Multi-Agent RL, RL from Human Feedback

Milan Straka

May 21, 2025

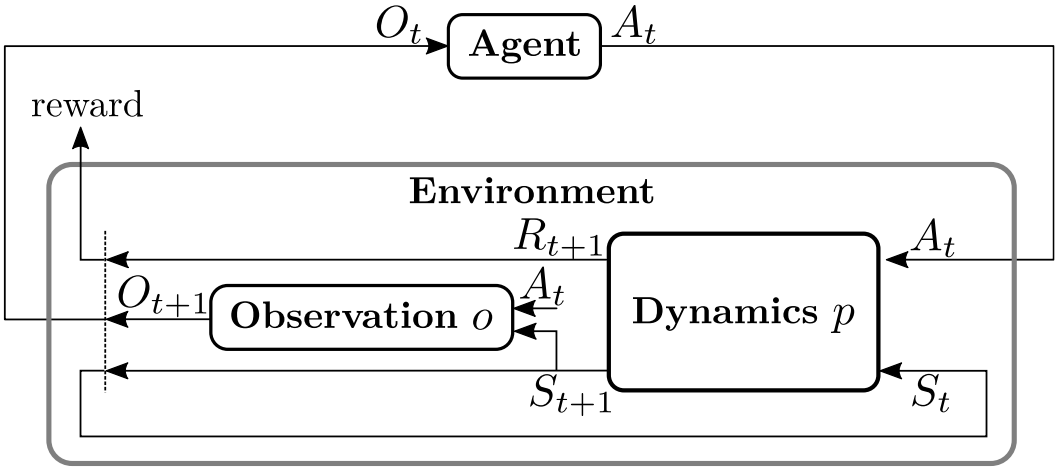
Charles University
LANGTECH
EUROPEAN UNION
European Structural and Investment Fund
Operational Programme Research, Development and Education

Charles University in Prague
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics

Multi-Agent Reinforcement Learning

Partially Observable MDPs

Recall that a **partially observable Markov decision process** extends the Markov decision process to a sextuple $(\mathcal{S}, \mathcal{A}, p, \gamma, \mathcal{O}, o)$, where the MDP components



- $\mathcal{S}$ is a set of states,
- $\mathcal{A}$ is a set of actions,
- $p(S_{t+1} = s', R_{t+1} = r | S_t = s, A_t = a)$ is a probability that action $a \in \mathcal{A}$ will lead from state $s \in \mathcal{S}$ to $s' \in \mathcal{S}$, producing a **reward** $r \in \mathbb{R}$,
- $\gamma \in [0, 1]$ is a **discount factor**,

are extended by:

- $\mathcal{O}$ is a set of observations,
- $o(O_{t+1} | S_{t+1}, A_t)$ is an observation model, where observation O_t is used as agent input instead of the state S_t.

Partially Observable Stochastic Game

A **partially observable stochastic game (POSG)** is a 9-tuple $(\mathcal{S}, N, \{\mathcal{A}^{i\in[N]}\}, \{\Omega^{i\in[N]}\}, \{R^{i\in[N]}\}, P, \{O^{i\in[N]}\}, \rho_0, \gamma)$, where

- $\mathcal{S}$ is the set of all possible *states*,
- N is the *number of agents*,
- $\mathcal{A}^i$ is the set of all possible *actions* for agent i, with $\mathcal{A}^{\Pi} \stackrel{\text{def}}{=} \prod_i \mathcal{A}^i$,
- Ω^i is the set of all possible *observations* for agent i,
- $P(s_{t+1} \in \mathcal{S} | s_t \in \mathcal{S}, \boldsymbol{a}_t \in \mathcal{A}^{\Pi})$ is the *transition model*,
- $R^i : \mathcal{S} \times \mathcal{A}^{\Pi} \times \mathcal{S} \to \mathcal{R}$ is the *reward function* for agent i,
- $O^i(\omega^i_{t+1} \in \Omega^i | s_{t+1} \in \mathcal{S}, a^i_t \in \mathcal{A})$ is the *observation model* for agent i, a distribution of observing w^i_{t+1} after performing action a^i_t leading to state s_{t+1},
- ρ_0 is the *initial state distribution*,
- $\gamma \in [0, 1]$ is a *discount factor*.

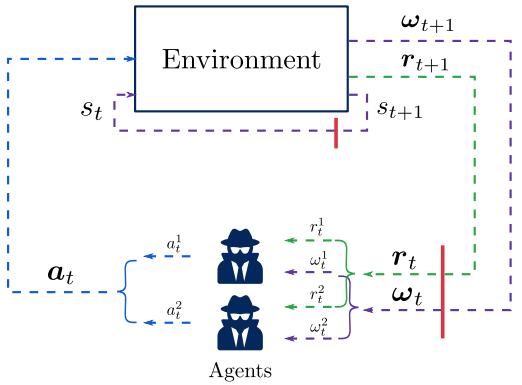


Figure 1.3: Partially Observable Stochastic Game (POSG)

Figure 1.3 of "Cooperative Multi-Agent Reinforcement Learning", https://dspace.cuni.cz/handle/20.500.11956/127431

Partially Observable Stochastic Game

We denote

- joint actions/policy/observation across all agents as vectors

$$\boldsymbol{a} \stackrel{\text{def}}{=} (a^1, \ldots, a^N) \in \mathcal{A}^{\Pi},$$

- joint actions/policy/observation for all agents but agent i as

$$\boldsymbol{a}^{-i} \stackrel{\text{def}}{=} (a^1, \ldots, a^{i-1}, a^{i+1}, \ldots, a^N),$$

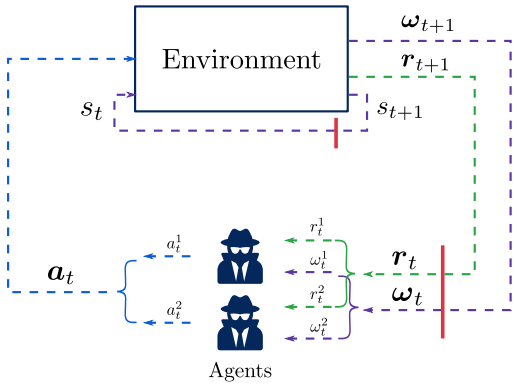


Figure 1.3: Partially Observable Stochastic Game (POSG)

Figure 1.3 of "Cooperative Multi-Agent Reinforcement Learning", https://dspace.cuni.cz/handle/20.500.11956/127431

Agent-Environment Cycle Game

However, when actually implementing POSG, various ambiguities exist in the order of execution. Therefore, **agent-environment cycle game (AECG)** has been proposed, a 12-tuple $(\mathcal{S}, N, \{\mathcal{A}^{i\in[N]}\}, \{\Omega^{i\in[N]}\}, \{R^{i\in[N]}\}, \{T^{i\in[N]}\}, P, \{O^{i\in[N]}\}, \nu, s_0, i_0, \gamma)$ where

- $\mathcal{S}$ is the set of all possible *states*,
- N is the *number of agents*, including 0 for "environment" agent; $[N^\cup] \stackrel{\text{def}}{=} [N] \cup \{0\}$,
- $\mathcal{A}^i$ is the set of all possible *actions* for agent i, with $\mathcal{A}^0 \stackrel{\text{def}}{=} \{\emptyset\}$, $\mathcal{A}^\cup \stackrel{\text{def}}{=} \bigcup_{i\in[N^\cup]} \mathcal{A}^i$,
- Ω^i is the set of all possible *observations* for agent i,
- $T^i : \mathcal{S} \times \mathcal{A}^i \to \mathcal{S}$ is the deterministic *transition function* for agent i,
- $P(s_{t+1} \in \mathcal{S} | s_t \in \mathcal{S})$ is the transition function for the environment,
- $R^i(r^i_{t+1} \in \mathbb{R} | s_t \in \mathcal{S}, j \in [N^\cup], a^j_t \in \mathcal{A}^j, s_{t+1} \in \mathcal{S})$ is the *reward distribution* for agent i,
- $O^i(\omega^i_{t+1} \in \Omega^i | s_{t+1} \in \mathcal{S})$ is the *observation model* for agent i,
- $\nu(j \in [N^\cup] | s_t \in \mathcal{S}, i \in [N^\cup], a^i_t \in \mathcal{A}^i)$ is the *next agent function*,
- $s_0 \in \mathcal{S}$ is the *initial state*,
- $i_0 \in [N^U]$ is the *initial agent*, • $\gamma \in [0, 1]$ is a *discount factor*.

Agent-Environment Cycle Game

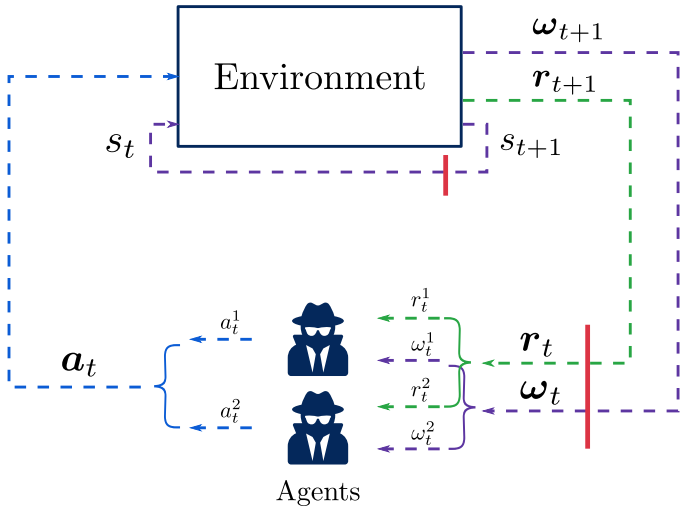


Figure 1.3: Partially Observable Stochastic Game (POSG)

Figure 1.3 of "Cooperative Multi-Agent Reinforcement Learning", https://dspace.cuni.cz/handle/20.500.11956/127431

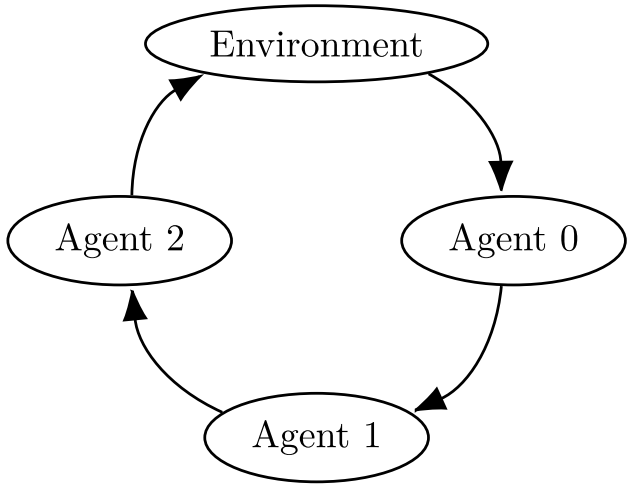


Figure 1.4: AEC diagram for MPE Simple Spread environment

Figure 1.4 of "Cooperative Multi-Agent Reinforcement Learning", https://dspace.cuni.cz/handle/20.500.11956/127431

It holds that for every POSG, there is an equivalent AECG, and vice versa.

- When starting with POSG, a single transition is represented as N+1 steps in AECG: first the N agents sequentially act, storing all their actions in the state, and then the environment performs the same transition as the POSG.
- Starting with AECG, we design POSG where the state is a triple *(AECG state, actor to act, reward to obtain)*, and every transition is one step from AECG.

Environment Settings

Depending on the reward function, there are several environment settings:

- **fully cooperative**, when $\forall i, \forall j : R^i(s_t, \boldsymbol{a}_t, s_{t+1}) = R^j(s_t, \boldsymbol{a}_t, s_{t+1})$;
- **zero-sum**, when $\sum_{i \in [N]} R^i(s_t, \boldsymbol{a}_t, s_{t+1}) = 0$.

Informally, we talk about **cooperative** and **competitive** settings, but we refrain from defining them exactly (the definitions that we had were incorrect).

The MARL Problem

We define a trajectory $\boldsymbol{\tau}$ as a sequence of states and actions

$$\boldsymbol{\tau} \stackrel{\text{def}}{=} (\boldsymbol{s}_0, \boldsymbol{a}_0, \boldsymbol{s}_1, \boldsymbol{a}_1, \boldsymbol{s}_2, \ldots),$$

where:

- $\boldsymbol{s}_0 \sim \rho_0$,
- $\boldsymbol{a}_t \sim \boldsymbol{\pi}(\cdot|\boldsymbol{s}_t)$,
- $\boldsymbol{s}_{t+1} \to P(\cdot|\boldsymbol{s}_t, \boldsymbol{a}_t)$.

A return for an agent i and trajectory $\boldsymbol{\tau}$ is

$$R^i(\boldsymbol{\tau}) \stackrel{\text{def}}{=} \sum_{t=0}^{|\boldsymbol{\tau}|} \gamma^t r^i_{t+1}.$$

The MARL Problem

For a given policy $\boldsymbol{\pi}$, the expected return for agent i is

$$J^i(\boldsymbol{\pi}) \stackrel{\text{def}}{=} \mathbb{E}_{\boldsymbol{\tau}\sim\boldsymbol{\pi}}\left[R^i(\boldsymbol{\tau})\right],$$

where a probability of a trajetory $\boldsymbol{\tau}$ is

$$P(\boldsymbol{\tau}|\boldsymbol{\pi}) \stackrel{\text{def}}{=} \rho_0(\boldsymbol{s}_0)\prod_{t=0}^{|\boldsymbol{\tau}|} P(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t,\boldsymbol{a}_t)\boldsymbol{\pi}(\boldsymbol{a}_t|\boldsymbol{s}_t).$$

For a given joing policy $\boldsymbol{\pi}^{-i}$, **best response** is

$$\hat{\pi}^i(\boldsymbol{\pi}^{-i}) \stackrel{\text{def}}{=} \arg\max_{\pi_i} J^i(\pi^i,\boldsymbol{\pi}^{-i}).$$

The MARL Goal

It is unfortunately not clear what the goal of MARL should be, given that it is a multi-criterion optimization problem.

One possibility is to seek for **Nash equilibrium**, which is a joint policy $\boldsymbol{\pi}_*$ fulfilling

$$\forall i \in [N], \forall \pi^i : J^i(\boldsymbol{\pi}_*) \geq J^i(\pi^i, \boldsymbol{\pi}_*^{-i}).$$

In other words, π_*^i is a best response to $\boldsymbol{\pi}_*^{-i}$ for all agents i.

A Nash equilibrium exists for any finite game (finite number of players, each with a finite number of strategies). Unfortunately, there can be multiple Nash equilibria with different payoffs (Nash equilibrium is just a "local" optimum).

- Stag hunt

A\B	Stag	Rabbit
Stag	2\2	0\1
Rabbit	1\0	1\1

- Prisoner's dilemma

A\B	Stay silent	Testify
Stay silent	-1\-1	-3\0
Testify	0\-3	-2\-2

MARL Training Schemes

Centralized Scheme

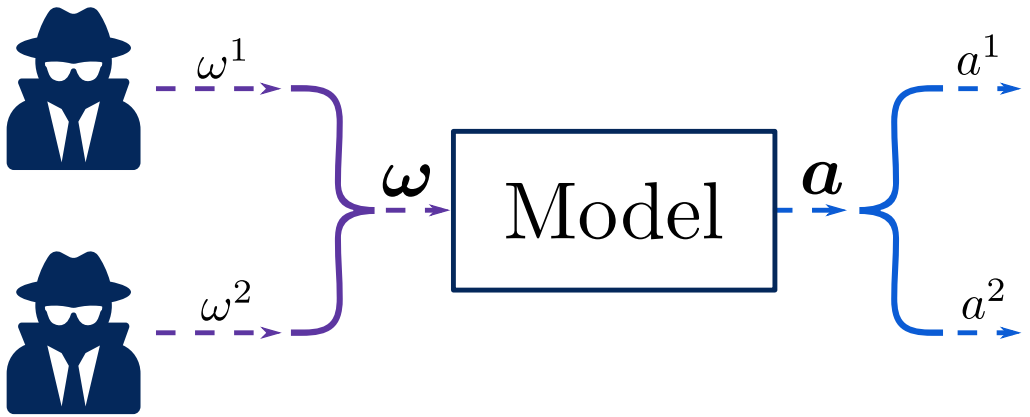


Figure 3.1: Centralized scheme

Figure 3.1 of "Cooperative Multi-Agent Reinforcement Learning", https://dspace.cuni.cz/handle/20.500.11956/127431

A joint model for all agents, a single critic.

Concurrent/Parameter-Sharing Scheme

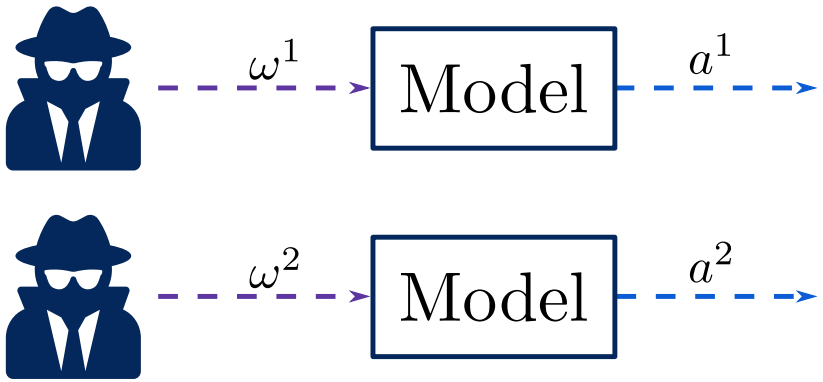


Figure 3.2: Concurrent scheme

Figure 3.2 of "Cooperative Multi-Agent Reinforcement Learning", https://dspace.cuni.cz/handle/20.500.11956/127431

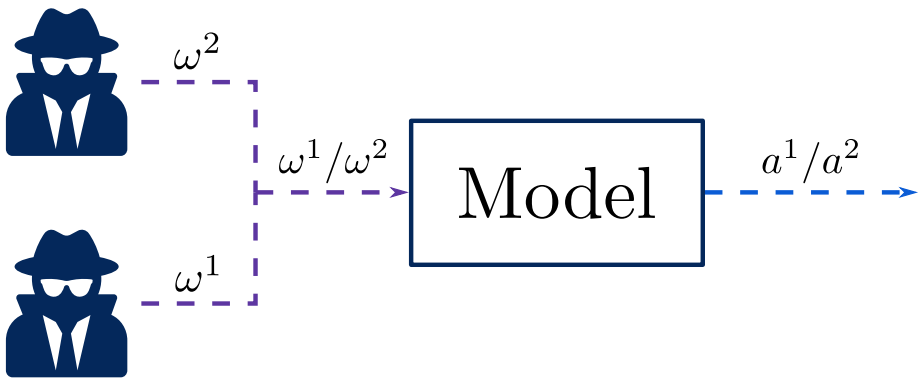


Figure 3.3: Parameter Sharing Scheme

Figure 3.3 of "Cooperative Multi-Agent Reinforcement Learning", https://dspace.cuni.cz/handle/20.500.11956/127431

Each agent is trained independently. When the agents are homogenous, their models can be optionally shared (the *parameter-sharing scheme*).

