# A Brief Survey of Policy Optimization Methods in Reinforcement Learning

**DDA4230 Final Project**
HU Sitian
121090193@link.cuhk.edu.cn
School of Data Science, The Chinese University of Hong Kong, Shenzhen

## Abstract

Policy optimization methods play a fundamental role in reinforcement learning, particularly for problems involving continuous action spaces and stochastic decision-making. Among these methods, Proximal Policy Optimization (PPO) has emerged as a widely used algorithm due to its favorable balance between stability, simplicity, and empirical performance. This paper reviews PPO and its related policy optimization approaches, tracing their development from trust-region based methods and examining representative extensions. To contextualize PPO within the broader landscape of policy optimization, deterministic policy gradient methods and recent preference-based optimization techniques are also discussed. The review highlights the core design principles, practical trade-offs, and open challenges that shape modern policy optimization.

## 1 Introduction

Policy optimization has become a central paradigm in modern reinforcement learning, particularly in domains involving continuous action spaces, high-dimensional observations, and complex decision-making requirements (11; 10; 1). By directly parameterizing and optimizing policies, policy optimization methods avoid many of the limitations of value-based approaches and have demonstrated strong empirical performance in robotics, control, and large-scale learning systems (5; 8; 9).

Despite their conceptual simplicity, early policy gradient methods suffer from significant practical challenges. In particular, unconstrained policy updates can lead to instability and performance collapse when policies change too aggressively between iterations (7). This issue has motivated a line of research focused on stabilizing policy optimization through principled constraints and surrogate objectives. Trust Region Policy Optimization (TRPO) represents a foundational effort in this direction, introducing explicit constraints on policy updates to guarantee monotonic improvement under certain assumptions (7).

Building on the theoretical insights of trust-region methods, Proximal Policy Optimization (PPO) was proposed as a simpler and more scalable alternative (8). PPO replaces explicit constrained optimization with a clipped surrogate objective that approximates trust-region behavior using first-order methods. Owing to its balance between stability, simplicity, and performance, PPO has become one of the most widely adopted policy optimization algorithms in practice (4). Subsequent extensions, such as Group Relative Policy Optimization (GRPO), further improve training robustness by modifying advantage normalization while retaining the core PPO framework (13).

To better understand the strengths and limitations of PPO-style methods, it is also instructive to examine alternative policy optimization paradigms. Deterministic policy gradient methods, including DPG and DDPG, offer a contrasting approach that emphasizes off-policy learning and deterministic policies for continuous control (10; 5; 9). More recently, preference-based optimization methods such as Direct Preference Optimization (DPO) have challenged the necessity of reinforcement learning altogether in certain alignment settings by directly optimizing policies from preference data (6; 12).

This paper provides a structured review of PPO and its related methods. We begin by introducing the fundamental concepts underlying policy optimization, including the actor-critic framework and key design choices such as policy stochasticity and data usage. We then review representative algorithms, tracing the evolution from trust-region methods to PPO and its extensions, and contrasting them with deterministic and preference-based approaches. Finally, we discuss the trade-offs among these methods and outline open challenges and future research directions.

## 2 Policy Optimization Background

Policy optimization methods aim to directly parameterize and optimize a policy that maximizes expected cumulative reward (11; 10). Unlike value-based approaches, which derive a policy indirectly from estimated value functions, policy optimization treats the policy itself as the primary object of learning. This perspective is particularly advantageous in environments with continuous action spaces or where stochastic policies are desirable for exploration and robustness (5; 8).

Most modern policy optimization algorithms are formulated within the actor-critic framework (11; 8). In this setting, the policy (actor) is represented by a parameterized distribution $\pi_\theta(a \mid s)$, while a value function (critic) estimates either the state value $V_\phi(s)$ or the action-value $Q_\phi(s, a)$. The critic provides low-variance learning signals that guide policy updates, enabling more stable and sample-efficient optimization than Monte Carlo policy gradient methods. This separation of roles forms the conceptual foundation upon which algorithms such as TRPO, PPO, and their variants are built (7; 8).

