Research Article

Alexander Dockhorn* und Rudolf Kruse

Modellheuristiken für Effizientes Forward Model Learning

Modelheuristics for Efficient Forward Model Learning

DOI ...
Empfangen 07.02.2021; angenommen ...

Zusammenfassung: Forward Model Learning, also das Erlernen vorwärtsgerichteter Modelle aus Daten, findet Anwendung in der vorhersagebasierten Regelung. Dazu werden Ein- und Ausgaben des Systems beobachtet, um ein Transitionsmodell zu erstellen und Vorhersagen über zukünftige Zeitschritte zu ermöglichen. Insbesondere komplexe Zustandsräume erfordern den Einsatz von spezialisierten Such- und Modellbildungsverfahren. In dieser Arbeit stellen wir Abstraktionsheuristiken für hochdimensionale Zustandsräume vor, welche es ermöglichen, die Modellkomplexität zu reduzieren und in vielen Fällen ein interpretierbares Ergebnis herbeizuführen. In zwei Fallstudien zeigen wir die Wirksamkeit des vorgestellten Verfahrens anhand von Methoden der Künstlichen Intelligenz in Spielen und in Motion Control Szenarien. Deren Übertragung ermöglicht vielversprechende Anwendungen in der Automatisierungstechnik.

Schlagwörter: Vorwärtsgerichtete Modelle, Zerlegung vorwärtsgerichteter Modelle, Lokale vorwärtsgerichtete Modelle, Objekt-basierte vorwärtsgerichtete Modelle, Autonome Steuerung

Abstract: Forward model learning, i.e., learning forward models from data, finds application in prediction-based control. This involves observing inputs and outputs of the system to build a transition model and make predictions about future time steps. In particular, complex state spaces require the use of specialized search and model building techniques. In this work, we present abstraction heuristics for high-dimensional state spaces, which allow to reduce the model complexity and, in many cases, yield an interpretable result. In the context of two case studies, we demonstrate the effectiveness of the presented procedure in the context of artificial intelligence in games and motion control scenarios. The transfer of these methods enables promising applications in automation engineering.

Keywords: Forward Model Learning, Decomposition of Forward Models, Local Forward Model, Object-based Forward Model, Autonomous Control

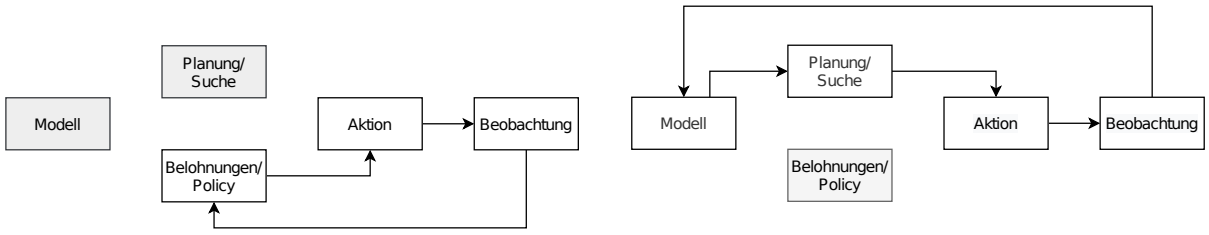
Übermittelt von: Alexander Dockhorn

***Korrespondenzautor: Alexander Dockhorn,** School of Electronic Engineering and Computer Science, Queen Mary University of London, Empire House, 67-75 New Road, London E1 1HH, Vereinigtes Königreich
Rudolf Kruse, Fakultät für Informatik, Otto-von-Guericke-Universität, Universitätsätsplatz 2, 39106 Magdeburg, Deutschland

1 Motivation

Der Lernprozess eines autonomen Agenten in einem für ihn unbekannten System kann in vielen Anwendungen durch Bestärkendes Lernen (Reinforcement Learning) und Modell-Lernen (Model Learning) algorithmisch realisiert werden. Bestärkendes Lernen basiert auf dem Erlernen einer Aufgabe anhand von Belohnungssignalen. Diese können sowohl externer (Handlungserfolg) als auch interner (intrinsische Motivation) Natur sein. Durch Wiederholungen lernt der Agent, den zu erwartenden Wert einer Aktion in verschiedenen Situationen abzuschätzen. Das Verhalten des Agenten ergibt sich schließlich durch Wahl der Aktion mit dem höchsten Wert.

In einigen Anwendungen ist jedoch eine sinnvolle Definition des Belohnungssignals kaum möglich. In solchen Fällen stellt das Modell-Lernen eine interessante Alternative dar. Hierbei nutzt der autonome Agent nicht das Belohnungssignal, sondern erlernt das ihm zugrundeliegende System durch die Beobachtung von Ein- und Ausgaben des Systems. Das resultierende Modell besteht dann aus einer Transitionsfunktion unter Verwendung von Such- bzw. Planungsverfahren, die für die Entscheidungsfindung genutzt werden können (siehe Abbildung 1).



(a) Reinforcement Learning **(b)** Model Learning

Abb. 1: Vergleich grundlegender Modellklassen. Eingezeichnete Pfeile stellen die gegenseitige Beeinflussung der Komponenten im Laufe des Lernprozesses dar. Grau hinterlegte Elemente sind für die entsprechende Modellklasse nicht von Bedeutung.

Das Erlernen der Transitionsfunktion im Falle hochdimensionaler Ein- und Ausgaberäume ist sehr komplex und bedingt spezialisierte Lernverfahren. Zur strukturellen Analyse dieser Lösungsansätze nutzt man typischerweise Benchmark-Anwendungen. Oft werden dazu Spiele genutzt, weil man hier über die benötigten hohen Datenmengen verfügt. Die Verwendung von Spielen zur Entwicklung und Verbesserung von Methoden der Künstlichen Intelligenz hat eine lange Tradition. Ein bekanntes Beispiel ist die Entwicklung des Agenten AlphaGO, welcher 2016 in der Lage war, den damals amtierenden Weltmeister Lee Sedol zu besiegen [27]. Das in diesem System genutzte Lösungsverfahren konnte erfolgreich auf andere Anwendungsfelder wie die Proteinfaltung übertragen werden [26]. Tiefe neuronale Netze, wie sie in AlphaGO verwendet wurden, können sehr gut komplexe Umgebungsmodelle abbilden [16]. Solche Modelle benötigen jedoch sehr viele Trainingsdaten und eine hohe Trainingszeit. Zudem ist die Entscheidungsfindung meistens intransparent.

In dieser Arbeit stellen wir deshalb Modellbildungsheuristiken vor, welche zunächst die Komplexität des Ein- und Ausgaberaums reduzieren und somit die Modellbildung vereinfachen. Hierbei werden Abhängigkeiten im System genutzt, um voneinander unabhängige Komponenten zu finden und gesondert zu modellieren. Neben der automatischen Erkennung dieser Abhängigkeiten können Heuristiken auch durch die Einbindung von Vorwissen generiert werden. Wir erklären diesen Prozess anhand zweier Fallstudien (Erlernen von Spielen, Motion Control) und zeigen die Wirksamkeit dieser Methodik. Hierbei wird gezeigt, dass durch den Einsatz der vorgeschlagenen Modellheuristiken selbst einfache Modelle wie z.B. Entscheidungsbäume in die Lage versetzt werden, komplexe Systeme auf verständliche Weise zu modellieren.

