Deep Reinforcement Learning

Prof. Dr.-Ing. Björn Schuller
07.01.2020

Content

Content

- **Major types of machine learning algorithms**
  - Supervised learning
    - Learning on: Labeled data
  - Unsupervised learning
    - Learning on: Unlabeled data
  - Reinforcement learning
    - Learning on: Reward-based interaction with environment

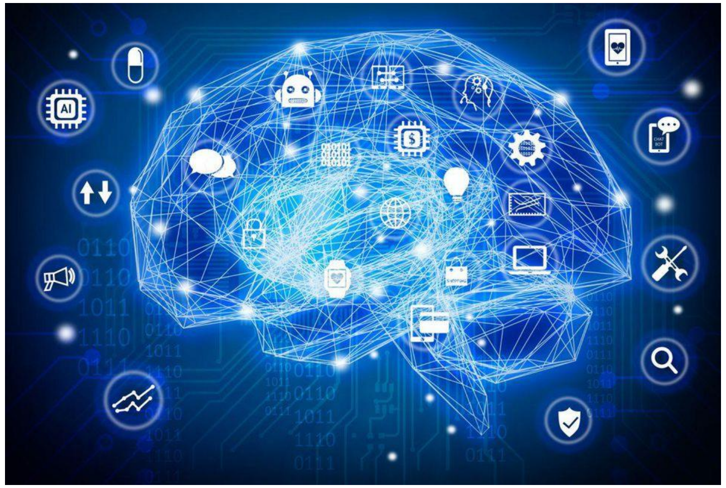


image source: https://images.app.goo.gl/Dn8cxSdvKJHoiCRTA

- **Supervised Learning**
  - Main topic of this lecture
  - Input: Set of known training data
    - E.g. pictures, sound waves, pre-processed features, etc.
  - Output: Predictions for training data
    - Classification (e.g. cat or dog)
    - Regression (e.g. cat or dog localisation)
  - Deep Learning (Neural Networks)
    - Predict output from input

- **Supervised Learning**
  - Good performance if
    - Big amount of labeled training data
  - Acquisition of labeled training data
    - Time-consuming
    - Not suited for every use-case

Unsupervised Learning

- Input: Set of training data without annotations
  - E.g. pictures, sound waves, pre-processed features, etc.
- Output: Clustering the training data
  - Exploratory data analysis (e.g. cat or plant)
- Deep Learning (Neural Networks)
  - Representation learning

- **Case: Interactive Agent**
  - Input: Environment
    - Described by feature
  - Output: Action
    - Given set of possible actions
  - Predefined Goal
    - E.g. cross the street (robot), chat-bot

- **Approaches**
  - Unsupervised Learning
    - Only clustering of environment data possible
  - Supervised Learning
    - Labeling best action for each data point
      - Very time-consuming (if possible)
      - Potentially many actions leading to goal

- **Approaches**
  - Reinforcement Learning
    - Agent exploring environment
    - Agent obtaining reward for every action
      - Reward based on environmental change to action
    - Agent learning rule system from reward

image source:
https://images.app.goo.gl/PqqiBdbrYc9oQabF6

Supervised learning -- Learning from teacher

Cross the street (robot)

Label: move

Label: wait

image source:
https://www.youtube.com
/watch?v=N1lhHHkUzYg

Unsupervised learning -- Clustering the environment

Cross the street (robot)

image source: https://www.youtube.com/watch?v=N1IhHHkUzYg

Reinforcement learning -- Learning from experience

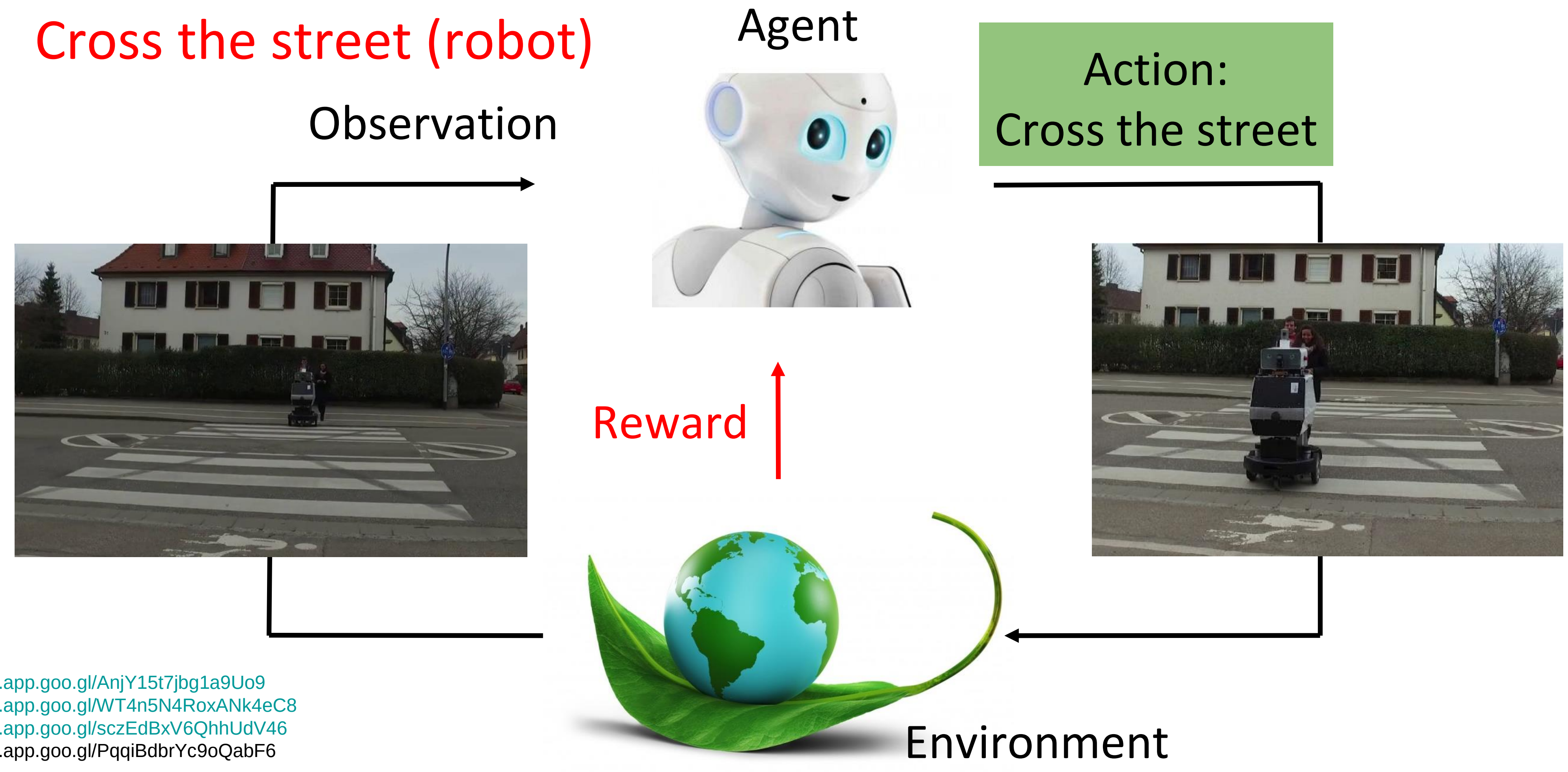


image source:
https://images.app.goo.gl/AnjY15t7jbg1a9Uo9
https://images.app.goo.gl/WT4n5N4RoxANk4eC8
https://images.app.goo.gl/sczEdBxV6QhhUdV46
https://images.app.goo.gl/PqqiBdbrYc9oQabF6

Interactive Agent

Supervised learning

Not flexible (sometimes generate good dialogue, sometimes bad)

Chat-bot

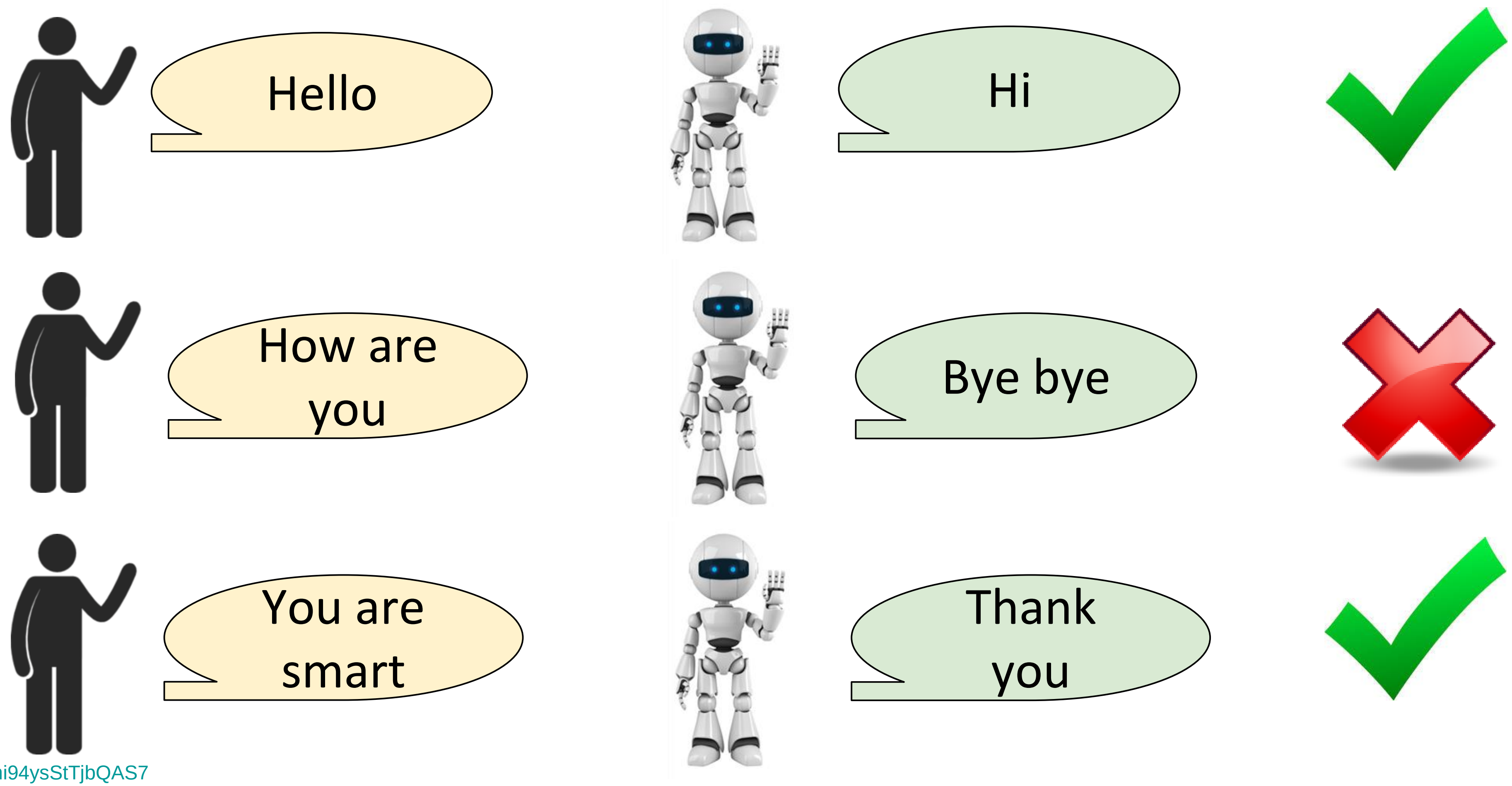


image source:
https://images.app.goo.gl/KPmi94ysStTjbQAS7
https://images.app.goo.gl/M4fxtmBjKXKHBPGR8

