Applied Deep Learning

Deep Reinforcement Learning

Alexander Pacha - TU Wien

Recap

- What do we mean when we talk about the Capacity of a ML model?
- What affects the effective capacity?
- Can increasing the training set size improve performance?

Reinforcement Learning

Reinforcement Learning
Learn how to make good sequences of decisions

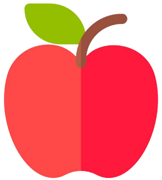
Classes of Learning Problems

Supervised Learning

Data: (x, y)
x is data, y is label

Goal: Learn function to map
$x \mapsto y$

Example:



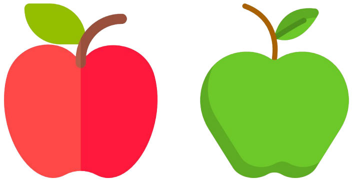
This thing is an apple

Unsupervised Learning

Data: x
x is data, no labels

Goal: Learn underlying structure (distribution)

Example:



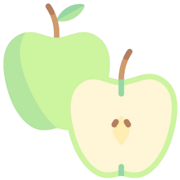
These two things are similar

Reinforcement Learning

Data: state-action pairs

Goal: Maximize future rewards over many time steps

Example:



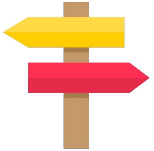
Eat this thing because it will keep you alive

Source: [1]

Challenges in Reinforcement Learning

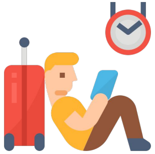
Optimization

Trying to make good decisions that yield the best outcome



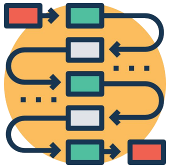
Delayed consequences

Rewards are received long after taking a certain decision



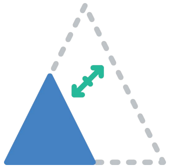
Exploration

Trying to learn how the world works by trying (and failing)

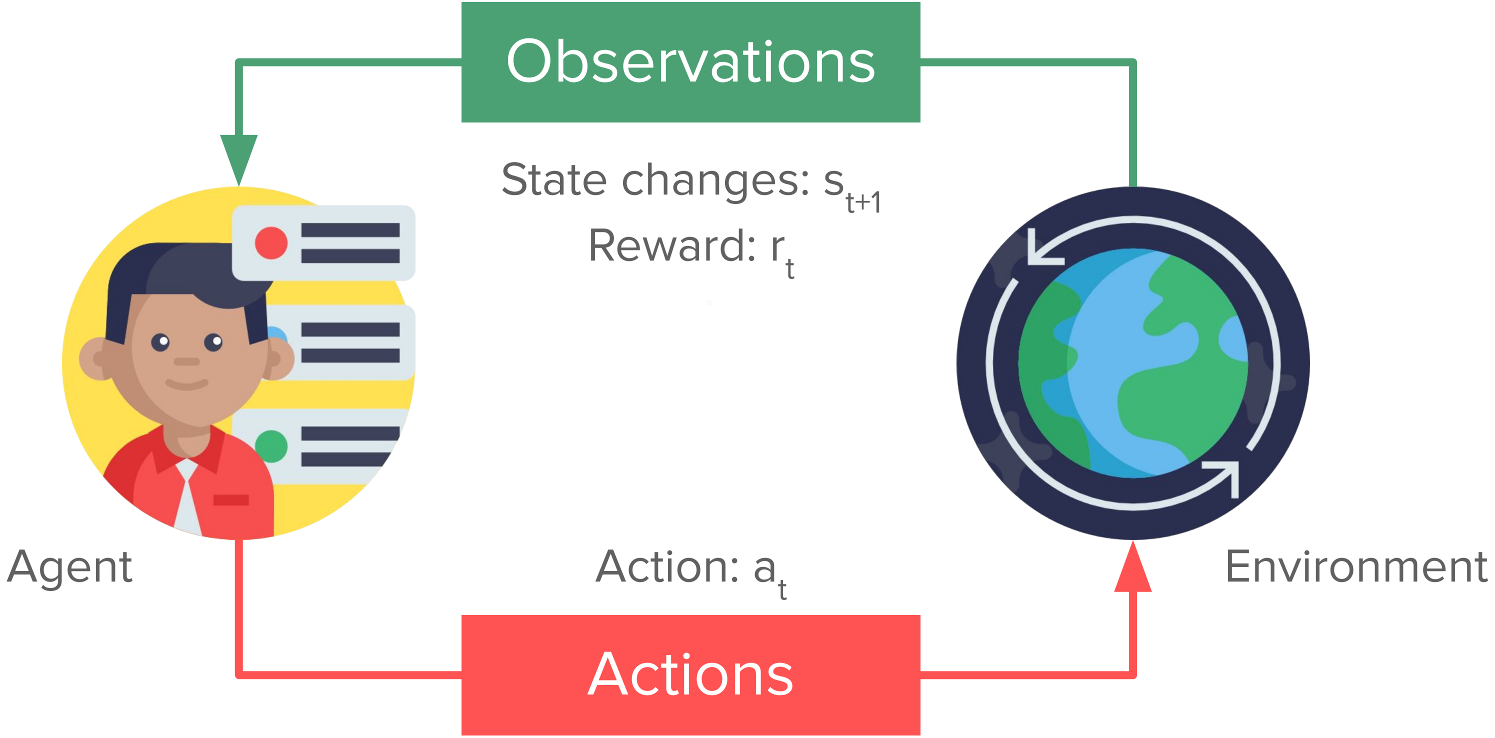


Generalization

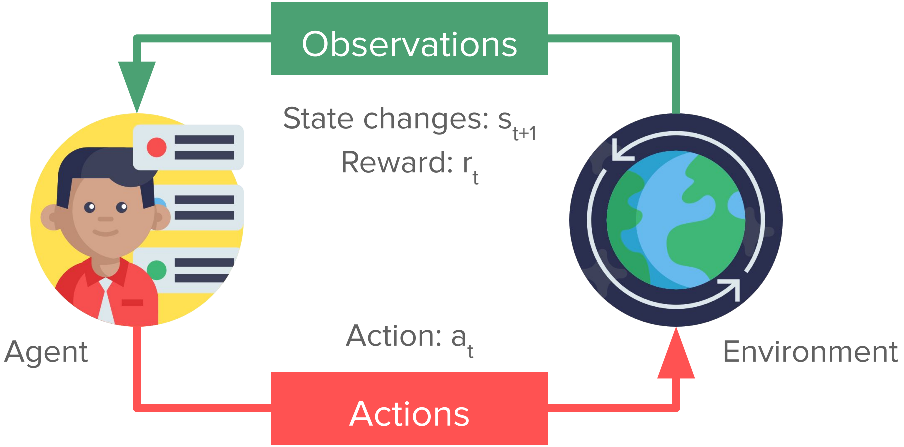
Mapping past experience to actions even if situation has never been seen



