100xEngineers

Lecture 10

Parameters for Training - Lora Talking about the steps, etc

How To Train Own Stable Diffusion LoRA Models – Full Tutorial

First – All The Things You Need For LoRA Training

The only thing you need to go through with training your own LoRA is the Kohya GUI which is a Gradio based graphical interface that makes it possible to train your own LoRA models and Stable Diffusion checkpoints without dabbling with CLI commands.

You will also need to install a few dependencies to be able to run Kohya GUI on your system.

To download the Kohya GUI, simply head over to its GitHub page, and locate the small “Releases” tab on the right of the page. Underneath, you should find a shortcut to the latest release of the Kohya interface – click on it.

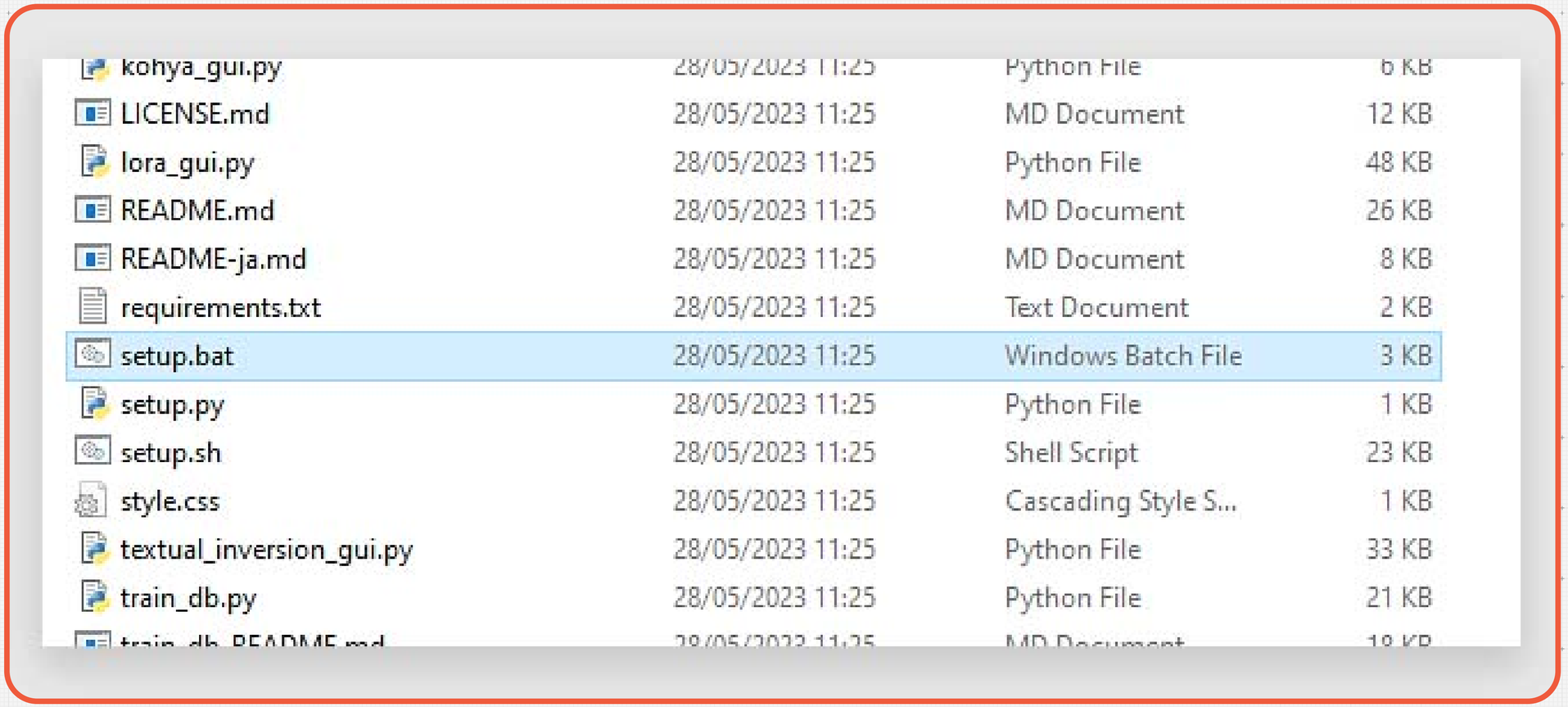
On the newly opened page, you will see two downloadable files – you are interested in the “Source code (zip)” file which will contain the latest release of the Kohya GUI.

After you download this file, extract it. Now, here comes the time for the quick installation.

Here are the dependencies that you'll need for the Kohya GUI to work:

- Git – a simple and quick install, if you don't have Git already installed in your system.
- Python 3.10 – Very important: during the installation click the checkbox that says "Add Python to PATH environment variable".
- Visual Studio 2015, 2017, 2019, and 2022 redistributable – the last thing you need to get the Kohya GUI running.

How To Install The Kohya GUI



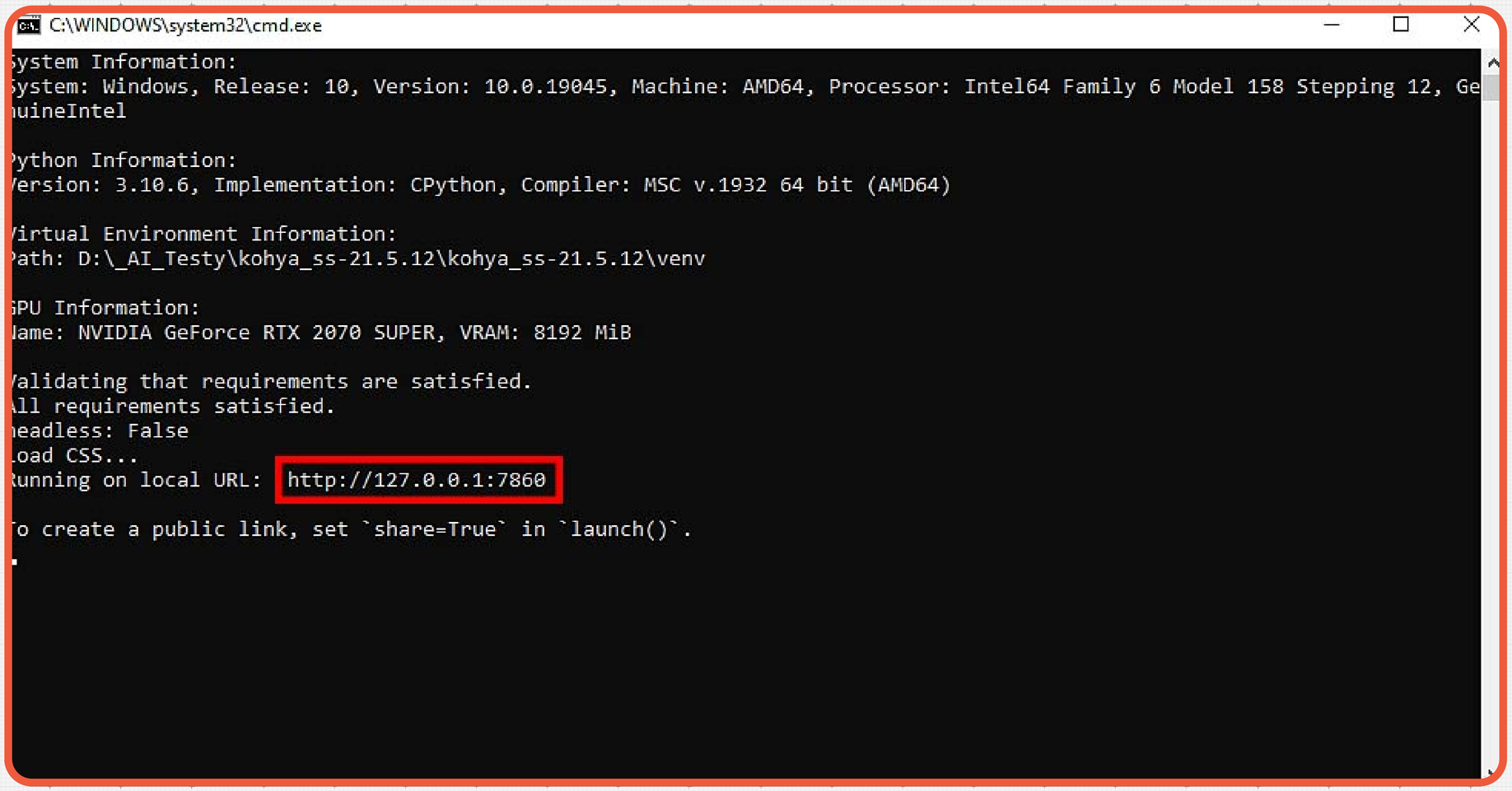
kohya_gui.py	28/05/2023 11:25	Python File	6 KB
LICENSE.md	28/05/2023 11:25	MD Document	12 KB
lora_gui.py	28/05/2023 11:25	Python File	48 KB
README.md	28/05/2023 11:25	MD Document	26 KB
README-ja.md	28/05/2023 11:25	MD Document	8 KB
requirements.txt	28/05/2023 11:25	Text Document	2 KB
setup.bat	28/05/2023 11:25	Windows Batch File	3 KB
setup.py	28/05/2023 11:25	Python File	1 KB
setup.sh	28/05/2023 11:25	Shell Script	23 KB
style.css	28/05/2023 11:25	Cascading Style S...	1 KB
textual_inversion_gui.py	28/05/2023 11:25	Python File	33 KB
train_db.py	28/05/2023 11:25	Python File	21 KB
train_db_README.md	28/05/2023 11:25	MD Document	18 KB

In your newly extracted folder, you will find the **“setup.bat” file**. Right click it and select **“Run as an Administrator”** to run the installation script with elevated privileges.

For most applications and in most cases, you'll want to set these as follows:

- In which compute environment are you running? – This machine
- Which type of machine are you using? – No distributed training
- Do you want to run your training on CPU only? – No
- Do you wish to optimize your script with torch dynamo? – No
- Do you want to use DeepSpeed? – No
- What GPU(s) (by id) should be used for training on this machine? – All
- Do you wish to use FP16 or BF16 (mixed precision)? – If you have a 30XX series GPU (or higher) – select BF16, otherwise select FP16.

Note: Keep in mind that the whole installation process might take quite a bit, depending on your network connection speed. The install script is downloading a lot of additional files.



