Character Consistency Engineering — PortraitMaster Method

Thank you for downloading and using the \*PortraitMaster\* workflow.  
  
If you're reading this, it means you've already completed the first step: generating your talent's base image. How did it go? Tough? At least now you’re a master of that workflow. And if you’ve reached this document in desperation because you’re stuck on a small trick, try using the `PortraitMaster\_Flux1\_Lite` version — fewer steps, but still delivers solid quality.  
  
This documentation is focused on building consistent character images to support the creation of high-quality LoRA models for use with Flux1. You won’t find general tutorials on ComfyUI here — this is a targeted technical guide for creators who want reliable character consistency across generations.  
  
PortraitMaster\_BatchGen is built on top of the Generative Workflow Suite 1.4.1.1, specifically variant `Workflow 1.4.1\_A – AI Image Automation from Single Reference`. This system enables fully automated image generation from a single image and a structured .txt prompt file — one line per image — leveraging Flux1’s stable diffusion backbone and the T5XXL + CLIP\_L dual-conditioning pipeline.  
  
It is optimized for consistent face retention, pose variability, and batch productivity.  
  
What we’ll cover in this document:  
- How to use `PortraitMaster\_BatchGen.json` to generate varied pose images  
- How to generate your first LoRA  
- How to produce 100–120 support images for final LoRA training  
- How to finalize a custom LoRA for your talent or influencer

# 1. PortraitMaster\_BatchGen.json — Generating Consistent Portraits

The first step of the process is to generate at least 5–10 portrait images of your talent, varying slightly in pose and angle to create a coherent starter set.  
  
To do this, use the `PortraitMaster\_BatchGen.json` workflow together with Gemini Flash APIs and the dedicated GPT agent: https://chatgpt.com/g/g-67f099ab7b64819187c8f0c3312ff3bb-character-prompt-catalogue.  
  
This tool allows you to upload the initial image and automatically generate up to 70 structured prompts, which can be integrated into the workflow to mass-produce coherent close-up portraits with high character consistency.  
  
This step is also crucial if you plan to train a custom LoRA based on your model or influencer.

# 2. How to Generate Your First LoRA

Once you’ve generated your initial image set, the next step is creating descriptive captions for each image. These captions will be used as paired text inputs during LoRA training. There are several ways to generate them:  
  
- Use a ComfyUI batch with captioning modules like `JoyCaption` or similar tools  
- Clone a Hugging Face captioning repository (e.g., BLIP or GIT) and run it locally in a Python environment  
  
For consistency, it's recommended to organize your dataset in a structured folder such as:  
`1\_concept\_Nohea01/`  
Where "Nohea01" is the trigger word of the model. Each image file (`1.jpg`, `2.jpg`, ...) must be accompanied by its respective caption file (`1.txt`, `2.txt`, ...).  
  
Captions should be written in full descriptive form, for example:  
  
This photograph features Nohea01, a young woman with straight, platinum blonde hair that falls to her shoulders. She has fair skin, light blue eyes, and subtle, natural makeup. Her expression is neutral, with slightly parted lips. She is wearing a white, ribbed, high-neck sweater that contrasts with her pale complexion. The background is a plain, light gray wall, creating a minimalist and clean setting that focuses attention on Nohea01. The lighting is soft and even, highlighting her features and giving the image a gentle, serene atmosphere. The overall style is modern and minimalistic, with an emphasis on simplicity and natural beauty.  
  
⚠️ Based on testing, verbose descriptors like the above perform significantly better than minimal keyword-based formats typically suggested in other guides.  
  
For the first LoRA training phase, select only 5 to 7 of the best images — preferably portraits that are square-cropped, consistent in resolution, lighting, and camera angle. This visual uniformity improves alignment during the training process and ensures better identity retention in future image generations.  
  
Inside the ZIP, you will find a preconfigured Kohya\_ss training file: `kohya\_ss 1 Step.json`, already set for 200 epochs with 1x repetition.  
  
