**Summary:**

深度学习数学这门课连续上了一段时间，该课程大部分的内容对我这种刚入门的萌新来说还很困难，毕竟课程的重心不在于一些深度学习基本公式的了解，而在于公式的来源以及一些推广，换言之，课程的重心不在于知道怎么用，而在于知道为什么这么用，这样可以对该公式进行扩展（就像上课做的那样），而不是单纯的只会使用公式。

通过该课程得学习，学到了一些GAN模型的扩展，通过GAN模型引出了一些其他的东西，例如JS散度，KL散度等概率论方面的知识，最令人感到有趣的是在课程上引入了一些鲜活的例子解释概率论的知识，例如将Bernoulli和Categorical等模型简单分解为若干球若干桶的实际模型，这样对于理解更加的方便。

在介绍GAN模型的时候也引入了一些关于softmax函数的使用，剩下的一大部分时间都在介绍softmax函数的一些理论知识及扩展，其中给我留下印象比较深的部分是在讲解softmax时，对sequence-Gan做了一些介绍，介绍的时候用到了这样一个例子，对NLP中词向量之间的相似度匹配做了这样一个简化，使用s:state和a:action这样两个状态对整个词向量的区分做了解释，如用V(s)，Q（s,a），Π（a/s）来对该模型做出解释，之前的模型都很少见到用GAN模型来做单词匹配的，提供了一些新思路。

总而言之，这门课虽然还是有很多不甚清楚的地方，但是总体来说还是有一些收获的。

**Analysis:**

最近看到了BigGAN的一些文章，虽然BigGAN以及VQ-VAE等一些模型以及在生成图像方面已经超越了BigGAN，不过从某些方面来说BigGAN是第一个将生成模型效率提升较大幅度的模型，在ImageNet的Inception Score上提升了足足100分，所以BigGAN还是有很大的参考价值的，所以使用了BigGAN作为本次报告的模型进行分析，也算是对BigGAN模型做一些研究。

**背景知识：**

首先对BigGAN的一些背景知识做一些介绍。我们可以知道，生成对抗网络由一个生成器G和一个判别器D组成，前者的作用是根据随机噪声数据生成逼真的随机样本，后者的作用是鉴别样本是真实的还是生成器生成的。在最早的一些GAN的版本中，GAN训练时的优化目标为达到如下极大值-极小值问题的纳什均衡：

https://pic1.zhimg.com/80/v2-d5fa47387debcd43db7065bd90187a40_hd.png

上面的公式也是GAN模型最原始的公式，当用于图像类任务时，G和D一般是卷积神经网络。如果不使用一些辅助性的增加稳定性技巧，训练过程非常脆弱，收敛效果差，需要精细设计的超参数和网络结构的选择以保证效果。

最近这些年的研究都趋向于修改初始的GAN算法，使得它更加的稳定，一般有两种思路解决问题，其中一种思路是修改训练时的目标函数以确保收敛。另外一种思路是通过梯度惩罚或归一化技术对D进行限定，这两种方法都是在抵抗对无界的目标函数的使用，确保对任意的G，D都能提供梯度。

另外，由于BigGAN是采用的Inception Score评分机制，所以作者还在文章中分析了Inception Score的评分机制，Inception Score使用了如下两种评判标准来检验模型的表现：成图片的质量、生成图片的多样性



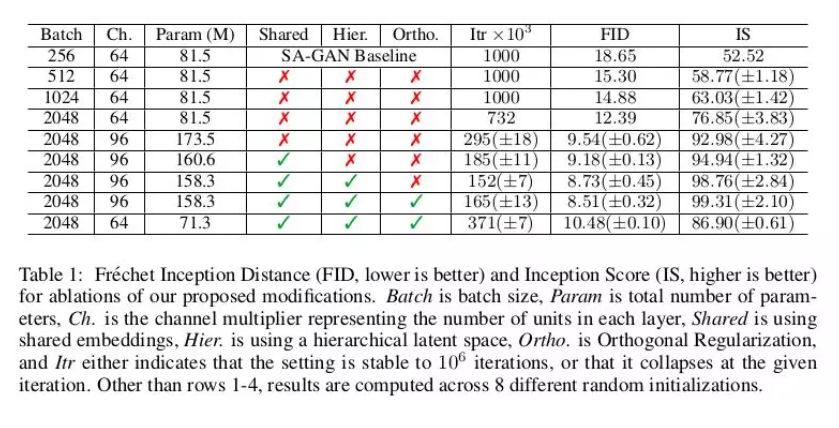
其中y为类别标签，x为样本向量，Pg为生成器生成的样本所服从的概率分布，DKL为KL散度。IS的值越大越好。p(y)是生成图片的类别y关于生成器G(z)的边缘概率分布。

