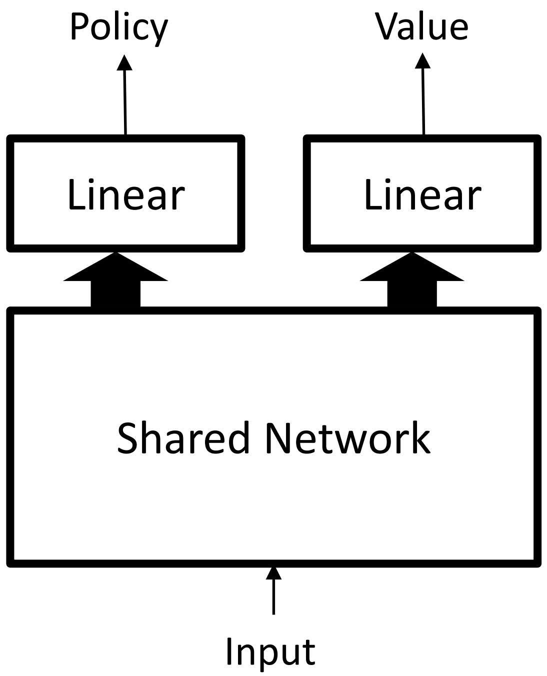
Phasic Policy Gradient PPG

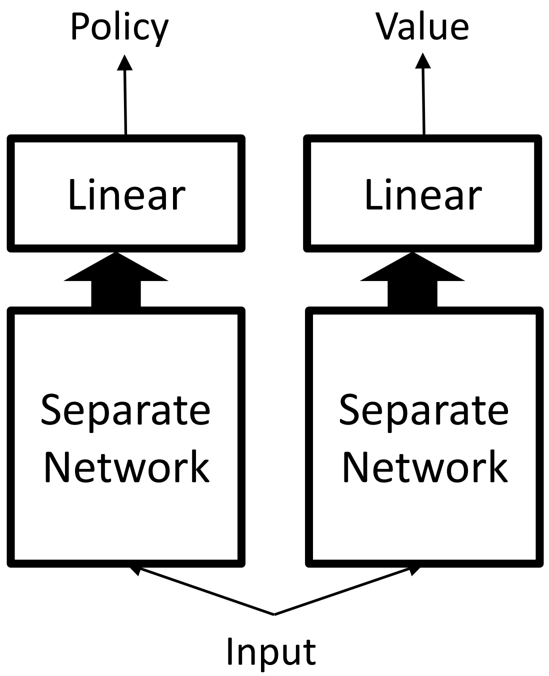
Sungkwon On
13 – March – 2022

Abstract

The traditional on-policy actor-critic method must choose between:

1. using a shared network – allows useful features to be shared
2. separate networks to represent the policy and value function – avoids interference between objectives



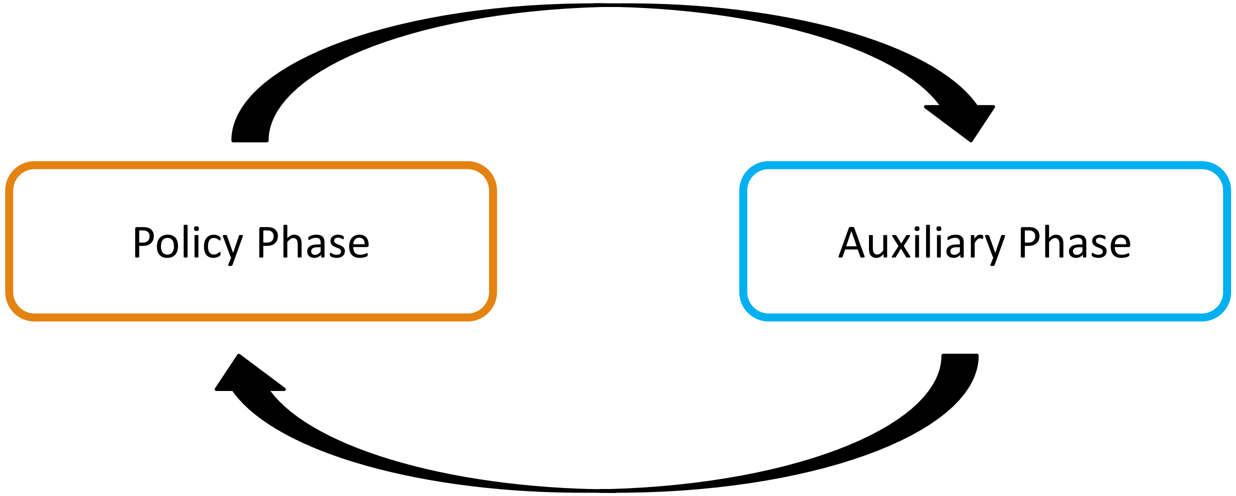


Abstract

Phasic Policy Gradient (PPG) modifies the traditional methods by separating policy and value function training into distinct phases.

PPG achieves the best of both methods by splitting optimization into two phases.

We can control the frequency of certain phase for a higher level of sample reuse.



