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 Intelligenz (KI) zu entwickeln und dessen Leistung auszuwerten. Die Diskussion dieser Resultate führt schlussendlich zu einer Antwort auf die Fragestellung. Die Entwicklung der KI besteht aus zwei Teilen. Der eine Teil umfasst die Definition der Kriterien, nach denen die Leistung der KI evaluiert wird (siehe Evaluierung der Leistung). Der andere Teil umfasst die Entwicklung der KI (siehe Grundprogramm), zusammen mit verschiedenen Variationen (siehe Variationen). Die Variationen haben jeweils einen unterschiedlichen Fokus auf die definierten Kriterien. Die Auswertung (siehe Auswertung) bezieht sich ebenfalls auf die definierten Kriterien. Die Leistung der KI wird dabei in einer Testumgebung für das Zeichnen von verschiedenen Arten von Strichbildern erfasst.

1.1 Grundprogramm

Die KI ist abhängig von den Kriterien, die dessen Leistung definieren (siehe Evaluierung der Leistung). Mit anderen Worten trainiert die KI auf diese Kriterien. Das Ziel des Grundprogrammes ist, die allgemeine Trainingsumgebung für die KI bereitzustellen. Dieses Grundprogramm ist unabhängig von einem spezifischen Kriterium und kann stattdessen auf ein ausgewähltes Kriterium trainiert werden. Reinforcement Learning Modelle mit einer undefinierten Reward Function (siehe **??**) stellen diese Eigenschaften bereit. Eine Reward function, basierend auf einem spezifischen Kriterium, ermöglicht das Training auf dieses Kriterium. Das Grundprogramm ist in Python unter der Verwendung des Keras Frameworks implementiert (siehe **??**).

1.1.1 Doodle-SDQ als Basis

Das Reinforcement Learning Modell des Grundprogrammes basiert auf Doodle-SDQ. (siehe **??**) Von Doodle-SDQ ist das neuronale Netz, bezogen

auf die Form des Inputs, des Outputs und den Hidden Layers, zu grossem Teil übernommen. Die relevanten Anpassungen zwischen Doodle-SDQ und dem 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 (siehe **??**) 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. Der global Stream (siehe **??**) des Inputs in das Neuronale Netz ändert sich bis auf die neue Grösse der Bilder nicht. Die Pixel der Bilder, wie auch die Zeichenfläche, nehmen den Wert von einem Bit an. Eine Null repräsentiert einen schwarzen (nicht gezeichneten) Pixel an dieser Stelle im Bild und eine Eins einen weissen (gezeichneten) Pixel. Die genaue Architektur des neuronalen Netzes ist im Schema autorefarchitecture angegeben. Jeder Block in der Abbildung repräsentiert Eine Layer des neuronalen Netzes, wobei die Form des Inputs und die Form des Outputs von jeder Layer angegeben ist.

Der Local Stream, also das nahe Umfeld um den Agent schrumpft von 11×11 Pixel auf 7×7 Pixel. Somit schrumpft gleichzeitig der Action-Space (siehe **??**) des Agenten von $2 \cdot 11 \cdot 11 = 242$ Actions auf $2 \cdot 7 \cdot 7 = 98$ Actions. Das bedeutet für den Agent, dass er sich pro Step um maximal drei Pixel von seiner Position wegbewegen kann. Diese Bewegung kann der Agent entweder zeichnend oder nicht zeichnend ausführen (siehe autorefzeich vs nicht zeich).

Falls der Agent die Action zeichnend ausführt, zieht das Programm einen Strich zwischen der alten und der neuen Position. Mit anderen Worten werden alle Pixel 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.

Actions des Agents, die ihn über die vorgegebene Zeichenfläche hinaus positionieren würden, sind nicht zulässig. Diese Actions können vom Agent nicht gewählt werden und ihr optimaler Q-Value (siehe autorefzeich vs nicht zeich) ist in jedem Fall 0. Das hat zur Folge, dass nach dem Training die allermeisten unzulässigen Actions einen Q-Value nahe oder gleich 0 haben. Das senkt die Wahrscheinlichkeit, dass der Agent versucht, eine unzulässige Action auszuführen.

