



Part 9

Stable Diffusion

进阶用法



Contents

1. 高清修复 (Hi-Res Fix)
2. SD放大 (SD Upscale)
3. 附加功能放大
4. Tiled 放大
5. 局部重绘
6. 图片对比

四种主要的放大修复方案

如何在SD中让图片变得更清晰&更有细节?

01

文生图：高清修复

“打回重画，再来一副”

分两步处理图像，先生成一个低分辨率版本，再根据它与指定的放大算法，再生成一个高分辨率版本，从而在不更改构图的情况下丰富细节。

02

图生图：SD放大

“分几块画，拼到一起”

根据指定的放大倍数，将图生图的图像拆分成若干小块按固定逻辑重绘，再拼合成一张大图，可以实现在低显存下绘制大尺寸图片。

03

附加功能放大

“简单方法，随时可用”

利用各种放大算法，在图像生成后对其进行单独的放大处理，使之拥有更高的分辨率尺寸。

04

Tiled放大

低显存放大最好的方法

Tiled Diffusion 会按照一定规则将画面分割成一个个小块，然后分别的去绘制，再使用巧妙的方式无缝拼接叠在一起，并借助一点点算法来优化整体效果。

1. 高清修复 (Hi-Res Fix)

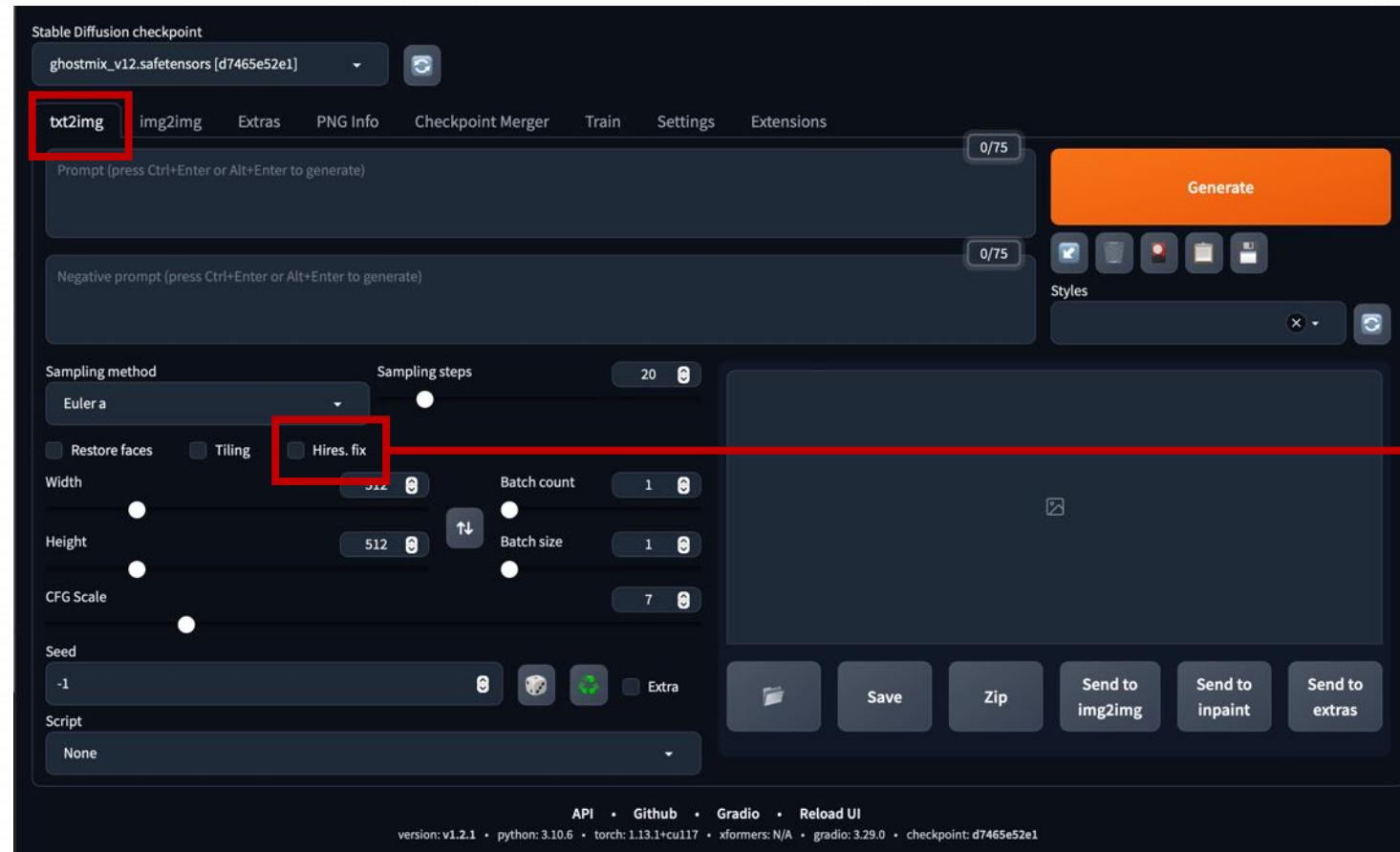
1.1 高清修复的概念 (Hi-Res Fix)

1.2 高清修复基本操作流程

1.3 放大算法差异简单辨析

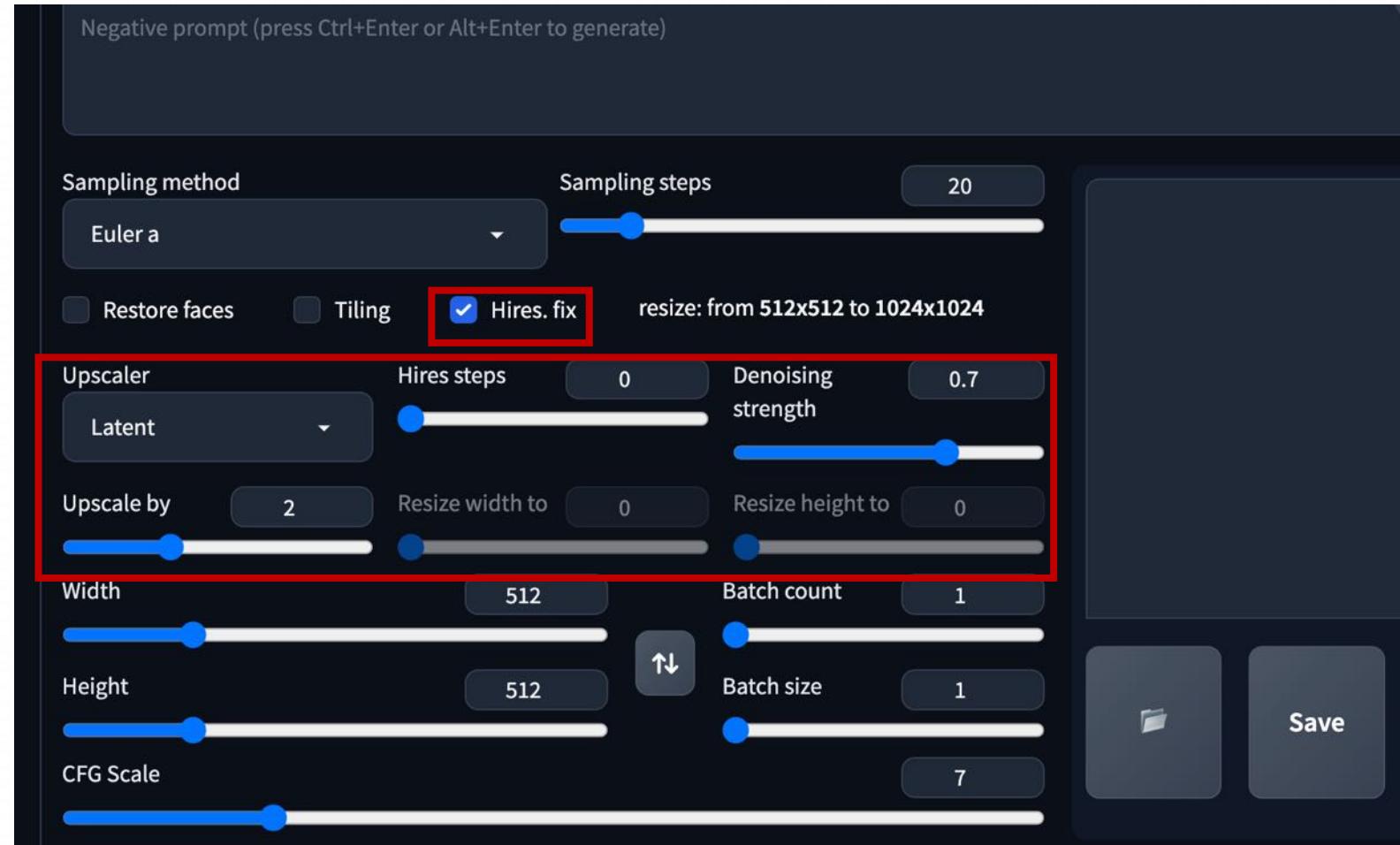
1.1 高清修复的概念 (Hi-Res Fix)

高清修复本质就是把低分辨率成品“图生图”一次，打回潜空间内，重新去噪，从而得到一张更大的图片。



文生图页面中有一个Hires.fix，这个功能有好几个叫法，有的是叫“高分辨率修复”，也有的翻译为“超分辨率”

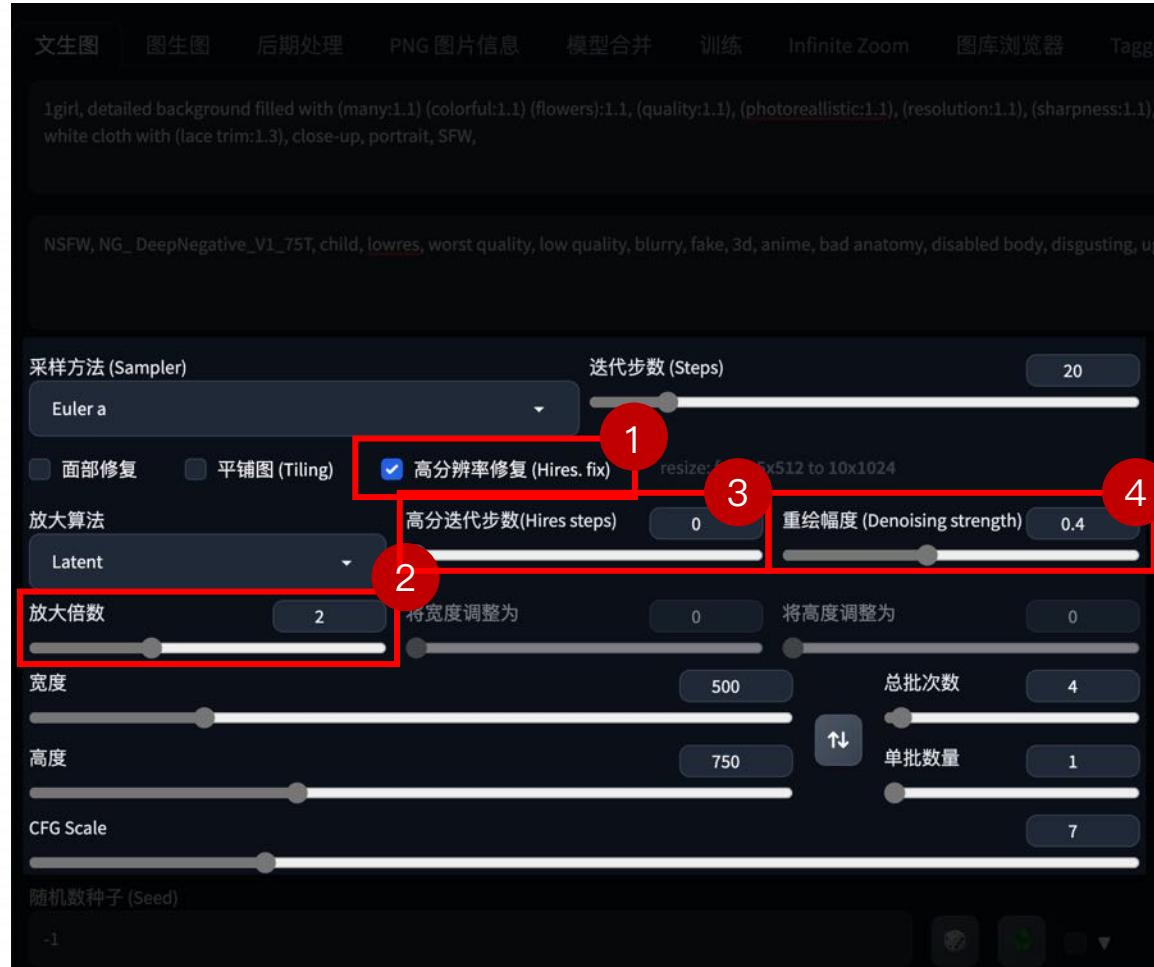
1.1 高清修复的概念 (Hi-Res Fix)



Hires.fix(高分辨率修复)

尺寸设置指你将这个图片由原始分辨率放大到多少，可以按倍数定，也可以按具体数值。要保持画面大的结构不发生变化，最好在denoising strength (重绘幅度)不要大于0.5)

1.2 文生图：高清修复基本操作流程



操作步骤：

1. 勾选“高分辨率修复”
2. 放大倍数：你将这个图片由原始分辨率放大到多少，倍数或具体数值尝试原图2倍
3. 高分迭代步数：保持0默认值，它和我们设置的采样迭代次数（20）保持一致
4. 重绘幅度：要保持画面大的结构不发生变化，最好不要超过0.5
5. 重绘算法

1.2 文生图：高清修复基本操作流程

Prompt

```
1girl, detailed background filled with (many:1.1)
(colorful:1.1) (flowers):1.1, (quality:1.1),
(photorealistic:1.1), (resolution:1.1), (sharpness:1.1),
(cinematic lighting), ISO 400,
white cloth with (lace trim:1.3), close-up, portrait, SFW,
```

翻译：一位女孩，细节的背景里有许多鲜艳的花朵，（高）质量，照片般真实的，（高）分辨率，锐化，电影级光照，景深，佳能EOS R6拍摄（+相机参数），
白色带蕾丝装饰的衣服，特写，头像



Negative Prompt

```
NSFW, NG_DeepNegative_V1_75T, child, lowres,
worst quality, low quality, blurry, fake, 3d,
anime, bad anatomy, disabled body, disgusting, ugly,
text, watermark,
```

图片设置参数

采样方法: Euler

迭代步数: 30

分辨率 500*750

CFG: 7

1.2 文生图：高清修复基本操作流程



这张图片存在的问题：

- 局部模糊
- 皮肤缺乏质感

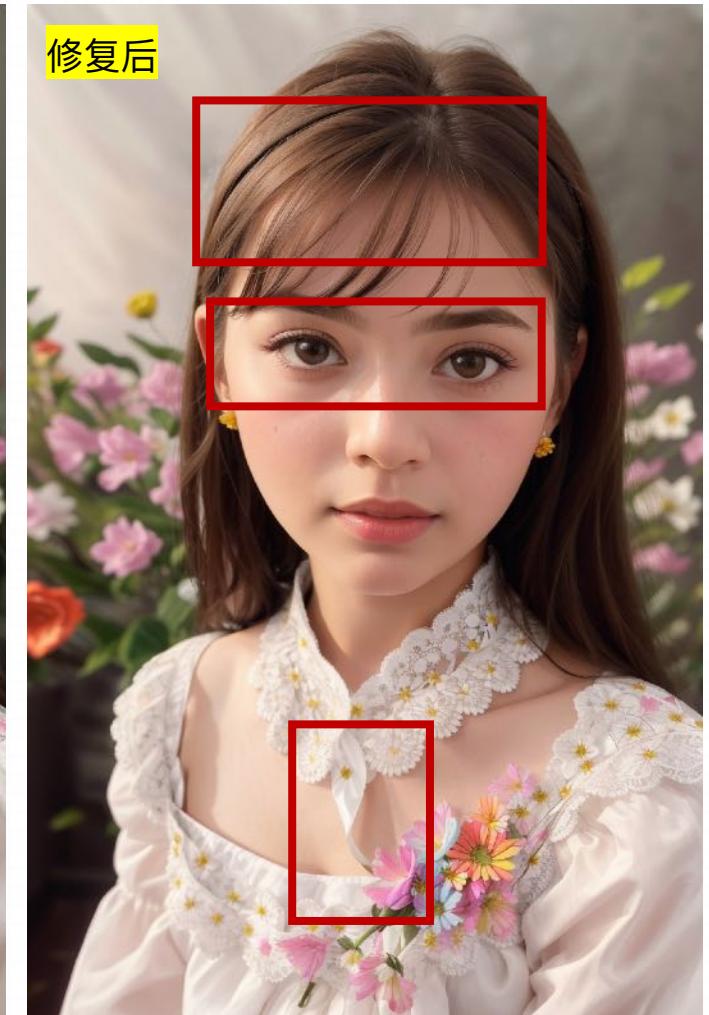
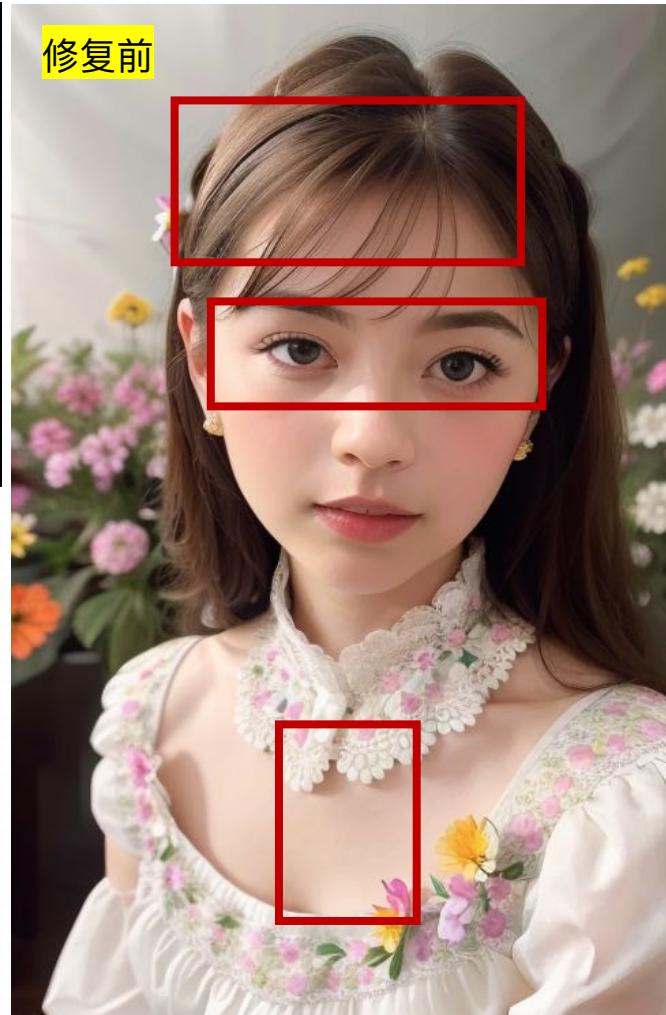
出现这种问题的原因：

