Merge Block Weight 魔法密录

1.0 正式版

此版本为正式版，基于之前的 beta 版进行一定修正和增改。

如有错误，欢迎指正。

作者：AIbedo0，一条菜狗，胤白凉，tddok（排名不分先后）

一.写在前面

文档发布以来，有人成功融合出了自己非常满意的模型，也有人认为分层融合是不可控的玄学。当然我们也认为确实有一定的**不可控性**，比如因为各种变量会导致分层负责的部分上下偏移，以及模型融合（不论是传统还是分层）本身的**不可预测性质。如果你想要一个完全可控的效果，这篇文档也许并不能满足你，可以马上关闭避免浪费了你的时间。**

因为作者的能力有限，不能保证文档的完全正确性，但我们保证，文档的大部分都是可用且具有正确性的。

二.MBW 简介

MBW 即 Merge Block Weight，GITHUB 地址为：https://github.com/bbc-mc/sdweb-merge-block-weighted-gui，作者是 BBC-MC

MBW 插件的作用是：将模型的 25 个 U-NET 层和 base_alpha 分别分配权重进行融合

既然这比传统融合复杂那么多，那有什么优势呢？我们列举了一些使用 MBW 融模的好处：

1.进行更加精细的模型微调
2.修复颜色较灰的模型，或迁移其他模型的整体色调（PS：画面灰先检查 VAE！）
3.迁移模型的光影和质感
4.其他模型带来的笔触感、线条
5.如果融合构图更好的 B 模型，可以同样在不改变画风的情况下，改善 A 模型的构图。
6.用真人模型进行人体修复
7.保持画风的基础上，缓解模型的过拟合
8.对手部进行一定修复
9.把整个模型的画风迁移至另一模型

10.可以对脸的画风进行一定程度的微调

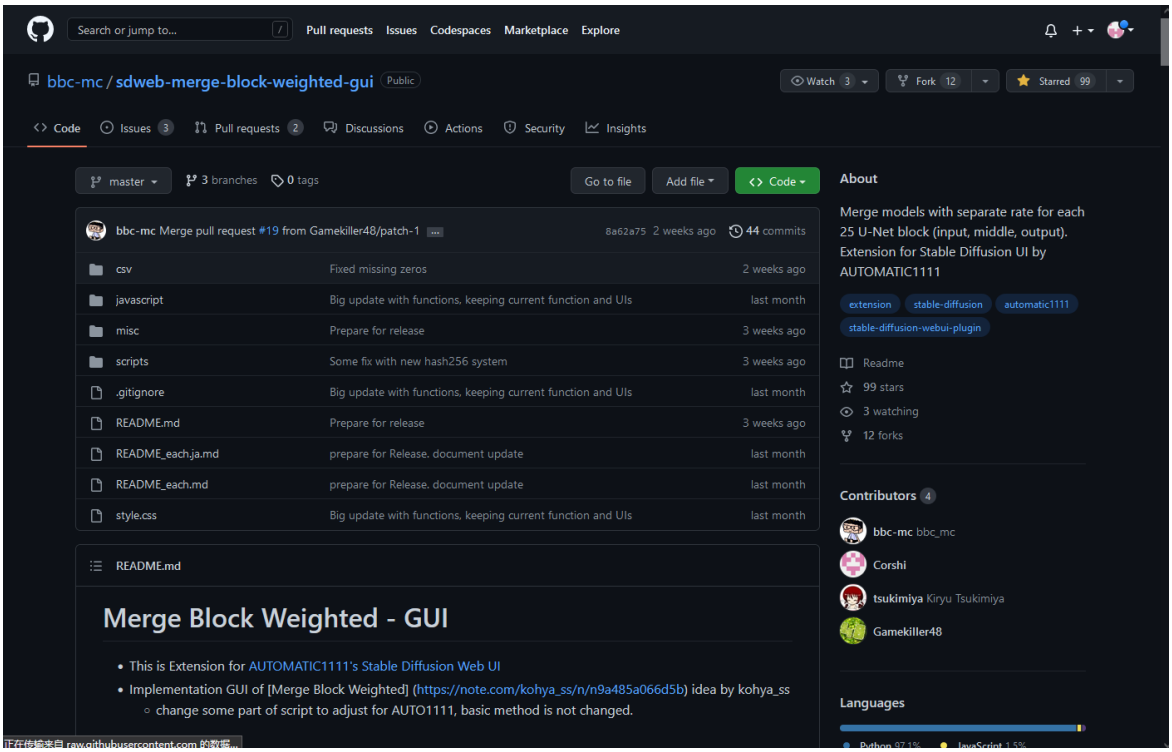
当然还有很多方向可以挖掘，以上只是作为一些参考，更多方向可以一起讨论。

三.安装及简单教程

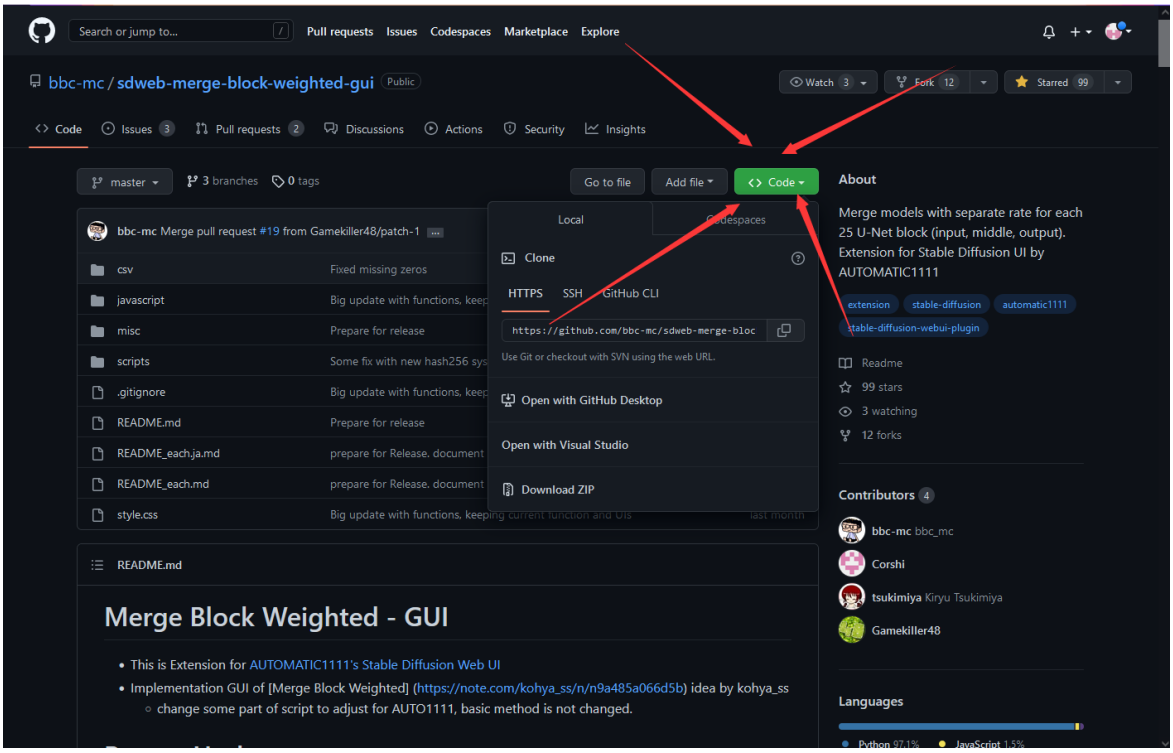
1.安装

这个插件不是 web-ui 自带的，所以需要先安装，以下讲解安装步骤。

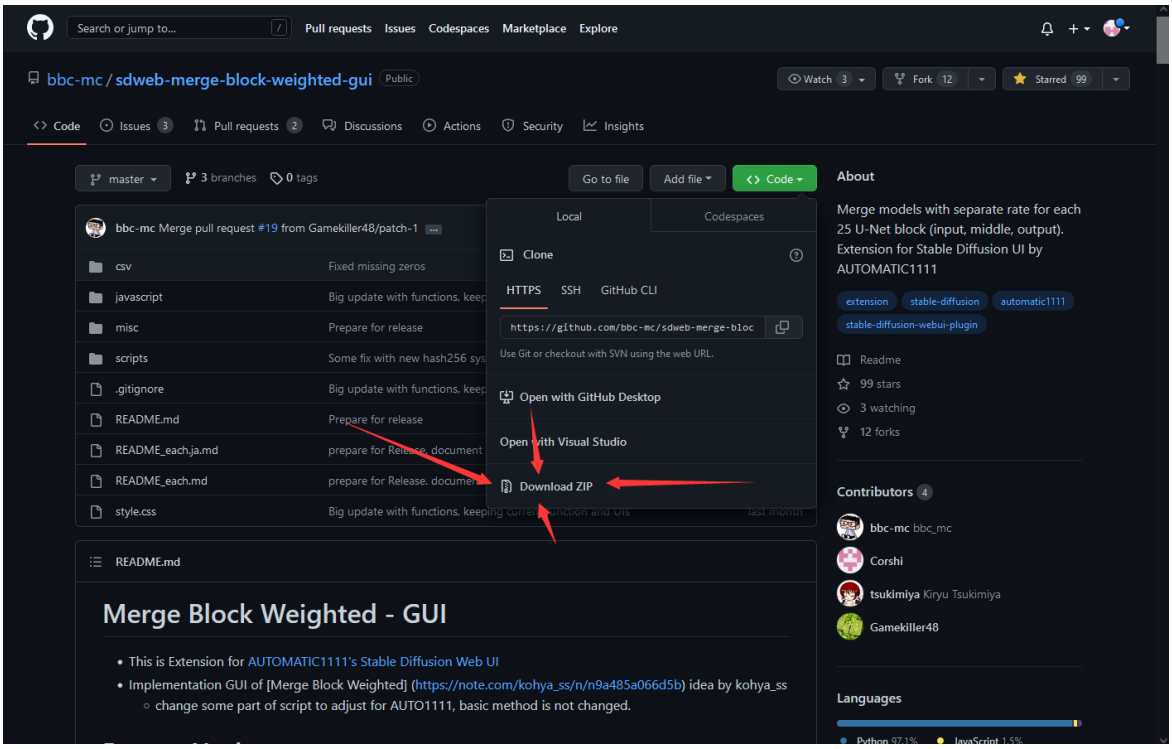
第一步：打开这个网站 https://github.com/bbc-mc/sdweb-merge-block-weighted-gui



