wirtschafts
wissenschaften
htw saar

Hochschule für
Technik und Wirtschaft
des Saarlandes
University of
Applied Sciences

Fakultät für
Wirtschafts-
wissenschaften
Business School

Forschungsbericht

Erweiterung der Kenntnisse im Bereich Künstliche Intelligenz (KI) mit den Themen Multiagentensysteme (MAS), Reinforcement Learning (RL), Künstliche Neuronale Netzwerke (KNN) & Deep Learning (DL)

Stefan Selle

Professor für Wirtschaftsinformatik
Fakultät für Wirtschaftswissenschaften
Hochschule für Technik und Wirtschaft des Saarlandes

Saarbrücken, 16.05.2018

Inhaltsverzeichnis

Abkürzungsverzeichnis

A3C	Asynchronous Advantage Actor Critic
AAL	Ambient Assisted Living
ACL	Agent Communication Language
API	Application Programming Interface
BDI	Belief-Desire-Interaction
BIOS	Basic Input Output System
BPTT	Backpropagation Through Time
BWL	Betriebswirtschaftslehre
CEO	Chief Executive Officer
CIFAR	Canadian Institute For Advanced Research
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CRM	Customer Relationship Management
CUDA	Compute Unified Device Architecture
cuDNN	CUDA Deep Neural Network Library
DDQN	Double Deep Q-Network
DL	Deep Learning
DL4J	Deep Learning for Java
DQN	Deep Q-Network
ES	Evolution Strategies
FIPA	Foundation for Intelligent Physical Agents
GB	Gigabyte
GHz	Gigahertz
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
htw saar	Hochschule für Technik und Wirtschaft des Saarlandes
I/O	Input/Output

IBM	International Business Machines
IDS	Intrusion Detection System
IMDb	Internet Movie Database
IoT	Internet of Things
IT	Information Technology
JADE	Java Agent Development Framework
kHz	Kilohertz
KI	Künstliche Intelligenz
km/h	Kilometer pro Stunde
KMU	Kleine und mittlere Unternehmen
KNN	Künstliches Neuronales Netzwerk
LKW	Lastkraftwagen
LSTM	Long Short-Term Memory
LTS	Long Term Support
M2M	Maschine-zu-Maschine
MAS	Multiagentensystem
MDP	Markov Decision Process
MHz	Megahertz
MLP	Multilayer Perceptron
MNIST	Modified National Institute of Standards and Technology
MSI	Micro-Star International
OCR	Optical Character Recognition
OWL	Web Ontology Language
PC	Personal Computer
PPO	Proximal Policy Optimization
RDF	Resource Description Framework
ReLU	Rectified Linear Unit
RL	Reinforcement Learning
RL4J	Reinforcement Learning for Java
RNN	Recurrent Neural Network
SARSA	State–Action–Reward–State–Action

SSH	Secure Shell
TCO	Total Cost of Ownership
US	United States
VKI	Verteilte Künstliche Intelligenz
VWL	Volkswirtschaftslehre
WiWi	Wirtschaftswissenschaften
XML	Extensible Markup Language
ZMS	Zentrum Mittelstand Saar

1 Einleitung

KI, also Künstliche Intelligenz, ist ein Thema, das aktuell für viel Resonanz in den Medien sorgt. Auf der aktuellen Entwicklerkonferenz I/O hat am 08.05.2018 der Google-CEO Sundar Pichai auf seiner Keynote den neuen Service Duplex vorgestellt [Kre18]. Dieser neue digitale Assistent ist auf bestimmte Gesprächssituationen trainiert worden und kann auf Zuruf eigenständig Telefonate führen, um bspw. Reservierungen in Restaurants vorzunehmen oder Termine beim Friseur zu buchen. Hierzu kann der Assistent auch selbstständig die notwendigen Telefonnummern im Internet suchen und die Ergebnisse nach dem Telefonat automatisch im digitalen Kalender des Nutzers eintragen. Das zugleich Faszinierende und Erschreckende an der eindrucksvollen Demonstration war aber, dass kaum noch zu erkennen ist, das eine Maschine das Telefongespräch führt.

Den Turing-Test würde Duplex aber (noch) nicht bestehen. Dieser ist nach dem englischen Mathematiker Alan Turing (23.06.1912 - 07.06.1954) benannt. In einem Gedankenexperiment führt dabei ein Mensch per Tastatur und Monitor ein Gespräch mit einem Gegenüber, ohne diesen zu sehen oder zu hören. Der Gegenüber ist entweder ein Mensch oder eine Maschine. Der Turing-Test gilt für eine Maschine dann als bestanden, wenn nach einer intensiven Befragung nicht zweifelsfrei entschieden werden kann, dass es sich bei dem Gegenüber nicht um einen Menschen handelt. In diesem Fall würde man der Maschine also menschenähnliches Denkvermögen und damit Intelligenz attestieren.

Ob nun natürlich oder künstlich: Der Begriff Intelligenz ist sehr schwer in Worte zu fassen und es gibt bis heute keine allgemein akzeptierte Definition. 1912 hat der deutsche Psychologe William Stern (29.04.1871 - 27.03.1938) eine noch vage Erklärung geliefert [Ste12]: « *Intelligenz ist die allgemeine Fähigkeit eines Individuums, sein Denken bewusst auf neue Forderungen einzustellen; sie ist allgemeine geistige Anpassungsfähigkeit an neue Aufgaben und Bedingungen des Lebens.* » In dieser Aussage wird der Fokus auf den Aspekt Adaptionsfähigkeit gerichtet. Mittlerweile kann Intelligenz jedoch als ein Sammelbegriff für kognitive Fähigkeiten aufgefasst werden. Hierzu gehören neben der Adaptionsfähigkeit auch höhere mentale Prozesse wie Erkenntnisvermögen, abstraktes Denken, Repräsentation, Urteilsfähigkeit, Problemlösen, Entscheidungsfindung, Kommunikationsfähigkeit, Interaktionsfähigkeit, Kreativität, Emotionalität, Empathie usw.

Wenn heutzutage der Begriff Künstliche Intelligenz benutzt wird, ist damit meistens die sogenannte schwache KI gemeint. In dieser geht es darum, Computersysteme zu entwickeln, die in sehr konkreten Anwendungsfällen zur Lösung spezifischer Probleme eingesetzt werden können. Dabei wenden diese Maschinen maschinelles Lernen an, d.h. sie optimieren sich bzw. ihre Parameter unter dem Einsatz geeigneter Algorithmen selbst.

Eine im Sinn der schwachen KI intelligente Maschine, die im März 2016 für Aufsehen sorgte, ist AlphaGo von Google DeepMind. Sie hat den südkoreanischen Profispieler Lee Sodol unter Turnierbedingungen mit 4 zu 1 im Brettspiel Go geschlagen [Spi16]. Das ist deshalb so bemerkenswert, weil Go sehr viel komplexer als Schach ist und deshalb nicht die gleichen Algorithmen wie bei einem Schachcomputer verwendet werden können. Ein Schachcomputer funktioniert nach dem Prinzip, zunächst alle Zug-Kombinationsmöglichkeiten

als einen Baum abzubilden und dann mit einer Suche den möglichst besten nächsten Zug in diesem Baum auszuwählen. Hierfür reicht die Rechenleistung moderner Computersysteme aus und bereits im Mai 1997 schlug der IBM-Computer *Deep Blue* den damaligen Weltmeister Garri Kasparow in einem Schachturnier mit 3,5 zu 2,5. Während beim Schach Zug um Zug Figuren vom Brett entfernt werden und somit die Komplexität sinkt, kommen beim Go Zug um Zug neue Spielsteine auf dem viel größeren Brett hinzu, welches aus 19 x 19 Feldern besteht. Es gibt mehr Kombinationsmöglichkeiten als das Universum Atome hat. Google DeepMind hat einen völlig anderen Ansatz benutzt, um *AlphaGo* zu trainieren. Tiefe Künstliche Neuronale Netzwerke (KNN) wurden benutzt, um aus einer Datenbank mit 30 Millionen Stellungen von bereits gespielten Go-Partien zwischen Experten, geeignete Zugkandidaten zu bestimmen. Diese Netzwerke werden mit dem überwachten Lernen (*Supervised Learning*) trainiert. Aufgrund der Größe bzw. Tiefe dieser KNN spricht man auch von *Deep Learning*. Des Weiteren kommt das bestärkende Lernen (*Reinforcement Learning*) zum Einsatz. Dabei spielt *AlphaGo* sehr viele Partien Go gegen sich selbst, macht dabei Fehler und lernt wiederum aus diesen. Dadurch wird das System sehr leistungsstark. In diesem speziellen Anwendungsfall schlägt die Maschine also den Menschen.

KI und die damit verbundene Verdrängung oder Transformation der bestehenden Arbeitsplätze durch intelligente Maschinen wird insbesondere in Deutschland umstritten diskutiert. Prominente Wissenschaftler und Unternehmer wie bspw. der erst kürzlich verstorbene britische theoretische Astrophysiker Stephen Hawking [Dav14] oder die US-amerikanischen *Tech*-Gründer Elon Musk (PayPal, Tesla, SpaceX) [Mus17] und Sergey Brin (Alphabet bzw. Google) [Bri18] warnen sogar vor den möglichen Risiken der KI. Wir sind uns zumindest darüber bewusst, dass die Digitalisierung als globales Phänomen das Arbeits- und Privatleben in den nächsten Jahren massiv verändern wird.

Aufgrund der intensiven Nutzung und großen Verbreitung mobiler Endgeräte, der Auslagerung von IT-Services an *Cloud*-Anbieter, dem Zusammenschluss vieler Geräte im Internet und dem massiven Einsatz sensorischer Maschinen in *Smart Factories* werden enorme Mengen an digitalen Daten generiert (*Big Data*), die ohne den Einsatz geeigneter Algorithmen nicht mehr verarbeitet und analysiert werden können. Zukunftsthemen wie das Internet der Dinge (*Internet of Things (IoT)*) und Industrie 4.0 (*Hightech*-Strategieprojekt der Bundesrepublik Deutschland) sind ohne KI als Motor der digitalen Transformation kaum vorstellbar.

