PPO off-policy 版本的实践经验

经典的 PPO 是一种 on-policy 算法，但是由于其截断机制的强大效果，同时也为了提高数据利用效率，在很多“野生”代码仓库中也会出现 PPO 的 off-policy 版本实现，并且实用效果往往还不错。因此，本文将介绍和概述 PPO off-policy 版本算法的设计动机与实际操作中的细节要点。

动机：为了更好地提升模型生产效率

强化学习算法的训练往往是一个收集数据与训练模型交替进行的过程。从之前 PPO 算法理论的推导可知，Policy Gradient 定理的梯度更新生效需要基于一个硬性的前置条件，即需使用当前策略在环境中采样生成训练数据。这意味着 PPO 算法需要使用严格的在线策略采集数据，并用实时数据更新策略。

对于实际生产实践，商业公司对模型生产效率有更高要求。如果通过模拟器可以稳定大量的生成训练场景中的数据，那么配合使用较大的训练批次样本数（Batch Size）就可以在训练中有效提高模型训练速度。

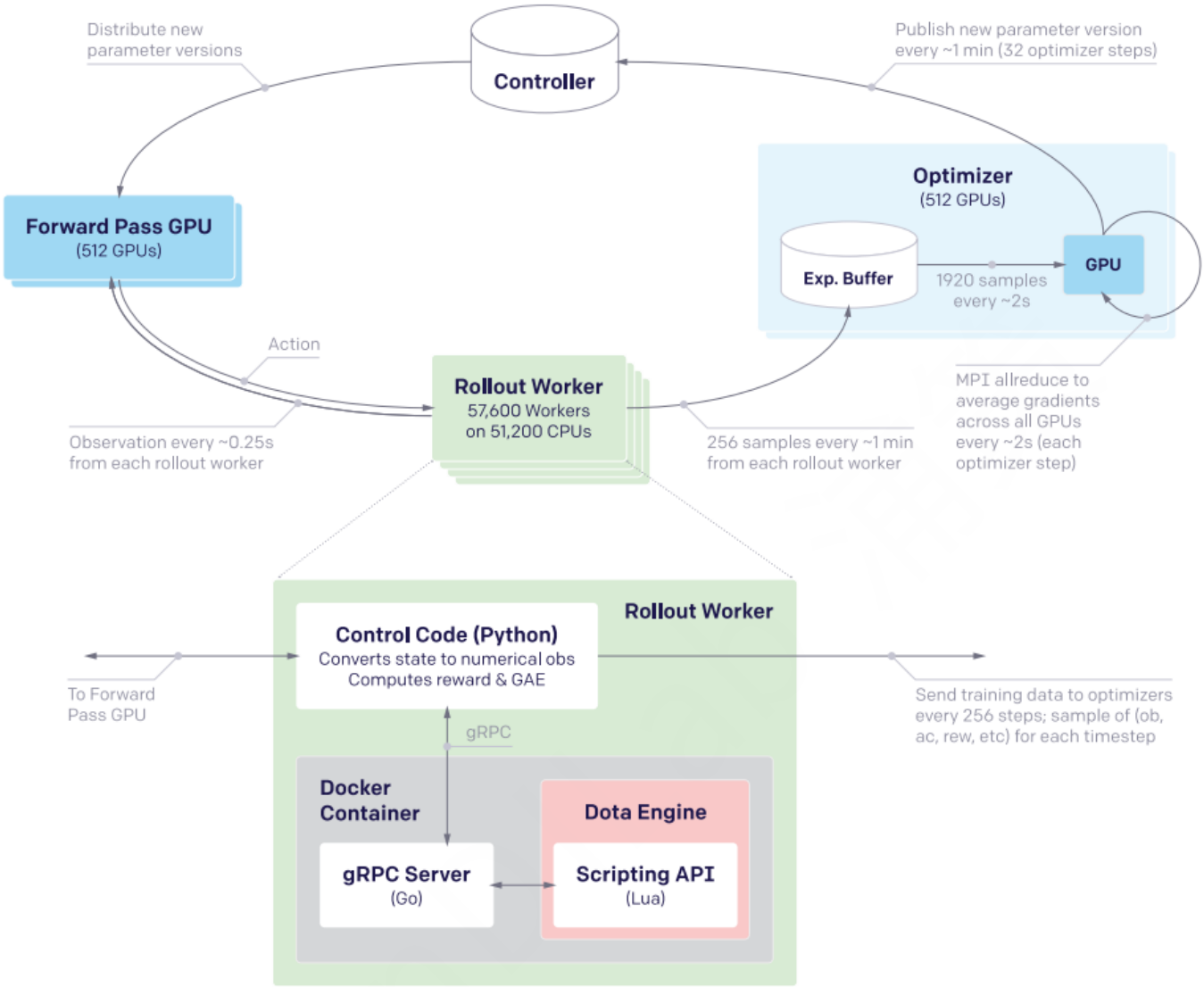
这样一来，如果继续绑定数据采集和模型训练这两个阶段，时间堵塞就会变得十分严重。同样的训练效率和训练方法下，输入更多的数据导致模型训练变慢，耗时过长，同时也会让环境数据采集的智能体需要更长的时间来等待模型更新，并暂停数据采集。尤其是复杂的环境交互或是分布式的计算场景，因为环境收集本身的速度受限，或是因为分布式通讯的架构使得数据模型传输同步变慢，这些种种原因都大大降低了算法面对大量有效数据的训练效率。因此需要有效的提升模型生产效率成了必须解决的问题。

