0.0 环境配置与测试

0.1 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 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)

0.2 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)，直接下载并放在.cache/jittor/cutlass 下即可

0.3 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。

0.4 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。

1.0 HiSD 复现与训练

1.1 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)

1.2 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。

1.3 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。

1.4 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。

1.5 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%
```

```
    with jt.no_grad():
        param_dict_src = dict(model_src.named_parameters())
        for p_name, p_tgt in model_tgt.named_parameters():
            p_src = param_dict_src[p_name]
            assert (p_src is not p_tgt)
            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。

```
=== display_memory_info ===
 total_cpu_ram: 251.5GB total_device_ram: 31.74GB
 hold_vars: 1527 lived_vars: 20177 lived_ops: 17271
 name: sfrl is_device: 1 used:  4.49GB(51.5%) unused:  4.23GB(48.5%) ULB:    72MB ULBO:    72MB total:  8.72GB
 name: sfrl is_device: 1 used:     0 B(-nan%) unused:     0 B(-nan%) total:     0 B
 name: sfrl is_device: 0 used:     0 B(-nan%) unused:     0 B(-nan%) total:     0 B
 name: sfrl is_device: 0 used: 170.3MB(13.9%) unused: 1.032GB(86.1%) ULB:    72MB ULBO:    72MB total: 1.198GB
 name: sfrl is_device: 0 used:     0 B(-nan%) unused:     0 B(-nan%) total:     0 B
 name: temp is_device: 0 used:     0 B(-nan%) unused:     0 B(-nan%) total:     0 B
 name: temp is_device: 1 used:     0 B(0%) unused:  4.73GB(100%) total:  4.73GB
 cpu&gpu: 14.65GB gpu: 13.45GB cpu: 1.198GB
 free: cpu(18.13GB) gpu(17.96GB)
 swap: total(    0 B) last(    0 B)
```

```
=== display_memory_info ===
 total_cpu_ram: 251.5GB total_device_ram: 31.74GB
 hold_vars: 1527 lived_vars: 24473 lived_ops: 21030
 name: sfrl is_device: 1 used: 6.109GB(58.4%) unused: 4.345GB(41.6%) ULB:    72MB ULBO:    72MB total: 10.45GB
 name: sfrl is_device: 1 used:     0 B(-nan%) unused:     0 B(-nan%) total:     0 B
 name: sfrl is_device: 0 used:     0 B(-nan%) unused:     0 B(-nan%) total:     0 B
 name: sfrl is_device: 0 used: 170.8MB(13.9%) unused: 1.031GB(86.1%) ULB:    72MB ULBO:    72MB total: 1.198GB
 name: sfrl is_device: 0 used:     0 B(-nan%) unused:     0 B(-nan%) total:     0 B
 name: temp is_device: 0 used:     0 B(-nan%) unused:     0 B(-nan%) total:     0 B
 name: temp is_device: 1 used:     0 B(0%) unused:  4.73GB(100%) total:  4.73GB
 cpu&gpu: 16.38GB gpu: 15.18GB cpu: 1.198GB
 free: cpu(18.07GB) gpu(16.23GB)
 swap: total(    0 B) last(    0 B)
```

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)。

2.0 HiSD 测试与评估

2.1 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。