A Study of Policy Optimization in Reinforcement Learning

CS 839 Project Report
University of Wisconsin-Madison

Dheeraj Dhillon

May 7, 2026

1 Introduction

Policy optimization methods learn a parameterized policy directly. Instead of first estimating an action-value function and then extracting actions from it, they optimize a policy $\pi_\theta(a|s)$ so that the expected discounted return is large. For an initial state distribution μ_0, we have the optimization

$$J(\pi_\theta) := \mathbb{E}_{s_0\sim\mu_0,\, a_t\sim\pi_\theta(\cdot|s_t)}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\right], \qquad \max_{\theta\in\Theta} J(\pi_\theta).$$

This direct parameterization is especially natural for stochastic policies and continuous action spaces. In such settings, enumerating actions or applying a maximization operator inside a value-based method can be awkward or intractable.

The basic tool behind these methods is the policy-gradient identity. REINFORCE [Williams, 1992] gives a trajectory-level likelihood-ratio estimator: if $R(\tau)=\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)$, then

$$\begin{aligned}
\nabla_\theta J(\pi_\theta) &= \mathbb{E}_{\tau\sim\pi_\theta}\left[R(\tau)\sum_{t=0}^{\infty}\nabla_\theta\log\pi_\theta(a_t|s_t)\right] && \text{(REINFORCE)}\\
&= \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^{\pi_\theta},\, a\sim\pi_\theta(\cdot|s)}\left[Q^{\pi_\theta}(s,a)\nabla_\theta\log\pi_\theta(a|s)\right] && \text{(action-value form)}\\
&= \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^{\pi_\theta},\, a\sim\pi_\theta(\cdot|s)}\left[A^{\pi_\theta}(s,a)\nabla_\theta\log\pi_\theta(a|s)\right]. && \text{(advantage form)}
\end{aligned}$$

The first line is the trajectory likelihood-ratio form. The latter two are the discounted-visitation forms of the policy-gradient theorem [Sutton et al., 1999]. They express the gradient using the discounted state visitation distribution d^{π_θ}, action values, and advantages, without requiring an explicit derivative of the state distribution. The advantage form also makes clear why state-value baselines are useful: they can reduce variance without changing the expected gradient.

1.1 Vanilla Policy Gradients

Vanilla policy-gradient methods are appealing because they give a direct first-order ascent direction for the policy parameters and work naturally with differentiable function approximators. In practice, however, the basic estimator is often too noisy and too local to be a reliable policy optimization method by itself. Estimates based on sampled returns can have high variance,

learning can be sample inefficient, and the data distribution changes as the policy changes. These issues make performance improvement sensitive to step size: small steps may learn slowly, while large steps can move the policy into a region where the local gradient estimate is no longer reliable.

The papers studied in this report can be viewed as responses to this problem. Conservative Policy Iteration (CPI) [Kakade and Langford, 2002] asks how to improve a policy while taking only a controlled mixture step toward a candidate policy. Natural Policy Gradient (NPG) [Kakade, 2001] changes the geometry of the update so that step size is measured in policy space rather than raw parameter space. Trust Region Policy Optimization (TRPO) [Schulman et al., 2015] combines a local surrogate objective with an explicit trust-region constraint, and Proximal Policy Optimization (PPO) [Schulman et al., 2017] turns the same surrogate viewpoint into a simpler first-order algorithm. The goal of this report is to study these methods as a single theoretical progression: from comparing two policies, to conservative improvement, to geometry-aware and trust-region policy optimization.

2 Comparing Policy Updates Through Performance Difference

A recurring idea behind CPI, TRPO, and PPO is to evaluate a candidate policy through quantities defined relative to the current policy. This policy-comparison viewpoint is central to *Approximately Optimal Approximate Reinforcement Learning* [Kakade and Langford, 2002], which introduced the conservative policy iteration framework and supplied the performance-difference and surrogate-objective machinery that later appears in TRPO and PPO. Instead of asking directly whether $\tilde{\pi}$ is better than π, we express the performance difference using the advantage function of π. This is the role of the performance difference lemma, and it leads naturally to the local surrogate objective. Together, these tools isolate the central difficulty of policy optimization: changing the policy changes both the action distribution at each state and the future state distribution induced by the policy.

Let $J(\pi)$ denote the discounted performance under initial distribution μ_0, as defined in the Introduction. For a fixed policy π, we write $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$, and define the normalized discounted state visitation distribution by $d_\pi(s) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t \Pr(s_t = s \mid \pi)$, where the common dependence on μ_0 is omitted for simplicity. With this normalization, d_π is a probability distribution over states. The TRPO paper uses the unnormalized discounted visitation frequency $\rho_\pi(s) = \sum_{t=0}^{\infty}\gamma^t \Pr(s_t = s \mid \pi)$. Throughout this report, we use the normalized version $d_\pi = (1-\gamma)\rho_\pi$. Therefore, formulas written with d_π include an explicit factor of $1/(1-\gamma)$ compared with the corresponding unnormalized ρ_π notation.

2.1 Performance Difference Lemma

The performance difference lemma gives an exact expression for the improvement of a candidate policy $\tilde{\pi}$ over a reference policy π:

$$J(\tilde{\pi}) - J(\pi) = \mathbb{E}_{\tau\sim\tilde{\pi}}\left[\sum_{t=0}^{\infty}\gamma^t A^\pi(s_t,a_t)\right]. \tag{1}$$

Grouping the same expectation by the states visited under $\tilde{\pi}$ gives the state-visitation form

$$J(\tilde{\pi}) = J(\pi) + \sum_{t=0}^{\infty}\sum_{s}\Pr(s_t = s \mid \tilde{\pi})\sum_{a}\tilde{\pi}(a|s)\gamma^t A^\pi(s,a)$$

$$= J(\pi) + \frac{1}{1-\gamma} \sum_s d_{\tilde{\pi}}(s) \sum_a \tilde{\pi}(a|s) A^{\pi}(s,a). \tag{2}$$

The identity is useful because the new policy is evaluated using the old policy's advantage function. Thus, the question becomes whether the actions chosen by $\tilde{\pi}$ tend to have positive advantage under π.

For completeness, we prove the trajectory form. Using $Q^{\pi}(s_t, a_t) = \mathbb{E}[r(s_t, a_t) + \gamma V^{\pi}(s_{t+1}) \mid s_t, a_t]$, we have

$$\begin{aligned}
\mathbb{E}_{\tau\sim\tilde{\pi}}\left[\sum_{t=0}^{\infty} \gamma^t A^{\pi}(s_t, a_t)\right] &= \mathbb{E}_{\tau\sim\tilde{\pi}}\left[\sum_{t=0}^{\infty} \gamma^t \left(Q^{\pi}(s_t, a_t) - V^{\pi}(s_t)\right)\right] \\
&= \mathbb{E}_{\tau\sim\tilde{\pi}}\left[\sum_{t=0}^{\infty} \gamma^t \left(r(s_t, a_t) + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t)\right)\right] \\
&= J(\tilde{\pi}) + \mathbb{E}_{\tau\sim\tilde{\pi}}\left[\sum_{t=0}^{\infty} \left(\gamma^{t+1} V^{\pi}(s_{t+1}) - \gamma^t V^{\pi}(s_t)\right)\right] \\
&= J(\tilde{\pi}) + \mathbb{E}_{\tau\sim\tilde{\pi}}\left[\sum_{t=1}^{\infty} \gamma^t V^{\pi}(s_t) - \sum_{t=0}^{\infty} \gamma^t V^{\pi}(s_t)\right] \\
&= J(\tilde{\pi}) - \mathbb{E}_{s_0\sim\mu_0}[V^{\pi}(s_0)] = J(\tilde{\pi}) - J(\pi).
\end{aligned} \tag{3}$$

This proves Equation (1).

2.2 Connection to Policy Iteration

The state-visitation form immediately recovers the policy-improvement idea behind exact policy iteration. If a candidate policy $\tilde{\pi}$ satisfies $\sum_a \tilde{\pi}(a|s) A^{\pi}(s,a) \geq 0$ for all s, then Equation (2) implies $J(\tilde{\pi}) \geq J(\pi)$. In particular, let $a_{\text{greedy}}(s) \in \arg\max_a Q^{\pi}(s,a)$, and let π_{greedy} place all probability on this action. Since $V^{\pi}(s)$ does not depend on a, $a_{\text{greedy}}(s) \in \arg\max_a A^{\pi}(s,a)$. Therefore, $\sum_a \pi_{\text{greedy}}(a|s) A^{\pi}(s,a) = \max_a A^{\pi}(s,a)$. Because $\sum_a \pi(a|s) A^{\pi}(s,a) = 0$, the maximum advantage is nonnegative at every state. Hence

$$J(\pi_{\text{greedy}}) - J(\pi) = \frac{1}{1-\gamma} \sum_s d_{\pi_{\text{greedy}}}(s) \max_a A^{\pi}(s,a) \geq 0. \tag{4}$$

In approximate settings this argument becomes more delicate: estimation error, function approximation, or a restricted policy class may produce negative expected advantage in some states. This is why the later algorithms do not simply replace π by a greedy candidate; they try to improve the policy while controlling how far the update moves.

2.3 The Local Surrogate Objective

Although Equation (2) is exact, it is difficult to optimize directly because the candidate policy $\tilde{\pi}$ appears in two places. It appears directly through $\tilde{\pi}(a|s)$, and indirectly through the state visitation distribution $d_{\tilde{\pi}}$. The latter dependence is the state-distribution shift problem: changing the policy changes not only which actions are selected, but also which states are likely to be visited later.

We therefore introduce a local surrogate objective that freezes the state distribution at the reference policy:

$$L_\pi(\tilde{\pi}) = J(\pi) + \frac{1}{1-\gamma} \sum_s d_\pi(s) \sum_a \tilde{\pi}(a|s) A^\pi(s,a). \tag{5}$$

Compared with Equation (2), this surrogate keeps the direct dependence on the candidate action distribution, but replaces $d_{\tilde{\pi}}$ with d_π. This makes L_π a tractable local proxy for performance improvement using data from the current policy.

