Phasic Policy Gradient（PPG）

本文介绍 PPO 的一种改进算法，阶段式策略梯度算法（PPG，Phasic Policy Gradient）[1]。阶段式策略梯度算法 PPG 主要研究如何在利用好共享网络带来的训练速度提升的同时，缓解神经网络参数在多目标（任务）优化时梯度互相干扰的问题。

下面我们将先分析经典 RL 文献中，Actor-Critic 算法共享主干底层编码器网络的优缺点，然后分析 Policy 与 Value 的在 RL 优化时的不同特性，最后介绍 PPG 算法的设计思路。

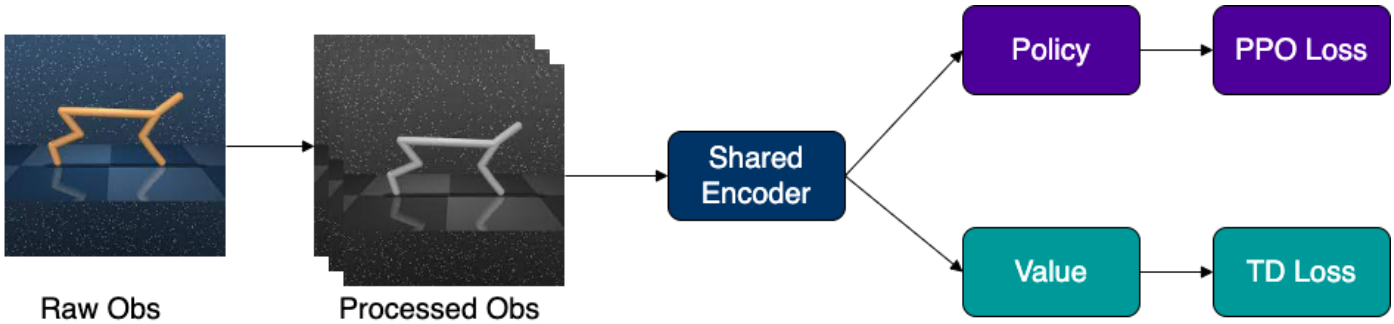
动机：共享还是不共享？这是一个问题

共享的优势

在强化学习的很多场景中，我们经常需要处理高维的环境观测输入，这会给我们的算法带来较大的挑战：**高维观测的信息密度低，数据中的有效信息常常较难捕捉，使得算法训练变得缓慢**。

而在强化学习 Actor-Critic 类算法的实践中，我们需要同时训练策略模型和价值模型，它们都需要相同的状态观测作为输入。实践中发现，既然这两个模型都采用了同一个输入，不如让其共享部分编码器网络（Encoder），互相促进对于高维表征的学习，输出共同的观测特征，用于两个模型的训练。

这种共享网络（shared network）方法有效地缓解了高维输入的特征提取问题，相比于完全独立的网络结构，它在大部分环境（Atari，Procgen等）下都可以获得更快收敛与更加稳定的效果。



（图1：共享编码器网络的 Actor-Critic 算法输入输出优化数据流）

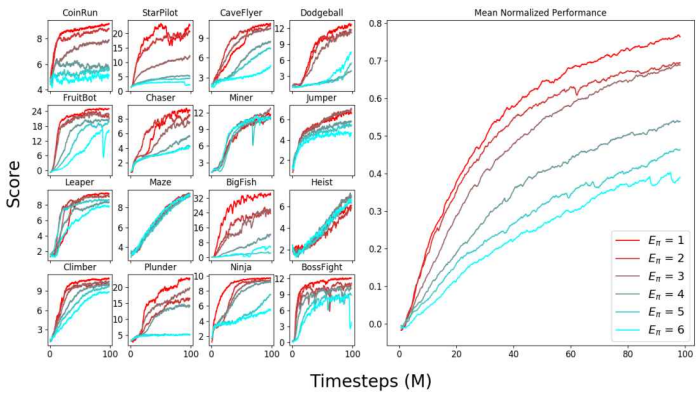
共享的劣势

不过，遵循机器学习中常见的 no free lunch 定理 [2]，共享网络模式的特性也是一体两面的，不仅有上面所述的优点，也存在一些潜在的弊端。经过研究发现，共享网络这种模型结构的弊端之一就是共享网络会引入互相干扰的目标函数。众所周知，模型的训练结果与模型的目标函数的设计息息相关。一个合理的目标函数定义了模型的设计目标与要求，也决定了模型的表现。具体而言，在强化学习 Actor-Critic 类算法中，Actor 与 Critic 模型的目标函数是不相同的，前者使用策略梯度算法更新，后者通过最小化贝尔曼误差（即隐式地求解贝尔曼方程）来更新价值网络。