From an optimization standpoint, policy gradient methods seek to maximize the expected return (11; 8)

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[ \sum_{t=0}^{T} \gamma^t r_t \right], \tag{1}$$

where $\tau$ denotes a trajectory sampled from the policy. The policy gradient theorem expresses the gradient of this objective as (11)

$$\nabla_\theta J(\theta) = \mathbb{E} \left[ \nabla_\theta \log \pi_\theta(a_t \mid s_t)\, A^\pi(s_t, a_t) \right], \tag{2}$$

where the advantage function $A^\pi(s, a)$ measures the relative quality of an action compared to the policy's average behavior. Actor-critic methods estimate this advantage using a learned value function, reducing variance and improving learning stability (8; 5).

A fundamental distinction among policy optimization algorithms lies in whether they operate in an on-policy or off-policy manner. On-policy methods, such as TRPO and PPO, require data collected from the current policy and typically offer strong stability guarantees at the cost of lower sample efficiency (7; 8). Off-policy methods, including deterministic policy gradient approaches, can reuse past experience through replay buffers, enabling greater data efficiency but often introducing additional instability due to distribution mismatch (10; 5).

Another key design choice concerns the form of the policy itself. Stochastic policies explicitly model action uncertainty and naturally support exploration through entropy regularization, making them well-suited for complex or partially observable environments (8; 4). Deterministic policies, in contrast, map states directly to actions and are commonly adopted in continuous control settings, where they enable low-variance gradient estimation (10; 5). These two paradigms give rise to distinct algorithmic families, including stochastic policy optimization methods such as PPO and deterministic policy gradient methods such as DPG and DDPG.

Together, these considerations motivate the development of stable and scalable policy optimization algorithms. Early trust-region methods address the instability of unconstrained policy updates (7), while subsequent approaches seek to simplify optimization and improve practical performance (8; 4). In the following sections, we review representative algorithms that embody these ideas, beginning with trust-region based policy optimization.

## 3 Core Policy Optimization Methods

### 3.1 Trust Region Policy Optimization (TRPO)

Trust Region Policy Optimization (TRPO) was proposed by Schulman et al. (2015) (7) as a principled solution to a fundamental instability problem in policy gradient methods. While vanilla policy gradient algorithms estimate unbiased gradients of the expected return (11), their practical performance often suffers from destructively large policy updates when multiple gradient steps are taken using the same batch of trajectories (7; 8). This phenomenon arises because standard policy gradient objectives do not explicitly control how far the updated policy deviates from the policy that generated the data.

TRPO is grounded in policy improvement theory. The starting point of the analysis is an exact identity that relates the performance of a new policy $\tilde{\pi}$ to an old policy $\pi$ via the advantage function:

$$\eta(\tilde{\pi}) = \eta(\pi) + \sum_s \rho_{\tilde{\pi}}(s) \sum_a \tilde{\pi}(a \mid s) A^{\pi}(s,a) \tag{3}$$

where $\rho_{\tilde{\pi}}(s)$ denotes the discounted state visitation frequency under the new policy (7). Since $\rho_{\tilde{\pi}}$ depends on $\tilde{\pi}$ in a highly non-linear manner, direct optimization of this expression is intractable. TRPO therefore introduces a local surrogate objective by replacing $\rho_{\tilde{\pi}}$ with $\rho_{\pi}$,

$$L_{\pi}(\tilde{\pi}) = \eta(\pi) + \sum_s \rho_{\pi}(s) \sum_a \tilde{\pi}(a \mid s) A^{\pi}(s,a) \tag{4}$$

This theoretical result motivates constraining the policy update so that the new policy remains close to the old one in terms of a divergence measure (7). Specifically, Schulman et al. show that the discrepancy between the true objective $\eta(\tilde{\pi})$ and the surrogate objective $L_{\pi}(\tilde{\pi})$ can be bounded using the maximum Kullback-Leibler (KL) divergence between the two policies. As a consequence, ensuring a small KL divergence guarantees that improvements in the surrogate objective translate into actual performance gains.

