Learning Agile Soccer Skills for a Bipedal Robot with Deep Reinforcement Learning

Tuomas Haarnoja[*,1], Ben Moran[*,1], Guy Lever[*,1], Sandy H. Huang[*,1], Dhruva Tirumala[1], Markus Wulfmeier[1], Jan Humplik[1], Saran Tunyasuvunakool[1], Noah Y. Siegel[1], Roland Hafner[1], Michael Bloesch[1], Kristian Hartikainen[2,4], Arunkumar Byravan[1], Leonard Hasenclever[1], Yuval Tassa[1], Fereshteh Sadeghi[3,4], Nathan Batchelor[1], Federico Casarini[1], Stefano Saliceti[1], Charles Game[1], Neil Sreendra, Kushal Patel, Marlon Gwira, Andrea Huber[1], Nicole Hurley[1], Francesco Nori[1], Raia Hadsell[1] and Nicolas Heess[1]
[*]Equal contributions, [1]DeepMind, [2]University of Oxford, [3]Google, [4]Work done at DeepMind

We investigate whether Deep Reinforcement Learning (Deep RL) is able to synthesize sophisticated and safe movement skills for a low-cost, miniature humanoid robot that can be composed into complex behavioral strategies in dynamic environments. We used Deep RL to train a humanoid robot with 20 actuated joints to play a simplified one-versus-one (1v1) soccer game. We first trained individual skills in isolation and then composed those skills end-to-end in a self-play setting. The resulting policy exhibits robust and dynamic movement skills such as rapid fall recovery, walking, turning, kicking and more; and transitions between them in a smooth, stable, and efficient manner—well beyond what is intuitively expected from the robot. The agents also developed a basic strategic understanding of the game, and learned, for instance, to anticipate ball movements and to block opponent shots. The full range of behaviors emerged from a small set of simple rewards. Our agents were trained in simulation and transferred to real robots zero-shot. We found that a combination of sufficiently high-frequency control, targeted dynamics randomization, and perturbations during training in simulation enabled good-quality transfer, despite significant unmodeled effects and variations across robot instances. Although the robots are inherently fragile, minor hardware modifications together with basic regularization of the behavior during training led the robots to learn safe and effective movements while still performing in a dynamic and agile way. Indeed, even though the agents were optimized for scoring, in experiments they walked 156 % faster, took 63 % less time to get up, and kicked 24 % faster than a scripted baseline, while efficiently combining the skills to achieve the longer term objectives. Examples of the emergent behaviors and full 1v1 matches are available on the supplementary website: https://sites.google.com/view/op3-soccer.

1. Introduction

Creating general *embodied intelligence*, that is creating agents that can act in the physical world with agility, dexterity, and understanding—as animals or humans do—is one of the long-standing goals of AI researchers and roboticists alike. Animals and humans are not just masters of their bodies, able to perform and combine complex movements fluently and effortlessly, but they also perceive and understand their environment and use their bodies to effect complex outcomes in the world.

Attempts at creating intelligent embodied agents with sophisticated motor capabilities go back many years, both in simulation (Sims, 1994) and in the real world (Kuindersma et al., 2016; Raibert, 1986). But progress has accelerated considerably in recent years, and learning-based approaches have contributed substantially to this acceleration. For instance, deep reinforcement learning (deep RL) has proven capable of solving complex motor control problems for simulated characters (Bansal et al., 2018; Heess et al., 2017; Liu et al., 2022; Merel et al., 2020; Peng et al., 2018), including complex, perception-driven whole body control or multi-agent behaviors. By treating the dynamics

Corresponding author(s): tuomash@deepmind.com, guylever@deepmind.com

and physical constraints as a black box, and directly optimizing for the tasks or behaviors of interest, deep RL yields generality and scalability, and it can synthesize adaptive and long-horizon behaviors.

In recent years, deep RL has increasingly been applied to physical robots. In particular, high-quality quadrupedal legged robots have become widely available and have been the target of a number of demonstrations of how learning can generate a broad range of robust locomotion behaviors (Hwangbo et al., 2019; Lee et al., 2020; Peng et al., 2020; Siekmann et al., 2021b; Xie et al., 2019). Locomotion in static environments accounts for just a subset of the many ways in which animals and humans can deploy their bodies to interact with the world. This has been recognized in a number of works that have studied whole-body control and mobile manipulation, especially with quadruped robots. Examples include climbing (Rudin et al., 2022a), soccer skills like dribbling or catching (Bohez et al., 2022; Huang et al., 2022; Ji et al., 2022, 2023), and the use of legs for simple manipulation (Cheng et al., 2023). While much work has focused on quadrupeds, which are inherently stable, a smaller number of works have tackled locomotion and other movements for bipeds and humanoids, which pose additional challenges especially around stability and safety (Bloesch et al., 2022; Li et al., 2023; Siekmann et al., 2021b; Yu et al., 2019). These examples are encouraging, yet creating sophisticated long-horizon, multi-skill behaviors which can be composed, adapt to different environment contexts, and are safe to be executed on real robots, remains a challenging problem due to the difficulties of reward specification and the need to balance conflicting objectives to achieve not only dynamic and agile but also safe movements.

Sports like soccer showcase many of the hallmarks of human sensorimotor intelligence. In its full complexity soccer requires a diverse set of highly agile and dynamic movements, include running, turning, side stepping, kicking, passing, fall recovery, object interaction, and many more, which need to be composed in diverse ways. Players further need to be able to make predictions about the ball, teammates, and opponents, and adapt their movements to the game context. Players also need to coordinate movements over long timescales to achieve tactical, coordinated play. This diversity of challenges has been recognized in the robotics and artificial intelligence communities, especially through the RoboCup (Kitano et al., 1997; RoboCup Federation, 2022) competition. The agile, flexible, and reactive behaviors—and smooth transitions between them—required to play soccer well are challenging and time consuming to manually design for a robot.

In this work, we study whole-body control and object interaction of small humanoids in dynamic multi-agent environments. We consider a subset of the full soccer problem and train a low-cost, miniature humanoid robot with 20 controllable joints to play a simplified one-vs-one (1v1) game of soccer, using proprioception and game state features as observations. With the in-built controllers, the robot moves slowly and clumsily. However, we used deep RL to synthesize dynamic and agile context-adaptive movement skills such as walking, running, turning, kicking and fall recovery that are composed by the agent in a natural and fluent manner into complex, long-horizon behavior. In experiments, our agent learned to anticipate ball movements, position itself to block the opponent's shots, and take advantage of rebounds. The behavior of the agent emerged through a combination of skill reuse and end-to-end training with simple rewards in a multi-agent setting. We trained the agents in simulation and transferred them to the robot, demonstrating that sim-to-real transfer is possible even for low-cost robots. Example game-play can be seen at the supplementary website.[1]

2. Experimental Setup

In this section, we introduce both the simulated and the real soccer environments that we used to train the agent, as well as the robot hardware.

[1] https://sites.google.com/view/op3-soccer

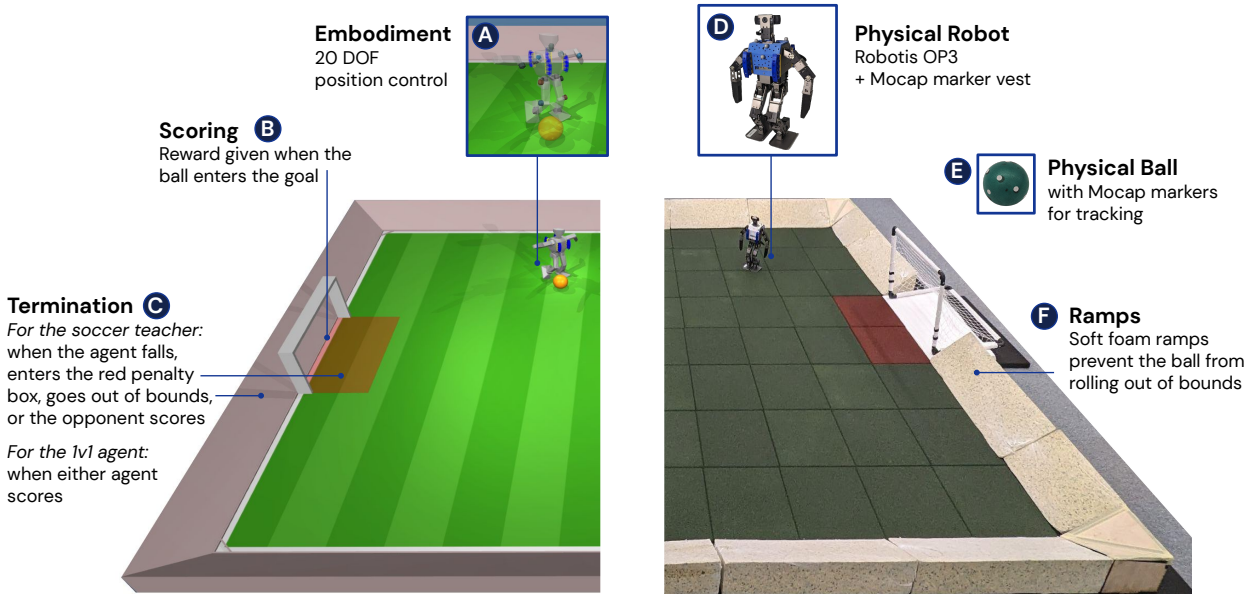


Figure 1 | We created matching simulated (left) and real (right) soccer environments. The pitch is 5 m long by 4 m wide. The real environment was also equipped with a motion capture (mocap) system for tracking the two robots and the ball.

2.1. Environment

We trained agents in simulation in a custom soccer environment and then transferred the policies to a corresponding real environment, as shown in Figure 1. The simulation environment uses the MuJoCo physics engine (Todorov et al., 2012) and is based on DeepMind Control Suite (Tunyasuvunakool et al., 2020). The environment consists of a soccer pitch that is 5 m long by 4 m wide, and two goals that each have an opening width of 0.8 m. In both the simulated and real environments, the pitch is bordered by ramps, which ensures that the ball stays in bounds. The real pitch is covered with rubber floor tiles to reduce the risk of falls damaging the robots and to increase the ground friction.

The aim in the 1v1 soccer task is to score a goal while preventing the opponent from scoring. In simulation, the agent is rewarded for scoring a goal when the center of the ball enters the goal. The episode terminates once a goal is scored or after 50 seconds. In simulation, we randomly reset the opponent's and the ball's location and orientation.

The agent's action is 20-dimensional and corresponds to the joint position set points of the OP3; actions are clipped to a manually selected range as given in Appendix A.1. The agent acts at 40 Hz. The agent's actions are passed through an exponential action filter to remove high frequency components: $\boldsymbol{u}_t = 0.8\boldsymbol{u}_{t-1} + 0.2\boldsymbol{a}_t$, where $\boldsymbol{u}_t$ is the filtered control applied to the robot at time step t, and $\boldsymbol{a}_t$ is the action output by the policy. The filtered actions are fed to PID controllers that then drive the joints (torques in simulation and voltages on the real robot) to attain the desired positions.

The agent's observations consist of proprioception and game state information. The proprioception consists of joint positions, IMU readings (i.e., linear acceleration, angular velocity, and gravity direction), and the state of the exponential action filter. The game state information, obtained via a motion capture setup in the real environment, consists of the agent's velocity, ball location and velocity, opponent location and velocity, and location of the two goals, which enables the agent to infer its global position. The locations are given as two-dimensional vectors corresponding to horizontal coordinates in the egocentric frame, and the velocities are obtained via finite differentiation from

the positions. All proprioceptive observations, as well as the observation of the agent's velocity, are stacked over the five most recent timesteps to account for delays and potentially noisy or missing observations. A more detailed description of the observations is given in Table 1.

	Observation	Dimension	Note
Proprioception	*joint positions*	$5 \cdot 20$	Joint positions in radians (stacked last 5 timesteps)
	linear acceleration	$5 \cdot 3$	Linear acceleration from IMU (stacked)
	angular velocity	$5 \cdot 3$	Angular velocity (roll, pitch, yaw) from IMU (stacked)
	gravity	$5 \cdot 3$	Gravity direction, derived from angular velocity using Madgwick filter (stacked)
	previous action	$5 \cdot 20$	Action filter state (stacked)
Game State	*ball position*	2	All positions and velocities are 2-dimensional vectors corresponding to the planar coordinates expressed in the agent's frame. The velocities are derived from positions via finite differentiation. Agent velocity is stacked over five timesteps.
	opponent position	2	
	goal position	2	
	opponent goal position	2	
	agent velocity	$5 \cdot 2$	
	ball velocity	2	
	opponent velocity	2	

Table 1 | The agent's observations.

2.2. Robot Hardware and Motion Capture

The Robotis OP3 (Robotis, 2023) (Figure 2) is a low-cost battery-powered, miniature humanoid platform used by robot researchers (Masuda and Takahashi, 2022). It is 51 cm tall, weighs 3.5 kg, and is actuated by 20 Robotis Dynamixel XM430-350-R servomotors. We controlled the servos by sending target angles, using position control mode with only proportional gain (i.e., no integral or derivative terms). Each actuator has a magnetic rotary encoder that provides the joint position observations to the agent. The robot also has an inertial measurement unit (IMU), which provides angular velocity and linear acceleration data. We found that the default robot control software was sometimes unreliable and caused indeterministic control latency. We thus wrote a custom driver that allows the agent to communicate directly and reliably with the servos and IMU via the Dynamixel SDK Python API. The control software runs on an embedded Intel Core i3 dual-core NUC, running Linux. The robot lacks GPUs or other dedicated accelerators, so all neural network computations were run on the CPU. The robot's "head" is a Logitech C920 web camera, which can optionally provide an RGB video stream at 30 frames per second.

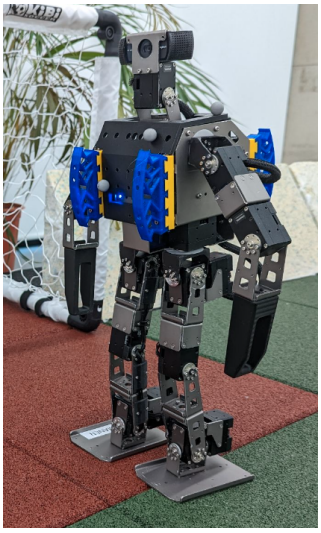


Figure 2 | One of the OP3 robots we used in our experiments.