Key Concepts



Key Concepts



Total reward:

$$R_t = \sum_{i=t}^{\infty} r_i = r_t + r_{t+1} \dots + r_{t+n} + \cdots$$

Discounted Total reward:

$$R_t = \sum_{i=t}^{\infty} \gamma^i r_i = \gamma^t r_t + \gamma^{t+1} r_{t+1} \dots + \gamma^{t+n} r_{t+n} + \cdots$$

γ: Discount factor $\in [0, 1]$

Key Concepts

Quality Function

$Q(s, a)$

Rewards that you expect when you perform action a in state s

Value Function

$V(s)$

Rewards that you expect when you are currently in state s

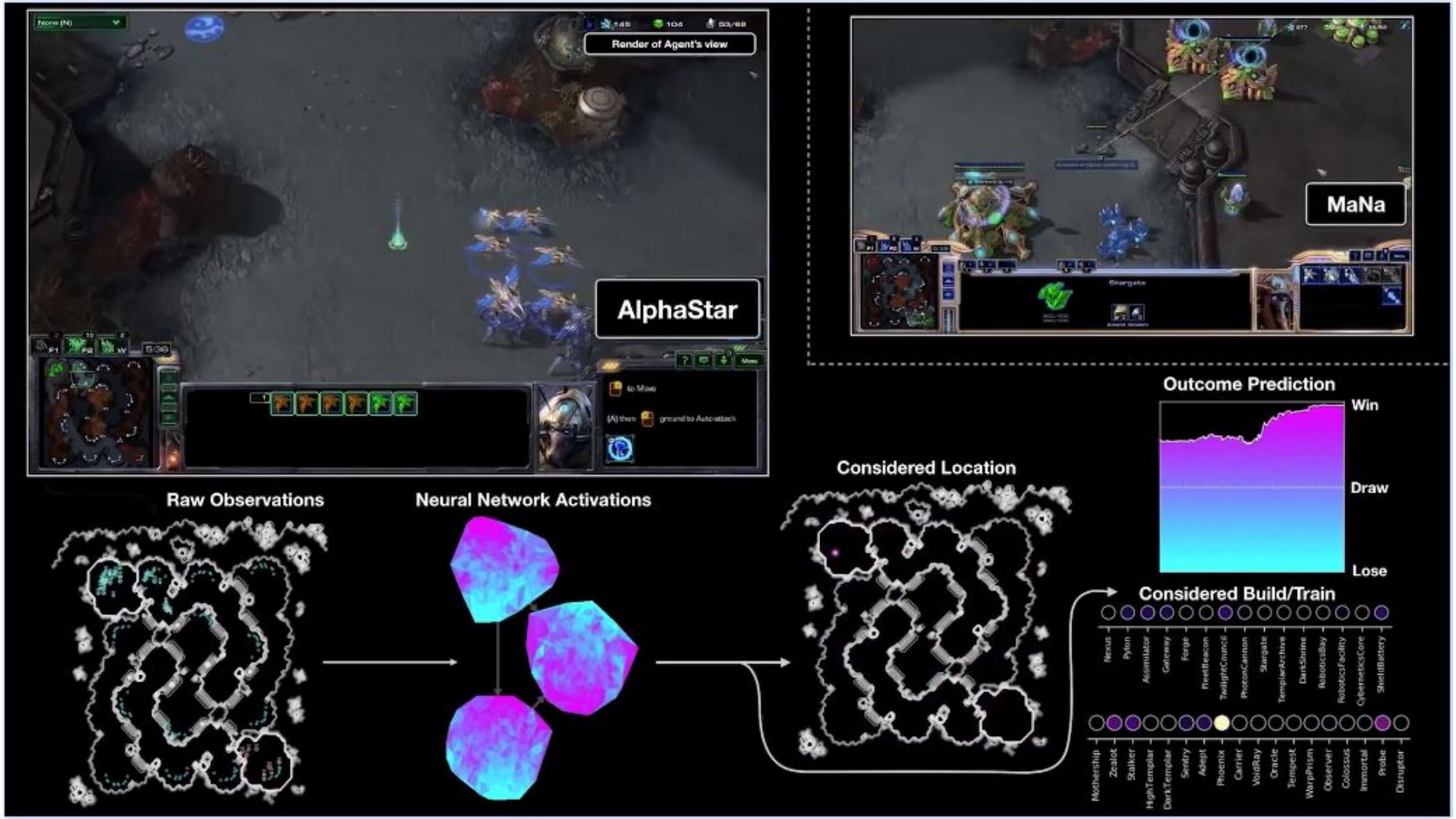
Policy Function

$\pi(a \mid s)$ or $\pi(s)$

Probability that action a is the best option in state s

Advantage function $A(s, a) = Q(s, a) - V(s)$

Value Function



From the Q-function to an Action

Discounted total rewards: $R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots$

$$Q(s, a) = \mathbb{E}[R_t]$$

Q-function captures **expected total future reward** an agent can achieve when taking action a in state s.

Given a Q-function, a policy function can be chosen to maximize future rewards:

$$\pi(s) = \underset{a}{\operatorname{argmax}}\, Q(s, a)$$

Deep Reinforcement Learning

Deep Reinforcement Learning Algorithms

Value Learning

Find **Q(s, a)**

$$a = \underset{a}{\operatorname{argmax}}\, Q(s, a)$$

Policy Learning

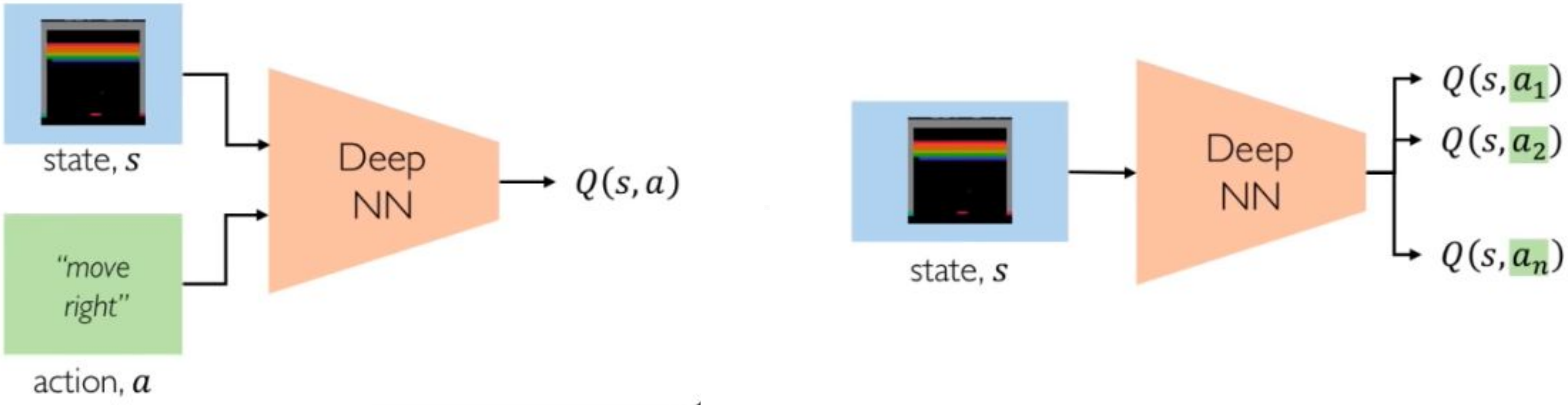
Find **π(s)**

Sample $a \sim \pi(s)$

Source: [1]

Deep Q Networks (DQN)

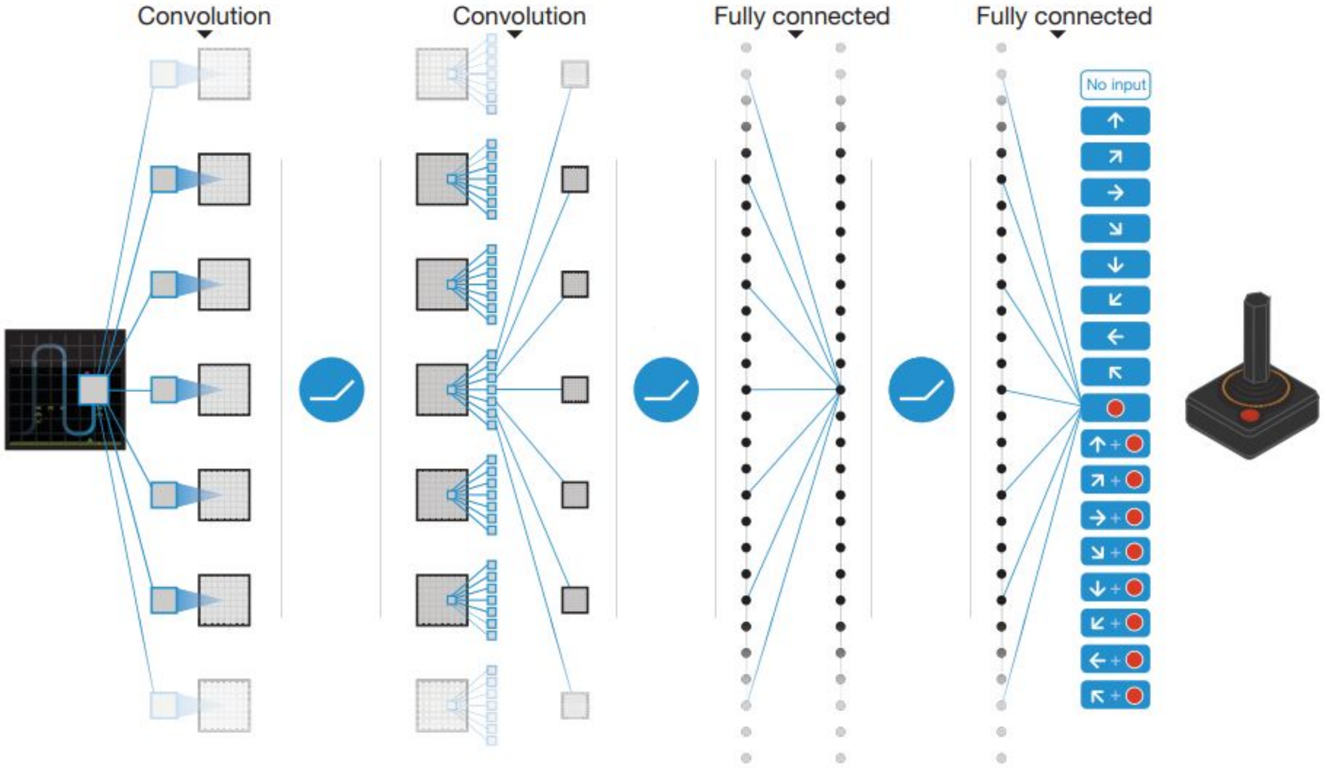
Neural Network learns to predict Q-function



$$\mathcal{L} = \mathbb{E}\left[\left\| \overbrace{\left(r + \gamma \max_{a'} Q(s',a')\right)}^{\text{Target}} - \overbrace{Q(s,a)}^{\text{Predicted}} \right\|^2\right]$$