Grundlage dieser Betrachtung stellt die Agent-Umgebung-Schnittstelle (Agent-Environment Interface) dar [28], welche wir im folgenden Abschnitt einführen und zur Herleitung der Lernverfahren für Agenten nutzen. In Abschnitt 3 erläutern wir die Dekomposition eines Umgebungsmodells zur Erstellung eines Transitionsmodells (Forward Models). Im Abschnitt 4 wird gezeigt, wie die Einbettung von Vorwissen über das System zur Generierung von Transitionsmodellen genutzt werden kann und präsentieren die daraus resultierenden Modellbildungsheuristiken des Lokalen Transitionsmodells und des Objekt-basierten Transitionsmodells. Abschnitt 5 fasst die Ergebnisse der beiden Fallstudien zusammen und diskutiert Vor- und Nachteile der demonstrierten Modellbildungsheuristiken. Im letzten Abschnitt behandeln wir offene Forschungsfragen bei diesem Ansatz sowie mögliche Anwendungsfelder der vorgestellten Methodik.

2 Grundlagen autonomer Lernverfahren

Die Agent-Umgebung-Schnittstelle [28] erlaubt dem Nutzer die Sicht auf die Lernprozesse von Agenten in ihnen unbekannten Umgebungen. Sie besteht aus den fünf Komponenten: Agent, Umgebung, Umgebungszustand, Aktionen des Agenten, und die Belohnung für den Agenten. Hierbei versucht der Agent selbstständig das Interagieren mit der Umgebung zu erlernen, um eine ihm gestellte Aufgabe zu erfüllen. Zu jedem Zeitpunkt befindet sich das System (die Umgebung und der Agent) in einem Zustand $S \in \mathcal{S}$. Abhängig von diesem wählt der Agent eine Aktion $A \in \mathcal{A}$, woraufhin sich der Zustand ändert. Hierbei kann es sich sowohl um diskrete als auch um zumindest teilweise kontinuierliche Aktionen und Zustände handeln. Die Veränderung des Zustands wird durch das Umgebungsmodell bestimmt und kann als Wahrscheinlichkeitsfunktion

modelliert werden:

$$P(S_t, R_t \mid S_{t-1}, A_{t-1}, \ldots, S_0, A_0)$$

Hierbei wird die Sequenz vorheriger Zustände und Aktionen des Agenten auf einen Folgezustand S_t und eine Belohnung (Reward) R_t abgebildet. Letztere gibt dem Agenten Auskunft über den Erfolg bzw. Misserfolg im Erfüllen der ihm gestellten Aufgabe. Das Umgebungsmodell kann in ein Transitionsmodell (State Transition Model) und ein Belohnungsmodell (Reward Model) aufgeteilt werden.

$$\text{Transitionsmodell:} \quad p(S_t \mid S_{t-1}, A_{t-1}, \ldots, S_0, A_0)$$
$$\text{Belohnungsmodell:} \quad q(R_t \mid S_{t-1}, A_{t-1}, \ldots, S_0, A_0)$$

Lernalgorithmen nutzen die Informationen aus vorherigen Interaktionen mit ihrer Umgebung, um diese Teilmodelle anhand der Beobachtungen zu approximieren. Hierbei liegt der Fokus von Bestärkenden Lernalgorithmen auf der Approximation des Belohnungsmodells, wohingegen Modelllernverfahren das Transitionsmodell approximieren. In dieser Arbeit fokussieren wir uns auf Letztere.

Während Bestärkendes Lernen [28] einen „eifrigen Lernansatz“ (eager learning) repräsentiert, welcher den Wert einer jeden Aktions-Zustandskombination bestimmt, müssen Modellernverfahren den Wert einer Aktion anhand des Transitionsmodells bestimmen. Hierfür wird die optimale Aktion durch ein Suchverfahren ermittelt, welches Simulationen der Umgebung mithilfe des gelernten Transitionsmodells nutzt, um die möglichen Resultate der Aktion zu bestimmen und daraus die beste Aktion abzuleiten. In großen Zustandsräumen ist eine vollständige Exploration des Zustandsraums mithilfe des Transitionsmodells nicht möglich.

Solche Situationen kann man gut in Wettbewerben wie der General Video Game AI Competition testen. Die Aufgabe ist es, einen Agenten zu finden, der unterschiedliche Spiele ohne vorheriges Training, jedoch unter Kenntnis des Transitionsmodells, bestmöglich spielen muss [22]. Typischerweise werden für solche Aufgaben heuristische Suchverfahren wie Monte Carlo Tree Search [5] oder dem Rolling Horizon Evolutionary Algorithm [14] verwendet. Der Einsatz von Suchverfahren bedingt ein akkurates Modell. Ist dieses nicht bekannt, muss es anhand von Modell-Lernverfahren gelernt werden, um eine effektive Suche zu erlauben. Alternativ können hybride Modelle verwendet werden, welche beispielsweise das Transitionsmodell durch Bestärkendes Lernen approximieren oder einen Agenten in einer durch ein gelerntes Modell simulierten Umgebung trainieren [4, 20, 23]. Eine besondere Herausforderung beim Lernen eines Transitionsmodells stellt die Handhabung von großen Zustandsräumen dar, für die wir im nächsten Abschnitt mögliche Lösungen präsentieren.

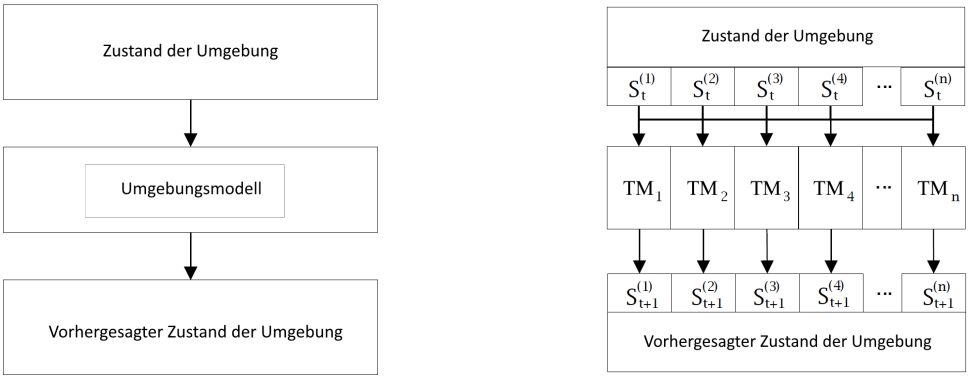
3 Dekomposition von Transitionsmodellen

Das Ziel des Modell-Lernens ist das Erlernen von Transitionsmodellen mit Hilfe von Daten und unter Berücksichtigung des Vorwissens über das System. Die verschiedenen Arten des Modell-Lernens unterscheiden sich in der Aufgabe und Funktionsweise des zu erlernenden Modells [20]. Transitionsmodelle treffen anhand des aktuellen Zustands des Systems und der Aktion des Agenten eine Vorhersage über den nächsten zu beobachtenden Zustand. In dieser Arbeit beschränken wir uns auf das Lernen von Transitionsmodellen, welche zur Realisierung einer prädiktiven Suche verwendet werden können. Für die bessere Darstellung des Verfahrens beziehen wir uns im Folgenden nur auf Markov-Entscheidungsprobleme, deren Ausgabe nur von dem zuletzt beobachteten Zustand und der letzten Aktion des Agenten abhängig ist. Durch die Unabhängigkeit von vorherigen Zeitschritten lässt sich das Transitionsmodell wie folgt reduzieren:

$$\begin{aligned} &p(S_t \mid S_{t-1}, A_{t-1}, \ldots, S_0, A_0) \\ &= p(S_t \mid S_{t-1}, A_{t-1}) \end{aligned}$$