The surrogate is not globally equal to the true performance, but it is locally accurate around the reference policy. Consider a differentiable policy class $\{\pi_\theta\}$, and let the reference policy be π_{θ_0}. From the exact identity and the surrogate definition,

$$J(\pi_\theta) - L_{\pi_{\theta_0}}(\pi_\theta) = \frac{1}{1-\gamma} \sum_s \Big(d_{\pi_\theta}(s) - d_{\pi_{\theta_0}}(s) \Big) \sum_a \pi_\theta(a|s) A^{\pi_{\theta_0}}(s,a). \tag{6}$$

At $\theta = \theta_0$, this difference is zero. Moreover, differentiating Equation (6) and evaluating at θ_0, both product-rule terms vanish: one contains $d_{\pi_\theta} - d_{\pi_{\theta_0}}$, and the other contains $\sum_a \pi_{\theta_0}(a|s) A^{\pi_{\theta_0}}(s,a) = 0$. Therefore,

$$L_{\pi_{\theta_0}}(\pi_{\theta_0}) = J(\pi_{\theta_0}), \qquad \nabla_\theta L_{\pi_{\theta_0}}(\pi_\theta)\big|_{\theta=\theta_0} = \nabla_\theta J(\pi_\theta)\big|_{\theta=\theta_0}.$$

Thus the surrogate matches the true objective to first order at the current policy. The rest of the report studies different ways of keeping this local approximation useful after the policy is updated.

3 Conservative Policy Iteration

Conservative Policy Iteration (CPI), introduced in *Approximately Optimal Approximate Reinforcement Learning* [Kakade and Langford, 2002], turns the surrogate viewpoint into an explicit policy-improvement rule. If a candidate policy improves the surrogate under the old state distribution, CPI does not replace the current policy immediately. Instead, it moves partway toward the candidate through a mixture update. The proof below shows how this mixture structure controls state-distribution shift and yields a monotonic improvement guarantee.

Let π denote the current policy. CPI first constructs an approximately greedy policy π', whose purpose is to improve the expected advantage under the current state distribution. In the exact tabular case, this corresponds to choosing actions that maximize $A^\pi(s,a)$ at each state. In approximate settings, π' is computed only approximately, for example by estimating advantages from data and then solving a supervised policy-improvement problem over the chosen policy class. For our purposes, the key quantity is the expected advantage of π' under the current discounted state distribution: $\mathcal{A}_\pi(\pi') := \mathbb{E}_{s\sim d_\pi}\left[\sum_a \pi'(a|s) A^\pi(s,a)\right]$. Since $L_\pi(\bar{\pi}) = J(\pi) + \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_\pi}\left[\sum_a \bar{\pi}(a|s) A^\pi(s,a)\right]$, maximizing the surrogate $L_\pi(\bar{\pi})$ over candidate policies is equivalent to maximizing the expected advantage term above.

CPI then updates the policy by mixing the current policy with π':

$$\pi_\alpha(a|s) = (1-\alpha)\pi(a|s) + \alpha\pi'(a|s), \qquad \alpha \in [0,1]. \tag{7}$$

Here α controls how conservative the update is. If $\alpha = 0$, the policy does not change. If $\alpha = 1$, the new policy is fully replaced by π'. CPI chooses an intermediate value of α to trade off improvement in the surrogate against the possible error caused by changing the state visitation distribution.

3.1 The CPI Performance Lower Bound

We now derive the lower bound that justifies the conservative mixture update. We define the statewise improvement of the candidate policy π' under the old advantage function by $G(s) := \sum_a \pi'(a|s)A^\pi(s,a)$, $\epsilon := \max_s |G(s)|$. Then $\mathcal{A}_\pi(\pi') = \mathbb{E}_{s\sim d_\pi}[G(s)]$ is the surrogate improvement direction introduced above. The constant ϵ controls the largest possible statewise deviation of this improvement term.

Using the performance difference lemma with $\tilde{\pi} = \pi_\alpha$, we have

$$J(\pi_\alpha) - J(\pi) = \frac{1}{1-\gamma} \sum_s d_{\pi_\alpha}(s) \sum_a \pi_\alpha(a|s) A^\pi(s,a). \tag{8}$$

Substituting the mixture update from Equation (7),

$$\begin{aligned} \sum_a \pi_\alpha(a|s) A^\pi(s,a) &= \sum_a \big((1-\alpha)\pi(a|s) + \alpha\pi'(a|s)\big) A^\pi(s,a) \\ &= (1-\alpha) \sum_a \pi(a|s) A^\pi(s,a) + \alpha \sum_a \pi'(a|s) A^\pi(s,a). \end{aligned} \tag{9}$$

The first term is zero because the expected advantage under the current policy is zero: $\sum_a \pi(a|s)A^\pi(s,a) = 0$. Therefore, $\sum_a \pi_\alpha(a|s)A^\pi(s,a) = \alpha G(s)$. Substituting this back into Equation (8), and expanding the discounted state distribution, gives

$$\begin{aligned} J(\pi_\alpha) - J(\pi) &= \frac{1}{1-\gamma} \sum_s d_{\pi_\alpha}(s) \alpha G(s) \\ &= \alpha \sum_{t=0}^{\infty} \gamma^t \sum_s P(s_t = s \mid \pi_\alpha) G(s) \\ &= \alpha \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{s_t\sim\pi_\alpha}[G(s_t)]. \end{aligned} \tag{10}$$

The remaining difficulty is that the expectation in Equation (10) is over states visited by the new mixture policy π_α, whereas the surrogate uses states visited by the old policy π. CPI controls this mismatch using the mixture structure.

One way to sample from π_α is to flip a coin at every time step: with probability $1-\alpha$, choose the action from π, and with probability α, choose the action from π'. Let N_t be the number of actions chosen from π' before time t, and let $\rho_t := \Pr(N_t > 0)$. Since all of the first t choices must come from π for $N_t = 0$,

$$\Pr(N_t = 0) = (1-\alpha)^t, \qquad \rho_t = \Pr(N_t > 0) = 1 - (1-\alpha)^t.$$

Conditioned on $N_t = 0$, the trajectory up to time t is exactly distributed as a trajectory generated by the old policy π. Hence $\mathbb{E}[G(s_t) \mid N_t = 0] = \mathbb{E}_{s_t\sim\pi}[G(s_t)]$.

To make the error term explicit, we define $E_0(t) := \mathbb{E}[G(s_t) \mid N_t = 0]$, $\quad E_1(t) := \mathbb{E}[G(s_t) \mid N_t > 0]$. Using the law of total expectation and $|G(s)| \le \epsilon$, we obtain the key step in the CPI proof:

$$\begin{aligned} \alpha\mathbb{E}_{s_t\sim\pi_\alpha}[G(s_t)] &= \alpha\left((1-\rho_t)E_0(t) + \rho_t E_1(t)\right) = \alpha E_0(t) + \alpha\rho_t\left(E_1(t) - E_0(t)\right) \\ &\ge \alpha E_0(t) - \alpha\rho_t\left|E_1(t) - E_0(t)\right| \ge \alpha E_0(t) - \alpha\rho_t\left(|E_1(t)| + |E_0(t)|\right) \end{aligned}$$

$$\geq \alpha E_0(t) - 2\alpha\epsilon\rho_t = \alpha\mathbb{E}_{s_t\sim\pi}[G(s_t)] - 2\alpha\epsilon\rho_t. \tag{11}$$

The term $2\alpha\epsilon\rho_t$ has two sources: the outside α comes from the mixture update, while the 2ϵ comes from comparing two conditional expectations of a function bounded between $-\epsilon$ and ϵ. Since $\rho_t = 1-(1-\alpha)^t$, Equation (11) becomes $\alpha\mathbb{E}_{s_t\sim\pi_\alpha}[G(s_t)] \geq \alpha\mathbb{E}_{s_t\sim\pi}[G(s_t)] - 2\alpha\epsilon\left(1-(1-\alpha)^t\right)$. Putting this into Equation (10), we obtain

$$\begin{aligned} J(\pi_\alpha) - J(\pi) &\geq \alpha\sum_{t=0}^{\infty}\gamma^t\left(\mathbb{E}_{s_t\sim\pi}[G(s_t)] - 2\epsilon\left(1-(1-\alpha)^t\right)\right) \\ &= \alpha\sum_{t=0}^{\infty}\gamma^t\mathbb{E}_{s_t\sim\pi}[G(s_t)] - 2\alpha\epsilon\sum_{t=0}^{\infty}\gamma^t\left(1-(1-\alpha)^t\right). \end{aligned} \tag{12}$$

The first term can be written using the normalized discounted state distribution:

$$\sum_{t=0}^{\infty}\gamma^t\mathbb{E}_{s_t\sim\pi}[G(s_t)] = \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_\pi}[G(s)] = \frac{1}{1-\gamma}\mathcal{A}_\pi(\pi').$$

For the second term,

$$\begin{aligned} \sum_{t=0}^{\infty}\gamma^t\left(1-(1-\alpha)^t\right) &= \sum_{t=0}^{\infty}\gamma^t - \sum_{t=0}^{\infty}\left(\gamma(1-\alpha)\right)^t \\ &= \frac{1}{1-\gamma} - \frac{1}{1-\gamma(1-\alpha)} \\ &= \frac{\gamma\alpha}{(1-\gamma)(1-\gamma(1-\alpha))}. \end{aligned} \tag{13}$$

Thus,

$$J(\pi_\alpha) - J(\pi) \geq \frac{\alpha}{1-\gamma}\mathcal{A}_\pi(\pi') - \frac{2\gamma\epsilon\alpha^2}{(1-\gamma)(1-\gamma(1-\alpha))}. \tag{14}$$

Since $1-\gamma(1-\alpha) = 1-\gamma+\gamma\alpha \geq 1-\gamma$, we also have the slightly weaker but simpler bound

$$J(\pi_\alpha) - J(\pi) \geq \frac{\alpha}{1-\gamma}\mathcal{A}_\pi(\pi') - \frac{2\gamma\epsilon\alpha^2}{(1-\gamma)^2}. \tag{15}$$

Now we rewrite this bound in terms of the surrogate objective. For the mixture policy,

$$L_\pi(\pi_\alpha) = J(\pi) + \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_\pi}\left[\sum_a \pi_\alpha(a|s)A^\pi(s,a)\right] = J(\pi) + \frac{\alpha}{1-\gamma}\mathcal{A}_\pi(\pi'). \tag{16}$$

Combining Equation (15) with Equation (16), we obtain the CPI lower bound

$$J(\pi_\alpha) \geq L_\pi(\pi_\alpha) - \frac{2\gamma\epsilon}{(1-\gamma)^2}\alpha^2. \tag{17}$$

This is the form that most clearly connects CPI to the generalized update in TRPO: the true performance is lower bounded by a local surrogate objective minus a penalty for moving too far from the current policy. The penalty is quadratic in α, so small mixture updates are favored unless the surrogate improvement is large enough to justify a larger move.

This lower bound is the monotonic-improvement mechanism used by CPI: true performance is bounded below by a surrogate gain minus an explicit movement penalty. It also shows why CPI is algorithmically different from gradient methods: in its idealized form, it requires an approximately greedy policy and a conservative mixing coefficient.

