ROBOT LOCOMOTION

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[1,2], Jan Humplik[1], Markus Wulfmeier[1], Saran Tunyasuvunakool[1], Noah Y. Siegel[1], Roland Hafner[1], Michael Bloesch[1], Kristian Hartikainen[1]‡, Arunkumar Byravan[1], Leonard Hasenclever[1], Yuval Tassa[1], Fereshteh Sadeghi[1]§, Nathan Batchelor[1], Federico Casarini[1], Stefano Saliceti[1], Charles Game[1], Neil Sreendra[3], Kushal Patel[3], Marlon Gwira[3], Andrea Huber[1], Nicole Hurley[1]¶, Francesco Nori[1], Raia Hadsell[1], Nicolas Heess[1]

We investigated 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. We used deep RL to train a humanoid robot to play a simplified one-versus-one soccer game. The resulting agent exhibits robust and dynamic movement skills, such as rapid fall recovery, walking, turning, and kicking, and it transitions between them in a smooth and efficient manner. It also learned to anticipate ball movements and block opponent shots. The agent's tactical behavior adapts to specific game contexts in a way that would be impractical to manually design. Our agent was trained in simulation and transferred to real robots zero-shot. A combination of sufficiently high-frequency control, targeted dynamics randomization, and perturbations during training enabled good-quality transfer. In experiments, the agent walked 181% faster, turned 302% faster, took 63% less time to get up, and kicked a ball 34% faster than a scripted baseline.

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 artificial intelligence (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 (*1*) and in the real world (*2*, *3*). Progress has recently accelerated considerably, and learning-based approaches have contributed substantially to this acceleration (*4*–*6*). In particular, deep reinforcement learning (deep RL) has proven capable of solving complex motor control problems for both simulated characters (*7*–*11*) and physical robots. High-quality quadrupedal legged robots have become widely available and have been used to demonstrate behaviors ranging from robust (*12*, *13*) and agile (*14*, *15*) locomotion to fall recovery (*16*); climbing (*17*); basic soccer skills such as dribbling (*18*, *19*), shooting (*20*), intercepting (*21*), or catching (*22*) a ball; and simple manipulation with legs (*23*). On the other hand, much less work has been dedicated to the control of humanoids and bipeds, which impose additional challenges around stability, robot safety, number of degrees of freedom, and availability of suitable hardware. The existing learning-based work has been more limited and focused on learning and transfer of distinct basic skills, such as walking (*24*), running (*25*), stair climbing (*26*), and jumping (*27*). The state of the art in humanoid control uses targeted model-based predictive control (*28*), thus limiting the generality of the method.

Our work focuses on learning-based full-body control of humanoids for long-horizon tasks. In particular, we used deep RL to train low-cost off-the-shelf robots to play multi-robot soccer well beyond the level of agility and fluency that is intuitively expected from this type of robot. Sports like soccer showcase many of the hallmarks of human sensorimotor intelligence, which has been recognized in the robotics community, especially through the RoboCup (*29*, *30*) initiative. We considered a subset of the full soccer problem and trained an agent to play simplified one-versus-one (1v1) soccer in simulation and directly deployed the learned policy on real robots (Fig. 1). We focused on sensorimotor full-body control from proprioceptive and motion capture observations.

The trained agent exhibited agile and dynamic movements, including walking, side stepping, kicking, fall recovery, and ball interaction, and composed these skills smoothly and flexibly. The agent discovered unexpected strategies that made more use of the full capabilities of the system than scripted alternatives and that we may not have even conceived of. An example of this is the emergent turning behavior, in which the robot pivots on the corner of a foot and spins, which would be challenging to script and which outperformed the more conservative baseline (see the "Comparison with scripted baseline controllers" section). One further problem in robotics in general and in robot soccer in particular is that optimal behaviors are often context dependent in a way that can be hard to predict and manually implement. We demonstrate that the learning approach can discover behaviors that are optimized to the specific game situation. Examples include context-dependent agile skills, such as kicking a moving ball; emergent tactics, such as subtle defensive running patterns; and footwork that adapts to the game situation, such as taking shorter steps when approaching an attacker in

[1]Google DeepMind, London, UK. [2]University College London, London, UK. [3]Proactive Global, London, UK.
*Corresponding author. Email: tuomash@google.com (T.H.); guylever@google.com (G.L.)
†These authors contributed equally to this work.
‡Present address: University of Oxford, Oxford, UK.
§Present address: Google, Mountain View, CA, USA.
¶Present address: Isomorphic Labs, London, UK.

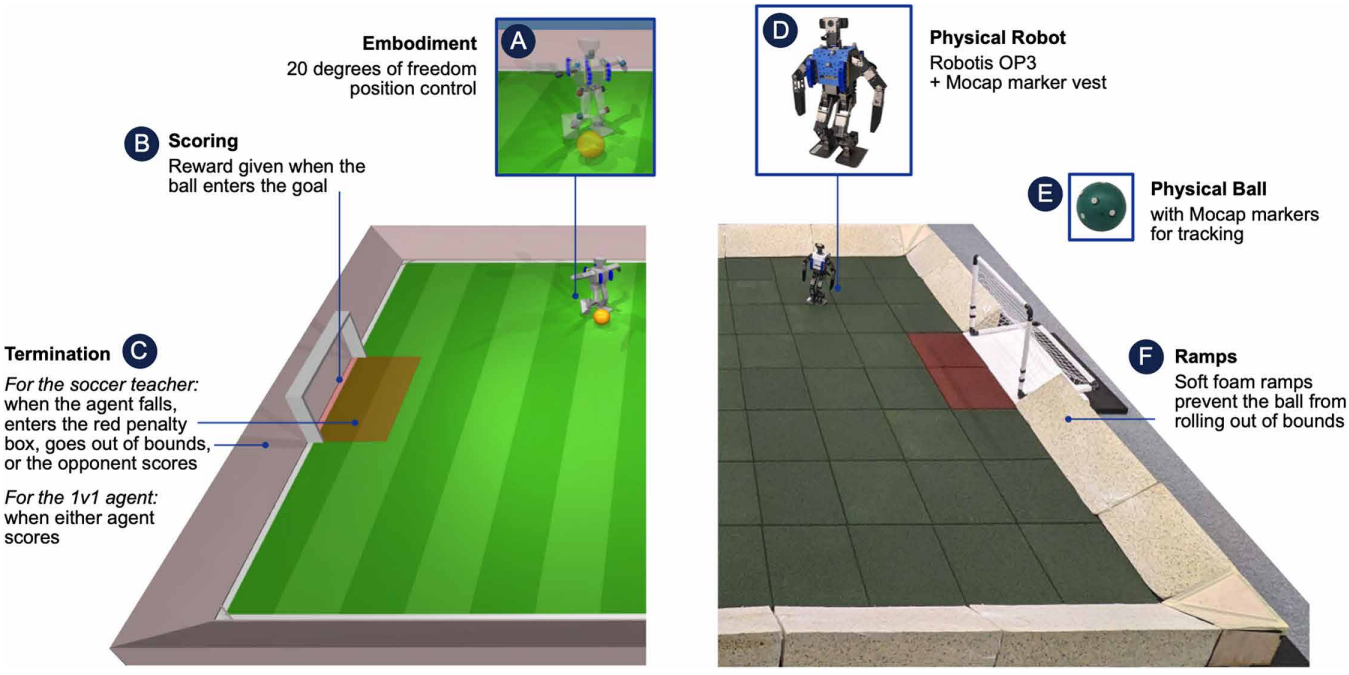


Fig. 1. The robot soccer environment. We created matching simulated (left) and real (right) soccer environments. The pitch is 5 m long by 4 m wide, tiled with 50-cm square panels in the real environment. The real environment was also equipped with a motion capture (mocap) system for tracking the two robots and the ball.

possession of the ball compared with when chasing a loose ball (see the "Behavior analysis" section). The agent learned to make predictions about the ball and the opponent, to adapt movements to the game context, and to coordinate them over long time scales for scoring while being reactive to ensure dynamic stability. Our results also indicate that, with appropriate regularization, domain randomization, and noise injected during training, safe sim-to-real transfer is possible even for low-cost robots. Example behaviors and gameplay can be seen in the accompanying movies and on the website https://sites.google.com/view/op3-soccer.

Our training pipeline consists of two stages. In the first stage, we trained two skill policies: one for getting up from the ground and another for scoring a goal against an untrained opponent. In the second stage, we trained agents for the full 1v1 soccer task by distilling the skills and using multiagent training in a form of self-play, where the opponent was drawn from a pool of partially trained copies of the agent itself. Thus, in the second stage, the agent learned to combine previously learned skills, refine them to the full soccer task, and predict and anticipate the opponent's behavior. We used a small set of shaping rewards, domain randomization, and random pushes and perturbations to improve exploration and to facilitate safe transfer to real robots. An overview of the learning method is shown in Fig. 2 and Movie 1 and discussed in Materials and Methods.

We found that pretraining separate soccer and get-up skills was the minimal set needed to succeed at the task. Learning end to end without separate soccer and get-up skills resulted in two degenerate solutions depending on the exact setup: converging to either a poor locomotion local optimum of rolling on the ground or focusing on standing upright and failing to learn to score (see the "Ablations" section for more details). Using a pretrained get-up skill simplified the reward design and exploration problem and avoided poor locomotion local optima. More skills could be used, but we used pretrained skills only when necessary because using a minimal set of skills allows emergent behaviors to be discovered by the agent and optimized for specific contexts (highlighted in the "Comparison with scripted baseline controllers" and "Behavior analysis" sections) rather than learning to sequence prespecified behaviors.

RESULTS

We evaluated the agent in a 1v1 soccer match on physical Robotis OP3 miniature humanoid robots (*31*) and analyzed the emergent behaviors. We isolated certain behaviors (walking, turning, getting up, and kicking), compared them with corresponding scripted baseline controllers (see the "Comparison with scripted baseline controllers" section), and qualitatively analyzed the behaviors in a latent space (see the "Behavior embeddings" section). To assess reliability, gauge performance gaps between simulation and reality and study the sensitivity of the policy to game state, we also investigated selected set pieces (see the "Behavior analysis" section). Last, we investigated the agent's sensitivity to the observations of the ball, goal, and the opponent using a value function analysis (see the "Value function analysis" section).

Selected extracts from the 1v1 matches can be seen in Fig. 3 and Movie 2. The agent exhibited a variety of emergent behaviors, including agile movement behaviors 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, and blocking shots; and strategic 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 behaviors fluidly.

