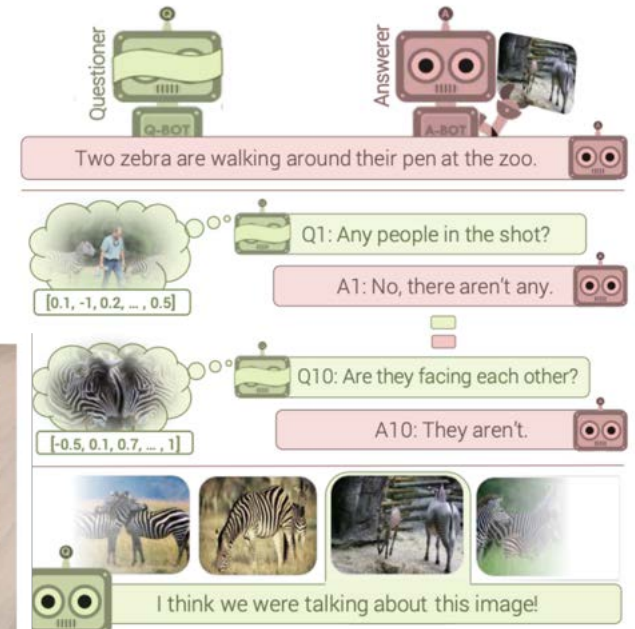
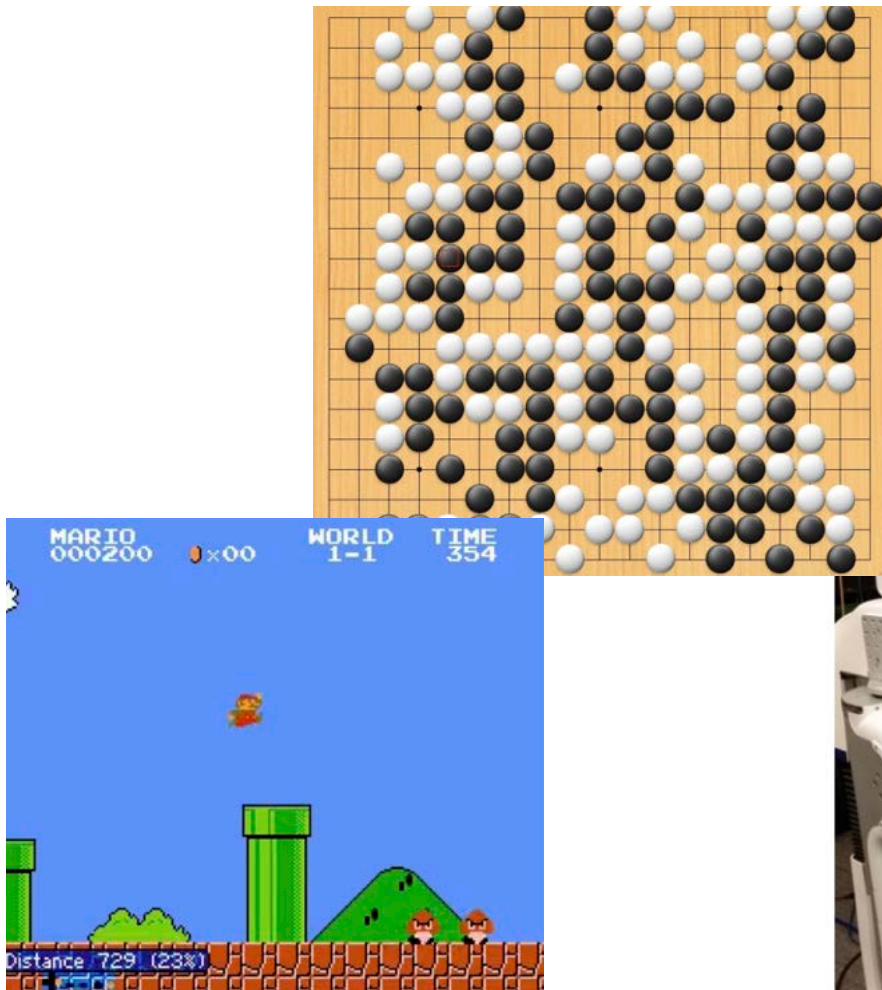


Deep RL challenges and applications



RL challenges: Overview

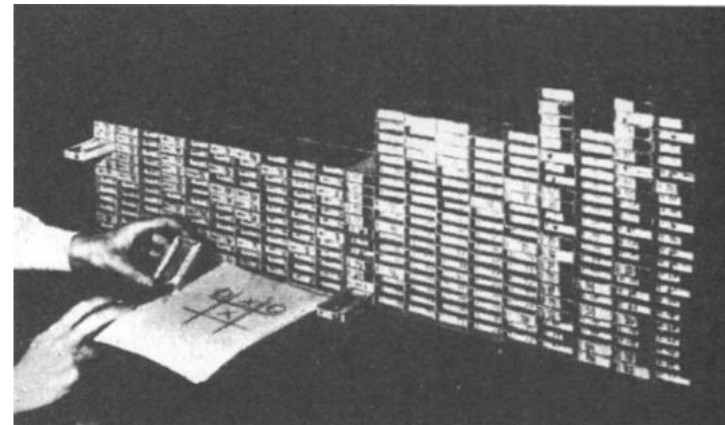
- Large state and action spaces, sparse rewards, planning many steps ahead
- Tasks that cannot be bootstrapped from random performance
- Not knowing the reward function up front
- Learning without a reward function *at all*
- Training faster, generalizing from one task to another
- Dealing with multiple agents: cooperation, competition, communication

RL challenges: Overview

- Large state and action spaces, sparse rewards, planning many steps ahead

Recall: Origins of game-playing AI

- 1952-1959: [Arthur Samuel](#) programmed a digital computer to learn to play checkers
- 1960: [Donald Michie](#) built a “machine” out of 304 matchboxes that could learn to play tic-tac-toe

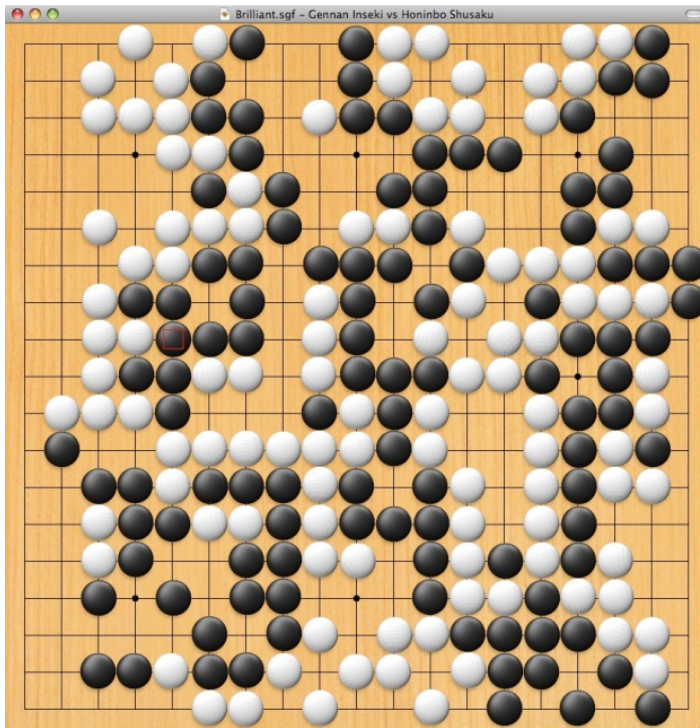


Recall: Origins of game-playing AI

- *“In 1959 Arthur Samuel published a paper titled ‘Some Studies in Machine Learning Using the Game of Checkers’, the first time the phrase ‘Machine Learning’ was used”*
- *“Donald Michie’s description of reinforcement learning comes from 1961, and is the first use of the term reinforcement learning when applied to a machine process ... There have been some developments in reinforcement learning since 1961, but only in details”*

[Rodney Brooks essay](#), 8/28/2017

Playing Go



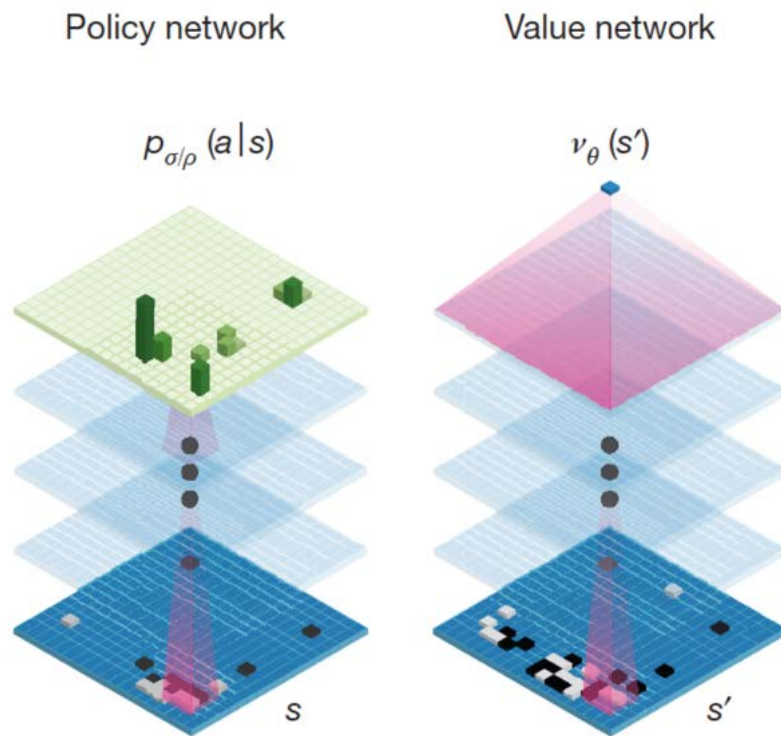
- Transition model is known and deterministic
- Key challenges: huge state and action space, very long sequences, sparse rewards

AlphaGo: Tree search guided by pattern recognition

- March 2016: [DeepMind's AlphaGo system beats world champion Lee Sedol 4-1](#)



AlphaGo: Tree search guided by pattern recognition



- **Policy network:** initialized by supervised learning on human games, improved by policy gradient
- **Value network:** trained to predict outcome of game based on self-play
- Networks are used to guide game play by **Monte Carlo tree search (MCTS)**

Network architecture

- Input: 19x19x48 feature map
- Architecture of policy networks: 13 layers, 5x5 filter in first layer, 3x3 filters in layers 2-12, 256 feature maps in layers 1-12, 1x1 filters followed by softmax in final layer, ensemble over 8 symmetries

Feature	# of planes	Description
Stone colour	3	Player stone / opponent stone / empty
Ones	1	A constant plane filled with 1
Turns since	8	How many turns since a move was played
Liberties	8	Number of liberties (empty adjacent points)
Capture size	8	How many opponent stones would be captured
Self-atari size	8	How many of own stones would be captured
Liberties after move	8	Number of liberties after this move is played
Ladder capture	1	Whether a move at this point is a successful ladder capture
Ladder escape	1	Whether a move at this point is a successful ladder escape
Sensibleness	1	Whether a move is legal and does not fill its own eyes
Zeros	1	A constant plane filled with 0
Player color	1	Whether current player is black

Feature planes used by the policy network (all but last feature) and value network (all features).