The robot and ball positions and orientations were provided by a motion capture system based on Motive 2 software (Optitrack, 2023). This system uses 14 Optitrack PrimeX 22 Prime cameras mounted on a truss around the soccer pitch. We tracked the two robots using reflective passive markers attached to a 3D printed "vest" covering the robot torso. We also tracked the ball position using motion capture by attaching reflective stickers to the ball (Figure 1). The positions of these three objects were streamed over the wireless network using the VRPN protocol and made available to the robots via ROS.

We made small modifications to the robot to reduce damage from the evaluation of a wide range of prototype agents. We added 3D-printed safety bumpers at the front and rear of the torso, to reduce the impact of falls. We also replaced the original sheet metal forearms with 3D-printed alternatives, with the shape based on the convex hull of the original arms, because the original hook-shaped limbs would sometimes snag on the robot's own cabling. We made small mechanical modifications to the hip joints to spread the off-axis loads more evenly, to reduce the risk of fatigue breakages.

3. Method

Our aim is to train an agent that composes the wide range of skills required for soccer—including walking, kicking, getting up from the ground, scoring, and defending—into long-horizon strategic behavior, that we can then transfer to a real robot. These behaviors do not emerge if we simply train agents on a sparse reward for scoring goals, because of two main challenges: exploration and learning transferable behaviors. These are general challenges when applying deep RL to robotics.

We overcome this by splitting training into two stages, as depicted in Figure 3. In the first stage, we train *teacher policies* for two specific skills: getting up from the ground and scoring against an untrained opponent. When training the latter skill, the episode terminates whenever the agent is on the ground. Without this termination, agents find a local minimum and learn to roll on the ground towards the ball to knock it into the goal, rather than walking and kicking. In the second stage, the teacher policies from the first stage are used to regularize the agent, while it learns to play effectively against increasingly strong opponents. The latter enables the agent to learn tactics and strategy such as defending and anticipating the opponent's moves. We use a form of self-play, where the opponents are sampled from previous versions of the trained agent. This is a form of automatic curriculum learning (Portelas et al., 2020): the strength of the opponents increases as the agent itself improves, thus prompting further improvement. To facilitate transfer, domain randomization, random perturbations, sensor noise, and delays are incorporated in the training in simulation.

3.1. Overview

We modeled the soccer environment described in Section 2.1 as a γ-discounted Partially Observable Markov Decision Process (POMDP) defined by a tuple $(\mathcal{S}, \mathcal{A}, P, r, p_0, \gamma)$, and solved it using deep RL techniques (Sutton and Barto, 2018). The state space $\mathcal{S}$ consists of each robot's global location, orientation, joint angles, and joint velocities, as well as the ball's location and velocity[2]. At each timestep t, a soccer player observes features $\boldsymbol{o}_t \triangleq \phi(\boldsymbol{s}_t)$, extracted from the state $\boldsymbol{s}_t \in \mathcal{S}$ as listed in Table 1. The player then selects a 20-dimensional continuous action $\boldsymbol{a}_t \in \mathcal{A}$, corresponding to the desired positions of the robot's joints. Actions are sampled from the player's (stochastic) *policy*, $\boldsymbol{a}_t \sim \pi(\cdot|\boldsymbol{h}_t)$, as a function of the observation-action history $\boldsymbol{h}_t \triangleq (\boldsymbol{o}_{t-H}, \boldsymbol{a}_{t-H}, \boldsymbol{o}_{t-H+1}, \ldots, \boldsymbol{o}_t)$, where H is the history length. The observation history is needed to compensate for the partial observability of the environment, such as the feedback latency and model variations used for domain randomization (see Section 3.3.2), the full opponent state (the agent does not have access to the opponent's joint angles), and the opponent's policy. After filtering (see Section 2.1), the action is executed in the environment, a new state is sampled according to the system dynamics $\boldsymbol{s}_{t+1} \sim P(\cdot|\boldsymbol{s}_t, \boldsymbol{a}_t)$, and the player receives a reward as a function of the new state, $r_{t+1} \triangleq r(\boldsymbol{s}_{t+1})$. The choice of reward function depends on the stage of training, and will be discussed in detail later in this section, but is generally expressed as a weighted sum of K reward components: $r(\boldsymbol{s}_t) = \sum_{k=1}^{K} \alpha_k \hat{r}_k(\boldsymbol{s}_t)$. These interactions give rise to a trajectory $\xi = \{(\boldsymbol{s}_t, \boldsymbol{a}_t, r_{t+1})\}_{t=0}^{T}$, over horizon length T. A policy π, along with the

[2]The get-up task has an additional task observation that is treated in the same way as the others and is discussed in detail in Section 3.2.1. For convenience, we overload the notation and use the same symbols to denote both cases.

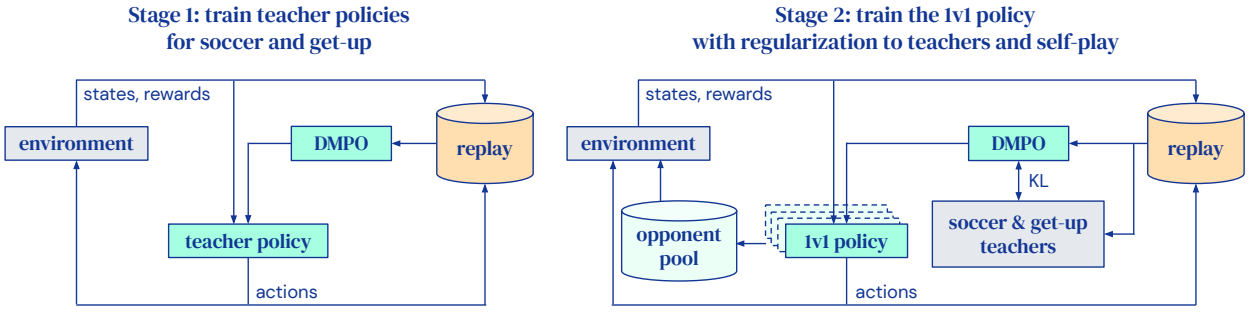


Figure 3 | We trained soccer agents in two stages. In the first stage (left), we train a separate soccer teacher and get-up teacher (Section 3.2.1). In the second stage (right), we distill these two teachers into a single agent that can both get up from the ground and play soccer (Section 3.2.2). The second stage also incorporates self-play: the opponent is uniformly randomly sampled from a pool that consists of policy snapshots from earlier in training. We found that this two-stage approach leads to qualitatively better behavior and improved sim-to-real transfer, compared to training an agent from scratch for the 1v1 soccer task.

system dynamics P and initial state distribution p_0, gives rise to a distribution over trajectories where $p(\xi) = p_0(\boldsymbol{s}_0) \prod_{t=0}^{T} \pi(\boldsymbol{a}_t|\boldsymbol{s}_t) P(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$.

The objective for the policy π is to maximize the discounted cumulative reward, taken in expectation over the induced trajectory distribution:

$$\mathcal{J}(\pi) \triangleq \mathbb{E}_{\xi}\left[\sum_{t=0}^{T} \gamma^t r(\boldsymbol{s}_t)\right]. \tag{1}$$

We optimized (1) using Maximum a Posteriori Policy Optimization (Abdolmaleki et al., 2018) with a distributional critic (DMPO) (Bellemare et al., 2017). We parameterized the policy as a deep feed-forward neural network with parameters θ. In order to better handle partial observability, we provided a history of length $H = 5$ to the agent for a subset of the observations, as specified in Table 1. The DMPO agent outputs the mean and diagonal covariance of a multivariate Gaussian, and the critic outputs a categorical distribution over Q-values. Details of the DMPO algorithm and the agent architecture are given in Appendix A.2.

Note that the trajectory distribution in the objective (1) depends on the particular choice of opponent. When training the 1v1 policy in the second stage, we sampled the opponent from a pool of previous snapshots of the agent, rendering the objective non-stationary. In practice though, we did not find this to hinder training, and the agent was able to learn and eventually converge to a well-performing policy. Similar approaches have been explored in prior work on multi-agent deep RL (Bansal et al., 2018; Heinrich et al., 2015; Lanctot et al., 2017).

3.2. Training

In this section, we describe the two-stage pipeline used for training agents in simulation. In the first stage, separate teacher policies for scoring goals and for getting up from the ground are trained. In the second stage, the teachers are distilled into a single 1v1 soccer agent, and the agent learns to interact with opponents through a form of self-play. Self-play also leads the agent to improve upon the robustness of the teacher policies, since it expands the set of environment states encountered by the agent.

3.2.1. Stage 1: Teacher Training

Soccer Teacher Training: The soccer teacher is trained to score as many goals as possible. Episodes terminate when the agent falls over, goes out of bounds, enters the goal penalty area (marked with red in Figure 1), or the opponent scores. At the start of each episode, the agent, the opponent, and the ball are initialized in random locations and orientations on the pitch. Both players are initialized in a default standing pose. The opponent is initialized with an untrained policy, which falls almost immediately and remains on the ground for the duration of each episode. Thus, the agent should learn to avoid the opponent at this stage, but no further complex opponent interactions would occur. We introduce shaping rewards that encourage ball interaction, to make exploration easier. We also include reward components to improve sim-to-real transfer and reduce robot breakages, as discussed in Section 3.3.3. The total reward is a weighted sum over reward components. The rewards and their weights for each training stage are listed in Table 2.

Type	**Reward** $\hat{r}_k$	**Description and purpose**	*Soccer teacher*	*Get-up teacher*	*1v1 policy*
Shaping	*Scoring*	+1 when the agent scores a goal and 0 otherwise.	1000	—	1000
	Conceding	−1 when the opponent scores a goal and 0 otherwise.	0	—	1000
	Velocity to ball	The magnitude of the agent's velocity toward the ball.	0.05	—	0.05
	Velocity	The magnitude of the player's forward velocity.	0.1	—	0.1
	Interference	A penalty, equal to the cosine of the angle between the player's velocity vector and the heading of the opponent, if the player is within 1 m of the opponent. This discourages the agents from interfering with and fouling the opponent.	1	—	1
	Termination	A penalty, equal to −1 if the player is on the ground, out of bounds, or in the goal penalty area. This encourages agents to get up from the ground quickly, avoid falling, and stay in bounds.	—	—	0.5
Sim-to-real	*Upright*	0 if the robot is upside down or if the tilt angle is greater than 0.4 radians. Increases linearly, and is equal to +1 if the tilt angle is less than 0.2 radians.	0.015	—	0.02
	Joint torque	A penalty, equal to the magnitude of the torque measured at the player's knees. This discourages the player from learning gaits which cause high forces on the knees, for example during ground impacts, which can damage a physical robot.	0.01	—	0.01
Teacher Training	*Target pose*	The negative of the product of *joint angle error* and *gravity error*. Joint angle error is the L2 norm of the difference of target joint angles and actual joint angles, scaled to $[0, 1]$, where 0 denotes an error of π and 1 denotes no error. Similarly, gravity error is the angle between the target gravity vector and the actual gravity vector, scaled to $[0, 1]$, where 0 denotes an error of $\pi/2$ and 1 denotes full alignment.	—	1	—

Table 2 | The reward components used for training the teachers and the 1v1 policy.

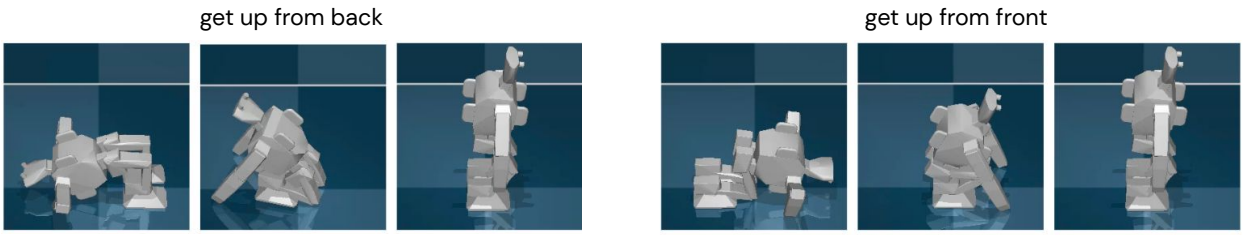


Figure 4 | The joint key poses used to train the get-up teacher, extracted from a scripted get-up controller (Robotis, 2023).

Get-Up Teacher Training: We found that training a separate teacher to learn a get-up skill resulted in qualitatively better final agent behavior and more robust transfer. The get-up teacher is trained using a sequence of target poses, to bias the policy towards a stable and collision-free trajectory. We used the pre-programmed get-up skill (Robotis, 2023) to extract three key poses for getting up from either the front or the back, as shown in Figure 4. Simply interpolating between the key poses enables a robot to get up in a stable but somewhat clumsy way. We instead use the key poses to guide learning, but do not constrain the final behavior.

The get-up teacher is trained to reach any target pose interpolated between the key poses. To that end, we introduce an additional task variable for the target pose, consisting of target joint angles $\mathbf{p}_{\text{target}}$ and target torso orientation[3] $\mathbf{g}_{\text{target}}$, and condition both the actor and the critic on this. The robot is initialized on the ground, and a new target pose is sampled on average every 1.5 seconds[4]. The agent is trained to maximize $\hat{r}_{\text{pose}}(\boldsymbol{s}_t) = -\tilde{p}_t \tilde{g}_t$, where $\tilde{p}_t = \left(\pi - \|\mathbf{p}_{\text{target}} - \mathbf{p}_t\|_2\right)/\pi$ is the scaled error in joint positions, and $\tilde{g}_t = \left(\pi - \arccos(\mathbf{g}_t^\intercal \mathbf{g}_{\text{target}})\right)/\pi$ is the scaled angle between the desired and actual gravity direction. $\mathbf{p}_t$ and $\mathbf{g}_t$ are the actual joint positions and gravity direction at timestep t, respectively. During training, the target poses are sampled uniformly at random. In the experiments, first the agent learned to reach the easiest of the targets, which are toward the beginning of the get-up sequences, and gradually progressed towards the more challenging poses toward the end of the sequences. We terminated training when the agent was able to reach all target poses robustly from an arbitrary initial state. Conditioning the converged policy on the last key pose (Figure 4) makes the agent stand up; we reused this as a get-up teacher in the next stage of training.

