1. 环境配置与测试
   1. Jittor安装错误：提示MSVC编译器路径问题，报告"C:\Users\LINYIN\.cache\jittor\msvc\VC\\_\\_\\_\\_\\_\bin\cl.exe"调用失败

故障原因：[Windows support · Issue #631 · Jittor/jittor](https://github.com/Jittor/jittor/issues/631)兼容性问题

方案1:降级至1.3.6.6版本(失效)

方案2:安装1.3.9.14运行test\_core，再降级到1.3.6.6运行test\_core(失效)

方案3:使用WSL(未试验)

方案4:在Linux下运行(可行，选用)

* 1. Cutlass.zip MD5失配，Jittor服务器提供文件有问题或已损坏

RuntimeError: MD5 mismatch between the server and the downloaded file /home/liuwenyu/.cache/jittor/cutlass/cutlass.zip

方案1:Github有大佬提供了正确的Cutlass.zip([share\_repo/cutlass.zip at master · yongqianxiao/share\_repo](https://github.com/yongqianxiao/share_repo/blob/master/cutlass.zip))，直接下载并放在.cache/jittor/cutlass下即可

* 1. Jittor框架中MNIST数据集下载链接失效

Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz to ./mnist\_data/train-images-idx3-ubyte.gz 0.00B [00:00, ?B/s]HTTP Error 404: Not Found Download File failed, url: http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz, path: ./mnist\_data/train-images-idx3-ubyte.gz

原数据集链接不可用

方案1:将Pytorch环境下下载的MNIST文件拷贝，并设置Jittor框架中Download=False。

* 1. 使用Numpy()获取数据时滞后

在使用MNIST数据集进行训练和测试时，发现正确率接近随机猜测。

检查数据输入发现Label在第一个Batch时为乱码，第二个Batch提供第一个Batch图片的label。

方案1:重装Jittor环境，问题解决

推测故障原因是在Jittor环境中安装Pytorch后，Pytorch依赖numpy版本覆盖了Jittor依赖的numpy，而Jittor使用numpy()来实现数据同步显示，因此出现问题。

1. HiSD复现与训练
   1. 出现论文未提及的鉴别器架构

新增了ALI对抗学习推理(声称)，实际上是允许鉴别器同时使用图像，该图像的Style code以及属性无关标签，三者同时进行推断。这有助于鉴别器验证图像实际表现与Style Code描述是否相符，一定程度上补全了鉴别器分析图像风格的能力，同时引导提取器提取出具有稳定表现力的风格代码(Gen\_Loss\_Adv\_Real)

* 1. AdaIN参数设置问题

出现提示：A Jittor Var is a Parameter when it is a member of Module if you don't want a Jittor Var member, be treated as a Parameter, just name it starts with underscore '\_’. The `Parameter` interface isn't needed in Jittor, this interface does nothing and is just for compatibility.

Jittor会将模型中定义的变量自动识别为参数，在 Jittor 中，只要一个 jt.Var 是 Module 的成员，就会被当作参数。不需要使用nn.Parameter显式的声明。AdaIN中权重与偏置是由其他部分传入，而非模型参数，因此应该使用\_weight和\_bias显式指明其非模型参数。

* 1. Yaml配置文件字符串问题

Yaml中不使用引号默认识别为字符串，但不能包括特殊符号和空格，使用单引号时不解析转义字符，使用双引号时解析转义字符。

* 1. 推理中梯度警告

[w 0721 22:30:41.996185 96 grad.cc:81] grads[233] 'mappers.1.post\_models.1.0.linear.weight' doesn't have gradient. It will be set to zero: Var(13057:1:1:1:i0:o0:s1:n0:g1,float32,mappers.1.post\_models.1.0.linear.weight,7f79cbbc0000)[256,256,] [w 0721 22:30:41.996191 96 grad.cc:81] grads[234] 'mappers.1.post\_models.1.0.linear.bias' doesn't have gradient. It will be set to zero: Var(13068:1:1:1:i0:o0:s1:n1:g1,float32,mappers.1.post\_models.1.0.linear.bias,7f79cb5fec00)[256,]……

代表某个参数在当前训练中没有参与计算，因此不会被纳入反向传播，为了安全Jittor将其梯度设置为0。

这部分警告在鉴别器训练时被触发。因为鉴别器使用生成器生成的图像和风格代码进行训练，但鉴别器不应当获得来自生成器的梯度信息，因此鉴别器获得的数据均被detach处理，这符合项目逻辑。

* 1. 内存泄露

训练过程中产生内存泄露，经过测试后定位在EMA模块，不执行EMA更新则不会引发这一问题。

1. **def** update\_average(model\_tgt, model\_src, beta=0.99):
2. #每次只更新1%
3. with jt.no\_grad():
4. param\_dict\_src = dict(model\_src.named\_parameters())
5. **for** p\_name, p\_tgt **in** model\_tgt.named\_parameters():
6. p\_src = param\_dict\_src[p\_name]
7. **assert** (p\_src **is** **not** p\_tgt)
8. p\_tgt.assign(beta \* p\_tgt + (1 - beta) \* p\_src)

使用jt.display\_memory\_info()每10个Batch检查一次内存信息，发现内存泄露。同时，Hold var不变，每次Lived Var和Lived Ops增加。

图片包含 日历

AI 生成的内容可能不正确。

图片包含 文本

AI 生成的内容可能不正确。

问题1:权重转移时应同时关闭model\_tgt和model\_src对梯度的记录，处理后内存泄露略微减小。

问题2:使用jt.no\_grad()保证其下属所有新建的变量不再计算梯度，但先前创建的变量仍然可能保留梯度。

方案1:计算过程中为每个变量都使用.numpy()处理，强制性保证其不携带任何梯度信息。(生效，但训练速度严重下降，GPU与CPU通讯频繁)

方案2:积极使用jt.clean()清理计算图，每次进行EMA后均清理计算图，无效，每次Lived Var和Ops仍然增加。

推理可知，计算图中被依赖的Var和Ops不断增加，计算图规模不断扩大，但梯度计算已经阻断，因此计算图扩大与梯度无关。

考虑到使用numpy处理会解决此问题，numpy()只携带数据，不携带任何梯度**和**计算图，numpy()同时也是Jittor框架进行同步计算的操作，因此考虑异步计算问题。

结论：Jittor使用惰性机制，所有计算均只创建计算图而非立刻执行，只有当计算结果被需要时才会执行并清除计算图。这代表如果某个变量始终不被需要，则该变量相关的计算图始终不会被清除。EMA每次更新model\_tgt,而tgt只在采样和保存时被使用，设置每1000batch采样一次，则计算图会堆积导致内存泄露。(这解释了为什么初期调试时设置10batch采样一次未发现该问题)。

方案3:每个变量计算过后立刻使用Sync强制同步(生效，但是慢)

方案4:model\_tgt所有变量的计算图构建完成后，对model\_tgt的所有参数调用sync，同时结合data\_iters.preload(),在GPU进行Sync的同时进行数据预读。(生效，速度显著变快)。

1. HiSD测试与评估
   1. 考虑引入PPL感知路径长度以更好的评估HiSD在风格解缠上的性能

但PPL只能在Latent-guide模式下使用。

Reference-guide提取出的风格代码在样本空间中分布不一定均匀，因此进行短距离的插值也有可能插值到样本分布稀疏的区域。而如果直接对比来自两张reference images的style code以及其生成的图像，这二者的距离可能很远，无法测量两张图像在style code微小变化后的变化。