Reinforcement Learning

—

Exercise 6: Value Function Approximation

Nico Meyer

Fraunhofer IIS

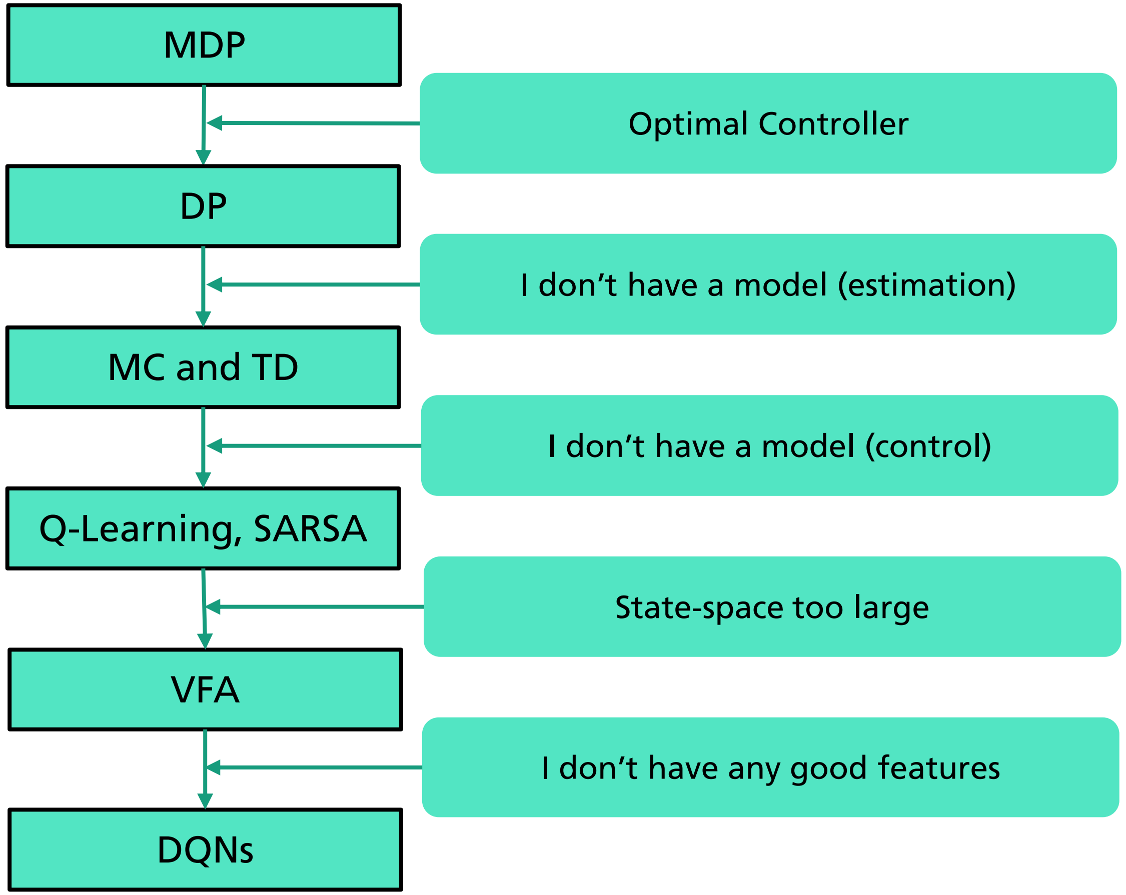
Fraunhofer-Institut für Integrierte Schaltungen IIS

Overview

Exercise Content

Week	*Date*	*Topic*	*Material*	*Who?*
0			*no exercises*	
1	23.04.	MDPs		Nico
2	30.04.	Dynamic Programming		Alex
3	07.05.	OpenAI Gym, PyTorch-Intro		Alex
4	14.05.	TD-Learning		Nico
5	22.05.	Practical Session (zoom@home)	**Attention: Lecture Slot!**	Nico + Alex
6	28.05.	TD-Control		Nico
7	04.06.	DQN		Nico
8	11.06.	VPG		Alex
9	18.06.	A2C		Nico
10	25.06.	Multi-armed Bandits		Alex
11	02.07.	RND/ICM		Alex
12	09.07.	MCTS		Alex
13	16.07.	BCQ		Nico

