

ReSketch AI

Maturarbeit

Ian Wasser, Robin Steiner

11. Oktober 2022

Betreut durch: Nicolas Ruh
Zweitbeurteilung: Dieter Koch

NKSA G19E

Abstract

ReSketch ist eine künstliche Intelligenz, die versucht, Strichbilder auf eine physische Weise nachzuzeichnen. Strichbilder sind in diesem Fall beispielsweise Ziffern oder Buchstaben. Um die Frage zu beantworten, inwiefern das möglich ist, sind definierende Kriterien des Nachzeichnens festgelegt. So soll die künstliche Intelligenz zum Beispiel nur Bewegungen ausführen können, die auch mit einem Stift möglich wären. Die künstliche Intelligenz erlernt das Nachzeichnen nach diesen Kriterien durch Deep Q-Learning, einem Reinforcement Learning Modell. Das Modell basiert auf der Arbeit hinter Doodle-SDQ (**zhou'learning'2018**), erfährt aber konzeptuelle Variationen, wie die Integration einer Physiksimulation. Die künstliche Intelligenz ist auf das Nachzeichnen von Ziffern trainiert. Ein Test dieser trainierten künstlichen Intelligenz auf Buchstaben und andere Arten von Strichbildern führt zur Antwort auf die Frage, ob eine künstliche Intelligenz das Nachzeichnen im Allgemeinen erlernen kann.

Vorwort

Diese Arbeit ist eine Untersuchung im Bereich der künstlichen Intelligenz. Die Fragestellung der Untersuchung wird mithilfe einer selbst programmierten künstlichen Intelligenz beantwortet.

Wir haben uns für das Thema künstliche Intelligenz entschieden, weil damit praktische Arbeit mit intellektueller Forschung verbunden werden kann. Das Thema ermöglicht ausgeprägte, praktische Programmierarbeiten, was uns zuspricht. Zusätzlich ermöglicht künstliche Intelligenz einfache Forschung. Mit einfacher Forschung ist dabei nicht der Grad der Komplexität gemeint, sondern die Vielfalt der Möglichkeiten. Es gibt Aspekte und Anwendungen der künstlichen Intelligenz, die für Schüler zugänglich sind und noch nicht zu weit erforscht sind, um neue Ideen zu finden. Ausserdem benötigt die Forschung an künstlicher Intelligenz nur einen Computer. Experimente und Tests können durch Programmcode ausgeführt werden. Die Auswertung der Experimente findet auf demselben Computer statt und die Genauigkeit der Ergebnisse stellt ebenfalls kein Problem dar, da der Computer die Zahlen direkt berechnet. Der Computer ist eine optimale Umgebung für eine erste Forschungsarbeit.

Diese Arbeit ist für uns eine erste vertiefte Erfahrung mit dem Gebiet der künstlichen Intelligenz. Wir erhoffen uns durch diese Erfahrung einen erweiterten Horizont, neues Wissen und verbesserte Programmierkenntnisse.

Vielen Dank an unseren Betreuer, Dr. Nicolas Ruh, für die hilfreichen Vorschläge, die ausgeprägte Beratung und das Vertrauen in uns. Vielen Dank auch an Dieter Koch für die Zweitbeurteilung und an Günther Wasser für das Korrekturlesen dieser Arbeit.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Der Computer ist ein Werkzeug, das dem Menschen Arbeit abnehmen kann. Um komplizierte Aufgaben zu übernehmen, muss sich der Computer jedoch an menschliches Verhalten, menschliches Urteilsvermögen und an menschliche Intelligenz annähern. Somit benötigt der Computer oder das steuernde Computerprogramm eine künstliche Intelligenz. Die Entwicklung eines intelligenten Computerprogrammes birgt verschiedene Herausforderung. Der fähigste und am weitesten verbreitete Ansatz an diese Herausforderungen liefert Machine Learning. Diese Arbeit ist eine Untersuchung Im Bereich Machine Learning. Spezifischer befindet sich die Arbeit im Bereich von Deep Reinforcement Learning, einem Teilgebiet von Machine Learning.

Die Fragestellung der Untersuchung lautet: Inwiefern kann eine künstliche Intelligenz lernen, Strichbilder auf eine physische Weise nachzuzeichnen?

Für ein gegebenes Strichbild soll die künstliche Intelligenz (KI) erlernen, ein möglichst gleiches Bild daneben zeichnen zu können. Der Prozess des Nachzeichnens ist dabei durch verschiedene Kriterien definiert, die in dieser Arbeit beschrieben werden. Die KI soll das Nachzeichnen von Strichbildern allgemein erlernen. Strichbilder können Ziffern, Buchstaben, Formen, Symbole und weitere einfache Zeichnungen sein. Die KI soll diese verschiedenen Motive vergleichbar gut nachzeichnen. Das Format der Zeichnungen ist dabei auf eine feste größe und eine Farbtiefe von 1 (schwarzweiss) beschränkt.

Nachzeichnen ist eine menschliche Tätigkeit. Menschen führen beim Zeichnen durch gewisse Handbewegungen einen Stift, wodurch das Nachzeichnen mit physischen Einschränkungen verbunden ist. Der Stift teleportiert sich nicht, sondern bewegt sich mit einer limitierten Geschwindigkeit. Die KI soll das Nachzeichnen mit ähnlichen physischen Einschränkungen erlernen. Das heisst, die KI soll lernen, einen Stift zu

führen. Die physischen Einschränkungen sind für die KI jedoch simuliert und im Vergleich zu der echten Welt vereinfacht. Trotzdem sollte es möglich sein, mit der KI einen zeichnenden Roboter zu steuern

Die Untersuchung behandelt folgende Unterfragen, welche die Fragestellung herunterbrechen und ausweiten.

- Wie kann die Architektur einer KI aussehen, die das Nachzeichnen erlernt?
- Wie lässt sich die Leistung der KI in ihrer Aufgabe beurteilen?
- Wie lässt sich die Leistung der KI in ihrer Aufgabe verbessern?
- Wie ändert sich die Leistung der KI für Strichbilder, die im Training nicht enthalten sind?
- Wie und inwiefern lässt sich das Verhalten der KI mit menschlichem Zeichnen vergleichen?
- Kann eine KI Strichbilder ohne Vorlage nachzeichnen?

Kapitel 2

Theoretische Grundlagen

Dieses Kapitel führt die Konzepte ein, die über die ganze Arbeit hinweg Anwendung finden. Auch die verwendeten Fachbegriffe werden in diesem Kapitel eingeführt. Es handelt sich dabei um Zusammenfassungen. Die Theorie wird auf den Teil reduziert, der für ein grundsätzliches Verständnis der Arbeit nötig ist. Weitere Informationen sind in den referenzierten Quellen einsehbar.

2.1 Machine Learning

Machine Learning ist ein Teilbereich der künstlichen Intelligenz. “Künstliche Intelligenz (KI) bezieht sich im Allgemeinen auf jedes menschenähnliche Verhalten durch eine Maschine oder ein System” (noauthor'what'nodate) Mit Maschinen und Systemen sind in den allermeisten Fällen Computer, beziehungsweise die steuernden Computerprogramme gemeint. Diese Computerprogramme bilden ein Modell von menschlichem Verhalten. Machine Learning Modelle entwickeln (oder erlernen) eine Mustererkennung durch die Analyse von Daten (noauthor'what'nodate-1). Mustererkennung bedeutet hier, dass der Algorithmus Zusammenhänge zwischen den analysierten Daten erkennt und auf dieser Basis Vorhersagen treffen kann. Vereinfacht gesagt, versucht ein Machine Learning Modell menschliches Urteilsvermögen zu erlernen (spaulding'is'2020).

Ein Beispielproblem für ein Machine Learning Modell ist die Erkennung von handgeschriebenen Ziffern. Ein Computerprogramm soll durch den Input eines Bildes mit einer handgeschriebenen Ziffer eine korrekte Beurteilung treffen, um welche Ziffer es sich handelt. Das heisst, der Output des Computerprogramms soll der Ziffer entsprechen, die auf dem Bild des Inputs zu sehen ist (siehe ??). Jedes Computerprogramm, das dieses Problem löst, fällt in den Bereich der künstlichen Intelligenz. Machine Learning

Modelle geben einen Ansatz für die Umsetzung eines solchen Computerprogramms.



Abbildung 2.1: Erkennung von handgeschriebenen Zahlen durch ein Machine Learning Modell. (eigene Abbildung)

Machine Learning Modelle, die das Beispielproblem lösen, basieren üblicherweise auf Supervised Learning. Das ist ein Teilbereich von Machine Learning, wobei das Machine Learning Modell aus Rückmeldungen der korrekten Beurteilung, der Zielvariable, als Reaktion auf ihre eigenen Beurteilungen lernt (**laurenz wuttke was 2021-1**). Die Zielvariable muss dabei im Voraus für jeden Datenpunkt in den analysierten Daten durch einen Menschen festgelegt sein (**trahasch 31 2020**). Das heißt, ausgedrückt durch den Fachbegriff, dass die Daten labeled sein müssen (**luis g serrano 21 2021**). Weitere Teilbereiche von Machine Learning sind Unsupervised Learning und Reinforcement Learning (**arora supervised 2020**). Siehe Abschnitt ?? für eine tiefgreifendere Einführung in Reinforcement Learning.

Machine Learning Modelle sind hauptsächlich in der Programmiersprache Python implementiert (**sadie bennett why 2019**). Tensorflow und Keras sind bekannte Machine Learning Frameworks für Python. Als Framework stellen diese beiden Hilfsmittel fertige Funktionen und Algorithmen bereit, die für die Entwicklung von Machine Learning Modellen nötig sind (**noauthor tensorflow 2015**)(**noauthor keras 2015**).

2.1.1 Funktionsweise eines Machine Learning Modelles

Dieser Abschnitt erklärt die Funktionsweise eines Machine Learning Modells, basierend auf dem Beispielproblem aus dem letzten Abschnitt (siehe ?? ??).

Bei den Daten, die das Machine Learning Modell analysiert, handelt es sich in diesem Fall um das MNIST Datenset (**yann lecun mnist nodate**). Dieses Datenset wurde 1998 vom NIST (National Institute of Standards and Technology) in den USA veröffentlicht und beinhaltet 70'000 Bilder von

handgeschriebenen Ziffern (yann'lecun'papers'nodate). Jedes Bild hat eine Auflösung von 28×28 Pixeln (siehe ?? für Beispiele).



Abbildung 2.2: Beispiele aus dem MNIST Datenset. (Eigene Abbildung)

Ein Machine Learning Modell durchläuft ein Training gefolgt von einer Testphase (noauthor'training'nodate). Während dem Training erlernt das Modell die Mustererkennung, um verlässliche Aussagen zu den Daten des Inputs zu treffen. Die Testphase misst die Genauigkeit des Modells, also die Wahrscheinlichkeit, mit der das Modell die richtige Lösung zu dem Input liefert. Nur in den seltensten Fällen erreicht diese Genauigkeit 100%. Das Modell garantiert somit nicht die richtige Lösung. Das Machine Learning Modell erlernt die Mustererkennung während dem Training durch die Analyse von Trainingsdaten aus einem Datenset. Das Modell gibt zu jedem Datenpunkt die Beurteilung, um welche Zahl es sich handelt. Das Datenset ist labeled (siehe ?? ??). Falls die Beurteilung des Modells nicht mit der bekannten korrekten Lösung übereinstimmt, passt sich das Modell auf automatisierte Weise an. Dadurch soll die Beurteilungen für zukünftige Datenpunkte genauer werden. Die Testphase misst die Genauigkeit des Modells auf Testdaten. Die Testdaten bestehen aus Datenpunkten, die in den Trainingsdaten nicht enthalten sind.

Zusammengefasst kann ein Machine Learning Modell Daten beurteilen und sich selbst anpassen, um die Beurteilungen zu verbessern. Diese Eigenschaften sind in künstlichen neuronalen Netzen (siehe ?? ??) vorhanden, wodurch diese in Machine Learning Modellen Anwendung finden.

2.1.2 Künstliche neuronale Netze

Ein neuronales Netz ist, im biologischen Sinne, "eine beliebige Anzahl Neuronen, die miteinander Verbunden sind" (noauthor'neuronales'2021). Ein Beispiel für ein neuronales Netz ist das menschliche Gehirn. Künstliche

2. THEORETISCHE GRUNDLAGEN

neuronale Netze modellieren biologische neuronale Netze in der Form von Programmcode (noauthor'artificial'nodate). Diese Arbeit behandelt künstliche neuronale Netze, nicht aber biologische. Somit handelt es sich bei jedem erwähnten neuronalen Netz, um ein künstliches neuronales Netz.

Der Grundbaustein eines neuronalen Netzes ist das Neuron. Im Modell stellt das Neuron ein Objekt dar, das eine beliebige Anzahl Inputs, aber nur einen Output hat (siehe ??) (pramoditha'concept'2021). Input und Output sind hierbei rationale Zahlen. Der Output des Neurons ist im einfachsten Modell, dem Perzeptron Neuron, grundsätzlich entweder 0 oder 1. Der Output ist 1, wenn die Summe der Inputs einen vorgegebenen Wert, den *Threshold*, überschreitet. Ansonsten ist der Output gleich 0. Jeder Input hat ein *Gewicht*, das einer rationalen Zahl entspricht. Vor der Addition der Inputs wird jeder Input mit seinem Gewicht multipliziert. Die Grösse des Gewichtes bestimmt somit den Einfluss des zugehörigen Inputs auf den Output des Neurons. (nielsen'neural'2015)(simplilearn'what'2021)

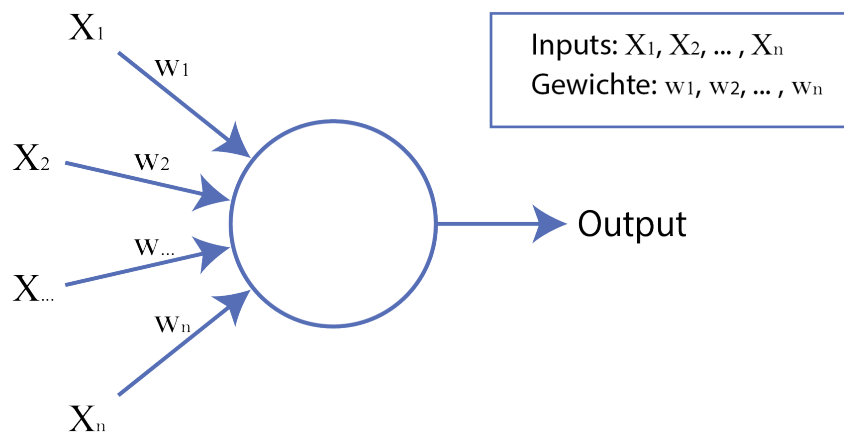


Abbildung 2.3: Perzeptron Neuron. (eigene Abbildung)

Neuronale Netze in Machine Learning Modellen verwenden kompliziertere Neuronen als das Perzeptron Neuron, wie zum Beispiel das Sigmoid Neuron. Die Neuronen unterscheiden sich in ihrer *Activation Function* und somit im Verhalten ihres Outputs (pragati'baheti'activation'2022). So nimmt der Output im Sigmoid Neuron beispielsweise auch Werte zwischen 0 und 1 an, in einem stetigen Übergang zwischen den beiden Grenzen (siehe ??) (kumar'sigmoid'2019).