3.2 Monotonic Improvement

The lower bound in Equation (17) gives a direct monotonic improvement argument. Define

$$M_\pi(\pi_\alpha) := L_\pi(\pi_\alpha) - \frac{2\gamma\epsilon}{(1-\gamma)^2}\alpha^2.$$

Then Equation (17) says $J(\pi_\alpha) \geq M_\pi(\pi_\alpha)$. At $\alpha = 0$, the mixture policy is exactly the old policy, $\pi_\alpha = \pi$, and there is no penalty. Therefore,

$$M_\pi(\pi) = L_\pi(\pi) = J(\pi).$$

Consequently, if CPI chooses an update satisfying $M_\pi(\pi_\alpha) \geq M_\pi(\pi)$, then

$$J(\pi_\alpha) \geq M_\pi(\pi_\alpha) \geq M_\pi(\pi) = J(\pi).$$

Thus the sequence of policies generated by repeatedly improving this lower bound is non-decreasing in true performance.

For a fixed candidate policy π', the simplified lower bound is the quadratic function

$$B(\alpha) := \frac{\alpha}{1-\gamma}\mathcal{A}_\pi(\pi') - \frac{2\gamma\epsilon}{(1-\gamma)^2}\alpha^2.$$

The linear term represents the predicted improvement from moving toward π', while the quadratic term penalizes the possible error caused by state distribution shift. If $\mathcal{A}_\pi(\pi') > 0$, then this lower bound is positive for sufficiently small positive α, and its unconstrained maximizer is $\alpha^* = \frac{(1-\gamma)\mathcal{A}_\pi(\pi')}{4\gamma\epsilon}$. In practice this value is clipped to the interval $[0, 1]$. Thus CPI chooses the largest useful mixture step according to the lower bound, rather than greedily replacing the policy.

3.3 From Conservative Mixtures to Trust Regions

The CPI bound is powerful because it gives a monotonic improvement argument, but the argument is tied to the special mixture form $\pi_\alpha = (1-\alpha)\pi + \alpha\pi'$.

This mixture structure makes the proof possible, because the probability of having deviated from π before time t is controlled by $1-(1-\alpha)^t$. For large parameterized policies, however, explicitly forming a mixture with an approximately greedy policy is restrictive. A more flexible update rule should keep the same idea of controlled change without requiring this particular mixture representation.

The mixture update also implies a simple total-variation control:

$$\begin{aligned} D_{\mathrm{TV}}(\pi_\alpha(\cdot|s), \pi(\cdot|s)) &= \frac{1}{2}\sum_a \left|(1-\alpha)\pi(a|s) + \alpha\pi'(a|s) - \pi(a|s)\right| \\ &= \frac{\alpha}{2}\sum_a |\pi'(a|s) - \pi(a|s)| \leq \alpha. \end{aligned} \tag{18}$$

Thus, the mixture coefficient α also bounds the statewise total-variation distance from the old policy. This suggests the TRPO generalization: keep the surrogate objective, but replace the special mixture form with a direct policy-distance constraint [Schulman et al., 2015].

4 Trust Region Policy Optimization

Trust Region Policy Optimization (TRPO) [Schulman et al., 2015] extends the surrogate-based improvement idea from mixture policies to general stochastic policies. The theoretical argument first controls policy movement with statewise total variation and coupling; the practical algorithm then replaces this with an average-KL trust region estimated from sampled trajectories.

4.1 Policy Distance and Coupling

We first define the policy-distance quantity used in the theoretical bound. For distributions p and q over actions, the total variation distance can be defined as $D_{\mathrm{TV}}(p,q) = \frac{1}{2}\sum_i |p_i - q_i|$, and the maximum statewise distance between policies as

$$D_{\mathrm{TV}}^{\max}(\pi,\tilde{\pi}) := \max_s D_{\mathrm{TV}}(\pi(\cdot|s),\tilde{\pi}(\cdot|s))\,.$$

This quantity controls the largest action-distribution change at any state. The reason it is useful is the coupling characterization of total variation: if $D_{\mathrm{TV}}(p,q) \le \alpha$, then one can define random variables (X,Y) with marginals p and q such that $\Pr(X \neq Y) \le \alpha$.

Accordingly, we call $(\pi,\tilde{\pi})$ an α-coupled policy pair if, for every state s, there exists a joint distribution over actions $(a,\tilde{a})$ with marginals $\pi(\cdot|s)$ and $\tilde{\pi}(\cdot|s)$ such that

$$\Pr(a \neq \tilde{a} \mid s) \le \alpha.$$

If $D_{\mathrm{TV}}^{\max}(\pi,\tilde{\pi}) \le \alpha$, such a coupling exists at every state. Thus, even without an explicit mixture representation, the two policies can be compared through the probability that their actions disagree. This is the mechanism that allows the CPI lower-bound argument to extend to general stochastic policies.

4.2 General Stochastic Policy Improvement Bound

We now prove the principal theoretically justified lower bound on performance. Recall the local surrogate

$$L_\pi(\tilde{\pi}) = J(\pi) + \frac{1}{1-\gamma}\sum_s d_\pi(s)\sum_a \tilde{\pi}(a|s)A^\pi(s,a).$$

For a fixed candidate policy $\tilde{\pi}$, we write

$$\bar{A}(s) := \sum_a \tilde{\pi}(a|s)A^\pi(s,a) = \mathbb{E}_{a\sim\tilde{\pi}(\cdot|s)}\left[A^\pi(s,a)\right].$$

Using the performance difference lemma and unrolling the discounted state visitation sums,

$$J(\tilde{\pi}) = J(\pi) + \sum_{t=0}^{\infty}\gamma^t \sum_s P(s_t = s \mid \tilde{\pi})\sum_a \tilde{\pi}(a \mid s)A^\pi(s,a) \tag{19}$$

$$= J(\pi) + \sum_{t=0}^{\infty}\gamma^t \mathbb{E}_{s_t\sim\tilde{\pi}}\left[\bar{A}(s_t)\right], \tag{20}$$

$$L_\pi(\tilde{\pi}) = J(\pi) + \sum_{t=0}^{\infty}\gamma^t \mathbb{E}_{s_t\sim\pi}\left[\bar{A}(s_t)\right], \tag{21}$$

The difference between the true performance and the surrogate is therefore

$$J(\tilde{\pi}) - L_\pi(\tilde{\pi}) = \sum_{t=0}^{\infty} \gamma^t \left(\mathbb{E}_{s_t \sim \tilde{\pi}}[\bar{A}(s_t)] - \mathbb{E}_{s_t \sim \pi}[\bar{A}(s_t)] \right). \tag{22}$$

Thus, the only difference between J and L_π is the state distribution used in the expectation. This is exactly the state distribution shift problem: $J(\tilde{\pi})$ uses states visited by the new policy, while $L_\pi(\tilde{\pi})$ freezes the state distribution at the old policy.

Assume that $(\pi, \tilde{\pi})$ is an α-coupled policy pair. First, we bound the magnitude of $\bar{A}(s)$. Since the expected advantage under the old policy is zero, $\mathbb{E}_{a \sim \pi(\cdot|s)}[A^\pi(s,a)] = 0$, we can write, for all s:

$$\begin{aligned}
\bar{A}(s) &= \mathbb{E}_{a \sim \tilde{\pi}(\cdot|s)} \left[A^\pi(s,a) \right] && (23)\\
&= \mathbb{E}_{\tilde{a} \sim \tilde{\pi}(\cdot|s)}[A^\pi(s,\tilde{a})] - \mathbb{E}_{a \sim \pi(\cdot|s)}[A^\pi(s,a)] \\
&= \mathbb{E}_{(a,\tilde{a}) \sim (\pi,\tilde{\pi})} \left[A^\pi(s,\tilde{a}) - A^\pi(s,a) \right] && (24)\\
&= \Pr(a \neq \tilde{a} \mid s) \mathbb{E}_{(a,\tilde{a}) \sim (\pi,\tilde{\pi}) | a \neq \tilde{a}} \left[A^\pi(s,\tilde{a}) - A^\pi(s,a) \right], && (25)
\end{aligned}$$

where the expectation is taken under the coupling of $\pi(\cdot|s)$ and $\tilde{\pi}(\cdot|s)$. On the event $a = \tilde{a}$, the difference is zero. Hence,

$$\begin{aligned}
|\bar{A}(s)| &\leq \Pr(a \neq \tilde{a} \mid s) \cdot \max_{a,\tilde{a}} |A^\pi(s,\tilde{a}) - A^\pi(s,a)| \\
&\leq \alpha \cdot 2 \max_{s,a} |A^\pi(s,a)|. && (26)
\end{aligned}$$

Defining $\epsilon := \max_{s,a} |A^\pi(s,a)|$, we obtain

$$|\bar{A}(s)| \leq 2\alpha\epsilon. \tag{27}$$

Next, we couple entire trajectories generated by π and $\tilde{\pi}$. At each time step, if both trajectories are at the same state, we sample their actions from the coupled action distribution. Let n_t denote the event that the two coupled trajectories have disagreed at least once before time t. Since the two policies disagree at each state with probability at most α,

$$\Pr(n_t = 0) \geq (1-\alpha)^t, \qquad \Pr(n_t > 0) \leq 1 - (1-\alpha)^t.$$

Conditioned on $n_t = 0$, the two trajectories have taken the same actions and therefore reached the same state at time t. Consequently, the $n_t = 0$ contribution cancels when comparing the two expectations in Equation (22). Therefore,

$$\begin{aligned}
&\left| \mathbb{E}_{s_t \sim \tilde{\pi}}[\bar{A}(s_t)] - \mathbb{E}_{s_t \sim \pi}[\bar{A}(s_t)] \right| \\
&\quad = \left| \Pr(n_t > 0) \left(\mathbb{E}_{s_t \sim \tilde{\pi} | n_t > 0}[\bar{A}(s_t)] - \mathbb{E}_{s_t \sim \pi | n_t > 0}[\bar{A}(s_t)] \right) \right| \\
&\quad \leq \Pr(n_t > 0) \left(\left| \mathbb{E}_{s_t \sim \tilde{\pi} | n_t > 0}[\bar{A}(s_t)] \right| + \left| \mathbb{E}_{s_t \sim \pi | n_t > 0}[\bar{A}(s_t)] \right| \right) \\
&\quad \leq \left(1 - (1-\alpha)^t\right) (2\alpha\epsilon + 2\alpha\epsilon) \\
&\quad = 4\alpha\epsilon \left(1 - (1-\alpha)^t\right). && (28)
\end{aligned}$$

The $n_t = 0$ terms are equal because this event means that the coupled trajectories agreed on every action before time t, and hence reached the same state at time t:

$$\mathbb{E}_{s_t \sim \tilde{\pi} | n_t = 0}[\bar{A}(s_t)] = \mathbb{E}_{s_t \sim \pi | n_t = 0}[\bar{A}(s_t)].$$

