HAW HAMBURG

Masterarbeit

Alexander Hoffmann

Robuster Reinforcement Learning Algorithmus zur Überbrückung der Sim-to-Real Gap im autonomen Fahren

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department Computer Science

Alexander Hoffmann

Robuster Reinforcement Learning Algorithmus zur Überbrückung der Sim-to-Real Gap im autonomen Fahren

Masterarbeit eingereicht im Rahmen der Masterprüfung
im Studiengang *Master of Science Informatik*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Stephan Pareigis
Zweitgutachter: Prof. Dr. Tim Tiedemann

Eingereicht am: 17. August 2023

Alexander Hoffmann

Thema der Arbeit

Robuster Reinforcement Learning Algorithmus zur Überbrückung der Sim-to-Real Gap im autonomen Fahren

Stichworte

Autonomes Fahren, Reinforcement Learning, Sim-to-Real Gap, Machine Learning, Deep Learning, Robuste Steuerung

Kurzzusammenfassung

Die Diskrepanz zwischen der Realität und einer Simulation stellt die Forschung im Bereich des autonomen Fahrens vor zahlreiche Herausforderungen. Simulationen erfassen nicht alle Aspekte der realen Welt, und das Training in der Simulation kann zu unvorhersehbarem Verhalten in der tatsächlichen Anwendung führen, aufgrund der Unterschiede zwischen der Simulation und der realen Umgebung. In dieser Arbeit wird ein Ansatz vorgestellt, um diese Unterschiede speziell in Bezug auf die Lenkung eines Fahrzeugs zu behandeln. Dieser Ansatz wird bei drei unterschiedlichen Reinforcement Learning Algorithmen angewandt: TD3, PPO und DQN. Die Idee besteht darin, anstatt das Netzwerk mit einem absoluten Lenkwinkel zu trainieren, mit der Differenz zwischen dem gewünschten und dem gegebenen Lenkwinkel zu arbeiten. Dadurch können Abweichungen in der Lenkung korrigiert werden, die durch technische Fehler in einer realen Anwendung verursacht wurden. Dieser Ansatz, der als TD3Δ, PPOΔ und DQNΔ bezeichnet wird, wird mit dem herkömmlichen TD3, PPO und DQN verglichen. Zusätzlich werden die Architekturen beschrieben, die in TD3Δ, PPOΔ und DQNΔ verwendet werden.

Alexander Hoffmann

Title of Thesis

Robust Reinforcement Learning Algorithm to Bridge the Sim2Real Gap in Autonomous Driving

Keywords

Autonomous Driving, Reinforcement Learning, Sim-to-Real Gap, Machine Learning, Deep Learning, Robust Control

Abstract

The discrepancy between reality and simulation presents numerous challenges in the field of autonomous driving research. Simulations do not capture all aspects of the real world, and training in a simulation can lead to unpredictable behavior in real-world applications due to differences between the simulation and the actual environment. This work presents an approach to specifically address these differences in relation to vehicle steering. This approach is applied to three different reinforcement learning algorithms: TD3, PPO, and DQN. The idea is to train the network not with an absolute steering angle but with the difference between the desired and actual steering angles. This allows for correcting steering deviations caused by technical errors in a real-world application. This approach, referred to as TD3Δ, PPOΔ, and DQNΔ, is compared with the conventional TD3, PPO, and DQN methods. Additionally, the architectures used in TD3Δ, PPOΔ, and DQNΔ are described.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungen

CNN Convolutional Neural Network.

CTE Cross Track Error.

DDPG Deep Deterministic Policy Gradient.

DQN Deep Q-Network.

E2E End-to-End.

FPS Frames per Second.

GAE Generalized Advantage Estimation.

GCP Google Cloud Platform.

KI Künstliche Intelligenz.

MDP Markov Decision Process.

PPO Proximal Policy Optimization.

RL Reinforcement Learning.

ROI Region of Interest.

SB3 Stable Baselines3.

TD3 Twin Delayed Deep Deterministic Policy Gradient.

1 Einleitung

1.1 Motivation

Die Motivation für diese Masterarbeit besteht darin, die Diskrepanz zwischen dem Training in einer Simulationsumgebung und der Anwendung in einer realen Umgebung mithilfe eines Reinforcement Learning (RL) Algorithmus zu reduzieren. Hierfür wird ein Ansatz gewählt, der das Sim-to-Real Gap bezüglich der autonomen Fahrzeugsteuerung, insbesondere der Lenkung, überbrücken soll. Die Arbeit von Fynn Luca Maaß und Stephan Pareigis aus dem Jahr 2022 dient als Inspiration für diese Arbeit. In ihrer Arbeit wurde eine neue Architektur namens PilotNetΔ für autonomes Fahren vorgestellt, die robust gegenüber der Diskrepanz zwischen Training und Anwendung ist [1]. Die Architektur basiert auf dem End-to-End (E2E)-Ansatz von NVIDIAs PilotNet [2] und soll das Problem des Sim-to-Real Gaps lösen. Im Gegensatz zu PilotNet ist PilotNetΔ in der Lage, Abweichungen in der Lenkung, die durch einen Offset verursacht wurden, zu korrigieren. In beiden Architekturen wird überwachtes Lernen verwendet, bei dem das Netzwerk anhand gesammelter Bilder und den entsprechenden Soll-Lenkwinkeln trainiert wird. In dieser Masterarbeit hingegen wird die RL Architektur verwendet, bei der ein Agent anhand von Versuch-und-Irrtum und den daraus resultierenden Belohnungen eine Policy erlernt.

1.2 Ziel der Arbeit

Das Ziel dieser Masterarbeit ist es, bei drei verschiedenen RL Algorithmen, nämlich Twin Delayed Deep Deterministic Policy Gradient (TD3), Proximal Policy Optimization (PPO) und Deep Q-Network (DQN), einen Ansatz auszuwählen, der das Sim-to-Real Gap überbrückt. Dabei sollen diese Algorithmen sowohl mit ihren jeweiligen Versionen ohne den Ansatz als auch untereinander verglichen werden. Es soll aufgezeigt werden,

dass der gewählte Ansatz theoretisch in der Lage ist, den Unterschied in der Lenksteuerung zwischen der Simulation und der realen Umgebung zu umgehen. Die angewandte Architektur bei all diesen Algorithmen soll beschrieben werden.

1.3 Struktur der Arbeit

Diese Arbeit ist wie folgt aufgebaut: Kapitel 2 behandelt die Grundlagen, in dem die einzelnen Algorithmen grob beschrieben werden, was RL allgemein ist und was der Sim-to-Real Gap beschreibt. In Kapitel 3 wird die gemeinsame Architektur der gewählten RL Algorithmen beschrieben. Die Kapitel 4 bis 6 beschreiben die Algorithmen einzeln. Die Ergebnisse des Trainings der einzelnen Algorithmen werden im Kapitel 7 vorgestellt und interpretiert. Abschließend folgt in Kapitel 8 das Fazit.

2 Grundlagen

Dieses Kapitel widmet sich der Beschreibung von Reinforcement Learning, den Grundlagen der einzelnen RL Algorithmen TD3, PPO und DQN, sowie der Erklärung des Sim-to-Real Gap Problems.

2.1 Reinforcement Learning

RL ist ein Verfahren des maschinellen Lernens, um das Markov Decision Process (MDP) [3] zu lösen. Ein MDP kann als ein 4-Tupel (S, A, P, R) beschrieben werden. Die Menge der Zustände ist S, und A ist eine Menge von Aktionen. P ist die Übergangswahrscheinlichkeit und wird wie folgt beschrieben: $P : S \times A \times S \to [0, 1]$. Sie gibt an, mit welcher Wahrscheinlichkeit eine Aktion in einem Zustand in einen anderen Zustand übergeht. R ist die Belohnungsfunktion: $R : S \times A \times S \to \mathbb{R}$. RL arbeitet nach dem Versuch-und-Irrtum-Prinzip. Es führt Aktionen in einer Umgebung aus und sammelt Daten über die Umgebung, wie den Zustand, die durchgeführte Aktion und die Belohnung. Dann versucht es seine *Policy* π, also seine Strategie, welche Aktionen in welchem Zustand ausgeführt werden sollen, so zu optimieren, dass die Summe aller Belohnungen maximiert wird. Für dieses Prinzip gibt es viele verschiedene Algorithmen, wie zum Beispiel DQN, PPO oder TD3. Diese drei werden in dieser Arbeit verwendet und beschrieben.

2.2 DQN

DQN ist ein Deep RL Algorithmus, der erstmals 2013 von DeepMind, einem Unternehmen für Künstliche Intelligenz (KI), vorgestellt wurde [4]. Er basiert auf Q-Learning [5], nutzt jedoch künstliche neuronale Netzwerke, speziell ein Q-Netzwerk. Q-Learning erlernt eine Q-Funktion, die jeder Aktion in einem bestimmten Zustand einen Wert zuweist. Dieser Wert bestimmt, wie gut die Aktion in diesem Zustand ist, und legt die Strategie des

Algorithmus fest. Der DQN wurde entwickelt, um mehrdimensionale Zustandsräume, wie beispielsweise Bilder, zu verarbeiten. Er nutzt einen kontinuierlichen Zustandsraum, jedoch einem diskreten Aktionsraum. Der Algorithmus verwendet einen Puffer, in dem Beobachtungen gesammelt werden. Eine zufällige Menge von Beobachtungen wird aus diesem Puffer genommen, um die Strategie zu erlernen. Zusätzlich nutzt er eine Kopie seines Q-Netzwerks, ein sogenanntes Target-Netzwerk, um das Q-Netzwerk zu trainieren. Später wird das Target-Q-Netzwerk mit den Gewichten des Q-Netzwerks aktualisiert. Dies führt zu einer gewissen Stabilität im Training, da die Q-Werte aus dem Target-Q-Netzwerk genutzt werden, um das Q-Netzwerk zu trainieren und das Target-Q-Netzwerk seltener aktualisiert wird als das Q-Netzwerk.

2.3 PPO

PPO ist ein modellfreier, on-policy RL Algorithmus, der 2017 von OpenAI vorgestellt wurde [6]. Er führt eine vorher festgelegte Anzahl von Aktionen in der Umgebung aus, basierend auf seiner aktuellen Strategie, und dabei sammelt er Beobachtungen. Mit diesen gesammelten Beobachtungen, einschließlich Aktionen, Zustände und Belohnungen, passt er seine aktuelle Strategie an, mit dem Ziel, die Summe der Belohnungen zu erhöhen. Dabei berechnet er die Advantage, also die Bewertung der Aktion in einem bestimmten Zustand. Diese Berechnung erfolgt aus den gesammelten Belohnungen und geschätzten Belohnungen. Für die Schätzung verwendet er eine Technik namens Generalized Advantage Estimation (GAE) [7]. Der Algorithmus nutzt eine Actor-Critic-Architektur, bestehend aus zwei künstlichen neuronalen Netzwerken: einem für die Aktionen, genannt Actor, und einem für die Bewertung der Aktionen in einem bestimmten Zustand, genannt Critic.

2.4 TD3

TD3 ist ein modellfreier, off-policy RL Algorithmus [8]. Er ist eine Erweiterung von Deep Deterministic Policy Gradient (DDPG) [9]. DDPG wird für kontinuierliche Zustands- und Aktionsräume verwendet und nutzt zwei künstliche neuronale Netzwerke: den Actor, der die Aktion für einen bestimmten Zustand liefert, und den Critic, der die Bewertung dieser Aktion in diesem Zustand liefert. Es wird ein Puffer genutzt, in dem Beobachtungen gesammelt werden. Von diesem Puffer wird eine zufällige Menge an Beobachtungen

genommen, um die beiden Netzwerke zu trainieren. Das Ziel ist es, die Bewertungen (Q-Werte) zu maximieren und dementsprechend die Aktion zu wählen. Um neue Wege zu finden, wird der Aktion eine Zeitlang meistens ein Rauschen hinzugefügt. Zudem nutzt er einen sogenanntes Target-Netzwerk, eine Kopie des Actor-Critic-Netzwerks. Dieses wird dazu verwendet, das eigentliche Actor-Critic-Netzwerk zu trainieren. Nach einer festgelegten Zeit werden die Gewichte des Target-Netzwerks mit denen des Actor-Critic-Netzwerks aktualisiert.

Der TD3 Algorithmus weist gegenüber DDPG drei Besonderheiten auf. Die erste bezieht sich auf die "Twin" Bezeichnung. Anstatt nur eine Q-Funktion zu erlernen, lernt TD3 zwei Q-Funktionen und verwendet den minimalen Wert der beiden, um den Fehler zu berechnen. Zusätzlich aktualisiert er den Actor und das Target-Netzwerk seltener im Vergleich zum Critic-Netzwerk, woher die Bezeichnung "Delayed" kommt. Die letzte Besonderheit ist ein geringes Rauschen, das der Aktion aus dem Target Actor beim Aktualisieren des Critic-Netzwerks hinzugefügt wird. All diese Besonderheiten dienen dazu, Stabilität in das Training zu bringen.