Sind Beobachtungen von Ein- und Ausgaben eines unbekannten Transitionsmodells vorhanden, können wir anhand eines überwachten Lernverfahrens ein approximatives Modell generieren. Bildet dieses Modell den aktuellen Zustand und die Aktion des Agenten direkt auf den Folgezustand ab, sprechen wir von einem Ende-zu-Ende Transitionsmodell (End-to-End Forward Model). Dieses Verfahren wird hauptsächlich in Kombination mit tiefen neuronalen Netzen verwendet, welche im Falle hinreichender Trainingsdaten oft in der Lage sind, komplexe Modelle zu erlernen [16]. Die Generierung von ausreichend Trainingsdaten ist jedoch oft zeit- und kostenaufwändig.

Aus diesem Grund nutzen wir Dekompositionstechniken: Anstatt den gesamten Zustandsübergang in einem Modell abzubilden, kann das Transitionsmodell von Zuständen bestehend aus mehreren Zustandsvariablen in Teilmodelle zerlegt werden. Hierfür werden voneinander unabhängige Mengen an Variablen bestimmt und dann in Komponentenmodellen abgebildet. Eine solche Zerlegung kann durch die Einbindung von Expertenwissen generiert oder durch einen Datensatz beobachteter Zustandsübergänge approximiert werden [9]. Bei einer vollständigen Unabhängigkeit beteiligter Zustandsvariablen kann ein Modell für die Vorhersage einer Variable bzw. deren Änderung bis zum nächsten Zeitschritt gelernt werden.



(a) Ende-zu-Ende Transitionsmodell

(b) Dekomponiertes Transitionsmodell

Abb. 2: Visueller Vergleich von Modellarchitekturen [6]. Während das Ende-zu-Ende Transitionsmodell eine direkte Abbildung des aktuellen Zustands auf den Folgezustand darstellt, werden im Dekomponierten Transitionsmodell mehrere Teilmodelle (TM) des Umgebungsmodells gelernt, welche jeweils den aktuellen Zustand auf den Folgezustand einer einzelnen Zustandskomponente abbilden. Das Ergebnis aller Teilmodelle wird aggregiert um den vorhergesagten Folgezustand zu erhalten.

$$\text{Komponente } i\text{:} \quad p(S_t^i \mid S_{t-1}, A_{t-1})$$

$$\text{Änderung von } i\text{:} \quad p(S_{t-1}^i - S_t^i \mid S_{t-1}, A_{t-1})$$

Die Prädiktion des Folgezustands ergibt sich aus der Aufteilung des Zustands in voneinander unabhängige Komponenten, der Prädiktion dieser Komponenten und der anschließenden Aggregierung der Resultate. Abbildung 2 [6] zeigt einen Vergleich des Prädiktionsablaufs zwischen Ende-zu-Ende und Dekomponierten Transitionsmodellen. Einen interessanten Zwischenweg stellt die Verwendung von Graph Neural Networks dar [31], welche die Abhängigkeitsstruktur der zur Verfügung stehenden Daten modellieren. Hierbei handelt es sich um ein Ende-zu-Ende Modell, welches durch die interne Kodierung von Abhängigkeiten auch komplexe Umgebungen modellieren kann. Wie auch bei anderen Ansätzen benötigt das Training eines solchen Netzes eine Vielzahl an Daten. Im Folgenden stellen wir Modellbildungsheuristiken vor, welche einen Teil der Abhängigkeitsstruktur vorgeben und somit das Lernen eines Transitionsmodells beschleunigen können.

4 Modellbildungsheuristiken

Wie von uns in einer früheren Arbeit gezeigt [9], kann die Bestimmung einer Dekomposition des Transitionsmodells durch Abhängigkeitsanalysen erfolgen. In diesem Aufsatz zeigen wir, wie das Vorwissen über das Transitionsmodell in Form von Modellbildungsheuristiken genutzt werden kann. Im Folgenden stellen wir lokale (Local Forward Model) und objekt-basierte Transitionsmodelle (Object-based Forward Model) vor.

4.1 Lokale Transitionsmodelle

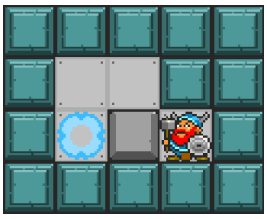
Lokale Transitionsmodelle ähneln in ihrer Funktionsweise faltenden neuronalen Netzen (Convolutional Neural Networks) [15]. Hierbei wird der Folgezustand anhand unabhängiger Beobachtungen einzelner Informationsquellen vorhergesagt, wobei der Folgezustand lediglich vom eigenen Zustand und dem Zustand „benachbarter“ Quellen abhängt. Nachfolgend gehen wir davon aus, dass voneinander weit entfernte Informationsquellen voneinander unabhängig sind und demnach der Folgezustand einer der Quellen jeweils nur von benachbarten Quellen abhängt. Ein solches System ist beispielsweise bei der Betrachtung von räumlichen Interaktionen aus der Vogelperspektive gegeben. Zwei Objekte oder Personen, die weit voneinander entfernt sind, können sich hierbei nicht direkt beeinflussen, wohingegen nebeneinanderliegende Objekte sich gegenseitig durch Interaktionen beeinflussen können. Die lokale Transitionsfunktion beschreibt die Veränderung einer Zustandsvariable abhängig von dem Wert der Zustandsvariablen in ihrer Nachbarschaft:

$$fm_i : \left(N(S_t^{(i)}), A_t\right) \longmapsto S_{t+1}^{(i)}$$

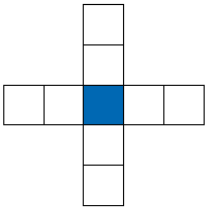
wobei $N(S_t^{(i)}) \subseteq \mathcal{S}$ die Menge benachbarter Zustandsvariablen der Zustandsvariable i zum Zeitpunkt t beschreibt. Sind eine Distanzfunktion d und ein Schwellwert ε gegeben, kann die lokale Nachbarschaft einer Zustandsvariable durch die folgende Funktion bestimmt werden:

$$N(S_t^{(i)}) = \left\{ S_t^{(j)} \mid d(S_t^{(i)}, S_t^{(j)}) \leq \varepsilon, \quad j \in 1, \ \ldots, \ n \right\}$$