1.1.2 Präparierung der Daten und Optimierung

Die Trainingsdaten bestehen aus 36′000 Bildern von handgeschriebenen Ziffern aus dem MNIST Datenset (siehe **??**). Die restlichen Bilder des MNIST Datensets machen die Testdaten aus. Die Bilder im Datenset sind als Bitmap

dargestellt, wobei jedes Element (jeder Pixel) einen Wert zwischen 0 und 255 annimmt. Die Zahl repräsentiert eine Graustufe, wobei 0 Schwarz ist und 255 Weiss. Diese Graustufen werden entfernt. Jeder Pixel mit einem Wert über 0 übernimmt den Wert 1, wodurch die Bilder nur noch aus Einsen und Nullen bestehen. Dabei ist 0 Schwarz und 1 Weiss (siehe autorefnorm vs. nogray). So stimmen die Bilder mit den Zeichnungen, die der Agent produzieren kann, überein.

Das Grundprogramm trainiert mit 4000 Bildern, von denen jede Ziffer 400 Bilder ausmacht. Die restlichen Bilder in den Trainingsdaten sind für mögliche Erweiterungen aufgehoben. Der Agent zeichnet jedes der 4000 Bilder ein Mal und trainiert somit für 4000 Episodes. Der Agent macht 64 Steps pro Episode. Er kann sich also pro Zeichnung 64 Mal bewegen. Das neuronale Netz passt sich in jedem vierten Step an, mit einem Batch von 64 zufällig ausgewählten Steps aus dem Replay Buffer.

Die Hyperparameter des Grundprogrammes, wie auch der Variationen (siehe Variationen) sind durch den Bayesian Optimization Algorithmus optimiert (siehe **??**). Die Implementierung des Algorithmus in Python stammt von [**fernando˙bayesian˙2022**]. Der Algorithmus ändert sich für verschiedene Variationen der KI nicht und ist somit Teil des Grundprogrammes, wobei er zu der optimalen Leistung der KI beiträgt.

Mit jeder Iteration des Baysian Optimization Algorithmus trainiert das Reinforcement Learning Modell für eine vom Algorithmus selbst bestimmte Anzahl Episodes. Die Zielvariable, die durch den Baysian Optimization Algorithmus maximiert werden soll, wird am Ende jeder Iteration des Trainings in der Testumgebung berechnet (siehe **??**). Auf welchem Kriterium die Zielvariable basiert, ist frei wählbar.

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 KI nachzeichnet. Für eine präzise und objektive Evaluierung sind alle Kriterien durch einen Zahlenwert definiert. Dieser Zahlenwert geht direkt aus Berechnungen vom Computerprogramm hervor. Die Kriterien und ihre jeweilige Berechnung sind 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 (siehe **??**). Diese zweite KI beurteilt ein Motiv nur als erkannt, wenn das zugehörige Neuron im Output des neuronalen Netzen einen Wert von über 0.75 hat. Das entspricht mit einer hohen Wahrscheinlichkeit der korrekten Beurteilung.

Um die verschiedenene Arten von Strichbildern, die die KI zeichnen soll, zu erkennen, existieren vortrainierte Machine Learning Modelle. Die in dieser Arbeit implementierten vortrainierten Modelle sind in der Tabelle ... ersichtlich. Diese Modelle sind mit den selben Daten trainiert, die in der Testumgebung (siehe **??**) als Vorlage zum Abzeichnen dienen.

Tabelle: Art — Entwickler — Trainiert mit — Genauigkeit Zahlen Buchstaben Strichbilder von Objekten

Dieses Kriterium ist in der Fragestellung (siehe **??**) angedeutet. Die Antwort auf die Frage fällt positiv aus, wenn die KI dieses Kriterium der Erkennbarkeit konsequet erfüllt. Neben der Erkennbarkeit existieren weitere Kriterien, die andere Aspekte 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 Step 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 (siehe autorefüber Pixel).

1.2.3 Geschwindigkeit

Dieses Kriterium beschreibt, wie schnell die Zeichnung der KI fertig ist. Der Wert dieses Kriteriums entspricht der Anzahl Steps bis zur Fertigstellung der Zeichnung. 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 Prozentuale Übereinstimmung) mindestens 70% beträgt und die Zahl der Definition entsprechend erkannt wird (siehe Erkennbarkeit). Wenn die Zeichnung bis zum Ende der Episode die Bedingungen einer fertigen Zeichnung nicht erfüllt, hat dieses Kriterium den Wert 64. Das entspricht der maximalen Anzahl Steps, die von der KI pro Zeichnung begangen werden.

