Stable Diffusion Ultimate Guide pt. 1: Setup

Let’s start from the basics, what is Stable Diffusion?

Have you ever wished you could turn your words into images? Imagine being able to write a description of a scene, a character, or an object and see it come to life on your screen. Sounds like magic, right?

Well, thanks to Stable Diffusion, this is now possible. Stable Diffusion is a revolutionary text-to-image diffusion model that was released to the public by Stability.ai on August 22, 2022. This cutting-edge tool allows users to input a text prompt and generate a corresponding image with incredible accuracy.

But how does it work? And what makes it different from other text-to-image models?

In this blog post, we will give you an introduction to Stable Diffusion, explain its main features and benefits, and we’ll install the 1111automatic UI.

What is Stable Diffusion?

Stable Diffusion is a system made up of several components and models. It is not one monolithic model. As we look under the hood, the first observation we can make is that there’s a text-understanding component that translates the text information into a numeric representation that captures the ideas in the text.

This numeric representation is then fed into a diffusion model that generates an image pixel by pixel. A diffusion model is a type of generative model that works by reversing the process of diffusion. Diffusion is when something spreads out from a source over time. For example, if you drop some ink into water, it will diffuse and create patterns.

A diffusion model does the opposite: it starts with noise (random pixels) and gradually removes it until it reaches an image that matches the input. This process is guided by a neural network that learns from data how to produce realistic images.

Stable Diffusion uses a state-of-the-art diffusion model called U-Diffuse, which was developed by Stability.ai. U-Diffuse stands for Unconditional Diffuse, meaning that it can generate images without any prior information or constraints. U-Diffuse can produce high-quality images at any resolution up to 1024x1024 pixels.

What makes Stable Diffusion different?

Stable Diffusion has several advantages over other text-to-image models:

- **It can handle complex and diverse prompts**: Unlike some models that only work well with simple or specific prompts, Stable Diffusion can generate images for any kind of text input. You can write anything from abstract concepts to detailed descriptions and get realistic results.
- **It can preserve consistency across multiple images**: One of the challenges of text-to-image generation is ensuring consistency between different images generated from the same prompt or related prompts. For example, if you ask for two images of “a cat wearing glasses”, you would expect them to look similar but not identical. Stable Diffusion can achieve this by using personal embeddings, which are unique identifiers for each prompt that help maintain coherence across multiple generations.
- **It can incorporate user feedback**: Another challenge of text-to-image generation is allowing users to control or modify the output according to their preferences. For example, if you don’t like how an image looks or want to change some details, you would want to have some way of telling the model what you want. Stable Diffusion allows users to do this by using textual inversion, which is a technique that lets users edit the text prompt based on the generated image and get an updated image accordingly.

How can I use Stable Diffusion?

Stable Diffusion is available as a free online tool at https://dreamstudio.ai.You can also access it through various platforms such as Discord, Twitter, Reddit, etc.

Pay attention to the fact that you are going to need an Nvidia GPU to use this tool correctly. Theoretically, you can use a CPU to generate the pictures but it’ll take a long time.

In this post, we are going to learn how to install everything we need on our machine.

AUTOMATIC1111/stable-diffusion-webui

Introduction to Stable Diffusion Web UI

Stable Diffusion Web UI is a browser interface for Stable Diffusion. It is based on Gradio library, which allows you to create interactive web interfaces for your machine learning models.

Stable Diffusion Web UI has many features that make it easy and fun to use. Some of them are:

- **Multiple Generation Modes:** other than the classic text2image mode you can also work with image2image and inpainting. By using extensions you can also unlock other important tools as Controlnets, Outpainting and much more.
- **Negative Prompt:** when you build a prompt, you have the choice to define a negative prompt that explains to the model what are the things you **don't want** in the generated picture.
- **Extensions:** you can install extensions built by the community to improve and expand an already amazing tools (we are going to take a look at some of them in future posts).
- **Hypernetworks**: You can fine tune the weights for CLIP and Unet, the language model and the image de-noiser used by Stable Diffusion, using a method called hypernetworks. Hypernetworks are generously donated to the world by our friends at Novel AI in autumn 2022. They work in the same way as Lora except for sharing weights for some layers.
- And much (much) more!

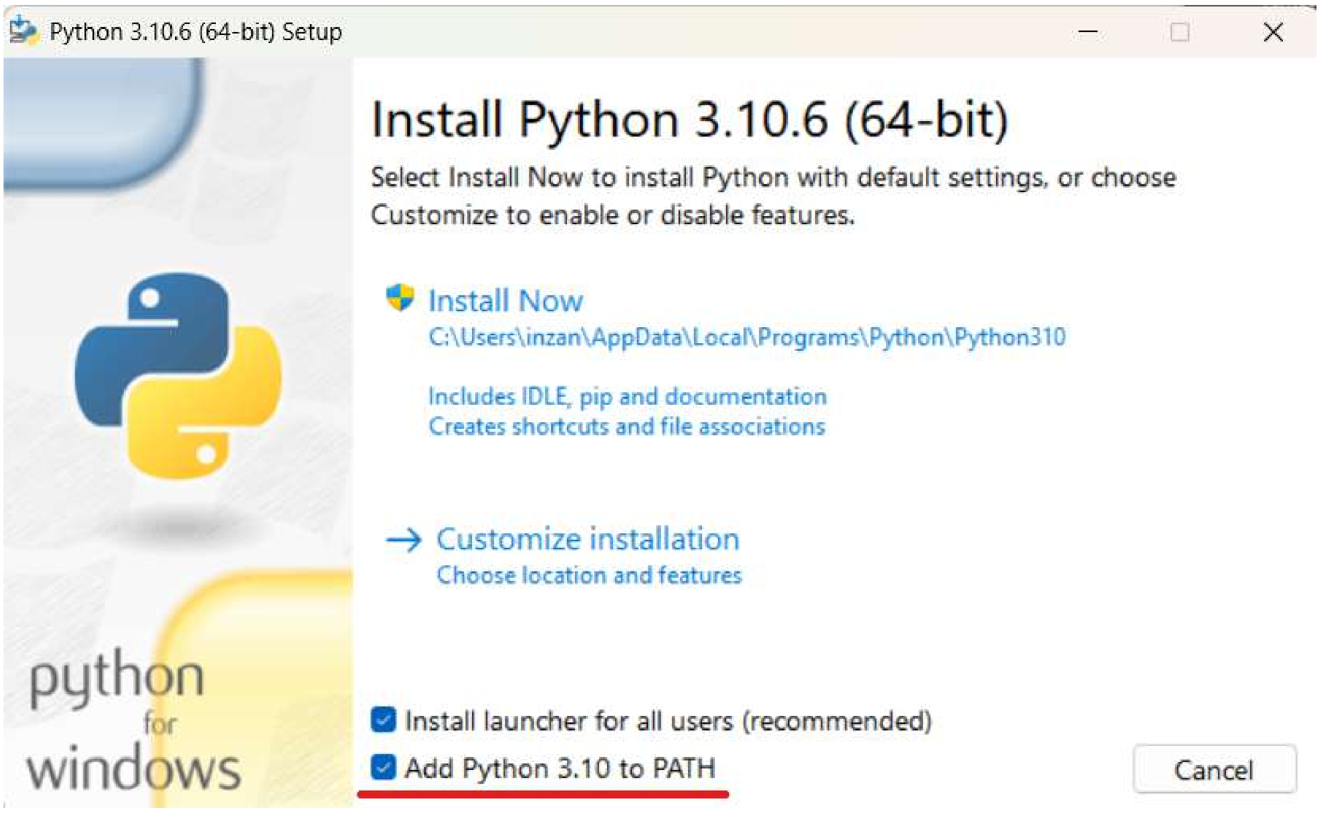
Installing the Requirements

To install the tool we need to install the requirements. Let's start by installing Python.

We need to install the version 3.10.6. If you already have Python installed on your machine you'll probably still need to install this version to make sure that everything is compatible. Go to the following link, scroll to the bottom of the page and choose the correct version (you'll probably need the Windows 64bit version):

Python Release Python 3.10.6 | Python.org

Once you've downloaded the installer, open it and **make sure to have flagged the "Add Python 3.10 to PATH"**. If you don't flag that option, the automatic1111 installer won't work later!


Python 3.10.6 (64-bit) Setup

Install Python 3.10.6 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\inzan\AppData\Local\Programs\Python\Python310

Includes IDLE, pip and documentation
Creates shortcuts and file associations

→ Customize installation
Choose location and features

[✓] Install launcher for all users (recommended)
[✓] Add Python 3.10 to PATH

Cancel

The option you need to flag

Once you’ve installed Python we can now install the second requirement: GIT.

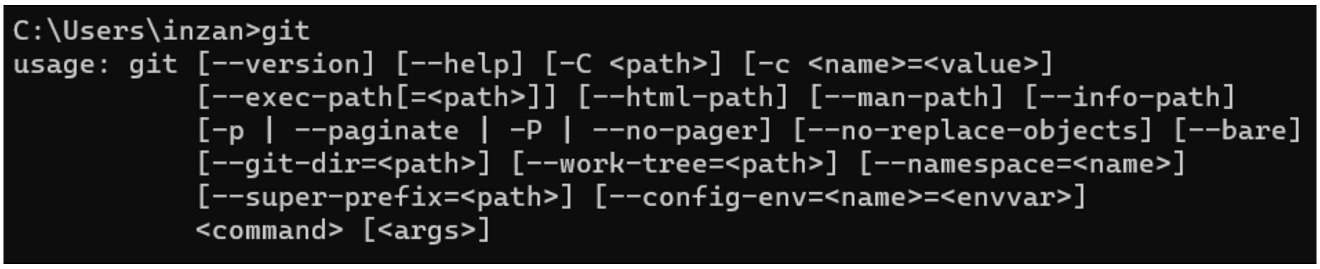
To download GIT, go to the following link and download the correct version:

Git — Downloads (git-scm.com)

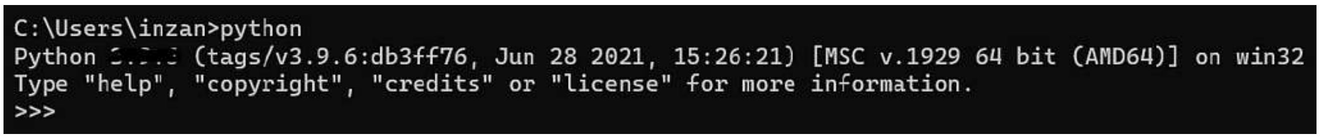
Once you’ve completed the download, open the executable and complete the installation. During the setup wizard you can leave everything as is.

Now we can check if everything is working as expected. Just open a terminal (cmd or powershell) and write:

git


```
C:\Users\inzan>git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           [--super-prefix=<path>] [--config-env=<name>=<envvar>]
           <command> [<args>]
```

python


```
C:\Users\inzan>python
Python [illegible] (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

You should see 3.10.6

If both of this commands works as expected we can now start installing the UI.

Installing the UI

GitHub — AUTOMATIC1111/stable-diffusion-webui: Stable Diffusion web UI

Go to an empty folder in which you want to install the UI, then press shift+right click and choose: “Open a new Terminal Window Here” (Or a Powershell window if you are on Win10).

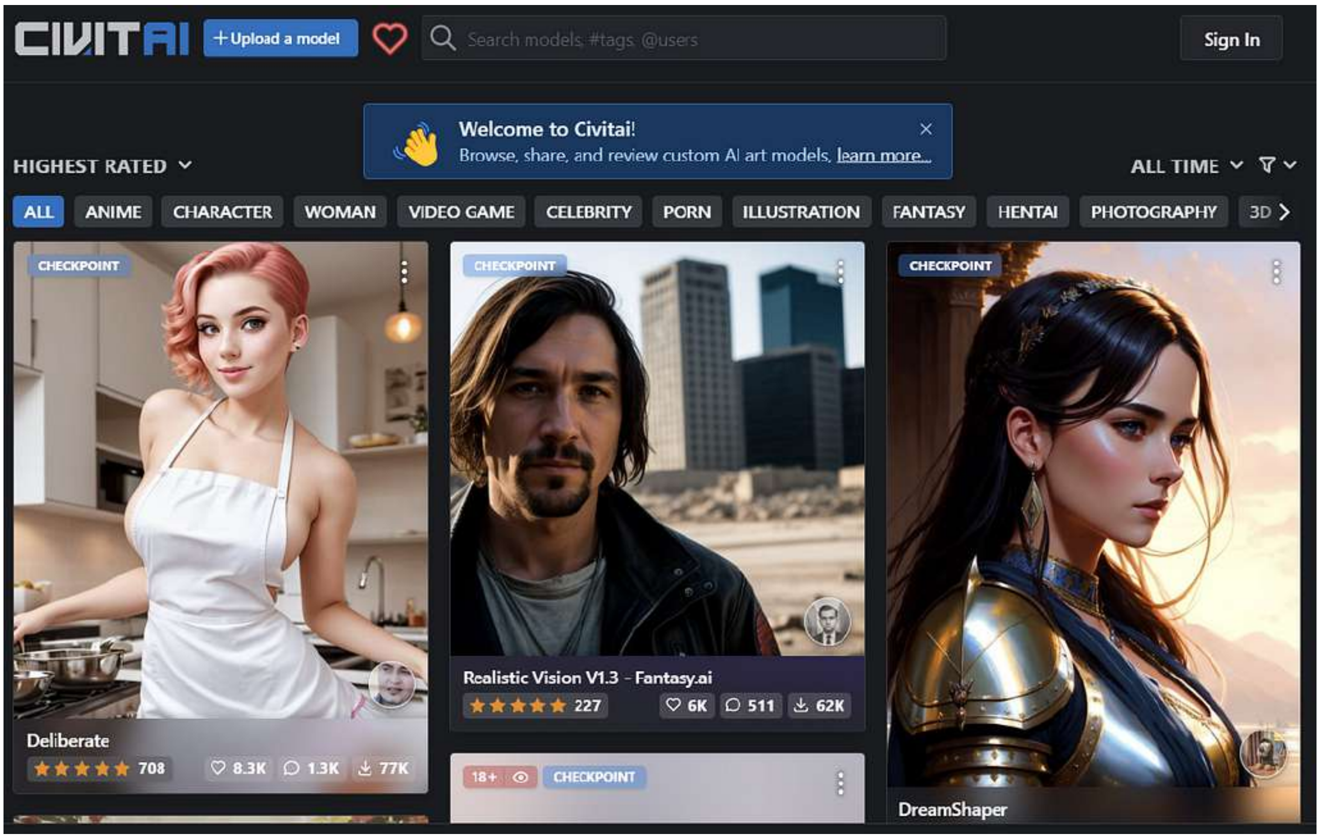
Now paste the following code in your terminal:

```
git clone https://github.com/AUTOMATIC1111/stable-diffusion-webui.git
```

Git will now start downloading the UI. Once the download has finished we can now run the UI. Go into the folder and double click on the file named: “webui-user.bat”.

This should start the download process for all the files needed by the UI. This can take a long time (depending on your internet connection). In the meantime, we can take a look at the models we can use.

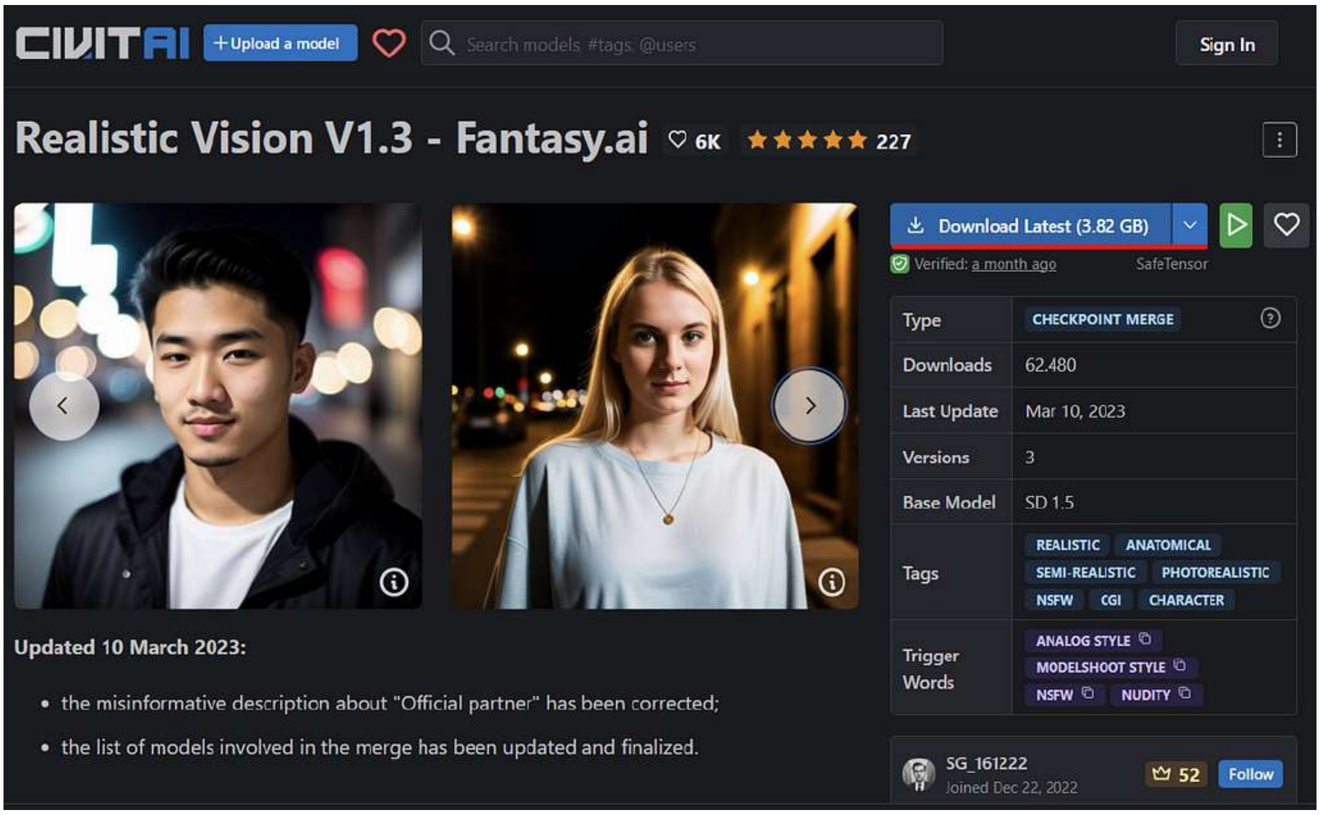
Installing a Model



By default the UI will download Stable Diffusion 1.4, but we can download a custom model to use with the UI. There are a lot of places where you can find a custom model, I usually find myseulf using Civit.AI:

[Civitai | Stable Diffusion models, embeddings, hypernetworks and more]()

On Civit.AI you'll find a lot of models divided by their type (realistic, illustrations, anime…), but you'll also find LORAs, Hypernetworks and much more. To download a model, just choose the one you'd like and click on the download button:



Realistic Vision V1.3 - Fantasy.ai ♡ 6K ★★★★★ 227

Verified: a month ago SafeTensor

Type	CHECKPOINT MERGE
Downloads	62.480
Last Update	Mar 10, 2023
Versions	3
Base Model	SD 1.5
Tags	REALISTIC, ANATOMICAL, SEMI-REALISTIC, PHOTOREALISTIC, NSFW, CGI, CHARACTER
Trigger Words	ANALOG STYLE, MODELSHOOT STYLE, NSFW, NUDITY

SG_161222 Joined Dec 22, 2022 52

Updated 10 March 2023:

- the misinformative description about "Official partner" has been corrected;
- the list of models involved in the merge has been updated and finalized.

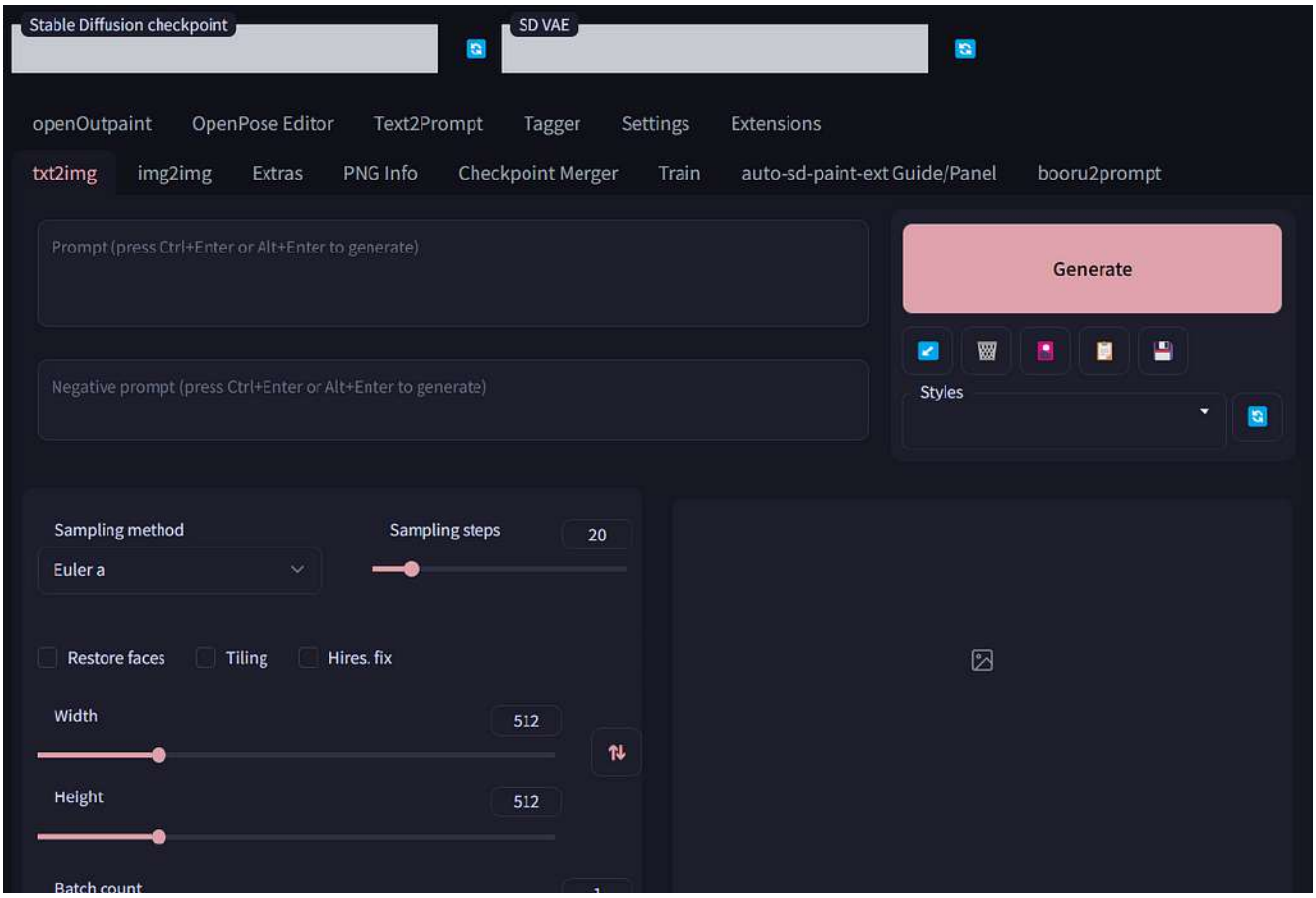
The downloaded models are usually available in two different formats: safetensors and ckpt. Both of them are supported by the UI.

Once the setup of the UI has completed you can install the new model you've downloaded by going to the UI folder and copying it into the path: models/stable-diffusion.

Using the Interface

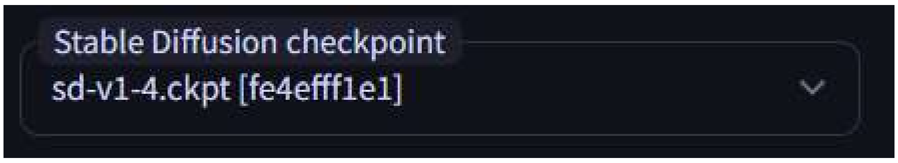
Now that we have completed the setup we need to open the interface. To do this you just need to open a new browser window and go to: http://localhost:7860.

This should open a window similiar to this one (I've got a lot of extensions and a different skin so it'll probably look different):



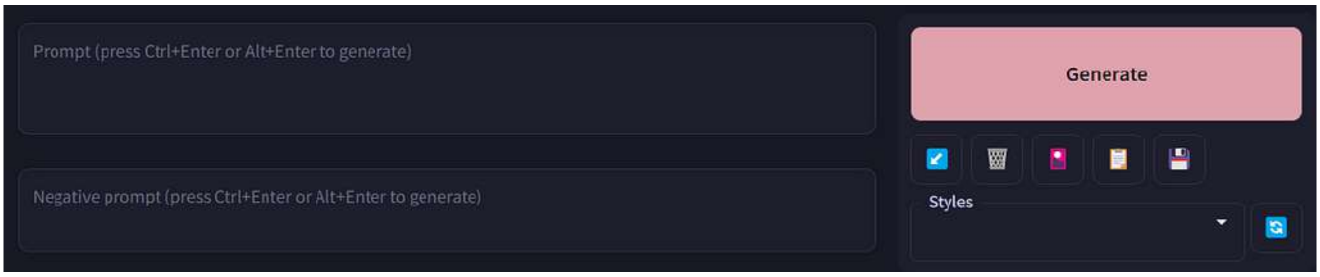
Let's have a look at the interface and explain the most important parameters.

On the top of the page we have the Stable Diffusion checkpoint. This is the model loaded right now. You should see the following model selected:



If you want to change the model you just need to click on the arrow and choose a new model (click on the refresh button on the right if you don't see the other models).

Then we have the prompt section:

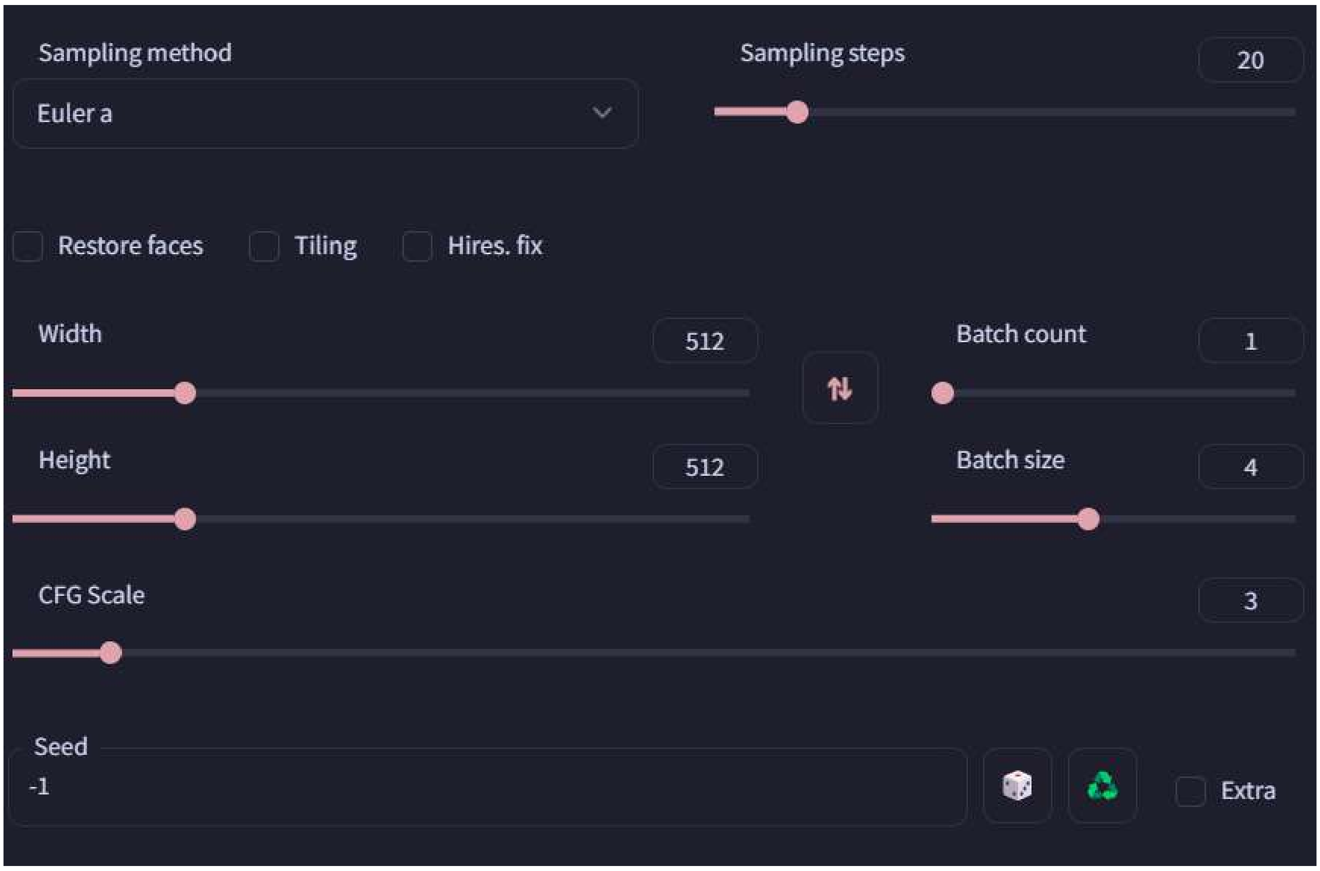

Prompt (press Ctrl+Enter or Alt+Enter to generate)
Negative prompt (press Ctrl+Enter or Alt+Enter to generate)
Generate
Styles

Here we can insert the prompt for our generation as well as the negative prompt. We can also do the following by clicking the icons under the generate button:

- Load the latest generation parameters
- Clear the prompt

- Load a Textual Inversion, Hypernetwork, Checkpoint, Lora
- Paste the selected style into the prompt and negative prompt
- Save the current prompt and negative prompt into a style

The last part we need to understand before we can start working with the UI is the parameters section:


Sampling method: Euler a
Sampling steps: 20
Restore faces
Tiling
Hires. fix
Width: 512
Height: 512
Batch count: 1
Batch size: 4
CFG Scale: 3
Seed: -1
Extra

Here we have the following parameters:

- **Sampling Method**: this changes the sampler used to diffuse the image. If you are a beginner I suggest you to use Euler or DDIM as they are usually the fastest.
- **Sampling Steps**: this changes the “length” of the diffusion. If this is too short the image won’t be completed. If you use Euler or DDIM 15–20 steps are usually good.
- **Restore Faces**: this fixes the faces by using another model. This is a destructive process, it’ll create a different face from the one that was originally generated.
- **Tiling**: this enables you to generate images to use as textures.
- **Hires. fix**: this creates the image in the size you’ve defined, upscales it using one of the upscaler models and then apply the same prompt using the image2image to add more details.
- **Width/Height**: this changes the size of the image generated. **Larger images need more VRAM to generate**.
- **Batch Count**: the number of batches of images to generate. This is multiplied by the “Batch Size” option

- **Batch Size**: the number of images generated in parallel. **You’ll need a lot of VRAM to increase this parameter**.
- **CFG Scale**: how much the prompt will influence the output. Low values will generate images much different from the prompt, while high values will generate a lot of distortion. Usually a value between 7–9 is a good starting point.
- **Seed**: this is the seed used to generate the image. Using the same seed with the same parameters and prompts will always generate the same picture. This can be useful to check how an image changes based on the parameters or prompts. -1 equals to random seed.

Generating an Image

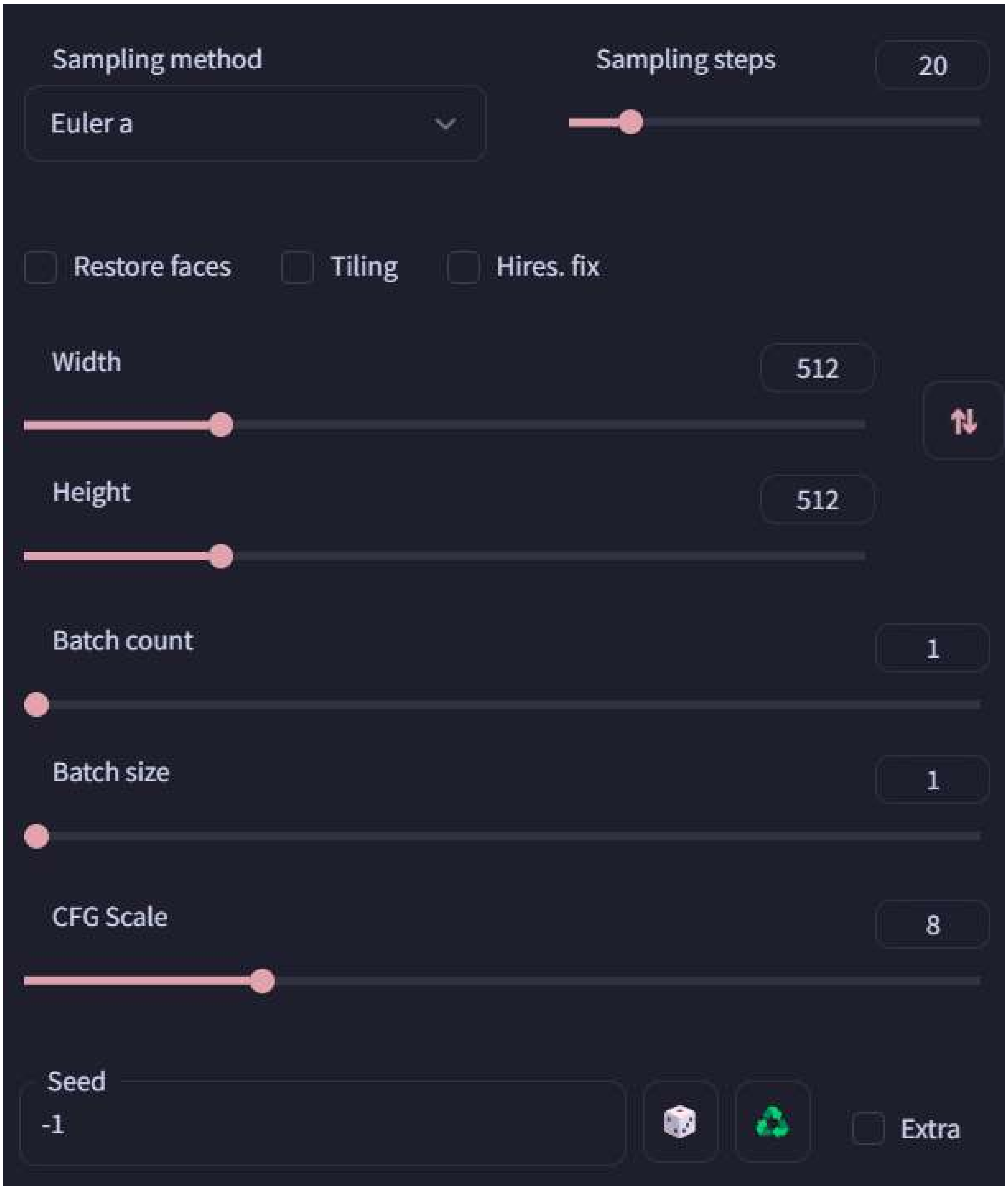
We can finally generate an image. Let’s start by writing the following prompt:

a painting of a dog riding a skateboard at the beach, masterpiece, detailed, 4k

We also add the following to the negative prompt:

blurry, out of focus, bad quality, lowres, text, error, cropped, worst quality, low quality, jpeg artifacts, ugly, duplicate, morbid, mutilated, out of frame, extra fingers, mutated hands, poorly drawn hands, poorly drawn face, mutation, deformed, blurry, dehydrated, bad anatomy, bad proportions, extra limbs, cloned face, disfigured, gross proportions, malformed limbs, missing arms, missing legs, extra arms, extra legs, fused fingers, too many fingers, long neck, username, watermark, signature

For the parameters you can use these as a good starting point:



Sampling method: Euler a

Sampling steps: 20

[ ] Restore faces [ ] Tiling [ ] Hires. fix

Width: 512

Height: 512

Batch count: 1

Batch size: 1

CFG Scale: 8

Seed: -1

[ ] Extra

If you run this prompt you should now see the generated output. For example these are 4 images generated using my prompt:

Stable Diffusion Ultimate Guide pt. 2: Prompting

In the previous chapter we've seen how to install the automatic1111 UI and it's requirements on your machine. In this chapter, we will cover some aspects of Stable Diffusion that can help you improve your results and customize your prompts. We will discuss:

- **Basic prompting**: how to use a single prompt to generate text, and how to evaluate its quality and stability.
- **Changing prompt weights**: how to adjust the importance of each prompt keyword in relation to the others.
- **Prompt Editing**: how to change the number of steps that the model takes for a specific prompt.
- **Prompt order**: how to order your prompts according to their relevance and impact.
- **Prompt matrix**: how to use a matrix of prompts to test different variations in a single run.

By learning these concepts, you will be able to master Stable Diffusion and create high-quality texts for any purpose.

Basic Prompting

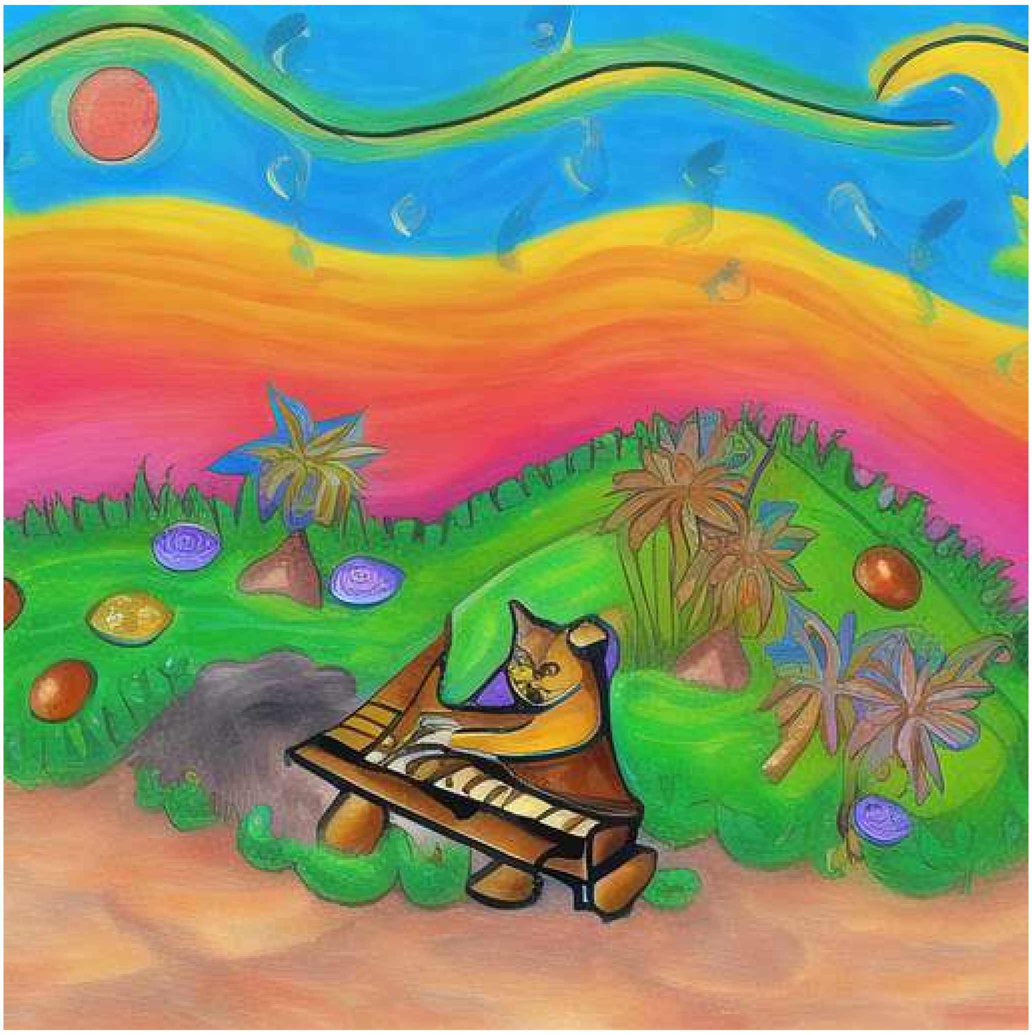
Learn the Checkpoint

Let's start by the basics. The first thing we need to understand when we write a prompt for a specific Stable Diffusion checkpoint is how the checkpoint has been trained.

For example the standard SD1.4 model has been trained to work with natural language prompts while other checkpoints work better with keywords separated by commas.

In SD1.4 this prompt:

a child drawing of a cat playing a piano on a desert island in the summer, detailed, HD, colorful



Works better than this one:

child drawing, cat, playing a piano, desert island, summer, detailed, HD, colorful

When you download a custom checkpoint, always check if there is any hint about how the model should be prompted in the description. Another quick way to understand how to correctly use the model is to check some example prompts made by the creator of the checkpoint or by the community.

As a quick reminder, you can find new checkpoints online. I usually get them from civitai.com:

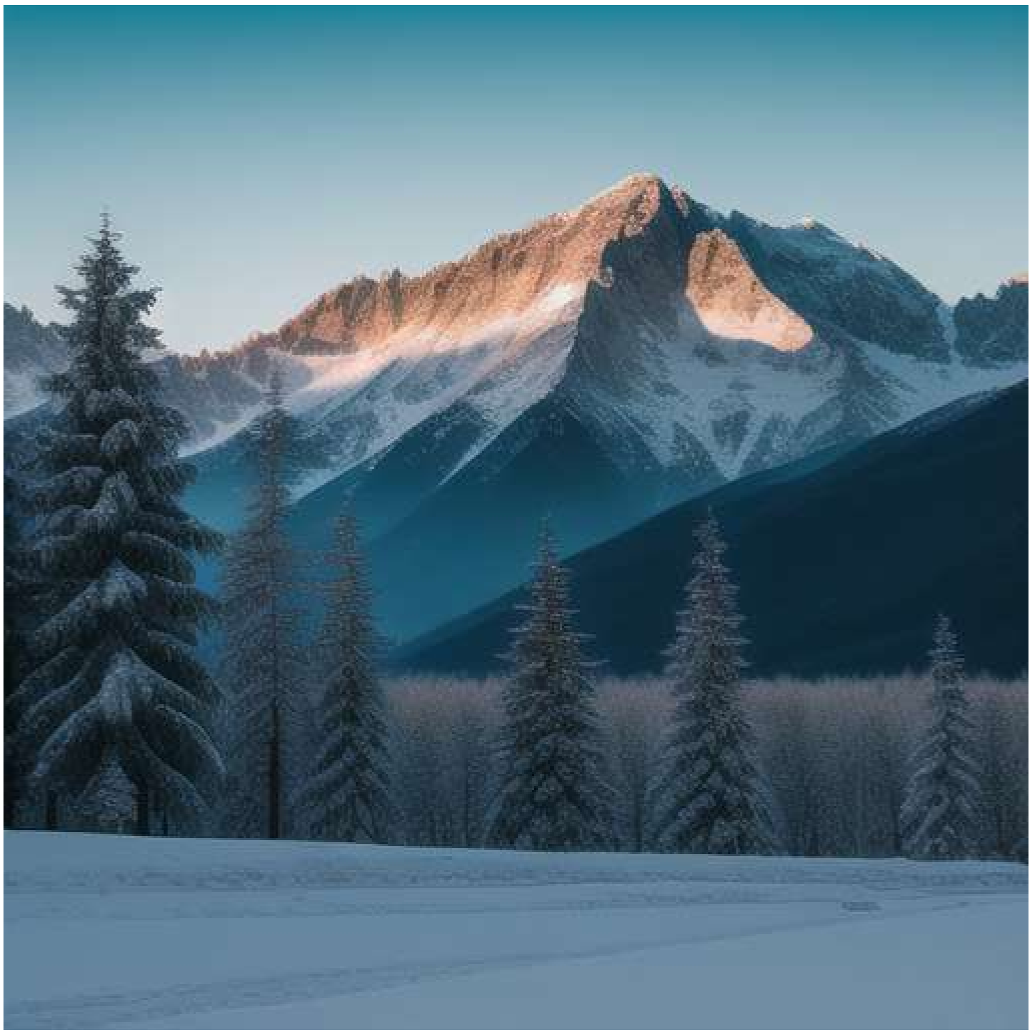
Civitai | Stable Diffusion models, embeddings, hypernetworks and more

Quality Prompts

Another important thing to remember when prompting is to use words that can impact the quality of your picture. When image are trained in Stable Diffusion they are usually accompanied by adjectives that describes the quality of the image. These can be positive (beautiful, detailed, masterpiece) or negative (bad, awful, deformed). Using these in your prompts can drastically change the quality of your picture.

For example the following prompt:

winter landscape, mountains, trees, 8k uhd, dslr, soft lighting, high quality, film grain, Fujifilm XT3, RAW photo



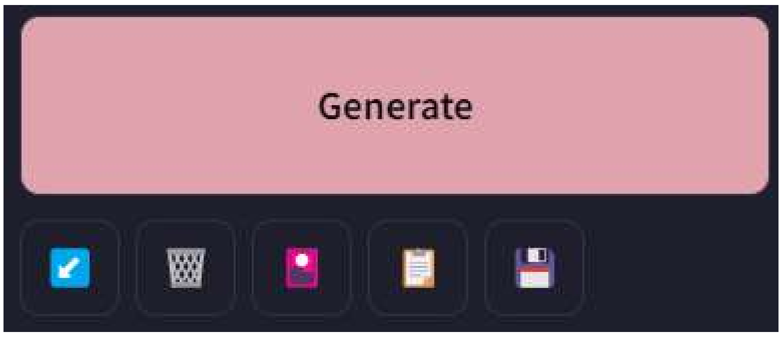
Is better than this one:

winter landscape, mountains, trees

Using Styles

A useful thing to know when you are using the automatic1111 UI is the possibility to save and load prompts by using the “Styles” functionality.

To do this you just need to write a prompt as you’d usually do and then just click the file disk icon on the right (under the generate button):



Then just choose a name for the style and save it.

Now you can call that prompt by selecting it in the styles box under the icons. If you want to paste the style into your active prompt you just have to click the clipboard on the left of the file disk icon.

Now that we've seen the basics of prompting let's have a look at something more interesting.

Prompt Weights

When writing a prompt each word you use has the same relevance inside of the sentence. But what if you'd like to emphasize the importance of a single part of your prompt?

You can do this in the automatic1111 UI by using brackets:

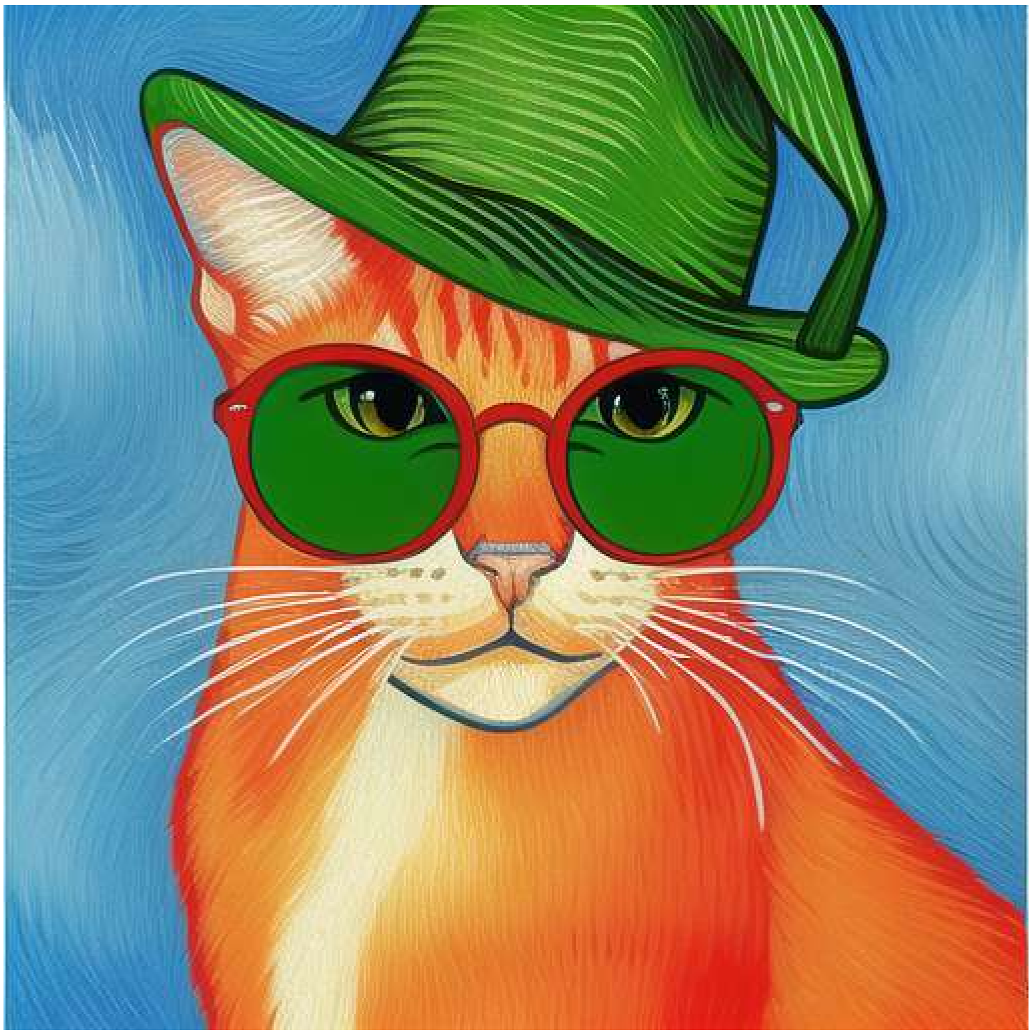
- `a (word)` - increase attention to `word` by a factor of 1.1
- `a ((word))` - increase attention to `word` by a factor of 1.21 (= 1.1 * 1.1)
- `a [word]` - decrease attention to `word` by a factor of 1.1
- `a (word:1.5)` - increase attention to `word` by a factor of 1.5
- `a (word:0.25)` - decrease attention to `word` by a factor of 4 (= 1 / 0.25)
- `a \(word\)` - use literal `()` characters in prompt

Pay attention that if you want to decrease the importance of a word by a specific factor you need to use round brackets, while if you want to quickly do it you have to use square brackets. If you do something like [word:0.5] it won't work (and you'll understand why in the next paragraph).

So for example the following prompt:

a red cat wearing a green hat and glasses, masterpiece, detailed, painting in the style of van gogh

Produces the following image:



But if you use the following prompt:

a red cat wearing a green hat and glasses, masterpiece, detailed, (painting in the style of van gogh:1.2)

The style will be more similar to the one specified in the prompt:

You can also use the same syntax in the negative prompt to change how the prompt will affect the final composition.

Prompt Editing

Prompt editing is an incredible feature that everyone using Stable Diffusion should know. Prompt editing allows you to start sampling one picture, but in the middle swap to something else. The base syntax for this is:

[from:to:when]

This enables you to start by creating a picture for a concept and then change it after a specific number of steps.

There are also 2 alternative syntaxes for this functionality:

- `[to:when]` - adds `to` to the prompt after a fixed number of steps (`when`)
- `[from::when]` - removes `from` from the prompt after a fixed number of steps (`when`)

You can specify the number of steps in two different ways:

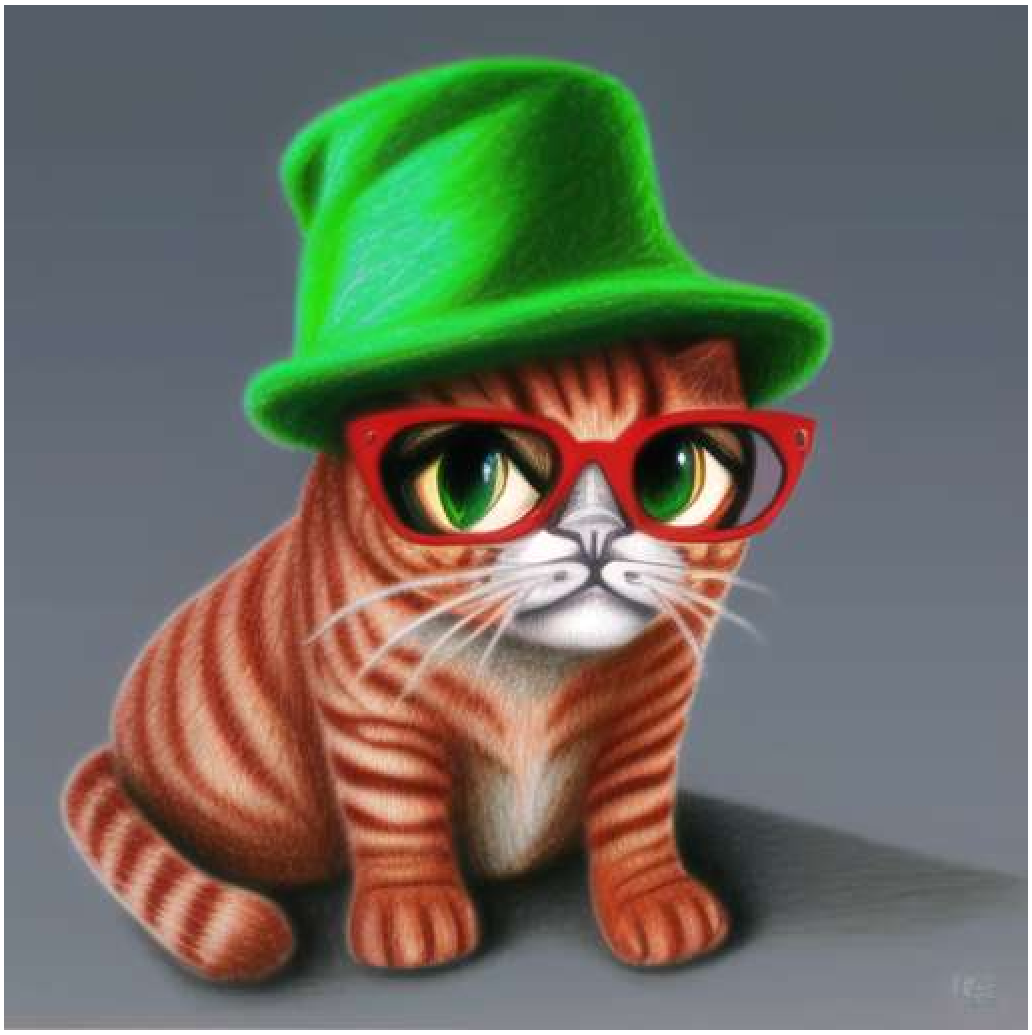
- Using an integer: this determines the exact number of steps after which the prompt is changed

- Using a float: this determines the percentage of steps after which the prompt is changed. For example: 0.5 means that half the steps are done with the from part while the other half is done with the to part.

Let’s have a look at some examples:

a red cat wearing a green hat and glasses, masterpiece, detailed, [3d model:(pencil drawing:1.4):0.5]

This prompt generates a drawing with a perspective similar to a 3D model:



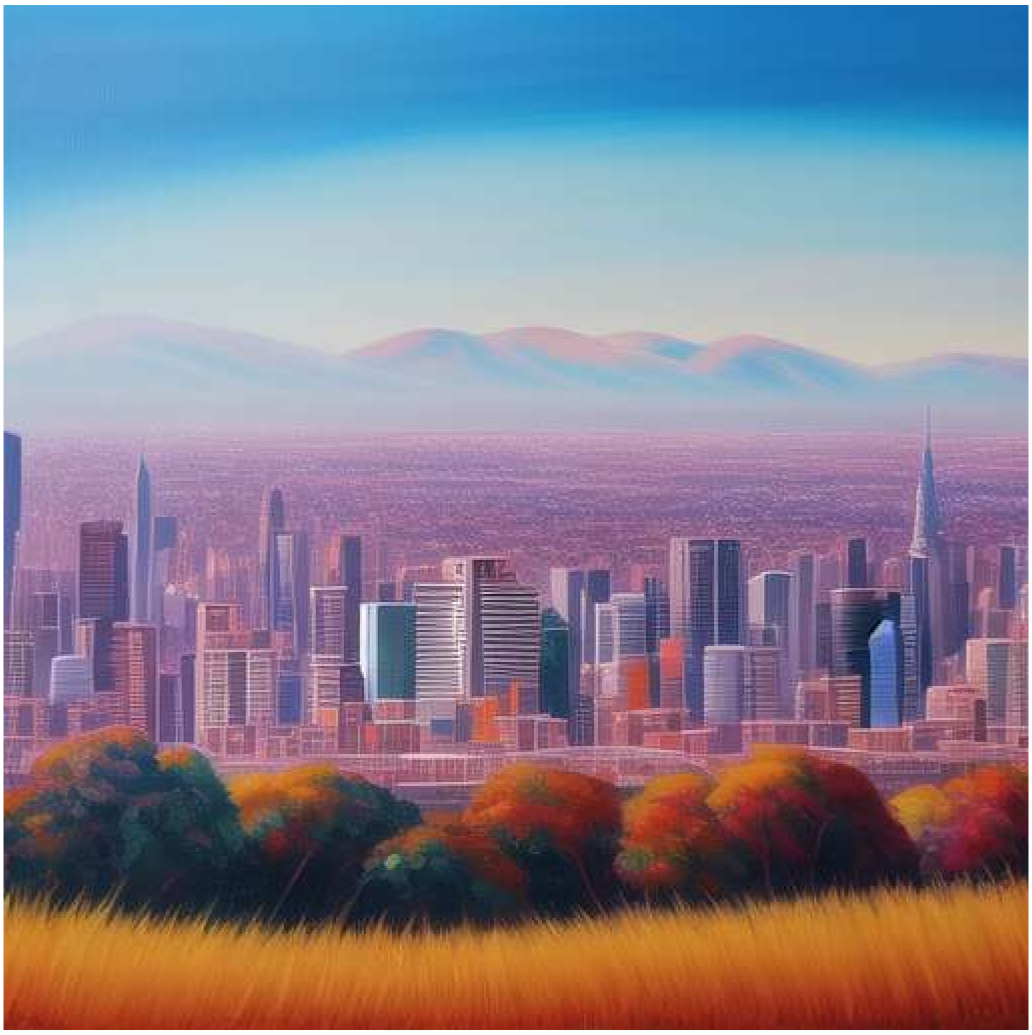
If you repeat the same prompt without the prompt editing part you’ll produce a totally different result:

Another example:

Let’s start from a simple prompt:

painting of a modern city

The output:



But what if we change the prompt to include abstract after 25% of the steps:

painting of a modern [abstract:0.25] city

Using the same seed this is now the result:

A similar image, but with a completely different style!

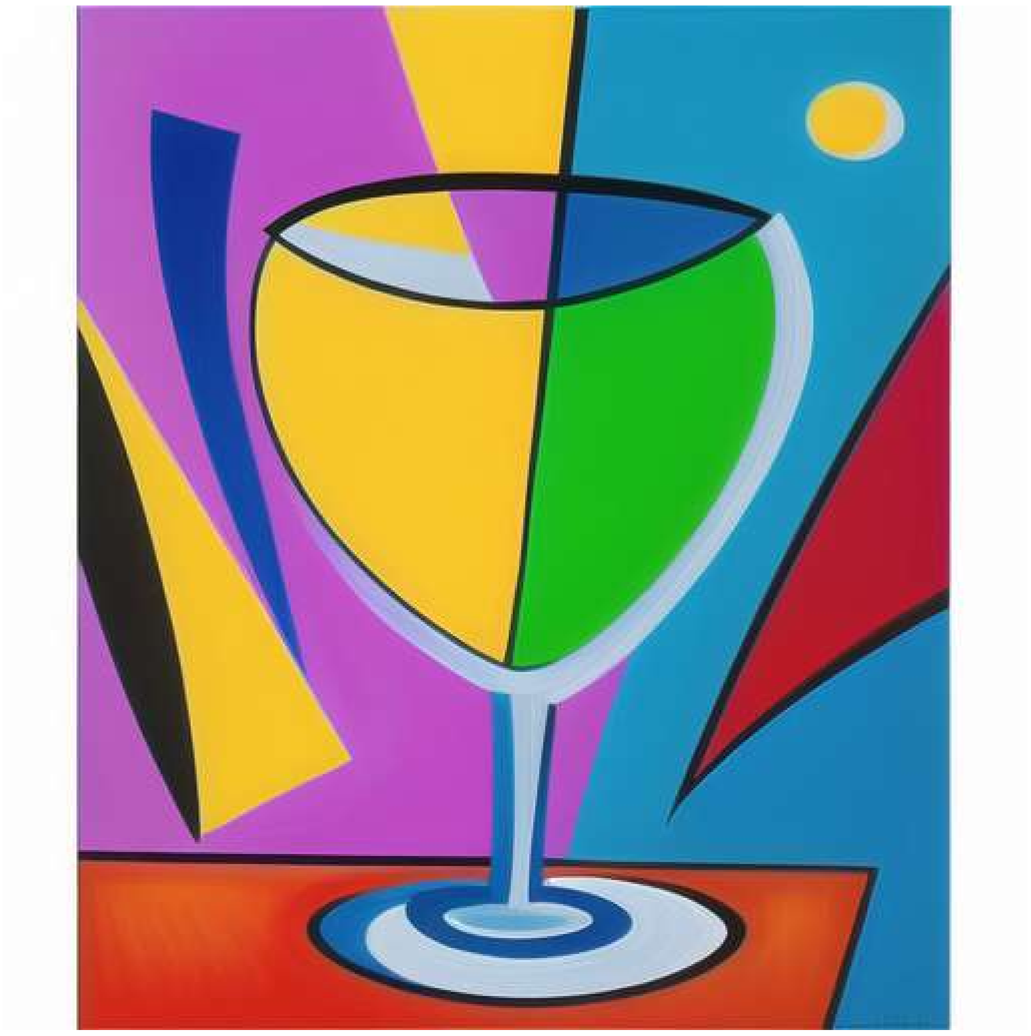
Exploring Prompt Order

Another interesting thing to know about prompting in Stable Diffusion is that the order of the words inside the prompt can change drastically the output image.

Take for example the following prompt:

painting of a glass on a table in the style of Pablo Picasso, abstract, (masterpiece, detailed, HD)

Output:



If we put abstract at the beginning of the prompt and run the same seed:

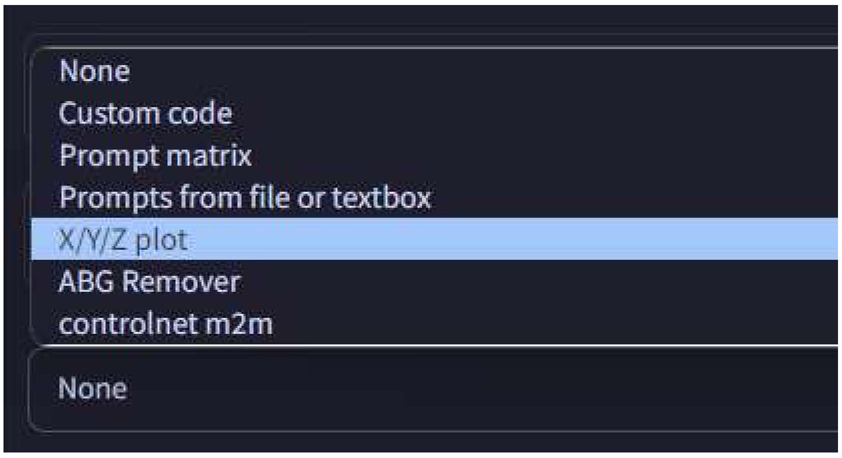
abstract painting of a glass on a table in the style of Pablo Picasso, (masterpiece, detailed, HD)

As you can see the final results is different from the original!

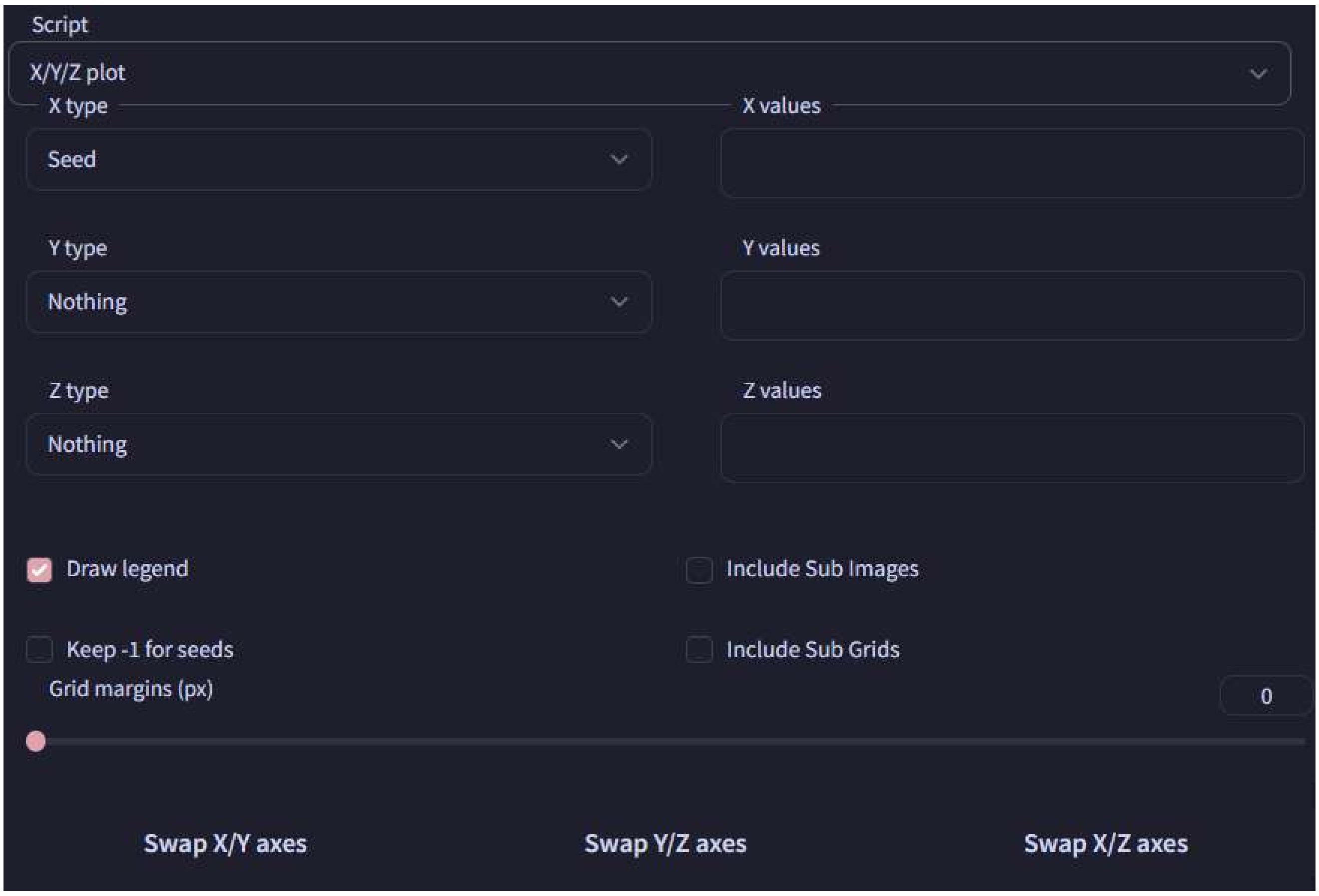
X/Y/Z Plot

Now that we've seen some of the functionalities you can use in the UI, how can you compare different prompts with each other?

