Multi-Target Regression zur Approximation der Spielphysik von Rocket League

Frederik Rohkrähmer

TU Dortmund

8. September 2022

Inhalt

Rocket League

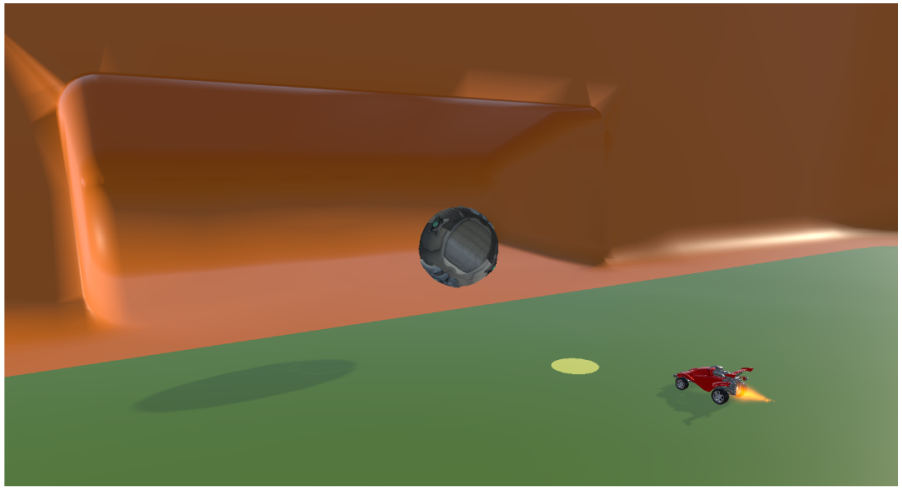
- Autofußball mit Raketenantrieb
- 2015 veröffentlicht von Psyonix
- Seit 2019 im Besitz von Epic Games
- Entwickelt in Unreal Engine 3

Spielprinzip

- Fußball mit Autos
- 1vs1 bis 4vs4
- Raketenantrieb
- (Doppel-)Sprünge
- Rollen
- Wand fahren

Projektgruppe 642

- Verteiltes Deep Reinforcement Learning zum Trainieren von Game AI in Rocket League
- Training in Unity-Reimplementation
- Reverse Engineering
- Sim-to-Sim Transfer
- Paper veröffentlicht
- Sim-to-Sim Striker schießt nur 75% der Tore



Idee

- Sim-to-Sim Idee vielversprechend, aber
- Unity Implementierung nicht gut genug!
- Neuer Ansatz:
- Supervised Learning statt Reverse Engineering

Problemstellung

Gegeben ein Spielzustand und ein Input, bestimme den Spielzustand des nächsten Frames so wie er in Rocket League folgen würde.

