Applied Deep Learning
Deep Reinforcement Learning

Mark Schutera

Karlsruhe Institute of Technology | ZF Friedrichshafen AG | Hochschule Ravensburg-Weingarten

Course overview

1. Deep Learning Foundations

3. Transfer Learning and Object Detection

5. Segmentation Networks

8. Deep Reinforcement Learning

10. Generative Adversarial Networks

12. Recurrent Neural Networks

Course overview

Project proposal
One page on introduction, methods, dataset
Deadline 3. Lecture

Intermediate presentation
Ten minutes on achievements, problems, next steps
Due 7. Lecture

Final presentation
Screencast (Slides and Audio)
Due 13. Lecture (20.01.2020)

Final documentation
Documentation and code on github
Deadline 13. Lecture (20.01.2020)

Course features

Sli.do

Every question matters.

Get the app.

Ask questions (with slide number) or vote on other students' questions during the lecture.

And give direct feedback.

#TOBEDETERMINED

Questions will be covered immediately or in the next lecture in more depth.

Github

Find slides, tutorials, flashcards and references on Github.

https://github.com/schutera/DeepLearningLecture_Schutera

You found typos, additional material such as links, algorithms, papers, literature or want to contribute to the slides and lecture notes..

..Feel free to contribute, e-mail me.

Course features

Sli.do

Every question matters.

Get the app.

Ask questions (with slide number) or vote on other students' questions during the lecture.

#TOBEDETERMINED

Questions will be covered directly or in the next lecture in more depth.

Github

Typos, additional material such as links, algorithms, paper, literature, lecture notes..

..Feel free to contribute.

Grade Bonus .3

Prepare flashcards based on Ian Goodfellow's Deep Learning Book

- Commit to flashcard set by emailing me, first come first serve
- Must be comprehensive

This lecture in one slide

Introduction to reinforcement learning

About reinforcement learning

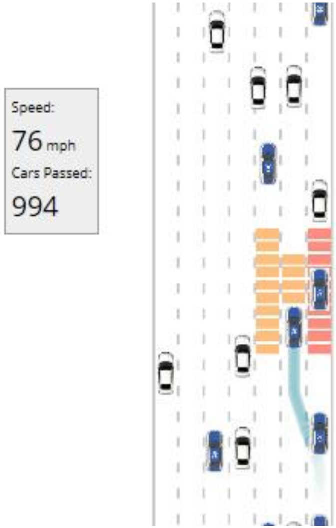
Reinforcement learning a problem statement

Reinforcement learning framework

Reinforcement Learning with neural networks

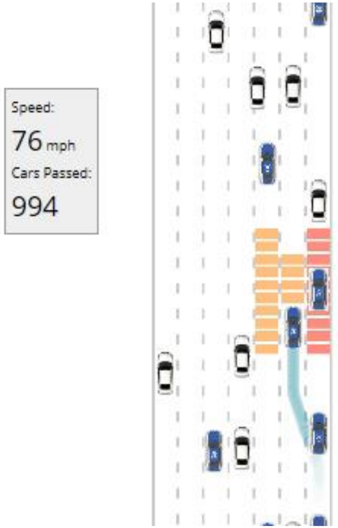
Introduction – Reinforcement learning

Reinforcement learning (RL) is an area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward.



Introduction – Reinforcement learning

Learn to make good sequences of decisions



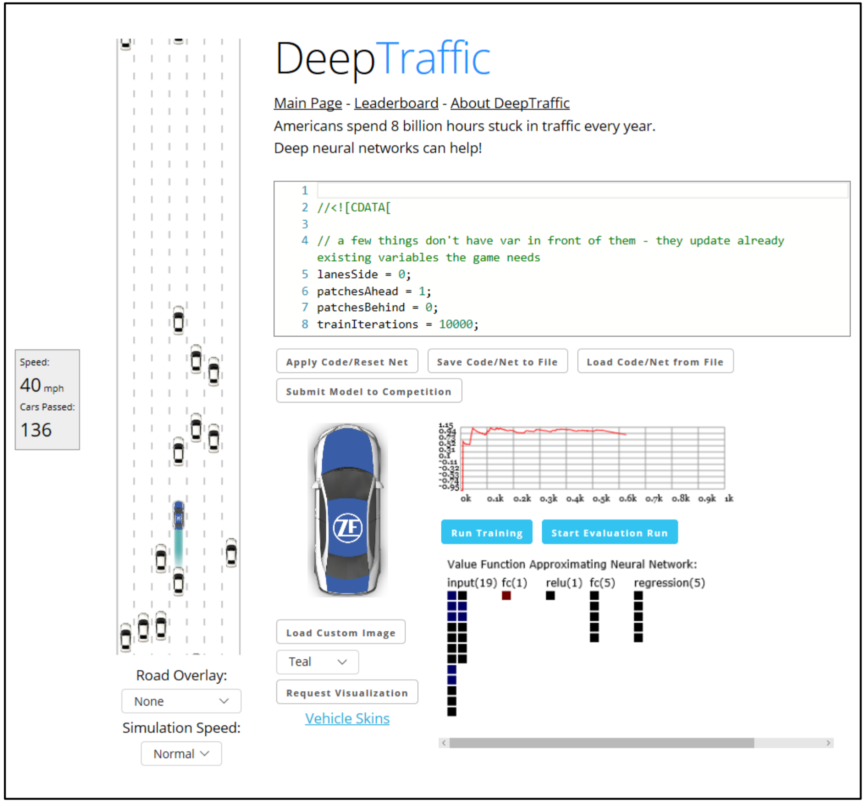
Introduction – Reinforcement learning

What makes reinforcement learning different from other machine learning paradigms

- Feedback / **supervision** directly comes from the **environment**
- **Model interacts** with the **environment**
  or at least ist own state in the environment