Softmax und *ReLU* (rectified linear unit) sind weitere Activation Functions. ReLU ist die am weitesten verbreitete Activation Function in Deep Learning

und wird häufig für *Hidden Layers* verwendet (source...). Softmax wird im Vergleich meistens für die Activation Function des *Output Layers* verwendet. Diese Begriffe werden im nächsten Abschnitt genauer erläutert. Softmax bestimmt die Ausgabe aus dem Output Layer so, dass die Werte von allen Neuronen sich zu Eins addieren. Softmax konvertiert somit die Ausgabe aus einem neuronalen Netz in eine Wahrscheinlichkeitsverteilung. Softmax kann mit einem Parameter, genannt Temperatur, kombiniert werden. Dieser Parameter beeinflusst die Standardabweichung der Wahrscheinlichkeitsverteilung. Mit steigender Temperatur wird die Verteilung dabei uniformer und mit sinkender Temperatur nimmt die Differenz zwischen den einzelnen Werten zu.

2. THEORETISCHE GRUNDLAGEN



Abbildung 2.4: Vergleich des Outputs eines Perzeptron Neurons und eines Sigmoid Neurons. (eigene Abbildung)

Neuronale Netze sind Verbindungen von Neuronen. Dabei dient der Output eines Neurons als Input für andere Neuronen. Der Output eines Neurons kann gleich für mehrere Neuronen ein Input sein. Die Neuronen sind in *Layers* geordnet (siehe autoreflectors). Neuronale Netze haben mindestens einen *Input Layer* und einen *Output Layer* (nielsen'neural'2015)(ognjanovski'everything'2020). Der Input Layer umfasst die Daten, welche das neuronale Netz beurteilen soll. Im Beispielpapier (siehe ?? ??) bestünde der Input Layer aus 28×28 Neuronen, wobei jedes Neuron die Graustufe (durch einen Wert von 0 bis 255) eines Pixels im Bild beschreibt. Der Input ist in diesem Fall zweidimensional. Die Dimensionen sind allerdings flexibel. Der Output Layer besteht im Beispiel aus 10 Neuronen, wobei jedes Neuron einer Beurteilung entspricht (das zweite Neuron beschreibt zum Beispiel die Ziffer Zwei als Beurteilung). Dasjenige Neuron mit dem höchsten Output entspricht der Beurteilung des neuronalen Netzes. (siehe ??).

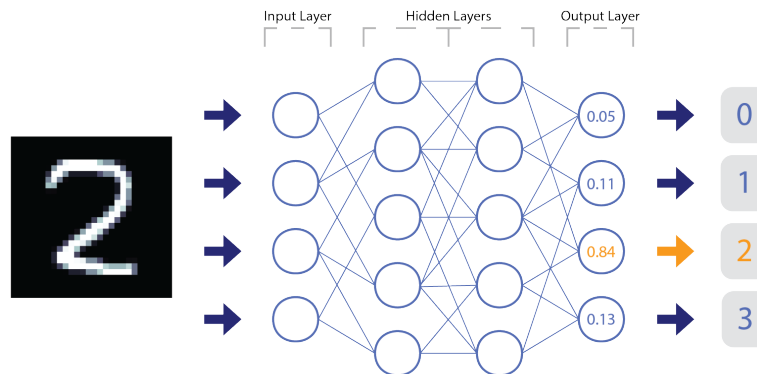


Abbildung 2.5: Neuronales Netz mit beschrifteten Layers. (eigene Abbildung)

Zwischen dem Input Layer und dem Output Layer kann es weitere *Hidden Layers* geben (malik'what'2019). Unterschiedliche Arten von Hidden Layers mit unterschiedlichen Funktionen existieren. Zwei der meist verwendeten Layers sind Fully Connected (Dense) Layers und Convolutional Layers (unzueta'convolutional'2022). In Fully Connected Layers dient der Output von jedem Neuron als Input für jedes Neuron in der nächsten Layer. In Convolutional Layers trifft das nicht zu (siehe ??). Die Funktion von Convolutional Layers umfasst, wichtige Merkmale aus dem Input hervorzuheben (deshpande'beginners'nodate). Concatenation Layers (jayawardana'concatenating'2021) sind eine weitere Form von Hidden Layers, die zwei verschiedene Layers als Input haben und diese verbindet. Machine Learning Modelle werden ab mehr als einer Hidden Layer als Deep Learning Modelle bezeichnet (jan-dirk'kranz'deep'2019).

Ein Machine Learning Modell passt während dem Training (siehe ?? ??) einzelne Gewichte im neuronalen Netz an, in der Hoffnung, dass die Genauigkeit der Beurteilung mit den angepassten Gewichten grösser ist. Die genaue Anpassung erfolgt in den meisten Machine Learning Modellen durch den Backpropagation Algorithmus (ognjanovski'everything'2020)(david'e'rumelhart'learning'nodate).

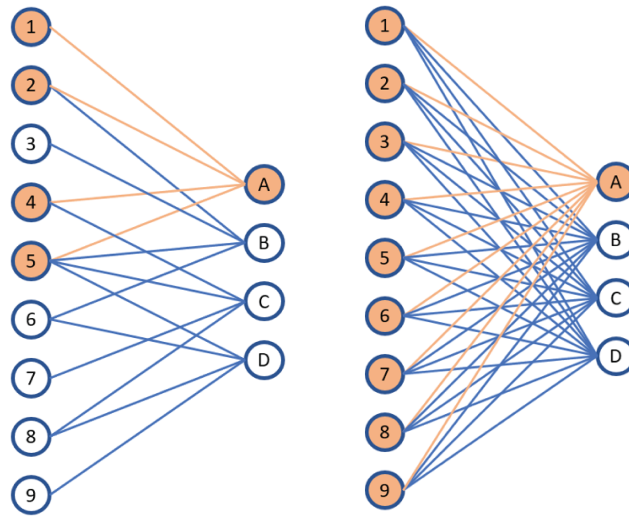


Abbildung 2.6: Vergleich zwischen Convolutional Layers (links) und Fully Connected Layers (rechts). (unzueta'convolutional'2022)

2.1.3 Hyperparameter

Machine Learning Modelle umfassen verschiedene Hyperparameter. Diese beschreiben unter anderem, wie lange das Training läuft oder wie stark sich das Modell nach einer falschen Beurteilung anpasst. Hyperparameter beeinflussen das Lernverhalten des Modells (nyuytiymbiy'parameters'2022), aber ihr optimaler Wert ist im Voraus nicht bekannt.

Hyperparameter können unter anderem durch den Bayesian Optimization Algorithmus optimiert werden (agnihotri'exploring'2020)(paretos'bayesian'2021). Dieser Algorithmus versucht, den Output einer Black Box Funktion zu maximieren oder zu minimieren (garnett'bayesian'nodate). Eine Black Box ist ein häufig komplexes System, dessen inneren Vorgänge nicht betrachtet werden (noauthor'black'2021). Bei einer Black Box Funktion ist folglich der Input und der Output bekannt, während die Verarbeitung des Inputs zum Output nicht betrachtet wird (siehe ??).



Abbildung 2.7: Prinzip einer Black Box Funktion. (noauthor'black'2021)

Machine Learning Modelle werden häufig als Black Box Funktionen angesehen, da die genauen Vorgänge des rechnerisch aufwendigen Trainings

durch einen aussenstehenden Betrachter nicht oder nur schwer erfassbar sind. (robbins'machine'2017). Um ein Machine Learning Modell als eine Black Box Funktion für den Bayesian Optimization Algorithmus zu verwenden, werden die zu optimierenden Hyperparameter als Input und eine Zielvariable als Output definiert. Die Zielvariable des Outputs entspricht dabei einem Wert, der die Leistung des Modells widerspiegelt und durch den Algorithmus maximiert werden soll. Ein Beispiel für die Zielvariable wäre die Genauigkeit des Machine Learning Modells (siehe ?? ??). Die inneren, nicht betrachteten Vorgänge in der Black Box Funktion entsprechen in diesem Fall dem Training des Modells.

Der Bayesian Optimization Algorithmus kann bis zu 20 Hyperparameter zuverlässig optimieren (moriconi'high-dimensional'2020). Der Algorithmus führt die Black Box Funktion für eine bestimmte Anzahl Iterationen mit jeweils verschiedenen Parametern durch. Die Wahl der Parameter basiert dabei auf Bayes' Theorem (garnett'bayesian'nodate). Diejenigen Parameter, die den höchsten gefundenen Wert der Zielvariable auslösen, werden gespeichert.

2.2 Reinforcement Learning

Reinforcement Learning bedeutet zusammengefasst: Lernen durch Interaktion mit einer Umgebung. (osinski'what'2018). Spezifischer soll ein Machine Learning Modell durch Rückmeldungen aus einer Umgebung ein bestimmtes Verhalten erlernen.

Reinforcement Learning Modelle führen somit die Umgebung ein. Anders als bei Supervised Learning und Unsupervised Learning (siehe ?? ??) sind die Daten, aus denen das Modell lernen soll, im Voraus nicht bekannt. Reinforcement Learning Modelle trainieren somit nicht oder nicht nur auf den Daten aus einem Datenset. Das liegt in der Natur der Umgebung, die häufig zu viele verschiedene Zustände einnehmen kann, als dass diese in einem Datenset gesammelt werden könnten. Ein Machine Learning Modell lernt aus einer Umgebung, indem es durch Beurteilungen (siehe ?? ??) mit dieser interagiert und dadurch Erfahrungen sammelt. (piyush'verma'what'2021)

Als Beispiel kann die echte Welt als eine Umgebung angesehen werden. Der Mensch wäre in diesem Fall das Reinforcement Learning Modell. Der Mensch lernt die Eigenschaften seiner Umgebung durch Interaktionen mit dieser kennen. Beispielsweise lernt ein Mensch die Schwerkraft durch das Hinfallen kennen. Durch diese Erfahrungen kann der Mensch ein gewisses Verhalten, zum Beispiel das Laufen, erlernen. Reinforcement Learning Modelle imitieren dieses Lernverhalten. So verwendet die Robotik häufig Reinforcement Learning, um einen Roboter das Laufen erlernen zu lassen.

Die Umgebung, mit der das Reinforcement Learning Modell lernt, ist dabei häufig nicht echt, sondern simuliert.

2.2.1 Funktionsweise von Reinforcement Learning

Dieser Abschnitt umfasst eine genauere Erklärung eines Reinforcement Learning Modells, in diesem Fall Deep Q-Learning (mnih'playing'2013), unter der Verwendung der korrekten Fachbegriffe.

Ein Reinforcement Learning Modell umfasst eine *Umgebung* und einen *Agent*. Der Agent ist dasjenige Element in der Umgebung, welches mit dieser interagiert und daraus lernt (sutton'reinforcement'2014). Die Umgebung verändert sich in Zeitschritten, genannt *Steps*. In jedem Step führt der Agent eine *Action* aus, welche die Umgebung beeinflusst. Die Entscheidung, welche Action der Agent wählt, basiert auf einer *Observation* der Umgebung (mnih'playing'2013). Die Observation umfasst alle Daten der Umgebung, die für die Entscheidung relevant sind. Sein Entscheidungsprozess findet durch ein neuronales Netz statt (siehe ?? ??). Der Input in dieses neuronale Netz ist die Observation der Umgebung und der Output beschreibt die Action, die der Agent ausführt. Jedes Neuron des Outputs beschreibt eine spezifische Action des Agents. Der Agent kann somit nur eine feste Anzahl Actions ausführen. Alle Actions zusammen werden *Action-Space* (sutton'reinforcement'2014) genannt. Jede Action im Action-Space besitzt einen *Q-Value*, der dem Output des zugehörigen Neurons entspricht (siehe ??) (wang'deep'2021).

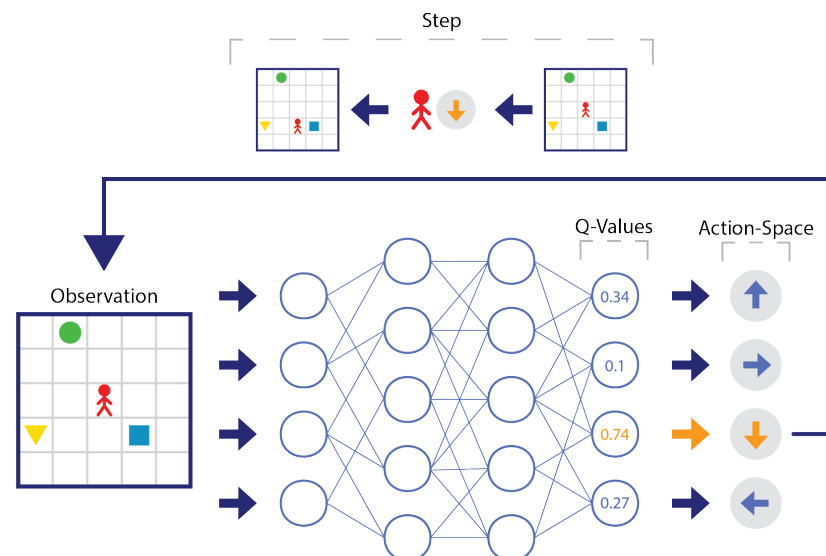


Abbildung 2.8: Funktionsweise eines Reinforcement Learning Modells. (eigene Abbildung)

Die schlussendliche Entscheidung der auszuführenden Action wird durch eine *Policy* bestimmt. Softmax und Epsilon Greedy sind 2 Beispiele für eine Policy (**sutton'reinforcement'2014**).

Die Epsilon-Greedy Policy sieht vor, dass die Entscheidung mit einer Wahrscheinlichkeit von ϵ (Epsilon) auf eine zufällige Action fällt. Ansonsten fällt die Entscheidung auf diejenige Action mit dem höchsten Q-Value. Durch die zufälligen Actions erkundet der Agent die Umgebung, in der Hoffnung, auf bessere Optionen für zukünftige Entscheidungen zu stossen. (**rajendra'koppula'exploration'nodate**). Die Softmax Policy wählt Actions auf Basis einer Wahrscheinlichkeitsverteilung. Jede Action wird also mit einer gewissen Wahrscheinlichkeit ausgeführt. Die Wahrscheinlichkeit für eine Action entspricht dem Q-Value dieser Action. Die Wahrscheinlichkeitsverteilung berechnet sich dabei aus der Softmax Activation Function von dem Output Layer des neuronalen Netzes.

Die Umgebung wird durch die Actions des Agents beeinflusst. Dieser Einfluss wird durch die *Reward-Function* gemessen. Die Reward-Function gibt eine rationale Zahl, den *Reward* aus (**sutton'reinforcement'2014**). Umso grösser der Reward, desto positiver ist der Effekt einer Action auf die Umgebung und umgekehrt. Ein positiver Einfluss auf die Umgebung durch eine Action ist so definiert, dass der Agent durch die Action das gewünschte Verhalten vorzeigt. Die Reward-Function definiert, welches Verhalten welchen Reward erzielt. Der Q-Value der gewählten Action wird mit dem Reward (und dem maximalen Q-Value aus den nächsten möglichen Actions) addiert. Diese Formel nennt sich Bellman-Gleichung (**mnih'playing'2013**). Der neue Q-Value hat somit einen kleineren Wert, wenn der Reward negativ ist, und einen grösseren Wert, wenn der Reward positiv ist. Das neuronale Netz wird daraufhin so trainiert, dass der Output für das Neuron, dessen Action ausgeführt wurde, näher am neu berechneten, besseren Q-Value ist (siehe ??). Der schlussendliche Effekt ist, dass Actions, die einen positiven Reward auslösen, wahrscheinlicher gewählt werden, und umgekehrt Actions, die einen negativen Rewards auslösen, unwahrscheinlicher gewählt werden. Der Agent versucht insgesamt durch seine Actions einen möglichst hohen akkumulierten Reward zu erzielen (**sutton'reinforcement'2014**). Der akkumulierte Reward entspricht der Summe der Rewards aus jedem Step (in einer Episode. Beachte den nächsten Abschnitt).