2.5 Sim-to-Real Gap

Mit Sim-to-Real Gap wird ein Problem bezeichnet, mit dem die Forschung für autonome Systeme konfrontiert ist, wenn sie diese Systeme in einer Simulationsumgebung entwickeln und in der realen Umgebung anwenden. Eine Simulation ist das am meisten verwendete Werkzeug beim maschinellen Lernen. Es ist kostengünstig, sicher und es gibt mehr Kontrolle über die Umgebung, in der ein autonomes System entwickelt werden soll. Doch da es unmöglich ist, eine reale Umgebung in der Simulation vollständig abzubilden, gibt es eine Diskrepanz zwischen Simulation und echter Anwendung, die zu unvorhersehbarem Verhalten eines autonomen Systems in der realen Umgebung führen kann. Es gibt viele Arbeiten, die das Sim-to-Real Gap lösen oder überbrücken wollen, wie zum Beispiel die Arbeit von Peng, Xue Bin et al. aus dem Jahr 2018 mit dem Ansatz von Dynamic Randomization, wo dem System zufällige Bewegungen bei der Simulation hinzugefügt werden, um es auch auf zufällige Zustände in der realen Umgebung vorzubereiten [10]. Oder es werden zusätzlich Daten aus der realen Umgebung zur Simulation hinzugefügt, wie in der Arbeit von Chebotar, Yevgen et al. im Jahr 2019 vorgestellt wurde [11]. Im Jahr 2016 wurde von Agrawal, Pulkit et al. ein Ansatz präsentiert, bei dem das System keine vordefinierte Simulationsumgebung hat, sondern sie durch die Interaktion mit der

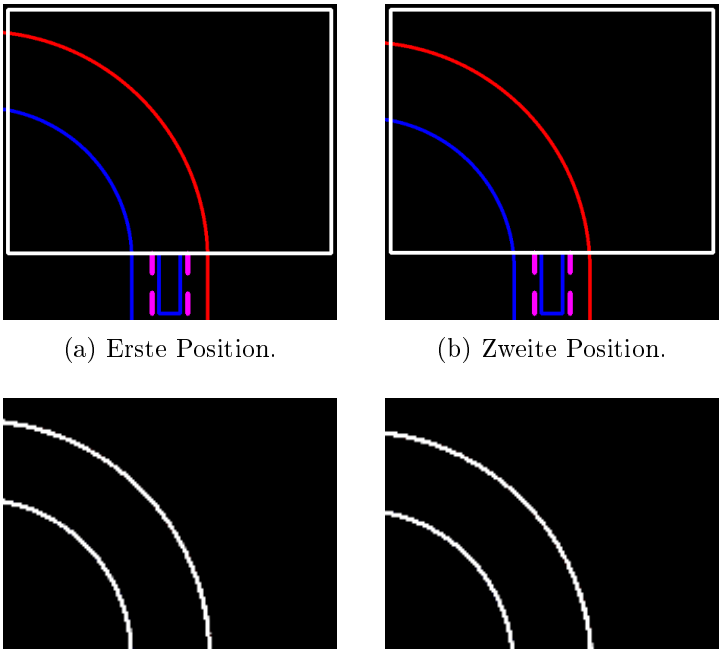
realen Umgebung selbst erstellt, wodurch die Simulation näher an der Realität herankommt [12]. Auch die Arbeit von Fynn Luca Maaß und Stephan Pareigis aus dem Jahr 2022, bezogen auf die Diskrepanz in der Steuerung, überbrückt das Sim-to-Real Gap, indem das System relative Lenkwinkel und nicht die absoluten Lenkwinkel verwendet [1].

3 Gemeinsame Architektur

Dieses Kapitel widmet sich der Beschreibung der gemeinsamen Architektur, die bei TD3, TD3Δ, PPO, PPOΔ, DQN und DQNΔ verwendet wird. Dabei werden der Fahrsimulator, die Soll-Lenkfunktion, die Abbruchkriterien und die relevanten Hyperparameter beschrieben. Zudem werden auch die Teststrecken nach dem Training vorgestellt.

3.1 Trainingsumgebung

Das Training wurde in einer virtuellen Maschine auf der Google Cloud Platform (GCP) durchgeführt [13]. Für die Nutzung der RL Algorithmen wurde die Stable Baselines3 (SB3) Bibliothek mit der Version 2.0.0 verwendet. SB3 ist eine Sammlung zuverlässiger Implementierungen von RL Algorithmen in PyTorch [14]. Um die Komplexität zu reduzieren, wurde ein 2D-Fahrsimulator von Daniel Riege [15] eingesetzt. Dabei werden bereits vorverarbeitete Kameraaufnahmen simuliert, um das Transformieren und Segmentieren von Kamerabildern zu umgehen. Die Aufnahmen zeigen den Bereich vor dem Fahrzeug aus der Vogelperspektive, wie in der Region of Interest (ROI) in Abbildung 3.1a zu sehen ist. Die Farben der Fahrspurränder können selbst ausgewählt werden, um zum Beispiel die Fahrbahnmitte und den rechten Rand der Fahrspur unterschiedlich zu kennzeichnen. In diesem Projekt sind zwar die Fahrspurränder in unterschiedlichen Farben gewählt, wie in Abbildung 3.2 zu sehen ist, doch spielen die Farben keine Rolle, da diese später entfernt werden, wie in Abbildung 3.1c aufgezeigt. Es werden 20 Bilder pro Sekunde, auf Englisch FPS, gemacht. Somit sind es 20 Simulationsschritte pro Sekunde. Der maximale Lenkwinkel beträgt 33°, mit einer Lenkgeschwindigkeit von 70° pro Sekunde. Bei 20 FPS entsprechen das 3.5° pro Schritt. Die Aufnahmen umfassen einen 9.6 mal 12.8 Meter großen Bereich vor dem Fahrzeug aus der Vogelperspektive. Zwei Aufnahmen werden hintereinander in Schwarzweiß-Bilder mit einer Größe von 120 mal 160 Pixel umgewandelt, wie in Abbildung 3.1c und 3.1d zu sehen ist, und an das künstliche neuronale Netzwerk übergeben. Dieses liefert eine Aktion und eine Bewertung



(a) Erste Position. (b) Zweite Position.

(c) Erster Bild. (d) Zweiter Bild.

Abbildung 3.1: Hier ist das Fahrzeug mit dem ROI dargestellt. Oben befinden sich die Positionen des Fahrzeugs, darunter die formatierten Bilder. Zwischen den Positionen (a) und (b) liegen 0.05 Sekunden, was einem Schritt in der Simulation entspricht. Dies entspricht 20 Frames per Second (FPS). Der ROI ist 9.6×12.8 Meter groß (a). In Pixel umgerechnet sind das 480×640 Pixel. Das aufgenommene Bild von 480×640 Pixeln liegt im RGB-Format vor und wird zu einem 120×160 Pixel großen Schwarzweißbild formatiert (c). Zwei solcher formatierter Bilder werden zusammengefügt und an das künstliche neuronale Netzwerk übergeben.

dieser Aktion und des dazugehörigen Zustands bei TD3 und PPO oder Q-Werte bei DQN zurück. Die künstlichen neuronalen Netzwerke werden in den einzelnen Kapiteln zu den RL Algorithmen behandelt. Die Aktion wird bei TD3, PPO und DQN in einen absoluten Lenkwinkel umgewandelt, und bei TD3Δ, PPOΔ und DQNΔ in einen relativen Lenkwinkel. Somit ist der Soll-Lenkwinkel bei TD3Δ, PPOΔ und DQNΔ der Ist-Lenkwinkel plus der relative Lenkwinkel. Die Geschwindigkeit wurde konstant auf $30\frac{km}{h}$ festgelegt.

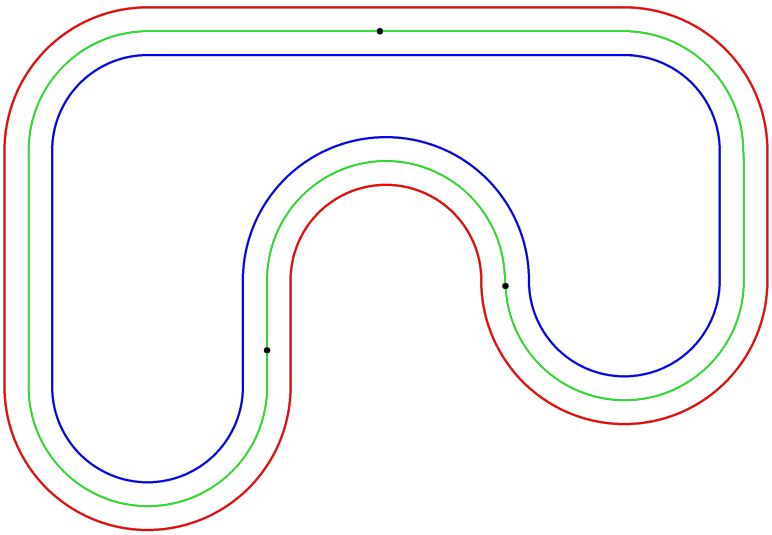


Abbildung 3.2: Hier wird die Karte gezeigt, auf der das Fahrzeug trainiert wird. Die Bahnbreite beträgt 3 Meter und entspricht 150 Pixel. Die grüne Linie auf der Fahrspurmitte wird nicht in die Bilder aufgenommen und dient zur Berechnung der Querabweichung auf Englisch Cross Track Error (CTE). Es gibt drei Spawnpunkte, die hier mit einem schwarzen Punkt markiert sind, und das Fahrzeug wird bei jedem Neustart einer Episode zufällig auf einer dieser Stellen platziert, sowohl in eine als auch in die andere Richtung.

3.2 Soll-Lenkwinkelfunktion

Für den klassischen TD3, DQN und PPO wird das Ergebnis von einem künstlichen neuronalen Netz in einen absoluten Lenkwinkel umgewandelt und direkt als Soll-Lenkwinkel L_{soll} verwendet. Für TD3Δ, DQNΔ und PPOΔ wird das Ergebnis in einen relativen Lenkwinkel umgewandelt, der zum aktuellen Ist-Lenkwinkel L_{ist} addiert wird. Die Gleichungen 3.1 und 3.2 beschreiben dieses Verhalten. Die Funktionen L_x^{a} für den absoluten Lenkwinkel und L_x^{r} werden in den einzelnen Kapiteln für TD3, DQN und PPO beschrieben. Das $x \in \{\text{td3}, \text{dqn}, \text{ppo}\}$ steht für die einzelnen RL Algorithmen.

$$L_{\text{soll}}(a) = L_x^{\text{a}}(a) \tag{3.1}$$

$$L_{\text{soll}}^{\Delta}(a) = L_{\text{ist}} + L_x^{\text{r}}(a) \tag{3.2}$$

3.3 Ist-Lenkwinkelfunktion

Der neue Ist-Lenkwinkel L_{ist} wird berechnet, indem die Differenz ΔL zwischen dem alten Ist-Lenkwinkel und dem Soll-Lenkwinkel L_{soll} zum vorherigen Ist-Lenkwinkel addiert wird. Dabei ist die maximale Änderung auf -3.5 im negativen und $+3.5$ im positiven Bereich begrenzt. Die Gleichungen 3.3, 3.4 und 3.5 beschreiben diese Berechnung. In der Gleichung 3.6 wird die Situation dargestellt, bei der ein Offset zum Ist-Lenkwinkel addiert wird.

$$\Delta L = L_{\text{soll}} - L_{\text{ist}} \tag{3.3}$$

$$\delta l = \begin{cases} -3.5 & \text{für } \Delta L < -3.5 \\ 3.5 & \text{für } \Delta L > 3.5 \\ \Delta L & sonst \end{cases} \tag{3.4}$$

$$L_{\text{ist}_n} = L_{\text{ist}_{n-1}} + \delta l \tag{3.5}$$

$$L_{\text{ist}}^{\text{Offset}} = L_{\text{ist}_n} + Offset \tag{3.6}$$

3.4 Abbruchkriterien

Für jede Episode gibt es nur zwei Abbruchkriterien. Das erste ist, wenn der CTE einen Meter übersteigt, und das zweite ist das Erreichen von 6000 Schritten und somit fünf Minuten Fahrzeit. In der Gleichung 3.7 ist dies veranschaulicht. Die Variable s repräsentiert hier die Anzahl der Schritte.

$$A(cte, s) = \begin{cases} 1 & \text{für } cte > 1 \\ 1 & \text{für } s \geq 6000 \\ 0 & \text{für } sonst \end{cases} \tag{3.7}$$

3.5 Belohnungsfunktion

Die Belohnungsfunktion wurde einfach gehalten und entspricht einem Meter minus der Querabweichung $cte \in \mathbb{R}$, wie in Gleichung 3.8 dargestellt. Da CTE > 1 ein Abbruchkriterium ist, bewegt sich die Belohnung (Reward) im Bereich von null bis eins, $r \in [0; 1]$.

Somit ist $r = 1$, wenn das Fahrzeug perfekt auf der Mittellinie der Fahrspur fährt.

$$r(cte) = 1 - cte \tag{3.8}$$

3.6 Hyperparameter

Im Folgenden sind nennenswerte Hyperparameter mit ihrer Bezeichnung und Beschreibung aufgelistet. Die Bezeichnungen sind auf Englisch und stammen aus SB3. In Klammern sind die RL Algorithmen aufgeführt, bei denen diese Parameter festgelegt werden.

learning-rate: Schrittweite, mit der die Gewichte in einem neuronalen Netzwerk angepasst werden. (TD3, PPO, DQN)

gamma: Ein Discount-Faktor, der angibt, wie stark zukünftige Belohnungen berücksichtigt werden. (TD3, PPO, DQN)

buffer-size: Die maximale Anzahl an Beobachtungen, die während des Trainings bei jedem Schritt gesammelt werden. Eine Beobachtung kann Zustand, Aktion, Belohnung und nächster Zustand umfassen. Ist der Puffer voll, werden die ältesten Beobachtungen überschrieben. (TD3, DQN)

batch-size: Die Anzahl an Beobachtungen aus einem Puffer, anhand derer der RL Algorithmus seine Strategie in einem oder mehreren Schritten lernt. Diese Beobachtungen können nacheinander oder zufällig aus dem Puffer genommen werden. (TD3, PPO, DQN)

target-policy-noise: Die Standardabweichung des Gaußschen Rauschens, das zur Zielstrategie hinzugefügt wird, um sie zu glätten. (TD3)

target-noise-clip: Der Grenzwert für das Gaußsche Rauschen, das zur Zielstrategie hinzugefügt wird. (TD3)

n-steps: Die Anzahl der Schritte, die vor jedem Update der Strategie durchgeführt werden. (PPO)

n-epochs: Die Anzahl der Trainingsdurchläufe pro Update der Strategie. Bei jedem Durchlauf werden eine in *batch-size* festgelegte Anzahl an Beobachtungen, die bei *n-steps* gesammelt wurden, nacheinander genommen und anhand dieser trainiert. (PPO)

gae-lambda: Eine Gewichtung zwischen tatsächlichen Belohnungen und den geschätzten Belohnungen aus dem GAE. Ein Wert von 1 bedeutet, dass nur die geschätzten Belohnungen verwendet werden. (PPO)

3.7 Teststrecken

Abbildung 3.3: Hier wird die erste Teststrecke gezeigt, auf der das Fahrzeug fahren gelassen wird. Die Bahnbreite ist 3 Meter große und entspricht 150 Pixel und die Länge beträgt 90 Metern und entspricht 4500 Pixel. Die grüne Linie auf der Fahrspurmitte wird für die Berechnung des CTE verwendet. Der Spawnpunkt befindet sich ganz links

.