Based on this insight, TRPO formulates policy optimization as a constrained maximization problem, where the surrogate objective is optimized subject to a trust region constraint defined by the expected KL divergence:

$$\begin{aligned} \max_{\tilde{\pi}} \quad & \mathbb{E}_{s\sim\rho_{\pi},\, a\sim\pi}\left[\frac{\tilde{\pi}(a \mid s)}{\pi(a \mid s)} A^{\pi}(s,a)\right], \\ \text{s.t.} \quad & \mathbb{E}_{s\sim\rho_{\pi}}\big[D_{\mathrm{KL}}\big(\pi(\cdot \mid s) \,\|\, \tilde{\pi}(\cdot \mid s)\big)\big] \le \delta, \end{aligned} \tag{5}$$

where $\delta$ is a small positive constant that specifies the size of the trust region (7).

In practice, this constrained problem is solved by approximating the KL divergence using a second-order Taylor expansion, which leads to the Fisher information matrix. The resulting update direction is computed using the conjugate gradient method, followed by a backtracking line search to ensure that the KL constraint is satisfied (7). The complete TRPO algorithm is summarized in Algorithm 1.

**Algorithm 1:** Trust Region Policy Optimization (TRPO)

```
for iteration = $1, 2, \ldots$ do
    Collect trajectories using policy $\pi_{\theta_{\text{old}}}$;
    Estimate advantages $\hat{A}_t$;
    Compute policy gradient and Fisher information matrix;
    Solve constrained optimization using conjugate gradient;
    Perform line search to satisfy KL constraint;
    Update policy parameters $\theta_{\text{old}} \leftarrow \theta$;
```

TRPO has been extensively applied in tasks with continuous action spaces, particularly in high-dimensional control problems such as robotics (5; 4). Its strength lies in the theoretical guarantee of monotonic policy improvement, which stabilizes learning by constraining the divergence between successive policies. Nonetheless, the method involves second-order optimization and computation of the Fisher information matrix, which increases computational cost and complicates implementation, limiting its practical scalability in some settings (7; 8).

### 3.2 Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO) was proposed by Schulman et al. (8) as a first-order alternative to TRPO (7) that retains its stability while significantly simplifying implementation. Instead of enforcing an explicit trust region constraint through second-order optimization, PPO introduces a surrogate objective that implicitly restricts policy updates using a clipping mechanism (11; 5).

As in TRPO, PPO considers the probability ratio between the new policy $\pi_\theta$ and the old policy $\pi_{\theta_{\text{old}}}$:

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \tag{6}$$

The basic surrogate objective used in conservative policy iteration is given by

$$L^{\text{CPI}}(\theta) = \mathbb{E}_t \left[ r_t(\theta) \hat{A}_t \right], \tag{7}$$

where $\hat{A}_t$ denotes an estimator of the advantage function. While maximizing this objective improves policy performance locally, repeated gradient updates can result in excessively large policy changes, thereby violating the assumptions underlying policy improvement (7).

To address this issue, PPO introduces a clipped surrogate objective that explicitly limits how much the probability ratio $r_t(\theta)$ is allowed to deviate from unity (8):

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[ \min \Big( r_t(\theta)\hat{A}_t,\ \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t \Big) \right]. \tag{8}$$

This objective can be interpreted as a pessimistic bound on the unclipped objective, discouraging policy updates that move too far from the data-collecting policy. Unlike TRPO, PPO does not enforce a hard constraint on the KL divergence; instead, it relies on the clipping operation to implicitly maintain proximity between successive policies (4).

PPO is typically implemented within an actor-critic framework (5), where the policy network (actor) is optimized using the clipped surrogate objective, and a value function (critic) is trained to approximate the state value $V(s)$. Advantage estimates $\hat{A}_t$ are commonly computed using Generalized Advantage Estimation (GAE) (7), which defines

$$\hat{A}_t^{\text{GAE}(\lambda)} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \left( r_{t+l} + \gamma V(s_{t+l+1}) - V(s_{t+l}) \right). \tag{9}$$

The final PPO objective combines the clipped policy loss, a value function regression loss, and an entropy bonus to encourage exploration (8):

$$L(\theta) = \mathbb{E}_t \left[ L_t^{\text{CLIP}}(\theta) - c_1 (V_\phi(s_t) - V_t^{\text{target}})^2 + c_2 \mathcal{H}(\pi_\theta(\cdot \mid s_t)) \right]. \tag{10}$$

The overall PPO algorithm alternates between data collection and multiple epochs of stochastic gradient ascent on the surrogate objective using minibatches (8; 4), allowing for efficient reuse of sampled trajectories while maintaining stable learning dynamics. The complete procedure is summarized in Algorithm 2.

