WWU MÜNSTER

Wirtschafts-
wissenschaftliche
Fakultät

Max Lütkemeyer, Leon Cena, Robin Killewald

Wie Computer Spiele spielen

Wissenschaftlich begleitetes Praktikum

im Rahmen des „InformatiCup2021“

am Lehrstuhl für Informatik
(Westfälische Wilhelms-Universität Münster)

Themenersteller:	Dr. Jens Lechtenbörger
Betreuer:	Dr. Jens Lechtenbörger
vorgelegt von:	Max Lütkemeyer, Leon Cena, Robin Killewald
Abgabetermin:	14. Juni 2021

Inhaltsverzeichnis

1 Einleitung 1
2 Geschichte 1
3 Theoretischer Hintergrund 2
4 Status Quo 5
5 Ausblick 8
Literaturverzeichnis 9

1 Einleitung

In diesem Aufsatz beschäftigen wir uns mit der Art und Weise, wie Computer Spiele spielen. Im Rahmen der Anfertigung der theoretischen Ausarbeitung unseres Projekts für den InformatiCup sind wir auf dieses umfangreiche Thema gestoßen. Dabei fordern die Komplexität und schnelle Entwicklung in diesem Bereich, dass ein Überblick über die Grundlagen und den Verlauf der Entwicklung vorhanden ist. Eine naheliegende und aktuelle Herangehensweise ist das Arbeiten mit künstlicher Intelligenz zur Lösung von Problemen. Auch hier muss der rasante technische Fortschritt beachtet werden.

Wir beantworten die Fragen, wie Computer Spiele gespielt haben, wie sie sie aktuell spielen und in der Zukunft möglicherweise spielen werden. Nach Bearbeitung dieser Leitgedanken soll am Ende ein Überblick über die Thematik vermittelt sein. Dafür wird zunächst die Geschichte näher thematisiert. Historische Fortschritte zeigen eine Entwicklung auf, der wir bis zum Status Quo hin folgen. Anschließend erörtern wir theoretische Hintergründe, die für die Thematik, vor allem für das aktuelle Geschehen, bedeutend sind. Zentral dabei ist der Begriff der Intelligenz und das (maschinelle) Lernen. Es folgt eine Diskussion über künstliche Intelligenz (KI) und deren Klassifizierung. Nachdem wir näher auf die historischen Entwicklungen und theoretischen Hintergründe eingegangen sind, werden wir den Status Quo vorstellen. Dabei gehen wir auf aktuelle Ansätze und technische Realisierungen ein. Abschließend geben wir einen Ausblick über interessante Möglichkeiten und Herausforderungen zukünftiger Ansätze.

2 Geschichte

Die Geschichte der Computerspiele geht mit Versuchen einher, Computer als Gegner gegen den Menschen antreten zu lassen oder sie selbst spielen zu lassen. Bereits bei den ersten Computerspielen wie Pong waren Entwickler versucht, mithilfe von Algorithmen einen Gegner zu schaffen. Als ein weiteres frühes Beispiel dient das Spiel „Nim“. In einem Artikel im New Yorker heißt es dazu: „Nim is a game for 2 people. In this case one of them is the machine.“ (Eugene F. Grant 1952).

Von steigender Popularität der Computerspiele und immer wieder verbesserten Technologien profitieren bis heute auch die Ansätze, Rechner selbst spielen zu lassen. Computer können dabei selber spielen oder das Spiel um Nicht-Spieler-Charaktere, kurz NPC, erweitern. Jedoch ist fraglich, ab welcher Stufe diese NPC so maßgeblich am Spiel teilnehmen, dass der Computer durch sie das Spiel auch spielt und nicht nur am Rand ergänzt. Gegner NPC, wie in der Shooter Branche geläufig (siehe Rivera et al. 2012), spielen oft

vergleichbar, wie es ein Nutzer tut. Unabhängig von der Branche wurden über die vergangene Zeit viele Ansätze genutzt, verbessert und erweitert.

