.25.0/en/api/loaders/unet" \l "diffusers.loaders.UNet2DConditionLoadersMixin.set_adapters) method to activate the "toy" adapter:

Copied

*# First, set the adapter.*

pipe.set\_adapters("toy")

*# Then, run inference.*

prompt = "toy\_face of a hacker with a hoodie"

lora\_scale= 0.9

image = pipe(

prompt, num\_inference\_steps=30, cross\_attention\_kwargs={"scale": lora\_scale}, generator=torch.manual\_seed(0)

).images[0]

image

If you want to switch to only the base model, disable all LoRAs with the [disable\_lora()](https://huggingface.co/docs/diffusers/v0.25.0/en/api/loaders/unet" \l "diffusers.loaders.UNet2DConditionLoadersMixin.disable_lora) method.

Copied

pipe.disable\_lora()

prompt = "toy\_face of a hacker with a hoodie"

lora\_scale= 0.9

image = pipe(prompt, num\_inference\_steps=30, generator=torch.manual\_seed(0)).images[0]

image

**Monitoring active adapters**

You have attached multiple adapters in this tutorial, and if you’re feeling a bit lost on what adapters have been attached to the pipeline’s components, you can easily check the list of active adapters using the [get\_active\_adapters()](https://huggingface.co/docs/diffusers/v0.25.0/en/api/loaders/lora" \l "diffusers.loaders.LoraLoaderMixin.get_active_adapters) method:

Copied

active\_adapters = pipe.get\_active\_adapters()

active\_adapters

["toy", "pixel"]

You can also get the active adapters of each pipeline component with [get\_list\_adapters()](https://huggingface.co/docs/diffusers/v0.25.0/en/api/loaders/lora" \l "diffusers.loaders.LoraLoaderMixin.get_list_adapters):

Copied

list\_adapters\_component\_wise = pipe.get\_list\_adapters()

list\_adapters\_component\_wise

{"text\_encoder": ["toy", "pixel"], "unet": ["toy", "pixel"], "text\_encoder\_2": ["toy", "pixel"]}

**Fusing adapters into the model**

You can use PEFT to easily fuse/unfuse multiple adapters directly into the model weights (both UNet and text encoder) using the [fuse\_lora()](https://huggingface.co/docs/diffusers/v0.25.0/en/api/loaders/lora" \l "diffusers.loaders.LoraLoaderMixin.fuse_lora) method, which can lead to a speed-up in inference and lower VRAM usage.

Copied

pipe.load\_lora\_weights("nerijs/pixel-art-xl", weight\_name="pixel-art-xl.safetensors", adapter\_name="pixel")

pipe.load\_lora\_weights("CiroN2022/toy-face", weight\_name="toy\_face\_sdxl.safetensors", adapter\_name="toy")

pipe.set\_adapters(["pixel", "toy"], adapter\_weights=[0.5, 1.0])

*# Fuses the LoRAs into the Unet*

pipe.fuse\_lora()

prompt = "toy\_face of a hacker with a hoodie, pixel art"

image = pipe(prompt, num\_inference\_steps=30, generator=torch.manual\_seed(0)).images[0]

*# Gets the Unet back to the original state*

pipe.unfuse\_lora()

You can also fuse some adapters using adapter\_names for faster generation:

Copied

pipe.load\_lora\_weights("nerijs/pixel-art-xl", weight\_name="pixel-art-xl.safetensors", adapter\_name="pixel")

pipe.load\_lora\_weights("CiroN2022/toy-face", weight\_name="toy\_face\_sdxl.safetensors", adapter\_name="toy")

pipe.set\_adapters(["pixel"], adapter\_weights=[0.5, 1.0])

*# Fuses the LoRAs into the Unet*

pipe.fuse\_lora(adapter\_names=["pixel"])

prompt = "a hacker with a hoodie, pixel art"

image = pipe(prompt, num\_inference\_steps=30, generator=torch.manual\_seed(0)).images[0]

*# Gets the Unet back to the original state*

pipe.unfuse\_lora()

*# Fuse all adapters*

pipe.fuse\_lora(adapter\_names=["pixel", "toy"])

prompt = "toy\_face of a hacker with a hoodie, pixel art"

image = pipe(prompt, num\_inference\_steps=30, generator=torch.manual\_seed(0)).images[0]

[←Train a diffusion model](https://huggingface.co/docs/diffusers/tutorials/basic_training)[Accelerate inference of text-to-image diffusion m](https://huggingface.co/docs/diffusers/tutorials/fast_diffusion)