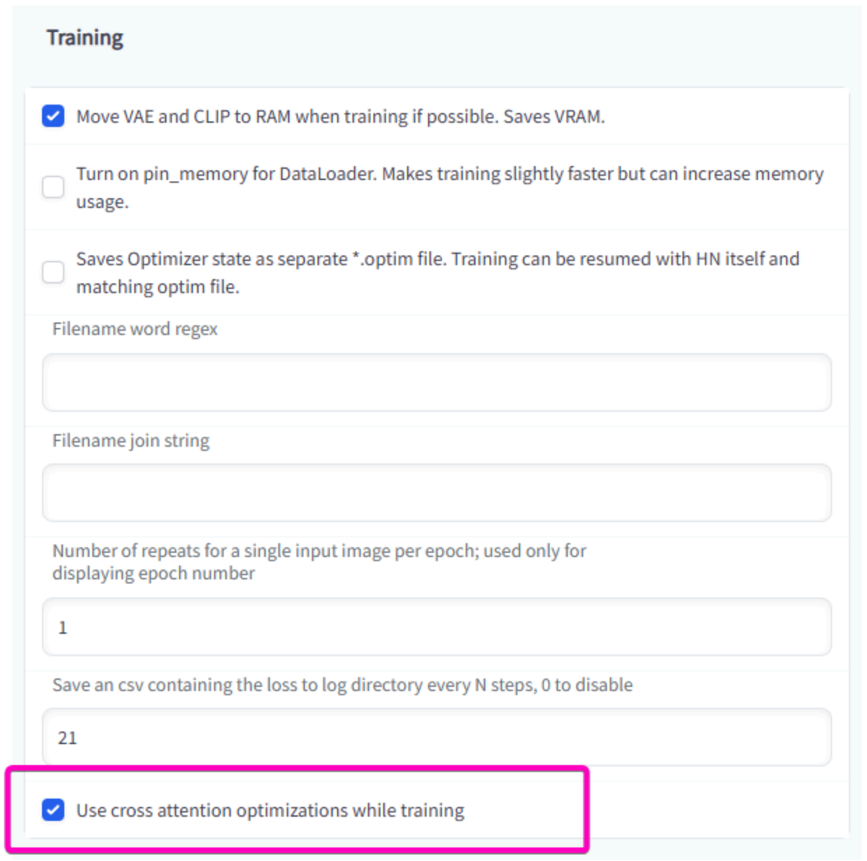
Textual Inversion - Training embeddings for Stable Diffusion 2.0+ in Automatic1111 UI

By Adam Desrosiers

This walkthrough (tentatively complete) is intended as a practical guide, not a deep theoretical dive into how and why this process works. I don't have a deep understanding of the underlying technology. I'm an artist, not a scientist, and my primary interest is in getting these cobbled-together tools to work for me, not spending hours reading papers from research scientists that are mostly going over my head. So in many ways, this guide may get things wrong. But it's workable. Following along should be a great way for you to get started. Ideally, someone with both a theoretical and a practical understanding of this process would write this guide, but I didn't see any such approachable, easy-to-follow walkthrough, so I'm writing this one instead.

I will describe how I created the embedding 'Classipeint' which you can download from Huggingface here.

I am doing my training on a local install of Automatic1111 on a laptop with an RTX3070 video card, with 8GB of video ram. In Automatic1111 make sure that in 'settings' under the 'Training' heading, you have checked the option 'Use cross attention optimizations while training' - this improves memory efficiency. Without it, my graphics card is unable to do any training at all. ***NB:*** It is also possible to do this in a Google Colab environment running Automatic1111, but this guide does not provide instruction on setting that up.


Training

- [✓] Move VAE and CLIP to RAM when training if possible. Saves VRAM.
- [ ] Turn on pin_memory for DataLoader. Makes training slightly faster but can increase memory usage.
- [ ] Saves Optimizer state as separate *.optim file. Training can be resumed with HN itself and matching optim file.

Filename word regex

Filename join string

Number of repeats for a single input image per epoch; used only for displaying epoch number: 1

