

Explainable AI

Stand der Forschung und Technik

Master Thesis

Studienrichtung:

MAS Data Science

Autor:

Marc Habegger

Semester:

MAS HS19

Experte:

Max Kleiner, Prof. Dr. Arno Schmidhauser

Datum:

9. März 2020

Inhaltsverzeichnis

1 Management Summary	1
2 Einleitung	2
3 Was bedeutet Erklärbarkeit?	5
3.1 Unterschiedliche Anforderungen an eine Erklärung	5
4 Zielgebiete von XAI	7
4.1 Bessere Anwendungen durch Einblick in die Funktionsweise	7
4.2 Datenschutz	7
4.3 Sicherheit	8
4.4 Regulatorische Bedingungen	9
4.5 Haftungsfragen	9
5 Wann wird XAI eingesetzt?	10
5.1 Data Understanding	11
5.2 Data Preparation	11
5.3 Modeling	11
5.4 Evaluation	12
5.5 Deployment	13
6 Modellvarianten	14
6.1 Erklärbare / Whitebox Modelle	14
6.1.1 Lineare Regression	15
6.1.2 Logistische Regression	15
6.1.3 GLM/GAM	16
6.1.4 Decision Tree	16
6.1.5 RuleFit	17
6.1.6 Naïve Bayes	18
6.2 Schwer erklärbare / Blackbox Modelle	18
7 XAI Verfahren	19
7.1 Grad CAM	19
7.2 Occlusion Sensitivity	20
7.3 LRP	22
7.4 Local Surrogate (LIME)	24
7.5 TCAV	26
7.6 SVCCA	26
8 Konkrete Anwendungen von XAI	28
8.1 Bilderkennung: Klassifikation Hund - Katze	28
8.1.1 Eigenes CNN Modell	28
8.1.2 Vortrainiertes Modell	35

8.2 Texterkennung: Stimmungs-Analyse von Film-Bewertungen	40
8.2.1 Whitebox Modell	41
8.2.2 Blackbox Modell	47
9 Schwächen von Modellen erkennen	50
9.1 Diskriminierung durch Bias	50
9.2 Manipulierte Bilder: Adversarial Attacks	50
9.3 Manipulierte Daten: Data Poisoning	52
10 Fazit	54
11 Ausblick	56
11.1 Aktuelle Probleme	56
11.2 Wünschenswerte Entwicklungen	56
11.2.1 Mathematisch begründete XAI	56
11.2.2 Modell agnostische Methoden	57
11.2.3 Automatisierung	57
11.3 Fehlende Lösungen	57
12 Anhang	58
Akronyme	62
Glossar	64

1 Management Summary

Machine Learning (ML), kombiniert mit grossen Datensätzen, bietet seit einigen Jahren die Möglichkeit Arbeiten, welche bisher nur durch Menschen ausgeführt werden konnten, durch Computersysteme zu erledigen. Durch ML erzeugte Modelle werden in vielen Gebieten erfolgreich eingesetzt.

Es wird zwischen zwei Arten von Modellen unterschieden. Zum einen Whitebox Modelle, welche eine einfache, zumeist verständliche Struktur haben und bei denen nachvollziehbar ist, wie Daten verarbeitet werden. Die andere Art von Modellen nennt sich Blackbox. Diese sind von einer derart grossen Komplexität, dass es in der Regel nicht möglich ist nachzuvollziehen wie Daten darin verarbeitet werden.

Explainable Artificial Intelligence (XAI) hat das Ziel, auch die schwer verständlichen Blackbox Modelle für Menschen verständlich zu machen und Erklärungen für die produzierten Resultate zu erzeugen. Diese "Erklärbarkeit" von Modellen wird von einigen Akteuren (Politik, Zulassungsbehörden, Nichtregierungsorganisationen) gefordert bevor der Einsatz von Blackbox Modellen in sensiblen Gebieten wie beispielsweise Medizin akzeptiert wird.

Diese Arbeit listet verschiedene Techniken auf, welche benutzt werden können um die Arbeitsweise von Blackbox Modellen aufzuzeigen. Zusätzlich werden auch einige Whitebox Algorithmen vorgestellt, die als Alternative gegenüber Blackbox Modellen verwendet werden können.

XAI ist ein aktives Gebiet der Forschung, viele Forschungsartikel schlagen neue Methoden vor wie Modelle besser erklärt werden können. Aus diesen Arbeiten sind Werkzeuge, entstanden welche die Analyse von Modellen durch erzeugte Erklärungen, meistens in der Form von Visualisierungen, unterstützen. Die in dieser Arbeit gesammelten Erkenntnisse über die verwendeten Werkzeuge und welche Probleme dabei aufgetreten sind, können als Anhaltspunkte für den Einsatz in weiteren Projekten verwendet werden.

Durch eigene Beispiele und Verweise auf andere Arbeiten wird gezeigt wie XAI eingesetzt werden kann um den Entwicklern ein besseres Verständnis über ihre Modelle zu verschaffen und dadurch zu verbessern.

Es zeigte sich das XAI momentan nicht in der Lage ist allgemein gültige Aussagen über die Qualität und Sicherheit von Modellen zu treffen. Es ist unsicher ob XAI überhaupt jemals dazu in der Lage sein wird. Es gibt Forscher die dafür plädieren in kritischen Fällen nur Modelle zu verwenden welche gut verstanden und erforscht sind.

2 Einleitung

Computerprogramme bestehen in der Regel aus Tausenden bis Millionen Zeilen von Anweisungen, welche für die verschiedenen Aufgabengebiete zuständig sind. Einige der Programmroutine stellen die grafische Benutzeroberfläche dar, während andere sich um das Speichern und Laden von Dateien kümmern. Ein Teil der Anweisungen definieren Regeln nach denen Daten verarbeitet und analysiert werden. Solche Regeln werden von Fachspezialisten definiert und durch Software-Entwickler umgesetzt. Mit einem derartigen Vorgehen konnten viele alltägliche Probleme gelöst werden. Software ist inzwischen in der Wirtschaft wie im privaten Umfeld allgegenwärtig geworden.

Die Ausformulierung solcher Regeln nach denen sich Software verhalten soll, ist aber ein aufwendiger und fehlerträchtiger Prozess. Gewisse Gebiete sind durch die Komplexität der Aufgabenstellung nur rudimentär in Regeln zu fassen. Solche Gebiete sind unter anderem Bilderkennung, Text- oder Sprachverständnis. In diesen Gebieten zeigen Menschen und Tiere bedeutend bessere Fähigkeiten als Computerprogramme.

Während programmierte Regeln in Computer Programmen sofort zur Verfügung stehen, müssen Menschen und Tiere ihre Fähigkeiten oftmals über längere Zeit trainieren und üben. Bis die gewünschten Fähigkeiten in genügender Qualität vorhanden sind, können Jahre vergehen. Durch solche biologische Prozesse als Vorbild wurde die Disziplin Machine Learning (ML) entwickelt, welche auch Computer in die Lage versetzen soll, bislang nur bei Mensch und Tier gefundene Fähigkeiten, zu erlangen. ML wird seit den 1960er Jahren angewendet, allerdings waren die erzielten Resultate lange Zeit für viele Anwendungen ungenügend. Durch die Verfügbarkeit von grossen Datenmengen (Big Data, Cloud) und der gesteigerten Rechenleistung der Rechner wurden nach der Jahrtausendwende so gute Fortschritte erzielt, dass immer mehr Anwendungsmöglichkeiten für ML Lösungen gefunden wurden.

Oftmals wird auch von Artificial Intelligence (AI) gesprochen. Machine Learning (ML) kann als Teilgebiet von AI angesehen werden.

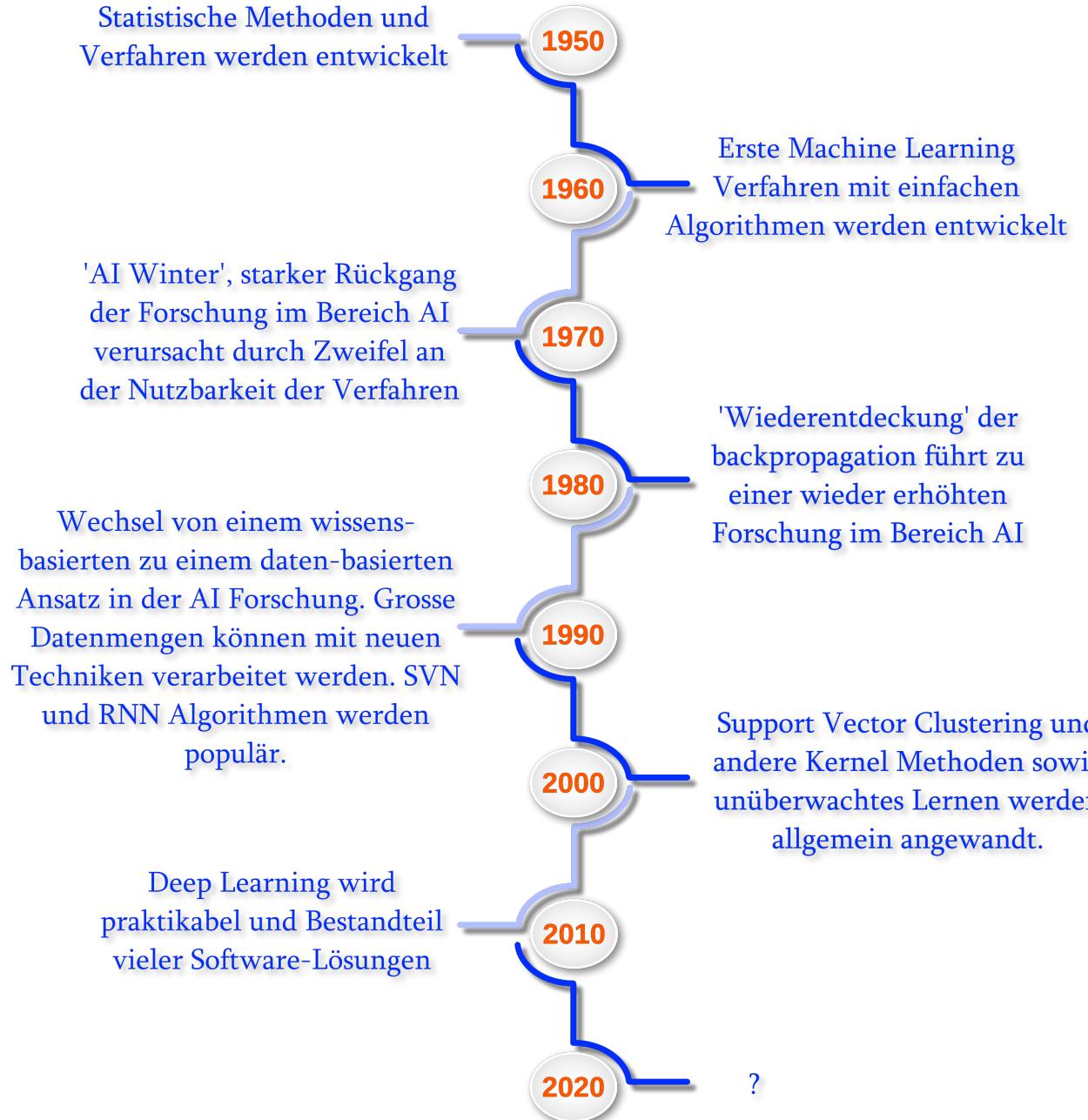


Abbildung 2.1: Entwicklung des Machine Learning als Zeitachse

Durch die Anwendung von Versuch und Irrtum bei kontinuierlicher Optimierung der internen Regeln erlangten ML Systeme bislang unerreichbare Stärken in vorher problematischen Gebieten. Allerdings ist der Preis dafür oftmals der, dass die automatisch erstellten Regeln für Menschen unverständlich und nicht nachvollziehbar sind. Für viele Dienste im Internet (Bildersammlungen, Empfehlungssysteme) werden in der Regel keine oder nur geringe Anforderungen an ein verständliches Modell gestellt. Aber es gibt einige Bereiche in denen besondere Ansprüche an die Nachvollziehbarkeit von Entscheidungen bestehen.

