Human-level performance in 3D multiplayer games with population-based reinforcement learning


Reinforcement learning (RL) has shown great success in increasingly complex single-agent environments and two-player turn-based games. However, the real world contains multiple agents, each learning and acting independently to cooperate and compete with other agents. We used a tournament-style evaluation to demonstrate that an agent can achieve human-level performance in a three-dimensional multiplayer first-person video game, Quake III Arena in Capture the Flag mode, using only pixels and game points scored as input. We used a two-tier optimization process in which a population of independent RL agents are trained concurrently from thousands of parallel matches on randomly generated environments. Each agent learns its own internal reward signal and rich representation of the world. These results indicate the great potential of multiagent reinforcement learning for artificial intelligence research.

End-to-end reinforcement learning (RL) methods (1–5) have so far not succeeded in training agents in multiagent games that combine team and competitive play owing to the high complexity of the learning problem that arises from the concurrent adaptation of multiple learning agents in the environment (6, 7). We approached this challenge by studying team-based multiplayer three-dimensional (3D) first-person video games, a genre that is particularly immersive for humans (8) and has even been shown to improve a wide range of cognitive abilities (9). We focused specifically on a modified version (10) of Quake III Arena (11), the canonical multiplayer 3D first-person video game, whose game mechanics served as the basis for many subsequent games and which has a thriving professional scene (12).

The task we considered is the game mode Capture the Flag (CTF), which is played on both indoor- and outdoor-themed maps that are randomly generated for each game (Fig. 1A and B). Two opposing teams consisting of multiple individual players compete to capture each other’s flags by strategically navigating, tagging, and evading opponents. The team with the greatest number of flag captures after five minutes wins. The opposing teams’ flags are situated at opposite ends of each map—a team’s base—and in indoor-themed maps, the base room is colored according to the team color. In addition to moving through the environment, agents can tag opponents by activating their laser gadget when pointed at an opponent, which sends the opponent back to their base room after a short delay, known as respawning. If an agent is holding a flag when they are tagged, this flag is dropped to the floor where they are tagged and is said to be stray. CTF is played in a visually rich simulated physical environment (movie S1), and agents interact with the environment and with other agents only through their observations and actions (moving forward and backward; strafing left and right; and looking by rotating, jumping, and tagging). In contrast to previous work (13–23), agents do not have access to models of the environment, state of other players, or human policy priors, nor can they communicate with each other outside of the game environment. Each agent acts and learns independently, resulting in decentralized control within a team.

Learning system

We aimed to devise an algorithm and training procedure that enables agents to acquire policies that are robust to the variability of maps, number of players, and choice of teammates and opponents, a challenge that generalizes that of ad hoc teamwork (24). In contrast to previous work (25), the proposed method is based purely on end-to-end learning and generalization. The proposed training algorithm stabilizes the learning process in partially observable multiagent environments by concurrently training a diverse population of agents who learn by playing with each other. In addition, the agent population provides a mechanism for meta-optimization.

In our formulation, the agent’s policy \( \pi \) uses the same interface available to human players. It receives raw red-green-blue (RGB) pixel input \( \mathbf{x}_t \) from the agent’s first-person perspective at time step \( t \), produces control actions \( \mathbf{a}_t \sim \pi(\mathbf{x}_t, ..., \mathbf{x}_0) \) by sampling from the distribution given by policy \( \pi \), and receives \( \rho_t \) game points, which are visible on the in-game scoreboard. The goal of RL in this context is to find a policy that maximizes the expected cumulative reward \( \mathbb{E}_\pi \left[ \sum_{t=1}^{T} r_t \right] \) over a CTF game with \( T \) time steps. We used a multistep actor-critic policy gradient algorithm (2) with off-policy correction (26) and auxiliary tasks (3) for RL. The agent’s policy \( \pi \) was parameterized by means of a multi-time-scale recurrent neural network with external memory (Fig. 2A and fig. S11) (27). Actions in this model were generated conditional on a stochastic latent variable, whose distribution was modulated by a more slowly evolving prior process. The variational objective function encodes a trade-off between maximizing expected reward and consistency between the two time scales of inference (28).

