Heinrich Heine
Universität
Düsseldorf

INSTITUT FÜR INFORMATIK
Softwaretechnik und Programmiersprachen
Universitätsstr. 1 D–40225 Düsseldorf

XAI unter Adversarial Attack: Evaluation von Erklärungen für manipulierte Eingaben

Bachelorarbeit

im Studiengang Informatik
zur Erlangung des akademischen Grades

Bachelor of Science (B.Sc.)

vorgelegt von
Fabian Faraz Farid

Beginn der Arbeit: 23. Januar 2023
Abgabe der Arbeit: 23. April 2023

Erstgutachter: Prof. Dr. Michael Leuschel
Zweitgutachter: tbd

Selbstständigkeitserklärung

Hiermit versichere ich die vorliegende Bachelorarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Düsseldorf, den 23. April 2023

Fabian Faraz Farid

Zusammenfassung

Fassen Sie hier die Fragestellung, Motivation und Ergebnisse Ihrer Arbeit in wenigen Worten zusammen.

Die Zusammenfassung sollte den Umfang einer Seite nicht überschreiten.

Danksagung

Ich möchte meinem Betreuer, Herrn Dunkelau, von ganzem Herzen danken für die unermüdliche Unterstützung, Anleitung und Ermutigung während der Erstellung dieser Bachelorarbeit. Ohne seine Weisheit und Erfahrung hätte ich diese Arbeit nicht erfolgreich abschließen können.

Ich möchte meiner Familie danken, insbesondere meinen Eltern Khatera Farid und Mohammad Yousuf Farid, und meinen Geschwistern Florance Farda, Fayaz und Fezza, für ihre Liebe und Unterstützung während meines gesamten Studiums. Ihre unerschütterliche Unterstützung hat mich dazu inspiriert, hart zu arbeiten und meine Träume zu verfolgen.

Ich möchte mich von ganzem Herzen bei Kirran, Aleena und Iram für ihre Unterstützung bei meiner Bachelorarbeit bedanken. Ich bin unglaublich dankbar, dass ich sie als meine Freunde und Familie an meiner Seite hatte, während ich diese Herausforderung gemeistert habe.
Kirran, ich danke dir für deine unermüdliche Unterstützung und ermutigenden Worte. Du warst immer für mich da und hast mich motiviert, weiterzumachen, auch wenn ich manchmal an mir selbst gezweifelt habe.
Iram, du warst eine unersetzliche Stütze für mich während des gesamten Prozesses. Deine klugen Ratschläge und deine ruhige Art haben mir geholfen, meine Gedanken zu ordnen und meine Arbeit zu verbessern.
Aleena, du hast mir immer ein offenes Ohr geschenkt und mich dazu ermutigt, meine Gedanken und Ideen zu teilen. Deine positive Energie und dein Glaube an mich haben mir geholfen, meine Ziele zu erreichen.
Ohne euch wäre diese Bachelorarbeit nicht möglich gewesen. Ihr habt mir nicht nur geholfen, fachliche Fragen zu klären und meine Arbeit zu verbessern, sondern auch für mich da zu sein, wenn ich Unterstützung gebraucht habe. Ich bin dankbar für eure Freundschaft und für die Liebe, die ihr mir während dieser Zeit gegeben habt. Ich möchte euch auch für eure Geduld und Verständnis bedanken, wenn ich mal gestresst und unruhig war. Ihr habt mir geholfen, ruhig zu bleiben und mich auf das Wesentliche zu konzentrieren. Ich danke euch von ganzem Herzen und werde eure Freundschaft und eure Liebe immer schätzen.

Ich möchte mich bei meinem Freund Davin bedanken, der mich während der Erstellung meiner Bachelorarbeit unterstützt hat. Du hast mir nicht nur geholfen, meine Gedanken zu ordnen, sondern auch meine Schreibblockade zu überwinden. Obwohl ich oft dachte, dass ich meine Arbeit niemals beenden würde, hast du mich motiviert und mir gezeigt, dass es möglich ist. Ich werde deine Weisheiten immer in Erinnerung behalten: „Wenn du denkst, du schaffst es nicht, denk an den letzten Burger, den du gegessen hast, und du wirst sehen, dass du alles erreichen kannst.“

Ich danke Ihnen allen von ganzem Herzen für Ihre Unterstützung und Ermutigung und kann nicht genug betonen, wie sehr ich Ihre Hilfe schätze.

Inhaltsverzeichnis

1 Einleitung

In den letzten Jahren hat die Entwicklung von künstlicher Intelligenz (KI) und maschinellem Lernen (ML) rasant zugenommen. Dies hat zu einer Vielzahl von Anwendungen geführt, die in vielen Bereichen des täglichen Lebens eingesetzt werden, wie zum Beispiel in der Medizin, der Finanzbranche und der Bildverarbeitung. Eine der aufregendsten Entwicklungen in diesem Bereich ist die Erklärbarkeit von KI-Systemen, auch bekannt als XAI (Explainable Artificial Intelligence).

Eine wichtige Herausforderung bei der Entwicklung von XAI-Systemen ist jedoch die Verwundbarkeit gegenüber adversarial attacks. Adversarial attacks sind absichtliche Manipulationen der Eingabedaten, die dazu führen können, dass das KI-System falsche Entscheidungen trifft. In dieser Bachelorarbeit untersuchen wir, wie die Erklärbarkeit von XAI-Systemen beeinflusst wird, wenn sie mit manipulierten Eingaben konfrontiert werden. Wir werden verschiedene Methoden zur Evaluation von Erklärungen für manipulierte Eingaben untersuchen und diskutieren, wie sie zur Verbesserung der Sicherheit von XAI-Systemen beitragen können.

Diese Arbeit leistet einen wichtigen Beitrag zum Verständnis der Verwundbarkeit von XAI-Systemen gegenüber adversarial attacks und bietet Ansätze für die Entwicklung von Technologien, die die Erklärbarkeit und Sicherheit von KI-Systemen verbessern können.

Adversarial Examples sind spezialisierte Eingaben, die Neuronale Netze verwirren. Neuronale Netze sind sehr fragil und können leicht manipuliert werden. Als ein gängiges Beispiel wird das Bild eines Pandas verwendet. Dabei wird das Bild mit einer sehr kleinen Pertubation addiert, wo das Bild, was zuvor selbstsicher als Panda erkannt worden ist, nun als ein Langarmaffe erkannt wird.

Diese kleine Pertubation ist für das menschliche Auge nicht sichtbar. Viele Machine-Learning Modelle sind Black-Box-Algorithmen[GMR+18], was es für Menschen schwierig macht zu verstehen, wie eine Entscheidung getroffen wurde. Da insbesondere in medizinischen oder anderen sicherheitskritischen Bereichen immer mehr der Einsatz von künstlicher Intelligenz kommt hat man in der Forschung angefangen sich mit Explainable Artificial Intelligence, kurz XAI, zu beschäftigen. XAI sind Prozesse und Methoden, die Menschen helfen zu verstehen wie Prozesse und Resultate einer von einer künstlichen Intelligenz getroffenen Entscheidung abgelaufen sind. Es wird dafür verwendet, um ein AI-Modell dessen Einfluss und potenzielle Fehler zu erklären. Sie hilft aber auch bei der Problembehandlung und Verbesserung des Machine-Learning Modells und es ermöglicht uns, das Verhalten von Modellen zu untersuchen, indem wir Modellerkenntnisse zu Bereitstellungsstatus, Fairness, Qualität und der Abweichung verfolgen. Rudin zeigte, dass XAI-Algorithmen jedoch schlecht für Missklassifikationen nützlich sind und dabei eine grundlegende Annahme sei, dass die erste Klassifikation korrekt ist. Das Thema dieser Arbeit wird es sein zu erforschen, wie sich welche Verfahren bei Fehlklassifikationen über Adversarial Examples verhalten.

Mit XAI kann man herausfinden wie die KI Handlungen ausführt und in welchem Zusam-

menhang und welche Entscheidungen dazu führen dass sie ausgeführt werden. In dieser Arbeit wird herausgefunden, ob man damit herausfinden kann warum plötzlich das Bild eines Pandas bei einer kleinen Pertubation als ein Langarmaffe erkannt wird. In dieser Arbeit werden die Angriffe gezielt erklärt und Lösungsansätze erforscht[GMR+18].

In dieser Arbeit wird ein neuronales Netz auf Cifar trainiert und es werden Adversarial Examples generiert. Dabei wird die Bibliothek Foolbox verwendet. Am Ende vergleiche ich den XAI-Ansatz und somit das Originalbild mit dem pertubiertem Bild. Es wird zudem diskutiert, ob der Ansatz gut geeignet ist und welche der verschiedenen Ansätze gut geeignet ist, um Adversarial Examples zu erkennen.

2 Grundlagen

Künstliche Intelligenz (KI) ist ein Bereich der Informatik, der sich mit der Entwicklung von intelligenten Maschinen befasst. Die Idee ist, dass Maschinen in der Lage sein sollten, menschenähnliche Intelligenz und kognitive Fähigkeiten zu entwickeln, um komplexe Aufgaben auszuführen, die normalerweise menschliche Intelligenz erfordern. KI-Systeme sollen in der Lage sein, aus Erfahrungen zu lernen, Muster zu erkennen, Entscheidungen zu treffen und menschenähnliche Sprache und Kommunikation zu verstehen. Sie können auf verschiedene Weise implementiert werden, wie z.B. durch regelbasierte Systeme, neuronale Netze, Machine Learning (ML), Deep Learning (DL) oder Natural Language Processing (NLP). Diese verschiedenen Implementierungsmethoden sind notwendig, um unterschiedliche KI-Anwendungen zu realisieren, wie z.B. Robotik, Sprachverarbeitung, Bilderkennung oder autonome Systeme. Diese KI-Systeme basieren in der Regel auf der Verarbeitung von Daten. Diese werden von Sensoren oder anderen Quellen erfasst und dann von der KI-Software verarbeitet. Die Software kann dann Muster in den Daten erkennen und Entscheidungen treffen, die auf diesen Mustern basieren. Diese Entscheidungen können automatisch ausgeführt werden oder von einem menschlichen Operator überwacht werden.

Es gibt jedoch einige Herausforderungen bei der Entwicklung von KI-Systemen. Eine der größten Herausforderungen besteht darin, dass KI-Systeme oft Black-Box-Systeme sind, was bedeutet, dass es schwierig sein kann, zu verstehen, wie sie zu ihren Entscheidungen kommen. Dies kann ein Problem darstellen, wenn KI-Systeme Entscheidungen treffen, die das Leben von Menschen beeinflussen, wie z.B. autonome Fahrzeuge, die Unfälle vermeiden müssen. Ein weiteres Problem bei der Entwicklung von KI-Systemen ist die Datenaufbereitung. KI-Systeme benötigen große Mengen an Daten, um gut zu funktionieren. Diese Daten müssen auch qualitativ hochwertig und repräsentativ sein für die Situationen, auf die das KI-System angewendet wird. Das bedeutet, dass es oft eine Menge Arbeit erfordert, Daten zu sammeln, zu bereinigen und zu organisieren, bevor sie von einem KI-System verarbeitet werden können. Ein drittes Problem bei der Entwicklung von KI-Systemen ist die Übertragbarkeit. Ein KI-System, das gut funktioniert, wenn es auf eine bestimmte Situation angewendet wird, funktioniert möglicherweise nicht so gut, wenn es auf eine andere Situation angewendet wird. Dies liegt daran, dass KI-Systeme oft auf spezifischen

Daten trainiert werden und somit nur auf ähnliche Daten gut funktionieren.

