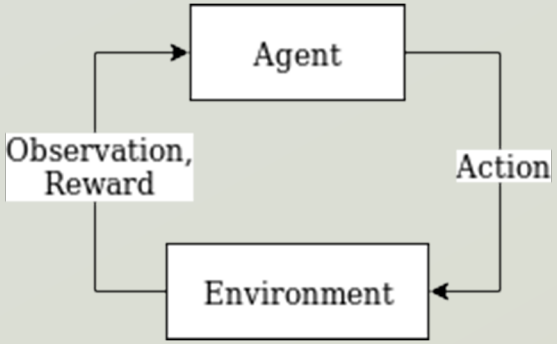
Learn to play Atari's Breakout with Deep Reinforcement Learning

By Jonas Heitz

zhaw School of Engineering
InIT Institute of Applied Information Technology

Environment

- Gym provided by OpenAI
  - Is a library which allows to simulate many reinforcement learning environments (incl. Atari games)
  - Provides a simple interface for the agent
    (e.g. observation, reward, termination_flag, _ = env.step(action))



Markov Decision Process(MDP)

- MDP< S , A , P , R , γ >
  - S = state space = continuous
  - A = action space = { start, left, right, stay }
  - P = state transition probability matrix = 50% for all
  - R = reward = {0,1}
  - γ = reward discounting factor = 1

Preprocessing

- The observations are 210×160 pixel images with 128 colors
  - Processing big colorized images is computationally expensive
  - So we convert the input to monochrome images with 105x80 pixel

