Modellierung von Unsicherheit und Wahrnehmung für Lernende Systeme

Andreas Sedlmeier

Dissertation
an der Fakultät für Mathematik, Informatik und Statistik
der Ludwig–Maximilians–Universität München

vorgelegt von
Andreas Sedlmeier

Tag der Einreichung: 22. Juli 2022

Modellierung von Unsicherheit und Wahrnehmung für Lernende Systeme

Andreas Sedlmeier

Dissertation
an der Fakultät für Mathematik, Informatik und Statistik
der Ludwig–Maximilians–Universität München

vorgelegt von
Andreas Sedlmeier

1. Berichterstatter:	Prof. Dr. Claudia Linnhoff-Popien
2. Berichterstatter:	Prof. Dr. Stefan Fischer
Tag der Einreichung:	22. Juli 2022
Tag der Disputation:	11. Januar 2023

Eidesstattliche Versicherung

(siehe Promotionsordnung vom 12.07.11, § 8, Abs. 2 Pkt. 5)

Hiermit erkläre ich an Eides statt, dass die Dissertation von mir selbstständig, ohne unerlaubte Beihilfe angefertigt ist.

Freising, 15. Februar 2023 Andreas Sedlmeier

Zusammenfassung

Durch die enormen Fortschritte im Bereich der künstlichen Intelligenz (KI) erreicht eine Vielzahl lernender Systeme den Alltag von Menschen. Aktuell agiert zwar der Großteil dieser lernenden, KI-basierten Systeme noch für den Endnutzer meist unsichtbar, rein online, (beispielsweise als Recommender-System in Online-Shops), doch erste Systeme wie selbstfahrende Autos und autonome Lieferroboter sind bereits in die physische Welt vorgedrungen. Ebenso wird mit Hochdruck an Visionen wie der vollautomatisierten Fabrik (Smart Factory) oder an adaptiven Stromnetzen (Smart Grids) gearbeitet. Spätestens mit der Präsenz der lernenden Systeme in der physischen Welt gewinnen Fragen der Zuverlässigkeit und Sicherheit höchste Relevanz. Aktuell existieren jedoch kaum Verfahren, die die Grenzen und stets vorhandenen Unsicherheiten der lernenden Systeme zuverlässig erfassen können. Ebenso wäre die Modellierung eines gemeinsamen Wahrnehmungsverständnisses zwischen Mensch und Maschine von großer Wichtigkeit, um Missverständnisse und Fehler in der Interaktion zu minimieren. In der vorliegenden Arbeit werden Ansätze vorgestellt, die eine solche Modellierung von Unsicherheit und Wahrnehmung, insbesondere in Kombination mit aktuellen Deep Learning Verfahren ermöglichen.
Im ersten Teil dieser Arbeit werden Ansätze zur Modellierung eines räumlichen Wahrnehmungsverständnisses für lernende Systeme behandelt. Hierfür wird eine Isovisten-basierte Quantifizierung der Wahrnehmung mit Machine Learning (ML) basierter Modellierung kombiniert. Evaluationsergebnisse zeigen, dass die so entwickelte Modellierung in der Lage ist, semantische Strukturen abzubilden. Eine in einem zweiten Schritt mit Bayesschen Verfahren erweiterte, probabilistische Wahrnehmungsmodellierung wird anschließend unter anderem zur Charakterisierung von Routen eingesetzt. Da besonders im Forschungsfeld autonom handelnder Systeme aktuell Reinforcement Learning (RL) Ansätze dominieren, stehen im zweiten Teil der Arbeit Methoden zur Unsicherheitsmodellierung in Value-basiertem Deep RL im Fokus. Es werden sowohl Methoden der approximativen Bayesschen Inferenz, als auch ensemble-basierte Verfahren dahingehend evaluiert, ob sie zuverlässig in der Lage sind, epistemische Unsicherheit zu modellieren. Da die zuvor vorgestellten Ansätze nicht direkt mit Policy-basiertem RL kombinierbar sind, wird im dritten Teil der Arbeit ein neu entwickelter, auf der Entropie der Policy basierender Ansatz zur Erkennung von Out-of-Distribution (OOD) Situationen vorgestellt.
Zusammengenommen schaffen die im Rahmen der vorliegenden Arbeit vorgestellten Modellierungsansätze von Unsicherheit und Wahrnehmung eine Grundlage zur Entwicklung zuverlässiger, sicherer, lernender Systeme.

Abstract

With the enormous progress in the field of artificial intelligence (AI), a plethora of learning systems is becoming an integral part of people's everyday lives. Currently, most of these learning, AI-based systems are restricted to virtual spaces (for example as recommender systems in online shops). But the first ones like self-driving cars or autonomous delivery robots have already entered the physical world. Work is also progressing to realize visions like fully automated industrial plants (Smart Factory) or adaptive electrical grids (Smart Grid). With this increasing presence of learning systems in the physical world, questions of reliability and safety are now of utmost importance. Currently, however, hardly any methods exist that are able to reliably capture the limitations and ever-present uncertainties of learning systems. Likewise, modeling a common understanding of perception between humans and machines is of great importance, to minimize misunderstandings and errors in their interaction. The work at hand presents approaches which allow this kind of modeling of uncertainty and perception, especially when combined with current Deep Learning methods.

The first part of this thesis addresses approaches for modeling a spatial perception for learning systems. For this purpose, an Isovist-based quantification of perception is combined with Machine Learning (ML) based modeling. Evaluation results show that the developed modeling approaches are capable of representing semantic structures. A probabilistic modeling approach, extended using Bayesian methods, is subsequently presented. It is used, among other things, for an uncertainty based characterization of trajectories. For the development of autonomous systems, Reinforcement Learning (RL) approaches currently dominate. Consequently, the second part of this thesis focuses on methods for uncertainty modeling in Value-based Deep RL. Methods of approximate Bayesian Inference as well as Ensemble-based approaches are evaluated, regarding their ability to reliability model epistemic uncertainty. As the previous approaches are not directly applicable to Policy-based RL, the third part of this thesis presents a newly developed policy entropy based approach for detecting out-of-distribution (OOD) situations.

Together, the approaches to modeling uncertainty and perception, presented in the context of the work at hand, provide a foundation for the future development of reliable, safe, learning systems.

Inhaltsverzeichnis

Danksagung

Ich möchte mich an dieser Stelle ganz herzlich bei einigen Personen bedanken, die mich in meiner Zeit am Lehrstuhl für Mobile und Verteilte Systeme an der Ludwig-Maximilians-Universität München begleitet haben.
Zunächst möchte ich Frau Prof. Dr. Claudia Linnhoff-Popien einen ganz besonderen Dank aussprechen, für die stets vertrauensvolle Zusammenarbeit und gegenseitige Wertschätzung über viele Jahre hinweg. Ich konnte durch die Möglichkeiten und Chancen, die sie mir eröffnet hat meinen Forschungsinteressen folgen und stets neue Ziele erreichen. Ihr Lehrstuhl bot mir dabei ein außergewöhnlich inspirierendes, wertschätzendes Umfeld. Herzlichen Dank dafür, dass ich ein Teil dieses wunderbaren Teams sein konnte.
Des Weiteren gilt mein herzlicher Dank Herrn Prof. Dr. Stefan Fischer, der sich freundlicherweise bereit erklärte, die Rolle des Zweitberichterstatters zu übernehmen. Ebenso danke ich Herrn Prof. Dr. Andreas Butz für die Übernahme des Vorsitzes der Prüfungskommission, sowie Herrn Prof. Dr. Albrecht Schmidt für seine Verfügbarkeit als Ersatzprüfer.
Ein großes Dankeschön geht an alle meine Kollegen und Freunde am Lehrstuhl. Ein Team wie dieses ist etwas Besonderes und wird mir immer in Erinnerung bleiben. Jeder Einzelne war in guten, wie in schwierigen Momenten immer bestrebt für Alle etwas Positives zu erreichen.
Namentlich erwähnen möchte ich Dr. Sebastian Feld, der mir zu Beginn meiner Zeit am Lehrstuhl eine besondere Unterstützung war, egal ob beim Schreiben gemeinsamer Paper bis spät in die Nacht, oder bei langen Gesprächen zur Situation am Lehrstuhl. Des Weiteren geht ein besonderer Dank an Prof. Dr. Lenz Belzner, der mir den Weg in die KI-Forschung eröffnet hat. Durch seine Begeisterung für das Thema, motivierende Unterstützung und kritische Nachfragen trug er einen entscheidenden Anteil zum Gelingen meiner Forschungsvorhaben bei.
Danken möchte ich auch meiner Familie, für die so entscheidend wertvolle Grundgelassenheit und Überzeugtheit, dass ich erreichen kann, was ich mir wünsche. Zuletzt geht mein Dank an meine Frau Regine, für ihr liebevolles Verständnis und ihre Geduld auch in arbeitsintensiven Phasen.

1 Einleitung

> Le dout n'est pas un état bien agréable, mais l'assurance est un état ridicule.
>
> Unsicherheit ist kein angenehmer Zustand, aber Gewissheit ein absurder.
>
> Voltaire, 1817 [105]

Diese zeitlose Erkenntnis bezog Voltaire 1817 selbstverständlich nicht auf lernende Maschinen, sondern auf das stets begrenzte Wissen von uns Menschen über die Realität. Und dennoch ist die gedankliche Brücke zu lernenden, von Menschen entwickelten Systemen, nicht weit. Spätestens in dem Moment, in dem diese Systeme nicht mehr nur im rein virtuellen Raum, sondern in der realen Welt agieren, sind sie auch den Effekten der Realität unterworfen. Hier ist die Möglichkeit die Realität zu erfassen stets begrenzt. Sensoren beispielsweise besitzen in der Regel nur eine begrenzte Auflösung und enthalten Rauschen. Die Folge ist omnipräsente Unsicherheit – ist das, was beobachtet und gemessen wurde, tatsächlich gewiss? Diese Unsicherheit endet auch nicht in den erfassten Daten, sondern setzt sich von dort aus über alle Aspekte des Lernens fort. Denn diese Beobachtungen, also aus dem Blickwinkel der Informatik – die erfasste Daten – bilden die Grundlage des maschinellen Lernens. Dies ist der Fokus des Forschungsbereiches der lernenden Systeme. Die dort entwickelten Technologien haben in den vergangenen Jahren einen enormen Fortschritt erfahren, insbesondere im Bereich des Machine Learning (ML) und der Künstlichen Intelligenz (KI). Der Einsatz von ML-Verfahren, und insbesondere Deep Learning basierter Verfahren, stellt aktuell den State-of-the-Art Ansatz zur Entwicklung lernender und autonom agierender Systeme dar. Diese ML-basierten Systeme enthalten keine klassisch vorprogrammierten Lösungen, sondern lernen statistische Modelle aus gesammelten Trainingsdaten.
Die in diesen Bereichen gewonnenen Erkenntnisse sind seit einigen Jahren nicht mehr auf den akademischen Bereich beschränkt, sondern durchziehen immer mehr Lebensbereiche. So ist beispielsweise nahezu jedes aktuelle Smartphone in der Lage, mittels Spracherkennung gesprochene Eingaben in Text umzuwandeln (Automatic Speech Recognition). Dies geschieht meist in einer Präzision, die noch vor wenigen Jahren als undenkbar galt [60, 142]. Ebenso übertreffen Deep Learning basierte Lösungen zur automatischen Sprachübersetzung [70] oder textbasierten Sprachgenerierung (Text-to-Speech) [131] sämtliche früheren, nicht ML-basierten Ansätze. Gleiches gilt auch für den Bereich der Com-

puter Vision. Prominentestes Beispiel sind hier die unter anderem im Bereich der Objekt- und Tiefenerkennung erzielten Fortschritte [74, 116, 117]. Diese bilden beispielsweise die Grundlage des rein kamerabasierten Ansatzes zur Entwicklung von autonomen Fahrzeugen, der aktuell vom Automobilhersteller Tesla verfolgt wird.
Es ist somit nicht verwunderlich, dass die Versprechen und Möglichkeiten, die diese Technologien mit sich bringen, sowohl in der öffentlichen Wahrnehmung, als auch in akademischen Kreisen, sehr präsent sind. Ein Bewusstsein über die stets existenten Grenzen und Schwächen dieser lernenden Systeme ist hingegen aktuell eher selten vorhanden [94]. Auch aus Forschungssicht sind die sogenannten *Non-functional requirements* (NFRs), also Aspekte wie Zuverlässigkeit und Sicherheit von lernenden Systemen, bisher eher vernachlässigt worden.
Doch genau diese Aspekte wie Zuverlässigkeit und Sicherheit besitzen eine hohe Relevanz, wenn die Vision lautet, lernende Systeme zu entwickeln, die autonom und gefahrlos in der echten Welt zwischen und mit Menschen interagieren.
Gerade auch aus Sicht der Industrie ist dieses Ziel von hoher Relevanz, da hier eine Vielzahl an Problemstellungen nur mehr durch lernende Systeme bewältigt werden kann. Für einen produktiven, sicheren Einsatz dieser Technologien ist es allerdings erforderlich, ein gemeinsames Wahrnehmungsverständnis und Bewusstsein über Unsicherheiten zwischen Mensch und Maschine zu schaffen. Nur so kann eine intuitive Interaktion und sichere Kooperation erreicht werden, die Missverständnisse und Fehler minimiert. Das Fehlen von Ansätzen zur Lösung dieser Herausforderungen ist eine der zentralen Ursachen, weshalb (besonders autonome) lernende Systeme im industriellen Umfeld großteils noch nicht in der Produktion angekommen sind. Hier befinden sich die meisten Ansätze noch in aktiver Forschung [49, 110, 118]. Abbildung 1.1 zeigt das Beispiel einer simulierten autonomen Produktfertigungsanlage (Smart Factory), wie sie in [110] zu Forschungszwecken eingesetzt wurde.
Es ist anzunehmen, dass dieses Ziel nicht immer zu erreichen ist, auch zwischen Menschen treten Missverständnisse und misslungene Kooperation auf. Doch genau deshalb ist es von zentraler Wichtigkeit, Unsicherheiten in allen Bereichen zu betrachten und lernende Systeme zu entwickeln, die sich dieser Unsicherheiten bewusst sind. Hierfür ist es notwendig anzuerkennen, dass jedes gelernte Modell Grenzen der Gültigkeit besitzt. Diese Tatsache hat George Box prominent in seinem bekannten Aphorismus zusammengefasst [19]:

> [...] all models are approximations. Essentially, all models are wrong, but some are useful. However, the approximate nature of the model must always be borne in mind [...].

Diesem Gedankengang folgend ist ein sicherer Einsatz (modellbasierter) lernender Systeme nur möglich, wenn die Grenzen derselben bekannt sind und beachtet werden. Dies ist eine der zentralen Herausforderungen im Forschungsbereich der AI Safety [5].

Abbildung 1.1: Simulation einer automatisierten Produktfertigungsanlage (Smart Factory) mit 16 autonomen, lernenden Einheiten (Zylinder).

Die meisten der aktuell verwendeten Architekturen, besonders die meisten tiefen Neuronalen Netze, sind jedoch nicht in der Lage, Aussagen zur Verlässlichkeit ihrer Vorhersagen zu liefern: Ihre Ausgaben bestehen meist lediglich aus sogenannten Punktprädiktionen. Hier schließt sich der Bogen zum Eingangszitat: Solche definitiven Punktprädiktionen sollten nicht unreflektiert akzeptiert werden. Unsicherheit ist stets vorhanden, und deren Betrachtung lohnt sich – auch wenn dies zusätzlichen Aufwand verursacht.
Es lässt sich also feststellen, dass aktuell eine Diskrepanz zwischen den Versprechungen der KI-Methoden und deren Einsatzbereitschaft in der echten Welt existiert. Es werden Ansätze und Lösungen benötigt, die die Grenzen dieser lernenden Systeme erfassen können, um potentielle Fehler und Schaden zu verhindern.
Die aktuelle Herausforderung ist es folglich, einerseits Wahrnehmungsmodellierungen zu finden, die die Realität geeignet abbilden, und es andererseits ermöglichen, die Ansätze mit probabilistischen Techniken, beispielsweise der Bayesschen Statistik zu verbinden. Diese Modellierung von Unsicherheit und Wahrnehmung kann es sowohl uns Menschen – als auch den Systemen selbst – ermöglichen, die Grenzen und damit die Verlässlichkeit der Vorhersagen zu erkennen [94]. Diese Problemstellung bildet den Rahmen für die in der vorliegenden Arbeit behandelten Themenkomplexe.

1.1 Motivation und Beiträge der Arbeit

Wie bereits angesprochen, versprechen intelligente, lernende Systeme auch im industriellen Umfeld, einen wertvollen Beitrag in vielen Bereichen zu leisten. Ihre Fähigkeit zur Optimierung von Prozessen und Anpassungsfähigkeit an neue Gegebenheiten ermöglicht unter anderem steigende Kosten- und Energie-

effizienz. Beispielsweise könnten sich in einem Smart Factory Anwendungsfall mehrere lernende Einheiten autonom koordinieren [110, 111, 136]. Auf diese Weise kann die Routenführung und Bearbeitung von Bauteilen durch die Fabrik optimiert werden, selbst wenn laufend neue Kundenanforderungen eintreffen oder Maschinen temporär ausfallen. Diese Fähigkeit zur dynamischen Anpassung bietet das Versprechen einer flexiblen, individualisierten industriellen Produktfertigung gerecht zu werden. Ein weiteres industrielles Beispiel wäre die Entwicklung von Smart Grids, also intelligenter Stromnetze. In diesen entscheiden eine Vielzahl an Teilnehmern (beispielsweise jeder mittels Photovoltaik stromproduzierende Haushalt) autonom, ob Elektrizität in das Netz eingespeist, oder entnommen werden soll. Diese Entscheidungen werden lokal getroffen und müssen sich an verschiedenste Bedingungen (Wetter, Kosten, Regularien, ...), neue Teilnehmer oder ausgefallene Verbindungen anpassen. Ebenso finden sich im Bereich der Medizinforschung eine Vielzahl aktueller Ansätze, die die mittels KI-Techniken erzielten Fortschritte zu nutzen versuchen. Besonders im Bereich der medizinischen Bildgebung, beispielsweise zur Krebserkennung [92] oder der Erkennung von Augenerkrankungen [84, 147], werden Ansätze erforscht, einen Teil der bisher von Ärzten durchgeführten Diagnose auf KI-Systeme zu verlagern.

Was all diese stark unterschiedlichen Anwendungsbereiche verbindet, ist die Tatsache, dass die Problemstellungen nur durch hochdimensionalen Zustandsbeschreibungen (Input-Features) erfasst werden können. Beispielsweise durch Röntgenbilder in medizinischen Anwendungen, hochauflösende Kamerasysteme in Smart Factories oder eine Vielzahl an Sensorinformationen in einem Smart Grid. Diese Komplexität der Zustandsbeschreibungen führt zu einer Vielzahl an Schwierigkeiten (auch Curse-of-dimensionality [11] genannt), unter anderem zu einer kombinatorischen Explosion an möglichen Systemzuständen. Folglich ist es in solchen Systemen zum einen nicht mehr möglich, alle Lösungen „klassisch vorzuprogrammieren“, zum anderen sind die optimalen Lösungen selbst Domänenexperten oftmals nicht vorab bekannt. Folglich ist die aktuell herrschende Wahrnehmung, dass die Komplexität dieser Problemstellungen nur mehr durch lernende Systeme bewältigt werden kann. Der Einsatz von ML-Verfahren, und insbesondere Deep Learning basierten Verfahren, stellt hier den State-of-the-Art Ansatz zur Entwicklung lernender und autonom agierender Systeme dar. Diese ML-basierten Systeme lernen statistische Modelle basierend auf Trainingsdaten.

Wie eingangs erwähnt, bringen diese lernenden Systeme jedoch auch eine Vielzahl neuer Herausforderungen mit sich [5]. Es existieren beispielsweise aktuell kaum Ansätze, die Verlässlichkeit der leistungsfähigsten, auf tiefen Neuronalen Netzen basierenden Systeme systematisch zu verifizieren [32, 118]. Die Modellierung von Unsicherheit kann als paralleler Ansatz dazu verstanden werden. Wenn die Systeme in die Lage versetzt werden können, die Unsicherheit in ihren Vorhersagen und den darauf basierenden Handlungen abzubilden, kann auf diesem Weg die Verlässlichkeit erhöht werden.

Auf dieser Wahrnehmung aufbauend, ist es das Ziel der vorliegenden Arbeit einen Beitrag für einen verlässlichen und sicheren Einsatz von lernenden Systemen zu leisten. Hierfür werden im Rahmen dieser Arbeit im Wesentlichen drei Lösungsansätze präsentiert. Diese greifen jeweils unterschiedliche Aspekte der Problemstellung auf.

Der erste, in Kapitel 3 präsentierte Ansatz stellt die Modellierung eines gemeinsamen Wahrnehmungsverständnisses zwischen Mensch und Maschine in den Fokus. Es werden Verfahren vorgestellt, die es lernenden Systemen ermöglichen, die Wahrnehmung von Raum zu quantifizieren und in Form von ML-Modellen nutzbar zu machen. Hierbei wird zum einen auf den Aspekt der Abbildung semantisch nachvollziehbarer, menschlicher Konzepte eingegangen. Zum anderen wird gezeigt, wie durch eine probabilistische Modellierung dieser Wahrnehmung *Überraschung* als zusätzliche Informationsquelle nutzbar gemacht werden kann.

Im zweiten, in Kapitel 4 behandelten Ansatz steht die Modellierung von Unsicherheit in Value-basiertem Deep Reinforcement Learning (RL) im Vordergrund. RL-basierte Problemformulierungen dominieren aktuell die Ansätze zur Lösung sequentieller Entscheidungsprobleme. Eine direkte Übertragung der im vorherigen Kapitel verwendeten exakten Bayesschen Inferenz ist jedoch in die im RL meist Deep Learning basierten Architekturen nicht möglich. Eine Modellierung von Unsicherheit ist jedoch besonders in diesen Problemstellungen von zentraler Relevanz, da meist nicht garantiert werden kann, dass alle möglichen Situationen, in denen sich das System befinden kann, auch im Training berücksichtigt wurden. Als Lösungsansatz wurden im Rahmen dieser Arbeit Methoden der approximativen Bayesschen Inferenz sowie Ensemble-basierte Verfahren für einen Einsatz im Deep RL erweitert und evaluiert. Zentrale Fragestellung ist, ob diese Methoden im Kontext von RL zuverlässig in der Lage sind, epistemische Unsicherheit zu erkennen.

Der Dritte, in Kapitel 5 präsentierte Ansatz, präsentiert eine alternative Methode der Unsicherheitsmodellierung, die mit Policy-basierten RL-Verfahren kombinierbar ist. Dies ist von zentraler Relevanz, da Policy-basierte RL-Verfahren besonders im Bereich der Robotik aktuelle State-of-the-Art Lösungen liefern [54, 68]. Im Fokus steht die Frage, ob diese Art der Unsicherheitsmodellierung zur zuverlässigen Erkennung von untrainierten Situationen, und damit der Bestimmung der Grenzen der Modelle, eingesetzt werden kann.

Zusammengefasst wurden im Rahmen der vorliegenden Arbeit Verfahren entwickelt, die es lernenden Systemen ermöglichen, Unsicherheit zu erfassen und damit nutzbar zu machen. Dies ist aktuell durch das Eintreten lernender Systeme in die physische Umwelt der Menschen von besonderer Relevanz. Die Modellierung von Unsicherheit stellt hierbei eine zusätzliche Informationsquelle dar. Diese kann sowohl zur Realisierung neuer Anwendungen genutzt werden, wie einer unsicherheitsbasierten Identifikation charakteristischer Positionen im Raum, als auch zur Verbesserung der Zuverlässigkeit und Sicherheit lernender Systeme, beispielsweise durch eine Erkennung untrainierter Situationen.

1.2 Vorveröffentlichungen

Die Inhalte der vorliegenden Arbeit wurden zu großen Teilen bereits auf internationalen wissenschaftlichen Konferenzen sowie als Journal-Beitrag publiziert. Im Folgenden wird ein Überblick über diese Veröffentlichungen gegeben sowie der Anteil des Autors der vorliegenden Arbeit herausgestellt. In den folgenden Kapiteln wird zudem an entsprechenden Stellen erneut auf die Inhalte der zugehörigen Vorveröffentlichungen verwiesen.
Grundsätzlich gilt, dass Prof. Dr. Claudia Linnhoff-Popien als Lehrstuhlinhaberin des Autors dieser Arbeit bei allen Vorarbeiten und insbesondere bei den Vorveröffentlichungen in denen sie als Co-Autorin aufgeführt ist, beratend mitgewirkt hat. Die Zuordnung der Beteiligung der sonstigen Autoren der jeweiligen Paper wird im Folgenden dargelegt.

Discovering and Learning Recurring Structures in Building Floor Plans [124] Diese Arbeit stellt ein Framework zur Erzeugung und Analyse wiederkehrender Strukturen der Raumwahrnehmung dar. Es ermöglicht die Erzeugung und Analyse eines Datensets, bestehend aus 2D-Isovistenmessungen entlang geospatialer Trajektorien. Deren Erfassung erfolgt innerhalb einer 3D-Simulationsumgebung, basierend auf realen Gebäudeplänen. Die Analyse-Ergebnisse zeigen, dass diese Isovistenmessungen die wiederkehrenden Strukturen der Raumwahrnehmung so enkodieren, dass Unsupervised Machine Learning Techniken in der Lage sind darin semantische Strukturen zu identifizieren. Dr. Sebastian Feld lieferte den ursprünglichen gedanklichen Anstoß, Isovistenmessungen zur Identifikation von semantischen Strukturen in Gebäudeplänen zu untersuchen. Zudem unterstützte er insbesondere bei der Aufbereitung der Features (Abschnitt 2.1 des Papers) sowie der Einordnung in das Themengebiet und der Abgrenzung zu verwandten Arbeiten (Abschnitt 2.3). Die Hauptinhalte des Papers: die Realisierung mittels einer Unity Simulationsumgebung sowie der Einsatz von Unsupervised und Deep Learning basierten Supervised ML-Techniken, stammen sowohl im Konzept, als auch in der Implementierung und Evaluation vom Autor der vorliegenden Arbeit. Die Inhalte dieser Veröffentlichung bilden den ersten Teil des in Kapitel 3 vorgestellten Ansatzes.

Learning indoor space perception [125] Diese Journal-Publikation stellt eine tiefergehende Analyse sowie Weiterentwicklung des in [124] vorgestellten Ansatzes dar. Sie evaluiert das Konzept auf zwei weiteren auf realen Gebäudeplänen basierenden Umgebungen. Zudem erfolgt eine detaillierte Analyse der Funktionsweise des Ansatzes zusätzlich zur kartenbasierten Betrachtung auch im PCA Feature-Raum. Des Weiteren wird ein Konzept vorgeschlagen und evaluiert, die Ähnlichkeit der visuellen Wahrnehmung unterschiedlicher Umgebungen mittels Cluster Similarity Analysis zu quantifizieren. Auch in dieser Arbeit lieferte Dr. Sebastian Feld Beiträge zu den Grundlagen sowie zur Verortung im Forschungsgebiet. Die Kerninhalte der Publikation, ebenso wie

die gesamte Implementierung und Evaluation der Ergebnisse wurden durch den Autor der vorliegenden Arbeit realisiert. Dies beinhaltet insbesondere das Konzept sowie die Umsetzung der Analyse im PCA Feature-Raum, sowie die Identifikation der Möglichkeit der Verwendung von Methoden der Cluster Similarity Analysis zur Quantifizierung der Ähnlichkeit von Wahrnehmung. Die Analysen zur Funktionsweise des Ansatzes im PCA Feature-Raum bilden den zweiten Teil von Kapitel 3.

Bayesian Surprise in Indoor Environments [45] In dieser Arbeit wird eine neuartige Methode zur Identifikation von unerwarteten Strukturen der Raumwahrnehmung vorgestellt. Diese basiert auf dem Konzept des Bayesian Surprise, das den Erwartungen eines Betrachters eine wichtige Rolle in der Wahrnehmung zuweist. Die Arbeit baut auf den in [125] sowie [124] entwickelten Ansätzen zur Generierung und Quantifizierung von Wahrnehmung mittels Isovisten auf. Diese werden hier aber als Datengrundlage für eine Wahrnehmungsmodellierung mittels Bayesian Surprise zur Charakterisierung von Trajektorien verwendet. Prof. Dr. Lenz Belzner unterstütze diese Arbeit beratend, insbesondere zu Fragen der Bayesschen Statistik und Modellierung. Jan Franz half als Master-Student bei der Implementierung der auf synthetischen Gebäudegrundrissen basierenden Evaluationsumgebungen. Markus Friedrich unterstützte beratend in der Konzipierung dieser Umgebungen. Dr. Sebastian Feld lieferte die ursprüngliche Idee, Bayesian Surprise mit Isovisten-basierter Wahrnehmung zu kombinieren. Von ihm stammt zudem das Konzept, die so quantifizierte Überraschung als neue Dimension des Kontexts, beispielsweise zur Identifikation von charakteristischen Positionen, zu nutzen. Die gesamte im Rahmen der Arbeit entwickelte Modellierung (sowohl im Konzept, als auch der Implementierung) stammt vom Autor der vorliegenden Arbeit. Dies beinhaltet im Speziellen den Einsatz von separaten Multinomialverteilungen zur Featuremodellierung, als auch die Verwendung von Conjugate Priors zur effizienten Berechnung des Bayesschen Posteriors. Des Weiteren wurde auch die Evaluation auf sämtlichen Umgebungen vom Autor der vorliegenden Arbeit durchgeführt. Die Inhalte dieser Publikation finden sich insbesondere im dritten Teil von Kapitel 3.

Uncertainty-Based Out-of-Distribution Detection in Deep Reinforcement Learning [126] Der Fokus dieser Veröffentlichung ist eine Analyse von Methoden zur Quantifizierung von Unsicherheit in Value-basiertem Deep RL. Hierfür werden sowohl Methoden der approximativen Bayesschen Inferenz, wie Monte-Carlo Dropout, als auch Ensemble-basierte Verfahren evaluiert. Zentrale Fragestellung ist, ob diese Methoden zuverlässig in der Lage sind epistemische Unsicherheit zu erkennen, um basierend darauf abweichende Daten erkennen zu können. Prof. Dr. Lenz Belzner unterstütze diese Arbeit beratend, insbesondere zu Fragen der Unterscheidung von aleatorischer und epistemischer Unsicherheit. Thomy Phan lieferte Beiträge bei der Implemen-

tierung der RL-Algorithmen und Evaluationsumgebungen. Dr. Thomas Gabor unterstütze bei der Interpretation und Diskussion der Evaluationsergebnisse. Die grundlegende Idee, das Konzept sowie die Umsetzung und Evaluation des Ansatzes stammen vom Autor dieser Arbeit. Dies beinhaltet sowohl den Ansatz zur Quantifizierung von epistemischer Unsicherheit, basierend auf Value-Funktionen, als auch das Design und die Realisierung geeigneter Evaluationsumgebungen. Diese Inhalte bilden die Grundlage von Kapitel 4.

Uncertainty-based Out-of-Distribution Classification in Deep Reinforcement Learning [127] Diese Publikation entwickelt die in [126] publizierten Ideen weiter. Kernelement ist die unsicherheitsbasierte Modellierung eines Out-of-Distribution (OOD) Erkennungsproblems als One-Class Klassifikationsproblem. Das vorgeschlagene Framework UBOOD (Uncertainty-Based OOD) kann dabei mit verschiedenen Verfahren zur epistemischen Unsicherheitsmodellierung kombiniert werden. Die Evaluationsergebnisse zeigen, dass die epistemische Unsicherheit der Value-Funktion eines RL-Agenten ein zuverlässiges Signal zur OOD Erkennung bietet. Prof. Dr. Lenz Belzner trug beratend, insbesondere zu Fragen der Unsicherheitsmodellierung bei. Thomy Phan unterstütze in der Konzipierung sowie der Implementierung der eingesetzten RL-Algorithmen und Umgebungen. In verschiedenen Diskussionen zum Verhalten von epistemischer Unsicherheit sowie zur Interpretation der Ergebnisse lieferte Dr. Thomas Gabor wertvolle Beiträge. Sowohl die Kernidee der Veröffentlichung, das Konzept zur Unsicherheitsmodellierung, sowie das Design und die Durchführung der Evaluation stammen vom Autor der vorliegenden Arbeit. Diese bilden den Kern von Kapitel 4.

Policy Entropy for Out-of-Distribution Classification [129] In dieser Arbeit wird ein neuartiger Ansatz zur Policy-basierten Erkennung von OOD Situationen vorgestellt. Die PEOC (Policy Entropy Out-of-Distribution Classifier) genannte Methode ist dabei für einen Einsatz in Kombination mit Policy-basierten RL-Algorithmen entwickelt. Kernelement des Verfahrens ist die Verwendung der Entropie der Policy eines Agenten als Score eines One-Class Klassifikators. Zu dieser Arbeit trug Steffen Illium unterstützend in der Konzeption der Benchmarking Pipeline bei. Robert Müller lieferte Beiträge zur Einordnung im Forschungsbereich, insbesondere des Maximum Entropy RL. Die Hauptinhalte der Publikation, die Idee, Konzeption und Evaluation stammen vom Autor der vorliegenden Arbeit. Die Inhalte der Veröffentlichung bilden die zentralen Elemente von Kapitel 5.

1.3 Aufbau der Arbeit

Der weitere Aufbau der vorliegenden Arbeit ist wie folgt strukturiert: In Kapitel 2 werden notwendige Grundlagen zum Verständnis von Modellierung im

Kontext von lernenden Systemen vermittelt. Hierbei werden sowohl die verschiedenen Kategorien von Machine Learning präsentiert, als auch die Struktur und Funktionsweise von Neuronalen Netzen, der aktuell wichtigsten Technologie zur Realisierung von ML. Des Weiteren werden in diesem Kapitel Grundkonzepte zur Quantifizierung von Unsicherheit sowie deren verschiedene Ausprägungen behandelt. Auf dieses Grundlagenkapitel folgt in Kapitel 3 die Vorstellung eines entwickelten Ansatzes zur Wahrnehmungsmodellierung räumlicher Strukturen. Die Kombination mit Bayesschen Techniken zur Unsicherheitsmodellierung zeigt in einer ausführlichen Evaluation auf simulierten Umgebungen die Leistungsfähigkeit des vorgeschlagenen Ansatzes. In Kapitel 4 werden die Herausforderungen der Unsicherheitsmodellierung im Kontext aktueller RL Probleme detailliert betrachtet. Im Fokus steht hier die Fragestellung, wie die Modellierung von Unsicherheit zur Realisierung von Zuverlässigkeit dieser autonomen Systeme beitragen kann. Es werden entwickelte Lösungsansätze basierend auf approximativen Bayesschen Inferenzmethoden, sowie Modell-Ensembles vorgestellt und anhand von speziell angepassten Benchmarkumgebung untersucht. Kapitel 5 entwickelt die gewonnen Erkenntnisse der Unsicherheitsmodellierung in RL-Systemen weiter. Es wird eine neuartige Methode zur Policy-basierten Erkennung von Out-of-Distribution Situationen vorgestellt und gegen aktuelle State-of-the-Art Lösungsansätzen evaluiert.
Abschließend erfolgt in Kapitel 6 eine Rekapitulation der wichtigsten Erkenntnisse dieser Arbeit, sowie ein Ausblick auf noch offene Fragestellungen und zukünftige Forschungsrichtungen.

2 Definitionen und Grundlagen

Im Folgenden werden Grundlagen der Modellierung im Kontext von lernenden Systemen vermittelt. Hierbei wird zunächst ein Überblick über die Kernidee der Modellerstellung gegeben. Anschließend werden die verschiedenen Kategorien von Machine Learning (ML) präsentiert sowie die Struktur und Funktionsweise von Neuronalen Netzen erläutert. Darauf folgen Grundkonzepte zur Quantifizierung von Unsicherheit sowie deren verschiedene Ausprägungen.

Zu Beginn stellt sich die Frage, wie der Begriff *Modellierung* im Kontext dieser Arbeit zu verstehen ist. Als Modellierung verstehen wir die Erzeugung von *Modellen*, die es erlauben Vorhersagen oder Prognosen bezüglich eines Systems zu treffen. Beobachtungen dieses Systems erzeugen Daten, weshalb das System oft auch als *datengenerierender Prozess* bezeichnet wird. Modelle sind somit immer Abbildungen *von etwas*, erfassen in der Regel jedoch nicht alle Eigenschaften des Originals [52]. Folglich gibt es stets Grenzen, außerhalb derer die Prognosen des Modells nicht valide sind. Diese Grenzen zu kennen ist für einen sicheren Einsatz von Modellen unerlässlich, jedoch oftmals nicht einfach zu bewerkstelligen [94]. Modelle ermöglichen es also zum einen, Annahmen des Modellerstellers kompakt zu repräsentieren und zum anderen Hypothesen zu vergleichen. Weichen die zukünftig beobachteten Daten von den Modellvorhersagen ab, kann die durch das Modell repräsentierte Hypothese widerlegt werden. Ebenso kann das Modell durch Einbeziehen der neuen Daten angepasst werden, mit dem Ziel das beobachtete System besser abzubilden und zukünftige Prognosen zu verbessern. Diese Eigenschaft des *sich Verbesserns* stellt dabei die Brücke zu Lernenden Systemen dar. Die Fähigkeit neue Daten zu integrieren, um zukünftige Beobachtungen besser Vorhersagen zu können, ist ein zentraler Aspekt des Lernens [120, 137].
Es existieren verschiedene Ansätze zur Modellierung: Mathematische Modelle [13], Graphische Modelle[1] [16] , Statistische Modelle [30], oder auch Spieltheoretische Modelle [102, 103]. Die aktuell leistungsfähigsten, lernenden Systeme basieren auf statistischen Modellen, die durch ML-Verfahren erzeugt werden [22, 71, 133].
Kernelement der Statistik und damit auch statistischer Modelle ist die Wahrscheinlichkeitstheorie – und damit Aussagen zu Gewissheiten, beziehungsweise Sicherheiten und Unsicherheiten. Zoubin Ghahramani führt diesen Aspekt

[1] Die in Unterunterabschnitt 2.1.3.1 eingeführten Markow Entscheidungsprozesse können beispielsweise als Graphische Modelle betrachtet werden.

weiter und bezeichnet die Modellierung selbst als Prozess der Quantifizierung der Unsicherheit von möglichen Prognosen [52]. Die Hoffnung einer solchen Betrachtungsweise ist, die zuvor erwähnten Grenzen der Modelle, innerhalb derer die Prognosen valide sind, durch die Regeln der Wahrscheinlichkeitstheorie greifbar zu machen. Auf diese Ansätze des sogenannten probabilistischen Machine Learning wird in Abschnitt 4.4 detailliert eingegangen.
Im Folgenden werden die Grundlagen der Techniken vorgestellt, die in dieser Arbeit zur Erzeugung von Modellen verwendet werden. Der Fokus liegt hier auf ML und insbesondere Deep Learning Methoden. Details der im Rahmen der einzelnen in dieser Arbeit vorgestellten Ansätze benötigten Grundlagen werden in den Folgekapiteln eingeführt. Dies sind Grundlagen zur Modellierung von Wahrnehmung und Raum (Abschnitt 3.3), Methoden zur Modellierung von Unsicherheit im Rahmen von tiefen Neuronalen Netzen (Abschnitt 4.4) sowie Grundkonzepte zur Performance-Evaluation von Binären Scoring-Klassifikatoren (Abschnitt 5.3). Zunächst folgt nun aber eine Einführung in die Grundkonzepte von Machine Learning.

2.1 Machine Learning

> A computer program is said to learn from experience E with respect to some class of tasks T, and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.
>
> Thomas Mitchell, 1997 [93]

An der oben zitierten, wohl bekanntesten Definition von Machine Learning (ML) ist bereits erkennbar, dass die Art und Weise, wie ML zu betreiben ist, nicht strikt festgelegt ist. Je nachdem, wie die Aufgabe T und die Messgröße P beschaffen ist, und in welcher Form die Erfahrung E zur Verfügung steht, kann der Einsatz unterschiedlicher ML-Techniken sinnvoll sein. Grundsätzlich verbindet alle ML-Techniken jedoch die Eigenschaft, dass das Wissen wie die Aufgabe T bestmöglich zu lösen ist, nicht fest kodiert vorgegeben wird, sondern das benötigte Wissen autonom durch das Auffinden von Mustern in den Daten extrahiert werden muss. [57, 98]
Im Folgenden wird ein kurzer Überblick über die Grundlagen der drei Hauptkategorien von ML-Techniken gegeben: Supervised Machine Learning, Unsupervised Machine Learning sowie Reinforcement Learning.

2.1.1 Supervised Machine Learning

Eine der frühesten und gleichzeitig bekanntesten Problemstellungen aus dem Bereich des Supervised Machine Learning ist die Erkennung von handgeschriebenen Ziffern. Dies ist ein Problem, dessen Lösung beispielweise für Postdienst-

leister von hohem Interesse war, um automatisiert Briefe basierend auf handgeschriebenen Postleitzahlen sortieren zu können. Abbildung 2.1 zeigt Beispieldaten aus einem 1994 vom United States Postal Service (USPS) zu Forschungszwecken veröffentlichten Datensatz [62].

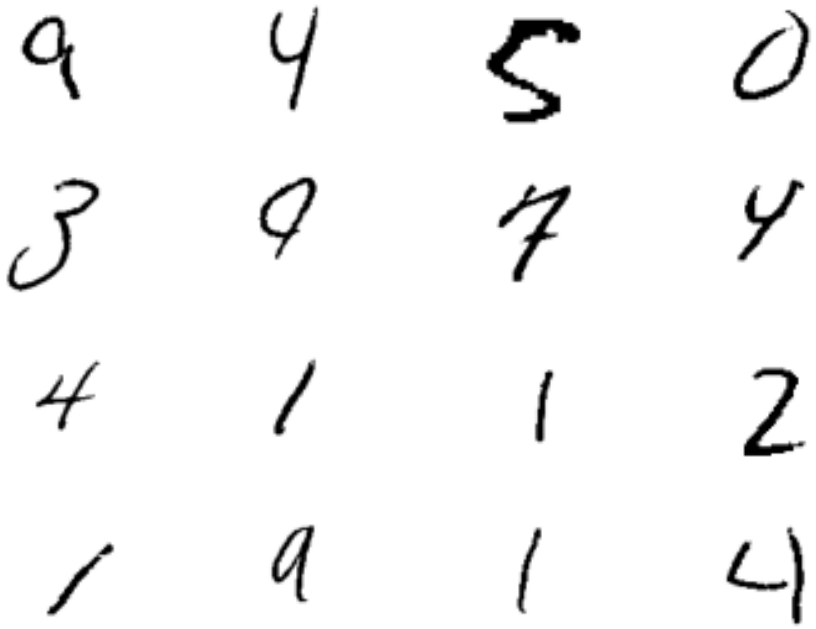


Abbildung 2.1: 4×4 Beispieldaten aus der *USPS Database for handwritten text recognition research* [62]. Jede handgeschriebene Ziffer liegt dabei als separate Bilddatei vor.

.

Aufgabe der Problemstellung ist die Erkennung der jeweiligen Ziffer, basierend auf den schwarz-weiß Bilddaten. Somit handelt es sich um ein Supervised Learning Problem, da der Datensatz sowohl die Eingabedaten (Bilder), als auch die Lösung des Problems (*Label* oder *Target* genannt) − die jeweilige zugehörige Ziffer − enthält. Diese Menge der Kombination aus Eingabedaten (*Input*) und Lösung (*Output*) wird als *Trainingsdaten* (oder *Trainingsset*) bezeichnet und stellt die *Erfahrung* E basierend auf der Definition nach Mitchell dar. Werden die Eingabedaten (Bilder) als realwertige Vektoren $x \in \mathbb{R}^D$ repräsentiert und die zugehörigen Ziffern als t, so ist das Ziel folglich das Finden einer Funktion $f(x) = t$. D ist dabei die Dimensionalität des Vektors, und jeder der D-Einträge eines einzelnen Vektors x wird dabei als *Feature* bezeichnet. Im oben genannten Beispiel würde die Größe der Eingabebilder (Breite und Höhe in Pixel) folglich die Dimension des Vektors festlegen: $x \in \mathbb{R}^{Breite \times Hoehe}$. Supervised Machine Learning Techniken verwenden nun ein parametrisierbares Modell $f_\theta(x)$ um diese Funktion zu repräsentieren. Während der *Trainingsphase* werden die Parameter θ unter Verwendung der Trainingsdaten optimiert.

Nach der Trainingsphase kann die Fähigkeit des Modells zur *Generalisierung* mit Hilfe von *Testdaten* (auch *Testset* genannt) untersucht werden. Hierbei wird untersucht, ob auch Daten die nicht Teil des Trainingssets waren vom Modell korrekt zugeordnet werden können.
Je nach Struktur der gewünschten Ausgabe y können die Problemtypen weiter unterschieden werden: Ist das Ziel wie im vorherigen Beispiel die Zuordnung einer abzählbaren Menge an *Kategorien* (die Ziffern $0-9$, auch *Klassen* genannt), wird von einem *Klassifikationsproblem* gesprochen. Ist y eine kontinuierliche Variable, so spricht man von einem *Regressionsproblem*. [16] Beispiel für ein Regressionsproblem könnte die Entwicklung eines Modells zur Vorhersage der Temperatur in Grad Celsius, basierend auf historischen Wetterdaten sein.
Sind zum Training des Modells nur Eingabedaten x, aber keine Lösungen t bekannt, so ist ein Training mittels Supervised Machine Learning nicht möglich. In diesem Fall spricht man von einem Unsupervised Learning Problem.

2.1.2 Unsupervised Machine Learning

Da bei Unsupervised Learning Problemen keine Lösungen y vorab spezifiziert werden, sind andere ML-Techniken als beim Supervised Machine Learning notwendig. Zudem unterscheidet sich meist die Zielsetzung, so dass basierend darauf drei Kategorien des Unsupervised Machine Learning unterschieden werden können.
Ist das Ziel eine automatische Einteilung der Daten x in Gruppen ähnlicher Struktur, so wird von *Clustering* gesprochen. Hier können erneut verschiedene Techniken unterschieden werden: Distanzbasierte Ansätze wie *k-means* [89], die die Daten basierend auf ihrer (beispielsweise quadratischen euklidischen) Distanz als Ähnlichkeitsmaß einteilen sowie dichtebasierte Ansätze wie DBSCAN [39], die Daten basierend auf ihrer Nachbarschaft strukturieren.
Ein davon abweichendes Ziel könnte es jedoch auch sein, eine Verteilungsfunktion $P(x)$ der Daten zu lernen, um beispielsweise die Wahrscheinlichkeit für das Auftreten eines bisher unbeobachteten Datenpunktes x' ermitteln zu können. Dies wird als *Density Estimation* bezeichnet. Interessant an diesen Modellen ist zudem ihre Fähigkeit auch neue Daten x' generieren zu können. Sie werden deshalb auch oft als generative Modelle bezeichnet.[2]
Ein drittes oftmals verfolgtes Ziel ist die sogenannte *Dimensionsreduktion*. Formal betrachtet ist das Problem hier das Lernen einer Funktion, die hochdimensionale Eingabedaten $x \in \mathbb{R}^D$ in niedrigdimensionale Repräsentationen $z \in \mathbb{R}^L$ abbildet. Wird hierfür ein parametrisches Modell verwendet ist die Aufgabe somit das Lernen der Funktion $f_\theta(x) = z$. Es existieren jedoch auch Verfahren, die keine generelle Funktion f erlernen, sondern ausschließlich für die Eingabedaten x_n das jeweilige z_n berechnen. Der wohl bekannteste Vertreter

[2] Aus dieser probabilistischen Perspektive betrachtet handelt es sich bei Unsupervised ML um das Lernen von unbedingten Modellen $P(x)$, während das Ziel von Supervised Learning die Erzeugung von bedingten Modellen der Form $P(y|x)$ ist. [98]

dieses Ansatzes zur Dimensionsreduktion ist die Principal Component Analysis (PCA) [109], die eine lineare, orthogonale Projektion der Eingabedaten in einen niedrigdimensionalen Subraum durchführt. Dieser niedrigdimensionale Raum kann anschließend besser visualisiert und analysiert werden als der hochdimensionale Ursprungsraum. PCA wählt dabei iterativ dasjenige Feature mit der höchsten Varianz, das orthogonal zur vorherig identifizierten Komponente steht. Auf diesem Weg werden die D ursprünglichen Feature durch $L < D$ unkorrelierte Linearkombinationen derselben ersetzt. Jedes der D ursprünglichen Feature trägt dabei zu einem unterschiedlichen Teil zu jeder der L neuen Feature (*Principal Components* genannt) bei. Anschließend ist es möglich, beispielsweise nur die beiden Feature auszuwählen und in 2D zu visualisieren, die die Varianz der Ursprungsdaten bestmöglich erklären.

2.1.3 Reinforcement Learning

Reinforcement Learning (RL) [137] unterscheidet sich von den zuvor vorgestellten Varianten des Machine Learning, da es sich primär auf *Lernen durch Interaktion* fokussiert. Kernelement ist somit die Existenz einer handelnden, lernenden Entität, die *Agent* genannt wird. Das Ziel dieses Agenten ist es in einer gegebenen Umgebung (*Environment* genannt) zu lernen, welche Aktionen (*Actions*) in einer gewissen Situation (repräsentiert durch den Zustand der Umgebung) optimal sind. Optimal bedeutet in diesem Zusammenhang, dass ein numerisches, *Reward* genanntes Signal über den Verlauf der Zeit maximiert wird. Hier wird bereits deutlich, dass es sich bei RL-Problemen um Probleme in einem zeitlichen Verlauf handelt und somit eine Abfolge von Zuständen und Aktionen betrachtet werden muss. Der Zustand der Umgebung (*State* genannt) wiederum steht dem Agenten (meist in Form eines Vektors von Daten) zur Verfügung. Das Resultat des Lernens ist die sogenannte *Policy* (Handlungsvorschrift), die eine Abbildung von Zustand auf Aktion erlaubt. Abbildung 2.2 zeigt diesen Zusammenhang der Interaktion zwischen Agent und Umgebung schematisch.

Im Gegensatz zu Supervised Machine Learning Problemen existiert also in RL-Problemen kein vorab gegebener Datensatz von Beispielen, der dieses Wissen enthält. Diese Tatsache begründet eine der zentralen Herausforderungen des RL: Der Agent muss abwägen, ob er A) seiner bisherigen Erfahrung folgt und die demnach vielversprechendsten Aktionen wählt um seinen Reward zu maximieren. Dies wird *Exploitation* genannt, oder B) Aktionen in Zuständen testet, die er bis dahin nicht ausgeführt hat, um möglicherweise bessere Ergebnisse zu entdecken. Dies wird *Exploration* genannt. Diese konkurrierenden Optionen, die beide zur Erreichung eines optimalen Lernziels notwendig sind, sind als das *Exploration-Exploitation Dilemma* bekannt.

Die in den letzten Jahren erzielten Fortschritte im Bereich der tiefen Neuronalen Netze haben das Forschungsfeld des *Deep Reinforcement Learning* hervorgebracht. Kernelement ist die Modellierung der Policy eines RL-Agenten

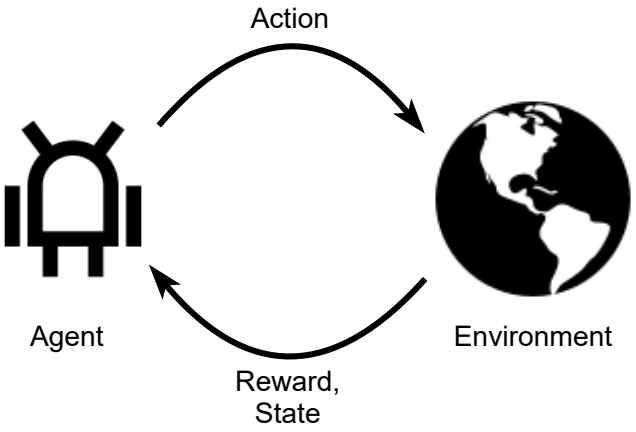


Abbildung 2.2: Schematische Darstellung der Interaktion zwischen Agent und Umgebung im Reinforcement Learning.

durch tiefe Neuronale Netze.

Im Folgenden wird ein kurzer Überblick über die formale Definition des RL-Problems gegeben. Für eine tiefergehende Betrachtung kann an dieser Stelle auf Sutton und Barto [137] verwiesen werden.

2.1.3.1 Markow Entscheidungsprozesse

Die grundlegende RL-Problemformulierung basiert auf Markov Entscheidungsprozessen (MDPs) [113]. Diese werden durch Tupel der Form $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$ definiert. $\mathcal{S}$ ist dabei eine (endliche) Menge an Zuständen. Ein Element $s_t \in \mathcal{S}$ ist derjenige Zustand, in dem sich der MDP zum Zeitpunkt t befindet. $\mathcal{A}$ ist die (endliche) Menge an möglichen Aktionen, wobei $a_t \in \mathcal{A}$ die zum Zeitpunkt t gewählte Aktion darstellt. $\mathcal{P}(s_{t+1}|s_t, a_t)$ definiert die Übergangswahrscheinlichkeitsfunktion. Ein Übergang (auch Transition genannt) wird durch Ausführen einer Aktion a_t im Zustand s_t ausgelöst. Der darauffolgende Zustand s_{t+1} wird durch $\mathcal{P}$ bestimmt. Eine Umgebung wird deterministisch genannt, wenn das Ausführen einer Aktion a_t im Zustand s_t mit Wahrscheinlichkeit 1 zu exakt einem Folgezustand s_{t+1} führt. In diesem Fall ist $\mathcal{P}(s_{t+1}|s_t, a_t) \in \{0, 1\}$. In stochastischen Umgebungen ist der Folgezustand s_{t+1} einer Aktion a_t im Zustand s_t hingegen nicht deterministisch, sodass mehr als ein Folgezustand mit Wahrscheinlichkeit > 0 eintreten kann. Der numerische Reward $\mathcal{R}(s_t, a_t)$ wird nach Ausführen von Aktion a_t im Zustand s_t erhalten. Im Regelfall wird $\mathcal{R}(s_t, a_t) \in \mathbb{R}$ angenommen.

Ziel der Problemstellung ist es nun, diejenige Policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ aus der Menge aller möglichen Policies Π zu finden, die den Erwartungswert des Returns G_t über einen potentiell unendlichen Zeithorizont maximiert:

$$G_t = \sum_{k=0}^{\infty} \gamma^k \cdot \mathcal{R}(s_{t+k}, a_{t+k}) \tag{2.1}$$

$\gamma \in [0, 1]$ stellt dabei den sogenannten Discount Factor dar.[3]
Basierend auf dieser Formalisierung stellt Reinforcement Learning einen Weg dar, die Menge aller möglichen Policies Π zu durchsuchen. Dies geschieht mittels eines Agenten, der mit der als MDP definierten Umgebung interagiert. Die Interaktion geschieht dabei in Form einer Sequenz von Aktionen $a_t \in \mathcal{A}, t = 0, 1, \ldots$. Es können weiter zwei unterschiedliche Problemstellungen unterschieden werden: Im sogenannten *fully observable* Fall kennt der Agent den aktuellen Zustand s_t sowie den Aktionsraum $\mathcal{A}$. *Den Zustand kennen* bedeutet in diesem Zusammenhang, dass die Eingabedaten, die dem Agenten (beispielsweise über Sensoren) zur Verfügung stehen, alle Aspekte der Umgebung erfassen, die für die Aktionswahl relevant sind. Die Auswirkung der Ausführung einer Aktion a_t in s_t, also die Übergangswahrscheinlichkeiten $\mathcal{P}(_{s_t+1}|s_t, a_t)$ sowie der Reward $\mathcal{R}(s_t, a_t)$, sind dem Agenten hingegen nicht bekannt. Im *partially observable* Fall sind nicht alle relevanten Elemente der Umgebung für den Agenten einsehbar, beispielsweise weil die Sensoren Rauschen enthalten oder komplette Aspekte der Umgebung nicht erfasst werden. [120]
Je nach verwendeter Lernstrategie des Agenten werden weiter zwei Kategorien des RL unterschieden.

