Reinforcement Learning

Deep RL

Matteo Papini

June 14, 2026

A Brief History of Deep RL

2013 **DQN** (Deep Q-Network) learns to play Atari games from pixels (Mnih et al., 2013, 2015)

2015 **DDPG** (Deep Deterministic Policy Gradient) (Lillicrap et al., 2015) solves continuous control problems

2015 **TRPO** (Trust-Region Policy Optimization) (Schulman et al., 2015)

2016 **A3C** (Asynchronous Advantage Actor-Critic) (Mnih et al., 2016)
$\rightarrow$ Later superseded by **A2C**, (Synchronous) Advantage Actor-Critic

2016 **AlphaGo** masters the game of Go (Silver et al., 2016) with MCTS + policy networks

2017 **PPO** (Proximal Policy Optimization) (Schulman et al., 2017), another trust-region method

2018 **SAC** (Soft Actor-Critic) (Haarnoja et al., 2018) shows significant improvement in continuous control

2018 DDPG strikes back with **TD3** (Fujimoto et al., 2018)

2022 PPO + **RLHF** behind InstructGPT (Ouyang et al., 2022) and later **ChatGPT**

2023 **REINFORCE** makes a comeback (Ahmadian et al., 2024)

2024 **GRPO** for mathematical reasoning (Shao et al., 2024)

Outline

Outline

Outline

A2C (Advantage Actor-Critic)

- First large-scale implementation of the Actor-Critic architecture (Barto et al., 1983)
- First proposed as **A3C** (**Asynchronous** Advantage Actor-Critic, Mnih et al., 2016)
- "Asynchronous" refers to the fact that many actors (policies) collect experience in parallel and report asynchronously to a central critic
- The synchronous version (**A2C**) turned out to make a more efficient use of GPUs `https://openai.com/index/openai-baselines-acktr-a2c/`

Advantage Estimation

Recall the "advantage" version of the Policy Gradient Theorem

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{1-\gamma} \mathbb{E}\left[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A|S) \mathbb{A}^{\pi_{\boldsymbol{\theta}}}(S,A)\right],$$

Where $\mathbb{A}^{\pi_{\boldsymbol{\theta}}}(S,A) = Q^{\pi_{\boldsymbol{\theta}}}(S,A) - V^{\pi_{\boldsymbol{\theta}}}(S)$ is the advantage function.

The main ingredient of A2C is **Advantage Estimation**

- Let $\delta_t = R_{t+1} + \gamma V^{\pi}(S_{t+1}) - V^{\pi}(S_t)$ be the TD (Temporal Difference) error
- Recall that $\mathbb{E}[\delta_t | S_t, A_t] = \mathbb{A}^{\pi}(S_t, A_t)$
- Train a state-critic $v_{\mathbf{w}}(s)$
- Estimate $\mathbb{A}^{\pi}(S_t, A_t)$ as $\delta_t^{\mathbf{w}} = R_{t+1} + \gamma v_{\mathbf{w}}(S_{t+1}) - v_{\mathbf{w}}(S_t)$
- Can be used to compute policy gradient sample $\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_t|S_t) \delta_t^{\mathbf{w}}$

Data collection of A2C

At each *iteration* (policy update), A2C collects a *rollout* using the current policy $\pi_{\boldsymbol{\theta}}$.

A rollout can be a single long trajectory or a sequence of variable-length episodes. Let T be the total size of the rollout and τ_t be the final timestep of the episode to which timestep t belongs.

Terminal states must be handled separately:

$$\delta_t^{\mathbf{w}} = \begin{cases} R_{t+1} + \gamma v_{\mathbf{w}}(S_{t+1}) - v_{\mathbf{w}}(S_t) & \text{if } t \neq \tau_t \\ R_{t+1} - v_{\mathbf{w}}(S_t) & \text{otherwise} \end{cases} \tag{1}$$

Critic targets (returns-to-go) can be computed for each $t = 0, \ldots, T-1$ as:

$$G_t = \sum_{k=t}^{\tau_t} \gamma^{k-t} R_{k+1}$$

A2C

Initialize the actor parameters $\boldsymbol{\theta}$ and the critic parameters $\mathbf{w}$

loop for every iteration

Collect a *rollout* of T total timesteps with $\pi_{\boldsymbol{\theta}}$

Compute returns-to-go G_t

Compute TD errors $\delta_t^{\mathbf{w}} = [\![R_{t+1} + \gamma v_{\mathbf{w}}(S_{t+1}) - v_{\mathbf{w}}(S_t)]\!]^*$ based on the current critic $v_{\mathbf{w}}$

Define the actor loss

$$\mathcal{L}_{\text{actor}}(\boldsymbol{\theta}) = -\sum_{t=0}^{T-1} \log \pi_{\boldsymbol{\theta}}(A_t|S_t)\delta_t^{\mathbf{w}}$$

Define the critic loss

$$\mathcal{L}_{\text{critic}}(\mathbf{w}) = \frac{1}{2}\sum_{t=0}^{T-1} \left(v_{\mathbf{w}}(S_t) - G_t\right)^2$$

Minimize $\mathcal{L}_{\text{actor}}(\boldsymbol{\theta}) + \mathcal{L}_{\text{critic}}(\mathbf{w})$ w.r.t. $\boldsymbol{\theta}$ and $\mathbf{w}$ with Adam (gradient descent)

end loop

* brackets denote stop-gradient

On-Policy vs Off-Policy Deep RL

- In **A2C**, the data (rollout) is discarded after each iteration (policy update)
- This makes A2C **on-policy**
- The same holds for the trust-region methods **TRPO, PPO, GRPO** (single-epoch versions)
- In contrast, the replay buffer of **DQN** is used across several iterations (subject to buffer size)
- This makes DQN **off-policy**
- The same holds for **DDPG**, **TD3** and **SAC**

Generalized Advantage Estimation (GAE, Schulman et al., 2016)

Recall k-step returns:

- $v_t^{(1)} = R_{t+1} + \gamma v_{\mathbf{w}}(S_{t+1})$
- $v_t^{(2)} = R_{t+1} + \gamma R_{t+1} + \gamma^2 v_{\mathbf{w}}(S_{t+1})$
- ...
- $v_t^{(k)} = R_{t+1} + \cdots + \gamma^{k-1} R_{t+k} + \gamma^k v_{\mathbf{w}}(S_{t+1})$

Recall the λ-return ($0 < \lambda < 1$):

$$v_t^{\lambda} = \sum_{k=1}^{\infty} \lambda^{k-1} v_t^{(k)}$$

Replace the 1-step return with the λ-step return in advantage estimation:

$$\begin{aligned} \widehat{\mathbb{A}}_t^{\mathrm{GAE}} &= v_t^{\lambda} - v_{\mathbf{w}}(S_t) \\ &= \sum_{k=0}^{\infty} (\gamma\lambda)^k \delta_{t+k}^{\mathbf{w}} \qquad \text{(exercise: can you prove this equality?)} \end{aligned}$$

Like in TD(λ), λ controls the bias-variance trade-off

Data collection of A2C + GAE

At each *iteration* (policy update), A2C collects a *rollout* using the current policy $\pi_{\boldsymbol{\theta}}$.

