NEUE
KANTONSSCHULE
AARAU

Nachzeichner KI

Maturarbeit

Ian Wasser, Robin Steiner

Oktober 2022

Betreut durch: Nicolas Ruh

NKSA

Inhaltsverzeichnis

Kapitel 1

Methode

Die Methode dieser Untersuchung besteht darin, die in der Fragestellung beschriebene künstliche zu entwickeln und die Leistung davon auszuwerten. Die Diskusssion dieser Resultate führt schliesslich zu einer Antwort auf die Fragestellung. Die Entwicklung der künstlichen Intelligenz besteht aus zwei Teilen. Der eine Teil umfasst die Definition der Kriterien, nach denen die Leistung der künstlichen Intelligenz evaluiert wird. Der zweite Teil umfasst die Entwicklung der künstlichen Intelligenz, zusammen mit verschiedenen Variationen. Die Variationen haben jeweils einen unterschiedlichen Fokus auf die definierten Kriterien. Die Auswertung bezieht sich ebenfalls auf die definierten Kriterien. Die Leistung der künstlichen Intelligenz wird dabei in einer Testumgebung für das Zeichnen von verschiedenen Arten von Strichbildern gemessen.

1.1 Grundprogramm

Die künstliche Intelligenz ist abhängig von den Kriterien, die dessen Leistung definieren (siehe Eval). Mit anderen Worten trainiert die künstliche Intelligenz auf diese Kriterien. Das Ziel des Grundprogrammes ist, eine Trainingsumgebung für die künstliche Intelligenz bereitzustellen. Dieses Grundprogramm ist unabhängig von einem spezifischen Kriterium und kann stattdessen auf ein ausgewählte Kriterium trainiert werden. Reinforcement Learning Modelle mit einer momentan undefinierten Reward Function haben diese Eigenschaften. Die Implementierung eines spezifischen Kriteriums in die Reward Function ermöglicht das Training auf dieses Kriterium.

1.1.1 Doodle-SDQ als Basis

Das Reinforcement Learning Modell von dem Grundprogramm basiert auf Doodle-SDQ von Das Grundprogramm basiert auf dem Reinforcement Algorithmus aus dem Programm Doodle-SDQ (Tao Zhou et Al (2018) [1]).

Ausser dem Reinforcement Learning Modell, hauptsächlich bezogen auf die Form des Inputs, des Outputs und den hidden Layers, sind keine weiteren Konzepte von Doodle-SDQ übernommen. Die relevanten Anpassungen von Doodle-SDQ für das Grundprogramm dieser Arbeit sind nachfolgend erläutert.

Bei der Umgebung handelt es sich, wie bei Doodle-SDQ, um eine Zeichenfläche, worauf sich der Agent frei bewegen kann. Die Ziffern, die während dem Training nachgezeichnet werden sollen, stammen aus dem MNIST Datenset und haben somit eine Grösse von 28×28 Pixeln. Die Fläche, worauf sich der Agent bewegen kann, hat somit auch eine Grösse von 28×28 Pixeln. Die 'globale' Eingabe in das Neuronale Netz ändert sich bis auf diese neue Grösse der Bilder nicht.

Die Bilder, wie auch die Zeichenfläche, haben die Datenstruktur einer Bitmap. Es handelt sich also um eine 28×28 Matrix, wobei jedes Element eine Null oder eine Eins ist. Eine Null repräsentiert einen schwarzen (nicht gezeichneten) Pixel an dieser Stelle im Bild und eine Eins einen weissen (gezeichneten) Pixel.

Die lokale Eingabe, also das nahe Umfeld um den Agenten schrumpft von 11×11 Pixeln auf 7×7 Pixeln. Somit schrumpft gleichzeitig der Actionspace des Agenten von $2 \cdot 11 \cdot 11 = 242$ Aktionen auf $2 \cdot 7 \cdot 7 = 98$ Aktionen. Das bedeutet für den Agenten, dass er sich pro Schritt um maximal drei Pixel von seiner Position wegbewegen kann. Diese Bewegung kann der Agent entweder zeichnend oder nicht zeichnend ausführen.