**Algorithm 2:** Proximal Policy Optimization (PPO)

```
for iteration = $1, 2, \dots$ do
    for actor = $1, 2, \dots, N$ do
        Run policy $\pi_{\theta_{\text{old}}}$ for $T$ timesteps;
        Compute advantage estimates $\hat{A}_1, \dots, \hat{A}_T$;
    Optimize surrogate objective $L$ with $K$ epochs;
    Update policy parameters $\theta_{\text{old}} \leftarrow \theta$;
```

PPO offers a practical alternative to TRPO, combining stable policy updates with computational efficiency (8). By employing a clipped surrogate objective, PPO maintains proximity between consecutive policies without the need for second-order optimization, simplifying implementation while retaining stability. It has been successfully applied across a wide range of domains, including simulated environments, robotics, and game-playing agents (5; 4). Despite these advantages, PPO relies on heuristic clipping, which may not always provide an optimal constraint, and its performance is sensitive to hyperparameter tuning. Nevertheless, its balance of simplicity, stability, and empirical performance has made it a widely adopted method in modern reinforcement learning research.

### 3.3 Grouped Relative Policy Optimization (GRPO)

While Proximal Policy Optimization achieves a favorable balance between theoretical grounding and practical simplicity (8), its performance can still degrade in large-scale or highly heterogeneous training settings (4). In particular, PPO relies on mini-batch stochastic gradient updates that implicitly assume homogeneous advantage statistics across samples. When this assumption is violated, for example due to varying episode lengths, reward scales, or task difficulty, policy updates may exhibit increased variance and reduced sample efficiency.

Grouped Relative Policy Optimization (GRPO) extends PPO by explicitly introducing a grouping mechanism into the policy optimization process (13). Rather than treating all collected samples as a single pool, GRPO partitions trajectories or time steps into groups according to predefined criteria, such as temporal segments, reward magnitude, task identity, or environment instances. Policy updates are then performed relative to group-specific baselines, yielding more structured and stable gradient estimates.

Formally, GRPO retains the clipped surrogate objective of PPO (8), but modifies the advantage normalization and aggregation procedure. For a given group $g$, the group-relative advantage is defined as

$$\hat{A}_t^{(g)} = \hat{A}_t - \frac{1}{|\mathcal{G}_g|} \sum_{t' \in \mathcal{G}_g} \hat{A}_{t'}, \tag{11}$$

where $\mathcal{G}_g$ denotes the set of time steps belonging to group $g$.

Using group-relative advantages, the GRPO objective is defined as

$$L^{\text{GRPO}}(\theta) = \mathbb{E}_t \left[ \min \left( r_t(\theta) \hat{A}_t^{(g)},\ \text{clip}\big(r_t(\theta), 1-\epsilon, 1+\epsilon\big) \hat{A}_t^{(g)} \right) \right], \tag{12}$$

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

From an implementation perspective, GRPO fits naturally within the actor-critic framework used by PPO (8). The key difference lies in how advantages are processed prior to policy optimization, making GRPO a lightweight yet effective extension rather than a fundamentally new optimization scheme.

**Algorithm 3:** Grouped Relative Policy Optimization (GRPO)

```
for iteration = $1, 2, \ldots$ do
    Collect trajectories using policy $\pi_{\theta_{\text{old}}}$;
    Compute advantage estimates $\hat{A}_t$ using GAE (8);
    Partition samples into groups $\{\mathcal{G}_g\}$;
    Compute group-relative advantages $\hat{A}_t^{(g)}$;
    Optimize clipped surrogate objective using $\hat{A}_t^{(g)}$;
    Update policy parameters $\theta_{\text{old}} \leftarrow \theta$;
```

By introducing group-wise normalization, GRPO reduces gradient variance induced by heterogeneous data distributions and improves training stability in large-batch or multi-task settings (13). Empirically, this modification can lead to faster convergence and improved robustness without incurring significant additional computational cost. Compared to PPO, GRPO preserves the algorithmic simplicity and ease of implementation while offering finer-grained control over policy updates (8; 4). Unlike TRPO (7), it avoids second-order optimization and explicit KL constraints, and unlike deterministic methods such as DDPG (5), it maintains stochastic policies that naturally support exploration.