A rollout can be a single long trajectory or a sequence of variable-length episodes. Let T be the total size of the rollout and τ_t be the final timestep of the episode to which timestep t belongs.

Advantage estimates can be computed for each $t = 0, \ldots, T-1$ as:

$$\widehat{\mathbb{A}}_t^{\mathrm{GAE}} = \sum_{k=t}^{\tau_t} (\gamma\lambda)^{k-t} \delta_k^{\mathbf{w}}$$

Critic targets (λ-returns) can be obtained simply as:

$$v_t^{\lambda} = \widehat{\mathbb{A}}_t^{\mathrm{GAE}} + v_{\mathbf{w}}(S_t) \tag{2}$$

A2C + GAE

Initialize the actor parameters $\boldsymbol{\theta}$ and the critic parameters $\mathbf{w}$

loop for every iteration

Collect a *rollout* of T total timesteps with $\pi_{\boldsymbol{\theta}}$

Compute TD errors $\delta_t^{\mathbf{w}} = [\![R_{t+1} + \gamma v_{\mathbf{w}}(S_{t+1}) - v_{\mathbf{w}}(S_t)]\!]$ based on the current critic $v_{\mathbf{w}}$

Compute advantage estimates $\widehat{\mathbb{A}}_t^{\mathrm{GAE}}$ and TD(λ) targets v_t^{λ}

Define the actor loss

$$\mathcal{L}_{\mathrm{actor}}(\boldsymbol{\theta}) = -\sum_{t=0}^{T-1} \log \pi_{\boldsymbol{\theta}}(A_t|S_t)\widehat{\mathbb{A}}_t^{\mathrm{GAE}}$$

Define the critic loss

$$\mathcal{L}_{\mathrm{critic}}(\mathbf{w}) = \frac{1}{2}\sum_{t=0}^{T-1} \left(v_{\mathbf{w}}(S_t) - v_t^{\lambda}\right)^2$$

Minimize $\mathcal{L}_{\mathrm{actor}}(\boldsymbol{\theta}) + \mathcal{L}_{\mathrm{critic}}(\mathbf{w})$ w.r.t. $\boldsymbol{\theta}$ and $\mathbf{w}$ via gradient descent

end loop

Outline

Deterministic Policy Gradient Theorem

- The policy gradient theorem only holds for **stochastic policies** (Sutton et al., 1999)

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{1-\gamma} \mathbb{E}_{\substack{S \sim d_{\pi_{\boldsymbol{\theta}}} \\ A \sim \pi_{\boldsymbol{\theta}}(\cdot|S)}} \left[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A|S) Q^{\pi_{\boldsymbol{\theta}}}(S, A) \right]$$

- Extended to **deterministic policies** by Silver et al. (2014)

Theorem (Deterministic Policy Gradient Theorem (Silver et al., 2014))

Let $\mu_{\boldsymbol{\theta}}(s)$ be a **deterministic** *policy* **differentiable** *in $\boldsymbol{\theta}$ and suppose that the Q-function is* **differentiable in the action** *argument. Then, it holds that:*

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{1-\gamma} \mathbb{E}_{S \sim d^{\mu_{\boldsymbol{\theta}}}} \left[\nabla_{\boldsymbol{\theta}} \mu_{\boldsymbol{\theta}}(S) \nabla_a Q^{\mu_{\boldsymbol{\theta}}}(S, a)|_{a=\mu_{\boldsymbol{\theta}}(S)} \right]$$

Deterministic Policy Gradient

- Deterministic Policy Gradient (DPG) is naturally actor-critic because it needs a critic $q_{\mathbf{w}}(s, a)$ that is **differentiable** w.r.t. a
- DPG is inherently **off-policy** because, the target policy being deterministic, it needs an explicit exploration mechanism
- Typically a **noisy policy** is used for exploration

$$A_t = \mu_{\boldsymbol{\theta}}(S_t) + \epsilon$$

where ϵ is noise sampled from a suitable distribution (e.g. zero-mean Gaussian)

(Deep) Deterministic Policy Gradient

Deep Deterministic Policy Gradient (DDPG): DPG with deep neural netwroks as policy and value approximators (Lillicrap et al., 2015)

- Replay buffer $\mathcal{B}$ to break dependencies
- Target network for both policy $\mu_{\boldsymbol{\theta}^-}(s)$ and value function $q_{\mathbf{w}^-}(s, a)$

DDPG Algorithm

Initialize the replay buffer $\mathcal{B}$ with capacity N
Initialize the actor parameters $\boldsymbol{\theta}$ and the critic parameters $\mathbf{w}$
Initialize the target actor parameters $\boldsymbol{\theta}^- \leftarrow \boldsymbol{\theta}$ and the target critic parameters $\mathbf{w}^- \leftarrow \mathbf{w}$
loop for every iteration
Observe the state S and select action $A = \mu_{\boldsymbol{\theta}}(S) + \epsilon$ where $\epsilon \sim \mathcal{N}$ is noise
Execute A and observe the next state S' and reward R
Store (S, A, R, S') in the replay buffer $\mathcal{B}$
Sample a random minibatch of transitions $(S_j, A_j, R_{j+1}, S_{j+1})$ from the replay buffer $\mathcal{B}$

Compute the targets

$$y_j = \begin{cases} R_{j+1} & \text{if } S_j \text{ is a terminal state} \\ R_{j+1} + \gamma q_{\mathbf{w}^-}(S_{j+1}, \mu_{\boldsymbol{\theta}^-}(S_{j+1})) & \text{otherwise} \end{cases}$$