Introduction – Reinforcement learning

- **Dense highway traffic**
  - **80** mph max. Speed
  - **7** Lanes
  - **20** Vehicles
  - **1 - 11** Trainable Vehicles



https://selfdrivingcars.mit.edu/deeptraffic/

- Deep Reinforcement Competition

Deep Reinforcement Learning with the Deep Q-Network

Deep Reinforcement Learning
Decision Making

- **Environment**
Either real-world or simulation of some sort

- Action
- State
- Reward

Deep Reinforcement Learning with the Deep Q-Network

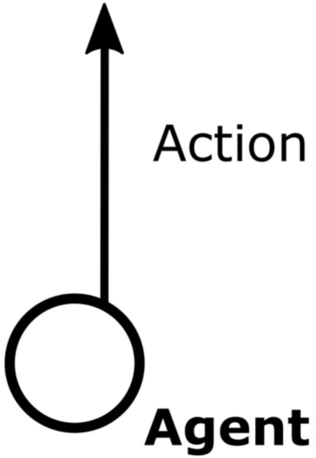
Deep Reinforcement Learning
Decision Making

- Environment

- **Action a**

Agent‘s actions affect
the subsequent data it receives

- State
- Reward



Deep Reinforcement Learning with the Deep Q-Network

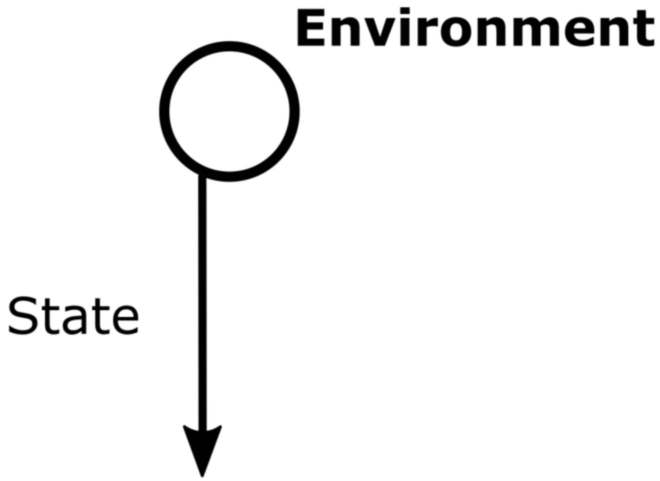
Deep Reinforcement Learning
Decision Making

- Environment
- Action

- **State s**

Feedback is delayed, not instantaneous

- Reward



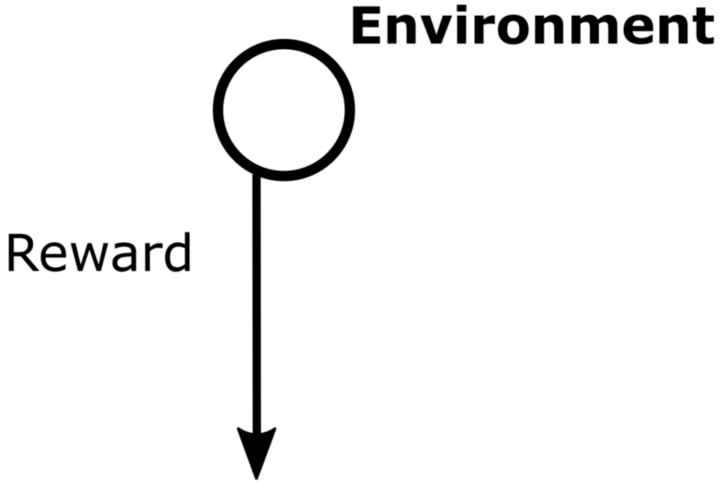
Deep Reinforcement Learning with the Deep Q-Network

Deep Reinforcement Learning
Decision Making

- Environment
- Action
- State

- **Reward r**

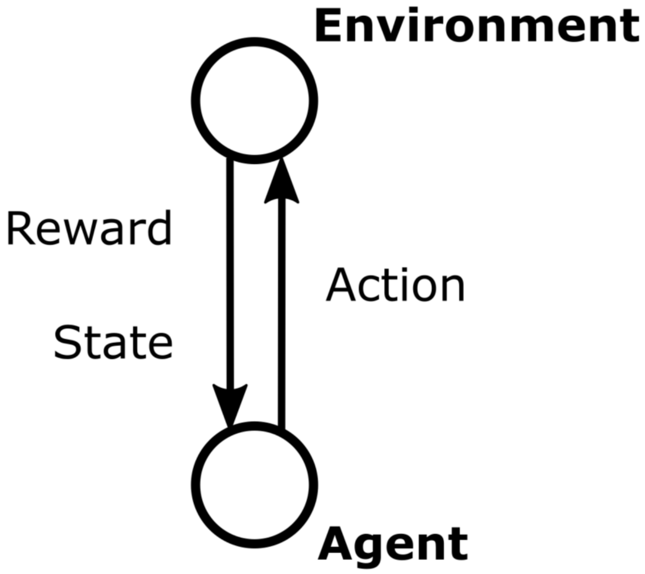
There is no supervisor, only a reward signal



Deep Reinforcement Learning

Time really matters (sequential signal stream)

