NEUE
KANTONSSCHULE
AARAU

ReSketch AI: Zeichnen mit Deep Reinforcement Learning

Wettbewerbsarbeit für Schweizer Jugend forscht

Ian Wasser, Robin Steiner

25. März, 2023

Betreut durch: Nicolas Ruh
Zweitbeurteilung: Dieter Koch
Externe Betreuung durch: Dr. Michael Tschannen

NKSA G19E

Abstract

ReSketch ist eine künstliche Intelligenz, die Strichbilder nachzeichnen kann. Strichbilder sind beispielsweise Ziffern oder Buchstaben. Die künstliche Intelligenz kann sich beim Zeichnen so bewegen, wie es mit einem echten Stift möglich wäre. ReSketch funktioniert mit Deep Q-Learning, einem Reinforcement Learning Modell. Das Modell basiert dabei auf der Arbeit hinter Doodle-SDQ (Zhou et al., 2018), erfährt aber verschiedene Erweiterungen. Die Leistung von dem Modell wird durch vordefinierte Kriterien evaluiert, deren Werte das Resultat dieser Arbeit ausmachen. ReSketch erreicht eine Übereinstimmung von 90% zwischen der Vorlage und dem nachgezeichneten Bild. Ausserdem kann die KI nach dem Training beliebige Arten von Strichbildern nachzeichnen, obwohl diese lediglich auf das Zeichnen von Zahlen trainiert ist. Eine zweite künstliche Intelligenz, die auf der nachzeichnenden KI basiert, entfernt sich von der ursprünglichen Aufgabe. Diese zweite KI erlernt das selbstständige Zeichnen von einem ausgewählten Motiv, ohne eine Vorlage davon zu erhalten. Zu diesem Zweck werden die generierten Zeichnungen der KI mit einem Klassifizierungsmodell bewertet. Mit einem spezifischen Training dieser generativen KI können verschiedene Handschriften emuliert werden.

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 dabei praktische Arbeit mit intellektueller Forschung verbunden wird. 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.

Der gesamte Code der künstlichen Intelligenz mit all ihren Komponenten und Versionen ist in dem folgenden GitHub Repository einsehbar: https://github.com/LarsZauberer/Nachzeichner-KI.

Vielen Dank an unseren Betreuer der Maturarbeit, Dr. Nicolas Ruh und an unseren Experten im Rahmen des nationalen Wettbewerbs von Schweizer Jugend Forscht, Dr. Michael Tschannen, für die hilfreichen Vorschläge, die ausgeprägte Beratung und das an uns weitergegebene technische Wissen. 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, welches 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 zeichnen?

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 eine Zusammenfassung. Die Theorie wird auf den Teil reduziert, der für das 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) beschreibt das Nachahmen von menschlichem Verhalten durch eine Maschine oder ein System" („Oxford English Dictionary", n. d.) 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. Mustererkennung bedeutet hier, dass der Algorithmus Zusammenhänge zwischen den analysierten Daten erkennt und auf dieser Basis Vorhersagen treffen kann. In einem Satz versucht ein Machine Learning Modell menschliches Urteilsvermögen zu erlernen (Spaulding, 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 Abbildung 2.1). 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.

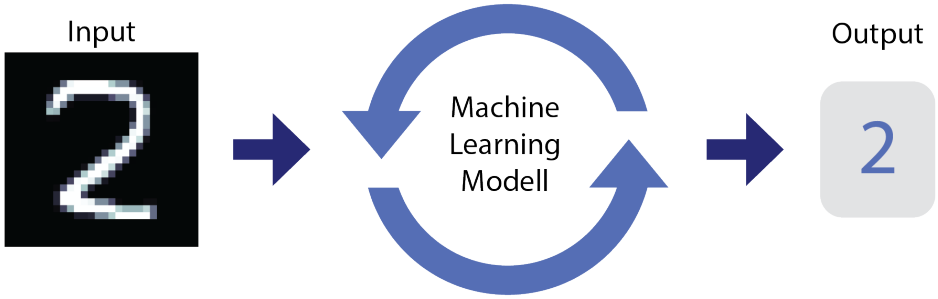

Abbildung 2.1: Erkennung von handgeschriebenen Zahlen durch ein Machine Learning Modell. (eigene Abbildung)

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 (Liu & Wu, 2012). Die Zielvariable muss dabei im Voraus für jeden Datenpunkt in den analysierten Daten durch einen Menschen festgelegt sein (Trahasch et al., 2020). Das heisst, ausgedrückt durch den Fachbegriff, dass die Daten labeled sein müssen (Serrano, 2021). Weitere Teilbereiche von Machine Learning sind Unsupervised Learning und Reinforcement Learning (Arora, 2020). Siehe Abschnitt 2.2 für eine tiefgreifendere Einführung in Reinforcement Learning.

Machine Learning Modelle sind hauptsächlich in der Programmiersprache Python implementiert (Sadie, 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 („TensorFlow“, 2015)(„Keras: the Python deep learning API“, 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 2.1).

Bei den Daten, die das Machine Learning Modell analysiert, handelt es sich in diesem Fall um das MNIST Datenset (LeCun et al., 1998). Dieses Datenset wurde 1998 vom NIST (National Institute of Standards and Technology) in den USA veröffentlicht und beinhaltet 70′000 Bilder von handgeschriebenen

Ziffern (Lecun et al., 1998). Jedes Bild hat eine Auflösung von 28×28 Pixeln (siehe Abbildung 2.2 für Beispiele).

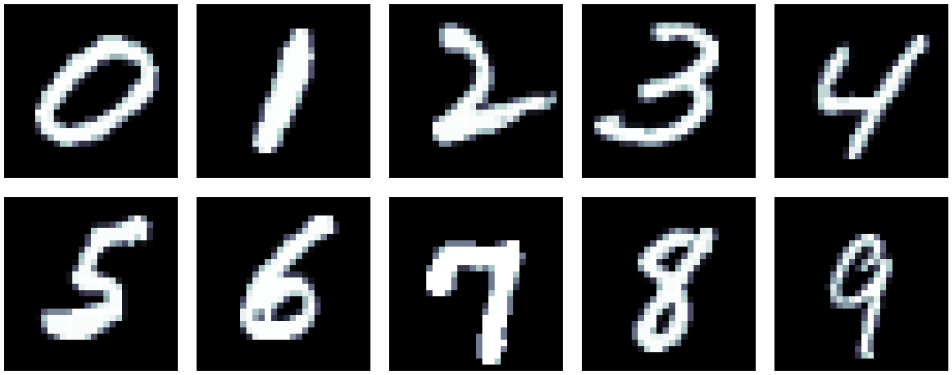

Abbildung 2.2: Beispiele aus dem MNIST Datenset. (Eigene Abbildung)

Ein Machine Learning Modell durchläuft ein Training gefolgt von einer Testphase. 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 2.1). 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 2.1.2) 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” („Neuronales Netz“, 2021). Ein Beispiel für ein neuronales Netz ist das menschliche Gehirn. Künstliche neuronale Netze modellieren biologische neuronale Netze in der Form von Programmcode (Nagyfi, 2018). 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 Abbildung 2.3) (Pramoditha, 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, 2015)

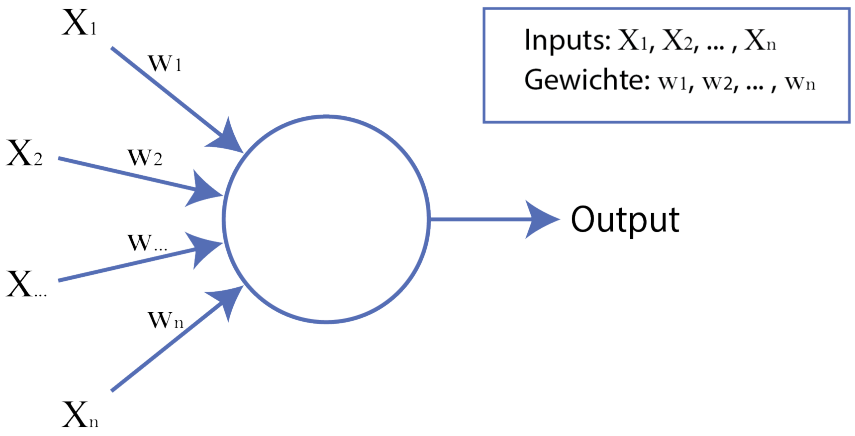

Abbildung 2.3: Perzeptron Neuron. (eigene Abbildung)

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, 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 Abbildung 2.4) (Kumar, 2019).

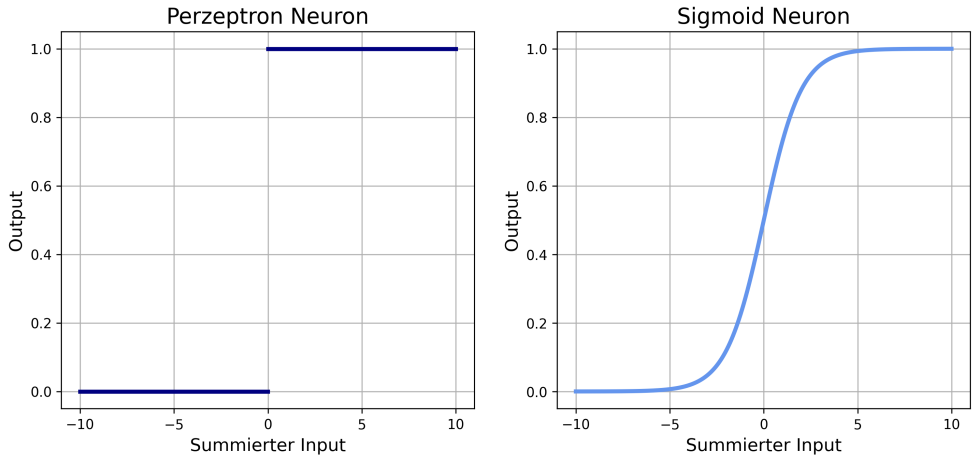

Abbildung 2.4: Vergleich des Outputs eines Perzeptron Neurons und eines Sigmoid Neurons. (eigene Abbildung)

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 (Agarap, 2019). Softmax wird im Vergleich meistens für die Activation Function des *Output Layers* verwendet (Koech, 2021). 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.

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. Neuronale Netze haben mindestens einen *Input Layer* und einen *Output Layer* (Nielsen, 2015). Der Input Layer umfasst die Daten, welche das neuronale Netz beurteilen soll. Im Beispielproblem (siehe 2.1) bestände 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).

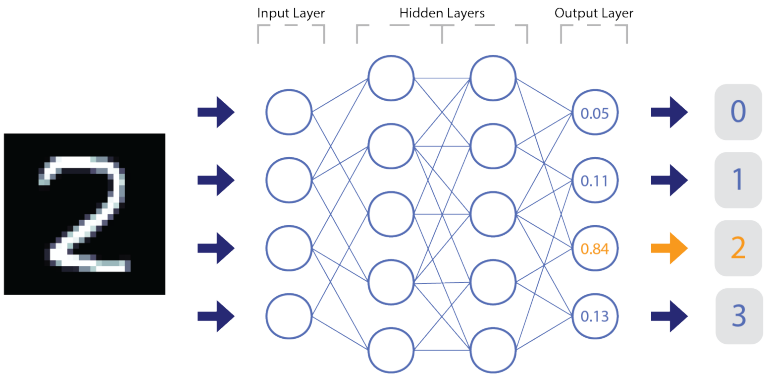

Abbildung 2.5: Neuronales Netz mit beschrifteten Layers. (eigene Abbildung)

Zwischen dem Input Layer und dem Output Layer kann es weitere *Hidden Layers* geben (Malik, 2019). Unterschiedliche Arten von Hidden Layers mit unterschiedlichen Funktionen existieren. Zwei der meist verwendeten Layers sind Fully Connected (Dense) Layers und Convolutional Layers (Unzueta, 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 Abbildung 2.6). Die Funktion von Convolutional Layers umfasst, wichtige Merkmale aus dem Input hervorzuheben (O'Shea & Nash, 2015). Concatenation Layers (Jayawardana, 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 (Kranz, 2019).

