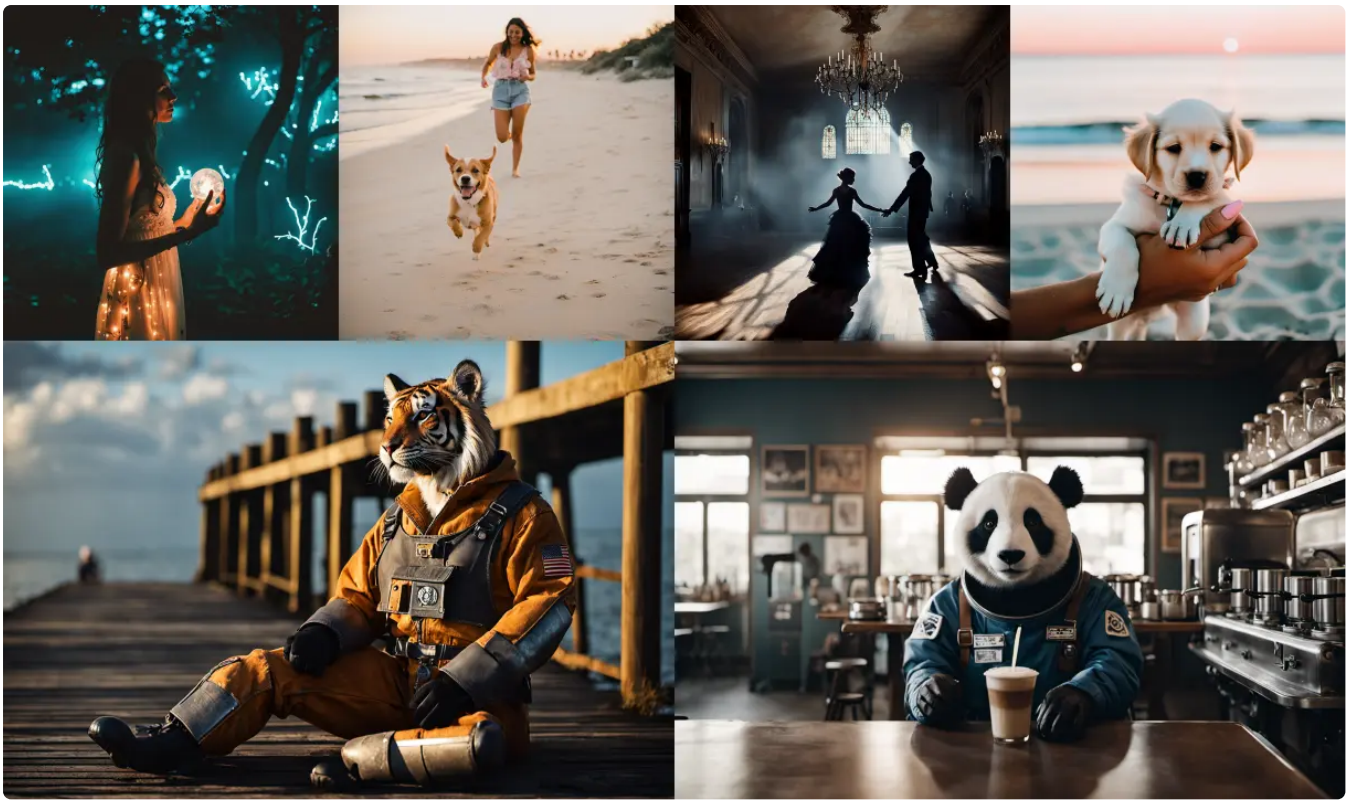
I Made Stable Diffusion XL Smarter by Finetuning it on Bad AI-Generated Images

