

Verteiltes Deep Reinforcement Learning System zum Trainieren von Game AI

- Praktikum 07.05.2021 -

[Aufgabe 1: ML-Agents Python Low Level API](#)

[Aufgabe 2: Training mit fremder/eigener Implementierung](#)

[Aufgabe 3: Unit & Integration Tests](#)

Ganz Wichtig: Arbeitet euch durch die Dokumentation!

[ML-Agents Toolkit Release 15 Dokumentation](#)

Aufgabe 1: ML-Agents Python Low Level API

- <https://github.com/Unity-Technologies/ml-agents/blob/65c1550cfaee89c980a7b9f722e8925363507834/docs/Python-API.md>
- Nutzt die Umwelt vom letzten Praktikumstag oder wählt eine neue Umwelt nach den bekannten Kriterien aus
- Erstellt einen Build der Umwelt, welche über nur **einen einzigen** Agenten verfügt!
- Erstellt mit Hilfe der Low Level API eine Klasse in welche die ausgewählte Umwelt als gym environment gewrapped wird
- Testet Eure Umsetzung in dem ihr eine Episode mit Zufallsaktionen spielen lasst

Aufgabe 2: Training mit fremder/eigener Implementierung

- Wählt eine fremde oder eigene Implementierung eines DRL Algorithmuses (DQN, A2C, PPO, SAC) aus
- Lasst mittels diesem Algorithmus eure Umwelt trainieren
 - ggf. müssen Hyperparameter und die Modellarchitektur angepasst werden

- Mögliche fremde Implementierungen (die großen Frameworks eignen sich nicht unbedingt für Aufgabe 3)
 - [Varuna PPO](#)
 - [Machine Learning with Phil SAC](#)
 - [Machine Learning with Phil DQN](#)
 - [ikostrikov PPO, A2C](#)
 - [PyTorch DQN](#)
 - [Deep Reinforcement Learning Hands-On](#)
 - PPO, A2C, DQN, ...
 - [Grokking Deep Reinforcement Learning](#)
 - PPO, A2C, DQN, SAC, ...
 - [RL Adventure](#)
 - DQN
 - [RL Adventure 2](#)
 - PPO, A2C, SAC, ...
- *DQN u.a. auch mit Extensions

Aufgabe 3: Unit & Integration Tests

- Implementiert Unit Tests für z.B.
 - cumulative discounted rewards
 - output size convolutional layers
 - annealing training parameters (learning rate, exploration, ...)
 - generalized advantage estimation
 - ...
- Implementiert eine Testumwelt als Integration Test