FH CAMPUS WIEN
UNIVERSITY OF APPLIED SCIENCES

Steuerung eines Roboterarms in Virtual Reality Unity mithilfe Supervised Learning

Bachelorarbeit

Eingereicht in teilweiser Erfüllung der Anforderungen zur Erlangung des akademischen Grades:

Bachelor of Science in Engineering

an der FH Campus Wien

Studienfach: Computer Science and Digital Communications

Autor:

Danko Dukic

Personenkennzeichen:

c1910475154

Betreuer:

DI Dr. techn. Mugdim Bublin

Datum:

05.06.2022

Erklärung der Urheberschaft:

Ich erkläre hiermit diese Bachelorarbeit eigenständig verfasst zu haben. Ich habe keine anderen Quellen, als die in der Arbeit gelisteten verwendet, noch habe ich jegliche unerlaubte Hilfe in Anspruch genommen

Ich versichere diese Bachelorarbeit in keinerlei Form jemandem Anderen oder einer anderen Institution zur Verfügung gestellt zu haben, weder in Österreich noch im Ausland.

Weiters versichere ich, dass jegliche Kopie (gedruckt oder digital) identisch ist.

Datum: 05.06.2022 Unterschrift: 

Abstract

The growing popularity of machine learning is gaining more and more attention due to the many different applications in different fields, such as robotics. Supervised learning in particular offers a wide range of new possibilities. There are several tools specifically designed for training robot arms that can be used with the Unity development environment. Unity also allows beginners to use these tools and to create and train simulation environments for intelligent robots using virtual reality. The aim of this thesis is to investigate how supervised learning can be applied to robotics using Unity, and how these can be simulated in virtual reality.

Kurzfassung

Die wachsende Beliebtheit des maschinellen Lernens gewinnt immer mehr an Aufmerksamkeit, aufgrund der vielen verschiedenen Einsatzmöglichkeiten in unterschiedlichen Bereichen, wie in der Robotik. Vor allem Supervised Learning bietet ein breites Spektrum an neuen Möglichkeiten. Es gibt verschiedene Tools, die speziell für das Training von Roboterarmen entwickelt wurden und mit der Unity-Entwicklungsumgebung verwendet werden können. Unity erlaubt es auch Einsteigern, diese Tools anzuwenden und Simulations-Umgebungen für intelligente Robotern mithilfe von Virtual Reality zu erschaffen und zu trainieren. Das Ziel dieser Arbeit ist es zu untersuchen, wie Supervised Learning mit Unity in der Robotik angewendet werden kann, und wie diese in Virtual Reality simuliert werden können.

Abkürzungen

AR	Augmented Reality
ARMA	Autoregression und gleitender Durchschnitt
CNN	Convolutional Neural Network
KI	Künstliche Intelligenz
ML	Maschinelles Lernen
MR	Mixed Reality
ROS	Robot Operating System
SVM	Support Vector Machine
VR	Virtual Reality

Schlüsselbegriffe

Maschinelles Lernen
Supervised Learning
Robotik
Unity
Virtual Reality

Inhaltsverzeichnis

1 Einleitung

In unserer schnelllebigen, modernen Welt wird oft behauptet, dass wir uns bereits im digitalen Zeitalter befinden. Dies ist jedoch nicht ganz richtig. Während wir zweifellos eine Menge Zeit mit unseren Computern, Smartphones und anderen digitalen Geräten verbringen, sind wir noch lange nicht in einem Zeitalter, in dem Maschinen die menschliche Arbeit vollständig übernommen haben. In der Tat ist es noch gar nicht so lange her, dass die erste industrielle Revolution stattfand.

Die erste industrielle Revolution brachte einen enormen Fortschritt für die Menschheit. Durch neue Technologien wie Dampfmaschinen und Spinnereien konnte die Produktivität gesteigert und die Lebensbedingungen verbessert werden. Die zweite industrielle Revolution, die in der Mitte des 19. Jahrhunderts begann, setzte diesen Trend fort. Durch den Einsatz von Elektrizität und Verbrennungsmotoren konnten Fabriken noch effizienter betrieben werden. Im Laufe der Zeit entwickelten sich auch die Transportsysteme weiter, was zu einer noch größeren Vernetzung der Welt führte.

Heutzutage stehen wir kurz vor dem Beginn der dritten industriellen Revolution. Diesmal wird sie von Informationstechnologien und dem Internet of Things angetrieben. Eine Schlüsselkomponente dieser neuen Revolution ist maschinelles Lernen – ein Bereich der künstlichen Intelligenz, der es Computer ermöglicht, aus Daten zu lernen und Probleme selbstständig zu lösen.

Maschinelles Lernen ist bereits heute in vielen Bereichen unseres Lebens präsent, auch wenn wir es oft gar nicht bemerken. So nutzen beispielsweise immer mehr Unternehmen maschinengesteuerte Lernsysteme, um große Datenmengen auszuwerten und dadurch bessere Geschäftsentscheidungen zu treffen.

Maschinelles Lernen wird nicht nur bei Geschäftsentscheidungen angewandte, sondern auch in der Robotik. Die Roboter, die diese Technologie anwenden, können zum Beispiel Staubsaugroboter bis hin zu Industrieroboter sein, die Fließbandarbeiten erledigen. Eine Mischform wären etwa ein humanoider Roboter. Unabhängig davon, wofür die Roboter eingesetzt werden, sind die Einsatzmöglichkeiten endlos. Ein Roboter, der maschinelles Lernen anwendet, um Tätigkeiten in Umgebungen zu erledigen, muss zuerst diese erlernen. Eine der interessantesten Arten des maschinellen Lernens, ist das "Supervised Learning", das für Deep Learning angewendet werden kann, um Bilder und Objekte zu klassifizieren.

Für das Umsetzen eines solchen Roboters wird ein Umfeld benötigt. Ohne dieser kann der Roboter nicht in die Realität verwirklicht werden. Für dieses Problem existieren diverse Entwicklungsumgebungen, wo Umgebungen zum Trainieren des Roboters simuliert werden können. Eine dieser Entwicklungsumgebungen ist Unity. Unity kann verwendet werden, um die Lernumgebungen zu entwickeln, sowie auch verwendet werden, um Modelle mithilfe Supervised Learning zu trainieren.

Bevor ein Roboter jedoch, in die Realität umgesetzt werden kann, können Virtualisierungs-Technologien verwendet werden, wie Virtual Reality, um Kosten und Zeit zu sparen.

Das Ziel dieser Arbeit ist es, zu untersuchen, wie die Entwicklungsumgebungen Unity, mit der Hilfe von Supervised Learning verwendet werden kann, um in einer Simulation, sowie auch in Virtual Reality, einem Roboterarm das Aufheben und wieder platzieren eines Objektes

beizubringen. Durch diese Arbeit sollen die LeserInnen ein grundlegendes Verständnis über das Gebiet des Supervised Learnings und der von Unity bereitgestellten Funktionen für die Entwicklung eines Roboters, sowie auch Virtual Reality erlangen.

Die Arbeit teilt sich in drei Bereiche auf. Im Kapitel 2 "Grundlagen des Supervised Learnings" wird zuerst erklärt, was maschinelles Lernen ist, warum Daten eine wichtige Rolle für das Supervised Learning spielen und wie Supervised Learning umgesetzt werden kann. Das Kapitel 3 "Virtual Reality für Unity" wird dann den LeserInnen, Virtual Reality näherbringen und erklären, wie dieses in ein Projekt implementiert werden kann. Schlussendlich wird im Kapitel 4 "Unityprojekt: Roboter mit Supervised Learing" die Umsetzung des Roboters mit Supervised Learning für den im Ziel beschriebenen Task erklärt.

2 Grundlagen des Supervised Learnings

Das Vorhersagen der Zukunft hat die Menschheit seit immer beschäftigt, sei es bei der Jagd von Tieren oder bei der Kriegsführung, wo die Information beziehungsweise die Vorhersage der Position eines Tieres oder des Feindes überlebenswichtig war. Das Wissen, was vor uns liegen könnte, war damals und auch heute für die Planung besonders wertvoll. Damals wurden Orakeln und Hellseher zur Seite herangezogen, um wichtige Planungsentscheidungen zu treffen, heutzutage werden komplexe Algorithmen angewendet, um dies zu bewerkstelligen. Mit der Anwendung von Algorithmen, für die Vorhersage der Zukunft, können wir unsere Geschäftsstrategien gestalten, Verluste minimieren und dadurch Gewinne steigern. Vorhersagen faszinieren uns seit jeher.

In der heutigen Zeit sind Daten das neue Öl. Data Science und maschinelles Lernen (ML) machen sich diese Macht der Daten zunutze, um Vorhersagen für uns zu treffen. Diese Fähigkeiten ermöglichen es uns, Trends und Anomalien zu untersuchen, verwertbare Erkenntnisse zu gewinnen und unseren Geschäftsentscheidungen eine Richtung zu geben. Dieses Kapitel soll den LeserInnen eine Übersicht über das Supervised Learning bieten.

Am Ende dieses Kapitels werden die LeserInnen mit den Konzepten von Data Science und ML vertraut sein, wobei der Schwerpunkt auf Supervised Learning liegt. Es werden Konzepte von Algorithmen des Supervised Learning untersucht, um Regressions- und Klassifikationsprobleme zu lösen. Es wird auch auf das Unsupervised und Reinforcement Learning eingegangen, um den LeserInnen einen Vergleich zu den Alternativen darzustellen. Um den Lernprozess zu vervollständigen, werden zuerst Daten, Datentypen und Datenquellen aufgearbeitet, um so ein besseres Verständnis den LeserInnen zu geben, warum Daten das A und O von Supervised Learning sind. In diesem Prozess werden wir Algorithmen des überwachten Lernens, ihre Grundlagen sowie den Prozess, der im Hintergrund abläuft und wie wir Daten nutzen, um Lösungen zu erstellen, erkunden.

2.1 Was ist maschinelles Lernen

Wenn wir ein Bild auf Instagram posten, bei Zalando einkaufen, tweeten oder Videos auf TikTok ansehen, sammelt jede dieser Plattformen Daten über uns. Durch die Nutzung von solchen Plattformen hinterlassen wir einen digitalen Fußabdruck. Die dadurch generierten Datenpunkte werden gesammelt und analysiert, und ML ermöglicht es solchen Social-Media-Plattformen, auf unsere Bedürfnisse Empfehlungen, genauer gesagt Werbung zu pushen. Bei TikTok und Netflix werden die Wiedergabelisten von UserInnen, basierend auf dem Genre der Videos, die den UserInnen gefallen, aktualisiert. Instagram kann UserInnen Beiträge empfehlen, indem es beobachtet, welche Art von Produkten sie häufig kaufen und Zalando kann den nächsten Kauf entsprechend dem Warenkorb vorschlagen.

Die Definition von ML lautet wie folgt: ”Beim maschinellen Lernen untersuchen wir statistische/mathematische Algorithmen, um aus Daten Muster zu lernen, die dann verwendet werden, um Vorhersagen über die Zukunft zu treffen.”. [Ver20]

ML ist nicht nur auf die Online-Medien beschränkt. Die Leistungsfähigkeit von ML wurde auf zahlreiche Bereiche, Regionen und Anwendungsfälle ausgedehnt. Anwendungen von ML

werden in laufe dieses Kapitels dargelegt.

Bei ML werden große Datensätze analysiert, um die darin enthaltenen Muster zu entdecken. Diese entdeckten Muster werden dann auf reale Daten angewandt, um Vorhersagen für die Zukunft zu treffen. Diese realen Daten sind unklar, und durch die Vorhersagen können unter anderem Unternehmen ihre jeweiligen Strategien mit der Hilfe von ML gestalten. Dafür müssen nicht Computer explizit für solche Aufgaben programmieren werden, vielmehr werden Algorithmen angewendet, die Entscheidungen auf der Grundlage historischer Daten und statistischer Modelle treffen.

Die Frage, die sich stellt, ist, wie sich ML in die Landschaft der Datenanalyse einfügt. Man stößt häufig auf die Begriffe wie Datenanalyse, Data Mining, ML und künstliche Intelligenz (KI). Ein weiterer Begriff, der weitverbreitet ist, ist Data Science, für die es aber keine genaue Definition gibt. Bevor die Anwendungsbereiche und Supervised Learning genauer erklärt werden, ist es wichtig, dass die LeserInnen die Beziehungen zwischen den erwähnten Begriffen kennen.

2.1.1 Beziehung zwischen Datenanalyse, Data Mining, ML und KI

Data Mining beschreibt die Prozesse des Sammelns von Daten. Die Sammlung erfolgt aus großen Datensätzen, Datenbanken und Data Lakes. Des Weiteren ist Data Mining auch der Prozess des Extrahierens von Informationen und Mustern aus diesen Daten und des Umwandelns dieser Erkenntnisse in eine nutzbare Struktur. Zu Data Mining gehört auch das Management, die Vorverarbeitung und die Visualisierung der Daten. Data Mining ist der allererste Schritt in jedem Datenanalyseprojekt.