However, the environment is then non-stationary, and using a replay buffer is problematic because of changing policies of other agents.

Centralized Training with Decentralized Execution

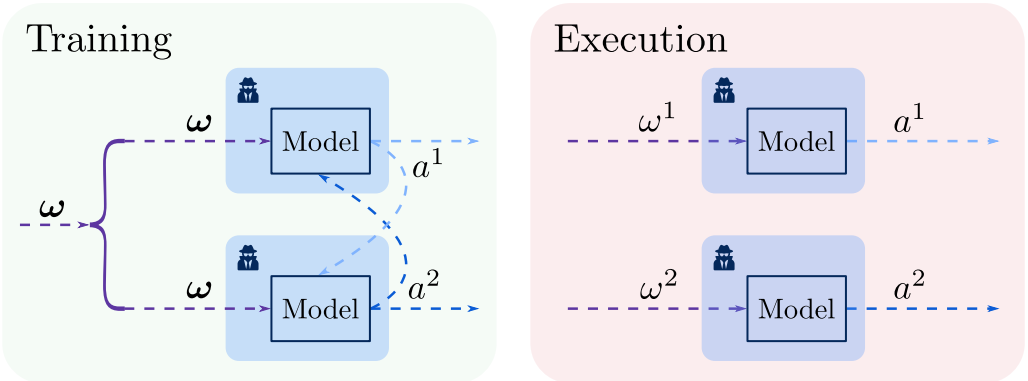


Figure 3.4: Centralized Training with Decentralized Execution (CT-DE)

Figure 3.4 of "Cooperative Multi-Agent Reinforcement Learning", https://dspace.cuni.cz/handle/20.500.11956/127431

Quite a common model, where the agents are independent, but the critics get the observations and actions of all agents.

MARL Algorithms

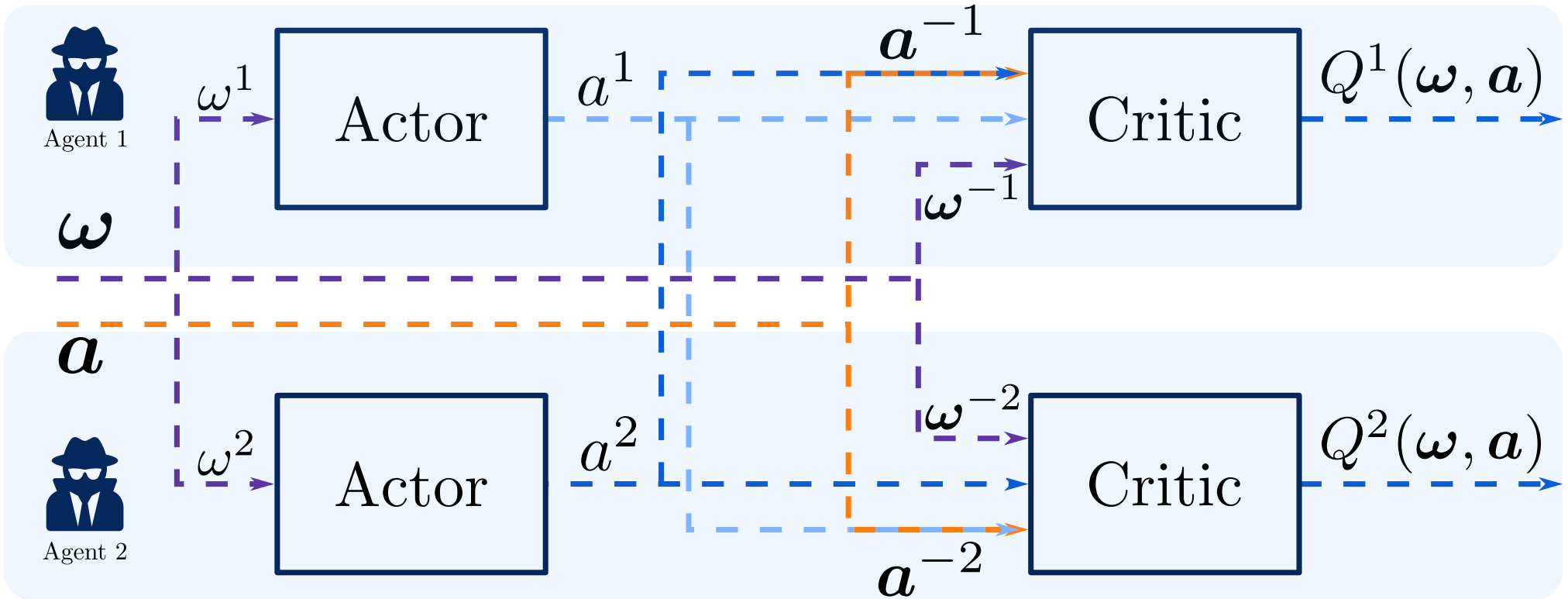


Figure 3.5: Multi-Agent Deep Deterministic Policy Gradient (MADDPG)

Figure 3.5 of "Cooperative Multi-Agent Reinforcement Learning", https://dspace.cuni.cz/handle/20.500.11956/127431

Multi-Agent Deep Deterministic Policy Gradient

Algorithm 3.1: Multi-Agent Deep Deterministic Policy Gradient

Input: initial policy parameters $\boldsymbol{\theta}$, Q-function parameters $\boldsymbol{\phi}$, empty replay buffer $\mathcal{D}$.

1 Set target parameters equal to main parameters $\boldsymbol{\theta}_{\text{targ}} \leftarrow \boldsymbol{\theta}$, $\boldsymbol{\phi}_{\text{targ}} \leftarrow \boldsymbol{\phi}$.

2 **repeat**

3 Observe joint-observation $\boldsymbol{\omega}$ and select joint-action $\boldsymbol{a} = \text{clip}(\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{\omega}) + \boldsymbol{\epsilon}, \boldsymbol{a}_{\text{low}}, \boldsymbol{a}_{\text{high}})$, where $\boldsymbol{\epsilon} \sim \mathcal{N}$.

4 Execute $\boldsymbol{a}$ in the environment.

5 Observe next observations $\boldsymbol{\omega}'$, rewards $\boldsymbol{r}$ and done signal for each agent $\boldsymbol{d}$.

6 Store $(\boldsymbol{\omega}, \boldsymbol{a}, \boldsymbol{r}, \boldsymbol{\omega}', \boldsymbol{d})$ in replay buffer $\mathcal{D}$.

7 **if** all($\boldsymbol{d}$) is true **then**

8 Reset environment state.

9 **end if**

10 Randomly sample a batch of transitions, $B = \{(\boldsymbol{\omega}, \boldsymbol{a}, \boldsymbol{r}, \boldsymbol{\omega}', \boldsymbol{d})\}$ from $\mathcal{D}$.

11 **for** agent i in $[N]$ **do**

12 Compute targets

$$y(r^i, \boldsymbol{\omega}', d^i) = r^i + \gamma(1 - d^i) Q^i_{\phi_{\text{targ}}}(\boldsymbol{\omega}', \boldsymbol{\mu}_{\boldsymbol{\theta}_{\text{targ}}}(\boldsymbol{\omega}'))).$$

13 Update Q-function by one step of gradient descent using

$$\nabla^i_{\phi} \frac{1}{|B|} \sum_{(\boldsymbol{\omega}, \boldsymbol{a}, \boldsymbol{r}, \boldsymbol{\omega}', \boldsymbol{d}) \in B} \left(Q^i_{\phi}(\boldsymbol{\omega}, \boldsymbol{a}) - y(r^i, \boldsymbol{\omega}', d^i)\right)^2.$$

14 Update policy by one step of gradient ascent using

$$\nabla^i_{\theta} \frac{1}{|B|} \sum_{(\boldsymbol{\omega}, \boldsymbol{a}) \in B} Q^i_{\phi}(\boldsymbol{\omega}, \boldsymbol{a}^{-i}, \mu^i_{\theta}(\omega^i)).$$

15 Update target networks with

$$\phi^i_{\text{targ}} \leftarrow \alpha \phi^i_{\text{targ}} + (1 - \alpha)\phi^i$$
$$\theta^i_{\text{targ}} \leftarrow \alpha \theta^i_{\text{targ}} + (1 - \alpha)\theta^i.$$

16 **end for**

17 **until** convergence

Algorithm 3.1 of "Cooperative Multi-Agent Reinforcement Learning", https://dspace.cuni.cz/handle/20.500.11956/127431

Alternatively, in multi-agent settings, in some experiments it was beneficial to estimate the gradient for the policy update using the current policy instead of the action from the replay buffer; if the line 14 is changed to

$$\nabla^i_{\boldsymbol{\theta}} \frac{1}{|B|} \sum_{\boldsymbol{\omega}} Q^i_{\varphi}\left(\boldsymbol{\omega}, \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{\omega})\right),$$

we talk about *Soft MADDPG*.

Multi-Agent Twin Delayed DDPG

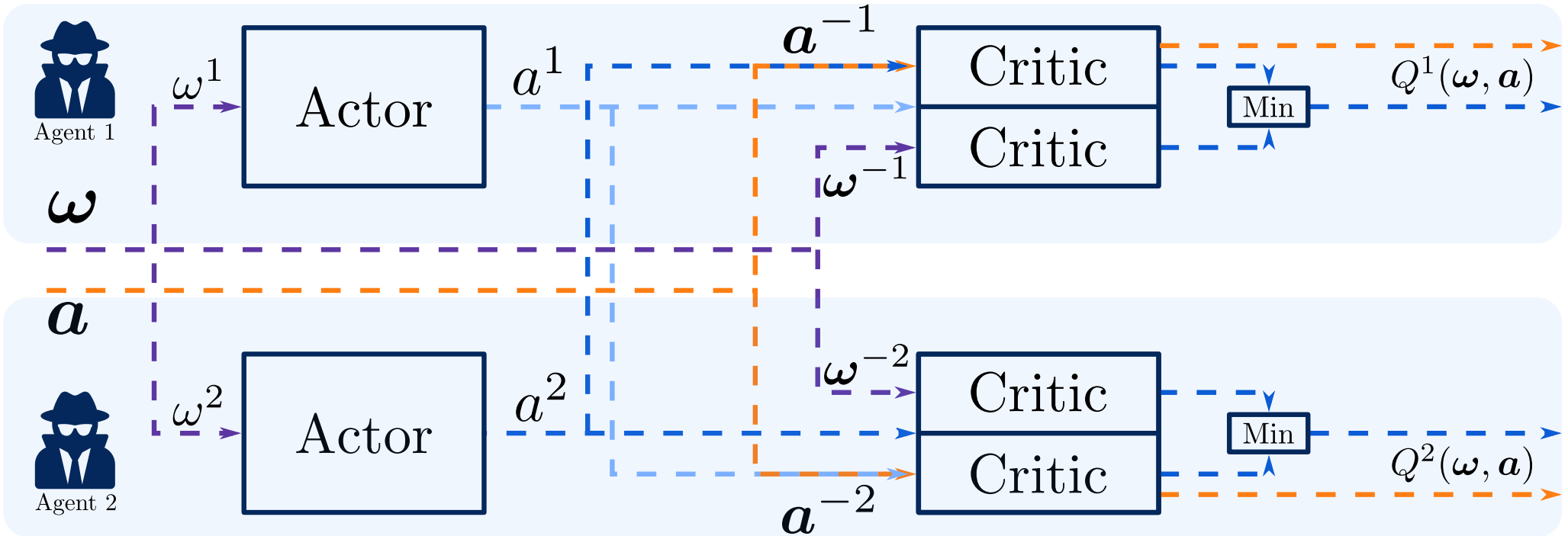


Figure 3.6: Multi-Agent Twin Delayed DDPG (MATD3)

Figure 3.6 of "Cooperative Multi-Agent Reinforcement Learning", https://dspace.cuni.cz/handle/20.500.11956/127431

Multi-Agent Twin Delayed DDPG

Algorithm 3.2: Multi-Agent Twin Delayed DDPG

Input: initial policy parameters $\boldsymbol{\theta}$, Q-function parameters $\boldsymbol{\phi}_1, \boldsymbol{\phi}_2$, empty replay buffer $\mathcal{D}$.

Set target parameters equal to main parameters $\boldsymbol{\theta}_{\text{targ}} \leftarrow \boldsymbol{\theta}$, $\boldsymbol{\phi}_{\text{targ},1} \leftarrow \boldsymbol{\phi}_1$, $\boldsymbol{\phi}_{\text{targ},2} \leftarrow \boldsymbol{\phi}_2$.

repeat

  Observe joint-observation $\boldsymbol{\omega}$ and select joint-action $\boldsymbol{a} = \text{clip}(\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{\omega}) + \boldsymbol{\epsilon}, \boldsymbol{a}_{\text{low}}, \boldsymbol{a}_{\text{high}})$, where $\boldsymbol{\epsilon} \sim \mathcal{N}$.

  Execute $\boldsymbol{a}$ in the environment.

  Observe next observations $\boldsymbol{\omega}'$, rewards $\boldsymbol{r}$ and done signal for each agent $\boldsymbol{d}$.

  Store $(\boldsymbol{\omega}, \boldsymbol{a}, \boldsymbol{r}, \boldsymbol{\omega}', \boldsymbol{d})$ in replay buffer $\mathcal{D}$.

  if all($\boldsymbol{d}$) is true **then**

    Reset environment state.

  end if

  Randomly sample a batch of transitions, $B = \{(\boldsymbol{\omega}, \boldsymbol{a}, \boldsymbol{r}, \boldsymbol{\omega}', \boldsymbol{d})\}$ from $\mathcal{D}$.

  for agent i in $[N]$ **do**

    Compute target actions

$$\boldsymbol{a}' = \text{clip}(\boldsymbol{\mu}_{\boldsymbol{\theta}_{\text{targ}}}(\boldsymbol{\omega}') + \text{clip}(\boldsymbol{\epsilon}, -c, c), \boldsymbol{a}_{\text{low}}, \boldsymbol{a}_{\text{high}}), \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma \boldsymbol{I}).$$

    Compute targets

$$y(r^i, \boldsymbol{\omega}', d^i) = r^i + \gamma(1 - d^i) \min_{j \in \{1,2\}} Q^i_{\phi_{\text{targ},j}}(\boldsymbol{\omega}', \boldsymbol{a}').$$

    Update Q-function by one step of gradient descent using

$$\forall j \in \{1, 2\}: \qquad \nabla^i_{\phi_j} \frac{1}{|B|} \sum_{(\boldsymbol{\omega}, \boldsymbol{a}, \boldsymbol{r}, \boldsymbol{\omega}', \boldsymbol{d}) \in B} \left(Q^i_{\phi_j}(\boldsymbol{\omega}, \boldsymbol{a}) - y(r^i, \boldsymbol{\omega}', d^i) \right)^2.$$

    if time to update policy function **then**

      Update policy by one step of gradient ascent using

$$\nabla^i_{\theta} \frac{1}{|B|} \sum_{(\boldsymbol{\omega}, \boldsymbol{a}) \in B} Q^i_{\phi_1}(\boldsymbol{\omega}, \boldsymbol{a}^{-i}, \mu^i_{\theta}(\omega^i)).$$

      Update target networks with

$$\phi^i_{\text{targ}} \leftarrow \alpha \phi^i_{\text{targ}} + (1 - \alpha)\phi^i$$
$$\theta^i_{\text{targ}} \leftarrow \alpha \theta^i_{\text{targ}} + (1 - \alpha)\theta^i.$$

    end if

  end for

until convergence

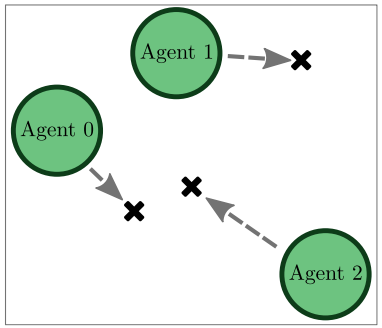
Algorithm 3.2 of "Cooperative Multi-Agent Reinforcement Learning", https://dspace.cuni.cz/handle/20.500.11956/127431

We can again consider a *Soft MATD3* variant.

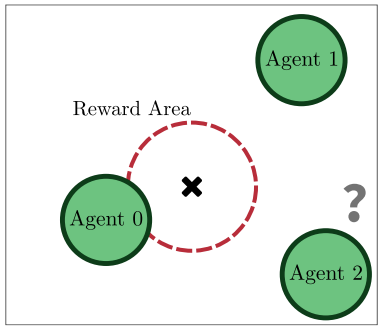
Furthermore, we can also use the minimum of both critics during policy update (shown to be beneficial by DDPG++ and SAC). The resulting algorithm is called *(Soft) MATD4*.