Exemplarisch werden hier einige dieser Gebiete aufgeführt:

Medizin

Machine Learning Anwendungen für die Krebserkennung bieten grosses Potenzial. Insbesondere die ermüdende Aufgabe auf Röntgen- oder MRT-Bildern Spuren eines Tumors zu erkennen, könnten durch ML abgelöst werden. Allerdings sind die Zulassungskriterien für solche Lösungen noch nicht definiert.

Justiz

Predictive Policing versucht mittels statistischer und Machine Learning Verfahren Orte oder Personengruppen zu erkennen, welche Schauplatz oder Täter/Opfer eines Verbrechens werden könnten.

Selbstfahrende Fahrzeuge

Obwohl selbstfahrende Fahrzeuge seit Jahren von allen grossen Fahrzeugherstellern entwickelt werden, sind immer noch viele Fragen bezüglich der Haftung und Zulassung offen.

Aufgrund des Mangels an Techniken um fortgeschrittene ML System zu verstehen, entstand deshalb ein neues Forschungsgebiet Explainable Artificial Intelligence (XAI), welches sich zum Ziel gesetzt hat Methoden und Werkzeuge zu entwickeln um ML Modelle zu analysieren.

3 Was bedeutet Erklärbarkeit?

Machine Learning (ML) erzeugt Resultate, welche je nach Anwendungsfall Entscheidungen für Klassen (Pferd, Schaf, Auto), Zuordnungen zu Gruppen (Premium-Kunde, Gelegenheitskäufer) oder numerische Werte (15 Grad Celsius am 3. April) sind. Da sowohl die Erzeugung des Modells als auch die Berechnung des Resultates automatisch erfolgt, können die Schritte auf dem Weg zu dem Resultat nicht direkt nachvollzogen werden.

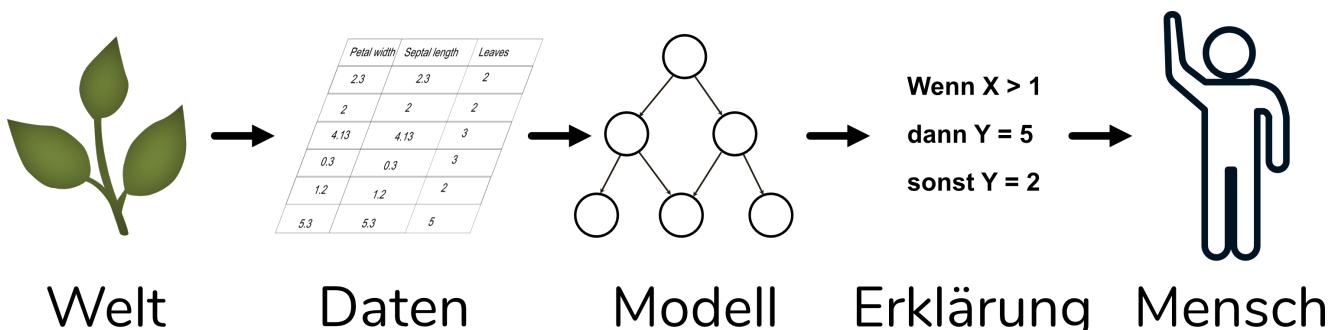


Abbildung 3.1: Ablauf einer erklärbaren Machine Learning Anwendung

Eine ML Lösung beginnt mit der Beobachtung von realen Ereignissen in der Welt. Dies können die Anzahl Blätter und deren Länge einer Pflanzengattung oder auch Häuserpreise in Brooklyn sein. Diese Beobachtungen werden gesammelt und bilden die Datengrundlage mit deren ein Modell erstellt wird. Aus diesem kann danach eine Erklärung erzeugt werden die Menschen hilft das Resultat besser zu verstehen.

Ein oft verwendetes Wort in diesem Zusammenhang ist die Interpretierbarkeit. Ein Modell, welches einfach und verständlich ist, hat eine hohe Interpretierbarkeit, es kann von Menschen einfach interpretiert werden.

3.1 Unterschiedliche Anforderungen an eine Erklärung

Eine Anwendung, welche Machine Learning (ML) einsetzt, kann in mehrere Bereiche unterteilt werden. Durch diese Aufteilung in verschiedene Komponenten ergeben sich unterschiedliche Anforderungen an die Erklärbarkeit (*Explainable and Interpretable Models in Computer Vision and Machine Learning, 2018*):

Daten

Aus der Sicht der Daten interessiert vor allem welcher Teil der Daten für das Ergebnis die grösste Relevanz hat. Basierend auf dieser Erkenntnis können die Daten gezielt erweitert oder reduziert werden, so dass ein ausgeglichenes Verhältnis erzeugt wird.

Modell

Kann man aus dem Modell ein Muster für eine bestimmte Kategorie ableiten? Dies kann helfen Fehlklassifizierungen von neuen Daten zu verhindern, in dem überprüft wird, ob das Modell die richtigen/plausiblen Features berücksichtigt.

Vorhersage

Das Erzeugen einer Erklärung weshalb ein bestimmtes Muster in den Daten zu der resultierenden Klassifizierung geführt hat. Dies ist insbesondere für Anwender / Kunden einer ML Lösung wichtig, um das Verständnis für die maschinelle Entscheidung zu erhöhen. Die daraus erzeugte Erklärung kann dazu genutzt werden um eine gesetzlich vorgeschriebene Anfechtbarkeit der Entscheidung zu ermöglichen.

Ebenso gibt es bei den Interessensgruppen unterschiedliche Anforderungen an die Erklärbarkeit einer ML Anwendung. Nach G. Ras (Ras et al., 2018) werden dabei folgende Gruppierungen unterschieden:

Experten

Diese Gruppe kann weiter unterteilt werden in

Forscher

entwickelt neue Methoden und Algorithmen, verbessert bestehende Algorithmen

Entwickler

setzt bestehende Methoden und Algorithmen ein, um eine konkrete Aufgabenstellung zu lösen

Benutzer

Auch bei den Benutzern gibt es verschiedene Ausprägungen

Eigentümer

Eigentümer/Auftraggeber benötigen performante Anwendungen welche gesetzeskonform sind

Anwender

sind daran interessiert wie eine Entscheidung, die sie betrifft, zustande kam

Person deren Daten verwendet werden

sind an Datenschutz, ohne eine Missbrauchsmöglichkeit ihrer Daten, interessiert

Anspruchsgruppen (Stakeholder)

wie Regulierungsbehörden oder Fachgremien setzen die Rahmenbedingungen für den Einsatz von ML Anwendungen in heiklen Gebieten

Die Anforderungen an ein erklärbare Modell unterscheiden sich sehr stark, je nach betrachteter Komponente und der Anwendergruppe. Daraus ergibt sich, dass verschiedene Techniken benötigt werden um AI / ML Lösungen generell erklärt zu machen. Ein wichtiges Merkmal einer Erklärung ist die Verständlichkeit für das jeweilige Publikum. Während Forscher auf dem Gebiet des ML mathematische Formeln bevorzugen, sind für Anwender Erklärungen auf der Basis einfacher Texte oder Bilder verständlicher.

Alle diese Punkte zeigen weshalb ein einziger, einheitlicher, Ansatz nicht zielführend ist. Die Werkzeuge und Methoden müssen passend auf den Kreis der Empfänger abgestimmt werden.

4 Zielgebiete von XAI

4.1 Bessere Anwendungen durch Einblick in die Funktionsweise

Ein Grundsatz jedes Machine Learning (ML) Projektes lautet, dass der Erfolg nicht garantiert ist. Ausgehend von bestehenden Daten kann man nicht sicher sein, dass diese ausreichende Informationen liefern, um ein Modell zu entwickeln, welches den Anforderungen entspricht. Selbst wenn genügend Daten vorhanden sind, besteht immer noch die Problematik, dass unzählige Algorithmen mit wiederum unzähligen Parametern existieren, welche angewendet werden können. Es gibt Lösungen, um mit dieser Problematik umzugehen. Dennoch ist es in der Regel ein langwieriger und oftmals teurer Prozess um ML Modelle auf das gewünschte Qualitätsniveau zu bringen.

Erkenntnisse durch Explainable Artificial Intelligence (XAI) Techniken können den Entwicklern helfen Irrwege und Probleme frühzeitig zu erkennen, um so Arbeitszeit und/oder Rechenleistung beim Definieren und Berechnen der Modelle einzusparen.

Ebenso kann ein Modell falsche Resultate liefern, sei es weil die ursprünglichen Trainingsdaten unvollständig waren, oder weil neue, vorher unbekannte, Bedingungen aufgetreten sind. Eine Erklärung weshalb dieses falsche Resultat erzeugt wurde, kann den Entwickler in die Lage versetzen ein verbessertes Modell zu erzeugen. Dies kann sowohl in der Phase der Aufarbeitung der Daten, welche durch das Modell verarbeitet werden sollen (preprocessing) oder beim Erzeugen eines Modelles (training) hilfreich sein.

4.2 Datenschutz

Jede ML Lösung setzt grosse Datenmengen voraus. Diese müssen den Anforderungen des Datenschutzes entsprechend aufbereitet und gegebenenfalls anonymisiert werden. Dennoch besteht die Gefahr, dass Modelle Rückschlüsse auf die Daten zulassen, mit denen sie trainiert wurden.

Das Gebiet Datenschutz ist durch Vorstöße von NGO's, Politikern und vor allem durch die Datenschutz-Grundverordnung (DSGVO) der EU, in den Fokus von Regierungsbehörden gelangt.

Der jüngste Bericht der Datenethikkommission (DEK) der Deutschen Regierung [8] geht in Kapitel 3 konkret auf ML Anwendungen ein.

Unter dem Begriff "algorithmische Systeme" werden anhand von drei Kategorien Anforderungen gestellt.

Die von der DEK definierten Bereiche sind:

1. algorithmenbasierte Entscheidungen sind menschliche Entscheidungen, die sich auf algorithmisch berechnete (Teil-)Informationen stützen

2. algorithmengetriebene Entscheidungen sind menschliche Entscheidungen, die durch die Ergebnisse algorithmischer Systeme in einer Weise geprägt werden, dass der tatsächliche Entscheidungsspielraum und damit die Selbstbestimmung des Menschen eingeschränkt werden
3. algoritmendeterminierte Entscheidungen führen automatisiert zu Konsequenzen, so dass im Einzelfall keine menschliche Entscheidung mehr vorgesehen ist

Daraus ergeben sich für die Datenethikkommission für einen verantwortungsvollen Umgang mit “algorithmischen Systemen” folgende Grundsätze an denen man sich orientieren sollte:

- Menschenzentriertes Design
- Vereinbarkeit mit gesellschaftlichen Grundwerten
- Nachhaltigkeit
- Qualität und Leistungsfähigkeit
- Robustheit und Sicherheit
- Minimierung von Verzerrungen und Diskriminierung
- Transparenz, Erklärbarkeit und Nachvollziehbarkeit
- Klare Rechenschaftsstrukturen

XAI kommt vor allem in den Bereichen “Minimierung von Verzerrungen und Diskriminierung” und “Transparenz, Erklärbarkeit und Nachvollziehbarkeit” zum tragen, kann aber auch bei “Robustheit und Sicherheit” und “Qualität und Leistungsfähigkeit” helfen, die gestellten Anforderungen zu erfüllen.

In Kapitel 4.3 wird näher auf die Gefahren eingegangen welche durch die Anwendung von ML entstehen können. XAI Werkzeuge können Entwicklern Schwachpunkte aufzeigen oder Experten in die Lage versetzen, angewandte Modelle nachträglich auf Probleme zu untersuchen.

4.3 Sicherheit

Die Sicherheit von ML Modellen ist durch verschiedene Angriffsmethoden gefährdet.