If you haven’t installed `kohya\_ss` yet, refer to the official guide:  
🔗 https://github.com/bmaltais/kohya\_ss  
  
\*\*About Kohya\_ss\*\*  
Kohya\_ss is a high-performance LoRA training framework based on PyTorch and Diffusers. It supports advanced training parameters, memory-efficient optimizations (such as 8-bit AdamW and Gradient Checkpointing), and flexible dataset augmentation. It is compatible with most SD 1.5 LoRA training pipelines and highly customizable through JSON configs.

# 3. How to Produce 100–120 Support Images for Final LoRA Training

Now that you’ve trained your first LoRA, it might already be sufficient for basic character consistency. However, if your goal is to reach a production-grade level of fidelity and versatility, you’ll need to generate a larger support dataset — ideally 80 to 120 images.  
  
As mentioned earlier, a high-quality and diverse dataset — in terms of pose, framing, camera distance, lighting, and expression — can greatly enhance your LoRA’s generalization power.  
  
\*\*There are two primary approaches to build this dataset:\*\*  
  
1. \*\*Manual pose generation using any Flux1-based workflow.\*\*  
 You can create controlled variations with different prompts or use advanced methods like ControlNet with OpenPose. While this provides granular control, it requires careful tuning and can be time-consuming.  
  
2. \*\*Use the GPT-powered automation tool:\*\*  
 If you already own the generator available at 👉 https://sergiovalsecchi.gumroad.com/l/uzjjo, you can use it to produce prompt sets for poses, outfits, expressions, and compositions. Feed these into your Flux1-based workflow, and in the final upscale step, insert the LoRA you just trained.  
  
 This method allows you to:  
 - Generate low-resolution images using Gemini Flash or Kontext modules  
 - Apply the trained LoRA in the last upscale step for consistency enforcement  
 - Automatically include your trigger word in the prompt for accurate referencing  
  
💡 If you don’t own the tool yet, contact me for a 50% discount coupon.  
  
Once this phase is complete, you’ll have a polished dataset suitable for final LoRA training with improved stability and coverage.

# 4. How to Finalize a Custom LoRA for Your Talent or Influencer

With your 100–120 image dataset ready, you’re now prepared to perform the final training step for your custom LoRA.  
  
⚠️ It’s critical to avoid overfitting. LoRA models trained with 1,000–1,600 total training steps usually provide an optimal balance between detail preservation and generalization.  
  
If your dataset includes, for example:  
- 40 face-focused images  
- 80 full-body or dynamic pose images  
  
You can organize them into two folders with different repeat factors:  
- `40\_face\_nohea01` → 40 images, repeated 40 times  
- `10\_pose\_nohea01` → 80 images, repeated 10 times  
  
The higher repeat value for face images prioritizes facial identity retention over general pose variation, which is especially useful for influencers or fashion talents.  
  
How repeat factors work in Kohya\_ss:  
The folder name defines the repeat value: the prefix must contain the number of repeats followed by an underscore. For example, `40\_face\_nohea01` means every image inside will be repeated 40 times during training.  
  
Practical Tips:  
- Balance your dataset\*\*: Use higher repetition for categories with fewer images or critical identity features (like faces).  
- Use `--max\_train\_steps`\*\* to precisely control total training steps (e.g., 1,200–1,600 steps).  
- Enable `--save\_every\_n\_steps` or `--save\_every\_n\_epochs`\*\* to save checkpoints at regular intervals.

🧮 **Bucket Resolution in Kohya\_ss**

To ensure Kohya\_ss correctly processes images of various aspect ratios, make sure to enable bucket-based resolution grouping by setting the parameters:

* --bucket\_reso\_steps
* --bucket\_no\_upscale

This allows Kohya\_ss to cluster images with similar proportions together for optimized training performance.  
Typical values for --bucket\_reso\_steps are **32** or **64**, depending on the model and the average image size.

Refer to the attached table for overfitting risk levels based on step count. For example, 1,000 steps presents low risk, while 2,000+ steps may start introducing artifacts or loss of generalization.  
  
Once the training is complete, test your LoRA across different workflows to evaluate identity retention, pose flexibility, and lighting adaptability.