CORRECTION POSTED 17 APRIL 2024

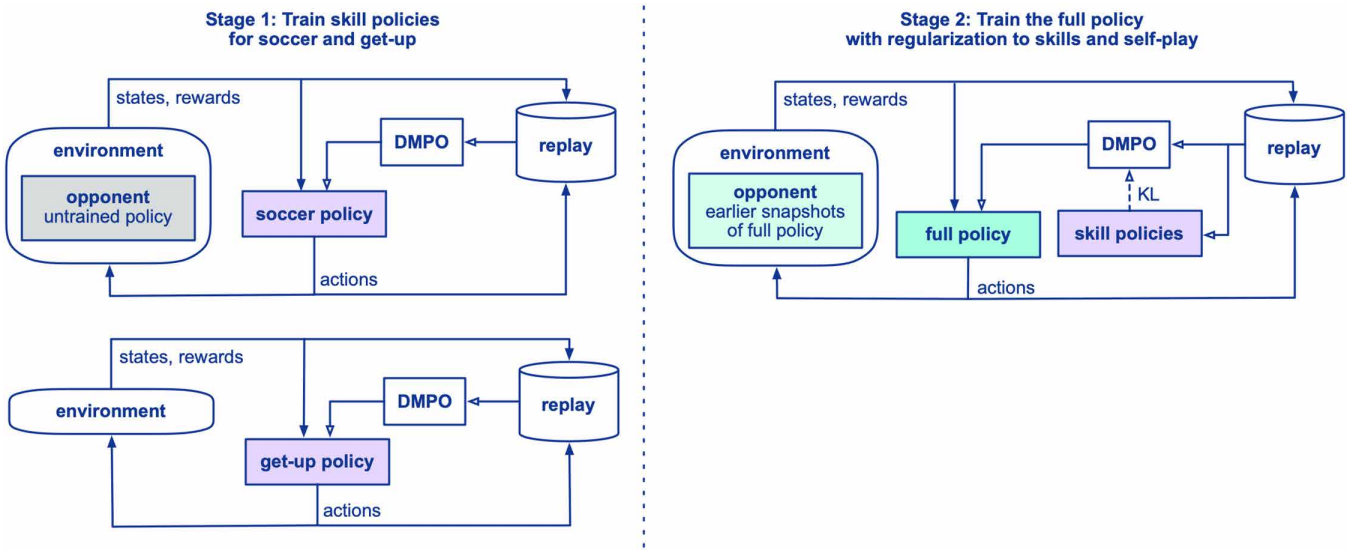
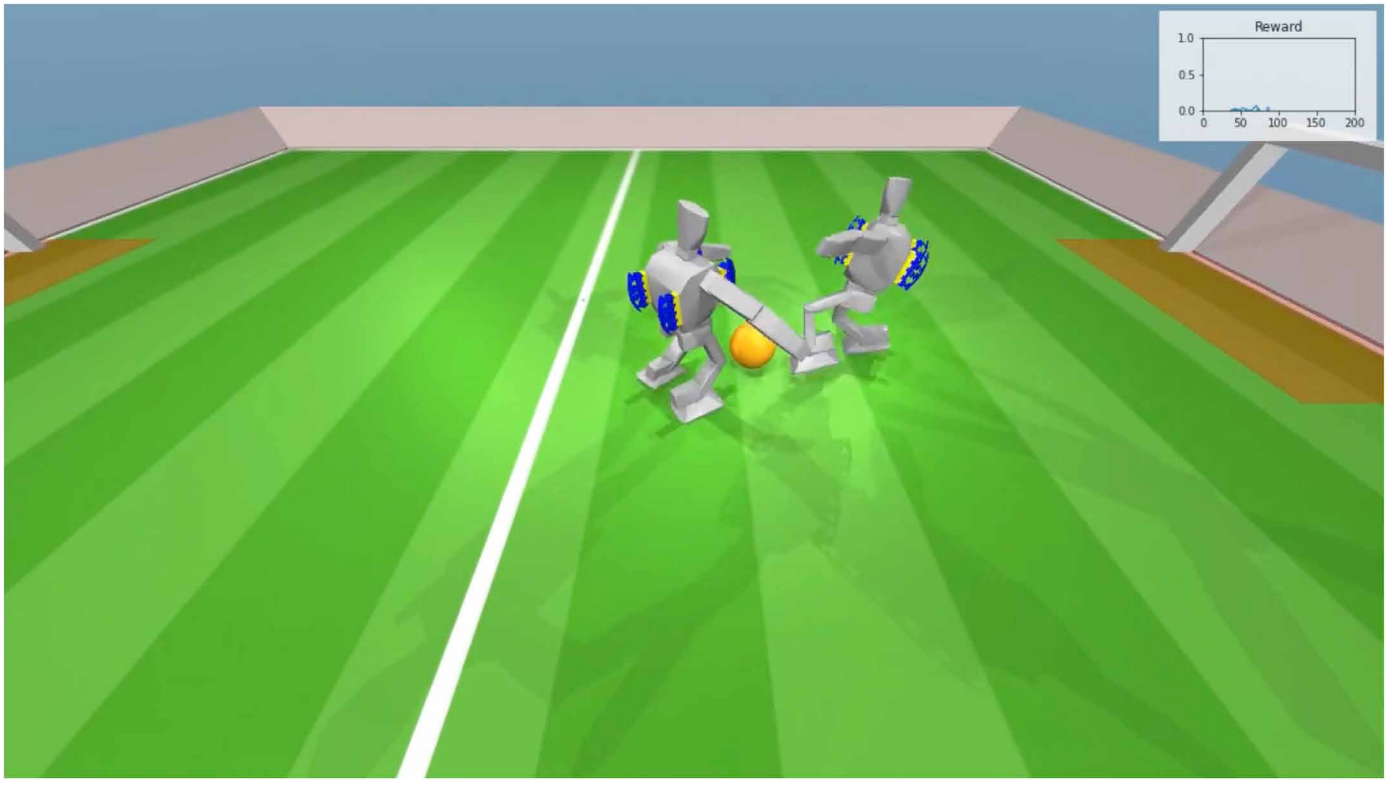


Fig. 2. Agent training setup. We trained agents in two stages. (**Left**) In stage 1, we trained a separate soccer skill and get-up skill ("Stage 1: Skill training" section). (**Right**) In stage 2, we distilled these two skills into a single agent that can both get up from the ground and play soccer ("Stage 2: Distillation and self-play" section). The second stage also incorporates self-play: The opponent is uniformly randomly sampled from saved 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 with training an agent from scratch for the full 1v1 soccer task.



Movie 1. Narrated video that summarizes the setup and approach.

Comparison with scripted baseline controllers

Certain key locomotion behaviors, including getting up, kicking, walking, and turning, are available for the OP3 robot (*32*), and we used these as baselines. The baselines are parameterized open-loop trajectories. For example, the walking controller, which can also perform turning, has tunable step length, step angle, step time, and joint offsets. We optimized the behaviors by performing a grid search over step length (for walking), step angle (for turning), and step time (for

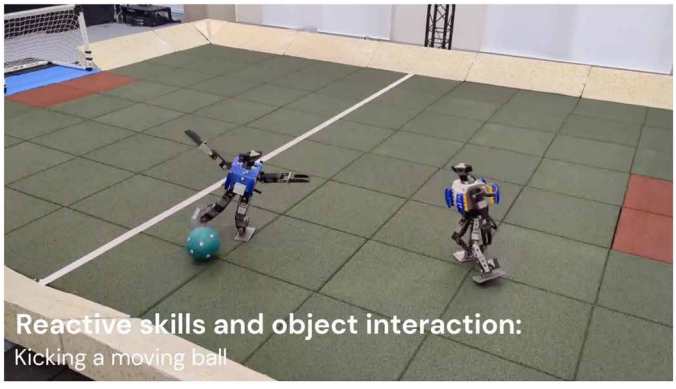
CORRECTION POSTED 17 APRIL 2024

Fig. 3. Gallery of robot behaviors. Each row gives an example of a type of behavior that was observed when the trained policy was deployed on real robots. The top six rows are photographs taken from five consecutive matches played on the same day, using the same policy. The bottom row of photographs demonstrate the same policy but were not taken from match play.

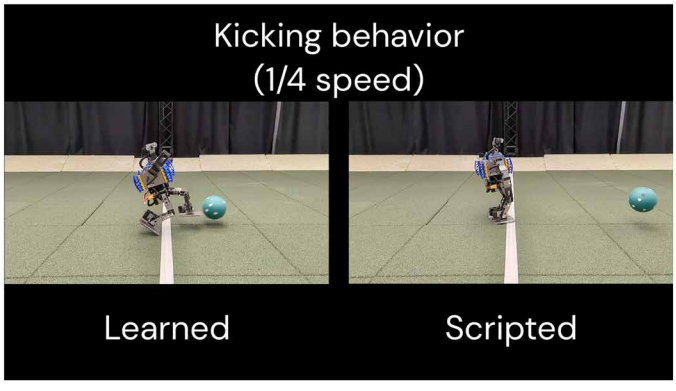
both) on a real robot. We adjusted the joint offsets where necessary to prevent the robot from losing balance or the feet colliding with each other. The kick and the get-up controllers were specifically designed for this robot and have no tunable parameters. To measure how well the learned deep RL agent performed on these key behaviors, we compared it both quantitatively and qualitatively against the baselines. See Movie 3 for an illustration of the baselines and a side-by-side comparison with the corresponding learned behaviors.

CORRECTION POSTED 17 APRIL 2024

Details of the comparison experiments are given in the "Baseline behavior comparisons: Experiment details" section in the Supplementary Materials, and the results are given in Table 1. The learned policy performed better than the specialized manually designed controller: It walked 181% faster, turned 302% faster, and took 63% less time to get up. When initialized near the ball, the learned policy kicked the ball with 3% 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 2.8 m/s (34% faster than the scripted controller), and the maximum kicking speed across episodes was 3.4 m/s. As well as outperforming the scripted get-up behavior, in practice, the learned policy also reacts to prevent falling in the first instance (see movie S4).



Movie 2. Highlight reel of behaviors and skills from real-world gameplay.



Movie 3. Side-by-side comparison of learned versus scripted behaviors.

Close observation of the learned policy (as shown in Movies 3 and 4 and figs. S5 and S6) reveals that it has learned to use a highly dynamic gait. Unlike the scripted controller, which centers the robot's weight over the feet and keeps the foot plates almost parallel to the ground, the learned policy leans forward and actively pushes off from the edges of the foot plate at each step, landing on the heels.

The forward running speed of 0.57 m/s and turning speed of 2.85 rad/s achieved by the learned policy on the real OP3 compare favorably with the values of 0.45 m/s and 2.01 rad/s reported in (*33*) on simulated OP3 robots. This latter work optimized parametric controllers of the type used by top-performing RoboCup teams. It was evaluated only in the RoboCup simulation, and the authors note that the available OP3 model featured unrealistically powerful motors. Although those results are not precisely comparable because of methodological differences, this gives an indication of how our learned policy compares with parameterized controllers implemented on the OP3 in simulation.

Behavior embeddings

A motivation for adopting end-to-end learning for the 1v1 soccer task was to obtain a policy that could blend many behaviors continuously, to react smoothly during play. To illustrate how the learned policy does this, we took inspiration from the analysis of *Drosophila* motion (*34*) and treated the motions as paths through 20-dimensional (20D) joint space. We used Uniform Manifold Approximation and Projection (UMAP) (*35*) to approximately embed these paths into 3D space to better visualize the behaviors.

Figure 4 compares the scripted and learned embeddings. The scripted walking controller is based on sinusoidal motions of the end effectors, and this periodicity means that the gait traces a cyclic path through joint space. This topological structure appears in the UMAP embedding, with the angular coordinate around the circle defined by the phase within the periodic gait (Fig. 4A). In contrast, embeddings of short trajectories from the learned policy reveal richer variation (Fig. 4B). Footsteps still appear as loops, but the gaits are no longer exactly periodic, so the embedded trajectories are helical rather than cyclic. Different conditions also elicit different gaits, such as fast running and slower walking, and these embed as different components. A kick common to these trajectory snippets appears as a smooth arc.

Last, Fig. 4C shows embeddings for long episodes of 1v1 play. The figure reveals a wide range of different cyclic gaits that map in a

Table 1. Performance at specific behaviors. The learned behavior is compared with the scripted baseline at the four behaviors. 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.

	Walking speed mean (SD)	Turning speed mean (SD)	Get-up time mean (SD)	Kicking speed mean (SD)
Scripted baseline	0.20 m/s (0.005 m/s)	0.71 rad/s (0.04 rad/s)	2.52 s (0.006 s)	2.07 m/s (0.05 m/s)
Learned policy (real robot)	0.57 m/s (0.003 m/s)	2.85 rad/s (0.19 rad/s)	0.93 s (0.12 s)	2.02 m/s (0.26 m/s)
With run up	–	–	–	2.77 m/s (0.11 m/s)
Learned policy (simulation)	0.51 m/s (0.01 m/s)	3.19 rad/s (0.12 rad/s)	0.73 s (0.01 s)	2.12 m/s (0.07 m/s)

CORRECTION POSTED 17 APRIL 2024

Movie 4. Slow-motion highlight reel of turning and kicking behaviors.

dense ball in this low-dimensional embedding space. However, kicking and getting up show much less variation, resulting in four distinct loops. This could be a result of the regularization to the scripted get-up controller: Although the final policy could in principle converge to any behavior, regularization steers learning toward a specific way of getting up. On the other hand, kick motion is dominated by swinging one of the legs as fast as possible, allowing less room for variations.