Dieses Forschungsprojekt beschäftigt sich nicht mit den gesellschaftlichen, politischen oder ökonomischen Fragen zur Künstlichen Intelligenz, auch wenn hierzu ebenfalls Handlungsbedarf besteht, um geeignete Rahmenbedingungen für KI zu gestalten. Stattdessen werden im Wesentlichen die Technologien untersucht, die z.Zt. einen großen Stellenwert in der KI-Forschung und den KI-Anwendungen einnehmen, nämlich Multiagentensysteme (MAS), *Reinforcement Learning*, Künstliche Neuronale Netzwerke (KNN) und *Deep Learning (DL)*. Das Ziel ist es also, Wissen in diesen Bereichen aufzubauen, welches dann in der Lehre und angewandten Forschung an der Hochschule für Technik und Wirtschaft des Saarlandes (htw saar) eingesetzt werden kann. Hierzu sollen auch bereits im Rahmen dieses Forschungsprojekts erste KI-Anwendungen durchgeführt werden. Aufgrund der neuen Schnittstelle zwischen Mensch und Maschine durch Sprachsteuerung, sowie der zunehmenden Maschine-zu-Maschine-Kommunikation, sind in den nächsten Jahren sehr viele neue, und vermutlich auch disruptive, Innovationen zu erwarten. Somit fällt dieses Projekt thematisch in den Forschungsschwerpunkt Schnittstellen der htw saar.

Die eigenen Vorarbeiten zu KI-Themen liegen bereits 20 Jahre zurück. Im Jahr 1998 habe ich meine Diplomarbeit im Studiengang Volkswirtschaftslehre der Universität Heidelberg zum Thema *Einsatz Künstlicher Neuronaler Netze auf dem Aktienmarkt* geschrieben [Sel98]. Somit ist es auch eine persönliche Motivation, diese Kenntnisse aufzufrischen (Thema: KNN) und zu erweitern (Themen: MAS, RL, DL).

Dieses Forschungsprojekt ist in vier Teilprojekte strukturiert, was sich auch in der Gliederung dieses Forschungsberichts und in den erstellten Anlagen widerspiegelt:

Kap.	**Kürzel**	**Name des Kapitels bzw. Titel der Anlage**	**Quelle**
2	PC	Vorarbeiten	
		Zusammenstellung, Installation und Konfiguration eines	
		Personal Computers für Deep Learning Projekte	[Sel18d]
3	MAS	Multiagentensysteme	
		Auswahl eines Frameworks für Multiagentensysteme	
		zum Einsatz in Forschung und Lehre	[Sel18a]
4	RL	Reinforcement Learning	
		Softwarelösungen für Reinforcement Learning	[Sel18c]
5	DL	Künstliche Neuronale Netzwerke	
		und Deep Learning	[Sel18b]

Tabelle 1.1: Gliederung des Forschungsberichts und Beziehungen zu den erstellten Anlagen

Im letzten Kapitel wird dann ein kurzes Fazit gezogen und ein Ausblick auf die zukünftige Entwicklung gegeben. Darunter sind noch offene Punkte zu verstehen bzw. Fragen, die sich erst im Laufe des Forschungsprojekts ergeben haben, und die noch weiter untersucht werden sollten. Des Weiteren werden die nächsten Schritte dargestellt, die notwendig sind, um das in diesem Forschungsprojekt erarbeitete fachliche Wissen und praktische Know-how an die zukünftigen Generationen von Studierenden der htw saar weiterzugeben und zu vermitteln. Schließlich wird auch noch auf die Fortführung des Projekts in der angewandten Forschung eingegangen.

2 Vorarbeiten

Ein Ziel dieses Forschungsprojekts ist, neben der Erweiterung der theoretische Kenntnisse zu den KI-Themen Multiagentensysteme, *Reinforcement Learning*, Künstliche Neuronale Netzwerke und *Deep Learning* auch erste praktische Anwendungen in diesem Bereich durchzuführen. Dabei sollen geeignete *Open Source* Softwarelösungen eingesetzt werden. Darunter versteht man Software, deren Quelltext (*Source Code*) öffentlich ist und der von Dritten eingesehen, genutzt und verändert werden darf. Somit fallen für diese Art von Software keine Lizenzgebühren für deren Nutzung an. Der Einsatz dieser Software ist aber trotzdem nicht kostenlos. Nach dem Konzept *Total Cost of Ownership (TCO)* werden Kosten auch durch Hardware, Installation, Konfiguration, Betrieb, Pflege und Wartung verursacht, also wird Budget für Sachmittel, Personal und Energie benötigt.

Zur Ausführung von Software wird die passende Hardware benötigt. Die aufwändigsten Berechnungen sind im Bereich *Deep Learning* zu erwarten, da tiefe KNN trainiert werden müssen, wobei viele Matrix-Multiplikationen durchzuführen sind. Hierzu wird normalerweise nicht der Hauptprozessor (*Central Processing Unit (CPU)*) sondern der Grafikprozessor (*Graphics Processing Unit (GPU)*) verwendet. Denn auf einer *High-End*-Grafikkarte sind mittlerweile sehr viele sogenannte Streamprozessoren untergebracht, sodass sich solche rechenintensiven Operationen gut parallelisieren und sehr effizient ausführen lassen. Das Ziel dieser Vorarbeiten war einerseits die Zusammenstellung, Installation und Konfiguration eines Personal Computers (PC), der für *Deep Learning* Projekte in der Lehre und angewandten Forschung an der htw saar eingesetzt werden kann, als auch anderseits die Erstellung einer Dokumentation, die dann wiederum als Blaupause benutzt werden kann, um ggf. weitere solcher Rechner zu administrieren.

Das Herzstück des verwendeten DL-Rechners ist die Grafikkarte GeForce GTX 1080 Ti von Nvidia. Sie bietet eine sehr hohe Leistung zu einem noch akzeptablen Preis von ca. 750 Euro netto. Bspw. verfügt sie über 3.584 Streamprozessoren, taktet mit 1.594 MHz und kann auf 11 GB eigenen Speicher zugreifen. Die anderen Hardware-Komponenten wurden passend zu dieser Grafikkarte ausgewählt. Der komplette Rechner kostet 2.663,41 Euro brutto (Stand 04.10.2017).

Zunächst wurde die Firmware des MSI-Mainboards aktualisiert, also ein BIOS-Update durchgeführt. Dann wurden spezielle Einstellungen bezüglich der Grafikkarten und Festplatten vorgenommen, damit die Installation des Betriebssystems gelingt und der Betrieb des DL-Rechners möglichst effizient abläuft. Das freie Linux-Betriebssystem Ubuntu 16.04 LTS wurde auf dem Rechner installiert. LTS steht für *Long Term Support*. Die Version 16.04 wird bis April 2021 von Ubuntu unterstützt, d.h. bis dahin gibt es regelmäßig Aktualisierungen, die einen stabilen und sicheren Betrieb gewährleisten. Anschließend wurde das Betriebssystem konfiguriert. Schwierig gestaltete sich dabei die Installation funktionsfähiger Treiber für die Grafikkarte. Die Standard-Treiber mussten deaktiviert werden, um proprietäre Treiber des Herstellers Nvidia installieren zu können. Es gab auch Abhängigkeiten zum Werkzeug CUDA 9.1 von Nvidia, mit dem sich parallele Rechnungen auf der GPU ausführen lassen. In deren Standard-Installation werden immer die nicht-funktionierenden Nvidia-Grafiktreiber automatisch mitinstalliert. Somit wurde

eine alternative Installationsart verwendet, in der man die Installation dieser Treiber deaktivieren kann. Außerdem wurde auch das Werkzeug cuDNN 7.0.5 installiert. Dies ist eine Software-Bibliothek von Nvidia, die auf CUDA aufsetzt und mit der sich tiefe KNN trainieren lassen. cuDNN steht für *CUDA Deep Neural Network Library*. Diese Software ist in C/C++ programmiert. Mittlerweile gibt es aber auch viele DL-Softwarelösungen, die cuDNN unterstützen und weitere *High-Level*-Schnittstellen zu anderen Programmiersprachen anbieten. Die DL-Bibliothek TensorFlow von Google bietet bspw. eine Programmierschnittstelle (*Application Programming Interface (API)*) zu Python an. Im Bereich *Data Science* ist die Programmiersprache Python weit verbreitet. Aus diesem Grund wurde zunächst Anaconda 5.0.1 für Python 3.6 installiert. Anaconda ist eine Python-Distribution, in der u.a. auch Pakete mitgeliefert werden, die zur Verarbeitung und Analyse von großen Datenmengen eingesetzt werden können. Außerdem ist das Jupyter Notebook enthalten, ein Webbasiertes Frontend, mit dem sich Python-Skripte einfach per Browser erstellen und ausführen lassen. Das Jupyter Notebook wurde so eingerichtet, dass es sich per SSH-Tunnel von einem anderen Rechner aus bedienen lässt. Hierzu wurde auf dem DL-Rechner ein SSH-Dienst eingerichtet und auf dem *Client* das Programm PuTTY verwendet. Schließlich wurde dann auch das DL-Framework TensorFlow 1.5.0 installiert. Normalerweise ist dies ein sehr einfacher Prozess. Aufgrund von Inkompatibilitäten der aktuell ausgelieferten TensorFlow-Version zu CUDA 9.1 und cuDNN 7.0.5 mussten jedoch die offenen Quellcodes erst kompiliert und das Programm danach mit dem Werkzeug Bazel zusammengebaut werden.

Nach der erfolgreichen Installation wurde dann abschließend ein Python-Skript als Anwendungsbeispiel ausgeführt, um die Leistung des Systems zu messen und zu dokumentieren. Hierzu wurden mittels der TensorFlow-Bibliothek immer größer werdende Matrizen miteinander multipliziert. Die Rechnungen wurden sowohl von der GPU als auch der CPU ausgeführt und die dafür benötigten Zeitdauern gemessen. Die Multiplikation zweier 1.400 x 1.400 Matrizen hat auf der CPU bereits 5,3 Sekunden gedauert. Dabei handelt es sich immerhin um den Intel Core i7-7700K Prozessor mit 4 Kernen und 4,2 GHz Taktfrequenz. Für die gleiche Berechnung benötigte die GPU nur 0,16 Sekunden, das ist die 33-fache Geschwindigkeit. In 5,3 Sekunden schaffte die GPU sogar eine Matrix-Multiplikation mit der Dimension von 26.000 x 26.000.

Das Ziel dieser Vorarbeiten wurde also erreicht. Ein fertig konfigurierter Rechner steht nun zur weiteren Verwendung zur Verfügung. Außerdem kann die Dokumentation als Blaupause in Form einer Anleitung dienen, um sehr schnell weitere DL-Rechner zu konfigurieren.