Ein Machine Learning Modell passt während dem Training (siehe 2.1.1) 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 (Rumelhart et al., 1985).

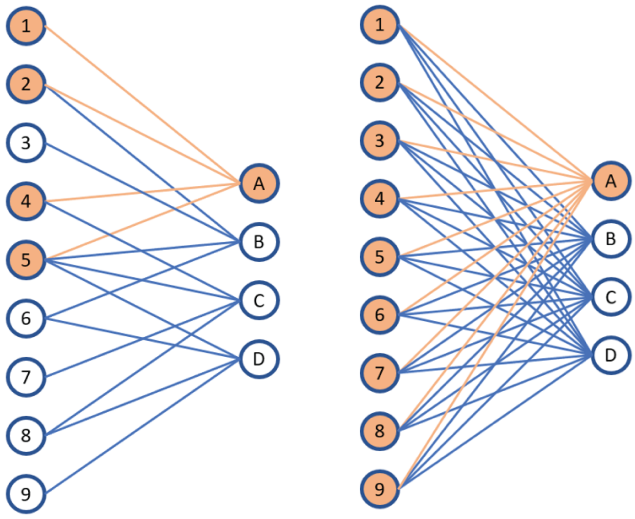

Abbildung 2.6: Vergleich zwischen Convolutional Layers (links) und Fully Connected Layers (rechts). (Unzueta, 2022)

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, 2022), aber ihr optimaler Wert ist im Voraus nicht bekannt.

Hyperparameter können unter anderem durch den Baysian Optimization Algorithmus optimiert werden (Agnihotri & Batra, 2020). Dieser Algorithmus versucht, den Output einer Black Box Funktion zu maximieren oder zu minimieren (Garnett, 2022, S. 15). Eine Black Box ist ein häufig komplexes System, dessen inneren Vorgänge nicht betrachtet werden („Black Box (Systemtheorie)“, 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).

Abbildung 2.7: Prinzip einer Black Box Funktion. („Black Box (Systemtheorie)“, 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. Um ein Machine Learning Modell als eine Black Box Funktion für den Baysian 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 2.1.1). Die inneren, nicht betrachteten Vorgänge in der Black Box Funktion entsprechen in diesem Fall dem Training des Modells.

Der Baysian Optimization Algorithmus kann bis zu 20 Hyperparameter zuverlässig optimieren (Moriconi et al., 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, 2022, S. 7). 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. 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 2.1) 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 2.1.1) mit dieser interagiert und dadurch Erfahrungen sammelt. (Verma & Diamantidis, 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 et al., 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 (R. S. Sutton & Barto, 2014, S. 53). 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 et al., 2013, S. 2). Die Observation umfasst alle Daten der Umgebung, die für die Entscheidung relevant sind. Sein Entscheidungsprozess findet durch ein neuronales Netz statt (siehe 2.1.2). 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* (R. S. Sutton & Barto, 2014, S. 67) genannt. Jede Action im Action-Space besitzt einen *Q-Value*, der dem Output des zugehörigen Neurons entspricht (siehe Abbildung 2.8).

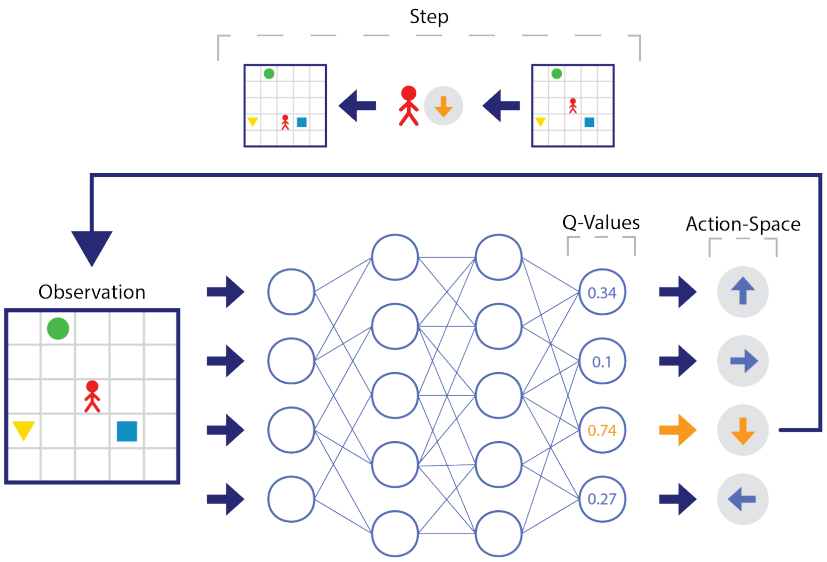

Abbildung 2.8: Funktionsweise eines Reinforcement Learning Modells. (eigene Abbildung)

Die schlussendliche Entscheidung der auszuführenden Action wird durch eine *Policy* bestimmt. Softmax und Epsilon Greedy sind 2 Beispiele für eine Policy (R. S. Sutton & Barto, 2014, S. 34).

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. (Coggan, 2004). Die Softmax Policy wählt Actions auf Basis einer Wahrscheinlichkeitsverteilung (R. Sutton, 1997). 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 (R. S. Sutton & Barto, 2014, S. 75). 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 et al., 2013, S. 3). 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 Abbildung 2.9). 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 (R. S. Sutton & Barto, 2014, S. 57). Der akkumulierte Reward entspricht der Summe der Rewards aus jedem Step (in einer Episode. Beachte den nächsten Abschnitt).

Das Training läuft in *Episodes* (R. S. Sutton & Barto, 2014, S. 14). 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 die das neuronale Netz trainiert. Das neuronale Netz trainiert also auf Daten aus der Vergangenheit der Umgebung. Diese Strategie nennt sich Experience Replay (Mnih et al., 2013, S. 5). Ausserdem trainiert das neuronale

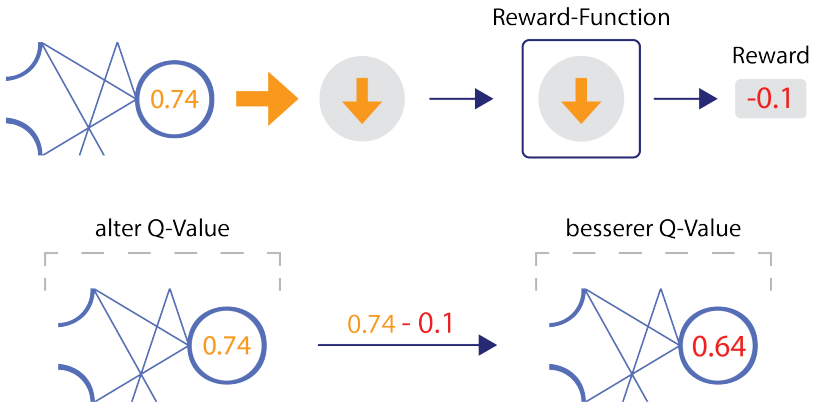

Abbildung 2.9: Funktionsweise einer Reward-Function. (eigene Abbildung)

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. (van Heeswijk, 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 Bilder durch das Platzieren von Elementen wie Strichen (Hertzmann, 2002). Beispiele für Arbeiten in diesem Bereich sind Strokenet (Zheng et al., 2018) und "Learning to Paint With Model-based Deep Reinforcement Learning" (Huang et al., 2019) Andere Ansätze simulieren die Führung eines Stiftes. (siehe Abbildung 2.10) Ein Beispiel dafür ist Doodle-SDQ (Zhou et al., 2018).

2.3.1 Doodle-SDQ

Doodle-SDQ ist ein Computerprogramm, das unter anderem durch Reinforcement Learning und spezifischer durch Deep Q-Learning (siehe 2.2.1) erlernt, Strichbilder aus dem Google QuickDraw Datenset („Quick, Draw! Image Recognition", 2022) nachzuzeichnen. Nachfolgend sind diejenigen Aspekte von Doodle-SDQ beschrieben, die für diese Arbeit relevant sind.

Die QuickDraw Bilder, die Doodle-SDQ nachzeichnen soll, sind zu einer einheitliche Grösse von 84×84 Pixeln verarbeitetet (Zhou et al., 2018, S. 7). Der Agent kann sich auf einer leeren Zeichenfläche von derselben Grösse

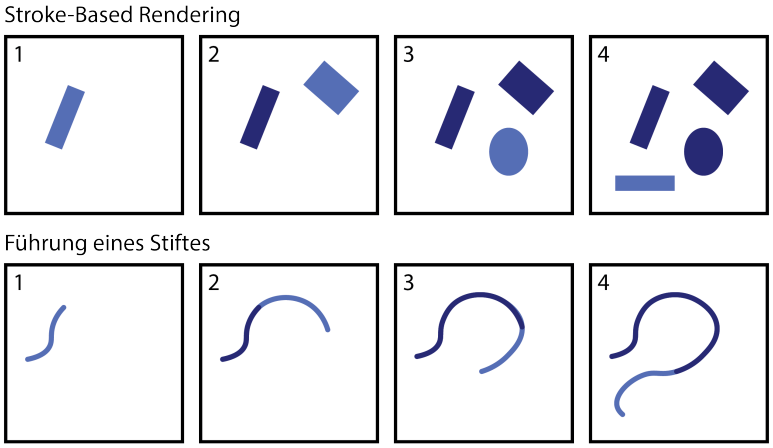

Abbildung 2.10: Vergleich zwischen Stroke-Based Rendering und dem Führen eines Stiftes. (eigene Abbildung)

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 et al., 2018, S. 5). 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 2.2.1) ist in zwei Teile gegliedert: der Global Stream und der Local Stream (Zhou et al., 2018, S. 4). 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 et al., 2018, S. 5)), in dem sich der Agent in einem Step bewegen kann. Das zweite Bild beschreibt denselben Bereich von der Zeichenfläche (Zhou et al., 2018, S. 4 ff.). Der Global Stream und der Local Stream werden durch eine Concatenation Layer (siehe 2.1.2) 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 3.5). 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 3.2). Der andere Teil umfasst die Entwicklung der KI. Dazu gehört die Entwicklung einer grundlegenden Architektur (siehe 3.1), sowie die Vollendung der KI mit verschiedenen Variationen und Ansätzen (siehe 3.3). Die Variationen sind dabei Versuche, die Leistung der KI zu maximieren oder ihr Verhalten zu verändern. Ein 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 eine Vorlage kreiert (siehe 3.4).

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 2.2.1). Das Grundprogramm beinhaltet dabei keine Reward-Function und ist somit keine funktionale Version der KI. Das Grundprogramm ist in Python unter der Verwendung des Keras Frameworks implementiert (siehe 2.1).

3.1.1 Doodle-SDQ als Basis

Das Reinforcement Learning Modell des Grundprogramms basiert auf Doodle-SDQ (siehe 2.3.1). Von Doodle-SDQ ist das neuronale Netz, bezogen 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 2.1) 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 2.3.1) 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 und eine Eins einen weissen (gezeichneten) Pixel. Die genaue Architektur des neuronalen Netzes ist in der folgenden Abbildung angegeben (siehe Abbildung 3.1).

