ESE 6510 Notes: TRPO, PPO

Jefferson Ng

February 23, 2026

1 Trust Region Policy Optimization (TRPO)

1.1 Policy gradient surrogate objective

Recall that the policy gradient objective is

$$\nabla_\theta J(\theta) = \mathbb{E}_{s_t, a_t \sim \pi_\theta}[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\, A^{\pi_\theta}(s_t, a_t)]\,.$$

Here the advantage function is defined as

$$A^{\pi_\theta}(s, a) = Q^{\pi_\theta}(s, a) - V^{\pi_\theta}(s),$$

In practice, we do not know all the complete A^{π_θ} of all policy parameters because it depends on the expectations of the unknown environment. Pay close attention to the distribution under the expectation — the gradient is defined relative to the state–action distribution generated by the **current policy**. Since we only have access to the current policy, we must estimate $\hat{A}_t$ for the current weight from its own rollouts. Recall that we can estimate the gradient from these trajectories with Monte Carlo estimator

$$\nabla_\theta L^{\mathrm{PG}}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \sum_{t} \nabla_\theta \log \pi_\theta\big(a_t^{(i)} \mid s_t^{(i)}\big)\, \hat{A}_t^{(i)}.$$

At $\theta = \theta_{\mathrm{old}}$, the sampling distribution used to compute this expectation matches the policy gradient of the objective. Therefore,

$$\nabla_\theta L^{\mathrm{PG}}(\theta_{\mathrm{old}}) = \nabla_\theta J(\theta_{\mathrm{old}}).$$

The policy gradient update

$$\theta_{\mathrm{new}} = \theta_{\mathrm{old}} + \alpha\, \nabla_\theta \widehat{L^{\mathrm{PG}}(\theta_{\mathrm{old}})}$$

is therefore performing ascent on a local linear approximation of J. Unlike θ_{old}, the surrogate L^{PG} does not need to match the true objective. The image below illustrates that if the gradient step is too large it diverges from the true objective.

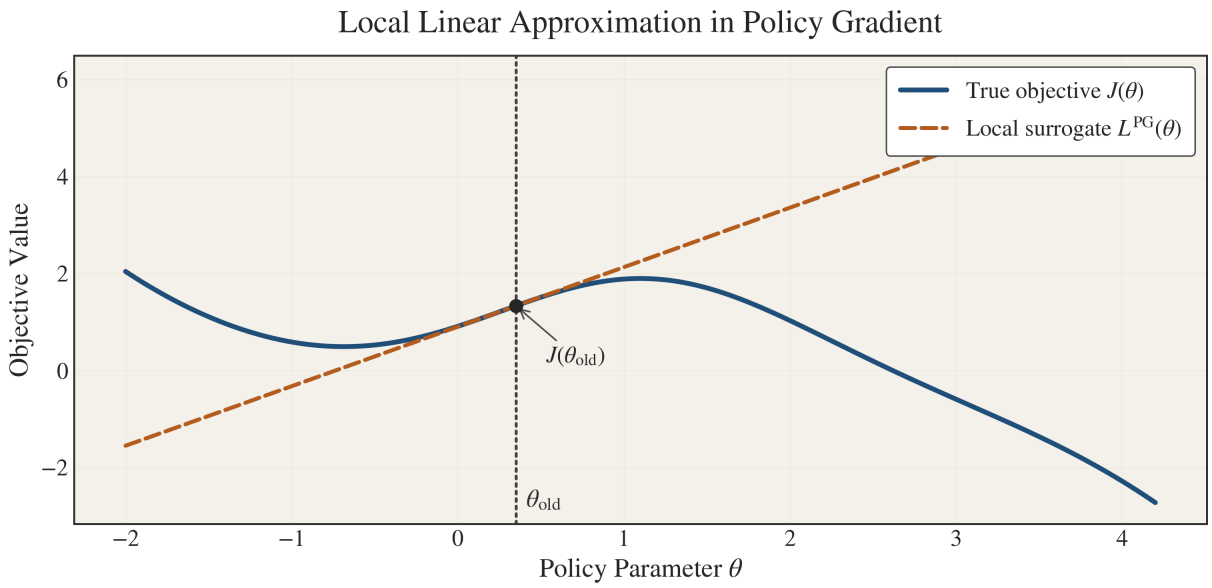

Figure 1: The curved line represents the true objective $J(\theta)$, while the dashed line is its first-order approximation at θ_{old}. At θ_{old}, the surrogate and true objective share the same value and gradient. However, as θ moves away from θ_{old}, the linear surrogate may diverge from the true objective and mispredict improvement. This illustrates the fundamentally local nature of policy gradient updates.