Das Training läuft in *Episodes* (**sutton'reinforcement'2014**). Eine Episode umfasst eine gewisse Anzahl Steps und am Anfang jeder Episode wird die Umgebung in einen Ausgangszustand zurückgesetzt. Die Resultate der Steps werden in dem *Replay-Buffer* gespeichert. Dazu gehören die Observations der Umgebung, die jeweiligen Actions und der jeweilige Reward. Der Replay-Buffer enthält Speicherplatz für eine bestimmte Anzahl Steps. Während dem Training werden zufällige Steps aus dem Replay-Buffer gewählt, auf

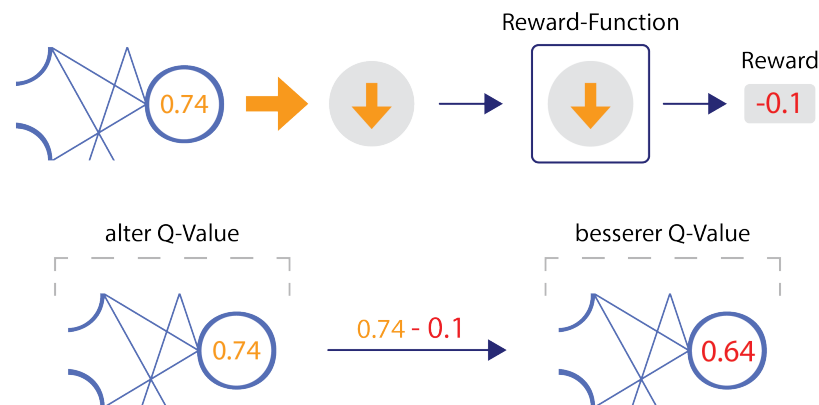


Abbildung 2.9: Funktionsweise einer Reward-Function. (eigene Abbildung)

die das neuronale Netz trainiert. Das neuronale Netz trainiert also auf Daten aus der Vergangenheit der Umgebung. Diese Strategie nennt sich Experience Replay ([mnih'playing'2013](#)). Ausserdem trainiert das neuronale Netz jeweils mit einem *Batch* an Steps, also mit einer gewissen Anzahl an Steps gleichzeitig. Der Replay-Buffer und der Batch sichern zu, dass das neuronale Netz mit einer grossen Vielfalt an Steps trainiert. Das bewirkt ein stabileres Lernverhalten als ein chronologisches Training auf einzelne Steps. ([phd'how'2021](#)).

2.3 Verwandte Arbeiten und Themen

Das Nachzeichnen von Strichbildern ist ein Teilbereich von der Tätigkeit des Zeichnens allgemein. Es gibt verschiedene Ansätze, um einen Computer zeichnen zu lassen. Ein häufiger Ansatz ist *Stroke-Based Rendering*. Stroke-Based Rendering beschreibt das Zeichnen von Bildern durch das Platzieren von Elementen wie Strichen ([aaron'hertzmann'stroke-based'2002](#)). Beispiele für Arbeiten in diesem Bereich sind Strokenet ([zheng'strokenet'2018](#)) und "Learning to Paint With Model-based Deep Reinforcement Learning" ([huang'learning'2019](#)). Andere Ansätze simulieren die Führung eines Stiftes. (siehe ??) Ein Beispiel dafür ist Doodle-SDQ ([zhou'learning'2018](#)).

2.3.1 Doodle-SDQ

Doodle-SDQ ist ein Computerprogramm, das unter anderem durch Reinforcement Learning und spezifischer durch Deep Q-Learning (siehe ??) erlernt, Strichbilder aus dem Google QuickDraw Datenset ([noauthor'quick'2022](#)) nachzuzeichnen. Nachfolgend sind diejenigen Aspekte von Doodle-SDQ beschrieben, die für diese Arbeit relevant sind.

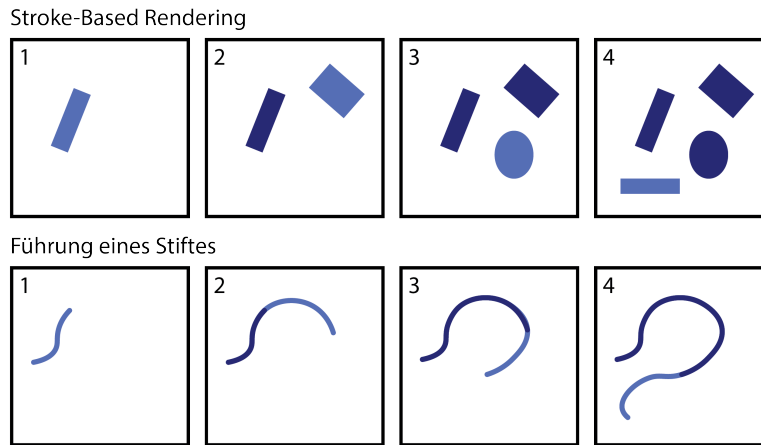


Abbildung 2.10: Vergleich zwischen Stroke-Based Rendering und dem Führen eines Stiftes. (eigene Abbildung)

Die QuickDraw Bilder, die Doodle-SDQ nachzeichnen soll, sind zu einer einheitliche Grösse von 84×84 Pixeln verarbeitet (**zhou'learning'2018**). Der Agent kann sich auf einer leeren Zeichenfläche von derselben Grösse bewegen und zeichnen. Die Umgebung umfasst diese Zeichenfläche, den Agent und das abzuzeichnende Bild.

Der Agent kann sich durch eine Action pro Step auf einem beliebigen Pixel in einem 11×11 Feld, in dessen Zentrum er ist, bewegen. Der Agent kann ausserdem jede dieser Bewegungen im zeichnenden Zustand oder im nicht zeichnenden Zustand ausführen. Der Action-Space hat somit insgesamt eine Grösse von $2 \cdot 11 \cdot 11 = 242$ Actions (**zhou'learning'2018**). Im zeichnenden Zustand wird ein Strich auf der Zeichenfläche zwischen der alten und der neuen Position des Agents gezeichnet. Der Agent begeht 100 Steps pro Episode. Eine neue Episode entspricht dabei einem neuen Bild, das abgezeichnet werden soll.

Die Observation der Umgebung und somit der Input in das neuronale Netz (siehe ?? ??) ist in zwei Teile gegliedert: der Global Stream und der Local Stream (**zhou'learning'2018**). Der Global Stream hat eine Form von $84 \times 84 \times 4$. Der Input ist somit dreidimensional. Die Form kann als 4 aufeinandergestapelte Bilder (genannt *Channels*) angesehen werden, die jeweils eine Grösse von 84×84 Pixeln haben. Dabei beschreibt eine rationale Zahl den Wert von jedem Pixel in einem Bild. Das erste Bild im Global Stream ist die Vorlage, die abgezeichnet werden soll. Das zweite Bild ist die Zeichenfläche im aktuellen Zustand. Das dritte Bild beschreibt die Position des Agents durch seine relative Entfernung zu jedem Punkt auf der Zeichenfläche. Das vierte Bild beschreibt, ob der Agent im zeichnenden Zustand ist oder nicht. Wenn alle Pixel dieses letzten Bildes den Wert 1 haben, ist der Agent im zeichnenden Zustand. Wenn alle Pixel den Wert 0 haben,

ist der Agent nicht im zeichnenden Zustand. Der Local Stream hat eine Form von $11 \times 11 \times 2$. Er ist somit auch dreidimensional und beschreibt zwei gestapelte Bilder (zwei Channels). Das erste Bild umfasst die Vorlage in dem 11×11 Bereich (bezeichnet als Local image Patch (**zhou'learning'2018**)), in dem sich der Agent in einem Step bewegen kann. Das zweite Bild beschreibt denselben Bereich von der Zeichenfläche (**zhou'learning'2018**). Der Global Stream und der Local Stream werden durch eine Concatenation Layer (siehe ?? ??) zusammengeführt. Vor der Zusammenführung hat das neuronale Netz mehrere Convolutional Layers und nach der Zusammenführung jeweils einen Dense Layer und Output Layer (ebenfalls Dense).

Kapitel 3

Methode

Die Methode dieser Untersuchung besteht darin, die in der Fragestellung beschriebene künstliche Intelligenz (KI) zu entwickeln und dessen Leistung auszuwerten (siehe ?? ??). 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 ?? ??). Der andere Teil umfasst die Entwicklung der KI. Dazu gehört die Entwicklung einer grundlegenden Architektur (siehe ?? ??), sowie die Vollendung der KI mit verschiedenen Variationen und Ansätzen (siehe ?? ??). Die Variationen sind dabei Versuche, die Leistung der KI zu maximieren oder ihr Verhalten zu verändern. Eine spezielle Variation, die sich dabei von der ursprünglichen Absicht entfernt, ist die Umwandlung in eine generative KI, die nicht mehr Nachzeichnet und stattdessen eigene Zeichnungen ohne Vorlage kreiert (siehe ?? ??).

3.1 Grundprogramm

Das Grundprogramm ist eine flexibel anwendbare Architektur der KI, die als Grundlage für eine Vielzahl an Variationen dient. Das Grundprogramm bietet dabei eine Trainingsumgebung für die KI, eine Zeichenumgebung und das Reinforcement Learning Modell (Deep Q-Learning), zusammen mit dem Agent und dem neuronalen Netz (siehe ?? ??). Das Grundprogramm beinhaltet dabei keine Reward-Funktion und ist somit keine funktionale Version der KI. Das Grundprogramm ist in Python unter der Verwendung des Keras Frameworks implementiert (siehe ?? ??).

3.1.1 Doodle-SDQ als Basis

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

3. METHODE

auf die Form des Inputs, des Outputs und den Hidden Layers, grösstenteils 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 bewegt. Das Grundprogramm wird auf das Nachzeichnen von Ziffern trainiert. Die Ziffern stammen aus dem MNIST Datenset (siehe ?? ??) und haben somit eine Grösse von 28×28 Pixeln. Die Fläche, worauf sich der Agent bewegen kann, hat folglich 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 in der folgenden Abbildung angegeben (siehe ??).

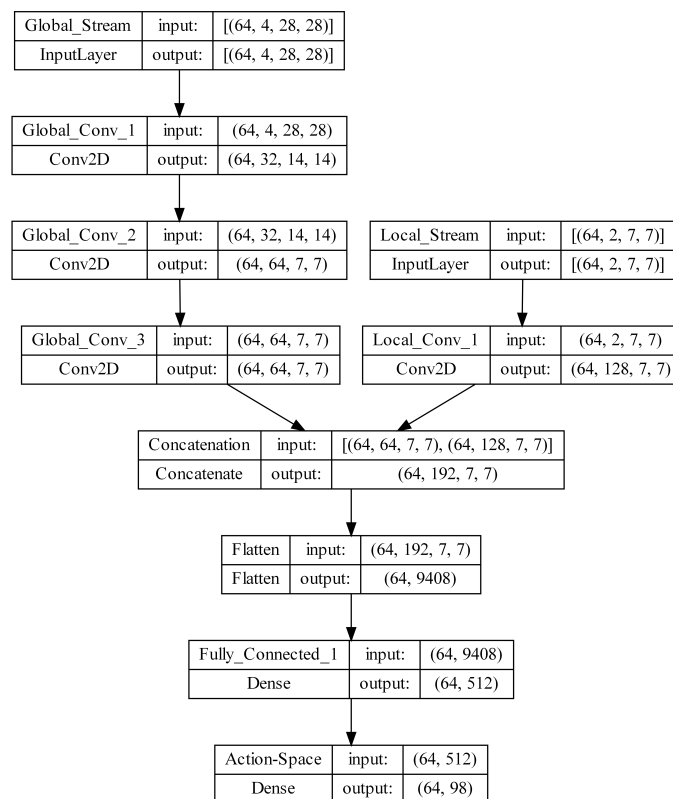


Abbildung 3.1: Architektur des neuronalen Netzes im Grundprogramm (eigene Abbildung, mit Keras erstellt). Jeder Block repräsentiert einen Layer. Die Form des Inputs und des Outputs ist von jedem Layer angegeben

Der Local Stream und damit auch der Local image Patch schrumpfen 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 ??).

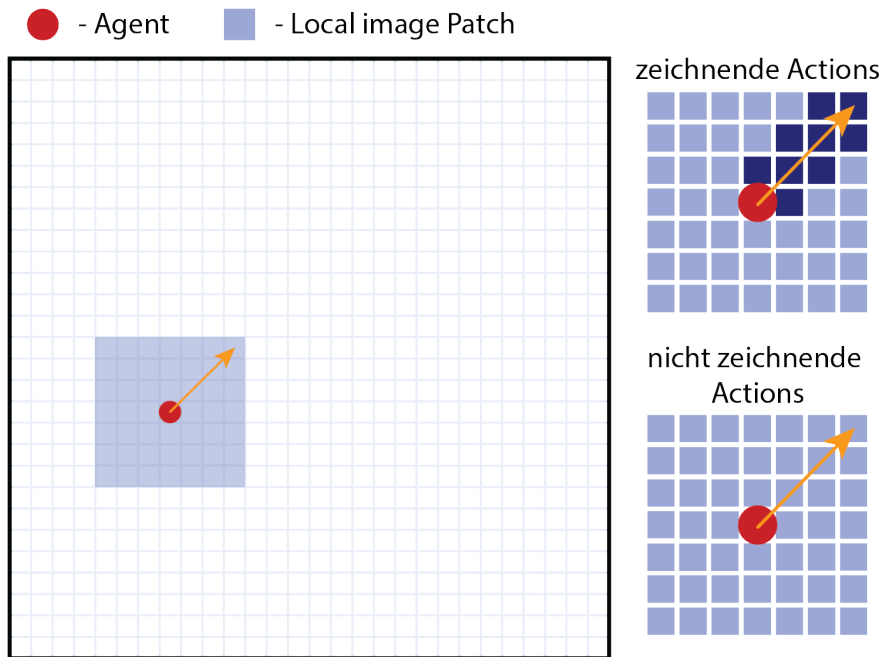


Abbildung 3.2: Action-Space im Grundprogramm

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

Das Grundprogramm verwendet als Policy für die Wahl der Actions standardmässig Epsilon-Greedy. Die Softmax Policy ist allerdings auch implementiert und nach belieben verwendbar.

3.1.2 Erweiterungen

Die folgenden Eigenschaften des Grundprogramms sind Erweiterungen der Architektur von Doodle-SDQ.

Die erste Erweiterung beschreibt die Behandlung von illegalen Actions. Alle Actions, die den Agent über die vorgegebene Zeichenfläche hinaus positionieren würden, sind nicht zulässig. Diese Actions sind für den Agent nicht wählbar und ihr optimaler Q-Value (siehe ?? ??) 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.