BigGAN能够实现如此大的突破，其中一个很大的原因就是 BigGAN 如它题目 Large Scale GAN Training for High Fidelity Natural Image Synthesis 描述的 Large Scale，在训练中 Batch 采用了很大的 Batch，已经达到了 2048（我们平常训练 Batch 正常都是 64 居多），在卷积的通道上也是变大了，还有就是网络的参数变多了，在 2048 的 Batch 下整个网络的参数达到了接近 16 亿。

这个就是 BigGAN 之所以称为 BigGAN 的原因，我想 BigGAN 的题目不仅仅在说明网络的庞大，还想暗示这篇文章会给人带来很大的印象，确实我是被“吓”到了。 这么大的提升当然不可能是一味的增大 Batch 和网络参数能实现的，其中包括了 Batch 的加大、先验分布 z 的适时截断和处理、模型稳定性的控制等。

**Batch size的增大：**

SAGAN 中的 Batch size 为 256，作者发现简单地将 Batch size 增大就可以实现性能上较好的提升，文章做了实验验证：



可以看到，在 Batch size 增大到原来 8 倍的时候，生成性能上的 IS 提高了 46％。文章推测这可能是每批次覆盖更多模式的结果，为生成和判别两个网络提供更好的梯度。

**截断技巧：**

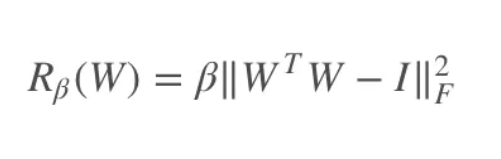
对于先验分布z，一般情况下都是选用标准正态分布 N(0,I) 或者均匀分布 U[−1,1]，文章对此存在疑惑，难道别的分布不行吗。通过实验，为了适合后续的“截断”要求，文章最终选择了 z∼N(0,I)。

所谓的“截断技巧”就是通过对从先验分布 z 采样，通过设置阈值的方式来截断 z 的采样，其中超出范围的值被重新采样以落入该范围内。这个阈值可以根据生成质量指标 IS 和 FID 决定。

通过实验可以知道通过对阈值的设定，随着阈值的下降生成的质量会越来越好，但是由于阈值的下降、采样的范围变窄，就会造成生成上取向单一化，造成生成的多样性不足的问题。往往 IS 可以反应图像的生成质量，FID 则会更假注重生成的多样性。

随着截断的阈值下降，生成的质量在提高，但是生成也趋近于单一化。所以根据实验的生成要求，权衡生成质量和生成多样性是一个抉择，往往阈值的下降会带来 IS 的一路上涨，但是 FID 会先变好后一路变差。

还有在一些较大的模型不适合截断，在嵌入截断噪声时会产生饱和伪影，如上图 (b) 所示，为了抵消这种情况，文章通过将 G 调节为平滑来强制执行截断的适应性，以便 z 的整个空间将映射到良好的输出样本。为此，文章采用正交正则化，它直接强制执行正交性条件：



其中 W 是权重矩阵，β 是超参数。这种正则化通常过于局限，文章为了放松约束，同时实现模型所需的平滑度，发现最好的版本是从正则化中删除对角项，并且旨在最小化滤波器之间的成对余弦相似性，但不限制它们的范数：



其中 1 表示一个矩阵，所有元素都设置为 1。通过上面的 Table1 中的 Hier. 代表直接截断，Ortho. 表示采用正则正交，可以看出来正则正交在性能上确实有所提升。