分辨率设置太低时，AI没有足够的操作空间让画面产生足够的细节

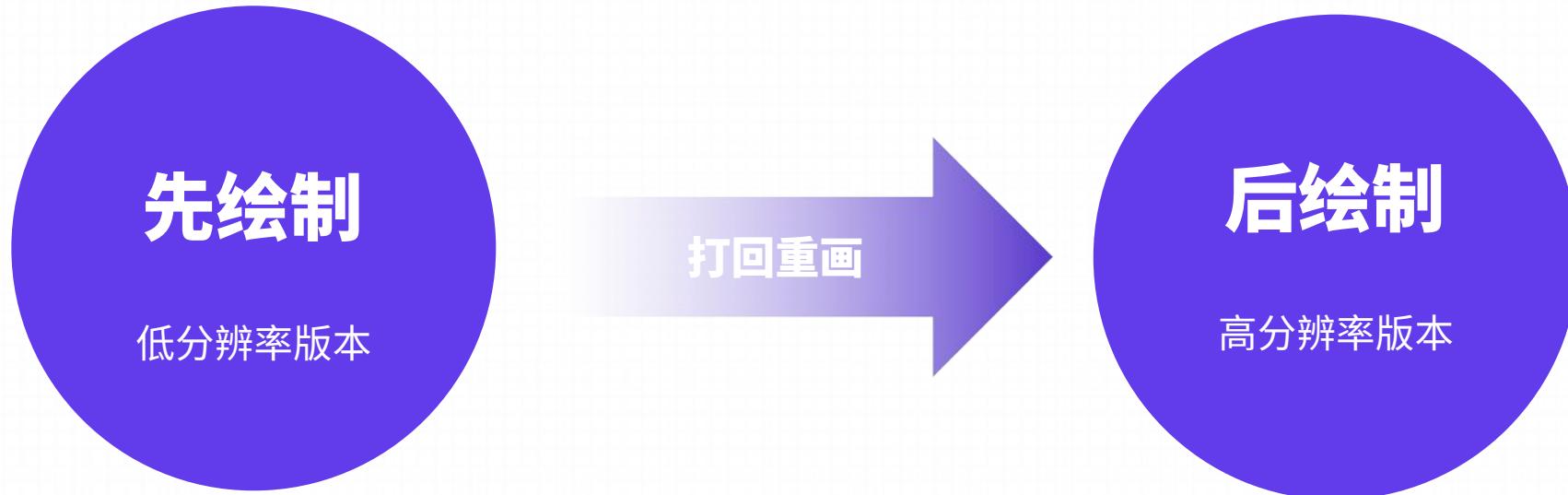
直接调高分辨率会出现的问题：

1. 会爆显存
2. 出现多人多头的问题

1.2 文生图：高清修复基本操作流程



1.2 文生图：高清修复基本操作流程



高清修复的图片会经历两步骤，首先是绘制一个低分辨率的版本，然后再绘制一个高分辨率的版本。

它的本质就是把这个低分辨率成品“图生图”一次，打回潜空间内，重新去噪，从而得到一张更大的图片。

1.2 文生图：高清修复基本操作流程

高清修复是一个对显存影响比较大的操作，画一次等于常规情况下画两到三次，即便是高清修复，也没办法突破显存的限制。



如图所示，分辨率为500*750，放大倍数为4倍，正常情况数值应该为2000*3000，但是系统识别到显寸，将分辨率只放大到了1025*1537。

- 我们可以先在低分辨率下反复生成图片，当确定了一个合适的画面以后，可以再借助固定随机种子的办法，来提高分辨率进行高清修复。
- 但高清修复有时候也会出现一些画面原本不存在的内容，如果只是为了放大图像，那么安全的参数是0.3–0.5，但如果你想赋予AI更多的想象力和发挥空间，那就设置在0.5 – 0.7

1.2 文生图：高清修复基本操作流程



优势：

- 不会改变画面构图(经由随机种子固定)
- 稳定的克服多人、多头等由分辨率产生的问题；
- 操作简便、清晰、直观。

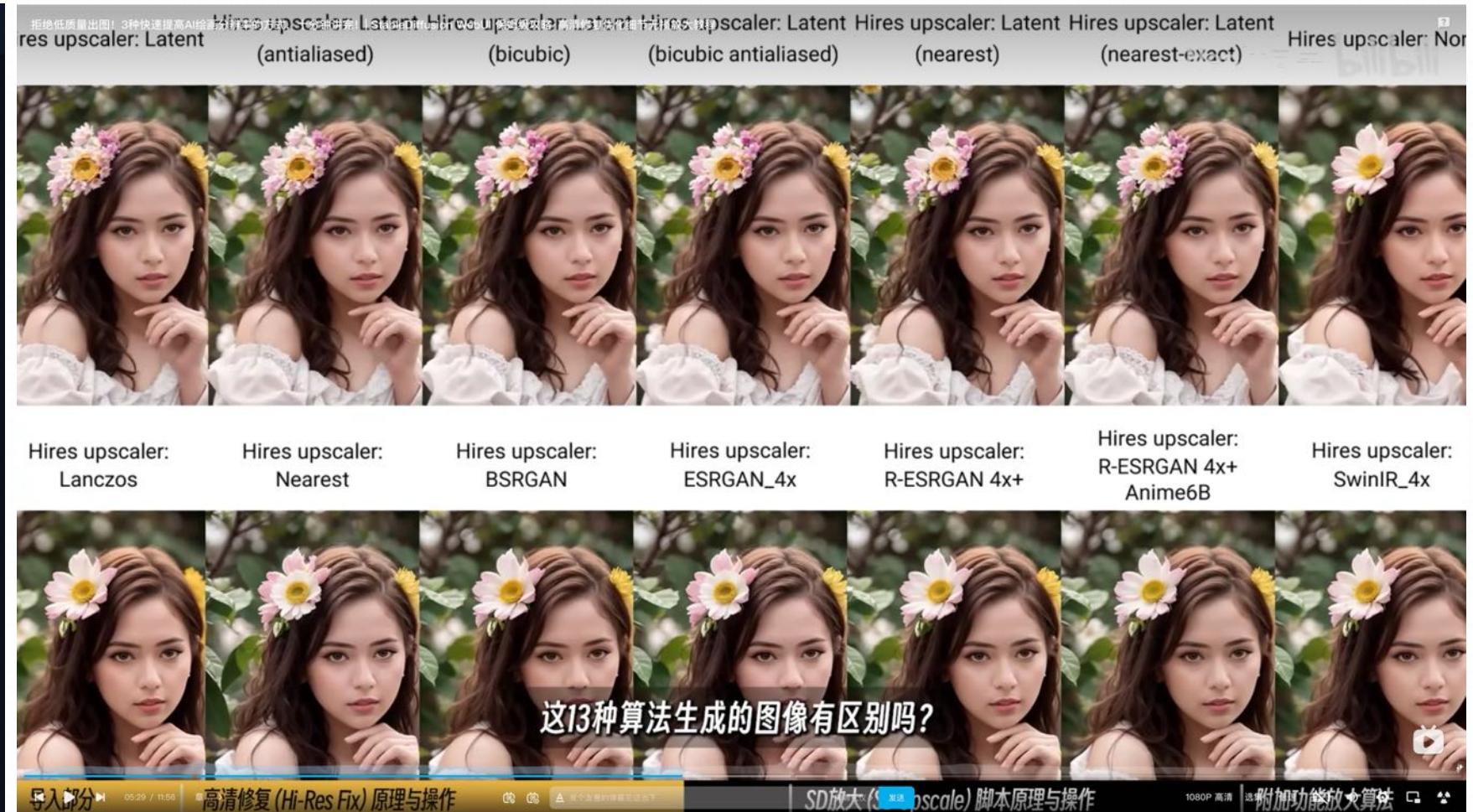
劣势：

- 仍然受到最大显存的限制；
- 计算速度相对较慢；
- 偶尔“加戏”，出现莫名的额外元素。

1.3 放大算法差异简单辨析

这些算法的区别在于决定了将低分辨率的版本的图片“打回重画”的时候，我们要如何操作？

<input checked="" type="checkbox"/> Latent
Latent (antialiased)
Latent (bicubic)
Latent (bicubic antialiased)
Latent (nearest)
Latent (nearest-exact)
无
Lanczos
近邻插值
ESRGAN_4x
LDSR
R-ESRGAN 4x+
R-ESRGAN 4x+ Anime6B
ScuNET GAN
ScuNET PSNR
SwinIR 4x



1.3 放大算法差异简单辨析

ESRGAN 算法

ESRGAN是Enhanced Super-Resolution Generative Adversarial Network的缩写，是一种基于生成对抗网络（GAN）的图像超分辨率算法。其主要思想是通过学习低分辨率（LR）图像与其高分辨率（HR）对应物之间的映射，来实现从LR图像到HR图像的映射过程，从而实现图像的超分辨率。相较于传统的基于插值的超分辨率算法，ESRGAN可以生成更加清晰、细节更加丰富的高分辨率图像。ESRGAN的训练数据集通常包括低分辨率图像及其对应的高分辨率图像，其训练过程中通过生成器网络（Generator）和判别器网络（Discriminator）相互对抗，以提高生成器的超分辨率效果。

LDSR 算法

LDSR是一种用于图像超分辨率的深度学习算法，其全称为“Deep Laplacian Pyramid Super-Resolution”。LDSR算法通过学习图像的低分辨率版本和高分辨率版本之间的关系来实现图像的超分辨率。LDSR算法采用了一种名为“深度拉普拉斯金字塔”的方法，该方法可以将原始图像分解为多个图像金字塔，以便更好地捕捉图像的细节和结构。LDSR 算法的核心思想是使用深度学习网络来学习输入图像的低分辨率版本与高分辨率版本之间的映射关系。具体来说，LDSR算法将输入图像的低分辨率版本作为网络的输入，将高分辨率版本作为网络的输出，并训练网络以最小化两者之间的差异。LDSR算法采用了深度卷积神经网络（DCNN）来实现这一目标。

LDSR算法具有高精度、高效率、易于实现等优点，在图像超分辨率等领域得到了广泛应用。

1.3 放大算法差异简单辨析

R-ESRGAN 4X+

R-ESRGAN 4x+是一种图像超分辨率重建算法。其全称为”Real-Time Enhanced Super-Resolution GenerativeAdversarial Network 4x+”，是一种基于生成式对抗网络 (GAN)的算法，是 ESRGAN (Enhanced Super-ResolutionGenerative Adversarial Networks) 的改进版本之一。它通过引入残差连接和递归结构，改进了 ESRGAN 的生成器网络，并使用 GAN (Generative Acversarial Networks) 进行训练。**R-ESRGAN 4X+ 在提高图像分辨率的同时，也可以增强图像的细节和纹理，并且生成的图像质量比传统方法更高。**它在许多图像增强任务中都取得了很好的效果，比如图像超分辨率，阁像去模糊和图像去噪等。

R-ESRGAN 4X+ AnimeB

P-ESRGAN 4x+ Anime6B 是一种基于超分辨率技术的图像增强算法，主要用于提高动漫图像的质量和清晰度。它基于R-ESRGAN 4x+ 算法，并使用了 Anime6B 数据集进行训练。Anime6B 数据集是一个专门用于动漫图像处理的数据集，其中包含了大量不同风格、不同质量的动漫图像，使得算法可以适应不同类型的动漫图像。

P-ESRGAN 4x+ Anime6B 算法主要通过提取图像特征、生成高分辨率图像和优化来实现图像增强。具体来说，它采用了一种名为残差块的结构来提取图像的高级特征，然后通过反卷积和上采样等方法生成高分辨萃图像。最后，通过对生成的图像进行优化和后处理，进一步提高图像的质量和清晰度。

1.3 放大算法差异简单辨析

SwinIR_4x

SwinIR_4x是一种基于Swin Transformer的图像超分辨率重建算法，可将低分辨率图像放大4倍，生成高分辨率图像。Swin Transformer是一种新型的Transformer模型，相对于传统的Transformer模型，在处理图像等二维数据时，具有更好的并行性和更高的计算效率。SwinIR_4x通过引入Swin Transformer和局部自适应模块（LAM）来提高图像重建的质量和速度。其中，局部自适应模块用于提高图像的局部细节，从而增强图像的真实感和清晰度。SwinIR_4x被广泛应用于计算机视觉领域，特别是图像重建、图像增强和图像超分辨率等方面。

Nearest 算法

Nearest算法（最近邻算法）是一种常见的机器学习算法，用于分类和回归问题。在分类问题中，最近邻算法根据样本之间的距离将新样本分配给最相似的已知样本所属的类别。在回归问题中，最近邻算法通过找到与新样本最相似的已知样本来预测输出值。

最近邻算法通常包括两个步骤：首先计算新样本与已知样本之间的距离或相似度，然后根据最相似的已知样本的标签或值进行预测。

最近邻算法的优点是它非常简单且易于实现，并且对于许多数据集而言效果很好。然而，该算法的缺点是它在处理高维数据和大规模数据时的计算开销非常大，并且对于噪声数据和类别之间的不平衡性表现较差。

1.3 放大算法差异简单辨析

结论

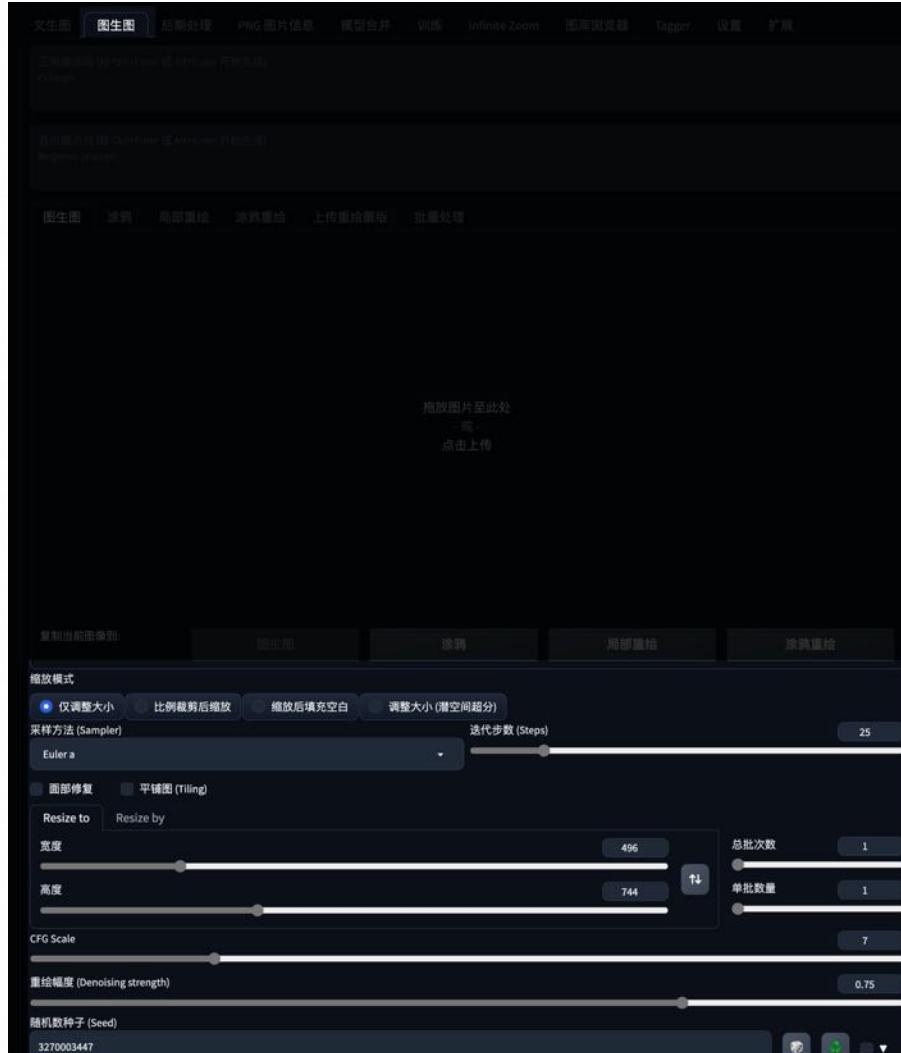
- **ESRGAN系列：** ESRGAN_4x适用于照片写实类（可能出现细节过于锐利的效果，但有些人喜欢这样的风格），R-ESRGAN 4x+（全能型），R-ESRGAN 4x+ Anime6B（适用于二次元），R-ESRGAN-General-WDN-4xV3（适用于厚涂插画）
- **SwinIR_4x：** SwinIR_4x需要表现光影的厚涂插画表现优于照片和二次元的图片放大，但逊于R-ESRGAN-General-WDN-4xV3
- **LDSR：** LDSR照片写实类图片表现很好，插画和二次元图片不行，而且体积很大，速度很慢
- 带有GAN的算法，在同样的重绘幅度下，对细节的保留会更加准确，网上有一种通行的说法，只要使用到这个放大算法，就去选择R-ESRGAN。二次元的图像就选择R-ESRGAN 4x+ Anime6B

2. SD放大 (SD Upscale)

2.1 图生图放大方式简析

2.2 SD放大原理及简单操作

2.1 图生图放大方式简析

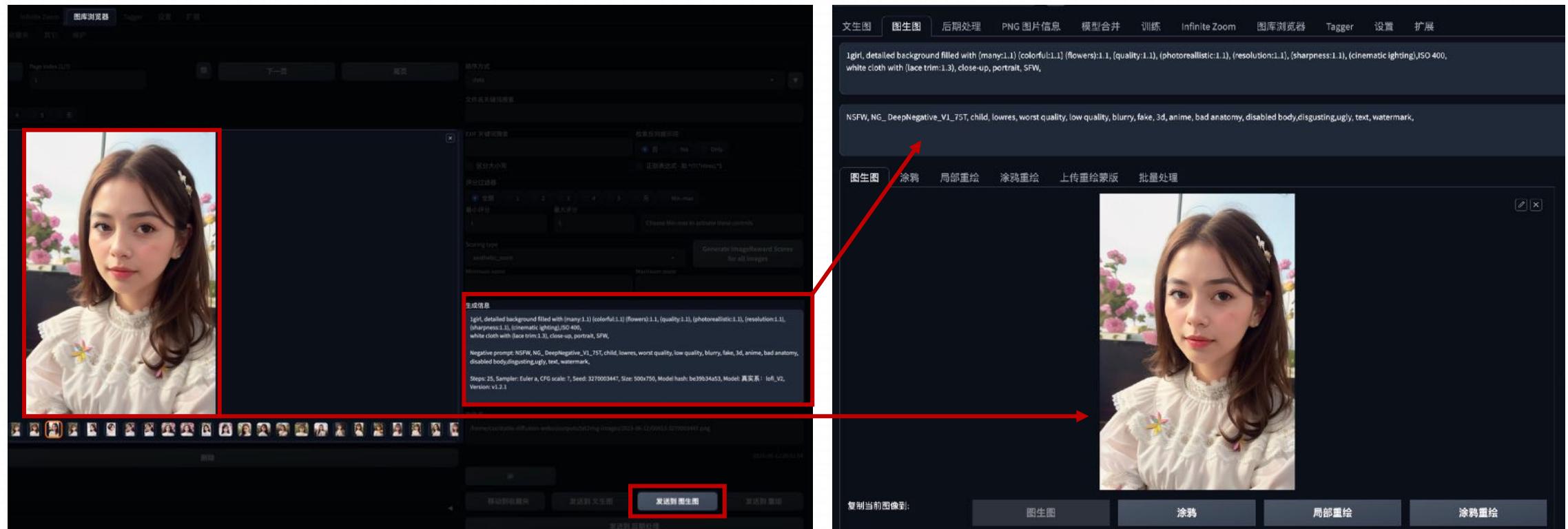


在图生图中其实是没有高清修复这个选项的，但经过文生图中高清放大的原理解释，可以意识到**图生图本身就是一种“高清修复”。**

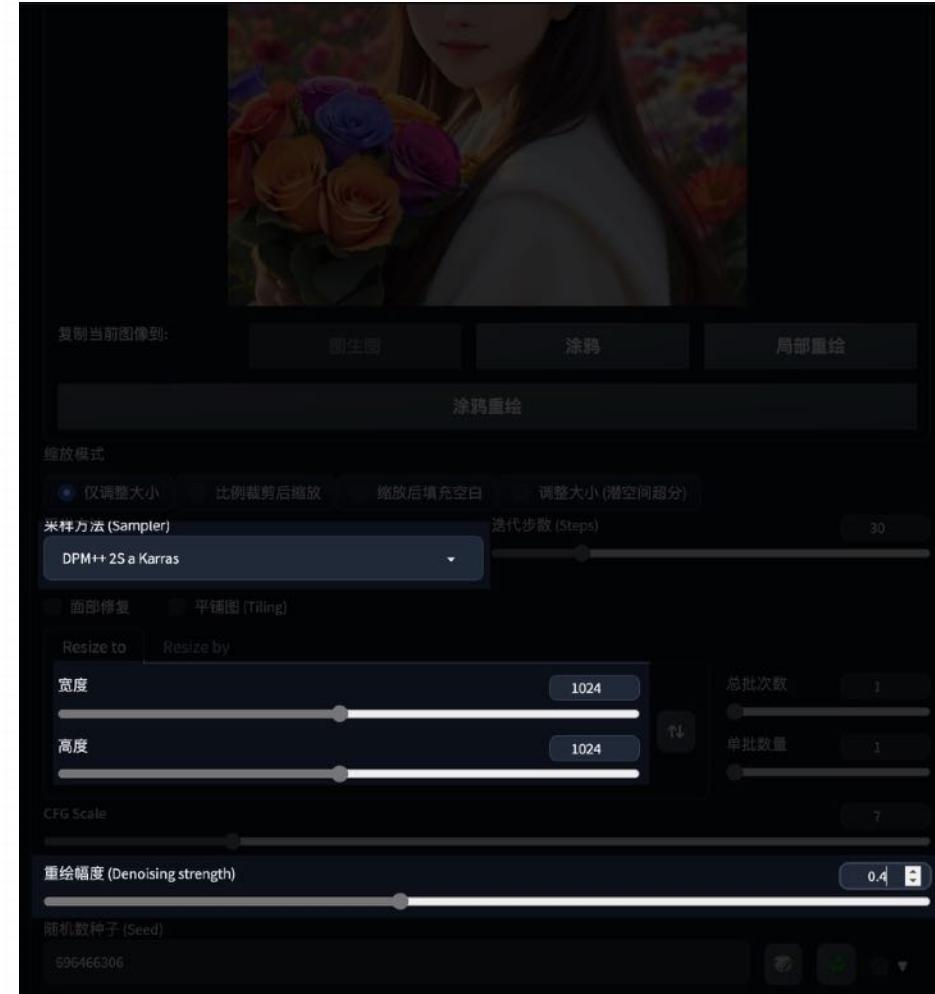
如果原图像像素低，那么放到图生图中，只要按更高的分辨率设置，就可以实现高分辨率修复了，但需要注意将**“重绘幅度”的数值调到0.3以下**。