Perform a gradient descent step on $(y_j - q_{\mathbf{w}}(S_j, A_j))^2$ w.r.t. the critic parameters $\mathbf{w}$
Perform a gradient ascent step on $q_{\mathbf{w}}(S_j, \mu_{\boldsymbol{\theta}}(S_j))$ w.r.t. the actor parameters $\boldsymbol{\theta}$
Update target networks $\boldsymbol{\theta}^- \leftarrow \rho\boldsymbol{\theta}^- + (1-\rho)\boldsymbol{\theta}$ and $\mathbf{w}^- \leftarrow \rho\mathbf{w}^- + (1-\rho)\mathbf{w}$
end loop

Twin Delayed Deep Deterministic Policy Gradient (TD3)

TD3 (Fujimoto et al., 2018) builds on DDPG introducing three main improvements:

- *Twin Critics (inspired by Double Q-Learning) to tackle overestimation bias*
  Train two critics $q^1_{\mathbf{w}}, q^2_{\mathbf{w}}$ (with their respective target networks $q^1_{\mathbf{w}^-}, q^2_{\mathbf{w}^-}$) and compute TD targets

$$y_j = \begin{cases} R_{j+1} & \text{if } S_j \text{ is terminal state} \\ R_{j+1} + \gamma \min_{i \in \{1,2\}} q^i_{\mathbf{w}^-}(S_{j+1}, \mu_{\boldsymbol{\theta}^-}(S_{j+1})) & \text{otherwise} \end{cases}$$

- *Delayed policy updates to reduce instability*
  Update the actor and both target networks at a lower frequency (every k iterations)
- *Target policy smoothing (inspired by Expected SARSA) to avoid overfitting by the deterministic policy*
  Add noise to actions in the target ($\epsilon \sim$ clipped Gaussian)

$$y_j = \begin{cases} R_{j+1} & \text{if } S_j \text{ is terminal state} \\ R_{j+1} + \gamma \min_{i \in \{1,2\}} q^i_{\mathbf{w}^-}(S_{j+1}, \mu_{\boldsymbol{\theta}^-}(S_{j+1}) + \epsilon) & \text{otherwise} \end{cases}$$

Outline

Entropy-Regularized RL

- **Entropy-regularized (or soft)** reward function (Nachum et al., 2017; Haarnoja et al., 2018)

$$r(s,a) + \lambda H(\pi(\cdot|s))$$

  where $\lambda > 0$ is the regularization parameter
- $H(\pi(\cdot|s))$ is the **entropy**

$$H(\pi(\cdot|s)) = -\sum_a \pi(a|s) \log \pi(a|s)$$

- The (differential Shannon) **entropy** is an index of uncertainty
  - Large $H(\pi(\cdot|s)) \to \pi(\cdot|s)$ "very stochastic"
  - Small $H(\pi(\cdot|s)) \to \pi(\cdot|s)$ "almost deterministic"
- Thus, we are encouraging π to be **stochastic**
- This "entropy bonus" can be added to most policy gradient algorithms

Entropy-Regularized RL

- Given a policy π, the **soft value functions** become

$$q_\pi^{\text{soft}}(s,a) \coloneqq r(s,a) + \mathbb{E}\left[\sum_{t=1}^{+\infty} \gamma^t \Big(r(S_t, A_t) + \lambda H(\pi(\cdot|S_t)) \Big) | S_0 = s, A_0 = a, \pi \right]$$

$$v_\pi^{\text{soft}}(s) \coloneqq \mathbb{E}_{A\sim\pi(\cdot|s)}\left[q_\pi^{\text{soft}}(s,A)\right] + \lambda H(\pi(\cdot|S_t))$$

  - When $\lambda \to 0$ they become the standard value functions

- The **greedy softmax policy** w.r.t. function $Q^{\text{soft}}(s,a)$ is defined as

$$\pi^{\text{soft}}(\cdot|s) = \arg\max_\pi \left\{ \mathbb{E}_{A\sim\pi(\cdot|s)}[q^{\text{soft}}(s,A)] + \lambda H(\pi(\cdot|s)) \right\} = \frac{\exp\left(\frac{q^{\text{soft}}(s,\cdot)}{\lambda}\right)}{\sum_a \exp\left(\frac{q^{\text{soft}}(s,a)}{\lambda}\right) \mathrm{d}a}$$

  - When $\lambda \to 0$ it becomes the **greedy** policy

Entropy-Regularized RL

- The state-value function induced by the greedy softmax policy π^{soft} w.r.t. q^{soft} is given by

$$\begin{aligned} v^{\text{soft}}(s) &= \mathbb{E}_{A\sim\pi^{\text{soft}}(\cdot|s)}[q^{\text{soft}}(s,A)] + \lambda H(\pi^{\text{soft}}(\cdot|s)) \\ &= \lambda \log \sum_a \exp\left(\frac{q^{\text{soft}}(s,a)}{\lambda}\right) \mathrm{d}a \end{aligned}$$

- We can rewrite the greedy softmax policy as

$$\pi^{\text{soft}}(a|s) = \exp\left(\frac{q^{\text{soft}}(s,a) - v^{\text{soft}}(s)}{\lambda}\right)$$

Theorem (Policy Improvement Theorem (Haarnoja et al., 2018))

Let $q_\pi^{soft}(s,a)$ be the soft Q-function of policy π. Let π^{soft} be the greedy softmax policy w.r.t. $q_\pi^{soft}(s,a)$. Then, it holds that:

$$q_{\pi^{soft}}^{soft}(s,a) \geq q_\pi^{soft}(s,a) \qquad \forall (s,a) \in \mathcal{S}\times\mathcal{A}$$

Soft Actor-Critic

- Designed for **continuous actions**
- **Actor-critic** approach in the entropy-regularized setting
  - The **actor** approximates the **greedy softmax policy** with a continuous-action (e.g., Gaussian) policy
  - The **critic** estimates the **soft Q-function** $q_{\mathbf{w}}(s, a)$
- The policy update is performed via KL **minimization**

$$\boldsymbol{\theta}' \in \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \mathbb{E}_{S \sim \mathcal{B}} \left[D_{\mathsf{KL}} \left(\pi_{\boldsymbol{\theta}}(\cdot|S) \middle\| \frac{\exp\left(\frac{q_{\mathbf{w}}(S,\cdot)}{\lambda}\right)}{\sum_a \exp\left(\frac{q_{\mathbf{w}}(S,a)}{\lambda}\right) \mathrm{d}a} \right) \right]$$

where $D_{\mathrm{KL}}(p\|q) = \sum_a p(a) \log \frac{p(a)}{q(a)}$ is the Kullback-Liebler divergence (measure of statistical distance) between distributions p and q