Die zweite Erweiterung betrifft den Action-Space mit der Einführung einer Stopp Action. Wenn der Agent diese Action wählt, bricht die Zeichnung ab. Der Agent kann somit frei entscheiden, wann die Zeichnung fertig ist. Die Stopp Action kann während dem Training dem Agent verboten werden. In diesem Fall wird die Action als unzulässig behandelt.

3.1.3 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 ??). So stimmen die Bilder mit den Zeichnungen, die der Agent produzieren kann, überein.

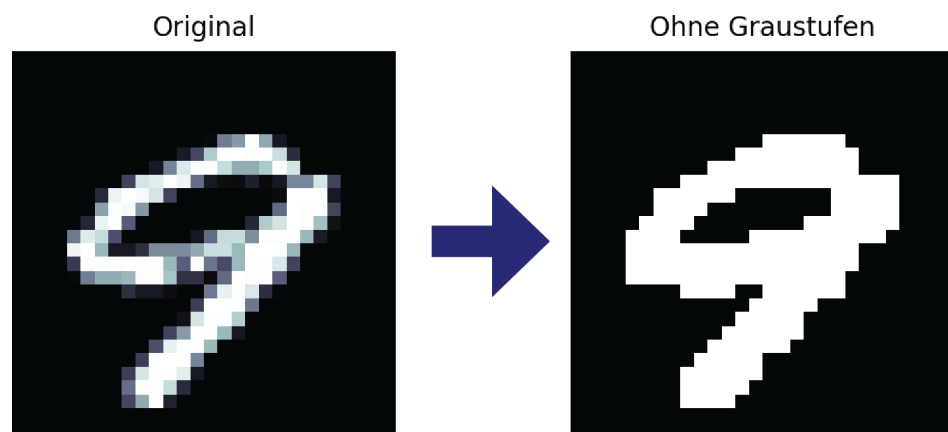


Abbildung 3.3: Entfernung der Graustufen im MNIST Datenset. (eigene Abbildung)

Das Grundprogramm trainiert mit 5'000 Bildern, von denen jede Ziffer 500 Bilder ausmacht. Die restlichen Bilder in den Trainingsdaten sind für mögliche Erweiterungen aufgehoben. Der Agent zeichnet jedes der 5'000

Bilder ein Mal und trainiert somit für 5'000 Episodes. Der Agent zeichnet für 64 Steps pro Episode, falls die Stopp Action nicht früher gewählt wird.

Die Hyperparameter aller Versionen der KI sind durch den Bayesian Optimization Algorithmus optimiert (siehe ?? ??). Die Implementierung des Algorithmus in Python stammt von (fernando'nogueira'bayesian'2014). Der Algorithmus ändert sich für verschiedene Variationen der KI nicht und ist somit Teil des Grundprogramms. Mit jeder Iteration des Bayesian Optimization Algorithmus trainiert das Reinforcement Learning Modell für eine vom Algorithmus selbst bestimmte Anzahl Episodes. Die Zielvariable, die durch den Bayesian Optimization Algorithmus maximiert werden soll, wird am Ende jeder Iteration des Trainings in der Testumgebung berechnet (siehe ?? ??). Unter welchem Kriterium (siehe ?? ??) der Wert der Zielvariable berechnet wird ist frei wählbar basiert, ist frei wählbar.

3.2 Evaluation der Leistung

In diesem Unterkapitel sind die Kriterien definiert, welche die Leistung der künstlichen Intelligenz evaluieren. Spezifischer entsprechen die Kriterien einer Definition des Nachzeichnens, nach der die künstliche Intelligenz bewertet wird. Für eine präzise und objektive Evaluation sind alle Kriterien durch einen Zahlenwert repräsentiert. Die Kriterien und ihre jeweilige Berechnung werden nachfolgend beschrieben.

3.2.1 Prozentuale Übereinstimmung

Dieses Kriterium beschreibt die prozentuale Übereinstimmung der weissen (gezeichneten) Pixel zwischen der Vorlage und der Zeichnung der KI (siehe ?? ??). Der Wert K dieses Kriteriums zu dem Step t berechnet sich aus folgender Formel:

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

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. Die Pixel, die nicht übereinstimmen, zählen negativ für $G(t)$. $G(t)$ und somit auch $K(t)$ können dadurch auch negative Werte annehmen. Der maximale Wert von $K(t)$ ist 1, was einer prozentualen Übereinstimmung von 100% entspricht (siehe ??).

3.2.2 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

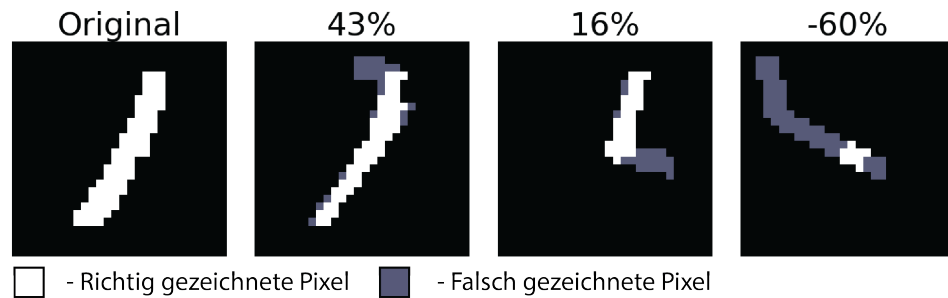


Abbildung 3.4: Drei Beispiele für den Wert des Kriteriums der Übereinstimmung. (eigene Abbildung)

den Wert 1. Wird in der Vorlage eine Fünf erkannt, aber in der Zeichnung eine Vier, hat das Kriterium den Wert 0

Das erkannte Motiv wird durch eine zweite KI beurteilt (siehe ?? ??). Diese klassifizierenden Machine Learning Modelle sind spezifisch auf Bilder ohne Graustufen trainiert, welche die nachzeichnende KI dieser Arbeit produzieren kann. Der Output Layer der klassifizierenden KI hat Softmax mit einer hohen Temperatur als Activation Function (siehe ?? ??). Das versichert, dass die KI nur eine eindeutige Klassifizierung trifft wenn die Wahrscheinlichkeit für dessen Richtigkeit hoch ist. Für die verschiedenen Arten von Strichbildern, welche die KI zeichnen soll, werden spezifische klassifizierende Machine Learning Modelle mit den zugehörigen Datensets trainiert. Diese Modelle, zusammen mit ihrer Genauigkeit, sind in der folgenden Tabelle (siehe ??) aufgeführt. Das Neuronale Netz der Modelle stammt aus einem online Machine Learning Kurs (**wang·deep·2021**).

Art	Trainiert mit	Genauigkeit [%]
Ziffern	MNIST	99
Buchstaben	EMNIST Letters	91
Strichbilder von Objekten	Auswahl aus QuickDraw	98

Tabelle 3.1: Vortrainierte Modelle

3.2.3 Geschwindigkeit

Dieses Kriterium beschreibt, wie schnell die Zeichnung der KI fertig ist. Der Wert des Kriteriums entspricht dabei der Anzahl Steps bis zur Fertigstellung. Der Punkt der Fertigstellung ist dabei bei der letzten zeichnenden Action des Agents definiert. Falls der Agent sich also nach dem Zeichnen weiterhin nicht-Zeichnende Actions ausführt, so werden diese nicht mehr zu der Zeichnung

gezählt, weil die Zeichnung nicht beeinflusst wird. Tiefere Werte in diesem Kriterium entsprechen einer höheren Geschwindigkeit

3.2.4 Zeichnende Zeit

Das Kriterium der zeichnenden Zeit beschreibt, in wie vielen Steps pro Episode die KI eine zeichnende Action ausführt. Dieser Wert ist als ein prozentualer Anteil angegeben und berechnet sich somit aus der Anzahl zeichnender Steps dividiert durch die Anzahl aller Steps pro Zeichnung. In diesem Kriterium sind generell hohe Werte erwünscht.

3.2.5 Übermalung

Das Kriterium der Übermalung beschreibt, wie viele Pixel pro Zeichnung mehrfach bemalt werden. Jeder Pixel, auf dem die KI also zwei oder mehr Mal malt, erhöht den Wert dieses Kriteriums um Eins. Jeder Pixel trägt dabei höchstens einmal zu diesem Kriterium bei. Das heisst, dass vielfach bemalte Pixel gleich behandelt werden wie zweimal bemalte Pixel. Möglichst tiefe Werte entsprechen generell der besten Leistung in diesem Kriterium

3.3 Variationen

Dieses Unterkapitel beschreibt die Variationen der KI. Jede Variation erweitert das Grundprogramm (siehe ?? ??) zu einer funktionalen, nachzeichnenden KI. Die meisten der Variationen unterscheiden sich dabei hauptsächlich im Fokus auf die definierten Kriterien (siehe ?? ??). Jede der Variationen versucht, die Leistung in einem ausgewählten Kriterium zu maximieren. Eine Anpassung der Reward-Function (siehe ?? ??) zielt auf diesen Effekt ab. Die verschiedenen Variationen sind teilweise untereinander kombinierbar. Eine der Variationen ist auf kein Kriterium spezialisiert und versucht stattdessen allgemein das Verhalten der KI zu verbessern.

3.3.1 Basis Reward-Function

Die Basis Reward-Function ist die einfachste Erweiterung des Grundprogramms. Diese Reward-Function implementiert das Kriterium der prozentualen Übereinstimmung (siehe ?? ??). Der Reward für eine Action berechnet sich aus der Differenz zwischen der prozentualen Übereinstimmung vor dem Ausführen der Action 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 folglich nicht der gesamten prozentualen Übereinstimmung zu einem Step. Stattdessen entspricht der Reward der Veränderung der prozentualen Übereinstimmung, ausgelöst durch die Action in einem Step. Der akkumulierte Reward (siehe ?? ??) entspricht dem absoluten Wert der prozentualen Übereinstimmung.

Die prozentuale Übereinstimmung ist das grundlegendste Kriterium für die Tätigkeit des Nachzeichnens und hat ausserdem den Vorteil, dass kleine Verbesserungen bereits zu einem positiven Reward führen können. Das ist in den ersten Episoden des Trainings für die KI von entscheidender Bedeutung. Aus diesem Grund ist die Basis Reward-Function zumindest in ähnlicher Form in allen anderen Variationen ebenfalls vertreten.

3.3.2 Spezialisierung 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 an, die unabhängig von der Art der Strichbilder ist (siehe ?? ??). Die klassifizierende KI kann allerdings jeweils nur eine Art von Strichbildern erkennen. Deswegen spezialisiert sich diese Variation allein auf das Nachzeichnen von Zahlen und verwendet dafür die entsprechende klassifizierende KI.

Die Variation verwendet die Stopp Action (siehe ?? ??). Sobald die nachzeichnende KI die Stopp Action wählt, überprüft die klassifizierende KI, ob das Bild erkennbar ist. Der Reward entspricht dabei dem Output des Neurons, welches die Zahl beurteilt, die nachgezeichnet werden sollte. Dieser Output wird als Reward um 0.8 verringert. Der maximale Reward für eine erkennbare Zahl ist somit $1 - 0.8 = 0.2$. Negative Rewards werden mit einem Faktor von 0.5 multipliziert, damit die KI nicht zu hohe negativen Rewards erhält.

3.3.3 Spezialisierung auf Geschwindigkeit

Diese Variation verwendet die Stopp Action in der Reward-Function um eine möglichst hohe Geschwindigkeit zu erzielen. Der Reward für die Geschwindigkeit berechnet sich dabei aus folgendem Prinzip: Umso früher der Agent die Stopp Action auswählt, desto höher ist der Reward für diese Action. Um zu verhindern, dass der Agent die Stopp Action direkt im ersten Step wählt, löst die Reward-Function erst ab einer prozentualen Übereinstimmung von mehr als 80% einen positiven Reward aus. Ansonsten ist der Reward der Stopp Action negativ. Der Reward über und unter dieser Schwelle von 80% ist eine statische Zahl. Eine prozentuale Übereinstimmung von 90% löst also keinen grösseren Reward aus als eine prozentuale Übereinstimmung von 80%. Experimente mit dynamischen Rewards

scheiterten bisher. Ein möglicher Grund dafür ist, dass die KI je nach Situation zu hohe oder zu niedrige Rewards erhält.

Der Reward wird mit einem Faktor multipliziert, der von der Geschwindigkeit abhängt. Dieser Faktor ist nicht linear und stattdessen durch die folgende Funktion definiert:

$$f_{speed} = 1 - \left(\frac{t}{64}\right)^3$$

(Siehe Abbildung Funktion) Dabei ist t der aktuelle Step und die Zahl 64 entspricht der maximalen Anzahl Steps, was somit den Wert der geringsten möglichen Geschwindigkeit ausmacht. Die Kurve der verwendeten Funktion verhindert, dass der Reward zu schnell abfällt.

3.3.4 Spezialisierung auf zeichnende Zeit

Diese Variation hat zum Ziel, den Wert des Kriteriums der zeichnenden Zeit zu maximieren. Das heisst, die KI soll sich so wenig wie möglich bewegen ohne dabei zu zeichnen. Dieser Effekt wird dadurch angestrebt, dass nicht zeichnende Aktionen einen negativen Reward von -0.05 auslösen. Ansonsten unterscheidet sich die Reward Function von der Basis Reward-Function nicht. Auch diese Variation verwendet die Stopp Action (spezifisch in Kombination mit der Spezialisierung auf Geschwindigkeit oder Erkennbarkeit), weil die KI nach einer gewissen Anzahl Steps keine nützlichen Actions mehr hat und ohne Stopp Action in diesem Fall unschuldig negative Rewards erhalten würde.

3.3.5 Spezialisierung auf keine Übermalung

Diese Variation versucht, die KI davon abzuhalten, Pixel mehrfach zu übermalen. Jeder übermalte Pixel senkt dabei den Reward der Basis Reward-Function um einen definierten Betrag. Pro Action ist allerdings das übermalen von drei Pixeln erlaubt und somit von negativen Rewards befreit. Diese Variation ist ohne die Stopp Action verwendbar, kann aber auch mit dieser kombiniert werden.

3.3.6 Physikalische Umgebung

Diese Variation spezialisiert sich auf kein Kriterium. Stattdessen verändert sie die Umgebung, sowie auch den Input und den Output des neuronalen Netzes (siehe ?? ??). Durch diese Veränderungen löst sich die Variation vom Grundprogramm.

Die Variation ergänzt die Umgebung durch physikalische Simulationen. 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

3. METHODE

sich der Agent pro Step bewegt. Die folgende Formel beschreibt, wie sich die Position des Agenten vom Step t bis zum nächsten Step $t + 1$ verändert:

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

$\vec{p}(t)$ 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 Agents 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 beschreiben Beschleunigungsvektoren im zeichnenden Zustand. Die anderen 21 Actions beschreiben dieselben Vektoren im nicht zeichnenden Zustand. Die 21 verschiedenen Beschleunigungsvektoren im Actions-Space sind in der folgenden Formation angeordnet: (siehe ??).