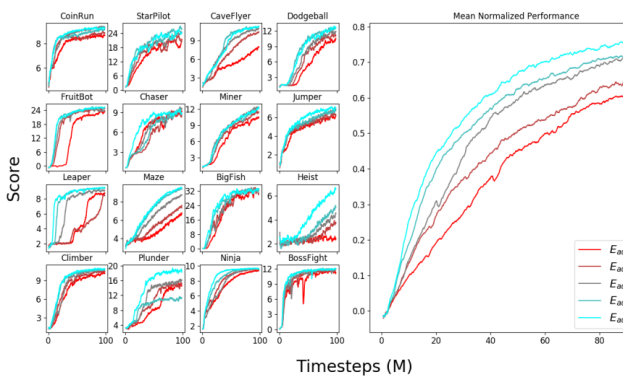
假如我们在 PPO 算法中使用共享网络，让策略模型（Policy Model）与价值模型（Value Model）互相共享一部分模型，比如负责特征处理的那部分网络（Encoder）。那么这一部分共享的网络就会被策略模型与价值模型各自的目标函数共同优化，进而使得它成为了两个不同的目标函数互相争夺的焦点。

Policy 与 Value 的不同特性

此外，由于策略提升定理的内在要求，策略模型一般需要严格使用 on-policy 数据，即收集数据的模型和训练的模型保持一致。而价值模型往往需要更为多样化的数据，用于更好地拟合价值函数，因此我们常会使用经验回放（Replay Buffer）中的数据来更新价值模型。
比如在 PPG 原论文的实验中，演示了控制和保持价值模型更新次数不变，以增加策略模型的训练次数的方式，来增加策略模型的数据重用度，会明显导致性能下降；而增加价值模型的数据重用度却可以提升价值模型的表现，具体如下图所示：



(a) Policy sample reuse



(b) Value function sample reuse

(图中 E_π 为策略数据的重用度， E_{aux} 为价值函数训练使用的数据的重用度)

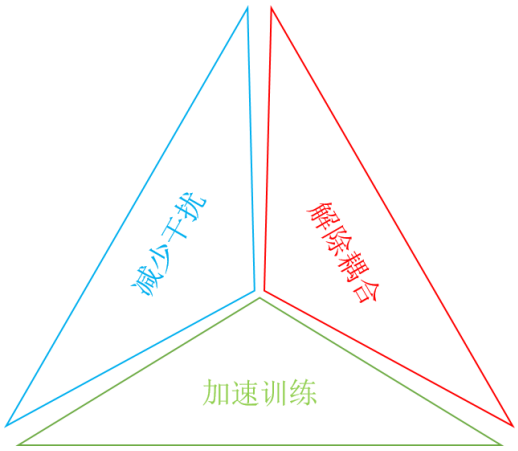
(图2: 分别在策略模型与价值模型上调整数据重用度，Procegen 环境上 PPO 智能体性能的变化情况。)

上述实验表明，在共享网络的情况下，如何协调两个目标函数的更新频率和数据重用度是一个关键点。比如使用更大的数据重用度（Reuse）来更新价值模型，那么共享网络的参数无疑会受到价值网络的目标函数的影响更大。

通过上述的直观分析和实验分析，我们会认识到，如果希望对PPO算法进行优化，其实有以下三个不同的优化的方向可供思考：

1. 不同模型通过诸如共享主干（编码器）网络的方式，来共享训练进展，加快训练速度
2. 假如网络被共享，其在双目标优化过程中，需要减少对于共享网络的参数同时更新所带来的干扰
3. 解除两个模型训练频率的耦合关系，相对独立地进行各自的优化过程，提高数据利用率

这三个目标就成为了一个看似不可能同时满足的三角，三者只能取其二。



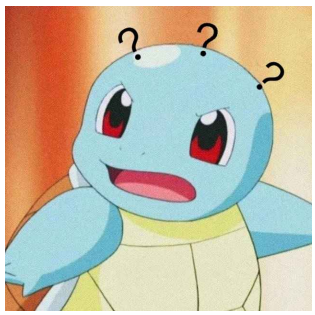
（图3: PPO算法的改进方向）

- 使用共享网络并较少利用历史数据回放，即，选择目标1和目标2就难以达成目标3。
- 使用共享网络并较多利用历史数据回放，即，选择目标1和目标3就难以达成目标2。
- 不使用共享网络，解耦两个模型各自的训练过程与训练数据，即，选择目标2和目标3就难以达成目标1。