For ad hoc teams, we postulated that an agent’s policy \( \pi_1 \) should maximize the probability \( \mathbb{P}(\pi_1’s \text{ team wins} | \omega, \pi_2, N) \) of winning for its team, \( \pi_1, \pi_2, ..., \pi_N \), which is composed of \( \pi_1 \) itself, and its teammates’ policies \( \pi_2, ..., \pi_N \), for a total of \( N \) players in the game

\[
\mathbb{P}(\pi_1’s \text{ team wins} | \omega, \pi_2, ..., \pi_N) = \mathbb{E}_{\pi_2, ..., \pi_N} \mathbb{E}_{\omega} \mathbb{P}(\pi_1 \text{ wins} | \omega, \pi_2, ..., \pi_N)
\]

wherein trajectories \( \tau \) (sequences of actions, states, and rewards) are sampled from the joint probability distribution \( \mathbb{P}_\omega(\mathcal{S}, \mathcal{A}) \) over game setup \( \omega \) and actions sampled from policies. The operator \( I[\pi] \) returns 1 if and only if \( \pi \) is true, and \( \mathbb{P}(\tau, \pi) \) returns the number of flag captures obtained by agents in \( \pi \) in trajectory \( \tau \). Ties are broken by \( \epsilon \), which is sampled from an independent Bernoulli distribution with probability 0.5. The distribution \( \Omega \) over specific game setups is defined over the Cartesian product of the set of maps and the set of random seeds. During learning and testing, each game setup \( \omega \) is sampled from \( \Omega \), and \( \omega \) over the final game outcome is too sparse to be effectively used as the sole reward signal for RL, and so we learn rewards \( r_t \) to direct the learning process toward winning; these are more frequently available than the game outcome. In our approach, we operationalized the idea that each agent has a dense internal reward function (39–41) by specifying \( r_t = \mathbf{w}(\rho_t, \text{points which are registered for events such as capturing a flag}) \) and, crucially, allowing the agent to learn the transformation \( \mathbf{w} \) so that
policy optimization on the internal rewards $r_i$ optimizes the policy “For The Win,” giving us the “FTW agent.”

Training agents in multiagent systems requires instantiations of other agents in the environment, such as teammates and opponents, to generate learning experience. A solution could be self-play RL, in which an agent is trained by playing against its own policy. Although self-play variants can prove effective in some multiagent games (14, 15, 42–46), these methods can be unstable and in their basic form do not support concurrent training, which is crucial for scalability. Our solution is to train in parallel a population of $P$ different agents $\pi = (\pi_p)_{p=1}^P$ that play with each other, introducing diversity among players in order to stabilize training (47). Each agent within this population learns from experience generated by playing with teammates and opponents sampled from the population. We sampled the agents indexed by $p$ for a training game by using a stochastic matchmaking scheme $m_p(\pi)$ that biases co-players to be of similar skill to player $p$. This scheme ensures that—a priori—the outcome is sufficiently uncertain to provide a meaningful learning signal and that a diverse set of teammates and opponents participate in training. Agents’ skill levels were estimated online by calculating Elo scores [adapted from chess (48)] on the basis of outcomes of training games. We also used the population to meta-optimize the internal rewards and hyperparameters of the RL process itself, which results in the joint maximization of

$$J_{\text{inner}}(\pi_p | w_p) = \mathbb{E}_{i \sim m_p(\pi)} - \omega \mathbb{E}_{p \sim P}$$

$$J_{\text{outer}}(w_p, \phi_p | \pi) = \mathbb{E}_{i \sim m_p(\pi), o \sim \Omega} P(\pi_p^w, o | \pi_p^w, o)$$

where

$$\pi_p^w = \text{optimize}_{\phi_p}(J_{\text{inner}}, w, \phi)$$

This can be seen as a two-tier RL problem. The inner optimization maximizes $J_{\text{inner}}$, the agents’ expected future discounted internal rewards. The outer optimization of $J_{\text{outer}}$ can be viewed as a meta-game, in which the meta-reward of winning the match is maximized with respect to internal reward schemes $w_p$ and hyperparameters $\phi_p$, with the inner optimization providing the meta transition dynamics. We solved the inner optimization with RL as previously described, and the outer optimization with population-based training (PBT) (49). PBT is an online evolutionary process that adapts internal rewards and hyperparameters and performs model selection by replacing underperforming agents with mutated versions of better agents. This joint optimization of the agent policy by using RL together with the optimization of the RL procedure itself toward a high-level goal proves to be an effective and potentially widely applicable strategy and uses the potential of combining learning and evolution (50) in large-scale learning systems.

**Tournament evaluation**

To assess the generalization performance of agents at different points during training, we performed a large tournament on procedurally generated maps with ad hoc matches that involved three types of agents as teammates and opponents: ablated versions of FTW (including state-of-the-art baselines), Quake III Arena scripted bots of various levels (51), and human participants with first-person video game experience. The Elo scores and derived winning probabilities for different ablations of FTW, and how the combination of components provide superior performance, are shown in Fig. 2B and fig. S1.

