
VINCENT BOHM

# Train RL Mario AGENT

Playing Super Mario Land

# Einleitung

## Aufgabe

Wählen eines AI Modells aus den behandelten Lerneinheiten.

## Ziel

Ziel ist es, einen intelligenten Agenten zu schaffen, der durch das Spiel navigiert, Hindernisse überwindet und das Level schnellstmöglich abschließt.

## Projektwahl

RL-Modell Entwickeln auf dem PyBoy-Emulator für "Super Mario Land" erstellen.

## Lernumgebung

"Super Mario Land" wegen seiner komplexen, variantenreichen Levelgestaltung und Linearität.

# Double Deep Q-Network-Architektur (DDQN)

## RL

Agent lernt durch (Trial and Error) nach optimale Policy Prinzip ein Verhaltensstrategie zu entwickeln.

## DQL

Verbindet Q-Learning mit neuronalen Netzwerken um komplexe Probleme in umfangreichen Zustandsräumen zu lösen.

Durch das Erlernen einer Q-Funktion, die den erwarteten Wert einer Aktion, in einem bestimmten Zustand darstellt, können DQNs in anspruchsvollen Umgebungen agieren

## DDQN

Basieren auf DQNs und zielen darauf ab, ein zentrales Problem – die systematische Überschätzung von Q-Werten – zu lösen.

Durch Trennung zwischen der Auswahl (Online-Netzwerk) und der Bewertung (Ziel-Netzwerk) von Aktionen minimieren DDQNs das Risiko einer Überbewertung.

Stabilisiert Lernprozess und beschleunigt die Konvergenz.

# Lernumgebung

## PyBoy

- Python basierter Emulator
- Schnittstelle zum Spiel
- Ermöglicht Auslesen und verändern von Internen werten
- Direkter Zugriff auf das Spiel

## Preprocessing-Techniken

- Frame Skipping
- Graustufenkonvertierung
- Resize Observation
- Frame Stacking

# Technik & Methode

## Reward Funktion

- Zeit
- Bewegung
- Tod -
- Level Belohnung +

## Hyperparameter

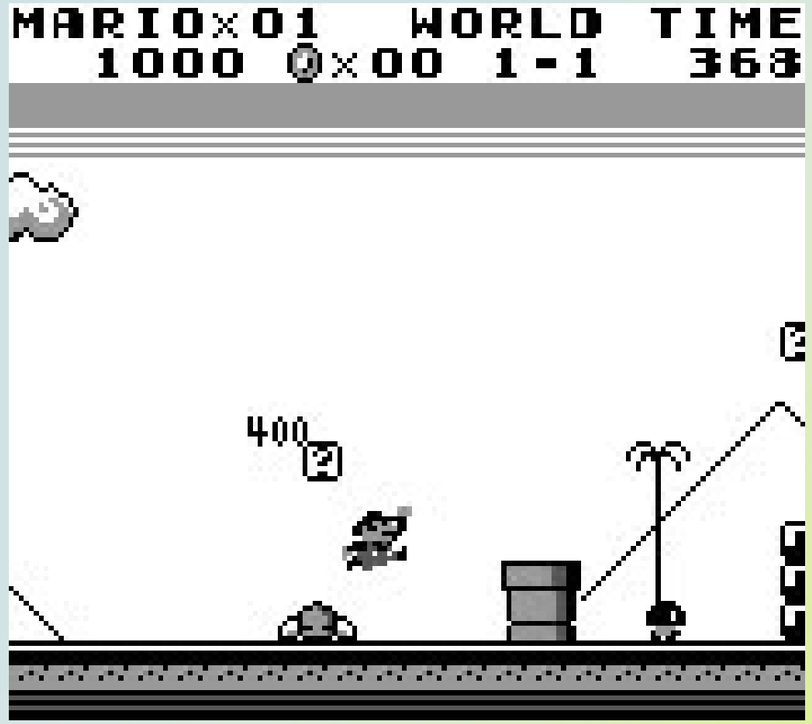
- Exploration 1
- Exp. Decay 0.99999975
- Exp Min 0.1
- Learning_rate 0.00025
- Batch 32
- Gamma 0.9
- Sync 10.000

## Aktionsraum

- Rechts
- Links
- Springen
- 2er Kombination

## Netzwerkarchitektur

- 3 Convolutional-Layer
- 2 Fully Connected-Layer.