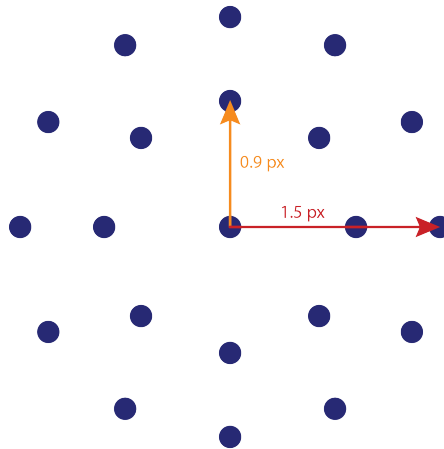


Abbildung 3.5: Action-Space in der physikalischen Umgebung. (eigene Abbildung)

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 in jedem Step, unabhängig von der gewählten Action, um 0.3 Pixel pro Step verringert. Das simuliert eine Reibungskraft, die auf den Agent einwirkt.

Um die Geschwindigkeit des Agents in die Observation (siehe ?? ??) zu integrieren, wird der Local image Patch (siehe ?? ??) verschoben. Im Grundprogramm entspricht der Mittelpunkt des Local image Patches genau der Position des Agents. Neu befindet sich der Mittelpunkt dort, wo sich der Agent laut seiner aktuellen Geschwindigkeit im nächsten Step 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 Step 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 ??).

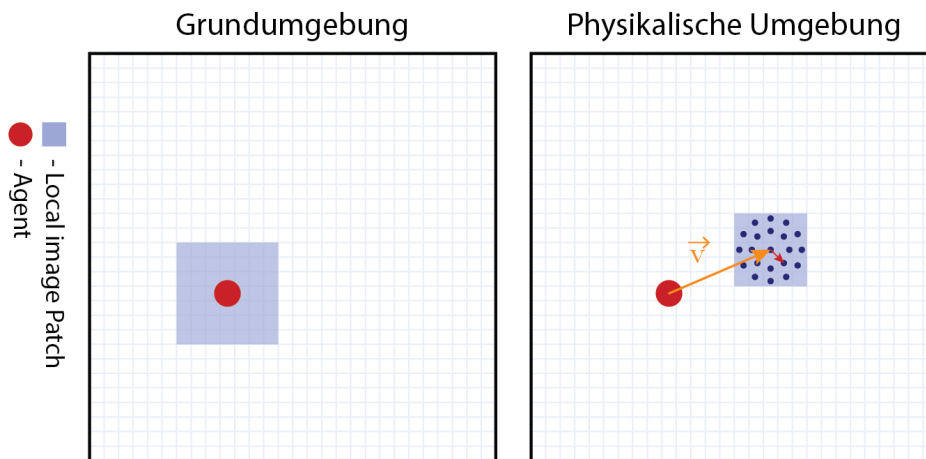


Abbildung 3.6: Angabe der Geschwindigkeit durch eine Verschiebung des Local image Patches. (eigene Abbildung)

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 Actions, die ihn in eine unzulässige Position bewegen würden, nicht auswählen. Wenn allerdings in der physikalischen Umgebung die Geschwindigkeit des Agents zu hoch ist, kann dieser keine Actions mehr wählen, die ihn innerhalb der Grenzen der Zeichenflächen halten würden. Als Lösung wird diesen Fällen die Geschwindigkeit des Agents auf den Nullvektor zurückgesetzt und die Reward-Function löst einen negativen Reward von -0.05 aus.

3.4 Generative Zeichner

Die generativen Zeichner sind eine Gruppe an Variationen der KI, die sich von der Aufgabe des Nachzeichnens entfernen. Trotzdem basieren auch diese Versionen auf dem Grundprogramm (siehe ?? ??) und verändern dieses kaum. Die Aufgabe der generativen Zeichner lautet, Motive ohne eine Vorlage zu zeichnen. So soll die KI nach dem Training mit einem bestimmten Motiv eigene Zeichnungen von diesem Motiv erstellen können. Beispielsweise soll die generative KI (also ein generativer Zeichner) auf das Zeichnen der Zahl Zwei trainiert werden, damit diese nach dem Training ohne eine Vorlage eine Zwei zeichnen kann.

3.4.1 Trainingsstrategie

Die Strategie des Trainings für die generative KI unterscheidet sich von derjenigen der nachzeichnenden KI stark, da die nachzeichnende KI nach dem Training ohne Vorlage nichts zeichnet.

Die Trainingsstrategie der generativen KI funktioniert folgendermassen: Im Input Channel des neuronalen Netzes (siehe ?? ??), wo der nachzeichnenden KI das Bild der Vorlage gegeben wird, hat die generative KI ein vollkommen schwarzes Bild (Alle Werte der Bitmap sind 0). Der Channel für die Vorlage wird nicht entfernt, weil während dem Training der generativen KI zu einem gewissen Anteil trotz dessen Aufgabe eine Vorlage gegeben wird. Die Wahrscheinlichkeit, dass für eine Episode der KI eine Vorlage gezeigt wird, sinkt dabei linear über das Training hinweg von 1 auf 0.25. Zu Beginn des Trainings unterscheidet sich die generative KI somit nicht von der nachzeichnenden KI. Erst nachdem die generative KI das Nachzeichnen weitgehend erlernt hat, erlernt diese ihre eigentliche Aufgabe, indem sie zunehmend seltener eine Vorlage erhält. die generative KI trainiert insgesamt für 6'000 Episoden

Die generative KI verwendet die Basis Reward-Function (siehe ?? ??) unabhängig davon, ob in der aktuellen Episode eine Vorlage gezeigt wird oder nicht. Da die Basis Reward-Function eine Vorlage benötigt, mit der die Zeichnung verglichen werden kann, hat die generative KI während dem Training im Hintergrund immer eine Vorlage. Das bedeutet, dass wenn dem neuronalen Netz der generativen KI keine Vorlage gezeigt wird, so hat die Reward-Function trotzdem eine Vorlage als Vergleich zu der Zeichnung. Prinzipiell ist ein generativer Zeichner während dem Training also ein blinder Nachzeichner. Der relevantere Reward für die generative KI basiert allerdings nicht auf dem Kriterium der prozentualen Übereinstimmung, welches die Basis Reward-Function verwendet, sondern auf dem Kriterium der Erkennbarkeit (siehe ?? ??). Da ein generativer Zeichner ein Motiv ohne weitere Spezifikationen zeichnen soll, ist die Erkennbarkeit des Motives von

grösster Priorität. Die generative KI verwendet deswegen neben der Basis Reward-Function ebenfalls die auf Erkennbarkeit spezialisierte Reward-Function (siehe ?? ??).

Anders als die nachzeichnende KI, startet die generative KI für jede Zeichnung am selben vordefinierten Ort. Das verringert den Trainingsaufwand, aber bedeutet auch, dass nach dem Training die KI nur eine einzige Ausgangslage kennt. Nach dem Training ist die Wahl der Actions deterministisch, Da die KI stets die Action mit dem höchsten Q-Value wählt (siehe ?? ??). Zusammen mit der unveränderlichen Ausgangslage führt der Determinismus dazu, dass die KI nur eine einzige Zeichnung produzieren kann.

Diskussion Gedanken (Generierer nur möglich mit nachzeichner) (Prozess des Nachzeichnens entscheidend für Generierer)

3.4.2 Nicht deterministische Zeichnungen

Damit die generative KI unterschiedliche Bilder zeichnet und nicht deterministisch ist, benötigt der Zeichenprozess ein Element von Zufall. Es gibt verschiedene Ansätze um diesen Zufall zu integrieren.

Der erste Ansatz ist, nach dem Training der generativen KI die Softmax Policy zur Wahl der Actions zu verwenden (siehe ?? ??). Die Softmax Policy macht die Wahl der Action in jedem Step vom Zufall abhängig. Ausserdem hat die Softmax Policy gegenüber von Epsilon-Greedy den Vorteil, dass Actions nie komplett vom Zufall bestimmt werden, sondern nach einer bestimmten Wahrscheinlichkeitsverteilung gewählt werden. Als weiterer Vorteil ist die Wahrscheinlichkeitsverteilung durch die Softmax Temperatur variierbar. Diese Eigenschaft ist für den Erfolg der Softmax Policy als Lösung für das Problem der deterministischen Zeichnungen entscheidend, weil die KI nur mit einer geringen Freiheit in der Wahl der Actions umgehen kann: Eine einzige falsche Action kann bereits dazu führen, dass das Motiv der Zeichnung nicht mehr erkennbar ist. Eine tiefe Softmax Temperatur, in einem Intervall zwischen 0.03 und 0.01, versichert, dass Actions mit einem geringen Q-Value nur in seltenen Fällen gewählt werden.

Trotz geringer Softmax Temperatur ist die Freiheit in der Wahl der Actions zu gross und führt dazu, dass die generative KI häufig keine erkennbaren Bilder produziert. Eine Weiterführung des ersten Ansatz ist deshalb die Beschränkung der Wahl auf die 5 Actions mit der höchsten Wahrscheinlichkeit. Diese Weiterführung hat mehr Erfolg als der erste Ansatz, weil Actions mit einer tiefen Wahrscheinlichkeit gar nicht mehr ausgeführt werden können. Ansonsten würde fast in jeder Zeichnung mindestens eine unwahrscheinliche Action ausgeführt werden.

Die nachzeichnende KI verwendet standardmässig Epsilon-Greedy anstelle der Softmax-Policy, weil die Leistung in allen Kriterien ohne die Softmax Activation deutlich höher ist. Wird allerdings wie im ersten Ansatz nach dem Training die Softmax Policy verwendet, so sollte die KI auch mit der Softmax Policy trainieren. Die Softmax Temperatur ist während dem Training statisch als 0.05 definiert. Eine dynamische Anpassung der Softmax Temperatur während dem Training beeinflusst das Verhalten der KI nicht bemerkbar.

Der zweite Ansatz ist, den Input in das neuronale Netz auf zufällige Weise anzupassen, da ein unterschiedlicher Input zu unterschiedlichen Beurteilungen führt. Eine Möglichkeit, um den Zufall zu integrieren, liegt im Channel des Inputs in das neuronale Netz, wo der KI eine Vorlage gezeigt wird. Der generativen KI wird generell ein in diesem Channel ein schwarzes Bild anstelle einer Vorlage gezeigt. Werden die Werte der Pixel dieses schwarzen Bildes mit zufälligen Werten zwischen 0 und 1 ersetzt, so verändert sich der Input in das neuronale Netz und somit auch der Output. Die KI trifft also Entscheidungen, die von zufälligen Werten abhängen. Damit dieser Ansatz funktioniert, muss bereits während dem Training diese Zufallsquelle integriert werden.

3.5 Auswertung

Die Auswertung beschreibt das sammeln der Leistungsdaten der verschiedenen Variationen und Versionen der KI. Diese Daten machen das Resultat der Methode aus. Die Leistungsdaten beruhen auf den objektiven Werten der definierten Kriterien (siehe ?? ??). Die Auswertung wird für die Variationen der nachzeichnenden KI und die Variationen der generativen KI separat beschrieben, da diese sich in den beiden Fällen unterscheidet.

3.5.1 Auswertung der nachzeichnenden KI

Die Variationen der nachzeichnenden KI werden auf ihre Leistung für drei verschiedene Datensets geprüft. Die drei Datensets enthalten verschiedene Arten von Strichbildern und jeweils eine klassifizierende KI, welche die spezifischen Strichbilder erkennen kann (siehe ??). Im Falle des QuickDraw Datensets wird die KI 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 ??). Die Bilder in den drei Datensets werden im Training nicht verwendet, sind aber gleich verarbeitet wie die Trainingsdaten (siehe ?? ??).

Eine Testumgebung erfasst die Leistung der KI (siehe ?? ??). Zwischen der Trainingsumgebung und der Testumgebung sind zwei relevante Unterschiede. Erstens trainiert die KI in der Testumgebung nicht. Die Testumgebung übernimmt eine trainierte Version der nachzeichnenden KI und verändert

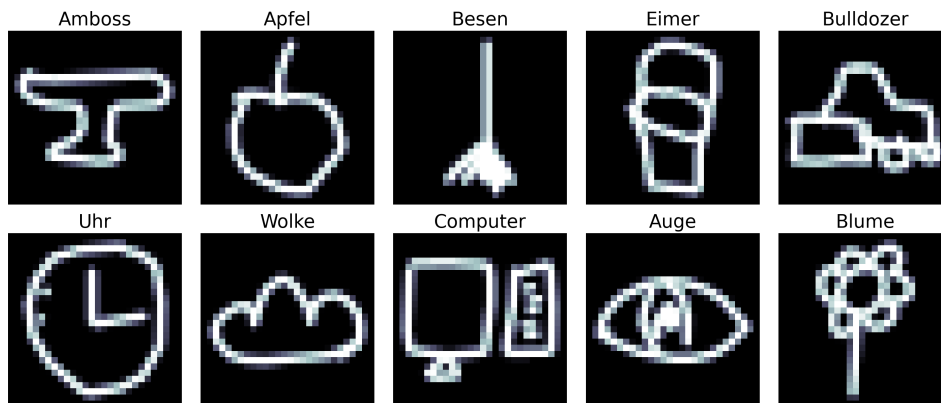


Abbildung 3.7: Beispiele der verwendeten Motive aus dem QuickDraw Datenset. (eigene Abbildung)

diese während dem Test nicht. Zweitens wählt der Agent Die Actions deterministisch. In jedem Step wird die Action mit dem höchsten Q-Value gewählt.

Im Test zeichnet die KI 2'000 Strichbilder nach. Am Ende jeder Zeichnung wird der Zahlenwert der verschiedenen Kriterien (siehe ?? ??) ihrer Definition entsprechend berechnet und gespeichert. Der Durchschnitt aller gespeicherten Werte eines Kriteriums entspricht der Leistung der KI in diesem Kriterium. Da das Kriterium der Erkennbarkeit entweder 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.

3.5.2 Auswertung der generativen Zeichner

Die Variationen der generativen KI werden auf ihre Leistung in dem Kriterium der Erkennbarkeit, der Geschwindigkeit und der zeichnenden Zeit geprüft (siehe ?? ??). Die übrigen Kriterien ergeben ohne eine nachzuzeichnende Vorlage keinen Sinn. Anders als im Training wird der generativen KI in der Testumgebung in keinem Fall eine Vorlage gezeigt. Die Startposition ist auf die selbe vordefinierten Position wie im Training gesetzt. Die Wahl der Actions ist nicht deterministisch, weil die KI in diesem Fall nur eine einzige Zeichnung produzieren könnte. Stattdessen unterliegt die Wahl der Actions den selben Zufallselementen (siehe ?? ??) wie im Training der jeweils getesteten Variation.

Im Test generiert die KI 500 Bilder. Das Motiv der Bilder hängt dabei davon ab, worauf die Variation trainiert ist. Eine Variation wird auf fünf verschiedene Motive trainiert und anschliessend getestet. Die drei Motive sind: eine Null,

3. METHODE

eine Zwei, eine Acht, ein F und eine Blume aus dem QuickDraw Datenset (siehe ??). Wenn eine Variation im Test die Softmax Policy verwendet (siehe ?? ??), dann ist die Softmax Temperatur unabhängig von derjenigen im Training bestimmbar. Die Softmax Temperatur beeinflusst dabei wie viel Varianz in den Zeichnungen ist.

Wie im Test der nachzeichnenden KI (siehe ?? ??), entspricht die Leistung einer Variation dem Durchschnitt der Werte in den verwendeten Kriterien nach jeder Zeichnung.