Datenanalyse ist der Prozess, der die Untersuchung der Daten beschreibt. Bei der Datenanalyse werden die Daten analysiert, Anomalien identifiziert und Erkenntnisse mithilfe von Tabellen, Diagrammen, Histogrammen, Kreuztabellen und so weiter gewonnen. Die Datenanalyse ist der wichtigste Schritt, der aussagekräftig ist, da die gewonnenen Erkenntnisse leicht zu verstehen, nachvollziehbar und einfach sind. Ohne der Datenanalyse wäre die Erstellung eines ML-Modells nicht durchführbar. [Ver20]

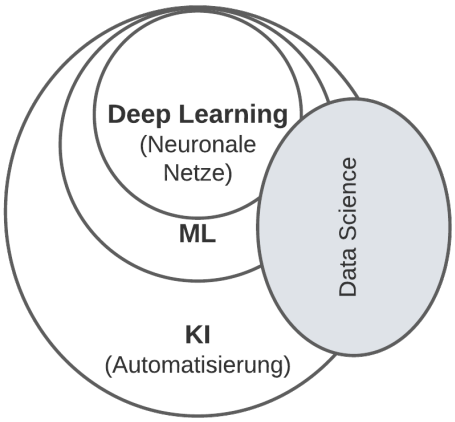

Abbildung 2.1: Beziehung zwischen KI, ML Deep Learning und Data Science.[Ver20]

Eine der häufigsten Fragen, die sich stellen lässt, ist, was die Beziehung zwischen ML, KI und Deep Learning ist und wie Data Science in das Ganze reinpasst. In der Abbildung

2.1 sieht man die Überschneidungen zwischen diesen Bereichen. Unter KI kann man sich eine automatisierte Lösung vorstellen, die Menschen intensive Aufgaben ersetzt. Mit der KI werden somit Zeit und Kosten reduziert und die gesamte Effizienz wird verbessert.

In den letzten Jahren ist Deep Learning mehr aufgeblüht. Das Herz und die Seele von Deep Learning sind die neuronalen Netze. Deep Learning ist ein Teilbereich von KI und ML und umfasst komplexe mathematische entwickelte Modelle, die zur Lösung von Geschäftsproblemen verwendet werden. Neuronale Netze werden meisten für die Klassifizierung von Bildern und zur Analyse von Text-, Audio- und Videodaten verwendet.

Data Science liegt an den Schnittstellen dieser Überschneidungen. In Data Science ist nicht nur ML wichtig, sondern auch das Statistikverständnis, Programmierverständnis und Geschäftsverständnis, um Geschäftsprobleme zu lösen. Die Aufgaben eines Data Scientist's sind, Geschäftsprobleme zu lösen und umsetzbare Erkenntnisse für das Unternehmen zu gewinnen.

Nun sollten den LeserInnen die Rolle von ML und seine Beziehung zu anderen datenbezogenen Bereichen klar sein. Daten spielen eine zentrale Rolle bei ML. Im nächsten Abschnitt geht es um die Daten, ihre Arten und Eigenschaften.

2.1.2 Daten, Datentypen und Datenquellen

Heutzutage werden wir mit verschiedenen Arten von Daten konfrontiert, weshalb wir auch ein Verständnis für Daten brauchen. In diesem Abschnitt werden die Arten von Datensätzen, die erzeugt werden, sowie Beispiele dafür besprochen. In der Abbildung 2.2 sieht man die Unterscheidung zwischen Daten, die relevant sind für ML.

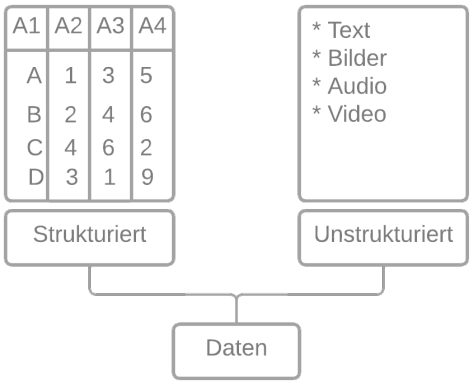

Abbildung 2.2: Daten können in strukturierte und unstrukturierte Daten unterteilt werden. [Ver20]

Es wird in strukturierte und unstrukturierte Daten unterteilt. Strukturierte Daten sind leichter zu bearbeiten als unstrukturierte. Bei unstrukturierten Daten kommt Deep Learning ins Spiel.

Ob online oder offline, erzeugen wir jeden Tag Daten bei Interaktionen und Transaktionen, an denen wir beteiligt sind. Auf Social-Media-Plattformen, im Einzelhandel, in der Bank oder auf dem Smartphone erzeugt jede Interaktion Daten. [Ver20]

Die Daten werden in strukturierte und unstrukturierte Daten unterteilt. Wenn wir ein Telefonat unternehmen, werden bei dem Telekommunikationsbetreiber die Daten des Anrufs aufbewahrt, wie die Kosten des Anrufs, die Dauer des Anrufs, die Tageszeit und so weiter.

Auch bei Online-Transaktionen bei den Banken werden Daten gesammelt, wie der Betrag der Transaktion, den Empfänger, den Grund für die Transaktion, Datum/Uhrzeit und so weiter. Diese Datenpunkte werden als strukturierte Daten bezeichnet, da sie in Zeilen- und Spaltenstrukturen dargestellt werden können. Großteil von uns erzeugten Daten, die verwendet und analysiert werden, sind strukturierte Daten. Strukturierte Daten werden in Datenbanken oder Servern mit SQL, MySQL, Oracle, AWS und so weiter gespeichert.

Daten, die nicht in einer Zeilen-Spalten-Struktur dargestellt werden können, zumindest nicht in ihrem Grundformat, sind unstrukturierte Daten. Unstrukturierte Daten sind Fotos (Instagram, Facebook), Textdaten (Tweets, Bewertungen, Kommentare), Audiodateien (Aufnahmen, Hörbücher) und Videos (YouTube-Beiträge, TikTok und so weiter). Diese Daten können trotzdem gespeichert und analysiert werden. Dies ist jedoch schwieriger als die Analyse von strukturierten Daten. Der wichtige Punkt ist es unstrukturierte Daten in Integer (Ganzzahlen) umzuwandeln, damit diese von Computern verstanden und verarbeitet werden können. Ein gutes Beispiel dafür sind farbige Bilder, welche aus Pixeln bestehen. Die Pixel haben einen RGB Wert (Rot, Grün, Blau) zwischen 0 und 255. Dies bedeutet, dass jedes Bild in Form von Matrizen mit Integern dargestellt werden kann. Dadurch können dann die Daten analysiert und verarbeitet werden von Computern.

Die Datenqualität wird oft ignoriert, und nicht als wichtiger Aspekt berücksichtigt. Sie entscheidet über die Qualität der Analyse und die gewonnenen Erkenntnisse. Bei der Datenqualität gilt das Motto: Garbage in, garbage out. In der Abbildung 2.3 sind die Merkmale eines guten Datensatzes abgebildet. Um Probleme zu vermeinend, sollte man viel Zeit in die Sicherstellung der Datenqualität investieren. [Ver20]

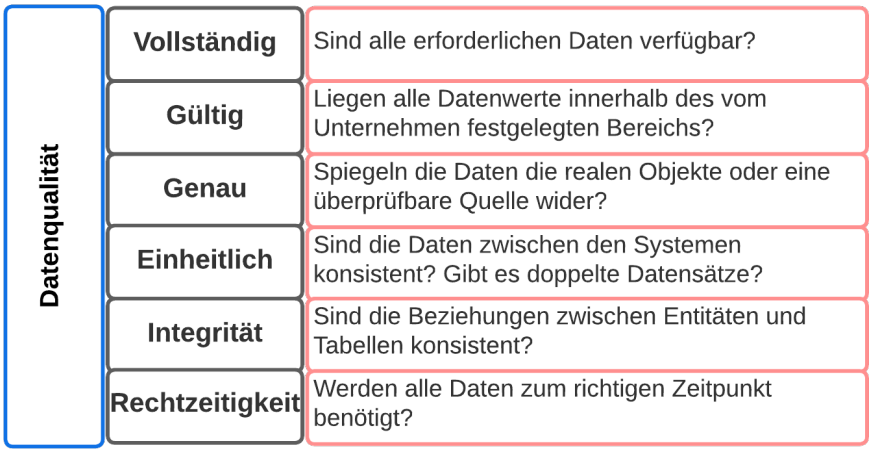

Abbildung 2.3: Bei der Entwicklung einer ML-Lösung spielt die Datenqualität eine erhebliche Rolle. [Ver20]

Die folgenden Standards sollten bei der Durchführung der Datenqualität sichergestellt werden: [Ver20]

- Bei der **Vollständigkeit** der Daten versteht man den Prozentsatz der verfügbaren Variablen. In der Praxis stellt man fest, dass sehr viele Attribute fehlen oder dass sie NULL Werte haben. Daher ist es von hoher Bedeutung, dass die Daten ordnungsgemäß beschafft werden, um die Vollständigkeit zu gewährleisten. Bei der Datenaufbereitung werden die Attribute ersetzt oder je nach den Anforderungen weggelassen. Zum Beispiel, wenn wir mit Transaktionsdaten von einem Einzelhandel arbeiten, muss sichergestellt werden, dass die Einnahmen für alle oder fast alle Monate verfügbar sind.

- Bei der **Datengültigkeit** soll schon in der Phase der Datenermittlung sichergestellt werden, dass alle wichtigen Leistungsindikatoren erfasst werden. Dafür werden FachexpertInnen zur Seite gezogen, um in diesen Prozess die Leistungsindikatoren zu berechnen und zu überprüfen. Dies kann etwa der Fall bei der Berechnung der durchschnittlichen Gesprächskosten eines Mobilfunkteilnehmers sein, wo die FachexpertInnen vorschlagen können, welche Leistungsindikatoren wichtig sind, wie das hinzufügen oder streichen von einige Kosten wie Frequenzkosten, Akquisitionskosten und so weiter.

- Mit der **Genauigkeit** der Daten sollte sichergestellt werden, dass alle erfassten Datenpunkte korrekt sind und keine falschen Informationen in den Daten erhalten sind. Diese falschen Informationen werden meistens durch menschliche Fehler oder Softwareproblemen ausgelöst. Wenn zum Beispiel im Einzelhandelsgeschäft die Anzahl der KäuferInnen erfasst wird, ist die Kundenanzahl am Wochenende in der Regel höher als unter der Woche. Solche Anomalien müssen in der Sondierungsphase sichergestellt werden.

- Daten sollten **einheitlich** sein und sollten sich nicht unterscheiden zwischen verschiedenen Systemen und Schnittstellen. Bei der Darstellung von wichtigen Leistungsindikatoren werden oft verschiedene Systeme verwendet. Dadurch können wir etwa die Anzahl der Klicks auf einer Website auf unterschiedlicher Weise erfassen. Damit die korrekte Analyse durchgeführt wird und konsistente Erkenntnisse gewonnen werden, muss die Konsistenz dieser Leistungsindikatoren gewährleistet sein.

- Mit der **Datenintegrität** des Systems stellen wir sicher, dass zum Beispiel bei der Speicherung von Daten in Datenbanken und Tabellen die Beziehungen zwischen den verschiedenen Attributen und Entitäten konsistent sind oder vorhanden sind. Eine robuste Datenstruktur ist erforderlich für einen vollständigen, korrekten und effizienten Data Mining Prozess.

- Bei der Datenanalyse ist das Ziel, Muster und Trends in Daten zu finden. Es gibt Bewegungen in Bezug auf die Tage/Zeit und saisonale Schwankungen, die bei den Daten auftreten können. Daher ist es oft zwingend, Daten von den letzten Jahren zu nehmen und zu erfassen, um bei wichtigen Leistungsindikatoren die Trends zu messen. Um Schwankungen zu erfassen, muss die **Rechtzeitigkeit** der erfassten Daten genug repräsentativ sein.

Duplikate, fehlende Werte, Junk-Werte und Ausreißer sind die häufigsten Probleme, die bei den Daten auftreten könne.

Nun sollten die LeserInnen verstehen, welche Merkmale hochwertige Daten aufweisen müssen für eine gute Analyse und was ML ist.

2.2 Anwendungsbereiche

In diesem Abschnitt werden die drei Bereiche, in den ML Algorithmen angewendet werden, erklärt. Zunächst werden Klassifizierung und Regression behandelt, auf die Algorithmen des Supervised Learning angewendet werden, und dann wird das Clustering funktionell beschrieben, auf das die Algorithmen des Unsupervised Learning angewendet werden.