2.1.3.2 Value-based Reinforcement Learning

Im *Value-based RL* wird versucht eine *Value Funktion* zu approximieren, aus der dann direkt eine Policy abgeleitet wird. Die Value Funktion bewertet wie gut es für einen Agenten ist, sich in einem gewissen Zustand s zu befinden. *Gut* bezieht sich in diesem Kontext darauf, wie hoch der Erwartungswert des zukünftigen Returns ist. Da dieser zukünftige Return von der Wahl der Aktionen des Agenten abhängt, werden Value Funktionen im Bezug auf Policies definiert:

$$V_\pi(s) = \mathbb{E}_\pi[G_t|s_t] \tag{2.2}$$

Eine optimale Value Funktion V_* ist diejenige Value Funktion, die verglichen mit allen anderen Value Funktionen, einen höheren Erwartungswert des zukünftigen Returns liefert:

$$V_*(s) \geq V_\pi(s), \forall \pi \in \Pi. \tag{2.3}$$

Abweichend von diesen, auch *State-Value Funktionen* genannten Funktionen V_π ist es auch möglich eine *Action-Value Funktion* Q_π (*Q-Funktion* genannt) zu

[3] Die Notation $[a, b]$ wird im Rahmen der gesamten Arbeit verwendet, um das geschlossene Interval von a nach b zu bezeichnen. Dieses beinhaltet also die Werte a und b.

definieren. Diese bewertet, wie gut es für einen Agenten ist, in einem Zustand s_t eine Aktion a_t auszuführen, wenn danach weiter der Policy π gefolgt wird.

$$Q_\pi(s,a) = \mathbb{E}_\pi[G_t|s_t,a_t] \tag{2.4}$$

Die optimale Action-Value Funktion ist entsprechend definiert:

$$Q_*(s,a) \geq V_\pi(s,a), \forall \pi \in \Pi. \tag{2.5}$$

Einer der verbreitetsten Ansätze des Value-based RL ist *Q-Learning* [143]. Grundidee dieses Ansatzes ist es, die optimale Action-Value Funktion Q_* zu approximieren, indem eine geratene, initiale Q-Funktion iterativ aktualisiert wird durch:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha[r_t + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t)] \tag{2.6}$$

Grundlage des Lernens ist somit *Erfahrung* in Form von Tupeln: $e_t = (s_t, a_t, s_{t+1}, r_t)$. Die Lernrate α ist dabei ein Hyperparameter, der bestimmt wie stark aktuelle Erfahrungen bisherige Werte beeinflussen. Die Menge aller Erfahrungstupel der Zeitschritte $t_1, \ldots, t_m$ in Bezug auf ein Trainingslimit m wird als das Trainings Set $\mathcal{T} = \{e_{t_1}, ..., e_{t_m}\}$ bezeichnet. Die so gelernte Action-Value Funktion Q konvergiert gegen die optimale Action-Value Funktion Q_*. Basierend auf dieser ist es trivial eine optimale Policy abzuleiten, indem in jedem Schritt einfach diejenige Aktion gewählt wird, die den Funktionswert maximiert: $\pi_*(s_t) = \arg\max_a Q(s_t,a)$.
In Umgebungen mit hochdimensionaler Zustandsbeschreibung oder bei Verwendung kontinuierlicher Aktionsräume werden meist parametrisierbare Funktionsapproximatoren wie Neuronale Netze verwendet um die Action-Value Funktion zu modellieren:
$Q(s_t,a_t;\theta) \approx Q_*(s_t,a_t)$, wobei θ die Parameter des Modells darstellen. Bei Verwendung von tiefen Neuronalen Netzen wird dies auch *Deep Reinforcement Learning* genannt [96]. Ein entsprechender Ansatz aus dem Bereich des Value-based RL, *Deep Q-Learning* [96], bildet die Grundlage der Unsicherheitsmodellierung in Kapitel 4.

2.1.3.3 Policy-based Reinforcement Learning

Die zweite große Kategorie von RL-Methoden sind Policy-based RL-Ansätze. Diese unterscheiden sich dadurch vom Value-based RL, dass die Policy π und damit die Aktionsauswahl direkt modelliert, und nicht von einer Value Funktion abgeleitet wird. Die heute am weitesten verbreiteten Varianten von Policy-based RL, Actor-Critic Methoden, stellen Hybride dar, die sowohl eine Policy als auch eine Value Funktion modellieren. Die Policy wird dabei als *Actor* bezeichnet, da sie für die Aktionswahl zuständig ist. Die Value Funktion wird der *Critic* genannt, da diese die ausgeführten Aktionen bewertet. Dabei können verschiedene Varianten des Critics eingesetzt werden. Im Falle eines Action-

Value Critics kann die Bewertung beispielsweise mittels des *TD-Error* δ_t wie folgt erfolgen:

$$\delta_t = r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t). \tag{2.7}$$

Dieser TD-Error wird anschließend genutzt um die soeben in Zustand s_t gewählte Aktion a_t zu bewerten. Besitzt der TD-Error ein positives Vorzeichen, war die bisherige Action-Value Schätzung $Q(s_t, a_t)$ zu pessimistisch und die Policy π sollte dahingehend aktualisiert werden, dass die Wahl von a_t zukünftig stärker bevorzugt wird. Ist der TD-Error negativ, sollte die Wahl abgeschwächt werden. Wird die Policy beispielsweise mittels eines Neuronalen Netzes als probabilistisches Modell realisiert, können die Parameter des Netzes so angepasst werden, dass die modellierte Wahrscheinlichkeit der Aktionswahl $P(a_t|s_t;\theta)$ erhöht oder verringert wird. [132, 137] Ein aktueller Vertreter des Policy-based RL, *PPO2* [33], kommt in Kapitel 5 zur Policy-basierten Erkennung von Out-of-Distribution Situationen zum Einsatz.

2.1.4 Neuronale Netze

Mit stetig steigender Rechenkapazität und der laufenden Entwicklung verbesserter Algorithmen haben sich Neuronale Netze in den vergangenen Jahren als eine der mächtigsten Ausprägungen von Computermodellen zur statistischen Mustererkennung erwiesen.
Grundlegend betrachtet sind (künstliche) Neuronale Netze universelle, parametrische Funktionsapproximatoren [16]: $f(x;\theta) = y$. Im sogenannten *Training* wird versucht, durch Optimierung der Parameter θ die durch das Neuronale Netz realisierte Funktion f bestmöglich an eine unbekannte Zielfunktion $f^*(x) = t$ anzunähern[4]. Abstrakt betrachtet, beschreibt die durch das Neuronale Netze definierte Funktion f eine Folge von Transformationen der Eingabedaten x. Im Folgenden wird ein Überblick über die funktionale Form dieser Netze und die wichtigsten Komponenten gegeben. Die Notation folgt dabei Bishop [16].

Zunächst werden M Linearkombinationen der Input Variablen (Features) $x_1, ..., x_D$ eines D-dimensionalen Eingabevektors x konstruiert. Zur besseren Verständlichkeit können die Parameter θ eines Netzes getrennt notiert werden als: *Gewichte* $w_{ji}^{(1)}$ und *Bias* $w_{j0}^{(1)}$, mit $j = 1, ..., M$ und $i = 1, ..., D$. Die (1) gibt an, dass es sich bei diesen Parametern um den ersten Layer des Netzes handelt.

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \tag{2.8}$$

[4]Die Vorhersage wird somit mit y bezeichnet, während t ein gegebenes *Target* (*Label*) des Datensets darstellt.

Das Ergebnis der obigen Gleichung a_j wird als *Aktivierung* bezeichnet. Jede dieser Aktivierungen wird durch eine *Aktivierungsfunktion* $h(\cdot)$ transformiert:

$$z_j = h(a_j) \tag{2.9}$$

Dies wird auch als der Output der *Hidden Units* (Versteckte / Verdeckte Neuronen) bezeichnet. Es existieren eine Vielzahl von Aktivierungsfunktionen $h(\cdot)$ [26, 73, 90, 99, 101]. Eine der aktuell meistverwendeten ist die *Rectified Linear Unit* (ReLU) [99] (siehe Abbildung 2.3b):

$$ReLU(a) = \max(a, 0) \tag{2.10}$$

Einige weitere Aktivierungsfunktionen werden im nächsten Absatz näher erläutert. Alle haben jedoch gemeinsam, dass sie differenzierbar sowie nichtlinear sind (siehe Abbildung 2.3).

Abschließend erfolgt eine weitere lineare Kombination der Werte:

$$a_k = \sum_{j=1}^{M} w_{kj}^2 z_j + w_{k0}^{(2)} \tag{2.11}$$

mit $k = 1, ..., K$ wobei K die Anzahl der Outputs ist. Dies ist gleichzeitig der zweite Layer des Neuronalen Netzes.
Die Aktivierungsfunktion des letzten Layers wird in der Regel abweichend gewählt, je nach Problemstellung und angenommener Verteilungsfunktion der Daten. Soll ein Regressionsproblem gelöst werden kommt zumeist die Identitätsfunktion zum Einsatz, so dass $y_k = a_k$. Bei binären Klassifikationsproblemen wird meist die Sigmoid Funktion eingesetzt (siehe Abbildung 2.3a), so dass die finale Ausgabe (Output) die folgende Form erhält:

$$y_k = \sigma(a_k) \tag{2.12}$$

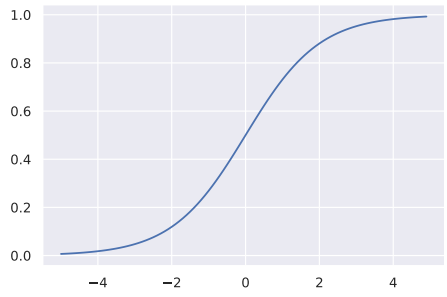
Die Sigmoid Funktion besitzt dabei die Form:

$$\sigma(a) = \frac{1}{1 + e^{-a}} \tag{2.13}$$

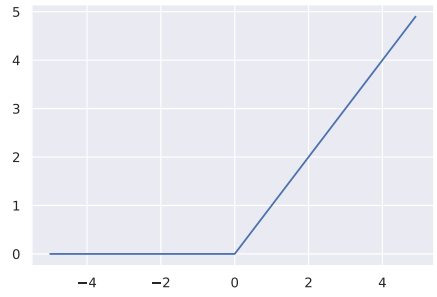
Soll als Ausgabe des Netzes die Standardabweichung einer Verteilung vorhergesagt werden, wird hingegen häufig die Softplus Funktion [55] (siehe Abbildung 2.3c) verwendet, da diese von $\mathbb{R}$ nach $\mathbb{R}_+$ abbildet und so garantiert, dass keine negative Standardabweichung vorhergesagt werden kann:

$$\sigma_+(a) = \log(1 + e^a) \tag{2.14}$$

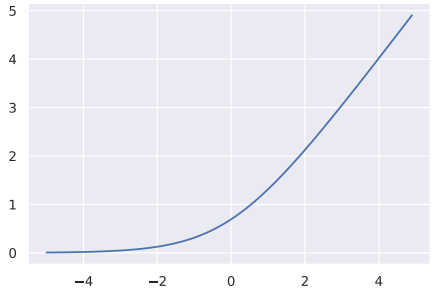
Die Eingabe eines Input Vektors und die sukzessive Berechnung aller Aktivierungen wird entsprechend des Informationsflusses durch das Netz als *Forward Propagation* bezeichnet.



(a) Sigmoid



(b) ReLU



(c) Softplus

Abbildung 2.3: Darstellung der häufig verwendeten Aktivierungsfunktionen Sigmoid, ReLU und Softplus.

Abbildung 2.4 zeigt eine schematische Abbildung eines zweischichtigen Neuronalen Netzes in Form eines Graphen, basierend auf [16]. Die Input Variablen $x_1, \ldots, x_D$ (Inputs), die Hidden Units sowie die Output Variablen sind durch Knoten dargestellt. Die Kanten des Graphen repräsentieren die Gewichte des Netzes. Der jeweilige Bias einer Schicht wird durch eine Kante von den zusätzlichen Knoten x_0 beziehungsweise z_0 dargestellt.
Diese grundlegende Struktur von Neuronalen Netzen lässt sich einfach durch Wiederholung von Gleichung 2.8, gefolgt von einer nichtlinearen Aktivierungsfunktion, auf sogenannte *Tiefe Neuronale Netze* generalisieren. Man spricht in diesem Zusammenhang dann von *Deep Learning*.
Bevor die Parameter des Netzes optimiert werden können, muss eine *Fehlerfunktion* (auch *Loss*, *Cost* oder *Objective* Function genannt) definiert werden. Im Fall von Regressionsproblemen wird häufig eine auf dem quadratischen Fehler basierende Fehlerfunktion minimiert:

$$C(\theta) = \frac{1}{2} \sum_{n=1}^{N} \|y(x_n, w) - t_n\|^2 \tag{2.15}$$

Zur Minimierung des Losses und der Anpassung der Parameter (auch *Parameterupdate* genannt) kommen Gradientenverfahren zum Einsatz. Da die

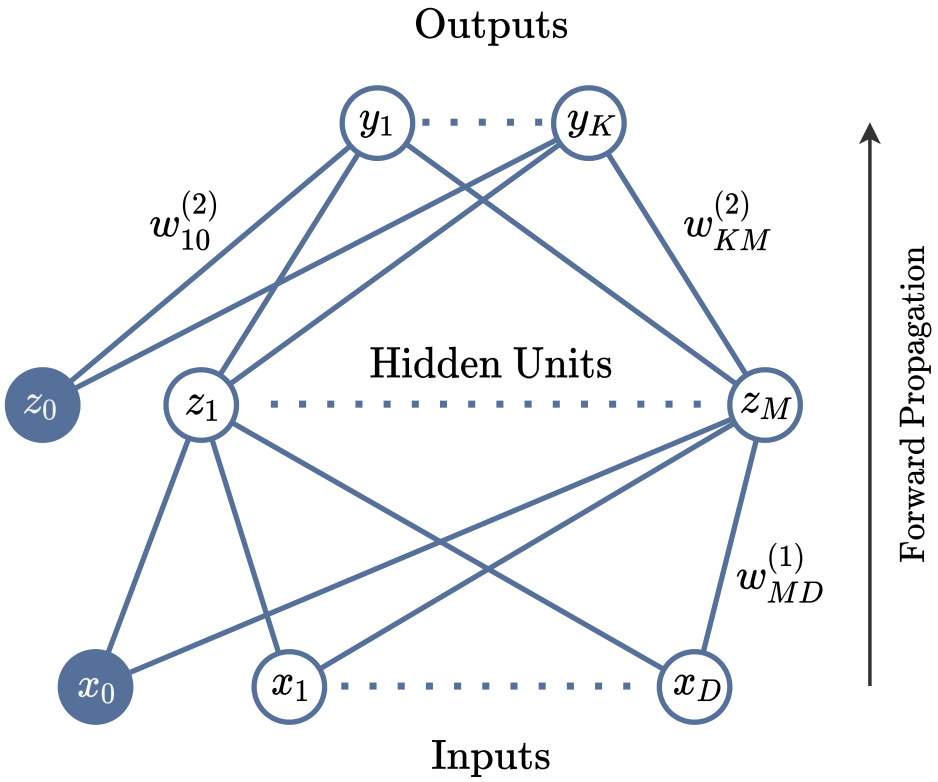


Abbildung 2.4: Schematische Abbildung eines zweischichtigen Neuronalen Netzes. Darstellung adaptiert von [16].

Struktur der Neuronalen Netze das Finden einer analytischen Lösung für die Ableitung $\Delta C(\theta) = 0$ nicht zulässt, werden iterative Verfahren eingesetzt. In der einfachsten Form des sogenannten *Gradient Descent* werden die Gewichte θ iterativ einen kleinen Schritt in Richtung des negativen Gradienten der Fehlerfunktion verschoben:

$$\theta^{(\tau+1)} = \theta^{(\tau)} - \eta \nabla C(\theta^{(\tau)}) \tag{2.16}$$

τ bezeichnet dabei den aktuellen Parameterupdate Schritt und η die Lernrate. In der Regel werden abgewandelte, effizientere Verfahren eingesetzt, bei denen nicht für jedes Parameterupdate die gesamten Trainingsdaten evaluiert werden müssen. Zwei der bekanntesten Vertreter sind *Minibatch Gradient Descent* [98] und *Stochastic Gradient Descent* [82].

2.2 Quantifizierung von Unsicherheit

> La Théorie des probabilités n'est au fond, que le bon sens réduit au calcul.
>
> Die Wahrscheinlichkeitstheorie ist im Grunde nichts anderes, als auf Berechnungen reduzierter gesunder Menschenverstand.
>
> Pierre Laplace, 1812 [81]

Wie zuvor bereits erwähnt, fokussiert sich diese Arbeit unter anderem auf Fragen zur Unsicherheit gelernter Modelle. Für ein besseres Verständnis soll zunächst aber ein grundlegender Überblick über hierfür relevante Themen aus dem Feld der Wahrscheinlichkeitstheorie gegeben werden.
Innerhalb des Gebiets der Wahrscheinlichkeitstheorie existieren zwei unterschiedliche Betrachtungsweisen: Die *Frequentistische Betrachtungsweise* interpretiert Wahrscheinlichkeiten als relative Häufigkeiten von Ereignissen, die wiederholt auftreten können. In der *Bayesschen Betrachtungsweise* quantifiziert Wahrscheinlichkeit die Unsicherheit oder Unwissenheit über einen betrachteten Gegenstand. Die vorliegende Arbeit folgt primär der *Bayesschen Betrachtungsweise*. Dennoch sind nicht alle eingesetzten Methoden und Modellierungsansätze rein Bayesscher Natur, worauf an entsprechenden Stellen hingewiesen wird.

2.2.1 Zufallsvariablen und Wahrscheinlichkeitsverteilungen

Gleich ob der Frequentistischen oder Bayesschen Betrachtungsweise gefolgt wird, ist ein zentrales Element die *Zufallsvariable* X. Ihre möglichen Werte repräsentieren mögliche Ausprägungen beziehungsweise Ergebnisse eines Experiments. Diskrete Zufallsvariablen besitzen eine endliche oder abzählbar unendliche Menge an möglichen Werten, während dies bei stetigen Zufallsvariablen nicht der Fall ist. Die Wahrscheinlichkeit, dass ein gewisser Wert eintritt wird mit $P(X = x)$ bezeichnet. Zufallsvariablen besitzen somit Wahrscheinlichkeitsverteilungen, die angeben wie wahrscheinlich, und damit wie sicher oder unsicher, das Eintreten eines gewissen Ergebnisses ist.
Wahrscheinlichkeitsverteilungen sind durch Kennzahlen, die sogenannten *Momente* der entsprechenden Zufallsvariable bestimmt. Das erste Moment ist der Mittelwert, oder auch Erwartungswert genannt, und wird mit μ bezeichnet. Basierend auf diesem Mittelwert können die sogenannten zentralen Momente[5] definiert werden, bei denen die Verteilung um μ betrachtet wird:

$$\mu_n = \mathbb{E}[(X - \mu)^k] \tag{2.17}$$

[5] Zentrale Momente, da sie um den Mittelwert μ zentriert sind.

mit $k \in \mathbb{N}$.

Das erste zentrale Moment ist stets 0, während das zweite zentrale Moment als *Varianz* bezeichnet wird: $\mu_2 = \mathbb{E}[(X - \mu)^2]$. Das dritte zentrale Moment wird in der Regel zusätzlich mit der Standardabweichung σ normiert und *Schiefe* (skewness) genannt: $\mu_3 = \mathbb{E}[(\frac{X-\mu}{\sigma})^2]$. Diese und weitere Kennzahlen werden in Unterabschnitt 3.3.1 im Rahmen der Isovistenanalyse verwendet um die Wahrnehmung von Raum mathematisch quantifizierbar zu machen.
Je nach Komplexität der tatsächlichen Wahrscheinlichkeitsverteilung muss jedoch beachtet werden, dass ein allzu vereinfachendes Beschreiben der Verteilung über zentrale Momente einen Großteil der Informationen verlieren kann. [16, 98]

2.2.2 Entropie

Ein weiteres Mittel um Wahrscheinlichkeitsverteilungen zu beschreiben ist die *Entropie* einer Wahrscheinlichkeitsverteilung. Entropie, in diesem Kontext der Informationstheorie auch *Shannon Entropie* genannt, wurde bereits 1948 von Claude Shannon [130] vorgestellt. Sie gibt Antwort auf die Frage, „wie zufällig“ eine zugrundeliegende Zufallsvariable ist. Anders formuliert, quantifiziert die Entropie die vorliegende Unsicherheit. Auf einer diskreten Zufallsvariable X mit n möglichen Ausprägungen ist sie definiert als:

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log P(x_i) \tag{2.18}$$

Bei Verwendung des Logarithmus zur Basis 2 gibt H die Anzahl der Bit an, die zur Repräsentation der Information minimal notwendig sind. Die maximale Entropie einer diskreten Verteilung liegt vor, wenn alle Ausprägungen die gleiche Wahrscheinlichkeit aufweisen. In diesem Fall liegt eine Gleichverteilung vor, was das Auftreten einer bestimmten Ausprägung maximal unsicher macht. Um die Entropie von Verteilungen unterschiedlicher Anzahl n an Ausprägungen vergleichen zu können, kann die *Normalisierte Entropie* H_n berechnet werden:

$$H_n(X) = \frac{H(X)}{\log n} = -\sum_{i=1}^{n} \frac{P(x_i) \log P(x_i)}{\log n} \tag{2.19}$$

Der Wertebereich ist damit begrenzt auf $H_n(X) \in [0, 1]$.

2.2.3 Problemtypen der Quantifizierung von Unsicherheit

Bis zu diesem Punkt wurde noch nicht betrachtet, welches Element eines Systems von Interesse mittels Zufallsvariablen beschrieben und untersucht wird. Grundsätzlich können basierend darauf zwei verschiedene Arten von Problemen der Quantifizierung von Unsicherheit unterschieden werden.

Im sogenannten *Forward Problem* ist das Ziel die Quantifizierung der Unsicherheit in den Vorhersagen des Modells. Solche *Forward Problems* entsprechen oftmals der Richtung der Kausalität des zugrundeliegenden physikalischen Systems, das betrachtet wird. Diese Probleme besitzen deshalb meist eine eindeutige Lösung. Ein entsprechendes Problem könnte beispielsweise die Vorhersage der Position eines Roboterarms nach der Durchführung gewisser Rotationen sein [16].

Sogenannte *Inverse Problems* hingegen haben zum Ziel, den Systemzustand basierend auf Beobachtungen der eingetretenen Ergebnisse abzuleiten. Sie stellen somit in gewisser Weise die umgekehrte Richtung der *Forward Problems* dar. Zur Lösung dieser Probleme kann das Bayessche Theorem (Unterabschnitt 3.3.2) eingesetzt werden [98]. Am Beispiel des Roboterarms betrachtet, wäre das Ziel nun nicht die Vorhersage der Endposition, sondern die Ermittlung notwendiger Rotationen, die zu dieser Zielposition führen. Hier existiert oftmals mehr als eine Lösung, weshalb multimodale Modellierungsansätze wie *Mixture Density Networks* [16] notwendig sind.

Neben diesen verschiedenen Typen von Problemstellungen ist es weiterhin wichtig, verschiedene Arten von Unsicherheit zu unterscheiden.

2.2.4 Aleatorische Unsicherheit

Aleatorische Unsicherheit (abgeleitet vom lateinischen Wort *alea*: „Würfel“) repräsentiert die inhärente Stochastizität eines Systems. Da diese Art der Unsicherheit untrennbar mit dem System verbunden ist, kann keine noch so große Menge an Daten die beobachtete Variabilität erklären. Folglich kann diese, auch *Datenunsicherheit* [98], *Noise* [58] oder *Risiko* [107] genannte Form, nicht durch Sammeln weiterer Daten reduziert werden. Aleatorische Unsicherheit ist somit irreduzibel. Ein Grund für das Auftreten könnte sein, dass gewisse Eigenschaften, die notwendig dafür wären das Verhalten des betrachteten Systems zu erklären, nicht erfassbar oder zumindest nicht Teil der Datenerfassung sind. Stellen wir uns vor, Ziel wäre die Modellierung (und damit Vorhersage) der Strecke, die verschiedene Autos auf einer Autobahn in einer gewissen Zeit zurücklegen. Würde bei der Datenerfassung lediglich Strecke und Zeit erfasst, aber nicht die Geschwindigkeit der Autos, so wäre die Variabilität in den gemessenen Daten nicht zu erklären. Gleich wie viele Daten auf diese Weise gesammelt werden würden, die Unsicherheit könnte nicht reduziert werden. Ein Modell kann diese Unsicherheit dennoch korrekt repräsentieren, indem anstelle einer Punktvorhersage eine Verteilung ausgegeben wird [104]. Einer der verbreitetsten Ansätze zur Modellierung aleatorischer Unsicherheit sind *Mixture Density Networks* [16]. Die Quantifizierung und Modellierung multimodaler aleatorischer Unsicherheit steht jedoch nicht im Fokus dieser Arbeit, hier kann auf Sedlmeier et al. [128] verwiesen werden.

2.2.5 Epistemische Unsicherheit

Epistemische Unsicherheit (abgeleitet vom griechischen *episteme*: „Wissen“) hingegen beschreibt das Unwissen darüber, welche Modellparameter das betrachtete System bestmöglich abbilden.
Osband [107] präsentiert ein hilfreiches Beispiel für ein besseres Verständnis des Unterschiedes von Aleatorischer und Epistemischer Unsicherheit: Das Werfen einer fairen Münze. Modelliert als Bernoulli Zufallsvariable beträgt $p = 0,5$. Der Ausgang jedes einzelnen Münzwurfes enthält damit Aleatorische Unsicherheit beziehungsweise Risiko. Daneben kann sich ein lernender Agent zudem auch über den genauen Wert von p (den Modellparameter) unsicher sein, was eine Form von Epistemischer Unsicherheit darstellt. Aus der Perspektive der Bayesschen Statistik betrachtet, erfasst Epistemische Unsicherheit somit die Variabilität des Posteriors des Agenten. Diese kann durch statistische Analyse der Daten aufgelöst werden. Osband weist darauf hin, dass die Grenze zwischen den Arten der Unsicherheit somit mit der Art der Modellierung verbunden ist. Wäre es möglich die exakten Dynamiken des Münzwurfes zu modellieren, wäre der Ausgang eines einzelnen Wurfes möglicherweise keinerlei Unsicherheit mehr unterworfen.
Abbildung 2.5 zeigt eine beispielhafte Darstellung von Aleatorischer und Epistemischer Unsicherheit anhand eines synthetischen 2-dimensionalen Datensatzes. Ziel sei die Modellierung der Funktion $f(x) = y$, basierend auf den vorhandenen, verrauschten Daten der tatsächlich zugrundeliegenden Funktion $y = -sin(x)$. Bedingt durch die Streuung auf der Y-Achse existiert Aleatorische Unsicherheit. Das Fehlen von Datenpunkten mit $x > 10$ stellt hingegen Epistemische Unsicherheit dar und könnte durch Sammeln weiterer Daten reduziert werden.

2.3 Zusammenfassung

In diesem Kapitel wurden wichtige Grundlagen zur Modellierung lernender Systeme vermittelt. In den vergangenen Jahren hat sich gezeigt, dass statistische Modelle die aktuell mächtigsten Werkzeuge zur Realisierung künstlicher Intelligenz darstellen. Die Verfügbarkeit großer Datensätze, in Kombination mit laufend weiterentwickelten ML-Algorithmen und performanter Hardware, hat zu enormen Erfolgen in den verschiedensten Anwendungsfeldern geführt. Diese reichen von der Spracherkennung, über die Bilderkennung, bis hin zum Training autonomer Roboter mittels RL-Techniken. Besonders im letzteren Bereich sind Fragen der Wahrnehmungsmodellierung oftmals entscheidend bei der Erreichung von Lernerfolg [137]. Ebenso sind Fragen zur Modellierung der stets präsenten Unsicherheit von Vorhersagen und Entscheidungen in all diesen Bereichen zentral.
Entsprechend finden sich die in diesem Kapitel vorgestellten Grundkonzepte der Wahrscheinlichkeitstheorie und probabilistischen Modellierung in allen drei

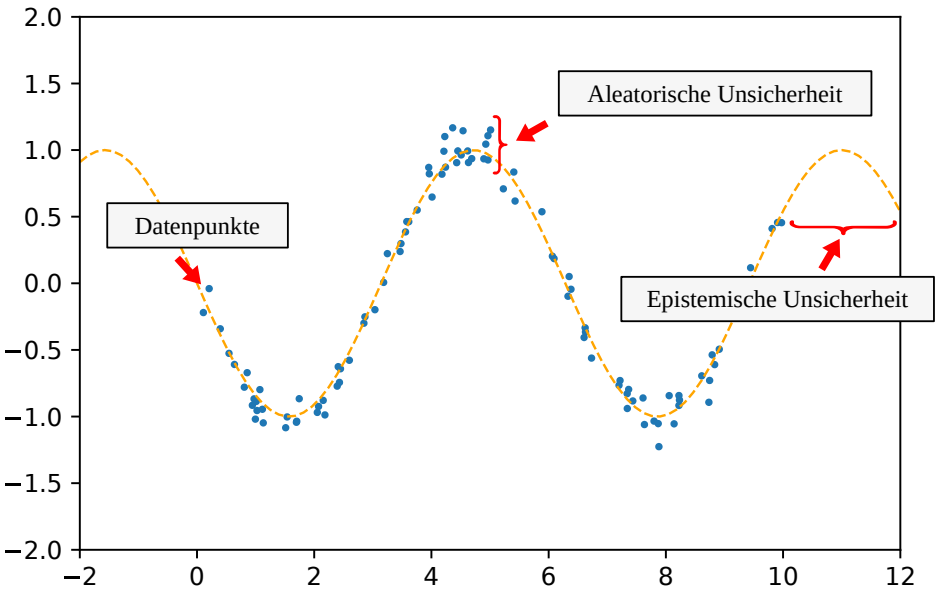


Abbildung 2.5: Beispielhafte Darstellung von Aleatorischer und Epistemischer Unsicherheit anhand eines synthetischen, 2-dimensionalen Datensatzes. Blaue Punkte repräsentieren die vorhandenen Daten, die orange Linie die unbekannte Zielfunktion. Ziel sei die Modellierung der Funktion $f(x) = y$.

der im Rahmen dieser Arbeit vorgestellten Ansätze wieder. So stehen beispielsweise Bayessche Methoden im Zentrum der Modellierung von Überraschung im zweiten Teil von Kapitel 3. Ebenso finden sich approximative Varianten dieser Bayesschen Methoden im Ansatz zur Modellierung aleatorischer und epistemischer Unsicherheit in Kapitel 4. Die Grundlage dieser Modelle bilden wiederum tiefe Neuronale Netze, die als universelle Funktionsapproximatoren die Value Funktionen von RL-Agenten realisieren. Die Quantifizierung von Unsicherheit mittels Entropie liefert die Basis des in Kapitel 5 vorgestellten Ansatzes.
Es zeigt sich somit, dass die im Rahmen der vorliegenden Arbeit neu entwickelten Verfahren zur Modellierung von Unsicherheit und Wahrnehmung stets auf die „altbekannten“ Werkzeuge der statistischen Modellierung zurückgreifen. Entscheidend ist ihre gewinnbringende Integration in die aktuellen Methoden der KI-Forschung.

3 Wahrnehmungsmodellierung

Durch den stetigen Fortschritt der technischen Entwicklung ist aktuell zu beobachten, wie mehr und mehr KI-basierte Systeme ihren Aufgaben in räumlicher Nähe zu Menschen nachgehen. Ein prominentes Beispiel sind autonome Lieferroboter, wie sie bereits seit einigen Jahren in den USA im produktiven Einsatz sind [47]. Diese sind in der Lage, großteils ohne menschliche Eingriffe Waren, beispielsweise Essenslieferungen zuzustellen. Mobile Roboter betreten somit zunehmend unser tägliches Leben, im privaten wie geschäftlichen Umfeld. Neben Anwendungen der Logistik sind insbesondere Systeme zur Unterstützung der Wegfindung verbreitet. Diese sind besonders in komplexen öffentlichen Umgebungen wie Flughäfen [119], Messen oder Krankenhäusern [29, 75] hilfreich. Auch im privaten Umfeld versprechen intelligente mobile Roboter eine Komfortsteigerung und Unterstützung, beispielsweise für ältere Personen durch sogenanntes Active Assisted Living (AAL) [115].
Gemeinsames Element all dieser Anwendungsfälle ist die räumliche Nähe zu Menschen. Um hier eine optimale und sichere Funktionsweise der mobilen Roboter zu erlangen, ist ein gemeinsames Wahrnehmungsverständnis zwischen Mensch und Maschine essentiell, um Missverständnisse und möglicherweise gefährliche Fehler zu minimieren. Die im Rahmen dieses Kapitels vorgestellten Ansätze zur Wahrnehmungsmodellierung mittels ML-Technologien können hier einen Beitrag leisten.

3.1 Vorveröffentlichungen

Die zentralen Inhalte dieses Kapitels wurden bereits in [45, 124, 125] publiziert. Wie in Abschnitt 1.2 dargestellt stammen die Hauptinhalte, insbesondere die Realisierung mittels einer Unity Simulationsumgebung sowie der Einsatz von Unsupervised Learning basierten ML-Techniken, wie in Abschnitt 3.6 und 3.5 dargestellt, sowohl im Konzept, als auch der Implementierung und Evaluation im Karten- und PCA Feature-Raum vom Autor der vorliegenden Arbeit.
Die ursprüngliche Idee zur Kombination von Bayesian Surprise mit Isovisten-basierter Wahrnehmung stammt von Dr. Sebastian Feld. Ebenso das Konzept, die so quantifizierte Überraschung als neue Dimension des Kontexts zu verwenden. Die gesamte in Abschnitt 3.7 vorgestellte Modellierung (sowohl im Konzept, als auch der Implementierung) stammt vom Autor der vorliegenden Arbeit. Des Weiteren wurde auch die in Abschnitt 3.8 präsentierte Evaluation auf sämtlichen Umgebungen vom Autor dieser Arbeit durchgeführt.

3.2 Motivation und Grundidee

Wie zu Beginn des Kapitels dargestellt, ist ein gemeinsames Wahrnehmungsverständnis essentiell für eine optimale Zusammenarbeit zwischen Menschen und mobilen Robotern. Dieses Wahrnehmungsverständnis kann anschließend genutzt werden, um beispielsweise erweiterte Kontextinformationen im Rahmen von Location-based Services (LBS) [79] zur Verfügung zu stellen. Diese LBS-Dienste verwenden Wissen über die momentane räumliche Situation, um dies zu realisieren. Auf diese Weise könnte beispielsweise die verbale Interaktion mit einem mobilen Roboter deutlich vereinfacht werden, wenn dieser ein Wahrnehmungsverständnis der ihn umgebenden Räume besäße. So könnten Sprachkommandos wie „Navigiere bis zum Ende des schmalen Korridors, durch die linke Türe in die große Haupthalle“ umgesetzt werden. Neben solchen Anwendungsszenarien aus dem Bereich der Indoor-Navigation [145] könnte ein solches Wahrnehmungsverständnis auch dazu beitragen gefährliche, da beispielsweise schlecht einsehbare, Situationen im Straßenverkehr zu erkennen. Ein anderes potentielles Anwendungsfeld ist die Nutzung zum Training von RL-Agenten. Hier könnte ein Wahrnehmungsverständnis, beispielsweise in Form des in Abschnitt 3.7 vorgestellten Konzepts des Bayesian Surprise, zur Optimierung der Exploration des RL-Agenten verwendet werden. Ein so konstruierter lernender Agent würde versuchen die modellierte Überraschung zu maximieren, um so neue, unbekannte Regionen zu entdecken.
Das Grundkonzept, um all diese potentiellen Anwendungsszenarien realisieren zu können, ist die Modellierung eines räumlichen Wahrnehmungsverständnisses für lernende Systeme. Die Basis bildet dabei eine Isovisten-basierte Quantifizierung der Wahrnehmung. Im Folgenden wird ein zu diesem Zweck entwickeltes Framework vorgestellt, das die Generierung von Isovistenmessungen entlang geospatialer Trajektorien innerhalb einer 3D-Simulationsumgebung ermöglicht. Diese Daten bilden die Grundlage für eine Machine Learning (ML) basierte Modellierung. Die so geschaffene Wahrnehmungsmodellierung wird anschließend mit Bayesschen Verfahren erweitert, um vorhandene Unsicherheit in der Wahrnehmung als zusätzliche Informationsquelle nutzbar zu machen.

3.3 Grundlagen

Im folgenden Abschnitt werden Grundlagen zur numerischen Repräsentation von Wahrnehmung und Raum eingeführt. Diese bilden die Basis für die im Rahmen der vorliegenden Arbeit entwickelten Konzepte der Wahrnehmungsmodellierung. Anschließend werden die Grundkonzepte der Bayesschen Inferenz vermittelt. Diese stellen die mathematische Ausgangslage der Modellierung Bayesscher Überraschung dar, die im darauffolgenden Abschnitt erläutert wird. Ein Verständnis dieser drei Elemente ist für die im Rahmen dieser Arbeit entwickelte Kombination von Bayesian Surprise und Isovisten-basierter Wahrnehmungsmodellierung (Abschnitt 3.7) notwendig.

3.3.1 Isovisten Analyse

Seit Ende der 1960er Jahre wurde intensiv an der numerischen Repräsentation von Raumwahrnehmung geforscht, insbesondere in Bezug auf die Wahrnehmung von Architektur. Als Beispiel kann hier Hayward und Franklin [59] erwähnt werden, die in ihrer Arbeit den Einfluss von begrenzenden Elementen wie Wänden oder Bäumen auf die Wahrnehmung von räumlicher Offenheit untersuchten. Mit fortschreitender Zeit hat sich der Begriff der *Space Syntax* herausgebildet, unter dem verschiedene Theorien und Hilfsmittel zur Untersuchung räumlicher Anordnungen zusammengefasst werden. Wichtiges Kernelement davon ist, dass hierbei keine geometrischen Messungen verwendet werden [61]. Obwohl ursprünglich unabhängig davon entwickelt, wird heute auch das Konzept der *Isovisten Analyse* oft als Teil der Space Syntax betrachtet.
Die *Isovisten Analyse* wurde ursprünglich von Tandy [138] vorgestellt. Als Isovist wird dabei die Menge aller Punkte im Raum bezeichnet, die von einem aktuellen Standpunkt aus sichtbar sind. In [14] wurde dieses Konzept darauffolgend formal definiert und um analytische Messgrößen erweitert. Dies eröffnet die Möglichkeit räumliche Wahrnehmung numerisch zu erfassen, zu quantifizieren und folglich zu modellieren.
Die von Benedikt [14] vorgestellten sechs Messgrößen eines Isovisten lauten wie folgt:

1. A_x: **Area** quantifiziert den Flächeninhalt eines Isovisten. Je höher der Messwert, desto mehr Fläche ist vom aktuellen Standpunkt aus sichtbar. Gleichzeitig bedeutet dies auch, dass der Standpunkt von einem großen Gebiet aus einsehbar ist.

2. P_x: **Real-surface Perimeter** quantifiziert den Anteil des Umfangs eines Isovisten der auf sichtbaren Hindernisoberflächen liegt, beispielsweise auf Wänden.

3. Q_x: **Occlusivity** hingegen quantifiziert den Anteil des Umfangs eines Isovisten, der im freien Raum liegt. Anders formuliert sind dies die verdeckten radialen Grenzlinien, die an den Rändern von Hindernissen beginnen und freien Raum durchschneiden um erneut an Hindernissen zu Enden. Dieser Wert quantifiziert somit Menge von verdeckten Sichtbereichen. Hohe Werte gehen mit hoher Unsicherheit in der visuellen Wahrnehmung einher.

4. $M_{2,x}$: **Variance** bezeichnet das zweite zentrale Moment der ausgesandten Strahlenlängen, die den Isovisten nach [14] definieren.

5. $M_{3,x}$: **Skewness** bezeichnet das dritte zentrale Moment der Strahlenlängen des Isovisten.

6. N_x: **Circularity** stellt einen isoperimetrischen Quotienten dar und evaluiert den Umfang des Isovisten im Verhältnis zur Fläche. Intuitiv betrachtet, beschreibt diese Messgröße wie ähnlich die Form des Isovisten

zu einem Kreis ist. Circularity wird berechnet als $N_x = |\partial V_x|^2/4\pi A_x$, wobei $|\partial V_x|$ den Umfang des Isovisten bezeichnet.

Abbildung 3.1 zeigt eine schematische Darstellung eines Isovisten. Der Agent befindet sich an der mit x markierten Position. Die grau hinterlegte Fläche ist der für den Agenten einsehbare Bereich und wird durch die Messgröße *Area* A_x quantifiziert. Rote Linien markieren die für die Messgröße *Real-surface Perimeter* P_x relevanten Grenzlinien, blaue Linien die in die Berechnung von *Occlusivity* Q_x einbezogenen.

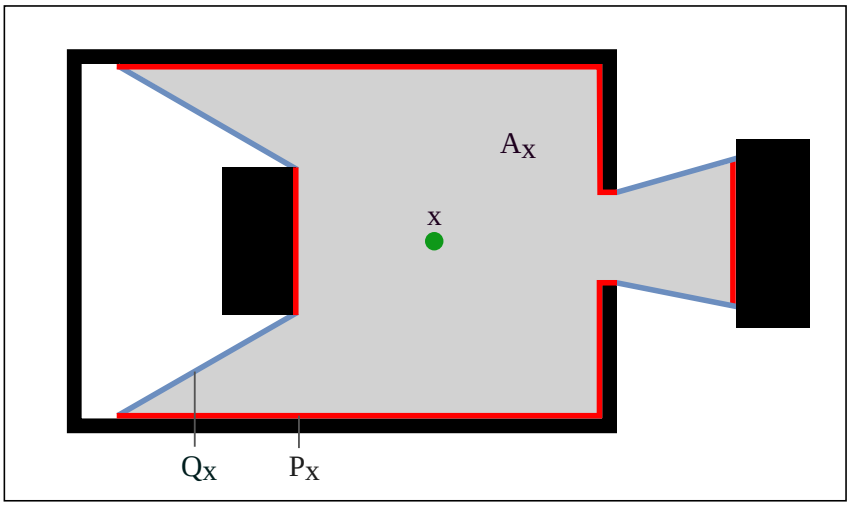


Abbildung 3.1: Schematische Darstellung der Grenzlinien und ausgewählter Kennzahlen eines Isovisten. x markiert den Standpunkt des Agenten. Schwarze Flächen stellen Wände dar.

Benedikt [14] bezeichnet die Menge der Isovisten, die entlang einer Trajektorie erfasst wurden als Isovistenfelder. Entlang einer solchen Bewegung eines Agenten durch eine Umgebung, verändern sich die erfassten Isovistenkennzahlen graduell. Dies stellt auch das grundlegende Vorgehen der Datengenerierung für die Wahrnehmungsmodellierung in diesem Kapitel dar: Es werden geospatiale Trajektorien durch simulierte Gebäudestrukturen generiert und bei jedem Schritt neben den Koordinaten des Agenten auch die Isovistenmessgrößen berechnet und abgespeichert. Prinzipiell kann die Berechnung der Isovisten im 3D-Raum erfolgen [37], die folgenden Untersuchungen beschränken sich jedoch auf Trajektorien auf einer 2D-Ebene und den korrespondierenden 2D Isovisten. Die Berechnung der Isovisten kann potentiell sehr aufwendig werden. Zur Berechnung der oben beschriebenen Kennzahlen ist es notwendig alle Eckpunkte sichtbarer Wände und Objekte zu bestimmen und zu verbinden. Feld, Werner und Linnhoff-Popien [46] konnten in ihrer Arbeit zeigen, dass *Area*, *Variance*, *Skewness* und *Circularity* mittels eines einfachen Ray-Scan Algorithmus approximiert werden können. Der in diesem Kapitel vorgestellte Ansatz baut auf diesen Ideen auf und erweitert sie um eine strahlenbasierte Berechnung von *Real-surface Perimeter* und *Occlusivity*.

3.3.2 Bayessche Inferenz

Im Feld der Bayesschen Statistik werden unbekannte Variablen nicht durch singuläre Werte beschrieben, sondern durch Wahrscheinlichkeitsverteilungen. Nachdem Daten beobachtet wurden, können diese Wahrscheinlichkeitsverteilungen mittels dem Bayesschen Theorem [8] aktualisiert werden:

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)} = \frac{P(x|y)P(y)}{\sum_{y \in Y} P(x,y)} \tag{3.1}$$

Die aktuell modellierte Wahrscheinlichkeitsverteilung $P(y)$ wird dabei *Prior* genannt. Das Bayessche Theorem erlaubt es nun, für jeden neu beobachteten Datenpunkt x die *Posterior* Verteilung $P(y|x)$ aus der aktuellen *Prior* Verteilung abzuleiten. Wird ein weiterer Datenpunkt beobachtet, wird dieses Vorgehen wiederholt, indem der *Posterior* als neuer Prior betrachtet wird. Dieses Vorgehen kann auch als eine ständige Modellierung und Quantifizierung von Unsicherheit interpretiert werden. Das wiederholte Aktualisieren der Wahrscheinlichkeitsverteilungen von unbekannten Variablen durch Anwendung des Bayesschen Theorems wird Bayessche Inferenz oder auch probabilistische Inferenz genannt. [15, 98]
Bayessche Inferenz ist einfach übertragbar auf das Gebiet des ML. Hierfür wird in Gleichung 3.1 x durch D ersetzt, die beobachteten Daten, sowie y durch θ, die unbekannten Parameter des ML-Modells. Alle Wahrscheinlichkeitsverteilungen sind zudem bedingte Wahrscheinlichkeitsverteilungen von m, der Klasse an möglichen probabilistischen Modellen, die in Betracht gezogen werden:

$$P(\theta|D,m) = \frac{P(D|\theta,m)P(\theta|m)}{P(D|m)} \tag{3.2}$$

Im Vergleich zu klassischen ML-Verfahren wird somit die direkte Optimierung von Modellparametern θ durch Bayessche Inferenz der bedingten Wahrscheinlichkeitsverteilungen ersetzt. Somit kann von einem Bayesschen Standpunkt aus der Vorgang des *Lernens* als die Transformation von *Prior* Annahmen mittels Daten in *Posterior* Annahmen beschrieben werden. [53]

3.3.3 Bayesian Surprise

Das Konzept von *Bayesian Surprise* wurde in [65] vorgestellt, als Ansatz zur Quantifizierung von Überraschung. Die Überraschung kann sich dabei auf unterschiedliche *Beobachter* beziehen, beispielsweise auf ein einzelnes Neuron, oder aber auch auf komplexe natürliche oder künstliche Systeme. Quantifiziert wird durch die Überraschung, wie beobachtete Daten einen Betrachter beeinflussen. Als Grundlage ihres Ansatzes nutzen die Autoren hierfür den Unterschied zwischen Posterior und Prior Annahmen des Beobachters. Diese Annahmen wiederum sind entsprechend der Bayesschen Betrachtungsweise bedingt auf der Menge der möglichen probabilistischen Modelle m. Oder in-

formell ausgedrückt, die Annahmen sind beschränkt durch die Hypothesen des Betrachters über die Welt.
Dieser Betrachtungsweise liegen zwei Annahmen zugrunde:

1. Überraschung kann nur dort auftreten, wo Unsicherheit herrscht. Exakt bekannte, deterministische und damit perfekt vorhersagbare Systeme enthalten für einen Beobachter keine Überraschung.

2. Überraschung ist subjektiv, und damit abhängig von der bisherigen Erfahrung eines Betrachters.

Die bisherige Erfahrung eines Betrachters beeinflusst demzufolge welche Hypothesen über die Welt für wie wahrscheinlich gehalten werden. Werden diese Hypothesen durch probabilistische Modelle m (Modellraum genannt) mit Parametern θ modelliert, so kann dies durch den Bayesschen Prior $P(\theta|m)$ erfasst werden. Die Beobachtung neuer Daten D führt nun dazu, dass diese Prior Verteilung $P(\theta|m)$ mittels des Bayesschen Theorems entsprechend Gleichung 3.2 in die Posterior Verteilung $P(\theta|D,m)$ aktualisiert wird. Da der Modellraum über die Modellierung konstant bleibt, kann die Konstante m für eine bessere Lesbarkeit entfernt werden: Die Prior Verteilung ist somit $P(\theta)$, die Posterior Verteilung $P(\theta|D)$. Entsprechend der Idee des *Bayesschen Surprise* ist die durch Daten D erzeugte Überraschung umso größer, je stärker sich der Posterior vom Prior unterscheidet. Ist der Posterior hingegen identisch zum Prior, erzeugen die Daten keinerlei Überraschung.
Diese Unterschiedlichkeit wird nach [65] mittels der Kullback-Leibler-Divergenz (KL-Divergenz) [78] bestimmt. Die Stärke der Überraschung S wird somit berechnet als:

$$S(D,m) = KL\Big(P(\theta|D)||P(\theta)\Big) = \int_m P(\theta|D)\log\left(\frac{P(\theta|D)}{P(\theta)}d\theta\right) \tag{3.3}$$

Die KL-Divergenz ist dabei ein Maß der Unterschiedlichkeit zweier Wahrscheinlichkeitsverteilungen. Aus der Perspektive der Informationstheorie betrachtet beschreibt die KL-Divergenz den entstehenden Verlust an Information, der eintritt, wenn eine Wahrscheinlichkeitsverteilung durch eine andere ersetzt würde. Unter dieser Sichtweise wird die Verwandtschaft zur Entropie einer Zufallsvariablen deutlich. Die KL-Divergenz wird deshalb auch als relative Entropie bezeichnet [16]. Für eine diskrete Zufallsvariable X mit n möglichen Ausprägungen ist die Entropie, wie in Unterabschnitt 2.2.2 erläutert, definiert als:

$$H(X) = -\sum_{i=1}^{n} P(x)\log(P(x)) \tag{3.4}$$

Diese Definition kann auf Verteilungen kontinuierlicher Zufallsvariablen X erweitert werden [16]:

$$H(X) = -\int P(X)\log(P(X))dX \tag{3.5}$$

Die KL-Divergenz zwischen zwei kontinuierlichen Verteilungen P und Q ist dann eine einfache Erweiterung:

$$\begin{aligned} KL(P||Q) &= -\int P(X)\log Q(X)dX - \Big(-\int P(X)\log P(X)dX\Big) \\ &= -\int P(X)\log\Big(\frac{Q(X)}{P(X)}\Big)dX \end{aligned} \tag{3.6}$$

3.4 Verwandte Arbeiten aus dem Bereich der Wahrnehmungsmodellierung

In der Literatur existiert eine Vielzahl von Arbeiten zur Analyse von Kartenrepräsentationen von räumlichen Umgebungen, die sich jedoch von ihrem Ansatz her grundlegend vom vorliegenden unterscheiden. Weber et al. [144] untersuchen beispielsweise die semantische Unterteilung von Gebäudeplänen in Räume, Zonen, Einheiten oder Stockwerke. Einen ähnlichen Fokus besitzen Arbeiten zur automatischen semantischen Annotation von Gebäudeplänen sowie darin enthaltenen Objekten, wie [3, 121]. Im Unterschied zu diesen Arbeiten ist der hier vorgestellte Ansatz jedoch weder geometrischer Natur, noch hat er zum Ziel existierende architekturelle Kartendarstellungen semantisch aufzubereiten. Vielmehr steht die den Gebäudeplänen sowie deren semantischen Einheiten zugrundeliegende Wahrnehmung von Raum und deren Modellierung im Fokus.
Dennoch existieren gewisse Überschneidungen mit verwandten Arbeiten, besonders aus dem Feld der mobilen Robotik, die ebenfalls zum Ziel haben Location-based Services zu realisieren. Die Arbeit von Buschka und Saffiotti [23] lässt sich beispielweise im Umfeld des Forschungsgebietes des *Simultaneous Localization and Mapping* (SLAM) [85] platzieren. Die Autoren beschreiben ein System aus visuellen Sensoren, die in der Lage sind, Räume zu detektieren und bereits besuchte Räume wiederzuerkennen. Der Fokus der vorliegenden Arbeit ist breiter, da das Ziel nicht nur die Wiedererkennung ist, sondern eine möglichst umfassende Wahrnehmungsmodellierung. Ein ähnliches Ziel verfolgen auch Anguelov et al. [6], die ein probabilistisches Framework zur Erkennung und Modellierung von Türen vorstellen. Das vorgestellte System basiert auf einer Kombination von 2D-Laserscans und Kameras. Die Erkennung von Türen ist ebenfalls im Fokus von Chen et al. [25], die hierfür Deep Learning Methoden basierend auf Kamerabildern einsetzen. Goerke und Braun [56] verwenden Supervised Learning Techniken zur semantischen Annotation. Ihre Datengrundlage basiert jedoch auf Laserscans mobiler Roboter. Diese Daten werden vorab als *Türen, Korridore, Freiraum, Raum* sowie *Unbekannt* gelabelt, so dass Supervised Learning Techniken darauf angewendet werden können. Der Einsatz von Unsupervised Learning Techniken führte, im Gegensatz zur vorliegenden Arbeit, den Autoren zufolge zu keinem befriedigenden Ergebnis.

Das zweite Kernelement des in diesem Kapitel vorgestellten Ansatzes ist die Modellierung Bayesscher Überraschung basierend auf Raumwahrnehmung. Das Konzept des Bayesian Surprise [65] zur Wahrnehmungsmodellierung wurde bereits in verschiedenen anderen Bereichen zum Einsatz gebracht. In [18] untersuchen die Autoren die Adaptation visueller Neuronen von Rhesusaffen auf wiederholte Stimuli. Hierfür setzen sie ein Bayessches Modell ein, das auf Bayesian Surprise basiert. Die Autoren konnten zeigen, dass das beobachtete Verhalten der Neuronen bei wiederholten Stimuli mit der verwendeten Modellierung übereinstimmte.

Eine andere Reihe von Forschungsarbeiten fokussiert sich auf die Frage, was menschliche Aufmerksamkeit in natürlichen Umgebungen auf sich zieht. [36, 64, 97] analysiert hierfür, ob mittels Bayesian Surprise vorhergesagt werden kann, wie gut Menschen in einem Bilderkennungsproblem abschneiden. Die Versuchsteilnehmer mussten hierbei in einer schnellen Abfolge von Bildern erkennen, ob die gezeigte natürliche Szenerie Tiere enthält. Die gewonnenen Ergebnisse zeigen, dass mittels Bayesian Surprise modellierte Überraschung hierfür ein guter Prediktor ist. Entsprechend eignet sich Bayesian Surprise als Metrik zur Vorhersage von menschlicher Aufmerksamkeit.