Falls der Agent die Aktion zeichnend ausführt, zieht das Programm einen Strich zwischen der alten und der neuen Position. Mit anderen Worten werden alle Pixel in der Zeichenfläche zwischen den beiden Positionen weiss. Der Strich hat eine festgelegte Breite von 3 Pixeln. Am Anfang jeder Episode, also bei jeder neuen Ziffer, die gezeichnet werden soll, startet der Agent in einer zufälligen Position im nicht zeichnenden Zustand. Am Anfang jeder Episode ist die Zeichenfläche leer, also vollkommen Schwarz.

Aktionen des Agenten, die ihn über die vorgegebene Zeichenfläche hinaus positionieren würden, sind nicht zulässig. Diese Aktionen können vom Agenten nicht gewählt werden und ihr optimaler Q-Wert ist als 0 definiert. Das hat zur Folge, dass nach der Trainingsphase die allermeisten unzulässigen Aktionen einen tiefen Q-Wert haben. Das senkt die Wahrscheinlichkeit, dass der Agent versucht, eine unzulässige Aktion auszuführen.

1.1.2 Präparierung der Daten und Training

Das in dieser Arbeit verwendete MNIST Datenset besteht aus 42000 Bildern von handgeschriebenen Zahlen zwischen Null und Neun. Die Bilder im Datenset sind als Bitmap dargestellt, wobei jedes Element (jeder Pixel) einen

Wert zwischen 0 und 255 annimmt. Die Zahl räpresentiert einen Punkt auf dem Spektrum von Grautönen, wobei 0 Schwarz ist und 255 Weiss. Diese Graustufen werden entfernt. Jeder Pixel mit einem Wert über 0 übernimmt den Wert 1. So stimmen die Bilder mit den Zeichnungen, die der Agent produzieren kann, überein.

Die Trainingsdaten bestehen aus 36000 der 42000 Bilder im Datenset. Die restlichen 6000 Bilder sind für die Testphase aufgehoben. Das Grundprogramm trainiert mit 4000 Bildern, von denen jede Zahl von Null bis Neun 400 Bilder ausmacht. Die restlichen Bilder in den Trainingsdaten sind für mögliche Erweiterungen aufgehoben. Der Agent zeichnet jedes der 4000 Bilder insgesamt drei mal. Mit anderen Worten Läuft die Trainingsphase für 3 Epochen mit jeweils 4000 Episoden. Der Agent macht 64 Schritte pro Episode. Er kann sich also pro Zeichnung 64 mal bewegen. Insgesamt trainiert der Algorithmus somit auf der Basis von $3 \cdot 4000 \cdot 64 = 780'000$ Schritten. Der Replay Buffer (siehe Replay Buffer) speichert die Schritte von 700 Episoden. Somit speichert er $700 \cdot 64 = 44'800$ Schritte. Das Training (des Neuronalen Netz) findet in jedem vierten Schritt statt mit einem Batch von 64 zufällig gewählten Schritten aus dem replay buffer.

1.1.3 Hyperparameteroptimierung

Die genauen Hyperparameter des Reinforcement Learning Modells, wie auch die der Variationen (siehe Variation) stehen im Anhang. Alle diese Hyperparameter sind durch eine Implementierung des Baysian Algorithmus [**fernando˙bayesian˙2022**] (siehe Theorie Baysian) optimiert.

Mit jeder Iteration des Baysian Algorithmus trainiert das Reinforcement Learning Modell für eine Epoch und 4000 Episodes. Das Training läuft dabei in jeder Iteration mit den selben 4000 Bildern als Vorlagen in der selben Reihenfolge ab. Das schliesst einen willkürlichen Einfluss der Trainingsdaten während der Optimierung aus. Der Wert, der durch den Baysian Algorithmus maximiert werden soll, wird am Ende jedes Trainings berechnet.

