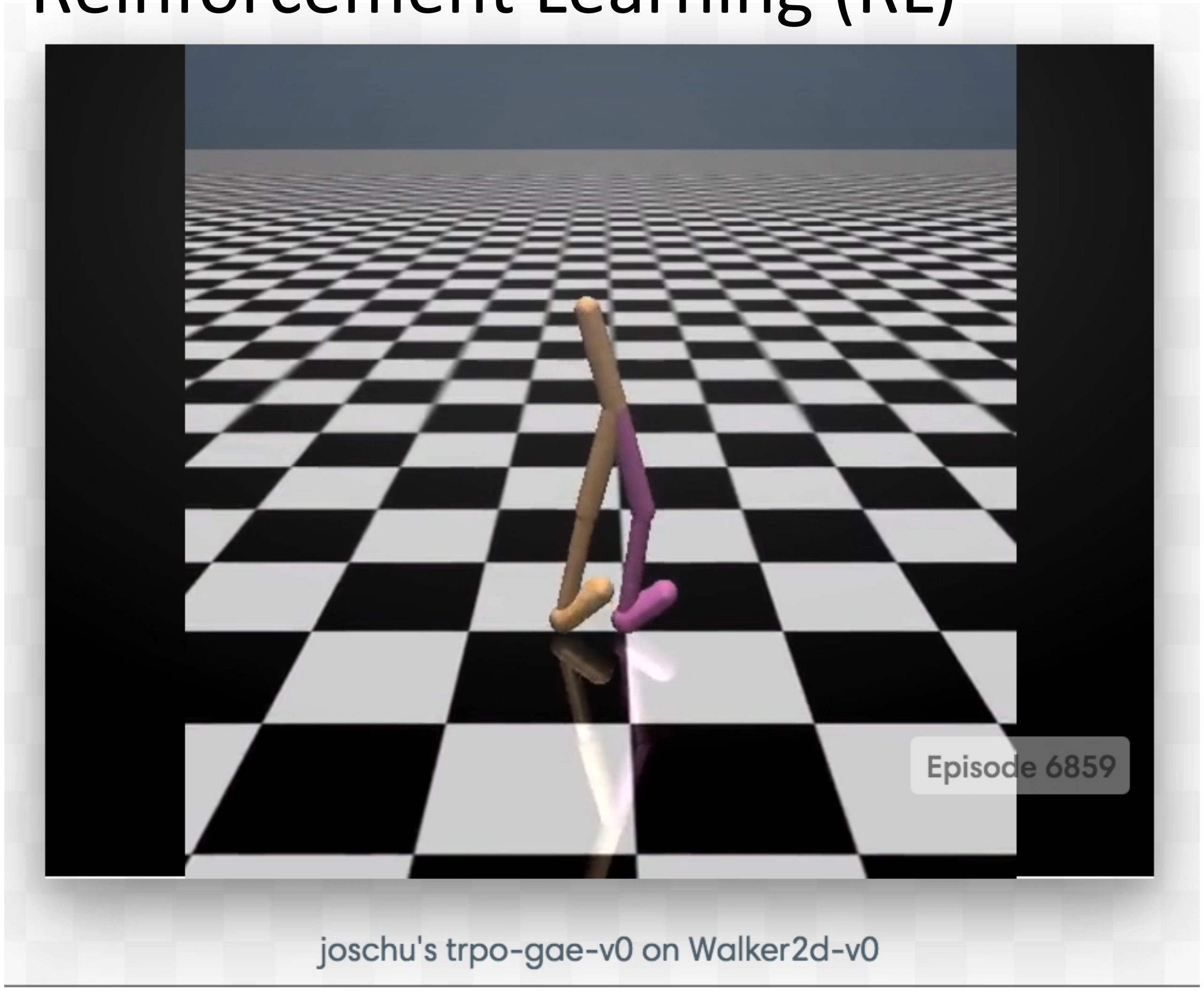
Reinforcement Learning

Min Sun
VSLab

Vision and Control

- Learning to play game with weak supervision:
  Reinforcement Learning (RL)