打开后可以看到这样的界面。

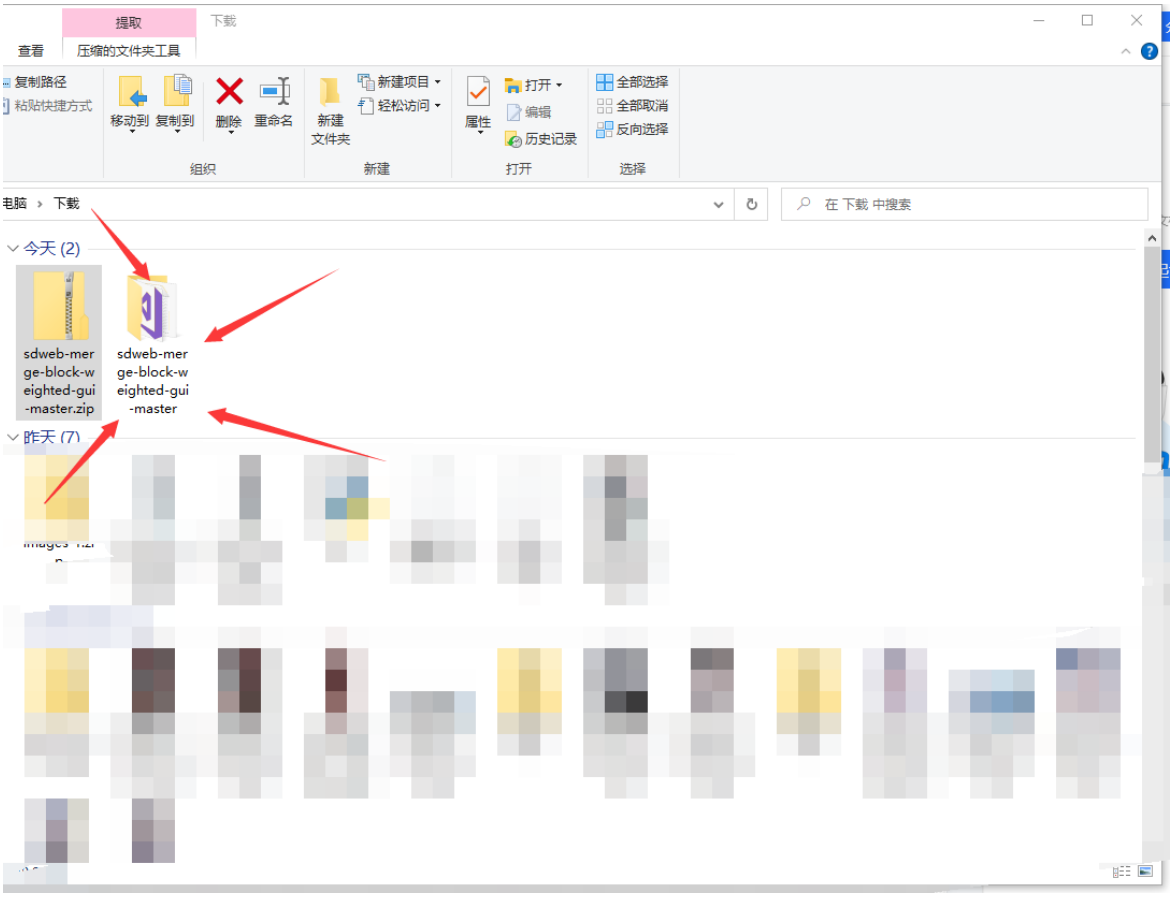


第二步：下载，先点击这个绿色的按钮。

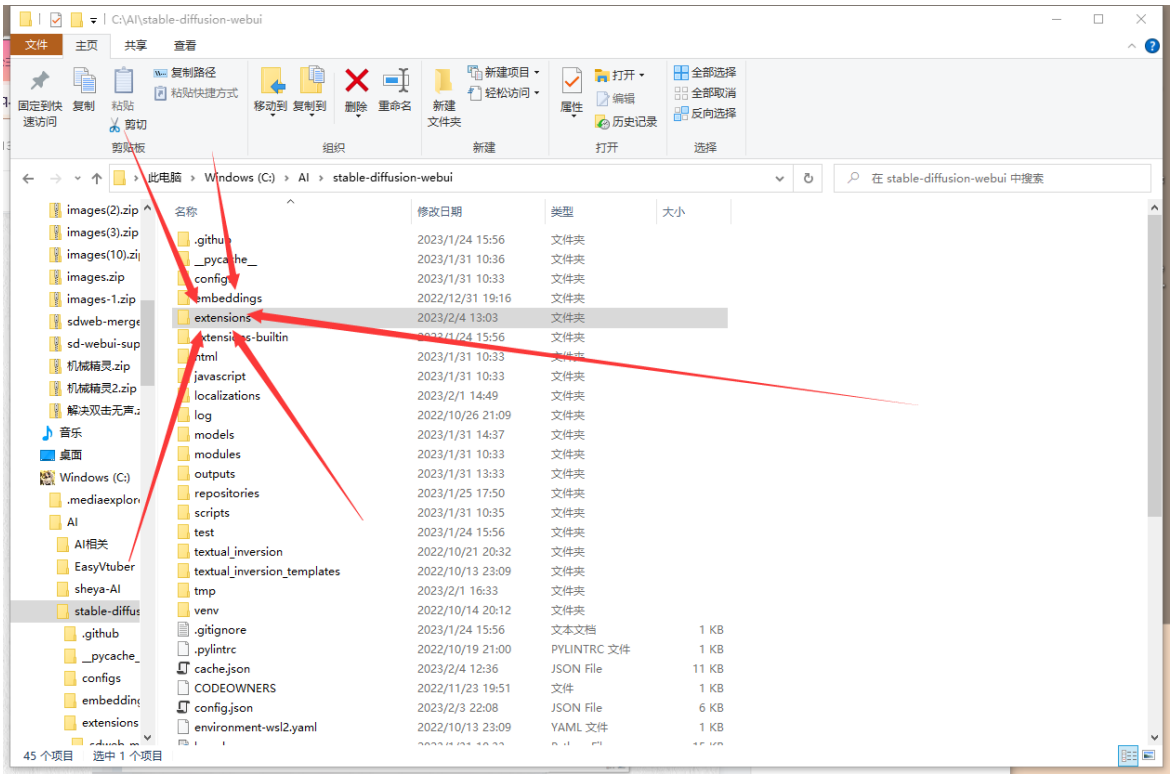


在点击这个 Download ZIP 按钮。

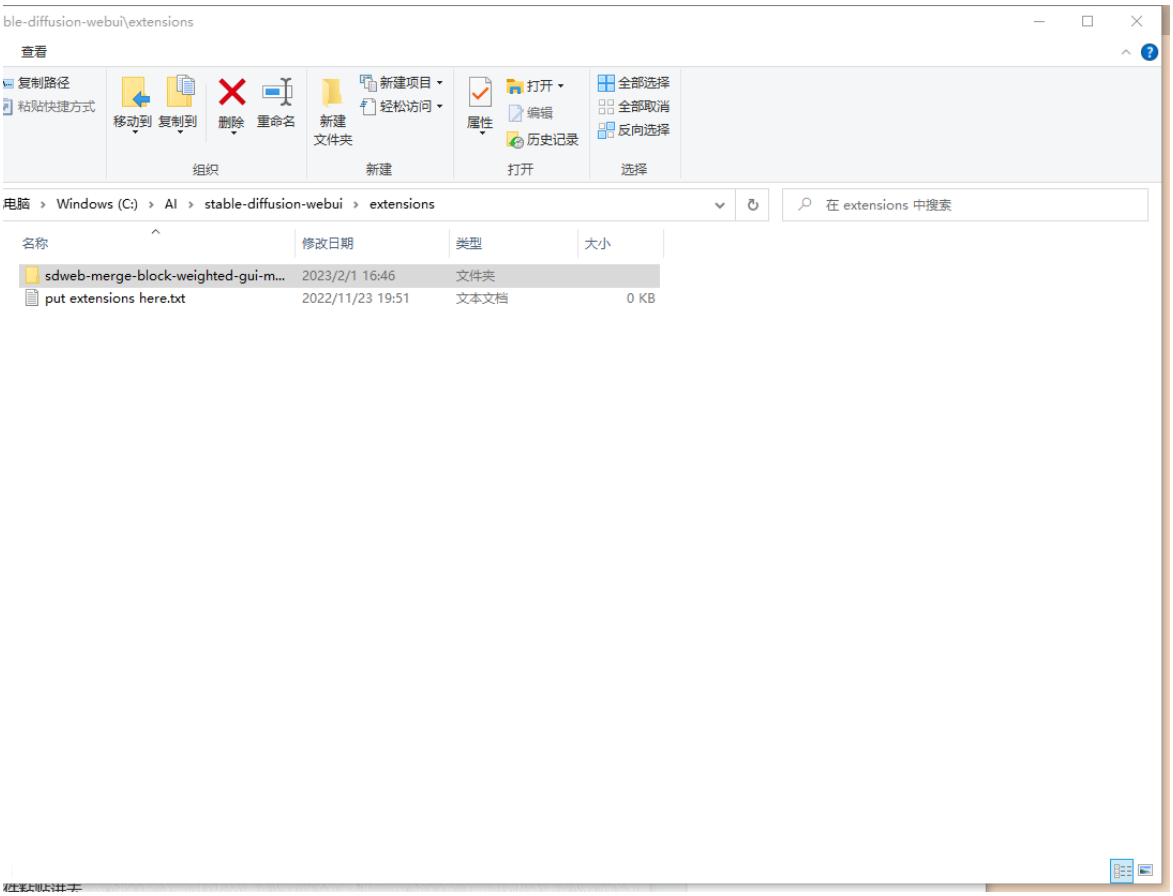
等待下载完毕。



第三步：安装，先解压下载的安装包，然后复制解压后的文件。



之后打开安装 web-ui 的文件夹，再打开 extensions 文件夹。



把复制的文件粘贴进去

最后重启 web-ui。

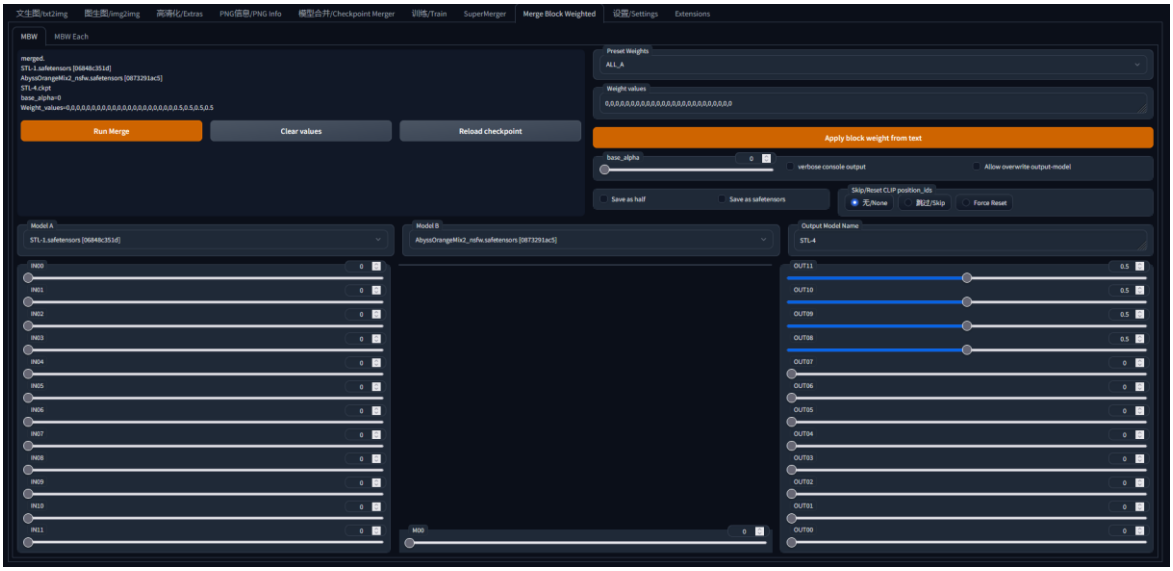
在页面顶部看到这个即为安装成功。

2.功能介绍

安装完成后就要进行使用了。

接下来分别讲解每个选项的作用。

打开融合页面后可以看到这样的一个界面。



以下为按钮的功能介绍,重要功能会加粗处理：

Run Merge :开始融合

Clear values:将下方的滑条全部置为 0.5

Reload checkpoint:重新读取模型列表

Model A:选择需要融合的模型 A

Model B:选择需要融合的模型 B

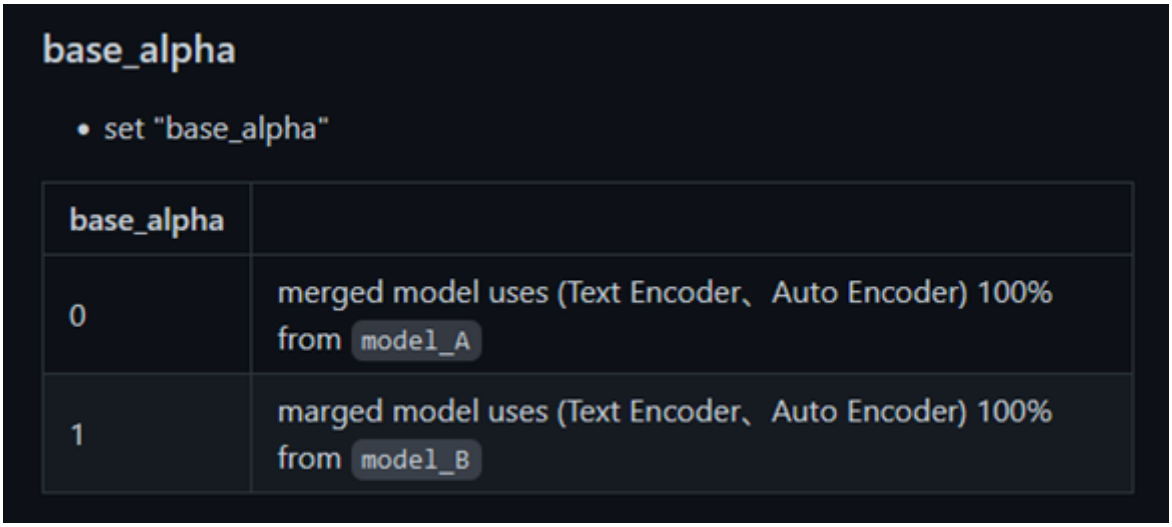
Output Model Name:融合后的模型名字

Preset Weights:官方的融合预设（选择预设后将直接设定给下方滑块）

Weight values:融合预设的文本(可自行填写)

Apply block weight from text:将 Weight values 中的值写入到下方的滑条中

base_alpha:A 和 B 模型的文本编码器，自动编码器(语义分析模块)的融合比(附上原文和机翻，具体作用后文讲解)



0　　模型使用(文本编码器，自动编码器)100%来自 model_A

1　　模型使用(文本编码器，自动编码器)100%来自 model_B

verbose console output:是否输出 CLIP 的附加信息(不影响模型融合)

Allow overwrite output-model:是否覆盖同名文件

Save as half:保存为半精度(float16)模型

Save as safetensors:保存为.safetensors 格式

Skip/Reset CLIP position_ids:跳过或重置 CLIP position_ids(注:Force Reset 选项即重置的意思)

IN0-11,M00,OUT0-11 共 25 个滑条:分别调整 A 和 B 两模型每层的融合比

当滑条的值为 1 时，意味着滑条控制的层将完全等于 B 的这一层，

当滑条的值为 0 时，意味着滑条控制的层将完全等于 A 的这一层

这是滑条的基础原理。

注:官方 readme 文档:https://github.com/bbc-mc/sdweb-merge-block-weighted-gui(生肉)

四.MBW 融模入门（猴子也能学会的分层融合）

下面是几个预设，方便娱乐玩家满足抽卡乐趣。不理解理论，纯套预设，完全不保证效果哦。当然也会给出一些对于预设的分析用于参考。

1.官方预设

第一行为 A 模型
第二行为预设融合结果
第三行为 B 模型
BASE ALPHA=0

GRAD_V
例图：

AbyssOrangeMix2_sfw x (1-alpha)
pastelMixStylizedAnime_pastelMixPrunedFP32 x alpha