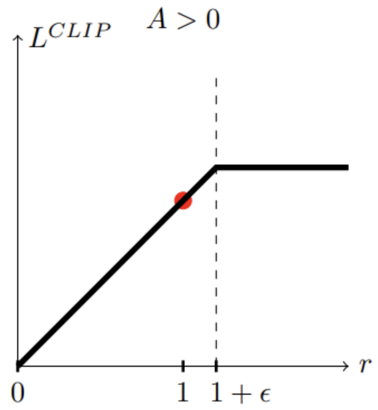
PPO review

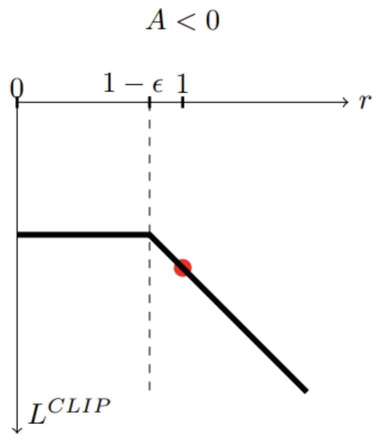
A traditional on-policy actor-critic method.

Probability ratio clipping prevents the policy from changing too much for one update.

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\Big[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\Big]$$

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}$$





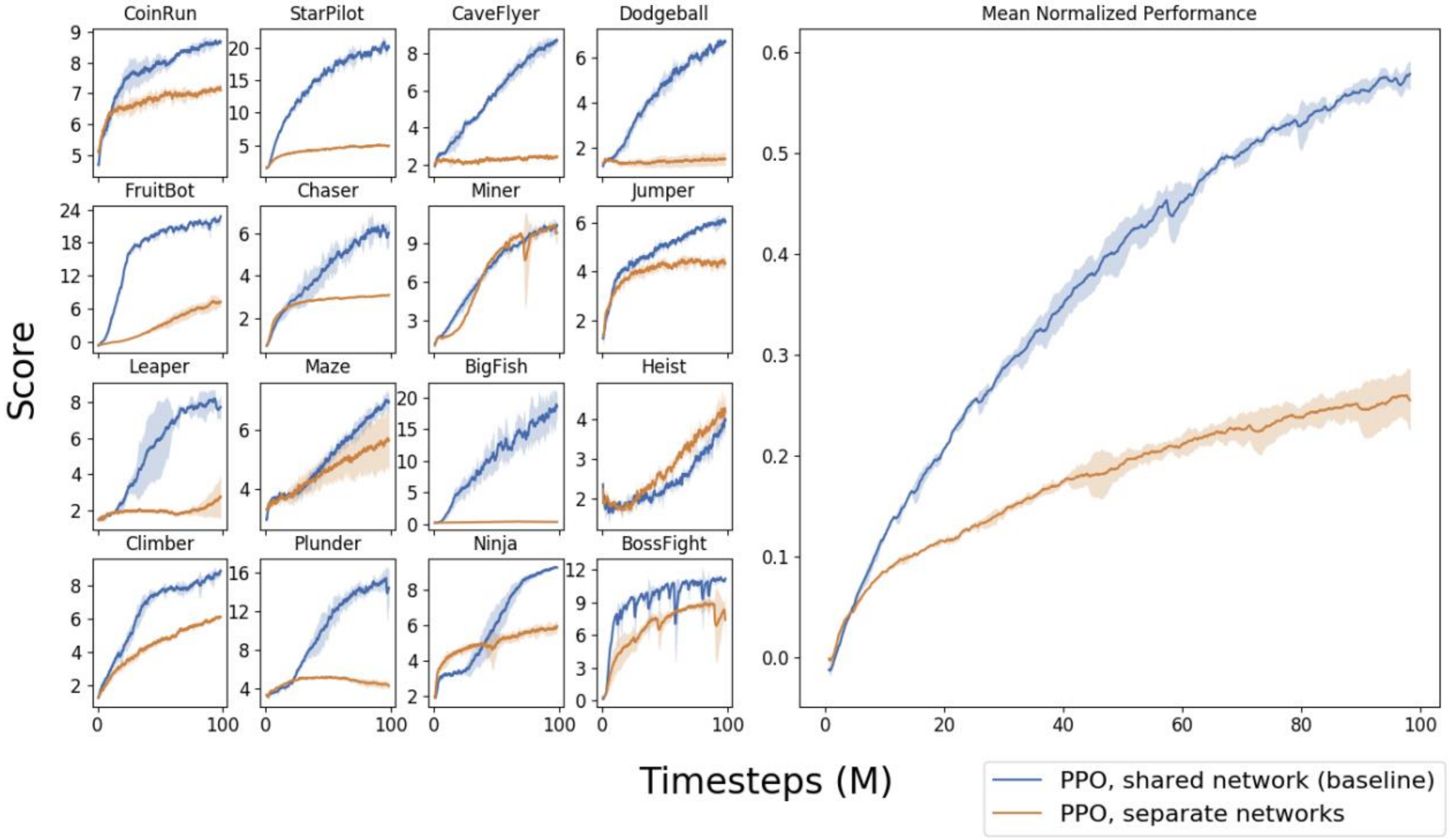
Procgen Environment

- Many common RL benchmarks ignore generalization
- Do agents learn robust skills or memorize trajectories?
- Atari: high diversity across envs but low diversity within a single env.