- Environment
- Action
- State
- Reward



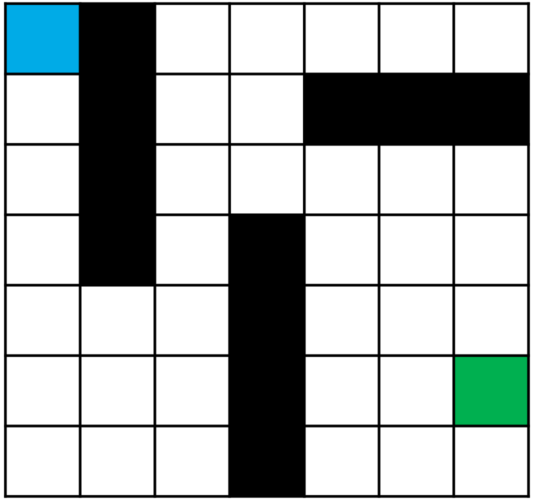
Maze Example

Conventional Reinforcement Learning

- **Environment**

  Maze, Start and End Position

- Action
- State
- Reward

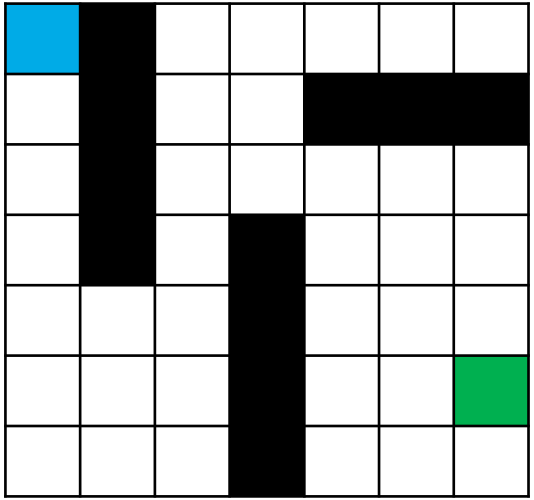


Maze Example

Conventional Reinforcement Learning

- Environment
- **Action**

  Move left, right, up, down
- State
- Reward



Maze Example

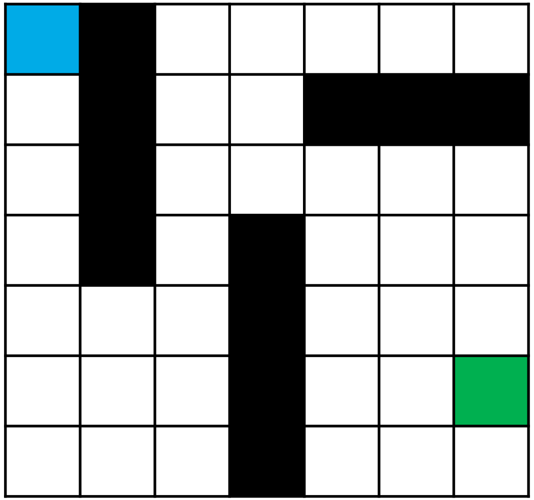
Conventional Reinforcement Learning

- Environment
- Action

- **State**

  Position in one of the maze cells

- Reward



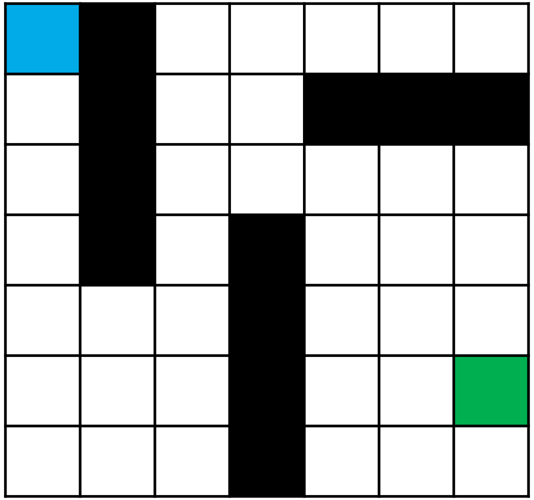
Maze Example

Conventional Reinforcement Learning

- Environment
- Action
- State

- **Reward**

-1 for valid move, +1 for transition into the end position



Maze Example

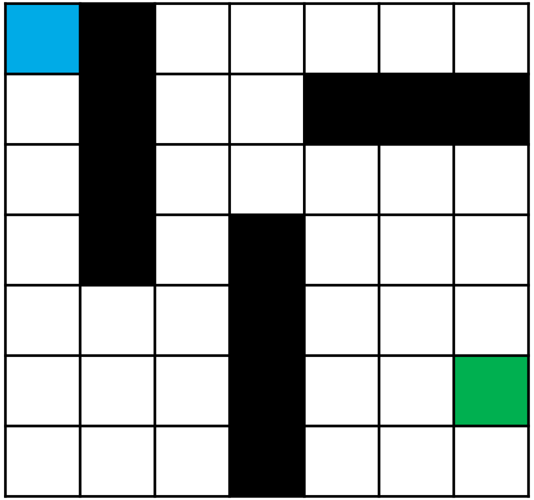
Conventional Reinforcement Learning

- Environment
- Action
- State

- **Reward**

-1 for valid move, +1 for transition into the end position

What happens for a reward of 0 for a valid move?



This lecture in one slide

Introduction to reinforcement learning

Reinforcement Learning with neural networks

Q-Learning

Fixed target network

Experience replay

Reward clipping

Frame skipping

Reward function

Self-Play

Deep dive Deep-Q-Learning

Datasets and benchmarking and current success stories

Reinforcement Learning and why it does not work yet

Q-Learning

Value-based algorithm

Lookup table of values with one entry for every state-action pair

$$Q(s, a)$$

Deep Q-Learning

Representation Learning

Lookup table of values with one entry for every state-action pair

$$Q(s, a)$$

High dimensional state-action spaces, make a mere Q-value function inapplicable

In this case a parametrized value function (neural network) is needed

$$Q(s, a; \theta_i)$$

Deep Q-Learning

Representation Learning

Idea follows a trial-and-error strategy, exploring the states and iteratively updating the state-action values

$$Q(s, a)$$

All actions ought to be **repeatedly sampled** in **all states** to ensure sufficient exploration.

Deep Q-Learning

Representation Learning

Idea follows a trial-and-error strategy, exploring the states and iteratively updating the state-action values

$$Q(s, a)$$

How does our final / optimal policy look like?

Given a state calculate the expected maximal reward of an action, following our policy.

$$Q^*(s,a) = \max_{\pi} \mathbb{E}\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots \middle| s_t = s,\, a_t = a,\, \pi\right]$$

Deep Q-Learning

Loss function

$$L_i(\theta_i) = \mathbb{E}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i)\right)^2\right]$$

Our network is trained the same way we are used to – by backpropagating a loss and adjusting our weights.

Deep Q-Learning
Loss function

$$L_i(\theta_i) = \mathbb{E}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i)\right)^2\right]$$

Our network is trained the same way we are used to – by backpropagating a loss and adjusting our weights.

The central idea here is not to find the best action, but to learn an accurate estimation of the Q-value function.

Deep Q-Learning
Loss function

$$L_i(\theta_i) = \mathbb{E}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i)\right)^2\right]$$

Our network is trained the same way we are used to – by backpropagating a loss and adjusting our weights.

The central idea here is not to find the best action, but to learn an accurate estimation of the Q-value function.