Unsupervised learning

Mostly the dialogues are bad

Chat-bot

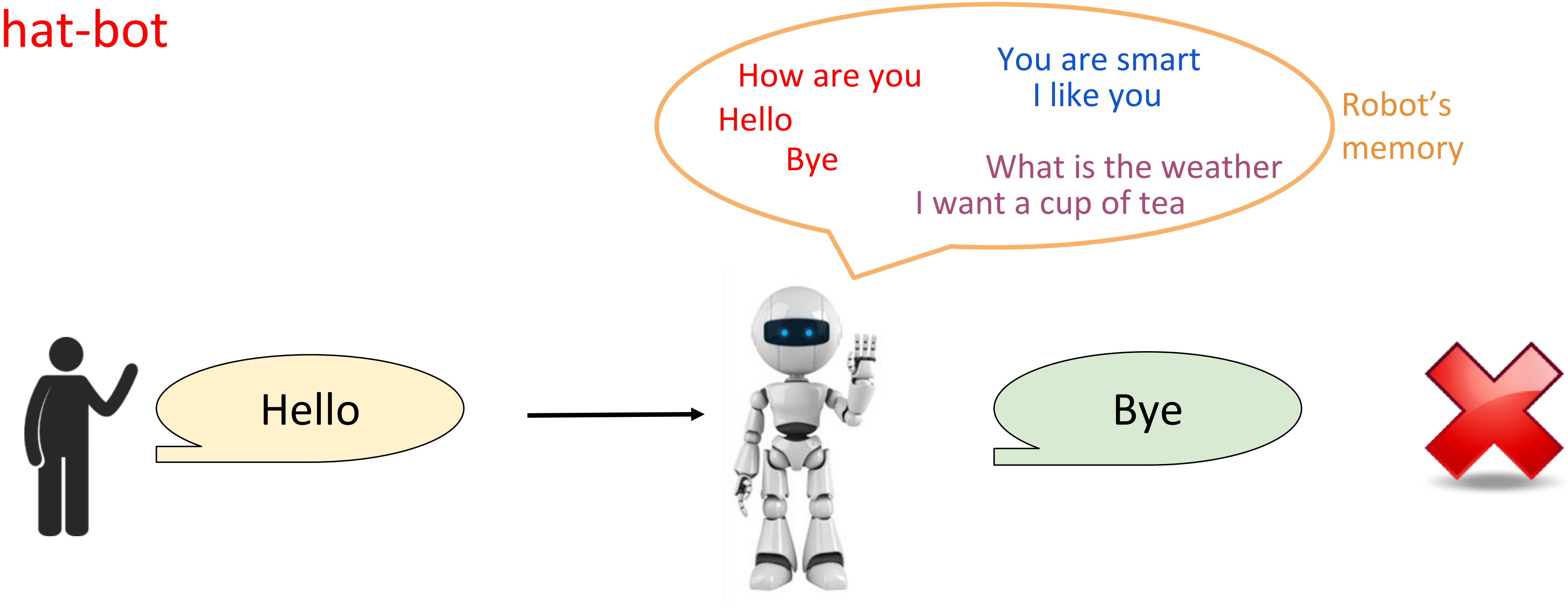


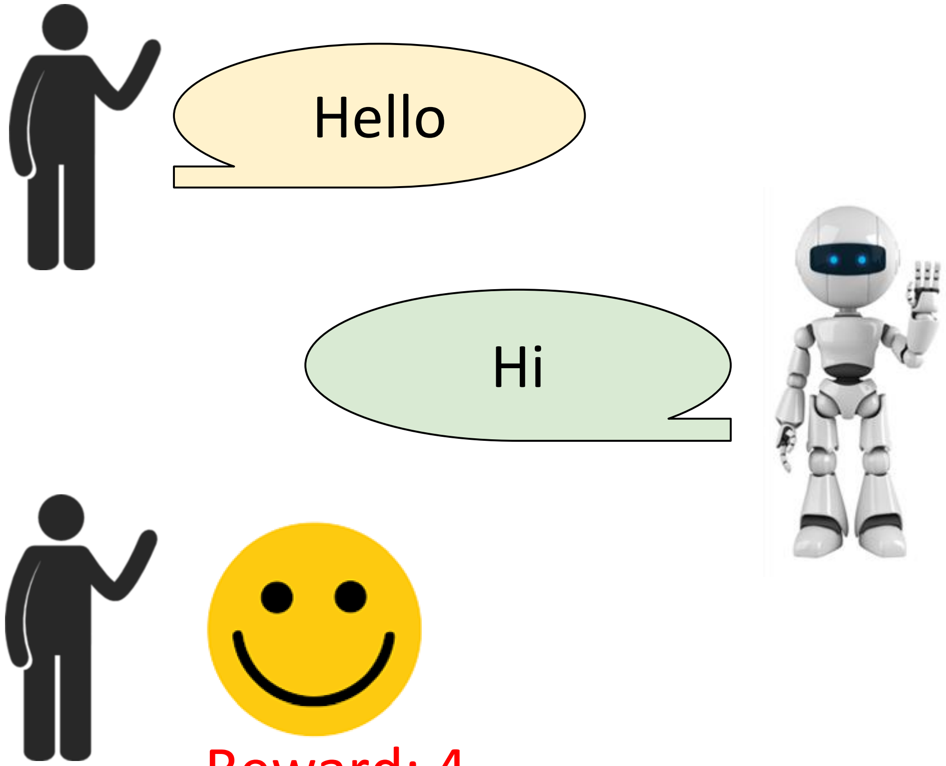
image source:
https://images.app.goo.gl/KPmi94ysStTjbQAS7
https://images.app.goo.gl/M4fxtmBjKXKHBPGR8

Interactive Agent

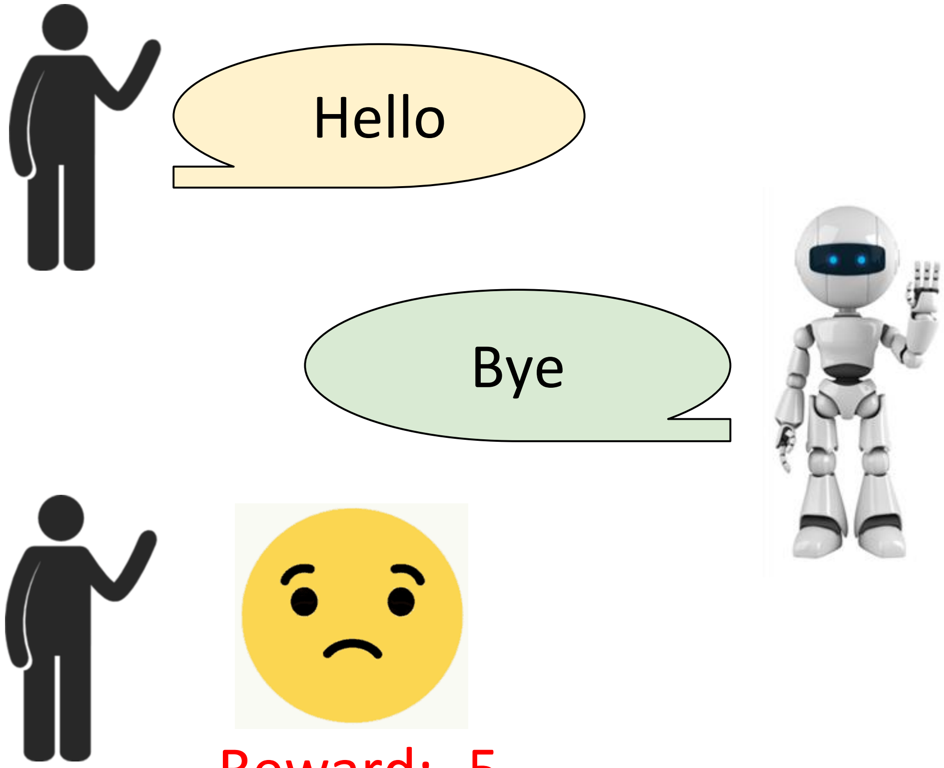
Reinforcement learning

Learn to maximize the expected reward

Chat-bot



Reward: 4



Reward: -5

image source:
https://images.app.goo.gl/KPmi94ysStTjbQAS7
https://images.app.goo.gl/M4fxtmBjKXKHBPGR8
https://images.app.goo.gl/RKkRin3WGZ9ThBLr7

- Supervised Learning
  - learning a static model to predict the label
  - classification, regression
- Unsupervised Learning
  - clustering, segmentation, dimension reduction
- Reinforcement Learning
  - learning a dynamic model
  - reward system, decision process, recommendation system

Playing games -- a widely study in reinforcement learning

Input:

- What the robot observes is pixels

Output:

- The robot give action commands

Procedure:

- Machine learns to win by maximizing the reward

Tools:

- Gym: https://gym.openai.com/
- Universe: https://openai.com/blog/universe/

image source: https://www.youtube.com/watch?v=iqXKQf2BOSE
reference: Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G. and Petersen, S., 2015. Human-level control through deep reinforcement learning. *Nature*, *518*(7540), p.529.