Supervised (SL) policy network

- Given position s , predict probability distribution over moves a
- Trained on 30M positions from human games (KGS Go Server), achieves 57% accuracy on move prediction
- Also train a smaller, faster *rollout policy* network (24% accuracy)

RL policy network

- Initialize with SL network
- Play current policy network against a randomly selected past snapshot, update parameters using policy gradients
- Use REINFORCE with reward $z = \pm 1$ (based on final outcome of the game) and baseline $v(s)$ provided by value network:

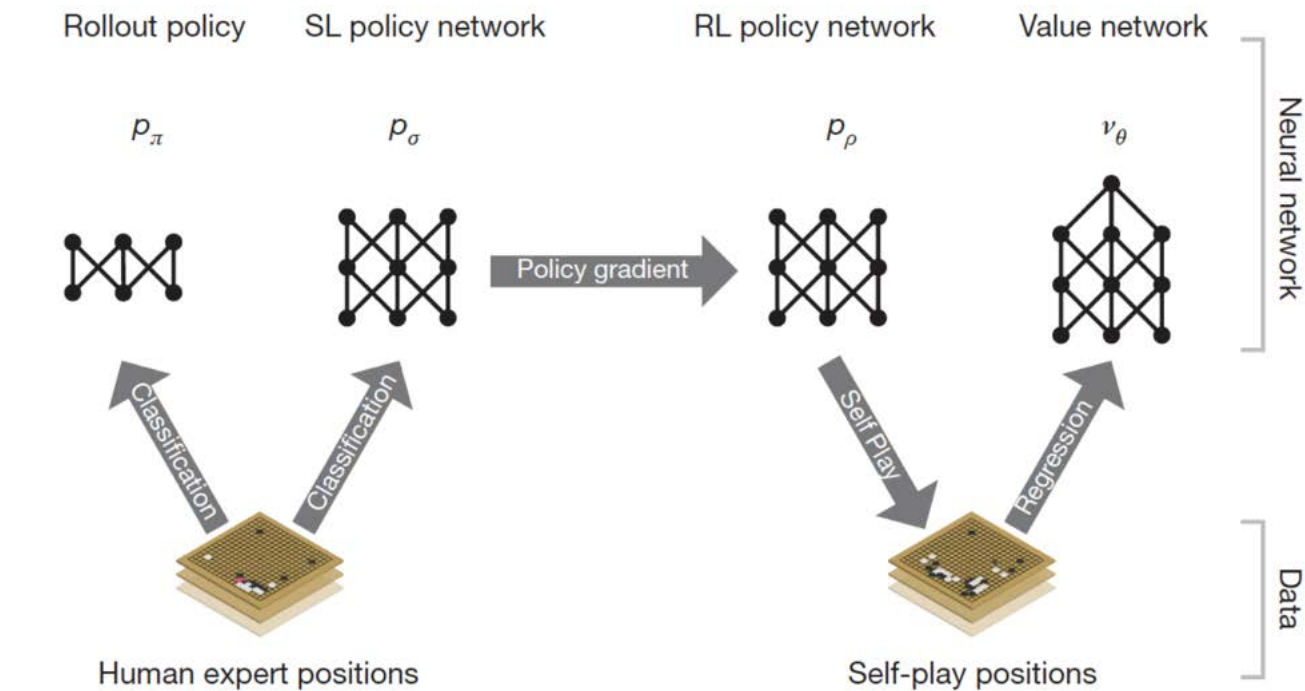
$$\begin{aligned}\nabla_{\theta} J(\theta) &\approx (z - v(s_t)) \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \\ \theta &\leftarrow \theta + \eta \nabla_{\theta} J(\theta)\end{aligned}$$

- RL network without any search wins against SL network 80% of the time, against open-source Pachi Go program 85% of the time

Value network

- Estimate $v(s)$, expected outcome of play starting with position s and following the learned policy for both players
- Train network by minimizing mean squared error (MSE) between actual and predicted outcome
- Training data: 30M positions, each sampled from a *different* self-play game (to avoid overfitting)

Summary of policy and value networks



Monte Carlo Tree Search

- At play time, select moves by MTCS guided by policy and value networks

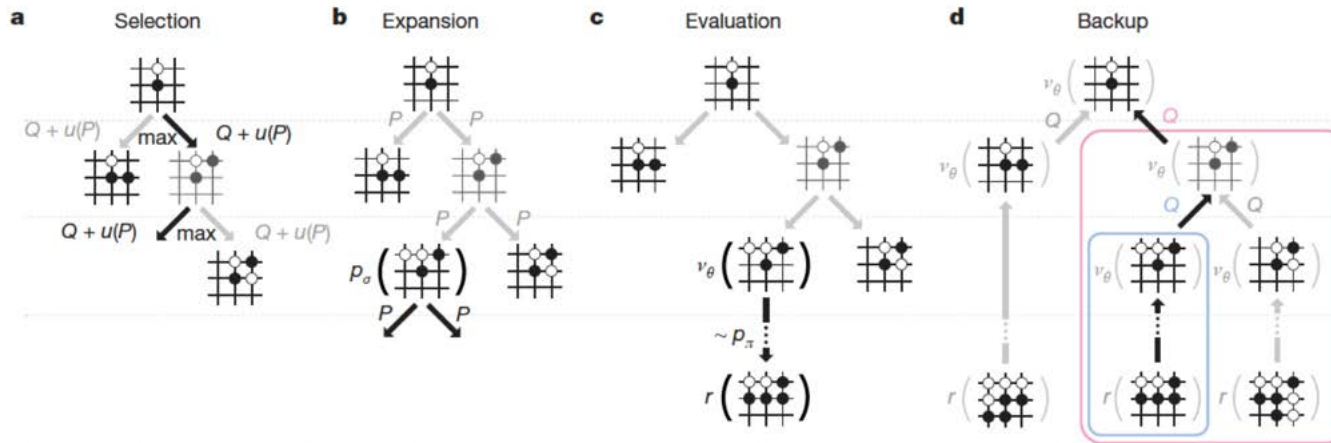


Figure 3 | Monte Carlo tree search in AlphaGo. a, Each simulation traverses the tree by selecting the edge with maximum action value Q , plus a bonus $u(P)$ that depends on a stored prior probability P for that edge. b, The leaf node may be expanded; the new node is processed once by the policy network p_π and the output probabilities are stored as prior probabilities P for each action. c, At the end of a simulation, the leaf node

is evaluated in two ways: using the value network v_θ ; and by running a rollout to the end of the game with the fast rollout policy p_π , then computing the winner with function r . d, Action values Q are updated to track the mean value of all evaluations $r(\cdot)$ and $v_\theta(\cdot)$ in the subtree below that action.

Evaluation

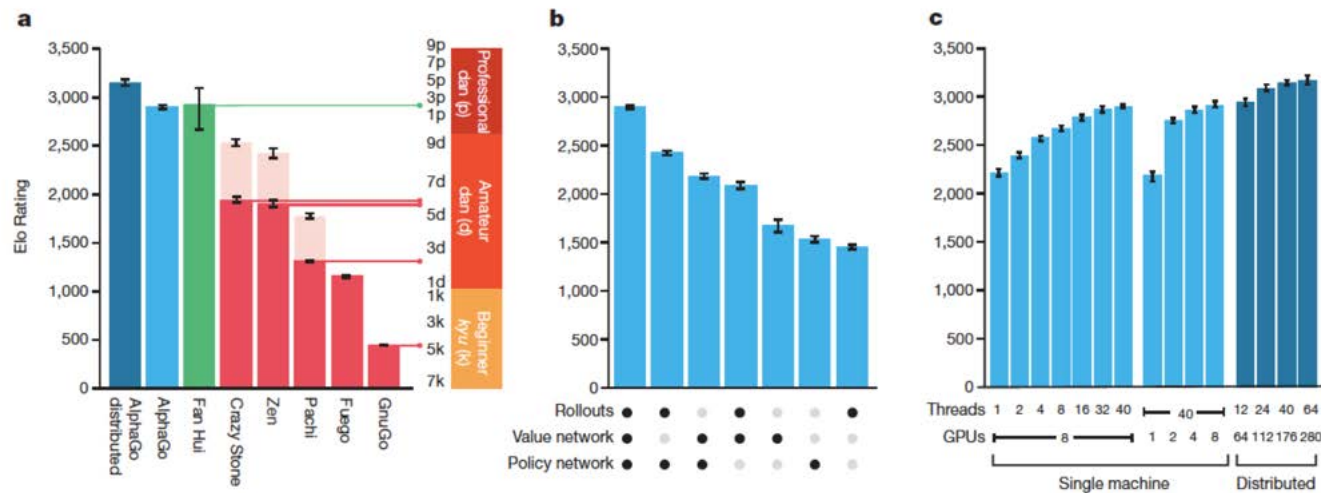


Figure 4 | Tournament evaluation of AlphaGo. **a**, Results of a tournament between different Go programs (see Extended Data Tables 6–11). Each program used approximately 5 s computation time per move. To provide a greater challenge to AlphaGo, some programs (pale upper bars) were given four handicap stones (that is, free moves at the start of every game) against all opponents. Programs were evaluated on an Elo scale³⁷: a 230 point gap corresponds to a 79% probability of winning, which roughly corresponds to one amateur *dan* rank advantage on KGS³⁸; an approximate correspondence to human ranks is also shown,

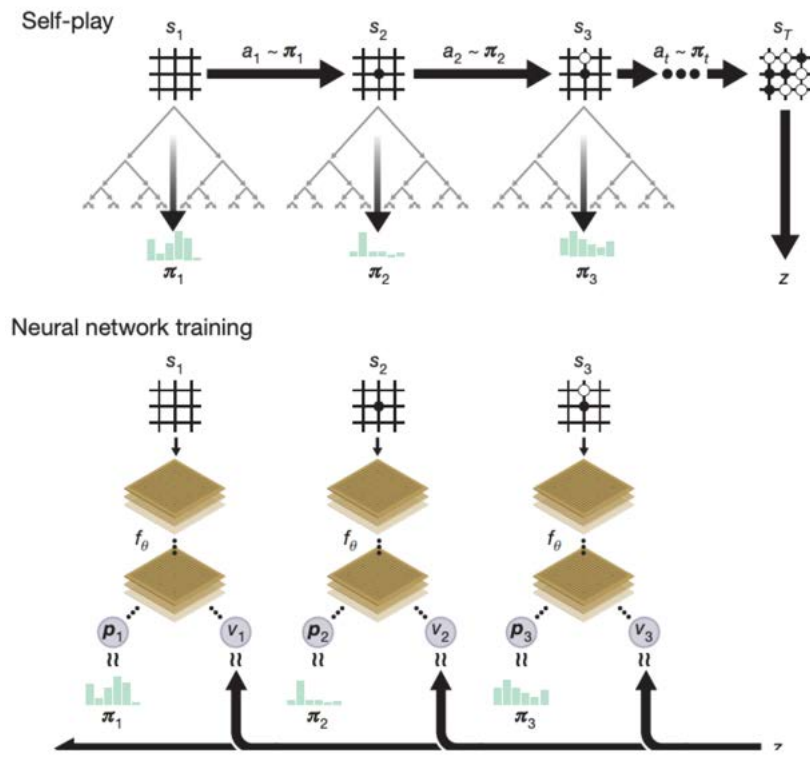
horizontal lines show KGS ranks achieved online by that program. Games against the human European champion Fan Hui were also included; these games used longer time controls. 95% confidence intervals are shown. **b**, Performance of AlphaGo, on a single machine, for different combinations of components. The version solely using the policy network does not perform any search. **c**, Scalability study of MCTS in AlphaGo with search threads and GPUs, using asynchronous search (light blue) or distributed search (dark blue), for 2 s per move.

AlphaGo Zero

- A fancier architecture (ResNets with BatchNorm)
- No hand-crafted features used as input
- Train a single network to simultaneously predict value and policy
- Training is done entirely by RL with self-play, starting with random moves
- MCTS is used inside the RL loop, not outside

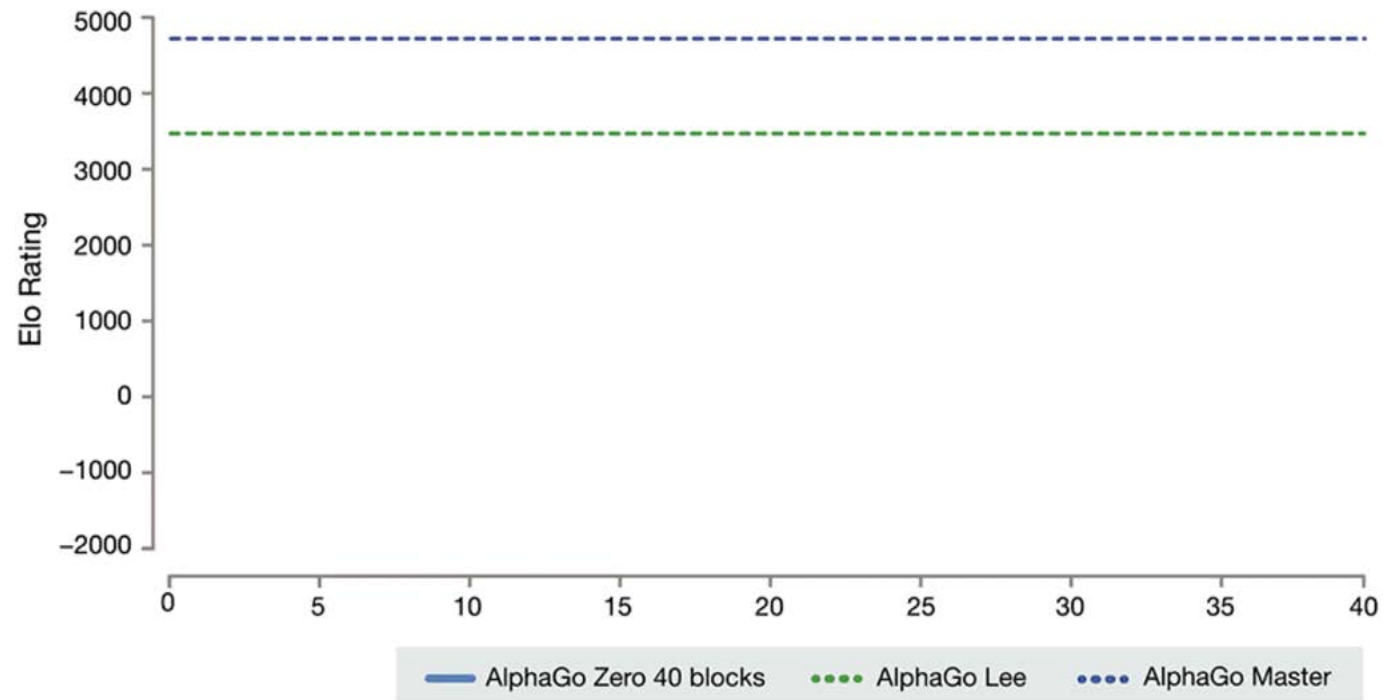
D. Silver et al., [Mastering the Game of Go without Human Knowledge](https://deepmind.com/blog/alphago-zero-learning-scratch/), Nature 550, October 2017
<https://deepmind.com/blog/alphago-zero-learning-scratch/>