alpha = 0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.
5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5
,0.5,0.5,0.5
beta = not used

seed = 1545 | 41 | 554

mbw weights
=

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1

评价：基础款中的基础款，中规中矩，能融合 B 模型的整体色调，融合一部分 B 模型的主体内容，以及一些细节。

R_SMOOTHSTEP*2

例图：

评价：GRAD_V 加强款，用力更猛。同样的能融合 B 模型的整体色调，主体内容，以及一些细节。

SMOOTHSTEP*3

例图：

评价：融合 B 模型的整体色调和整体构图，以及部分画面主体的处理方式。

R_SMOOTHSTEP*4

例图：

评价：在保留**一部分** A 模型主体内容的基础上融合 B 模型

SMOOTHSTEP/3

例图：

评价：融合 B 模型的整体色调和整体构图。SMOOTHSTEP*3 的构图加强版，颜色削弱版。

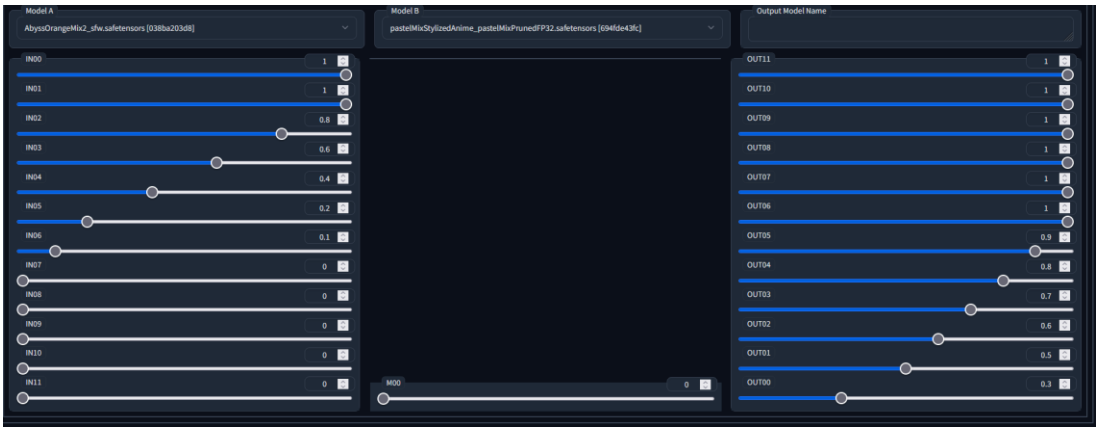
R_SMOOTHSTEP/4

例图：

评价：在保留**大部分** A 模型主体内容的基础上融合 B 模型，R_SMOOTHSTEP*4 的轻量版。

2.非官方预设

2-1.画风改换



参数文本：1,1,0.8,0.6,0.4,0.2,0.1,0,0,0,0,0,0,0.3,0.5,0.6,0.7,0.8,0.9,1,1,1,1,1,1

BASE ALPHA=0

使用以上预设。

此处使用 AMO2_sfw（深渊橘子 2）和蜡笔模型（PM）进行融合。输出文件名自填。

以下为融合的效果图：

AbyssOrangeMix2_sfw x (1-alpha)
pastelMixStylizedAnime_pastelMixPrunedFP32 x alpha
alpha = 0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5
beta = not used

seed = 1545 | 41 | 554

mbw weights
=

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

0,1,1,0.8,0.6,0.4,0.2,0,1,0,0,0,0,0,0,0,0.3,0.5,0.6,0.7,0.8,0.9,1,1,1,1,1

1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1

1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1

不同层分别是：

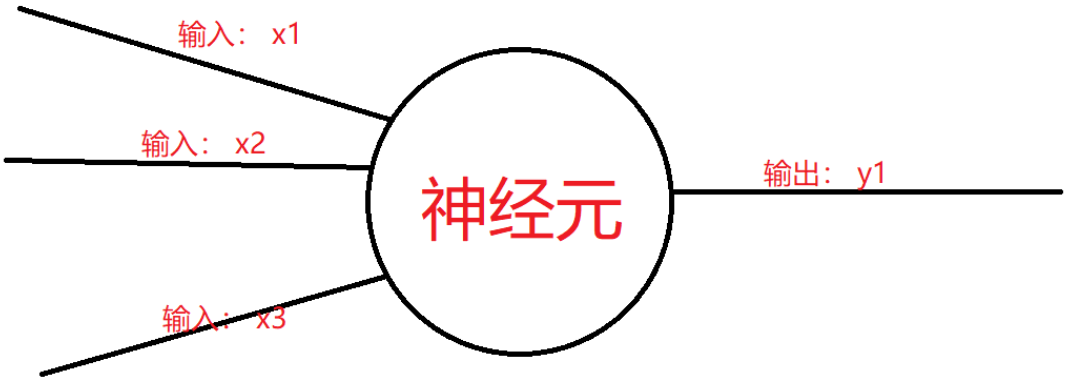
1. 橘子模型原模型
2. 预设融合效果
3. 传统融合效果
4. 蜡笔模型原模型

五.MBW 融模进阶

1.基础知识

1-1.神经网络是什么

要了解神经网络，我们要首先了解神经元的结构。



这是一个神经元，输入　x1，x2，x3 都是纯数值，输出 y1 也是

神经元会通过一定的计算来得到输出，那神经元是如何计算的呢？

首先，每个神经元会有几个数值作为它的权重，帮助它进行计算，

权重数=输入的个数+1

图中的神经元有三个输入，共四个权重，我给它们起名为 w1，w2，w3，w4.