2.1 图生图放大方式简析

如果是用SD生成的图像，我们可以通过点击图像浏览器，然后找到对应的图像，点击右下方的发送到图生图，就可以把这张图的所有信息同步到“图生图”选项卡中，以方便我们的下一步操作。

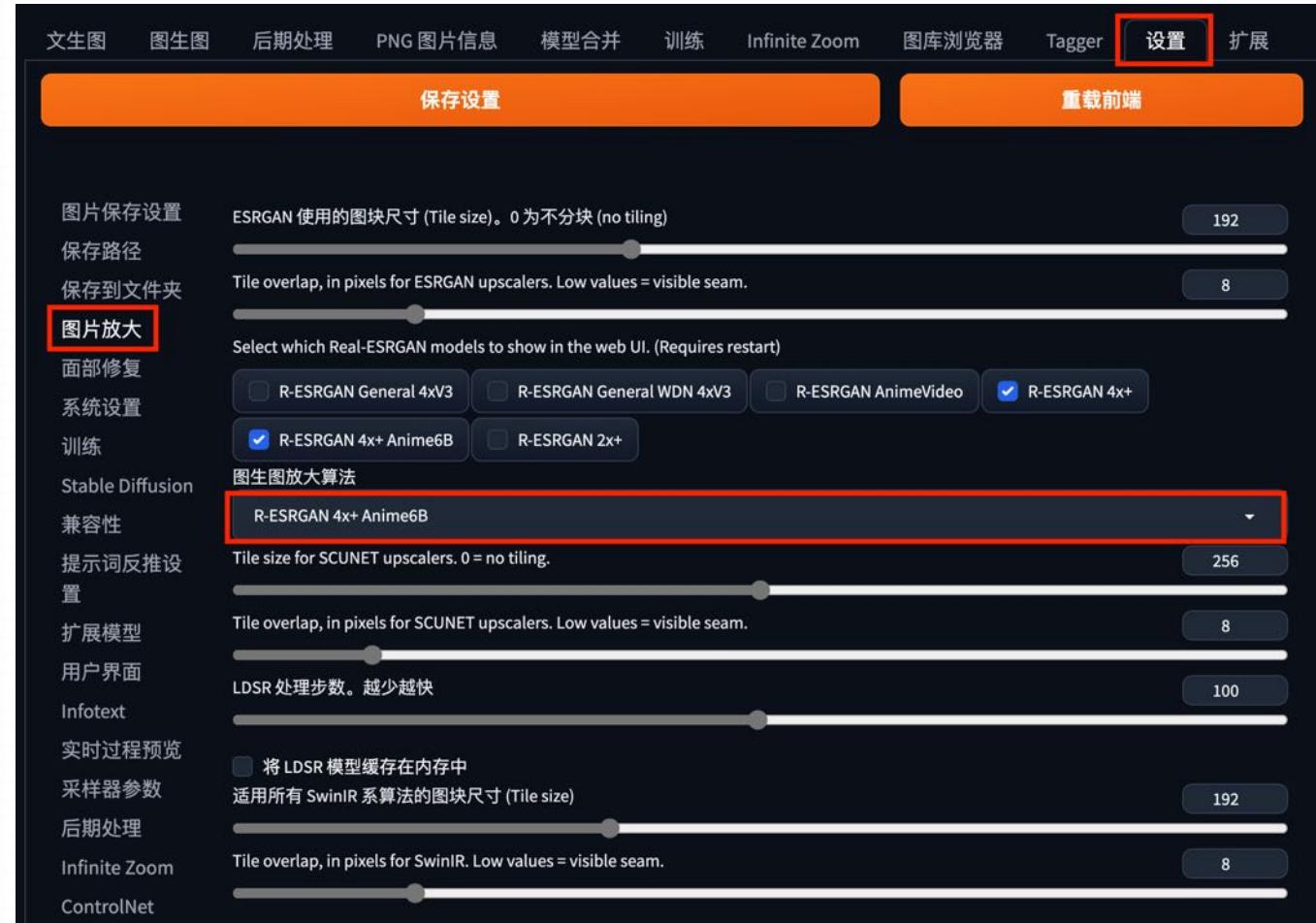


2.1 图生图放大方式简析



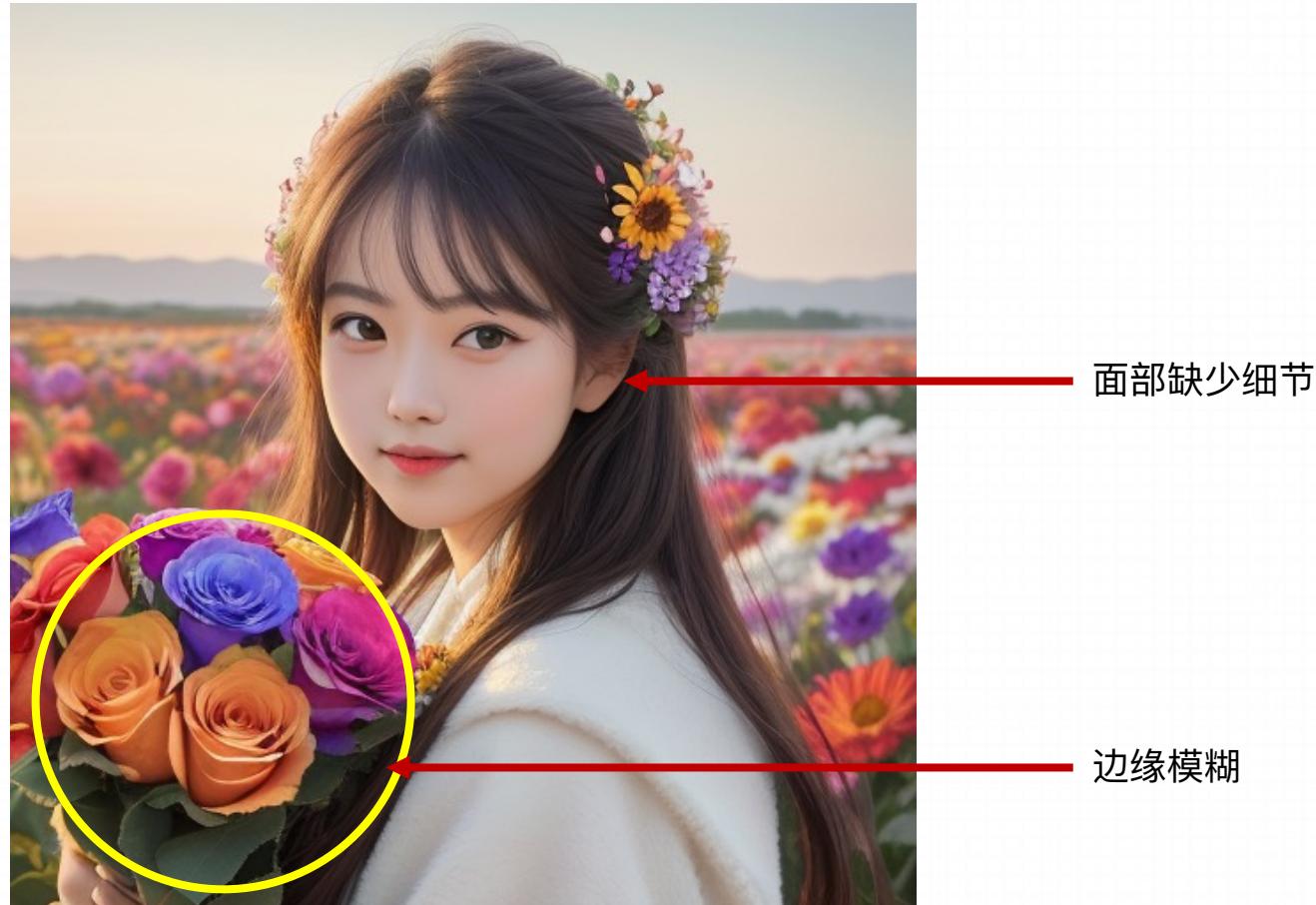
2.1 图生图放大方式简析

在设置的放大相关选项中，可以自行定义用于图生图的放大过程的算法。



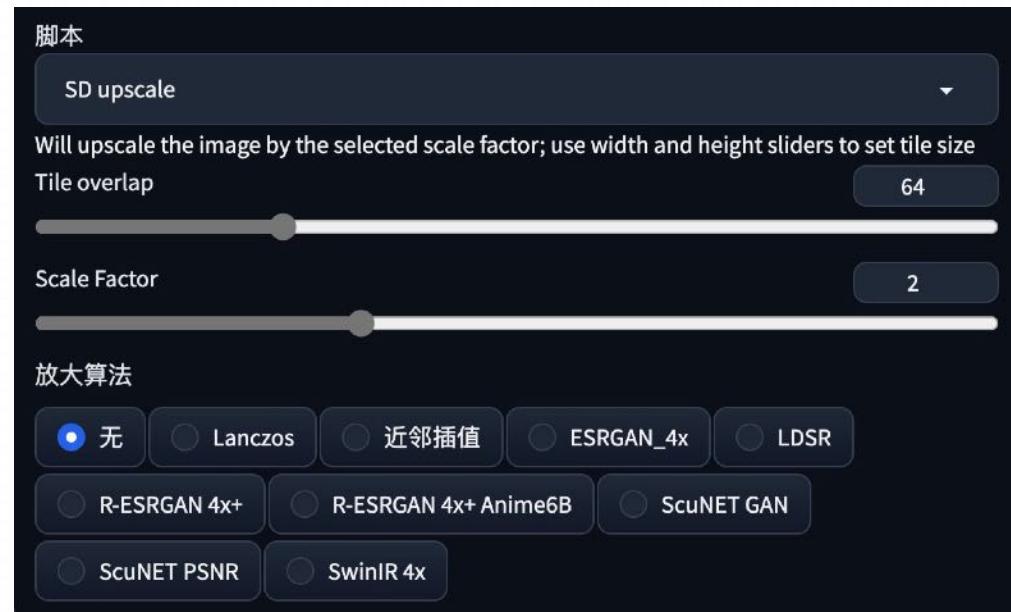
2.2 SD放大原理及简单操作

细节缺少和混乱



2.2 SD放大原理及简单操作

加载脚本：致力于为你实现一些软件本身并不包含但却非常实用的附加功能



- **Scale Factor:** 缩放系数相当于放大倍数，设置为2，意为将图像放大为原图的2倍。
- 图块重叠的像素 (Tile overlap)：默认设置为64。然后再原图的分辨率上需要分别添加64个像素（例：原图的分辨率从600*600变为664*664）。
- 点击生成，图像会被均匀的切分成四块依次放大

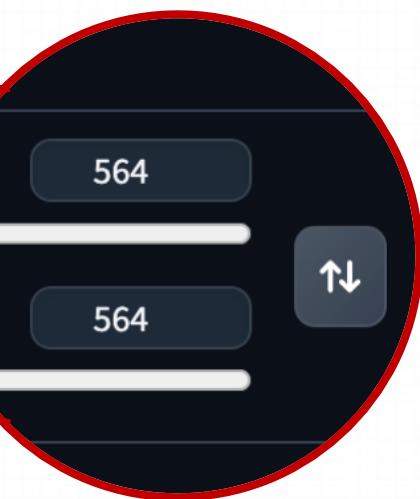
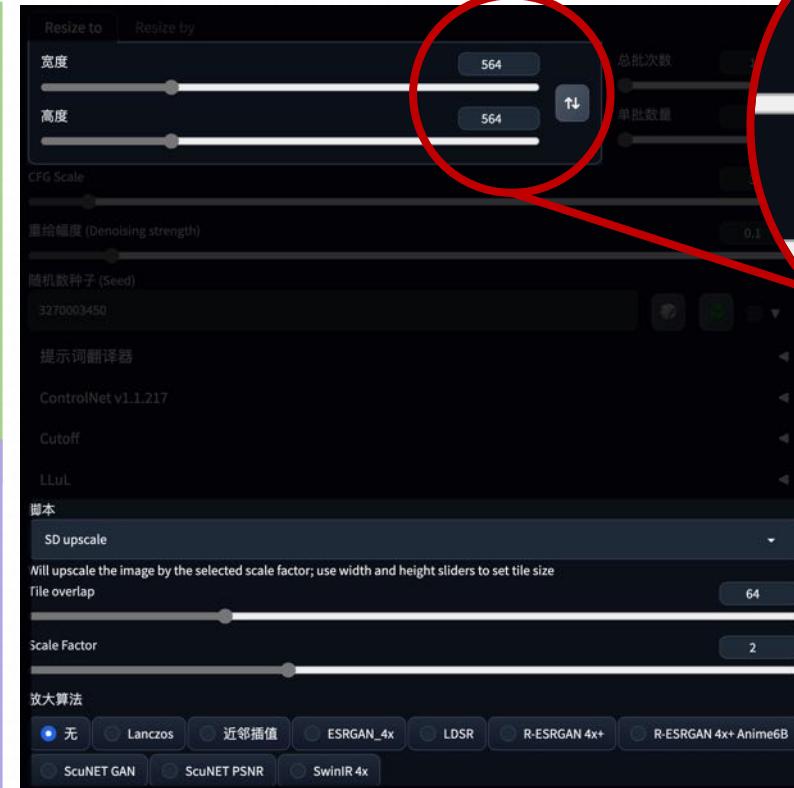
2.2 SD放大原理及简单操作



2.2 SD放大原理及简单操作



64*64 是“缓冲带”作用，如果还觉得生硬可以把缓冲带的宽度扩大为 128*128



原图分辨率是500*500，设置了64的缓冲区域，那么我们就需要将分辨率改为564*564

注：开启 SD 放大后，最终宽高 = (设置的宽高-重叠像素) × 放大倍率。因此，设置重叠像素后，要在设置宽高的基础上加上该像素值。

对应的代价就是单张图片的大小会变大

2.2 SD放大原理及简单操作



原图



放大倍数: 3
重绘幅度: 0.3
Seed: 696466306

A screenshot of a generative AI application interface. It shows the zoomed-in image on the right and a smaller thumbnail of the original image on the left. Below the image are several buttons: a profile icon, a save icon, a download icon, a "Send to Raw Image" button, a "Send to Retouch" button, and a "Send to Post-processing" button. At the bottom, there is a text input field containing the generation parameters: "1girl, detailed background filled with (many:1.1) (colorful:1.1) (flowers):1.1, (quality:1.1), (photorealistic:1.1), (resolution:1.1), (sharpness:1.1), (cinematic lighting); ISO 400," followed by a progress bar indicating "2% ETA: 01:04".

生成过程

2.2 SD放大原理及简单操作

优势：

- 可以突破内存限制获得更大的分辨率（最高可达4倍宽高）；
- 画面精细度高，对细节的丰富效果出色；

劣势：

- 分割重绘的过程较为不可控（语义误导和分界线割裂）；
- 操作繁琐且相对不直观；
- 偶尔“加戏”，出现莫名的额外元素。

补充内容

用类似的方式，可以把图片变的非常大，且没有过多的算力压力，即使是放大四倍，也就是 $4 \times 4 = 16$ 张图的事情。但因为放的比较大，原图会被裁切成很多小块，这个时候 AI 很难将提示自己关于每个区块的内容对号入座，就容易让画面变得混乱。此外，如果有人脸，身体的关键部位恰好处在分界线上的时候，也有很大概率会产生不和谐画面。

解决方法就是降低重绘幅度，保持一致性，并且增大缓冲区尺寸。

3. AI upscaler

3.1 AI upscaler

3.2 AI upscaler 放大操作

3.1 AI upscaler

传统算法放大图像的问题：

传统的调整图像大小的算法，例如nearest neighbor插值和Lanczos插值，在放大图像时效果并不好。这些算法通过执行一些数学运算来放大画布，并使用原始图像的像素值来填充新像素。如果图像本身就是损坏或失真的，这些算法就无法准确地填充所缺失的信息。

3.1 AI upscaler

AI的upscaler是基于海量数据训练的模型。训练原理可以这样理解：首先模仿现实世界中的图像失真，将高质量图像有意破坏，再将这些破坏后的图像缩小到比较小的尺寸，最后通过训练神经网络，将这些图像还原成最初的样子。

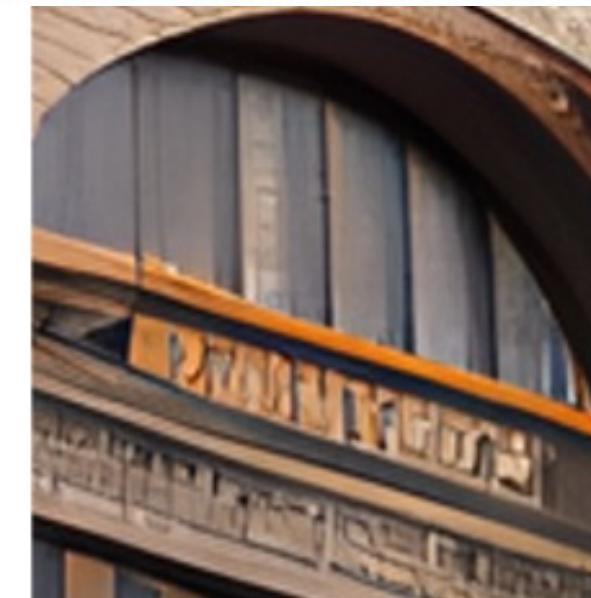
下面是传统 (Lanczos)和AI (R-ESRGAN) upscaler的比较示例。由于AI upscaler中所嵌入的先验知识，它可以放大图像并同时恢复细节。



Original



Lanczos



R-ESRGAN

3.2 AI upscaler 放大操作

Upscaler可以把一张已经绘制为成品的图片，通过人工智能算法放大到一定的尺寸，这个功能一般用在图片生成后的处理上。换句话说它就像是一个重绘幅度为零的高清修复。它的原理和市面上绝大多数AI修复照片的应用原理是相似的，而因为不涉及再扩散的过程，它的运行速度很快，几秒钟可能就能做好一张图。