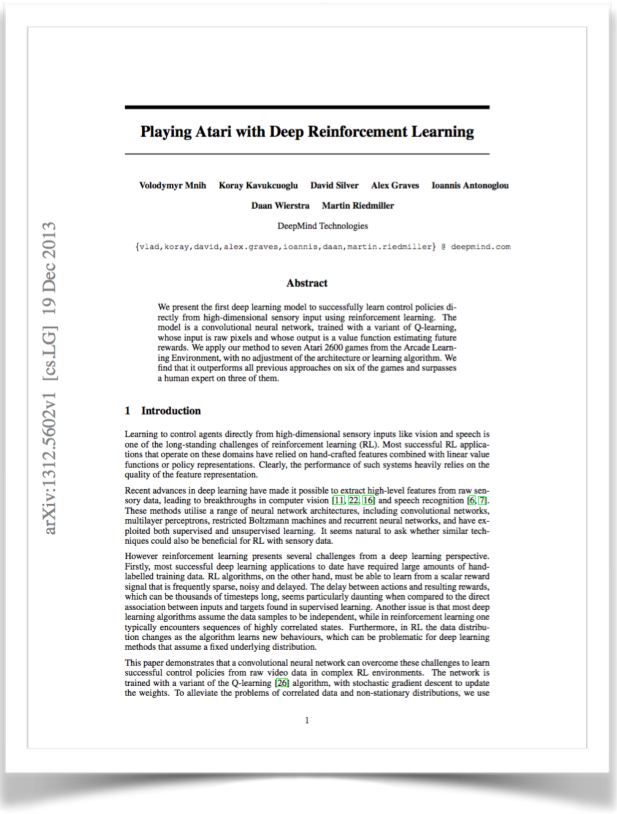


https://gym.openai.com/

Where It All Begins …

Playing Atari with Deep Reinforcement Learning

by DeepMind in NIPS 2013 Deep Learning Wrokshop



Playing Atari with Deep Reinforcement Learning

Volodymyr Mnih Koray Kavukcuoglu David Silver Alex Graves Ioannis Antonoglou
Daan Wierstra Martin Riedmiller
DeepMind Technologies
{vlad,koray,david,alex.graves,ioannis,daan,martin.riedmiller} @ deepmind.com

Abstract

We present the first deep learning model to successfully learn control policies directly from high-dimensional sensory input using reinforcement learning. The model is a convolutional neural network, trained with a variant of Q-learning, whose input is raw pixels and whose output is a value function estimating future rewards. We apply our method to seven Atari 2600 games from the Arcade Learning Environment, with no adjustment of the architecture or learning algorithm. We find that it outperforms all previous approaches on six of the games and surpasses a human expert on three of them.

1 Introduction

Learning to control agents directly from high-dimensional sensory inputs like vision and speech is one of the long-standing challenges of reinforcement learning (RL). Most successful RL applications that operate on these domains have relied on hand-crafted features combined with linear value functions or policy representations. Clearly, the performance of such systems heavily relies on the quality of the feature representation.

Recent advances in deep learning have made it possible to extract high-level features from raw sensory data, leading to breakthroughs in computer vision [11, 22, 16] and speech recognition [6, 7]. These methods utilise a range of neural network architectures, including convolutional networks, multilayer perceptrons, restricted Boltzmann machines and recurrent neural networks, and have exploited both supervised and unsupervised learning. It seems natural to ask whether similar techniques could also be beneficial for RL with sensory data.

However reinforcement learning presents several challenges from a deep learning perspective. Firstly, most successful deep learning applications to date have required large amounts of hand-labelled training data. RL algorithms, on the other hand, must be able to learn from a scalar reward signal that is frequently sparse, noisy and delayed. The delay between actions and resulting rewards, which can be thousands of timesteps long, seems particularly daunting when compared to the direct association between inputs and targets found in supervised learning. Another issue is that most deep learning algorithms assume the data samples to be independent, while in reinforcement learning one typically encounters sequences of highly correlated states. Furthermore, in RL the data distribution changes as the algorithm learns new behaviours, which can be problematic for deep learning methods that assume a fixed underlying distribution.