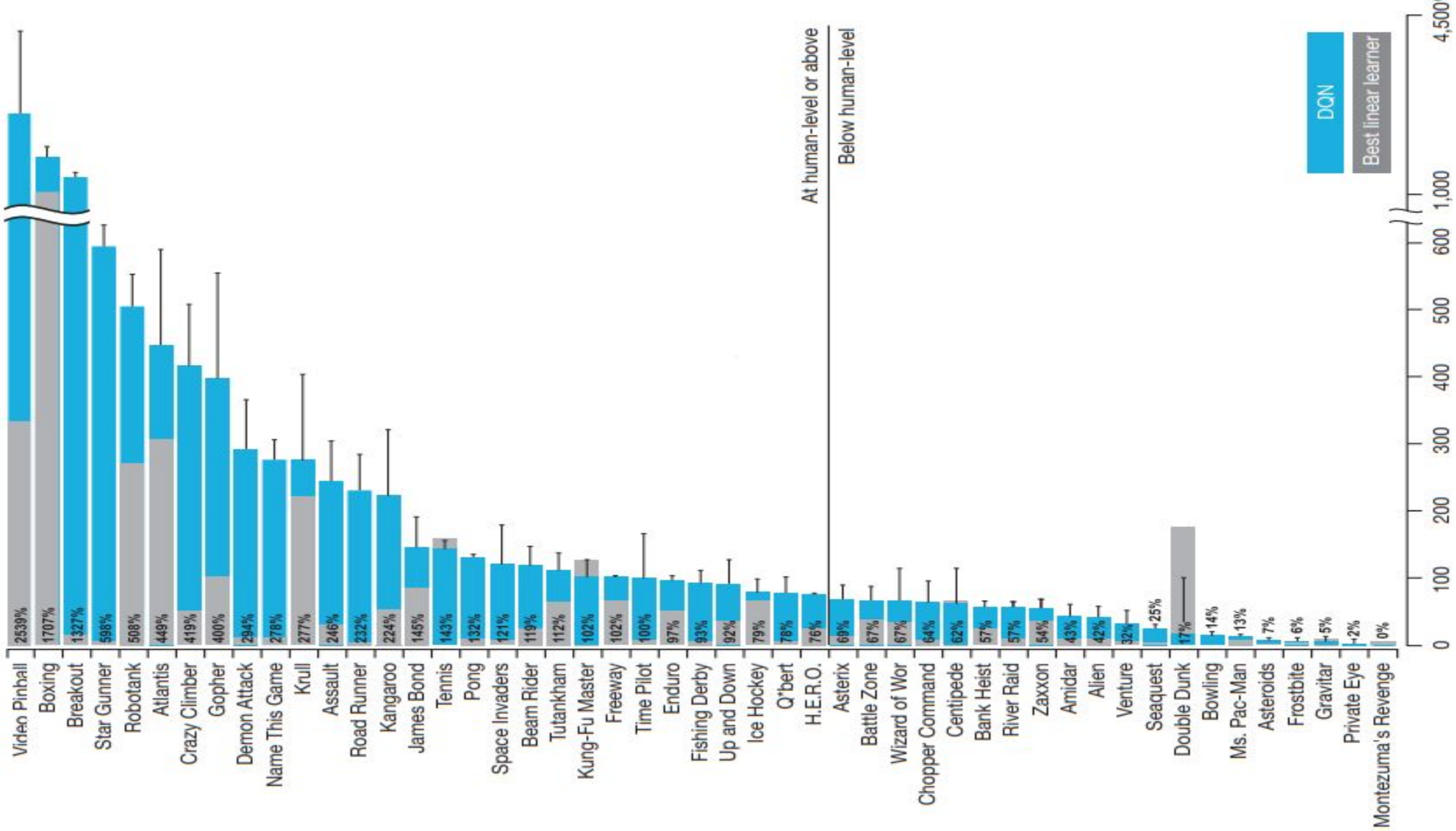
Source: [1]

DQN for Atari Games



Source: [1]

DQN for Atari Games



Source: [1]

Downsides of Q-learning

Can only handle limited complexity

- Model scenarios with large or continuous action spaces cannot be handled

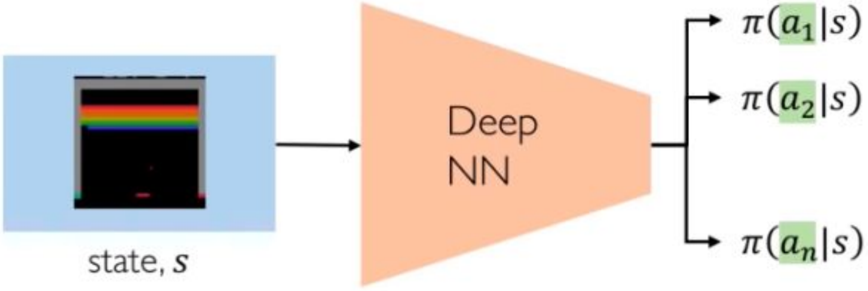
Limited flexibility

- Cannot learn stochastic policies since policy is deterministically computed from the Q function

Policy Gradient (PG)

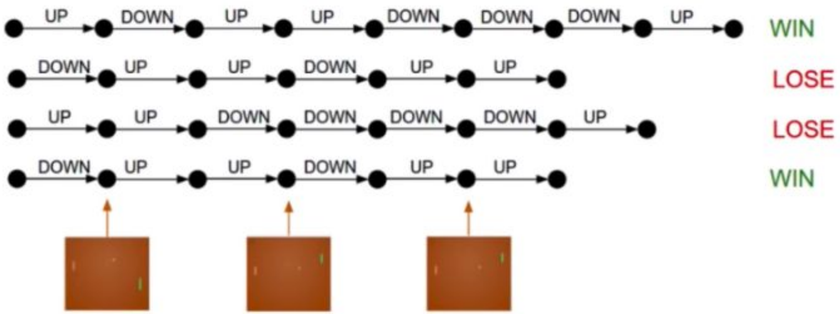
DQN: Approximate Q and infer optimal policy

Policy Gradient: Directly learn the policy



Source: [1]

Training Policy Gradient



1. Run a policy for a while
2. Increase probability of actions that lead to high rewards
3. Decrease probability of actions that lead to low rewards

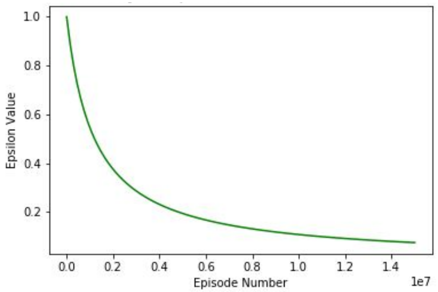
```
function REINFORCE
  Initialize θ
  for episode ~ π_θ
    {s_i, a_i, r_i}_{i=1 to T-1} ← episode
    for t=1 to T-1
      ∇ ← ∇_θ log π_θ(a_t|s_t) R_t
      θ ← θ + α∇
  return θ
```

Source: [1]

Exploration

ε - Greedy Exploration

ε - Value



p(x) = ε

Explore

Choose random action

p(x) = 1-ε

Exploit

Pick top action from policy

Source: [7]