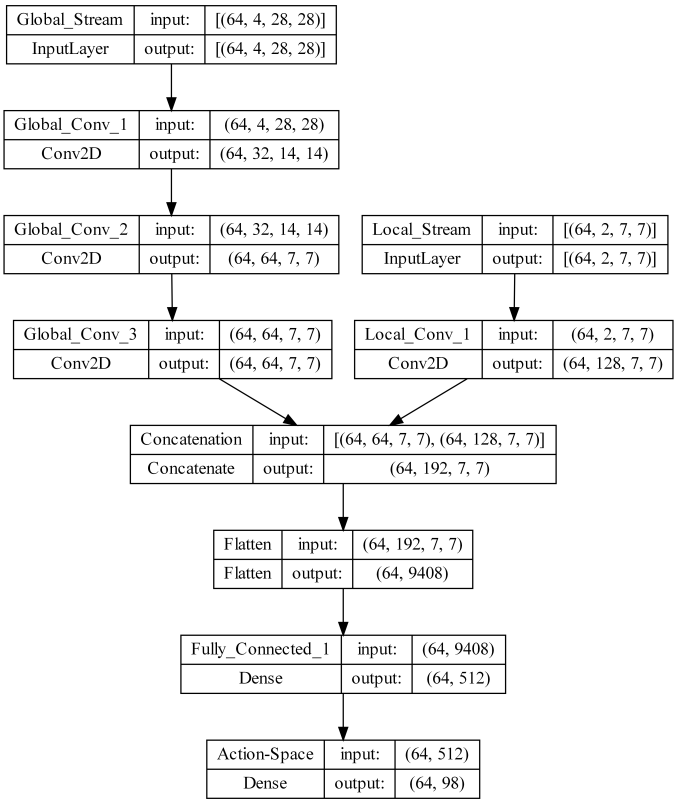

Abbildung 3.1: Architektur des neuronalen Netzes im Grundprogramm. Jeder Block repräsentiert einen Layer. Die Form des Inputs und des Outputs ist von jedem Layer angegeben (eigene Abbildung, mit Keras erstellt)

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 2.2.1) 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).

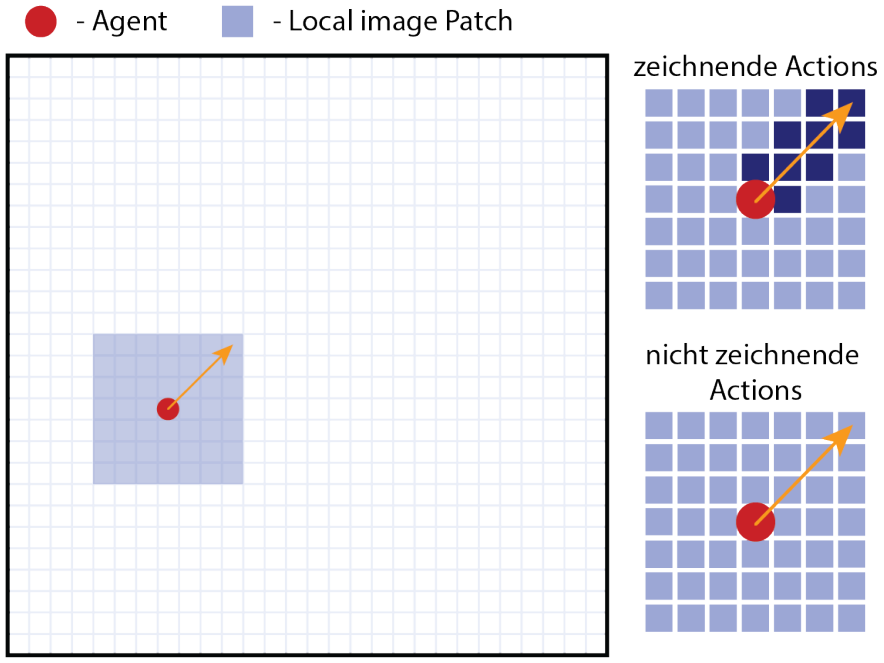

Abbildung 3.2: Action-Space im Grundprogramm

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 an 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. (siehe 2.2)

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 2.2.1) 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 deaktiviert werden. In diesem Fall wird die Action als unzulässig behandelt.

3.1.3 Präparierung der Daten und Optimierung

Die Trainingsdaten bestehen aus 30′000 Bildern von handgeschriebenen Ziffern aus dem MNIST Datenset (siehe 2.1). Weitere 12′000 Bilder 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 Abbildung 3.3). So stimmen die Bilder mit den Zeichnungen, die der Agent produzieren kann, überein.

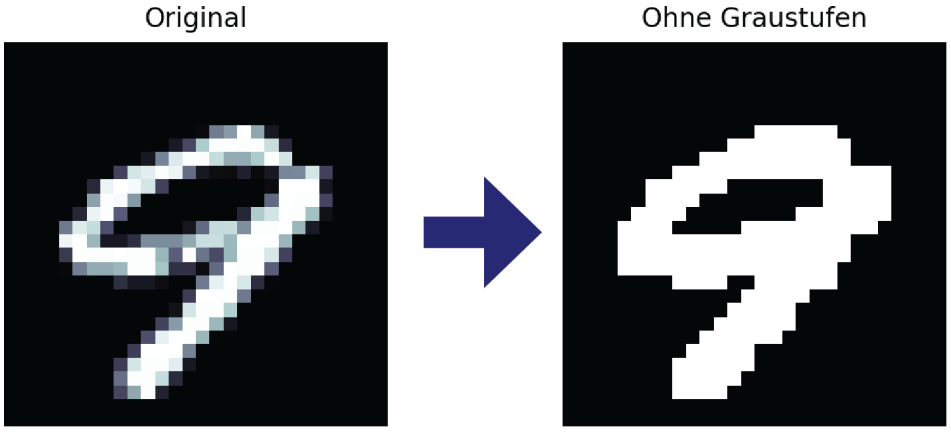

Abbildung 3.3: Entfernung der Graustufen im MNIST Datenset. (eigene Abbildung)

Das Grundprogramm trainiert mit 5′000 zufälligen Bildern aus den Trainingsdaten, 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 (siehe 3.1.2) nicht früher gewählt wird.

Die Hyperparameter aller Versionen der KI sind durch den Bayesian Optimization Algorithmus optimiert (siehe 2.1.3). Die Implementierung des Algorithmus in Python stammt von (Nogueira, 2014). Der Algorithmus ändert sich für die verschiedenen Variationen der KI nicht und ist somit Teil des Grundprogramms. Mit jeder Iteration des Baysian Optimization Algorithmus trainiert das Reinforcement Learning Modell für eine vom Algorithmus selbst bestimmte Anzahl Episodes. Die Zielvariable, die durch den Baysian Optimization Algorithmus maximiert werden soll, wird am Ende jeder Iteration des Trainings in der Testumgebung berechnet (siehe 3.5.1). Mit welchem Kriterium (siehe 3.2) der Wert der Zielvariable berechnet wird, 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 3.1). 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 Abbildung 3.4).

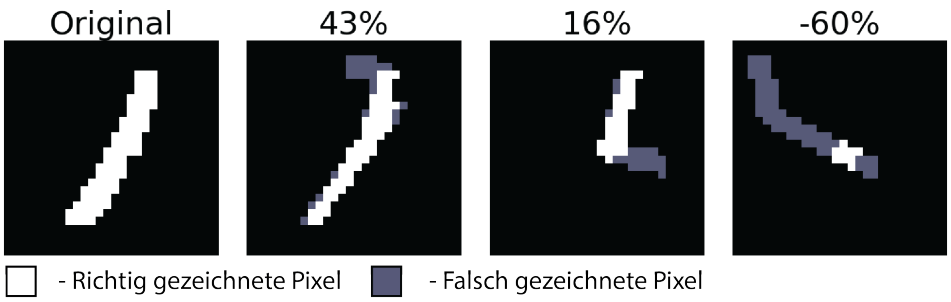

Abbildung 3.4: Drei Beispiele für den Wert des Kriteriums der Übereinstimmung. (eigene Abbildung)

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 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 2.1.1). Diese klassifizierden Machine Learning Modelle sind spezifisch auf Bilder ohne Graustufen trainiert, welche die nachzeichnende KI dieser Arbeit produzieren kann. Der Output Layer der klassifiziernden KI hat Softmax mit einer hohen Temperatur als Activation Function (siehe 2.1). 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 Tabelle 3.1) aufgeführt. Das neuronale Netz der Modelle stammt aus einem online Machine Learning Tutorial (DataFlair, 2020).

Art	Trainiert mit	Genauigkeit [%]
Ziffern	MNIST	98.9
Buchstaben	EMNIST Letters	90.7
Strichbilder von Objekten	Auswahl aus QuickDraw	96.6

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 bei der letzten zeichnenden Action des Agents definiert. Falls der Agent nach dem Zeichnen weiterhin nicht zeichnende Actions ausführt, so werden diese nicht zu der Zeichnung gezählt, weil diese 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 zweimal oder häufiger zeichnet, 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 3.1) zu einer funktionalen, nachzeichnenden KI. Die meisten Variationen unterscheiden sich dabei hauptsächlich im Fokus auf die definierten Kriterien (siehe 3.2). Jede der Variationen versucht, die Leistung in einem ausgewählten Kriterium zu maximieren. Eine Anpassung der Reward-Function (siehe 2.2.1) zielt auf diesen Effekt ab. Die verschiedenen Variationen sind teilweise untereinander kombinierbar. Eine der Variationen (siehe 3.3.6) 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 3.2.1). 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 berechnet sich der Reward aus der Veränderung der prozentualen Übereinstimmung, ausgelöst durch die Action in einem Step. Der akkumulierte Reward (siehe 2.2.1) enstspricht dem absoluten Wert der prozentualen Übereinstimmung.

Die prozentuale Übereinstimmung hat 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 3.2.2). 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 3.1.2). 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 soll. 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 für die Stopp Action 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 in diesem Bereich 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:

$$f_{speed} = 1 - \left(\frac{t}{64}\right)^{2.5}$$

(Siehe Abbildung 3.5) 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.

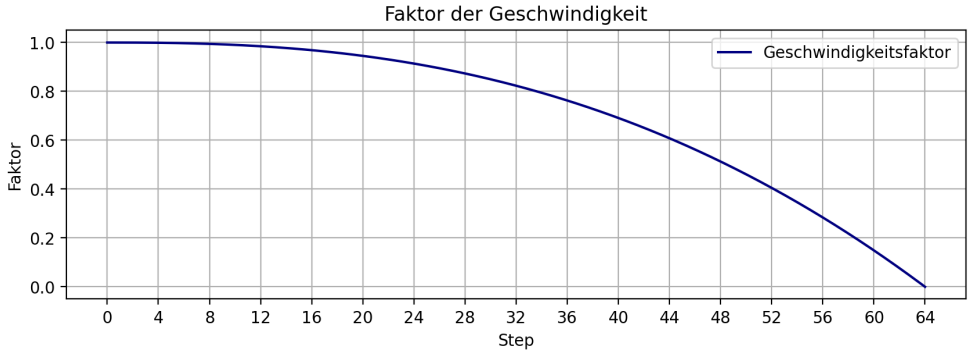

Abbildung 3.5: Veränderung des mit dem Reward zu multiplizierenden Faktor in Abhängigkeit der Anzahl Steps (eigene Abbildung)

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 Actions einen negativen Reward von -0.01 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 2.2.1). 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 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 2.2.1) 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 3.6).