This paper demonstrates that a convolutional neural network can overcome these challenges to learn successful control policies from raw video data in complex RL environments. The network is trained with a variant of the Q-learning [26] algorithm, with stochastic gradient descent to update the weights. To alleviate the problems of correlated data and non-stationary distributions, we use

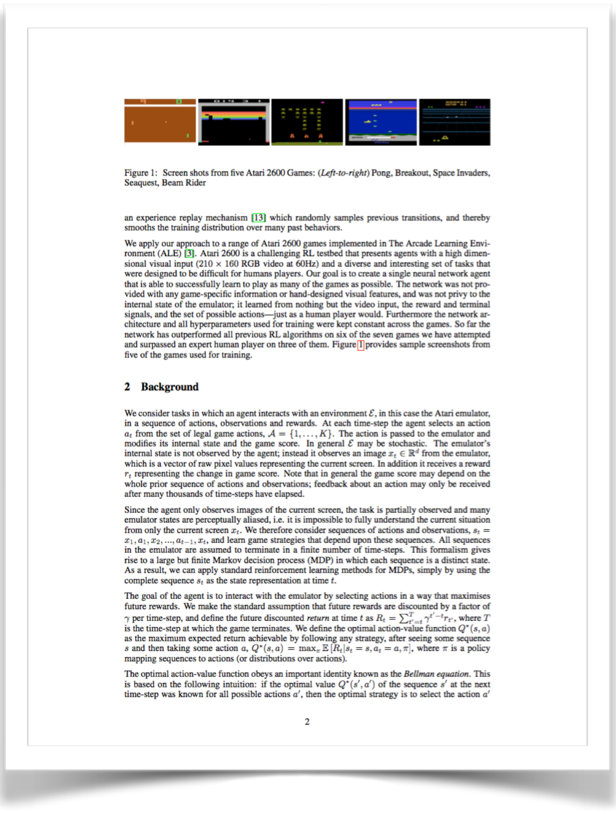


Figure 1: Screen shots from five Atari 2600 Games: (*Left-to-right*) Pong, Breakout, Space Invaders, Seaquest, Beam Rider

an experience replay mechanism [13] which randomly samples previous transitions, and thereby smooths the training distribution over many past behaviors.

We apply our approach to a range of Atari 2600 games implemented in The Arcade Learning Environment (ALE) [3]. Atari 2600 is a challenging RL testbed that presents agents with a high dimensional visual input (210×160 RGB video at 60Hz) and a diverse and interesting set of tasks that were designed to be difficult for humans players. Our goal is to create a single neural network agent that is able to successfully learn to play as many of the games as possible. The network was not provided with any game-specific information or hand-designed visual features, and was not privy to the internal state of the emulator; it learned from nothing but the video input, the reward and terminal signals, and the set of possible actions—just as a human player would. Furthermore the network architecture and all hyperparameters used for training were kept constant across the games. So far the network has outperformed all previous RL algorithms on six of the seven games we have attempted and surpassed an expert human player on three of them. Figure 1 provides sample screenshots from five of the games used for training.

2 Background

We consider tasks in which an agent interacts with an environment $\mathcal{E}$, in this case the Atari emulator, in a sequence of actions, observations and rewards. At each time-step the agent selects an action a_t from the set of legal game actions, $\mathcal{A} = \{1, \ldots, K\}$. The action is passed to the emulator and modifies its internal state and the game score. In general $\mathcal{E}$ may be stochastic. The emulator's internal state is not observed by the agent; instead it observes an image $x_t \in \mathbb{R}^d$ from the emulator, which is a vector of raw pixel values representing the current screen. In addition it receives a reward r_t representing the change in game score. Note that in general the game score may depend on the whole prior sequence of actions and observations; feedback about an action may only be received after many thousands of time-steps have elapsed.