Save an csv containing the loss to log directory every N steps, 0 to disable: 21

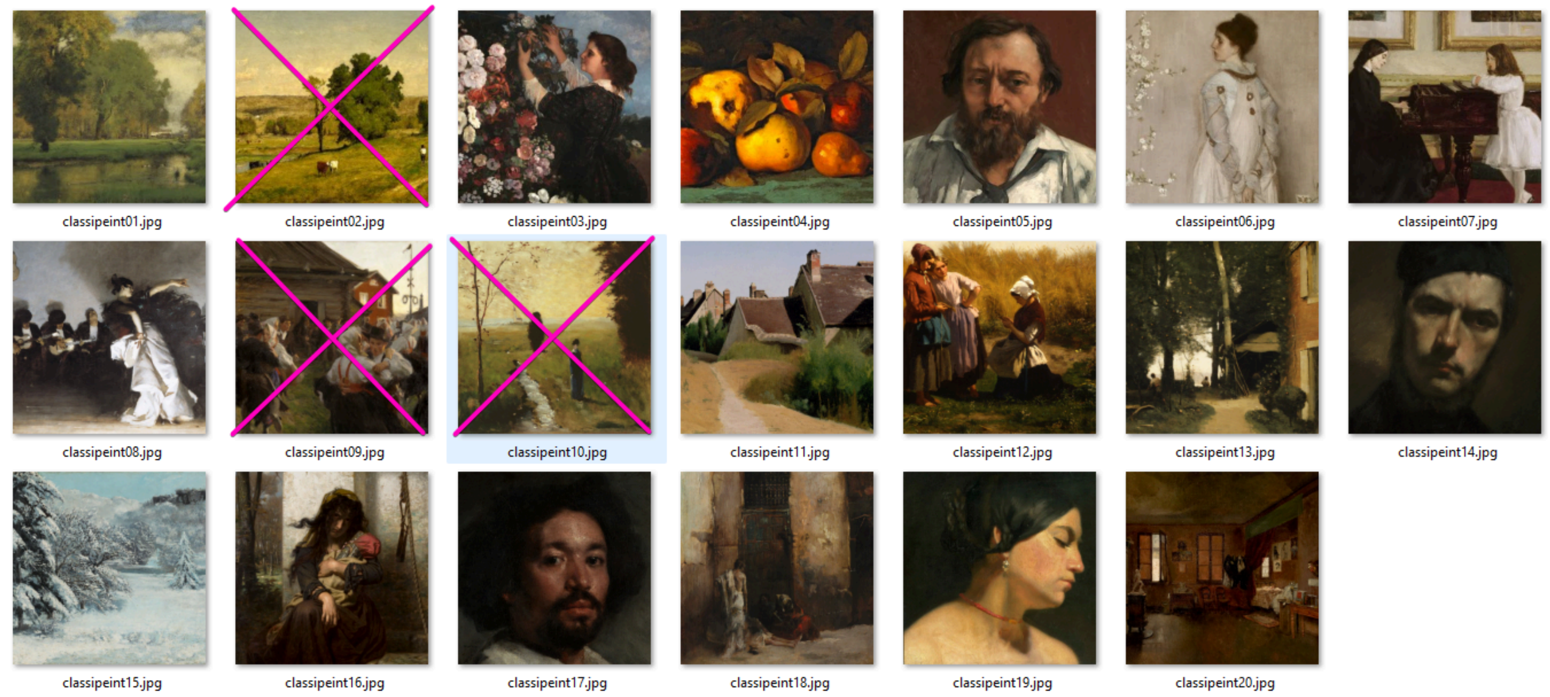
- [✓] Use cross attention optimizations while training

Image Dataset

The first step is to collect a good set of images. Here's how I went about that here.

I want real physical texture. And I want to see the glare and highlights and shadows of real-world lighting as it plays across the surface of a painting. I also want to stick with naturalistic coloring. So I started with Gustave Courbet (I kept 4 or

5 here?) then just hunted around through the Google Arts and Culture site for related work and made an initial selection seen below - three are crossed out but my initial training attempts included them.
I tried training with this set a few times but I kept seeing that the creamsicle orange glow was too prevalent in output images, and the impressionist mark-making of one of the landscapes just took over everything. And the image with all the villagers dancing or milling about, seemed to push too many scenes to crowding images with lots of people (something my first embedding laxpaint does a ton and I was determined to avoid it this second go-round).



With my reduced image set, I started to get pretty good styles. Because it was an odd number of images at 17, I set my batches to 1, and the gradient accumulation to the full set of images. I'll get back to those numbers later and explain what to do. But let's take a little tangent to discuss your image dataset and number of images, as understanding it now can save you a headache later:

Side note: understanding the numbers for your dataset

You want to know the basic capability of your graphic card - how many batches it can train on. That is, how many images can your card load into video ram at one time. The higher the number, the more efficient and quickly your process will go. So, the first time you are doing any TI embedding, you may want to quickly throw together any collection of images and get to the process (outlined below) where you actually start a training, and test the different batch sizes to find out where you get a memory error.

You cannot necessarily just use the largest batch size your memory card can handle, because your batch size and the gradient accumulation steps need to correlate evenly with your total number of images in your dataset.

As also defined below, for when you are starting the training:
Batch Size refers to the amount of images your GPU can load into VRAM at the same time. More VRAM and you can have a higher batch. At 8GB, my card allows a maximum batch of 3.
Gradient Accumulation steps refers to how many batches will process before restarting for a new round of training.

So, there exists a basic math correlation between these two numbers and the total number of images you're training from. Gradient Accumulation Steps should be a number that, multiplied by your batch, equals the total number of training images. So, if I have 33 images I can have a batch of 3 images, with gradient of 11, which means that after one cycle of training we will have trained from all images of my data set. If I did a batch size of 2, then my gradient accumulation should be 15 for a total number of 30 images (I would cull out 3). And if I have a total number of images that's a prime number like 17? I should do batch size of 1 and set a gradient accumulation equal to all of my 17 images. It's slower to do a batch size of 1 but it works fine.