## 4 Deterministic Policy Gradient Methods

### 4.1 Deterministic Policy Gradient (DPG)

Deterministic Policy Gradient (DPG) was introduced by Silver et al. (10) as an extension of policy gradient methods to deterministic policies, motivated by the inefficiency of stochastic policy gradients in continuous action spaces. While stochastic policies require integrating over the entire action distribution, deterministic policies directly map states to actions, enabling more efficient gradient estimation when the action dimension is high.

In DPG, the policy is parameterized as a deterministic function $\mu_\theta(s)$, which specifies a single action for each state. The objective is to maximize the expected return under this policy,

$$J(\theta) = \mathbb{E}_{s \sim \rho^\mu} \left[ Q^\mu(s, \mu_\theta(s)) \right], \tag{13}$$

where $\rho^\mu(s)$ denotes the discounted state visitation distribution induced by the deterministic policy $\mu$. Unlike stochastic policy gradients, which differentiate through the policy's action probabilities, DPG leverages the structure of deterministic policies to derive a more direct gradient expression.

Silver et al. (10) establish the Deterministic Policy Gradient Theorem, which states that the gradient of the objective can be written as

$$\nabla_\theta J(\theta) = \mathbb{E}_{s \sim \rho^\mu} \left[ \nabla_\theta \mu_\theta(s) \nabla_a Q^\mu(s, a) \big|_{a=\mu_\theta(s)} \right]. \tag{14}$$

This result is significant because it eliminates the expectation over actions that appears in stochastic policy gradients, replacing it with a gradient of the action-value function with respect to the action. As a consequence, the variance of the gradient estimator does not scale with the dimensionality of the action space, making DPG particularly well suited for continuous control problems.

In practice, the action-value function $Q^\mu(s, a)$ is unknown and must be approximated. DPG therefore adopts an actor-critic architecture, where the critic learns an approximation $Q_\phi(s, a)$ using temporal-difference learning (11), and the actor updates its parameters by backpropagating through the critic. The critic is trained to minimize the Bellman error,

$$\mathcal{L}(\phi) = \mathbb{E}_{(s,a,r,s')} \left[ \left( Q_\phi(s, a) - \left( r + \gamma Q_\phi(s', \mu_\theta(s')) \right) \right)^2 \right]. \tag{15}$$

The actor update then follows directly from the deterministic policy gradient theorem, using the learned critic to compute $\nabla_a Q_\phi(s, a)$ (5). Importantly, the DPG framework supports off-policy learning, allowing experience collected under a different behavior policy to be reused for both actor and critic updates. Algorithm 4 summarizes the DPG procedure in an actor-critic setting.

**Algorithm 4:** Deterministic Policy Gradient (DPG)

```
for iteration = $1, 2, \ldots$ do
    Collect transitions $(s_t, a_t, r_t, s_{t+1})$ using behavior policy;
    Update critic by minimizing Bellman error;
    Update actor using deterministic policy gradient: $\nabla_\theta J(\theta) = \nabla_\theta \mu_\theta(s) \nabla_a Q_\phi(s, a)\big|_{a=\mu_\theta(s)}$;
```

DPG provides a theoretically grounded and computationally efficient alternative to stochastic policy gradient methods for continuous control (10; 5). By avoiding action sampling in the gradient computation, it achieves lower variance updates and improved scalability with respect to action dimensionality. These properties make DPG an attractive foundation for robotics and physical control tasks. However, deterministic policies also introduce limitations. Because exploration cannot be achieved through inherent policy stochasticity, DPG relies on externally injected noise in the behavior policy, which can be difficult to tune and may lead to insufficient state-space coverage. Moreover, the accuracy of the policy gradient critically depends on the quality of the learned critic, making the algorithm sensitive to function approximation errors. These challenges motivate subsequent extensions that incorporate deep neural networks, experience replay, and target networks to stabilize training. Most notably, Deep Deterministic Policy Gradient (DDPG) builds directly upon the DPG framework (5), adapting it to high-dimensional state spaces and forming a widely used baseline for continuous control.