MARL Evaluation

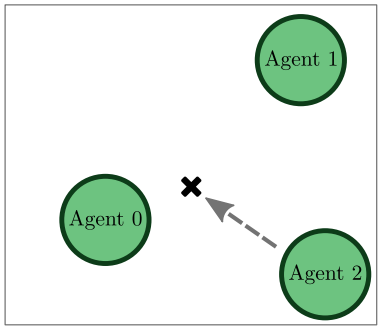
MARL Evaluation, Simple Target



(a) Cooperative navigation



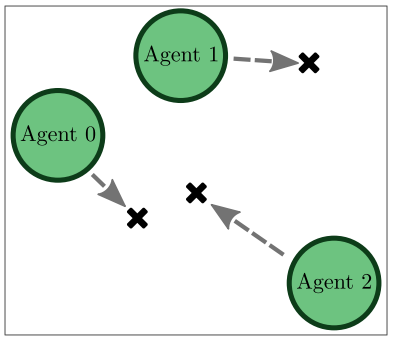
(b) MPE Simple Spread



(c) MPE Simple Target

Figure 6.1: Cooperative navigation

Figure 6.1 of "Cooperative Multi-Agent Reinforcement Learning", https://dspace.cuni.cz/handle/20.500.11956/127431



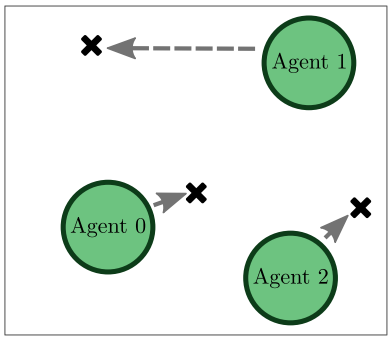


Figure 6.2: MPE Simple Collect

Figure 6.2 of "Cooperative Multi-Agent Reinforcement Learning", https://dspace.cuni.cz/handle/20.500.11956/127431

Reward is given for touching a landmark, and for unoccupied landmarks also for distance of the nearest agent (orignally any agent, but easier variant is an agent not occupying a landmark).

The agents have non-negligible size and get negative reward for colliding.

Actions can be discrete ($\varnothing$, $\leftarrow$, $\rightarrow$, $\uparrow$, $\downarrow$; ST Gumbel-softmax is used) or continuous.

In the *Simple Collect* variant, the targets disappear after being occupied for some time, and a new one appears on a random location.

MARL Evaluation, Simple Target, Continuous Actions

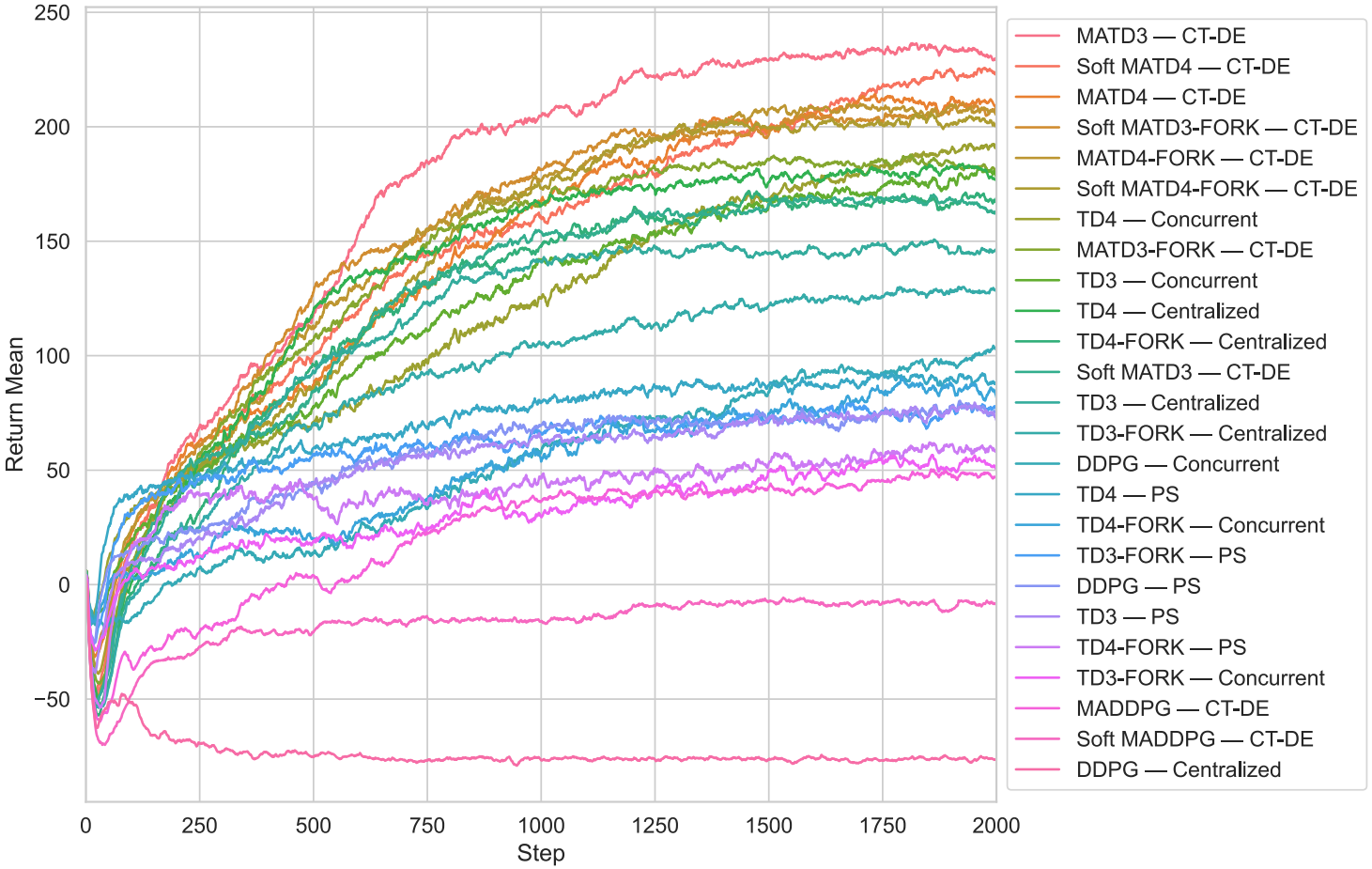


Figure 6.4: Training of MPE Simple Target Continuous

Figure 6.4 of "Cooperative Multi-Agent Reinforcement Learning", https://dspace.cuni.cz/handle/20.500.11956/127431

MARL Evaluation, Simple Target, Continuous Actions

Algorithm — Scheme	Step 1000	Step 1500	Step 2000
MATD3 — CT-DE	**204.55 ± 75.20**	**229.22 ± 61.24**	**229.13 ± 60.49**
Soft MATD4 — CT-DE	160.05 ± 91.58	198.95 ± 60.09	222.70 ± 63.45
MATD4 — CT-DE	167.36 ± 93.70	197.96 ± 83.78	209.31 ± 73.73
Soft MATD3-FORK — CT-DE	181.43 ± 120.70	196.62 ± 115.77	207.12 ± 120.15
MATD4-FORK — CT-DE	176.10 ± 103.08	207.62 ± 63.07	205.87 ± 54.62
Soft MATD4-FORK — CT-DE	173.04 ± 121.90	199.92 ± 115.03	200.56 ± 112.56
TD4 — Concurrent	124.86 ± 56.81	168.19 ± 57.67	191.26 ± 66.95
MATD3-FORK — CT-DE	165.90 ± 109.04	185.39 ± 121.30	181.14 ± 117.56
TD3 — Concurrent	141.68 ± 95.89	166.04 ± 94.70	178.08 ± 70.62
TD4 — Centralized	165.42 ± 99.01	175.77 ± 91.21	177.23 ± 94.04
TD4-FORK — Centralized	147.53 ± 158.98	168.67 ± 162.29	169.09 ± 161.63
Soft MATD3 — CT-DE	153.94 ± 90.68	166.21 ± 118.72	162.48 ± 111.84
TD3 — Centralized	140.86 ± 172.41	144.93 ± 173.38	146.84 ± 173.46
TD3-FORK — Centralized	105.11 ± 155.83	121.86 ± 128.41	127.91 ± 128.45
DDPG — Concurrent	57.89 ± 158.75	84.37 ± 156.04	102.91 ± 163.00
TD4 — PS	79.51 ± 31.69	86.90 ± 24.38	88.14 ± 25.81
TD4-FORK — Concurrent	53.72 ± 52.19	74.92 ± 71.39	81.69 ± 76.66
TD3-FORK — PS	66.37 ± 36.13	71.32 ± 41.40	76.44 ± 31.63
DDPG — PS	69.52 ± 38.57	73.21 ± 29.49	75.89 ± 32.83
TD3 — PS	64.14 ± 99.31	72.93 ± 110.74	71.72 ± 107.94
TD4-FORK — PS	46.26 ± 44.39	52.97 ± 61.03	57.93 ± 46.61
TD3-FORK — Concurrent	29.50 ± 58.41	49.89 ± 69.70	51.17 ± 79.36
MADDPG — CT-DE	36.53 ± 121.71	42.91 ± 125.80	46.68 ± 130.79
Soft MADDPG — CT-DE	-16.09 ± 68.93	-7.41 ± 69.93	-8.69 ± 71.27
DDPG — Centralized	-76.77 ± 43.85	-76.46 ± 43.56	-76.39 ± 40.30

Table 6.3: Training of MPE Simple Target Continuous

Table 6.3 of "Cooperative Multi-Agent Reinforcement Learning", https://dspace.cuni.cz/handle/20.500.11956/127431

MARL Evaluation, Simple Target, Discrete Actions

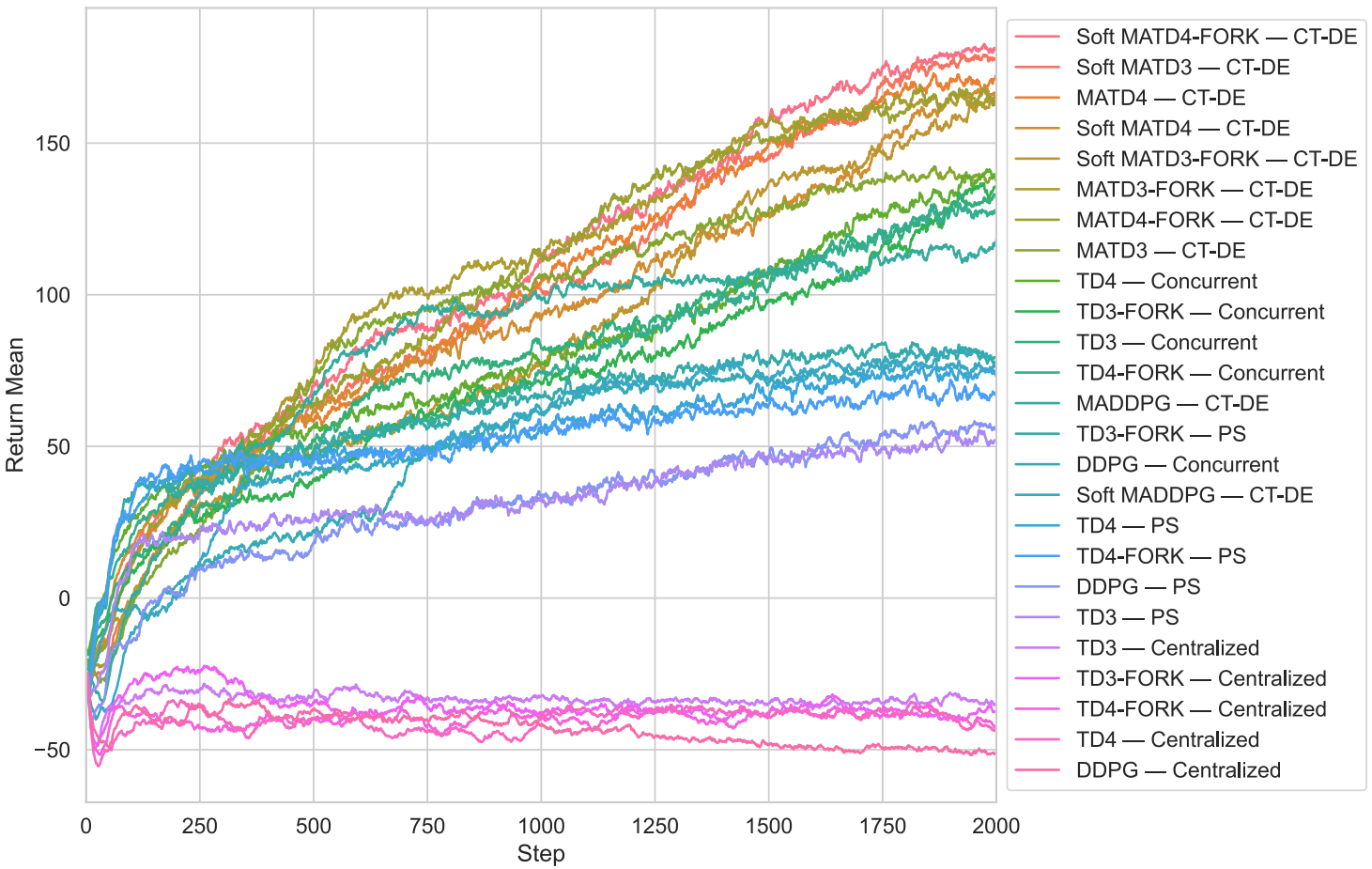


Figure 6.5: Training of MPE Simple Target Discrete

Figure 6.5 of "Cooperative Multi-Agent Reinforcement Learning", https://dspace.cuni.cz/handle/20.500.11956/127431

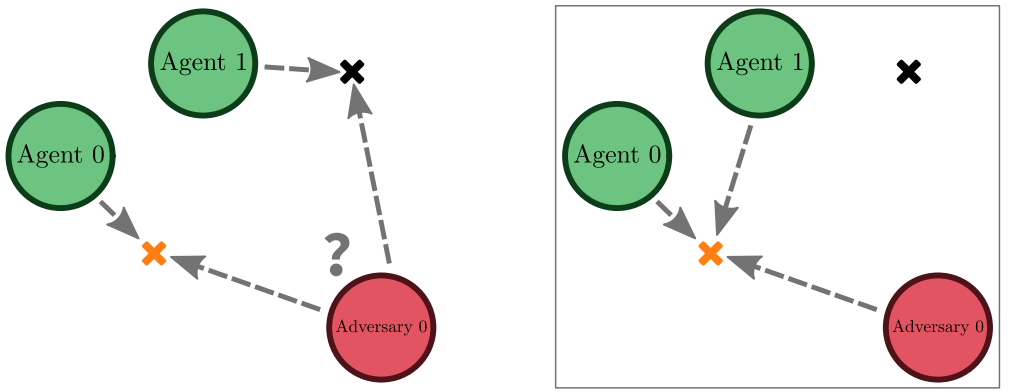
MARL Evaluation, Simple Target, Discrete Actions

Algorithm — Scheme	Step 1000	Step 1500	Step 2000
Soft MATD4-FORK — CT-DE	111.75 ± 77.96	**157.47 ± 75.75**	**181.75 ± 53.02**
Soft MATD3 — CT-DE	101.32 ± 83.43	145.85 ± 88.63	177.61 ± 89.56
MATD4 — CT-DE	104.05 ± 66.35	150.05 ± 87.94	172.31 ± 46.00
Soft MATD4 — CT-DE	92.24 ± 75.62	126.30 ± 84.29	166.52 ± 98.42
Soft MATD3-FORK — CT-DE	75.79 ± 70.77	137.09 ± 104.59	165.54 ± 84.61
MATD3-FORK — CT-DE	**112.65 ± 65.49**	156.95 ± 78.50	163.67 ± 53.54
MATD4-FORK — CT-DE	111.99 ± 64.69	150.89 ± 85.10	163.32 ± 65.31
MATD3 — CT-DE	105.28 ± 69.32	127.97 ± 67.64	138.51 ± 72.62
TD4 — Concurrent	76.77 ± 42.61	106.56 ± 50.66	137.84 ± 47.28
TD3-FORK — Concurrent	70.46 ± 61.78	98.35 ± 98.07	135.50 ± 84.69
TD3 — Concurrent	82.29 ± 59.64	109.08 ± 57.14	131.59 ± 47.93
TD4-FORK — Concurrent	74.34 ± 50.74	102.12 ± 70.92	128.22 ± 73.07
MADDPG — CT-DE	98.26 ± 92.25	107.20 ± 112.55	118.11 ± 95.47
TD3-FORK — PS	66.57 ± 36.22	77.96 ± 34.42	79.53 ± 29.01
DDPG — Concurrent	61.63 ± 67.56	71.28 ± 70.27	77.90 ± 67.64
Soft MADDPG — CT-DE	60.59 ± 100.51	71.92 ± 106.74	75.72 ± 108.28
TD4 — PS	54.58 ± 41.65	65.69 ± 36.96	74.00 ± 40.12
TD4-FORK — PS	56.13 ± 37.18	63.85 ± 44.07	66.86 ± 34.37
DDPG — PS	33.35 ± 69.91	46.41 ± 78.70	55.19 ± 82.42
TD3 — PS	32.00 ± 73.67	46.71 ± 85.10	52.29 ± 87.22
TD3 — Centralized	-32.49 ± 32.23	-34.73 ± 28.95	-34.45 ± 32.21
TD3-FORK — Centralized	-38.20 ± 34.78	-38.26 ± 31.54	-37.41 ± 32.86
TD4-FORK — Centralized	-39.79 ± 27.42	-37.66 ± 28.74	-42.60 ± 23.18
TD4 — Centralized	-38.51 ± 30.20	-38.95 ± 30.45	-42.84 ± 36.64
DDPG — Centralized	-41.75 ± 29.24	-48.40 ± 31.10	-50.95 ± 31.73

Table 6.4: Training of MPE Simple Target Discrete

Table 6.4 of "Cooperative Multi-Agent Reinforcement Learning", https://dspace.cuni.cz/handle/20.500.11956/127431

MARL Evaluation, Simple Confuse



(a) Optimal strategy (b) Suboptimal strategy

Figure 6.3: Physical Deception

Figure 6.3 of "Cooperative Multi-Agent Reinforcement Learning", https://dspace.cuni.cz/handle/20.500.11956/127431

Some number of cooperaing agents gets rewarded based on the minimum distance of any agent to the target landmark; but are penalized based on the distance of a single adversary to the target landmark.