PPO comparison of shared and separate networks

Advantage of Network sharing:
Features trained by each objective can be used to better optimize the other

Aside:
The difference in the performance of the two methods is low for low input dimensional environments.



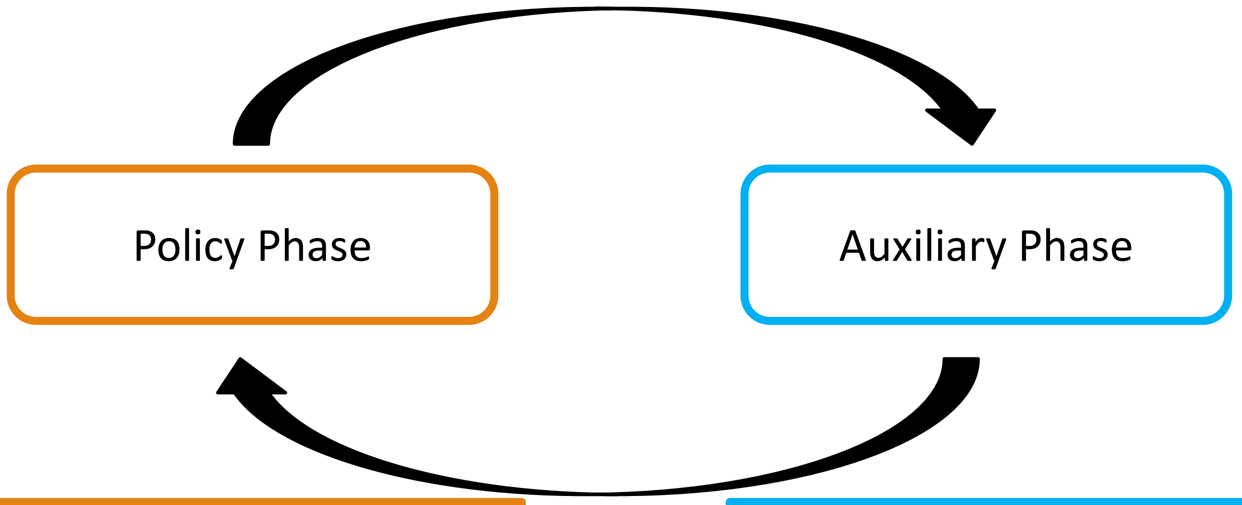
The Downside of Shared Networks

- There will be some amount of interference between policy and value function networks
  - Not clear how to balance the competing objectives of the policy and value function
  - Risk that the optimization of one objective will interfere with the optimization of the other
- The policy and value function must be trained on the same data (same level of sample reuse)
  - Undesired restriction!

- PPG;
  - Preserve sharing of useful features
  - Decouples policy and value function training

  Therefore PPG can;
  - More aggressively train value function (using higher sample reuse)
  - Reduce negative interference between policy and value function optimization

Algorithm



Policy Phase:

- PPO update
- Optimize both Policy and Value Network
- Store all states and value targets in replay buffer

Auxiliary Phase:

- Optimize Value Network while cloning prior policy

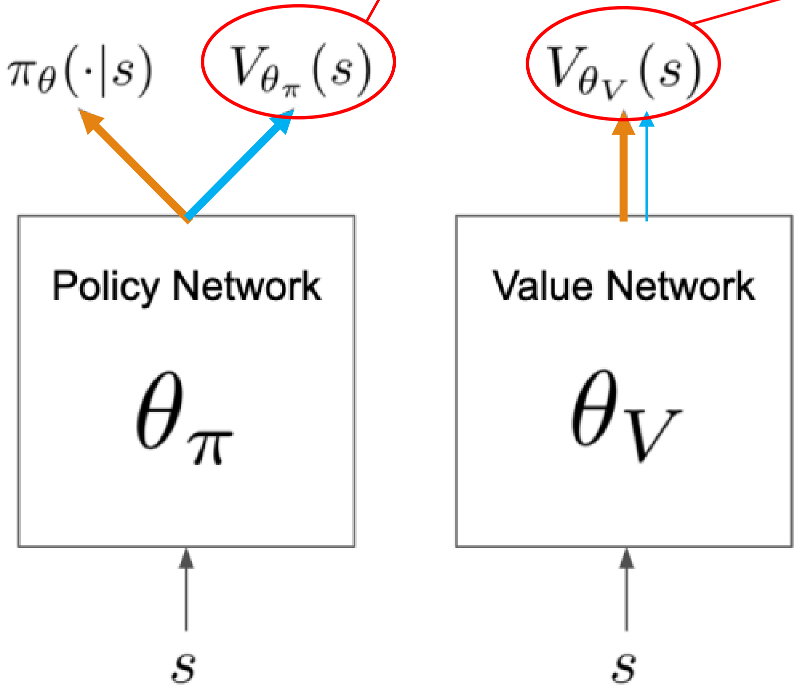
Network Structure

Only used to train policy network

True Value function. Used to compute target and advantage.