This is important to keep in mind as you prepare your image dataset. Know your potential batch sizes, and do the arithmetic so that you can have an even equation of batch size x gradient accumulation = total images in dataset

But the problem I then saw with this first image set was that somehow Stable Diffusion recognized that these images are all old, it seems. The few outfits are period dresses. And the buildings in Courbet's dirt road with houses are very old simple buildings. So, no matter how I might want to prompt for a contemporary subject I wasn't getting it. It was versatile enough to give me various seasons, indoor-outdoor, still-life and figures ... but not different timeframes.

I tested multiple trainings with various settings for image captions and how those are treated, but nothing was working. At this point, I've probably run 8 or 9 embedding runs over the course of a weekend. This represents a ton of computing time - my laptop has not melted yet. So how do I approach this differently? I just need a better image data set.

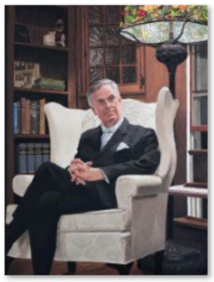
So what do we need to do to get an image set that will train not just the texture I want, but also all of the flexibility I want? Naturally, we need a contemporary painter's work. Especially someone who likes painting the modern world, like cars and telephone poles. Hey, I know who that is - that's me! Some of my photos (now part of this embedding) are below. I'm not up to the level of a master like Courbet or Velazques, but I hoped we had *enough* in common and that I'd not drag down the painterly brilliance of these old masters and just insert some modern subject matter.