1.3 Variationen

Dieses Kapitel beschreibt verschiedene Variationen ausgehend vom Grundprogramm (siehe Grundprogramm). Bei einigen dieser Variationen handelt es sich um konkrete Implementierungen der definierten Kriterien in die Reward Function (siehe **??**). Die Reward Function kann dabei auch auf mehreren Kriterien basieren. Der Unterschied zwischen den Variationen liegt im Fokus auf die Kriterien. Einige Variationen sind untereinander kombinierbar. Andere Variationen führen Strukturelle Änderungen an der KI ein, die über die Reward Function hinaus gehen. Das Ziel der strukturellen Änderungen ist eine grundsätzliche Verbesserung, oder zumindest eine Anpassung des Verhaltens und der Leistung der künstlichen Intelligenz.

1.3.1 Basis Reward-Function

Die Basis Reward Function ist die einfachste Erweiterung des Grundprogrammes (siehe Grundprogramm) zu einer funktionierenden künstlichen Intelligenz. Diese Reward Function implementiert das Kriterium der prozentualen Übereinstimmung (siehe Prozentuale Übereinstimmung). Der Reward für eine Action berechnet sich aus der Differenz zwischen der prozentualen Übereinstimmung vor dem Ausführen der Action, und und der prozentualen Übereinstimmung nach dem Ausführen der Action (also $K(t-1)$ und $K(t)$). Somit wird der Reward R zum Step t durch folgende Formel berechnet.

$$R(t) = K(t) - K(t-1)$$

Der Reward eines Steps entspricht somit nicht der gesamten prozentualen Übereinstimmung zu diesem Step. Stattdessen Entspricht der Reward der Veränderung der prozentualen Übereinstimmung, ausgelöst durch die Action zu diesem Step. Der addierte Reward aller Steps enstspricht dem absoluten Wert der prozentualen Übereinstimmung.

1.3.2 Training auf Geschwindigkeit

Der numerische Wert für die Zeit bis zur Fertigstellung der Zeichnung (siehe Geschwindigkeit) kann in die Reward-Function integriert werden.

Dadurch trainiert die künstliche Intelligenz auf eine minimale Zeit bis zur Fertigstellung. Die Variation verwendet die grundsätzlich die Basis Reward-Function (siehe Basis Reward-Function). Die Anpassung davon sieht folgendermassen aus: Am Ende jeder Zeichnung wird der Reward jedes Steps mit einem Faktor f multipliziert. Dieser Faktor berechnet sich aus folgender Formel:

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

S_{max} entspricht der Anzahl Steps, die der Agent pro Zeichnung (Episode) begeht (siehe Präparierung der Daten und Optimierung). S Entspricht der Anzahl Steps bis zur Fertigstellung der Zeichnung. Der Faktor nimmt einen Wert zwischen 1 und 2 an. Ein grösserer Faktor f entspricht einer schnellen Fertigstellung und deswegen einem hohen Reward. Wenn der Agent die Zeichnung bis zum Ende einer Episode nicht fertigstellen, ist $f = 1$ (siehe Geschwindigkeit). In diesem Fall unterscheidet sich die Reward-Function nicht von der Basis Reward-Function. Wenn die Zeichnung früher fertiggestellt wird, zeichnet der Agent trotzdem S_{max} Steps. Das verhindert eine ungleichmässige Verteilung zwischen verschiedenen Episodes im Replay-Buffer (siehe **??**). In diesem Fall wird S nur in dem Step gespeichert, in dem die Zeichnung zum ersten Mal die Bedingung einer Fertigstellung erfüllt.

Der Fokus des Trainings auf eine maximale Geschwindigkeit erfährt einen weiteren anstieg, durch eine Anpassung der Bedingung für eine fertige Zeichnung während dem Training. Die minimale prozentuale Übereinstimmung einer fertigen Zeichnung ist als 75% definiert. Zu beginn des Trainings wird dieser Wert auf 25% heruntergesetzt, und über das Training hinweg linear bis auf 75% erhöht. Dadurch löst die Reward Function bei einer unfertigen Zeichnung bereits positive Rewards für die Geschwindigkeit aus.