在一般的神经网络中，权重是训练的时候确定的，但是我们的这篇文档不涉及训练，所以就不讲它是怎么来的了。

接下来我们认识一个函数，Logistic 函数，它是干嘛的呢？

简单的说，它可以把负无穷到正无穷之间的某个数转换为 0 到 1 之间的某个数。

接下来就可以计算输出 y1 了

x1*w1+x2*w2+x3*w3=tmp

把 tmp 放进 Logistic 函数里

Logistic（tmp）=tmp2

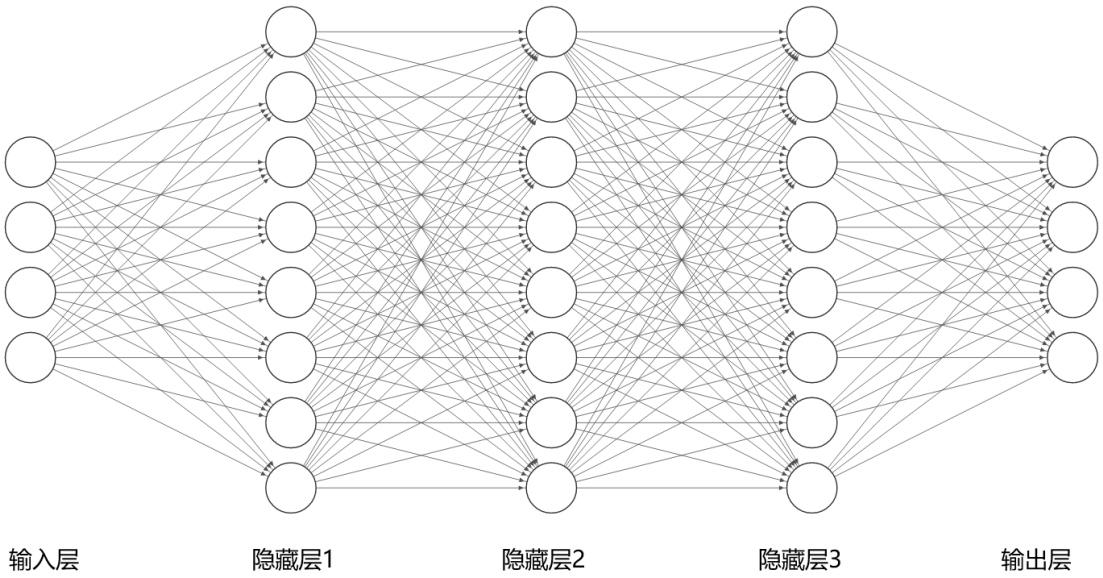
再把 tmp2 进行偏置，简单的说，就是加上 w4

tmp2+w4=y1

这样，我们就得到了 y1，神经元就是通过以上的过程，计算出结果的。

那，什么是神经网络呢？

简单的说，神经网络就是由无数神经元组成的网络。



每个神经网络都由，输入层，隐藏层，和输出层组成。

图中的神经网络有三层隐藏层，共 5 层，每层隐藏层都由 8 个神经元组成，它们接收来自上层的数据，进行计算后将结果送往下一层。

注：有兴趣的可以自行深入了解，https://www.bilibili.com/video/BV1bx411M7Zx/

1-2.绘图模型的工作原理

要理解模型的工作原理，就首先需要知道模型的组成。

模型有 3 大部分组成：

VAE（图像编/解码器）

U-net 网络（预测需要去除的噪声）

text encoder（将 tag 翻译为 embedding）

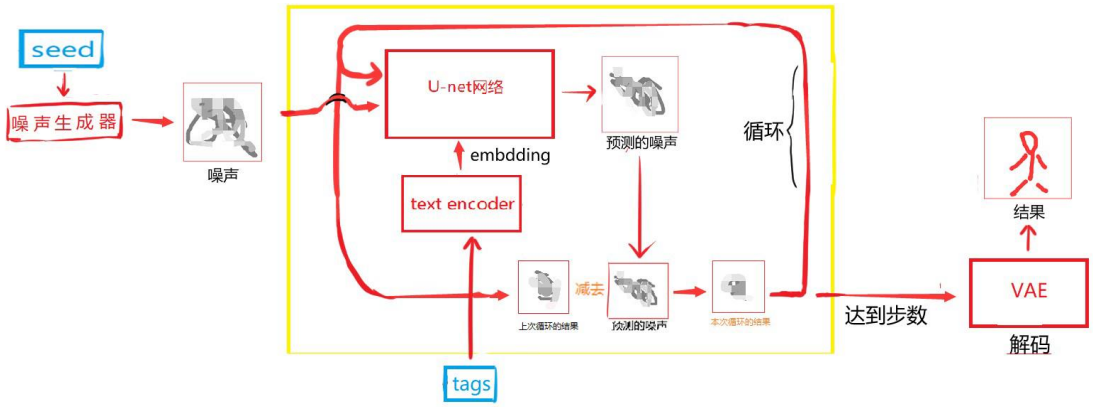
接下来我将分别讲解每个部分的作用。

VAE 会把 AI 输出的图片进行翻译，让它变成正常的图片，所以 VAE 会影响画面色彩，因为不同的 VAE 有不同的翻译方式。

U-net 会预测需要去除的噪声。简单的说，就是找出需要改变的地方并给出改变的数据。

text encoder 会把 tag 翻译成 U-net 网络能理解的形式（embedding），让模型能画出你想要的画面

接下来是模型的工作方式。



首先，噪声生成器会根据 seed（种子）生成一副噪声图，然后 text encoder 会把 tag 翻译成 embedding 并发给 U-net 网络，

U-net 网络根据 embedding 和初始噪声图生成预测的噪声 1，然后程序再从初始噪声图中减去预测的噪声 1，得到新的图片 1，

再把新的图片 1 像初始噪声图一样放入 U-net 中得到预测的噪声 2，再从新的图片中减去新的图片 2.....

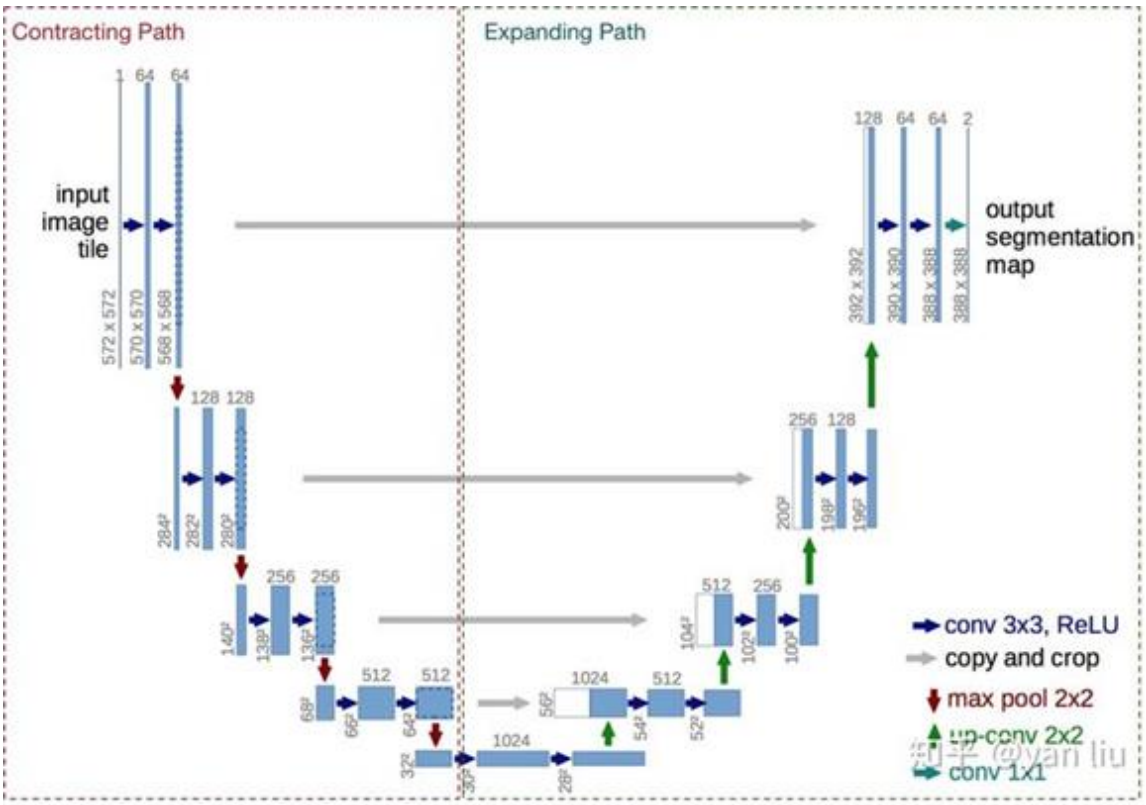
如此往复直到达到规定步数为止。

达到步数后，就将得到的新的图片 N 放入 VAE 解码，这样你就得到了一张图片。

1-3.U-net 的工作原理及结构

根据之前的基础知识，你应该知道 U-net 是作为一个修正者一样的角色存在的，它会一次

次的修复图片，直到达到步数为止。



以上是一个 U-net 网络的结构简图。

U-net 是一种神经网络的结构,在 MBW 插件中,U-net 有 25 个可调层,我们将其分为:

对应 IN 滑条的 IN 区

对应 M00 滑条的 M 区

对应 OUT 滑条的 OUT 区

IN 区有 12 层,M 区有 1 层,OUT 区有 12 层

图的左半边代表的是 IN 区，图的中间代表的是 M 区，图的右半边代表的是 OUT 区。

层大致的排列方式是：

IN00,IN01,IN02.....IN11,M00,OUT00,OUT01,OUT02.....OUT11

牢记模型结构能让你深刻的理解造成不同层差异的原因。

1-4.base_alpha

在之前的章节中，一直有一个参数没有讲解，那就是 base_alpha.

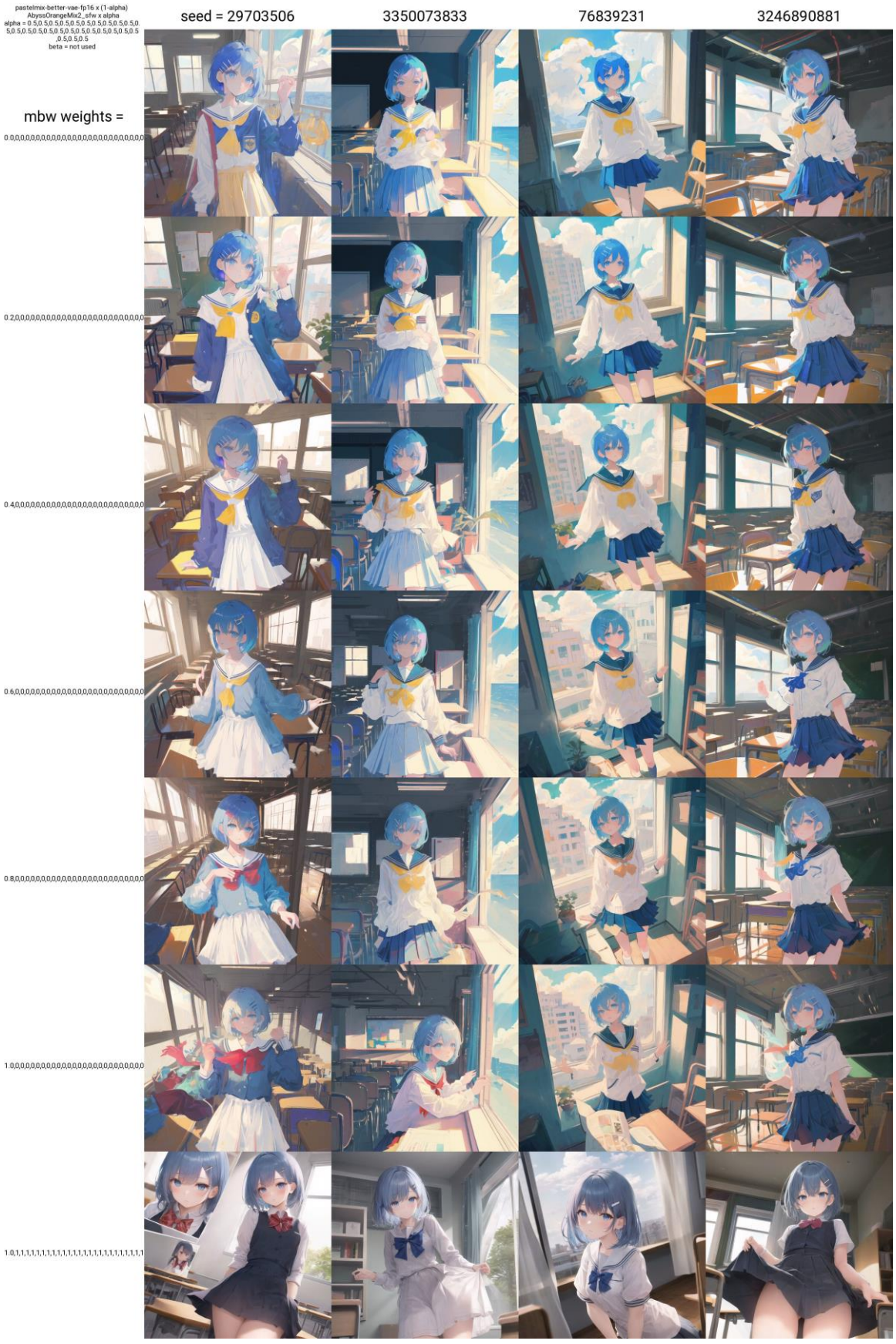
上文中我们对 base_alpha 的解释是：A 和 B 模型的语义分析模块的融合比，