The adversary gets rewarded based on its distance to the target landmark; however, it does not know which landmark is the target one.

Actions can be again either discrete or continuous.

MARL Evaluation, Simple Confuse, Continuous Actions

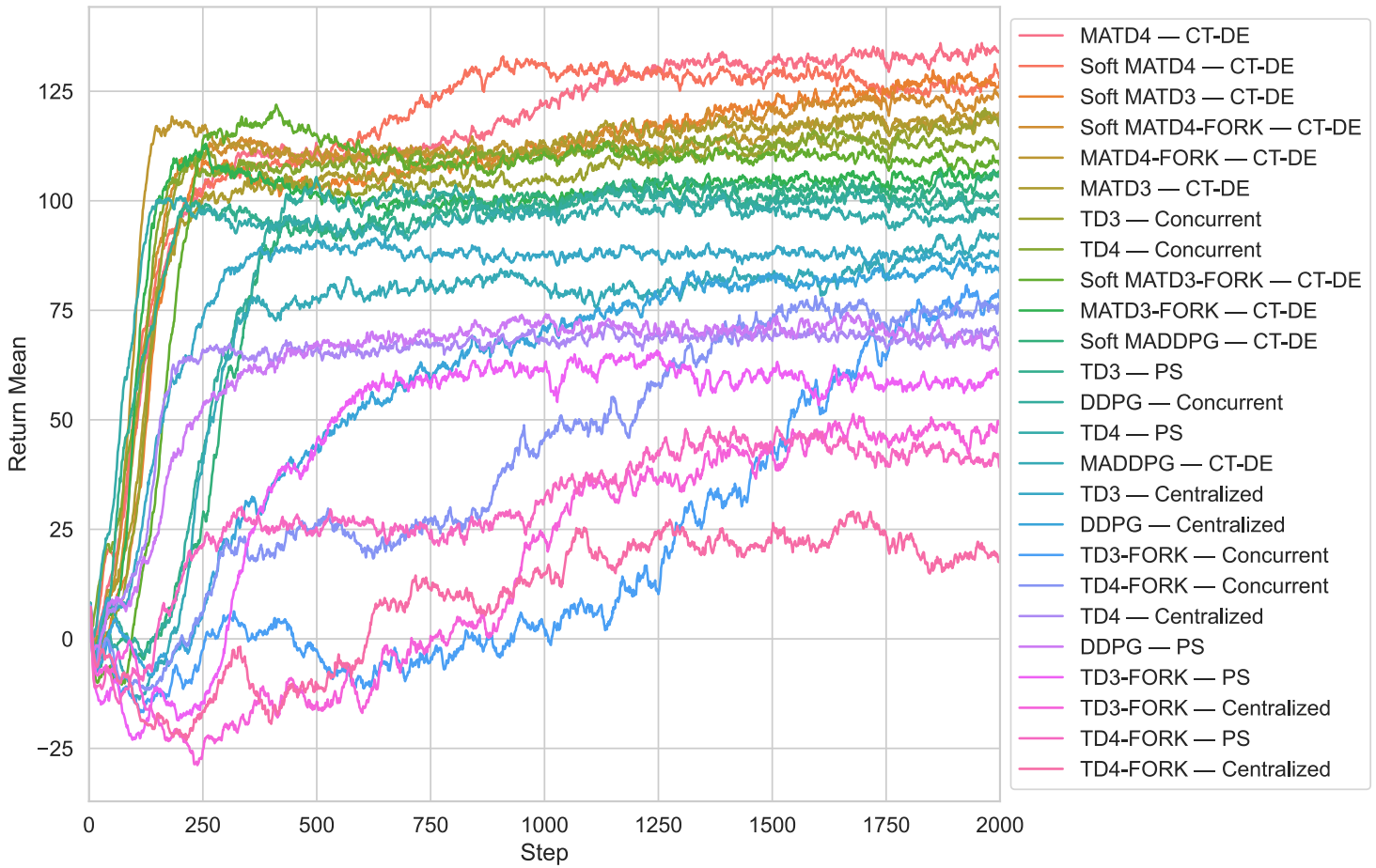


Figure 6.6: Training of MPE Simple Confuse Continuous

Figure 6.6 of "Cooperative Multi-Agent Reinforcement Learning", https://dspace.cuni.cz/handle/20.500.11956/127431

MARL Evaluation, Simple Confuse, Continuous Actions

Algorithm — Scheme	Step 1000	Step 1500	Step 2000
MATD4 — CT-DE	122.18 ± 33.17	**131.30 ± 30.50**	**133.66 ± 27.11**
Soft MATD4 — CT-DE	**131.06 ± 37.29**	129.13 ± 31.19	127.84 ± 28.16
Soft MATD3 — CT-DE	112.32 ± 29.91	121.77 ± 28.70	126.46 ± 35.92
Soft MATD4-FORK — CT-DE	111.54 ± 21.17	120.62 ± 33.04	123.30 ± 33.84
MATD4-FORK — CT-DE	112.46 ± 31.82	115.03 ± 32.07	119.04 ± 36.35
MATD3 — CT-DE	113.12 ± 30.99	116.09 ± 30.69	118.65 ± 29.63
TD3 — Concurrent	105.01 ± 22.77	111.73 ± 31.89	116.86 ± 33.29
TD4 — Concurrent	111.86 ± 21.07	112.31 ± 21.51	112.78 ± 24.54
Soft MATD3-FORK — CT-DE	111.02 ± 27.48	110.57 ± 32.56	107.96 ± 25.47
MATD3-FORK — CT-DE	101.94 ± 19.59	105.07 ± 22.73	106.14 ± 31.57
Soft MADDPG — CT-DE	100.63 ± 20.79	103.23 ± 24.06	105.31 ± 21.57
TD3 — PS	98.97 ± 128.10	100.34 ± 123.74	100.67 ± 127.73
DDPG — Concurrent	97.02 ± 32.39	100.35 ± 26.67	98.67 ± 24.01
TD4 — PS	96.92 ± 123.85	98.33 ± 120.15	96.89 ± 124.90
MADDPG — CT-DE	80.88 ± 111.29	82.77 ± 105.02	91.45 ± 119.99
TD3 — Centralized	87.51 ± 109.69	87.82 ± 108.53	88.24 ± 102.05
DDPG — Centralized	70.40 ± 109.99	82.32 ± 109.73	83.46 ± 115.35
TD3-FORK — Concurrent	2.43 ± 85.80	42.99 ± 69.73	77.41 ± 37.24
TD4-FORK — Concurrent	45.92 ± 84.69	72.06 ± 90.34	76.21 ± 74.87
TD4 — Centralized	68.44 ± 118.04	68.00 ± 120.47	70.01 ± 116.58
DDPG — PS	73.56 ± 118.83	72.27 ± 124.91	66.27 ± 125.84
TD3-FORK — PS	62.00 ± 114.66	61.09 ± 113.96	60.19 ± 119.57
TD3-FORK — Centralized	22.97 ± 98.41	43.69 ± 93.67	49.01 ± 102.98
TD4-FORK — PS	31.90 ± 118.33	44.34 ± 122.45	38.96 ± 115.48
TD4-FORK — Centralized	14.61 ± 101.07	24.73 ± 104.00	18.10 ± 105.13

Table 6.5: Training of MPE Simple Confuse Continuous

Table 6.5 of "Cooperative Multi-Agent Reinforcement Learning", https://dspace.cuni.cz/handle/20.500.11956/127431

MARL Evaluation, Simple Confuse, Discrete Actions

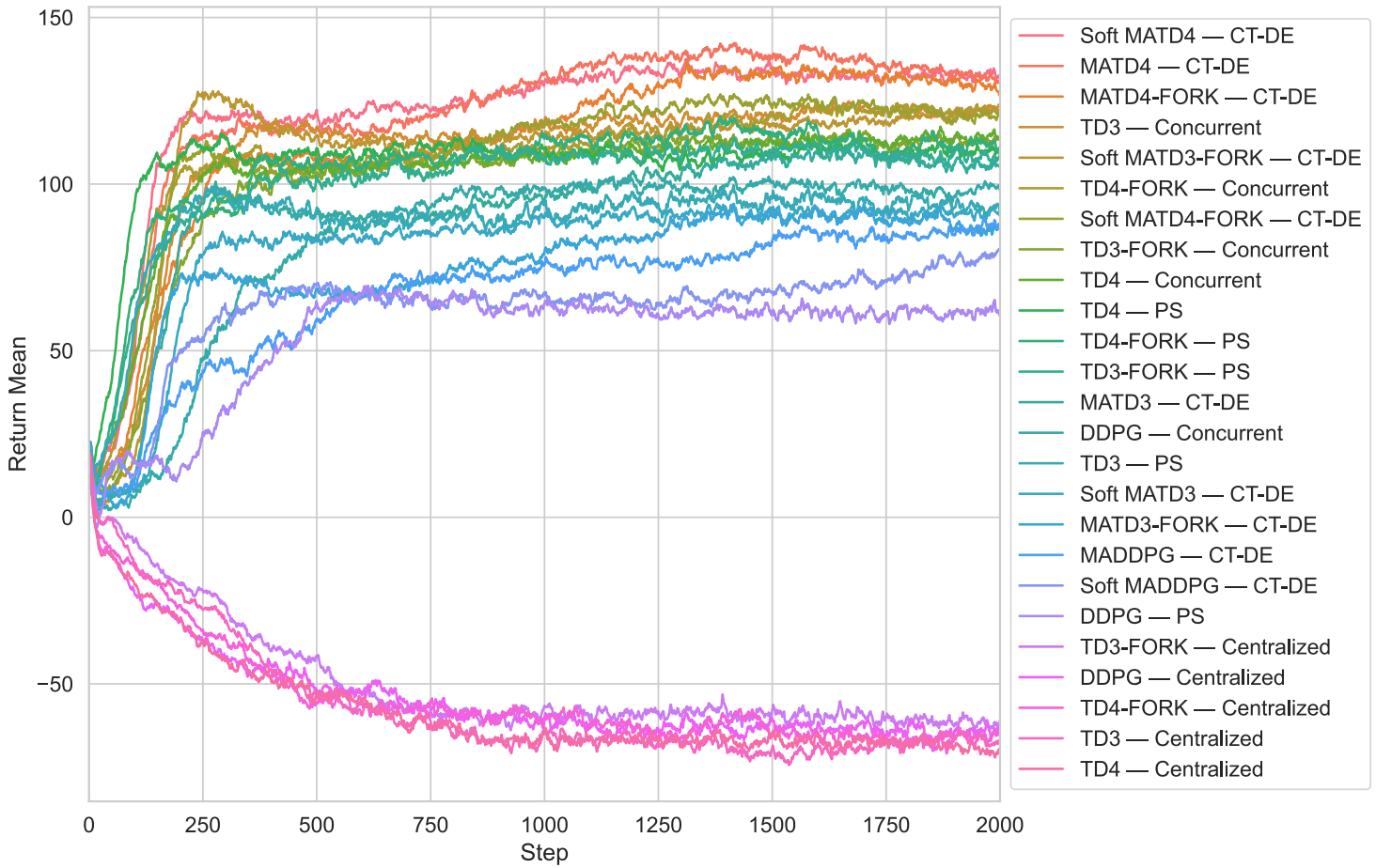


Figure 6.7: Training of MPE Simple Confuse Discrete

Figure 6.7 of "Cooperative Multi-Agent Reinforcement Learning", https://dspace.cuni.cz/handle/20.500.11956/127431

MARL Evaluation, Simple Confuse, Discrete Actions

Algorithm — Scheme	Step 1000	Step 1500	Step 2000
Soft MATD4 — CT-DE	130.73 ± 64.02	134.33 ± 67.35	**131.67 ± 60.95**
MATD4 — CT-DE	**133.16 ± 62.79**	**137.66 ± 60.18**	130.77 ± 62.34
MATD4-FORK — CT-DE	118.88 ± 30.88	133.05 ± 35.41	126.85 ± 31.52
TD3 — Concurrent	117.66 ± 70.83	121.21 ± 68.82	122.93 ± 73.97
Soft MATD3-FORK — CT-DE	114.68 ± 76.54	117.71 ± 75.47	121.95 ± 73.92
TD4-FORK — Concurrent	110.27 ± 25.70	120.18 ± 32.67	121.65 ± 33.89
Soft MATD4-FORK — CT-DE	117.52 ± 36.84	122.71 ± 40.63	120.39 ± 33.28
TD3-FORK — Concurrent	109.80 ± 28.23	112.24 ± 31.18	113.66 ± 31.04
TD4 — Concurrent	107.70 ± 25.13	112.05 ± 30.07	113.13 ± 32.46
TD4 — PS	110.53 ± 93.17	108.77 ± 95.13	111.26 ± 90.47
TD4-FORK — PS	113.83 ± 136.90	114.20 ± 129.01	108.52 ± 115.81
TD3-FORK — PS	107.67 ± 93.65	109.83 ± 93.05	107.64 ± 96.69
MATD3 — CT-DE	95.42 ± 91.09	107.58 ± 103.40	105.22 ± 104.55
DDPG — Concurrent	98.94 ± 27.85	100.38 ± 26.36	98.39 ± 25.54
TD3 — PS	94.36 ± 148.68	95.72 ± 151.30	92.01 ± 153.11
Soft MATD3 — CT-DE	89.92 ± 105.15	91.97 ± 109.25	91.08 ± 108.21
MATD3-FORK — CT-DE	79.58 ± 108.00	90.22 ± 110.49	87.85 ± 109.62
MADDPG — CT-DE	76.21 ± 79.41	82.52 ± 86.87	87.00 ± 92.56
Soft MADDPG — CT-DE	67.05 ± 78.25	67.61 ± 75.00	80.64 ± 65.77
DDPG — PS	64.00 ± 155.87	60.61 ± 157.34	60.46 ± 159.04
TD3-FORK — Centralized	-56.55 ± 34.65	-58.02 ± 33.39	-60.90 ± 40.31
DDPG — Centralized	-59.94 ± 37.19	-60.30 ± 36.79	-64.07 ± 38.17
TD4-FORK — Centralized	-60.94 ± 33.82	-65.66 ± 32.61	-65.76 ± 32.89
TD3 — Centralized	-63.76 ± 30.93	-69.51 ± 28.80	-67.00 ± 32.73
TD4 — Centralized	-65.01 ± 34.23	-65.31 ± 34.02	-70.42 ± 35.32

Table 6.6: Training of MPE Simple Confuse Discrete

Table 6.6 of "Cooperative Multi-Agent Reinforcement Learning", https://dspace.cuni.cz/handle/20.500.11956/127431

Multi-Agent Hide-and-Seek

As another example, consider https://openai.com/blog/emergent-tool-use/.

Here the agents are trained in the centralized training with decentralized execution settings, using the PPO algorithm.

Reinforcement Learning with Human Feedback

Reinforcement Learning with Human Feedback

Let us assume our goal is to train a robot to cook eggs.

For such complex tasks, there might not be suitable reward functions available, and implementing it manually might be difficult or infeasible.

If we have demonstrations for the seeked behavior, we might use inverse reinforcement learning or imitation learning, but some behaviors might be difficult to demonstrate (like controlling a robot with many degrees of freedom).

Reinforcement Learning with Human Feedback

A possible approach in such a case is to use **reinforcement learning with human feedback**, proposed in a 2017 paper.

Because human feedback is costly, using as little as feedback is desirable. Therefore, the authors propose to use the feedback to train a reward function, which can then be used by classical RL algorithms.

A practical approach is for the human raters to *compare two video clips of agent's behavior*.

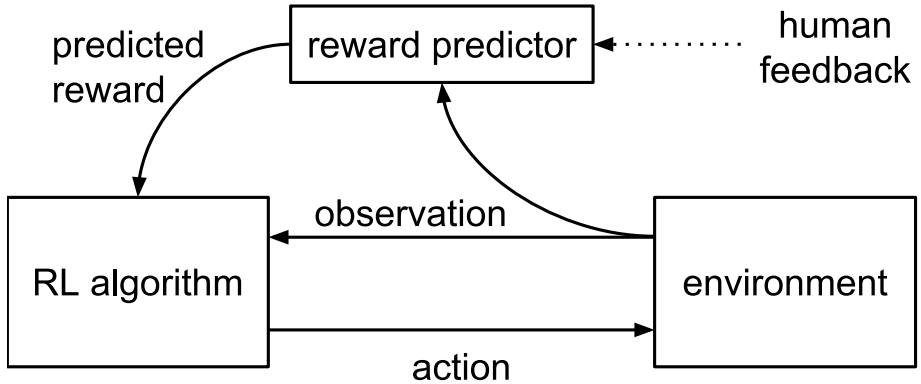


Figure 1: Schematic illustration of our approach: the reward predictor is trained asynchronously from comparisons of trajectory segments, and the agent maximizes predicted reward.

Figure 1 of "Deep Reinforcement Learning from Human Preferences", https://arxiv.org/abs/1706.03741

- Comparing two videos was found out to be considerably easier than providing absolute numerical score.
- Rating videos is nearly as fast as comparing individual states, but significantly more helpful.

General Setting

We consider a partially observable MDP, where in time t agent obtains an observation o_t and produces an action a_t.

In the traditional RL setting, the environment would also produce the reward r_t; we will instead predict a reward $\hat{r}(o_t, a_t)$.

A **trajectory segment** is a sequence of observations and actions

$$\sigma = ((o_0, a_0), (o_1, a_1), \ldots, (o_{k-1}, a_{k-1})).$$

We write $\sigma^1 \succ \sigma^2$ to indicate that a human rater preferred trajectory σ^1 over trajectory σ^2.

We maintain two models:

- the policy $\pi : \mathcal{O} \to \mathcal{A}$, and
- the reward function estimate $\hat{r} : \mathcal{O} \times \mathcal{A} \to \mathcal{R}$.