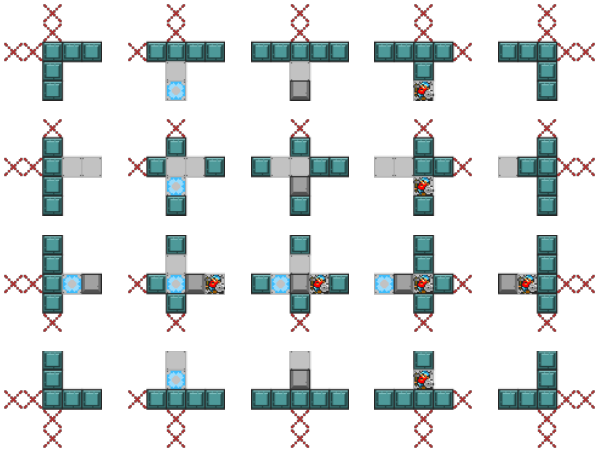
Motiviert von der ersten Anwendungen eines Lokalen Transitionsmodells zur Modellierung von Conway's



(a) Zustand des Spiels Sokoban bestehend aus 5×4 Elementen



(b) Nachbarschaftsumgebung des zentralen Elements.



(c) Extrahierte Nachbarschaften für den dargestellten Zustand. Die jeweiligen Muster entsprechen den in Abbildung 3(a) dargestellten Elementen. Durch ein rotes Kreuz markierte Elemente sind außerhalb des beobachtbaren Zustands und werden durch einen Standardwert ersetzt.

Abb. 3: Extraktion von Nachbarschaften am Beispiel Sokoban

Game of Life [18] haben wir das Prinzip auf nicht-binäre Zustandsvariablen erweitert [13]. Im Folgenden gehen wir auf die Modellierung des Spiels Sokoban ein, in welchen ein Agent versucht, graue Blöcke auf die hierfür vorgesehenen Zielpositionen zu verschieben. Nachfolgend gehen wir von einem Zustand S aus, welcher sich als Matrix T darstellen lässt.

$$T = \begin{bmatrix} T(1,1) & \dots & T(1,m) \\ \vdots & \ddots & \vdots \\ T(n,1) & \dots & T(n,m) \end{bmatrix}$$

Hierbei beschreibt jeder Eintrag $T(i,j)$ den nominalen Zustand einer Zustandsvariable an der Position (i,j). Im Kontext von Sokoban kodiert der Wert $T(i,j)$ die Art des Elements an der Position, z.B. Charakter, grauer Block, Wand, etc. (vgl. Abbildung 3a). Die lokale Transitionsfunktion ergibt sich aus:

$$fm_{x,y} : \Big(N(x,y)_t, A_t\Big) \longmapsto T(x,y)_{t+1}$$

Sie beschreibt die Veränderung eines Elements abhängig von der aktuellen Nachbarschaft des Elements und der Aktion des Agenten. Tests mit dem Spiel Sokoban [13] zeigten, dass präzise Modelle anhand weniger Wiederholungen gelernt und auf unbekannte Zustände übertragen werden können. Die Extraktion von Nachbarschaftsumgebungen ist beispielhaft in Abbildung 3 dargestellt. Das resultierende Modell wird in Abbildung 4a schematisch visualisiert. Da diese beiden Anwendungen nur eine kleine Bandbreite von möglichen Spielen abdecken, werden in Abschnitt 5 dieser Arbeit weitere Experimente zum Lernen von unbekannten Spielen vorgestellt.

4.2 Objekt-basierte Transitionsmodelle

Im Rahmen eines Objekt-basierten Transitionsmodells wird nicht von räumlichen, sondern von semantischen Abhängigkeiten ausgegangen. Der Agent und die Umgebung, in der er sich bewegt, können aus semantisch abgegrenzten Komponenten bestehen. Bezogen auf Spiele kann beispielsweise das Verhalten einzelner Gegner separat voneinander modelliert werden. Bei einem Objekt-basierten Transitionsmodell werden Informationen über separierbare Entitäten genutzt, um eine entsprechende Aufteilung in voneinander unabhängige Modellkomponenten zu realisieren.

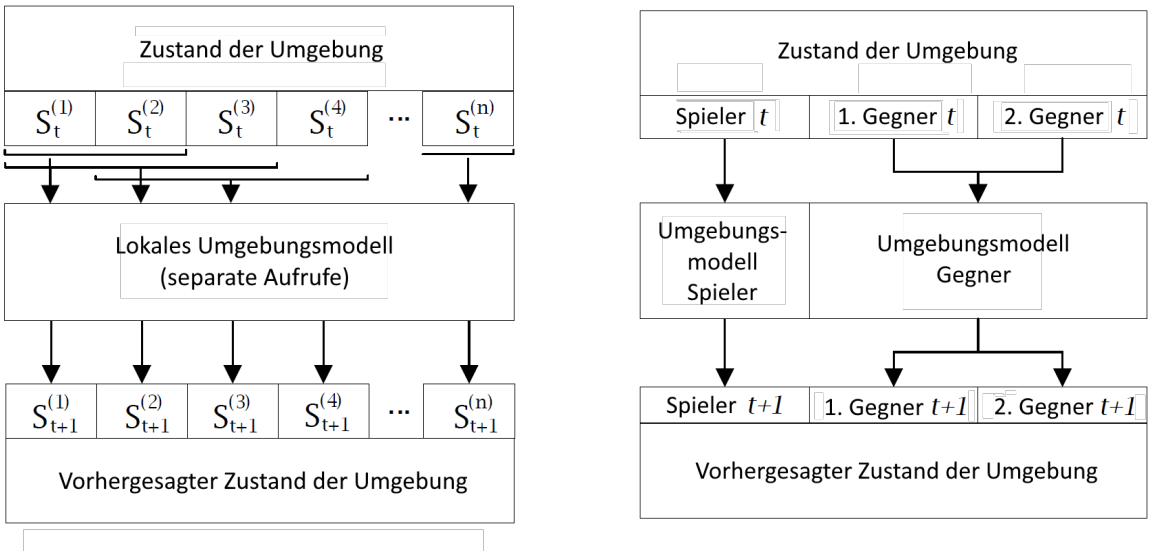
Im Folgenden gehen wir davon aus, dass der Agent neben der Zustandsbeschreibung in Form seiner Zustandsvariablen auch eine objekt-basierte Partitionierung dieser Zustandsvariablen kennt. Eine Partitionierung kann die Mengen von Sensorwerten $1,\dots,m$ beschreiben, die zu einem gemeinsamen Objekt in der Umgebung gehören, wie z. B. dem Avatar des Agenten oder eines Nicht-Spieler-Charakters.

$$\begin{aligned} \mathcal{S} &= (S^{(1)}, S^{(2)}, \dots, S^{(n)}) \\ &= (\underbrace{S^{(1,1)}, \dots, S^{(1,i)}}_{\text{Objekt 1}}, \dots, \underbrace{S^{(m,1)}, \dots, S^{(m,k)}}_{\text{Objekt m}}) \end{aligned}$$

Für jedes Objekt approximieren wir ein Transitionsmodell fm_i, welches dessen aktuellen Zustand und die Aktion des Agenten auf den nächsten Zustand abbildet:

$$fm_i : \Big((S_t^{(i,1)}, \dots, S_t^{(i,k)}),\ A_t\Big) \longmapsto (S_{t+1}^{(i,1)},\ \dots,\ S_{t+1}^{(i,k)})$$

wobei jedes Objekt eine unterschiedliche Anzahl an Komponenten umfassen kann.