Overview
Overall Picture



Value Function Approximation

Deep Q-Networks

Value Function Approximation

Why it is necessary and how it can be done

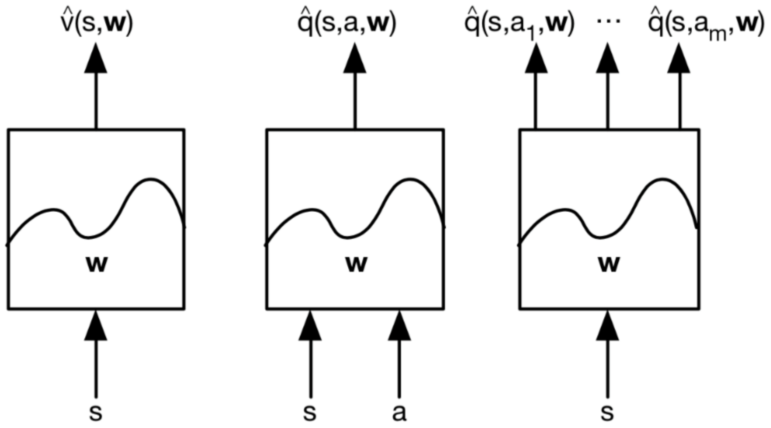
- Challenge: In real world problems, the state space can be large
  - Backgammon: 10^{20} states
  - Computer Go: 10^{170} states
  - Robot arm: **infinite** number of states! (continuous)

- Exact:
  - A table with a distinct value for each case
- Approximate:
  - Approximate V or Q with a function approximator (e.g., NN, polynomials, RBF, …)

$$\hat{v}(s, \boldsymbol{w}) \approx v_\pi(s)$$
$$\hat{q}(s, a, \boldsymbol{w}) \approx q_\pi(s, a)$$