好像很复杂的样子

其实不复杂，其实 U-net 这个“画家“是并不能直接理解 tags 的，所以它需要一个翻译，翻译

是谁呢，那就是语义分析模块。

接下来给出几张只调整 base_alpha 的对比图


pastelmix-better-vae-fp16 x (1-alpha)
AbyssOrangeMix2_sfw x alpha
alpha = 0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5
beta = not used
seed = 29703506
3350073833
76839231
3246890881
mbw weights =
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0.2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0.4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0.6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0.8,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1.0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1.0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1

1-6 排是蜡笔模型语义分析模块融合 AOM2_sfw 语义分析模块模型以不同配比融合的对比图，最后一排是 AOM2_sfw 的原模型。

可以看到，随着 AOM2_sfw 语义分析模块的占比渐渐提高，可以看到蜡笔模型的构图和画风在渐渐变化。由此我们可以验证，语义分析模块可以通过影响 U-net 对 Tag 的理解从而影响模型的构图和画风。

补充：有些训练可能给 tag 赋予了新的含义，才导致画风改变，比如从正常画面变成素描（sketch）风格或者 Q 版（chibi）风格

以下为 TE(text encoder)影响画风的实验测试：

base bloom x (1-alpha)
JazzJack x alpha
alpha = 0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5
beta = not used

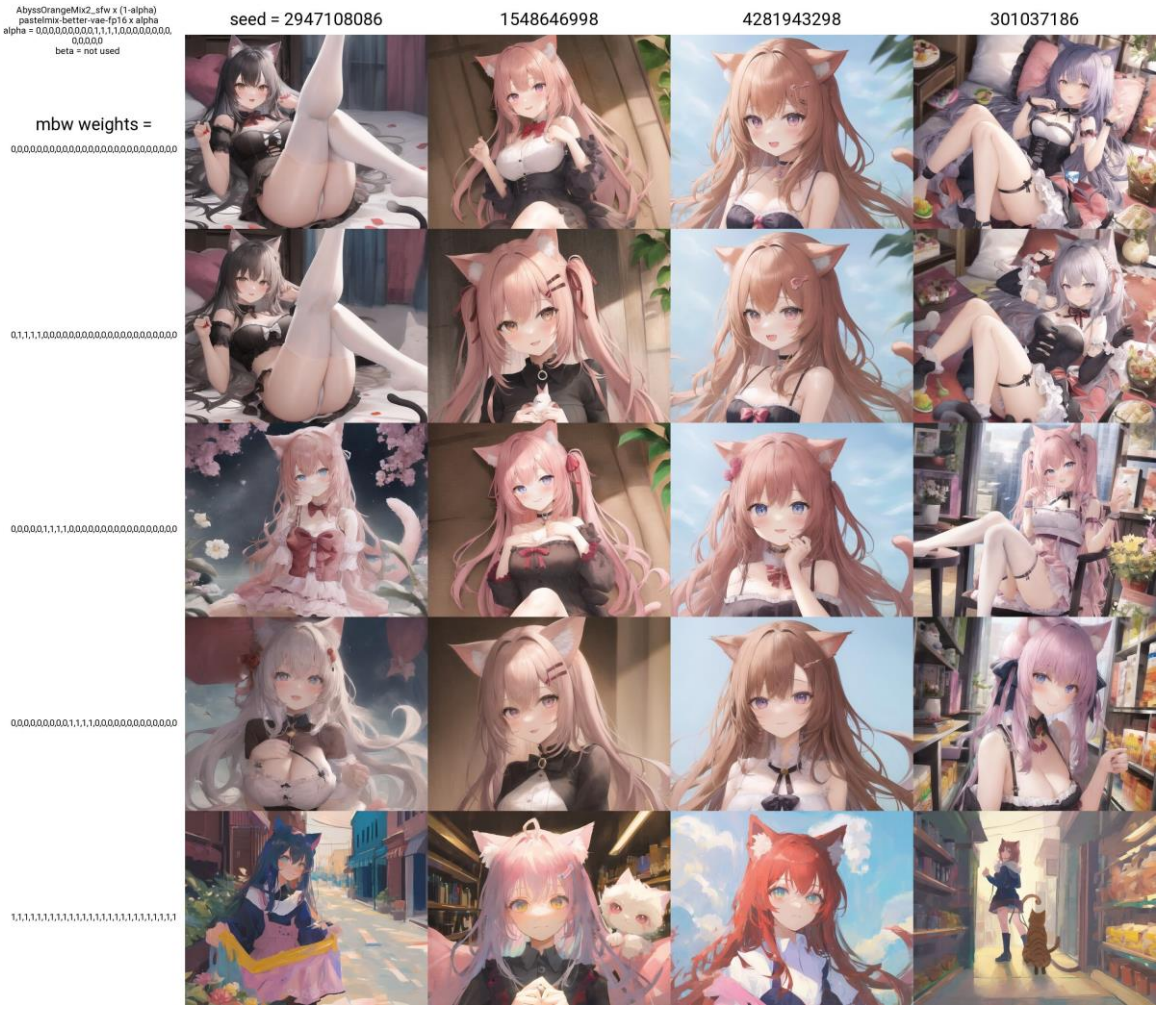
mbw weights =

	seed = 3346165981	389883228	661662040
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0			
1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0			
0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0			
1,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0			
0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0			
1,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0			
0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0			
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1			

1-5.IN 区块

IN 层的实验为了**防止 base alpha 的影响**，尽量屏蔽了构图相关的参数，只跑单 TAG 并且 CFG 在 3~5 左右

base alpha 在长 TAG 下影响很大，甚至能导致：即使改变所有 IN 层结构，构图都不会有太大的变化，这一点需要注意。




AbyssOrangeMix2_sfw x (1-alpha)
pastelmix-better-vae-fp16 x alpha
alpha = 0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0
beta = not used
seed = 1019924797
2980606875
4090342307
3317338561
mbw weights =
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1

以上图片中每一层的模型分别为：

1.橘子原模型

2.橘子融蜡笔 IN00-03（权重为 1）

3.橘子融蜡笔 IN04-07（权重为 1）

4.橘子融蜡笔 IN08-11（权重为 1）

5.蜡笔原模型

首先，要理解 IN 区和 OUT 区的区别是什么，我们要先分清楚一副画由那些部分构成。

构成画风的三大部分分别是：

1.平面构成（即平面形状，决定了构图）

2.色彩构成（画面色彩）

3.形状构成（即物体的三维形状，决定了光影）

由于目前的 AI 并没有太强形状构成的能力，所以我们现在忽略掉它。

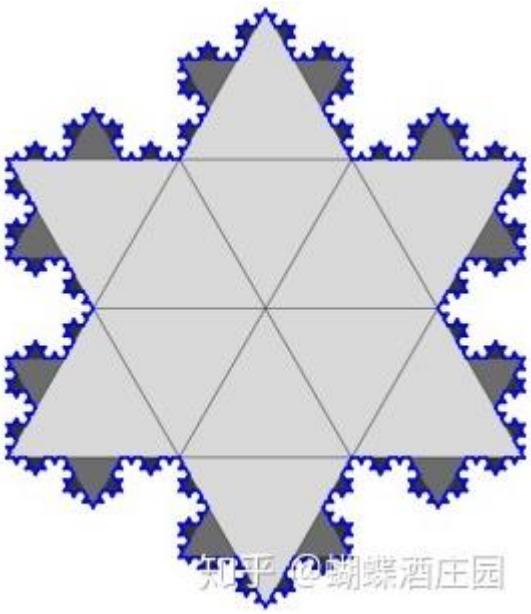
在一个 U-net 中，**IN 区块负责平面构成的相关工作，而 OUT 区块负责色彩构成的相关工作。**

那么平面构成是干啥的呢？

简明扼要的说，**平面构成就是组合各种图形，控制它们的大小，位置和形状。**

在 IN 区块中，越靠近 00 的层负责的平面构成越精细化，细节化；而靠近 11 的层负责的平面构成就越大体化，整体化。

例如有以下一幅图：



图中：

灰色的部分的是高层的平面构成所管理的（IN08-11）

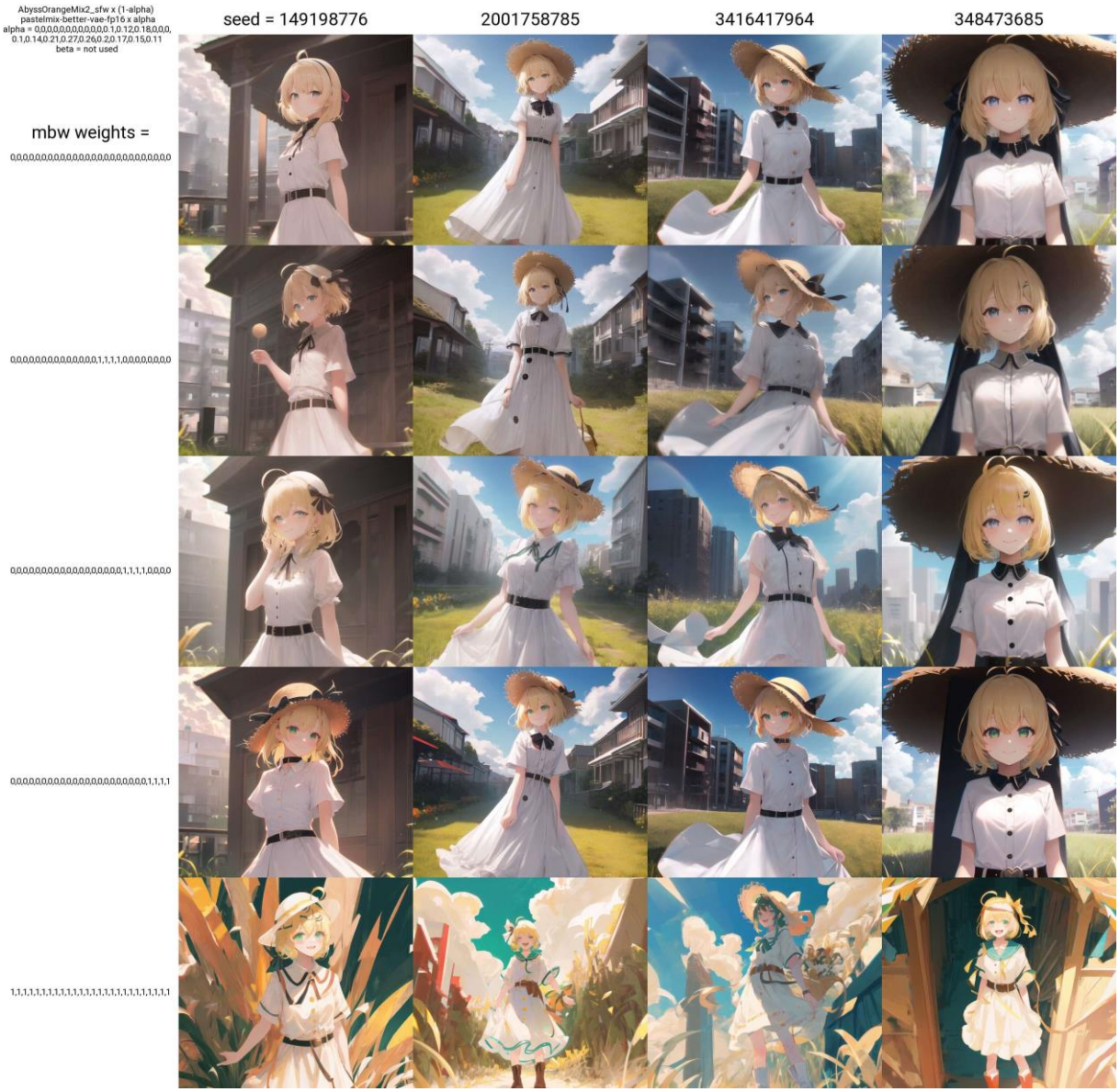
深灰色的部分就是中层的平面构成所管理的（IN04-07）

蓝色的部分就是细节化的平面构成所管理的（IN00-03）

像是决定一个图案的大体形状的工作会在高层完成，而大体形状上添加细节则是由低层完成的。

总结：**一个图形在画面中的占比越大，负责控制它的层就越高，反之就越低。**

1-6.OUT 区块


AbyssOrangeMix2_sfw x (1-alpha)
pastelmix-better-vae-fp16 x alpha
alpha = 0,0,0,0,0,0,0,0,0,0,0,0,1,0,12,0,18,0,0,0,
0,1,0,14,0,21,0,27,0,26,0,2,0,17,0,15,0,11
beta = not used
seed = 149198776
2001758785
3416417964
348473685
mbw weights =
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1