Die Arbeit [141] ist mit diesen Ansätzen verwandt, untersucht jedoch ob mittels Bayesian Surprise überraschende Ereignisse in Videos von Überwachungskameras erkannt werden können. Die Ergebnisse zeigen, dass dies der Fall ist und Bayesian Surprise bisherigen Ansätzen in dieser Problemstellung deutlich überlegen ist. Die Datengrundlage in [63] ist ähnlich, da auch hier Videos zum Einsatz kommen. Die Zielsetzung unterscheidet sich jedoch, da hier weniger der zeitliche Aspekt im Fokus ist, sondern die räumliche Lokalisation relevanter Bereiche im Vordergrund steht. Die Autoren stellen eine hohe Übereinstimmung des Bayesian Surprise Modells mit Messungen menschlicher Augenbewegungen fest. Sie schlussfolgern, dass überraschende Elemente die menschliche Wahrnehmung anziehen, und dies wiederum mittels Bayesian Surprise quantifiziert werden kann.

Die bisher genannten Arbeiten sind mit dem in diesem Kapitel vorgestellten Ansatz verwandt, da dieser ebenfalls Bayesian Surprise auf visuellen Daten zum Einsatz bringt. Die verwendeten, Isovisten-basierten Daten unterscheiden sich jedoch stark von den Video- oder Bilddaten der soeben vorgestellten Arbeiten, da in der vorliegenden Arbeit keine Objekt- oder Ereigniserkennung in natürlichen Szenarien das Ziel ist. Anstelle dessen wird die Raumwahrnehmung in Gebäudestrukturen untersucht. Die Hypothese ist, dass mittels Bayesian Surprise charakteristische Stellen in Gebäuden identifiziert werden können. Diese würden sich optimal zur Positionierung von Kartenmaterial oder anderen Informationen von LBS-Systemen eignen, da sie die menschliche Aufmerksamkeit auf sich ziehen.

3.5 Konzept zur Wahrnehmungsmodellierung

Der folgende Abschnitt ist in zwei Teile unterteilt: (1) die Vorstellung des entwickelten Frameworks zur Erzeugung von Isovistenmessungen entlang geospatialer Trajektorien innerhalb einer 3D Simulationsumgebung, sowie (2) die Wahrnehmungsmodellierung wiederkehrender Strukturen. Im Rahmen der Wahrnehmungsmodellierung wird zuerst das entwickelte Daten Preprocessing beschrieben, gefolgt von der Erzeugung temporaler Features. Anschließend werden die verwendeten Verfahren zum Unsupervised Learning von unbekannten Strukturen erläutert sowie das Konzept zu deren qualitativer, visueller Analyse im Karten- und Feature-Raum. Details hinsichtlich der genauen Implementierung und Parametrisierung der Algorithmen folgen in Abschnitt 3.6.

3.5.1 Isovisten Generierung

Der folgende Abschnitt präsentiert zunächst das entwickelte Framework zur Erzeugung eines 2D-Isovisten Datensets. Die Isovisten werden dabei entlang geospatialer Trajektorien berechnet, die eine 3D-Simulationsumgebung durchlaufen.
Die für die folgenden Evaluationen verwendeten Umgebungen der Simulation lassen sich in zwei Typen gliedern:

1. Umgebungen basierend auf realen Gebäudestrukturen

2. Umgebungen basierend auf synthetischen Gebäudegrundrissen

Die Erstellung ersterer, der Umgebungen basierend auf realen Gebäudestrukturen, beginnt mit der Vektorisierung realer Gebäudepläne. In diesem Schritt werden alle zusätzlichen Informationen der Pläne wie Einrichtungsgegenstände oder auch Türen entfernt, so dass lediglich Wände in der Vektor-Ausgabe verbleiben. Diese Vektordateien werden in einem zweiten Schritt in Blender [150], ein Open-Source 3D-Modellierungstool geladen, um eine 3D-Extrusion durchzuführen. Auf diese Weise entsteht eine einfache 3D-Repräsentation des Gebäudegrundrisses.
Die Erstellung zweiterer, der Umgebungen basierend auf synthetischen Gebäudegrundrissen, erfolgt direkt durch 3D-Modellierung in Blender.
Die Simulation selbst wurde mittels Unity [139] realisiert, einer 3-D Spiele-Engine und Entwicklungsumgebung. Nach dem Import der Blender 3D-Modelle in Unity wird für jede Umgebung ein sogenanntes Navigation Mesh [134] erzeugt. Dies ermöglicht eine automatische Navigation und Pfadfindung innerhalb der Umgebung.
Die Erzeugung der Trajektorien erfolgt je nach Umgebungstyp unterschiedlich. Bei den auf realen Gebäudestrukturen basierenden Umgebungen wird zunächst ein zufälliger Zielpunkt innerhalb des navigierbaren Bereichs gewählt. Dieser stellt das Navigationsziel eines Agenten innerhalb der Umgebung dar. Hat

der Agent das Navigationsziel erreicht, wird ein neuer zufälliger Zielpunkt bestimmt und das Verfahren wiederholt sich. Auf diese Weise können zufällige Trajektorien potentiell unendlicher Länge innerhalb der Umgebung realisiert werden.

Bei den auf synthetischen Gebäudegrundrissen basierenden Umgebungen erfolgt die Bestimmung der Zielpunkte nicht zufällig. Stattdessen werden vorab definierte Zielpunkte innerhalb der Umgebung fixiert. Erreicht der Agent den letzten definierten Zielpunkt, wird die Simulation beendet.

Die Bewegung und Navigation des Agenten auf einen Zielpunkt zu erfolgt dabei mittels eines Unity internen Navigationsalgorithmus.

Nach jedem Schritt des Agenten wird eine 2D-Isovistenberechnung basierend auf der entsprechenden Position durchgeführt. Nach erfolgter Berechnung werden die Isovistenkennzahlen zusammen mit der X- und Y-Koordinate des Agenten in eine Logdatei geschrieben.

Die Berechnung der Isovisten basiert dabei auf [14] und ermittelt die in Unterabschnitt 3.3.1 vorgestellten sechs Kennzahlen *Area, Real-surface Perimeter, Occlusivity, Variance, Skewness* und *Circularity*. Da eine diskrete, strahlenbasierte Isovistenberechnung verwendet wird, stellen die Kennzahlen lediglich Annäherungen an die tatsächlichen Isovistengrößen dar. Die Genauigkeit der Approximation kann jedoch durch Konfiguration der Anzahl an verwendeten Strahlen kontrolliert werden. Abbildung 3.2 zeigt eine Visualisierung der von der Position des Agenten ausgesandten Strahlen. Punkte im Raum, an denen die Strahlen mit Wänden der Umgebung kollidieren (Hitpoints genannt) werden erfasst und zur Berechnung der Isovistenkennzahlen verwendet.

Eine der komplexeren Berechnungen ist die notwendige Unterscheidung zwischen *Real-surface Perimeter* und *Occlusivity*. Benedikt [14] schreibt dazu, dass die *Occlusivity* eines Isovisten die Länge der verdeckten radialen Grenze R_x eines Isovisten V_x wiedergibt. Sie repräsentiert die Tiefe, in der Oberflächen der Umgebung sich, vom aktuellen Blickpunkt aus gesehen, gegenseitig verdecken. Um diese Kennzahl vom Anteil der tatsächlichen Oberflächen (*Real-surface Perimeter*) unterscheiden zu können, wurde ein Algorithmus entwickelt, der auf den Mesh-Triangles der Umgebung basiert:

Hierfür wird im Uhrzeigersinn für jeden ausgesendeten Strahl ein Vergleich des aktuellen Hitpoints mit dem des vorherigen Strahls durchgeführt. Hat der vorherige Strahl ein Mesh-Triangle getroffen, dessen Ecken keine Koordinate mit dem aktuell getroffenen Mesh-Triangle teilen, definieren wir das aktuelle Triangle als *unverbundenes Mesh-Triangle*. In diesem Fall wird der euklidische Abstand zwischen aktuellem und vorherigem Hitpoint der *Occlusivity* zugerechnet. Besitzen das aktuell getroffene Mesh-Triangle und das vorherige eine gemeinsame Eck-Koordinate, so wurde ein *verbundenes Mesh-Triangle* getroffen und die *Real-surface Perimeter* Kennzahl des Isovisten wird um den Abstand der Hitpoints erhöht. Abbildung 3.3 zeigt eine entsprechende Visualisierung der berechneten Linien innerhalb der Unity Engine. Rote Linien sind die vom aktuellen Standpunkt des Agenten ausgesandten Strahlen, grüne zei-

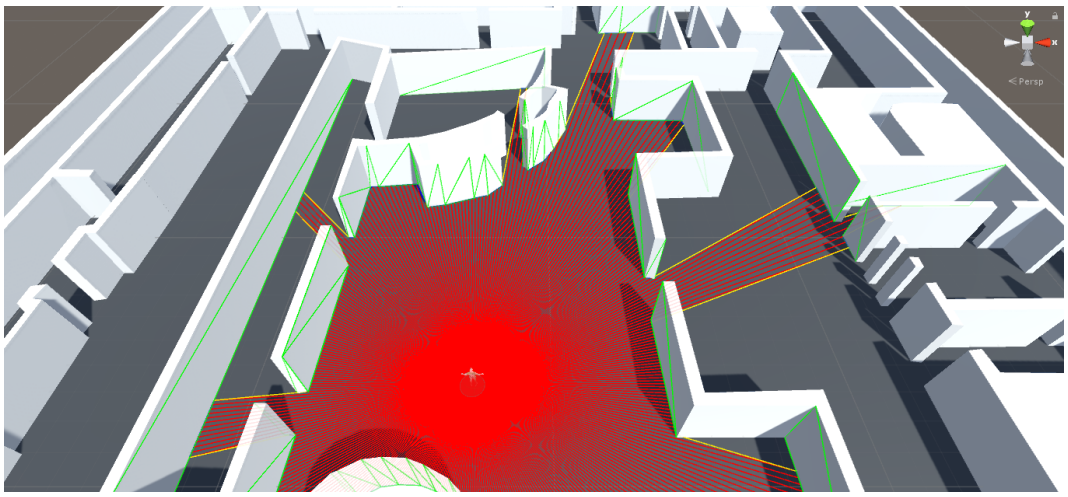


Abbildung 3.2: 3D-Visualisierung der Unity Simulation unter Verwendung einer realen Gebäudestruktur. Die Abbildung zeigt einen Agenten, der zur Isovistenberechnung 360 Strahlen (rote Linien) vom aktuellen Standort aus aussendet. Grüne, gelbe und blaue Linien beschreiben Elemente und Kennzahlen der Isovistenberechnung (siehe Abbildung 3.3).

gen die Kanten getroffener Mesh-Triangles. Blaue Linien markieren berechnete *Real-surface Perimeter* Distanzen und gelbe Linien *Occlusivity*. Am Beispiel der Wand unten links in Abbildung 3.3 ist zudem erkennbar, wie die Diskretisierung der Berechnung mit beschränkter Strahlenanzahl zu einem leichten Verlust der Genauigkeit führt. Der berechnete *Real-surface Perimeter* (blaue Linie unten links) fällt dabei zu gering aus, da kein Strahl die exakten Eckpunkte der Wand trifft.

3.5.2 Wahrnehmungsmodellierung wiederkehrender Strukturen

Basierend auf dem zuvor erläuterten Framework zur Erzeugung von Isovistenmessungen wird nun das entwickelte Konzept zur Wahrnehmungsmodellierung wiederkehrender Strukturen erläutert. Dieses enthält verschiedene Aspekte, begonnen mit einer Vorverarbeitung der erzeugten Daten im Daten Preprocessing, gefolgt von der Generierung temporaler Features. Anschließend erfolgt das Unsupervised Learning von unbekannten Strukturen sowie eine qualitative, visuelle Analyse im Karten- und Feature-Raum.

3.5.2.1 Daten Preprocessing

In einem ersten Schritt des Preprocessings werden die in den Logdateien gespeicherten Isovistenkennzahlen vektorisiert. Auf diese Weise werden 6-dimensionale Vektoren erzeugt. Jede Dimension dieser Vektoren repräsentiert

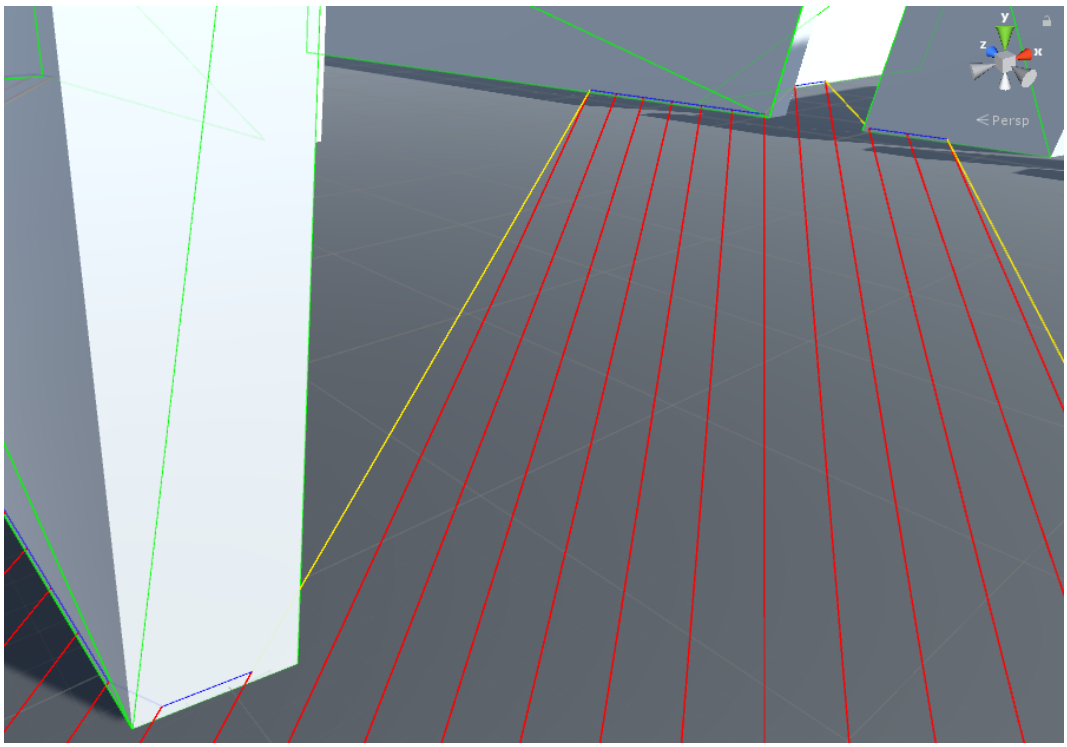


Abbildung 3.3: Visualisierung berechneter *Real-surface Perimeter* und *Occlusivity* Distanzen innerhalb der Unity Engine. Rote Linien zeigen vom aktuellen Standpunkt des Agenten ausgesandte Strahlen, grüne visualisieren Kanten getroffener Mesh-Triangles. Blaue Linien Markieren berechnete *Real-surface Perimeter* Distanzen und gelbe *Occlusivity*.

eines der sechs berechneten Isovistenfeatures. Da die meisten ML-Ansätze standardisierte Datensätze benötigen, um eine gute Modellierung zu erzeugen, wird ein Skalierungs- und Normalisierungsprozess durchgeführt. Einer der Gründe hierfür ist, dass besonders bei multidimensionalen Clustering-Verfahren die Distanz zwischen den jeweiligen Features der Daten eine wichtige Rolle spielt. Da die Features aber oftmals verschiedene Skalen besitzen, ist es sinnvoll, diese Distanzen erst nach durchgeführter Standardisierung zu bestimmen. [41] Durch diese Standardisierung wird sichergestellt, dass alle Isovistenfeatures wie *Area* oder *Occlusivity* gleichbehandelt werden und keines einen größeren Einfluss hat, nur weil die Werte anders skaliert sind. Der Skalierungs- und Normalisierungsprozess wird wie folgt durchgeführt: Zunächst wird von jedem Feature-Wert der Mittelwert des jeweiligen Features subtrahiert, wodurch die Daten zentriert werden. In einem zweiten Schritt werden die Daten skaliert, indem jeder Feature-Wert durch die Standardabweichung des jeweiligen Features dividiert wird.

Dieser Prozess hat eine weitere positive Auswirkung: Er entfernt den Einfluss, den unterschiedlich skalierte Simulationsumgebungen auf die Isovistenkenn-

zahlen ausüben. Nach durchgeführter Skalierung spielt es keine Rolle mehr, ob alle Simulationsumgebungen basierend auf Karten gleichen Maßstabs erzeugt wurden.

3.5.2.2 Erzeugung temporaler Features

An dieser Stelle ist es wichtig zu bedenken, dass die auf diese Weise erzeugten 6-dimensionalen Feature-Vektoren von statischer Natur sind. Obwohl die Isovistenberechnungen entlang einer Trajektorie durch die Simulation erzeugt wurden, enthält jeder einzelne Feature-Vektor nur die Isovistenkennzahlen einer einzelnen Position entlang dieser Trajektorie. Das Konzept von Bewegung und die daraus resultierende dynamische Veränderung der Wahrnehmung entlang eines Pfades sind somit bisher nicht direkt als temporale Komponente in den Features enthalten. Die Grundidee des nächsten Schrittes ist folglich die Berücksichtigung des temporalen Aspektes. Sie kann mit der Zielsetzung umschrieben werden, nicht nur die *Wahrnehmung von Raum* modellierbar zu machen, sondern auch die durch Bewegung verursachte Veränderung dieser Raumwahrnehmung.
Zur Umsetzung dieser Idee wurde ein weiterer Daten Preprocessing Schritt entwickelt, der die zeitliche Dimension der Daten nutzbar macht: Hierfür wird das Simple Moving Average (SMA) Verfahren [7] genutzt, das in der statistischen Analyse von Zeitreihen häufig eingesetzt wird. Für jeden Feature-Wert jedes Datenpunktes wird das Delta zwischen dem aktuellen Feature-Wert x_c und dem SMA von n vorhergehenden Daten berechnet:

$$x_c - \frac{1}{n}\sum_{i=1}^{n} x_{c-i} \tag{3.7}$$

Auf diese Weise wird die Anzahl der Features verdoppelt, so dass die resultierenden Feature-Vektoren nun 12-dimensional sind.

3.5.2.3 Unsupervised Learning von unbekannten Strukturen

Basierend auf diesen Preprocessing Schritten ist es nun möglich, automatisiert verstecke Strukturen in den Isovistenmessungen zu modellieren. Ziel dieses Schrittes ist es, die Isovistendaten in sinnvolle Cluster zu gruppieren, so dass jedes ein für Menschen nachvollziehbares Konzept repräsentiert. Dies könnte beispielsweise die Raumwahrnehmung in Räumen gegenüber der Raumwahrnehmung in Korridoren sein. Für das Auffinden versteckter Strukturen in ungelabelten Daten, wie den hier vorhandenen, sind besonders Unsupervised Machine Learning Techniken geeignet. Die Grundlagen hiervon wurden in Unterabschnitt 2.1.2 erläutert. Für die im Folgenden durchgeführten Experimente wird *k-means* [89], ein zentroidenbasierter Clustering Algorithmus, sowie DBSCAN [39], ein dichtebasierter Clustering Algorithmus eingesetzt.

Zur Bestimmung des Parameters k, der Anzahl an Clustern des k-mean Algorithmus, wird der Silhouettenkoeffizient berechnet [40]. Dies erlaubt es zum einen, einen ersten Einblick in die interne Struktur der Daten zu erhalten, zum anderen ermöglicht es eine geeignete Zahl an Clustern zur weiteren Evaluation zu wählen.

3.5.2.4 Qualitative, visuelle Analyse im Karten- und Feature-Raum

Die Evaluation der so erfolgten Modellierung erfolgt in einem anschließenden Schritt qualitativ mittels visueller Untersuchungen. Diese Untersuchung kann einerseits, unter Hinzunahme der X- und Y-Koordinaten der Datenpunkte, im sogenannten Karten-Raum erfolgen. Hierbei werden die einzelnen Datenpunkte an der entsprechenden Koordinate der Umgebung eingezeichnet und basierend auf ihrem Cluster-Label eingefärbt. Dies erlaubt eine semantische Interpretation der verschiedenen Cluster und die Untersuchung, ob diese für Menschen nachvollziehbare Konzepte repräsentieren.
Andererseits ist eine Untersuchung im sogenannten Feature-Raum möglich, also ohne Hinzunahme der räumlichen Koordinaten. Da die Daten je nach Repräsentation bis zu 12-dimensional sind, ist keine direkte Visualisierung möglich. Dieses Problem kann durch Verfahren zur Dimensionsreduktion gelöst werden. Für die folgende Evaluation kommt dabei die Principal Component Analysis (PCA) [109] zum Einsatz. Dabei werden die originalen Isovistenfeatures durch unkorrelierte Linearkombinationen derselben ersetzt. Anschließend können diese Hauptkomponenten genannten Featurekombinationen visualisiert werden. Durch eine Reduktion auf drei Hauptkomponenten können diese als Achsen einer $3D$-Visualisierung dargestellt werden [1]. Werden die Datenpunkte zusätzlich basierend auf ihrem Clusterlabel eingefärbt, ist ein visueller Vergleich der Clusterzugehörigkeit möglich. Dies erlaubt zum einen eine Einsicht in die tieferliegende Struktur der Wahrnehmungsmodellierung, und zum anderen einen Vergleich bezüglich der Unterschiedlichkeit verschiedener Umgebungen.

3.6 Evaluation der Wahrnehmungsmodellierung

Die Evaluation der zuvor beschriebenen Wahrnehmungsmodellierung erfolgt auf zwei strukturell unterschiedlichen Umgebungen. Beide Umgebungen basieren auf realen Gebäudeplänen. Die erste Umgebung entspricht einem Teil eines Universitätsgebäudes der Ludwig-Maximilians-Universität München (LMU). Dieses enthält sich stark wiederholende Strukturen bestehend aus Korridoren und sich ähnelnden Räumen (siehe Abbildung 3.5). Die zweite Umgebung basiert auf dem Gebäudeplan der Technischen Universität München (TUM) und enthält die Haupthalle, umgebende Räume, sowie vier die Universität umgebende Straßen. Im Vergleich enthält diese Umgebung unregelmäßigere Strukturen, die von großen Vorlesungssälen sowie Korridoren zwischen diesen geprägt sind (siehe Abbildung 3.6).

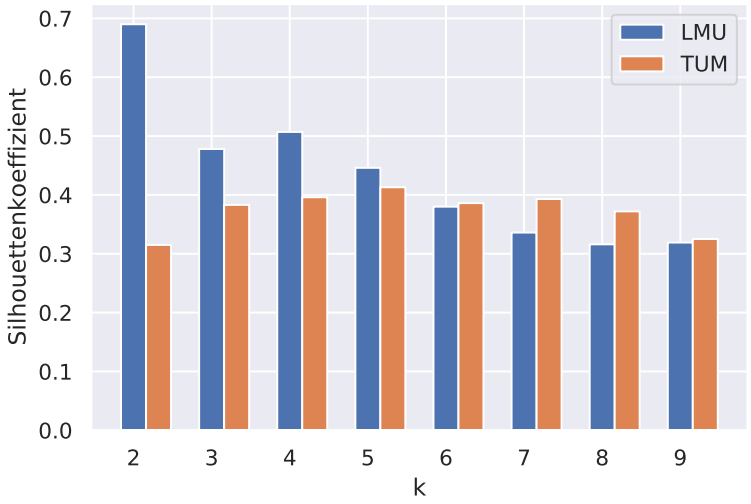


Abbildung 3.4: Silhouettenkoeffizient eines k-means basierten Clusterings unter Verwendung der sechs statischen Isovistenfeatures auf den Umgebungen LMU und TUM.

Basierend auf diesen zwei Umgebungen wurden in einem nächsten Schritt 370000 Isovistenmessungen entlang zufälliger Trajektorien auf der LMU Umgebung, sowie 220000 Messungen auf der TUM Umgebung durchgeführt. Jede Messung erfasst die in Unterabschnitt 3.3.1 erläuterten sechs statischen Isovistenfeatures: .
Diese 6-dimensionale Featuremenge bildet die Grundlage für den nächsten Schritt, die Modellierung versteckter Strukturen durch Clustering. Bei Verwendung des k-means Algorithmus muss die Anzahl an zu bildenden Clustern vorab durch den Hyperparameter k festgelegt werden. Um zu analysieren, ob es eine eindeutig optimale Belegung gibt, wurde der Silhouettenkoeffizient für eine verschiedene Anzahl k an Clustern, auf beiden Datensätzen LMU sowie TUM berechnet.

3.6.1 Evaluation im Karten-Raum

Wie in Abbildung 3.4 ersichtlich ist, erzeugt $k = 2$ den eindeutig höchsten Silhouettenkoeffizienten auf der LMU Umgebung. Auf der TUM Umgebung sind die Unterschiede weniger deutlich, mit einem Maximum bei $k = 5$. Es gibt jedoch keinen einzelnen, umgebungsübergreifend optimalen Wert für k. Es ist zudem interessant zu analysieren, ob die Anzahl an Clustern auch eine semantische Auswirkung bei der Zuordnung einzelner Datenpunkte hat. Aus diesen Gründen wurde die folgende visuelle Datenanalyse nicht nur mit einer einzelnen Anzahl k an Clustern, sondern mit einer variablen Anzahl durchgeführt. Im Folgenden werden einige ausgewählte Visualisierungen im Karten-Raum genauer analysiert. Weitere Visualisierungen mit anderen Werten für k befinden

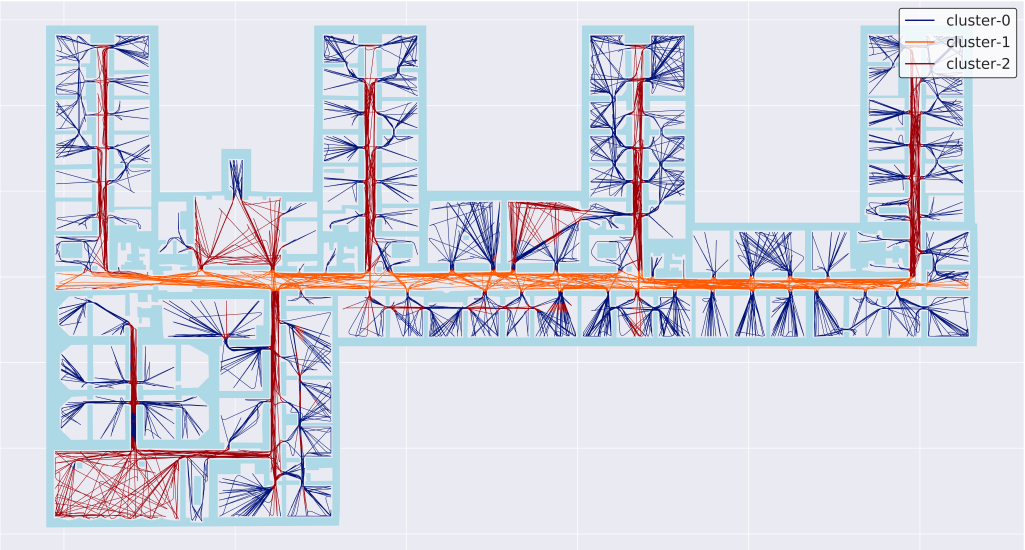


Abbildung 3.5: Visualisierung eines k-means basierten Clusterings mit $k = 3$, basierend auf den statischen Isovistenfeatures, die auf der LMU Umgebung gemessen wurden.

sich im Anhang.
Von einem semantischen Standpunkt aus betrachtet, erzeugte ein Wert von $k = 3$ die interessantesten Ergebnisse auf der LMU Umgebung. Wie in Abbildung 3.5 ersichtlich ist, wurden dabei drei verschiedene Strukturen der Umgebung in getrennte Cluster separiert. So wurde eine klare Trennung zwischen dem großen horizontalen Korridor (Cluster-1), den großteils vertikalen Korridoren (Cluster-2), sowie den kleineren Räumen (Cluster-0) deutlich. An dieser Stelle ist es wichtig zu bedenken, dass menschliche Konzepte nicht notwendigerweise durch die Cluster reflektiert werden müssen. Somit ist die semantische Bedeutung eines Clusters immer menschlicher Interpretation unterworfen.
Abbildung 3.6 zeigt die Visualisierung des k-means basierten Clusterings mit $k = 4$ auf der TUM Umgebung. Auch wenn die hier separierten Strukturen im Vergleich zur LMU Umgebung etwas weniger klar getrennt sind, zeigen sich doch deutliche semantische Einheiten. Die durch die Isovisten erfasste Wahrnehmung an Standorten der vier die Universität umgebenden Straßen wurden klar Cluster-3 zugeordnet. Hier dominieren sehr weite Sichtfelder in horizontale oder vertikale Richtung, während die Sicht in die jeweils andere Richtung stark beschränkt ist. Cluster-1 gruppiert Wahrnehmungen der großen, zentral gelegenen Haupthalle, sowie der zwei links und rechts davon gelegenen Hallen. Kleinere Räume sind großteils Cluster-0 zugeordnet, wobei hier die Trennung zu Cluster-2 semantisch nicht immer eindeutig zu erfassen ist.
Neben dem distanzbasierten Clusteringverfahren k-means wurde auch das dichtebasierte Clusteringverfahren DBSCAN evaluiert. Da hier die Anzahl an Clustern nicht direkt spezifiziert, sondern nur indirekt über zwei Dichteparameter ϵ und $minPTS$ beeinflusst werden kann, ist es deutlich schwieriger eine für

menschliche Interpretation sinnvolle Anzahl an Clustern zu finden. Zudem war es aufgrund der hohen Arbeitsspeicheranforderungen von DBSCAN nicht möglich die komplette Menge an Isovistendaten zu clustern. Abbildung 3.7 zeigt ein beispielhaftes Resultat eines DBSCAN Clusterings mit $\epsilon = 3, \text{minPTS} = 2500$, der ersten 100000 statischen Isovistenmessungen auf der LMU Umgebung. Das von DBSCAN erzeugte *cluster- -1* (siehe Abbildung 3.7) enthält dabei Outlier, die keinem der anderen Cluster zugeordnet werden konnten, jedoch auch untereinander kein Cluster bilden.

Aufgrund dieser Schwierigkeiten bei der Verwendung von DBSCAN wurden die weiteren Analysen basierend auf k-means Clustering durchgeführt.

Nach der Betrachtung der statischen Isovistenfeatures, und dem Auffinden von Strukturen, die als Räume oder Gänge interpretiert werden könnten, erfolgt nun eine Analyse der dynamischen Features. Hierfür wurde, wie in Unterunterabschnitt 3.5.2.2 erläutert, für jeden Feature-Wert jedes Datenpunktes das Delta zwischen dem aktuellen Feature-Wert und dem SMA von n vorhergehenden Daten berechnet. Auf diese Weise kann die temporale Dimension der Daten erfasst werden. Die erneute Anwendung eines k-means Clusterings auf diesen rein temporalen Features erzeugt bei Analyse im Karten-Raum ein gänzlich anderes Bild. Abbildung 3.8 zeigt das Ergebnis der LMU Umgebung, bei Verwendung von $k = 3$ Clustern und $n = 5$, also einer Delta-Berechnung zum Mittel der 5 vorhergehenden Datenpunkte. Es ist deutlich sichtbar, dass semantisch andersartige Cluster entstehen, die vor allem Datenpunkte an Durchgängen, speziell an Türdurchgängen enthalten. Diese Interpretation leuchtet intuitiv ein, da besonders Türdurchgänge Komponenten von Gebäuden sind, die oftmals Strukturen unterschiedlicher Art verbinden. Aus diesem Grund führt eine Bewegung durch diese hindurch zu Veränderungen in der Wahrnehmung. Die Resultate zeigen, dass eine Modellierung dieser dynamischen Wahrnehmungsveränderung durch Erzeugung temporaler Isovistenfeatures möglich ist.

Durch Kombination der statischen und dynamischen Cluster kann eine kombinierte Menge an Datenlabels erzeugt werden, die beide Aspekte abbildet. Abbildung 3.9 zeigte die Visualisierung des so erzeugten Datensets. Eine mögliche semantische Interpretation der Cluster könnte sein:

Cluster-0 (Blau): Kleine Räume

Cluster-1 (Orange): Große Korridore

Cluster-2 (Rot): Kleine Korridore

Cluster-3 (Violett): Große Räume

Cluster-4 (Grün): Durchgänge (Türen, etc.)

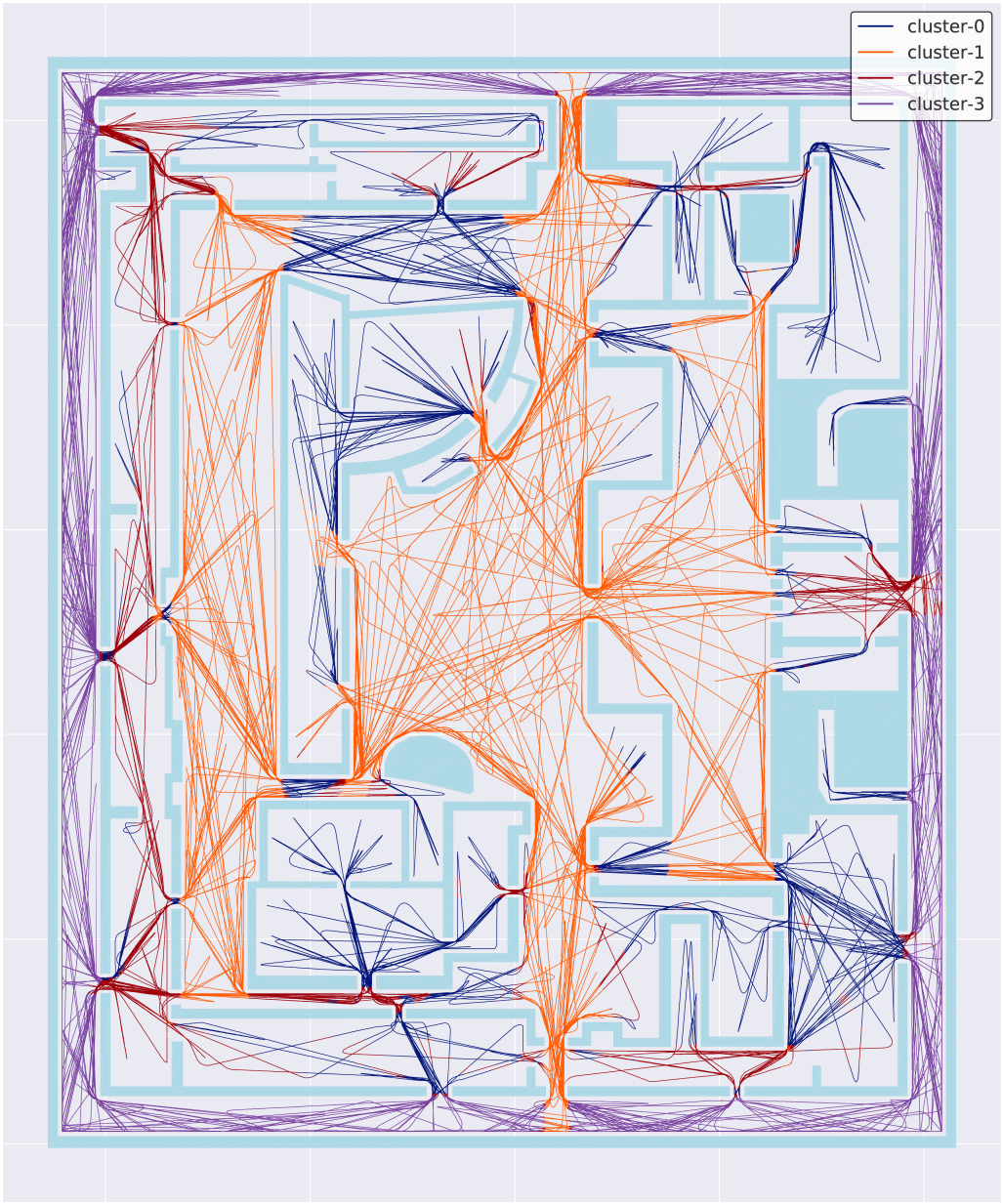


Abbildung 3.6: Visualisierung eines k-means basierten Clusterings mit $k = 4$, basierend auf den statischen Isovistenfeatures, die auf der TUM Umgebung ermittelt wurden.

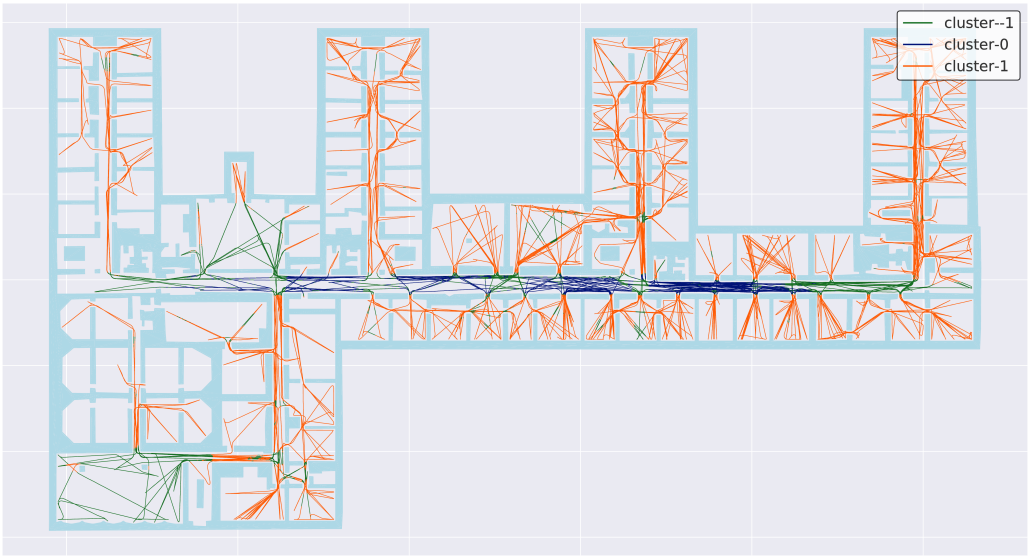


Abbildung 3.7: Visualisierung eines DBSCAN-basierten Clusterings mit $\epsilon = 3, \text{minPTS} = 2500$, basierend auf den statischen Isovistenfeatures die auf der LMU Umgebung gemessen wurden. Da es aus Performancegründen nicht möglich war, die gesamte Menge an 370000 Datenpunkten zu clustern, wurden lediglich die ersten 100000 Datenpunkte betrachtet.

3.6.2 Evaluation im PCA Feature-Raum

Um nach der Analyse im Karten-Raum eine tiefere Intuition dafür zu erlangen, wie die Unsupervised Machine Learning Methoden die Wahrnehmung modellieren, wurde eine visuelle Datenanalyse im Feature-Raum durchgeführt. Durch Visualisierung der Ergebnisse verschiedener Clusterkonfigurationen kann so ein tieferes Verständnis der in den Isovistendaten vorhandenen Struktur erlangt werden. Zu diesem Zweck wurde wie in Unterunterabschnitt 3.5.2.4 erläutert, eine Principal Component Analyse (PCA) durchgeführt. Hierbei wurden die originalen Isovistenfeatures durch diejenigen drei Hauptkomponenten ersetzt, die die Varianz in den Daten maximal erklären. Jede der drei Achsen der folgenden Visualisierungen entspricht einer dieser Hauptkomponenten, während die Farbe eines Datenpunktes das jeweilige Clusterlabel darstellt.

Abbildung 3.10 zeigt das Resultat dieser Schritte basierend auf den auf der LMU Umgebung erfassten Isovistendaten. Die Färbung der Datenpunkte der Abbildungen 3.10a-3.10d basiert dabei auf unterschiedlichen Werten k des verwendeten k-means Clusterings. In dieser Feature-Raum basierten Visualisierung, ohne X,Y-Koordinaten, wird zunächst unabhängig vom verwendeten Parameter k die klare Trennung zwischen einem linken Cluster (Orange) und den anderen Clustern rechts deutlich. Diese klare Trennung spiegelt die Tatsache wieder, dass die LMU Umgebung ihren höchsten Silhouettenkoeffizienten (0,69) bei der Verwendung von zwei k-means Clustern zeigte. Bei Betrachtung

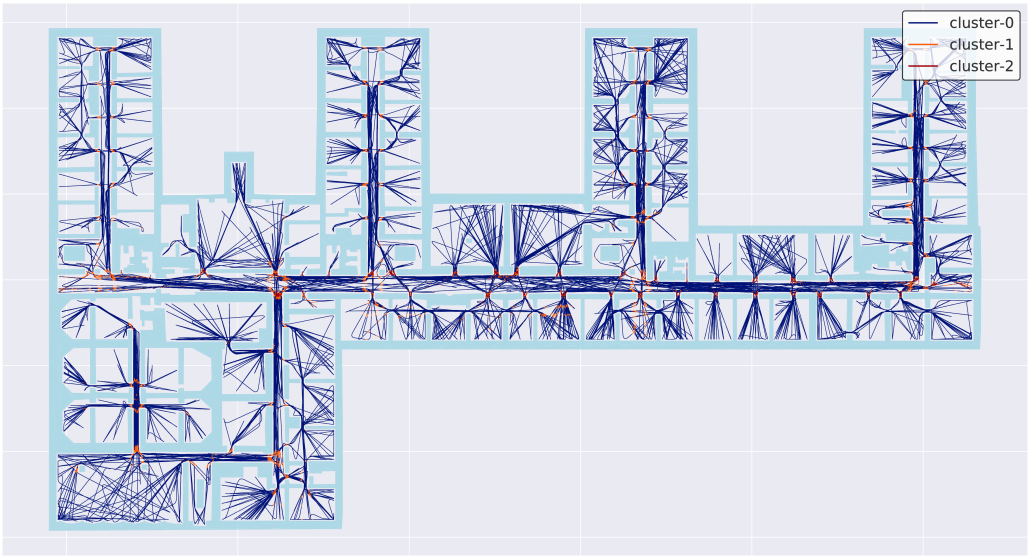


Abbildung 3.8: Visualisierung eines k-means basierten Clusterings mit $k = 3$, basierend auf den dynamischen Isovistenfeatures, erzeugt mittels SMA $n = 5$ auf der LMU Umgebung.

im Karten-Raum, also bei Verwendung der X,Y-Koordinaten der Datenpunkte als Achsen, wurde die Bedeutung dieses auch bei $k = 3$ existenten Clusters deutlich: Alle Isovistenmessungen dieser Datenpunkte wurde an Positionen erfasst, die sich im großen Korridor in der Mitte der Umgebung befinden (siehe Abbildung 3.5).
Abbildung 3.11 stellt die LMU und TUM Umgebungen gegenüber. Hier zeigt sich, dass sich auch im PCA Feature-Raum die unterschiedliche Struktur der Umgebungen widerspiegelt. Während in der LMU Umgebung eine sehr klare Struktur zu erkennen ist, sind die Übergänge der Strukturen der TUM Umgebung weniger deutlich. Dennoch können einzelne Bereiche unterschieden werden, die eine semantische Interpretation bei Betrachtung im Karten-Raum (siehe Abbildung 3.6) ermöglichen.

3.7 Konzept zur Modellierung Bayesscher Überraschung

Die in den vorhergehenden Abschnitten präsentierten Ergebnisse zeigen, dass wiederkehrende räumliche Strukturen auch zu wiederkehrenden Mustern in den entwickelten Wahrnehmungsmodellierungen führen. Je nach verwendeter Modellierung ist es so möglich, unterschiedliche statische oder dynamische Aspekte zu erfassen. Im Folgenden wird die so geschaffene Grundlage der Wahrnehmungsmodellierung weiterentwickelt. Die untersuchte Fragestellung ist, ob es möglich ist, die Isovisten-basierte Wahrnehmungsmodellierung zur Identifi-

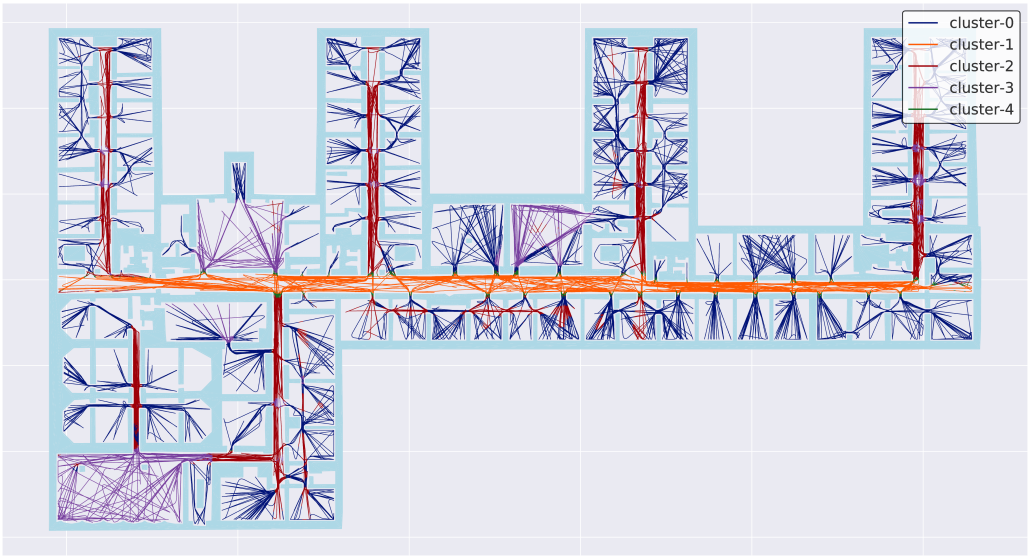
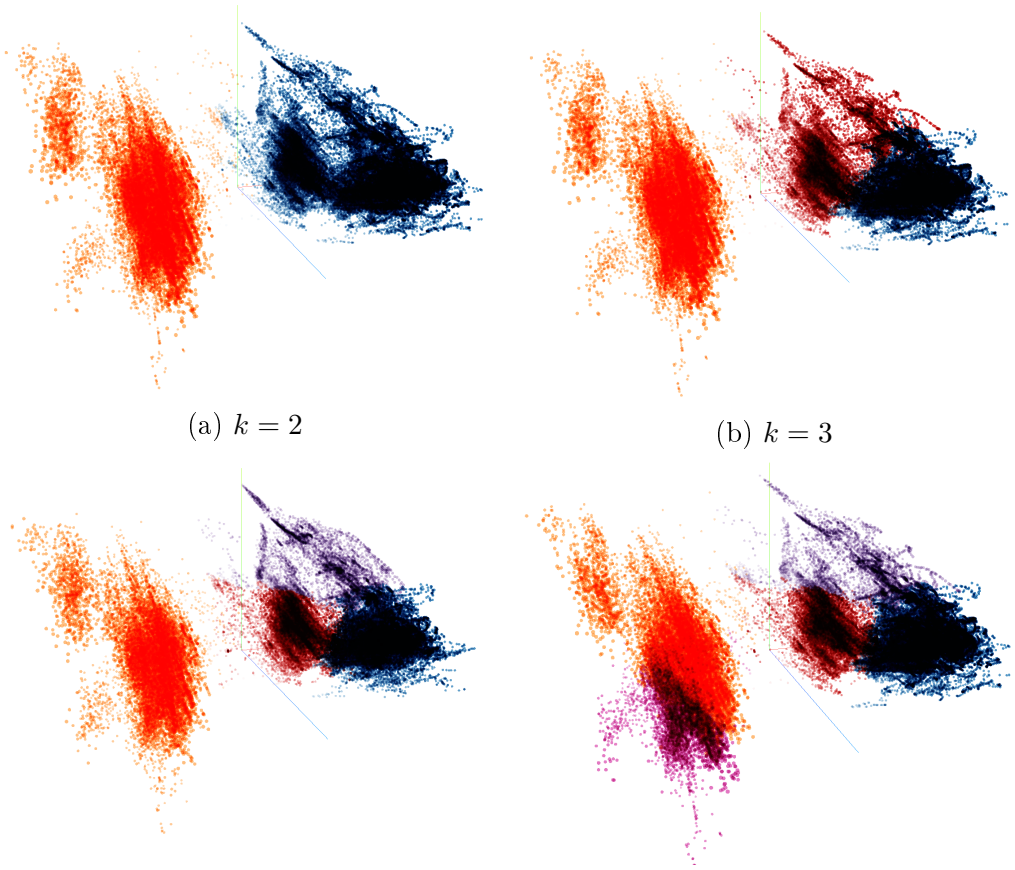


Abbildung 3.9: Visualisierung der Clusterzuordnungen nach Kombination der statischen und dynamischen Clusterlabels. K-means $k = 3$, SMA $n = 5$.

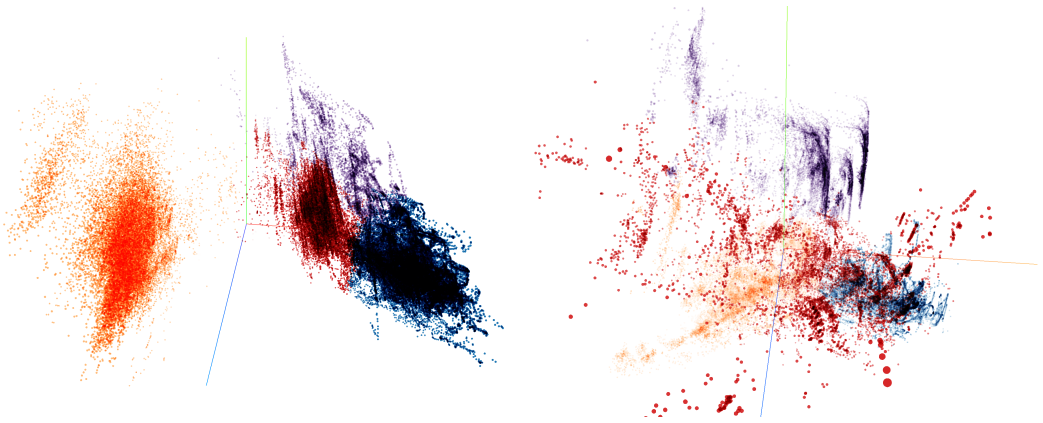
kation charakteristischer Stellen in Gebäuden zu verwenden. Der untersuchte Ansatz ist hierbei die Modellierung von Überraschung mittels Bayesian Surprise.
Das grundlegende Konzept ist wie folgt: Zunächst erfolgt entsprechend Unterabschnitt 3.5.1 die Generierung und Sammlung von Isovistendaten. Hierbei kommen zum einen Umgebungen basierend auf synthetischen Gebäudegrundrissen zum Einsatz. Ihre Erstellung erfolgt direkt durch 3D-Modellierung in Blender. Ziel der Untersuchungen auf diesen Umgebungen ist es, verschiedene zentrale Elemente der Modellierung Bayesscher Überraschung gezielt zu untersuchen. Zum anderen kommen Umgebungen zum Einsatz, die auf realen Gebäudestrukturen basieren. Ziel ist hier die Untersuchung, ob die Erkenntnisse der synthetischen Umgebungen auf reale Gebäudestrukturen übertragbar sind. Zudem kann hier analysiert werden, ob Trajektorien durch diese realen Strukturen anhand der Bayesian Surprise charakterisiert werden können.
An dieser Stelle ist es wichtig zu beachten, dass die Umgebungen basierend auf realen Gebäudestrukturen, keine Türen enthalten, während in den synthetischen Gebäudestrukturen abstrahierte Türen existieren, die stets geschlossen sind. Dem Agenten ist es nicht möglich durch diese Türen hindurchzusehen, sie schränken den Sichtbereich somit ein. Es ist dem Agenten jedoch möglich sich durch diese Türen hindurch zu bewegen und somit Räume zu wechseln. Eine genauere Analyse der Struktur und dahinterliegenden Konzepte der verwendeten Umgebungen findet sich in Unterabschnitt 3.8.1.
In Unterabschnitt 3.3.3 wurde erläutert, dass die theoretische Formulierung von Bayesian Surprise agnostisch gegenüber des verwendeten Verteilungstypen ist. Folglich muss die Entscheidung, welche Art von Verteilung zur Model-



(a) $k = 2$ (b) $k = 3$

(c) $k = 4$ (d) $k = 5$

Abbildung 3.10: PCA Dekomposition der auf der LMU Umgebung erfassten statischen Isovistendaten. Achsen entsprechen den drei PCA Hauptkomponenten. Die Farbe der Datenpunkte entspricht einer k-means basierten Clusterzuordnung bei Verwendung verschiedener Werte für k.