打开方式一：在后期处理选项卡中拖入一张图片进行处理

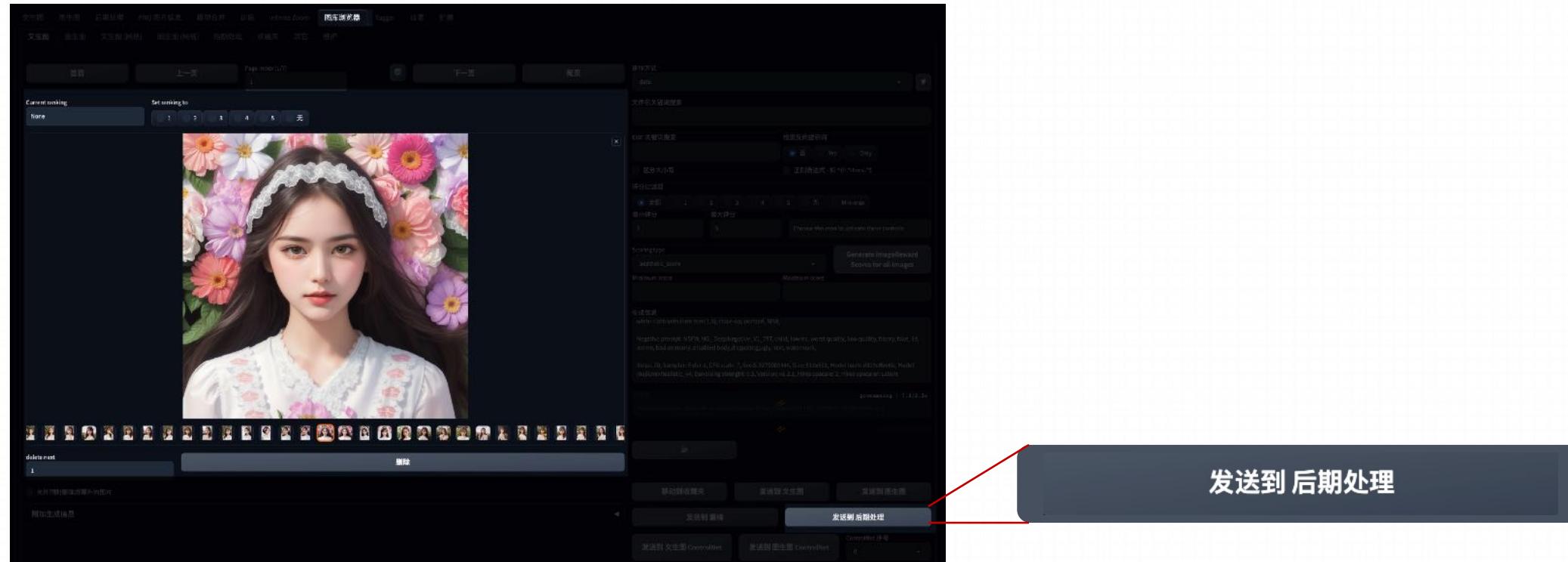


<https://stable-diffusion-art.com/ai-upscaler/>

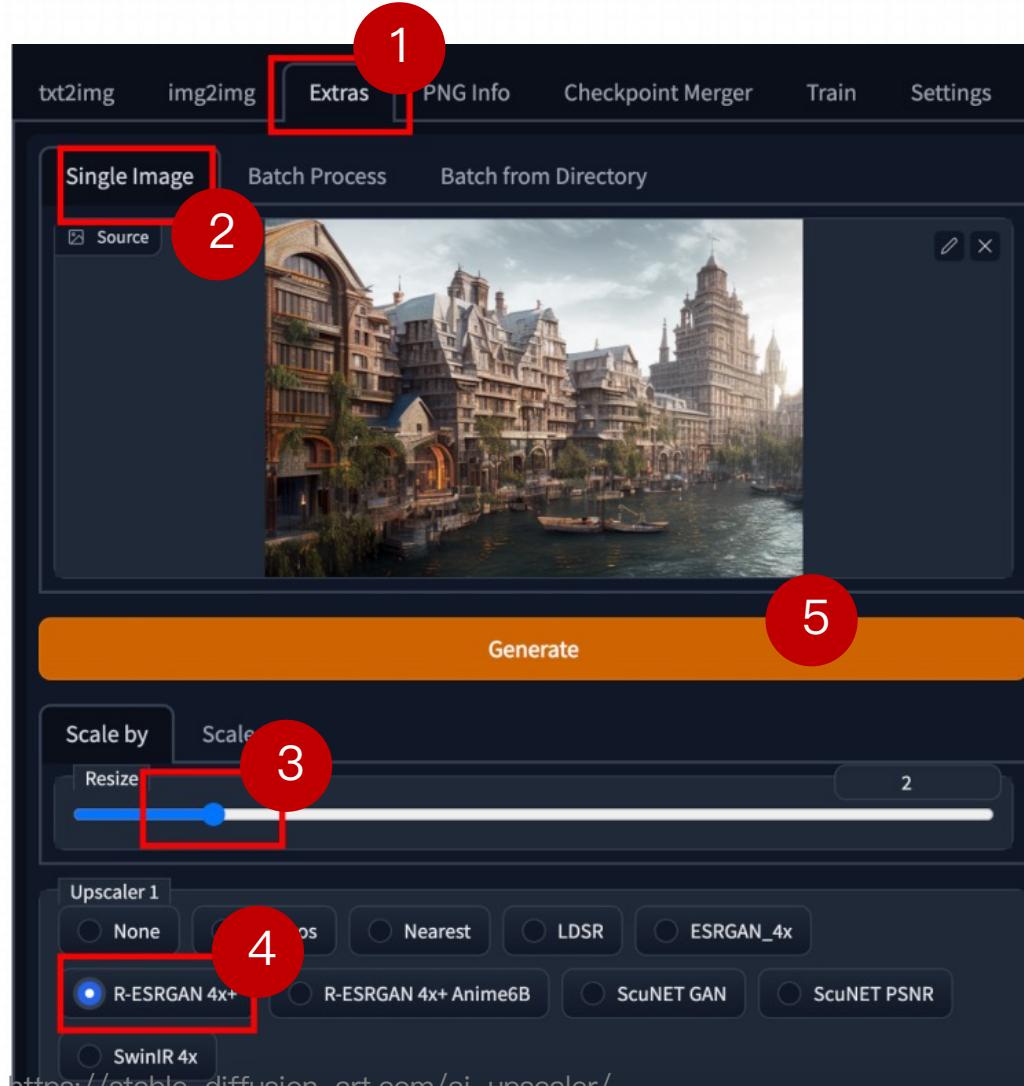
3.2 AI upscaler 放大操作

Upscaler可以把一张已经绘制为成品的图片，通过人工智能算法放大到一定的尺寸，这个功能一般用在图片生成后的处理上。换句话说它就像是一个**重绘幅度为零的高清修复**。它的原理和市面上绝大多数AI修复照片的应用原理是相似的，而因为不涉及再扩散的过程，它的运行速度很快，几秒钟可能就能做好一张图。

打开方式二：在图库浏览器中找到这张图，然后在右下方点击“发送到后期处理”。



3.2 AI upscaler 放大操作



<https://stable-diffusion-art.com/ai-upscaler/>

使用方式：

1. 转到“Extras”选项卡，然后选择“Single Image”
2. 将想要放大的图像上传到源(source)画布。
3. 设置缩放比例。许多AI upscaler 将参数值默认为4
(也就是将升级4次，将图片分辨率升级为4倍大小)，
我们也可以就用默认的值。但如果不想图像太大，
将其设置为较低的值，例如2。
4. 选择放大算法，和之前一样，人像选择R-ESRGAN
4x+，二次元选择R-ESRGAN 4x+ Anime6B
5. 点击Generate开始升级。
6. 完成后，放大的图像将出现在右侧的输出窗口中。
右键单击可以保存图像。

3.2 AI upscaler 放大操作

放大算法选择

LDSR : Latent Diffusion Super Resolution (LDSR) –潜在扩散超分辨率

此 upscaler 最初与 StableDiffusionV1.4 一起发布，它是一个经过训练的潜在扩散模型 (latent diffusion model)，用于提高图像分辨率。尽管生成的质量非常好，但生成速度非常慢，因此不建议使用。

ESRGAN 4x : Enhanced Super-Resolution Generative Adversarial Networks (ESRGAN)-

增强型超分辨率生成对抗网络

此upcaler 模型是对以前的SRGAN模型的增强，它曾赢得了2018年感知图像恢复和操纵挑战赛(2018 Perceptual Image Restoration and Manipulation challenge)。它倾向于保留精细的细节并产生清晰锐利的图像。

R-ESRGAN 4x : Real-ESRGAN (R-ESRGAN)

该模型是ESRGAN的增强版,可以还原各种现实世界的图像。比如说修复源于相机镜头或者数字压缩等的各种程度的失真。与ESRGAN相比，它倾向于产生更平滑的图像。此外， R-ESRGAN在逼真写实的照片图像上表现是最佳的。

3.2 AI upscaler 放大操作

放大图像的示例

下面是使用 R-ESRGAN 放大复杂场景的示例



<https://stable-diffusion-art.com/ai-upscaler/>

3.2 AI upscaler 放大操作

后期处理中的放大方式简单方便，负担小且随时可以调用。所以无论是图生图还是文生图，当得到了一个相对比较清晰的版本了以后，都可以再过一次AI放大，来实现进一步的精细化。但在效果上，由于它是在拉伸放大的基础上，适当将线条进行润滑，且将色块边缘进行模糊处理，所以在整体的精致细腻程度上并不如前两种的放大方式。

优势：

- 使用方便，操作简单，随时可以调用；
- 计算速度快，无重绘压力；
- 完全不改变图片内容；

劣势：

- 效果不太显著！

4. Tiled 放大

4.1 Tiled Diffusion 原理与基本操作

4. Tiled Diffusion 原理与基本操作

高清修复的本质上一个额外的图生图，它无法突破显存上限且位置速度慢。

SD放大以及进阶的ultimate Upscale，有一些分区块处理的雏形，但可控性较差，且绘制速度偏慢。

Tiled Diffusion直接表现出来的优势就是效果好、效率高、可控性强，并且以极其有效的方式优化了显存。

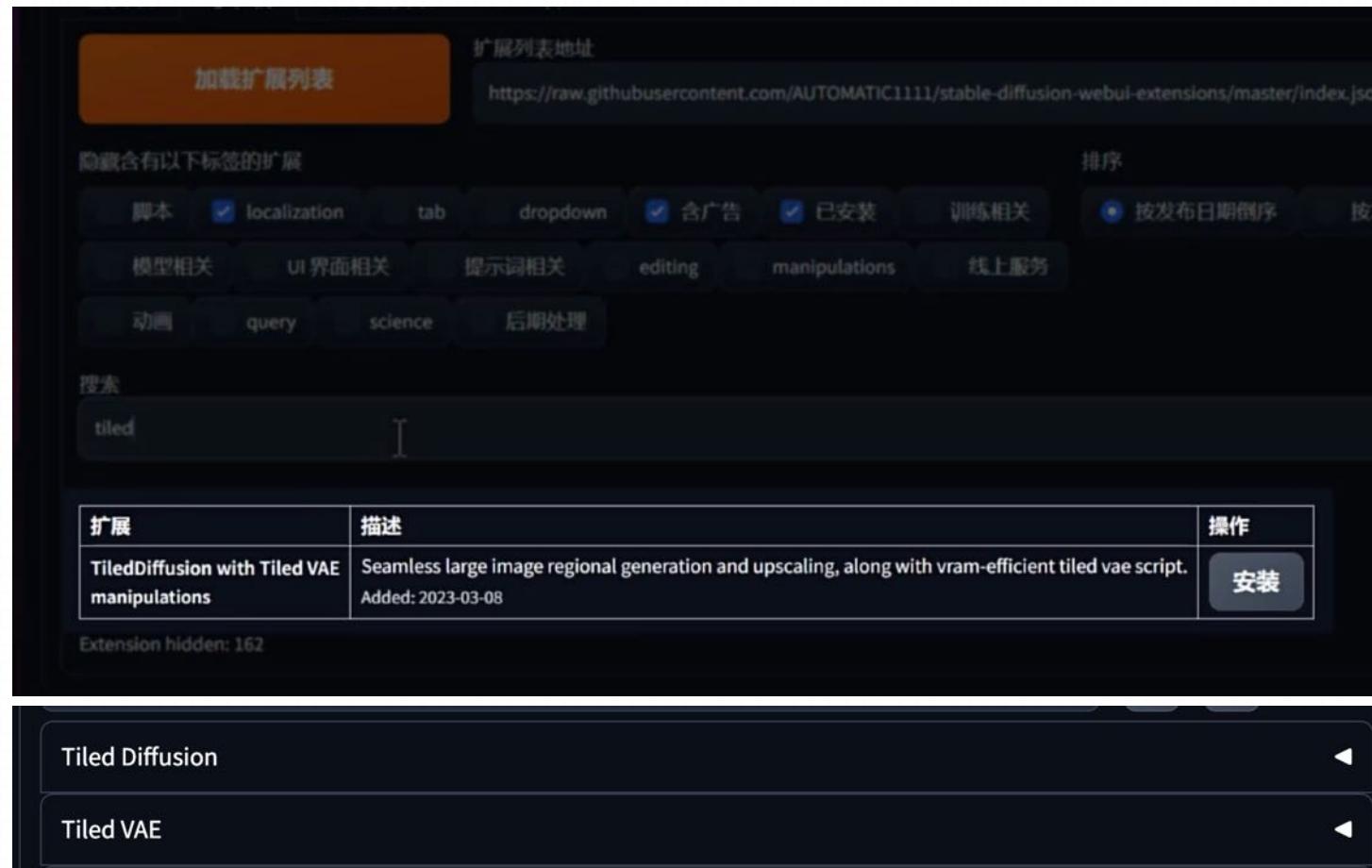
4. Tiled Diffusion 原理与基本操作

这个扩展的核心就是将图像“Tile”—— 区块化处理，这样处理的核心作用是提高分辨率以及降低绘制显存。

直接画一张很大的图片对电脑来说肯定会有压力。而 Tiled Diffusion 这个扩展会按照一定的规则将它分割成一个个画起来更加轻松的小块，然后分别的去绘制，再使用巧妙的方式无缝拼接叠在一起，并借助一点点算法来优化整体效果。这样就可以极大的降低 AI 绘制过程中的负担。

4. Tiled Diffusion 原理与基本操作

Tiled Diffusion 安装方式：



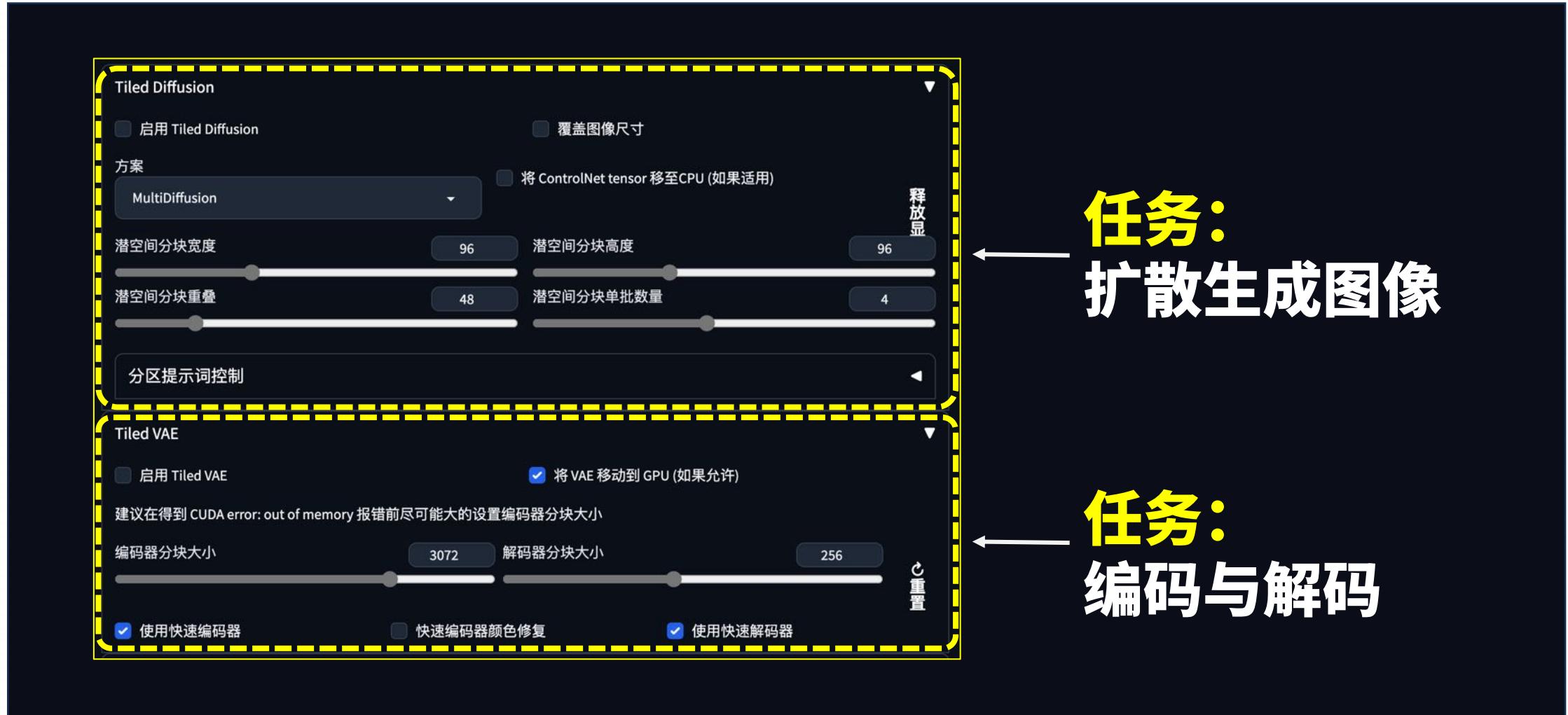
安装方式一：

1. 在插件中搜索“TiledDiffusion”直接进行安装。
2. 重启Webui，在[文生图]和[图生图]页面会出现Tiled Diffusion 和 “Tiled VAE”两个选项卡，证明安装成了。

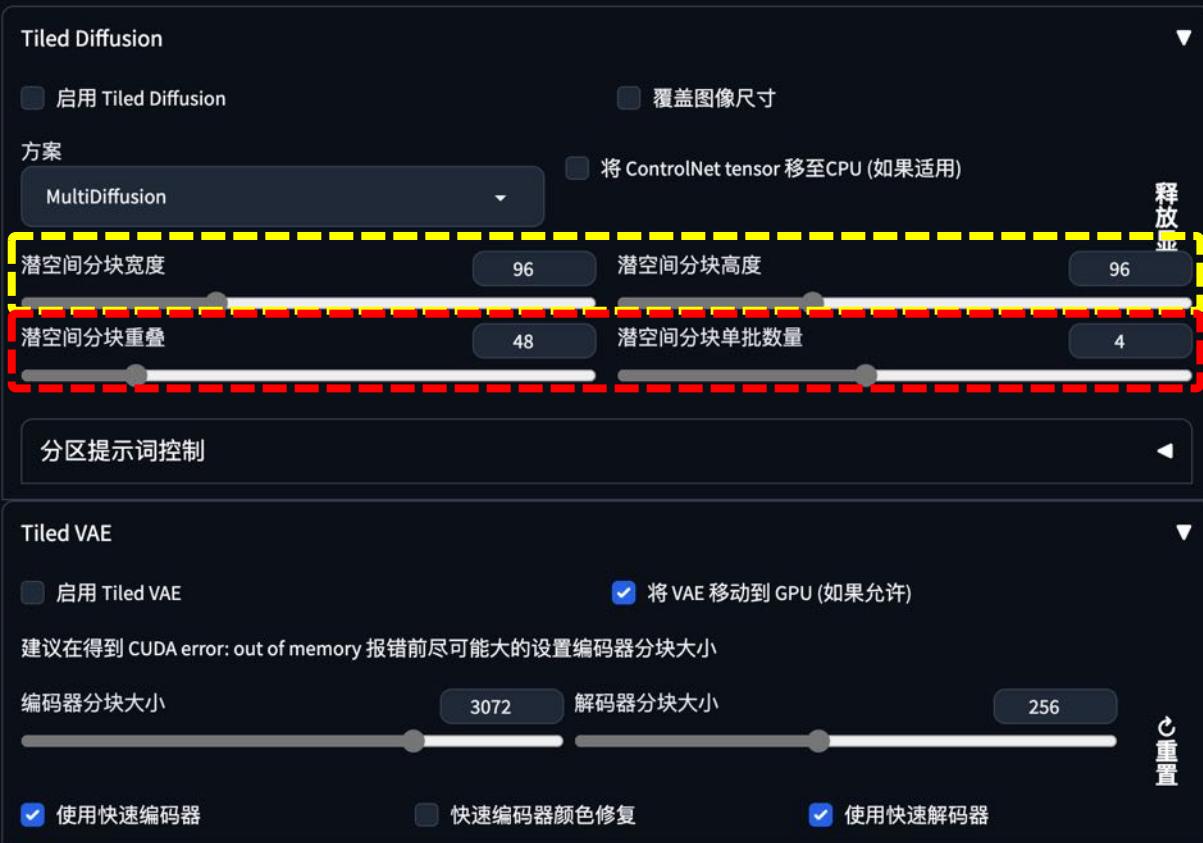
安装方式二：

1. 在网上下载压缩包，解压后放到 Stable_Diffusion_webui/Extensions文件夹中。
2. 重启Webui

4. Tiled Diffusion 原理与基本操作



4. Tiled Diffusion 原理与基本操作



The screenshot shows the Tiled Diffusion settings interface. It includes two main sections: "Tiled Diffusion" and "Tiled VAE".