3.2.2. Stage 2: Distillation and Self-Play

In the second stage, the agent competes against increasingly stronger opponents, while regularizing its behavior to the teacher policies. This resulted in a single agent that is capable of a range of soccer skills: walking, kicking, getting up from the ground, scoring, and defending. The setup is the same as for training the soccer teacher, except now episodes terminate only when either the agent or the opponent scores. When the agent is on the ground, out of bounds, or in the goal penalty area, it receives a fixed penalty per timestep and all positive reward components are ignored. For instance, if the agent is on the ground when a goal is scored, then it receives a zero for the scoring reward component. At the beginning of an episode, the agent is initialized either laying on the ground on its front, on its back, or in a default standing pose, with equal probability.

[3]The target torso orientation is expressed as the gravity direction in the egocentric frame, which is independent of the robot yaw angle (heading), since that is irrelevant for the get-up task.

[4]The sampling intervals are exponentially distributed, to make the probability of a target pose switch independent of time, in order to preserve the Markov property.

Distillation: Given the soccer and get-up teachers, the agent must learn to transition smoothly between these two skills and improve upon their robustness. Since the teacher policies are trained in the same environment for the same embodiment, we can use policy distillation (Parisotto et al., 2015; Rusu et al., 2015), by adding a regularization term to encourage the output of the agent's policy to be similar to that of the teachers'. This approach is related to prior work that regularizes a student policy to a common shared policy across tasks (Teh et al., 2017) or a default policy that receives limited state information (Galashov et al., 2019), as well as work on kickstarting (Schmitt et al., 2018) and reusing learned skills for humanoid soccer in simulation (Liu et al., 2022).

Unlike most prior work, in our setting the teacher policies are useful in mutually exclusive sets of states: the get-up teacher is only useful in states where the agent is on the ground, and the soccer teacher is only useful in states where the agent is upright. Thus the agent should only be regularized to one teacher at a time, depending on which state it is in. Based on this domain-specific knowledge, we split the training objective into two components—one that combines the soccer training objective $\mathcal{J}(\pi_\theta)$ with KL regularization to the soccer teacher π_f and one that combines it with KL regularization to the get-up teacher π_g:

$$\mathcal{J}_f(\pi_\theta) \triangleq (1-\lambda_f)\,\mathcal{J}(\pi_\theta) - \lambda_f \mathbb{E}_\xi\Big[KL\big(\pi_\theta(\cdot|\mathbf{s})\|\pi_f(\cdot|\mathbf{s})\big)\Big] \tag{2}$$

$$\mathcal{J}_g(\pi_\theta) \triangleq (1-\lambda_g)\,\mathcal{J}(\pi_\theta) - \lambda_g \mathbb{E}_\xi\Big[KL\big(\pi_\theta(\cdot|\mathbf{s})\|\pi_g(\cdot|\mathbf{s})\big)\Big]. \tag{3}$$

In states where the agent is upright, the training objective $\mathcal{J}_f(\pi_\theta)$ applies; in all other states, the training objective $\mathcal{J}_g(\pi_\theta)$ applies. Thus the combined objective is

$$\mathbb{E}_\xi\Big[\mathbb{1}[\mathbf{s}\in\mathcal{U}]\mathcal{J}_f(\pi_\theta) + \mathbb{1}[\mathbf{s}\notin\mathcal{U}]\mathcal{J}_g(\pi_\theta)\Big], \tag{4}$$

where $\mathcal{U}$ is the set of all states in which the agent is upright.

To enable the agent to outperform the teachers, the weights λ_f and λ_g are adaptively adjusted such that there is no regularization once the agent's performance (as estimated by the critic's output) is above chosen thresholds Q_f and Q_g, respectively. This approach was proposed in (Abdolmaleki et al., 2021) for a similar setting. Specifically, λ_f is updated by stochastic gradient descent to minimize

$$c(\lambda_f) = \lambda_f\big(\mathbb{E}_\xi\big[Q^{\pi_\theta}(\mathbf{s},\mathbf{a})\big] - Q_f\big), \tag{5}$$

using a softplus transform and clipping to enforce that $0 \le \lambda_f \le 1$. (The analogous scheme is used to update λ_g.) When the agent's predicted return is less than Q_f, then λ_f increases to 1, at which point the agent effectively performs behavioral cloning to the soccer teacher. Once the agent's predicted return surpasses Q_f, then λ_f decreases to 0, at which point the agent learns using pure RL on the soccer training objective. This enabled the agent to improve beyond any simple scheduling of the teacher policies; our agents learned effective transitions between the two teacher skills and finetuned the skills themselves.

Self-Play: The performance and learned strategy of an agent depends on which opponents it is trained against. The soccer teacher is trained against an untrained opponent, which falls immediately and stays on the ground. An alternative approach is to train the agent against a copy of itself as the opponent. However, Bansal et al. (2018) found that this led to unstable learning, whereas uniformly randomly sampling the opponent per episode from all previously trained policies led to improved performance. We applied their approach in this work. In the second stage of training, snapshots of the agent's policy are regularly saved. For each episode the opponent is uniformly randomly sampled

from a pool of policies that consists of an untrained agent and the first quarter of the saved snapshots. We found that using the first quarter, rather than all snapshots, improved stability of training, by ensuring that the performance of the opponent improves over time but does not catch up too quickly to that of the agent. In our experiments, this training led to agents that were agile and defended against the opponent scoring. This version of self-play is inspired by prior work, for instance Neural Fictitious Self-Play and PSRO (Heinrich et al., 2015; Lanctot et al., 2017).

Playing against a mixture of opponents results in significant partial observability due to aliasing with respect to the opponent in each episode. As noted by Liu et al. (2019), this can cause significant problems for critic learning, since value functions fundamentally depend upon the opponent's strategy and ability. To address this, we condition the critic on an integer identification of the opponent.

3.3. Sim-to-Real Transfer

Our approach relies on zero-shot transfer of trained policies to the real robot. This section details the approaches we took to maximize the success of zero-shot transfer: we reduced the sim-to-real gap via simple system identification, improved the robustness of our policies via domain randomization and perturbations during training, and included shaping reward terms to obtain behavior that is less likely to damage the robot.

3.3.1. System Identification

We identified the actuator parameters by applying a sinusoidal control signal of varying frequencies to a motor with a known load attached to it, and optimized over the actuator model parameters in simulation to match the resulting joint angle trajectory. For simplicity, we chose a position controlled actuator model with torque feedback and with only damping (1.084 Nm/(rad/s)), armature (0.045 kg m^2), friction (0.03), maximum torque (4.1 N m), and proportional gain (21.1 N/rad) as free parameters. The values in parentheses correspond to the final values after applying this process. The model does not exactly correspond to the servos' operating mode, which controls the coil voltage instead of output torque, but we found our simplified model to match the training data sufficiently well. We believe the good match is the result of using a position control mode that hides model mismatch from the agent by applying fast stabilizing feedback at a high frequency. Indeed, we also experimented with direct current control, but found the sim-to-real gap was too large, which caused zero-shot transfer to fail. We expect that the sim-to-real gap could be further reduced by considering a more accurate model.

3.3.2. Domain Randomization and Perturbations

To further improve transfer to a robot, we applied domain randomization and random perturbations during training. Domain randomization helps overcome the remaining sim-to-real gap and the variation in dynamics across robots, due to wear and other factors such as battery state. We selected a small number of axes to vary, since excess randomization could result in a conservative policy, which would reduce the overall performance. Specifically, we randomized the floor friction (0.5 to 1.0), joint angular offsets (±2.9°), and varied the orientation (up to 2°) and position (up to 5 mm) of the IMU, and attached a random external mass (up to 0.5 kg) to a randomly chosen location on the robot torso. We also added random time delays (10 ms to 50 ms) to the observations to emulate latency in the control loop. The values were resampled in the beginning of each episode, and then kept constant for the whole episode. In addition to domain randomization, we found that applying random perturbations to the robot during training substantially improved the robustness of the agent, leading to better transfer. Specifically, we applied an external impulse force 5 N m to 15 N m lasting

for 0.05 s to 0.15 s to a randomly selected point on the torso every 1 s to 3 s.

3.3.3. Regularization for Safe Behaviors

We limited the range of possible actions for each joint to allow sufficient range of motion while minimizing the risk of self-collisions (see Table 4 in Appendix A.1). We also included two shaping reward terms to improve sim-to-real transfer and reduce robot breakages (see Table 2). In particular, we found that highly dynamic gaits and kicks often led to excessive stress on the knee joints from the impacts between the feet and the ground or the ball, which caused the gears to break. We mitigated this by regularizing the policies via a penalty term to minimize the time integral of torque peaks (thresholded above 5 N m) as calculated by MuJoCo for the constraint forces on the targeted joints. In addition to the knee breakages, we noticed the agent would often lean forward when walking. This made the gait faster and more dynamic, but when transferred to a real robot, it would often cause the robot to loose balance and fall forward. As a fix, we added a reward term for keeping an upright pose within the threshold of 11.5°. Incorporating these two reward components led to robust policies for transfer that rarely broke knee gears, and performed almost as well at scoring goals and defending against the opponent.

4. Results

We trained agents to play one-vs-one soccer, which we analyze in this section. Training the get-up and soccer teachers took 14 and 158 hours (6.5 days), respectively, and distillation and self-play took 68 hours (see Appendix B for details). We evaluated the agent in a physical 1v1 soccer match on real robots, and analyzed the emergent behaviors (Section 4.1). Then we isolated some of the behaviors (walking, getting up, and kicking) and compared them to corresponding scripted baseline controllers (Section 4.2). To assess reliability and gauge any gap in performance between simulation and reality we also investigated selected set pieces, where the agent was tasked to score from various configurations (Section 4.3). To demonstrate the agent's awareness of certain game features and their impact on the outcome of the game, we looked into the learned value function and studied its sensitivity to game features (Section 4.4). Finally, we ran ablations to investigate the effect of regularizing to teachers and incorporating self-play (Section 4.5).

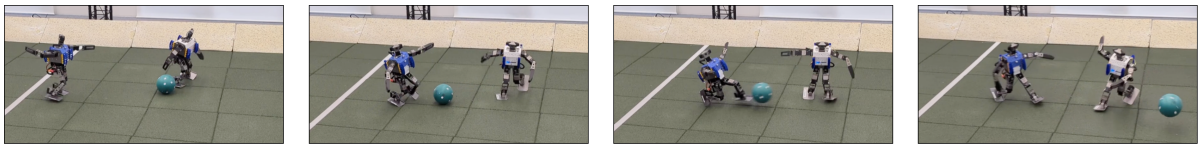
4.1. 1v1 Soccer Behaviors

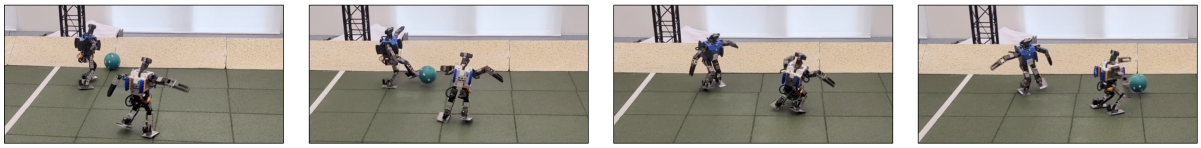
Selected extracts from 1v1 matches can be seen in Figure 5. The soccer agent exhibited a variety of emergent behaviors, including agile movement skills such as getting up from the ground, quick recovery from falls, running, and turning; object interaction such as ball control and shooting, kicking a moving ball, blocking shots; and strategical behaviors such as defending by consistently placing itself between the attacking opponent and its own goal, and protecting the ball with its body. During play the agents transitioned between all of these skills in a fluid way. We encourage the reader to view these behaviors, as well as full 1v1 matches on the supplementary website.[5]

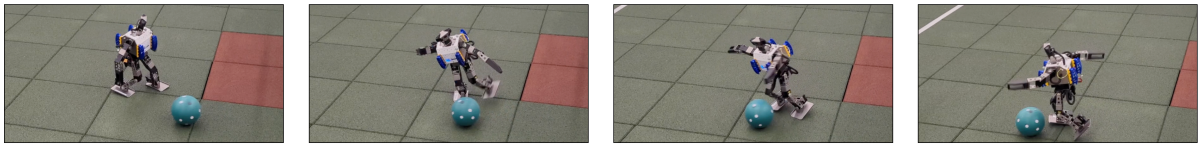
4.2. Comparison to Scripted Baseline Skills

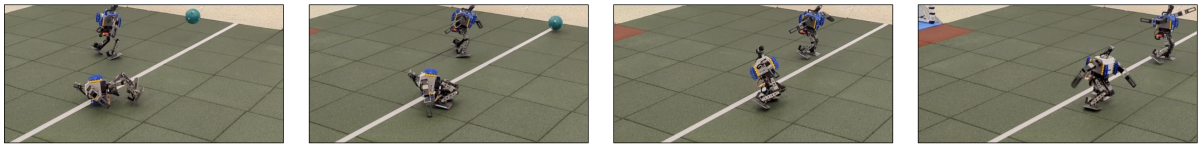
Some key locomotion skills, including getting up, kicking, and walking, are available for the OP3 (Robotis, 2023), and we used these as baseline behaviors. To measure how well learned policies perform at specific key skills, we compared our final 1v1 policy with the scripted controllers, both quantitatively and qualitatively.

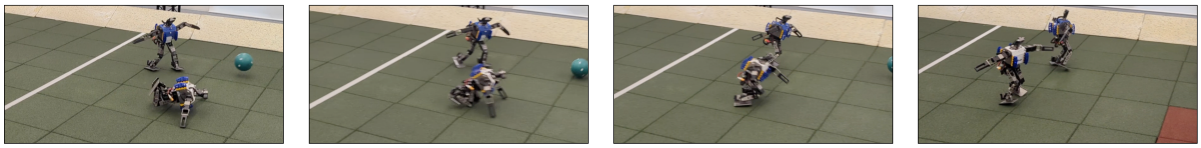
[5] https://sites.google.com/view/op3-soccer

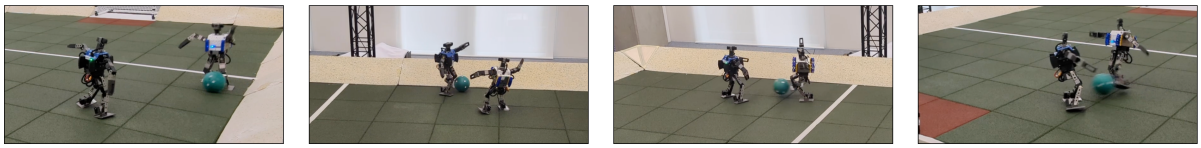












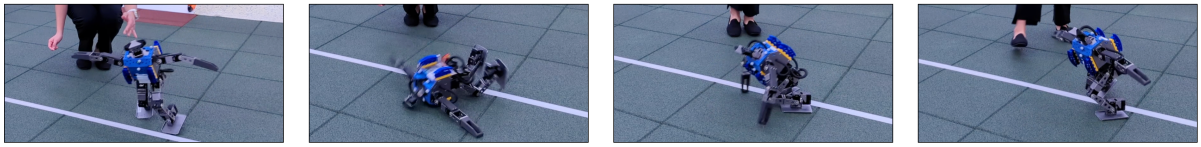


Figure 5 | Gallery of behaviors on the robot.