2.2.1 Klassifizierung

Der Prozess der Zuweisung von einer oder mehrere vordefinierten Kategorien zu jedem Objekt wird als Klassifizierung genannt. Die einfachste Art der Klassifizierung ist die binäre Klassifizierung, wo jedes Objekt einer von zwei Kategorien zugeordnet wird. Die binäre Klassifizierung kann zu einer Mehrfachklassifizierung erweitert werden, indem man mehrere Kategorien angibt. Die Mehrfachklassifizierung wird in die binäre Klassifizierung zerlegt, wenn es sich bei der Aufgabe um eine weiche Klassifizierung handelt, wo man jedem Objekt mehrere Kategorien als eine zuordnen kann. Die binäre Klassifizierung, die Mehrfachklassifizierung und ihre Zerlegung in binäre Klassifizierungen werden kurz in diesem Abschnitt erklärt. [Jo21]

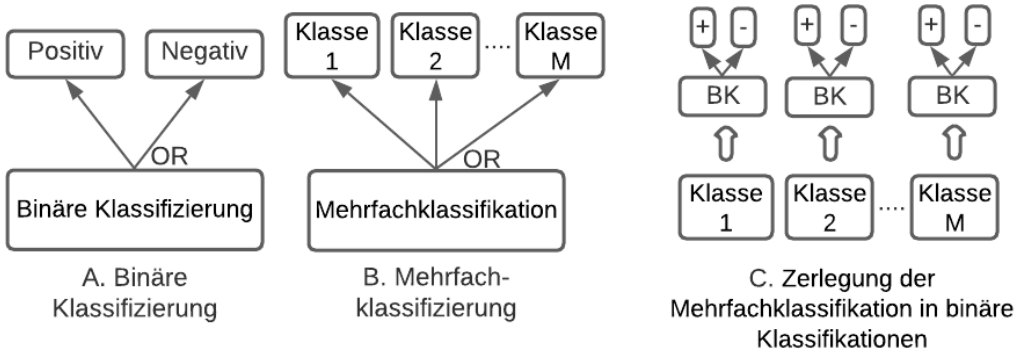

Abbildung 2.4: Klassifizierungsarten [Jo21]

In der Abbildung 2.4 (a) ist die binäre Klassifizierung funktional dargestellt. Es werden den beiden vordefinierten Kategorien, positive und negative Klassen, den Proben zugeordnet. Jedes Element wird in die positive oder negative Klasse, bei der binären Klassifizierung, eingestuft. Bei der Implementierung von ML Algorithmen ist es anfänglich immer davon auszugehen, dass es sich bei dem gegebenen Problem um eine binäre Klassifizierung handelt. Mehrfachklassifizierung oder Regression können in binäre Klassifizierungen zerlegt werden.

In der Abbildung 2.4 (b) ist die Mehrfachklassifikation funktional dargestellt. Es sind mehr als zwei Klassen vordefiniert, M Klassen oder $M > 2$, und zu jeder Kategorie werden Stichprobenbeispiele zugewiesen. Bei der Mehrfachklassifizierung werden Objekte in eine oder mehrere M Klassen klassifiziert. Wenn mehr als eine Kategorie unter die vordefinierten zugeordnet werden darf, ist dies dann eine weiche Klassifizierung. Die Aufgaben können in eine binäre Klassifikation zerlegt werden.

In der Abbildung 2.4 (c) ist der Prozess der Zerlegung der Mehrfachklassifikation in binäre Klassifikationen dargestellt. Die erste zu lösenden Aufgabe sind die gegebenen M Kategorien von der Mehrfachklassifikation. Es wird dann zu jeder Kategorie ein binärer Klassifikator zugeordnet, wo dann jedes Element in die entsprechende Kategorie eingeordnet wird. Die Mehrfachklassifikation wird in so viele binäre Klassifikationsaufgaben zerlegt, wie es Kategorien gibt.

Einige Bemerkungen zu dem Einsatz von ML Algorithmen für Klassifizierungsaufgaben sind: Die Kategorien für die Zeichenerkennung und der Spam-Filterung werden durch den gesunden Menschenverstand gewählt/bestimmt. Wo es schwieriger wird, die Kategorien zu bestimmen, ist es bei der themenbasierten Text- und Bildklassifizierung. Verschachtelte Kategorien sind bei er hierarchischen Klassifizierung innerhalb einer Kategorie erlaubt. Bei der

Klassifizierung mit mehreren Ansichten sind mehrere Kategoriensysteme zulässig. Ein Kategoriesystem kann ein Baum oder eine Liste von vordefinierten Kategorien sein. [Jo21]

2.2.2 Regression

Der Prozess der Schätzung eines Ausgabewerts auf der Grundlage mehrerer Faktoren wird als Regression genannt. Die im vorigen Abschnitt erörterte Klassifizierung ist der Ausgabewert diskret, während bei der Regression, die in diesem Abschnitt behandelt wird, der Ausgabewert kontinuierlich ist. Es wird in zwei Arten der Regression unterschieden, der univariaten Regression und der multivariaten Regression. Bei der univariaten Regression wird nur ein Ausgabewert geschätzt, während bei der multivariaten Regression, mehr als ein Ausgabewert geschätzt wird. Typische Fälle, wo die Regression verwendet wird, ist die Vorhersage der Zeitreihen und die Annäherung an nichtlineare Funktionen. Die Funktionalität der Regression wird in diesem Abschnitt kurz erklärt. [Jo21]

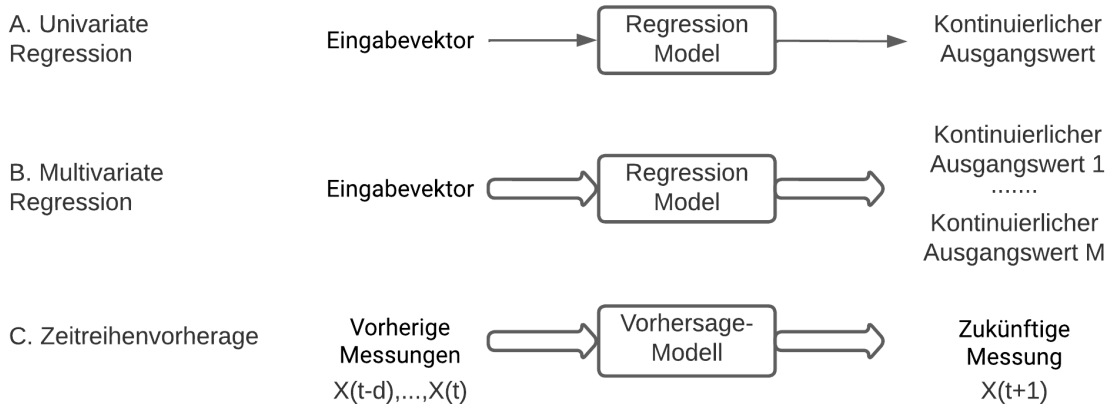

Abbildung 2.5: Regressionsarten [Jo21]

In der Abbildung 2.5 (a) ist eine Art der univariate Regression abgebildet, die nur eine einzige Ergebnisvariable hat. Die nichtlineare Funktionsannäherung, bei der nur eine einzige Ausgangsvariable mit nur einer Eingangsvariablen geschätzt werden kann, ist ein typisches Beispiel für eine univariate Regression. Die Zeitreihenvorhersage, bei der eine einzelne zukünftige Variable durch die Analyse vergangener Variablen und einer aktuellen Variable vorhergesagt wird, ist auch Teil der univariaten Regression. Das Vorhersagen eines einzelnen Wechselkurses oder Aktienkurses ist auch ein typisches Beispiel für die univariate Regression. Jedoch ist die Vorhersage eines Rückganges oder Anstieges eines Wertes und nicht des Wertes selbst eine Klassifizierung.

In der Abbildung 2.5 (b) ist die Funktion der multivariaten Regression dargestellt. Bei dieser Art der Regression werden mehrere Ausgangswerte zu den als Eingangsvektor gegebenen Eingabedaten erzeugt. Multivariate Zeitreihenprognose bezeichnet den Prozess der Vorhersage von mehreren zukünftigen Werten durch die Analyse vergangener und gegenwärtiger Werte. Zu der multivariaten Regression gehört auch die Vorhersage von mehreren Aktienkursen und Wechselkursen von verschiedenen Währungspaaren. Die multivariate Regression kann auch in unabhängige univariate Regressionen zerlegt werden.

In der Abbildung 2.5 (c) ist die Funktionsweise der Zeitreihenvorhersage dargestellt, welche eine besondere Art der Regression ist. Diese Regression beruht auf früheren Messungen und

einer aktuellen Messung. Klassische Ansätze für die Zeitreihenprognose sind Autoregression, gleitende Durchschnitt und ARMA (Autoregression und gleitender Durchschnitt). Diese Modelle wurden in den 90er durch neuronale Netze und in den 2000er durch SVM (Support Vector Machine) ersetzt. Um die Leistung mit der Verwendung von neuronalen Netzen bei Zeitreihenprognose zu verbessern, wurde die Schätzung des gleitenden Durchschnitts empfohlen.

Einige Bemerkungen zu Regression aufgaben sind: Die Regression ist neben der Klassifizierung der Task, für die Algorithmen des Supervised Learning geeignet sind. Der Ausgabewert bei der Regression ist als ein kontinuierlicher Wert angegeben, während der Ausgabewert bei der Klassifizierung als ein diskreter Wert angegeben ist. Die Eingabe- als auch die Ausgabewerte müssen zwischen null und eins normalisiert sein, da die Eingaben in unterschiedlichen Maßstäben angegeben werden. Der Ausgabewert, der aus dem Regressionsprozess resultiert, sollte auf seinen ursprünglichen skalierten Wert de-normalisiert werden. [Jo21]

2.2.3 Clustering

Der Prozess der Segmentierung einer Gruppe von Elementen in Untergruppen, die jeweils ähnliche Elemente enthalten, wird als Clustering genannt. Die in den vorigen Abschnitten beschriebene Klassifizierung und Regression wird für Algorithmen des Supervised Learnings angewendet, hingegen das Clustering für Algorithmen des Unsupervied Learnings angewendet wird. Damit man Daten mit dem Clustering verarbeitet, wird die Definition einer Ähnlichkeitsmetrik zwischen Elementen benötigt. Clustering wird für die Automatisierung von Sammlungen von gelabelten Exempeln verwendet, sodass es möglich ist, Clustering mit Klassifizierung zu integrieren, um Datenelemente zu organisieren. Die Funktionalität des binären Clustering und des multiplen Clustering wird in diesem Abschnitt kurz erklärt. [Jo21]

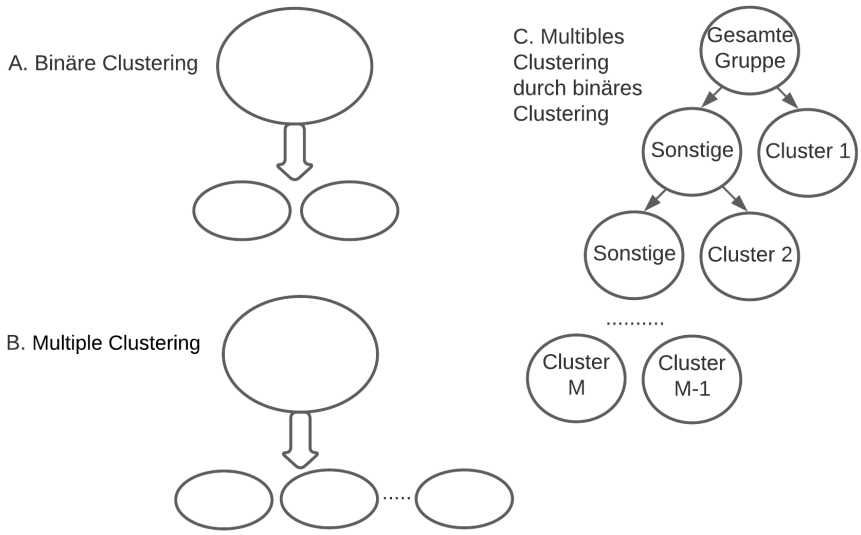

Abbildung 2.6: Clusteringarten [Jo21]

In der Abbildung 2.6 (a) wird das binäre Clustering dargestellt, womit Gruppen auf der Grundlage von Ähnlichkeiten zwischen Elementen in zwei Untergruppen unterteilt werden. Bevor man die Elemente in Untergruppen aufteilt, muss man zunächst eine Gruppe von Elementen als Eingabe geben. Danach wird ein zufälliges Element aus der Gruppe genommen, um Untergruppen zu bilden, eine, die dem Element ähnlich ist und eine, die dem Element un-

ähnlich ist. Es kann auch alternativ zwei Elemente zufällig ausgewählt werden als anfängliche Cluster Prototypen, die restlichen werden dann in ähnlichere Gruppen aufgeteilt.

In der Abbildung 2.6 (b) wird das Multiple Clustering dargestellt, womit eine Gruppe in mehr als zwei Untergruppen unterteilt wird. Zuerst wird eine einzelne Gruppe als Eingabe gegeben, wie beim binären Clustering. Ein typischer Ansatz für die Durchführung von Mehrfachclustern ist der k-means-Algorithmus. Dabei werden zufällig so viele Elemente wie Cluster ausgewählt, während die restlichen auf der Grundlage ihrer Ähnlichkeit zu den ausgewählten Elementen angeordnet werden. Bei dem harten Clustering werden die Gruppen in exklusive Cluster unterteilt und bei dem weichen Clustering in sich überschneidende Cluster.

In der Abbildung 2.6 (c) ist die Zerlegung einer Multiple-Clustering-Aufgabe in Binäre-Clustering-Aufgaben dargestellt. Es werden die Gruppen in zwei Gruppen aufgeteilt, in eine Gruppe mit ähnlichen Elementen und in eine mit unähnlichen Elementen. Dieser Prozess ähnelt dem Divisive-Algorithmus für das Clustern von Objekten.