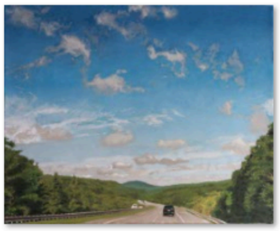
caleb.jpg

CRW_4409.jpg

CRW_4551.jpg

CRW_4559.jpg


CRW_5135.jpg


CRW_5163.jpg

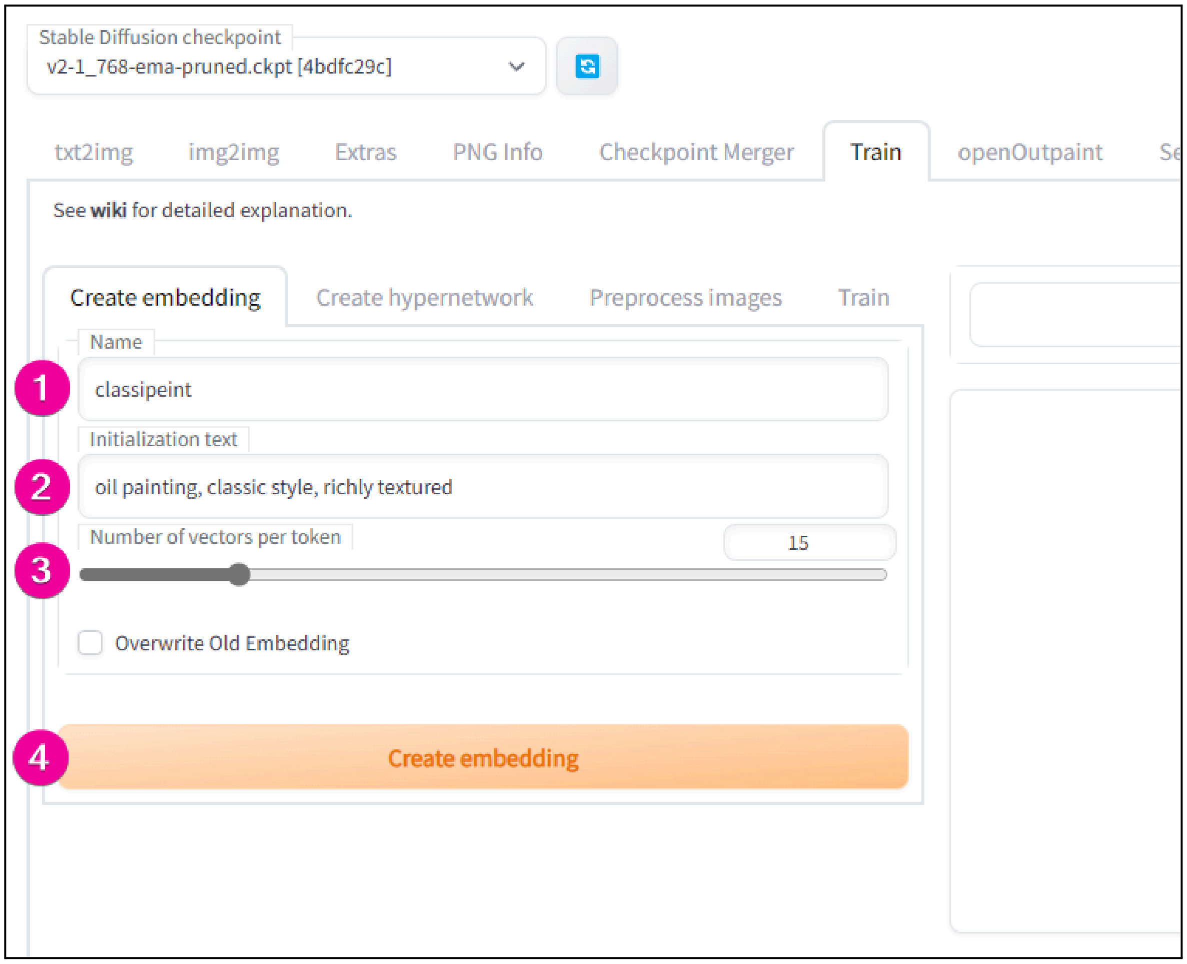
CRW_5466.jpg

CRW_5559.jpg

I did add a few other samples by contemporary artists, but I won't mention who because, you know, the pitchforks and torches crowd. My choices were not about copying style so much as they were about getting a little more racial diversity and finding a couple high-resolution photos with extreme lighting on paint texture. Also, these artists are not struggling, and I feel perfectly fine that they will continue to live in the lap of luxury painting portraits of presidents and bishops. I think anyhow my own contributions are likely more impactful - so many contemporary painters hate depicting ugly roads and cars and telephone poles and that was primarily what needed expanding on and my paintings were that contribution.

Create the embedding

So now you've got your images together and you're ready to train. Probably multiple times. Hopefully, one of those times will be where you wanted to go. But maybe not.



Stable Diffusion checkpoint: v2-1_768-ema-pruned.ckpt [4bdfc29c]

txt2img | img2img | Extras | PNG Info | Checkpoint Merger | Train | openOutpaint | Se

See **wiki** for detailed explanation.