Substituting the bound from (28) into Equation (22), we get

$$\begin{aligned}
|J(\tilde{\pi}) - L_{\pi}(\tilde{\pi})| &= \left| \sum_{t=0}^{\infty} \gamma^t \left(\mathbb{E}_{s_t \sim \tilde{\pi}}[\bar{A}(s_t)] - \mathbb{E}_{s_t \sim \pi}[\bar{A}(s_t)] \right) \right| \\
&\leq \sum_{t=0}^{\infty} \gamma^t \left| \left(\mathbb{E}_{s_t \sim \tilde{\pi}}[\bar{A}(s_t)] - \mathbb{E}_{s_t \sim \pi}[\bar{A}(s_t)] \right) \right| \\
&\leq \sum_{t=0}^{\infty} \gamma^t 4\alpha\epsilon \left(1 - (1-\alpha)^t\right) \\
&= 4\alpha\epsilon \left(\sum_{t=0}^{\infty} \gamma^t - \sum_{t=0}^{\infty} (\gamma(1-\alpha))^t \right) \\
&= 4\alpha\epsilon \left(\frac{1}{1-\gamma} - \frac{1}{1-\gamma(1-\alpha)} \right) \\
&= \frac{4\alpha^2\gamma\epsilon}{(1-\gamma)(1-\gamma(1-\alpha))} \\
&\leq \frac{4\alpha^2\gamma\epsilon}{(1-\gamma)^2}.
\end{aligned} \tag{29}$$

Therefore,

$$J(\tilde{\pi}) \geq L_{\pi}(\tilde{\pi}) - \frac{4\gamma\epsilon}{(1-\gamma)^2}\alpha^2. \tag{30}$$

Finally, by the coupling characterization of total variation distance, if $\alpha = D_{\mathrm{TV}}^{\max}(\pi, \tilde{\pi})$, then such an α-coupling exists. Hence the general stochastic policy bound is

$$J(\tilde{\pi}) \geq L_{\pi}(\tilde{\pi}) - \frac{4\gamma\epsilon}{(1-\gamma)^2} \left(D_{\mathrm{TV}}^{\max}(\pi, \tilde{\pi})\right)^2. \tag{31}$$

Using Pinsker's inequality in the weaker form used by the TRPO paper, $D_{\mathrm{TV}}(p, q)^2 \leq D_{\mathrm{KL}}(p\|q)$, we obtain the KL-based lower bound

$$J(\tilde{\pi}) \geq L_{\pi}(\tilde{\pi}) - C D_{\mathrm{KL}}^{\max}(\pi, \tilde{\pi}), \qquad C := \frac{4\gamma\epsilon}{(1-\gamma)^2}, \tag{32}$$

where $D_{\mathrm{KL}}^{\max}(\pi, \tilde{\pi}) := \max_s D_{\mathrm{KL}}\left(\pi(\cdot|s) \,\|\, \tilde{\pi}(\cdot|s)\right)$. This is the TRPO policy-improvement bound: a local surrogate minus a policy-distance penalty for general stochastic updates.

4.3 Monotonic Improvement Algorithm

The KL-based lower bound leads to a policy iteration scheme with guaranteed non-decreasing performance, at least in the exact setting. At iteration i, define $M_i(\pi) := L_{\pi_i}(\pi) - C D_{\mathrm{KL}}^{\max}(\pi_i, \pi)$. Equation (32) implies that

$$J(\pi) \geq M_i(\pi) \qquad \text{for all candidate policies } \pi.$$

Moreover, at the current policy π_i,

$$M_i(\pi_i) = L_{\pi_i}(\pi_i) - C D_{\mathrm{KL}}^{\max}(\pi_i, \pi_i) = J(\pi_i).$$

If we choose $\pi_{i+1} \in \arg\max_\pi M_i(\pi)$, then

$$J(\pi_{i+1}) \geq M_i(\pi_{i+1}) \geq M_i(\pi_i) = J(\pi_i).$$

Thus, maximizing this lower bound at every iteration gives a sequence of policies whose true performance is non-decreasing. This is a minorization-maximization algorithmic argument: M_i lower-bounds J and touches it at the current policy π_i.

Algorithm 1 Policy iteration algorithm with theoretically justified non-decreasing performance

1: Initialize policy π_0.
2: **for** $i = 0, 1, 2, \ldots$ until convergence **do**
3: Compute advantage values $A^{\pi_i}(s, a)$.
4: Set $C_i = \frac{4\gamma\epsilon_i}{(1-\gamma)^2}$, where $\epsilon_i = \max_{s,a} |A^{\pi_i}(s, a)|$.
5: Compute

$$\pi_{i+1} \in \arg\max_\pi \left[L_{\pi_i}(\pi) - C_i D_{\mathrm{KL}}^{\max}(\pi_i, \pi) \right].$$

where $L_{\pi_i}(\pi) = J(\pi_i) + \frac{1}{1-\gamma} \sum_s d_{\pi_i}(s) \sum_a \pi(a \mid s) A^{\pi_i}(s, a)$
6: **end for**

This algorithm is theoretical: it assumes exact advantages, exact state distributions, and exact optimization. Practical TRPO replaces it with a sampled constrained update over parameterized policies.

4.4 From the Theoretical Bound to a Practical Trust Region

To obtain a practical algorithm, TRPO parameterizes the policy as $\pi_\theta(a|s)$ and estimates the surrogate and constraint from finite samples. We write

$$J(\theta) := J(\pi_\theta), \qquad L_{\theta_{\mathrm{old}}}(\theta) := L_{\pi_{\theta_{\mathrm{old}}}}(\pi_\theta),$$

and use θ_{old} for the policy parameters before the update.

The bound in Equation (32) suggests optimizing a penalized objective

$$\max_\theta \left[L_{\theta_{\mathrm{old}}}(\theta) - C D_{\mathrm{KL}}^{\max}(\theta_{\mathrm{old}}, \theta) \right],$$

where $L_{\theta_{\mathrm{old}}}(\theta)$ abbreviates $L_{\pi_{\theta_{\mathrm{old}}}}(\pi_\theta)$. However, $C = 4\gamma\epsilon/(1-\gamma)^2$ can be very large when γ is close to one, making the penalty difficult to use directly [Achiam, 2017].

TRPO therefore replaces the penalty with a hard trust-region constraint:

$$\max_\theta \quad L_{\theta_{\mathrm{old}}}(\theta) \quad \text{subject to} \quad D_{\mathrm{KL}}^{\max}(\theta_{\mathrm{old}}, \theta) \leq \delta. \tag{33}$$

This keeps the central theoretical idea: optimize the local surrogate, but restrict the update so the new policy does not move too far from the old one. The trust-region radius δ now controls the allowed policy movement directly, avoiding the need to choose the large theoretical penalty coefficient C as an optimization hyperparameter.

The maximum KL constraint is still difficult because it requires controlling the KL divergence at every state. TRPO therefore makes one more practical approximation and replaces the maximum statewise KL by the average KL under the old policy's state distribution:

$$\bar{D}_{\mathrm{KL}}(\theta_{\mathrm{old}}, \theta) := \mathbb{E}_{s \sim d_{\theta_{\mathrm{old}}}} \left[D_{\mathrm{KL}} \left(\pi_{\theta_{\mathrm{old}}}(\cdot|s) \,\|\, \pi_\theta(\cdot|s) \right) \right].$$

This gives the practical constrained update

$$\theta_{\text{new}} \in \arg\max_{\theta} \quad L_{\theta_{\text{old}}}(\theta) \quad \text{subject to} \quad \bar{D}_{\text{KL}}(\theta_{\text{old}}, \theta) \leq \delta. \tag{34}$$

This is the trust-region optimization problem that TRPO approximately solves at every iteration. The remaining approximations are practical rather than conceptual: estimate the objective and average KL from samples, then solve the nonlinear constrained problem by a local approximation around θ_{old}.

4.5 Sample-Based Estimation of the Objective and Constraint

The surrogate objective can be expanded as

$$L_{\theta_{\text{old}}}(\theta) = J(\theta_{\text{old}}) + \frac{1}{1-\gamma} \sum_s d_{\theta_{\text{old}}}(s) \sum_a \pi_\theta(a|s) A^{\theta_{\text{old}}}(s,a).$$

The term $J(\theta_{\text{old}})$ is independent of θ, so it can be dropped for optimization. The advantage can also be replaced by an action-value function without changing the optimizer, because

$$\begin{aligned} \sum_s d_{\theta_{\text{old}}}(s) \sum_a \pi_\theta(a|s) A^{\theta_{\text{old}}}(s,a) &= \sum_s d_{\theta_{\text{old}}}(s) \sum_a \pi_\theta(a|s) \left(Q^{\theta_{\text{old}}}(s,a) - V^{\theta_{\text{old}}}(s) \right) \\ &= \sum_s d_{\theta_{\text{old}}}(s) \sum_a \pi_\theta(a|s) Q^{\theta_{\text{old}}}(s,a) - \sum_s d_{\theta_{\text{old}}}(s) V^{\theta_{\text{old}}}(s). \end{aligned} \tag{35}$$

The second term is a state-value term independent of θ. Therefore the original TRPO paper writes the sample-based objective using Monte Carlo estimates of $Q^{\theta_{\text{old}}}$ [Schulman et al., 2015]. A state-value baseline can reduce variance without changing the expected policy gradient. Later implementations often learn this baseline and use advantage estimators such as GAE [Schulman et al., 2016; Sutton and Barto, 2018]; this is the distinction between our Monte Carlo TRPO/NPG runs and the PPO runs. Using the Q-form, the practical constrained objective is

$$\max_{\theta} \quad \sum_s d_{\theta_{\text{old}}}(s) \sum_a \pi_\theta(a \mid s) Q^{\theta_{\text{old}}}(s,a) \quad \text{s.t.} \quad \mathbb{E}_{s \sim d_{\theta_{\text{old}}}} \left[D_{\text{KL}} \left(\pi_{\theta_{\text{old}}}(\cdot \mid s) \,\|\, \pi_\theta(\cdot \mid s) \right) \right] \leq \delta. \tag{36}$$

Importance sampling lets us estimate the action expectation using actions drawn from a sampling distribution $q(a|s)$, provided q has support wherever π_θ may place probability. For each state,

$$\sum_a \pi_\theta(a|s) Q^{\theta_{\text{old}}}(s,a) = \mathbb{E}_{a \sim q(\cdot|s)} \left[\frac{\pi_\theta(a|s)}{q(a|s)} Q^{\theta_{\text{old}}}(s,a) \right].$$

Therefore, Equation (36) can be written in expectation form as

$$\max_{\theta} \quad \mathbb{E}_{s \sim d_{\theta_{\text{old}}},\, a \sim q} \left[\frac{\pi_\theta(a|s)}{q(a|s)} Q^{\theta_{\text{old}}}(s,a) \right] \quad \text{s.t.} \quad \mathbb{E}_{s \sim d_{\theta_{\text{old}}}} \left[D_{\text{KL}} \left(\pi_{\theta_{\text{old}}}(\cdot|s) \,\|\, \pi_\theta(\cdot|s) \right) \right] \leq \delta. \tag{37}$$

In implementation, we replace these expectations by sample averages, and $Q^{\theta_{\text{old}}}$ is replaced by Monte Carlo return estimates or, in advantage-based implementations, by estimates $\widehat{A}$.