解决办法：解耦和并行模型训练和数据采集

对于强化学习算法而言，一般来说，在模型设计和目标函数不变的情况下，需要通过提升数据利用率来提高算法的训练效率。

具体的，在 PPO 算法中，假如可以允许使用非当前策略生成的数据，比如一定更新次数内的旧策略生成的数据，这样就可以解耦环境数据采集一侧和模型训练梯度更新一侧的进程，让大规模情况下的数据并行采集训练成为可能。即，采用大规模的 Actor 收集环境数据，Actor 中的策略为某一折旧程度下的策略，每一份收集的数据会被记录其为具体哪一轮更新的策略序号。模型训练一侧的 Learner 则收集获得所有的数据，训练最新的策略，标记策略序号，并分发给各个 Actor 更新其模型。

2019 年 4 月 13 日，OpenAI Five 成为第一个在电子竞技比赛中击败世界冠军的人工智能系统。OpenAI 的研究者们在其 DOTA2 环境下多智能体的训练实践中，参见《**Dota 2 with Large Scale Deep Reinforcement Learning**》[1]，就采用了这一种 Asynchronous PPO （APPO）方法，其训练框架如下图所示：



（图 1：OpenAI Five 训练系统概览图 [1]。）

> 在 OpenAI Five 的训练框架中，Rollout Worker 组件主要负责调用策略，并与环境执行交互，计算优势函数，并将采集的数据加入历史缓存中。Optimizer 组件负责将缓存中的数据提取，并执行模型的优化。

Offpolicy PPO 原理

虽然在原理上，Policy Gradient 算法的成立条件在 Offpolicy PPO 的实践中并不成立，然而往往通过一些合适的参数设计，Offpolicy PPO 的实践也可以取得很好的效果。接下来我们来分析这其中的原因：

1、PPO clip trick

我们回顾一下 PPO 算法的优化目标为：

$$\mathbb{E}_t[\min(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}\hat{A}^{\theta_k}(s_t,a_t), \text{clip}(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}, 1-\epsilon, 1+\epsilon)\hat{A}^{\theta_k}(s_t,a_t))]$$