Create embedding | Create hypernetwork | Preprocess images | Train

1 Name: classipeint

2 Initialization text: oil painting, classic style, richly textured

3 Number of vectors per token: 15

Overwrite Old Embedding

4 Create embedding

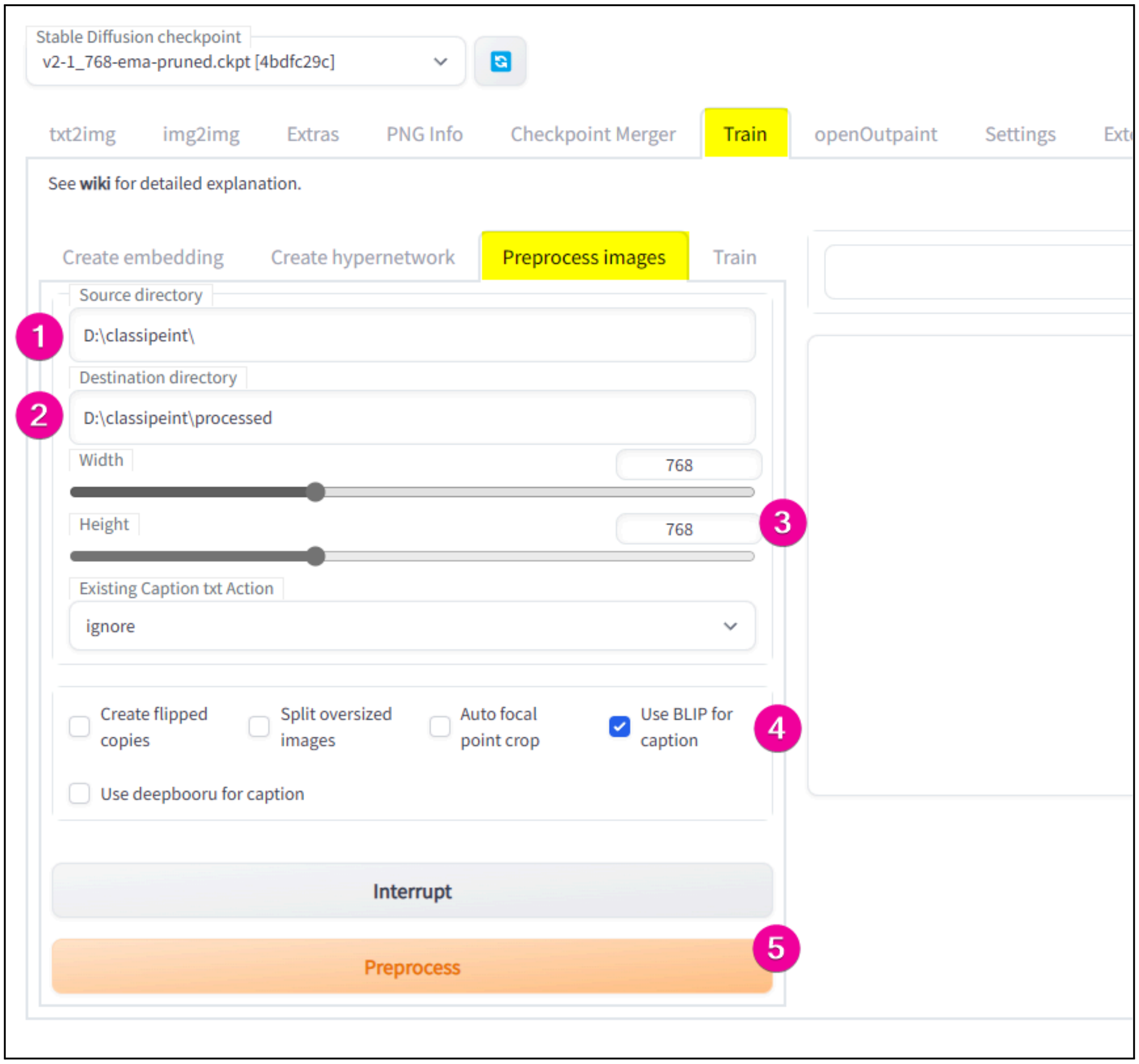
On the first section of the 'Train' tab

1. **Name:** you need to select a name. This will become the filename. The filename can later be changed to whatever you like. The crucial thing here is that you use some kind of term that you're pretty sure won't interfere with how the model was trained. Like, don't name it "MonaLisa" - cuz Stable Diffusion knows what Mona Lisa is already.
2. **Initialization Text:**
   For classipeint, I left just the initial asterisk, which worked. However, better results are likely to be had when you provide an initialization text. A smart initialization text helps guide the clear targeting of your embedding. Be concise, and direct. Leaving it an asterisk is kind of asking the process to read your mind. So if I train a similar painting style, I would choose an initialization text such as “oil painting, classic style, richly textured” - this should guide the process toward those concepts in Stable Diffusion’s latent space. Maybe there are better terms, so you may need to try this multiple times.
3. **Number of vectors per token.** The amount of vectors required can loosely correlate to how complex your initialization text is. If it takes a lot to describe the style you wish to capture, you will need more vectors per token. A good starting point is generally 8, but you may wish to increase that for particularly rich, complex styles or reduce it for simple ones. With more vectors you will be better able to duplicate the style, but may find it less flexible. You also lose potential tokens for your prompt - the resultant embedding essentially represents a complex prompt concept, so if you assign 50 vectors to your embedding, not only will it be rather inflexible, but you will only be able to add a small handful of terms to what you want to prompt for. For this embedding I used 15 (going off of nothing more than random guesswork at the time). Whether it needed that much? Only way to know for sure would be to run the same training settings but changing only this setting, which I haven’t tried doing.

I *have* subsequently re-trained another embedding and, working from this concept of loosely correlating the complexity of my initialization text with the number of vectors, I felt I had a better grasp of the process and could anticipate results a little more.
Much of this still doesn't feel like a hard science or clear-cut equation. You gotta just be comfortable with some guesswork and, in the process of trying/failing multiple times, hopefully gain some intuition for what will work better as you refine your next attempt.

4. Click on **'Create embedding'**. You will now have a .pt file in the folder 'embeddings' within your Automatic1111 installation. The next steps will modify that file. But we will also be generating a handful of other files that capture different stages of the training process. These are crucial.

Preprocess images



Stable Diffusion checkpoint: v2-1_768-ema-pruned.ckpt [4bdfc29c]