AbyssOrangeMix2_sfw x (1-alpha)
pastelmix-better-vae-fp16 x alpha
alpha = 0,0,0,0,0,0,0,0,0,0,0.1,0.12,0.18,0,0,0,
0.1,0.14,0.21,0.27,0.26,0.2,0.17,0.15,0.11
beta = not used

mbw weights =

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1

1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1

seed = 966799708	1814953204	3499636517	2186567929

以上图片中每一层的模型分别为：

1. 橘子
2. 橘子融蜡笔 out00-03（权重为 1）
3. 橘子融蜡笔 out04-07（权重为 1）
4. 橘子融蜡笔 out08-11（权重为 1）
5. 蜡笔

OUT 层相对比 IN 层好理解很多，基本就是：

上层（11 往下）负责整体调色，

中层（5、6 往两边扩散）定义画面主体（比如人物的脸型、衣服、头发）的画风，

下层（00 往上）负责笔触细节刻画等。

可以根据这个结论，去对照上面的实验图，理解 OUT 层的作用。

然后我们分析下三张实验图，首先除了个别特例，大部分的构图都很稳定。

out00-03 影响的部分可以看初音这张，

双马尾的发丝更多了，动态更强了一些，以及有一部分脸型微调，右边这张的打光也发生了变化。外套的颜色没有发生太大的变化，左侧的手部姿势有一些改变，衣服的褶皱画法有变化。**可以认为这个部分是对于一些细节的微调**。

而 OUT 中层（04-07）画面主体（人物）的画风直接变得完全不同了。可以认为，**主体（人物）的大部分画风**就来源于这四层。

out08-11 则观察不到什么细节变化（右边的头发打光差距有点大，但是左边又没有，暂时认为是随机扰动），画风也比较稳定（感觉脸部画风有一些变化），但是可以发现衣服和头发的明暗以及色相有所偏移，下面表现的更紫了，以及橘子的传统艺能：虚化，也观察不到了。

一般来说，不同模型之间，OUT 中层（04-07）差异比较大，融合后的变化也比较剧烈。

2.简单规律

在给出一些案例之前,我们会先给出一些初步的调节规律,方便进行微调和自行调整.

注：再这里及后面的系列例子中，我们讨论的是关于 U-net 不同层与最终结果图的关系，与生成的过程无关。我们将在以后，从过程的角度来分析，就目前而言，从结果分析的结论就足够使用了。

对于 U-net 来说,不同的区域干不同的事,它有着明确的分工。**IN 区块负责平面构成的相关工作，而 OUT 区块负责色彩构成的相关工作。**

在 IN 区中编号越高，对平面构成的影响就越偏向大体。

在 OUT 区中，编号越高，对细化过程的影响就越局域化，对上色过程的影响就越大体化。

以下是不同层对应东西的列举：

注意！U-net 的不同层不只对应这些例子！也有可能不对应这些例子！举这些例子仅为方便理解！

以下例子仅在人物作为画面主体，占据大部分画面时有效果：

IN 区：

0-1 对细节进行收集和分析（影响小发梢，小笔触等）

2-3 总结细节为小的局部（手，眼，口，耳，小发片，等等）

4-5 总结小的局部为大的局部（四肢，头，大发片，大物件，躯干等等）

6-7 总结局部为主体（即组装肢体为人，或者组装为 tag 指定的其他主体等等）

8-9 决定主体位置和其他大方向的调整（人的动作，位置，姿势，角度等等）

10-11 决定了整幅图的构图（视觉引导，等等）

M 区：

00 利用 IN 区总结的信息决定要画的东西（影响未知）

OUT 区：

0-1 对大型进行初步的绘制（细化：？？？

上色：画面笔触的色彩倾向，明暗倾向等）

2-3 对大型进行更进一步的绘制（细化：？？？

上色：画面细节的色彩倾向，明暗倾向等）

4-5 对主体进行绘制（细化：绘制主体的动作，姿势，角度，等

上色：画面小局部的色彩倾向，明暗倾向等）

6-7 对主体的细节进行绘制（细化：四肢，头，大发片，躯干等

上色：画面局部的色彩倾向，明暗倾向等）

8-9 进行细小物体绘制和画面主体的色彩（细化：手，脚眼，口，耳，小发片，小物体等，

上色：画面主体的色彩倾向，明暗倾向等）

10-11 进行细节的绘制和整体的色彩（细化：笔触，小发梢等，

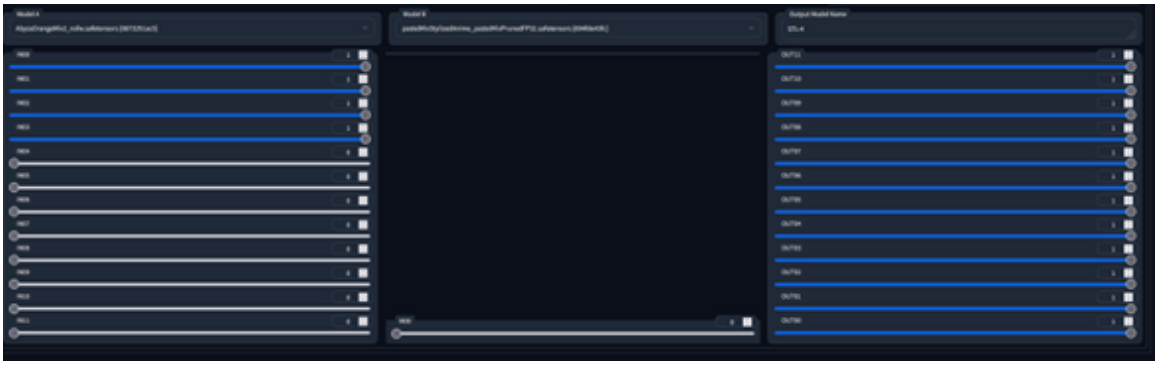
上色：画面整体色彩倾向，明暗倾向等）

然后要明确说明的是：**没有哪一层是完全具象的代表某个物体**，没有明确代表手部或脸部的层。分层只是表达：对于整个画布的大小而言，大区域、主体区域和小区域的处理方法。

3.案例分析

3-1-1.画风更换（橘子蜡笔）

A 模型为需要更换画风的模型，B 模型为想要的画风的模型,Base_alpha 为 0



这里使用深渊橘 2（AOM2）和蜡笔模型（PM）进行演示，

第 1 层是 AOM2_nsfw 原模

第 2 层是使用上图所示的参数去融合

第 3 层是传统的融合，参数为 0.5

第 4 层是 PM 原模

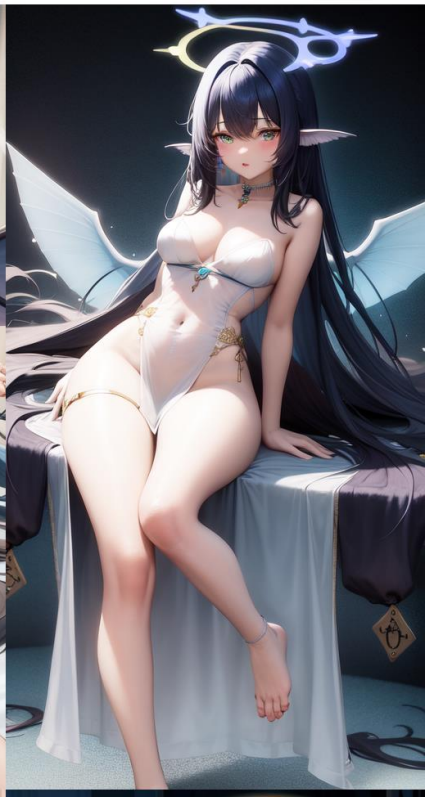
AbyssOrangeMix2_nsfw x (1-alpha)
pastelMixStylizedAnime_pastelMixPrunedFP32 x alpha
alpha = 0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5
beta = not used

seed = 1

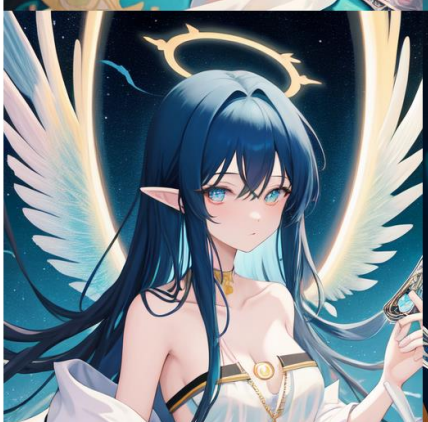
2

mbw weights =

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0



0,1,1,1,1,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1



可以明显基于 2 和 3 层进行对比可以看出 MBW 在风格迁移中效果明显优于传统方式，它可以在不对构图做较大改动的情况下完成风格迁移，并且比起传统融合，能更好的迁移 B 模型画风，基本保持了蜡笔的风格。而反观传统融合，弱化了蜡笔的风格。

（备注：如果只是画风迁移，IN 层也可以为 0 或适当减弱）

以此例举一反三，如果把 OUT 全层改成 0.5，也可实现类似传统融合的，两种画风融合在一起的效果，但是和传统融合的区别是不会改变 IN 层的成分。

3-2.人体修复（BasilMix 和 novel）

A 为需要修复的模型，B 人体好的模型,base_alpha 为 0

可以看到，分层融合成功的讲 BasliMix 在人体方面的优点加入了 novel 模型提高了 novel 模型的出图效果。

3-3.修复过拟合模型（兔头的桃子提供）

noco_6000 x (1-alpha)
□□□□ x alpha
alpha = 1,1,1,1,1,1,1,1,1,1,1,1,1,0.25,0.25,0.25
,0.25,0.25,0.25,0,0,0,0,0,0
beta = not used

seed = 125338682

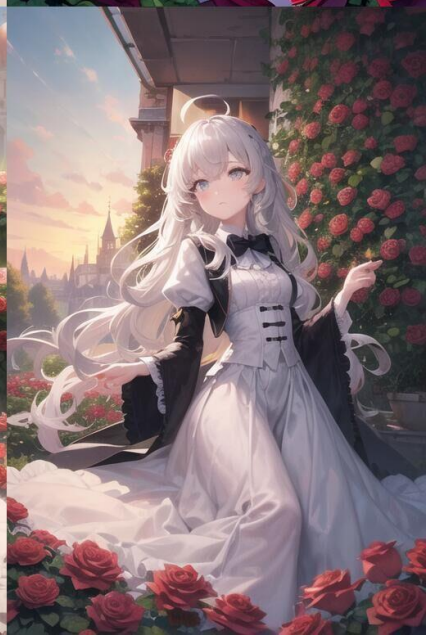
567961329

mbw weights =

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

1,1,1,1,1,1,1,1,1,1,1,1,1,0.25,0.25,0.25,0.25,0.25,0,0,0,0,0

1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1



第一行 A 模型
第二行参数：base alpha=1，in 全层=1，M 层=1，out00-05=0.25
第三行 B 模型
其实就是抄 B 模型构图，这个做法其实很像第一个画风更换的例子，只不过是 AB 模型对调了。