The loss is thus the difference between the true **reward** from the environment
And the **estimated reward** of our model

Deep Q-Learning

Fixed Target Network

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s') \sim \mathrm{U}(D)} \left[\left(r + \gamma \max_{a'} Q(s',a';\theta_i^-) - Q(s,a;\theta_i) \right)^2 \right]$$

$$\theta_i^-$$

The parameters for the target network are only updated every C iterations

Deep Q-Learning

Fixed Target Network

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim \mathrm{U}(D)}\left[\left(r + \gamma \max_{a'} Q(s',a';\theta_i^-) - Q(s,a;\theta_i)\right)^2\right]$$

The parameters for the target network are only updated every C iterations

This reduces the risk of oscillations or divergence and increases stability of the training process

Deep Q-Learning

Experience Replay

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s') \sim \mathrm{U}(D)} \left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i) \right)^2 \right]$$

State action pairs together with the expected reward and resulting state transition are stored for N time steps.

Deep Q-Learning

Experience Replay

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim \mathrm{U}(D)}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i)\right)^2\right]$$

State action pairs together with the expected reward and resulting state transition are stored for N time steps.

During training we then draw from the experience replay memory

Deep Q-Learning

Experience Replay

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim \mathrm{U}(D)}\left[\left(r + \gamma \max_{a'} Q(s',a';\theta_i^-) - Q(s,a;\theta_i)\right)^2\right]$$

State action pairs together with the expected reward and resulting state transition are stored for N time steps.

During training we then draw from the experience replay memory

Actions are drawn randomly with probability Epsilon ϵ

ϵ-greedy policy

And trained within a mini-batch.

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim \mathrm{U}(D)}\left[\left(r + \gamma \max_{a'} Q(s',a';\theta_i^-) - Q(s,a;\theta_i)\right)^2\right]$$

Deep Q-Learning

ϵ-greedy policy

Exploration

ϵ = 1

Meaning actions are completely randomly selected resulting in maximum exploration within our action-state space

Exploitation

Deep Q-Learning

ϵ-greedy policy

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim U(D)}\left[\left(r + \gamma \max_{a'} Q(s',a';\theta_i^-) - Q(s,a;\theta_i)\right)^2\right]$$

Exploration

Exploitation

ϵ = 0

Meaning no random actions what so ever.

Actions are selected with respect to the maximum expected reward

Deep Q-Learning

ϵ-greedy policy

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s') \sim \mathrm{U}(D)}\left[\left(r + \gamma \max_{a'} Q(s',a';\theta_i^-) - Q(s,a;\theta_i)\right)^2\right]$$

Exploration

Exploitation

During training which ϵ-value would you want to use?

Deep Q-Learning

ϵ-greedy policy

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim \mathrm{U}(D)}\left[\left(r + \gamma \max_{a'} Q(s',a';\theta_i^-) - Q(s,a;\theta_i)\right)^2\right]$$

Exploration

Exploitation

During training which ϵ-value would you want to use?

ϵ = 1 for **full exploration first** (think baby)

This at some point might be inefficient since you do not profit from already learned action-state pairs.

Decrease ϵ over time (think grown-up)

Deep Q-Learning

ϵ-greedy policy

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim \mathrm{U}(D)}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i)\right)^2\right]$$

Exploration

Exploitation

During inference which ϵ-value would you want to use?

Deep Q-Learning

ϵ-greedy policy

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim \mathrm{U}(D)}\left[\left(r + \gamma \max_{a'} Q(s',a';\theta_i^-) - Q(s,a;\theta_i)\right)^2\right]$$

Exploration

Exploitation

During inference which ϵ-value would you want to use?

ϵ = 0 for **full exploitation** (think adult who has his routines and does not really reflect what she is doing)

This at some point might get you stuck in a deadlock

Thus **allow for some randomness** in your choice of action ϵ = 0.05 (think adult who tries to drink water instead of soda)

Deep Q-Learning

Reward Clipping

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s') \sim \mathrm{U}(D)} \left[\left(r + \gamma \max_{a'} Q(s',a';\theta_i^-) - Q(s,a;\theta_i) \right)^2 \right]$$

Keep the range of values of a reward within 1 and -1

Deep Q-Learning

Reward Clipping

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim \mathrm{U}(D)}\left[\left(r + \gamma \max_{a'} Q(s',a';\theta_i^-) - Q(s,a;\theta_i)\right)^2\right]$$

Keep the range of values of a reward within 1 and -1

This scaling allows for robust weight updates, keeping the **error derivates** small

And also allows for using the **same learning** over multiple different environments

Deep Q-Learning

Frame Skipping

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim \mathrm{U}(D)}\left[\left(r + \gamma \max_{a'} Q(s',a';\theta_i^-) - Q(s,a;\theta_i)\right)^2\right]$$

As Reinforcement Learning has a **sequential nature**, it is to be assumed that **adjacent frames are similar** and contribute similar information gain.

Deep Q-Learning

Frame Skipping

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim \mathrm{U}(D)}\left[\left(r+\gamma \max_{a'} Q(s',a';\theta_i^-) - Q(s,a;\theta_i)\right)^2\right]$$

As Reinforcement Learning has a sequential nature, it is to be assumed that adjacent frames are similar and contribute similar information gain.

Idea is to **reduce sample rate** or in other words to skip frames during training to increase information gain and thus training and **sample and compute efficiency**

Deep Q-Learning

Reward Function Definition

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim U(D)}\left[\left(r + \gamma \max_{a'} Q(s',a';\theta_i^-) - Q(s,a;\theta_i)\right)^2\right]$$

Definition of the reward function is the means to influence what‘s learned.

This might often not end up as expected

Deep Q-Learning

Reward Function Definition

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim \mathrm{U}(D)}\left[\left(r + \gamma \max_{a'} Q(s',a';\theta_i^-) - Q(s,a;\theta_i)\right)^2\right]$$

Shaped rewards

Increasing rewards in states that are closer to the end goal.

Easier to learn, but also prone to induce bias.

Sparse rewards

Reward shaping

Deep Q-Learning

Reward Function Definition

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim U(D)}\left[\left(r + \gamma \max_{a'} Q(s',a';\theta_i^-) - Q(s,a;\theta_i)\right)^2\right]$$

Shaped rewards

Sparse rewards

Reward at the goal state.

Hard to learn and the lack of positive reinforcement might even make it too difficult.

Rewards shaping

Deep Q-Learning
Reward Function Definition

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s') \sim \mathrm{U}(D)}\left[\left(r + \gamma \max_{a'} Q(s',a';\theta_i^-) - Q(s,a;\theta_i)\right)^2\right]$$

Shaped rewards
Sparse rewards

Reward shaping
Design an objective function that diverts from the actual objective.
This can help in settings with sparse rewards.