August 21, 2023 · 11 min

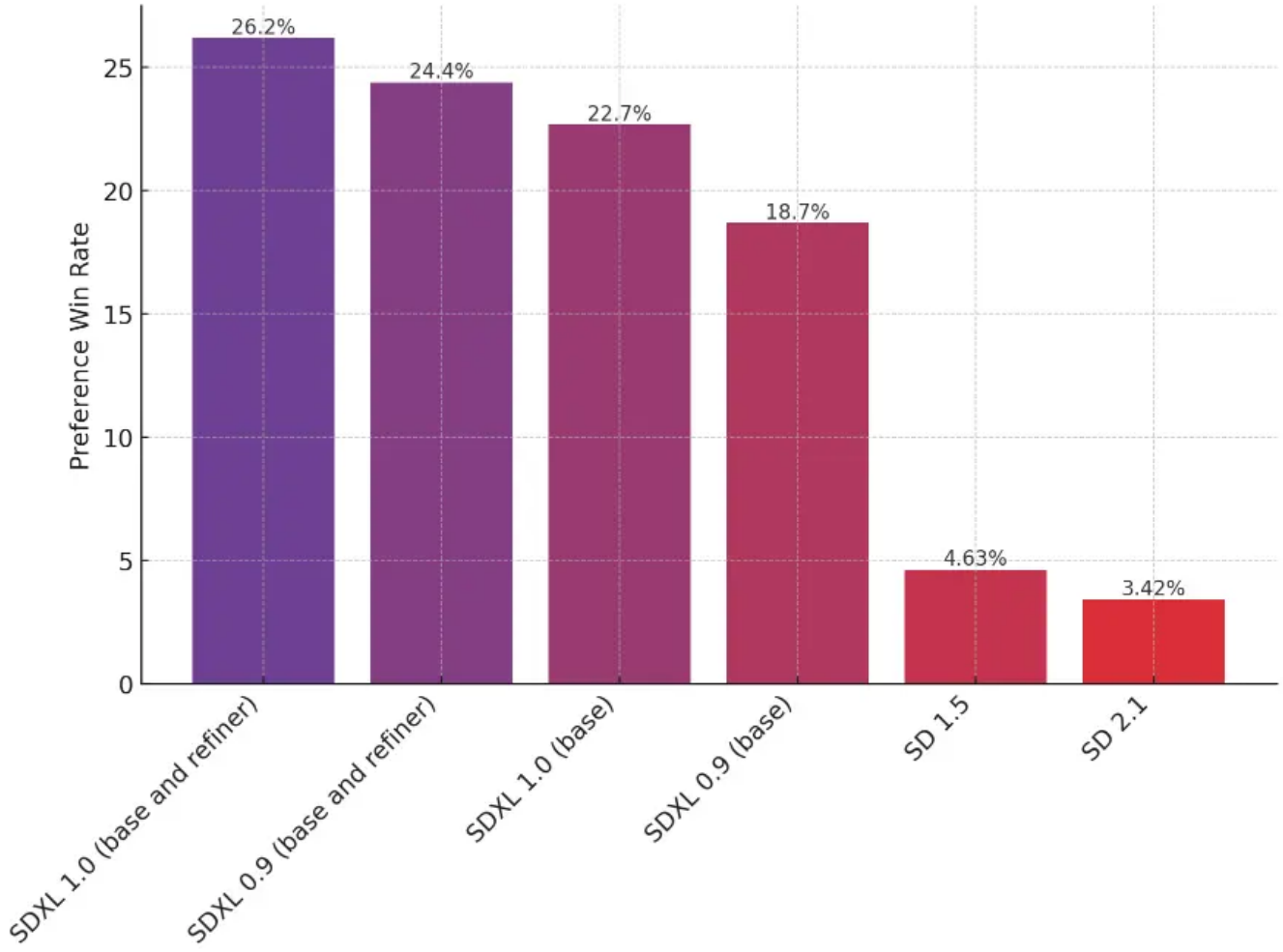
Inside you there are two AI-generated wolves.

Last month, Stability AI released Stable Diffusion XL 1.0 (SDXL) and open-sourced it without requiring any special permissions to access it.