Model Stealing Attacks

Ziel eines solchen Angriffes ist es, entweder direkt die internen Parameter eines Modells zu extrahieren, oder ein neues Modell zu erstellen das sich gleich oder möglichst ähnlich zu dem Original verhält. Dadurch können Geschäftsgeheimnisse entwendet oder Wettbewerbsvorteile eliminiert werden.

Membership Inference Attacks

Diese Art von Angriff versucht herauszufinden ob ein Datensatz dazu verwendet wurde das vorliegende Modell zu trainieren. Neben Datenschutzproblemen kann dies auch zu weiteren Angriffsflächen führen.

Adversarial Image Perturbations (AIP)

Oft genügen kleine Änderungen an einem Bildinhalt um ein neuronales Netz dazu zu bringen die vorhergesagte Klasse zu wechseln. Wenn ein Angreifer eine solche Anfälligkeit entdeckt, kann es möglich sein ein Bild so zu verändern, dass es für menschliche Augen gleich (oder beinahe) aussieht wie das Original, jedoch mit einem völlig anderen Ergebnis. Es liegt auf der Hand, dass ein solches Verhalten eines Modells für Systeme einer Zutrittskontrolle unerwünscht ist. Es sind jedoch auch andere Formen der Täuschung denkbar, welche für den Betreiber zu Verlusten führen können (Waren Rücksendungen, Pfandflaschen Automaten, Qualitätskontrollen bei Lieferanten). Dieses Thema wird näher in Kapitel 9.2 erläutert.

In der Arbeit von Seong Joon Oh "Towards Reverse-Engineering Black-Box Neural Networks" (Oh et al., 2019) werden XAI Methoden vorgestellt mit denen Blackbox Modelle auf diese Probleme untersucht werden können.

4.4 Regulatorische Bedingungen

In vielen Bereichen des Alltags gelten Regeln, welche Diskriminierungen verhindern oder von der Gesellschaft als ungerecht empfundene Handlungen untersagen. Dies kann von einer Transportpflicht eines Verkehrsbetriebes bis hin zu einem Kündigungsschutz während einer Schwangerschaft reichen. Dies gerät oft in Konflikt mit dem Recht auf Vertragsfreiheit, die es Personen und Firmen frei lässt, mit welchen Vertragsparteien Geschäfte getätigter werden. Durch die Anwendung von AI Systemen können solche Diskriminierungen unbeabsichtigt eingeführt werden. Kaum eine Bank könnte eine Regel aufrechterhalten, welche Personen mit dunkler Hautfarbe Kredite verweigert. Ein Modell, das einem Bias unterliegt, kann jedoch genau solch eine Entscheidung bewirken.

Grundsätzliche Bedenken wegen dem Missbrauchspotential führten die Stadt San Francisco zu einem Verbot von Gesichtserkennungssystemen durch die Polizei und anderen Behörden [29]. Auch die Datenschutz-Grundverordnung der EU stellt Bedingungen für den Einsatz von ML auf. Hierzu existiert ein Leitfaden des Branchenverbandes bitkom, welcher diese Aspekte erläutert [3]. Einige Organisationen, beispielsweise *OpenAI* [43], versuchen Standards voranzutreiben welche die Industrie zu einem "fairen" Einsatz von AI und ML bringen soll. Auch Firmen definieren interne Leitfäden für den Umgang mit diesen Technologien. Google als Beispiel hat zu diesem Zweck einen Satz Regeln aufgestellt: *AI at Google: our principles* [46].

XAI kann insbesondere bei der Begründung von Entscheidungen, wie sie zum Beispiel von der DSGVO gefordert wird, einen wichtigen Beitrag leisten.

4.5 Haftungsfragen

Unfälle, welche durch ML Techniken verursacht, oder zumindest begünstigt wurden, stellen eine Gefahr bei deren Einsatz dar. Bei selbstfahrenden Fahrzeugen ist die Frage nach der Schuld des Herstellers naheliegend, wie dieser Unfall mit einem Tesla zeigte: *Tesla Autopilot System Found Probably at Fault in 2018 Crash* [4].

Aber auch eine Diskriminierung durch ein ML Modell mit einem Bias kann eine Klage durch die Betroffenen hervorrufen. XAI kann eingesetzt werden um das Risiko zu verringern, indem Schwachstellen in Modellen frühzeitig erkannt werden, siehe Kapitel 9.

5 Wann wird XAI eingesetzt?

Ein Projekt, welches Machine Learning (ML) einsetzt, durchläuft in der Regel mehrere Phasen. Die Anwendung von XAI ist nicht in jeder Phase sinnvoll. Dieses Kapitel soll einen kurzen Überblick über die Schritte hin zu einem fertigen Modell geben und erläutern an welchen Punkten XAI eingesetzt werden kann.

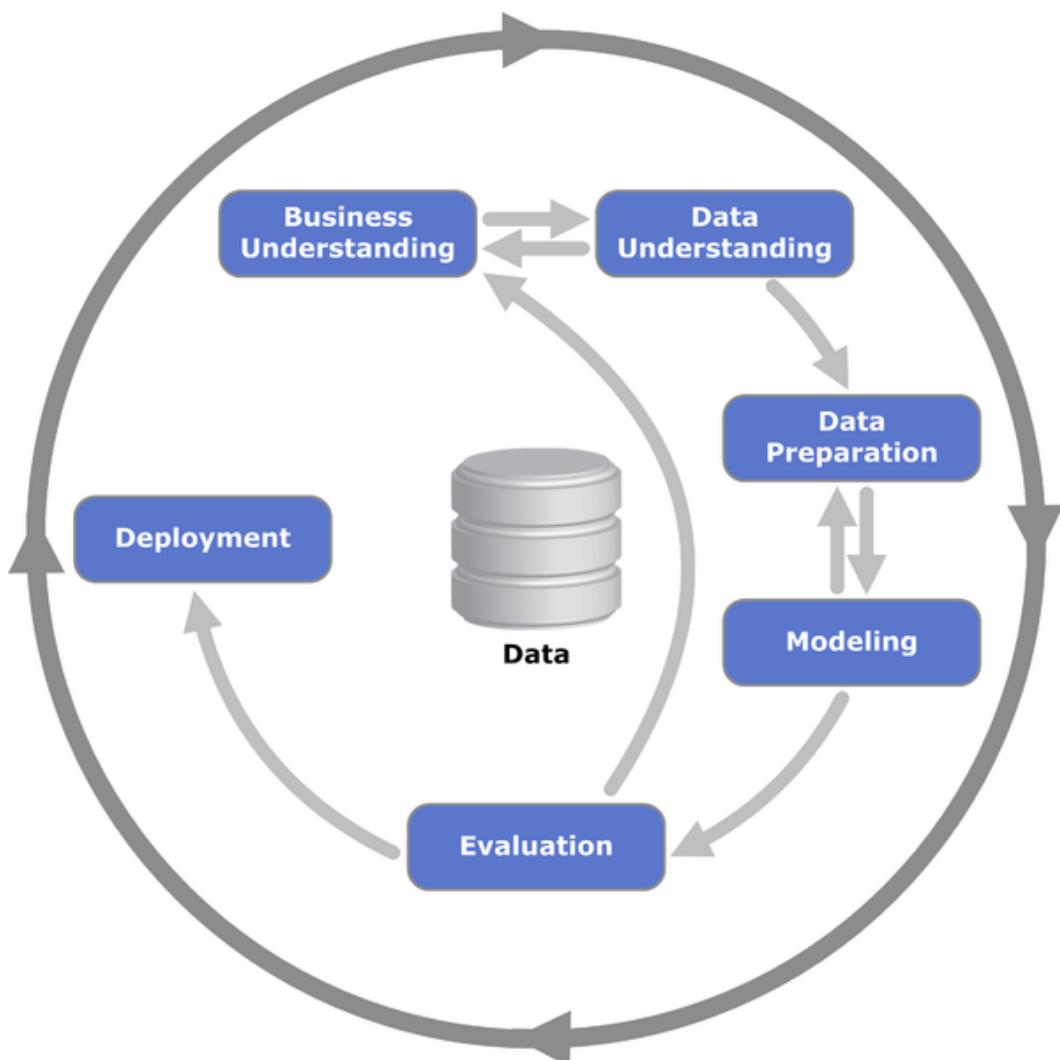


Abbildung 5.1: CRISP-DM Prozessablauf

Quelle: commons.wikimedia.org¹

¹Kenneth Jensen (https://commons.wikimedia.org/wiki/File:CRISP-DM_Process_Diagram.png), „CRISP-DM Process Diagram“, <https://creativecommons.org/licenses/by-sa/3.0/legalcode>

Abhängig von der gewählten Art des Modells und der gewünschten Art der Erklärung ist die Anwendung von XAI in verschiedenen Phasen möglich.

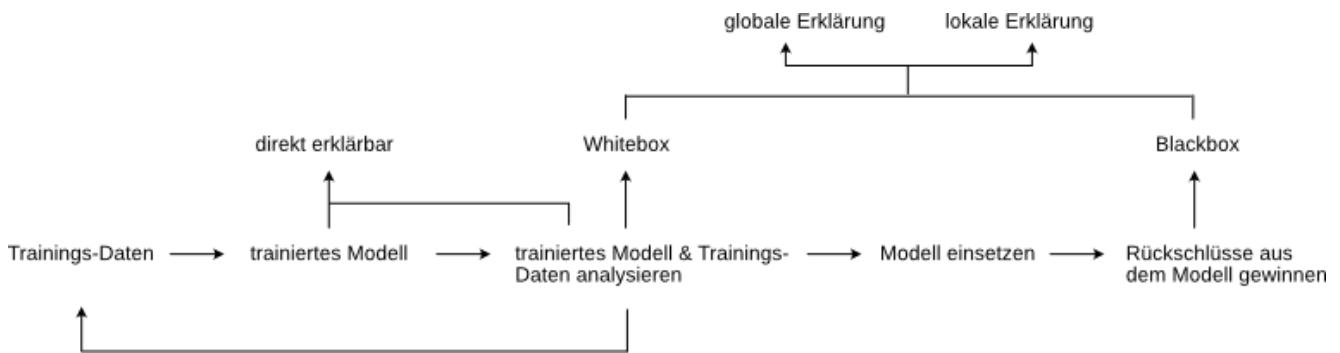


Abbildung 5.2: Erklärungen im ML Ablauf

Es ist ebenfalls möglich ein bereits eingesetztes Modell nachträglich zu untersuchen. Die Vorgehensweise entspricht dabei einer Blackbox Analyse wie bei einem Modell aus einer externen Quelle.

5.1 Data Understanding

Die Phase 'Data Understanding' ist allgemein als Exploration bekannt. In diesem Abschnitt eines ML Projektes verschafft man sich einen Überblick über die Daten und gewinnt Informationen darüber welche weiteren Schritte notwendig sind. Die Werkzeuge, die man zu diesem Zweck braucht, sind in der Regel einfacher und vor allem schneller als komplexere XAI Techniken, die sich mit den in späteren Phasen erzeugten Modellen beschäftigen. Während der Exploration werden häufig Diagramme und Grafiken erzeugt, da diese schnell interpretiert werden können.

5.2 Data Preparation

Data Preparation, oder auch Preprocessing genannt, ist ein notwendiger Schritt bei dem Daten auf die Verwendung für das Training eines Modells vorbereitet werden. Oft wird auch eine Auswahl der später verwendeten Daten (Features) getroffen, nicht immer macht es Sinn alle verfügbaren Daten einzusetzen.

5.3 Modeling

In der Modeling Phase entscheidet man sich für die Art des Modells, das man trainieren möchte. Oftmals trifft man eine Auswahl mehrerer unterschiedlicher Algorithmen und entscheidet sich nach einem vereinfachten Training für den Algorithmus mit dem besten Ergebnis.

Nach Seong Joon Oh (2019) hat die Art des Modells grossen Einfluss auf die Möglichkeiten der Erklärbarkeit. Generell wird unterschieden zwischen

Whitebox Modelle

Einfachere Modelle, diese weisen oft geringere Erfolgsquoten als Blackbox Modelle auf. Auch fehlen manchen dieser Modelle Möglichkeiten um komplexere Probleme wie Zusammenhänge zwischen Eigenschaften (feature interaction) zu modellieren. Im Gegenzug sind diese Modelle bedeutend einfacher zu erklären und interpretieren.