Trotz dieser Herausforderungen hat die KI das Potenzial, viele Bereiche des Lebens zu verbessern. KI-Systeme können in der Medizin eingesetzt werden, um Krankheiten zu diagnostizieren und zu behandeln, oder in der Landwirtschaft, um Pflanzenkrankheiten zu erkennen und zu bekämpfen. Sie können auch in der Automobilindustrie eingesetzt werden, um autonome Fahrzeuge zu entwickeln, die sicherer und effizienter sind als menschliche Fahrer. Eine weitere wichtige Anwendung von KI ist in der Forschung und Entwicklung. KI-Systeme können genutzt werden, um komplexe Probleme in der Wissenschaft zu lösen, indem sie große Datenmengen analysieren und Muster erkennen, die für menschliche Forscher schwer zu erkennen sind. Dies kann zu Fortschritten in der Medizin, Materialwissenschaft und Werkstofftechnik, Klimaforschung und vielen anderen Bereichen führen. Ein weiteres Anwendungsgebiet für KI ist in der Wirtschaft. KI-Systeme können genutzt werden, um Unternehmen bei der Entscheidungsfindung zu unterstützen, indem sie große Datenmengen analysieren und Muster erkennen, die für menschliche Analysten schwer zu erkennen sind. Dies kann dazu beitragen, Geschäftsprozesse zu optimieren, Kosten zu senken und Umsätze zu steigern.

Allerdings gibt es auch Bedenken bezüglich der Auswirkungen von KI auf die Gesellschaft. Einige argumentieren, dass KI-Systeme Arbeitsplätze ersetzen und zu einer zunehmenden Arbeitslosigkeit führen könnten. Andere befürchten, dass KI-Systeme dazu genutzt werden könnten, Menschen zu überwachen oder zu kontrollieren, was zu einem Verlust an Privatsphäre und Freiheit führen könnte. Es gibt auch ethische Bedenken bezüglich der Verwendung von KI. Einige argumentieren, dass KI-Systeme diskriminierende oder ungerechte Entscheidungen treffen könnten, wenn sie auf Daten trainiert werden, die bestimmte Gruppen von Menschen benachteiligen. Es gibt auch Bedenken bezüglich der Verantwortlichkeit von KI-Systemen, insbesondere bei autonomen Systemen wie selbstfahrenden Autos. Um diese Bedenken anzugehen, müssen KI-Systeme transparenter und verantwortlicher gemacht werden. Dies erfordert die Entwicklung von Methoden zur Überwachung und Kontrolle von KI-Systemen, um sicherzustellen, dass sie fair, sicher und zuverlässig sind. Es erfordert auch eine engere Zusammenarbeit zwischen Regierungen, Unternehmen und der Gesellschaft, um sicherzustellen, dass KI im Einklang mit ethischen und gesellschaftlichen Standards entwickelt und eingesetzt wird.

2.1 Künstliche Intelligenz

Nach Russell[RN21] hat die KI acht Definitionen über zwei Dimensionen, welche in Abbildung 1 dargestellt sind. Die zwei Dimensionen nach Russell[RN21] sind menschenorientiert und rational. Beim menschenorientiertem Ansatz geht man vom Menschen und dessen Verhaltensmuster selbst aus. Wie ist der Mensch in der Lage, zu denken und zu entscheiden? Dabei setzt man auf viele Beobachtungen und Hypothesen bezüglich menschlichen Verhaltens an. Beim rationalem Ansatz benutzt man die Mathematik und die Technik in den Vordergrund und versucht mittels dessen die menschlichen Verhaltensmuster zu erklären.

Thinking Humanly	**Thinking Rationally**
"The exciting new effort to make computers think . . . *machines with minds*, in the full and literal sense." (Haugeland, 1985) "[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning . . ." (Bellman, 1978)	"The study of mental faculties through the use of computational models." (Charniak and McDermott, 1985) "The study of the computations that make it possible to perceive, reason, and act." (Winston, 1992)
Acting Humanly	**Acting Rationally**
"The art of creating machines that perform functions that require intelligence when performed by people." (Kurzweil, 1990) "The study of how to make computers do things at which, at the moment, people are better." (Rich and Knight, 1991)	"Computational Intelligence is the study of the design of intelligent agents." (Poole *et al.*, 1998) "AI . . . is concerned with intelligent behavior in artifacts." (Nilsson, 1998)

Figure 1.1 Some definitions of artificial intelligence, organized into four categories.

Abbildung 1: Russell's[RN21] Definitionen der künstlichen Intelligenz

Dabei behandeln die beiden oberen Definitionen die Aspekte des Gedankenprozesses und der Entscheidungsfindung eines Systems, während die beiden unteren Definitionen die Aspekte des Lernens und der Anwendung eines Systems behandeln[RN21].

Diese Definitionen und Ansätze haben in der Vergangenheit dazu geführt, dass die KI sich stets weiterentwickelt hat. Sie wurden von unterschiedliche Personen neu weiterentwickelt bis es zu der KI gekommen ist, die wir heute kennen. Sei es in Videospielen, in Unternehmen, in der Medizin oder der Agrarwirtschaft. Heutzutage kann die KI wie oben beschrieben in viele kleinere Teilbereiche eingesetzt werden und hat sich in vielen Bereichen etabliert. Man kann Stimmen verändern, Gesichter erkennen, Sprache erkennen und vieles mehr. Durch die massive Datenansammlung über die vergangenen Jahre und die Entwicklung von Rechenleistung, ist die KI immer mehr kreativer geworden und kann immer mehr Aufgaben übernehmen, wie zum Beispiel Gemälden zu malen, Texte verfassen und zu analysieren aber auch Code zu schreiben und zu kompilieren.

2.2 Machine Learning und Deep Learning

Wichtige Bausteine der KI sind Machine Learning (ML) und Deep Learning (DL). Deep Learning und Machine Learning sind zwei verwandte, aber unterschiedliche Ansätze zur

künstlichen Intelligenz. Wie in Abbildung 5 beschrieben nutzen beide Methoden Algorithmen, um Muster in Daten zu erkennen und Vorhersagen zu treffen, aber es gibt wesentliche Unterschiede in den Ansätzen und Technologien, die bei beiden Methoden verwendet werden.

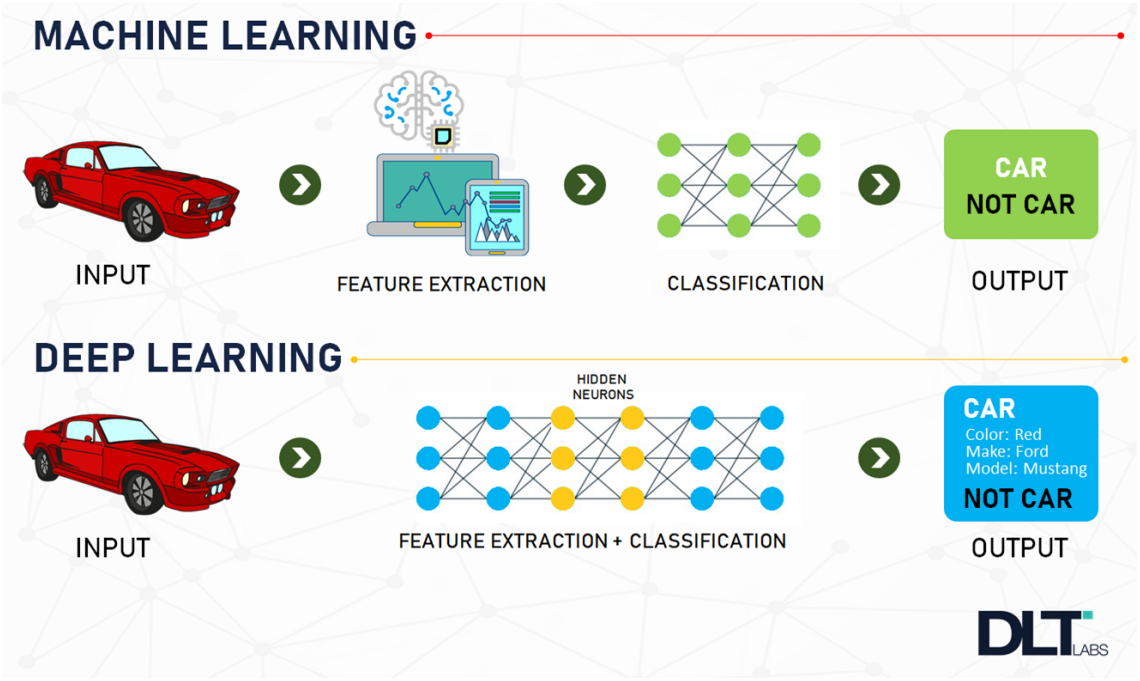

Abbildung 2: Der Unterschied zwischen Machine Learning und Deep Learning[Lab20]. Deep Learning erkennt Muster ohne menschliches Eingreifen

Machine Learning ist ein Ansatz, bei dem ein Algorithmus darauf trainiert wird, Muster in Daten zu erkennen und Vorhersagen auf der Grundlage dieser Muster zu treffen. Die meisten Machine Learning-Modelle verwenden eine begrenzte Anzahl von Schichten und sind in der Regel auf eine bestimmte Aufgabe spezialisiert. Diese Modelle können entweder überwacht oder unüberwacht trainiert werden. Beim überwachten Lernen werden dem Modell Trainingsdaten bereitgestellt, die beschriftet sind, damit das Modell die Muster in den Daten erkennen und lernen kann, welche Labels den Mustern zugeordnet werden sollten. Beim unüberwachten Lernen werden dem Modell unbeschriftete Daten bereitgestellt, und das Modell erkennt automatisch Muster in den Daten und gruppiert sie zusammen.

Deep Learning hingegen ist ein Ansatz, bei dem ein Algorithmus auf der Grundlage neuronaler Netze trainiert wird. Diese Netze bestehen aus vielen Schichten, die miteinander verbunden sind und die Informationen auf komplexe Weise verarbeiten können. Deep Learning-Modelle können sehr komplexe Muster in Daten erkennen und sind in der Lage, auf einer Vielzahl von Aufgaben eingesetzt zu werden, einschließlich Bild- und Spracherkennung, Sprachverarbeitung und autonomen Fahrzeugen. Ein weiterer Unterschied zwischen Machine Learning und Deep Learning ist die Art der verwendeten Daten. Während Machine Learning-Modelle in der Regel mit strukturierten Daten arbeiten, die in einer tabellarischen Form

vorliegen, können Deep Learning-Modelle auch mit unstrukturierten Daten arbeiten, wie z.B. Bildern oder Texten. Deep Learning-Modelle verwenden oft Convolutional Neural Networks (CNNs) und Recurrent Neural Networks (RNNs), die speziell für die Verarbeitung von Bild- und Sprachdaten entwickelt wurden. Die beiden Modelle sind unterschiedlich komplex. Machine Learning-Modelle sind in der Regel einfacher als Deep Learning-Modelle und können leichter verstanden und interpretiert werden. Deep Learning-Modelle führen daher zur höheren Komplexität der Trainings- und Validierungsprozesse.

2.2.1 Machine Learning

Machine Learning (ML) ist ein Teilgebiet der KI, das sich mit der Entwicklung von Algorithmen und Modellen befasst, die es Maschinen ermöglichen, aus Daten zu lernen und Vorhersagen zu treffen oder Entscheidungen zu treffen, ohne explizit programmiert zu werden. Im Gegensatz zu traditioneller Programmierung, bei der der Programmierer einen festgelegten Algorithmus schreibt, um bestimmte Aufgaben auszuführen, nutzt Machine Learning Algorithmen, die aus Daten lernen und selbstständig Muster und Zusammenhänge erkennen können. Molnar[Mol20] stellt diese einfach veranschaulichende Grafik in der Abbildung 3 zur Verfügung, die die Unterschiede zwischen der Programmierung und dem maschinellem Lernen einfach darstellt.

Machine Learning Algorithmen können in drei Kategorien unterteilt werden: Supervised Learning, Unsupervised Learning und Reinforcement Learning (RL)[RM19].

- Überwachtes Lernen, *Supervised Learning*, bezieht sich auf die Verwendung von gelabelten Daten, um ein Modell zu trainieren, das in der Lage ist, neue Daten korrekt zu klassifizieren oder vorherzusagen. Zum Beispiel kann ein Bilderkennungsmodell anhand von gelabelten Bildern trainiert werden, um Bilder von Hunden und Katzen zu unterscheiden. Das Modell lernt dabei, die Merkmale von Hunden und Katzen zu erkennen, und kann dann auf neue Bilder angewendet werden, um zu entscheiden, ob es sich um einen Hund oder eine Katze handelt.