Since the agent only observes images of the current screen, the task is partially observed and many emulator states are perceptually aliased, i.e. it is impossible to fully understand the current situation from only the current screen x_t. We therefore consider sequences of actions and observations, $s_t = x_1, a_1, x_2, ..., a_{t-1}, x_t$, and learn game strategies that depend upon these sequences. All sequences in the emulator are assumed to terminate in a finite number of time-steps. This formalism gives rise to a large but finite Markov decision process (MDP) in which each sequence is a distinct state. As a result, we can apply standard reinforcement learning methods for MDPs, simply by using the complete sequence s_t as the state representation at time t.

The goal of the agent is to interact with the emulator by selecting actions in a way that maximises future rewards. We make the standard assumption that future rewards are discounted by a factor of γ per time-step, and define the future discounted *return* at time t as $R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$, where T is the time-step at which the game terminates. We define the optimal action-value function $Q^*(s,a)$ as the maximum expected return achievable by following any strategy, after seeing some sequence s and then taking some action a, $Q^*(s,a) = \max_\pi \mathbb{E}\left[R_t | s_t = s, a_t = a, \pi\right]$, where π is a policy mapping sequences to actions (or distributions over actions).

The optimal action-value function obeys an important identity known as the *Bellman equation*. This is based on the following intuition: if the optimal value $Q^*(s',a')$ of the sequence s' at the next time-step was known for all possible actions a', then the optimal strategy is to select the action a'

slides by
Yen-Chen Lin

Control: Learning to Act

Play **Breakout** equals to

- **Input:** screen images
- **Output:** actions
  (**do nothing | left | right**)

Supervised Classification

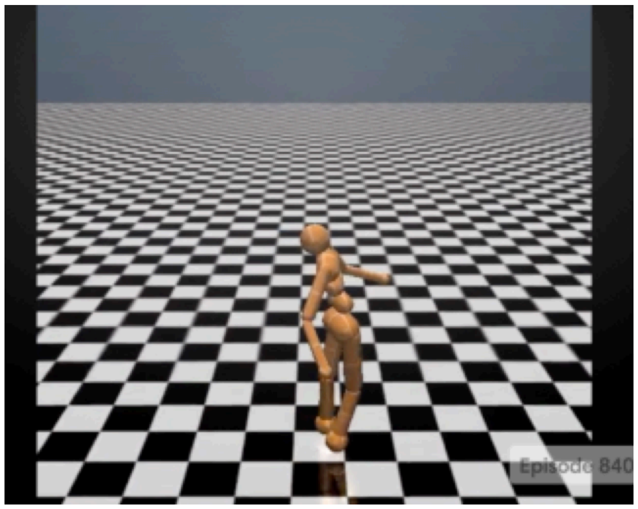
slides by
Yen-Chen Lin

Supervised Solution

- **Training data:** Record experts game sessions
- **Target label:** Action experts take at every step

⚠ Problems:

- What if there's no expert?
- This is not how human learns



slides by
Yen-Chen Lin

How Human Learns

- Don’t need somebody to tell us a million times which move to choose at each screen
- Just need **occasional feedback** that we did the right thing