1.2 Importance sampling and policy ratios

As shown above, the policy gradient update relies on data generated by the current policy. However, once the parameters move from θ_{old} to a new θ, the sampling distribution no longer matches the distribution assumed in the gradient theorem. To correct this mismatch, we use a classical tool from probability: importance sampling.

Importance sampling (probability view)

Importance sampling allows us to rewrite an expectation under one distribution using samples from another.
Let $p(x)$ and $q(x)$ be densities such that $q(x) > 0$ whenever $p(x) > 0$. Then

$$\mathbb{E}_{x\sim p}[f(x)] = \int f(x)p(x)\,dx = \int f(x)\frac{p(x)}{q(x)}\,q(x)\,dx = \mathbb{E}_{x\sim q}\left[f(x)\frac{p(x)}{q(x)}\right].$$

The ratio

$$w(x) = \frac{p(x)}{q(x)}$$

is called the *importance weight*. It reweights samples drawn from q so that, in expectation, they behave as if drawn from p.

Starting from the policy gradient objective,

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim p_\theta}[\nabla_\theta \log p_\theta(\tau)\, R(\tau)]\,,$$

where the trajectory distribution factorizes as

$$p_\theta(\tau) = \rho(s_0) \prod_{t=0}^{T-1} \pi_\theta(a_t \mid s_t)\, P(s_{t+1} \mid s_t, a_t),$$

and therefore

$$\log p_\theta(\tau) = \sum_{t=0}^{T-1} \log \pi_\theta(a_t \mid s_t).$$

Applying importance sampling to express the gradient using data collected from $\pi_{\theta_{\text{old}}}$,

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim p_{\theta_{\text{old}}}} \left[\frac{p_\theta(\tau)}{p_{\theta_{\text{old}}}(\tau)} \nabla_\theta \log p_\theta(\tau)\, R(\tau) \right].$$

Using trajectory factorization,

$$\frac{p_\theta(\tau)}{p_{\theta_{\text{old}}}(\tau)} = \prod_{t=0}^{T-1} \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}.$$

Thus, the importance-sampled form of the policy gradient becomes

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim \pi_{\theta_{\text{old}}}} \left[\frac{\pi_\theta(\tau)}{\pi_{\theta_{\text{old}}}(\tau)} \nabla_\theta \log \pi_\theta(\tau)\, R(\tau) \right].$$

TRPO now introduces a key shift. Rather than optimizing this gradient expression directly, it constructs an auxiliary objective that matches the true policy gradient at the current policy while being simpler to optimize using fixed data. We first define the per-step likelihood ratio

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

The key identity underlying the construction is

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

Therefore,

$$\nabla_\theta r_t(\theta)\Big|_{\theta=\theta_{\text{old}}} = \nabla_\theta \log \pi_\theta(a_t|s_t)\Big|_{\theta_{\text{old}}},$$

since $r_t(\theta_{\text{old}}) = 1$.
This shows that, locally around θ_{old}, an objective built from $r_t(\theta)$ has the same first-order behavior as the true objective.

1.3 Trust Region Policy Optimization (TRPO)

The surrogate objective above matches the true policy gradient at θ_{old}, but this agreement is only *local*. It relies on a first-order approximation, and therefore becomes unreliable if the new policy moves too far from the old one. In other words, although $L^{\text{IS}}(\theta)$ provides a direction of improvement, it does not guarantee that large updates will increase the true objective $J(\theta)$. This naturally raises a central question:

How far can we update the policy before the surrogate becomes unreliable?

A natural first idea would be to restrict the parameter change directly:

$$\|\theta - \theta_{\text{old}}\| \leq \varepsilon.$$

However, a small change in parameters does not necessarily imply a small change in the induced trajectory distribution:

$$\|\theta - \theta_{\text{old}}\| \leq \varepsilon \;\not\Rightarrow\; p(\tau;\theta) \approx p(\tau;\theta_{\text{old}}).$$

Instead, TRPO constrains the change in the *policy distribution* using the KL divergence at each state:

$$D_{\text{KL}}(\pi_{\theta_{\text{old}}}(\cdot \mid s) \parallel \pi_\theta(\cdot \mid s)).$$

Rather than limiting parameter movement directly, TRPO limits how far the new policy is allowed to deviate from the old one in distribution space. This can be enforced in two equivalent ways: either as a hard constraint or as a soft penalty.