(a) LMU (b) TUM

Abbildung 3.11: Visualisierung einer PCA Dekomposition der statischen Isovistenfeatures, gesammelt auf der LMU- beziehungsweise TUM-Simulationsumgebung. Jede Achse repräsentiert eine der drei Hauptkomponenten, die die Varianz in den Daten bestmöglich erklären. Datenpunkte sind basierend auf einem k-means Clustering mit $k = 4$ eingefärbt.

lierung des Systems verwendet wird, abhängig vom Anwendungsfall getroffen werden. Wenn das Ziel die Durchführung von Bayesscher Inferenz (siehe Unterabschnitt 3.3.2) ist, ist es wichtig zu beachten, dass die Wahl des Verteilungstypen einen Einfluss auf die Berechnungskomplexität hat. Bei Verwendung von komplexen Verteilungen ist eine exakte Bestimmung des Bayesschen Posteriors meist unmöglich, da die Berechnung exponentiell in der Anzahl der Parameter ist [17]. Einer der effizientesten Wege, um exakte Bayessche Inferenz durchführen zu können, ist die Verwendung von sogenannten Conjugate Priors, da hier die Berechnung des Posteriors analytisch möglich ist. Der entscheidende Aspekt bei der Verwendung von Conjugate Priors ist, dass der Posterior zur selben funktionalen Familie wie der Prior gehört.
Im Rahmen der folgenden Evaluationen werden die Daten mittels getrennter Multinomialverteilungen je Feature modelliert. Anders ausgedrückt wird angenommen, dass die Daten für jedes Feature einer Multinomialverteilung folgen. Dem Prinzip von Conjugate Priors folgend wurde die funktionale Form des Priors $P(M)$ so gewählt, dass der Posterior $P(M|D)$ die selbe Form besitzt. Bei der Modellierung mittels Multinomialverteilungen kann gezeigt werden, dass in diesem Fall die korrekte Form von $P(M)$ die Dirichlet Verteilung ist [87].
Diese besitzt die Wahrscheinlichkeitsdichte:

$$P(M) = \frac{1}{B(\alpha)} \prod_{i=1}^{K} x_i^{\alpha_i - 1}, \tag{3.8}$$

mit

$$B(\alpha) = \frac{\prod_{i=1}^{K} \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{K} \alpha_i)} \tag{3.9}$$

und Konzentrationsparametern $\alpha_1, ..., \alpha_K > 0$, K der Anzahl an Kategorien und $\Gamma(.)$ der Gammafunktion.
Da die Multinomialverteilung eine diskrete Wahrscheinlichkeitsverteilung ist, während die Isovistenmessung in Form kontinuierlicher Features vorliegt, ist es notwendig die Daten zu diskretisieren. Hierfür werden die Daten D jedes Features in eine fixe Anzahl k aneinandergrenzender Intervalle (bins) b gleicher Breite eingeteilt. Intervall b_1 beginnt dabei bei $min(D)$ und Intervall b_k endet bei $max(D)$.
Ein offensichtlicher Einwand gegen die soeben vorgestellte Modellierungsvariante ist, dass die hier notwendige Diskretisierung zu einem Verlust an numerischer Präzision führt. Von diesem Standpunkt aus gesehen wäre die Verwendung einer kontinuierlichen Verteilung die bessere Wahl, um kontinuierliche Features zu modellieren. Auch wenn die Modellierung im vorliegenden Kontext beispielsweise mittels kontinuierlicher Normalverteilungen möglich gewesen wäre, wurde dennoch eine bewusste Entscheidung gegen diesen Weg getroffen, da hier die zu vermutende Multimodalität der Daten in der Modellierung nicht abgebildet werden könnte. Abhilfe könnte wiederum die Verwendung komplexerer Wahrscheinlichkeitsverteilungen schaffen, die Multimodalität abbilden können (zum Beispiel Gaußsche Mischmodelle [16]). Hier wäre es dann jedoch nicht mehr möglich den Bayesschen Posterior exakt analytisch zu bestimmen und es müssten approximative Inferenzmethoden zum Einsatz gebracht werden (Details zu approximativen Inferenzmethoden im probabilistischen ML folgen in Abschnitt 4.4).
In der Bayesschen Statistik enkodiert der Prior, wie in Unterabschnitt 3.3.2 erläutert, die Annahmen bezüglich der betrachteten Wahrscheinlichkeitsverteilung noch bevor Daten beobachtet wurden. Folglich kann die Wahl der Prior Verteilung einen starken Einfluss auf das resultierende Modell haben. Dies ist insbesondere zu Beginn der Fall, wenn die Menge an beobachteten Daten noch gering ist. Mit Sammlung von zunehmend mehr Daten und wiederholtem Updaten des Priors in den Posterior reduziert sich der Einfluss der initialen Wahl des Priors. Für die folgenden Evaluierungen wurde eine Gleichverteilung über die k Intervalle je Feature als Prior gewählt. Es wird somit die selbe initiale Wahrscheinlichkeit für alle Intervalle angenommen.
Nach jedem Schritt, in dem ein neuer Datenpunkt beobachtet wurde, wird mittels Bayesscher Inferenz der aktuelle Prior jedes Features in den neuen Posterior upgedated. Darauf folgt die Bestimmung des Bayesian Surprise durch Berechnung der KL-Divergenz zwischen dem jeweiligen Prior $P(M)$ und Posterior $P(M|D)$: $KL(P(M|D)||P(M))$. Auf diesem Weg wird die Überraschung pro Feature, pro Schritt entlang der Trajektorie berechnet. Um eine einzelne, kombinierte Maßzahl der Überraschung zu erhalten, kann eine (gewichtete)

Summe der Einzelüberraschungen erzeugt werden.
Neben diesen statistischen Aspekten, die die Modellierung beeinflussen, hat auch die gewählte Schrittlänge innerhalb der Simulationsumgebung einen Einfluss auf das resultierende Modell: Diese legt fest, wie viele Messungen und somit Datenpunkte je Länge der Trajektorie erfasst werden. Da jeder Datenpunkt zu einem Update des Modells führt und damit zu einer Berechnung von Bayesian Surprise, beeinflusst der Schrittlängenparameter die Entwicklung des Modells über die Zeit. Wird eine große Schrittlänge gewählt, werden nur wenige Messungen entlang der Trajektorie durchgeführt. Folglich kann es dann vorkommen, dass Wahrnehmungsveränderungen, die durch Strukturänderungen hervorgerufen werden, die kleiner als die Schrittlänge sind (zum Beispiel sehr kleine Räume oder Kreuzungen in Gängen), verpasst werden. Auf die gleiche Weise würde aber eine sehr kleine Schrittlänge das Modell auf einer großen Anzahl von nur minimal abweichenden Messungen überkonditionieren.

3.8 Evaluation von Bayesscher Überraschung

Im folgenden Abschnitt wird nun zunächst das zur Evaluation verwendete Experimentalsetup vorgestellt. Zunächst werden die unterschiedlichen, zur Evaluation verwendeten Umgebungen präsentiert und der jeweilige Fokus der Untersuchung dargelegt. Darauf folgt die Darstellung der verwendeten Parametrisierung, sowohl der Modellierung mittels Bayesscher Überraschung, als auch der sonstigen Evaluationsparameter. Abschließend werden die Evaluationsergebnisse im Detail präsentiert und diskutiert.

3.8.1 Experimentalsetup

Zur Evaluation des vorgestellten Ansatzes wurden sieben verschiedene Umgebungen verwendet. Fünf von diesen basieren auf speziell erzeugten, synthetischen Gebäudegrundrissen. Das Ziel ist hier die Untersuchung verschiedener zentraler Elemente der Modellierung Bayesscher Überraschung. Zwei der Umgebungen basieren hingegen auf realen Gebäudestrukturen zur Untersuchung der Echtwelt Eignung des Ansatzes. Im Folgenden werden nun die einzelnen Umgebungen kurz präsentiert und der Fokus der Untersuchung dargestellt.
Die Umgebung **BasicSimple** stellt das einfachste untersuchte Szenario dar: Eine Sequenz von identisch geformten Räumen, die sich von Türen getrennt wiederholen. Der Fokus liegt hier auf der Adaptierung des Modells an die Struktur der Umgebung. Die Überraschung sollte sich hier entlang der Trajektorie zunehmend abschwächen.
Die Umgebung **Alternating** ist vergleichsweise ähnlich zur vorherigen. Die Räume sind hier jedoch nicht durch Türen getrennt, sondern durch kleinere Verbindungsräume. Somit wechseln sich hier zwei Typen von Räumen wiederholt ab. Von dieser Umgebung wurde zudem eine Variante **AlternatingDoors** erzeugt, bei der die Verbindungsräume erneut durch Türen abgetrennt sind.

Die Hypothese zum Verhalten der Bayesian Surprise ist hier wie folgt: Die Modellierung sollte einerseits in der Lage sein, die zwei unterschiedlichen Typen von Räumen zu unterscheiden, was sich in erhöhter Überraschung bei einem Raumwechsel zeigt. Andererseits müsste die gleichmäßige Wiederholung dieser Wechsel zu einer Adaptation des Modells führen, was die absolute Stärke der Überraschungen entlang der Trajektorie reduzieren sollte.

Ziel der Umgebungen **AlternatingSurprise** und **AlternatingSurprise-Doors** ist die Untersuchung des Verhaltens der Modellierung bei Adaptation an sich wiederholende Strukturen, vergleichbar mit den vorherigen Umgebungen, die dann jedoch durch das plötzliche Auftreten einer stark abweichenden Struktur (in Form eines großen Raums) unterbrochen wird. Es ist zu erwarten, dass die Modellierung hier eine starke Überraschung zeigt. Diese sollte jedoch nicht dazu führen, dass das zuvor gelernte Konzept der sich regelmäßig wiederholenden Strukturen komplett verworfen wird.

Neben den soeben beschriebenen synthetischen Gebäudegrundrissen kommen Umgebungen basierend auf realen Grundrissen zum Einsatz. Die Umgebung **LMU** basiert auf dem bereits in Abschnitt 3.6 verwendeten Universitätsgebäude der Ludwig-Maximilians-Universität München. Der gewählte Ausschnitt unterscheidet sich jedoch und beschränkt sich auf den unteren, linken Bereich (siehe Abbildung 3.17). Die Umgebung **TUM** basiert ebenfalls auf dem bereits zuvor verwendeten Gebäudegrundriss der Technischen Universität München (TUM). Hier ist der untersuchte Ausschnitt identisch mit der vorherigen Evaluation in Abschnitt 3.6 und enthält die Haupthalle sowie umgebende Räume.

Zur Erzeugung der Isovistenmessungen wurden, wie zuvor in Unterabschnitt 3.5.1 beschrieben, 360 Strahlen vom Standpunkt des Agenten aus gesendet. Dieser Wert bot in der Mehrzahl der Fälle einen guten Mittelweg zwischen Isovistengenauigkeit und Performance. Dennoch führten in einer beschränkten Anzahl an Fällen Messungenauigkeiten zu unerwünschten Effekten im erzeugten Modell. Diese Fälle werden im Detail in den folgenden Abschnitten diskutiert. Die Bewegung des Agenten erfolgt bei den synthetischen Umgebungen vordefiniert, horizontal von links nach rechts durch die Umgebung. Bei den LMU und TUM Umgebungen folgt der Agent komplexeren Routen mit mehreren Richtungswechseln.

Die in Abschnitt 3.7 beschriebene Modellierung der Bayesschen Überraschung wurde wie folgt parametrisiert: Alle Features wurden mittels getrennten Dirichlet Verteilungen mit $k = 10$ modelliert. Folglich wurden die Messwerte jedes Features in 10 aneinandergrenzende Intervalle gleicher Breite eingeteilt. Zu Beginn wurden gleichverteilte Priorverteilungen verwendet, die allen Intervallen gleiche Wahrscheinlichkeiten zuordnen. Die Schrittlänge zwischen zwei Updates der Modelle wurde so gewählt, dass diese auf einer geraden Trajektorie in etwa einen Meter beträgt. Dieser Wert hat sich auf den evaluierten Umgebungen als geeigneter Mittelweg zwischen der Anzahl an Modellupdates und der benötigten Größe der Strukturänderungen erwiesen. Eine größere Schrittlänge würde zu einer verringerten Anzahl an Messungen entlang der

Trajektorie führen, wodurch kleinere Strukturänderungen der Umgebung nicht erfasst würden.

3.8.2 Adaptierung an gleichartige Strukturen

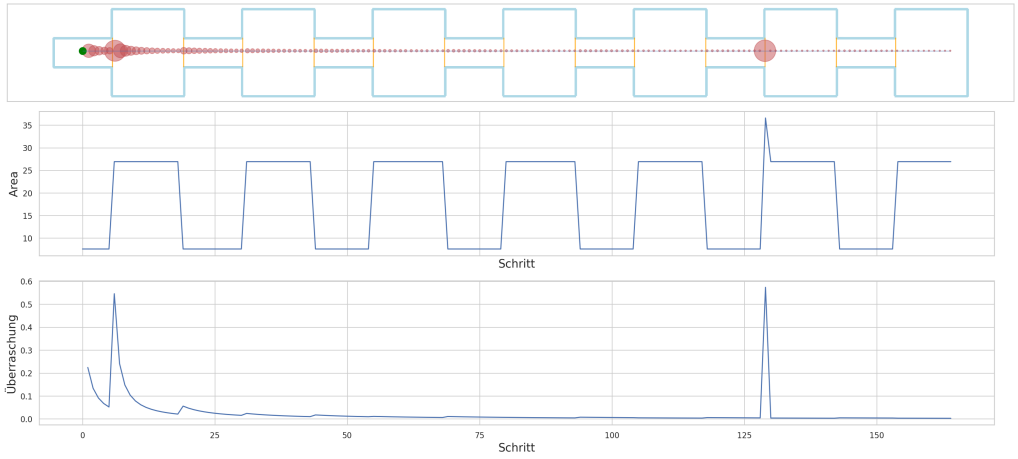
Eine grundlegende Qualität von Bayesian Surprise ist die Adaptierung des Modells an die Struktur der Umgebung. Demzufolge sollten häufig eintretende Beobachtungen zu zunehmend geringerer Überraschung führen. Der verwendete Ansatz zur Wahrnehmungsmodellierung mittels Bayesian Surprise sollte somit in der Lage sein, einerseits neuartige Ereignisse zu erkennen, sich aber gleichzeitig auch schnell auf gleichartige Strukturen, ebenso wie gleichartige Strukturwechsel einzustellen. Die für diese Untersuchung eingesetzte Umgebung ist **AlternatingDoors**. Abbildung 3.12a Oben zeigt die Umgebung im Karten-Raum. Der Agent beginnt links und bewegt sich horizontal nach rechts entlang der dargestellten Trajektorie. Die Größe der in rot dargestellten Kreise repräsentiert die Stärke der Überraschung an der entsprechenden Position des Agenten.
Wir beginnen mit der Betrachtung des vermutlich am einfachsten zu interpretierenden Features: *Area*. Da die Umgebung wie in Unterabschnitt 3.8.1 erläutert Türen enthält, ist in jedem Raum nur ein einzelner, statischer Wert für die Fläche zu beobachten: Entweder ein geringer Wert in den Verbindungsräumen oder ein hoher Wert in den größeren Räumen. Abbildung 3.12a Mitte zeigt die ermittelten Werte des Features *Area* (Y-Achse) entlang der Trajektorie (X-Achse). Der beobachtete Ausreißer im vorletzten Raum wird im Folgenden noch detailliert diskutiert. Zu Beginn startet der Agent in einem kleinen Verbindungsraum auf der linken Seite der Umgebung. Die durch die erste Beobachtung erzeugte Überraschung (Abbildung 3.12a Unten) fällt moderat aus. Dieses Phänomen trat in allen weiteren untersuchten Umgebungen gleichermaßen auf und ist der gewählten Initialisierung des Modell Priors als Gleichverteilung geschuldet. Der Agent bewegt sich nun horizontal nach rechts durch den Verbindungsraum, während das beobachtete Feature *Area* konstant bleibt. Dies führt wie vermutet zu einer Abschwächung der Überraschung, da die Beobachtungen zunehmend erwartet werden. Mit Betreten des ersten großen Raums beobachtet der Agent erstmalig hohe Werte des Features *Area*. Das Auftreten dieser Werte besitzt im aktuellen Modell des Agenten zu diesem Zeitpunkt eine geringe Wahrscheinlichkeit. Dies erzeugt eine starke Überraschung, da das basierend auf diesen neuen Daten aktualisierte Modell stark vom bisherigen abweicht. Mit fortschreitendem Durchqueren des großen Raumes adaptiert sich der Agent erneut an die aktuelle Raumwahrnehmung (bezüglich des Features *Area*), was sich in zunehmend reduzierter Überraschung widerspiegelt. Das Eintreten in den zweiten Verbindungsraum erzeugt einen erneuten, kurzzeitigen Anstieg der Überraschung. Mit weiterer Wiederholung dieser Raumabfolge schwächt sich dieser Effekt zunehmend ab. Der Agent hat sich an die Strukturen und Strukturwechsel der Umgebung ad-

aptiert. Zu Beginn des vorletzten großen Raumes enthalten die Daten einen Ausreißer (siehe Abbildung 3.12a Mitte). Hier ist erkennbar, dass der gemessene Wert ein singulär auftretendes, globales Maximum darstellt, der zuvor nicht aufgetreten ist. Diese Messung basiert auf einem Annäherungsfehler der Simulationsumgebung. Wie in Unterabschnitt 3.8.1 dargestellt, wird das tatsächliche Isovistenfeature *Area* durch 360 Strahlen angenähert. Dies erfolgt durch ein Verbinden der Strahlenendpunkte und der Berechnung der Fläche des resultierenden Polygons. Dadurch kann es, abhängig von der Position des Agenten, zu Annäherungs- und Rundungsfehlern kommen. Die Tatsache, dass die hier beobachtete Amplitude nicht regelmäßig auftritt, ist der Schrittlänge des Agenten geschuldet. Dadurch werden Messungen an unterschiedlichen Positionen innerhalb von strukturell gleichen Räume vorgenommen.

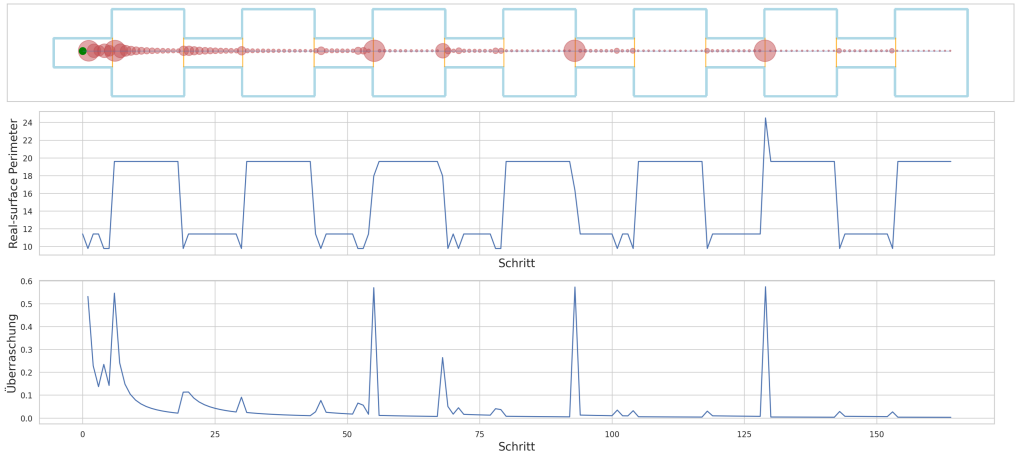
Die Werte der beobachteten Features *Real-surface Perimeter* (Abbildung 3.12b) und *Circularity* (Abbildung 3.12c) verhalten sich ähnlich zum Feature *Area*. Auch hier ist eine klare, alternierende Struktur mit hohen und niedrigen Werten zu erkennen. Aufgrund der strahlenbasierten Annäherung sind die Werte des *Real-surface Perimeters* im Vergleich jedoch weniger binär, und es treten ebenfalls Ausreißer in den Messungen auf. Dies führt in Kombination mit der verwendeten Diskretisierung in Intervalle zu wiederholten Ausreißern der ermittelten Überraschung. In Abschnitt 3.9 werden mögliche Ansätze zur Problembehebung diskutiert. Trotz dieser unerwünschten Phänomene ist der grundlegende Trend einer sich abschwächenden Überraschung klar erkennbar. In anderen Worten: Eine Adaptierung an gleichartige Strukturen ist festzustellen.

Die beobachteten Messwerte der Features *Variance* (Abbildung 3.13a Mitte) und *Skewness* (Abbildung 3.13b Mitte) verhalten sich grundsätzlich ähnlich. Die genauen Werte sind nun jedoch stark von der Position des Agenten innerhalb des Raumes abhängig. Nahe den Wänden der Räume sind hohe Werte zu beobachten, während in der Mitte tendenziell geringere Werte auftreten. Im Falle der *Skewness* ist ein zusätzliches lokales Maximum in der Mitte der großen Räume zu erkennen. Aufgrund der (bewusst) nicht erfolgten Synchronisation zwischen Schrittlänge und Raumgrößen sind die beobachteten Werte in unterschiedlichen Räumen, wie zuvor auch, nicht identisch. Dies spiegelt sich in teils leicht erhöhter Überraschung in Abbildung 3.13a und Abbildung 3.13b Unten wider. Dennoch ist auch hier in beiden Evaluationen ein Trend sich abschwächender Überraschung erkennbar, der auf eine Adaptierung an gleichartige Strukturen zurückzuführen ist.

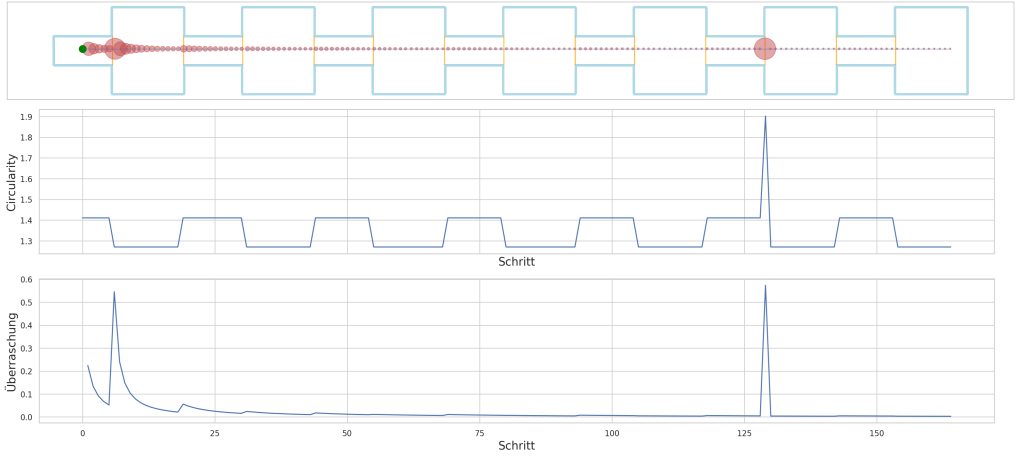
Das Feature *Occlusion* kann nicht in der bisher betrachteten Umgebung **AlternatingDoors** untersucht werden, da das Vorhandensein von Türen dazu führt, dass keine verdeckten Flächen auftreten. Deshalb wurde zur Betrachtung dieses Features das Verhalten auf der Umgebung **Alternating** untersucht, die keine Türen enthält (Abbildung 3.13c). Die ermittelten Feature-Werte (mittlere Abbildung) alternieren zwischen hohen und mittleren Werten. Hohe Werte treten auf, wenn ein Großteil des Sichtbereichs verdeckt ist. Dies ist der Fall,



(a) Area



(b) Real-surface Perimeter



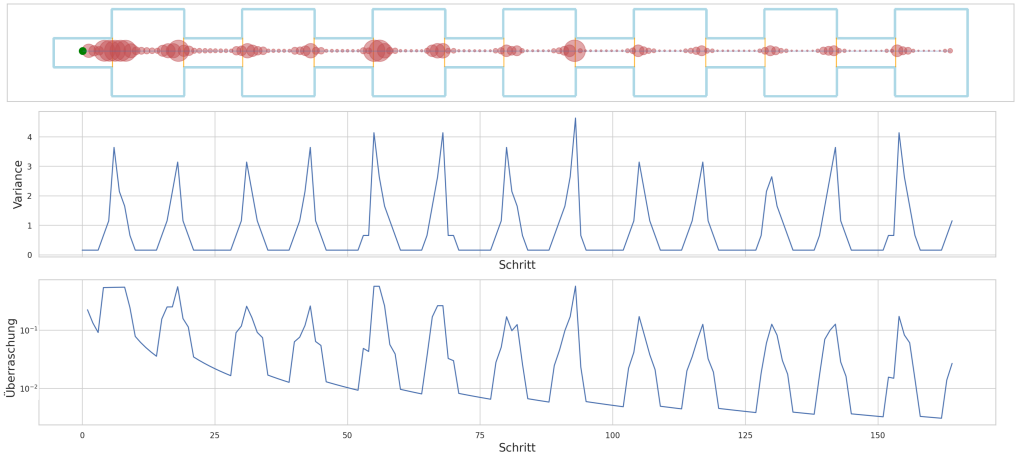
(c) Circularity

Abbildung 3.12: Visualisierung der Überraschungswerte je Feature *Area*, *Real-surface Perimeter* und *Circularity* auf der Umgebung **AlternatingDoors**. Die jeweils oberste Zeile visualisiert die Überraschung (rote Kreise) im Karten-Raum entlang der Trajektorie (Startpunkt: grüner Kreis, Bewegung nach rechts). Mitte: Messwerte des Isovistenfeatures entlang der Trajektorie. Unten: Berechneter Überraschungswert in linearer Skala.

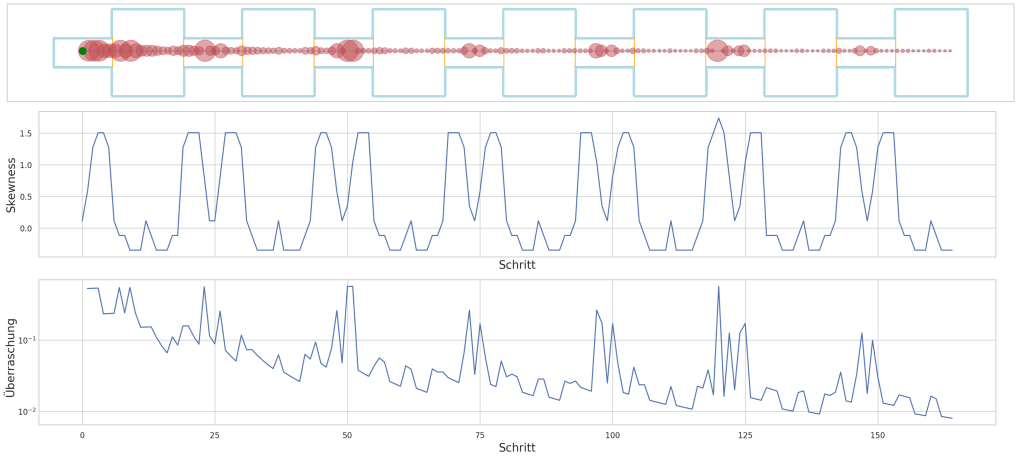
wenn sich der Agent in den Durchgangsräumen befindet und in große Räume blickt. Im umgekehrten Fall, wenn der Blick von einem großen Raum aus in die Durchgangsräume fällt, ist die Verdeckung im Vergleich geringer. Verursacht durch die nicht von Türen beschränkten Raumübergänge verlaufen die Änderungen der Feature-Werte, im Vergleich zu den vorherigen, gradueller. Dies führt zu stark schwankenden Werten der Bayesschen Überraschung. Besonders zu Beginn der Trajektorie tritt eine Vielzahl unerwarteter Werte auf, die zunächst zu einer konstant hohen Überraschung führen. Die Stärke der Ausschläge nimmt dennoch über den Verlauf zunehmend ab, so dass auch hier erkennbar ist, dass sich das Modell an die auftretenden Strukturen adaptiert. Wie auch zuvor enthalten die Messungen einen Ausreißer (globales Maximum der Feature-Werte im vorletzten Durchgangsraum), der zu einer kurzen, starken Überraschung führt. Die Entwicklung der Werte im letzten Raum weicht von den vorherigen Räumen ab. Es treten gegen Ende Werte auf, die zwischen den bisherigen Werten von Durchgangsräumen und großen Räumen liegen. Dies ist darauf zurückzuführen, dass der letzte Raum nur einen vorhergehenden Durchgangsraum, aber keinen folgenden besitzt. Auch diese Abweichung der Struktur wird durch die Wahrnehmungsmodellierung erkannt und führt zu erhöhter Überraschung.
In Abschnitt 3.7 wurde die Möglichkeit zur Erstellung einer kombinierte Maßzahl der Überraschung genannt. Diese wurde in der folgenden Untersuchung als ungewichtete Summe der Einzelüberraschungen erzeugt. Abbildung 3.14 zeigt die so ermittelten Werte auf der Umgebung **BasicSimple**, in der die einzelnen Räume nicht durch Türen abgetrennt sind. Auch in dieser kombinierten Variante zeigt sich das zuvor beobachtete Verhalten: Zu Beginn werden neu wahrgenommene Strukturen korrekt als solche erkannt und schlagen sich in hoher Überraschung nieder. Über den Verlauf der Trajektorie adaptiert sich die Modellierung an die auftretenden Strukturen, was zu einer Reduktion der Überraschung führt. Nur gegen Ende weicht der ermittelte kombinierte Featurewert erneut von den vorherigen ab, was zu einer erhöhten Überraschung führt.

3.8.3 Beibehalten von Konzepten nach kurzer Überraschung

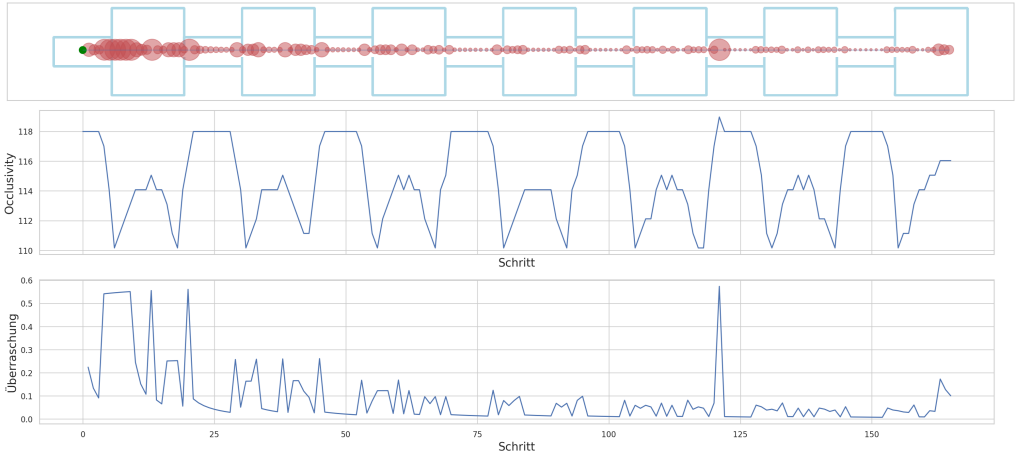
Im vorherigen Abschnitt konnte gezeigt werden, dass die Wahrnehmungsmodellierung strukturelle Veränderungen erkennt, und sich an diese adaptieren kann. Im Folgenden wird nun die Hypothese untersucht, dass eine plötzliche, starke Überraschung gelernte Konzepte nicht verwirft. Die Untersuchung beginnt mit einer Betrachtung des Features *Area* auf der Umgebung **AlternatingSurpriseDoors**. Die Verwendung einer Umgebung mit Türen führt dazu, dass die Feature-Werte einheitlicher sind und der Eintritt in abweichende Strukturen klar abgetrennt ist. Darauf folgt eine Betrachtung des Features *Occlusion* auf der Umgebung **AlternatingSurprise**, die keine Türen enthält.



(a) Variance



(b) Skewness



(c) Occlusion

Abbildung 3.13: Visualisierung der Überraschungswerte je Feature *Variance*, *Skewness* und *Occlusion*. Die jeweils oberste Zeile visualisiert die Überraschung (rote Kreise) im Karten-Raum entlang der Trajektorie (Startpunkt: grüner Kreis, Bewegung nach rechts). Für eine bessere Lesbarkeit sind die Überraschungswerte von *Variance* und *Skewness* in logarithmischer Skala abgebildet, während *Occlusion* in linearer Skala dargestellt ist.

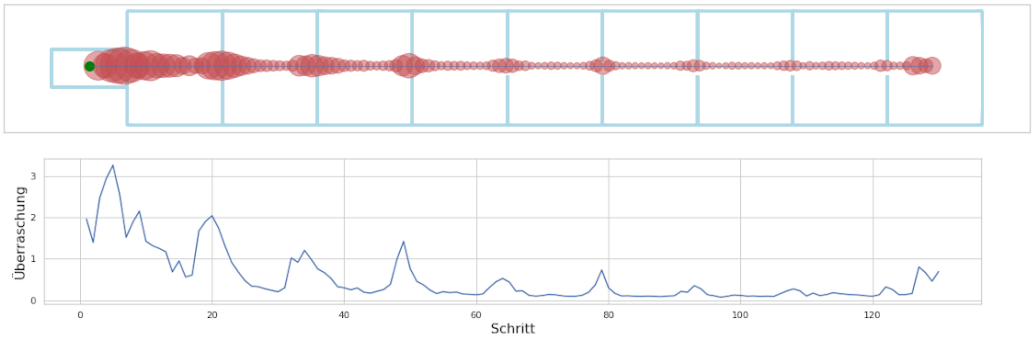
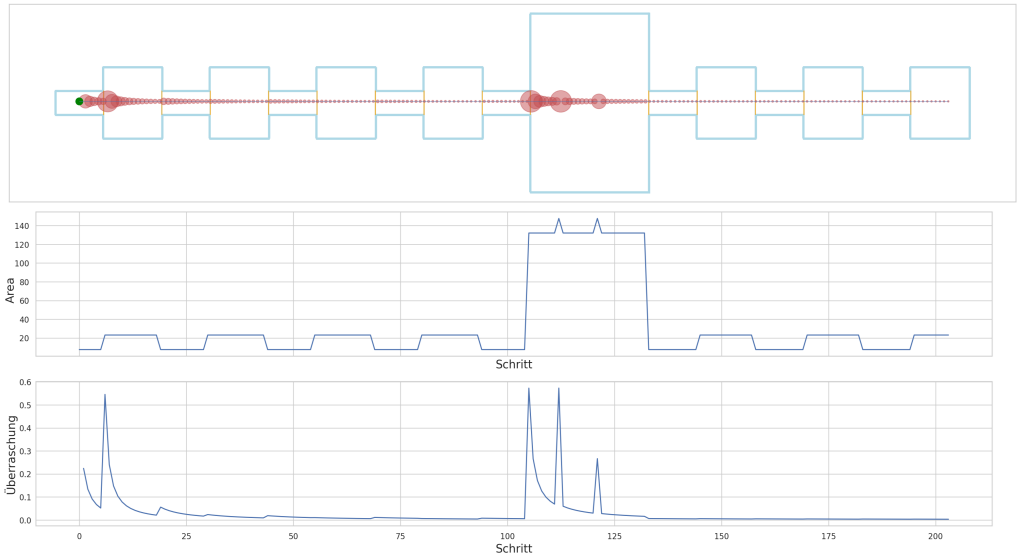


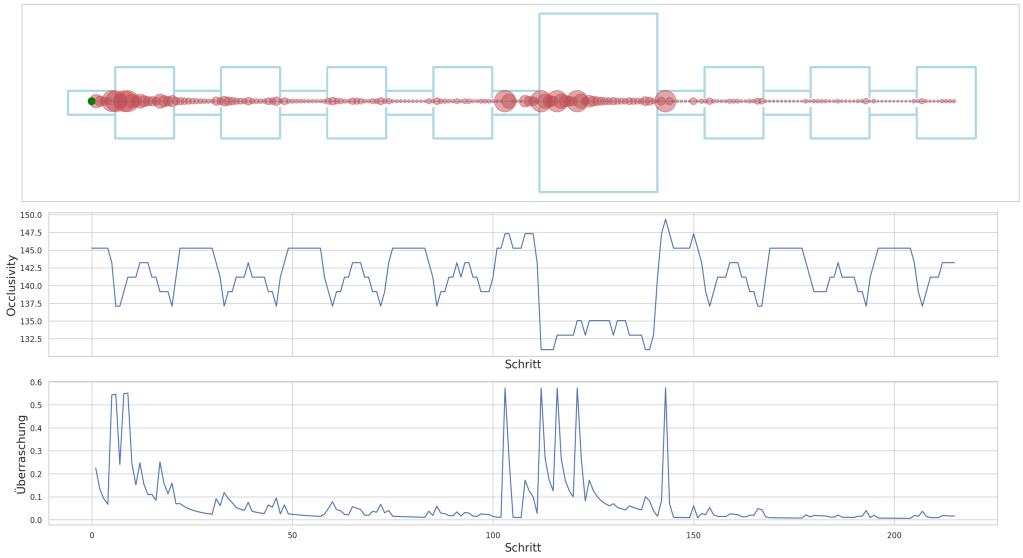
Abbildung 3.14: Visualisierung der additiv kombinierten Überraschungswerte auf der Umgebung **BasicSimple**. Oben: Visualisierung der Überraschung (rote Kreise) im Karten-Raum entlang der Trajektorie (Startpunkt: Grüner Kreis, Bewegung nach Rechts). Eine Adaptation auf die sich wiederholenden Strukturen ist deutlich sichtbar.

Ziel ist hier die Untersuchung der Übertragbarkeit auf Umgebungen mit sich kontinuierlich veränderndem Wahrnehmungseindruck. Abbildung 3.15a und Abbildung 3.15b Oben zeigen das jeweils grundlegende Layout der Umgebungen. Grundelement ist eine sich wiederholende Abfolge von kleinen und mittelgroßen Räumen, vergleichbar mit den vorherigen Umgebungen, die dann jedoch durch das plötzliche Auftreten einer stark abweichenden Struktur in Form eines sehr großen Raums unterbrochen wird. Anschließend folgt erneut die sich zu Beginn wiederholende Abfolge von Räumen.

Abbildung 3.15a Mitte zeigt, dass klar getrennte Werte für das Feature *Area* auf der Umgebung **AlternatingSurpriseDoors** gemessen wurden. Niedrige und mittlere Werte wechseln sich alternierend ab. Der in der Mitte der Umgebung plötzlich auftretende große Raum führt wie zu erwarten zu einem sofortigen Anstieg der Messwerte. Mittig in diesem Raum treten erneut zwei durch Annäherungsfehler verursachte Ausreißer auf. Nach Verlassen dieses Raums entsprechen die Feature-Werte erneut den zu Beginn ermittelten. Die Struktur der Umgebung spiegelt sich klar in den ermittelten Überraschungswerten (Abbildung 3.15a Unten). Zu Beginn, und bei erstmaligem Eintreten des Agenten in einen mittelgroßen Raum, zeigen sich hohe Überraschungswerte. Diese fallen dann wie erwartet durch Adaptation auf ein sehr geringes Level ab. Bei Betreten des großen Raums tritt eine starke Abweichung zwischen Erwartung und Beobachtung der Wahrnehmungsmodellierung auf. Das Einbeziehen dieser abweichenden Beobachtung in die Modellierung führt zu einer starken Veränderung des Posteriors des Modells, was sich wiederum in einem stark erhöhten Überraschungswert ausdrückt. Entlang der Durchquerung dieses Raums – unterbrochen nur durch die Annäherungsfehler – führt die laufende Einbeziehung dieser Beobachtungen zu immer kleineren Veränderungen des Modells, so dass sich auch die Überraschung abschwächt. Der entscheidende Punkt der Unter-



(a) Area



(b) Occlusion

Abbildung 3.15: Visualisierung der Überraschungswerte des Features *Area* auf der Umgebung **AlternatingSurpriseDoors** und *Occlusion* auf **AlternatingSurprise**.

suchung ist nun das Verhalten der Wahrnehmungsmodellierung bei Verlassen des großen Raums und erneutem Eintreten in einen kleinen Raum. Abbildung 3.15a Unten zeigt an dieser Stelle der Trajektorie einen leichten Abfall des Überraschungswertes. Dies zeigt, dass die Wahrnehmungsmodellierung die vor dem überraschend aufgetretenen großen Raum häufig aufgetretene Struktur des kleinen Raums wiedererkennt. Die nun erneut dem Beginn der Trajektorie entsprechende wiederholende Abfolge von Räumen erzeugt nur mehr das gleiche geringe Level an Überraschung wie zuvor. Somit zeigt sich, dass die Wahrnehmungsmodellierung gelernte Konzepte nicht verwirft, wenn plötzlich eine kurze Überraschung durch andersartige Wahrnehmung auftritt.
Es folgt nun die Betrachtung des Features *Occlusion* auf der strukturell vergleichbaren Umgebung **AlternatingSurprise**. Diese enthält jedoch keine Türen, so dass die auftretenden Verdeckungen durch das Feature *Occlusion* erfasst werden. Abbildung 3.15b Mitte zeigt, dass das Feature stabile Werte innerhalb der kleinen Räume aufweist, von denen aus gesehen der Großteil der umgebenden Räume verdeckt ist. An den Randbereichen der mittelgroßen Räume sind die Werte geringer, steigen jedoch zur Mitte der Räume hin an. Die *Occlusion*-Werte verändern sich dann jedoch bereits im letzten kleinen Raum vor dem überraschend großen Raum. Da hier mehr sichtbare Fläche verdeckt ist, steigen die Werte an, was sich wiederum in erhöhter Überraschung niederschlägt. Das Eintreten in den großen Raum führt zu einem starken Absinken der Feature-Werte und hoher Überraschung. Der folgende Übergang in den kleinen Raum ist erneut überraschend, was auf das Auftreten eines globalen Maximums der *Occlusion*-Werte zurückzuführen ist. Über den Rest der Trajektorie ist die Überraschung erneut minimal, da die Abfolge der mittleren und kleinen Räume der ersten Hälfte der Trajektorie entspricht. In dieser Tatsache zeigt sich erneut, dass plötzliche große Überraschung die bereits gelernten Strukturen der Wahrnehmung nicht verwirft.

3.8.4 Übertragbarkeit auf realistische Umgebungen

Mit den vorherigen Resultaten konnte auf synthetischen Umgebungen gezeigt werden, dass die entwickelte Wahrnehmungsmodellierung in der Lage ist sich an wiederkehrende Strukturen zu adaptieren. Zudem werden diese Konzepte auch bei kurzzeitig eintretender Überraschung durch abweichende Strukturen nicht verworfen. Als nächstes erfolgt nun eine Betrachtung der Übertragbarkeit dieses Verhaltens auf realistische Umgebungen.
Die Untersuchung erfolgt auf der Umgebung **TUM**, die auf dem bereits zuvor verwendeten Gebäudegrundriss der Technischen Universität München (TUM) basiert. Abbildung 3.16 zeigt blau eingefärbt die Begrenzungen dieser komplexen Umgebung im Karten-Raum. Der verwendete Ausschnitt beinhaltet die Haupthalle der Universität sowie vier umgebende Straßen. Die Umgebung ist geprägt von einem zentral gelegenen, großen Innenhof sowie Gebäuden mit stark unterschiedlich geformten Räumen und einer Vielzahl an Eingängen. Der

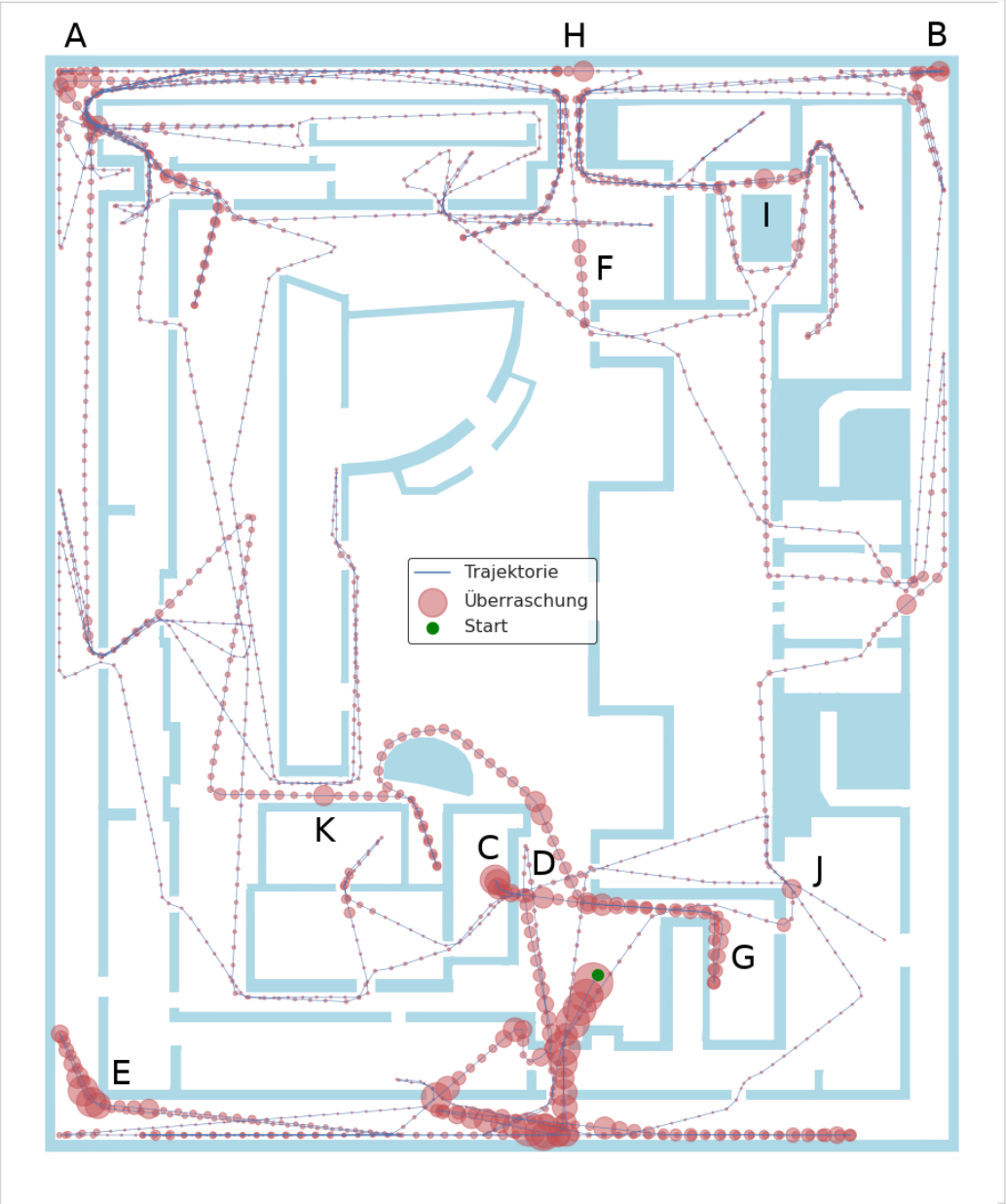


Abbildung 3.16: Visualisierung einer sehr langen Trajektorie durch die Umgebung **TUM**. Die additiv kombinierten Überraschungswerte aller sechs Isovistenfeatures sind entsprechend ihrer Stärke durch Kreise varrierender Größe in Rot abgebildet. Buchstaben markieren Regionen hoher Überraschung.

Startpunkt der vom Agenten beschrittenen Trajektorie durch diese Umgebung ist in Grün markiert. Die entlang der Trajektorie berechneten Überraschungswerte sind entsprechend ihrer Stärke durch Kreise unterschiedlicher Größe in Rot abgebildet. Hierbei wurde die additiv kombinierte Überraschung aller sechs Isovistenfeatures verwendet. Im Folgenden werden einige ausgewählte Bereiche der Trajektorie detailliert betrachtet.

Betrachtet man das Isovistenfeature *Real-surface Perimeter* isoliert von den Restlichen, so befinden sich die höchsten Werte der Überraschung an den in Abbildung 3.16 markierten Punkten **A**, **B** und **C**. Die Punkte **A** und **B** befinden sich exakt an Positionen, von denen aus der Agent sowohl die horizontal, als auch die vertikal verlaufende Straße überblicken kann. Die hier gemessenen Werte sind außergewöhnlich und führen folglich zu hoher berechneter Überraschung. An Punkt **C** tritt ebenfalls hohe Überraschung auf. Die Ursache hierfür ist jedoch genau umgekehrt: Im Bezug auf das aktuelle Modell des Agenten war der an Punkt **C** stark eingeschränkte Sichtbereich sehr überraschend.

Die Punkte **D** und **E** wurden ebenfalls aufgrund ihrer sehr hohen Überraschungswerte ausgewählt. Diese ist hier jedoch besonders durch das Feature *Occlusion* verursacht, also verursacht durch einen hohen Grad an Verdeckung. Der Agent ist hier, aus einer Reihe von kleineren Räumen kommend, kurz davor den großen Innenhof zu betreten. Der Sichtbereich öffnet sich, gleichzeitig ist aber noch ein Großteil der Freifläche des Innenhofes durch die Gebäudeecke zur Linken von Punkt **D** verdeckt. Charakteristisch für diese Stelle ist somit die Tatsache, dass der Agent hier aus einer Reihe von kleineren Räumen kommend kurz davor ist, den großen Innenhof zu betreten.

Der Punkt **E** markiert eine Position, an der der Agent die unterhalb des Gebäudes horizontal verlaufende Straße von rechts nach links durchschreitet. An der Stelle, an der der Agent um die Ecke biegt, wird das Sichtfeld stark durch die Gebäudeecke beschränkt. Die so durch die Verdeckung verursachte Überraschung hält für einige Schritte an, fällt dann jedoch sofort ab, als der Agent in die horizontal verlaufende Straße zurückkehrt.

An den markierten Punkten **F** und **G** sorgt das Feature *Area* für hohe Überraschung. Besonders an Punkt **F** ist die Bewegung des Agenten gut zu erkennen. Dieser betritt den Innenhof aus dem Gebäude unterhalb von **F** und nimmt einen großen Sichtbereich wahr. Dieser bleibt nicht statisch, sondern wird entlang der Bewegung des Agenten nach oben zunehmend größer, da sich der Sichtbereich nach links in die Umgebung öffnet. Dies führt zu den erhöhten Überraschungswerten entlang dieses Abschnitts der Trajektorie.

Auch an Position **G** ist die Bewegung zu erkennen. Der Agent bewegt sich von unten nach oben, wobei der Eingang zur Rechten den Sichtbereich erweitert. Dieser öffnet sich bei Verlassen des Durchgangs zur Linken noch weiter.

Das Feature *Variance* führte an Position **H** zu ansteigender Überraschung. An dieser Stelle kann der Agent weit nach links und rechts sehen, gleichzeitig aber auch in den unterhalb liegenden Innenhof. Eine vergleichbar hohe Varianz des Sichtbereichs ist dem Agent zuvor nicht begegnet, so dass die Überraschung

entsprechend groß ausfällt.
Überraschung die durch die Features *Skewness* verursacht wurde ist an den Positionen **I** und **J** markiert. Im kleinen Durchgang über Position **I** tritt ein schiefer Sichtbereich auf, der für das zu diesem Zeitpunkt aktuelle Wahrnehmungsmodell des Agenten neu war. Die Sicht ist an dieser Stelle in den meisten Regionen stark beschränkt, und nur an einer Stelle – der offenen Türe zur Linken – in weite Distanz möglich. An Position **J** ist ein vergleichbarer Effekt feststellbar. Der Agent befindet sich nahe an der linken Wand, was die Sicht stark einschränkt. Gleichzeitig ist der Blick in einige wenige Richtungen in weite Distanz möglich, zum Beispiel durch die offene Türe in Richtung von Position **D**.
Abschließend markiert Position **K** eine Stelle der Umgebung, an der das Feature *Circularity* zu einem Anstieg der Überraschung führte. Basierend auf dem zu diesem Zeitpunkt aktuellen Wahrnehmungsmodell des Agenten ist der Sichtbereich außergewöhnlich. Dieser ist nach oben und unten durch den schmalen Korridor stark beschränkt, während der Blick nach links und rechts in weite Distanz möglich ist.
Zusammenfassend lässt sich feststellen, dass die berechneten Überraschungswerte interessante Regionen der Umgebung klar kennzeichnen. Durch die verwendete Wahrnehmungsmodellierung konnten Straßenecken, Aussichten über den Innenhof, das Verlassen von Gebäuden in Gebiete mit weiter Aussicht, aber auch Gebiete mit stark beschränktem Sichtbereich ausfindig gemacht werden. Interessant sind zudem Gebiete, in denen die Überraschung entlang der Bewegung des Agenten zunehmend ansteigt.

3.8.5 Charakterisierung einer Trajektorie

Nachdem im vorherigen Abschnitt die Übertragbarkeit des Modellierungsansatzes auf realistische Umgebungen gezeigt werden konnte, wird nun die Möglichkeit zum Einsatz in Form einer Anwendung demonstriert. Zu diesem Zweck wurde eine manuell definierte Trajektorie in einem Bereich der **LMU** Umgebung verwendet. Entlang dieser Trajektorie erfolgte die Wahrnehmungsmodellierung und Berechnung der Überraschungswerte. Abbildung 3.17a zeigt in Rot die Messpunkte entlang dieser Trajektorie im Karten-Raum. Der Agent startet in einem Raum im linken Teil der Umgebung und bewegt sich gegen den Uhrzeigersinn entlang der Trajektorie. Abbildung 3.17b visualisiert die berechneten Überraschungswerte erneut im Karten-Raum durch rote Kreise, deren Größe mit der Stärke der Überraschung korreliert. Um die Interpretierbarkeit zu vereinfachen, beschränkt sich die Betrachtung in diesem Abschnitt auf das Feature *Area*. Abbildung 3.17c bildet die Messwerte und entsprechenden Überraschungswerte im zeitlichen Verlauf der Trajektorie entlang der X-Achse ab. Lokale Maxima der Überraschung sind in beiden Abbildungsvarianten durch die Buchstaben **A-I** markiert.
Zu Beginn ist aus Abbildung 3.17c klar erkennbar, dass der Agent in einem

kleinen Raum beginnt und von der initialen Wahrnehmung überrascht ist, sich aber schnell an diese Wahrnehmung adaptiert. Die Überraschung fällt folglich, begründet durch die Konstanz des Eindrucks, ab. Der erste erneute Anstieg der Überraschung befindet sich an Position **A**, als der Agent den Raum verlassen hat und den schmalen vertikalen Gang betritt. Der Sichtbereich ist hier im Vergleich erweitert. An Position **B** tritt der nächste Ansprung der Überraschung auf. An dieser Position befinden sich offene Türen exakt parallel zueinander. Diese Konstellation ist für den Agenten ebenfalls neu und führt folglich zu hoher Überraschung. Direkt unterhalb von Position **B** tritt eine vergleichbare Situation erneut auf, führt nun aber aufgrund von Adaptierung zu geringerer Überraschung. Ein extrem starker Anstieg der Überraschung ist an der folgenden Position **C** erkennbar. Hier betritt der Agent die große Halle im unteren Bereich der Umgebung. Da die Messwerte des Features *Area* über die nächsten Schritte konstant hoch bleiben, fällt die Überraschung entsprechend wieder ab. An Position **D** befindet sich der Agent zwar noch innerhalb der Halle, der Sichtbereich ist jedoch durch das Hindernis zur Linken beschränkt. Dies spiegelt sich im Abfall der Messwerte und einem Anstieg der Überraschung wider. Nach Verlassen der Halle betritt der Agent an Position **E** einen schmalen Korridor und betritt danach einen kleineren Raum. Aufgrund der verwendeten Diskretisierung der Werte in Intervalle, sind die wahrgenommenen Werte konstant, so dass sich keine starken Veränderungen des Wahrnehmungsmodells ergeben. Die nächste überraschende Situation tritt an Position **F** auf, als der Agent den langen vertikalen Korridor betritt. Das globale Maximum, sowohl in den Feature-Werten, als auch in der ermittelten Überraschung befindet sich an Position **G**. Hier betritt der Agent einen sehr langen horizontalen Korridor (Abbildung 3.17b zeigt nur einen Ausschnitt, die Umgebung enthält rechts noch weitere nicht sichtbare Bereiche. Eine Abbildung der gesamten Umgebung befindet sich im Anhang.). Folglich ist der Sichtbereich des Agenten sehr groß. Ebenso kann der Agent an Position **H** in die darüberliegenden Räume blicken. Der letzte Anstieg der Überraschung ist an Position **I** zu beobachten, an einer Stelle an der der Sichtbereich durch den Korridor wieder stärker eingeschränkt ist.