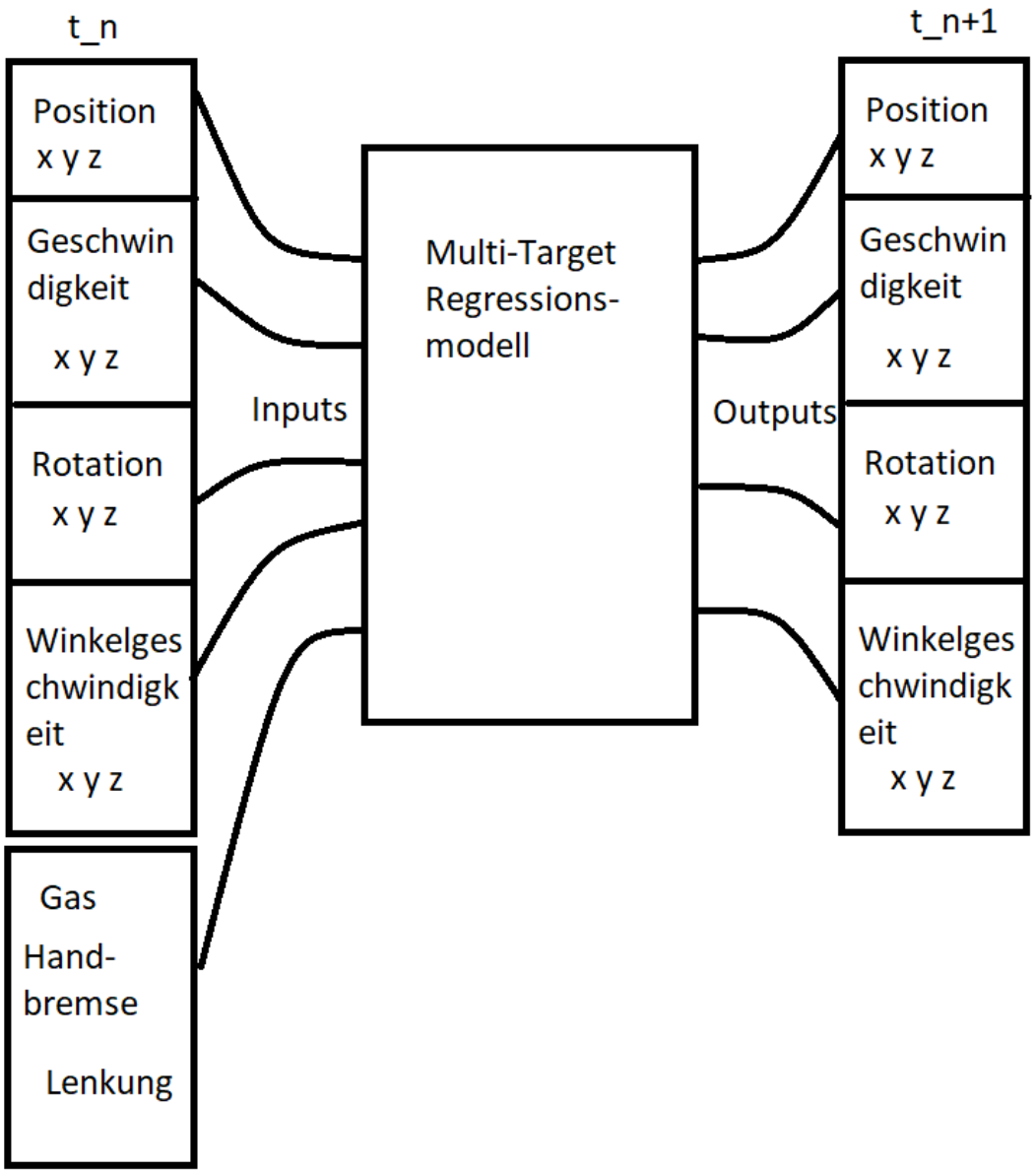
Ziel

Erstelle ein Multi-Target Regressionsmodell, welches dieses Problem löst.

Multi-Target Regression

- Klassifikation: Diskreter Output
- Regression: Kontinuierlicher Output
- Spielzustand enthält mehrere kontinuierliche Größen (Targets)
- Position, Geschwindigkeit, Rotation, Winkelgeschwindigkeit
- Vorhersage mehrerer kontinuierlicher abhängiger Variablen:
- Multi-Target Regression

Multi-Target Regression



Datenbeschaffung

- Benötigt:
- Aufeinanderfolgende Spielzustände
- mit annotierten getätigten Inputs
- die möglichst viele verschiedene Spielsituationen abdecken
- **So einen Datensatz gibt es *noch* nicht!**

Datengenerierung

- Mehrere Möglichkeiten Spielzustände auszulesen:
- RLBot
- RLGym
- BakkesMod

RLGym

Spielerinputs neu simulieren

- RLGym am praktikabelsten
- Beschleunigbar
- Parallelisierbar
- Aber nur Windows
- Vorgehen:
- In jedem Frame Input über RLGym ausführen lassen
- Spielzustand aufzeichnen

Spielerinputs generieren

- Inputsequenzen benötigt die die Anforderungen erfüllen
- Händisch erstellen zu aufwändig
- Eigene Inputs aufzeichnen auch zu aufwändig
- Lösung:
- Inputs aus Replays