Deep Q-Learning

Reward Function Definition

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim \mathrm{U}(D)}\left[\left(r + \gamma \max_{a'} Q(s',a';\theta_i^-) - Q(s,a;\theta_i)\right)^2\right]$$

What kind of reward function is this?

How could you minimize the introduced bias?

How could you transform it into a differently shaped reward?

Deep Q-Learning

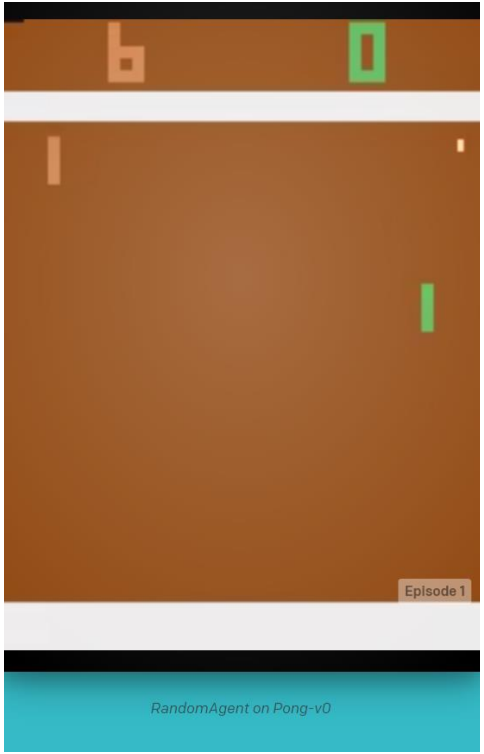
Self-Play

Idea is that agent plays itself and thus improves on the learning objective. There are variations to this strategy

Both sides learn simultaneously

Left side learns

Right side learns



https://gym.openai.com/envs/Pong-v0/

References

[1] [Bergstra and Bengio, 2012] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

[2] [Fridman *et al.*, 2018] Lex Fridman, Benedikt Jenik, and Jack Terwilliger. Deeptraffic: Driving fast through dense traffic with deep reinforcement learning. *CoRR*, abs/1801.02805, 2018.

[3] [Mnih *et al.*, 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[4] [Silver *et al.*, 2017] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[5] [Tuyls and Weiss, 2012] Karl Tuyls and Gerhard Weiss. Multiagent learning: Basics, challenges, and prospects. *Association for the Advancement of Artificial Intelligence*, 2012.

[6] [Olivas *et al.*, 2009] Emilio Soria Olivas, Jose David Martin Guerrero, Marcelino Martinez Sober, Jose Rafael Magdalena Benedito, and Antonio Jose Serrano Lopez. *Handbook Of Research On Machine Learning Applications and Trends: Algorithms, Methods and Techniques - 2 Volumes*. Information Science Reference - Imprint of: IGI Publishing, Hershey, PA, 2009.

OpenAI Gym – Classic Control Problems

Several classic control problems that long have been used to evaluate reinforcement learning algorithms

Description

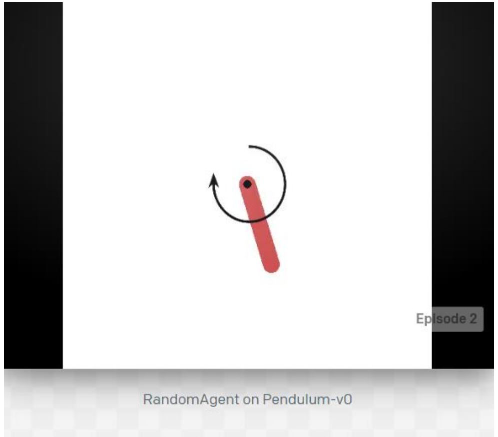
Toolkit for developing and comparing reinforcement learning algorithms

Environments

CartPole

Pendulum

MountainCar



http://gym.openai.com/

OpenAI Gym – Atari Games

Several classic control problems that long have been used to evaluate reinforcement learning algorithms

Description

Large state spaces, and or large action spaces. Long planning horizons. Different variants of the same game allows for transfer learning or generalization evaluations

Environments

SpaceInvaders

Pong

Breakout

Breakout-ram-v0
Maximize score in the game Breakout, with RAM as input

SpaceInvaders-ram-v0
Maximize score in the game SpaceInvaders, with RAM as input

http://gym.openai.com/

OpenAI Gym – Continuous Control Systems

Continuous control problems in locomotion tasks

Description

Large state spaces, and or large action spaces. Long planning horizons. Different variants of the same game allows for transfer learning or generalization evaluations

Environments

Ant

Half Cheetah

Hopper

Humanoid

Ant-v2
Make a 3D four-legged robot walk.

Humanoid-v2
Make a 3D two-legged robot walk.

http://gym.openai.com/

Starcraft Gym – PySC2

It exposes Blizzard Entertainment's StarCraft II Machine Learning API as a Python RL Environment. This is a collaboration between DeepMind and Blizzard to develop StarCraft II into a rich environment for RL research.

http://starcraftgym.com/

Description

Mini-Map games and Full-Map Leaderboards with extensive evaluation tooling

AlphaStar Versus Serral

https://www.youtube.com/watch?v=DMXvkbAtHNY

Best practice

Benchmarking metrics and experimental guidelines

Number of trials

Results might vary significantly by just changing the random seed. Thus we need to statistically ground our experiments

Hyperparameter Tuning

Benchmark Environments and Metrics

Best practice

Benchmarking metrics and experimental guidelines

Number of trials

Hyperparameter Tuning

Ensuring a fair comparison between learning algorithms. For example by ablation analysis (removing a certain hyperparameter, such as no dropout)

Benchmark Environments and Metrics

Best practice

Benchmarking metrics and experimental guidelines

Number of trials

Hyperparameter Tuning

Benchmark Environments and Metrics

Ideally a large mixture of environments should be covered. Same holds for metrics, do not only report on the optimal metrics, report on all.