### 4.2 Deep Deterministic Policy Gradient (DDPG)

Deep Deterministic Policy Gradient (DDPG) was proposed by Lillicrap et al. (5) to address the practical limitations of applying the deterministic policy gradient framework (10) to high-dimensional, continuous control problems. While DPG provides an elegant theoretical formulation for deterministic policies, its original formulation assumes access to accurate action-value gradients and stable function approximation (11), assumptions that break down when both the policy and value function are represented by deep neural networks.

The core contribution of DDPG lies in stabilizing deterministic policy gradient learning under deep function approximation. Building directly upon the actor-critic structure of DPG, DDPG introduces two key mechanisms adopted from deep Q-learning (4): experience replay and target networks. These additions are crucial for mitigating the instability caused by correlated data and rapidly changing targets during training.

As in DPG, the actor in DDPG is a deterministic policy $\mu_\theta(s)$, and the critic approximates the action-value function $Q_\phi(s, a)$. The critic is trained using a temporal-difference objective, but with a slowly updated target network to define the bootstrap target (5),

$$\mathcal{L}(\phi) = \mathbb{E}_{(s,a,r,s') \sim \mathcal{D}} \left[ \left( Q_\phi(s, a) - \left( r + \gamma Q_{\phi^-}(s', \mu_{\theta^-}(s')) \right) \right)^2 \right], \tag{16}$$

where $\phi^-$ and $\theta^-$ denote the parameters of the target critic and target actor, respectively, and $\mathcal{D}$ is a replay buffer containing past transitions. By decoupling the target values from the rapidly changing online networks, DDPG significantly improves the stability of critic learning.

The actor update follows the deterministic policy gradient theorem (10), using the learned critic to guide policy improvement. Specifically, the policy parameters are updated according to

$$\nabla_\theta J(\theta) = \mathbb{E}_{s \sim \mathcal{D}} \left[ \nabla_\theta \mu_\theta(s) \nabla_a Q_\phi(s, a) \big|_{a = \mu_\theta(s)} \right]. \tag{17}$$

Unlike stochastic policy optimization methods such as TRPO (7) and PPO (8), exploration in DDPG is not induced by policy stochasticity. Instead, exploration noise is added explicitly to the actor during data collection, typically using temporally correlated processes such as Ornstein-Uhlenbeck noise (5).

Target networks are updated using soft updates to ensure smooth evolution (5),

$$\theta^- \leftarrow \tau\theta + (1 - \tau)\theta^-, \quad \phi^- \leftarrow \tau\phi + (1 - \tau)\phi^-, \tag{18}$$

where $\tau \ll 1$ controls the update rate. Together with experience replay, this mechanism allows DDPG to reuse past experience efficiently while avoiding divergence caused by non-stationary targets. Algorithm 5 summarizes the complete DDPG procedure.

**Algorithm 5:** Deep Deterministic Policy Gradient (DDPG)

```
Initialize actor $\mu_\theta$ and critic $Q_\phi$ with random parameters;
Initialize target networks $\mu_{\theta^-} \leftarrow \mu_\theta$, $Q_{\phi^-} \leftarrow Q_\phi$;
Initialize replay buffer $\mathcal{D}$;
for episode $= 1, 2, \dots$ do
    Initialize exploration noise process;
    Observe initial state $s_0$;
    for $t = 0, 1, \dots, T$ do
        Select action $a_t = \mu_\theta(s_t) + \mathcal{N}_t$;
        Execute action $a_t$, observe $r_t, s_{t+1}$;
        Store $(s_t, a_t, r_t, s_{t+1})$ in $\mathcal{D}$;
        Sample minibatch from $\mathcal{D}$;
        Update critic by minimizing Bellman loss;
        Update actor using deterministic policy gradient;
        Update target networks using soft updates;
```

By integrating experience replay and target networks, DDPG transforms the deterministic policy gradient framework into a practical algorithm capable of handling high-dimensional state spaces and continuous actions (5). Nevertheless, DDPG inherits several fundamental limitations from deterministic policy optimization (4). Its performance is highly sensitive to hyperparameter choices, particularly the scale and structure of exploration noise. Furthermore, errors in the critic can be amplified through the deterministic actor updates, leading to brittle learning dynamics and poor robustness compared to trust-region-based methods such as PPO (8). These limitations have motivated a line of subsequent research aimed at improving stability and robustness, including Twin Delayed DDPG (TD3) (2) and Soft Actor-Critic (SAC) (3).