Blackbox Modelle

Blackbox Modelle liefern oft sehr gute Ergebnisse. Durch den komplexen inneren Aufbau sind sie aber schwer zu verstehen und es ist nicht möglich den Einfluss einer einzelnen Eigenschaft (feature) auf das Ergebnis aufzuzeigen. Auch Zusammenhänge zwischen den einzelnen Eigenschaften sind schwer zu entdecken.

Wenn eine Anforderung in der Aufgabenstellung eine vollständige Erklärbarkeit fordert, sind Whitebox Modelle sicherlich zu bevorzugen. Durch XAI können nun aber auch die schwierig zu verstehenden Blackbox Modelle nachträglich erklärt werden. Allerdings ist die Qualität (Aussagekraft, Vollständigkeit, Verständlichkeit) einer derart erzeugten Erklärung oft kleiner als die gegebene Verständlichkeit eines Whitebox Modells. Hier gilt es abzuwegen wie die Punkte Erklärbarkeit gegen Verständlichkeit gewichtet werden.

Man kann auch wie Rudin (Rudin, 2018) argumentieren, dass in heiklen Aufgabengebieten auf Blackbox Modelle zugunsten von Whitebox Modellen verzichtet werden soll.

5.4 Evaluation

In der Evaluation Phase werden die Eigenschaften eines Modells überprüft. Neben Überprüfung der Vorhersagequalität werden die Fehlerraten untersucht und mit den Anforderungen abgeglichen. XAI kann in dieser Phase wertvolle Einsichten in das Modell liefern. Diese können dazu verwendet werden um das Modell in einem weiteren Durchlauf zu verbessern. Zudem kann durch Prüfung auf Bias und andere Schwachstellen abgeschätzt werden, ob durch den Einsatz dieses Modells Probleme entstehen.

Wenn man Erklärungen von einem Modell erstellen will, muss man sich zuerst über den Spielraum der gewünschten Erklärung sicher sein. Man unterscheidet dabei zwischen einer lokalen (local interpretability) und einer globalen (global interpretability) Erklärung.

Globale Erklärungen

Globale Erklärungen können das Verständnis für den Umgang mit sämtlichen Daten liefern. Dies ist besonders hilfreich, wenn geprüft werden soll, ob das Modell einem Bias unterliegt. Informationen zu den aufgelisteten Verfahren können in den untenstehend aufgeführten Artikeln gefunden werden.

Verfahren für globale Erklärungen

- Trepan (Craven & Shavlik, 1995)
- Extract Rules (1994)
- Oracle Guides (Johansson & Niklasson, 2009)
- BETA (Lakkaraju et al., 2017)

Lokale Erklärungen

Lokale Erklärungen gelten für eine bestimmte Klasse und erklären weshalb das Modell diese Klasse als Vorhersage gewählt hat. Diese Arbeit befasst sich nur mit lokalen Erklärungen, sie sind einfach zu prüfen und anzuwenden.

Verfahren für lokale Erklärungen

- Grad CAM 7.1
- Occlusion Sensitivity 7.2
- Gradients Input
- Layer-wise Relevance Propagation (LRP) 7.3

Aus der Vielzahl von Werkzeugen, die existieren um ML Modelle zu analysieren, gilt es die für den jeweiligen Use Case relevanten Werkzeuge anzuwenden. Eine Arbeit mehrerer IBM Forscher zeigt verschiedene Bibliotheken auf und gibt einen Leitfaden für deren Anwendung (Arya et al., 2019).

5.5 Deployment

Als Deployment bezeichnet man die Einführung eines ML Modells in ein produktiv nutzbares System. XAI kann nun von Anwendern geforderte Begründungen für eine Entscheidung liefern oder bei Problemen helfen das Modell zu verbessern.

6 Modellvarianten

6.1 Erklärbare / Whitebox Modelle

Falls eine Anforderung besteht, dass die erzeugten Resultate interpretierbar sein müssen, kann direkt ein grundsätzlich interpretierbares Modell erzeugt werden. Der Preis dafür kann jedoch eine geringere Performance sein. Natürlich ist in diesem Fall XAI unnötig.

Folgende Algorithmen gelten generell als interpretierbar:

- Linear Regression 6.1.1
- Logistic Regression 6.1.2
- Generalized Additive Models (GAM) oder Generalized Linear Models (GLM) 6.1.3
- Decision Tree 6.1.4
- Rule Fit 6.1.5
- Learned fair representations (LFR)
- Monotonic gradient boosting (M-GBM)
- Private aggregation of teacher ensembles (PATE)
- Scalable Bayesian rule list (SBRL)
- Supersparse linear integer models (SLIM)
- Naïve Bayes Classifier
- K-Nearest Neighbors

Diese Liste hat keinen Anspruch auf Vollständigkeit, zudem können auch diese Modelle sehr komplex werden was die Verständlichkeit reduziert.

Auswahlhilfe Whitebox Modell

Als einfache Entscheidungshilfe welches Modell am besten zu der gestellten Aufgabe passt, kann diese Tabelle benutzt werden.

Als Methode kann entweder eine Klassifikation (klass.), Regression (regr.) oder sogar beide Arten angewandt werden. "Linear" bedeutet in diesem Fall, dass die Eingangsdaten sich gleichförmig auf das Resultat auswirken. "Monoton" trifft auf Fälle zu bei denen Änderungen in den Eingangsdaten sich in der gleichen Richtung (positiv oder negativ) auf das Resultat widerspiegeln. "Zusammenhang" bedeutet, dass zwei Eigenschaften zusammen einen stärkeren Einfluss haben als nur für sich alleine betrachtet (feature interaction).

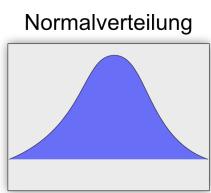
¹Quelle: *Interpretable Machine Learning, A Guide for Making Black Box Models Explainable* [39]

Algorithmus	Linear	Monoton	Zusammenhang	Methode
Linear regression	Ja	Ja	Nein	regr.
Logistic regression	Nein	Ja	Nein	klass.
Decision trees	Nein	einige	Ja	klass.,regr.
RuleFit	Ja	Nein	Ja	klass.,regr.
Naïve Bayes	Nein	Ja	Nein	klass.
k-nearest neighbors	Nein	Nein	Nein	klass.,regr.

Tabelle 6.1: Entscheidungstabelle erklärbare Modelle ¹

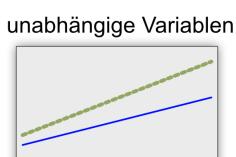
6.1.1 Lineare Regression

Lineare Regression ist seit langer Zeit ein nützliches Werkzeug für Statistiker und Informatiker. Die Zusammenhänge zwischen dem berechneten Ergebnis und den Eingangsvariablen können einfach nachvollzogen werden. Lineare Regression ist weit verbreitet, auch in nicht Informatik nahen Gebieten wie Medizin oder Soziologie. Ein Nachteil dieser Methode ist jedoch eine kleinere Leistungsfähigkeit in Bezug auf die Vorhersagequalität, so dass heutzutage oftmals auf leistungsfähigere, jedoch schlechter verständliche, Algorithmen zurückgegriffen wird. Insbesondere im Gebiet der Klassifikation zeigt die lineare Regression Schwächen.



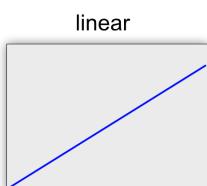
Die Formel der linearen Regression lautet:

$$y = \beta_0 + \beta_1 x_1 + \dots + \beta_p x_p + \epsilon$$



Um die lineare Regression erfolgreich anzuwenden müssen drei Bedingungen erfüllt sein:

- Normalverteilte Daten
- Die Variablen sind unabhängig, d.h. die Werte beeinflussen sich nicht, im Gegensatz zum Beispiel bei Geschlecht und Schwangerschaft
- Die vorhergesagten Werte sind linear



Wenn diese Bedingungen nicht erfüllt sind, kann die lineare Regression kaum erfolgreich angewandt werden.

Abbildung 6.1: Voraussetzungen
lineare Regression

6.1.2 Logistische Regression

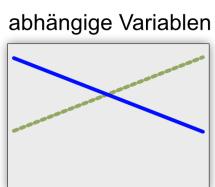
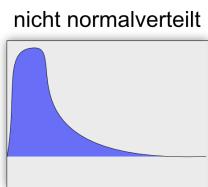
Während lineare Regression für numerische Problemstellungen verwendet wird, ist die logistische Regression ein Werkzeug für Klassifizierungen.

$$\text{logistic}(\eta) = \frac{1}{1 + \exp(-\eta)}$$

Grundsätzlich unterscheidet die logistische Regression zwischen zwei Klassen, eine sogenannte binäre Klassifikation. Man kann aber die logistische Regression zu einer multinomialen logistischen Regression erweitern, welche mehrere Klassen unterstützt. Die logistische Regression hat die meisten Vor- und Nachteile der linearen Regression, wobei die schwierigere Interpretation des Modells gegenüber einem linearen Modell in diesem Zusammenhang ein wichtiger Nachteil ist. Gegenüber anderen Klassifikatoren bietet die logistische Regression jedoch den grossen Vorteil, dass nicht nur die Klasse, sondern auch die Wahrscheinlichkeit für diese Klasse als Resultat erzeugt wird.

6.1.3 GLM/GAM

Lineare Regression hat einige Schwächen wie z. Bsp. die Annahme der Normalverteilung. Die Variablen sollten nicht korreliert sein. Bei einem nichtlinearen Zusammenhang zwischen den Daten und dem Resultat kann lineare Regression ebenfalls nicht eingesetzt werden. Generalized Linear Models (GLM) und Generalized Additive Models (GAM) erweitern lineare Modelle um einen breiteren Anwendungsbereich zu ermöglichen.



Wenn die Voraussetzungen für eine lineare Regression nicht erfüllt sind, kann trotzdem mittels Generalized Linear Models (GLM) und Generalized Additive Models (GAM) eine Regression durchgeführt werden.

Die Formeln für die beiden Varianten lauten:
GAM

$$g(E_Y(y|x)) = \beta_0 + \beta_1 x_1 + \dots + \beta_p(x_p)$$

GLM

$$g(E_Y(y|x)) = \beta_0 + f_1(x_1) + f_2(x_2) + \dots + f_p(x_p)$$

wobei GLM in der Formel von GAM den Term

$$\beta_j x_j$$

durch eine Funktion ersetzt

$$f_j(x_j)$$

Abbildung 6.2: Ausschluss-Bedingungen
lineare Regression

Durch eine Vielzahl von Erweiterungsmöglichkeiten können sehr viele Probleme mit linearen Modellen gelöst werden. Da diese Methoden bereits längere Zeit verwendet werden, ist die Erfahrung damit gross. Mathematik- oder Statistiksoftware haben in der Regel GLM und GAM integriert. Als Nachteil gilt aber generell eine schlechtere Performance gegenüber komplexeren Verfahren. Zudem sinkt die, prinzipiell vorhandene, Interpretierbarkeit mit der Anzahl Erweiterungen des klassischen linearen Modells.

6.1.4 Decision Tree

Ein Decision Tree (Entscheidungsbaum) kann bei einer geringen Anzahl von Parametern einfach visualisiert werden und gibt einen guten Überblick über die internen Abläufe, die zu einem Resultat führen.

Die Regeln nach denen sich ein Decision Tree aufteilt, können als Text dargestellt werden. Intuitiv besser verständlich sind jedoch grafische Darstellungen, welche entweder den Baum als Struktur oder als Fläche darstellen.