Traditionell konnten die logischen Probleme und Aufgaben mit algorithmischen Anweisungen Maschinen beigebracht werden. Ein Rechner kann TicTacToe optimal spielen, wenn er dabei auf entsprechenden Verhaltensregeln aufbaut. Die Intelligenz hinter den Spielzügen stammt jedoch vom Entwickler. Auf der Spiellogik aufbauende, intelligent wirkende, nicht lernende Verhaltensweisen werden unter dem Begriff symbolische künstliche Intelligenz zusammengefasst. Manche Probleme und vorzugsweise aktuelle Spiele zeigen diesen Ansätzen mit ihrer Komplexität Grenzen auf. Smolensky beschreibt, wie die Nachbildungen menschlicher Intelligenz oder des menschlichen Gehirns aus logischen Regeln, keine enormen Erfolge feiern: „In AI, the trouble with the ‚Boolean dream' is that symbolic rules and the logic used to manipulate them tend to produce rigid and brittle systems." (Smolensky 1987, S. 99). Konträr dazu bieten hier Ansätze des Machine Learning zuvor ungenutzte Möglichkeiten. Diese Ansätze sollen weitergehend betrachtet und diskutiert werden.

3 Theoretischer Hintergrund

Wenn wir über künstliche Intelligenz sprechen, wollen wir zunächst Intelligenz allgemein sprechen. Für diesen gibt es keine einheitliche Definition. In der Forschung werden die Grundlagen, die eine Intelligenz ausmachen, diskutiert (siehe Davis 1996, S. 92-95). Dabei sind unter Menschen logisches Denkvermögen, analytische oder kreative Fähigkeiten, sowie Empathie und Weiteres relevant. Es werden Ansätze zu Tests über die menschliche Intelligenz genutzt, die auf bestimmte der angesprochenen Aspekte stärker und auf andere weniger stark eingehen (vgl. Guttman und Levy 1991, S. 81-82).

Die Definition von künstlicher Intelligenz ist ebenfalls problematisch, vielleicht weil sie den besprochenen Begriff Intelligenz selbst beinhaltet (vgl. Konar 2000, S. 26). Bei Bemessungen der Intelligenz, speziell beim Spielen von Computerspielen, müssen jedoch nicht derart viele konkurrierende Fähigkeiten betrachtet werden. Es soll uns hier nicht um allgemeine Intelligenz, sondern um welche in einer bestimmten Anwendungsdomäne, dem Spiel, gehen. Als beispielhaftes Kriterium für Intelligenz kann Wissenstransfer angesetzt werden. Demnach ist eine KI intelligent, die Gelerntes auf neue, nicht bekannte Situationen übertragen kann. In späteren Beispielen wird diese Fähigkeit weiter thematisiert. Konar beschreibt unter dem Begriff „Soft Computing" (siehe Konar 2000, S. 38-41) verschiedene Ansätze dazu und zitiert Prof. Zadeh, nachdem der Begriff einen aufstrebenden Ansatz für die Datenverarbeitung, der der bemerkenswerten Fähigkeit des mensch-

lichen Geistes entspricht, in einer Umgebung von Unsicherheit und Ungenauigkeit zu denken und zu lernen, umfasst (vgl. Zadeh 1994). Die Güte von KIs lässt sich anhand von Tests vergleichen. Anwendungsbezogene Tests haben die stärkste Aussagekraft bei der Beurteilung und im Vergleich (vgl. Li et al. 2018, Kapitel 2.4). Diese Tests sind jedoch dadurch zumeist nicht universal auf verschiedene Spiele anwendbar.

Als Beispiel dienen hier die Tests zur Bewertung von Intelligenz im OpenAI Hide and Seek Projekt. Beim Verstecken und Suchen befinden sich Agenten in verschiedenen Umgebungen mit Räumen und bewegbaren Objekten. Das maschinelle Lernen ermöglicht ihnen, ihr Umfeld zielgerichtet zu nutzen. Die Tests beziehen sich dabei konkret auf die Anwendung, indem sie in Tests zur Erkennung und Erinnerung, sowie Tests zur Manipulation der Umgebung geteilt werden und entsprechende Aufgaben und Ansprüche formulieren (vgl. Baker et al. 2020, Kapitel 6.2).

Maschinelles Lernen

Maschinelles Lernen (ML) ist ein weit genutzter Ansatz in der künstlichen Intelligenz (vgl. Cai et al. 2020, S. 8). Anhand von Eingabe- und Ergebnisdaten wird versucht ein Transformationsgesetz für Eingaben abzuleiten (siehe Abbildung 1). Diese Daten nennt man auch *training data*. Mit diesem Modell kann ein ML-Agent Entscheidungen treffen, die der Entwickler nicht explizit programmiert hat. Er lernt somit abhängig von der Situation eigenständig passende Entscheidungen zu treffen und agiert ohne vorgeschriebene Verhaltensregeln (siehe Cai et al. 2020, S. 7). In Abbildung 1 wird gezeigt, dass in klassischer Programmierung Daten sowie Regeln zu Resultaten verarbeitet werden, wohingegen bei maschinellem Lernen Daten und Resultate in Regeln resultieren, mit denen Entscheidungen getroffen werden können.