Soft Actor-Critic Algorithm

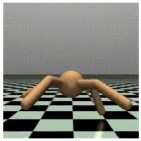
Initialize the replay buffer $\mathcal{B}$ with capacity N
Initialize the actor parameters $\boldsymbol{\theta}$ randomly
Initialize the critic parameters $\mathbf{w}$ randomly
Initialize the target critic parameters $\mathbf{w}^- \leftarrow \mathbf{w}$
loop for every iteration
Observe the state S and select action $A \sim \pi_{\boldsymbol{\theta}}(\cdot|S)$
Execute A and observe the next state S' and reward R
Store (S, A, R, S') in the replay buffer $\mathcal{B}$

Sample a random minibatch of transitions $(S_j, A_j, R_{j+1}, S_{j+1})$ from the replay buffer $\mathcal{B}$
Compute the targets

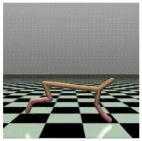
$$y_j = \begin{cases} R_{j+1} & \text{if } S_j \text{ is terminal state} \\ R_{j+1} + \gamma q_{\mathbf{w}^-}(S_{j+1}, A') - \lambda \log \pi_{\boldsymbol{\theta}}(A'|S_{j+1}), \quad A' \sim \pi_{\boldsymbol{\theta}}(\cdot|S_{j+1}) & \text{otherwise} \end{cases}$$

Perform a gradient descent step on $(y_j - q_{\mathbf{w}}(S_j, A_j))^2$ w.r.t. the critic parameters $\mathbf{w}$
Perform a gradient descent step on D_{KL} w.r.t. the actor parameters $\boldsymbol{\theta}$
Update the target network $\mathbf{w}^- \leftarrow \rho \mathbf{w}^- + (1-\rho)\mathbf{w}$
end loop

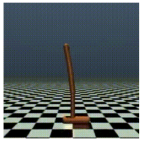
Mujoco Locomotion Tasks



Ant



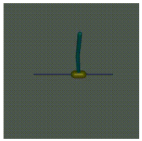
Half Cheetah



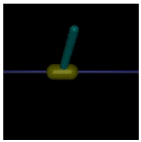
Hopper

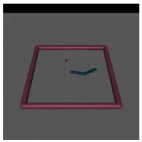
Humanoid

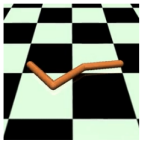
Humanoid Standup



Inverted Double Pendulum



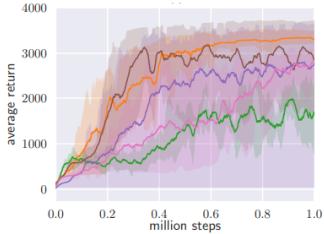




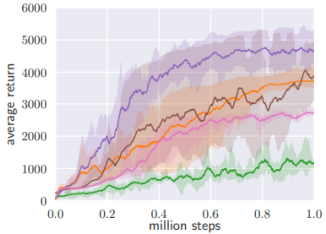
A common benchmark for continuous control using Deep RL

`https://gymnasium.farama.org/environments/mujoco/`

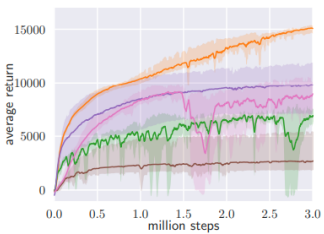
SAC - Experimental Results



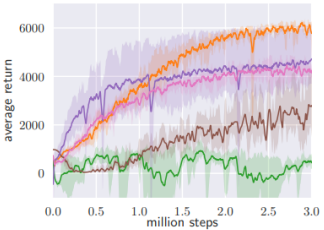
(a) Hopper-v1



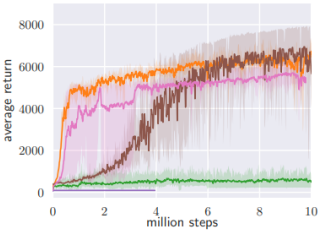
(b) Walker2d-v1



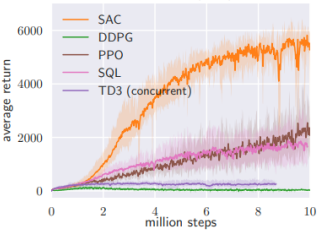
(c) HalfCheetah-v1



(d) Ant-v1



(e) Humanoid-v1



(f) Humanoid (rllab)

Pictures from (Haarnoja et al., 2018)

Outline

Outline

Importance Sampling

- Data $X \sim q$
- Want to estimate $\mu = \mathbb{E}_{x \sim p}[f(x)]$ with $p \neq q$
- **Importance-weighted** estimate

$$\widehat{\mu} = \frac{p(X)}{q(X)} f(X)$$

- **Unbiased**

$$\mathbb{E}[\widehat{\mu}] = \sum_x q(x) \frac{p(x)}{q(x)} f(x) = \mathbb{E}_{x \sim p}[f(x)] = \mu$$

- **Variance** can be large if p and q are very different!