RL loop

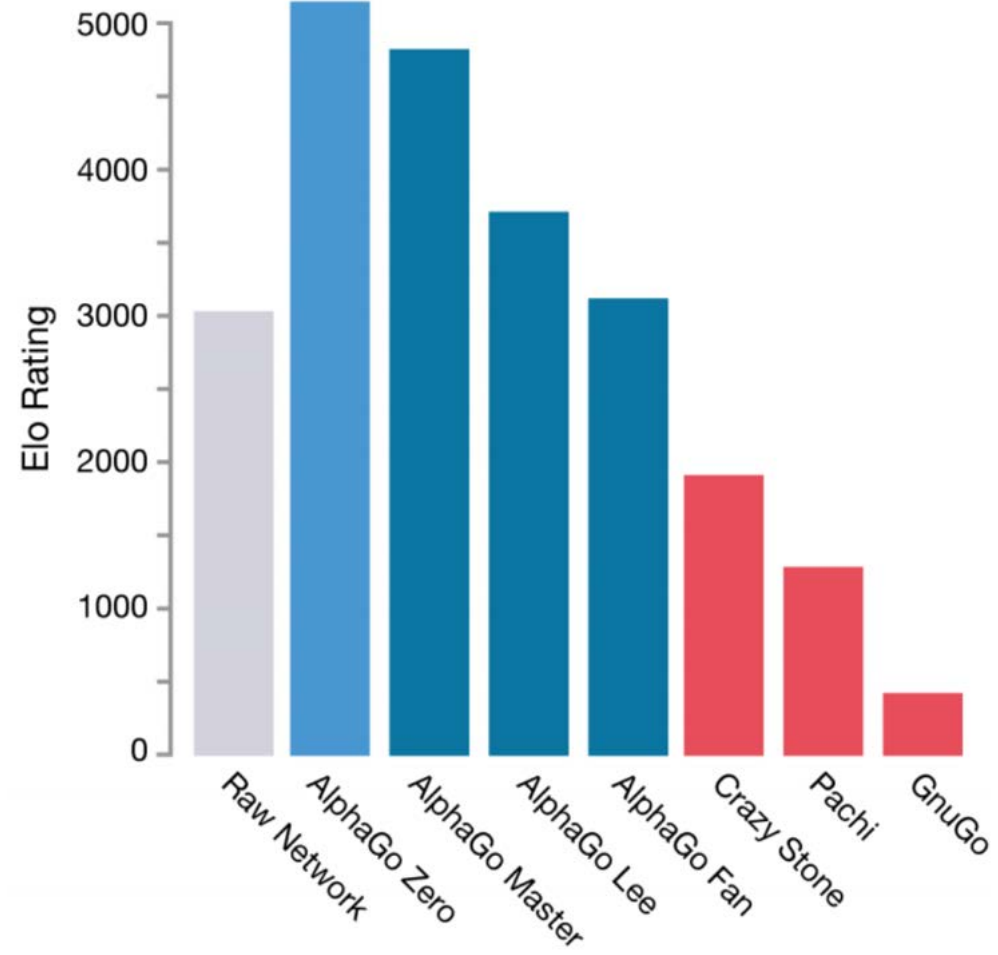


- Given a position s , use current network to predict move probabilities $p(a|s)$ and value $v(s)$
- Run MCTS guided by the network to obtain refined move probabilities π and final value z
- Update network parameters to make $p(a|s)$ and $v(s)$ closer to π and z (MSE loss on v and cross-entropy on p)

Results



Results



AlphaZero

The Telegraph

News | Science

🏠 » News » Science

Entire human chess knowledge learned and surpassed by DeepMind's AlphaZero in four hours



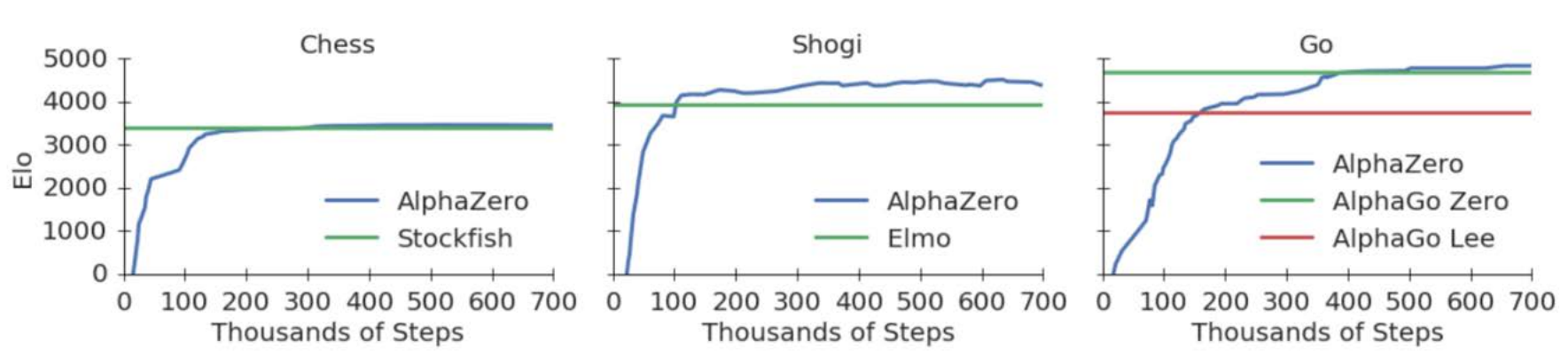
Save 17



Demis Hassabis, the head of DeepMind which has already shown it can defeat the world's best Go champions CREDIT: JUNG YEON-JE

<https://www.telegraph.co.uk/science/2017/12/06/entire-human-chess-knowledge-learned-surpassed-deepminds-alphazero/>

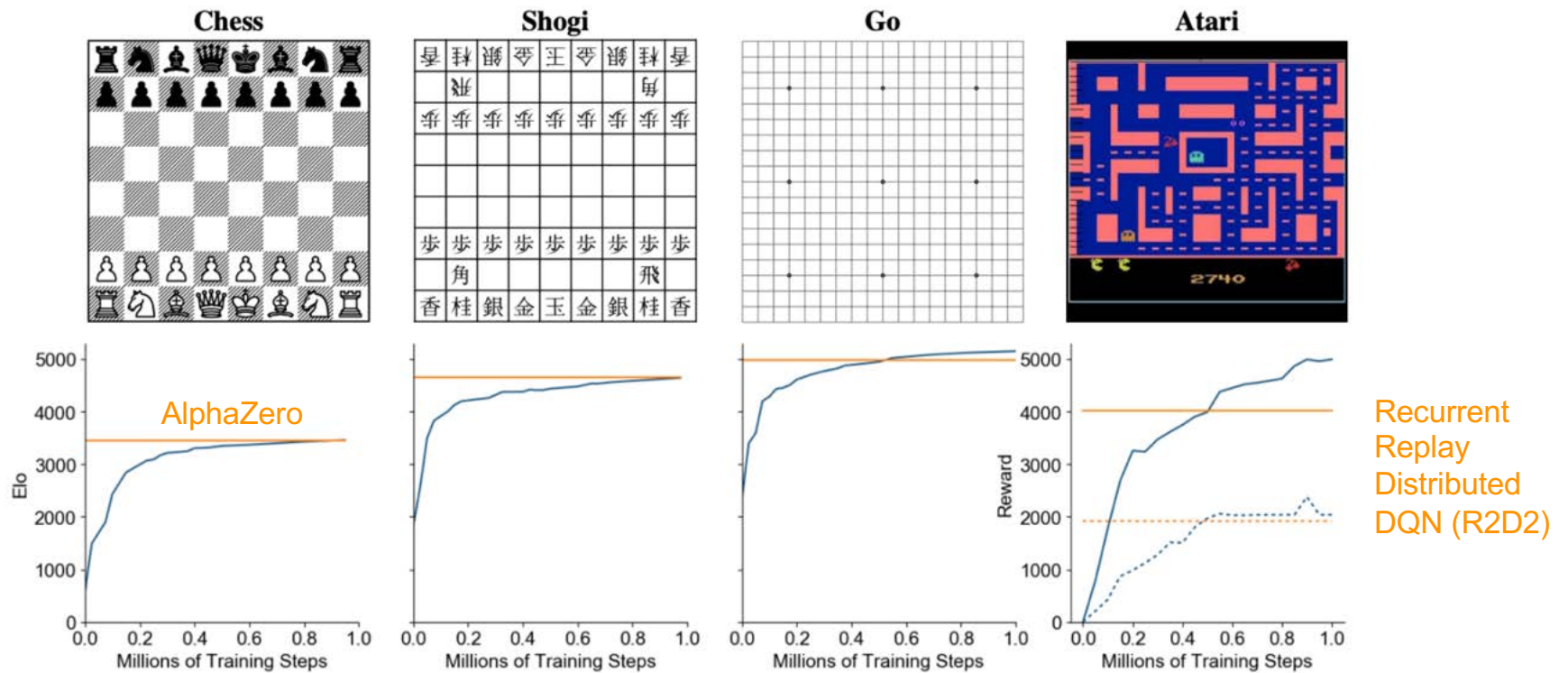
AlphaZero



D. Silver et al., [Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm](#), arXiv 2017

MuZero

- Model-based RL, does not require a perfect simulator



J. Schrittwieser et al., [Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model](#), 2020

RL challenges: Overview

- Large state and action spaces, sparse rewards, planning many steps ahead
- Tasks that cannot be bootstrapped from random performance
- Not knowing the reward function up front
- Learning without a reward function *at all*

Imitation learning

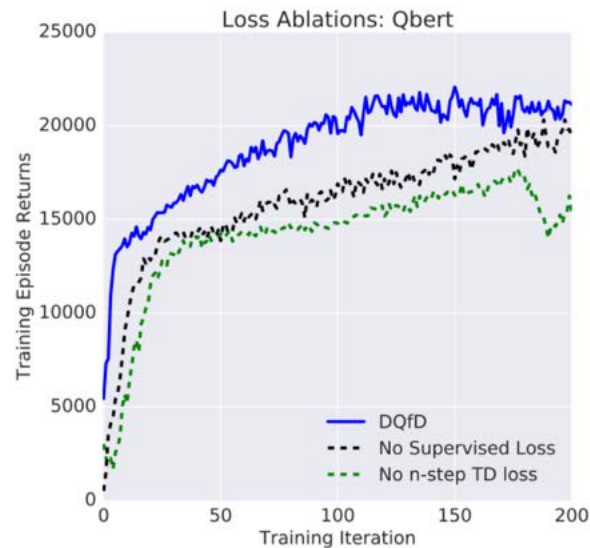
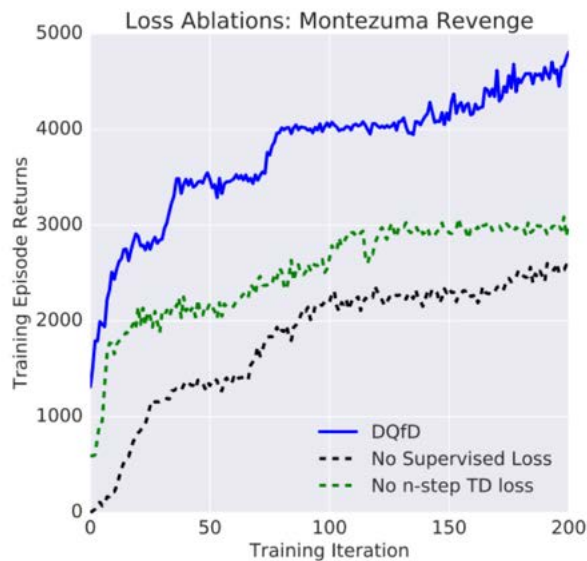


- In some applications, you cannot bootstrap yourself from random policies
 - High-dimensional state and action spaces where most random trajectories fail miserably
 - Expensive to evaluate policies in the physical world, especially in cases of failure
 - In such cases, a reward function may not be easy to formulate either
- **Solution:** learn to imitate trajectories or demonstrations provided by an “expert” or “teacher”

Deep Q-learning from demonstrations (DQfD)

- In addition to environment access, assume you are given some demonstration data up front
 - Pre-train on the demonstration data before starting any interaction with the environment (supervised learning)
 - Keep demonstration data in replay buffer permanently and give it a priority bonus (sample more frequently)
- Combination of losses:
 - Supervised loss: a large-margin supervised loss to push the value of the demonstrator's actions above the other action values
 - Regularization loss: L2 regularization on network weights to prevent overfitting on the demonstration data
 - N-step TD loss: a form of DQN loss

