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. 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 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 Testumgebung) 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 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 (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 Variationen vom Grundprogramm (siehe Grundprogramm). Bei einigen dieser Variationen handelt es sich um konkrete Implementierungen der definierten Kriterien in die Reward Function. Die Variationen überschneiden sich teilweise in den Kriterien, die sie Implementieren. Der Unterschied der Variationen liegt im Fokus auf die verschiedenen Kriterien. Einige Variationen sind auch untereinander kombinierbar. Variationen können auch strukturelle Änderungen an der künsltichen Intelligenz, abgesehen von der Reward Function haben. Das Ziel dieser strukturellen Änderungen ist eine grundsätzliche Verbesserung, oder zumindest eine Veränderung der Leistung der künstlichen Intelligenz.

1.3.1 Basis Reward Function

Die Basis Reward Function ist die einfachste Erweiterung des Grundprogrammes zu einer funktionierenden künstlichen Intelligenz. Diese Reward Function implementiert das Kriterium der prozentualen Übereinstimmung (ref proz Übereinstimmung). Der Reward für eine Aktion berechnet sich aus der Differenz zwischen der prozentualen Übereinstimmung vor dem Ausführen der Aktion und der prozentualen Übereinstimmung nach dem Ausführen der Aktion. Somit wird der Reward R zum Schritt t durch folgende Formel berechnet. (Für $K(t)$: siehe eval)

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

Der Reward eines Schrittes entspricht dadurch nicht der gesamten prozentualen Übereinstimmung, sondern lediglich der Veränderung dieser, die durch den Schritt auslöst. Der addierte Reward aller Schritte enstspricht dem absoluten Wert der prozentualen Übereinstimmung.

1.3.2 Training auf Geschwindigkeit

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

trainiert die künstliche Intelligenz auf die kürzeste Zeit bis zur Fertigstellung. Die Variation verwendet die grundsätzlich die Basis Reward Function. Die Anpassung davon 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. Die Anpassung der Bedingung für eine fertige Zeichnung während dem Training ermöglicht eine weitere Verbesserung der Geschwindigkeit. Die minimale prozentuale Übereinstimmung einer fertigen Zeichnung ist als 80% definiert. Zu beginn des Trainings wird dieser Wert auf 30% heruntergesetzt, und über das Training hinweg linear bis auf 80% erhöht. Dadurch löst die Reward Function bereits bei einer unfertigen Zeichnung positive Rewards für die Geschwindigkeit aus, was den Fokus auf eine maximale Geschwindigkeit weiter erhöht.

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 eval 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 beinhaltet die künstliche Intelligenz, die handgeschriebene Ziffern erkennt (siehe eval erkennbarkeit). Diese künstliche Intelligenz beurteilt in jedem Schritt, welche Ziffern in der Vorlage und der aktuellen Zeichnung erkannt werden. 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. Zwei Ansätze lösen dieses Problem. Beide Ansätze sind Teil dieser Variation

Der erste Ansatz ist es, die künstliche Intelligenz, die Ziffern erkennt, erst ab einer gewissen prozenutalen Übereinstimmung (siehe prozentuale Übereinstimmung)

einzusetzen. Das heisst, dass die korrekte Erkennung erst ab einer prozentualen Übereinstimmung von 20% einen positiven Reward auslöst. Diese zusätzliche Bedingung ist notwendig, weil die Einschätzung der Ziffern durch die künstlichen Intelligenz 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.

Der zweite Ansatz beinhaltet ebenfalls das Kriterium der prozentualen Übereinstimmung. Dieses Kriterium ist in dieser Variation gleich wie in der Basis Reward Function implementiert. Der Unterschied ist, dass der Reward durch diese Reward Function über das Training hinweg linear kleiner wird. Das Training startet mit 100% dieses Rewards und sinkt bis zur letzten Episode des Trainings auf 10%. Gleichzeitig nimmt der Reward durch die korrekte Erkennung der Ziffer linear zu. Die Reward Function ändert sich somit über den Verlauf des Trainings hinweg. Das hat den Vorteil, dass die künstliche Intelligenz am Anfang des Trainings durch das Kriterium der prozentualen Übereinstimmung bereits für kleine Erfolge positive Rewards bekommt. Das Kriterium der Erkennbarkeit ermöglichst erst bei einer korrekten Erkennung einen Reward. Diese korrekte Erkennung ist für eine untrainierte künstliche Intelligenz schwer zu erreichen, wodurch sie nur in seltenen Fällen einen positiven Reward erreichen würde.

1.3.4 Physikalische Umgebung

Diese Variation spezialisiert auf kein Kriterium. Stattdessen ist die Umgebung, in der sich der Agent bewegt, verändert. 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, die hauptsächlich die Reward Function anpassan, kompatibel.

Die Variation verändert die Umgebung, in der der Agent sich bewegt, in eine simulierte physikalische Umgebung. Diese Umgebung definiert die physichen Rahmenbedingungen des Zeichnens neu und bringt diese optimalerweise näher an die Realität (siehe Diskussion).

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

$$p_{t+1} = p_t + v_t$$

p_n beschreibt die Postition des Agenten zu Schritt n und v_n beschreibt die Geschwindigkeit des Agenten zu Schritt n. Die Position rundet in jedem Schritt auf ganze Zahlen. Das kommt daher, dass die Geschwindigkeit auch Dezimalzahlen annehmen kann, aber die Positionnur durch ganze Zahlen dargestellt ist.