Sample Reuse

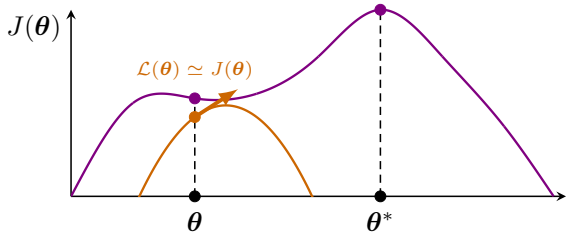
- We want to use samples collected with some policy to evaluate other candidate policies
- We could use **importance weighting** but **variance** might explode
- Use a **surrogate loss** function $\mathcal{L}$ that might be **biased**, but with smaller variance

True loss

$J(\boldsymbol{\theta})$

Surrogate objective

$\mathcal{L}(\boldsymbol{\theta})$



- A surrogate loss is good if

$$\arg\max_{\boldsymbol{\theta} \in \mathbb{R}^d} J(\boldsymbol{\theta}) \approx \arg\max_{\boldsymbol{\theta} \in \mathbb{R}^d} \mathcal{L}(\boldsymbol{\theta})$$

Performance Improvement

Recall once again the Performance Difference Lemma:

Theorem (Performance Difference Lemma, Kakade and Langford, 2002; Howard, 1960)

Given two policies π and π':

$$J(\pi') - J(\pi) = \mathbb{E}_{\substack{S \sim d_{\pi'} \\ A \sim \pi'(\cdot|S)}} [\mathbb{A}^{\pi}(S, A)]$$

Surrogate Loss Construction

- We would like to **maximize** $J(\pi')$ w.r.t. π'
- This is equivalent to maximize the **performance improvement** as π is fixed

$$J(\pi') - J(\pi) = \mathbb{E}_{\substack{S\sim d_{\pi'} \\ A\sim\pi'(\cdot|S)}} [\mathbb{A}^{\pi}(S, A)]$$

- Suppose we have samples from policy π, the expectations are w.r.t. to the new policy π'
- We could use **importance weighting** as

$$J(\pi') - J(\pi) = \mathbb{E}_{\substack{S\sim d_{\pi} \\ A\sim\pi(\cdot|S)}} \left[\frac{d_{\pi'}(S)\pi'(A|S)}{d_{\pi}(S)\pi(A|S)} \mathbb{A}^{\pi}(S, A) \right]$$

- But we don't know $d_{\pi'}$ nor d_{π}. Also, we might introduce a lot of variance in the estimators.

Surrogate Loss Construction

- Instead, we derive a **lower bound** to the performance improvement

$$J(\pi') - J(\pi) = \underbrace{\mathbb{E}_{\substack{S\sim d_\pi \\ A\sim\pi'(\cdot|S)}} \left[\mathbb{A}^\pi(S,A)\right]}_{\text{(A)}} - \underbrace{\sum_s \left(d_\pi(s) - d_{\pi'}(s)\right) \mathbb{E}_{A\sim\pi'(\cdot|s)} \left[\mathbb{A}^\pi(s,A)\right]}_{\text{(B)}}$$

- For (A), we use **importance weighting** but on the policy only

$$\text{(A)} = \mathbb{E}_{\substack{S\sim d_\pi \\ A\sim\pi(\cdot|S)}} \left[\frac{\pi'(A|S)}{\pi(A|S)} \mathbb{A}^\pi(S,A)\right]$$

- For (B), we bound with the **total variation distance** of the policies (Pirotta et al., 2013)

$$\text{(B)} \leq \frac{2\gamma\epsilon}{(1-\gamma)^2} \mathbb{E}_{S\sim d_\pi} \left[D_{\text{TV}}\left(\pi'(\cdot|S), \pi(\cdot|S)\right)\right]$$

where

$$\epsilon \coloneqq \sup_{s\in\mathcal{S}} \left|\mathbb{E}_{A\sim\pi'(\cdot|s)} \left[\mathbb{A}^\pi(s,A)\right]\right| \qquad D_{\text{TV}}\left(\pi'(\cdot|s), \pi(\cdot|s)\right) \coloneqq \frac{1}{2} \sum_a \left|\pi'(\cdot|s) - \pi(\cdot|s)\right|$$

Surrogate Loss

- The resulting **surrogate loss** is given by

$$\mathcal{L}(\pi') \coloneqq \mathbb{E}_{\substack{S\sim d_\pi \\ A\sim\pi(\cdot|S)}} \left[\frac{\pi'(A|S)}{\pi(A|S)}\mathbb{A}^\pi(S,A)\right] - \frac{2\gamma\epsilon}{(1-\gamma)^2}\mathbb{E}_{S\sim d_\pi}\left[D_{\mathsf{TV}}\left(\pi'(\cdot|S), \pi(\cdot|S)\right)\right]$$

- For every π', we have $J(\pi) \geq \mathcal{L}(\pi)$
- If $\pi = \pi'$ then $\mathcal{L}(\pi) = J(\pi)$
- If policy π' is close to π, $\mathcal{L}$ is a good surrogate for J
- **Monotonic Performance Improvement** (Pirotta et al., 2013)

$$\text{If} \qquad \pi^+ \in \arg\max_{\pi'} \mathcal{L}(\pi') \qquad \text{then} \qquad J(\pi^+) \geq J(\pi)$$

- Several approaches proposed based on this surrogate objective (Kakade and Langford, 2002; Wagner, 2011; Pirotta et al., 2013; Schulman et al., 2015; Achiam et al., 2017)

Outline

Towards TRPO

- The direct optimization of $\mathcal{L}(\pi')$ is not trivial
- The **total variation divergence** D_{TV} is hard to optimize (non-differentiable)
  - We bound it with the **KL-divergence** via Pinsker's inequality

$$D_{\text{TV}}\left(\pi'(\cdot|s), \pi(\cdot|s)\right) \leq \sqrt{\frac{1}{2} D_{\text{KL}}\left(\pi'(\cdot|s), \pi(\cdot|s)\right)}$$

- The optimization of the penalized objective is too **conservative**
  - We replace the penalization with a **constraint**

$$\max_{\pi'} \mathcal{L}^{\text{TRPO}}(\pi') := \mathbb{E}_{\substack{S\sim d_\pi \\ A\sim\pi(\cdot|S)}} \left[\frac{\pi'(A|S)}{\pi(A|S)} \mathbb{A}^{\pi}(S, A)\right]$$
$$\text{s.t. } \mathbb{E}_{S\sim d_\pi}\left[D_{\text{KL}}\left(\pi'(\cdot|S), \pi(\cdot|S)\right)\right] \leq \delta$$

where δ is an additional hyperparameter

Trust Region Policy Optimization

- Policies are parametric $\pi_{\boldsymbol{\theta}}(a|s)$ (Schulman et al., 2015)
- The advantage function $\mathbb{A}^{\pi}(s, a)$ is estimated using a **critic** $v_{\mathbf{w}}(s)$
  - Usually with **generalized advantage estimation** (Schulman et al., 2016)
- The **constrained** optimization problem is solved approximately using (conjugate) natural gradient