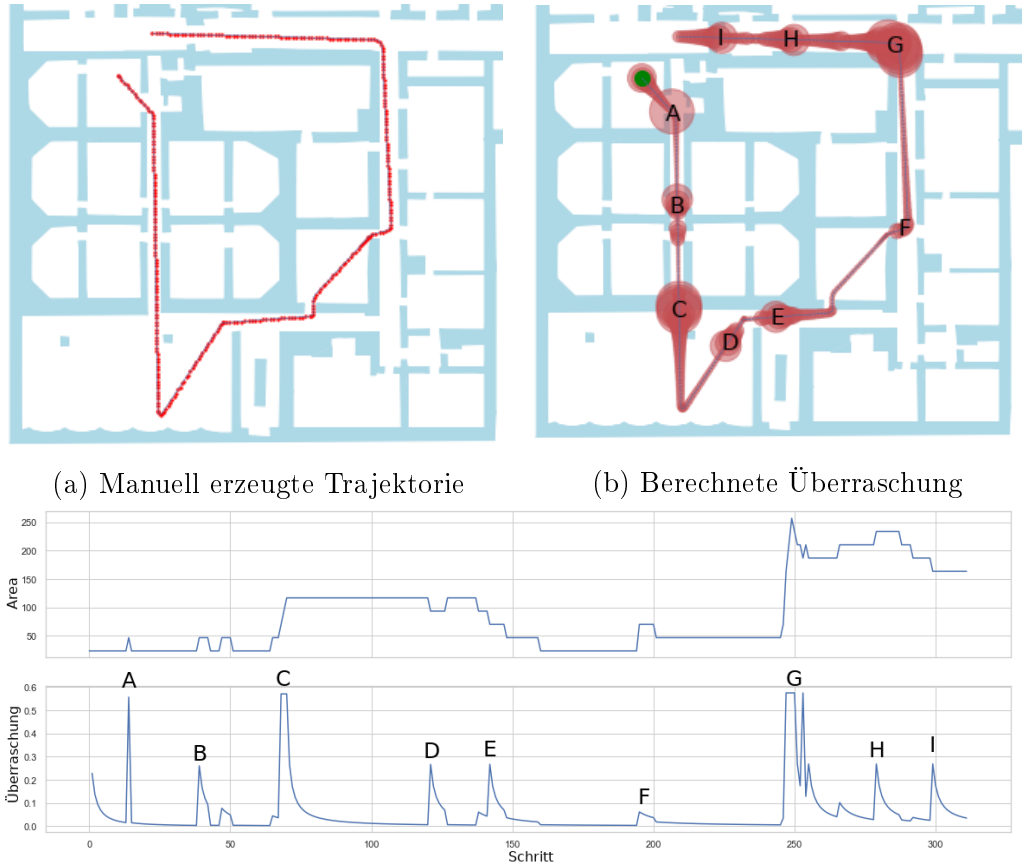
Das Ziel einer zu entwickelten Anwendung könnte es nun sein, die vom Agenten beschrittene Route zusammenzufassen oder zu charakterisieren. Ein möglicher Weg wäre es, an Stellen hoher Überraschung eine Art von „Screenshots“ des Sichtbereichs des Agenten zu erzeugen. Abbildung 3.18 zeigt solche „Screenshots“ die beispielhaft durch quadratisches Beschneiden der Umgebungsvisualisierung im Karten-Raum erzeugt wurden. Diese 9 Ausschnitte wurden genau an den Positionen **A-I** erzeugt, an denen zuvor lokale Maxima der Überraschungswerte ermittelt wurden. Die Charakteristik der Route ist klar erkennbar: Der Agent betritt den vertikalen Korridor (**A**), passiert die parallelen Türen (**B**) und betritt die Halle (**C**). Der Agent verlässt die Halle im oberen rechten Bereich (**D**) und durchquert einen horizontalen Durchgang (**E**). Der schmale vertikale Korridor wird schließlich durch eine rechtsliegende Türe be-

treten und nach oben durchquert (**F**). Anschließend biegt der Agent nach links in den langen horizontalen Korridor (**G**), passiert eine weitere rechtsliegende Türe und bewegt sich weiter nach links. Die Route endet mit der Durchquerung eines schmäleren Durchgangs innerhalb des Korridors (**I**).
Abschließend lässt sich somit feststellen, dass die mit diesem Verfahren erzeugte Abfolge von „Screenshots“ an Positionen hoher Überraschung eine klare Charakterisierung der beschrittenen Trajektorie bietet.

3.9 Zusammenfassung

Ziel dieses Kapitels war die Entwicklung von Verfahren, die es lernenden Systemen ermöglichen, die Wahrnehmung von Raum numerisch zu erfassen und damit nutzbar zu machen. Grundlegende Voraussetzung zur Erreichung dieses Ziels war dabei, dass wiederkehrende Strukturen in der Wahrnehmung auch wiederkehrende Strukturen in der numerischen Repräsentation erzeugen müssen. Nur dann sind diese mittels statistischer Modelle erfassbar.
Zur Generierung der Datenbasis wurde das in [14] vorgestellte Konzept der Isovisten als geeignet identifiziert. Basierend darauf wurde ein Framework zur Erzeugung von Isovistenmessungen entlang geospatialer Trajektorien innerhalb einer 3D-Simulationsumgebung entwickelt. Die auf diesem Weg erfassten Daten konnten anschließend mittels Daten Preprocessing aufbereitet, und ihre zeitliche Dimension durch das SMA Verfahren nutzbar gemacht werden. Auf dieser Datengrundlage erfolgte die Wahrnehmungsmodellierung unter Verwendung von Unsupervised Machine Learning Methoden. Dieser Ansatz konnte anschließend auf simulierten Umgebungen, deren Struktur auf realen Gebäudeplänen basiert, erfolgreich evaluiert werden. Es wurde gezeigt, dass die erfassten Isovistenmessungen entlang der Trajektorien die wiederkehrenden Strukturen innerhalb der Gebäude widerspiegeln. Diese wiederkehrenden Muster konnten erfolgreich durch Unsupervised Machine Learning Methoden modelliert werden. Eine qualitative visuelle Analyse sowohl im Karten- als auch im Feature-Raum bestätigte die erfolgreiche Modellierung von semantisch sinnvollen Clustern.
Als zweiter Schritt wurde dieses entwickelte Verfahren auf Basis von Isovistenfeatures mittels Bayesian Surprise erweitert. Bayesian Surprise modelliert die subjektive Erwartung eines Agenten bezüglich des Auftretens von Ereignissen durch bedingte Wahrscheinlichkeitsverteilungen derselben. Diese Verteilungen und damit die Erwartungen des Agenten können laufend durch Einbeziehen neuer Wahrnehmungen aktualisiert werden. Ein möglicher Anwendungsfall ist die Identifikation von charakteristischen Stellen in Gebäuden. Da diese Stellen menschliche Aufmerksamkeit auf sich ziehen, wären sie optimal zur Positionierung von Kartenmaterial oder anderen Informationen von LBS-Systemen.
Der Beitrag des entwickelten Verfahrens kann in zwei Elemente geteilt werden: Zunächst wurde ein Weg aufgezeigt, wie Isovistenmessungen entlang räumlicher Trajektorien zur Verwendung im Rahmen von Bayesian Surprise aufbe-

reitet werden können. Diese dienen als Eingabedaten für den zweiten Schritt, die Wahrnehmungsmodellierung mittels Bayesian Surprise. Hier konnte gezeigt werden, dass der Einsatz der Dirichlet Verteilung als Conjugate Prior zur Multinomialverteilung eine effiziente und dennoch exakte Bayessche Posteriorberechnung bei Verwendung von Isovistendaten ermöglicht.
Im Rahmen der durchgeführten Evaluation konnten die folgenden Kernelemente gezeigt werden: (1) Die entwickelte Wahrnehmungsmodellierung reagiert auf unerwartete Ereignisse entlang der Trajektorie, adaptiert sich gleichzeitig aber auf wiederholende Strukturen. (2) Das Auftreten plötzlicher starker Überraschung, verursacht durch abweichende Wahrnehmungen, verwirft gelernte Konzepte der Wahrnehmungsmodellierung nicht. (3) Der vorgestellte Ansatz zur Wahrnehmungsmodellierung ist übertragbar auf realistische Umgebungen. (4) Eine Trajektorie kann durch „Screenshots“ von Positionen, an denen hohe Überraschung auftritt, zusammengefasst beschrieben und charakterisiert werden.
Die Evaluationsergebnisse haben jedoch auch Schwächen der verwendeten Modellierung aufgezeigt. Verursacht durch die Diskretisierung der Isovistenwerte, in Kombination mit der strahlenbasierten Isovistenannäherung, traten vereinzelt Ausreißer in den Messwerten auf. Diese verursachten Ausreißer in der berechneten Überraschung. Als möglicher Lösungsansatz wäre es von Interesse, in zukünftigen Arbeiten die Auswirkung der Verwendung von kontinuierlichen Verteilungen anstelle von Diskreten zu evaluieren. Hier könnte der Einsatz von Gaussian Mixture Models [16] ein vielversprechender Weg sein, der in der Lage ist, die Multimodalität der erfassten Isovistenwerte abzubilden. Dies würde zwar einerseits die Berechnungskomplexität deutlich erhöhen, da dann approximative Inferenzmethoden zur Schätzung des Bayesian Posteriors notwendig wären. Verschiedene approximative Bayessche Inferenzmethoden werden im folgenden Kapitel 4 genauer vorgestellt. Andererseits könnten so graduellere Übergänge modelliert werden, da die Diskretisierung der Messwerte in Intervalle entfallen würde. Dies könnte zur Generierung mächtigerer Wahrnehmungsmodelle führen, die sich mit der menschlichen Interpretation noch besser decken.



(a) Manuell erzeugte Trajektorie (b) Berechnete Überraschung

(c) Messwerte *Area* (Oben) und berechnete Überraschung (Unten).

Abbildung 3.17: Demonstration der Umsetzbarkeit des Konzepts, in Form einer Anwendung. (a) Detailausschnitt der Umgebung **LMU** mit einer manuell definierten Trajektorie (Rot). Diese beginnt in einem kleinen Raum auf der linken Seite der Umgebung und läuft gegen den Uhrzeigersinn. (b) Berechnete Überraschungswerte basierend auf dem Feature *Area* im Karten-Raum. Buchstaben markieren Regionen hoher Überraschung. (c) Oben: Feature-Werte *Area* entlang der Trajektorie Unten: Dazu korrespondierende, berechnete Überraschungswerte linearer Skala.

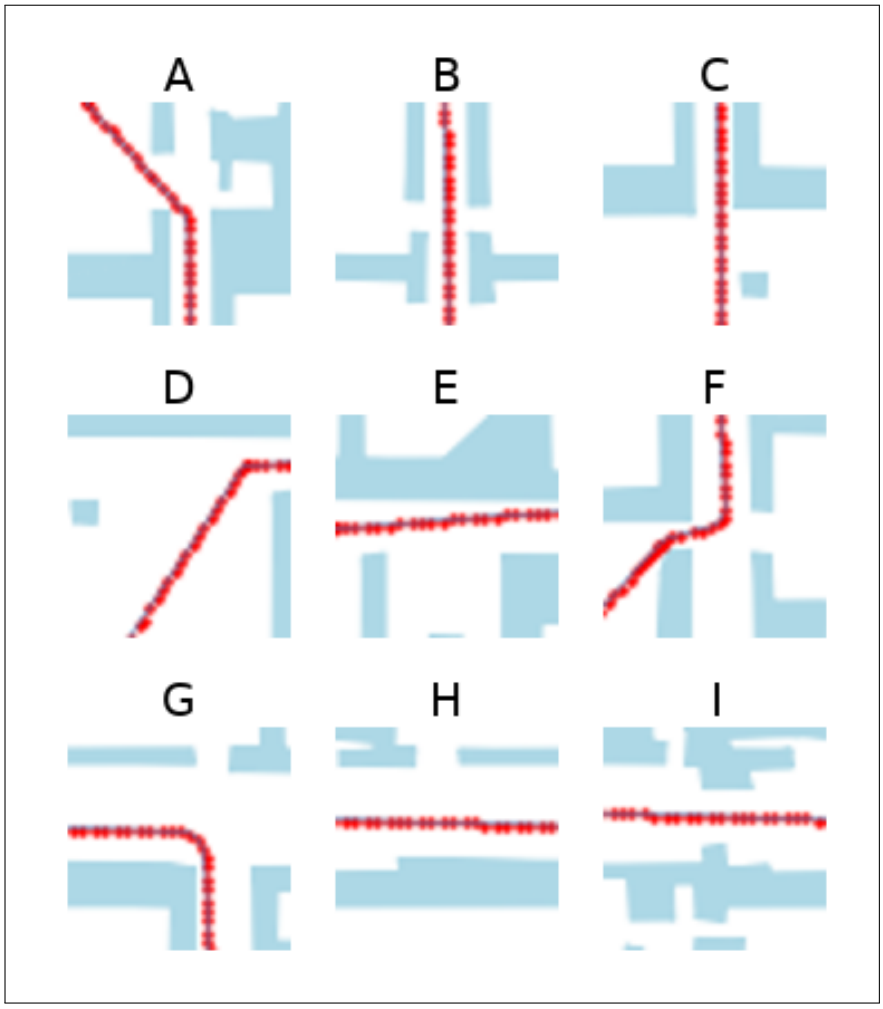


Abbildung 3.18: Ausschnitte („Screenshots“) des Sichtbereichs des Agenten an Positionen hoher Überraschung. Diese visuelle Zusammenfassung charakterisiert die durch den Agenten beschrittene Route.

4 Unsicherheitsmodellierung

Im vorherigen Kapitel konnte gezeigt werden, wie mit Hilfe verschiedener ML-Techniken Modelle von Wahrnehmung erzeugt werden können. Diese erlauben es den lernenden Systemen unter anderem, ihre Unsicherheit in der eigenen Wahrnehmung zu nutzen. Strukturell betrachtet ist jedoch beispielsweise der in Abschnitt 3.7 vorgestellte Ansatz nicht trivial mit anderen Lernansätzen kombinierbar. Besonders im Forschungsfeld mobiler Roboter dominieren aktuell Reinforcement Learning (RL) Ansätze [83, 86]. In die dort eingesetzten Lernarchitekturen lässt sich die im vorherigen Kapitel verwendete Bayessche Inferenz basierend auf Multinomial- und Dirichlet-Verteilungen nicht direkt übertragen. Aktuelle State-of-the-Art Architekturen im RL nutzen hingegen tiefe Neuronale Netze als Funktionsapproximatoren, beispielsweise von Value-Funktionen.

Wie in Kapitel 1 dargestellt, ist jedoch der Großteil dieser Architekturen nicht in der Lage Aussagen zur Unsicherheit von Vorhersagen zu treffen. Dies wäre jedoch besonders bei einem Einsatz dieser lernenden Systeme in Echtweltszenarien enorm relevant. Wenn nicht erkannt werden kann, ob das System auf die Situation, in der es sich befindet, trainiert worden ist, können fehlerhafte Vorhersagen zu Fehlentscheidungen und potentiell gefährlichen Handlungen führen. Dieses Problem stellt auch ein zentrales Hindernis für einen industriellen Einsatz von RL-Systemen dar, da hier zum einen meist eine extrem geringe Toleranz gegenüber Fehlern vorherrscht, und zum anderen auch regulatorische Vorgaben den Nachweis eines garantiert sicheren Betriebs verlangen.

Die im Rahmen dieses Kapitels vorgestellten Verfahren zur Unsicherheitsmodellierung und Out-of-Distribution Erkennung können hier einen wichtigen Beitrag leisten.

4.1 Vorveröffentlichungen

Die Kerninhalte dieses Kapitels wurden vom Autor bereits in [126] sowie [127] veröffentlicht. Wie in Abschnitt 1.2 dargestellt, stammt die grundlegende Idee, das Konzept sowie die Umsetzung und Evaluation des Ansatzes vom Autor dieser Arbeit. Dies beinhaltet insbesondere den Ansatz zur Quantifizierung von epistemischer Unsicherheit, basierend auf RL-Value-Funktionen, als auch das Design und die Realisierung geeigneter Evaluationsumgebungen.

4.2 Motivation und Grundidee

Wie eingangs dargestellt, lässt sich die Komplexität einer Vielzahl aktueller Problemstellungen nur mehr durch lernende Systeme bewältigen.

Handelt es sich dabei um eine Problemstellung, in der ein oder mehrere Agenten aktiv mit einer Umgebung in einem zeitlichen Verlauf interagieren, sind RL-Methoden, wie in Abschnitt 2.1.3 dargestellt, der State-of-the-Art Lösungsweg. Trotz beeindruckender Erfolge dieser Ansätze in den vergangenen Jahren [96, 133, 140] werden die entsprechenden Systeme bisher fast ausschließlich in rein virtuellen Umgebungen (Werbeoptimierung, Videospiele etc.) oder kontrollierten Laborumgebungen eingesetzt.

Bevor mit RL trainierte Systeme in Echtwelt-Situationen eingesetzt werden können, ist es unabdinglich, die Verlässlichkeit und Sicherheit gründlich zu evaluieren. Beispielsweise kann eine naive Spezifikation von Zielvorgaben, wie von Amodei et al. [5] ausgeführt, zu unerwarteten, möglicherweise gefährlichen Resultaten führen. Ebenso entsteht das grundlegende Problem, dass es in der Regel nicht möglich ist, alle Situationen, die im Produktiveinsatz auftreten können, vorab zu lernen, da die lernenden Systeme meist auf hochdimensionale Zustandsbeschreibungen in einem zeitlichen Verlauf angewiesen sind.

Eine Ursache warum dies nicht möglich ist sind Kapazitätsgründe: Das Sammeln von Echtwelt-Situationen, in Form von Kamera- und anderen Sensor-Daten, ist meist mit hohem Aufwand und Kosten verbunden. Zudem muss der Agent zur Lösung von RL-Problemen während des Trainings in der Lage sein, aktiv mit der Umgebung zu interagieren, um Auswirkungen von Aktionen ermitteln zu können. Im Gegensatz zu reinen Supervised-Learning Ansätzen ist es für RL-Probleme also nicht ausreichend, vor dem Training ein „statisches“ Datenset zu sammeln. Erst die Interaktion mit der Umgebung generiert die notwendigen Daten in Form der Zustandsbeobachtung und des Reward-Signals (siehe 2.1.3). Diese Art des Trainings in der realen Welt ist zeitaufwändig und teuer. Zudem ist es oftmals aus Sicherheitsgründen nicht möglich gewisse Situationen (beispielsweise Unfälle im Straßenverkehr zum Training autonomer Autos) herbeizuführen.

Somit sind die zum Training zur Verfügung stehenden Daten in der Praxis stets limitiert und nicht in der Lage alle möglichen Fälle abzudecken. Ein üblicher Weg ist es deshalb, das Training in eine Simulationsumgebung zu verlagern. Die Simulationsumgebung dient somit als Datengenerator für das lernende System.[1] Dies bringt einige Vorteile mit sich, beispielsweise kann das Training durch schnellere Ausführung als Echtzeit beschleunigt werden. Ebenso können gefährliche Situationen gefahrlos simuliert werden. Andererseits entsteht dadurch jedoch ein neues Problem, da die Simulationsumgebung die Realität in der Regel nicht perfekt und erschöpfend abbilden kann. Dieses Delta wird

[1] Dies ist auch der Weg, der im Rahmen dieser Arbeit durch die Verwendung von prozeduralen Umgebungsgeneratoren beschritten wird.

auch als „Reality Gap“ [67] bezeichnet.[2] Das Problem, dass im Training nicht alle möglichen Situationen abgedeckt werden können, wurde durch ein Training in Simulation somit nicht gelöst, sondern nur verschoben, da weiterhin die im Produktiveinsatz auftretenden Echtweltdaten von den Trainingsdaten abweichen können.
Dieses grundlegende Problem, das auch *Distribution Shift* [77] oder *Dataset Shift* [114] genannt wird, lässt sich in vielen Einsatzfällen der realen Welt nicht verhindern. Die Wunschvorstellung, dass die (besonders mit Deep Learning Methoden) gelernten Modelle auch auf diesen abweichenden Datenverteilungen gute Ergebnisse liefern, wird aktuell in der Praxis nur selten erreicht. Auch wenn eine perfekte Generalisierung der Modelle natürlich erstrebenswert ist, ist diese aktuell kaum zu garantieren. Genau hier setzt der in diesem Kapitel vorgestellte Ansatz an: Auch wenn die Generalisierung und Performance der Modelle bei Distribution Shift aktuell nicht garantiert werden kann, so eröffnet die Modellierung von Unsicherheit doch zumindest die Möglichkeit adäquat auf die jeweilige Situation reagieren zu können.
Eine weitere Ursache, die dazu führt, dass dieses Problem selbst bei enormem Kapital- und Zeitaufwand zur Datensammlung nicht endgültig gelöst werden kann ist, dass nicht alle Situationen, deren Auftreten möglich ist, vorab bekannt sind. Es ist also nicht möglich zu bestimmen, ob der Trainingsdatensatz bzw. Trainingsdatengenerator das Problem bereits vollständig abdeckt, oder durch äußere Umstände neue Situationen und damit Daten auftreten können, die beim Training nicht in Betracht gezogen wurden. Um in dieser Ausgangssituation nun die Anforderungen an Verlässlichkeit und Sicherheit trotzdem erfüllen zu können, sind zusätzliche Schritte notwendig.
Ein möglicher Weg, der im Folgenden näher untersucht wird, ist das Erkennen von untrainierten, abweichenden Situationen. Diese werden im Folgenden als Out-of-Distribution (OOD) bezeichnet. Die Begriffe *Situation*, *Zustand* und *Daten* werden dabei äquivalent verwendet, da es sich aus Sicht eines RL-Agenten stets um die nach einer ausgeführten Aktion a_t von der Umgebung erhaltenen Informationen (State s_t in Unterabschnitt 2.1.3) handelt.
Diese Zielsetzung der Erkennung von OOD Situationen enthält zusätzlich zu den zuvor aufgezeigten Problemen einen weiteren Aspekt, der für weitere Komplexität sorgt: Da die untrainierten Situationen per Definition vorab unbekannt sind, muss die Erkennung möglich sein, ohne dass Beispiele dieser Daten vorab verfügbar sind. Es ist folglich nicht möglich, beispielsweise mittels Supervised Learning, einen klassischen Klassifikator zur Unterscheidung zu konstruieren, der zum Optimierungszeitpunkt sowohl trainierte als auch untrainierte Situationen als Eingabe benötigt. Diese Art der Problemstellung wird je nach Anwendungsgebiet und verwendetem Ansatz als Out-of-Distribution (OOD) Erkennung, Anomalieerkennung, Novelty Detection oder Outlier Detection bezeichnet [112]. Grundsätzlich ist das Ziel hierbei stets Daten, die sich von den

[2] Diese „Sim2Real“ genannte Problemstellung sowie der entsprechende Forschungsbereich sind jedoch nicht im Fokus der vorliegenden Arbeit.

Trainingsdaten unterscheiden, zu erkennen beziehungsweise als abweichend zu klassifizieren. Dieses Ziel wird auch als One-Class Klassifikation bezeichnet. Für einen umfassenden Überblick zu klassischen Methoden auf niedrigdimensionalen Daten kann auf Pimentel et al. [112] verwiesen werden.
Ist eine solche Erkennung abweichender Daten zuverlässig gegeben, ist die Ausgangsbasis geschaffen, um auf diesen Zustand adäquat reagieren zu können. Wie eine solch adäquate Reaktion aussehen kann hängt vom jeweiligen Einsatzgebiet ab. Ein naheliegender Ansatz wäre es die Kontrolle des Systems an einen Menschen zu übergeben. Amodei et al. [5] weist hier jedoch auf verschiedene Probleme hin: Die Skalierbarkeit eines solchen Ansatzes ist durch den limitierenden Faktor Mensch schwer zu gewährleisten. Zudem kann es in zeitkritischen Anwendungsfällen schlicht zu viel Zeit erfordern, bis der Mensch die Kontrolle geeignet übernehmen kann. In diesen Fällen könnte ein Umschalten auf eine weniger performante, dafür robustere, automatische Verhaltensweise geeignet sein. Wie genau die Behandlung der untrainierten Situationen aussehen sollte, ist jedoch nicht im Fokus dieser Arbeit. Stattdessen steht hier die Schaffung der notwendigen Ausgangslage im Vordergrund: Das zuverlässige Erkennen von unbekannten, untrainierten Situationen in einem RL-Kontext.

4.3 Grundlagen

Im folgenden Abschnitt werden zunächst die zum Verständnis notwendigen Grundlagen zur One-Class Klassifikation eingeführt. Darauf folgt eine Betrachtung üblicher Verfahren zur Evaluation von Binären Klassifikatoren. In diesem Rahmen werden Grundbegriffe wie *True Positive* und *False Positive* ebenso eingeführt wie Metriken zur Qualitätsbewertung.

4.3.1 One-Class Klassifikation

In Abschnitt 2.1.1 wurden Klassifikationsprobleme als Teil der Supervised Learning Probleme eingeführt. Die Annahme war hier, dass Trainingsdaten in der Form von Eingabevektoren x zusammen mit ihren korrespondierenden Zielvektoren y zur Verfügung stehen. Im Rahmen von Klassifikationsproblemen stellen die Zielvektoren y die Zielkategorien, auch Labels genannt dar. Aus mathematischer Sicht ist das Ziel dieser Klassifikationsprobleme die Konstruktion einer Funktion $f(x)$, die jedem Eingabevektor x' eine der Zielkategorien zuweist. [16]
Bei einem Klassifikationsproblem mit zwei Klassen könnten die Zielkategorien beispielsweise enkodiert werden als: $y_i \in \{-1, 1\}$, so dass $f(x'|X) : \mathbb{R}^D \Rightarrow [-1, 1]$. Im Falle eines One-Class Klassifikationsproblems unterscheidet sich dies nun insofern, als dass zum Training lediglich Eingabedaten und Labels einer Klasse zur Verfügung stehen. Ursache hierfür ist oftmals, dass die Generierung von Daten der abweichenden Klassen teuer ist, mit Sicherheitsrisiken

verbunden, oder eine Beeinflussung der Systeme schlicht nicht möglich ist[3]. Ziel von ML-basierten Ansätzen ist es nun, ein Modell $M(\Theta)$ zu erzeugen, das die Funktion $f(x)$ implementiert und somit Eingabedaten korrekt klassifiziert. Verschiedene Klassifikationsalgorithmen können basierend auf dem Typ ihrer Ausgabe unterschieden werden: Zur Kategorie der diskreten Klassifikatoren gehören Algorithmen, die direkt eine diskrete Ausgabe erzeugen, die einem der Klassenlabels entspricht. Die andere Kategorie (Scoring Klassifikatoren genannt) erzeugt eine kontinuierliche Ausgabe, die als Score bezeichnet wird. Dieser kann je nach Algorithmus eine interpretierbare Größe sein, beispielsweise die Wahrscheinlichkeit der Klassenzugehörigkeit. Um einen binären Klassifikator zu erhalten muss dieser Score anschließend mittels eines zusätzlichen Schwellwertes t auf eine der Klassen abgebildet werden.
Werden probabilistische ML-Verfahren zur Erzeugung des Modells M verwendet, wird primär der Begriff der Out-of-Distribution Erkennung verwendet. Im probabilistischen ML ist die Grundannahme, dass beobachtete Daten X von einem unbekannten, zugrundeliegenden datengenerierenden Prozess erzeugt werden. Die von diesem Prozess generierten Daten folgen dabei einer gewissen Wahrscheinlichkeitsverteilung $D = P(X)$. Durch Training von Neuronalen Netzen kann diese Wahrscheinlichkeitsverteilung approximiert werden als $\hat{D} \simeq D$. Das Neuronale Netz stellt in diesem Fall also das Modell M dar. Im Rahmen der Out-of-Distribution Erkennung wird nun ein Schwellwert $t = z(x)$ auf $\hat{D}$ definiert, so dass Daten, die über diesem Wert liegen ($z(x) > t$), als Out-of-Distribution klassifiziert werden. [112]

4.3.2 Evaluation von Binären Klassifikatoren

Im Rahmen von Klassifikationsproblemen mit exakt zwei Klassen, also binärer Klassifikation, werden die beiden Klassen in der Regel als *Positive P* und *Negative N* bezeichnet. Ein binärer Klassifikator weist einem Eingabedatum eine dieser beiden Klassen zu. Im Rahmen der in dieser Arbeit vorgestellten Out-of-Distribution Klassifikationsansätze, werden Daten die vom Modell abweichen (also OOD sind) als *Positive P* definiert. Die verwendeten Begrifflichkeiten folgen dabei den Definitionen zur Evaluation binärer Klassifikatoren nach Fawcett [44], die im Folgenden vorgestellt werden:
Betrachtet man die möglichen Resultate bei Anwendung eines binären Klassifikators auf ein Eingabedatum, so sind vier Fälle zu unterscheiden:

1. Das Eingabedatum besitzt die tatsächliche Klasse P (Positiv) und wird vom Klassifikator auch als P klassifiziert, so wird dies als *Richtig Positiv*: **tp** (*True Positive*) bezeichnet.

[3] So ist es beispielsweise in der Regel nicht möglich, sämtliche Fehlerfälle von Industrieanlagen oder auch Fehlentscheidungen autonomer Fahrzeuge in der Realität auszuführen, da Menschenleben gefährdet werden könnten. Ebenso existiert eine Vielzahl natürlicher Systeme, die schlicht nicht zur Erzeugung abweichender Daten beeinflusst werden können, beispielsweise astronomische Himmelskörper oder Prozesse der Erdatmosphäre.

2. Wird das selbe Eingabedatum fälschlicherweise als N klassifiziert, so ist dies ein *Falsch Positiv*: **fp** (*False Positive*).

3. Das Eingabedatum besitzt die tatsächliche Klasse N (Negativ) und wird vom Klassifikator auch als N klassifiziert: *Richtig Negativ*: **tn** (*True Negative*).

4. Wird das selbe Eingabedatum fälschlicherweise als P klassifiziert, so ist dies ein *Falsch Negativ*: **fn** (*False Negative*).

Die Anzahl der vorhergesagten Positiven wird als PP (Predicted Positive) bezeichnet, während die Anzahl der vorhergesagten Negativen als PN (Predicted Negative) bezeichnet wird.
Um die Qualität eines binären Klassifikators zu bewerten, kann nun für ein Set an Testdaten die Anzahl der jeweiligen Resultate gezählt werden. Basierend auf diesen Werten ist es möglich verschiedene Kennzahlen zu approximieren:
Richtig-positiv-Rate (True positive rate, recall):

$$\textbf{tpr} \approx \frac{tp}{P} = \frac{\text{Anzahl Richtig Positiv}}{\text{Absolute Anzahl Positiv}}$$

Falsch-positiv-Rate (False positive rate):

$$\textbf{fpr} \approx \frac{fp}{N} = \frac{\text{Anzahl Falsch Positiv}}{\text{Absolute Anzahl Negativ}}$$

Richtig-negativ-Rate (True negative rate, specificity):

$$\textbf{tnr} \approx \frac{tn}{N} = \frac{\text{Anzahl Richtig Negativ}}{\text{Absolute Anzahl Negativ}}$$

Falsch-negativ-Rate (False negative rate):

$$\textbf{fnr} \approx \frac{fn}{P} = \frac{\text{Anzahl Falsch Negativ}}{\text{Absolute Anzahl Positiv}}$$

Grundsätzlich ist die Qualität eines Klassifikators umso besser, je höher die *tpr* und je geringer die *fpr* ist.
Um die Qualität weiter auf eine einzelne Kennzahl zusammenzufassen, können verschiedene Verhältnisse berechnet werden [69]. Die hier wohl bekannteste Kenngröße ist die *Genauigkeit* (eng. Accuracy). Diese ist definiert als

$$ACC = \frac{tp + tn}{P + N} \tag{4.1}$$

Die Präzision, auch Positive Predictive Value genannt, wird berechnet als

$$PPV = \frac{tp}{PP} \tag{4.2}$$

Diese kann weiter mit der tpr kombiniert werden zum sogenannten F1-Score:

$$F_1 = 2 \times \frac{PPV \times tpr}{PPV + tpr} \tag{4.3}$$

Dieser bildet das harmonische Mittel aus Präzision und Richtig-positiv-Rate und ist eine häufig eingesetzte Metrik zur Bewertung von ML-basierten, binären Klassifikatoren. Auch die Evaluation des in diesem Kapitel vorgestellten Ansatzes verwendet den F1-Score zum Performance-Vergleich verschiedener Klassifikatoren.

4.4 Modellierung von Unsicherheit und probabilistisches ML

In Abschnitt 2.2 wurden die Grundlagen von Unsicherheit, deren unterschiedliche Typen sowie Möglichkeiten zur Quantifizierung eingeführt. Dieser Abschnitt beschäftigt sich nun mit existierenden Möglichkeiten zur Modellierung dieser Unsicherheit bei Verwendung von aktuellen ML-Verfahren.

Wie in Kapitel 3.3.2 vorgestellt, liefert die Bayessche Inferenz einen systematischen Weg zum Umgang mit Unsicherheit. Durch die Kombination dieser mit Neuronalen Netzen können sogenannte Bayessche Neuronale Netze [91, 100] erzeugt werden. Realisiert wird dies, indem jedes Gewicht eines Neuronalen Netzes nun nicht mehr wie üblich durch eine einzelne Fließkommazahl repräsentiert wird, sondern stattdessen durch eine eigene Wahrscheinlichkeitsverteilung [91]. Wird dieser Ansatz auf tiefe Neuronale Netze angewendet, entstehen jedoch aufgrund der hohen Anzahl an Parametern Performanceprobleme. Eine exakte Bestimmung des Posteriors ist hier meist unmöglich, da die Berechnung exponentiell in der Anzahl der Parameter ist [17]. Erst in den letzten Jahren hat der Forschungsbereich der Bayesschen Neuronalen Netze durch die Entwicklung neuer, approximativer Verfahren, die die Performanceprobleme auf Kosten der Posterior-Qualität lösen, wieder mehr Aufmerksamkeit erhalten. Diese Ansätze lassen sich in zwei Kategorien gruppieren: Approximative Inferenzmethoden sowie Modell-Ensembles. Im Folgenden werden zwei Ansätze detailliert vorgestellt, die später zur Evaluation des entwickelten OOD Verfahrens verwendet werden: Monte-Carlo Dropout und Bootstrap Ensembles.

4.4.1 Monte-Carlo Dropout

Ein viel beachteter Ansatz zur Modellierung von Unsicherheit in tiefen Neuronalen Netzen wurde von Gal und Ghahramani [50] vorgestellt. Der *Monte-Carlo Dropout* (MC Dropout), oder auch *Dropout Variational Inference* genannte Ansatz kann in die Kategorie der approximativen Bayesschen Verfah-

ren eingeordnet werden[4]. Im Gegensatz zu klassischen Bayesschen Neuronalen Netzen werden die Gewichte nicht durch Wahrscheinlichkeitsverteilungen modelliert, sondern weiterhin durch einzelne Fließkommazahlen. Stattdessen wird vor jedem Gewichts-Layer des Netzes ein Dropout-Layer eingefügt. Dropout [135] wendet eine zufällige, unabhängige Bernoulli Maske auf die Aktivierungen an und schützt so vor einer zu starken Co-Adaption der Gewichte. Diese Dropout-Layer werden nun während des Trainings, und im Gegensatz zur üblichen Verwendungsweise von Dropout, auch während der Vorhersagephase aktiv geschaltet. Unsicherheitsvorhersagen können nun erzeugt werden, indem mehrere voneinander unabhängige Forward-Passes durch das Neuronale Netz durchgeführt werden. Aufgrund der aktiven Dropout-Layer sind die Forward-Passes nun nicht mehr deterministisch, sondern stochastisch. Durch empirische Schätzung der ersten beiden Momente der so erzeugten Vorhersageverteilung kann eine Monte-Carlo Schätzung erzeugt werden.

4.4.2 Modell-Ensembles

Ein alternativer, wenn auch nicht strikt Bayesscher Weg, beruht auf der Konstruktion von Modell-Ensembles [34]. Modell-Ensembles bestehen aus einer Menge von Modellen, deren individuelle Vorhersagen auf geeignete Weise kombiniert werden. Die empirische Erkenntnis ist dabei, dass Ensembles oftmals bessere Ergebnisse erzielen, als einzelne Modelle. Im Folgenden liegt der Fokus jedoch nicht auf der Vorhersagequalität selbst, sondern auf der Zielsetzung, Modell-Ensembles zur Unsicherheitsmodellierung einzusetzen.
Osband et al. [108] basieren ihren Ansatz dabei auf der Idee des statistischen Bootstrap [35]. Im Vergleich zu einfachen Modell-Ensembles wie in [80] kann so zusätzlich die Diversität der einzelnen Ensemblemitglieder erhöht werden, was die Unsicherheitsschätzung verbessert. Zusätzlich erhöhen Osband et al. die Performance im Vergleich zu klassischen Ensembles, indem die unteren Layer eines Neuronalen Netzes zwischen allen Ensemblemitgliedern geteilt werden. Aus Architektursicht entsteht somit ein einzelnes Neuronales Netz, das K unabhängige, sogenannte *Heads* in der Ausgabeschicht besitzt. Jeder dieser Heads repräsentiert dabei ein einzelnes Ensemblemitglied, das sich die unteren Layer des Netzes mit allen anderen Ensemblemitgliedern teilt. Um den Bootstrap zu realisieren wird für jeden Datenpunkt eine Boolsche Maske (Bootstrap Maske genannt) der Größe K generiert. Diese Maske legt fest, für welchen Head der jeweilige Datenpunkt im Training sichtbar ist. Die Werte der Maske wiederum werden durch Ziehen von K Stichproben aus einer Bernoulli-Zufallsverteilung bestimmt. Bernoulli-Zufallsverteilungen werden durch einen Parameter p definiert. Dieser legt im Kontext der Boolschen Maske fest, wie wahrscheinlich

[4]Es sollte jedoch darauf hingewiesen werden, dass die Einordnung von MC Dropout als Bayessches Verfahren nicht unumstritten ist. Osband, Aslanides und Cassirer [106] zeigen verschiedene Beispiele auf, in denen die Approximation des Bayesschen Posteriors mittels MC Dropout beliebig schlecht ausfallen kann.

ein einzelner Eintrag der Maske *True* ist, und damit, wie wahrscheinlich es ist, dass ein Datenpunkt für einen Head im Training sichtbar ist.
In [106] wurde eine Weiterentwicklung dieses Ansatzes präsentiert, die die Unsicherheitsschätzung weiter verbessern soll. Die grundlegende Architektur bleibt erhalten, wird jedoch um ein sogenanntes *Prior Netz* erweitert. Dieses Prior Netz wird mit zufälligen Gewichten initialisiert, die jedoch im Training nicht verändert werden. Dennoch ist die Ausgabe dieses Netzes datenabhängig. Durch Aufaddieren des datenabhängigen Outputs des Prior Netzes auf den Output der verschiedenen Bootstrap Heads wird der Ensemble Posterior berechnet. Die Autoren zeigen, dass diese Architektur der reinen Bootstrap Architektur überlegen ist. Als Ursache wird die Tatsache vermutet, dass bei der reinen Bootstrap Architektur die initialen Gewichte des Netzes sowohl als Prior, als auch als Trainingsstartpunkt fungieren.

4.5 Verwandte Arbeiten aus dem Bereich der Unsicherheitsmodellierung für lernende Systeme

Verschiedene Methoden aus beiden Kategorien, Variational Inference sowie Modell-Ensembles, wurden bereits für verschiedenste Anwendungsfälle eingesetzt. In [74] bringen die Autoren MC Dropout in verschiedenen Computer-Vision Problemen zum Einsatz. Sie untersuchen dabei die Leistungsfähigkeit des Ansatzes zur Quantifizierung von Unsicherheit in verschiedenen Bildklassifikations-Datensets.
Ebenfalls auf Bilddaten erfolgt die Analyse in [84]. Die Autoren untersuchen in dieser Studie, ob die MC Dropout Architektur in der Lage ist, hilfreiche Unsicherheitsschätzungen bei der Erkennung von Krankheiten zu liefern. Die erzielten Ergebnisse zeigen, dass die Klassifikationsperformance erhöht werden konnte, indem unsichere Ergebnisse an eine weiterführende Untersuchung verwiesen wurden.
Im Forschungsbereich der Entscheidungstheorie wurde in [108] Unsicherheitsmodellierung in Form von Modell-Ensembles zum Einsatz gebracht. Die so modellierte epistemische Unsicherheit wird implizit genutzt, um die Exploration eines RL-Agenten zu verbessern. Eine explizite Berechnung oder Quantifizierung der epistemischen Unsicherheit wird jedoch nicht vorgenommen.
Ebenfalls zur Entscheidungsfindung setzt [72] Modell-Ensembles ein. Hier erfolgt die Unsicherheitsmodellierung jedoch nicht direkt im Agenten, sondern in einer eigenständigen, zusätzlichen Komponente, die zur Vorhersage einer Kollisionswahrscheinlichkeit verwendet wird.
In [9] stellen die Autoren einen probabilistischen RL-Ansatz vor. Dieser versucht die komplette Value-Verteilung eines Agenten in einer RL-Umgebung zu modellieren, anstatt die Values auf einzelne Werte zu reduzieren. Auf diese Weise wird aleatorische Unsicherheit in der Value-Verteilung modelliert. Somit

unterscheidet sich das Ziel dieses Ansatzes grundlegend von dem im Folgenden vorgestellten Verfahren UBOOD zur Erkennung von OOD Situationen mittels Modellierung von epistemischer Unsicherheit.

4.6 Konzept zur Unsicherheitsmodellierung für lernende Systeme

Der folgende Abschnitt präsentiert das entwickelte Konzept zur Modellierung von Unsicherheit für lernende Systeme. Grundelement ist die Kombination und Erweiterung existierender Architekturen des probabilistischen ML für die Unsicherheitsmodellierung in RL-Systemen. Der Fokus liegt hierbei auf der Anpassung der approximativen Bayesschen Inferenzmethode MC Dropout sowie von Bootstrap Ensembles. Zielsetzung der so entwickelten Architekturen ist es, beide Formen der Unsicherheit, aleatorische sowie epistemische, korrekt modellieren zu können. Zur Anwendung kommen die Architekturen schließlich mit dem Ziel, eine auf der epistemischen Unsicherheit basierende Erkennung von OOD Situationen zu realisieren.

4.6.1 Architekturen zur Deep-Q Learning basierten Unsicherheitsmodellierung

Die entwickelten Architekturen, die im Folgenden vorgestellt werden, wurden für einen Einsatz im Rahmen von Value-basiertem Deep RL entwickelt. Wie in Unterabschnitt 2.1.3 erläutert, kommen im Deep-Q Learning tiefe Neuronale Netze als Funktionsapproximatoren zum Einsatz. Diese werden eingesetzt, um die optimale Action-Value Funktion Q^* zu approximieren: $Q(s_t, a_t; \Theta) \approx Q^*(s_t, a_t)$.

4.6.1.1 MVE Bootstrap Ensemble

In Unterabschnitt 4.4.2 wurde die in [108] vorgestellte Architektur der Bootstrap Ensembles bereits eingeführt. Osband et al. konnten in ihrer Arbeit zeigen, dass diese Architektur in der Lage ist, gerichtete tiefe Exploration in einem RL-Setting, basierend auf epistemischer Unsicherheit, zu realisieren. Ziel der im Rahmen dieser Arbeit neu entwickelten Erweiterung dieser Architektur ist eine direktere Quantifizierung und Nutzung der modellierten Unsicherheit zur OOD Erkennung. Folglich ist es notwendig, neben der epistemischen Unsicherheit, auch die Aleatorische korrekt zu erfassen, um beide voneinander unterscheiden zu können.
Unsere Architekturerweiterung, *MVE Bootstrap Ensemble* genannt, basiert auf dem von Nix und Weigend [104] entwickelten Mean-Varianz-Estimation (MVE) Verfahren für Neuronale Netze zur Modellierung von Mittelwert und Standardabweichung einer Wahrscheinlichkeitsverteilung. Durch Anfügen einer zusätzli-

chen *auxiliary output unit* in jedem Head (σ-Neuron genannt), wird das Netz in die Lage versetzt, neben der klassischen Punktvorhersage des Mittelwerts der Zielverteilung auch die Varianz mit auszugeben. Die auf diese Weise modellierte Varianz erfasst die inhärent in den Daten vorhandene aleatorische Unsicherheit (siehe Unterabschnitt 2.2.4). Da die Varianz einer Verteilung niemals negativ sein kann, muss eine entsprechende Aktivierungsfunktion (beispielsweise die Softplus- oder Exponentialfunktion) gewählt werden, um die Ausgabe eines σ-Neurons entsprechend einzugrenzen. Als Lossfunktion zur Optimierung der Netzgewichte wird die negative Log-Likelihood der Ausgabeverteilung minimiert.

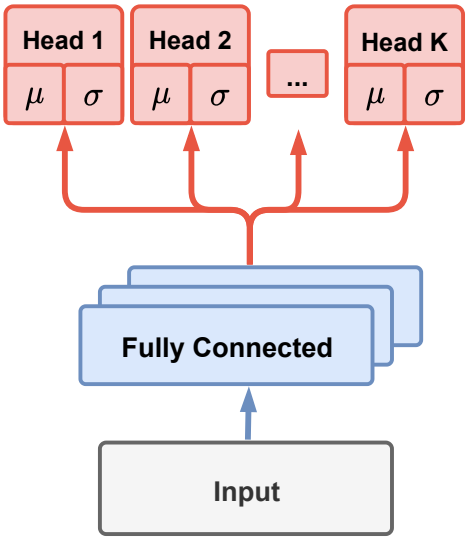


Abbildung 4.1: MVE Bootstrap Ensemble Erweiterung. Die Erweiterung der klassischen Architektur um σ-Neuronen ermöglicht die Modellierung von aleatorischer und epistemischer Unsicherheit.

Wie in Abbildung 4.1 ersichtlich, weicht die vorgestellte Architektur leicht von der von Nix et al. entwickelten ab, da sich alle Heads der Ausgabeschicht die versteckten Layer teilen. Das Teilen von Layern realisiert auch hier, wie bereits in der Bootstrap Ensemble Architektur vorgestellt, Performancezugewinne. Im Falle der Verwendung der Bootstrap Architektur mit zusätzlichem Prior Netz werden die Heads äquivalent erweitert.

4.6.1.2 MVE MC Dropout

Einem vergleichbaren Ansatz folgte [74] in der dort vorgestellten Architektur zur Modellierung von aleatorischer und epistemischer Unsicherheit mittels MC Dropout. Wir bezeichnen diesen im Folgenden als MVE MC Dropout Architektur. Auch hier ermöglicht die Erweiterung der Ausgabeschicht die zusätzliche Modellierung aleatorischer Unsicherheit. Die Netzgewichte können wie auch im MVE Bootstrap Ensemble durch Maximierung der Log-Likelihood der Ausga-

beverteilung optimiert werden. Abbildung 4.2 zeigt eine schematische Darstellung dieser Architektur, sowie die Ermittlung epistemischer Unsicherheit durch Berechnung der Varianz der Mittelwerte mehrerer Forward-Passes.

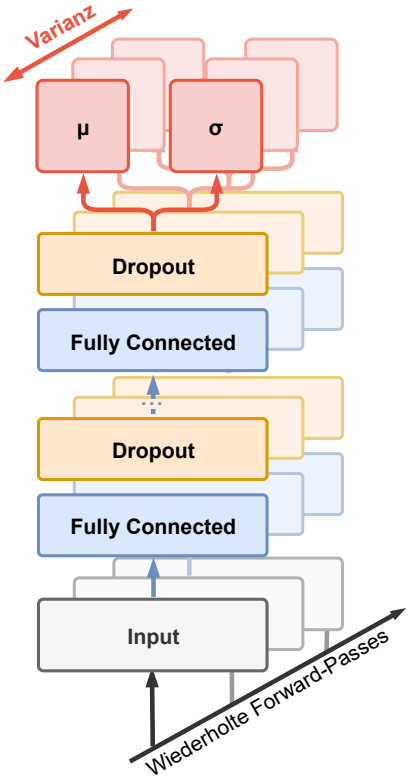


Abbildung 4.2: MVE MC Dropout Erweiterung. Die Erweiterung der klassischen Architektur um σ-Neuronen ermöglicht die Modellierung von sowohl aleatorischer als auch epistemischer Unsicherheit.

Wir passen diese Architektur weiter auf den Einsatz im Rahmen von RL-Problemen an. Wie in Unterabschnitt 2.1.3 beschrieben ist eine Kernherausforderung von RL-Problemen die Nicht-Stationarität der Daten. Grund hierfür ist, dass die zum Lernen verfügbaren Daten erst während des Trainings aktiv erzeugt werden. Die Datenverteilung ändert sich somit laufend. Dies würde eine manuelle Optimierung der Dropout-Raten beim Einsatz von klassischem Dropout weiter erschweren. In [51] stellten Gal et al. eine Weiterentwicklung des Dropout Verfahrens namens *Concrete Dropout* vor. Dieses erfordert keine Vorspezifizierung der Dropout-Raten, sondern lernt diese individuell je Layer. Wir kombinieren dies mit dem MVE MC Dropout Ansatz und nennen die resultierenden Architektur *MVE Monte-Carlo Concrete Dropout* (MVE MC-CD).

4.6.2 Modellierung epistemischer Unsicherheit zur Erkennung von OOD Situationen

Im folgenden Abschnitt wird der UBOOD (Uncertainty Based Out-of-Distribution) Klassifikator vorgestellt. Hierbei handelt es sich um einen neu entwickelten Ansatz zur Erkennung von OOD Situationen für tiefe, value-basierte RL-Anwendungsfälle, der die zuvor präsentierten Architekturerweiterungen zum Einsatz bringt. Grundlegendes Element dieses Ansatzes ist die Reduzierbarkeit der epistemischen Unsicherheit innerhalb der Action-Value Funktionsapproximation des Agenten.
UBOOD Klassifikatoren können grundsätzlich basierend auf verschiedenen, Value-basierten Deep RL-Architekturen konstruiert werden. Im Folgenden fokussieren wir uns jedoch auf den in Unterabschnitt 2.1.3 vorgestellten Deep-Q Learning [96] Ansatz. Dieser hat zum Ziel, die optimale Action-Value Funktion Q^* zu approximieren: $Q(s_t, a_t; \Theta) \approx Q^*(s_t, a_t)$.
Basierend auf dieser definieren wir $U_Q : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ als die epistemische Unsicherheitsfunktion einer gegebenen Q-Funktions-Approximation Q. Wird eine geeignete Methode zur Modellierung von epistemischer Unsicherheit in tiefen Neuronalen Netzen angewendet, so reduziert der Vorgang des Agenten-Trainings die Unsicherheit $U_Q(s, a)$ auf den im Training verwendeten State-Action Tupeln $(s, a) \in \mathbb{I}$, dem In-Distribution Daten-Set. Im Training verwendet bedeutet dies, dass für diese Zustände (s, a) ein Folgezustand s' sowie ein Reward-Signal r beobachtet und als (s, a, s', r) Tupel für die Optimierung von Q verwendet wurde. Im Unterschied dazu definieren die State-Action Tupel, die nicht im Training angetroffen wurden, d.h. $(s, a) \notin \mathbb{I}$ das Set der Out-of-Distribution Daten $\mathbb{O}$. Die epistemische Unsicherheit dieser State-Action Tupel wird durch das Training nicht reduziert. Folglich wird die epistemische Unsicherheit der Out-of-Distribution Daten höher sein, als die der In-Distribution Daten:

$$U_Q(\mathbb{O}) > U_Q(\mathbb{I}) \tag{4.4}$$

UBOOD Klassifikatoren nutzen die Ausgabe der epistemischen Unsicherheitsfunktion U_Q direkt als realwertigen Klassifikationsscore. Im Folgenden wird die Funktionsweise erläutert, wie basierend auf diesem Klassifikationsscore eine binäre Klassifikationsentscheidung getroffen werden kann.
Grundlage der Klassifikationsentscheidung bei Scoring Klassifikatoren bildet, wie in Unterabschnitt 4.3.1 erläutert, ein Klassifikationsschwellwert t. Dieser wird realisiert durch eine Funktion $y = z(x)$, die es erlaubt, einem Eingabedatum x ein binäres Label y zuzuweisen, und es so zu klassifizieren. Eine Modifikation von z beeinflusst folglich das Klassifikationsverhalten des Klassifikators. Je nach Anwendungsfall kann der Klassifikator so an die Anforderungen angepasst werden. Dies erfordert jedoch Anwendungswissen und ist nicht generalisierbar. Als ersten Ansatz zu einer möglichen generischen Lösung, im Kontext der Erkennung von OOD Situationen, präsentieren wir folgenden einfachen

Ansatz. Dieser ermöglicht eine dynamische Berechnung eines Klassifikationsschwellwerts t ohne Einsatz von zusätzlichem externen Anwendungswissen:

1. Berechne die durchschnittliche Unsicherheit der In-Distribution Samples:

$$\overline{U_Q} = \frac{1}{|\mathbb{I}|} \sum_{(s,a)\in\mathbb{I}} U_Q(s,a) \tag{4.5}$$

2. Interpretiere U_Q als Wahrscheinlichkeitsverteilung und definiere den Klassifikationsschwellwert als:

$$t = \overline{U_Q} + \sigma(U_Q) \tag{4.6}$$

Auf diese Weise wird ein dynamischer Schwellwert realisiert, der auf der Unsicherheitsverteilung basiert. Da sich diese Verteilung durch Sammeln von Daten im Verlauf des Trainings verändert, passt sich auch der Schwellwert dynamisch an. An dieser Stelle soll aber darauf hingewiesen werden, dass es jederzeit möglich ist, komplexere Algorithmen zur Realisierung der Schwellwertfunktion $z(x)$ zu entwickeln. Beispielsweise könnten multimodale Wahrscheinlichkeitsverteilungen verwendet werden, um U_Q präziser zu modellieren. Ebenso ist es möglich anwendungsfallspezifisches Wissen in die Implementierung von $z(x)$ einfließen zu lassen.

4.7 Experimentalsetup

Der folgende Abschnitt führt die zur Durchführung der Experimente verwendeten Simulationsumgebungen ein. Da im Rahmen der OOD Erkennung spezielle Eigenschaften der Umgebungen erforderlich sind, wird auf diese genauer eingegangen. Darauf folgend werden die verschiedenen evaluierten Algorithmen sowie deren Parametrisierung detailliert vorgestellt.

4.7.1 Simulationsumgebungen

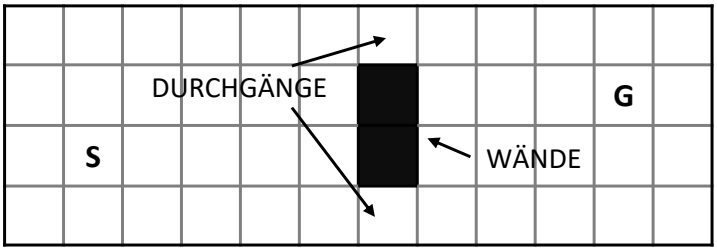
Die Evaluation von OOD Erkennungsverfahren für RL-Anwendungen steht zunächst vor einem nicht einfach lösbaren Problem: Es gibt nur wenige geeignete Datensets und Trainingsumgebungen, die eine kontrollierte und reproduzierbare Erzeugung von OOD Datenpunkten erlauben. Nur wenn diese Eigenschaften gegeben sind, ist ein valider Vergleich verschiedener OOD Erkennungsansätze realisierbar. Dies unterscheidet den Forschungsbereich der OOD Erkennung für RL von beispielsweise dem Forschungsbereich der Bildklassifikation, für den Benchmark-Datensätze wie *notMNIST* existieren, die OOD Datenpunkte enthalten. Für RL-Anwendungen existieren zum Entstehungszeitpunkt dieser Evaluation keine vergleichbaren Datensätze.

Um dieses Problem anzugehen, wurden zwei RL-Umgebungen entwickelt, die systematische Modifikationen nach dem Trainingsprozess ermöglichen. Auf diese Weise ist es möglich, OOD Zustände während der Evaluation zu erzeugen. Die Domänen unterscheiden sich grundsätzlich in der Art der Modellierung ihres Zustandsraums: Eine Umgebung verwendet einen Gridworld-basierten, diskreten Zustandsraum, während in der anderen ein kontinuierlicher Zustandsraum zum Einsatz kommt.