Mit dem gewählten Beschleunigungsvektor $\vec{a}(t)$ berechnet sich die Geschwindigkeit im nächsten Step $t+1$ aus dem aktuellen Step t durch folgende Formel:

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

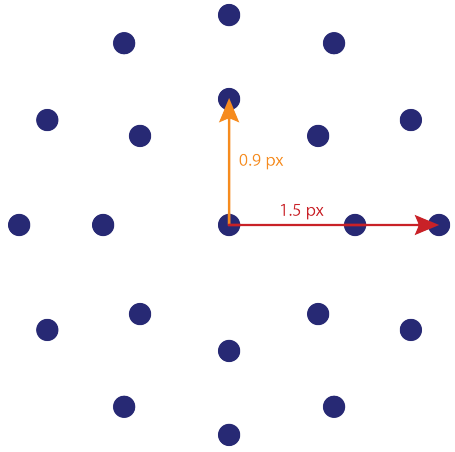

Abbildung 3.6: Action-Space in der physikalischen Umgebung. (eigene Abbildung)

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 2.2.1) zu integrieren, wird der Local image Patch (siehe 2.3.1) 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 3.7).

Ein weiteres Problem ist, dass der Agent sich durch seine Geschwindigkeit aus den vorgegebenen Grenzen der Zeichenfläche begeben kann. Im Grundprogramm (siehe 3.1.1) 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.

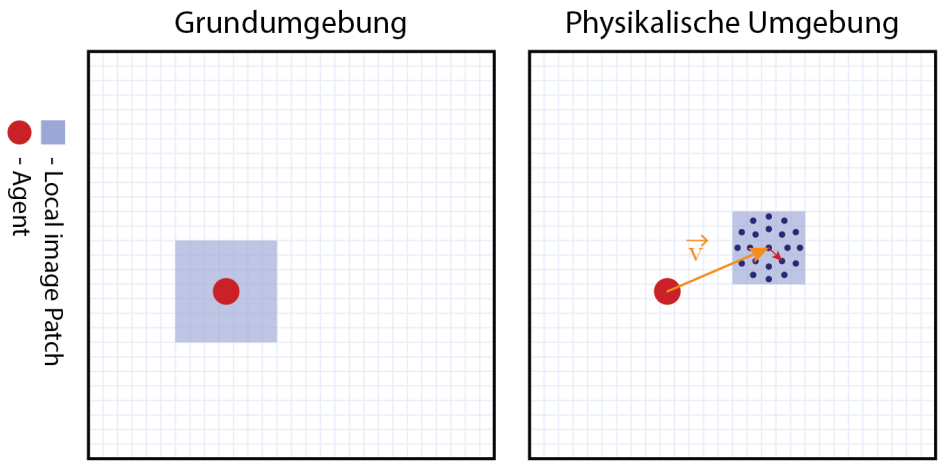

Abbildung 3.7: Angabe der Geschwindigkeit durch eine Verschiebung des Local image Patches. (eigene Abbildung)

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 Variationen auf dem Grundprogramm (siehe 3.1) 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 2.1.2), 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 des gewünschten Motivs 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 des gewünschten Motivs weitgehend erlernt hat, erlernt diese ihre eigentliche Aufgabe, indem sie zunehmend seltener eine Vorlage erhält. Die generative KI trainiert insgesamt für 6′000 Episodes.

Die generative KI verwendet die Basis Reward-Function (siehe 3.3.1) 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, dann 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 auf dem Kriterium der prozentualen Übereinstimmung, welches die Basis Reward-Function verwendet, sondern auf dem Kriterium der Erkennbarkeit (siehe 3.2.2). Da ein generativer Zeichner ein Motiv ohne weitere Spezifikationen zeichnen soll, ist die Erkennbarkeit des Motives von grösster Bedeutung. Die generative KI verwendet deswegen neben der Basis Reward-Function ebenfalls die auf Erkennbarkeit spezialisierte Reward-Function (siehe 3.3.2).

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 deterministsch, da die KI stets die Action mit dem höchsten Q-Value wählt (siehe 3.5). Zusammen mit der unveränderlichen Ausgangslage führt das dazu, dass die KI nur eine einzige Zeichnung produzieren kann.

3.4.2 Nicht deterministische Zeichnungen

Damit die generative KI sich nicht mehr deterministsch bewegt und stattdessen unterschiedliche Bilder zeichnen kann, benötigt der Zeichenprozess ein Element von Zufall. Es gibt verschiedene Ansätze, um diesen Zufall zu implementieren.

Softmax

Der erste Ansatz ist, selbst nach dem Training der generativen KI die Softmax Policy zur Wahl der Actions zu verwenden (siehe 2.2.1). 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 Wahrscheinlichkeitsverteilung gewählt werden. Als weiterer Vorteil ist

die Wahrscheinlichkeitsverteilung durch die Softmax Temperatur variierbar. Diese Eigenschaft ist für den Erfolg von diesem Ansatz 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.

Die Freiheit der Wahl der Actions wird für die nicht deterministische generative KI folgendermassen eingeschränkt: Der Agent kann nur unter den fünf Actions mit den höchsten Q-Values (siehe 2.2.1) auswählen. Ohne diese Beschränkung würde die Wahl immer noch zu häufig auf Actions mit einer geringen Wahrscheinlichkeit (also einem geringen Q-Value) fallen. Ausserdem wird die Wahrscheinlichkeitsverteilung durch Softmax nur für die fünf wählbaren Actions berechnet. Die Softmax Temperatur ist für diese Berechnung tief (kleiner als 1), damit die Differenz zwischen den Wahrscheinlichkeiten der Actions hoch bleibt.

Um im Test mit der Softmax Policy umgehen zu können, sollte die generative KI bereits mit der Softmax Policy trainieren. Es stellt sich heraus, dass die KI mit der Softmax Activation Function für den Output Layer einen deutlich tieferen akkumulierten Reward erreicht und dadurch auch keine guten Leistungen erbringt. Deswegen wird Softmax nur für die Wahl der Actions verwendet und nicht als Activation Function. Dadurch beeinflusst Softmax das neuronale Netz in keiner Weise und die KI erreicht ihre übliche Leistung (siehe 4.1.1). Die Softmax Temperatur wird über das Training hinweg linear um einen kleinen Betrag erhöht. Dadurch steigt der Einfluss des Zufalls im Verlaufe des Trainings.

Random-Noise

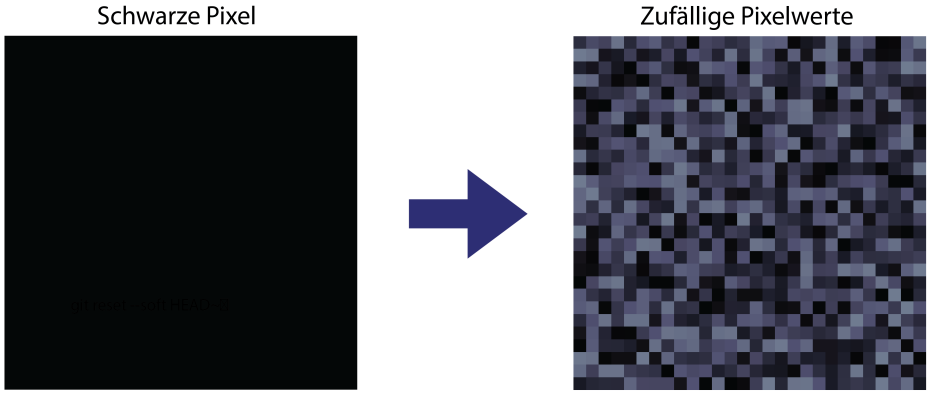

Abbildung 3.8: Verwendung von zufälligen Werten als Eingabe in das neuronale Netz (eigene Abbildung)

Der zweite Ansatz, um die generative KI nicht deterministisch zeichnen zu lassen, verändert den Input in das neuronale Netz. Der Channel, mit dem der KI eine Vorlage gezeigt wird, kann um ein Zufallselement erweitert werden. Der generativen KI wird generell in diesem Channel ein schwarzes Bild anstelle einer Vorlage gezeigt. Werden die Werte der Pixel dieses schwarzen Bildes mit zufälligen Werten zwischen 0 und 0.5 ersetzt (siehe Abbildung 3.8), so verändert sich der Input in das neuronale Netz und somit auch der Output. Die KI trifft also Entscheidungen auf der Basis von zufälligen Werten, wodurch sich das Resultat der Zeichnungen unterscheiden kann. Damit dieser Ansatz funktioniert, muss bereits während dem Training dieses Zufallselement implementiert 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 3.2). Die Auswertung wird für die Variationen der nachzeichnenden KI und die Variationen der generativen KI separat beschrieben, da diese sich unterscheiden.

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 Tabelle 3.1). 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 Abbildung 3.9). Die Bilder in den drei Datensets werden im Training nicht verwendet, sind aber gleich verarbeitet wie die Trainingsdaten (siehe 3.1.3).

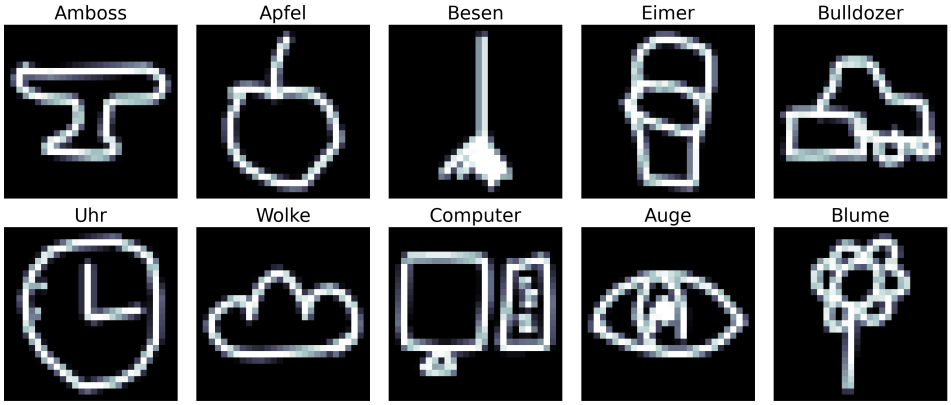

Abbildung 3.9: Beispiele der verwendeten Motive aus dem QuickDraw Datenset. (eigene Abbildung)

Eine Testumgebung erfasst die Leistung der KI (siehe 2.1.1). 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 diese während dem Test nicht. Zweitens wählt der Agent die Actions deterministisch. In jedem Step wird diejenige Action mit dem höchsten Q-Value gewählt.

Im Test zeichnet die KI 1′000 Strichbilder nach. Am Ende jeder Zeichnung wird der Zahlenwert der verschiedenen Kriterien (siehe 3.2) 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 3.2). Die übrigen Kriterien sind ohne eine nachzuzeichnende Vorlage nicht auswertbar. 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 3.4.2) wie im Training der jeweils getesteten Variation.

Im Test generiert die KI 1000 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 fünf Motive sind: eine Null, eine Zwei, eine Acht, ein F und eine Blume aus dem QuickDraw Datenset (siehe Abbildung 3.10). Wenn eine Variation im Test die Softmax Policy verwendet (siehe 2.2.1), dann ist die Softmax Temperatur unabhängig von derjenigen im Training bestimmbar.

