Reinforcement Learning

Andrei Nica, Tudor Berariu

Outline

Reinforcement Learning Introduction

Reward driven agent behavior

Challenges

Approaches

RL agents taxonomy

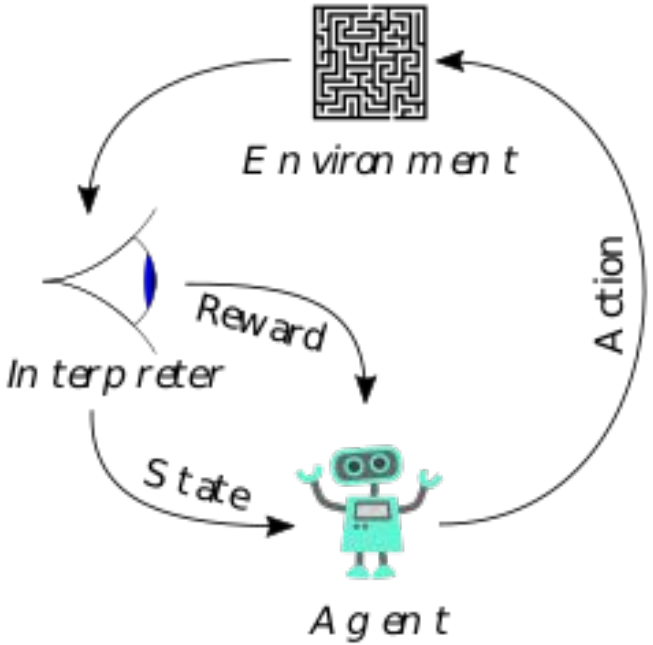
Deep RL

Reinforcement Learning Introduction

Reinforcement learning framework

Elements:

- Environment
- Agent
- Observations, state
- Reward signal
- *Policy*
- *Model of the environment*



https://en.wikipedia.org/wiki/Reinforcement_learning

Why study RL?

General solution

End-to-end learning from raw data to goal

Agent-based model

Enable real-world sequential decision making

The reinforcement learning problem is the AI problem!

Why study RL?

What are humans driven by?

Learning - basis of intelligence?

Learning how we learn

A single algorithm?

Model for testing our hypothesis

Why now?

1. Advancements in deep learning

2. Advancements in reinforcement learning

3. Advancements in computational capability

Results like: Atari games, Robotics, Beating Go & solving problems in production

History

Origin threads (1950s):

- behaviourist psychology
- optimal control problems
- *temporal difference learning*

60s, 70s

- Short decline - confusion RL / supervised learning in NN

70s

- Revival - framework importance - Harry Klopf

1989 - RL merge - Chris Watkins's invented the Q-learning algorithm

Applications

Think of any problem?

- Robotics
- Autonomous driving
- Language & dialogue
- Business operations
- Advertising
- Finance

Reward driven agent behavior

MDP

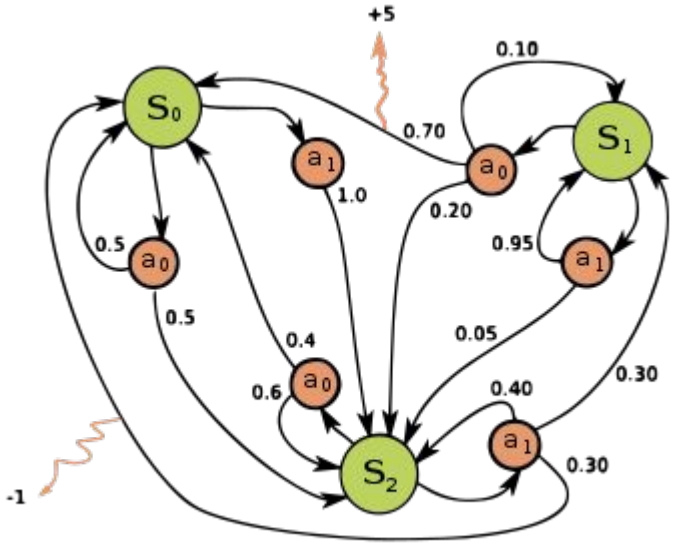
Defined by tuple $(S, A, P_\cdot(\cdot,\cdot), R_\cdot(\cdot,\cdot), \gamma)$

S - finite state space

A - finite action space

P - transition function

R - reward function



https://en.wikipedia.org/wiki/Markov_decision_process

Markov property:

Future is conditionally independent of the past given the present

Goal

Find policy that maximizes expected discounted sum over a potentially infinite horizon

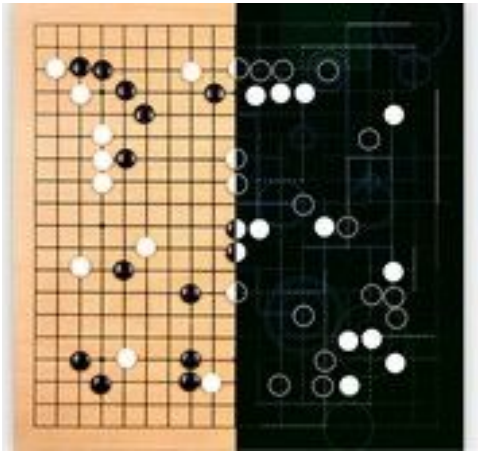
$$\sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1})$$