阶段式策略梯度算法 PPG 就是为了尝试改进 PPO 算法，解决上述问题而提出的。其算法目标就是为了，能不能既享有类似于共享网络带来的加速训练的优点，同时又能缓解或避免参数在多个目标的梯度更新时的互相干扰的问题。

PPG 核心算法：升维

正如上文所说，Actor-Critic 算法使用两个网络和其目标函数，但却最好需要同时满足三个目标。不过两支弓箭怎么同时射中三个目标，这确实有点强人所难。

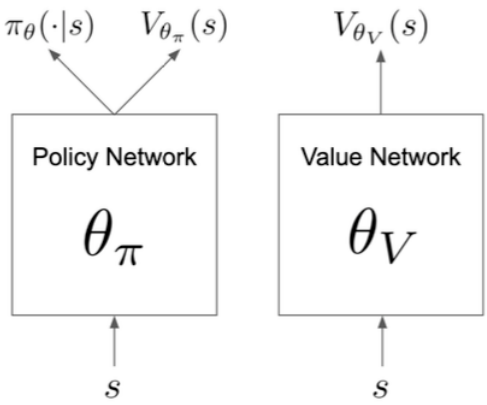


不过 PPG 算法的作者们（Karl Cobbe, Jacob Hilton, Oleg Klimov, John Schulman）通过“升维”的思路找到了这个问题的一种解法：增加一个网络。与其做这种三选二的选择，他们表示那就再增加一支弓箭，三个目标全都要了。

创新往往源于**守正出奇**。

- 所谓守正，就是要按照应有的方式来面对、适应和遵循万物的规律。既然 Actor-Critic 算法中的策略模型和价值模型各自有不同的目标函数与适用数据，那么我们就应该选用两个独立的模型来构建它们，减小了两个目标函数之间的干扰，然后用两个独立的训练过程来解耦地训练它们。而不应该简单粗暴地在模型之间使用共享网络。

- 所谓出奇，就是要使用科学的方法来应对，引导和改变原有的局面。现在的问题是，两个模型独立之后就没办法利用共享特征编码器来加速训练。为了解决这个问题，那其实可以再设计一个新的优化目标函数，间接地传递信息帮助优化。那么问题来了，应该如何设计这个目标函数，让两个独立的网络来共享训练进展和信息呢？要是这两个模型输出的都是同样的数值就好了，这样我们就可以让两个独立的模型的输出进行互相借鉴互相学习。不过很明显，PPO的两个模型的输出并不相同，一个是策略模型，一个是价值模型，牛头不对马嘴，似乎做不到呀。不过仔细想想，其实也不是不行，比如，干脆可以让策略模型也输出价值模型的输出，这样一来，是不是就可以互相对照和比较了。



（图4：PPG 算法网络结构原理图）

于是，如图4所示，PPG的作者在原有的 PPO 的策略模型输出 $\pi_\theta(\cdot|s)$ 与价值模型输出 $V_{\theta_V}(s)$ 之外，额外地让策略模型增加了另一个价值输出 $V_{\theta_\pi}(s)$。他们期望通过特殊的优化方式，即让策略模型的价值输出 $V_{\theta_\pi}(s)$ 与价值模型输出 $V_{\theta_V}(s)$ 互相比较，从而让两个独立的策略模型和价值模型，借助这种方式隐式地传输或共享来自于各自训练过程中的信息。那么，具体是如何实现的呢？PPG其实主要分为两个大的阶段。

PPG 引入了下面的的**辅助阶段**（auxiliary phase）：

- 方法是对（独立的）价值模型进行**特征知识蒸馏**，将价值模型当前学习到的关于环境的信息，通过最小化策略模型的价值输出 $V_{\theta_\pi}(s)$ 与价值模型输出 $V_{\theta_V}(s)$ 之间的差值，蒸馏到策略模型中去，具体的目标函数设计如下：

$$L_{\text{Auxiliary}}(\theta_p) = \mathbb{E}_t[||V_{\theta_v}(s_t) - V_{\theta_\pi}(s_t)||_2^2]$$

