-12 月汇报-Monthly Report

Generative Adversarial Networks:

Generative Adversarial Networks (GANs) not using any method to approximate the distribution of a data. Instead, using deep learning techniques to leaned distribution of the true data, and then it can output a result that conforms to the learned distribution of the data ultimately. The advantage of this is that we can directly generate data via learn the distribution of this data. The disadvantage is that we do not know what the distribution of the real data looks like. The training of GAN is similar to a two-player game in game theory, and after confrontation, it reaches a Nash Equilibrium. In one epoch, we first iterate D for k times, samples noise data generated by G and real data, then take the gradient of D's computation graph to update D. Then, we sample noise data for G and take the derivative of G's computation graph to update G. It is important to note that D and G must be "balanced in power," which means that one should not be too strong (and the other too weak, similarly). For example, if D is too strong, then G's outputs can easily be classified as 0, and the result is 1. After being mapped through the log function, the final number will converge to 0, and the gradient will also be close to 0, which makes it difficult for G to be updated. The theoretical optimal solution is $\frac{p_{data}(x)}{p_{data}(x)+p_g(x)}$. which is a valid conclusion to determine whether two distributions are the same distributions.

GAN 不使用任何方法去近似一份数据的分布. 而是使用深度的方法, 学习数据的分布, 使得最终可以输出一个符合学习数据分布的结果. 好处是说我们可以直接生成数据, 而不使用其他方法学习到这份数据的分布, 坏处是说我们不知道真实数据的分布是什么样子. GAN 的训练类似于博弈论中的双人游戏, 经过对抗, 最终达到纳什均衡. 在一个 epoch 中, 我们首先对 D 进行 k 次迭代，采样经过 G 的噪声数据和真实数据，然后对 D 的计算图求梯度，以此更新 D。然后采样噪音数据给 G，对 G 的计算图进行求导，更新 G。需要注意的是 D 和 G 要"势均力敌", 也就是说不要某方过强(另一方则过弱, 同理), 举个例子: 如果 D 过强, 那么 G 的输出都能更容易被判定为 0, 结果为 1, 经过 log 函数映射, 最终的数是收敛到 0, 那么求梯度也是靠近 0, 这就使得 G 很难被更新. 理论上最优解为 $\frac{p_{data}(x)}{p_{data}(x)+p_g(x)}$。（证略，公式还在看(⊙﹏⊙)）。这也是在判断两个分布是否是同一种分布的有效结论.

GAN Train process:

for epoch **do:**

  for k steps **do:**

    Sample from m noise from $p_g(z)$

    Sample from m examples from $p_{data}(x)$

    Calculate gradient to update discriminator:

$$\nabla_{\theta_d}\frac{1}{m}\sum_{i=1}^{m}\left[\log D\left(\boldsymbol{x}^{(i)}\right)+\log\left(1-D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right].$$

  end

  Sample from m noise from $p_g(z)$

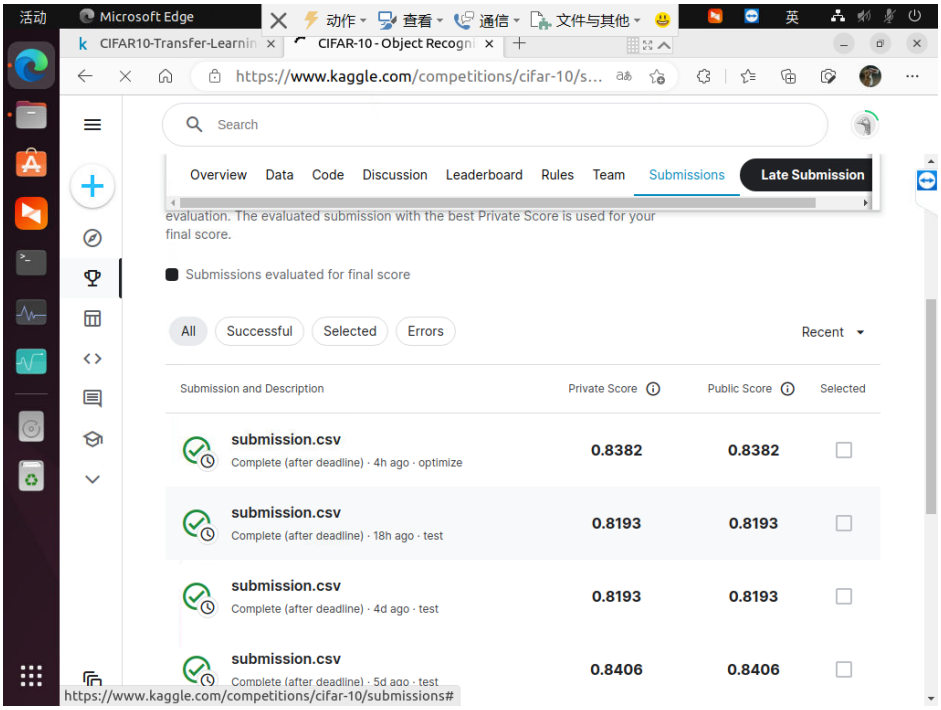
  Calculate gradient to update generator:

$$\nabla_{\theta_g}\frac{1}{m}\sum_{i=1}^{m}\log\left(1-D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end

工作

1. 继续学习《动手学深度学习》，目前学习到 NLP 的部分了，NLP 之前没太了解过，学习起来比较费劲，在啃书，进度比较慢。
2. 复现了一些经典的神经网络，VGG，GoogLeNet，ResNet, 以及最基础的 GAN 等。并基于 ResNet18 试了一下 Kaggle 上的 CIFAR10 比赛入门了一下。不过分数也就停留到.8 左右…不知道是我的 trick 不够好, 还是其他的什么原因, 仍再改进.



3. 读了一下 AlexNet 和 ResNet 的论文，看了沐神的论文精读系列。
4. 更新了毕设的功能，从 SSD 算法换成了 YoLo v5，并实现了上传视频和摄像头实时检测的功能。现在效果还可以，漏检率也很低。但是在检测视频时, 框有些抖动, 不连续. 这方面我应该如何解决???

下个月工作

1. 继续学习深度学习知识。预计下个月就能把李沐的课看完了.
2. 复现 RCNN, YoLo 网络。还在实现 faster R-CNN…但是最近有点难受, 提不起来兴趣, 所以进度有点缓慢…
3. 学习 AutoEncoder, VAE 等.

问题

1. Attention 要加在什么地方。是每个卷积之后？还是一个 Stage 或者 block 之后。而且我加了 attention 之后效果并没有变好很多。不知道是哪里出了问题。。。具体还没学到更深层次的东西，可能后面在学学，理解会变得深吧。
2. DataParallel, 和 DistributeDataParallel, 不知道老师在数据并行这方面了不了解. 我看 DP 的 reduce 都是在一个 GPU 上做的. 并行度不如 DDP. 有必要换成 DDP 吗, 还是说新手不用太在意这些细节的东西.
3. 目标检测时, 框有些抖动, 不连续不稳定, 这方面我应该如何解决, 追踪目标???