Behavior analysis

Reliability and sim-to-real analysis

To gauge the reliability of the learned agent, we designed a get-up-and-shoot set piece, implemented in both the simulation (training) environment and the real environment. This set piece is a short episode of 1v1 soccer in which the agent must get up from the ground

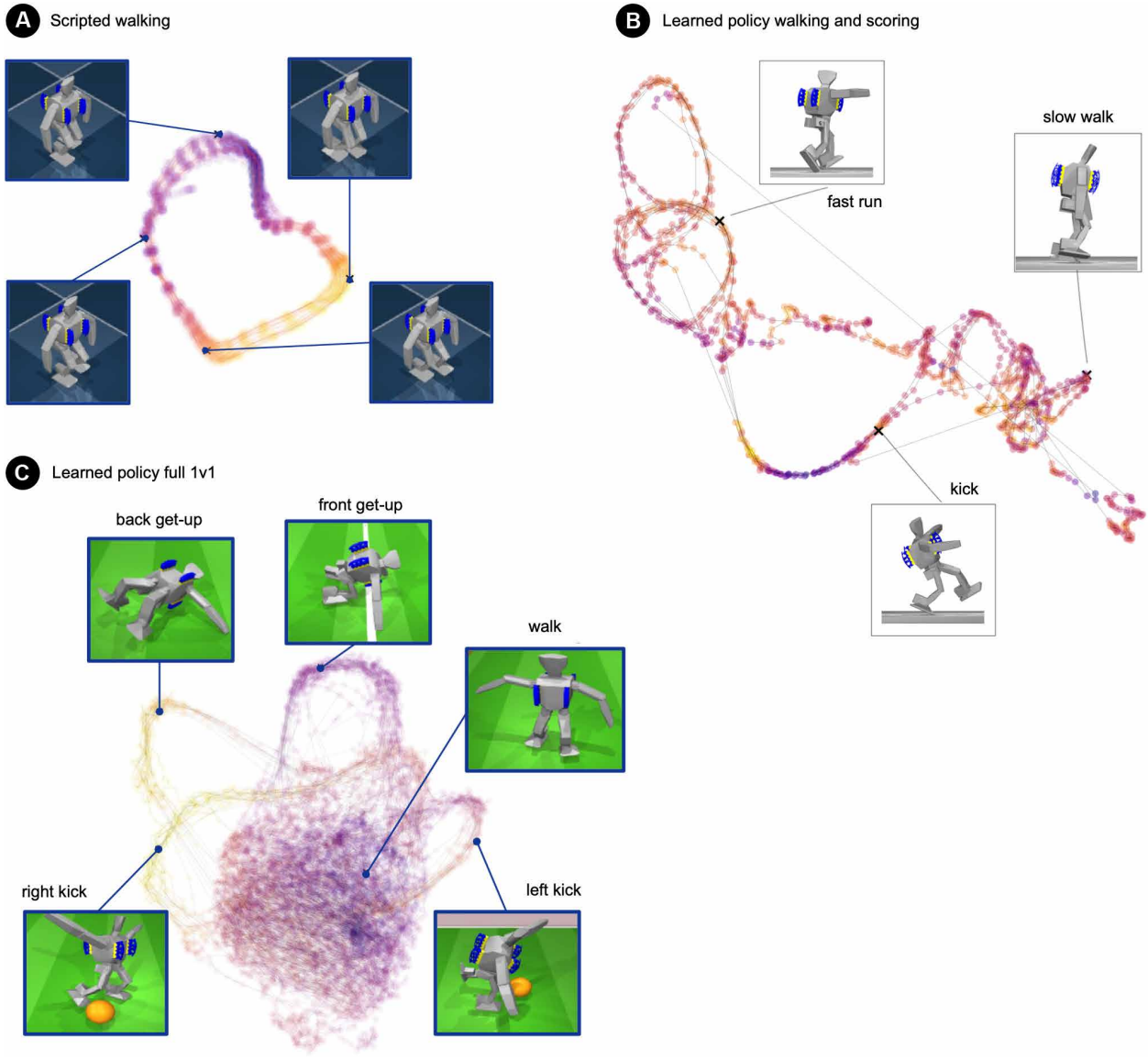


Fig. 4. Joint angle embeddings. Embedding of the joint angles recorded while executing different policies, as described in the "Behavior embeddings" section. (**A**) The embedding for the scripted baseline walking policy. (**B**) The embedding for the soccer skill. (**C**) The embedding for the full 1v1 agent.

and score within 10 s; see the "Set piece: Experiment details" section in the Supplementary Materials for full details.

We played 50 episodes of the set piece each in the simulation and the real environment. In the real environment, the robot scored 29 of 50 (58%) of goals and was able to get up from the ground and kick the ball every time. In simulation, the agent scored more consistently, scoring 35 of 50 (70%) of goals. This indicates a drop in performance due to transfer to the real environment, but the robot was still able to reliably get up, kick the ball, and score the majority of the time. Results are given in Fig. 5A and Table 1.

To further gauge the sim-to-real gap, we also analyzed the four behaviors discussed in the "Comparison with scripted baseline controllers" section (walking, turning, getting up, and kicking) in simulation and compared the results with those obtained using the real

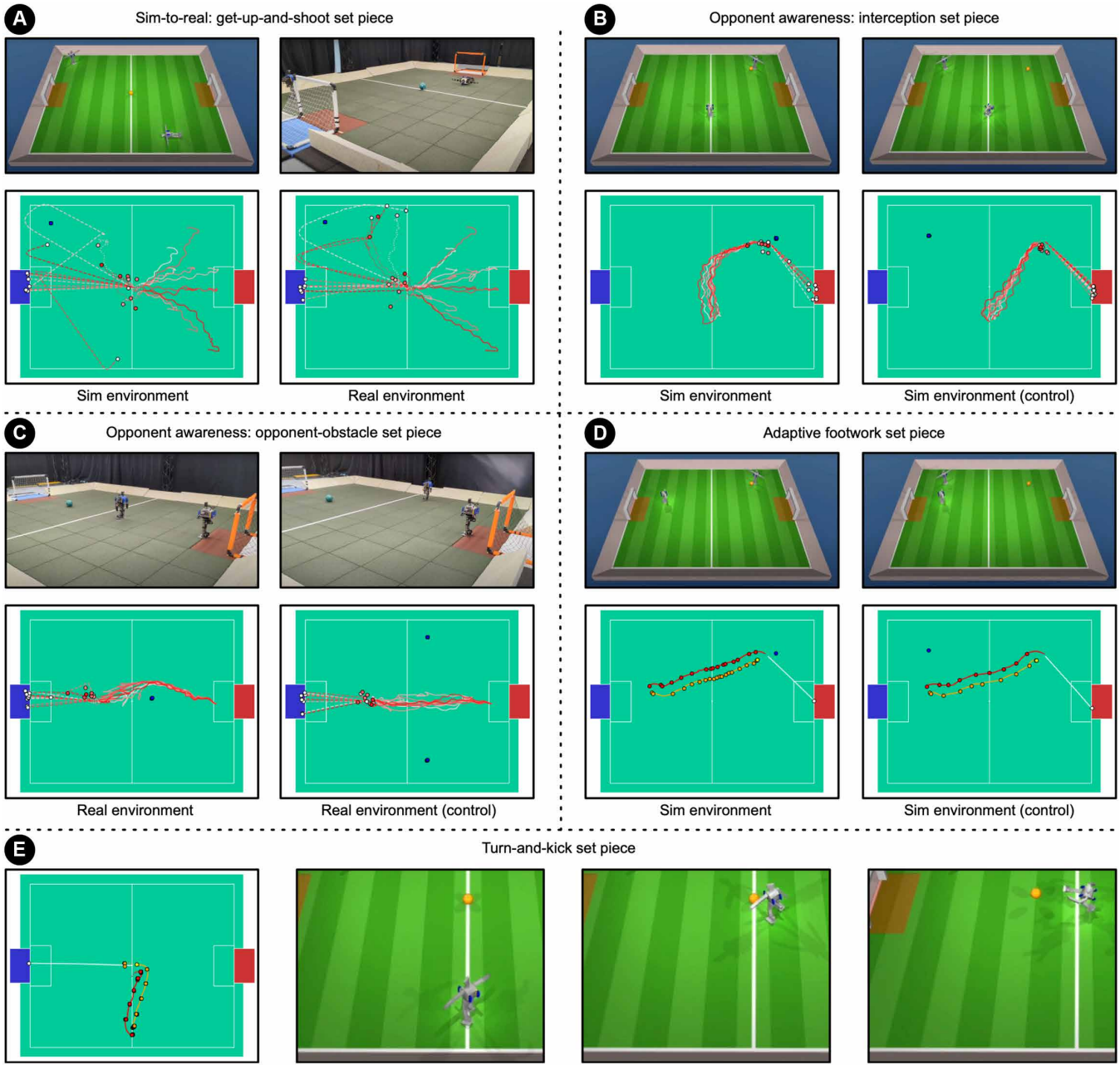


Fig. 5. Behavior analysis. (**A** to **C**) Set pieces. Top rows: Example initializations for the set piece tasks in simulation and on the real robot. Second rows: Overlayed plots of the 10 trajectories collected from the set piece experiments showing the robot trajectory before kicking (solid lines) and after kicking (dotted lines), the ball trajectory (dashed lines), final ball position (white circle), final robot position (red-pink circles), and opponent position (blue circle). Each red-pink shade corresponds to one of the 10 trajectories. (**D**) Adaptive footwork set piece. Right foot trajectory (orange), left foot trajectory (red), ball trajectory (white), point of the kick (yellow), and footsteps highlighted with dots. (**E**) Turn-and-kick set piece. Right three panels: A sequence of frames from the set piece. Left: A plot of the footsteps from the corresponding trajectory. The agent turned, walked ~2 m, turned, kicked, and lastly balanced using 10 footsteps. Please refer to the "Behavior analysis" section for a discussion of these results.

Table 2. Performance in the get-up-and-shoot set piece. Performance in simulation and on the real robot are compared. Values in parentheses are standard errors.

	Scoring success rate		Mean time to first touch	
	Sim env.	Real env.	Sim env.	Real env.
Get-up and shoot	0.70 (0.07)	0.58 (0.07)	4.6 s (0.16 s)	4.7 s (0.14 s)

OP3. Results are shown in Table 2: When implemented on the real robot, the learned policy walked (13%) faster, turned (11%) more slowly, took (28%) more time to get up, and kicked (5%) more slowly than when implemented in simulation. These results indicate no extreme sim-to-real gap in the execution of any behavior. The gap with the baseline behavior performance is substantially larger, for instance. The turning behavior is highly optimized (pivoting on a corner of the foot) and can be seen both in simulation and on the real robot in Movie 4.

Opponent awareness

To gauge the learned behavior's reaction to the opponent in a controlled setting, we implemented interception and opponent-obstacle set pieces in the simulation and real environments, respectively. The interception set piece is an 8-s episode of 1v1 soccer in which the opponent is initialized in possession of the ball and remains stationary throughout the episode. In 10 trials, the agent first walked to the path between the ball and its own goal, to block a potential shot from the opponent, before turning toward the ball and approaching the ball and opponent. This type of defensive behavior is manually implemented in some RoboCup teams to defend the goal against an attacker with possession (*36*). Our learned policy, in contrast, discovered this tactic "on its own" by optimizing for task reward (which includes minimizing opponent scoring) rather than via manual specification. In 10 control trials, the opponent was initialized away from the ball, and, in these situations, the agent approached the ball directly.

The opponent-obstacle set piece is a 10-s episode of 1v1 soccer in which the opponent is initialized midway between the ball and the agent, 1.5 m from each, and remains stationary throughout the episode, and the ball is 1.5 m from the goal. In 10 trials, the agent walked around the opponent every time to reach the ball and scored in 9 of the 10 trials. In 14 control trials, the opponent was initialized to either side of the pitch, not obstructing the agent's path to the ball, and, in this case, the agent approached the ball directly and scored in 13 of the 14 trials. These results demonstrate that behaviors that subtly adapt to the position of the opponent emerge during training, resulting in a policy that is optimized for specific contexts. Results and initial configurations are given in Fig. 5 (B and C).

Adaptive footwork

In the adaptive footwork set piece, the opponent is initialized in possession of the ball and remains stationary throughout the episode, and the agent is placed in a defensive position. In 10 trials, the agent took an average of 30 short footsteps to approach the attacker and ball. In comparison, in 10 control trials, in which the opponent is positioned away from the ball, the agent took an average of 20 longer strides as it rushed to the ball. The particular short-stepping tactic discovered by the agent is reminiscent of human 1v1 defensive play in which short quick steps are preferred to long strides to maximize reactivity. Although the reason for using short footsteps is unclear in our environment (it could be, for example, that the agent takes extra care to stay on the path between ball and goal and so moves more slowly), this result demonstrates that the agent adapts its gait to the specific context of the game. Results are illustrated in Fig. 5D.

