Abstract

Für eine Bachelorarbeit an der
Hochschule Furtwangen
Fakultät Digitale Medien

Arbeitstitel:

Computer Vision als Werkzeug für die Manipulation von Videospielen – Konzeption und Entwicklung eines Cheats mit Image Recognition und Reinforcement Learning

Alexander Reiprich
Bregstraße 8
78120 Furtwangen im Schwarzwald
alexander.reiprich@hs-furtwangen.de
Matrikel-Nr.: 263006

Videospiele bilden einen wichtigen Aspekt der Unterhaltung in der heutigen Gesellschaft. Spieleentwickler verbringen viel Zeit damit, Spiele zu entwickeln und zu testen, um dem Konsumenten schlussendlich das ansprechendste Produkt zu liefern. Besonders bei Spielen mit einer Online-Multiplayer-Funktionalität ist ein nicht unwichtiger Teil hiervon die Entwicklung von Maßnahmen gegen die Manipulation durch andere Programme. Diese können in großem Maße das Spielerlebnis anderer Spieler beeinträchtigen, in dem sie dem Nutzer unfaire Vorteile geben.

Während ein Großteil der sogenannten Cheating-Software direkt auf die Spieldateien zugreift, gibt es auch andere Möglichkeiten durch ein Programm auf das Spiel Einfluss zu nehmen. In dieser Arbeit soll die Technologie der Computer Vision im Hinblick auf Cheating in Videospielen untersucht werden. Als Untersuchungsgegenstand liegt hierbei das 2007 erschienene Rhythmusspiel „osu!“ zugrunde, welches extern durch ein Programm gesteuert werden soll.

In „osu!“ spielt der Spieler verschiedene Level, auch „Beatmaps“ genannt, welche von anderen Nutzern erstellt wurden. Dabei bewegt er den Cursor und klickt im Takt der Musik auf auftauchende Kreise. Die Schwierigkeit des Spiels ist dabei von den gespielten „Beatmaps“ abhängig. Je schwerer das Level, desto kleiner werden die Kreise, oder desto schneller tauchen sie auf. Hier wird die Geschicklichkeit und die Reaktionsfähigkeit des Spielers auf die Probe gestellt, da nur durch das Klicken im richtigen Moment die höchste Punktzahl erreicht werden kann.

Eine detailliertere Erklärung des Spielprinzips sowie weitere Informationen können hier gefunden werden.

Die Arbeit soll in einen theoretischen sowie praktischen Teil untergliedert werden, wobei der Fokus der Arbeit auf den praktischen Teil gerichtet sein soll. Im Theorieteil wird sich mit dem generellen Problem Cheating in Videospielen auseinandergesetzt. Hierbei soll behandelt werden, wieso ein Spieler zu solchen Methoden greift, wie diese aussehen können und wie Spieleentwickler versuchen, das Eingreifen dieser Cheats auf ihr Spiel zu verhindern.

Im praktischen Teil der Arbeit soll ein Cheat entwickelt werden, welcher das bereits angesprochene Rhythmusspiel „osu!“ spielen soll. Dieses Spiel wurde gewählt, da es sich für ein solches Vorhaben durch zwei Faktoren besonders gut eignet: Der wichtigste Aspekt ist das Gameplay – es ist simpel, einfach zu verstehen, besitzt aber dennoch eine

hohe Lernkurve. Ebenso ist das Aussehen der Spielelemente anpassbar, wodurch man das Gameplay optimal für Image Recognition ausrichten kann, sodass ein Programm wenig Schwierigkeiten hat, den Vorgang auf dem Bildschirm zu interpretieren.

Die technische Umsetzung soll in Python mit Unterstützung verschiedener Libraries erfolgen. Im Mittelpunkt soll dabei die Verwirklichung mit einem Reinforcement Learning Ansatz stehen.