1.3.3 Training auf Erkennbarkeit

Das Kriterium der Erkennbarkeit kann, anders als die anderen Kriterien, nur teilweise in die Reward Function integriert werden. Das Kriterium strebt eine Erkennbarkeit, unabhängig von der Art der Strichbilder, an (siehe Erkennbarkeit). Die künstliche Intelligenz trainiert allerdings nur auf das Nachzeichnen von Ziffern. Aus diesem Grund trainiert diese Variation nur auf die Erkennbarkeit von Ziffern, und lässt die anderen Arten von Strichbildern ausser vor.

Die Reward-Function (siehe **??**) dieser Variation beinhaltet eine zweite KI, die handgeschriebene Ziffern erkennt (siehe Unterabschnitt 1.2.1). Diese zweite KI beurteilt in jedem Step, welche Ziffern sie in der Vorlage und in der aktuellen Zeichnung erkennt. 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-Function allerdings nicht. Der Agent kann

den akkumulierten Reward nicht maximieren. Zwei Ansätze gehen auf dieses Problem ein. Beide Ansätze sind Teil dieser Variation.

Der erste Ansatz schlägt vor, die zweite KI erst ab einer gewissen prozentualen Übereinstimmung (siehe Prozentuale Übereinstimmung) einzusetzen. In diesem Fall löst die korrekte Erkennung erst ab einer prozentualen Übereinstimmung von 20% einen positiven Reward aus. Diese zusätzliche Bedingung ist notwendig, weil die Beurteilungen der zweiten KI teilweise für einen menschlichen Betrachter fragwürdig sind. Zum Beispiel schätzt die zweite KI eine leere Zeichenfläche mit einer hohen Wahrscheinlichkeit als eine Eins (siehe autorefmnist rec) ein. Das ist ein Problem, weil dadurch der Agent einen positiven Reward für eine leere Zeichenfläche erhält. Das stört das weitere Lernverhalten, weil es die Wahrscheinlichkeit erhöht, dass die KI nicht mehr zeichnet.

Der zweite Ansatz implementiert neben der Reward-Function der Erkennbarkeit erneut die Basis Reward-Function (siehe Basis Reward-Function). Die Relevanz der beiden Reward-Functions ändert sich allerdings über das Training hinweg. Die Rewards werden in jedem Step mit einem bestimmten Faktor multipliziert. Zu Beginn des Trainings ist der Faktor für den Reward der Basis Reward-Function $f_B = 1$ und der Faktor für den Reward basierend auf der Erkennbarkeit $f_E = 0$. Vom Start ausgehend sinkt f_B linear und f_E steigt linear. Ab einem gewissen Punkt bleiben beide Faktoren stehen (autorefdecrementor graph). Blieben die Faktoren ab diesem Punkt nicht konstant, würde die Variation, gestützt auf Beobachtungen, an Stabilität der Leistung verlieren.

Das Zusammenspiel der beiden Reward-Functions hat den Vorteil, dass die künstliche Intelligenz zu Beginn des Trainings durch die Basis Reward-Function für kleine Erfolge positive Rewards erzielt. Die Reward-Function der Erkennbarkeit ermöglicht das nicht, da sie erst für eine korrekte Erkennung einen Reward auslöst. Eine korrekte Erkennung ist für eine untrainierte KI schwer zu erreichen. Deswegen muss die KI durch die Basis Reward-Function gewissermassen vortrainiert werden, um schlussendlich von der Reward-Function der Erkennbarkeit zu profitieren

1.3.4 Physikalische Umgebung

Diese Variation spezialisiert sich auf kein Kriterium. Stattdessen verändert sich die Umgebung, in der sich der Agent bewegt (siehe **??**). Auch der Input und der Output des neuronalen Netzes sind angepasst. Durch diese Veränderungen unterscheidet sich die Variation vom Grundprogramm. Sie bleibt allerdings mit den anderen Variationen (siehe Training auf Erkennbarkeit und Training auf Geschwindigkeit) kompatibel, Da diese ausschliesslich die Reward-Function anpassen.

Die Variation ergänzt die Umgebung durch physikalische Simulationen. Diese physikalische Umgebung definiert die physichen Rahmenbedingungen des

Zeichnens neu, mit dem Ziel, diese näher an die Realität zu bringen (siehe **??**).