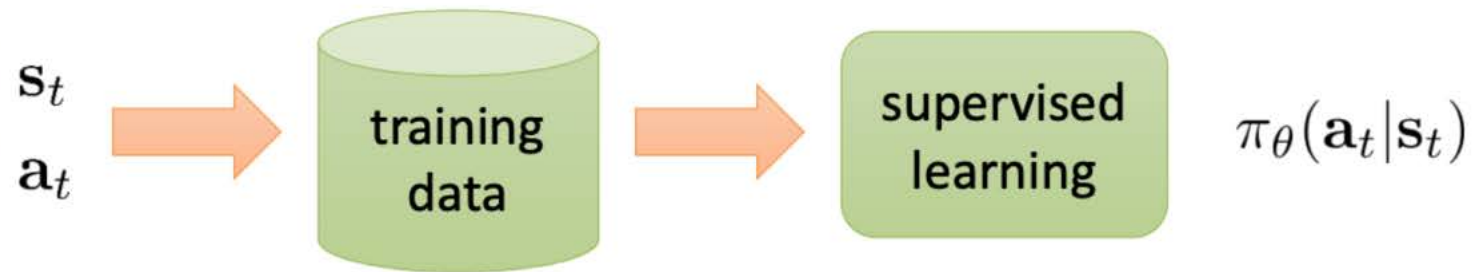
DQfD: Results



Game	DQfD	Prev. Best	Algorithm
Alien	4745.9	4461.4	Dueling DQN (Wang et al. 2016)
Asteroids	3796.4	2869.3	PopArt (van Hasselt et al. 2016)
Atlantis	920213.9	395762.0	Prior. Dueling DQN (Wang et al. 2016)
Battle Zone	41971.7	37150.0	Dueling DQN (Wang et al. 2016)
Gravitar	1693.2	859.1	DQN+PixelCNN (Ostrovski et al. 2017)
Hero	105929.4	23037.7	Prioritized DQN (Schaul et al. 2016)
Montezuma Revenge	4739.6	3705.5	DQN+CTS (Ostrovski et al. 2017)
Pitfall	50.8	0.0	Prior. Dueling DQN (Wang et al. 2016)
Private Eye	40908.2	15806.5	DQN+PixelCNN (Ostrovski et al. 2017)
Q-Bert	21792.7	19220.3	Dueling DQN (Wang et al. 2016)
Up N Down	82555.0	44939.6	Dueling DQN (Wang et al. 2016)

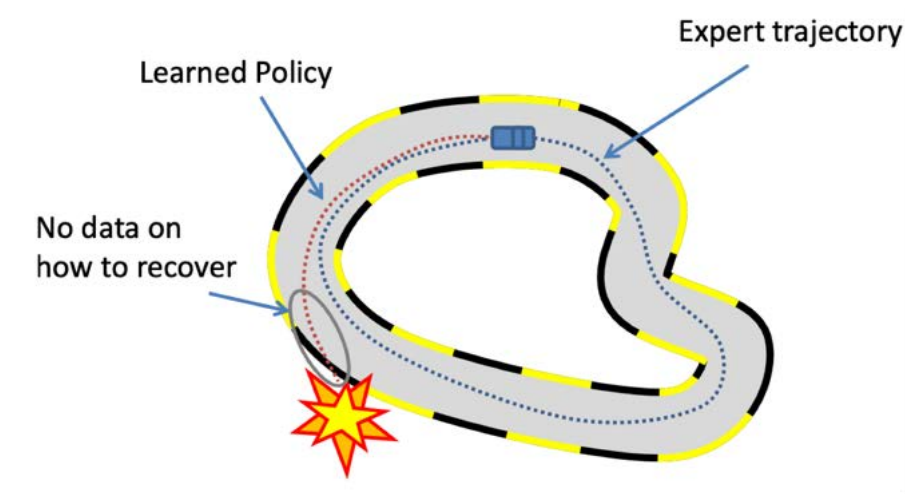
Imitation learning without rewards

- Setting: you have no reward function, but you have access to an expert that can be queried to obtain a demonstration action for any state
- Can be approached as supervised learning



Imitation learning without rewards

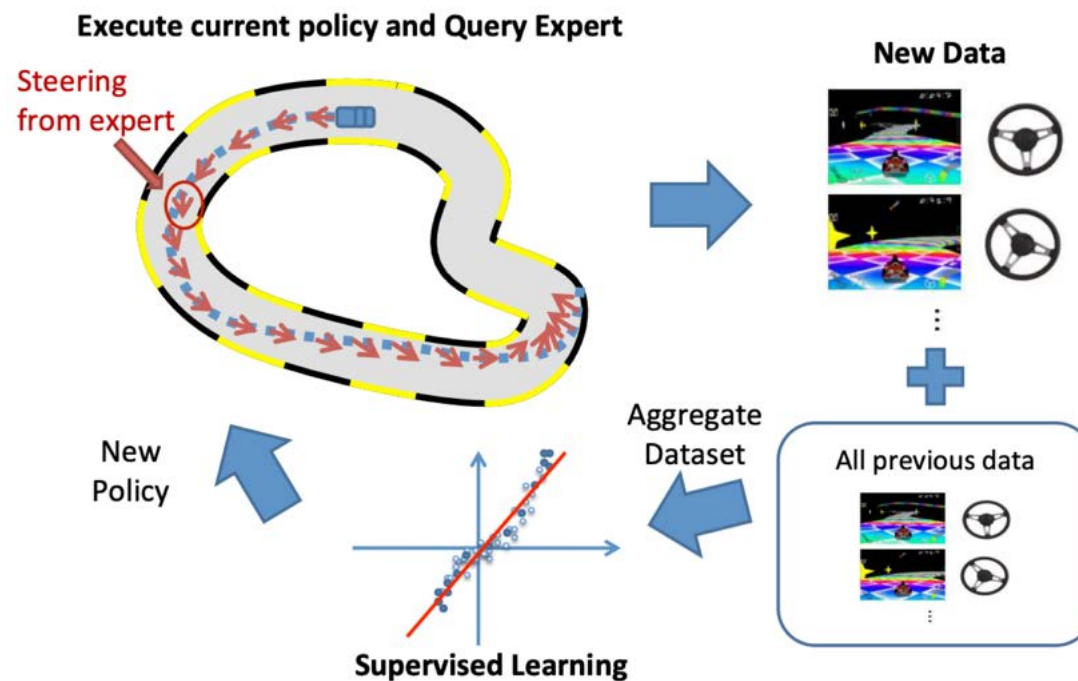
- Setting: you have no reward function, but you have access to an expert that can be queried to obtain a demonstration action for any state
- Can be approached as supervised learning
- Problem: data is not i.i.d., mistakes can send you to a part of the state space you don't know how to handle



[Image source](#)

Dataset Aggregation (DAgger)

- Get (state, action) pairs by querying the expert, train policy to imitate expert, collect more trajectories, repeat



[Image source](#)

S. Ross, G. Gordon, D. Bagnell, [A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning](#), AISTATS 2011

Dataset Aggregation (DAgger)

```
Initialize  $\mathcal{D} \leftarrow \emptyset$ .  
Initialize  $\hat{\pi}_1$  to any policy in  $\Pi$ .  
for  $i = 1$  to  $N$  do  
  Let  $\pi_i = \beta_i \pi^* + (1 - \beta_i) \hat{\pi}_i$ .  
  Sample  $T$ -step trajectories using  $\pi_i$ .  
  Get dataset  $\mathcal{D}_i = \{(s, \pi^*(s))\}$  of visited states by  $\pi_i$   
  and actions given by expert.  
  Aggregate datasets:  $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_i$ .  
  Train classifier  $\hat{\pi}_{i+1}$  on  $\mathcal{D}$ .  
end for  
Return best  $\hat{\pi}_i$  on validation.
```

Learning visuomotor policies



- Key idea: collect expert trajectories through specially instrumented training regimen, then learn to imitate them from raw sensory observations

S. Levine et al. [End-to-end training of deep visuomotor policies](#). JMLR 2016

Learning visuomotor policies



- *Underlying state* x : true object position, robot configuration
- *Observations* o : image pixels
- Two-part approach:
 - Learn *guiding policy* $p(u|x)$ using trajectory-centric RL and control techniques
 - Learn *visuomotor policy* $\pi(u|o)$ by imitating $p(u|x)$

S. Levine et al. [End-to-end training of deep visuomotor policies](#). JMLR 2016

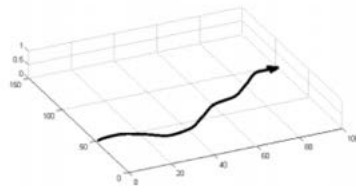
Learning visuomotor policies

Instrumented training:

training time



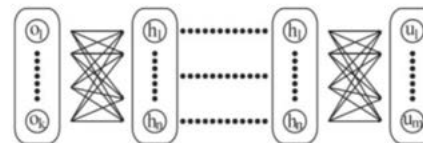
$\mathbf{x}_t \rightarrow \mathbf{u}_t$



test time

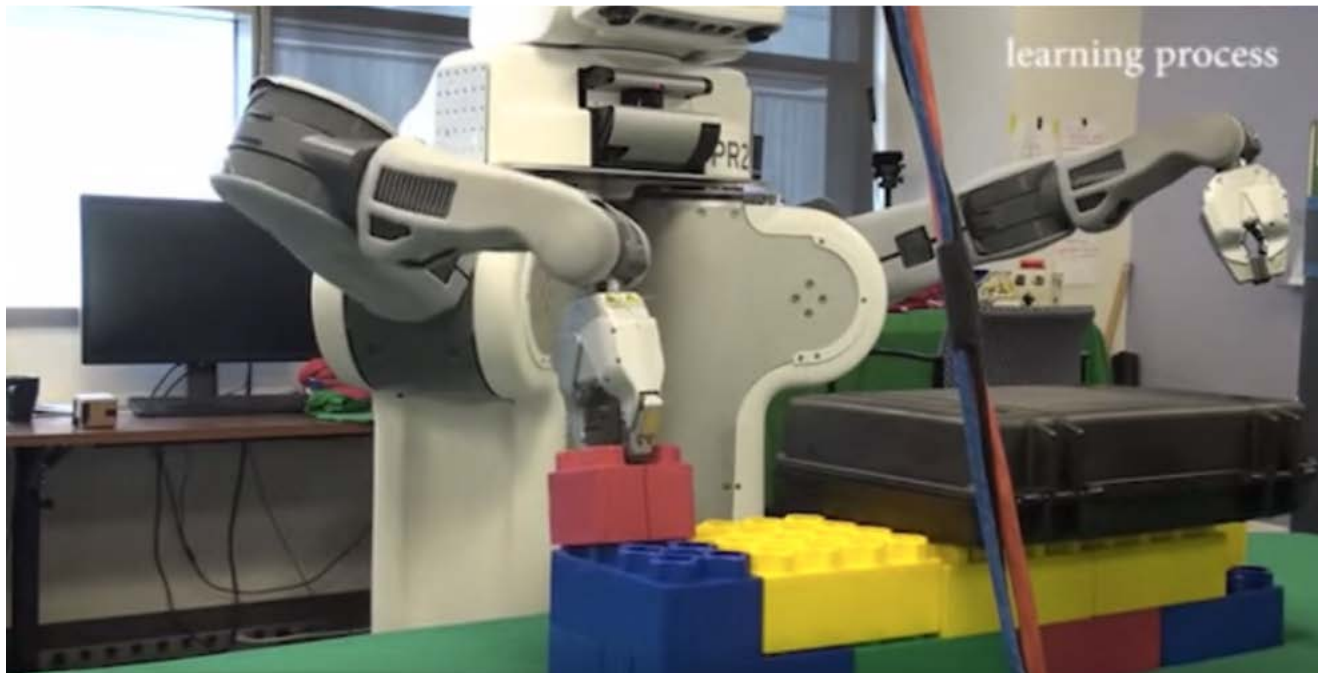


$\mathbf{o}_t \rightarrow \mathbf{u}_t$



[Source](#)

Learning visuomotor policies



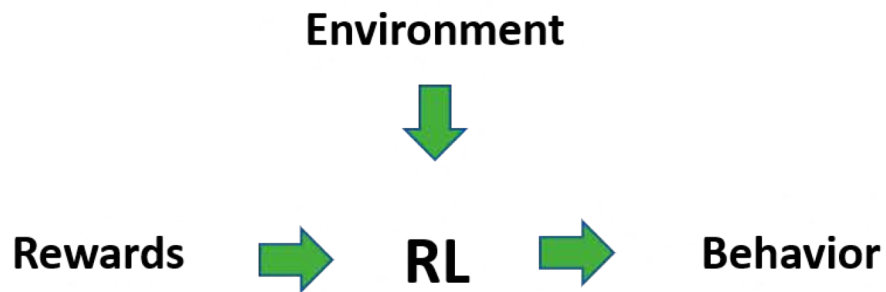
[Overview video](#), [training video](#)