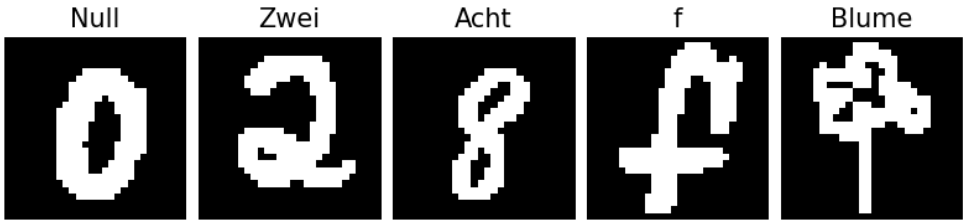

Abbildung 3.10: Beispiele der getesteten Motive der generativen KI (eigene Abbildung)

Wie im Test der nachzeichnenden KI (siehe 3.5.1), entspricht die Leistung einer Variation dem Durchschnitt der Werte in den verwendeten Kriterien für jede 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 3.2.1)
- Rec (siehe 3.2.2)
- Speed (siehe 3.2.3)
- Drawtime (siehe 3.2.4)
- Overdrawn (siehe 3.2.5)

Die Abkürzungen der Variationen der nachzeichnenden KI lauten:

- Base (siehe 3.3.1)
- Rec (siehe 3.3.2)
- Speed (siehe 3.3.3)
- No-Penlift (+ Speed) (siehe 3.3.4)
- No-Overdraw (siehe 3.3.5)
- Physics (siehe 3.3.6)

Die Abkürzungen der Variationen der generativen KI lauten:

- Softmax (siehe 3.4.2)
- Random-Noise (siehe 3.4.2)

4.1 Tabellen

Der erste Teil der Resultate besteht aus Tabellen, welche die Daten über die Leistung der KI aus der Testumgebung (siehe 3.5) 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 — 1000 Tests

	Sim [%]	Rec [%]	Speed	Drawtime [%]	Overdrawn
Base	90.8	97.1	54.7	0.73	269
Rec	76.8	95.2	29.7	83.4	114
Speed	80.2	96.7	20.5	90.5	78
No-Penlift	79.7	96.4	20.8	0.981	97
No-Overdraw	83.3	97.3	36.3	54.5	40
Physics	69.9	88.9	59.3	61.8	161

Tabelle 4.2: Testen auf EMNIST Letters Datenset — 1000 Tests

	Sim [%]	Rec [%]	Speed	Drawtime [%]	Overdrawn
Base	89.6	85.0	60.5	81.9	315
Rec	76.6	76.6	45.2	86.4	201
Speed	78.0	77.0	32.6	92.2	130
No-Penlift	77.8	73.4	31.5	100	151
No-Overdraw	80.2	77.6	50.8	56.4	60
Physics	67.9	57.2	61.0	74.7	190

Tabelle 4.3: Testen auf QuickDraw Datenset — 1000 Tests

	Sim [%]	Rec [%]	Speed	Drawtime [%]	Overdrawn
Base	81.8	93.7	56.5	73.9	227
Rec	71.6	82.4	42.3	84.1	157.9
Speed	74.5	85.7	29.5	89.3	93
No-Penlift	72.7	84.4	28.1	99.0	114
No-Overdraw	78.8	89.2	44.3	62.8	57
Physics	60.3	74.5	59.5	66.7	144

4.1.2 Tabellen der generativen KI

Die Tabellen der generativen KI beschreiben die Leistung der beiden Variationen auf die ausgewählten Kriterien. Für beide Variationen gibt es eine Tabelle, die jeweils die Leistung dieser Variation für die fünf verschiedenen Motive (siehe 3.5.2). beschreibt.

Tabelle 4.4: Testen der Softmax Variation — 1000 Tests

	Rec [%]	Speed	Drawtime [%]
Null	90.0	20.9	85.7
Zwei	92.9	16.6	84.1
Acht	83.9	37.6	75.4
F	92.8	19.2	84.8
Blume	94.8	50.9	75.7

Tabelle 4.5: Testen der Random-Noise Variation — 1000 Tests

	Rec [%]	Speed	Drawtime [%]
Null	99.4	15.7	91.0
Zwei	100	13.4	91.6
Acht	99.4	17.0	93.0
F	97.9	20.7	77.3
Blume	99.1	61.8	73.6

4.2 Bildersammlung

Eine Sammlung von gezeichneten Strichbildern von ausgewählten Variationen 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

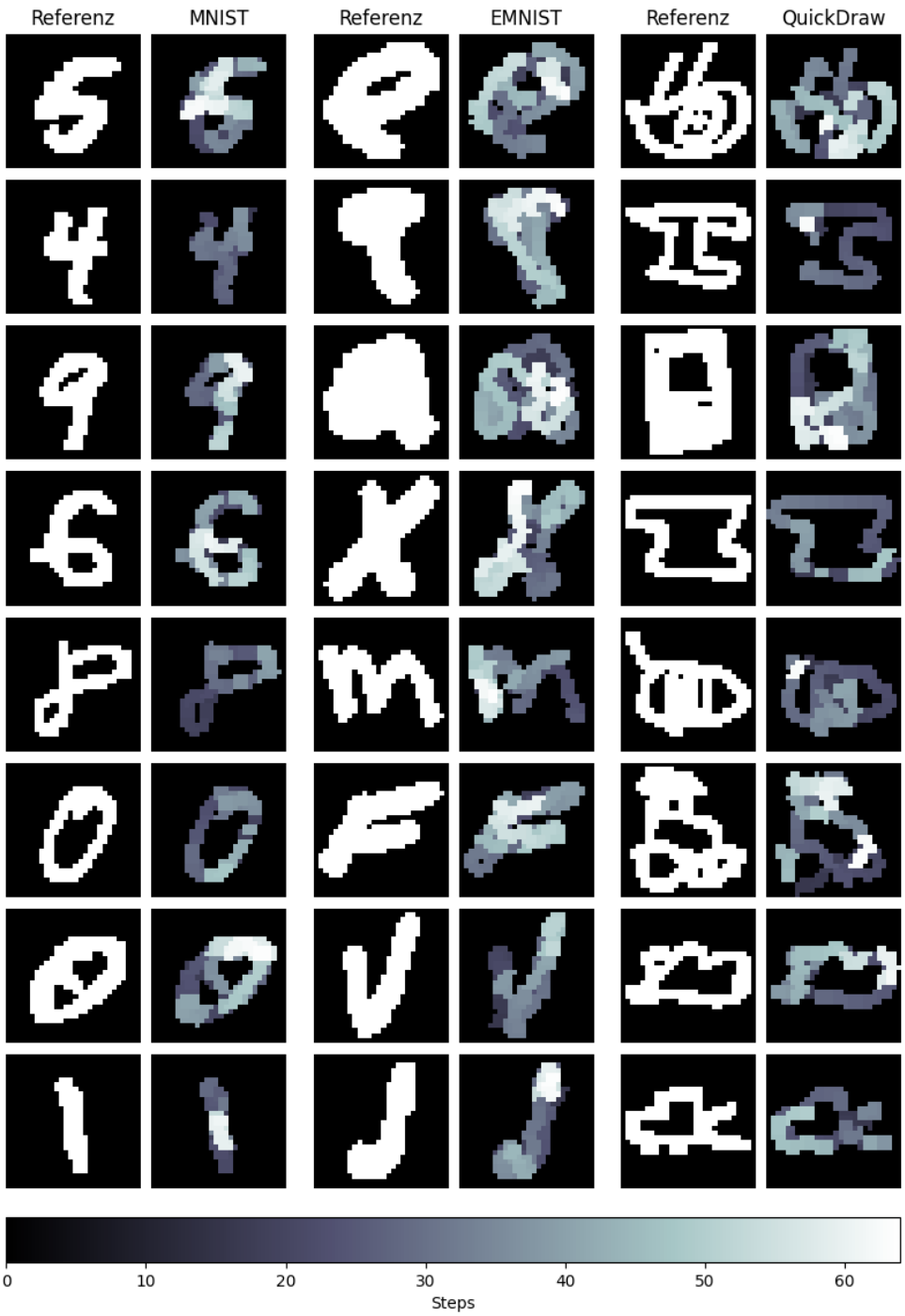

Abbildung 4.1: Bildersammlung: Base Variation

Abbildung 4.2: Bildersammlung: Speed Variation

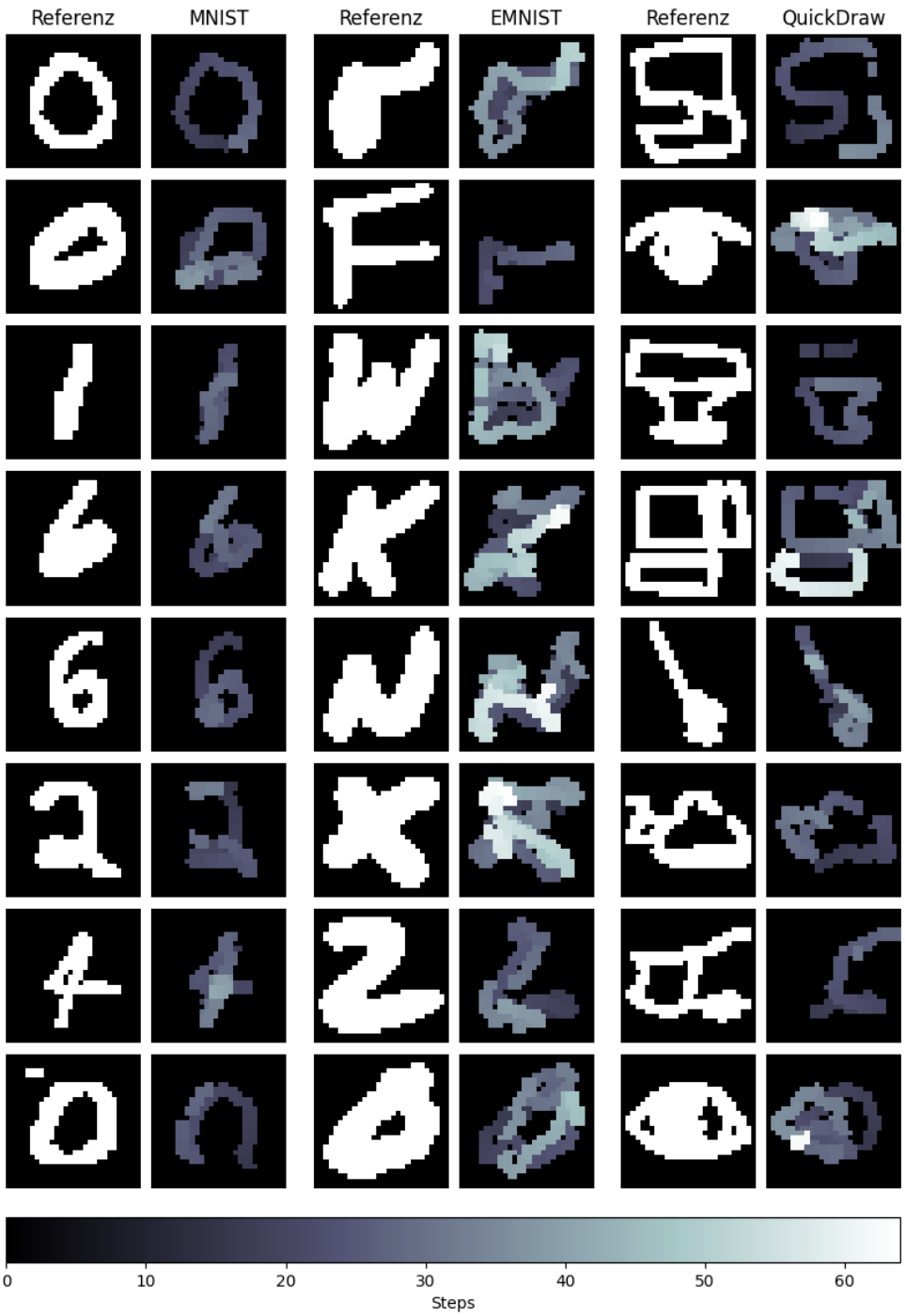

Abbildung 4.3: Bildersammlung: Rec Variation

Abbildung 4.4: Bildersammlung: Overdraw Variation

4.2.2 Bildersammlung der generativen KI

Die Bildersammlungen für die Variationen der generativen KI haben fünf Spalten. Jede Spalte zeigt Zeichnungen der KI von einem der ausgewählten Motive (siehe 3.5.2).