Nach dem Training wird das Fahrzeug für jeden Algorithmus (TD3, TD3Δ, PPO, PPOΔ, DQN und DQNΔ) auf drei Fahrstrecken fahren gelassen. Die erste Strecke ist 90 Meter lang und 3 Meter breit, wie in Abbildung 3.3 dargestellt. Nach 10 Metern wird ein Offset zum Ist-Lenkwinkel addiert. Es wird beobachtet, wie sich das Fahrzeug verhält, und die Position des Fahrzeugs sowie der relative oder absolute Lenkwinkel an dieser Position werden protokolliert. Diese Daten werden graphisch dargestellt, beschrieben und erklärt und sind im Kapitel 7 zu finden.

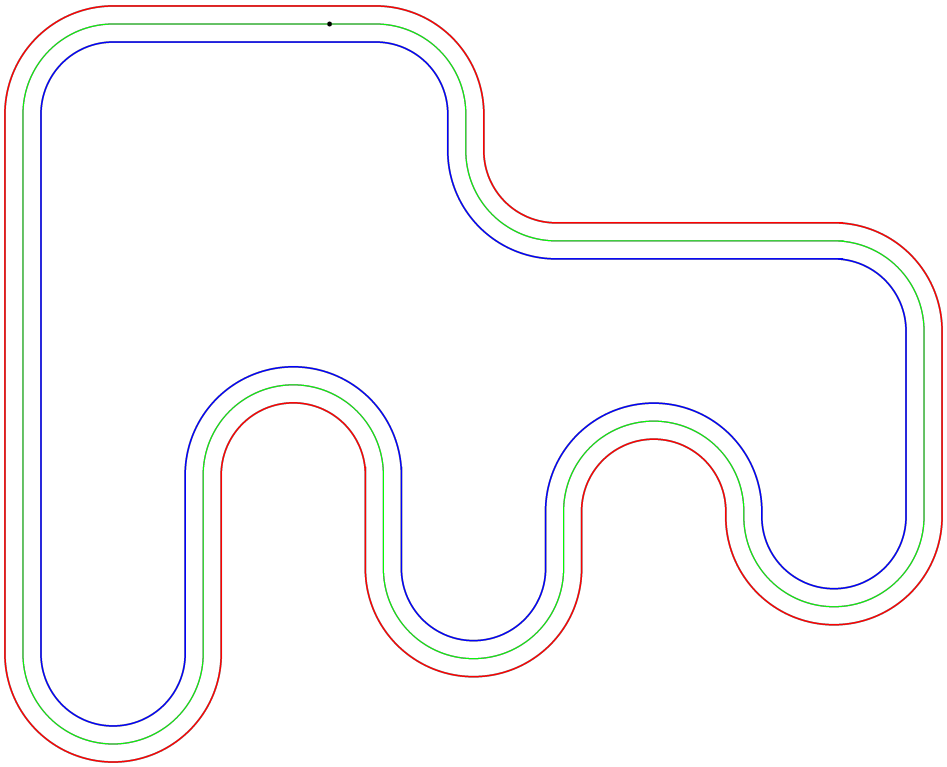


Abbildung 3.4: Hier wird die zweite Teststrecke gezeigt, auf der das Fahrzeug fahren gelassen wird. Die Bahnbreite beträgt 3 Meter, was 150 Pixel entspricht. Es gibt sechs Linkskurven und drei Rechtskurven. Die grüne Linie auf der Fahrspurmitte wird für die Berechnung des CTE verwendet. Der Spawnpunkt ist oben mit einem schwarzen Punkt markiert.

Die Abbildung 3.4 zeigt die zweite Strecke, auf der das Fahrzeug für alle RL Algorithmen fahren gelassen wird. Die Fahrspur ist 3 Meter breit und bildet eine geschlossene Schleife. Es gibt nur einen Spawnpunkt, und das Fahrzeug fährt gegen den Uhrzeigersinn. Das Fahrzeug fährt fünf Minuten lang mit einem Offset von 10°. Der CTE wird protokolliert.

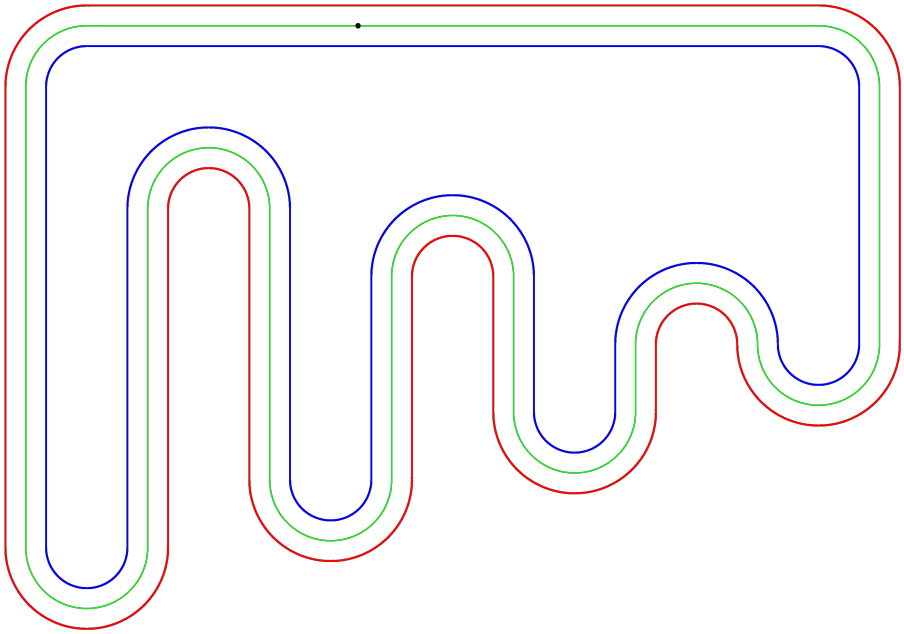


Abbildung 3.5: Hier wird die dritte Teststrecke gezeigt, auf der das Fahrzeug fahren gelassen wird. Die Bahnbreite beträgt 3 Meter, was 150 Pixel entspricht. Es gibt sechs Linkskurven und drei Rechtskurven. Die grüne Linie auf der Fahrspurmitte wird für die Berechnung des CTE verwendet. Der Spawnpunkt ist oben mit einem schwarzen Punkt markiert.

Die Abbildung 3.5 zeigt die dritte Strecke, auf der das Fahrzeug für alle RL Algorithmen fahren gelassen wird. Die Kurven wurden enger gewählt, um die Grenzen der Algorithmen auszutesten. Auch hier soll die Fahrt fünf Minuten dauern. Es wurde auf einen Offset verzichtet.

4 TD3/TD3Δ

Dieses Kapitel widmet sich der detaillierten Beschreibung der Architektur von TD3 und TD3Δ, wobei das verwendete kürzliche neuronale Netzwerk, die Explorationsphase, wichtige Hyperparameter und die Lenkwinkelfunktionen betrachtet werden.

4.1 Actor-Critic Netzwerk

In der Abbildung 4.1 ist die Architektur des Actor-Critic Netzwerks dargestellt und beschrieben. Es handelt sich um zwei Netzwerke: eins für die Aktion (Actor) und eins für die Bewertung (Critic). Zwei Schwarzweißbilder werden sowohl durch das Actor- als auch durch das Critic-Netzwerk geschickt. Zusätzlich zu den Bildern erhält das Critic-Netzwerk das Ergebnis des Actor-Netzwerks. Somit wird der Actor trainiert, die richtige Aktion zurückzuliefern, und der Critic lernt, wie er diese Aktion in Bezug auf die eingegebenen Bilder bewerten soll. Diese Bewertung wird wiederum genutzt, um den Actor zu trainieren.

4.2 Hyperparameter

Tabelle 4.1: Hyperparameter für den TD3/TD3Δ Algorithmus

Name	**Wert**
learning-rate	1×10^{-6}
gamma	0.9
buffer-size	3×10^{5}
batch-size	256
target-policy-noise	0.05
target-noise-clip	0.08

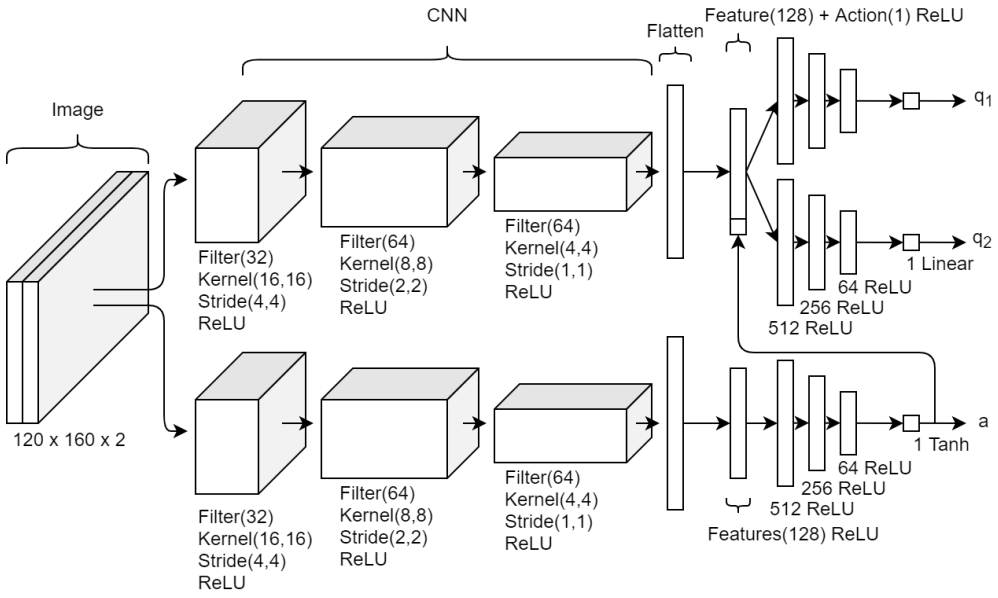


Abbildung 4.1: Hier ist die Architektur des künstlichen neuronalen Netzes für den TD3/TD3Δ Algorithmus dargestellt. Zwei hintereinander aufgenommene, in Schwarzweiß konvertierte Bilder mit der Größe von 120×160 Pixel werden durch durch das Critic-Netzwerk oben und Actor-Netzwert unten geschickt. Sie durchlaufen jeweils drei Faltungsschichten mit den Filtergrößen 32, 64 und 64 und werden nach dem Abflachen auf eine Merkmalschicht mit der Größe 128 abgebildet. Die Größen der Faltungskerne sind nacheinander 16×16, 8×8 und 4×4, und die Schrittlängen sind 4×4, 2×2 und 1×1. Die Aktivierungsfunktionen sind gleichgerichtet linear. Beim Critic-Netzwerk wird die Merkmalschicht zusätzlich mit dem Ergebnis des Actor-Netzwerkes konkateniert. Danach werden die Werte parallel durch drei Netzwerke mit jeweils drei vollständig verbundene Schichten, mit 512, 256 und 64 Knoten, geschickt und auf drei Wert abgebildet. Die Aktivierungsfunktionen sind bei allen Schichten auch gleichgerichtete linear. Am Ende werden die Werte für Critic auf ein linearen Wert und bei Actor auf einen Wert mit Tangens-Hyperbolicus als Aktivierungsfunktion abgebildet. Das Ergebnis des Actor-Netzwerkes $a \in [-1; 1]$ ist der Wert für die Aktion und des Critic-Netzwerkes sind die Werte für die Bewertungen $q_{1,2} \in \mathbb{R}$.

In der Tabelle 4.1 sind die gewählten Parameter für den TD3/TD3Δ aufgelistet. Es hat sich gezeigt, dass nur bei eine Lernrate von 1×10^{-6} der Algorithmus in der Lage ist, anhand der Bilder, ein Strategie zu finden und stabil zu bleiben.

4.3 Lenkwinkelfunktion

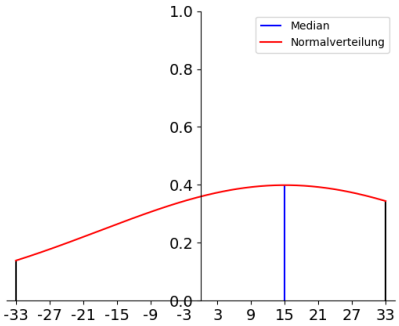
Der Wert für die Aktion, den das Actor-Critic-Netzwerk liefert, wird in einen absoluten Lenkwinkel von $-33°$ bis $33°$ für TD3 oder einen relativen Lenkwinkel von $-3.5°$ bis $3.5°$ für TD3Δ umgewandelt. Dafür werden einfache lineare Funktionen verwendet. Zusätzlich wird während der Explorationsphase zum Wert a ein Zufallswert aus der Normalverteilung addiert. Die Gleichungen 4.1 und 4.2 beschreiben diese Funktionen.

$$L^{\mathrm{a}}_{\mathrm{td3}}(a) = 33(a + X) \; mit \; X \sim \mathcal{N}(0,\, \sigma^2) \tag{4.1}$$

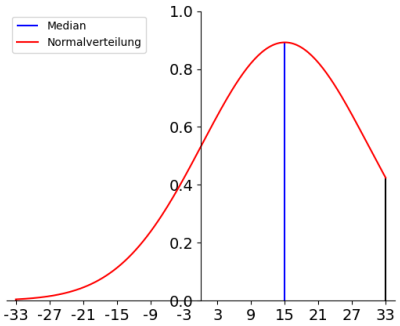
$$L^{\mathrm{r}}_{\mathrm{td3}}(a) = 3.5(a + X) \; mit \; X \sim \mathcal{N}(0,\, \sigma^2) \tag{4.2}$$

4.4 Explorationsphase

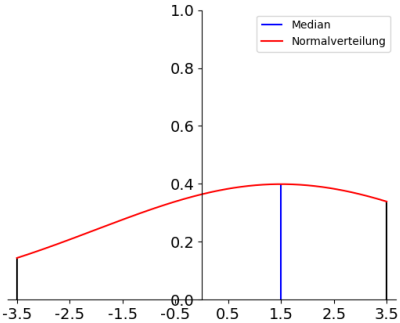
Die Explorationsphase umfasst 360×10^3 Schritte oder fünf Stunden. Nach maximal 10 Stunden und somit 720×10^3 Schritte wird das Training beendet. Nach jeweils 6000 Schritte wird eine neue Episode gestartet und das Fahrzeug zufällig auf der Bahn platziert. Die Spawnpunkte sind in Abbildung 3.2 markiert. Das Training wird auch beendet, wenn zwölf Episoden hintereinander mit jeweils 6000 Schritte ohne Unterbrechung erreicht wurden, was eine Stunde Fahrt entspricht. Innerhalb der 360×10^3 Schritte wird bei jedem Schritt Rauschen zum Lenkwinkel hinzugefügt, das aus einer Normalverteilungsfunktion stammt. Bei jedem Schritt wird die Standardabweichung der Normalverteilung reduziert, wie in Abbildung 4.2 dargestellt, bis nach 360×10^3 Schritte kein Rauschen mehr hinzugefügt wird. Dann beginnt die Exploitationsphase.