Universität Augsburg University

Score (reward)

Object: Kill all the aliens

Shields

Fire: Move left or right

image source: https://www.youtube.com/watch?v=iqXKQf2BOSE

Applications of Reinforcement Learning

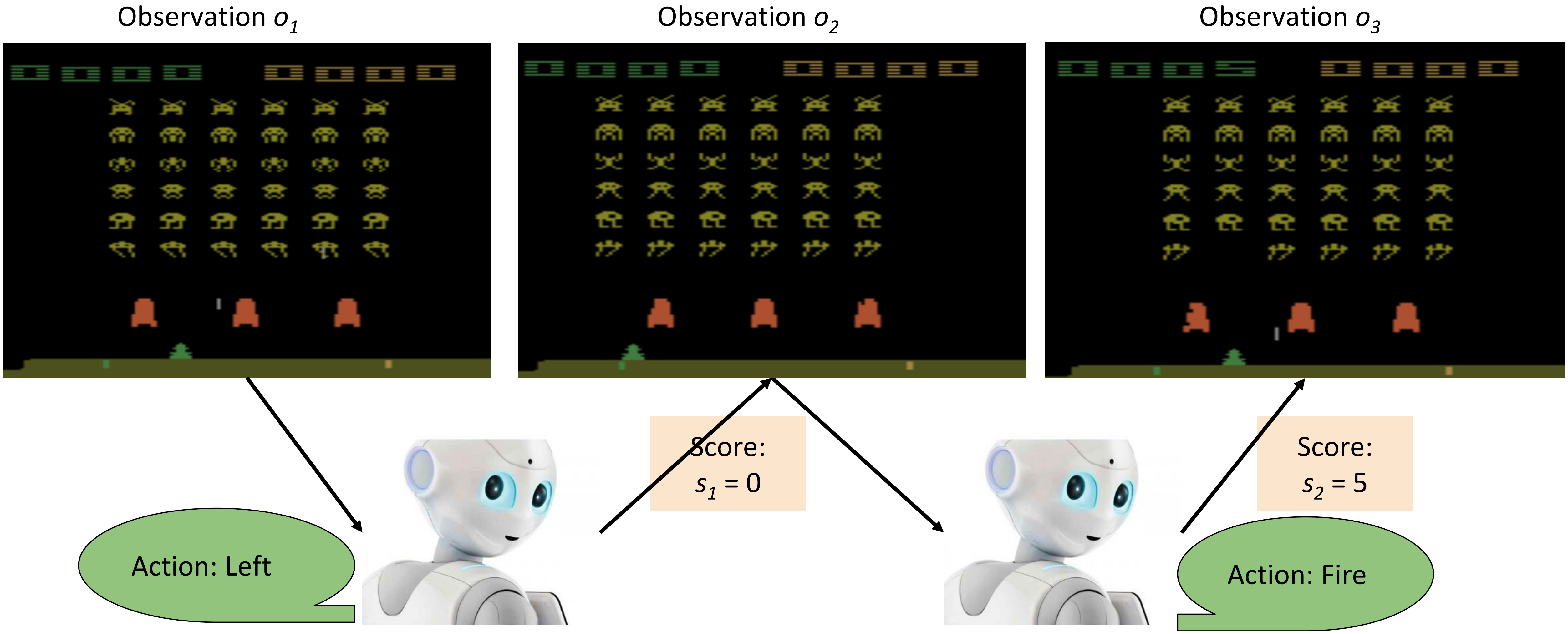
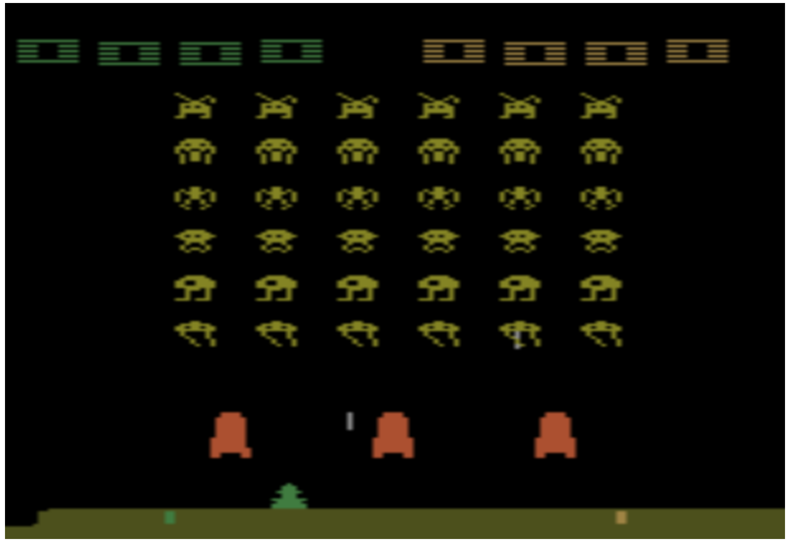


image source: https://images.app.goo.gl/PqqiBdbrYc...

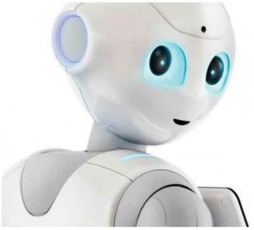
Applications of Reinforcement Learning

Observation o_1



Observation o_2

Observation o_3



Learning to maximize the expected cumulative score.

Training t iterations

Win

or

Game over

(Score: s_t)

Applications of Reinforcement Learning

More applications

Driving

Four actions and six parameters:

- Dash (power, direction)
- Turn (direction)
- Tackle (direction)
- Kick (power, direction)

video source: https://www.youtube.com/watch?v=lV5JhxsrSH8

reference:
Hausknecht, M. and Stone, P., 2015. Deep reinforcement learning in parameterized action space. *arXiv preprint arXiv:1511.04143*.
Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D. and Wierstra, D., 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Applications of Reinforcement Learning

Autonomous Helicopter

Four actions:

- longitudinal (front-back) cyclic pitch control
- latitudinal (left-right) cyclic pitch control
- main rotor collective pitch control
- tail rotor collective pitch control

video source: https://www.youtube.com/watch?v=0JL04JJjocc
reference:
Abbeel, P., Coates, A. and Ng, A.Y., 2010. Autonomous helicopter aerobatics through apprenticeship learning. *The International Journal of Robotics Research*, *29*(13), pp.1608-1639.

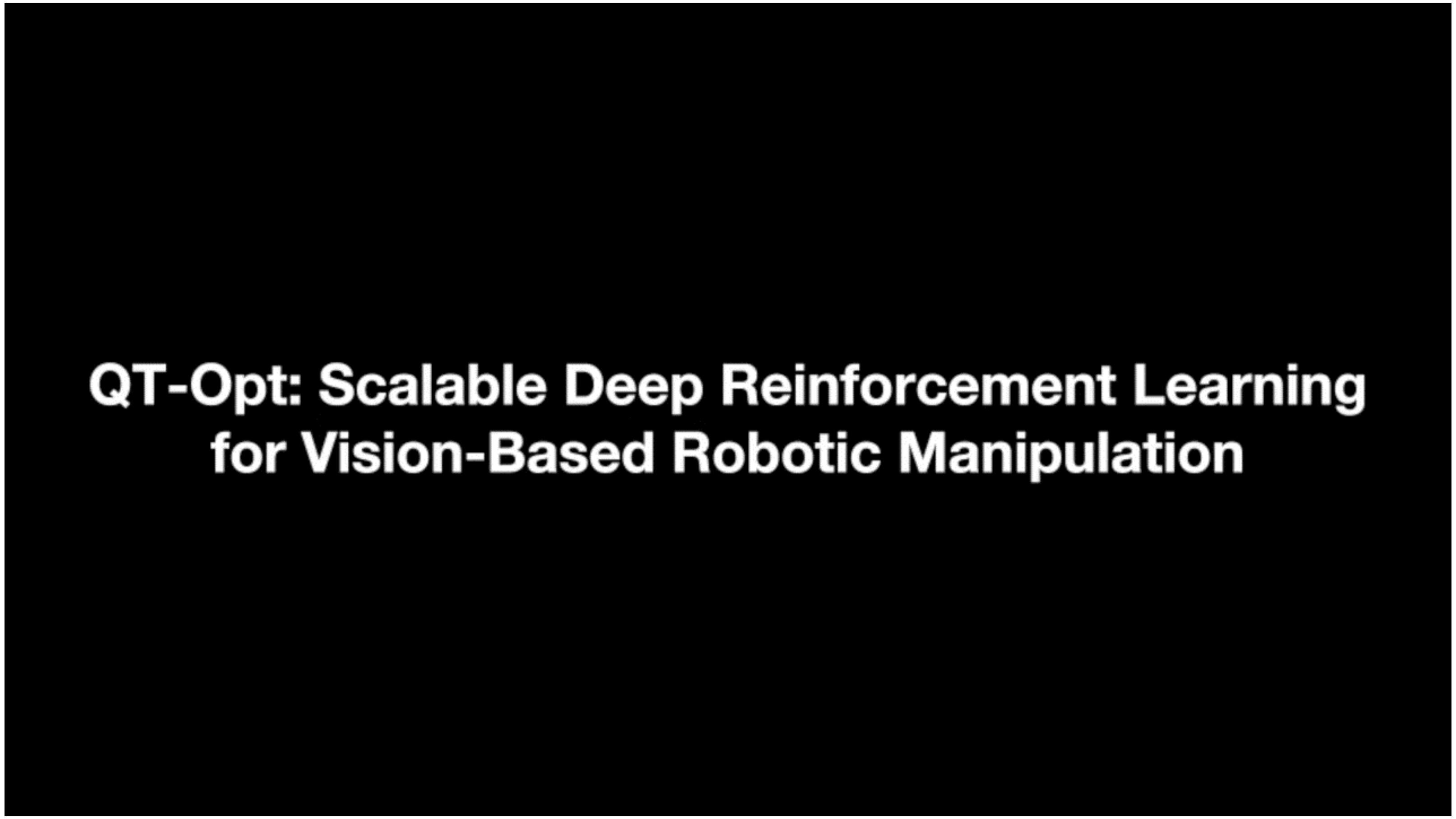
Robot

Discrete actions:

- open
- close
- terminate

Continuous actions:

- cartesian vector
- gripper rotation



video source: https://www.youtube.com/watch?v=W4joe3zzglU
reference:
Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V. and Levine, S., 2018. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293*.