- To capture and learn the motion of moving objects we bundle 4 images together

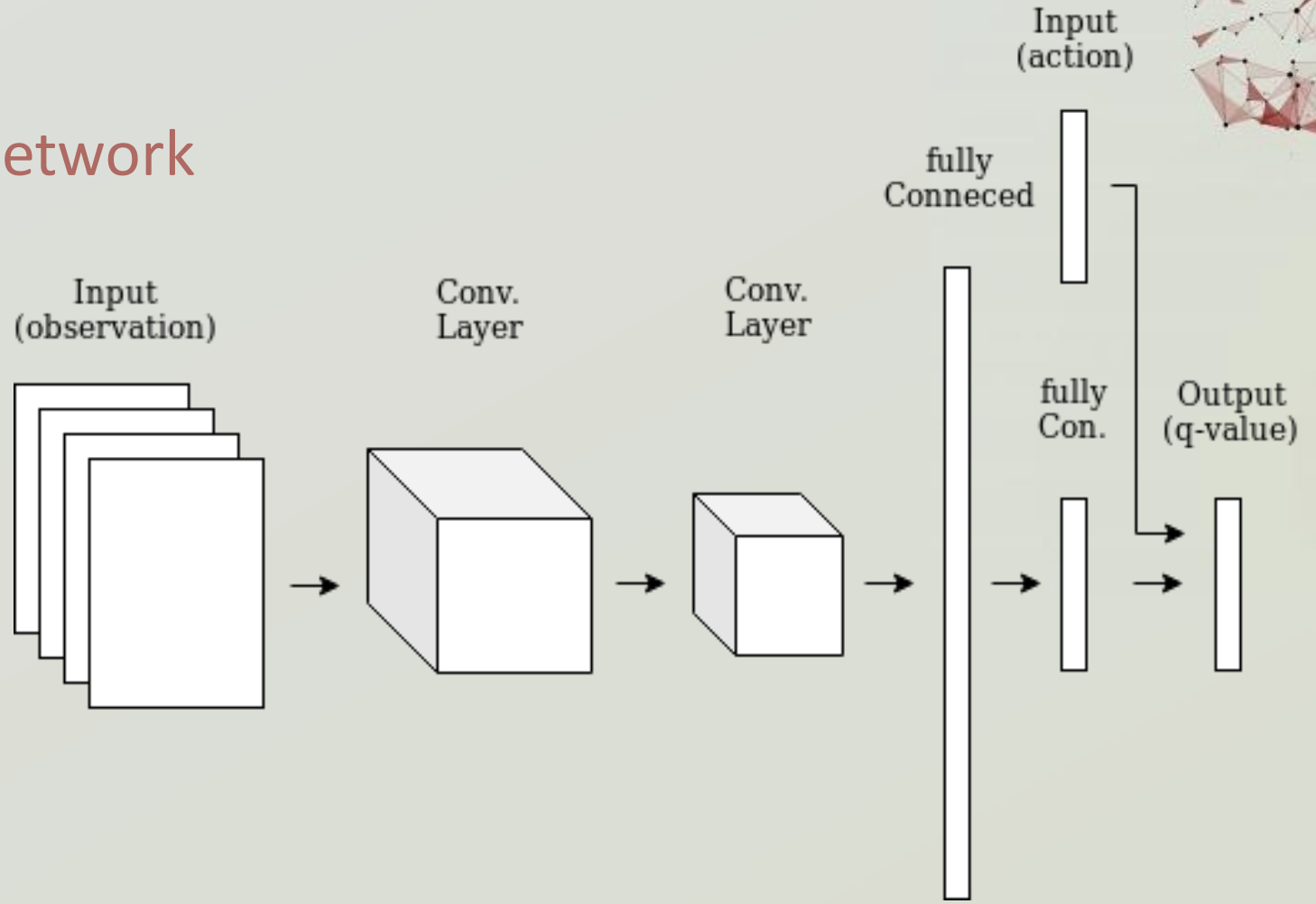
Q-Learning

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

- In some sense, we need to create a model which is trying to predict the max of its own output

Model

- Deep-Q-Network



Tricks

zh aw
School of Engineering
InIT Institute of Applied Information Technology

(3) Phases of Learning

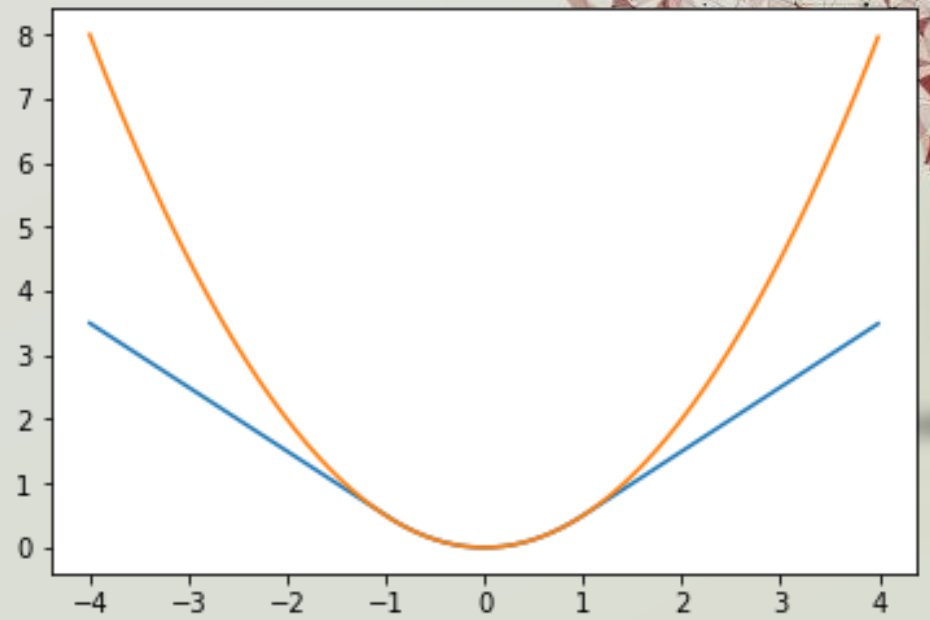
- Observation
  - Random actions only (ε = 1)
  - 50’000 iterations
- Exploration
  - Randomization-ε annealing (ε = 1 – 0.01)
  - 1000’000 iterations
- Training
  - Fixed randomization-ε (ε = 0.01)
  - 100’000 iterations

Experience Replay

- Every (state, action, new state, reward)-pair gets stored in finite-size memory
  - Ideal 1 million elements
  - Attention: requires a lot of memory
- The network gets trained on 32 independent entries every iteration.
- This is the key method to prevents the network from diverging

Huber Loss

- Definition
  - Mean squared error for small values
  - Mean absolute error for large values



Huber loss (in blue) and squared error (in orange)

- Huber Loss leads the network to less radical changes

Target Network

- $Q(s, a) = r + \gamma \max_{a'}(Q(s', a'))$ is recursive
  - While training Q(s, a) **and** Q(s', a') get changed
  - Getting close to a target which moves as well is hard (chasing its own tail)

- Solution:
  - Using a cloned network for Q(s', a') which gets a weight updated every 10'000 iteration

Code

github.com/heitzjon/drl_atari_breakout