# RL MARIO NET

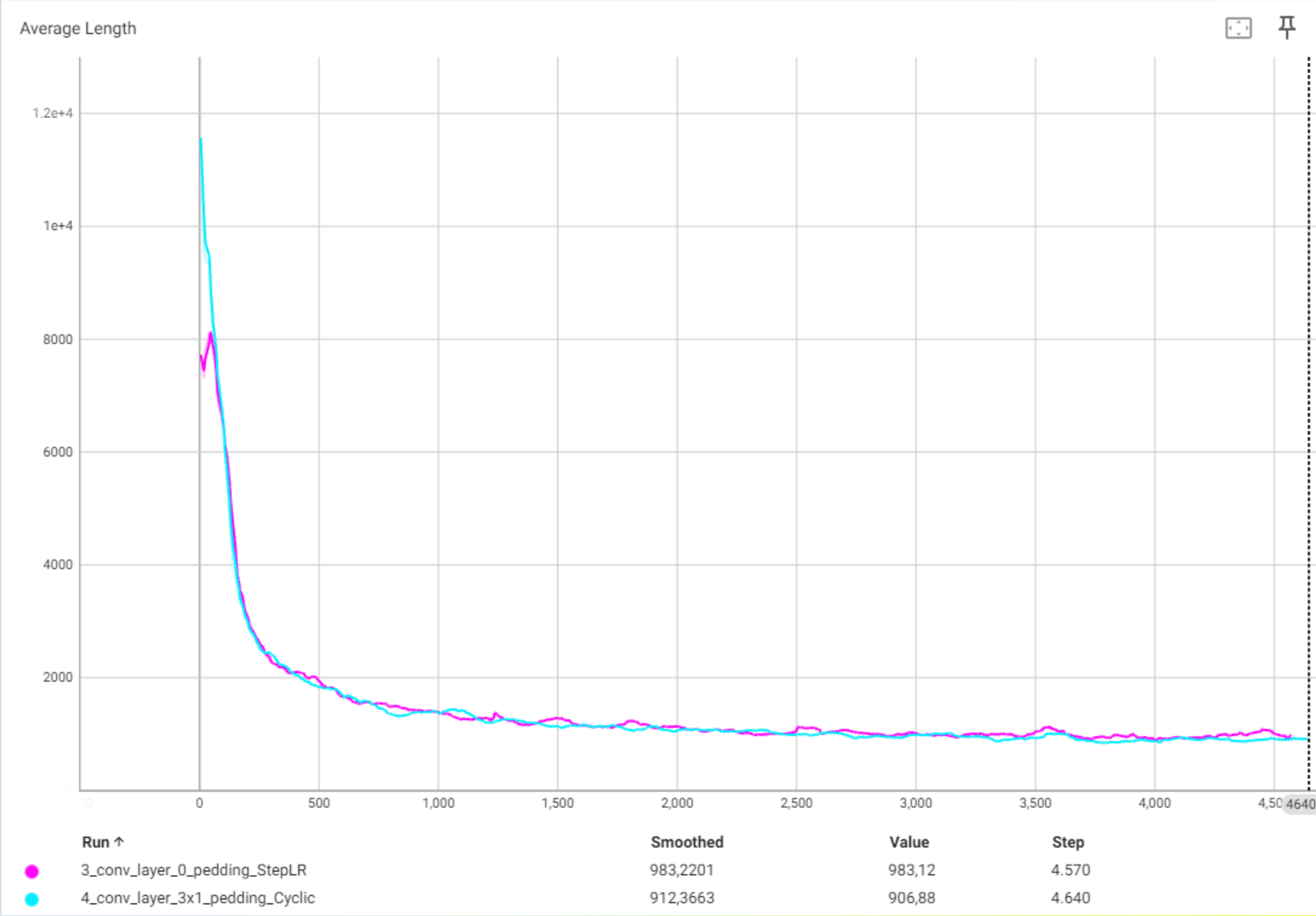
Vincent Bohm WS 2023/24

# Experimente

## Experimentelles Setup

- Vergleich der Lern-Performance zweier unterschiedlich großer Netzwerke.