Difficulties of reinforcement learning

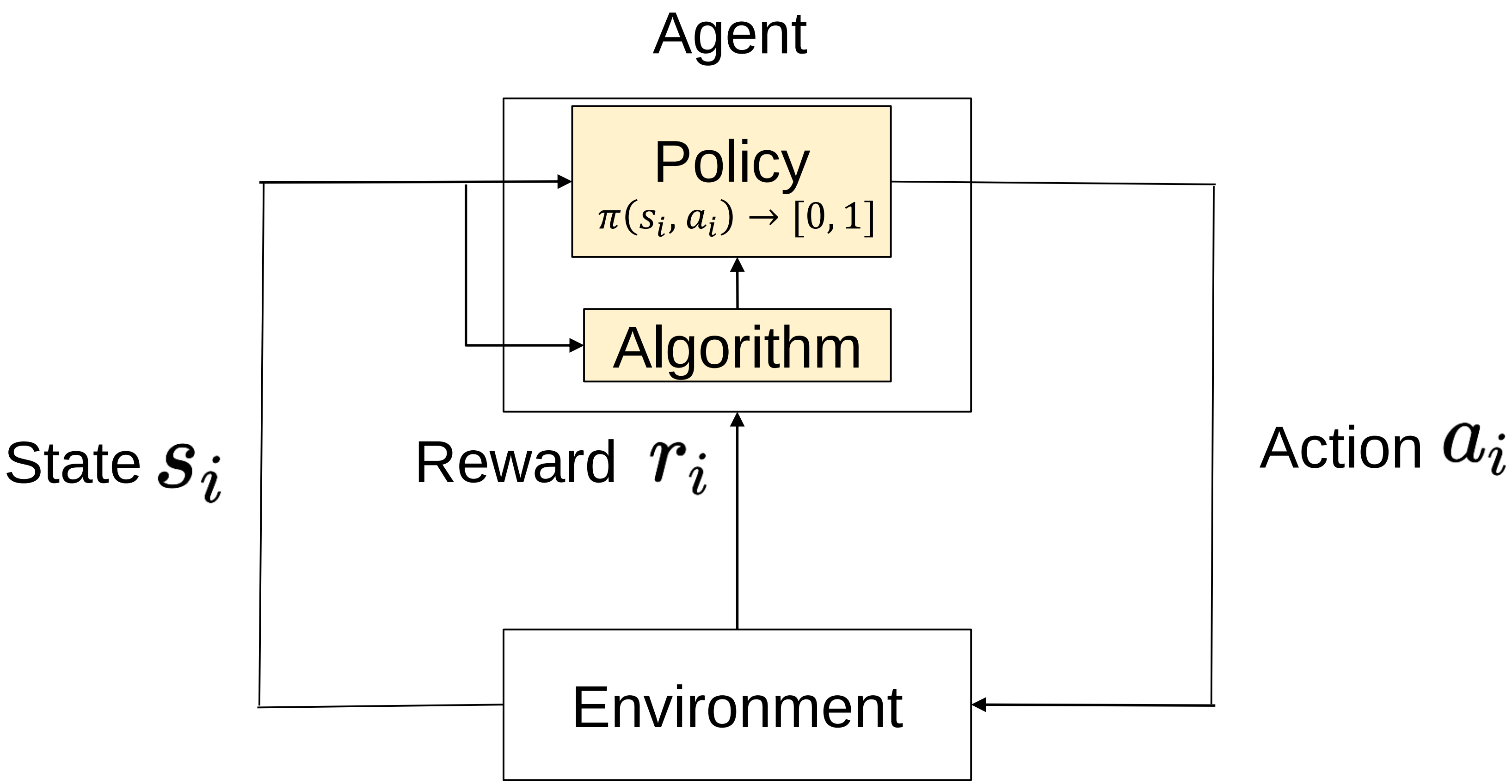
- Reward delay
  - Only ‘fire’ obtains reward, but the moving direction (left or right) can affect the result of ‘fire’
    - learn to coordinate the actions
- Agent’s actions affect the reward
  - learn to explore the actions which have not been tried

Content

- Introduction
- Reinforcement Learning
- Deep Reinforcement Learning

Universität Augsburg University

- **Framework of Reinforcement Learning**



- **State**
  - Description of environment and agent
  - Often hand-crafted features
  - Current state s_i extracted from current situation
  - Knowledge of possible state $s_i \in S$
  - Examples
    - Coordinates of Super Mario, distance to closest obstacle

$$s_i = (x_M, y_M, d_O)$$

- **Action**
  - Actions available to agent
  - Knowledge of available actions $a_i \in A$
  - Examples
    - Jump, Move right, Move left

      $A = \{J, R, L\}$
  - Leading to new situation s_{i+1}

- **Policy**
  - Probability of choosing action a_i in situation s_i
    $\pi(s_i, a_i) \rightarrow [0, 1]$
  - Learning target during training
  - Applied at each timestep

- **Reward**
  - User rewarded with numerical value $r_i \in \mathbb{R}$
  - Can depend on situation after action s_{i+1}
  - Applied at each timestep
  - Basis for learning goal

- **Problem**
  - State-transitions and rewards difficult to predict
  - Non-deterministic environment
    - Same action at different points in time
      → Different reward
      → Different resulting state
  - E.g. Football player shooting on goal
    - Hit: High reward
    - Miss: Low reward

- **Exploitation vs. Exploration**
  - Exploitation
    - Choosing most promising action
    - Guaranteed high reward (if well explored)
    - No new information
  - Exploration
    - Choose less explored action
    - Possibly low reward
    - Gain of information

- **Goal of RL**
  - Maximizing expected future reward (value) V_π
    - Expected reward depending on policy $\pi(s_i, a_i) \rightarrow [0,1]$

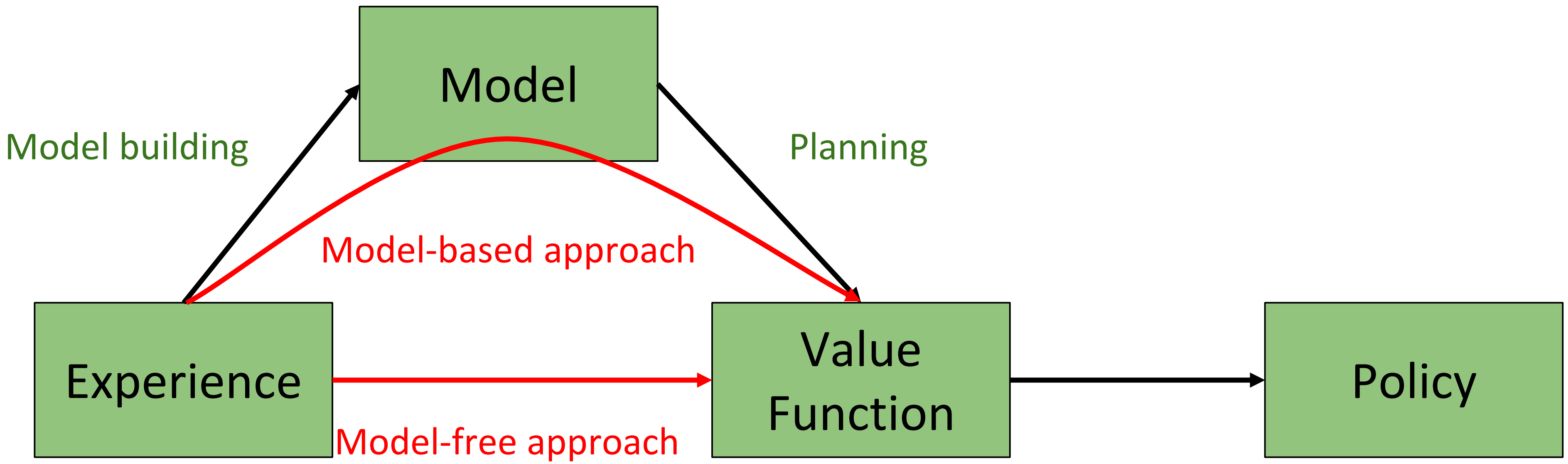
    → Learn policy to maximize reward
  - Problem
    - Infinite runtime of algorithm

      → Infinite sum of rewards
    - Define expected future reward V_π

Expected future reward (value) functions

- Finite-horizon model $V_\pi = E(\sum_{t=0}^{h} r_t)$
  - Finite horizon of steps
- **Infinite-horizon model** $V_\pi = E(\sum_{t=0}^{\infty} \gamma^t r_t), 0 \leq \gamma \leq 1$
  - γ : discount rate
- Average-reward model $V_\pi = \lim_{h\to\infty} E(\frac{1}{h}\sum_{t=0}^{h} r_t)$

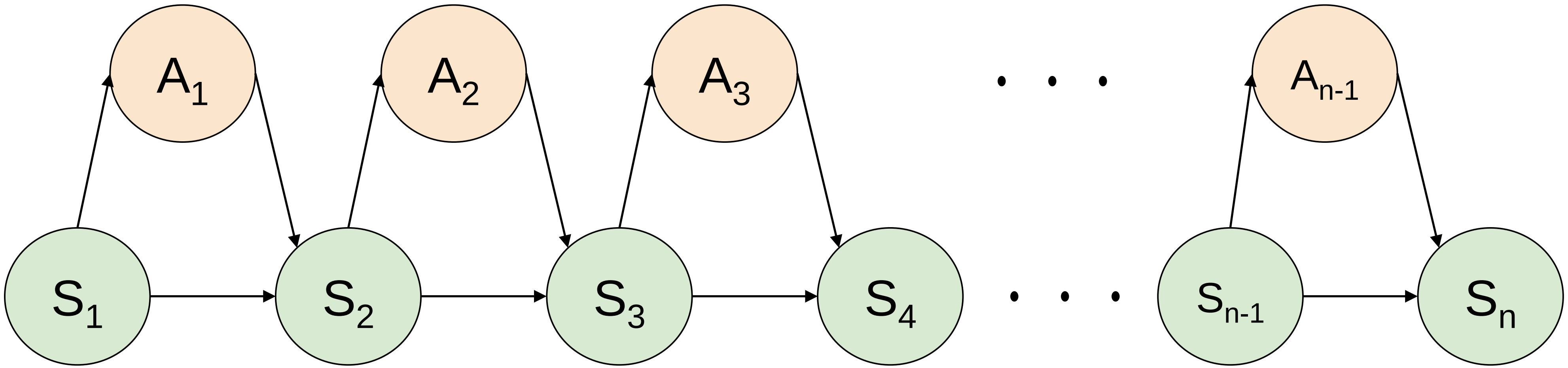
- **Two basic reinforcement learning approaches**



reference: Asadi, K., 2015. Strengths, weaknesses, and combinations of model-based and model-free reinforcement learning. *Department of Computing Science University of Alberta*.