Overcoming Sparse Rewards

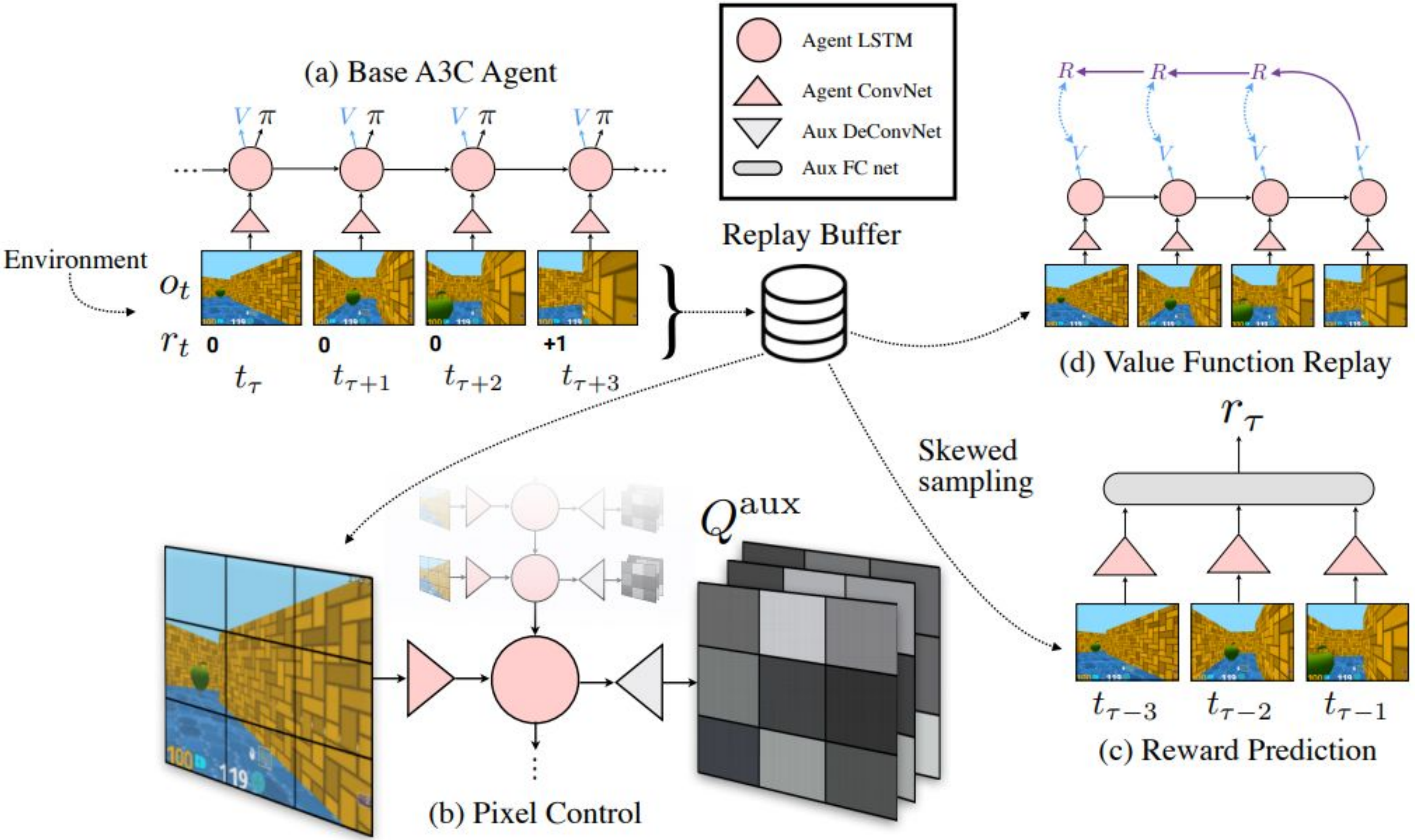
Reward Shaping

- Design other reward mechanisms that produce more frequent positive rewards
- Easier for the agent to converge to a solution

But:

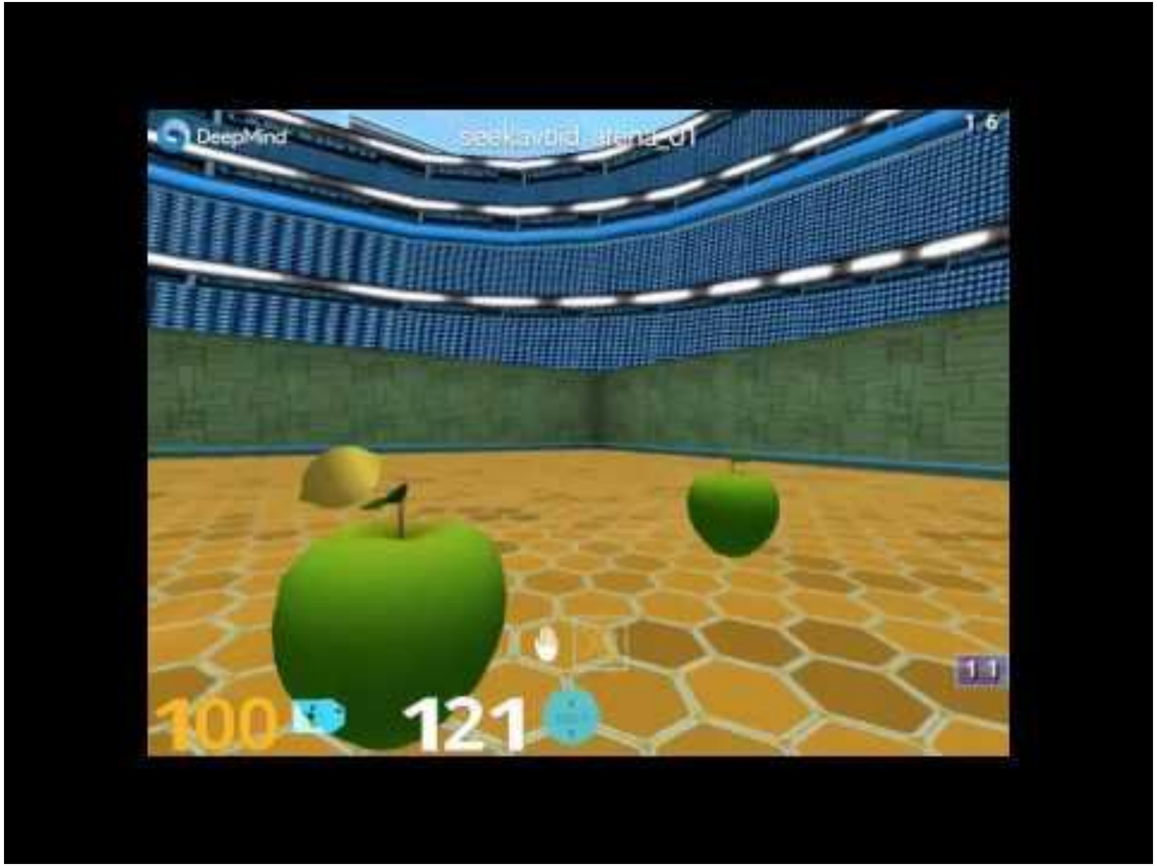
- Custom process that needs to be redone for each task
- Alignment Problem: When you shape reward function, it might happen that the agent finds a surprising solution to get a high reward, but not at all do what you want it to do
- Limits the solution to the way how humans solved the task