+ We only need to store the approximator parameters

- Convergence properties do not hold anymore

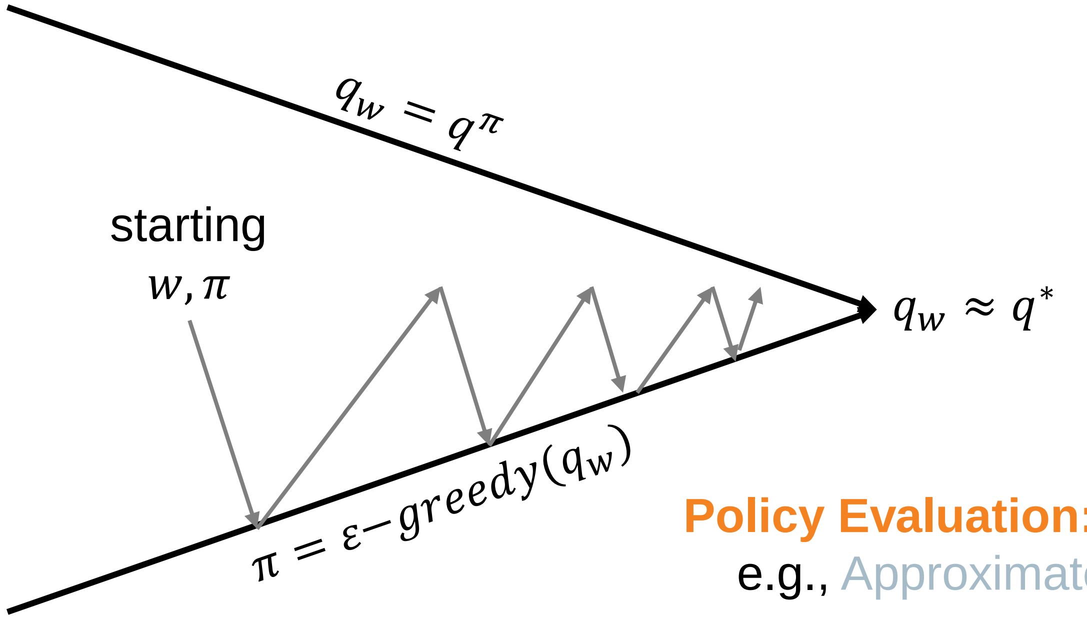


David Silver. 2016.

Value Function Approximation

Policy Evaluation and Improvement

- Our goal is to learn good parameters w that approximate the true value function well:



Policy Evaluation: Estimate $\hat{\boldsymbol{q}}(\cdot,\cdot,\boldsymbol{w}) \approx q_\pi$
e.g., Approximate Policy Evaluation

Policy Improvement: Generate $\pi' \geq \pi$
e.g., ϵ-greedy Policy Improvement

Fraunhofer IIS

Value Function Approximation

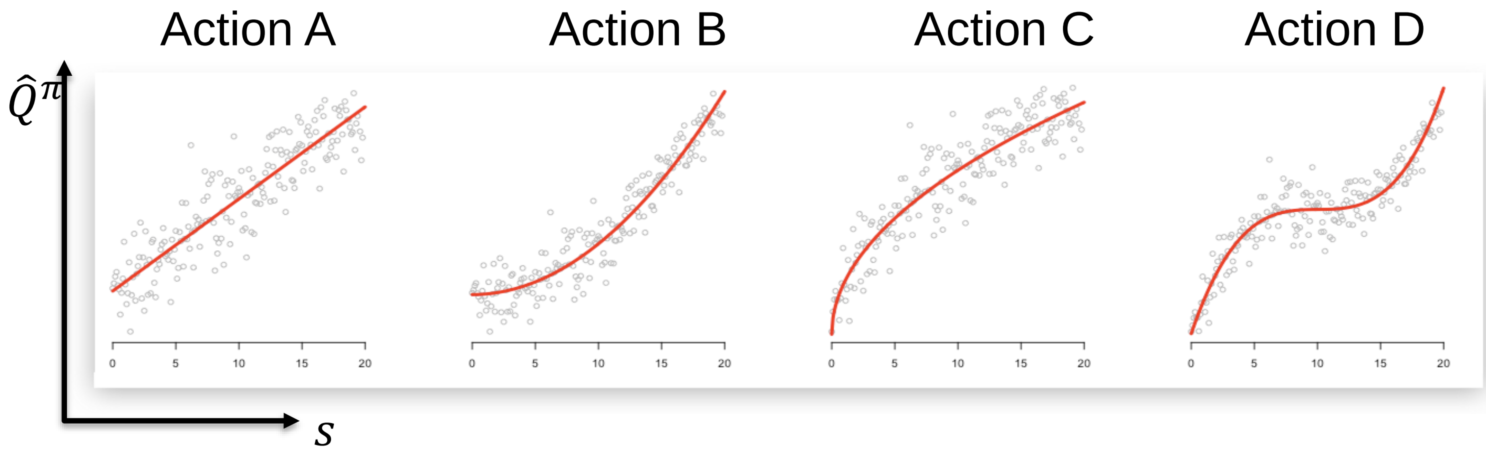
Linear Approximation

- Linear Value Function Approximation (careful: non-linear features)

$$\hat{Q}^{\pi}(s,a;w) = \phi(s,a)^T w$$

- Example features: Polynomial Basis, for instance:

$$(s_1,s_2)^T \rightarrow (1,s_1,s_2,s_1 s_2)^T$$
$$(s_1,s_2)^T \rightarrow (1,s_1,s_2\,,s_1 s_2,s_1^2,s_2^2,s_1 s_2^2,s_1^2 s_2,s_1^2 s_2^2)$$



Value Function Approximation

Convergence Guarantees

- Idea: Why don‘t we replace linear approximation with NNs?
  - Because theory tells us that this doesn‘t work out

Algorithm	Table Lookup	Linear	Non-linear
Monte-Carlo Control	✓	(✓)	✗
SARSA	✓	(✓)	✗
Q-learning	✓	✗	✗

(✓) =chatters around near-optimal value function

Besides some few hand-crafted and tuned successes NNs have not been managed to be applied “as is” to RL -> at least till 2014!!