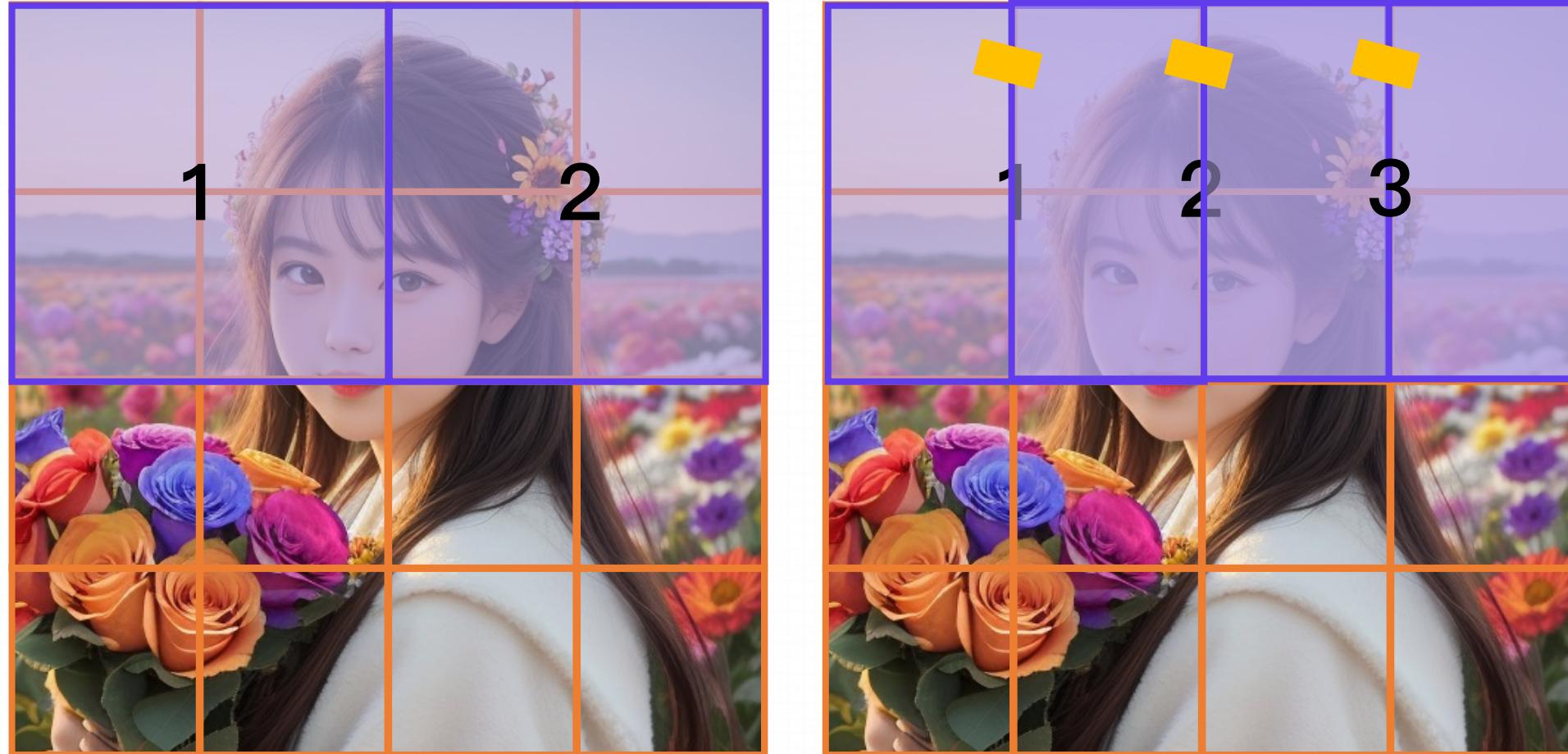
Tiled Diffusion: This section has a "方案" dropdown set to "MultiDiffusion". It features three sliders under a dashed red box: "潜空间分块宽度" (Width) at 96, "潜空间分块高度" (Height) at 96, and "潜空间分块重叠" (Overlap) at 48. A "分区提示词控制" (Partition Prompt Control) button is below these sliders.

Tiled VAE: This section has a "编码器分块大小" (Encoder Tile Size) slider at 3072 and a "解码器分块大小" (Decoder Tile Size) slider at 256. It includes checkboxes for "使用快速编码器" (Use Fast Encoder), "快速编码器颜色修复" (Fast Encoder Color Repair), and "使用快速解码器" (Use Fast Decoder). A "将 VAE 移动到 GPU (如果允许)" (Move VAE to GPU if possible) checkbox is checked.

Text on the right:

- 区块大小主要决定了把图片拆分成多少块，并间接影响处理、绘制速度。而重叠大小则决定这些块拼在一起时融不融洽。
- Tiled VAE 中的内容一般不需要进行设置，维持默认即可生成不错的图像。
- 两种情况例外：
 - 当生成过程中看到显存不足 (CUDA Out of Memory) 的错误时，可以适当降低编码快区块大小，避免爆显存
 - 当使用的tile太小且图片变得灰暗和不清晰时，可以启用编码器颜色修复，让颜色更正常。

4. Tiled Diffusion 原理与基本操作



区块大小主要决定了把图片拆分成多少块，并间接影响处理、绘制速度。而重叠大小则决定这些块拼在一起时融不融洽。越大的重叠在“消除”接缝感时起到的效果就越好，但重叠大肯定意味着绘制的块数会增加进而增大总工作量。

4. Tiled Diffusion 原理与基本操作

Tiled Diffusion & VAE 设置方案

区块大小设置

维持宽高相等，把它设置成边长在**64–160**之间的正方形即可；

- 大部分模型在正方形的比例下绘制效果最为出色；
- 作者也建议设置为96或128可以获得更快的速度；

重叠区域设置

使用 MultiDiffusion 时选择**32或48**、使用 Mixture of Diffusers 时选择**16或32**就可以很好地支持出图；

- Mixture of Diffusers引入了高斯平滑的处理，重叠可以设置得更低；
- 重叠大小不绝对，更多地要根据区块大小灵活调整。

4. Tiled Diffusion 原理与基本操作



产出图像设置方案1:

- 图生图里设置最终宽高（自己算）
- 取消“保持输入图片大小”
- 下方“放大倍数”设为1

产出图像设置方案 2:

- 图生图里设置宽高
- 勾选“保持输入图片大小”
- 下方“放大倍数”设为想放大的倍数

4. Tiled Diffusion 原理与基本操作

prompt: (masterpiece, best quality), 1girl, solo, flower, long hair, outdoors, letterboxed, school uniform, day, sky, looking up, short sleeves, parted lips, shirt, cloud, black hair, sunlight, white shirt, serafuku, upper body, from side, pink flower, blurry, brown hair, blue sky, depth of field

Negative prompt: EasyNegativeV2



ENSD: 31337,
Size: 1024x512,
Seed: 2311942344,
Model: CF5_Counterfeit-V3.0_fix_fix_fix,
Steps: 25,
Sampler: DPM++ 2M Karras,
CFG scale: 10,
Clip skip: 2,
Model hash: db6cd0a62d,
Hires upscale: 2,
Hires upscaler: R-ESRGAN 4x+ Anime6B,
Denoising strength: 0.45

4. Tiled Diffusion 原理与基本操作

prompt: (masterpiece, best quality), 1girl, solo, flower, long hair, outdoors, letterboxed, school uniform, day, sky, looking up, short sleeves, parted lips, shirt, cloud, black hair, sunlight, white shirt, serafuku, upper body, from side, pink flower, blurry, brown hair, blue sky, depth of field

Negative prompt: EasyNegativeV2



Size: 3632 * 1816

5. 局部重绘

5.1 局部重绘基本操作

5.2 绘制 (inPaint Sketch) 功能应用

5.3 上传蒙版 (inPaint Upload) 功能应用

三种主要的局部重绘方案

如何在 SD 中让图片变得更清晰&更有细节?

01

局部重绘基本操作

了解在StableDiffusion的图生图功能中，对一张图片进行局部重绘的基本操作方式，并结合实践梳理参数设置要点。

02

绘制(inPaint Sketch)功能应用

在局部重绘的基础上，了解与绘制(Sketch)功能结合运用的方式，并尝试使用inPaint Sketch进行手部修复。

03

上传蒙版 (inPaint Upload) 功能应用

在局部重绘的基础上，了解经由上传蒙版图片Upload Mask) 定义重绘区域的方式，并尝试利用Photoshop生成更为精确的蒙版图片。

5.1 局部重绘

A. 局部重绘的概念及基本操作流程



需求:

在这个图的基础上进行改进，让女孩闭眼。

解决方式:

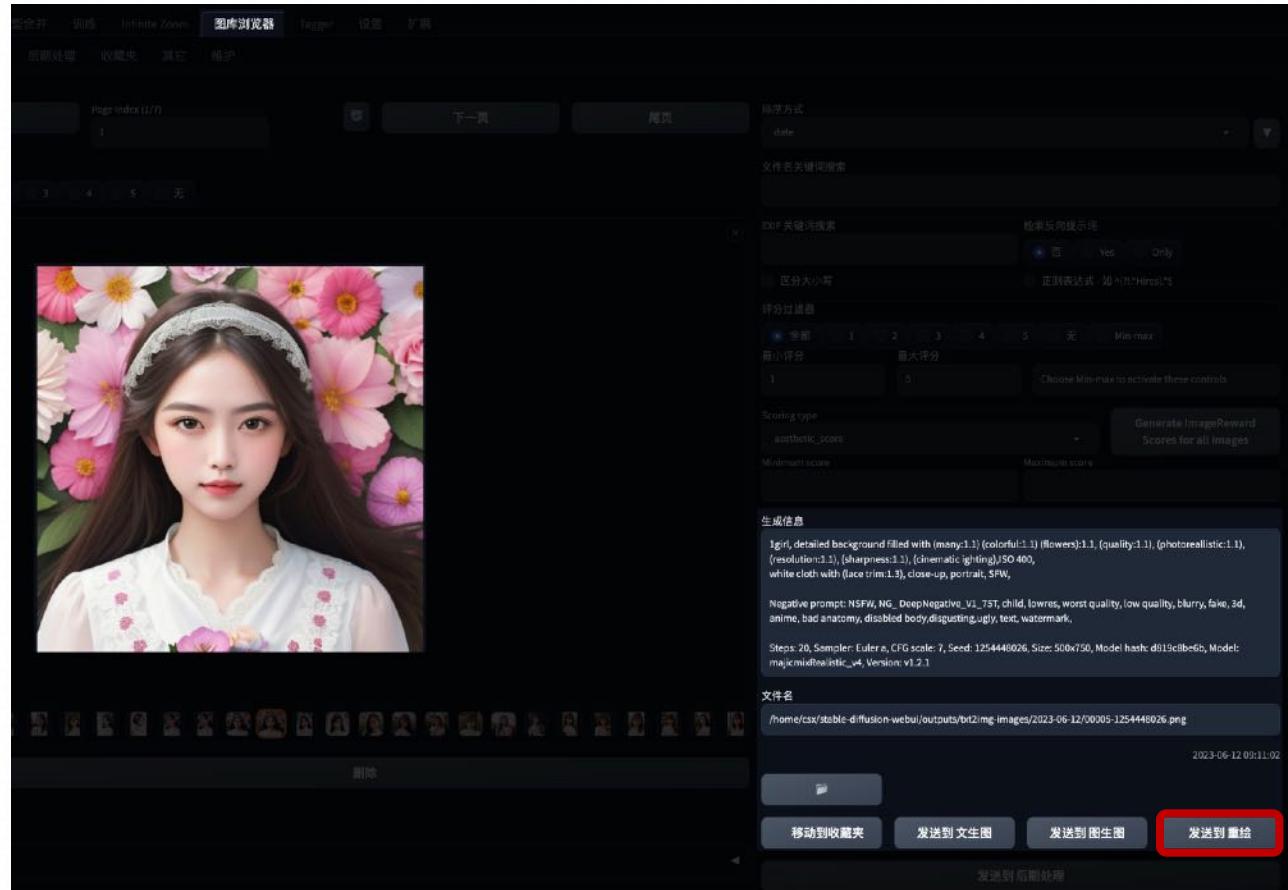
加入closed eyes 提示词，并借助随机种子固定画面。

这种方式的问题

1. 即便随机种子完全一致，当提示词发生变化时，这个新随机过程是不可控的。可能出现人物换了姿势、画面大幅度变化的情况。
2. 如果你的图片已经经过了高清修复或者upscale放大，这意味着你要再重新画一张非常大的图，耗费的时间会更长

5.1 局部重绘

A. 局部重绘的概念及基本操作流程

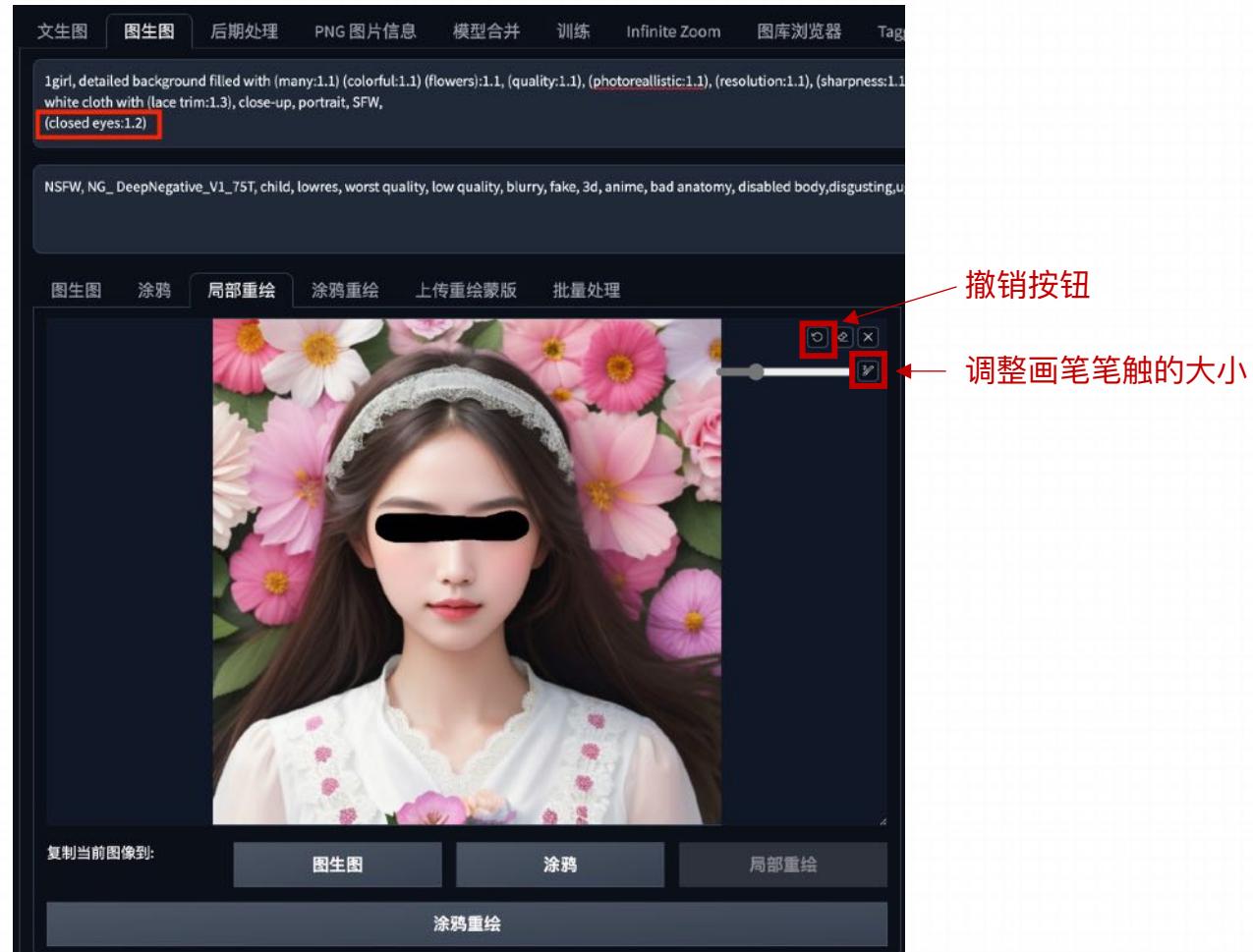


操作步骤：

1. 在图库浏览器里直接点击右下角的局部重绘按钮，转到图生图（局部重绘是图生图的下属分支功能）。
2. 修改笔触大小，涂抹需要修改的区域
3. 在Prompt中输入涂抹区域想要变成的信息，并增加一些权重。
4. 参数设置：默认设置
5. 点击生成