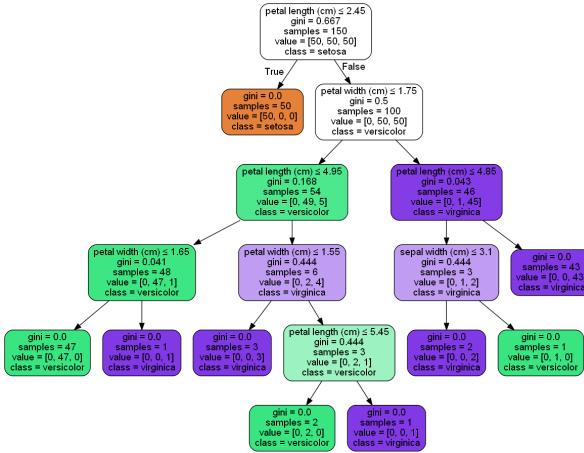


Abbildung 6.3: Entscheidungsbaum visualisiert.

```

1 | --- petal width (cm) <= 0.80
2 | | --- class: 0
3 | --- petal width (cm) > 0.80
4 | | --- petal width (cm) <= 1.75
5 | | | --- class: 1
6 | | | --- petal width (cm) > 1.75
7 | | | | --- class: 2

```

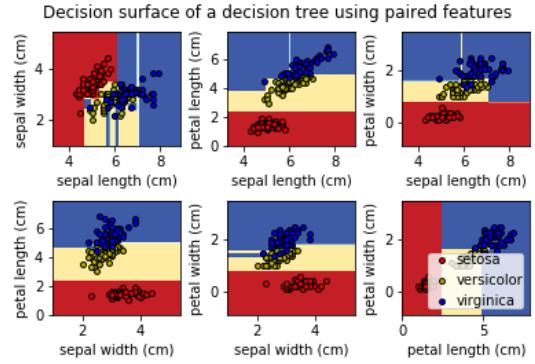


Abbildung 6.4: Entscheidungsbaum als Flächen dargestellt

Der für diese Visualisierungen verwendete Source Code ist in Kapitel 12 aufgeführt und verwendet *scikit-learn* [55].

6.1.5 RuleFit

RuleFit (Friedman & Popescu, 2008) verwendet Entscheidungsbäume um daraus Regeln abzuleiten, welche neue Features erzeugen. Diese neu erzeugten Features werden dann von einem linearen Modell interpretiert. Durch den Einsatz des linearen Modells verspricht RuleFit, in Kombination mit der von diesem Algorithmus gewohnten Interpretierbarkeit, eine bessere Performance.

Eine Python Bibliothek, welche RuleFit implementiert, ist *skope-rules* [59]. Dies ist ein Beispiel aus der scope-rules Dokumentation wie durch RuleFit generierte Regeln aussehen können:

```

1 12 rules have been built.
2 The 5 most precise rules are the following:
3 BILL_AMT2 > 517.5 and PAY_1 > 1.5 and PAY_AMT_old_std <= 2563.13525391
4 BILL_AMT_old_std <= 9353.94921875 and PAY_1 > 1.5 and PAY_2 > -0.5
5 PAY_2 > 1.5 and PAY_AMT_old_mean <= 12955.75 and PAY_old_mean > 0.625
6 BILL_AMT2 > 1870.0 and PAY_2 > 1.5 and PAY_AMT_old_mean <= 2525.375
7 BILL_AMT1 > 410.0 and PAY_1 > 1.5 and PAY_old_mean > 0.125

```

Abbildung 6.5: scope-rules Ausgabe der Regeln²

²Quelle: <https://skope-rules.readthedocs.io/>

6.1.6 Naïve Bayes

Naïve Bayes Filter werden seit einiger Zeit sehr erfolgreich für die Spam Klassifikation von Emails eingesetzt. Aber auch für andere Klassifikationen ist Naïve Bayes aufgrund der schnellen Berechnung bei guten Resultaten beliebt. Der Name kommt von der “naiven” Annahme, dass die Features voneinander unabhängig sind, die sogenannte “Annahme von Unabhängigkeit”. Obwohl das in der Realität fast nie der Fall ist, funktioniert Naïve Bayes in der Regel sehr gut.

$$P(C_k|x) = \frac{1}{Z} P(C_k) \prod_{i=1}^n P(x_i|C_k)$$

Naïve Bayes gilt als interpretierbares Modell aufgrund der Annahme der Unabhängigkeit. Für jedes Feature kann einfach bestimmt werden wie stark der Einfluss auf die Vorhersage ist, da man die bedingte Wahrscheinlichkeit interpretieren kann.

6.2 Schwer erklärbare / Blackbox Modelle

Wenn aufgrund der geforderten Performance oder den vorhandenen Daten ein Verfahren eingesetzt wird, das als nicht interpretierbar gilt, kann man XAI Werkzeuge einsetzen um eine Erklärung zu generieren. Diese Techniken werden im Kapitel 7 näher erläutert.

Grundsätzlich gelten alle Arten von komplexeren neuronalen Netzen mit vielen Ebenen (Deep Neural Network) als schwer interpretierbar, beispielsweise:

- Convolutional Neural Network (CNN)
- Long short-term memory (LSTM)
- Generalized Additive Models (GAM)
- Recurrent Neural Network (RNN)

Allerdings gelten auch Whitebox Modelle mit vielen Parametern als schwer erklärbar. Es ist möglich einen Decision Tree mit 10'000 Eingangsparametern zu trainieren, welcher dreistellige Tiefen erreichen kann. In diesen Fällen verwischt die Grenze zwischen White- und Blackbox.

7 XAI Verfahren

7.1 Grad CAM

Grad CAM ist eine Technik (Selvaraju et al., 2016) um die, für eine Bildklassifikation relevanten Bereiche eines Bildes, hervorzuheben. Dies geschieht auf der Basis von Gradienten. Grad CAM gilt als Verfahren für lokale Erklärbarkeit und wird somit immer zusammen mit einer gewählten Klassifikation angewendet.

In diesem Beispiel wurden die fünf wahrscheinlichsten Klassen eines Deep Neural Network für das Bild links oben dargestellt. Während die relevantesten Bereiche (gelb gefärbt) um den Kopf und vor allem um die Ohren der Katze sind, zeigt die Darstellungen für die unpassende Klasse “toilett_tissue” auf Bereiche des Bodens.

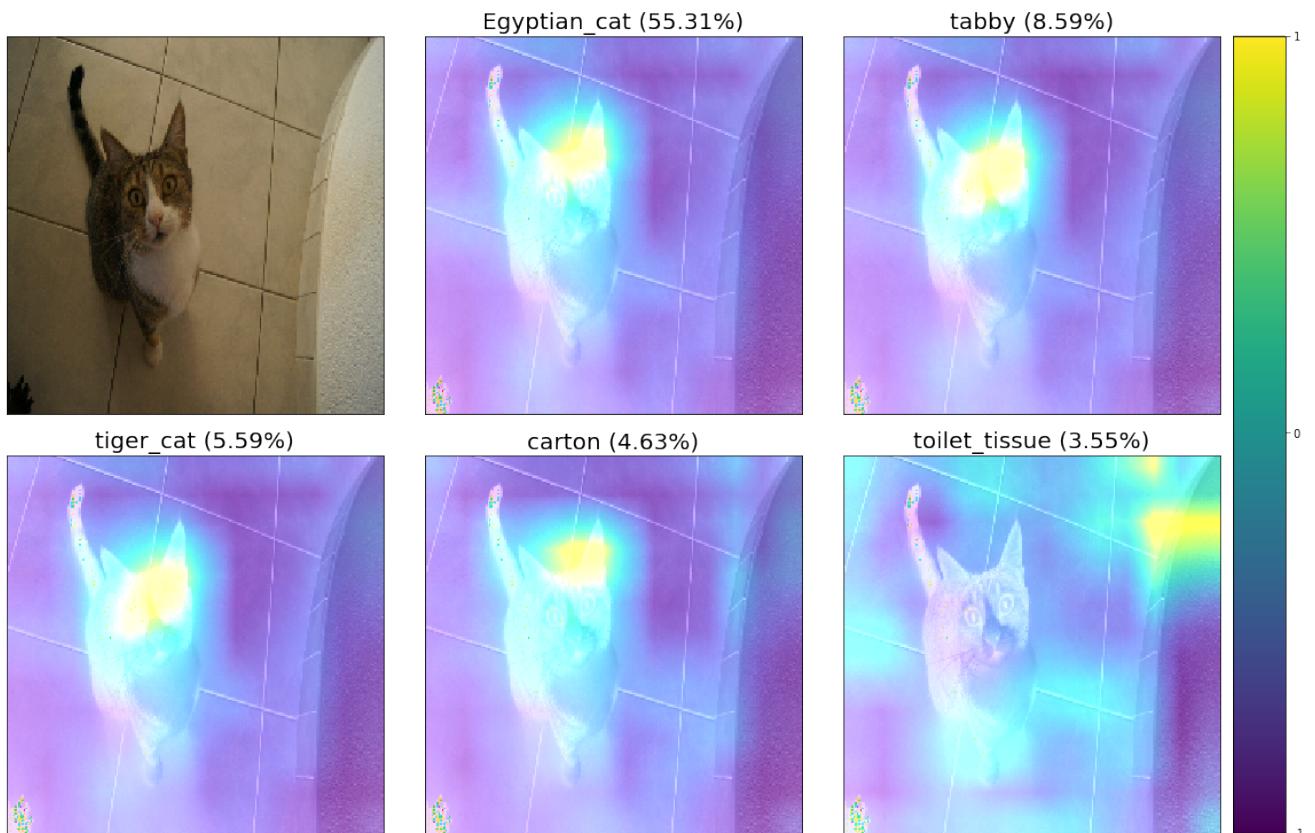


Abbildung 7.1: Grad CAM Analyse für verschiedene Klassen

Die Pixel, welche einen positiven Einfluss auf die gewählte Klassen haben, sind gelb dargestellt, während negative Einflüsse blau eingefärbt wurden. Das obenstehende Bild wurde mit dem Python Skript [23] erstellt.

Anwendung von Grad CAM auf Videos

Da die Berechnung von Grad CAM schnell ist, kann sie einfach auf Videosequenzen angewendet werden. Ein Beispiel wie eine solche Analyse in bewegten Bildern aussieht, kann in folgendem Artikel gefunden werden: *Visual explanation for video recognition* [16]

Visualisierung mit tf_explain

Um ein Bild der Klasse mit der höchsten Wahrscheinlichkeit zu erzeugen kann dieses Python Skript verwendet werden. Benötigt werden dazu die Bibliotheken *Tensorflow* [63] und *tf-explain* [38], sowie das Modell *VGG16 Modell für Imagenet Klassifikationen* [64].

```
1 import tensorflow as tf
2 from keras.applications.vgg16 import preprocess_input
3 from keras.applications.vgg16 import decode_predictions
4 from tf_explain.core.grad_cam import GradCAM
5
6 from matplotlib import pyplot as plt
7
8 model = tf.keras.applications.VGG16(weights="imagenet", include_top=True)
9
10 imageOrig = tf.keras.preprocessing.image.load_img('D:/Master Thesis/Repo/Test
11     Images/tabby.2.JPG', target_size=(224, 224))
12 imageArr = tf.keras.preprocessing.image.img_to_array(imageOrig) #output Numpy
13     -array
14
15 imageReshaped = imageArr.reshape((1, imageArr.shape[0], imageArr.shape[1],
16     imageArr.shape[2]))
17
18 image = preprocess_input(imageReshaped)
19 predictions = model.predict(imageReshaped)
20
21 import numpy as np
22 prediction = np.argsort(predictions)[0,::-1][:-1]
23
24 labels = decode_predictions(predictions, 15)
25
26 img = tf.keras.preprocessing.image.img_to_array(imageOrig)
27 data = ([img], None)
28
29 explainer = GradCAM()
30 img = explainer.explain(class_index=prediction[0], model=model, layer_name='
31     block5_conv3', validation_data=data)
32 plt.imshow(img)
33 plt.show()
```

Listing 7.1: Grad CAM Visualisierung für die wahrscheinlichste Klasse

7.2 Occlusion Sensitivity

Eine weitere Variante um relevante Bildbereiche aufzudecken ist Occlusion Sensitivity, vorgestellt durch Zeiler und Fergus im Jahr 2013 (Zeiler & Fergus, 2013). Occlusion Sensitivity bestimmt, durch erzeugte Störungen im Ursprungsbild, den Einfluss der einzelnen Bildbereiche auf die Vorhersage.