Rocket League Replays

- Komprimierte Datei die Spielverlauf speichert (.replay)
- Replays speichern KEINE Inputs
- Aber:
- Inputs lassen sich emulieren
- `https://github.com/oxrock/TrainingDataExtractor`
- Replays massenhaft online erhältlich z.B. auf
- `https://ballchasing.com`

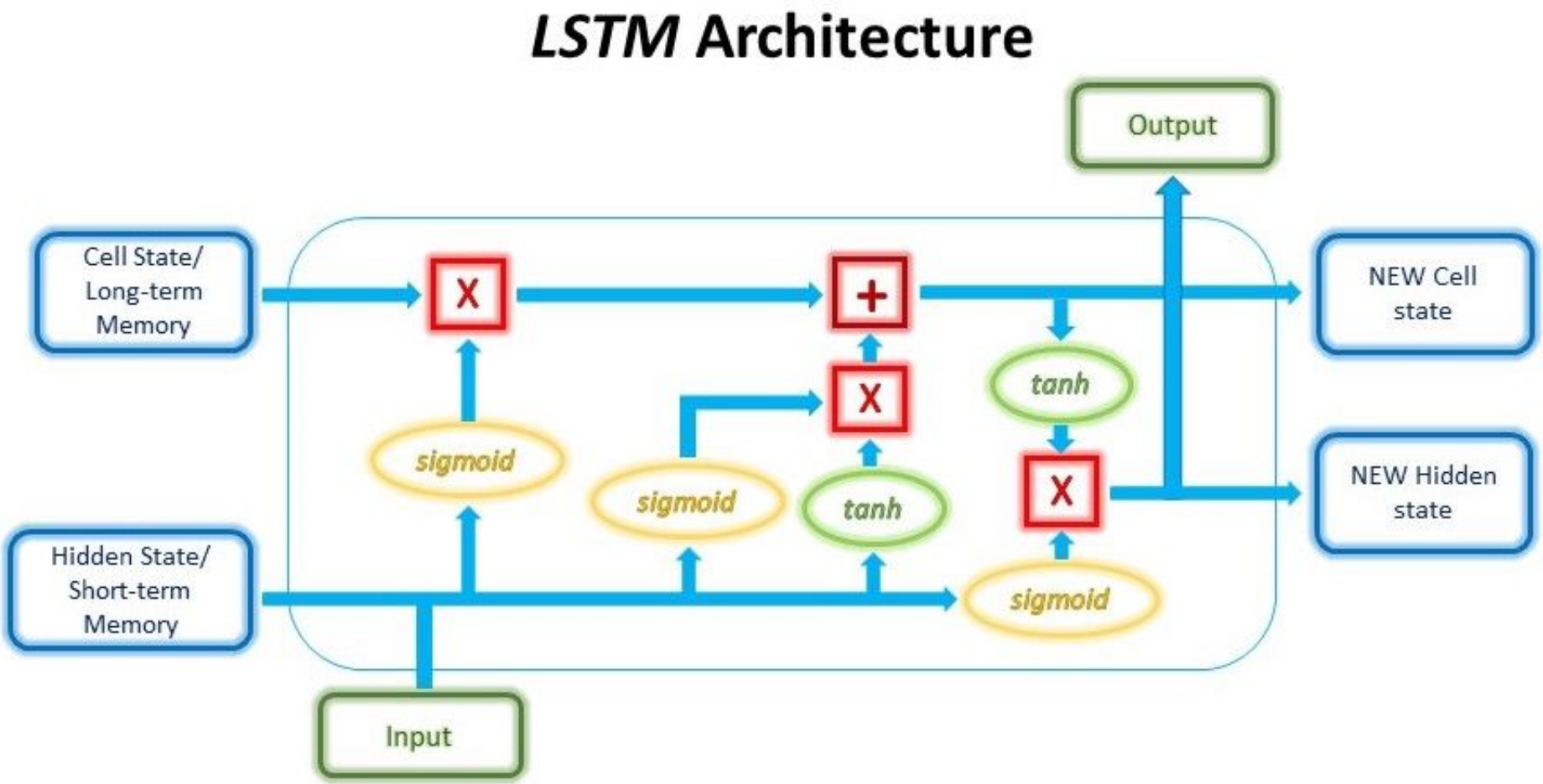
Inputsequenzen aufbereiten

- Replays speichern Spielzustände mit ca. 25fps
- Rocket League läuft mit 120fps
- Replays in mehrere Sequenzen zerteilen
- Beginn: Anstoß, Ende: Tor
- 25fps Sequenzen zu 120fps interpolieren
- Mehrere Möglichkeiten