To try different prompts in one go you can to the script section on the bottom of the prompting menu and choose the X/Y/Z Plot:



You should now see a menu that looks like this:


Script: X/Y/Z plot

X type: Seed | X values:

Y type: Nothing | Y values:

Z type: Nothing | Z values:

[✓] Draw legend
[ ] Include Sub Images
[ ] Keep -1 for seeds
[ ] Include Sub Grids

Grid margins (px): 0

Here you can choose different attributes to plot on the three different axes. So let’s try building a plot to check how a seed changes based on the prompts.

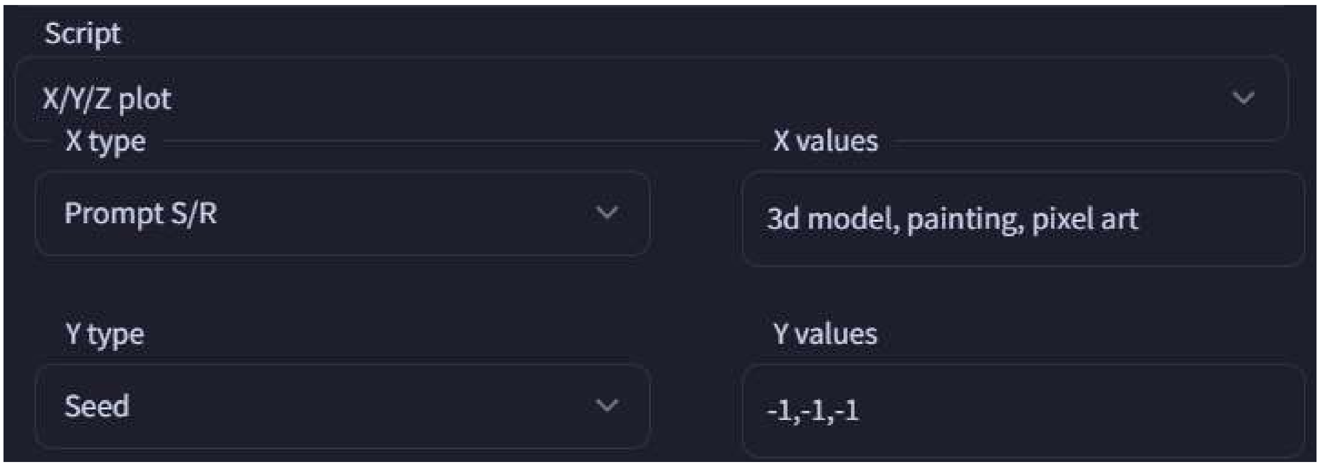
Let’s start by writing a prompt. I’ll use the following:

3d model of a cat standing on a table in a forest, (masterpiece, detailed, HD)

Now we can change the “X Type” option to “Prompt S/R”. This function search for a string in your prompt and replaces with a list of strings you define inside the X values generating one image per word. The first word must be already present inside the prompt. In my case I’ll use: 3d model, painting, pixel art.

On the “Y Type” option we choose “Seed”. In this field you can specify a list of seeds separated by a comma; this will generate an image for each seed you specify. If you want a random seed just use -1. In our case I’m going to use: -1,-1,-1.

So your settings should look like this:


Script

X/Y/Z plot

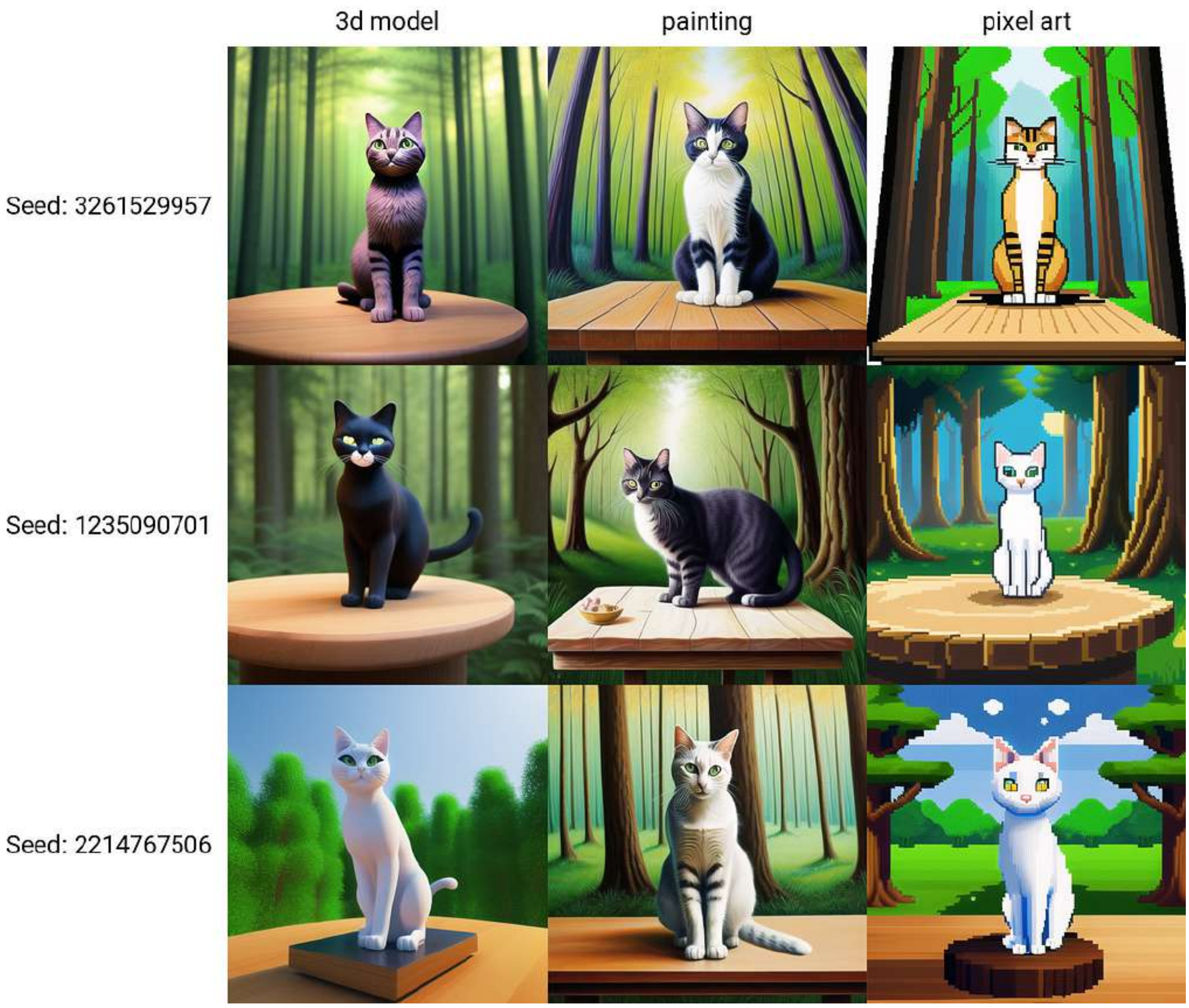
X type: Prompt S/R

X values: 3d model, painting, pixel art

Y type: Seed

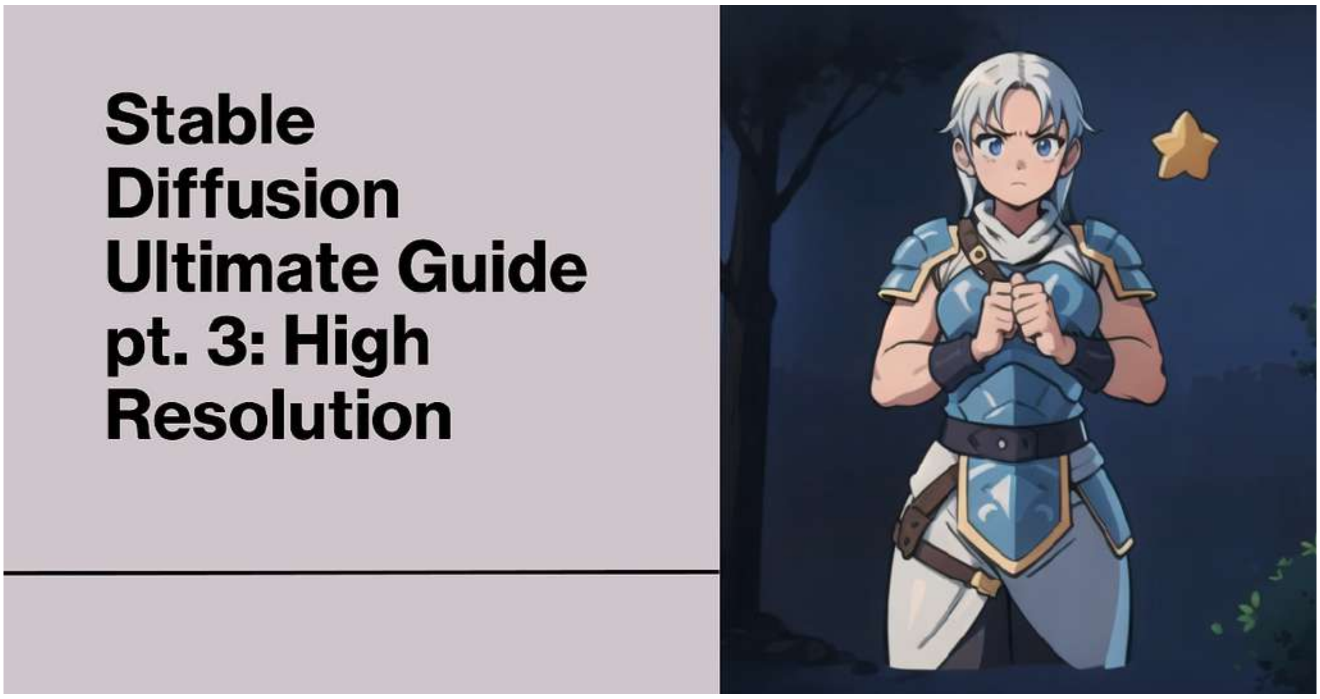
Y values: -1,-1,-1

If you run the prompt you should see a result similar to this:



And that’s all for today!

Stable Diffusion Ultimate Guide pt. 3: High Resolution



In the third chapter, we are going to have a look at how to work with higher resolution images. Stable Diffusion v1.5 is trained on 512x512 images (while v2 is also trained on 768x768) so it can be difficult for it to output images with a much higher resolution than that. To do this consistently, we'll need to do a few tricks.

For today's example, I'm going to use my own custom checkpoint (model) that you can find here:
Mistoon_Anime | Stable Diffusion Checkpoint | Civitai

Requirements

To follow this example, you'll need:

- The Automatic1111 Stable Diffusion UI. If you don't know how to install it, check out the first part of this series here: https://medium.com/p/bd7dbcd5ce4b
- A GPU that can handle images of 768x1152 size. I'm using a RTX 4080 with 16 GB of VRAM. In case you have access to less VRAM, you'll probably be able to work with a lower resolution.
- An image editing program. I'm using Krita (which is an awesome free digital painting application), but you can use your preferred tool (Photoshop, Gimp, Paint.Net…).
- (Optional): a textual inversion for the negative part. For this example I'm using this one.

Let's Start

Open the Stable Diffusion UI and start from a blank prompt. If you want, you can copy my prompt or use yours.

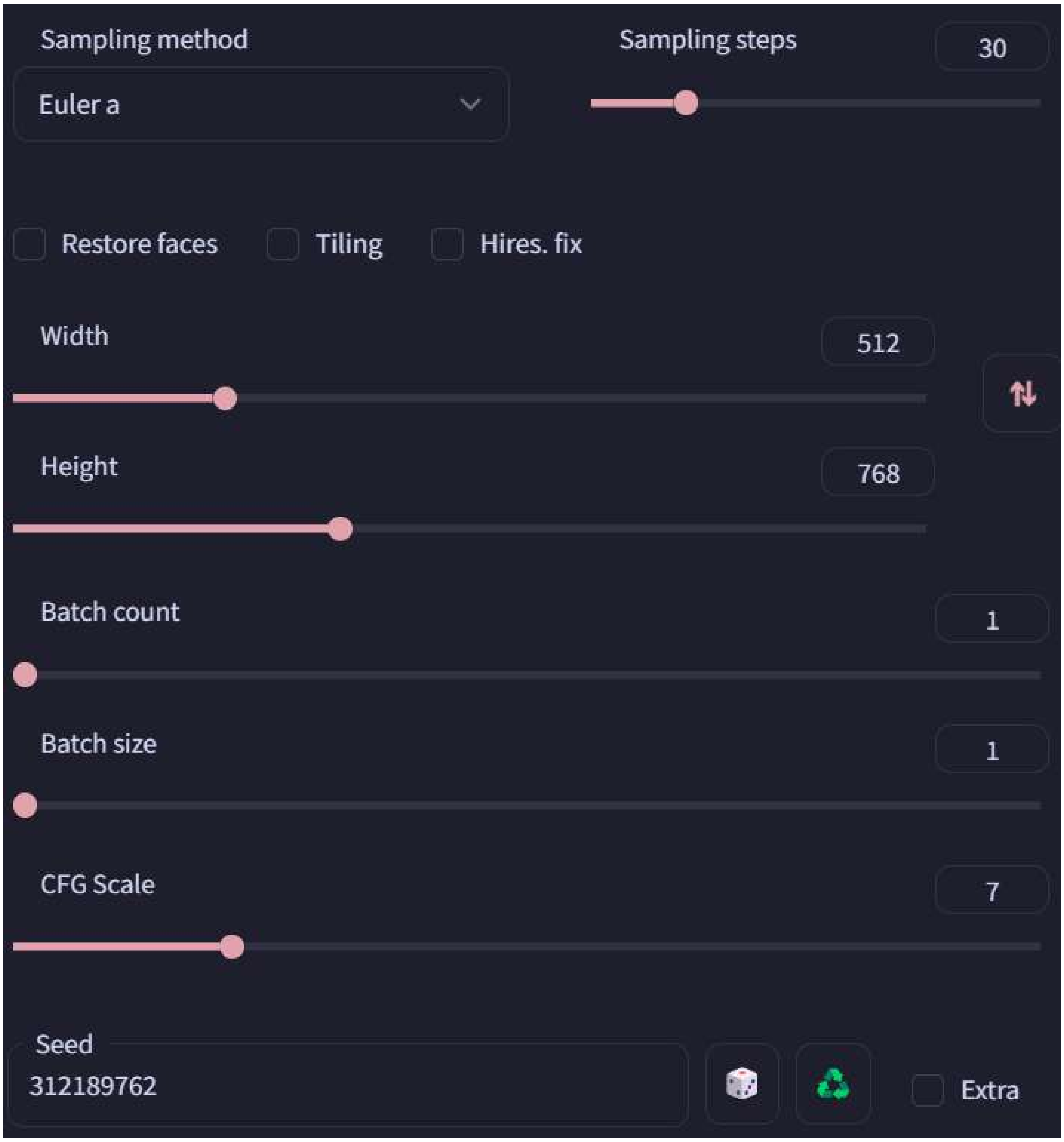
Prompt:

girl, standing, short hair, red hair, pigtail, suit, tie, professional, office, indoors, glasses, dynamic pose

Negative Prompt:

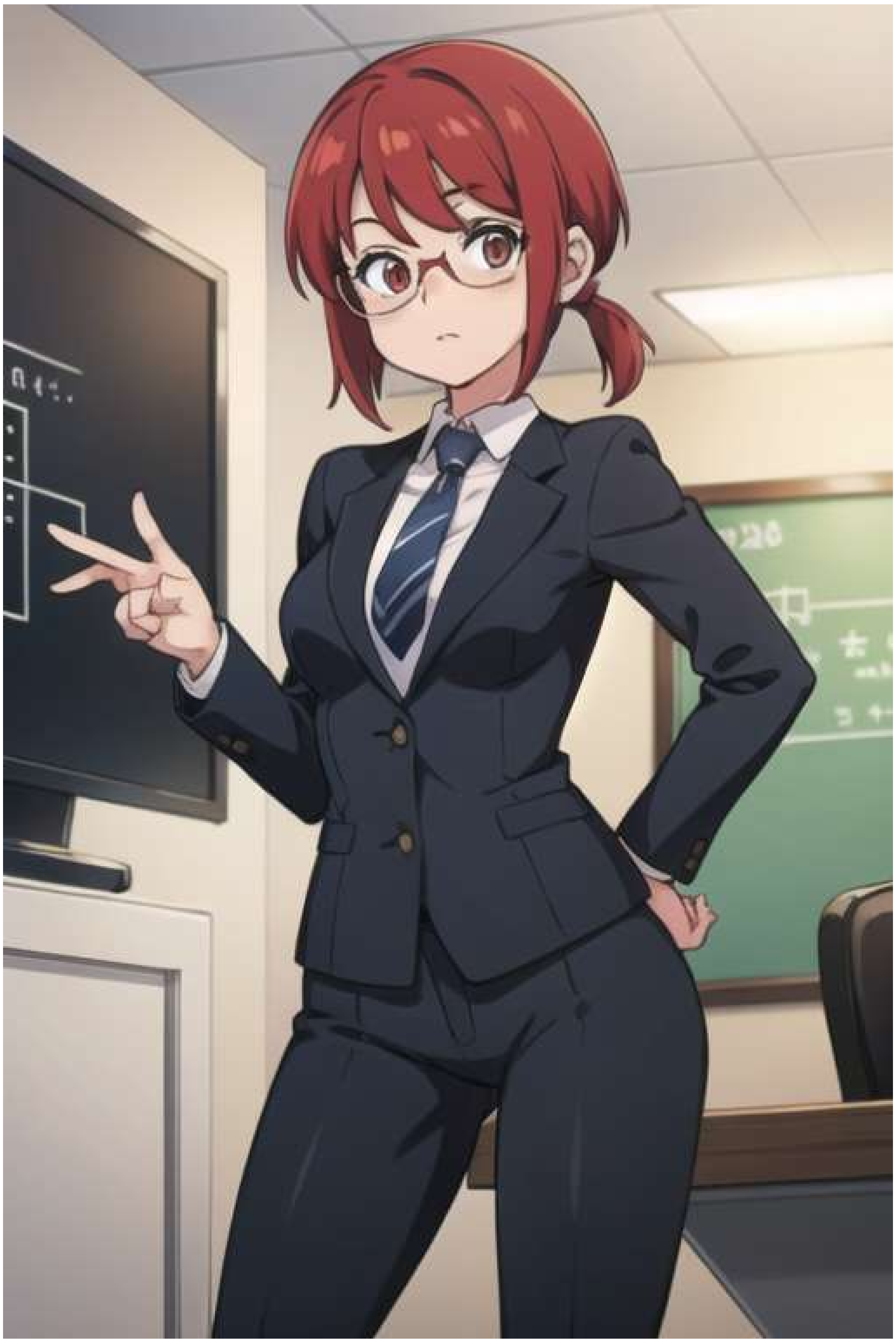
verybadimagenegative_v1.3

Other Properties:



Sampling method: Euler a
Sampling steps: 30
Restore faces: ☐
Tiling: ☐
Hires. fix: ☐
Width: 512
Height: 768
Batch count: 1
Batch size: 1
CFG Scale: 7
Seed: 312189762
Extra: ☐

If you use these settings with the seed “**312189762**”, you should see the following image:



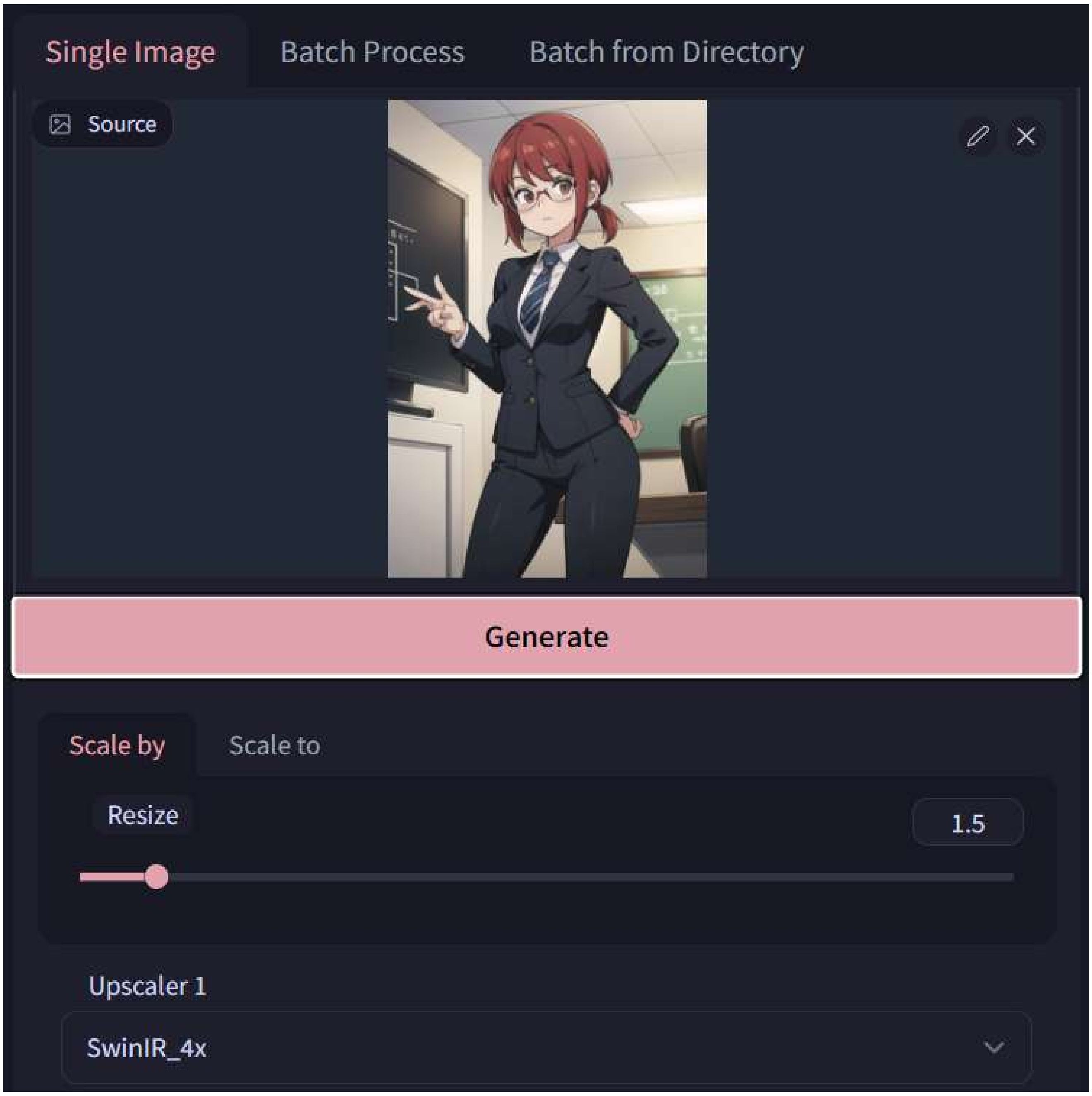
Let’s say that we like the general result, but we want to fix a few things:

- The right hand looks distorted, so we’ll have to change it.
- The TV screen on the left has some strange drawings on it. I’d like to make it pitch black.
- I’d like to change the tie color to red.

To make these changes we could use the inpainting functionality and change the single elements, but today I’m going to show you how not only to change these elements, but also upscale the resolution in a convincing way.

The first thing we need to do is to upscale the image using the upscale functionality from the Extra tab. We are going to do this before editing because the upscale can introduce some distortion that we can fix later. You can copy the image and paste it inside that tab, or just click on the "Send to Extras" button:

Once you've opened the tab, change the settings to match my picture and then run the upscale by clicking the generate button:

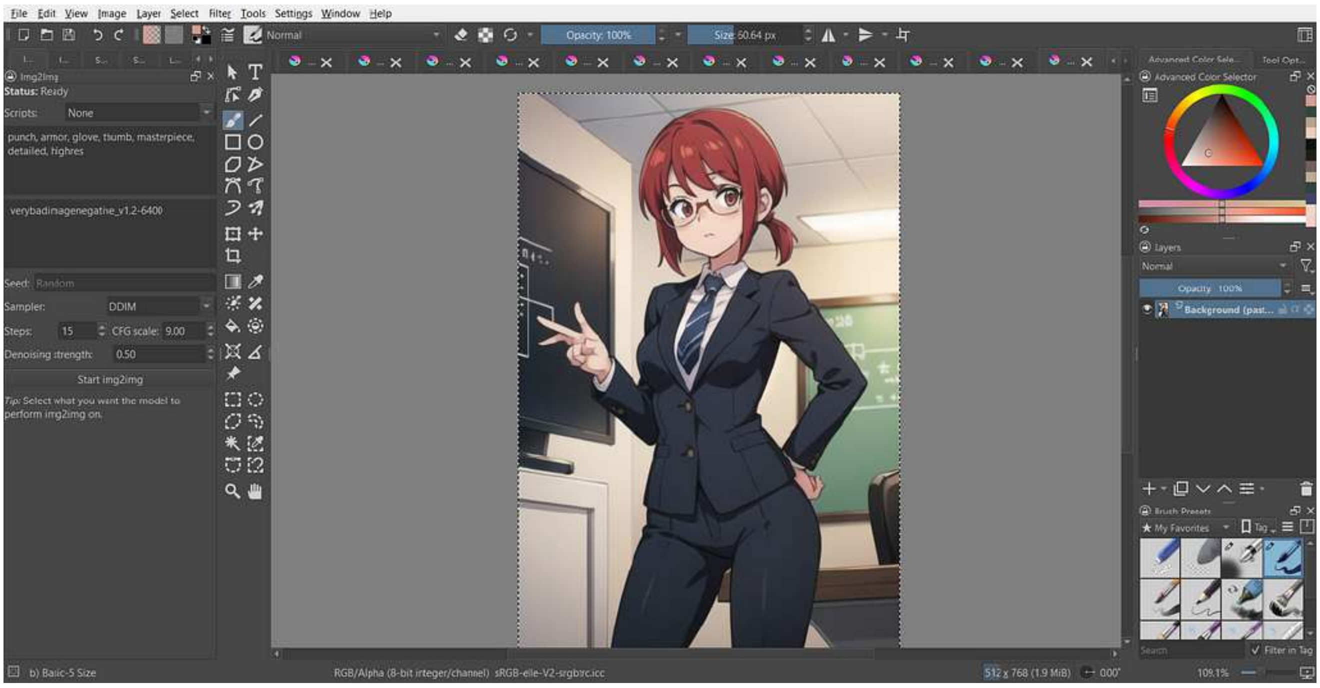


Resize: 1.5

Upscaler 1: SwinIR_4x

If you've never used the "SwinIR_4x" upscaler, it will take some time to download it.

Now, let's open the image editing software (as I've said before, I'm going to use Krita). Copy the picture from the upscale output and once inside the software just paste it (use ctrl+shift+n in Krita to paste into a new window):



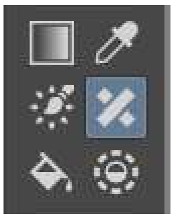
Img2Img
Status: Ready
Scripts: None
punch, armor, glove, thumb, masterpiece, detailed, highres
verybadimagenegative_v1.2-6400
Seed: Random
Sampler: DDIM
Steps: 15 CFG scale: 9.00
Denoising strength: 0.50
Start img2img
Tip: Select what you want the model to perform img2img on.

Once inside the editor, let’s start by cleaning the hand. We are not going to draw a correct hand, but we just need to give it a shape that SD can use to create a new hand.

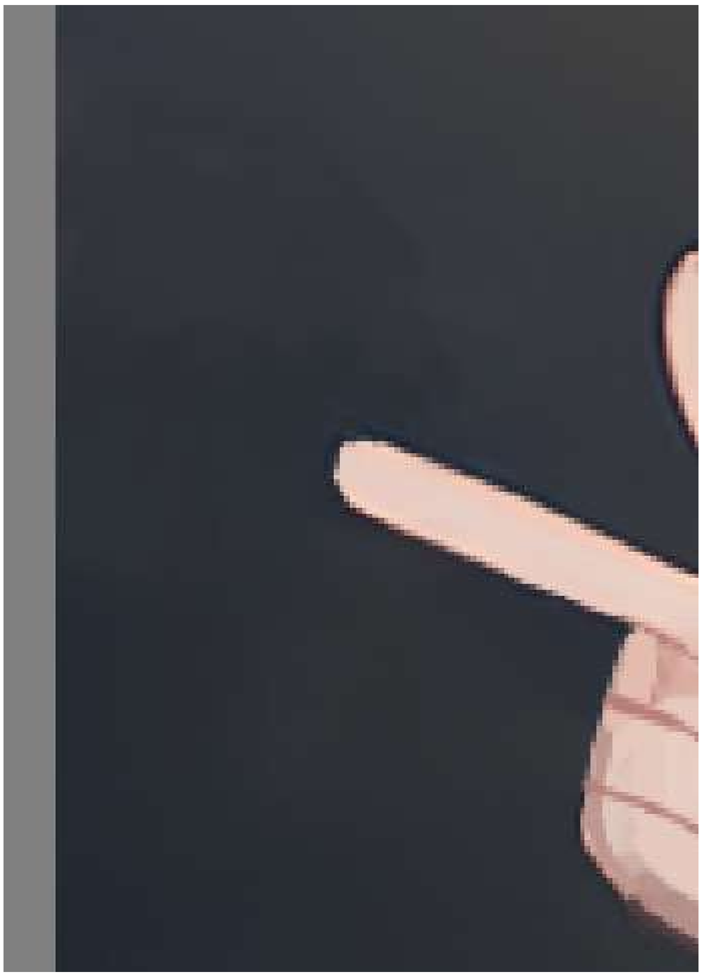
Select the brush tool (b) and choose a color from the TV by holding CTRL and then clicking with the left mouse button on the television screen. Once you’ve selected the colors, remove the crooked finger and clean the hand.

Something like this should work (as I’ve said before, you don’t need to be precise):

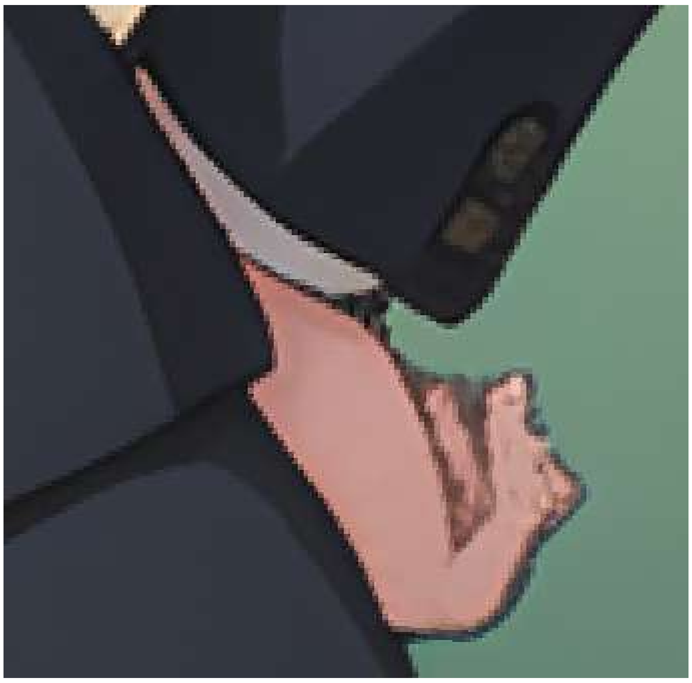
Now let’s do the same on the TV screen. In this case, you can probably use the “Smart Patch Tool” to rapidly clean most of the screen:



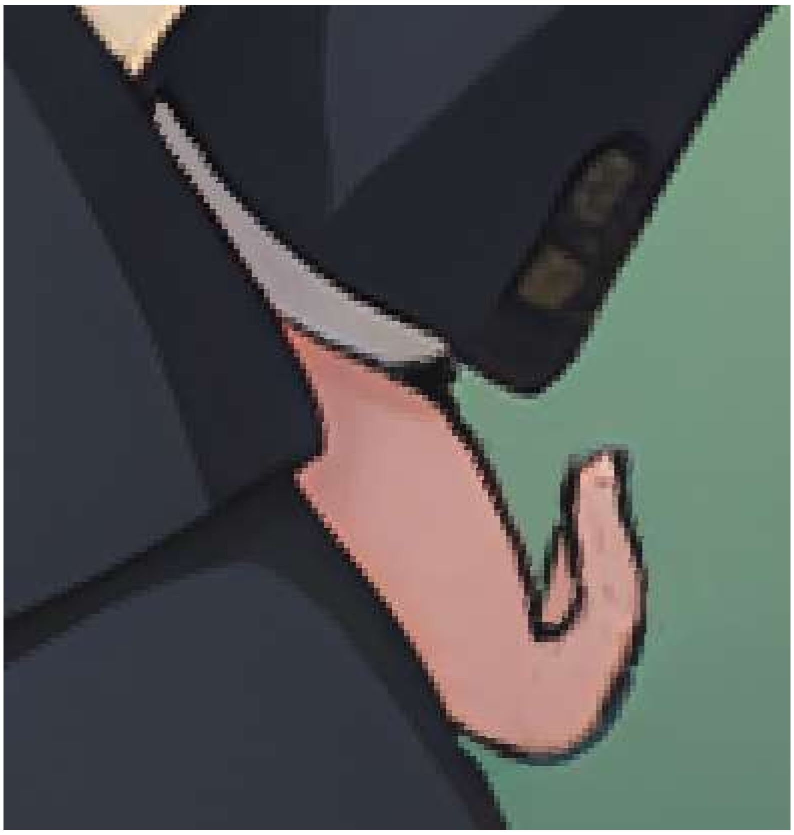
The result should look like this:



Before we work on the tie, let’s also clean the other hand:



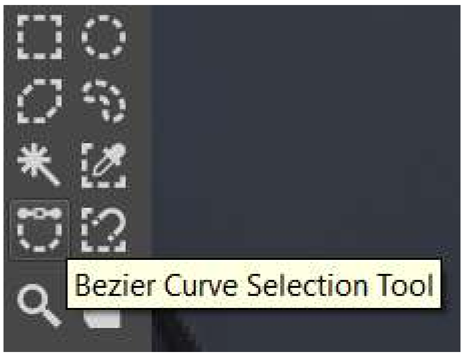
As you can see, the upscale has created a lot of distortion. We can simplify the shape so that SD will be able to create a nicer hand easily:



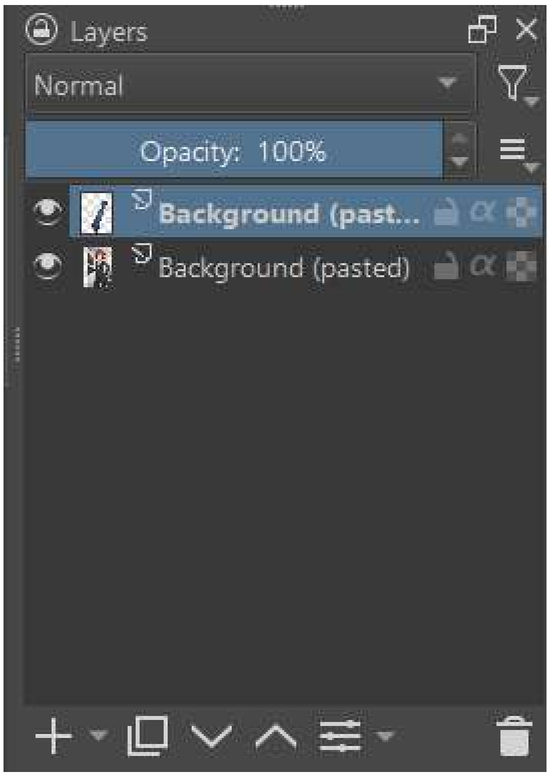
Now let’s start working on the tie. We have different ways to do this:

- We can just fill the shape of the tie with the red color
- We can use the hue tool to change the color of that part of the image

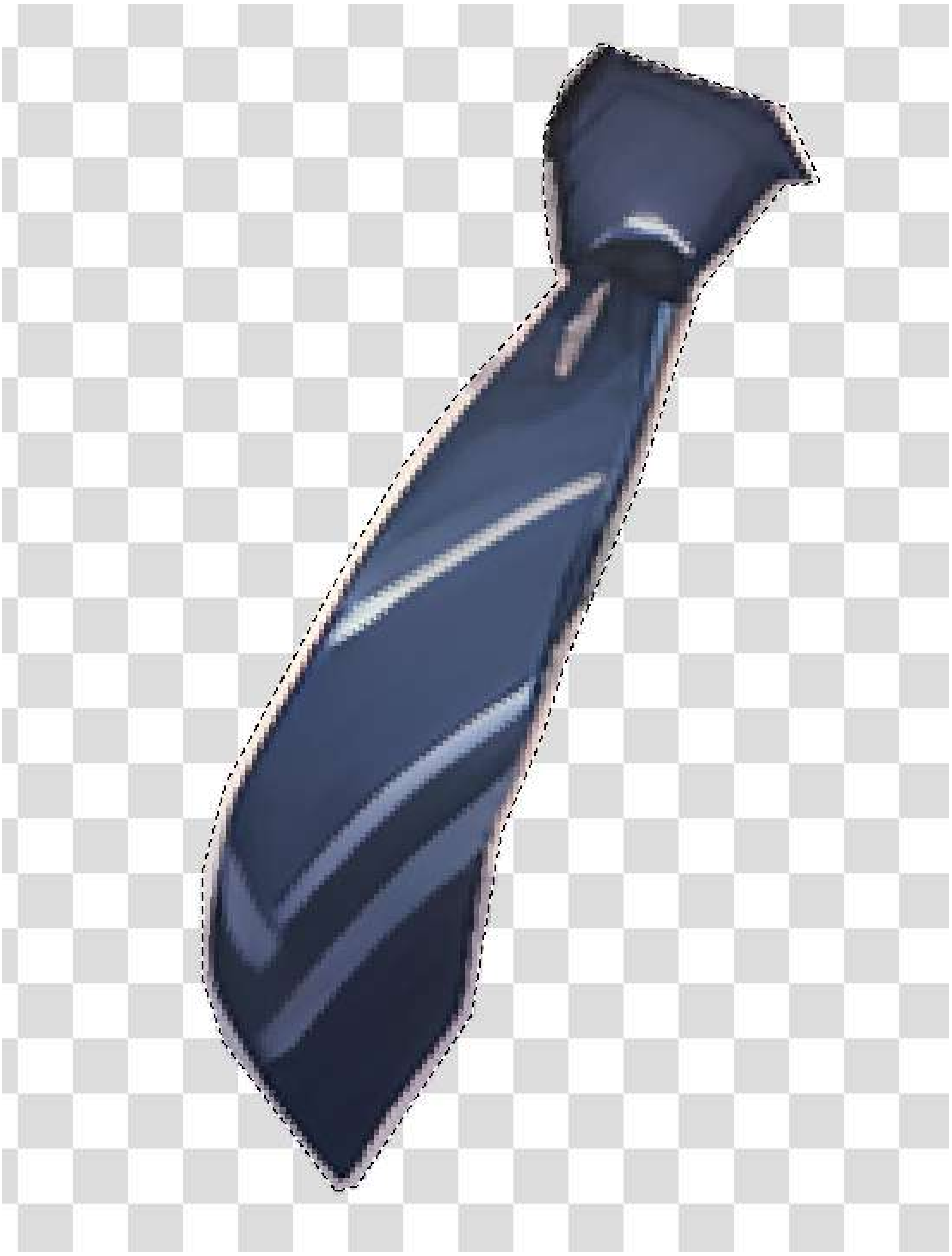
In this example, we are going to use the second method. Let’s start by selecting the tie using the “Bezier Curve Selection Tool”:



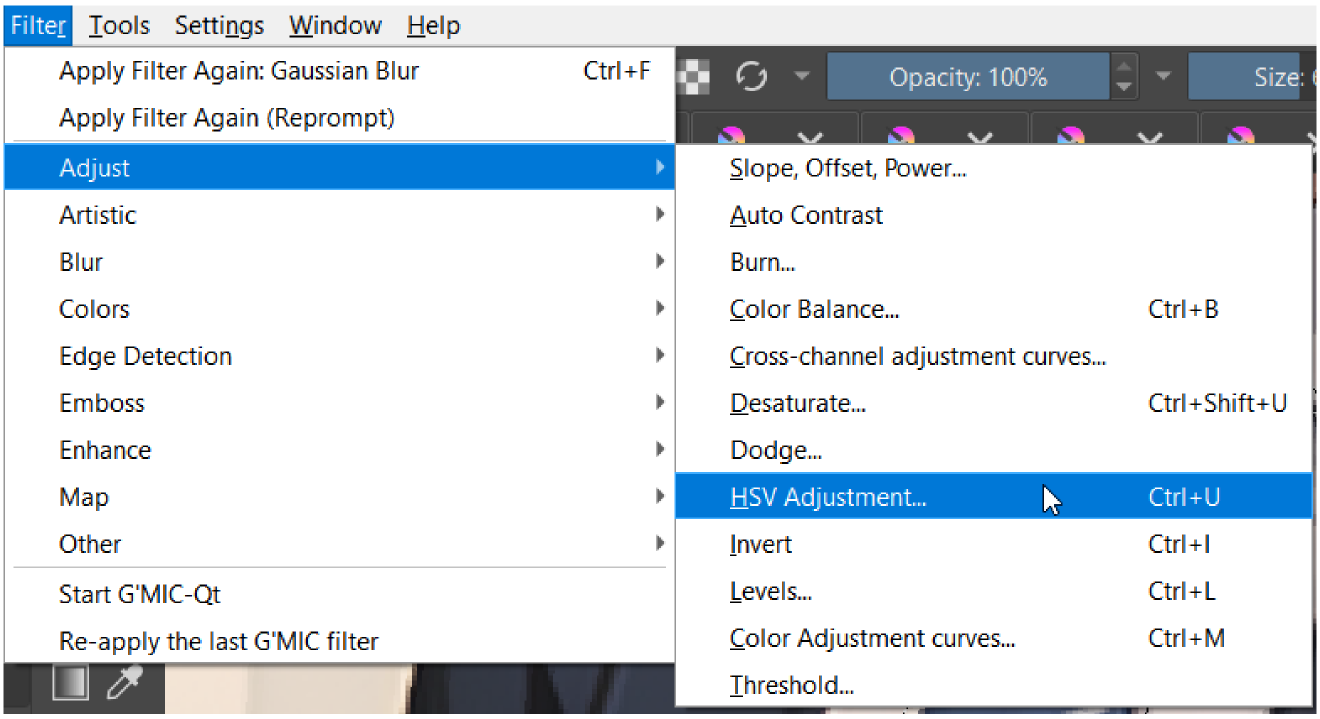
Just click around the tie to select the points. You don’t need to be super-accurate, we can fix this later. Once you’ve completed the selection, just copy the selection by clicking ctrl+c and then paste it using ctrl+v. You should now see a new layer containing your selection:



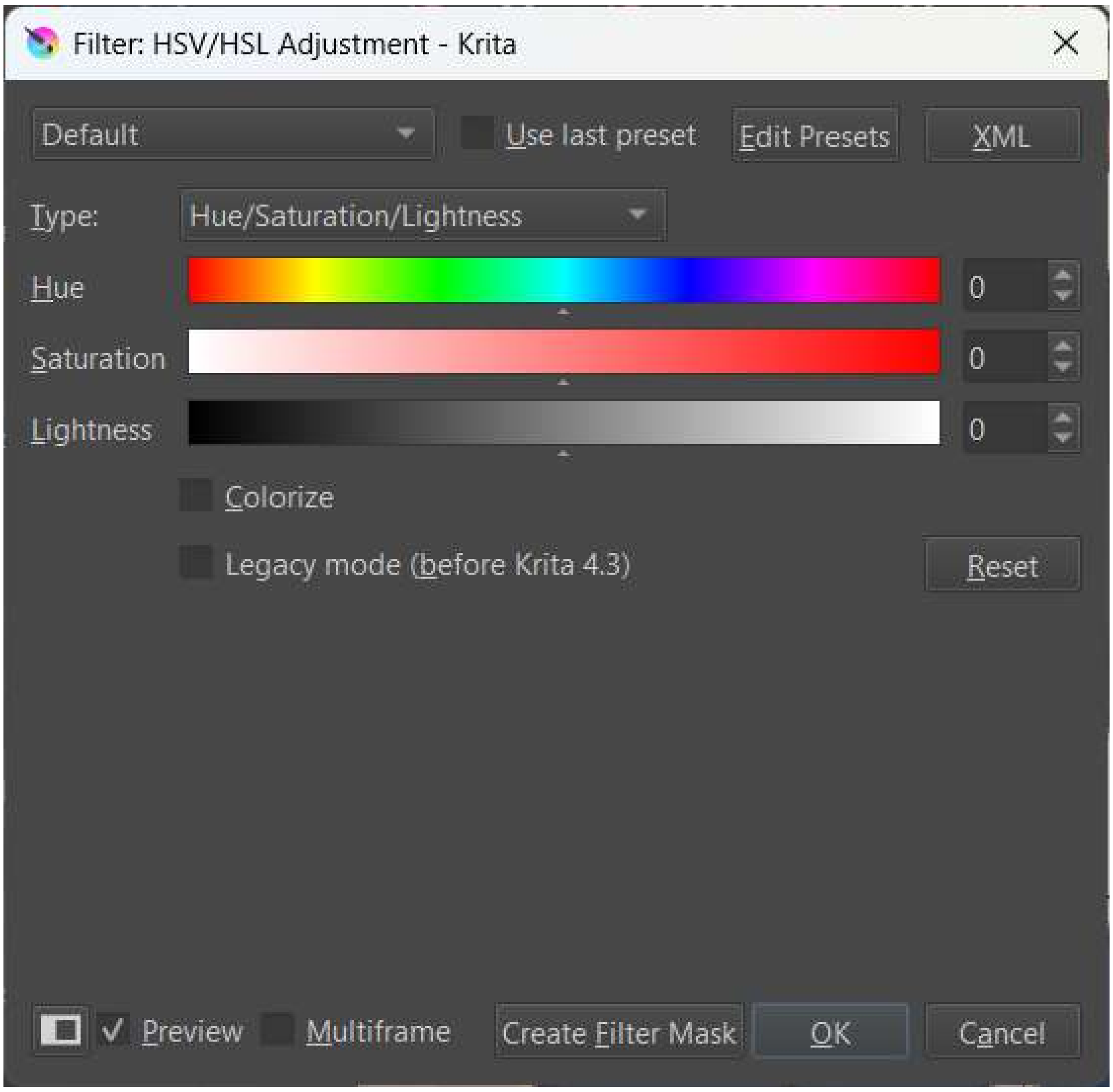
And this is my selection (with the background disabled):



Now that you have the layer selected, go to the toolbar on the top of the screen and choose: Filter -> Adjust -> HSV Adjustment



You should now see a window that looks like this:


Filter: HSV/HSL Adjustment - Krita

Default

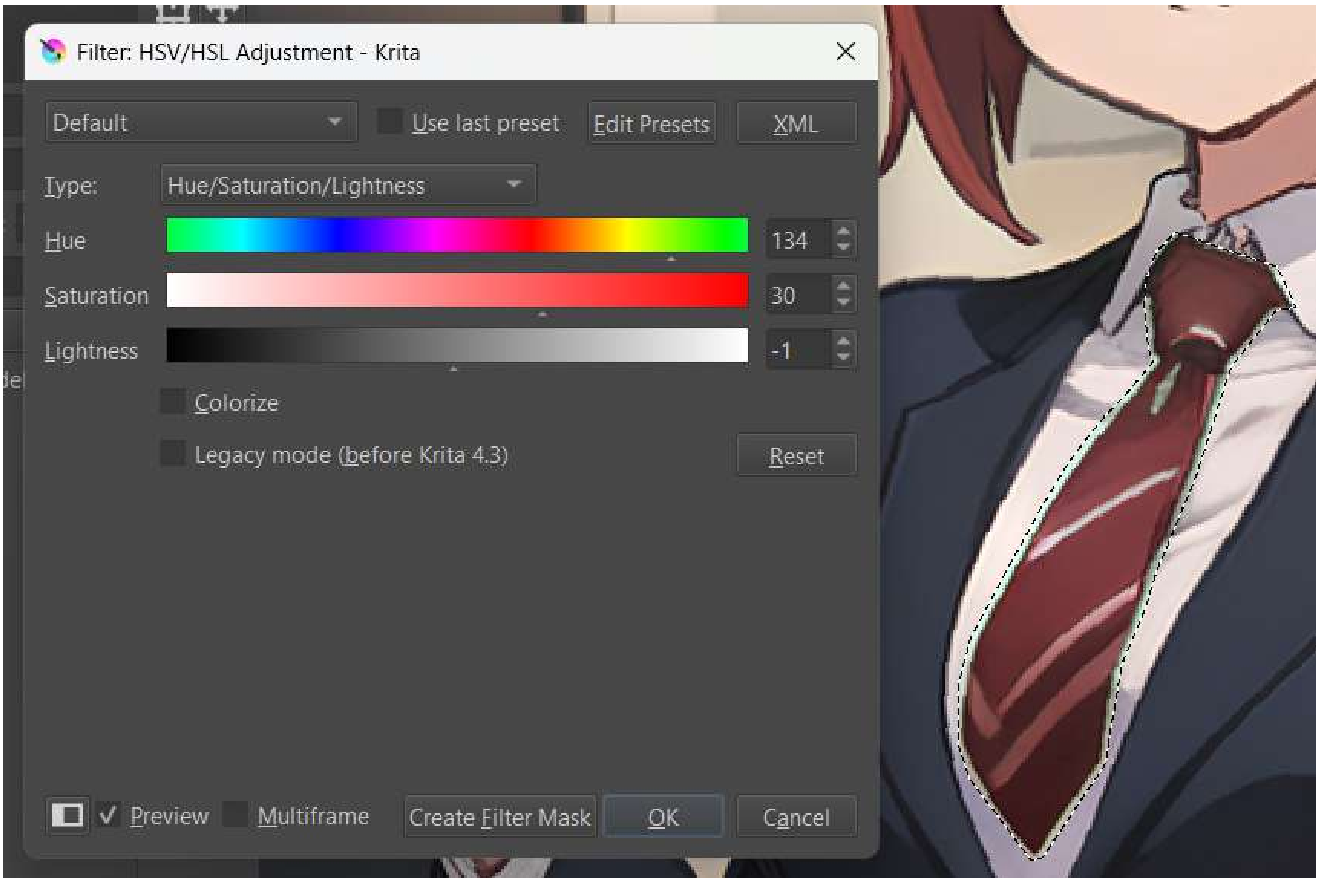
Type: Hue/Saturation/Lightness

Hue 0

Saturation 0

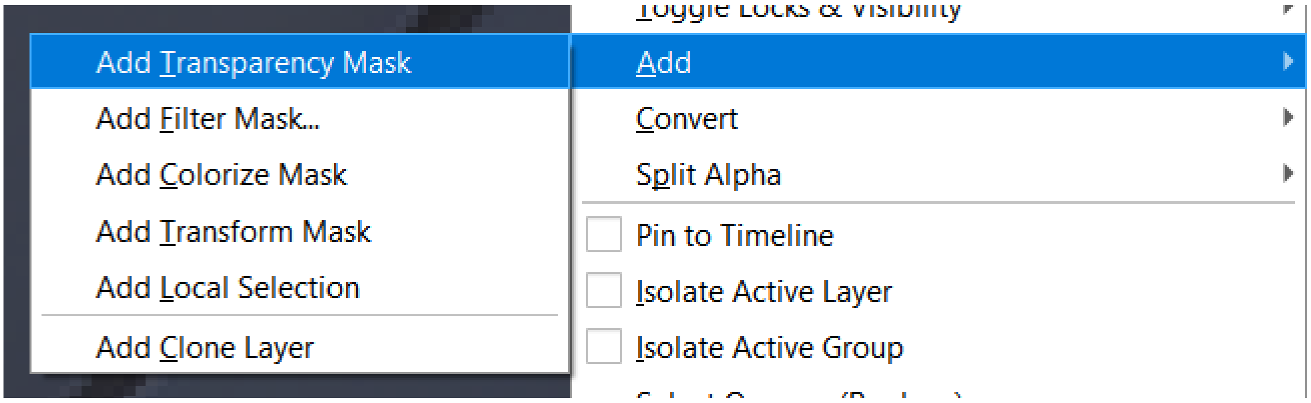
Lightness 0

Now try to change the settings and see if you can change the tie to the red color, then click on “OK”:

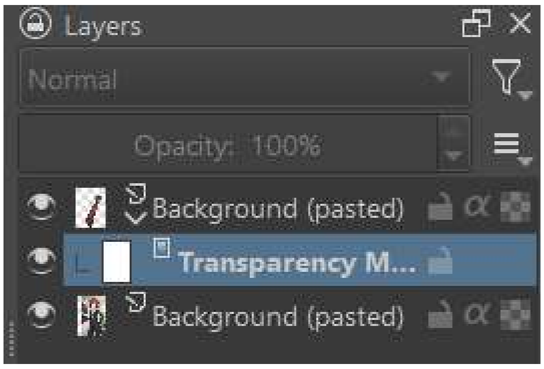


Now we need to clean the border of the tie. Let’s just select all by doing ctrl+a and then pick the brush tool (b).

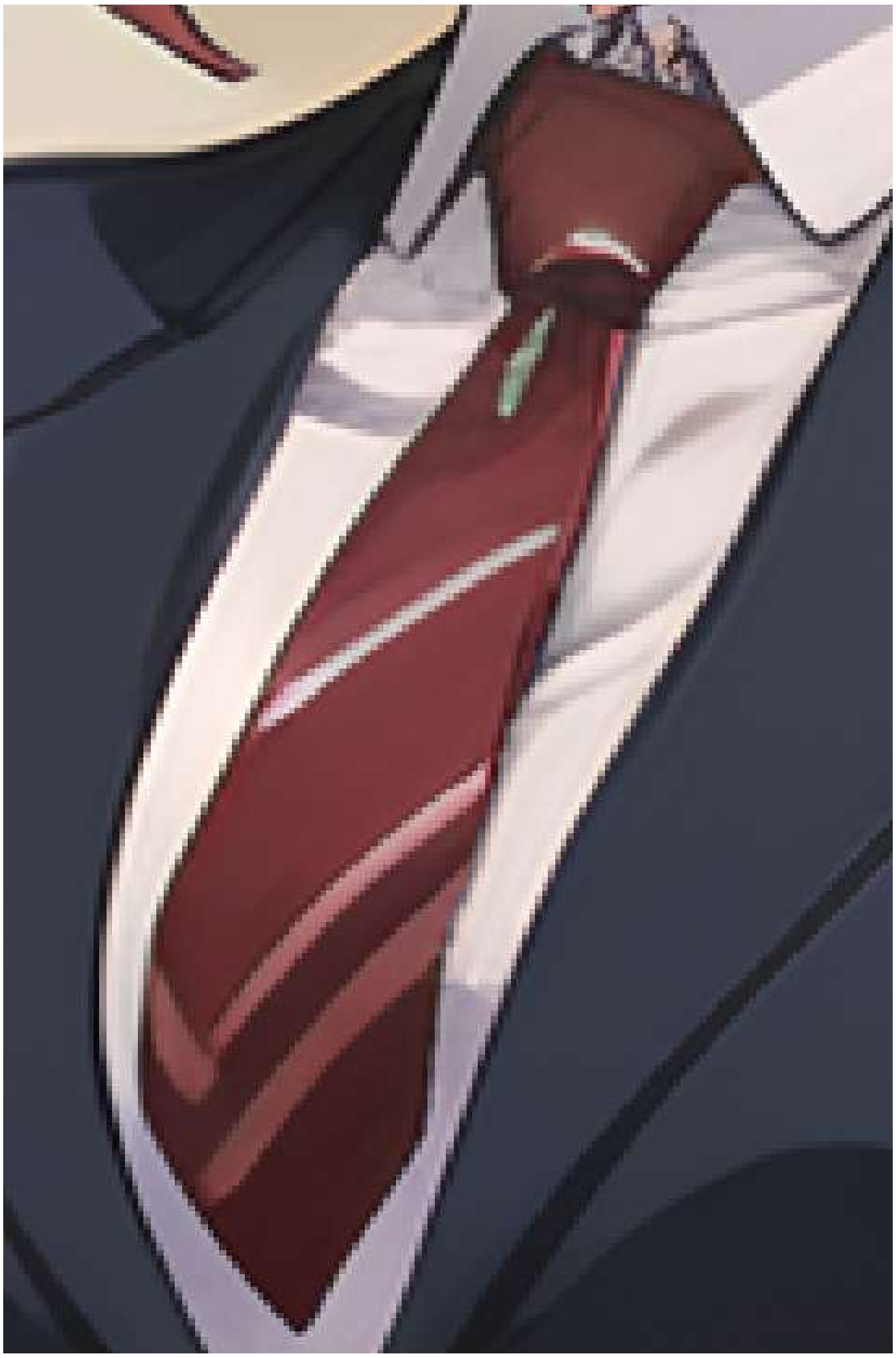
To clean this, I usually create a transparency layer and then use that to remove the incriminated pixels. To do this, right-click on the tie layer and choose: Add -> Add Transparency Mask



You should now see the transparency mask under the layer:

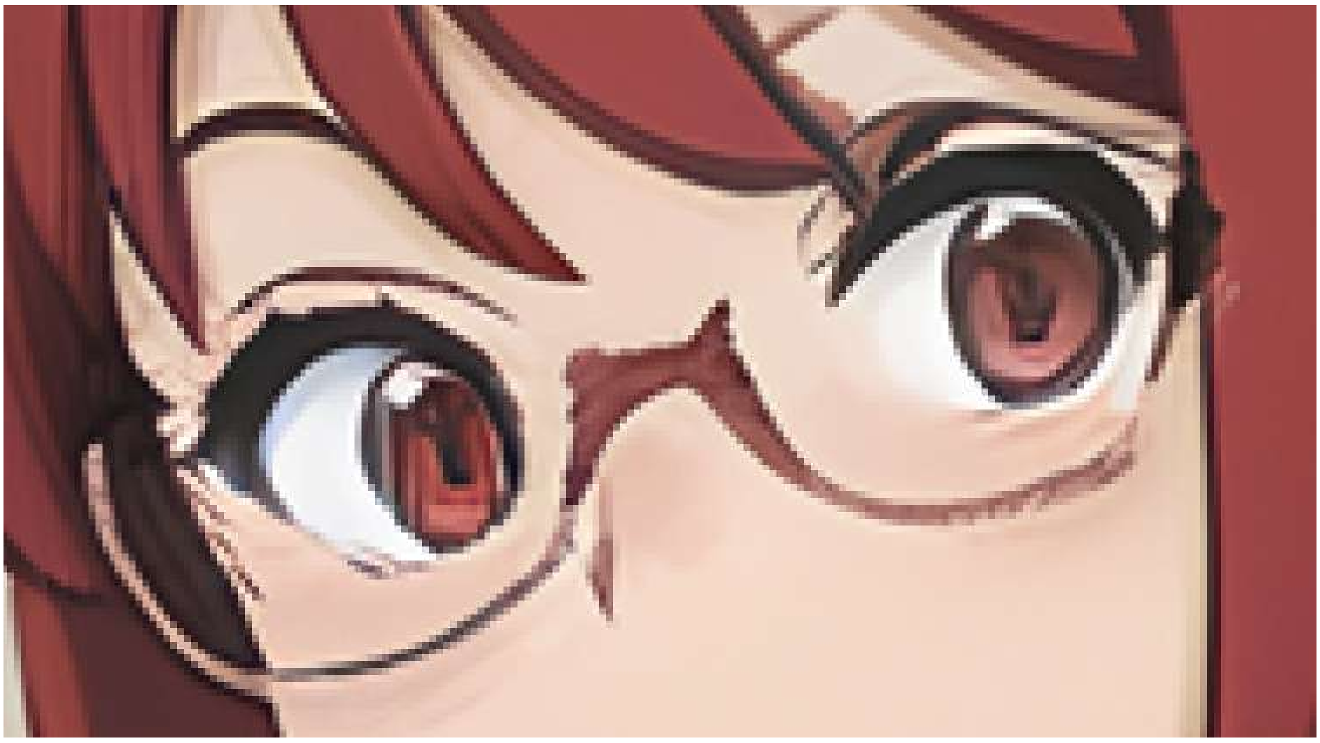


When you have this mask selected, you have only access to greyscale colors. In this mask, white equals to full opacity while white means full transparency. Choose the color black using the selector in the upper right corner and start coloring the border of the tie until you're satisfied with the result:

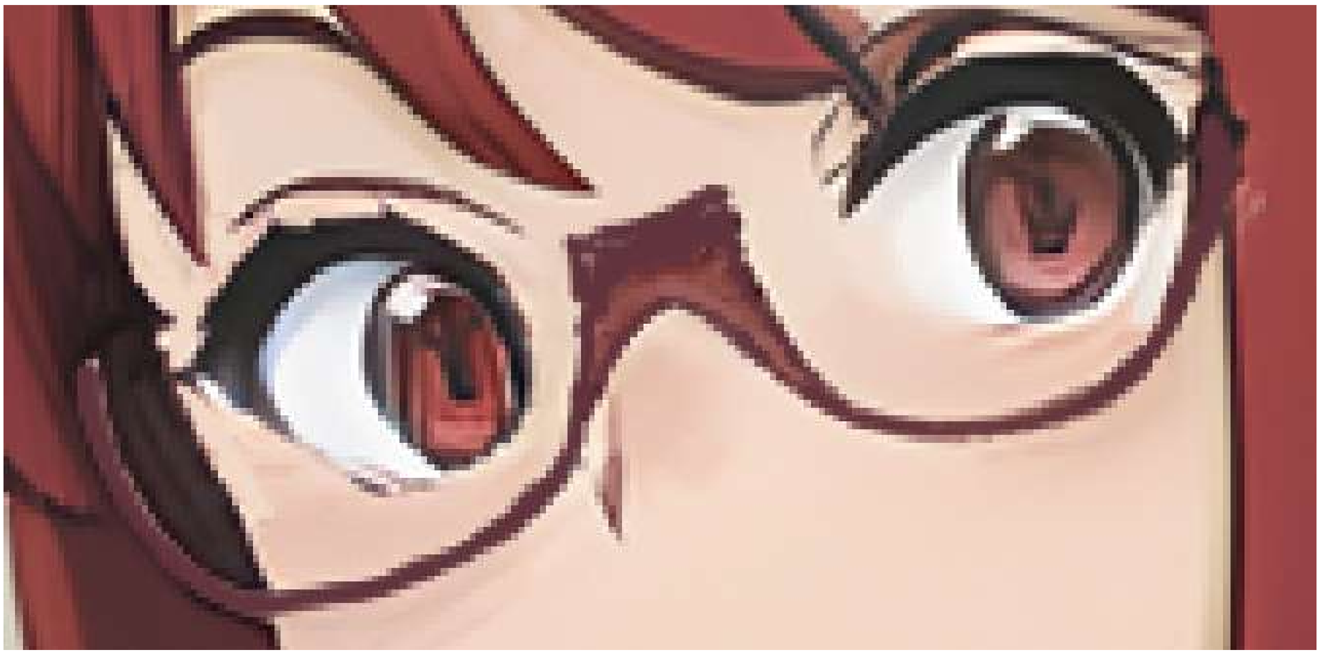


We also need to fix the color of the stripes inside the tie. Select the tie layer (don’t select the transparency layer, otherwise you won’t be able to color the tie) and the use the brush tool to color it:

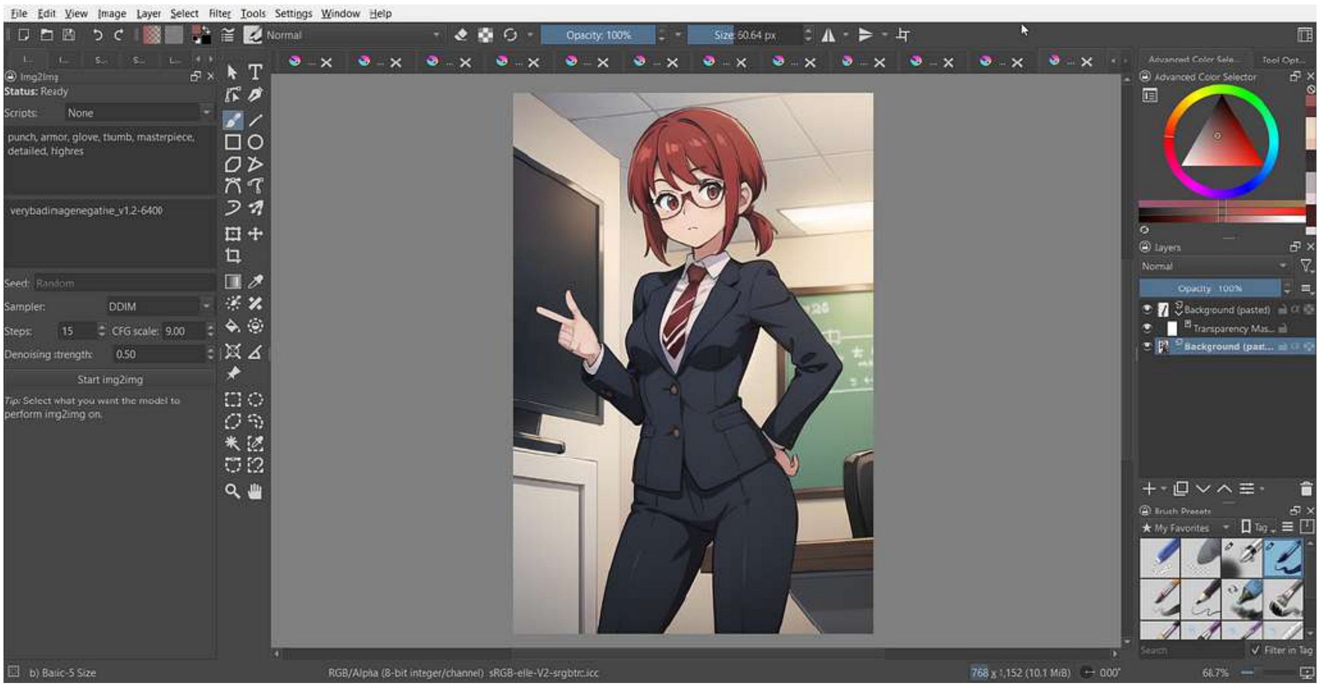
The last thing I’m going to fix are the glasses. After the upscale, they have been distorted a lot:



After the edits:

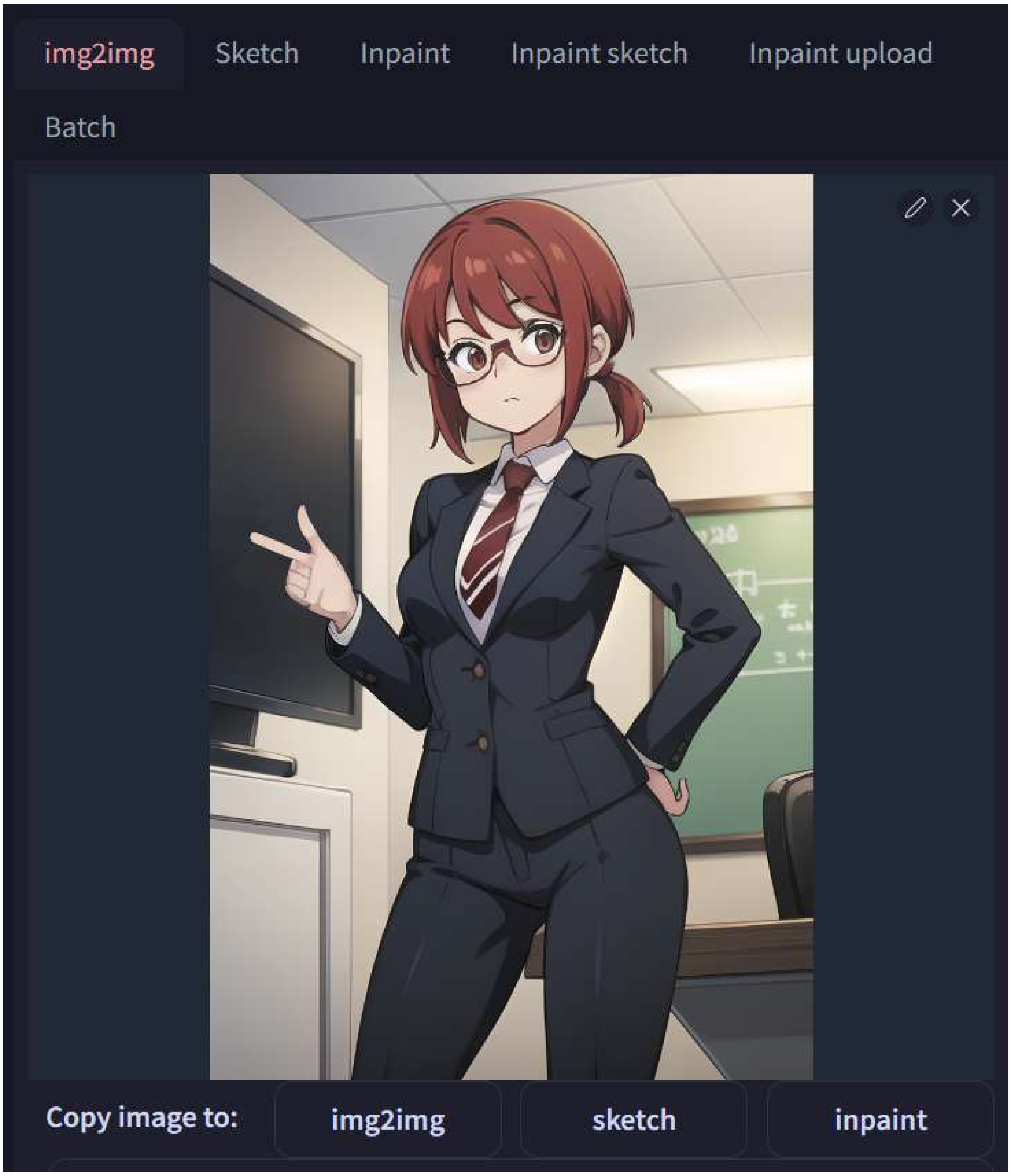


So this should be the final look of the image:

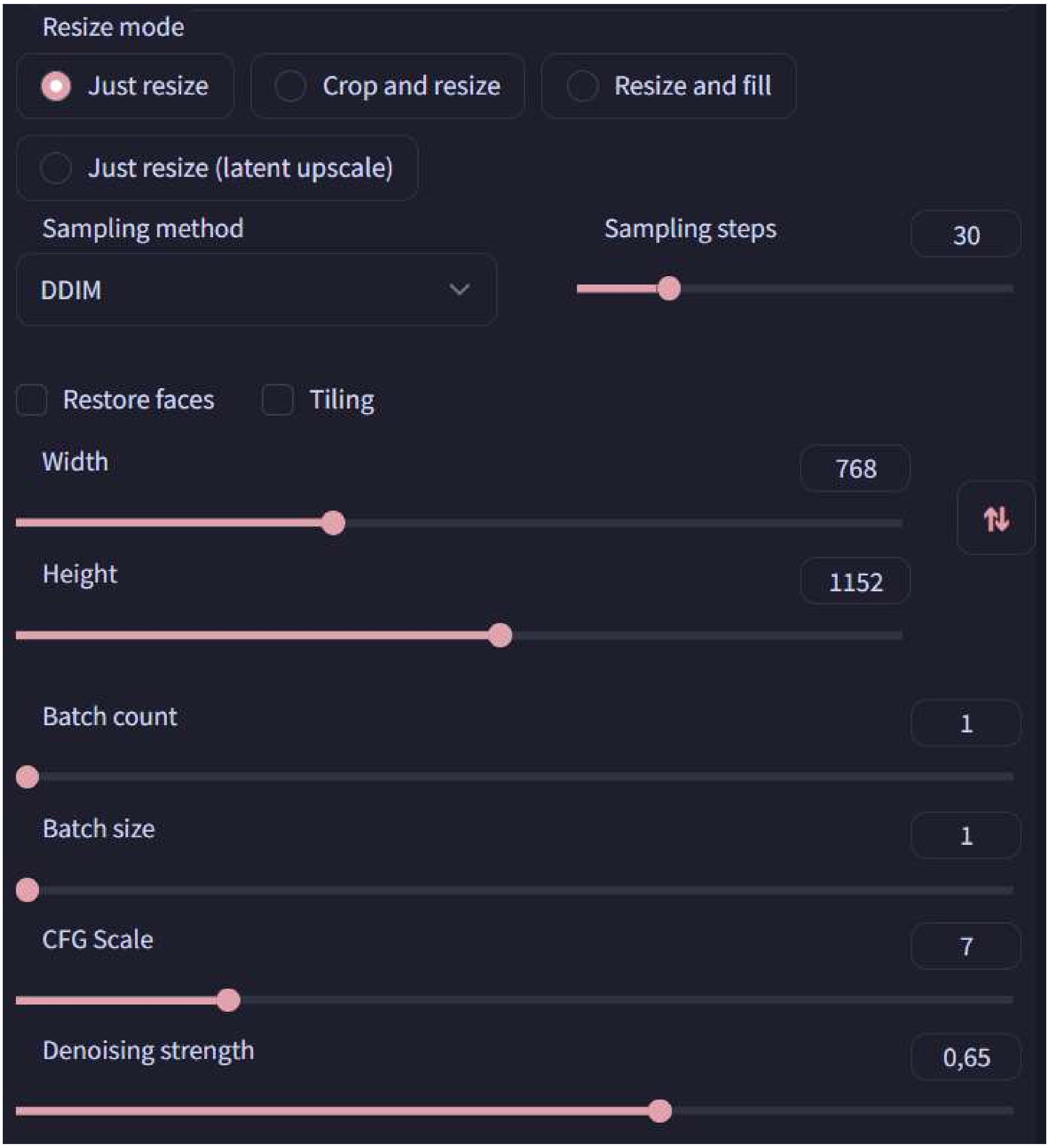


Img2Img
Status: Ready
Scripts: None
punch, armor, glove, thumb, masterpiece, detailed, highres
verybadimagenegative_v1.2-6400
Seed: Random
Sampler: DDIM
Steps: 15 CFG scale: 9.00
Denoising strength: 0.50
Start img2img
Tip: Select what you want the model to perform img2img on.

Now we can select all by using ctrl+a and then copy everything by pressing ctrl+shift+c. Now go back to Stable Diffusion and go to the img2img tab. Once there, paste your image:



Now go back to the txt2img tab and copy over the prompt and negative prompt to the img2img tab. After that we need to change a few of the settings in the img2img tab:


Resize mode
- [✓ Just resize]
- Crop and resize
- Resize and fill
- Just resize (latent upscale)

Sampling method: DDIM

Sampling steps: 30

Restore faces ☐ Tiling ☐

Width: 768

Height: 1152

Batch count: 1

Batch size: 1

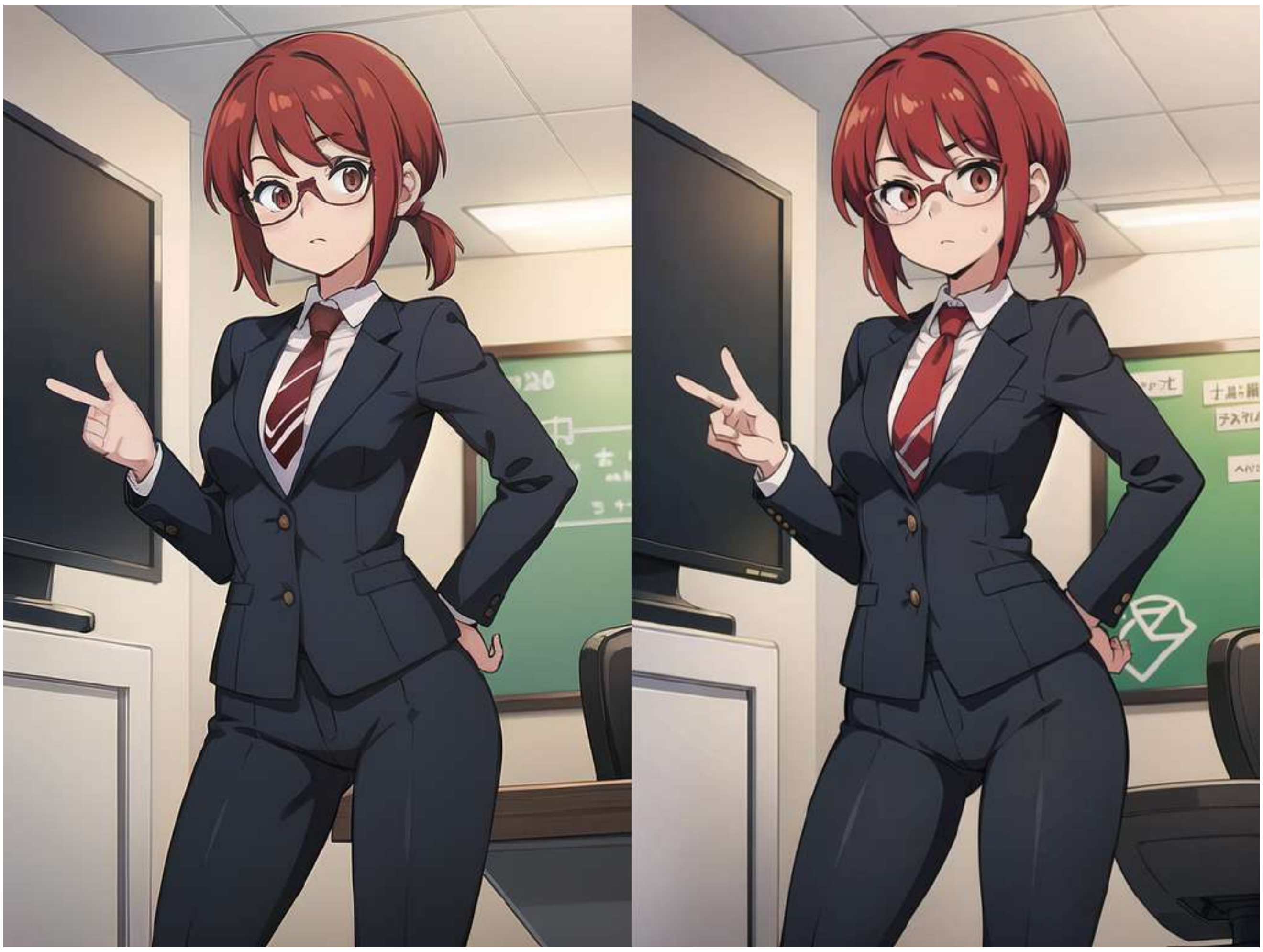
CFG Scale: 7

Denoising strength: 0,65

The most important settings here are:

- Height and Width: these should correspond to the size of the picture we've edited. If your GPU doesn't have enough VRAM, you can try to lower these (make sure to keep the aspect ratio)
- Denoising Strength: this parameter changes how much the input image is changed. With 1 you'll get a completely different image, while with 0 you'll get the same image. I usually use a value between 0.5 and 0.7 depending on how much I'd like to change the starting image.
- Sampling Method: when I work with the img2img or the inpainting process, I usually select DDIM with 30 steps

We can now process the image. You can repeat this step until you get a result you're completely satisfied with, in my case the second generation was pretty good:



On the left the input, on the right the final generation

As you can see by using a "Denoising Strength" of 0.65 the image was modified a little, but the general style is still there.

Going deeper: using LoRAs

This technique is particularly useful when working with LoRAs using a high weight. LoRAs tend to introduce distortion on the pictures and reduce the overall details, so by using this method you can clean the image and get a higher resolution.

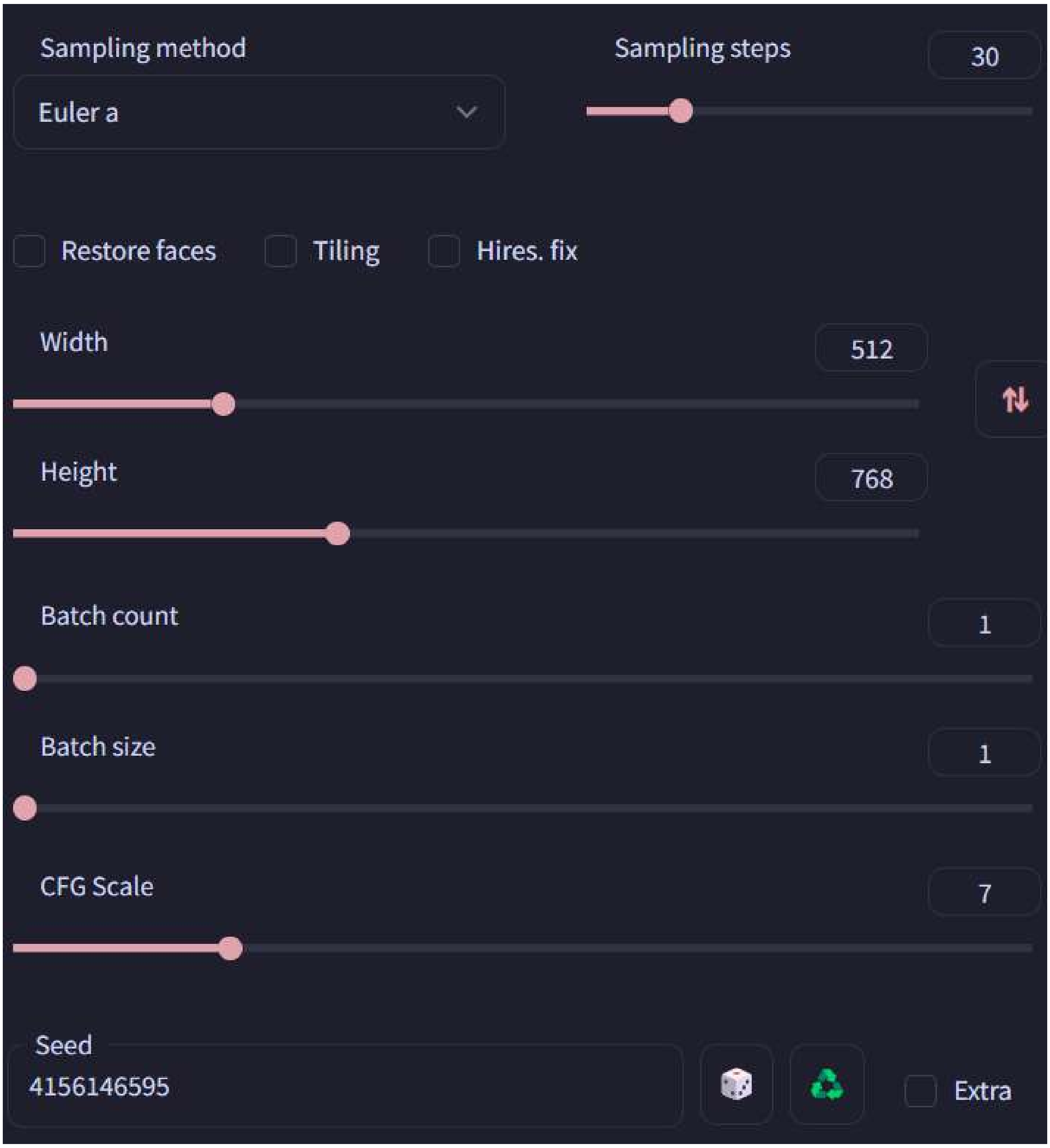
Let's check an example. I'm going to use my 80s fantasy LoRA which is prone to introducing distortion on Anime images (as it was trained on movie posters):

80s Fantasy | Stable Diffusion LORA | Civitai

Let's start by using this prompt and the same settings as before:

girl, standing, armor, fantasy, isekai, chainmail, fighting pose, outdoors, night, stars, aurora, 80s, <lora:80s_fantasy:0.7>

And let's use the following settings:



Sampling method: Euler a
Sampling steps: 30

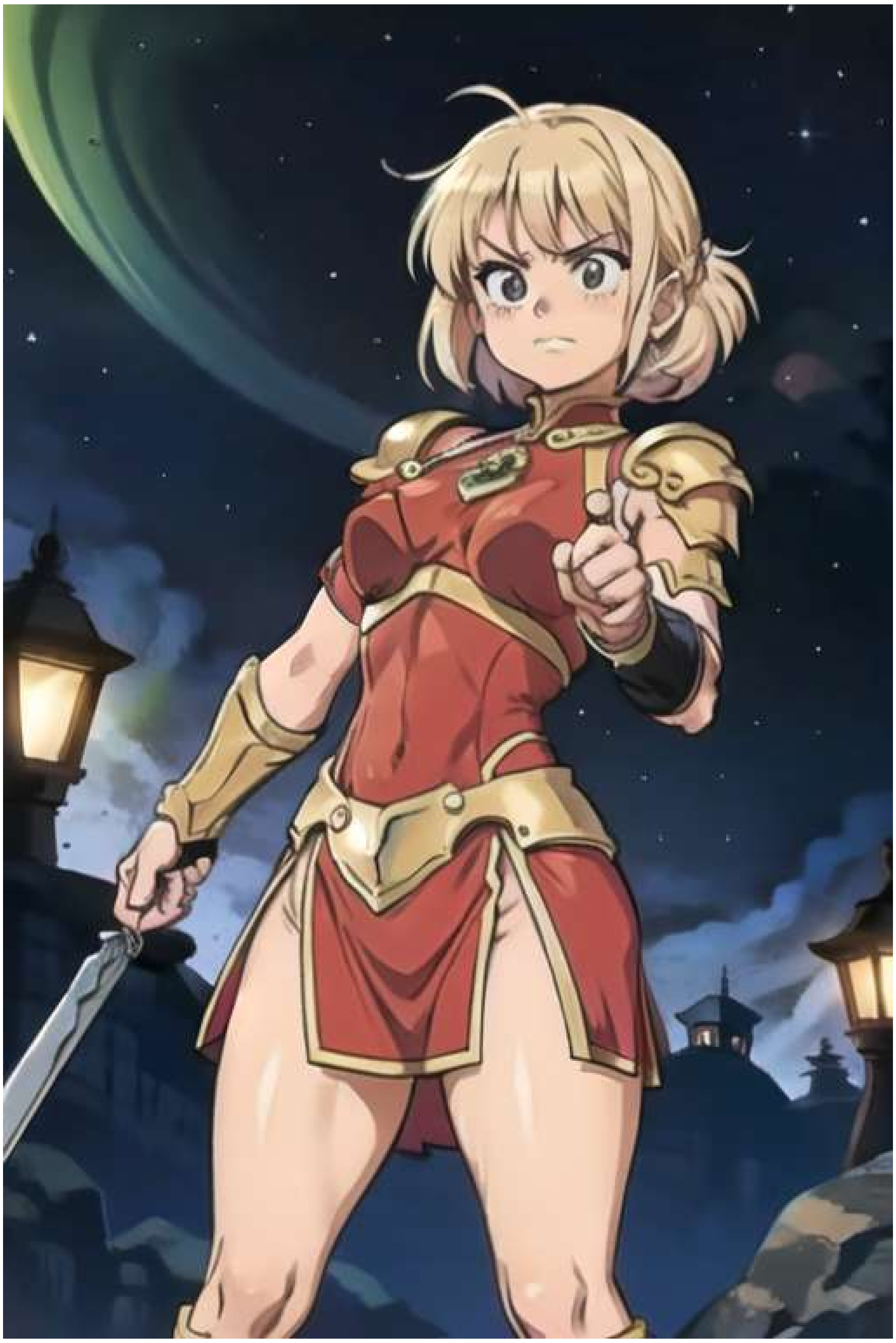
☐ Restore faces ☐ Tiling ☐ Hires. fix

Width: 512
Height: 768
Batch count: 1
Batch size: 1
CFG Scale: 7

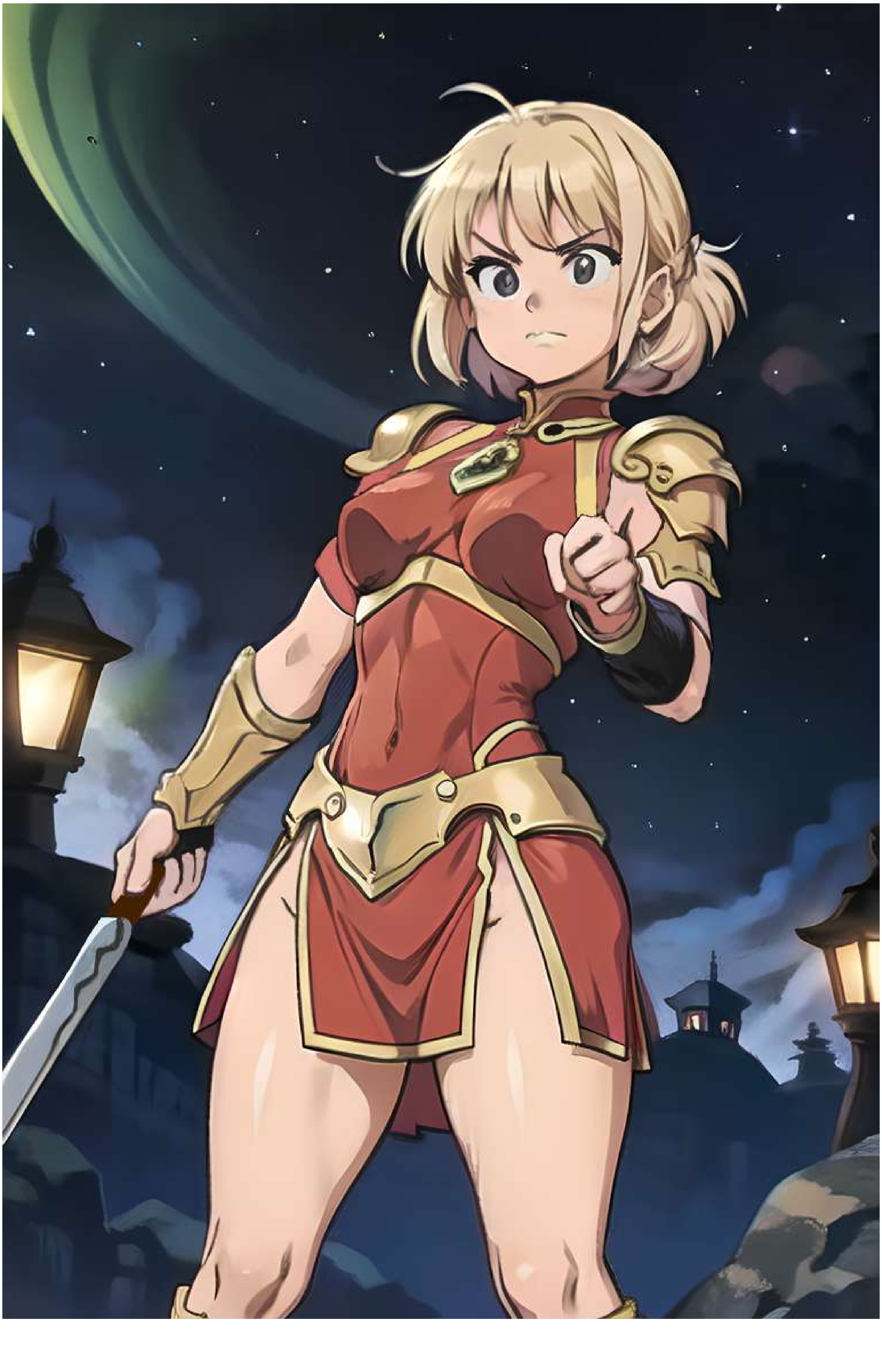
Seed: 4156146595 ☐ Extra

The seed is "4156146595" if you want to copy-paste it.

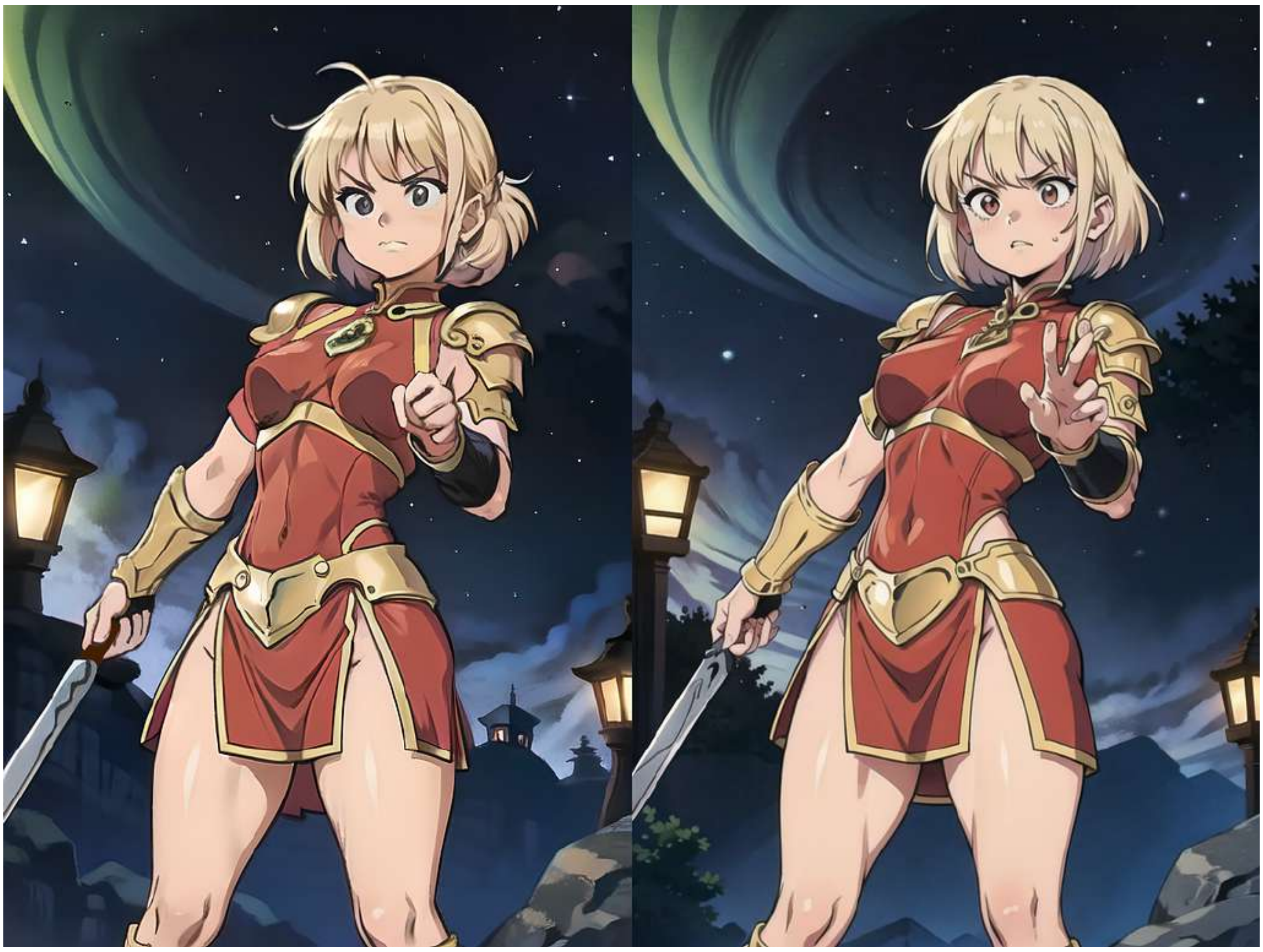
You should get this image:



As you can see, the general style is ok, but the colors are distorted and the overall image is a bit blurry. Let’s start by upscaling it, as we’ve done in the previous example, and then use Krita to clean some of the details. This is the final result:



Now we just need to pass the image to the img2img as we've done in the previous example. In this case you can experiment with the weight of the LoRA to reduce the distortion. This is the final result for me:



As you can see, the style from the LoRA is still there, but the image is a lot more defined and clean.

Stable Diffusion Ultimate Guide pt. 4: Inpainting

In the last 3 chapters, we've seen:

- How to set up the Automatic1111 SD GUI
- How to write effective prompts for a specific model
- The workflow to create sharp high resolution pictures.

Today we are going to have a look at how to inpaint a picture to change a specific part. This is a fundamental step in generating high quality images without having to generate thousands of them.

The model we are going to use for this example is my Mistoon_Ruby v1:

Mistoon_Ruby | Stable Diffusion Checkpoint | Civitai

Requirements

To follow this example, you'll need:

- The Automatic1111 Stable Diffusion UI.
- A GPU that can handle images of 768x1152 size. I'm using a RTX 4080 with 16 GB of RAM. In case you have access to less VRAM, you'll probably be able to work with a lower resolution.
- (Optional, but highly recommended): An image editing program. I'm using Krita (which is an awesome free digital painting application), but you can use your preferred tool (Photoshop, Gimp, Paint.Net…).
- (Optional, but highly recommended): a textual inversion for the negative part. For this example I'm using this **one** and this **one**.
- (Optional): my model Mistoon_Ruby in both the Standard and Inpainting version

Let’s create a picture to inpaint

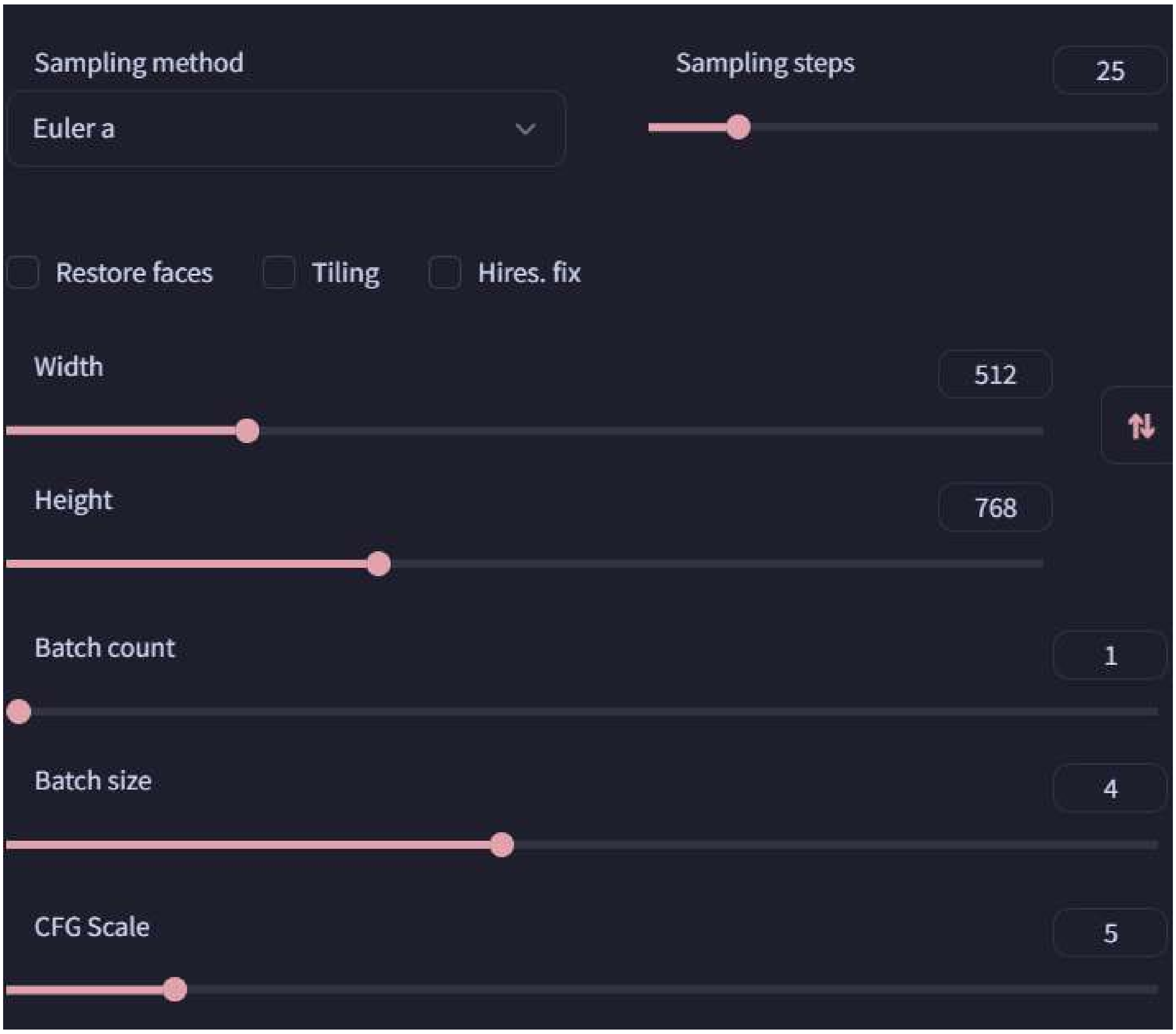
Let’s start simple, I’m going to use the following prompt:

```
girl, standing, desert, rocks, day, sky, clouds, samurai, katana, kimono,
fighting pose, upper body, cinematic
```

For the negative prompt, I’m going to use the textual inversions I’ve indicated in the requirements:

```
verybadimagenegative_v1.3, badhandv4
```

The rest of the settings are the following:



Sampling method: Euler a

Sampling steps: 25

Restore faces ☐ Tiling ☐ Hires. fix ☐

Width: 512

Height: 768

Batch count: 1

Batch size: 4

CFG Scale: 5

The seed is 2364540557.

If you run this prompt with my Ruby model, you should get an image that looks like this:

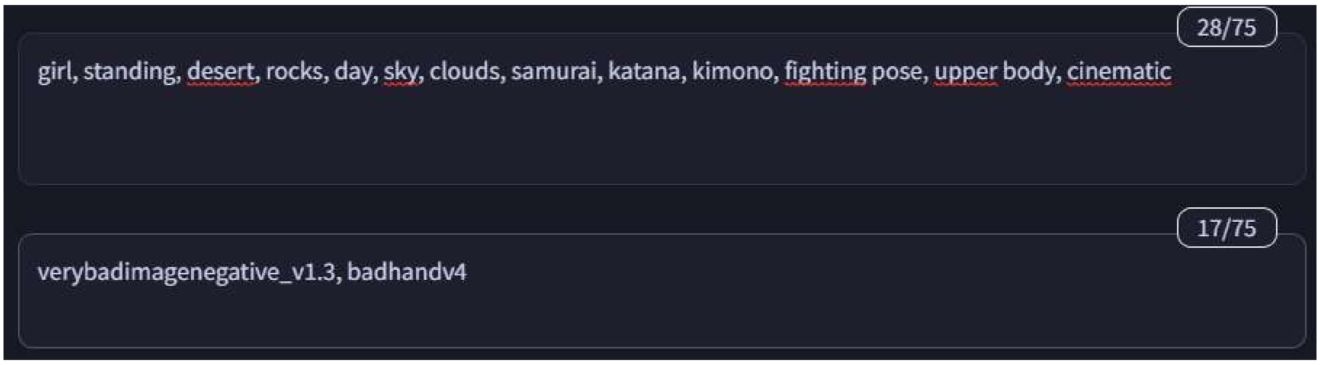
Let’s upscale this using the procedure I’ve explained in the previous chapter.

And you should get something that looks like this:

This is a good starting point, so copy the picture and go to the inpainting tab (or press the send to inpainting button).

Inpainting

So how does the inpainting process work? Let's start by looking at the interface:



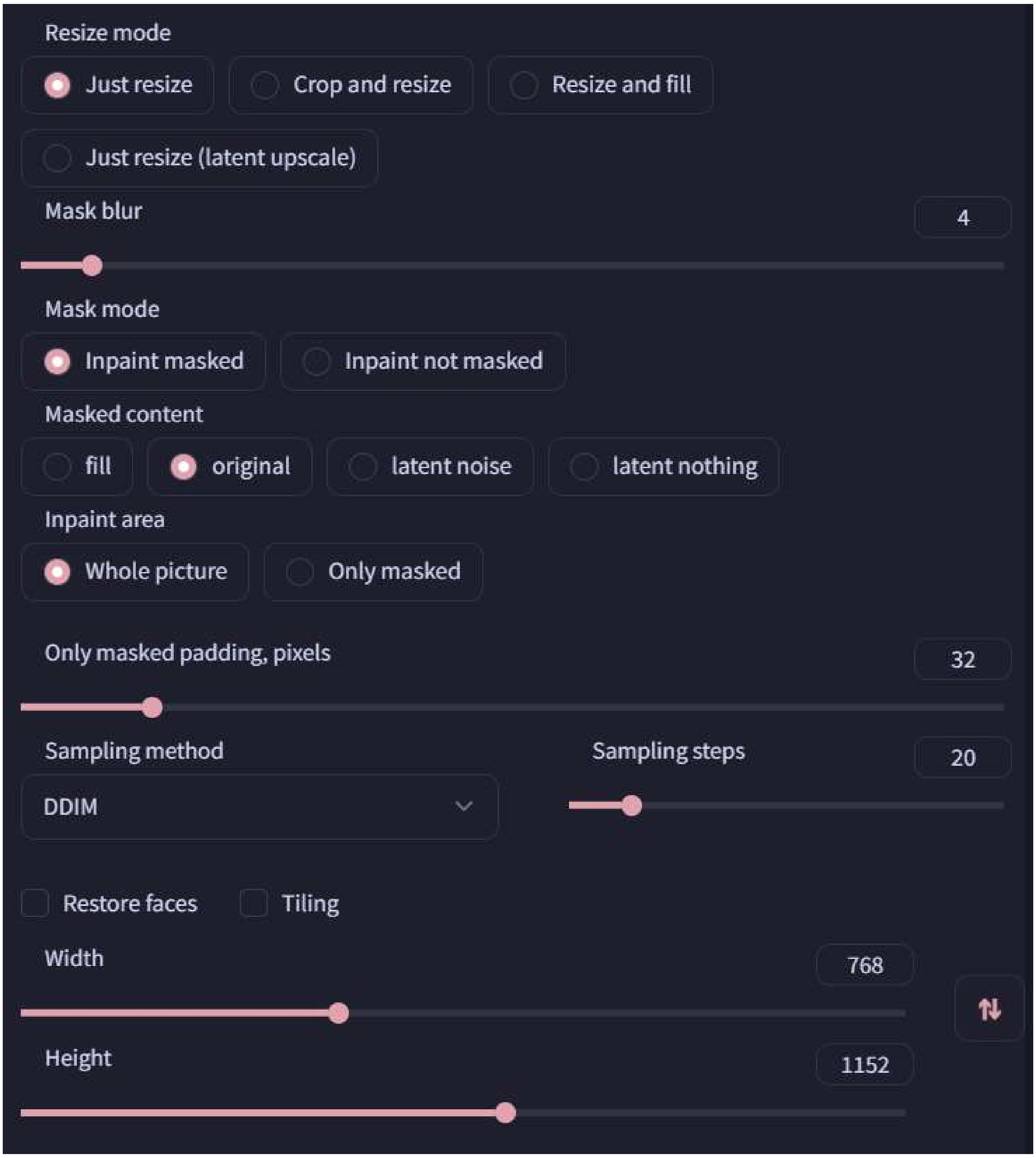
28/75

girl, standing, desert, rocks, day, sky, clouds, samurai, katana, kimono, fighting pose, upper body, cinematic

17/75

verybadimagenegative_v1.3, badhandv4

As usual, you'll find the prompt and negative prompt section. This can work differently based on how you are inpainting the image, but I'm going to explain this later.

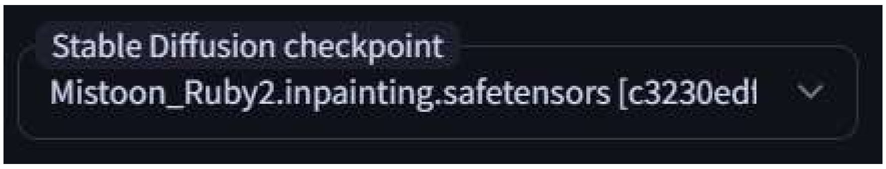

Resize mode
[✓ Just resize] Crop and resize Resize and fill Just resize (latent upscale)
Mask blur 4
Mask mode
[✓ Inpaint masked] Inpaint not masked
Masked content
fill [✓ original] latent noise latent nothing
Inpaint area
[✓ Whole picture] Only masked
Only masked padding, pixels 32
Sampling method DDIM
Sampling steps 20
Restore faces Tiling
Width 768
Height 1152

Let’s check out what each single setting does:

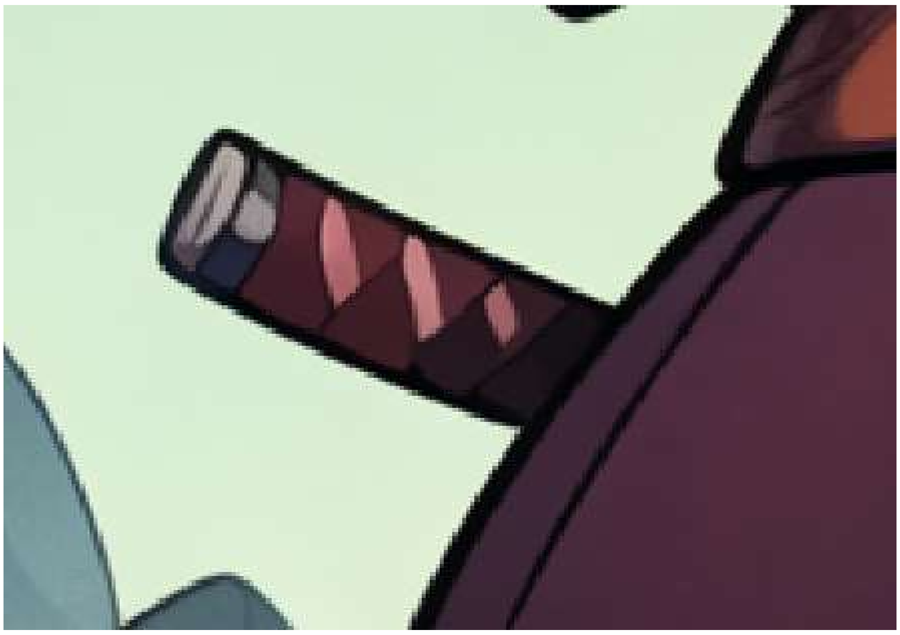
- **Resize Mode**: this setting works the same as the img2img one. It changes how the image is upscaled when necessary.
- **Mask Blur**: this changes how much the inpainting mask is blurred before running the prompt. This helps to avoid sharp edges on the inpainted image.
- **Mask Mode**: this changes how the mask works. “Inpaint masked” changes only the content under the mask you’ve created, while “Inpaint not masked” does the opposite.
- **Masked Content**: this changes the process used to inpaint the image. Pay attention to the fact that some options only works for specific sampling methods (for example, DDIM works only with Original or Fill)

- **Inpaint Area**: this changes which part of the image is inpainted. If you use whole picture, this will change only the masked part while considering the rest of the image as a reference, while if you click on “Only Masked” only that part of the image will be recreated.
- The rest of the settings are the same as the img2img panel. The most important one is the: “Denoising Strength”. This changes how much the generated image will differ from the original one. If you set this to 1 it will generate a completely new image, while 0 while give you the same picture.

One last thing before we start, we need to change the model to the inpainting version. This is not required, but usually improves the quality of the output. To do this, select the inpainting model from the selection above:


Stable Diffusion checkpoint
Mistoon_Ruby2.inpainting.safetensors [c3230edl

Ok, now we have set up everything, let’s start from something simple. Let’s remove the handle of the sword appearing from her back:

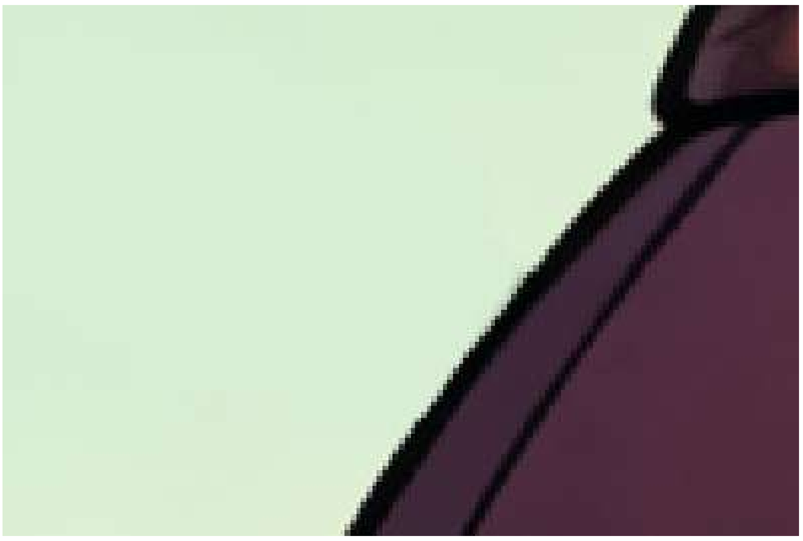


To do this, we have two different ways:

- We can use the inpainting editor already available inside the GUI. This editor has two different options: inpaint which changes the image using the same colors of the original or inpaint sketch which gives you the ability to color and replace a specific part of the picture. I usually don’t use the interface as it’s pretty limited and hard to get the results I want.
- We can use an image editor (I’m going to use Krita), to color and edit the image to match what we have in mind.

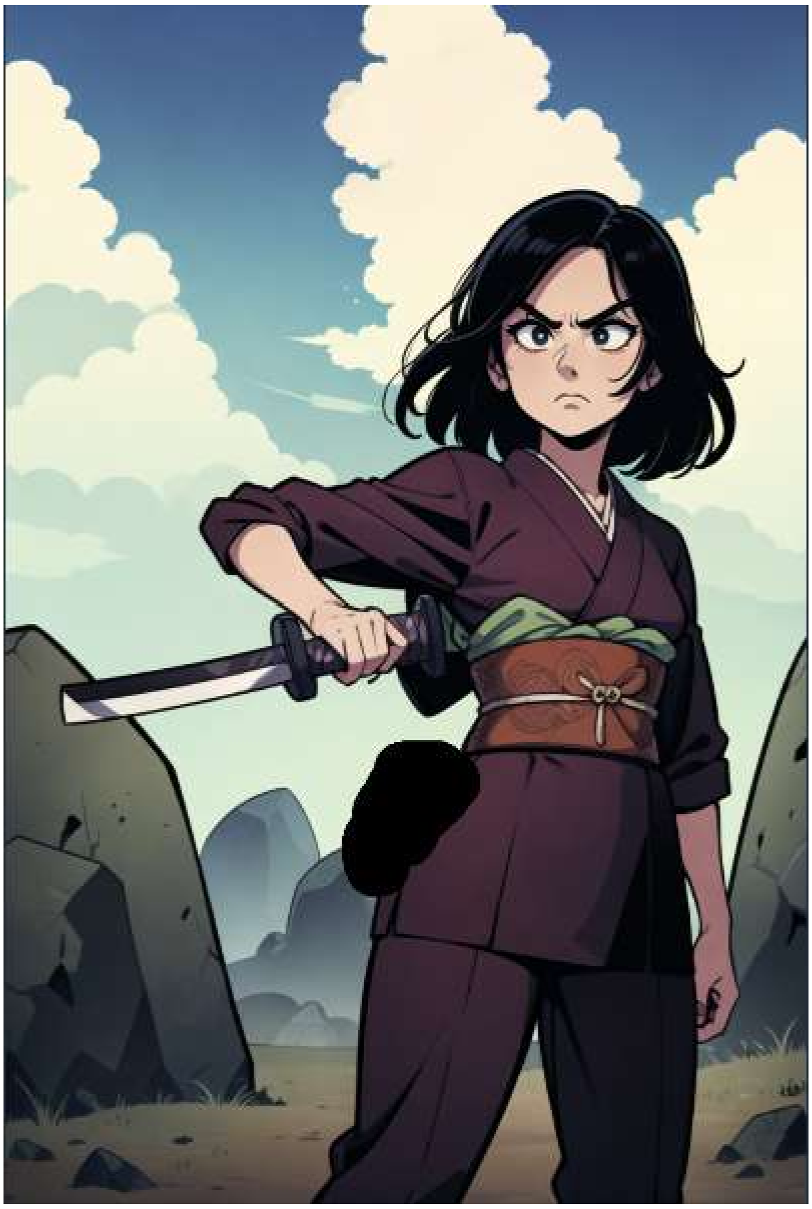
In our case, we are going to choose the second option, so let’s open Krita (or your favorite image editing software).

Once in Krita, we just select the Brush (b) and while keeping “Ctrl” pressed, we select the color of the background. After that, we draw over the handle:

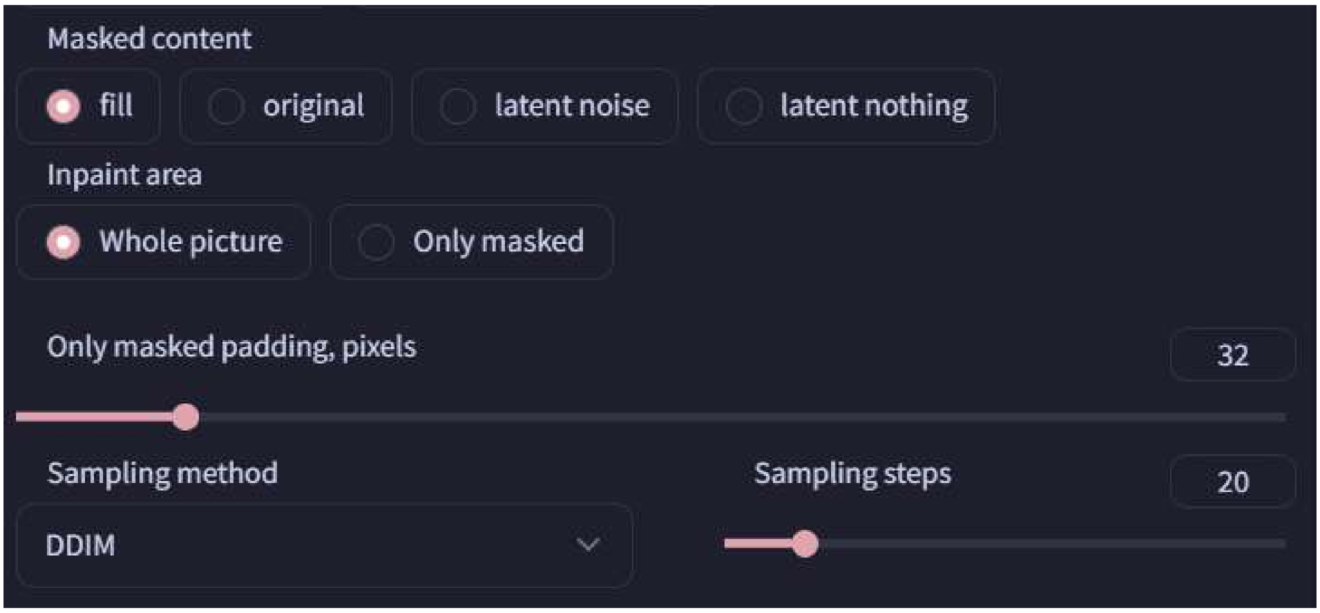


In this case, we could also skip the inpainting part as the result is already good, but let’s just try and see what it does. Copy the picture back to the inpainting tab (select all with ctrl+a and then copy with ctrl+shift+c).

Now, using the inpaint brush, let’s apply a mask over the part we’ve just edited:



Let’s now choose DDIM as our sampler and Fill as our inpaint method:



Masked content

[✓ fill] original latent noise latent nothing

Inpaint area

[✓ Whole picture] Only masked

Only masked padding, pixels 32

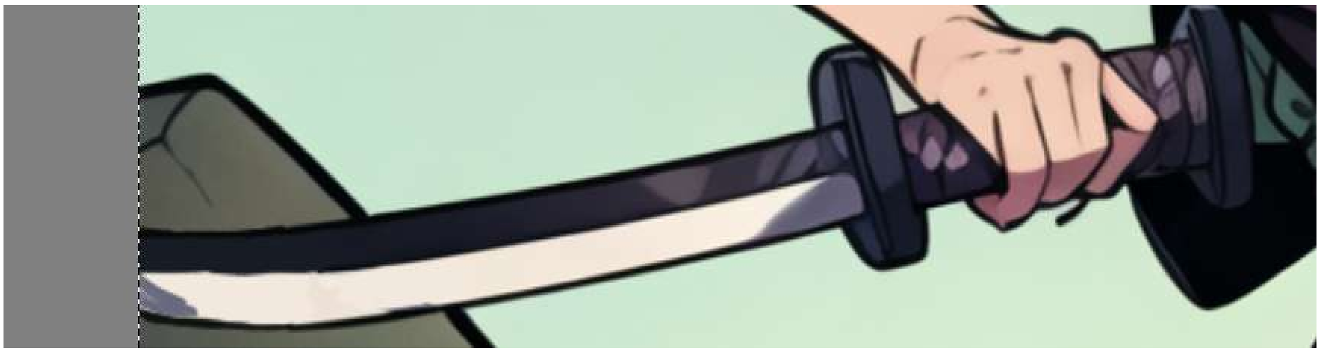
Sampling method: DDIM

Sampling steps: 20

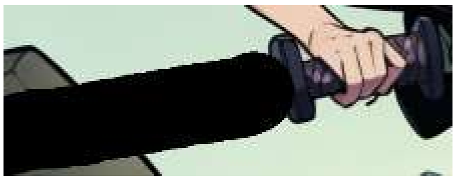
Let’s run the generation:

We now have a new rock in the background! Let’s do something more interesting, and fix the sword. To do this, copy the image and paste it back to Krita (ctrl+shift+n to create a new image or ctrl+v to paste it into a new layer).

Once there, let’s draw the missing part of the blade. When drawing on a picture, I recommend creating a new layer, so you can edit the image in a non-destructive way. You don’t need to be precise, just draw the rough shape using the same colors from the picture:

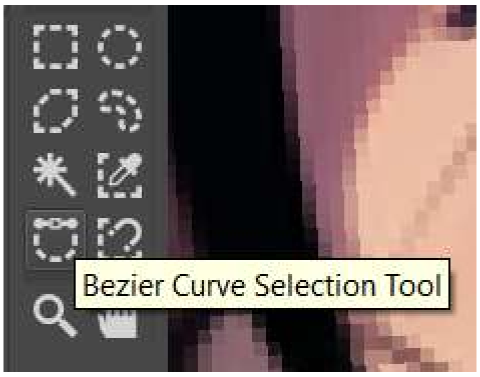


As you can see, it’s far from perfect. Let’s copy this back into SD, and apply a mask over the blade:

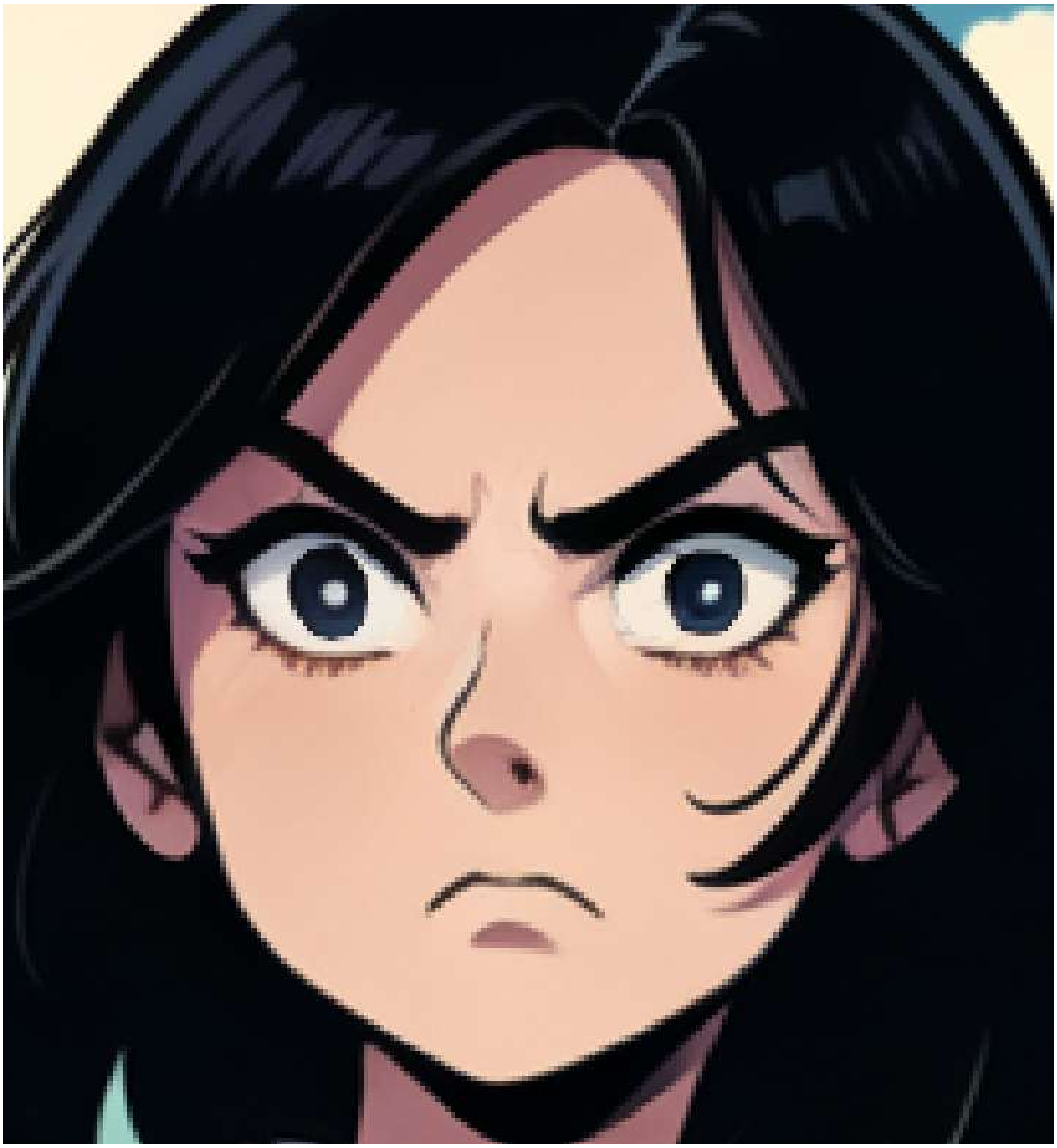


Try experimenting with the denoising strength (in my case, I’m using: 0.65):

Now we can fix the face. As usual, copy the picture back to Krita. Now let’s choose the “Bezier Curve Selection Tool”:

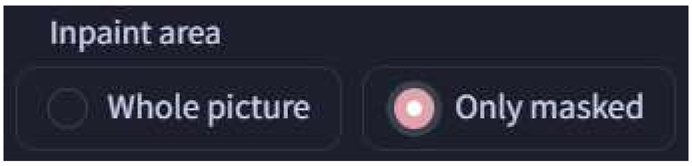


With this, let’s make a selection over the right eye, copy and paste it to a new layer, and then move it a little to the right. You’ll also need to cover the eye on the layer below using a white brush. And while we are at it let’s also clean all the other imperfections, this should be the final result:



Now let’s go back to the UI and paste it back to the picture. Then we can create a mask over the face:

For this step, let’s also use the “Only Masked” option under the “Inpaint Area” section. This will output a sharper and more detailed image:

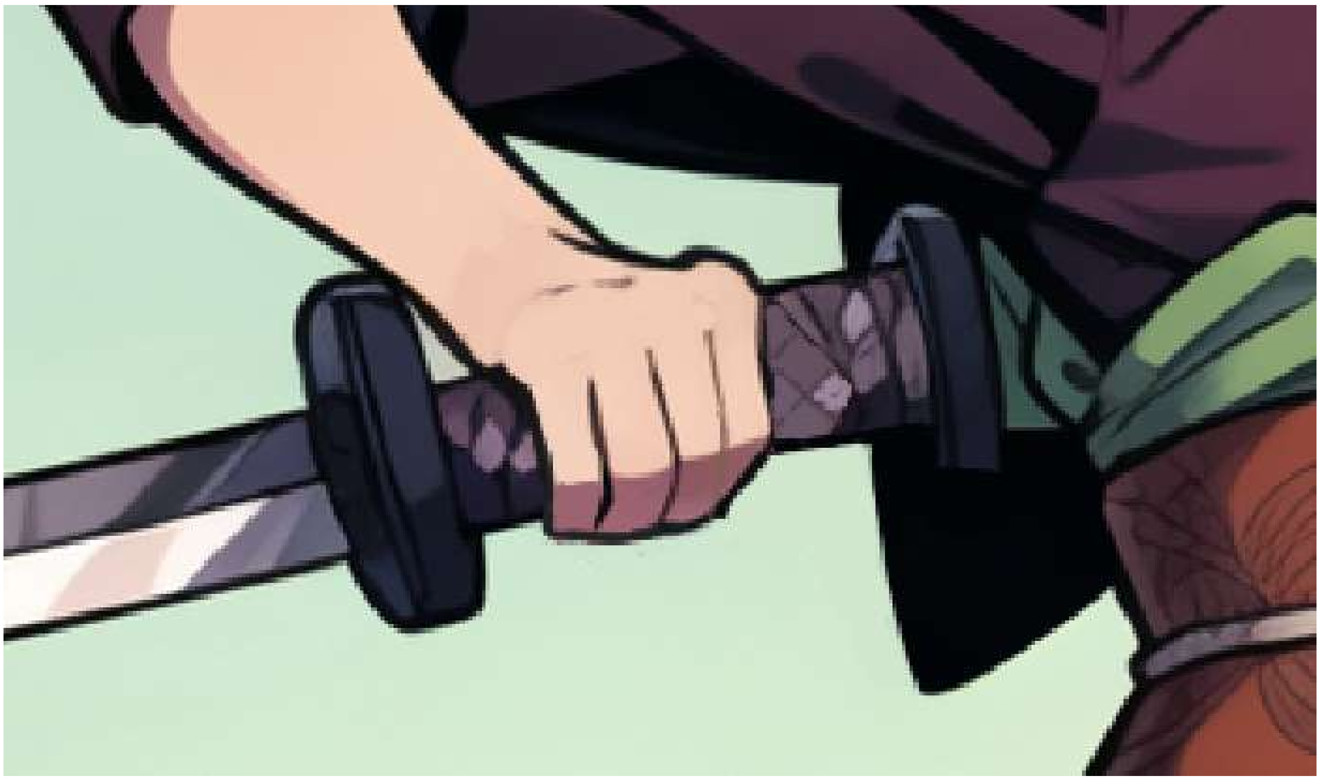


Inpaint area

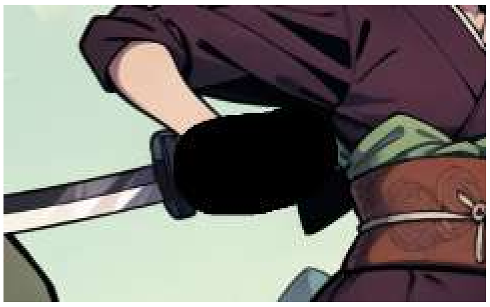
Whole picture [✓ Only masked]

As usual, feel free to play with the settings until you’re satisfied with the result:

Now let’s try fixing the right hand. Hands are usually the most difficult part for any image generation model, so it can be difficult to get it right. Let’s start by roughly fixing the hand in Krita:



Now let’s go back to the UI and create the mask:



Now let’s lower the “Denoising Strength” to 0.5 and run the generation:

The last thing I want to change is the ribbon, this time let’s go to the txt2img tab and create something to add to the image.

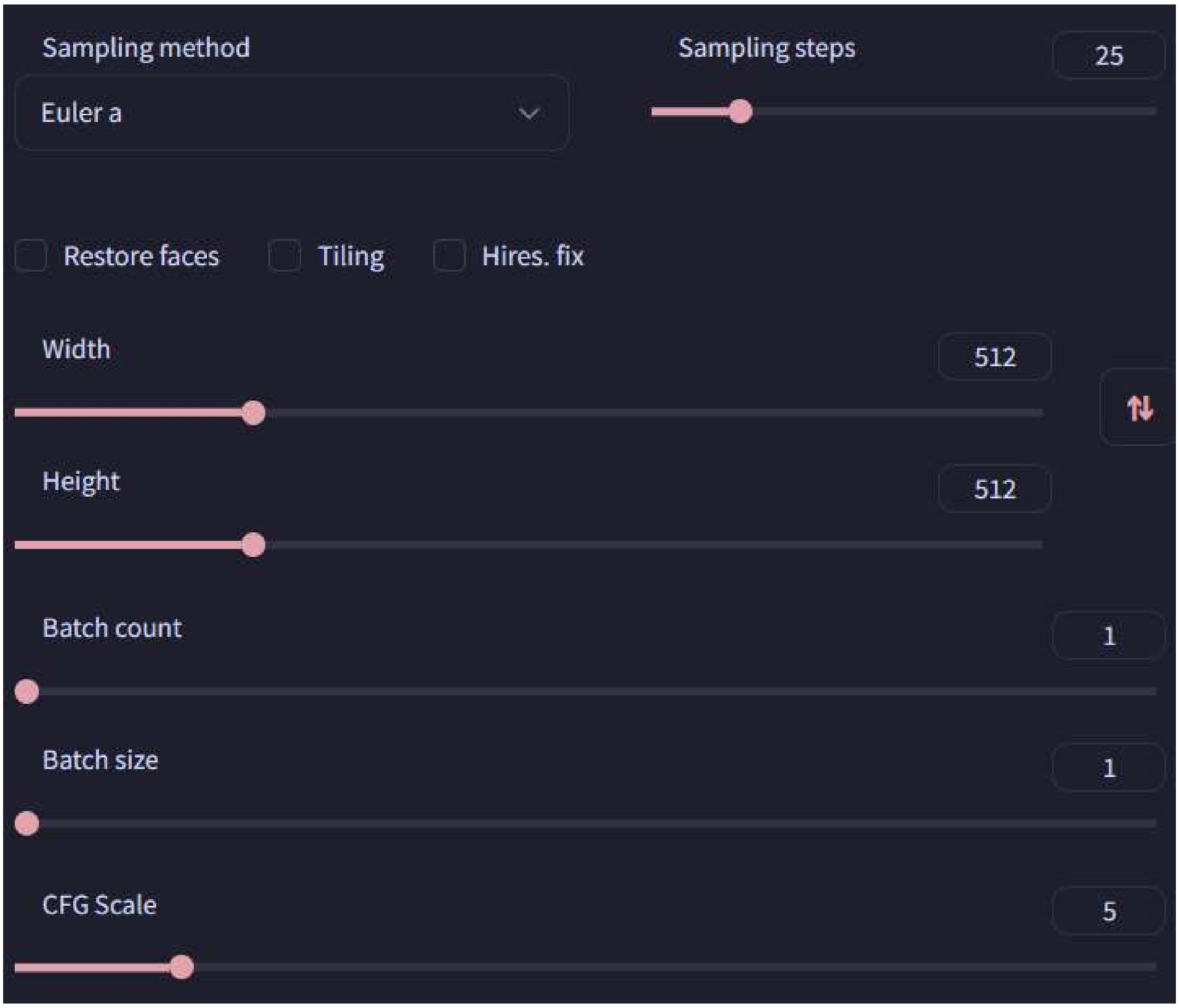
This is the prompt I’m going to use:

```
(gold:1.2), dragon head, icon, (transparent background:1.4)
```

The negative prompt:

```
verybadimagenegative_v1.3, badhandv4, cropped
```

The other settings:



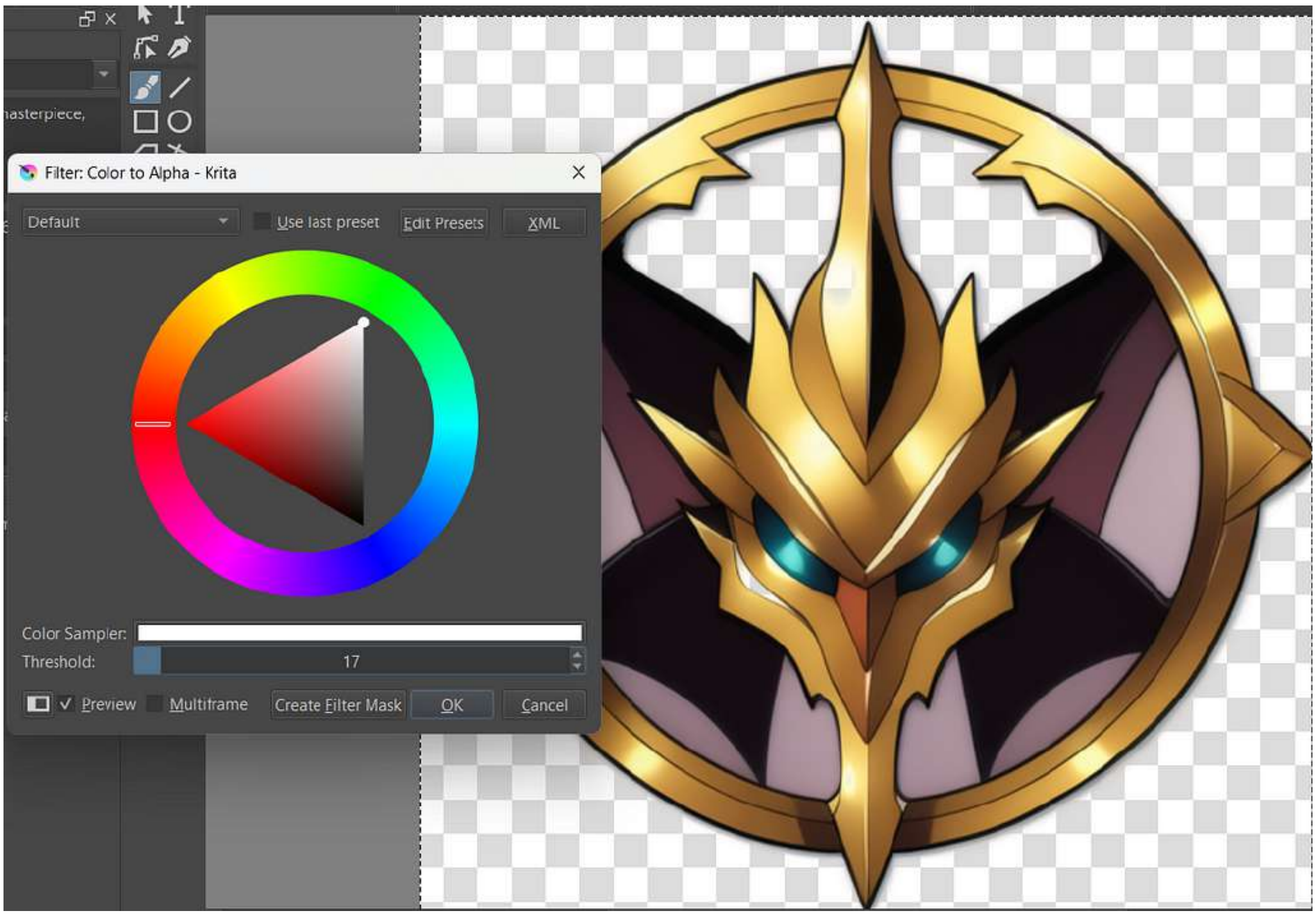
Sampling method: Euler a
Sampling steps: 25
Restore faces ☐ Tiling ☐ Hires. fix ☐
Width: 512
Height: 512
Batch count: 1
Batch size: 1
CFG Scale: 5

The seed is: 2407360043

And this is the final output:

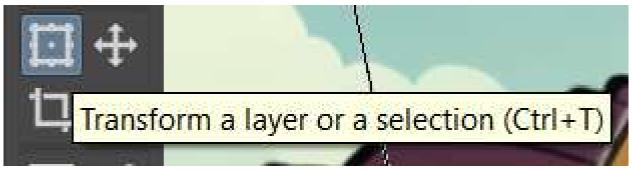
Now let’s copy this into a new image in Krita (you can use ctrl+shift+n).

The first thing we need to do is to remove the white background. To do this, go to Filters → Colors → Color To Alpha. Change the threshold to remove the background and press: “OK”:

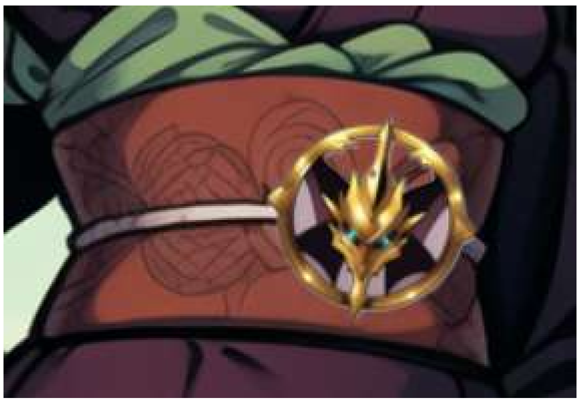


Color Sampler:
Threshold: 17

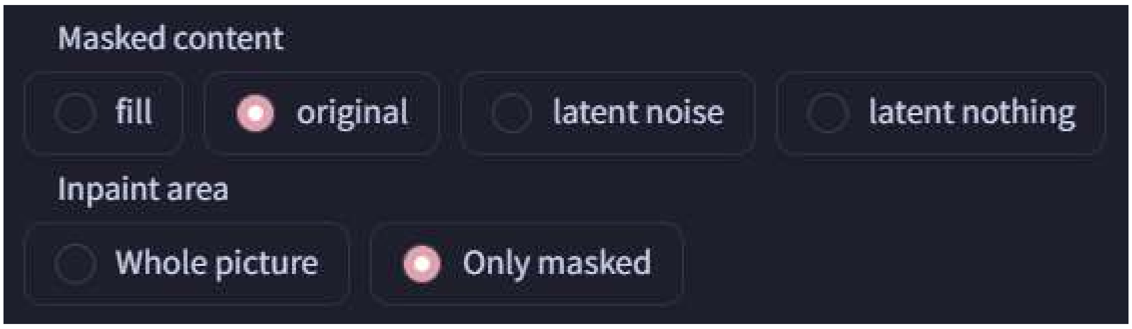
Now copy the image and paste it back to the previous one. Once there, select the “Transform” tool from the sidebar:



Now resize the picture to fit the image:



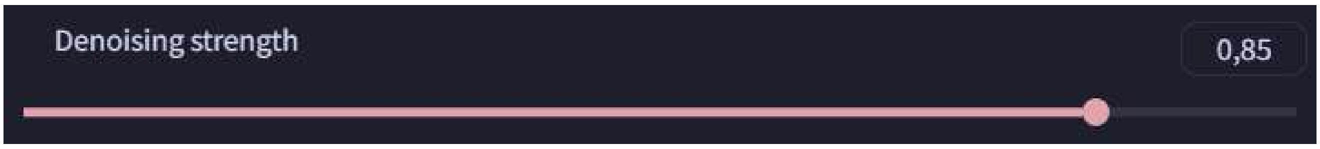
Now copy back the image to the SD UI and create the mask. After that, copy the prompt from the txt2img tab into the inpaint panel and use the following settings:


Masked content

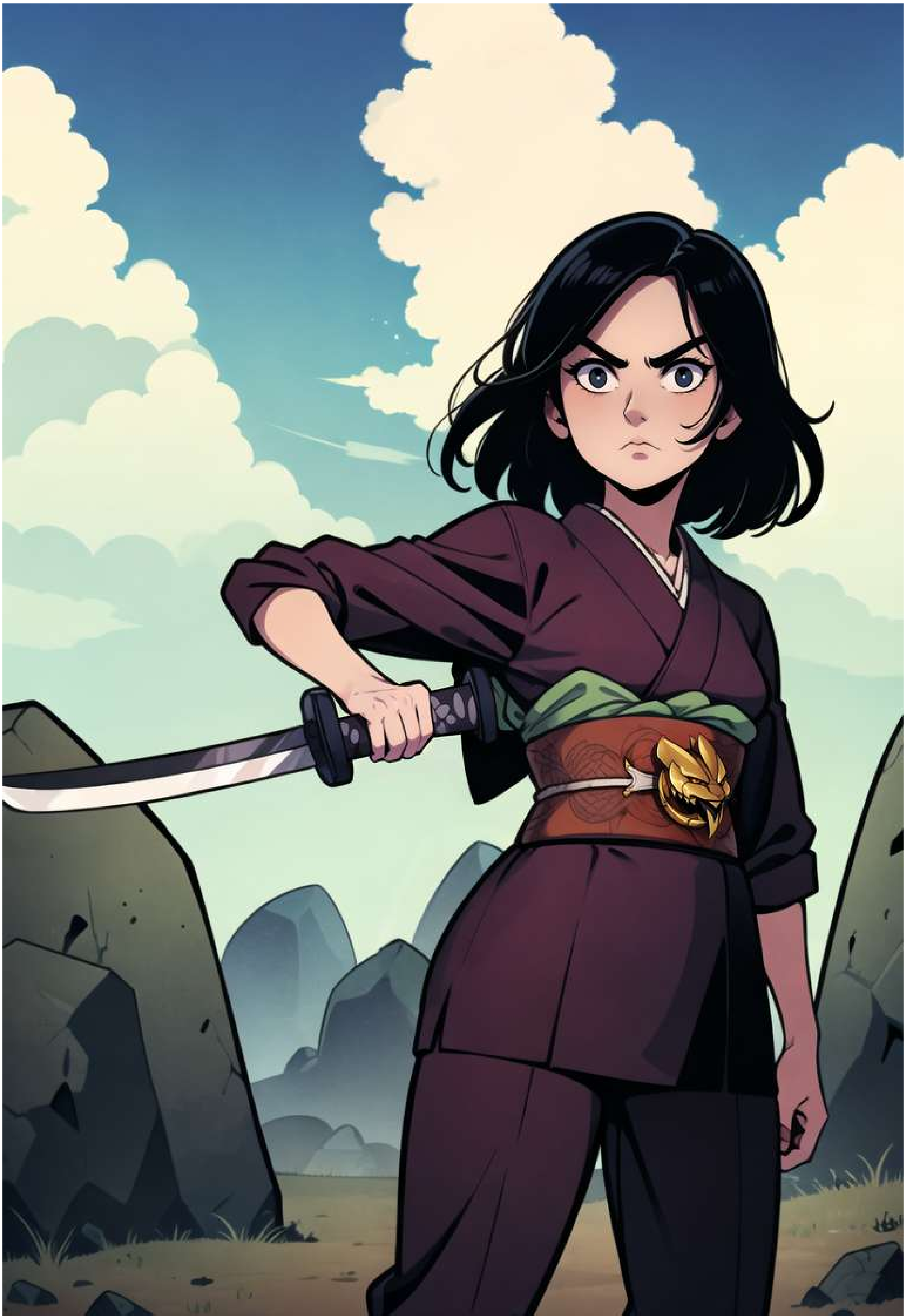
fill [✓ original] latent noise latent nothing

Inpaint area

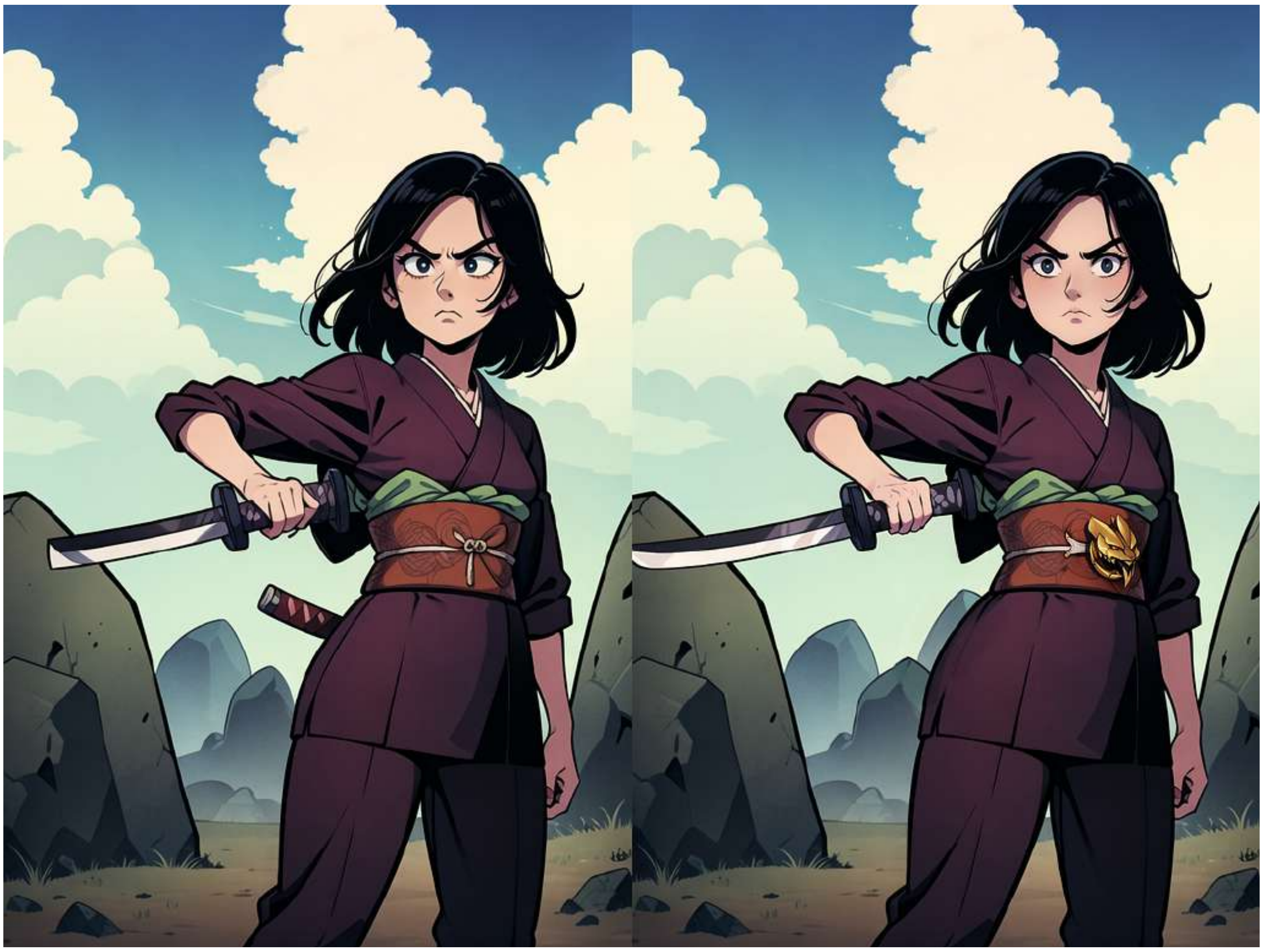
Whole picture [✓ Only masked]


Denoising strength 0,85

Now you should have something like this:



So let’s have a look at how we’ve changed the image from the original one:



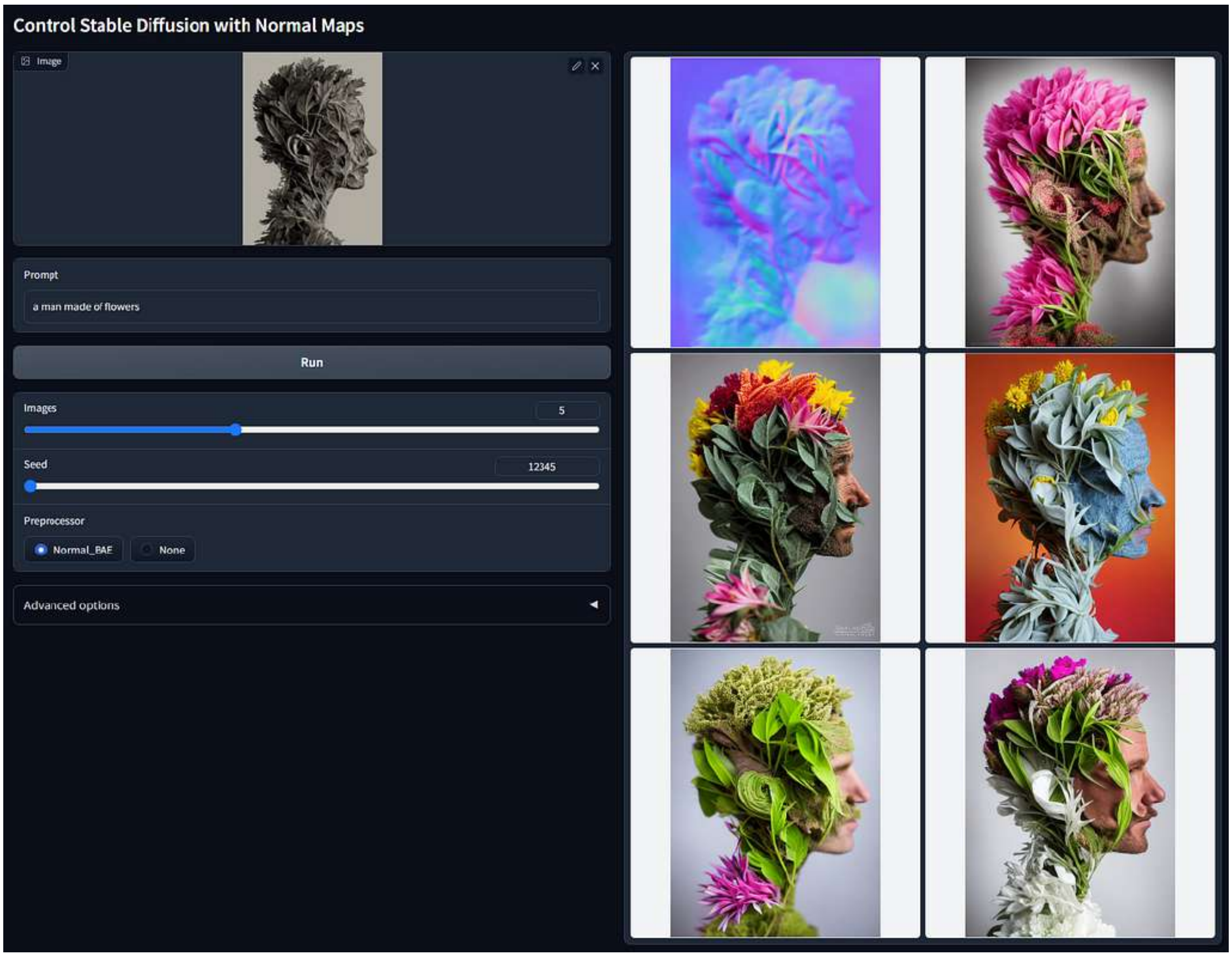
On the left the original, on the right the final

Not bad right? And that’s all for today, feel free to experiment with the settings and the different approaches.

Stable Diffusion Ultimate Guide pt. 5: ControlNet

In the fifth chapter of our Stable Diffusion guide, we are going to take a look at an incredibly powerful tool for image generation: ControlNet.

ControlNet enables us to guide the generation of our pictures in a non-destructive way. For example, take a look at the following example:



Prompt: a man made of flowers
Images: 5
Seed: 12345
Preprocessor: [✓ Normal_BAE] None

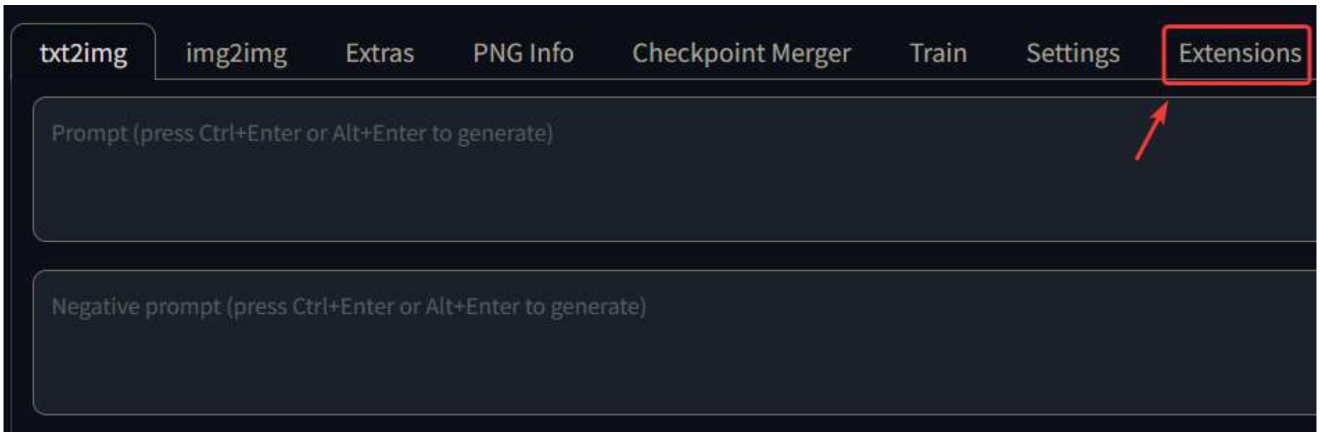
Courtesy of the ControlNet Github page

In the above example, we have an input image which gets elaborated using a Normal map (you can see the map as the first result on the right). This normal map is then used by SD to generate an output incredibly similar to the original input!

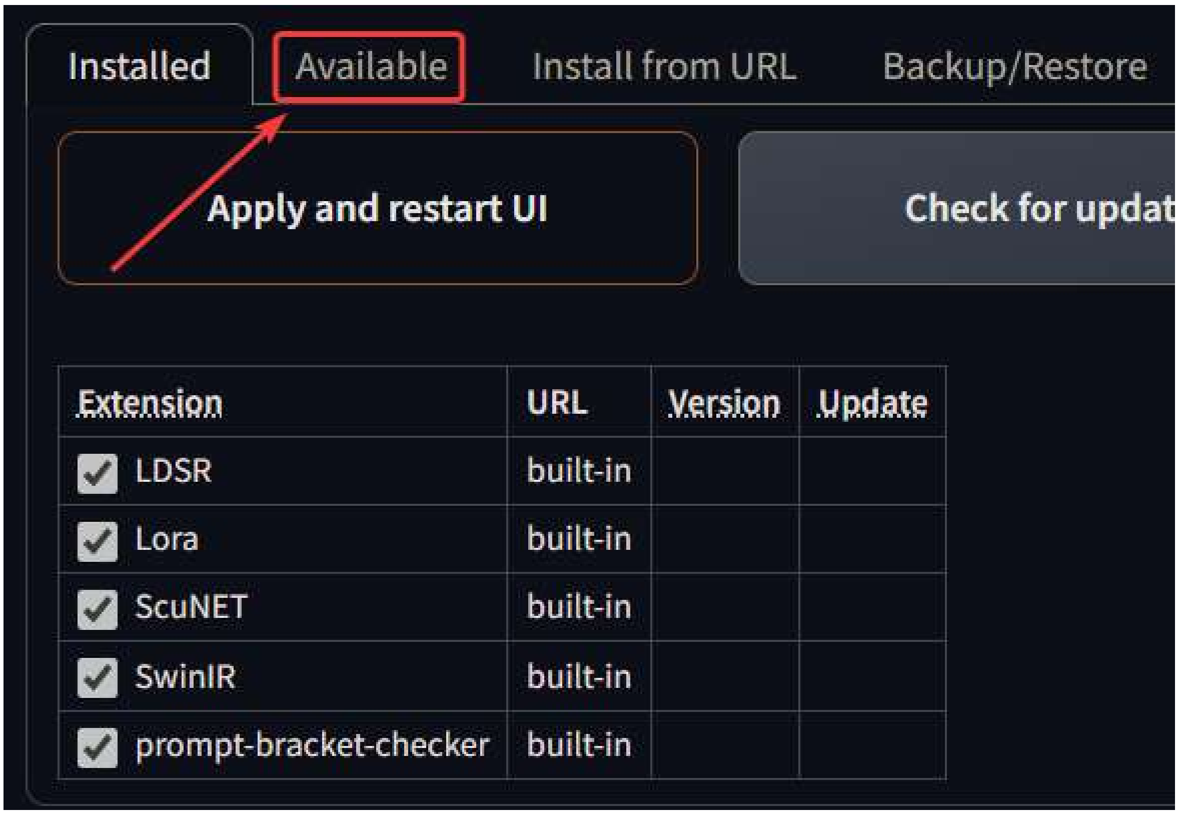
Right now, there are 14 different ControlNet models (11 production-ready and 3 experimental). In this guide, we are going to see how to install everything we need and then use the ControlNet extension to generate awesome pictures.

Setting up the extension

Let’s start by installing the extension, click on the “extension” tab on the top-right of the page:

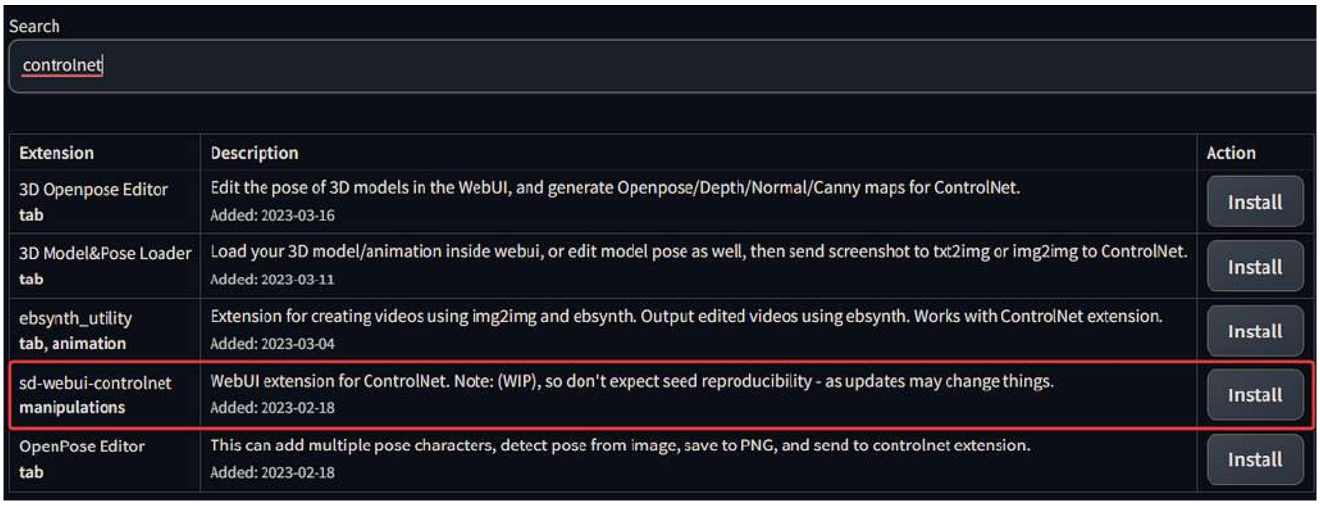

txt2img
img2img
Extras
PNG Info
Checkpoint Merger
Train
Settings
Extensions
Prompt (press Ctrl+Enter or Alt+Enter to generate)
Negative prompt (press Ctrl+Enter or Alt+Enter to generate)

Now go to the “Available” tab:



Extension	URL	Version	Update
✓ LDSR	built-in		
✓ Lora	built-in		
✓ ScuNET	built-in		
✓ SwinIR	built-in		
✓ prompt-bracket-checker	built-in		

Click on “Load from” and wait for the UI to load the list of extensions. Once you see the list, search for controlnet. In the list you should find: “sd-webui-controlnet”. Click on install, wait for it to install and then close the UI (you need to close the command prompt, not the page).



Extension	Description	Action
3D Openpose Editor tab	Edit the pose of 3D models in the WebUI, and generate Openpose/Depth/Normal/Canny maps for ControlNet. Added: 2023-03-16	Install
3D Model&Pose Loader tab	Load your 3D model/animation inside webui, or edit model pose as well, then send screenshot to txt2img or img2img to ControlNet. Added: 2023-03-11	Install
ebsynth_utility tab, animation	Extension for creating videos using img2img and ebsynth. Output edited videos using ebsynth. Works with ControlNet extension. Added: 2023-03-04	Install
sd-webui-controlnet manipulations	WebUI extension for ControlNet. Note: (WIP), so don't expect seed reproducibility - as updates may change things. Added: 2023-02-18	Install
OpenPose Editor tab	This can add multiple pose characters, detect pose from image, save to PNG, and send to controlnet extension. Added: 2023-02-18	Install

Before starting the UI with the new extension we need to download the Controlnets models. Go to the following

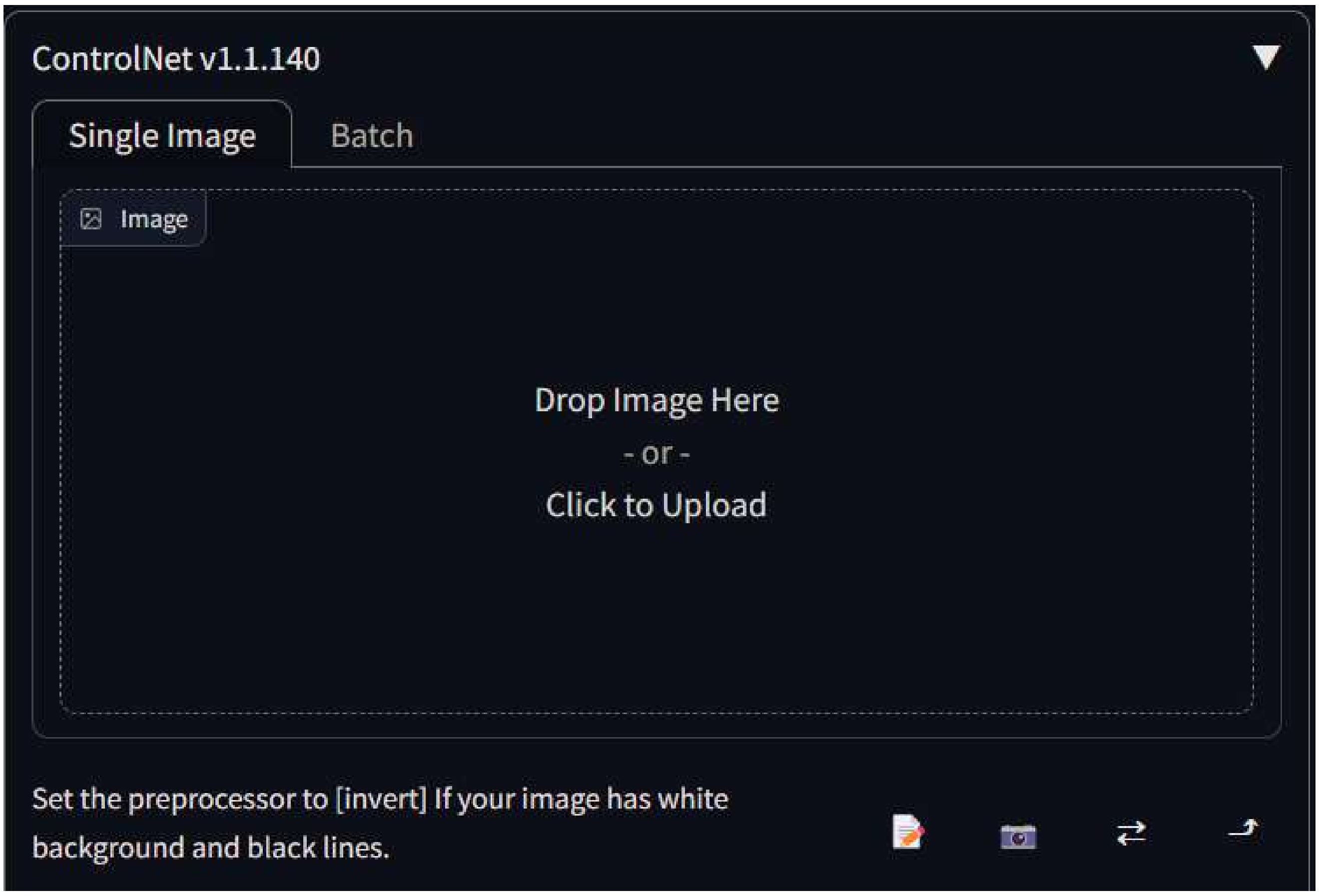
ControlNet 1.1 Models - Canny | Stable Diffusion Controlnet | Civitai

Here you'll find all the 14 different models. For this example, I'm going to use Canny and Openpose, but feel free to download the ones you like.

Once you've finished downloading the models you need, just put them inside this folder:

```
stable-diffusion-webui\extensions\sd-webui-controlnet\models
```

Now we can run the UI again, you should see the prompt installing the requirements. When everything is ready, open the UI page in the browser and you should now see a new tab in the txt2img and img2img page:



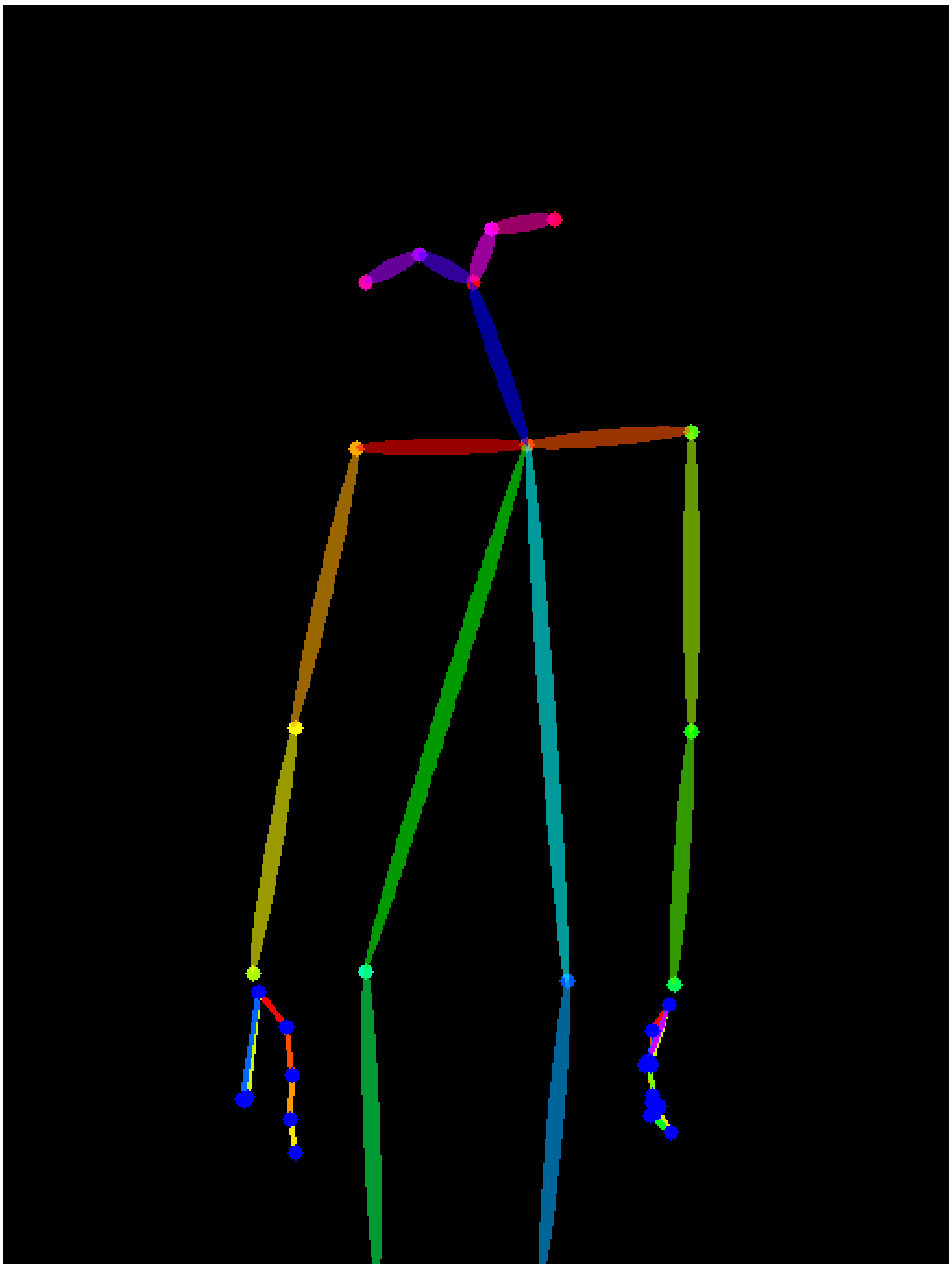
Set the preprocessor to [invert] If your image has white background and black lines.

Using ControlNet

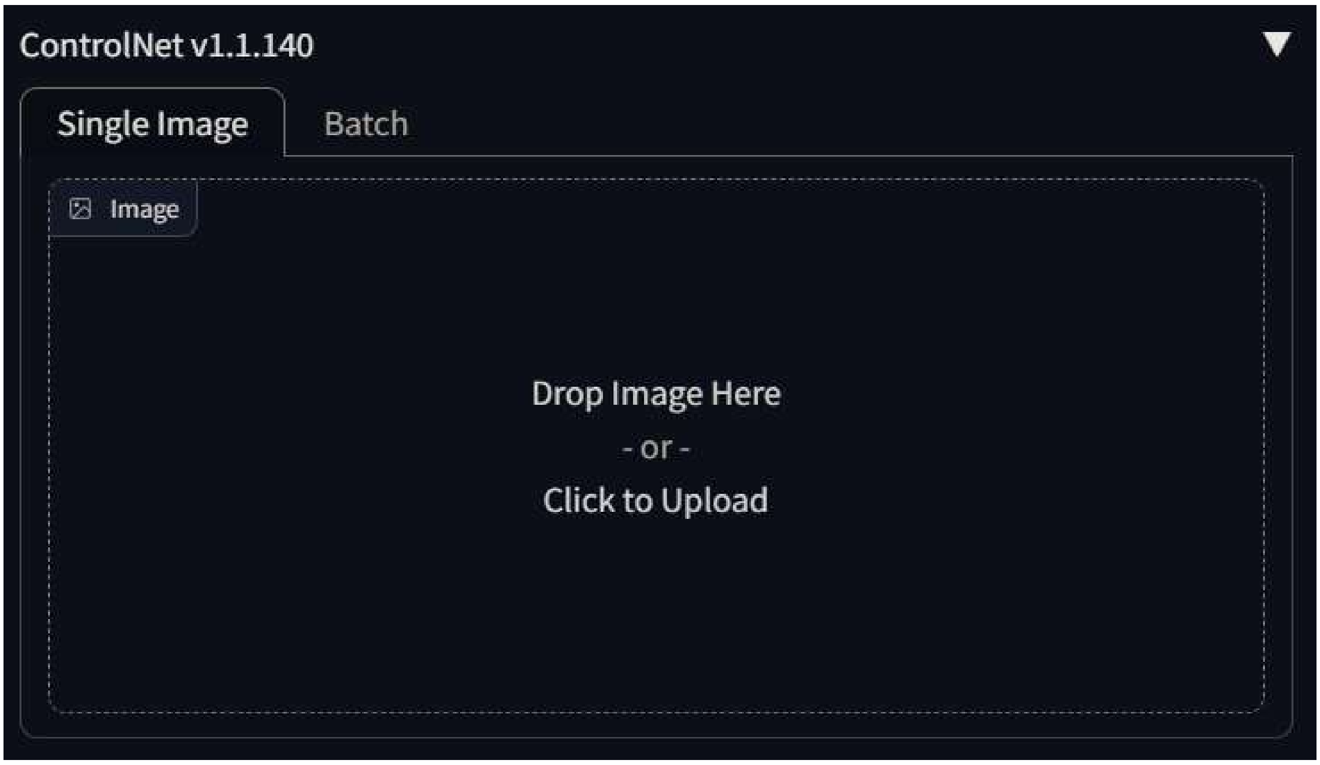
Let's start from the Openpose ControlNet. For this tutorial I'm going to use my custom model, but you can use any model you want:

Mistoon_Anime - v1.0 | Stable Diffusion Checkpoint | Civitai

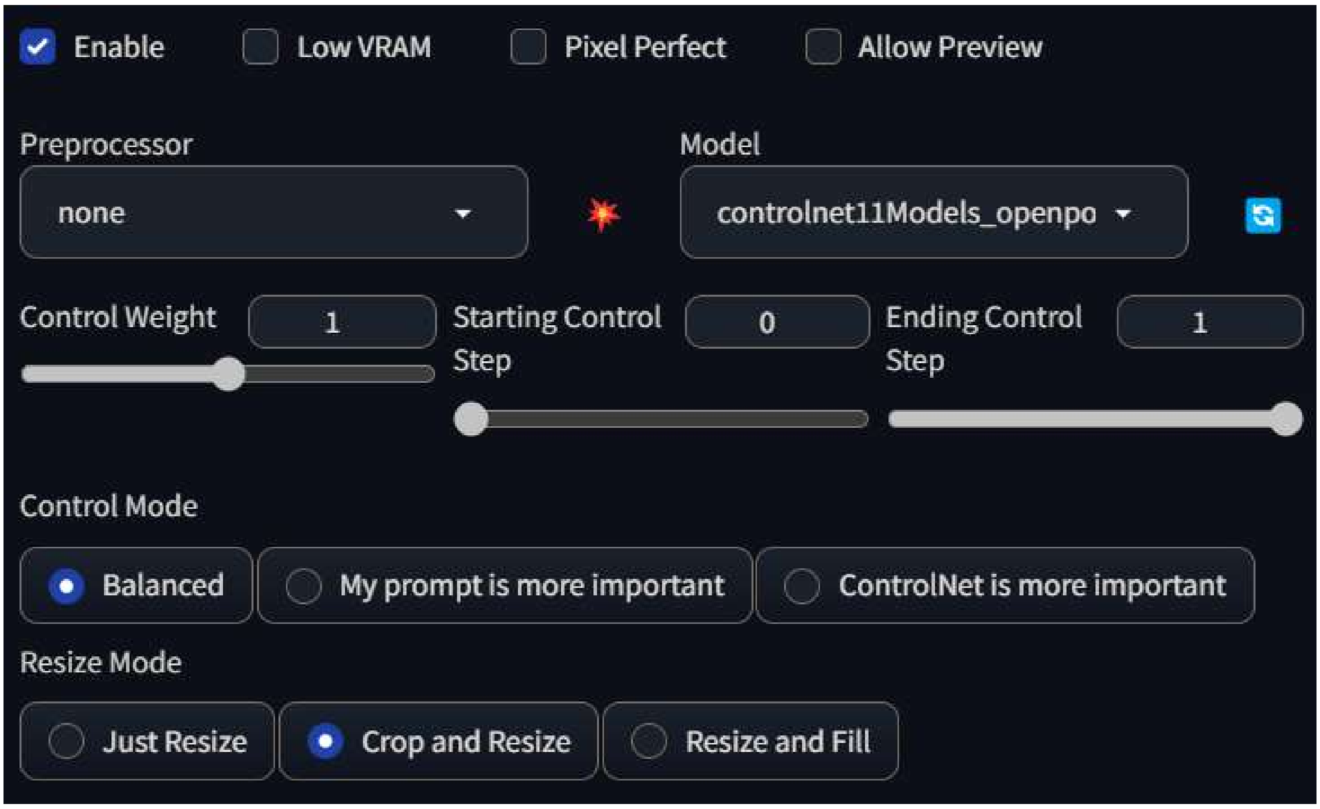
For this example we are going to use the following pose:



You can download the picture from above or search for different poses on CivitAI (there are hundreds). Now go to the Controlnet tab and drag and drop (or click on the box and select) the image you’ve just downloaded:



Now let’s have a look at the options available:


Enable [✓] Low VRAM [ ] Pixel Perfect [ ] Allow Preview [ ]

Preprocessor: none

Model: controlnet11Models_openpo

Control Weight: 1

Starting Control Step: 0

Ending Control Step: 1

Control Mode: [✓ Balanced] My prompt is more important ControlNet is more important

Resize Mode: Just Resize [✓ Crop and Resize] Resize and Fill

The first thing we need to do is to click on the “Enable” checkbox, otherwise the ControlNet won’t run. Once we’ve enabled it, we need to choose a preprocessor and a model. In this case, we won’t need a preprocessor because the image we are using is already processed (we are going to use it for the second example). In the model section, we need to choose the ControlNet openpose model.

We can also change the weight and the starting and ending control steps for the model. The weight will change how much the pose picture will influence the final picture. Lowering the weight will make the output pose different from the one passed to the ControlNet, while a higher weight will increase the similarity. The other two options will change the number of steps on which the ControlNet will be applied. For example, if you have 20 steps, and you change the “Starting Control Step” to 0.5, this will mean that the first 10 steps will be generated without the ControlNet, while the second half will be generated with the network on.

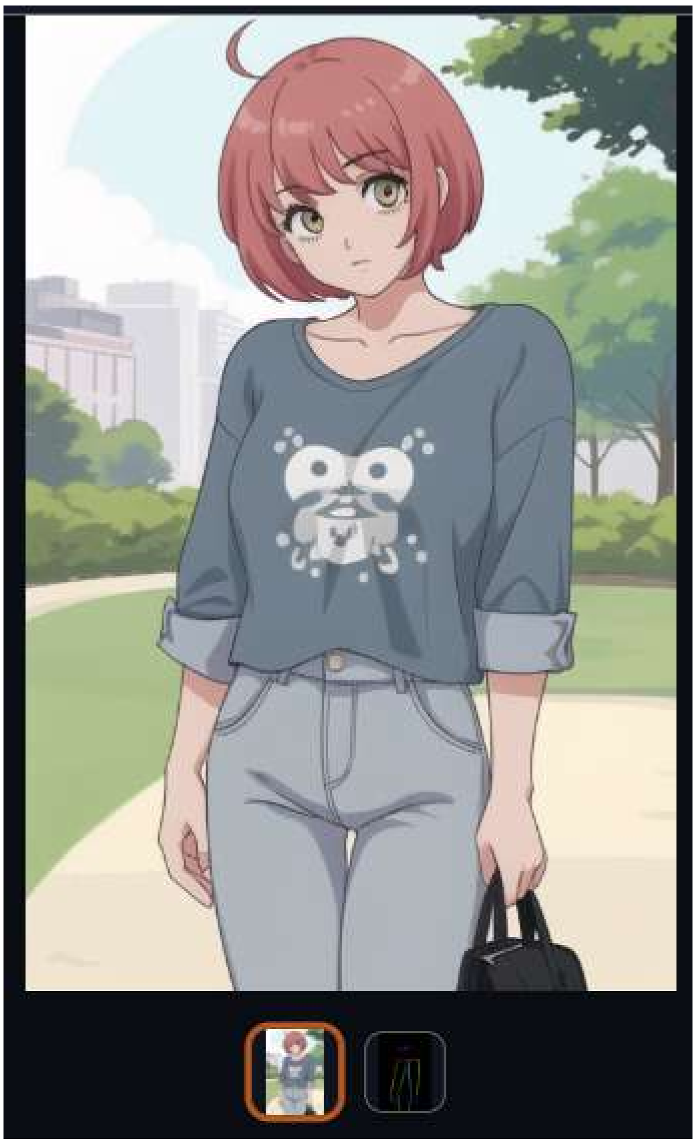
The other options are:

- Control Mode: this changes the importance of the ControlNet over your prompt. Change this if you see that the ControlNet is too strong or weak over the prompt.
- Resize Mode: this changes how the ControlNet input picture is resized to match your output settings.

Now we can generate our picture. Let’s use the following prompt and run the generation:

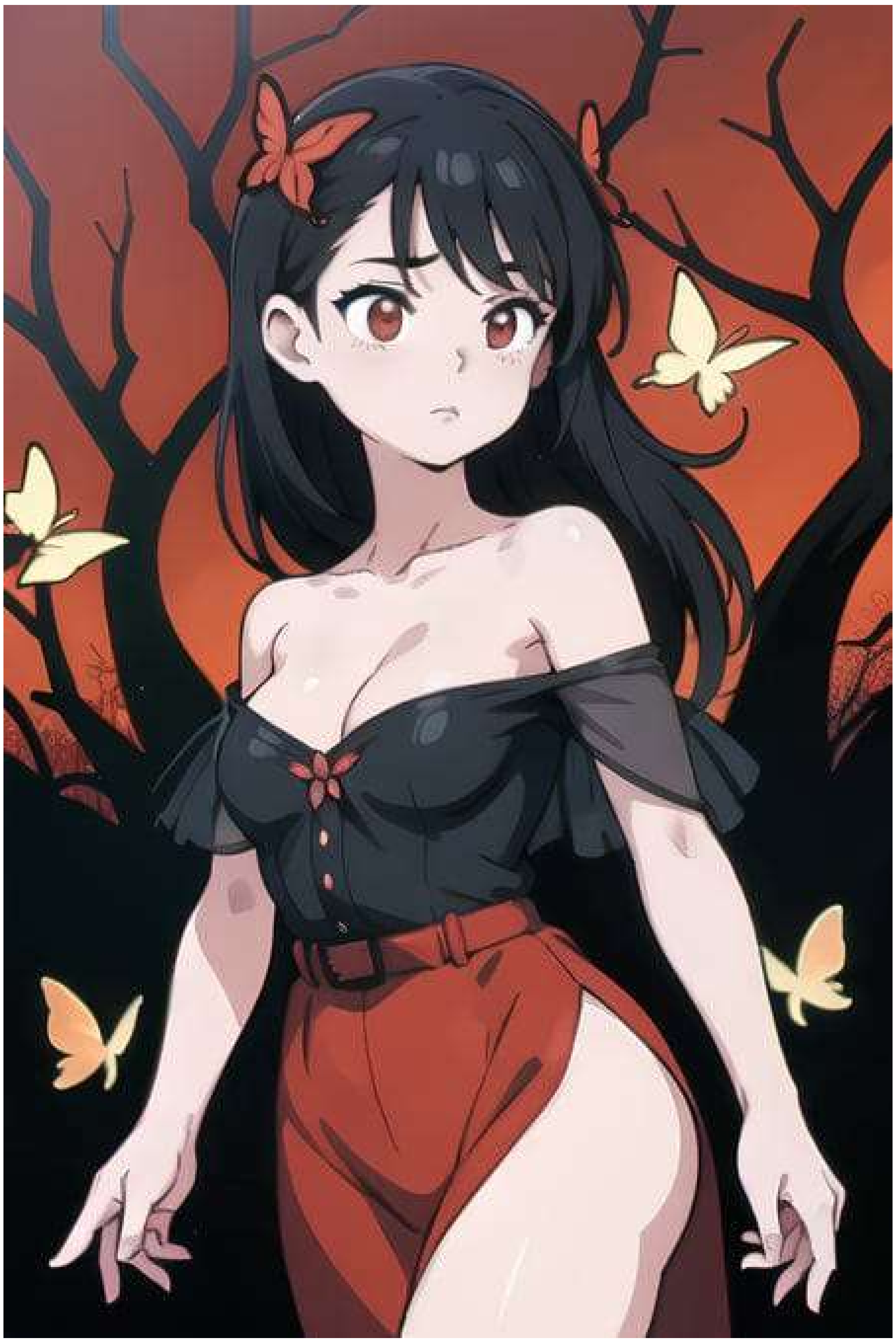
```
girl, casual clothes, summer, outdoors, park, short_hair
```

If everything worked correctly, you should see two different outputs:

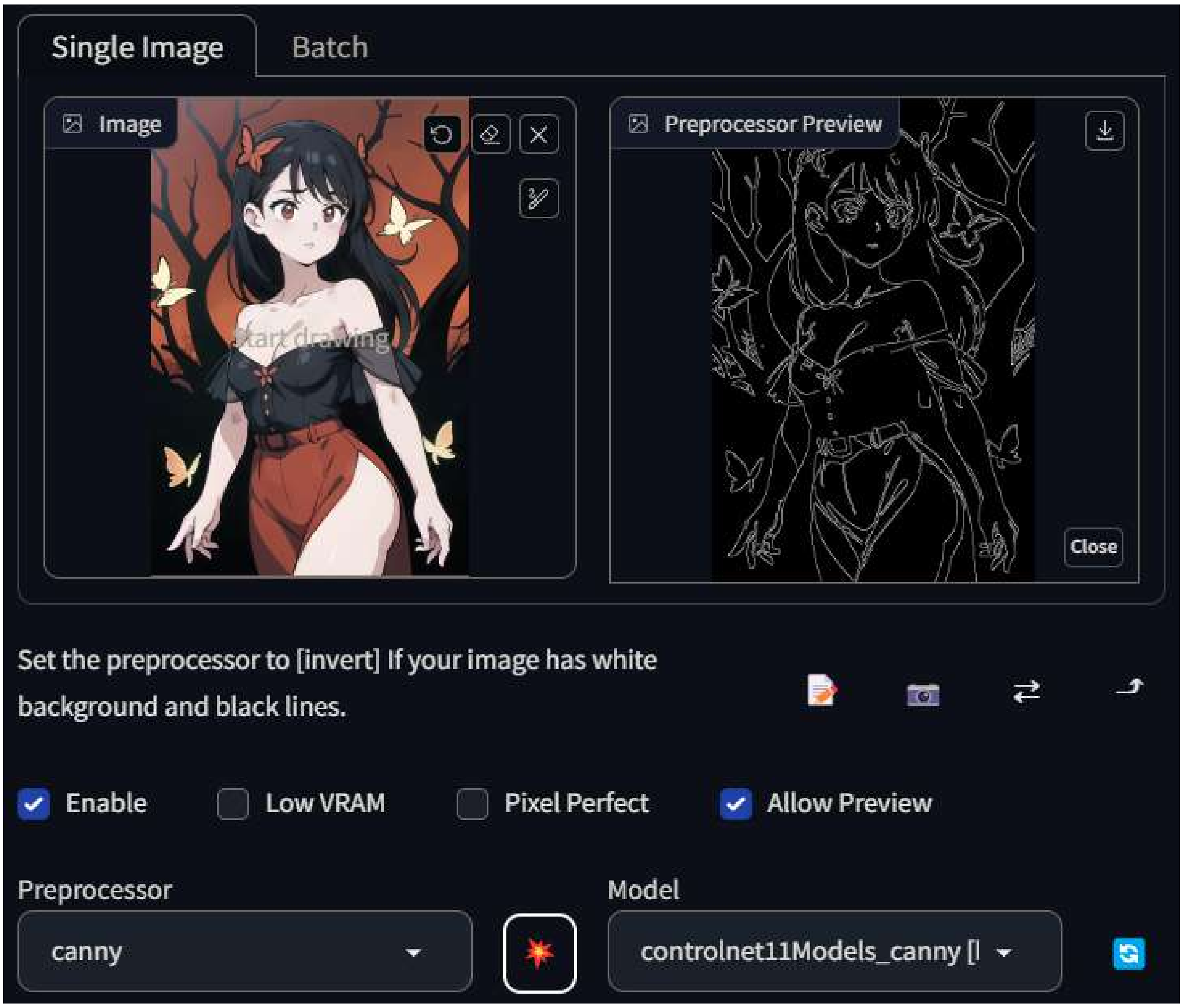


As you can see from the above picture, the pose is correctly applied.

Let’s try using the Canny model. For this example we are going to use the following image:



As we’ve done before, just download the picture and pass it to the ControlNet extension. Once you’ve done this, enable the “Allow Preview” checkbox, choose “Canny” as preprocessor and click on the “explosion” icon. You should now see the preview of the preprocessor:



Set the preprocessor to [invert] If your image has white background and black lines.

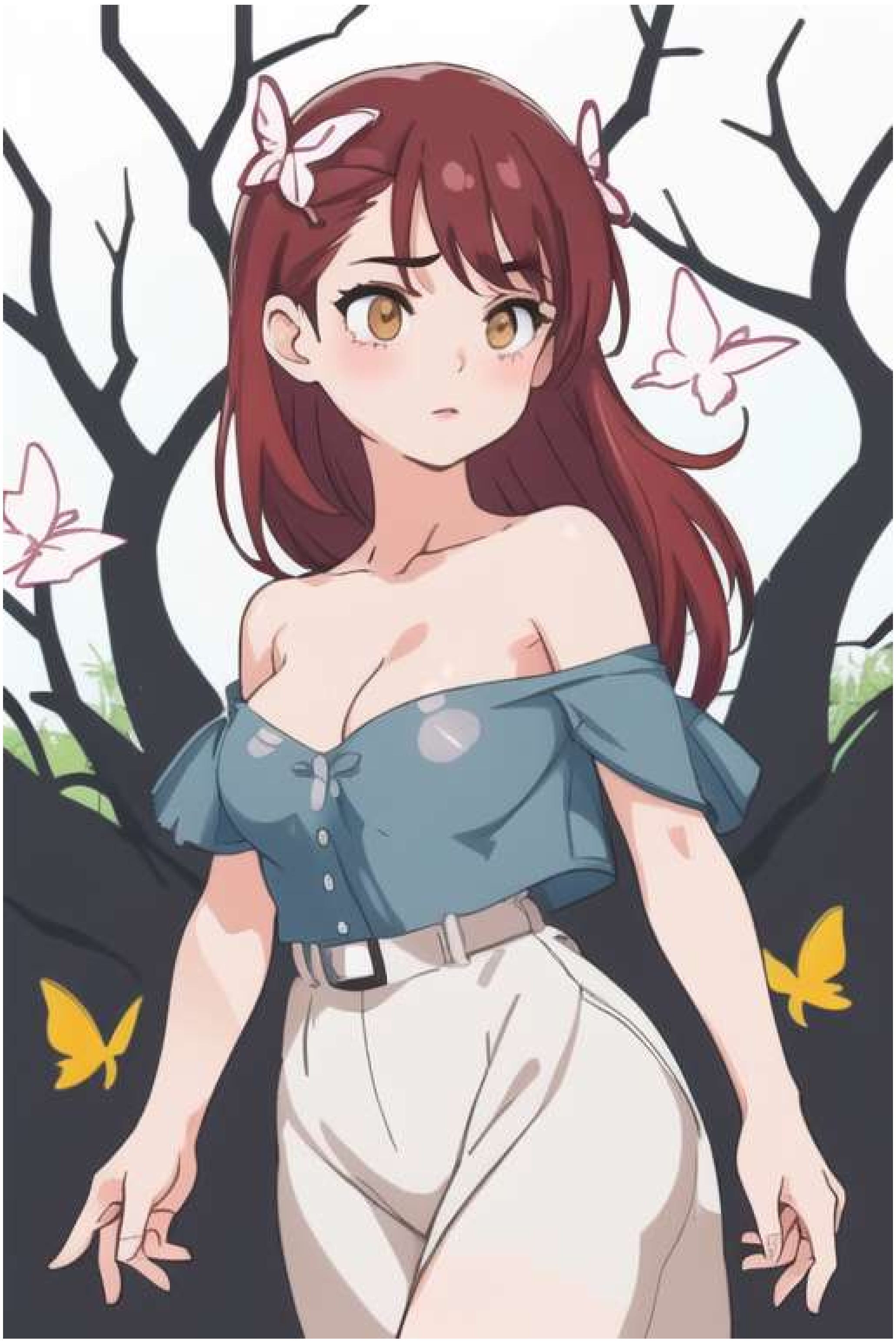
[✓ Enable] Low VRAM Pixel Perfect [✓ Allow Preview]

Preprocessor: canny

Model: controlnet11Models_canny [I

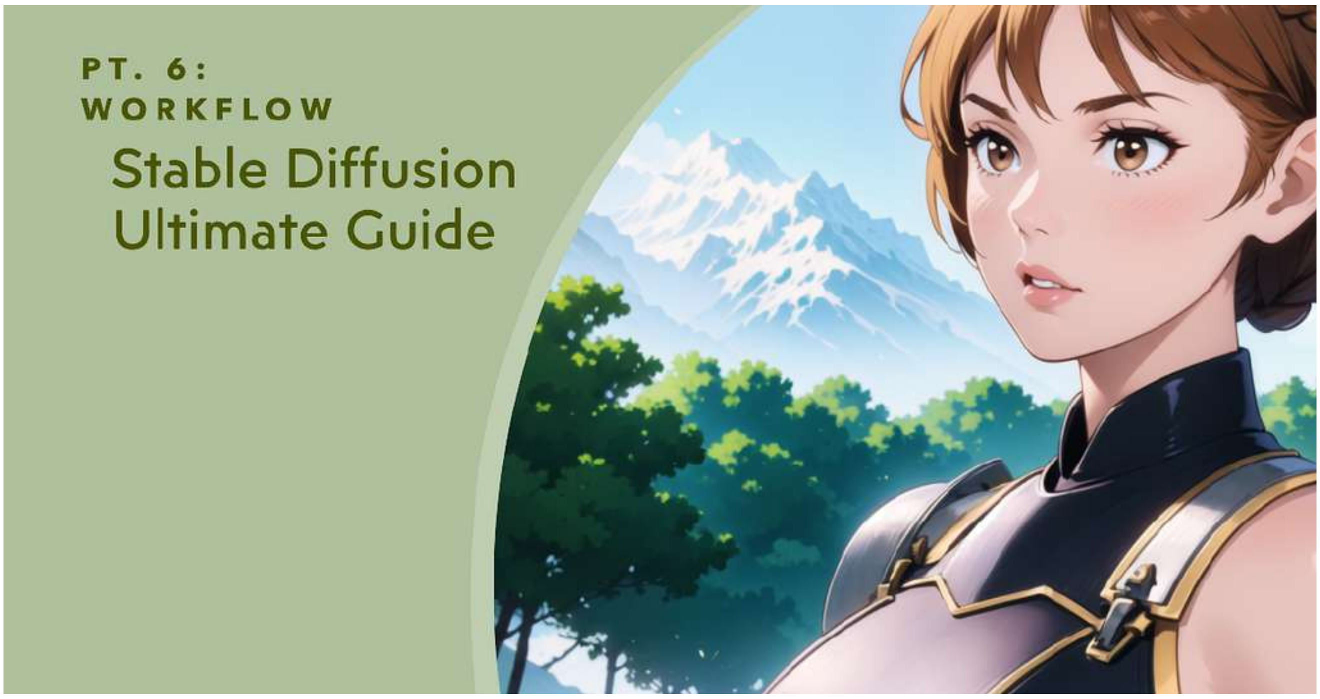
Now that we have seen that the preview looks good, we can now change the model to the controlnet_canny one and run the generation again.

You should now see a picture that looks similar to this one:



As you can see the image is incredibly similar to the original one, but with different colors. If we want to maintain the general composition, but still change the image, we can lower the weight to 0.35 and run this again:

Stable Diffusion Ultimate Guide pt. 6: Workflow



In this sixth chapter of the Stable Diffusion guide, we are going to take a look at how I usually create my pictures.

As usual, I'll be using my custom model:
[Mistoon_Anime - v1.0 | Stable Diffusion Checkpoint | Civitai]

Requirements

To follow this example, you'll need:

- The Automatic1111 Stable Diffusion UI.
- A GPU that can handle images of 768x1152 size. I'm using a RTX 4080 with 16 GB of RAM. In case you have access to less VRAM, you'll probably be able to work with a lower resolution.
- An image editing program. I'm using Krita (which is an awesome free digital painting application), but you can use your preferred tool (Photoshop, Gimp, Paint.Net…).
- (Optional): a textual inversion for the negative part. In this example I'm using these: one and two.
- (Optional): the following two LoRAs: Summer and Stylish

Let's Start

Open the Stable Diffusion UI and start from a blank prompt. If you want, you can copy my prompt or use yours.

Prompt (you'll need the LoRAs mentioned above):

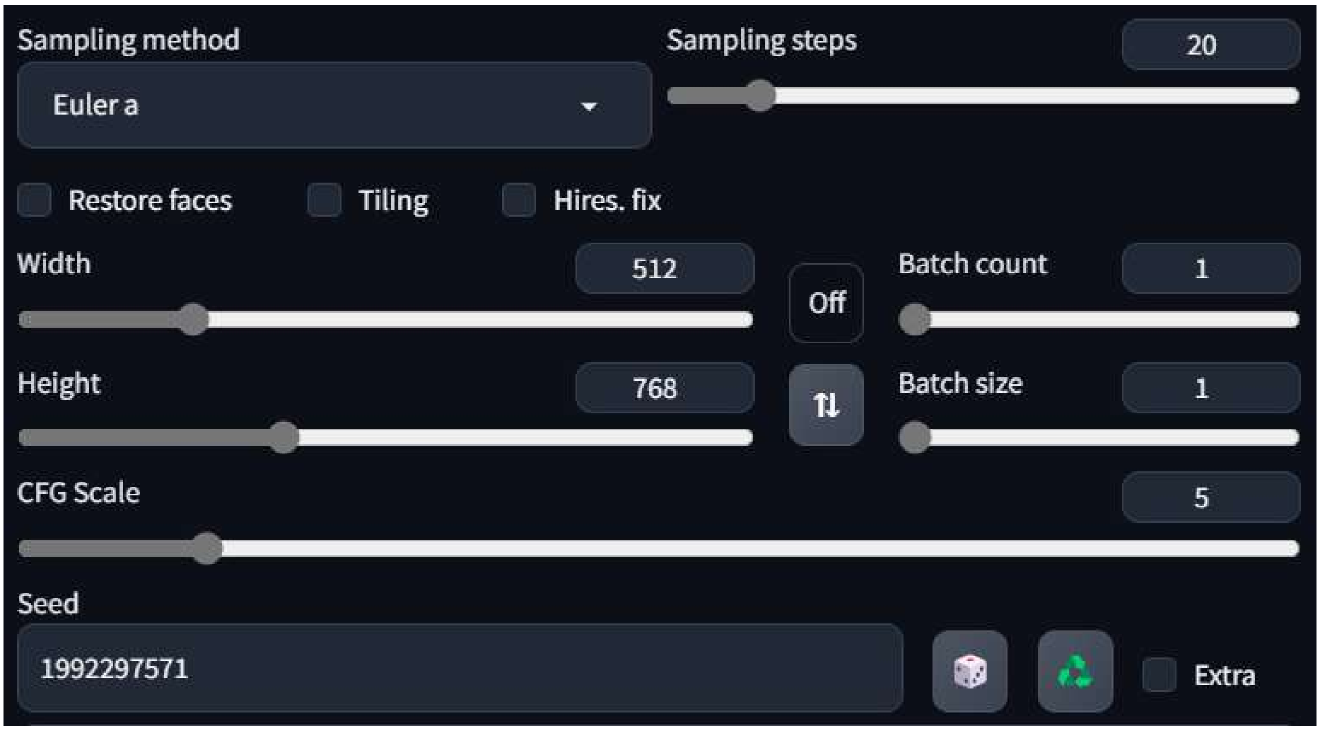
```
girl,outdoors,
knight,(armor:1.2),upper_body
```

```
<lora:summer:0.6> <lora:stylish:0.6>
```

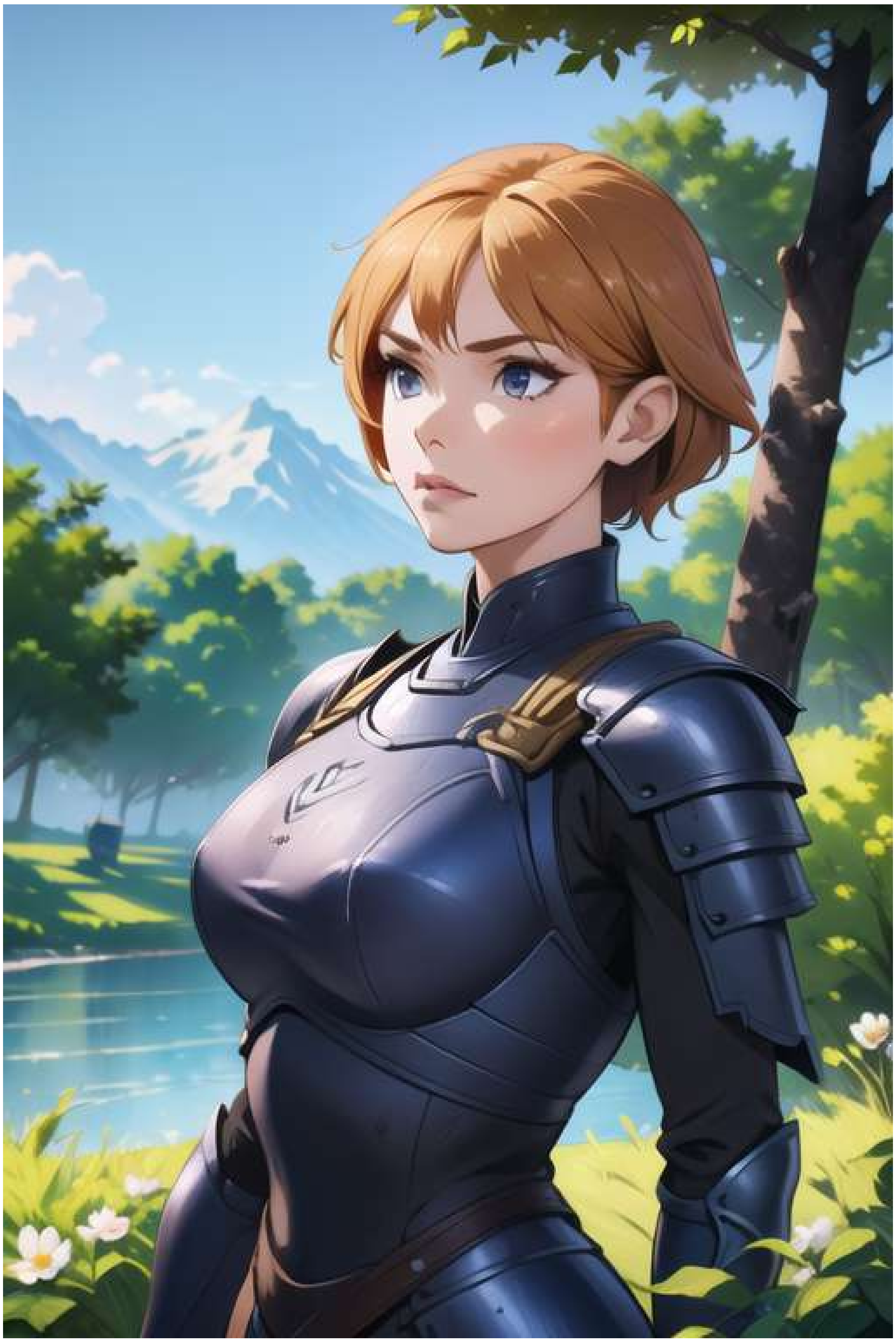
Negative Prompt (you'll need the Textual inversions mentioned above):