Die Abbildung links erläutert die Vorgehensweise des Occlusion Sensitivity Algorithmus. Occlusion Sensitivity entfernt unterschiedliche Bildbereiche des Originalbildes und misst dabei den Einfluss auf die Klassifizierung. Die Grösse des verdeckten Bildbereiches bestimmt die Auflösung der schlussendlich erzeugten Heatmap. Allerdings wird bei einem kleineren Bereich auch die Anzahl an Durchläufen (und somit die Bearbeitungszeit) erhöht, welche nötig sind um das ganze Bild abzudecken.

Abbildung 7.2: Occlusion Sensitivity Beispiel

Quelle: *Network Visualization Based on Occlusion Sensitivity* [9]

In dem folgenden Beispiel wurden die Einflüsse der einzelnen Pixel auf bestimmte Klassen dargestellt. Die Färbung zeigt dabei ob der Einfluss negativ (blau) oder positiv (gelb) auf die Klassifikation für die bestimmte Klasse ist. Diese Visualisierung wurde durch die Bibliothek *tf-explain* [38] und dem vortrainierten Modell *VGG16 Modell für Imagenet Klassifikationen* [64] erzeugt.

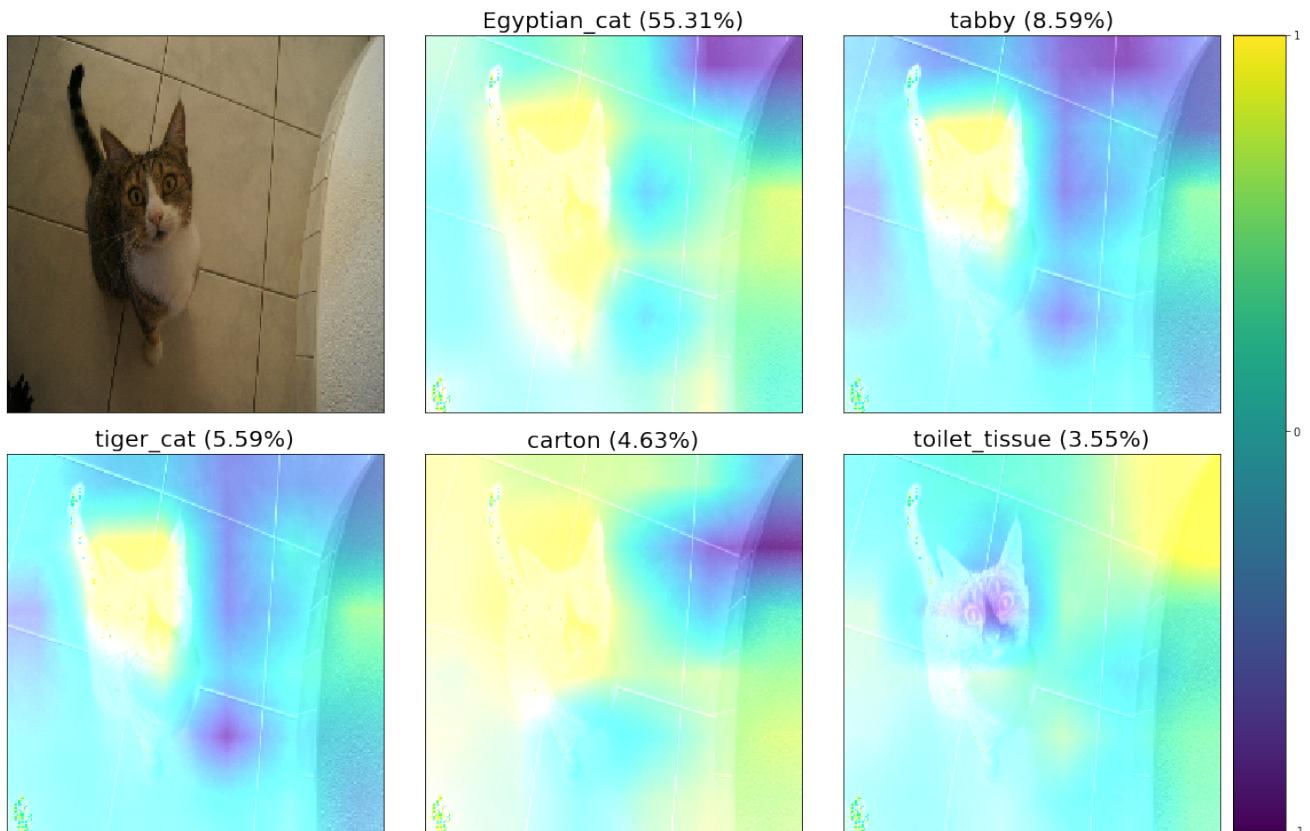


Abbildung 7.3: Testbild mit Occlusion Sensivity

Während bei den drei Klassen, welche für Katzenrassen stehen (Egyptian_cat, tabby, tiger_cat), hauptsächlich der Körper oder Kopf der Katze als relevant gilt, sind bei den letzten Klassifikationen, die Objekte darstellen (carton, toilet_tissue), vor allem Hintergrundbereiche wichtig.

Visualisierung mit tf_explain

Um ein Bild der Klasse mit der höchsten Wahrscheinlichkeit zu erzeugen kann dieses Python Skript verwendet werden. Benötigt werden dazu die Bibliotheken *Tensorflow* [63] und *tf-explain* [38] sowie das Modell *VGG16 Modell für Imagenet Klassifikationen* [64].

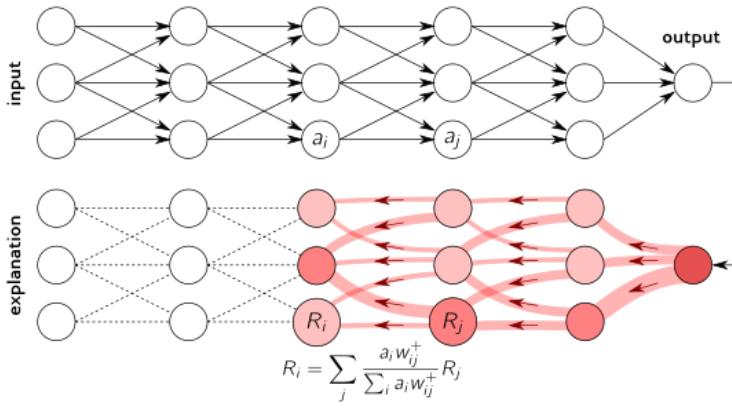
```
1 import tensorflow as tf
2 from keras.applications.vgg16 import preprocess_input
3 from keras.applications.vgg16 import decode_predictions
4 from tf_explain.core.occlusion_sensitivity import OcclusionSensitivity
5
6 from matplotlib import pyplot as plt
7
8 model = tf.keras.applications.vgg16.VGG16(weights="imagenet", include_top=True)
9
10 imageOrig = tf.keras.preprocessing.image.load_img('D:/Master Thesis/Repo/Test
11     Images/tabby.2.JPG', target_size=(224, 224))
12 imageArr = tf.keras.preprocessing.image.img_to_array(imageOrig) #output Numpy
13     -array
14
15 imageReshaped = imageArr.reshape((1, imageArr.shape[0], imageArr.shape[1],
16     imageArr.shape[2]))
17
18 import numpy as np
19 prediction = np.argsort(predictions)[0, ::-1][:1]
20
21 labels = decode_predictions(predictions, 15)
22
23 img = tf.keras.preprocessing.image.img_to_array(imageOrig)
24 data = ([img], None)
25
26 explainer = OcclusionSensitivity()
27 img = explainer.explain(class_index=prediction[0], model=model, patch_size=
28     40, validation_data=data)
29 plt.imshow(img)
30 plt.show()
```

Listing 7.2: Occlusion Sensitivity Visualisierung für die wahrscheinlichste Klasse

Der Parameter "patch_size" auf Zeile 27 definiert die Grösse des ausgeblendeten Bereiches. Ein kleinerer Wert erhöht die Auflösung des ausgegebenen Bildes, verlängert jedoch auch die Berechnungszeit.

7.3 LRP

Layer-wise Relevance Propagation (LRP) ist eine Technik welche bereits 2015 vorgestellt wurde (Bach et al., 2015). LRP kann verwendet werden um neuronale Netze zu untersuchen und ist für viele Problemstellungen wie Bilderkennung, Textanalyse oder Spracherkennung einsetzbar.



LRP durchläuft ein neuronales Netz rückwärts und berechnet so den Einfluss jedes Eingangs-Neurons.
Diese Vorgehensweise wird als "Deep Taylor Decomposition" bezeichnet.

Abbildung 7.4: LRP Berechnung

Quelle: www.heatmapping.org

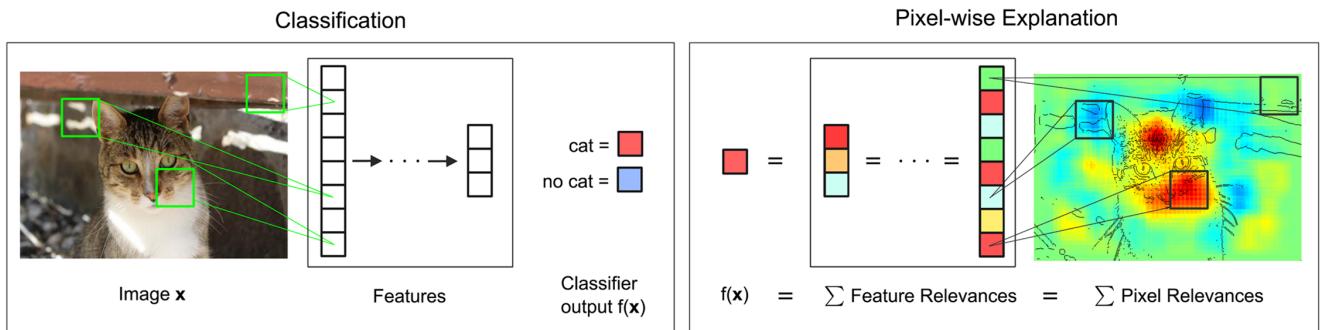


Abbildung 7.5: Pixel basierte Relevanz

Quelle: Bach (2015)

Während Pixel basierte Erklärungstechniken wie Grad CAM lokale Einflüsse hervorheben, setzt das LRP Verfahren auf globale Einflüsse in den Bildern. Dies kann in der Visualisierung dazu führen dass andere Bildbereiche hervorgehoben werden.

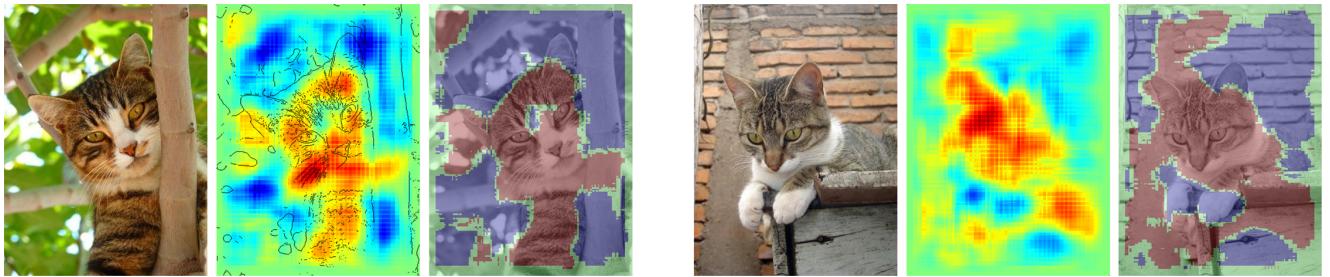


Abbildung 7.6: Vergleich Pixel-wise und LRP

Quelle: Bach (2015)

Die Erläuterung der Funktionsweise der Deep Taylor Decomposition findet sich auf der Webseite [A Quick Introduction to Deep Taylor Decomposition](#) [47].