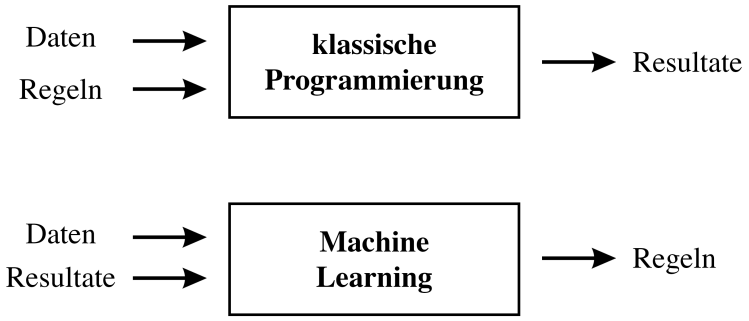

Abbildung 1 Unterscheidung von klassischer Programmierung und maschinelles Lernen (Cai et al. 2020, S. 7)

Ein ML-Agent kann mehrere Lernmethoden verfolgen. Wichtig für Spiele sind Supervised Learning (siehe Goyal et al. 2017) und Reinforcement Learning (siehe Heinrich und Silver 2016) (für Vergleich und Beispiele siehe Kachalsky et al. 2017). Im Falle von Supervised Learning stehen dem Modell beispielhafte optimale Datensätze als Trainingsdaten zur Verfügung. Anhand dieser Informationen bestimmt der Algorithmus eine generalisierte Entscheidungsregel (siehe Cai et al. 2020, S. 9). Einen wiederum unterschiedlichen Weg verfolgt der Ansatz des Reinforcement Learning, da oft optimale Datensätze nicht zur Verfügung stehen. Hierbei interagiert der ML-Agent mit einer Umgebung und lernt mit einer Nutzenfunktion, die es zu maximieren gilt. Der Nutzen dient dabei als Feedback für Entscheidungen und das Lernen hängt von dem Ergebnis der Nutzenfunktion ab (vgl. Cai et al. 2020, S. 372). Dabei findet ein stetiger Wechsel zwischen Anwenden des Gelernten und neuer Erkundung der Umgebung statt (vgl. Cai et al. 2020, S. 380).

Bei ML wird die Transformation der Eingangsdaten häufig mit künstlichen neuronalen Netzwerken (KNN) erledigt (siehe Cai et al. 2020, S. 3). Diese basieren in ihrer grundlegenden Idee auf dem biologischen Gehirn und stellen eine Sammlung von künstlichen Neuronen dar, die als Knoten im Graphen repräsentiert werden. In Abbildung 2 wird die beispielhafte Struktur eines neuronalen Netzwerks mit mehreren Schichten illustriert, wobei Neuronen als Knoten und Verbindungen als Kanten repräsentiert werden.

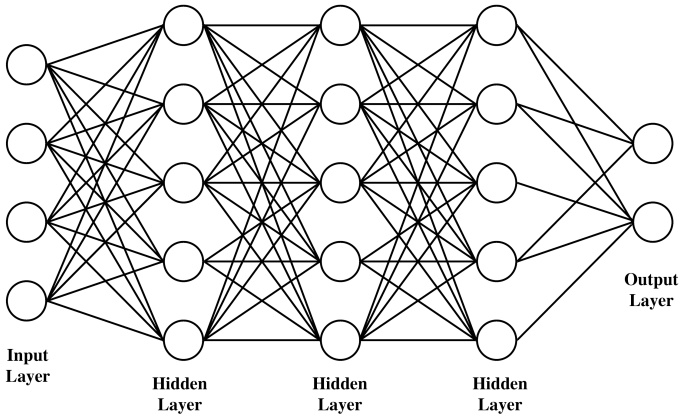

Abbildung 2 Vereinfachtes künstliches neuronales Netz (ohne Gewichtungen)