- **Markov Decision Process (MDP)**
  - Modeling the problem
  - MDP defined by
    - Set of possible states S
    - Set of possible actions A
    - Reward function $\mathrm{R}(s_i, a_j) \rightarrow \mathbb{R}$
      - Expected reward when choosing action a_j in situation s_i
    - Transition function $\mathrm{T}\left(s_i, a_j, s_l\right) \rightarrow [0, 1]$ Probability of transitioning from state s_i to state s_l when choosing action

- **Markov Decision Process (MDP)**



Optimal Value Function

- Assuming MDP model known
- Optimal reward gained from state
  - I.e. using optimal policy
- Infinite-horizon case $V^*(s) = \max_{\pi} E(\sum_{t=0}^{\infty} \gamma^t r_t)$
- As Bellman equation
  - Optimal solution consisting of optimal solutions for subproblems $V^*(s) = \max_{a}(R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V^*(s'))$

Universität Augsburg University

- **Value iteration**

  - Learn optimal value iteratively

initialize $V(s)$ *arbitrarily*

loop until policy good enough

    loop for $s \in S$

        loop for $a \in A$

$$Q(s,a) := R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V(s')$$

$$V(s) := \max_a Q(s,a)$$

    end loop

end loop

  - Derive optimal policy from optimal values

- **Example -- chat bot**

before training with MDP

S: RELEASE AIRLINE	*Do you want to choose another airline?*
U:	What?
S: RETRIEVAL	
S: CONSTRAIN DEPART. TIME	*When do you want to leave?*
U:	UH? I want to go to Boston.
S: OUTPUT DATA	*There are 12,000 flights...*
U:	I don't understand, can you please show me the flights from San Francisco to Boston?
S: CLOSE DIALOGUE	*Thank you for using AT&T.*

after training with MDP

S: GREETING	*This is the AT&T flight information. How may I help you?*
U:	I want to go to Boston.
S: CONSTRAIN ORIGIN	*Where do you want do leave from?*
U:	*San Francisco*
S: CONSTRAIN AIRLINE	Which airline?
U:	*Delta*
S: RETRIEVAL (30 flights)	
S: CONSTRAIN DEPART. TIME	*When do you want to leave?*
U:	*In the late afternoon.*
S: RETRIEVAL (30 flights)	
S: OUTPUT DATA	*Flight...leaves at ..., flight...leaves at...*
U:	Thanks.
S: CLOSE DIALOGUE	*Thank you for using AT&T.*

reference: Levin, E., Pieraccini, R. and Eckert, W., 1998, May. Using Markov decision process for learning dialogue strategies. In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181)* (Vol. 1, pp. 201-204). IEEE.

- **Problem**
  - MDP model not known
  - Model-based algorithms
    - Learn MDP model → Derive optimal policy
  - Model-free algorithms
    - Learn policy without learning model

- **Model-based approach**

  - Example: Certainty equivalence
    - Straight forward method
    - Random selection of actions → Explore environment
    - Keep track of transitions and rewards
    - Calculate T and R statistically
  - Other Algorithms
    - Dyna
    - Prioretized Sweeping

- **Model-free approach**
  - Learn policy rather than MDP model
  - Algorithms
    - **Q-learning**
    - Adaptive heuristic critic
    - Model-free learning with average reward
  - Model-free vs Model-based
    - Superiority of either heavily discussed

- **Q-learning**
  - Popular Algorithm for RL
  - Given situation s
  - Expected reward $Q^*(s,a)$
    - choosing action a
    - subsequently choosing optimal action
  - Optimal Value $V^*(s) = \max_a Q^*(s,a)$

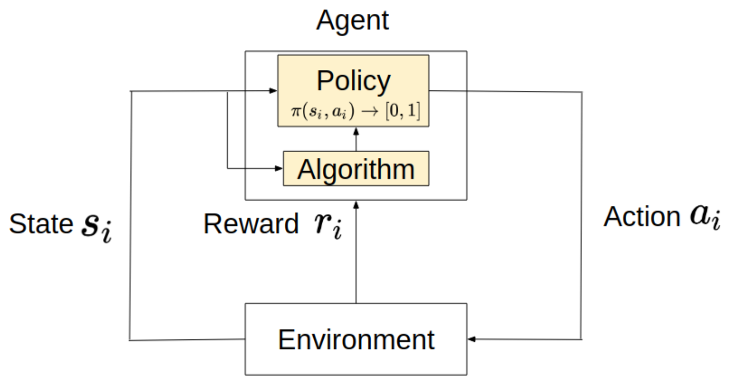
$$\rightarrow Q^*(s,a) = R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') \max_{a'} Q^*(s',a')$$

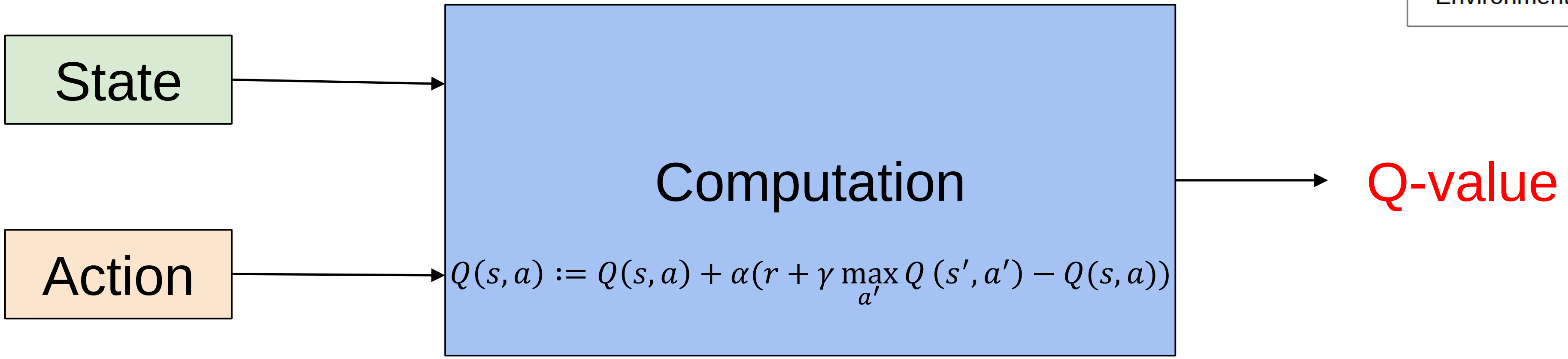
- **Q-learning**

  - Learning Q-Values
    - Performing action a in situation s
    - Obtaining reward r transitioning into state s'

    $$\rightarrow Q(s,a) := Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right)$$
  - Exploration: Choose actions (partially randomly)
  - Exploitation: Choose action with optimal Q-Value

Q-learning


Agent
Policy
$\pi(s_i, a_i) \to [0,1]$
Algorithm
State s_i
Reward r_i
Action a_i
Environment


State
Action
Computation
$Q(s,a) := Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a))$
Q-value

- **Q-learning**
  - The computation part is usually a table -- Q-table

		Action							
State	1	up	down	left	right	left-up	left-down	right-up	right-down
	2	0	0	0	0	0	0	0	0
	3	0	0	0	0	0	0	0	0
	4	0	0	0	0	0	0	0	0
	5	0	0	0	0	0	0	0	0
	6	0	0	0	0	0	0	0	0
	7	0	0	0	0	0	0	0	0
	8	0	0	0	0	0	0	0	0

training →

		Action							
State	1	up	down	left	right	left-up	left-down	right-up	right-down
	2	0.12	0.34	0.35	0.78	0.01	0.25	0.57	0.68
	3	...	...	...	...	...	...	...	...
	4	...	...	...	...	...	...	...	...
	5	...	...	...	...	...	...	...	...
	6	...	...	...	...	...	...	...	...
	7	...	...	...	...	...	...	...	...
	8	...	...	...	...	...	...	...	...

- **Improvement of Q-learning**

  - Fuzzy Q-learning
    - a collection of fuzzy rules as an agent that produces continuous-valued actions
  - Object focused Q-learning
    - treats the state space as a collection of objects organized into different object classes
  - …

references:
Glorennec, P.Y. and Jouffe, L., 1997, July. Fuzzy Q-learning. In *Proceedings of 6th international fuzzy systems conference* (Vol. 2, pp. 659-662). IEEE.
Cobo, L.C., Isbell, C.L. and Thomaz, A.L., 2013, May. Object focused q-learning for autonomous agents. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems* (pp. 1061-1068). International Foundation for Autonomous Agents and Multiagent Systems.

Content

Universität Augsburg University

- **Reinforcement Learning**
  - So far
    - Provided set of situations
    - Provided set of actions
    - Provided reward function
    - Determine optimal policy function
  - Problem
    - Where does the set of situations come from?

- **Situation Sets**
  - Situation Variables often handcrafted
    - Subjective selection of features
    - Only applicable to one task
  - Situation Variables based on sensory input
    - Pre-processing necessary (high dimensionality)
    - Applicable to multiple tasks
    - E.g. computer game: pixels of screen
    - Often used in Deep Reinforcement Learning (DRL)

- **General Approach**

  - Try to learn Q-Values

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') \max_{a'} Q^*(s',a')$$

  - So far linear updates to Q-Values
    - Guaranteed Convergence
    - No Generalisation between actions and situations

$$Q(s,a) \coloneqq Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

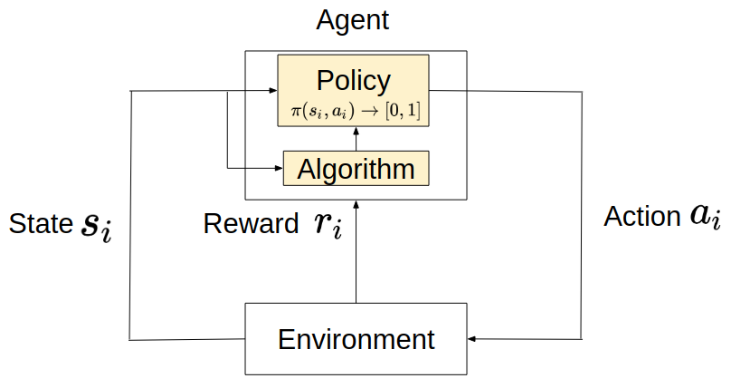
- **Approximator Approach**

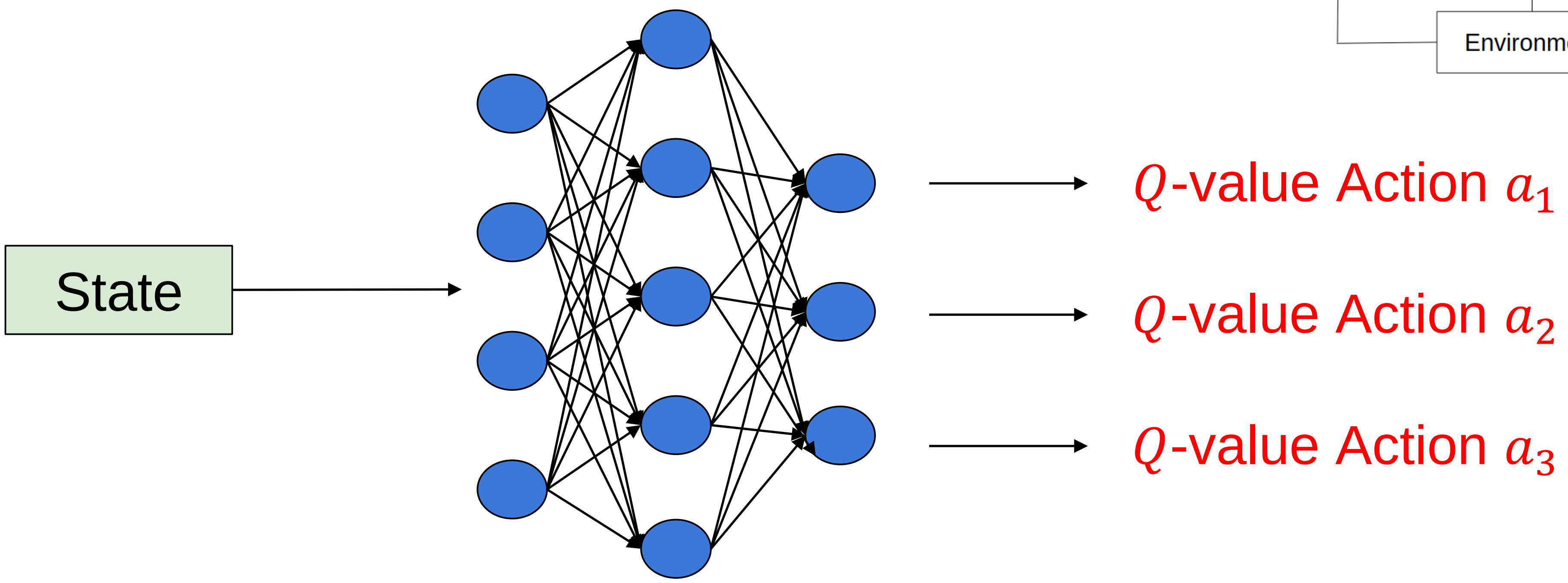
  - Estimate action Value $Q(s, a, \theta) \approx Q^*(s, a)$
    - Estimation often linear in θ
  - Deep Q-Network
    - Using DNN for estimation of Q^*
    - Non-linear functions in θ
    - Convergence not guaranteed
    - Good generalisation