Zur Optimierung der Hyperparameter wird der Baysian Algorithmus verwendet. Es wird ein bereits implementierung in Python genutzt, um eine schnellere Ergebnisse zu erzielen. [**fernando˙bayesian˙2022**]

Als Blackboxfunktion wurde das Agent im Environment genommen. Dieser trainiert 4000 Episoden und der Durchschnitt des Rewards der letzten 12 Episoden wird als Wert zur optimierung verwendet. Um eine bessere Vergleichbarkeit dieser Werte sicherzustellen, werden immer dieselben 4000 Referenzbilder gezeigt.

1.2 Evaluierung der Leistung

In diesem Unterkapitel sind die Kriterien definiert, die die Leistung der künstlichen Intelligenz evaluieren. Mit anderen Worten beschreiben die Kriterien, wie gut die künstliche Intelligenz nachzeichnet. Für eine präzise Evaluierung sind alle Kriterien durch einen Zahlenwert repräsentierbar. Dieser Zahlenwert geht direkt aus Berechnungen im Computerprogramm hervor. Die Kriterien und ihre jeweiligen Berechnungen werden nachfolgend beschrieben.

1.2.1 Erkennbarkeit

Das Kriterium der Erkennbarkeit beschreibt, ob in der Vorlage das gleiche Motiv wie in der Zeichnung der künstlichen Intelligenz erkannt wird. Wenn Beispielsweise in beiden Fällen eine Fünf erkannt wird, hat das Kriterium den Wert 1. Wird in der Vorlage eine Fünf erkannt, aber in der Zeichnung eine Vier, hat das Kriterium den Wert 0

Welches Motiv in der Zeichnung erkannt wird, ist durch eine zweite künstliche Intelligenz bestimmt. Diese künstliche Intelligenz beurteilt ein Motiv nur als erkannt, wenn das zugehörige Neuron im Output des neuronalen Netzen einen Wert von über 0.9 hat. Das entspricht einer hohen Wahrscheinlichkeit zur korrekten Beurteilung.

Für die verschiedenene Arten von Strichbildern, die die künsltiche Intelligenz zeichnen soll, existieren vortrainierte Modelle von neuronalen Netzen. Die vortrainierten Modelle, die in dieser Arbeit verwendet wurden sind in der Tabelle ... ersichtlich. Diese Modelle sind mit den selben Daten trainiert, die für die künstlichen Intelligenz dieser Arbeit als Vorlage zum Abzeichnen dienen.

Tabelle: Art — Entwickler — Trainiert mit — Genauigkeit

Zahlen Buchstaben Strichbilder von Objekten

Dieses Kriterium ist in der Fragestellung dieser Untersuchung angedeutet. Die Frage ist bestätigt, wenn die künstliche Intelligenz dieses Kriterium konsequet erfüllt (und auf physische Weise zeichnet). Es gibt aber noch weitere Kriterien, die andere Facetten der Leistung der künstlichen Intelligenz betreffen.

1.2.2 Prozentuale Übereinstimmung

Dieses Kriterium ist durch die die prozentuale Übereinstimmung der weissen (gezeichneten) Pixel zwischen der Vorlage und der Zeichnung der künstlichen Intelligenz definiert. Der Wert K dieses Kriteriums zu einem bestimmten

Schritt t berechnet sich aus folgender Formel:

$$K(t) = \frac{G(t)}{G_{max}} \tag{1.1}$$

G_{max} entspricht der Anzahl aller weissen Pixeln in der Vorlage. $G(t)$ entspricht der Anzahl der weissen Pixel, die zwischen der Vorlage und der Zeichenfläche übereinstimmen. Wenn der gleiche Pixel (am gleichen Ort) in der Vorlage und in der Zeichenfläche weiss ist, erhöht sich diese Anzahl um Eins. Wenn in der Zeichenfläche ein weisser Pixel gezeichnet ist, der in der Vorlage schwarz ist, sinkt die Anzahl um Eins. $G(t)$ und somit auch $K(t)$ können dadurch auch negative Werte annehmen. Der maximale Wert von $K(t)$ ist 1, was einer Genauigkeit von 100% entspricht.