TRPO Algorithm

Initialize the actor parameters $\boldsymbol{\theta}$ and the critic parameters $\mathbf{w}$

loop for every iteration

Collect a rollout of T total timesteps with $\pi_{\boldsymbol{\theta}}$

Compute value targets y_t and advantage estimates $\widehat{\mathbb{A}}_t$ (e.g. with GAE) based on the current critic $v_{\mathbf{w}}$

Estimate the loss gradient as

$$\mathbf{g} = \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_t|S_t)\widehat{\mathbb{A}}_t$$

Use conjugate gradient to estimate natural gradient $\mathbf{x} = \mathbf{F}(\boldsymbol{\theta})^{-1}\mathbf{g}$

Update the actor parameters $\boldsymbol{\theta}$ via line search

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\sqrt{\frac{2\delta}{\mathbf{x}^{\mathrm{T}}\mathbf{F}(\boldsymbol{\theta})\mathbf{x}}}\mathbf{x}$$

finding the largest α that improves the sample loss and satisfies the (sample) KL-divergence constraint

Define the critic loss

$$\mathcal{L}_{\text{critic}}(\mathbf{w}) = \frac{1}{2}\sum_{t=0}^{T-1}\left(v_{\mathbf{w}}(S_t) - y_t\right)^2$$

Minimize $\mathcal{L}_{\text{critic}}(\mathbf{w})$ w.r.t. $\mathbf{w}$ via gradient descent

end loop

Multi-Epoch TRPO

- TRPO (and the other trust-region methods we will see) are only on-policy if the dataset collected with policy $\pi_{\boldsymbol{\theta}}$ is only used for a single policy update (epoch)
- Multi-Epoch implementations of these algorithms are common
- Multi-Epoch algorithms are off-policy for all but the first epoch of each iteration

Multi-Epoch TRPO

Initialize the actor parameters $\boldsymbol{\theta}$ and the critic parameters $\mathbf{w}$

loop for every iteration

Collect a rollout of T total timesteps with $\pi_{\boldsymbol{\theta}}$

Compute value targets y_t and advantage estimates $\widehat{\mathbb{A}}_t$ (e.g. with GAE) based on the current critic $v_{\mathbf{w}}$

loop for every epoch

Estimate the loss gradient as

$$\mathbf{g} = \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_t|S_t)\widehat{\mathbb{A}}_t$$

Use conjugate gradient to estimate natural gradient $\mathbf{x} = \mathbf{F}(\boldsymbol{\theta})^{-1}\mathbf{g}$

Update the actor parameters $\boldsymbol{\theta}$ via line search

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\sqrt{\frac{2\delta}{\mathbf{x}^{\mathrm{T}}\mathbf{F}(\boldsymbol{\theta})\mathbf{x}}}\mathbf{x}$$

finding the largest α that improves the sample loss and satisfies the (sample) KL-divergence constraint

Define the critic loss

$$\mathcal{L}_{\text{critic}}(\mathbf{w}) = \frac{1}{2}\sum_{t=0}^{T-1}\left(v_{\mathbf{w}}(S_t) - y_t\right)^2$$

Minimize $\mathcal{L}_{\text{critic}}(\mathbf{w})$ w.r.t. $\mathbf{w}$ via gradient descent

end loop

end loop

Outline

From TRPO to PPO

- More **practical** version of TRPO by Schulman et al. (2017)
- Avoid computing the **natural gradient**
- Approximation of the **KL-divergence constraint**
- **First Possibility**: replace the constraint with an **adaptive penalization**

$$\boldsymbol{\theta}_{k+1} \in \arg\max_{\boldsymbol{\theta}\in\mathbb{R}^d} \mathbb{E}_{\substack{S\sim d_{\pi_{\boldsymbol{\theta}}} \\ A\sim\pi_{\boldsymbol{\theta}}(\cdot|S)}} \left[\frac{\pi_{\boldsymbol{\theta}'}(A|S)}{\pi_{\boldsymbol{\theta}}(A|S)}\mathbb{A}^{\pi}(S,A)\right] - \lambda_k \mathbb{E}_{S\sim d_{\pi_{\boldsymbol{\theta}}}}\left[D_{\mathsf{KL}}\left(\pi_{\boldsymbol{\theta}'}(\cdot|S), \pi_{\boldsymbol{\theta}}(\cdot|S)\right)\right]$$

  - Heuristic adaptive λ_k

From TRPO to PPO

- **Second Possibility**: **clipped** objective function
  - Original TRPO **surrogate loss** function

$$\mathcal{L}^{\text{TRPO}}(\boldsymbol{\theta}') = \mathbb{E}_{\substack{S\sim d_{\pi_{\boldsymbol{\theta}}} \\ A\sim\pi_{\boldsymbol{\theta}}(\cdot|S)}}\left[\frac{\pi_{\boldsymbol{\theta}'}(A|S)}{\pi_{\boldsymbol{\theta}}(A|S)}\mathbb{A}^{\pi}(S,A)\right] = \mathbb{E}_{\substack{S\sim d_{\pi_{\boldsymbol{\theta}}} \\ A\sim\pi_{\boldsymbol{\theta}}(\cdot|S)}}\left[\omega_{\boldsymbol{\theta}'/\boldsymbol{\theta}}(S,A)\mathbb{A}^{\pi}(S,A)\right]$$

  - Construct a lower bound by clipping

$$\mathcal{L}^{\text{PPO}}(\boldsymbol{\theta}') \coloneqq \mathbb{E}_{\substack{S\sim d_{\pi_{\boldsymbol{\theta}}} \\ A\sim\pi_{\boldsymbol{\theta}}(\cdot|S)}}\left[\min\left\{\omega_{\boldsymbol{\theta}'/\boldsymbol{\theta}}(S,A)\mathbb{A}^{\pi}(S,A), \text{clip}(\omega_{\boldsymbol{\theta}'/\boldsymbol{\theta}}(S,A), 1-\epsilon, 1+\epsilon)\mathbb{A}^{\pi}(S,A)\right\}\right]$$

- Clipping empirically prevents policy from moving too much away from $\boldsymbol{\theta}$, mitigating both bias (due to $d^{\pi_{\boldsymbol{\theta}}} \neq d^{\pi_{\boldsymbol{\theta}'}}$) and variance (due to policy importance weights)