我认为 BigGAN 中的“截断技巧”很像 Glow [7] 中的退火技巧，BigGAN 通过控制采样的范围达到生成质量上的提高，Glow 是通过控制退火系数（也是控制采样范围）达到生成图像平滑性的保证。

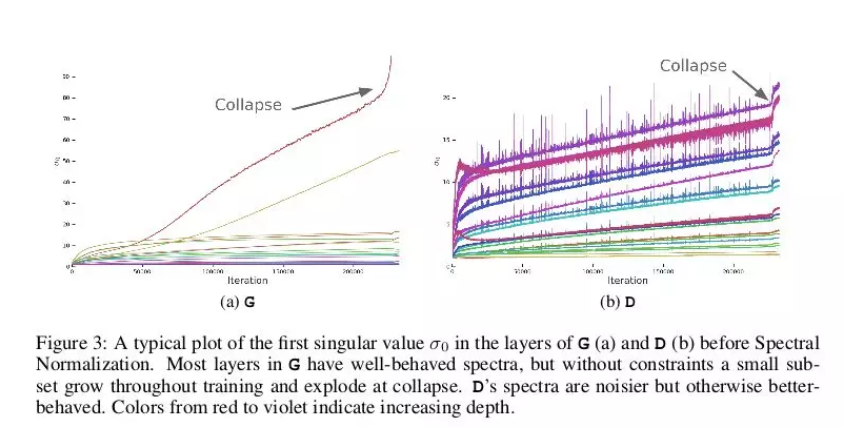
**模型稳定性的控制：**

对于GAN模型，模型的稳定性只需要通过控制G和D这两个参数就可以从一定程度上控制模型。

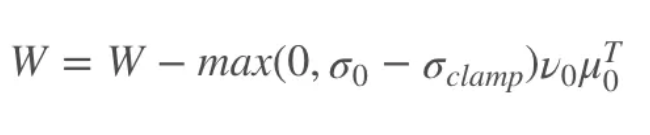
对G的控制:

在探索模型的稳定性上，文章在训练期间监测一系列权重、梯度和损失统计数据，以寻找可能预示训练崩溃开始的指标。实验发现每个权重矩阵的前三个奇异值 σ0,σ1,σ2 是最有用的，它们可以使用 Alrnoldi 迭代方法进行有效计算。

实验如下图 (a) 所示，对于奇异值 σ0，大多数 G 层具有良好的光谱规范，但有些层（通常是 G 中的第一层而非卷积）则表现不佳，光谱规范在整个训练过程中增长，在崩溃时爆炸。



为了解决 G 上的训练崩溃，通过适当调整奇异值 σ0 以抵消光谱爆炸的影响。首先，文章调整每个权重的顶部奇异值 σ0，朝向固定值σreg或者朝向第二个奇异值的比例 r，即朝向 r⋅sg(σ1)，其中 sg 是控制梯度的操作，适时停止。另外的方法是使用部分奇异值的分解来代替 σ0，在给定权重 W，它的第一个奇异值向量 μ0 和 ν0 以及固定的，将权重限制在：



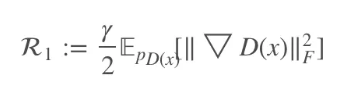
上述整个操作就是为了将权重的第一个奇异值 σ0 控制住，放置突然性的爆炸。

上述整个操作就是为了将权重的第一个奇异值 σ0 控制住，放置突然性的爆炸。

对D的控制:

和 G 的切入点相同，文章依旧是考虑 D 网络的光谱，试图寻找额外的约束来寻求稳定的训练。如上图 3 中 (b) 所示，与 G 不同，可以看到光谱是有噪声的，但是整个过程是平稳增长在崩溃时不是一下爆炸而是跳跃一下。