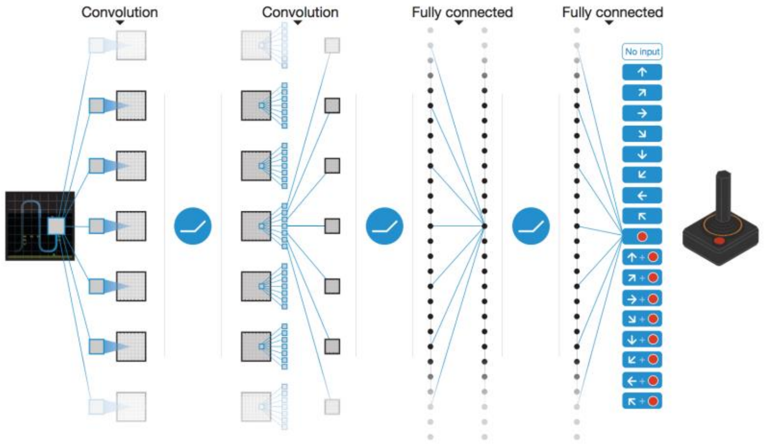
Fraunhofer IIS

Value Function Approximation

Deep Q-Networks (DQNs)

- Then a game-changing result was published in Nature:
  DQN from DeepMind (now Google DeepMind)

- How does it work?
  - A convolutional neural network reads the image from the game (i.e., a framestack that uses the last $N = 4$ frames).
  - The CNN is a value function approximator for the $Q(s, a)$ function.
  - The reward is the game score.
  - The network weights are tuned using backpropagation signals of the rewards.