5.1 局部重绘

A. 局部重绘的概念及基本操作流程



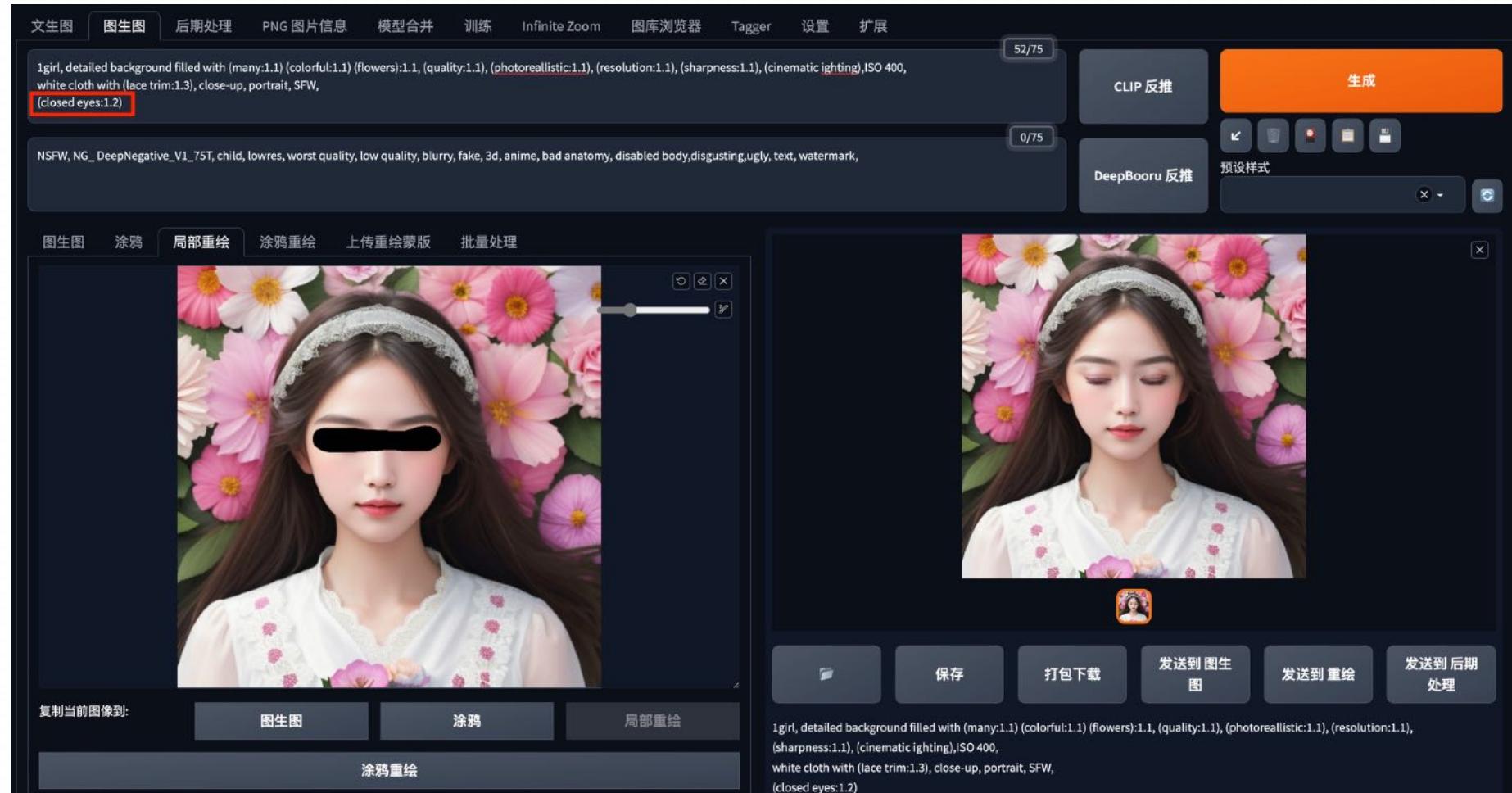
具体操作步骤：

切换到局部重绘后，其他的设置区域都是基本一致的。

1. Prompt保持不变，后面加上closed eyes: 1.2权重
2. 图生图重绘幅度开到一个比较高的数值0.75
3. 画笔笔尖，涂出黑色区域，覆盖AI重画区域

5.1 局部重绘

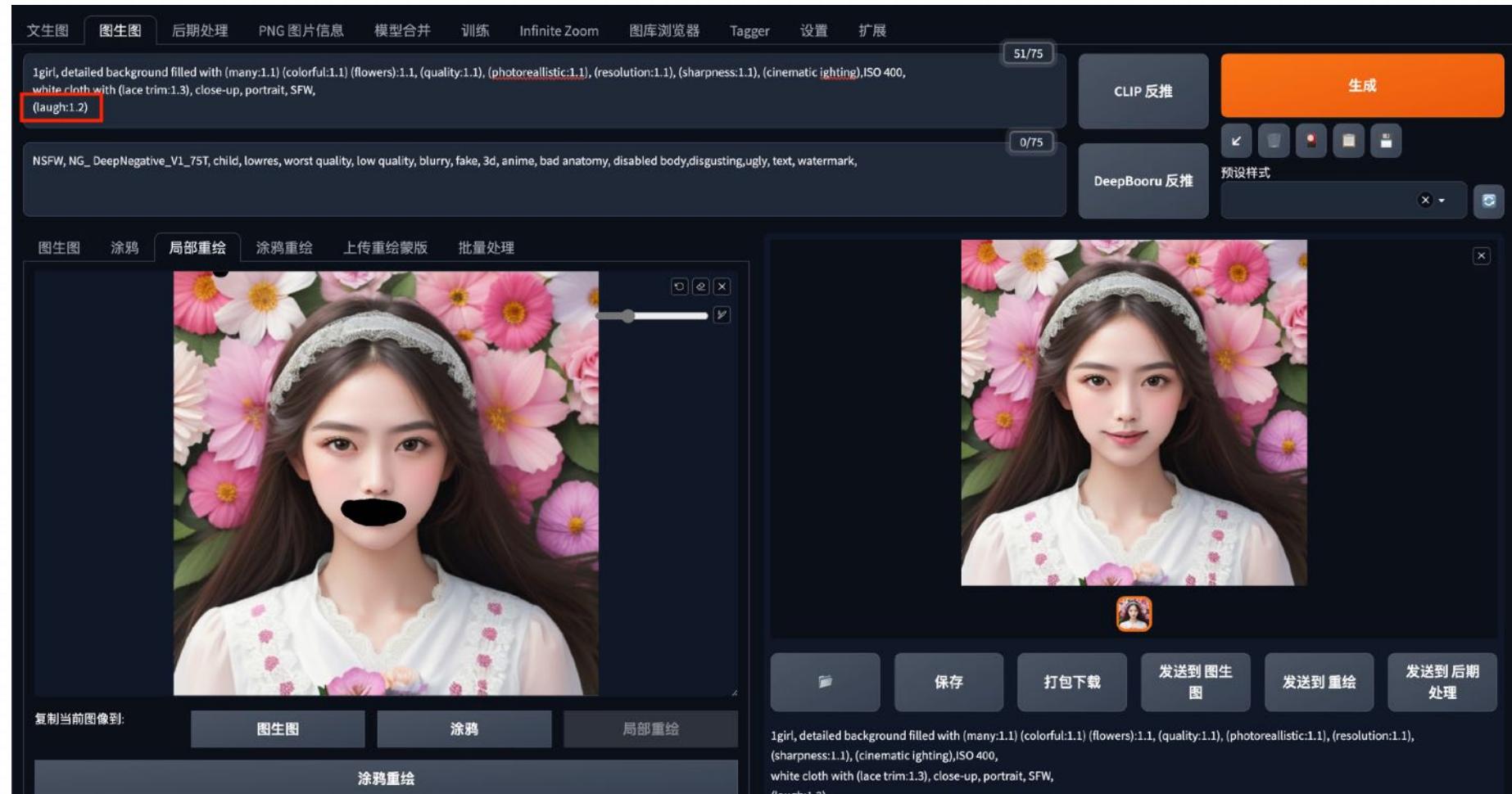
A. 局部重绘的概念及基本操作流程



重新加噪并去噪的过程，眼睛部分被强调了，其他部分保留了相对一致的外观，单独图生图，又拼回了原图。

5.1 局部重绘

A. 局部重绘的概念及基本操作流程



5.1 局部重绘

B. 局部重绘参数含义梳理

蒙版是图像处理领域的专业名词，它反一些用以限定处理区域的范围对象，简而言之，蒙版是一个“蒙”住了某些关键区域的“版”子。



蒙版边缘模糊

- 它决定了你重构区域边缘和其他部分是如何接触的。它类似于Photoshop中羽化效果。

蒙版模糊 = 0: 如果一点模糊都没有，四周一定会出现的边缘不是很和谐。

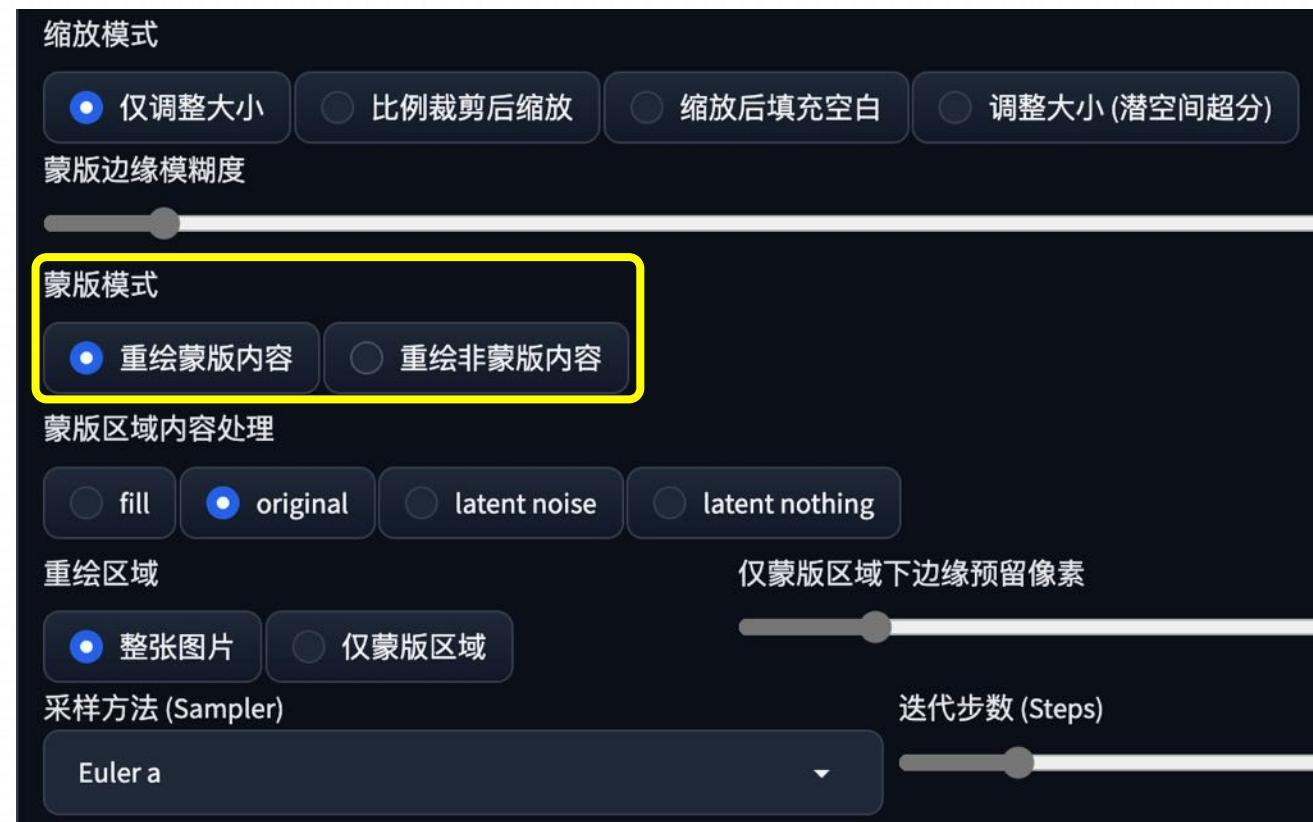
蒙版模糊 < 10: 10以下的模糊数值，可以让重复区域拼接进去的过程变得更加丝滑。

蒙版模糊 > 20: 如果数值太高，会影响内部区域的读取，也会影响到周边其他区域。

- 但整体控制在10以下是比較安全。

5.1 局部重绘

B. 局部重绘参数含义梳理



至于蒙版蒙住的内容，你可以想象是在把这一小块拿给AI图生图的时候，它接收到的信息。

蒙版模式

- 重绘蒙版内容：进修改蒙版内的画面
- 重绘非蒙版内容：修改没有被蒙版框选的画面

5.1 局部重绘

B. 局部重绘参数含义梳理

想要进一步把控修改效果，就需要调整下面的局部重绘参数。



蒙版区域内容处理

- Original: 保留原来的图像，不进行预处理
- fill: 用图像的颜色（高强度模糊）填充它，相当于是吧原图高度模糊以后再输入进去生成
- Latent noise/latent nothing: 把图生图的过程进一步复杂化了，加入了加噪、去噪的过程，理论上会对图像的改变更显著。

5.1 局部重绘

B. 局部重绘参数含义梳理



重绘区域

- 整张图片: AI会基于你的新的要求（提示词、参数），把整张图重新画一遍。但最后只保留你画出来的这一块区域拼回去。
- 仅蒙版区域: 仅重新生成蒙版区域，但因为AI无法识别画面全面，有时会出现很多拼接不合理的画面。

注意:

- 一些针对性强的修改，反而会希望缩小图片尺幅。但这个时候你需要降低重绘幅度，避免变形，并且对提示词做净化处理（权重1.2）

5.1 局部重绘

B. 局部重绘参数含义梳理



补充

如果用“仅蒙版”模式，旁边的边缘预留像素会发挥和放大修复。它和刚刚在图片放大部分中讲的“缓冲带”有类似的作用，默认数值是32，可以保证不消耗的平衡效果。如果你的重合区域大，可以适当增大它，区域小就反过来缩小一点。

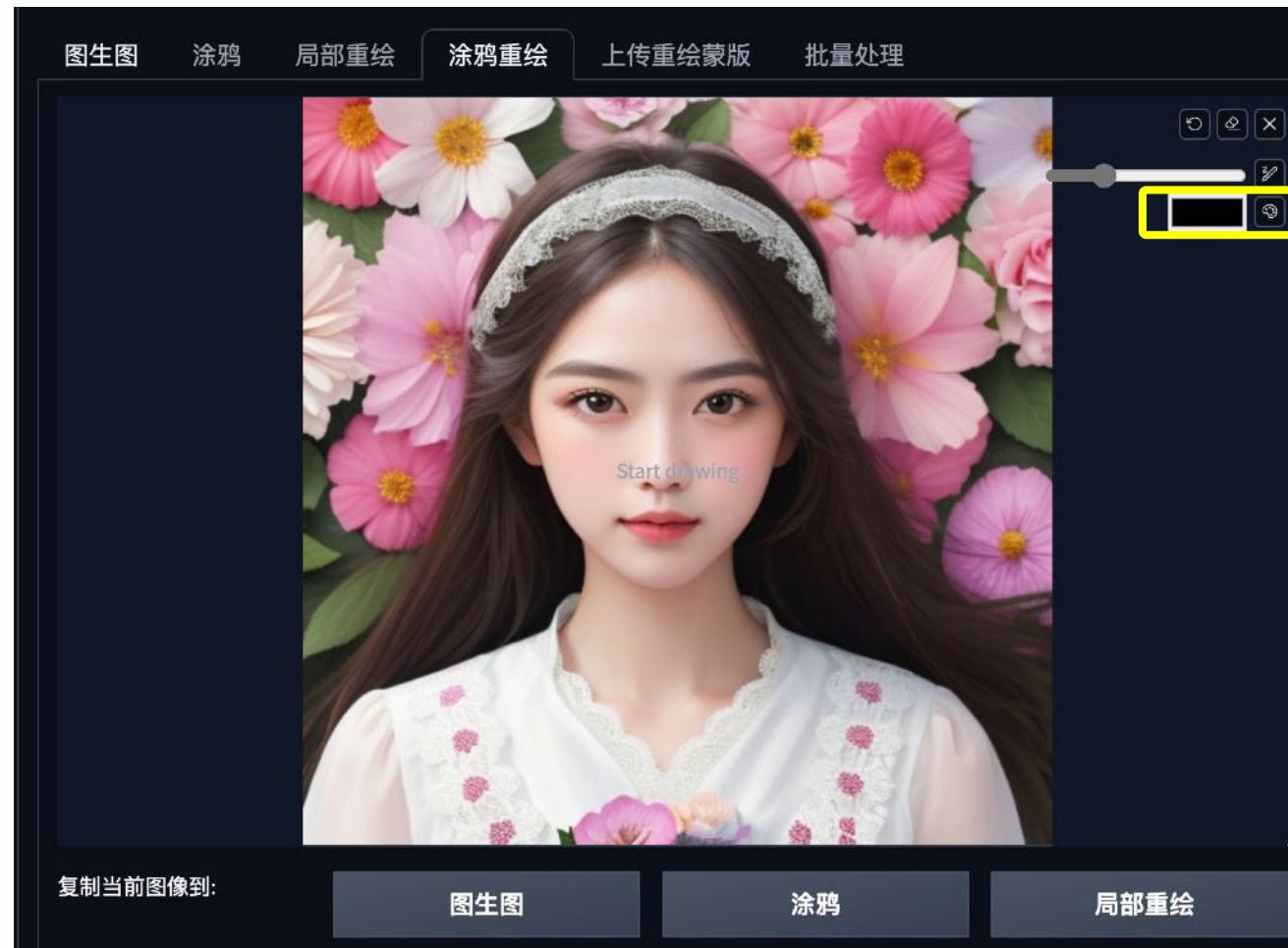
5.2 涂鸦重绘 (inPaint Sketch)

尽管局部重绘可以圈定一小块用于重新生成的区域，但它还是不够强大。如果要用它去修复一只画坏的手，即便区域被精准的限制住，但重新生成的过程仍然充满了不确定性。AI 可以将它画坏一次，也有可能会画坏两次，三次，无数次。WebUI里有一个进阶的功能，可以帮助我们增强局部重绘的可塑造性，从而轻松解决这个问题。



5.2 涂鸦重绘 (inPaint Sketch)

A. inPaint Sketch 基本操作流程



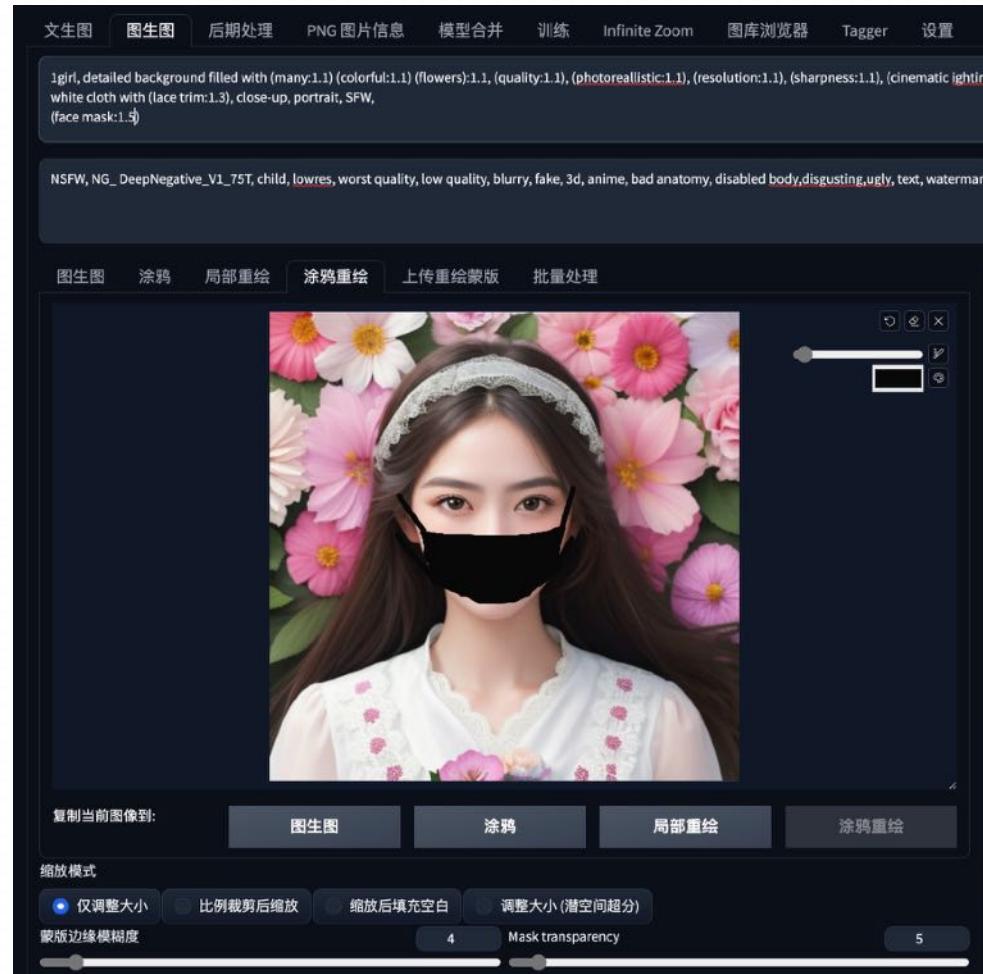
调色板

操作步骤：

1. 选择涂鸦重绘选项卡
2. 加载一张已经画好的图片
3. 画面中会多出来一个调色盘按钮，调色板可以自由更改画笔的颜色（画笔多了一部分输入内容的能力）。

5.2 涂鸦重绘 (inPaint Sketch)