Eine Beispielanwendung des Fraunhofer Instituts zeigt den Einsatz von LRP in den Gebieten Bilderkennung, Textanalyse und Visual Question Answering [Explainable AI Demos](#) [27].

Die Python Implementierung der LRP Toolbox (Lapuschkin et al., 2016) ist auf github vorhanden: [The LRP Toolbox for Artificial Neural Networks](#) [33].

7.4 Local Surrogate (LIME)

Die Technik LIME wurde 2016 erstmals vorgestellt (M. T. Ribeiro et al., 2016). Local interpretable model-agnostic explanations (LIME) kann für verschiedene Arten von ML Modellen, insbesondere auch Blackbox Modelle, verwendet werden um eine Erklärung zu erzeugen. Dabei wird durch stetiges Verändern eines Eingangsbildes der Einfluss auf das Resultat geprüft. Mit den veränderten Eingangsdaten und den durch das Blackbox Modell erzeugten Resultaten wird ein neues Modell trainiert, das anschliessend untersucht werden kann.

Folgende Schritte werden bei der Anwendung von LIME durchgeführt:

- Die Klasse für die man eine Erklärung erstellen will muss festgelegt werden
- Die ursprünglichen Daten werden verändert und die Resultate des Blackbox Modells für diese Daten werden aufgezeichnet
- Die neu erzeugten Datensätze werden nach der Nähe zu der gesuchten Klasse gewichtet
- Ein neues Modell mit den gewichteten (neuen) Datensätzen wird erzeugt
- Die Vorhersage des Blackbox Modells wird durch Interpretation des neu generierten Modells erklärt

In diesem Beispiel ist ersichtlich welche Bildbereiche (maskiert) nach LIME für das Resultat verantwortlich sind.

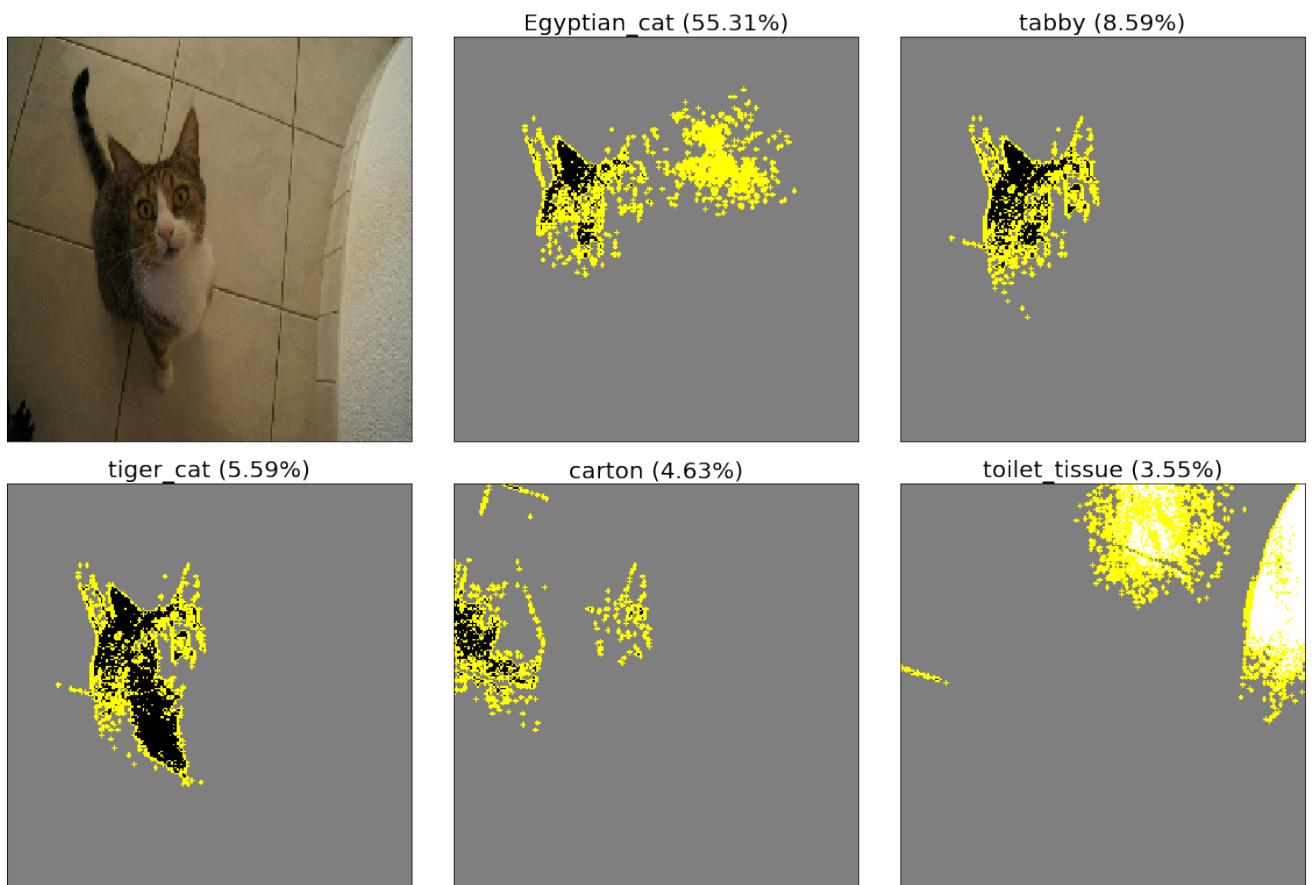


Abbildung 7.7: Darstellung relevanter Bildinhalte durch LIME

Gegenüber den vorherigen Methoden ist LIME durch die Erzeugung temporärer Modelle bedeutend aufwendiger und dadurch auch langsamer.

Visualisierung mit lime

Um ein Bild der Klasse mit der höchsten Wahrscheinlichkeit zu erzeugen kann dieses Python Skript verwendet werden. Benötigt werden dazu die Bibliotheken *Tensorflow* [63] und das Package *lime* aus *Tutorial - Image Classification Keras* [66], sowie das Modell *VGG16 Modell für Imagenet Klassifikationen* [64].

```
1 import tensorflow as tf
2 from keras.applications.vgg16 import preprocess_input
3 from keras.applications.vgg16 import decode_predictions
4
5 from matplotlib import pyplot as plt
6
7 model = tf.keras.applications.VGG16(weights="imagenet", include_top=True)
8
9 imageOrig = tf.keras.preprocessing.image.load_img('D:/Master Thesis/Repo/Test
10     Images/tabby.2.JPG', target_size=(224, 224))
11 imageArr = tf.keras.preprocessing.image.img_to_array(imageOrig) #output Numpy
12     -array
13
14 imageReshaped = imageArr.reshape((1, imageArr.shape[0], imageArr.shape[1],
15     imageArr.shape[2]))
16
17 image = preprocess_input(imageReshaped)
18 predictions = model.predict(imageReshaped)
19
20 import numpy as np
21 prediction = np.argsort(predictions)[0, ::-1][:1]
22
23 labels = decode_predictions(predictions, 15)
24
25 import lime
26 from lime import lime_image
27
28 imgAsArray = tf.keras.preprocessing.image.img_to_array(imageOrig)
29 out = []
30 x = np.expand_dims(imgAsArray, axis=0)
31 x = preprocess_input(x)
32 out.append(x)
33 imagesStack = np.vstack(out)
34
35 explainer = lime_image.LimeImageExplainer()
36
37 explanation = explainer.explain_instance(imagesStack[0], model.predict,
38     top_labels=1, hide_color=0, num_samples=1000)
39
40 from skimage.segmentation import mark_boundaries
41
42 temp, mask = explanation.get_image_and_mask(explanation.top_labels[0],
43     positive_only=True, num_features=5, hide_rest=True)
44 image = mark_boundaries(temp / 2 + 0.5, mask)
45 plt.imshow(image)
46 plt.show()
```

Listing 7.3: LIME Visualisierung für die wahrscheinlichste Klasse

7.5 TCAV

Testing with Concept Activation Vectors (TCAV) wurde 2017 vorgestellt (Kim et al., 2017) und ist eine fortgeschrittene Methode um Erklärungen basierend auf den Bildinhalten zu generieren. Zu diesem Zweck werden zusätzliche Modelle als Beispiele für Bildinhalte erzeugt.

Ein als Zebra klassifiziertes Bild kann so zum Beispiel damit begründet werden, dass auf dem Bild Streifen und ein Pferd entdeckt wurden.

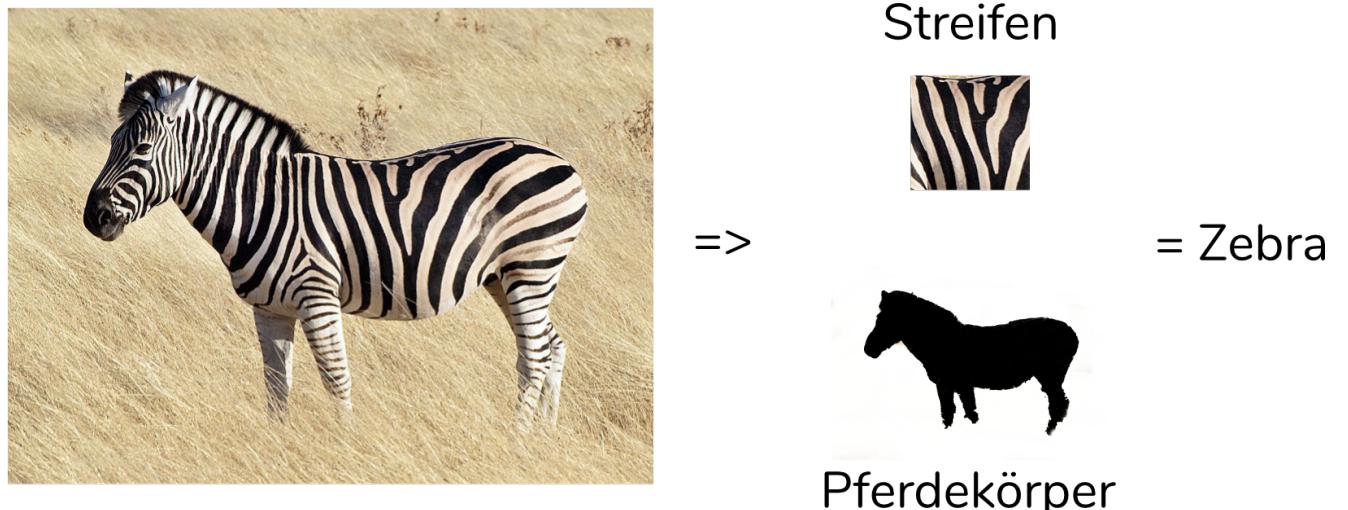


Abbildung 7.8: Darstellung Vorgehensweise TCAV

Da bei diesem Verfahren für jede Kategorie von Bildbestandteilen ein neuronales Netz trainiert werden muss, und für jedes dieser Netze Beispieldaten vorhanden sein müssen, ist der Aufwand für den Einsatz von TCAV gross.

Eine Implementierung dieses Verfahrens kann auf Github gefunden werden [30].

7.6 SVCCA

Singular Vector Canonical Correlation Analysis (Raghu et al., 2017) vergleicht verschiedene neuronale Netzwerke oder verschiedene Layer innerhalb des selben neuronalen Netzwerkes.

Durch den Vergleich der Vektoren verschiedener Klassen innerhalb eines Netzes kann auf die Ähnlichkeit der Klassen untereinander rückgeschlossen werden. Die beiden Klassen "Husky" und "Eskimo Dog" werden in der untenstehenden Grafik als parallel verlaufende, beinahe überlappende Linien dargestellt, was auf die starke Ähnlichkeit der beiden Hunderassen hinweist.