![Fig. 1. CTF task and computational training framework.](image-url)
The FTW agents clearly exceeded the win-rate of humans in maps that neither agent nor human had seen previously—that is, zero-shot generalization—with a team of two humans on average capturing 16 fewer flags per game than a team of two FTW agents (fig. S1, bottom, FF versus hh). Only as part of a human-agent team did we observe a human winning over an agent-agent team (5% win probability). This result suggests that trained agents are capable of cooperating with never-seen-before teammates, such as humans. In a separate study, we probed the exploitability of the FTW agent by allowing a team of two professional games testers with full control resolution; this may be responsible for the subtle differences in observation between agents and humans must take into account the intrinsic limitations in reaction time and sensorimotor skills (fig. S10A) ([28], section 3.1). For example, humans have superior observation and control resolution; this may be responsible for humans successfully tagging at long range where agents could not (humans, 17% tags above 5 map units; agents, 0.5%). By contrast, at short range, agents have superior tagging reaction times to humans: By one measure, FTW agents respond to newly appeared opponents with a mean of 258 ms, compared with 559 ms for humans (fig. S10B). Another advantage exhibited by agents is their tagging accuracy, in which FTW agents achieve 80% accuracy compared with humans’ 48%. By artificially reducing the FTW agents’ tagging accuracy to be similar to humans (without retraining them), the agents’ win rate was reduced though still exceeded that of humans (fig. S10C). Thus, although agents learned to make use of their potential for better tagging accuracy, this is only one factor contributing to their overall performance.

To explicitly investigate the effect of the native superiority in the reaction time of agents compared with that of humans, we introduced an artificial 267-ms reaction delay to the FTW agent (in line with the previously reported discrepancies, and corresponding to fast human reaction times in simple psychophysical paradigms) ([28–34]). This response-delayed FTW agent was fine-tuned from the nondelayed FTW agent through a combination of RL and distillation across the agent population: learning rate, Kullback-Leibler divergence (KL) weighting, and internal time scale $\tau$, plotted as mean and standard deviation across the population.

**Agent analysis**

We hypothesized that trained agents of such high skill have learned a rich representation of the game. To investigate this, we extracted ground-truth state from the game engine at each point in time in terms of 200 binary features such as “Do I have the flag?”, “Did I see my teammate recently?”, and “Will I be in the opponent’s base soon?” We say that the agent has knowledge of a given feature if logistic regression on the internal state of the agent accurately models the feature. In this sense, the internal representation of the agent was found to encode a wide variety of knowledge about the game situation (fig. S4). The FTW agent’s representation was found to encode features related to the past particularly

---

Fig. 2. Agent architecture and benchmarking. (A) How the agent processes a temporal sequence of observations $x_t$ from the environment. The model operates at two different time scales, faster at the bottom and slower by a factor of $\tau$ at the top. A stochastic vector-valued latent variable is sampled at the fast time scale from distribution $Q_{\theta}$ on the basis of observations $x_t$. The action distribution $\pi_{\phi}$ is sampled conditional on the latent variable at each time step $t$. The latent variable is regularized by the slow moving prior $P_{\psi}$, which helps capture long-range temporal correlations and promotes memory. The network parameters are updated by using RL according to the agent’s own internal reward signal $r_{\phi}$, which is obtained from a learned transformation $w$ of game points $p_w$. $w$ is optimized for winning probability through PBT, another level of training performed at yet a slower time scale than that of RL. Detailed network architectures are described in fig. S11. (B) (Top) The Elo skill ratings of the FTW agent population throughout training (blue) together with those of the best baseline agents by using hand-tuned reward shaping (RS) (red) and game-winning reward signal only (black), compared with human and random agent reference points (violet, shaded region shows strength between 10th and 90th percentile). The FTW agent achieves a skill level considerably beyond strong human subjects, whereas the baseline agent’s skill plateaus below and does not learn anything without reward shaping [evaluation procedure is provided in (28)]. (Bottom) The evolution of three hyperparameters of the FTW agent population: learning rate, Kulback-Leibler divergence (KL) weighting, and internal time scale $\tau$, plotted as mean and standard deviation across the population.

---

well; for example, the FTW agent was able to classify the state “both flags are stray” (flags dropped not at base) with 91% AUCROC (area under the receiver operating characteristic curve), compared with 70% with the self-play baseline. Looking at the acquisition of knowledge as training progresses, the agent first learned about its own base, then about the opponent’s base, and then about picking up the flag. Immediately useful flag knowledge was learned before knowledge related to tagging or their teammate’s situation. Agents were never explicitly trained to model this knowledge; thus, these results show the spontaneous emergence of these concepts purely through RL-based training.