Example SDXL 1.0 outputs. via Stability AI

The release went mostly under-the-radar because the generative image AI buzz has cooled down a bit. Everyone in the AI space is too busy with text-generating AI like ChatGPT (including myself!). Notably, it's one of the first open source models which can natively generate images at a 1024x1024 resolution without shenanigans, allowing for much more detail. SDXL is actually two models: a base model and an optional refiner model which siginficantly improves detail, and since the refiner has no speed overhead I strongly recommend using it if possible.



Comparisons of the relative quality of Stable Diffusion models. Note the significant increase from using the refiner. via Stability AI

The lack of hype doesn't mean SDXL is boring. Now that the model has full support in the diffusers Python library by Hugging Face with appropriate performance optimizations, we can now hack with it since the SDXL demos within diffusers are simple and easy to tweak:

```
import torch
from diffusers import DiffusionPipeline, AutoencoderKL

# load base SDXL and refiner
vae = AutoencoderKL.from_pretrained("madebyollin/sdxl-vae-fp16-fix",
                                    torch_dtype=torch.float16)
base = DiffusionPipeline.from_pretrained(
    "stabilityai/stable-diffusion-xl-base-1.0",
    vae=vae,
    torch_dtype=torch.float16,
    variant="fp16",
    use_safetensors=True,
)
_ = base.to("cuda")
```

```
refiner = DiffusionPipeline.from_pretrained(
    "stabilityai/stable-diffusion-xl-refiner-1.0",
    text_encoder_2=base.text_encoder_2,
    vae=base.vae,
    torch_dtype=torch.float16,
    variant="fp16",
    use_safetensors=True,
)
_ = refiner.to("cuda")


# generation using both models (mixture-of-experts)
high_noise_frac = 0.8
prompt = "an astronaut riding a horse"
negative_prompt = "blurry, bad hands"

image = base(
    prompt=prompt,
    negative_prompt=negative_prompt,
    denoising_end=high_noise_frac,
    output_type="latent",
).images

image = refiner(
    prompt=prompt,
    negative_prompt=negative_prompt,
    denoising_start=high_noise_frac,
    image=image,
).images[0]
```

I booted up a cloud virtual machine with a new midrange L4 GPU ($0.24/hr total with a Spot instance on Google Cloud Platform) and went to work. With a L4 GPU, each 1024x1024 image takes about 22 seconds to generate and you can only generate one image at a time on midrange GPUs unlike previous Stable Diffusion models since it uses 100% of the GPU's power, so some more patience is necessary. You *can* generate at a smaller resolution faster but it is strongly not recommended because the results are much, much worse.

diffusers also implemented support for two new features I haven't experimented with in my previous Stable Diffusion posts: prompt weighting and Dreambooth LoRA training and inference. Prompt weighting support with diffusers leverages the Python library compel to allow weighting of terms more mathematically. You can add any number of `+` or `-` to a given word to increase or decrease its "importance" in the resulting positional text embeddings, and therefore the final generation. You can also wrap phrases: for example, if you are generating `San`

`Francisco landscape by Salvador Dali, oil on canvas` and it does a photorealistic San Francisco instead, you can wrap the artistic medium such as `San Francisco landscape by Salvador Dali, (oil on canvas)+++` to get Stable Diffusion to behave as expected. In my testing, it fixes most of the prompt difficulty introduced in Stable Diffusion 2.0 onward, especially with a higher classifier-free guidance value (by default, `guidance_scale` is 7.5; I like to use 13)

> *All generated examples from the LoRA models in this blog post use a `guidance_scale` of 13.*

LoRA the Explorer

But what's most important is Dreambooth LoRA support, which is what makes bespoke Stable Diffusion models possible. Dreambooth is a technique to finetune Stable Diffusion on a very small set of source images and a trigger keyword to allow the use a "concept" from those images in other contexts given the keyword.