Source directory: D:\classipeint\

Destination directory: D:\classipeint\processed

Width: 768

Height: 768

Existing Caption txt Action: ignore

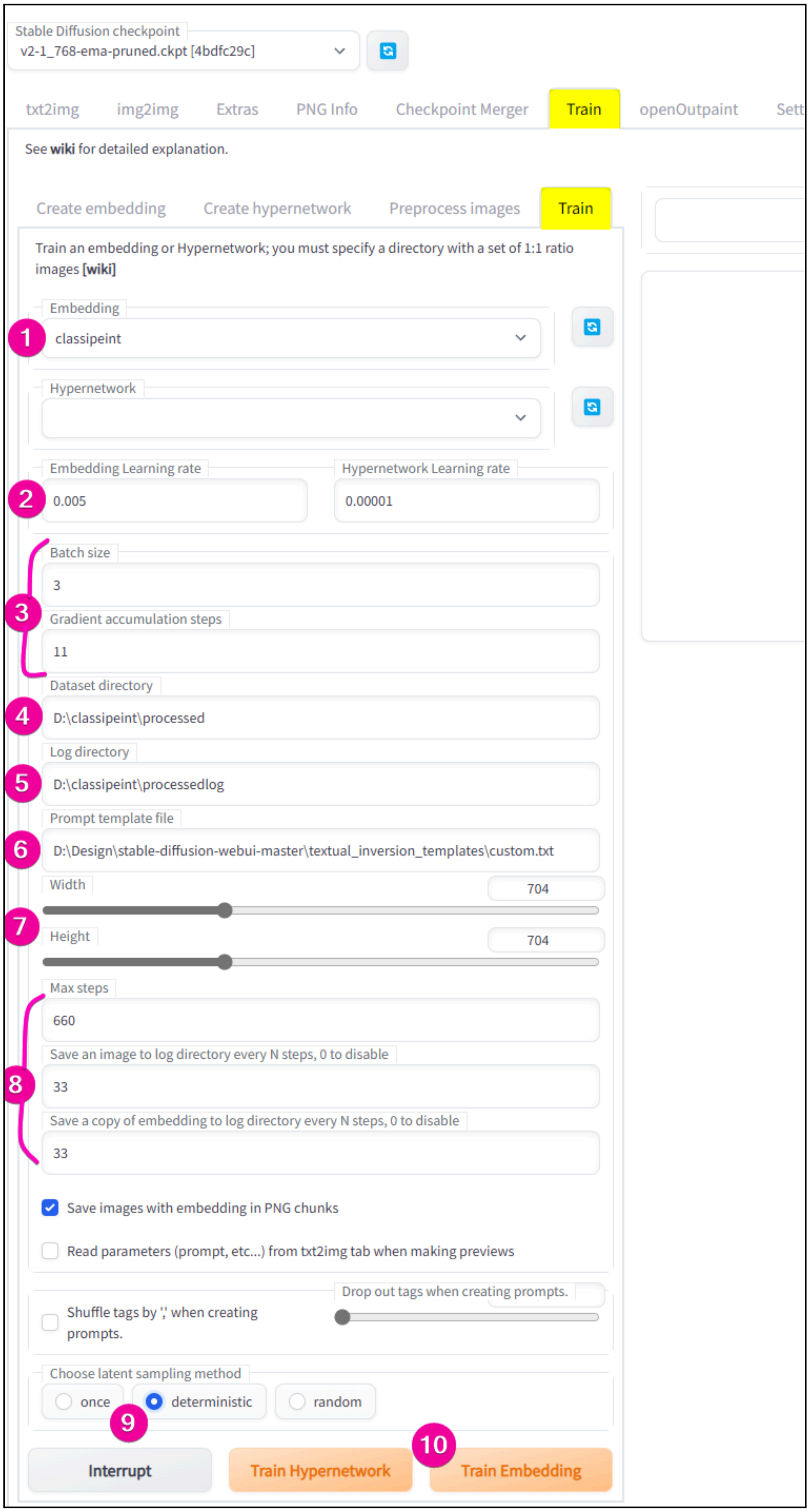
- [ ] Create flipped copies
- [ ] Split oversized images
- [ ] Auto focal point crop
- [x] Use BLIP for caption
- [ ] Use deepbooru for caption

So In the second section of the 'Train' tab 'Preprocess images' you will need to tell Automatic where your images are. Then you are going to ask Automatic to 'preprocess them'. I recommend manually cropping your images before you get to this point. So you have full control over that. But I still go through this step in order to generate the accompanying text files where you will write your image captions.

1. **Source Directory**: This is the folder on your computer where your initial collection of photos are. As I recommended above, they should already be square crops. This process can size them down to 768x768 if you didn't do that when cropping. Be sure your source images are at least as big as this – you will get blurry imagery by using smaller images.

2. **Destination Directory**: After clicking on the ‘Preprocess’ button, Automatic will size, crop, and add alongside your images a text file with a brief description of the image. This field defines where you want that to be. You will need to know this folder for the last Train tab too – it’s where you tell Automatic “Train the embedding I created on these images”
3. **Use BLIP for caption**: with this selected, Automatic will generate a text file next to each image. Go through each one, edit them to make sure they’re coherent, and make them succinctly but accurately describe the image.
   The crucial concept to keep in mind with these captions is that, at least in theory, you want to use terms that describe what is *incidental* to what you are trying to train. In this case, I am trying to train a painterly style. So the *fundamental* concepts “painterly, classic, oil paint” and all the terms of the initialization text, need to be avoided. Instead, simply describe the content. “Two girls in period dresses. One plays the piano while the other leans on the piano. The piano is up against a wall which has multiple framed artworks hanging on it” Something like that. It should then be that the Textual Inversion process picks apart the fundamental element (the style described in the initialization text) from the incidental elements (the represented imagery and illustrated themes) of the source photos. This is not a perfect process, it is merely a guide. You can caption ‘blue sky’ hoping that you don’t train blue skies, but you still will see the blue skies show up to some extent in your style if it’s there in your dataset.
4. **Preprocess** – you gotta click the button! Then go into that folder and edit your image captions.

The Training Process



Stable Diffusion checkpoint: v2-1_768-ema-pruned.ckpt [4bdfc29c]

See **wiki** for detailed explanation.

Train an embedding or Hypernetwork; you must specify a directory with a set of 1:1 ratio images **[wiki]**

1 Embedding: classipeint

Hypernetwork:

2 Embedding Learning rate: 0.005

Hypernetwork Learning rate: 0.00001

3 Batch size: 3

Gradient accumulation steps: 11

4 Dataset directory: D:\classipeint\processed

5 Log directory: D:\classipeint\processedlog

6 Prompt template file: D:\Design\stable-diffusion-webui-master\textual_inversion_templates\custom.txt

7 Width: 704

Height: 704

8 Max steps: 660

Save an image to log directory every N steps, 0 to disable: 33

Save a copy of embedding to log directory every N steps, 0 to disable: 33

[✓] Save images with embedding in PNG chunks

[ ] Read parameters (prompt, etc...) from txt2img tab when making previews

[ ] Shuffle tags by ',' when creating prompts.