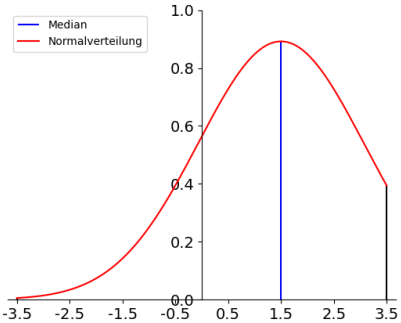
(a) $\sigma^2 = 1$.



(b) $\sigma^2 = 0.2$.



(c) $\sigma^2 = 1$.



(d) $\sigma^2 = 0.2$.

Abbildung 4.2: Hier sind die Wahrscheinlichkeitsverteilungen für den absoluten und relativen Lenkwinkel bei TD3 und TD3Δ dargestellt. Auf der x-Achse ist der Lenkwinkel α abgetragen, mit seinen maximalen Grenzen von $-33°$ bis $33°$ für TD3 und von $-3.5°$ bis $3.5°$ für TD3Δ. Der rote Graph stellt die gaußsche Glockenkurve dar und zeigt die Wahrscheinlichkeit an, mit der ein bestimmter Lenkwinkel basierend auf dem Median, dargestellt als blaue Linie, ausgewählt wird. Der Median ist der Wert, der aus den Rückgabewerten des künstlichen neuronalen Netzwerks berechnet wird, nämlich $1 \widehat{=} 33°$ für TD3 oder $1 \widehat{=} 3.5°$ für TD3Δ. Während der Explorationsphase wird der Graph immer schmaler und die Abweichung vom gelieferten Lenkwinkel wird kleiner. Sobald die Explorationsphase beendet ist, wird lediglich der Lenkwinkel verwendet.

5 PPO/PPOΔ

Dieses Kapitel widmet sich der detaillierten Beschreibung der Architektur von PPO und PPOΔ, wobei das verwendete kürzliche neuronale Netzwerk, die Explorationsphase, wichtige Hyperparameter und die Lenkwinkelfunktionen betrachtet werden.

5.1 Actor-Critic Netzwerk

In Abbildung 5.1 ist die Architektur des Actor-Critic Netzwerks dargestellt und beschrieben. Es handelt sich um ein Netzwerk, das sowohl eine Aktion als auch eine Bewertung ausgibt. Zwei Schwarzweißbilder werden durch ein gemeinsames Convolutional Neural Network (CNN) geleitet, und das Ergebnis wird sowohl an das Teilnetzwerk des Actors weitergegeben, das die Aktion liefert, als auch an den Critic, der die Bewertung dieser Aktion im entsprechenden Zustand liefert.

5.2 Hyperparameter

Tabelle 5.1: Hyperparameter für den PPO/PPOΔ Algorithmus

Name	**Wert**
learning-rate	1×10^{-6}
gamma	0.9
batch-size	128
n-steps	512
n-epochs	20
gae-lambda	0.5

In der Tabelle 5.1 sind die gewählten Parameter für den PPO/PPOΔ aufgelistet. Um den Algorithmus dazu zu bringen, eine Strategie zu finden, war es nötig den *gae-lambda*

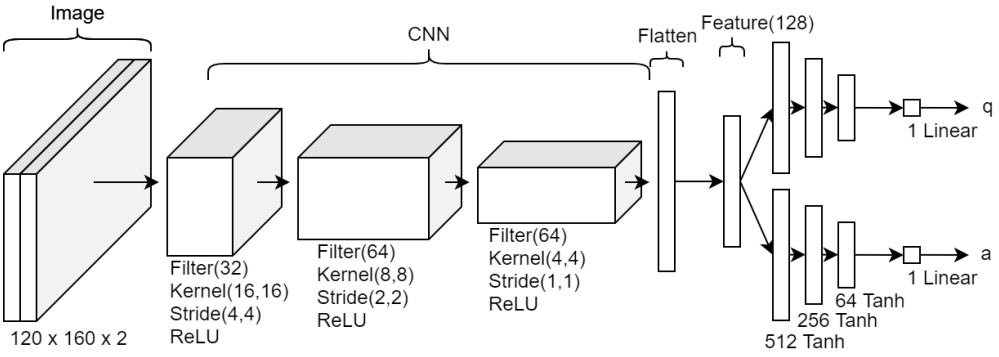


Abbildung 5.1: Hier ist die Architektur des künstlichen neuronalen Netzes für den PPO/PPOΔ Algorithmus dargestellt. Zwei hintereinander aufgenommene, in Schwarzweiß konvertierte Bilder mit der Größe von 120×160 Pixel werden durch das Actor-Critic Netzwerk geschickt. Sie durchlaufen drei Faltungsschichten mit den Filtergrößen 32, 64 und 64 und werden nach dem Abflachen auf eine Merkmalschicht mit der Größe 128 abgebildet. Die Größen der Faltungskerne sind nacheinander 16×16, 8×8 und 4×4, und die Schrittlängen sind 4×4, 2×2 und 1×1. Die Aktivierungsfunktionen sind gleichgerichtet linear. Danach werden die Werte sowohl beim Actor als auch beim Critic jeweils durch drei vollständig verbundene Schichten geschickt. Diese Schichten haben 512, 256 und 64 Knoten sowie die Aktivierungsfunktion Tangens-Hyperbolicus. Die Ergebnisse werden auf einen linearen Wert $q \in \mathbb{R}$ für den Critic und auf einen Wert $a \in [-1, 1]$ für den Actor abgebildet.

Parameter zu verkleinern. Er wurde auf 0.5 gesetzt, um die tatsächlichen und geschätzten Belohnungen gleichermaßen einzubeziehen.

5.3 Lenkwinkelfunktion

Der Wert für die Aktion, den das Actor-Critic Netzwerk liefert, wird in einen absoluten Lenkwinkel von $-33°$ bis $33°$ für PPO und in einen relativen Lenkwinkel von $-3.5°$ bis $3.5°$ für PPOΔ umgewandelt. Hierbei werden einfache lineare Funktionen verwendet, wie sie in den Gleichungen 5.1 und 5.2 beschrieben sind.

$$L^{\mathrm{a}}_{\mathrm{ppo}}(a) = 33a \tag{5.1}$$

$$L^{\mathrm{r}}_{\mathrm{ppo}}(a) = 3.5a \tag{5.2}$$

5.4 Exploration

Die Exploration geschieht durch das Einbeziehen der Entropie in die Verlustfunktion. Dabei wurde der Koeffizient für die Entropie, also das Gewicht, mit dem die Entropie einbezogen wird, auf 0.01 gesetzt. Ansonsten wird eine gute Strategie anhand der tatsächlichen und geschätzten Belohnungen gesucht.

6 DQN/DQNΔ

Dieses Kapitel widmet sich der detaillierten Beschreibung der Architektur von DQN und DQNΔ, wobei das verwendete kürzliche neuronale Netzwerk, die Explorationsphase, wichtige Hyperparameter und die Lenkwinkelfunktionen betrachtet werden.

6.1 Q-Netzwerk

In der Abbildung 6.1 ist die Q-Netzwerk Architektur dargestellt und beschrieben. Es bekommt zwei hintereinander aufgenommene Schwarzweißbilder und liefert die Bewertungen für diesen Input und die diskreten Aktionen a. Hierbei repräsentiert a die Werte von 0 bis 10 für DQNΔ und von 0 bis 20 für DQN. Je nachdem, wie der Parameter für die Exploration eingestellt ist, wird entweder die diskrete Aktion mit der höchsten Bewertung oder eine zufällige Aktion an den Agenten weitergegeben.

6.2 Hyperparameter

Tabelle 6.1: Hyperparameter für den DQN/DQNΔ Algorithmus

Name	**Wert**
learning-rate	1×10^{-6}
gamma	0.9
buffer-size	3×10^{5}
batch-size	256

In Tabelle 6.1 sind die gewählten Parameter für den DQN/DQNΔ aufgelistet. Diese unterscheiden sich nicht von den Parametern des TD3/TD3Δ Algorithmus.

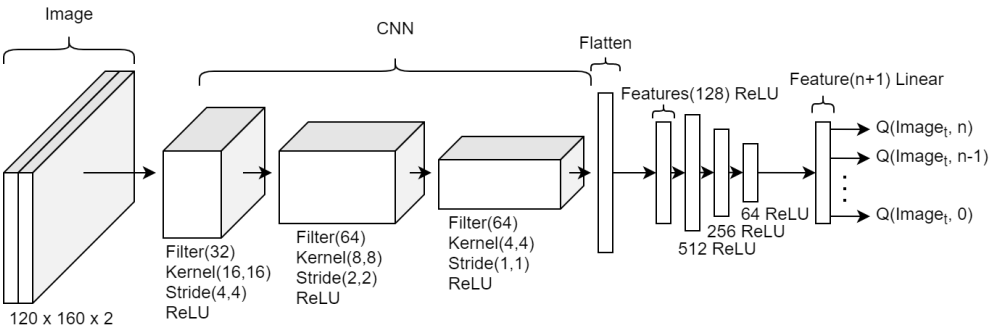


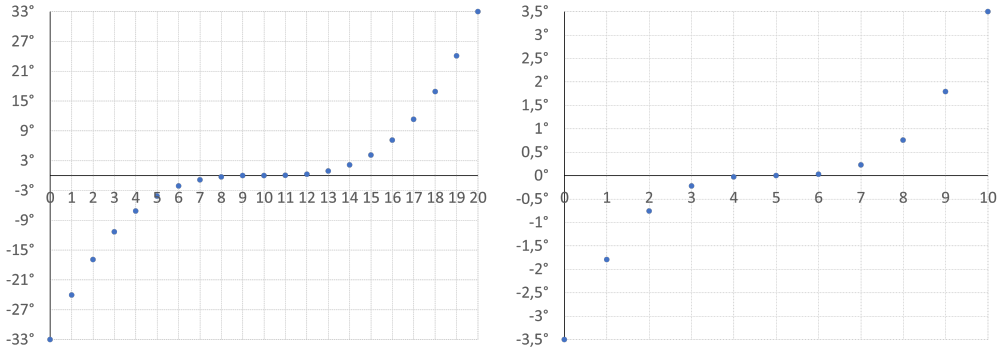
Abbildung 6.1: Hier ist die Architektur des künstlichen neuronalen Netzes für den DQN/DQNΔ Algorithmus dargestellt. Zwei hintereinander aufgenommene, in Schwarzweiß konvertierte Bilder mit der Größe von 120×160 Pixel werden durch das Q-Netzwerk geschickt. Sie durchlaufen drei Faltungsschichten mit den Filtergrößen 32, 64 und 64 und werden nach dem Abflachen auf eine Merkmalschicht mit der Größe 128 abgebildet. Die Größen der Faltungskerne sind nacheinander 16×16, 8×8 und 4×4, und die Schrittlängen sind 4×4, 2×2 und 1×1. Die Aktivierungsfunktionen sind gleichgerichtet linear. Danach werden die Werte durch drei vollständig verbundene Schichten mit 512, 256 und 64 Knoten geschickt und auf eine Merkmalschicht mit der Größe 11 für DQNΔ oder 21 für DQN abgebildet. Die Aktivierungsfunktionen sind bei allen Schichten, außer der letzten Merkmalschicht, ebenfalls gleichgerichtet linear. Das Ergebnis des Q-Netzwerks $Q(s,a) \in \mathbb{R}$ ist die Bewertung der diskreten Aktion $a \in \{0,1,2,\ldots,n\}$ im Zustand $s \in \{Image_0, Image_1, Image_2, \ldots, Image_t\}$, wobei $n = 10$ bei DQNΔ und $n = 20$ bei DQN ist.

6.3 Lenkwinkelfunktionen

Die diskrete Aktion wird in einen absoluten Lenkwinkel von $-33°$ bis $33°$ für DQN oder in einen relativen Lenkwinkel von $-3.5°$ bis $3.5°$ für DQNΔ umgewandelt. Hierbei wird eine Funktion dritten Grades verwendet. Zusätzlich dazu wird während der Explorationsphase mit einer bestimmten Wahrscheinlichkeit ϵ eine zufällige diskrete Aktion $X \sim \mathcal{U}(0,20)$ für DQN beziehungsweise $X \sim \mathcal{U}(0,10)$ für DQNΔ gewählt. Die Gleichungen 6.1 und 6.2 beschreiben diese Funktionen, und Abbildung 6.2 veranschaulicht diese Funktionen ohne den Zufallswertes.

$$L^{\mathrm{a}}_{\mathrm{dqn}}(a) = \begin{cases} \frac{33(a-10)^3}{10^3} & mit\ Wahrscheinlichkeit\ 1-\epsilon \\ \frac{33(X-10)^3}{10^3} & mit\ Wahrscheinlichkeit\ \epsilon \end{cases} \tag{6.1}$$

$$L^{\mathrm{r}}_{\mathrm{dqn}}(a) = \begin{cases} \frac{3.5(a-5)^3}{5^3} & mit\ Wahrscheinlichkeit\ 1-\epsilon \\ \frac{3.5(X-5)^3}{5^3} & mit\ Wahrscheinlichkeit\ \epsilon \end{cases} \tag{6.2}$$



(a) Diskrete Verteilung für den absoluten Lenkwinkel.

