计算机图形学第二次大作业实验报告

本文参考https://github.com/rosinality/style-based-gan-pytorch进行实现。

1. **实验目的**
   1. 学习StyleGAN，了解并学习其网络设计思想，并参照源码进行网络的搭建，验证结果。
   2. 学习jittor深度学习框架并jittor进行科研开发。
   3. 利用jittor搭建StyleGAN并在其上利用彩色字符数据集进行训练，可以根据随机输入的向量生成新的字符，并可以进行隐向量插值。
2. **实验原理**

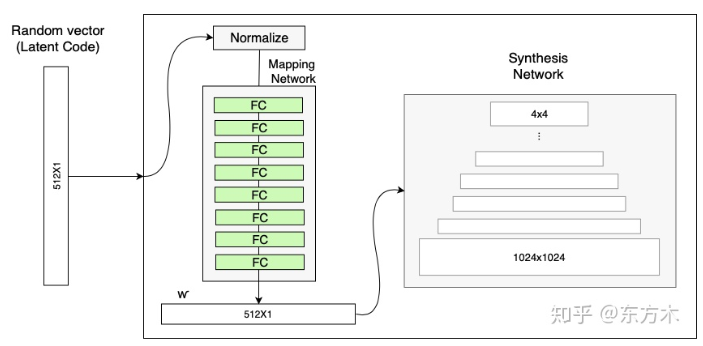
StyleGAN由NVIDIA于2019年发表在CVPR上，该论文是由同公司在2017年提出的ProGAN发展而来。ProGAN的特征也同样在StyleGAN的实现和训练中起到了至关重要的作用。

ProGAN的创新之处在于利用渐进式训练解决了高分辨率图像的生成问题，其具体做法为，从训练分辨率非常低（如4×4）的generator和discriminator开始，每次增加一个更高分辨率的网络层。

而与多数GAN一样，ProGAN在控制生成图像的特定特征方面能力十分有限。而StyleGAN的创新之处便在于通过分别修改每一层级的输入，在不影响其它层级的情况下控制该层级所表达的图像特征。下面具体介绍其主要创新：

1. 映射网络（Mapping Network）：

StyleGAN关注了ProGAN的生成器网络，发现渐进层使用得当的话可以用于控制图像的不同视觉特征。且所影响的特征会随着层分辨率的提高变得更精细。StyleGAN的第一点改进是，给Generator的输入加上了由8个全连接层组成的Mapping Network，并且Mapping Network的输出W’与输入层（512×1）的大小相同。如下图所示。

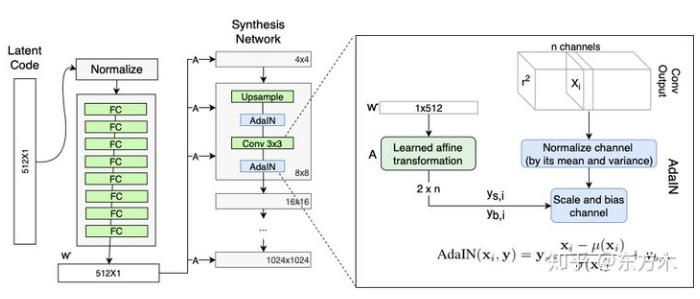


添加Mapping Network的目标是将输入向量编码为中间向量，之后将该中间向量传给生成网络，获得18个控制向量，以使得控制向量的不同元素可以控制不同的视觉特征。若不加这个Mapping Network，那么后续得到的18个控制向量之间就会存在特征纠缠的现象（比如我们想通过调节其中一个控制向量来改变人脸的肤色，但发现在另一个分辨率上的控制内容，例如人的毛发样式，也被改变了，这就叫做特征纠缠）。可以说，Mapping Network的作用就是为输入向量的特征解缠提供一条学习的通路。

1. 样式模块（AdaIN）：

StyleGAN的第二点改进是将特征解缠后的中间向量W’变换为样式控制向量，从而参与影响生成器的生成过程。

Generator从生成4×4一直不断扩大，直到变化到1024×1024大小，共由9个生成阶段。每个生成阶段都有两个控制向量（A）对其施加影响，其中一个控制向量在Upsample之后对其影响一次，另外一个控制向量在Convolution之后对其影响一次，影响的方式都采用AdaIN（自适应实例归一化）。因此，中间向量W’总共被变换成18个控制向量（A）传给生成器。



上图为样式模块（AdaIN）的具体实现，将W’通过一个可学习的仿射变换（A，实际上是一个全连接层）扩变为放缩因子y\_(s,i)与偏差因子y\_(b,i)，这两个因子会与标准化之后的卷积输出做一个加权求和，就完成了一次W’影响原始输出x\_i的过程。

1. 删除传统输入：

StyleGAN生成图像的特征是由W’和AdaIN控制的，那么生成器的初始输入可以被忽略，并用常量值替代。这样做的理由是，首先可以降低由于初始输入取值不当而生成出一些不正常的照片的概率（这在GANs中非常常见），另一个好处是它有助于减少特征纠缠，对于网络在只使用W’不依赖于纠缠输入向量的情况下更容易学习。