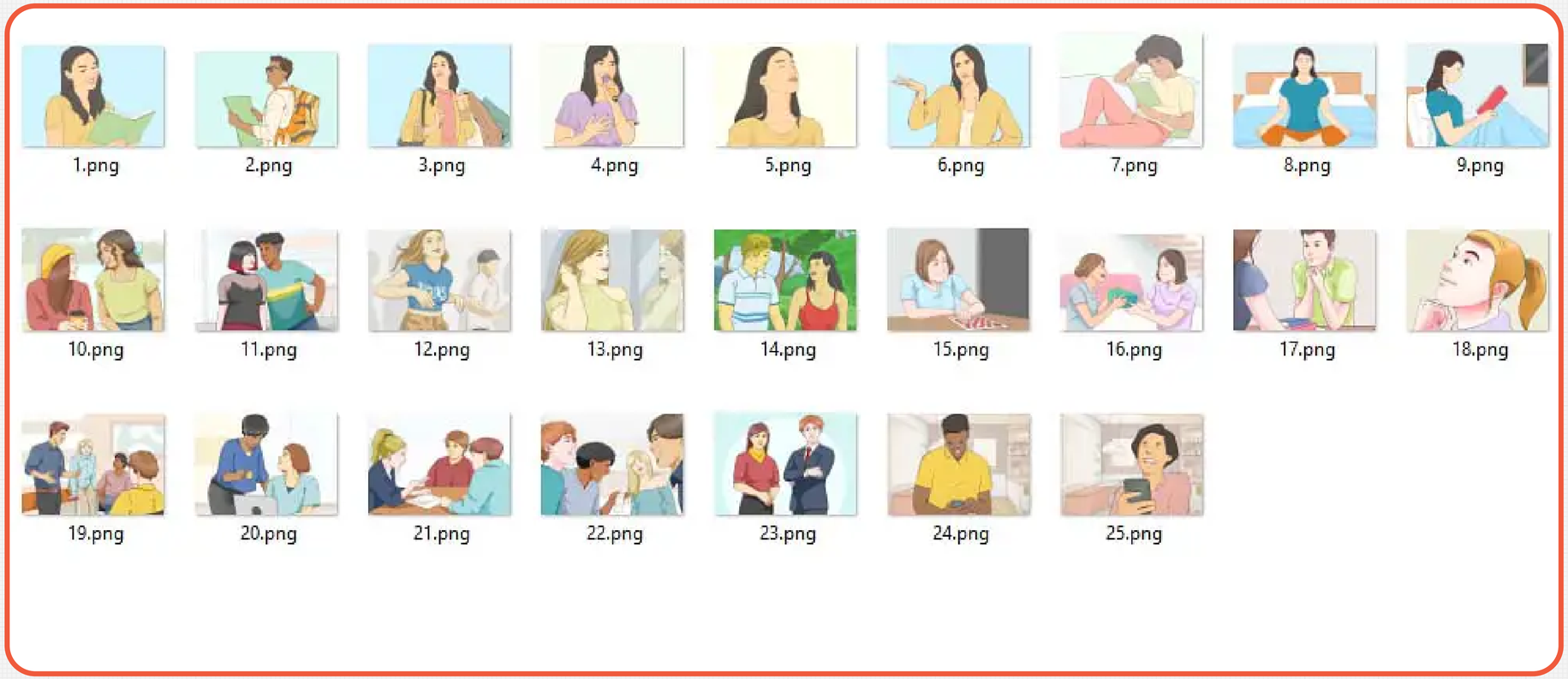
Once the Kohya terminal opens, the GUI will be hosted on your localhost on the port 7860. Enter **“http://127.0.0.1:7860/”** (without the parentheses) into your web browser address bar.

There you go! The Kohya GUI is up and running, and half the work is done!
Now, the only other things you need, is a set of images you want to train your new LoRA with.

Generally you will want to get somewhere between 15 to 100 quality images with consistent styling to train your very first LoRA model. For now, the picture size doesn’t matter.

Check out also: Using Hypernetworks Tutorial Stable Diffusion WebUI – How To

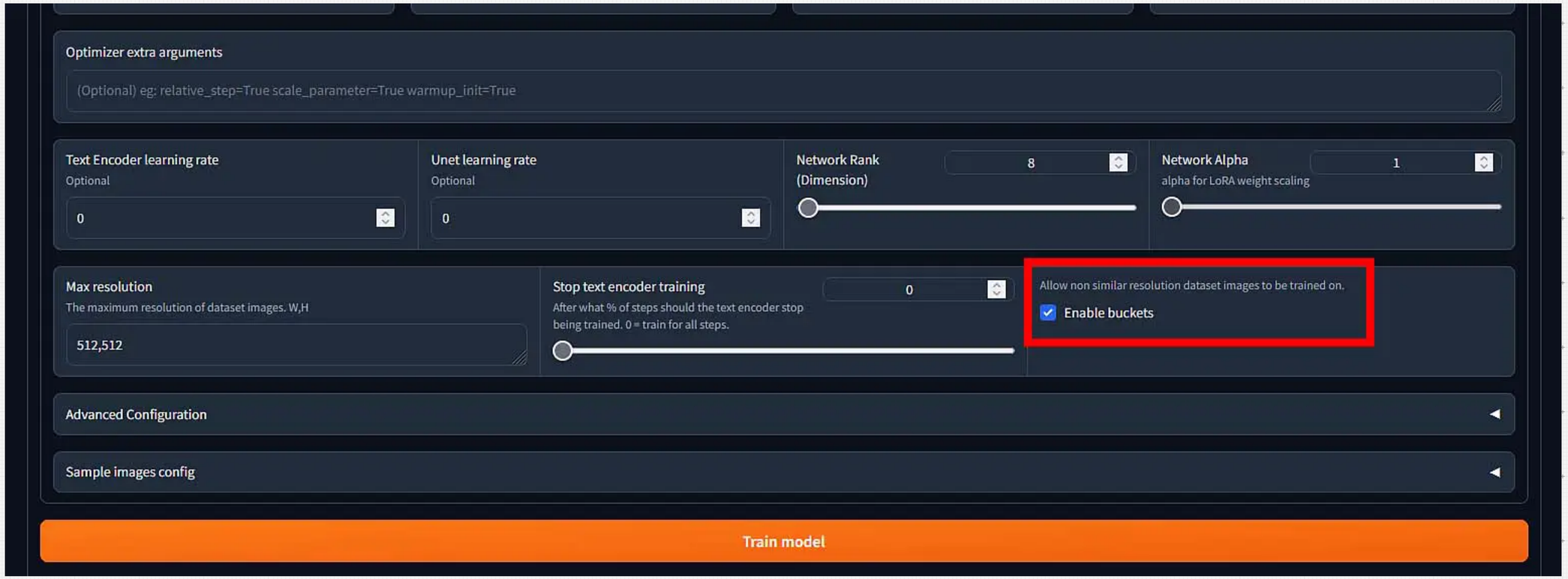
How Many Images Do You Need To Train a LoRA Model?



For starting out, I recommend picking around 20-50 images in the style you're going for.

Check out also: Stable Diffusion WebUI Settings Explained – Beginners Guide

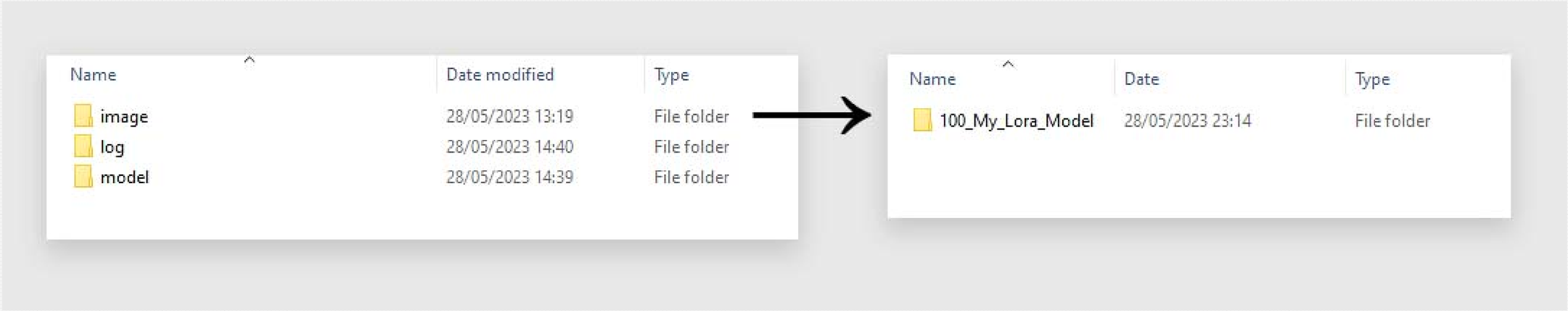
Do You Have To Cut Pictures to 512×512 px For LoRA Training?



The short answer is: no, you don't have to resize your images to 512×512 pixels before LoRA training, at least not when using the Kohya GUI.

How To Train a LoRA Model in 4 Simple Steps

Step 1: Prepare 3 Folders



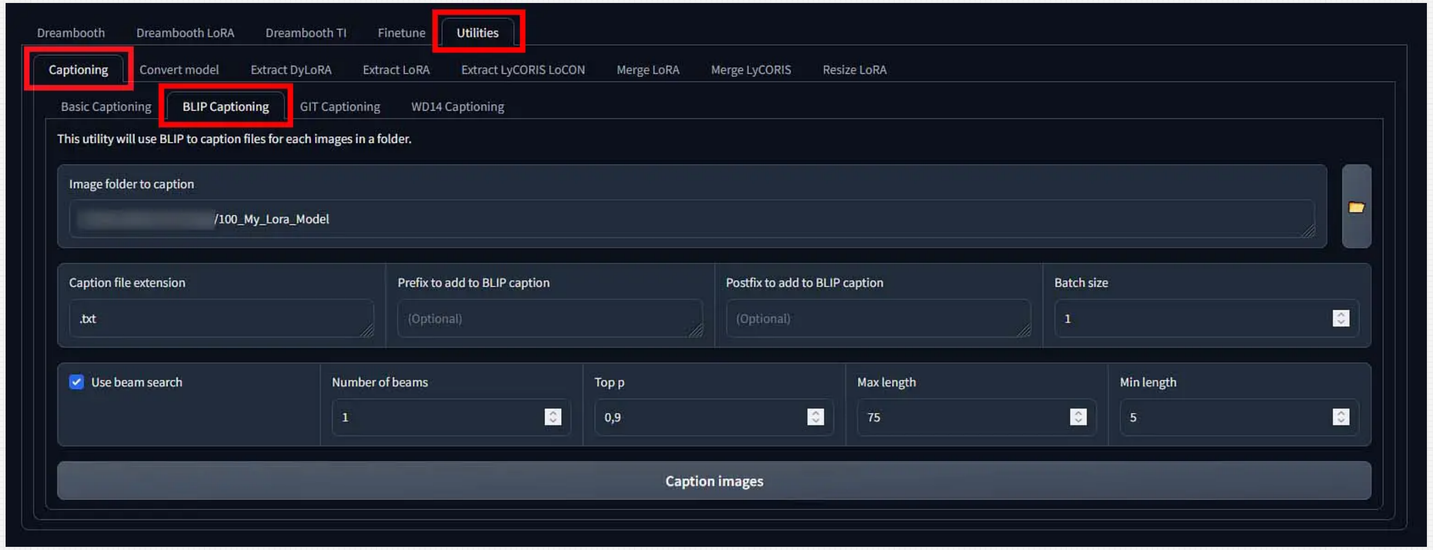
Now, you need 3 folders: image, log and model. These names are arbitrary, but we recommend to use them for clarity.

The images folder will contain another folder with your actual training images set – this folder is a folder which name should change depending on how many training steps per image you want when training your LoRA – this is very important.

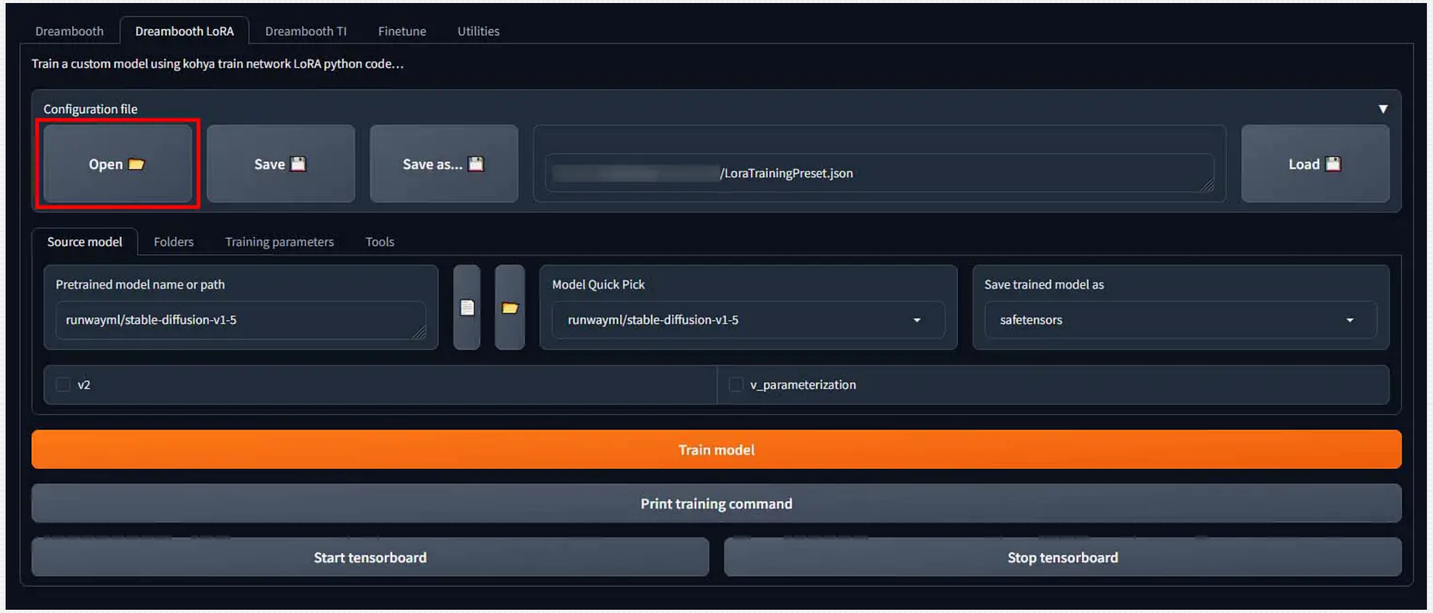
How many steps should you go for? If you only have around 10 images, set this number to around 150, if you have around 20 images and up, you can safely leave it as 100. For the sake of simplicity in our case we'll leave it as 100.