The models are updated by three asynchronously running processes:

1. The policy π interacts with the environment, obtaining a set of trajectories $\{\tau^1, \ldots, \tau^i\}$.

   The parameters of π are updated using a traditional RL algorithm utilizing the estimated reward $\hat{r}$.

2. A pair of segments (σ^1, σ^2) from trajectories $\{\tau^1, \ldots, \tau^i\}$ generated in step 1 is selected and sent to human rater for evaluation.

3. The reward estimate $\hat{r}$ is trained using supervised learning to fit the comparisons selected by the annotators so far.

Optimizing the Policy

During policy optimization, we can utilize the usual RL algorithms. The main difference is that the rewards are non-stationary; therefore, policy-gradient methods are preferred.

During training we normalize $\hat{r}$ to have zero mean and constant standard deviation (the absolute values of the rewards are undetermined in our model).

Preference Collection

The visualization given to a rater is a video between 1 and 2 seconds long. The rater indicate which of the segments they prefer, whether the segments are equally good, of that they were not able to compare the two segments.

All human judgements are stored in a dataset $\mathcal{D}$ of triples $(\sigma^1, \sigma^2, \boldsymbol{\mu})$, where σ^1, σ^2 are the compared segments and $\boldsymbol{\mu}$ is a distribution over $1, 2$ indicating which segment was deemed better. The $\boldsymbol{\mu}$ is either one hot distribution when a single segment was preferred, or a uniform distribution when a rater annotated both videos as being comparable.

Reward Function Fitting

We follow the Bradley-Terry model, which is a binary probability model for the outcome of pairwise comparisons between players

$$P(\underbrace{i > j}_{\text{“}i \text{ beats } j\text{”}}) = \frac{p_i}{p_i + p_j},$$

where p_i is a positive real-valued score assigned to player i.

There are many possible parametrizations, Bradley and Terry used $p_i = e^{\beta_i}$, in which case

$$P(i > j) = \frac{e^{\beta_i}}{e^{\beta_i} + e^{\beta_j}} = \frac{1}{1 + e^{\beta_j - \beta_i}} = \sigma(\beta_i - \beta_j).$$

Note that with a scaling factor 400 and base 10, this is equivalent to the Elo rating system with Elo ratings R_i and R_j:

$$P(i > j) = \frac{1}{1 + 10^{(R_j - R_i)/400}} = \sigma\big((R_i - R_j)\tfrac{\log 10}{400}\big).$$

Reward Function Fitting

In our case, we assume that

$$\hat{P}(\sigma^1 \succ \sigma^2) = \frac{\exp \sum \hat{r}(o_t^1, a_t^1)}{\exp \sum \hat{r}(o_t^1, a_t^1) + \exp \sum \hat{r}(o_t^2, a_t^2)} = \sigma\big(\sum \hat{r}(o_t^1, a_t^1) - \sum \hat{r}(o_t^2, a_t^2)\big).$$

To fit this model we minimize the cross-entropy between the observed distribution of human rating $\boldsymbol{\mu}$ and these predictions:

$$\mathcal{L} = -\mathbb{E}_{(\sigma^1,\sigma^2,\boldsymbol{\mu})\sim\mathcal{D}}\big[\mu_1 \log \hat{P}(\sigma^1 \succ \sigma^2) + \mu_2 \hat{P}(\sigma^2 \succ \sigma^1)\big].$$

The authors mention several tricks:

- A whole ensemble of predictors are trained, each using randomly sampled $|\mathcal{D}|$ examples (with replacement) from $\mathcal{D}$. The overall $\hat{r}$ is then defined by first independently normalizing each of the predictors and then averaging the results.
- A fraction $1/e$ of the data is used as development data, and strength of L^2-regularization is set so that the development loss is between 1.1 and 1.5 of the training loss.
- In the definition of $\hat{P}$, we assume there is a 10% chance of uniform random outcome.

Reward Function Fitting

To select the queries for the raters, we

- sample a large amount of pairs of trajectory segments of length k,
- perform prediction on each pair using all the predictors in our ensemble,
- choose the pairs with the highest variance in the predictions, which approximates the largest uncertainty.

Experiments

In first experiments, the authors attempted to solve existing RL benchmarks (Atari, MuJoCo) without observing the true rewards, using human feedback instead.

For comparison, synthetic feedback (instead of human feedback) was also considered in these tasks, where $\hat{P}(\sigma^1 \succ \sigma^2)$ is defined by using the real reward of the trajectory segments.

For MuJoCo, 700 human ratings were used; for Atari, 5.5k queries were sent to human raters. Overall the ratings took between 30 minutes and 5 hours for every experiment.

Experiments: MuJoCo

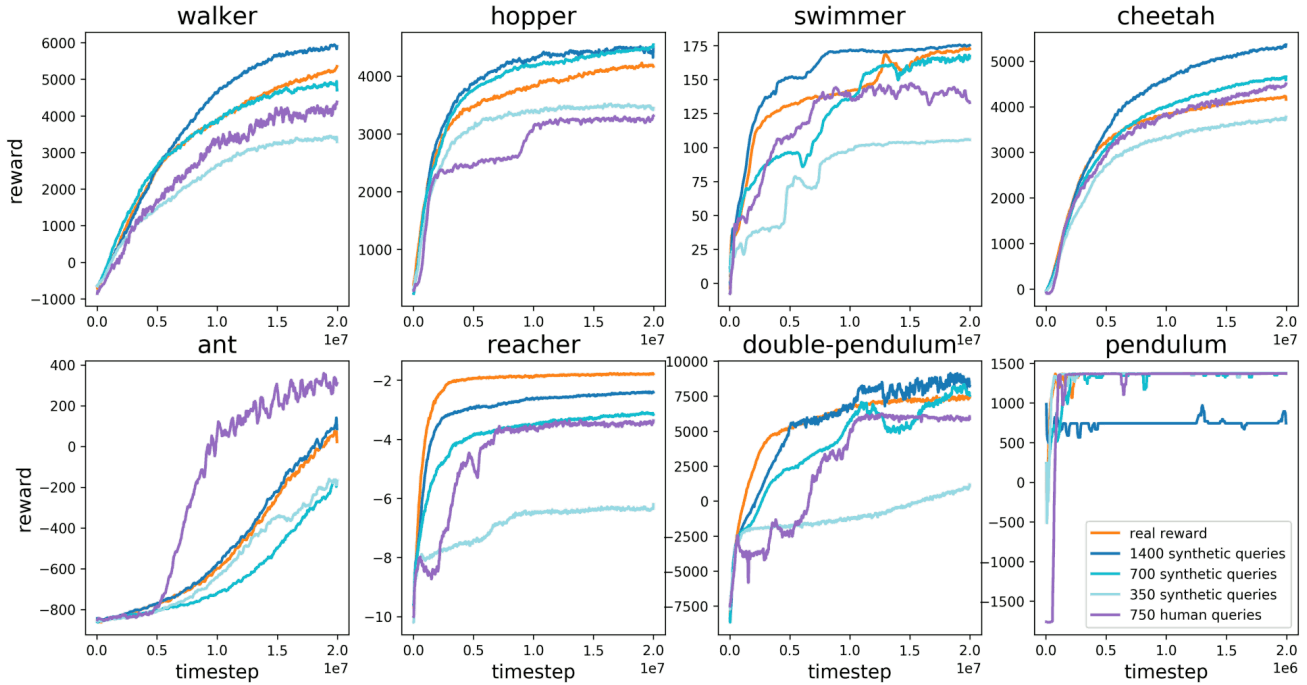


Figure 2: Results on MuJoCo simulated robotics as measured on the tasks' true reward. We compare our method using real human feedback (purple), our method using synthetic feedback provided by an oracle (shades of blue), and reinforcement learning using the true reward function (orange). All curves are the average of 5 runs, except for the real human feedback, which is a single run, and each point is the average reward over five consecutive batches. For Reacher and Cheetah feedback was provided by an author due to time constraints. For all other tasks, feedback was provided by contractors unfamiliar with the environments and with our algorithm. The irregular progress on Hopper is due to one contractor deviating from the typical labeling schedule.

Figure 2 of "Deep Reinforcement Learning from Human Preferences", https://arxiv.org/abs/1706.03741

Experiments: Atari

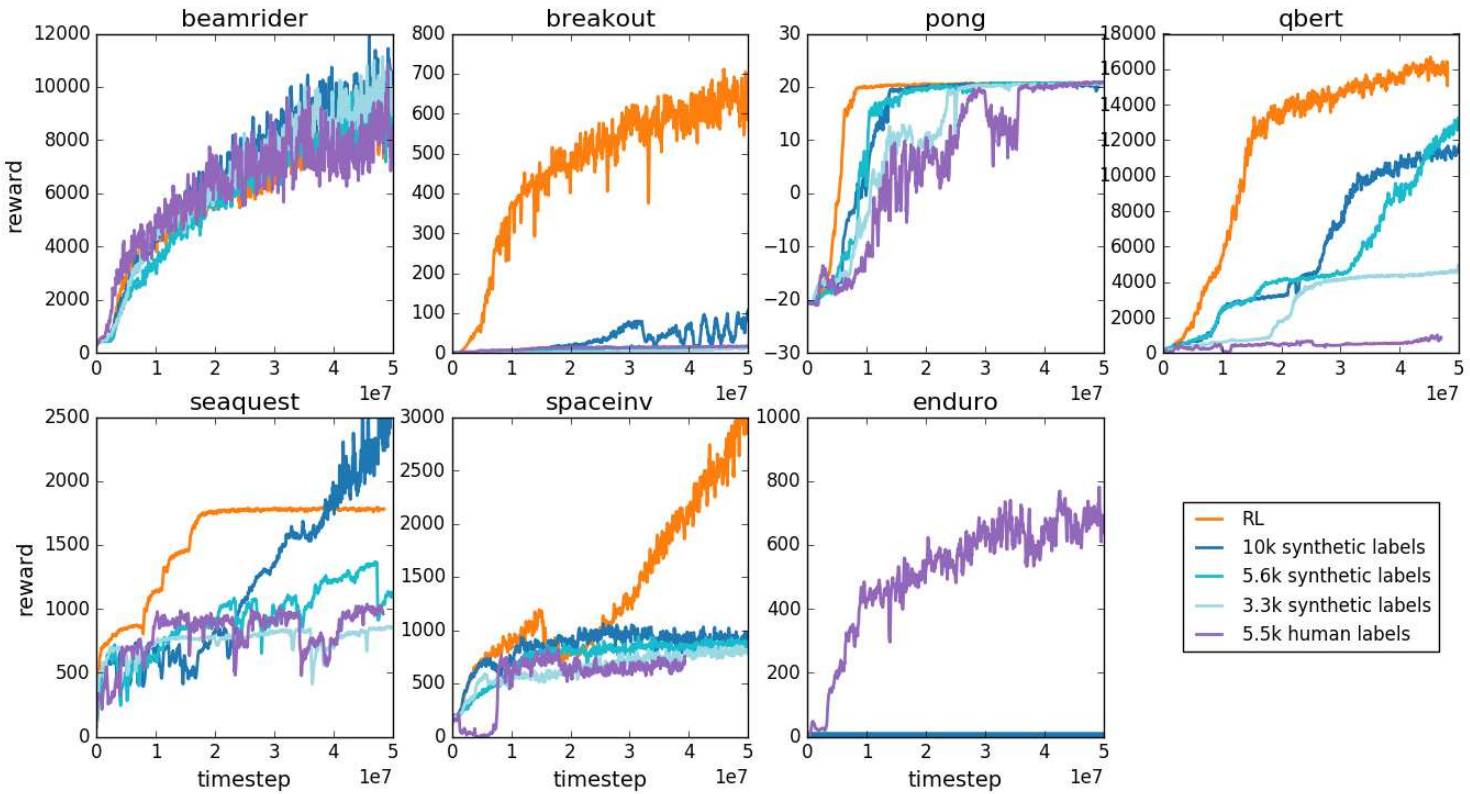


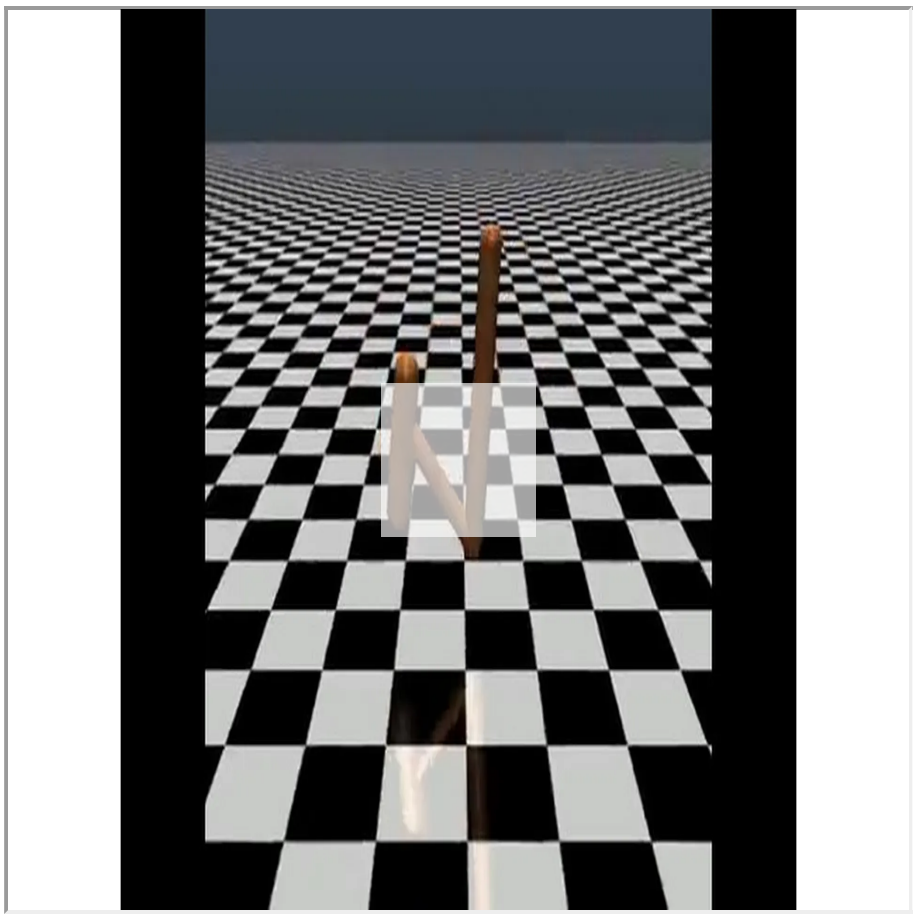
Figure 3: Results on Atari games as measured on the tasks' true reward. We compare our method using real human feedback (purple), our method using synthetic feedback provided by an oracle (shades of blue), and reinforcement learning using the true reward function (orange). All curves are the average of 3 runs, except for the real human feedback which is a single run, and each point is the average reward over about 150,000 consecutive frames.

Figure 3 of "Deep Reinforcement Learning from Human Preferences", https://arxiv.org/abs/1706.03741

Experiments: Novel Behavior

The authors also demonstrated the effectiveness of human feedback in tasks without available reward functions:

- the Hopper robot was trained to perform a backflip using 900 queries in less than an hour;
- the Half-Cheetah was trained to move while standing on just one leg via 800 queries;
- in Enduro, the agent was trained to keep the same speed as other cars using 1 300 queries.



Ablations: MuJoCo

- random queries: queries picked uniformly at random instead of according to variance of predictions;
- no ensemble: a single predictor is trained; random queries are implied;
- no online queries: all queries generated at the beginning of the training;
- no regularization: w/o validation L^2 condition;
- no segments: rate only images, not videos;
- target: instead of fitting $\hat{r}$ using comparison, we fit given real r using MSE.

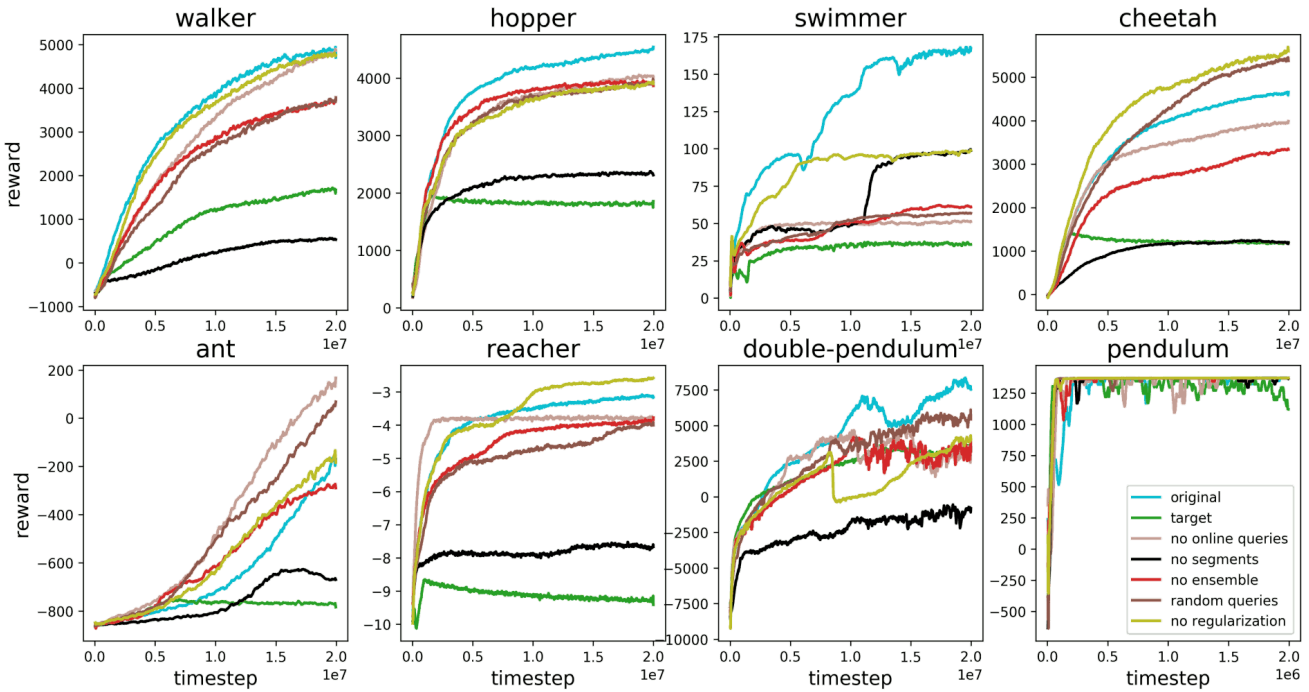


Figure 5: Performance of our algorithm on MuJoCo tasks after removing various components, as described in Section Section 3.3. All graphs are averaged over 5 runs, using 700 synthetic labels each.