Form 1 (Constrained TRPO).

$$\max_\theta \quad \mathbb{E}_t\big[r_t(\theta)\hat{A}_t\big] \quad \text{s.t.} \quad \mathbb{E}_t[D_{\text{KL}}(\pi_{\theta_{\text{old}}}(\cdot \mid s_t) \parallel \pi_\theta(\cdot \mid s_t))] \leq \varepsilon.$$

Here, the policy update is explicitly restricted to remain within a KL-radius ε of the old policy.

Form 2 (Penalized TRPO).

$$\max_\theta \quad \mathbb{E}_t\big[r_t(\theta)\hat{A}_t\big] - \beta\, \mathbb{E}_t[D_{\text{KL}}(\pi_{\theta_{\text{old}}}(\cdot \mid s_t) \parallel \pi_\theta(\cdot \mid s_t))],$$

where $\beta > 0$ is a Lagrange multiplier.
In this formulation, the KL term is incorporated directly into the objective as a regularizer. By tuning β, the penalty enforces the same trust-region constraint implicitly. For a suitable choice of β, the penalized and constrained formulations are equivalent.

$$L_1^{\text{TRPO}} = L_2^{\text{TRPO}}.$$

A stricter variant replaces the expected KL constraint with a maximum KL constraint across states:

$$\max_s D_{\text{KL}}(\pi_{\theta_{\text{old}}}(\cdot \mid s) \parallel \pi_\theta(\cdot \mid s)) \leq \varepsilon.$$

Under mild regularity assumptions, this yields a monotonic improvement guarantee:

$$J(\theta_{\text{new}}) \geq J(\theta_{\text{old}}).$$

TRPO is therefore a trust-region method: it uses a second-order approximation of the KL constraint, leading to a natural-gradient style update.