Q-Network

- Prediction of Q-values
  - Depend on parameters θ_i of Neural Net at iteration i
- Training of Q-Values
  - Given situation s
  - Calculate Q-Value $Q(s, a, \theta_i)$ for actions $a \in A$
  - Perform action a e.g. with highest Q-Value (greedy)
  - Get reward r and get to situation s'

- **Q-Network**


Agent
Policy
$\pi(s_i, a_i) \to [0,1]$
Algorithm
State s_i
Reward r_i
Action a_i
Environment


State
Q-value Action a_1
Q-value Action a_2
Q-value Action a_3

- **Q-Network**

  - Training of Q-Values
    - Compare Q-Value $Q(s, a, \theta)$ with Q-Values for s'
    - Approximating target value

$$y = r + \gamma \max_{a'} Q(s', a', \theta_i^-)$$

    - θ_i^-: DNN parameters of prior iteration (e.g. $i - 1$)

- **Q-Network**

  - Training of Q-Values
    - Define loss function
      - For minibatch of tuples (s, a, r, s')
      - Mean squared error

$$L_i(\theta_i) = E_{s,a,r}[(E_{s'}[y] - Q(s, a, \theta_i))^2]$$

    - Calculate gradient $\nabla_{\theta_i} L(\theta_i)$ for update steps

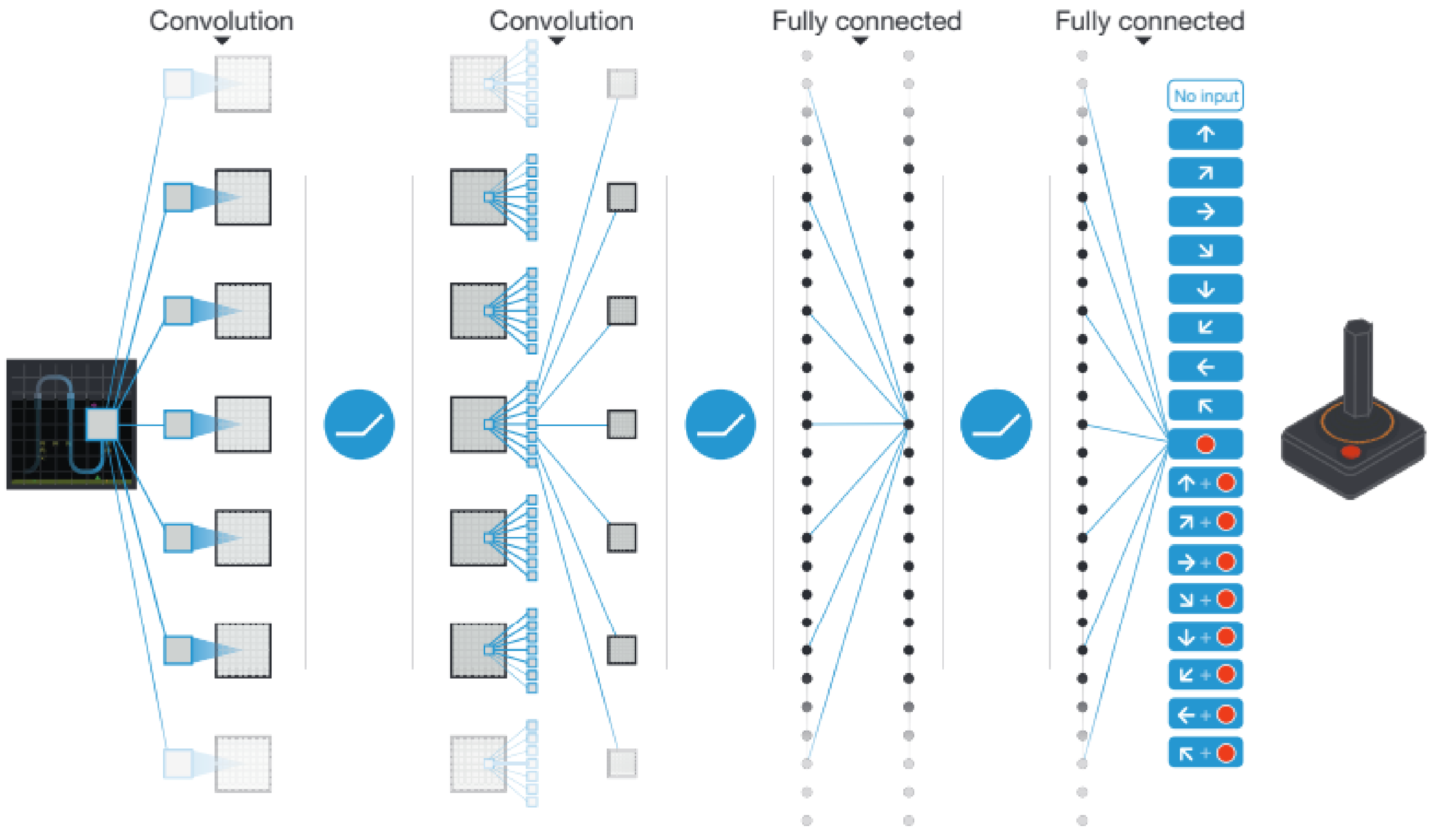
- **Q-Network**

  - Summary
    - Deep Reinforcement Learning approach
    - Model-free
    - Learning Q-Values with DNNs
    - Labels based on previous data

      → No supervised learning

- Deep Q-learning

Convolutional Neural Networks applied to select actions



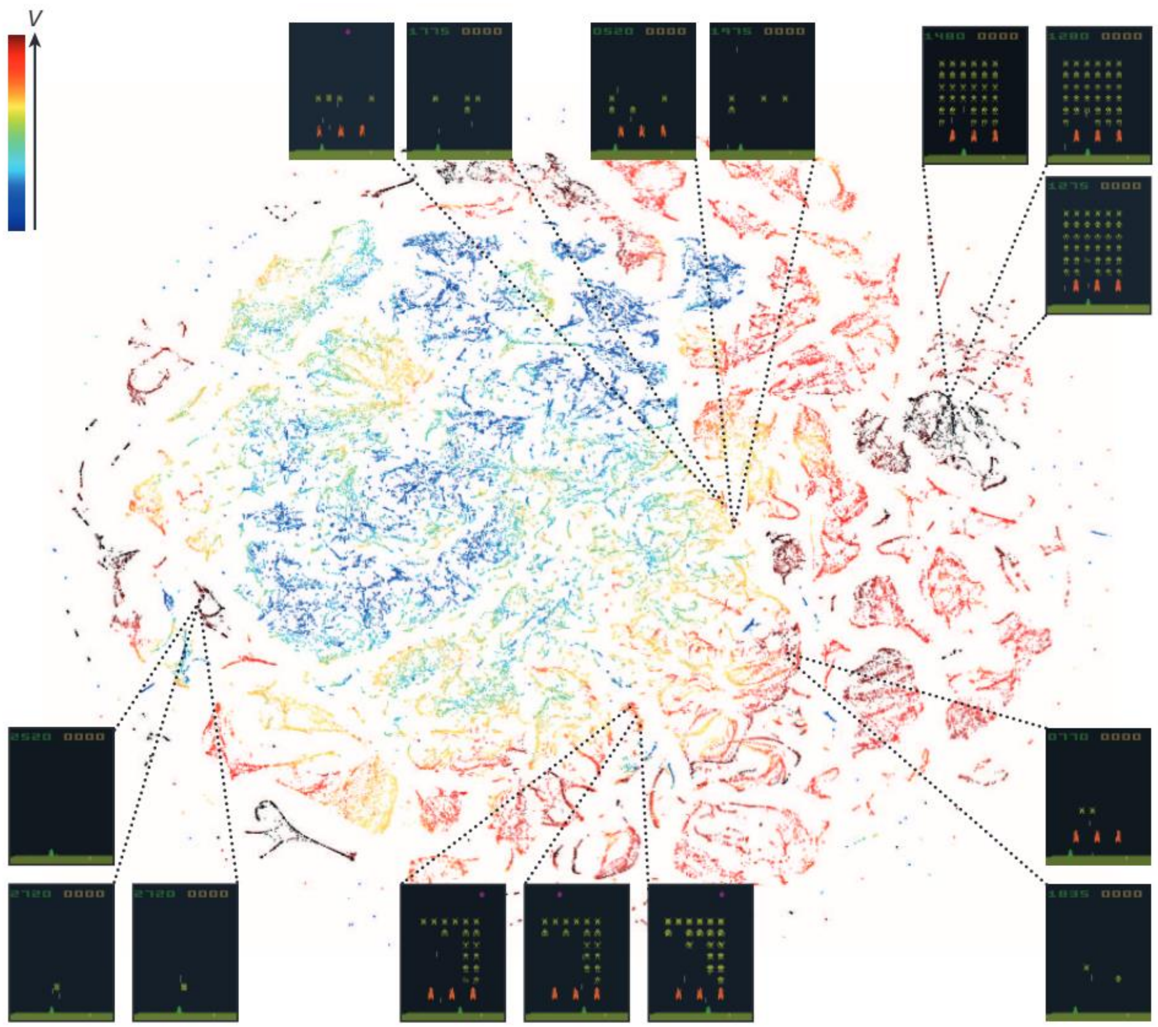
reference: Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G. and Petersen, S., 2015. Human-level control through deep reinforcement learning. *Nature*, *518*(7540), p.529.