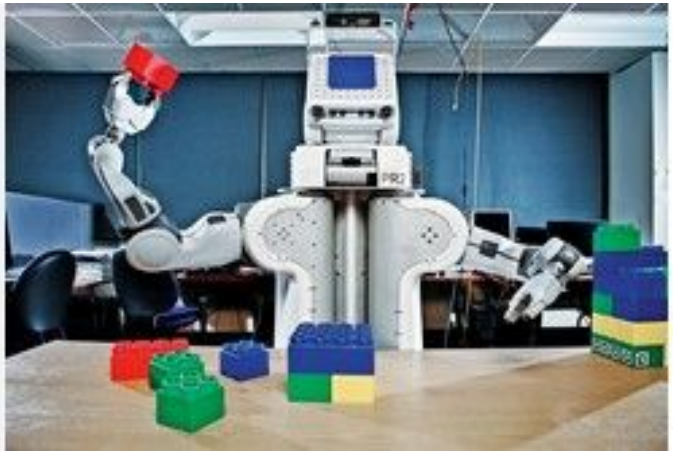
Each time-step ***t*** - agent observes state ***s*** -
Takes action ***a*** following policy ***π*** -
Receives reward ***r***

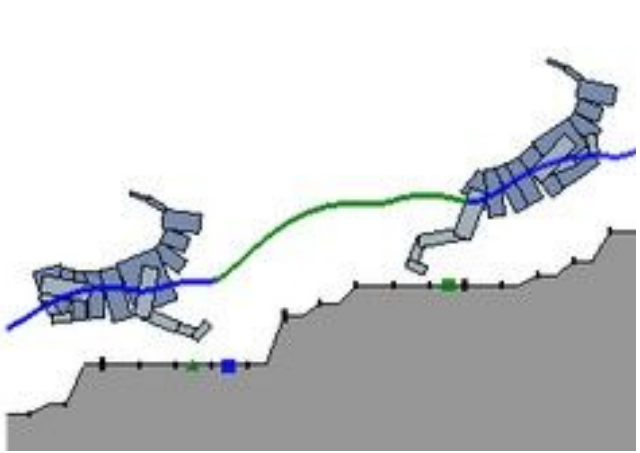
Problem examples

Observation: binary matrix | joint coordinates, 2D camera frame | Joint coordinates & 2D image

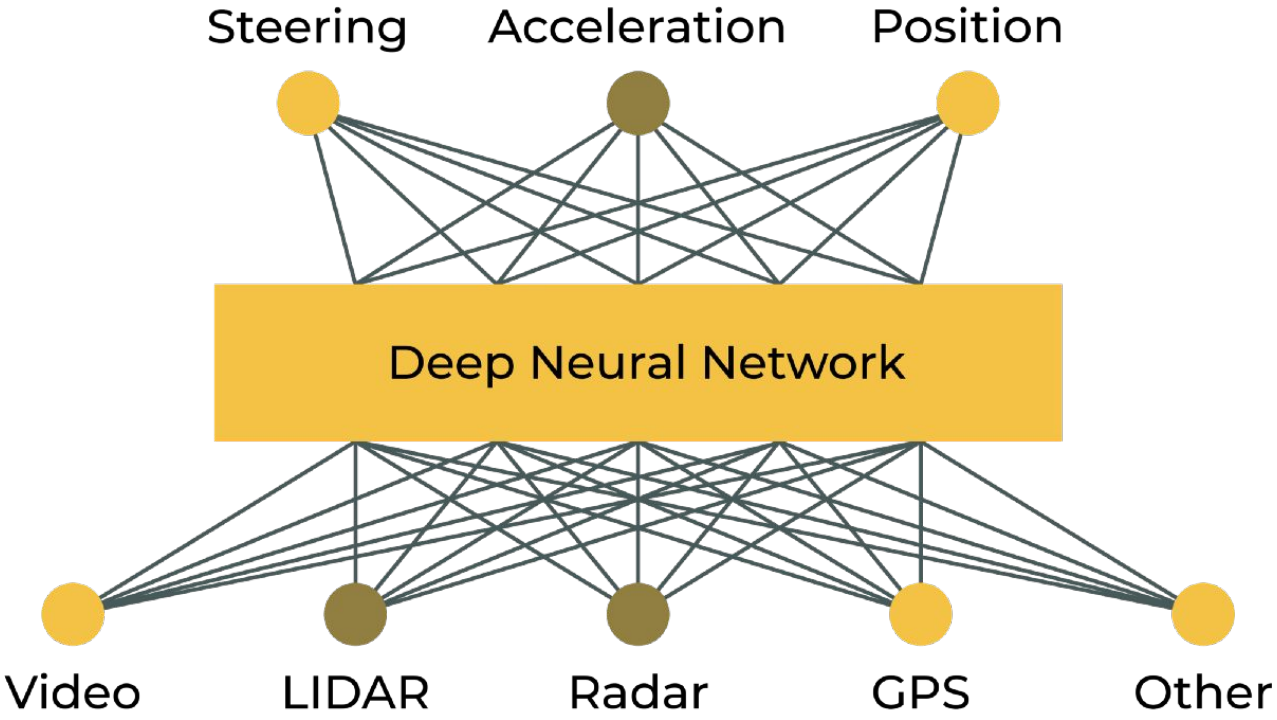
Reward: win/lose | difference with goal shape | maximum distance reached horizontally







And why not more?



https://medium.com/autocomm-technical-blog/end-to-end-autonomous-driving-a-not-so-novel-paradigm-42d81dcdb977

Challenges

Curse of dimensionality

Full {State x Action} space too big to experience.