- Unüberwachtes Lernen, *Unsupervised Learning*, bezieht sich auf die Verwendung von ungelabelten Daten, um Muster und Zusammenhänge zu erkennen. Ein Beispiel dafür ist die Clusteranalyse, bei der ähnliche Datenpunkte in Gruppen zusammengefasst werden. Das Ziel dabei ist es, die natürlichen Gruppen in den Daten zu erkennen und Zusammenhänge zu identifizieren, ohne dass man im Voraus weiß, welche Gruppen es gibt oder wie diese aussehen.

- Bestärkendes Lernen, *Reinforcement Learning (RL)*, bezieht sich auf die Verwendung von Feedback, um ein Modell zu trainieren, das auf bestimmte Aktionen und Situationen reagieren kann. Zum Beispiel kann ein Agent in einem Videospiel lernen, wie er Belohnungen erhält, indem er verschiedene Aktionen ausführt und dabei Feedback erhält, ob diese Aktionen nützlich oder schädlich sind.

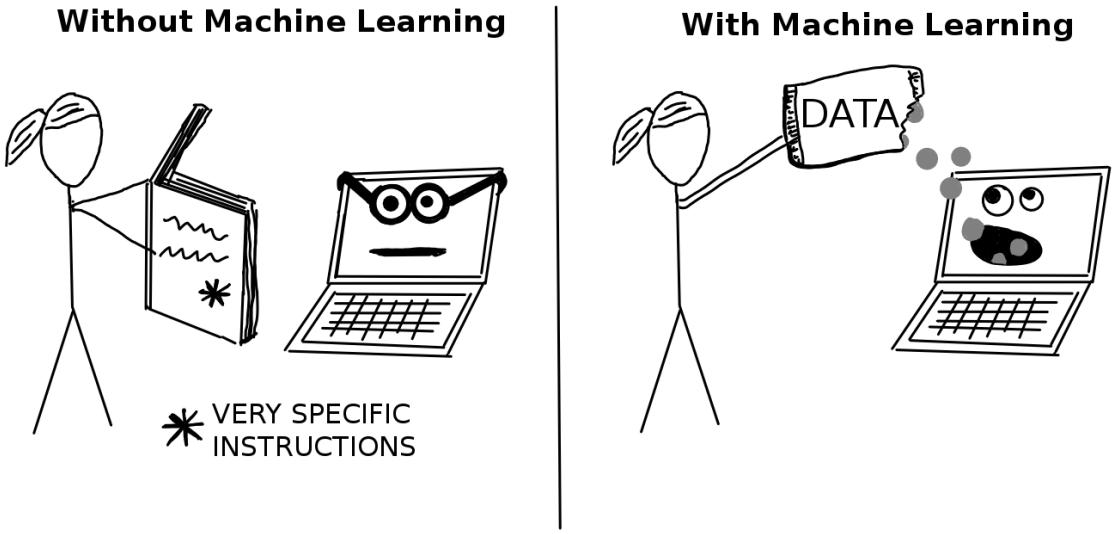

Abbildung 3: Molnar's[Mol20] einfache Darstellung wie ML funktioniert

Beim ML werden die Daten in Trainingsdaten und Testdaten unterteilt. Die Trainingsdaten werden benutzt, um das Modell zu trainieren und die Testdaten werden benutzt, um das Modell zu testen. Das Modell wird dann auf die Testdaten angewendet und die Ergebnisse werden mit den tatsächlichen Ergebnissen verglichen. Je besser das Modell die Testdaten vorhersagen kann, desto besser ist das Modell[Gér19]. Das Modell lernt, wie die Daten miteinander verbunden sind und wie es die Daten verarbeiten kann, um die gewünschten Ergebnisse zu erzielen.

Machine Learning hat eine Vielzahl von Anwendungen, einschließlich Bild- und Spracherkennung, Spam-Filterung, medizinischer Diagnose und Robotik. In der Bilderkennung können Machine Learning Algorithmen verwendet werden, um Gesichter zu erkennen oder Objekte auf einem Bild zu identifizieren. In der Spracherkennung können Machine Learning Algorithmen verwendet werden, um gesprochene Sprache in Text zu übersetzen oder die Absicht hinter einer bestimmten Aussage zu erkennen[SB18]. In der Medizin können Machine Learning Algorithmen verwendet werden, um Krankheiten zu diagnostizieren und Behandlungen vorherzusagen. Beispielsweise können Machine Learning Algorithmen verwendet werden, um Muster in medizinischen Daten zu erkennen und prädiktive Modelle zu erstellen, die Ärzten helfen, bessere Entscheidungen zu treffen. In der Robotik können Machine Learning Algorithmen verwendet werden, um autonome Systeme zu erstellen, die in der Lage sind, komplexe Aufgaben auszuführen, wie z.B. das Steuern von Drohnen oder das Betreiben von selbstfahrenden Autos.

Eine der größten Herausforderungen im Machine Learning ist die Überwachung der Modelle, um sicherzustellen, dass sie fair und ethisch korrekt sind. Wenn das Trainingsset beispielsweise systematische Verzerrungen aufweist oder ungleichmäßig auf verschiedene Gruppen

verteilt ist, kann dies zu unfairen Vorhersagen führen. Darüber hinaus können Machine Learning Algorithmen auch anfällig für Angriffe sein, wenn sie beispielsweise bewusst mit falschen Daten trainiert werden, um gezielte Fehlvorhersagen zu erzeugen. Ein weiteres Problem im Machine Learning ist die Erklärbarkeit von Modellen. Da Machine Learning Algorithmen auf der Verarbeitung großer Datenmengen basieren, kann es schwierig sein, die Entscheidungen, die sie treffen, zu erklären. Dies kann insbesondere in Bereichen wie der Medizin oder dem Finanzwesen problematisch sein, wo Entscheidungen aufgrund von Machine Learning Vorhersagen große Auswirkungen auf das Leben von Menschen haben können. Dennoch hat Machine Learning das Potenzial, unser Leben in vielen Bereichen zu verbessern. Durch die Automatisierung von Aufgaben und die Vorhersage von Ereignissen kann Machine Learning Zeit und Ressourcen sparen und neue Erkenntnisse gewinnen, die für menschliche Experten möglicherweise schwer zu erkennen sind.

Um Machine Learning erfolgreich einzusetzen, ist es jedoch wichtig, dass Unternehmen und Organisationen eine klare Vorstellung davon haben, welche Probleme sie lösen möchten, und dass sie die richtigen Datenquellen haben, um die Algorithmen zu trainieren. Darüber hinaus ist es wichtig, sicherzustellen, dass die Modelle fair, transparent und ethisch korrekt sind, um eine gerechte und verantwortungsvolle Anwendung von Machine Learning zu gewährleisten.

2.2.2 Deep Learning

Deep Learning (DL) ist ein Teilgebiet des Machine Learning, das sich auf die Verarbeitung und Analyse von komplexen Datenstrukturen wie Bildern, Texten und Audio spezialisiert hat. Im Gegensatz zu traditionellen Machine Learning Algorithmen, die auf manuell definierten Merkmalen und Regeln basieren, verwendet Deep Learning hierarchische Modelle, um automatisch relevante Merkmale aus den Daten zu extrahieren und zu lernen[Tra19]. Die grundlegende Architektur von Deep Learning besteht aus einem Netzwerk von künstlichen Neuronen, die miteinander verbunden sind und in Schichten organisiert sind[GBC16]. Jedes Neuron nimmt Eingabe in Form von numerischen Werten entgegen und führt eine lineare Transformation durch, gefolgt von einer nichtlinearen Aktivierungsfunktion. Die Ausgabe jedes Neurons wird dann an die nächsten Neuronen in der nächsten Schicht weitergeleitet, um eine Ende-zu-Ende Verarbeitung zu erreichen wie in Abbildung 4 beschrieben.

Die Netzwerkstruktur von Deep Learning Modellen ist flexibel und kann je nach Anwendungsfall variieren. Einige der gängigsten Architekturen sind Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs) und Autoencoders. CNNs sind besonders gut geeignet für Bildverarbeitungsaufgaben, da sie speziell für die Erkennung von Mustern in Bildern entworfen wurden. RNNs hingegen sind ideal für die Verarbeitung von Sequenzen wie Texten und Sprache, da sie in der Lage sind, den Kontext von vorherigen Eingaben zu berücksichtigen. Deep Learning Algorithmen können in verschiedenen Anwendungen eingesetzt werden, wie zum Beispiel in der Bild- und Spracherkennung, der automatischen Übersetzung, der medizinischen Diagnose und der Robotik[Bur19]. Einige der bemerkens-

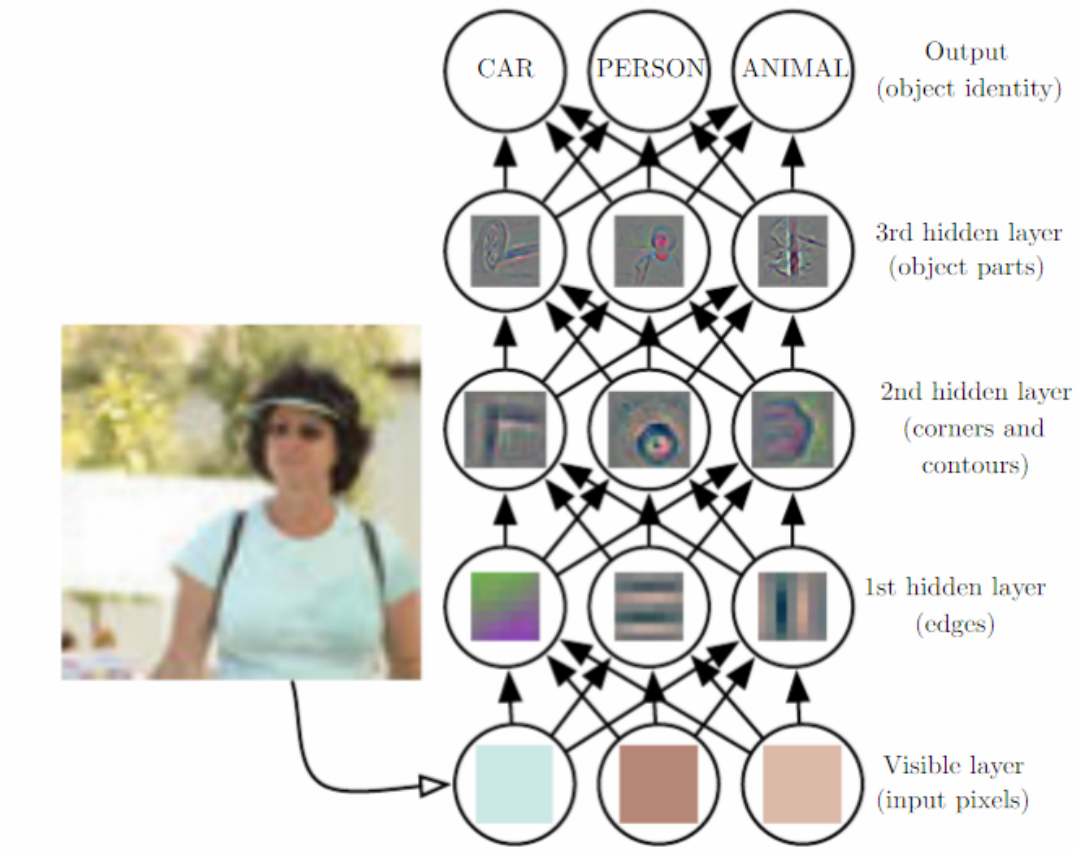

Abbildung 4: Illustration eines Deep Learning-Modells[GBC16]

wertesten Fortschritte in diesen Bereichen wurden durch Deep Learning ermöglicht, wie zum Beispiel die Entwicklung von selbstfahrenden Autos, die Erkennung von Krebszellen in medizinischen Bildern und die Übersetzung von Sprachen in Echtzeit.