文章假设这些噪声是由于对抗训练优化导致的，如果这种频谱噪声与不稳定性有因果关系，那么相对采用的反制是使用梯度惩罚，通过采用 R1 零中心梯度惩罚：



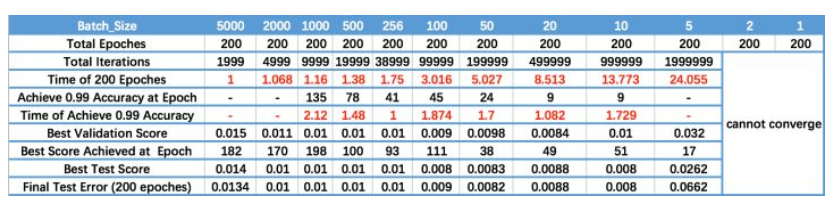
使用正交正则化，DropOut 和 L2 的各种正则思想重复该实验，揭示了这些正则化策略的都有类似行为：对 D 的惩罚足够高，可以实现训练稳定性但是性能成本很高。

如果对 D 的控制惩罚力度大，确实可以实现训练的稳定，但是在图像生成性能上也是下降的，而且降的有点多，这种权衡就是很纠结的。

可以通过约束 D 来强制执行稳定性，但这样做会导致性能上的巨大成本。使用现有技术，通过放松这种调节并允许在训练的后期阶段发生崩溃（人为把握训练实际），可以实现更好的最终性能，此时模型被充分训练以获得良好的结果。

在阅读文章的过程中，也发现BigGAN中也存在一些不合理的部分，有的属于在文章中解决了的部分，而有的则属于文章中只是提到但是并没有深入解决的问题，以下内容属于一些发现的问题，这些问题在上面的内容也提到了一些，并且有些问题做了一些解决，如：一味加深网络可能会妨碍生成的性能；共享类的思想在控制超参数上是很麻烦的，虽然可能会提高训练速度；WeightNorm 替换 G 中的 BatchNorm 并没有达到好的效果；除了频谱规范化之外，尝试将 BatchNorm 添加到 D（包括类条件和无条件），但并未取的好的效果；在 G 或 D 或两者中使用 5 或 7 而不是 3 的滤波器大小，5 的滤波器可能会有些许提升，但是计算成本也上去了；尝试在 128×128 的 G 和 D 中改变卷积滤波器的扩张，但发现在任一网络中即使少量的扩张也会降低性能；尝试用 G 中的双线性上采样代替最近领近的上采样，但这降低了性能。

这里，对第一条缺陷提出了一些解决方案，文章中提出，一味加深网络可能会妨碍生成的性能，首先讨论一下batch size对模型的影响，这里跑一个 LeNet 在 MNIST 数据集上的效果,效果如下：



运行结果如上图所示，其中绝对时间做了标幺化处理。运行结果得到分析如下：Batch\_Size 太小，算法在 200 epoches 内不收敛。随着 Batch\_Size 增大，处理相同数据量的速度越快。随着 Batch\_Size 增大，达到相同精度所需要的 epoch 数量越来越多。由于上述两种因素的矛盾， Batch\_Size 增大到某个时候，达到时间上的最优，由于最终收敛精度会陷入不同的局部极值，因此 Batch\_Size 增大到某些时候，最终达到精度上的收敛。

这与我们最初的分析是吻合的，那么如何确定这个timing来确定我们的batch size，这一问题也与BigGAN中提到的一定程度的一味加强batch size会导致性能下降相似，可以得到相似的解决方案。

因为每批次都对模式进行了更多的覆盖，为生成和判别两个网络提供更好的梯度，由于 BigGAN 是训练 ImageNet 的各个类，所以通过加入条件标签 c 实现条件生成，如果在 BatchNorm 下嵌入条件标签 c 将会带来很多的参数增加，文章采用了共享嵌入，而不是为每个嵌入分别设置一个层，这个嵌入线性投影到每个层的 bias 和 weight，该思想借鉴自 SNGAN 和 SAGAN，降低了计算和内存成本，并将训练速度（达到给定性能所需的迭代次数）提高了 37％。

在文章中，作者介绍了一些调参方法，不过我的个人习惯batch size 以 256 为分界线。向下和向上训练后比较测试结果。若向下更好则折半，直到结果不再提升，大的batch size空间不太够用，小的batch size训练时间过长，所以需要找到一个比较好的分界线来确定，而bigGAN对这一问题比较一笔带过，虽然只是一个小问题，但毕竟对于大规模的数据而言，有些部分是跟小数据的训练不一样的，所以可以在此方面进行一些落笔，以便于复现训练。