## 5 Preference-based Optimization Beyond Reinforcement Learning

### 5.1 Direct Preference Optimization (DPO)

Direct Preference Optimization (DPO) is a recently proposed optimization paradigm designed primarily for aligning large language models with human preferences (6). Unlike traditional reinforcement learning from human feedback (RLHF) (8), which relies on explicit reward modeling and policy optimization algorithms such as PPO (8), DPO reformulates preference optimization as a direct supervised learning problem.

The core motivation behind DPO is to eliminate the instability and complexity introduced by reward model training and reinforcement learning. In standard RLHF pipelines, a reward model is first trained to approximate human preferences, after which a policy is optimized using PPO. This multi-stage procedure is known to be sensitive to hyperparameters, reward hacking, and optimization instability. DPO addresses these issues by analytically deriving a closed-form objective that directly optimizes the policy using preference data (6).

Assume access to a dataset of human preferences over pairs of responses, denoted as $(x, y^+, y^-)$, where $y^+$ is preferred over $y^-$ given input $x$. DPO starts from the KL-regularized reinforcement learning objective and shows that the optimal policy can be expressed in terms of a reference policy $\pi_{\text{ref}}$. The resulting DPO objective is given by

$$\mathcal{L}_{\text{DPO}}(\theta) = \mathbb{E}_{(x,y^+,y^-)}\left[\log \sigma\left(\beta\left(\log \pi_\theta(y^+ \mid x) - \log \pi_\theta(y^- \mid x)\right) - \beta\left(\log \pi_{\text{ref}}(y^+ \mid x) - \log \pi_{\text{ref}}(y^- \mid x)\right)\right)\right],$$

where $\sigma(\cdot)$ denotes the sigmoid function and $\beta$ controls the strength of the implicit KL regularization (6).

Importantly, this formulation removes the need for an explicit reward model and avoids on-policy sampling altogether. Optimization reduces to standard supervised gradient descent, significantly improving training stability and computational efficiency. Conceptually, DPO can be viewed as collapsing the reward learning and policy optimization stages of RLHF into a single objective (6).

**Algorithm 6:** Direct Preference Optimization (DPO)

```
Input: Preference dataset $\mathcal{D} = \{(x, y^+, y^-)\}$, reference policy $\pi_{\text{ref}}$
Initialize policy parameters $\theta$;
for iteration $= 1, 2, \ldots$ do
    Sample minibatch from $\mathcal{D}$;
    Compute DPO loss $\mathcal{L}_{\text{DPO}}(\theta)$;
    Update policy parameters using gradient descent;
```

Empirically, DPO has demonstrated performance comparable to or better than PPO-based RLHF across multiple language model alignment benchmarks (6), while being significantly simpler to implement. However, DPO is inherently limited to settings where preference data is available and does not generalize to online reinforcement learning or continuous control tasks. As a result, DPO should be viewed not as a replacement for PPO (8), but as a complementary approach.

## 6 Discussion

The policy optimization methods analyzed in this paper represent a systematic evolution in reinforcement learning, moving from mathematically rigorous but computationally expensive constraints to more heuristic yet practical objectives. As illustrated in Table 1, the transition from TRPO to PPO and eventually GRPO signifies a clear trend toward balancing optimization stability with implementation simplicity (7; 8; 13). TRPO's reliance on second-order optimization provided theoretical guarantees of monotonic improvement, yet its high computational overhead prompted the development of PPO's clipped surrogate objective, which achieves similar trust-region behavior using only first-order gradients. GRPO further refines this by introducing group-relative benchmarks, effectively reducing variance without the need for a separate critic network, which is particularly advantageous in large-scale model alignment tasks.