Trotz der beeindruckenden Leistungen von Deep Learning gibt es auch einige Herausforderungen, die bewältigt werden müssen. Eine wichtige Frage ist die Erklärbarkeit von Deep Learning Modellen. Da die Netzwerke aus vielen Schichten und Neuronen bestehen, können die Entscheidungen, die sie treffen, schwer zu interpretieren sein. In einigen Fällen kann dies ein Hindernis für die Akzeptanz und den Einsatz von Deep Learning Modellen sein. Ein weiteres Problem ist der Bedarf an großen Datensätzen, um Deep Learning Algorithmen zu trainieren. Da diese Modelle auf der Verarbeitung großer Datenmengen basieren, können sie Schwierigkeiten haben, auf kleineren oder weniger repräsentativen Datensätzen zu generalisieren. Dies kann zu Overfitting führen, wo das Modell zu gut auf die Trainingsdaten passt, aber schlechte Vorhersagen auf neuen Daten macht[Mur12]. Schließlich gibt es auch ethische Fragen im Zusammenhang mit der Anwendung von Deep Learning. Zum Beispiel können

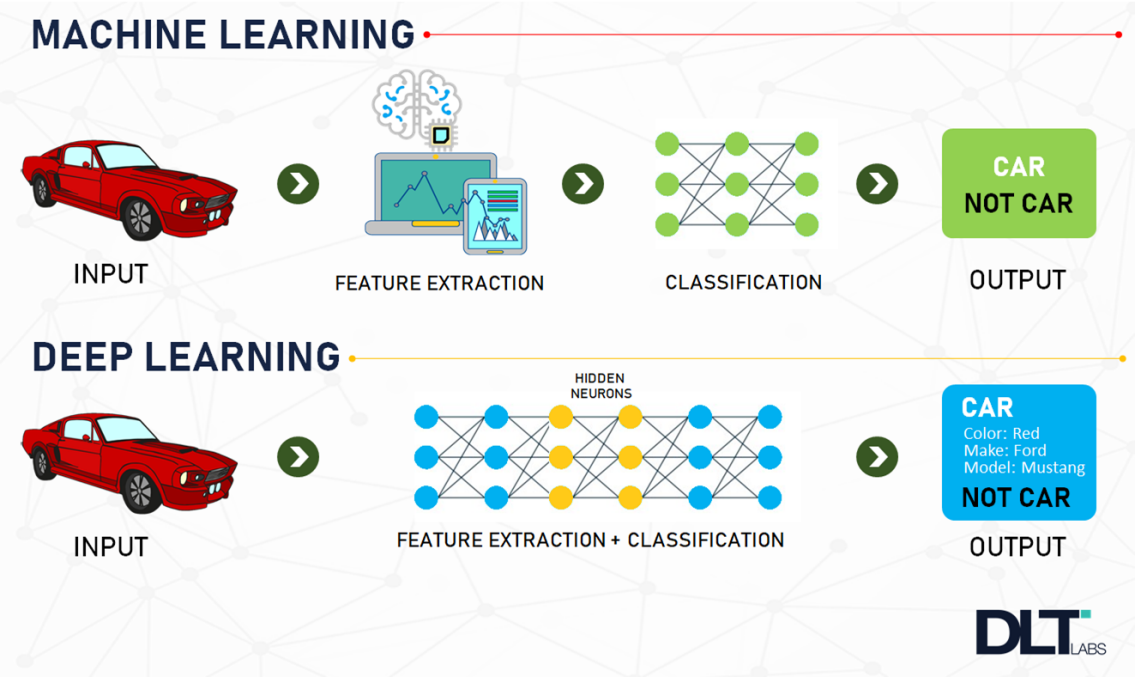

Abbildung 5: Der Unterschied zwischen Machine Learning und Deep Learning[Lab20]. Deep Learning erkennt Muster ohne menschliches Eingreifen

Algorithmen, die auf der Analyse von Daten basieren, Vorurteile und Diskriminierung reproduzieren, wenn die Daten selbst solche Verzerrungen enthalten. Es ist wichtig, dass Entwickler von Deep Learning Modellen und Anwendungen sich dieser Herausforderungen bewusst sind und Maßnahmen ergreifen, um sie zu minimieren.

Um Deep Learning Algorithmen zu trainieren, müssen große Datensätze zur Verfügung gestellt werden. Diese Datensätze müssen sorgfältig ausgewählt und vorbereitet werden, um sicherzustellen, dass sie eine angemessene Vielfalt von Beispielen enthalten und nicht durch Verzerrungen oder Vorurteile beeinflusst sind. Ein weiterer Ansatz zur Verbesserung der Erklärbarkeit von Deep Learning Modellen besteht darin, spezielle Tools und Techniken zu verwenden, um zu verstehen, wie das Modell Entscheidungen trifft und welche Merkmale es bei der Entscheidungsfindung berücksichtigt[Mur12]. Ein wichtiges Thema in der Forschung und Entwicklung von Deep Learning ist die Interpretation von Modellen und die Gewährleistung der Verantwortlichkeit von Entscheidungen. Eine Möglichkeit, dies zu erreichen, ist die Verwendung von Modellen, die das Entscheidungsfindungsprozess offenlegen können, indem sie beispielsweise eine Heatmap erstellen, die zeigt, welche Regionen eines Bildes zur Entscheidungsfindung beigetragen haben[Bis06]. Eine andere Möglichkeit besteht darin, ein erklärbares Modell zu erstellen, das einfachere Entscheidungsregeln verwendet, die leichter zu verstehen und zu interpretieren sind. Ein weiteres wichtiges Thema ist die Sicherheit von Deep Learning Modellen, insbesondere in sicherheitskritischen Anwendungen wie autonomem Fahren oder kritischen Infrastrukturen. Deep Learning Modelle können

anfällig für Angriffe sein, wie zum Beispiel durch das Einfügen von bösartigen Beispielen in den Trainingsdatensatz oder durch gezielte Angriffe auf das Modell selbst. Es ist wichtig, sicherheitsrelevante Bedrohungen frühzeitig zu erkennen und Maßnahmen zu ergreifen, um sie zu minimieren.

2.3 Adversarial Examples

Adversarial Examples sind ein Phänomen, bei dem geringfügige Änderungen an einem Eingabebild oder -datensatz dazu führen, dass ein künstliches neuronales Netzwerk (KNN) eine falsche Ausgabe liefert oder die Ausgabe stark beeinträchtigt wird. Diese Änderungen sind für das menschliche Auge oft nicht wahrnehmbar, können jedoch einen erheblichen Einfluss auf die Leistung des KNN haben[GPAM+14]. Adversarial Examples sind ein wachsendes Problem in der künstlichen Intelligenz, insbesondere in Anwendungen wie Bilderkennung, Gesichtserkennung und autonomen Fahrzeugen. Ein Angreifer kann gezielt Adversarial Examples erstellen und verwenden, um die Leistung von KNNs zu beeinträchtigen, was potenziell gefährliche Auswirkungen haben kann.

Es gibt verschiedene Methoden zur Erstellung von Adversarial Examples, die alle auf der Ausnutzung von Schwachstellen in der Architektur und den Algorithmen von KNNs basieren. Eine der bekanntesten Methoden zur Erstellung von Adversarial Examples ist die Fast Gradient Sign Method (FGSM)[GPAM+14]. Dabei wird das Eingabebild durch Hinzufügen von Rauschen in Richtung des Gradienten der Kostenfunktion verändert, um die Ausgabe des KNN zu beeinflussen. Eine andere Methode ist die Optimization-Based Attack, bei der das Eingabebild schrittweise verändert wird, um die Ausgabe des KNN zu maximieren oder zu minimieren[GPAM+14]. Das Erstellen von Adversarial Examples kann auf verschiedene Weise erfolgen, zum Beispiel durch das Hinzufügen von Rauschen, das Ändern von Pixeln oder das Hinzufügen von Texturen. Ein Angreifer kann auch gezielte Angriffe durchführen, indem er das Modell mit gezielten Eingaben anstatt mit zufälligen Eingaben angreift.

Adversarial Examples sind ein ernsthaftes Problem für die Zuverlässigkeit von KNNs, da sie die Leistung von KNNs beeinträchtigen können. Diese Beeinträchtigung kann in einigen Fällen kritisch sein, wie zum Beispiel bei autonomen Fahrzeugen, bei denen ein falsch erkanntes Objekt schwerwiegende Konsequenzen haben kann. Adversarial Examples können auch für gezielte Angriffe genutzt werden, um Systeme zu untergraben oder zu täuschen. Es gibt verschiedene Ansätze, um Adversarial Examples zu erkennen und zu verteidigen. Einer der Ansätze ist die Verwendung von Abwehrmechanismen, die darauf abzielen, die Robustheit des KNN gegenüber Adversarial Examples zu erhöhen. Einige der gängigen Abwehrmechanismen umfassen die Verwendung von verteidigenden Gradienten, die Verwendung von adversarialen Trainingsmethoden, die Verwendung von zufälligen Rauschen und die Verwendung von Ensembling-Techniken[GSS15]. Ein weiterer Ansatz zur Verteidigung gegen Adversarial Examples ist die Verwendung von Modellen, die gegen Adversarial Examples robust sind. Diese Modelle werden speziell entwickelt, um Adversarial Examples zu widerstehen, indem sie Robustheit als Designziel verwenden. Beispiele für

solche Modelle sind RobustFill und Defensive Distillation. Ein weiterer Ansatz ist die Verwendung von White-Box-Attacken, bei denen der Angreifer Kenntnisse über das interne Modell des KNN hat und gezielt Angriffe ausführt, um das Modell zu untergraben. Dies kann durch Reverse Engineering des Modells, durch Ausnutzung von Sicherheitslücken oder durch Zugang zu den Trainingsdaten erfolgen.

Es ist jedoch wichtig zu beachten, dass die Verteidigung gegen Adversarial Examples ein fortlaufender Forschungsbereich ist und es keine perfekte Lösung gibt. Die meisten Verteidigungsmechanismen haben ihre Grenzen und können möglicherweise nicht gegen alle Arten von Adversarial Examples bestehen. Adversarial Examples sind ein wichtiges Thema in der künstlichen Intelligenz, da sie die Zuverlässigkeit von KNNs beeinträchtigen und potenziell gefährliche Auswirkungen haben können. Es ist wichtig, weiterhin Forschung auf diesem Gebiet zu betreiben, um robustere KNNs zu entwickeln und effektivere Verteidigungsmechanismen zu finden, um die Sicherheit und Verlässlichkeit von KNNs in wichtigen Anwendungen zu gewährleisten.

2.4 XAI

Explainable AI (XAI) ist ein relativ neues Forschungsfeld, das sich mit der Entwicklung von Methoden zur Interpretation und Erklärung der Entscheidungen von künstlichen Intelligenzsystemen beschäftigt. XAI zielt darauf ab, komplexe Algorithmen, insbesondere Machine-Learning-Modelle, zugänglicher und transparenter zu machen, indem es den Anwendern und Entwicklern ermöglicht, die Logik und das Verhalten dieser Systeme zu verstehen und zu interpretieren. Traditionell sind KI-Modelle, insbesondere Deep-Learning-Modelle, schwierig zu verstehen, da sie aus Hunderten oder Tausenden von Schichten und Neuronen bestehen können. Diese Modelle sind häufig Black-Box-Systeme, bei denen die Entscheidungen nicht nachvollziehbar sind, da sie auf einer Vielzahl von Datenpunkten basieren, die für das menschliche Auge nicht erkennbar sind. Dies kann zu mangelndem Vertrauen in die Entscheidungen des Systems führen und stellt ein Hindernis für die breite Akzeptanz und Einführung von KI in verschiedenen Branchen dar[Rot20].