Root of many other challenges

Tabular? Feature extraction?

*Go game - lower bound number of legal moves ~2 * 10^170*

Exploration vs expoitation

Hard maze

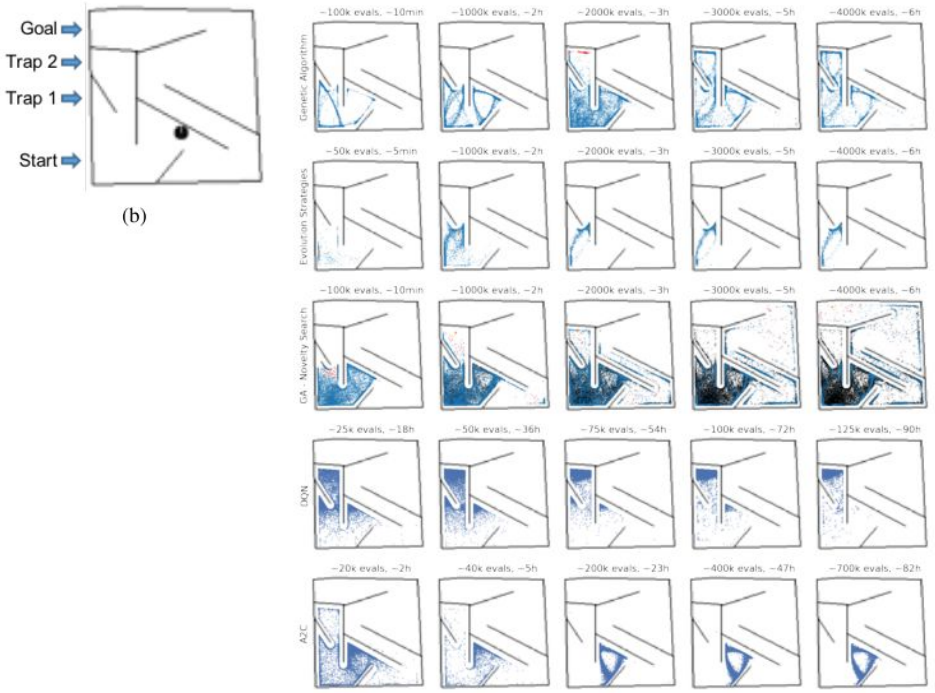


Figure 5. **How different algorithms explore the deceptive Image Hard Maze over time.** Traditional reward-maximization algorithms do not exhibit sufficient exploration to avoid the local optimum (going up). In contrast, a GA optimizing for novelty only (GA-NS) explores the entire environment and ultimately finds the goal. For the evolutionary algorithms (GA-NS, GA, ES), blue crosses represent the population (pseudo-offspring for ES), red crosses represent the top T GA offspring, orange dots represent the final positions of GA elites and the current mean ES policy, and the black crosses are entries in the GA-NS archive. All 3 evolutionary algorithms had the same number of evaluations, but ES and the GA have many overlapping points because they revisit locations due to poor exploration, giving the illusion of fewer evaluations. For DQN and A2C, we plot the end-of-episode position of the agent for each of the 20K episodes prior to the checkpoint listed above the plot.

"Deep Neuroevolution: Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning" (https://arxiv.org/abs/1712.06567)

Partial observability

State != observation

Uncertainty of the true information

POMDP - build belief state

Considered more in Multi agent systems

Missing information - from all observations

E.G. Empty map with agent - orientation

Stochastic or deterministic

Reward, Transition, Actions

Inherent randomness

Still maximize expected return.

Continuous or discrete

State or Action space

Algorithms - different approaches

Q-Learning & TD learning - discrete

Possible solution - discretization (coarse coding or tile coding)

Credit assignment problem

Temporal credit assignment from environment

Feature credit assignment

MAS - agent credit assignment

Change behavior for learning - *E.g. maze of colored tiles*

Non-stationary environments

Moving goal

Limited by how fast can a agent learning - Can make learning impossible

MAS - learn beside other agents

Generalization

Property or goal of learning?

Cannot experience everything

Compositionality

Approaches

Basic notations

$$\pi(s) : \mathbb{S} \rightarrow \mathbb{A}$$

$$total\ discounted\ reward = \sum_{i=1}^{T} \gamma^{i-1} r_i$$

$$V^{\pi}(s) = \mathbb{E}[\sum_{i=1}^{T} \gamma^{i-1} r_i] \quad \forall s \in \mathbb{S}$$

$$V^{*}(s) = \max_{\pi} V^{\pi}(s) \quad \forall s \in \mathbb{S}$$

$$\pi^{*} = \arg\max_{\pi} V^{\pi}(s) \quad \forall s \in \mathbb{S}$$

$$Q : \mathbb{S} \times \mathbb{A} \rightarrow \mathbb{R}$$

Dynamic programming

Solving a complex problem by breaking it down into a collection of simpler subproblems

Compute the optimal policy but only given the perfect model of the environment as a MDP.

TD learning

Updates some estimates of a function by using prior estimates

Bootstrapping model. No model of environment needed.