The last two folders – log and model folders will be automatically willed with your final trained LoRA model file and log files from the training process.

Step 2: Caption Your Images Using BLIP



Step 3: Input Right Settings

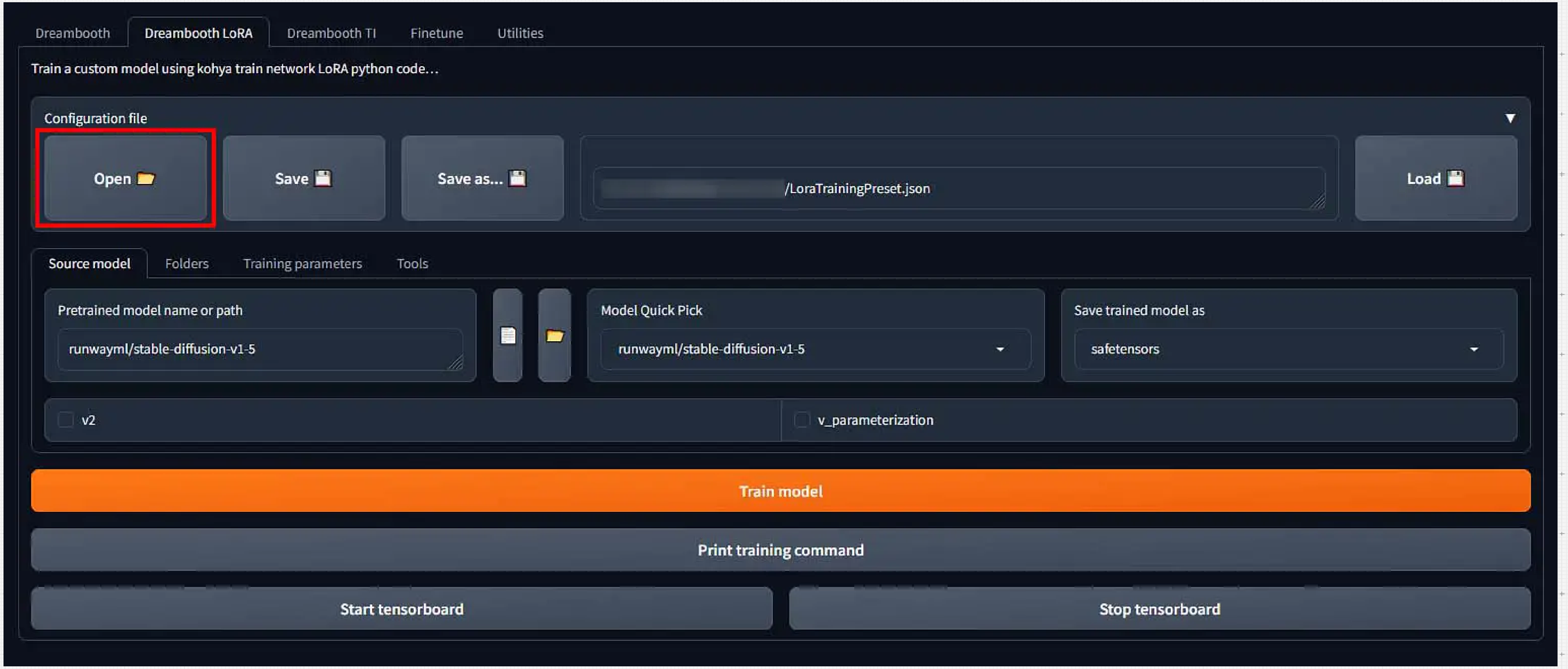


For the training to be successful, you need to provide Kohya GUI with a text file containing a short description of each of the images in your training set. Luckily, the Kohya GUI allows you to utilize the BLIP model to automatically caption all the images you've prepared. You can find it under Utilities -> Captioning -> BLIP Captioning.

The option "Prefix to add to BLIP caption" allows you to automatically add a certain text prefix to all of the generated caption files. Use it if you want to link the LoRA embedding with for instance a specific style or character name you're training your LoRA on.

After the captioning process is complete, you can further refine your captions by editing the .txt files manually. This is especially helpful if your images contain non-generic elements or characters which can't be recognized by BLIP (ex. anime characters, photos of your friends, etc.)

Step 3: Input Right Settings

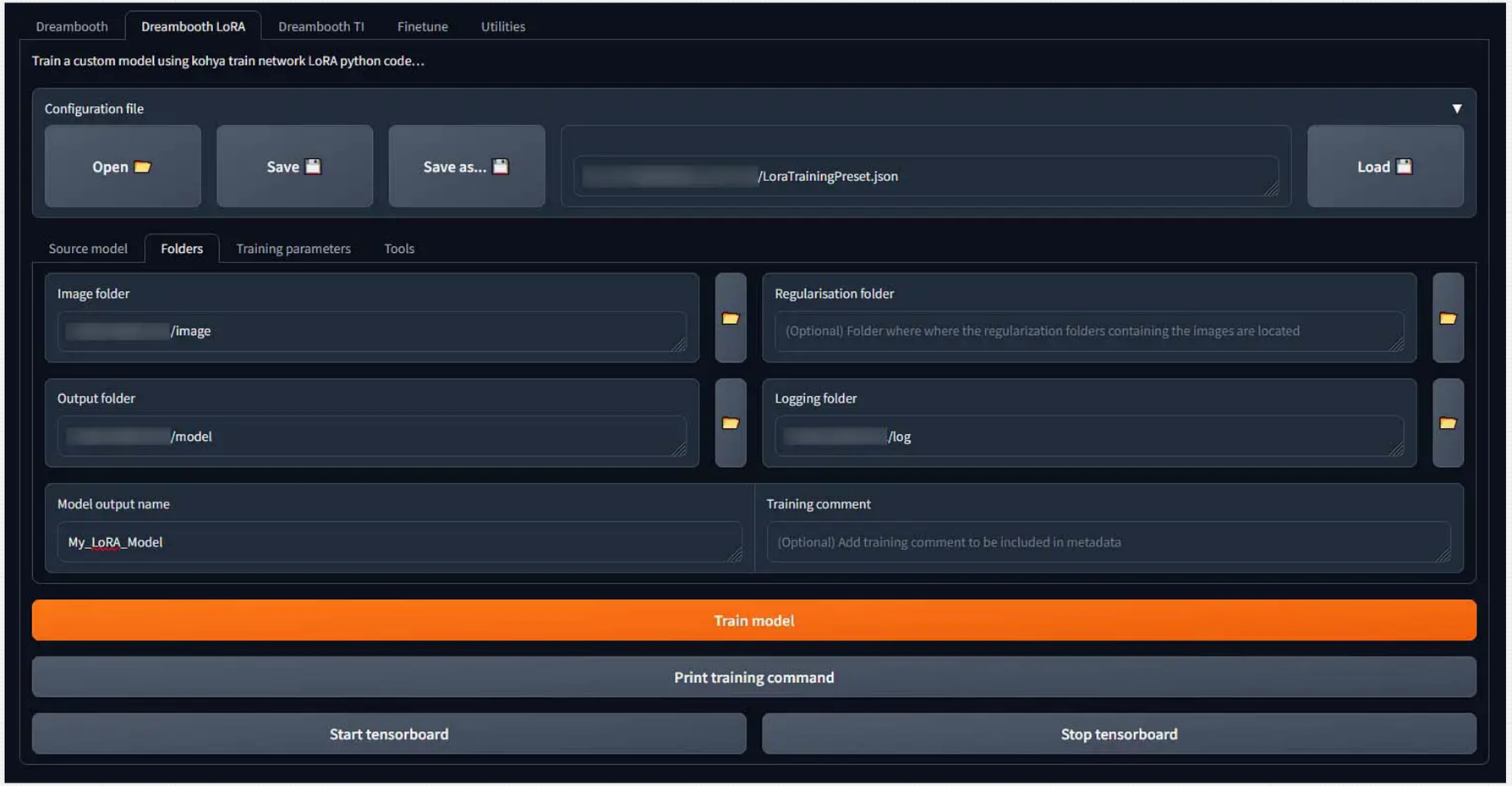


Now, we're going to actually train our LoRA – to make the process fun, we will use a few images from WikiHow.com that you all know and love!

You can learn more about where do the bizarre WikiHow images come from in this article. It's a really interesting story!

First things first, download our settings preset file here. Importing it will do most of the initial setup work for you.

Step 4: Train Your LoRA Model



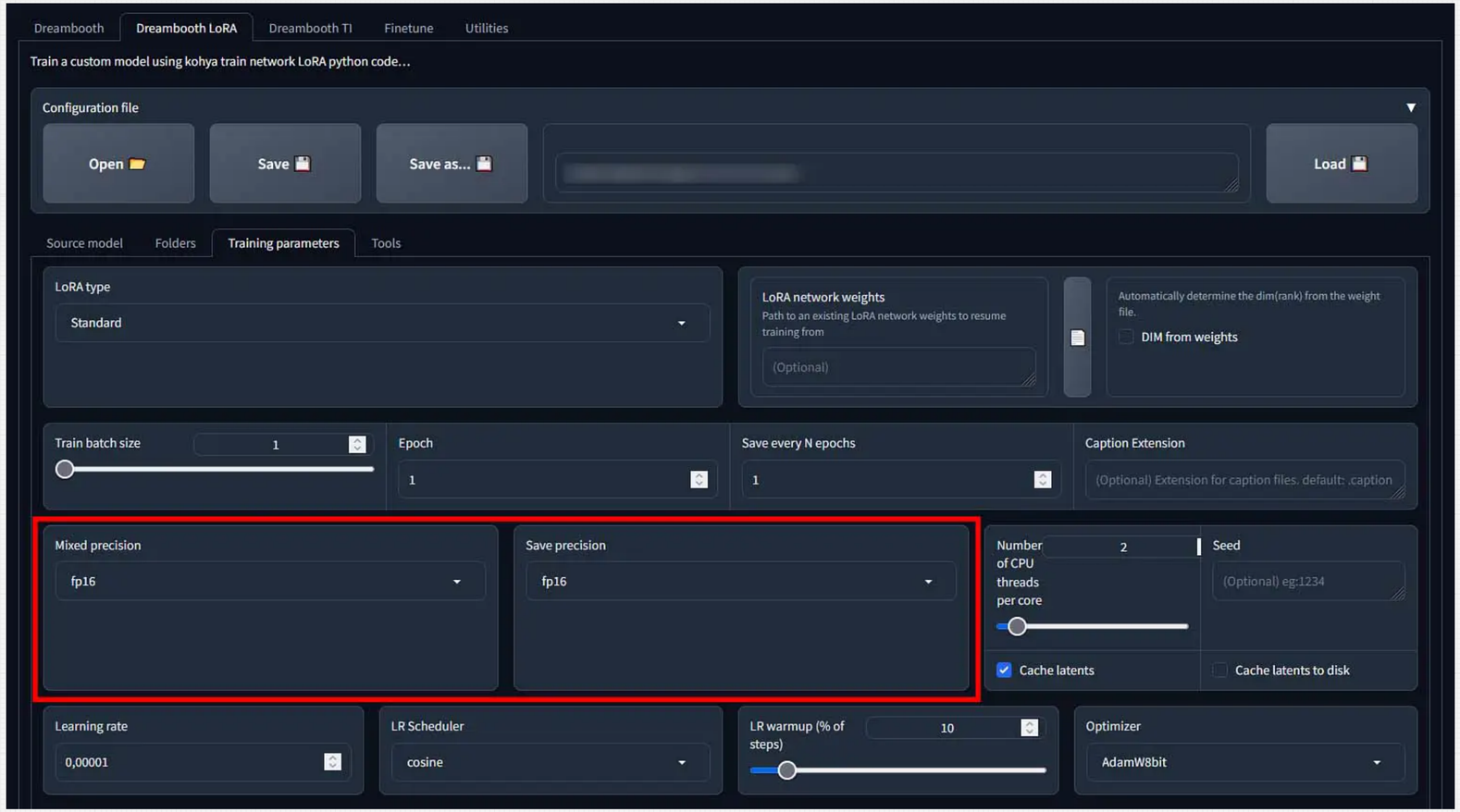
One last thing you need to do before training your model is telling the Kohya GUI where the folders you created in the first step are located on your hard drive.