Policy Phase

Auxiliary Phase



Objectives

Policy Phase

$$L^{clip} = \hat{\mathbb{E}}_t \left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t) \right]$$

$$L^{value} = \hat{\mathbb{E}}_t \left[\frac{1}{2} (V_{\theta_V}(s_t) - \hat{V}_t^{\text{targ}})^2 \right]$$

Auxiliary Phase

$$L^{joint} = L^{aux} + \boxed{\beta_{clone} \cdot \hat{\mathbb{E}}_t \left[KL[\pi_{\theta_{old}}(\cdot|s_t), \pi_\theta(\cdot|s_t)] \right]}$$

Preserves the original policy

$$L^{aux} = \frac{1}{2} \cdot \hat{\mathbb{E}}_t \left[(V_{\theta_\pi}(s_t) - \hat{V}_t^{\text{targ}})^2 \right]$$

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

Entropy bonus

Hyperparameters

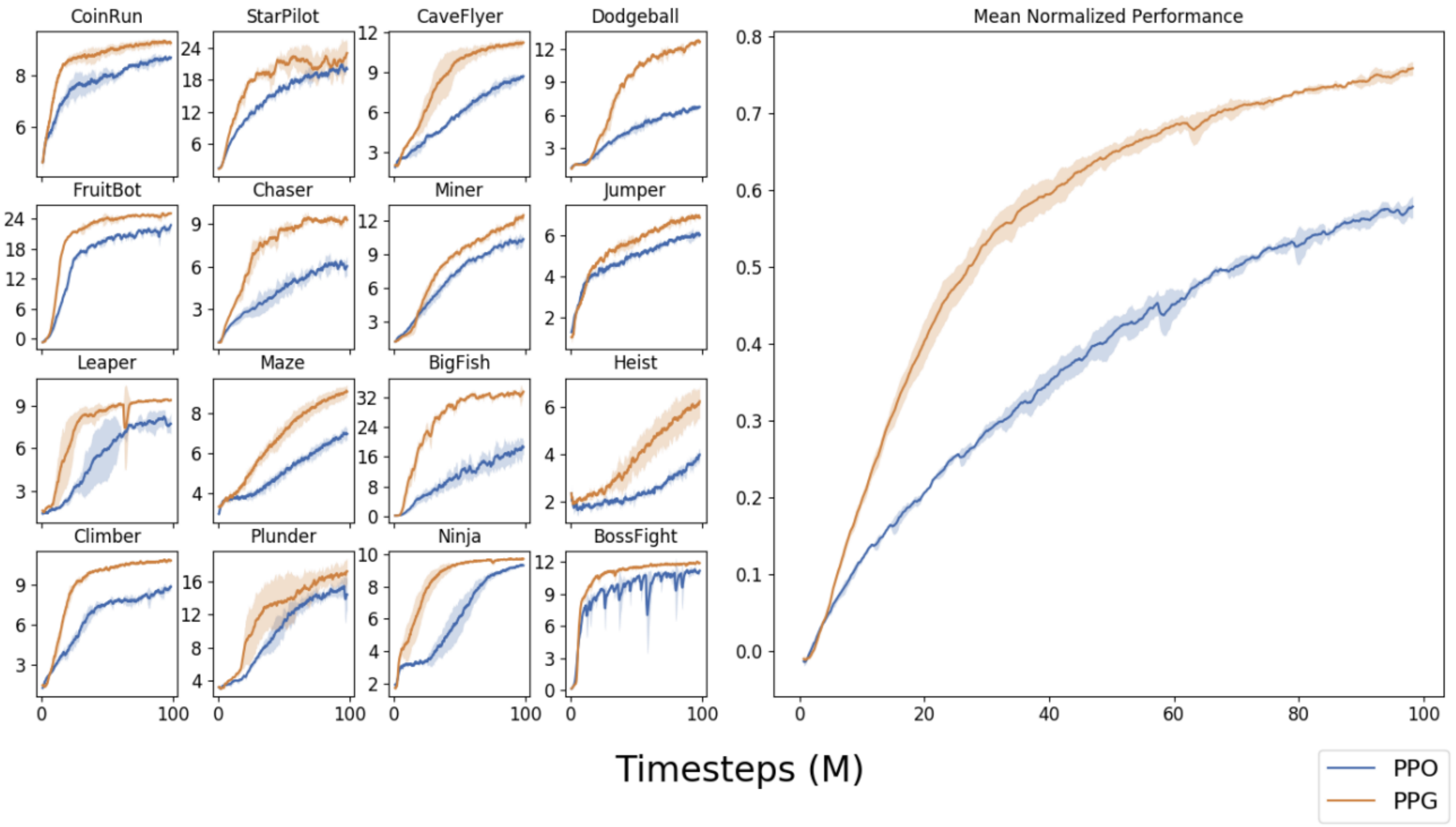
- N_π controls the number of policy updates in each policy phase