(b) Diskrete Verteilung für den relativen Lenkwinkel.

Abbildung 6.2: Hier werden die Gleichungen 6.1 und 6.2 veranschaulicht. Aus dem Q-Netzwerk wird die Aktion $a \in \{0, 1, 2, \ldots, 10\}$ für DQNΔ und $a \in \{0, 1, 2, \ldots, 20\}$ für DQN mit der höchsten Bewertung ausgewählt. Diese wird auf einen Lenkwinkel von -33 bis 33 für DQN oder von -3.5 bis 3.5 für DQNΔ abgebildet werden. Dafür wird eine Funktion dritten Grades verwendet, wobei $a = 5$ für DQNΔ und $a = 10$ für DQN als der Nullwinkel festgelegt werden. Je näher sich a dem Wert 5 oder 10 nähert, desto feiner ist der Lenkwinkel.

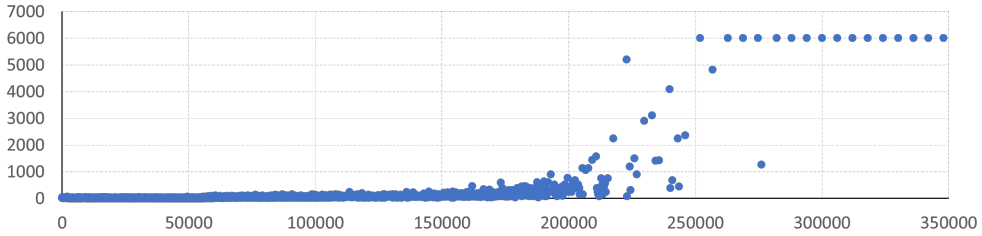
6.4 Explorationsphase

Die Explorationsphase umfasst 72×10^3 Schritte, was einer Stunde entspricht. Das Training wird nach maximal 10 Stunden oder 720×10^3 Schritten abgeschlossen. Nach jeweils 6000 Schritten wird eine neue Episode gestartet, und das Fahrzeug wird zufällig auf der Strecke platziert. Die Spawnpunkte sind in Abbildung 3.2 markiert. Das Training endet auch, wenn zwölf aufeinanderfolgende Episoden, jeweils mit 6000 Schritten ohne Unterbrechung, erreicht werden, was einer Stunde Fahrzeit entspricht. Innerhalb der ersten 72×10^3 Schritte wird mit einer gewissen Wahrscheinlichkeit ϵ eine zufällige Aktion a ausgewählt. Diese Wahrscheinlichkeit beträgt anfangs 100% und wird linear reduziert, so dass sie nach 72×10^3 Schritten bei 1% liegt. Danach beginnt die Exploitationsphase.

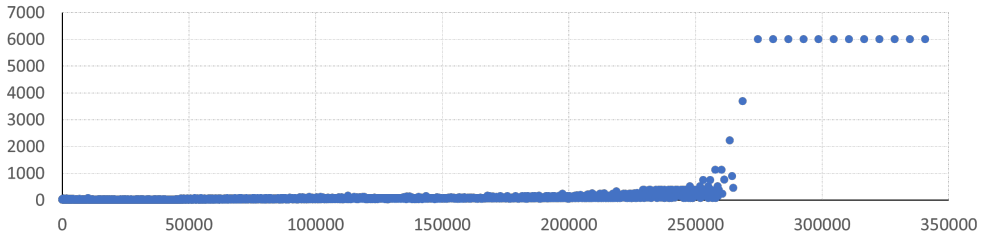
7 Ergebnisse

Dieses Kapitel widmet sich den Ergebnissen des Trainings von TD3, DQN und PPO, mit absolutem und relativem Lenkwinkel. Diese werden miteinander verglichen und interpretiert.

7.1 TD3/TD3Δ



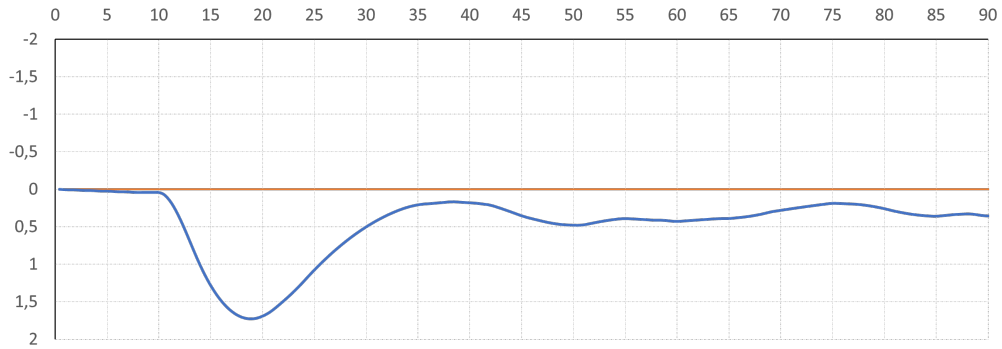
(a) Absoluter Lenkwinkel (TD3).



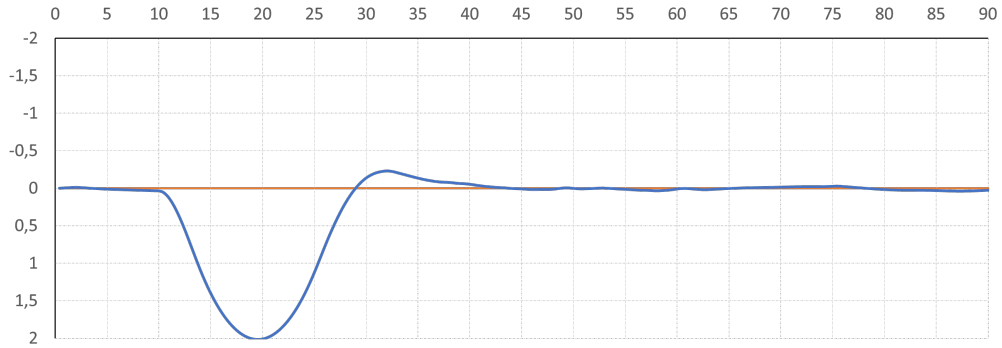
(b) Relativer Lenkwinkel (TD3Δ).

Abbildung 7.1: Die Graphen zeigen die maximal erreichten Schritte nach jeder Episode für den TD3/TD3Δ Algorithmus. Auf der x-Achse sind die Gesamtschritte anstelle der Episoden dargestellt, während auf der y-Achse die Anzahl der Schritte angezeigt wird, die das Fahrzeug ohne Unterbrechung zurückgelegt hat. Dabei wurde ein Maximum von 6000 Schritten festgelegt. In der Abbildung 7.1a wurde das Training mit dem absoluten Lenkwinkel und in der Abbildung 7.1b mit dem relativem Lenkwinkel durchgeführt.

Die Abbildung 7.1 zeigt, wie lange der TD3/TD3Δ Algorithmus gebraucht hat, um eine gute Strategie zu finden. Wenn 6000 Schritte 12 Mal hintereinander ohne Unterbrechung durchgeführt wurden, gilt das Training als erfolgreich. Es ist erkennbar, dass der Algorithmus beim absoluten Lenkwinkel früher beginnt, eine gute Strategie zu entwickeln, bei ungefähr 2×10^5 Schritten. Beim relativen Lenkwinkel beginnt dies ungefähr bei 2.5×10^5 Schritten. Dennoch benötigen beide Ansätze etwa die gleiche Anzahl von Schritten, um die Aufgabe zu bewältigen.



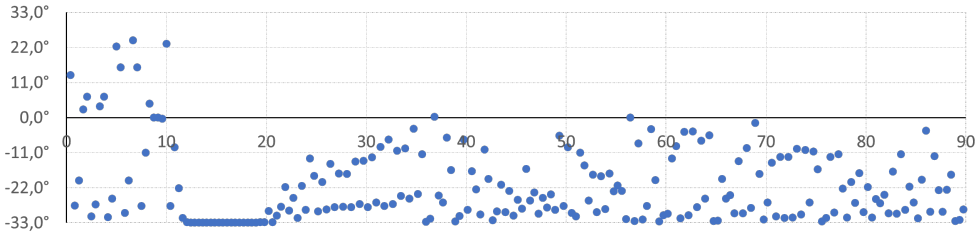
(a) Absoluter Lenkwinkel (TD3).



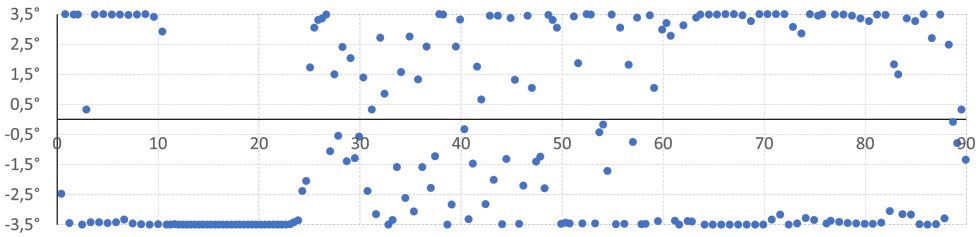
(b) Relativer Lenkwinkel (TD3Δ).

Abbildung 7.2: Die Graphen zeigen die Fahrt des Fahrzeugs auf einer 90 Meter langen, geraden Strecke für den TD3/TD3Δ Algorithmus. Sowohl auf der x- als auch auf der y-Achse sind die Werte in Metern eingetragen. Negative Werte liegen in Fahrrichtung links. Nach 10 Metern fährt das Fahrzeug nach rechts bis zu einem Abstand von ungefähr 1.75 Metern für den absoluten und 2 Metern für den relativen Lenkwinkel, und kehrt dann wieder zurück. In Abbildung 7.2a bleibt das Fahrzeug nach ungefähr 35 Metern durchschnittlich 0.33 Meter rechts von der Fahrspurmitte. In Abbildung 7.2b kehrt das Fahrzeug wieder zurück und fährt nach 40 Metern fast exakt auf der Fahrspurmitte.

Die Abbildung 7.2 zeigt die erste Teststrecke nach dem Training. Auf einer 90 Meter langen, geraden Bahn fährt das Fahrzeug mit $30\frac{km}{h}$. Nach 10 Metern wird zum Ist-Lenkwinkel ein Offset von 25° hinzugefügt. Es ist zu erkennen, dass das Fahrzeug erst nach rechts ausbricht, dann jedoch wieder zurückfährt. Beim absoluten Lenkwinkel kehrt das Fahrzeug nicht vollständig zur Fahrspurmitte zurück, sondern bleibt im durchschnittlichen Abstand von 0.33 Meter rechts von dieser. Dabei sind starke hin und her Lenkbewegungen zu erkennen. Das Fahrzeug versucht ständig, den 25° Offset auszugleichen, was sich im überwiegend negativen absoluten Lenkwinkel in Abbildung 7.3a widerspiegelt. Im Vergleich dazu zeigt Abbildung 7.2b für den relativen Lenkwinkel, dass das Fahrzeug zur



(a) Absoluter Lenkwinkel (TD3).



(b) Relativer Lenkwinkel (TD3Δ).

Abbildung 7.3: Die Grafiken zeigen die Werte für den Lenkwinkel, die das künstliche neuronale Netz erzeugt hat, auf einer 90 Meter langen, geraden Teststrecke für TD3/TD3Δ. Auf der x-Achse sind die Werte in Metern und auf der y-Achse in Grad eingetragen. Negative Werte bedeuten, dass das Fahrzeug nach links fährt. In den ersten 10 Metern sind die Werte für den absoluten Lenkwinkel zwischen $-33°$ und 33° zerstreut. Nach 10 Metern befinden sie sich bei $-33°$. Nach 20 Metern befinden sich die Werte meistens zerstreut im negativen Bereich. Der durchschnittliche negative Lenkwinkel nach 35 Metern beträgt $-22°$. Beim relativen Lenkwinkel befinden sich die Werte in den ersten 10 Metern meistens bei $-3.5°$ oder 3.5°. Danach bei $-3.5°$. Zwischen 20 und 60 Metern erstrecken sich die Werte über den gesamten Bereich, danach befinden sie sich wieder entweder bei $-3.5°$ oder 3.5°

Fahrspurmitte zurückfindet und nahezu exakt auf ihr weiterfährt. Da der relative Lenkwinkel zum Ist-Lenkwinkel addiert wird, wird der Offset ignoriert. In Abbildung 7.3b sind die Werte für den relativen Lenkwinkel dargestellt. In der Phase, in der das Fahrzeug zur Fahrspurmitte zurückkehrt, lenkt es stark nach links, um sich an die Fahrspurmitte anzupassen. Die Werte sind über den gesamten Bereich von $-3.5°$ bis $3.5°$ verteilt. Nachdem es sich an die Fahrspurmitte angepasst hat, bleibt der Lenkwinkel meist in der Nähe von $-3.5°$ oder $3.5°$, wobei er abwechselnd zwischen diesen Werten wechselt. Das ist die beste Strategie, die der TD3 Algorithmus für den relativen Lenkwinkel gefunden hat, um auf der Fahrspurmitte zu bleiben.