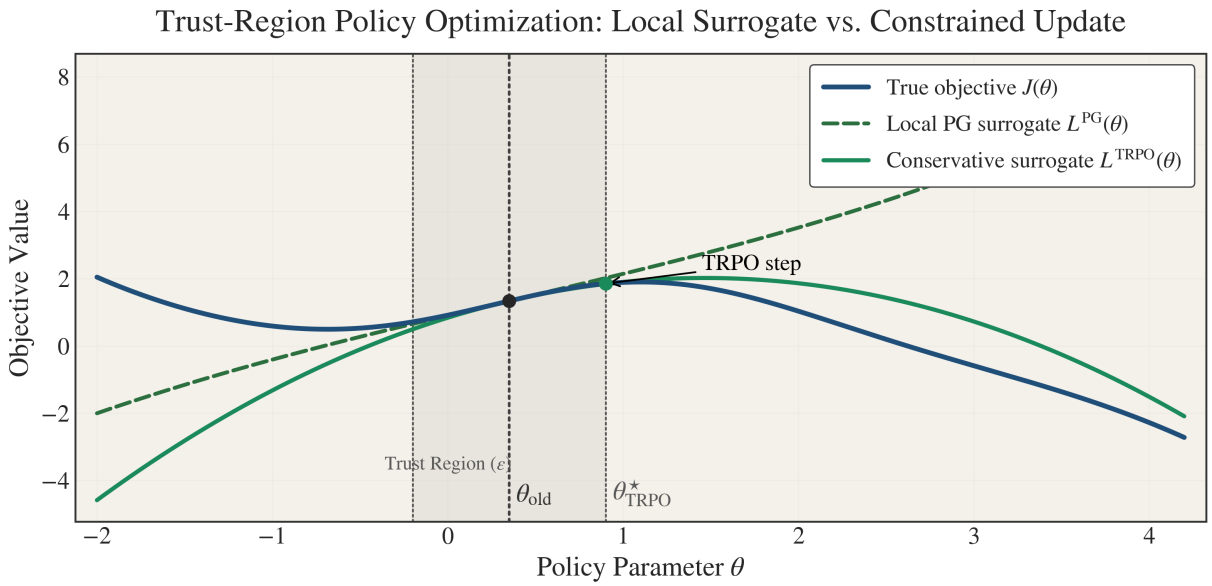

Figure 2: Illustration of Trust-Region Policy Optimization (TRPO). The true objective $J(\theta)$ is shown in blue, while the local surrogate used by policy gradient methods approximates it near θ_{old}. Although the surrogate matches the true objective in value and gradient at θ_{old}, it can mispredict improvement if the update is too large. TRPO addresses this by introducing a trust region, defined via a KL-divergence constraint, which restricts the new policy to remain close to the old one. The conservative surrogate $L^{\text{TRPO}}(\theta)$ therefore produces a controlled update $\theta^{\star}_{\text{TRPO}}$ that stays within this region, avoiding overly aggressive steps that could degrade true performance.

Why not always use TRPO?

1. Second-order method. TRPO relies on a quadratic approximation of the KL constraint, which introduces Fisher matrix computations and natural-gradient updates. This makes the method conceptually and implementation-wise more involved.

2. Computational overhead. Each update requires conjugate-gradient iterations and a line search to enforce the KL constraint, increasing training time and engineering complexity.

3. Practical robustness. Although TRPO provides a theoretical monotonic improvement guarantee, in practice simpler first-order methods (e.g., PPO) often achieve comparable or better empirical performance with significantly less complexity.

2 Proximal Policy Optimization (PPO)

TRPO enforces a hard trust region via a KL constraint. PPO instead constructs a modified first-order objective that implicitly discourages large updates while remaining simple to implement.

2.1 Clipped Surrogate Objective

Recall that after importance sampling the policy gradient surrogate becomes

$$L^{\mathrm{PG}}(\theta) = \mathbb{E}_t\Big[r_t(\theta)\,\hat{A}_t\Big], \qquad r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\mathrm{old}}}(a_t \mid s_t)}.$$

The ratio $r_t(\theta)$ measures how much the new policy changes the probability of the *sampled action a_t at state s_t* relative to the old policy.

$$r_t = 1 \quad \Longleftrightarrow \quad \pi_\theta(a_t \mid s_t) = \pi_{\theta_{\mathrm{old}}}(a_t \mid s_t).$$

If $r_t = 1.2$, the new policy assigns 20% higher probability to that specific action. If $r_t = 0.8$, it assigns 20% lower probability.

The vanilla surrogate

$$r_t\hat{A}_t$$

is linear in r_t. If $\hat{A}_t > 0$, increasing the probability of this advantageous action always increases the objective. If $\hat{A}_t < 0$, decreasing its probability always increases the objective. Nothing limits how far the optimizer may push probabilities.

PPO modifies the objective to

$$L^{\mathrm{CLIP}}(\theta) = \mathbb{E}_t\left[\min\Big(\underbrace{r_t(\theta)\hat{A}_t}_{\text{standard PG improvement}}, \underbrace{\mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t}_{\text{bounded improvement}}\Big)\right].$$

For a single timestep, define the per-sample objective

$$\ell_t(\theta) = \min\Big(r_t(\theta)\hat{A}_t,\ \mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\Big).$$