Auxiliary Tasks



Source: [6]

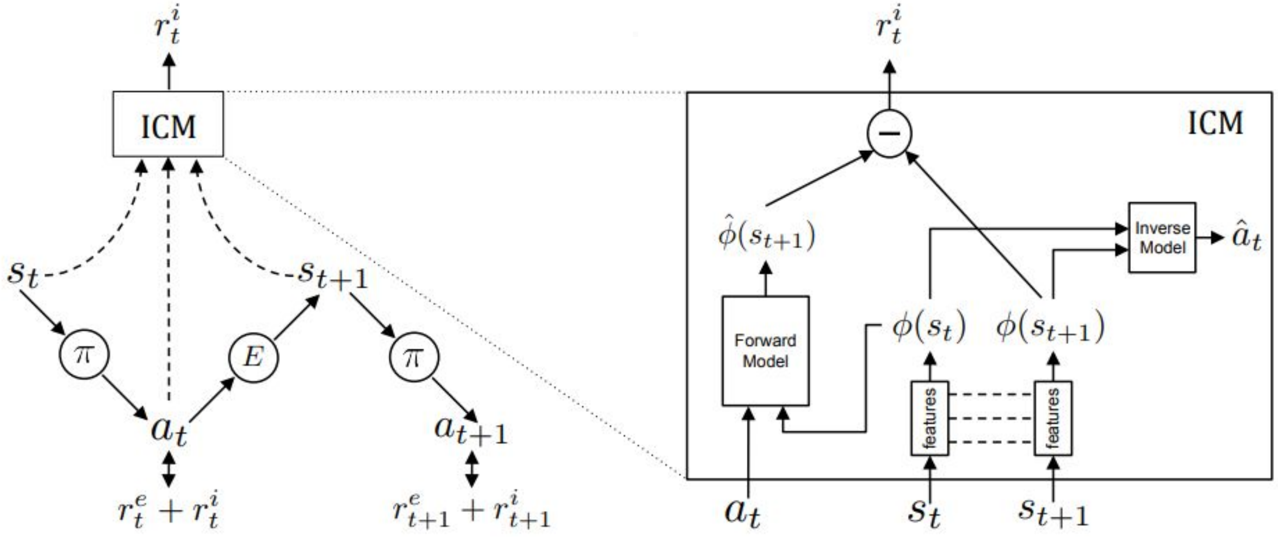
Auxiliary Tasks



Source: [6]

Curiosity-driven Exploration

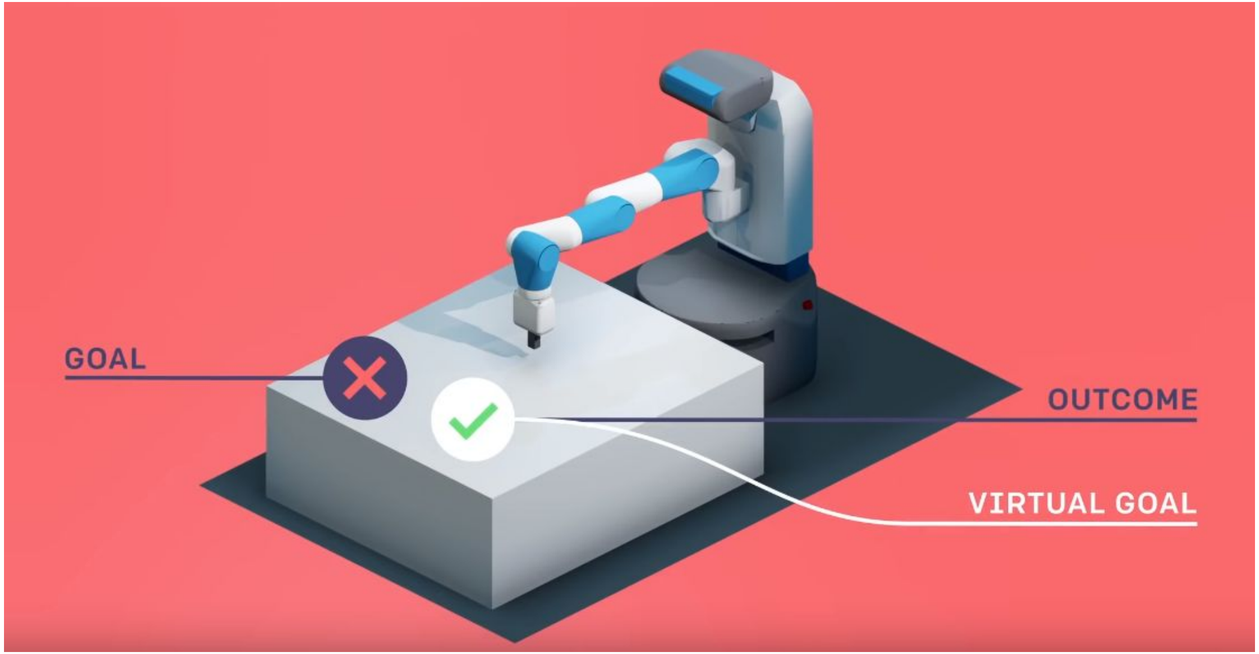
- Extract features from the current frame into a latent space
- Forward network tries to predict same latent representation for the next frame
  - If agent has seen environment before, the prediction will be accurate
  - If agent has never seen situation before, the prediction will be poor
- Agent is rewarded for exploring unseen parts of the environment



Source: [8]

Hindsight Experience Replay (HER)

- If an episode is not successful, the agent does not get any reward
- Instead of giving no reward, we create a virtual goal and pretend, it is what we wanted



Sources: [7, 9]

Practical Reinforcement Learning

OpenAI gym

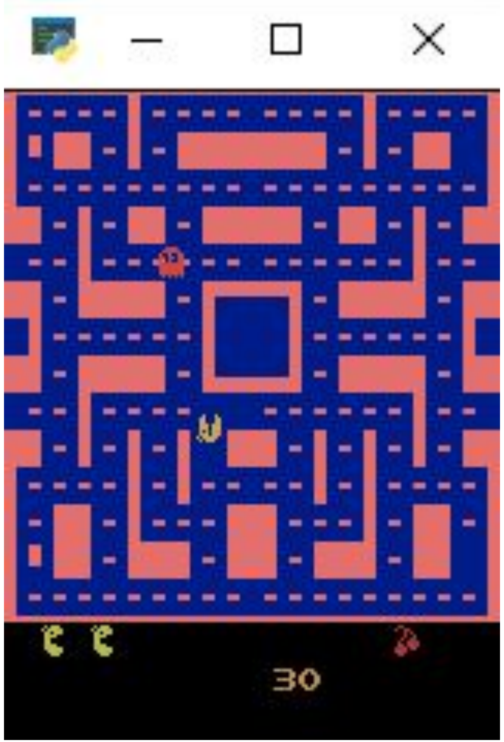
https://gym.openai.com/

Gym

Gym is a toolkit for developing and comparing reinforcement learning algorithms. It supports teaching agents everything from walking to playing games like Pong or Pinball.