3 Multiagentensysteme

Ein Agent bzw. Softwareagent ist ein Computerprogramm, das sich ähnlich wie ein menschlicher Dienstleistungsagent bzw. Assistent verhält. Ein Beispiel für ein Softwareagent ist ein *Chatbot*. Der Begriff setzt sich zusammen aus den englischen Wörtern *Chat* für Plaudern und *Robot* für Roboter bzw. Maschine. In sozialen Netzwerken gibt es Nutzerprofile, die nicht von Menschen, sondern von Maschinen gepflegt werden. Diese Chatbots können textbasierte Dialoge mit anderen Teilnehmern führen. Negative Schlagzeilen haben Chatbots vor allem in Form von sogenannten Propaganda-Bots während des letzten US-Präsidentschaftswahlkampfs erlangt. Aus Sicht von Unternehmen können Chatbots aber wertvolle Dienste leisten und bspw. im Dialog mit externen Kunden oder potenziellen Bewerbern eingesetzt werden. Im Bereich *Mobile Computing* ist mittels Smartphone auch eine Kommunikation per natürlicher Sprache zwischen Mensch und Maschine möglich. Diese Systeme sind mit einer Spracherkennungssoftware ausgestattet und liefern die Antwort per Sprachausgabe an den Benutzer zurück. Bekannte digitale Assistenten sind bspw. Apples Siri, Amazons Alexa, Microsofts Cortana oder Googles Assistent. Diese Softwareagenten liefern ggf. sinnvolle Antworten auf unsere alltäglichen Fragen, können aber auch bereits Produkte im Internet bestellen oder vernetzte Geräte in unserem Haushalt steuern. Man kann also zweierlei feststellen: Die Softwareagenten werden immer intelligenter und die Interaktionen zwischen Menschen und Maschinen nehmen zu.

Multiagentensysteme (MAS) sind Computersysteme, bei denen mehrere Agenten zusammenwirken. MAS gehören zum Teilgebiet der Verteilten Künstlichen Intelligenz (VKI). Maschine-zu-Maschine-Kommunikation (M2M) ist ein typischer Bestandteil dieser Systeme und eine wesentliche Voraussetzung, damit die Agenten gemeinsam eine komplexe Aufgabe bearbeiten, die sie alleine nicht bewältigen könnten. Ganz ähnlich funktionieren Unternehmen bzw. Organisationen, allerdings werden diese normalerweise zentral gesteuert. In einem MAS existiert dagegen keine zentrale Steuereinheit, sondern die Agenten agieren weitgehend autonom, weshalb das System sehr robust ist. Das Internet selbst basiert auf einem ähnlichen Konzept und wurde so entwickelt, dass es möglichst ausfallsicher ist. Im *Internet of Things (IoT)* werden vor allem Dinge, also Maschinen, miteinander vernetzt. Daraus ergeben sich diverse Anwendungsfelder wie bspw. das *Smart Home* oder die *Smart Factory*.

Das Ziel dieses Teilprojekts war das Suchen und Finden eines geeigneten Software-Frameworks für Multiagentensysteme, das möglichst universell bei der Entwicklung und Simulation typischer MAS-Anwendungen eingesetzt werden kann. Zunächst wurde Grundlagenwissen aufgebaut und dokumentiert. Die wesentlichen Vorarbeiten zum Thema Agenten stammen von Stuart Russel und Peter Norvig [RN03]. Sie haben Agenten hinsichtlich ihres inneren Aufbaus und ihres Verhaltens klassifiziert und dabei folgende Typen definiert: einfacher Reflex-Agent, modellbasierte Reflex-Agent, zielbasierter Agent, nutzenbasierter Agent und lernender Agent. Der lernende Agent entspricht dem deliberalen Agenten von Michael Wooldridge [Woo09]. Er nimmt seine Umwelt über Sensoren (z.B. Temperaturfühler) war und kann diese wiederum über Aktoren (z.B. Heizungsregler) beeinflussen. Er verfügt aber auch über ein Modell dieser Welt, und kann adaptiv

seinen internen Zustand anpassen. Aufgrund von Ziel- und Planer-Komponenten lassen sich rationale Entscheidungsprozesse abbilden. Somit kann dieser Typ von Agent auch als intelligenter bzw. kognitiver Agent bezeichnet werden. Das Konzept des BDI-Agenten setzt genau hier an. Die Komponenten *Beliefs* (Annahmen), *Desires* (Ziele) und *Intentions* (Absichten) beschreiben den internen Zustand des Agenten. Werden Agenten nun zu einem MAS zusammengeschlossen, dann sind Kommunikation und Koordination zentrale Aspekte, deren Funktionsweise definiert werden muss. Die *Foundation for Intelligent Physical Agents (FIPA)* hat zur Standardisierung eine Referenzarchitektur entwickelt. Dort ist sehr genau spezifiziert, wie bspw. die Kommunikation per *Agent Communication Language (ACL)* abläuft und die ausgetauschten Nachrichten aufgebaut sein müssen.

Standards im IT-Bereich sorgen für Investitionssicherheit. Somit wurden Softwarelösungen gesucht, die Standards wie FIPA und BDI verwenden. Der Einsatz eines Frameworks hat generell viele Vorteile, u.a. muss durch dessen Verwendung nicht alles komplett neu entwickelt werden. Die Software muss auch einige zusätzlichen Anforderungen erfüllen: universeller Einsatzzweck, Plattformunabhängigkeit, Quelloffenheit für flexible Weiterentwicklung. Letztendlich wurden aus zahlreichen Frameworks fünf herausgefiltert, die dann näher analysiert wurden. Anhand der Merkmale Aktualität, Funktionsumfang, Hilfe bzw. Dokumentation, Reputation und Präsentation wurden diese bewertet und miteinander verglichen. Hierzu wurde eine Nutzwertanalyse durchgeführt, wobei die fünf Merkmale der Einfachheit halber gleich gewichtet wurden. JADE erzielte bei diesem Vergleich den höchsten Nutzwert. Dieses Framework wurde von Telecom Italia entwickelt und wird seit dem Jahr 2000 als *Open Source* Lösung angeboten und kontinuierlich weiterentwickelt. Es gibt eine große *Community*, sehr gute Dokumentationen und die aktuelle Version ist nur wenige Monate alt.

Mit JADE wurde eine Beispielanwendung zum Kontext Buchhandel durchgeführt. Zunächst wurden verschiedene Konzepte analysiert, wie damit Agenten in der objektorientierten Programmiersprache Java erstellt werden können. Die Idee hinter der Beispielanwendung ist ein einfacher Marktplatz: Ein `BookBuyerAgent` möchte ein bestimmtes Buch kaufen, ein oder mehrere `BookSellerAgents` verkaufen Bücher. Der Kommunikationsablauf kann mittels ACL-Nachrichten realisiert werden. Die Installation, Konfiguration und der Betrieb von JADE wurde dann exemplarisch anhand dieser Beispielanwendung unter dem Betriebssystem Windows ausgeführt.

Das Framework hat alle Erwartungen erfüllt, lediglich die *Performance* war nicht optimal. Da die Kommunikation der Agenten asynchron erfolgt und die Agenten als Java-*Threads* ausgeführt werden, ergaben sich teilweise Wartezeiten im Sekundenbereich während der Agentenkommunikation. Somit sollte JADE nicht für zeitkritische Anwendungen verwendet werden. Das Ziel dieses Teilprojekts wurde erreicht: JADE ist ein Multiagentensystem, das in angewandter Forschung und Lehre an der htw saar zukünftig mit der eben genannten Einschränkung eingesetzt werden kann.

4 Reinforcement Learning

Direkt nach der Geburt kann eine Antilope noch nicht einmal richtig stehen. Aber auch ohne die Hilfe ihrer Mutter lernt sie in nur wenigen Stunden, sicher zu stehen, zu springen und mit über 30 km/h zu laufen. Das ist auch notwendig, denn das Ziel lautet: Überleben. Raubtiere, die auf Beutezug sind, haben ein Gespür dafür entwickelt, die schwächsten Tiere in einer Herde zu identifizieren. Die Jungtiere haben also nur dann eine Überlebenschance, wenn sie mit den anderen mithalten können. Interessant ist in diesem Zusammenhang das verwendete Lernverfahren. Denn auch ohne Lehrer bekommt die Antilope eine Rückmeldung, ob der Versuch zu stehen oder zu laufen erfolgreich war oder nicht. Sie probiert zunächst einige Aktionen aus und folgt dann der Strategie, die erfolgsversprechend zu sein scheint. Diese Form von Lernen wird auch bestärkendes Lernen (*Reinforcement Learning*) genannt.

Im Bereich der Künstliche Intelligenz beschäftigt sich das maschinelle Lernen damit, Lernvorgänge der Natur auf Maschinen zu übertragen. *Reinforcement Learning* kann bspw. dazu benutzt werden, dass Roboter laufen lernen. Aber auch Computersysteme können damit trainiert werden. So hat das DeepMind-Team von Google wirkungsvoll demonstriert, wie ein Computersystem das Atari-Spiel Breakout lernt. Anstelle der Sensoren bekommt das System die Bildschirmausgabe des Computerspiels als Pixelgrafik und den aktuellen Spielstand mitgeteilt. Einen Joystick braucht das System nicht zur Steuerung, denn die digitalen Signale, die der Joystick generieren würde, können auch direkt an das Spiel als Aktionen übertragen werden. Im Durchschnitt erzielte der Algorithmus 168 Punkte, während menschliche Spieler nur auf 31 Punkte kamen [Mni+13].

Das Konzept zu *Reinforcement Learning* lässt sich verallgemeinern und auf viele Probleme anwenden. Genaugenommen ist RL keine Methode, sondern es sind spezielle Arten von (Lern-)Problemen. Im Mittelpunkt dieser Probleme steht ein Softwareagent (vgl. Kap. 3), der autonom mit seiner Umwelt interagiert, um ein bestimmtes, vorgegebenes Ziel zu erreichen. Der Agent macht Beobachtungen, trifft eigene Entscheidungen und führt Aktionen aus, die seinen Umweltzustand ändern. Er bekommt Rückmeldungen in Form von Belohnungen (*Rewards*) und versucht eine Strategie bzw. Politik (*Policy*) zu finden, um das vorgegebene Ziel zu erreichen. Er kann Erfahrungen sammeln (*Exploration*) oder das bislang gelernte Wissen direkt anwenden, um kurzfristig Belohnungen zu erhalten (*Exploitation*). Um das langfristige Ziel zu erreichen und einen möglichst hohen *Return* zu bekommen, muss eine Kombination aus *Exploration* und *Exploitation* eingesetzt werden. Viele wissenschaftliche Arbeiten zu diesem Gebiet stammen von Richard S. Sutton und Andrew G. Barto und sind in deren RL-Standardwerk zusammengetragen [SB98].