Perzeptron

Eine einfache Form eines KNN ist ein Perzeptron. Es besteht nur aus einem Neuron, genauer einer linear threshold unit (LTU), und bildet einen Eingabevektor binär ab. Es wird eine gewichtete Summe z abhängig von dem Eingabevektor x und einem Gewichtungsvektor w gebildet (Gleichung 3.1) und anschließend eine Stufenfunktion $h(z)$ (hier:

heaviside Gleichung 3.2) angewandt. Bei künstlichen Neuronen spricht man dann von Aktivierungsfunktionen (vgl. Géron 2019, S.284 f.).

$$z = x^T w \tag{3.1}$$

$$h(z) = \begin{cases} 0 & z < 0 \\ 1 & z \geq 0 \end{cases} \tag{3.2}$$

Damit ein Perzeptron wertvolle Ausgaben macht, muss es trainiert werden. Ein Perzeptron wird in einer Art der Hebbschen Lernregel trainiert. Verbindungen zwischen Perzeptronen, die einen Fehler verringern, werden nach einer Iteration höher gewichtet. Durch einen Vergleich mit Soll-Daten und einer Vorhersage mit dem Model, können diese gefunden werden (Gleichung 3.3). Die Differenz des Soll-Ergebnisses y_j und des Ergebnisses $\hat{y}_j$ multipliziert mit der Lernrate η und dem Eingangswert x_i ergibt die Gewichtsänderung (vgl. Géron 2019, S. 287).

$$w_{i,j}^{(\text{nächster Schritt})} = w_{i,j} + \eta(y_j - \hat{y}_j)x_i \tag{3.3}$$

Da man mit einem Perzeptron eigentlich nur lineare binäre Klassifikation gut erledigen kann, bietet es sich an Perzeptronen in Schichten zu organisieren. So können multivariate Merkmale verarbeitet werden. Es gibt verschiedene Vorgehensweisen die einzelnen Schichten zu verbinden. Die einfachste Variante ist alle mit allen zu verbinden, einem sogenannten Dense Layer (vgl. Géron 2019, S.285). Heutige Modelle sind deutlich komplexer als einfache Perzeptronen. Perzeptronen können z.B. kein XOR repräsentieren. Um schwierigere Probleme zu lösen muss man Perzeptronen stapeln. Man spricht dann von einem Mehrschichtigem Perzeptron (vgl. Géron 2019, S. 288). Das ist das vereinfachte grundlegende Modell auf der die folgenden aktuellen Beispiele aufbauen.

4 Status Quo

Populär im Bereich der künstlichen Intelligenz im Zusammenhang mit Spielen ist die Organisation OpenAI LP. Mit ihrem OpenAI Gym stellen sie die bekanntesten Umgebungen zum Testen von verschiedenen Modellen zur Verfügung. Modelle innerhalb von Simulationen zu Trainieren, die nicht zwingend in Echtzeit ablaufen müssen, ist sehr viel effizienter, da so der Faktor Zeit beschleunigt werden kann. Die Komplexität einer Simulation kann so jedoch weiter steigen.

Zauberwürfel

In (OpenAI et al. 2019) wird thematisiert, wie eine humanoide Roboter-Hand einen Zauberwürfel löst. Das Ziel ist eine KI für allgemeine Zwecke zu entwickeln. Die Hardware,

also die Hand, war nicht neu, nur die Software. Das Modell sollte die Fähigkeit haben, mit Situationen zurecht zu kommen, die es vorher während des Trainings nie erlebt hat (vgl. OpenAI et al. 2019, Kapitel 1). Da es sehr Aufwendig wäre, jedes mal wenn der Würfel runter gefallen ist, ihn zurück auf die Hand zu legen, bietet es sich hier gut an die Umgebung zu Simulieren. Durch Simulation kann auch hier der Lernprozess beschleunigt werden. Doch die Herausforderung bei einer Simulation ist, dass sie umfangreich genug für die Anwendung in der realen Welt sein muss.