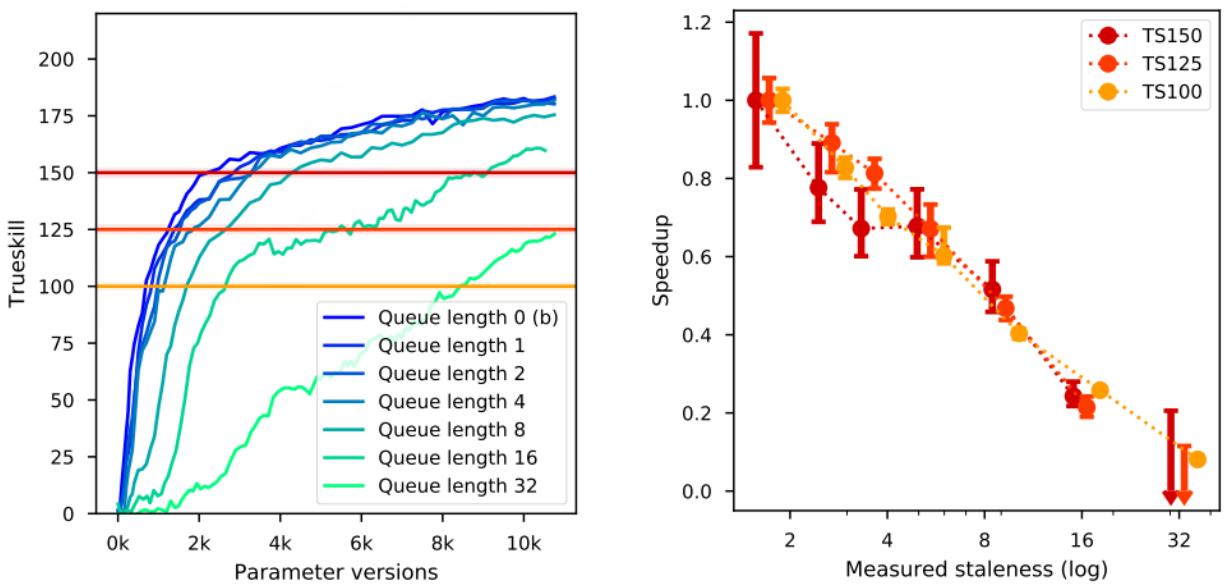
在训练中，我们需要将 PG 算法中重要性采样的项 $\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}$ 截断至 $[1-\epsilon, 1+\epsilon]$ 的范围内。需要注意的是，上式中的优势函数 $\hat{A}^{\theta_k}(s_t,a_t)$ 在算法中并不传播梯度，所有的梯度更新将围绕着 $\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}$ 中的参数 θ 展开。因此，假如对于某条轨迹数据，$\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} \leq 1-\epsilon$ 或 $\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} \geq 1+\epsilon$，那么 $\text{clip}(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}, 1-\epsilon, 1+\epsilon)$ 将是一个常数，从而参数 θ 不会被此条轨迹数据更新。

基于此可以认为，对于那些和当前策略偏离太大的策略生成的数据，即当其重要性采样比率与 1 偏差大于 ϵ 之后，将在训练过程中被自然地过滤掉。PPO 里，所有将会产生的策略梯度更新，都只会在一个较小的偏离范围内发生，从而使得 PG 原理依然生效。即使训练中使用了大量 offpolicy 数据，也仅会采纳其中一部分合理数据带来的梯度更新。PPO clip trick 这一技巧为 offpolicy PPO 算法的实践使用提供了较强的保障。

2、尽可能小的数据陈旧程度

当解耦数据采样与模型训练的串行关系之后，Learner 在训练时被给予的数据将可能不再是来自拥有最新模型版本的 Actor 采集到的数据。假如标记当前 Learner 已生成的最新版本的模型序号为 M，而 Learner 用于训练模型的数据来源的对应 Actor 所使用的模型序号记为 N，则可以定义数据的陈旧程度：

$$\text{staleness} := M - N$$



（图 2：OpenAI Five Staleness 实验对比图。）

OpenAI Five 的实验发现，通过增大队列长度来制造人工的陈旧数据，当数据陈旧程度越高，模型训练获得一定等级水平的智能体的速度越慢。这是由于当陈旧程度变大之后，那些无效的训练数据的比重

也会增加，从而让有效的梯度更新变少。因此，在实践中，监控上述数据统计量，并及时调整训练超参数，保证训练一侧的实时数据陈旧程度处于较低的程度，对训练结果和效率的提高，都非常重要。