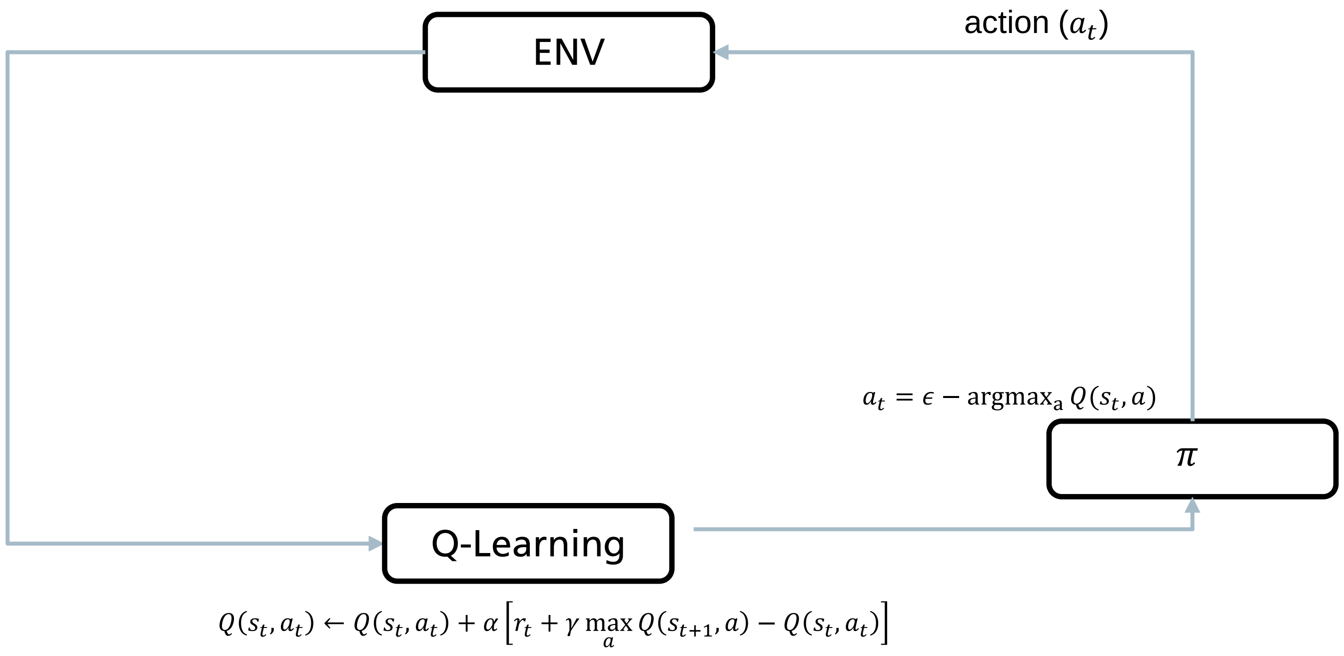


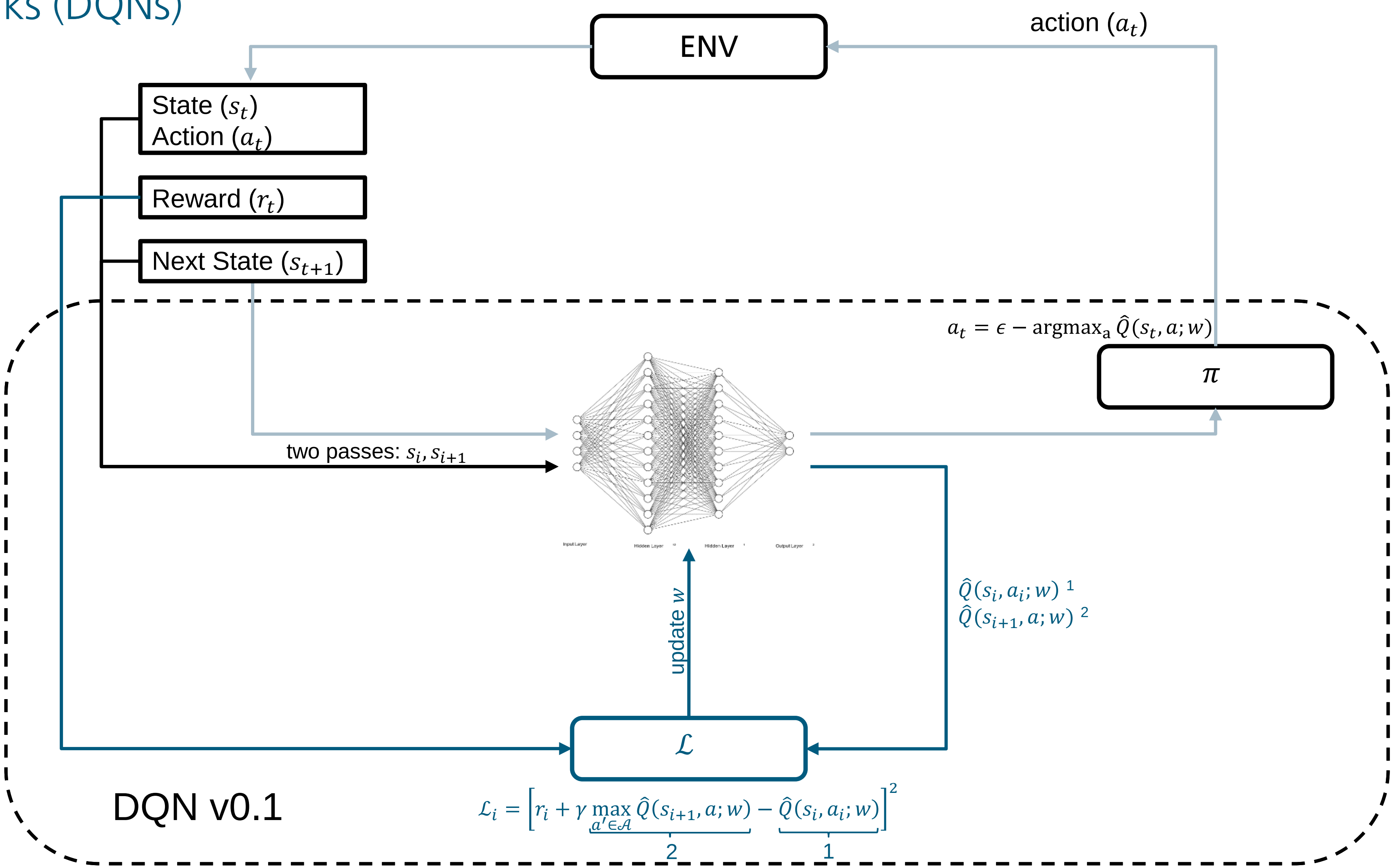
https://storage.googleapis.com/deepmind-media/dqn/DQNNaturePaper.pdf