1.2.3 Geschwindigkeit

Dieses Kriterium beschreibt, wie schnell die Zeichnung der künstlichen Intelligenz fertig ist. Der Wert dieses Kriteriums entspricht der Anzahl Steps, die die künstliche Intelligenz macht, bis die Zeichnung fertig ist. Eine kleinere Anzahl Steps entspricht einer schnelleren Fertigstellung der Zeichnung und somit einer besseren Leistung nach diesem Kriterium

Eine Zeichnung gilt als fertig, wenn die prozentuale Übereinstimmung (siehe proz Übereinstimmung) mindestens 70% beträgt und die Zahl nach der Definition (siehe erkennbarkeit) erkannt wird. Wenn die Zeichnung bis zum Ende die Bedingungen einer fertigen Zeichnung nicht erfüllt, hat dieses Kriterium den Wert 64. Das entspricht der maximalen Anzahl Steps, die die künstliche Intelligenz für eine Zeichnung begeht.

1.3 Physikalische Umgebung

Bei der physikalischen Umgebung für die Untersuchung handelt es sich um ein sehr simples inertial System. Also ein System in dem die Gesetze der Kinematik gelten. Das Ziel dieser Veränderung ist es die Anzahl möglicher Aktionen des Agenten zu reduzieren, wodurch es einfacher werden soll diese zu erlernen. Zusätzlich soll die simulierte Umgebung auch einer reellen Situation näher kommen und somit sich mehr an das menschliche Zeichnen annähern.

Die Umgebung nimmt Kraftvektoren entgegen und berechnet darauf hin die neue Stiftposition auf der Zeichenebene. Der Kraftvektor wird gemäss der Gesetze der Kinematik zu der schon bestehenden Geschwindigkeit hinzugefügt. Jeder Zeitschritt des Agenten repräsentiert in der Physik die vergangene Zeit $t = 1$. ($\frac{\vec{F}}{m} \cdot t = \vec{v}$). Pro Zeitschritt wird die Geschwindigkeit auch kleiner durch eine simulierte Reibung ($m \cdot g \cdot \mu = F_R$). Für die Position wird immer die Geschwindigkeit zu der jetzigen Position addiert.

Wie aus den Formeln herauszulesen ist, besitzt der Stift und auch die Umgebung physikalische Eigenschaften (m, g, μ), welche für ein optimales Ergebnis später optimiert werden.

Der Agent hat in der Umgebung die Möglichkeit haben seinen Stift mit Kraftvektoren zu beschleunigen. Nicht nur eine Beschleunigung ist möglich, sondern das Gleiten mit der restlichen Geschwindigkeit ist für den Agenten möglich.

Um das menschlichte Zeichnen noch näher zu bringen kann sich der Agent zusätzlich entscheiden, wie stark der Stift auf die Zeichenfläche drückt. Das bewirkt, dass um den Stift herum Pixel zusätzlich bemalt werden. Zu der Stärker kommt auch dazu, dass der Agent sich entscheiden kann, gar nicht auf das Papier zu drücken, sondern sich ganz vom Papier zu heben und nicht zu zeichnen.

In der Realität müsste der Druck auch mit einer Beschleunigung gelöst werden, was zur Vereinfachung weggelassen wird.

Training auf physikalische Umstände

Da die Aktiondes Agenten nur ein einfacher Integer ist, hat jeder Kraftvektor einen eigenen Index bekommen. Es gibt insgesamt $20 + 1(\text{gar nicht bewegen})$ Richtungen in die der Agent sich bewegen kann. Der Stift wird somit immer in eine dieser Richtungen, um den Betrag 1 beschleunigt. Mit den Optionen des Drucks kommt die Formel für die Anzahl Aktionsmöglichkeiten heraus: $21 \cdot (\underbrace{1}_{\text{nicht zeichnen}} + \underbrace{s}_{\text{maximale Stärke des Zeichnens}})$. So sind die Möglichkeiten der Aktionen des Agenten viel stärker begrenzt als in der ursprünglichen Variante.