```
verybadimagenegative_v1.3 badhandv4
```

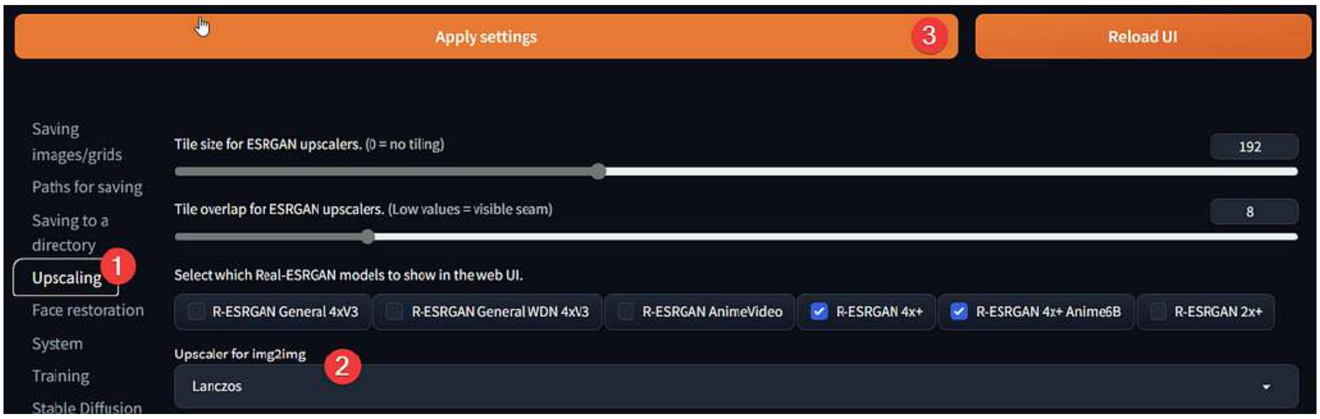
Other Properties:


Sampling method: Euler a
Sampling steps: 20
Restore faces: ☐
Tiling: ☐
Hires. fix: ☐
Width: 512
Height: 768
Batch count: 1
Batch size: 1
CFG Scale: 5
Seed: 1992297571
Extra: ☐

If you want to follow along with the same picture you can use the same seed I've used: 1992297571:



Before we can start working with this picture, you'll probably need to enable an option. Go to the settings tab and change the Upscaler for img2img settings as shown below (don't forget to click on apply settings!):



Tile size for ESRGAN upscalers. (0 = no tiling) 192

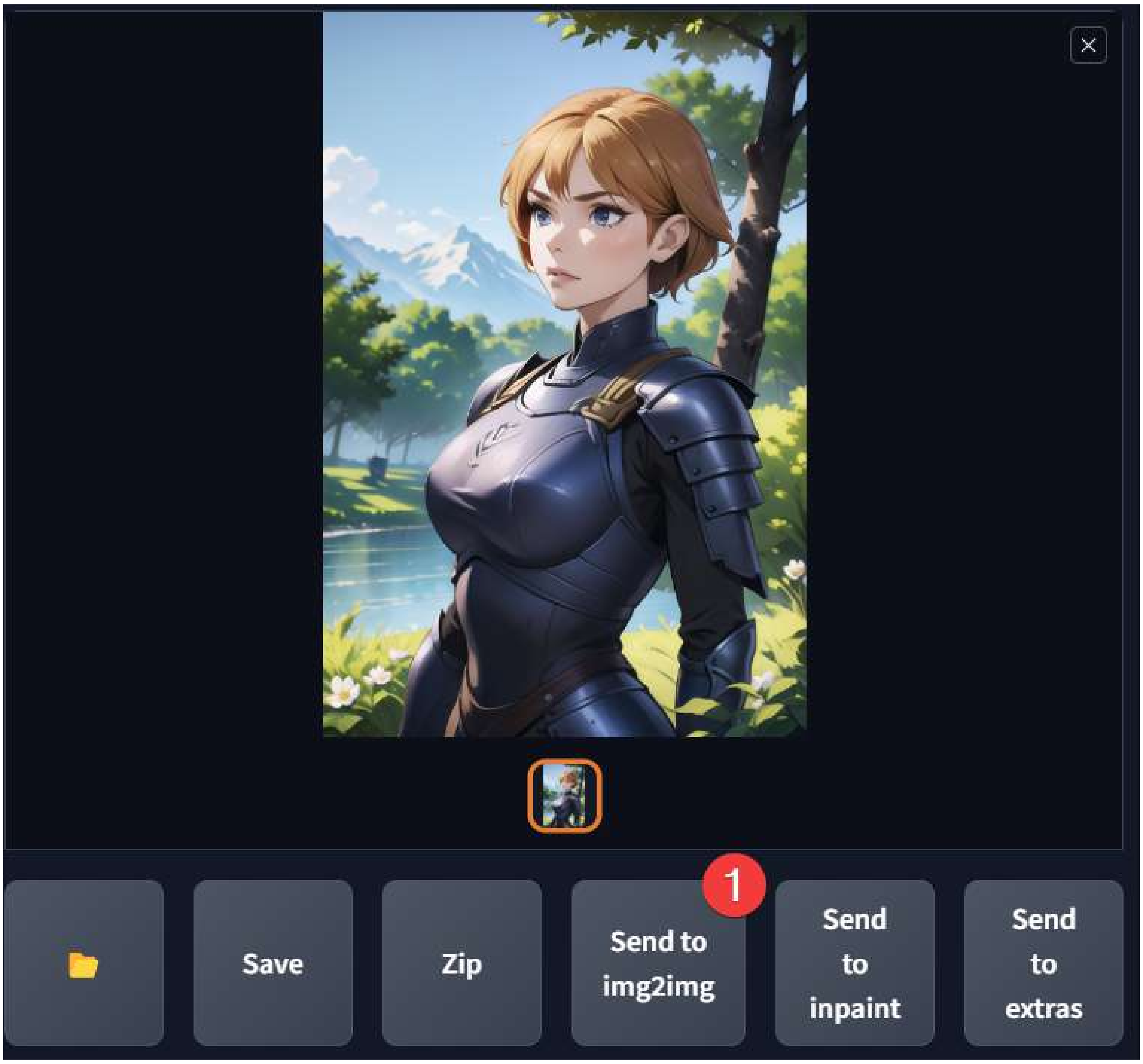
Tile overlap for ESRGAN upscalers. (Low values = visible seam) 8

Select which Real-ESRGAN models to show in the web UI.

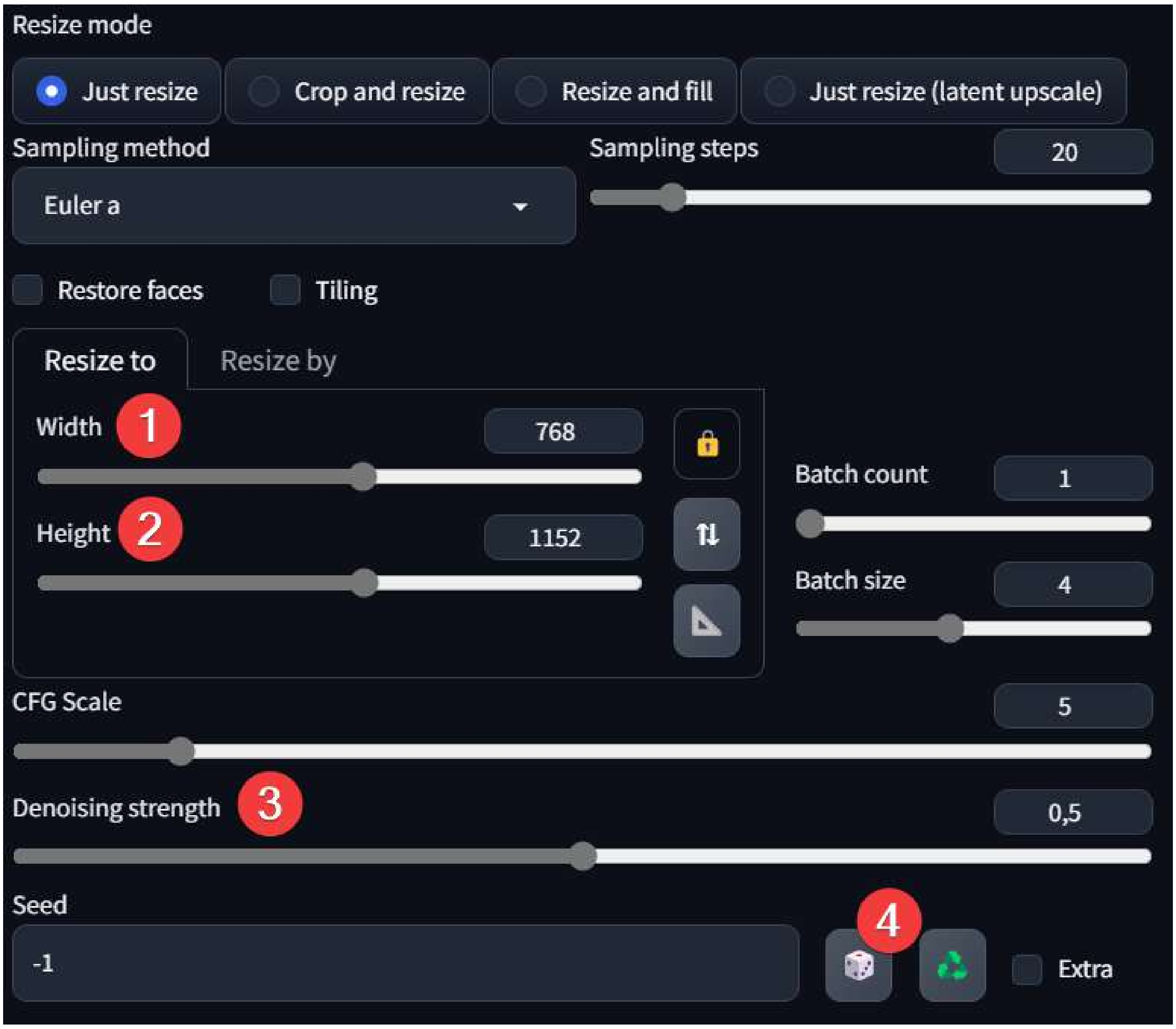
- [ ] R-ESRGAN General 4xV3
- [ ] R-ESRGAN General WDN 4xV3
- [ ] R-ESRGAN AnimeVideo
- [✓] R-ESRGAN 4x+
- [✓] R-ESRGAN 4x+ Anime6B
- [ ] R-ESRGAN 2x+

Upscaler for img2img: Lanczos

Now we can go back to the txt2img tab and click on the “Send to img2img” button:



Now the img2img tab should open automatically. The first thing I usually do is just to upscale the picture to 768x1152. To do this, we just need to change a few parameters on the page:


Resize mode

(● Just resize) ○ Crop and resize ○ Resize and fill ○ Just resize (latent upscale)

Sampling method: Euler a

Sampling steps: 20

☐ Restore faces ☐ Tiling

Resize to | Resize by

Width 1: 768

Height 2: 1152

Batch count: 1

Batch size: 4

CFG Scale: 5

Denoising strength 3: 0,5

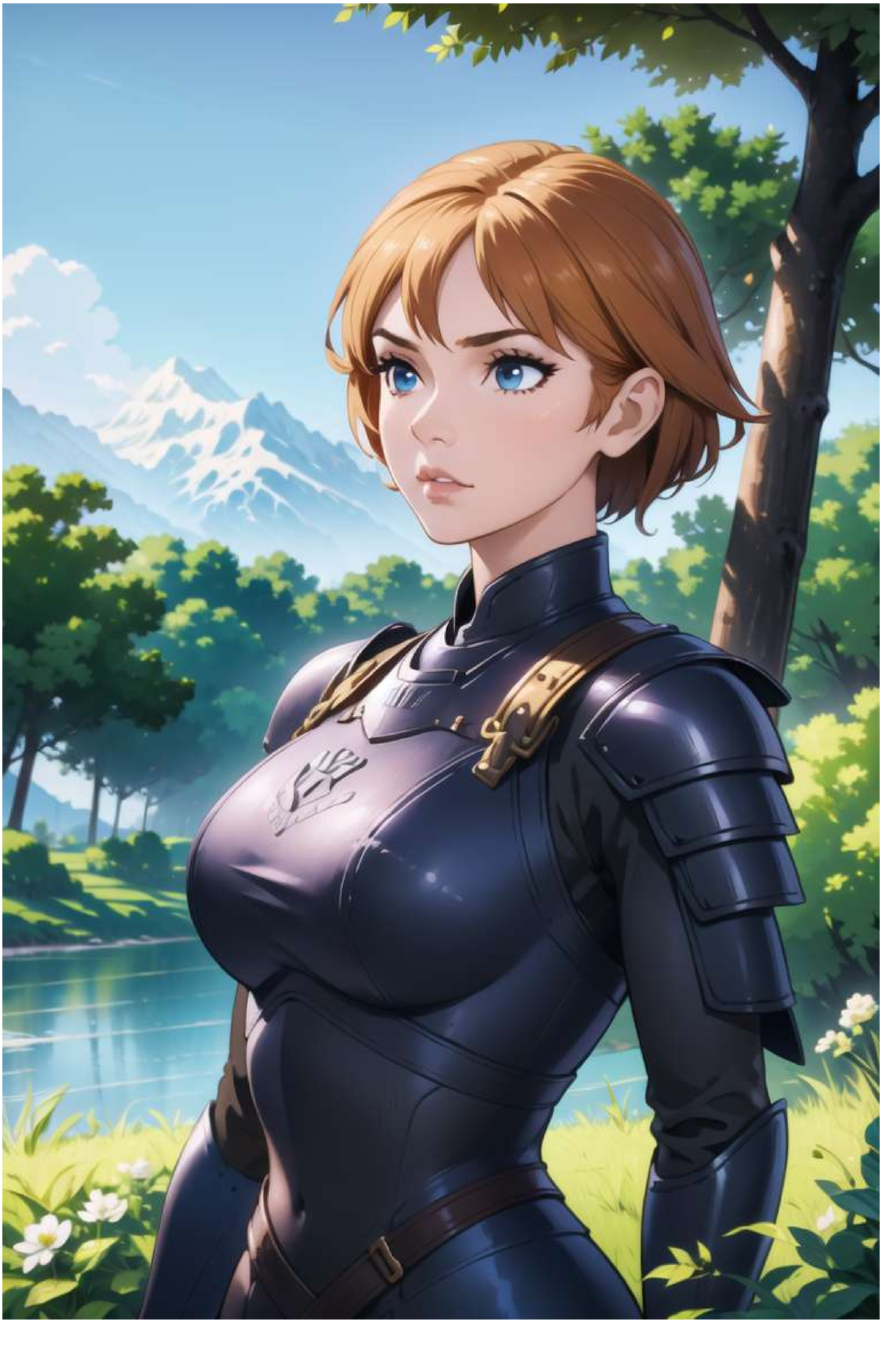
Seed: -1 4

☐ Extra

1–2: Change the resolution to 768 (Width) and, 1152 (Height)
3: Change the denoising strength to 0.5. This should add details to the picture without changing the original picture too much.
4: Set the seed to random. If you've used the seed I've mentioned above you'll probably find it on this page as well, just write -1 or click the dice button

After that, change the Batch Count and Size accordingly to what your system is capable of managing (if you don't know what you are doing, just keep them at 1) and click generate.

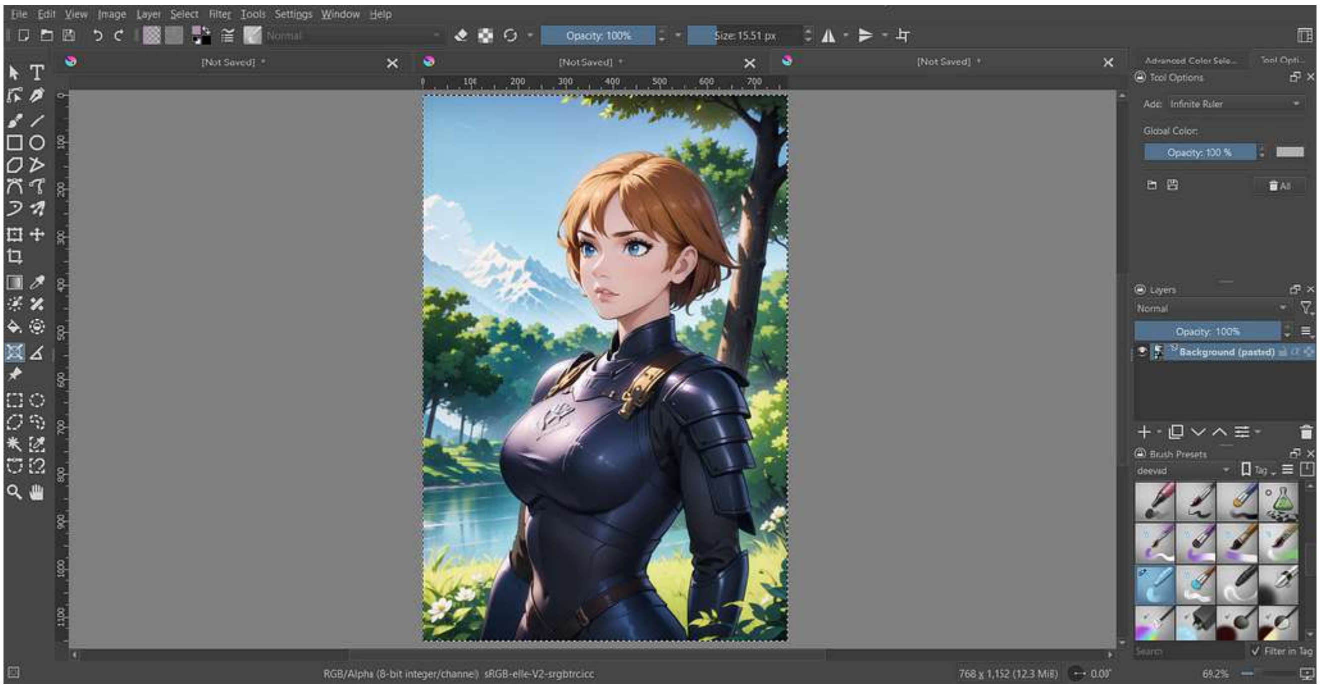
Now choose the best looking picture, or just generate another batch until you are satisfied. I'll be using this one:



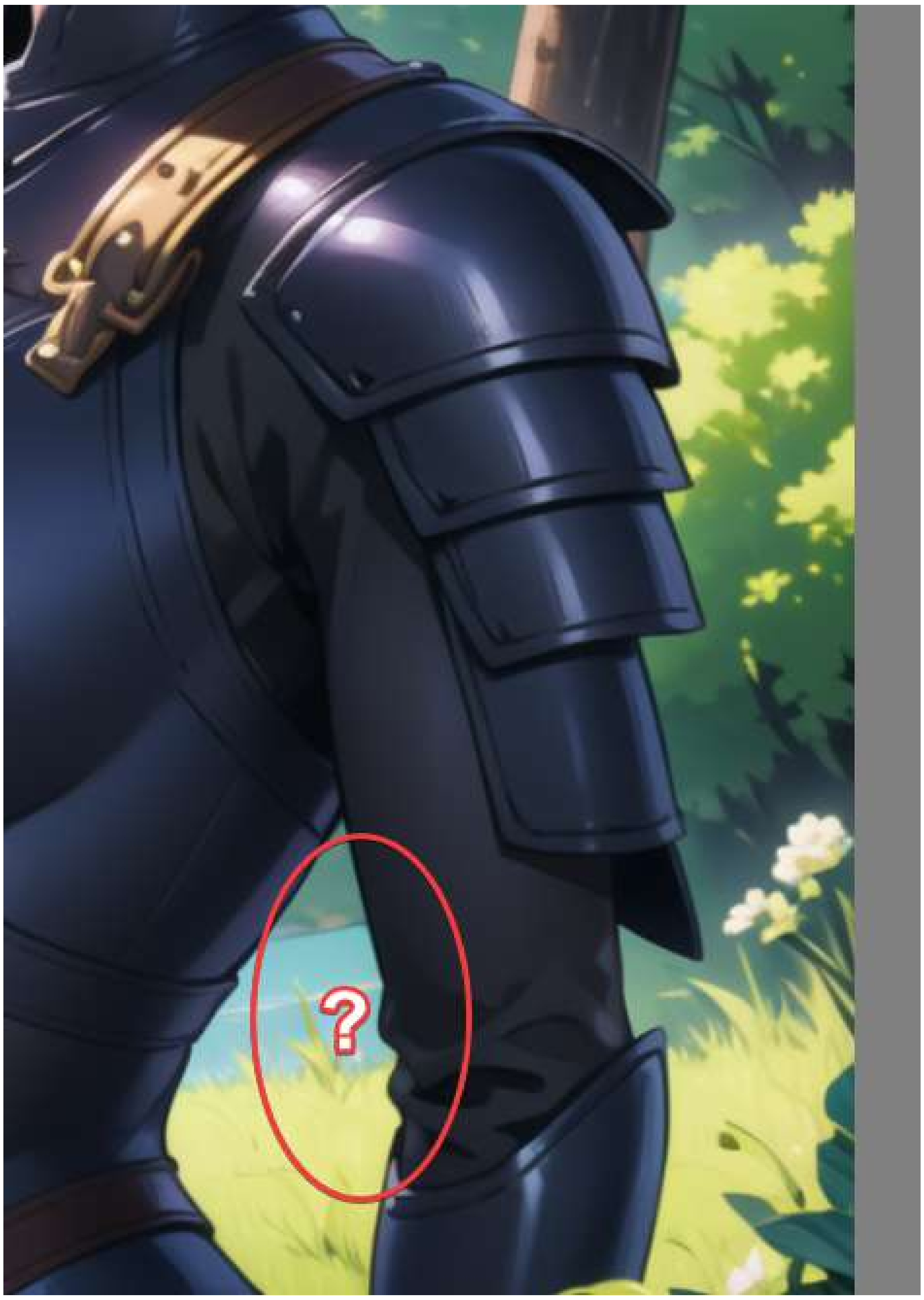
Now that we've a detailed enough picture, it's time to work on some changes. These are the changes we are going to work on:

1. Remove the armor from the left arm
2. Add gold details to the armor
3. Change the overall structure of the armor

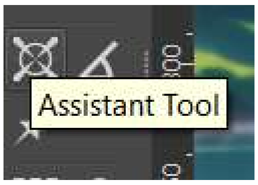
Let's start by opening Krita and pasting the image into it (just press CTRL+SHIFT+N with the picture copied from the browser):



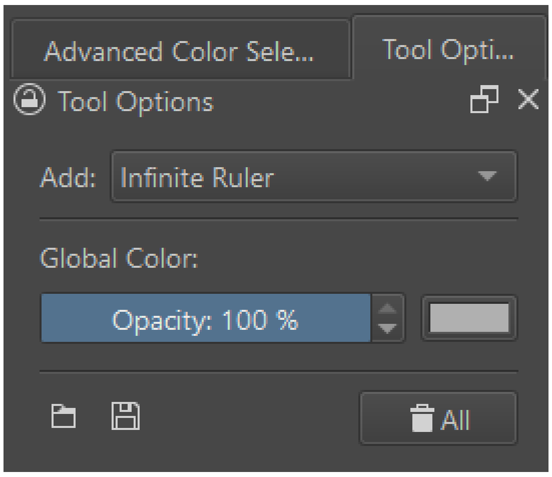
Let's start working on the arm. Before we begin there is another thing we need to fix: the tree behind the character is missing a piece:



To help us draw the missing part of the tree we can use the “infinite ruler” from the assistant tools. We just need to click on the icon:



Then you need to go to the menu on the right and select the tab “Tool Options”. In that tab you’ll find the “Infinite Ruler” setting:


Advanced Color Sele... | Tool Opti...

Tool Options

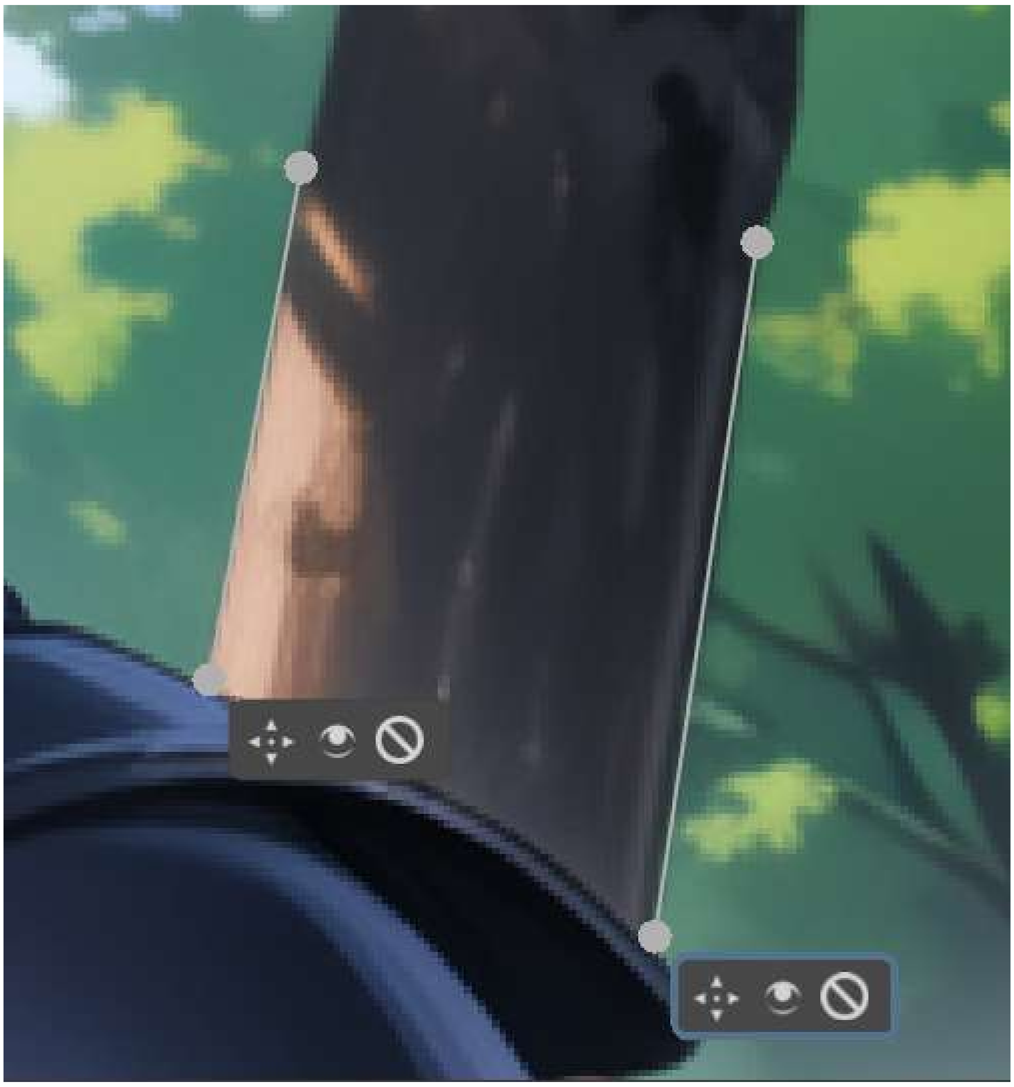
Add: Infinite Ruler

Global Color:

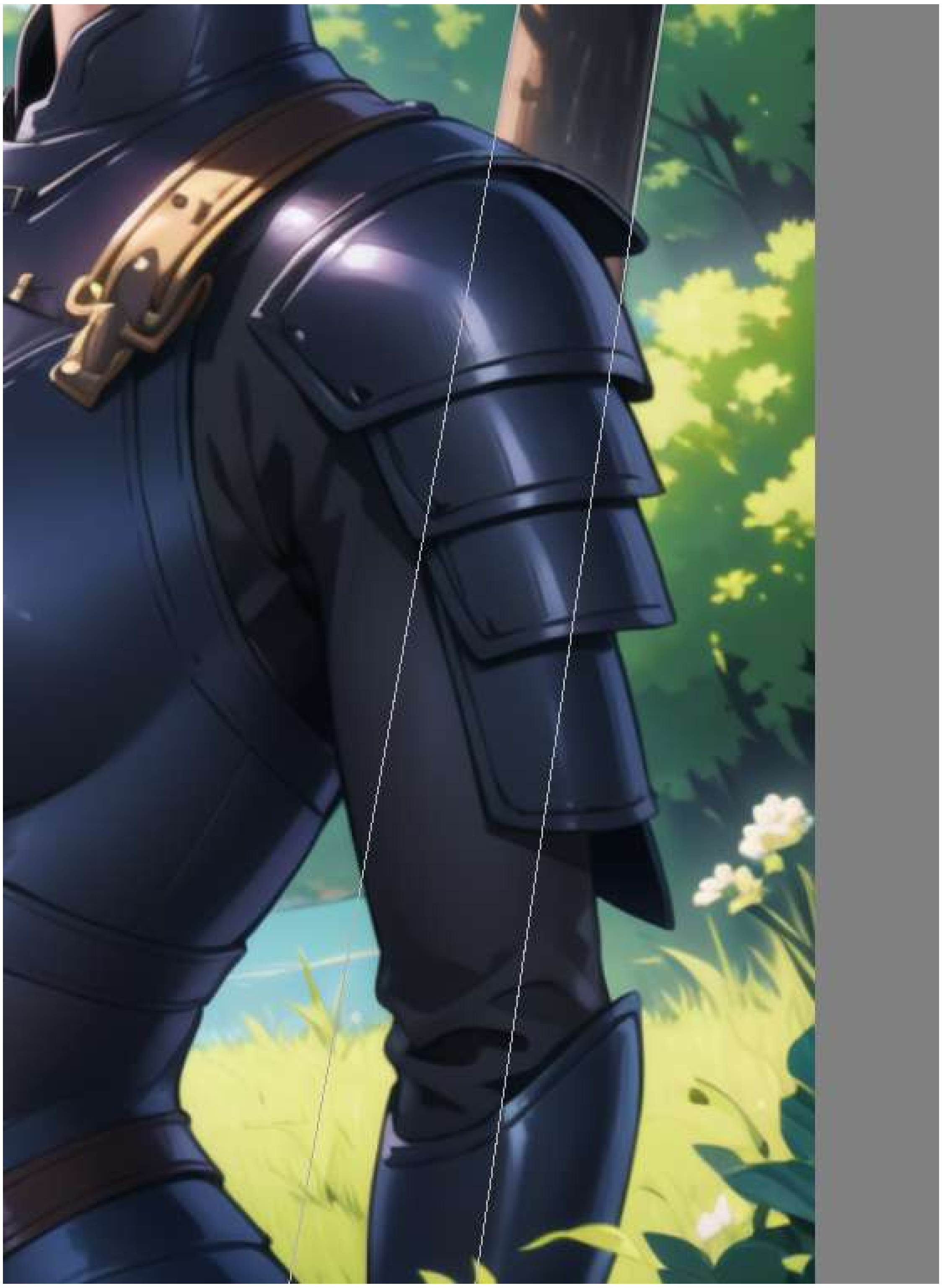
Opacity: 100 %

All

Now we can click on the picture to draw the two rulers to help us draw the tree:

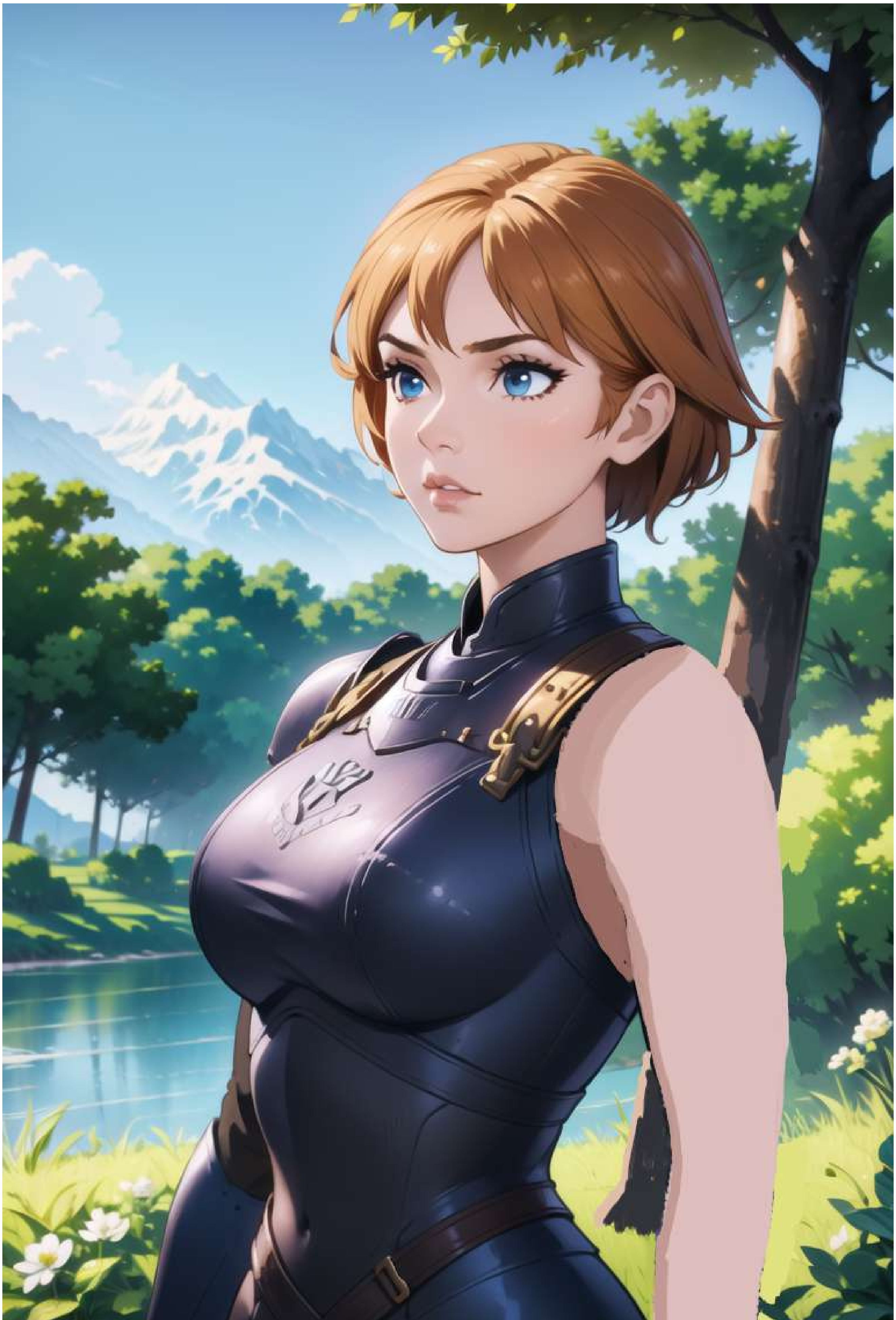


Now select the brush tool (or press B) and you should see the full rulers:



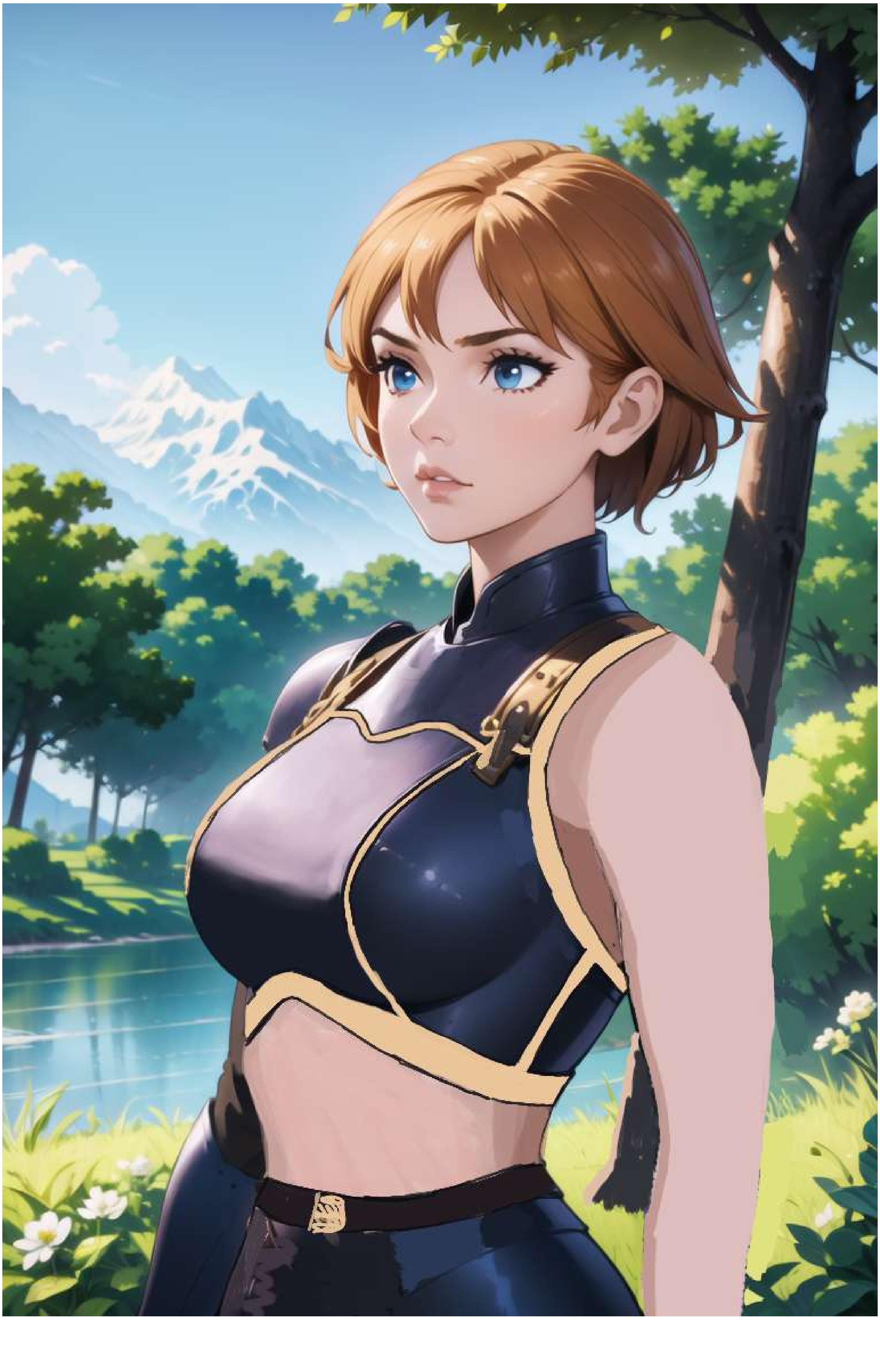
We can now start working on the arm. You don't need to be a professional artist to make these changes (I am pretty bad at drawing), you just need to draw in a way that SD will be able to understand when working on the picture.

After a few changes, this is how my picture looks like:

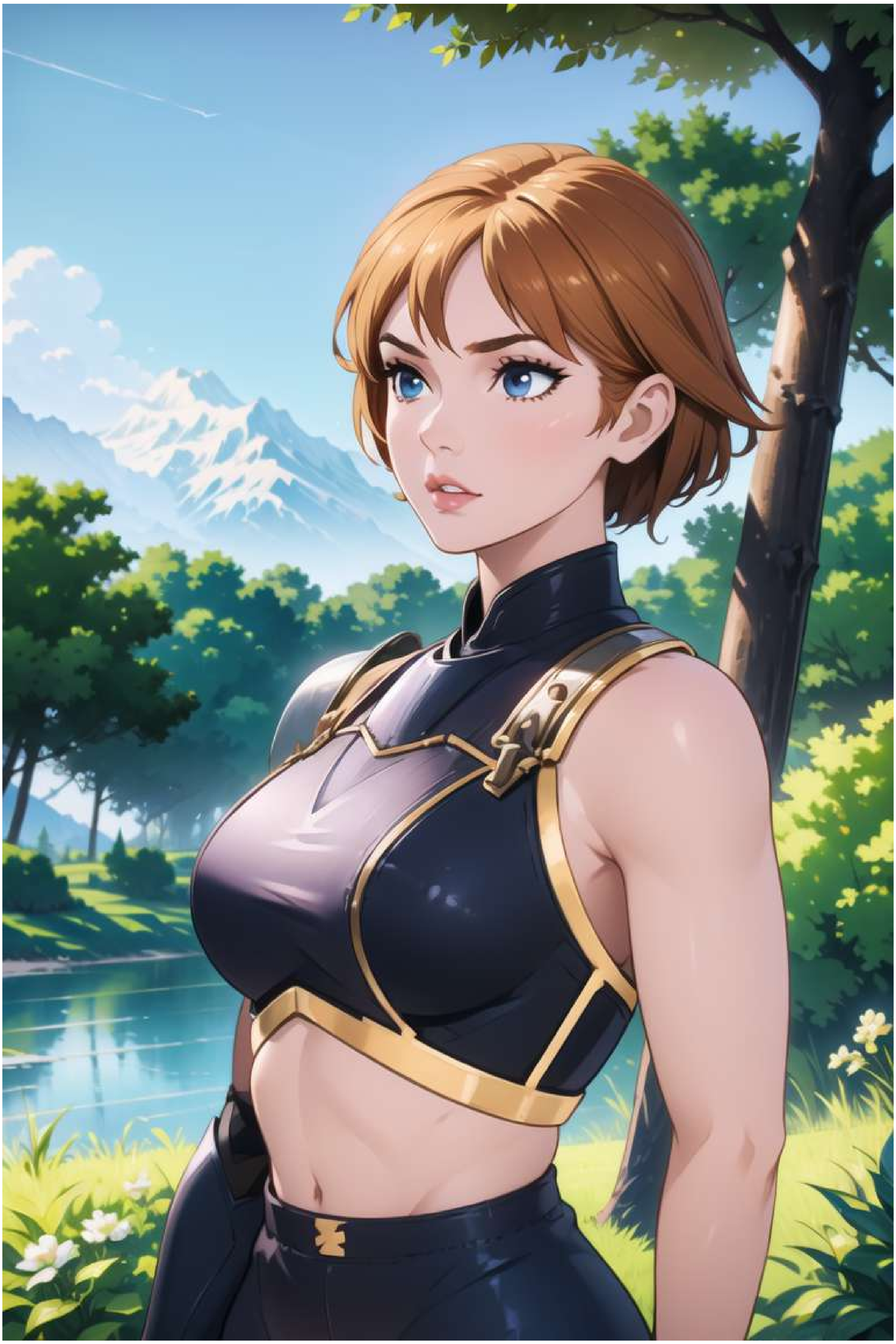


We are going to do the repeat the same process for the other steps. If you need to use a picture as a reference in Krita, you can just copy that picture and then add it as a reference using CTRL+SHIFT+R.

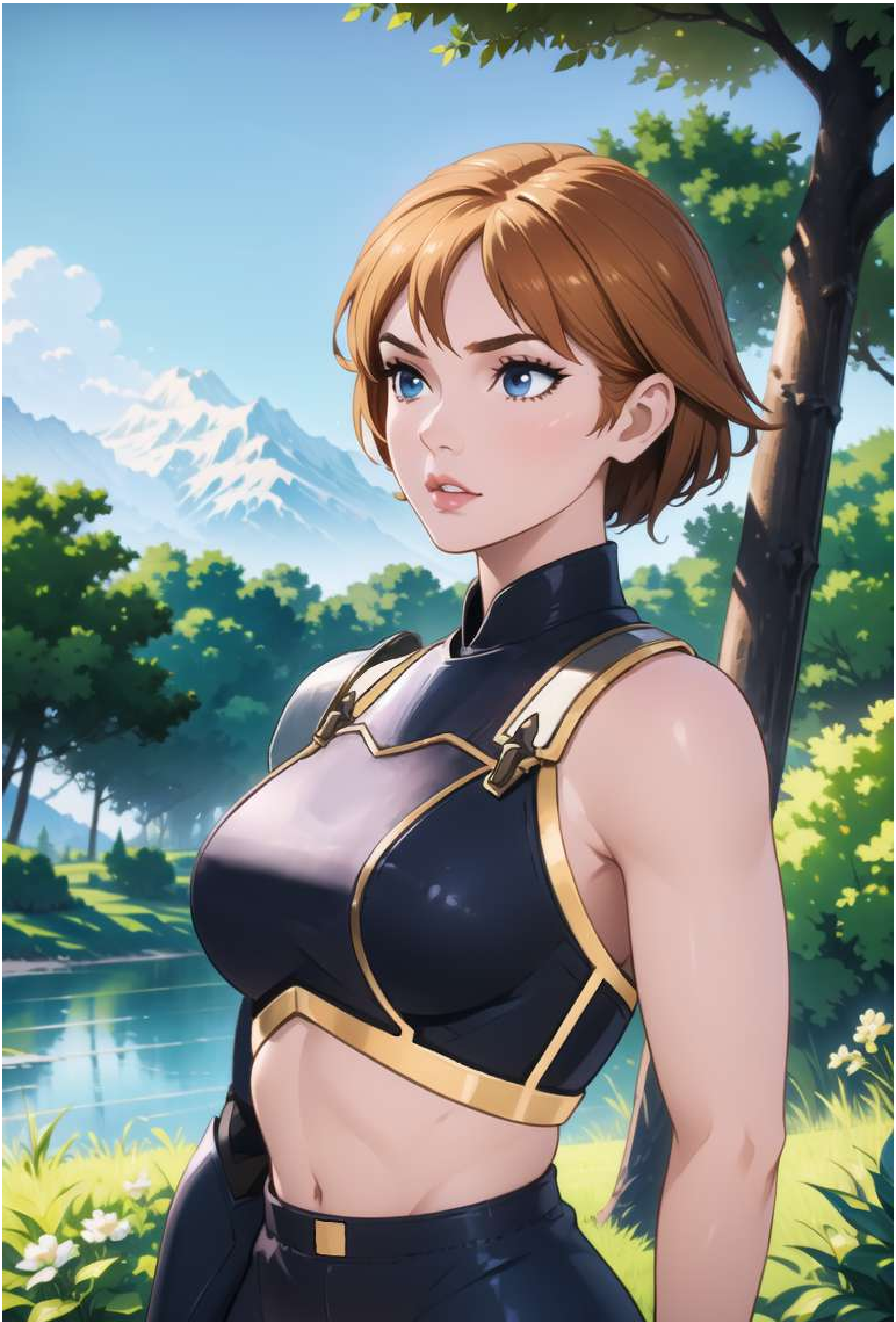
And this is the final result after the other edits:



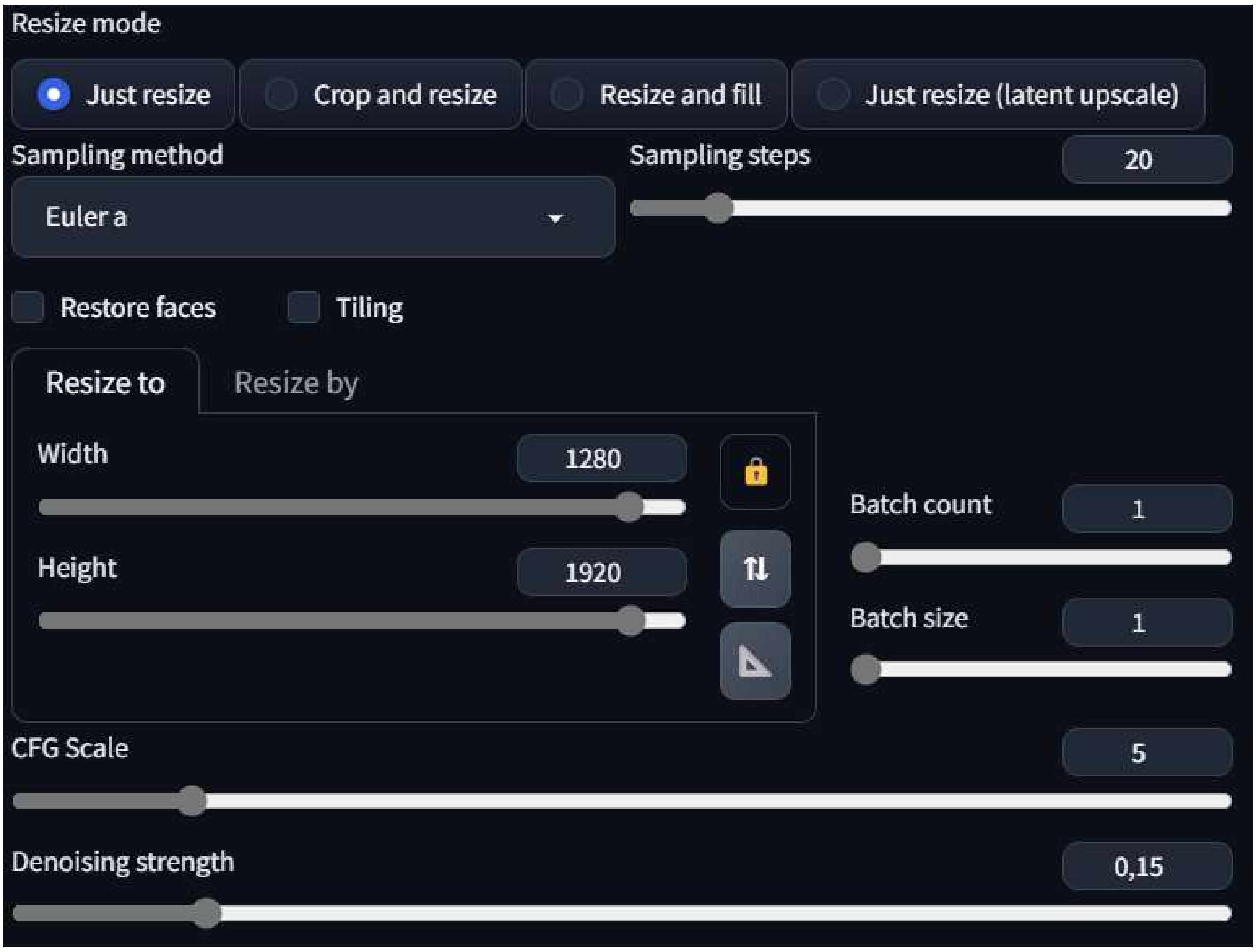
Now we can copy back this picture to Stable Diffusion and let’s run the generation again with a lower denoising strength. After a few generations, I’ve chosen this one for the next step:



Now, before we upscale this picture to 1280x1920 (repeating the same steps we've done before) we need to fix the imperfections introduced with the previous generation. Let's copy the picture back to Krita and clean it. I won't detail step-by-step of the process, as it's not really related to the topic of the tutorial. In my case, the cleaned picture looks like this:



Now we can paste the picture in the img2img tab and change the settings to match these:


Resize mode

(● Just resize) ○ Crop and resize ○ Resize and fill ○ Just resize (latent upscale)

Sampling method: Euler a

Sampling steps: 20

☐ Restore faces ☐ Tiling

Resize to | Resize by

Width: 1280

Height: 1920

Batch count: 1

Batch size: 1

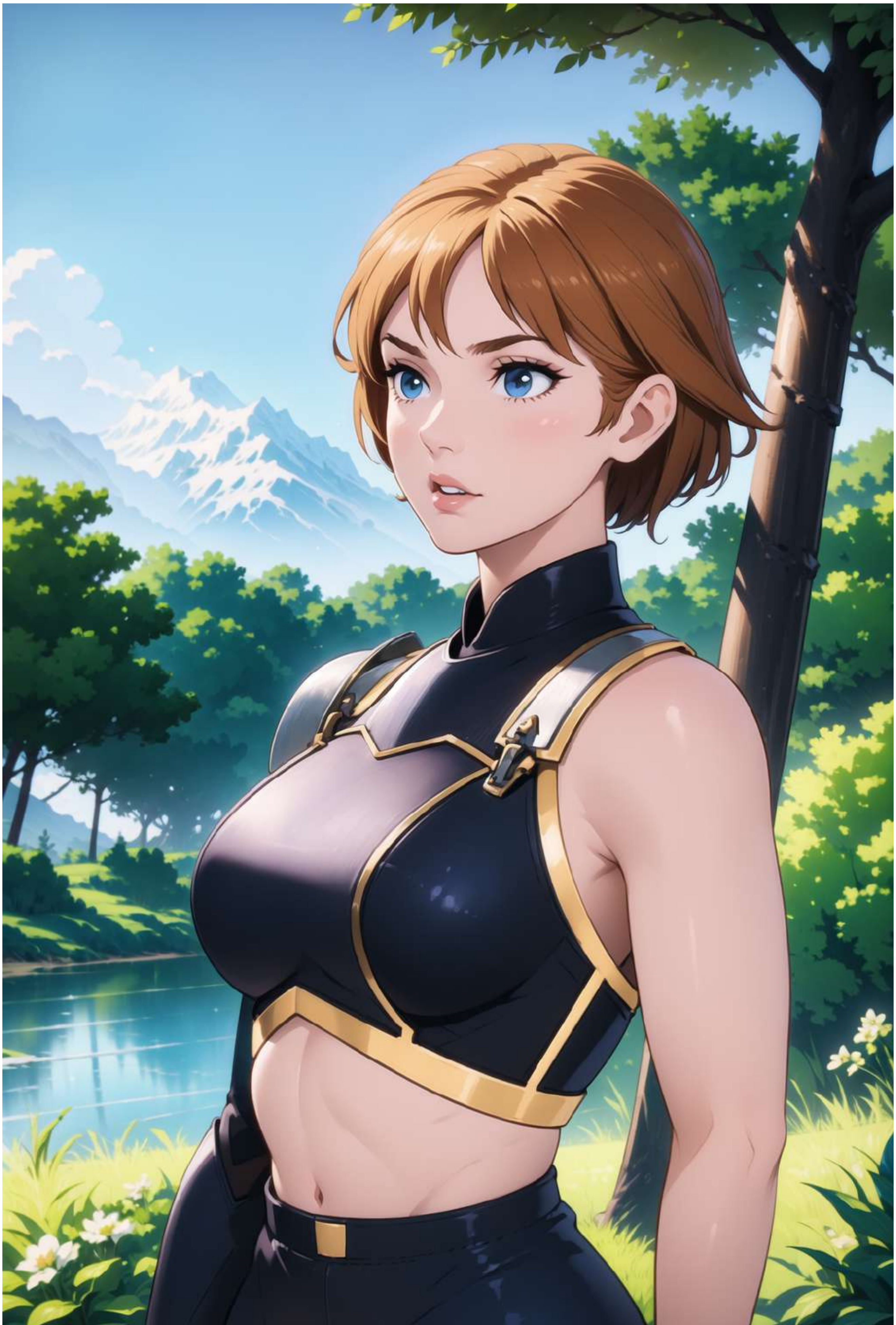
CFG Scale: 5

Denoising strength: 0,15

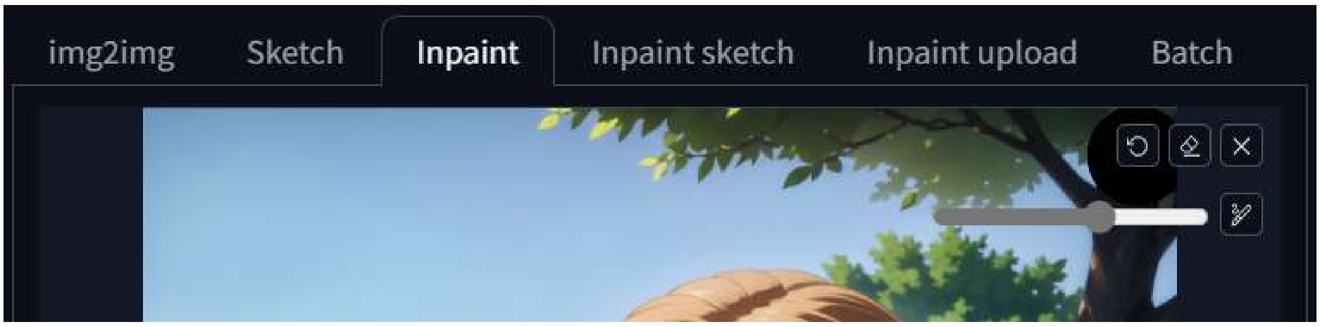
To generate a picture of this size, you'll need a capable GPU. If you receive an out-of-memory error, just reduce the size until you are able to generate the picture (remember to always keep the size ratio). If you create a picture at a lower resolution, you can always upscale it later.

Also notice that the "Denoising Strength" is very low this time. This is because we don't need to make changes to the picture

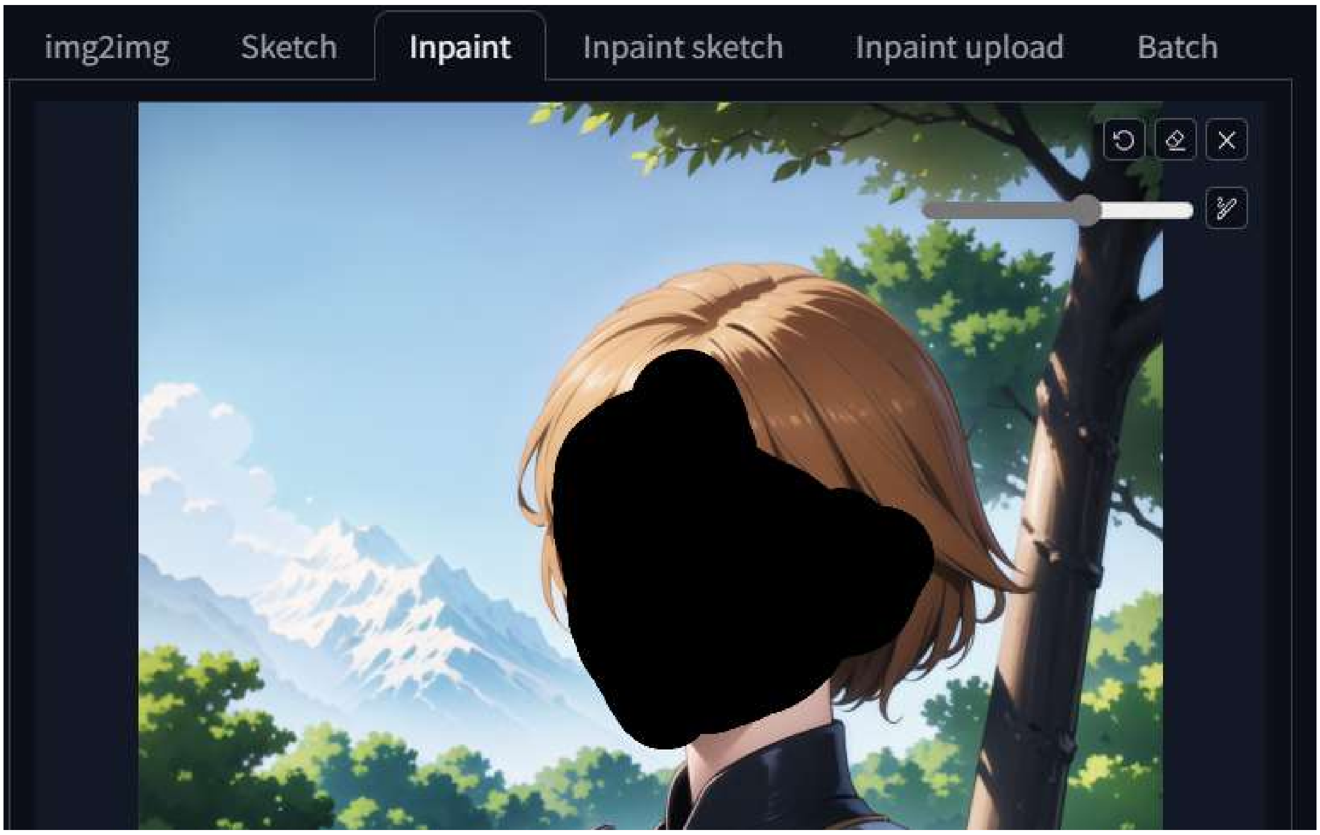
The final result should be something like this:



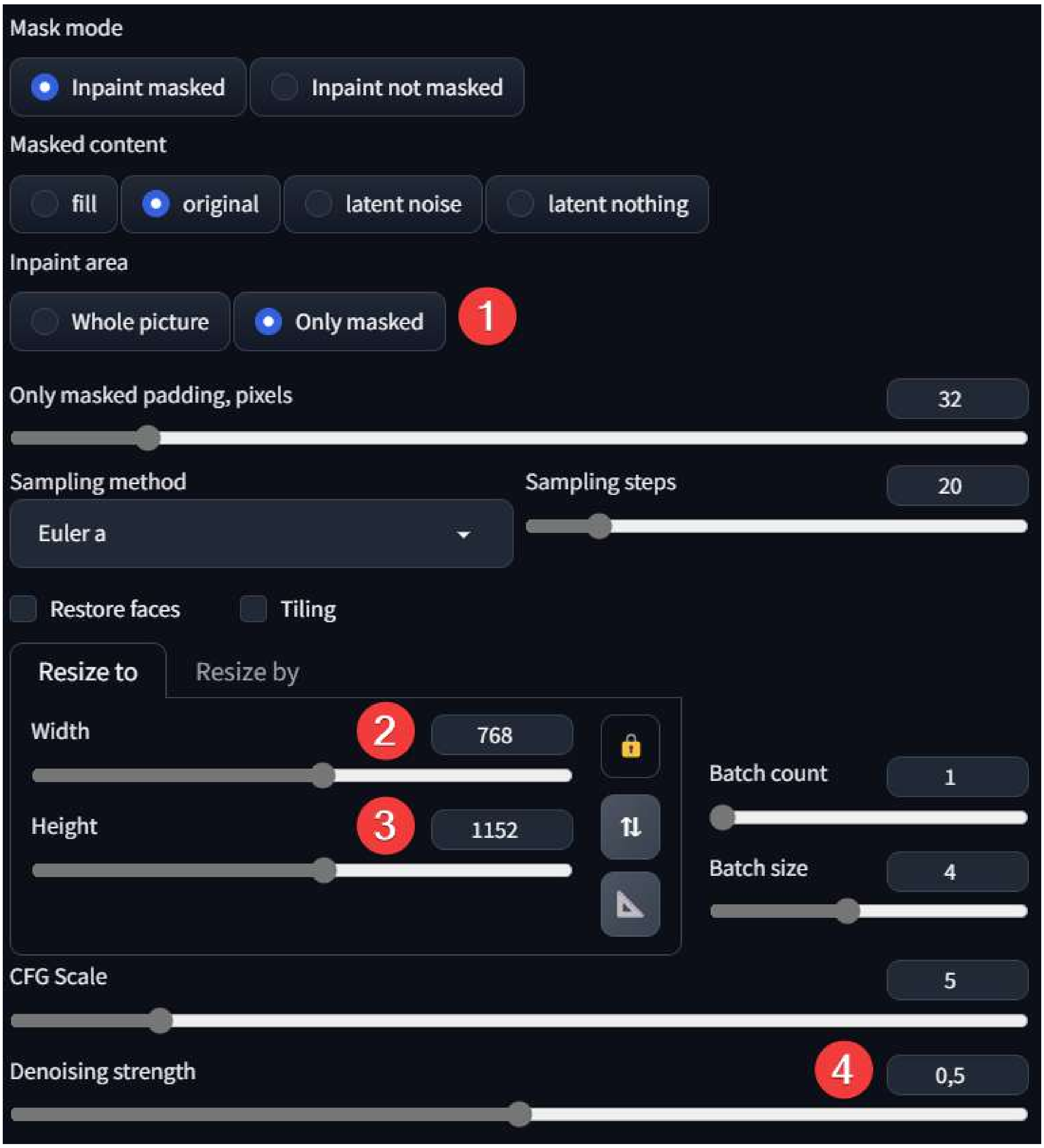
The next step is to improve the facial details. For this step, I also want to change the eye color. To do this, let's start by adding "brown_eyes" to the prompt and the go to the inpaint tab:



Now that we are in the inpaint section, we need to draw a mask on the face of the character (I've made a tutorial about this, check my previous chapter). I usually draw the mask a bit larger than it should be as it seems to output better results. So in my case this is what I've drawn:



Now that we have our mask we need to change a few settings below:



Mask mode
[✓ Inpaint masked] Inpaint not masked

Masked content
fill [✓ original] latent noise latent nothing

Inpaint area
Whole picture [✓ Only masked]

Only masked padding, pixels 32

Sampling method Euler a

Sampling steps 20

Restore faces Tiling

[✓ Resize to] Resize by

Width 768

Height 1152

Batch count 1

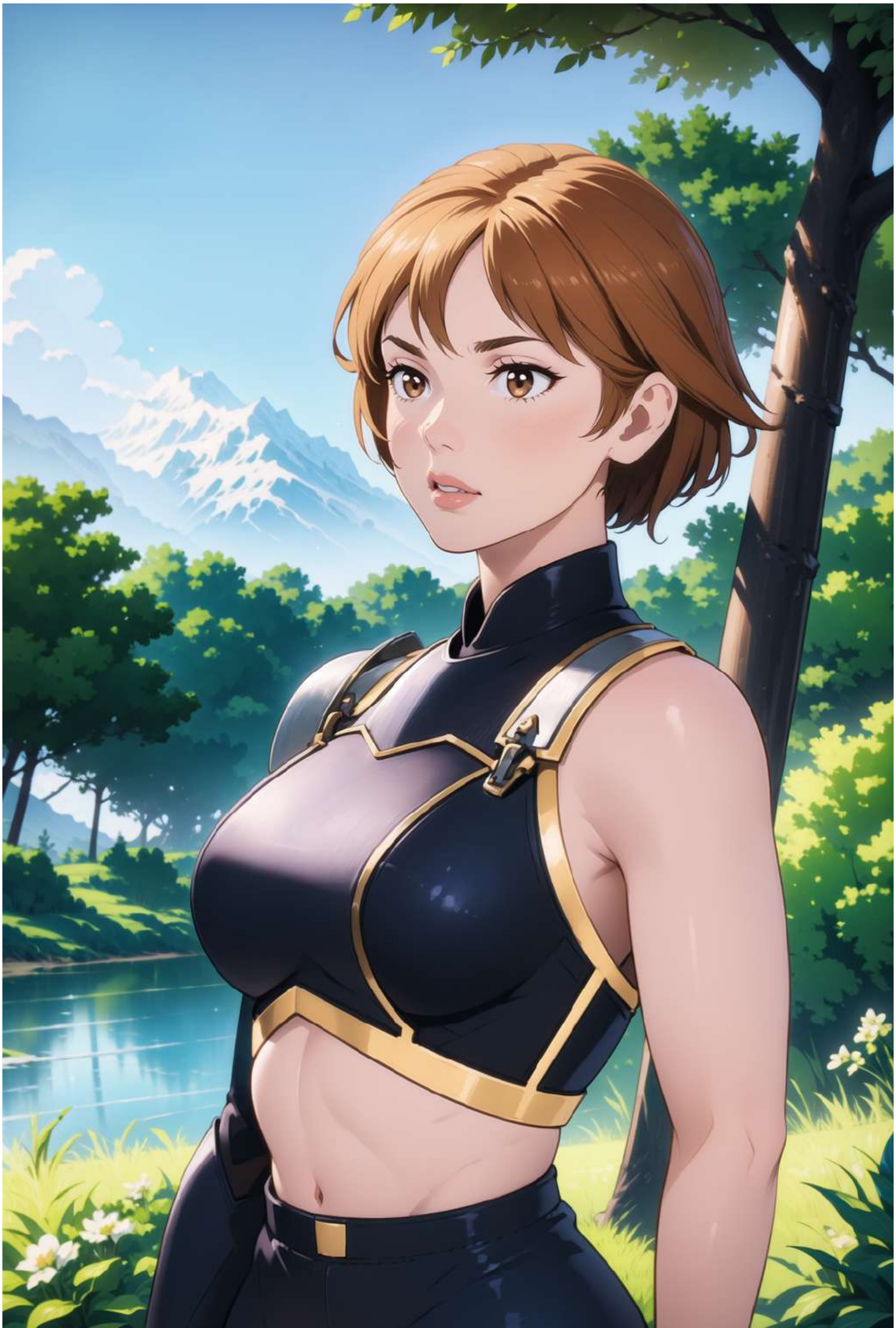
Batch size 4

CFG Scale 5

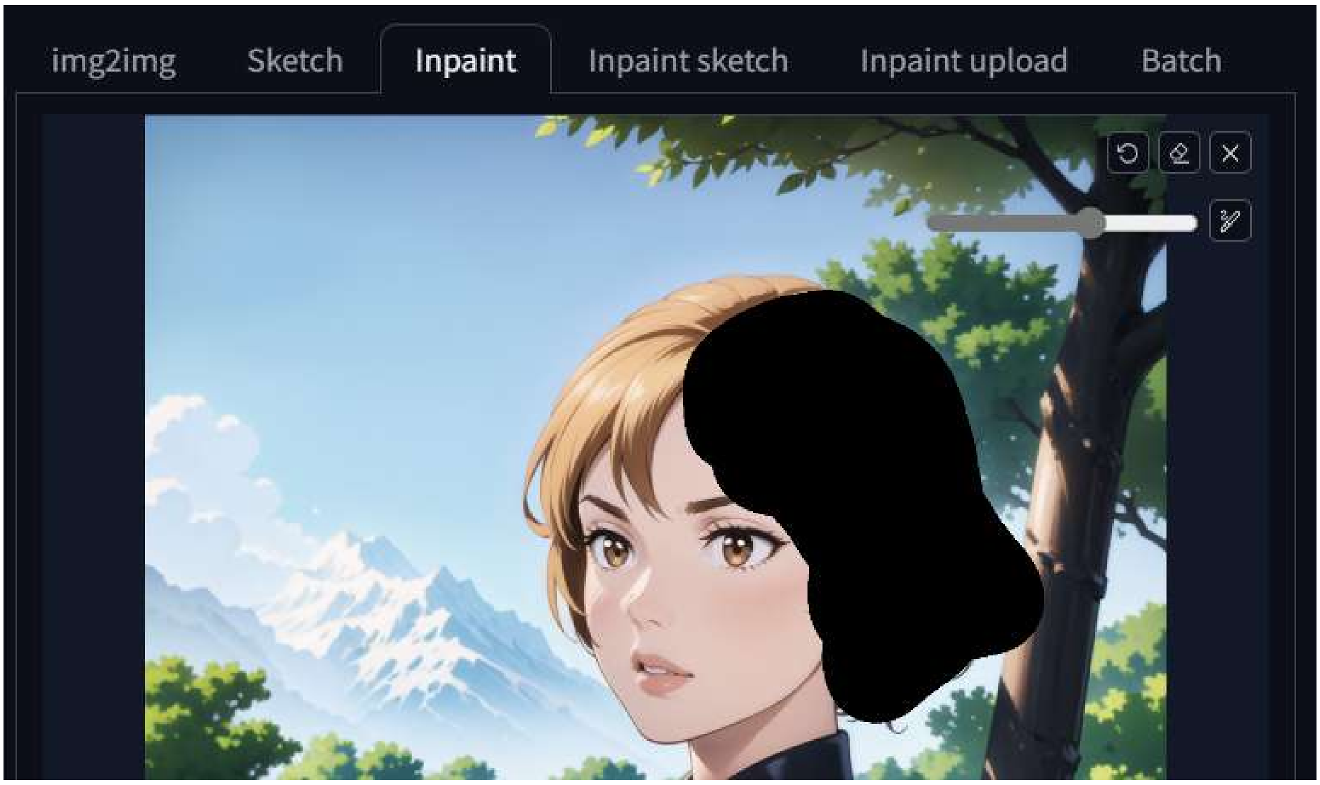
Denoising strength 0,5

1: Set the inpaint area to only masked. This forces SD to inpaint just the area we have selected using the full resolution, giving us a much more detailed output
2–3: Change the size to 768x1152 or 512x768. Using a higher resolution usually gives you much more details, but is prone to making mistakes. Using a resolution closer to 512px is safer.
4: Change the denoising to 0.5. This is my go-to denoising strength in most of the cases. If you feel that the image is not changing enough, increase it otherwise decrease it.

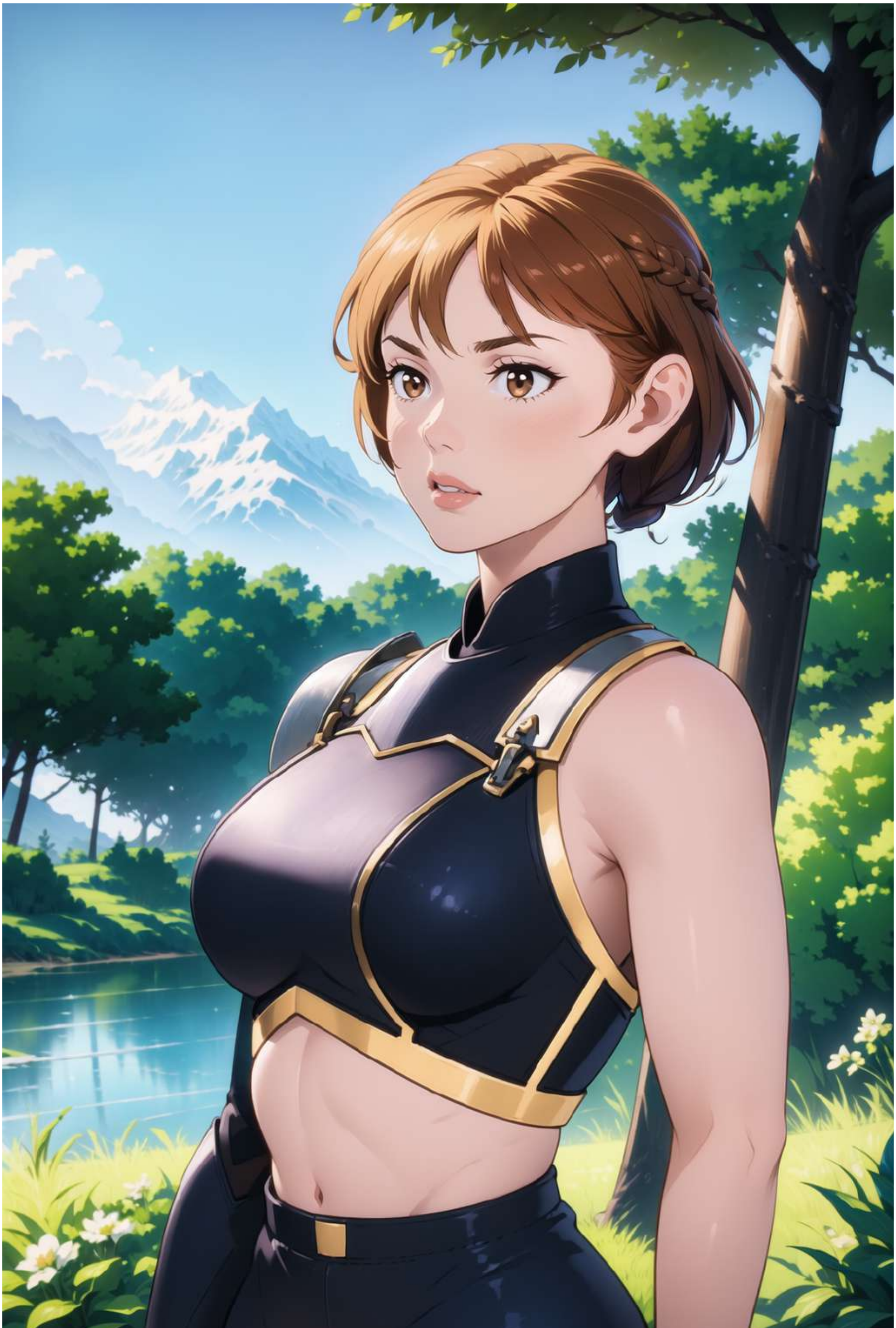
Now, as usual, run the generation and pick the best result. In my case this is now the picture:



Now I also want to change the hairstyle a little bit. Let’s create a mask on the left side of her head:



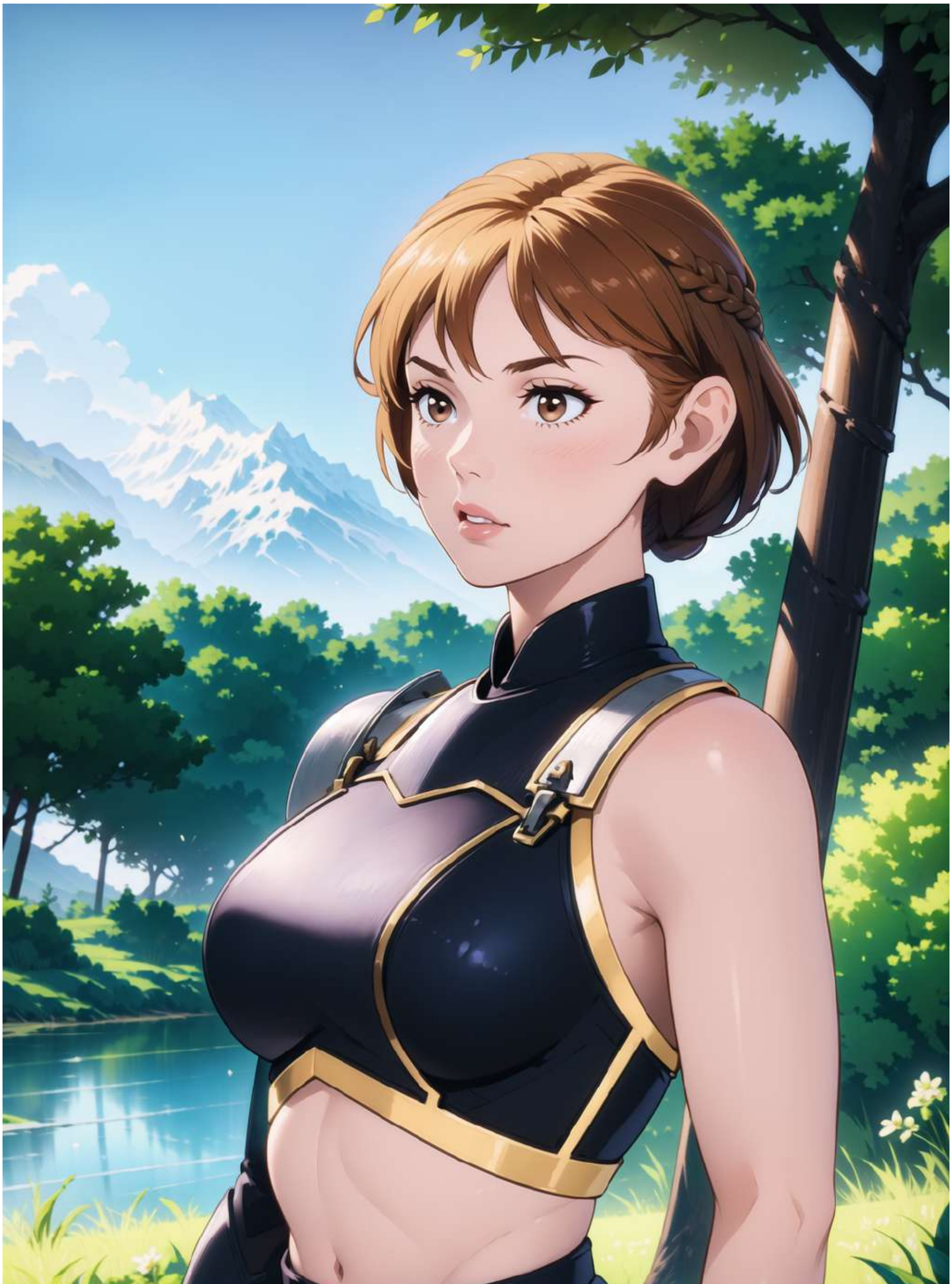
Now add to the prompt (braid:1.2), change the denoising strength to a higher value (0.6–0.65) and the size to 512x768. Then run the generation (you’ll probably need to run this a few times before getting a good result):



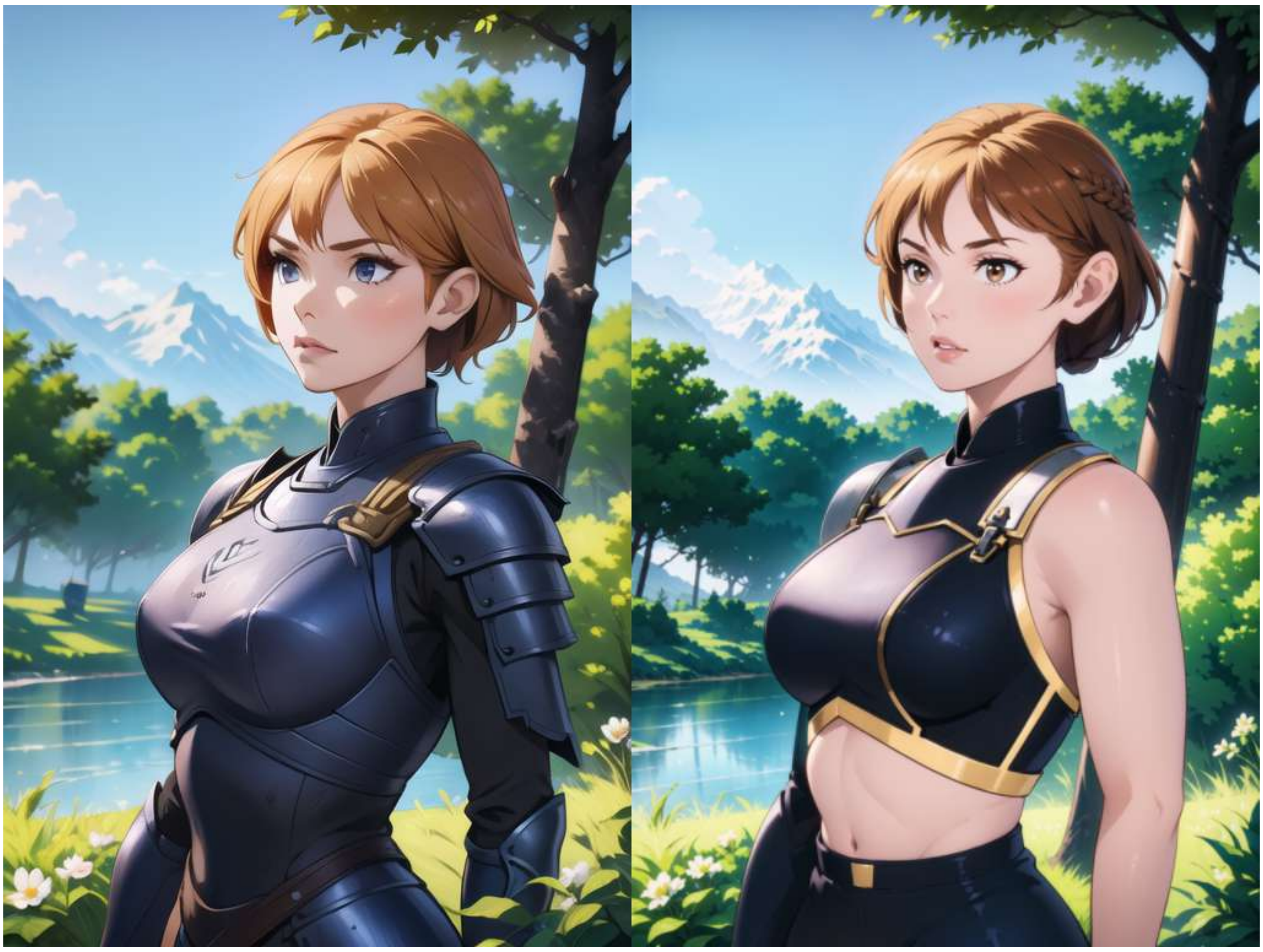
We could go on and add more details, but for this tutorial these edits should give you an idea. We are finally reaching the final step: cleaning the image. This step is divided in two different activities:

- Cleaning the image manually using Krita (or the editor of your choice)
- (Optional) Passing the final image to a final img2img generation with a very low denoising strength

Here’s the complete picture:



And here’s a comparison with the original picture

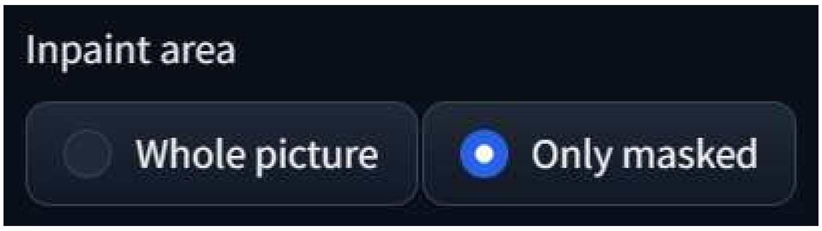


Stable Diffusion Ultimate Guide pt. 7: Tips and Tricks

In this chapter we are going to take a look at a few tips and tricks to use in Stable Diffusion and also some scripts, embeddings and extensions that can help you create awesome pictures.

Tips and Tricks

Use "Only Masked" inpainting for better quality



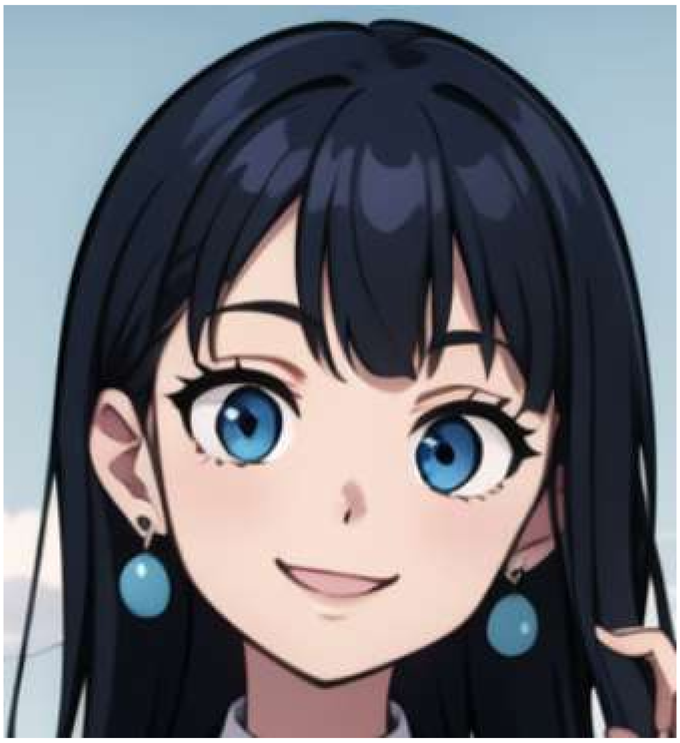
Inpaint area

Whole picture [✓ Only masked]

One of the most overlooked features for a beginner when Inpainting is the "Only Masked" checkbox. This option change how the Inpainting process works:

- **Whole Picture**: This means that the input of the inpainting process will be the whole picture, while the output will be the original picture with the replaced inpainted part
- **Only Masked**: This means that the input of the inpainting process will be only the masked area (with some padding). By doing this the masked area will be upscaled to match the resolution defined inside the UI, then it will diffuse the inpainted area, and finally it will place back the generated picture inside the original one.

Using the "Only Masked" checkbox is really useful when adding details to a high resolution picture. For example, here's a face generated from img2img using a resolution of 768x1152:



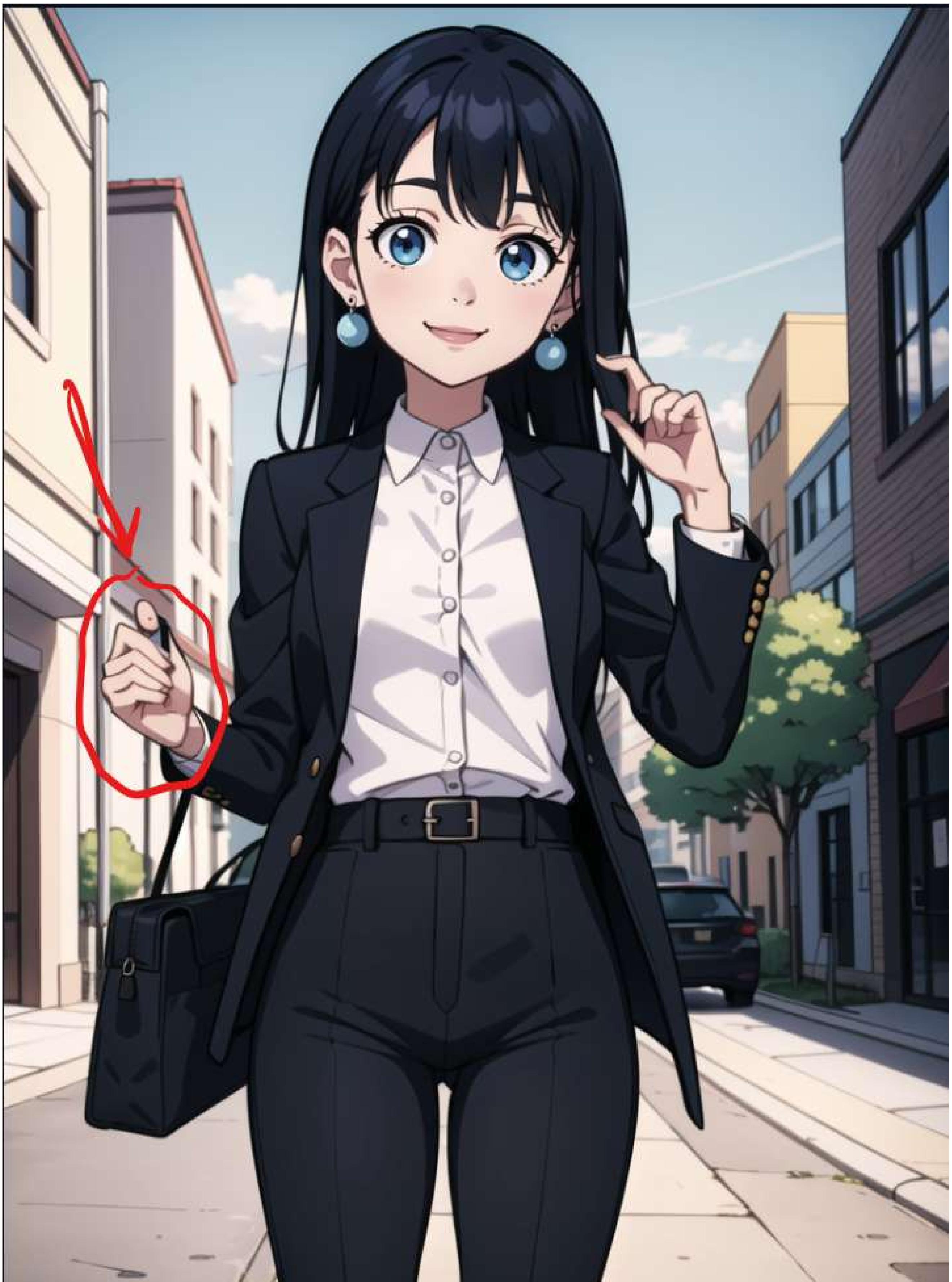
And here's the same resolution output but inpainted using the "Only Masked" option (512x768):

As you can see, the difference is incredible!

Enlarge the mask when inpainting small details (Only Masked)

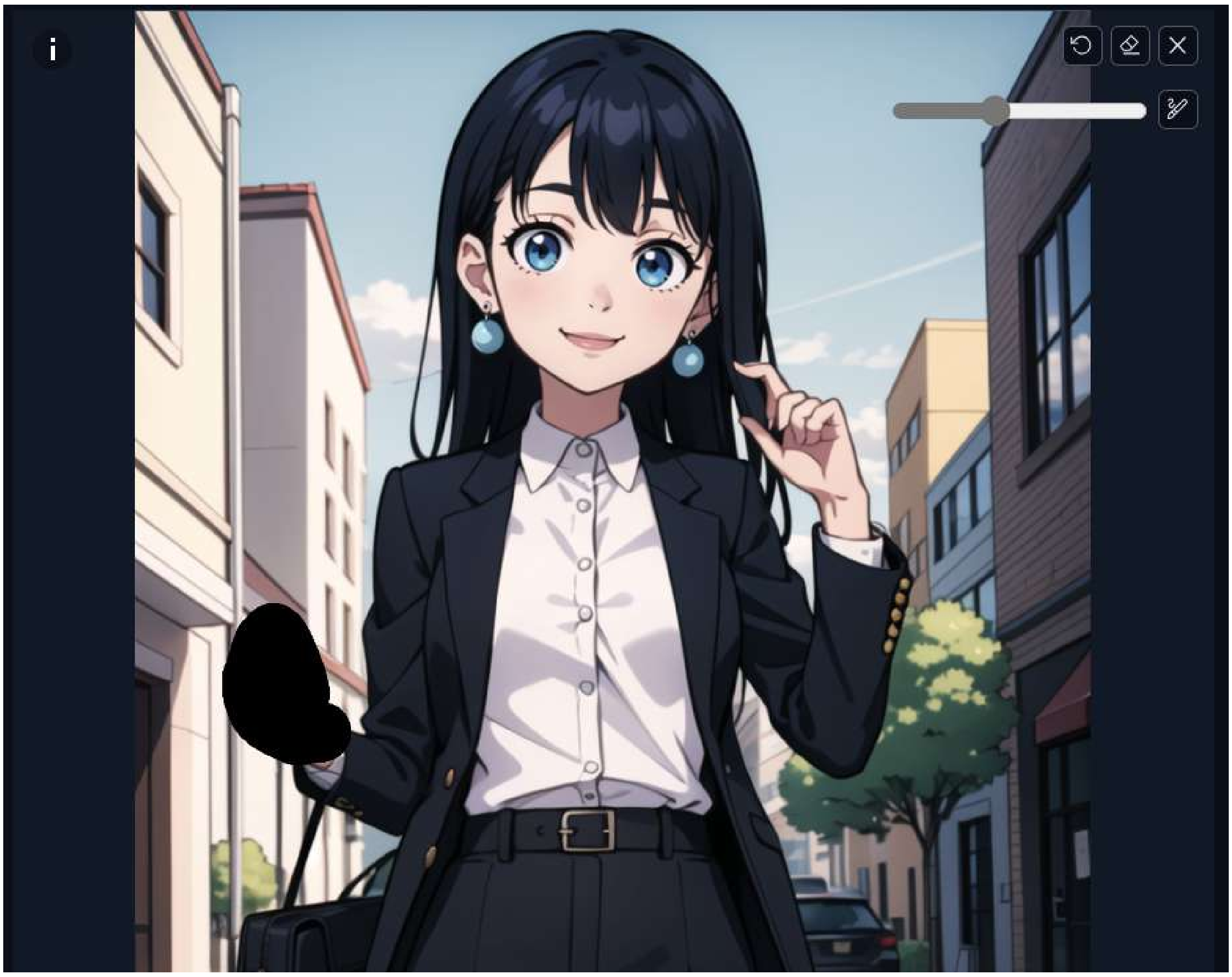
When working with smaller details using the “Only Masked” option, it can be difficult to generate a picture that fits the masked selection.

For example, if we try to apply a mask around the right hand of this character:

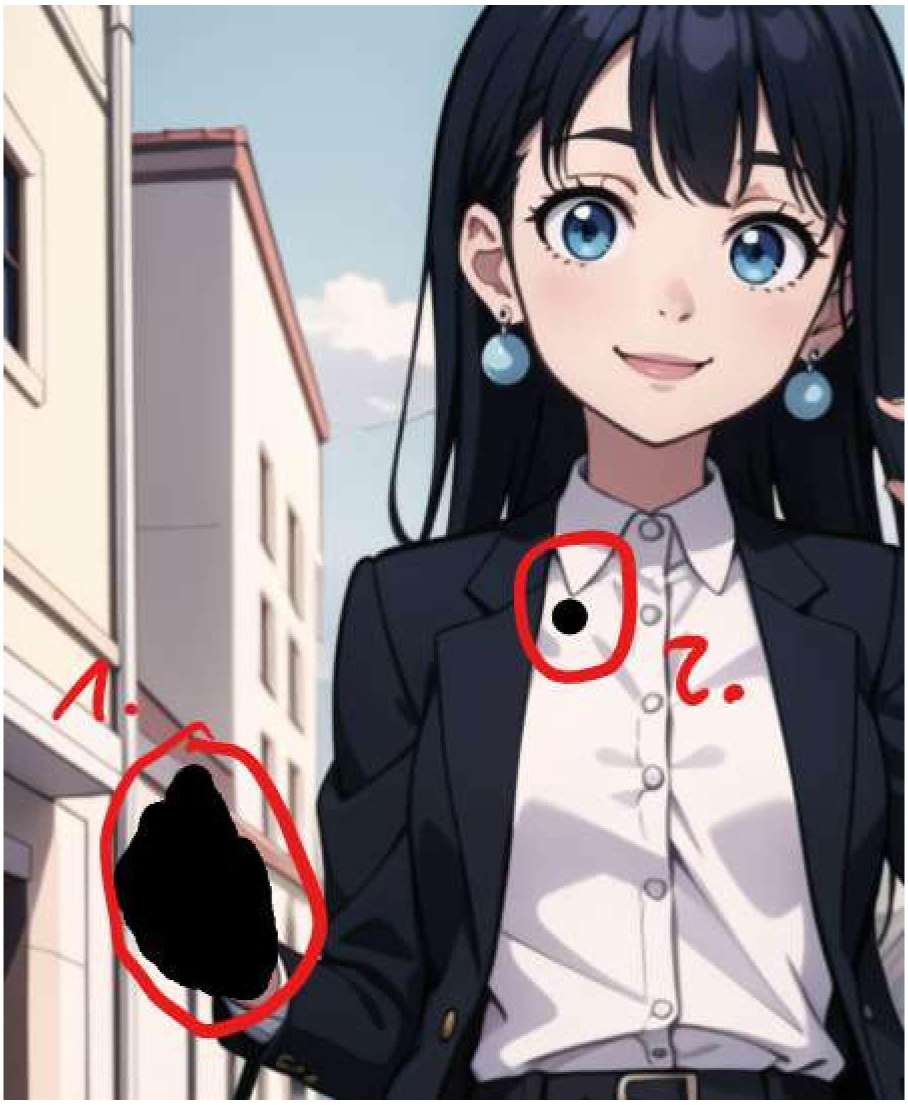


You’ll get results like these:

These are pretty funny, but not very useful. You could also change the prompt to match what you'd like to see in the inpainted part, but I often find myself using this trick. Let's say that we have this mask applied to the picture:



To make it work correctly, we just need to enlarge the mask to let Stable Diffusion see more of the picture. To do this, we add a small dot in a place far from the hand (usually near the face).:



And here's the final result:

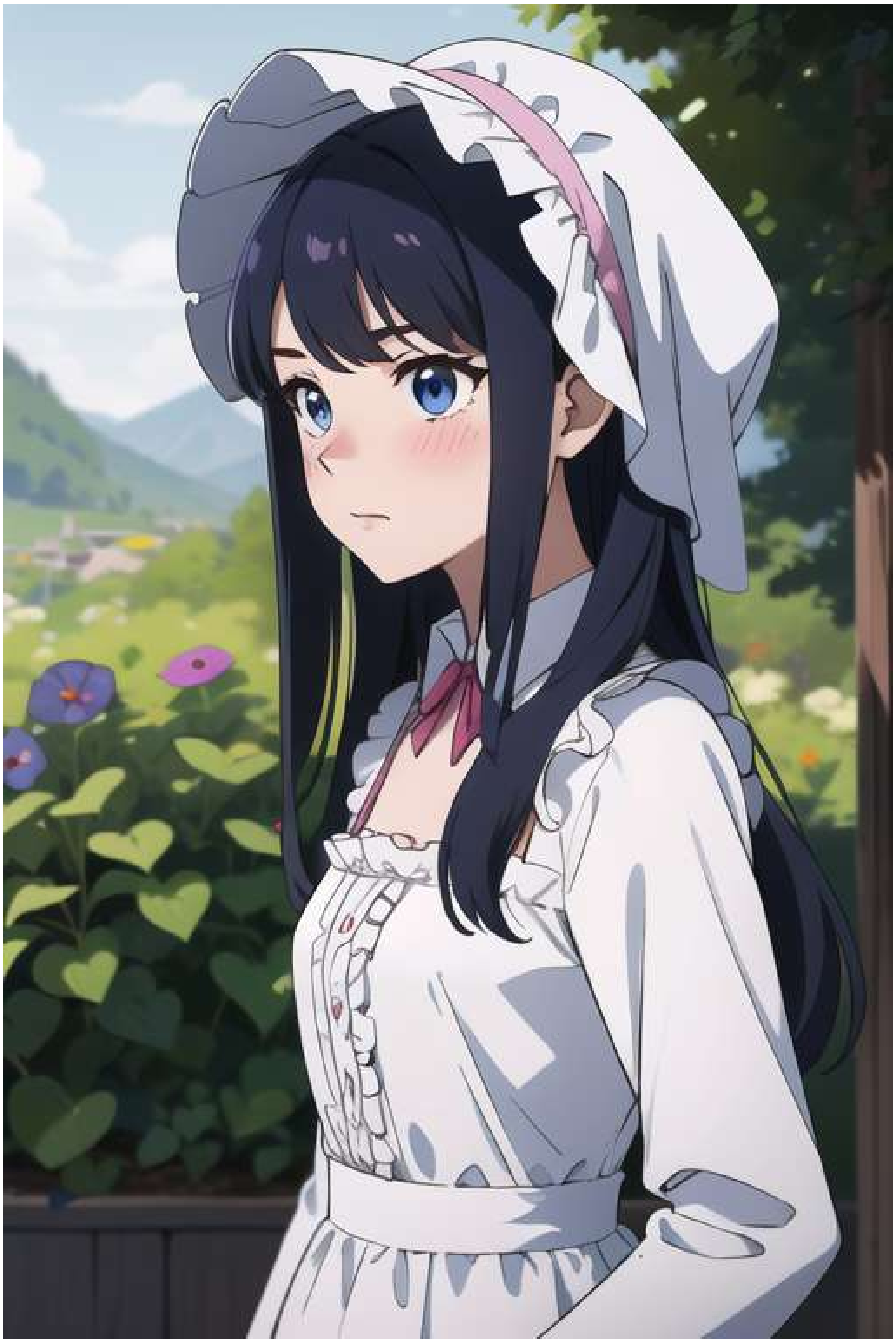
Experiment with high LoRA weights

The checkpoints of Stable Diffusion tend to be repetitive when working with similar prompts. Using some LoRA with a high weight is a good way to create some variations of the images. Let's see an example:

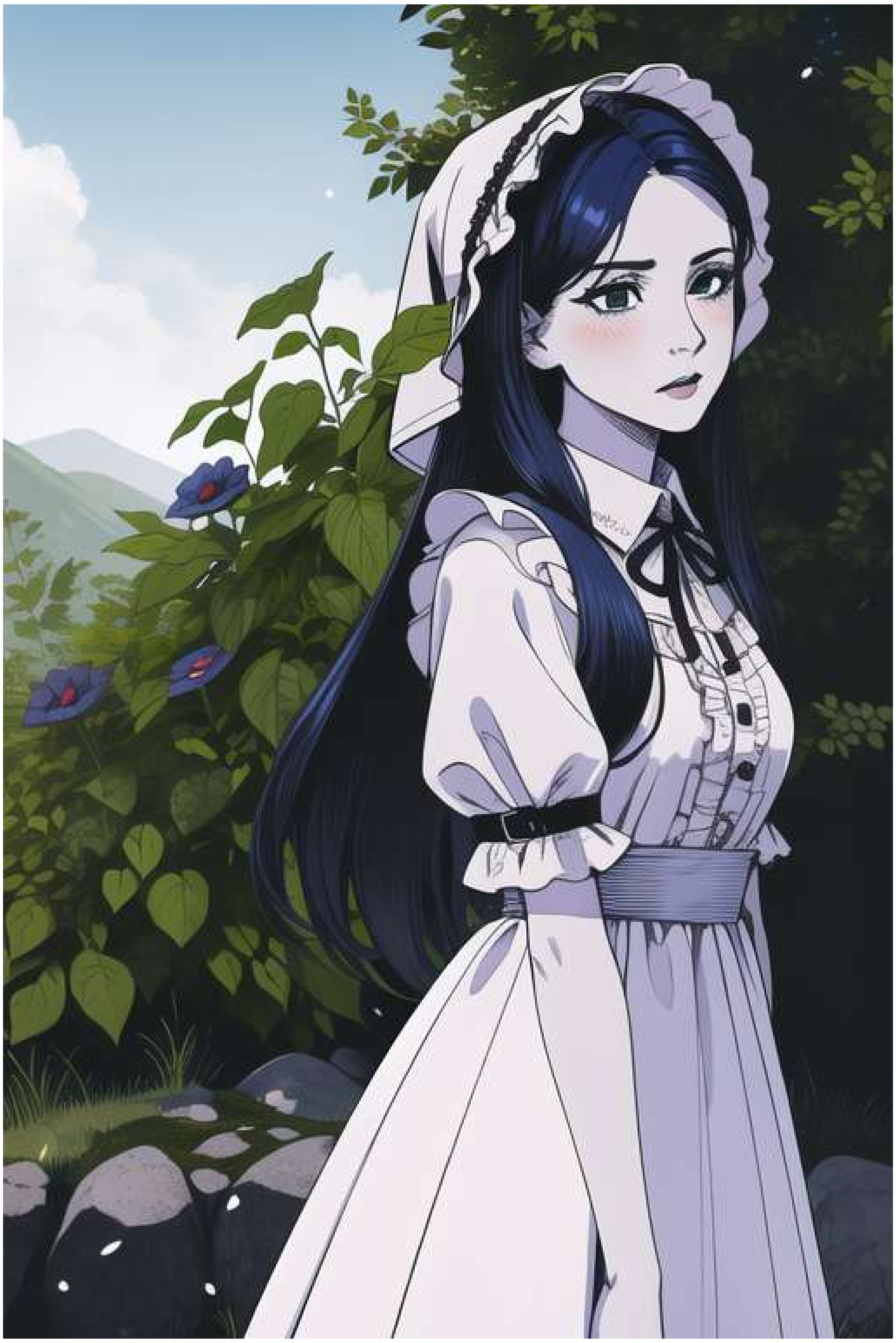
Here's the prompt:

detailed_background,outdoors,sidelocks,long_hair,bonnet,high_collar,frilled_dress,orokudesu,blush,morning_glory,absurdres,blue_hair,white_shirt,frills,1girl,highres,dress

If I use my Mistoon_Anime v2 model, I'll get a bright and colorful picture, but what if I want something gloomier? Let's firstly take a look at a random picture generated without LoRA:



As you can see, the picture has a particular style which is influenced by my checkpoint. Let's try using my custom LoRA sketchy to create something with a different vibe. I'm going to use a high LoRA weight of 0.8 because I'm more interested on the overall style than the distortion caused by the LoRA:



Now that we have our picture, let’s use this in the img2img tab, but without the LoRA we’ve applied:

As you can see, we have now a picture with the style of the original checkpoint, but with the vibe of the LoRA.

You can do the same thing when using high (or low) values of CFG!

LoRAs and Embeddings

Some negative prompts

When starting with Stable Diffusion, a lot of people underestimate the importance of the Negative Prompt. This feature can drastically change the final result, especially when used with good embeddings.

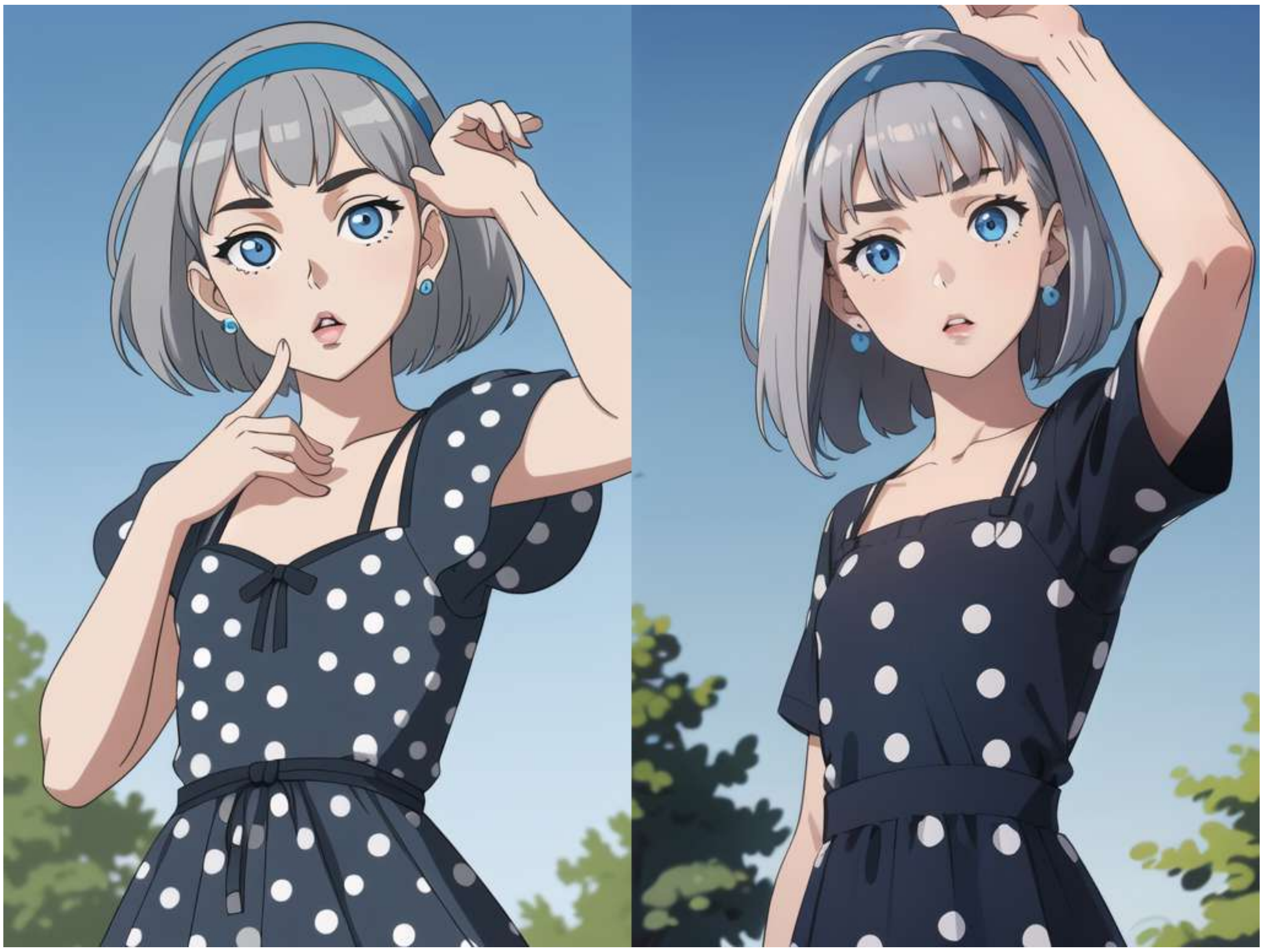
Here’s my standard negative prompt:

```
By bad artist -neg badhandv4 notxt nobg verybadimagenegative_v1.3
```

This prompt uses the following embeddings:

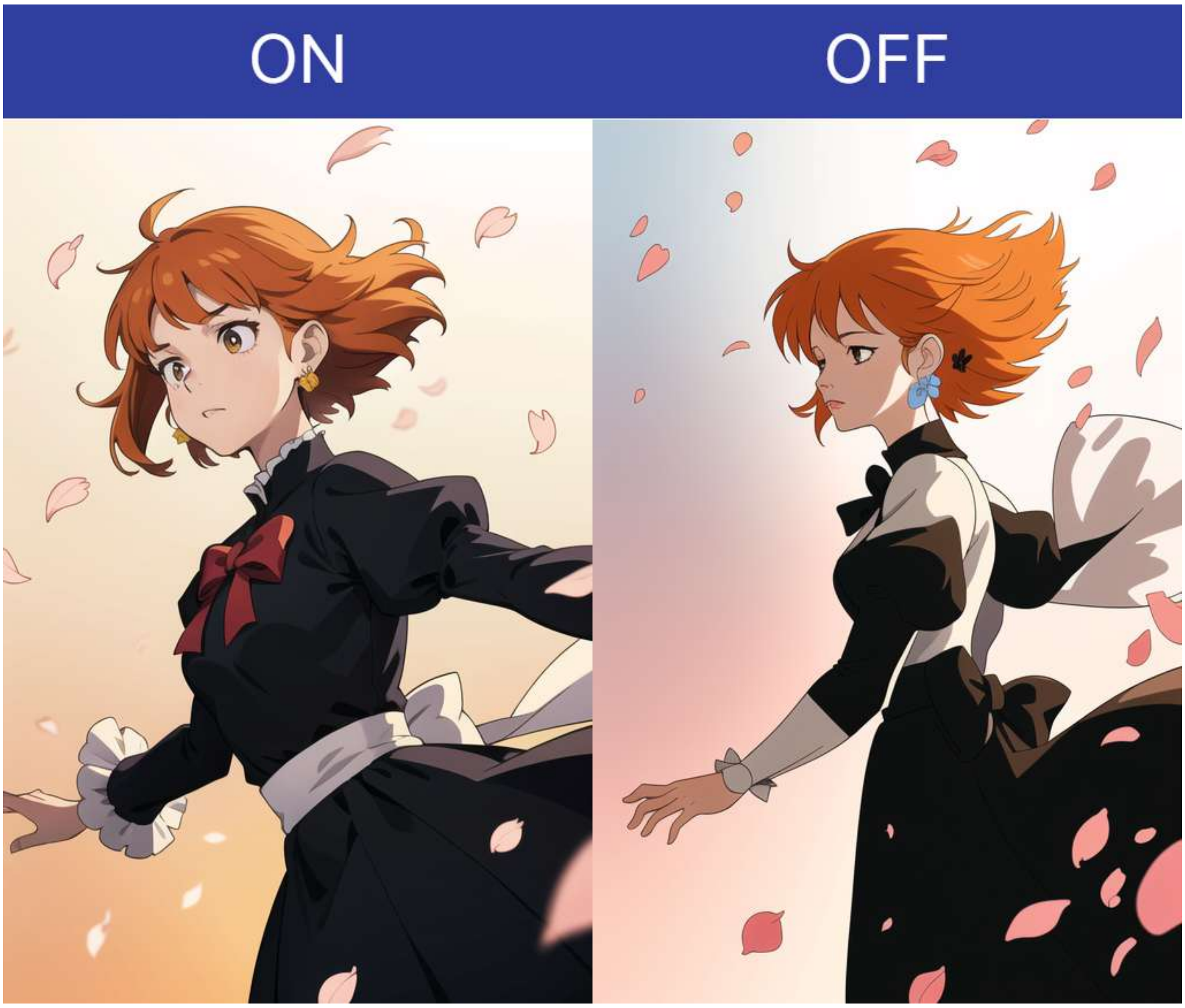
- badhandv4 — AnimeIllustDiffusion — badhandv4 | Stable Diffusion Textual Inversion | Civitai
- [Bad artist Negative embedding — Bad artist anime | Stable Diffusion Textual Inversion | Civitai](https://civitai.com/models/5224?modelVersionId=6057
- veryBadImageNegative — veryBadImageNegative_v1.3 | Stable Diffusion Textual Inversion | Civitai
- My textual inversions: https://civitai.com/models/33873?modelVersionId=25820

And here’s an example of a picture without and with the embeddings:



You can also find my personal embeddings here:

[Inzaniak's Lazy Textual Inversions - BadPic | Stable Diffusion Embedding | Civitai]



Lately I've been using just BadPic when creating pictures, without any other embedding or negative prompt.

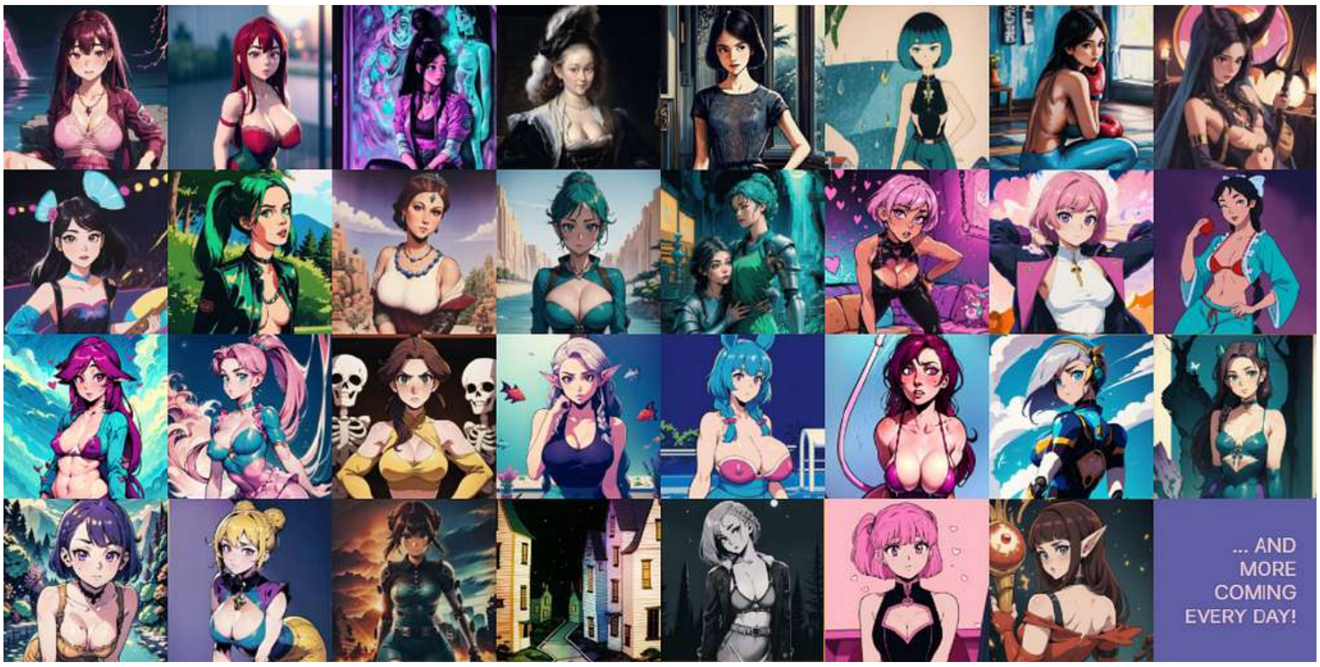
Add Details to your pictures

If you want to add details to your pictures, you can use a “detailer” LoRA like this one: [Detail Tweaker LoRA (细节调整LoRA) - v1.0 | Stable Diffusion LoRA | Civitai]()

With this, you can increase the amount of details in your pictures without drastically changing their composition.

Change the style of your art

LoRAs are an incredible tool for any Stable Diffusion prompt engineer. There are a lot of LoRAs available and they are incredibly easy to train. If you check my CivitAI+Patreon pages, I have more than 100 different LoRAs available:



Here are the links if you are interested:

- CivitAI: Inzaniak Creator Profile | Civitai
- Patreon: https://www.patreon.com/Inzaniak

Scripts and Extensions

Ranbooru

GitHub - Inzaniak/sd-webui-ranbooru

Ranbooru is my custom script that can be used to parse tags from random pictures from different boorus. This script is growing daily and has now a lot of different features that you can check out on the Github above.

This script is incredibly useful when I want to mass-test my LoRAs and checkpoint on a high amount of different tags.

ControlNet

GitHub - lllyasviel/ControlNet: Let us control diffusion models!

ControlNet is probably the best extension available for Stable Diffusion. With this extension, you can apply a series of ML models to influence the diffusion process. For example, you can use a picture of a particular pose as an input to change your generation, or you can use a black and white sketch to create a colored version of that picture.

Once you’ve learned the basics, I highly recommend to also check out the stable-diffusion-art comprehensive guide:

ControlNet v1.1: A complete guide - Stable Diffusion Art

Aspect Ratio Helper

GitHub - thomasasfk/sd-webui-aspect-ratio-helper: Simple extension to easily maintain aspect ratio…

If you want to easily maintain aspect ratio while changing dimensions in AUTOMATIC1111 webui, you might want to try this extension.

It is a simple extension that adds a dropdown of configurable aspect ratios, to which the dimensions will auto-scale. You can also lock the aspect ratio of the current dimensions or the current image. You can also scale the dimensions by percentage or by maximum value.