Figure 5 of "Deep Reinforcement Learning from Human Preferences", https://arxiv.org/abs/1706.03741

Ablations: Atari

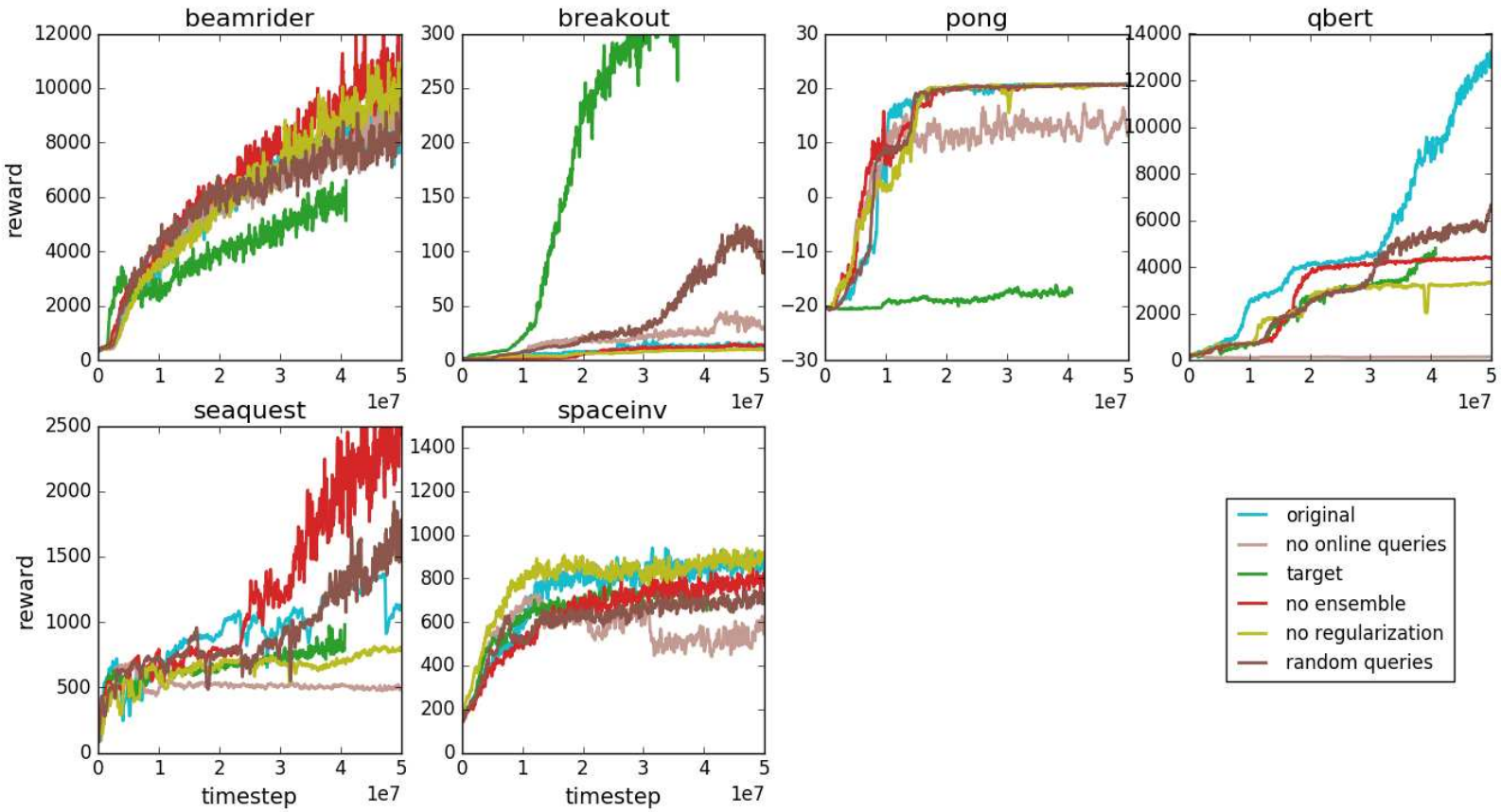


Figure 6: Performance of our algorithm on Atari tasks after removing various components, as described in Section 3.3. All curves are an average of 3 runs using 5,500 synthetic labels (see minor exceptions in Section A.2).

Figure 6 of "Deep Reinforcement Learning from Human Preferences", https://arxiv.org/abs/1706.03741

Instructions: MuJoCo

- Look at the clips and select the one in which better things happen. Only decide on events you actually witness in the clip.
- Here's a guide on what constitutes good and bad behavior in each specific domain:
  - **Hopper**: the "center" of the robot is the joint closest to the pointy end. The first priority is for the center of the robot to move to the right (moving to the left is worse than not moving at all). If the two robots are roughly tied on this metric, then the tiebreaker is how high the center is.
  - **Cheetah**: the robot should move to the right as fast as possible.
  - **Pendulum**: the pendulum should be pointing approximately up. There will be a lot of ties where the pendulum has fallen and a lot of "can't tells" where it is off the side of the screen. If you can see one pendulum and it hasn't fallen down, that's better than being unable to see the other pendulum.
- If both clips look about the same to you, then click "tie". If you don't understand what's going on in the clip or find it hard to evaluate, then click "can't tell".

Instructions: Atari

- First play the game yourself for 5 minutes. Before providing feedback to the AI, play the game yourself for a five minutes to get a sense of how it works. It's often hard to tell what the game is about just by looking at short clips, especially if you've never played it before.
- Look at the clips and select the one in which better things happen. For example, if the left clip shows the AI shooting an enemy ship while the right clip shows it being shot by an enemy ship, then better things happen in the left clip and thus the left clip is better. Only decide on actions you actually witness in the clip.
- Here's a guide on what constitutes good and bad play in each specific game:
  - **Breakout**: hit the ball with the paddle, break the colored blocks, and don't let the ball fall off the bottom of the screen
  - **Pong**: knock the ball past the opponent's orange paddle on the left (good), and don't let it go past your green paddle on the right (bad)
- Don't worry about how the agent got into the situation it is in (for instance, it doesn't matter if one agent has more lives, or is now on a more advanced level); just focus on what happens in the clip itself.

Learning to Summarize from Human Feedback

RLHF was used to improve summarization in a 2020 paper.

The Reddit TL;DR dataset and CNN/Daily Mail dataset were utilized. The Reddit TL;DR contains 3M posts from reddit.com with summaries written by original posters; the authors filtered the data (including requiring the summaries to be between 24 and 48 tokens) and kept 123 169 posts with ~5% kept as a validation set.

The following models were considered:

- **Pretrained models**: Pretrained LLMs, with several high-quality examples in the prompt.
- **Supervised baselines**: Finetuned variants of the above models trying to predict the summaries from the filtered TL;DR dataset.
- **Reward models**: Starting from the supervised baselines, RLHF was applied. A new output linear layer producing a single scalar was added to the model and the model was trained to predict rating $r_{\boldsymbol{\theta}}(x, y_i)$ that corresponds the most to the observed ratings according to the Bradley-Terry model:

$$\mathcal{L} = -\mathbb{E}_{(x,y_0,y_1,i)\sim\mathcal{D}}\left[\log(\sigma(r_{\boldsymbol{\theta}}(x, y_i) - r_{\boldsymbol{\theta}}(x, y_{1-i})))\right].$$

Learning to Summarize from Human Feedback

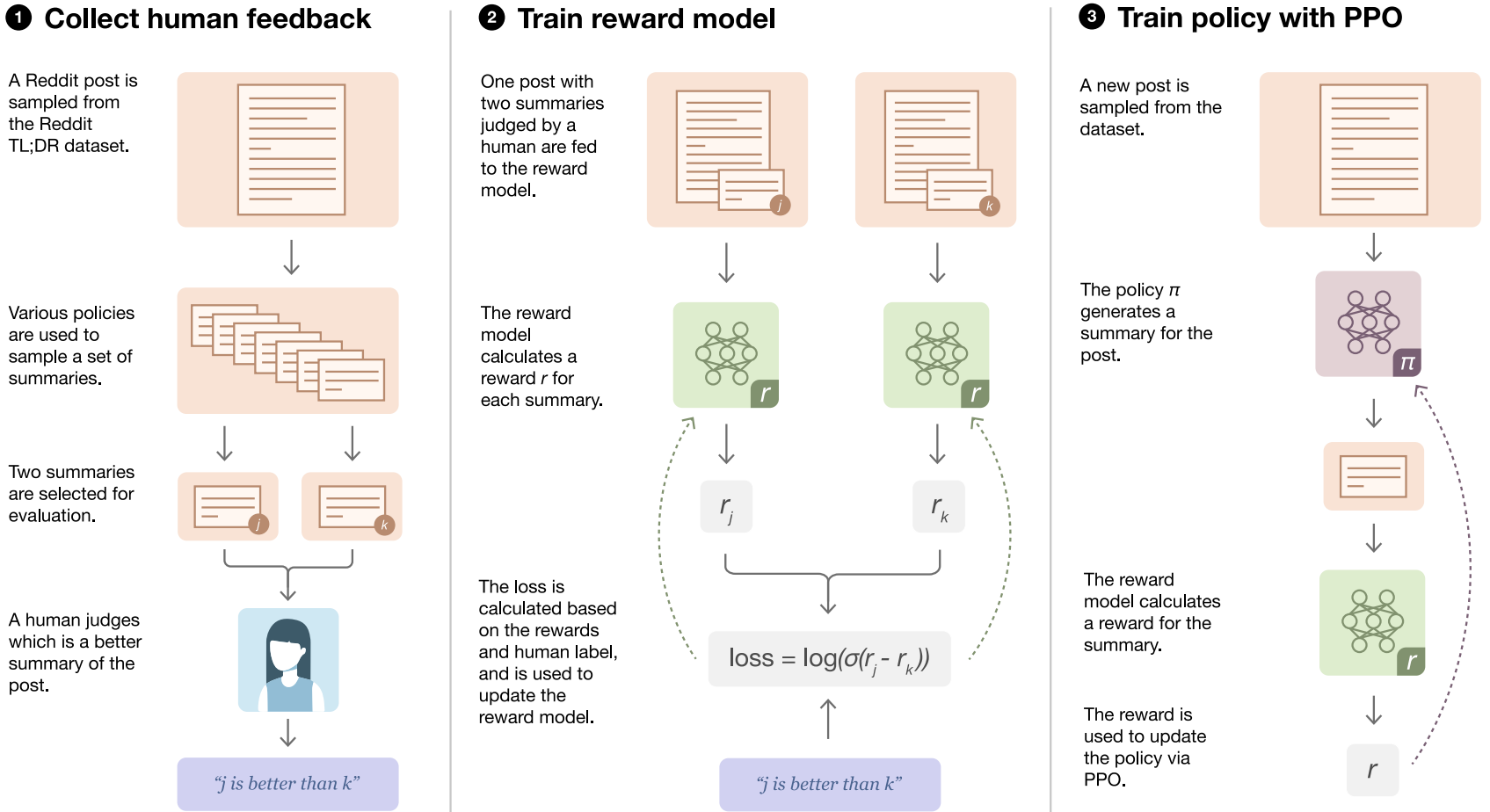


Figure 2: Diagram of our human feedback, reward model training, and policy training procedure.

Figure 2 of "Learning to summarize from human feedback", https://arxiv.org/abs/2009.01325

The human feedback policies are then trained using the reward model.

Generating the entire summary is considered a sequence of actions, each being a generation of a single BPE token, and the PPO algorithm is used with the following reward:

$$R(x, y) \stackrel{\text{def}}{=} r_{\boldsymbol{\theta}}(x, y) - \beta \log \left[\pi^{\text{RL}}_{\boldsymbol{\varphi}}(y|x) / \pi^{\text{SFT}}(y|x)\right],$$

where the per-token KL term serves two purposed:

- it acts as an entropy bonus to avoid collapsing to a single node;
- it ensures the policy does not learn to produce completely different outputs too different from those the reward model saw during reward fitting.

Learning to Summarize from Human Feedback

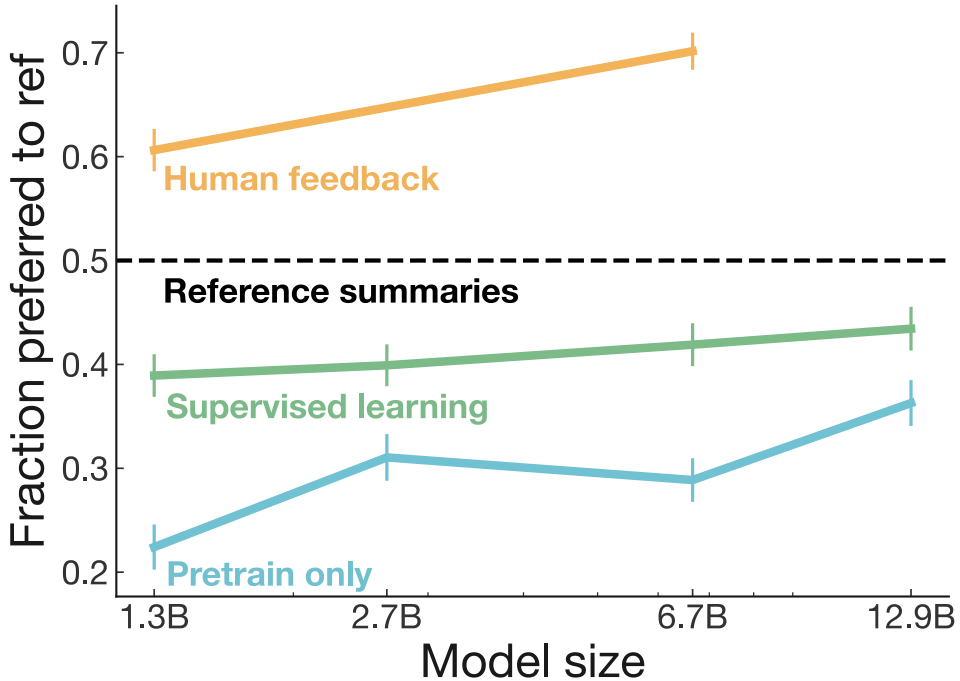


Figure 1: Fraction of the time humans prefer our models' summaries over the human-generated reference summaries on the TL;DR dataset.[4]Since quality judgments involve an arbitrary decision about how to trade off summary length vs. coverage within the 24-48 token limit, we also provide length-controlled graphs in Appendix F; length differences explain about a third of the gap between feedback and supervised learning at 6.7B.

Figure 1 of "Learning to summarize from human feedback", https://arxiv.org/abs/2009.01325

In total, 64 832 ratings were collected, and they were publicly released.

Learning to Summarize from Human Feedback

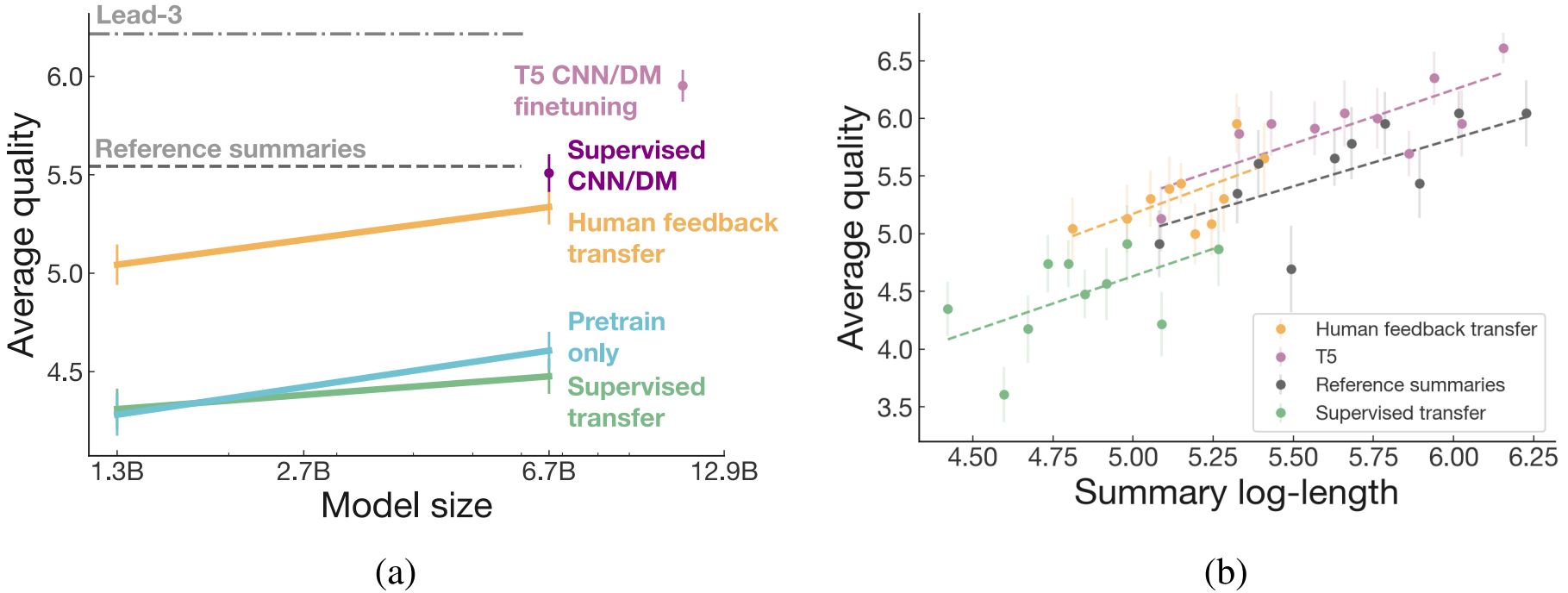


Figure 4: Transfer results on CNN/DM. (a) Overall summary quality on CNN/DM as a function of model size. Full results across axes shown in Appendix G.2. (b) Overall scores vs. length for the 6.7B TL;DR supervised baseline, the 6.7B TL;DR human feedback model, and T5 fine-tuned on CNN/DM summaries. At similar summary lengths, our 6.7B TL;DR human feedback model nearly matches T5 despite never being trained to summarize news articles.