Es stellte sich zu beginn des Training heraus, dass der Agent das Konzept der Geschwindigkeit noch nicht verstanden hat. Um dieses Problem zu lösen wurde die Geschwindigkeit als Kontext zur Observation hinzugefügt und somit als Eingabe des Netzes verwendet. Dies brachte bessere Ergebnisse, allerdings gelang es noch bessere Ergebnisse zu erziehlen, in dem man die Geschwindigkeit nicht roh als Zahlen mitgibt, sondern den lokalen Patch in die Richtung der Geschwindigkeit, also der nächsten Position des Agenten verschiebt.

Ein weiteres Problem während dem Training war, dass der Agent viel zu hohe Geschwindigkeiten angenommen hat und somit oft aus der Zeichenebene verschwinden wollte. Dieses Problem wurde durch die Reward Function gelöst, sodass er bei solchem Verhalten bestraft wird.

Parameteroptimizerung

Nach einigem Trainieren zeigt sich, dass der Agent am meisten Erfolg mit einer maximalen Zeichenstärke von 1 hat. Zur Optimierung der physikalischen Parameter wird gleich, wie bei der Hyperparameteroptimierung (siehe Hyperparameteroptimierung) der Baysian Algorithmus verwendet um das bestmöglichste Ergebnis zu erzielen.

1.4 Schrifterkennung als Kriterium

Dieses Kapitel behandelt das Kriterium der Erkennbarkeit der gezeichneten Zahl. Dieses Kriterium nähert qualitativ die Menschlichkeit der Zeichnung, im Vergleich zum Kriterium der prozentualen Genauigkeit, weiter an. Die Annäherung an menschliches Verhalten liegt dabei in der Annahme, dass für Menschen Erkennbarkeit wichtiger als Genauigkeit ist. Diese Annahme erklärt zum Beispiel Unterschiede in der Handschrift von verschiedenen Menschen. Handgeschriebene Zahlen sehen je nach Person verschieden aus, bleiben aber in den allermeisten Fällen für andere Personen erkennbar. So wäre es auch für das Computerprogramm profitabel, Erkennbarkeit der Genauigkeit vorzuziehen.

Das Computerprogramm benötigt eine Funktion, um die Erkennbarkeit einer Zahl zu beurteilen. Diese Funktion ist durch Schrifterkennungssoftware, also eine weitere künstliche Intelligenz, zugänglich (siehe Theorie Schrifterkennung). Die in dieser Arbeit verwendete Schrifterkennungssoftware stammt von Mazzia, Vittorio und Salvetti, Francesco und Chiaberge, Marcello. [**mazzia˙salvetti˙efficient-capsnet˙2021**] Die Schrifterkennungssoftware erreicht bei einem Test von 5000 Bildern aus den Trainingsdaten eine Genauigkeit von .....%

Eine Zahl gilt nur als erkannt, wenn das zugehörige Neuron einen Wert von über 0.9 hat. Das entspricht einer hohen Wahrscheinlichkeit zur korrekten Beurteilung.

1.4.1 Training auf Erkennbarkeit

Mit der Schrifterkennungssoftware ist die Erkennbarkeit der Zeichnungen für jede Version des Computerprogramms bestimmbar. Das Computerprogramm, beziehungsweise der Reinforcement Learning Algorithmus, kann aber auch spezifisch auf dieses Kriterium trainiert werden.

Um das Computerprogramm auf Erkennbarkeit zu trainieren, beinhaltet die Reward-Funktion eine Implementierung der Schrifterkennungssoftware. Die Schrifterkennungssoftware beurteilt in jedem Schritt die erkannte Zahl in der Vorlage und der aktuellen Zeichnung. Wenn die erkannte Zahl in der Vorlage und der Zeichnung gleich ist, erhält der Agent einen Reward von 0.1. In diesem Zustand funktioniert die Reward-Funktion nicht. Das heisst,

der Agent kann den akkumulierten Reward nicht vergrössern. Für dieses Problem gibt es mindestens zwei Gründe.