Table 1: Detailed Comparison of Policy Optimization Methods

| Method | Policy Type | Data Usage | Computational Cost | Key Feature |
|---|---|---|---|---|
| TRPO | Stochastic | On-policy | High | KL-Constraint |
| PPO | Stochastic | On-policy | Medium | Clipped Objective |
| GRPO | Stochastic | On-policy | Medium | Group Normalization |
| DPG | Deterministic | Continuous | Off-policy | Low |
| DDPG | Deterministic | Off-policy | Medium | Actor-Critic |
| DPO | Stochastic | Offline | Low | Reward-free |

A critical trade-off identified across these methods is the tension between sample efficiency and training stability. Deterministic methods such as DDPG offer superior sample efficiency through off-policy learning and experience replay, making them suitable for continuous control tasks where data collection is expensive (5). However, they are notoriously sensitive to hyperparameter tuning and often suffer from overestimation bias in value functions. Conversely, stochastic on-policy methods like PPO and GRPO provide robust convergence properties but require significantly more environment interactions to reach optimal performance (4). This inefficiency remains a primary bottleneck for deploying such algorithms in high-fidelity physical simulations or real-world robotics.

Furthermore, the emergence of DPO represents a paradigm shift in how we approach policy optimization (6). By bypassing the traditional reward modeling and reinforcement learning loop, DPO reduces the problem to a supervised classification-like task. While this drastically lowers computational costs and avoids the instabilities of RL, it introduces a new dependency on static, high-quality preference datasets. This shift highlights a fundamental unsolved problem: the "exploration-alignment" gap. DPO cannot explore beyond the distribution of its training data, whereas traditional RL methods (PPO, DDPG) can discover novel strategies through active environment interaction. Consequently, the choice of algorithm must be dictated by the availability of an interactive simulator versus a fixed dataset of human feedback.

## 7 Conclusion

This paper has reviewed the core contributions of modern policy optimization, ranging from trust-region foundations to recent preference-driven frameworks. The main contribution of these methods is the establishment of reliable mechanisms for policy updates—whether through gradient clipping, deterministic mapping, or direct preference alignment, enabling RL to scale to high-dimensional spaces such as autonomous driving and large language models (LLMs) (9; 12).

Despite these advancements, significant limitations persist. Most notably, the high sample complexity of on-policy methods and the instability of off-policy value estimation remain major hurdles. Additionally, the problem of "reward hacking," where an agent optimizes for a flawed reward signal, and the "distributional drift" in offline methods like DPO, are yet to be fully resolved.

Future work should focus on developing hybrid architectures that integrate the sample efficiency of off-policy learning with the stability of trust-region constraints. Another promising direction is the incorporation of world models to allow agents to "dream" and plan in latent spaces, further reducing the need for real-world data. In terms of applications, these algorithms are poised to move beyond digital domains into complex real-world systems, including personalized healthcare, adaptive industrial process control, and collaborative robotics. Ultimately, the synthesis of preference-based learning and active exploration will be essential for creating truly robust and aligned autonomous agents.

## References

[1] Arulkumaran, K., Deisenroth, M. P., Brundage, M., & Bharath, A. A. (2017). A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866*.

[2] Fujimoto, S., van Hoof, H., & Meger, D. (2018). Addressing function approximation error in actor-critic methods. *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 1587–1596.

[3] Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 1861–1870.

[4] Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., & Meger, D. (2018). Deep reinforcement learning that matters. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

[5] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2016). Continuous control with deep reinforcement learning. *International Conference on Learning Representations (ICLR)*.

[6] Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C. D., & Finn, C. (2023). Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems (NeurIPS)*, 36, 18428–18441.

[7] Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015). Trust region policy optimization. *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 1889–1897.

[8] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

[9] Siboo, S., Bhattacharyya, A., Raj, R. N., & Ashwin, S. H. (2023). An empirical study of DDPG and PPO-based reinforcement learning algorithms for autonomous driving. *IEEE Access*, 11, 125094–125108.

[10] Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., & Riedmiller, M. (2014). Deterministic policy gradient algorithms. *Proceedings of the 31st International Conference on Machine Learning (ICML)*, 387–395.

[11] Sutton, R. S., McAllester, D., Singh, S., & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems (NeurIPS)*, 1057–1063.

[12] Wang, Z., Bi, B., Pentyala, S. K., Ramnath, K., Chaudhuri, S., Mehrotra, S., & Asur, S. (2024). A comprehensive survey of LLM alignment techniques: RLHF, RLAIF, PPO, DPO and more. *arXiv preprint arXiv:2407.16216*.

[13] Zhang, Y., Liu, H., & Zhang, Q. (2024). Group relative policy optimization for stable policy learning. *International Conference on Learning Representations (ICLR)*.