Figure 4 of "Learning to summarize from human feedback", https://arxiv.org/abs/2009.01325

Learning to Summarize from Human Feedback

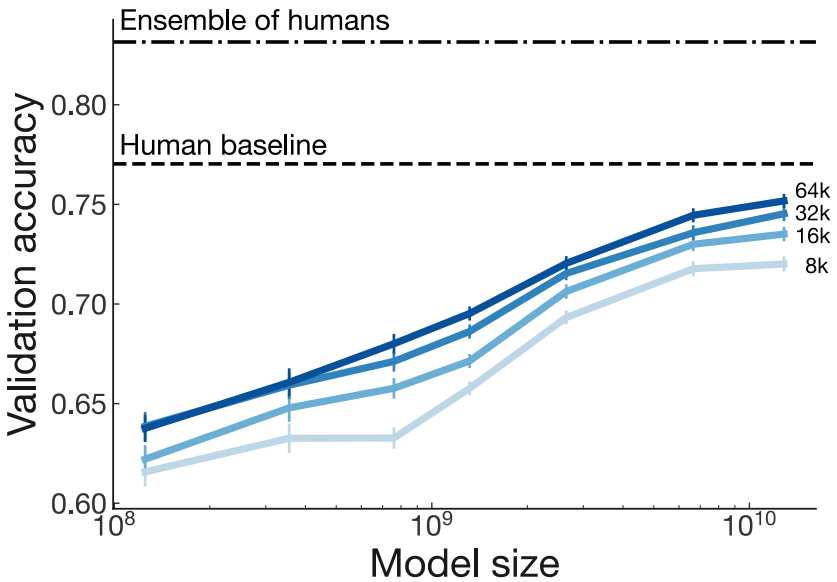


Figure 6: Reward model performance versus data size and model size. Doubling amount of training data leads to a ~1.1% increase in reward model validation accuracy, whereas doubling the model size leads to a ~1.8% increase. The 6.7B model trained on all data begins approaching the accuracy of a single human.

Figure 6 of "Learning to summarize from human feedback", https://arxiv.org/abs/2009.01325

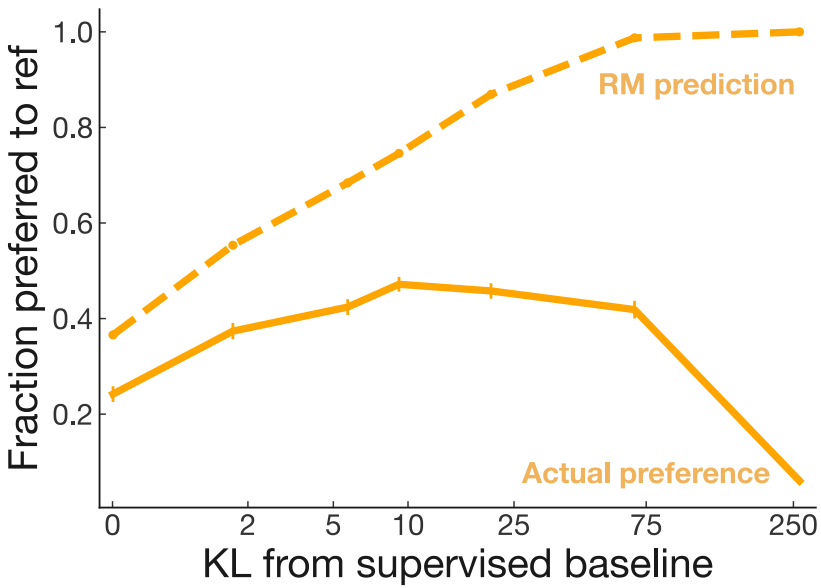


Figure 5: Preference scores versus degree of reward model optimization. Optimizing against the reward model initially improves summaries, but eventually overfits, giving worse summaries. This figure uses an earlier version of our reward model (see rm3 in Appendix C.6). See Appendix H.2 for samples from the KL 250 model.

Figure 5 of "Learning to summarize from human feedback", https://arxiv.org/abs/2009.01325

Training LMs to Follow Instructions with Human Feedback

In 2022, InstructGPT (ChatGPT predecessor) was trained to follow instructions using human feedback.

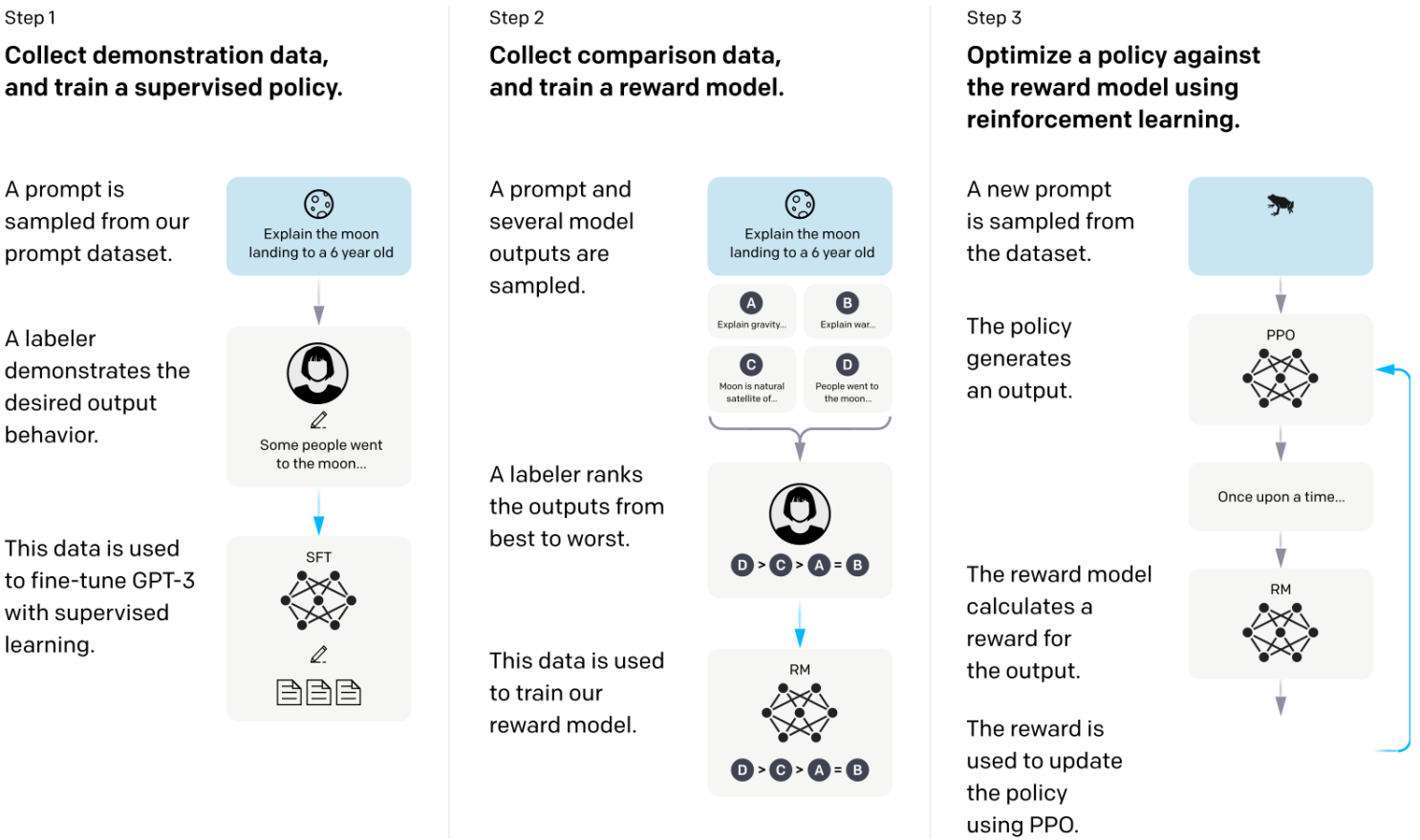


Figure 2 of "Training language models to follow instructions with human feedback", https://arxiv.org/abs/2203.02155

The same Bradley-Terry model is used to train the reward function.

To speed up comparison collection, the authors presented the labelers between $K = 4$ and $K = 9$ responses to rank, producing $\binom{K}{2}$ comparisons for every prompt.

However, the comparisons in a single prompt are very correlated, so sampling them randomly during an epoch caused the model to overfit. Instead, all $\binom{K}{2}$ comparisons were used in a single batch, which is also more efficient (only K passes of the reward model, compared to $\binom{K}{2}$ passes).

The loss function for the reward model is analogous to before:

$$\mathcal{L} = -\frac{1}{\binom{K}{2}} \mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}} \left[\log(\sigma(r_{\boldsymbol{\theta}}(x, y_w) - r_{\boldsymbol{\theta}}(x, y_l))) \right].$$

Training LMs to Follow Instructions with Human Feedback

The SFT models are finetuned using the trained reward model again using the PPO algorithm, employing the following objective:

$$\mathbb{E}_{(x,y)\sim\mathcal{D}_{\pi_{\boldsymbol{\varphi}}^{\mathrm{RL}}}}\left[r_{\boldsymbol{\theta}}(x,y)-\beta\log\left(\pi_{\boldsymbol{\varphi}}^{\mathrm{RL}}(y|x)/\pi^{\mathrm{SFT}}(y|x)\right)\right].$$

The authors also proposed a variant called PPO-ptx., which also includes an additional supervised term in the objective:

$$\mathbb{E}_{(x,y)\sim\mathcal{D}_{\pi_{\boldsymbol{\varphi}}^{\mathrm{RL}}}}\left[r_{\boldsymbol{\theta}}(x,y)-\beta\log\left(\pi_{\boldsymbol{\varphi}}^{\mathrm{RL}}(y|x)/\pi^{\mathrm{SFT}}(y|x)\right)\right]+\gamma\mathbb{E}_{x\sim\mathcal{D}_{\mathrm{pretrain}}}\left[\log(\pi_{\boldsymbol{\varphi}}^{\mathrm{RL}}(x))\right].$$

Training LMs to Follow Instructions with Human Feedback

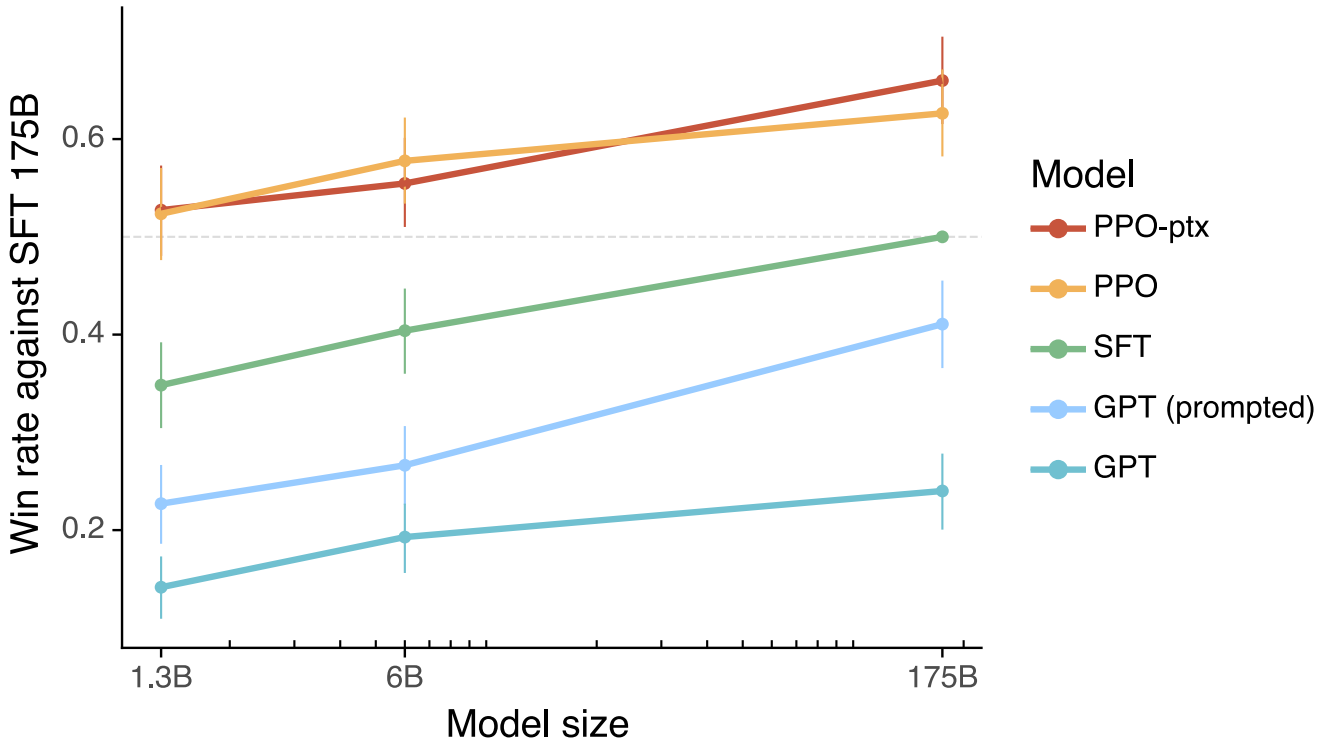


Figure 1: Human evaluations of various models on our API prompt distribution, evaluated by how often outputs from each model were preferred to those from the 175B SFT model. Our InstructGPT models (PPO-ptx) as well as its variant trained without pretraining mix (PPO) significantly outperform the GPT-3 baselines (GPT, GPT prompted); outputs from our 1.3B PPO-ptx model are preferred to those from the 175B GPT-3. Error bars throughout the paper are 95% confidence intervals.

Figure 1 of "Training language models to follow instructions with human feedback", https://arxiv.org/abs/2203.02155

Training LMs to Follow Instructions with Human Feedback

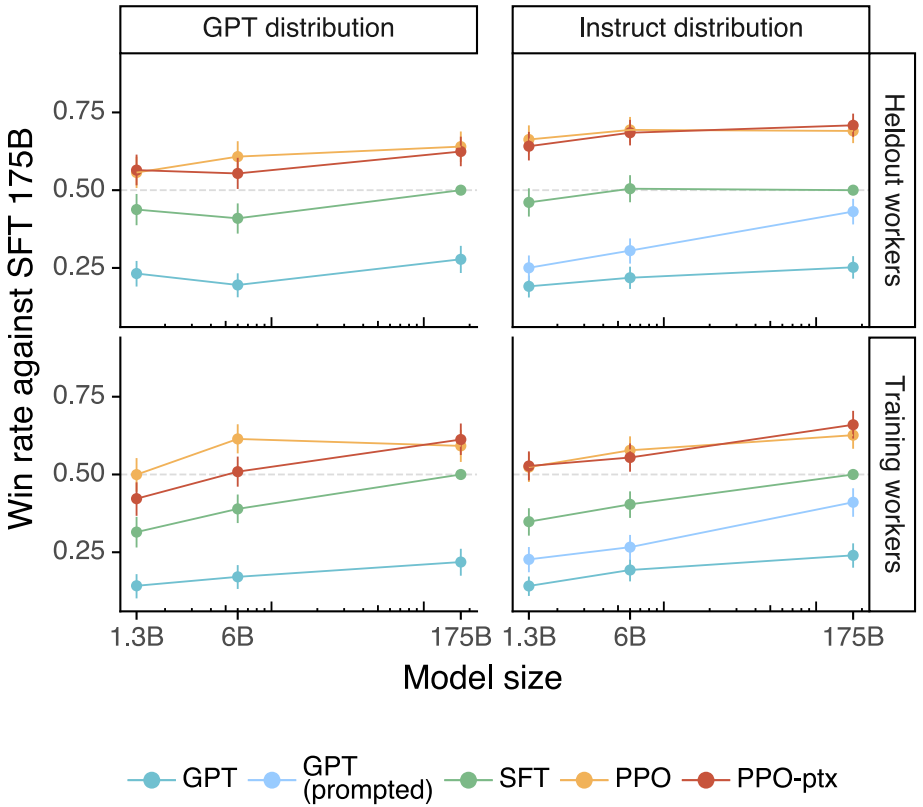


Figure 3: Preference results of our models, measured by winrate against the 175B SFT model. Left: results on prompts submitted to GPT models on the API; Right: results on prompts submitted to InstructGPT models on the API; Top: results from held-out labelers; Bottom: results from training labelers. We omit GPT (prompted) from the evals on prompts submitted to GPT-3 models (left) as these prompts are already designed to perform well for GPT-3, as opposed to prompts submitted to InstructGPT models (right).