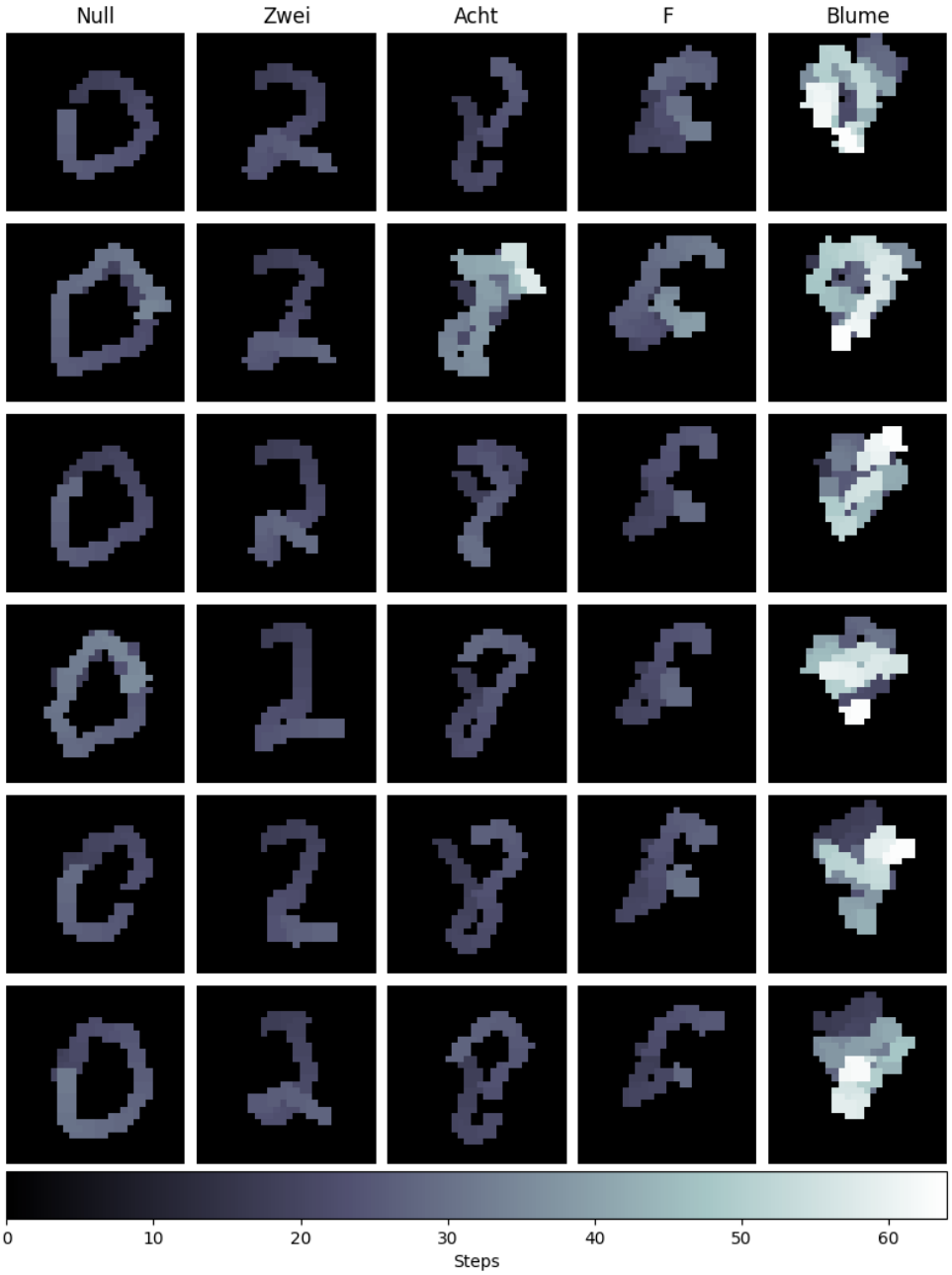

Abbildung 4.5: Bildersammlung: Softmax Variation

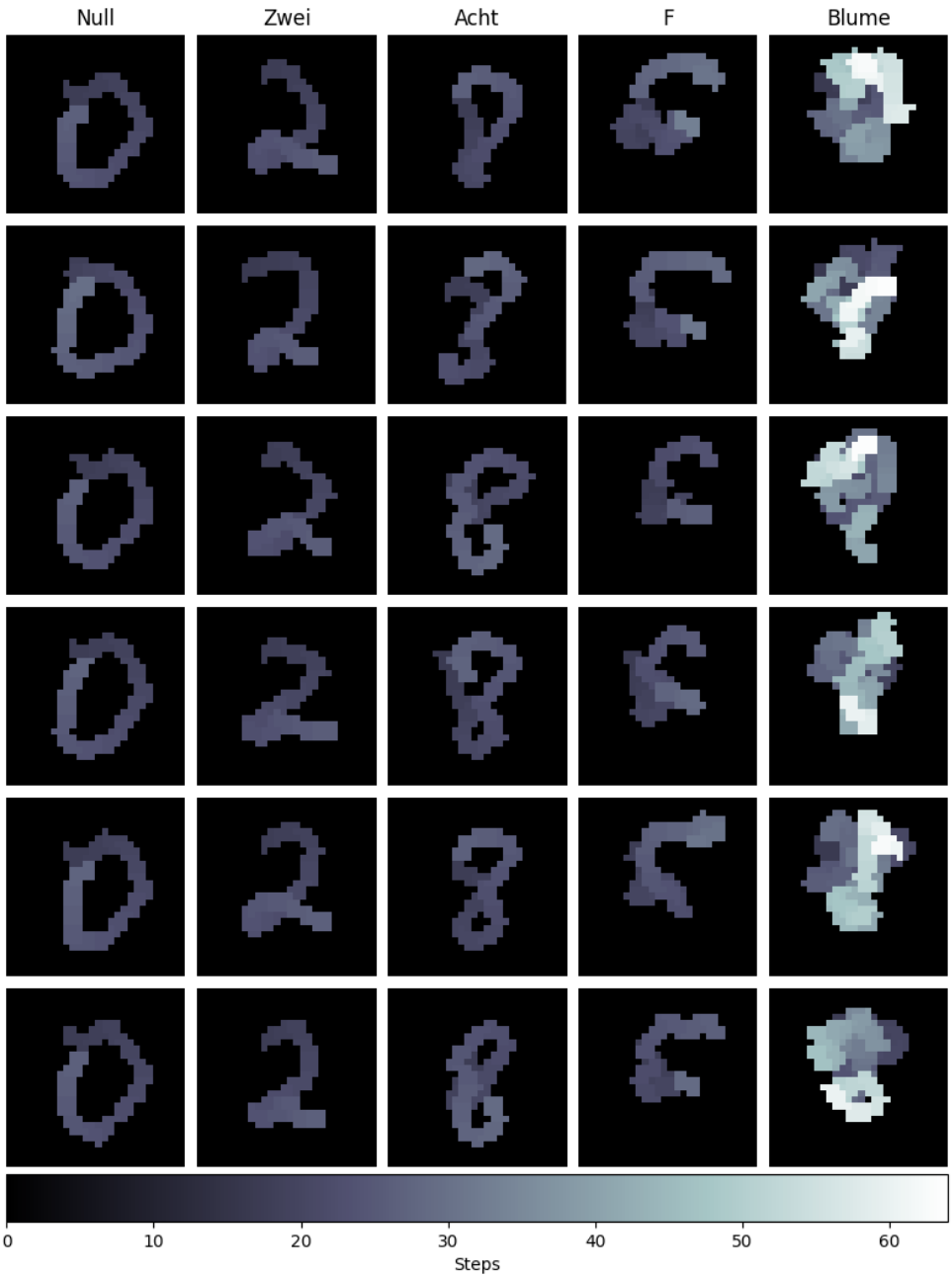

Abbildung 4.6: Bildersammlung: Random-Noise Variation

Kapitel 5

Diskussion

Die Diskussion analysiert die Resultate der Methode, um daraus eine Antwort auf die Fragestellung zu bilden. 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 4.1.1). Die Architektur einer nachzeichnenden KI kann somit so aussehen, wie sie in dieser Arbeit beschrieben ist (siehe 3.1).

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 3.2). 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 wenige 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 98% erreicht. Die klassifizierende KI für Buchstaben ist allerdings deutlich schlechter mit einer Genauigkeit von 90%, was das Training und die Resultate beeinflusst.

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 von einer subjektiveren Sichtweise, 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?

Die verschiedenen Variationen zeigen, dass ein spezifisches Training auf ein Kriterium in den meisten Fällen die Leistung der nachzeichnenden KI in diesem Kriterium erhöht. Einige Variationen sind dabei effektiver als andere:

Die *Speed* Variation und dessen Kombination mit der *No-Penlift* Variation erreichen die tiefsten Werte im Kriterium der Geschwindigkeit um 9 Steps für das Nachzeichnen von Zahlen (siehe 4.1.1). Diese Variationen können somit die Leistung der KI in ihrem ausgewählten Kriterium verbessern.

Die *Rec* Variation erreicht einen beinahe identischen Wert im Kriterium der Erkennbarkeit wie die *Base* Variation und einige andere. Diese Variation konnte die Leistung somit nicht verbessern. Allerdings ist die Erkennbarkeit mit 97% für Zahlen der *Base* Variation auch kaum übertreffbar, vor allem weil die klassifizierende KI nur eine Genauigkeit von 98% erreicht.

Die *No-Penlift* Variation erreicht unter den Variationen die beste Leistung im Kriterium der zeichnenden Zeit (siehe 3.2.4) mit 98.1% für das Nachzeichnen von Zahlen. Die Variation erbringt somit den erwünschten Effekt.

Die *Overdraw* Variation erreicht unter den Variationen die tiefsten Werte im Kriterium der Übermalung (siehe 3.2.5) mit 40 übermalten Pixeln pro Zeichnung und kann somit ebenfalls die Leistung der KI nach dem gewünschten Kriterium verbesseren.

Die *Base* Variation erreicht die beste Leistung im Kriterium der prozentualen Übereinstimmung, obwohl die meisten Variationen eine sehr ähnliche Reward Function (siehe 3.3.1) verwenden. Das deutet darauf hin, dass in diesem Kriterium eine bessere Leistung erreicht wird, wenn die Reward Function ausschliesslich für dieses Kriterium trainiert.

Die *Physics* Variation verbessert die Leistung der nachzeichnenden KI nicht. Für jedes Kriterium existieren andere Variationen, die deutlich bessere Werte erreichen. Dieses Resultat deutet darauf hin, dass die KI mit den Physiksimulationen nicht gut umgehen kann. Möglicherweise liegt das Scheitern von diesem Ansatz daran, dass die Umgebung dadurch zu viele Faktoren hat, welche die Bewegungen der KI beeinflussen.

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

In allen Versionen bleibt die Leistung der KI zwischen den drei Datensets (siehe Tabelle 3.1) vergleichbar, wie die folgenden Tabellen (siehe Tabelle 5.1 und Tabelle 5.2) zeigen.

Tabelle 5.1: Vergleich der *Base* Variation für die drei Datensets

	Sim [%]	Rec [%]	Speed	Drawtime [%]	Overdrawn
MNIST	90.8	97.1	54.7	0.73	269
EMNIST	89.6	85.0	60.5	81.9	315
QuickDraw	81.8	93.7	56.5	73.9	227

Tabelle 5.2: Vergleich der *Speed* Variation für die drei Datensets

	Sim [%]	Rec [%]	Speed	Drawtime [%]	Overdrawn
MNIST	80.2	96.7	20.5	90.5	78
EMNIST	78.0	77.0	32.6	92.2	130
QuickDraw	74.5	85.7	29.5	89.3	93

Die Werte in den Tabellen stammen direkt aus den Resultaten (siehe 4) und zeigen, dass ein Vergleich der Leistung innerhalb einer Variation für die drei getesteten Datensets nur geringe Unterschiede ergibt. Allerdings ist die Leistung der KI für das Zeichnen von Buchstaben aus dem EMNIST Letters Datenset in den meisten Kriterien leicht schlechter.