Zur Geschwindigkeit des Agenten wird in jedem Schritt ein Beschleunigungsvektor addiert. Jede Action, die der Agent wählen kann, entspricht einem anderen Beschleunigungsvektor. Der Action Space besteht neu aus 42 Aktionen. Davon entsprechen 21 Aktionen Beschleunigungsvektoren im zeichnenden Zustand. Die anderen 21 Aktionen entsprechen den selben Vektoren im nicht zeichnenden Zustand. Die 21 verschiedenen Beschleunigungsvektoren sehen folgendermassen aus: (siehe Abbildung action Kreis) Ein Vektor entspricht dem Nullvektor. Dieser verändert die Geschwindigkeit des Agenten nicht. 8 Vektoren sind um den Agenten herum mit einer Länge von 0.8 Pixeln in gleichmässigem Absant 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 Pixeln. Der Betrag der Geschwindigkeit des Agenten wird, unabhängig von der gewählten Aktion, in jedem Schritt um 0.4 Pixel verringert. Das Simuliert eine Reibungskraft, die auf den Agenten einwirkt. Mit dem gewählten beschleunigungsvektor a_t berechnet sich die Geschwindigkeit im nächsten Schritt $t+1$ aus dem aktuellen Schritt t durch folgende Formel:

$$v_{t+1} = v_t + a_t - 0.4$$

Die Änderungen in der Umgebung erfordern weitere Anpassungen im neuronalen Netz. Ohne diese Anpassungen lernt die künstliche Intelligenz nicht. Das Problem ist, dass die aktuelle Geschwindigkeit kein Teil der Observation ist. Die Entscheidungen des Agenten berücksichtigen dadurch dessen Geschwindigkeit nicht. Die Verschiebung von dem local patch bietet eine Lösung für dieses Problem. Im Grundprogramm entspricht der Mittelpunkt von dem Local Patch genau der Position des Agents. Neu befindet sich der Mittelpunkt dort, wo sich der Agent laut seiner aktuellen Geschwindigkeit im nächsten Schritt befindet (Die tatsächliche neue Position des Agents wird durch die Action seiner Wahl bestimmt. Wie im Grundprogramm gibt der local patch den Bereich an, in dem sich der Agent im nächsten Schritt befinden wird). Diese Verschiebung vom Local Patch ist eine implizite Angabe der Geschwindigkeit des Agents. Der Local Patch hat neu eine Grösse von 5×5 Pixeln an der Stelle von 7×7 Pixeln.

Ein weiteres Problem ist, dass der Agent sich durch seine Geschwindigkeit aus den vorgegebenen Grenzen der Zeichenfläche begeben kann. Im Grundprogramm (siehe) kann der Agent Aktionen, die ihn in diese Unzulässigen Positionen bewegen würden, nicht auswählen. Wenn die Geschwindigkeit zu hoch

wird, kann der Agent allerdings gar keine Aktionen mehr wählen, die ihn innerhalb der Grenzen der Zeichenflächen halten würden. 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. Dieser negative Reward soll zu hohe Geschwindigkeiten an gewissen Positionen auf der zeichenfläche vermeiden.

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 für verschiedene Variationen der künstlichen Intelligenz.

Die Variationen werden auf ihre Leistung für drei verschiedene Datensets überprüft. Die drei Datensets beinhalten verschiedene Arten von handgemachten Strichbildern Das erste Datenset, MNIST (mnist ref), beeinhaltet Ziffern, die in den Trainingsdaten nicht vorkommen. Das zweite Datenset, EMNIST letters (emnist ref), beeinhaltet die 26 Kleinbuchstaben des Alphabets. Das dritte Datenset, QuickDraw (quickdraw ref), beeinhaltet Zeichnungen von insgesamt 345 verschiedenen Motiven. die Variationen werden allerdings nur auf das Nachzeichnen von zehn Motiven überprüft. Die zehn Motive sind: ‘Amboss’, ‘Apfel’, ‘Besen’, ‘Eimer’, ‘Bulldozer’, ‘Uhr’, ‘Wolke’, ‘Computer’, ‘Auge’ und ‘Blume’. Die Bilder in den drei Datensets sind gleich verarbeitet wie die Trainingsdaten (siehe Prep daten).

Folgende Variationen der künstlichen Intelligenz werden ausgewertet:

- Grundumgebung + Basis Reward Functin
- Grundumgebung + Erkennbarkeit
- Grundumgebung + Geschwindigkeit
- Grundumgebung + Erkennbarkeit + Geschwindigkeit
- physikalische Umgebung + Basis Reward Functin
- physikalische Umgebung + Erkennbarkeit
- physikalische Umgebung + Geschwindigkeit
- physikalische Umgebung + Erkennbarkeit + Geschwindigkeit

1.4.1 Testumgebung

Die Leistungen der verschiedenen Variationen der künstlichen Intelligenz werden in einem Test, in einer Testumgebung ausgewertet. Die Testumgebung unterscheidet sich kaum von der Trainingsumgebung. Es gibt im Wesentlichen drei Unterschiede. Erstens trainiert die künstliche Intelligenz in der Testumgebung nicht. Die Testumgebung übernimmt eine trainierte künstliche Intelligenz und verändert diese während dem Test nicht. Zweitens wählt der Agent in

keinem Fall mehr eine zufälligen Aktion. Stattdessen wählt er immer die Aktion mit dem höchsten Q-Value (gleichbedeutend mit $\varepsilon = 0$). Der Dritte Unterschied liegt in den Strichbildern, die für die künstliche Intelligenz als Vorlage dienen. Im Test zeichnet das Computerprogramm 1040 Bilder aus einem der drei zur Verfügung stehenden Datensets.

Am Ende jeder Episode (d.h jeder Zeichnung) wird der Zahlenwert für die verschiedenen Kriterien 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 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 eine prozentuale Angabe (in Dezimalform) darüber, in wie vielen Fällen das richtige Motiv erkannt wird.

Abbildungsverzeichnis