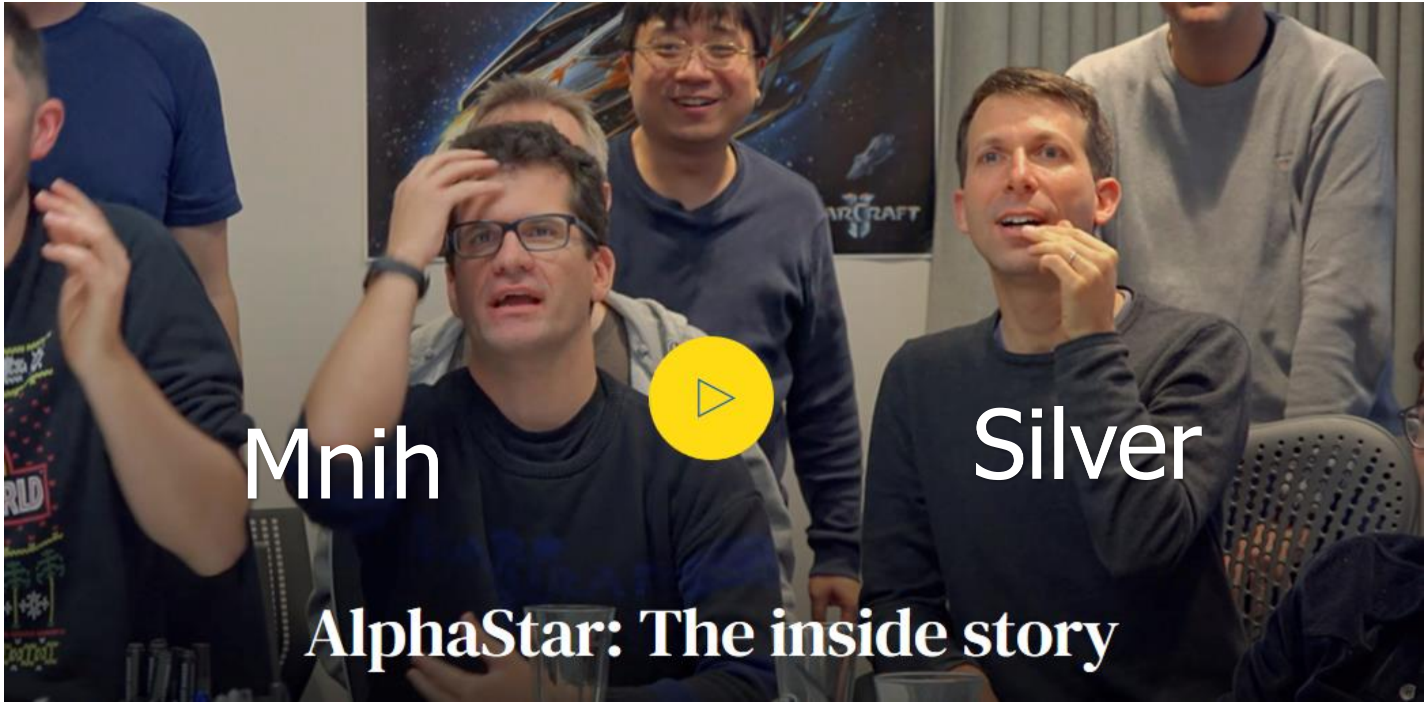
Success over time

Backgamon agent based on Reinforcement Learning (*Tesauro et al.*)	1995
Superhuman performance in Atari games (*Mnih et al.*)	2013
Mastering the game of Go (*Silver et al.*)	2016
Mastering the game of StarCraft II (*Deep Mind*)	2019

Field of Application

Robotics, Autonomous Driving, etc.

Deep Reinforcement Learning - Overview State of the Art



https://deepmind.com/blog/article/alphastar-mastering-real-time-strategy-game-starcraft-ii

TD-gammon

Backgammon playing agent entirely based on reinforcement learning and self-play.

Based on value-function, similar to Q-learning.

Approximation of the value-function based on multi-layer perceptron with one hidden layer.

Temporal Difference Learning and TD-Gammon

By Gerald Tesauro

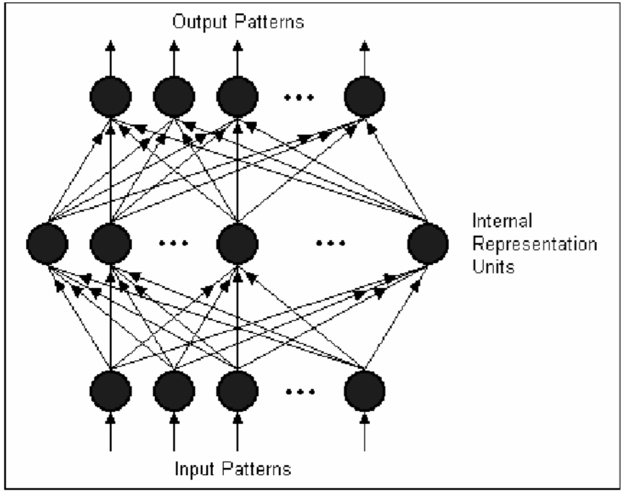


Figure 1. An illustration of the multilayer perception architecture used in TD-Gammon's neural network. This architecture is also used in the popular backpropagation learning procedure. Figure reproduced from [9].

https://cling.csd.uwo.ca/cs346a/extra/tdgammon.pdf

Deep Q-Network

Deep Q-Network with experience replay, fixed target network, mini-batch training, and convolution architecture.

Exploration and exploitation set by epsilon variable.

Ability to master Atari 2600 computer games using pixel input.

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

	B. Rider	Breakout	Enduro	Pong	Q*bert	Seaquest	S. Invaders
Random	354	1.2	0	−20.4	157	110	179
Sarsa [3]	996	5.2	129	−19	614	665	271
Contingency [4]	1743	6	159	−17	960	723	268
DQN	**4092**	**168**	**470**	**20**	**1952**	**1705**	**581**
Human	7456	31	368	−3	18900	28010	3690
HNeat Best [8]	3616	52	106	19	1800	920	**1720**
HNeat Pixel [8]	1332	4	91	−16	1325	800	1145
DQN Best	**5184**	**225**	**661**	**21**	**4500**	**1740**	1075

Table 1: The upper table compares average total reward for various learning methods by running an ϵ-greedy policy with $\epsilon = 0.05$ for a fixed number of steps. The lower table reports results of the single best performing episode for HNeat and DQN. HNeat produces deterministic policies that always get the same score while DQN used an ϵ-greedy policy with $\epsilon = 0.05$.