Mathematisch können viele RL-Lernprobleme durch ein Markov Entscheidungsprozess (*Markov Decision Process*) beschrieben werden, bei dem die Bellman-Gleichung der Ausgangspunkt bei der Entwicklung von Lernalgorithmen ist. Dabei werden Zustände über spezielle Q-Wertefunktionen bewertet und eine Strategie gesucht, möglichst optimale Zustände zu besetzen. Bekannte Algorithmen sind bspw. Q-Learning und SARSA. In sehr großen Zustandsräumen werden die Wertefunktionen geschätzt bzw. durch Funktionen approximiert. Künstliche Neuronale Netzwerke sind ebenfalls im Bereich Maschinenler-

nen sehr populär und können u.a. zu einer solchen Funktionsapproximation verwendet werden. Mittlerweile lassen sich auch sehr tiefe KNN erfolgreich trainieren und man spricht von *Deep Learning*. Dieser Durchbruch hat auch dem *Reinforcement Learning* einen weiteren Schub gegeben und es sind bspw. Algorithmen wie Deep Q-Network (DQN), DeepSARSA und Varianten davon entwickelt worden.

Das Ziel dieses Teilprojekts war es, eine geeignete *Open Source* RL-Softwarelösung zu finden, die sowohl in der Lehre als auch in der angewandten Forschung an der htw saar eingesetzt werden kann. Auf der Plattform GitHub wurden am 02.02.2018 bereits 5.307 Projekte zu diesem Thema gefunden. Mit über 60 Prozent sind die meisten dieser Lösungen in der Programmiersprache Python programmiert. Die Architektur einer modernen RL-Software besteht aus den lose gekoppelten Komponenten Umgebung, Agent bzw. Algorithmus und Experiment.

Insgesamt wurden 37 Softwarelösungen zum Thema *Reinforcement Learning* betrachtet und vorgestellt: 10 Umgebungen, 13 Bibliotheken bzw. Frameworks, 8 Komplettpakete bzw. Speziallösungen sowie 6 Tutorials. Die 10 Umgebungen unterscheiden sich im Wesentlichen im Umfang und Art der Problemstellungen. Dieses Spektrum reicht von klassischen RL-Problemen, über Brettspiele (Go etc.) hin zu Computerspiele (Atari, Nintendo, Flash usw.) und sogar 3D-Egoshooter oder Echtzeit-Simulatoren zum autonomen Fahren. Diese Softwarelösungen besitzen definierte Schnittstellen, um vom Agenten Aktionen entgegenzunehmen, und um die aktuellen Umweltzustände und *Rewards* an den Agenten zurückzuliefern. Die 13 nächsten Lösungen sind entweder eine Sammlung von Algorithmen zur Lösung von RL-Problemen oder ein Rahmenwerk, mit dem sich sehr einfach RL-Experimente ausführen lassen. Vor dem Hintergrund, den DL-Rechner als Testsystem (vgl. Kap. 2) zu benutzen, und der Anforderung, eine möglichst modulare Lösung zu verwenden, wurden diese Bibliotheken und Frameworks genauer analysiert. Es stellte sich heraus, dass die Umgebung *OpenAI Gym* von allen unterstützt wird und 11 Lösungen ein API für Python anbieten. Eine Nutzwertanalyse mit 5 Kategorien (Funktionalität, Popularität, Aktualität, Reputation und Dokumentation) wurde durchgeführt. Dabei konnten die Lösungen TensorLayer, TensorForce, Coach und Ray RLlib am meisten überzeugen. Leider konnten TensorForce und Coach nicht erfolgreich auf dem Testsystem installiert werden. Aus diesem Grund wurden auch noch die übrigen Lösungen ausprobiert. Nur Keras-RL, OpenAI Lab und TensorFlow Agents konnten zusätzlich installiert und fehlerfrei ausgeführt werden. Im nächsten Schritt wurden die Dokumentationen und Quelltexte der verbliebenen Softwarelösungen untersucht und typische Experimente durchgeführt. Hierzu kam die Umgebung *OpenAI Gym* mit dem Problem *Cart Pole* zum Einsatz.

Am meisten konnte die Bibliothek Ray RLlib des Frameworks Ray der kalifornischen Universität Berkeley überzeugen. Die Dokumentation und Quelltexte machen einen guten Eindruck. Ein vorbereitetes Python-Skript kann benutzt werden, um schnell erste Experimente durchzuführen. Mit Hilfe selbst-programmierter Python-Skripte wurden mehrere Experimente im *Jupyter Notebook* durchgeführt, wobei sich die Agenten leicht austauschen lassen. Insgesamt wurden 6 Experimente zu den folgenden Agenten bzw. Algorithmen durchgeführt: A3C, ES, PPO, DQN, DDQN und Dueling-DQN. Hier konnte PPO in nur 1:39 Minuten ein perfektes Ergebnis erzielen und das Problem lösen. Die Softwarelösung Ray RLlib wird somit für die Behandlung von *Reinforcement Learning* Probleme empfohlen.

5 Künstliche Neuronale Netzwerke und Deep Learning

Das menschliche Gehirn ist ein Wunder der Natur. Es besteht aus etwa 100 Milliarden (10^{11}) miteinander vernetzten Nervenzellen, den Neuronen. Etwa 10% der Neuronen dienen der Eingabe und Ausgabe. Die restlichen 90% sind mit anderen Neuronen verknüpft, die Informationen speichern oder bestimmte Umwandlungen des Signals vornehmen, das sich durch das Netzwerk fortpflanzt. Neuronen sind komplexe Zellen, die auf elektrochemische Signale (Neurotransmitter) reagieren. Die Taktfrequenz des biologisch neuronalen Netzes liegt im unteren kHz-Bereich und ist damit um mehrere Dimensionen kleiner als die Geschwindigkeit der Prozessoren eines konventionellen Computersystems. Die Leistungen des menschlichen Gehirns beruhen daher in erster Linie auf der hohen Parallelität bei der Informationsverarbeitung. Neuronen sind nämlich über eine Vielzahl von Synapsen (ca. 100 Billionen, also 10^{14}) miteinander verknüpft. Synapsen können wachsen, verkümmern oder ganz verschwinden. Diese Wachstumsprozesse sind für Gedächtnis und Lernen verantwortlich.

Künstliche Neuronale Netzwerke sind nach diesem Vorbild der Natur konstruiert. Die Basis bildet das künstliche Neuron als kleinste Struktureinheit. Es besteht aus mehreren Eingabeleitungen, die mit Gewichten versehen sind, einem Berechnungskörper und einer Ausgabeleitung. Die Eingabesignale werden mit den Gewichtsfaktoren multipliziert und zu einem Netto-Eingabesignal aufsummiert. Eine Aktivierungsfunktion wandelt dieses Zwischenergebnis dann in ein Ausgabesignal um. Das Perzeptron ist ein sehr einfaches KNN, das nur Schwellenwertentscheidungen treffen kann. Wenn das Netto-Eingabesignal über einem Schwellenwert (*Bias*) liegt, dann ist das Neuron aktiviert und es feuert als Ausgabe die Zahl 1, ansonsten die Zahl 0. Das *Multilayer Perceptron* ist ein KNN, das aus vielen solchen Perzeptronen besteht. Es ist ein vollständig vernetztes vorwärtsgekoppeltes Netzwerk, in denen die Neuronen in mehreren Schichten (*Layers*) angeordnet sind: Eingabeschicht, verborgene Schichten, Ausgabeschicht. Es kann mit dem *Backpropagation*-Algorithmus trainiert werden. Lernen bedeutet, dass die Parameter des Netzwerks, also die Gewichte der Verbindungen und die Schwellenwerte der Neuronen, automatisch angepasst werden. Hierzu bekommt das Netzwerk Trainingsdaten zum Verarbeiten und für jeden Datensatz werden die Ausgabesignale des Netzwerks berechnet. Diese werden dann mit den bekannten, tatsächlichen Ergebnissen (*Teaching Inputs*) verglichen, deshalb nennt man diese Art des Lernens auch überwachtes Lernen (*Supervised Learning*). Eine Kosten- bzw. *Loss*-Funktion bewertet dann die Abweichungen dieser Werte zueinander. Diese Abweichungen, auch Delta genannt, werden dann schichtweise rückwärts durch das Netzwerk propagiert, um die Parameter zu aktualisieren. Mathematisch betrachtet handelt es sich um ein Optimierungsproblem, denn die Summe aus diesen Abweichungen soll minimiert werden. Hierzu werden die ersten partiellen Ableitungen nach den Parametern berechnet, die sogenannten Gradienten. Man spricht deshalb auch von einem Gradientenabstiegsverfahren, weil die Parameteranpassung in Richtung des steilsten Gradienten erfolgt. In einem "Fehlergebirge" ist das der direkte Weg, den man nehmen muss, um möglichst schnell ins "Tal" (entspricht dem Minimum) zu kommen. Das Gradientenabstiegsverfahren ist aber kein perfektes Verfahren, es muss nicht unbedingt im globalen

Minimum konvergieren. Je nach "Fehlergebirge" und der Wahl der Schrittweite (Lernrate) können typische Probleme wie flache Plateaus, Oszillationen, Steckenbleiben in lokalen Minima usw. auftreten. Netzwerke mit vielen Parametern in vielen verborgenen Schichten können durch das Training zwar zu einer sehr guten Anpassung kommen, ggf. dann aber schlecht generalisieren. Werden dem Netzwerk also neue Daten präsentiert, dann sind die Leistungen unbefriedigend, weil es nur auswendig gelernt hat und das Wissen nicht anwenden kann. Man spricht auch von einer Überanpassung (*Overfitting*). Werden die Netzwerke immer größer und tiefer, dann kommt noch das Problem der verschwindenden Gradienten hinzu. Aufgrund sehr vieler Matrix-Multiplikationen von kleinen Zahlen, entsprechend den Gewichten der Verbindungen, gerät das Training bei Anwendung des Gradientenabstiegsverfahrens ins Stocken.

Tiefe KNN werden verwendet, um große Datenmengen (vereinfacht: *Big Data*) besser verarbeiten zu können. Um die beschriebenen Probleme zu vermindern bzw. zu lösen, werden Techniken des *Deep Learning* eingesetzt. Hierzu wurden u.a. neuartige KNN-Architekturen entwickelt, Neuronen mit speziellen Aktivierungsfunktionen (ReLU) verwendet, grafische Prozessoren (GPUs) zur schnellen Matrix-Multiplikation eingesetzt usw. Viele wichtige Vorarbeiten sind in dem Standardwerk *Deep Learning* von Ian Goodfellow, Yosuah Bengio und Aaron Courville zusammengefasst [GBC16].

Das *Convolutional Neural Network (CNN)* ist ein Netzwerk mit dem visuellen Cortex als biologisches Vorbild. Es wird deshalb auch häufig im Bereich Bilderkennung eingesetzt. Durch die Kombination von sogenannten *Convolutional Layer* und *Pooling Layer* werden die Bildinformationen schrittweise abgetastet, um daraus Stufe für Stufe *Features* zu generieren. Im Fall der Gesichtserkennung würden in der ersten Stufe zunächst Kanten im Bild identifiziert werden, also Bereiche in denen es große Helligkeitsunterschiede gibt. In der zweiten Stufe könnten Merkmale wie Augen, Nase, Mund, Ohren usw. identifiziert werden. In der letzten Stufe würden dann die kompletten Gesichter als Merkmale betrachtet werden. Das CNN generiert diese *Features* allein, d.h. ohne fremde (menschliche) Hilfe. Das Lernen erfolgt somit nicht-überwacht (*Unsupervised Learning*).