A. inPaint Sketch 基本操作流程



需求：给画面中的人物添加口罩

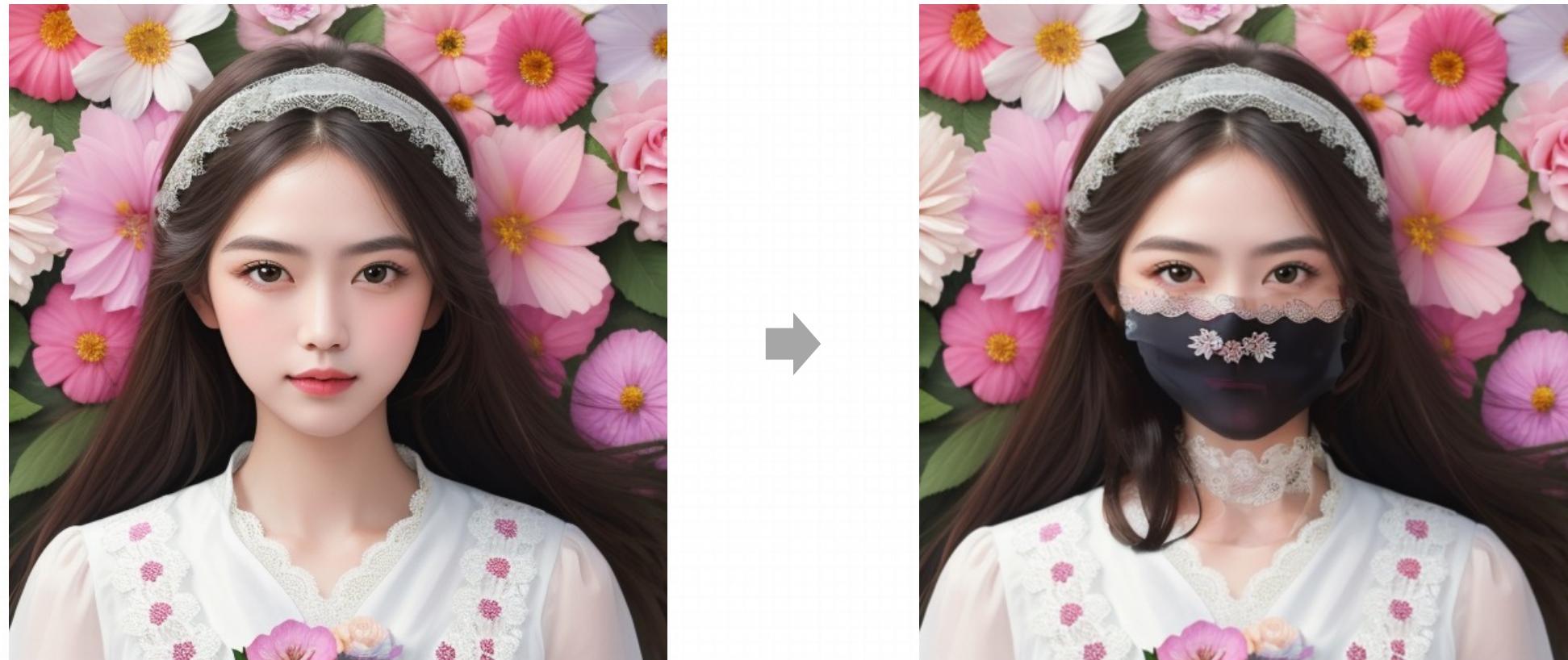
操作方式：

1. 绘制“口罩”的模样
2. 在原来的Prompt后面，添加与口罩相关的词汇，例如：face mask
3. 将重复幅度适当开大，例如0.8

这些画上去的有颜色的线条，会同时构成这个画面的一部分，并参与到图生图的过程里，他就像一个画笔一样，我们只需要确定修改的区域，通过修改Prompt的方式来修改绘制区域的修改内容。

5.2 涂鸦重绘 (inPaint Sketch)

A. inPaint Sketch 基本操作流程

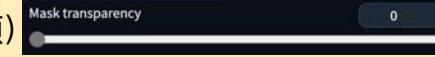


5.2 涂鸦重绘 (inPaint Sketch)

B. 使用inPaint Sketch 修复细节



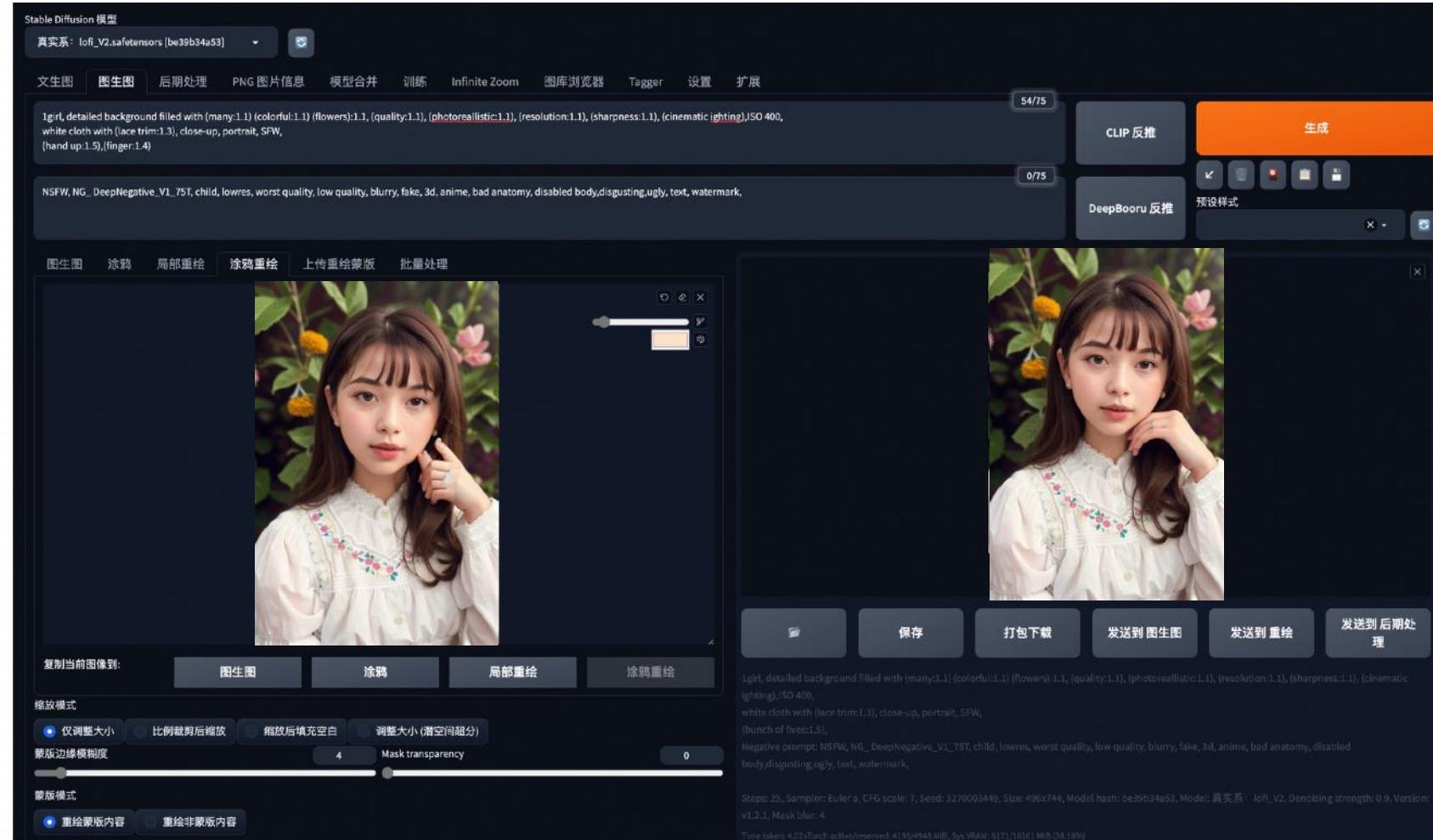
操作步骤：

1. 在inpaint sketch里加载图片
2. 使用头发的颜色覆盖手部，再使用肉色勾勒手的轮廓（注意拾色器里有一个吸管使用它，你可以吸取旁边画面里的一些颜色，带来作画的便利。）
3. 在正面提示词上添加内容
4. 在负面提示词，用上negative embeddings权重1.2，进一步抑制错手的现象
5. （比起一般局部重绘，这里主要多了一个蒙版透明度的选项）
简单理解就是这些颜色印在你的画面上的显著程度。一般我们维持默认0就是完全不透明的。如果你觉得颜色重了，可以适当开大一点点，让它稍微透明变弱一些。
6. 适当降低重绘幅度，因为太大的重复幅度会令我们勾勒出来的手部线条被模糊。同理，上面的蒙版模糊度也不宜太大，默认维持数值4。
7. 开始生成

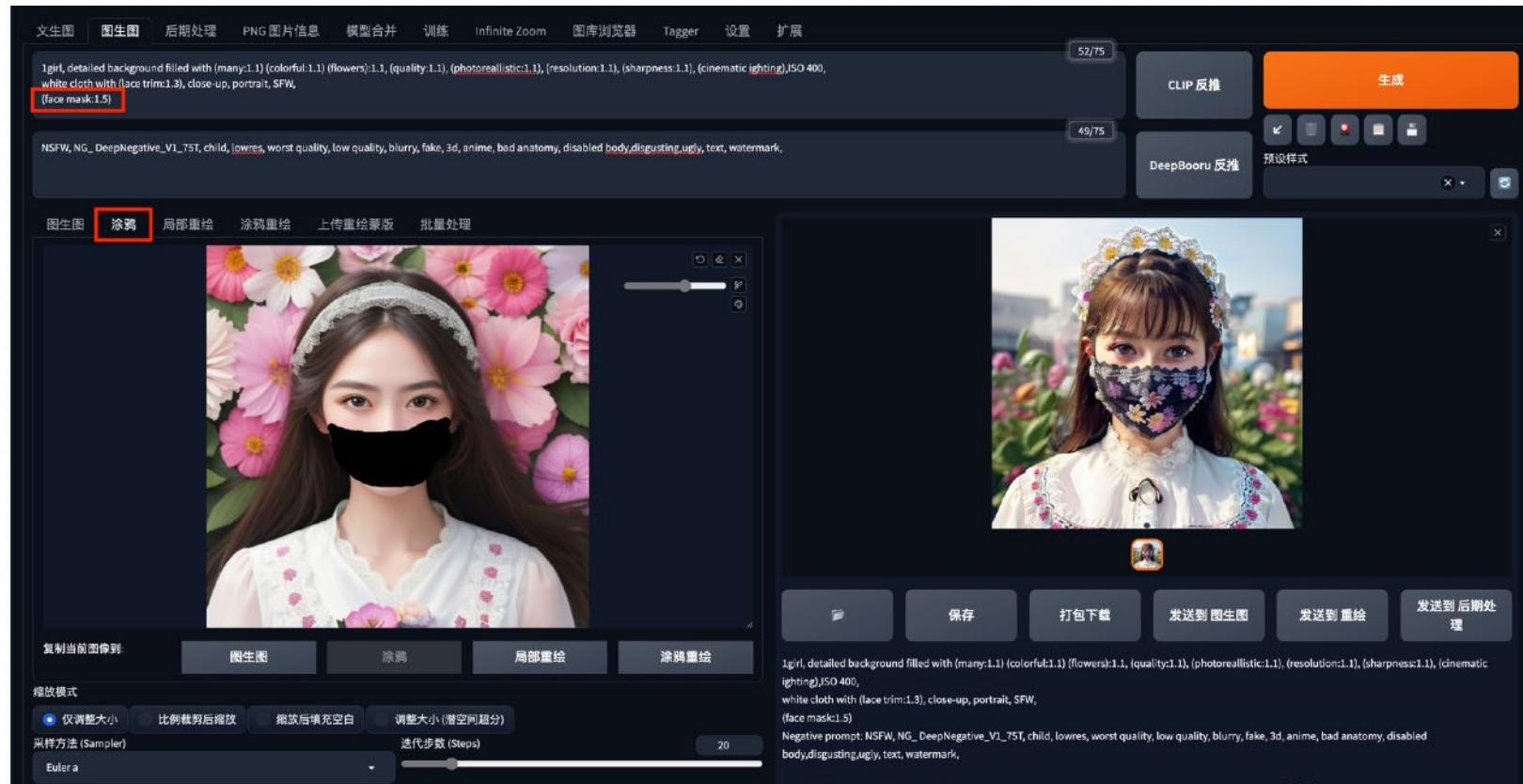
注：如果无法生成我们需要的画面，那就把seed值去掉，让系统多生成几次，并按照我们讲的参数细节反复调教一下。

5.2 涂鸦重绘 (inPaint Sketch)

B. 使用inPaint Sketch 修复细节



5.3 涂鸦 (Sketch)



Sketch选项卡中也可以对图像进行重绘，但是在这里进行重绘，并非是对于蒙版进行绘制，而是对整个图像进行修改，它会将新绘制的内容加进图片里，然后重新对整张图做一个完整的图生图。这样的结果是会导致其他部分的细节发生变化。

5.4 上传蒙版重绘 (inPaint Upload)

inpaint upload 对于有图像处理软件技术的用户来说，可以让你更加精确的控制重绘的区域范围。



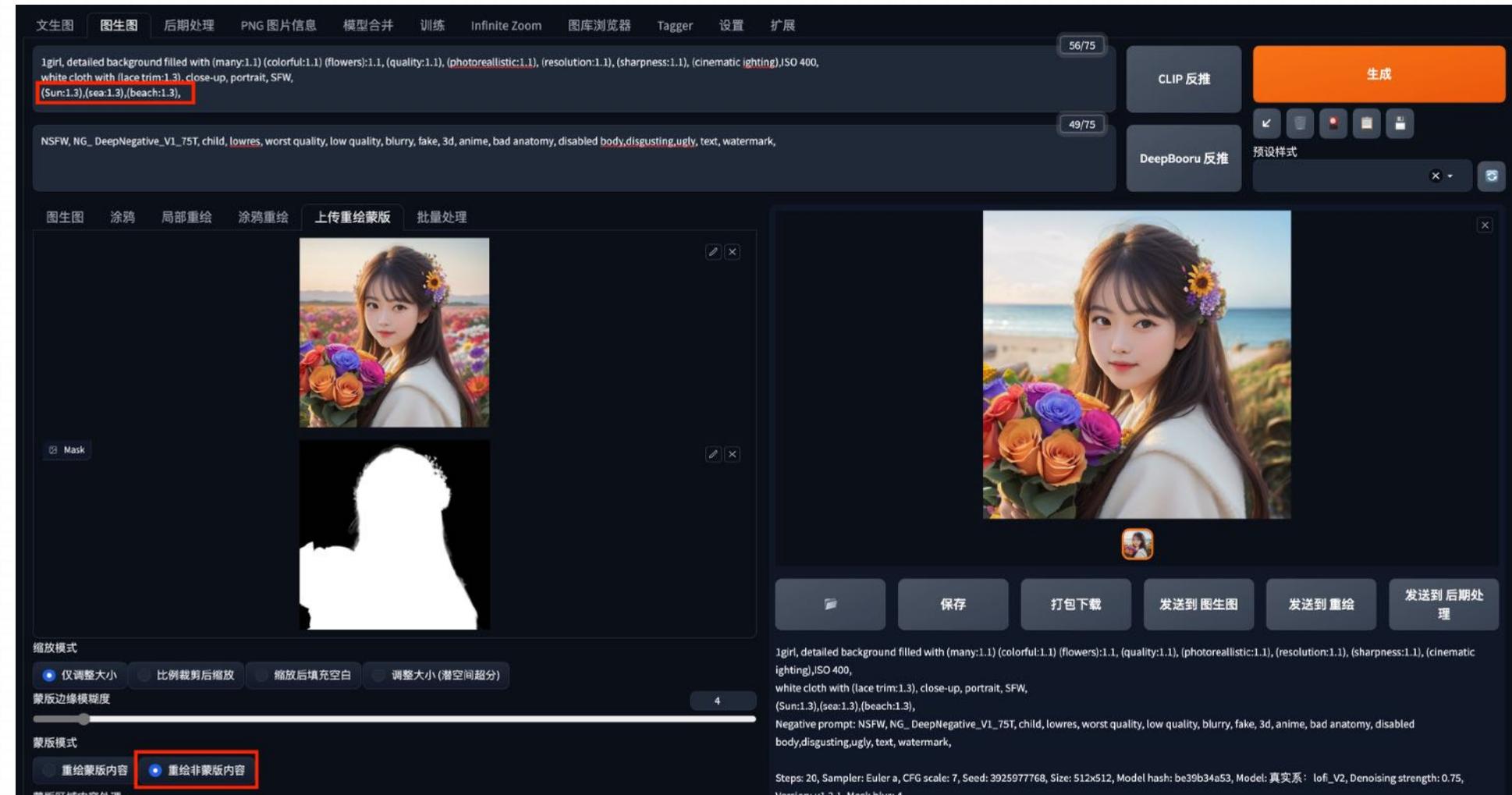
操作步骤：

1. 点击“上传重绘蒙版”标签，页面会显示两个图像上传区域.
2. 上方用户上传我们需要重绘的图片；下方需要上传一张蒙版图片来告诉AI那个区域需要重绘。

在这个蒙版图像里，白色的区域是蒙版区域，是默认需要进行重绘的，黑色部分是默认保留的。
(当然你也可以进入下面的重复参数来交换它们)

5.4 上传蒙版重绘 (inPaint Upload)

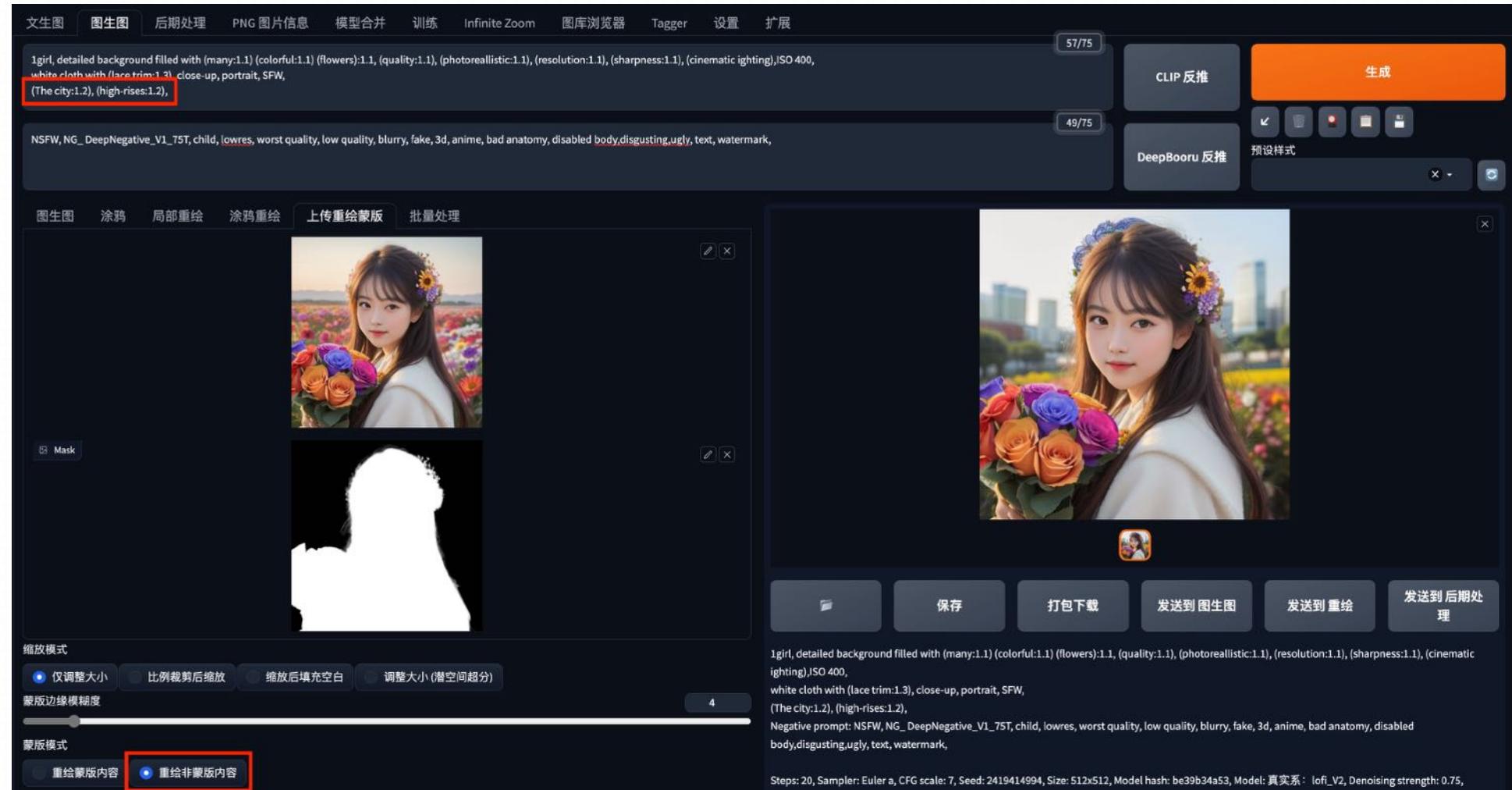
将背景更改为阳光，大海，沙滩 (Sun:1.3), (sea:1.3), (beach:1.3),