Drop out tags when creating prompts.

Choose latent sampling method: once [✓ deterministic] random

9

And onto the last section of the Train tab – Train.

1. **Embedding** – it's a dropdown list of all the embeddings Automatic recognizes as installed in the appropriate folder for this GUI. Find and select the name of the embedding you are going to train. If you had a bad result from a previous attempt, delete this file, go back to the 'Create Embedding' section, and generate it anew.
2. **Embedding Learning Rate** - magic happens here. Try 0.005 or 0.004 … or something else? Who knows! Some people say you should use different training rates at different stages of the process … others say that's contrary to the way the magic that formed Stable Diffusion works! Go with your gut.
3. **Batch Size & Gradient Accumulation Steps** –
   a. **Batch Size** refers to the amount of images your GPU can load into VRAM at the same time. More VRAM and you can have a higher batch. With 8GB VRAM, my card allows a maximum batch of 3.
   b. **Gradient Accumulation steps** refers to how many batches will process before starting a new round of training.
   There is a basic math correlation between these two numbers and the total number of images you're training from. Gradient Accumulation Steps should be a number that, multiplied by your batch, equals the total number of training images. So, I have 33 images. Batched for 3 images at a time, with gradient of 11, means that after one cycle of training we will have trained from all images of my data set. If I did a batch size of 2, then my gradient accumulation should be 15 for a total number of 30 images (I would cull out 3). And if I have a total number of images that's a prime number like 17? I should do batch size of 1 and Gradient accumulation equal to all of my 17 images. It's slower to do a batch size of 1 but it works fine.
4. **Dataset Directory** – this is the folder where your pre-processed images are, including their new caption text documents. You did carefully write all those captions according to how the spirit of TI moved you, right?
5. **Log Directory** – this can be anywhere on your hard drive, but know where it is. This is the location where Automatic will put the various stages of the training process as well as the image outputs that give a quick glimpse of how the embedding is proceeding (it uses one of the captions of your source images and adds 'by <token>' so it is a quick output of what the TI process has helped Stable Diffusion learn so far. You will go back to this folder when training is done.
   Alternatively, you can check the option "Read parameters from text2img tab" and you can craft there your own prompt. The output images will then use the prompt (make sure it includes the proper embedding name!), size, sampler etc. I am leaning toward using this more as it gives me a better sense of what the embedding looks like when it's not just a square (which I never generate) with the Euler sampler (which I never use).
6. **Prompt Template file** – The default setting here refers to a text file called style_filewords.txt. I used this default for the classipeint embedding. Since I created this embedding I have moved to using instead a custom template file. The style_filewords.txt did work for Classipeint, but wasn't getting me great results for my next (rather weirder, more abstract) embedding. What you may want to consider is creating a new text file in the Automatic1111 textual_inversion_templates folder and writing this simple template file:
   [filewords], [name]
   Save it something like "custom.txt" and then in this 'Prompt Template file' field you use that 'custom.txt'. You may also try the default 'style_filewords.txt' or 'style.txt' but at least for now I will be doing future training of styles with that custom file.
7. **Width & Height** – I'm unclear on this setting. I think it might only relate to the size of images being output as defined by the 'save image to log' setting. But there was one training session where I accidentally set this lower than the default 768x768 and that was the training run that worked … so was it related to this setting? Probably not, but since then I've been setting this lower. Pretty sure it's unnecessary to get this right. The images produced in the 'save image to log step are anyhow using, I think, the Euler sampler which I don't make much use of in SD2 so the images created serve as only a poor snapshot representation to me of where the training is at. I may start moving to checking the setting 'read parameters from the txt2img tab' and writing a prompt with a sampler I select at a custom size, but that's not how I ran my first few embeddings.
8. **Max Steps, Save image to log, and Save copy of embedding:** these are all related. You want to keep your numbers a factor of your total dataset – the images training from. In this case, it's 33 images. So I want to save an image to a log directory (that snapshot of where training is right now) every full training cycle. I want to also keep a copy of the embedding after each. And set a max number that suits your patience level. You can make it much lower than 660. Make it 330 … just whatever it is, make it a factor of your dataset. 33x20=660. If you didn't train enough, you can resume at any time. For class, I interrupted the process at 330

steps and started testing the different embedding points. In the log folder, the filename is appended a number identifying its steps. So I had files that look like classipeint-330.pt and classipeint-297.pt. I assumed at first I hadn't trained enough. I was trying to get my embedding to make a nice painting of a goblin, and they were coming out wretched, though it would do a human portrait fine. But just to be thorough, I went through each version and then going back really quite early, like step 165 I think, I found the embedding I'm now using. I had hit my target early – the training rate, and dataset, and vector amount and this low number of training steps, all add up to exactly what I wanted.