To do this, head over to the “Folders” tab, and input the exact file paths to your image, output and log folder. The paths will depend on the place where you’ve actually created these folders in the first place. You can also click the little folder buttons next to each path input to select the appropriate folders using your file browser.

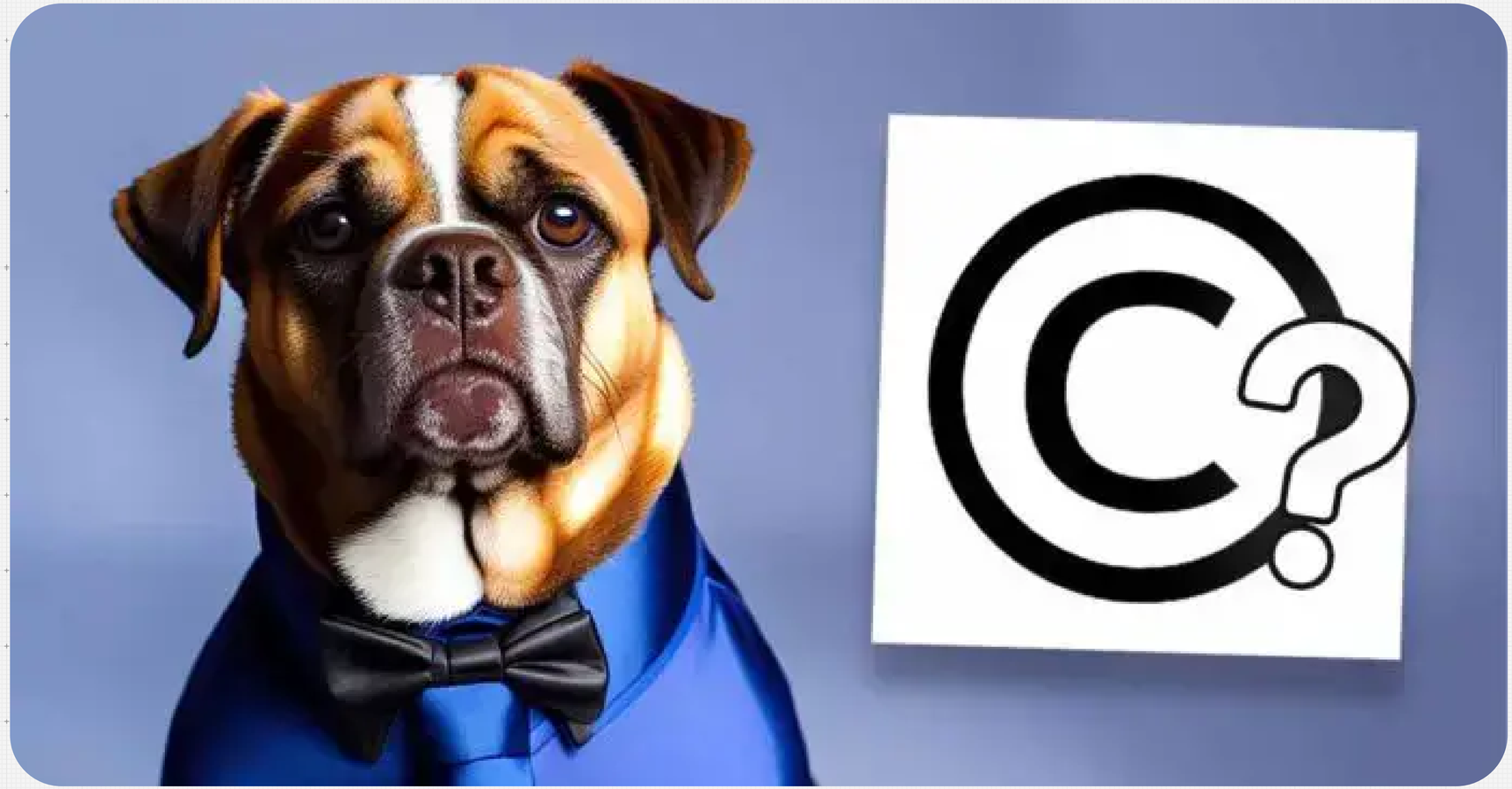
Very important: When selecting the location of the image folder, select the image folder itself, not the actual “100_My_Lora_Model” folder with your training images that should be inside the image folder.

The “Regularization folder” field doesn’t have to be filled in in our case, so it’s safe to leave it empty.

I’m Getting The bf16 Mixed Precision Error – How To Fix That?

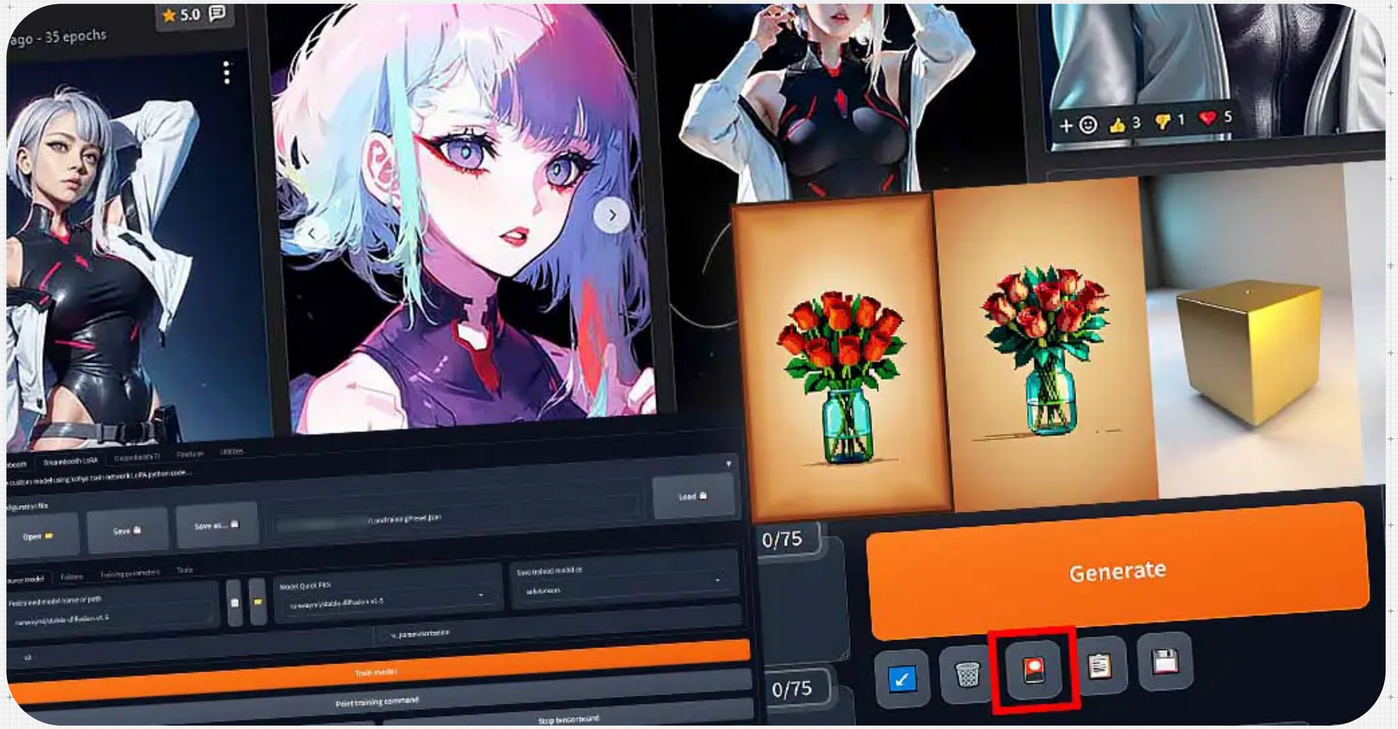


LoRA Models & Fine-tuning – Important Legal Remarks



Read more about quite controversial legal issues with training material here:
Why Do Watermarks Appear In Stable Diffusion Generated Images? (Explained!)

So, That's It – You've Trained Your Very First LoRA model!



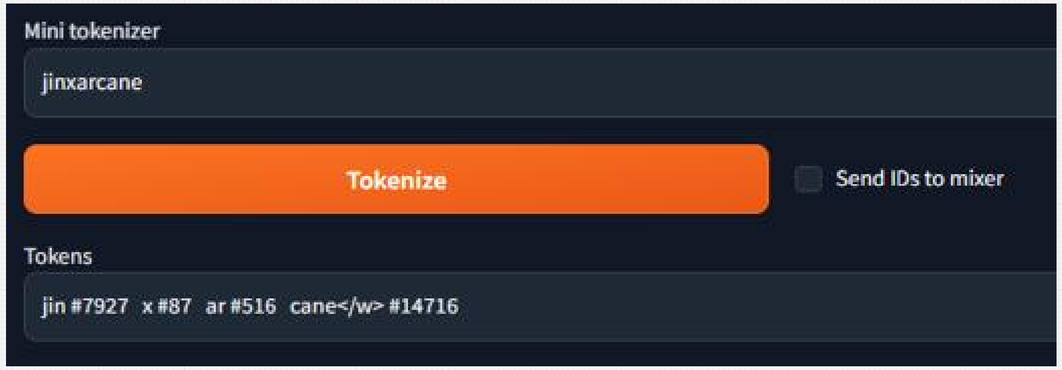
Yes, it's this easy! Now you can experiment with different training settings, different image sets, captioning styles and much much more!

With the current version of Stable Diffusion Automatic1111 WebUI, you can use LoRA models trained with Kohya GUI without any additional plugins or extensions! **To see how to use LoRA models with the Automatic1111 WebUI, click here.**

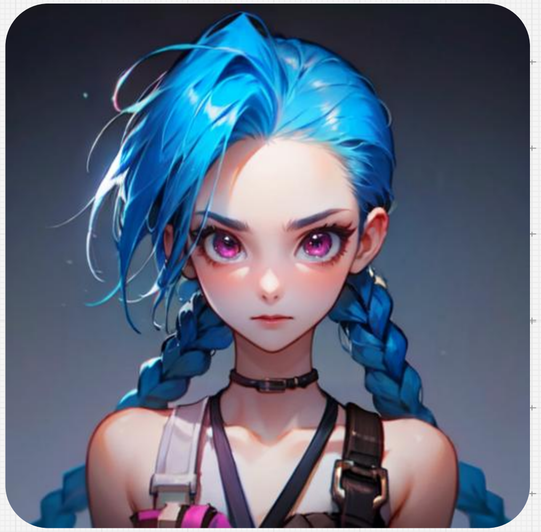
[ADVANCED] Multi-concepts and balancing datasets

In the previous section, we saw that Arcane Jinx concept had some "pink" artefacts on the outfits. This problem is due to **concept bleeding.** Let me explain.

This LoRA was composed of 5 different outfits whose tags were: **jinxdef, jinxarcane, jinxelf, jinxfirecracker,** and **jinxrnd.** Let's look at the tokens of the closest outfits (**jinxdef** and **jinxarcane**).