XAI-Methoden zielen darauf ab, dieses Problem zu lösen, indem sie Transparenz und Erklärbarkeit in die Entscheidungsfindung von KI-Systemen bringen. Es gibt verschiedene Ansätze für XAI, darunter das Hervorheben von wichtigen Merkmalen und Mustern, die von KI-Modellen verwendet werden, das Erstellen von Entscheidungsbäumen und Regeln, die die Entscheidungsprozesse der Modelle widerspiegeln, sowie die Verwendung von Visualisierungen und Interaktionsmethoden, um dem Benutzer die Entscheidungen zu erklären[Agg18]. Eine der Hauptmethoden des XAI ist die Attributionsanalyse, bei der jeder Eingabevariable ein Attributionswert zugeordnet wird, der angibt, wie wichtig sie für die Entscheidung des Modells war. Die Attributionsanalyse kann für verschiedene Zwecke verwendet werden, wie z.B. für die Erkennung von Fehlern in KI-Modellen, für die Erklärung von Entscheidungen gegenüber Stakeholdern und für die Erkennung von Biases in Modellen. Ein weiterer wichtiger Ansatz für XAI ist die Modellvereinfachung, bei der

komplexe Modelle in einfachere Modelle umgewandelt werden, die leichter zu interpretieren und zu verstehen sind. Die Modellvereinfachung kann durch Techniken wie das Entfernen von unnötigen Funktionen, die Aggregation von Schichten oder das Ersetzen von tiefen neuronalen Netzen durch flache Modelle erfolgen.

XAI hat Anwendungen in verschiedenen Bereichen, darunter in der Medizin, wo es wichtig ist, dass Ärzte die Entscheidungen von KI-Modellen nachvollziehen und überprüfen können, in der Finanzindustrie, wo es notwendig ist, Entscheidungen von KI-Modellen gegenüber Kunden zu erklären, sowie in der Robotik, wo es wichtig ist, dass der Benutzer versteht, wie der Roboter seine Entscheidungen trifft.

Es gibt jedoch auch Herausforderungen im Bereich XAI. Eine der größten Herausforderungen besteht darin, dass komplexe KI-Modelle schwierig zu interpretieren sind, insbesondere, wie sie aufgrund der Art und Weise, wie sie trainiert werden, inhärent unerklärlich sein können. Ein weiteres Problem ist, dass einige XAI-Methoden nicht immer zuverlässig sind und irreführende oder falsche Erklärungen liefern können. Es ist auch schwierig, XAI-Methoden zu validieren und zu testen, da es schwierig ist, die tatsächliche Güte der Erklärungen zu messen. Es gibt jedoch vielversprechende Entwicklungen im Bereich XAI. Ein Ansatz ist die Verwendung von Interpretabilitätszertifikaten, die eine Garantie dafür bieten, dass das Modell interpretierbar ist und dass die Erklärungen korrekt und konsistent sind. Ein weiterer Ansatz ist die Entwicklung von Hybridmodellen, die aus einer Kombination von Black-Box-Modellen und Interpretable-Modellen bestehen und die Vorteile beider Ansätze kombinieren.

In Zukunft wird es wichtig sein, dass Unternehmen und Entwickler von KI-Systemen XAI-Methoden bei der Entwicklung von Modellen und Anwendungen berücksichtigen und implementieren, um sicherzustellen, dass ihre Systeme transparent und verständlich sind. Es wird auch wichtig sein, dass Regierungen und Regulierungsbehörden Richtlinien und Standards für die Verwendung von XAI in verschiedenen Branchen entwickeln, um sicherzustellen, dass KI-Systeme fair, transparent und ethisch sind.

2.4.1 LIME

LIME (Local Interpretable Model-Agnostic Explanations) ist eine Methode der Explainable AI (XAI), die entwickelt wurde, um die Transparenz von Machine-Learning-Modellen zu erhöhen. Die Methode ist besonders nützlich, wenn es schwierig ist, die Entscheidungsprozesse von Black-Box-Modellen zu verstehen, da sie auf lokaler Ebene Erklärungen für Entscheidungen liefert. In diesem Text werden wir LIME in Bachelorarbeit Niveau wissenschaftlich erklären. Machine-Learning-Modelle sind oft als Black-Box-Systeme konzipiert, d.h. sie sind so konzipiert, dass sie ein bestimmtes Ergebnis erzielen, ohne dass der Benutzer die zugrundeliegenden Entscheidungsprozesse versteht. Dies kann ein Problem darstellen, da es schwierig ist, das Vertrauen in die Modelle aufrechtzuerhalten, wenn Benutzer nicht in der Lage sind, die Entscheidungsprozesse nachzuvollziehen. Es gibt jedoch einen wachsenden Bedarf an Transparenz und Interpretierbarkeit von Machine-Learning-Modellen,

insbesondere in kritischen Anwendungen wie der Medizin und der Finanzindustrie.

LIME ist eine Methode, die dazu beitragen kann, die Transparenz von Machine-Learning-Modellen zu erhöhen, indem sie lokale Erklärungen für Entscheidungen liefert. Die Methode funktioniert, indem sie eine Approximation des Black-Box-Modells erstellt, die lokal interpretierbar ist. Dies wird erreicht, indem die relevanten Merkmale der Eingabevariablen identifiziert werden, die zur Entscheidung des Modells beitragen, und indem ein lokales Modell erstellt wird, das auf diesen Merkmalen basiert. Die Methode kann auf verschiedene Arten angewendet werden, je nach den Bedürfnissen des Benutzers. In der Regel wird sie verwendet, um zu erklären, warum eine bestimmte Entscheidung von einem Modell getroffen wurde, indem sie die relevanten Merkmale der Eingabevariablen hervorhebt und die Beiträge jedes Merkmals zur Entscheidung quantifiziert[RHS20].

Ein Beispiel dafür wäre die Verwendung von LIME, um die Entscheidungen eines Bilderkennungsmodells zu erklären. Angenommen, ein Modell wurde trainiert, um Hunde und Katzen auf Bildern zu erkennen, aber der Benutzer möchte verstehen, welche Merkmale das Modell verwendet, um diese Entscheidungen zu treffen. LIME könnte verwendet werden, um ein lokales Modell zu erstellen, das auf einem bestimmten Bild basiert, und um die Merkmale hervorzuheben, die das Modell zur Entscheidungsfindung verwendet hat.

Eine der wichtigsten Eigenschaften von LIME ist, dass es „modellagnostisch“ ist, d.h. es kann auf jedes Machine-Learning-Modell angewendet werden, unabhängig von der Architektur oder den Trainingsdaten. Dies bedeutet, dass es ein universelles Werkzeug ist, das auf einer Vielzahl von Modellen und Anwendungen angewendet werden kann. Ein weiterer Vorteil von LIME ist, dass es eine flexible Methode ist, die an die Bedürfnisse des Benutzers angepasst werden kann. Es gibt verschiedene Varianten von LIME, die für verschiedene Anwendungen und Arten von Modellen optimiert sind. Zum Beispiel kann LIME für Textdaten verwendet werden, um zu erklären, wie ein Modell Entscheidungen trifft, oder für Tabellendaten, um die Bedeutung von Merkmalen zu erklären, die zur Entscheidungsfindung beitragen.

LIME basiert auf dem Konzept der lokalen Linearisierung, d.h. es versucht, die Entscheidungsfunktion des Modells in einem kleinen Bereich um eine bestimmte Eingabevariableninstanz zu approximieren. Um dies zu erreichen, führt LIME zwei Schritte aus:

1. Die Identifikation von Merkmalen, die zur Entscheidung des Modells beitragen. Dies wird erreicht, indem eine große Anzahl von Stichproben aus der Verteilung der Eingabevariablen generiert und an das Modell weitergegeben wird. Die Merkmale, die sich am stärksten auf die Entscheidungen des Modells auswirken, werden als wichtigste Merkmale identifiziert.

2. Die Erstellung eines lokalen Modells, das auf diesen Merkmalen basiert. Dies wird erreicht, indem die Eingabevariableninstanz und ihre wichtigsten Merkmale zufällig verändert werden, um eine Reihe von Stichproben zu erstellen. Für jede Stichprobe wird das Modell erneut ausgeführt, und die Ausgaben werden verwendet, um ein lineares Modell zu erstellen, das das Verhalten des Modells im lokalen Bereich approximiert.

Dieses Modell kann dann verwendet werden, um die Beiträge jedes Merkmals zur Entscheidung des Modells zu quantifizieren.

Die Erklärungen, die von LIME bereitgestellt werden, können in verschiedenen Formen präsentiert werden, einschließlich visueller Darstellungen wie Hitze- oder Streudiagrammen. Diese Darstellungen können dazu beitragen, das Verständnis der Entscheidungsprozesse von Machine-Learning-Modellen zu verbessern und die Vertrauenswürdigkeit der Modelle zu erhöhen.

Obwohl LIME eine nützliche Methode zur Erklärung von Entscheidungen von Machine-Learning-Modellen ist, hat sie auch ihre Einschränkungen[RHS20]. Zum Beispiel kann LIME nur lokale Erklärungen liefern, dh Erklärungen für einzelne Eingabevariableninstanzen. Dies bedeutet, dass sie möglicherweise nicht in der Lage ist, umfassendere Muster in den Entscheidungsprozessen von Modellen aufzudecken. Darüber hinaus kann die Qualität der Erklärungen von der Qualität des lokalen Modells abhängen, das erstellt wurde, um das Verhalten des Modells im lokalen Bereich zu approximieren[Rud19]. Darüber hinaus gibt es weitere Entwicklungen in der XAI, die auf LIME aufbauen oder alternative Ansätze verfolgen[RHS20]. Ein Beispiel dafür ist SHAP (Shapley Additive Explanations), eine Methode, die auf dem Konzept der Shapley-Values aus der Spieltheorie basiert. SHAP ermöglicht es, die Beiträge einzelner Merkmale zur Entscheidung des Modells in einem globalen Kontext zu quantifizieren, anstatt nur lokale Erklärungen zu liefern.

2.4.2 SHAP

SHAP (SHapley Additive exPlanations) ist ein Framework für Explainable Artificial Intelligence (XAI), das es ermöglicht, komplexe Machine-Learning-Modelle zu verstehen und zu interpretieren. Es handelt sich um eine Methode zur Berechnung von Shapley Values, die eine wichtige Rolle in der Spieltheorie und insbesondere in der Koalitionsbildung spielen. SHAP wurde entwickelt, um diese Ideen auf Machine-Learning-Modelle anzuwenden und somit zu erklären, wie das Modell Entscheidungen trifft und welche Merkmale am meisten zu diesen Entscheidungen beitragen. Eine der wichtigsten Eigenschaften von SHAP ist, dass es eine modellagnostische Methode ist. Das bedeutet, dass SHAP unabhängig von dem verwendeten Machine-Learning-Modell angewendet werden kann. Es kann also sowohl auf Entscheidungsbäume als auch auf neuronale Netze angewendet werden. Das macht SHAP sehr flexibel und anpassungsfähig an verschiedene Modelle und Anwendungsbereiche.

Um das Verständnis von SHAP zu erleichtern, ist es hilfreich, sich das Konzept der Shapley Values näher anzusehen. Die Shapley Values sind eine Methode, um den Wertbeitrag jedes Spielers in einer Koalition zu berechnen. In der Spieltheorie werden die Shapley Values verwendet, um die Verteilung von Gewinnen in einer Gruppe von Spielern zu bestimmen, die zusammenarbeiten, um ein Ziel zu erreichen. Die Shapley Values basieren auf der Idee, dass jeder Spieler einen Beitrag zum Gesamtergebnis leistet, aber nicht jeder Beitrag gleich wichtig ist. Die Shapley Values berechnen den Wertbeitrag jedes Spielers auf der Grundlage

der Beiträge, die er zu verschiedenen Koalitionen geleistet hat, und berücksichtigen dabei, dass die Beiträge der Spieler gegenseitig abhängig sind. In Bezug auf Machine Learning können die Shapley Values verwendet werden, um zu berechnen, wie viel jeder Merkmalswert zum Modellergebnis beiträgt. Dies ist besonders nützlich, um zu verstehen, welche Merkmale am meisten zum Ergebnis beitragen und welche weniger wichtig sind. SHAP wendet die Ideen der Shapley Values auf Machine-Learning-Modelle an, indem es berechnet, wie viel jeder Merkmalswert zum Modellergebnis beiträgt.