slides by
Yen-Chen Lin

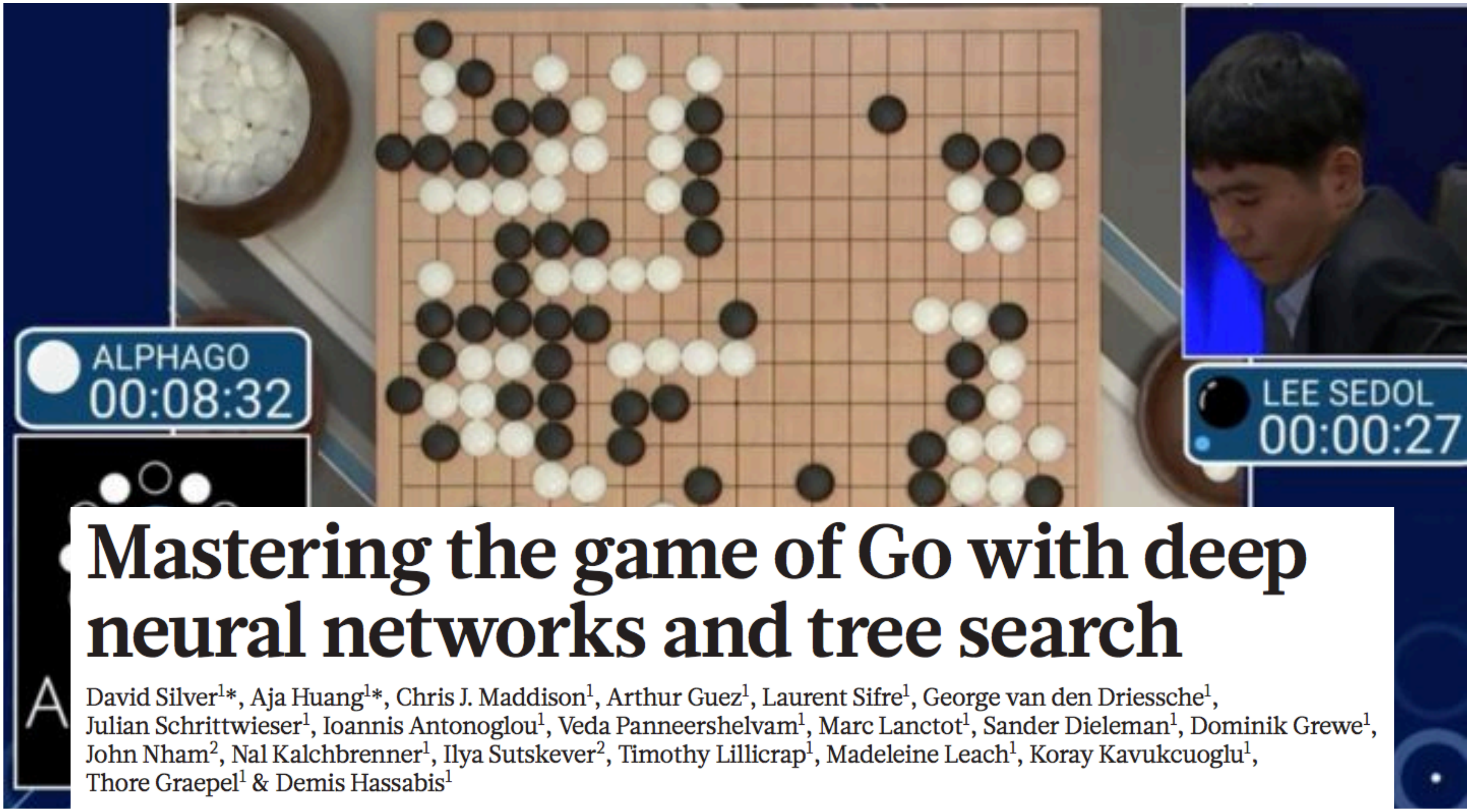
Reinforcement Learning

- Somewhere between **supervised** and **unsupervised** learning
- **Sparse** and **time-delayed** labels

Based only on those rewards, the agent has to learn to behave in the environment.
A rational agent should optimize total reward.

slides by
Yen-Chen Lin

AlphaGo



Mastering the game of Go with deep neural networks and tree search

David Silver[1]*, Aja Huang[1]*, Chris J. Maddison[1], Arthur Guez[1], Laurent Sifre[1], George van den Driessche[1], Julian Schrittwieser[1], Ioannis Antonoglou[1], Veda Panneershelvam[1], Marc Lanctot[1], Sander Dieleman[1], Dominik Grewe[1], John Nham[2], Nal Kalchbrenner[1], Ilya Sutskever[2], Timothy Lillicrap[1], Madeleine Leach[1], Koray Kavukcuoglu[1], Thore Graepel[1] & Demis Hassabis[1]

Lecture 1: Introduction to Reinforcement Learning

David Silver

Sequential Decision Making

- Goal: *select actions to maximise total future reward*
- Actions may have long term consequences
- Reward may be delayed
- It may be better to sacrifice immediate reward to gain more long-term reward
- Examples:
  - A financial investment (may take months to mature)
  - Refuelling a helicopter (might prevent a crash in several hours)
  - Blocking opponent moves (might help winning chances many moves from now)