Simple TD(0) for Value function

Most SOA based on this idea. Hard to experience entire episode & more sample efficient

Tabular TD(0) for estimating v_π

Input: the policy π to be evaluated
Initialize $V(s)$ arbitrarily (e.g., $V(s) = 0$, for all $s \in \mathcal{S}^+$)
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    $A \leftarrow$ action given by π for S
    Take action A, observe R, S'
    $V(S) \leftarrow V(S) + \alpha\big[R + \gamma V(S') - V(S)\big]$
    $S \leftarrow S'$
  until S is terminal

Policy Iteration

Solve MDP. Faster to learn than value iteration *

2 steps: Evaluate & Improve

Policy iteration (using iterative policy evaluation)

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Repeat
   $\Delta \leftarrow 0$
   For each $s \in \mathcal{S}$:
   $v \leftarrow V(s)$
   $V(s) \leftarrow \sum_{s',r} p(s', r|s, \pi(s))\big[r + \gamma V(s')\big]$
   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   old-action $\leftarrow \pi(s)$
   $\pi(s) \leftarrow \operatorname{argmax}_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$
   If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Value Iteration

Offline planning (knowledge of the world)

Policy determined from value

Value iteration

Initialize array V arbitrarily (e.g., $V(s) = 0$ for all $s \in \mathcal{S}^+$)

Repeat
  $\Delta \leftarrow 0$
  For each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \max_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma V(s')\big]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, $\pi \approx \pi_*$, such that
  $\pi(s) = \operatorname{argmax}_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma V(s')\big]$

Monte Carlo Method

Classic method for estimating value function and discovering the optimal policy only from experience.

Value of states = average sampled returns

Use of samples inefficiently.

Cannot replicate on some environments. (Same starting point)

Beating Go champions: Supervised learning + policy gradients + value functions + Monte Carlo tree search: D. Silver et al. “Mastering the game of Go with deep neural networks and tree search”. Nature (2016).

Q-Learning

$$\pi^*(s) = \arg\max_a Q^*(s,a) \quad \forall s \in \mathbb{S}$$

$$Q^*(s,a) = R(s,a) + \gamma\, \mathbb{E}_{s'}[V^*(s')]$$
$$Q^*(s,a) = R(s,a) + \gamma \sum_{s' \in \mathbb{S}} p(s'|s,a) V^*(s')$$

Since,

$$V^*(S) = \max_a Q^*(s,a)$$

$$V^*(S) = \max_a \left[R(s,a) + \gamma \sum_{s' \in \mathbb{S}} p(s'|s,a) V^*(s') \right]$$

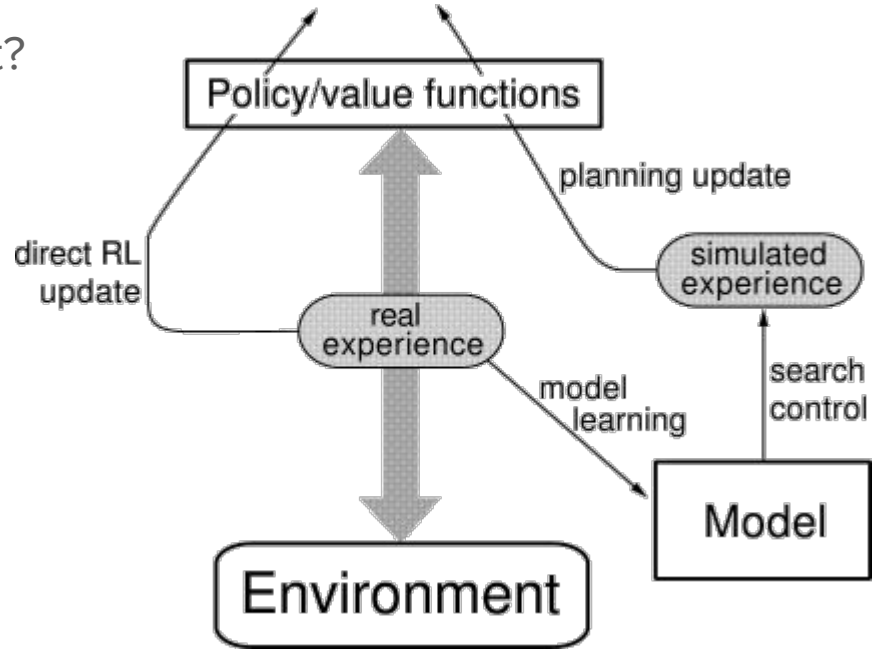
$$Q(s_t,a_t) \leftarrow (1-\alpha) \cdot \underbrace{Q(s_t,a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1},a)}_{\text{estimate of optimal future value}} \right)}^{\text{learned value}}$$

Planning and learning

What if we had a model of the environment?

How can we use it?

Can we learn the model?



RL agents taxonomy

RL agents taxonomy

Value | Policy driven

Model-based | Model-free

On-policy | Off-Policy

Deep learning

What changed

Appearance of large labeled data

Massively parallel computing with GPUs

Backprop-friendly activation functions (ReLu, ELU, LeakyRelu, SELU ...)

Improved architectures (Resnets, inception modules, and Highway networks ...)

Software platforms - Frameworks (Tensorflow, Pytorch, CNTK, Caffe ...)

New regularization techniques (dropout, batch normalization, layer norm ...)

Robust optimizers (Adam, Adadelta, RMSprop, ...)