S. Levine et al. [End-to-end training of deep visuomotor policies](#). JMLR 2016

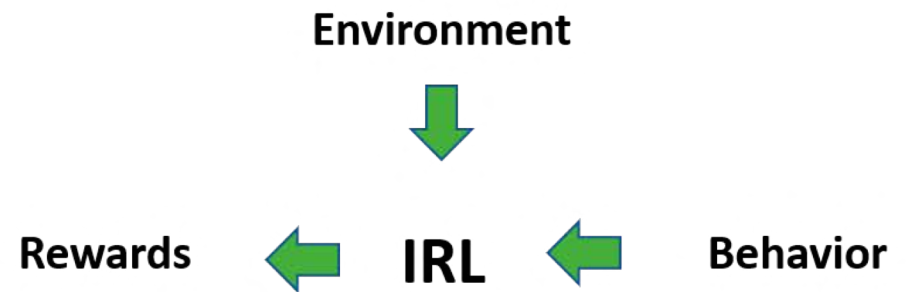
Inverse reinforcement learning (IRL)

- Use expert demonstrations to infer reward function, then learn policy using regular RL techniques

Reinforcement Learning



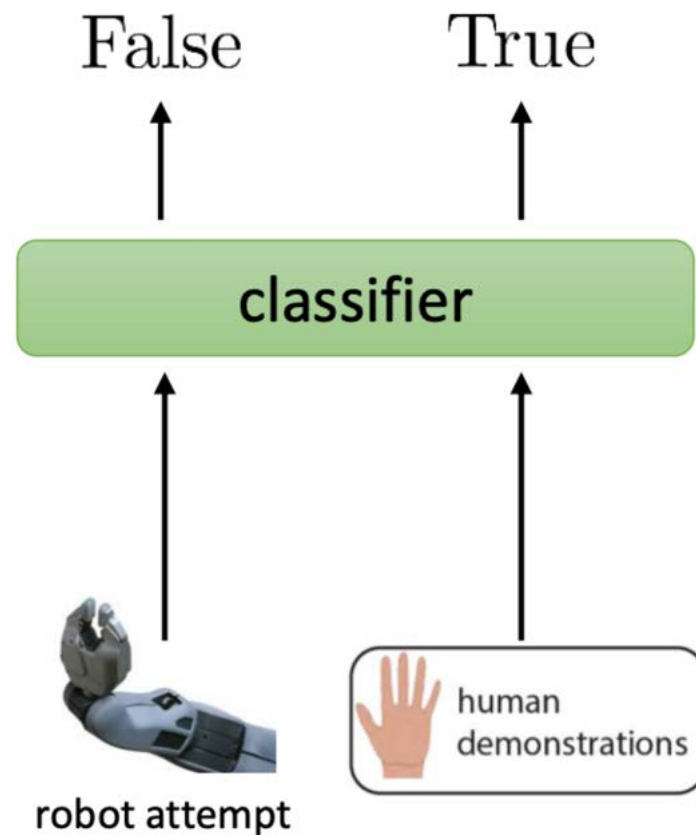
Inverse Reinforcement Learning



[Image source](#)

Generative adversarial imitation learning

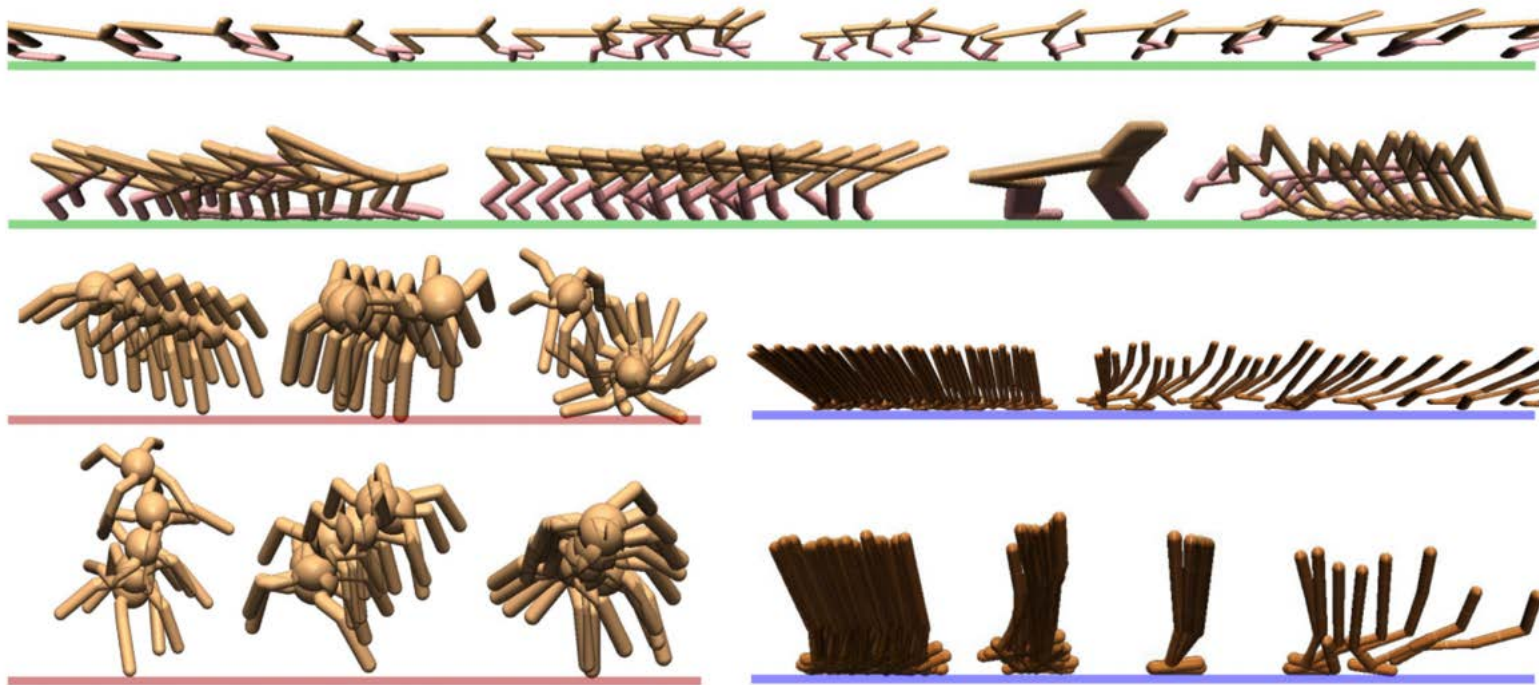
- Generator samples trajectories using current learned policy, discriminator tries to distinguish those trajectories from expert ones, discriminator signal is used to update policy



[Image source](#)

Pseudo-rewards from diversity

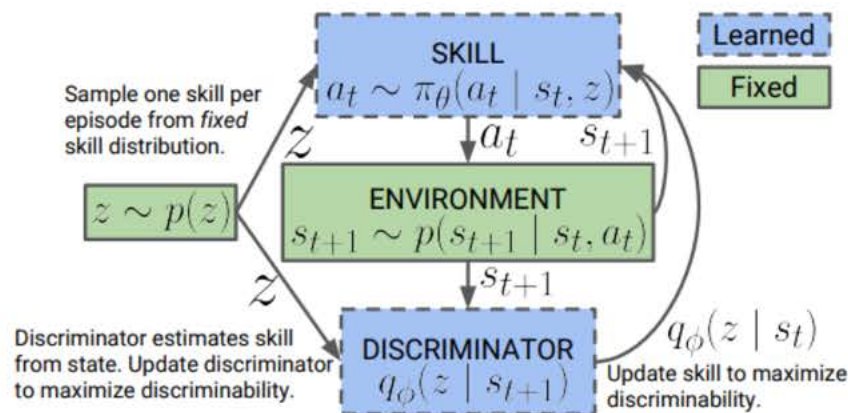
- Learn skills that are distinguishable from each other, as diverse as possible



B. Eysenbach, A. Gupta, J. Ibarz, S. Levine, [Diversity is all you need: Learning skills without a reward function](#), ICLR 2018

Pseudo-rewards from diversity

- Learn skills that are distinguishable from each other, as diverse as possible



Algorithm 1: DIAYN

while not converged do

 Sample skill $z \sim p(z)$ and initial state $s_0 \sim p_0(s)$

for $t \leftarrow 1$ **to** $steps_per_episode$ **do**

 Sample action $a_t \sim \pi_\theta(a_t | s_t, z)$ from skill.

 Step environment: $s_{t+1} \sim p(s_{t+1} | s_t, a_t)$.

 Compute $q_\phi(z | s_{t+1})$ with discriminator.

 Set skill reward $r_t = \log q_\phi(z | s_{t+1}) - \log p(z)$

 Update policy (θ) to maximize r_t with SAC.

 Update discriminator (ϕ) with SGD.

Figure 1: **DIAYN Algorithm:** We update the discriminator to better predict the skill, and update the skill to visit diverse states that make it more discriminable.

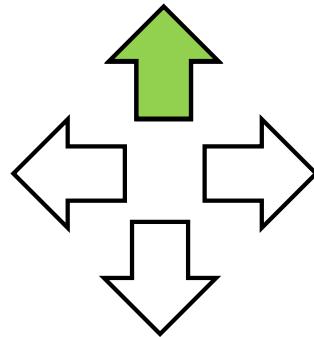
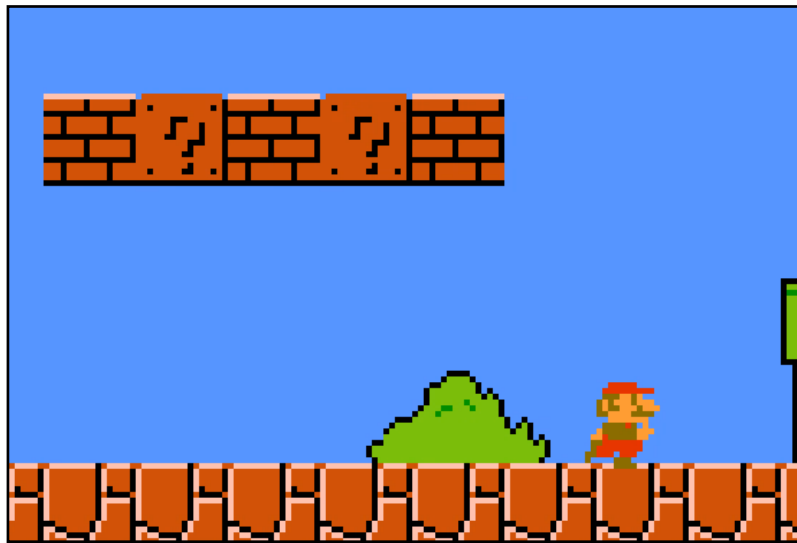
Pseudo-rewards from curiosity

- Encourage agent to prefer actions whose outcome it cannot predict

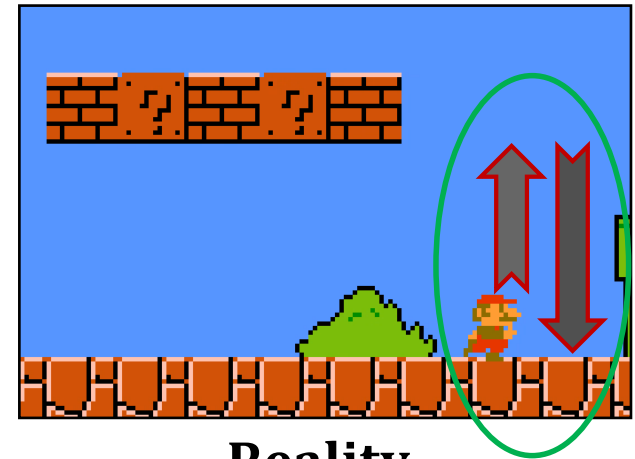


[Video](#)