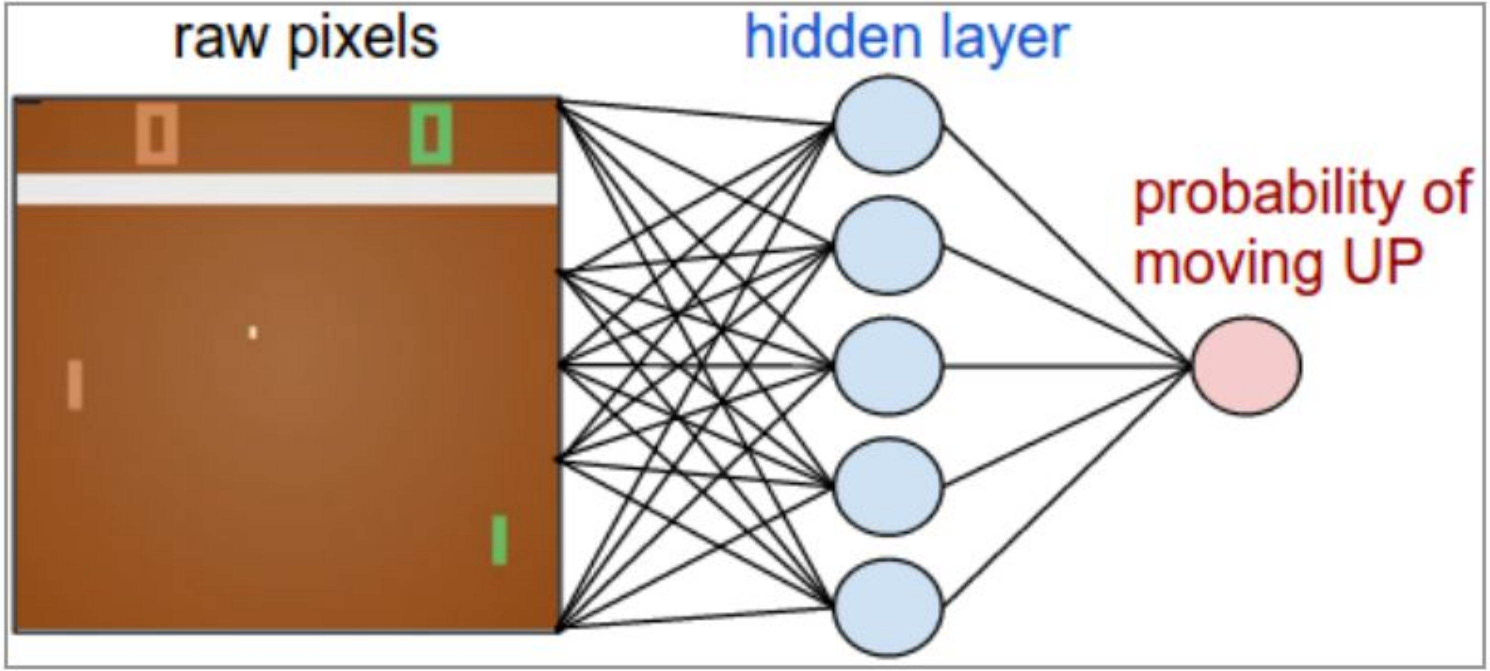
Playing Pacman

```python
import gym


env = gym.make("MsPacman-v0")
state = env.reset()
done = False
total_rewards = 0


while not done:
    env.render()
    action = env.action_space.sample()
    state, reward, done, info = env.step(action=action)
    total_rewards += reward
```



Pong from Pixels

[Karpathys example on Google Colab]



Source: [13]

A Reference Framework - rlpyt

https://github.com/astooke/rlpyt

rlpyt

Deep Reinforcement Learning in PyTorch

Modular, optimized implementations of common deep RL algorithms in PyTorch, with unified infrastructure supporting all three major families of model-free algorithms: policy gradient, deep-q learning, and q-function policy gradient. Intended to be a high-throughput code-base for small- to medium-scale research (large-scale meaning like OpenAI Dota with 100's GPUs). Key capabilities/features include:

BAIR

BERKELEY ARTIFICIAL INTELLIGENCE RESEARCH

Implements common algorithms:

- Policy Gradient
- Deep-Q Learning
- Q-Function Policy Gradient

Policy Gradient Algorithms

https://lilianweng.github.io/lil-log/2018/04/08/policy-gradient-algorithms.html

- Policy Gradient Algorithms
  - REINFORCE
  - Actor-Critic
  - Off-Policy Policy Gradient
  - A3C
  - A2C
  - DPG
  - DDPG
  - D4PG
  - MADDPG
  - TRPO
  - PPO
  - PPG
  - ACER
  - ACTKR
  - SAC
  - SAC with Automatically Adjusted Temperature
  - TD3
  - SVPG
  - IMPALA

Spinning up Reinforcement Learning

https://spinningup.openai.com/en/latest/

AlphaGo (2016)

1. Initial training: human data

2. Self-play and reinforcement learning

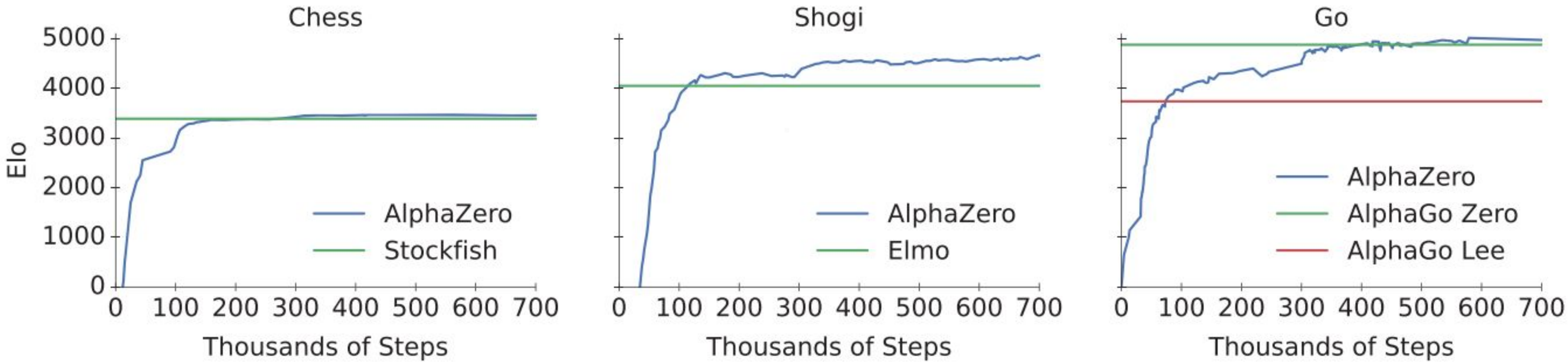
3. “Intuition” about board state

AlphaGo beat top human Go player in 2016.