- DQNs are „Q-Learning on steroids" (Deep NN as VFA)
- Training possible in Tensorflow (or Pytorch, Keras, …)
- Objective function for gradient descent:

$$L(w_i) = \mathbb{E}_{s,a,r,s' \sim D_i}[(y_i - Q(s,a,w))^2]$$

Value Function Approximation
Deep Q-Networks (DQNs)

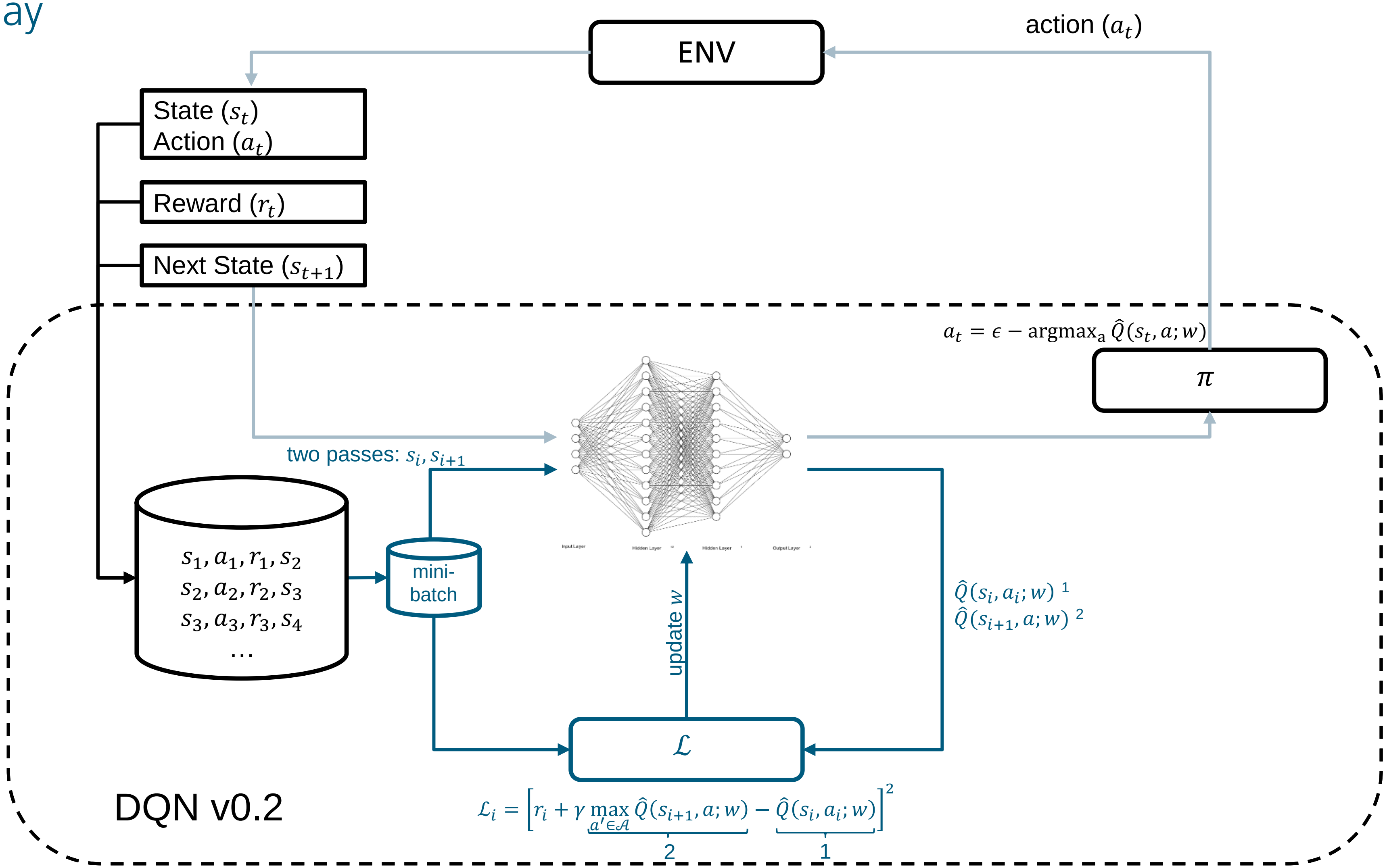




Fraunhofer IIS

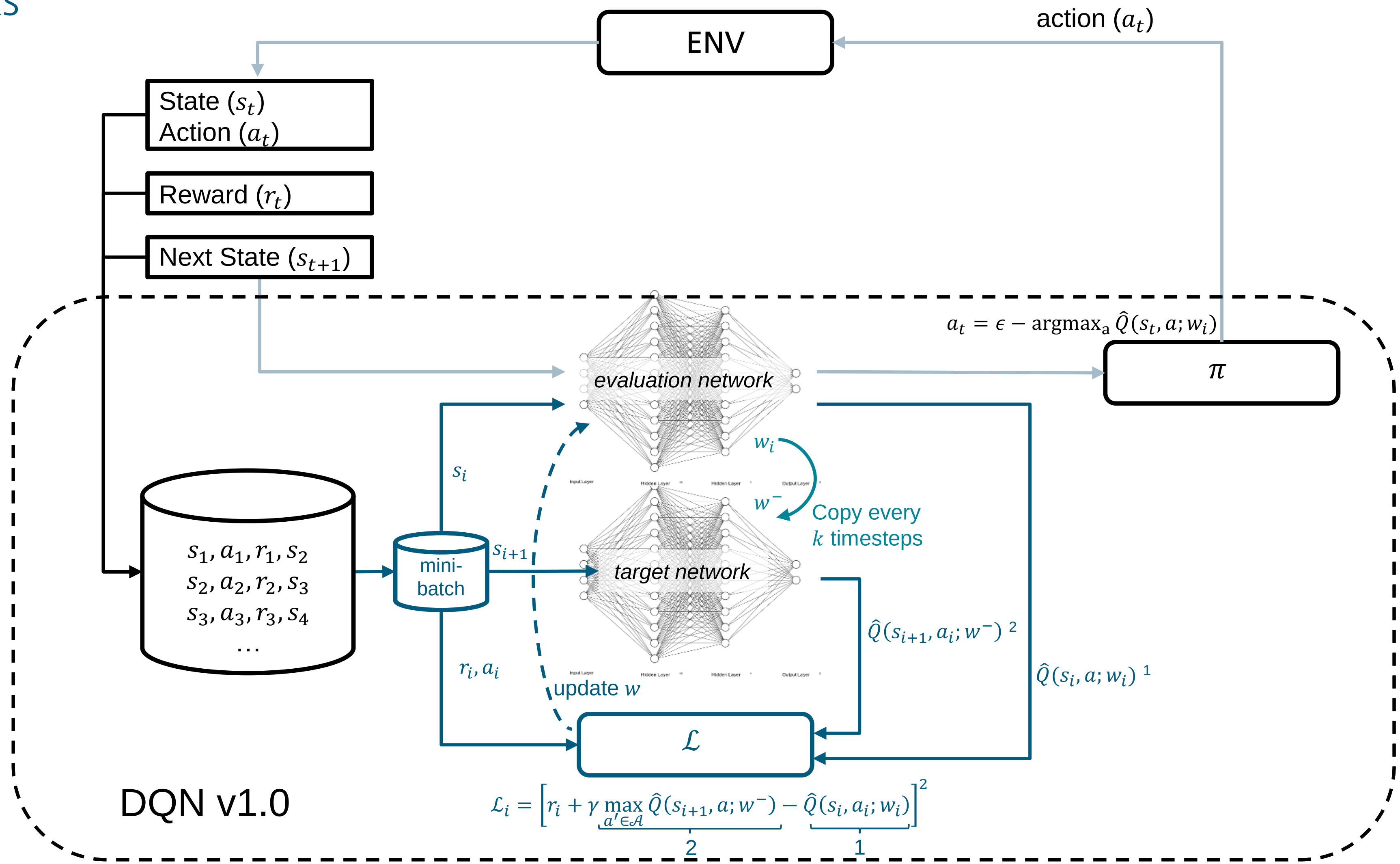
Value Function Approximation

Experience Replay



Value Function Approximation

Target Networks



Value Function Approximation

DQN Algorithm

Algorithm 1: deep Q-learning with experience replay.

Initialize replay memory D to capacity N

Initialize action-value function Q with random weights θ

Initialize target action-value function $\hat{Q}$ with weights $\theta^- = \theta$

For episode $= 1, M$ **do**

  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$

  For $t = 1$,T **do**