此例中，OUT 层可以选择不融，这里是因为原模型细节刻画不足，选择融少部分 OUT 层。

以此例举一反三，如果把 IN 层的参数降低，也能保留原来模型的部分 IN 层结构，同时改善构图，所以不一定抄文档参数的全 1。

3.实践案例

（施工中 ing）

4.加入讨论

如果想加入文档的编辑工作，或者讨论分层融合技术，**请加 Q 群：437595725**

附录 1：其他作者的资料

与我们类似的英文文档：
https://rentry.org/BlockMergeExplained

与我们类似的日文文档：
https://economylife.net/u-net-marge-webui1111

插件作者的两篇实验：
https://note.com/bbcmc/n/n2d4765b4bd47
https://note.com/bbcmc/n/n886576e5e770

其他一些作者：
https://note.com/fei_san/n/n0ac2fa12f0d2
Stable Diffusion UNET 结构 https://zhuanlan.zhihu.com/p/582266032
B 站 UP 主飞鸟白菜：https://www.bilibili.com/video/BV1RT411D7h7

【AI 绘画】大魔导书：AI 是如何绘画的？Stable Diffusion 原理全解（一）

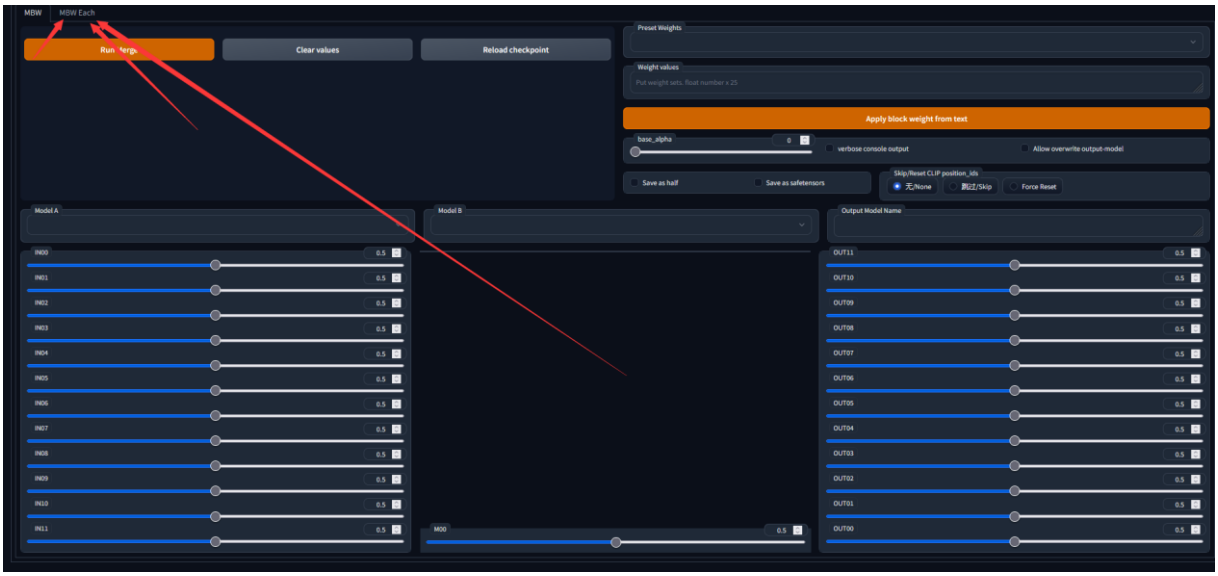
https://www.bilibili.com/read/cv21564981

应该是 MBW 始祖（）：

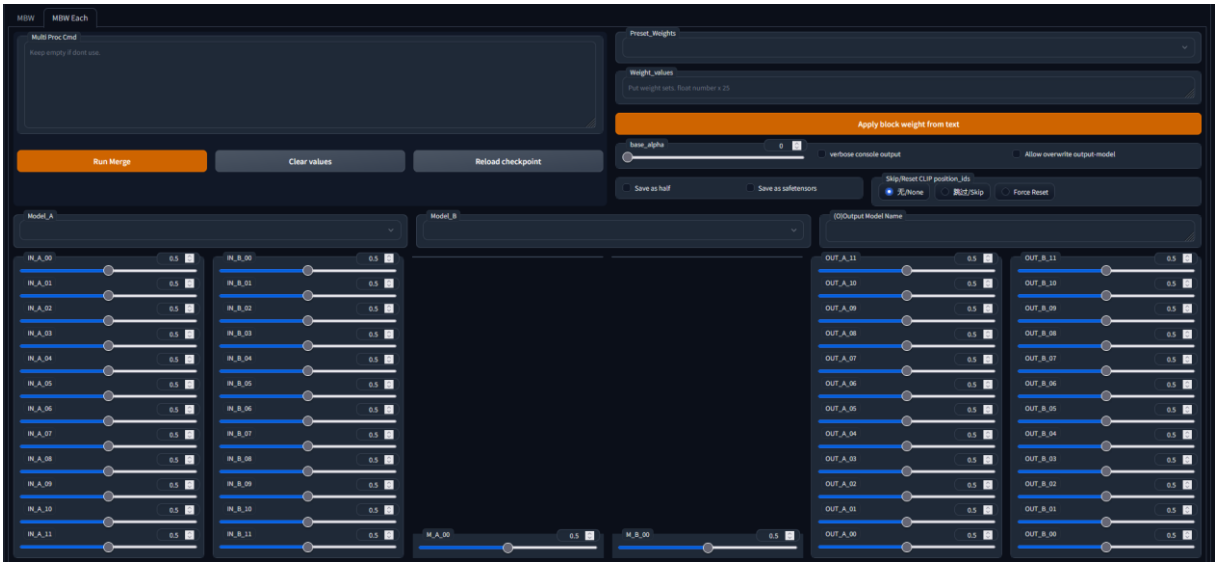
https://note.com/kohya_ss/n/n9a485a066d5b

附录 2：MBW Each（简称 MBWE）

1.如何打开 MBWE：



点击图中按钮即可打开，打开后如下：



2.功能介绍（此处默认你已经看过进阶篇）：

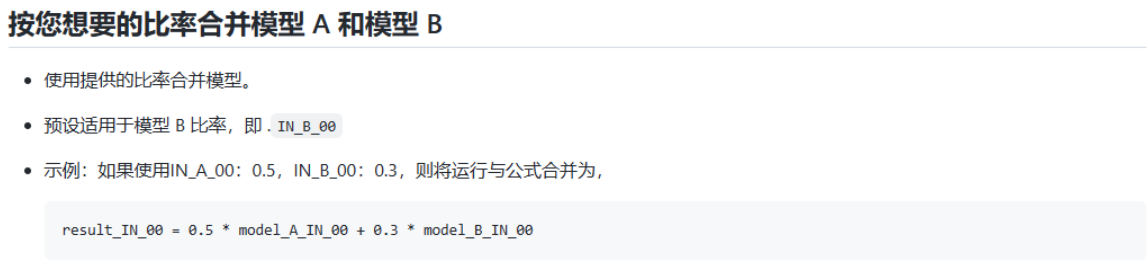
Multi Proc Cmd ：全自动融合命令输入框

其他不变。

50 个滑条：
AB 两滑条的融合方式如下，例如 IN00 层：

（IN00-A 滑条的值*A 模型的 IN00 层）+（IN00-B 滑条的值*B 模型 00 层），以此类推。

此处放上插件源文档的介绍和机器翻译：



附录 3：SuperMerger（施工中.....）

附录 4：一些有趣的实验

实验 1：替换构图中的主体

图 1 为橘子 图 2 为蜡笔 图 3 为蜡笔融橘子的 in4-7 层

分析结果：in4-7 可能是控制画面主体的构图

实验 2：另一种色彩画风转移的参数

IN00	1			OUT11	1
IN01	1			OUT10	1
IN02	1			OUT09	1
IN03	1			OUT08	1
IN04	0.75			OUT07	1
IN05	0.5			OUT06	1
IN06	0.33			OUT05	0.75
IN07	0.2			OUT04	0.6
IN08	0.1			OUT03	0.5
IN09	0			OUT02	0.33
IN10	0			OUT01	0.15
IN11	0	M00	0	OUT00	0.15