To further 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 it, 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 seen in Fig. 5E, 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.

Value function analysis

Next, we investigated the learned value function in several setups to understand what game states the agent perceived as advantageous versus disadvantageous. For Fig. 6A, we created a synthetic observation with the robot located centrally on the court at coordinates (0, 0), facing toward the opponent goal with the ball 0.1 m in front of it at location (0.1, 0). The opponent was placed to one side, at location (0, 1.5). The plot shows the predicted value as a function of the ball's velocity vector; high values are assigned to ball velocities directed toward the goal and to ball velocities consistent with keeping the ball in the area under the agent's control.

A similar analysis shows that the value function also captures that the opponent impeded scoring (Fig. 6B). We used the same positions for the agent and ball and plotted the predicted value as a function of the opponent position. States in which the opponent was positioned between the ball and the goal had a much lower value.

Figure 6 (C and D) plots the predicted value as a function of the agent's location in scenarios like those used for the interception behavior described above in the "Behavior analysis" section. When the opponent was far from the ball, the agent preferred to be in a position from which it could shoot, and contour gradients tended to point directly toward this location. In contrast, when the opponent was near the ball, the agent's preferred positions were between the ball and the defended goal, and the rigid contours favored curved paths that guided the agent to this defensive line. This behavior was seen in the interception set piece in Fig. 5B.

DISCUSSION

Comparison with robot learning literature

RL for robots has been studied for decades [see (*5*, *6*) for an overview] but has only recently gained more popularity because of the

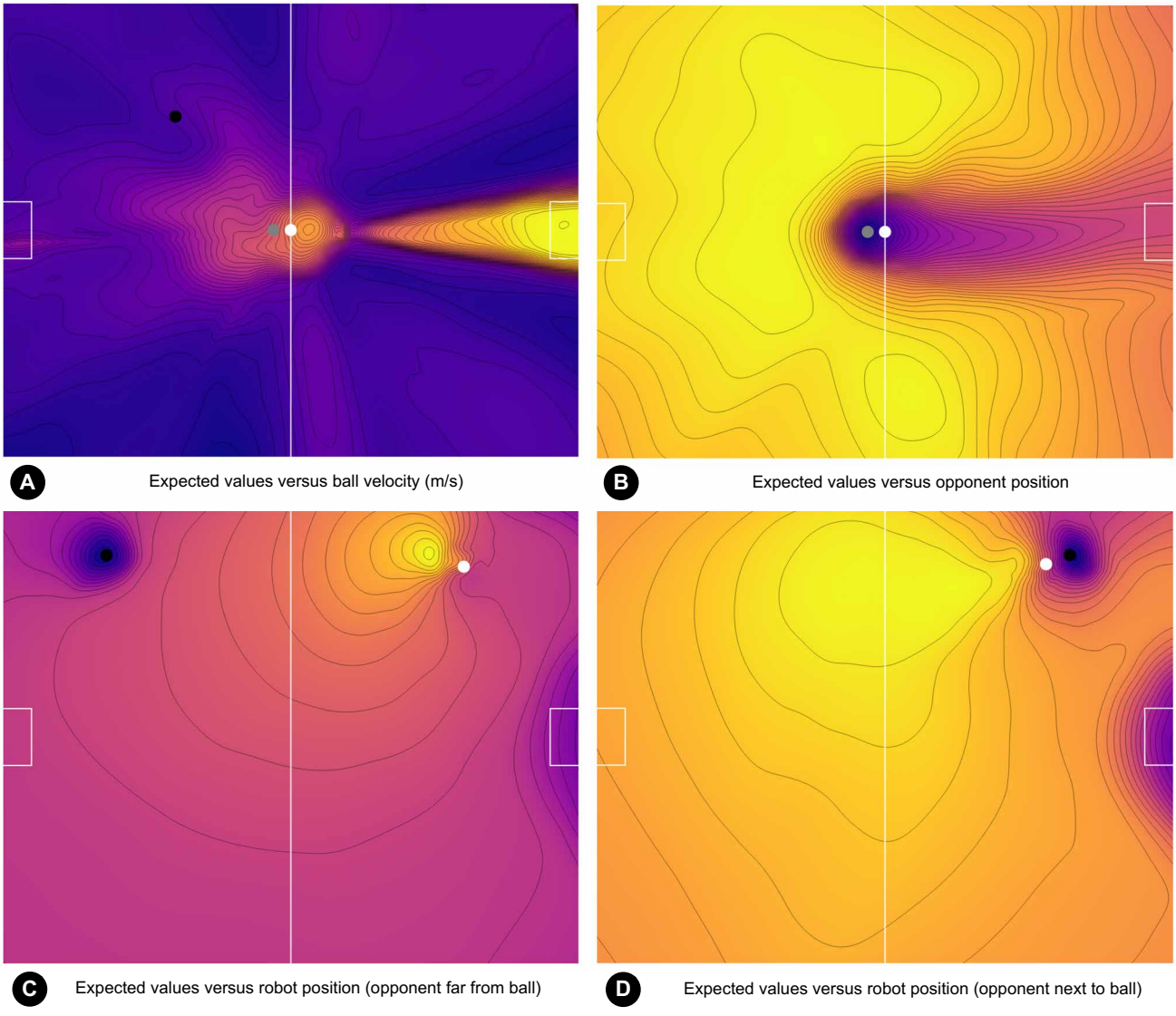


Fig. 6. Critic's predicted values. Projections of the expected values learned by the critic in selected game states. (In each panel, the ball is marked in white, the opponent in black, and the agent in gray, and brighter colors indicate preferred states.) (**A**) Varying the ball (*x*, *y*) velocity (instantaneous velocity in meters per second, mapped to the position after 1 s). High-value states are those in which the ball is either traveling toward the goal or remaining near the agent. (**B**) Varying the opponent position around the pitch. The predicted value is lower when the opponent is located between the ball and the target goal. (**C**) Varying the agent position when the opponent is far from the ball (high value at positions near where the agent shoots, low value around the opponent reflecting the interference penalty). (**D**) Varying the agent position when the opponent is close to the ball. (The value function has higher value ridges at locations blocking the opponent's shot.)

development of better hardware and algorithms (*4*, *37*). In particular, high-quality quadrupedal robots have become widely available, which have been used to demonstrate robust, efficient, and practical locomotion in a variety of environments (*12*, *13*, *38*, *39*). For example, Lee *et al.* (*12*) applied zero-shot sim-to-real deep RL to deploy learned locomotion policies in natural environments, including mud, snow, vegetation, and streaming water. Our work similarly relies on zero-shot sim-to-real transfer and model randomization but instead focuses on a range of dynamic motions, stability, long horizon tasks, object manipulation, and multiagent competitive play. Most recent work in this area relies on some form of sim-to-real transfer (*14*, *15*, *17*, *19*, *27*, *40*–*42*), which can help to reduce the safety and data efficiency concerns associated with training directly on hardware. A common theme is that an unexpectedly small number of techniques can be sufficient to reduce the sim-to-real gap (*37*, *43*), which is also supported by our results. However, there have also been successful attempts at training legged robots to walk with deep RL directly on hardware (*44*–*48*). Training on hardware can lead to better performance, but the range of behaviors that can be learned has so far been limited because of safety and data efficiency concerns. Similar to our work, prior work has shown that learned gaits can achieve higher velocities compared with scripted gaits (*49*–*51*). However, the gaits have been specifically trained to attain high speeds instead of emerging as a result of optimizing for a higher level goal.

Quadrupedal platforms constitute most legged locomotion research, but an increasing number of works consider bipedal platforms. Recent works have produced behaviors including walking and running (*24*, *25*), stair climbing (*26*), and jumping (*27*). Most recent works have focused on high-quality, full-sized bipeds and humanoids, with a much smaller number (*48*, *52*–*54*) targeting more basic platforms whose simpler and less precise actuators and sensors pose additional challenges in terms of sim-to-real transfer. In addition, there is a growing interest in whole-body control, that is, tasks in which the whole body is used in flexible ways to interact with the environment. Examples include getting up from the ground (*55*) and manipulation of objects with legs (*23*, *56*). Recently, RL has been applied to learn simple soccer skills, including goalkeeping (*21*), ball manipulation (*18*, *19*), and shooting (*20*). 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.

Comparison with RoboCup

Robot soccer has been a longstanding grand challenge for AI and robotics, since at least the formation of the RoboCup competition (*29*, *30*) in 1996, and it has also inspired our 1v1 soccer task. The OP3 robot has been used for the humanoid RoboCup league, but our environment and task are substantially simpler than the full RoboCup problem. The main differences are that we focused on 1v1 soccer instead of multiplayer teams; our environment did not align with the field or ball specifications or follow the rules of RoboCup (for example, the kick-off, player substitutions, explicit communication channels, fouls, and game length); and we used full state information rather than rely solely on vision.

The majority of successful RL approaches to RoboCup focus on learning specific components of the system and often feature manually designed components or high-level strategies. In Simulation 2D League, RL has been used to learn various ball handling skills (*57*, *58*) and multiagent behaviors such as defense (*59*) and ball control (*60*–*62*). One successful learning-based approach applied to the Simulation 3D League is layered learning (*63*, *64*). There, RL was used to train multiple skills, including ball control and pass selection, which were combined via a predefined hierarchy. Our system predefines fewer skills (leaving the agent to discover useful skills like kicking), and we focused on learning to combine the skills seamlessly. In addition, RL has been used for fast running (*65*, *66*), but compared with our work, the learned behaviors were not demonstrated on hardware. A smaller set of works has focused on applying RL to real robots. Policy gradient methods have been used to optimize parameterized walking (*67*) and kicking (*68*) on a quadrupedal robot for the RoboCup Four-Legged League. Riedmiller *et al.* (*59*) applied RL to learn low-level motor speed control, as well as a separate dribbling controller for the wheeled Middle-Size League. Simulation grounding by sim-to-real transfer for humanoids has also been investigated by Farchy *et al.* (*69*). However, they focused on learning the parameters of a manually designed walk engine, whereas we learned a neural network policy to output joint angles for the full soccer task directly.

Limitations

Our work provides a step toward practical use of deep RL for agile control of humanoid robots in a dynamic multiagent setting. However, there are several topics that could be addressed further. First, our learning pipeline relied on some domain-specific knowledge and domain randomization, as is common in the robot learning literature (*5*, *6*, *12*, *37*, *43*). Domain-specific knowledge was used for reward function design and for training the get-up skill, which requires access to hand-designed key poses, which can be difficult or impractical to choose for more dynamic platforms. In addition, the distillation step assumed that we could manually choose the correct skill (either get-up or soccer) for each state, although a method in which the distillation target is automatically selected has been demonstrated in prior work (*11*), which we anticipate would work in this application. Second, we did not leverage real data for transfer; instead, our approach relied 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 applied our method to a small robot and did not consider additional challenges that would be associated with a larger form factor.

Our current system could be improved in a number of ways. We found that tracking a ball with motion capture was particularly challenging: Detection of the reflective tape markers is sensitive to the angle at which they face the motion-capture cameras; only the markers on the upper hemisphere of the ball can be registered; and the walls of the soccer pitch can occlude the markers, especially near the corners. We believe that moving away from motion capture is an important avenue for future work and discuss potential avenues for this in the "Future Work" section. We also found that the performance of the robots degraded quickly over time, mainly because of the hip joints becoming loose or the joint position encoders becoming miscalibrated; thus, we needed to regularly perform robot maintenance routines. Further, our control stack was not optimized for speed. Our nominal control time step was 25 ms, but, in practice, the agent often failed to produce an action within that time. The time step was selected as a compromise between speed and consistency, but we believe that a higher control rate would result in improved performance. Last, we did not model the servo motors in simulation but instead approximated them with ideal actuators that can produce the exact torque requested by a position feedback controller. As a consequence, for example, we found that the agent's behaviors are very sensitive to the battery charge level, limiting the operation time per charge to 5 to 10 min in practice.

