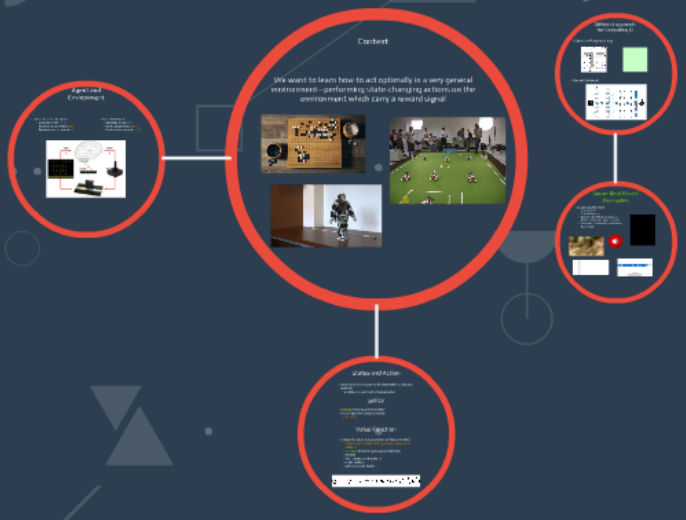
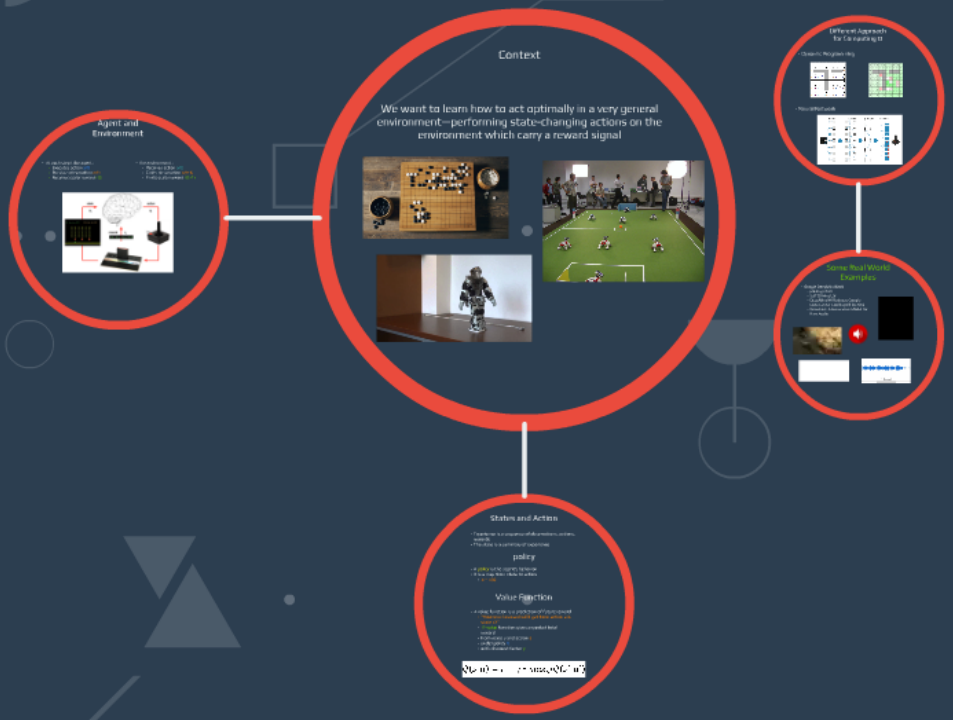
Deep Q learning

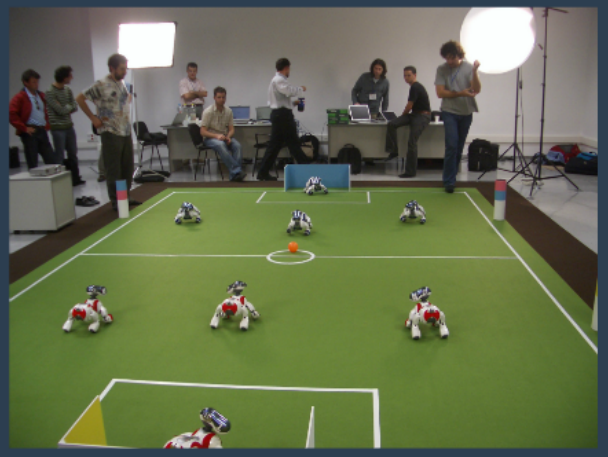


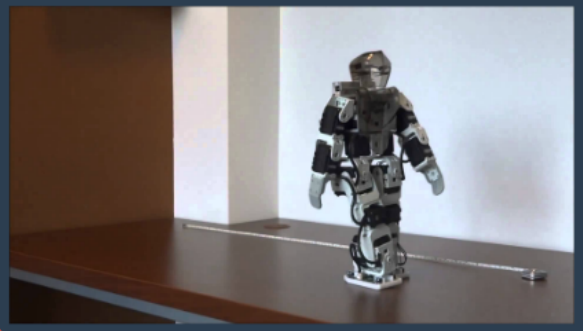
Deep Q learning



Context

We want to learn how to act optimally in a very general environment—performing state-changing actions on the environment which carry a reward signal