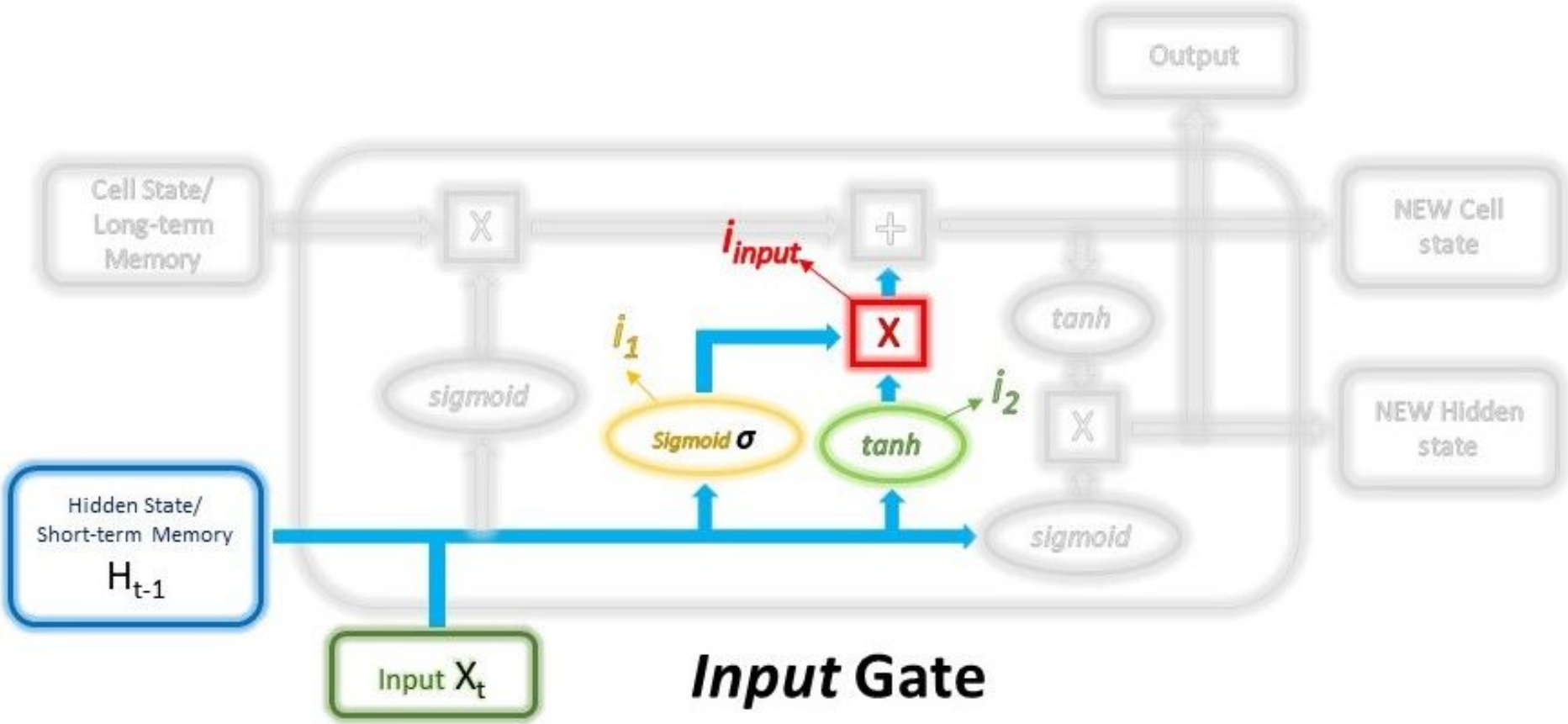
Beschaffenheit des Datensatzes

- Zunächst nur 1vs1 Spiele betrachten
- Spielzustand ist nicht gedächtnislos!
- Sprungverhalten von versteckten Variablen abhängig
- Daher:
- Multi-Target Regressionsmodell mit Gedächtnis wählen
- Geeignet:
- Long Short Term Memory (LSTM)