    With probability ε select a random action a_t

    otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$

    Execute action a_t in emulator and observe reward r_t and image x_{t+1}

    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$

    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D

    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D

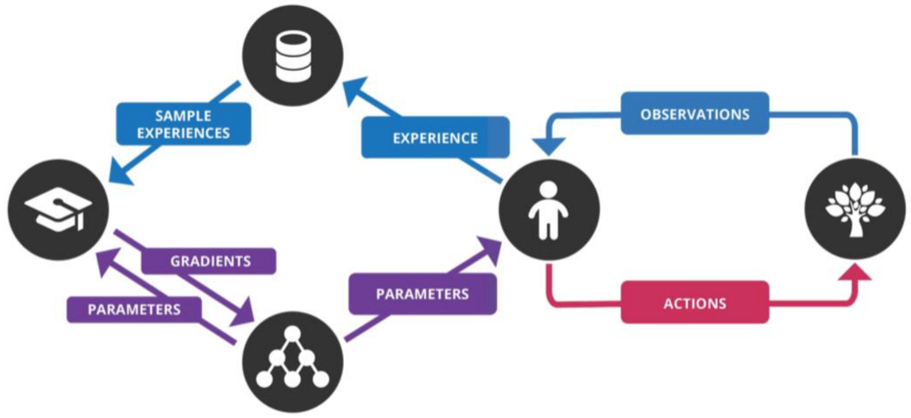
    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^-\right) & \text{otherwise} \end{cases}$

    Perform a gradient descent step on $\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$ with respect to the network parameters θ

    Every C steps reset $\hat{Q} = Q$

  End For

End For



https://sites.google.com/view/deep-rl-bootcamp/lectures

Fraunhofer IIS

Exercise Sheet 6

Deep Q-Networks

Thank you for your attention!

Fraunhofer IIS

Fraunhofer-Institut für Integrierte Schaltungen IIS