OpenAI hat hierfür den Automatic Domain Randomization (ADR) Algorithmus entwickelt. Dieser generiert zahlreiche verschiedene Umgebungen, in denen die KI dann trainiert. Folgen davon sind, dass keine perfekte physikalische Simulation erstellt werden muss oder dass für Umgebungsparameter kein Wissen zur Anwendungsdomäne mehr gebraucht wird. So kann man das trainierte Wissen aus den Simulationen mit einem deutlich größerem Erfolg in die reale Welt übertragen (siehe OpenAI et al. 2019, Kapitel 5). Es hat sich herausgestellt, dass sogar viele verschiedene Varianten, die nie trainiert wurden (z.B. eine behinderte Hand), zu Erfolgen führten, nur langsamer (vgl. OpenAI et al. 2019, Kapitel 8.4). Da sich das Lernverhalten immer mehr dem menschlichen annähert, sind auch diese Effekte vergleichbar am Menschen beobachtbar. In einem neuen Supermarkt können wir Einkaufen, auch wenn wir ihn das erste Mal betreten. Dafür wird jedoch oftmals mehr Zeit benötigt.

AlphaGo Zero

2017 stellte Alphabets Tochtergesellschaft DeepMind ihre Künstliche Intelligenz AlphaGo Zero vor. Wie auch beim Vorgänger AlphaGo war das Ziel der Entwicklung, eine Maschine zum besten Go (für Regeln siehe Benson 1976, Abstract) Spieler der Welt zu krönen (vgl. Silver et al. 2017, Methods). Genutzt wird ein neuronales Netzwerk, das über Reinforcement Learning, beginnend mit zufälligen Gewichtungen von Entscheidungen, hin zu einem intelligenten Verhalten arbeitet. Eingaben für das neuronale Netz beinhalten den aktiven Spieler und wie die einzelnen Felder belegt sind und über die letzten 7 Spielzüge belegt waren. Das ist wichtig, da Go Wiederholungen in Spielzügen verbietet und dadurch auf das bereits Geschehende geachtet werden muss. AlphaGo Zeros Entscheidungen bauen somit auf Informationen, wie sie sich ein menschlicher Gegner ebenfalls merken könnte.

Vor der Einordnung des Erfolgs des Projekts ist vorweg, neben den Vergleichen zu Vorgängern, besonders interessant, einen direkten Vergleich zum überwachten Lernen zu fassen. Zwar kann einfach gezeigt werden, dass die KI gegen AlphaGo, die ihre Intelligenz

aus tausenden professionellen Spielen nimmt, gewinnt, jedoch unterscheidet sich auch die Struktur der Netzwerke. Genau zu diesen Zwecken der Vergleichbarkeit wurde ein zusätzliches Netz mit identischer Struktur menschlichen Zügen unterwiesen. Asymptotisch verliert ein überwachter Ansatz deutlich gegen Reinforcement Learning. Hier kann die Software gegen sich selbst antreten und ihr eigener Lehrer sein (siehe Bai und Jin 2020). Spiele müssen nicht in Echtzeit stattfinden und so wurden über 40 Tage rund 29 Millionen Partien gespielt. Nach drei Tagen war die KI stärker als ihr Vorgänger AlphaGo. Nach 40 Tagen überholte sie die bis dato stärkste weitere KI AlphaGo Master.

OpenAI Five

Das Projekt (Berner et al. 2019) hat sich damit beschäftigt, eine KI für Dota 2 zu entwickeln. Besonders interessant ist, dass das Problem eigentlich aus dem Rahmen fällt, für den Reinforcement Learning zuvor genutzt wurde. Die Projektverantwortlichen erhofften sich, getrieben durch Innovationsdrang, dass sich die Qualität ihrer Intelligenz so verbessern würde. Es stellte sich heraus, dass keine grundlegende Veränderung an Umgebung oder Restriktionen der Komplexität nötig waren. Lediglich der Rechenumfang musste auf weitere Hardware entlastet werden (vgl. Berner et al. 2019, Kapitel 1).

Bei einem Spiel wie Dota 2 gibt es in regelmäßigen Abständen Gamepatches. Diese verändern das Spiel mal weniger, mal mehr, sodass das Risiko bestand, dass das Modell nach einem Patch unbrauchbar würde. Viele Teile des Models sind nach einem Patch noch sinnvoll, aber einzelne Parameter sind schwierig zu reproduzieren, weswegen es einfacher ist, das Modell dann neu zu trainieren. Solche Änderungen sind ein großes Problem für viele andere Anwendungsdomänen. Jedoch konnte das Team das gelernte Wissen transferieren und auf neue Situationen anwenden. Dafür haben sie sogenannte „surgeries“ durchgeführt, die aus dem alten Model π_0 ein neues kompatibles Modell $\hat{\pi}_0$ erstellten (siehe Berner et al. 2019, Kapitel 3.3). Mussten z.B. Teile des Modells entfernt werden, wurden sie durch Konstanten ersetzt (vgl. Berner et al. 2019, S. 29). Durch ihre optimierte Policy wurde der entwickelte Bot immer und immer besser. Am Ende hatte das Team einen leistungsfähigen Bot trainiert, welcher gegen das derzeit weltbeste Team (Team OG) gewann (vgl. Berner et al. 2019, Abstract).