1. 随机变化：

人们的脸上有许多小的特征，可以看作是随机的，例如：雀斑、发髻线的准确位置、皱纹、使图像更逼真的特征以及各种增加输出的变化。将这些小特征插入GAN图像的常用方法是在输入向量中添加随机噪声。为了控制噪声仅影响图片样式上细微的变化，StyleGAN采用类似于AdaIN机制的方式添加噪声，即在AdaIN模块之前向每个通道添加一个缩放过的噪声，并稍微改变其操作的分辨率级别特征的视觉表达方式。加入噪声后的生成人脸往往更加逼真与多样。

1. 样式混合（Style Mixing）：

StyleGAN生成器在合成网络的每个级别中使用了中间向量，这有可能导致网络学习到这些级别是相关的。为了降低相关性，模型随机选择两个输入向量，并为它们生成了中间向量W’。然后，它用第一个输入向量来训练一些网络级别，然后（在一个随机点中）切换到另一个输入向量来训练其余的级别。随机的切换确保了网络不会学习并依赖于一个合成网络级别之间的相关性。

虽然它并不会提高所有数据集上的模型性能，但是它能够以一种连贯的方式来组合多个图像。该模型生成了两个图像A和B，然后通过从A中提取低级别的特征并从B中提取其余特征再组合这两个图像，这样能生成出混合了A和B的样式特征的新人脸，也就是本实验中所对应的隐向量插值。

1. **技术细节**
   1. 主要文件及用途：
      1. progressive\_data.py负责生成多分辨率的图像，用于渐进式训练。
      2. model.py 定义stylegan中generator与discriminator的结构。
      3. train.py对网络进行训练。
      4. generate.py利用训练完成的模型生成图像示例，包括随机生成和隐向量插值两种。
   2. 根据pytorch版本实现jittor版本的代码：

Jittor与pytorch的接口都十分类似，可以很方便地进行迁移和实现，不同点只在于一些函数的命名与功能的分合（例如forward、backward与execute、step，ctx参数与jt.Function中的self参数，以及weight中存储的数据不同）。因此本次实验主要根据style-based-gan-pytorch进行了迁移。