Kapitel 4

Resultate

Die Resultate tragen die in der Testumgebung erfassten Daten und Werte über die KI zusammen. Für die verschiedenen Variationen und Kriterien werden nachfolgend Abkürzungen eingeführt, die für die Präsentation und die Diskussion der Resultate fortan verwendet werden.

Die Abkürzungen der Kriterien lauten:

- Sim (siehe ?? ??)
- Rec (siehe ?? ??)
- Speed (siehe ?? ??)
- Drawtime (siehe ?? ??)
- Overdrawn (siehe ?? ??)

Die Abkürzungen der Variationen der nachzeichnenden KI lauten:

- Base (siehe ?? ??)
- Rec (siehe ?? ??)
- Speed (siehe ?? ??)
- No-Penlift (+ Speed) (siehe ?? ??)
- No-Overdraw (siehe ?? ??)
- Physics (siehe ?? ??)

Die Abkürzungen der Variationen der generativen KI lauten:

- Deterministic
- Softmax

4.1 Tabellen

Der erste Teil der Resultate besteht aus Tabellen, welche die Daten über die Leistung der KI aus der Testumgebung (siehe ?? ??) zusammentragen. Die Tabellen sind dabei in diejenigen der nachzeichnenden KI und diejenigen der generativen KI unterteilt.

4.1.1 Tabellen der nachzeichnenden KI

Die Tabellen für die nachzeichnende KI beschreiben die Leistung jeder Variation in jedem Kriterium. Es gibt insgesamt drei Tabellen für die nachzeichnende KI. Der Unterschied in den Tabellen liegt im Datenset, mit denen die KI jeweils getestet wurde.

Tabelle 4.1: Testen auf MNIST Datenset — 2000 Tests

	Übereinstimmung %	Erkennbarkeit %	Geschwindigkeit
Grund-Basis	86.5	86.6	24.5
Grund-MNIST	66.8	64.3	51.2
Grund-Speed	85.7	82.3	23.3
Grund-MNIST-Speed	61.4	55.1	56.8
Physik-Basis	56.4	46.4	62.5
Physik-MNIST	38.4	35.7	63.9
Physik-Speed	63.0	58.2	61.2
Physik-MNIST-Speed	29.2	27.3	63.7

Tabelle 4.2: Testen auf EMNIST Letters Datenset — 2000 Tests

	Übereinstimmung %	Erkennbarkeit %	Geschwindigkeit
Grund-Basis	86.8	74.5	38.2
Grund-MNIST	65.2	45.0	57.4
Grund-Speed	88.1	73.5	36.1
Grund-MNIST-Speed	62.2	40.0	60.9
Physik-Basis	57.6	32.4	63.5
Physik-MNIST	43.3	23.6	63.9
Physik-Speed	56.3	35.0	63.6
Physik-MNIST-Speed	30.2	13.9	64.0

Tabelle 4.3: Testen auf QuickDraw-Datenset — 2000 Tests

	Übereinstimmung %	Erkennbarkeit %	Geschwindigkeit
Grund-Basis	79.1	80.5	39.1
Grund-MNIST	57.3	62.5	59.9
Grund-Speed	80.0	80.3	35.0
Grund-MNIST-Speed	54.9	58.9	62.5
Physik-Basis	48.1	55.7	63.8
Physik-MNIST	30.5	38.9	64.0
Physik-Speed	50.0	58.3	63.6
Physik-MNIST-Speed	22.4	31.1	64.0

4.1.2 Tabellen der generativen KI

Die Tabellen der generativen KI beschreiben die Leistung der zugehörigen Variationen auf die ausgewählten Kriterien. Es gibt insgesamt fünf Tabellen. Jede Tabelle beschreibt die Leistung im Zeichnen eines des ausgewählten Motive.

Tabelle 4.4: Testen auf QuickDraw-Datenset — 2000 Tests

	Übereinstimmung %	Erkennbarkeit %	Geschwindigkeit
Grund-Basis	79.1	80.5	39.1
Grund-MNIST	57.3	62.5	59.9
Grund-Speed	80.0	80.3	35.0
Grund-MNIST-Speed	54.9	58.9	62.5
Physik-Basis	48.1	55.7	63.8
Physik-MNIST	30.5	38.9	64.0
Physik-Speed	50.0	58.3	63.6
Physik-MNIST-Speed	22.4	31.1	64.0

4.2 Bildersammlung

Eine Sammlung von gezeichneten Strichbildern ergänzt die Resultate. Die Zeichnungen, die in der Sammlung vertreten sind, stammen aus einer zufälligen Auswahl aus dem Test der jeweiligen Variation der KI. Die Bilder haben einen Farbverlauf, der den zeitliche Verlauf des Zeichnens darstellt. Die Helligkeit eines Striches ist proportional zu dem Step, in dem dieser gezeichnet wurde. Das bedeutet, dass dunklere Striche früher gezeichnet wurden als hellere Striche. Bewegungen des Agents, in denen dieser nicht zeichnete, sind in den Bildern nicht erkennbar.

4.2.1 Bildersammlung der nachzeichnenden KI

Die Strichbilder für die Bildersammlung der nachzeichnenden KI sind jeweils in Paaren angeordnet. Das linke Bild im Paar zeigt die Vorlage aus dem Datenset und das rechte Bild zeigt die nachgezeichnete Variante von der KI. Jede Spalte der Bildersammlung zeigt Bilder aus einem anderen Datenset

4. RESULTATE



Abbildung 4.1: Grund-Basis Bildersammlung

4. RESULTATE

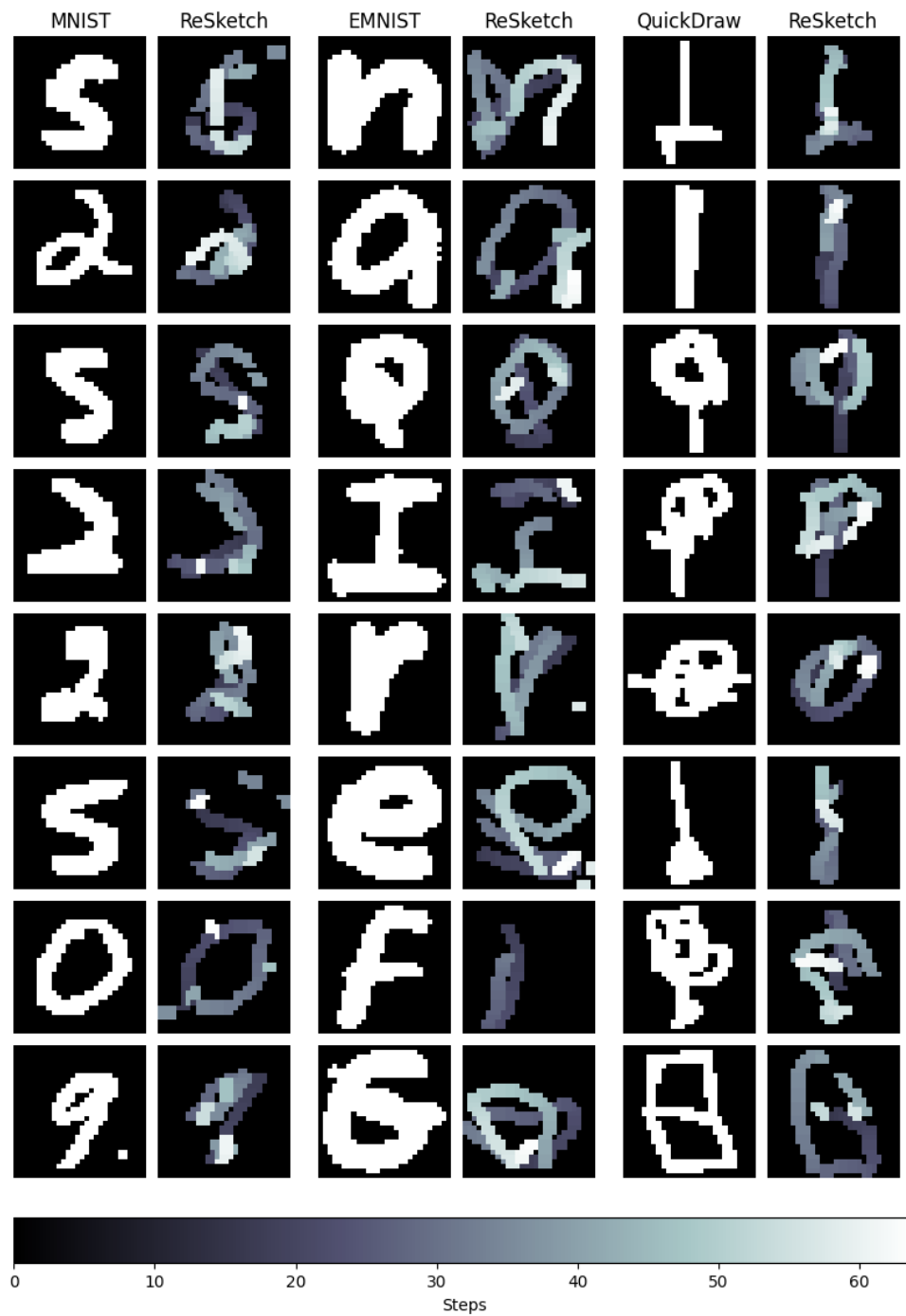


Abbildung 4.3: Physik-Basis Bildersammlung

(siehe ?? ??).



Abbildung 4.4: Physik-Speed Bildersammlung

Kapitel 5

Diskussion

Die Diskussion analysiert die Resultate der Methode, um daraus eine Antwort auf die Fragestellung abzuleiten. Zu diesem Zweck werden die einzelnen Unterfragen dieser Arbeit behandelt. Darauf folgt ein Fazit und ein Ausblick zusammen mit möglichen Anwendungsbereichen für die verschiedenen Versionen der künstlichen Intelligenz

5.1 Diskussion der Fragestellung

Die Fragestellung dieser Untersuchung lautet: Inwiefern kann eine künstliche Intelligenz lernen, Strichbilder auf eine physische Weise nachzuzeichnen? Die Unterfragen erfassen die verschiedenen Aspekte dieser Fragestellung und erweitern diese sogar. Die Diskussion der Unterfragen macht somit die Antwort auf die Fragestellung aus.

5.1.1 Wie kann die Architektur einer KI aussehen, die das Nachzeichnen erlernt?

Das Deep Q-Learning Modell hinter der nachzeichnenden KI erreicht hohe Werte in allen Kriterien und lässt sich, wie die verschiedenen Variationen zeigen, auf der Grundlage von unterschiedlichen Kriterien trainieren (siehe ?? ??). Die Architektur einer nachzeichnenden KI kann somit so aussehen, wie sie in dieser Arbeit beschrieben ist (siehe ?? ??).

5.1.2 Wie lässt sich die Leistung der KI in ihrer Aufgabe beurteilen?

Die Leistung der KI lässt sich durch fünf vordefinierte Kriterien beurteilen (siehe ?? ??). Die Werte der Kriterien sind quantifizierbar und automatisiert berechenbar, was sie für die Anwendung in einer künstlichen Intelligenz geeignet macht. Allerdings wären auch andere Kriterien denkbar, die das

Nachzeichnen definieren. Dazu gehören auch Subjektive Definitionen, nach denen die KI das Nachzeichnen nicht unbedingt erlernen kann.

Unter den Kriterien ist die Erkennbarkeit ein Sonderfall, weil die Bestimmung dieser auf eine klassifizierende KI angewiesen ist. Die klassifizierende KI erreicht keine Genauigkeit von 100% und schätzt gewisse Zeichnungen aus der Sicht der meisten Menschen falsch ein. Da die klassifizierende KI nicht perfekt funktioniert, ist auch das Training der KI, in den Variationen wo diese verwendet wird, nicht optimal. Aus diesem Grund sind auch die Resultate im Kriterium der Erkennbarkeit minimal verfälscht. Die Verfälschung ist allerdings sehr gering, da die klassifizierende KI je nach Typ eine Genauigkeit von bis zu 99% erreicht

Die Kriterien der Geschwindigkeit, der zeichnenden Zeit und der Übermalung werden verwendet, weil deren Kombination ein Mass für den 'Schwung' in den Bewegungen der KI gibt. 'Schwung' heisst aus einer subjektiveren Ansicht, dass die Zeichnungen schnell und möglichst in einem Ansatz fertiggestellt werden.

5.1.3 Wie lässt sich die Leistung der KI in ihrer Aufgabe verbessern?

Erreicht eine gute Leistung in vielen Kriterien, indem die sich lediglich auf das Kriterium der Übereinstimmung fokussiert. Einführung der Stopp Action verbessert allerdings die Geschwindigkeit überall, wo diese verwendet wird (Rec, Speed, No-Liftpen) Rec Version verbessert die Leistung nicht, weil die Leistung der Grundversion bereits gut genug ist No-Liftpen Variation verbessert den Wert in Kriterium der zeichnenden Zeit auch nur marginal im Vergleich zu Speed Version, weil der Wert bereits sehr hoch ist. No-Overdraw + Speed erreicht im Vergleich zu der Speed Version leicht bessere Werte No-Overdraw ohne Speed erreicht im Vergleich zu der Base Version viel bessere Werte. Die Physics Version verschlechtert die Leistung in allen Kriterien. Die KI kann mit den physikalischen Simulationen nicht gut umgehen • Wie ändert sich die Leistung der KI für Strichbilder, die im Training nicht enthalten sind

Bezogen auf die definierten Kriterien erreicht die Grund-Basis Version Werte, die durch die implementierten Variationen nicht verbessert werden. Die Grund-Speed Version ist hierbei eine Ausnahme (siehe ?? ??). Insgesamt sind die meisten Variationen der KI ein gescheiterter Versuch der Verbesserung der Leistung. Die Grund-Basis Version erfuhr allerdings während dessen Entwicklung signifikante Verbesserungen. Die grössten Verbesserungen stammen aus der Optimierung der Hyperparameter durch den Bayesian Optimization Algorithmus (siehe ?? ??). Diese Optimierung steigerte die Leistung der Grund-Basis Version im Kriterium der prozentualen Übereinstimmung um 40 – 50%.

5.1.4 Wie ändert sich die Leistung der KI für Strichbilder, die im Training nicht enthalten sind?

In allen acht Versionen bleibt die Leistung der KI zwischen den drei Datensets (siehe ??) vergleichbar. Die Tabellen (siehe ?? und ??) zeigen die Leistung der Grund-Basis Version und der Physik-Basis Version in den drei definierten Kriterien, getestet auf die drei Datensets. Der Wert der Übereinstimmung zwischen dem MNIST Datenset und dem EMNIST Datenset ist beinahe identisch. Für beide Versionen ist der Wert der Übereinstimmung für das QuickDraw Datenset niedriger. Insgesamt ist die KI in diesem Kriterium jedoch kaum beeinflusst durch die Wahl des Datensets. Die Analyse der anderen zwei Kriterien führt zu einer ähnlichen Schlussfolgerung. Interessant ist, dass vor allem die Grund-Basis Version eine viel höhere Geschwindigkeit im Zeichnen von MNIST Zahlen hat, als im Zeichnen von EMNIST Buchstaben. Obwohl die Formen zu einem grossen Teil ähnlich sind, scheint die KI durch das spezifische Training auf MNIST Ziffern eine höhere Geschwindigkeit zu entwickeln.