On the training side, we found that our self-play setup sometimes resulted in unstable learning. A population-based training scheme (*11*) could have improved stability and led to better multiagent performance. Second, our method includes several auxiliary reward terms, some of which are needed for improved transfer (for example, upright reward and knee torque penalty) and some for better exploration (for example, forward speed). We chose to use a weighted average of the different terms as the training reward and tuned the weights via an extensive hyperparameter search. However, multi-objective RL (*70*, *71*) or constrained RL (*72*) might be able to obtain better solutions.

Future work

Multiagent soccer

An exciting direction for 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 learned division of labor, a simple form of collaboration: If its teammate was closer to the ball, then the

agent did not approach the ball. However, it also learned fewer agile behaviors. Insights from prior work in simulation (*11*) could be applied to improve performance in this setting.

Playing soccer from raw vision

Another important direction for future work is learning from onboard sensors only, without external state information from a motion capture system. In comparison with state-based agents that have direct access to the ball, goal, and opponent locations, vision-based agents need to infer information from a limited history of high-dimensional egocentric camera observations and integrate the partial state information over time, which makes the problem significantly harder (*73*).

As a first step, we investigated how to train vision-based agents that only use an onboard RGB camera and proprioception. We created a visual rendering of our lab using a neural radiance field model (*74*) based on the approach introduced by Byravan *et al.* (*75*). The robot learned behaviors including ball tracking and situational awareness of the opponent and goal. See the "Playing soccer from raw vision" section in the Supplementary Materials for our preliminary results with this approach.

MATERIALS AND METHODS

Environment

We trained the agent in simulation in a custom soccer environment and then transferred to a corresponding real environment as shown in Fig. 1. The simulation environment used the MuJoCo physics engine (*76*) and was based on the DeepMind control suite (*77*). The environment consisted of a soccer pitch that was 5-m long by 4-m wide and two goals that each had an opening width of 0.8 m. In both the simulated and real environments, the pitch was bordered by ramps, which ensured that the ball returned to the bounds of the pitch. The real pitch was covered with rubber floor tiles to reduce the risk of falls damaging the robots and to increase the ground friction.

The agent acts at 40 Hz. The action is 20D and corresponds to the joint position set points of the robot. The actions were clipped to a manually selected range (see the "Environment details" section in the Supplementary Materials) and 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 were 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 consisted of proprioception and game state information. The proprioception consists of joint positions, linear acceleration, angular velocity, gravity direction, and the state of the exponential action filter. The game state information, obtained via a motion capture setup in the real environment, consisted of the agent's velocity, ball location and velocity, opponent location and velocity, and location of the two goals, which enabled the agent to infer its global position. The locations were given as 2D vectors corresponding to horizontal coordinates in the egocentric frame, and the velocities were obtained via finite differentiation from the positions. All proprioceptive observations, as well as the observation of the agent's velocity, were stacked over the five most recent time steps to account for delays and potentially noisy or missing observations. We found stacking of proprioceptive observations to be sufficient for learning high-performing policies, so we chose not to stack the game state (opponent, ball, and goal observations) to reduce the size of the observation space. However, including a full observation history could be important for improving the agent's ability to react and adapt to the opponent in more subtle ways. A more detailed description of the observations is given in the "Environment Details" section in the Supplementary Materials.

Robot hardware and motion capture

We used the Robotis OP3 robot (*31*), which is a low-cost, battery-powered, miniature humanoid platform. It is 51 cm tall, weighs 3.5 kg, and is actuated by 20 Robotis Dynamixel XM430350-R servomotors. We controlled the servos by sending target angles using position control mode with only proportional gain (in other words, without any 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 measurements. We found that the default robot control software was sometimes unreliable and caused nondeterministic control latency, so we 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 with 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.

The robot and ball positions and orientations were provided by a motion capture system based on Motive 2 software (*78*). This system uses 14 Optitrack PrimeX 22 Prime cameras mounted on a truss around the soccer pitch. We tracked the robots using reflective passive markers attached to a 3D printed "vest" covering the robot torso and tracked the ball using attached reflective stickers (Fig. 1). The positions of these three objects were streamed over the wireless network using the Virtual-Reality Peripheral Network (VRPN) protocol and made available to the robots via Robot Operating System (ROS).

We made small modifications to the robot to reduce damage from the evaluation of a wide range of prototype agent policies. 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 sometimes snagged on the robot's own cabling. We also made small mechanical modifications to the hip joints to spread the off-axis loads more evenly, to minimize fatigue breakages.

Policy optimization

We modeled the soccer environment as a partially observable Markov decision process defined by $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \mu_0, \gamma)$, with states $\boldsymbol{s} \in \mathcal{S}$, actions $\boldsymbol{a} \in \mathcal{A}$, transition probabilities $\mathcal{P}(\boldsymbol{s}' \mid \boldsymbol{s}, \boldsymbol{a})$, reward function $r(\boldsymbol{s})$, distribution over initial states μ_0, and discount factor $\gamma \in [0,1)$. At each time step t, the agent observes features $\boldsymbol{o}_t \triangleq \phi(\boldsymbol{s}_t)$, extracted from the state $\boldsymbol{s}_t \in \mathcal{S}$ as described in the "Environment" section. Actions are 20D and continuous, corresponding to the desired positions of the robot's joints. The reward is a weighted sum of K reward components, $r(\boldsymbol{s}) = \sum_{k+1}^{K} \alpha_k \hat{r}_k(\boldsymbol{s})$; the "Reward functions" section in the Supplementary Materials describes the components that we used for each stage of training.

A trajectory is defined as $\xi = \{(\boldsymbol{s}_t, \boldsymbol{a}_t, r_{t+1})\}_{t=0}^{\infty}$. A policy $\pi(\boldsymbol{a} \mid \boldsymbol{s})$, along with the system dynamics $\mathcal{P}$ and initial state distribution

μ_0, gives rise to a distribution over trajectories: $\mu_\pi(\xi) = \mu_0(\boldsymbol{s}_0) \prod_{t=0}^{\infty} \pi(\boldsymbol{a}_t | \boldsymbol{s}_t)\, \mathcal{P}(\boldsymbol{s}_{t+1} | \boldsymbol{s}_t, \boldsymbol{a}_t)$. The aim is to obtain a policy π that maximizes the expected discounted cumulative reward or return

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

We parameterized the policy as a deep feed-forward neural network with parameters θ that outputs the mean and diagonal covariance of a multivariate Gaussian. We trained this policy to optimize Eq. 1 using maximum a posteriori policy optimization (MPO) (*79*), which is an off-policy actor-critic RL algorithm. MPO alternates between policy evaluation and policy improvement. In the policy evaluation step, the critic (or Q function) is trained to estimate $Q^{\pi_\theta}(\boldsymbol{s}, \boldsymbol{a})$, which describes the expected return from taking action $\boldsymbol{a}$ in state $\boldsymbol{s}$ and then following policy π_θ: $Q^{\pi_\theta}(\boldsymbol{s}, \boldsymbol{a}) \triangleq \mathbb{E}_{\xi \sim \mu_0(\xi | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a})}[\sum_{t=0}^{T} \gamma^t r(\boldsymbol{s}_t)]$. We used a distributional critic (*80*) and refer to the overall algorithm as distributional MPO (DMPO). In the policy improvement step, the actor (or policy) is trained to improve in performance with respect to the Q values predicted by the critic. Details of the DMPO algorithm, learning hyperparameters, and the agent architecture are given in the "Agent training and architecture" section in the Supplementary Materials.

Note that the choice of opponent affects the transition probabilities P and thus the trajectory distribution μ_π. When training the soccer skill, the opponent is fixed, but when training the full 1v1 agent in the second stage, we sampled the opponent from a pool of previous snapshots of the agent, rendering the objective both nonstationary and partially observed. In practice, though, the agent was able to learn and eventually converge to a well-performing policy. Similar approaches have been explored in prior work on multiagent deep RL (*8*, *81*, *82*).

Training

Our training pipeline has two stages. This is because directly training agents on the full 1v1 task leads to suboptimal behavior, as described in the "Ablations" section. In the first stage, separate skill policies for scoring goals and getting up from the ground are trained. In the second stage, the skills are distilled into a single 1v1 agent, and the agent is trained via self-play. Distillation to skills stops after the agent's performance surpasses a preset threshold, which enabled the final behaviors to be more diverse, fluent, and robust than simply composing the skills. Self-play provides an automatic curriculum and expands the set of environment states encountered by the agent. The "Training curves" section in the Supplementary Materials contains learning curves and training times for each of the skills and the full 1v1 agent.

Stage 1: Skill training

Soccer skill training

The soccer skill 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 Fig. 1), the opponent scores, or a time limit of 50 s is reached. At the start of each episode, the players and the ball are initialized randomly 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. The reward is a weighted sum over reward components. This included components to encourage forward velocity and ball interaction and to make exploration easier as well as components to improve sim-to-real transfer and reduce robot breakages, as discussed in the "Regularization for safe behaviors" section.

Get-up skill training

The get-up skill was trained using a sequence of target poses to bias the policy toward a stable and collision-free trajectory. We used the preprogrammed get-up trajectory (*32*) to extract three key poses for getting up from either the front or the back (see the "Get-up skill training" section in the Supplementary Materials for an illustration of the key poses).

We trained the get-up skill to reach any target pose interpolated between the key poses. We conditioned both the actor and critic on the target pose, which consists of target joint angles $\mathbf{p}_{\text{target}}$ and target torso orientation $\mathbf{g}_{\text{target}}$. The target torso orientation is expressed as the gravity direction in the egocentric frame. This is independent of the robot yaw angle (heading), which is irrelevant for the get-up task. Conditioning on the joint angles steers the agent toward collision-free and stable poses, whereas conditioning on the gravity direction ensures that the robot intends to stand up rather than just matching the target joint angles while lying on the ground. The robot is initialized on the ground, and a new target pose is sampled uniformly at random every 1.5 s on average. The sampling intervals are exponentially distributed to make the probability of a target pose switch independent of time, to preserve the Markov property. The agent is trained to maximize $\hat{r}_{\text{pose}}(\boldsymbol{s}_t) = -\tilde{p}_t \tilde{g}_t$, where $\tilde{p}_t = (\pi - \| \mathbf{p}_{\text{target}} - \mathbf{p}_t \|_2)\pi$ is the scaled error in joint positions and $\tilde{g}_t = (\pi - \arccos(\mathbf{g}_t^\intercal \mathbf{g}_{\text{target}}))\pi$ is the scaled angle between the desired and actual gravity direction. $\boldsymbol{p}_t$ and $\boldsymbol{g}_t$ are the actual joint positions and gravity direction at time step t, respectively. Conditioning the converged policy on the last key pose, corresponding to standing, makes the agent get up. We used this conditioned version as a get-up skill in the next stage of training.

Stage 2: Distillation and self-play

In the second stage, the agent competes against increasingly stronger opponents while initially regularizing its behavior to the skill policies. This resulted in a single 1v1 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 skill, except that episodes terminate only when either the agent or the opponent scores or after 50 s. When the agent is on the ground, out of bounds, or in the goal penalty area, it receives a fixed penalty per time step, 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 or on its back or in a default standing pose, with equal probability.

Distillation

We used policy distillation (*83*, *84*) to enable the agent to learn from the skill policies by adding a regularization term that encourages the output of the agent's policy to be similar to that of the skills. This approach is related to prior work that regularizes a student policy to either a common shared policy across tasks (*85*) or a default policy that receives limited state information (*86*), as well as work on kickstarting (*87*) and reusing learned skills for humanoid soccer in simulation (*11*).