Einige Bemerkungen zu der Aufgabe des Clusterings: Algorithmen des Unsupervised Learnings werden diese Aufgabe angewendet, wie schon anfänglich erwähnt. Das Ergebnis des Clusterings von Datenelementen ist eine flache Liste von unbeschrifteten Clustern oder eine hierarchische Struktur. Eine weitere Aufgabe des Datenclustering ist die Benennung von Clustern. Um die Datenklassifizierung zu automatisieren, mit der Vordefinition von Kategorien und die Mustersammlung als vorbereitende Aufgaben, können Datenclustering und Clusterbenennung kombiniert werden. [Jo21]

2.3 Supervised Learning

Mit Supervised Learning können wir Muster in Daten finden, mit den Features von Merkmalen und Labels. Bevor wir fortfahren, ist es wichtig, die zwei neuen datenbezogenen Begriffe zu definieren.

Die Features stellen die Eigenschaften jedes Eintrags dar bei der Verwendung eines Datensatzes, während die Labels die Weise und Art wie die Einträge definiert werden sind. Um dies besser zu verstehen, schauen wir uns jetzt ein Beispiel an, wie es in der Praxis umgesetzt werden könnte.

Wir möchten unser Haus verkaufen, aber wir sind uns nicht sicher zu welchem Preis wir das tun könnten, um das realistischste Angebot von den Käufern zu bekommen. Als Hilfestellung haben wir Zugang zu einem großen Datensatz mit Informationen über alle Häuser und deren Preise in der Stadt, in der wir leben.

Bei diesem Beispiel wären die Features die Details von allen gelisteten Häusern, wie die Anzahl der Bäder, Schlafzimmer, Stockwerke und ob sie einen Balkon und/oder Garten haben. Die Labels wären dann die Preise der Häuser. Der Datensatz könnte so wie in der Tabelle 2.1 aussehen.

In der Realität hat so ein Datensatz viel mehr Einträge als unser Beispiel, und es wären auch mehr Features aufgezeichnet. Nichtsdestotrotz hilft dieses kleine Beispiel das Konzept der Features und Labels zu veranschaulichen.

Anhand von Daten in der Tabelle können wir mit ML den Preis vorhersagen, zu welchen wir unser Haus zum besten und realistischsten Preis verkaufen können.

Wenn unser Haus zum Beispiel ein Badezimmer und Schlafzimmer, einen Balkon, aber keinen Garten hätte, könnten wir das Haus auf den Markt für 500.000 € stellen und nicht für 750.000 € aufgrund der uns gegebenen Datensätze. [Ger20]

Preis	Anzahl der Bäder	Schlafzimmer	Stockwerke	Balkon	Garten
€1,700,000	2	4	2	Nein	Ja
€2,500,000	4	6	3	Ja	Ja
€500,000	1	1	1	Ja	Nein
€750,000	1	1	1	Ja	Ja
€475,000	1	1	1	Nein	Nein
€650,000	1	1	1	Ja	Ja
€875,000	1	2	1	Ja	Ja
€1,500,000	2	3	2	Nein	Ja
€400,000	1	1	1	Nein	Nein

Tabelle 2.1: Beispiel eines gelabelten Datensatzes [Ger20]

Wie wir bei diesem kurzen Beispiel sehen können, kann die Preisermittlung manuell durchgeführt werden, indem wir uns die Datensätze anschauen, aber in der Wirklichkeit wäre die Datenmenge deutlich größer und aufwendiger für die manuelle Ermittlung. Daher wird ML für die Berechnung genutzt, da es viel schneller ist als wir.

Andere Beispiele, wo das Supervised Learning nützlich ist, ist bei der Erkennung von Spam bei E-Mails oder die Vorhersage der Gewinnwahrscheinlichkeit einer Fußballmannschaft anhand von alten aufgezeichneten Daten oder die Vorhersage, ob es bei Versicherungsansprüchen von KundInnen es sich um einen Betrug handelt.

Bei Supervised Learning lässt sich zusammenfassend sagen, dass die Vorhersage auf der Grundlage von gelabelten Daten gemacht wird.

2.3.1 Schritte in einem Algorithmus für Supervised Learning

In diesem Kapitel werden die Schritte für den Supervised Learning Algorithmus untersucht. Für jeden Schritt bei einem Supervised Learning Problems gelten die Grundsätze der Datenqualität, -vollständigkeit und -robustheit.

Schritte für Supervised Learning Algorithmus	
1. Zielvariable	• Zielvariable definieren, d.h. für das Geschäftsproblem die wichtigsten Leistungsindikatoren • Ist es ein Regressionsproblem oder ein Klassifikationsproblem?
2. Data Mining	• Die Trainingsdaten für das Problem beschaffen • Trainingsdaten müssen vollständig und repräsentativ sein
3. Datenaufbereitung	• Daten bereinigen, fehlende Werte behandeln, Duplikate entfernen, etc. • Zielvariable und unabhängige Variable bestimmen
4. Explorative Datenanalyse	• Daten in Form von Diagrammen generieren, um Erkenntnisse zu gewinnen. Hilfreich für die Kontrolle, ob die Analyse richtig ist.
5. Statistische Modellierung	• Algorithmus auswählen wie logistische Regression, Entscheidungsbaum usw. • Für jeden Algorithmus geeignete Ergebnisse generieren.
6. Vergleich der Resultate	• Ergebnisse der Algorithmen vergleichen und den besten Algorithmus wählen

Tabelle 2.2: Die allgemeinen Schritte eines Supervised Learning Algorithmus. [Ver20]

In der Tabelle 2.2 sind die Schritte dargestellt, die wir in diesem Kapitel folgen werden, um ein Problem des Supervised Learnings zu lösen. Anzumerken ist, dass es sich bei den Schritten um einen iterativen Prozess handelt. Bei der Umsetzung von den Supervised Learning Algorithmus stoßt man immer auf neue Erkenntnis, die einen veranlassen Schritte zurückzugehen oder man merkt, dass ein Attribut nicht mehr gültig beziehungsweise nützlich ist. Für ML sind solche Iterationen ein wesentlicher Bestandteil, und sowohl auch für das Supervised Learning.

Schritt 1: Wenn wir ein Supervised Learning Algorithmus lösen wollen, brauchen wir die Zielvariable und die unabhängigen Variablen. Für die Lösung des Supervised Learning Problems ist die Zielvariable von zentraler Bedeutung, denn eine falsche Definition der Zielvariablen kann dazu führen, dass das Ergebnis umgekehrt wird.

Wenn wir erkennen wollen, ob eine eingehende E-Mail Spam ist oder nicht, können die Zielvariablen in den Trainingsdaten "Spam-Kategorie" sein. Wenn es sich um E-Mail Spam handelt, kann die Spam-Kategorie "1" sein und wenn es keine Spam-E-Mail ist, kann die Spam-Kategorie "0" sein. Die Ausgabe des Modells ist ein Wahrscheinlichkeitswert dafür, ob die eingehenden E-Mails Spam sind oder nicht. Wenn der Wahrscheinlichkeitswert höher wird, ist auch die Wahrscheinlicher höher, dass es sich bei den E-Mails um Spam handelt.

In diesem Schritt wird auch bestimmt, ob es sich um ein Regressions- oder um ein Klassifikationsproblem handelt, nachdem wir die Zielvariable festgelegt haben. Wenn wir feststellen, dass es sich um ein Klassifizierungsproblem handelt, müssen wir eine Ebene tiefer gehen, um zu klassifizieren, ob es ein binäres Klassifizierungsproblem oder ein Mehrfachklassifikationsproblem ist.

Bei dem ersten Schritt haben wir eine Zielvariable festgelegt und die Entscheidung getroffen, ob es sich bei dem Problem um ein Regressions- oder ein Klassifikationsproblem handelt.

Schritt 2: Die Trainingsdaten werden in diesem Schritt für das Modell bestimmt. Es ist wichtig, die besten Grundsätze hinsichtlich der Datenqualität in diesem Schritt zu beachten. Die Trainingsdaten setzen sich sowohl aus den Zielvariablen und auch aus den unabhängigen Variablen. Sie werden aus allen möglichen Datenquellen eingenommen und sollten auch repräsentativ genug sein, um die Vollständigkeit zu gewährleisten.

Schritt 3: Die Daten werden für die statistische Modellierung vorbereitet. Die gesammelten Daten sind oft chaotisch und haben Anomalien. Bei der Modellierung können Duplikate, NULL Werte usw. in den gesammelten Daten gefunden werden. Was auch noch in den Daten gefunden werden kann, sind String-Werte, Namen mit ganzen Zahlen usw., und diese Daten müssen bereinigt werden. Die Zielvariable und sowohl die unabhängigen Variablen werden bei der Modellierung identifiziert, dabei ist die Zielvariable entweder kontinuierlich oder kategorisch und die unabhängigen Variablen kann kategorial oder kontinuierlich sein.

Bei diesem Schritt könne auch neue abgeleitete Variablen erstellt werden, wie die maximale Dauer, die Anzahl der Durchläufe, der durchschnittliche Umsatz usw.

Schritt 4: Als Nächstes werden Erkenntnisse aus den Daten gewonnen, mit der explorativen Datenanalyse. Dabei werden Beziehungen zueinander, Korrelationen, Verteilungen der unabhängigen Variablen, Trends usw. erstellt und es wird dadurch ein gutes Verständnis für die Daten gewonnen. In diesen Schritt werden auch gerne oft neue Variablen erstellt. [Ver20]

Schritt 5: Nach der Vorbereitung der Daten und der explorativen Datenanalyse, kom-

men wir jetzt zu der statistischen Modellierung. Dabei wird ein Modell aus einer Reihe von verschiedenen Algorithmen für das Supervised Learning erstellt. In der Regel werden die Algorithmen anhand von der Grundlage des Problems und der Erfahrung gewählt. Es werden dann für die verschiedenen Methoden Genauigkeit-Diagramme erstellt.

Folgende Schritte werden durchgeführt für das Training des Algorithmus:

1. Für das Training, Testen und Validieren kann der gesamte Datensatz in einem Verhältnis von 60:20:20 aufgeteilt werden. Es kann auch ein Verhältnis von 80:20 für die Trainingsdaten und Testdaten verwendet werden.
2. Wenn aber die Rohdaten recht hoch sind, wie beim Training mit Bildern, wo bis zu Millionen von Daten gebraucht werden, kann das Verhältnis von 98:1:1 für das Training, Testen und Validieren verwendet werden.
3. Die drei Datensätze sollten nach dem Zufallsprinzip immer aus den ursprünglichen Rohdaten gezogen werden, damit es nicht zu einer Selektionsverzerrung kommt. Wenn die Test- oder Validierungsdatensätze nicht wirklich repräsentativ für die Trainingsdaten sind, kann die Wirksamkeit nicht korrekt gemessen werden.
4. In manchen Fällen kann die Stichprobenverzerrung nicht vermieden werden. Wie bei der Modellierung einer Nachfrageprognoselösung, wo wir das Training des Algorithmus mit Daten aus dem vergangenen Zeitraum verwenden. Bei der Erstellung der Trainings- und Testdatensätze wird die zeitliche Dimension verwendet.
5. Zum Trainieren des Algorithmus wird der Trainingsdatensatz verwendet. Die Zielvariable ist die vorherzusagende Variable beziehungsweise die anstrebende Variable und die unabhängigen Variablen dienen als Leitfaktor.
6. Mit dem Testdatensatz wird die Testgenauigkeit verglichen. Während der Trainingsphase werden die Test-/Validierungsdaten nicht dem Algorithmus zur Verfügung gestellt.
7. Die Testgenauigkeit ist viel wichtiger als die Trainingsgenauigkeit, weil der Algorithmus ungesehenen Daten besser generalisieren sollte. Weshalb der Schwerpunkt auf der Testgenauigkeit ist.
8. Die Genauigkeit ist in manchen Fällen nicht der ausschlaggebende Leistungsindikator, wenn wir den maximieren wollen. Zum Beispiel bei der Vorhersage einer möglichen betrügerischen Kreditkartentransaktion, wo die Genauigkeit nicht die Zielmetrik ist.
9. Während der Entwicklung der Modelle werden verschiedene Eingabeparameter durchgegangen, die sogenannten Hyperparameter. Die Abstimmung dieser wird vorgenommen, um das bestmögliche und stabilste Ergebnis zu erreichen.

   Nach der Fertigstellung des Algorithmus und der Anpassungen dessen wird der Validierungsdatensatz diesem nur einmal ausgesetzt.

Schritt 6: Final werden die verschiedenen Algorithmen, aus dem Schritt 4, verglichen und gegenübergestellt. Die Lösung dieses Schrittes ist das beste oder optimalste Ergebnis. Das Einzige, was zu erledigen ist, ist die Implementierung in der Produktionsumgebung. [Ver20]

Das sind die generellen Schritte für einen Algorithmus des Supervised Learnings.

Das ist das Ende der Diskussion über den Supervised Learnig Algorithmus. In den nächsten Abschnitten werden Unsupervised, Semisupervised und Reinforcement Learning kurz erklärt, damit die LeserInnen die Alternativen in ML kennenlernen.

2.4 Unsupervised Learning

Wie in der Abbildung 2.7 (a) zu sehen ist, sind die Trainingsdaten beim Unsupervised Learning nicht gelabelt. Unsupervised Learning versucht, ohne einen Lehrer zu lernen.