Demo image of how Dreambooth works. via Google

Training Stable Diffusion itself, even the smaller models, requires many expensive GPUs training for hours. That's where LoRAs come in: instead, a small adapter to the visual model is trained, which can be done on a single cheap GPU in 10 minutes, and the quality of the final model + LoRA is comparable to a full finetune (colloquially, when people refer to finetuning Stable Diffusion, it usually means creating a LoRA). Trained LoRAs are a discrete small binary file, making them easy to share with others or on repositories such as Civitai. A minor weakness with LoRAs is that you can only have one active at a time: it's possible to merge multiple LoRAs to get the benefits of all of them but it's a delicate science.

Before Stable Diffusion LoRAs became more widespread, there was textual inversion, which allows the text encoder to learn a concept, but it takes hours to train and the results can be unwieldy. In a previous post, I trained a textual inversion on the memetic Ugly Sonic, as he was not in Stable Diffusion's source dataset and therefore he would be unique. The generation results were mixed.

Ugly Sonic, but not the good kind of ugly.

I figured training a LoRA on Ugly Sonic would be a good test case for SDXL's potential. Fortunately, Hugging Face provides a train_dreambooth_lora_sdxl.py script for training a LoRA using the SDXL base model which works out of the box although I tweaked the parameters a bit. The generated Ugly Sonic images from the

trained LoRA are much better and more coherent over a variety of prompts, to put it mildly.

Ugly Sonic, but with **teeth**.

WRONG!

With that success, I decided to redo another textual inversion experiment by instead training a LoRA on heavily distorted, garbage images conditioned on `wrong` as a prompt in the hopes that the LoRA could then use `wrong` as a "negative prompt" and steer away from such images to generate less-distorted images. I wrote a Jupyter Notebook to create synthetic "wrong" images using SDXL

itself, this time using a variety of prompt weightings to get more distinct examples of types of bad images, such as `blurry` and `bad hands`. Ironically, we need to use SDXL to create high resolution low quality images.

Examples of the synthetic `wrong` images, which unintentionally resemble 2000's-era punk rock album covers.

More examples of the synthetic `wrong` images, which focus on the uncanny valley aspect of modern AI-generated images in which they look normal at a glance but looking closer reveals incremental horror. This is also why it's important to generate examples at the full 1024x1024 resolution.

I trained and loaded the LoRA into Stable Diffusion XL base model (the refiner does not need a LoRA) and wrote a comparison Jupyter Notebook to compare the results with a given prompt from:

- The base + refiner pipeline with no LoRA. (our baseline)
- The pipeline with no LoRA using `wrong` as the negative prompt (to ensure that there isn't a placebo effect)
- The pipeline **with the LoRA** using `wrong` as the negative prompt (our target result)

Each generation has the same seed, so photo composition should be similar across all three generations and the impact of both the `wrong` negative prompt and the LoRA vs. the base should be very evident.

Let's start with a simple prompt from the SDXL 0.9 demos:

`A wolf in Yosemite National Park, chilly nature documentary film photography`

The `wrong` prompt on the base model adds some foliage and depth to the forest image, but the LoRA adds a lot more: more robust lighting and shadows, more detailed foliage, and changes the perspective of the wolf to look at the camera which is more interesting.

We can get a different perspective of the wolf with similar photo composition by adding "extreme closeup" to the prompt and reusing the same seed.

`An extreme close-up of a wolf in Yosemite National Park, chilly nature documentary film photography`

In this case, the LoRA has far better texture, vibrance, and sharpness than the others. But it's notable that just adding a `wrong` prompt changes the perspective.

Another good test case is food photography, especially weird food photography like I generated with DALL-E 2. Can SDXL + the `wrong` LoRA handle non-Euclidian hamburgers with some prompt weighting to ensure they're weird?

```
a large delicious hamburger (in the shape of five-dimensional alien geometry)++++, professional food photography
```

The answer is that it can't, even after multiple prompt engineering attempts. However, this result is still interesting: the base SDXL appears to have taken the "alien" part of the prompt more literally than expected (and gave it a cute bun hat!) but the LoRA better understands the spirit of the prompt by creating an "alien" burger that humans would have difficulty eating, plus shinier presentation aesthetics.