Unlike most prior work, in our setting, the skill policies are useful in mutually exclusive sets of states: The soccer skill is useful only when the agent is standing up; otherwise, the getup skill is more useful. Thus, in each state, we regularized the agent's policy π_θ to only one of the two skills. We achieved this by replacing the critic's predicted Q values used in the policy improvement step with a weighted sum of the predicted Q values and Kullback-Leibler (KL) regularization to the relevant skill policy:

$$\begin{cases} (1-\lambda_s)\ \mathbb{E}_{\boldsymbol{a}\sim\pi(\cdot|\boldsymbol{s})}[Q^{\pi_\theta}(\boldsymbol{s},\boldsymbol{a})] - \lambda_s \mathrm{KL}(\pi_\theta(\cdot\,|\,\mathbf{s}) \,\|\, \pi_s(\cdot\,|\,\mathbf{s})) & \text{if } \boldsymbol{s}\in\mathcal{U} \\ (1-\lambda_g)\ \mathbb{E}_{\boldsymbol{a}\sim\pi(\cdot|\boldsymbol{s})}[Q^{\pi_\theta}(\boldsymbol{s},\boldsymbol{a})] - \lambda_g \mathrm{KL}(\pi_\theta(\cdot\,|\,\mathbf{s}) \,\|\, \pi_g(\cdot\,|\,\mathbf{s})) & \text{if } \boldsymbol{s}\notin\mathcal{U} \end{cases} \tag{2}$$

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

To enable the agent to outperform the skill policies, the weights λ_s and λ_g are adaptively adjusted such that there is no regularization once the predicted Q values are above the preset thresholds Q_s and Q_g, respectively. This approach was proposed by Abdolmaleki *et al.* (*88*) for a similar setting and is closely related to the Lagrangian multiplier method used in constrained RL (*89*). Specifically, λ_s (or λ_g) is updated by stochastic gradient descent to minimize

$$c(\lambda_s) = \lambda_s(\mathbb{E}_\xi[Q^{\pi_\theta}(\mathbf{s},\mathbf{a})] - Q_s) \tag{3}$$

using a softplus transform and clipping to enforce that $0 \le \lambda_s \le 1$. When the agent's predicted return is less than Q_s, then λ_s increases to 1, at which point the agent effectively performs behavioral cloning to the soccer skill. Once the agent's predicted return surpasses Q_s, then λ_s 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 skill policies; our agents learned effective transitions between the two skills and finetuned the skills themselves.

Self-play

The performance and learned strategy of an agent depends on its opponents during training. The soccer skill plays against an untrained opponent, and, thus, this policy had limited awareness of the opponent. To improve agents' high-level gameplay, we used self-play, where the opponent is drawn from a pool of partially trained copies of the agent itself (*8*, *81*, *82*). Snapshots of the agent are regularly saved, and the first quarter of the snapshots is included in the pool, along with an untrained agent. We found that using the first quarter, rather than all snapshots, improved stability of training by ensuring that the performance of the opponent improves slowly over time. In our experiments, self-play training led to agents that were agile and defended against the opponent scoring.

Playing against a mixture of opponents results in significant partial observability because of aliasing with respect to the opponent in each episode. This can cause significant problems for critic learning, because value functions fundamentally depend on the opponent's strategy and ability (*90*). To address this, we conditioned the critic on an integer identification of the opponent.

Sim-to-real transfer

Our approach relies on zero-shot transfer of trained policies to real robots. This section details the approaches that 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 the policies via domain randomization and perturbations during training, and included shaping reward terms to obtain behaviors that are less likely to damage the robot.

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 N·m/(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. This model does not exactly correspond to the servos' operating mode, which controls the coil voltage instead of output torque, but we found that it matched the training data sufficiently well. We believe that this is because using a position control mode hides model mismatch from the agent by applying fast stabilizing feedback at a high frequency. We also experimented with direct current control but found that 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.

Domain randomization and perturbations

To further improve transfer, we applied domain randomization and random perturbations during training. Domain randomization helps overcome the remaining sim-to-real gap and the inherent variation in dynamics across robots because of wear and other factors such as battery state. We selected a small number of axes to vary because 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) and joint angular offsets (±2.9°), 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 to 50 ms) to the observations to emulate latency in the control loop. These domain randomization settings were resampled at 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 of 5 to 15 N·m, lasting for 0.05 to 0.15 s, to a randomly selected point on the torso every 1 to 3 s. Zero-shot transfer did not work for agents trained without domain randomization and perturbations: When deployed on physical robots, these agents fell over with every one or two steps and were unable to score.

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 (table S1 in the "Environment details" section in the Supplementary Materials). We also included two shaping reward terms to improve sim-to-real transfer and reduce robot breakages (see table S3 in the "Training details" in the Supplementary Materials). 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 gear breakage. We mitigated this by regularizing the policies via a penalty term to minimize the time

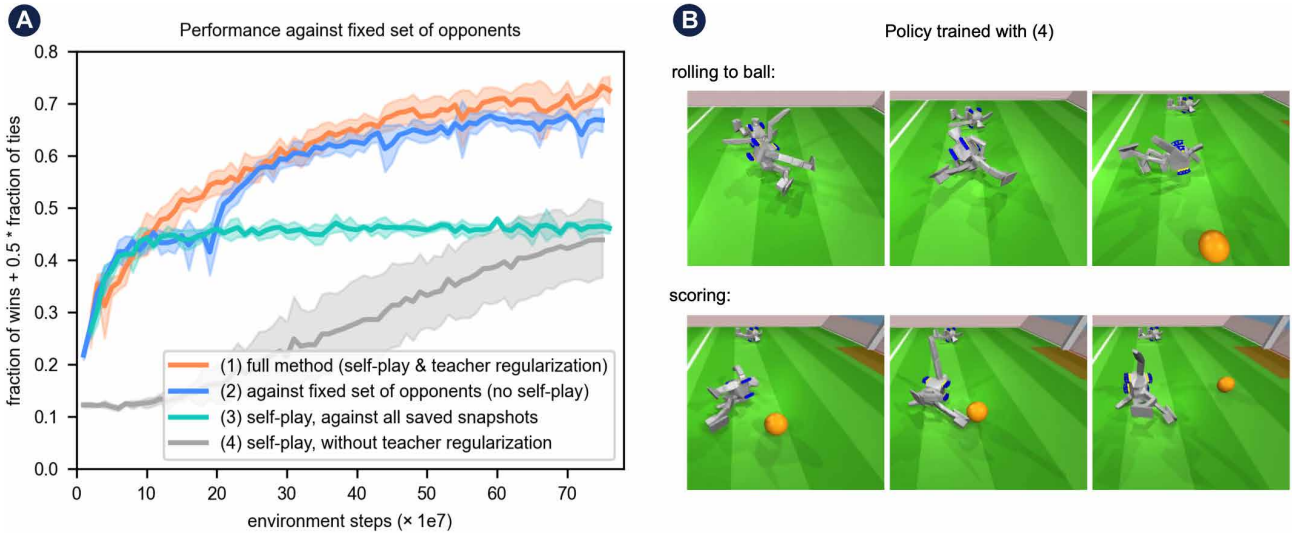


Fig. 7. Comparison against ablations. (**A**) The plot compares our full method (1) against ablations: At regular intervals across training, each policy was evaluated against a fixed set of six opponents, playing 100 matches against each. The first player to score in the episode wins the match; if neither player scores within 50 s, then it is a tie. The *y* axis corresponds to the fraction of wins (with a draw counting as one-half of a win) averaged across the six opponents. We ran five seeds per method, and the error bars show 95% confidence intervals. Our full method (1) includes both self-play and regularization to pretrained skills. In (2), instead of self-play, agents were trained against the same fixed set of six opponents as in the evaluation. Agents trained with self-play in (1) performed better when evaluated against this fixed set of opponents despite not playing directly against them during training. In (3), self-play consists of training against all saved snapshots rather than the first quarter. This led to less-stable learning and converged to worse performance. In (4), agents were trained without regularization to skills and without reward shaping. (**B**) Two sequences of frames taken from an agent trained with (4). These agents did not learn to get up from the ground; instead, they learned to score by rolling to the ball and knocking it into the goal.

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 that the agent often leaned 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 lose balance and fall forward. To mitigate this effect, 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 well at scoring goals and defending against the opponent.

Ablations

We ran ablations to investigate the importance of regularization to skill policies and using self-play, described below. We also ran ablations on the reward components (see the "Reward ablations" section in the Supplementary Materials).

Importance of regularization to skill policies

First, we trained agents without regularization to skill policies. When we gave agents a sparse reward, only for scoring or conceding goals, they learned a local optimum of rolling to the ball and knocking it into the goal with a leg (Fig. 7, right). Alternatively, with the reward shaping used in our method, with a penalty for being on the ground, agents only learned to get up and stand still. They never learned to walk around or score, despite the inclusion of shaping reward terms for walking forward and toward the ball. These results suggest that, in this setting, the exploration problem was too difficult when the agent needed to learn to both get up and play soccer. Our method overcomes this by first separately training policies for these two skills.

Importance of self-play

We also ablated the use of self-play in the second stage while keeping the skill policy regularization and the shaped reward the same. For evaluation, we played agents against a fixed set of six diverse opponents, trained with a variety of approaches; this set includes the final 1v1 agent trained with our full pipeline. Figure 7 shows a comparison of our full training method against two alternatives: training directly against the fixed set of opponents throughout learning, rather than using self-play, and using self-play but sampling opponents from all previous snapshots of the policy rather than the first quarter. The latter led to unstable learning and converged to poor performance. The former performed slightly worse than agents trained with our self-play method despite having the advantage of training directly against the opponents used for evaluation.

Supplementary Materials

This PDF file includes:

Methods
Figs. S1 to S7
Tables S1 to S5
References (*92*–*112*)

Other Supplementary Material for this manuscript includes the following:

Movies S1 to S5
MDAR Reproducibility Checklist

REFERENCES

1. K. Sims, "Evolving virtual creatures" in *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques* (ACM, 1994), pp. 15–22.
2. M. H. Raibert, *Legged Robots That Balance* (MIT Press, 1986).
3. S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, R. Tedrake, Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot. *Auton. Robots* **40**, 429–455 (2016).