4.2.1. Quantitative Analysis

Walking Speed: To compare walking speed, we ran 10 episodes with both the learned policy and the scripted walking controller. For each episode, we initialized the robot in the standard "walking ready" pose, and ran both the policy or controller for 10 seconds. Since the scripted controller does not actively correct the robot's heading, it often quickly veered off course. Therefore, to make the comparison favorable for the baseline, we measured the maximum speed in any horizontal direction over any one second window within the episode. No direct penalty was applied for falling, but any window in which the robot fell[6] was excluded, since a robot could feasibly attain high instantaneous speed by diving forward and falling. The learned policy does not have a specific walking skill that can be executed in isolation, so we encouraged the policy to walk forwards by initializing the robot close to its own goal, and placing the ball in the opponent's goal area.

Get-Up Ability: Similar to walking, the learned policy does not have a specific get-up skill that can be executed in isolation. However, the agent prefers to stay upright, so simply initializing the robot on the floor triggers the get-up behavior. To compare the get-up skills, we thus placed the robot on the floor, face down in a T-pose, and activated either the learned soccer policy or the baseline get-up controller. We considered the robot as standing as soon as it reached a height of 36 cm, measured at the motion capture marker on the shoulder, and only counted those instances where the robot remained above that height for the following one second, to discount instances where the robot got up quickly but subsequently stumbled.[7] We collected 10 episodes from both skills and compared the time required to reach the specified get-up threshold. Both the scripted behavior and the learned policy succeeded in standing up on every trial, so it was not necessary to apply a penalty for failing.

Kicking Power: To compare kicking ability, we placed the ball 1.5 m from the opponent's goal and placed the robot behind the ball, facing the goal. For the scripted skill, which is "blind" in the sense that it does not take into account the ball's location, we were careful to place the robot such that the swung leg was exactly behind the ball, to maximize the impact. The learned policy is conditioned on the ball position, so we aligned the ball between the legs, allowing the agent choose which leg to use for the kick. We determined the kicking speed by calculating the total distance travelled (in any direction) in the 0.2 s window following the first contact with the ball. We collected 10 episodes from both behaviors. We observed that the learned policy can kick more powerfully if we initialize the ball further away from the robot, to allow it to take a few steps before the kick. Therefore, we collected 10 additional episodes with the ball 0.5 m from the goal and an approach distance of 2.5 m. This latter setup was not possible to replicate with the scripted controller, since it cannot take into account the ball's location.

Results are given in Table 3. The reinforcement learning policy performed better than the specialized manually-designed skills: it walked 156 % faster and took 63 % less time to get up. When initialized near the ball it kicked the ball with 5 % less speed; both achieved a ball speed of around 2 m/s. However, with an additional run-up approach to the ball, the learned policy's mean kicking speed was 2.6 m/s (24 % faster than the scripted skill) and the maximum kicking speed across episodes was 3.4 m/s. The reinforcement learning approach was able to learn highly optimized techniques, pushing the limits of the hardware, and its behaviors were consistent and reliable, with a failure rate of zero across the three comparison tests. Moreover, even though the robot outperformed the scripted

[6] We defined falling as the shoulder dipping below the height of 30 cm.

[7] When the robot is fully upright, the motion capture marker on the shoulder reaches a height of 41 cm, hence a threshold of 36 cm corresponds to approximately a 5cm tolerance in shoulder height. This is necessary since in general the learned policy, after getting up, quickly starts moving away from the standing position, towards the ball for example.

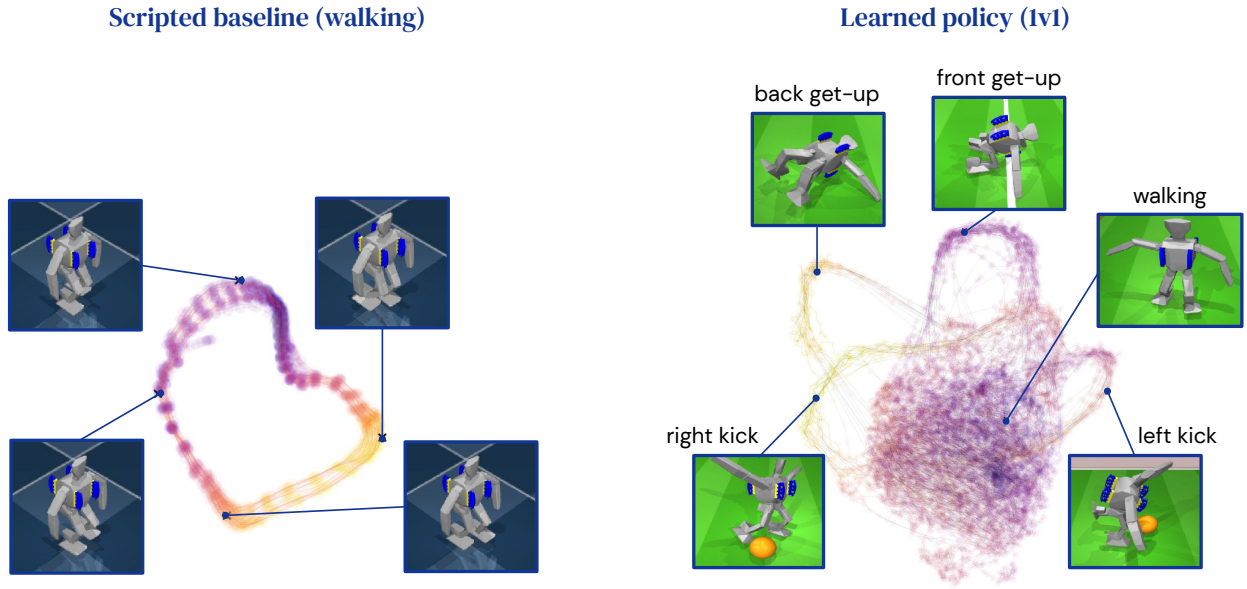


Figure 6 | Embedding of the joint angles recorded while executing either the scripted baseline walking policy (left) or the learned 1v1 policy (right).

get-up skill from an initialization on the ground, in practice the agent acts to prevent itself from falling to the ground: even when pushed the robot will quickly react to counter the fall and thus recovers more quickly than the results here indicate. See the supplementary project website and Figure 5 for an example of the quick recovery during gameplay and from adversarial pushes. Furthermore, more complex behaviors, and combinations of behaviors such as turning to kick a moving ball at a target, are more challenging to manually design, but the learning approach was able to learn highly optimized techniques for those more complex combinations of behaviors.

4.2.2. Skill Embeddings

To understand the locomotion of the robot under different policies, we took inspiration from the analysis of Drosophila motion in (DeAngelis et al., 2019), which views a gait as a motion through 20-dimensional joint space. Trajectories produced by a given policy will then describe a manifold embedded in this space. We used Uniform Manifold Approximation and Projection (UMAP) (McInnes et al., 2018) to approximate this manifold in a three-dimensional space to visualize the trajectories.

The scripted walking gait is based on periodic sinusoidal motions of the end effectors, and so the gait traced a cyclic path through this space. This topological structure can be seen in the 3-dimensional UMAP embedding, with the angular coordinate around the circle defined by the phase within the periodic gait (Figure 6, left).

In contrast, the trajectories resulting from 1v1 play by the learned policy show much richer structure (Figure 6, right). The additional variations in pose result in the walking motion being embedded as a dense ball, rather than a circle. We also see four topologically distinct loops. By visualizing trajectory clips sampled along each of these loops, we can see that two correspond to getting up (from the front and back), and two correspond to kicking (with the left and right feet).

This perspective therefore offers another way to see how the learned policy is able to fluidly blend both pre-trained skills (such as the get-up motions) and emergent skills (such as the kicks and walking gaits) into continuous agile behavior.

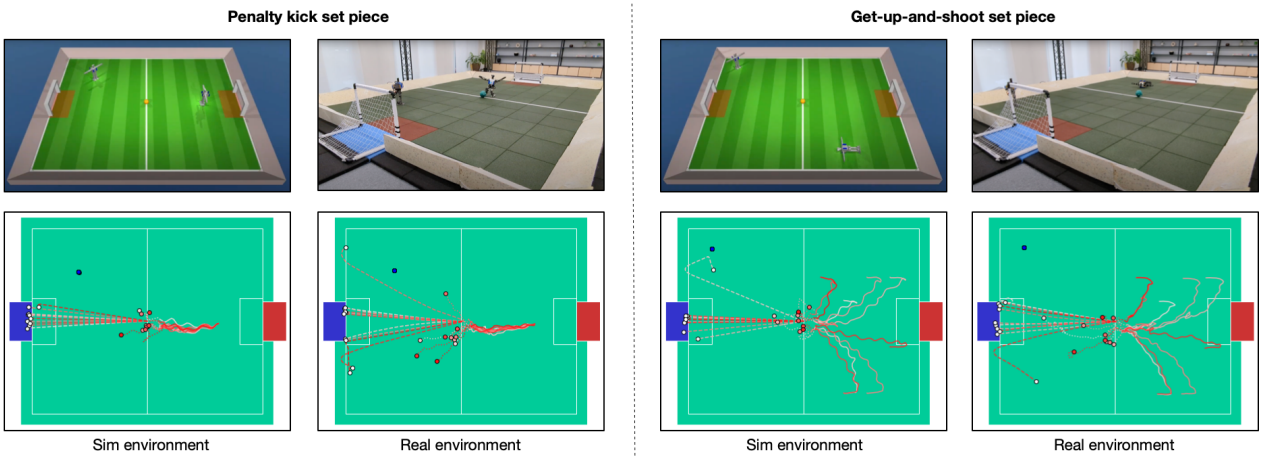


Figure 7 | Top row: Example frames or initializations for the set-piece tasks in sim and on the real robot. Bottom row: Overlayed plots of the trajectories collected from the set-piece experiment. The plots show the robot trajectory before kicking (solid red lines) and after kicking (dotted red line), the ball trajectory (dashed red line), and the final ball position (white circle) and opponent position (blue circle). Behaviors in the set pieces in simulation were reliable and consistent: the robot and ball trajectories clearly overlap and the agent scores nearly every time. In contrast, the behaviors in the real environment were somewhat less consistent—the ball trajectories vary more and the robot misses the goal slightly more often.

4.3. Set Piece Analysis

To gauge the reliability of the learned policies, we designed *penalty kick* and *get-up-and-shoot* set pieces, implemented in both the simulation (training) environment and the real environment. The penalty kick set piece is a short episode of 1v1 soccer in which the ball is initialized at the centre of the pitch, and the agent is initialized in a default T-pose[8], offset 1.5 m behind the ball closer to its own goal. The opponent is initialized in a T-pose 1 m away from the goal line and the sideline, in its own half, and remains stationary throughout the episode. The episode ends after 6 seconds or after a goal is scored. The get-up-and-shoot set-piece is identical to the penalty kick, except the agent is initialized face down in a T-pose behind the ball at a random position[9], and the agent has 10 seconds in which to score a goal. The initial configurations are shown in Figure 7.

In the real environment the robot scored 7 out of 10 goals (70 %) in the *penalty kick* task and 8 out of 10 (80 %) goals in the *get-up-and-shoot* task, and was able to get up from the ground and kick the ball every time. In simulation the agent scored more consistently in both tasks, and the time taken to make contact with the ball was slightly lower, with lower variance. This indicates a slight drop in performance and increased variance in behavior due to transfer to the real environment (including the real robot, ball, floor surface etc.), but the robot was still able to reliably get up, kick the ball, and score. Results are given in Figure 7 and Table 3, and example episodes are included in the supplementary website: `https://sites.google.com/view/op3-soccer`.

[8]T-pose refers to the joint angle configuration where all joint positions are set to zero.

[9]The initial position of the robot was sampled uniformly from a grid of positions in the center of the robot's own half; the positions used in sim and real were identical.

Set-piece performance	Scoring success rate Sim env.	Real env.	Mean time to first touch Sim env.	Real env.
Penalty kick	90%	70%	2.9s (0.16s)	3.2s (0.32s)
Get-up and score	90%	80%	4.2s (1.2s)	4.7s (1.4s)

Comparison to baseline skills	Max walking speed mean (std. dev.)	Get-up time mean (std. dev.)	Kicking speed mean (std. dev.)
Scripted baseline	0.27m/s (0.01m/s)	2.5s (0.02s)	2.1m/s (0.16m/s)
Learned policy	0.69m/s (0.04m/s)	0.93s (0.39s)	2.0m/s (0.81m/s)
With run up	-	-	2.6m/s (0.86m/s)

Table 3 | Performance and reliability in the set pieces and comparison to the baseline skills. Values in braces are unbiased estimates of the standard deviation. The learned policy's mean kicking power was roughly equivalent to the scripted behavior from a standing pose, but with an additional run up approach to the ball the learned policy achieved a more powerful kick (bottom row).

4.4. Value Function Analysis

We investigated the learned value function in order to verify directly that the agent is sensitive to the observations of the ball, goal, and the opponent. We created a synthetic observation with the robot located centrally on the court at coordinates (0,0), facing toward the desired goal with the ball 10cm in front of it at location (0.1, 0). The opponent was placed to one side, at location (0, 1.5). In this setting, we plotted the expected value of the critic network's predicted reward distribution, as a function of the ball's velocity vector. We found that the critic learned to assign high value to ball velocities directed toward the goal Figure 8, and also to ball velocities consistent with keeping the ball in the area under the agent's control.