图 12 分别为 AB 模型同种子同参数对比图 图 4 为图 3 参数的融合结果

分析结果：不太好分析因为影响的地方有点多了（）不过这个参数的本意是只想融合蜡笔的大部分色彩和小部分笔触，仅供读者参考

实验 3：真人替换二次元脸型

分析：这个测试想探究真人风格和二次元风格所在的层，操作为：把橘子的全部或部分 OUT 层迁移到 BASIL 真人模型。可以观察到：

OUT 全层：脸型和眼睛比起橘子几乎不变，眼睛颜色有变化，不排除是 IN 层的影响。

OUT0-3：看不出太具体的变化，只能认为对脸部有一定影响。（猜测是脸部大小在画面占比上贴近橘子）

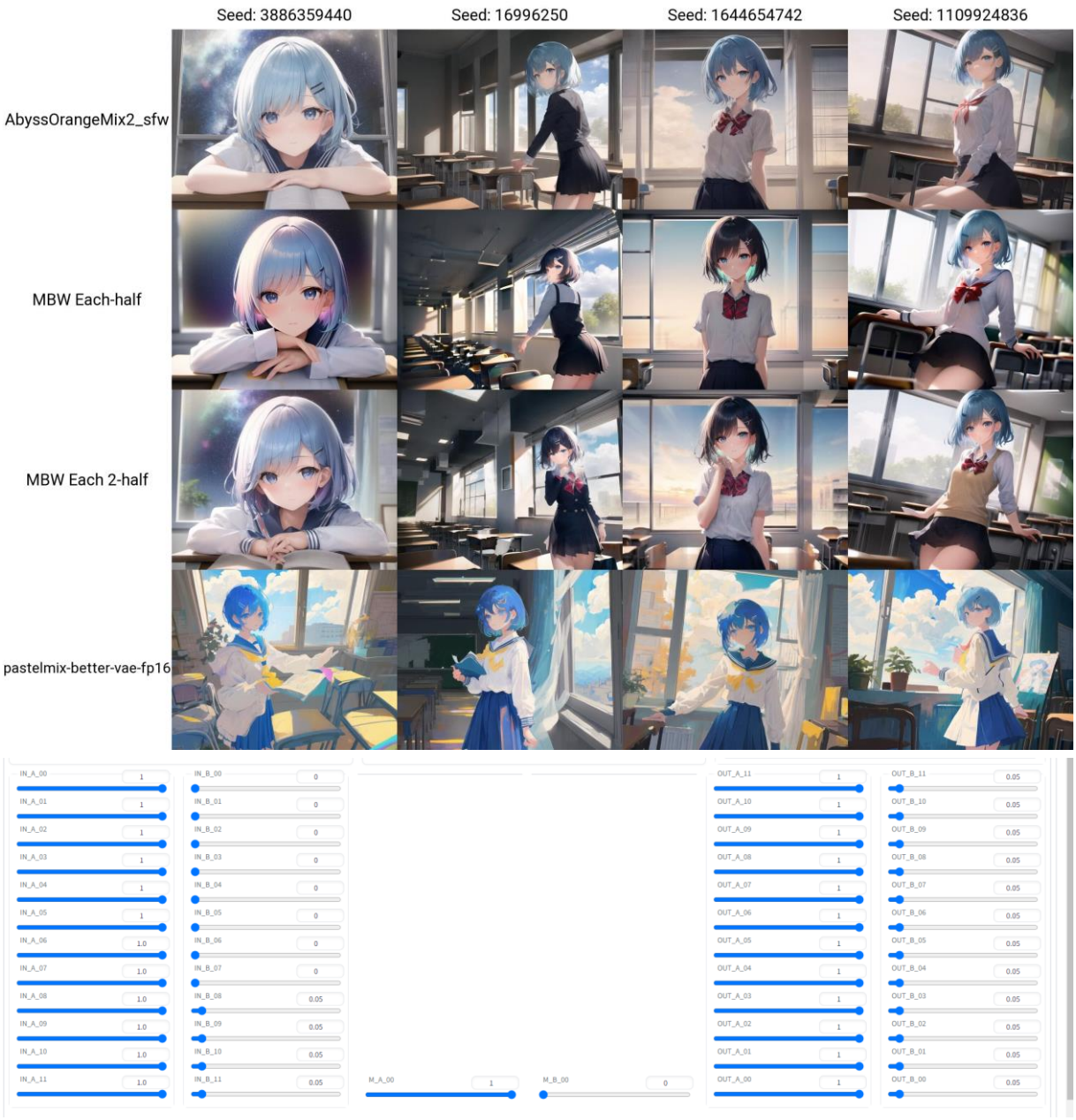
OUT4-6：明显贴近二次元脸型，但是和全 OUT 层差异极大。

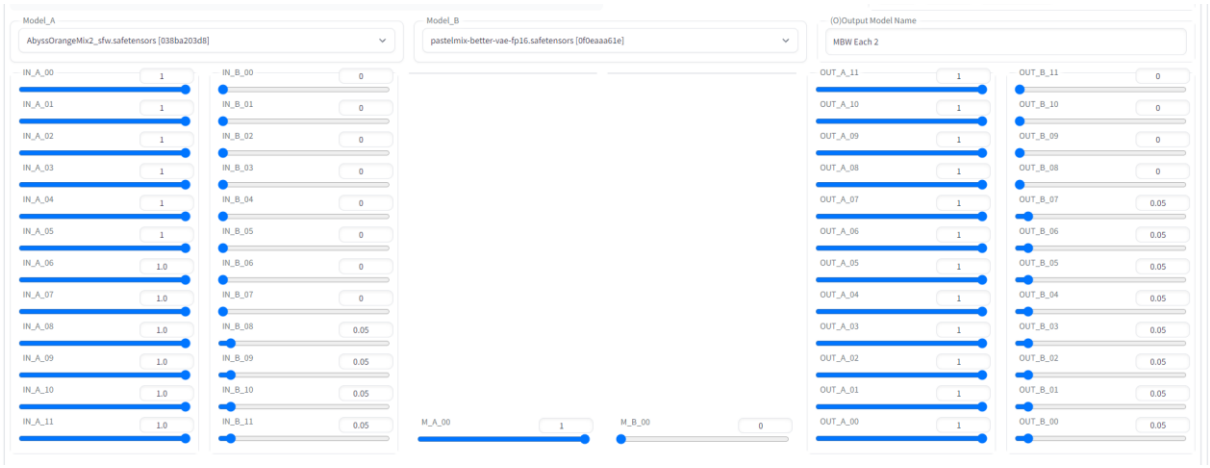
OUT0-6：和全 OUT 层差异较小，接近橘子画风。（同时图三观察到皮肤光影变得写实了，也许皮肤的质感来自于 OUT07-11？）

OUT7-9：在特写镜头里，眼睛似乎变大了一点（见图四）。但是在全身镜头似乎没变化，好像也大了点？（见图三）

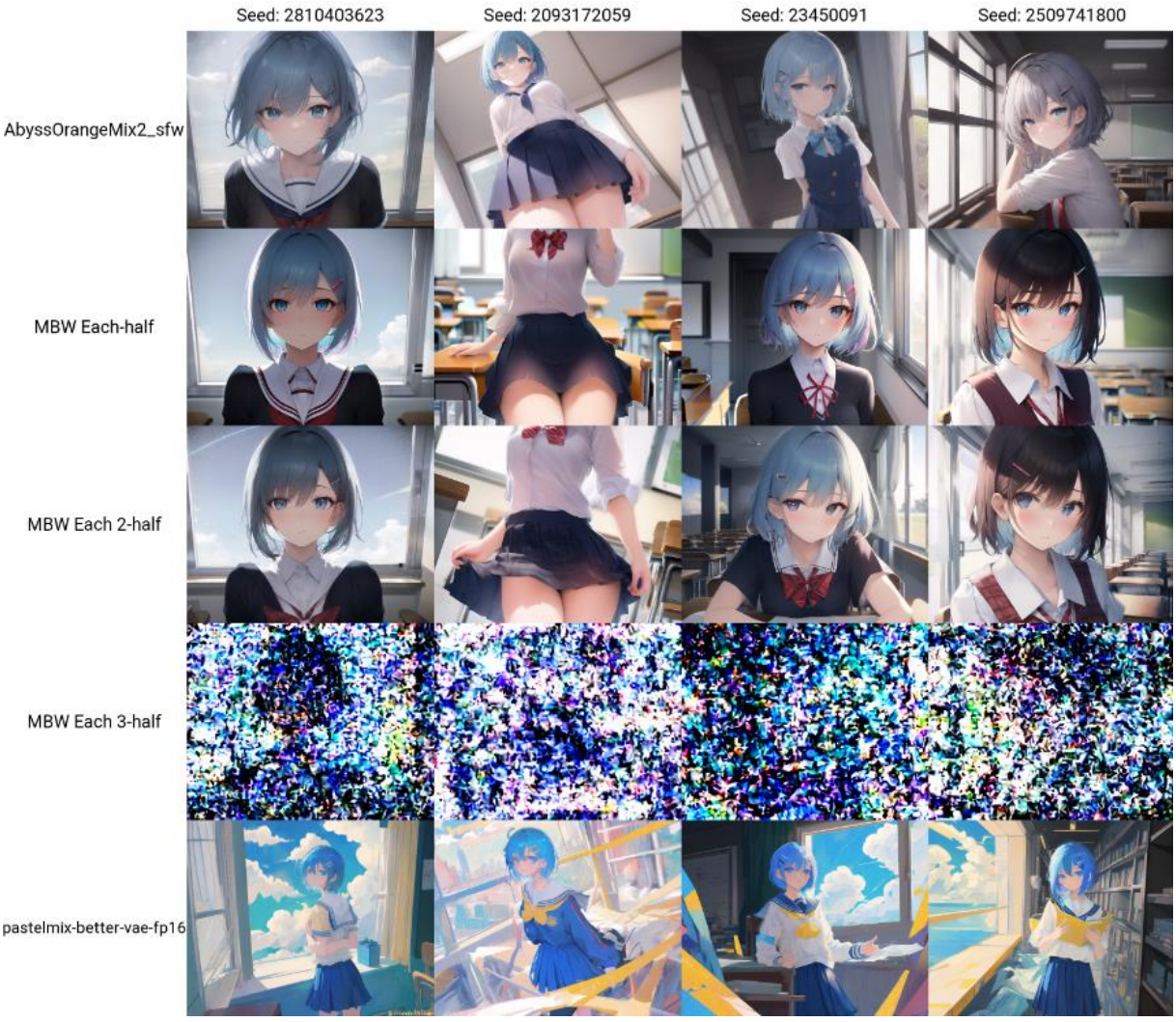
根据以上分析得出大概结论，脸部画风大部分存储在 4-6，小部分在 0-3，7-9 几乎忽略不计，更精细的结果需要进一步测试。

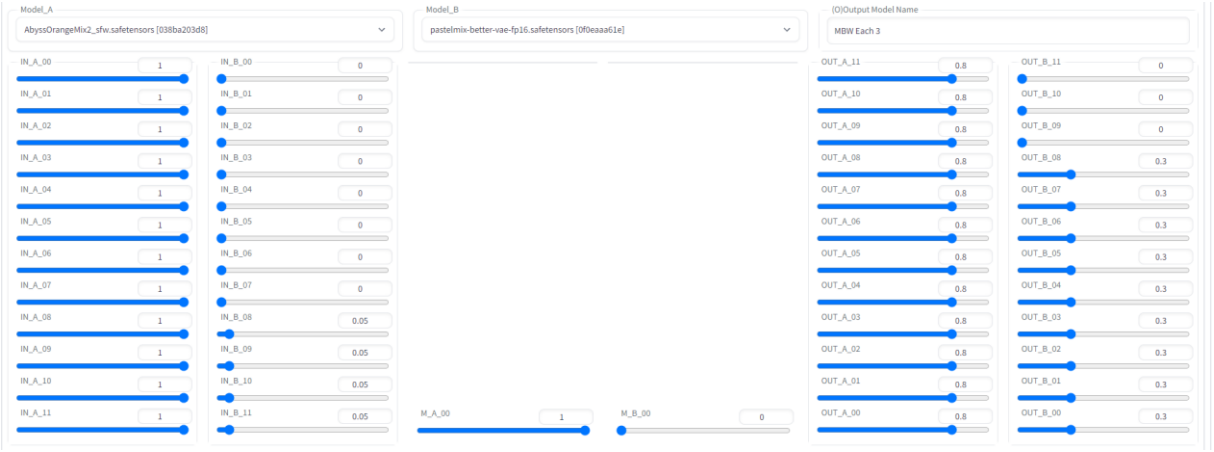
实验 4：尝试 MBW EACH




Model_A
AbyssOrangeMix2_sfw.safetensors [038ba203d8]
Model_B
pastelmix-better-vae-fp16.safetensors [0f0eaaa61e]
(O)Output Model Name
MBW Each 2

分析：完全没搞懂


Seed: 2810403623
Seed: 2093172059
Seed: 23450091
Seed: 2509741800
AbyssOrangeMix2_sfw
MBW Each-half
MBW Each 2-half
MBW Each 3-half
pastelmix-better-vae-fp16



更是寄中寄，目前只发现这一种用法就是毁模型（）

实验 5：来自 ALBEDO 的测试

OUT08 B模型Basil，灰阶（发灰）、色彩，具体细节未测出
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0

OUT09 B模型Basil，轻微的灰阶（发灰）、轻微眼睛（不确定）
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0

OUT10 B模型Basil，轻微灰阶（发灰），效果接近泛光
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0

OUT11 B模型Basil，灰阶（发灰），AOM2式的泛光效果，细节影响未测出
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1

OUT09-11 B模型***，部分衣服纹理，灰阶正常。
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.6,1,1