- E_π and E_v control the sample reuse for the policy and value function respectively
- Note: E_v influences the training of the true value function, not the auxiliary value function

- E_{aux} controls the sample reuse during the auxiliary phase
- Sample reuse for value function training is controlled by E_{aux} rather than E_v

Comparison to PPO

PPG outperforms PPO in environments with high dimensional input space



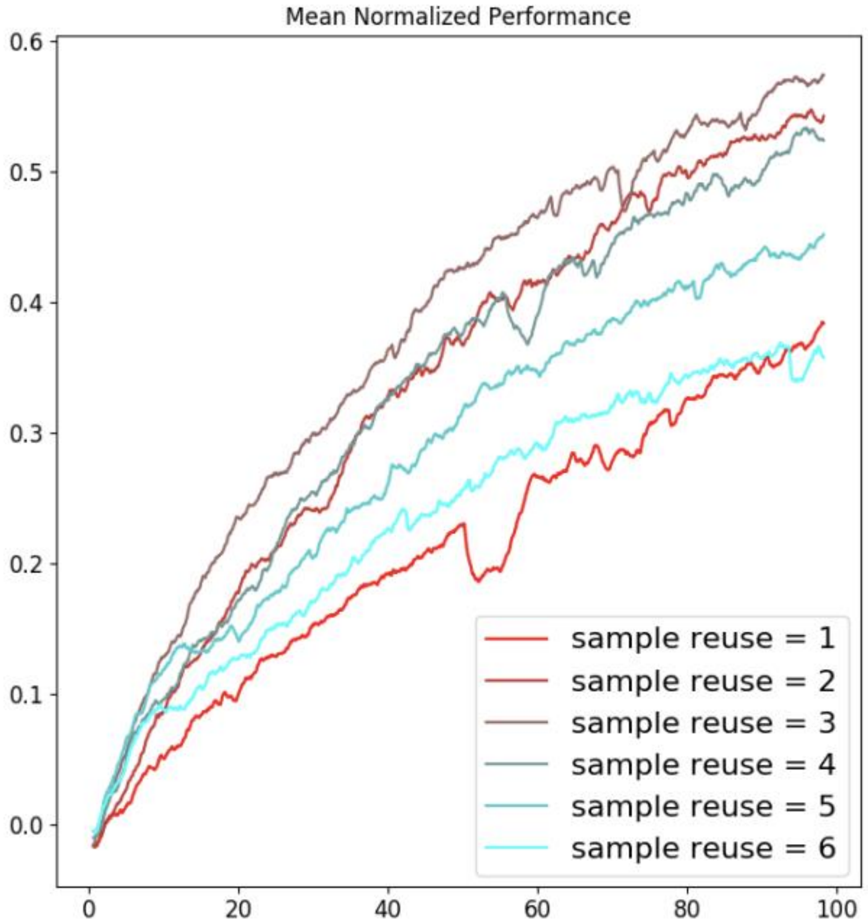
Sample reuse in PPO

Sample reuse can be controlled by varying the number of iterations of the optimization.

For PPO, sample reuse = 3 is empirically optimal.

But not sure which of the sample reuse is beneficial, policy or value function.

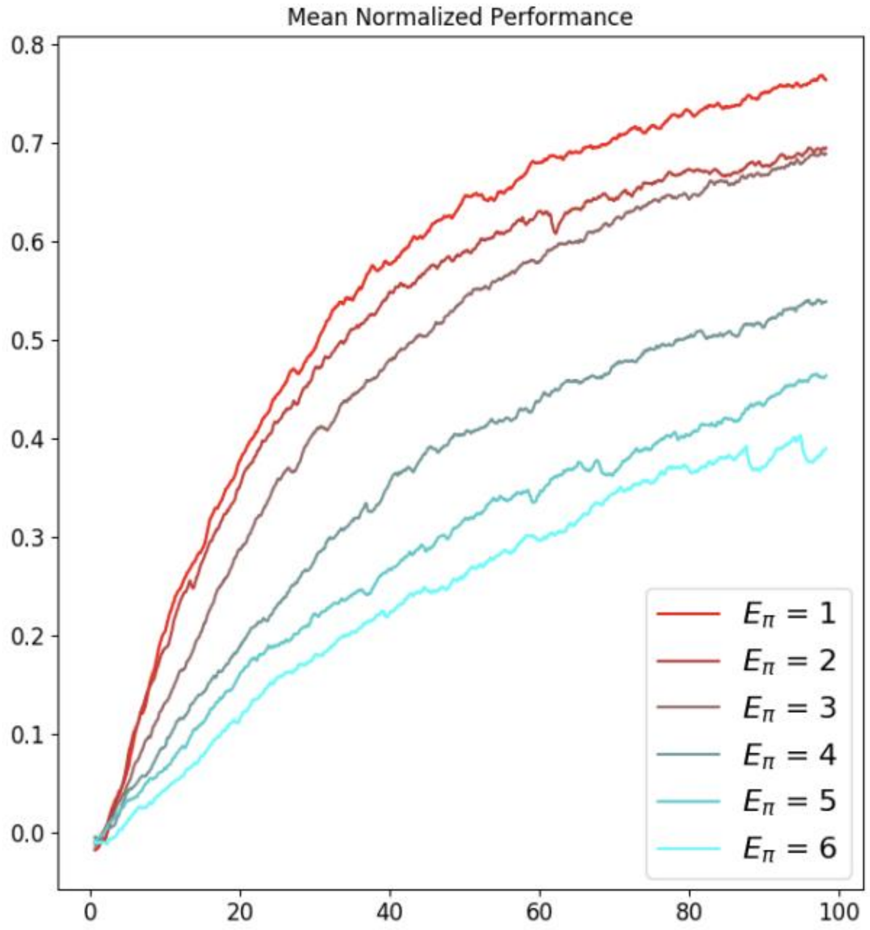
In PPG, optimization of policy and value function is separated, therefore we can test them separately.