A visualization of how the agent represents knowledge was obtained by performing dimensionality reduction of the agent’s activations through use of t-distributed stochastic neighbor embedding (t-SNE) (Fig. 3) (55). Internal agent state clustered in accordance with conjunctions of high-level game-state features: flag status, respawn state, and agent location (Fig. 3B). We also found individual neurons whose activations coded directly for some of these features—for example, a neuron that was active if and only if the agent’s teammate was holding the flag, which is reminiscent of concept cells (56). This knowledge was acquired in a distributed manner early in training (after 45,000 games) but then represented by a single, highly discriminative neuron later in training (at around 200,000 games). This observed disentangling of game state is most pronounced in the FTW agent (fig. S8).

Fig. 3. Knowledge representation and behavioral analysis. (A) The 2D t-SNE embedding of an FTW agent’s internal states during gameplay. Each point represents the internal state $\left(h', h''\right)$ at a particular point in the game and is colored according to the high-level game state at this time—the conjunction of (B) four basic CTF situations, each state of which is colored distinctly. Color clusters form, showing that nearby regions in the internal representation of the agent correspond to the same high-level game state. (C) A visualization of the expected internal state arranged in a similarity-preserving topological embedding and colored according to activation (fig. S5). (D) Distributions of situation conditional activations (each conditional distribution is colored gray and green) for particular single neurons that are distinctly selective for these CTF situations and show the predictive accuracy of this neuron. (E) The true return of the agent’s internal reward signal and (F) the agent’s prediction, its value function (orange denotes high value, and purple denotes low value). (G) Regions where the agent’s internal two-scale time representation diverges (red), the agent’s surprise, measured as the KL between the agent’s slow- and fast–time scale representations (28). (H) The four-step temporal sequence of the high-level strategy “opponent base camping.” (I) Three automatically discovered high-level behaviors of agents and corresponding regions in the t-SNE embedding. (Right) Average occurrence per game of each behavior for the FTW agent, the FTW agent without temporal hierarchy (TH), self-play with reward shaping agent, and human subjects (fig. S9).
The agent did not, resulting in more efficient flag capturing (fig. S2).

Analysis of temporally extended behaviors provided another view on the complexity of behavioral strategies learned by the agent (57) and is related to the problem a coach might face when analyzing behavior patterns in an opponent team (58). We developed an unsupervised method to automatically discover and quantitatively characterize temporally extended behavior patterns, inspired by models of mouse behavior (59), which groups short game-play sequences into behavioral clusters (fig. S9 and movie S3). The discovered behaviors included well-known tactics observed in human play, such as “waiting in the opponents base for a flag to reappear” (“opponent base camping”), which we only observed in FTW agents with a temporal hierarchy. Some behaviors, such as “following a flag-carrying teammate,” were discovered and discarded midway through training, whereas others such as “performing home base defense” are most prominent later in training (Fig. 4).

Conclusions

In this work, we have demonstrated that an artificial agent using only pixels and game points as input can learn to play highly competitively in a rich multiagent environment: a popular multiplayer first-person video game. This was achieved by combining PBT of agents, internal reward optimization, and temporally hierarchical RL with scalable computational architectures. The presented framework of training populations of agents, each with their own learned rewards, makes minimal assumptions about the game structure and therefore could be applicable for scalable and stable learning in a wide variety of multiagent systems. The temporally hierarchical agent represents a powerful architecture for problems that require memory and temporally extended inference. Limitations of the current framework, which should be addressed in future work, include the difficulty of maintaining diversity in agent populations, the greedy nature of the meta-optimization performed by PBT, and the variance from temporal credit assignment in the proposed RL updates. Our work combines techniques to train agents that can achieve human-level performance at previously insurmountable tasks. When trained in a sufficiently rich multiagent world, complex and surprising high-level intelligent artificial behavior emerged.
REFERENCES AND NOTES

Human-level performance in 3D multiplayer games with population-based reinforcement learning


Science 364 (6443), 859-865.
DOI: 10.1126/science.aau6249

Artificial teamwork
Artificially intelligent agents are getting better and better at two-player games, but most real-world endeavors require teamwork. Jaderberg et al. designed a computer program that excels at playing the video game Quake III Arena in Capture the Flag mode, where two multiplayer teams compete in capturing the flags of the opposing team. The agents were trained by playing thousands of games, gradually learning successful strategies not unlike those favored by their human counterparts. Computer agents competed successfully against humans even when their reaction times were slowed to match those of humans.

Science, this issue p. 859