Figure 3 of "Training language models to follow instructions with human feedback", https://arxiv.org/abs/2203.02155

Training LMs to Follow Instructions with Human Feedback

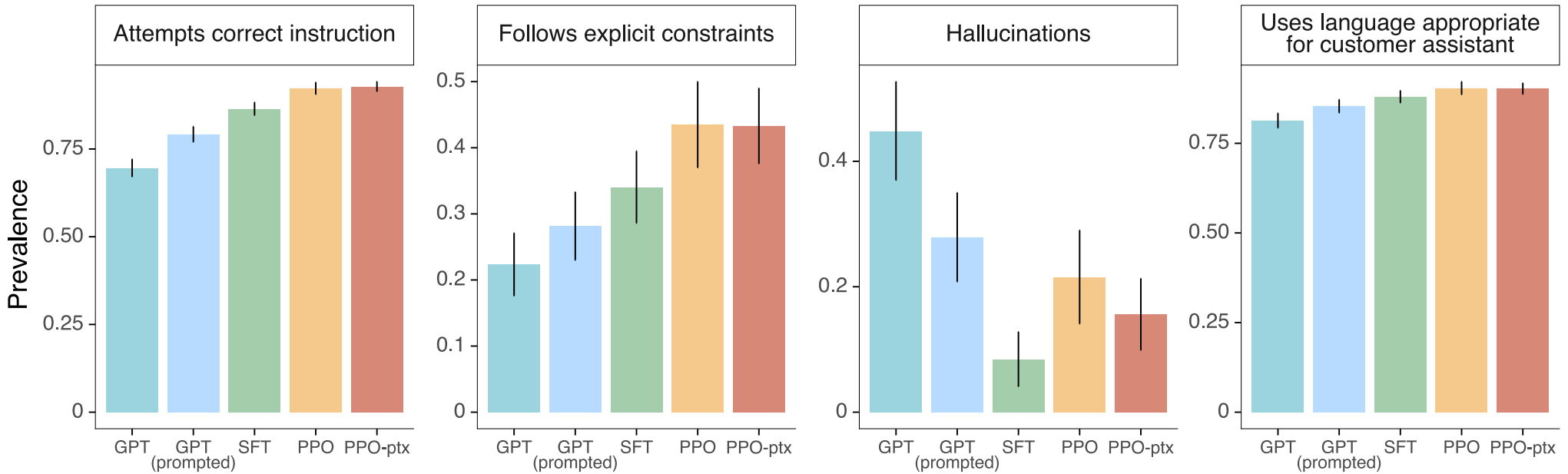


Figure 4: Metadata results on the API distribution. Note that, due to dataset sizes, these results are collapsed across model sizes. See Appendix E.2 for analysis that includes model size. Compared to GPT-3, the PPO models are more appropriate in the context of a customer assistant, are better at following explicit constraints in the instruction and attempting the correct instruction, and less likely to 'hallucinate' (meaning, making up information on closed domain tasks like summarization).

Figure 4 of "Training language models to follow instructions with human feedback", https://arxiv.org/abs/2203.02155

Training LMs to Follow Instructions with Human Feedback

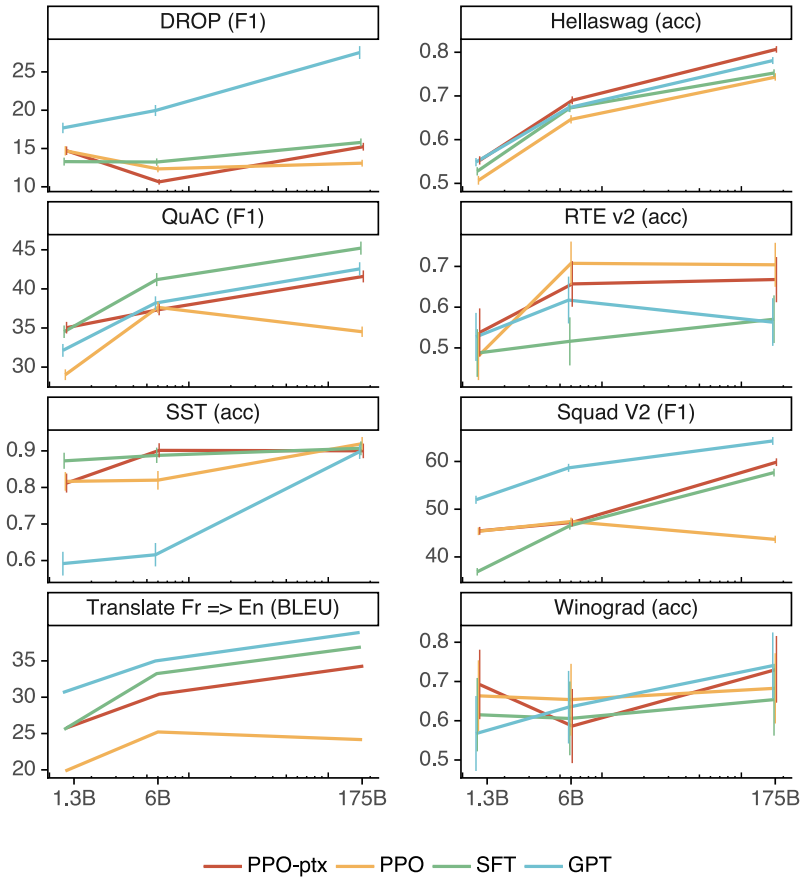


Figure 28: Zero-shot performance of our models on various public NLP datasets. The 175B PPO models consistently show performance regressions, which is mitigated by adding updates on the pretraining data during fine-tuning. Few-shot performance is shown in Figure 29. Error bars for translation are not available because we use a software package that does not report them.

Figure 28 of "Training language models to follow instructions with human feedback", https://arxiv.org/abs/2203.02155

Training LMs to Follow Instructions with Human Feedback

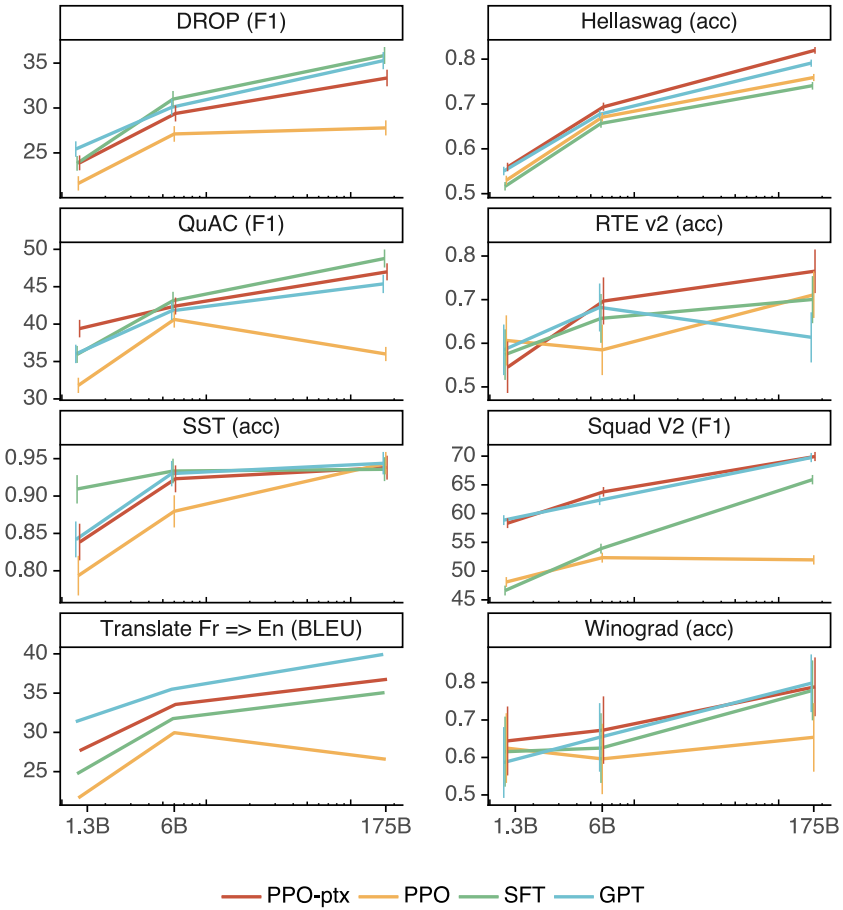


Figure 29: Few-shot performance of our models on various public NLP datasets (compare to zero-shot performance shown in Figure 28

Figure 29 of "Training language models to follow instructions with human feedback", https://arxiv.org/abs/2203.02155

Direct Preference Optimization

Direct Preference Optimization

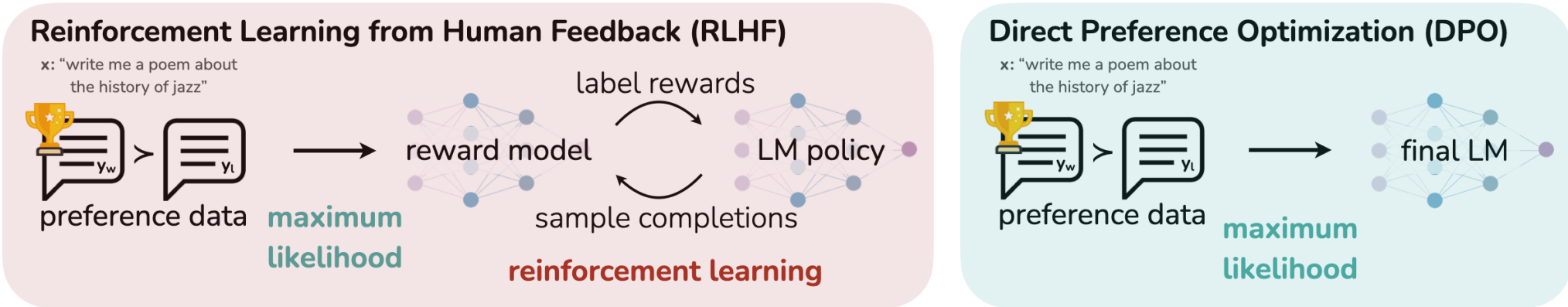


Figure 1: **DPO optimizes for human preferences while avoiding reinforcement learning.** Existing methods for fine-tuning language models with human feedback first fit a reward model to a dataset of prompts and human preferences over pairs of responses, and then use RL to find a policy that maximizes the learned reward. In contrast, DPO directly optimizes for the policy best satisfying the preferences with a simple classification objective, without an explicit reward function or RL.

Figure 1 of "Direct Preference Optimization: Your Language Model is Secretly a Reward Model", https://arxiv.org/abs/2305.18290

Direct Preference Optimization

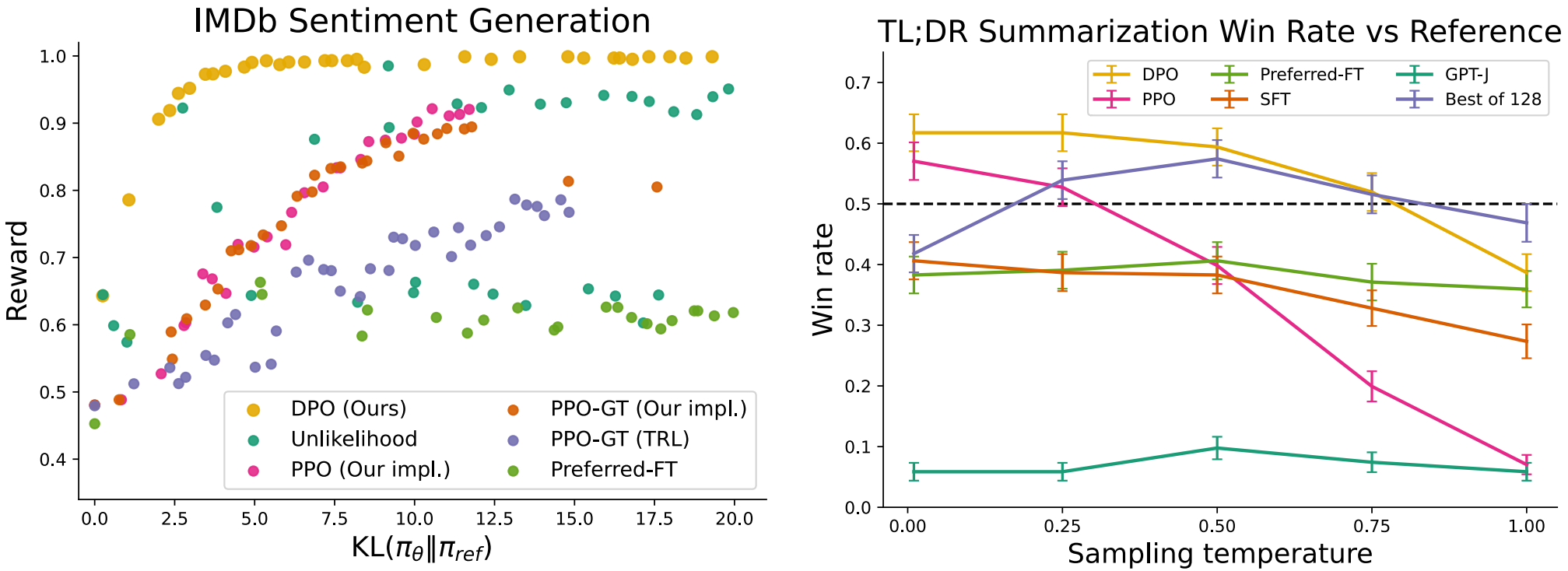
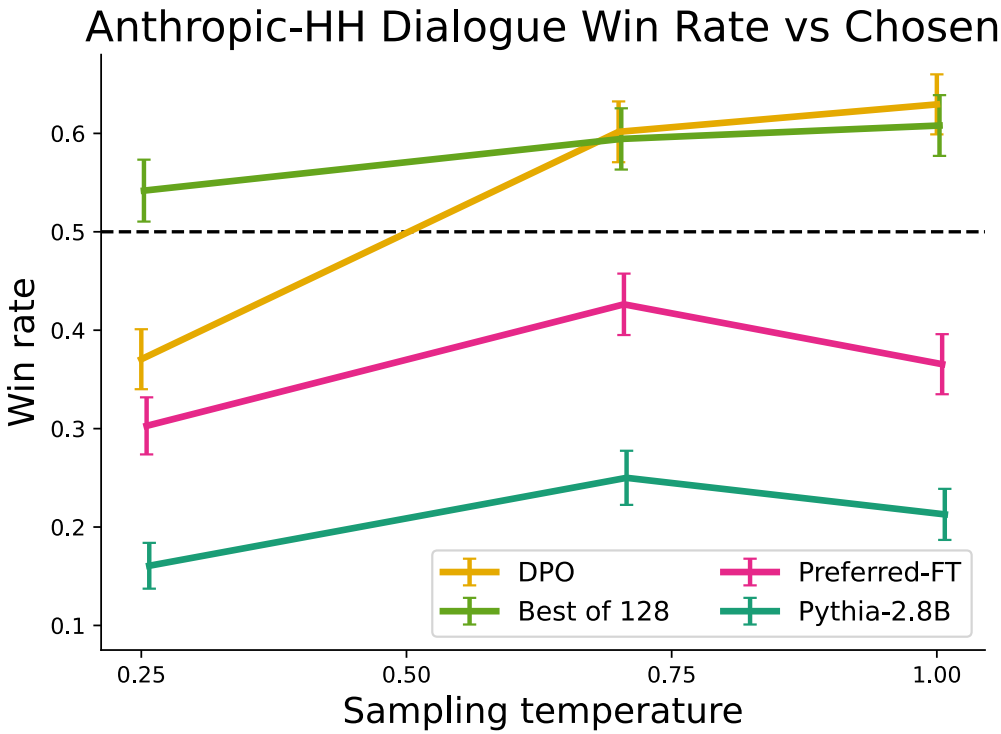


Figure 2: **Left.** The frontier of expected reward vs KL to the reference policy. DPO provides the highest expected reward for all KL values, demonstrating the quality of the optimization. **Right.** TL;DR summarization win rates vs. human-written summaries, using GPT-4 as evaluator. DPO exceeds PPO's best-case performance on summarization, while being more robust to changes in the sampling temperature.

Figure 2 of "Direct Preference Optimization: Your Language Model is Secretly a Reward Model", https://arxiv.org/abs/2305.18290

Direct Preference Optimization



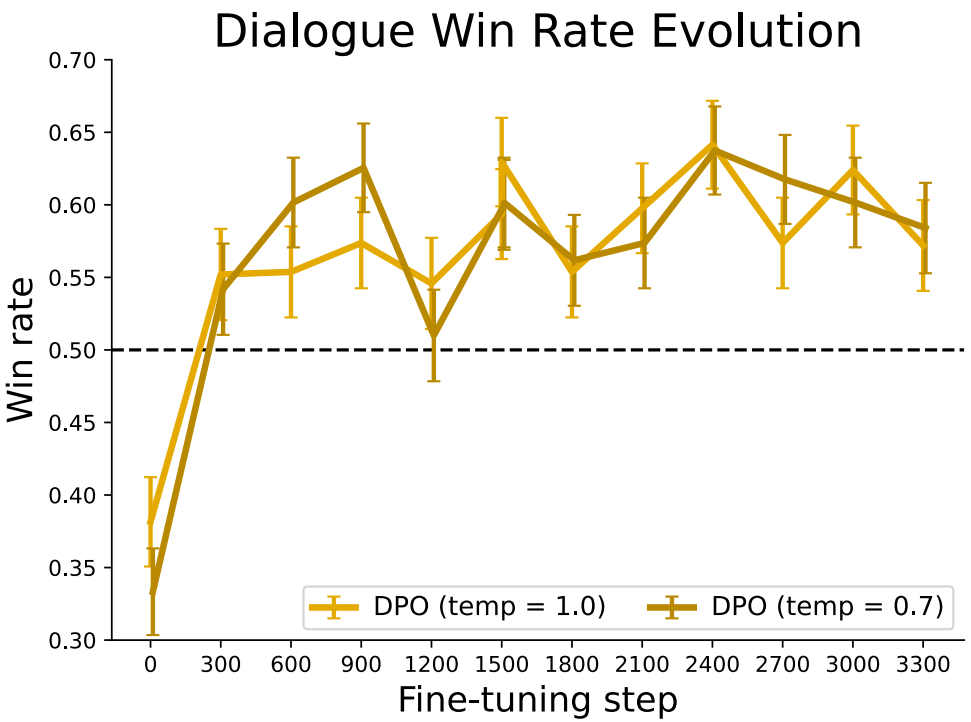


Figure 3: **Left.** Win rates computed by GPT-4 for Anthropic-HH one-step dialogue; DPO is the only method that improves over chosen summaries in the Anthropic-HH test set. **Right.** Win rates for different sampling temperatures over the course of training. DPO's improvement over the dataset labels is fairly stable over the course of training for different sampling temperatures.

Figure 3 of "Direct Preference Optimization: Your Language Model is Secretly a Reward Model", https://arxiv.org/abs/2305.18290