Rekurrente bzw. rekursive Netzwerke werden dagegen häufig im Bereich Sprach- und Texterkennung eingesetzt. Das *Recurrent Neural Network (RNN)* ist ein Netzwerk mit Rückkopplungen. In diesem werden die Signale nicht nur von den Eingabeneuronen über die verdeckten Neuronen zu den Ausgabeneuronen transportiert, sondern auch in entgegengesetzter Richtung. Das biologische Pendant ist der Neocortex mit den höheren Gehirnfunktionen wie Motorik oder Sprache, der auch eine wichtige Rolle für das Gedächtnis spielt. Die Eingabedaten werden bei diesen Netzwerken sequenzweise verarbeitet, wobei eine Sequenz bspw. einen Satz aus Wörtern darstellen kann. Einheiten mit Verbindungen auf sich selbst, also mit einfachen Rückkopplungen, lassen sich hinsichtlich dieser Sequenz wie ein ausgerolltes Netzwerk betrachten und mit einem *Backpropagation*-Algorithmus trainieren, den man nun *Backpropagation Through Time* nennt. Um auch hier dem Problem der verschwindenden Gradienten entgegenzuwirken, wurde die Einheit *Long Short-Term Memory (LSTM)* entwickelt. Diese besitzt eine innere Struktur aus einer Zelle und drei *Gates*, um den Signalfluss zu steuern und eine Art Gedächtnis abzubilden. Eine Variante davon ist die *Gated Recurrent Unit (GRU)*, die zwar etwas einfacher aufgebaut, aber zu ähnlich erfolgreichen Leistungen fähig ist.

Das Ziel dieses Teilprojekts war es, typische Anwendungen durchzuführen, bei denen Künstliche Neuronale Netzwerke bzw. Techniken des *Deep Learning* zum Einsatz kom-

men, die dann wiederum als Basis in der Lehre und angewandten Forschung an der htw saar dienen können. Hierzu wurden zunächst 22 aktuell verfügbare *Open Source* Softwarelösungen zum Thema *Deep Learning (DL)* betrachtet, die auf der Internet-Plattform *GitHub* vorhanden sind. Die mit Abstand populärste DL-Bibliothek ist TensorFlow, die von Mitarbeitern des Google Brain Teams entwickelt wurde. Mathematische Operationen mit Tensoren werden graphenbasiert formuliert. Typische Operationen wie Matrix-Multiplikationen, die beim Lernverfahren ausgeführt werden müssen, lassen sich so einfach formulieren und durch das Verwenden von grafischen Prozessoren auch sehr effizient auf diesen parallelisiert ausführen. TensorFlow benutzt hierfür bspw. CUDA von Nvidia. TensorFlow bietet zwar bereits eine Schnittstelle zur Programmiersprache Python an, jedoch steht mit der *High-Level*-API Keras eine Erweiterung zur Verfügung, die sehr viele Programmier-Bausteine enthält, die leicht eingesetzt und wiederverwendet werden können. Als *Backend* könnte theoretisch auch das populäre Theano benutzt werden. Diese von der Universität Montréal entwickelte Lösung wird aber seit 2018 nicht mehr weiterentwickelt. Somit wurde die Kombination der Softwarepakete TensorFlow und Keras verwendet, um drei typische DL-Anwendungen durchzuführen.

Die erste Anwendung stammt aus dem Bereich Bilderkennung und Objektklassifikation. Die Datenbank MNIST enthält 70.000 Datensätze von Graustufen-Bildern im einheitlichen Format 28 x 28 Pixel zu handgeschriebenen Ziffern null bis neun. Das Ziel ist es also, ein Modell zu trainieren, das die jeweilige Ziffer automatisch identifizieren kann. Somit stellt diese Anwendung einen Schritt im Prozess *Optical Character Recognition* dar. Zwei KNN wurden hierzu mit 60.000 Bildern trainiert, ein klassisches MLP mit der Topologie 784-256-128-10 und ein modernes CNN mit einer komplexeren Topologie. Das Training wurde auf der Grafikkarte Geforce GTX 1080 Ti von Nvidia ausgeführt. Nach 12 Sekunden und 16 Epochen erreichte das MLP eine Genauigkeit von 98% auf den 10.000 Bildern der Testdaten. Das CNN kam sogar auf eine Genauigkeit von 99,35%, wobei 20 Epochen in 41 Sekunden trainiert wurden. Das sind bereits hervorragende Klassifikationsergebnisse.

Auch die zweite Anwendung ist dem Bereich Bilderkennung zuzuordnen. Diesmal standen mit der CIFAR-10-Datenbank eine Sammlung von 60.000 Farb-Bildern mit der Größe 32 x 32 Pixel zur Verfügung, auf denen zehn Klassen von Objekten zu erkennen sind: Flugzeuge, Autos, Vögel, Katzen, Rehe, Hunde, Frösche, Pferde, Schiffe und LKW. Als Modell wurde ein komplexes CNN mit 822.570 freien Parametern verwendet und 47 Epochen in etwas über 6 Minuten trainiert. Die Genauigkeit betrug knapp 82%. Das ist sehr ordentlich, wenn man bedenkt, dass diese Anwendung viel komplizierter als das Erkennen der Ziffern ist.

Anwendung Nummer Drei ist eine Stimmungserkennung (*Sentiment Analysis*), d.h. sie gehört in den Bereich *Text Mining*. Als Datenbasis werden 50.000 Nutzer-Kritiken zu Filmen der Internetplattform IMDb verwendet. Diese unstrukturierten Text-Rezensionen sind sehr eindeutig hinsichtlich einer positiven oder negativen Kritik. Letztendlich wird in dieser Analyse auch wieder eine Klassifikation durchgeführt, diesmal eine binäre, weil nur zwei Klassen von Rezensionen unterschieden werden. In dieser Anwendung mussten die Daten zunächst vorverarbeitet werden. Die Wort-Sequenzen unterschiedlicher Länge der Rezensionen enthielten bereits ganze Zahlen für die verschiedenen Wörter, die nach der auftretenden Häufigkeit sortiert waren. Jede Sequenz wurde zunächst auf einen Vektor der Dimension 16 abgebildet, der kontinuierliche Werte enthält (*Word Embedding*). Es wurden dann vier verschiedene KNN mit 25.000 Rezensionen trainiert und dabei folgende Ergebnisse auf der Testmenge der anderen 25.000 Kritiken erzielt. Mit dem MLP der

Topologie 256-16-1 konnte nach 38 Epochen bzw. 10 Sekunden eine Genauigkeit von 86% erzielt werden. Das CNN schaffte nach 56 Epochen in knapp 30 Sekunden 87%. Für das Training der LSTM-Einheiten wurden bereits 3 Minuten benötigt, wobei nach den 126 Epochen ebenfalls eine Genauigkeit von 87% erreicht wurde. Schließlich kam das GRU-Netzwerk in der Hälfte der Zeit und 72 Epochen auf eine Genauigkeit von 87,4%.

Mit nur wenigen Zeilen Python-Quelltext war es möglich, verschiedene Anwendungen mit Hilfe von mehreren KNN zu bearbeiten. Dabei wurden die Daten vorbereitet, die Modelle konfiguriert, trainiert und evaluiert. Das Training auf der *High-End*-Grafikkarte von Nvidia verlief schnell und die Modelle zeigten bereits gute Genauigkeiten. Die Kombination aus TensorFlow und Keras ist somit eine gute Basis für den Einsatz in der Lehre und angewandten Forschung an der htw saar. Das Ziel dieses Teilprojekts wurde also erreicht.

6 Ausblick

In den vier Teilen dieses Forschungsprojekts wurden die Grundlagen zu den KI-Themen Multiagentensysteme, *Reinforcement Learning*, Künstliche Neuronale Netzwerke und *Deep Learning* gelegt, passende *Open Source* Softwarelösungen evaluiert, ausgewählt und damit erste Anwendungen erfolgreich durchgeführt. Die Ziele wurden erreicht, aber in jedem dieser Teile lassen sich noch Dinge verbessern und offene Fragen untersuchen.

Einige Möglichkeiten des Multiagentensystems JADE wurden noch nicht ausgeschöpft. Hierzu gehört bspw. das Verschieben von Agenten auf andere Plattformen, insbes. auf mobile Endgeräte wie Smartphones. Die Agenten können in ihren Nachrichten auch Ontologien (vereinfacht: gemeinsames Vokabular) in Form der *Web Ontology Language (OWL)* verwenden und hierzu das *Resource Description Framework (RDF)* einsetzen, welches auf *Extensible Markup Language (XML)* basiert. Die Kommunikation zwischen den Agenten könnte noch wesentlich komplexer gestaltet werden und auch die Zusammenarbeit lässt sich genauer untersuchen. Des Weiteren sollte auch das BDI-Konzept eingesetzt werden. Hierfür kann bspw. die Erweiterung Jadex benutzt werden. Schließlich sollen die Agenten selbst möglichst intelligent im Sinn der schwachen KI werden. Dabei könnten spezielle Lernverfahren wie *Reinforcement Learning* und *Deep Learning* integriert werden.

Die Softwarelösung Ray RLlib wurde für die Behandlung von *Reinforcement Learning* Probleme ausgewählt. Mit den meisten Agenten bzw. Algorithmen konnte das *Cart Pole* Problem gelöst werden, jedoch nicht mit dem DQN-Agenten. Mit Ray Tune verfügt das Framework Ray über einen Hyperoptimierer. Dies ist ein Werkzeug, um eine optimale Konfiguration der Agenten zu finden. Es könnte also benutzt werden, um das unbefriedigende Abschneiden des DQN-Agenten genauer zu analysieren. Außerdem sollten weitere RL-Probleme studiert werden. Andererseits konnten einige Softwarelösungen wie bspw. Coach gar nicht getestet werden, weil während der Installation Fehler aufgetreten sind. Coach ist aber sehr interessant, denn diese Lösung hat sehr viele Algorithmen implementiert und stellt ein sehr modernes, Komponentenbasiertes Framework dar.