(a) Lokale Transitionsmodelle: Für jedes beobachtete Element wird die Nachbarschaftsumgebung extrahiert und ein separater Aufruf des Modells durchgeführt. Die Ergebnisse werden aggregiert um den Folgezustand vorherzusagen.

(b) Objekt-basierte Transitionsmodelle: Für jedes beobachtete Element wird ein separater Aufruf des zugehörigen Modells durchgeführt. Elemente gleichen Typs verwenden dasselbe Modell. Die Ergebnisse werden aggregiert um den Folgezustand vorherzusagen.

Abb. 4: Visueller Vergleich der vorgestellten Modellbildungsheuristiken [6]

Sind die Attribute eines Objekts zudem voneinander unabhängig, kann ein Dekomponiertes Transitionsmodell verwendet werden, um das Transitionsmodell des Objekts zu modellieren.

$$fm_{i,j} : \left((S_t^{(i,1)}, \ldots, S_t^{(i,k)}),\ A_t \right) \longmapsto S_{t+1}^{(i,j)}$$

Dies führt zu einer mehrstufigen Abstraktion des Systems. Um den nächsten Zustand vorherzusagen, wird der aktuelle Zustand zunächst partitioniert und in Folge dessen jedes Objektmodell aufgerufen. Die Ausgaben dieser werden nachfolgend aggregiert, um den vorhergesagten Zustand zu erhalten.

$$\begin{aligned} fm(S_t, A_t) &= (fm_1(S_t, A_t),\ \ldots,\ fm_n(S_t, A_t)) \\ &= ((fm_{1,1}((S_t^{(1,1)}, \ldots, S_t^{(1,k)}), A_t),\ \ldots, \\ &\qquad fm_{m,k}((S_t^{(m,1)}, \ldots, S_t^{(m,k')}), A_t))) \\ &= (S_{t+1}^{(1)},\ S_{t+1}^{(2)},\ \ldots,\ S_{t+1}^{(n)}) = S_{t+1} \end{aligned}$$

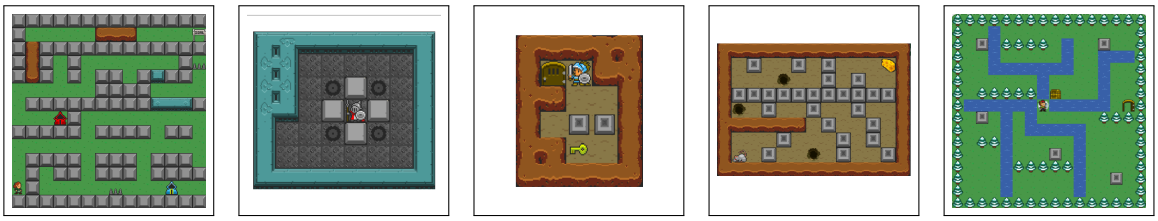
Abbildung 4 stellt die vorgestellten Modellbildungsheuristiken überblicklich dar. Insbesondere der Vergleich zu Abbildung 2 zeigt wie die zusätzliche Annahme von Unabhängigkeiten beobachtbarer Zustandskomponenten zu einer Reduzierung der Modellanzahl und die Komplexität der Eingaben enthaltener Teilmodelle führt.

5 Fallstudien

Wir haben in diesem Aufsatz vier Arten von Transitionsmodellen mit deren Modellbildungsheuristiken vorgestellt und können nun deren Einsatz in einer variantenreichen Auswahl an Szenarien demonstrieren. Die Ergebnisse dieser Fallstudien sind im Rahmen früherer Arbeiten [3, 6, 7, 8, 10, 11, 12, 13, 18] entstanden und werden hier zusammengetragen, um ein vollständiges Bild der Vor- und Nachteile der vorgestellten Verfahren zu vermitteln.

5.1 General Game-Learning

In unserer ersten Fallstudie widmen wir uns dem Thema des General Game-Learning. Hierbei ist es die Aufgabe sogenannten KI-Agenten, eine Menge ihm unbekannter Spiele ohne Kenntnis deren Regelwerke zu erlernen. Das General Video Game AI Framework (GVGAI) [22] stellt eine Sammlung aus über 100 Spielen mit jeweils 5 Leveln bereit, welche über eine einheitliche Schnittstelle gesteuert werden können. Diese Spiele sind grafisch sehr einfach präsentiert, sie bieten jedoch eine Vielzahl an unterschiedlichen Funktionen (siehe Abbildung 5). Diese umfassen unterschiedliche Bewegungsarten, Siegbedingungen, Geg-



(a) Labyrinth **(b)** Sokoban **(c)** Bait **(d)** Escape **(e)** Vortex

Abb. 5: Fünf Game Learning Umgebungen und ihre Repräsentation im GVGAI Framework.

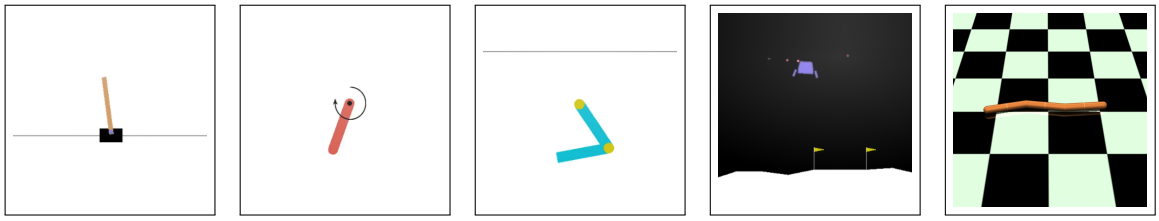
ner, sammelbaren Gegenständen, Interaktionen zwischen Objekten, etc.

Die in diesem Aufsatz vorgestellte Methoden des Modell-Lernens wurden erstmals in Verknüpfung mit hierarchischen Wissensbasen zur Modellierung von GVGAI Spielen verwendet [7]. In dieser Studie wurden 10 Spiele des GVGAI Frameworks ausgewählt, um Modelle anhand von wenigen Durchläufen zu erlernen. Die Studie nutzt die gleichen Limitierungen wie der Learning-Track der GVGAI Competition [21]. Hierbei sind KI-Agenten auf lediglich 5 Minuten Echtzeit-Interaktion mit 3 Leveln eines ihnen zuvor unbekannten Spiels limitiert, nach welchen ihre Leistung anhand von 2 weiteren Leveln getestet wird. Die von unserem Agenten gelernten Modelle waren in der Lage, trotz der stark limitierten Trainingszeit zahlreiche Komponenten der Spiele korrekt abzubilden. Der Agent nutzte Monte Carlo Tree Search und Breitensuche für die Entscheidungsfindung und konnte trotz der verbleibenden Vorhersagefehler andere Agenten deutlich übertreffen. Weitere Analysen zeigen, dass Monte Carlo Tree Search am robustesten gegenüber dem verbleibenden Vorhersagefehler ist und die beste Gesamtleistung unter allen getesteten Agenten erzielen konnte [3].