1. **实验结果**

本次实验使用超参数如下：

Phase: 200000

learning rate: 0.001

batch\_size:32

init\_size: 8

max\_size: 64

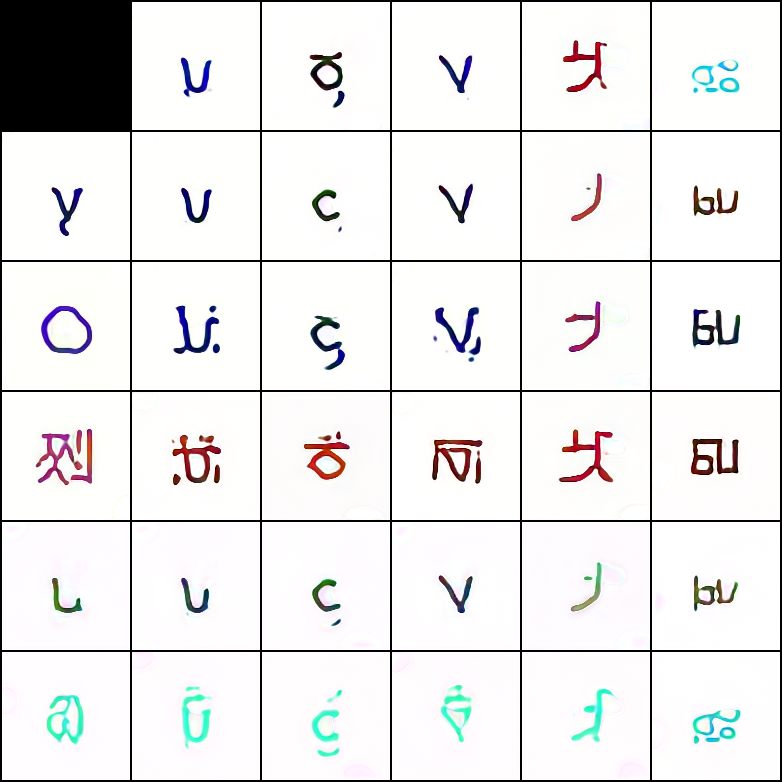
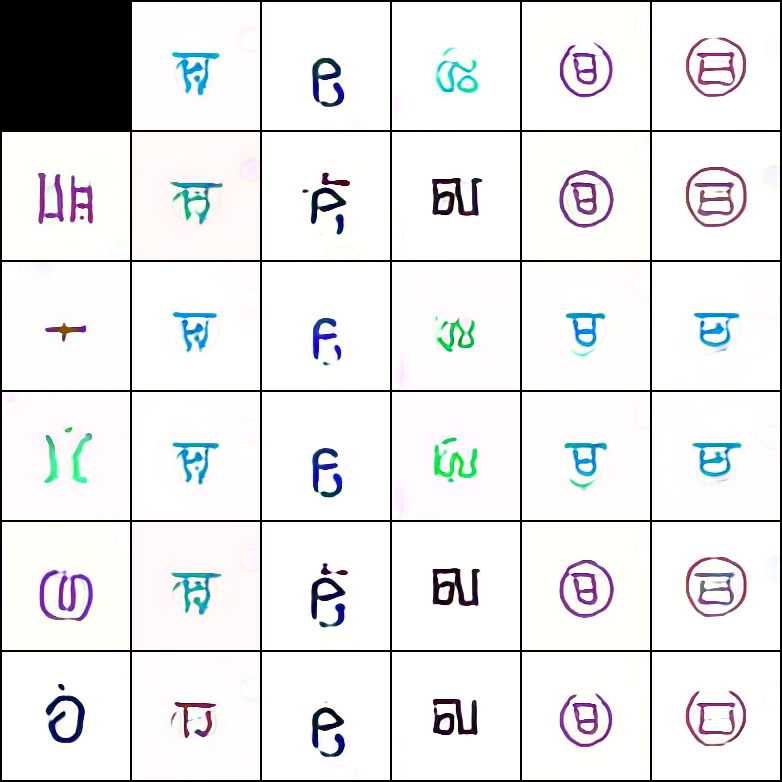
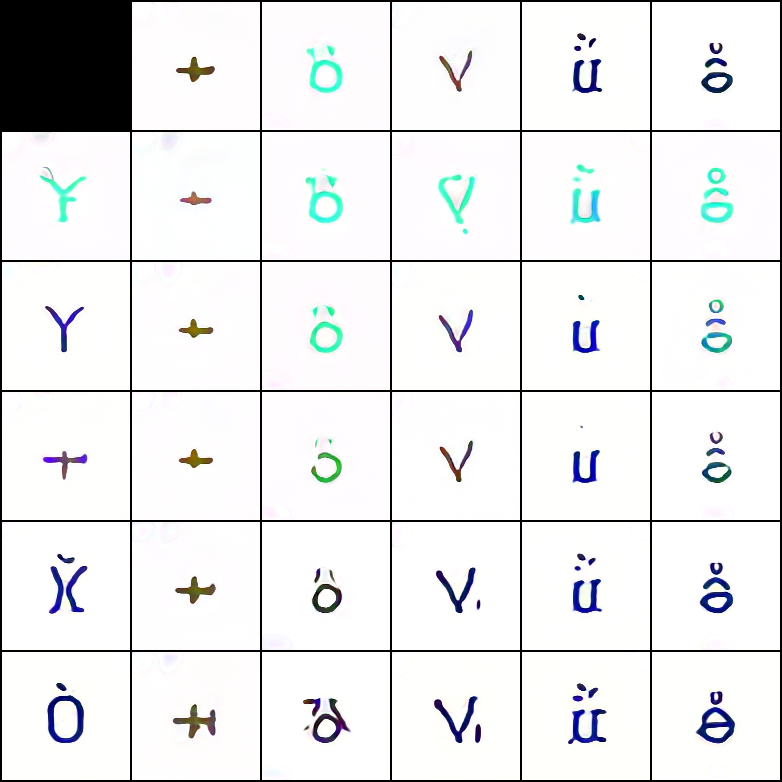
loss全程采用wgan\_gp。值得注意的是，一开始因为jittor的版本问题只有r1 loss可以使用，然而我尝试了很多超参数都无法得到不过拟合的结果。可见对于不同的数据集采用不同的loss是十分重要的。

最终的实验结果（80000代）：

随机向量sample生成结果：



隐向量插值结果：



在style mixing的设置中，我们将target向量用作前两层的style，source向量作为剩下层的style。style mixing结果因此保留了target向量对应结果的粗糙信息和source向量对应结果的精细信息。

因此可以看到（第一列为target向量对应结果，第一行为source向量对应结果），style mixing整体上基本保留了source的形状和target的纹理。

渐进式训练可以通过观察progress.mp4进行查看，可以发现在每次扩大size的初期都会初期一段背景颜色的深到浅和形状从混乱到清晰，这便是对原来的低分辨率层所代表的各种特征进行解耦，学习新的层所代表的特征。而在128×128的时候，随着训练的进程，字符也越来越清晰。

1. **结论**

通过本次实验，我大致了解了StyleGAN的思想，并且在参考Pytorch版代码和求助师兄的情况下完成了Jittor版的StyleGAN，通过尝试超参数在大约80000代时得到了较好的结果。这里非常感谢我们的助教大大，他在接到我关于jittor的问题后一个多小时就远程操作对进行了解决，效率十分之高。

其实最大的收获是，我之前也有了解stylegan的结构创新，但一直想当然地以为隐向量插值是选两个随机向量，然后再两者之间进行插值再输入网络，一直都不太理解这其中的原理。这回通过真正实现和查看了代码，发现原来是在不同的生成网络层使用不同的Style向量，以实现在保留粗略的低层Style向量的信息，保留精细的高层Style向量的信息。可见科研果然应该有求甚解的追求才行。