The TRPO paper also describes a vine estimator, which branches multiple rollouts from selected simulator states. Because this requires reset access and is less aligned with online settings, we focus on the model-free single-path estimator.

4.5.1 Single-Path Sampling

We use the single-path method as the standard model-free sampling scheme. We collect trajectories by running the old policy $\pi_{\theta_{\text{old}}}$: $s_0, a_0, s_1, a_1, \ldots, s_{T-1}, a_{T-1}, s_T$. In this case, $q(a|s) = \pi_{\theta_{\text{old}}}(a|s)$. For each visited state-action pair (s_t, a_t), the original TRPO paper estimates $Q^{\theta_{\text{old}}}(s_t, a_t)$ by the discounted return along the trajectory:

$$\widehat{Q}_t = \sum_{\ell=t}^{T-1} \gamma^{\ell-t} r_\ell.$$

Repeating this over N trajectories of horizon T gives $M = NT$ state-action-return tuples. Given a dataset $\mathcal{D} = \{(s_n, a_n, \widehat{Q}_n)\}_{n=1}^M$, the empirical Q-based surrogate for a candidate policy π_θ is

$$\widehat{L}_Q(\theta) = \frac{1}{M} \sum_{n=1}^{M} \frac{\pi_\theta(a_n|s_n)}{\pi_{\theta_{\text{old}}}(a_n|s_n)} \widehat{Q}_n. \tag{38}$$

If advantage estimates are used instead, one obtains the analogous $\widehat{L}_A$ by replacing $\widehat{Q}_n$ with $\widehat{A}_n$. The only difference is how the signal multiplying the likelihood ratio is estimated.

The empirical average KL constraint is

$$\widehat{\bar{D}}_{\text{KL}}(\theta_{\text{old}}, \theta) = \frac{1}{M} \sum_{n=1}^{M} D_{\text{KL}}\left(\pi_{\theta_{\text{old}}}(\cdot|s_n) \,\|\, \pi_\theta(\cdot|s_n)\right). \tag{39}$$

The gradient of the Q-based empirical surrogate at $\theta = \theta_{\text{old}}$ has the usual policy-gradient form:

$$\nabla_\theta \widehat{L}_Q(\theta)\big|_{\theta=\theta_{\text{old}}} = \frac{1}{M} \sum_{n=1}^{M} \nabla_\theta \left(\frac{\pi_\theta(a_n|s_n)}{\pi_{\theta_{\text{old}}}(a_n|s_n)}\right)_{\theta=\theta_{\text{old}}} \widehat{Q}_n = \frac{1}{M} \sum_{n=1}^{M} \nabla_\theta \log \pi_\theta(a_n|s_n)\big|_{\theta=\theta_{\text{old}}} \widehat{Q}_n. \tag{40}$$

Thus, at the old policy, the sampled TRPO surrogate recovers the action-value policy-gradient estimator. Replacing $\widehat{Q}_n$ by $\widehat{A}_n$ gives the usual advantage-form version.

4.6 Approximately Solving the Constrained Optimization

At each iteration, TRPO aims to solve the sampled trust-region problem

$$\max_\theta \quad \widehat{L}(\theta) \quad \text{subject to} \quad \widehat{\bar{D}}_{\text{KL}}(\theta_{\text{old}}, \theta) \le \delta. \tag{41}$$

Both terms are nonlinear functions of θ, especially for neural-network policies. The practical algorithm therefore computes a local step around θ_{old}, making use of Taylor series expansion. Let $d := \theta - \theta_{\text{old}}$, and $g := \nabla_\theta \widehat{L}(\theta)\big|_{\theta=\theta_{\text{old}}}$. The surrogate is linearized:

$$\widehat{L}(\theta_{\text{old}} + d) \approx \widehat{L}(\theta_{\text{old}}) + g^\top d,$$

so the relevant objective is $g^\top d$. The average KL constraint is expanded to second order. At the old policy, $\widehat{\bar{D}}_{\text{KL}}(\theta_{\text{old}}, \theta_{\text{old}}) = 0$, and the first derivative with respect to θ vanishes at $\theta = \theta_{\text{old}}$, as it is locally minimized at that point. Thus the first nonzero term is quadratic:

$$\widehat{\bar{D}}_{\text{KL}}(\theta_{\text{old}}, \theta_{\text{old}} + d) \approx \frac{1}{2} d^\top A d, \qquad A := \nabla_\theta^2 \widehat{\bar{D}}_{\text{KL}}(\theta_{\text{old}}, \theta)\big|_{\theta=\theta_{\text{old}}}.$$

The local trust-region subproblem is therefore

$$\max_{d} \quad g^\top d \qquad \text{subject to} \qquad \frac{1}{2} d^\top A d \le \delta. \tag{42}$$

The matrix A is the Hessian of the average KL divergence at the old policy. It is also the Fisher information matrix of the policy, averaged over sampled states. This gives the subproblem its geometric interpretation: directions that cause large changes in the policy distribution are expensive, while directions that change the policy only slightly are cheap.

Solving Equation (42) by Lagrange multipliers gives the search direction. With $\lambda > 0$,

$$\mathcal{L}(d, \lambda) = g^\top d - \lambda \left(\frac{1}{2} d^\top A d - \delta \right).$$

Setting the derivative with respect to d to zero gives $g - \lambda A d = 0$, so, $d = \frac{1}{\lambda} A^{-1} g$, assuming A is invertible. The optimum lies on the boundary of the quadratic trust region: $\frac{1}{2} d^\top A d = \delta$. Substituting $d = \frac{1}{\lambda} A^{-1} g$, we obtain

$$\delta = \frac{1}{2} \left(\frac{1}{\lambda} A^{-1} g \right)^\top A \left(\frac{1}{\lambda} A^{-1} g \right) = \frac{1}{2\lambda^2} g^\top A^{-1} g. \tag{43}$$

Therefore, $\lambda = \sqrt{\frac{g^\top A^{-1} g}{2\delta}}$, and the optimal local step is

$$d^\star = \sqrt{\frac{2\delta}{g^\top A^{-1} g}} A^{-1} g. \tag{44}$$

Thus the direction is $A^{-1} g$, and the scalar factor chooses the largest step along that direction allowed by the quadratic KL approximation.

4.7 Connection to Natural Policy Gradient

The appearance of $A^{-1} g$ is not accidental. Natural Policy Gradient (NPG) [Kakade, 2001] replaces Euclidean distance in parameter space with a distributional metric on policies, following the natural-gradient perspective of [Amari, 1998]. Since policies are probability distributions over actions, the KL divergence provides a natural local measure of distance between nearby policies. Its second-order approximation gives the Fisher information matrix. Ordinary gradient ascent can be coordinate dependent: a small Euclidean change in parameters may cause a large change in the action distribution, while a larger Euclidean change may barely change the policy. The Fisher matrix corrects for this by measuring steps in policy-distribution space. This also explains parameterization invariance: under a change of coordinates, the ordinary Euclidean gradient can change, while the natural gradient represents the steepest direction under the geometry induced by the policy manifold.

In this geometry, the natural-gradient direction is $\widetilde{\nabla} J(\theta) = F(\theta)^{-1} \nabla_\theta J(\theta)$. In the TRPO subproblem, A plays the role of the Fisher matrix and g is the empirical policy-gradient estimate. Therefore $A^{-1} g$ is exactly the natural policy-gradient direction. If A is singular, the corresponding minimum-norm version uses a pseudoinverse, giving $A^\dagger g$. This also explains the relation between TRPO and earlier natural-gradient or covariant policy-search methods [Bagnell and Schneider, 2003].

In our experiments, we compute this direction using the same conjugate-gradient and Fisher-vector-product machinery as TRPO; what we remove is the KL-based rescaling and acceptance test. The resulting fixed-step update is $\theta_{k+1} = \theta_k + \eta F(\theta_k)^\dagger \nabla_\theta J(\theta_k)$. The boundary scaling derived above can be viewed as a particular choice of η that saturates the local quadratic KL constraint. The trust-region radius δ changes the length of the step, not the direction. TRPO can therefore be viewed as a practical trust-region refinement of NPG: it keeps the natural-gradient direction, scales it by a KL radius, and then uses backtracking line search to guard against errors in the local linear-quadratic approximation. This line search is important because the actual surrogate and KL are nonlinear; a nominal natural-gradient step can otherwise violate the empirical KL constraint or degrade performance. The NPG literature also connects this geometry to compatible function approximation and greedy improvement, which is one reason NPG is more than a simple rescaling of the vanilla policy gradient. A useful intuition is the near-deterministic policy case [Xie, 2026]. When a policy is close to assigning all probability to one action, ordinary parameter changes can have a misleading scale. The Fisher/KL metric instead measures how much the action distribution itself changes, which is precisely the quantity TRPO wants to keep under control.

4.8 Conjugate Gradient and Fisher-Vector Products

For neural-network policies, explicitly forming A is usually infeasible. If the policy has P parameters, storing A requires $O(P^2)$ memory and a direct inversion would cost on the order of $O(P^3)$. TRPO avoids this by approximately solving the linear system $Ax = g$ with conjugate gradient. The method does not require the full matrix A; it only needs products Av. These can be computed as Hessian-vector products of the empirical average KL: $Av = \nabla_\theta^2 \widehat{\bar{D}}_{\mathrm{KL}}(\theta_{\mathrm{old}}, \theta)\big|_{\theta=\theta_{\mathrm{old}}} v$. Equivalently, these are Fisher-vector products. Automatic differentiation can compute them without materializing the dense Fisher matrix. The paper also notes that Fisher-vector products can be computed on a subset of the batch, for example about 10% of the data, because the Fisher matrix is acting as a local metric. With this subsampling, the cost of computing the natural-gradient step can be close to the cost of computing the policy gradient itself. TRPO uses the analytic Hessian of the KL divergence rather than the empirical covariance matrix of score gradients. The analytic estimate is

$$\widehat{A}_{ij} = \frac{1}{M} \sum_{n=1}^{M} \frac{\partial^2}{\partial\theta_i \partial\theta_j} D_{\mathrm{KL}}\left(\pi_{\theta_{\mathrm{old}}}(\cdot|s_n) \,\|\, \pi_\theta(\cdot|s_n)\right)\Big|_{\theta=\theta_{\mathrm{old}}},$$

whereas the empirical Fisher estimator is $\widehat{A}_{ij}^{\mathrm{emp}} = \frac{1}{M}\sum_{n=1}^{M} \frac{\partial}{\partial\theta_i} \log \pi_\theta(a_n|s_n) \frac{\partial}{\partial\theta_j} \log \pi_\theta(a_n|s_n)\Big|_{\theta=\theta_{\mathrm{old}}}$. The second construction is the "empirical FIM" ablation reported in the TRPO experiments: it is identical to single-path TRPO except that the Fisher matrix is estimated by the covariance matrix of the sampled score gradients rather than by the analytic KL Hessian. The analytic KL Hessian integrates over the action distribution at each sampled state and does not depend on the particular sampled action a_n. This gives computational benefits in the large-scale setting, because it avoids storing a dense Hessian or all policy-gradient vectors from the batch.