A similar analysis shows the agent also learned that the opponent impedes scoring, as shown in Figure 8. This plot uses the same positions for the agent, goals and ball, but now gives the expected value as a function of the opponent position in egocentric coordinates. States in which the opponent is positioned between the ball and the goal have a much lower value, as expected.

4.4.1. Adaptive footwork

To demonstrate the efficiency and fluidity of the discovered gait, we analyzed the footstep pattern of the agent in a *turn-and-kick* set piece. In this task the agent is initialized near the sideline and facing parallel with the sideline, with the ball in the center. The natural strategy for the agent to score is therefore to turn to face the ball, then walk around the ball and turn again to kick, in a roughly mirrored 'S' pattern. As can be seen in Figure 9 the agent achieved this with only 10 footsteps. Turning, walking and kicking were seamlessly combined, and the agent adapted its footwork by taking a single penultimate shorter step to position itself to turn and kick.

4.5. Ablations

We ran ablations to investigate the contribution of each component of our pipeline for training soccer agents. We focused on the importance of regularization to teachers and using self-play.

4.5.1. Without Regularization to Teachers

We tried training agents directly in the 1v1 task, without regularizing to teacher policies for get-up and soccer. We experimented with either using a sparse reward or using the same shaped reward as

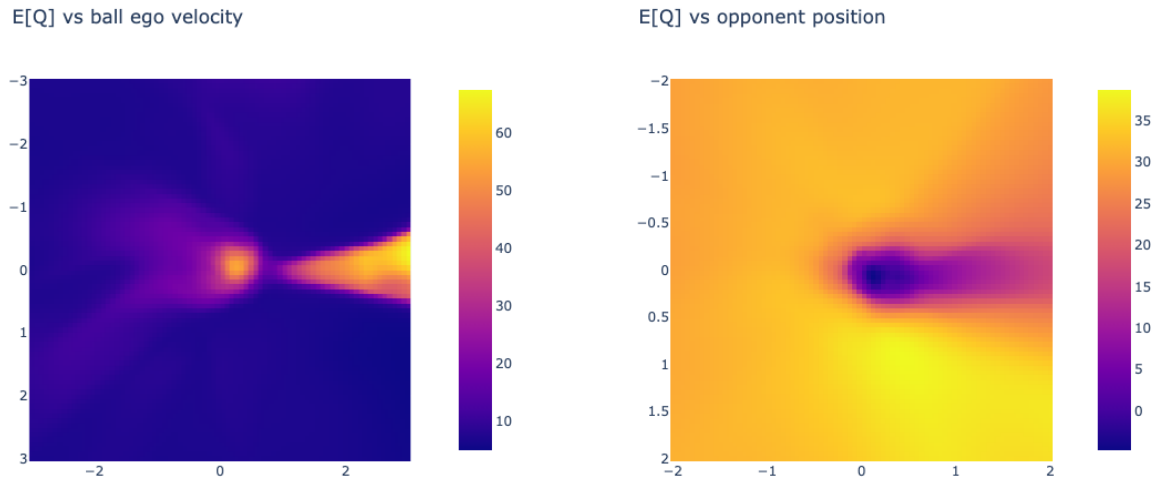


Figure 8 | Left: Heat map of the critic's expected return vs the ball (x,y) velocity in m/s. High value states are those when the ball is either travelling toward the goal, or remaining near the agent. Right: Heat map of the critic's expected return vs the opponent (x,y) position in metres relative to the agent. The predicted value is lower when the opponent is located between the ball and the target goal. (Observation coordinates fixed as described in text.)

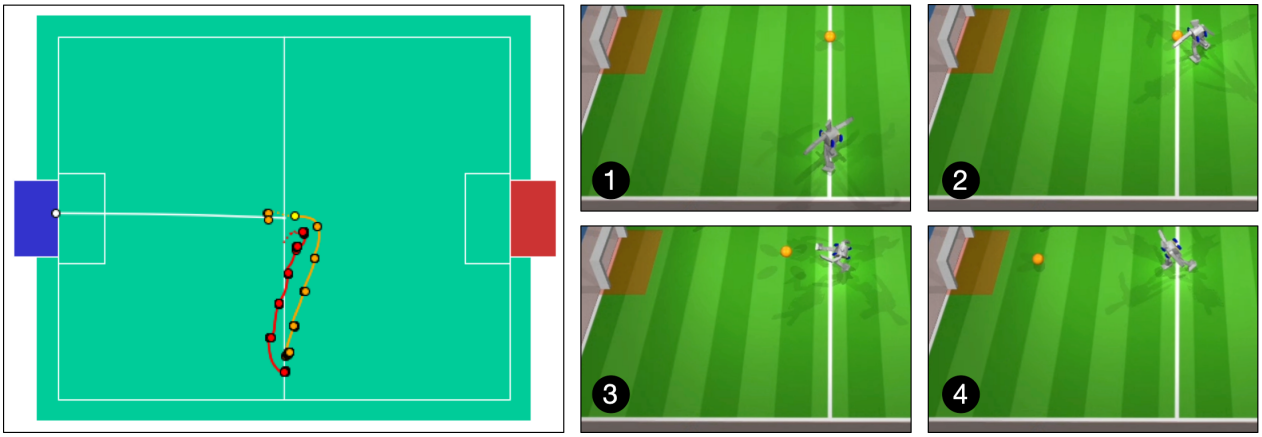


Figure 9 | Right: a sequence of frames from the *turn-and-kick* set piece. Left: a plot of the footsteps from the corresponding trajectory showing: right foot trajectory (orange), left foot trajectory (red), ball trajectory (white), point of the kick (yellow), and footsteps highlighted with dots. The agent turned, walked approximately 2 m, turned, kicked, and finally balanced using 10 footsteps.

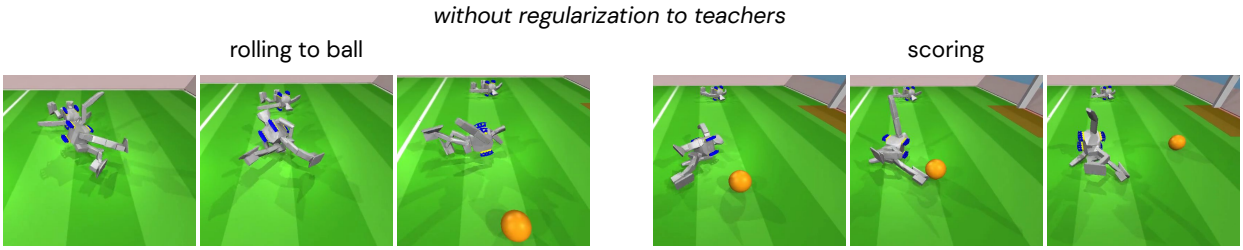


Figure 10 | When trained without regularization to get-up and soccer teachers, agents learn to score by rolling to the ball and knocking it into the goal.

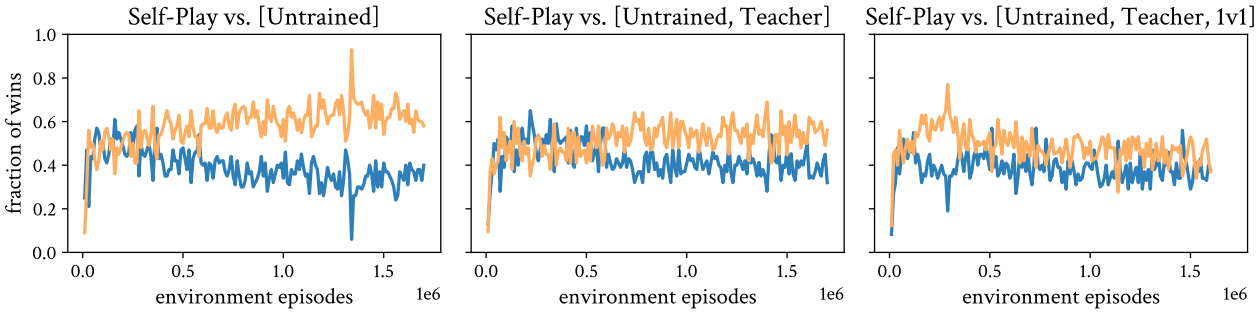


Figure 11 | Comparison of our 1v1 agent trained with self-play versus baselines trained against a fixed set of opponents: an untrained agent (left); an untrained agent and soccer teacher (middle); or an untrained agent, soccer teacher, and the 1v1 agent (right). We play agents head-to-head over the course of training, and plot the win fraction across 100 matches for the self-play agent (orange) and the non-self-play baseline agent (blue). The first agent to score in the episode wins the match; if no agent scores within 50 seconds, then it is a tie.

for the 1v1 policy (see Table 2).

Sparse Reward: When we trained agents with a sparse reward, given for scoring or conceding goals, agents learned a local optimum of rolling to the ball and knocking it into the goal with a leg (Figure 10).

Shaped Reward: Including the termination punishment reward component from the 1v1 task prevents the agent from finding this local optimum: if the agent is on the ground, it receives a small negative penalty (of −0.5) and does not receive any positive rewards from other reward components. When we used the same shaped reward as for the 1v1 policy, including the termination punishment, then the agent did not learn to score, walk, or even get up. When the agent was initialized in a standing position, it stayed standing still; when it was initialized on the ground, it flailed around and was unable to get up. If we instead left out the termination punishment from the shaped reward, we saw the same behavior as with the sparse reward, where agents learn to score effectively without getting up.

4.5.2. Without Self-Play

We also investigated the importance of using self-play to train agents in the second stage. We trained agents in the same way as for the 1v1 policy, regularizing to the same get-up and soccer teachers and with the same shaped reward. We only varied which opponents the agents played against: only

the untrained agent (that the soccer teacher is trained against); the untrained agent and the soccer teacher (that cannot get up if it falls); or the untrained agent, the soccer teacher, and the 1v1 policy trained with self-play. We played agents head-to-head after regular intervals of training, and plot the win fraction of each agent across 100 matches in Figure 11. The agent trained with self-play outperforms agents trained against the untrained agent and/or soccer teacher, with a larger difference later on in training. Even when the self-play agent is included as an opponent during training, the resulting agent does not consistently win against it (Figure 11, right). Note that training against the self-play agent as an opponent cannot be implemented in practice, without the use of self-play.

5. Related Work

5.1. Robot Learning

Legged robots have seen a surge in research interest in recent years. This interest has been fueled in equal parts by a proliferation of hardware platforms suitable for research, and the realization that locomotion behaviors can be effectively created by deep RL techniques (e.g. Heess et al., 2017). While there have been some attempts at training legged robots to walk with deep RL directly on hardware (Bloesch et al., 2022; Ha et al., 2021; Haarnoja et al., 2019; Smith et al., 2022; Wu et al., 2023), the vast majority of works (Bohez et al., 2022; Hwangbo et al., 2019; Kumar et al., 2021; Lee et al., 2020; Li et al., 2023; Peng et al., 2020; Radosavovic et al., 2023; Rudin et al., 2022a; Smith et al., 2023) rely on some form of sim-to-real transfer, thus sidestepping many of the safety and data efficiency concerns associated with training directly on hardware. A common theme is that a surprisingly small number of techniques can be sufficient to reduce the sim-to-real gap, in part thanks to the relatively high quality of modern legged hardware. Recent work has demonstrated the ability to traverse a variety of complex terrains (Agarwal et al., 2023; Choi et al., 2023; Lee et al., 2020; Miki et al., 2022), blind and purely guided by onboard perception.

Both the ready availability of quadrupedal platforms and their relative stability and safety compared to bipeds have made them the primary target of locomotion research. However, bipedal hardware has also been improving, and recent works have produced behaviors including walking and running (Choi et al., 2023; Robotics, 2022), stair climbing (Siekmann et al., 2021b), and jumping (Li et al., 2023). Albeit very robust, these behaviors are relatively conservative, presumably due to safety concerns. Most recent works have focused on high-quality, full-sized bipeds and humanoids, with a much smaller number (Bloesch et al., 2022; Masuda and Takahashi, 2022; Nachum et al., 2019; Yu et al., 2019) targeting more basic platforms whose simpler and less precise actuators and sensors pose additional challenges in terms of sim-to-real transfer.

Whether bipedal or quadrupedal, navigation represents only a fraction of animal and human capabilities. Motivated by this observation, there is a growing interest in whole body control, i.e. tasks in which the whole body is used in flexible ways to interact with the environment. Examples include climbing (Rudin et al., 2022a), getting-up from the ground (Ma et al., 2023), catching objects (Ma et al., 2023), and mobile manipulation with legs (Cheng et al., 2023). Recently, reinforcement learning has been applied to learn simple soccer skills, including goalkeeping (Huang et al., 2022), ball manipulation on diverse terrains (Bohez et al., 2022; Ji et al., 2023), and shooting (Ji et al., 2022). These works focus on a narrower set of skills than the 1v1 soccer game, and the quadrupedal platform is inherently more stable and therefore presents an easier learning challenge.

RL research has been facilitated by the availability of high-performant simulation environments (Rudin et al., 2022b). However, a common challenge is the design of reward functions that lead to safe, functional, and dynamic behaviors that take full advantage of the hardware capabilities. Even the creation of basic gaits can require significant ingenuity (Lee et al., 2020; Margolis and Agrawal,

2023; Peng et al., 2020; Siekmann et al., 2021a), and the synthesis of more complex movements that can be easily repurposed and composed remains a challenge. This question has been studied to a much wider extent in simulation, for instance in the literature on character animation, where complex movement skills have been demonstrated for simulated humanoid and other characters, often using motion capture data as a starting point (Merel et al., 2020).

5.2. Skill and Transfer Learning

Skill learning and transfer of learned behaviors has been a long-standing active area of research (Bowling and Veloso, 1998; MacAlpine and Stone, 2018; Stone, 2000; Thrun and Schwartz, 1994). The data requirements for training neural networks as policies have recently emphasized the requirement for increased data efficiency. Different mechanisms for transfer have been proposed, ranging from direct reuse of parameters in flat or hierarchical agents (Parisotto et al., 2015; Peng et al., 2019; Salter et al., 2022; Sutton et al., 1999; Won et al., 2021; Wulfmeier et al., 2021), over auxiliary objectives (Liu et al., 2022; Ross et al., 2011; Tirumala et al., 2022), to transfer via a skill's generated experience (Riedmiller et al., 2018; Vezzani et al., 2022). A related line of work is kickstarting, which makes use of a trained teacher policy to enable a student policy to learn more quickly and obtain better performance, on the same task (Abdolmaleki et al., 2021; Galashov et al., 2019; Schmitt et al., 2018; Team et al., 2023). These approaches transfer knowledge from the teacher to the student via a distillation loss, defined as either the cross-entropy or KL-divergence between the output of the student and teacher networks (Parisotto et al., 2015; Rusu et al., 2015). Abdolmaleki et al. (2021) frames kickstarting as a multi-objective problem, that must trade off between regularization to the teacher policy versus the RL objective of maximizing expected return. Our approach builds on the linear scalarization (or weighted-sum) approach to combining reward and kickstarting proposed in Abdolmaleki et al. (2021), and we follow their proposal to adjust the level of regularization to the teacher depending on the student's performance. We extend this approach to work in our setting, where we have more than one teacher policy. In particular on real robotics platforms, skill transfer has been critical for increased data efficiency (Hafner et al., 2021; Wulfmeier et al., 2019).