- **(Modell 1)**
  - 3 Conv_layer
  - kein Pedding
  - StepLR
- **(Modell 2)**
  - 4 Conv_layer
  - 3 x 1 Pedding
  - Cyclic

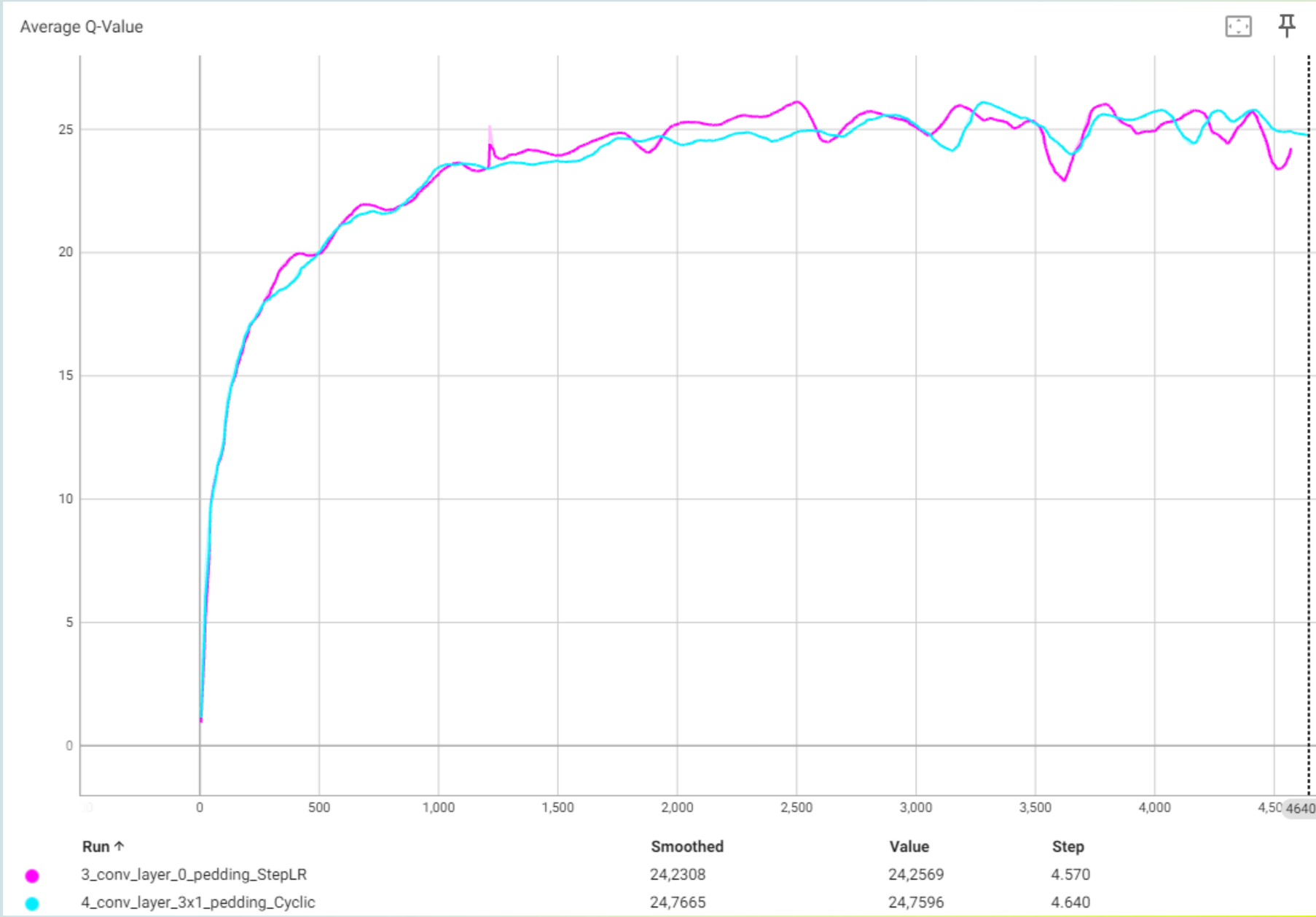


# Experimente

## Experimentelles Setup

- Vergleich der Lern-Performance zweier unterschiedlich großer Netzwerke.