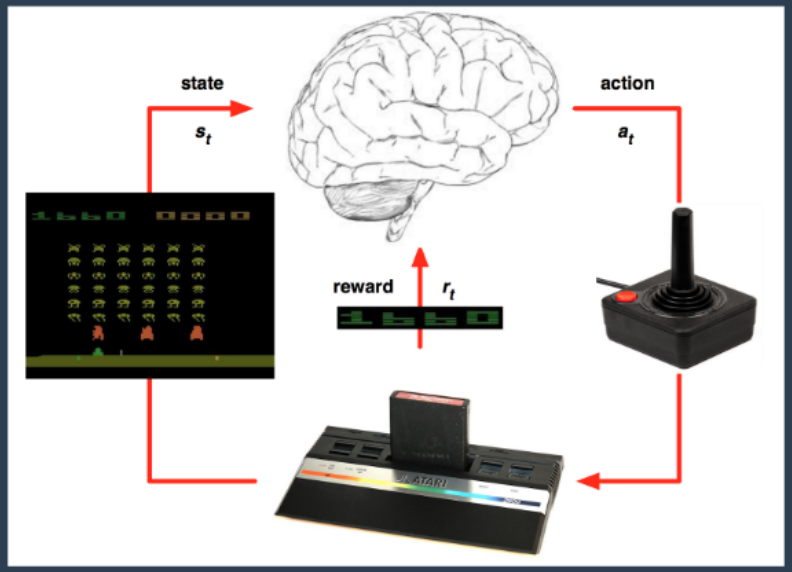




Agent and Environment

- At each step t the agent :
  - Executes action a(t)
  - Receives observation o(t)
  - Receives scalar reward r(t)

- The environment :
  - Receives action a(t)
  - Emits observation o(t+1)
  - Emits scalar reward r(t+1)



States and Action

- Experience is a sequence of observations, actions, rewards
- The state is a summary of experience

policy

- A policy is the agent's behavior
- It is a map from state to action
  - a = π(s)

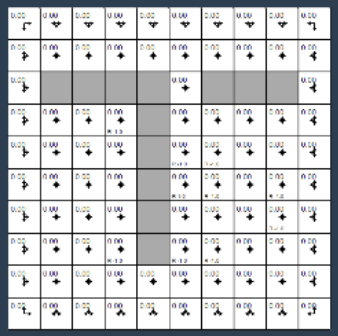
Value Function

- A value function is a prediction of future reward
  - "How much reward will I get from action a in state s?"
  - Q-value function gives expected total reward
  - from state s and action a
  - under policy π
  - with discount factor γ

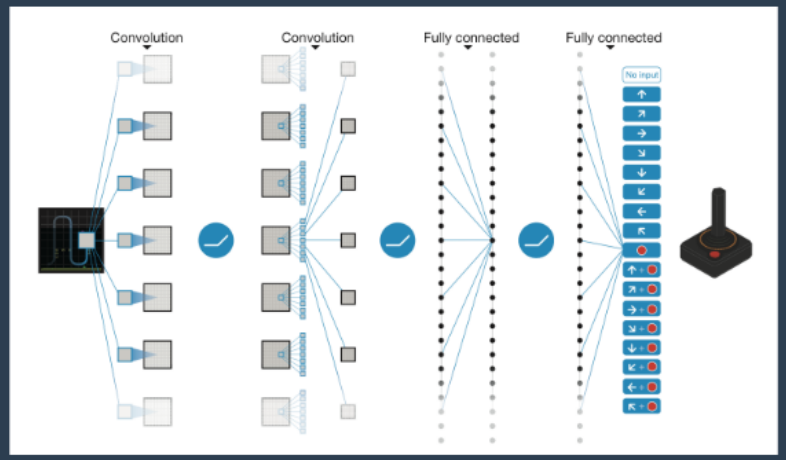
$$Q(s,a) = r + \gamma * max_{a'} Q(s',a')$$

Different Approach for Computing Q

- Dynamic Programming

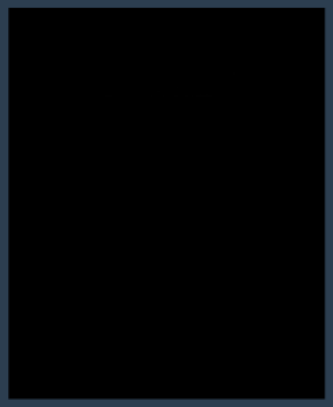


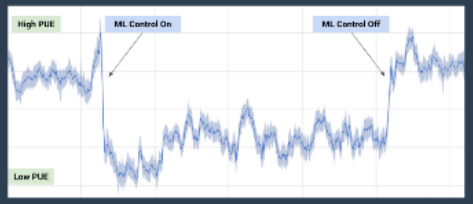
- Neural Network

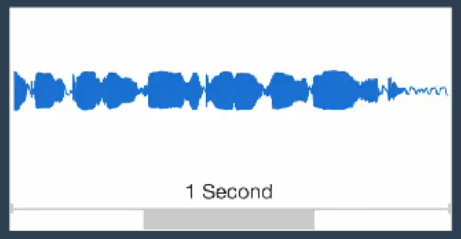


Some Real World Examples

- Google DeepMind DQN
  - playing Atari
  - Self Driving Car
  - DeepMind AI Reduces Google Data Center Cooling Bill by 40%
  - WaveNet: A Generative Model for Raw Audio







Deep Q learning