Dieser Umstand könnte zum Grund haben, dass die Buchstaben aus dem EMNIST Letters Datenset häufig deutlich dicker gemalt sind als die anderen Motive (siehe Abbildung 5.1). Ausserdem erreicht die klassifizierende KI für Buchstaben eine niedrigere Genauigkeit als für Zahlen und

QuickDraw-Motive (siehe Tabelle 3.1), wodurch die Leistung im Kriterium der Erkennbarkeit stärker verfälscht wird als für die anderen Motive.

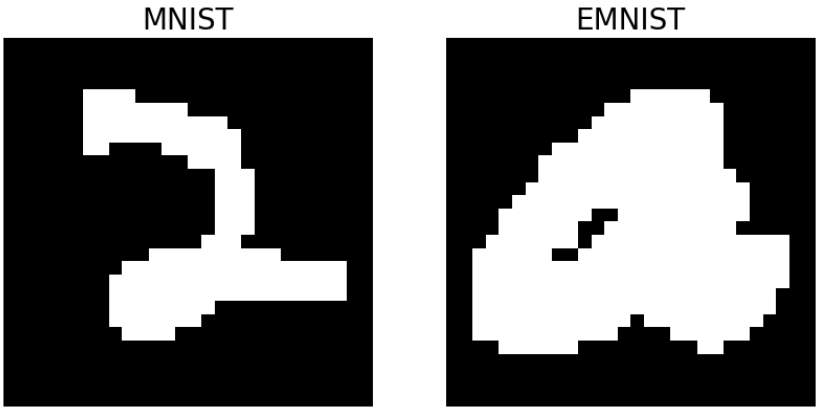

Abbildung 5.1: Vergleich zwischen dem MNIST Datenset und dem EMNIST Datenset (eigene Abbildung)

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

Die nachzeichnende KI bewegt sich mit beschränkter Geschwindigkeit und kann nur an dem Ort zeichnen, wo sie sich gerade befindet. Durch diese Einschränkungen in der Freiheit der Bewegungen wären diese in die physische Welt übersetzbar. Das heisst, dass ein zeichnender Roboter prinzipiell mit der nachzeichnenden KI bedienbar wäre.

Die *Physics* Variation stellt einen Versuch dar, menschliches Zeichnen weiter anzunähern. Dieser Versuch ist allerdings gescheitert, weil die KI dadurch allgemein schlechter zeichnet als die anderen Variationen. Die *Speed*, *Rec* und *No-Penlift* Variationen (siehe 4) nähern menschliches Zeichnen eher an, da diese Variationen den 'Schwung' der Bewegungen erhöhen. Schwung ist ein wichtiges Element für den menschlichen Prozess des Zeichnens. Die *Speed* Variation kommt insgesamt dem menschlichen Zeichnen am nächsten, da diese in allen Kriterien gute Werte erreicht und somit nicht nur mit 'Schwung' zeichnet, sondern auch mit Präzision.

5.1.6 Kann eine KI Strichbilder ohne Vorlage zeichnen?

Ja. Eine abgewandelte Form der nachzeichnenden KI kann Strichbilder von einem ausgewählten Motiv zeichnen, solange diese im Training mit dem gewähten Motiv trainiert hat. Beide Versionen der generativen KI erreichen dabei eine Erkennbarkeit des gewünschten Motivs von mehr als 90%. Die *Random-Noise* Variation erreicht insgesamt marginal bessere Werte.

Die Resultate sprechen somit dafür, dass die generative KI Motive ohne Vorlage zeichnen kann. Eine subjektivere Einschätzung der generierteren Bilder ergibt allerdings, dass die KI viele Motive nicht erkennbar zeichnet. Besonderes das Motiv: 'F' und das Motiv: 'Blume' sind nicht immer erkennbar, selbst wenn die klassifizierende KI diese so einschätzt. In dieser Fehleinschätzung liegt das Hauptproblem der generativen KI. Wenn die klassifizierende KI ein Bild falsch einschätzt, erhält die generative KI der Situation entsprechend falsche Rewards. Die klassifizierende KI für Zahlen erreicht die höchste Genauigkeit (siehe 3.1) und aus diesem Grund sind die Zahlen der generativen KI für menschliche Beurteiler am besten erkennbar.

Der Test auf das Zeichnen von verschiedenen Zahlen spricht dafür, dass die generative KI die meisten Motive nachzeichnen könnte, solange diese auf eine klassifizierenden KI mit einer hohen Zuverlässigkeit Zugriff hat.

Da die KI ohne eine 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 darin, 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. Das Lernen der künstlichen Intelligenz gleicht somit dem Lernen von Kindern. Auch Kinder erlernen in ihren ersten Schuljahren häufig das Schreiben von Zahlen und Buchstaben, indem sie diese mit einem Stift nachfahren. Erst nachdem sie in dieser Aufgabe geübt sind, beginnen die meisten Kinder, eigene Zeichen und Symbole zu Schreiben.

5.2 Fazit und Ausblick

Die Resultate und deren Interpretation in der Beantwortung der Unterfragen (siehe 5.1) deuten darauf hin, dass verschiedene Variationen der künstlichen Intelligenz das Nachzeichnen von beliebigen Strichbildern erlernt haben. 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, da viele Bilder diese nicht Erfüllen. In der Aufhebung von diesen Einschränkungen liegt Entwicklungspotenzial. Eine weitere Annahme liegt in der Definition des Nachzeichnens. Die KI erlernt das Nachzeichnen nach einer selbst bestimmten Definition. Die Kriterien dieser Definition (siehe 3.2) 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% erreicht, ist das Training nicht optimal. Ausserdem sind die Resultate im Kriterium der Erkennbarkeit durch die Imperfektion der klassifizierenden KI leicht verfälscht. Die Verlässlichkeit der klassifizierenden KI ist vor allem für die generative KI, da ihr Training stark davon abhängt.

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.

Die generative KI kann ohne eine Vorlage selbstständig Zeichnungen (bis jetzt vor allem Zahlen) produzieren und entwickelt so in gewissem Sinne eine eigene Handschrift. Die nachzeichnende KI ist ein entscheidender Vorläufer dieser KI, weil die generative KI zuerst das Nachzeichnen erlernen muss, bevor sie selbstständig zeichnen kann. So erlernt die KI das Zeichnen ohne Vorlage, indem sie zuerst aus verschiedenen Beispielen lernt. Zu einem gewissen Grad adapiert die KI die Handschrift dieser Beispiele. Da die KI allerdings mit tausenden Beispielen trainiert, kann die Adaption und Vermischung dieser auch als eine eigene Handschrift der KI angesehen werden.

Die generative KI kann hauptsächlich Symbole wie Zahlen zeichnen. Eine mögliche Erweiterung wäre, dass die KI das Schreiben von ganzen Wörtern erlernt, weil ansonsten noch nicht von einer tatsächlichen Handschrift gesprochen werden kann.

5.3 Anwendungsbereiche

Für die nachzeichnende KI und die generative KI sind nachfolgend einige Anwendungsbereiche beschrieben. Es handelt sich dabei um Einsatzorte, bei denen diese künstlichen Intelligenzen einen Nutzen bieten können.

Die Nachzeichnende KI könnte in der Robotik Anwendung finden. Die aktuelle Version ist für diese Anwendung allerdings zu eingeschränkt, was die Art der Bilder angeht, die sie zeichnen kann. Sollte eine weiterentwickelte Version vielfältigere Bilder produzieren können, so würde ein Roboter mit der nachzeichnenden KI von beliebigen Bildern Zeichnungen anlegen können.

Ein weiterer Anwendungsbereich der nachzeichnenden KI wäre die Vektorisierung. Damit ist die Umwandlung von Rastergrafiken in Vektorgrafiken gemeint (siehe Abbildung 5.2). Diese Konvertierung benötigt

einen Nachzeichenprozess. 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.

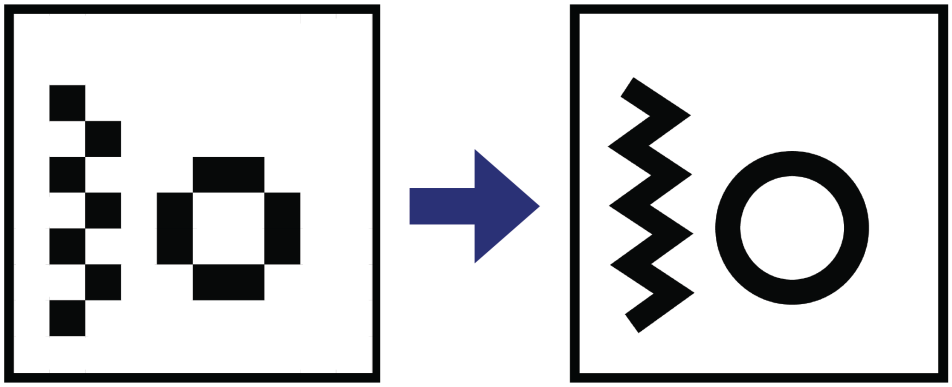

Abbildung 5.2: Umwandlung von Rastergrafiken in Vektorgrafiken (eigene Abbildung)

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 emulieren. Inwiefern die Zeichnungen der KI tatsächlich der Handschrift einer Person entsprechen, ist allerdings nur subjektiv bestimmbar.

Kapitel 6

Zusammenfassung

Diese Arbeit untersucht, ob eine KI Vorlagen von Strichbildern wie Zahlen, Buchstaben und andere einfache Zeichnungen so nachzeichnen kann, dass ein zeichnender Roboter prinzipiell durch die KI bedienbar wäre. Eine weiterführende Frage untersucht die Möglichkeiten einer generativen künstliche Intelligenz, die selbstständig, ohne eine Vorlage, Strichbilder zeichnen kann.

Um die Fragestellungen zu beantworten, werden die entsprechenden künstlichen Intelligenzen unter der Verwendung von Deep Q-Learning und einem Convolutional Neural Network (CNN) in Python mit der Keras API entwickelt. Die nachzeichnende KI lernt in einer Umgebung, wo sie sich wie ein virtueller Stift auf einer Zeichenfläche bewegt und dabei Bilder von handgeschriebenen Ziffern aus dem MNIST Datenset nachzeichnet. Die KI lernt aus Bewertungen ihrer Stiftbewegungen. Die Bewertungen basieren auf fünf quantitativen Kriterien. Beispiele für die Kriterien sind die prozentuale Übereinstimmung zwischen der Vorlage und dem nachgezeichneten Bild, die Geschwindigkeit und die Erkennbarkeit, die mit einer weiteren klassifizierenden künstlichen Intelligenz bestimmt wird. Die Werte in diesen Kriterien machen die Ergebnisse dieser Arbeit aus. Die generative KI erlernt zu Beginn des Trainings mit entsprechenden Vorlagen das Nachzeichnen des gewünschten Strichbildes. Erst im späteren Verlauf des Trainings wird der KI immer seltener eine Vorlage gezeigt, bis diese vollkommen selbstständig zeichnet. Wenn die KI ohne eine Vorlage zeichnet, erfährt diese durch die Einschätzung eines klassifizierenden Machine Learning Modells, ob die eigene Zeichnung erkennbar ist.