Der erste Grund liegt in der Einschätzung der Schrifterkennungssoftware, die teilweise für einen menschlichen Betrachter Fragwürdig ist. Zum Beispiel schätzt die Schrifterkennungssoftware eine leere Zeichenfläche mit einer hohen Wahrscheinlichkeit als eine Eins ein. Das ist in diesem Fall ein Problem, weil dadurch der Agent einen positiven Reward (eine Belohnung) für eine leere Zeichenfläche erhält. Das stört das weitere Lernverhalten, indem es wahrscheinlicher wird, dass der Agent nicht mehr zeichnet. Die verwendete Lösung ist, die Schrifterkennungssoftware erst ab einer gewissen prozentualen Genauigkeit einzusetzen. So löst die Schrifterkennungssoftware erst Rewards aus, wenn die prozentuale Genauigkeit der Zeichnung mehr als 20% beträgt.

Der zweite Grund liegt in einer zu hohen Abhängigkeit auf den Zufall. Damit die Schrifterkennungssoftware einen Reward auslösen kann, muss der Agent eine gut erkennbare Zahl zeichnen. Diese Zeichnung muss durch Zufall entstehen, weil der Agent zuvor ohne einen Reward nicht besser wird. Dieses Problem wird ebenfalls durch eine Implementierung des Kriteriums der prozentualen Genauigkeit gelöst. Dieses Kriterium ermöglicht in der Reward-Funktion, wie im Grundprogramm (siehe Grundprogramm), einen positiven Reward selbst für kleine Verbesserungen. Dieser Reward ist vor allem am Anfang des Trainings wichtig. Deswegen nimmt der Reward über das ganze Training hinweg linear ab, während der Reward für eine richtig erkannte Zahl linear zunimmt.

1.5 Geschwindigkeit als Kriterium

Eine schnelle Fertigstellung der Zeichnung ist eine weitere Annäherung an menschliches Zeichnen. Unter anderem hat das Kriterium des letzten Kapitels damit zu tun. Die Erkennbarkeit der gezeichneten Zahl ist wichtiger als die Genauigkeit, weil Menschen dadurch unvorsichtiger und schneller vorgehen.

Die Zeit bis zur Fertigstellung wird durch die Anzahl Schritte bis zu diesem Punkt gemessen. Eine geringere Anzahl Schritte entspricht einer schnelleren Fertigstellung. Die Zeichnung gilt als fertig, wenn die prozentuale Genauigkeit mindestens 70% beträgt und die Zahl nach der Definition im letzten Kapitel erkannt wird.

Training auf Geschwindigkeit

Der numerische Wert für die Zeit bis zur Fertigstellung der Zeichnung (die Geschwindigkeit) kann in die Reward-Funktion integriert werden. Dadurch wird der Reinforcement Learning Algorithmus auf die maximale Geschwindigkeit

trainiert. Die angepasste Reward-Funktion sieht folgendermassen aus: Am Ende jeder Zeichnung wird der Reward jedes Schrittes mit einem Faktor F multipliziert. Dieser Faktor berechnet sich aus folgender Formel:

$$F = 2 - \frac{S}{S_{max}}$$

S_{max} entspricht der Anzahl Schritte, die der Agent pro Zeichnung (Episode) begeht (siehe Grundprogramm). S Entspricht der Anzahl Schritte zur fertigstellung der Zeichnung. Der Faktor nimmt einen Wert zwischen 1 und 2 an. Wenn der Agent die Zeichnung bis zum Ende einer Episode nicht fertigstellen kann, ist $F = 1$. In diesem Fall wird der Reward also nicht angepasst. Der Agent zeichnet immer S_{max} Schritte pro Episode. Das verhindert eine ungleichmässige Verteilung der verschiedenen Episoden im replay-buffer. S wird bis an das Ende der Episode gespeichert, wenn die Anforderungen für die Fertigstellung einer Zeichnung das erste Mal erfüllt sind.

Literatur

[1] Tao Zhou u. a. "Learning to Sketch with Deep Q Networks and Demonstrated Strokes". In: *arXiv:1810.05977 [cs]* (14. Okt. 2018). arXiv: 1810.05977. URL: http://arxiv.org/abs/1810.05977 (besucht am 31. 03. 2022).

Abbildungsverzeichnis