Der Einfluss des Gesamtfehlers wurde in weiteren Aufsätzen analysiert [12, 13]. Bei einem Vergleich von unterschiedlichen Klassifikatoren zum Lernen von Modellen können selbst ganz einfache Modelle wie Entscheidungsbäume zu einem geringen Vorhersagefehler und einer schnelleren Prädiktion führen [12]. Des Weiteren sind die resultierenden Modelle durch ihre simple Struktur verständlicher als vergleichbare Neuronale Netze. Interessant ist, dass die Leistung jedoch nicht ausschließlich von der Genauigkeit des Modells abhängt [6], sondern teilweise Modelle mit geringerer Genauigkeit zu einer vergleichsweise besseren Leistung geführt haben. Ist beispielsweise die Vorhersage eines von der Aufgabenbewältigung unabhängigen Objekts fehlerhaft, so hat diese nur einen geringen Einfluss auf die Gesamtleistung des Agenten gezeigt.

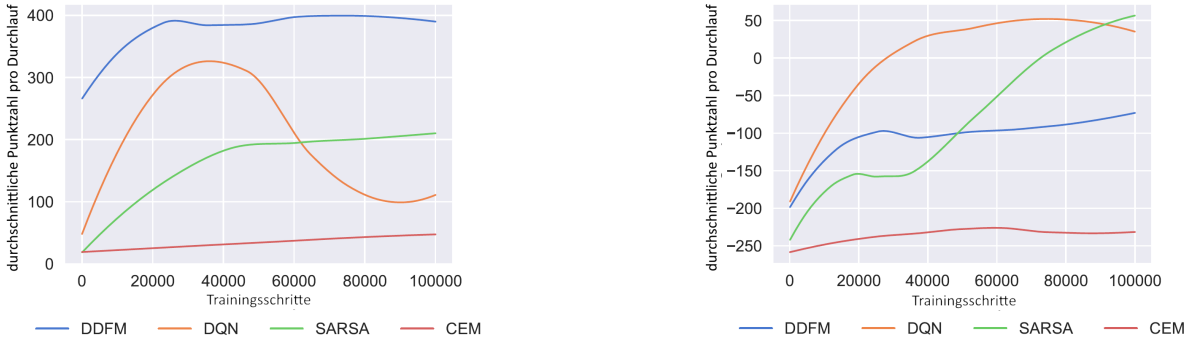
In einer vergleichenden Studie [6] zwischen Lokalen und Objekt-basierten Transitionsmodellen anhand von 30 Spielen des GVGAI Frameworks zeigte sich, dass die Leistung eines Modells stark von den zugrundeliegenden Eigenschaften des Systems abhängt. Hierbei ist es nicht verwunderlich, dass die Leistung sinkt, wenn die Annahmen zur Unabhängigkeit nicht erfüllt sind. Ein weiteres Resultat der Arbeit ist die Klassifizierbarkeit der Probleme anhand der Leistung der Modelle. So können Unterkategorien wie „Spiele in denen ein Labyrinth vorkommt“ oder „Spiele in denen Objekte verschoben werden“ anhand der Leistung der getesteten Agenten identifiziert werden.

Wie bereits zuvor erwähnt ist eine besondere Eigenschaft des vorgestellten Agenten die Interpretierbarkeit seines Modells und der darauf basierenden Entscheidungsfindung. Während im Bestärkenden Lernen die Bewertung einer Aktion nur durch das Einbeziehen des gesamten Lernverlaufs nachvollziehbar wird, können Entscheidungen von Modell-lernenden Agenten anhand ihres aktuellen Modells und des resultierenden Planungsverlaufs analysiert werden. Hierbei ist es möglich, antizipierte Zustände zu visualisieren und das vom Agenten erwartete Ergebnis seiner Aktionen nachzuvollziehen. Wie in der Arbeit von Dockhorn et. al [13] demonstriert, ermöglicht die Visualisierung von antizipierten Zuständen die Erklärung von Entscheidungen durch ein lokales Transitionsmodell des Spiels Sokoban. Hierbei weicht der vom Modell vorhergesagte Zustand deutlich vom eigentlichen Ergebnis der Aktionsfolge ab. Eine solche Abweichung kann für die Generierung weiterer Testbeispiele verwendet werden, um das Modell zu verbessern. Auch ohne eine Visualisierung können Regel-basierte Verfahren wie die zuvor genannten hierarchische Wissensbasen für den Menschen verständlich formuliert werden [3]. Hierbei konnten beispielsweise die Bewegungsabläufe eines GVGAI Agenten durch 11 simple Regel approximiert werden, z.B. „Wird die Aktion *Links* ausgeführt und befindet sich links vom Agenten kein Objekt, so bewegt sich der Agent nach links“.



(a) Cart Pole (b) Pendulum (c) Acrobot (d) Lunar Lander (e) Swimmer

Abb. 6: Fünf Steuerungsaufgaben und ihre Repräsentation im OpenAI Gym Framework.



(a) Cart Pole Trainingsverlauf (b) Acrobot Trainingsverlauf

Abb. 7: Vergleich des Trainingsverlaufs [10] eines Differentialen Dekomponierten Transitionsmodells (DDFM) und den Reinforcement Learning Agenten DQN [19], Sarsa [28], und CEM [29]. Die Diagramme zeigen den durchschnittlichen Episodenertrag pro Trainingsschritt, geglättet mit der lokalen Regression (loess). Jeder Agent wurde 10 Mal für jeweils 100000 Schritte trainiert.

5.2 Motion Control

In unserer zweiten Fallstudie verlassen wir den Anwendungsbereich Spiele und transferieren die präsentierten Modelle auf Aufgaben der Bewegungsregelung. Hierfür betrachten wir die Ergebnisse von fünf Anwendungen, welche dem Open AI Gym Framework entnommen sind [10]. Diese unterscheiden sich von den vorherigen Szenarien durch die kontinuierlichen Zustandsräume und Belohnungen sowie die teilweise kontinuierlichen Aktionsräume.

Für die Modellierung der jeweiligen Systeme wird ein Differentielles Dekomponiertes Transitionsmodell verwendet. Während in vorherigen Anwendungen hauptsächlich Klassifikationsmodelle (z.B. Entscheidungsbäume und hierarchische Wissensbasen) verwendet wurden, werden hier aufgrund der kontinuierlichen Zustandsräume Regressionsmodelle verwendet (z.B. lineare Regression und Entscheidungsbaum-Regression). In der Fallstudie [10] werden Agenten über 100.000 Zeitschritte trainiert und deren Leistung mit Deep Reinforcement Learning Agenten verglichen. Für jedes Szenario werden anwendungsspezifische Leistungsmerkmale zur Bewertung der Agenten verwendet. Beispielsweise wird bei der Cart Pole Umgebung die Anzahl der Zeitschritte bei denen der Stab einen Winkel von 15 Grad nicht überschreitet und der Wagen den Bildausschnitt nicht verlässt gemessen. Für eine detaillierte Beschreibung der jeweiligen Belohnungsfunktionen verweisen wir auf die Dokumentation des Frameworks (siehe https://gym.openai.com/envs/).