https://arxiv.org/pdf/1312.5602.pdf

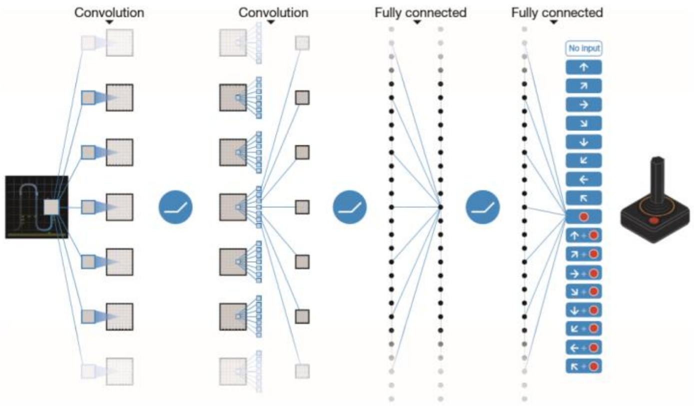
Deep Q-Network

LETTER

doi:10.1038/nature14236

Human-level control through deep reinforcement learning

Volodymyr Mnih[1]*, Koray Kavukcuoglu[1]*, David Silver[1]*, Andrei A. Rusu[1], Joel Veness[1], Marc G. Bellemare[1], Alex Graves[1], Martin Riedmiller[1], Andreas K. Fidjeland[1], Georg Ostrovski[1], Stig Petersen[1], Charles Beattie[1], Amir Sadik[1], Ioannis Antonoglou[1], Helen King[1], Dharshan Kumaran[1], Daan Wierstra[1], Shane Legg[1] & Demis Hassabis[1]



One system that is abel to learn several games without any tweaking from game to game.

Reward clipping

High-dimensional state spaces and high dimensional action spaces

https://storage.googleapis.com/deepmind-media/dqn/DQNNaturePaper.pdf

Rainbow

Extensions to the Deep Q-Network (ablation study)

Double Q-Learning
Prioritized Replay
Dueling Networks
Multi-step learning
Distributional Reinforcement Learning
Noisy Nets

Rainbow: Combining Improvements in Deep Reinforcement Learning

Matteo Hessel DeepMind | **Joseph Modayil** DeepMind | **Hado van Hasselt** DeepMind | **Tom Schaul** DeepMind | **Georg Ostrovski** DeepMind
Will Dabney DeepMind | **Dan Horgan** DeepMind | **Bilal Piot** DeepMind | **Mohammad Azar** DeepMind | **David Silver** DeepMind

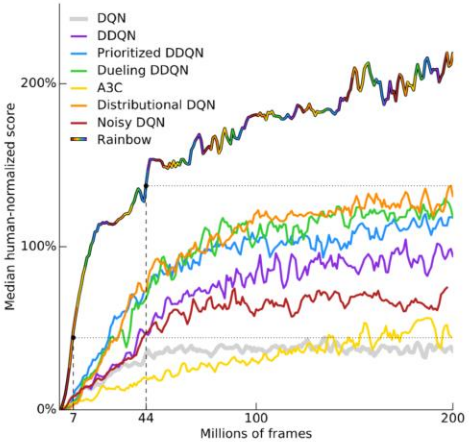


Figure 1: **Median human-normalized performance** across 57 Atari games. We compare our integrated agent (rainbow-colored) to DQN (grey) and six published baselines. Note that we match DQN's best performance after 7M frames,

https://arxiv.org/pdf/1710.02298.pdf

Alpha Star

Combination of Deep Reinforcement Learning with an LSTM module, pointer network

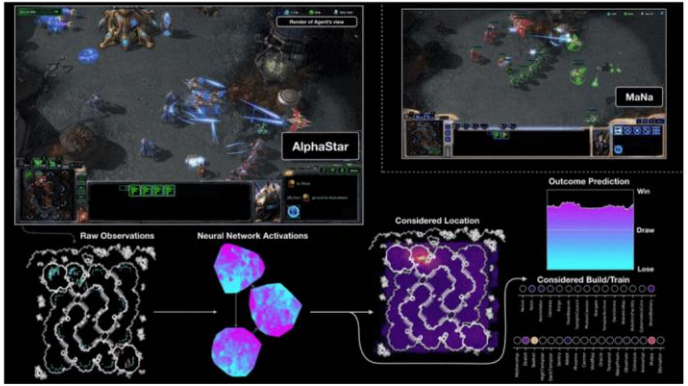
And a novel multi-agent, population-based learning algorithm.

Each agent experiences 200 years of real-time StarCraft play

StarCraft II: A New Challenge for Reinforcement Learning

Oriol Vinyals Timo Ewalds Sergey Bartunov Petko Georgiev
Alexander Sasha Vezhnevets Michelle Yeo Alireza Makhzani Heinrich Küttler
John Agapiou Julian Schrittwieser John Quan Stephen Gaffney Stig Petersen
Karen Simonyan Tom Schaul Hado van Hasselt David Silver Timothy Lillicrap
DeepMind

Kevin Calderone Paul Keet Anthony Brunasso David Lawrence
Anders Ekermo Jacob Repp Rodney Tsing
Blizzard



A VISUALISATION OF THE ALPHASTAR AGENT DURING GAME TWO OF THE MATCH AGAINST MANA. THIS SHOWS THE GAME FROM THE AGENT'S POINT OF VIEW: THE RAW OBSERVATION INPUT TO THE NEURAL NETWORK, THE NEURAL NETWORK'S INTERNAL ACTIVATIONS, SOME OF THE CONSIDERED ACTIONS THE AGENT CAN TAKE SUCH AS WHERE TO CLICK AND WHAT TO BUILD, AND THE PREDICTED OUTCOME. MANA'S VIEW OF THE GAME IS ALSO SHOWN, ALTHOUGH THIS IS NOT ACCESSIBLE TO THE AGENT.