5.3. Multi-agent Reinforcement Learning

Our approach to training against a mixture of previous opponents is motivated by established methods for multi-agent training. Reinforcement learning with pure self-play can exhibit unstable or cyclic behavior, since learning focuses on the exploitation of a single current policy (e.g. Balduzzi et al., 2019), and by overfitting can become exploitable. Therefore many applications of reinforcement learning to multi-agent domains train against a mixture of opponents. In normal form games Fictitious Play (Brown, 1951)—in which successive best responses to the mixture of previous policies are computed—converges to a Nash equilibrium for two-player, zero-sum games, and it has been generalized to extensive form games using reinforcement learning by, for example, Heinrich et al. (2015); Lanctot et al. (2017). Alternatively, but similarly, Vinyals et al. (2019) achieved stability and robustness by playing against a league of opponents. Our work is an efficient implementation of these ideas since we train against a mixture of previous opponents, but using one continuous training epoch, rather than successive generations. This improves efficiency, but could carry an increased risk of getting stuck in a local optima. Orthogonally, self-play provides a natural auto-curriculum in multi-agent RL (Baker et al., 2020; Bansal et al., 2018), which can be important since finding the best response to a set of strong opponents from scratch can be challenging for RL in some domains; for example, in soccer, strong opponents could dominate play and a learner might get very little experience of ball interaction. Our method effectively features a similar auto-curriculum property, since we trained in one continuous epoch in which the opponents are initially weak, and, subsequently, are automatically

calibrated to the strength of the current agent as earlier agent checkpoints are successively added to the opponent pool.

5.4. RoboCup and Other Competitive Games

Robot soccer has been a longstanding grand challenge for AI and robotics, since at least the formation of the RoboCup competition Kitano et al. (1997); RoboCup Federation (2022) in 1996. One original aim of RoboCup, was to achieve human-level football with a team of autonomous robots by 2050. The majority of successful approaches to the real humanoid robot soccer competition do not learn all components of the system and often feature manually designed components or high-level strategies. However, historically, reinforcement learning has been applied to learning certain skills including running, kicking, dribbling, and stable locomotion (Abreu et al., 2019; Hausknecht and Stone, 2011; Riedmiller et al., 2008; Röfer et al., 2019; Saggar et al., 2007), or specific aspects or sub-tasks of the robot soccer problem (Kalyanakrishnan and Stone, 2010; Kalyanakrishnan et al., 2007; Riedmiller et al., 2009; Stone et al., 2005; Tuyls et al., 2002). Simulation grounding by sim-to-real transfer and has also been used to investigated in the context of skills (Farchy et al., 2013). One successful learning-based approach approach to the simulated 3d humanoid RoboCup league is Layered Learning MacAlpine and Stone (2018); Stone (2000). There reinforcement learning was used to train multiple skills, including ball control and pass selection, which were combined via a pre-defined hierarchy. Our system pre-defines fewer skills (leaving the agent to discover useful skills like kicking, if possible) and we focused on learning to combine the skills, seamlessly. Reinforcement learning based approaches have also been successful at the 2d simulation league (e.g. Riedmiller et al., 2000).

Many breakthroughs in artificial intelligence have been demonstrated on competitive games which have often been used as grand challenges since at least the 1950s (Campbell et al., 2002; Moravčík et al., 2017; Samuel, 1959; Silver et al., 2016; Tesauro, 1995; Vinyals et al., 2019). One recent line of work considers the emergence of complex behaviors arising from competition between agents, or the auto-curriculum of self play (Al-Shedivat et al., 2018; Baker et al., 2020; Bansal et al., 2018; Liu et al., 2022). Those breakthroughs involve simulated domains and simple embodiments or action spaces, whereas the current work focuses on a more complex embodiment and physical robot.

6. Discussion

6.1. Limitations

Our work provides a step towards practical use of deep RL for agile control of humanoid robots in a dynamic multi-agent setting. However, there are several limitations our work does not cover, or touches only superficially. First, the low-level behaviors necessarily compromise between stability and dynamism; we could potentially obtain behaviors that improve in both aspects by using insights from multi-objective RL. Second, we do not leverage real data for transfer; instead, our approach relies solely on sim-to-real transfer. Fine-tuning on real robots or mixing in real data during training in simulation could help improve transfer and enable an even wider spectrum of stable behaviors. Third, we chose position-based actions due to the simplicity and stabilizing effect of a fast position controller, but we believe that other control modes, such as direct current control (Pratap Singh et al., 2023) or dynamic control of the feedback parameters (Masuda and Takahashi, 2022) could lead to more human-like behavior, and improve perturbance rejection and safety. Fourth, we applied our method to a small robot and did not consider additional challenges that would be associated with a larger form factor.

6.2. Comparison to RoboCup

The inspiration for our 1v1 soccer task is RoboCup, where the OP3 has been used for the humanoid league. However, our environment and task are substantially simpler than the full RoboCup problem. The main differences are that we focused on teams of one player instead of multiple players (including a designated goalkeeper); our simulated and real environments do not align with the field or ball specifications; our environment does not follow the rules of RoboCup (e.g., the kick-off, player substitutions, and game length), and we use full state information rather than relying solely on vision—although in the following Section 6.3 we discuss preliminary vision-only results.

An exciting direction of future work would be to train teams of two or more agents. It is straightforward to apply our proposed method to train agents in this setting. In our preliminary experiments for 2v2 soccer, we saw that the agent quickly learns division of labor, i.e., a simple form of collaboration: if its teammate is closer to the ball, the agent stops and waits for the teammate to approach and kick the ball. However, the agents also learned more conservative and less agile behaviors, perhaps because it is more difficult to learn to avoid obstacles and anticipate the movement of the ball. Insights from related work on training teams of soccer players in simulation (Liu et al., 2022) could be applied to improve agent performance in this setting.

6.3. Playing Soccer from Raw Vision

The results in the main text rely on external state information from a motion capture system. Ultimately we want robots to play soccer anywhere, using on-board sensors only. As a first step, we investigated how to train vision-based agents that only use onboard RGB camera and proprioception.

Learning from visual information requires overcoming additional challenges. In comparison to state-based agents that have direct access to the ball, goal, and opponent locations from a motion capture system, vision-based agents need to infer the same information from a limited history of egocentric camera observations. The camera's field of view renders the environment only partially observable. The agent is now required to handle more complex, high-dimensional inputs and integrate partial state information over time, which makes the problem significantly harder (Tassa et al., 2018).

In addition, the direct use of raw vision data further exaggerates the gap between simulation and real robot environment. In order to bridge this gap, we created a visual rendering of our lab using a Neural Radiance Field (NeRF) model (Mildenhall et al., 2021) based on the approach introduced by Byravan et al. (2023). In line with progress in domain randomization (Tobin et al., 2017) for physical and visual properties, we randomized training across multiple collected NeRFs to improve the agent's real-world performance. For each individual NeRF, we combined the MuJoCo rendering of dynamic objects like the ball and opponent over the NeRF image during agent training. This superimposed image was then fed to an LSTM-based agent for learning (full details in Appendix C.1). We trained agents in the 1v1 setting and the reward and task setup from vision was otherwise identical to that used for training with state information.

The agent was then evaluated in the vision-based penalty kick set piece (see Figure 12 and Section 4.3 for further details), the robot was able to score 6 out of 10 goals (60 %) in the real environment and hit the goal post 3 times (30 %). In addition, we found the agent could successfully integrate information over time to track a moving ball and position itself appropriately to score. These results provide an initial indication for the ability of deep RL systems to solve soccer from vision. We have added further videos on the project website to demonstrate both 1v1 soccer behavior as well as penalty kick performance. Further analysis and details can be found in Appendix C.1.

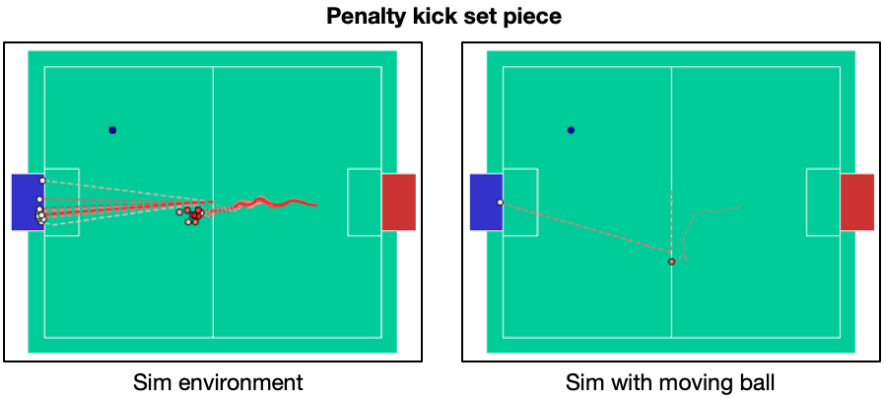


Figure 12 | Overlaid plots of the trajectories collected from the simulated vision set-piece experiment. We refer the reader to our supplementary website for videos of experiments on real robots. Robot trajectory before kick (solid red lines), after kick (dotted red line) and corresponding ball trajectory (dashed red line), ball (white circle) and opponent (blue circle). (Left) Analysis of penalty kick from fixed starting positions of ball and robots. The robot scored in all trials. (Right) A single trajectory showing adaptive movement behavior when the ball is initialized to be rolling along the negative y-axis.

7. Conclusion

We applied deep RL to low-cost humanoid robots and taught them to play 1v1 soccer. We trained an agent in simulation in two stages: In the first stage, we trained two independent skills, one for getting-up from the ground, and a second for scoring in the presence of a random opponent. In the second stage, we distilled the skills into a single soccer agent, and improved its game tactics via a form of self-play, where the agent was tasked to play against an early copy of itself. We then zero-shot transferred the agent to a real robot. We found simple system identification and light domain randomization combined with relatively strong random force perturbations during training to result in surprisingly effective transfer.

The agent learned dynamic skills that are beyond what one would intuitively expect from the hardware. The motions are dynamic and, in experiments, the emergent behaviors exceeded the performance of corresponding scripted baseline controllers by a large margin, while stitching the behaviors together seamlessly and effectively. The agent learned to utilize these low-level behaviors in order to optimize the high-level task of scoring goals, while stopping the opponent from scoring, as a single flat controller mapping from observations to motor commands. Even though the learned policies could be improved in terms of stability and perception, our results are encouraging, and we believe that similar methods could be applied to larger robots to solve practical real-world tasks.

Acknowledgements: We would like the thank Daniel Hennes at DeepMind for developing the plotting tools used for the soccer matches, and Martin Riedmiller and Michael Neunert at DeepMind for their helpful comments.

References

A. Abdolmaleki, J. T. Springenberg, Y. Tassa, R. Munos, N. Heess, and M. Riedmiller. Maximum a posteriori policy optimisation. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, 2018.

A. Abdolmaleki, S. H. Huang, G. Vezzani, B. Shahriari, J. T. Springenberg, S. Mishra, D. TB, A. Byravan, K. Bousmalis, A. Gyorgy, C. Szepesvari, R. Hadsell, N. Heess, and M. Riedmiller. On multi-objective policy optimization as a tool for reinforcement learning. *arXiv preprint arXiv:2106.08199*, 2021.

M. Abreu, L. P. Reis, and N. Lau. Learning to run faster in a humanoid robot soccer environment through reinforcement learning. In *Robot World Cup*, pages 3–15. Springer, 2019.

A. Agarwal, A. Kumar, J. Malik, and D. Pathak. Legged locomotion in challenging terrains using egocentric vision. In *Conference on Robot Learning*, pages 403–415. PMLR, 2023.

M. Al-Shedivat, T. Bansal, Y. Burda, I. Sutskever, I. Mordatch, and P. Abbeel. Continuous adaptation via meta-learning in nonstationary and competitive environments. In *International Conference on Learning Representations (ICLR)*, 2018.

B. Baker, I. Kanitscheider, T. Markov, Y. Wu, G. Powell, B. McGrew, and I. Mordatch. Emergent tool use from multi-agent autocurricula. In *8th International Conference on Learning Representations (ICLR), 2020*, 2020.

D. Balduzzi, M. Garnelo, Y. Bachrach, W. Czarnecki, J. Pérolat, M. Jaderberg, and T. Graepel. Open-ended learning in symmetric zero-sum games. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pages 434–443. PMLR, 2019.

T. Bansal, J. Pachocki, S. Sidor, I. Sutskever, and I. Mordatch. Emergent complexity via multi-agent competition. In *6th International Conference on Learning Representations (ICLR)*, 2018.

M. G. Bellemare, W. Dabney, and R. Munos. A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

M. Bloesch, J. Humplik, V. Patraucean, R. Hafner, T. Haarnoja, A. Byravan, N. Y. Siegel, S. Tunyasuvunakool, F. Casarini, N. Batchelor, et al. Towards real robot learning in the wild: A case study in bipedal locomotion. In *Conference on Robot Learning*, pages 1502–1511. PMLR, 2022.

S. Bohez, S. Tunyasuvunakool, P. Brakel, F. Sadeghi, L. Hasenclever, Y. Tassa, E. Parisotto, J. Humplik, T. Haarnoja, R. Hafner, et al. Imitate and repurpose: Learning reusable robot movement skills from human and animal behaviors. *arXiv preprint arXiv:2203.17138*, 2022.

M. Bowling and M. Veloso. Reusing learned policies between similar problems. In *Proceedings of the AI* AI-98 Workshop on New Trends in Robotics*. Citeseer, 1998.