9. **Choose Latent Sampling Method:** Choose deterministic. This is a faster method compared to previous methods, or to the 'random' method from the original research paper, with negligible quality difference.
10. **Train Embedding:** Now you let your computer do its work - push the big orange button and go do something else for a bit!

Analyzing results, and making adjustments

As mentioned in Step 8 of the Training Process, you should now have a lot of versions of your embedding created in your log folder. You will also have a version in your main 'embeddings' folder that can be used according to the filename you initially gave it in Automatic1111.

Testing for variety of subjects

I start testing with the fully trained model, running a number of prompts as I normally would. And you want to run prompts that represent how you want it ideally to behave across various subject matter. For instance, if you are creating an embedding based on logo designs, you will run a bunch of prompts for varied kinds of logos.

In this instance, because it's a painting style, I expect it to be a flexible style that can be applied to all kinds of subject matter, from still life, to fantasy imagery, to science fiction, to portraiture, and landscape. It's also important to me that when I prompt for people, it's not going to show just white faces, or Asian faces etc. I want to be able to represent basically any time and place and subject. So I run like 5 different prompts to see if the embedding will depict a modern city, a still life, a fantasy creature, a landscape, and a sci-fi space scene.

Does it actually depict these things? Do I need to wrestle with it to show something and how much? For instance, it may have a tendency to show old period architecture or clothing. Does just a simple prompt for 'contemporary' balance that? Then things are fine. But if I need 5 positive prompts and another 5 negative prompts to overcome its tendency, then I consider this training a failure.

Failure to depict what you want means, to me, that you haven't sufficiently represented your subjects in your initial dataset. You will probably need to find more examples and include them in the dataset. Delete your embedding from the main Automatic1111 embeddings folder (you still have all those copies in the log folder from the training process) and re-generate your embedding up at the 'create the embedding' stage.

Testing for the style you're aiming for / learning rate and number of training steps

This will naturally be something you're looking at with the other assessment (about versatility) above. So you may be doing two kinds of adjustments at the same time, or, if you're lucky, just one.
As you run your test prompts, how much does it reflect the style you wanted to capture? How does it fail?

- If it doesn't look like what you are confident is the consistent thread connecting images in your dataset, you should look primarily at your initialization text and at the captions you wrote for your individual images. Please refer to those issues above - you may have identified the wrong kinds of things in your captions or misdirected the TI process with a bad initialization text description.
- If there is an exaggeration of some style or color that persists across too many images and is hard to avoid with prompts, this is probably an issue with your dataset. Look at your test image results, identify what is too frequently popping up, and find in your dataset where those things exist prominently. You may need to remove or replace those images.
    - I experienced this in another embedding primarily with seeing what look like tree branches or drippy stalactites everywhere. Basically because just 1 image in the dataset had that imagery strongly and somehow that one image just asserted itself over others. As I mentioned above in discussing this

embedding's dataset I also had to cull images because of such over-representation of coloring, mark making, and the display of crowds of people.

- Is the style "fully baked"? This assessment takes imagination. What is the style you are trying to distill, and what does it look like when it's fully extracted from your dataset? Is it all there? For my embedding here I'm asking, does it consistently show naturalistic colors? Do I see brush strokes? Do I get that built-up paint, especially in highlights? All-in-all, can you put some of these images before me and fool me that it's a photograph of a real-world painting?
  If the answer is yes, you may be nearly done (if it passed the other assessments above). But what if it's not all there? You know the dataset represents some style that's just not being represented. You need to do one of two things: train longer or increase the learning rate and start over.
  - Train longer - default to this. If you haven't trained a really long time (I think over a thousand steps is pretty long) and it seems to be getting close to what you want and it's not showing ugly artifacts, then just keep training
  - Increase training rate - If you keep training and it's always looking like it's never quite there, or you added 300 training steps and it's only marginally improved, it may never get to the target. That's when you start over with a higher training rate
- So, it's fully baked. But, is it "overbaked"? Are the edges looking a little burnt? Lines drawn too harshly? You may have overtrained it. But, if you're lucky, you could still have a winner in one of the early stages of your embedding. Go into your log, and grab a selection of embedding files representing different stages of the training process, and copy them into your Automatic1111 embeddings folder. Run test prompts with these, going back to earlier stages. You will see the point at which you started overbaking it and if, right before that, you see your style fully represented but without the harsh artifacts, then you have a winner. But what if you see the artifacts are gone, but you no longer have all the style represented? Then you need to drop your learning rate and start over. You trained too fast and by the time you reached a full representation of your style you also "overbaked" it.

Thanks for reading! Obviously, you don't owe me anything. But if this guide was helpful to you and you're just overflowing with gratitude, I'd happily accept a donation. Feel free to **buy me a coffee** 🙂

And if you read this walkthrough and thought to yourself, "boy, I sure wish there was more of his writing that I could get my hands on" well I've got that for you in my first published work, "Willem & Ellene". It's a short but dense story that discusses ideas ranging from physics, metaphysics, psychology, and aesthetics.
Buy it at Amazon here for Kindle or paperback.