Für diesen Ansatz sind zwei Komponenten essentiell: Das Environment und der Agent. Das Environment bildet die Umgebung ab, in der sich der Agent befindet. Die Umsetzung dieser Umgebung ist in diesem Fall etwas speziell, da es sich bei osu! um ein Echtzeit-Spiel handelt. Grundlage für die Erstellung des Environments wird OpenAI Gym, bzw. der indirekte Nachfolger Gymnasium sein. Da eine eigene Umgebung für das Spiel erstellt werden muss, welches auch in Echtzeit funktioniert, wird die Library Real-Time Gym genutzt, welche eine Erweiterung des OpenAI Gym/Gymnasium ist. Real-Time Gym erlaubt eine Einschränkung der Sende-/Empfangszeit nach dem sogenannten Delayed Markov Decision Process, was in der OpenAI Library standardmäßig nicht möglich ist. Neben dem Environment ist der Agent der zweite wichtige Teil des Projektes. Dieser arbeitet in der vorher erstellten Umgebung, interpretiert dort die observierten Inputs und führt dementsprechende Aktionen aus. Die Library, welche sich um den Agent kümmern soll, ist PyTorch. Für dieses Programm soll ein Deep-Q-Network genutzt werden, um dem Agent das Spiel beizubringen.

Es gibt einige essentielle Methoden innerhalb der einzelnen Libraries, welche für die Realisierung genutzt werden sollen. Innerhalb der Library Gymnasium spielen folgende Methoden eine besondere Rolle: gymnasium.make() für die Erstellung des Environments, environment.step() für das Fortschreiten innerhalb des Environments, sowie environment.reset() um das Vorgehen zurückzusetzen. Optional ist environment.render(). Dieses erzeugt eine grafische Oberfläche, auf welchem man die Umgebung visuell darstellen kann. Ob dies tatsächlich benötigt wird, ist jedoch noch offen, da man das Geschehen noch auf andere Wege, wie z.B. einem zweiten Bildschirm verfolgen kann.

Für den Agent gibt es innerhalb von PyTorch mehrere Funktionen, welche in mehrere selbstgeschriebene Klassen aufgeteilt werden. Zum einen benötigt man eine Klasse, welche sich um die Verwaltung des sog. Experience Replay Memory kümmert. Dies ist eine Funktion des Agents, in welchem die Erfahrungen durch vorangegangene Actions gespeichert werden sollen, welche dann zufällig zum Training der AI genutzt werden

können. Dies verhindert das Erstellen von zufälligen Zusammenhängen, welche beim Lernprozess auftreten können. Da diese Klasse selbst gebaut werden muss, basiert sie auf lokalen Variablen wie z.B. einem Memory-Array, welche in Methoden wie push() zum Hinzufügen einer Memory, oder sample() zum zufälligen Auswählen einer Memory, manipuliert werden. Die zweite benötigte Klasse ist das Q-Network selbst. Dieses wird in einer __init__-Funktion mithilfe mehrerer PyTorch-Funktionen, wie z.B. pytorch.nn.Linear() aufgesetzt und dann mit z.B. pytorch.nn.Flatten() bearbeitet. Ebenso wird eine forward-Funktion benötigt, welche die nächste ausgeführte Aktion bestimmt. Dabei wird der Großteil der Arbeit, anders als bei dem Experience Replay Memory, von PyTorch selbst erledigt, sodass man nur einen Aufruf auf das im __init__-Teil erstellte Netzwerk ausführen muss.

Da osu! grob runtergebrochen nur eine Art Geschicklichkeitsspiel ist, soll das Programm unter dem Konzept des „model-free reinforcement learning“ arbeiten, es soll also bloß der momentane Stand des Spiels betrachtet werden, und nicht noch etwaige Vorhersagen von zukünftigen Stadien in welchen sich das Spiel befinden kann, wie im „model-based“ Ansatz.

Das Ergebnis der Arbeit soll ein vollständiges Programm sein, mit welchem man einen beliebigen Song in „osu!“ mit 100 % Genauigkeit abschließen kann.

Dabei ist natürlich die Möglichkeit, dass das Vorhaben aufgrund von technischen Limitierungen nicht in diesem Rahmen erfüllt werden kann, nicht auszuschließen. Durch die geringe Zeit zwischen den Inputs und der generellen Geschwindigkeit des Spiels kann es sein, dass „Beatmaps“ ab einer gewissen Schwierigkeit nicht mehr von dem Programm zu bewältigen sind. In diesem Falle soll versucht werden, das Programm möglichst gut zu optimieren und auf den Erkenntnissen der Entwicklung basierend Annahmen zu treffen, inwiefern unterschiedliche Technologien oder Herangehensweisen an dieses Problem andere Ergebnisse erzielen würde.