A notable improvement with Stable Diffusion 2.0 was text legibility. Can SDXL and the `wrong` LoRA make text even more readable, such as text-dense newspaper covers?


```
lossless PDF scan of the front page of the January 2038 issue of the Wall Street Journal featuring a cover story about (evil robot world domination)++
```

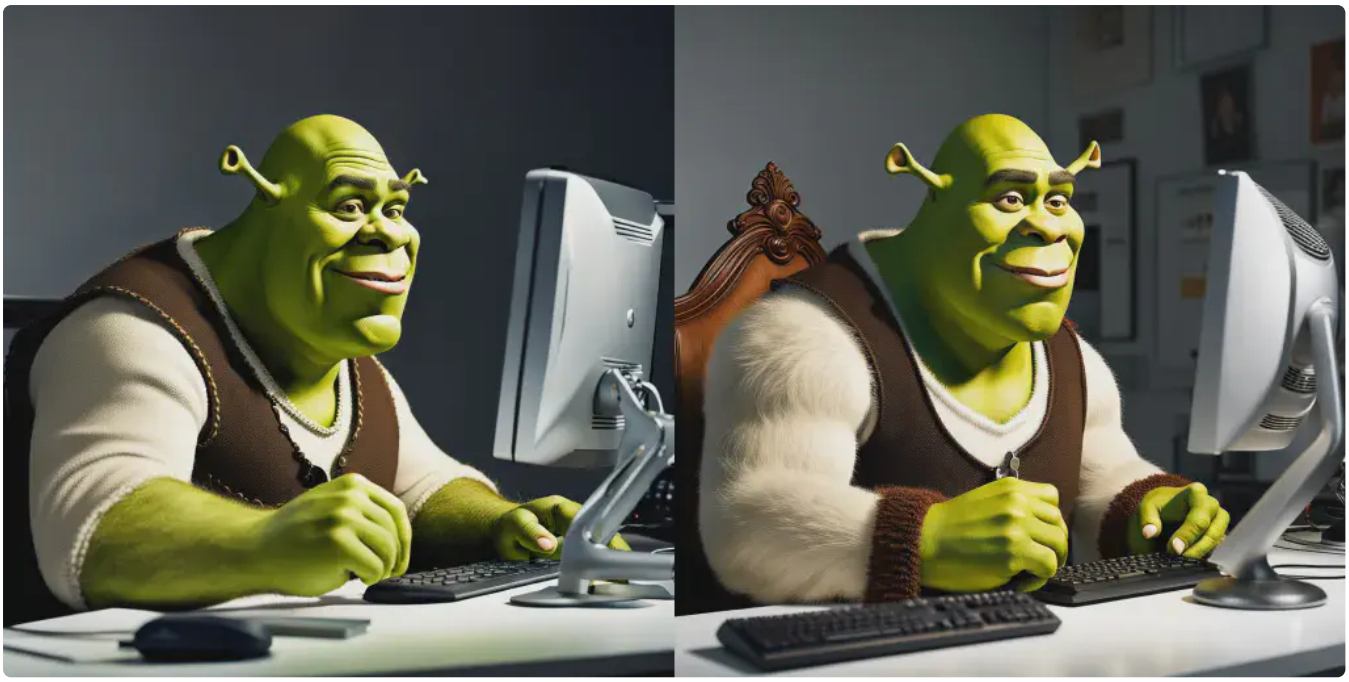
Text legibility is definitely improved since Stable Diffusion 2.0 but appears to be the same in all cases. What's notable with the LoRA is that it has improved cover typesetting: the page layout is more "modern" with a variety of article layouts, and headlines have proper relative font weighting. Meanwhile, the base model even with the `wrong` negative prompt has a boring layout and is on aged brown paper for some reason.