PPO Algorithm with Clipping

Initialize the actor parameters $\boldsymbol{\theta}$ and the critic parameters $\mathbf{w}$
loop for every iteration
Collect a rollout of T total timesteps with $\pi_{\boldsymbol{\theta}}$
Compute value targets $y_t$and advantage estimates $\widehat{\mathbb{A}}_t$ (e.g. with GAE) based on the current critic $v_{\mathbf{w}}$
Define the

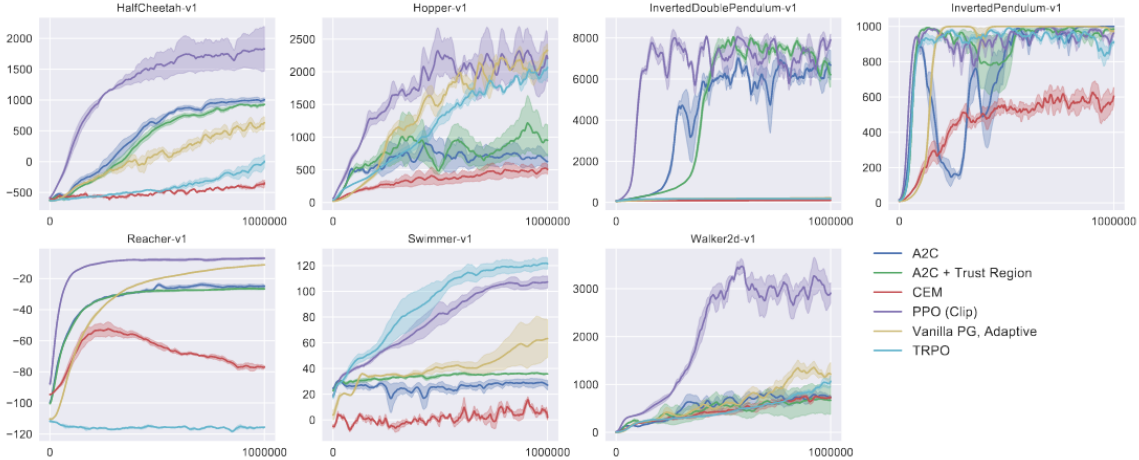
$$\mathcal{L}_{\text{actor}}(\boldsymbol{\theta}) = -\sum_{t=0}^{T-1} \min\left\{\omega_{\boldsymbol{\theta}'/\boldsymbol{\theta}}(S_t^i, A_t^i)\widehat{\mathbb{A}}_t, \text{clip}(\omega_{\boldsymbol{\theta}'/\boldsymbol{\theta}}(S_t^i, A_t^i), 1-\epsilon, 1+\epsilon)\widehat{\mathbb{A}}_t\right\}$$

Define the critic loss

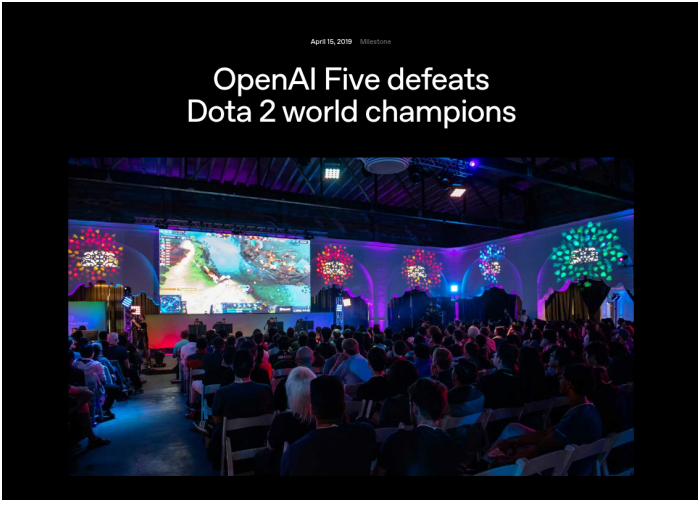
$$\mathcal{L}_{\text{critic}}(\mathbf{w}) = \frac{1}{2}\sum_{t=0}^{T-1}\left(v_{\mathbf{w}}(S_t) - y_t\right)^2$$

Minimize $\mathcal{L}_{\text{actor}}(\boldsymbol{\theta}) + \mathcal{L}_{\text{critic}}(\mathbf{w})$ w.r.t. $\boldsymbol{\theta}$ and $\mathbf{w}$ via gradient descent (one or more epochs)
end loop

PPO - Experimental Results

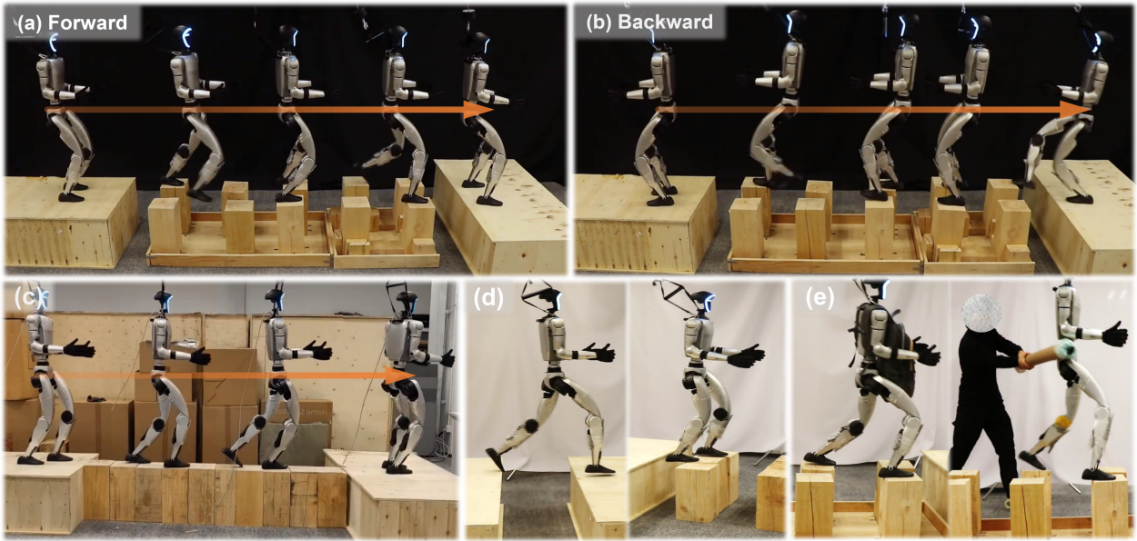


Pictures from (Schulman et al., 2017)