Die Kombination aus TensorFlow und Keras bietet sich an, um mit nur wenigen Zeilen Python-Quelltext verschiedene KI-Anwendungen mit Hilfe von KNN und DL zu bearbeiten, d.h. Modelle zu konfigurieren, zu trainieren und zu evaluieren. Die bisher eingesetzten Netzwerke können sicherlich noch bezüglich der Genauigkeit auf den Testdaten verbessert werden. Optimierungstechniken wie Kreuz-Validierung und Regularisierung kamen bislang noch nicht zum Einsatz. Hinsichtlich der Topologie und den Trainingseinstellungen lassen sich ebenfalls Hyperparameterstudien durchführen. Neben den drei durchgeführten Beispielanwendungen aus den Bereichen Bild- und Texterkennung könnten auch noch weitere Anwendungen untersucht werden, um die Kenntnisse zu vertiefen. Bspw. stellt die *Data Science* Plattform Kaggle in zahlreichen Wettbewerben Daten zur Verfügung, die sich mittels Techniken des Maschinenlernens bearbeiten lassen.

Eine weitere Idee besteht darin, die einzelnen Softwarelösungen in eine Gesamtlösung zu integrieren. Damit können dann auch sehr komplexe Probleme im Bereich des *Internet of Things* bearbeitet werden. Die Basis für eine erfolgreiche Integration ist das Multiagen-

tensystem JADE. Dieses ist in der objektorientierten Programmiersprache Java entwickelt worden. In den anderen Bereichen ist jedoch Python die dominante Programmiersprache und die empfohlenen Lösungen Ray RLlib, TensorFlow und Keras bieten zu Python eine Programmierschnittstelle (*Application Programming Interface (API)*) an. Es gibt allerdings mit DL4J und der Erweiterung RL4J auch Softwarelösungen, die eine Java-API bereitstellen. Somit sollten diese beiden Lösungen ebenfalls betrachtet und im Hinblick auf eine Integration in JADE evaluiert werden. Eine zweite Alternative wäre die Verwendung einer weiteren Technologie, um die Java- und Python-Welten miteinander zu verbinden. Jython ist eine Java-Implementierung der Programmiersprache Python, die das Ausführen von Python-Programmen auf Java-Plattformen ermöglicht. Diese Brückentechnologie könnte also das Integrationsproblem lösen. Falls nicht, müssten als dritte Alternative Schnittstellen definiert und diese in JADE implementiert werden, damit eine Zusammenarbeit mit Python-Agenten möglich ist. Die Lösung Keras bietet mit Keras-RL auch eine Erweiterung zum *Reinforcement Learning* an, die statt Ray RLlib verwendet werden könnte. Dadurch würde es weniger Schnittstellen zwischen den Systemen geben. Auch diese Lösung sollte also nochmals bezüglich der Anforderung Integrationsfähigkeit evaluiert werden.

Betrachtet man die magischen Quadranten aus dem Jahr 2018 zum Thema *Data Science* und *Maschine Learning* des Marktforschungs- und Beratungsunternehmens Gartner (siehe Abb. 6.1), so fällt auf, dass keine, der in diesem Forschungsprojekt empfohlenen Softwarelösungen, dort aufgeführt ist.

Abbildung 6.1: Gartners magische Quadranten zu Data Science und Maschine Learning [Ido+18]

Das liegt sicherlich daran, dass in diesem Forschungsprojekt der Fokus auf Themen wie *Reinforcement Learning*, KNN und *Deep Learning* gelegt wurde und es in diesem Bereich momentan nur Softwarelösungen gibt, in denen der Anwender noch programmieren muss – vorzugsweise in Python. Das könnte sich aber demnächst ändern. Die *Data Science* Software *KNIME Analytics Platform*, die in den magischen Quadranten als eine der führenden Lösungen angesehen wird, wird bereits seit mehreren Jahren in der Fakultät für Wirtschaftswissenschaften (WiWi) der htw saar erfolgreich in der Lehre eingesetzt. Diese Software, die ursprünglich an der Universität Konstanz entwickelt wurde, bietet die Möglichkeit, grafische Workflows zu erstellen, um sehr schnell und benutzerfreundlich Aufgaben im Bereich *Data Science* zu bearbeiten (siehe Abb. 6.2).

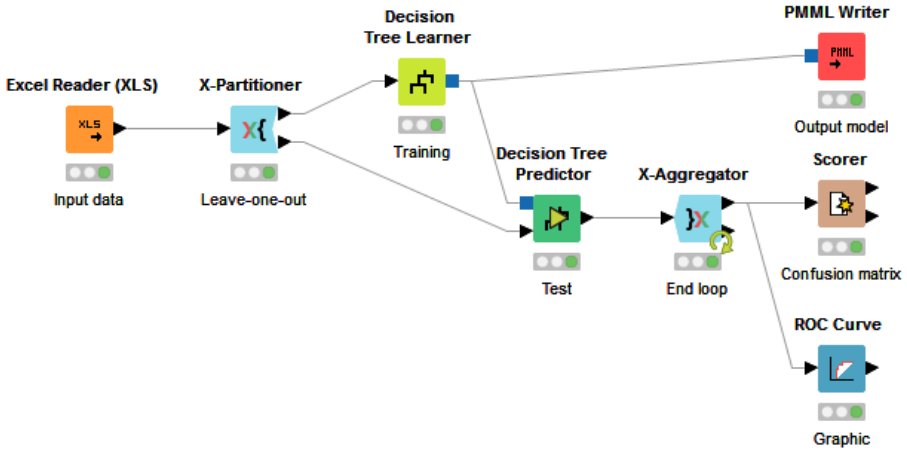

Abbildung 6.2: Workflow in der KNIME Analytics Platform

Der Anwender muss dabei nicht über Programmierkenntnisse verfügen. Mittlerweile gibt es zahlreiche Erweiterungen für KNIME, sogenannte *Extensions*, mit denen sich auch andere Softwarebibliotheken einbinden und nutzen lassen. Zum Thema *Deep Learning* werden Erweiterungen zu DL4J und Keras angeboten [KNI18]. Diese Erweiterungen sollten zukünftig im Hinblick auf den Einsatz in der Lehre geprüft werden.

Das Thema *Data Science* wird derzeit an der Fakultät WiWi der htw saar in den folgenden Master-Modulen gelehrt:

- Angewandte Methoden der Informationsbeschaffung (MMF-130)
- Angewandte Informatik (DFMMS-222)
- Big Data Analysis (MAMS-120)
- Data Science (MASCM-141)

In diesen Modulen wird auch das Werkzeug KNIME eingesetzt. Zukünftig könnten also in diesen Modulen ggf. auch KI-Anwendungen mit Hilfe der Erweiterungen DL4J und Keras bearbeitet werden, vorausgesetzt auch die Hardware wird nachgerüstet (s.u.).

Die in diesem Forschungsprojekt behandelten Themen sind aber auch insbes. für Studierende des Schwerpunkts Wirtschaftsinformatik des Bachelor-Studiengangs Betriebswirtschaft interessant. Im 5. und 6. Semester vertiefen sich die Studierenden in zwei von fünf Schwerpunkten. Vier Module der Vertiefung Wirtschaftsinformatik werden derzeit angeboten. KI-Themen sind aber noch kein fester Bestandteil in dieser Vertiefung. Im 5. Semester werden bereits Grundlagen der Programmierung vermittelt, und zwar am Beispiel der objektorientierten Programmiersprache Java (Modul BBWL-571 Software En-

gineering). Der Umstieg auf die Skriptsprache Python ist in kurzer Zeit machbar, wenn bereits Java-Kenntnisse vorhanden sind. Es sollte aber auch darüber nachgedacht werden, direkt zur Programmiersprache Python zu wechseln. Im Rahmen von Projektarbeiten (Modul BBWL-622) könnten dann KI-Themen durch Studierende bearbeitet werden. Für ein solches Pilotprojekt würde ein PC mit einer *High-End*-Grafikkarte als Ausstattung benötigt werden. Die Kosten halten sich also in überschaubare Grenzen. Die Installation und Konfiguration des Rechners könnten die Studierenden auch selbst bewerkstelligen. Eine entsprechende Dokumentation ist bereits vorhanden (vgl. Kap. 2).

Generell sollte überlegt werden, ob es sinnvoll ist, ein Labor der Fakultät WiWi als KI-Labor aufzurüsten. Hierzu müssten die Rechner in diesem Labor ebenfalls mit geeigneten Grafikkarten ausgestattet werden. Die Kosten hierfür bewegen sich im Bereich von ca. 30.000 Euro. Problematisch ist allerdings, dass in diesem Forschungsprojekt Linux als Betriebssystem verwendet wurde und an der Fakultät WiWi flächendeckend Windows eingesetzt wird. Das notwendige Wissen im Bereich Linux-Administration ist nur teilweise innerhalb des dezentralen IT-Service-Teams der Fakultät WiWi vorhanden. Somit müssten die benötigten Softwarelösungen (CUDA, cuDNN, Ray RLlib TensorFlow, Keras usw.) unter dem Betriebssystem Windows installiert und getestet werden.

Neben inhaltlichen Änderungen der bestehenden Module des Studienangebots der Fakultät WiWi, sollte zukünftig auch über die Schaffung neuer Angebote nachgedacht werden. Dies könnten neue Module aber auch neue Studiengänge sein, die Themen wie Digitalisierung und Künstliche Intelligenz nicht nur aufgreifen, sondern als zentrales Bausteine begreifen.

Eine weitere Verwendungsmöglichkeit der Ergebnisse dieses Forschungsprojekts ist die angewandte Forschung. In den vier Anlagen wurden hierzu bereits einige Ideen skizziert, die in Tab. 6.1 zusammengestellt sind.

Nr	Thema	Bereich	Quelle
1	Dynamic Packaging mit Agenten	Tourismus	[Sel18a]
2	Smart Home: AAL	Pflege und Gesundheit	[Sel18a]
3	Chatbots im Recruiting	Personalmanagement	[Sel18d]
4	Intelligente Wirtschaftsprüfung	Rechnungs- und Prüfungswesen	[Sel18d]
5	Spieltheorie mit RL	BWL und VWL	[Sel18c]
6	Multi-Channel CRM	Marketing	[Sel18c]
7	Soziale Netzwerke: Hasskommentare	Wirtschaftsinformatik	[Sel18b]
8	Predictive Maintenance	Produktion und Logistik	[Sel18b]
9	IT-Security: IDS	Informatik	[Sel18b]

Tabelle 6.1: Ideen zu zukünftigen KI-Forschungsprojekten

Die Anwendungsmöglichkeiten sind vielfältig und die Forschungsprojekte interdisziplinär. Die meisten davon könnten auch in Zusammenarbeit mit Unternehmen, insbes. KMU aus dem Saarland, durchgeführt werden, zu denen die htw saar sehr gute Kontakte pflegt (z.B. Orbis AG, Data One GmbH, SAP St. Ingbert, DHC Dr. Herterich & Consultants GmbH, T-Systems usw.). Das Zentrum Mittelstand Saar (ZMS) kann hier ebenfalls als Schnittstelle von Wissenschaft und Innovation involviert werden. Mit der FITT gGmbH

steht außerdem noch das Institut für Technologietransfer der htw saar als weiterer potenzieller und kompetenter Partner zur Verfügung.