Source: [1]

AlphaZero (2018)

- Framework to learn board games without human pre-learning
- Entirely learned through self-play and the game rules

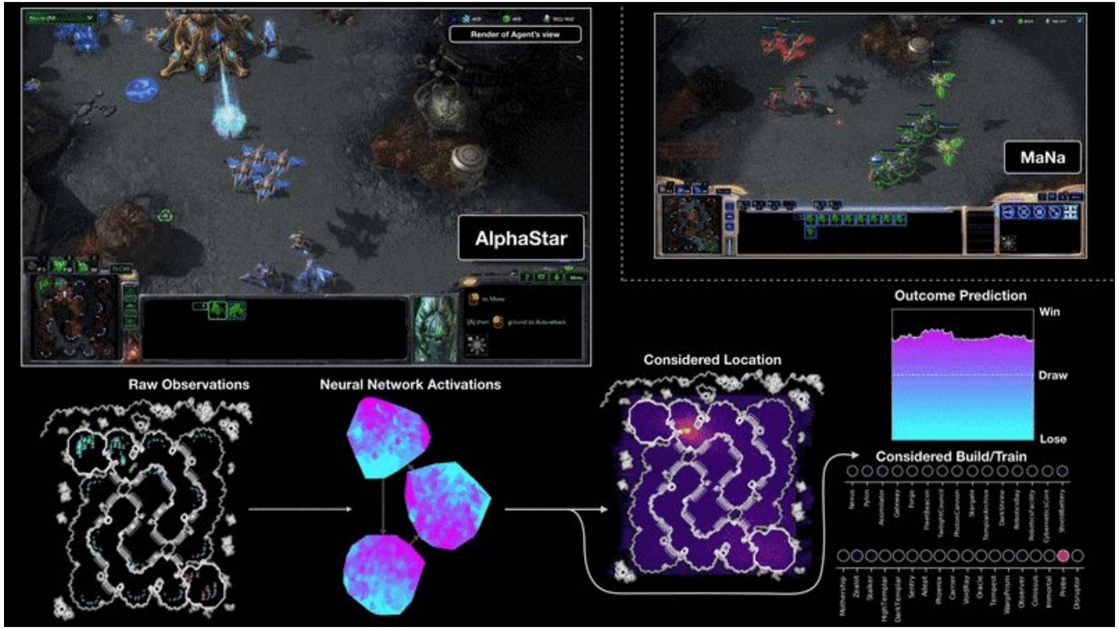


Demis Hassabis: “Play style feels ‘alien’: It sometimes wins by offering counterintuitive sacrifices, like offering up a queen and bishop to exploit a positional advantage. It’s like chess from another dimension”

Source: [12]

AlphaStar (2019)

- Deep Neural Network beat Professional Star Craft II players
- Internally has a transformer network, LSTM, Pointer Network and Centralised Value Baseline.



Source: [10]

Summary

- Reinforcement Learning is a class of problems that operates on state-action pairs
- An agent perceives a state an performs action in an environment
- Common functions
  - Q-function measures expected rewards from taking action a in state s
  - V-function measure expected rewards when starting in state s
  - Policy-function estimates the best action a to take when in state s
- Value-learning vs. Policy-learning
- Many challenges:
  - Partially observable environment
  - Sparse/Late rewards
  - Credit assignment problem
- RL can be seen as supervised learning, but on a continuously changing dataset (the episodes).

Literature

1. Amini, Soleimany, MIT 6.S191 - Introduction to Deep Learning
2. Stooke et al. rlpyt: A Research Code Base for Deep Reinforcement Learning in PyTorch, 2019.
3. Luketina et al. A Survey of Reinforcement Learning Informed by Natural Language, 2019.
4. Li et al. Stanford CS231n: Reinforcement Learning in Visual Computing, 2017.
5. Brunskill et al. Stanford CS234: Reinforcement Learning, 2019.
6. Jaderberg et al. Reinforcement Learning with Unsupervised Auxiliary Tasks, 2016.
7. Steenbrugge. Overcoming spare rewards in Deep RL, 2018.
8. Pathak et al. Curiosity-driven Exploration by Self-supervised predicition, 2017.
9. Andrychowicz et al. Hindsight Experience Replay, 2017.
10. The AlphaStar Team. AlphaStar: Mastering the Real-Time Strategy Game StarCraft II, 2019.
11. Mnih et al. Human-level control through deep reinforcement learning, 2015.
12. Silver et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play, 2018.
13. Wang et al. Dueling Network Architectures for Deep Reinforcement Learning, 2016.
14. Karpathy, Deep Reinforcement learning: Pong from Pixels, 2016.

Icon credits

Free icons from Flaticon:

- https://www.flaticon.com/free-icon/asking_900415
- https://www.flaticon.com/free-icon/dictionary_917168
- https://www.flaticon.com/free-icon/apple_590764
- https://www.flaticon.com/free-icon/apple_1155289
- https://www.flaticon.com/free-icon/apple_1135536
- https://www.flaticon.com/free-icon/delay_1995982
- https://www.flaticon.com/free-icon/direction_2345147
- https://www.flaticon.com/free-icon/planning_762620
- https://www.flaticon.com/free-icon/scale_462315
- https://www.flaticon.com/free-icon/employee_1256663
- https://www.flaticon.com/free-icon/worlwide_1256630
- https://www.flaticon.com/free-icon/hammer_588409
- https://www.flaticon.com/free-icon/idea_427735
- https://www.flaticon.com/free-icon/inspection_1814540
- https://www.flaticon.com/free-icon/magic-ball_867840
- https://www.flaticon.com/free-icon/learning_2126425
- https://www.flaticon.com/free-icon/money_1160083
- https://www.flaticon.com/free-icon/compass_2345124
- https://www.flaticon.com/free-icon/reward_1426739