Der Agent hat neu eine Geschwindigkeit, die durch einen Vektor $\vec{v}$ dargestellt ist. Die Geschwindigkeit beschreibt, um wie viele Pixel und in welche Richtung sich der Agent pro Step bewegt. Die folgende Formel beschreibt, wie sich die Position des Agenten vom Step t bis zum nächsten Step $t+1$ ändert:

$$\vec{p}(t+1) = \vec{p}(t) + \vec{v}(t)$$

$\vec{p}(t)$ beschreibt die Position des Agents als einen Ortsvektor auf der Zeichenfläche zum Step t und $\vec{v}(t)$ beschreibt die Geschwindigkeit des Agenten zum Step t. Die Position rundet in jedem Step auf ganze Zahlen. Das kommt daher, dass die Geschwindigkeit auch Dezimalzahlen annehmen kann, aber die Position nur durch ganze Zahlen dargestellt ist.

Zur Geschwindigkeit des Agent wird in jedem Step ein Beschleunigungsvektor addiert. Jede Action, die der Agent wählen kann, entspricht einem anderen Beschleunigungsvektor. Der Action-Space (siehe **??**) besteht neu aus 42 Actions. 21 der 42 Actions entsprechen Beschleunigungsvektoren im zeichnenden Zustand. Die anderen 21 Actions entsprechen den selben Vektoren im nicht zeichnenden Zustand. Die 21 verschiedenen Beschleunigungsvektoren haben folgende Form: Ein Vektor entspricht dem Nullvektor. Dieser verändert die Geschwindigkeit des Agents nicht. 8 Vektoren sind um den Agent herum mit einer Länge von 0.8 in gleichmässigem Abstand von einander angeordnet. Zusammen bilden diese Vektoren einen Kreis um den Agent. Die restlichen 12 Vektoren sind in einem grösseren Kreis gleichmässig angeordnet. Die Vektoren haben dabei eine Länge von 1.2 (siehe autorefactionspace). Mit dem gewählten Beschleunigungsvektor $\vec{a}(t)$ berechnet sich die Geschwindigkeit im nächsten Step $t+1$ aus dem aktuellen Step t durch folgende Formel:

$$\vec{v}(t+1) = \vec{v}(t) + \vec{a}(t)$$

Der Betrag der Geschwindigkeit $\vec{v}(t+1)$ des Agents wird, unabhängig von der gewählten Action, in jedem Step um 0.4 verringert. Das simuliert eine Reibungskraft, die auf den Agent einwirkt.

Die Veränderung in der Umgebung erfordert weitere Anpassungen im neuronalen Netz (siehe **??**). Ohne diese Anpassungen kann die KI den akkumulierten Reward nicht maximieren. Das Problem ist, dass die aktuelle Geschwindigkeit des Agents kein Teil der Observation (siehe **??**) ist. Der Agent berücksichtigt deswegen seine Geschwindigkeit nicht in seinen Entscheidungen. Die Lösung dieses Problems bietet eine Verschiebung des Local image patch (siehe **??**). Im Grundprogramm entspricht der Mittelpunkt des Local image patchs genau der Position des Agents. Neu befindet sich der Mittelpunkt dort, wo sich der Agent laut seiner aktuellen Geschwindigkeit im nächsten

Schritt befinden wird. Durch diese Verschiebung des Local image Patches erhält der Agent Informationen über seine Geschwindigkeit, ohne dessen numerischen Wert zu kennen. Wie im Grundprogramm gibt der Local image patch den gesamten Bereich an, in dem sich der Agent im nächsten Schritt befinden kann. Die tatsächliche neue Position des Agents wird durch die Action seiner Wahl bestimmt. Die Grösse des Local image patches schrumpft von 7×7 Pixeln auf 5×5 Pixel, da alle möglichen Positionen des Agents nach einem Step auf einem 5×5 Feld Platz haben (siehe autoreflocal patch).

Ein weiteres Problem ist, dass der Agent sich durch seine Geschwindigkeit aus den vorgegebenen Grenzen der Zeichenfläche begeben kann. Im Grundprogramm (siehe Doodle-SDQ als Basis) kann der Agent Actions, die ihn in eine unzulässige Position bewegen würden, nicht auswählen. Wenn allerdings in der phyiskalischen Umgebung die Geschwindigkeit des Agents zu hoch ist, kann dieser keine Actions mehr wählen, die ihn innerhalb der Grenzen der Zeichenflächen hielten. Wenn der Agent durch zu hohe Geschwindigkeit über die Grenze hinausgeht, wird seine Geschwindigkeit auf den Nullvektor zurückgesetzt und die Reward Function löst einen negativen Reward von -0.05 aus. Der negative Reward soll die Häufigkeit dieser Vorfälle vermindern

1.4 Auswertung

Die Auswertung der Daten über die Leistung der künstlichen Intelligenz liefert das Resultat der Methode. Die Auswertung berechnet den Zahlenwert der definierten Kriterien (siehe Evaluierung der Leistung) für verschiedene Variationen der KI.