Die Evaluierung [10] zeigt, dass das verwendete Verfahren für die Szenarien Cart Pole, Pendulum und Swimmer schnell zu sehr guten und stabilen Lösungen kommt, hingegen bei Lunar Lander und Acrobot selbst nach Ablauf der Trainingszeit nicht konvergiert und nur geringe Leistungsverbesserungen zeigt (vgl. Abbildung 7). Letzteres liegt an nicht erfüllten Unabhängigkeitsannahmen des verwendeten Modells. Weiterhin kann die Verwendung einer Abhängigkeitsanalyse die Modellbildung unterstützen [12]. Hierfür muss jedoch ein geeigneter Zeitpunkt gefunden werden, zu welchem bereits ausreichend Daten vorhanden sind, um zu einem repräsentativen Ergebnis zu führen.

In einer weiteren Arbeit werden Methoden zur effizienten Nutzung der Trainingszeit zur Verbesserung der Modellgenauigkeit vorgeschlagen [8]. Diese zeigt, dass eine nicht zufällige Exploration während des Trainings zu einer deutlichen Verbesserung der Modellgenauigkeit führen kann. Ein Training, welches die Ungenauigkeit der Vorhersagen für die Auswahl zukünftiger Aktionen mit in Betracht zieht, hat sich als effektivste Variante herausgestellt und geht einher mit Active Learning Strategien [1].

6 Schlussbemerkungen

In dieser Arbeit stellen wir das Dekomponierte Transitionsmodell vor und analysieren die darauf aufbauenden Modellbildungsheuristiken des Lokalen und des Objektbasierten Transitionsmodells. Anhand zweier Fallstudien zeigen wir den Einsatz der vorgestellten Modellierungstechniken in den Bereichen des General Game-Learnings und Motion Control. In Benchmarks konnte eine Vorhersagebasierte Regelung im Vergleich zu anderen etablierten Methoden gute Leistungen erzielen. Die in diesem Aufsatz vorgestellte Möglichkeit, unzureichend bekannte Systeme zu modellieren, zu erlernen und zu adaptieren, hat ein hohes Anwendungspotential.

Ein anwendungsübergreifendes Problem bei der Verwendung von approximierenden Modellen zur Vorhersagebasierten Regelung stellt die Kumulierung von Fehlern dar. Ist beispielsweise die Abweichung eines Transitionsmodells für einen einzelnen Schritt nur gering, so können diese bei aufeinander folgenden Prädiktionen zu immer größer werdenden Abweichungen führen. Der Einsatz von autoregressiven Modellen stellt eine in der Zeitreihenprädiktion häufig verwendete Alternative dar [2]. Weiterhin können nicht-parametrische Mehrschritt-Vorhersagemodelle für die Vorhersage von nicht-linearen Dynamiken verwendet werden [17]. Schlussendlich kann die Messung der Unsicherheit einer Vorhersage zur Bewertung des Vertrauens in eine Simulation verwendet werden, um eine robustere Entscheidungsfindung zu erlauben [8].

Die Generalisierbarkeit von erlernten Modellen ist insbesondere bei selten auftretenden Szenarien problematisch. Für diese liegen oftmals wenige bis gar keine Beobachtungen vor, so dass das gelernte Modell diese nicht ausreichend abbilden kann. Verfahren des One-Shot Learnings könnten dazu beitragen, selten auftretende Ereignisse in die Modellbildung einfließen zu lassen [25, 30]. Eine Alternative zur Modellierung seltener Ereignisse wurde im Kontext von Spielen vorgestellt [11]: hierbei werden separate Modelle für die Abbildung von Ausnahmen trainiert, um die Komplexität des Hauptmodells gering zu halten.

Eine weitere Problematik bei der Vorhersage-basierten Entscheidungsfindung ist ihr Bedarf an zusätzlicher Rechenzeit während der Evaluierung. Während Bestärkende Lernverfahren nach dem aufwändigen Trainingsprozess die Lösung schnell ermitteln können, müssen Vorhersagebasierte Verfahren einen aufwändigen Suchprozess durchlaufen. Die Reduktion der Modellkomplexität unterstützt die schnelle Anwendung des gelernten Forward Models und reduziert die Suchzeit. Insbesondere die Verwendung hierarchischer Modelle können diesen Aspekt weiter verbessern. Eine weitere Alternative stellt der Einsatz von Modell-basierten Bestärkenden Lernmethoden dar, welche Methoden des Modell- und des Bestärkenden Lernens in einem gemeinsamen Modell vereinen [24].

In zukünftigen Arbeiten wird es darum gehen, den Modellbildungsprozess weiter zu automatisieren und dessen Effizienz bezüglich der Anzahl nötiger Trainingsdaten zu verbessern. Insbesondere hierarchische Modelle stellen eine vielversprechende Erweiterung dar, welche das Potential haben, auch komplexere Systeme verständlich zu modellieren. Neben der Verbesserung der Modellbildung ist auch die effektive Verwendung von ungenauen Modellen im Rahmen des Planungsprozesses zu untersuchen. Durch die Einbettung zusätzlicher Informationen über die Modellgenauigkeit können Entscheidungen getroffen werden, die robuster gegenüber Unsicherheiten sind. Aufgrund der vielfältigen Anwendbarkeit und der inhärenten Erklärbarkeit der resultierenden Modelle sind wir davon überzeugt, dass die vorgestellten Verfahren zu vielen interessanten Anwendungen führen können.

Literatur

[1] Charu C Aggarwal u. a. „Active Learning: A Survey“. In: *Data Classification: Algorithms and Applications*. CRC Press, 2014, S. 571–605.

[2] Hirotugu Akaike. „Autoregressive Model Fitting for Control“. In: *Annals of the Institute of Statistical Mathematics* 23.1 (Dez. 1971), S. 163–180. DOI: 10.1007/bf02479221.

[3] Daan Apeldoorn und Alexander Dockhorn. „Exception-Tolerant Hierarchical Knowledge Bases for Forward Model Learning“. In: *IEEE Transactions on Games* (2020), S. 1–1. ISSN: 2475-1502. DOI: 10.1109/TG.2020.3008002.

[4] Arthur Argenson und Gabriel Dulac-Arnold. *Model-Based Offline Planning*. 2021. arXiv: 2008.05556 [cs.LG].

[5] C. B. Browne u. a. „A Survey of Monte Carlo Tree Search Methods“. In: *IEEE Transactions on Computational Intelligence and AI in Games* 4.1 (2012), S. 1–43. DOI: 10.1109/TCIAIG.2012.2186810.

[6] Alexander Dockhorn. „Prediction-based Search for Autonomous Game-playing“. Diss. Otto von Guericke University Magdeburg, 2020, S. 1–231.

[7] Alexander Dockhorn und Daan Apeldoorn. „Forward Model Approximation for General Video Game Learning“. In: *Proceedings of the 2018 IEEE Conference on Computational Intelligence and Games (CIG'18)*. IEEE, Aug. 2018, S. 425–432. DOI: 10.1109/CIG.2018.8490411.