In einer aktuellen Studie vom April 2018 der Unternehmens- und Strategieberatung McKinsey & Company wurden mehr als 400 spezifische Anwendungsfälle von 19 Branchen bezüglich des Einsatzpotenzials moderner KI-Techniken analysiert [Chu+18]. Die Analysten kommen zu dem Schluss, dass zukünftig in allen Branchen hohe Wertschöpfungen durch den Einsatz dieser KI-Techniken zu erwarten sind, insbes. im Handel, in der Logistik und im Tourismus (siehe Abb. 6.3).

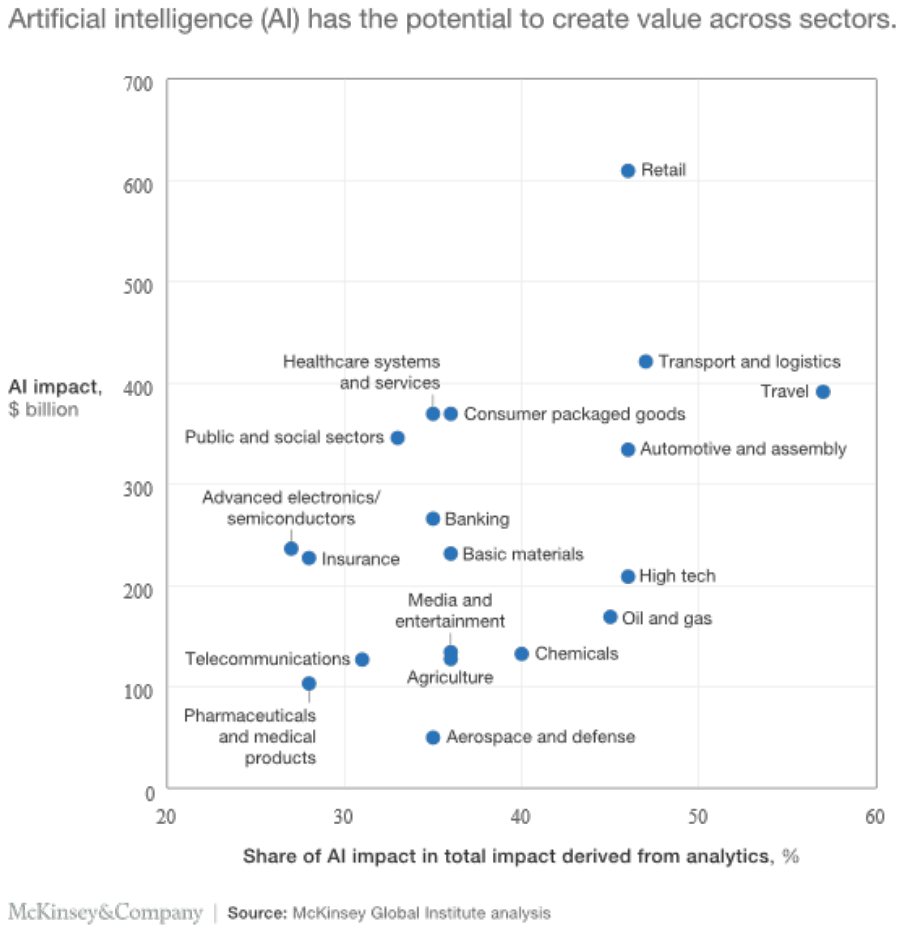

Abbildung 6.3: Einfluss moderner KI-Techniken auf verschiedene Branchen [Chu+18]

Die modernen KI-Techniken, die in dieser Studie betrachtet wurden (siehe Abb. 6.4), sind im Wesentlichen auch diejenigen, die im Rahmen dieses Forschungsprojekts analysiert wurden, nämlich *Reinforcement Learning* und *Deep Learning*. Allerdings wird in der McKinsey-Studie auch das sogenannte *Transfer Learning* betrachtet. Dabei wird ein KI-Modell anwendungsspezifisch trainiert und dann auf ein neues, unterschiedliches Problem angesetzt. Zukünftig sollte also auch in dieser dritten Domäne Wissen aufgebaut werden, die als nächster Stufe des Maschinenlernens angesehen wird [Rud17].

Abbildung 6.4: KI-Techniken [Chu+18]

Mit Sorge wird allerdings die Entwicklung der Verteilung der weltweiten Forschungs- und Anwendungsaktivitäten in diesem Bereich gesehen. Hierzu wurden allerdings keine quantitativen Analysen durchgeführt, insofern kann keine wissenschaftlich fundierte Aussage getroffen werden. Nichtsdestotrotz soll an dieser Stelle der subjektive Eindruck festgehalten werden: Sehr viele, der in den letzten Jahren veröffentlichten wissenschaftlichen Abhandlungen zu den KI-Themen, die in diesem Forschungsprojekt betrachtet wurden, stammen aus den USA oder aus China, nicht aber zwangsläufig von Mitarbeitern aus Universitäten oder öffentlichen Forschungseinrichtungen, sondern vielmehr von Angestellten aus den Forschungs- und Entwicklungsabteilungen großer Technologieunternehmen. Positiv ist, dass Unternehmen inzwischen ihren Mitarbeitern erlauben, Forschungsergebnisse zu publizieren. Negativ ist dagegen, dass die Forschung an den Hochschulen scheinbar zurückgedrängt wird. Exemplarisch sei in diesem Zusammenhang auch das Beispiel *Theano* genannt: Diese bislang populäre Software-Bibliothek für *Deep Learning* der Universität Montréal wird seit 2018 aufgrund starker Konkurrenz aus der Industrie nicht mehr weiterentwickelt. Genau so hat es Pascal Lamblin, einer der Theano-Entwickler, in seiner öffentlichen Nachricht am 28.09.2017 formuliert [Lam17]. Dies ist offensichtlich eine Anspielung auf das Produkt TensorFlow, das vom Google Brain Team entwickelt wurde, und eine dominante Stellung innerhalb der DL-Softwarelösungen einnimmt.

In der Einleitung wurde der neue Google-Dienst Duplex beschrieben. Bspw. telefoniert dieser digitale Assistent mit einem Friseursalon, um einen Termin zu vereinbaren. Duplex klingt dabei wie ein Mensch, insbes. auch deshalb, weil die Maschine hin und wieder Füllwörter wie „hmm“ oder „äh“ in dem Gespräch einstreut [Beu18]. Direkt nach der Demonstration ist eine Ethik-Debatte ausgelöst worden und infolge massiver Kritik vieler Nutzer aus dem Internet hat Google inzwischen versprochen, dass sich Duplex zukünftig bei Anrufen angemessen als Maschine zu erkennen geben wird [Pre18]. Was aber passiert eigentlich, wenn in dem Friseurgeschäft ebenfalls eine Maschine das Telefongespräch entgegen nimmt? Sprechen dann beide Maschinen noch in Englisch, Deutsch oder der jeweiligen Landessprache miteinander?

Facebook hat hierzu 2017 ein Experiment durchgeführt, bei dem sich zwei Chatbots miteinander unterhalten haben [Kü17]. Die beiden Agenten Alice und Bob sollten dabei versuchen, eine bestimmte Anzahl von Objekten (z.B. zwei Bücher, einen Hut und drei Bälle) so untereinander aufzuteilen, dass jeder für sich zufrieden ist. Für jeden Agenten haben die Objekte einen anderen geheimen Wert. Sie verhandeln über die Aufteilung dieser Objekte, um schließlich zu einem Kompromiss zu gelangen. Zwei interessante Aspekte wurden bei diesem Multiagentensystem-Experiment entdeckt. Die Maschinen haben abseits der geltenden Grammatikregeln eine effizientere Ausdrucksweise entwickelt. Sie haben Sätze formuliert, die für den Menschen überhaupt keinen Sinn ergeben. Anders ausgedrückt: Sie haben eine Art Geheimsprache entwickelt. Dieses Problem lässt sich jedoch lösen, indem die Agenten eine Belohnung bekommen, wenn sie die Grammatikregeln korrekt anwenden. Ein zweites Phänomen führte dann aber zum Abbruch des Experiments. Um die Objekte zu bekommen, die für den jeweiligen Agenten besonders wertvoll sind, haben die Agenten eine Strategie gelernt, zunächst vorzugeben, dass sie eigentlich kein Interesse an diesen Objekten haben. Letztendlich konnten sie diese dann doch bekommen, und zwar zu einem wesentlich besseren Preis, als wenn sie direkt vorgaben, diese haben zu wollen. Die Agenten haben also gelernt, zu täuschen und ihre eigentlichen Ziele bzw. Absichten zu verbergen. Man könnte sie auch als schlau beschreiben, weil sie dabei so raffiniert vorgegangen sind. Das ist zwar menschlich, aber für eine Maschine auch irgendwie unmoralisch. Dieses Beispiel zeigt also bereits: Es gibt noch unbeantwortete ethische Fragen beim Umgang und Einsatz von Künstlicher Intelligenz.

Je länger man sich mit dem Thema Künstliche Intelligenz beschäftigt, desto mehr Respekt hat man vor den Leistungen des menschlichen Gehirns, die es tagtäglich vollbringt. Die KI hat derzeit ihre Stärken in sehr speziellen Anwendungsbereichen und kann in diesen den Menschen teilweise auch bereits übertrumpfen wie bspw. beim Brettspiel Go. Hierzu ist aber ein enormer technischer Aufwand notwendig. Das verteilte System von *AlphaGo* bestand aus 1.202 Prozessoren (CPUs) und 176 Grafikprozessoren (GPUs). Die einzelne Grafikkarte Nvidia Geforce GTX 1080 Ti, die im Rahmen dieses Forschungsprojekts verwendet wurde, nimmt bei Volllast bereits eine Leistung von 250 Watt auf. Zum Vergleich verbraucht das Gehirn eines erwachsenen Menschen nur etwa 20 Watt. Das Besondere daran ist, dass jeder einzelne Mensch in der Summe über unglaublich viele Fertigkeiten verfügt, die er im Lauf seines Lebens gelernt hat. Maschinen sind dagegen noch sehr weit davon entfernt, die gleichen intellektuellen Fertigkeiten zu erlangen. Eine starke Künstliche Intelligenz, die auch über soziale Intelligenz, Empathie, ein Bewusstsein etc. verfügt, ist also immer noch zurecht *Science Fiction*.