Tabelle 5.1: Grund-Basis Leistung

	Übereinstimmung %	Erkennbarkeit %	Geschwindigkeit
MNIST	86.5	86.6	24.5
EMNIST	86.8	74.5	38.2
QuickDraw	79.1	80.5	39.1

Tabelle 5.2: Physik-Basis Leistung

	Übereinstimmung %	Erkennbarkeit %	Geschwindigkeit
MNIST	56.4	46.4	62.5
EMNIST	57.6	32.4	63.5
QuickDraw	48.1	55.7	63.8

5.1.5 Wie und inwiefern lässt sich das Verhalten der KI mit menschlichem Zeichnen vergleichen?

Die Antwort auf diese Frage leitet sich nicht aus den objektiven Resultaten ab, sondern basiert auf subjektiven Beobachtungen. Die Bewegungen in der Physik-Version der künstlichen Intelligenz basieren grundsätzlich auf denselben Gesetzen wie die Bewegungen in der echten Welt. Allerdings sind die Bewegungen stark vereinfacht im Vergleich zu menschlichen Bewegungen. Ausserdem ist für die künstliche Intelligenz der Druck des Stiftes nicht veränderbar. Zumindest konzeptuell nähert die künstliche Intelligenz menschliches Zeichnen, bezogen auf die physischen

Einschränkungen, an. Einige menschliche Gewohnheiten sind bei der künstlichen Intelligenz allerdings nicht beobachtbar. Zum Beispiel beginnt die künstliche Intelligenz beim Zeichnen von Ziffern an zufälligen Orten, während Menschen in der Regel für jede Ziffer an derselben Stelle ansetzen.

5.1.6 Kann eine KI Strichbilder ohne Vorlage nachzeichnen?

Beantwortung der Fragestellung

- Wie lässt sich die Leistung der KI in ihrer Aufgabe beurteilen? Durch die definierten Kriterien. Sind alle Objektiv -; Gut. Es wären Allerdings auch andere Kriterien denkbar Beurteilung durch Erkennbarkeit. Ist Sonderfall, weil Verwendung von Classifier. Classifier nicht 100 Kriterium der Geschwindigkeit, der zeichnenden Zeit und der Übermalung sind spezifisch dafür, dass die Ki das Motiv möglichst in einem Schwung zeichnet. • Wie lässt sich die Leistung der KI in ihrer Aufgabe verbessern.

Keine grossen Änderungen -; Spezifischere Analyse mit spezifischeren Werten • Wie und inwiefern lässt sich das Verhalten der KI mit menschlichem Zeichnen vergleichen KI bewegt sich mit beschränkter Geschwindigkeit und kann nur am Ort zeichnen, wo sie sich gerade befindet. Diese Simulation eines Stiftes wäre in die physische Welt übersetzbar. Das heisst, in zeichnender Roboter wäre durch die KI bedienbar. Physik Version stellt ein Versuch dar, Verhalten der KI noch näher an menschliches Zeichnen zu bringen, indem die Umgebung der physischen Welt angenähert ist. Dieser Versuch ist gescheitert, weil die KI dadurch allgemein schlechter zeichnet und menschliches Zeichnen deswegen nicht repräsentieren kann Training auf die einzelnen Kriterien der KI nähert allerdings menschliches Zeichnen an. Das Kriterium der Geschwindigkeit, der zeichnenden Zeit und der Übermalung ergeben zusammen ein Mass für den SSchwung einer Zeichnung. Indem die Leistung in diesen Kriterien maximiert wird, steigt der Schwung mit dem die KI zeichnet. Schwung ist ein wichtiges Element für den menschlichen Prozess des Zeichnens. Indem die KI also auf ihren Schwung trainiert wird, nähert sie das menschliche Zeichnen durchaus an. • Kann die KI Strichbilder ohne Vorlage zeichnen Ja. Eine abgewandelte Form der nachzeichnenden KI kann Strichbilder mit einem ausgewählten Motiv zeichnen, solange im Training auf dieses Motiv trainiert wird. Da die KI ohne Vorlage eigene Zeichnungen produzieren kann, entwickelt sie gewissermassen eine eigene Handschrift. Diese Tatsache stellt nicht nur einen Fortschritt in der Nachahmung von menschlichem Verhalten durch Computer dar, sondern gibt auch Einblicke, wie Menschen ihre Handschrift entwickeln. Es stellt sich heraus, dass die KI dieser Arbeit nur lernen kann, eigene Bilder zu zeichnen, wenn diese zuvor Beispiele vom gewünschten Motiv gesehen hat und das Nachzeichnen von diesen Motiven erlernt hat.

Fazit und Ausblick • Die Resultate deuten darauf hin, dass die nachzeichnende KI tatsächlich das Nachzeichnen von beliebigen Strichbildern erlernt hat. • Diese Antwort unterliegt allerdings einigen Annahmen und Voraussetzungen, die weiter diskutiert werden können. • Beispielsweise ist das Format der Strichbilder vorausgesetzt. Die KI kann in einem bestimmten Format beliebige Bilder nachzeichnen, aber in einem anderen Format kann sie nicht zeichnen. • Spezifisch kann die KI kleine Bilder mit einer Auflösung von 28x28 Pixeln in schwarzweiss und einer festen Strichbreite zeichnen. • Diese Voraussetzungen schränken die nachzeichnende KI ein. In der Auflösung von diesen Einschränkungen liegt Entwicklungspotenzial. • Eine weitere Voraussetzung liegt in der Definition des Nachzeichnens. Die KI erlernt das nachzeichnen nach einer selbst bestimmten Definition. Die Kriterien dieser Definition sind für den Trainingsprozess einer künstlichen Intelligenz sinnvoll gewählt, aber es wären auch andere Kriterien denkbar. • Die Qualität der nachzeichnenden KI und der generativen KI hängt von der Verlässlichkeit der klassifizierenden KI ab, die im Training verwendet wird. Da diese nicht eine Genauigkeit von 100% • Ausserdem sind die Hyperparameter von einigen Variationen der KI vermutlich nicht optimal gewählt, da ein angemessener Optimierungsprozess zu viel Rechenaufwand bedeuten würde. Die Leistung der KI liegt somit vermutlich leicht unter ihrem vollen Potenzial

Anwendungsbereiche • Nachzeichnende KI In der Robotik. Für zeichnende Roboter. Sollte eine weiterentwickelte Version der nachzeichnenden KI vielfältigere Bilder produzieren können, So könnte ein Roboter durch die KI von beliebigen Bildern Zeichnungen Anlegen können. Weiterer Anwendungsbereich wäre die Vektorisierung. Damit ist die Umwandlung von Rastergrafiken in Vektorgrafiken gemeint. Diese Konvertierung benötigt einen Nachzeichnenprozess. Die nachzeichnende KI kann für diese Aufgabe verwendet werden. Um in der Praxis Anwendung zu finden, müsste die KI allerdings was den Rechenaufwand angeht effizienter werden. • Generative KI Eine Anwendung der Generativen KI wäre die Nachahmung von Handschriften. Wenn der KI im Training zufällige Beispiele von einem Motiv gezeigt werden, entwickelt die KI einen eigenen Weg um dieses Motiv zu zeichnen. Wenn der KI allerdings Strichbilder von einer ausgewählten Person gegeben werden, würde die KI die Handschrift dieser Person annähern. Inwiefern die Zeichnungen der KI tatsächlich der Handschrift einer Person entsprechen, ist allerdings nur subjektiv bestimmbar

5.2 Fragestellung und Unterfragen

Die Fragestellung und die Unterfragen decken nicht alle Erkenntnisse aus den Resultaten ab (siehe ?? ??). Einige allgemeine Feststellungen geben Einblick, wie die Resultate zu verstehen sind.

Die Grund-Basis Version und die Grund-Speed Version (siehe ?? ??) erreichen in allen Kriterien für alle Datensets die beste Leistung. Die Resultate zwischen den Versionen sind dabei bis auf den Wert des Kriteriums der Geschwindigkeit (siehe ?? ??) fast ununterscheidbar. In diesem Kriterium erreicht die Grund-Speed Variation leicht bessere Werte als die Grund-Basis Version. Beispielsweise stellt die Grund-Speed Version Zeichnungen aus dem Quickdraw Datenset durchschnittlich vier Schritte früher als die Grund-Basis Version fertig. Insgesamt erreicht somit die Grund-Speed Version die beste Leistung. Unter den Versionen, die auf der physikalischen Umgebung basieren, erreichen ebenfalls die Physik-Basis und die Physik-Speed Versionen die beste Leistung. Die Werte sind allerdings deutlich tiefer als bei der Grund-Basis und der Grund-Speed Version.

Die Grund-MNIST Version und die Physik-MNIST Version sind in allen Kriterien schlechter als die Basis und Speed Versionen. Auch im Kriterium der Erkennbarkeit (siehe ?? ??) von MNIST Ziffern bringt die MNIST Variation keinen Vorteil. Die Physik-MNIST-Speed Version erbringt insgesamt die schlechteste Leistung. Eine Erklärung dafür ist, dass diese Version eine Kombination von allen Variationen (siehe ?? ??) ist und somit von der leistungsstarken Grund-Basis Version am stärksten abweicht.

5.2.1 Beantwortung der Unterfragen

Insgesamt sechs Unterfragen werden beantwortet (siehe ?? ??). Diese Unterfragen weiten die Fragestellung aus und tragen zu der schlussendlichen Antwort auf die Fragestellung bei. Die Antworten beruhen auf den Resultaten, aber auch auf Erkenntnissen aus der Methode selbst (siehe ?? ??).

Wie kann die Architektur einer KI aussehen, die das Nachzeichnen erlernt?

Unter der Annahme, dass die KI dieser Arbeit das Nachzeichnen erlernt, (siehe ?? ??), kann die Architektur genau so aussehen, wie sie in dieser Arbeit beschrieben ist (siehe ?? ??).

Wie lässt sich die Leistung der KI in ihrer Aufgabe beurteilen?

Die Leistung der KI lässt sich durch die definierten Kriterien (siehe ?? ??) beurteilen. Das Kriterium der Übereinstimmung ist durch einen objektiven und absoluten Wert repräsentiert, wodurch es das aussagekräftigste Kriterium ist. Ausserdem ist der maximale Wert des Kriteriums (100%) unabhängig vom gezeichneten Bild und dem Verhalten der KI. Dadurch ist das Kriterium geeignet für Vergleiche zwischen Versionen der KI.

Die Kriterien der Erkennbarkeit und der Geschwindigkeit sind an subjektive Annahmen gebunden. Zum Beispiel wird für das Kriterium der

Geschwindigkeit ein subjektiver Punkt der Fertigstellung definiert (siehe ?? ??). Dadurch sinkt ihre Aussagekraft. Allerdings verändern sich die Annahmen nicht und die Kriterien sind in jedem Fall durch einen Zahlenwert repräsentiert. Somit eignen sich auch diese Kriterien für Vergleiche zwischen Versionen der KI. Aus der Annahme heraus, dass für Menschen beim Nachzeichnen Erkennbarkeit wichtiger als absolute Genauigkeit ist, ergibt sich das Kriterium der Erkennbarkeit als besonders wichtig. Aus diesem Grund ist das Kriterium in der Fragestellung (siehe ?? ??) festgehalten.

Wie lässt sich die Leistung der KI in ihrer Aufgabe verbessern?

Bezogen auf die definierten Kriterien erreicht die Grund-Basis Version Werte, die durch die implementierten Variationen nicht verbessert werden. Die Grund-Speed Version ist hierbei eine Ausnahme (siehe ?? ??). Insgesamt sind die meisten Variationen der KI ein gescheiterter Versuch der Verbesserung der Leistung. Die Grund-Basis Version erfuhr allerdings während dessen Entwicklung signifikante Verbesserungen. Die grössten Verbesserungen stammen aus der Optimierung der Hyperparameter durch den Bayesian Optimization Algorithmus (siehe ?? ??). Diese Optimierung steigerte die Leistung der Grund-Basis Version im Kriterium der prozentualen Übereinstimmung um 40 – 50%.

Welche Einflüsse haben physische Einschränkungen auf die Leistung der KI?

Die physischen Einschränkungen, die auf Physiksimulationen basieren (siehe ?? ??), verschlechtern die Leistung der KI bezogen auf die definierten Kriterien. Alle Versionen, die auf der Grundumgebung basieren, erzielen höhere Werte als die gleichen Versionen basierend auf der physikalischen Umgebung. Die physikalische Umgebung hat zum Ziel, die Bewegungen der KI realistischer zu gestalten als die Grundumgebung. In diesem Bereich kann der Einfluss nicht objektiv bestimmt werden. Aus Beobachtungen der Bilder, welche in der physikalischen Umgebung gezeichnet sind (siehe ?? ??), gehen ebenfalls keine Erkenntnisse in diesem Bereich hervor. Die Bilder unterscheiden sich nicht bedeutend von denjenigen aus der Grundumgebung.

Wie ändert sich die Leistung der KI für Strichbilder, die im Training nicht enthalten sind?

In allen acht Versionen bleibt die Leistung der KI zwischen den drei Datensets (siehe ??) vergleichbar. Die Tabellen (siehe ?? und ??) zeigen die Leistung der Grund-Basis Version und der Physik-Basis Version in den drei definierten Kriterien, getestet auf die drei Datensets. Der Wert der

Übereinstimmung zwischen dem MNIST Datenset und dem EMNIST Datenset ist beinahe identisch. Für beide Versionen ist der Wert der Übereinstimmung für das QuickDraw Datenset niedriger. Insgesamt ist die KI in diesem Kriterium jedoch kaum beeinflusst durch die Wahl des Datensets. Die Analyse der anderen zwei Kriterien führt zu einer ähnlichen Schlussfolgerung. Interessant ist, dass vor allem die Grund-Basis Version eine viel höhere Geschwindigkeit im Zeichnen von MNIST Zahlen hat, als im Zeichnen von EMNIST Buchstaben. Obwohl die Formen zu einem grossen Teil ähnlich sind, scheint die KI durch das spezifische Training auf MNIST Ziffern eine höhere Geschwindigkeit zu entwickeln.

Tabelle 5.3: Grund-Basis Leistung

	Übereinstimmung %	Erkennbarkeit %	Geschwindigkeit
MNIST	86.5	86.6	24.5
EMNIST	86.8	74.5	38.2
QuickDraw	79.1	80.5	39.1

Tabelle 5.4: Physik-Basis Leistung

	Übereinstimmung %	Erkennbarkeit %	Geschwindigkeit
MNIST	56.4	46.4	62.5
EMNIST	57.6	32.4	63.5
QuickDraw	48.1	55.7	63.8

Wie und inwiefern lässt sich das Verhalten der KI mit menschlichem Zeichnen vergleichen?

Die Antwort auf diese Frage leitet sich nicht aus den objektiven Resultaten ab, sondern basiert auf subjektiven Beobachtungen. Die Bewegungen in der Physik-Version der künstlichen Intelligenz basieren grundsätzlich auf denselben Gesetzen wie die Bewegungen in der echten Welt. Allerdings sind die Bewegungen stark vereinfacht im Vergleich zu menschlichen Bewegungen. Ausserdem ist für die künstliche Intelligenz der Druck des Stiftes nicht veränderbar. Zumindest konzeptuell nähert die künstliche Intelligenz menschliches Zeichnen, bezogen auf die physischen Einschränkungen, an. Einige menschliche Gewohnheiten sind bei der künstlichen Intelligenz allerdings nicht beobachtbar. Zum Beispiel beginnt die künstliche Intelligenz beim Zeichnen von Ziffern an zufälligen Orten, während Menschen in der Regel für jede Ziffer an derselben Stelle ansetzen.