- **(Modell 1)**
  - 3 Conv_layer
  - 0 Pedding
  - StepLR
- **(Modell 2)**
  - 4 Conv_layer
  - 3 x 1 Pedding
  - Cyclic

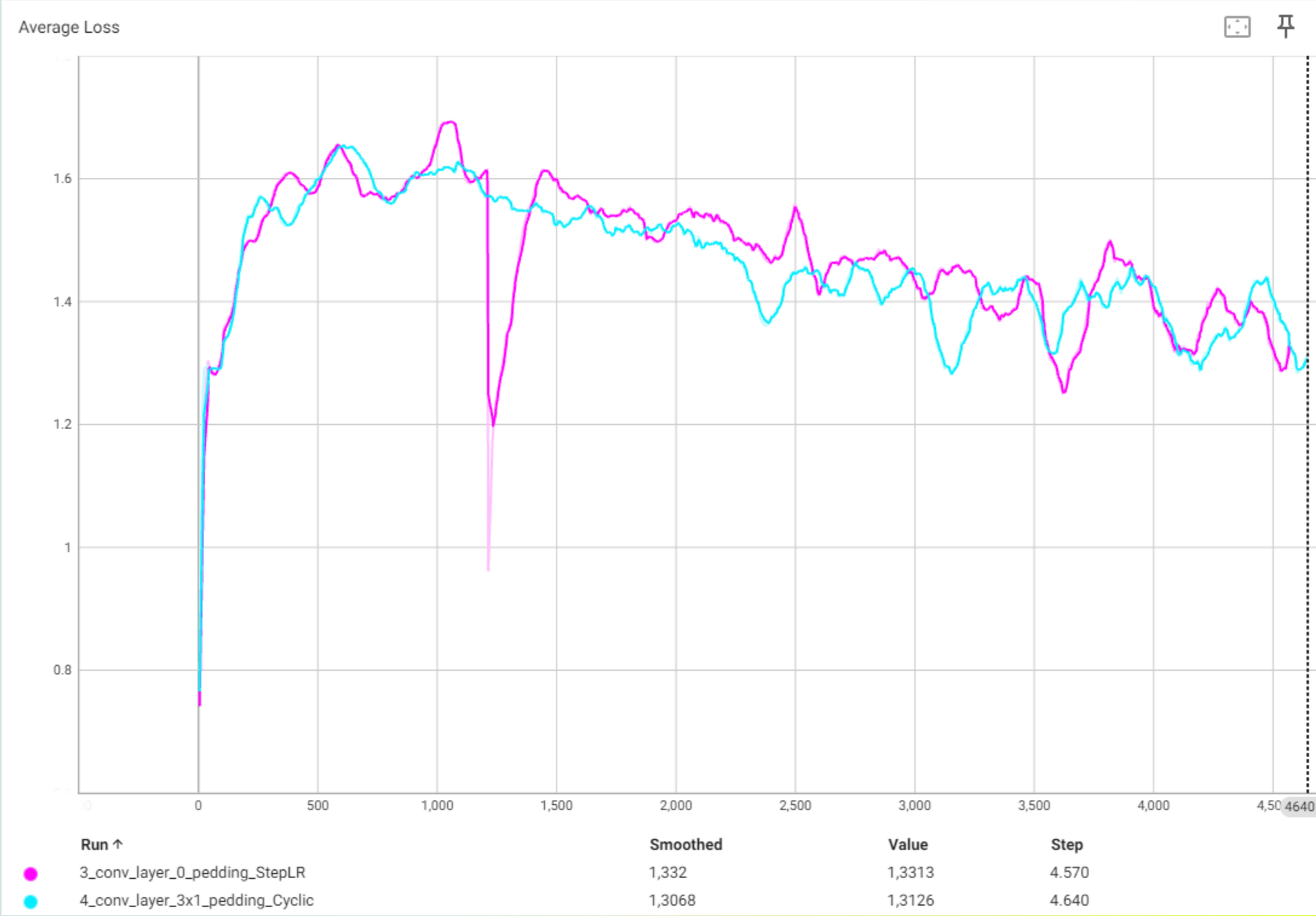


# Experimente

## Experimentelles Setup

- Vergleich der Lern-Performance zweier unterschiedlich großer Netzwerke.