What about people? Does the `wrong` LoRA resolve AI's infamous issue with hands especially since we included many examples of such in the LoRA training data? Let's revamp a presidential Taylor Swift prompt from my first attempt with Stable Diffusion 2.0:

```
USA President Taylor Swift (signing papers)++++, photo taken by the Associated Press
```

Look at Taylor's right arm: in the default SDXL, it's extremely unrealistic and actually made *worse* when adding `wrong`, but in the LoRA it's fixed! Color grading with the LoRA is much better, with her jacket being more distinctly white instead of a yellowish white. Don't look closely at her hands in any of them though: creating people with SDXL 1.0 is still tricky and unreliable!

It's now clear that `wrong` + LoRA is more interesting in every instance than just the `wrong` negative prompt so we'll just compare base output vs. LoRA output. Here's some more examples of base model vs. `wrong` LoRA:



`realistic human Shrek blogging at a computer workstation, hyperrealistic award-winning photo for vanity fair` — Hands are better, lighting is better. Clothing is more detailed, and background is more interesting.

`pepperoni pizza in the shape of a heart, hyperrealistic award-winning professional food photography` — Pepperoni is more detailed and has heat bubbles, less extra pepperoni on the edges, crust is crustier (?)

`presidential painting of realistic human Spongebob Squarepants wearing a suit, (oil on canvas)+++++` — Spongebob has a nose again, and his suit has more buttons.

`San Francisco panorama attacked by (one massive kitten)++++, hyperrealistic award-winning photo by the Associated Press` — The LoRA actually tries to follow the prompt.

`hyperrealistic death metal album cover featuring edgy moody realistic (human Super Mario)++, edgy and moody` — Mario's proportions are more game-accurate and character lighting is more edgy and moody.

The `wrong` LoRA is available here, although I cannot guarantee its efficacy in interfaces other than diffusers. All the Notebooks used to help generate these images are available in this GitHub repository, including a general SDXL 1.0 + refiner + `wrong` LoRA Colab Notebook which you can run on a free T4 GPU. And if you want to see the higher resolutions of generated images used in this blog post, you can view them in the source code for the post.

What's Wrong with Being Wrong?