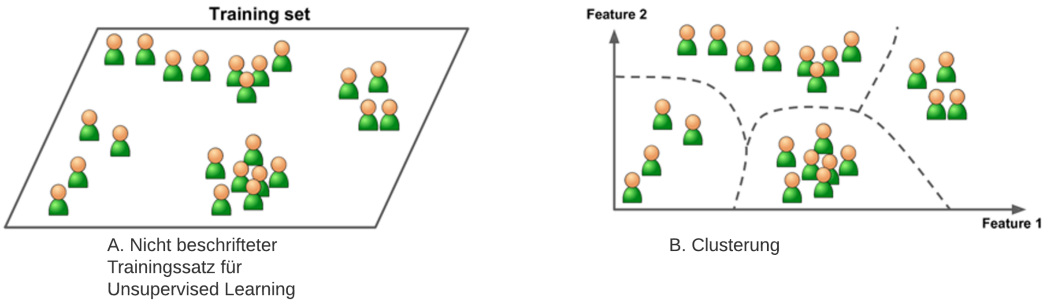

Abbildung 2.7: Unsupervised Learning Clustering Lösung [Gé19]

Als Beispiel nehmen wir an, dass wir eine Menge an Daten über die BesucherInnen unseres Online-Shops haben. Wir möchten Gruppen von ähnlichen BesuchernInnen identifizieren, indem wir einen Clustering-Algorithmus ausführen, der in der Abbildung 2.7 (b) zu sehen ist. Der Algorithmus findet die Gruppen, zu welchen die BesucherInnen gehören, ohne unserer Hilfe. Als Ergebnis könnte rauskommen, dass 50 % unserer BesucherInnen weiblich sind, in der Kategorie Schuhe shoppen und normalerweise dies am Abend tun, während 30 % Kleider-Liebhaber sind, die den Online-Shop am Wochenende besuchen, und so weiter. Wenn wir wollen, können wir auch einen hierarchischen Clustering-Algorithmus verwenden, der uns die herausgefundenen Gruppen in noch kleinere und genauere Gruppen unterteilen kann. Dies kann uns helfen, bestimmte Artikel, um bestimmte Uhrzeiten, auf die Gruppen zuzuschneiden. [Gé19]

2.5 Semisupervised Learning

In der Abbildung 2.8 ist das Semisupervised Learning zu sehen. Semisupervised Learning Algorithmen verwenden teilweise gelabelten Trainingsdaten. Diese Daten bestehen aus einem großen Teil nicht gelabelter Daten und aus einem kleinen Teil gelabelter Daten.

Semisupervised Learning wird gerne bei Smartphone Foto-Apps angewendet, wie bei Google Fotos. Wenn wir zum Beispiel unsere Gruppenfotos aufnehmen und es in einer solchen App speichern, kann diese erkennen, dass auf den Fotos 11, 12 und 13 dieselbe Person X zu sehen ist und auf den Fotos 21, 22 und 23 eine andere Person Y zu sehen ist. Das wäre der Unsupervised Learning Teil, das Clustering. Meisten wird man von solchen Apps gebetet, die erkannten Personen zu benennen. Mit nur einem einzigen Label pro Person, kann die App alle Personen auf jedem neuen Foto benennen, was besonders für die Suche der Fotos von der Person sehr hilfreich ist. In so einem Fall funktioniert das System perfekt, aber in der Praxis nicht immer zu 100 %. Meistens werden oft ein paar Cluster pro Person erstellt und es werden auch Personen verwechselt, die sich ähnlich sehen, sodass man mehrere Labels pro Person vergeben und gegebenenfalls manuell Cluster löschen muss. [Gé19]

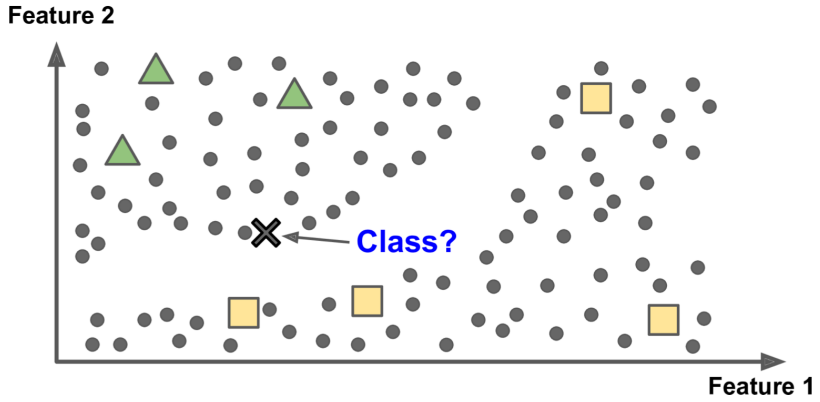

Abbildung 2.8: Semisupervised Learning [Gé19]

2.6 Reinforcement Learning

Reinforcement Learning basiert auf Aktion und Belohnung. Im Vergleich zu Supervised und Unsupervised Learning unterscheidet sich Reinforcement Learning dadurch, dass es nicht von gelabelten und ungelabelten Daten versucht, strukturelle Schlussfolgerungen zu finden. Es verwendet einen Agenten, um die Umwelt zu beobachten und Aktionen anhand dieser zu wählen. Die Aktionen führen dann zu einer Belohnung, die entweder positiv, wenn die Aufgabe richtig gemacht wird, oder negativ, wenn die Aufgaben nicht laut den Vorgaben gemacht werden, ausfallen kann. Dieser Vorgang ist in der Abbildung 2.9 zu sehen.

Mit den Belohnungen und Bestrafungen, lernt der Agent im Laufe der Zeit, welche Strategie die beste ist, um den größten Nutzen zu erzielen.

Die Strategie oder auch Policy genannt, legt fest, welche Aktion der Agent in einer bestimmten Situation wählen sollte.

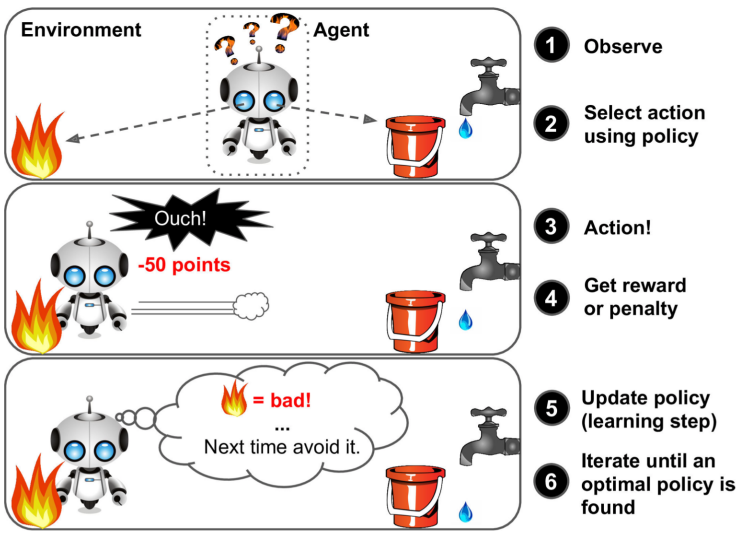

Abbildung 2.9: Reinforcement Learning [Gé19]

Reinforcement Learning Algorithmen werden gerne bei Robotern angewendet, um ihnen das Laufen beizubringen. Das bekannteste Beispiel für Reinforcement Learning, ist die Implemen-

tierung von dem Programm AlphaGo von DeepMind. Das Team machte 2017 Schlagzeilen, als sie den Weltmeister von Go besiegt haben. Dies hat das Programm geschafft, indem es Millionen von Spielen analysierte und dann viele Spiele gegen sich selbst spielte. Beachtenswert ist es, dass das Lernen, während dem Spiel gegen den Champion ausgeschaltet war. Das Programm wendete einfach die erlernten Strategien an. [Gé19]

3 Virtual Reality für Unity

Während virtuelle Realität kein neues Konzept ist, sind einige der kürzlich für die Spieleindustrie eingeführten VR-Technologien doch sehr innovativ.

In diesem Kapitel wird auf die Einsatzmöglichkeit und auf die Funktionsweise von VR eingegangen. Zum Schluss wird erklärt, wie VR in das Projekt im Kapitel 4 eingebunden wurde. Zunächst wird aber erklärt, was für Realität wir heutzutage mit Technologien erzeugen können.

3.1 Realitäten

Virtual Reality

Das Thema Virtual Reality, auch bekannt als VR, hat in den letzten fünf Jahren immer mehr an Popularität gewonnen, besonders jetzt, wo verschiedene Technologieunternehmen sich für die Weiterentwicklung eingesetzt haben, wie Meta (Facebook) mit derer Metaverse.

Die Definition von VR ist sehr weit gefächert und beinhaltet alles, was in unserer digitalen Welt passiert, wie soziales Networking, Online-Marketing und so weiter. Für die meisten User ist VR momentan, nur ein Zwecks für die Benutzung von computergenerierter, simulierter Darstellung von natürlichen oder Fantasiewelten, in denen sie eintauchen können. Dabei können sie mit den simulierten Umgebungen und den darinnen enthaltenen Objekten interagieren, sei es durch das Sehen, Hören oder in manchen Fällen auch Fühlen. VR bietet den Usern die Möglichkeit vollständig in interaktive Umgebungen einzutauchen und mit ihnen zu agieren, wie durch das Umsehen, bewegen und durch die Interaktion mit Objekten, ohne die Illusion aufzugeben, sich in einer Simulation zu befinden. Für das Eintauchen in die VR-Welt werden Headsets, Eingabegeräte wie Handschuhe, Kontroller und Ganzkörperanzüge verwendet. [dROW17]

Erweiterte Realität

Die erweiterte Realität, oder auch AR (Augmented Reality) genannt, ist eine Anwendung, die die natürliche Welt mit Informationen oder virtuellen Objekten erweitert. Abwendungen wie Apple ARKit ermöglichen den Usern, Objekte in die Welt zu setzen, wie zum Beispiel bei online shoppen von Möbeln oder auch bei AR-Smartphone Spielen. [Tim19]

Mixed Realität

Bei der Mixed Realität, auch MR genannten, werden Techniken von VR und AR verbunden, um die natürliche Umgebung mit virtuellen Objekten zu erweitern. Dabei werden Kameras und verschieden Sensoren verwendet, um Objekte mit einer virtuellen Schicht zu besehen. Dies ist zum Beispiel beim Verkauf beziehungsweise bei der Konfiguration von Autos sehr nützlich, wo dann die User live ihre Wunschkonfigurationen sehen und anfassen können. [Tim19]

3.2 Anwendungsbereiche von VR

Die Möglichkeiten von VR sind nicht nur auf die Spieleindustrie beschränkt, sondern auch auf andere Branchen. Viele Technikunternehmen haben schon das mögliche Potenzial von VR erkannt und versuchen dieses auch für sich zu gewinnen. Momentan steckt VR noch in den Kindesschuhen, und wird weiter erforscht, um für die Verbraucher und Unternehmen diese Technologie erschwinglicher, brauchbarer und verfügbarer zu machen.

Medizinische und psychische Gesundheit

In der Medizinbranche sind die Einsätze von 3D-Visualisierungen nichts neues, es werden verschieden Arten von Aufnahmen für die Diagnose und Buchhandlung genutzt wie MRT, CT und Röntgenbilder für Diagnose von PatientInnen. Neu dazu ist die Benutzung von VR für die Behandlung von verschiedenen Krankheiten, wie in der Neurowissenschaft, der Psychologie und der Schmerzbehandlung. VR hat besonders wirksam bei Behandlungen von Phobien, bipolaren Störungen, stressbedingten Störungen und anderen psychologischen Erkrankungen erwiesen. Der Einsatz von VR hat sich in dem Gesundheitswesen und der Patientenbehandlung als nützlich bewiesen. [Cou22]

Fahrzeugtechnik und Design

Besonders in den design- und technischen-Anwendungen hat sich VR als effektiv erwiesen. Mit VR wird der Designschritt von physischen Prototypen um einige Schritte schneller in die Realität umgesetzt. Der Einsatz von VR ermöglicht in der Automobilindustrie, die Visualisierung von den Produkten, bevor diese in physische Prototypen fabriziert werden. Den größten Nutzen vom Einsatz von VR haben die Designer, da die Herstellung von physischen Prototypen zu viel Zeit auf sich nimmt und diese in VR viel schneller erfolgen kann. Die Erstellung der Prototypen in VR ermöglicht dem Designer ein schnelleres Handeln bei wichtigen Designentscheidungen. VR wird auch nicht nur für den Herstellungprozess verwendet, sondern auch bei dem Verkaufsschritt mit den Kunden, wo diese ihre Fahrzeugkonfigurationen an Ort und Stelle sehen können, bevor sie sich für eine Konfiguration entscheiden. [Cou22]

Ausbildung und Schulung

Ein weiterer Bereich, wo sich VR nützlich erwiesen hat, ist in der Ausbildung und Schulung. Mit VR werden den Nutzern die Ideen direkt und praktisch vermittelt, währen bei anderen altmodischen Arten Medien verwendet werden, um Ideen verbal zu vermitteln. VR ermöglicht einen sofort, durch die Kombination aus Interaktion und realem Eintauchen einen effizient Schulungsstill. Die Einsatzbereiche, in denen die Ausbildung über VR erfolgt, wären zum Beispiel bei der Flugausbildung, Katastrophentraining und so weiter. In den letzten Jahren nahm auch der Einsatz von VR im Bildungswesen stark zu, wie bei der Hilfe zum erlernen von neuen Sprachen, indem VR-Umgebungen mit Anwendungen wie Linguisticator genutzt werden bei SchülerInnen. [Cou22]

3.3 Funktionsweise von VR

Die Funktionsweise von VR basiert auf zwei Konzepten: Immersion und Präsenz. Immersion wird durch die simulierten Umgebungselemente wie Geräusche und Design erzeugt. Präsenz

ist das Gefühl, in der Welt zu sein.