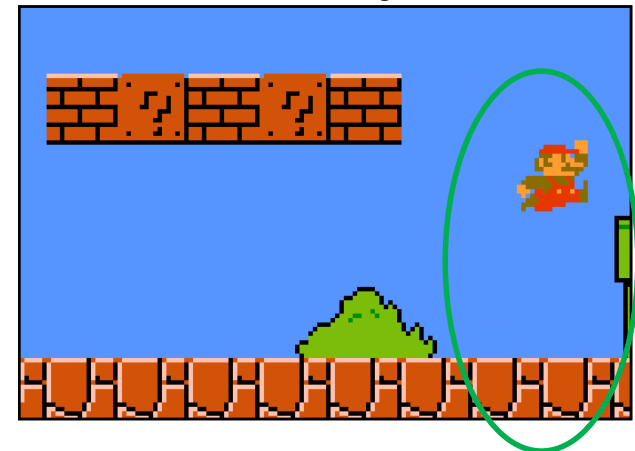
Prediction error as curiosity



Prediction

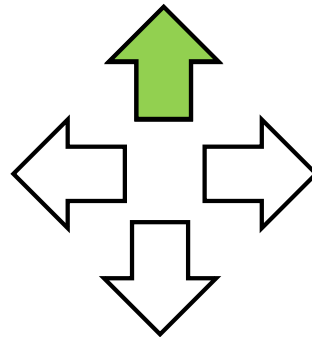
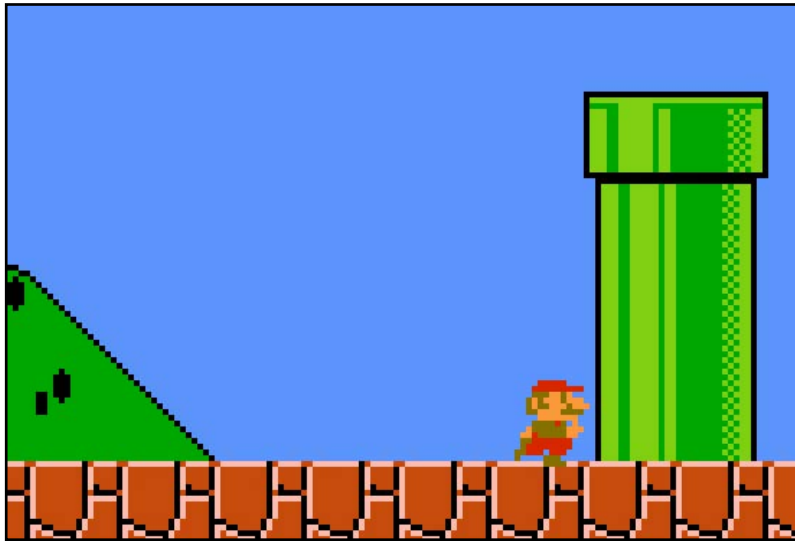


Reality

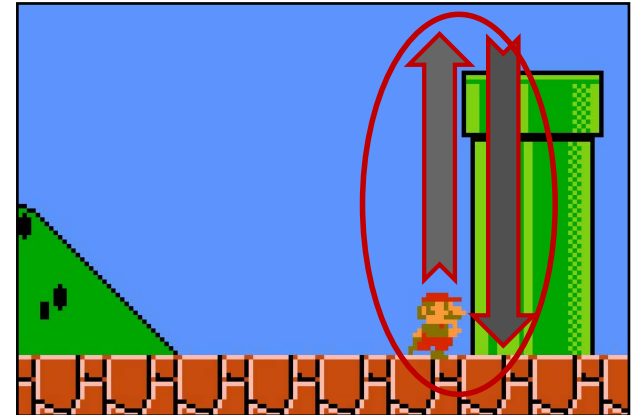


Source: D. Pathak et al. (via A. Efros)

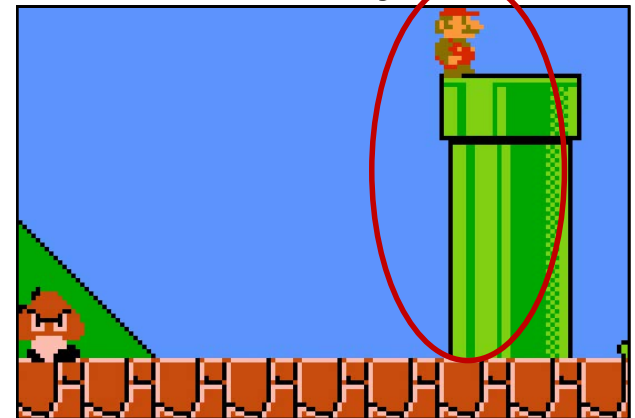
Prediction error as curiosity



Prediction



Reality



Source: D. Pathak et al. (via A. Efros)

Intrinsic curiosity module (ICM)

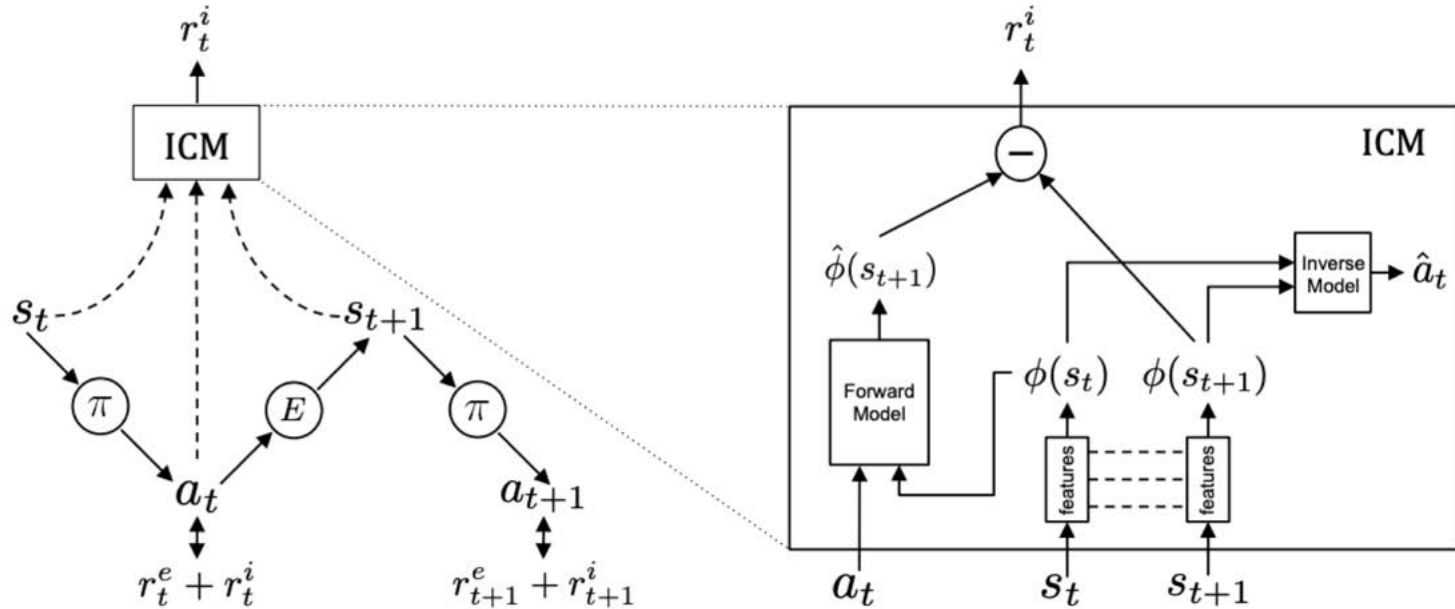


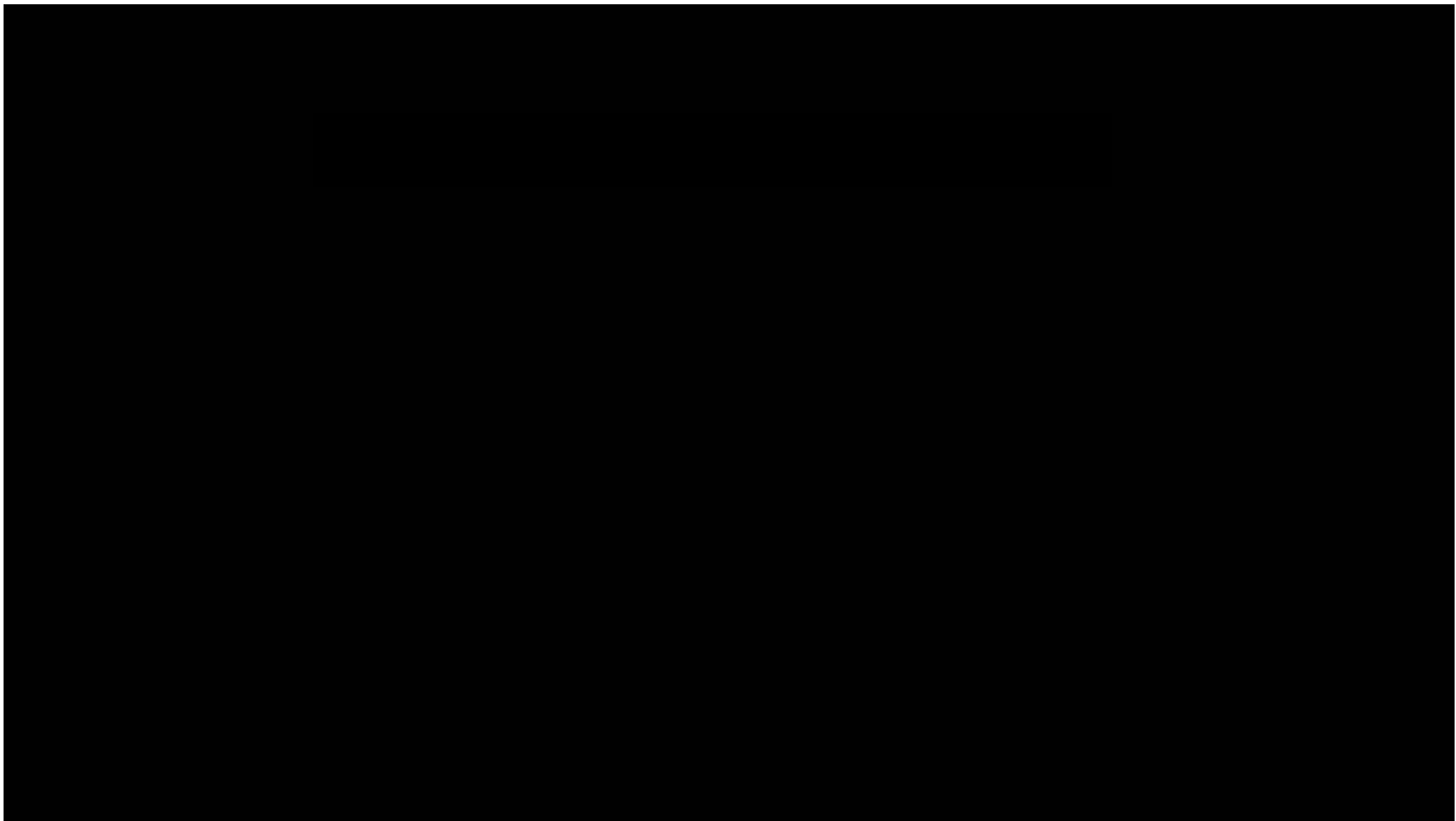
Figure 2. The agent in state s_t interacts with the environment by executing an action a_t sampled from its current policy π and ends up in the state s_{t+1} . The policy π is trained to optimize the sum of the extrinsic reward (r_t^e) provided by the environment E and the curiosity based intrinsic reward signal (r_t^i) generated by our proposed Intrinsic Curiosity Module (ICM). ICM encodes the states s_t, s_{t+1} into the features $\phi(s_t), \phi(s_{t+1})$ that are trained to predict a_t (i.e. inverse dynamics model). The forward model takes as inputs $\phi(s_t)$ and a_t and predicts the feature representation $\hat{\phi}(s_{t+1})$ of s_{t+1} . The prediction error in the feature space is used as the curiosity based intrinsic reward signal. As there is no incentive for $\phi(s_t)$ to encode any environmental features that can not influence or are not influenced by the agent's actions, the learned exploration strategy of our agent is robust to uncontrollable aspects of the environment.

No external reward, only curiosity



After this start of training

Source: D. Pathak et al. (via A. Efros)



Do these skills generalize?