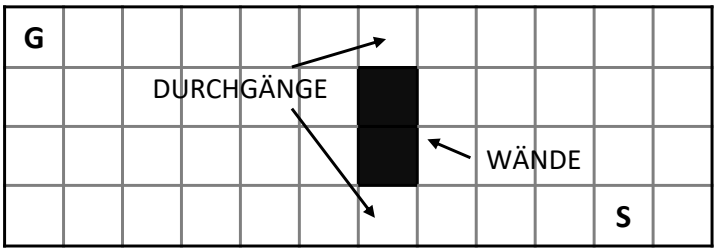
4.7.2 Gridworld

Evaluationsumgebung 1, Gridworld genannt, basiert auf einer einfachen, diskreten Gitterstruktur mit dem Ziel der Routenfindung. Das grundlegende Layout besteht aus zwei Räumen, die durch eine vertikale Wand voneinander getrennt sind. Bewegungen zwischen den Räumen sind nur durch zwei Durchgänge möglich, wie in Abbildung 4.3 visualisiert. Der Agent beginnt jede Episode an einer zufälligen Position auf dem Grid (Markierung **S** in Abbildung 4.3). Aufgabe des Agenten ist, die jeweilige Zielposition zu finden (Markierung **G** in Abbildung 4.3), die ebenfalls jede Episode zufällig variiert. Der Agent bewegt sich in jedem Zeitschritt eine Gridzelle fort, durch Auswahl einer der vier diskreten Handlungsmöglichkeiten: *{oben,unten,links,rechts}*. Der Zustand der Umgebung wird in Form von drei sogenannten Featureplanes der Größe 12×4 modelliert. Jede dieser Ebenen repräsentiert die räumlichen Positionen aller Umgebungsobjekte eines gewissen Typs: Agent, Ziel oder Wand. Jeder Schritt des Agenten erzeugt Kosten von -1, mit Ausnahme der Transition, durch die das Ziel erreicht wird. Das Rewardsignal dieses Zustandsübergangs, der die Episode beendet, beträgt $+100$.

Die Evaluation der UBOOD Klassifikatoren findet auf acht Konfigurationen der Umgebung statt. Alle Konfiguration besitzen die selbe Größe von 12×4 und variieren die Y-Koordinate der Startposition des Agenten sowie des Ziels zufällig im Intervall $[0, 4)$. Die Konfigurationen unterscheiden sich hingegen in den Intervallen, die die Variation der X-Koordinate der Startposition des Agenten (Startintervall) sowie des Ziels (Zielintervall) beschränken. Konfiguration 0 ist die Einzige, die im Training zum Einsatz kommt. Hier beträgt das Startintervall $[0, 5)$ und das Zielintervall $[7, 12)$. Jede Umgebungskonfiguration 1-7 ist dadurch definiert, dass das Startintervall, im Vergleich zum jeweils vorherigen, um 1 nach rechts verschoben wird, während das Zielintervall um 1 nach links verschoben wird. So hat Konfiguration 1 beispielsweise das Startintervall $[1, 6)$ und das Zielintervall $[6, 11)$. Dies erzeugt Umgebungsinstantiierungen mit zunehmendem Unterschied von der Trainingskonfiguration 0, wie im Beispiel in Abbildung 4.3b zu sehen.



(a) Beispiel einer Instantiierung der Umgebung: Konfiguration 0



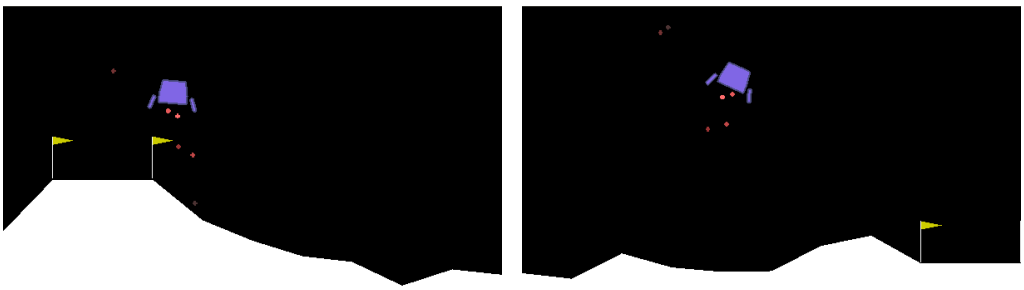
(b) Beispiel einer Instantiierung der Umgebung: Konfiguration 7

Abbildung 4.3: Beispielinstantiierungen der Gridworld Routenfindungsumgebung bei Verwendung verschiedener Konfigurationen. Die Markierung **S** zeigt die Startposition des Agenten, während **G** das Ziel markiert. Beide Positionen werden jede Episode zufällig innerhalb des in der jeweiligen Konfiguration definierten Wertebereichs gesetzt. (a) zeigt eine Beispielplatzierung bei Verwendung der Umgebungskonfiguration 0, die während des Trainings aktiv ist. (b) zeigt eine Instantiierung der Umgebungskonfiguration 7, die sich maximal von der Trainingskonfiguration unterscheidet.

4.7.3 LunarLander

Evaluationsumgebung 2, LunarLander genannt, basiert auf dem OpenAI LunarLander Environment [21] mit kontinuierlichem Zustandsraum. Ziel des Agenten ist es hier, eine Rakete unbeschädigt innerhalb einer definierten Landezone zu landen. Die Terraingenerierung erfolgt dabei jede Episode zufällig. Das zu lösende Problem ist somit die Optimierung einer Raketentrajektorie. Da sich die ursprüngliche Umgebung nicht für OOD Evaluationen eignet, wurde für die folgenden Experimente eine modifizierte Umgebungsversion entwickelt. Während die Landezone der Ursprungsumgebung statisch definiert ist, ermöglicht unsere Modifikation die zufällige Platzierung innerhalb eines spezifizierten Intervalls. Um diese Veränderung für den Agenten lernbar zu machen, wurde zusätzlich die Zustandsbeschreibung um die Position der Landezone erweitert. Die erweiterte Zustandsbeschreibung enthält nun zusätzlich die X-Koordinaten des linken, sowie rechten Endpunkts der Landezone sowie deren Y-Koordinate. Der Aktionsraum des Agenten ist diskret modelliert, und besteht aus den vier Aktionen: {Nichts aktiv, linker Antrieb aktiv, Hauptantrieb aktiv, rechter An-

trieb aktiv}.
Um die Qualität der UBOOD Klassifikatoren auf Umgebungen mit unterschiedlich starker Abweichung evaluieren zu können, wurden sechs verschiedene Konfigurationen definiert. Während die Rakete in allen Konfigurationen an derselben Position startet, unterscheiden sich hingegen die Intervalle, die die zufällige Position der X-Koordinate des Zentrums (X-Intervall) der Landezone sowie deren Y-Koordinate (Y-Intervall) beschränken. Konfiguration 0, die einzige im Training verwendete, beschränkt das X-Intervall auf $[2, 5)$ und das Y-Intervall auf $[6, 12)$. Dadurch befindet sich die Landezone im Training stets im oberen linken Bereich der Umgebung. Abbildung 4.4a zeigt eine Beispielinstantiierung dieser Konfiguration. Die weiteren Konfigurationen 1-5 werden dadurch definiert, dass das X-Intervall im Vergleich zum jeweils vorherigen um 1 nach rechts verschoben wird, während das Y-Intervall um 1 nach links verschoben wird. Dies führt dazu, dass die Landezone zunehmend weiter auf der unteren rechten Seite der Umgebung platziert wird, wie in Abbildung 4.4b zu sehen ist. Wie auch in der Gridworld Umgebung können so OOD Datenpunkte mit zunehmender Abweichung von der Trainingskonfiguration 0 generiert werden.



(a) Beispielinstantiierung der Umgebung: Konfiguration 0

(b) Beispielinstantiierung der Umgebung: Konfiguration 5

Abbildung 4.4: Visualisierung der Beispielinitialisierungen der LunarLander Umgebung bei Verwendung verschiedener Konfigurationen. Die Fläche zwischen den Fähnchen definiert die Landezone innerhalb derer der lilafarbene Agent unbeschädigt landen muss. (a) Konfiguration 0, die während des Trainings aktiv ist. Die Landezone befindet sich hier stets im oberen linken Bereich der Umgebung. Daten die unter Verwendung dieser Konfiguration gesammelt werden definieren das In-Distribution Set. (b) Konfiguration 5 initialisiert die Landezone im unteren rechten Bereich der Umgebung und unterscheidet sich damit maximal von der Trainingskonfiguration.

An dieser Stelle soll noch einmal darauf hingewiesen werden, dass das Training in beiden Umgebungen, Gridworld sowie LunarLander, ausschließlich auf

der jeweiligen Konfiguration 0 durchgeführt wird. Um das Verhalten der Unsicherheitsmodellierung sowie der UBOOD Klassifikatoren über den Trainingsverlauf analysieren zu können, werden während des Trainings laufend Kopien der Netze abgespeichert. Für jede dieser Netzkopien können anschließend mehrere Auswertungsdurchläufe auf unterschiedlichen Umgebungskonfigurationen durchgeführt werden. Daten, die während dieser Auswertungsdurchläufe erzeugt werden fließen nicht in das Training mit ein.

4.7.4 Algorithmen und Parametrisierung

Wie in Unterabschnitt 4.6.2 erläutert, können UBOOD Klassifikatoren basierend auf unterschiedlichen Ansätzen erzeugt werden. Die Voraussetzung ist lediglich, dass eine geeignete Methode zur Modellierung von epistemischer Unsicherheit in tiefen Neuronalen Netzen zum Einsatz kommt. Im Rahmen der durchgeführten Evaluation wurden so drei unterschiedliche UBOOD Klassifikatoren untersucht. Diese basieren im Kern auf den in Abschnitt 4.6 vorgestellten erweiterten Architekturen zur Unsicherheitsmodellierung:

- **UB-MC**: UBOOD mit MVE Monte-Carlo Concrete Dropout (MCCD)
- **UB-B**: UBOOD mit MVE Bootstrap Ensemble
- **UB-BP**: UBOOD mit MVE Bootstrap-Prior Ensemble

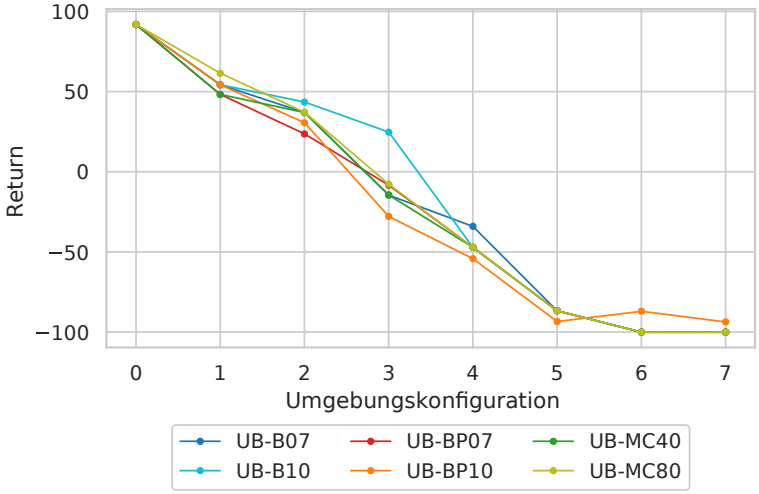
Die UB-MC Variante verwendet ein Neuronales Netz bestehend aus zwei Layern in der versteckten Schicht mit jeweils 64 Neuronen. Die Ausgabeschicht besteht aus zwei separaten Neuronen, die μ und σ einer Normalverteilung repräsentieren. Im σ-Neuron kommt zusätzlich die Softplus Funktion zum Einsatz, um einen positiven Wertebereich zu erzwingen. Da Concrete Dropout Layer wie in Abbildung 4.2 visualisiert zum Einsatz kommen, muss keine explizite Dropout-Rate als Hyperparameter festgelegt werden. Die Lossfunktion wird wie in Abschnitt 4.6 erläutert durch Minimieren der negativen Log-Likelihood der durch die Ausgabeschicht definierten Normalverteilung berechnet. Die Quantifizierung der epistemischen Unsicherheit erfolgt durch Berechnung der Varianz der Mittelwerte mehrerer Monte-Carlo Forward-Passes. Um hier den Effekt einer unterschiedlichen Anzahl an Monte-Carlo Durchläufen vergleichen zu können, werden zwei Parametrisierungen des UB-MC UBOOD Klassifikators erstellt: **UB-MC40** verwendet 40 Forward-Passes zur Approximation der epistemischen Unsicherheit, während bei **UB-MC80** 80 Durchläufe erfolgen.
Sowohl die UB-B als auch die UB-BP Ensemblevarianten verwenden ebenfalls Neuronale Netze bestehend aus zwei Layern in der versteckten Schicht mit jeweils 64 Neuronen. Diese versteckten Schichten werden wie in Unterabschnitt 4.4.2 dargelegt von allen $K = 10$ Heads der Ausgabeschicht gemeinsam verwendet. Jeder Head besteht wiederum aus zwei separaten Neuronen, die μ und σ einer Normalverteilung repräsentieren. Im σ-Neuron kommt zusätzlich

die Softplus Funktion zum Einsatz, um einen positiven Wertebereich zu erzwingen. Die Quantifizierung der epistemischen Unsicherheit erfolgt durch Berechnung der Varianz der Mittelwerte der 10 Heads. Bootstrap Ensembles sind wie in Unterabschnitt 4.4.2 erläutert weiter durch den Hyperparameter p der Bootstrap Maske parametrisiert. Um hier evaluieren zu können, ob der Einsatz von Bootstrapping einen Vorteil gegenüber reinen „nicht-bootstrap“ Ensembles mit sich bringt, werden jeweils zwei Parametrisierungen, sowohl des UB-B, als auch des UB-BP UBOOD Klassifikators erstellt: Bei **UB-B07** und **UB-BP07** wird $p = 0,7$ gesetzt. Bootstrapping ist somit aktiv. Die Wahrscheinlichkeit, dass ein Datenpunkt für einen einzelnen Head des Ensembles sichtbar ist, beträgt 70%. Bei **UB-B10** und **UB-BP10** wird $p = 1,0$ gesetzt. Dies deaktiviert Bootstrapping und erzeugt ein klassisches Ensemble, da jeder Datenpunkt mit Wahrscheinlichkeit 100% für jeden Head im Training sichtbar ist.
In allen zuvor beschriebenen Neuronalen Netzen kommt ReLU als Aktivierungsfunktion der Neuronen zum Einsatz, mit Ausnahme der μ-Neuronen, in denen keine Aktivierungsfunktion aktiv ist, sowie der σ-Neuronen, bei denen wie beschrieben die Softplus Funktion eingesetzt wird.
In Unterabschnitt 4.3.1 wurde erläutert, dass die Grundlage der Klassifikationsentscheidung bei Scoring Klassifikatoren ein Klassifikationsschwellwert t ist. Realisiert wird dieser durch eine Funktion $y = z(x)$, die es erlaubt einem Eingabedatum x ein binäres Label y zuzuweisen. Wie in Unterabschnitt 4.6.2 präsentiert, erfolgt bei UBOOD Klassifikatoren die Berechnung des Klassifikationsschwellwerts t dynamisch. Der Schwellwert wird basierend auf der durchschnittlichen Unsicherheit der In-Distribution Samples berechnet: $t = \overline{U_Q} + \sigma(U_Q)$. Entsprechend der Veränderung der Unsicherheitsverteilung während des Trainings passt sich der Schwellwert somit dynamisch an.

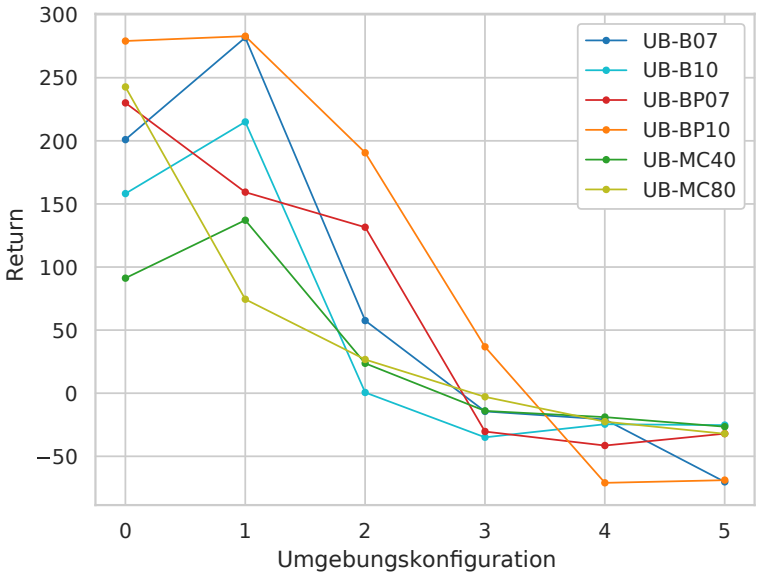
4.8 Evaluationsergebnisse

Der folgende Abschnitt präsentiert die Ergebnisse, die unter Verwendung des soeben beschriebenen Experimentalsetups erzielt wurden. Der Fokus liegt hierbei auf dem Verhalten der Unsicherheitsmodellierung, sowie der erzielten Klassifikationsperformance der unterschiedlichen UBOOD Klassifikatoren.

Zunächst ist festzustellen, dass alle evaluierten Verfahren auf beiden Evaluationsumgebungen erfolgreiche RL-Policies erlernen konnten. Die in Abschnitt 4.6 vorgestellten Architekturerweiterungen stehen somit dem primären RL-Ziel des Erlernens optimaler Policies nicht entgegen. Primär im Fokus steht jedoch das Verhalten der Architekturen in OOD Situationen. Wie in Unterabschnitt 4.7.1 erläutert, werden diese Situationen erzeugt, indem unterschiedliche Umgebungskonfigurationen generiert und anschließend die Policies darauf evaluiert werden. Trainingskonfiguration 0 ist dabei stets die Einzige, die während des Trainings zum Einsatz kommt.
Die folgenden Konfigurationen weisen mit ansteigender Nummer (Gridworld:



(a) Gridworld



(b) LunarLander

Abbildung 4.5: Erreichter Return der verschiedenen UBOOD Varianten auf unterschiedlichen Umgebungskonfigurationen: (a) der Gridworld und (b) der LunarLander Umgebung. Das Policy Training erfolgte jeweils über 10000 Episoden auf Konfiguration 0. Umgebungskonfigurationen > 0 modifizieren die jeweilige Umgebung mit zunehmender Stärke, wie in Unterabschnitt 4.7.1 erläutert. Alle gezeigten Werte sind Durchschnitte von 30 Evaluierungsdurchläufen.

1-7, LunarLander: 1-5) einen zunehmenden Unterschied von der Trainingskonfiguration auf, sie sind somit „zunehmend OOD“. Dieser Effekt spiegelt sich wie zu erwarten ist, im Return der Policies auf der jeweiligen Umgebungskonfigura-

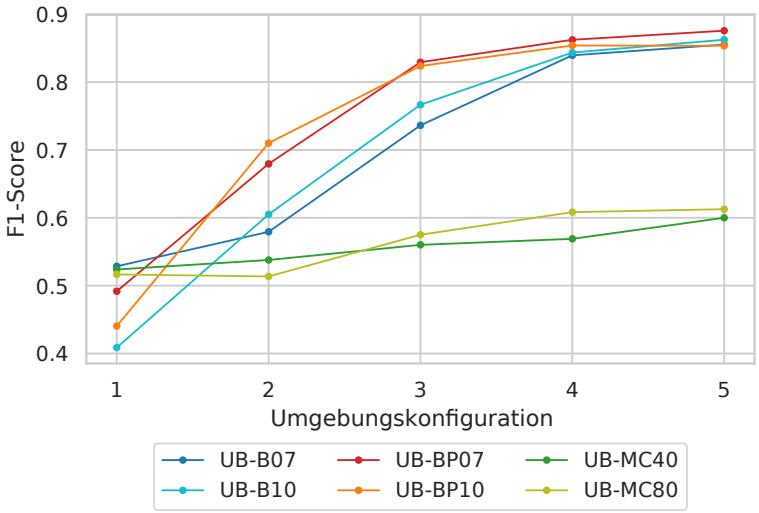
tion wider. Abbildung 4.5 zeigt den je Umgebung erreichten durchschnittlichen Return der trainierten Policies nach 10000 Trainingsepisoden. Um Varianz, verursacht durch die Stochastizität der Umgebungen, zu reduzieren, wurden hierfür Werte von 30 Evaluierungsdurchläufen gemittelt.

Wie am Abfall der Kurven in Abbildung 4.5 entlang der X-Achse zu erkennen ist, führt die zunehmende Abweichung einer Evaluationsumgebung von der Trainingsumgebung zu einer Reduktion des durchschnittlich erreichten Returns. Dieser Effekt ist über alle evaluierten UBOOD Varianten hinweg und auf beiden Evaluationsumgebungen Gridworld und LunarLander zu beobachten.

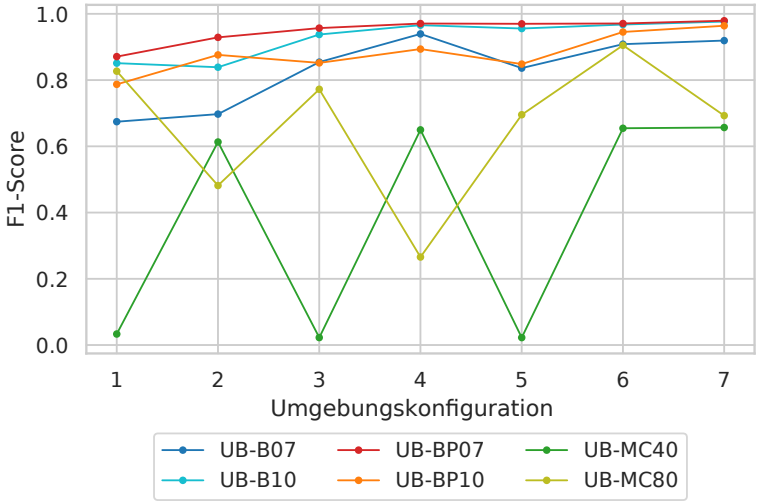
Im Folgenden wird die Leistungsfähigkeit des vorgeschlagenen UBOOD Ansatzes zur Erkennung von OOD Situationen analysiert. Hierfür wurden 30 Evaluationsdurchläufe je Umgebungskonfiguration sowohl der Gridworld, als auch der LunarLander Umgebung durchgeführt. Zustände, die unter Verwendung der Trainingskonfiguration 0 generiert wurden, sind Teil der In-Distribution Menge und definieren so die *Negativen* der jeweiligen Umgebung. Zustände der anderen Konfigurationen (Gridworld: 1-7, LunarLander: 1-5) wurden der Out-of-Distribution (OOD) Menge zugeordnet und definieren die *Positiven*. Um die Klassifikationsqualität des UBOOD Ansatzes evaluieren zu können, wird der F1-Score verwendet (siehe Unterabschnitt 4.3.2). Dieser bildet das harmonische Mittel von Präzision und Recall.

Abbildung 4.6 zeigt den erreichten F1-Score in Abhängigkeit des verwendeten Ansatzes zur Unsicherheitsmodellierung. Zunächst werden die erzielten Ergebnisse der Ensemble-basierten Klassifikatoren UB-B und UB-BP betrachtet. Die insgesamt besten Ergebnisse auf der LunarLander Umgebung wurden durch UB-BP Klassifikatoren erzielt (Abbildung 4.6a). Diese UBOOD Klassifikatoren verwenden ein MVE Bootstrap-Prior Ensemble zur Unsicherheitsmodellierung. Sie erreichen damit F1-Scores von maximal $0,903$ im Falle von UB-BP07 auf der am stärksten abweichenden Umgebungskonfiguration 5. Die Klassifikationsperformance von UB-BP10 fällt im Mittel geringfügig schlechter aus als die von UB-BP07. Es ist also festzuhalten, dass der Effekt von Bootstrapping im Vergleich zur reinen Ensemble-Variante (UB-BP10) einen geringfügig positiven Effekt mit sich bringt.

UBOOD Klassifikatoren, die Unsicherheitsmodellierung mittels Ensembles ohne zusätzlichen Prior realisieren (UB-B07 und UB-B10), zeigen ebenfalls gute F1-Scores, reichen jedoch im Mittel nicht an die Performance von Ensembles mit Prior heran (UB-BP07 und UB-BP10). Somit lässt sich feststellen, dass die Erweiterung der Architektur um einen zusätzlichen Prior, auch im Kontext der OOD Erkennung bei RL-Agenten, die Unsicherheitsmodellierung positiv beeinflusst. Dieser Gesamteindruck bestätigt sich auch auf der Gridworld Umgebung (Abbildung 4.6b). UB-BP07 erzielt hier ebenfalls die besten Klassifikationsresultate, knapp gefolgt von UB-B10. Die Performance der UBOOD Varianten UB-B07 und UB-BP10 fällt geringfügig niedriger aus. Insgesamt lässt sich feststellen, dass die Klassifikationsperformance aller Ensemble-basierten UBOOD Klassifikatoren (UB-B und UB-BP) mit zunehmender Stärke der Umgebungs-



(a) LunarLander



(b) Gridworld

Abbildung 4.6: F1-Scores der UBOOD Klassifikatoren, evaluiert auf verschiedenen Konfigurationen der (a) LunarLander und (b) Gridworld Umgebung. Zustände, die basierend auf der Trainingskonfiguration 0 gesammelt wurden, definieren die *Negativen* und damit die In-Distribution Menge der jeweiligen Umgebung. Zustände der anderen Konfigurationen definieren die *Positiven* und damit die Out-of-Distribution (OOD) Menge. Die X-Achse zeigt Evaluationsdurchläufe, die jeweils mit Zuständen der Trainingskonfiguration 0 und Zuständen der entsprechenden Umgebungskonfiguration > 0 durchgeführt wurden. Zustände wurden dabei über je 30 Episodenwiederholungen aggregiert.

modifikation (X-Achse in Abbildung 4.6) ansteigt.

Insgesamt sind die erreichten F1-Scores der Ensemble-basierten Klassifikatoren (UB-B und UB-BP) im Mittel deutlich höher als die der Monte-Carlo Dropout basierten (UB-MC). Diese erreichen auf der LunarLander Umgebung lediglich Werte zwischen $0,513$ und $0,612$. Die erzielten Ergebnisse auf der Gridworld Umgebung fallen zwar teilweise höher aus, jedoch auf Kosten von sehr hoher Varianz über die Umgebungsmodifikationen hinweg. Erreichte F1-Scores liegen hier zwischen $0,020$ und $0,656$ für die UB-MC40 Variante beziehungsweise $0,265$ und $0,905$ für die UB-MC80 Variante. Im Unterschied zu den Ensemble-basierten UBOOD Klassifikatoren ist hier über die Umgebungsmodifikationen hinweg kein klarer Trend erkennbar.

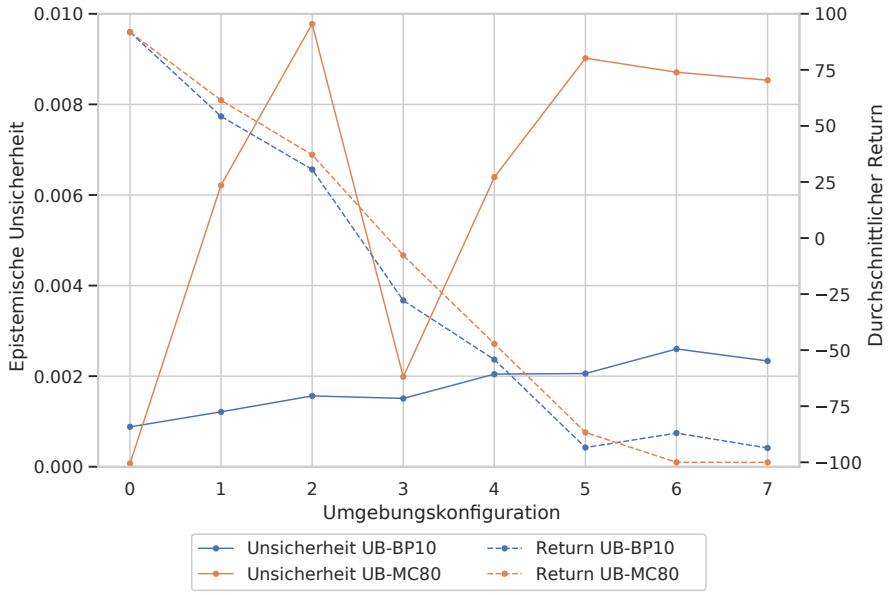


Abbildung 4.7: Gridworld

Abbildung 4.8: Gegenüberstellung von epistemischer Unsicherheit und durchschnittlichem Return in Abhängigkeit der Umgebungskonfiguration. Während bei beiden UBOOD Varianten der erreichte Return entlang der X-Achse abnimmt, gibt nur die Ensemble-basierte Variante UB-BP10 parallel dazu ansteigende Unsicherheit aus. Die Monte-Carlo Dropout basierte Variante UB-MC80 erzeugt stark schwankende Unsicherheitswerte. Beispielsweise fällt die ausgegebene epistemische Unsicherheit von Umgebungskonfiguration 2 zu 3 stark ab, obwohl gleichzeit der erzielte Return ebenfalls abnimmt. Alle gezeigten Werte wurden über je 30 Episodenwiederholungen gemittelt.

Um mögliche Ursachen für den deutlichen Performanceunterschied zwischen Ensemble- und Monte-Carlo Dropout basierten UBOOD Klassifikatoren zu bestimmen, wurden weitere Untersuchungen des Verhaltens der Unsicherheitsmodellierung einiger UBOOD Varianten durchgeführt.

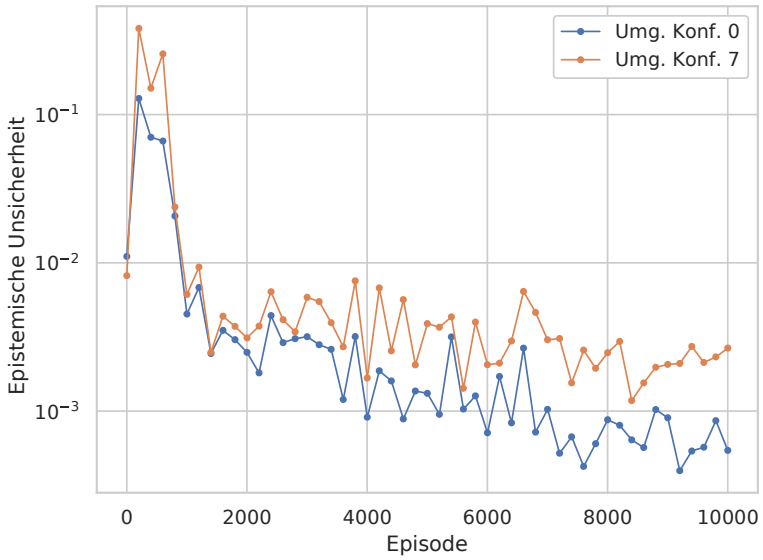
Abbildung 4.8 zeigt zunächst eine Gegenüberstellung von Unsicherheit und Return in Abhängigkeit der Umgebungskonfiguration am Beispiel der UB-BP10 und UB-MC80 UBOOD Varianten auf der Gridworld Umgebung. Eine zielführende Unsicherheitsmodellierung sollte bei zunehmender Abweichung der Umgebung zunehmende Unsicherheitswerte ausgeben. Obwohl bei beiden UBOOD Varianten der erreichte Return entlang der X-Achse abnimmt, gibt nur die Ensemble-basierte Variante UB-BP10 wie erhofft dazu ansteigende Unsicherheit aus. UB-MC80, die Monte-Carlo Dropout basierte Variante erzeugt dagegen stark schwankende Unsicherheitswerte. Evaluationen auf der LunarLander Umgebung zeigten entsprechendes Verhalten.

Als zweite Untersuchung der Ursachenermittlung wurden die epistemischen Unsicherheitswerte über den Trainingsverlauf hinweg analysiert. Hierfür wurden wie in Unterabschnitt 4.7.1 erläutert während des Trainings laufend Kopien der Netze abgespeichert. Für jede dieser Netzkopien wurden anschließend mehrere Auswertungsdurchläufe auf unterschiedlichen Umgebungskonfigurationen durchgeführt: Evaluationen auf der Umgebungskonfiguration 0 zeigen die Unsicherheitsmodellierung der In-Distribution Daten, während Umgebungskonfiguration 7 die Modellierung bei maximal abweichenden Situationen testet.

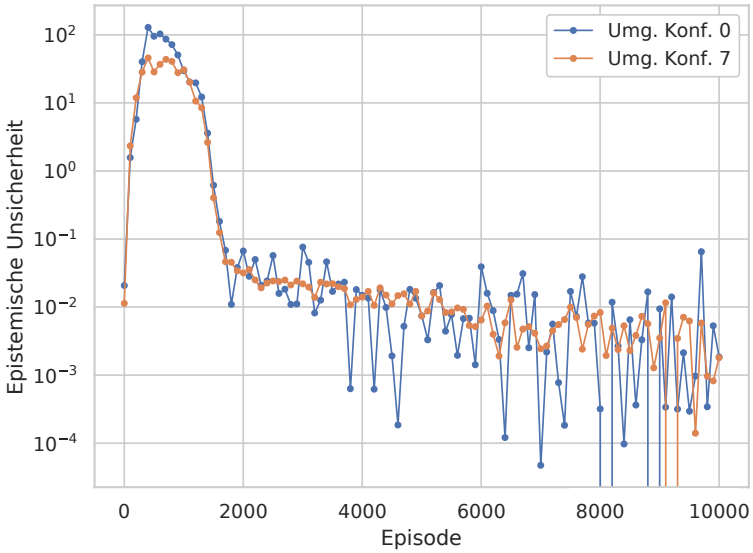
Abbildung 4.9 stellt exemplarisch die ausgegebenen epistemischen Unsicherheiten der UB-B07 und UB-MC80 Varianten auf der Gridworld Umgebung gegenüber. Die Ensemble-basierte UBOOD Variante UB-B07 zeigt über den Trainingsverlauf von 10000 Episoden das erwartete Verhalten: Die epistemische Unsicherheit auf der Trainingskonfiguration 0 nimmt mit zunehmender Verfügbarkeit von Daten über die Trainingsepisoden ab (blaue Kurve, Abbildung 4.9a), während die Unsicherheit auf der OOD Umgebungskonfiguration 7 auf einem höheren Level konvergiert (orangefarbene Kurve, Abbildung 4.9a). Dies bestätigt erstens die Annahme, dass die epistemische Unsicherheit der Action-Value Funktionsapproximation des Agenten durch mehr Daten reduzierbar ist. Und zweitens, dass dies nur für die In-Distribution Situationen gilt, und nicht für OOD Situationen. Diese zunehmende Divergenz bildet die Grundlage für eine zuverlässige OOD-Klassifikation.

Wie in Abbildung 4.9b exemplarisch zu sehen ist, zeigt die Dropout-basierte UBOOD Variante UB-MC80 diesen Effekt hingegen nicht. Die ausgegebenen epistemischen Unsicherheitswerte, zwischen In-Distribution- und OOD Situationen, unterscheiden sich mit fortschreitendem Training nicht zunehmend. Dies ist die Ursache, weshalb bei Einsatz dieser Unsicherheitsmodellierungsansätze keine zuverlässige OOD-Klassifikation möglich ist.

Das soeben beschriebene Verhalten war konsistent über alle evaluierten Parametrisierungen der Ensemble- und Dropout-basierten Unsicherheitsmodellierungsansätze hinweg.



(a) UB-B07



(b) UB-MC80

Abbildung 4.9: Durchschnittliche epistemische Unsicherheit der (a) Ensemble-basierten Variante UB-B07 und (b) Monte-Carlo Concrete Dropout basierten Variante UB-MC80 auf der Gridworld Umgebung. *Umg. Konf. 0* zeigt ausgegebene Unsicherheitswerte auf der Trainingskonfiguration (In-Distribution), *Umg. Konf. 7* Unsicherheiten auf der maximal abweichenden Umgebungskonfiguration (Out-of-Distribution). Alle gezeigten Werte wurden über je 30 Episodenwiederholungen gemittelt.

4.9 Zusammenfassung

In diesem Kapitel wurden verschiedene Wege zur Unsicherheitsmodellierung bei Verwendung von aktuellen ML-Verfahren betrachtet. Der Fokus lag hier auf Ansätzen, die mit Value-basierten RL-Methoden kombiniert werden können. Dies ist von hoher Relevanz, da es unabdinglich ist, die Verlässlichkeit und Sicherheit solcher lernenden Systeme zu garantieren, bevor sie in Echtwelt-Situationen eingesetzt werden. Als möglicher Weg um dieses Ziel zu erreichen, wurde das Erkennen von abweichenden Out-of-Distribution (OOD) Situationen vorgestellt. Die hier diskutierten Forschungsergebnisse präsentieren erste unsicherheitsbasierte OOD Erkennungsansätze, die in sequentiellen Entscheidungsproblemen eingesetzt werden können.

Hierfür wurde das OOD-Erkennungsproblem als One-Class Klassifikationsproblem modelliert. Dies bildet die Grundlage des entwickelten UBOOD (Uncertainty Based Out-of-Distribution) Ansatzes. Grundlegendes Element hiervon ist die Reduzierbarkeit der epistemischen Unsicherheit innerhalb der Action-Value Funktionsapproximation des Agenten. Um den UBOOD Ansatz zu realisieren, wurden zunächst existierende Architekturen des probabilistischen ML aus dem Bereich der approximativen Bayesschen Inferenzmethoden sowie Ensemblemethoden für die Unsicherheitsmodellierung in RL-Systemen geeignet angepasst. Diese Architektur-Erweiterungen erlauben schließlich eine Erkennung von OOD Situationen in tiefen, Value-basierten RL-Anwendungsfällen, basierend auf epistemischer Unsicherheit. Zur Evaluation verschiedener UBOOD Varianten wurden zwei RL-Umgebungen entwickelt, die systematische Modifikationen nach dem Trainingsprozess ermöglichen. Dies erlaubt es, OOD Zustände während der Evaluation zu erzeugen.

Die Evaluationsergebnisse zeigen, dass der UBOOD Ansatz ein gangbarer Weg ist. Die Hypothese hat sich bestätigt, dass die epistemische Unsicherheit der Value-Funktion eines RL-Agenten zur Erkennung von OOD Situationen genutzt werden kann. Es hat sich jedoch auch gezeigt, dass die OOD Klassifikationsgüte direkt von der Qualität der Unsicherheitsmodellierung abhängig ist. Somit ist eine möglichst gute Unsicherheitsmodellierung Grundvoraussetzung. Auf beiden evaluierten Umgebungen zeigten UBOOD Varianten, die eine Ensemble-basierte Unsicherheitsmodellierung einsetzen (UB-B / UB-BP) sehr gute Resultate. F1-Scores von bis zu $0,903$ ermöglichen eine zuverlässige Unterscheidung von In- und Out-of-Distribution Situationen. Auch die Erwartungen an das Verhalten der Unsicherheitsmodellierung bei zunehmender Stärke der Veränderung der Umgebung haben sich bei den Ensemble-basierten Varianten erfüllt. Mit zunehmender Stärke der OOD Situationen wurden höhere epistemische Unsicherheitswerte ausgegeben und bessere F1-Klassifikationscores erzielt. Die Hinzunahme eines Priors in den UB-BP Varianten zeigte ebenfalls einen positiven Einfluss, was sich in höheren F1-Scores niederschlägt.

Diesen positiven Resultaten stehen die UBOOD Varianten gegenüber, die auf der approximativen Bayesschen Inferenzmethode Concrete MC Dropout ba-

sieren. Diese UB-MC Varianten waren nicht in der Lage eine zuverlässige OOD Klassifikationsqualität zu erreichen. Auch wenn die Erhöhung der Anzahl an verwendeten Monte-Carlo Forward-Passes von 40 auf 80 zu einer leichten Verbesserung führte, reichte die Leistungsfähigkeit nicht an die Ensemble-basierten UBOOD Varianten heran. Die Ursache hierfür zeigte eine Analyse der Unsicherheitsmodellierung über den Trainingsverlauf (Abbildung 4.9). Bei den Ensemble-basierten UBOOD Varianten divergieren die ausgegebenen epistemischen Unsicherheitswerte zwischen In- und OOD Situationen mit fortschreitendem Training zunehmend (Abbildung 4.9a). Dies bestätigt die Annahme, dass die epistemische Unsicherheit der Action-Value Funktionsapproximation des Agenten durch mehr Daten reduzierbar ist: Die zunehmende Verfügbarkeit von Daten über die Trainingsepisoden hinweg reduziert die Unsicherheit von Situationen der Trainingskonfiguration, nicht jedoch von Situationen der OOD Konfigurationen. Diese zunehmende Divergenz ermöglicht eine zuverlässige OOD-Klassifikation. Die Dropout-basierten UBOOD Varianten hingegen zeigen diesen Effekt nicht (Abbildung 4.9b). Da sich der Unterschied der ausgegebenen epistemischen Unsicherheitswerte, zwischen In-Distribution- und OOD Situationen, mit fortschreitendem Training nicht vergrößert, ist bei Einsatz dieser Unsicherheitsmodellierungsansätze keine zuverlässige OOD-Klassifikation möglich. Dieses grundlegende Verhalten war konsistent über alle evaluierten Parametrisierungen der Ensemble- und Dropout-basierten UBOOD Varianten hinweg.
Diesbezüglich ist es interessant, dass die erzielten Ergebnisse mit kürzlich veröffentlichten Resultaten von Beluch et al. [12] übereinstimmen. Die Autoren verglichen in ihrer Arbeit ebenfalls Ensemble-basierte Unsicherheitsmodellierungsansätze mit Monte-Carlo Dropout Basierten, jedoch in einer Active Learning Aufgabe zur Bildklassifikation. Die präsentierten Evaluationsergebnisse zeigen eine Überlegenheit der Ensembleansätze bei dieser Aufgabenstellung. Die Autoren konnten ebenfalls zeigen, dass die Ensembleansätze besser kalibrierte Unsicherheitsschätzungen liefern. Als mögliche Erklärung für die geringere Qualität der Monte-Carlo Dropout basierten Ansätzen wird die durch Dropout verringerte Kapazität der Modelle angeführt. Zudem wird eine im Vergleich zu Ensembles geringere Diversität vermutet.
Bezogen auf die in Abschnitt 4.8 präsentierten Ergebnisse der UBOOD Klassifikatoren würde dies die geringeren F1-Klassifikationsscores der UB-MC Varianten erklären.

5 Policy-basierte Erkennung von Out-of-Distribution Situationen

Die im vorherigen Kapitel 4 vorgestellten Ansätze wurden mit dem Ziel entwickelt, eine Unsicherheitsmodellierung im Kontext Value-basierter RL-Methoden zu ermöglichen. Die Evaluationsergebnisse bestätigten, dass dies erreicht werden kann, wenn Ansätze aus dem Bereich des probabilistischen ML mit Netzarchitekturen des Value-basierten Deep Q-Learnings kombiniert werden. Neben dem Erfolg dieser Ansätze existiert jedoch weiterhin eine offene Problemstellung: In einer Vielzahl aktueller Anwendungsgebiete werden State-of-the-Art-Lösungen mittels Policy-basiertem RL erzielt [54, 68]. Dies ist insbesondere im Bereich der Robotik der Fall. Die Gewährleistung der Sicherheit und Zuverlässigkeit der Systeme ist hier von besonderer Wichtigkeit, wenn die Roboter in räumlicher Nähe zu Menschen agieren sollen.
Der im Rahmen dieses Kapitels vorgestellte PEOC (Policy Entropy Out-of-Distribution Classifier) Ansatz ist speziell für einen Einsatz in Kombination mit Policy-basierten RL-Algorithmen entwickelt worden. Er kann somit einen ersten Beitrag zur Verbesserung der Sicherheit autonomer lernender Systeme leisten.

5.1 Vorveröffentlichungen

Die Hauptinhalte dieses Kapitels wurden vom Autor bereits in [129] veröffentlicht. Wie in Abschnitt 1.2 dargestellt stammen die grundlegende Idee, das Konzept sowie die Umsetzung und Evaluation des Ansatzes vom Autor dieser Arbeit. Dies beinhaltet insbesondere das entwickelte Verfahren zur Verwendung der Entropie der Policy eines Agenten als Klassifikationsscore zur Erkennung von Out-of-Distribution Situationen.

5.2 Motivation und Grundidee

Wie Eingangs erläutert stellt Policy-basiertes RL den aktuellen State-of-the-Art im Umfeld mobiler Roboter dar. Die Motorik dieser Roboter bildet in der Regel kontinuierliche Aktionsräume, die mit diskreten, Value-basierten RL-Ansätzen nicht direkt modellierbar sind. Als Beispiel kann hier Akkaya et al.

[4] genannt werden, die den Policy-basierten RL-Algorithmus PPO [123] zur Steuerung einer humanoiden Roboterhand verwenden. Auch Lee et al. [83] setzen einen Policy-basierten Ansatz TRPO [122] ein, um die gesamte Steuerung der Bewegung eines vierbeinigen Roboters über komplexes Terrain zu trainieren.
Um einen verlässlichen Einsatz dieser Systeme in Echtwelt-Situationen zu realisieren, ist es aber auch hier unabdingbar, die stets existente Unsicherheit in den Aktionsentscheidungen zu beachten. Es ist somit von hoher Relevanz, Ansätze zu entwickeln, die eine solche Unsicherheitsbestimmung bei Einsatz von Policy-basiertem RL ermöglichen. Ist eine entsprechende Modellierung der Unsicherheit gegeben, kann diese unter anderem dafür eingesetzt werden, untrainierte, vom Training abweichende Situationen zu erkennen. Dies ermöglicht es dem System in einem anschließenden Schritt angemessen auf diese Bedingungen zu reagieren.
In diesem Kapitel wird der hierfür entwickelte *PEOC* Ansatz vorgestellt. Der PEOC Ansatz ist auf verschiedene RL-Algorithmen aus den Policy-Gradient und Actor-Critic Familien [137] anwendbar. Einzige Voraussetzung ist die Verwendung einer stochastisch modellierten Policy. Kernidee von PEOC ist es, die Entropie der RL-Policy als Score eines Binären Klassifikators zur Erkennung von OOD Situationen zu verwenden. Da die OOD Erkennung als One-Class Klassifikationsproblem modelliert wird, werden zur Trainingszeit keine OOD Datenpunkte benötigt. Dies stellt einen Vorteil gegenüber anderen Ansätzen dar, besonders bei einem Einsatz des Systems in Echtweltszenarien, da hier die Erzeugung von In-Distribution Daten meist deutlich einfacher und kostengünstiger ist, als die Erzeugung von abweichenden OOD Daten.

5.3 Grundlagen zur Performance-Evaluation von Binären Scoring-Klassifikatoren

Der vorgestellte Ansatz PEOC zur Erkennung von Out-of-Distribution Situationen basiert auf einem One-Class Klassifikationsansatz. Die zum Verständnis der Performance-Evaluation notwendigen Grundlagen werden im Folgenden dargestellt. Allgemeine Grundlagen zur One-Class Klassifikation wurden bereits in 4.3.1 erläutert, während Grundlagen der Entropie als Mittel zur Quantifizierung der Zufälligkeit einer Zufallsvariable in Unterabschnitt 2.2.2 vermittelt wurden.
Wie zuvor erläutert, können binäre Klassifikationsalgorithmen dahingehend unterschieden werden, ob sie eine diskrete Ausgabe erzeugen, die einem der Klassenlabels entspricht, oder eine kontinuierliche Ausgabe, die als Score bezeichnet wird. Die Evaluation der Performance dieser auch Scoring-Klassifikatoren genannten Algorithmen ist aufwendiger, da hier ein zusätzlicher Parameter t zur Schwellwertdefinition benötigt wird. Da die Wahl von t die Klassifikationsentscheidung beeinflusst, verändert dies folglich auch die

Richtig-positiv- und Falsch-positiv-Rate des Klassifikators.

Um dennoch die Qualität eines binären Klassifikators bewerten und verschiedene Klassifikatoren vergleichen zu können, wurden Verfahren entwickelt, mit deren Hilfe die Abhängigkeit vom Schwellwertparameter visualisiert und interpretiert werden kann.

Eine weit verbreitete Darstellungsmethode sind ROC (Receiver Operating Characteristic) Graphen. Ursprünglich als Hilfsmittel für militärische Radarempfänger entwickelt, wird diese Darstellungsmethode heute in den verschiedensten Anwendungsgebieten, wie der Medizin [84], Meteorologie [76] oder der Entwicklung von ML-Algorithmen [48] eingesetzt. Die Darstellung basiert auf einem zweidimensionalen Graphen, bei dem die Richtig-positiv-Rate (*tpr*) auf der Y-Achse sowie die Falsch-positiv-Rate (*fpr*) auf der X-Achse aufgetragen wird. Ein Punkt im Graphen entspricht somit einem diskreten Klassifikator mit entsprechender *tpr* und *fpr*. Bei Verwendung von Scoring-Klassifikatoren mit einem Schwellwertparameter entsteht durch Variation des Parameters t eine Kurve im ROC Raum. Die ROC Graphen Analyse ermöglicht damit sowohl einen Vergleich der Qualität unterschiedlicher Klassifikatoren als auch eine Abwägung der Auswahl eines geeigneten Schwellwertparameters. Da ein optimaler Klassifikator alle *Positiven* korrekt als solche klassifiziert und keine *Falsch-positiven* aufweist, besitzt er eine $\text{tpr} = 1$ und eine $\text{fpr} = 0$, liegt im ROC Graphen also an der Koordinate $(0, 1)$. Ein Klassifikator an der Koordinate $(1, 0)$ würde alle Eingabedaten als *Negative* klassifizieren und besäße somit eine $\text{tpr} = 0$ und eine $\text{fpr} = 1$. Die Diagonale im ROC Graphen repräsentiert damit einen Klassifikator der die Klassenzugehörigkeit rät. Bei einer vorhergesagten gleichverteilten Klassenzugehörigkeit läge der Klassifikator auf der Diagonale an der Koordinate $(0.5, 0.5)$ Würde hingegen in 90% der Fälle als *Positiv* klassifiziert, würde die *tpr* zwar auf $0,9$ steigen, die *fpr* jedoch auch, so dass der Klassifikator ebenso auf der Diagonale im ROC Graphen läge. Ein Klassifikator der über der Diagonale liegt ist somit „besser als zufällig“ und nutzt in den Eingabedaten enthaltene Informationen für die Klassifikationsentscheidung.

Um die Performance mehrerer schwellwertabhängiger Scoring-Klassifikatoren vergleichen zu können, ist es nützlich die Qualität auf eine einzelne Kennzahl zu reduzieren. Hierfür kann die Fläche unterhalb der ROC Kurve: *ROC AUC* (Area under the curve) berechnet werden. Da die Y- und X-Achsen als *tp* bzw. *fp Rate* auf einen Wertebereich von $[0, 1]$ limitiert sind, liegt auch die Fläche unter der ROC Kurve und damit der ROC AUC Wert zwischen 0 und 1.

Äquivalent zu dem vorgestellten ROC Graphen Ansatz findet sich in der Forschungsliteratur auch der Precision-Recall (PR) Ansatz. Dieser visualisiert die Präzision ($\text{prec} = 1 - \text{fpr}$) gegenüber *tpr*. Der PR-Ansatz weist nach Davis und Goadrich [31] Vorteile auf, wenn eine starke Unausgewogenheit der Daten, also eine hohe Schiefe (skew) der Klassenverteilung vorliegt. Wie zuvor ausgeführt, stellt im Rahmen der OOD Erkennung die Verfügbarkeit vieler Daten des Normalzustandes, und weniger oder im Extremfall keiner Daten der

abweichenden Fälle, die Ausgangslage dar. Es wäre folglich anzunehmen, dass Precision-Recall Graphen stets das Mittel der Wahl zur Auswertung von OOD Klassifikatoren sind. Im Rahmen der vorliegenden Arbeit werden dennoch ROC Graphen sowie ROC AUC Werte zur Bewertung der Klassifikatorqualität verwendet. Der Grund hierfür ist, dass der Einsatz einer Simulationsumgebung die Erzeugung einer ausgewogenen Klassenverteilung zum Testen der Klassifikatoren ermöglicht. In diesem Fall sind ROC Graphen zu bevorzugen, da sie einige Nachteile der PR Graphen, wie beispielsweise unerreichbare Regionen, nicht aufweisen [20]. Würden die Klassifikatoren stattdessen basierend auf Echtweltdaten trainiert und analysiert, muss an dieser Stelle darauf hingewiesen werden, dass der Einsatz von PR Graphen in Erwägung gezogen werden sollte.

5.4 Konzept zur Policy-basierten Erkennung von Out-of-Distribution Zuständen

In diesem Abschnitt wird das entwickelte PEOC (Policy Entropy Out-of-Distribution Classifier) Konzept vorgestellt. Es wurde speziell zur Erkennung von Out-of-Distribution Situationen bei Einsatz von Policy-basierten RL-Ansätzen entwickelt. Die Basis des PEOC Verfahrens bildet die Policy π eines RL-Agenten. Voraussetzung ist die Verwendung einer stochastisch modellierten Policy. Diese Ausgangslage ist bei einer Vielzahl aktueller State-of-the-Art RL-Algorithmen gegeben, wie beispielsweise PPO [123] oder A3C [95].
Im Folgenden werden Zustände, die im Training verwendet wurden, als In-Distribution States (IND-States) $s_i \in \mathbb{I}$ bezeichnet. Die Menge aller In-Distribution States $\mathbb{I}$ enthält somit alle im Training verwendeten Zustände. Alle möglichen, nicht im Training gesehenen Zustände $s_o \notin \mathbb{I}$ definieren die Menge der Out-of-Distribution States (OOD-States). Das Problem der Erkennung von Out-of-Distribution Zuständen wird als One-Class Klassifikationsproblem modelliert, so dass nur die Menge der IND-States $\mathbb{I}$ zur Trainingszeit des Klassifikators notwendig ist.
Wie zuvor erwähnt, können PEOC Klassifikatoren basierend auf verschiedenen RL-Algorithmen konstruiert werden, beispielsweise aus den Policy-Gradient und Actor-Critic Familien [137]. In den entsprechenden Algorithmen wird die Policy als bedingte Wahrscheinlichkeitsverteilung $\pi = P(a|s)$ modelliert. Sie gibt somit die Wahrscheinlichkeit an, dass eine Aktion a in einem Zustand s ausgeführt wird. Zur Quantifizierung der Unsicherheit bzw. Zufälligkeit einer Wahrscheinlichkeitsverteilung kann, wie in 2.2.2 dargestellt, die Shannon Entropie H als Maß verwendet werden. Angewendet auf die stochastische Policy eines RL-Agenten quantifiziert die Policy Entropy $H(\pi)$ folglich, wie zufällig die vom Agenten gewählten Aktionen sind.
Abbildung 5.1 zeigt eine schematische Darstellung der grundlegenden Elemente dieses Ansatzes zur Unsicherheitsquantifizierung mittels Policy Entropy. Die

Verwendung der normalisierten Entropie H_n ermöglicht dabei eine Vergleichbarkeit über verschieden große Aktionsräume. Dies wäre im Rahmen der in dieser Arbeit durchgeführten Experimente nicht zwingend notwendig, da alle verwendeten Aktionsräume die gleiche Größe besitzen, dennoch erleichtert es die Interpretation der Entropiewerte über den Trainingsverlauf. Die genauen Varianten der State Repräsentation, des eingesetzten Neuronalen Netzes sowie die Form der Outputs des Policy Head können an den jeweiligen Anwendungsfall angepasst werden.