5.2.2 Beantwortung der Fragestellung

Die Fragestellung lautet: Inwiefern kann eine künstliche Intelligenz lernen, Strichbilder auf eine physische Weise nachzuzeichnen, sodass diese durch ein automatisches System erkannt werden? (siehe ?? ??) Diese Frage hat mehrere Aspekte, die teilweise bereits durch die Unterfragen (siehe ?? ??) erfasst werden. Für die schlussendliche Antwort folgt eine genauere Ausführung der Aspekte.

Die KI zeichnet durch Physiksimulationen und durch allgemeine Einschränkungen der Bewegungsfreiheit auf eine annähernd physische Weise. Das Zeichnen ist nur annähernd physisch, da alle Bewegungen simuliert sind und somit in keiner physischen Umgebung stattfinden. Ausserdem sind die Simulationen nicht vollkommen realistisch (siehe ?? ??).

Die künstliche Intelligenz erlernt das Nachzeichnen bezogen auf die Kriterien, nach denen es definiert ist, erfolgreich. Dafür sprechen die Werte der besten Versionen für das Nachzeichnen von Ziffern (siehe ?? ??). Die hohen Werte im Kriterium der Erkennbarkeit der besten Versionen bestätigen ausserdem, dass die Zeichnungen der KI in den meisten Fällen von einem automatischen System (einer zweiten KI) erkannt werden.

Laut der Fragestellung soll die KI das Nachzeichnen von Strichbildern erlernen. Damit ist implizit das Nachzeichnen von allen möglichen Arten von Strichbildern gemeint. Die Leistung der KI kann nicht auf alle möglichen Strichbilder überprüft werden, aber der Test mit drei verschiedenen Datensets ergibt vielversprechende Resultate (siehe ?? ??). Die KI erlernt das Nachzeichnen von Ziffern, Kleinbuchstaben und zehn zufälligen Motiven aus dem QuickDraw Datenset erfolgreich. Durch die Vielfalt im QuickDraw Datenset kann die Annahme getroffen werden, dass die KI zumindest einen grossen Teil der denkbaren Strichbilder erfolgreich nachzeichnen kann.

Die zusammenfassende Antwort auf die Frage lautet somit: Eine künstliche Intelligenz kann das Nachzeichnen von Strichbildern auf annähernd physische Weise in dem Sinne lernen, dass die fertige Zeichnung von einem automatischen System grösstenteils erkannt wird, die Übereinstimmung zwischen der Vorlage und der Zeichnung gross ist und die Zeichnung nicht viel Zeit in Anspruch nimmt.

Diese Antwort bezieht sich auf die genaue Frage, wie sie in der Einleitung steht. Der nächste Abschnitt beurteilt die Frage durch die Erkenntnisse aus dieser Arbeit und geht auf mögliche Erweiterungen ein.

5.3 Fazit und Ausblick

Die Resultate erlauben eine positive Antwort auf die Fragestellung (siehe ?? ??). Diese Antwort setzt allerdings einige Annahmen voraus, die weiter

diskutiert werden können. Die grösste Annahme bezieht sich auf die Definition des Nachzeichnens. Diese Arbeit definiert Nachzeichnen durch drei Kriterien und durch physische Rahmenbedingungen. Die Kriterien sind für eine künstliche Intelligenz sinnvoll gewählt (siehe ?? ??), allerdings wären auch andere Kriterien möglich. Nachzeichnen ist eine menschliche Tätigkeit. Dieser menschliche Aspekt ist in den definierten Kriterien wenig enthalten. Die zweite Annahme bezieht sich auf die Definition eines Strichbildes. Die KI kann nur eine bestimmte Art von Strichbildern nachzeichnen. Mehrfarbige Bilder kann die KI beispielsweise nicht zeichnen.

Die physischen Rahmenbedingungen unterscheiden sich von denjenigen, die ein Mensch erfährt. Das kommt daher, dass die physischen Rahmenbedingungen für die KI lediglich simuliert sind. Das verunmöglicht eine umfassende Antwort auf die Frage, ob die künstliche Intelligenz auf eine physische Weise zeichnet. Dieses Problem könnte mit einem Roboter gelöst werden, der die künstliche Intelligenz in eine reale, physische Umgebung überführt. Der Roboter könnte verschiedenste Strichbilder auf einem echten Stück Papier, und somit zwangsläufig auf physische Weise nachzeichnen.

Aktuell sind die Bewegungen der künstlichen Intelligenz in gewissen Belangen eingeschränkt. So ist beispielsweise die Druckstärke nicht variierbar. Ausserdem zeichnet die künstliche Intelligenz vorwiegend kleine Strichbilder. Experimente mit grösseren Konstrukten, wie ganze Wörter, wären eine mögliche Erweiterung.

Alles in allem sind eine Vielzahl an denkbaren Fragen und Ideen möglich, die auf ReSketch, der künstlichen Intelligenz hinter dieser Arbeit, basieren.

5.4 Selbstreflexion

Die Selbstreflexion gibt genauere Einblicke in die Vorgehensweise hinter dieser Arbeit. Diese Dokumentation ist grundsätzlich eine Zusammenfassung der wichtigsten Ereignisse. Viele Aspekte bleiben verschwiegen. Die Selbstreflexion geht näher auf drei wichtige Aspekte ein, die in der zusammengefassten Dokumentation nicht genug betont sind.

Die Dokumentation ist mit LaTeX und spezifischer der ETH Thesis Formatvorlage (**noauthor'cadmo'2014**) formatiert. Ein Grossteil der Abbildungen stammt von den Autoren und ist in Adobe Illustrator oder der Python Library Matplotlib erstellt.

5.4.1 Optimierung der KI

Insgesamt sind acht Versionen der KI präsentiert. Im Verlaufe des Projektes gab es viele weitere Versuche, die Leistung der künstlichen Intelligenz zu

verbessern. Diese Versuche führten allerdings häufig dazu, dass die KI den akkumulierten Reward (siehe ?? ??) nicht mehr maximieren konnte. In der Dokumentation sind deswegen nur diejenigen Versuche festgehalten, die tatsächlich funktionieren. Häufig ist der Grund hinter dem Scheitern oder dem Erfolg einer KI schwer zu erkennen, was die Optimierung allgemein verkompliziert.

Die Strategie hinter der Optimierung der KI besteht in den meisten Fällen aus wiederholtem ausprobieren mit Anpassungen zwischen jedem Versuch. Hilfsmittel, wie der Bayesian Optimization Algorithmus, erleichtern diese Aufgabe massgeblich. Diese Strategie der Optimierung ist für einen Computer sehr ressourcenintensiv. In den längsten Optimierungsphasen liefen die beiden Computer, auf denen die Arbeit verrichtet wurde, zusammen länger als 48 Stunden.

5.4.2 Analyse der KI

Eine Analyse der künstlichen Intelligenz ist notwendig, um die Fragestellung und die Unterfragen zu beantworten. Aber auch während der Entwicklung ist eine stetige Analyse nötig, um die KI zu verstehen und zu verbessern.

Die Analyse besteht hauptsächlich darin, die Leistung der künstlichen Intelligenz zu beurteilen. Das geschieht mittels der Kriterien, die für diesen Zweck definiert sind (siehe ?? ??). Die Kriterien sind dabei so definiert, dass sie für jede mögliche Variation identisch bleiben. Der durchschnittliche akkumulierte Reward ist beispielsweise absichtlich kein Kriterium. Der akkumulierte Reward ist abhängig von der Reward-Function (siehe ?? ??) und unterscheidet sich somit zwischen Variationen.

Eine weitere Form der Analyse stammt aus der Sammlung von Daten über das Lernverhalten der KI. So wird aus jedem Training ein Graph erstellt, der die durchschnittliche Leistung der KI in jeder Episode erfasst (siehe ??). Die Leistung ist dabei durch den akkumulierten Reward in jeder Episode repräsentiert. Wie erwähnt, können Versionen der KI nicht anhand ihres akkumulierten Rewards verglichen werden. Der akkumulierte Reward zeigt allerdings für einzelne Versionen am präzisesten, inwiefern diese den akkumulierten Reward maximieren können.

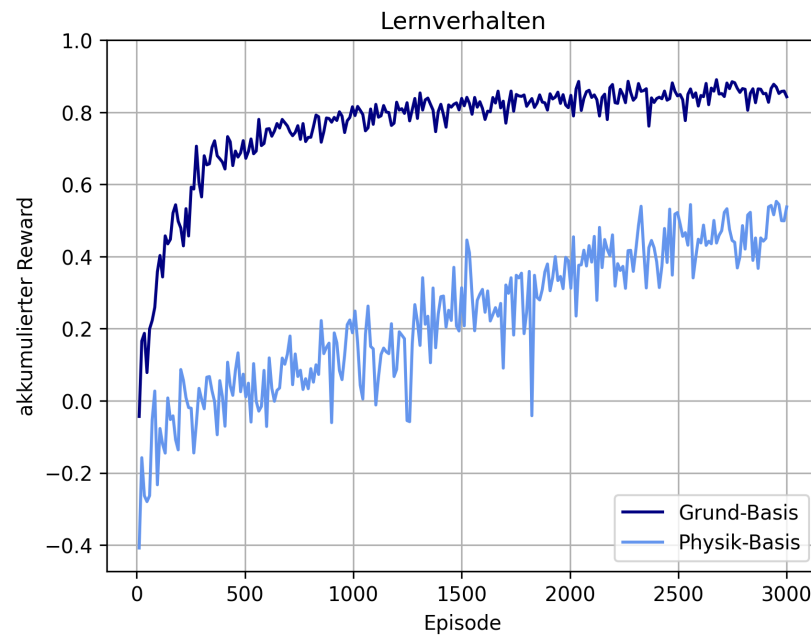


Abbildung 5.1: Akkumulierter Reward zu jeder Episode (Lernverhalten) der Grund-Basis Version und der Physik-Basis Version. (eigene Abbildung)

5.4.3 Verwendung von Git und GitHub

Die Verwendung von Git und Github (siehe ?? ??) erleichtert die Arbeit an einem Projekt von dieser Grösse massgeblich. Die Programme ermöglichen einfache Zusammenarbeit am Programmcode und an der Dokumentation. GitHub dient dabei zusätzlich als Hilfsmittel zur Organisation durch die integrierte Funktion der Project Boards. Diese Funktion hätte allerdings zu grösserem Ausmass Verwendung finden können.

Die Funktion der Branches und Commits von Git wurden durch die Arbeit hindurch konsequent verwendet. Dabei wurde die Git Flow Arbeitsweise, abgesehen von den Release Branches und den Hotfix Branches, angewendet (siehe ?? ??). Neben den Feature Branches wurden ausserdem Dokumentation Branches eingeführt. In jedem Dokumentation Branch wurde ein Kapitel der Dokumentation verfasst. Für die Zusammenführungen der wichtigsten Branches wurde das Prinzip der Pull Request (siehe ?? ??) angewendet. Die Pull Request musste dabei für jeden Branch von demjenigen Autor akzeptiert werden, der die Pull Request nicht stellte.

Ein weiterer Vorteil von Git und Github ist die Zugänglichkeit des Projektes. Das gesamte Projekt ist unter folgendem Link einsehbar: <https://github.com/LarsZauberer/Nachzeichner-KI>. Im Projektordner sind vortrainierte Variationen der künstlichen Intelligenz enthalten. Das Projekt auf GitHub erfährt möglicherweise Erweiterungen, die in dieser Arbeit nicht mehr erfasst sind.

Kapitel 6

Zusammenfassung

Diese Untersuchung beantwortet die Frage, inwiefern eine künstliche Intelligenz Strichbilder auf eine physische Weise nachzeichnen kann, sodass diese durch ein automatisches System erkannt werden. Mit Strichbildern sind in diesem Fall Ziffern aus dem MNIST Datenset, Buchstaben aus dem EMNIST Letters Datenset und weitere Motive aus dem QuickDraw Datenset gemeint.

Zur Beantwortung der Fragestellung wird der Begriff des Nachzeichnens definiert. Zu der Definition gehören die Rahmenbedingungen, nach denen eine Tätigkeit als Nachzeichnen gilt, und die Kriterien, welche die Leistung im Nachzeichnen beurteilen. Zu den Rahmenbedingungen gehören unter anderem die physischen Einschränkungen und die ausführbaren Aktionen der KI. Um die Leistung der KI im Nachzeichnen zu bewerten, sind drei Kriterien definiert: Die Übereinstimmung der Pixel, die Erkennbarkeit der Zeichnung und die Geschwindigkeit des Zeichnens. Die Erkennbarkeit der Zeichnung wird durch eine zweite künstliche Intelligenz ermittelt.

Mit der Definition lautet das Ziel, eine künstliche Intelligenz zu entwickeln, welche die gesetzten Rahmenbedingungen erfüllt und eine möglichst gute Leistung nach den definierten Kriterien erzielt. Bei der grundsätzlichen Architektur der KI handelt es sich um ein Deep Q-Learning Modell, das auf der Arbeit hinter 'Doodle-SDQ' (**zhou'learning'2018**) basiert.

Für die Rahmenbedingungen gibt es zwei Ansätze: Eine Grundumgebung und eine physikalische Umgebung. In der Grundumgebung kann sich die KI schrittweise um eine begrenzte Anzahl Pixel auf einer simulierten Zeichenfläche fortbewegen. Ausserdem startet die KI in einer zufälligen Position auf der Zeichenfläche. Die physikalische Umgebung ist von simulierter Physik begleitet. So kann die KI durch Beschleunigungen ihre aktuelle Geschwindigkeit anpassen und sich dadurch fortbewegen. Dabei wird die KI durch simulierte Reibung kontinuierlich abgebremst.

Für die KI existieren weitere Variationen, die dessen Leistung nach einem bestimmten Kriterium verbessern sollen. So existiert ein spezifisches Training auf eine verbesserte Erkennbarkeit und Geschwindigkeit der KI. Durch Kombinationen der Variationen und der Rahmenbedingungen existieren schlussendlich acht Versionen der künstlichen Intelligenz.

Die acht Versionen der künstlichen Intelligenz sind alle auf das Nachzeichnen von Ziffern trainiert. Ein Experiment bestimmt, ob diese Versionen das Nachzeichnen allgemein erlernen. Die Leistung der Versionen wird auf das Nachzeichnen von Strichbildern aus dem Quickdraw und dem EMNIST Letters Datenset gemessen. Wenn die Leistung für diese Strichbilder vergleichbar bleibt mit der Leistung für die Trainingsdaten, ermöglicht das eine positive Antwort für die Fragestellung.

Einige Versionen der künstlichen Intelligenz zeigen hierbei vielversprechende Ergebnisse. Die Grundversion, ohne weitere Variationen, zeichnet in 87% der Fälle eine erkennbare Ziffer, in 75% der Fälle einen erkennbaren Buchstaben, und in 80% der Fälle ein erkennbares Motiv aus dem QuickDraw Datenset.

Abbildungsverzeichnis

Tabellenverzeichnis