You can see that both embeddings share two tokens, **jin** and **x**. These two tokens will be trained alongside the remaining parts **def** and **arcane** when training. In other words, the "jinx" tag is **trained by all outfits,** whereas the custom tokens **def** and **arcane** are **trained on a unique outfit.** To illustrate this, I generated an image with only the tag "jinx", highlighting the "fusion" that occurs:



Not once were my images tagged **jinx,** **yet the LoRA correctly generated an image of jinx.** This notion is important because it means a tag other than my activation tag was trained! Now why is it only generating the outfit associated with **jinxdef** and not the other ones? Simply because, out of the dataset, it made up almost **60%** of the whole.

In short, the tag **jinxdef = jinx + def** concepts. This means that, when training **jinxdef,** it will bleed into **jinxarcane.** For my use case, this bleeding is fine because parts of jinx (like the blue hair, twin tails, eyes, etc.) are similar. But for other parts, like the outfits, this bleeding is destructive. Let me illustrate with an example:

These are two images generated for the tag **jinx and def** with their respective attention heatmap (see DAAM section to reproduce this). As you can see from the heatmap, **jinx and def** both impact the outfits and face. While this is fine for **def** to change the face, it is not acceptable for **jinx** to change the outfits because this means that other activation tag, such as **jinxarcane**, will see their outfits slightly modified by the **jinx** part of the tag. In the previous section, where we saw the pink artefacts on the outfit for jinx arcane, this was due to the bleeding of the **jinxdef** outfit into the **jinx** token.

Note: this also means you can take advantage of this. For example, if you had a "school" uniform, you could use the activation tag "jinxschool" which would make use of the "school" tag to generate the image (essentially modifying the concept of school to fit the new outfit).

For more complex concepts, this "concept bleeding" is essential to understand because they are composed of multiple new simple concepts that have potential bleeding from every other tag. If improperly tagged, a difficult concept might never come to life.

Now, how do we fix this? Three solutions:

- use **exclusive activation tag:** e.g. I could tag jinxdef as **jinx** and jinxarcane as **arcane**, but this would mean modifying the concept of "arcane" to represent Jinx Arcane. You can also use "rare" tokens to combat this for example **sks**.
- **modify the number of repeats** to balance the dataset. In this case, I have four times as much jinxdef as jinxarcane. By setting jinxarcane to a repeats of 3, I balance the jinx token between two outfits, and thus get no more problems
- use **pivotal tuning** to create a new embedding

In my case, I want **jinxdef** and **jinxarcane** to benefit from each other. Therefore, I will use the second solution. As for the other tags, I should use the third solution because they are entirely new concepts, but I chose to use the first solution instead.

Take some examples from here
https://stable-diffusion-art.com/train-lora/

What can LoRAs do?

- **Quality improvements** (eg. Detail Tweaker)
- **Styles/aesthetics** (eg. Arcane style, Studio Ghibli style, Anime Lineart)
- **Characters or people** (eg. Makima, Cute_girl_mix)
- **Clothing or objects** (eg. Hanfu, Taiwanese Foo)
- **Settings** (eg. School building)

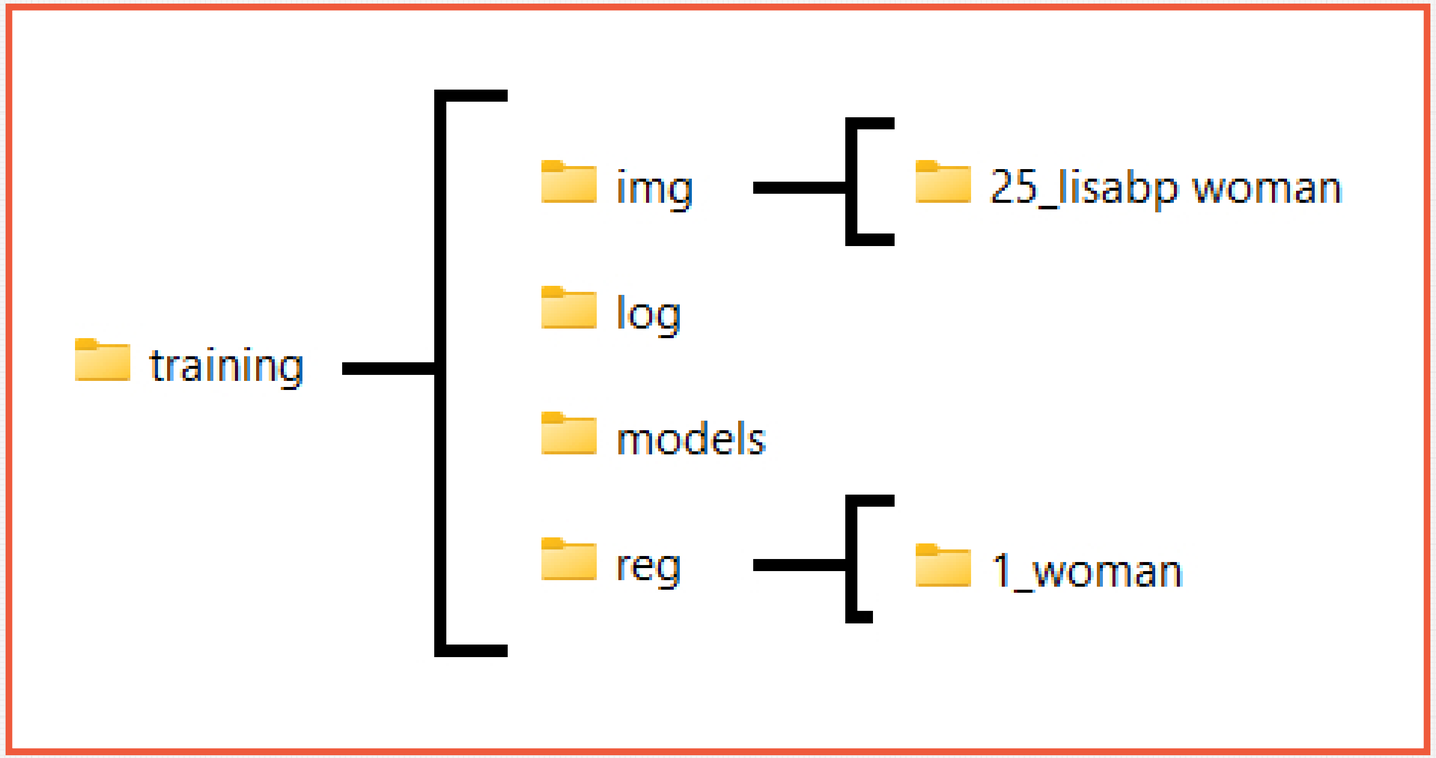
Training LoRAs

Baking your first LoRA!

Training LoRAs

Step 1: Prepare training images

Step 2: Folder Structure



Step 3: Regularization Images for Training

using the "woman" dataset woman_v1-5_mse_vae_ddim50_cfg7_n4420.zip.

Download the set that you think is best for your subject. Extract the zip folder.

Copy over at least X number of images to the subfolder in the reg folder you created in the previous step (mine is called 1_woman), where X = REPEATS * NUMBER_OF_TRAINING_IMAGES.

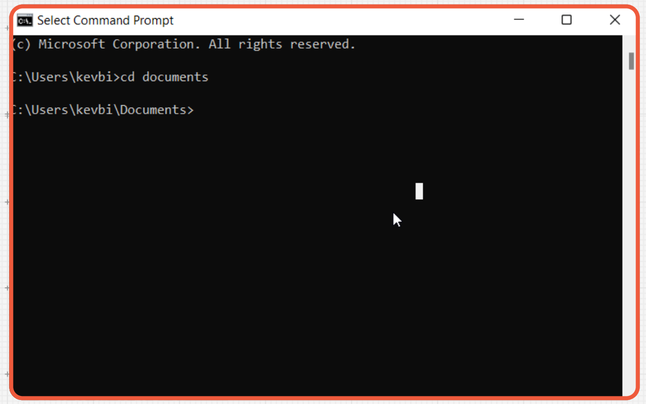
In our case this is 6*34 = 204 images.

Step 4: Install Koyha SS

Required Dependencies

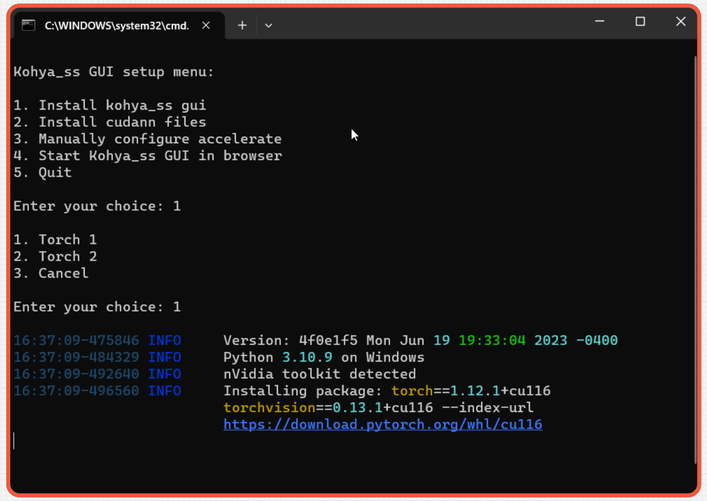
- Install Python 3.10
    - make sure to tick the box to add Python to the 'PATH' environment variable
- Install Git
- Install Visual Studio 2015, 2017, 2019, and 2022 redistributable

Step 4: Install Koyha SS



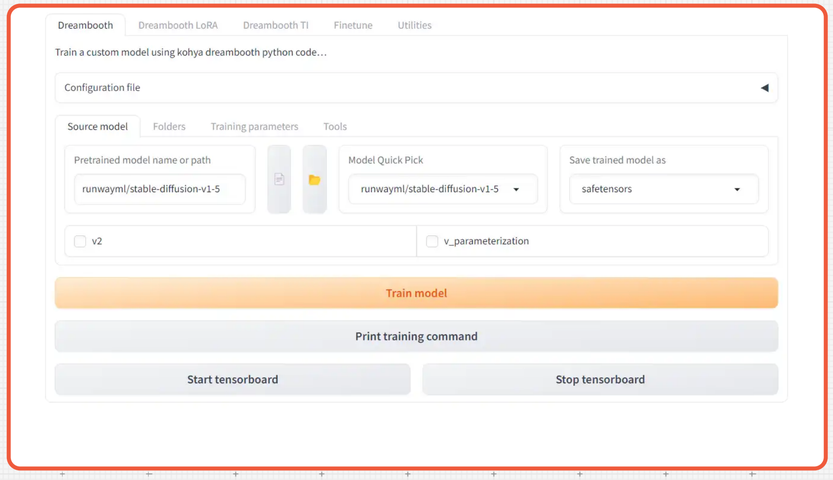
Use this command to move into folder (press Enter to run it):