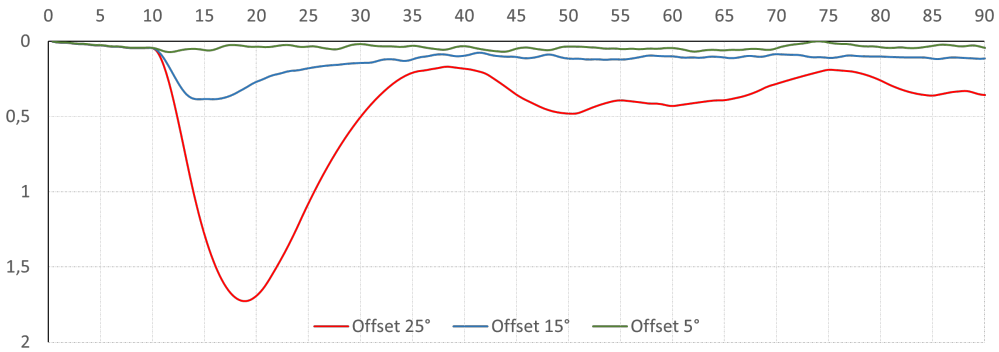


Abbildung 7.4: Der Graph zeigt die Fahrt des Fahrzeugs auf einer geraden, 90 Meter langen Teststrecke beim TD3 Algorithmus mit mehreren Offsets von 25°, 15° und 5°. Sowohl auf der x- als auch y-Achse sind die Werte in Metern eingetragen. Positive Werte liegen in Fahrtrichtung rechts. Nach 10 Metern fährt das Fahrzeug nach rechts und wieder zurück, wobei es bei 25° 1.75 Meter und bei 15° 0.7 Meter erreicht. Bei dem Offset von 5° ist kein Auslenken zu erkennen. Bei allen Offsets fährt das Fahrzeug rechts neben der Fahrspurmitte weiter, wobei es bei dem Offset von 25° einen durchschnittlichen Abstand von 0.33 Metern, bei 15° 0.1 Metern und bei 5° 0.04 Metern bleibt.

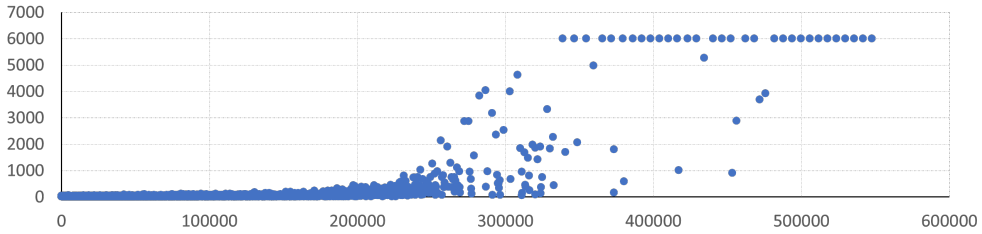
Die Abbildung 7.4 zeigt die Fahrt mit verschiedenen Offsets von 25°, 15° und 5° auf derselben Teststrecke unter Verwendung des herkömmlichen TD3 Algorithmus. Es ist erkennbar, dass das Fahrzeug näher an der Mitte der Fahrspur fährt, je kleiner der Offset ist. Zusätzlich ist bei einem Offset von 25° eine stärkere hin-und-her Lenkbewegung zu beobachten. Der relativ kleine Abstand zwischen der Fahrspurmitte und dem Fahrzeug bei einem Offset von 25°, nachdem es zurückgefahren ist, lässt sich mit der erlernten Strategie erklären. Es hat gelernt, in einem Abstand von durchschnittlich 0.33 Metern mit einem durchschnittlichen Lenkwinkel von 22.5° nach links zu fahren. Das entspricht ungefähr dem Offset von 25°, und wenn sich das Fahrzeug nach rechts bewegt, ist der

Tabelle 7.1: Offset und durchschnittlicher Lenkwinkel für TD3

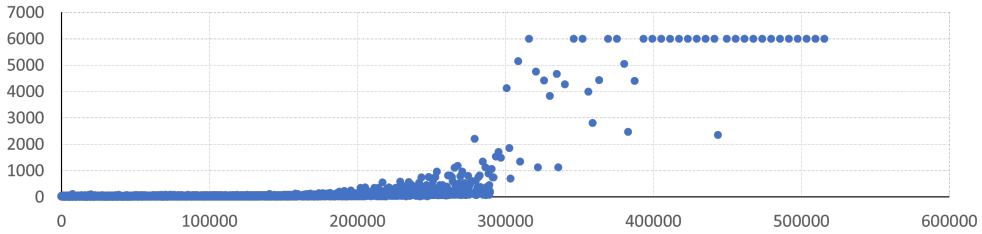
Offset	25°	15°	5°
Lenkwinkel	−22.5°	−13.6°	−6.06°

Lenkwinkel größer als 25°. Somit gleicht es in diesem Abstand den Offset aus und fährt neben der Fahrspurmitte. Für die anderen Offsets gilt dasselbe, nur näher an der Fahrspurmitte. In Tabelle 7.1 sind für die drei Offsets die durchschnittlichen Lenkwinkel aufgelistet.

7.2 PPO/PPOΔ



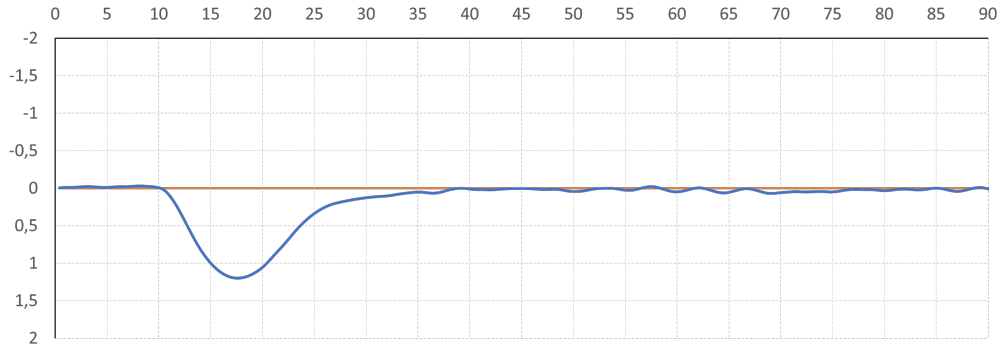
(a) Absoluter Lenkwinkel (PPO).



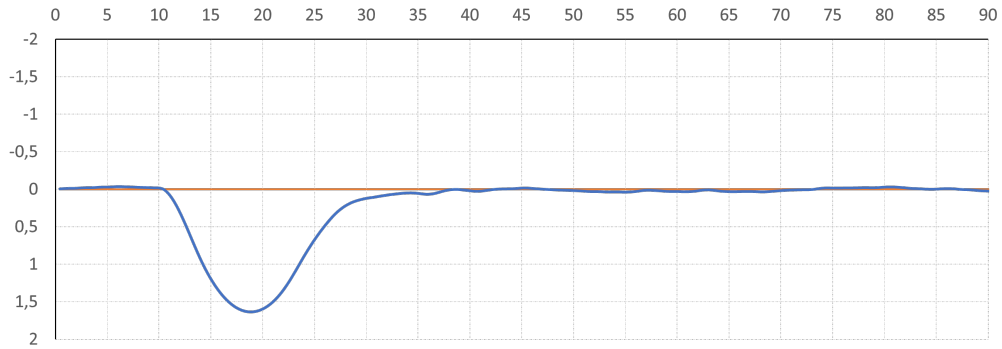
(b) Relativer Lenkwinkel (PPOΔ).

Abbildung 7.5: Die Graphen zeigen die maximal erreichten Schritte nach jeder Episode für den PPO/PPOΔ Algorithmus. Auf der x-Achse sind die Gesamtschritte des Trainings anstelle der Episoden dargestellt, während auf der y-Achse die Anzahl der Schritte angezeigt wird, die das Fahrzeug ohne Unterbrechung zurückgelegt hat. Dabei wurde ein Maximum von 6000 Schritten festgelegt. In Abbildung 7.5a wurde das Training mit dem absoluten Lenkwinkel durchgeführt, während in Abbildung 7.5b der relative Lenkwinkel verwendet wurde.

Die Abbildung 7.5 zeigt, wie lange der RL Algorithmus gebraucht hat, um eine gute Strategie zu finden. Wenn 6000 Schritte 12 Mal hintereinander ohne Unterbrechung gefahren wurden, gilt das Training als erfolgreich. Es ist zu sehen, dass der Algorithmus mit dem absoluten Lenkwinkel länger gebraucht hat als mit dem relativen Lenkwinkel. Doch ob das Fahrzeug zu weit nach links oder rechts fährt und somit die ein Meter Grenze überschreitet, kann auch zufällig sein. Daher kann hier keine genaue Bewertung über die Dauer des Trainings getroffen werden. In der Regel benötigt der Algorithmus mit dem absoluten oder dem relativen Lenkwinkel ungefähr gleich viel Zeit.



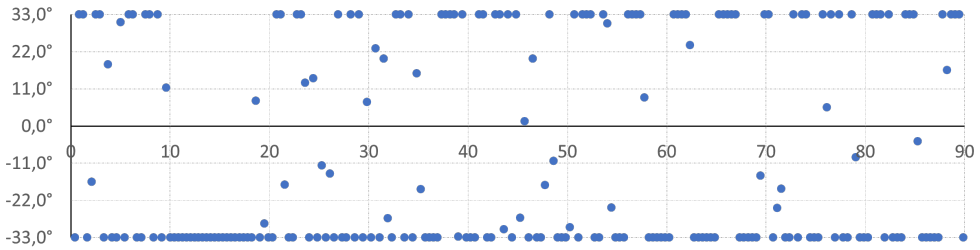
(a) Absoluter Lenkwinkel (PPO).



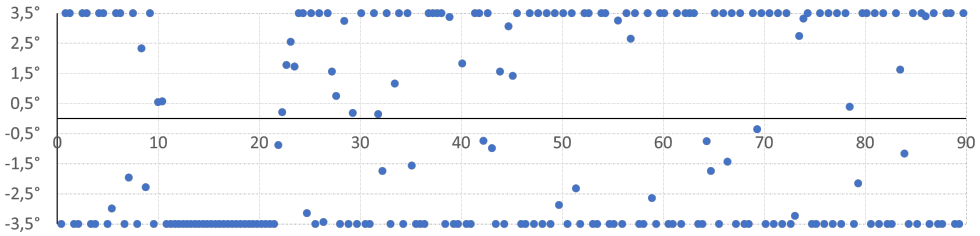
(b) Relativer Lenkwinkel (PPOΔ).

Abbildung 7.6: Die Graphen zeigen die Fahrt des Fahrzeugs auf einer 90 Meter langen, geraden Strecke für den PPO/PPOΔ Algorithmus. Sowohl auf der x- als auch auf der y-Achse sind die Werte in Metern eingetragen. Negative Werte liegen in Fahrrichtung links. Nach 10 Metern fährt das Fahrzeug nach rechts, bis es einen Abstand von ungefähr einem Meter für den absoluten Lenkwinkel und 1.25 Meter für den relativen Lenkwinkel erreicht. Anschließend kehrt es wieder zurück. Sowohl für den absoluten als auch für den relativen Winkel fährt das Fahrzeug nach ungefähr 30 Metern fast exakt auf der Fahrspurmitte.

Die Abbildung 7.6 zeigt die erste Teststrecke nach dem Training. Auf einer 90 Meter langen geraden Bahn fährt das Fahrzeug mit $30\frac{km}{h}$. Nach 10 Metern wird dem Ist-Lenkwinkel ein Offset von 25° hinzugefügt. Es ist zu sehen, dass das Fahrzeug nach rechts ausfährt und wieder zurückkehrt. Sowohl mit absolutem Lenkwinkel als auch mit dem relativen Lenkwinkel fährt das Fahrzeug nach dem Ausscheren fast gerade auf der Fahrspurmitte. Dies hängt mit der erlernten Strategie zusammen. Um die Aufgabe zu bewältigen, hat der Algorithmus gelernt, bei kleinen Abweichungen entweder vollständig nach links oder rechts zu lenken, wie in der Abbildung 7.7 zu sehen ist. Somit beträgt der Lenkwinkel sowohl für das Lenken nach links als auch nach rechts 33°. Das bedeutet, bei geringen Abweichungen übersteigt der gewählte Lenkwinkel den Offset und gleicht ihn somit aus. Dies geschieht bereits sehr nahe bei der Fahrspurmitte für alle Offsets, deren Beträge kleiner als 30.8° sind. Der Offset von 30.8° war der Winkel, bei dem das Fahrzeug den



(a) Absoluter Lenkwinkel (PPO).



(b) Relativer Lenkwinkel (PPOΔ).

Abbildung 7.7: Die Grafiken zeigen die Werte für den Lenkwinkel, die das künstliche neuronale Netz erzeugt hat, auf einer 90 Meter langen, geraden Teststrecke für PPO/PPOΔ. Auf der x-Achse sind die Werte in Metern und auf der y-Achse in Grad eingetragen. Negative Werte bedeuten, dass das Fahrzeug nach links fährt. In den ersten 10 Metern liegen die Werte sowohl für den absoluten als auch den relativen Lenkwinkel bei ihren minimalen oder maximalen Werten von $-33°$ und 33°, bzw. $-3.5°$ und 3.5°. Danach befinden sich die Werte beim Minimum, und nach 20 Metern kehren sie wieder zum Minimum oder Maximum zurück.

Weg nicht mehr zurückgefunden hat, da es zu weit weg gefahren ist und die Fahrspur aus seiner Sicht verschwand.

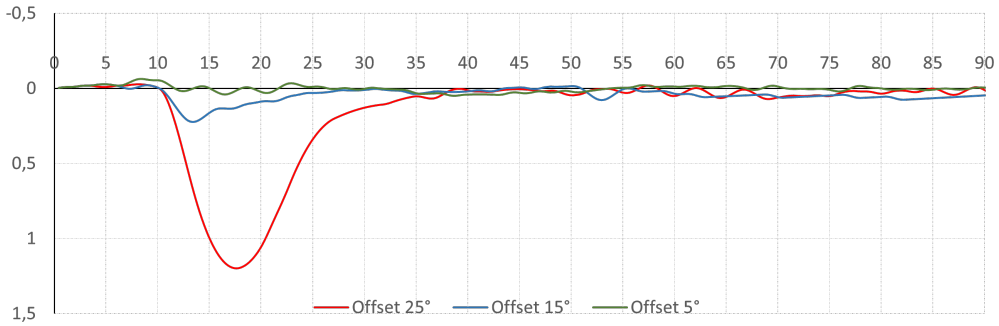


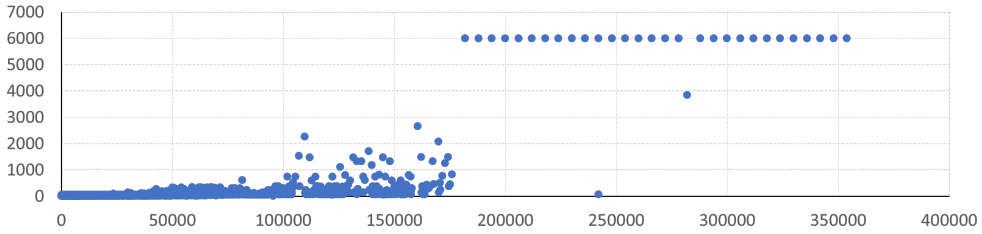
Abbildung 7.8: Der Graph zeigt die Fahrt des Fahrzeugs auf einer geraden, 90 Meter langen Teststrecke beim PPO Algorithmus mit mehreren Offsets von 25°, 15° und 5°. Sowohl auf der x- als auch y-Achse sind die Werte in Metern eingetragen. Positive Werte liegen in Fahrtrichtung rechts. Nach 10 Metern fährt das Fahrzeug nach rechts und zurück und bleibt bei allen Offsets dicht rechts neben der Fahrspurmitte. Die durchschnittlichen Abstände ab 40 Meter betragen für den Offset von 25° 0.03 Meter, für 15° 0.04 und für den Offset von 5° 0.01 Meter.