The clipping operator

$$\mathrm{clip}(\rho, 1-\epsilon, 1+\epsilon)$$

forces the ratio to remain within

$$1-\epsilon \le r_t \le 1+\epsilon.$$

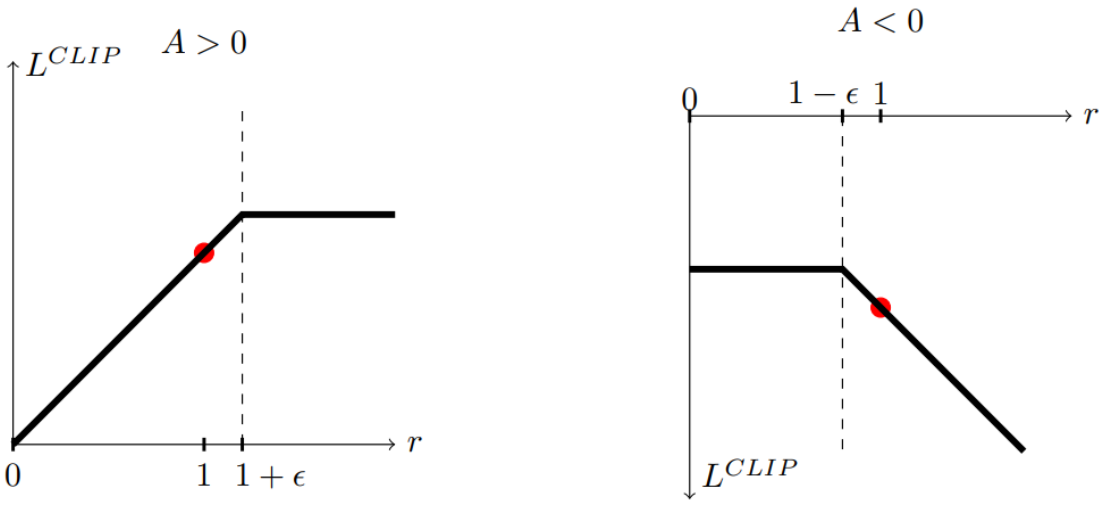

Figure 3: **PPO clipped surrogate objective.** Per-sample objective as a function of the probability ratio r_t. The dashed line is the unclipped surrogate $r_t\hat{A}_t$, and the solid line is the clipped objective $\min(r_t\hat{A}_t,\ \text{clip}(r_t, 1-\epsilon, 1+\epsilon)\hat{A}_t)$. For $\hat{A}_t > 0$ (left), the objective increases with r_t and saturates at $r_t = 1+\epsilon$. For $\hat{A}_t < 0$ (right), it improves as r_t decreases and saturates at $r_t = 1-\epsilon$. Adapted from Schulman et al., 2017.

Clipping acts on the *ratio* r_t, not directly on the product $r_t\hat{A}_t$. When $\hat{A}_t < 0$, multiplying by a negative number reverses inequalities, so the clipped expression can sometimes be larger than $r_t\hat{A}_t$. The $\min(\cdot,\cdot)$ ensures the clipped objective never exceeds the original surrogate.

Numerical illustration (positive advantage).
Let $\hat{A}_t = 2$ and $\epsilon = 0.2$. Suppose the new policy increases the probability of the sampled action by 30%, so $r_t = 1.3$.

$$\text{Unclipped: } r_t\hat{A}_t = 1.3 \times 2 = 2.6.$$

$$\text{Clipped: } (1+\epsilon)\hat{A}_t = 1.2 \times 2 = 2.4.$$

$$\ell_t(\theta) = \min(2.6, 2.4) = 2.4.$$

Although the raw surrogate suggests a larger improvement (2.6), PPO only credits 2.4. The extra gain is ignored.

Numerical illustration (negative advantage).
Let $\hat{A}_t = -2$ and $\epsilon = 0.2$. Suppose the new policy decreases the probability too aggressively, so $r_t = 0.6$.

$$\text{Unclipped: } r_t\hat{A}_t = 0.6 \times (-2) = -1.2.$$

$$\text{Clipped: } (1-\epsilon)\hat{A}_t = 0.8 \times (-2) = -1.6.$$

$$\ell_t(\theta) = \min(-1.2, -1.6) = -1.6.$$

Without the min, the clipped term would appear artificially better. Taking the minimum ensures the modified objective is always a conservative (lower) estimate of improvement.

Thus PPO approximates trust-region behavior without explicitly computing KL divergence. Instead of constraining the entire policy distribution, it directly limits how much each sampled action probability can change relative to the old policy. In practice, this first-order mechanism is sufficient to stabilize training while remaining simple to implement.

2.2 KL-Based Control and Adaptive Updates

In addition to clipping, the original PPO paper also proposes a KL-penalty variant:

$$L^{\mathrm{KLPEN}}(\theta) = \mathbb{E}_t\Big[r_t(\theta)\hat{A}_t - \beta D_{\mathrm{KL}}\big(\pi_{\theta_{\mathrm{old}}}(\cdot \mid s_t) \parallel \pi_\theta(\cdot \mid s_t)\big)\Big].$$