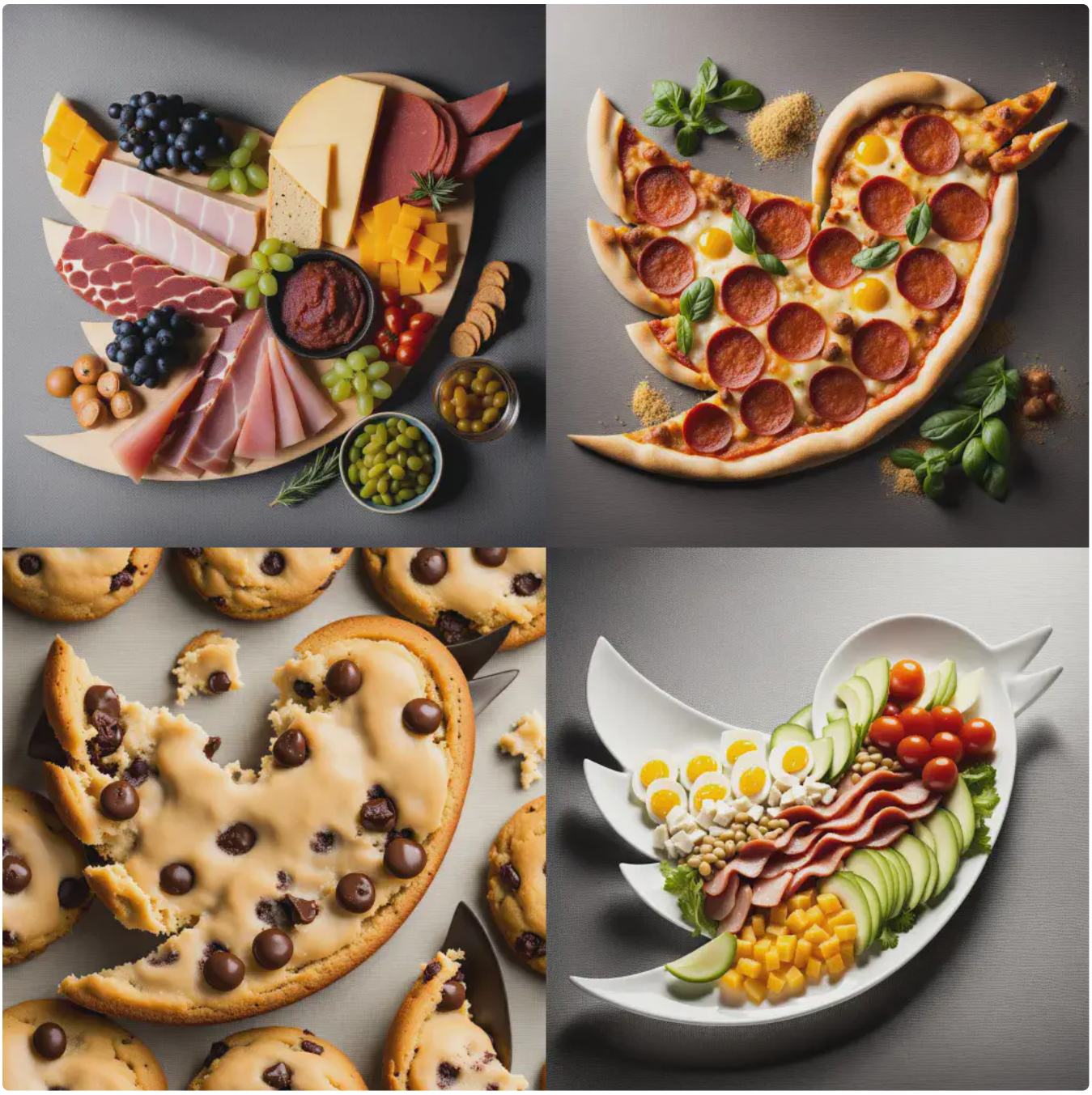
I'm actually not 100% sure what's going on here. I thought that the `wrong` LoRA trick would just improve the quality and clarity of the generated image, but it appears the LoRA is *making SDXL behave smarter* and more faithful to the spirit of the prompt. At a technical level, the negative prompt sets the area of the latent space where the diffusion process starts; this area is the same for both the base model using the `wrong` negative prompt and the LoRA which uses the `wrong` negative prompt. My intuition is that the LoRA reshapes this undesirable area of the vast highdimensional latent space to be more similar to the starting area, so it's unlikely normal generation will hit it and therefore be improved.

Training on SDXL on bad images in order to improve it is technically a form of Reinforcement Learning from Human Feedback (RLHF): the same technique used

to make ChatGPT as powerful as it is. While OpenAI uses reinforcement learning to improve the model from positive user interactions and implicitly reducing negative behavior, here I use *negative* user interactions (i.e. selecting knowingly bad images) to implicitly increase positive behavior. But with Dreambooth LoRAs, you don't nearly need as much input data as large language models do.

There's still a lot of room for development for "negative LoRAs": my synthetic dataset generation parameters could be much improved and the LoRA could be trained for longer. But I'm very happy with the results so far, and will be eager to test more with negative LoRAs such as merging with other LoRAs to see if it can enhance them (especially a `wrong` LoRA + Ugly Sonic LoRA!)

Believe it or not, this is just the tip of the iceberg. SDXL also now has support for ControlNet to strongly control the overall shape and composition of generated images:



Examples of SDXL generations using ControlNet specifying the (former) Twitter/X logo.

ControlNet can *also* be used with LoRAs, but that's enough to talk about in another blog post.

A note on ethics: the primary reason I've been researching into improving AI image generation quality is for transparent AI journalism, including reproducible prompts and Jupyter Notebooks to further the transparency. Any new novel improvements in AI image generation by others in the industry may no longer be disclosed publicly given that you can make a lot of money by doing so in the current venture capital climate. I do not support or condone the replacement of professional artists with AI.