Es gibt zwei weitere Faktoren, die für die Funktionsweise wichtig sind: das 3D-Sehen und das Head-Tracking.

3D-Sehen

Der Mensch verfügt über die Fähigkeit des 3D-Sehens, da er zwei Augen hat, welche unterschiedliche Blickwinkel auf ein Objekt bieten und das Gehirn anschließend die räumliche Beschaffenheit berechnet.

Bei der VR-Brille werden lediglich zwei unterschiedliche Bilder erzeugt. Diese werden dem Betrachter, in leichter Verschiebung voneinander, je nach Auge gezeigt. Das Gehirn berechnet anschließend die räumliche Differenz der beiden Bilder und erzeugt so die Illusion von Tiefe, also das 3D-Sehen. Um ein scharfes Bild zu erhalten, ist es notwendig, vor jedes Auge eine Linse zu setzen.

Es ist wichtig, bei der Entwicklung auf die ”Pixel per inch” und die ”Frames per second” zu achten, um ein flimmerfreies Ergebnis zu erzielen. Die ”Pixel per inch” sind wesentlich, um das ”Screen door effect” zu minimieren. Wenn die ”Frames per second” - Rate zu gering ist, werden die Bewegungen nicht korrekt dargestellt. [Tim19]

Head-Tracking

Die Nutzung von Head-Tracking bei VR-Brillen basiert auf einer Bewegungserfassung. Durch Nachverfolgung der Kopfbewegung erlangt man drei weitere Freiheitsgrade in der Rotation und Ebene. Sollte man sich innerhalb dieser Freiheitsgrade in eine bestimmte Richtung bewegen, so passt sich auch das Bild an. Zwei Arten des Trackings existieren: „inside-out tracking“ und „outside-in“. Beim „inside-out tracking“ wird mithilfe von Sensoren, welche an der Kamera angebracht sind, die Position verfolgt; im Gegensatz dazu geschieht dies beim „outside-in tracking“ mithilfe extra aufgestellter Sensoren.

Die Verzögerung von der erfassten Position bis zum fertigen Bild wird als Latenz bezeichnet. Diese kann das Spielen unangenehm machen, da das Bild immer hinter der aktuellen Position hinterher hängt. Daher sollte die Latenzzeit so minimal wie möglich sein. Bewegungskrankheit, die durch solche Verzögerungen auftreten kann, sollte ebenfalls minimiert werden, um den Nutzern eine optimale Erfahrung zu bieten. [Tim19]

3.4 VR-Integrationen für Unity

Unity ist eine Entwicklungsumgebung für 2D und 3D Inhalte, die besonders in der Spieleentwicklung eingesetzt wird, und eine Cross-Plattform, die es ermöglicht Projekte für verschiedene Plattformen zu entwickeln. Auch wenn Unity hauptsächlich für die Spieleentwicklung verwendet wird, ist die Plattform auch für die Simulationen von Robotern für die Forschung von ML sehr nützlich. Mit dem Package Manager von Unity, kann der Editor für verschiedene Vorhaben erweitert werden, was die Implementierung von anderen Softwares ermöglicht. In der Abbildung 3.1 ist die Umgebung von Unity zu sehen. [Tec]

Bevor die Umsetzung der Integration von einer VR-Brille in das Unityprojekt 4 beschrieben wird, ist es wichtig zuerst die Grundlagen von Unity und derer Umgebung zu verstehen. Es werden jetzt die wichtigsten Schritte und Aspekte für die Nutzung der Unity-Umgebung bei einem Projekt erklärt.

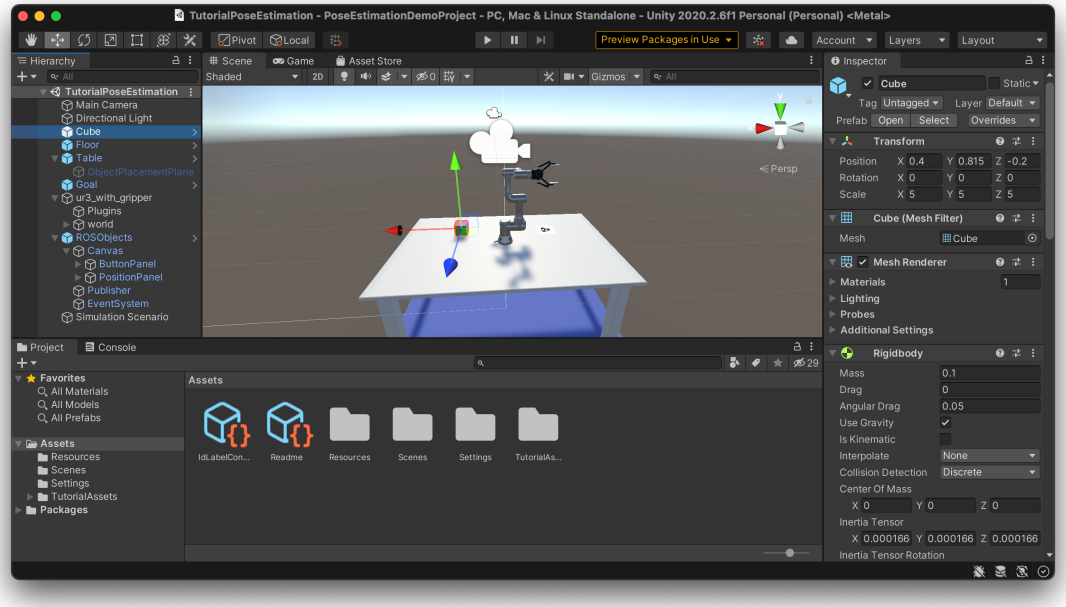

Abbildung 3.1: Entwicklungsumgebung vom Unityprojekt

Installation von Unity

Zuerst einmal müssen wir Unity herunterladen. Dafür gehen wir auf die Seite: `https://store.unity.com` und laden uns die kostenlose Personal-Version herunter.

Die Personal-Version von Unity ist kostenlos und für unsere Zwecke in diesem Projekt völlig ausreichend. Der einzige Unterschied zur Plus-Version besteht darin, dass bei der kostenlosen Version der Schriftzug "Made with Unity" erscheint, während wir bei der Plus-Version einen eigenen Startbildschirm erstellen können.

Der Unity Download-Assistent fordert uns während der Installation von Unity auf, die Komponenten des Editors auszuwählen, die wir installieren möchten. Es ist wichtig, dass die folgenden Komponenten ausgewählt sind: Unity 2020.2.6 (da ab dieser Version die Implementierung von VR-Brillen unterstützt wird), Dokumentation, Standard-Assets und Beispielprojekt.

Navigieren auf der Unity-Oberfläche

Die Symbolleiste am oberen Rand des Unity-Editors enthält die Transformationswerkzeuge, die Bedienelemente für die Werkzeuggriffe, die Bedienelemente für Wiedergabe, Pause und Play, das Auswahlfeld für Konten, das Auswahlfeld für Ebenen und das Auswahlfeld für das Layout.

In der Unity-Benutzeroberfläche, wie in Abbildung 3.1 zu sehen, befinden sich verschiedene Fensteransichten, die wir uns im Folgenden ansehen werden. [Hal18]

- **Scene View**

  Die Scene-Ansicht ist ein wesentlicher Bestandteil des Unity-Editors, und in ihr konstruieren wir unser Spiel. Die Szenen selbst sind logische Einheiten, die aus GameObjects bestehen und alle relevanten Funktionen für unsere Szene bereitstellen. Es ist wichtig

zu wissen, dass jedes Objekt in einer Unity-Szene ein GameObject ist.

- **Game View**

  Die Spielansicht ist die Darstellung des Spiels aus der Perspektive der aktiven Kamera. Dies ist auch der Ort, wo wir das tatsächliche Spiel sehen und spielen. Wir können es im Unity-Editor erstellen und ausführen, als eine eigenständige Anwendung, in einem Webbrowser oder auf einer VR-Brille.

- **Asset Store**

  Ein wesentlicher Grund, Unity für die Spieleentwicklung zu wählen, ist der Unity Asset Store. Der Unity Asset Store ist eine Online-Galerie, in der Künstler, Entwickler und Inhaltsersteller Inhalte hochladen können, um sie zu kaufen und zu verkaufen. Der Unity-Editor hat eine integrierte Registerkarte, die eine Verbindung zum Asset Store herstellt. Wenn etwas aus dem Asset Store benötigt wird, kann es nützlich sein, diesen Bereich im Layout zu öffnen, aber es ist nicht unbedingt notwendig. Es kann über die Seite `https://assetstore.unity.com` auf den Asset Store zugegriffen werden.

- **Hierarchy Window**

  Das Hierarchiefenster in Unity gibt eine Liste aller Objekte in der aktuellen Szene in einem hierarchischen Format wieder. Über das Dropdown-Menü ”Erstellen” in der oberen linken Ecke können neue GameObjects erstellt werden. Durch das Suchfeld können Entwickler gezielt nach bestimmten GameObjects anhand ihres Namens suchen. GameObjects können in Unity andere GameObjects in einer sogenannten ”Eltern-Kind” -Beziehung enthalten.

- **Project Window**

  Durch das Projektfenster haben wir einen Überblick über den kompletten Inhalt des Ordners Assets. Es ist nützlich, im Projektfenster Ordner zu erstellen, um Elemente wie Texturen, Szenen, Materialien, Skripte und Modelle zu organisieren.

- **Console View**

  Die Konsolenansicht ist ein wichtiges Hilfsmittel für das Debugging von Unity-Anwendungen. C# -Skriptfunktionen ermöglichen die Ausgabe von Informationen in der Konsolenansicht zur Laufzeit, um das Debugging zu erleichtern. Die Konsolenansicht bietet verschiedene Formen der Ausgabe, die mit den drei Schaltflächen oben rechts in der Konsolenansicht ein- und ausgeschaltet werden können.

- **Inspector Window**

  Wir können den Inspector verwenden, um die angehängten Komponenten und ihre Eigenschaften zu sehen und zu bearbeiten. Die Szenen in Unity setzen sich aus GameObjects zusammen, welche aus Komponenten wie Skripte, Meshes, Collidern und weiteren Elemente bestehen. Um die angehängten Komponenten sowie deren Eigenschaften anzusehen und zu bearbeiten, können wir ein GameObject auswählen und in dem Inspector er direkt umsetzen. Es gibt verschiedene Techniken, mit den man benutzerdefinierte Eigenschaften für GameObjects erstellen kann, die dann geändert werden können. Auch lassen sich die Eigenschaften von Kameras, Assets und Materialien mithilfe des Inspectors anzeigen und ändern. [Hal18]

3.5 VR Nutzung im Projekt

Wie vorhin erklärt, kann der Game View auf eine VR-Brille übertragen werden. Dies hat den besonderen Vorteil, dass während der Entwicklung von Spielen, beziehungsweise auch bei unserem Projekt 4, Änderungen vorgenommen und getestet werden können, ohne dass das Projekt kompiliert und übertragen werden muss auf die VR-Brille. Ohne der VR-Integration in Unity 2020.2.6 wäre das Testen deutlich mühsamer.

Für das Umsetzen von VR im Projekt wurde die Meta Quest 2, bekannt auch als Oculus Quest 2, verwendet. In der Tabelle 3.1 ist das Toolkit der Quest 2 zu sehen, das in Unity verwendet wurde. [fD]

Name	Oculus Integration
Entwickler	Facebook Technologies, LLC
Unity Unterstützung	2018.4.3 oder höher

Tabelle 3.1: Meta Quest Toolkit

Es werden jetzt die Schritte für die Integration der Quest 2 in das Projekt [Kapitel Unity Projekt] erklärt.

1. Um die Oculus für den Game View verwenden zu können, muss das Programm Oculus Link. Oculus Link ermöglicht, mit einem USB-C Kabel, die Nutzung der Brille mit einem Windows Computer.
2. Als Nächstes müssen erforderliche Pakete installiert werden. Eines dieser Pakete ist, befindet sich momentan in der Preview und wird nicht im Unity Package Manager angezeigt. Um dies zu umgehen, muss das Feld ”Enable Preview Packages” eingeschaltet sein. Das Feld findet sich unter dem

   Reiter *Edit > Project Settings > Package Manager*.
3. Das wichtigste Paket, das benötigt wird, ist das ”Oculus XR Plugin”. Dieses Paket ist in dem Toolkit ”Oculus Integration” mit inbegriffen. Die Oculus Integration kann über den Asset Store installiert werden.
4. Es muss jetzt sichergestellt werden, dass das Plugin im Projekt aktiviert ist.