Die Variationen werden auf ihre Leistung für drei verschiedene Datensets überprüft. Die drei Datensets beinhalten verschiedene Arten von handgemachten Strichbildern. (siehe Unterabschnitt 1.2.1) Das erste Datenset, MNIST, beeinhaltet Ziffern, die in den Trainingsdaten nicht vorkommen. Das zweite Datenset, EMNIST Letters, beeinhaltet die 26 Kleinbuchstaben des Alphabets. Das dritte Datenset, QuickDraw, beeinhaltet Zeichnungen von insgesamt 345 verschiedenen Motiven. Die KI wird allerdings nur auf das Nachzeichnen von zehn Motiven überprüft. Die zehn Motive sind: 'Amboss', 'Apfel', 'Besen', 'Eimer', 'Bulldozer', 'Uhr', 'Wolke', 'Computer', 'Auge' und 'Blume' (siehe autorefDataset images für Beispiele aus den Datensets). Die Bilder in den drei Datensets sind gleich verarbeitet wie die Trainingsdaten (siehe Präparierung der Daten und Optimierung).

Die Variationen (siehe Variationen) der KI umfassen zwei Umgebungen und drei Reward-Functions **??**. Für jede Variation ist zur Vereinfachung eine Abkürzung definiert.

- Grundprogramm Umgebung: Grund
- Physikalische Umgebung: Physik

- Training auf Erkennbarkeit: Basis
- Training auf Geschwindigkeit: Speed
- Training auf Erkennbarkeit (von MNIST Ziffern): MNIST

Die folgenden Kombinationen an Variationen der künstlichen Intelligenz werden ausgewertet. Diese Kombinationen stellen einzelne Versionen der KI dar

- Grund-Basis
- Grund-MNIST
- Grund-Speed
- Grund-MNIST-Speed
- Physik-Basis
- Physik-MNIST
- Physik-Speed
- Physik-MNIST-Speed

1.4.1 Testumgebung

Die Leistungen der verschiedenen Variationen der künstlichen Intelligenz werden in einer Testumgebung (siehe **??**) ausgewertet. zwischen der Trainingsumgebung und der Testumgebung sind drei relevante Unterschiede erkennbar. Erstens trainiert die KI in der Testumgebung nicht. Die Testumgebung übernimmt eine trainierte Version der KI und verändert diese während dem Test nicht. Zweitens wählt der Agent in keinem Fall mehr eine zufällige Action. Stattdessen wählt er immer die Action mit dem höchsten Q-Value (gleichbedeutend mit $\varepsilon = 0$) (siehe **??**). Der Dritte Unterschied liegt in den Strichbildern, die für die künstliche Intelligenz als Vorlage dienen. Im Test zeichnet das Computerprogramm 2000 Bilder aus einem der drei zur Verfügung stehenden Datensets (siehe **??**).

Am Ende jeder Episode (das heisst jeder Zeichnung), wird der Zahlenwert für die verschiedenen Kriterien (siehe Evaluierung der Leistung) nach ihrer Definition ausgewertet und gespeichert. Die KI zeichnet auch in der Testumgebung für 64 Steps. Wenn eine Zeichnung der Definition entsprechend früher fertig ist (siehe Geschwindigkeit), wird die Anzahl Steps zu diesem Zeitpunkt als Wert des Kriteriums der Geschwindigkeit gespeichert und die KI zeichnet weiter. Der Durchschnitt aller gespeicherten Werte eines Kriteriums entspricht der Leistung der getesteten Variation in diesem Kriterium. Das Kriterium der Erkennbarkeit (siehe Erkennbarkeit) verwendet zur Auswertung je nach dem verwendeten Datenset im Test, dasjenige vortrainierte Modell, das auf dem selben Datenset trainiert ist. Da das Kriterium der Erkennbarkeit jeweils den Wert 0 oder 1 hat, ergibt der Durchschnitt aus allen Werten dieses Kriteriums eine prozentuale Angabe (in Dezimalform) darüber, in wie vielen Fällen das richtige Motiv erkannt wird.

Abbildungsverzeichnis