Deep RL

Deep reinforcement learning

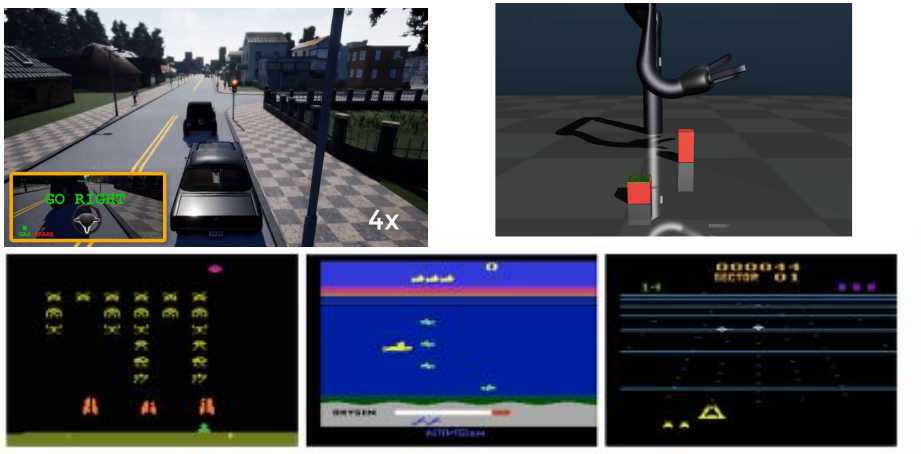
Deep models allow RL algorithms to solve complex problems

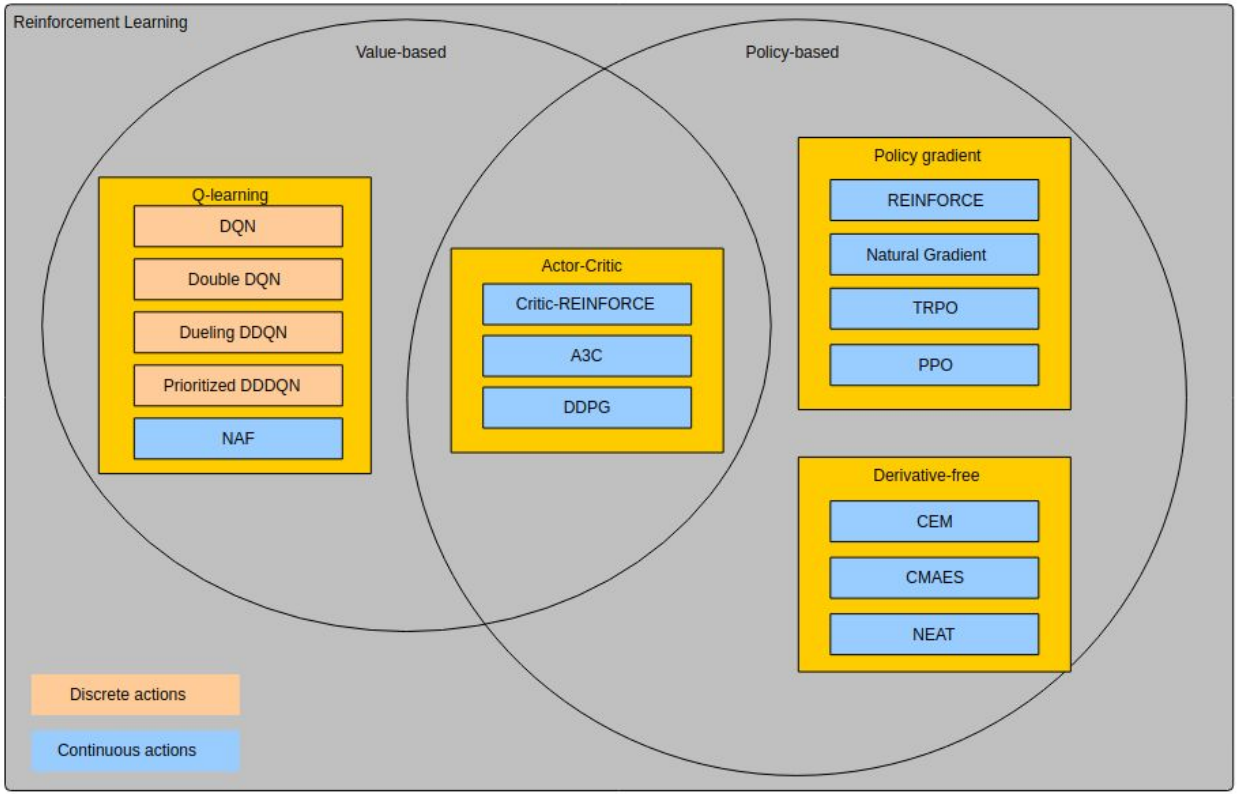
- End to end learning
- Directly from raw observations
- Process complex sensory input
- Compute really complex functions
- Choose complex actions

Environments

Benchmark

- ALE Atari 2600 (57 games)
- Starcraft II
- DeepMind Lab (Quake III Arena)
- MuJoCo
- Carla & Microsoft AirSim
- Safety environments
- Behavioural psychology
- House environments
- City simulators





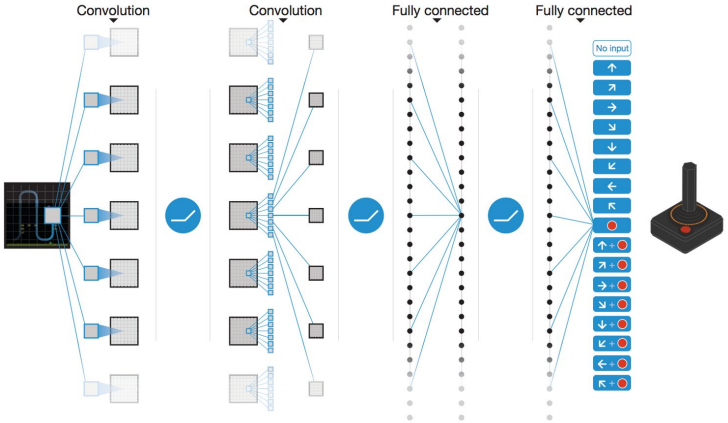
https://github.com/eleurent/phd-bibliography