   Unter *Edit > Project Settings > XR Plug-in Management*, müssen ”Initialize XR on Startup” und ”Oculus” aktiviert sein.
5. Als nächstes muss man das Headset aufsetzen und im Einstellungsmenü, die Option ”Oculus Link” einschalten.
6. Als letzter Schritt muss man in Unity, bei dem Projekt den Play Modus aktivieren.

Nach diesen sechs Schritten sollte man dann, den Game View mit dem Headset sehen können.

4 Unityprojekt: Roboter mit Supervised Learing

Dieses Kapitel dient für die Anwendung der besprochenen Theorie, im Kapitel 2. Dafür wird ein Projekt vorgezeigt und die wichtigsten Schritte werden erklärt, wie Superised Learning zum Einsatz in dem Projekt verwendet wurde, um den Lesenden die Thematik näherzubringen. Das verwendete Roboter- und ML-Modell wurde von dem frei zur verfügbar gestellten GitHub-Projekt [UTc] übernommen.

Für das Projekt wird die Entwicklungsumgebung Unity verwendet, um den Roboterarm in einer virtuellen Umgebung zu simulieren. Das Ziel ist es, den Lesenden anhand des Projektes zu zeigen, das Supervised Learning eine größere Spannweite an Anwendungsbereichen hat, als nur Datenanalyse und Verarbeitung von großen Datenmengen wie in dem Hausbeispiel aus dem Kapitel 2 beschrieben. Das Supervised Learning wird in diesem Projekt für das Deep Learning verwendet, um die Bildklassifizierung beziehungsweise die Objekterkennung durchzuführen. Das Netzwerk des Deep Learnings wird zu dem Vorhersagen eines Labels verwendet (Eingabe und Ausgabe sind bekannt). Da die Labels von den aufgenommenen Bildern bekannt sind, wird mit dem Netzwerk, die Fehlerrate gesenkt, weshalb es das Deep Learning-Modell "überwacht" ist.

Die Welt und die Möglichkeiten, die die Robotik bittet, ist groß und muss noch erforscht werden. Sei es bei der Positionierung in Raum oder auch von Objekten, das richtige umgehen mit Sensorrauschen und weiteren Hindernissen, müssen Roboter verstehen und richtig mit diesen Hindernissen handeln, um für den Einsatz in unserer Welt robust und genau zu sein. Mit dem Projekt wird gezeigt, wie das Unity Computer Vision Perception Package verwendet werden kann, um Daten zu sammeln und ein ML Modell zu trainieren, um vorhersagengen von Positionen für bestimmte Objekte treffen zu können. Das trainierte Modell wird dann in einen virtuellen UR3-Roboterarm in Unity integrieren, um dann den Task des Aufhebens und Platzierens eines Objekts mit unbekannten und beliebigen Posen darzustellen beziehungsweise zu simulieren.

In der realen Welt arbeiten die Roboter in dynamischen Umgebungen, an denen sie sich anpassen müssen, um für uns in bestimmten Szenarien dienen zu können. Damit sie von Anwendung sein können, müssen die Roboter Objekte erkennen können und mit denen interagieren können. Um mit den Objekten interagieren zu können, müssen die Roboter ein Verständnis über die Position und Orientierung von den Objekten im Koordinatensystem haben, was auch als "Pose" genannt wird. Benutzerdefinierte Referenzmarker und Computer-Vision-Techniken wurden in früheren Ansätzen zur Ermittlung der Pose herangenommen. Diese Herangehensweise ist hilfreich für bestimmte Umgebungen, leider versagt sie schnell bei Umgebungen, die dynamisch sind oder die von der erwarteten Realität abweicht. Mit neuen ML Techniken kann die Lücke, die mit traditionellen Computer Vision Ansätzen entstanden ist, gefühlt werden. Bei den neuen Methoden werden Modelle erstellt, die mit vielen möglichen Beispielen sich beibringen, Vorhersagen für Ausgaben anhand von bestimmten Eingaben zu treffen. [UTa]

Für die Umsetzung des Projektes werden Bilder und Posenmarker verwendet, um das Modell, welches die Pose von den Objekten vorhersagt, zu trainieren. Mit dem Modell ist

der Roboter dann in der Lage, die Pose von einem Objekt aus einem Bild, das es noch nie gesehen hat, vorherzusagen. Damit das Modell leistungsfähig ist, werden zich tausende Bilder von dem Objekt gesammelt und gelabelt. In der realen Welt wird dieses Vorhaben lange, mühsam, teuer und in manchen Fällen auch schwierig zum umzusetzen sein. Wenn man es trotzdem umsetzt und die Daten sammelt und labelt, ist dieser Prozess sehr einseitig, langwierig und besonders fehleranfällig. Die Frage, die sich stellt, ist, wie wir dann für das Training eines Modells, die Daten effizient und schnell sammeln und labeln können.

Mit Unity Computer Vision. Unity Computer Vision kann synthetische Daten effizient und schnell für ML Modelle generieren. In diesem Projekt wird dieses angewendet, um in Unity automatisch gelabelte Daten zu generieren, um dann das ML Modell mit denen zu trainieren. Das Modell wird anschließend auf einen simulierten UR3-Roboterarm mit dem Robot Operating System (ROS) in Unity eingesetzt, um dann den Task des Aufhebens und Platzierens eines Objekts zu bewältigen.

Bevor die Umsetzung des Projektes weiter beschrieben wird, werden zuerst die Begrifflichkeiten der Tools, die im Projekt verwendet werden, kurz aufgefasst.

Unity Computer Vision Perception Package

Mit dem Perception Package Toolkit können wir große Datensätze für das Training und die Validierung von Computer Vision verwenden. Das Tool konzentriert sich auf eine Handvoll von kamerabasierten Anwendungsfällen des ML. [UTb]

UR3-Roboterarm

Universal Robots (UR) ist ein Unternehmen aus Dänemark, das sich auf die Herstellung von industriellen, kollaborierenden Leichtbaurobotern spezialisiert. Der UR3 ist ein kleiner Tischroboter, der für leichte Montageaufgaben und Werkbankszenarios eingesetzt wird. Der UR3 kann mit dem Package "URDF Importer" (`https://github.com/Unity-Technologies/URDF-Importer`) kostenlose in Unity heruntergeladen werden. [UR]

Robot Operating System (ROS)

ROS ist ein Open-Source-Softwareentwicklungskit für Robotik Anwendungen. ROS bietet Entwicklern aus allen Branchen eine Standard-Softwareplattform, die sie von der Forschung und Prototypentwicklung bis hin zur Bereitstellung und Produktion begleitet. Für Unity kann es hier heruntergeladen werden (`https://github.com/Unity-Technologies/ROS-TCP-Connector`). [ROS]

4.1 Synthetische Datengewinnung

Die Entwicklungsumgebungen Unity ist neben der Spieleentwicklung auch ein hilfreiches Tool für die Datenerfassung, wie durch die Erzeugung von synthetischen Daten. Für dieses Problem kann Unity Computer Vision verwendet werden. Unity Computer Vision ermöglicht es einem, mit geringem Aufwand große Mengen an gelabelten und vielseitigen Daten zu sammeln. In dem Projekt werden tausende Beispielbilder von dem Objekt, Würfel, gesammelt. Das Objekt wird in verschiedenen Posen und Lichtverhältnissen aufgenommen. Für die Umsetzung beziehungsweise die Generierung der unterschiedlichen Aufnahmen wird die Methode Domänenrandomisierung angewendet. [TFR+17]

Die Domänenrandomisierung wird verwendetet, um bei dem Zeitpunkt des Trainings eine ausreichende simulierte Variabilität zu schaffen, damit dann das Trainierte Modell zum Zeitpunkt der Überprüfung auf reale Daten verallgemeinern kann. Um die Realität nachzuahmen, wird nicht das Modell in einer einzigen festen Umgebung trainiert. Es ist von Bedeutung das die Daten eine ausreichende variationen aufweisen, dafür werden die verschiedenen Aspekte der Umgebung während des Trainings randomisiert. Durch die Randomisierung wird das ML-Modell gezwungen, mit vielen unterschiedlichen kleinen visuellen Variationen zu arbeiten, was es robuster macht. Es werden die folgenden Aspekte der Domäne bei dem Training der Proben randomisiert:

- Objekt-Drehung
- Objekt-Positionen
- Licht Zufallsgenerator

Die Vielfalt der Daten führt dazu, dass in der Regel ein robustes Deep Learning Modell geschaffen wird.

Ohne der Entwicklungsumgebungen Unity müssten wir den Würfel manuell bewegen und ein Foto machen, um die Daten für das Modell zu sammeln. Das Modell im Projekt wurde mit 30.000 Bildern trainiert. Täten wir es mit der Hand, würden wir über 40 Stunden brauchen, wenn die Aufnahme 5 Sekunden pro Bild gedauert hätte, um die Daten zu sammeln. Dies beinhaltet nicht die Zeit, die das Labeling in Anspruch genommen hätte, da es durchgeführt werden muss.

Mit dem Unity Computer Vision könne wir innerhalb von wenigen Minuten tausenden Datensätze sammeln, wie bei dem Projekt, wo 30.000 Trainingsbilder und weitere 3.000 Validierungsbilder mit den entsprechenden Labels erstellt wurden. Bei der Erstellung und Sammlung der Trainingsbilder, für das Projekt, war der Tisch, der Roboter und die Position der Kamera fixiert. Es wurde nur die Position des Würfels und die Beleuchtung in jeder Aufnahme zufällig randomisiert. Bei der Erstellung der Bilder wurden sie dann entsprechend gelabelt und in einer JSON-Datei gespeichert. Die Daten, die gespeichert wurden, waren die Posen, die durch die 3D-Position (x,y,z) und einer Quaternion-Ausrichtung (qx,qy,qz,qw) beschrieben wurden.

Die Möglichkeiten von Unity Computer Vision sind nicht nur beschränkt auf die Variierung der Pose des Würfels und auf die Variierung der Beleuchtung in der Umgebung vom Projekt. Unity Computer Vision würde noch im Projekt die Variierung von weiter Aspekten der Szene erlauben, wie zum Beispiel die Tischneigung zur Roboterhand, oder die Höhenverstellung des Würfels. Bei dem Durchführen der Posenschätzung verwenden wir im Projekt Supervised Machine Learning Techniken, die die Daten analysieren und ein trainiertes Modell erstellen.

4.1.1 Modelltraining für die Vorhersage der Pose

Wie schon im Kapitel 2 erwähnt, wird bei Supervised Learning ein Modell trainiert, um bestimmte Ergebnisse vorherzusagen. Für das Training verwendet das Modell eine Reihe von Eingaben und entsprechenden Ausgaben für das Trainieren. In unserem Fall werden Bilder und Posen-Labels verwendet für das Training. Das Projekt verwendet ein Convolutional Neural Network (CNN) oder auf Deutsch ”faltendes neuronales Netzwerk”, um die Position eines Objekts vorherzusagen.

Ein CCN ist ein Deep Learning Algorithmus, der verschieden Aspekte bei Eingangsbildern eine Bedeutung (erlernbare Gewichtungen und Verzerrungen) zuweist und dann in der Lage ist, zwischen ihnen zu unterscheiden. Die Vorverarbeitung, die bei einem CNN erforderlich ist, ist viel geringer als bei anderen Klassifizierungsalgorithmen. Im Vergleich zu primitiven

Methoden, wo die Filter von Hand entwickelt werden, werden bei CNNs diese Filter mit Trainings erlernt. [Sah18]

Für die Architektur des CCNs wurde dem menschlichen Gehirn und dessen Konnektivitätsmuster der Neuronen zur Inspiration genommen. In den sogenannten rezeptiven Feldern reagieren nur einzelne Neuronen auf Reize in einem begrenzten Bereich des Gesichtsfeldes. Mit einer Ansammlung solcher Felder, die sich überlappen, ist man dann in der Lage den gesamten visuellen Bereich abzudecken. In der Abbildung 4.1 ist ein Deep Learning Modellarchitektur zu sehen, die das CNN verwendet, um die Pose im Projekt zu schätzen.

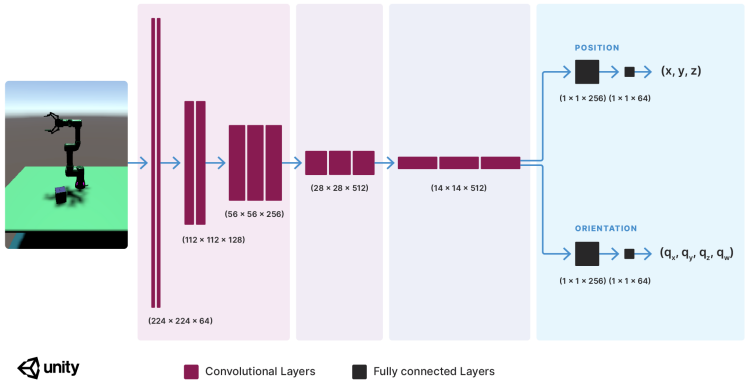

Abbildung 4.1: Architektur des Deep Learning Modells mit CNN, für die Posen Schätzung. [UTa]

Da die 3D-Postion des Würfels den Projektentwicklern wichtig war (GitHub-Projektlink [UTc]), wurde die Arbeit von [TBD+17] herangenommen und erweitert, um auch die Ausrichtung des Würfels bei der Ausgabe von dem Netzwerk einzubeziehen.