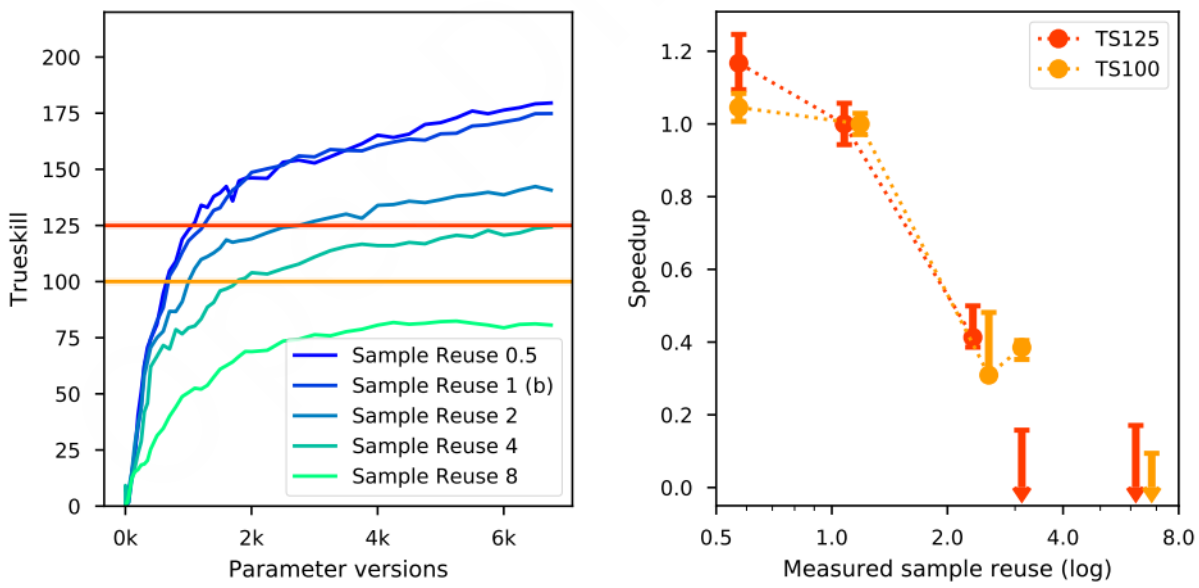
那么，应该如何时刻保持数据的陈旧程度一直处于较低水平呢？一方面，需要在数据传输的节奏上设计一个合理的速率，比如在 OpenAI Five 的训练系统中，环境每 256 个时间步之后，数据就会被打包和向外发送用于训练。这是因为如果收集更多的数据再发送，数据陈旧程度会变大，而如果更快频率地发送数据，可能会受到通讯方面的限制而影响系统整体效率。另一方面，在数据的利用率上，需要遵循采集速率和训练速率保持一致的原则，根据实际硬件设计的数量，来安排具体的方案。

3、采集速率和训练速率保持一致的原则

这个原则是为了让数据采集生成数据的速度，和模型训练消耗数据的速度一致，从而让所有采集到的数据都不会被浪费，也不会因为采集数据不足，而使得有一些数据在训练过程中被多次使用，而在数据陈旧程度上给模型训练带来不利的影响。

在具体的实践中，来自环境智能体 Actor 的数据会被发送到一个数据缓存池中，模型的优化器 Optimizer 会根据一定的训练速率（具体数值取决于具体的硬件性能和模型大小），从数据缓存池中采样获取数据。然后调节和平衡智能体的采集数据的速度，让采集速率和训练速率保持相同，在此状态下，整个训练系统会有最优的配置状态。

在 OpenAI Five 的训练系统中，研究者们设计开展了在数据缓存池中，让数据使用不同次数的对比实验，如下图所示，当采样数据的使用次数小于 1 时，智能体的表现提升的速率最快。



（图 3：OpenAI Five 数据采样使用次数实验对比图。）

参考文献

[1] Berner C, Brockman G, Chan B, et al. Dota 2 with large scale deep reinforcement learning[J]. arXiv preprint arXiv:1912.06680, 2019.