Um SHAP auf ein Machine-Learning-Modell anzuwenden, müssen zunächst die Shapley Values berechnet werden. Dazu wird das Modell mit allen möglichen Merkmalskombinationen ausgeführt, um die Beiträge jedes Merkmals zum Modellergebnis zu berechnen. Da es in der Regel nicht praktikabel ist, alle möglichen Merkmalskombinationen auszuführen, verwendet SHAP eine Monte-Carlo-Simulation, um die Shapley Values zu approximieren. Das Ergebnis ist eine Liste von Shapley Values, die angibt, wie viel jeder Merkmalswert zum Modellergebnis beigetragen hat. Die Shapley Values können auf verschiedene Arten interpretiert werden. Eine Möglichkeit besteht darin, die Shapley Values als Gewichte zu verwenden, um das Modellergebnis zu erklären. Auf diese Weise können die Shapley Values verwendet werden, um zum Beispiel zu erklären, warum ein bestimmtes Modell eine bestimmte Entscheidung getroffen hat oder welche Merkmale den größten Einfluss auf das Modellergebnis hatten. Ein weiterer Vorteil von SHAP besteht darin, dass es sowohl für Black-Box als auch für White-Box Modelle verwendet werden kann. Das bedeutet, dass es sowohl auf Modelle angewendet werden kann, bei denen der interne Entscheidungsprozess unbekannt ist, als auch auf Modelle, bei denen der interne Entscheidungsprozess bekannt ist.

2.4.3 Saliency Maps

Saliency Maps sind ein Werkzeug der sogenannten Explainable Artificial Intelligence (XAI), das dazu verwendet wird, Entscheidungen von künstlichen neuronalen Netzwerken zu visualisieren und zu interpretieren. Dabei wird versucht, die Aufmerksamkeit des Netzes auf bestimmte Teile des Eingabebildes zu lenken und somit zu zeigen, welche Bereiche des Bildes zur Entscheidungsfindung beigetragen haben. Saliency Maps basieren auf der Idee, dass das neuronale Netzwerk während des Trainings bestimmte Eigenschaften des Inputs identifiziert und in den Gewichten der einzelnen Neuronen kodiert. Diese Gewichte werden auch als Aktivierungen bezeichnet und geben an, wie stark das Netzwerk auf eine bestimmte Eingabe reagiert. Saliency Maps nutzen diese Aktivierungen, um die „wichtigsten“ Regionen eines Eingabebildes zu identifizieren.

Es gibt verschiedene Ansätze zur Berechnung von Saliency Maps, wie beispielsweise den Gradientenbasierten Ansatz, den Perturbationsbasierten Ansatz und den Region-Adaptive Ansatz. Jeder dieser Ansätze hat seine eigenen Vor- und Nachteile und kann für bestimmte Anwendungen besser geeignet sein. Saliency Maps können auch in Kombination mit anderen XAI-Methoden wie SHAP verwendet werden, um die Interpretierbarkeit von Modellen weiter

Quellcode 1: Schritt 1

```
1:  import numpy as np
2:  import pickle
3:
4:  def load_cifar_batch(filename):
5:      with open(filename, 'rb') as f:
6:          datadict = pickle.load(f, encoding='latin1')
7:          X = datadict['data']
8:          Y = datadict['labels']
9:          X = X.reshape(10000, 3, 32, 32).transpose(0,2,3,1).astype("float")
10:         Y = np.array(Y)
11:         return X, Y
12:
13: def load_cifar10():
14:     xs = []
15:     ys = []
16:     for b in range(1,6):
17:         f = 'cifar-10-batches-py/data_batch_%d' % (b,)
18:         X, Y = load_cifar_batch(f)
19:         xs.append(X)
20:         ys.append(Y)
21:     Xtrain = np.concatenate(xs)
22:     Ytrain = np.concatenate(ys)
23:     del X, Y
24:     Xtest, Ytest = load_cifar_batch('cifar-10-batches-py/test_batch')
25:     return Xtrain, Ytrain, Xtest, Ytest
```

zu verbessern. Indem die wichtigen Regionen eines Eingabebildes mit den SHAP-Werten für die Merkmale des Modells korreliert werden, können Experten und Nutzer des Modells ein besseres Verständnis dafür entwickeln, wie Entscheidungen getroffen werden.

3 Experimenteller Aufbau

Neuronale Netze sind ein wichtiger Bestandteil der künstlichen Intelligenz und ermöglichen es, komplexe Aufgaben wie Bilderkennung, Sprachverarbeitung und Mustererkennung zu lösen. Im Folgenden werden wir uns mit der Trainingsphase eines neuronalen Netzes in Python auf dem CIFAR-10-Datensatz befassen und die Schritte detailliert erläutern[GKP19].

Zunächst müssen wir den CIFAR-10-Datensatz laden und vorbereiten. Der CIFAR-10-Datensatz besteht aus 60.000 32x32 Pixel Farbbildern in 10 Klassen, wobei jede Klasse 6.000 Bilder enthält. Der Datensatz ist in fünf Batches aufgeteilt, wobei jeder Batch ein separates File ist. Wir laden die Daten mit dem folgenden Quellcode 1.

Wir teilen die Daten in Trainings- und Testdaten auf, wobei wir 50.000 Bilder für das Training

Quellcode 2: Schritt 2

```
X_train, Y_train, X_test, Y_test = load_cifar10()

# Normalization
mean_image = np.mean(X_train, axis=0)
std_image = np.std(X_train, axis=0)

X_train = (X_train - mean_image) / std_image
X_test = (X_test - mean_image) / std_image
```

Quellcode 3: Schritt 3

```
import tensorflow as tf

# Define the model
model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu',
        input_shape=(32, 32, 3)),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Conv2D(64, (3,
```

und 10.000 Bilder für das Testen verwenden. Die Daten müssen normalisiert werden, um sicherzustellen, dass sie einen ähnlichen Skalenbereich haben. Hierfür subtrahieren wir den Durchschnitt der Pixelwerte und dividieren durch die Standardabweichung in Quellcode 2.

Als nächstes definieren wir das neuronale Netzwerk. Wir verwenden ein Convolutional Neural Network (CNN), das aus mehreren Convolutional Layers und Pooling Layers besteht, gefolgt von einigen Fully Connected Layers. Das CNN wird wie folgt in Quellcode 3 definiert.

Das CNN besteht aus drei Convolutional Layers mit jeweils 32, 64 und 64 Filtern der Größe 3x3 und der ReLU-Aktivierungsfunktion. Die Convolutional Layers werden von Pooling Layers gefolgt, die die räumliche Dimension des Feature-Maps halbieren. Nach den Convolutional Layers wird die Ausgabe des CNNs in einen Vektor umgeformt und durch zwei Fully Connected Layers weiterverarbeitet, bevor sie auf die Ausgabe der 10 Klassen abgebildet wird.

Als nächstes müssen wir das Modell kompilieren und trainieren. Wir verwenden die Categorical Crossentropy Loss-Funktion und den Adam-Optimizer. Wir definieren auch die Metrik Accuracy, um die Leistung des Modells während des Trainings zu überwachen. Wir trainieren das Modell über 50 Epochen mit einem Batch-Size von 64 Bildern in Quellcode 4.

Während des Trainings können wir den Fortschritt des Modells überwachen, indem wir die Loss-Funktion und die Accuracy des Trainings- und Testsets in Quellcode 5 plotten.

Nach dem Training können wir das Modell testen, indem wir die Accuracy auf dem Testset

Quellcode 4: Schritt 4

```
  # Compile the model
model.compile(optimizer='adam',
              loss=tf.keras.losses.
                SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

# Train the model
history = model.fit(X_train, Y_train, epochs=50, batch_size=64,
                    validation_data=(X_test, Y_test))
```

Quellcode 5: Schritt 5

```
  import matplotlib.pyplot as plt

# Plot the loss and accuracy
plt.plot(history.history['loss'], label='train_loss')
plt.plot(history.history['val_loss'], label='val_loss')
plt.plot(history.history['accuracy'], label='train_acc')
plt.plot(history.history['val_accuracy'], label='val_acc')
plt.legend()
plt.show()
```

in Quellcode 6 auswerten.

Beim Durchlaufen der obigen Schritte erhalten wir eine Test Accuracy von 0.699 bei 50 Epochen. Die Ergebnisse des Trainings werden in Abbildung 6 dargestellt.

4 Erklärungen für Adversarial Examples

Adversarial Examples sind Bilder, die absichtlich manipuliert wurden, um ein neuronales Netzwerk dazu zu bringen, eine falsche Vorhersage zu treffen. Im Folgenden beschreibe ich, wie solche Adversarial Examples generiert werden können, indem ich den Fast Gradient Sign Method (FGSM) Algorithmus implementiere. Erst lade ich die Daten mit dem folgenden Quellcode 7.

Quellcode 6: Schritt 6

```
  # Evaluate the model
  test_loss, test_acc = model.evaluate(X_test, Y_test, verbose=2)
print('Test accuracy:', test_acc)
```

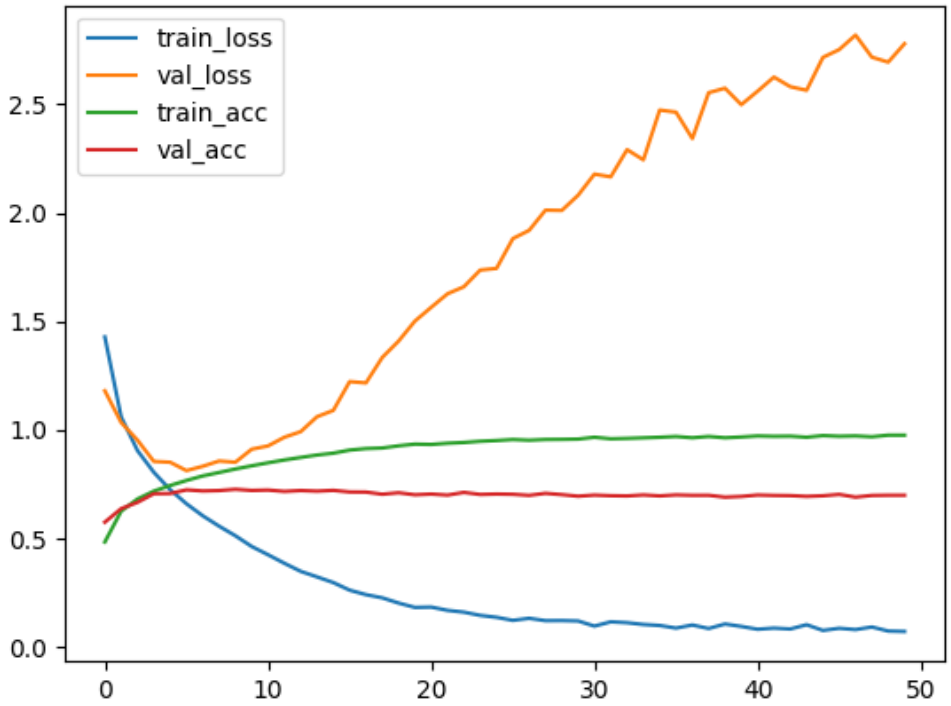

Abbildung 6: Ergebnisse des Trainings eines Neuronalen Netzwerks auf CIFAR-10

Danach erstelle ich die Funktion zum Generieren von Adversarial Examples mit dem folgenden Quellcode 8. Diese Funktion berechnet den Gradienten des Verlusts der Vorhersage des Modells mit Bezug auf das Eingabebild und gibt das Vorzeichen des Gradienten zurück, um das Vorzeichen der Änderungen zu erhalten, die das Modell dazu bringen, eine falsche Vorhersage zu treffen.

In Quellcode 9 wird ein Adversarial Example generiert, indem eine Perturbation zum ausgewählten Bild hinzugefügt wird. Das Ziel-Label ist in diesem Fall das Label der Klasse 3. Das Epsilon gibt die Größe der Störung an, die auf das Bild angewendet wird. Die Vorhersage des Modells auf dem manipulierten Bild wird ausgegeben.