Trained on Level-1

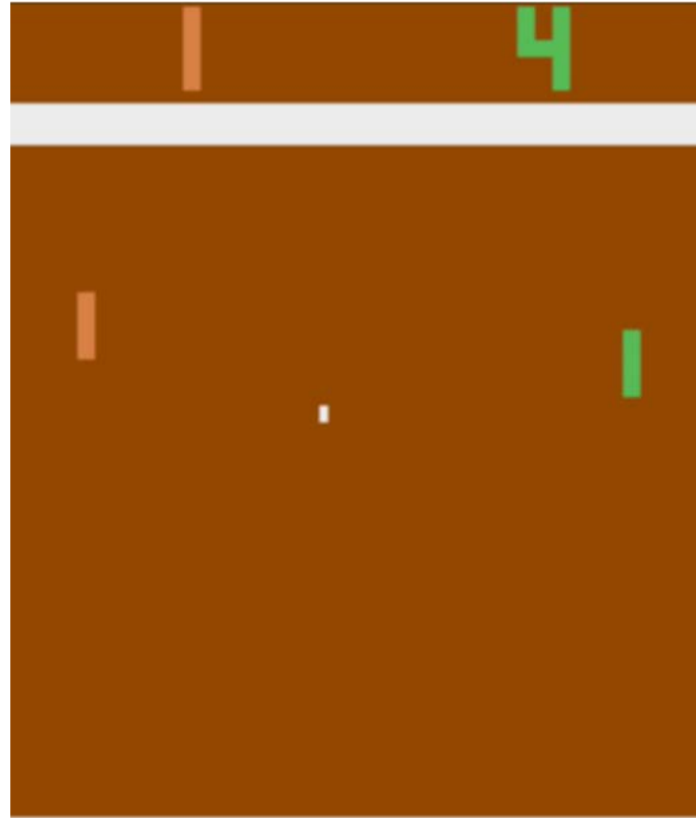


Testing on Level-2



Source: D. Pathak et al. (via A. Efros)

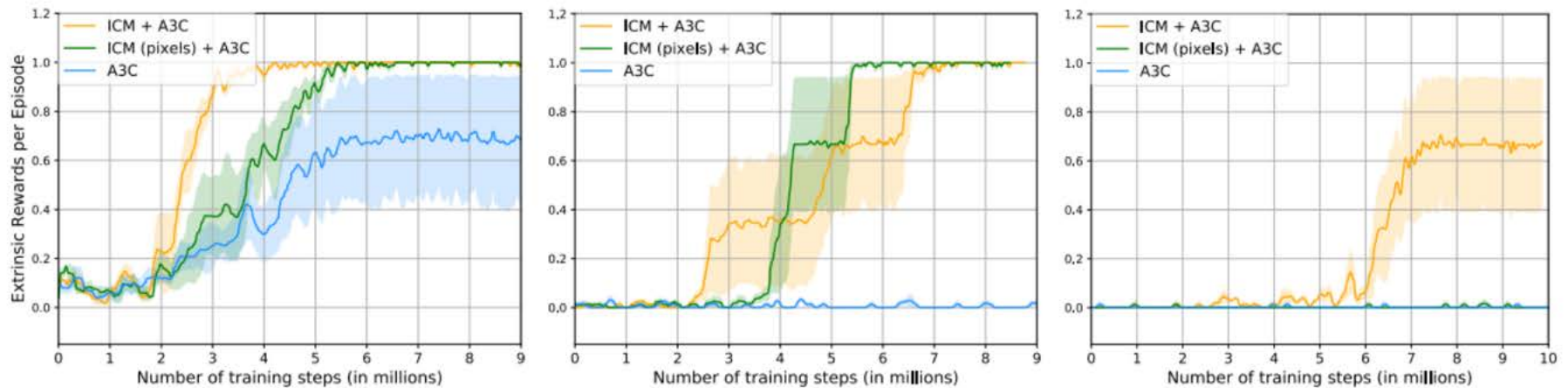
Curiosity on both sides... makes a rally



Environment: Multi-player Pong

Source: D. Pathak et al. (via A. Efros)

Some quantitative results



(a) "dense reward" setting

(b) "sparse reward" setting

(c) "very sparse reward" setting

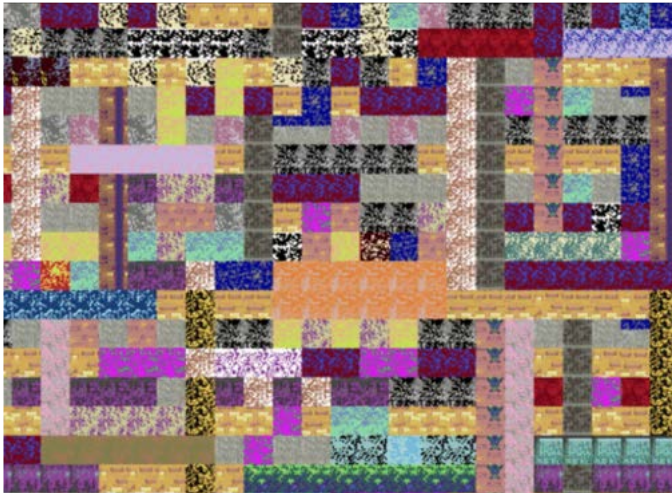
Figure 5. Comparing the performance of the A3C agent with no curiosity (blue) against the curiosity in pixel space agent (green) and the proposed curious ICM-A3C agent (orange) as the hardness of the exploration task is gradually increased from left to right. Exploration becomes harder with larger distance between the initial and goal locations: "dense", "sparse" and "very sparse". The results depict that succeeding on harder exploration task becomes progressively harder for the baseline A3C, whereas the curious A3C is able to achieve good score in all the scenarios. Pixel based curiosity works in dense and sparse but fails in very sparse reward setting. The protocol followed in the plots involves running three independent runs of each algorithm. Darker line represents mean and shaded area represents mean \pm standard error of mean. We did not perform any tuning of random seeds.

RL challenges: Overview

- Large state and action spaces, sparse rewards, planning many steps ahead
- Tasks that cannot be bootstrapped from random performance
- Not knowing the reward function up front
- Learning without a reward function *at all*
- Training faster, generalizing from one task to another

The importance of features

- Why do RL agents take so long to learn compared to humans?
- Hypothesis: humans have useful visual priors or features they can exploit



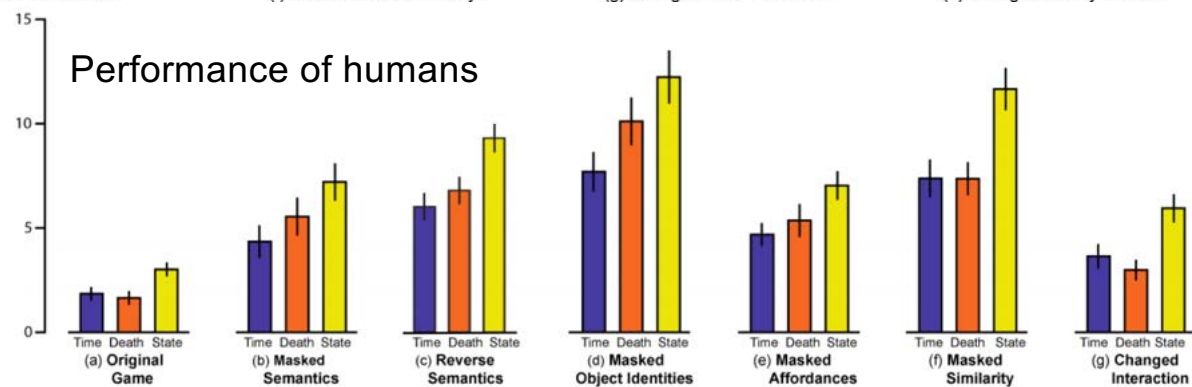
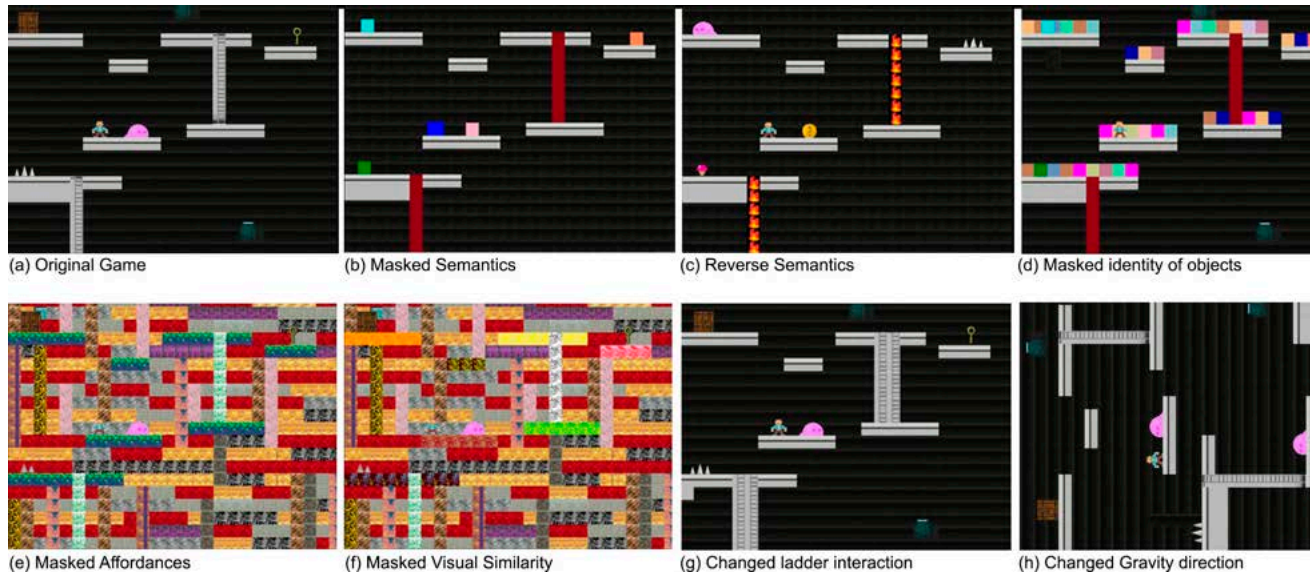
Human gameplay on game version without object priors



Human gameplay on original version

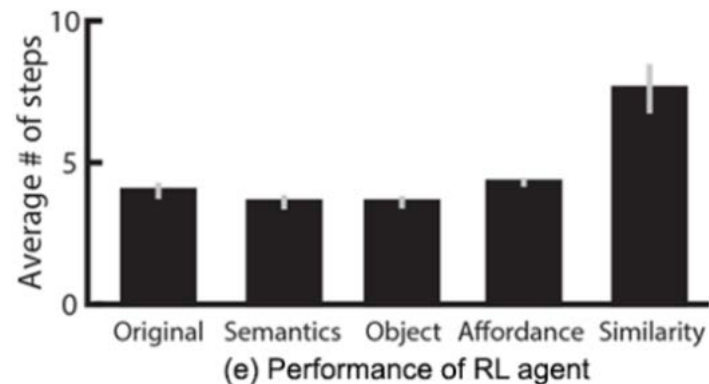
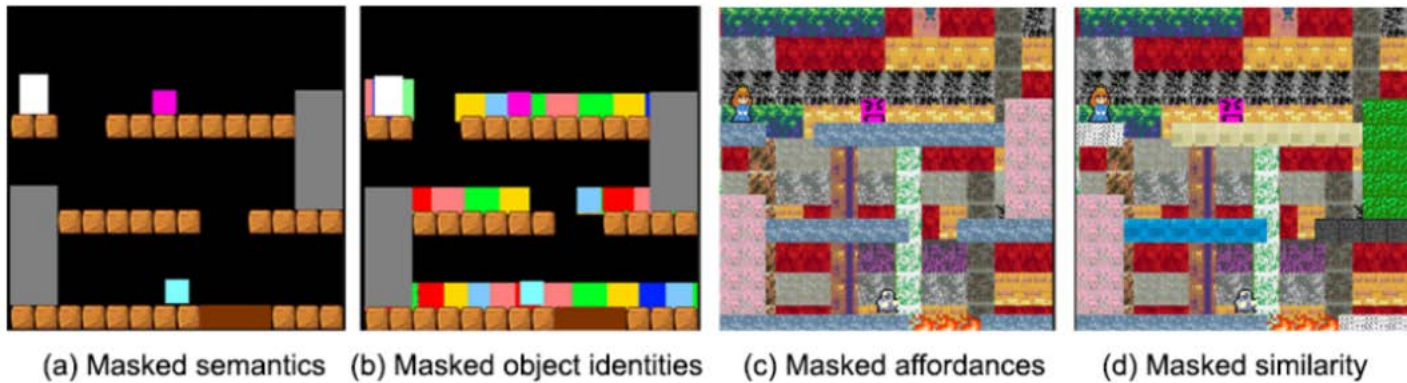
R. Dubey, P. Agrawal, D. Pathak, T. Griffiths, A. Efros, [Investigating Human Priors for Playing Video Games](#), ICML 2018

The importance of features



The importance of features

- RL agent is relatively unaffected by most transformations that hinder humans



RL challenges: Overview

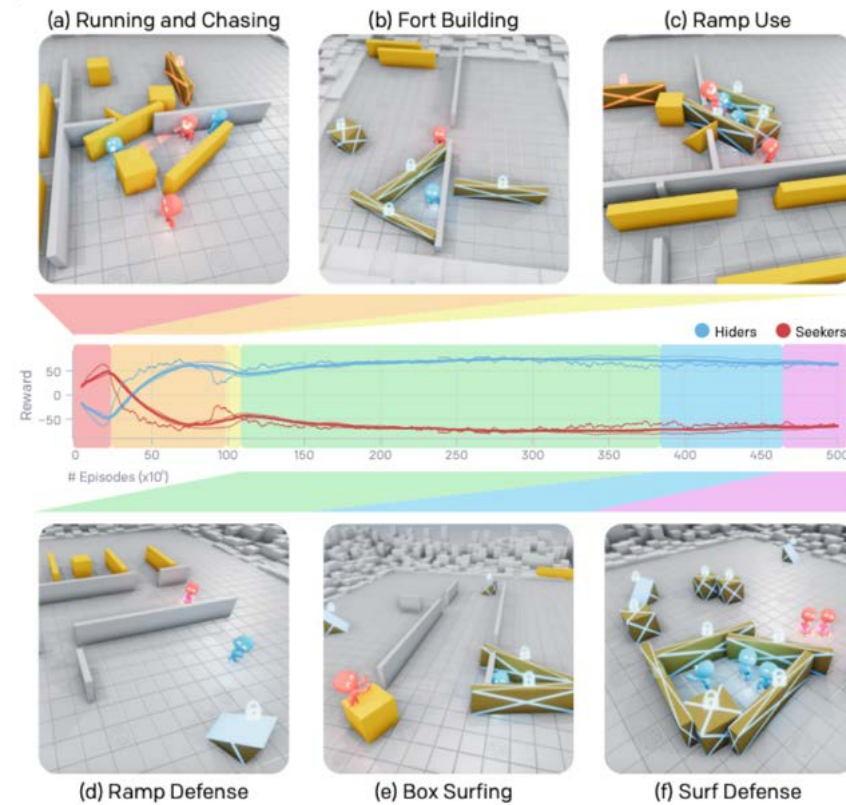
- Large state and action spaces, sparse rewards, planning many steps ahead
- Tasks that cannot be bootstrapped from random performance
- Not knowing the reward function up front
- Learning without a reward function *at all*
- Training faster, generalizing from one task to another
- Dealing with multiple agents: cooperation, competition, communication