1. cd FOLDER_NAME
2. cd ..
3. dir
4. cd documents



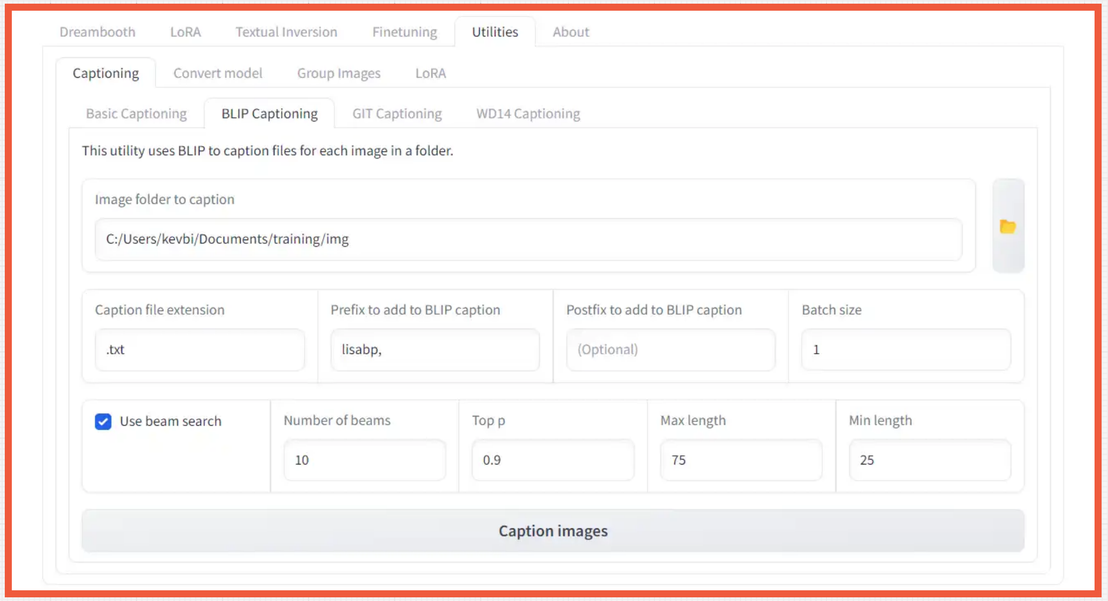
Use this command to move into folder (press Enter to run it):

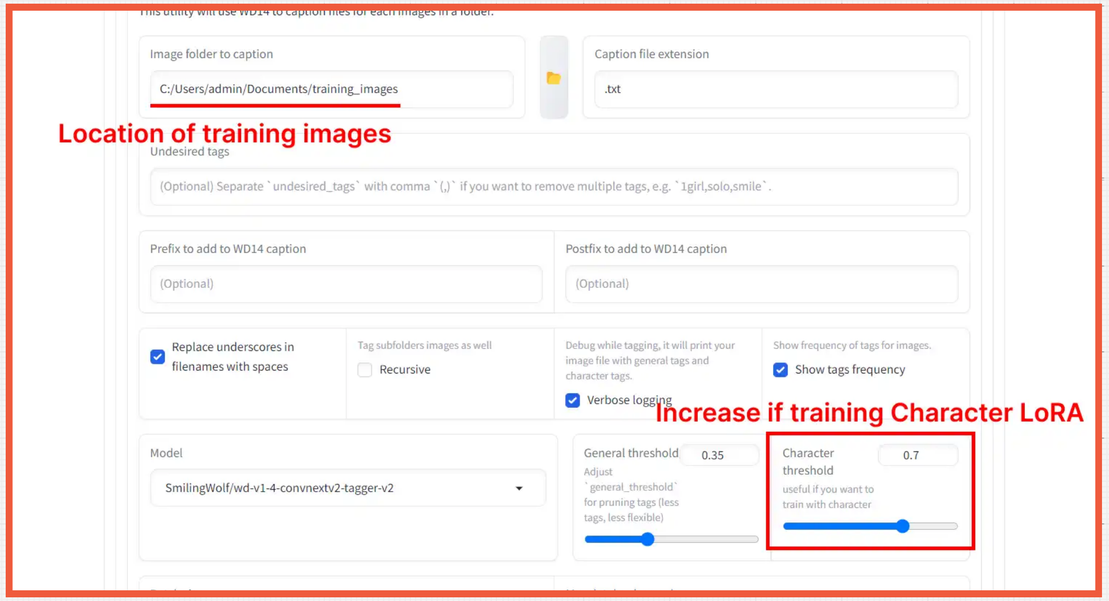
git clone https://github.com/bmaltais/kohya_ss.git
cd kohya_ss
.\setup.bat



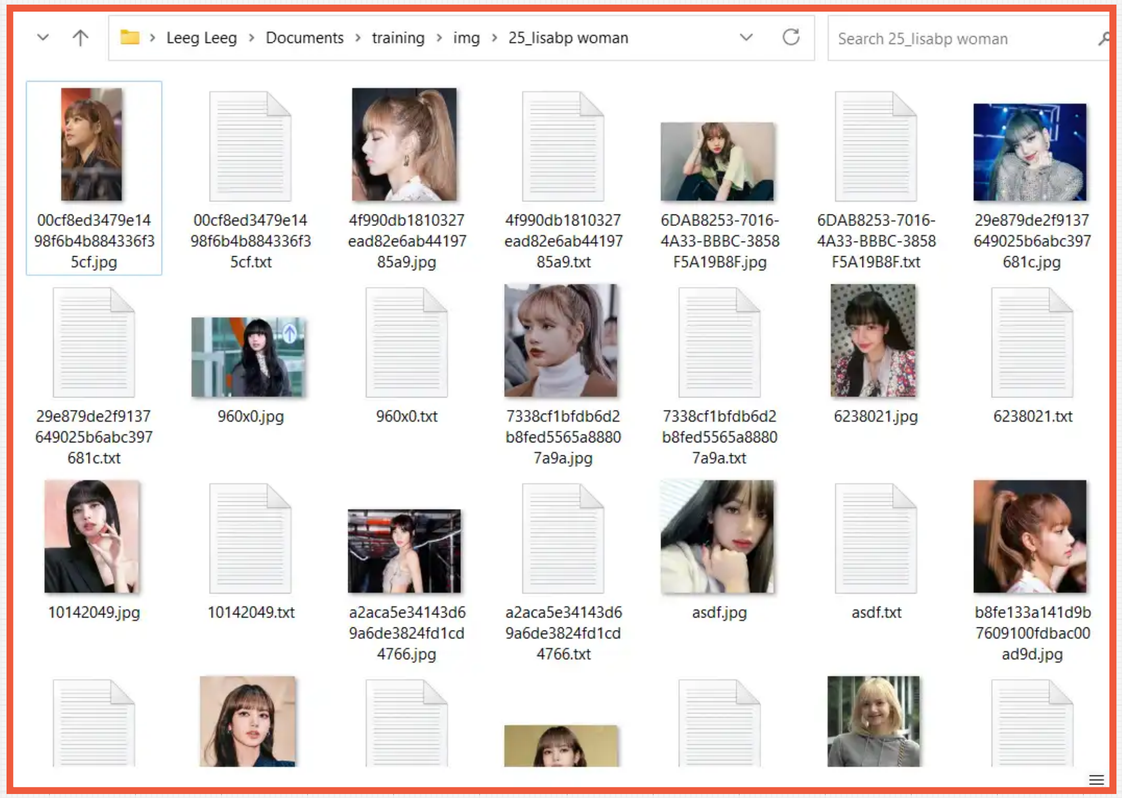
Running on local URL: http://127.0.0.1:7860