Here $\beta > 0$ acts as a soft Lagrange multiplier. Large deviations from the old policy are penalized directly through KL divergence. This resembles TRPO conceptually, but the optimization mechanics differ. TRPO solves a constrained optimization problem

$$\max_\theta \ \mathbb{E}_t\big[r_t(\theta)\hat{A}_t\big] \quad \text{s.t.} \quad \mathbb{E}_t\big[D_{\mathrm{KL}}\big(\pi_{\theta_{\mathrm{old}}}(\cdot \mid s_t) \,\|\, \pi_\theta(\cdot \mid s_t)\big)\big] \le \delta,$$

and enforces the constraint using a second-order Taylor expansion of the KL, which yields a quadratic constraint involving the Fisher information matrix. and enforces the constraint using a second-order Taylor expansion of KL, leading to a quadratic form involving the Fisher information matrix. The resulting update requires solving a constrained problem with natural-gradient style steps. In contrast, the KL-penalty version of PPO simply adds $-\beta D_{\mathrm{KL}}$ to the objective and optimizes it using standard first-order gradient methods. No Fisher matrix inversion or second-order solve is performed.

After each update, the empirical KL divergence is measured:

$$d = \mathbb{E}_t\big[D_{\mathrm{KL}}\big(\pi_{\theta_{\mathrm{old}}} \parallel \pi_\theta\big)\big].$$

The penalty coefficient can then be adjusted to target a desired divergence:

$$\beta \leftarrow \begin{cases} \beta/2, & d < D_{\mathrm{target}}/1.5, \\ 2\beta, & d > 1.5 D_{\mathrm{target}}, \\ \beta, & \text{otherwise.} \end{cases}$$

If the policy moves too much, increase the penalty. If it moves too little, decrease the penalty. This forms a feedback controller on update magnitude.

Even when using the clipped objective, KL can still be monitored and used to adapt the learning rate. A generic PPO update is

$$\theta_{\mathrm{new}} = \theta_{\mathrm{old}} + \alpha \, \nabla_\theta \widehat{L^{\mathrm{CLIP}}}(\theta).$$

The step size α may also be adjusted based on the measured KL:

$$\alpha_{\text{new}} = \begin{cases} \alpha/2, & d > D_{\text{target}}, \\ 2\alpha, & d < D_{\text{target}}, \\ \alpha, & \text{otherwise.} \end{cases}$$

Although the adaptive rules for β and α appear structurally similar, they operate at different levels.
Adjusting β modifies the objective function itself:

$$\nabla_\theta L^{\text{KLPEN}} = \nabla_\theta(r_t \hat{A}_t) - \beta \nabla_\theta D_{\text{KL}}.$$

Thus increasing β changes the *direction* of the gradient, placing more weight on staying close to the old policy. In contrast, adjusting the learning rate α does not change the objective or the gradient direction. It only rescales the step size in parameter space. Both mechanisms use KL divergence as a feedback signal, but β performs objective-level regularization, whereas α performs optimizer-level step control.

Thus PPO offers multiple first-order mechanisms to control update size:

- Clipping probability ratios (objective-level control),
- Penalizing KL divergence with adaptive β,
- Adjusting learning rate based on KL.

All of these approximate the trust-region behavior of TRPO, but avoid second-order optimization.

2.3 Implementation Details

Parallel actors. Data is collected from N environments in parallel, then pooled for optimization.
Early termination. If an episode ends before horizon T, the environment is reset. Since the objective decomposes over timesteps, this does not bias the gradient estimator.
Multiple epochs. Each batch is reused for several SGD passes. Clipping (or KL monitoring) prevents divergence.
Entropy regularization. An entropy bonus encourages exploration:

$$L_{\text{total}} = L^{\text{CLIP}} + c_H \mathbb{E}_t\big[H(\pi_\theta(\cdot \mid s_t))\big].$$

Value function loss. A squared-error loss trains the critic:

$$L_V = \mathbb{E}_t\big[(V_\theta(s_t) - \hat{V}_t)^2\big].$$

2.4 Extensions

MAPPO vs. IPPO.
IPPO trains each agent independently with its own policy and value function.
MAPPO uses decentralized policies but a centralized critic conditioned on joint state or joint actions, reducing non-stationarity in cooperative settings.

Self-play.
Policies are trained against previous checkpoints, creating an automatic curriculum as performance improves.

Population-Based Training (PBT).
Maintain a population $\{(\theta^{(j)}, h^{(j)})\}_{j=1}^{M}$. Periodically:

- Copy parameters from strong to weak members,
- Mutate hyperparameters.

This performs online hyperparameter search coupled to training.