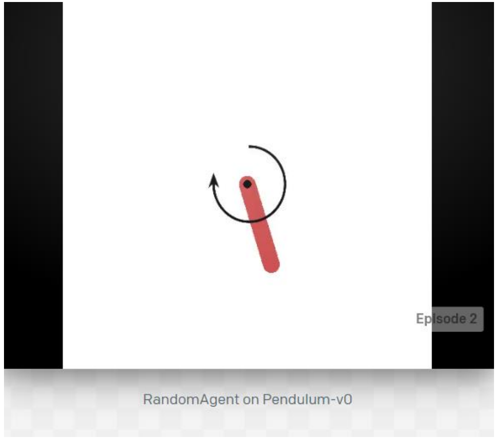
https://deepmind.com/blog/article/alphastar-mastering-real-time-strategy-game-starcraft-ii

Deep Reinforcement Learning is nice but …

… real-world applications are still missing for the most part

… it learns to model environments that are unknown, that is usually not a requirement in domain specific problems

Technical Mechanics and Control Theory



http://gym.openai.com/

Deep Reinforcement Learning is nice but …

… real-world applications are still missing for the most part

… it learns to model environments that are unknown, that is usually not a requirement in domain specific problems

Technical Mechanics and Control Theory

ATLAS Boston Dynamics

Paper

Deploying Rainbow to benchmark on current Traffic Light Control policies. Applied to real-world intersections in Friedrichshafen.

Tutorial

Distributed traffic light control at uncoupled intersections with real-world topology by deep reinforcement learning

Mark Schutera
Institute for Automation and Applied Informatics
Karlsruhe Institute of Technology
Research and Development
ZF Friedrichshafen AG
mark.schutera@kit.edu

Niklas Goby
Chair for Information Systems Research
University of Freiburg
IT Innovation Chapter Data Science
ZF Friedrichshafen AG
niklas.goby@is.uni-freiburg.de

Stefan Smolarek
IT Innovation Chapter Data Science
ZF Friedrichshafen AG
stefan.smolarek@zf.com

Markus Reischl
Institute for Automation and Applied Informatics
Karlsruhe Institute of Technology
markus.reischl@kit.edu

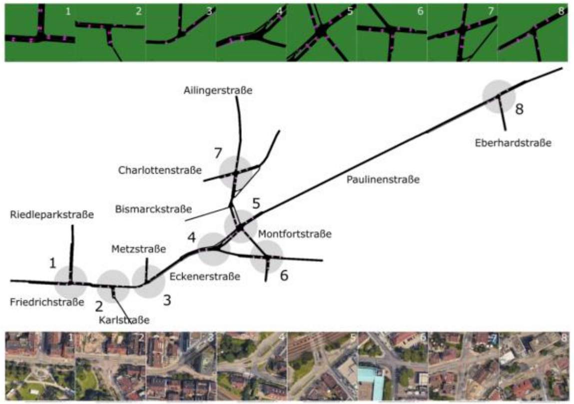


Figure 1: Overview of the Friedrichshafen roadnetwork with streetnames and the locations of the considered junctions in the center. In the top row the junctions are displayed as being present in SUMO. On the bottom row the google maps visualizations of the intersections themselves are shown.

https://arxiv.org/pdf/1811.11233.pdf

Deep Reinforcement Learning - Take it from here

Paper

Analysis and comparison of transfer learning with multi-agent learning in highway traffic.

Extended analysis of congestion and vehicle behavior.

Tutorial

Transfer Learning versus Multi-agent Learning regarding Distributed Decision-Making in Highway Traffic

Mark Schutera[1,4], Niklas Goby[2,3], Dirk Neumann[2], Markus Reischl[1]

[1] Institute for Automation and Applied Informatics, Karlsruhe Institute of Technology
[2] Chair for Information Systems Research, University of Freiburg
[3] IT Innovation Chapter Data Science, ZF Friedrichshafen AG
[4] Research and Development, ZF Friedrichshafen AG

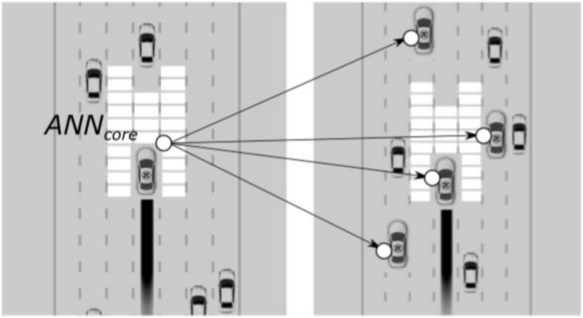


Figure 1: Two screenshots from the micro-traffic simulation. The highlighted cells depict the catchment areas of the safety system, which automatically slows down the car to prevent collisions. The vehicles with the logo represent trainable agents, while those without a logo are not trainable and exhibit random behavior. The left figure shows the training process of a core network ANN_{core}, whereas on the right figure illustrates the pretrained core network being deployed among multiple agents.

https://arxiv.org/pdf/1810.08515.pdf

Deep Reinforcement Learning - Take it from here

Paper

Tutorial

jupyter DQN-TUT Last Checkpoint: 09/19/2019 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Applied Deep Learning Tutorial

contact: Mark.schutera@kit.edu

Deep Reinforcement Learning with Deep-Q-Network (DQN)

Introduction

In this tutorial, you will attempt to implement a Deep-Q-Network that is able to do a classic control. The approaches are build upon the paper by DeepMind: Playing Atari with Deep Reinforcement Learning paper, which first introduces the notion of a Deep Q-Network.

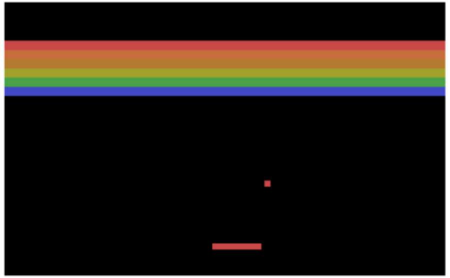


Fig. 1: Breakout environment of the Atari game

Core idea

As you probably remember from the lecture, during trial and error we can learn a policy for our Atari game, and model it within our Q-matrix. This is done with a deep neural network. After training, this Q-matrix gives us an estimate of the expected reward when taking action a in state s: Q(s, a). Playing the action with the maximum Q-value in any given state is the same as playing optimal, or following a full exploitation strategy.

OpenAI Gym

OpenAI Gym is a library that can simulate a large number of reinforcement learning environments, including Atari games (these need to be installed additionaly). You will need Python 3.5+

> pip install gym

Taking our cart pole on a first ride

Thanks for your time
Questions?

Contact!

mark.schutera@kit.edu