Die Abbildung 7.8 zeigt die Fahrt mit verschiedenen Offsets von 25°, 15° und 5° auf derselben Teststrecke mithilfe des herkömmlichen PPO Algorithmus. Bei allen Offsets liegen die durchschnittlichen Abstände zwischen der Fahrspurmitte und dem Fahrzeug unter 0.05 Metern. Da der Unterschied zwischen den Abständen zu gering ist, kann keine klare Aussage darüber getroffen werden, ob die Wahl des Offsets einen signifikanten Einfluss hat. Es war sogar zu beobachten, dass der Abstand bei 15° größer war als bei einem Offset von 25°. In Tabelle 7.2 sind die durchschnittlichen Lenkwinkel für die Offsets von 25°, 15° und 5° aufgeführt.

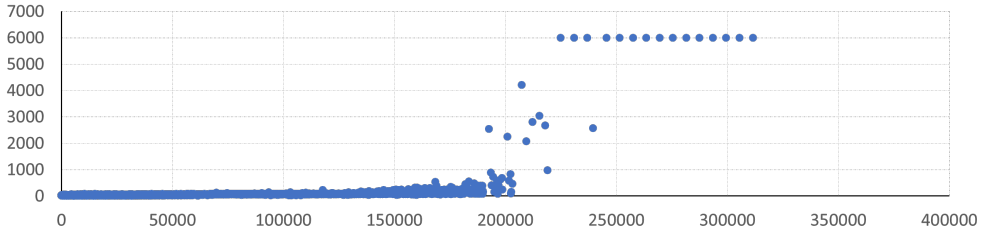
Tabelle 7.2: Offset und durchschnittlicher Lenkwinkel für PPO

Offset	25°	15°	5°
Lenkwinkel	−4.41°	−4.00°	−1.02°

7.3 DQN/DQNΔ



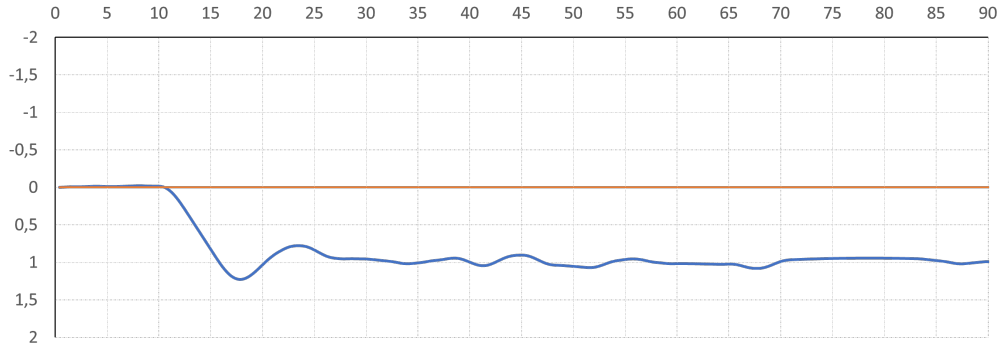
(a) Absoluter Lenkwinkel (DQN).



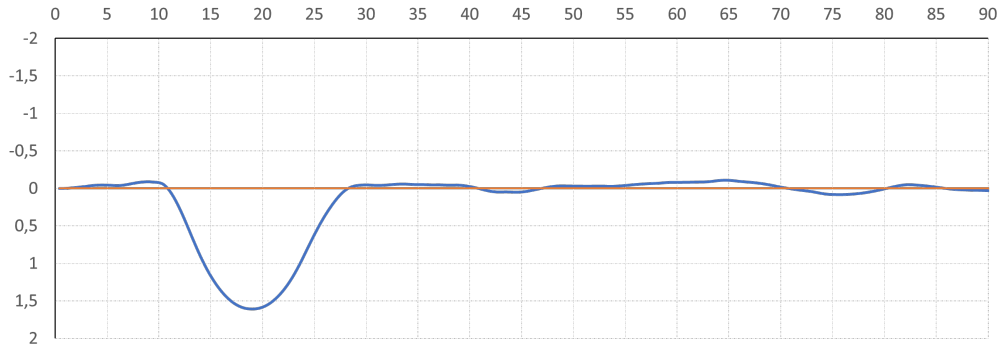
(b) Relativer Lenkwinkel (DQNΔ).

Abbildung 7.9: Die Graphen zeigen die maximal erreichten Schritte nach jeder Episode für den DQN/DQNΔ Algorithmus. Auf der x-Achse sind die Gesamtschritte des Trainings anstelle der Episoden dargestellt, während auf der y-Achse die Anzahl der Schritte angezeigt wird, die das Fahrzeug ohne Unterbrechung zurückgelegt hat. Dabei wurde ein Maximum von 6000 Schritten festgelegt. In Abbildung 7.9a wurde das Training mit dem absoluten Lenkwinkel durchgeführt, während in Abbildung 7.9b der relative Lenkwinkel verwendet wurde.

Die Abbildung 7.9 zeigt, wie lange der RL Algorithmus gebraucht hat, um eine effektive Strategie zu entwickeln. Wenn 6000 Schritte 12 Mal hintereinander ohne Unterbrechung durchgeführt wurden, gilt das Training als erfolgreich. Es ist erkennbar, dass der Algorithmus für den absoluten Lenkwinkel zwar früher begonnen hat, eine geeignete Strategie zu finden, jedoch insgesamt länger gebraucht hat, um die Aufgabe zu erfüllen. Bei etwa 2.8×10^5 Schritten hat er einmal die 6000 Schritte nicht erreicht. Dadurch hätten der Algorithmus für den absoluten und den relativen Lenkwinkel ungefähr gleich lange gebraucht.



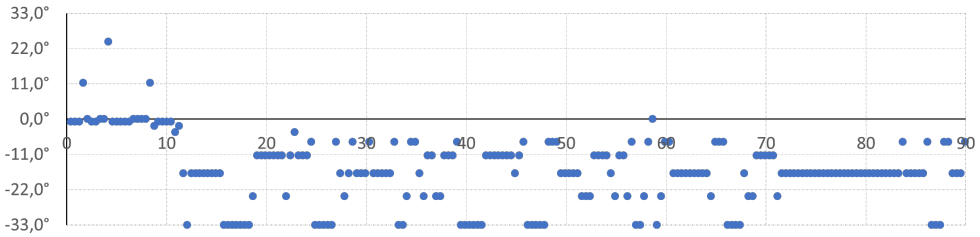
(a) Absoluter Lenkwinkel (DQN).



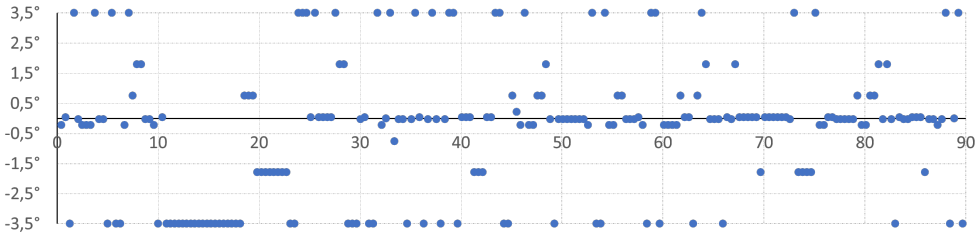
(b) Relativer Lenkwinkel (DQNΔ).

Abbildung 7.10: Die Graphen zeigen die Fahrt des Fahrzeugs auf einer 90 Meter langen, geraden Strecke unter Verwendung des DQN/DQNΔ Algorithmus. Sowohl auf der x- als auch auf der y-Achse sind die Werte in Metern angegeben. Negative Werte liegen in Fahrtrichtung links. Nach 10 Metern fährt das Fahrzeug nach rechts und erreicht einen Abstand von ungefähr 1.25 Metern für den absoluten Lenkwinkel und 1.5 Meter für den relativen Lenkwinkel. In Abbildung 7.10a bleibt das Fahrzeug mit einem durchschnittlichen Abstand von 0.98 Metern rechts neben der Fahrspurmitte. In Abbildung 7.10b kehrt das Fahrzeug nach 30 Metern zurück und fährt auf der Fahrspurmitte.

Die Abbildung 7.10 zeigt die erste Teststrecke nach dem Training. Auf einer 90 Meter langen geraden Bahn fährt das Fahrzeug mit $30\frac{km}{h}$. Nach 10 Metern wird dem Ist-Lenkwinkel ein Offset von 17° für den absoluten und 25° für den relativen Lenkwinkel hinzugefügt. Es ist erkennbar, dass das Fahrzeug nach rechts von der Fahrbahn abkommt. Beim absoluten Lenkwinkel bewegt sich das Fahrzeug durchschnittlich 0.98 Meter neben der Mitte der Fahrspur weiter, während es beim relativen Lenkwinkel zur Fahrspurmitte zurückkehrt und ungefähr auf ihr weiterfährt. Beim absoluten Lenkwinkel musste der Offset auf 17°



(a) Absoluter Lenkwinkel (DQN).



(b) Relativer Lenkwinkel (DQNΔ).

Abbildung 7.11: Die Grafiken zeigen die Werte für den diskreten Lenkwinkel, die das künstliche neuronale Netz erzeugt, auf einer 90 Meter langen, geraden Teststrecke für DQN/DQNΔ. Auf der x-Achse sind die Werte in Metern und auf der y-Achse in Grad eingetragen. Negative Gradwerte bedeuten, dass das Fahrzeug nach links fährt. Bis zu 10 Metern liegen die Werte für den absoluten Lenkwinkel meistens bei $-0.89°$ und $0.03°$. Nach 10 Metern verteilen sich die Werte diskret im negativen Bereich, wobei $-16.9°$ häufiger vorkommt. Für den relativen Lenkwinkel befinden sich die Werte zwischen 10 und 20 Metern auf ihrem Minimum, und nach 20 Metern verteilen sie sich diskret zwischen Minima und Maxima, wobei sie sich meistens bei $0.03°$ oder $-0.03°$ befinden.

reduziert werden, da 17° die Grenze darstellte, bei der das Fahrzeug noch zurück findet. In Abbildung 7.11a ist zu erkennen, wie das Fahrzeug aufgrund des Offsets gegensteuert. Die Werte sind diskret im negativen Bereich verteilt. Da der am häufigsten gewählte Lenkwinkel bei einem Durchschnittsabstand von 0.98 Metern zur Fahrspurmitte $-16.9°$ beträgt, gleicht das Fahrzeug in diesem Bereich den 17° Offset aus. In der Tabelle 7.3 ist aufgezeigt, wie oft ein bestimmter diskreter absoluter Lenkwinkel für DQN bei ei-

Tabelle 7.3: Häufigkeiten der gewählten Lenkwinkel ab 20 Metern für DQN

Lenkwinkel	$-33°$	$-24.1°$	$-16.9°$	$-11.3°$	$-7.13°$	$-4.12°$	$-0.03°$
Anzahl	28	17	65	32	25	1	1

Tabelle 7.4: Häufigkeiten der gewählten Lenkwinkel ab 30 Metern für DQNΔ

Lenkwinkel	−3.5°	−1.79°	−0.76°	−0.22°	−0.03°	0°	0.03°
Anzahl	18	9	1	17	35	2	28

Lenkwinkel	0.22°	0.76°	1.79°	3.5°
Anzahl	1	10	5	19

nem Offset von 17° ab 20 Metern gewählt wurde. Beim relativen Lenkwinkel sind die diskreten Werte nach dem Zurückfinden auf den gesamten Bereich von −3.5° bis 3.5° verteilt, wobei sie sich hauptsächlich bei den Lenkwinkeln 0.03° oder −0.03° befinden. In der Tabelle 7.4 ist aufgezeigt, wie oft ein bestimmter diskreter relativer Lenkwinkel für DQNΔ bei einem Offset von 17° ab 30 Metern gewählt wurde.

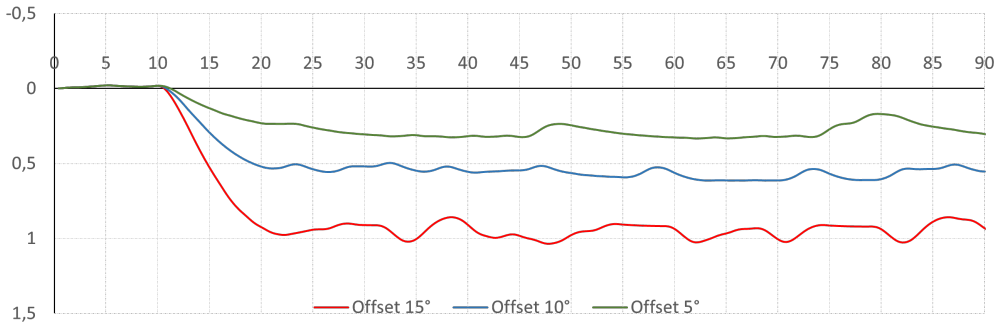


Abbildung 7.12: Der Graph zeigt die Fahrt des Fahrzeugs auf einer geraden, 90 Meter langen Teststrecke beim DQN Algorithmus mit mehreren Offsets von 15°, 10° und 5°. Sowohl auf der x- als auch y-Achse sind die Werte in Metern eingetragen. Positive Werte liegen in Fahrtrichtung rechts. Nach 10 Metern fährt das Fahrzeug nach rechts und bleibt bei einem Offset von 15° im Durchschnitt 0.95 Meter neben der Fahrspurmitte, bei 10° 0.56 Metern und bei 5° 0.29 Metern.