- **(Modell 1)**
  - 3 Conv_layer
  - 0 Pedding
  - StepLR
- **(Modell 2)**
  - 4 Conv_layer
  - 3 x 1 Pedding
  - Cyclic

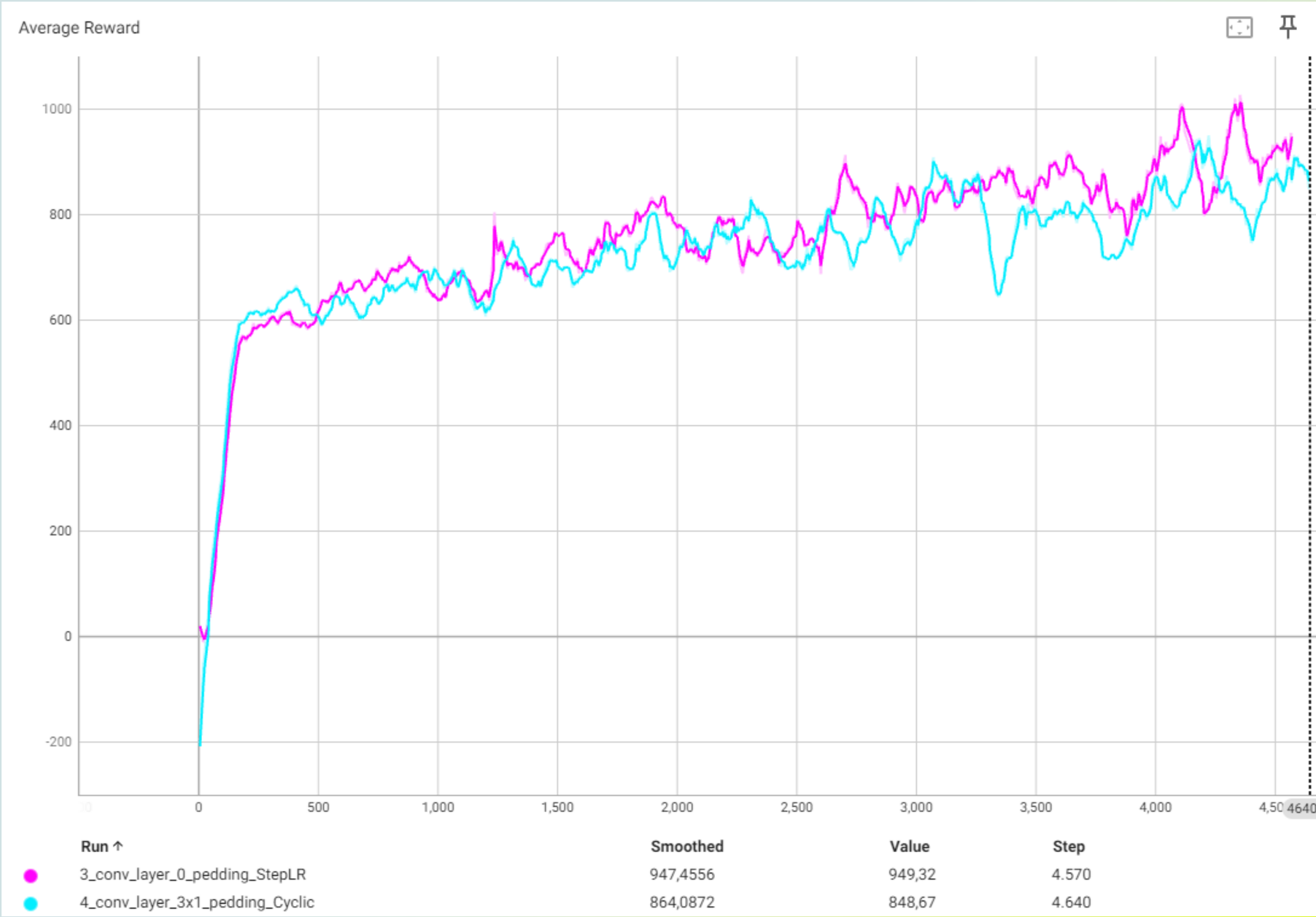


# Experimente

## Experimentelles Setup

- Vergleich der Lern-Performance zweier unterschiedlich großer Netzwerke.

- **(Modell 1)**
  - 3 Conv_layer
  - 0 Pedding
  - StepLR
- **(Modell 2)**
  - 4 Conv_layer
  - 3 x 1 Pedding
  - Cyclic



# Schlussfolgerung und Ausblick

## Zusammenfassung

- Keine deutlichen unterschiede im lernverhalten
- Kleineres Modell Ressourcen sparender

## Verbesserungen und Ausblick

Möglichkeiten zur Weiterentwicklung:

- Training des Agenten über verschiedene Spiellevel
- Erweiterung des Aktionsraums.
- Anpassen der Hyperparameter
- Multiagent Training

# DANKE

## Github Repository:

**l4nz8/q_play**