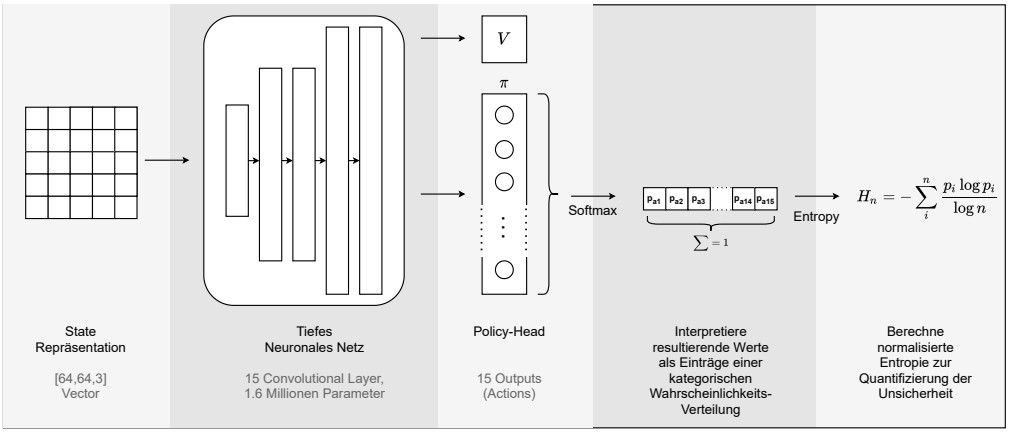


Abbildung 5.1: Schematische Darstellung der grundlegenden Modellarchitektur zur Unsicherheitsquantifizierung mittels Policy Entropy. Die in grau aufgeführten Parameter der State Repräsentation, die Architektur des tiefen Neuronalen Netzes sowie die Anzahl der Outputs, entsprechen dem Experimentalsetup in Abschnitt 5.7.

Wie in Unterabschnitt 2.1.3 beschrieben ist das Ziel von RL-Algorithmen den erwarteten zukünftigen Return zu maximieren. Dies wird erreicht, indem die Parameter der Policy so optimiert werden, dass die für dieses Ziel optimalen zustandsabhängigen Aktionen gewählt werden.
Kern des PEOC Verfahrens ist, dass unter der Annahme, dass in der gegebenen Problemdomäne optimales Verhalten nur durch gezielte, also nicht-zufällige Aktionen erreicht werden kann, die Entropie der Policy $H(\pi)$ für Zustände s_i, die im Training verwendet wurden, sinken muss. Wenn dieser Abfall der Entropie für im Training verwendete Zustände s_i stärker ist als für nicht gesehene Zustände s_o, kann die Policy Entropy $H(\pi)$ als Score eines binären Klassifikators zur Out-of-Distribution Erkennung verwendet werden.
Die Konstruktion eines optimalen Klassifikators ist somit möglich, wenn die Policy Entropy für alle Zustände $s_i \in \mathbb{I}$ kleiner ist als die Entropie aller Zustände $s_o \in \mathbb{O}$. Formal ausgedrückt:

$$H(\pi(s_i)) < H(\pi(s_o)), \forall s_i \in \mathbb{I}, \forall s_o \in \mathbb{O} \tag{5.1}$$

In diesem Fall existiert eine Entscheidungsgrenze, die eine perfekte Trennung der In- und Out-of-Distribution Zustände ermöglicht. Ausgedrückt in den Evaluationsgrößen binärer Klassifikatoren bedeutet dies, dass ein perfekter Klassifikator konstruiert werden kann, der eine True-Positive-Rate $\text{tpr} = 1$ und eine False-Positive-Rate $\text{fpr} = 0$ besitzt.

5.5 Verwandte Arbeiten

Frühere Arbeiten im Bereich von RL-Problemen haben sich bereits der Entropie als Messgröße bedient. Diese Arbeiten zur Entropie Regularisierung sowie das sogenannte Maximum Entropy RL unterscheiden sich jedoch in ihrer Zielsetzung grundsätzlich vom hier vorgestellten Ansatz, da bei diesen stets die Verbesserung der RL-Policy im Vordergrund steht. Im Gegensatz dazu verwendet das in dieser Arbeit vorgestellte Verfahren die Entropie nicht zur Verbesserung des Lernerfolgs, sondern nach abgeschlossenem Training zur Erkennung von OOD Situationen.
Dennoch ist der folgende Überblick zu diesen Verfahren relevant. Zum einen dienten diese als Inspiration für das im Rahmen dieser Arbeit entwickelte Verfahren, zum anderen wurde zur Evaluation von PEOC ein RL-Algorithmus eingesetzt, der Entropie Regularisierung verwendet.

5.5.1 Entropie Regularisierung

Williams und Peng [146] zeigen in ihrer Arbeit, wie die Policy Entropy $H(\pi)$ als Mittel zur Regularisierung verwendet werden kann. Regularisierung ist ein Ansatz aus dem Bereich der Statistik und wird meist zur Verhinderung von Overfitting, also einer Überspezialisierung auf die Trainingsdaten, eingesetzt [16]. Die Autoren der oben genannten Veröffentlichung erreichen diesen Effekt, indem sie $H(\pi)$ auf die Zielfunktion des RL-Algorithmus aufaddieren. Aktionen, die im jeweiligen Zustand eine höhere Entropie aufweisen, werden dadurch bevorzugt, weshalb $H(\pi)$ in diesem Kontext auch als Entropie-Bonus bezeichnet wird. Die Hypothese ist dabei, dass auf diesem Weg die Wahrscheinlichkeit einer vorzeitigen Konvergenz des Lernalgorithmus auf lokale Optima verringert werden kann. Kürzlich entwickelte RL-Algorithmen wie A3C [95] oder PPO [123] greifen diese Idee wieder auf und verbessern damit erfolgreich die Ergebnisqualität.

5.5.2 Maximum Entropy RL

Eine Erweiterung dieser Idee wird als Maximum Entropy Reinforcement Learning bezeichnet [42, 146]. Ziel dieser Algorithmen ist es, eine RL-Policy zu finden, die maximale Entropie aufweist, während weiterhin das eigentliche Aufgabenziel erreicht wird. Das so formulierte Optimierungsziel ist es, gleichzeitig

den gemeinsamen Erwartungswert der Summe der Rewards und der Entropie zu maximieren. Es werden also im Unterschied zum Ansatz der Entropie Regularisierung nicht nur Aktionen bevorzugt, die im aktuellen Zustand höhere Entropie aufweisen, sondern auch Aktionen, die im Erwartungswert zu Zuständen mit hoher Entropie führen.

5.6 Benchmarking Pipeline für OOD Klassifikation im Reinforcement Learning

Der folgende Abschnitt stellt eine Benchmarking Pipeline vor, die speziell für die OOD Klassifikation im Rahmen von RL-Systemen entwickelt wurde. Notwendig wird dies durch die unterschiedliche Funktionsweise, im Training sowie Testen, zwischen klassischen und Policy-basierten OOD Klassifikatoren. Um fehlerhafte beziehungsweise irreführende Vergleiche zu vermeiden, wurde ein strukturiertes Vorgehen entwickelt, das im Folgenden vorgestellt wird. Die gesamte Pipeline umfasst dabei das Training von RL-Policies, die Sammlung von IND- und OOD-Daten, die Optimierung von klassischen, Nicht-Policy-basierten Benchmark Klassifikatoren sowie die finale Qualitätsevaluation. Dieser Prozess ist in Abbildung 5.2 schematisch dargestellt.

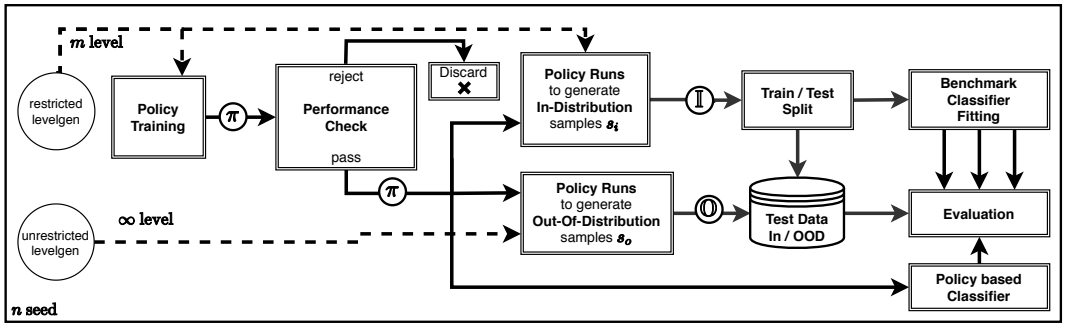


Abbildung 5.2: Benchmarking Pipeline für OOD Klassifikation im Reinforcement Learning. Die Wiederholung des gesamten Prozesses ermöglicht die Ermittlung zentraler Momente der Klassifikatorperformance, indem n unterschiedliche Zufalls-Seeds verwendet werden.

Einzelne Durchläufe des Prozesses unterliegen dabei verschiedenen Quellen von Varianz in der Qualitätsevaluation. Ursache hierfür sind zufällige Initialisierungen der Parameter, unter anderem der Gewichte des Neuronalen Netzes, sowie des Levelgenerators. Um dennoch zuverlässige Qualitätsaussagen treffen zu können, können durch mehrmaliges Wiederholen des gesamten Prozesses (Parameter n in Abbildung 5.2) zentrale Momente wie Erwartungswert und Varianz der ROC AUC Verteilung bestimmt werden. Ein einzelner Prozessdurchlauf wird dabei als *Prozesswiederholung* bezeichnet. Im Folgenden werden die einzelnen Schritte der Pipeline vorgestellt.

Als vorbereitender Schritt wird der Levelgenerator in jeder *Prozesswiederholung* auf einer unterschiedlichen Menge an Level der Größe m initialisiert (*restricted levelgen*). Diese Level dienen als Trainingsumgebung für das *Policy Training* über eine fixe Anzahl an Zeitschritten. Als Ergebnis des Trainings wird die resultierende Policy π einer Qualitätsprüfung unterzogen. Dies ist ein sinnvoller und notwendiger Schritt, der im Praxiseinsatz von RL-Anwendungen ebenfalls vollzogen werden muss, mit dem Ziel nur die besten Policies zum Einsatz zu bringen. Die Evaluation der OOD Erkennung auf schwachen Policies, die in der Praxis nicht eingesetzt würden, würde zu keiner relevanten Aussage führen. An dieser Stelle sei nochmals darauf hingewiesen, dass der vorgestellte Ansatz **nicht** die Verbesserung der Policy Performance zum Ziel hat, sondern die faire Evaluation von OOD Klassifikatoren in einem RL-Setting.

Schlägt dieser *Performance Check* genannte Test fehl, wird die Policy verworfen, und die nächste *Prozesswiederholung* wird gestartet. Im Erfolgsfall wird die Policy weiter zur Durchführung mehrerer Leveldurchläufe (*Policy Runs* genannt) verwendet. Zum einen unter Verwendung der selben Menge an m Level, die im *Policy Training* eingesetzt wurde. Während dieser *IND Runs* genannten Durchläufe werden alle Zustände (s_i) gesammelt. Diese bilden die Menge der In-Distribution Zustände $\mathbb{I}$. Zum anderen wird die Policy zur Durchführung separater Leveldurchläufe (*OOD Runs* genannt) verwendet, in denen der Levelgenerator unbeschränkt initialisiert wird. Dies führt dazu, dass in diesen Durchläufen mit hoher Wahrscheinlichkeit vom Training sowie den *IND Runs* abweichende Level generiert werden. Während dieser *OOD Runs* werden alle Zustände (s_o) gesammelt. Diese bilden die Menge der Out-of-Distribution Zustände $\mathbb{O}$.

Um einen fairen Vergleich mit klassischen Out-of-Distribution Klassifikatoren zu realisieren, wird die Menge der In-Distribution Zustände $\mathbb{I}$ geteilt (*Train/Test Split*). Der *Train* Teil wird zur Optimierung der klassischen, nicht-policy-basierten Benchmark Klassifikatoren verwendet. Policy-basierte Klassifikatoren wie PEOC benötigen keinen zusätzlichen Optimierungsschritt auf den IND Daten, da der Klassifikator auf dem Policy Netz basiert, das während des *Policy Trainings* bereits optimiert wurde.

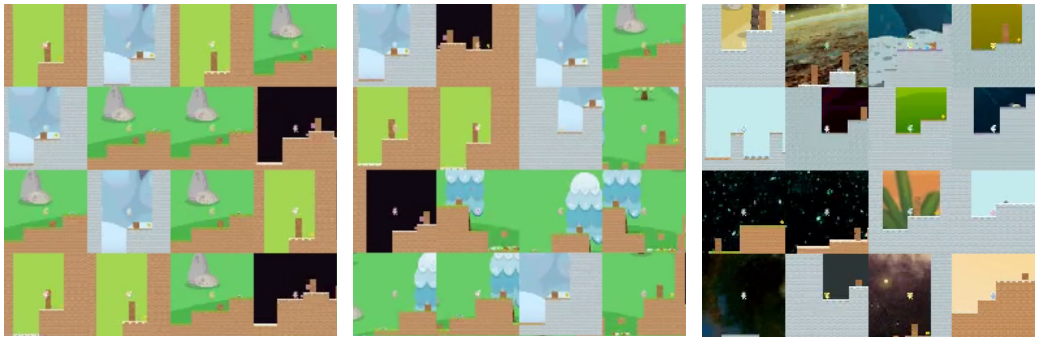
Der *Test* Teil von $\mathbb{I}$ wird anschließend mit der kompletten Menge $\mathbb{O}$ kombiniert und bildet so die *Test Data*. Auf diesen Daten wird die finale Qualitätsevaluation aller Klassifikatoren durchgeführt. Für dieses Datenset ist für jedes enthaltene Datum, basierend darauf, ob es ursprünglich aus $\mathbb{I}$ oder $\mathbb{O}$ stammt, die tatsächliche Klassenzugehörigkeit bekannt. Folglich ist es möglich ROC Graphen sowie ROC AUC Werte der verschiedenen Klassifikatoren zu berechnen. Die Wiederholung des gesamten Prozesses unter Durchführung mehrerer *Prozesswiederholungen* erlaubt es, wie zuvor erläutert, zentrale Momente wie Erwartungswert und Varianz der Klassifikatorperformance zu ermitteln.

5.7 Experimentalsetup

Eine stets wiederkehrende Schwierigkeit bei der Entwicklung und Evaluation von Out-of-Distribution Klassifikatoren ist die Verfügbarkeit von ausreichend Daten, sowohl der IND Klasse zum Training, als auch der OOD Klasse zur Evaluation. Diese Situation ist auch für RL-Probleme gegeben, in denen die Daten in Form von Zustandsbeobachtungen durch Interaktion von Agent und Umgebung generiert werden. Um neben IND Zuständen auch OOD Zustände generieren zu können, sind folglich unterschiedliche Umgebungsvarianten notwendig. Standard Benchmarkumgebungen für RL wie OpenAI Gym [**gym**] oder das Arcade Learning Environment [10] sind hierfür nicht geeignet, da es mit diesen nicht möglich ist unterschiedliche In- und Out-of-Distribution Zustände zu generieren. Ein wachsendes Forschungsinteresse an Fragestellungen zur Generalisierungsqualität von RL-Algorithmen hat in den letzten Jahren zur Entwicklung neuer Benchmarkumgebungen geführt. Das besondere an diesen Umgebungen ist, dass sie getrennte Trainings- und Test-Varianten realisieren [28, 43, 148]. Dadurch sind einige dieser Umgebungen auch zur Untersuchung von Fragestellungen rund um die Out-of-Distribution Erkennung geeignet.
An dieser Stelle kann angemerkt werden, dass die Verwendung von Videospiel-Umgebungen im Rahmen der RL-Forschung nicht ungewöhnlich ist. Diese eignen sich aufgrund ihrer episodischen Natur und überschaubaren Komplexität gut zur Evaluation von RL-Ansätzen. So verwenden [66] beispielsweise ein kompetitives First-Person Multiplayer-Spiel zur Evaluation. Die Brücke zu industriellen Use-Cases ist hier oftmals kleiner als es vielleicht zunächst den Anschein hat. So stellt eine Gridworld die Grundlage der im Rahmen von [110] entwickelten Smart Factory Umgebung dar. In [24] wiederum wird eine Restaurantsimulation zum Training kooperativer Multiagentensysteme eingesetzt. Das zu lösende, grundlegende Problem der Koordination und Ablaufsteuerung (Scheduling) existiert in vergleichbarer Weise in einer Vielzahl von industriellen Problemstellungen.
Im folgenden Abschnitt wird der in Abschnitt 5.4 vorgestellte PEOC Ansatz anhand der in Abschnitt 5.6 präsentierten Benchmarking Pipeline evaluiert. Als RL-Umgebung kommt eine ursprünglich für Generalisierungstests entwickelte Simulation zum Einsatz, die im Folgenden näher dargestellt wird. Des Weiteren werden die verwendeten Algorithmen (RL sowie Benchmarking Klassifikatoren) sowie deren Parametrisierung vorgestellt.

5.7.1 Simulationsumgebung

Zur Evaluation des neu entwickelten PEOC Ansatzes wird die CoinRun Simulationsumgebung in der Implementierung von Cobbe et al. [27] verwendet. Die Entscheidung, speziell diese als Levelgenerator einzusetzen, wurde getroffen, da der prozedurale Levelgenerierungsansatz gut geeignet ist, um eine diverse Menge an unterschiedlichen In- und Out-of-Distribution Situationen zu erzeugen.



(a) (b) (c)

Abbildung 5.3: Die Abbildungen zeigen je 4x4 Beispielzustände einer *Prozesswiederholung*, generiert in unterschiedlichen Phasen: a) *Policy Training* b) In-Distribution *Policy Runs* c) Out-of-Distribution *Policy Runs*. In den Phasen a) und b) ist der Levelgenerator beschränkt auf die gleichen 4 Level, während in Phase c) keine Beschränkung der Levelanzahl vorgenommen wird. In der Abbildung nicht zu sehen ist die Verwendung unterschiedlicher Zufalls-Seeds für jede *Prozesswiederholung*, was zu unterschiedlichen Levelsets der Größe 4 in den Phasen a) und b) je Wiederholung führt.

CoinRun selbst ist als Spiel in Anlehnung an klassische Plattform Jump&Runs gestaltet. Die Level werden prozedural generiert und mit statischen und beweglichen Hindernissen versehen. Ziel des Agenten ist, es die einzige Münze am rechten Ende des Levels einzusammeln, während Gegnern und Abgründen auf dem Weg dorthin ausgewichen werden muss. Der Aktionsraum des Agenten ist diskret mit einer Größe von 15 modelliert. Das einzige Rewardsignal in der Umgebung in Höhe von $10,0$ kann durch Einsammeln der Münze erhalten werden. Somit beträgt der maximal erreichbare Return einer Episode in CoinRun ebenfalls $10,0$. Ein einzelner Zustand der Umgebung ist realisiert als Farbbild bestehend aus 4096 Pixel. Diese stehen dem Agenten in Form eines 64x64x3 großen Vektors x_i zur Verfügung. Die Dimensionalität der Zustandsmenge beträgt somit $\mathbf{X} = [x_1, x_2, \ldots, x_n] \in \mathbb{R}^{12288 \times n}$.

Beispielzustände verschiedener Level während der unterschiedlichen Trainings- und Testphasen sind in Abbildung 5.3 abgebildet. Einzelne Level unterscheiden sich durch die prozedurale Generierung visuell durch variierende Farbschemata und Hintergründe, als auch logisch durch variierende Anzahl und Positionen der Plattformen und Hindernisse.

Der in CoinRun enthaltene Levelgenerator kann über verschiedene Parameter konfiguriert werden. Dies ermöglicht unter anderem eine Beschränkung der Anzahl unterschiedlicher Level (Parameter m der Benchmarking Pipeline), die während der *Policy Training* und In-Distribution *Policy Runs* generiert werden. Ebenso ist es möglich, den Zufalls Seed am Startzeitpunkt der Umgebung

zu konfigurieren, so dass durch Verwendung unterschiedlicher Seeds, in den *Prozesswiederholungen* n verschiedene Levelsets entstehen.

5.7.2 Algorithmen und Parametrisierung

Im Folgenden werden die zur Evaluation verwendeten Algorithmen sowie deren gewählte Parametrisierung vorgestellt.
Als RL-Algorithmus kommt Proximal Policy Optimization [123] in der für GPUs optimierten Variante PPO2 [33] zum Training der Agenten zum Einsatz. Die Entscheidung für diesen Algorithmus sowie die konkrete Implementierung wurde getroffen, da verwandte Arbeiten [27, 28] mit diesen gute Lernerfolge auf der CoinRun Umgebung erzielen konnten. Bei der Wahl des verwendeten tiefen Neuronalen Netzes wurde ebenfalls den Erfahrungen der verwandten Arbeiten gefolgt. Zum Einsatz kommt ein Netz der IMPALA Architektur [38], die speziell zum effizienten Training von Actor-Critic Agenten entwickelt wurde. Die gewählte IMPALA Variante entspricht der großen Netzversion aus [38], bestehend aus 15 Convolutional Layern und insgesamt $1,6$ Millionen Parametern.
Das Policy Training erfolgt über 25×10^5 Zeitschritte, parallel ausgeführt in 64 Instantiierungen der Simulationsumgebung. Innerhalb dieser Zeitschritte führt PPO2 ein Update der Policy durch, wenn $64 * 256$ neue Zustände aus den Instantiierungen gepuffert wurden.
Um verschiedene PEOC Varianten konstruieren zu können, wird nach dem ersten, sowie nach dem letzten Update der Policy eine Kopie gespeichert. Basierend auf diesen Policy Kopien erfolgt die Konstruktion von zwei getrennten PEOC Klassifikatoren, die *PEOC-1* beziehungsweise *PEOC-150* genannt werden.
Die in Abschnitt 5.6 vorgestellte Benchmarking Pipeline wird wie folgt parametrisiert: Die gesamte Pipeline wird insgesamt in 40 *Prozesswiederholungen* durchlaufen, um zuverlässige Qualitätsaussagen treffen zu können. Der Levelgenerator wird in jedem Durchlauf mit einem unterschiedlichen Zufalls Seed initialisiert. Zur Beschränkung der Levelanzahl während der *Policy Training* und *IND Runs* wird der Parameter $m = 4$ gesetzt. Als Regel für den *Performance Check* der trainierten Policies wird festgelegt, dass der durchschnittliche Return am Ende der Trainingsprozesse gegen den maximal möglichen Wert von 10 konvergieren muss. Konvergenz gilt als erreicht, wenn der Mittelwert der Returns der letzten 5 Policy-Updates größer $9,80$ ist. Folglich werden alle weiteren Schritte der Pipeline nur mehr basierend auf den Policies durchlaufen, die den *Performance Check* passieren. Mit jeder dieser Policies werden anschließend $\sim 30 \times 10^3$ Simulationsschritte im Rahmen der *IND Runs* auf den Levelsets der Größe 4, sowie $\sim 10 \times 10^3$ Schritte im Rahmen der *OOD Runs* auf unbeschränkter Levelsetgröße durchgeführt.
Zur Umsetzung des fairen Vergleichs mit klassischen Out-of-Distribution Klassifikatoren wird die Menge der In-Distribution Daten $\mathbb{I}$ in Trainings- und Test-

Daten aufgeteilt. Dabei werden $\frac{2}{3}$ der Daten zufällig dem Trainingsset zugewiesen, die verbleibenden $\frac{1}{3}$ dem Testset. Folglich stehen im Training zur Optimierung der Benchmarking Klassifikatoren $\sim 20 \times 10^3$ In-Distribution Zustandsbeobachtungen zur Verfügung. Das Testset wird, wie in Abschnitt 5.6 beschrieben, mit den Daten aus $\mathbb{O}$ vereinigt, so dass ein finales Testset der Größe $\sim 20 \times 10^3$ bei einer Gleichverteilung von IND- und OOD-Daten entsteht.
Eine Auflistung der verwendeten RL-Parameter sowie die verwendete Konfiguration der Benchmarking Pipeline sind in Tabelle 5.1 sowie 5.2 aufgeführt.
Zum Vergleich der Klassifikationsqualität des vorgeschlagenen PEOC Verfahrens werden drei klassische, nicht Policy-basierte State-of-the-Art Klassifikatoren evaluiert. Diese werden in jeder Prozesswiederholung auf dem Trainingsset der In-Distribution Daten optimiert. Die Evaluation erfolgt auf den identischen Daten, auf denen auch die Policy-basierten Klassifikatoren ausgewertet werden: dem Testset bestehend aus In- und Out-of-Distribution Daten. Die verwendeten klassischen Klassifikatoren werden im Folgenden kurz vorgestellt.
Der verwendete Autoencoder-basierte binäre Klassifikator folgt dem von Aggarwal [2] vorgestellten Ansatz. Zum Einsatz kommt ein tiefes Neuronales Netz mit vier versteckten Schichten der Größe $[64, 32, 32, 64]$. Als Klassifikationsscore dient der Rekonstruktionsfehler bei Vorhersage eines Datenpunktes.
Die beiden weiteren verwendeten binären Klassifikatoren *Single-Objective Generative Adversarial Active Learning* (SO-GAAL) und *Multiple-Objective Generative Adversarial Active Learning* (MO-GAAL) wurden von Liu et al. [88] basierend auf einem generativen Ansatz entwickelt. Grundidee ist die automatische Generierung von informativen, abweichenden (Out-of-Distribution) Daten während des Klassifikatortrainings. Die Hoffnung ist durch diese generierten Daten eine optimale Entscheidungsgrenze zwischen In- und Out-of-Distribution Daten finden zu können. *MO-GAAL* erweitert den Ansatz von *SO-GAAL* dabei um die Verwendung mehrerer Datengeneratoren mit unterschiedlichen Zielvorgaben. Im Rahmen der hier vorgestellten Evaluationen werden dabei zehn Generatoren eingesetzt.
Bei der Umsetzung aller drei Algorithmen *Autoencoder*, *SO-GAAL* und *MO-GAAL* wird auf die Implementierung von Zhao, Nasrullah und Li [149] zurückgegriffen.

Lernalgorithmus	PPO2
Neuronales Netz Architektur	IMPALA Large
Anzahl Convolutional Layer	15
Absolute Anzahl Parameter	1.6×10^6
Parallele Umgebungen	64
Policy Update Puffergröße	$64 * 256 = 16384$
Policy Kopien	Update 1 und Update 150

Tabelle 5.1: Reinforcement Learning Konfiguration

Prozesswiederholungen	$n = 40$
Beschränkung der Levelanzahl	$m = 4$
Performance Check Bedingung	Durchschnitt der Returns der letzten 5 Policy-Updates > 9.80

Tabelle 5.2: Benchmarking Pipeline Parameter

5.8 Evaluationsergebnisse

Im Folgenden werden die durchgeführten Experimente sowie die Zwischenresultate der Durchläufe der Trainingspipeline beschrieben.
Wie in Unterabschnitt 5.7.2 dargestellt, wurden insgesamt 40 *Prozesswiederholungen* durchgeführt. Von den so trainierten 40 Policies erfüllten acht die strikte Qualitätsanforderung im *Performance Check*. Abbildung 5.4 links zeigt in dunkelblau den mittleren Return μ dieser acht Policies (Y-Achse) über den Trainingsverlauf von 150 Policy-Updates (X-Achse). Die hellblaue Region visualisiert dabei die Standardabweichung σ. Es ist zu sehen, dass die Policies über den Trainingsverlauf von 150 Policy-Updates (25×10^5 Zeitschritte der Simulationsumgebung) gegen den in der Umgebung maximal erreichbaren Wert von 10 konvergieren. Es kann festgestellt werden, dass die Optimierung über den Trainingsverlauf zunehmend erfolgreich war und zu nahezu optimalen Policies geführt hat. An dieser Stelle soll noch einmal darauf hingewiesen werden, dass die 32 Policies, die den *Performance Check* am Ende des Trainings nicht erfüllt haben, nicht weiter ausgewertet wurden, und nicht in den Abbildungen aufgeführt sind.
Abbildung 5.4 rechts zeigt in dunkelblau die mittlere relative Entropie μ dieser acht Policies (Y-Achse) über den Trainingsverlauf von 150 Policy-Updates (X-Achse). Hier ist zu erkennen, dass die relative Entropie der Policies bei einem maximal möglichen Wert von 1 beginnt. Die Policies starten in einem Zustand, in dem alle Aktionen mit gleicher Wahrscheinlichkeit ausgeführt werden, sie sind somit maximal zufällig. Über den Trainingsverlauf von 150 Policy-Updates nimmt die relative Entropie dann jedoch zügig ab. Die Annahme, dass in der evaluierten Umgebung optimales Verhalten nicht-zufällige Aktionen benötigt, hat sich somit bestätigt. Die Entropie der Policy muss mit fortschreitenden Policy-Updates sinken, um den erwarteten zukünftigen Return zu maximieren. Des Weiteren ist zu erkennen, dass die Entropie am Ende des Trainings noch nicht konvergiert ist. Es ist zu vermuten, dass fortgeführtes Training zu einer weiteren Reduktion der Entropie führt. Dies wurde jedoch im Rahmen der durchgeführten Untersuchungen nicht evaluiert, da bereits bei 150 Policy-Updates optimale Policies im Hinblick auf den erreichten Return gelernt wurden.
Ob der Abfall der Entropie für im Training verwendete Zustände stärker ist, als für nicht gesehene Zustände, kann an dieser Stelle noch nicht beantwortet

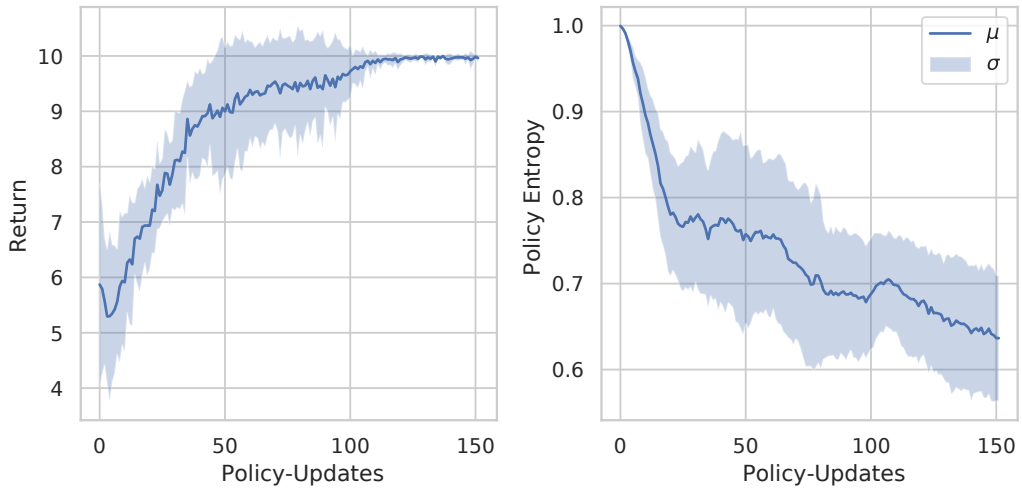


Abbildung 5.4: Return und Policy Entropy über den Trainingsverlauf der acht Policies, die den *Performance Check* erfüllt haben. Durchgezogene Linien stellen den Mittelwert μ dar, eingefärbte Regionen die Standardabweichung σ.

werden. Um hier eine Aussage treffen zu können, sind die folgenden Schritte der Benchmarking Pipeline zur Out-of-Distribution Evaluation notwendig.

In den auf das Policy-Training und den *Performance Check* folgenden *Policy Runs* wurden im Rahmen der *IND Runs* auf den Levelsets der Größe 4 je Policy $\sim 30 \times 10^3$ Simulationsschritte durchgeführt. Die hier angetroffenen Zustände s_i wurden jeweils als Menge $\mathbb{I}$ der In-Distribution Daten gesammelt. Im Rahmen der *OOD Runs* bildeten die Zustände s_o der je Policy durchgeführten $\sim 10 \times 10^3$ Schritte auf unbeschränkter Levelsetgröße die Menge $\mathbb{O}$ der Out-of-Distribution Daten.

Die Optimierung der drei Benchmarking Klassifikator-Algorithmen auf dem Trainingsset der In-Distribution-Daten verlief trotz extrem hoher Dimensionalität der Zustandsbeschreibungen ($x_i \in \mathbb{R}^{12288}$) erfolgreich, so dass die Evaluation aller Klassifikatoren auf dem finalen Testset der Größe $\sim 20 \times 10^3$ durchgeführt werden konnte.

An dieser Stelle sei noch einmal daran erinnert, dass der verwendete Prozess der Benchmarking Pipeline zu einem gleichverteilten Testset von In- und Out-of-Distribution Daten führt, so dass Visualisierungsansätze wie Precision-Recall Graphen keine Vorteile gegenüber ROC Graphen besitzen. Die Auswertung der Klassifikationsqualität der verschiedenen Klassifikatoren erfolgte somit wie in Abschnitt 5.3 erläutert anhand von ROC Graphen.

Die ROC Graphen der acht Evaluationsdurchläufe sind in Abbildung 5.5 dargestellt. Auf der Y-Achse jedes Graphen ist die Richtig-positiv-Rate (tpr) des jeweiligen Klassifikators abgebildet, sowie die Falsch-positiv-Rate (fpr) auf der X-Achse. Da alle evaluierten Klassifikatoren Scoring Klassifikatoren mit

Schwellwertparameter sind, entsteht durch Variation des Schwellwertes eine Kurve im ROC Raum. Die durchgezogenen Linien visualisieren die ROC Kurven der fünf evaluierten Klassifikatoren: Die neu entwickelten Policy-basierten Klassifikatoren *PEOC-1* sowie *PEOC-150*, sowie zum Vergleich die State-of-the-Art Klassifikatoren *SO-GAAL*, *MO-GAAL*, sowie den Autoencoder-basierten Ansatz *Autoenc*. Die blau gestrichelte diagonale Linie stellt den Referenzwert eines Zufallsklassifikators dar. Die Unterschiede zwischen den acht Durchläufen sind den Zufälligkeiten in der Simulationsumgebung, sowie der Stochastik in den Lernalgorithmen geschuldet. Durch diese Zufälligkeiten stehen den Klassifikatoren je Durchlauf unterschiedliche In- und Out-of-Distribution Sets zur Verfügung.

Allgemein ist festzustellen, dass die ROC Kurven der Klassifikatoren in einem Großteil der Fläche des ROC Graphen über der Diagonale des Zufallsklassifikators liegen. Sie sind somit in der Lage, sinnvolle Entscheidungen zu treffen und die Situationen besser als zufällig korrekt als In- oder Out-of-Distribution zu klassifizieren. Es ist jedoch keiner der Klassifikationsansätze in der Lage perfekte Klassifikationsentscheidungen ($\text{tpr} = 1.0, \text{fpr} = 0.0, \text{AUC} = 1.0$) zu treffen. Dies ist daran zu erkennen, dass keine der Kurven den optimalen Punkt links oben im Graphen ($tpr = 1.0, fpr = 0.0$) erreicht.

Über die Prozesswiederholungen hinweg fällt es schwer, eine allgemeine Aussage zum relativen Verhalten der einzelnen ROC Kurven zueinander zu treffen. Je nach gewähltem Schwellwert erreichen unterschiedliche Klassifikatoren das bessere Ergebnis, was dazu führt, dass sich die einzelnen ROC Kurven überschneiden. Nur in den Wiederholungen Abbildung 5.5e und Abbildung 5.5g liegt die Kurve eines einzelnen Klassifikators (*PEOC-1*), über den Großteil der Fläche des ROC Graphen, über der anderer Klassifikatoren. Des Weiteren ist ein Teils senkrechter Anstieg der ROC Kurve des *Autoenc*-basierten Klassifikators auffällig, wie beispielsweise in Abbildung 5.5b zu sehen ist. In diesem Fall war der Klassifikator bei entsprechend gesetztem Schwellwert in der Lage, 100% der In-Distribution Situationen korrekt als solche zu klassifizieren, während gleichzeitig 50% der Out-of-Distribution Situationen korrekt erkannt wurden. Ein solches Verhalten kann gewünscht beziehungsweise akzeptabel sein, in Anwendungsfällen, in denen das Auftreten von Out-of-Distribution Situationen nicht extrem kritisch ist, aber ein fälschlich ausgelöster Alarm sehr teuer ist.

Wie in Abschnitt 5.3 erläutert, kann die Fläche unterhalb einer ROC Kurve, die sogenannte AUC als Qualitätsmaß zum Vergleich mehrerer schwellwertabhängiger Scoring Klassifikatoren verwendet werden. Bezüglich dieser Metrik zeigen sich Unterschiede der einzelnen Klassifikatoren. Vergleicht man die erzielten AUC Werte der einzelnen Klassifikatoren, so zeigt sich, dass *PEOC-1* mit Ausnahme einer Prozesswiederholung (Abbildung 5.5f) die höchsten Werte erzielt. Diese liegen zwischen $0,6858$ und $0,7844$. Trotzdem gibt es Ausnahmen, beispielsweise in Prozesswiederholung Abbildung 5.5f, in der die beiden generativen Klassifikatoren *SO-GAAL* und *MO-GAAL* die höchsten AUC Werte mit

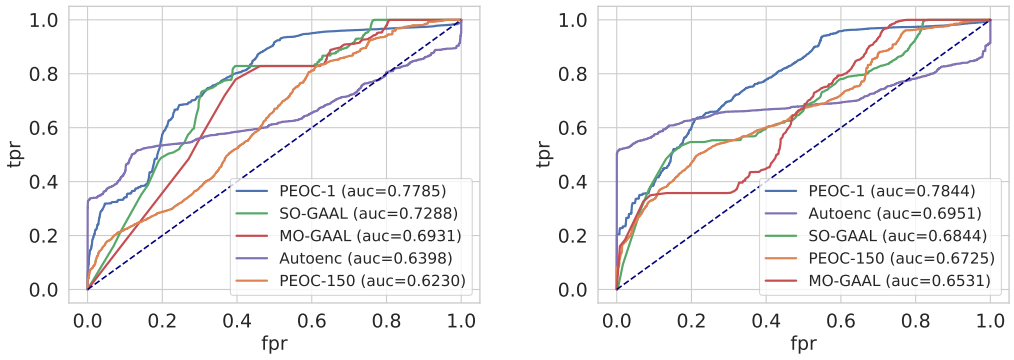
0,7697 und 0,7853 erzielen. Der Policy-basierte Klassifikator *PEOC-150*, der auf dem letzten Policy Update basiert, erreicht vergleichsweise geringe AUC Werte. Die Klassifikationsqualität ist mit AUC Werten zwischen 0,5155 und 0,7605 tendenziell unzuverlässig.
Im Folgenden wird nun die Klassifikationsqualität über alle acht Prozesswiederholungen hinweg analysiert. Hierfür werden Kenngrößen der Verteilung der AUC Werte berechnet und in Form eines Box-Plots (Abbildung 5.6) je Klassifikator dargestellt. Der Median der Werte (50% Quantil, Q_2) wird dabei durch den Strich innerhalb einer Box visualisiert. Die untere Grenze der Box entspricht dem 25% Quantil (Q_1), die obere Grenze dem 75% Quantil (Q_3) der Daten. Durch Berechnung der Distanz zwischen Q_1 und Q_3 kann der Interquantilabstand berechnet werden:

$$IQR = Q_3 - Q_1 = q_n(0,75) - q_n(0,25)$$

Über der Box grenzt eine sogenannte Antenne an. Diese zeigt die Distanz von Q_3 bis zum größten beobachteten Datenpunkt innerhalb eines 1,5-fachen IQR an. Ebenso zeigt die untere Antenne die Distanz von Q_1 bis zum kleinsten beobachteten Datenpunkt innerhalb eines 1,5-fachen IQR an.
Bereits auf den ersten Blick ist in Abbildung 5.6 ersichtlich, dass *PEOC-1* (blaue Box) den anderen Klassifikatoren deutlich überlegen ist. Die dargestellte Box des Klassifikators liegt auf nahezu der gesamten Fläche über den Boxen der anderen Klassifikatoren. Lediglich das unteren Ende der Box (Q_1) von *PEOC-1* überlappt leicht mit dem oberen Ende der Box (Q_3) von *SO-GAAL*. Auch im Median (Strich innerhalb einer Box) übertrifft *PEOC-1* die anderen Klassifikatoren mit einem AUC von 0,7390 deutlich. Zudem ist auffällig, dass die AUC Werte konzentrierter sind, als bei den anderen Klassifikatoren: Die Antennen des Box-Plots von *PEOC-1* liegen sehr nah am oberen beziehungsweise unteren Ende der Box.
PEOC-150 dagegen zeigt einen deutlich niedrigeren Median AUC Wert von 0,6262. Auch erstrecken sich die Antennen über nahezu die gesamte Y-Achse der Abbildung, die Datenverteilung weist somit eine deutlich höhere Streuung auf. Der zweitbeste Klassifikator nach dieser Metrik, mit deutlichem Abstand zu *PEOC-1*, ist *SO-GAAL*, mit einem Median AUC von 0,6928, gefolgt von *MO-GAAL* mit 0,6550 und *Autoenc* mit 0,6471.
Im Worst-Case-Verhalten zeigt *PEOC-1* erneut das beste Ergebnis, mit einem Minimum AUC von 0,6858. Auch hier zeigt sich ein deutlicher Abstand zur zweiten Position, *MO-GAAL* mit 0,5170.
Alle Kennzahlen der AUC Verteilungen sind in Tabelle 5.3 aufgeführt.

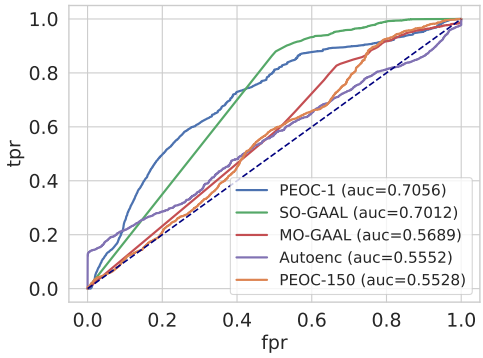
5.9 Zusammenfassung

In diesem Kapitel wurde ein neuartiger Ansatz zur Policy-basierten Erkennung von Out-of-Distribution Situationen vorgestellt. Das entwickelte PEOC Kon-

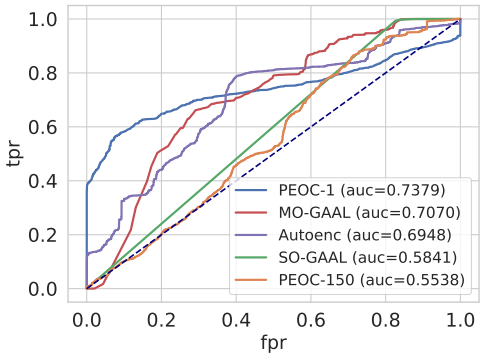


(a)

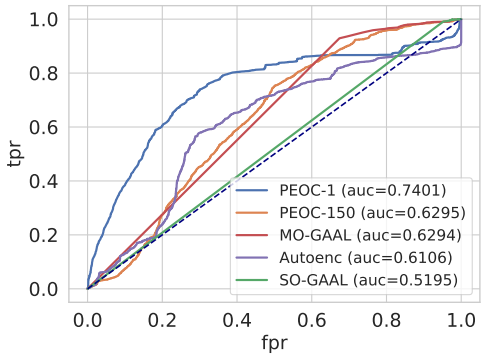
(b)



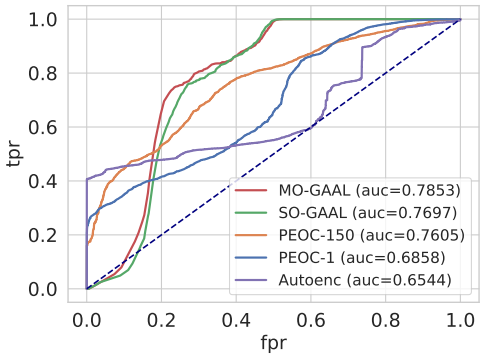
(c)



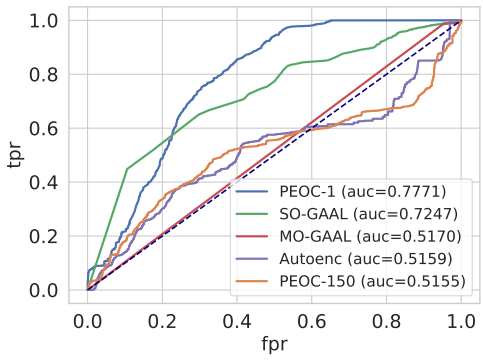
(d)



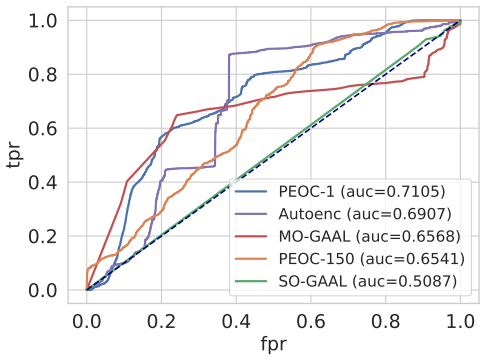
(e)



(f)



(g)



(h)

Abbildung 5.5: Klassifikator Verhalten dargestellt in Form von ROC Plots. Jeder Plot a)-h) zeigt eine einzelne Prozesswiederholung.

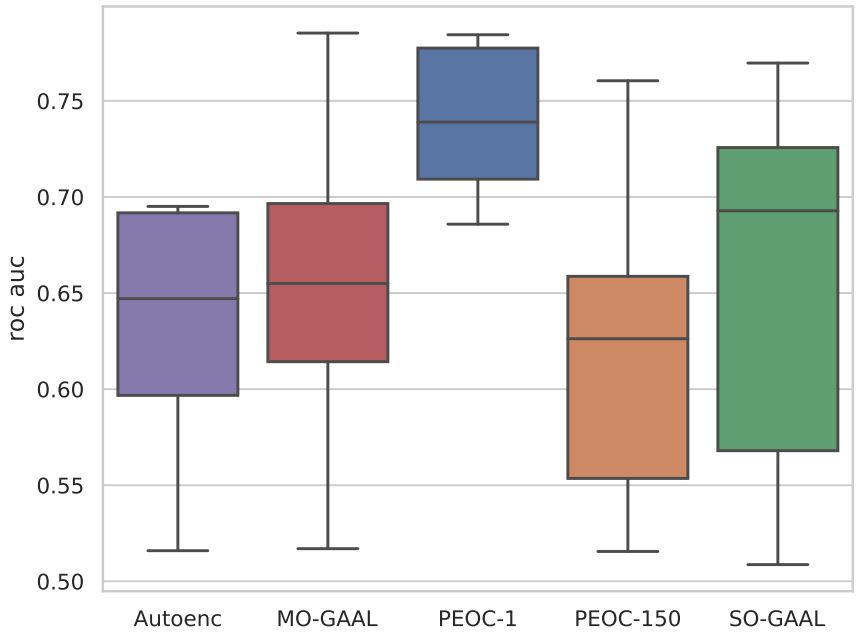


Abbildung 5.6: Vergleich des Verhaltens der Klassifikatoren basierend auf den ROC AUC Werten. Dargestellt sind die Ergebnisse von acht Prozesswiederholungen. Der Median der Datenverteilung ist durch den Strich innerhalb der Box dargestellt.

zept eignet sich speziell zur Kombination mit Policy-basierten RL-Ansätzen. Ist die Voraussetzung einer stochastisch modellierten Policy gegeben, kann das PEOC Verfahren bei verschiedenen State-of-the-Art RL-Algorithmen wie PPO oder A3C eingesetzt werden. Kernelement des PEOC Verfahrens ist die Annahme, dass der Abfall der Entropie der Policy für im Training verwendete Zustände stärker ist, als für nicht gesehene Zustände. Durch die Modellierung der OOD Erkennung als One-Class Klassifikationsproblem werden zur Trainingszeit keine OOD Datenpunkte benötigt. Dies erleichtert einen Einsatz in Echtweltszenarien, in denen die Erzeugung von In-Distribution Daten meist

Klassifikator	**Median**	$\boldsymbol{Q_1}$	$\boldsymbol{Q_3}$	**Maximum**	**Minimum**
PEOC-1	0,7390	0,7093	0,7775	0,7844	0,6858
PEOC-150	0,6262	0,5535	0,6587	0,7605	0,5155
Autoenc	0,6471	0,5968	0,6917	0,6951	0,5159
SO-GAAL	0,6928	0,5680	0,7257	0,7697	0,5087
MO-GAAL	0,6550	0,6143	0,6966	0,7853	0,5170

Tabelle 5.3: Kennzahlen der ROC AUC Verteilungen der acht Prozesswiederholungen je Klassifikator.

deutlich einfacher und kostengünstiger ist als die Erzeugung von OOD Daten. Durch die Entwicklung einer generischen Benchmarking Pipeline für OOD Klassifikation im Reinforcement Learning konnte ein allgemeingültiger Prozess geschaffen werden, der es ermöglicht, verschiedene OOD Klassifikationsansätze strukturiert und fair zu vergleichen. Dies ist von hoher Relevanz, da die unterschiedliche Funktionsweise von klassischen und Policy-basierten OOD Klassifikatoren eine systematische Herangehensweise benötigt, um Fehlerquellen zu minimieren. Die Möglichkeit der Durchführung von mehreren Prozesswiederholungen erlaubt es, die Varianz in der Klassifikationsperformance zu quantifizieren. Dies ist notwendig, da die meisten aktuellen OOD Klassifikationsverfahren, wie auch PEOC, stochastische Elemente enthalten. Die Berechnung zentraler Momente der so geschaffenen ROC AUC Verteilungen verschiedener Klassifikatoren ermöglicht schließlich einen aussagekräftigen Performancevergleich.
Die Evaluationsergebnisse zeigen, dass PEOC ein vielversprechender Weg zur Erkennung von OOD Situationen ist. Die Hypothese zum unterschiedlichen Abfall der Entropy von In- und OOD Daten konnte in den durchgeführten Evaluationen belegt werden. Somit ist es möglich, die Policy Entropy zielführend als Score eines binären Klassifikators zur OOD Erkennung einzusetzen. Im Vergleich mit alternativen Ansätzen zur One-Class OOD Klassifikation zeigte sich, dass die Variante *PEOC-1* den anderen Klassifikatoren deutlich überlegen ist. Mit einem AUC Wert von $0,7390$ im Median übertraf *PEOC-1* den nächstbesten Klassifikator *SO-GAAL* mit einem Median AUC von $0,6928$ deutlich. Zudem wiesen die AUC Werte über die Evaluationswiederholungen hinweg die geringste Abweichung vom Median auf, die Performance war somit die zuverlässigste aller evaluierten Klassifikatoren. Dennoch ist festzuhalten, dass keiner der evaluierten Klassifikatoren ein perfektes Klassifikationsergebnis über beliebige Schwellwerte ermöglicht. Folglich ist, basierend auf dem jeweiligen Anwendungsfall, stets eine Abwägung zu treffen, ob eine höhere Falsch-positiv-Rate (fpr) in Kauf genommen werden kann, um so eine höhere Richtig-positiv-Rate (tpr) zu erreichen.

6 Zusammenfassung und Ausblick

Der Fokus der vorliegenden Arbeit war die Entwicklung von Modellierungsansätzen von Unsicherheit und Wahrnehmung, um so eine Grundlage für zuverlässige, sichere lernende Systeme zu schaffen. Dies ist aktuell von besonderer Relevanz, da die enormen Fortschritte im Bereich der künstlichen Intelligenz (KI) dazu führen, dass die entsprechenden Systeme den Alltag von immer mehr Menschen erreichen. So sind beispielsweise erste selbstfahrende Autos in den Städten zu beobachten, oder autonome Lieferroboter, die sich Bürgersteige mit Passanten teilen. Durch dieses Eintreten der lernenden Systeme in die physische Umwelt der Menschen müssen Fragen der Zuverlässigkeit und Sicherheit in den Fokus rücken. Dies ist insbesondere für industrielle Anwendungen wie Smart Factories oder Smart Grids von hoher Relevanz, da hier meist eine geringe Toleranz für Fehler existiert. Es werden somit Ansätze und Lösungen benötigt, die die Grenzen dieser lernenden Systeme erfassen können, um so mögliches Fehlverhalten zu verhindern. Die vorliegende Arbeit konnte hier durch die Entwicklung von Ansätzen der Unsicherheits- und Wahrnehmungsmodellierung einen grundlegenden Beitrag leisten. Die wichtigsten Ergebnisse werden im Folgenden noch einmal knapp zusammengefasst.

Der erste vorgestellte Ansatz hatte die Modellierung eines räumlichen Wahrnehmungsverständnisses für lernende Systeme zum Ziel. Hierfür konnte zunächst gezeigt werden, wie eine Isovisten-basierte Quantifizierung der Wahrnehmung als Datengrundlage für Machine Learning (ML) basierte Verfahren nutzbar gemacht werden kann. Diese Daten bilden den Input für eine ML-basierte Modellierung. Um den Ansatz zu evaluieren, wurde ein zu diesem Zweck entwickeltes Framework vorgestellt, das die Generierung von Isovistenmessungen entlang geospatialer Trajektorien innerhalb einer 3D-Simulationsumgebung ermöglicht. Im Rahmen der durchgeführten Evaluation konnte anschließend gezeigt werden, dass die erfassten Isovistenmessungen entlang der Trajektorien die wiederkehrenden Strukturen innerhalb der Gebäude widerspiegeln. Mittels einer qualitativen visuellen Analyse sowohl im Karten- als auch Feature-Raum konnte bestätigt werden, dass die verwendeten Unsupervised ML-Methoden erfolgreich semantisch sinnvolle Cluster modellieren. Die so realisierte Isovisten-basierte Wahrnehmungsmodellierung wurde daraufhin mittels probabilistischer Modellierung erweitert. Die Kombination mit Bayesian Surprise ermöglicht die Nutzung der in der Wahrnehmung vorhandenen Unsicherheit als zusätzliche Informationsquelle. Bayesian Surprise

modelliert dabei die subjektive Sicht eines Agenten bezüglich der zu erwartenden Wahrnehmungen durch bedingte Wahrscheinlichkeitsverteilungen derselben. Im Rahmen der Evaluation konnte gezeigt werden, dass der Einsatz der Dirichlet Verteilung als Conjugate Prior zur Multinomialverteilung eine effiziente und dennoch exakte Bayessche Posteriorberechnung bei Verwendung von Isovistendaten erlaubt. Dies ermöglicht unter anderem eine Zusammenfassung und Charakterisierung von Trajektorien.