- Deep Q-learning

Space Invaders

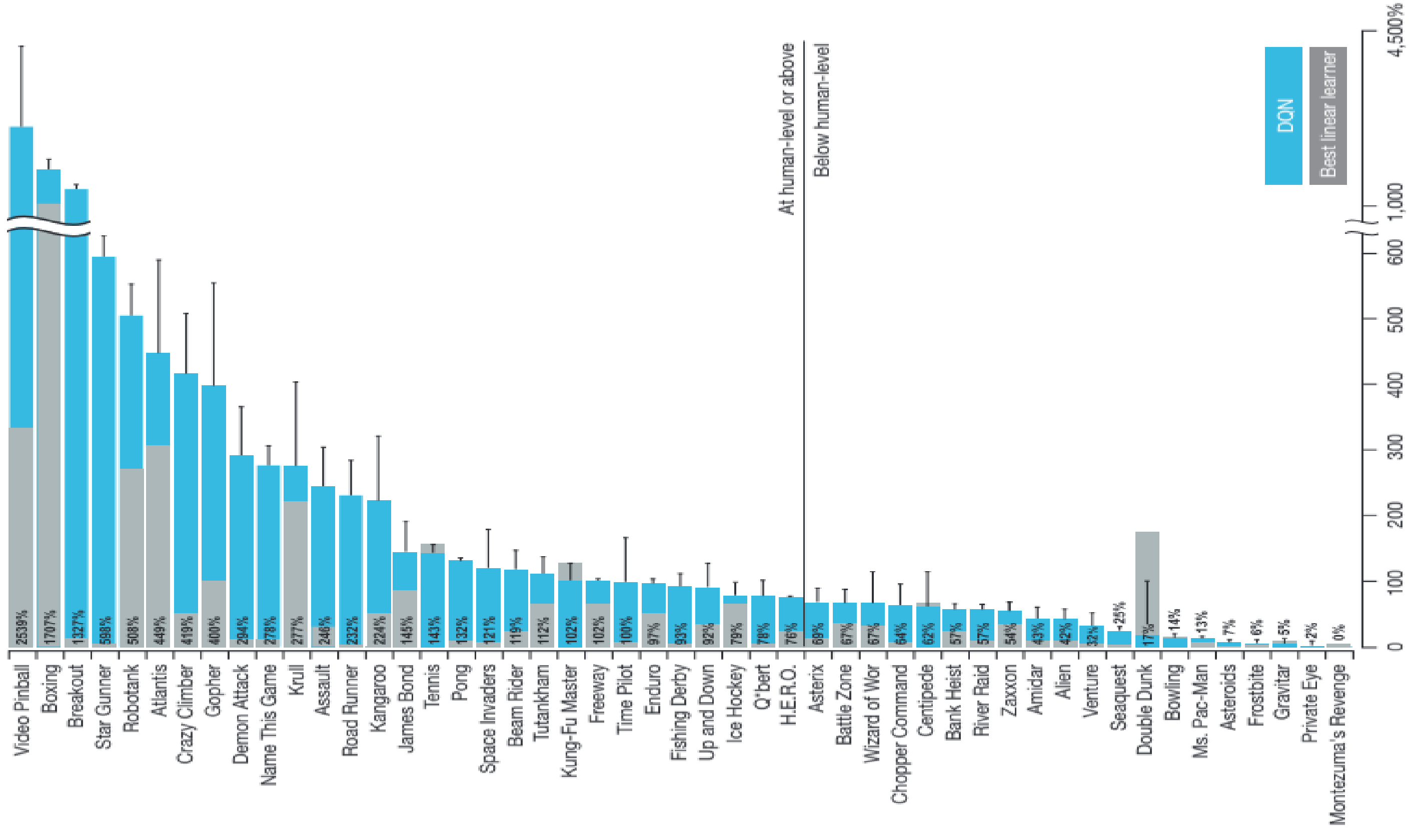
t-SNE of the embedding of the representations in the last hidden layer

- High values for the full (top right) and complete (bottom left) screen, because a complete screen leads to a new full screen
- Partially completed screens (bottom) have lower state values, because less immediate reward is available



- **Deep Q-learning**

  outperforms the best approaches in many tasks



- Deep Double Q-learning

Q-learning $Q^*(s,a) = R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') \max_{a'} Q^*(s',a')$

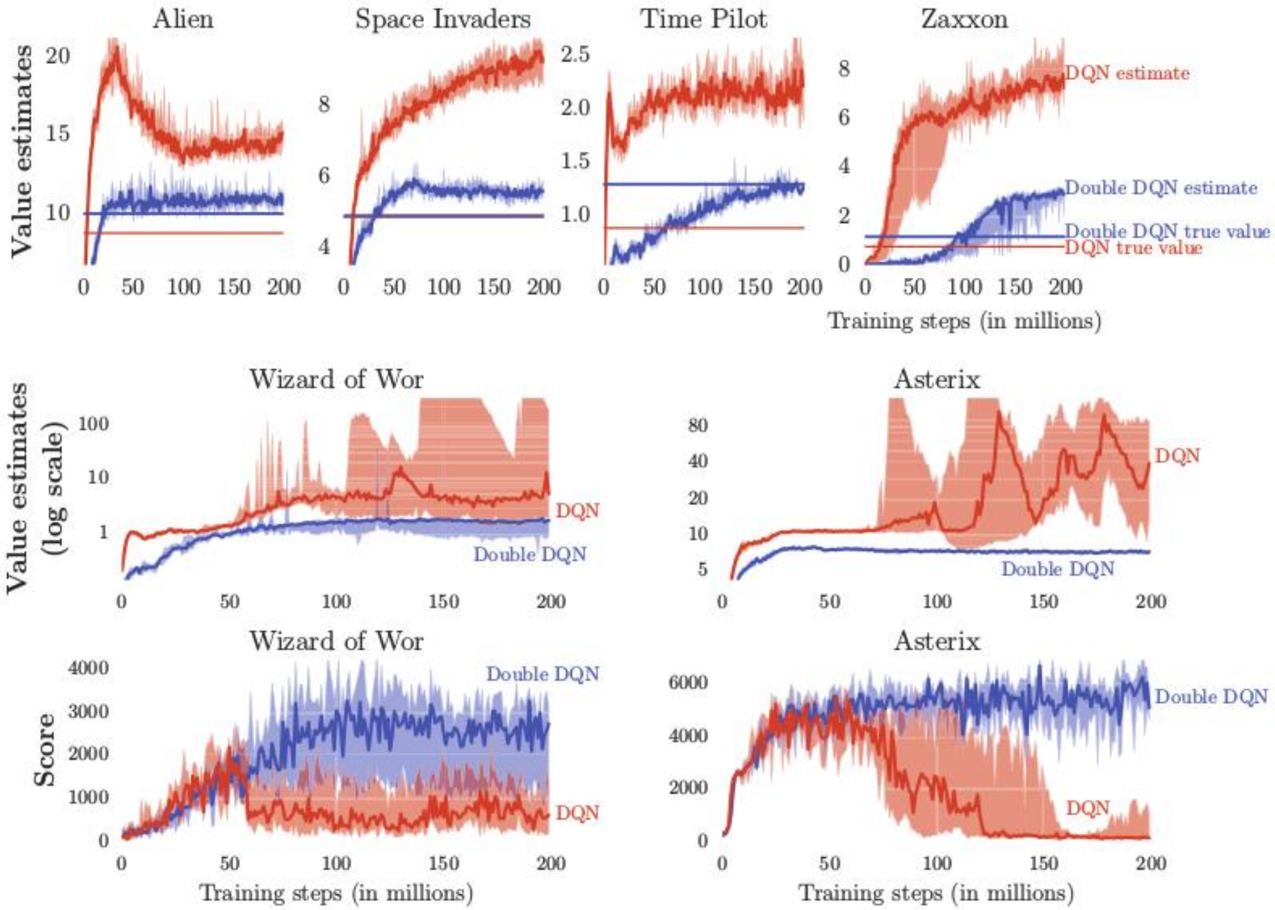
Double Q-learning $Q^*(s,a) = R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') \max_{a'} Q'(s',a')$

- Q-learning estimates the value of the greedy policy according to the current values
- Double Q-learning uses a second set of weights to fairly evaluate of the policy

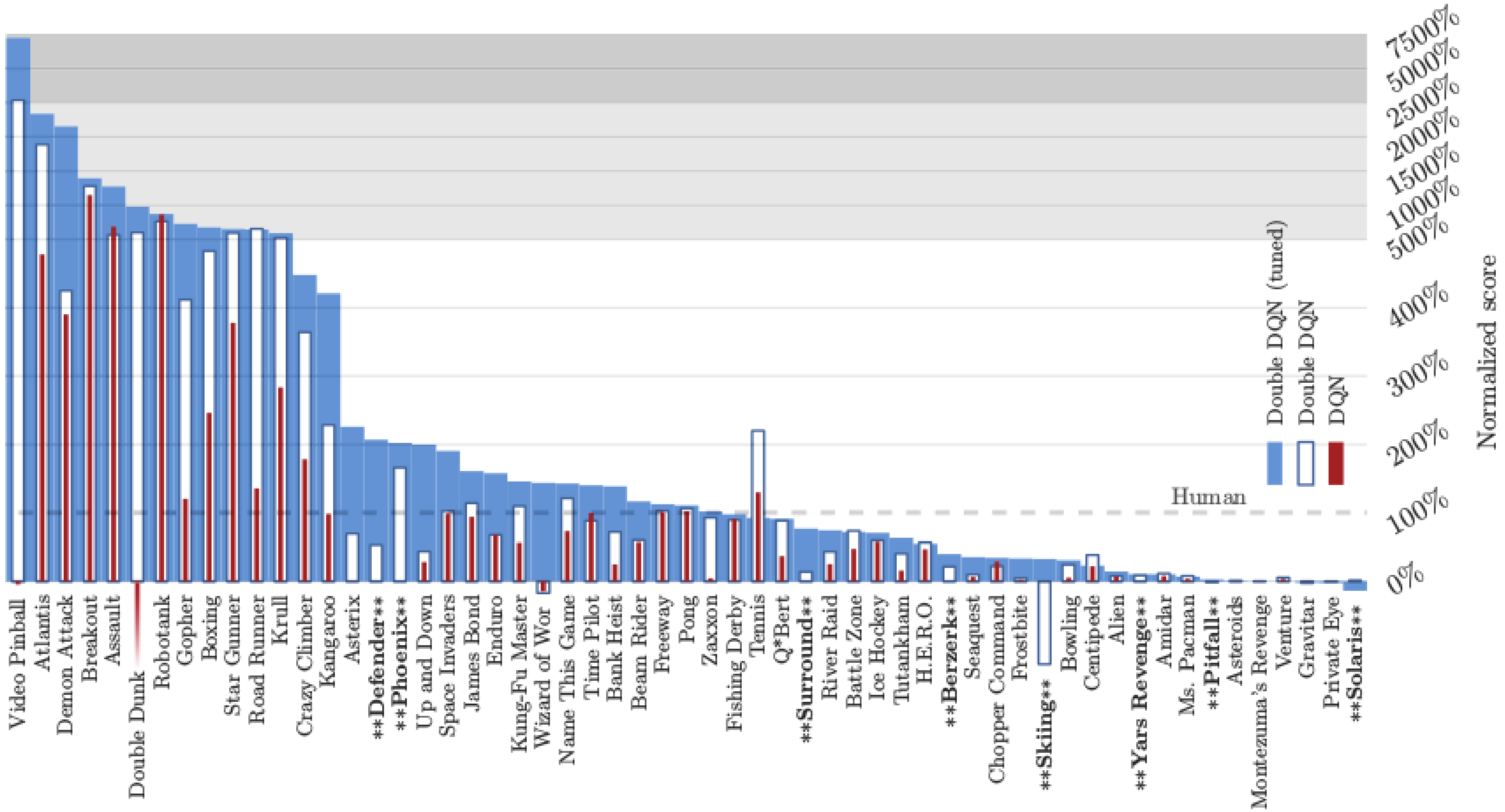
reference: Van Hasselt, H., Guez, A. and Silver, D., 2016, March. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*.