4.9 Backtracking Line Search

The step in Equation (44) is optimal only for the local linear-quadratic approximation. The true surrogate and the empirical KL are nonlinear in θ. Therefore, the full proposed step may violate the KL constraint or fail to improve the surrogate. Let $x \approx A^{-1}g$ be the conjugate-gradient solution. The proposed full step is $\Delta = \sqrt{\frac{2\delta}{x^\top A x}}\, x$. TRPO then tries candidates $\theta_{\mathrm{candidate}} = \theta_{\mathrm{old}} + \beta^j \Delta$, for

Algorithm 2 Trust Region Policy Optimization

Require: Initial policy parameters θ_0, KL radius δ, CG iterations n_{cg}, backtracking coefficient $\beta \in (0,1)$, maximum line-search trials $J_{\max}$

for $i = 0, 1, 2, \ldots$ **do**
  Collect trajectories using the current policy π_{θ_i}.
  Estimate returns $\widehat{Q}_n$ or advantages $\widehat{A}_n$.
  Construct the empirical surrogate $\widehat{L}_i(\theta)$.
  Construct the empirical average KL $\widehat{\bar{D}}_{\text{KL}}(\theta_i, \theta)$.
  Compute $g_i = \nabla_\theta \widehat{L}_i(\theta)\big|_{\theta=\theta_i}$.
  Define the Fisher-vector product $v \mapsto \nabla^2_\theta \widehat{\bar{D}}_{\text{KL}}(\theta_i, \theta)\big|_{\theta=\theta_i} v$.
  Use n_{cg} conjugate-gradient iterations to compute $x_i \approx A_i^{-1} g_i$.
  Form $\Delta_i = \sqrt{2\delta/(x_i^\top A_i x_i)}\, x_i$.
  Set $\theta_{i+1} = \theta_i$.
  for $j = 0, 1, \ldots, J_{\max}$ **do**
    Set $\theta_{\text{cand}} = \theta_i + \beta^j \Delta_i$.
    if $\widehat{L}_i(\theta_{\text{cand}}) \geq \widehat{L}_i(\theta_i)$ and $\widehat{\bar{D}}_{\text{KL}}(\theta_i, \theta_{\text{cand}}) \leq \delta$ **then**
      Accept the step: $\theta_{i+1} = \theta_{\text{cand}}$.
      break
    end if
  end for
end for

$j = 0, 1, 2, \ldots$, where $\beta \in (0,1)$ is a backtracking coefficient. The step is accepted only if it improves the empirical surrogate and satisfies the empirical KL constraint:

$$\widehat{L}(\theta_{\text{candidate}}) \geq \widehat{L}(\theta_{\text{old}}) \qquad \text{and} \qquad \widehat{\bar{D}}_{\text{KL}}(\theta_{\text{old}}, \theta_{\text{candidate}}) \leq \delta.$$

Thus, the candidate step is checked against the actual nonlinear surrogate and KL, rather than trusted solely because it solves the local quadratic model. This acceptance check is the practical safeguard that fixed-step NPG lacks. Thus, TRPO turns the CPI-style lower-bound principle into an implementable algorithm for general stochastic policies: optimize a sampled surrogate, constrain the average KL, and accept only steps that remain reliable under the empirical objective and constraint.

5 Proximal Policy Optimization

Proximal Policy Optimization (PPO) [Schulman et al., 2017] is a first-order successor to TRPO. It keeps the conservative-update motivation but avoids the KL-constrained second-order solve. Starting from the same importance-ratio surrogate, PPO clips probability ratios so that once an update has moved far enough in a direction that improves the objective, further movement is no longer rewarded. This gives a soft trust-region effect while allowing multiple epochs of minibatch optimization on the same on-policy batch. The paper also studies an adaptive KL penalty variant, but reports the clipped objective as the stronger practical choice.

5.1 The CPI/TRPO Surrogate in Sampled Form

Recall that TRPO optimizes the surrogate objective

$$L_{\pi_{\text{old}}}(\pi_\theta) = J(\pi_{\text{old}}) + \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_{\pi_{\text{old}}}}\left[\sum_a \pi_\theta(a|s)A^{\pi_{\text{old}}}(s,a)\right].$$

In sample-based policy optimization, this objective is written using importance ratios. Let $r_t(\theta) := \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$. The empirical surrogate used in CPI/TRPO can then be written as

$$L^{\text{CPI}}(\theta) = \widehat{\mathbb{E}}_t\left[r_t(\theta)\widehat{A}_t\right], \tag{45}$$

where $\widehat{\mathbb{E}}_t$ denotes the empirical average over sampled timesteps and $\widehat{A}_t$ is an estimate of the advantage function under the old policy.

This objective is natural because it asks the new policy to increase the probability of actions with positive estimated advantage and decrease the probability of actions with negative estimated advantage. However, if Equation (45) is optimized without any restriction, the policy can move too far from $\pi_{\theta_{\text{old}}}$. This is the same difficulty that motivated the trust-region constraint in TRPO. PPO addresses it by changing the objective so that large probability-ratio changes are no longer rewarded.

5.2 The Clipped Surrogate Objective

The main PPO objective is the clipped surrogate objective:

$$L^{\text{CLIP}}(\theta) = \widehat{\mathbb{E}}_t\left[\min\left(r_t(\theta)\widehat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\,\widehat{A}_t\right)\right], \tag{46}$$

where $\epsilon > 0$ is a hyperparameter, commonly chosen around 0.1 or 0.2.

The clipping operation replaces the ratio in the clipped term by its projection onto the interval $[1-\epsilon, 1+\epsilon]$. The key idea is not that the new policy is mathematically forbidden from leaving this interval. Rather, the objective removes the incentive to improve the surrogate by pushing the probability ratio too far away from 1. This is why PPO is called "proximal": it encourages the new policy to remain close to the old policy, but it does so through the objective rather than through an explicit hard constraint.

The role of the minimum in Equation (46) is important. It makes the objective pessimistic relative to the unclipped surrogate. The clipping effect is easiest to see by separating the sign of the advantage. If $\widehat{A}_t > 0$, increasing $r_t(\theta)$ improves the surrogate because the sampled action was better than expected. PPO prevents unlimited improvement by clipping the ratio above $1+\epsilon$:

$$\min\left(r_t(\theta)\widehat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\widehat{A}_t\right) = \min\left(r_t(\theta)\widehat{A}_t,\ (1+\epsilon)\widehat{A}_t\right)$$

whenever $r_t(\theta) > 1+\epsilon$. Thus, PPO does not reward increasing the probability of a good action beyond the clipping threshold.

Similarly, if $\widehat{A}_t < 0$, then decreasing $r_t(\theta)$ improves the surrogate, because the sampled action was worse than expected. PPO prevents the ratio from becoming too small by clipping below $1-\epsilon$. In

this case, the clipped objective stops giving extra benefit for decreasing the probability of a bad action too aggressively.

Thus, PPO keeps the local CPI/TRPO surrogate when the policy update is small, but modifies it when the update becomes too large. Around $\theta = \theta_{\text{old}}$, we have $r_t(\theta) = 1$, so L^{CLIP} and L^{CPI} agree to first order. Farther away from the old policy, the objectives differ, and the clipped objective discourages destructive policy updates. It is important to note that this pessimism is with respect to the sampled CPI surrogate L^{CPI}, not a formal monotonic lower bound on the true performance J. In this sense, PPO preserves the conservative-update intuition of TRPO, but does not inherit TRPO's exact policy-improvement bound.

5.3 Adaptive KL Penalty Variant

PPO also considers an alternative objective based on a KL penalty:

$$L^{\text{KLPEN}}(\theta) = \widehat{\mathbb{E}}_t \left[r_t(\theta)\widehat{A}_t - \beta D_{\text{KL}} \left(\pi_{\theta_{\text{old}}}(\cdot|s_t) \,\|\, \pi_\theta(\cdot|s_t) \right) \right]. \tag{47}$$

This objective is closer in spirit to the theoretical TRPO lower bound, where the surrogate is penalized by a policy-distance term. The difficulty is that a fixed penalty coefficient β may not work well throughout training. If β is too small, the policy may move too far; if β is too large, the update may become too conservative.

To address this, PPO adapts β to target a desired average KL divergence d_{targ}. After each policy update, compute $d = \widehat{\mathbb{E}}_t \left[D_{\text{KL}} \left(\pi_{\theta_{\text{old}}}(\cdot|s_t) \,\|\, \pi_\theta(\cdot|s_t) \right) \right]$. Then update β according to the rule

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

This adaptive penalty version makes the KL penalty responsive to the observed update size. However, the PPO paper reports that the clipped objective generally performed better than the adaptive KL penalty variant, which is why PPO is most commonly associated with Equation (46).

5.4 Full Actor-Critic Objective

In practical deep RL implementations, PPO is usually used in an actor-critic form. The policy network is updated using the clipped surrogate objective, while a value function is learned in parallel to estimate returns and advantages. An entropy bonus is often added to encourage exploration. The combined objective is

$$L^{\text{CLIP+VF+S}}(\theta) = \widehat{\mathbb{E}}_t \left[L_t^{\text{CLIP}}(\theta) - c_1 L_t^{\text{VF}}(\theta) + c_2 S[\pi_\theta](s_t) \right], \tag{48}$$

where $L_t^{\text{VF}}(\theta) = \left(V_\theta(s_t) - V_t^{\text{targ}} \right)^2$ is the value-function loss, $S[\pi_\theta](s_t)$ is an entropy bonus, and c_1, c_2 are coefficients controlling the relative weights of the value loss and entropy bonus. In our implementation, V_t^{targ} is the discounted return target stored for value-function fitting, while the policy update uses the separate GAE advantage estimate.