4. J. Peters, S. Schaal, Reinforcement learning of motor skills with policy gradients. *Neural Netw.* **21**, 682–697 (2008).
5. M. P. Deisenroth, G. Neumann, J. Peters, "A survey on policy search for robotics" in *Foundations and Trends in Robotics*, vol. 2, no. 1–2 (Now Publishers, 2013), pp. 1–142.
6. J. Kober, J. A. Bagnell, J. Peters, Reinforcement learning in robotics: A survey. *Int. J. Robot. Res.* **32**, 1238–1274 (2013).
7. N. Heess, D. Tirumala, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, A. Eslami, M. Riedmiller, D. Silver, Emergence of locomotion behaviours in rich environments. arXiv:1707.02286 (2017).
8. T. Bansal, J. Pachocki, S. Sidor, I. Sutskever, I. Mordatch, "Emergent complexity via multi-agent competition" in *6th International Conference on Learning Representations* (ICLR, 2018).
9. X. B. Peng, P. Abbeel, S. Levine, M. van de Panne, DeepMimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transac. Graph.* **37**, 1–14 (2018).
10. J. Merel, S. Tunyasuvunakool, A. Ahuja, Y. Tassa, L. Hasenclever, V. Pham, T. Erez, G. Wayne, N. Heess, Catch & Carry: Reusable neural controllers for vision-guided whole-body tasks. *ACM Transac. Graph.* **39**, 1–14 (2020).
11. 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, N. Heess, From motor control to team play in simulated humanoid football. *Sci. Robot.* **7**, eabo0235 (2022).
12. J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, M. Hutter, Learning quadrupedal locomotion over challenging terrain. *Sci. Robot.* **5**, eabc5986 (2020).
13. S. Choi, G. Ji, J. Park, H. Kim, J. Mun, J. H. Lee, J. Hwangbo, Learning quadrupedal locomotion on deformable terrain. *Sci. Robot.* **8**, eade2256 (2023).
14. J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, M. Hutter, Learning agile and dynamic motor skills for legged robots. *Sci. Robot.* **4**, eaau5872 (2019).
15. X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, S. Levine, Learning agile robotic locomotion skills by imitating animals. arXiv:2004.00784 (2020).
16. J. Lee, J. Hwangbo, M. Hutter, Robust recovery controller for a quadrupedal robot using deep reinforcement learning. arXiv:1901.07517 (2019).
17. N. Rudin, D. Hoeller, M. Bjelonic, 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)* (IEEE, 2022), pp. 2497–2503.
18. Y. Ji, G. B. Margolis, P. Agrawal, DribbleBot: Dynamic legged manipulation in the wild. arXiv:2304.01159 (2023).
19. S. Bohez, S. Tunyasuvunakool, P. Brakel, F. Sadeghi, L. Hasenclever, Y. Tassa, E. Parisotto, J. Humplik, T. Haarnoja, R. Hafner, M. Wulfmeier, M. Neunert, B. Moran, N. Siegel, A. Huber, F. Romano, N. Batchelor, F. Casarini, J. Merel, R. Hadsell, N. Heess, Imitate and repurpose: Learning reusable robot movement skills from human and animal behaviors. arXiv:2203.17138 (2022).
20. Y. Ji, Z. Li, Y. Sun, X. B. Peng, S. Levine, G. Berseth, 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)* (IEEE, 2022), pp. 1479–1486.
21. X. Huang, Z. Li, Y. Xiang, Y. Ni, Y. Chi, Y. Li, L. Yang, X. B. Peng, K. Sreenath, Creating a dynamic quadrupedal robotic goalkeeper with reinforcement learning. arXiv:2210.04435 [cs.RO] (10 October 2022).
22. B. Forrai, T. Miki, D. Gehrig, M. Hutter, D. Scaramuzza, Event-based agile object catching with a quadrupedal robot. arXiv:2303.17479 (2023).
23. X. Cheng, A. Kumar, D. Pathak, Legs as manipulator: Pushing quadrupedal agility beyond locomotion. arXiv:2303.11330 (2023).
24. Z. Xie, P. Clary, J. Dao, P. Morais, J. W. Hurst, M. van de Panne, Iterative reinforcement learning based design of dynamic locomotion skills for Cassie. arXiv:1903.09537 [cs.RO] (22 March 2019).
25. Agility Robotics, "Cassie sets world record for 100m run," 2022; www.youtube.com/watch?v=DdojWYOK0Nc.
26. J. Siekmann, K. Green, J. Warila, A. Fern, J. Hurst, Blind bipedal stair traversal via sim-to-real reinforcement learning. arXiv:2105.08328 (2021).
27. Z. Li, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, K. Sreenath, Robust and versatile bipedal jumping control through multi-task reinforcement learning. arXiv:2302.09450 [cs.RO] (1 June 2023).
28. R. Deits T. Koolen, "Picking up momentum," Boston Dynamics, January 2023; www.bostondynamics.com/resources/blog/picking-momentum.
29. H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawa, "RoboCup: The robot world cup initiative" *in Proceedings of the First International Conference on Autonomous Agents* (ACM, 1997), pp. 340–347.
30. RoboCup Federation, "Robocup project," May 2022; https://robocup.org.
31. Robotis, "Robotis OP3 manual," March 2023; https://emanual.robotis.com/docs/en/platform/op3/introduction.
32. Robotis, "Robotis OP3 source code," April 2023; https://github.com/ROBOTIS-GIT/ROBOTIS-OP3.
33. M. Bestmann J. Zhang, "Bipedal walking on humanoid robots through parameter optimization" in *RoboCup 2022: Robot World Cup XXV*, vol. 13561 of *Lecture Notes in Computer Science*, A. Eguchi, N. Lau, M. Paetzel-Prüsmann, T. Wanichanon, Eds. (Springer, 2022), pp. 164–176.
34. B. D. DeAngelis, J. A. Zavatone-Veth, D. A. Clark, The manifold structure of limb coordination in walking *Drosophila*. *eLife* **8**, e46409 (2019).
35. L. McInnes, J. Healy, J. Melville, UMAP: Uniform manifold approximation and projection for dimension reduction. arXiv:1802.03426 (February 2018).
36. T. Röfer, T. Laue, A. Baude, J. Blumenkamp, G. Felsch, J. Fiedler, A. Hasselbring, T. Haß, J. Oppermann, P. Reichenberg, N. Schrader, D. Weiß, "B-Human team report and code release 2019," 2019; http://b-human.de/downloads/publications/2019/CodeRelease2019.pdf.
37. J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, S. Levine, How to train your robot with deep reinforcement learning: Lessons we have learned. *Int. J. Robot. Res.* **40**, 698–721 (2021).
38. T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, M. Hutter, Learning robust perceptive locomotion for quadrupedal robots in the wild. *Sci. Robot.* **7**, eabk2822 (2022).
39. A. Agarwal, A. Kumar, J. Malik, D. Pathak, "Legged locomotion in challenging terrains using egocentric vision" in *Conference on Robot Learning* (MLResearchPress, 2023), pp. 403–415.
40. I. Radosavovic, T. Xiao, B. Zhang, T. Darrell, J. Malik, K. Sreenath, Learning humanoid locomotion with transformers. arXiv:2303.03381 [cs.RO] (14 December 2023).
41. A. Kumar, Z. Fu, D. Pathak, J. Malik, RMA: Rapid motor adaptation for legged robots. arXiv:2107.04034 (2021).
42. L. Smith, J. C. Kew, T. Li, L. Luu, X. B. Peng, S. Ha, J. Tan, S. Levine, Learning and adapting agile locomotion skills by transferring experience. arXiv:2304.09834 (2023).
43. F. Muratore, F. Ramos, G. Turk, W. Yu, M. Gienger, J. Peters, Robot learning from randomized simulations: A review. *Front. Robot. AI* **9**, 799893 (2022).
44. P. Wu, A. Escontrela, D. Hafner, P. Abbeel, K. Goldberg, "DayDreamer: World models for physical robot learning" *in Conference on Robot Learning* (MLResearchPress, 2023), pp. 2226–2240.
45. T. Haarnoja, S. Ha, A. Zhou, J. Tan, G. Tucker, S. Levine, "Learning to walk via deep reinforcement learning" in *Proceedings of Robotics: Science and Systems (RSS)*, A. Bicchi, H. Kress-Gazit, S. Hutchinson, Eds. (RSS, 2019).
46. S. Ha, P. Xu, Z. Tan, S. Levine, J. Tan, "Learning to walk in the real world with minimal human effort" in *Conference on Robot Learning* (MLResearchPress, 2021), pp. 1110–1120.
47. L. Smith, I. Kostrikov, S. Levine, A walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning. arXiv:2208.07860 (2022).
48. M. Bloesch, J. Humplik, V. Patraucean, R. Hafner, T. Haarnoja, A. Byravan, N. Y. Siegel, S. Tunyasuvunakool, F. Casarini, N. Batchelor, F. Romano, S. Saliceti, M. Riedmiller, S. M. A. Eslami, N. Heess, "Towards real robot learning in the wild: A case study in bipedal locomotion" in *Conference on Robot Learning* (MLResearchPress, 2022), pp. 1502–1511.
49. G. Ji, J. Mun, H. Kim, J. Hwangbo, Concurrent training of a control policy and a state estimator for dynamic and robust legged locomotion. *IEEE Robot. Autom. Lett.* **7**, 4630–4637 (2022).
50. G. B. Margolis, G. Yang, K. Paigwar, T. Chen, P. Agrawal, Rapid locomotion via reinforcement learning. arXiv:2205.02824 (2022).
51. Y. Jin, X. Liu, Y. Shao, H. Wang, W. Yang, High-speed quadrupedal locomotion by imitation-relaxation reinforcement learning. *Nat. Mach. Intell.* **4**, 1198–1208 (2022).
52. I. Mordatch, K. Lowrey, E. Todorov, "Ensemble-CIO: Full-body dynamic motion planning that transfers to physical humanoids" in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2015), pp. 5307–5314.
53. W. Yu, V. C. Kumar, G. Turk, C. K. Liu, "Sim-to-real transfer for biped locomotion" in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2019), pp. 3503–3510.
54. S. Masuda and K. Takahashi, Sim-to-real learning of robust compliant bipedal locomotion on torque sensor-less gear-driven humanoid. arXiv:2204.03897 (2022).
55. Y. Ma, F. Farshidian, M. Hutter, Learning arm-assisted fall damage reduction and recovery for legged mobile manipulators. arXiv:2303.05486 (2023).
56. O. Nachum, M. Ahn, H. Ponte, S. Gu, V. Kumar, Multi-agent manipulation via locomotion using hierarchical sim2real. arXiv:1908.05224 (2019).
57. M. Riedmiller, A. Merke, D. Meier, A. Hoffmann, A. Sinner, O. Thate, R. Ehrmann, "Karlsruhe Brainstormers a reinforcement learning approach to robotic soccer" in *RoboCup-2000: Robot Soccer World Cup IV*, vol. 2019 of *Lecture Notes in Computer Science*, P. Stone, T. Balch, G. Kraetzschmar, Eds. (Springer, 2000), pp. 367–372.
58. K. Tuyls, S. Maes, B. Manderick, "Reinforcement learning in large state spaces" in *RoboCup 2002: Robot Soccer World Cup VI*, vol. 2752 of *Lecture Notes in Computer Science*, G. A. Kaminka, P. U. Lima, R. Rojas, Eds. (Springer, 2002), pp. 319–326.
59. M. Riedmiller, T. Gabel, R. Hafner, S. Lange, Reinforcement learning for robot soccer. *Auton. Robots.* **27**, 55–73 (2009).