Das trainierte Modell ist dann in der Lage, die Position des Würfels bis hin zu einer Genauigkeit von 1 cm, und auch die Ausrichtung mit einer Genauigkeit von 3 Grad vorauszusagen.

Die Frage ist, ob diese Genauigkeit in der Simulation auch genug ausreichend ist, um den Task des Aufhebens und Platzierens des Objekts zu bewältigen. Im letzten Abschnitt wird dann das Ergebnis des gemessenen Modells präsentiert. Zunächst aber die Implementierung von ROS.

4.1.2 Bewegungsplanung mit ROS in Unity

Der Roboterarm, der für das Projekt verwendet wurde, ist der UR3-Roboterarm von dem Unternehmen Universal Robots mit einem Robotiq 2F-140-Greifer. Der Robotiq 2F-140-Greifer ist eine anpassungsfähige Zwei-Finger-Greiflösung, die optimal als ein End-of-Arm-Werkzeug ist. Der Greifer eignet sich optimal für Umgebungen mit hohem Wechselaufkommen und geringem Volumen.

Die beiden Komponenten wurden mit dem Unity Robotics URDF Importer Package in die Unity-Szene importiert. Für die Kommunikation wird das Unity Robotics ROS-TCP Connector Package verwendet. Das ROS Movelt Package wird für die Bewegungssteuerung und -planung des Roboters verwendet.

Mit dem trainierten Deep Learning-Modell können wir die Pose des Würfels genau vorhersagen. Diese vorhergesagte Pose wird dann als die Zielpose in dem Task des Aufhebens und Platzierens des Objekts verwendet.

Im Vergleich zu traditionellen Vorgehensweisen, wo Objekte von Robotern aufgehoben werden, ist die tatsächliche Pose des Zielobjekts wichtig für die Durchführung. Während in unserem Projekt, der Roboter das Aufheben und Platzieren ohne die Vorkenntnis der Pose des Würfels ausführt und die Pose von dem Deep Learning-Modell vorhergesagt bekommt.

Der Task besteht aus 4 Schritten:

1. In Unity wird ein Bild von dem Zielwürfel aufgenommen.
2. Anschließend wird das Bild an unser trainiertes Deep Learning-Modell geschickt. Die Ausgabe ist dann eine Vorhersage der möglichen Pose.
3. Die vorhergesagte Pose wird dann an den Movelt-Bewegungsplaner geschickt, um einen möglichen Plan aus der Pose zu berechnen.
4. Danach sendet der ROS eine Trajektorie an Unity, die dann vom Roboter ausgeführt werden muss, um den Würfel holen zu können.

Bei der Durchführung des Task, wird der Würfel bei jeder Iteration an eine zufällige Stelle bewegt. Anzumerken ist, dass in der Unity Szene die aktuelle tatsächliche Position des Würfels angezeigt wird, um als verglich zu der Vorhersage zu dienen. In der realen Welt steht diese Information dem User nicht zur Verfügung. Bei der Umsetzung des Projekts auf einen realen Roboter, muss die Position des Würfels nur durch sensorische Daten bestimmt werden. Dies macht das umgesetzte Posenschätzungsmodell des Projektes möglich.

4.1.3 Ergebnisse

Das Projekt zeigt, wie man eine Schätzung der Objektposition ausführt. Dafür wird erklärt, wie mit Unity synthetische Daten erzeugt wurden, um ein Deep Learning-Modell zu trainieren. Welches dann mit der Verbindung zu ROS verwendet wurde, um unseren Roboterarm in der Simulation das Problem des Aufnehmens und Hinlegens zu lösen. Weiters wurde das Unity Computer-Vision-Tool verwendet, um die synthetischen, gelabelten Trainingsdaten zu erzeugen.

In der Tabelle 4.1.a ist die Leistungsfähigkeit des Models für die Vorhersage der Pose des Würfels angeführt und in der Tabelle 4.1.b sind die Resultate der Erfolge des Greifens in der Simulation, mit unserem trainierten Modell gemessen. Für das Ergebnis 4.1.b wurden 100 Versuche durchgeführt.

A. Leistungsfähigkeit	Trainingsfehler	Validierungsfehler
Übersetzung	0,012 (12 % der Größe des Würfels)	0,01 (10 % der Größe des Würfels)
Orientation (Radiant)	0,06	0,05
B. Resultate	Erfolg	Misserfolge
	87	13

Tabelle 4.1: Ergebnis der Leistungsfähigkeit und Erfüllung der Aufgabe

Wie aus der Tabelle zu entnehmen, hat der simulierte Roboter in 87 % der Fälle zuverlässig den Würfel in Unity aufgehoben.

Die genauen Ausführungsschritte des Projekts sind in [UTc] dokumentiert. Mit diesen und vielen weiten Tools, die für Unity entwickelt wurden, kann das Projekt in verschiedene Richtungen erweitert werden.

5 Zusammenfassung Ausblick

Maschinelles Lernen ist ein unglaublich vielseitiges Gebiet mit zahlreichen Anwendungsmöglichkeiten. ML ist besonders nützlich für die Vorhersage von Mustern und Trends. Durch das Lernen von Mustern in großen Datensätzen können ML-Algorithmen Muster erkennen und Trends vorhersagen, die auf den ersten Blick nicht offensichtlich sind. Dieser Technologie machen sich Data Science bis hin zu der Robotik zunutze.

Vor allem eröffnet Supervised Learning neue Türen und ermöglicht es, riesige Datensätze, die in der Klassifizierung, Regression oder bei der Bildklassifizierung, mit Verbindung mit Deep Learning, verwendet werden zu verarbeiten.

Dank Unity und seien vielen nützlichen Tools kann man in kürzester Zeit Robotersimulationen erstellen – perfekt für alle, die schnell und einfach in die Welt der Robotik eintauchen wollen. Besonders das Unity Computer Vision Perception Package ist für das Training eines Roboters hilfreich, da dieses für die Sammlung von großen Datensätzen für das Training von verschieden Vorhaben, sowie für die Validierung von Computer Vision verwendet werden kann. Zusätzlich mit der Verbindung von Supervised Learning und der Domänenrandomisierung, die die Umgebung für die Sammlung von synthetischen Daten ermöglicht, ist es möglich Daten für das Training eines Modells eines Roboters innerhalb kürzester Zeit zu sammeln. Mit dem Robot Operating System Entwicklungskit können dann die erstellten Modelle für die Bewegungsplanung und -steuerung eines Industrieroboters in der Simulation sowie in der Realität verwendet werden.

Das Kapitel 4 zeigt die wichtigsten Schritte, wie der Roboter in Unity verwirklicht wurde. Es wird dabei zunächst erläutert, wie die Roboterhardware an die Unity-Umgebung angebunden wird. Dann wird gezeigt, wie ein einfache das Szenario in Unity aufgebaut wird, damit der Roboter ein Objekt aufheben und platzieren kann. Zuletzt wird erläutert, wie weitere Funktionen des Roboters in Unity integriert werden können. Das Ergebnis des Projekts zeigt, dass die Verwendung von Supervised Learnings für das Trainieren des Modells, eine 87 % Genauigkeit des Tasks ermöglicht.

Im Kapitel 3 wurde zuvor erklärt, wie Virtual Reality im Projekt umgesetzt wurde. Zuvor wurden die Einsatzmöglichkeiten von Virtual Reality und die Funktionsweise von VR-Brillen erläutert. Um die Umsetzung des Projekts sowie die Implementierung von VR in dieser, wurde zunächst die Entwicklungsumgebung von Unity erläutert. Es wurde dabei die wichtigsten Schritte und Aspekte für die Nutzung der Unity-Umgebung bei einem Projekt erklärt. Für die Darstellung des Projekts mit einer VR-Brille wurde die Meta Quest 2 verwendet. Die Integration von VR in Unity, stellte sich einfach an, da in verschiedenen Foren von Meta Hilfestellungen zu finden sind.

Schlussendlich kann gesagt werden, dass die Verwendung von Unity für Anfänger, dank den ganzen Online-Foren, sich als einfach erwies. Dennoch stellt sich die Entwicklung, besonders im Bereich des maschinellen Lernens in Unity, als kompliziert an. Weshalb für die Umsetzung, öffentliche Projekte zur Hand genommen wurde. Die Verwendung von VR, bei der Entwicklung, erwies sich als nützlich für Entwickler, die Unity verwenden, um Projekte in die Realität umzusetzen, da mit VR wertvolle Ressourcen, Zeit und somit auch Geld gespart werden kann.

5.1 Related Work

Andere Ansätze, die für das ML in der Robotik verwendet werden können, sowie deren Algorithmen werden in [Gé19] genauer erklärt.

In der Arbeit [TFR+17] wird eine Alternative für die Umsetzung der im Kapitel 4 besprochenen Domänenrandomisierung, in die Realität umgesetzt. Weitere Modelle, die für das Greifen mit einem Roboter möglich sind, werden in [TBD+17] erläutert.

5.2 Ausblick

Aufgrund des weiten Themas von ML, gäbe es verschiedene Möglichkeiten, diese Arbeit fortzusetzen und zu verbessern. Es gibt verschiedene Algorithmen, die man ausprobieren könnte, um bessere Ergebnisse zu erzielen. Man könnte auch versuchen, andere Datensätze zu verwenden, um zu sehen, ob die Ergebnisse verbesserbar sind.

Das finale Ziel dieser Arbeit wäre, die Umsetzung und Implementierung des Projektes in einen Realen Roboter, der in der Industrie angewendet werden kann.

Literaturverzeichnis

[Cou22] Christopher Coutinho. *Unity® Virtual Reality Development with VRTK4 - A No-Coding Approach to Developing Immersive VR Experiences, Games, Apps.* Apress, New York, 2022. 19

[dROW17] Andrea de Giorgio, Mario Romero, Mauro Onori, and Lihui Wang. Human-machine collaboration in virtual reality for adaptive production engineering. *Procedia Manufacturing*, 11:1279–1287, 2017. 27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 27-30 June 2017, Modena, Italy. 18

[fD] Oculus for Developers. Oculus developer center. https://developer.oculus.com/downloads/package/unity-integration/. Zugriff am: 09.04.2022. 23

[Ger20] Charlie Gerard. *Practical Machine Learning in JavaScript - TensorFlow.js for Web Developers.* Apress, New York, 2020. 11, 12, 34

[Gé19] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow - Concepts, Tools, and Techniques to Build Intelligent Systems.* Ö'Reilly Media, Inc.", Sebastopol, 2019. 15, 16, 17, 30, 33

[Hal18] Jared Halpern. *Developing 2D Games with Unity - Independent Game Programming with C.* Apress, New York, 2018. 21, 22

[Jo21] Taeho Jo. *Machine Learning Foundations - Supervised, Unsupervised, and Advanced Learning.* Springer Nature, Singapore, 2021. 8, 9, 10, 11, 33

[ROS] ROS. Robot operating system. https://www.ros.org/. Zugriff am: 18.04.2022. 25

[Sah18] Sumit Saha. A comprehensive guide to convolutional neural networks-the eli5 way. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3 Dec 2018. Zugriff am: 22.04.2022. 27

[TBD+17] Joshua Tobin, Lukas Biewald, Rocky Duan, Marcin Andrychowicz, Ankur Handa, Vikash Kumar, Bob McGrew, Jonas Schneider, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. Domain randomization and generative models for robotic grasping, 2017. 27, 30

[Tec] Unity Technologies. Die weltweit führende plattform zur inhaltserstellung in echtzeit. https://unity.com/de. Zugriff am: 07.04.2022. 20

[TFR+17] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world, 2017. 25, 30

[Tim19] Kasimir Timmers. Einführung in die vr entwicklung mit unity. Seminararbeit, FH Aachen, 12 2019. 18, 20

[UR] Universal-Robots. Ur3e. https://www.universal-robots.com/de/produkte/ur3-roboter/. Zugriff am: 18.04.2022. 25

[UTa] Unity-Technologies. Teaching robots to see with unity. https://blog.unity.com/technology/teaching-robots-to-see-with-unity. Zugriff am: 23.04.2022. 24, 27, 33

[UTb] Unity-Technologies. Unity-technologies/com.unity.perception: Perception toolkit for sim2real training and validation in unity. https://github.com/Unity-Technologies/com.unity.perception. Zugriff am: 18.04.2022. 25

[UTc] Unity-Technologies. Unity-technologies/robotics-object-pose-estimation: A complete end-to-end demonstration in which we collect training data in unity and use that data to train a deep neural network to predict the pose of a cube. this model is then deployed in a simulated robotic pick-and-place task. https://github.com/Unity-Technologies/Robotics-Object-Pose-Estimation. Zugriff am: 16.04.2022. 24, 27, 28

[Ver20] Vaibhav Verdhan. *Supervised Learning with Python - Concepts and Practical Implementation Using Python*. Apress, New York, 2020. 3, 4, 5, 6, 12, 13, 14, 33, 34

Abbildungsverzeichnis

Tabellenverzeichnis