5.4 上传蒙版重绘 (inPaint Upload)

将背景更改为城市，高楼

(city:1.3), (high-rises:1.2),



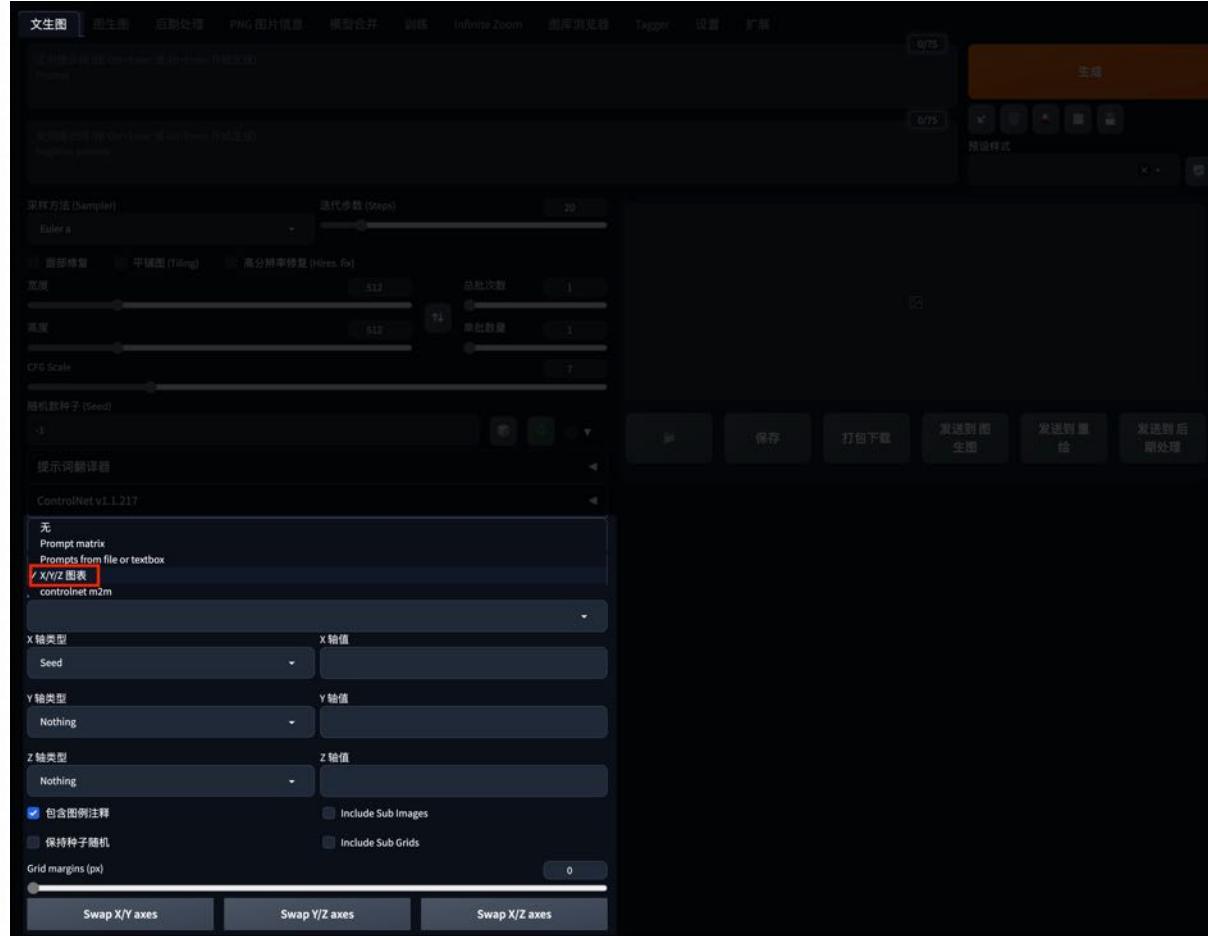
6. 图片对比

6.1 XYZ图表的使用

6.2 提示词矩阵

6.1 XYZ 图表的使用

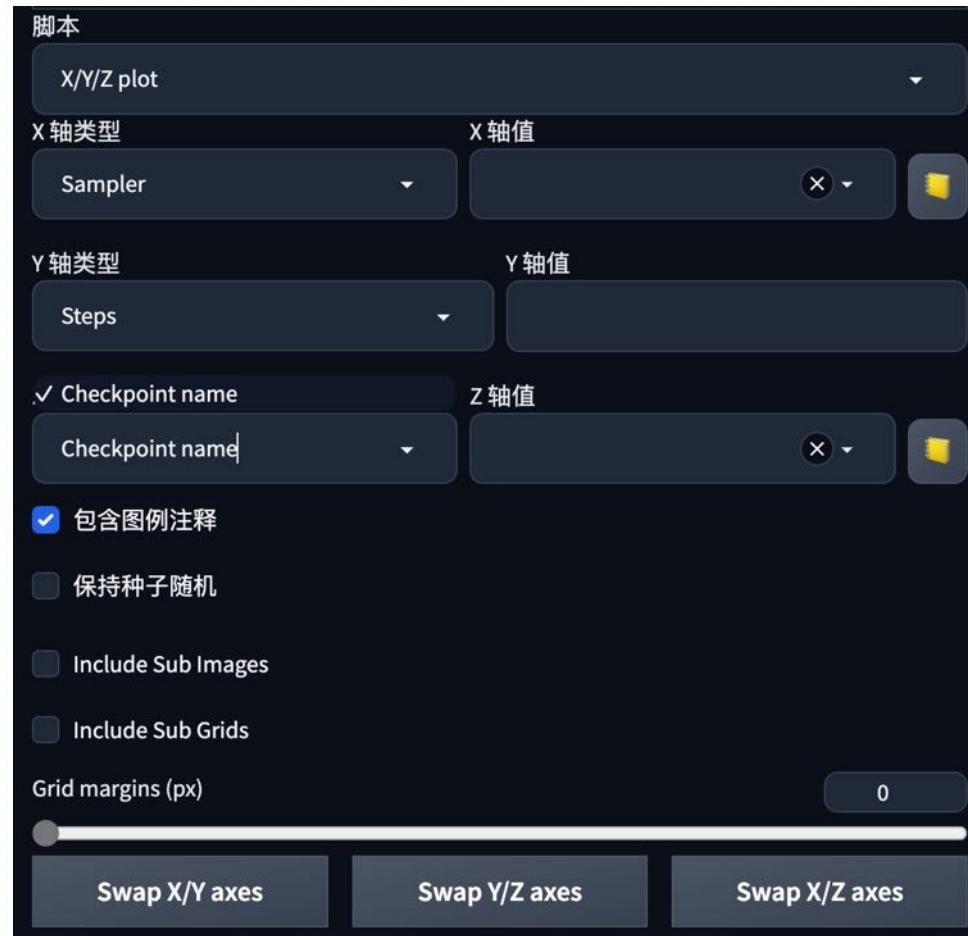
XYZ 三维图表是一种用于可视化三维数据的图表。它由三个坐标轴组成，分别代表三个变量。这个工具的作用就是可以同时查看至多三个变量对于结果的影响。具体在 sd 中，xyz 图表可以帮助我们观察到三个变量的变化对图像的影响，以便我们更好的调整参数。



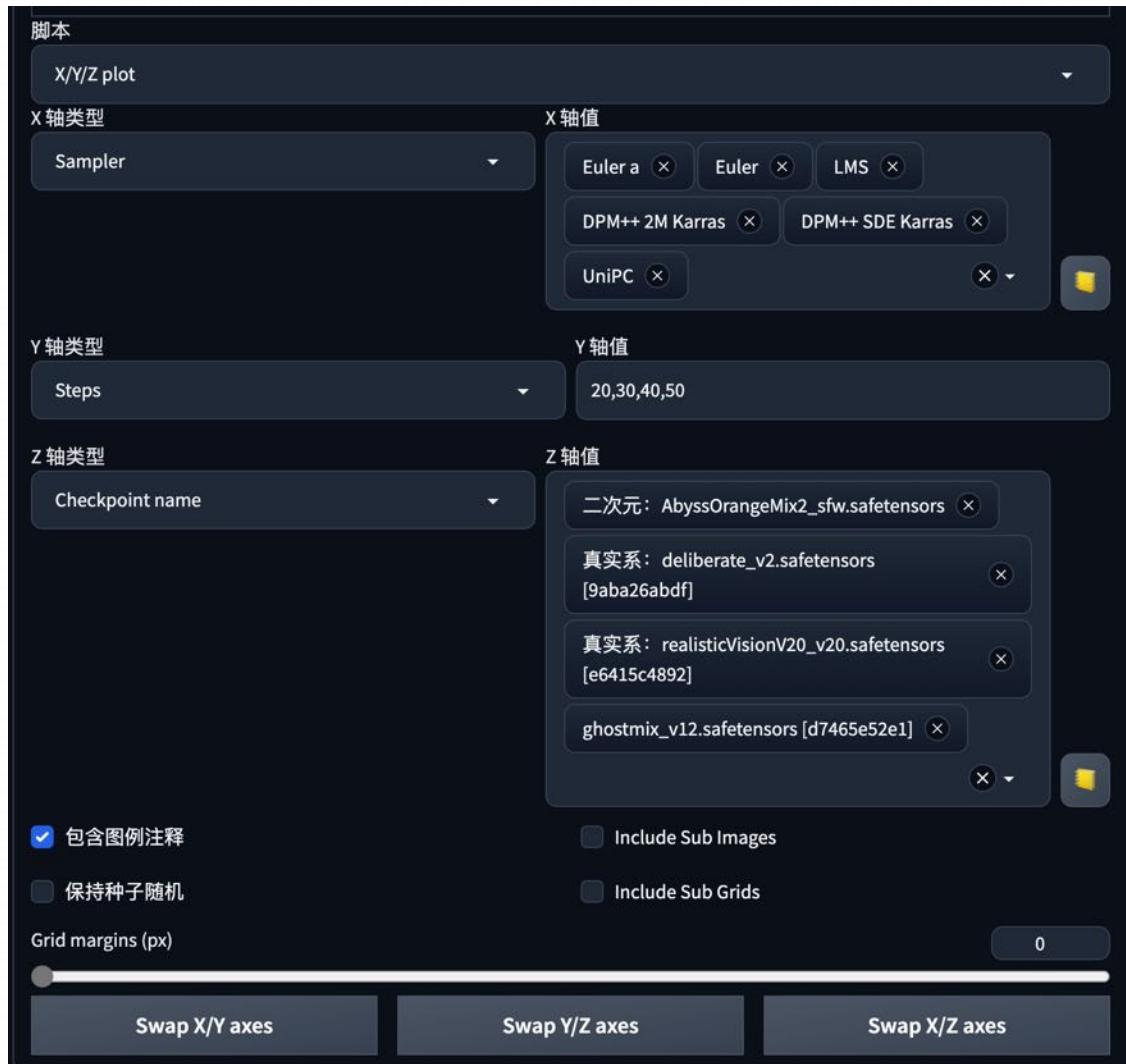
这项功能可以在 文生图/图生图 界面的左下角中“脚本”一栏内选择“X/Y/Z 图表”以启用。

6.1 XYZ 图表的使用

举个例子，我们想要测试采样器（Sampler）, 迭代步数（Steps）和基础底模型（Checkpoint name）三个变量对图片的影响，点开 X 类型，选择为采样器，Y 和 Z 依次设置为迭代步数和基础底模型。



6.1 XYZ 图表的使用



在填各个轴数值的时候，将对应的参数以英文逗号隔开即可。以 **X 轴** 的 sampler 为例，我们可以把要对比的采样器的名称复制进去。另外，也可以点击右边的“book”图标，加载本地所有的采样器，然后删除掉不需要的即可。

在填 **Y 轴** 的迭代步数时，除了可以直接输入数值外，也可以采用另外两种方式来写。

- 起点-终点（间距）**，例如:20–50 (+10) 就表示，从 20 到 50，每一步加 10。
- 起点-终点[步数]**，例如 20–50[4]就表示，从 20 到 50，一共四步。

这两种方式的效果都与直接输入“20,30,40,50”等效。

Z 轴可以把模型的名字复制进去，或者点击右边的“book”图标，选择需要比对的模型。

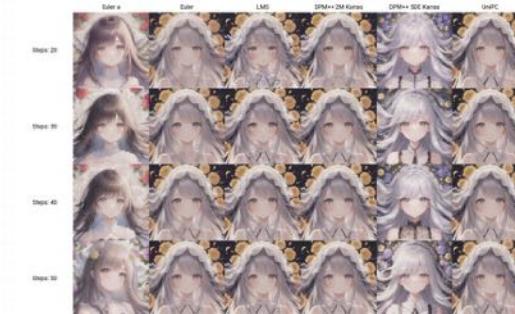
6.1 XYZ 图表的使用

最终我们会得到一个下面的网格图。

真实的realisticVisionV20_v20.safetensors
[e6415c4892]



真实的AbyssOrangeMix2_sfw.safetensors



ghostmix_v12.safetensors
[d7465e52e1]



真实的deliberate_v2.safetensors
[9aba26abdf]



真实的lofi_V2.safetensors
[be39b34a53]



6.2 提示词矩阵

A. 基本使用方式

在许多情况下，一大串从网络上抓取的提示词在某些模型中表现良好，但在更换模型后可能就无法使用。有时候，一些看似无用的提示词被移除后，画面的感觉会变得怪异，不清楚到底是哪里受到了影响。这时候，就可以使用 Prompt matrix 来深入探究原因。与之前介绍的 X/Y/Z plot 相比，Prompt matrix 的使用方式相似，都可以生成一组图表，但它们的设置方式有很大的差别。

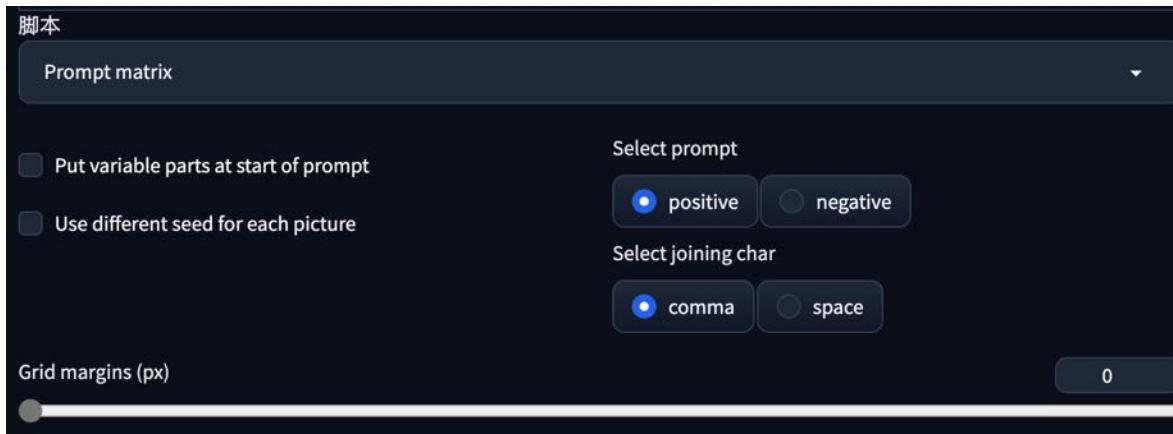
我们举个例子来说明这个脚本的使用方法：

例如我们在别人的提示词中看到了这两个词：dutch angle（倾斜的角度），autumn lights（秋天光影），虽然每个单词都懂，但是我们还是无法知道这些词对图像的影响。

6.2 提示词矩阵

A. 基本使用方式

1) 首先我们把选择提示词这个部分选中正面。（默认就是选中正面，不需要去更改）。其他参数我们保持默认即可。



2) 接下来我们将两个词填入正向提示词中，这里的格式是“|测试提示词 1|测试提示词 2”。

```
(extremely detailed CG unity 8k wallpaper), best quality, masterpiece, hdr, ultra highres |  
dutch angle |  
autumn lights
```

在上面的这组Prompt中，第一行的提示词会被用在每一张图上，第二、三行被“|”分割的提示词，会被当成矩阵提示词，交错添加在最终的图上。

6.2 提示词矩阵

A. 基本使用方式

3) 然后，我们可以产出这么一张图。



图一：什么额外提示词都没加的状态；

第二列：全都是添加了“dutch angle（倾斜角度）”这个提示词的效果。

第二行：都增加了“autumn lights（秋天光影）”提示词的效果。

图四：“dutch angle”和“autumn lights”两个 Prompt都有的效果。

6.2 提示词矩阵

B. 进阶使用

提示词矩阵并非仅支持单个词进行测试，还可以将一组词组合在一起进行测试。例如：

(extremely detailed CG unity 8k wallpaper), best quality, masterpiece, hdr,
ultra highres |

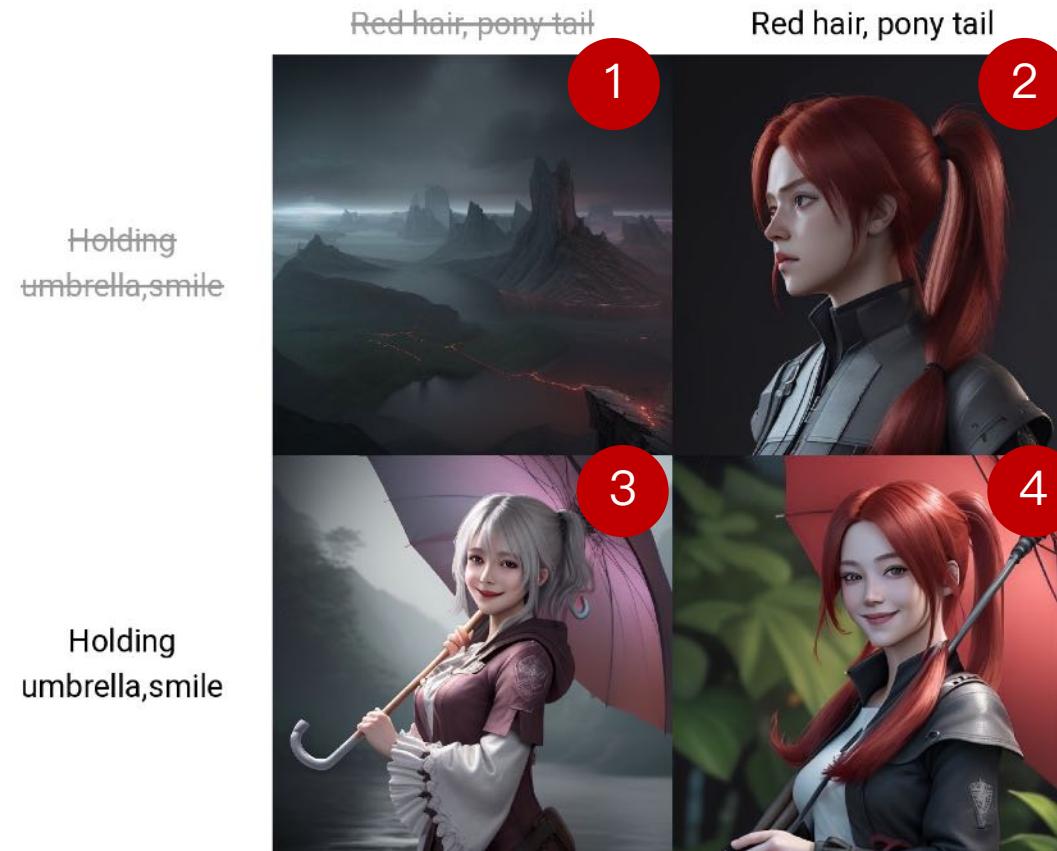
Red hair, pony tail |

Holding umbrella, smile

6.2 提示词矩阵

B. 进阶使用

以此作为提示词产出的图如下：



图一：什么额外 prompt 都没加的结果；

第二列：添加了“red hair, pony tail”这个提示词的效果。

第二行：增加了“Holding umbrella, smile”提示词的效果。

图四：“red hair, pony tail”和“Holding umbrella, smile”两个Prompt都有的效果。



智能大数据可视化实验室

INTELLIGENT BIG DATA VISUALIZATION LAB