Step 5: Caption Images

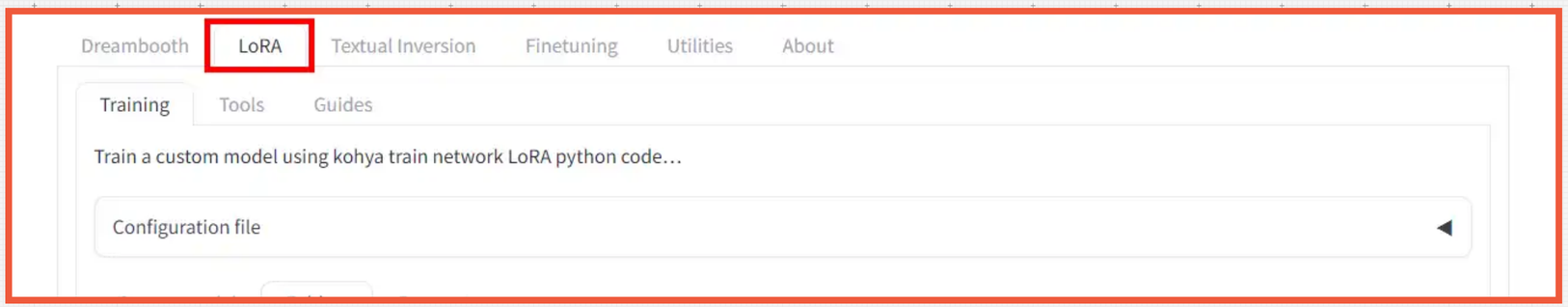




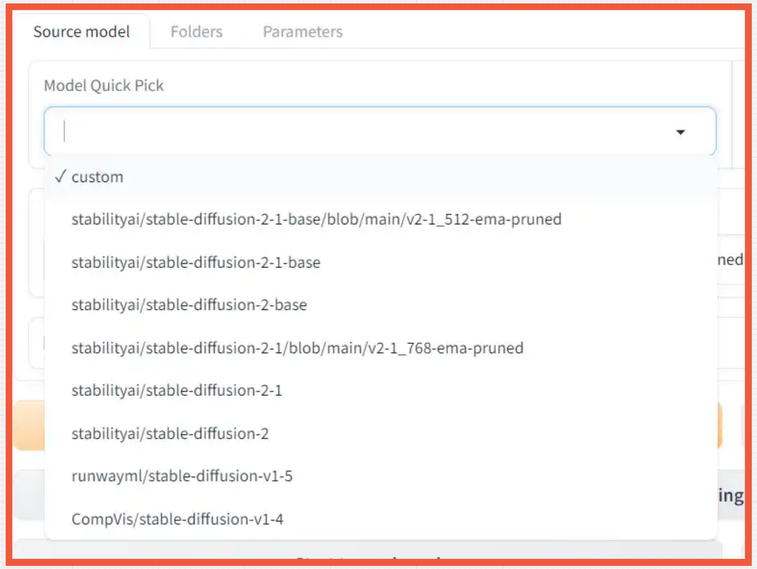
Step 5: Caption Images

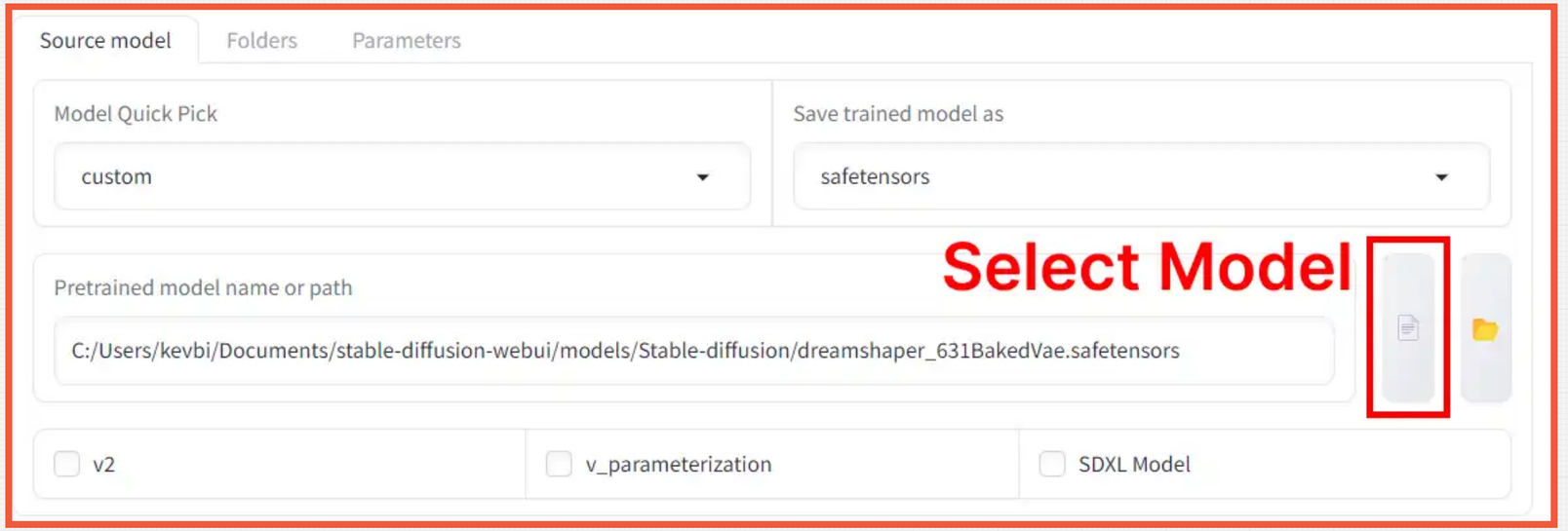


Step 6: Training

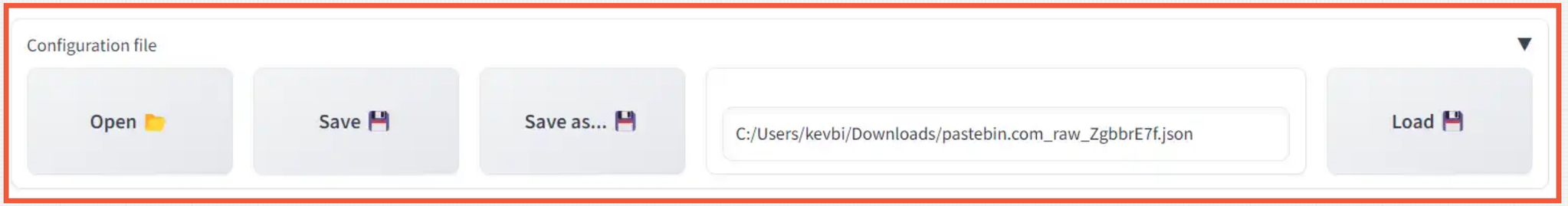


Model Selection





Configuration File

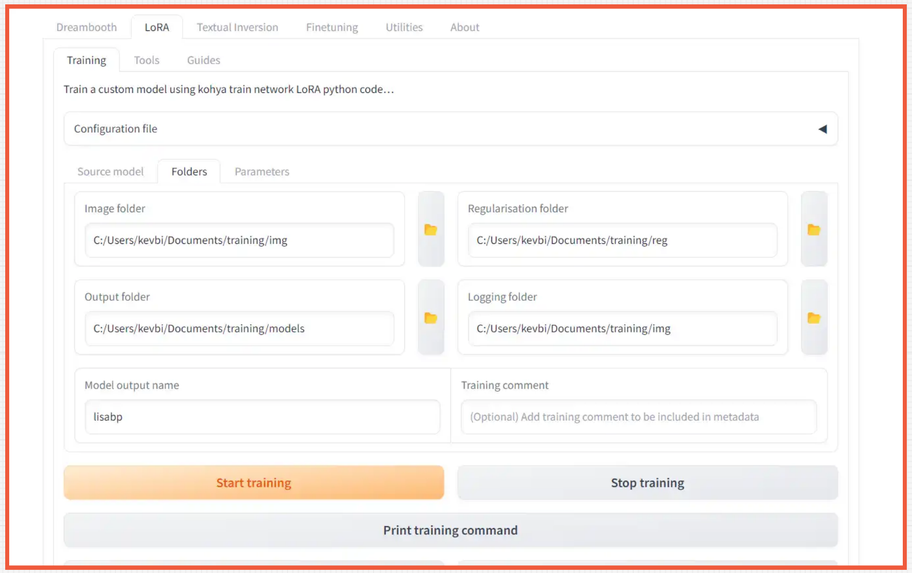


Download the configuration file (Go to link, then Right Click page -> Save as), then change the extension so the filename is pastebin.com_raw_ZgbbrE7f.json

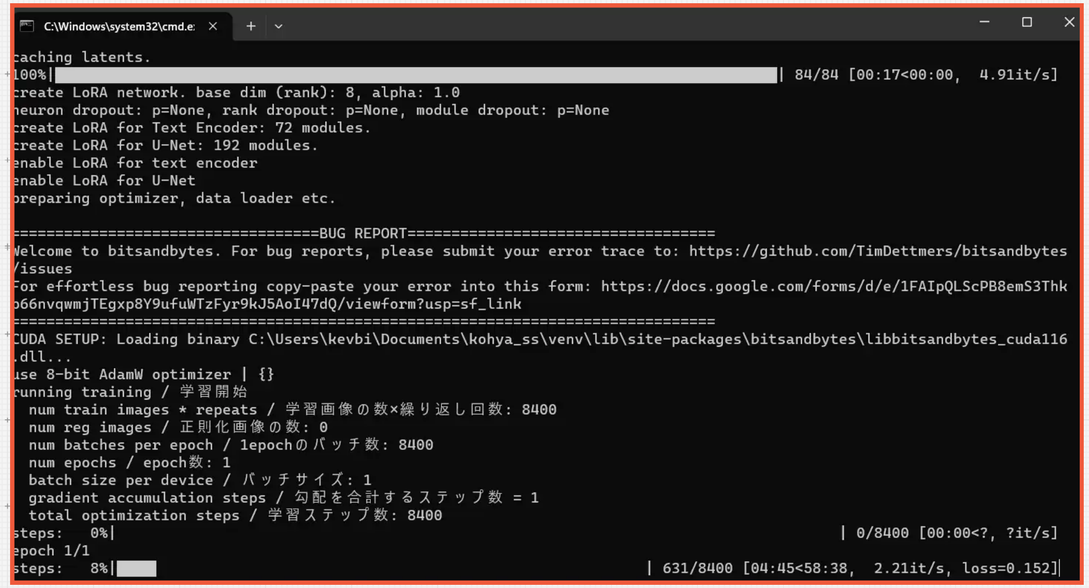
Set Folders

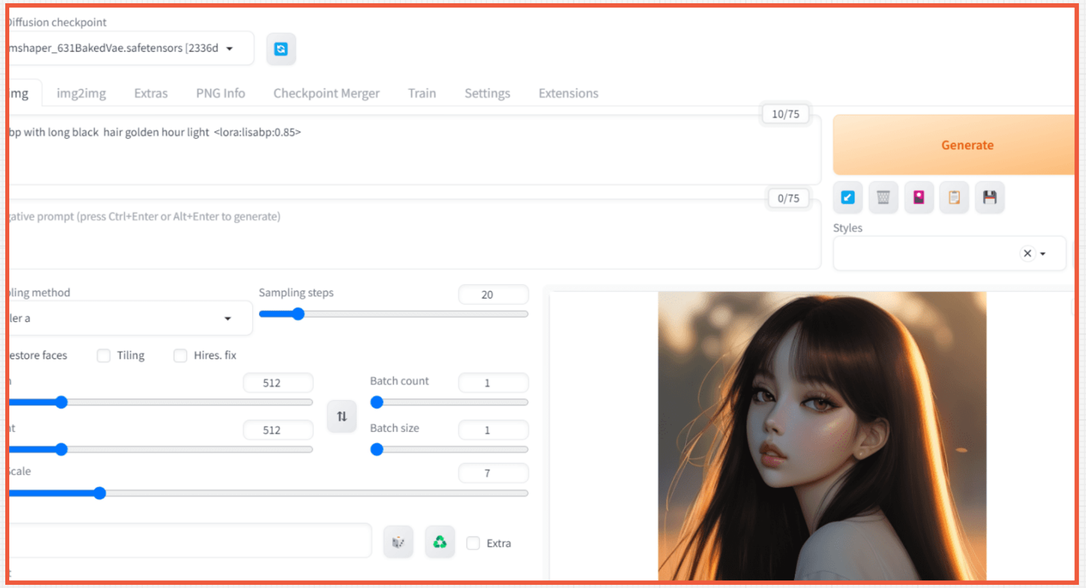
Go to the "Folders" tab and set your folders:

- Image folder: location of img folder you made in previous step (NOT the numbered subfolder inside it)
- Output folder: folder to put your LoRAs in when complete
- Regularization folder (optional): location of reg folder that contains a subfolder with your regularization images
- Logging folder (optional): folder to output your logs to
- Model output name: Name of your outputted LoRA. I usually like to name the LoRA the trigger word followed by a version number like "_v1", in case I train a few (most of the time, it takes multiple tries to get it right).



Start Training!

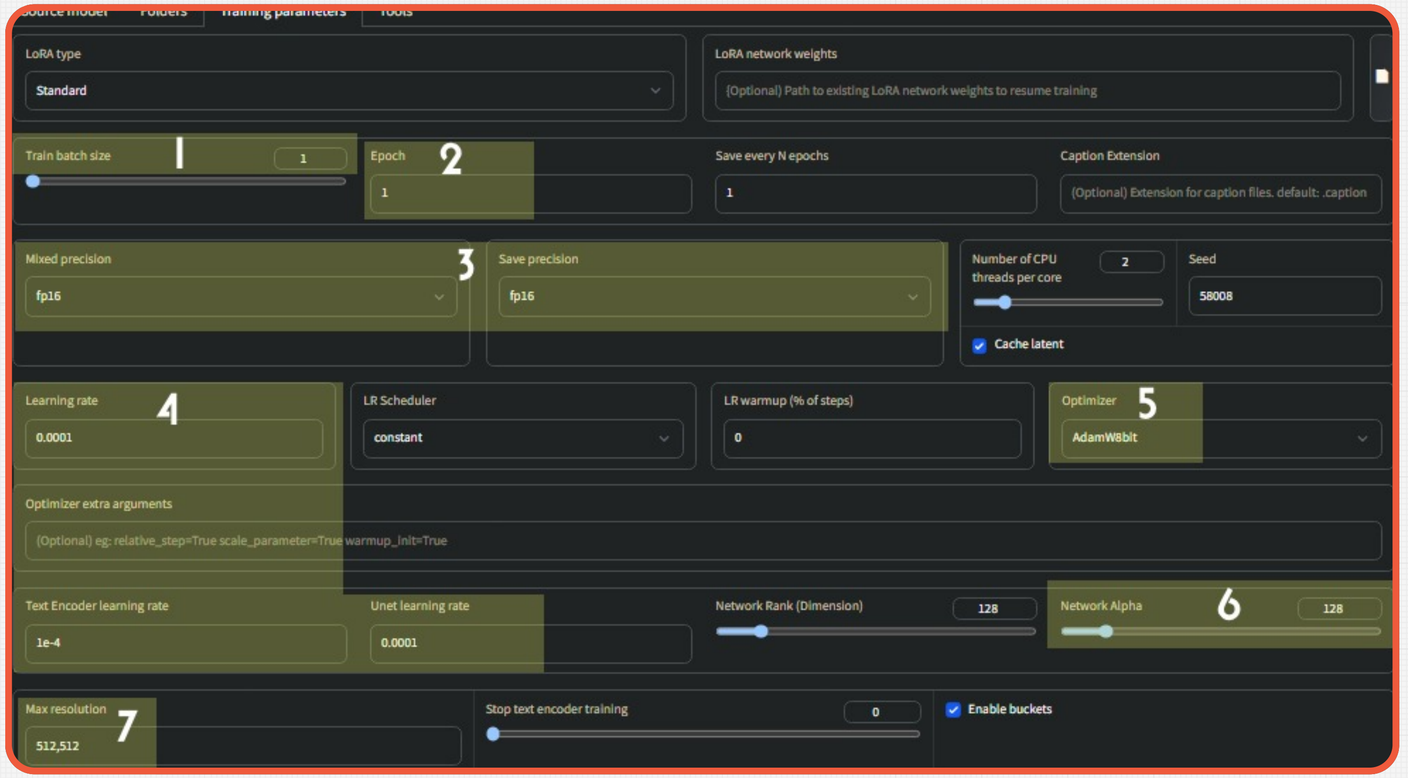




Make sure to include the LoRA keyphrase your trigger word in your prompt. As an example:
(Most likely you don't want to set the weight at the maxium of 1)

lisabp with long black hair golden hour light <lora:lisabp:0.85>

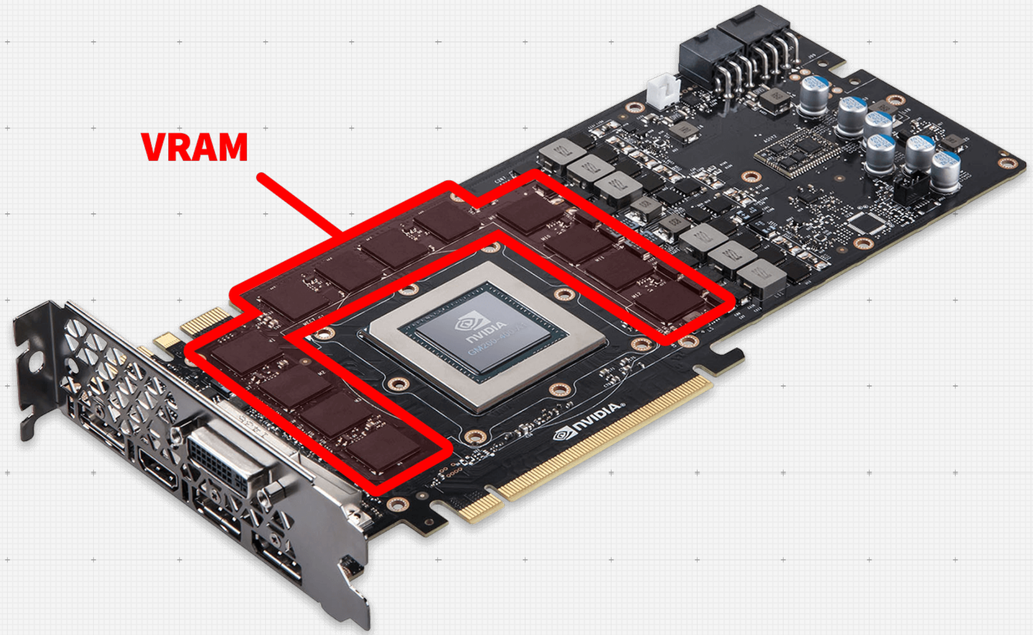
1. Batch size
2. Epoch
3. Mixed Precision / Save Precision
4. Learning Rate / Text Encoder Learning Rate / Unet Learning Rate
5. Optimizer
6. Network Rank (Dimension) / Network Alpha
7. Max Resolution



1. Batch size

How many images are trained simultaneously.