Die Abbildung 7.12 zeigt die Fahrt mit verschiedenen Offsets von 15°, 10° und 5° auf derselben Teststrecke mit dem normalen DQN Algorithmus. Auch hier fährt das Fahrzeug näher neben der Fahrspurmitte, je kleiner der Offset ist. Zudem ist eine stärkere und häufigere hin und her Lenkbewegung zu beobachten, je größer der Offset ist. In Tabelle 7.5 sind die durchschnittlichen Werte für den absoluten Lenkwinkel bei DQN aufgezeigt. Es ist zu erkennen, dass der durchschnittlich gewählte Lenkwinkel den gewählten Offset ausgleicht.

Tabelle 7.5: Offset und durchschnittlicher Lenkwinkel für DQN

Offset	17°	15°	10°	5°
Lenkwinkel	−17.9°	−17°	−9.96°	−5.16°

7.4 Vergleich

Tabelle 7.6: Dauer des Trainings der einzelnen Algorithmen

Algorithmus	**Gesamt**	**Schritt**
TD3	534 *min*	92.1 *ms*
TD3Δ	520 *min*	91.6 *ms*
PPO	286 *min*	31.3 *ms*
PPOΔ	271 *min*	31.5 *ms*
DQN	397 *min*	67.3 *ms*
DQNΔ	350 *min*	67.4 *ms*

In der Tabelle 7.6 sind die Zeiten aufgelistet, wie lange die einzelnen Algorithmen gebraucht haben, um eine gute Strategie zu finden, und wie lange sie für einen Schritt benötigen. Eine gute Strategie gilt dann als gefunden, wenn das Training erfolgreich beendet wurde. Das Fahrzeug ist also eine Stunde lang gefahren, ohne den maximalen Abstand von einem Meter zur Fahrspurmitte zu überschreiten.

Der Algorithmus PPO/PPOΔ hat schneller eine gute Strategie gefunden als der Algorithmus DQN/DQNΔ und der Algorithmus TD3/TD3Δ. Allerdings besteht seine Strategie darin, entweder ganz links oder ganz rechts zu fahren. Dadurch gab es kaum einen Unterschied zwischen dem absoluten und dem relativen Lenkwinkel. Der PPO/PPOΔ Algorithmus benötigte pro Schritt die geringste Rechenzeit.

Der Algorithmus TD3/TD3Δ hat die meiste Rechenzeit in Anspruch genommen und auch länger gebraucht, um eine gute Strategie zu finden, im Vergleich zu den anderen beiden Algorithmen. Die Steuerung war wesentlich gleichmäßiger als bei DQN/DQNΔ und PPO/PPOΔ. Da er auch Lenkwinkel zwischen den Minima und Maxima wählt, die kleiner sind als der Offset, ist der Unterschied zwischen der Fahrt mit dem absoluten und relativen Lenkwinkel deutlich erkennbar.

Der Algorithmus DQN/DQNΔ war langsamer als PPO/PPOΔ und schneller als TD3/TD3Δ. Allerdings ist aufgrund des diskreten Lenkwinkels die Steuerung größtenteils sprunghaft. Zusätzlich musste der Offset bei der Teststrecke für den absoluten Lenkwinkel reduziert

werden, da das Fahrzeug sonst Kreise fährt. Es ist jedoch zu erwähnen, dass der Offset von $\pm 25°$ oder auch $\pm 17°$ ein Extrembeispiel ist und in der realen Umgebung nicht auftreten würde, es sei denn, die Steuerung ist defekt.

Bei allen Algorithmen war die Variante mit dem relativen Lenkwinkel schneller fertig als die mit dem absoluten Lenkwinkel.

Tabelle 7.7: Durchschnittlicher und maximaler CTE mit Offset für alle RL Algorithmen

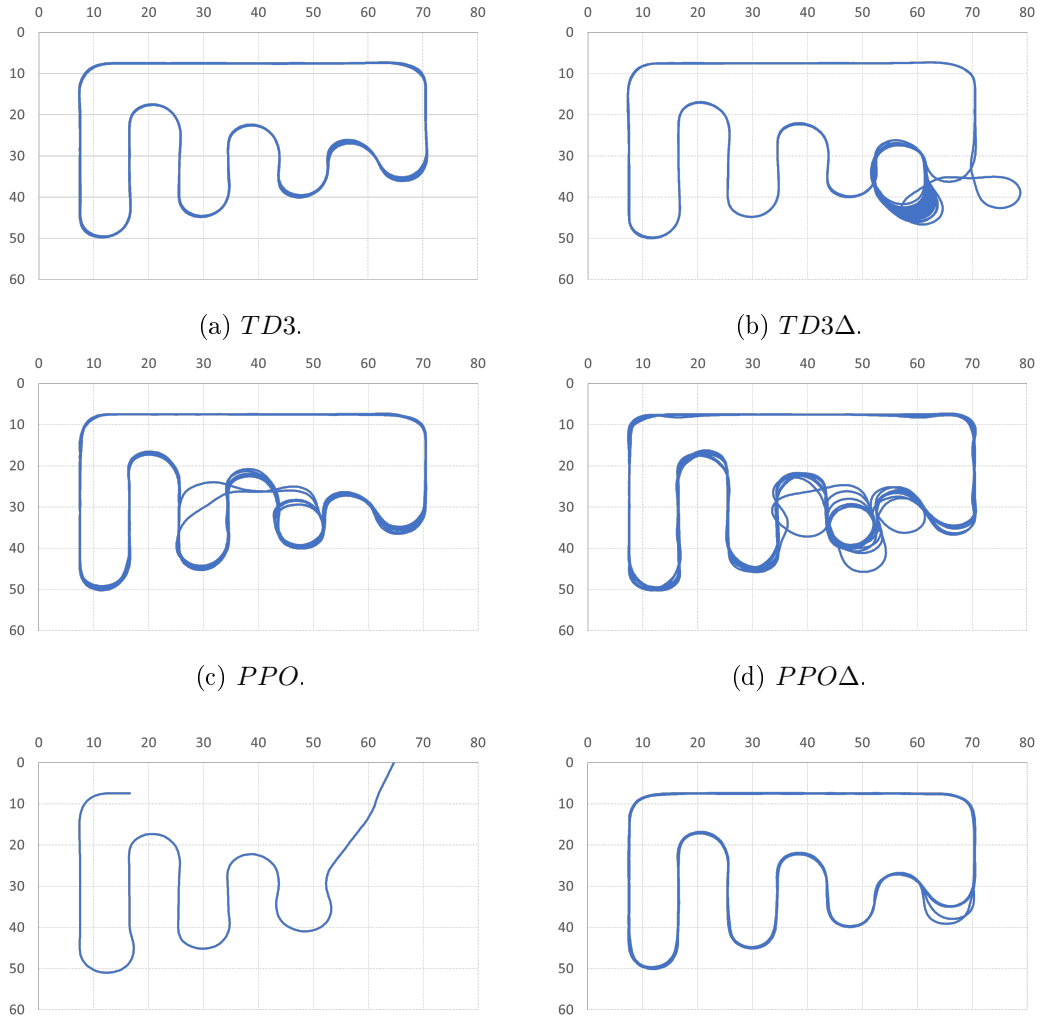
Algorithmus	**durchschn. CTE**	**max. CTE**
TD3	0.112 m	0.471 m
TD3Δ	0.06771 m	0.355 m
PPO	0.046 m	0.362 m
PPOΔ	0.092 m	1.437 m
DQN	–	–
DQNΔ	0.088 m	0.419 m

In der Tabelle 7.7 sind der durchschnittliche und der maximale CTE aufgelistet für die Fahrt auf der zweiten Teststrecke. Dabei wird der CTE am Anfang, wo das Fahrzeug raus- und zurückfährt, nicht berücksichtigt, um den maximalen CTE nicht zu verfälschen. Es wird ein Offset von $10°$ verwendet. TD3Δ hat im Vergleich zu TD3 einen kleineren durchschnittlichen und maximalen CTE und ist somit besser gefahren. DQN war auch bei mehreren Versuchen nicht in der Lage, die zweite Teststrecke zu beenden, und hat somit am schlechtesten abgeschnitten. Am besten hat der PPO mit dem absoluten Lenkwinkel abgeschnitten. Auch hier wurde gezeigt, dass die Strategie, bei kleinen Abweichungen gleich ganz nach links oder ganz nach rechts zu fahren, die bessere ist. PPOΔ hat am zweitschlechtesten abgeschnitten, da er im Gegensatz zu PPO auch Werte liefert, die kleiner als die maximale Lenkwinkel pro Schritt sind.

Tabelle 7.8: Durchschnittlicher und maximaler CTE ohne Offset für alle RL Algorithmen

Algorithmus	**durchschn. CTE**	**max. CTE**
TD3	$0,247$ m	$1,130$ m
TD3Δ	–	–
PPO	–	–
PPOΔ	–	–
DQN	–	–
DQNΔ	$0,277$ m	$3,676$ m

In Tabelle 7.8 sind der durchschnittliche und der maximale CTE für die Fahrt auf der dritten Teststrecke, der anspruchsvollsten Strecke ohne jeglichen Offset, aufgeführt. Ein



(a) $TD3$. (b) $TD3\Delta$.

(c) PPO. (d) $PPO\Delta$.

(e) DQN. (f) $DQN\Delta$.

Abbildung 7.13: Hier sind die Fahrtverläufe auf der dritten Teststrecke für TD3, TD3Δ, PPO, PPOΔ, DQN und DQNΔ dargestellt. Auf der x- und y-Achse sind die Werte in Metern angegeben.

fehlender Wert zeigt an, dass das Fahrzeug die Teststrecke nicht bewältigt hat. Wie bereits aus der Tabelle ersichtlich ist, konnten lediglich zwei Algorithmen die dritte Teststrecke erfolgreich beenden. In der Abbildung 7.13 ist deutlich erkennbar, dass die Algorithmen TD3Δ, PPO, PPOΔ und DQN auf der Strecke mit engeren Kurven im Vergleich zur Trainingsstrecke Schwierigkeiten haben, das Fahrzeug stets auf der Fahrspur zu halten. Dabei

hat DQN überhaupt keinen Weg zurück zur Fahrstrecke gefunden. Dennoch wird auch für die anderen Algorithmen die Strecke als nicht bestanden gewertet, wenn das Fahrzeug einen Kreis zieht und an der Strecke neu ansetzt. Die besten Ergebnisse erzielten die Algorithmen TD3 und DQNΔ. Es ist erkennbar, dass alle Algorithmen Schwierigkeiten mit der letzten Kurve hatten, doch TD3 konnte sie am besten bewältigen.

8 Fazit

In dieser Arbeit wurde erklärt, was RL und der Sim-to-Real Gap sind. Es wurden die einzelnen Algorithmen TD3, PPO und DQN beschrieben und die Architekturen, die in dieser Arbeit für diese einzelnen RL Algorithmen verwendet wurden, vorgestellt. Der Ansatz, um den Sim-to-Real Gap zu überbrücken, wurde ebenfalls vorgestellt. Nach dem Training mit diesem Ansatz und ohne diesen Ansatz wurden Tests durchgeführt. Sowohl die Ergebnisse des Trainings als auch die Ergebnisse der Tests wurden in dieser Arbeit festgehalten und erklärt. Es hat sich gezeigt, dass der Unterschied zwischen der Simulation und der realen Umgebung in Bezug auf die Lenksteuerung unter Verwendung eines relativen Lenkwinkels ignoriert werden kann. Mit dem relativen Lenkwinkel waren alle Algorithmen in der Lage, den Weg zur Fahrspurmitte zurückzufinden, nachdem ein Offset zum Ist-Lenkwinkel hinzugefügt wurde. Zusammenfassend bietet dieser Ansatz eine robuste Methode, um den Sim-to-Real Gap zu überwinden.

Literaturverzeichnis

[1] S. Pareigis and F. Maaß, "Robust Neural Network for Sim-to-Real Gap in End-to-End Autonomous Driving," in *International Conference on Informatics in Control, Automation and Robotics 2022*. SciTePress, 2022, pp. 113–119.

[2] M. Bojarski, C. Chen, J. Daw, A. Değirmenci, J. Deri, B. Firner, B. Flepp, S. Gogri, J. Hong, and L. Jackel, "The NVIDIA pilotnet experiments," 2020.

[3] D. L. Poole and A. K. Mackworth, "Artificial Intelligence: Foundations of Computational Agents," in *Artificial Intelligence: Foundations of Computational Agents*, 2nd ed. Cambridge University Press, 2017, pp. 458–462.

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," 2013.

[5] C. J. C. H. Watkins, "Learning from delayed rewards," 1989, publisher: King's College, Cambridge United Kingdom.

[6] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms."

[7] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," 2015.

[8] S. Fujimoto, H. Hoof, and D. Meger, "Addressing Function Approximation Error in Actor-Critic Methods," in *Proceedings of the 35th International Conference on Machine Learning*. PMLR, 2018, pp. 1587–1596.

[9] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2015.

[10] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 3803–3810.

[11] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox, "Closing the sim-to-real loop: Adapting simulation randomization with real world experience," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8973–8979.

[12] P. Agrawal, A. V. Nair, P. Abbeel, J. Malik, and S. Levine, "Learning to poke by poking: Experiential learning of intuitive physics," vol. 29, 2016.

[13] Cloud-Computing-Dienste | Google Cloud. [Online]. Available: https://cloud.google.com/

[14] Stable-Baselines3 Docs - Reliable Reinforcement Learning Implementations — Stable Baselines3 2.0.0 documentation. [Online]. Available: https://stable-baselines3.readthedocs.io/en/v2.0.0/

[15] R. Daniel. tinycarlo. [Online]. Available: https://github.com/danielriege/tinycarlo

Erklärung zur selbstständigen Bearbeitung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort	Datum	Unterschrift im Original