Die nachzeichnende KI erreicht eine Übereinstimmung von 90% zwischen der Vorlage und dem nachgezeichneten Bild und stellt die Zeichnungen mit durchschnittlich 20 Bewegungen fertig. Die KI erzielt dabei eine vergleichbare Leistung für verschiedene Typen von Strichbildern, obwohl diese nur auf das Nachzeichnen von Zahlen trainiert ist. Die generative KI zeichnet das

gewünschte Strichbild in 90 bis 100 Prozent der Fälle so nach, dass die Zeichnung für eine passende klassifizierende KI erkennbar ist.

Die Ergebnisse sprechen dafür, dass die nachzeichnende KI erlernt, Strichbilder nach den vordefinierten Kriterien nachzuzeichnen. Die Strichbilder müssen allerdings von einem bestimmten Format sein und für andere Kriterien des Zeichnens erlaubt die KI keine Aussage. Die Bewegungen der KI sind ausserdem simuliert und somit nicht physisch. Trotzdem ist durch die beschränkte Freiheit der Bewegungen ein zeichnender Roboter durch sie bedienbar. Die Ergebnisse der generativen KI zeigen, dass eine künstliche Intelligenz auch selbstständig zeichnen kann, sofern diese im Training Beispiele des gewünschten Motivs gesehen hat und das Nachzeichnen von diesen Motiven bereits erlernt hat. Die Leistung der generativen KI ist allerdings direkt abhängig von der Zuverlässigkeit der klassifizierenden KI. Falsche Einschätzungen der Zeichnungen beeinflussen die generative KI negativ, selbst wenn diese selten vorkommen.

Die nachzeichnende KI kann beliebige Strichbilder nachzeichnen. Diese Strichbilder sind allerdings klein, schwarzweiss und begrenzt detailliert. Mit einem vielseitigeren Format könnte die KI für verschiedene Anwendungen, die einen Nachzeichenprozess benötigen, verwendet werden. Dazu gehört unter anderem die Umwandlung von Rastergrafiken in Vektorgrafiken. Die generative KI zeichnet aus eigenem Antrieb und entwickelt so gewissermassen eine eigene Handschrift. Mit dem Training auf Schriftstücke einer ausgewählten Person wäre diese KI durchaus dazu in der Lage, die Handschrift dieser Person nachzuahmen. Schlussendlich beschreibt diese Arbeit zwei künstliche Intelligenzen mit grossem Entwicklungspotenzial.

Literatur

Agarap, A. F. (2019, 7. Februar). Deep Learning using Rectified Linear Units (ReLU). https://doi.org/10.48550/arXiv.1803.08375

Agnihotri, A., & Batra, N. (2020). Exploring bayesian optimization. *Distill*, *5*(5), e26. https://doi.org/10.23915/distill.00026

Arora, S. (2020, 29. Januar). *Supervised vs unsupervised vs reinforcement* [AITUDE]. Verfügbar 25. Juni 2022 unter https://www.aitude.com/supervised-vs-unsupervised-vs-reinforcement/

Black Box (Systemtheorie) [Page Version ID: 215860839]. (2021, 24. September). In *Wikipedia*. Verfügbar 2. Oktober 2022 unter https://de.wikipedia.org/w/index.php?title=Black_Box_(Systemtheorie)&oldid=215860839

Coggan, M. (2004). Exploration and exploitation in reinforcement learning.

DataFlair. (2020, 18. September). *Handwritten character recognition with neural network* [DataFlair]. Verfügbar 21. März 2023 unter https://data-flair.training/blogs/handwritten-character-recognition-neural-network/

Garnett, R. (2022). *Bayesian optimization book* [in preparation]. Cambridge University Press. Verfügbar 2. Oktober 2022 unter https://bayesoptbook.com//

Hertzmann, A. (2002, 15. April). *Stroke-Based Rendering*. Verfügbar 16. September 2022 unter https://www.cs.ucdavis.edu/~ma/SIGGRAPH02/course23/notes/S02c23_3.pdf

Huang, Z., Zhou, S., & Heng, W. (2019). Learning to paint with model-based deep reinforcement learning. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 8708–8717. https://doi.org/10.1109/ICCV.2019.00880

Jayawardana, K. (2021, 3. Januar). *Concatenating multiple activation functions and multiple poling layers for deep neural networks* [Medium]. Verfügbar 16. September 2022 unter

https://towardsdatascience.com/concatenating-multiple-activation-functions-and-multiple-poling-layers-for-deep-neural-networks-d48a4b273d30

Keras: the Python deep learning API. (2015). Verfügbar 9. Oktober 2022 unter https://keras.io/

Koech, K. E. (2021, 18. November). *Softmax activation function — how it actually works* [Medium]. Verfügbar 21. März 2023 unter https://towardsdatascience.com/softmax-activation-function-how-it-actually-works-d292d335bd78

Kranz, J.-D. (2019, 3. April). *Deep Learning vs Machine Learning - Was ist der Unterschied?* [IT-Talents.de]. Verfügbar 18. Juni 2022 unter https://it-talents.de/it-wissen/programmieren/deep-learning-vs-machine-learning-was-ist-der-unterschied/

Kumar, N. (2019, 18. Dezember). *Sigmoid neuron — deep neural networks* [Medium]. Verfügbar 16. September 2022 unter https://towardsdatascience.com/sigmoid-neuron-deep-neural-networks-a4cd35b629d7

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition [Conference Name: Proceedings of the IEEE]. *Proceedings of the IEEE*, *86*(11), 2278–2324. https://doi.org/10.1109/5.726791

LeCun, Y., Cortes, C., & Burges, C. J. (1998). MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges. Verfügbar 16. September 2022 unter http://yann.lecun.com/exdb/mnist/

Liu, Q., & Wu, Y. (2012). Supervised Learning. https://doi.org/10.1007/978-1-4419-1428-6_451

Malik, F. (2019, 20. Mai). *What are hidden layers?* [FinTechExplained]. Verfügbar 16. September 2022 unter https://medium.com/fintechexplained/what-are-hidden-layers-4f54f7328263

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013, 19. Dezember). Playing Atari with Deep Reinforcement Learning. https://doi.org/10.48550/arXiv.1312.5602

Moriconi, R., Deisenroth, M. P., & Kumar, K. S. S. (2020, 25. September). High-dimensional Bayesian optimization using low-dimensional feature spaces. https://doi.org/10.48550/arXiv.1902.10675

Nagyfi, R. (2018, 4. September). *The differences between artificial and biological neural networks* [Medium]. Verfügbar 21. März 2023 unter https://towardsdatascience.com/the-differences-between-artificial-and-biological-neural-networks-a8b46db828b7

Neuronales Netz [Page Version ID: 210963657]. (2021, 15. April). In *Wikipedia*. Verfügbar 16. September 2022 unter https://de.wikipedia.org/w/index.php?title=Neuronales_Netz&oldid=210963657

Nielsen, M. A. (2015). Neural networks and deep learning [Publisher: Determination Press]. Verfügbar 22. April 2022 unter http://neuralnetworksanddeeplearning.com

Nogueira, F. (2014). *Bayesian Optimization: Open source constrained global optimization tool for Python* [original-date: 2014-06-06T08:18:56Z]. Verfügbar 25. Juli 2022 unter https://github.com/fmfn/BayesianOptimization

Nyuytiymbiy, K. (2022). Parameters and hyperparameters in machine learning and deep learning. *Medium*. Verfügbar 16. September 2022 unter https://towardsdatascience.com/parameters-and-hyperparameters-aa609601a9ac

O'Shea, K., & Nash, R. (2015, 2. Dezember). An Introduction to Convolutional Neural Networks. https://doi.org/10.48550/arXiv.1511.08458

Oxford english dictionary. (n. d.). Verfügbar 20. März 2023 unter https://www.oed.com/

Pragati, B. (2022, 19. Juli). *Activation functions in neural networks [12 types & use cases]*. Verfügbar 16. September 2022 unter https://www.v7labs.com/blog/neural-networks-activation-functions,%20https://www.v7labs.com/blog/neural-networks-activation-functions

Pramoditha, R. (2021, 29. Dezember). *The concept of artificial neurons (perceptrons) in neural networks* [Medium]. Verfügbar 1. Oktober 2022 unter https://towardsdatascience.com/the-concept-of-artificial-neurons-perceptrons-in-neural-networks-fab22249cbfc

Quick, Draw! Image Recognition [original-date: 2019-04-30T10:10:02Z]. (2022, 28. August). Verfügbar 28. August 2022 unter https://github.com/Lexie88rus/quick-draw-image-recognition

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1985). Learning Internal Representations by error propagation. Verfügbar 16. September 2022 unter https://apps.dtic.mil/dtic/tr/fulltext/u2/a164453.pdf

Sadie, B. (2019, 10. Juni). *Why machine learning is primarily written in Python* [IBM Developer]. Verfügbar 2. Oktober 2022 unter https://developer.ibm.com/blogs/why-machine-learning-is-primarily-written-in-python/

Serrano, L. (2021, 14. Dezember). *2.1 what is the difference between labelled and unlabelled data? · grokking machine learning*. Verfügbar 16. September 2022 unter https://livebook.manning.com/book/grokking-machine-learning/2-1-what-is-the-difference-between-labelled-and-unlabelled-data-/v-4/

Spaulding, N. W. (2020, 9. Juli). Is Human Judgment Necessary? Artificial Intelligence, Algorithmic Governance, and the Law. In M. D. Dubber, F. Pasquale & S. Das (Hrsg.), *The Oxford Handbook of Ethics of AI*. Oxford University Press. https://doi.org/10.1093/oxfordhb/9780190067397.013.25

Sutton, R. (1997). *2.3 Softmax Action Selection*. Verfügbar 4. März 2023 unter http://incompleteideas.net/book/2/node4.html

Sutton, R. S., & Barto, A. G. (2014). Reinforcement learning: An introduction. *MIT Press*, 352.

TensorFlow. (2015). Verfügbar 2. Oktober 2022 unter https://www.tensorflow.org/

Trahasch, Tobias Hagen, Tobias Lauer, Volker Sänger, Stephan & Klaus Dorer. (2020, 8. August). *3.1 Einführung — Menschen Lernen Maschinelles Lernen - ML2*. Verfügbar 16. September 2022 unter https://imla.gitlab.io/ml-buch/ml2-buch/3-1-lr-einfuehrung.html

Unzueta, D. (2022, 15. März). *Convolutional layers vs fully connected layers* [Medium]. Verfügbar 16. September 2022 unter https://towardsdatascience.com/convolutional-layers-vs-fully-connected-layers-364f05ab460b

van Heeswijk, W. (2021, 30. August). *How to model experience replay, batch learning and target networks* [Medium]. Verfügbar 16. September 2022 unter https://towardsdatascience.com/how-to-model-experience-replay-batch-learning-and-target-networks-c1350db93172

Verma, P., & Diamantidis, S. (2021, 27. April). *What is Reinforcement Learning? – Overview of How it Works — Synopsys*. Verfügbar 16. September 2022 unter https://www.synopsys.com/ai/what-is-reinforcement-learning.html

Zheng, N., Jiang, Y., & Huang, D. (2018). StrokeNet: A neural painting environment. Verfügbar 31. März 2022 unter https://openreview.net/forum?id=HJxwDiActX

Zhou, T., Fang, C., Wang, Z., Yang, J., Kim, B., Chen, Z., Brandt, J., & Terzopoulos, D. (2018). Learning to Doodle with Deep Q Networks and Demonstrated Strokes. *arXiv:1810.05977 [cs]*. Verfügbar 31. März 2022 unter http://arxiv.org/abs/1810.05977

Abbildungsverzeichnis

Tabellenverzeichnis