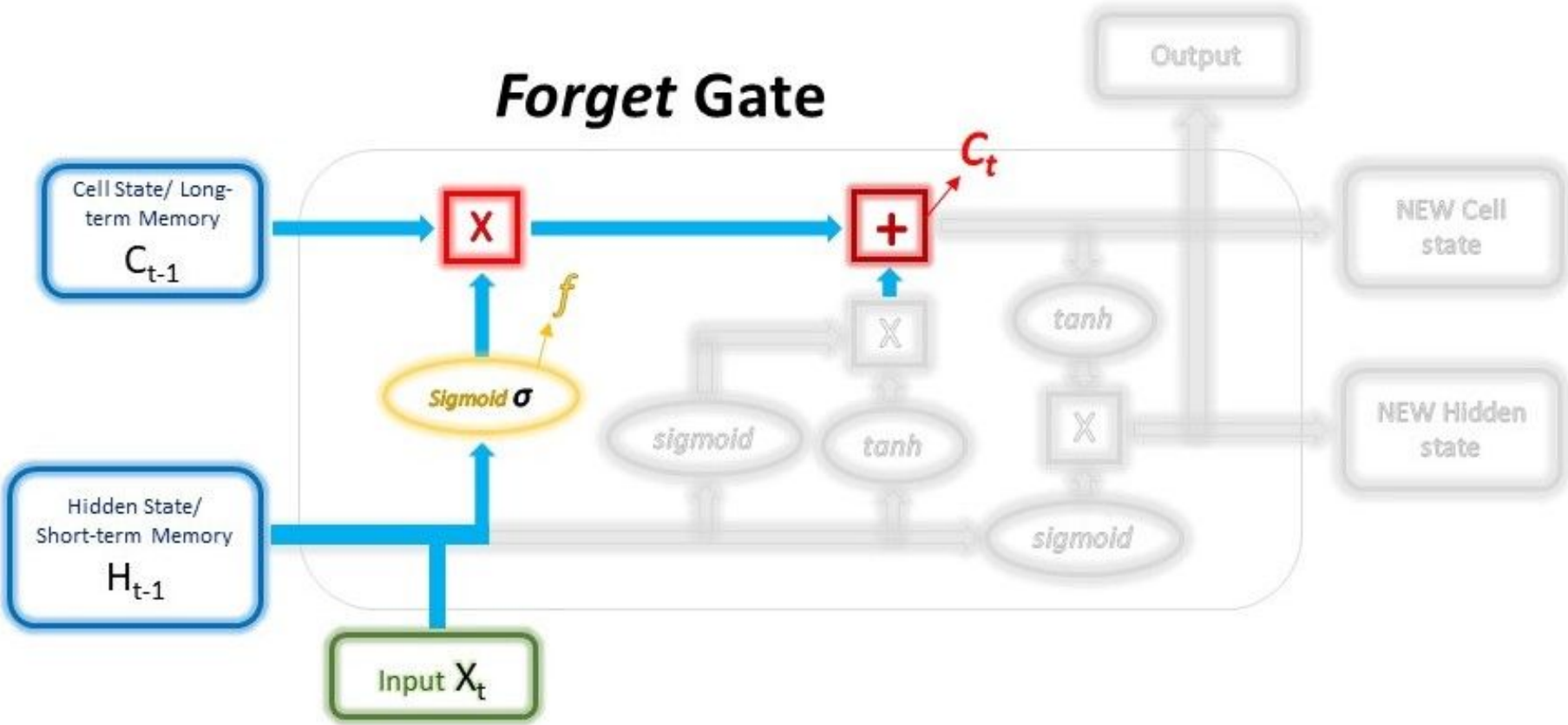
Long Short Term Memory



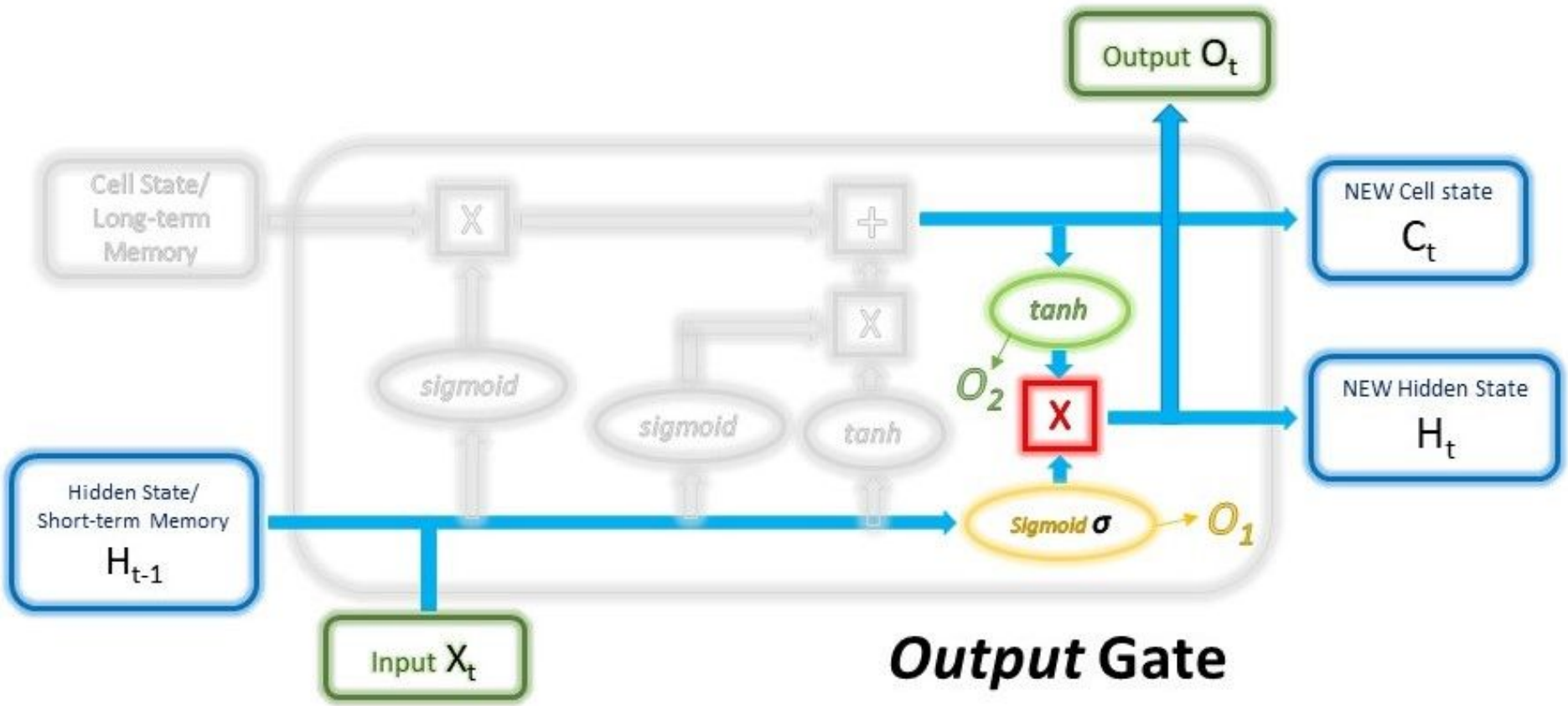
Long Short Term Memory



Long Short Term Memory



Long Short Term Memory



Zusammenfassung

- .replay Dateien besorgen
- Emulierte Inputs generieren
- Auf 120 fps interpolieren
- Mit RLGym in Rocket League ausführen
- Spielzustände aufzeichnen
- LSTM trainieren

Ausblick

- Wenns gut läuft:
- Erweitern auf 2v2, 3v3, 4v4
- Agenten mit dem Modell trainieren lassen