spe_ed

Nachdem verschiedene Ansatzweisen und aktuelle Projekte aufgezeigt wurden, wollen wir Inhalte im Bezug auf den Informaticup reflektieren. Wie auch bei der Kernidee für

diese Arbeit stand dabei das Spielen durch Computer im Fokus des Wettbewerbs. Die einzelnen Etappen und Stufen, in denen ein Computer am Spiel teilhaben kann, wurden besonders bei der schrittweisen Implementierung deutlich. Angefangen bei einfachen, regelbasierten Algorithmen, die zum Beispiel Hindernissen ausweichen, haben wir uns zu Ansätzen vorgearbeitet, in denen Entscheidungen auf komplexeren Hintergründen bauen. Es wurden unter anderem Algorithmen genutzt um optimale Wege zu festen Zielen zu folgen oder in die Zukunft prognostiziert um in der Gegenwart aktuelle Entscheidungen treffen zu können. Wir haben diese Entwicklung bis zu einem Punkt verfolgt, an dem der Computer autonom und intelligent handelt. Er kann selbstständig seine Taktiken wechseln und viele Spielzüge in die Zukunft planen. Bei unseren Implementierungen außerhalb der Versuche mit neuronalen Netzen wirken die Entscheidungen (oft) intelligent, da sie auf Berechnungen beruhen, die ein Mensch in vergleichbarer Zeit quantitativ nicht schafft. Letztlich konnten wir keine KI mit neuronalen Netzen trainieren, die deutlich besser spielt. Neben Faktoren wie guten Eingabeparametern, Rechenleistung, Güte des Netzes und Zeit zum Trainieren zählt auch die Anwendung selbst mit zu den Faktoren, die maschinelles Lernen erschweren.

5 Ausblick

Mit Simulation kann man viele Trainingsprozesse beschleunigen. Jedoch brauchte das Team von OpenAI Five für 45.000 Jahre Simulation ca. 10 Monate tatsächliche Rechenzeit. „Scaling will become even more important [...] as the tasks become more challenging.“ (Berner et al. 2019, Seite 16). Diese Zeit zu verringern ist eins der Kernprobleme für Reinforcement Learning, da bei immer komplizierter werdenden Anwendungsdomänen nicht unendlich Zeit zum Trainieren verfügbar sein kann.

Mit immer besser werdenden Grafikkarten wird z.B. die Fehlerrückführung bei neuronalen Netzen immer schneller. Dies wird KI unabhängig von Software schneller machen (siehe Cai et al. 2020, S. 3). Viele führende Unternehmen, allen voran Google, forschen aktuell an Quanten-Computern. Dabei bestehen aktuell noch Probleme wie verfälschte Berechnungen durch Quanteneffekte bei der Anfertigung in derart kleinen Größenordnungen, denen entgegen aber immer weitere Fortschritte erbracht werden (siehe Niu und Boixo 2019). Google arbeitet daran, dass solche Computer nicht nur die Kryptographie revolutionieren, sondern auch im Bereich Maschinelles Lernen essentieller Bestandteil werden (siehe Broughton et al. 2020, S. 2).

Literaturverzeichnis

Bai, Y., und Jin, C. 2020. „Provable Self-Play Algorithms for Competitive Reinforcement Learning“ in *Proceedings of the 37th International Conference on Machine Learning*, Band 119 von *Proceedings of Machine Learning Research*, H. Daumé III und A. Singh, (hrsg.), online: PMLR, S. 551–560.

Baker, B., Kanitscheider, I., Markov, T., Wu, Y., Powell, G., McGrew, B., und Mordatch, I. 2020. „Emergent Tool Use From Multi-Agent Autocurricular,“ in *International Conference on Learning Representations (ICLR)*, D. Song, K. Cho, und M. White, (hrsg.), online: https://arxiv.org/abs/2002.04017.

Benson, D. B. 1976. „Life in the Game of Go,“ *Information Sciences* (10:2), S. 17–29.

Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., Józefowicz, R., Gray, S., Olsson, C., Pachocki, J., Petrov, M., de Oliveira Pinto, H. P., Raiman, J., Salimans, T., Schlatter, J., Schneider, J., Sidor, S., Sutskever, I., Tang, J., Wolski, F., und Zhang, S. 2019. „Dota 2 with Large Scale Deep Reinforcement Learning,“ *CoRR* (abs/1912.06680). https://arxiv.org/abs/1912.06680.

Broughton, M., Verdon, G., McCourt, T., Martinez, A. J., Yoo, J. H., Isakov, S. V., Massey, P., Niu, M. Y., Halavati, R., Peters, E., und andere 2020. „Tensorflow quantum: A software framework for quantum machine learning,“ *arXiv* (2003.02989.). https://arxiv.org/abs/2003.02989.

Cai, S., Bileschi, S., Nielsen, E. D., und Chollet, F. 2020. *Deep Learning with JavaScript*, Shelter Island, New York, USA: Manning Publications Co.

Davis, R. 1996. „What Are Intelligence? And Why?“ *AI Magazine* (19:1).

Eugene F. Grant, R. L. 1952. „IT,“ *The New Yorker* (2. August).

Goyal, A., Khandelwal, I., Anand, R., Srivastava, A., und Swarnalatha, P. 2017. „A Comparative Analysis of the Different Data Mining Tools by Using Supervised Learning Algorithms,“ in *Proceedings of the Eighth International Conference on Soft Computing and Pattern Recognition (SoCPaR 2016)*, A. Abraham, A. K. Cherukuri, A. M. Madureira, und A. K. Muda, (hrsg.), Vellore, Indien: Springer International Publishing, S. 105–112.

Guttman, L., und Levy, S. 1991. „Two Structural Laws for Intelligence Tests,“ *Intelligence* (15:1), S. 79–103.

Géron, A. 2019. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*, Sebastopol, USA: O'Reilly Media.

Heinrich, J., und Silver, D. 2016. „Deep Reinforcement Learning from Self-Play in Imperfect-Information Games," *arXiv* (abs/1603.01121). https://arxiv.org/abs/1603.01121.

Kachalsky, I., Zakirzyanov, I., und Ulyantsev, V. 2017. „Applying Reinforcement Learning and Supervised Learning Techniques to Play Hearthstone," in *16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, B. Chen, (hrsg.), Cancun, Mexiko: IEEE, S. 1145–1148.

Konar, A. 2000. *Artificial Intelligence and Soft Computing, Behavioral and Cognitive Modeling of the Human Brain*, Washigton DC, USA: CRC Press.

Li, L., Lin, Y.-L., Zheng, N.-N., Wang, F.-Y., Liu, Y., Cao, D., Wang, K., und Huang, W.-L. 2018. „Artificial intelligence test: a case study of intelligent vehicles," *Artificial Intelligence Review* (50:10).

Niu, M. Y., und Boixo, S. 2019. „Improving Quantum Computation with Classical Machine Learning," *Google AI Blog* (3. Oktober). https://ai.googleblog.com/2019/10/improving-quantum-computation-with.html [Abrufdatum: 04.05.2021].

OpenAI, Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., Petron, A., Paino, A., Plappert, M., Powell, G., Ribas, R., Schneider, J., Tezak, N., Tworek, J., Welinder, P., Weng, L., Yuan, Q., Zaremba, W., und Zhang, L. 2019. „Solving Rubik's Cube with a Robot Hand," *CoRR* (abs/1910.07113). http://arxiv.org/abs/1910.07113.

Rivera, G., Hullett, K., und Whitehead, J. 2012. „Enemy NPC Design Patterns in Shooter Games," in *Proceedings of the First Workshop on Design Patterns in Games*, DPG '12, Raleigh, North Carolina, USA: Association for Computing Machinery.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., und Hassabis, D. 2017. „Mastering the game of Go without human knowledge," *Nature* (550:7676), S. 354–359. DOI: 10.1038/nature24270.

Smolensky, P. 1987. „Connectionist AI, Symbolic AI, and the Brain," *Artificial Intelligence Review* (1:2), S. 95–109.

Zadeh, L. 1994. „Fuzzy logic, neural networks, and soft computing," *Communications of the ACM* (37:3). DOI: 10.1145/175247.175255.