Im nächsten Schritt möchte ich die generierten Adversarial Examples visualisieren, In diesem Quellcode 10 werden die generierten Adversarial Examples visualisiert, um zu sehen, wie sie sich vom Originalbild unterscheiden.

Damit wurde ein Adversarial Example generiert.

Quellcode 7: Schritt 1

```
import tensorflow as tf
import numpy as np

# Laden der Daten (hier verwende ich Cifar10)
(x_train, y_train), (x_test, y_test) =
    tf.keras.datasets.cifar10.load_data()

# Vorbereiten der Daten
x_test = x_test.astype('float32') / 255
```

Quellcode 8: Schritt 2

```
def generate_adversarial_pattern(input_image, input_label):
    loss_object = tf.keras.losses.CategoricalCrossentropy()

    with tf.GradientTape() as tape:
        tape.watch(input_image)
        prediction = model(input_image)
        loss = loss_object(input_label, prediction)

    gradient = tape.gradient(loss, input_image)
    signed_grad = tf.sign(gradient)
    return signed_grad
```

Quellcode 9: Schritt 3

```

epsilon = 0.01

image = x_test[0]
image = np.expand_dims(image, axis=0)

label = np.zeros((1, 10))
label[:, 3] = 1

perturbation = generate_adversarial_pattern(image, label).numpy()

adversarial = image + epsilon * perturbation

prediction = model.predict(adversarial)

print('Vorhersage des Modells auf dem manipulierten Bild:',
        np.argmax(prediction))
```

Quellcode 10: Schritt 4

```
1:  import matplotlib.pyplot as plt
2:
3:  # Visualisieren des Originalbilds
4:  plt.figure()
5:  plt.imshow(np.squeeze(image))
6:  plt.title('Originalbild')
7:  plt.show()
8:
9:  # Visualisieren des manipulierten Bilds
10: plt.figure()
11: plt.imshow(np.squeeze(adversarial))
12: plt.title('Adversarial Beispiel')
13: plt.show()
```

4.1 LIME

In diesem Quellcode 11 importieren wir zunächst das LIME-Paket und erstellen ein LimeImageExplainer-Objekt. Dann definieren wir eine Funktion, die ein Eingabebild aufnimmt und die vorhergesagten Klassenwahrscheinlichkeiten unter Verwendung des Modells zurückgibt. Wir wählen ein zu erklärendes Bild und verwenden das explainer-Objekt, um eine Erklärung für dieses Bild zu erzeugen. Schließlich verwenden wir die Methode, um ein Bild zu erhalten, das die wichtigen Regionen des Eingabebildes für die vorhergesagte Klasse hervorhebt, und wir zeichnen die resultierende Erklärung.

4.2 SHAP

Es gibt verschiedene Methoden, um die Shapley Values zu berechnen. Eine der häufigsten Methoden ist die Kernel-Shap-Methode, bei der eine Stichprobe von Modellvorhersagen verwendet wird, um die Shapley-Werte zu schätzen. Diese Methode ist jedoch rechenaufwendig und kann bei großen Datensätzen und komplexen Modellen langsam sein. Eine andere Methode zur Berechnung von Shapley-Werten ist die Tree-Shap-Methode, die speziell für Entscheidungsbäume entwickelt wurde. Bei dieser Methode werden die Shapley-Werte basierend auf der Struktur des Entscheidungsbaums und der Reihenfolge, in der die Features im Baum untersucht werden, berechnet.

4.3 Saliency Map

Um eine Saliency Map zu erstellen, wird das neuronale Netzwerk auf das Eingabebild angewendet und die Aktivierung jedes Neurons berechnet. Anschließend wird ein Gradienten auf das Neuron zurückpropagiert, um zu ermitteln, wie stark sich die Ausgabe des Netzes

Quellcode 11: LIME auf dem Adversarial Example einsetzen

```
import lime
import lime.lime_image

# Create explainer object
explainer = lime.lime_image.LimeImageExplainer()

# Define function to predict class probabilities of input image
def predict_fn(images):
    return model.predict(images)

# Choose an image to explain
image_idx = 0
image = x_test[image_idx]

# Generate explanation for the chosen image
explanation = explainer.
      explain_instance(image, predict_fn, top_labels=1,
                       hide_color=0, num_samples=1000)

# Plot the explanation
explanation_image, mask =
            explanation.get_image_and_mask(explanation.top_labels[0],
                                           positive_only=True,
                                           num_features=5,
                                           hide_rest=False)
plt.imshow(explanation_image)
plt.title('LIME Explanation')
plt.show()
```

ändern würde, wenn sich die Aktivierung des Neurons ändern würde. Diese Werte werden gewichtet und auf das Eingabebild angewendet, um die Saliency Map zu erstellen. Eine Saliency Map ist somit eine Heatmap, die die wichtigsten Regionen des Bildes hervorhebt, auf die das Netzwerk reagiert hat. Durch die visuelle Darstellung der Saliency Map können Experten und Nutzer des Modells leichter verstehen, welche Regionen des Eingabebildes zur Entscheidungsfindung beigetragen haben. Zum Beispiel kann eine Saliency Map verwendet werden, um zu zeigen, welche Merkmale eines Röntgenbildes das neuronale Netzwerk dazu veranlasst haben, eine Diagnose zu stellen.

4.4 Vergleich der drei Ansätze

Shap und Lime sind beide Modelle-agnostische Methoden, die auf einer Vorhersage-abbauenden Technik basieren und die Auswirkungen von Features auf die Vorhersage des Modells quantifizieren. Shap basiert auf der Shapley-Wert-Theorie aus der Spieltheorie und ist mathematisch fundiert, während Lime auf lokalen Linearmodellen basiert und einfach zu implementieren ist.

Saliency Maps sind eine Technik, die auf der Analyse von Gradienten von Modellen basiert und zeigen, welche Bereiche eines Bildes von einem Modell zur Entscheidungsfindung genutzt wurden. Sie können insbesondere für die Bildinterpretation und die visuelle Erklärung von Deep-Learning-Modellen nützlich sein.

Welches Verfahren am besten geeignet ist, hängt von der Art des Modells, der Daten und der Fragestellung ab, die Sie beantworten möchten. Tatkräftig war die LIME Methode, durch die einfache Implementierung konnte es leicht auf die Adversarial Examples angewendet werden. Die Saliency Map war die zweitbeste Methode, da sie sehr gut die wichtigen Bereiche des Bildes hervorheben konnte. Die Shap Methode war die schlechteste Methode, da die Implementierung sehr kompliziert war und es nicht so gut funktioniert hat wie die anderen beiden Methoden. Ich konnte leider während dieser Zeit aufgrund von privaten Problemen nicht ausgiebiger mich damit befassen.

5 Verwandte Arbeiten

Bereits 2020 haben Gil Fidel, Ron Bitton und Asaf Shabtai daran geforscht. Sie haben es geschafft zu zeigen, dass man mit State of the Art Attacks (RQ1) Adversarial Examples aufdecken kann. Deren Model basierte auf einen großen Datensatz. Sie kamen zu dem Schluss, dass ihr vorgeschlagenes Verfahren zu einem Allgemeinen Framework weiter erweitert werden kann. Das erinnert an Antivirus- oder IDS-Systeme, die gutartige und bösartige Proben kontinuierlich sammeln und analysieren, Signaturen extrahieren und basierend auf diesen Signaturen Proben als bösartig oder gutartig klassifizieren oder zur manuellen Analyse weiterleiten. Ein Framework wie dieses, das sowohl die SHAP basierte Signaturen, die in diesem Dokument diskutiert werden als auch verwendete Signaturen in anderen könnten

State Of The Art Detektoren erweitern, sich gegen Adversarial Examples zu verteidigen.

In August 2022 haben Ching-Yu Kao, Junhao Chen, Karla Markert und Konstantin Böttinger mehrere Szenarien gefunden, in denen die Adversarial Samples nicht verworfen oder in der Warteschlange gehalten werden sollten, um auf menschliches Eingreifen zu warten. Sie haben herausgefunden, dass automatisierte Fahrsysteme ein Beispiel dafür ist. Dabei müssen die falschen Proben sofort korrigieren, indem der Fahrer wieder eingreift, um Gefahren zu vermeiden. In dem Artikel entwickelten sie die Interpretationsmethode innovativ für die Korrektur von Adversarial Examples. Somit waren sie die Ersten, die XAI-Methoden verwendet haben, um Adversarial Eingaben zu korrigieren. Außerdem kamen sie zu dem Schluss, dass iGOS-basierte Methoden CAM-basierte Methoden und andere grundlegende Methoden übertreffen.

In ihren Arbeiten wurden Explainable Artificial Intelligence (XAI) Methoden unter Adversarial Attack evaluiert, um herauszufinden, welche der drei prominenten Techniken – SHAP, LIME und Saliency Map – am besten geeignet ist, um Erklärungen für manipulierte Eingaben bereitzustellen. Adversarial Attacks sind gezielte Angriffe auf KI-Modelle, die darauf abzielen, deren Leistung durch Einführung von veränderten Eingaben zu beeinträchtigen. Durch das Verständnis, wie diese Angriffe die Leistung von XAI-Techniken beeinflussen, können wir bessere Modelle entwickeln und die Sicherheit und Zuverlässigkeit von KI-Anwendungen erhöhen.

Nach eingehender Analyse der verschiedenen XAI-Methoden und der Durchführung von Experimenten mit Adversarial Attacks, zeigt sich, dass SHAP (SHapley Additive exPlanations) unter den betrachteten Methoden am besten geeignet ist, um Erklärungen für manipulierte Eingaben zu liefern. Mehrere Gründe lassen sich für diese Präferenz anführen:

1. Konsistenz und Stabilität: Im Vergleich zu LIME (Local Interpretable Model-agnostic Explanations) und Saliency Maps zeigte SHAP eine höhere Konsistenz und Stabilität bei der Erklärung von Vorhersagen, selbst unter adversarialen Bedingungen. SHAP gewährleistet, dass die Erklärungen kohärent und fair sind, indem es auf der Spieltheorie basiert und Shapley-Werte verwendet, um die Wichtigkeit von Merkmalen zuzuweisen. 2. Lokale und globale Interpretierbarkeit: SHAP ermöglicht sowohl lokale als auch globale Interpretierbarkeit, indem es detaillierte Erklärungen für individuelle Vorhersagen und Zusammenfassungen der Merkmalswichtigkeit über alle Vorhersagen bereitstellt. Diese umfassende Perspektive trägt dazu bei, die Auswirkungen von Adversarial Attacks auf die Vorhersagen besser zu verstehen und entsprechende Gegenmaßnahmen zu ergreifen. 3. Modellagnostizität: SHAP ist modellagnostisch, d.h., es kann auf verschiedene Arten von KI-Modellen angewendet werden, was einen breiteren Anwendungsbereich und eine höhere Flexibilität im Vergleich zu LIME und Saliency Maps ermöglicht.

Trotz der Überlegenheit von SHAP gegenüber den anderen XAI-Methoden in Bezug auf Adversarial Attacks ist es wichtig zu betonen, dass keine der betrachteten Methoden immun gegen die Auswirkungen solcher Angriffe ist. Die Erklärungen, die von diesen Techniken bereitgestellt werden, können immer noch durch manipulierte Eingaben beeinflusst werden,

was zu einer Verringerung der Zuverlässigkeit und Vertrauenswürdigkeit der bereitgestellten Erklärungen führt. Daher ist es notwendig, weitere Forschungen zu betreiben, um die Robustheit von XAI-Methoden gegenüber Adversarial Attacks zu erhöhen und die bestehenden Techniken weiterzuentwickeln oder neue Ansätze zu entwickeln, die besser in der Lage sind, solchen Bedrohungen standzuhalten.

Schließlich sollte betont werden, dass die Wahl der am besten geeigneten XAI-Methode für eine bestimmte Anwendung nicht nur von ihrer Robustheit gegenüber Adversarial Attacks abhängt, sondern auch von anderen Faktoren wie der Domäne, in der das KI-Modell eingesetzt wird, der Komplexität des Modells und den spezifischen Anforderungen an die Erklärbarkeit. In einigen Fällen kann es sogar ratsam sein, mehrere XAI-Methoden in Kombination anzuwenden, um ein umfassenderes Verständnis der Vorhersagen und der zugrunde liegenden Prozesse zu gewährleisten.