Im zweiten im Rahmen der Arbeit vorgestellten Ansatz stand die Modellierung von Unsicherheit in Value-basiertem Deep Reinforcement Learning (RL) im Vordergrund. RL-basierte Problemformulierungen sind aktuell dominant zur Lösung sequentieller Entscheidungsprobleme. In die dort meist Deep Learning basierten Architekturen lässt sich die im vorherigen Kapitel verwendete exakte Bayessche Inferenz jedoch nicht trivial übertragen. Die Modellierung von Unsicherheit ist allerdings gerade hier von zentraler Relevanz, da es in den meisten Problemstellungen nicht möglich ist sicherzustellen, dass alle möglichen Situationen, in denen sich das System befinden kann, auch im Training berücksichtigt wurden. Um hier einen Beitrag zu leisten, wurden neue Verfahren zur Unsicherheitsmodellierung und Out-of-Distribution (OOD) Erkennung in sequentiellen Entscheidungsproblemen vorgestellt. Der entwickelte Ansatz UBOOD (Uncertainty Based Out-of-Distribution) basiert auf einer Formulierung des OOD Problems als One-Class Klassifikationsproblem. Die Tatsache, dass zur Trainingszeit keine OOD Datenpunkte benötigt werden, erleichtert einen Einsatz in Echtweltszenarien, in denen die Erzeugung von In-Distribution Daten meist deutlich einfacher und kostengünstiger ist, als die Erzeugung von OOD Daten. Zur Umsetzung wurden zunächst existierende Architekturen des probabilistischen ML aus dem Bereich der approximativen Bayesschen Inferenzmethoden sowie von Ensemblemethoden für die Unsicherheitsmodellierung in RL-Systemen geeignet angepasst. Die Evaluation erfolgte auf zwei speziell entwickelten RL-Umgebungen, die systematische Modifikationen nach dem Trainingsprozess ermöglichen. So konnten gezielt OOD Zustände während der Evaluation erzeugt und evaluiert werden. Die Evaluationsergebnisse zeigten, dass die Nutzung der epistemischen Unsicherheit der Value-Funktion eines RL-Agenten zur Erkennung von OOD Situationen einen gangbaren Weg darstellt. Die Güte der OOD Klassifikation ist jedoch direkt von der Qualität der Unsicherheitsmodellierung abhängig. Hier lieferten Ensemble-basierte Architekturen konstant bessere Ergebnisse als Monte-Carlo Dropout basierte.

Im dritten im Rahmen dieser Arbeit vorgestellten Ansatz wurde eine neuartige Methode zur Policy-basierten Erkennung von OOD Situationen vorgestellt. Motivation hierfür ist, dass insbesondere im Bereich der Robotik State-of-the-Art Lösungen aktuell mittels Policy-basiertem RL erzielt werden [54, 68]. Auch hier ist es von zentraler Wichtigkeit, die Sicherheit und Zuverlässigkeit der Systeme zu gewährleisten, insbesondere wenn die Roboter in räumlicher Nähe zu Menschen agieren. Die im Rahmen des hier vorgestellten Ansatzes vorgeschlagene Lösung lautet, die stets existente Unsicherheit in den Aktionsentschei-

dungen zu beachten. Kernelement des vorgestellten, PEOC (Policy Entropy Out-of-Distribution Classifier) genannten Verfahrens, ist die Verwendung der Entropie der Policy eines Agenten als Score eines One-Class Klassifikators. Die Evaluationsergebnisse zeigten, dass PEOC ein vielversprechender Weg zur Erkennung von OOD Situationen ist. Somit ist es möglich, die Policy Entropy zielführend als Score eines binären Klassifikators zur OOD Erkennung einzusetzen. Im Vergleich mit aktuellen State-of-the-Art Ansätzen zur One-Class OOD Klassifikation zeigte sich zudem, dass PEOC den anderen Klassifikatoren deutlich überlegen war. Zudem wiesen die AUC Werte über die Evaluationswiederholungen hinweg die geringste Abweichung vom Median auf, die Performance war somit die zuverlässigste aller evaluierten Klassifikatoren.

Zusammengefasst betrachtet konnte mit den entwickelten Ansätzen das Ziel einer Unsicherheits- und Wahrnehmungsmodellierung für aktuelle lernende Systeme erreicht werden. Die entwickelte Isovisten-basierte Modellierung der Wahrnehmung ist in der Lage, nachvollziehbare, semantisch sinnvolle Cluster zu bilden. Die probabilistische Erweiterung der Modellierung mittels Bayesian Surprise eröffnet zudem den Weg, die subjektive Sicht eines Agenten bezüglich der zu erwartenden Wahrnehmungen zu nutzen. Es konnte gezeigt werden, dass in diesem Rahmen eine exakte Bayessche Posteriorberechnung, unter Verwendung von Conjugate Prior Verteilungen, möglich ist. Im Rahmen der Unsicherheitsmodellierung für Value-basierte RL-Ansätze konnten existierende Architekturen des probabilistischen ML erfolgreich für die Unsicherheitsmodellierung im RL-Kontext angepasst werden. Die entwickelten Methoden UBOOD und PEOC sind somit mit aktuellen Deep Learning Architekturen kombinierbar und auch für einen Einsatz in der Praxis geeignet. Die geschaffenen Unsicherheitsmodellierungen stellen eine Ausgangsbasis dar, um basierend darauf weitere Ansätze zu entwickeln, die die Präsenz von Unsicherheit berücksichtigen und so die Zuverlässigkeit und Sicherheit der Systeme erhöhen.

Trotz dieser positiven Ergebnisse muss festgehalten werden, dass keiner der entwickelten Modellierungsansätze frei von Schwächen ist. Beispielsweise vermied die im Kontext von Bayesian Surprise zur Modellierung verwendete, exakte Bayessche Inferenz zwar einerseits Approximationsfehler, doch dieser Vorteil wurde durch Schwächen überlagert, die eine Konsequenz des notwendigen Einsatzes einfacher Verteilungstypen waren. Im Gegenzug waren die auf tiefen Neuronalen Netzen aufbauenden, approximativen Bayesschen Ansätze und Ensemble-Techniken zwar in der Lage, komplexe Verteilungen zu modellieren, die vorhergesagten Unsicherheitswerte waren dann jedoch (besonders im Falle von MC Dropout) nicht immer zuverlässig. Folglich war weder der auf dieser Unsicherheit basierende Value-basierte Ansatz UBOOD, noch der Policy-basierte Ansatz PEOC in der Lage, ein perfektes OOD Klassifikationsergebnis zu liefern. Die Konsequenz ist, dass je nach Anwendungsfall stets abzuwägen ist, ob durch Anpassung des Klassifikationsschwellwertes eine höhere Falsch-positiv-Rate (fpr) in Kauf genommen werden kann, um so eine höhere Richtig-positiv-Rate (tpr) zu erreichen. Dies könnte beispielsweise in sicher-

heitskritischen Anwendungen angemessen sein, um die Wahrscheinlichkeit für nicht erkannte Fehler zu reduzieren.
Diese im vorherigen Abschnitt zusammengefassten Schwächen der entwickelten Ansätze zeigen bereits den vermutlich vielversprechendsten Weg für zukünftige Arbeiten auf: Die weitere Verbesserung der Qualität der Unsicherheitsmodellierung. Diese bildet die Grundlage und beeinflusst damit alle weiter darauf aufbauenden Ansätze, gleich ob es die Erkennung von OOD Situationen ist, die Identifikation von charakteristischen Stellen der Wahrnehmung, oder ein zukünftiger Einsatz zur Exploration im RL. Nur wenn es möglich ist, die tatsächlich präsente Unsicherheit zuverlässig zu modellieren, ist Verlass auf darauf aufbauende Anwendungen. Die Ergebnisse der vorliegenden Arbeit haben gezeigt, dass hier aktuell Abwägungen getroffen werden müssen. Exakte Bayessche Methoden sind approximativen Methoden nicht eindeutig vorzuziehen, wenn dies auf Kosten der Ausdrucksstärke der eingesetzten Modelle geht. Das zentrale Element ist also die Weiterentwicklung und Optimierung probabilistischer Modellierungsansätze. Diese haben das Potential, die Qualität einer enormen Bandbreite an Anwendungen im Rahmen lernender Systeme zu verbessern. Sie können einen zentralen Baustein der Vision bilden, lernende Systeme zu entwickeln, die sich ihrer Unsicherheiten bewusst sind und so zuverlässig und sicher im Umfeld von Menschen agieren.

Abbildungsverzeichnis

Literatur

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu und X. Zheng. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: `http://tensorflow.org/`.

[2] C. C. Aggarwal. "Outlier analysis". In: *Data mining*. Springer, 2015, S. 237–263. DOI: `10.1007/978-3-319-14142-88`.

[3] C. Ah-Soon und K. Tombre. "Variations on the analysis of architectural drawings". In: *Document Analysis and Recognition, 1997., Proceedings of the Fourth International Conference on*. Bd. 1. IEEE. Ulm, Germany, 1997, S. 347–351. DOI: `10.1109/ICDAR.1997.619869`.

[4] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas et al. "Solving rubik's cube with a robot hand". In: *arXiv preprint arXiv:1910.07113* (2019).

[5] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman und D. Mané. "Concrete problems in AI safety". In: *arXiv preprint arXiv:1606.06565* (2016).

[6] D. Anguelov, D. Koller, E. Parker und S. Thrun. "Detecting and modeling doors with mobile robots". In: *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*. Bd. 4. IEEE. New Orleans, LA, USA, 2004, S. 3777–3784. DOI: `10.1109/ROBOT.2004.1308857`.

[7] M. Balsamo, W. Knottenbelt und A. Marin. *Computer Performance Engineering: 10th European Workshop, EPEW 2013, Venice, Italy, September 16-17, 2013, Proceedings*. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013. ISBN: 9783642407253. DOI: `10.1007/978-3-642-40725-3`.

[8] M. Bayes und M. Price. "An Essay towards Solving a Problem in the Doctrine of Chances. By the Late Rev. Mr. Bayes, F. R. S. Communicated by Mr. Price, in a Letter to John Canton, A. M. F. R. S." In: *Philosophical Transactions (1683-1775)* 53 (1763), S. 370–418. ISSN: 02607085. DOI: 10.1098/rstl.1763.0053. URL: http://www.jstor.org/stable/105741.

[9] M. G. Bellemare, W. Dabney und R. Munos. "A Distributional Perspective on Reinforcement Learning". In: *Proceedings of the 34th International Conference on Machine Learning*. Hrsg. von D. Precup und Y. W. Teh. Bd. 70. Proceedings of Machine Learning Research. PMLR, 2017, S. 449–458.

[10] M. G. Bellemare, Y. Naddaf, J. Veness und M. Bowling. "The arcade learning environment: An evaluation platform for general agents". In: *Journal of Artificial Intelligence Research* 47 (2013), S. 253–279. DOI: 10.1613/jair.3912.

[11] R. Bellman, R. Corporation und K. M. R. Collection. *Dynamic Programming*. Rand Corporation research study. Princeton University Press, 1957. ISBN: 9780691079516.

[12] W. H. Beluch, T. Genewein, A. Nürnberger und J. M. Köhler. "The Power of Ensembles for Active Learning in Image Classification". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, S. 9368–9377. DOI: 10.1109/CVPR.2018.00976.

[13] E. Bender. *An Introduction to Mathematical Modeling*. Dover Books on Computer Science Series. Dover Publications, 2000. ISBN: 9780486411804.

[14] M. L. Benedikt. "To take hold of space: isovists and isovist fields". In: *Environment and Planning B: Planning and design* 6.1 (1979), S. 47–65. DOI: 10.1068/b060047.

[15] C. M. Bishop. "Model-based machine learning". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 371.1984 (2013), S. 20120222. DOI: 10.1098/rsta.2012.0222.

[16] C. M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN: 0387310738.

[17] D. M. Blei, A. Kucukelbir und J. D. McAuliffe. "Variational Inference: A Review for Statisticians". In: *Journal of the American Statistical Association* 112.518 (2017), S. 859–877. DOI: 10.1080/01621459.2017.1285773.

[18] S. E. Boehnke, D. J. Berg, R. A. Marino, P. F. Baldi, L. Itti und D. P. Munoz. "Visual adaptation and novelty responses in the superior colliculus". In: *European Journal of Neuroscience* 34.5 (2011), S. 766–779. DOI: 10.1111/j.1460-9568.2011.07805.x.

[19] G. E. Box und N. R. Draper. *Empirical model-building and response surfaces.* John Wiley & Sons, 1987. ISBN: 0-471-81033-9.

[20] K. Boyd, V. S. Costa, J. Davis und C. D. Page. "Unachievable region in precision-recall space and its effect on empirical evaluation". In: *Proceedings of the 29th International Conference on Machine Learning. International Conference on Machine Learning.* Bd. 2012. NIH Public Access. 2012, S. 349. URL: https://pubmed.ncbi.nlm.nih.gov/24350304.

[21] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang und W. Zaremba. *OpenAI Gym.* 2016. eprint: arXiv:1606.01540.

[22] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell et al. "Language models are few-shot learners". In: *arXiv preprint arXiv:2005.14165* (2020).

[23] P. Buschka und A. Saffiotti. "A virtual sensor for room detection". In: *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on.* Bd. 1. IEEE. Lausanne, Switzerland, 2002, S. 637–642. DOI: 10.1109/IRDS.2002.1041463.

[24] M. Carroll, R. Shah, M. K. Ho, T. Griffiths, S. Seshia, P. Abbeel und A. Dragan. "On the utility of learning about humans for human-ai coordination". In: *Advances in Neural Information Processing Systems* 32 (2019), S. 5174–5185.

[25] W. Chen, T. Qu, Y. Zhou, K. Weng, G. Wang und G. Fu. "Door recognition and deep learning algorithm for visual based robot navigation". In: *Robotics and Biomimetics (ROBIO), 2014 IEEE International Conference on.* IEEE. Bali, Indonesia: IEEE, 2014, S. 1793–1798. DOI: 10.1109/ROBIO.2014.7090595.

[26] D.-A. Clevert, T. Unterthiner und S. Hochreiter. "Fast and accurate deep network learning by exponential linear units (elus)". In: *arXiv preprint arXiv:1511.07289* (2015).

[27] K. Cobbe, C. Hesse, J. Hilton und J. Schulman. "Leveraging procedural generation to benchmark reinforcement learning". In: *International conference on machine learning.* PMLR. 2020, S. 2048–2056.

[28] K. Cobbe, O. Klimov, C. Hesse, T. Kim und J. Schulman. "Quantifying generalization in reinforcement learning". In: *International Conference on Machine Learning.* PMLR. 2019, S. 1282–1289.

[29] C. Cosma, M. Confente, M. Governo und R Fiorini. "An autonomous robot for indoor light logistics". In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*. Bd. 3. IEEE. 2004, S. 3003–3008. DOI: 10.1109/IROS.2004.1389866.

[30] D. R. Cox. *Principles of statistical inference*. Cambridge university press, 2006. DOI: 10.1017/CBO9780511813559.

[31] J. Davis und M. Goadrich. "The relationship between Precision-Recall and ROC curves". In: *Proceedings of the 23rd international conference on Machine learning*. 2006, S. 233–240. DOI: 10.1145/1143844.1143874.

[32] R. De Lemos, H. Giese, H. A. Müller, M. Shaw, J. Andersson, M. Litoiu, B. Schmerl, G. Tamura, N. M. Villegas, T. Vogel et al. "Software engineering for self-adaptive systems: A second research roadmap". In: *Software Engineering for Self-Adaptive Systems II*. Springer, 2013, S. 1–32. DOI: 10.1007/978-3-642-35813-5_1.

[33] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu und P. Zhokhov. *OpenAI Baselines*. https://github.com/openai/baselines. 2017.

[34] T. G. Dietterich. "Ensemble methods in machine learning". In: *International workshop on multiple classifier systems*. Springer. 2000, S. 1–15. DOI: 10.1007/3-540-45014-9_1.

[35] B. Efron. "Bootstrap Methods: Another Look at the Jackknife". In: *Breakthroughs in Statistics: Methodology and Distribution*. New York, NY: Springer New York, 1992, S. 569–593. ISBN: 978-1-4612-4380-9. DOI: 10.1007/978-1-4612-4380-9_41.

[36] W. Einhaeuser, T. N. Mundhenk, P. F. Baldi, C. Koch und L. Itti. "A bottom-up model of spatial attention predicts human error patterns in rapid scene recognition". In: *Journal of Vision* 7.10 (2007), S. 1–13. DOI: 10.1167/7.10.6.

[37] B. Emo. "Exploring isovists : the egocentric perspective". In: *International Space Syntax Symposium*. London, UK: Space Syntax Laboratory, 2015, S. 1–8.

[38] L. Espeholt, H. Soyer, R. Munos et al. "IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures". In: 2018.

[39] M. Ester, H.-P. Kriegel, J. Sander, X. Xu et al. "A density-based algorithm for discovering clusters in large spatial databases with noise." In: *Kdd*. Bd. 96. 34. 1996, S. 226–231.

[40] M. Ester und J. Sander. *Knowledge discovery in databases: Techniken und Anwendungen*. Springer-Verlag, 2013.

[41] B. Everitt und T. Hothorn. *An introduction to applied multivariate analysis with R*. New York, USA: Springer Science & Business Media, 2011. DOI: 10.1007/978-1-4419-9650-3.

[42] B. Eysenbach und S. Levine. "Maximum entropy rl (provably) solves some robust rl problems". In: *arXiv preprint arXiv:2103.06257* (2021).

[43] J. Farebrother, M. C. Machado und M. Bowling. "Generalization and Regularization in DQN". In: (2018). arXiv: 1810.00123 [cs.LG].

[44] T. Fawcett. "An introduction to ROC analysis". In: *Pattern recognition letters* 27.8 (2006), S. 861–874. DOI: 10.1016/j.patrec.2005.10.010.

[45] S. Feld, A. Sedlmeier, M. Friedrich, J. Franz und L. Belzner. "Bayesian Surprise in Indoor Environments". In: *27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL '19)*. ACM. 2019. DOI: 10.1145/3347146.3359358.

[46] S. Feld, M. Werner und C. Linnhoff-Popien. "Approximated Environment Features With Application to Trajectory Annotation". In: *6th IEEE Symposium Series on Computational Intelligence (IEEE SSCI 2016)*. IEEE. Athens, Greece: IEEE, 2016. DOI: 10.1109/SSCI.2016.7849993.

[47] M. Figliozzi und D. Jennings. "Autonomous delivery robots and their potential impacts on urban freight energy consumption and emissions". In: *Transportation research procedia* 46 (2020), S. 21–28. DOI: 10.1016/j.trpro.2020.03.159.

[48] S. Fort, J. Ren und B. Lakshminarayanan. "Exploring the Limits of Out-of-Distribution Detection". In: *arXiv preprint arXiv:2106.03004* (2021).

[49] T. Gabor, A. Sedlmeier, M. Kiermeier, T. Phan, M. Henrich, M. Picklmair, B. Kempter, C. Klein, H. Sauer, R. Schmid und J. Wieghardt. "Scenario Co-Evolution for Reinforcement Learning on a GridWorld Smart Factory Domain". In: *Genetic and Evolutionary Computation Conference (GECCO)*. ACM, 2019.

[50] Y. Gal und Z. Ghahramani. "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning". In: *international conference on machine learning*. 2016, S. 1050–1059.

[51] Y. Gal, J. Hron und A. Kendall. "Concrete Dropout". In: *Advances in Neural Information Processing Systems 30*. Hrsg. von I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan und R. Garnett. Curran Associates, Inc., 2017, S. 3581–3590.

[52] Z. Ghahramani. "Bayesian non-parametrics and the probabilistic approach to modelling". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 371.1984 (2013), S. 20110553. DOI: 10.1098/rsta.2011.0553.

[53] Z. Ghahramani. "Probabilistic machine learning and artificial intelligence". In: *Nature* 521.7553 (2015), S. 452–459. DOI: 10.1038/nature14541.

[54] S. Gilroy*, D. Lau*, L. Yang*, E. Izaguirre, K. Biermayer, A. Xiao, M. Sun, A. Agrawal, J. Zeng, Z. Li und K. Sreenath. "Autonomous Navigation for Quadrupedal Robots with Optimized Jumping through Constrained Obstacles". In: *IEEE International Conference on Automation Science and Engineering (CASE)*. Lyon, France, 2021. DOI: 10.1109/CASE49439.2021.9551524.

[55] X. Glorot, A. Bordes und Y. Bengio. "Deep sparse rectifier neural networks". In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop und Conference Proceedings. 2011, S. 315–323.

[56] N. Goerke und S. Braun. "Building semantic annotated maps by mobile robots". In: *Proceedings of the Conference Towards Autonomous Robotic Systems*. Londonderry, United Kingdom: University of Ulster, 2009, S. 149–156.

[57] I. Goodfellow, Y. Bengio und A. Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[58] D. Hafner, D. Tran, T. Lillicrap, A. Irpan und J. Davidson. "Noise contrastive priors for functional uncertainty". In: *Uncertainty in Artificial Intelligence*. PMLR. 2020, S. 905–914.

[59] S. C. Hayward und S. S. Franklin. "Perceived openness-enclosure of architectural space". In: *Environment and Behavior* 6.1 (1974), S. 37–52. DOI: 10.1177/001391657400600102.

[60] Y. He, T. N. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, R. Pang, Q. Liang, D. Bhatia, Y. Shangguan, B. Li, G. Pundak, K. C. Sim, T. Bagby, S.-y. Chang, K. Rao und A. Gruenstein. "Streaming End-to-end Speech Recognition for Mobile Devices". In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019, S. 6381–6385. DOI: 10.1109/ICASSP.2019.8682336.

[61] B. Hillier und J. Hanson. *The social logic of space*. Cambridge, UK: Cambridge university press, 1984. DOI: 10.1017/CBO9780511597237.

[62] J. J. Hull. "A database for handwritten text recognition research". In: *IEEE Transactions on pattern analysis and machine intelligence* 16.5 (1994), S. 550–554. DOI: 10.1109/34.291440.

[63] L. Itti und P. F. Baldi. "A Principled Approach to Detecting Surprising Events in Video". In: *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. San Siego, CA, 2005, S. 631–637. DOI: 10.1109/CVPR.2005.40.

[64] L. Itti und P. F. Baldi. "Modeling what attracts human gaze over dynamic natural scenes". In: *Computational Vision in Neural and Machine Systems*. Hrsg. von L. Harris und M. Jenkin. Cambridge, MA: Cambridge University Press, 2006.

[65] L. Itti und P. Baldi. "Bayesian Surprise Attracts Human Attention". In: *Advances in Neural Information Processing Systems*. Hrsg. von Y. Weiss, B. Schölkopf und J. Platt. Bd. 18. MIT Press, 2005.

[66] M. Jaderberg, W. M. Czarnecki, I. Dunning, L. Marris, G. Lever, A. G. Castañeda, C. Beattie, N. C. Rabinowitz, A. S. Morcos, A. Ruderman, N. Sonnerat, T. Green, L. Deason, J. Z. Leibo, D. Silver, D. Hassabis, K. Kavukcuoglu und T. Graepel. "Human-level performance in 3D multiplayer games with population-based reinforcement learning". In: *Science* 364.6443 (2019), S. 859–865. DOI: 10.1126/science.aau6249.

[67] S. James, P. Wohlhart, M. Kalakrishnan, D. Kalashnikov, A. Irpan, J. Ibarz, S. Levine, R. Hadsell und K. Bousmalis. "Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, S. 12627–12637. DOI: 10.1109/CVPR.2019.01291.

[68] Y. Ji, B. Zhang und K. Sreenath. "Reinforcement Learning for Collaborative Quadrupedal Manipulationof a Payload over Challenging Terrain". In: *IEEE International Conference on Automation Science and Engineering (CASE)*. Lyon, France, 2021. DOI: 10.1109/CASE49439.2021.9551481.

[69] Y. Jiao und P. Du. "Performance measures in evaluating machine learning based bioinformatics predictors for classifications". In: *Quantitative Biology* 4.4 (2016), S. 320–330. DOI: 10.1007/s40484-016-0081-2.

[70] M. Johnson, M. Schuster, Q. V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F. Viégas, M. Wattenberg, G. Corrado et al. "Google's multilingual neural machine translation system: Enabling zero-shot translation". In: *Transactions of the Association for Computational Linguistics* 5 (2017), S. 339–351.

[71] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko et al. "Highly accurate protein structure prediction with AlphaFold". In: *Nature* 596.7873 (2021), S. 583–589. DOI: 10.1038/s41586-021-03819-2.

[72] G. Kahn, A. Villaflor, V. Pong, P. Abbeel und S. Levine. "Uncertainty-aware reinforcement learning for collision avoidance". In: *arXiv preprint arXiv:1702.01182* (2017).

[73] B. Kalman und S. Kwasny. "Why tanh: choosing a sigmoidal function". In: *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*. Bd. 4. 1992, 578–581 vol.4. DOI: 10.1109/IJCNN.1992.227257.

[74] A. Kendall und Y. Gal. "What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?" In: *Advances in Neural Information Processing Systems 30*. Hrsg. von I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan und R. Garnett. Curran Associates, Inc., 2017, S. 5574–5584.

[75] Z. H. Khan, A. Siddique und C. W. Lee. "Robotics utilization for healthcare digitization in global COVID-19 management". In: *International journal of environmental research and public health* 17.11 (2020), S. 3819. DOI: 10.3390/ijerph17113819.

[76] V. V. Kharin und F. W. Zwiers. "On the ROC score of probability forecasts". In: *Journal of Climate* 16.24 (2003), S. 4145–4150. DOI: 10.1175/1520-0442(2003)016%3C4145:OTRSOP%3E2.0.CO;2.

[77] P. W. Koh, S. Sagawa, H. Marklund, S. M. Xie, M. Zhang, A. Balsubramani, W. Hu, M. Yasunaga, R. L. Phillips, I. Gao et al. "Wilds: A benchmark of in-the-wild distribution shifts". In: *International Conference on Machine Learning*. PMLR. 2021, S. 5637–5664.

[78] S. Kullback und R. A. Leibler. "On Information and Sufficiency". In: *The Annals of Mathematical Statistics* 22.1 (1951), S. 79–86. ISSN: 00034851. DOI: 10.1214/aoms/1177729694. URL: http://www.jstor.org/stable/2236703.

[79] A. Küpper. *Location-Based Services: Fundamentals and Operation*. Wiley, 2005. ISBN: 9780470092316. DOI: 10.1002/0470092335.

[80] B. Lakshminarayanan, A. Pritzel und C. Blundell. "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles". In: *Advances in Neural Information Processing Systems 30*. Hrsg. von I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan und R. Garnett. Curran Associates, Inc., 2017, S. 6402–6413.

[81] P. Laplace. *Essai Philosophique Sur Les Probabilités*. H. Remy, 1829.

[82] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard und L. D. Jackel. "Backpropagation applied to handwritten zip code recognition". In: *Neural computation* 1.4 (1989), S. 541–551. DOI: 10.1162/neco.1989.1.4.541.

[83] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun und M. Hutter. "Learning quadrupedal locomotion over challenging terrain". In: *Science robotics* 5.47 (2020). DOI: 10.1126/scirobotics.abc5986.

[84] C. Leibig, V. Allken, M. S. Ayhan, P. Berens und S. Wahl. "Leveraging uncertainty information from deep neural networks for disease detection". In: *Scientific reports* 7.1 (2017), S. 17816. DOI: 10.1038/s41598-017-17876-z.

[85] J. J. Leonard und H. F. Durrant-Whyte. "Simultaneous map building and localization for an autonomous mobile robot". In: *Intelligent Robots and Systems' 91.'Intelligence for Mechanical Systems, Proceedings IROS'91. IEEE/RSJ International Workshop on*. Ieee. Osaka, Japan, 1991, S. 1442–1447. DOI: 10.1109/IROS.1991.174711.

[86] Z. Li, X. Cheng, X. B. Peng, P. Abbeel, S. Levine, G. Berseth und K. Sreenath. "Reinforcement Learning for Robust Parameterized Locomotion Control of Bipedal Robots". In: *IEEE International Conference on Robotics and Automation (ICRA)*. Xi'an, China, 2021. DOI: 10.1109/ICRA48506.2021.9560769.

[87] J. Lin. "On the dirichlet distribution". Magisterarb. Kingston Ontario, Canada: Queen's University, 2016.

[88] Y. Liu, Z. Li, C. Zhou, Y. Jiang, J. Sun, M. Wang und X. He. "Generative adversarial active learning for unsupervised outlier detection". In: IEEE, 2019. DOI: 10.1109/TKDE.2019.2905606.

[89] S. Lloyd. "Least squares quantization in PCM". In: *IEEE transactions on information theory* 28.2 (1982), S. 129–137. DOI: 10.1109/TIT.1982.1056489.

[90] A. L. Maas, A. Y. Hannun, A. Y. Ng et al. "Rectifier nonlinearities improve neural network acoustic models". In: *Proceedings of the 30th International Conference on Machine Learning*. Bd. 28. Atlanta, GA, USA: JMLR.org, 2013.

[91] D. J. MacKay. "A practical Bayesian framework for backpropagation networks". In: *Neural computation* 4.3 (1992), S. 448–472. DOI: 10.1162/neco.1992.4.3.448.

[92] S. M. McKinney, M. Sieniek, V. Godbole, J. Godwin, N. Antropova, H. Ashrafian, T. Back, M. Chesus, G. S. Corrado, A. Darzi et al. "International evaluation of an AI system for breast cancer screening". In: *Nature* 577.7788 (2020), S. 89–94. DOI: 10.1038/s41586-019-1799-6.

[93] T. M. Mitchell. *Machine Learning*. 1. Aufl. USA: McGraw-Hill, Inc., 1997. ISBN: 0070428077.

[94] B. Mittelstadt, C. Russell und S. Wachter. "Explaining explanations in AI". In: *Proceedings of the conference on fairness, accountability, and transparency*. 2019, S. 279–288. DOI: 10.1145/3287560.3287574.

[95] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver und K. Kavukcuoglu. "Asynchronous Methods for Deep Reinforcement Learning". In: 2016. arXiv: 1602.01783.

[96] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski et al. "Human-level control through deep reinforcement learning". In: *Nature* 518.7540 (2015), S. 529. DOI: 10.1038/nature14236.

[97] T. N. Mundhenk, W. Einhaeuser und L. Itti. "Automatic Computation of an Image's Statistical Surprise Predicts Performance of Human Observers on a Natural Image Detection Task". In: *Vision Research* 49.13 (2009), S. 1620–1637. DOI: 10.1016/j.visres.2009.03.025.

[98] K. P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022. URL: probml.ai.

[99] V. Nair und G. E. Hinton. "Rectified linear units improve restricted boltzmann machines". In: *Icml*. 2010.

[100] R. Neal. *Bayesian Learning for Neural Networks*. Lecture Notes in Statistics. Springer New York, 1996. ISBN: 9781461207450. DOI: 10.1007/978-1-4612-0745-0.

[101] R. M. Neal. "Connectionist learning of belief networks". In: *Artificial Intelligence* 56.1 (1992), S. 71–113. ISSN: 0004-3702. DOI: 10.1016/0004-3702(92)90065-6.

[102] J. von Neumann. "Zur Theorie der Gesellschaftsspiele". In: *Mathematische Annalen* 100.1 (1928), S. 295–320. DOI: 10.1007/BF01448847.

[103] J. von Neumann und O. Morgenstern. "Theory of Games and Economic Behavior". In: *Science and Society* 9.4 (1944), S. 366–369.

[104] D. A. Nix und A. S. Weigend. "Estimating the mean and variance of the target probability distribution". In: *Proceedings of 1994 ieee international conference on neural networks (ICNN'94)*. Bd. 1. IEEE. 1994, S. 55–60. DOI: 10.1109/ICNN.1994.374138.

[105] *Oeuvres complètes de Voltaire*. Oeuvres complètes de Voltaire Bd. 12,Teil 1. chez Th. Desoer, 1817. URL: https://books.google.com/books?id=wDQTAAAAQAAJ.

[106] I. Osband, J. Aslanides und A. Cassirer. "Randomized Prior Functions for Deep Reinforcement Learning". In: *ArXiv e-prints* (2018). arXiv: 1806.03335 [stat.ML].

[107] I. Osband. "Risk versus uncertainty in deep learning: Bayes, bootstrap and the dangers of dropout". In: *NIPS workshop on bayesian deep learning*. Bd. 192. 2016.

[108] I. Osband, C. Blundell, A. Pritzel und B. Van Roy. "Deep Exploration via Bootstrapped DQN". In: *Advances in Neural Information Processing Systems 29*. Hrsg. von D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon und R. Garnett. Curran Associates, Inc., 2016, S. 4026–4034.

[109] K. Pearson. "On lines and planes of closest fit to systems of points in space". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (1901), S. 559–572. DOI: 10.1080/14786440109462720.

[110] T. Phan, L. Belzner, T. Gabor, A. Sedlmeier, F. Ritz und C. Linnhoff-Popien. "Resilient Multi-Agent Reinforcement Learning with Adversarial Value Decomposition". In: *35th AAAI Conference on Artificial Intelligence (AAAI 2021)*. 2021.

[111] T. Phan, T. Gabor, A. Sedlmeier, F. Ritz, B. Kempter, C. Klein, H. Sauer, R. Schmid, J. Wieghardt, M. Zeller et al. "Learning and Testing Resilience in Cooperative Multi-Agent Systems". In: *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems*. 2020, S. 1055–1063.

[112] M. A. Pimentel, D. A. Clifton, L. Clifton und L. Tarassenko. "A review of novelty detection". In: *Signal Processing* 99 (2014), S. 215 –249. ISSN: 0165-1684. DOI: 10.1016/j.sigpro.2013.12.026.

[113] M. L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014. ISBN: 978-1-118-62587-3. DOI: 10.1002/9780470316887.

[114] J. Quiñonero-Candela, M. Sugiyama, N. D. Lawrence und A. Schwaighofer. *Dataset shift in machine learning*. Mit Press, 2009.

[115] P. Rashidi und A. Mihailidis. "A survey on ambient-assisted living tools for older adults". In: *IEEE journal of biomedical and health informatics* 17.3 (2012), S. 579–590. DOI: 10.1109/JBHI.2012.2234129.

[116] J. Redmon, S. Divvala, R. Girshick und A. Farhadi. "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, S. 779–788. DOI: 10.1109/CVPR.2016.91.

[117] J. Redmon und A. Farhadi. "Yolov3: An incremental improvement". In: *arXiv preprint arXiv:1804.02767* (2018).

[118] F. Ritz, T. Phan, R. Müller, T. Gabor, A. Sedlmeier, M. Zeller, J. Wieghardt, R. Schmid, H. Sauer, C. Klein und C. Linnhoff-Popien. "SAT-MARL: Specification Aware Training in Multi-Agent Reinforcement Learning". In: *Proceedings of the 13th International Conference on Agents and Artificial Intelligence - Volume 1: ICAART,* INSTICC. SciTePress, 2021, S. 28–37. ISBN: 978-989-758-484-8. DOI: 10.5220/0010189500280037.

[119] P. Ruppel, F. Gschwandtner, C. K. Schindhelm und C. Linnhoff-Popien. "Indoor navigation on distributed stationary display systems". In: *2009 33rd Annual IEEE International Computer Software and Applications Conference*. Bd. 1. IEEE. 2009, S. 37–44. DOI: 10.1109/COMPSAC.2009.15.

[120] S. J. Russell und P. Norvig. *Artificial Intelligence - A Modern Approach, Third International Edition*. Pearson Education, 2010. ISBN: 978-0-13-207148-2.

[121] H. Samet und A Soffer. "Automatic interpretation of floor plans using spatial indexing". In: *Progress in Image Analysis and Processing* 3 (1994), S. 233.

[122] J. Schulman, S. Levine, P. Abbeel, M. Jordan und P. Moritz. "Trust region policy optimization". In: *International conference on machine learning*. PMLR. 2015, S. 1889–1897.

[123] J. Schulman, F. Wolski et al. "Proximal Policy Optimization Algorithms". In: 2017. arXiv: 1707.06347.

[124] A. Sedlmeier und S. Feld. "Discovering and Learning Recurring Structures in Building Floor Plans". In: *LBS 2018: 14th International Conference on Location Based Services*. Springer, 2018, S. 151–170. DOI: 10.1007/978-3-319-71470-7_8.

[125] A. Sedlmeier und S. Feld. "Learning indoor space perception". In: *Journal of Location Based Services* 12.3-4 (2018), S. 179–214. DOI: 10.1080/17489725.2018.1539255.

[126] A. Sedlmeier, T. Gabor, T. Phan und L. Belzner. "Uncertainty-Based Out-of-Distribution Detection in Deep Reinforcement Learning". In: Bd. 4. 1. Springer, 2020, S. 74–78. DOI: 10.1007/s42354-019-0238-z.

[127] A. Sedlmeier, T. Gabor, T. Phan, L. Belzner und C. Linnhoff-Popien. "Uncertainty-based Out-of-Distribution Classification in Deep Reinforcement Learning". In: *Proceedings of the 12th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART,* INSTICC. SciTePress, 2020, S. 522–529. ISBN: 978-989-758-395-7. DOI: 10.5220/0008949905220529.

[128] A. Sedlmeier, M. Kölle, R. Müller, L. Baudrexel und C. Linnhoff-Popien. "Quantifying Multimodality in World Models". In: *Proceedings of the 14th International Conference on Agents and Artificial Intelligence - Volume 1: ICAART,* INSTICC. SciTePress, 2022, S. 367–374. ISBN: 978-989-758-547-0. DOI: 10.5220/0010898500003116.

[129] A. Sedlmeier, R. Müller, S. Illium und C. Linnhoff-Popien. "Policy Entropy for Out-of-Distribution Classification". In: *Artificial Neural Networks and Machine Learning – ICANN 2020*. Hrsg. von I. Farkaš, P. Masulli und S. Wermter. Cham: Springer International Publishing, 2020, S. 420–431. ISBN: 978-3-030-61616-8. DOI: 10.1007/978-3-030-61616-8_34.

[130] C. E. Shannon. "A mathematical theory of communication". In: *The Bell System Technical Journal* 27.3 (1948), S. 379–423. DOI: 10.1002/j.1538-7305.1948.tb01338.x.

[131] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan et al. "Natural tts synthesis by conditioning wavenet on mel spectrogram predictions". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, S. 4779–4783. DOI: 10.1109/ICASSP.2018.8461368.

[132] D. Silver. *Lectures on Reinforcement Learning*. URL: https://www.davidsilver.uk/teaching/. 2015.

[133] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan und D. Hassabis. "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play". In: *Science* 362.6419 (2018), S. 1140–1144. DOI: 10.1126/science.aar6404.

[134] G. Snook. "Simplified 3D Movement and Pathfinding Using Navigation Meshes". In: *Game Programming Gems*. Hrsg. von M. DeLoura. Charles River Media, 2000, S. 288–304.

[135] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever und R. Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1 (2014), S. 1929–1958.

[136] P. Stone und M. Veloso. "Multiagent systems: A survey from a machine learning perspective". In: *Autonomous Robots* 8.3 (2000), S. 345–383. DOI: 10.1023/A:1008942012299.

[137] R. S. Sutton und A. G. Barto. *Introduction to reinforcement learning*. Bd. 135. MIT press Cambridge, 1998. DOI: 10.1109/TNN.1998.712192.

[138] C. Tandy. "The isovist method of landscape survey". In: *Methods of Landscape Analysis*. Hrsg. von C. Murray. London, UK: Landscape Research Group, 1967, S. 9–10.

[139] *Unity – Game Engine*. http://unity3d.com. Accessed: 2022-11-20.

[140] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev et al. "Grandmaster level in StarCraft II using multi-agent reinforcement learning". In: *Nature* 575.7782 (2019), S. 350–354. DOI: 10.1038/s41586-019-1724-z.

[141] R. C. Voorhies, L. Elazary und L. Itti. "Application of a Bottom-Up Visual Surprise Model for Event Detection in Dynamic Natural Scenes". In: *Proc. Vision Science Society Annual Meeting (VSS10)*. 2010. DOI: 10.1167/10.7.215.

[142] Q. Wang, I. L. Moreno, M. Saglam, K. Wilson, A. Chiao, R. Liu, Y. He, W. Li, J. Pelecanos, M. Nika und A. Gruenstein. "VoiceFilter-Lite: Streaming Targeted Voice Separation for On-Device Speech Recognition". In: *Proc. Interspeech 2020*. 2020, S. 2677–2681. DOI: 10.21437/Interspeech.2020-1193.

[143] C. J. C. H. Watkins. "Learning from delayed rewards". Diss. King's College, Cambridge, 1989.

[144] M. Weber, C. Langenhan, T. Roth-Berghofer, M. Liwicki, A. Dengel und F. Petzold. "a. SCatch: semantic structure for architectural floor plan retrieval". In: *International Conference on Case-Based Reasoning*. Springer. Alessandria, Italy, 2010, S. 510–524.

[145] M. Werner. *Indoor location-based services: Prerequisites and foundations*. Springer, 2014. DOI: 10.1007/978-3-319-10699-1.

[146] R. J. Williams und J. Peng. "Function optimization using connectionist reinforcement learning algorithms". In: *Connection Science* 3.3 (1991), S. 241–268. DOI: 10.1080/09540099108946587.

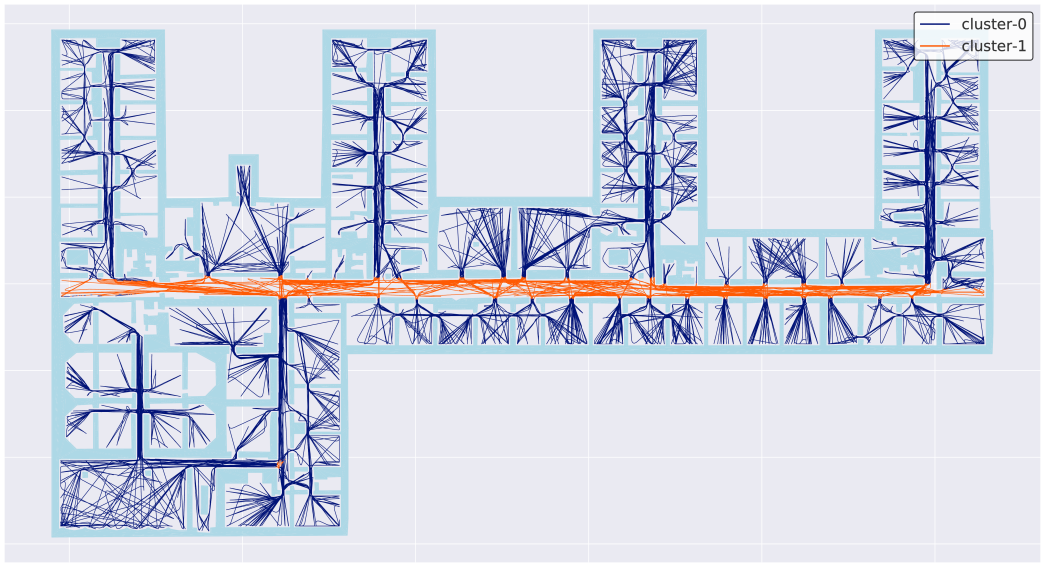
[147] J. Yim, R. Chopra, T. Spitz, J. Winkens, A. Obika, C. Kelly, H. Askham, M. Lukic, J. Huemer, K. Fasler et al. "Predicting conversion to wet age-related macular degeneration using deep learning". In: *Nature Medicine* 26.6 (2020), S. 892–899. DOI: 10.1038/s41591-020-0867-7.

[148] C. Zhang, O. Vinyals, R. Munos und S. Bengio. "A Study on Overfitting in Deep Reinforcement Learning". In: (2018). arXiv: 1804.06893 [cs.LG].

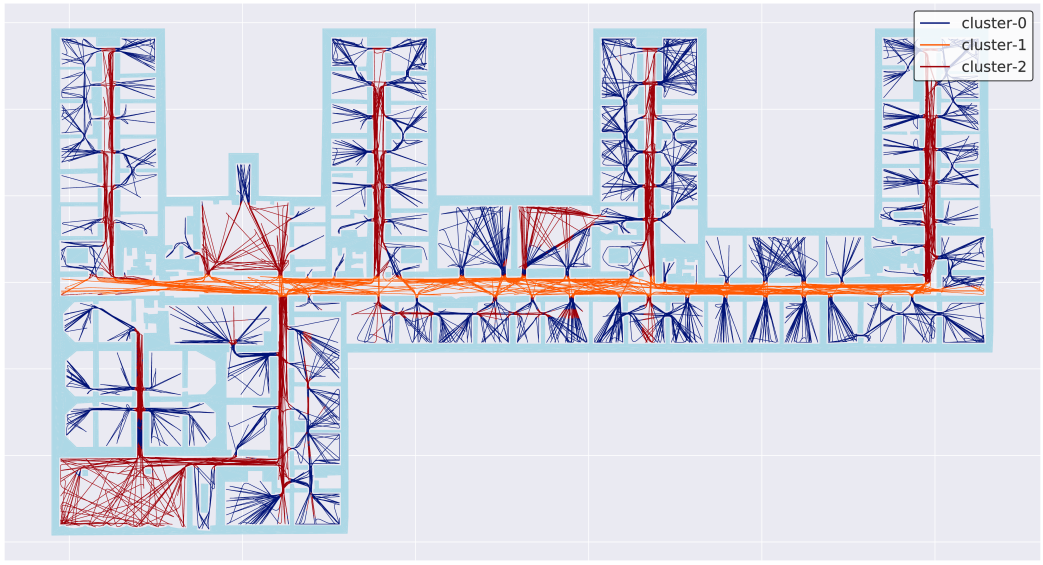
[149] Y. Zhao, Z. Nasrullah und Z. Li. "PyOD: A Python Toolbox for Scalable Outlier Detection". In: *Journal of Machine Learning Research* 20.96 (2019), S. 1–7.

[150] *blender.org - Home of the Blender project - Free and Open 3D Creation Software*. https://www.blender.org/. Accessed: 2022-11-20.

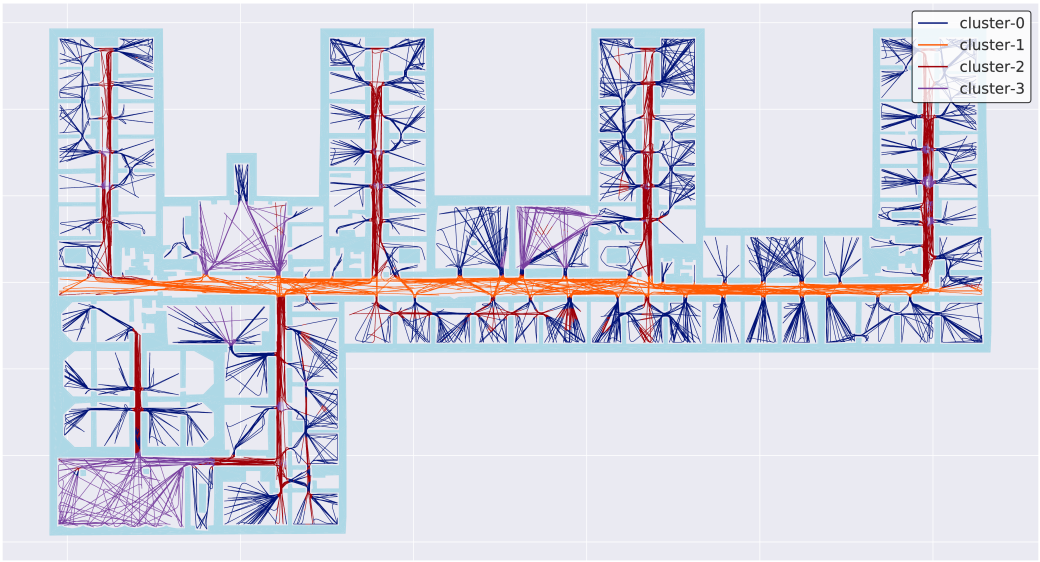
Anhang



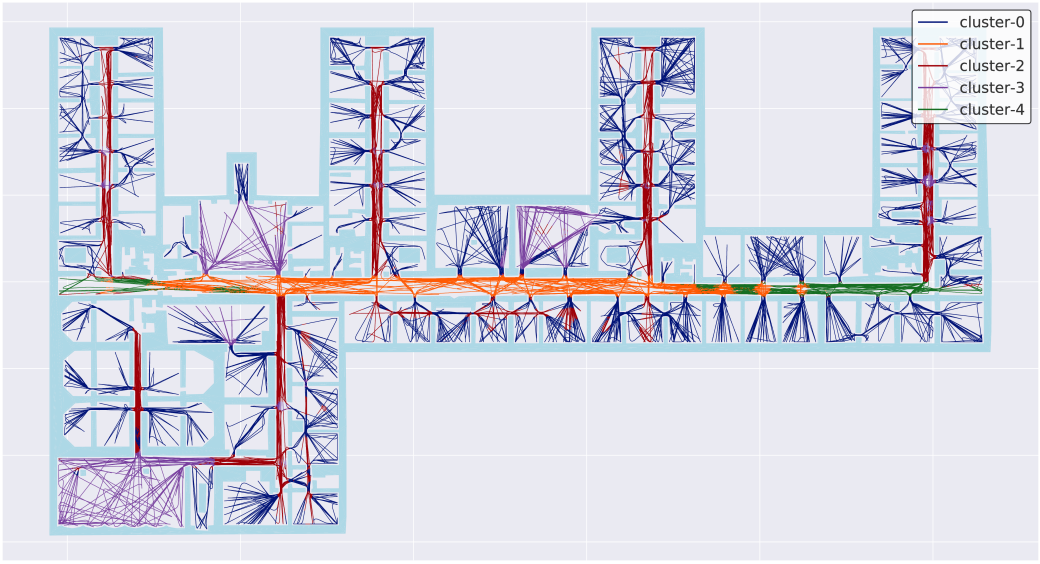
(a) $k = 2$



(b) $k = 3$



(c) $k = 4$



(d) $k = 5$

Abbildung A1: Visualisierung von k-means basierten Clusterings, basierend auf den statischen Isovisten Features die auf der LMU Umgebung gemessen wurden.

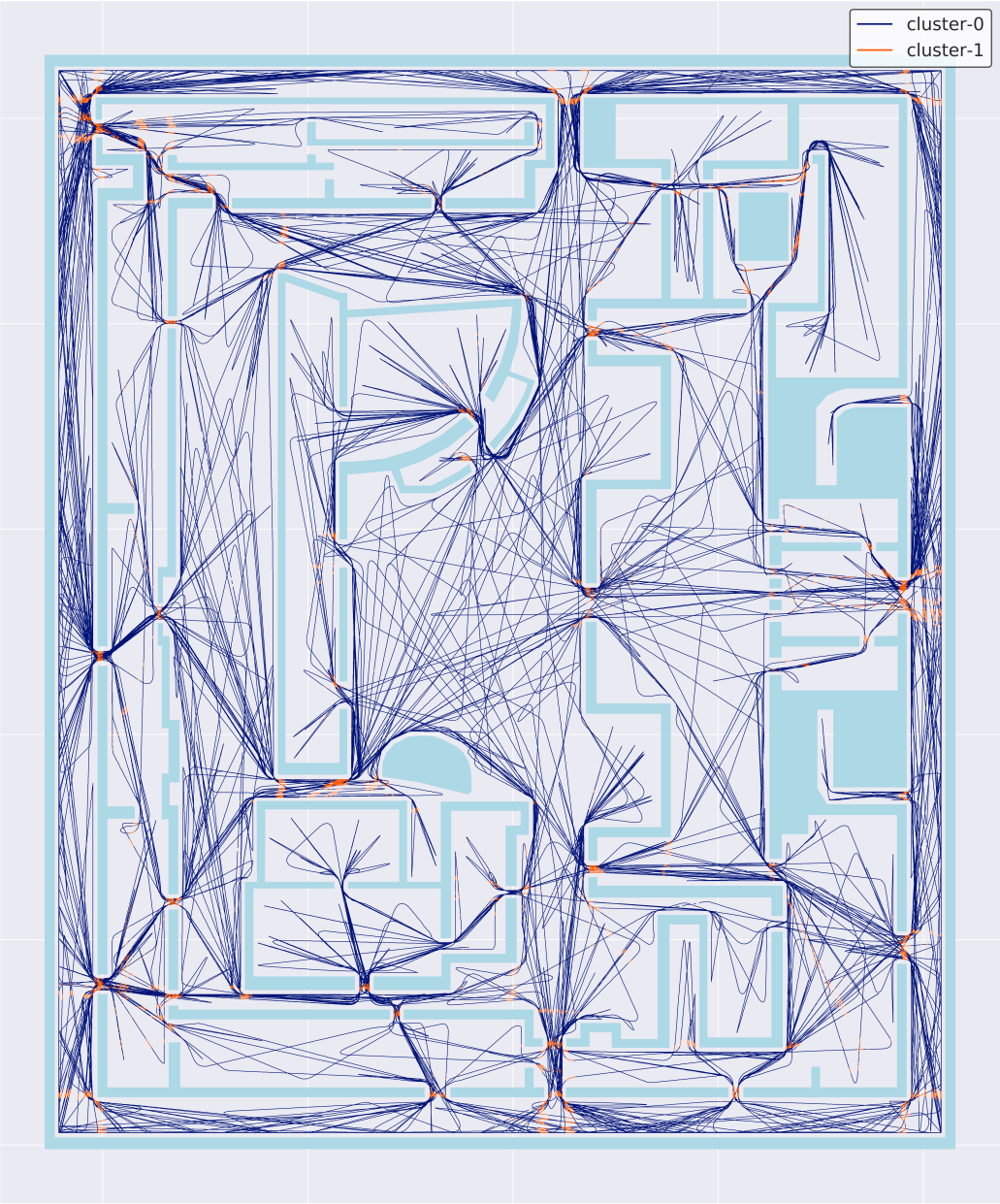


Abbildung A2: Visualisierung eines k-means basierten Clusterings mit $k = 2$, basierend auf den dynamischen Isovisten Features, erzeugt mittels SMA $n = 5$ auf der TUM Umgebung.

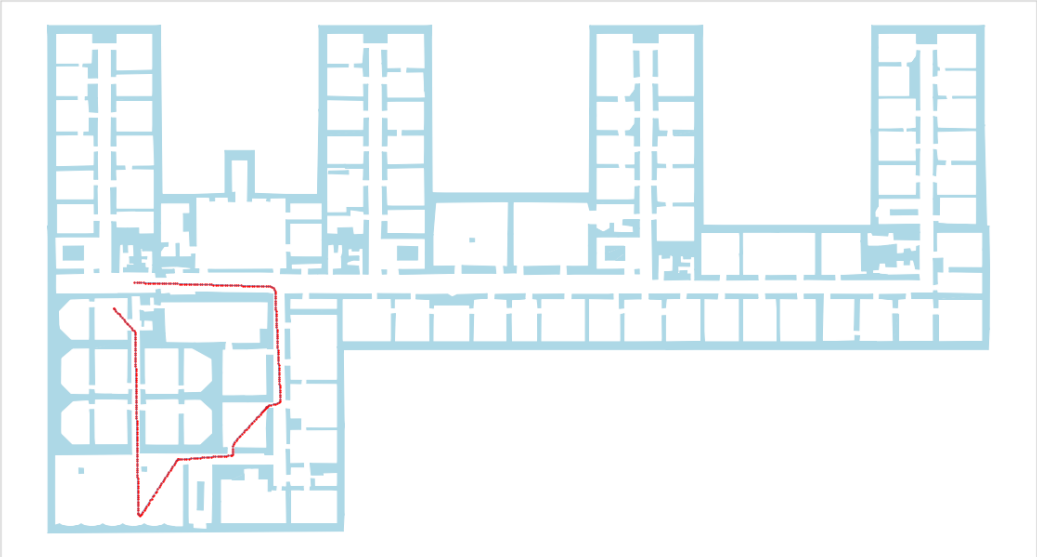


Abbildung A3: Visualisierung der gesamten LMU Umgebung, mit manuell definierter Trajektorie aus Unterabschnitt 3.8.5.