Deep Double Q-learning

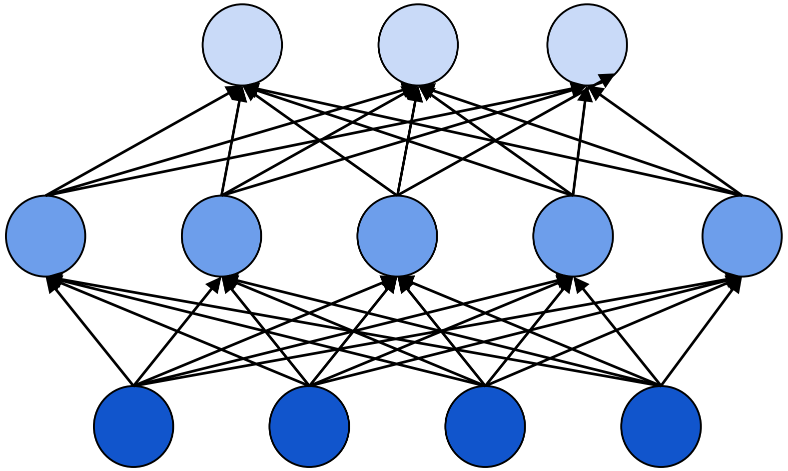
- The middle row shows the value estimates (in log scale) for two games in which DQN's overoptimism is quite extreme.
- The bottom row shows the detrimental effect of this on the score achieved by the agent as it is evaluated during training: the scores drop when the overestimations begin. Learning with Double DQN is much more stable.



- Deep Double Q-learning



- Evolving Learning



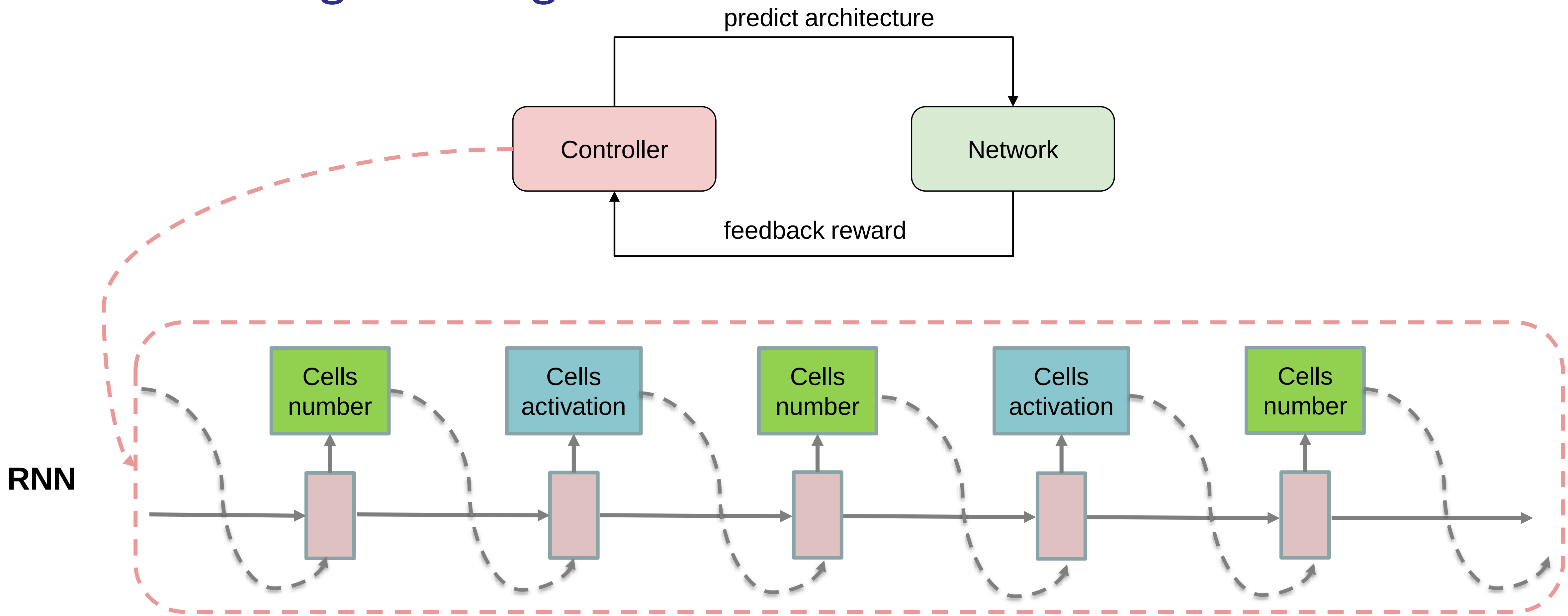
How to select deep learning architectures, and hyper-parameters ?

Brute-force search
- require high experiences
- time-consuming

Is it possible to use reinforcement learning ?
- Evolving learning

reference: Zhang, Z., Han, J., Qian, K. and Schuller, B.W., 2018. Evolving Learning for Analysing Mood-Related Infant Vocalisation. In *Interspeech* (pp. 142-146).

- Evolving Learning



- **Evolving Algorithms**

  - Example: RNN for classification problem
    - Goal: Learn hyperparameters for each layer i
      - Number of hidden nodes h_i
      - Activation function φ_i
    - Given set of hyper-parameters $\tau = \{h_1, \varphi_1, h_2, \varphi_2, \dots\}$
      - Create child-network specified by hyperparameters
      - Train child-network on classification task
      - Obtain reward: Accuracy on development set R

Evolving Algorithms

- Example: RNN for classification problem
  - Probability for child network τ: $p(\tau|\theta)$
    - Depends on Parameter of Controller Network θ
  - For optimal architecture of Controller Network
    - Maximize expected reward

$$J(\theta) = \sum_{\tau} R(\tau)p(\tau|\theta)$$

    - Update θ with gradient $\nabla_{\theta} J(\theta)$

- Evolving Learning

prediction accuracy — controller parameters

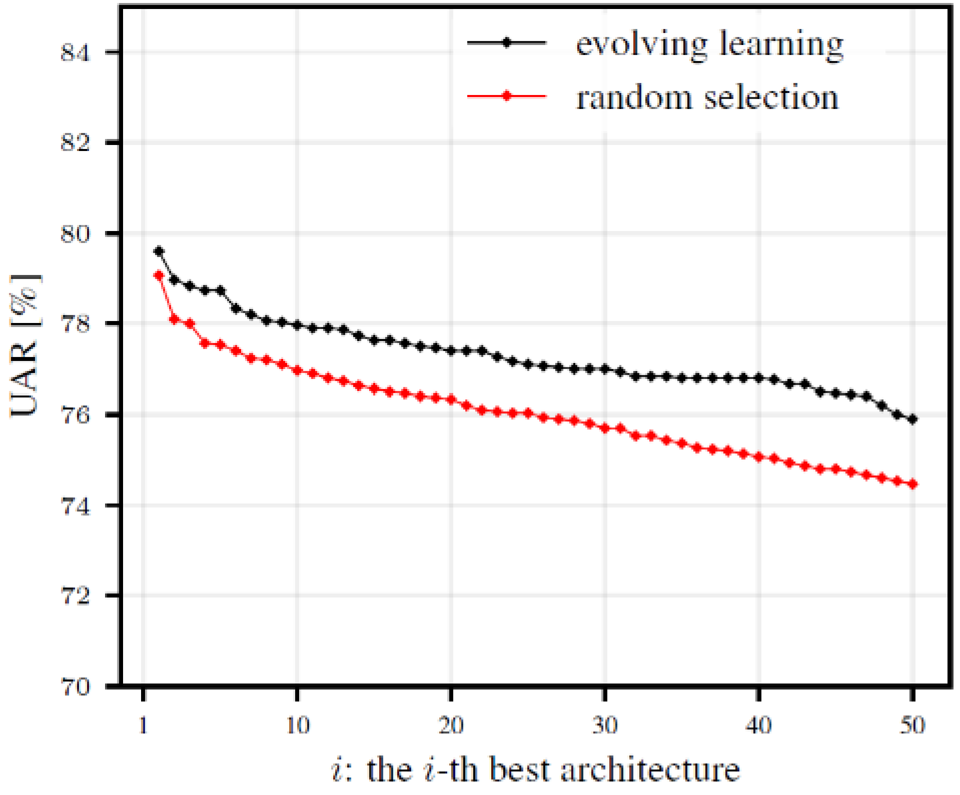
$$\text{reward} \quad J(\theta) = \sum_{\tau} \boxed{R(\tau)}\,\boxed{p}\,(\boxed{\tau}\,|\,\boxed{\theta})$$

probability — network architecture

Search space of hyper-parameters

Types	Hyper-parameters
# layers	1, 2, 3, 4, 5
# node per layer	0, 40, 80, 120, 160, 200
Activation functions	Tanh, ReLU, sigmoid

- Evolving Learning vs other approaches



approaches	UAR[%]
Seq2seq [33]	62.1
End-to-end [34]	63.5
BoAW [35]	67.7
ComParE [28]	71.9
Evolved GRU-RNN	**70.1**

Conclusions

Reinforcement learning

- Model-based approach
  - Markov decision process
- Model-free approach
  - **Q-learning**
  - Adaptive heuristic critic
  - Model-free learning with average reward

Deep reinforcement learning

- Deep Q-learning, deep double Q-learning
- Evolving learning