60. P. Stone, R. S. Sutton, G. Kuhlmann, Reinforcement learning for RoboCup-soccer keepaway. *Adapt. Behav.* **13**, 165–188 (2005).
61. S. Kalyanakrishnan P. Stone, "Learning complementary multiagent behaviors: A case study" in *RoboCup 2009: Robot Soccer World Cup XIII*, vol. 5949 of *Lecture Notes in Computer Science*, J. Baltes, M. G. Lagoudakis, T. Naruse, S. S. Ghidary, Eds. (Springer, 2010), pp. 153–165.
62. S. Kalyanakrishnan, Y. Liu, P. Stone, "Half field offense in RoboCup soccer: A multiagent reinforcement learning case study" in *RoboCup-2006: Robot Soccer World Cup X*, vol. 4434 of *Lecture Notes in Artificial Intelligence*, G. Lakemeyer, E. Sklar, D. Sorenti, T. Takahashi, Eds. (Springer, 2007), pp. 72–85.
63. P. Stone M. Veloso, "Layered learning" in *European Conference on Machine Learning* (Springer, 2000), pp. 369–381.
64. P. MacAlpine, P. Stone, Overlapping layered learning. *Artif. Intell.* **254**, 21–43 (2018).
65. M. Abreu, L. P. Reis, N. Lau, "Learning to run faster in a humanoid robot soccer environment through reinforcement learning" in *Robot World Cup* (Springer, 2019) pp. 3–15.
66. L. C. Melo, D. C. Melo, M. R. Maximo, Learning humanoid robot running motions with symmetry incentive through proximal policy optimization. *J. Intell. Robot. Syst.* **102**, 54 (2021).
67. M. Saggar, T. D'Silva, N. Kohl, P. Stone, "Autonomous learning of stable quadruped locomotion" in *RoboCup-2006: Robot Soccer World Cup X*, vol. 4434 of *Lecture Notes in Artificial Intelligence*, G. Lakemeyer, E. Sklar, D. Sorenti, T. Takahashi, Eds. (Springer, 2007), pp. 98–109.
68. M. Hausknecht, P. Stone, "Learning powerful kicks on the Aibo ERS-7: The quest for a striker" in *RoboCup-2010: Robot Soccer World Cup XIV*, vol. 6556 of *Lecture Notes in Artificial Intelligence*, J. R. del Solar, E. Chown, P. G. Plöger, Eds. (Springer, 2011), pp. 254–65.
69. A. Farchy, S. Barrett, P. MacAlpine, P. Stone, "Humanoid robots learning to walk faster: From the real world to simulation and back" in *Proceedings of 12th International Conference on Autonomous Agents and Multiagent Systems* (AAMAS, 2013), pp. 39–46.
70. D. M. Roijers, P. Vamplew, S. Whiteson, R. Dazeley, A survey of multi-objective sequential decision-making. *J. Artif. Intell. Res.* **48**, 67–113 (2013).
71. A. Abdolmaleki, S. Huang, L. Hasenclever, M. Neunert, F. Song, M. Zambelli, M. Martins, N. Heess, R. Hadsell, M. Riedmiller, "A distributional view on multi-objective policy optimization" in *International Conference on Machine Learning* (MLResearchPress, 2020), pp. 11–22.
72. A. Ray, J. Achiam, D. Amodei, Benchmarking safe exploration in deep reinforcement learning. arXiv:2310.03225 (2019).
73. 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, M. A. Riedmiller, Deepmind control suite. arXiv:1801.00690 [cs.AI] (2 January 2018).
74. B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, R. Ng, NeRF: Representing scenes as neural radiance fields for view synthesis. *Commun. ACM* **65**, 99–106 (2022).
75. A. Byravan, J. Humplik, L. Hasenclever, A. Brussee, F. Nori, T. Haarnoja, B. Moran, S. Bohez, F. Sadeghi, B. Vujatovic, 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)* (IEEE, 2023), pp. 9362–9369.
76. E. Todorov, T. Erez, Y. Tassa, "Mujoco: A physics engine for model-based control" in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE, 2012) pp. 5026–5033.
77. S. Tunyasuvunakool, A. Muldal, Y. Doron, S. Liu, S. Bohez, J. Merel, T. Erez, T. Lillicrap, N. Heess, Y. Tassa, Dm control: Software and tasks for continuous control. *Software Impacts* **6**, 100022 (2020).
78. Optitrack, "Motive optical motion capture software," March 2023; https://optitrack.com/.
79. A. Abdolmaleki, J. T. Springenberg, Y. Tassa, R. Munos, N. Heess, M. Riedmiller, "Maximum a posteriori policy optimisation" in *Proceedings of the 6th International Conference on Learning Representations* (ICLR, 2018).
80. M. G. Bellemare, W. Dabney, R. Munos, "A distributional perspective on reinforcement learning" in *Proceedings of the 34th International Conference on Machine Learning* (ACM, 2017), pp. 449–458.
81. J. Heinrich, M. Lanctot, D. Silver, "Fictitious self-play in extensive-form games" in *Proceedings of the 32nd International Conference on Machine Learning*, vol. 37 of *JMLR Workshop and Conference Proceedings*, F. R. Bach, D. M. Blei, Eds. (ACM, 2015), pp. 805–813.
82. M. Lanctot, V. Zambaldi, A. Gruslys, A. Lazaridou, K. Tuyls, J. Perolat, D. Silver, T. Graepel, A unified game-theoretic approach to multiagent reinforcement learning. *Adv. Neural Inf. Process. Syst.* **30**, 4190–4203 (2017).
83. A. A. Rusu, S. G. Colmenarejo, C. Gulcehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, R. Hadsell, Policy distillation. arXiv:1511.06295 (2015).
84. E. Parisotto, J. L. Ba, R. Salakhutdinov, Actor-mimic: Deep multitask and transfer reinforcement learning. arXiv:1511.06342 (2015).
85. Y. Teh, V. Bapst, W. M. Czarnecki, J. Quan, J. Kirkpatrick, R. Hadsell, N. Heess, R. Pascanu, Distral: Robust multitask reinforcement learning. *Adv. Neural Inf. Process. Syst.* **30** (2017).
86. A. Galashov, S. Jayakumar, L. Hasenclever, D. Tirumala, J. Schwarz, G. Desjardins, W. M. Czarnecki, Y. W. Teh, R. Pascanu, N. Heess, "Information asymmetry in KLregularized RL" in *International Conference on Learning Representations*, New Orleans, LA, 6 to 9 May 2019.
87. S. Schmitt, J. J. Hudson, A. Z'ıdek, S. Osindero, C. Doersch, W. M. Czarnecki, J. Z. Leibo, H. Küttler, A. Zisserman, K. Simonyan, S. M. A. Eslami, Kickstarting deep reinforcement learning. arXiv:1803.03835 (2018).
88. 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, M. Riedmiller, On multi-objective policy optimization as a tool for reinforcement learning. arXiv:2106.08199 (2021).
89. A. Stooke, J. Achiam, P. Abbeel, "Responsive safety in reinforcement learning by pid lagrangian methods" in *Proceedings of the 37th International Conference on Machine Learning* (ICML, 2020), pp. 9133–9143.
90. S. Liu, G. Lever, J. Merel, S. Tunyasuvunakool, N. Heess, T. Graepel, "Emergent coordination through competition" in *International Conference on Learning Representations*, New Orleans, LA, 6 to 9 May 2019.
91. S. Thrun A. Schwartz, Finding structure in reinforcement learning. *Adv. Neural Inf. Process. Syst.* **7**, (1994).
92. M. Bowling, M. Veloso, "Reusing learned policies between similar problems" in *Proceedings of the AI* AI-98 Workshop on New Trends in Robotics* (1998); https://cs.cmu.edu/afs/cs/user/mmv/www/papers/rl-reuse.pdf.
93. X. B. Peng, M. Chang, G. Zhang, P. Abbeel, S. Levine, "MCP: learning composable hierarchical control with multiplicative compositional policies" in *Advances in Neural Information Processing Systems*, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'AlchéBuc, E. B. Fox, R. Garnett, Eds. (MIT Press, 2019), pp. 3681–3692.
94. M. Wulfmeier, D. Rao, R. Hafner, T. Lampe, A. Abdolmaleki, T. Hertweck, M. Neunert, D. Tirumala, N. Siegel, N. Heess, M. Riemiller, "Data-efficient hindsight off-policy option learning" in *International Conference on Machine Learning* (MLResearchPress, 2021), pp. 11340–11350.
95. J. Won, D. Gopinath, J. K. Hodgins, Control strategies for physically simulated characters performing two-player competitive sports. *ACM Trans. Graph.* **40**, 1–11 (2021).
96. R. S. Sutton, D. Precup, S. Singh, Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artif. Intell.* **112**, 181–211 (1999).
97. S. Salter, M. Wulfmeier, D. Tirumala, N. Heess, M. Riedmiller, R. Hadsell, D. Rao, "Mo2: Model-based offline options" in *Conference on Lifelong Learning Agents* (MLResearchPress, 2022), pp. 902–919.
98. S. Ross, G. Gordon, 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* (AISTATS, 2011), pp. 627–635.
99. 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. *J. Mach. Learn. Res.* **23**, 9989–10056 (2022).
100. M. Riedmiller, R. Hafner, T. Lampe, M. Neunert, J. Degrave, T. van de Wiele, V. Mnih, N. Heess, J. T. Springenberg, "Learning by playing solving sparse reward tasks from scratch" in *Proceedings of the 35th International Conference on Machine Learning* (ACM, 2018) pp. 4344–4353.
101. G. Vezzani, D. Tirumala, M. Wulfmeier, D. Rao, A. Abdolmaleki, B. Moran, T. Haarnoja, J. Humplik, R. Hafner, M. Neunert, C. Fantacci, T. Hertweck, T. Lampe, F. Sadeghi, N. Heess, M. Riedmiller, Skills: Adaptive skill sequencing for efficient temporally-extended exploration. arXiv:2211.13743 (2022).
102. 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, L. M. Zhang, Human-timescale adaptation in an open-ended task space. arXiv:2301.07608 (2023).
103. R. Hafner, T. Hertweck, P. Klöppner, M. Bloesch, M. Neunert, M. Wulfmeier, S. Tunyasuvunakool, N. Heess, M. Riedmiller, "Towards general and autonomous learning of core skills: A case study in locomotion" in *Conference on Robot Learning* (MLResearchPress, 2021), pp. 1084–1099.
104. M. Wulfmeier, A. Abdolmaleki, R. Hafner, J. T. Springenberg, M. Neunert, T. Hertweck, T. Lampe, N. Siegel, N. Heess, M. Riedmiller, Compositional transfer in hierarchical reinforcement learning. arXiv:1906.11228 (2019).
105. D. Balduzzi, M. Garnelo, Y. Bachrach, W. Czarnecki, J. Pérolat, M. Jaderberg, T. Graepel, "Open-ended learning in symmetric zero-sum games" in *Proceedings of the 36th International Conference on Machine Learning (ICML)*, vol. 97 of *Proceedings of Machine Learning Research*, K. Chaudhuri, R. Salakhutdinov, Eds. (MLResearchPress, 2019), pp. 434–443.
106. G. W. Brown, "Iterative solution of games by fictitious play" in *Activity Analysis of Production and Allocation*, T. C. Koopmans, Ed. (Wiley, 1951).

107. 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, C. Gulcehre, Z. Wang, T. Pfaff, Y. Wu, R. Ring, D. Yogatama, D. Wunsch, K. McKinney, O. Smith, T. Schaul, T. P. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps, D. Silver, Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* **575**, 350–354 (2019).
108. B. Baker, I. Kanitscheider, T. Markov, Y. Wu, G. Powell, B. McGrew, I. Mordatch, "Emergent tool use from multi-agent autocurricula" in *8th International Conference on Learning Representations* (ICLR, 2020).
109. R. S. Sutton, A. G. Barto, *Reinforcement Learning: An Introduction* (MIT Press, 2018).
110. J. Schulman, S. Levine, P. Abbeel, M. Jordan, P. Moritz, "Trust region policy optimization" in *Proceedings of the 32nd International Conference on Machine Learning (ICML)* (ACM, 2015), pp. 1889–1897.
111. V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, Human-level control through deep reinforcement learning. *Nature* **518**, 529 (2015).
112. T. Haarnoja, B. Moran, G. Lever, S. H. Huang, D. Tirumala, J. Humplik, M. Wulfmeier, S. Tunyasuvunakool, N. Y. Siegel, R. Hafner, M. Bloesch, K. Hartikainen, A. Byravan, L. Hasenclever, T. Y., F. Sadeghi, N. Batchelor, F. Casarini, S. Saliceti, C. Game, N. Sreendra, K. Patel, M. Gwira, A. Huber, N. Hurley, F. Nori, R. Hadsell, N. Heess, Data release for: Learning agile soccer skills for a bipedal robot with deep reinforcement learning [data set], 2024; https://doi.org/10.5281/zenodo.10793725.

Acknowledgements: We thank D. Hennes at Google DeepMind for developing the plotting tools used for the soccer matches and M. Riedmiller and M. Neunert at Google DeepMind for their helpful comments. **Funding:** This research was funded by Google DeepMind. **Author contributions:** Algorithm development: T.H., B.M., G.L., S.H.H., D.T., J.H., M.W., S.T., N.Y.S., R.H., M.B., K.H., A.B., L.H., Y.T., F.S.; Software infrastructure and environment development: T.H., B.M., G.L., S.H.H., J.H., S.T., N.Y.S., R.H., M.B., Y.T.; Agent analysis: T.H., B.M., G.L., S.H.H.; Experimentation: T.H., B.M., G.L., S.H.H., D.T., J.H., N.Y.S.; Article writing: T.H., B.M., G.L., S.H.H., D.T., M.W., N. Heess; Infrastructure support: N.B., F.C., S.S., C.G., N.S., K.P., M.G.; Management support: N.B., F.C., A.H., N. Hurley; Project supervision: F.N., R.H., N. Heess; Project design: T.H., N. Heess. **Data availability:** The data used for our quantitative figures and tables have been made available for download at https://zenodo.org/records/10793725 (*91*).

Submitted 31 May 2023
Accepted 14 March 2024
Published 10 April 2024
10.1126/scirobotics.adi8022

Science Robotics

Learning agile soccer skills for a bipedal robot with deep reinforcement learning

Tuomas Haarnoja, Ben Moran, Guy Lever, Sandy H. Huang, Dhruva Tirumala, Jan Humplik, Markus Wulfmeier, Saran Tunyasuvunakool, Noah Y. Siegel, Roland Hafner, Michael Bloesch, Kristian Hartikainen, Arunkumar Byravan, Leonard Hasenclever, Yuval Tassa, Fereshteh Sadeghi, Nathan Batchelor, Federico Casarini, Stefano Saliceti, Charles Game, Neil Sreendra, Kushal Patel, Marlon Gwira, Andrea Huber, Nicole Hurley, Francesco Nori, Raia Hadsell, and Nicolas Heess

Sci. Robot. **9** (89), eadi8022. DOI: 10.1126/scirobotics.adi8022

Editor's summary

Generating robust motor skills in bipedal robots in the real world is challenging because of the inability of current control methods to generalize to specific tasks. Haarnoja *et al.* developed a deep reinforcement learning–based framework for full-body control of humanoid robots, enabling a game of one-versus-one soccer. The robots exhibited emergent behaviors in the form of dynamic motor skills such as the ability to recover from falls and also tactics like defending the ball against an opponent. The robot movements were faster when using their framework than a scripted baseline controller and may have potential for more complex multirobot interactions. —Amos Matsiko