Die Ergebnisse dieser Arbeit liefern wertvolle Erkenntnisse für Forscher, Entwickler und Anwender von KI-Systemen, indem sie ein tieferes Verständnis für die Leistung von XAI-Methoden unter Adversarial Attacks bieten und die Stärken und Schwächen der verschiedenen Ansätze aufzeigen. Die Identifizierung von SHAP als am besten geeignete Methode für die Bereitstellung von Erklärungen für manipulierte Eingaben ist ein wichtiger Schritt in Richtung der Entwicklung robuster und zuverlässiger KI-Systeme, die den Herausforderungen von Adversarial Attacks standhalten können.

Dennoch bleibt noch viel zu tun, um die Robustheit und Sicherheit von XAI-Methoden weiter zu erhöhen. Zukünftige Forschungen sollten sich auf die Entwicklung von Ansätzen konzentrieren, die die Anfälligkeit von XAI-Methoden für Adversarial Attacks reduzieren und ihre Fähigkeit verbessern, vertrauenswürdige und nützliche Erklärungen unter solchen Bedingungen zu liefern. Eine vielversprechende Richtung könnte darin bestehen, Methoden zur Erkennung und Abwehr von Adversarial Attacks direkt in den XAI-Prozess zu integrieren, um die Zuverlässigkeit der bereitgestellten Erklärungen weiter zu erhöhen.

Insgesamt zeigt deren Arbeit, dass die Erforschung von XAI-Methoden im Zusammenhang mit Adversarial Attacks von großer Bedeutung ist, um das volle Potenzial von KI-Systemen auszuschöpfen und vertrauenswürdige Lösungen für eine Vielzahl von Anwendungen zu entwickeln. Durch die kontinuierliche Verbesserung der XAI-Methoden und die Erhöhung ihrer Robustheit gegenüber Adversarial Attacks werden wir in der Lage sein, KI-Systeme zu schaffen, die sicherer, verständlicher und verantwortungsvoller sind – ein entscheidender Schritt für den verantwortungsvollen Einsatz von Künstlicher Intelligenz in unserer Gesellschaft.

6 Fazit

XAI (Explainable Artificial Intelligence) ist ein wichtiges Forschungsgebiet in der Künstlichen Intelligenz, das sich mit der Entwicklung von Techniken und Methoden zur Erklärung von

Entscheidungen von KI-Systemen beschäftigt. In vielen Anwendungsbereichen ist es wichtig zu verstehen, warum ein bestimmtes Modell eine bestimmte Entscheidung getroffen hat. Insbesondere ist es wichtig, zu verstehen, wie ein Modell auf manipulierte Eingaben reagiert, um Sicherheitslücken zu identifizieren und zu beheben.

In dieser Bachelorarbeit wurde untersucht, welche der drei Methoden SHAP, LIME und Saliency Map am besten geeignet ist, um Erklärungen für manipulierte Eingaben unter Adversarial Attack zu liefern. Eine Adversarial Attack ist eine Art von Angriff, bei dem ein Angreifer gezielt manipulierte Eingaben verwendet, um ein KI-Modell dazu zu bringen, falsche Entscheidungen zu treffen. Die drei Methoden unterscheiden sich in ihrem Ansatz zur Erklärung von Entscheidungen. SHAP basiert auf der Shapley-Wert-Theorie und gibt für jedes Feature in einem Modell an, welchen Beitrag es zur Vorhersage des Modells leistet. LIME nutzt lokale lineare Modelle, um die Entscheidungen des Modells in einem bestimmten Bereich zu approximieren. Saliency Map zeigt, welche Bereiche eines Bildes für die Entscheidungen des Modells am wichtigsten sind, indem sie die Gradienten des Modells berechnet.

Um die Leistung der drei Methoden bei der Erklärung von Entscheidungen unter Adversarial Attack zu vergleichen, wurden in der Arbeit verschiedene Experimente durchgeführt. Die Accuracy wurde getestet und bemessen, um die Robustheit der drei Methoden gegenüber Adversarial Attack zu evaluieren. Dann wurde untersucht, wie gut die drei Methoden in der Lage sind, manipulierte Eingaben zu erkennen und zu erklären.

In der Theorie wurde gezeigt, dass SHAP und LIME robust gegenüber Adversarial Attack sind und gute Ergebnisse bei der Erklärung von Entscheidungen unter normalen Bedingungen liefern. In meinem Experiment konnte ich das bei LIME sehr gut nachvollziehen da die Erklärungen sehr einfach waren. Allerdings sind sie weniger robust gegenüber Angriffen, die speziell auf ihre Methoden abzielen. Saliency Map hingegen ist weniger robust gegenüber Adversarial Attack, aber sie ist besser in der Lage, manipulierte Eingaben zu erkennen und zu erklären. Um die Ergebnisse der Experimente zu visualisieren, wird empfohlen, interaktive Dashboards und Heatmaps zu verwenden. Diese können den Benutzern eine intuitive Möglichkeit bieten, die Erklärungen zu interpretieren und zu vergleichen. Ein weiterer Vorteil dieser Visualisierungsarten ist, dass sie einfacher zu verstehen und zu bedienen sind als komplexere visuelle Darstellungen.

Insgesamt zeigt die Bachelorarbeit, dass die Wahl der am besten geeigneten Methode zur Erklärung von Entscheidungen unter Adversarial Attack von verschiedenen Faktoren abhängt, einschließlich der Art des Modells, der Daten und der Anwendungsbedingungen. SHAP und LIME eignen sich am besten für die Erklärung von Entscheidungen unter normalen Bedingungen, während Saliency Map besser geeignet ist, um manipulierte Eingaben zu erkennen und zu erklären. Die Verwendung interaktiver Dashboards und Heatmaps als Visualisierungsmethode kann die Interpretation der Erklärungen erleichtern und die Benutzerfreundlichkeit erhöhen.

In Zukunft könnten weitere Methoden und Techniken entwickelt werden, um die Robustheit

von Erklärungsmethoden gegenüber Adversarial Attack zu verbessern und gleichzeitig eine hohe Genauigkeit bei der Erklärung von Entscheidungen zu gewährleisten. Eine Möglichkeit könnte sein, mehrere Methoden zu kombinieren oder neue Methoden zu entwickeln, die speziell für die Erklärung von Entscheidungen unter Manipulationen von Eingaben konzipiert sind. Darüber hinaus könnte die Untersuchung von Erklärungsmethoden für Adversarial Attack auch für die Entwicklung von Sicherheitsmaßnahmen in der Künstlichen Intelligenz von Nutzen sein. Die Erkenntnisse könnten dazu beitragen, bessere Schutzmechanismen gegen Angriffe zu entwickeln und die Vertrauenswürdigkeit von KI-Systemen zu erhöhen.

Insgesamt zeigt die Bachelorarbeit, dass die Wahl der am besten geeigneten Methode zur Erklärung von Entscheidungen unter Adversarial Attack von verschiedenen Faktoren abhängt und eine sorgfältige Evaluation und Abwägung der Vor- und Nachteile jeder Methode erfordert. Die Verwendung von interaktiven Dashboards und Heatmaps als Visualisierungsmethoden kann dabei helfen, die Ergebnisse intuitiver und benutzerfreundlicher zu gestalten.

Abbildungsverzeichnis

Tabellenverzeichnis

Algorithmenverzeichnis

Quellcodeverzeichnis

Literatur

[Agg18] Aggarwal, C.C.: *Neural Networks and Deep Learning: A Textbook*. Springer International Publishing, 2018 https://books.google.de/books?id=achqDwAAQBAJ. – ISBN 9783319944630

[Bis06] Bishop, C.M.: *Pattern Recognition and Machine Learning*. Springer, 2006 (Information Science and Statistics). https://books.google.de/books?id=qWPwnQEACAAJ. – ISBN 9780387310732

[Bur19] Burkov, A.: *The Hundred-Page Machine Learning Book*. Andriy Burkov, 2019 https://books.google.de/books?id=0jbxwQEACAAJ. – ISBN 9781999579517

[GBC16] Goodfellow, I. ; Bengio, Y. ; Courville, A.: *Deep Learning*. MIT Press, 2016 (Adaptive Computation and Machine Learning series). https://books.google.de/books?id=Np9SDQAAQBAJ. – ISBN 9780262035613

[Gér19] Géron, A.: *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2019 https://books.google.de/books?id=HnetDwAAQBAJ. – ISBN 9781492032595

[GKP19] Gulli, A. ; Kapoor, A. ; Pal, S.: *Deep Learning with TensorFlow 2 and Keras: Regression, ConvNets, GANs, RNNs, NLP, and More with TensorFlow 2 and the Keras API*. Packt Publishing, 2019 (Expert insight). https://books.google.de/books?id=hebZzAEACAAJ. – ISBN 9781838823412

[GMR+18] Guidotti, Riccardo ; Monreale, Anna ; Ruggieri, Salvatore ; Turini, Franco ; Giannotti, Fosca ; Pedreschi, Dino: A Survey of Methods for Explaining Black Box Models. In: *ACM Comput. Surv.* 51 (2018), aug, Nr. 5. http://dx.doi.org/10.1145/3236009. – DOI 10.1145/3236009. – ISSN 0360–0300

[GPAM+14] Goodfellow, Ian J. ; Pouget-Abadie, Jean ; Mirza, Mehdi ; Xu, Bing ; Warde-Farley, David ; Ozair, Sherjil ; Courville, Aaron ; Bengio, Yoshua: *Generative Adversarial Networks*. https://arxiv.org/abs/1406.2661. Version: 2014

[GSS15] Goodfellow, Ian J. ; Shlens, Jonathon ; Szegedy, Christian: *Explaining and Harnessing Adversarial Examples*. 2015

[Lab20] Labs, DLT: Understanding Machine Learning & Deep Learning. (2020). https://dltlabs.medium.com/understanding-machine-learning-deep-learning-f5aa95264d61

[Mol20] Molnar, Christoph: *Interpretable Machine Learning*. Leanpub, 2020 https://books.google.de/books?id=jBm3DwAAQBAJ. – ISBN 9780244768522

[Mur12] Murphy, K.P.: *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012 (Adaptive Computation and Machine Learning series). https://books.google.de/books?id=NZP6AQAAQBAJ. – ISBN 9780262018029

[RHS20] Rami Haffar, Josep Domingo-Ferrer Najeeb Moharram J. Najeeb Moharram Jebreel ; Sánchez, David: *Explaining Image Misclassification in Deep Learning via Adversarial Examples*. https://crises-deim.urv.cat/web/docs/publications/conferences/1155.pdf. Version: 2020

[RM19] Raschka, S. ; Mirjalili, V.: *Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2*. Packt Publishing, 2019 https://books.google.de/books?id=sKXIDwAAQBAJ. – ISBN 9781789958294

[RN21] Russell, S.J. ; Norvig, P.: *Artificial Intelligence: A Modern Approach*. Pearson, 2021 (Pearson series in artificial intelligence). https://books.google.de/books?id=B4xczgEACAAJ. – ISBN 9781292401133

[Rot20] Rothman, D.: *Hands-On Explainable AI (XAI) with Python: Interpret, visualize, explain, and integrate reliable AI for fair, secure, and trustworthy AI apps*. Packt Publishing, 2020 https://books.google.de/books?id=2f30DwAAQBAJ. – ISBN 9781800202764

[Rud19] Rudin, Cynthia: *Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead*. 2019

[SB18] Sutton, R.S. ; Barto, A.G.: *Reinforcement Learning, second edition: An Introduction*. MIT Press, 2018 (Adaptive Computation and Machine Learning series). https://books.google.de/books?id=sWV0DwAAQBAJ. – ISBN 9780262039246

[Tra19] Trask, A.W.: *Grokking Deep Learning*. Manning, 2019 https://books.google.de/books?id=0zczEAAAQBAJ. – ISBN 9781638357209