- 为了让知识蒸馏的过程不影响策略网络原有输出的动作分布，这里需要多一个小技巧，即使用加权的 KL 散度，$\beta_{\text{clone}}\text{KL}(\pi_{\theta_{\text{old}}}|\pi_\theta)$，来对（知识蒸馏的过程中）原有策略模型发生较大动作分布偏离的情形进行惩罚。通过这个小技巧，可以保证这个新加的辅助目标函数基本不会与 PPO 原有的策略函数产生矛盾。即目标函数修正如下：

$$L_{\text{Joint}}(\theta_p) = L_{\text{Auxiliary}}(\theta_p) + \beta_{\text{clone}}\text{KL}(\pi_{\theta_{\text{old}}}|\pi_\theta)$$

当然，PPG 算法还有 PPO 算法的常规流程，即**策略阶段** (policy phase)：

- 在策略阶段，模型会沿用 PPO 的训练流程，按照相同的目标函数更新策略模型和价值模型的参数。因此分为2个部分：
  - 策略模型的目标函数为（其实就是沿用 PPO ，截断式优化目标 + Entropy Bonus）：

$$L_{\text{Policy}}(\theta_\pi) = \mathbb{E}_t[\min(r_t(\theta_\pi)\hat{A}_t, \text{clip}(r_t(\theta_\pi), 1-\epsilon, 1+\epsilon)\hat{A}_t)]$$

- 价值模型的目标函数为：

$$L_{\text{Value}}(\theta_v) = \mathbb{E}_t[||V_{\theta_v}(s_t) - V_{\text{target}}(s_t)||_2^2]$$

- PPO 常规优化这部分使用 $V_{\theta_V}(s)$ 来计算优势函数的估计 $\hat{A}_t$ 和价值函数目标的估计 $\hat{V}_t^{targ}$

PPG 算法的整个流程就是围绕这三个目标函数依次更新，轮流进行策略阶段和辅助阶段而开展的，因此算法称为“阶段式策略梯度算法”。PPG 算法的伪代码如图5所示：

Algorithm 1 PPG

for phase $= 1, 2, \ldots$ **do**
  Initialize empty buffer B
  for iteration $= 1, 2, \ldots, N_\pi$ **do** ▷ Policy Phase
    Perform rollouts under current policy π
    Compute value function target $\hat{V}_t^{\text{targ}}$ for each state s_t
    for epoch $= 1, 2, \ldots, E_\pi$ **do** ▷ Policy Epochs
      Optimize $L^{clip} + \beta_S S[\pi]$ wrt θ_π
    for epoch $= 1, 2, \ldots, E_V$ **do** ▷ Value Epochs
      Optimize L^{value} wrt θ_V
    Add all $(s_t, \hat{V}_t^{\text{targ}})$ to B
  Compute and store current policy $\pi_{\theta_{old}}(\cdot|s_t)$ for all states s_t in B
  for epoch $= 1, 2, \ldots, E_{aux}$ **do** ▷ Auxiliary Phase
    Optimize L^{joint} wrt θ_π, on all data in B
    Optimize L^{value} wrt θ_V, on all data in B

（图5： PPG算法伪代码）

总结

综上所述，PPG的优势之处主要在于如下几点：

- PPG 使用了独立的网络保持了独立解耦的策略阶段与辅助阶段的训练，它通过在策略模型中新增了一个用于价值输出的网络，配合使用特征蒸馏的技巧，将原独立价值网络的信息，蒸馏到策略模型中，从而加速了训练过程。
- 因为PPG中知识蒸馏方向是单向的，所以它与直接共享参数不同，这种方式不会有共享网络的负面影响。它既成功地让价值模型帮助了策略模型的训练，又让策略模型的更新不会直接影响价值模型。
- 独立的价值网络也可以更多地重用来自于回放的历史数据，从而大大提高了数据利用效率。

这样来看，通过使用更多的训练目标，分别匹配多个需求，才能在各个目标之间达成不错且游刃有余的平衡。在 PPG 的原论文中，作者实验发现了数据重用越大，策略模型表现越差，价值模型表现越好的现象。并进一步从现象中受到启发，提出了PPG 算法，然后作者实验比较了 PPG 和 PPO 的性能，发现 PPG 在 Procgen 的多个环境中都有更优异的表现。

此外，作者还研究了一些细节与小技巧：

- PPG 的辅助阶段的训练频率设计的最优取值
- PPG 中使用截断技巧和 KL 散度对实验表现的影响和比较
- PPG 中为了降低模型参数，使用共享网络但切断策略阶段的价值模型对共享网络的更新

具体实验相关的内容可以参考原论文实验部分。

参考文献

[1] Cobbe K W, Hilton J, Klimov O, et al. Phasic policy gradient[C]//International Conference on Machine Learning. PMLR, 2021: 2020-2027.

[2] https://en.wikipedia.org/wiki/No_free_lunch_theorem