Policy Sample Reuse

Best performance at 1 iteration of policy optimization.

PPO benefitted from sample reuse purely due to additional training of the value function.



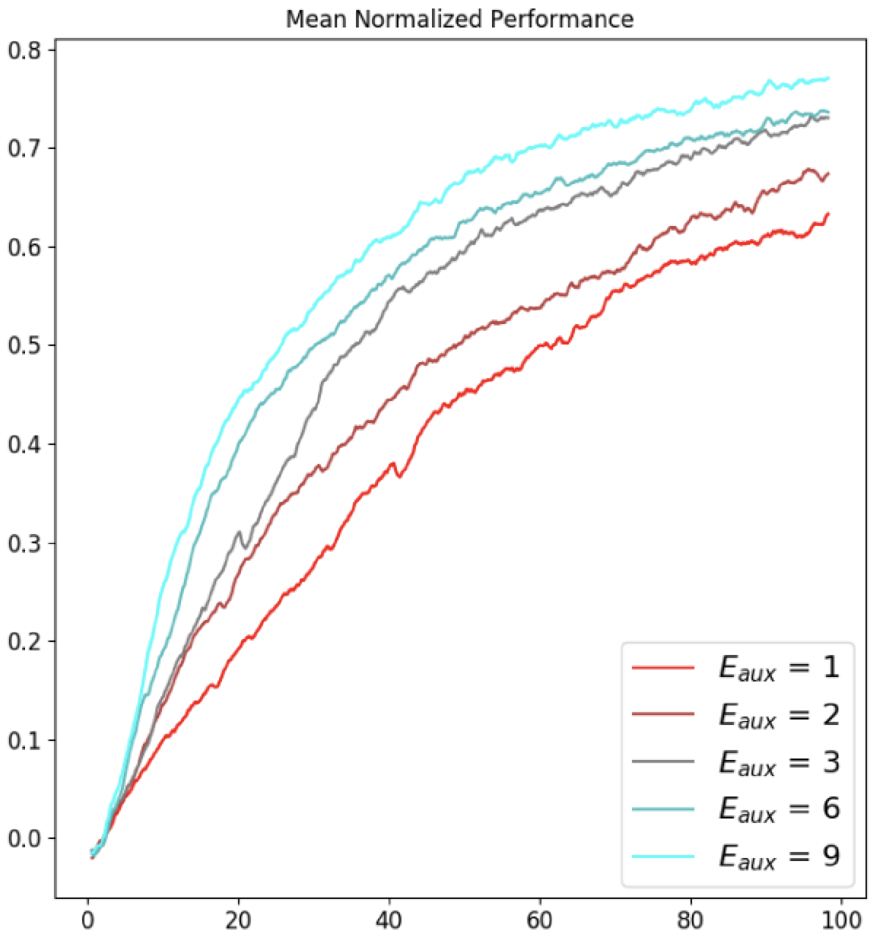
Value Sample Reuse

Expectation before the experiment:
A trade-off between overfitting and slow training.

More sample reuse during the auxiliary phase is beneficial (performance tapering off around 6 epochs).
E_{aux} chose to be 6 to prevent overfitting and decrease computational cost.

Additional epochs offer 2 benefits;

- L^{joint} → Better trained features are shared with the policy
- L^{value} → More accurate value function → reduce the variance of the policy gradient in future policy phase.