The advantage estimates are often computed using Generalized Advantage Estimation (GAE) [Schulman et al., 2016]. Defining the temporal-difference residual $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$,

the truncated GAE estimator has the form $\widehat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + (\gamma\lambda)^2\delta_{t+2} + \cdots$, with truncation at the end of the sampled rollout segment. The parameter λ controls the bias-variance tradeoff: larger λ gives estimates closer to Monte Carlo returns, while smaller λ gives more bootstrapped, lower-variance estimates.

5.5 PPO Algorithm

PPO alternates between collecting data with the current policy and performing several epochs of minibatch optimization on the same collected batch. This is one of the key practical differences from vanilla policy gradient methods, which typically use each batch for only one gradient update.

Algorithm 3 Proximal Policy Optimization

Require: Initial policy parameters θ_0, clipping parameter ϵ, number of epochs K, minibatch size M

for $i = 0, 1, 2, \ldots$ **do**
  Set $\theta_{\text{old}} \leftarrow \theta_i$.
  Collect trajectories using $\pi_{\theta_{\text{old}}}$.
  Estimate returns and advantages $\widehat{A}_t$, for example using GAE.
  for $k = 1, \ldots, K$ **do**
    Sample a minibatch of timesteps.
    Compute probability ratios $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$.
    Optimize the clipped objective $\widehat{\mathbb{E}}_t \left[\min\left(r_t(\theta)\widehat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\widehat{A}_t\right)\right]$ possibly together with value-function and entropy terms.
  end for
  Set $\theta_{i+1} \leftarrow \theta$.
end for

PPO therefore retains the local surrogate viewpoint while replacing TRPO's constrained second-order update with a clipped objective optimized by minibatch gradient methods. This makes it less theoretically direct than TRPO, but much simpler to implement and tune in practice. The PPO paper emphasizes this same tradeoff: PPO is designed to retain many of TRPO's practical benefits while using only first-order optimization and permitting multiple epochs of minibatch updates on each on-policy batch. This combination of clipping, critic-based advantage estimation, and data reuse helps explain PPO's practical influence in modern policy optimization.

6 Implementation, Experiments, and Results

The theory above describes how CPI, NPG, TRPO, and PPO control policy updates. We now test these ideas in code on MuJoCo locomotion and Atari tasks; the implementation is available at https://github.com/djdhillxn/trpo. The continuous-control experiments use TRPO and NPG to study the effect of step-size selection along an approximate natural-gradient direction. PPO is included in its modern actor-critic form, with clipping, advantage estimation, a learned critic, and minibatch reuse. Thus, the empirical section has two roles: it compares TRPO and NPG as update mechanisms, and it compares PPO against them as a practical training setup.

Table 1: MuJoCo locomotion results over three completed runs. Final return is reported as mean $\pm$ standard deviation across seeds. "Best" is the best training rollout return observed across the run set. Env steps are total environment interactions per run. Mean KL is the average logged policy KL over epochs and runs; for TRPO and NPG it is a direct diagnostic of update size, while for PPO it is reported only descriptively.

Environment	Method	Env steps	Final return	Best return	Mean KL
Swimmer-v5	TRPO	20M	78.0 ± 17.9	97.8	0.00705
Swimmer-v5	NPG	20M	31.7 ± 0.2	32.3	0.00002
Swimmer-v5	PPO-Clip	20M	112.3 ± 4.3	116.8	0.00826
Hopper-v5	TRPO	200M	1386.7 ± 165.4	2186.7	0.00916
Hopper-v5	NPG	200M	552.3 ± 273.9	939.6	0.00020
Hopper-v5	PPO-Clip	20M	2149.5 ± 83.9	3201.1	0.01496
Walker2d-v5	TRPO	200M	746.7 ± 374.7	1270.6	0.00906
Walker2d-v5	NPG	200M	73.3 ± 46.1	109.3	0.00002
Walker2d-v5	PPO-Clip	20M	3027.2 ± 804.9	4599.5	0.01396

6.1 Experimental Design

We evaluate two benchmark families. The MuJoCo experiments use `Swimmer`, `Hopper`, and `Walker2d`, where we run TRPO, NPG, and PPO. For TRPO and NPG, the comparison is deliberately narrow: both methods use the same policy class, rollout estimator, discount factor, and conjugate-gradient/Fisher-vector-product computation. Our NPG implementation is therefore not a dense-matrix inversion version of the original algorithm; it uses the same approximate natural-gradient direction as TRPO, but applies a fixed scalar step instead of a KL-scaled accepted step. This makes the comparison primarily about how the step length is chosen.

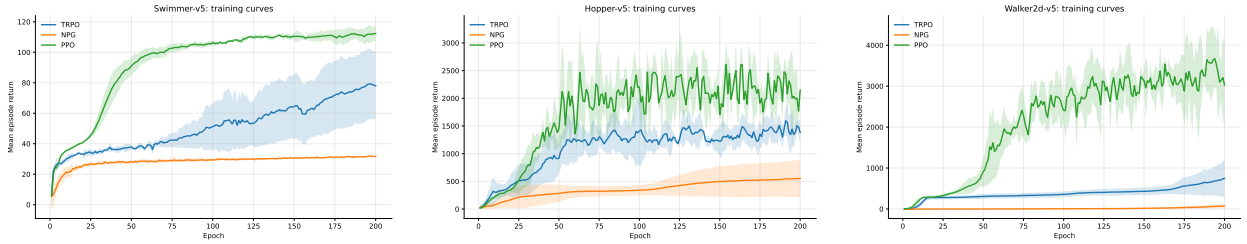
PPO is run as a compact actor-critic baseline. It uses the clipped surrogate, GAE, a learned value function, and several minibatch passes over each collected batch. This also explains the smaller rollout budget used for PPO on Hopper and Walker2d: PPO uses $100k$ environment steps per epoch, while TRPO and NPG use $1M$. Over 200 epochs this is $20M$ environment interactions for PPO and $200M$ for TRPO/NPG. Swimmer is the exception; all three methods use $100k$ steps per epoch.

The Atari experiments use the games from the TRPO paper: `BeamRider`, `Breakout`, `Enduro`, `Pong`, `Qbert`, `Seaquest`, and `SpaceInvaders`. There we compare single-path TRPO and PPO-Clip from image input using the same compact CNN policy body. Because these runs are expensive, they are intended as a smaller reproduction study rather than a benchmark-scale replication.

6.2 Continuous-Control Results

Table 1 and Figure 1 show a consistent pattern across the three MuJoCo tasks: PPO-Clip obtains the strongest returns, TRPO improves over the baseline NPG setup, and NPG moves very little in realized KL. The TRPO–NPG gap is therefore best interpreted as an update-size effect rather than a difference in search direction. The main result is the sample-efficiency gap on Hopper and Walker2d. PPO reaches higher returns while using one-tenth of the environment interactions used by TRPO and NPG. Because PPO also uses a critic, GAE, and minibatch reuse, this is a comparison between training setups rather than objectives alone; that practical combination is precisely what makes PPO attractive.

The TRPO–NPG contrast is different. Since the two methods share the same search direction in



(a) Swimmer-v5 (b) Hopper-v5 (c) Walker2d-v5

Figure 1: MuJoCo locomotion learning curves for TRPO, NPG, and PPO-Clip. Curves show run-averaged training rollout return, with shaded regions indicating across-run variation.

Table 2: Hopper-v5 NPG step-size study over three runs. Each row uses the same policy architecture, estimator, batch size, and Fisher-vector-product computation; only the fixed NPG step multiplier changes. Mean and max KL summarize the realized policy movement after each update, and $|\Delta\widehat{J}|$ summarizes epoch-to-epoch training-return volatility.

Multiplier	Step size	Final return	Best return	Mean KL	Max KL	Mean $\lvert\Delta\widehat{J}\rvert$	Max $\lvert\Delta\widehat{J}\rvert$
$1\times$	0.015	552.3 ± 273.9	939.6	0.00020	0.00137	3.2	35.5
$3\times$	0.045	1169.0 ± 164.2	1362.7	0.00296	0.01194	21.7	206.0
$9\times$	0.135	1406.3 ± 215.9	3053.6	0.03640	0.15601	192.1	1420.6

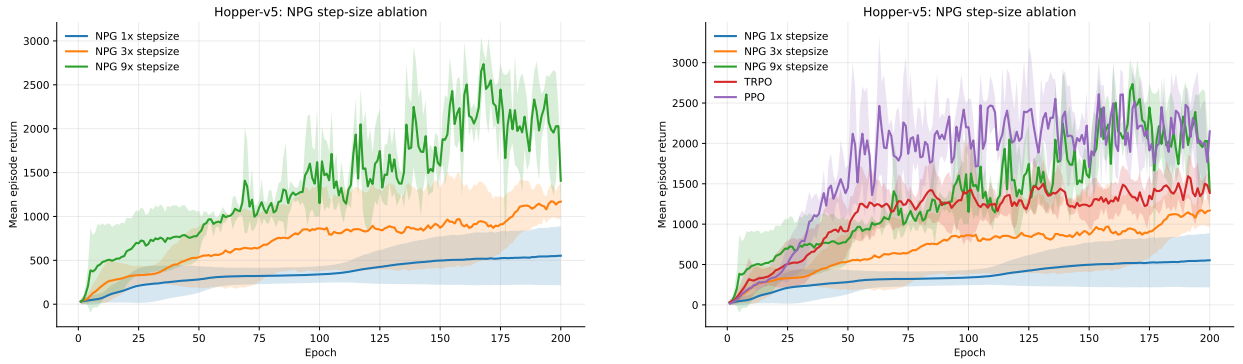
our implementation, their gap mainly reflects step selection. NPG moves very little in KL and learns slowly; TRPO uses the trust-region scale more effectively and obtains higher returns.

6.3 Hopper NPG Step-Size Study

The locomotion results suggest that baseline NPG is limited mainly by how far it moves along the natural-gradient direction. We therefore run a focused step-size study on Hopper-v5, where the default NPG and TRPO runs are clearly separated. The comparison keeps the NPG update rule fixed and changes only the scalar multiplier applied to the natural-gradient direction. If $x_k \approx A_k^{-1} g_k$ denotes the approximate natural-gradient direction, then NPG updates by $\theta_{k+1} = \theta_k + \alpha_{\text{NPG}} x_k$. We compare the default Hopper value $\alpha_{\text{NPG}} = 0.015$ with $3\times$ and $9\times$ larger steps: $\alpha_{\text{NPG}} \in \{0.015,\ 0.045,\ 0.135\}$. All three settings use the same policy architecture, estimator, batch size, and Fisher-vector product computation. The only intended change is the fixed scalar multiplying the natural-gradient direction. The sweep shows the expected tradeoff. The default step is safe but too small, so learning is slow. The $3\times$ step improves learning while still keeping policy movement moderate. The $9\times$ step reaches better policies in some runs, but its KL values and return swings are much larger. In other words, larger fixed natural-gradient steps can help, but they also make the update more erratic. TRPO's advantage is that it adds an acceptance rule to the same basic direction.