G. W. Brown. Iterative solution of games by fictitious play. In T. C. Koopmans, editor, *Activity Analysis of Production and Allocation*. Wiley, New York, 1951.

A. Byravan, J. Humplik, L. Hasenclever, A. Brussee, F. Nori, T. Haarnoja, B. Moran, S. Bohez, F. Sadeghi, B. Vujatovic, and N. Heess. NeRF2Real: Sim2real transfer of vision-guided bipedal motion skills using neural radiance fields. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*. arXiv, 2023.

M. Campbell, A. J. H. Jr., and F. Hsu. Deep Blue. *Artif. Intell.*, 134(1-2):57–83, 2002.

X. Cheng, A. Kumar, and D. Pathak. Legs as manipulator: Pushing quadrupedal agility beyond locomotion. *arXiv preprint arXiv:2303.11330*, 2023.

S. Choi, G. Ji, J. Park, H. Kim, J. Mun, J. H. Lee, and J. Hwangbo. Learning quadrupedal locomotion on deformable terrain. *Science Robotics*, 8(74):eade2256, 2023.

B. D. DeAngelis, J. A. Zavatone-Veth, and D. A. Clark. The manifold structure of limb coordination in walking *Drosophila*. *eLife*, 8:e46409, jun 2019. ISSN 2050-084X. doi: 10.7554/eLife.46409.

A. Farchy, S. Barrett, P. MacAlpine, and P. Stone. Humanoid robots learning to walk faster: From the real world to simulation and back. In *Proc. of 12th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, May 2013.

A. Galashov, S. Jayakumar, L. Hasenclever, D. Tirumala, J. Schwarz, G. Desjardins, W. M. Czarnecki, Y. W. Teh, R. Pascanu, and N. Heess. Information asymmetry in KL-regularized RL. In *International Conference on Learning Representations*, 2019.

S. Ha, P. Xu, Z. Tan, S. Levine, and J. Tan. Learning to walk in the real world with minimal human effort. In *Conference on Robot Learning*, pages 1110–1120. PMLR, 2021.

T. Haarnoja, S. Ha, A. Zhou, J. Tan, G. Tucker, and S. Levine. Learning to walk via deep reinforcement learning. In *Proceedings of Robotics: Science and Systems (RSS)*, 2019.

R. Hafner, T. Hertweck, P. Klöppner, M. Bloesch, M. Neunert, M. Wulfmeier, S. Tunyasuvunakool, N. Heess, and M. Riedmiller. Towards general and autonomous learning of core skills: A case study in locomotion. In *Conference on Robot Learning*, pages 1084–1099. PMLR, 2021.

M. Hausknecht and P. Stone. Learning powerful kicks on the Aibo ERS-7: The quest for a striker. In J. R. del Solar, E. Chown, and P. G. Plöger, editors, *RoboCup-2010: Robot Soccer World Cup XIV*, volume 6556 of *Lecture Notes in Artificial Intelligence*, pages 254–65. Springer Verlag, Berlin, 2011.

N. Heess, D. Tirumala, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, A. Eslami, M. Riedmiller, et al. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017.

J. Heinrich, M. Lanctot, and D. Silver. Fictitious self-play in extensive-form games. In F. R. Bach and D. M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 805–813. JMLR.org, 2015.

X. Huang, Z. Li, Y. Xiang, Y. Ni, Y. Chi, Y. Li, L. Yang, X. B. Peng, and K. Sreenath. Creating a dynamic quadrupedal robotic goalkeeper with reinforcement learning. *CoRR*, abs/2210.04435, 2022.

J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26), 2019. doi: 10.1126/scirobotics.aau5872.

Y. Ji, Z. Li, Y. Sun, X. B. Peng, S. Levine, G. Berseth, and K. Sreenath. Hierarchical reinforcement learning for precise soccer shooting skills using a quadrupedal robot. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1479–1486. IEEE, 2022.

Y. Ji, G. B. Margolis, and P. Agrawal. DribbleBot: Dynamic legged manipulation in the wild. *arXiv preprint arXiv:2304.01159*, 2023.

S. Kalyanakrishnan and P. Stone. Learning complementary multiagent behaviors: A case study. In J. Baltes, M. G. Lagoudakis, T. Naruse, and S. S. Ghidary, editors, *RoboCup 2009: Robot Soccer World Cup XIII*, pages 153–165. Springer Verlag, 2010.

S. Kalyanakrishnan, Y. Liu, and P. Stone. Half field offense in RoboCup soccer: A multiagent reinforcement learning case study. In G. Lakemeyer, E. Sklar, D. Sorenti, and T. Takahashi, editors, *RoboCup-2006: Robot Soccer World Cup X*, volume 4434 of *Lecture Notes in Artificial Intelligence*, pages 72–85. Springer Verlag, Berlin, 2007. ISBN 978-3-540-74023-0.

H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa. RoboCup: The robot world cup initiative. In *Proceedings of the first international conference on Autonomous agents*, pages 340–347, 1997.

S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake. Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot. *Autonomous Robots*, 40(3):429–455, 2016.

A. Kumar, Z. Fu, D. Pathak, and J. Malik. RMA: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034*, 2021.

M. Lanctot, V. Zambaldi, A. Gruslys, A. Lazaridou, K. Tuyls, J. Pérolat, D. Silver, and T. Graepel. A unified game-theoretic approach to multiagent reinforcement learning. In *Advances in neural information processing systems*, pages 4190–4203, 2017.

J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.

Z. Li, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath. Robust and versatile bipedal jumping control through multi-task reinforcement learning. *CoRR*, abs/2302.09450, 2023. doi: 10.48550/arXiv.2302.09450.

S. Liu, G. Lever, J. Merel, S. Tunyasuvunakool, N. Heess, and T. Graepel. Emergent coordination through competition. *ICLR*, 2019.

S. Liu, G. Lever, Z. Wang, J. Merel, S. M. A. Eslami, D. Hennes, W. M. Czarnecki, Y. Tassa, S. Omidshafiei, A. Abdolmaleki, N. Y. Siegel, L. Hasenclever, L. Marris, S. Tunyasuvunakool, H. F. Song, M. Wulfmeier, P. Muller, T. Haarnoja, B. D. Tracey, K. Tuyls, T. Graepel, and N. Heess. From motor control to team play in simulated humanoid football. *Sci. Robotics*, 7(69), 2022.

Y. Ma, F. Farshidian, and M. Hutter. Learning arm-assisted fall damage reduction and recovery for legged mobile manipulators. *arXiv preprint arXiv:2303.05486*, 2023.

P. MacAlpine and P. Stone. Overlapping layered learning. *Artif. Intell.*, 254:21–43, 2018.

G. B. Margolis and P. Agrawal. Walk these ways: Tuning robot control for generalization with multiplicity of behavior. In *Conference on Robot Learning*, pages 22–31. PMLR, 2023.

S. Masuda and K. Takahashi. Sim-to-real learning of robust compliant bipedal locomotion on torque sensor-less gear-driven humanoid. *arXiv preprint arXiv:2204.03897*, 2022.

L. McInnes, J. Healy, and J. Melville. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *ArXiv e-prints*, Feb. 2018.

J. Merel, S. Tunyasuvunakool, A. Ahuja, Y. Tassa, L. Hasenclever, V. Pham, T. Erez, G. Wayne, and N. Heess. Catch & Carry: Reusable Neural Controllers for Vision-Guided Whole-Body Tasks. *ACM Transactions on Graphics (TOG)*, 39(4), 2020.

T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science Robotics*, 7(62):eabk2822, 2022.

B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

M. Moravčík, M. Schmid, N. Burch, V. Lisý, D. Morrill, N. Bard, T. Davis, K. Waugh, M. Johanson, and M. Bowling. Deepstack: Expert-level artificial intelligence in no-limit poker. *Science*, 356, 01 2017.

O. Nachum, M. Ahn, H. Ponte, S. Gu, and V. Kumar. Multi-agent manipulation via locomotion using hierarchical sim2real. *arXiv preprint arXiv:1908.05224*, 2019.

Optitrack. Motive optical motion capture software. https://optitrack.com/software/motive/, March 2023.

E. Parisotto, J. L. Ba, and R. Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv preprint arXiv:1511.06342*, 2015.

X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne. DeepMimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics (TOG)*, 37(4):143, 2018.

X. B. Peng, M. Chang, G. Zhang, P. Abbeel, and S. Levine. MCP: learning composable hierarchical control with multiplicative compositional policies. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, pages 3681–3692, 2019.

X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine. Learning agile robotic locomotion skills by imitating animals. *arXiv preprint arXiv:2004.00784*, 2020.

R. Portelas, C. Colas, L. Weng, K. Hofmann, and P.-Y. Oudeyer. Automatic curriculum learning for deep rl: A short survey. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 4819–4825, 7 2020.

R. Pratap Singh, Z. Xie, P. Gergondet, and F. Kanehiro. Learning bipedal walking for humanoids with current feedback. *arXiv e-prints*, pages arXiv–2303, 2023.

I. Radosavovic, T. Xiao, B. Zhang, T. Darrell, J. Malik, and K. Sreenath. Learning humanoid locomotion with transformers. *CoRR*, abs/2303.03381, 2023. doi: 10.48550/arXiv.2303.03381.

M. H. Raibert. *Legged robots that balance*. MIT press, 1986.

M. Riedmiller, A. Merke, D. Meier, A. Hoffmann, A. Sinner, O. Thate, and R. Ehrmann. Karlsruhe Brainstormers - a reinforcement learning approach to robotic soccer. In *RoboCup-2000: Robot Soccer World Cup IV, LNCS*, pages 367–372. Springer, 2000.

M. Riedmiller, T. Gabel, R. Hafner, and S. Lange. Reinforcement learning for robot soccer. *Autonomous Robots*, 27(1):55–73, 2009.

M. Riedmiller, R. Hafner, T. Lampe, M. Neunert, J. Degrave, T. van de Wiele, V. Mnih, N. Heess, and J. T. Springenberg. Learning by playing solving sparse reward tasks from scratch. In *Proceedings of the 35th International Conference on Machine Learning*, pages 4344–4353, 2018.

M. A. Riedmiller, R. Hafner, S. Lange, and M. Lauer. Learning to dribble on a real robot by success and failure. In *2008 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2207–2208. IEEE, 2008.

RoboCup Federation. Robocup project. https://www.robocup.org, May 2022.

A. Robotics. Cassie sets world record for 100m run, 2022. URL https://www.youtube.com/watch?v=DdojWYOKONc.

Robotis. Robotis OP3 source code. https://github.com/ROBOTIS-GIT/ROBOTIS-OP3, April 2023.

Robotis. Robotis OP3 manual. https://emanual.robotis.com/docs/en/platform/op3/introduction, March 2023.

T. Röfer, T. Laue, G. Felsch, A. Hasselbring, T. Haß, J. Oppermann, P. Reichenberg, and N. Schrader. B-Human 2019 – complex team play under natural lighting conditions. In S. Chalup, T. Niemueller, J. Suthakorn, and M.-A. Williams, editors, *RoboCup 2019: Robot World Cup XXIII*, pages 646–657, Cham, 2019. Springer International Publishing.

S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

N. Rudin, D. Hoeller, M. Bjelonic, and M. Hutter. Advanced skills by learning locomotion and local navigation end-to-end. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2497–2503. IEEE, 2022a.

N. Rudin, D. Hoeller, P. Reist, and M. Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on Robot Learning*, pages 91–100. PMLR, 2022b.

A. A. Rusu, S. G. Colmenarejo, C. Gulcehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, and R. Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.

M. Saggar, T. D'Silva, N. Kohl, and P. Stone. Autonomous learning of stable quadruped locomotion. In G. Lakemeyer, E. Sklar, D. Sorenti, and T. Takahashi, editors, *RoboCup-2006: Robot Soccer World Cup X*, volume 4434 of *Lecture Notes in Artificial Intelligence*, pages 98–109. Springer Verlag, Berlin, 2007. ISBN 978-3-540-74023-0.

S. Salter, M. Wulfmeier, D. Tirumala, N. Heess, M. Riedmiller, R. Hadsell, and D. Rao. Mo2: Model-based offline options. In *Conference on Lifelong Learning Agents*, pages 902–919. PMLR, 2022.

A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM J. Res. Dev.*, 3(3): 210–229, July 1959. ISSN 0018-8646.

S. Schmitt, J. J. Hudson, A. Zídek, S. Osindero, C. Doersch, W. M. Czarnecki, J. Z. Leibo, H. Küttler, A. Zisserman, K. Simonyan, and S. M. A. Eslami. Kickstarting deep reinforcement learning. *ArXiv*, abs/1803.03835, 2018.

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)*, 2015.

J. Siekmann, Y. Godse, A. Fern, and J. Hurst. Sim-to-real learning of all common bipedal gaits via periodic reward composition. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7309–7315. IEEE, 2021a.

J. Siekmann, K. Green, J. Warila, A. Fern, and J. Hurst. Blind bipedal stair traversal via sim-to-real reinforcement learning. *arXiv preprint arXiv:2105.08328*, 2021b.

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. *Nature*, 529(7587):484–489, 2016.

K. Sims. Evolving virtual creatures. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, page 15–22, New York, NY, USA, 1994. Association for Computing Machinery. ISBN 0897916670. doi: 10.1145/192161.192167.

L. Smith, I. Kostrikov, and S. Levine. A walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning. *arXiv preprint arXiv:2208.07860*, 2022.

L. Smith, J. C. Kew, T. Li, L. Luu, X. B. Peng, S. Ha, J. Tan, and S. Levine. Learning and adapting agile locomotion skills by transferring experience. *arXiv preprint arXiv:2304.09834*, 2023.

P. Stone. *Layered learning in multiagent systems - a winning approach to robotic soccer*. Intelligent robotics and autonomous agents. MIT Press, 2000.

P. Stone, R. S. Sutton, and G. Kuhlmann. Reinforcement learning for RoboCup-soccer keepaway. *Adaptive Behavior*, 13(3):165–188, 2005.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018. ISBN 0262039249.

R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. de Las Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, T. P. Lillicrap, and M. A. Riedmiller. Deepmind control suite. *CoRR*, abs/1801.00690, 2018.