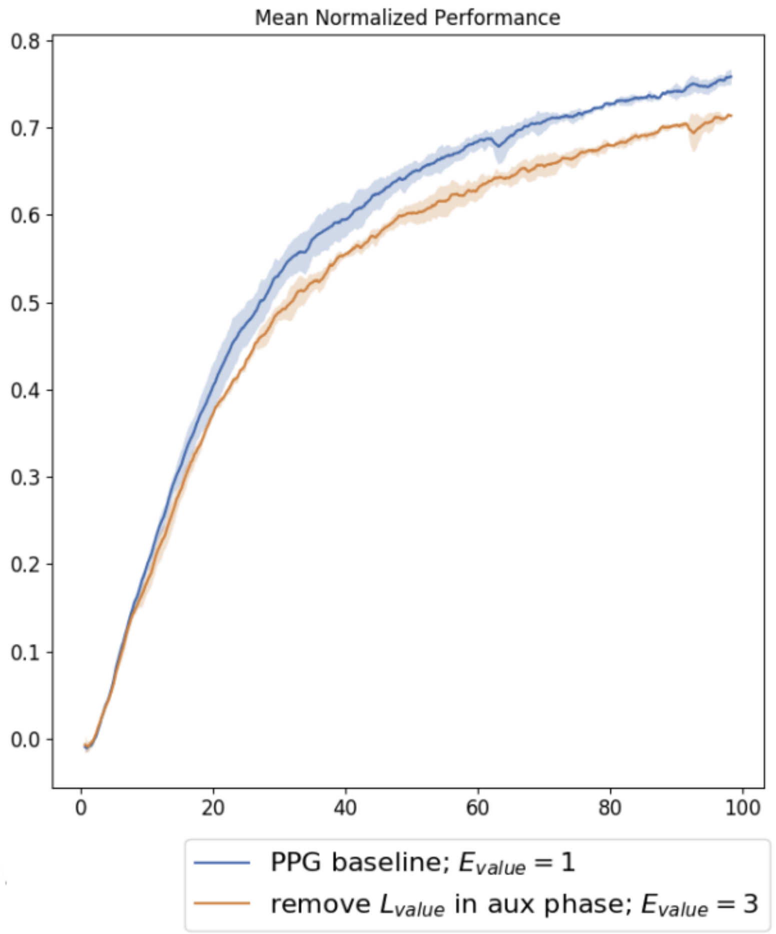
Auxiliary Phase Value Function Training

There are 2 ways to train the value function network.

One in Policy Phase and another in Auxiliary Phase.

We can remove L_{value} in auxiliary phase and adjust E_{value}.

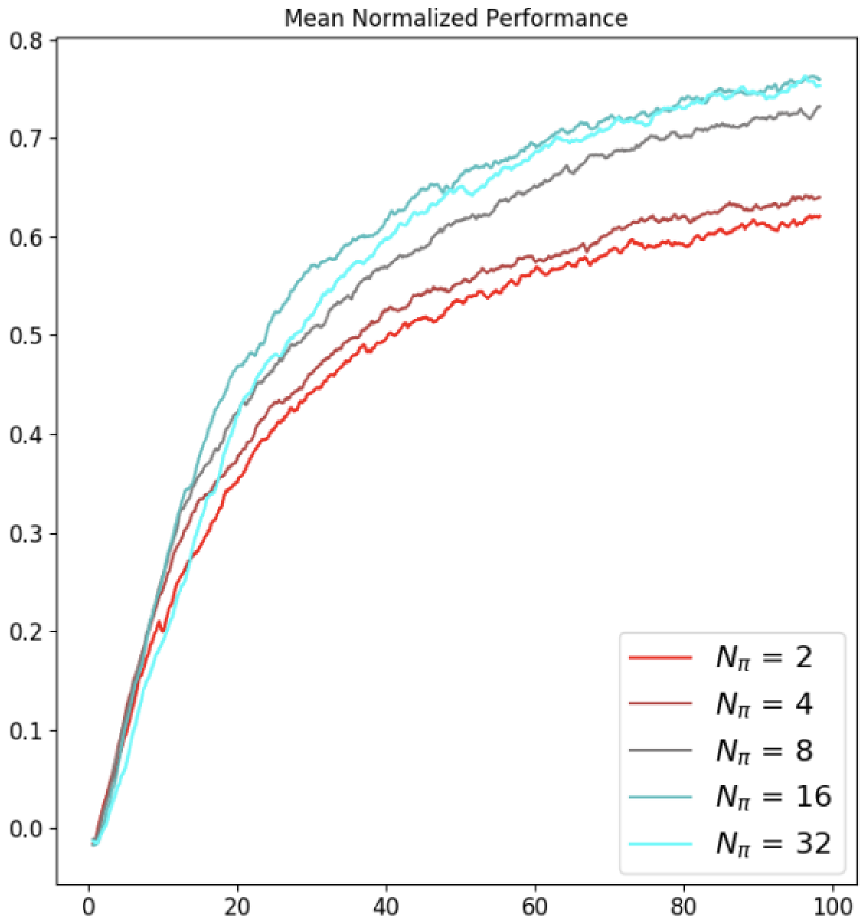
The performance barely suffers



Auxiliary Phase Frequency

Better performance for less frequent auxiliary phase.

Each auxiliary phase interferes with policy optimization, and performing frequent auxiliary phases exacerbates this effect.