Higher values speedup training at the cost of VRAM usage



If you have:

6-8GB VRAM - leave this at 1
10GB - set to 2
12GB - set to 3

(**NOTE:** You will need to change learning rate proportional to batch size, see 4)

2. Epoch

The number of epochs is how many complete passes the learning algorithm makes through the entire training dataset.

Every epoch, the algorithm goes through all the training data, updating the LoRA based on the accumulated information.

Repeats refers to how many times each individual image is trained within an epoch.

The number of repeats are set by this folder name inside the img folder: **11_lisabp woman. This means we'll have 11 repeats per image.**

If I have 34 images and 11 repeats, **One epoch will be 34 * 11 = 374 steps.**

If I specify 8 epochs, I will be training on a total of **8 * 374 = 2958 steps**.

More epochs usually yields better results.

I recommend **not exceeding 3500 total steps for a LoRA.**

Save every n epochs

This is a great way to output models midway throughout the training.

Sometimes models trained on too many steps are overcooked.

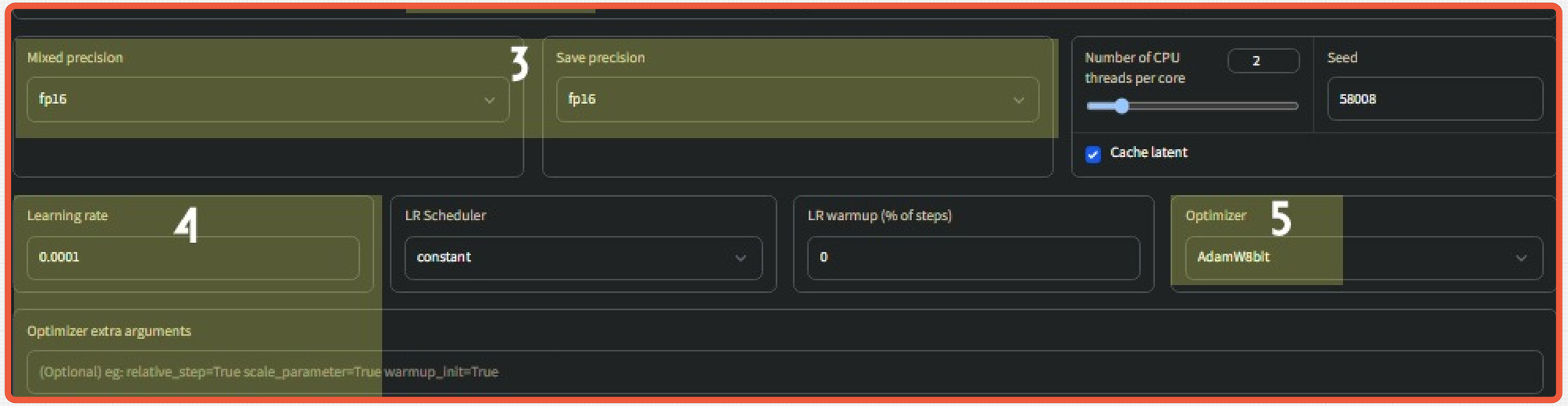
The model you want might not be the final model, but one of the earlier ones.

If I have 986 steps an epoch, 3 epochs, and set my Save every n epoch to 1, I will get 3 models trained on 986, 1972 and 2958 steps.

3. Mixed Precision / Save Precision

Mixed Precision: bf16 trains faster than fp16, but only works on 30XX and 40XX GPUs.

Always keep Save Precision as fp16 though.



4. Learning Rate / Text Encoder Learning Rate / Unet Learning Rate

You should adjust all three of these parameters according to your batch size. They will be your Batch size divided by 1000.

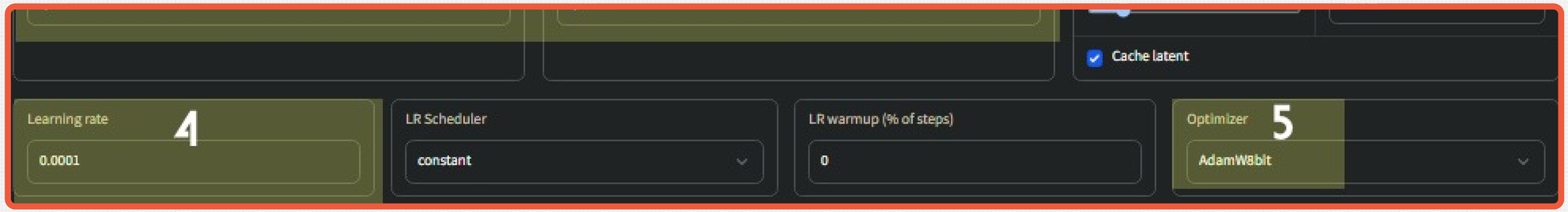
Example: Batch size of 3

Learning rate: 0.0003
Text encoder learning rate: 0.0003
Unet learning rate: 0.0003

5. Optimizer

On 10XX GPUs, you won't be able to use AdamW8bit

Instead, change this to **AdamW**



6. Network Rank (Dimension) / Network Alpha

Generally, it is best to keep Network Rank (Dimension) and Network Alpha the same.

For style LoRAs, **256 is a good number** to use for both Network Rank & Network Alpha.

Character LoRAs will need no more than 128 for both Network Rank & Network Alpha.

7. Max Resolution

If you have a high amount of VRAM, you can train at a higher resolution such as 768x768. Otherwise, leave this at 512x152

Epochs and Steps

I find that character LoRAs come out best when trained on between **1500-3000 steps.**

There are a number of ways to achieve this, but first let's understand how steps are calculated:

total steps = (number of images * repeats * epochs * regularization multiplier) / batch size

number of images is self explanatory, it's the number of training images you gathered earlier and put in a subfolder.
Repeats is the number of times our training algorithm will go over each image per epoch. We get the repeats from the number we used in the name our training image folder earlier. Our folder name was 6_lisabp woman, and 6 was the number of repeats.
Epochs is a setting you can change in the Parameters subtab. It shows how many complete passes the learning algorithm makes through the entire training dataset.
Regularization multiplier is either 1 or 2. If you aren't using regularization images, this is 1 (nothing happens). If you are using regularization images, this is 2 (steps are doubled).
Batch size is a setting you can change in the Parameters subtab. It refers to how many images are being trained at once.

What can LoRAs do?

- Quality improvements (eg. Detail Tweaker)
- Styles/aesthetics (eg. Arcane style, Studio Ghibli style, Anime Lineart)
- Characters or people (eg. Makima, Cute_girl_mix)
- Clothing or objects (eg. Hanfu, Taiwanese Foo)
- Settings (eg. School building)

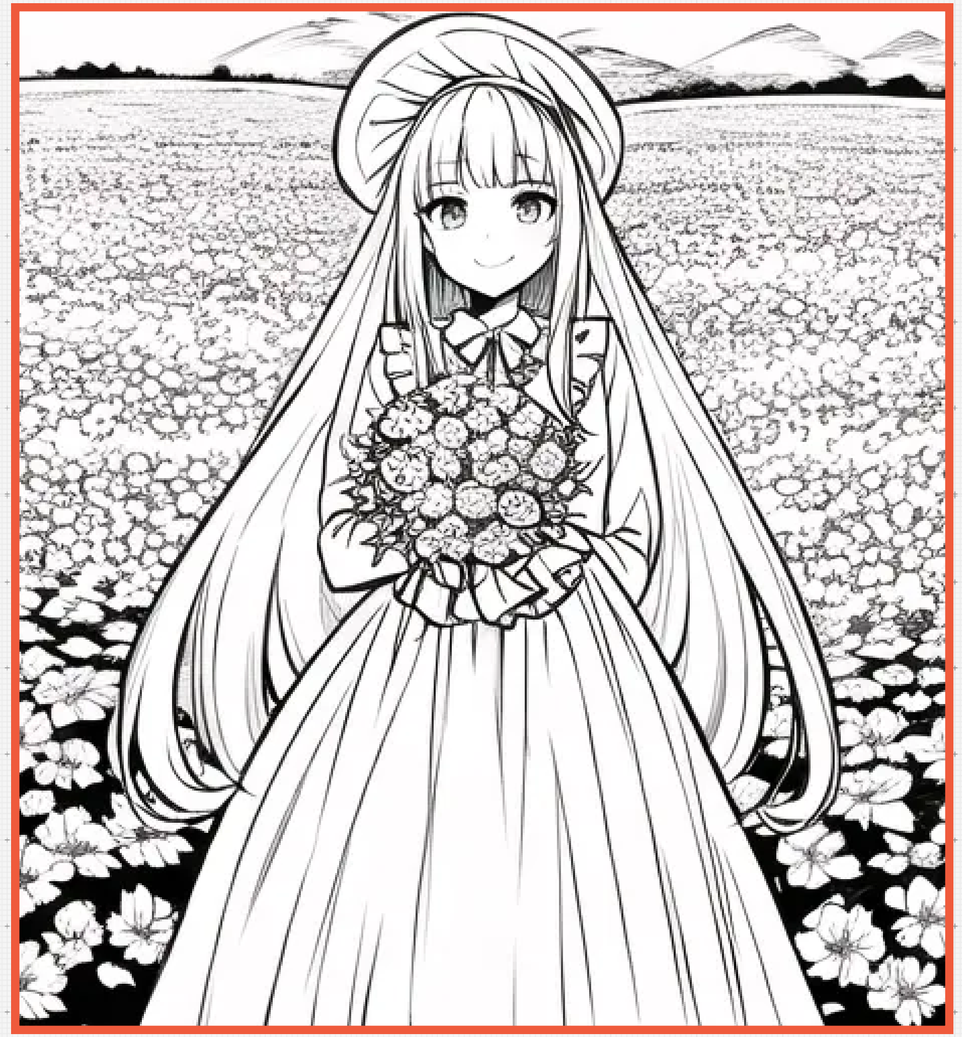
No loRA

using Colorwater (style LoRA)

No loRA

using Hanfu (clothing LoRA)

No loRA



using Anime Lineart (style LoRA)

Resources

- https://stable-diffusion-art.com/lora/
- https://softwarekeep.com/help-center/how-to-use-stable-diffusion-lora-models
- *https://www.andyhtu.com/post/image-captioning-methods*
- *https://aituts.com/stable-diffusion-lora/*
- *https://ashejunius.com/alpha-and-dimensions-two-wild-settings-of-training-lora-in-stable-diffusion-d7ad3e3a3b0a*

100xEngineers

Question and Answers