Multiple agents and communication



[Source](#)

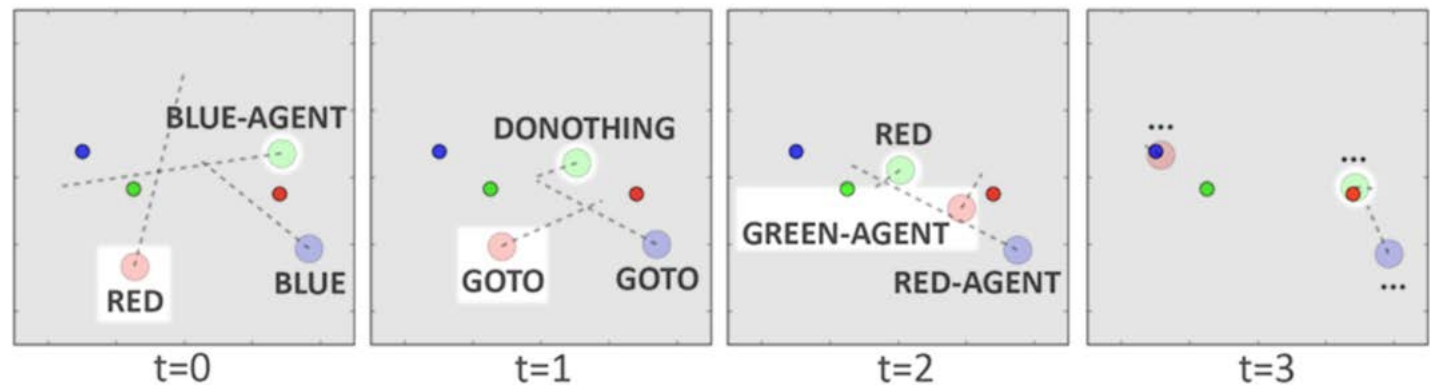
Hide and seek



<https://openai.com/blog/emergent-tool-use/>

B. Baker et al., [Emergent tool use from multi-agent autocurricula](#), ICLR 2020

Emergence of language

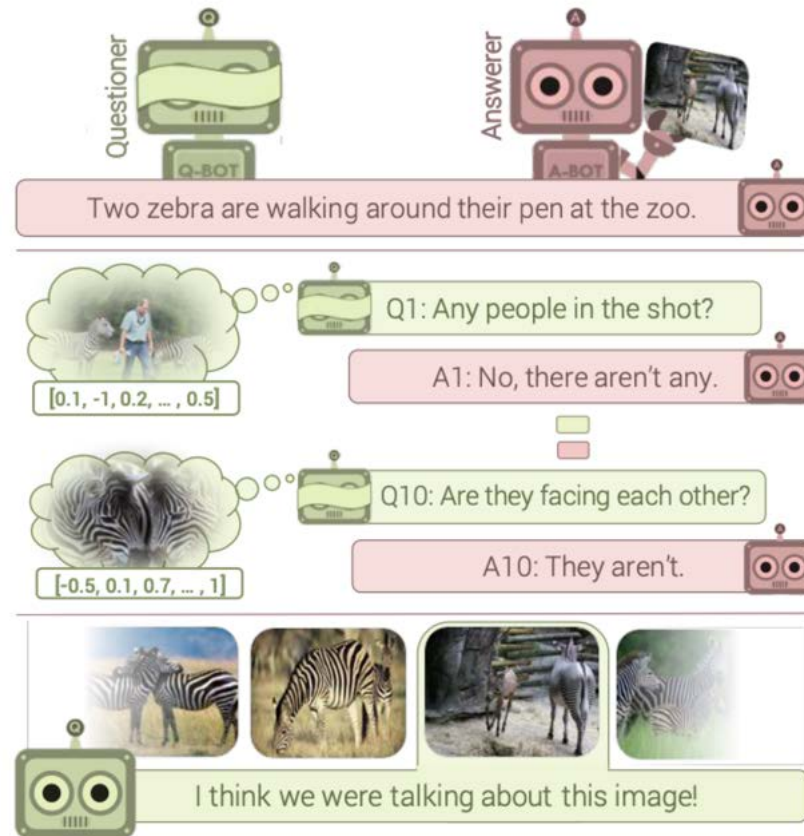


In the above step-by-step run, at $t=0$ the red agent says a word corresponding to the red landmark (center right), then at $t=1$ says a word that is equivalent to 'Goto', then in $t=2$ says 'green-agent'. The green-agent hears its instructions and immediately moves to the red landmark.

<https://openai.com/blog/learning-to-communicate/>
[Video](#)

A. Mordatch and P. Abbeel, [Emergence of Grounded Compositional Language in Multi-Agent Populations](#), AAI 2018

Visual dialog






A. Das et al., [Learning Cooperative Visual Dialog Agents with Deep Reinforcement Learning](#), ICCV 2017

RL for negotiation

Divide these objects between you and another Turker. Try hard to get as many points as you can!

Send a message now, or enter the agreed deal!

Items	Value	Number You Get
	8	<input type="text" value="1"/>
	1	<input type="text" value="1"/>
	0	<input type="text" value="0"/>

Mark Deal Agreed ✓

Dialogue interface showing messages:

- Fellow Turker: I'd like all the balls
- You: Ok, if I get everything else
- Fellow Turker: If I get the book then you have a deal
- You: No way - you can have one hat and all the balls
- Fellow Turker: Ok deal

Type Message Here:

Message Send

Figure 1: A dialogue in our Mechanical Turk interface, which we used to collect a negotiation dataset.

M. Lewis et al., Deal or No Deal? [End-to-End Learning for Negotiation Dialogues](#), EMNLP 2017

RL for negotiation

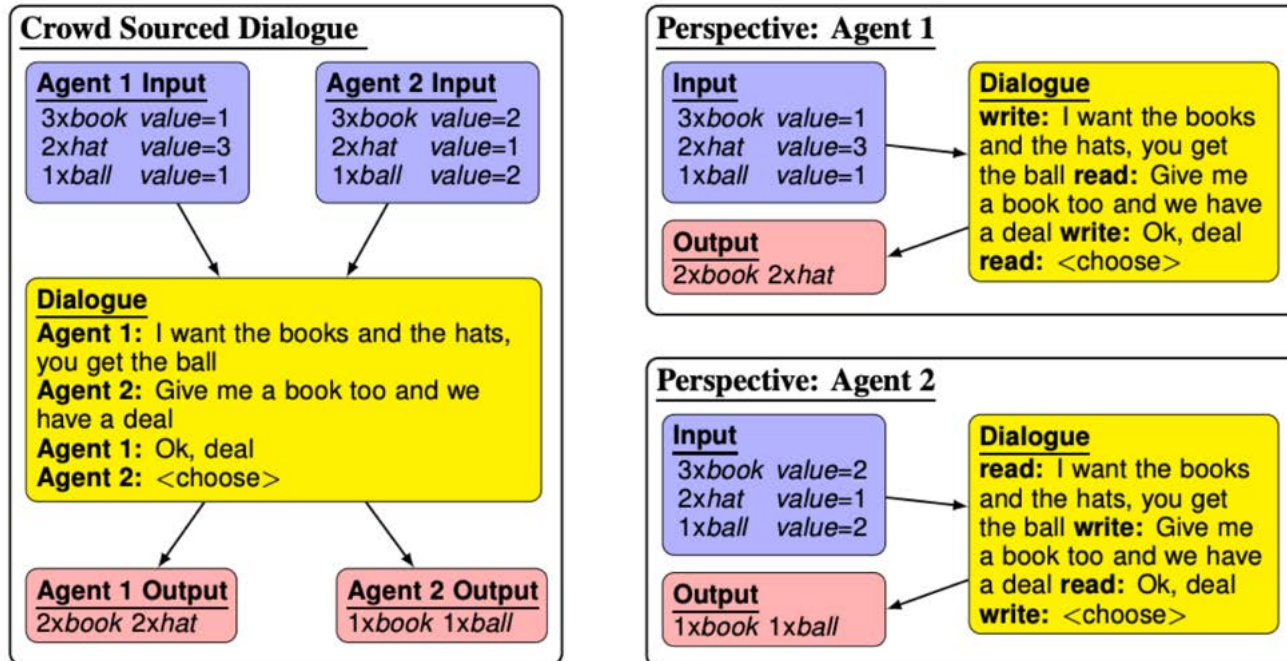


Figure 2: Converting a crowd-sourced dialogue (left) into two training examples (right), from the perspective of each user. The perspectives differ on their input goals, output choice, and in special tokens marking whether a statement was read or written. We train conditional language models to predict the dialogue given the input, and additional models to predict the output given the dialogue.

RL for negotiation

Input		
Human	1x book <i>value=3</i> 1x hat <i>value=1</i> 3x ball <i>value=2</i>	
RL+ROLLOUTS	1x book <i>value=6</i> 1x hat <i>value=4</i> 3x ball <i>value=0</i>	
RL+ROLLOUTS	You can have the balls if i can have the hat and book	
Human	Nope, give me the books and you can have everything else	
RL+ROLLOUTS	Ok. I will take the book and one hat	
Human	I get all the balls?	
RL+ROLLOUTS	Yes. I can do that.	
Human	Ok, deal	
Output		Reward
RL+ROLLOUTS	1x book 1x hat	10/10
Human	3x ball	6/10

Figure 5: Example of a model stubbornly holding out for maximum points in a dialogue.

Input		
RL+ROLLOUTS	4x book <i>value=0</i> 2x hat <i>value=5</i> 1x ball <i>value=0</i>	
Human	4x book <i>value=1</i> 2x hat <i>value=2</i> 1x ball <i>value=2</i>	
RL+ROLLOUTS	You can have the ball if I can have everything else.	
Human	No can't do that the ball is not going to give me anything	
RL+ROLLOUTS	I need the hats and you can have the rest	
Human	Deal	
RL+ROLLOUTS	Great! Thank you!	
Output		Reward
RL+ROLLOUTS	2x hat	10/10
Human	4x book 1x ball	6/10

Figure 7: Dialogue in which the model's initial interest in the valueless books allows it to compromise while achieving a maximum score.

RL for negotiation

Input		
Human	3x book <i>value=2</i> 3x hat <i>value=1</i> 1x ball <i>value=1</i>	
RL+ROLLOUTS	3x book <i>value=1</i> 3x hat <i>value=2</i> 1x ball <i>value=1</i>	
Human	I'd like the books and the hats.	
RL+ROLLOUTS	I will take the books and hats	
Human	I really need the books and at least one other object.	
RL+ROLLOUTS	Ok, you can have one book and one ball	
Human	If I can have all the books, I can leave you the rest.	
RL+ROLLOUTS	Deal	
Output		Reward
RL+ROLLOUTS	3x hat 1x ball	7/10
Human	3x book	6/10

Figure 6: Example of model compromising.

Hype and debunking

THE SUN, A NEWS UK COMPANY ▼

< DEAR DEIDRE | **TECH** | TRAVEL | MOTORS | PUZZLES | SUN BINGO | SUN

All Tech | Science | Phones & Gadgets | Gaming

ROBOSTOP Facebook shuts off AI experiment after two robots begin speaking in their OWN language only they can understand

Experts have called the incident exciting but also incredibly scary

James Beal | Andy Jehring
1 Aug 2017, 0:03 | Updated: 2 Aug 2017, 4:56

GIZMODO | We come from the future



HOME LATEST REVIEWS EARTHER SCIENCE IO9 FIELD GUIDE VIDEO

FACEBOOK

No, Facebook Did Not Panic and Shut Down an AI Program That Was Getting Dangerously Smart

 Tom McKay
7/31/17 8:31PM • Filed to: SKYNET ▼

153 14

<https://gizmodo.com/no-facebook-did-not-panic-and-shut-down-an-ai-program-1797414922>
[See also: Dhruv Batra's response](#)

Even more challenges

- Scaling RL in the physical world
- Planning
- Reasoning
- Integrating memory, knowledge