6.4 Atari Results

Table 3 and Figure 3 summarize the Atari runs. TRPO and PPO use the same compact CNN policy body, so this comparison uses the same action-policy capacity. The runs are smaller than the original Atari studies, but they still provide a useful image-based test of the update rules.



(a) Fixed-step NPG variants

(b) NPG variants with TRPO and PPO-Clip

Figure 2: Hopper-v5 learning curves for the NPG step-size study. The left panel isolates the fixed-step natural-gradient variants, while the right panel places them beside TRPO and PPO-Clip for scale.

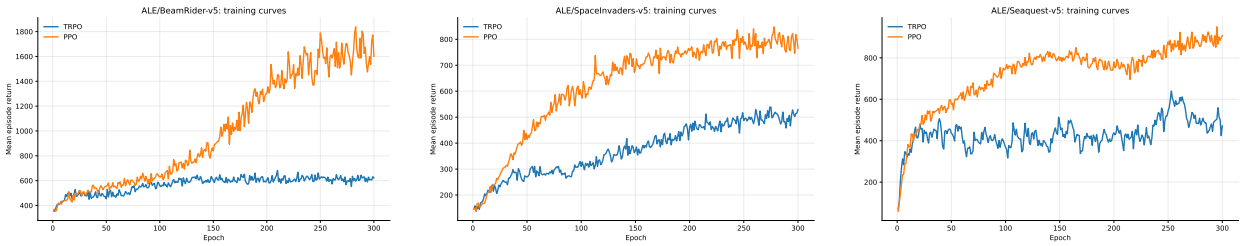
Table 3: Atari results for completed informative runs. Each listed game has one run per method. Returns are seed-0 training rollout means. Pong TRPO is partial at epoch 252; the other rows are 300-epoch runs. Enduro is omitted because both methods remained at zero return, and Breakout was configured but not run in this batch. Mean KL is averaged over epochs.

Game	TRPO final	PPO final	TRPO best	PPO best	TRPO KL	PPO KL
BeamRider	622.1	1601.7	682.2	1838.4	0.00509	0.00043
SpaceInvaders	528.9	765.6	539.2	848.0	0.00618	0.00042
Seaquest	471.1	907.7	640.0	949.8	0.00766	0.00044
Qbert	777.0	722.8	852.2	837.9	0.00837	0.00037
Pong	-4.0	-20.2	-2.5	-19.9	0.00687	0.00020

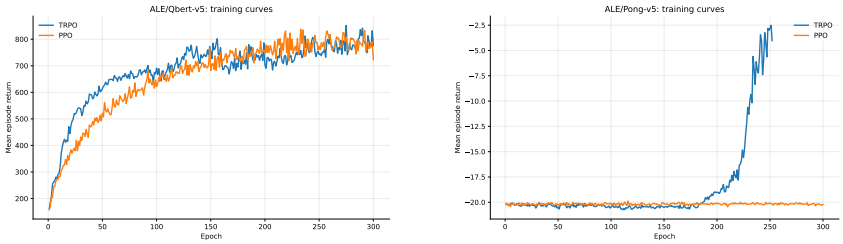
PPO-Clip is stronger on BeamRider, Space Invaders, and Seaquest. Qbert is close, and Pong favors the partial TRPO run. These results do not settle Atari performance, but they do show that the PPO implementation can extract more return from the same compact policy class on several image-based tasks.

6.5 Discussion and Limitations

The experiments support two conclusions. First, the TRPO–NPG comparison shows the value of controlling step length. In our implementation the search direction is essentially shared, but fixed-step NPG is either too timid or, at larger multipliers, more erratic. TRPO is more reliable because it tests the proposed step against the surrogate and KL constraint before accepting it. Second, PPO is the strongest practical method in these runs. On Hopper and Walker2d it obtains better returns with one-tenth the environment interactions used by TRPO/NPG, which is the clearest empirical sample-efficiency result of the project. This should be interpreted as a comparison between training setups, not as proof that clipping alone explains the gap: PPO also uses GAE, a learned critic, and repeated minibatch updates. The study is still limited by compute. The Atari experiments use one run per listed game and a small CNN; some games would likely need longer training or a more standard Atari PPO schedule. A larger follow-up could add more seeds, separate evaluation rollouts, and more extensive Atari tuning, and the codebase is already set up for those extensions.



(a) BeamRider (b) Space Invaders (c) Seaquest



(d) Qbert (e) Pong

Figure 3: Atari learning curves comparing single-path TRPO and PPO-Clip under the shared compact CNN policy class. Curves show the available seed-0 training rollout returns.

7 Conclusion

This project studied policy optimization as the problem of turning local policy-gradient information into reliable policy improvement. The performance difference lemma and surrogate objective make the central difficulty explicit: a policy update may look good under the old visitation distribution while changing the future states that are visited. CPI controls this with conservative mixtures, NPG with the Fisher geometry of policy space, TRPO with a KL trust region, and PPO with a clipped first-order surrogate. Our experiments reflect the same progression: TRPO improves over fixed-step NPG by controlling step size, while PPO performs strongly as a modern actor-critic training setup and, on Hopper and Walker2d, achieves better returns with one-tenth the environment interactions. Progress beyond vanilla policy gradients came not from abandoning policy gradients, but from controlling how their updates are estimated, scaled, constrained, and reused.

References

Achiam, Joshua (2017). *Advanced Policy Gradient Methods*. CS 294: Deep Reinforcement Learning, Fall 2017, University of California–Berkeley; OpenAI.

Amari, Shun-ichi (1998). "Natural Gradient Works Efficiently in Learning". In: *Neural Computation* 10.2, pp. 251–276.

Bagnell, J. Andrew and Jeff G. Schneider (2003). "Covariant Policy Search". In: *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pp. 1019–1024.

Kakade, Sham (2001). "A Natural Policy Gradient". In: *Advances in Neural Information Processing Systems*. MIT Press, pp. 1057–1063.

Kakade, Sham and John Langford (2002). "Approximately Optimal Approximate Reinforcement Learning". In: *ICML*. Vol. 2, pp. 267–274.

Schulman, John et al. (2015). "Trust Region Policy Optimization". In: *Proceedings of the International Conference on Machine Learning*. Vol. 37. JMLR Workshop and Conference Proceedings.
Schulman, John et al. (2016). "High-Dimensional Continuous Control Using Generalized Advantage Estimation". In: *Proceedings of the International Conference on Learning Representations (ICLR)*.
Schulman, John et al. (2017). "Proximal Policy Optimization Algorithms". In: *arXiv preprint arXiv:1707.06347*.
Sutton, Richard S. and Andrew G. Barto (2018). "Reinforcement Learning: An Introduction". In: 2nd ed. Chapter title: Policy Gradient Methods. Cambridge, MA: MIT Press. Chap. 13, pp. 321–337.
Sutton, Richard S. et al. (1999). "Policy Gradient Methods for Reinforcement Learning with Function Approximation". In: *Advances in Neural Information Processing Systems 12*, pp. 1057–1063.
Williams, Ronald J. (1992). "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning". In: *Machine Learning* 8.3–4, pp. 229–256.
Xie, Tengyang (2026). *Mathematical Principles of Reinforcement Learning*. Lecture notes for CS 839, University of Wisconsin–Madison. Chapter 10: Natural Policy Gradient.

A Implementation Details

The reported returns come from the per-epoch training batches recorded in `metrics.csv`. We report both final-epoch return and best training return, since several runs discover better policies before the final epoch. For MuJoCo, environments are created directly through Gymnasium and seeded at reset; observation normalization was disabled in the completed runs. For Atari, we use ALE v5 with sticky actions disabled, preprocessing frame skip 4, up to 30 no-op resets, 84×84 grayscale observations, and four-frame stacks. Life loss is not treated as terminal, and Atari observations are stored as raw pixels and divided by 255 inside the CNN body.

TRPO and PPO use the same Atari policy body: a 16-channel 8×8 convolution with stride 4, a 16-channel 4×4 convolution with stride 2, and a 20-unit fully connected layer. PPO adds a separate critic with the same CNN body and a scalar output head. For MuJoCo, TRPO and NPG use the smaller MLP architectures from the original TRPO continuous-control experiments: one hidden layer of 30 units for Swimmer and one hidden layer of 50 units for Hopper and Walker2d. PPO uses one hidden layer of 30 units for Swimmer and two hidden layers of 64 units for Hopper and Walker2d, so Hopper and Walker2d compare complete training setups rather than identical policy parameterizations.

The single-path TRPO and NPG runs use Monte Carlo returns as policy weights. PPO uses GAE advantages, fits a separate critic to discounted returns, and normalizes advantages. The logged `train_return_mean` is the mean return over episodes completed inside the training batch for that epoch. The logged `approx_kl` is computed after the policy update against the old policy used for that update.

Table 4: Main training hyperparameters used in our experiments.

Setting	MuJoCo TRPO/NPG	MuJoCo PPO	Atari TRPO	Atari PPO
Methods	TRPO, NPG	PPO-Clip	TRPO	PPO-Clip
Seeds completed	3 per task	3 per task	1 per game	1 per game
Epochs	200	200	300 current runs	300
Steps/epoch	Swimmer: 100k; Hopper/Walker2d: 1M	100k	100k	100k
Total env steps	Swimmer: 20M; Hopper/Walker2d: 200M	20M	30M completed runs	30M completed runs
Estimator	Monte Carlo returns	GAE	Monte Carlo returns	GAE
Discount γ	0.99	0.99	0.99	0.99
GAE/return λ	1.0	0.95	1.0	0.95
TRPO max KL δ	0.01	–	0.01	–
CG iterations	10	–	10	–
CG/FVP damping	0.1	–	0.1	–
Backtracking coeff.	0.8	–	0.8	–
Baseline NPG step size	Swimmer: 0.05; Hopper: 0.015; Walker2d: 0.005	–	–	–
PPO clip ratio	–	0.2	–	0.1, linearly annealed
PPO policy LR	–	$3 \cdot 10^{-4}$	–	$2.5 \cdot 10^{-4}$, annealed
PPO value LR	–	10^{-3}	–	$2.5 \cdot 10^{-4}$, annealed
PPO epochs/update	–	10	–	4
PPO minibatch size	–	2048	–	2048
PPO target KL	–	0.01	–	0.01
Entropy coefficient	–	0	–	0.01