(Berner et al., 2019)

(Wurman et al., 2022)



(Wang et al., 2025)

RLHF
(Reinforcement Learning from Human Feedback)

(Ouyang et al., 2022)

`https://openai.com/index/chatgpt/`

Group Relative Policy Optimization (GRPO)

- **GRPO** is a trust-region method introduced by Shao et al. (2024) for fine-tuning LLMs to solve math problems
- In question answering rewards are sparse (correct/incorrect) and concentrated at the end of the episode (answer) $\implies$ Monte Carlo policy evaluation
- It keeps the clipping of PPO but drops the critic entirely: it's an **actor-only** policy gradient
- A "group" is a batch of trajectories (question-answer pairs)
- **Group-Relative Advantage Estimation** (mean and std over the batch for a fixed timestep t)

$$\overline{R}_t = \frac{R_t - \text{mean}(R_t^i)}{\text{std}(R_t^i)}$$

$$\widehat{\mathbb{A}}_t = \sum_{k \geq t} \overline{R}_{k+1}$$

- Very similar to modern implementations of REINFORCE (Ahmadian et al., 2024)

Algorithm	Class	On-policy	Continuous states	Finite actions	Continuous actions	Deterministic Policy	Reference
DQN	value-based	✗	✓	✓	✗	✓	(Mnih et al., 2015)
REINFORCE	actor-only	✓	✓	✓	✓	✗	(Williams, 1992)
A2C	actor-critic	✓	✓	✓	✓	✗	(Mnih et al., 2016)
DDPG	actor-critic	✗	✓	✗	✓	✓	(Lillicrap et al., 2015)
TD3	actor-critic	✗	✓	✗	✓	✓	(Fujimoto et al., 2018)
SAC	actor-critic	✗	✓	✓†	✓	✗	(Haarnoja et al., 2018)
TRPO	trust-region actor-critic	✓*	✓	✓	✓	✗	(Schulman et al., 2015)
PPO	trust-region actor-critic	✓*	✓	✓	✓	✗	(Schulman et al., 2017)
GRPO	trust-region actor-only	✓*	✓	✓	✓	✗	(Shao et al., 2024)

† See (Christodoulou, 2019)
* Single-epoch version only

Master theses on Reinforcement Learning

- Exploration in policy gradient algorithms
- Multi-agent RL
- Theory of RL with function approximation
- Representation learning and self-supervised RL
- Symbolic RL (reward specification, reasoning...)
- ...

Contact: matteo.papini@unimi.it

https://sites.google.com/view/lailaunimi/

References I

J. Achiam, D. Held, A. Tamar, and P. Abbeel. Constrained policy optimization. *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 70:22–31, 2017. URL `https://arxiv.org/abs/1705.10528`.

A. Ahmadian, C. Cremer, M. Gallé, M. Fadaee, J. Kreutzer, O. Pietquin, A. Üstün, and S. Hooker. Back to basics: Revisiting reinforce-style optimization for learning from human feedback in llms. In *ACL (1)*, pages 12248–12267. Association for Computational Linguistics, 2024.

A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Trans. Syst. Man Cybern.*, 13(5):834–846, 1983.

C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, R. Józefowicz, S. Gray, C. Olsson, J. Pachocki, M. Petrov, H. P. de Oliveira Pinto, J. Raiman, T. Salimans, J. Schlatter, J. Schneider, S. Sidor, I. Sutskever, J. Tang, F. Wolski, and S. Zhang. Dota 2 with large scale deep reinforcement learning. *CoRR*, abs/1912.06680, 2019.

P. Christodoulou. Soft actor-critic for discrete action settings. *CoRR*, abs/1910.07207, 2019.

S. Fujimoto, H. van Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 1582–1591. PMLR, 2018.

T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80, pages 1861–1870, 2018. URL `https://arxiv.org/abs/1801.01290`.

R. A. Howard. Dynamic programming and markov processes. 1960.

S. M. Kakade and J. Langford. Approximately optimal approximate reinforcement learning. *Proceedings of the 19th International Conference on Machine Learning (ICML)*, pages 267–274, 2002.

T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015. URL `https://arxiv.org/abs/1509.02971`.

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

References II

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nat.*, 518(7540):529–533, 2015.

V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *ICML*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1928–1937. JMLR.org, 2016.

O. Nachum, R. Gu, X. Xie, and S. Levine. Bridging the gap between value and policy based reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70, pages 2812–2821, 2017. URL `https://arxiv.org/abs/1702.02282`.

L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. F. Christiano, J. Leike, and R. Lowe. Training language models to follow instructions with human feedback. In *NeurIPS*, 2022.

M. Pirotta, M. Restelli, A. Pecorino, and D. Bascetta. Safe policy iteration in approximate reinforcement learning. *Journal of Machine Learning Research*, 14:2605–2649, 2013.

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

J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. In *Proceedings of the 4th International Conference on Learning Representations (ICLR)*, 2016. URL `https://arxiv.org/abs/1506.02438`.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL `https://arxiv.org/abs/1707.06347`.

Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, M. Zhang, Y. K. Li, Y. Wu, and D. Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300, 2024.

References III

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

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. P. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of go with deep neural networks and tree search. *Nat.*, 529(7587):484–489, 2016.

R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In S. A. Solla, T. K. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 1057–1063. MIT Press, 1999.

P. Wagner. A reinterpretation of the policy oscillation phenomenon in approximate policy iteration. *Advances in neural information processing systems*, 24, 2011.

H. Wang, Z. Wang, J. Ren, Q. Ben, T. Huang, W. Zhang, and J. Pang. Beamdojo: Learning agile humanoid locomotion on sparse footholds. *CoRR*, abs/2502.10363, 2025.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992.

P. R. Wurman, S. Barrett, K. Kawamoto, J. MacGlashan, K. Subramanian, T. J. Walsh, R. Capobianco, A. Devlic, F. Eckert, F. Fuchs, L. Gilpin, P. Khandelwal, V. R. Kompella, H. Lin, P. MacAlpine, D. Oller, T. Seno, C. Sherstan, M. D. Thomure, H. Aghabozorgi, L. Barrett, R. Douglas, D. Whitehead, P. Dürr, P. Stone, M. Spranger, and H. Kitano. Outracing champion gran turismo drivers with deep reinforcement learning. *Nat.*, 602(7896):223–228, 2022.