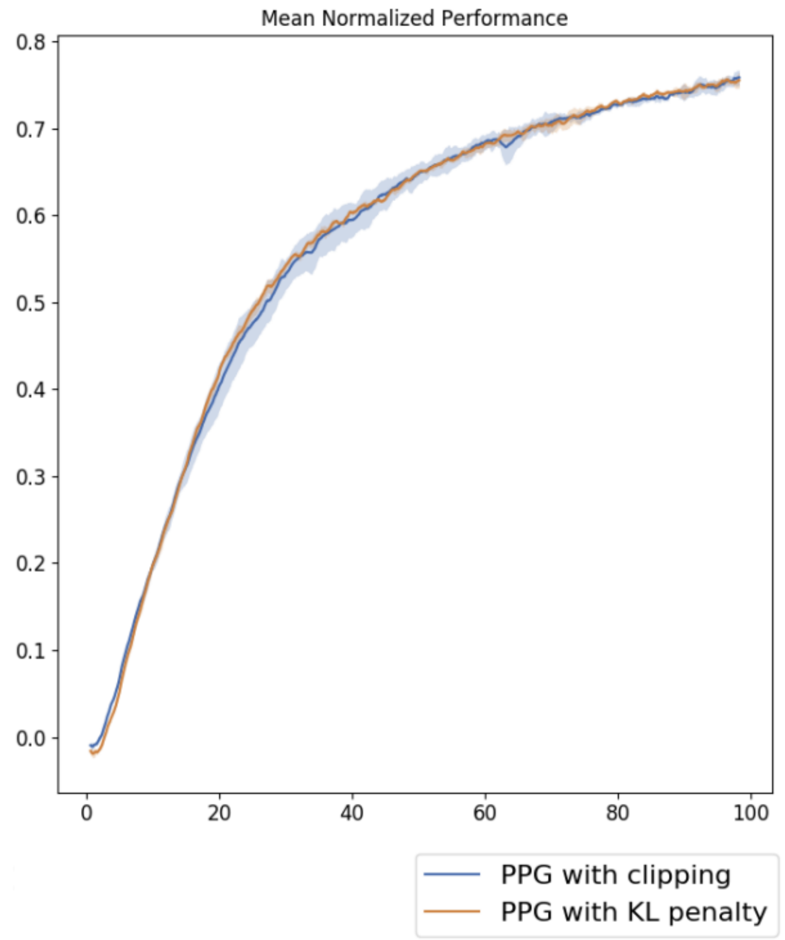


KL vs clipping

PPO provides 2 surrogate objective functions;

- Clipped Surrogate Objective
- Adaptive KP Penalty Coefficient

The results of the two methods are very similar with PPG.



Single-Network PPG

By default, PPG uses 2x memory as PPO.

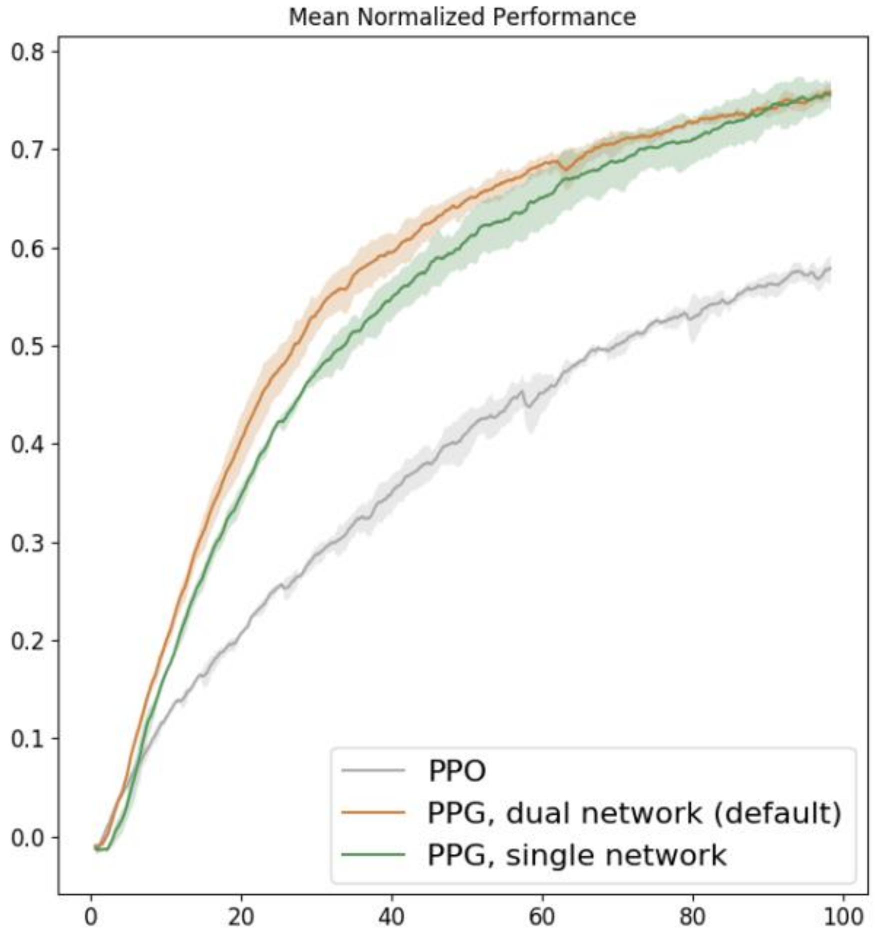
Single PPG halves memory footprint with only slight drop in performance.

Key Idea: detach value function gradient (at the last layer) during policy phases.

Both variants of PPG:

- Have no policy gradient interference
- Benefit from sharing representations

Single Network PPG uses less wall clock time.



Questions?