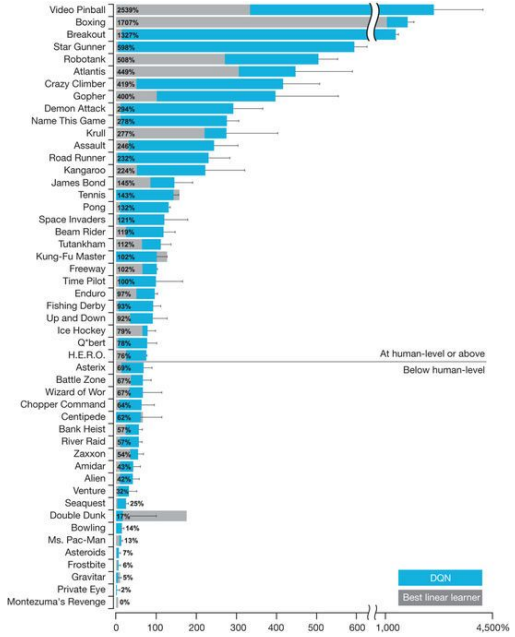
Value driven

Estimate value function or Q-function of the optimal policy

DQN - 2015

“Human-level control through deep reinforcement learning”



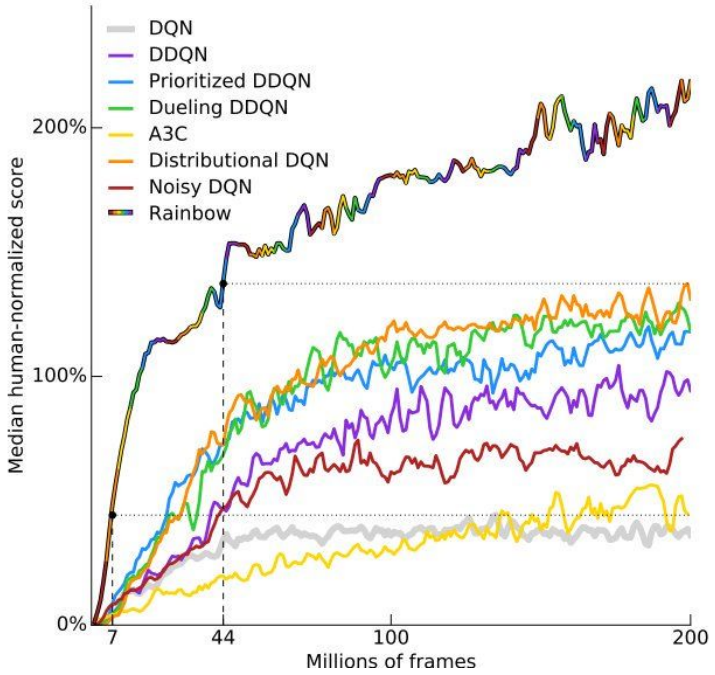


Value driven

Rainbow network (DeepMind)

“Rainbow: Combining Improvements in Deep Reinforcement Learning”

- DQN
- Prioritized replay
- Multi-step learning
- Noisy DQN
- Dueling DQN
- Double DQN
- Distributional DQN



Policy driven

Policy gradients: directly differentiate $\theta^\star = \arg\max_\theta E_{\tau \sim p_\theta(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right]$

Reinforce; “Parameterization trick”

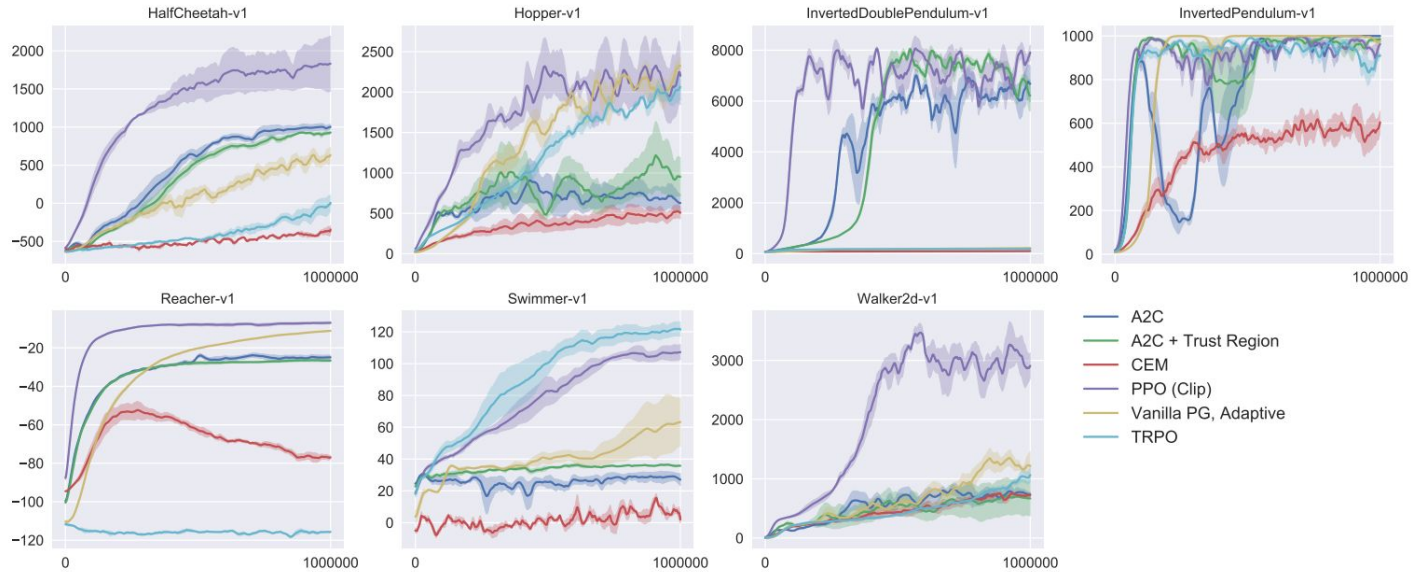
Examples:

- Reinforce
- Trust region Policy Optimization (TRPO)
- Proximal Policy Optimization (PPO)
- Natural gradient
- Distributed Proximal Policy Optimization (DPPO)

PPO

“Proximal Policy Optimization Algorithms”

Best at continuous action space



Actor-Critic

Estimate value function or Q-function of the current policy, use it to improve policy

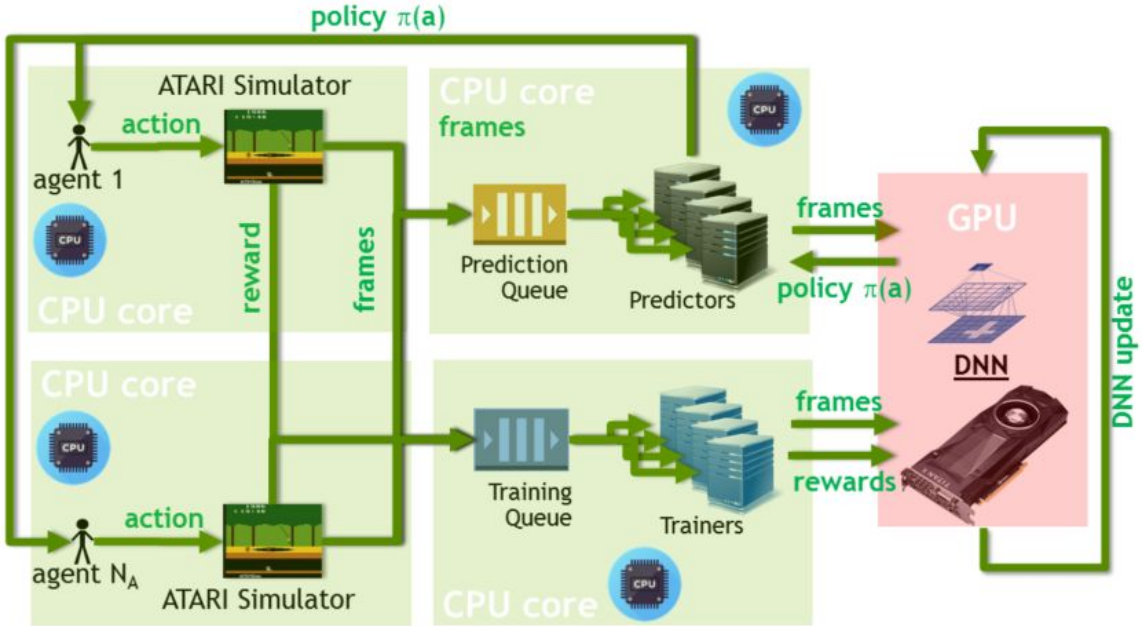
Best of both worlds

Examples:

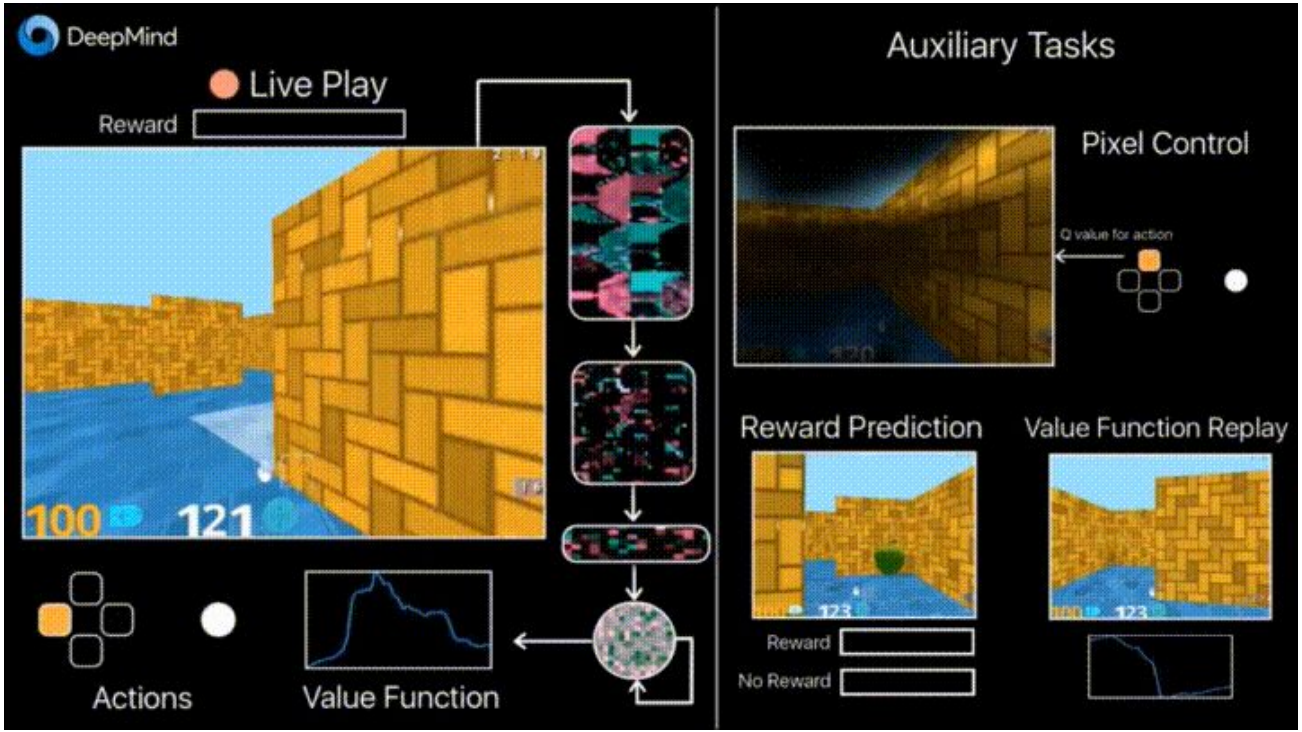
- Critic-Reinforce
- Deep deterministic policy gradient (DDPG)
- Asynchronous Advantage Actor-Critic (A3C)
- GPU A3C

GA3C

“GA3C: Reinforcement Learning through Asynchronous Advantage Actor-Critic on a GPU”



Auxiliary tasks

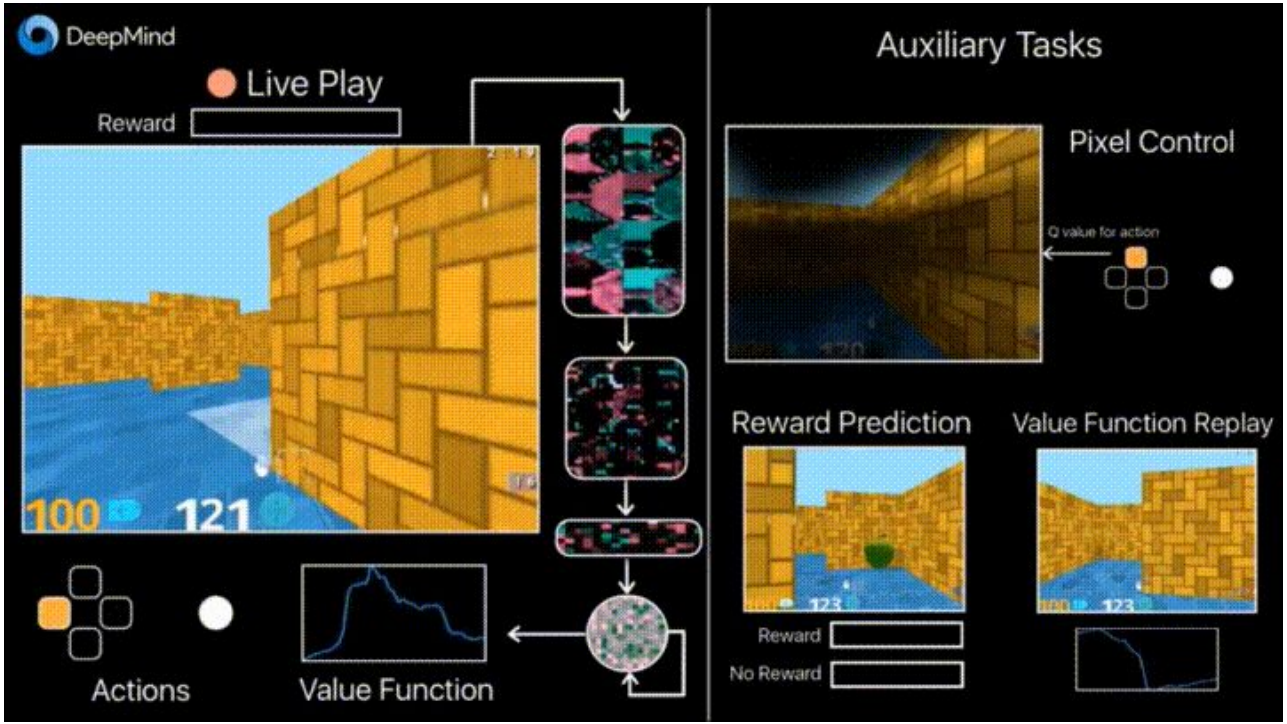


Auxiliary tasks

UNREAL Agent

“Reinforcement Learning with Unsupervised Auxiliary Tasks”

- Base A3C Agent
- Pixel Control
- Reward Prediction
- Value Function Replay



Future work

Future work

What are some challenges DRL and possible solutions

- Reward assignment; Slow learning
- Transfer learning; Multi-task learning; Imitation learning; Inverse RL; Hierarchical learning; Lifelong learning; Meta learning; Curriculum learning

Multi-agent systems

E.g. Communication emergence in MAS