Hinsichtlich des Energieverbrauchs muss allerdings zwischen der Trainings- und der Anwendungsphase unterschieden werden. Für das Training benötigt das KI-System eine enorme Leistung, weil es aus *Big Data* lernen muss. Die Anwendung dagegen kann auf einem gewöhnlichen System erfolgen, weil nur wenige neue Daten auf einem bereits trainierten KI-System verarbeitet werden müssen. Da die meisten PCs keine *High-End*-Grafikkarten enthalten, es sei denn es sind Rechner von *Gamer*, d.h. Computer-Spielern, oder von *Miner*, die Kryptowährungen wie Bitcoin schürfen, finden die Trainingsprozesse in der *Cloud* der KI-Anbieter und nicht auf dem lokalen Rechner der Nutzer statt. Je mehr Daten zum Trainieren des KI-Systems zur Verfügung stehen, desto besser kann der Algorithmus aus diesen lernen. Selbst wenn der Nutzer über einen PC mit einer *High-End*-Grafikkarte verfügt und genügend Daten lokal zum Trainieren der KI-Anwendung vorhanden wären, gibt es aber noch einen entscheidenden dritten Grund, warum die Nutzerdaten aus Sicht der Dienstanbieter in die *Cloud* geladen werden müssen. « *Data is the new oil* » hat Clive Humby bereits 2006 festgestellt [Pal06]. Der Nutzer bezahlt also mit seinen persönlichen Daten. Diese müssen allerdings noch wie beim Öl verfeinert bzw. veredelt werden. Als Gegenleistung wird dem Nutzer dann ein intelligenter Dienst geboten, ein Daten-Produkt. Das Thema Datenschutz muss also ebenfalls bei der Entwicklung von KI-Anwendungen beachtet werden. Einen wesentlichen Unterschied zwischen Öl und Daten gibt es aber doch: Während der Rohstoff Öl auf dem Planeten Erde immer mehr zuneige geht, wächst die Menge der digitalen Daten exponentiell an.

Der Prozess der Automatisierung ist irreversibel. Durch KI-Anwendungen wird dieser Prozess sogar beschleunigt. In einem Industrieunternehmen stehen im Kern Produktionsprozesse im Fokus, die durch den Einsatz von Maschinen optimiert werden. Unabhängig von der Art und Branche des Unternehmens rücken nun auch vermehrt Dienstleistungsprozesse in den Mittelpunkt der Automatisierungsbemühungen. Approximiert man die aktuellen Leistungen der KI in den Bereichen *Reinforcement Learning* und *Deep Learning* in Richtung Zukunft, dann werden voraussichtlich also zunächst einzelne betriebliche Aktivitäten durch intelligente Computersysteme ersetzt, in denen große Datenmengen verarbeitet und analysiert werden müssen. Vereinfacht ausgedrückt werden die Aktivitäten ersetzt, in denen die Maschine besser ist als der Mensch (Qualität, Effektivität) oder in denen die Maschine günstiger ist als der Mensch (Kosten, Effizienz). Das bedeutet aber nicht, dass auch ganze Stellen wegfallen müssen. Arbeit ist genügend vorhanden, die Aktivitäten werden sich aber verschieben. Die zukünftigen Arbeitnehmer, insbes. Sachbearbeiter und Manager, müssen somit auf diese neuen Aktivitäten vorbereitet werden. Sie müssen u.a. lernen, mit diesen intelligenten Systemen zusammenzuarbeiten. Auch intelligente Maschinen sind noch immer die Werkzeuge des Menschen und nicht umgekehrt. Die Kontrolle und Gestaltung der neuen Arbeitswelt liegt also bei uns. Eine zentrale Aufgabe der Hochschulen ist es, die Studierenden auf die Jobs von morgen vorzubereiten. Ein erster Schritt ist mit der Durchführung dieses Forschungsprojekts gelegt.

Abschließend möchte ich ein persönliches Fazit ziehen. Die Durchführung des Forschungsprojekts war eine positive Erfahrung. Mit der benötigten Ruhe und in der gewährten Zeit konnte ich ein spannendes Thema intensiv bearbeiten. Durch autodidaktische Studien habe ich neue Erkenntnisse gewinnen können und neue Impulse für die Lehre und angewandte Forschung bekommen. Nebenbei konnte ich weit zurückliegende Kenntnisse in Linux und Latex auffrischen und mit Python habe ich den Einstieg in eine für mich neue Programmiersprache gefunden. Ein Forschungssemester ist generell ein wertvolles Gut. Es ist auch ein nützliches Instrument, um die individuelle fachliche Weiterbildung im Sinn von lebenslanges Lernen zu fördern. Dafür bin ich sehr dankbar!

Quellenverzeichnis

[Beu18] P. Beuth. *Wir müssen reden, Google*. 11. Mai 2018. URL: http://www.spiegel.de/netzwelt/web/google-duplex-hallo-was-spricht-da-a-1207355.html (besucht am 11. 05. 2018).

[Bri18] S. Brin. *2017 Founders' Letter*. 27. Apr. 2018. URL: https://abc.xyz/investor/founders-letters/2017/index.html (besucht am 01. 05. 2018).

[Chu+18] M. Chui, J. Manyika, M. Miremadi, N. Henke, R. Chung, P. Nel und S. Malhotra. *Notes from the AI Frontier – Insights from Hundreds of Use Cases*. 2018. URL: https://www.mckinsey.com/featured-insights/artificial-intelligence/notes-from-the-ai-frontier-applications-and-value-of-deep-learning.

[Dav14] S. Davies. *Hawking warns on Rise of the Machines*. 2. Dez. 2014. URL: https://www.ft.com/content/9943bee8-7a25-11e4-8958-00144feabdc0 (besucht am 01. 05. 2018).

[GBC16] I. Goodfellow, Y. Bengio und A. Courville. *Deep Learning*. 1. Aufl. Cambridge, Massachusettts, USA: The MIT Press, 2016.

[Ido+18] C. J. Idoine, P. Krensky, E. Brethenoux, J. Hare, S. Sicular und S. Vashisth. *Magic Quadrant for Data Science and Machine-Learning Platforms*. 22. Feb. 2018. URL: https://www.gartner.com/doc/reprints?id=1-4RMUF5K&ct=180222 (besucht am 04. 05. 2018).

[KNI18] KNIME. *Deep Learning*. 2018. URL: https://www.knime.com/nodeguide/analytics/deep-learning (besucht am 04. 05. 2018).

[Kre18] M. Kremp. *Google Duplex ist gruselig gut*. 9. Mai 2018. URL: http://www.spiegel.de/netzwelt/web/google-duplex-auf-der-i-o-gruselig-gute-kuenstliche-intelligenz-a-1206938.html (besucht am 10. 05. 2018).

[Kü17] E. Kühl. *Eine Sprache macht noch keinen Terminator*. 2. Aug. 2017. URL: https://www.zeit.de/digital/internet/2017-08/kuenstliche-intelligenz-sprache-lernen-facebook-chatbot (besucht am 10. 05. 2018).

[Lam17] P. Lamblin. *MILA and the Future of Theano*. 28. Sep. 2017. URL: https://groups.google.com/forum/#!topic/theano-users/7Poq8BZutbY (besucht am 10. 05. 2018).

[Mni+13] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra und M. A. Riedmiller. „Playing Atari with Deep Reinforcement Learning". In: *CoRR* abs/1312.5602 (2013). URL: http://arxiv.org/abs/1312.5602.

[Mus17] E. Musk. *Competition for AI*. 4. Sep. 2017. URL: https://twitter.com/elonmusk/status/904638455761612800 (besucht am 01. 05. 2018).

[Pal06] M. Palmer. *Data is the New Oil*. 3. Nov. 2006. URL: http://ana.blogs.com/maestros/2006/11/data_is_the_new.html (besucht am 10. 05. 2018).

[Pre18] Deutsche Presseagentur. *Google-Software gibt sich bei Anrufen als Programm zu erkennen*. 12. Mai 2018. URL: http://www.handelsblatt.com/unternehmen/it-medien/computerprogramm-duplex-google-software-gibt-sich-bei-anrufen-als-programm-zu-erkennen/21517726.html (besucht am 12. 05. 2018).

[RN03] S. Russel und P. Norvig. *Artificial Intelligence – A Modern Approach*. 2. Aufl. Upper Saddle River, New Jersey, USA: Pearson Education International, 2003.

[Rud17] S. Ruder. *Transfer Learning*. 21. März 2017. URL: http://ruder.io/transfer-learning (besucht am 10. 05. 2018).

[SB98] R. S. Sutton und A. G. Barto. *Reinforcement Learning: An Introduction*. 1. Aufl. Cambridge (MA), USA: MIT Press, 1998.

[Sel18a] S. Selle. *Auswahl eines Frameworks für Multiagentensysteme zum Einsatz in Forschung und Lehre*. Techn. Ber. Saarbrücken: Fakultät für Wirtschaftswissenschaften, Hochschule für Technik und Wirtschaft des Saarlandes, 6. Jan. 2018.

[Sel18b] S. Selle. *Künstliche Neuronale Netzwerke und Deep Learning*. Techn. Ber. Saarbrücken: Fakultät für Wirtschaftswissenschaften, Hochschule für Technik und Wirtschaft des Saarlandes, 12. Mai 2018.

[Sel18c] S. Selle. *Softwarelösungen für Reinforcement Learning*. Techn. Ber. Saarbrücken: Fakultät für Wirtschaftswissenschaften, Hochschule für Technik und Wirtschaft des Saarlandes, 6. März 2018.

[Sel18d] S. Selle. *Zusammenstellung, Installation und Konfiguration eines Personal Computers für Deep Learning Projekte*. Techn. Ber. Saarbrücken: Fakultät für Wirtschaftswissenschaften, Hochschule für Technik und Wirtschaft des Saarlandes, 30. Jan. 2018.

[Sel98] S. Selle. „Einsatz Künstlicher Neuronaler Netze auf dem Aktienmarkt“. Diplomarbeit. Heidelberg: Studiengang Volkswirtschaftslehre, Universität Heidelberg, 1998.

[Spi16] Spiegel. *Software schlägt Go-Genie mit 4 zu 1*. 15. März 2016. URL: http://www.spiegel.de/netzwelt/gadgets/alphago-besiegt-lee-sedol-mit-4-zu-1-a-1082388.html (besucht am 10. 05. 2018).

[Ste12] W. Stern. *Die psychologischen Methoden der Intelligenzprüfung und deren Anwendung an Schulkindern*. 1. Aufl. Leipzig: Verlag von Johann Ambrosius Barth, 2012. URL: https://archive.org/stream/diepsychologisch00ster#page/n9/mode/2up.

[Woo09] M. Wooldridge. *An Introduction to MultiAgent Systems*. 2. Aufl. Chichester, West Sussex, UK: John Wiley & Sons, 2009.