A. A. Team, J. Bauer, K. Baumli, S. Baveja, F. M. P. Behbahani, A. Bhoopchand, N. Bradley-Schmieg, M. Chang, N. Clay, A. Collister, V. Dasagi, L. Gonzalez, K. Gregor, E. Hughes, S. Kashem, M. Loks-Thompson, H. Openshaw, J. Parker-Holder, S. Pathak, N. P. Nieves, N. Rakicevic, T. Rocktäschel, Y. Schroecker, J. Sygnowski, K. Tuyls, S. York, A. Zacherl, and L. M. Zhang. Human-timescale adaptation in an open-ended task space. *ArXiv*, abs/2301.07608, 2023.

Y. Teh, V. Bapst, W. M. Czarnecki, J. Quan, J. Kirkpatrick, R. Hadsell, N. Heess, and R. Pascanu. Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems*, 2017.

G. Tesauro. Temporal difference learning and TD-gammon. *Commun. ACM*, 38(3):58–68, Mar. 1995.

S. Thrun and A. Schwartz. Finding structure in reinforcement learning. *Advances in neural information processing systems*, 7, 1994.

D. Tirumala, A. Galashov, H. Noh, L. Hasenclever, R. Pascanu, J. Schwarz, G. Desjardins, W. M. Czarnecki, A. Ahuja, Y. W. Teh, et al. Behavior priors for efficient reinforcement learning. *The Journal of Machine Learning Research*, 23(1):9989–10056, 2022.

J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

S. Tunyasuvunakool, A. Muldal, Y. Doron, S. Liu, S. Bohez, J. Merel, T. Erez, T. Lillicrap, N. Heess, and Y. Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020.

K. Tuyls, S. Maes, and B. Manderick. Reinforcement learning in large state spaces. In G. A. Kaminka, P. U. Lima, and R. Rojas, editors, *RoboCup 2002: Robot Soccer World Cup VI*, volume 2752 of *Lecture Notes in Computer Science*, pages 319–326. Springer, 2002.

G. Vezzani, D. Tirumala, M. Wulfmeier, D. Rao, A. Abdolmaleki, B. Moran, T. Haarnoja, J. Humplik, R. Hafner, M. Neunert, et al. Skills: Adaptive skill sequencing for efficient temporally-extended exploration. *arXiv preprint arXiv:2211.13743*, 2022.

O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. P. Agapiou, M. Jaderberg, A. S. Vezhnevets, R. Leblond, T. Pohlen, V. Dalibard, D. Budden, Y. Sulsky, J. Molloy, T. L. Paine, Ç. Gülçehre, Z. Wang, T. Pfaff, Y. Wu, R. Ring, D. Yogatama, D. Wünsch, K. McKinney, O. Smith, T. Schaul, T. P. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps, and D. Silver. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782): 350–354, 2019. doi: 10.1038/s41586-019-1724-z.

J. Won, D. Gopinath, and J. K. Hodgins. Control strategies for physically simulated characters performing two-player competitive sports. *ACM Trans. Graph.*, 40(4):146:1–146:11, 2021.

P. Wu, A. Escontrela, D. Hafner, P. Abbeel, and K. Goldberg. DayDreamer: World models for physical robot learning. In *Conference on Robot Learning*, pages 2226–2240. PMLR, 2023.

M. Wulfmeier, A. Abdolmaleki, R. Hafner, J. T. Springenberg, M. Neunert, T. Hertweck, T. Lampe, N. Siegel, N. Heess, and M. Riedmiller. Compositional transfer in hierarchical reinforcement learning. *arXiv preprint arXiv:1906.11228*, 2019.

M. Wulfmeier, D. Rao, R. Hafner, T. Lampe, A. Abdolmaleki, T. Hertweck, M. Neunert, D. Tirumala, N. Siegel, N. Heess, et al. Data-efficient hindsight off-policy option learning. In *International Conference on Machine Learning*, pages 11340–11350. PMLR, 2021.

Z. Xie, P. Clary, J. Dao, P. Morais, J. W. Hurst, and M. van de Panne. Iterative reinforcement learning based design of dynamic locomotion skills for Cassie. *CoRR*, abs/1903.09537, 2019. URL http://arxiv.org/abs/1903.09537.

W. Yu, V. C. Kumar, G. Turk, and C. K. Liu. Sim-to-real transfer for biped locomotion. In *2019 ieee/rsj international conference on intelligent robots and systems (IROS)*, pages 3503–3510. IEEE, 2019.

Appendix

A. Training Details

A.1. Joint Limits

The joint limits used through out training are given in Table 4. We prefix the joints corresponding to the left and right side with "l" and "r", respectively. Zero joint angles correspond to the T-pose.

ID	Joint Name	Min	Max
19	head_pan	−0.79	0.79
20	head_tilt	−0.63	−0.16
16	l_ank_pitch	−0.4	1.8
18	l_ank_roll	−0.4	0.4
6	l_el	−1.4	0.2
12	l_hip_pitch	−1.6	0.5
10	l_hip_roll	−0.4	−0.1
8	l_hip_yaw	−0.3	0.3
14	l_knee	−0.2	2.2
2	l_sho_pitch	−2.2	2.2
4	l_sho_roll	−0.8	1.6
16	r_ank_pitch	−1.8	0.4
17	r_ank_roll	−0.4	0.4
5	r_el	−0.2	1.4
11	r_hip_pitch	−0.5	1.6
9	r_hip_roll	0.1	0.4
7	r_hip_yaw	−0.3	0.3
13	r_knee	−2.2	0.2
1	r_sho_pitch	−2.2	2.2
3	r_sho_roll	−1.6	0.8

Table 4 | Joint Limits

A.2. Agent Training and Architecture

A.2.1. Distributional MPO

We trained agents using MPO with a distributional critic, which we refer to as Distributional MPO (DMPO). MPO, or Maximum a Posteriori Policy Optimization (Abdolmaleki et al., 2018), is an actor-critic deep RL algorithm based on policy iteration. Policy iteration algorithms alternate between performing *policy evaluation* and *policy improvement*. Policy evaluation estimates the Q-function (i.e., critic) given the current policy (i.e., actor). Policy improvement updates the policy given the current Q-function.

Policy Evaluation: In policy evaluation, the Q-function is learned for the current policy π_t, parameterized by the target policy network. Any off-the-shelf Q-learning algorithm can be used to learn the Q-function. We chose to use n-step Q-learning (Sutton and Barto, 2018) with $n = 5$. In addition, we parameterize the Q-function as a deep feed-forward neural network that outputs a categorical distribution, as introduced in Bellemare et al. (2017).

Policy Improvement: In policy improvement, the aim is to improve the policy with respect to the current Q-function Q_t and current policy π_t. MPO has a two-step policy improvement procedure. The

first step computes an improved action distribution $q(\boldsymbol{a}|\boldsymbol{s})$ that optimizes the standard RL objective together with a KL-divergence constraint to the current policy:

$$\begin{aligned} \max_q \mathbb{E}_\xi \left[\int_{\boldsymbol{a}} q(\boldsymbol{a}|\boldsymbol{s}) Q(\boldsymbol{s},\boldsymbol{a}) \mathrm{d}\boldsymbol{a} \right] \\ \text{s.t. } \mathbb{E}_\xi \Big[\mathrm{KL}\big(q(\cdot|\boldsymbol{s}) \| \pi_t(\cdot|\boldsymbol{s})\big) \Big] < \epsilon \,. \end{aligned} \tag{6}$$

The KL-divergence constraint ensures that the improved action distribution does not deviate too much from the current policy, which stabilizes learning. (6) has a closed-form solution for each state $\boldsymbol{s}$, where

$$q(\boldsymbol{a}|\boldsymbol{s}) \propto \pi_t(\boldsymbol{a}|\boldsymbol{s}) \exp\left(\frac{Q(\boldsymbol{s},\boldsymbol{a})}{\eta}\right), \tag{7}$$

and the temperature η is the solution to the convex dual function involving ϵ:

$$\eta = \operatorname*{argmin}_\eta \eta\epsilon + \eta \mathbb{E}_\xi \left[\log \int_{\boldsymbol{a}} \pi_t(\boldsymbol{a}|\boldsymbol{s}) \exp\left(\frac{Q(\boldsymbol{s},\boldsymbol{a})}{\eta}\right) \mathrm{d}\boldsymbol{a} \right]. \tag{8}$$

In practice, rather than setting η to the exact solution of the convex dual function, it is effective to optimize η via gradient descent, initializing with the solution found in the previous iteration.

The second step distills the improved action distribution q into a new parametric policy π_{t+1} (with parameters θ_{t+1}) via a supervised learning loss:

$$\begin{aligned} \theta_{t+1} = \operatorname*{argmax}_\theta \mathbb{E}_\xi \left[\int_{\boldsymbol{a}} q(\boldsymbol{a}|\boldsymbol{s}) \log \pi_\theta(\boldsymbol{a}|\boldsymbol{s}) \mathrm{d}\boldsymbol{a} \right] \\ \text{s.t. } \mathbb{E}_\xi \Big[\mathrm{KL}\big(\pi_t(\cdot|\boldsymbol{s}) \| \pi_\theta(\cdot|\boldsymbol{s})\big) \Big] < \beta \,. \end{aligned} \tag{9}$$

The additional KL-divergence constraint enforces that the policy does not change too much in a single iteration of policy improvement, which avoids premature convergence and improves learning stability (Abdolmaleki et al., 2018; Schulman et al., 2015).

A.2.2. Agent Hyperparameters

Table 5 lists the hyperparameters we used for agent architectures and training. Both the policy and critic were feed-forward neural networks. During training we sample a batch of trajectory segments from the replay buffer, of size B and segment length T, perform one step each of policy evaluation and policy improvement on this batch, and repeat. To stabilize learning, we maintain a separate online and target network for the policy and the Q-function (Mnih et al., 2015). The policy evaluation and improvement steps update the parameters of the online network, and the online network parameters are copied to the target network at fixed intervals.

B. Training

Learning curves for the training process in simulation are given in Figure 13. The get-up teacher learns to get up relatively quickly and trained in total for approximately $2.4 \cdot 10^8$ environment steps, equivalent to approximately 70 days of simulation time, or 14 hours of wall-clock time. The soccer teacher was trained for $2 \cdot 10^9$ environment steps, which took 158 hours of training, equivalent to approximately 580 days of simulated matches. As shown in Figure 13, the soccer teacher learned to score goal consistently in a fraction of that time, after which the learning curve plateaus. We found that soccer agents trained for longer were better at anticipating ball movement and handling the ball, for instance kicking a moving ball. Given the two teachers, the full 1v1 policy learned relatively very quickly and we finished training after $9 \cdot 10^8$ environment steps or 68 hours wall clock time.

Hyperparameter	Setting
Q-function / critic network:	
layer sizes	(400, 400, 400, 300)
support	[−150, 150]
number of atoms	51
n-step returns	5
discount factor γ	0.99
policy / actor network:	
layer sizes	(256, 256, 128)
minimum variance	0.001
policy and Q-function networks:	
layer norm on first layer?	yes
tanh on output of layer norm?	yes
activation (after each hidden layer)	ELU
MPO for policy improvement:	
actions sampled per state	20
ϵ	0.1
KL-constraint on mean, β_μ	0.0025
KL-constraint on covariance, β_Σ	10^{-6}
Adam learning rate for dual variables	10^{-2}
Adam learning rate for policy and Q-function networks	10^{-4}
training	
batch size, B	256
trajectory segment length, T	48
replay buffer size	10^6
target network update period for actor	25
target network update period for critic	100

Table 5 | Hyperparameters for agent architectures and training.

C. Additional Results

C.1. Training from Vision

To create a realistic visual simulation of the real world, we created a NeRF model (Mildenhall et al., 2021) of our soccer pitch inspired by the approach of Byravan et al. (2023). The NeRF model captures static properties of the scene including the background, goal locations and the pitch itself. We overlaid these images with renders from the MuJoCo camera of dynamic objects i.e. the ball and opponent. For robustness, we randomized over visual properties of the ball (e.g. color). In addition, we randomized training over NeRF models captured at different points in time to be robust to changes in the background and lighting conditions. The reward and environment setup was otherwise identical to that used for state. To increase learning efficiency, for every iteration of training a new agent, we mixed offline data gathered from previous iterations with online replay data.

We used the same agent (MPO with distributional critic) for vision as for our state-based analysis. The policy was conditioned on visual information and proprioception while the critic was given the same information as in state-based training. For our policy we used the same network configuration as Byravan et al. (2023).

Set piece analysis We conducted a simulated penalty shootout using a vision trained policy as shown in Figure 12 (left). The robot consistently scores across 10 trials in the set piece configuration.

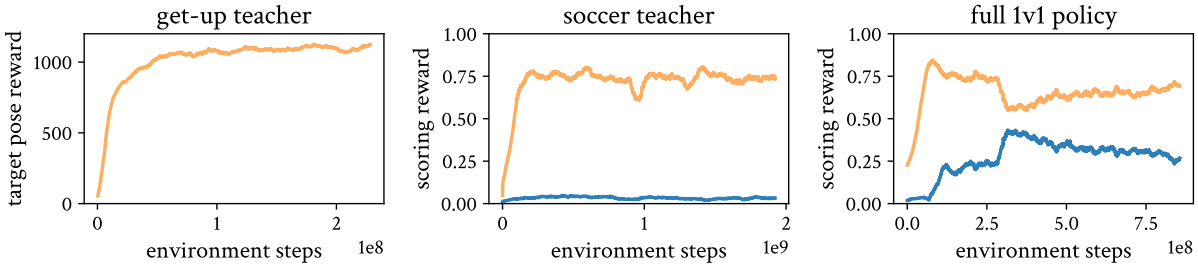


Figure 13 | Learning curves for get-up, soccer, and the full 1v1 task. We use the get-up and soccer agents at the end of training as the teachers for the 1v1 task. The get-up agent took 14 hours wall-clock time to train, the soccer teacher took 158 hours to train, and the full 1v1 policy took 68 hours to train. For the soccer and 1v1 tasks, the orange line denotes the scoring reward of the agent, where 1 corresponds to scoring a goal. The blue line denotes the negative of the conceding penalty, where 1 corresponds to the opponent scoring a goal.

The robot scored 6 out of 10 times (60 %) when transferred to the real world and hit the post 3 times. Variations in the visual elements of the scene and sensor noise could explain the discrepancy.

In addition in Figure 12 (right) we plot a trajectory where the ball is initialized moving in the negative y-direction. The robot predicts the movement of the ball and proactively alters its gait to strike at a particular location. Videos of agent behavior from vision can be found in the supplementary website https://sites.google.com/view/op3-soccer.