[8] Alexander Dockhorn und Rudolf Kruse. „Balancing Exploration and Exploitation in Forward Model Learning“. In: *Advances in Intelligent Systems Research and Innovation*. Elsevier, 2020, S. 1–20.

[9] Alexander Dockhorn und Rudolf Kruse. „Detecting Sensor Dependencies for Building Complementary Model Ensem-

bles“. In: *Proceedings of the 28. Workshop Computational Intelligence, Dortmund, 29.-30.* 2018, S. 217–234.

[10] Alexander Dockhorn und Rudolf Kruse. „Forward Model Learning for Motion Control Tasks“. In: *2020 IEEE 10th International Conference on Intelligent Systems (IS)*. IEEE, Aug. 2020, S. 1–5. DOI: 10.1109/IS48319.2020.9199978.

[11] Alexander Dockhorn, Chris Saxton und Rudolf Kruse. „Association Rule Mining for Unknown Video Games“. In: *Fuzzy Approaches for Soft Computing and Approximate Reasoning: Theories and Applications*. Hrsg. von Marie-Jeanne Lesot und Christophe Marsala. Springer, 2020.

[12] Alexander Dockhorn, Tim Tippelt und Rudolf Kruse. „Model Decomposition for Forward Model Approximation“. In: *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, Nov. 2018, S. 1751–1757. DOI: 10.1109/SSCI.2018.8628624.

[13] Alexander Dockhorn u. a. „Learning Local Forward Models on Unforgiving Games“. In: *2019 IEEE Conference on Games (CoG)*. London: IEEE, Aug. 2019, S. 1–4. ISBN: 978-1-7281-1884-0. DOI: 10.1109/CIG.2019.8848044. arXiv: 1909.00442.

[14] R. D. Gaina, S. M. Lucas und D. Perez-Liebana. „Rolling Horizon Evolution Enhancements in General Video Game Playing“. In: *2017 IEEE Conference on Computational Intelligence and Games (CIG)*. 2017, S. 88–95. DOI: 10.1109/CIG.2017.8080420.

[15] Ian Goodfellow, Yoshua Bengio und Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[16] David Ha und Jürgen Schmidhuber. „Recurrent World Models Facilitate Policy Evolution“. In: *Advances in Neural Information Processing Systems*. Hrsg. von S. Bengio u. a. Bd. 31. Curran Associates, Inc., 2018, S. 2450–2462.

[17] Juš Kocijan u. a. „Gaussian Process Model Based Predictive Control“. In: *Proceedings of the 2004 American Control Conference*. Bd. 3. IEEE. 2004, S. 2214–2219.

[18] Simon M Lucas u. a. „A Local Approach to Forward Model Learning: Results on the Game of Life Game“. In: *2019 IEEE Conference on Games (CoG)*. IEEE, Aug. 2019, S. 1–8. DOI: 10.1109/CIG.2019.8848002. arXiv: 1903.12508v1.

[19] Volodymyr Mnih u. a. „Human-level control through deep reinforcement learning“. In: *Nature* 518.7540 (Feb. 2015), S. 529–533. DOI: 10.1038/nature14236. arXiv: 1312.5602.

[20] Duy Nguyen-Tuong und Jan Peters. „Model Learning for Robot Control: A Survey“. In: *Cognitive Processing* 12.4 (2011), S. 319–340. DOI: 10.1007/s10339-011-0404-1.

[21] Diego Perez-Liebana u. a. „General Video Game AI: Competition, Challenges and Opportunities“. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Bd. 30. 1. 2016.

[22] Diego Perez-Liebana u. a. *General Video Game Artificial Intelligence*. Bd. 3. 2. https://gaigresearch.github.io/gvgaibook/. Morgan & Claypool Publishers, 2019, S. 1–191.

[23] Aske Plaat, Walter Kosters und Mike Preuss. *Deep Model-based Reinforcement Learning for High-dimensional Problems, a Survey*. 2020. arXiv: 2008.05598 [cs.LG].

[24] Athanasios S Polydoros und Lazaros Nalpantidis. „Survey of Model-based Reinforcement Learning: Applications on Robotics“. In: *Journal of Intelligent & Robotic Systems* 86.2 (2017), S. 153–173.

[25] Adam Santoro u. a. „One-shot Learning with Memory-Augmented Neural Networks“. In: *CoRR* abs/1605.06065 (2016). arXiv: 1605.06065.

[26] Andrew W Senior u. a. „Improved Protein Structure Prediction Using Potentials from Deep Learning“. In: *Nature* 577.7792 (2020), S. 706–710. DOI: 10.1038/s41586-019-1923-7.

[27] David Silver u. a. „Mastering the Game of Go with Deep Neural Networks and Tree Search“. In: *Nature* 529.7587 (Jan. 2016), S. 484–489. DOI: 10.1038/nature16961.

[28] Richard S. Sutton und Andrew G. Barto. *Reinforcement Learning*. 2. Aufl. Cambridge: The MIT Press, 2018. ISBN: 9780262039246.

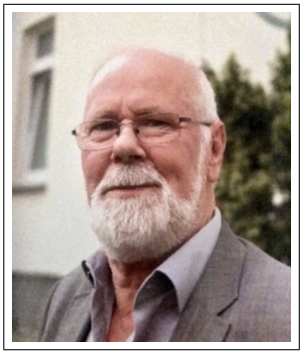
[29] István Szita und András Lorincz. „Learning Tetris Using the Noisy Cross-Entropy Method“. In: *Neural Computation* 18.12 (2006), S. 2936–2941. DOI: 10.1162/neco.2006.18.12.2936.

[30] Yan Wu und Yiannis Demiris. „Towards One Shot Learning by Imitation for Humanoid Robots“. In: *2010 IEEE International Conference on Robotics and Automation*. IEEE. 2010, S. 2889–2894.

[31] Jie Zhou u. a. *Graph Neural Networks: A Review of Methods and Applications*. 2021. arXiv: 1812.08434 [cs.LG].

Alexander Dockhorn ist Postdoctoral Research Associate an der Queen Mary University of London. Seinen Doktortitel erhielt er an der Otto-von-Guericke-Universität in Magdeburg. In seiner Forschung untersucht er die Fähigkeiten von prädiktionsbasierten Suchagenten in Spielen mit einem besonderen Interesse zur Modellierung von Unsicherheiten. Eine vollständige Liste seiner Projekte und Veröffentlichungen finden Sie auf seiner Webseite:

https://adockhorn.github.io/



Rudolf Kruse ist emeritierter Professor für Informatik an der Otto-von-Guericke-Universität Magdeburg. Er promovierte und habilitierte 1980 bzw. 1984 in Mathematik an der Technischen Universität Braunschweig. Nach einem Aufenthalt bei der Fraunhofer Gesellschaft kam er 1986 als Professor für Informatik an die Technische Universität Braunschweig. Von 1996-2017 war er Leiter der Arbeitsgruppe Computational Intelligence in Magdeburg. Weitere Informationen finden sie unter:

www.is.ovgu.de/Team/Rudolf+Kruse.html