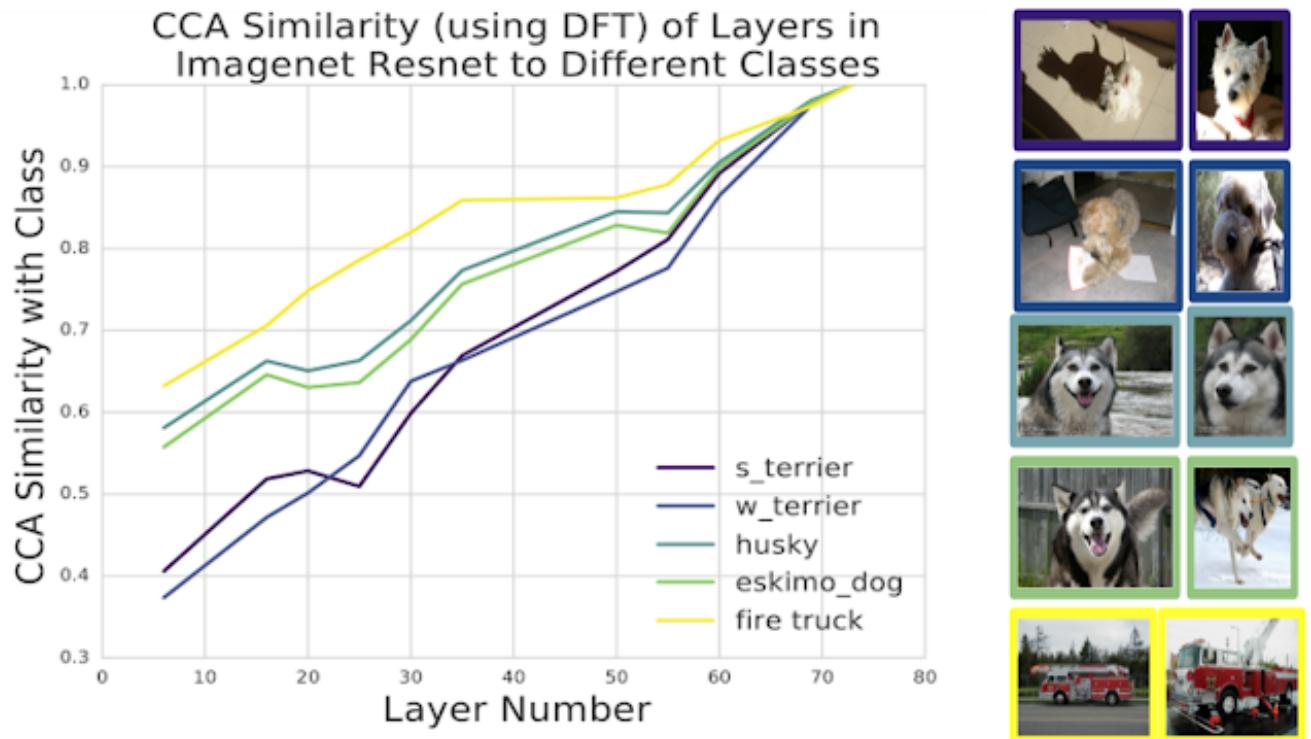


Abbildung 7.9: Vergleich verschiedener Klassen mit SVCCA¹

Durch die Information welche Klassen ähnlich sind und ab welchem Trainingsaufwand zwei Klassen erfolgreich unterschieden werden können, ist es möglich die Trainingszeit zu reduzieren. Es können auch die Trainingsdaten für bestimmte Klassen erhöht oder qualitativ verbessert werden um eine einfachere Unterscheidung der Klassen zu erzielen.

Weitere Informationen und Erklärungen zu diesem Verfahren befinden sich auf der Seite *Interpreting Deep Neural Networks with SVCCA* [48].

¹Quelle: Google AI Blog, Interpreting Deep Neural Networks with SVCCA

8 Konkrete Anwendungen von XAI

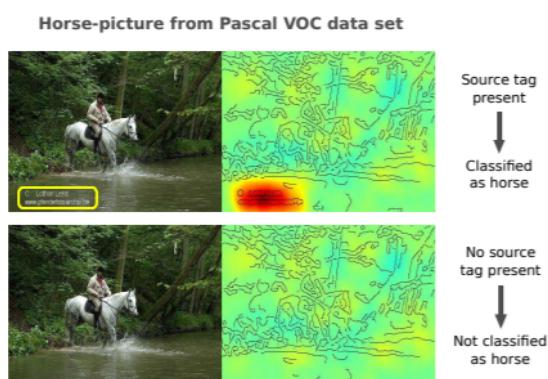
8.1 Bilderkennung: Klassifikation Hund - Katze

In den letzten Jahrzehnten wurden grosse Fortschritte in der Bilderkennung gemacht. Verantwortlich dafür sind vor allem neuronale Netze, insbesondere die Techniken Convolutional Neural Network (CNN) oder andere Varianten von Deep Neural Network (DNN). Neuronale Netze, insbesondere die für Bilderkennung weit verbreiteten DNN, sind ohne weitere Hilfsmittel kaum zu analysieren. Durch den starken Fokus dieser Techniken in der Bilderkennung sind auch viele Methoden entwickelt worden, um das Verhalten eines Modells bei der Bildanalyse darzustellen.

8.1.1 Eigenes CNN Modell

Versuch: Eingangsdaten mit BIAS

Daten, welche für das Trainieren von Machine Learning (ML) Modellen verwendet werden, können eine unbekannte Struktur enthalten. Oft ist diese für die gewünschte Vorhersage nicht relevant, verfälscht jedoch das Ergebnis. Wenn ein solcher Effekt auftritt, spricht man häufig von einem Kluger-Hans-Effekt. Ein bekanntes Beispiel dieses Effektes trat bei einem neuronalen Netz auf, das für einen Wettbewerb eingereicht wurde (Pascal VOC Everingham et al., 2010) und betraf die Erkennung von Pferden (Lapuschkin et al., 2019).



Die meisten Bilder mit Pferden in dem bereitgestellten Trainingsdatensatz für die Pascal VOC Challenge enthielten einen Quellen-Verweis. Aufgrund dessen lernte das neuronale Netz anstatt des Erkennens von Pferden, das Erkennen dieser Verweise. Da dieser Hinweis nur auf Pferdebildern vorhanden war, wurden dadurch alle Bilder mit solch einem Hinweis als "Pferd" klassifiziert.

Abbildung 8.1: Klassifizierung eines Pferdes in Pascal VOC¹

¹Quelle: Unmasking Clever Hans Predictors and Assessing What Machines Really Learn 2019

In diesem Versuch soll diese Ausgangslage nachgestellt werden. Die Annahme ist, dass ein grosser Teil der Bilder von Katzen von einem Dienstleister stammen, welcher sein Logo jeweils in der linken unteren Ecke platziert.

Daten

Die Daten stammen von einer Ausschreibung auf kaggle.com [7] und enthalten 25'000 Bilder von Katzen und Hunden. Für das trainieren des Netzes werden 20'000 Bilder und für das validieren 5'000 Bilder verwendet.

Verwendete Bibliotheken

Das Convolutional Neural Network (CNN) wurde mit Tensorflow [63] berechnet und durch tf_explain [38] visualisiert.



Um eine Verfälschung der Daten zu simulieren wurde nebenstehendes Logo in die Trainings- und Testdaten eingefügt. Insgesamt wurden 90% der Katzenbilder mit diesem Logo gekennzeichnet. Hundebilder weisen kein Logo auf. Die Platzierung des Logos ist auf der linken Seite und, je nach Auflösung des Bildes, zwischen der unteren Ecke und der Mitte des Bildes.

Abbildung 8.2: fiktives Logo



Abbildung 8.3: Ursprüngliches Bild

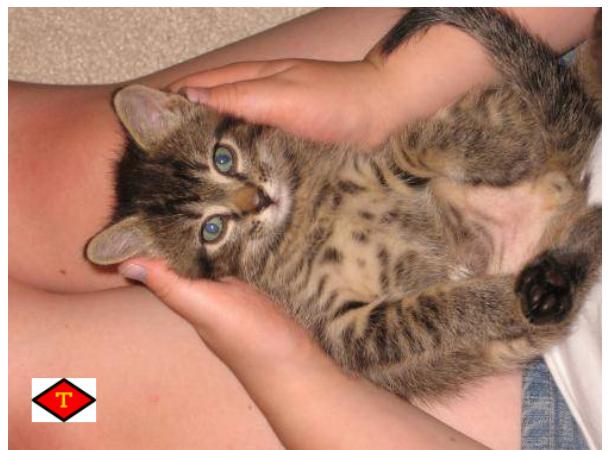


Abbildung 8.4: Manipuliertes Bild

Wenn die Hypothese korrekt ist, dann sollten Katzenbilder anhand des Logos erkannt werden. Dieser Bildbestandteil ist in den meisten Trainingsbildern für die Klasse "Katze" identisch. Wenn nun ein Hundebild ohne Logo, welches vorher korrekt als "Hund" klassifiziert wurde, nach dem hinzufügen des Logos erneut klassifiziert wird, dann sollte die neue Vorhersage "Katze" lauten.

Aufbau des neuronalen Netzes

Die Grundlage dieses Experimentes bildet ein Blog Post auf der Seite “Towards Data Science” [61]. Das erzeugte CNN ist in der ursprünglichen Version ein Binärer Klassifikator. Die für die Visualisierung gewählte Bibliothek “tf_explain” [38] kann jedoch keine binären Klassifizierer visualisieren, weshalb das Modell auf die Erkennung von zwei Klassen angepasst wurde.

```
1 model.add(Conv2D(32, 3, strides=(1, 1), padding='same', input_shape=
     input_shape, activation='relu'))
2 model.add(Conv2D(32, 3, strides=(1, 1), padding='same', activation='relu'))
3 model.add(MaxPooling2D(pool_size=(2, 2)))
4
5 model.add(Conv2D(64, 3, strides=(1, 1), padding='same', activation='relu'))
6 model.add(Conv2D(64, 3, strides=(1, 1), padding='same', activation='relu'))
7 model.add(MaxPooling2D(pool_size=(2, 2)))
8
9 model.add(Conv2D(128, 3, strides=(1, 1), padding='same', activation='relu'))
10 model.add(Conv2D(128, 3, strides=(1, 1), padding='same', activation='relu'))
11 model.add(MaxPooling2D(pool_size=(2, 2)))
12
13 model.add(Conv2D(256, 3, strides=(1, 1), padding='same', activation='relu'))
14 model.add(Conv2D(256, 3, strides=(1, 1), padding='same', activation='relu'))
15 model.add(MaxPooling2D(pool_size=(2, 2)))
16
17 model.add(Flatten())
18 model.add(Dense(256, activation='relu'))
19 model.add(Dropout(0.5))
20
21 model.add(Dense(256, activation='relu'))
22 model.add(Dropout(0.5))
23
24 model.add(Dense(2))
25 model.add(Activation('sigmoid'))
26
27 model.compile(loss='binary_crossentropy',
                 optimizer=RMSprop(lr=0.0001),
                 metrics=['accuracy'])
```

Listing 8.1: CNN für Dog vs. Cats

Training

Über 20 Epochen wurde das CNN trainiert, mit folgenden Resultaten:

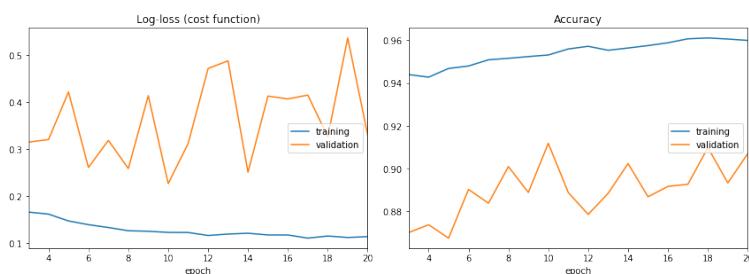


Abbildung 8.5: Log-loss / Accuracy Dog vs. Cat

Die Kurven der Erkennungs- und Fehlerraten zeigen ein untypisches Bild. Normalerweise steigt die Accuracy mit fortschreitendem Training und die Werte der Log-loss Funktion fallen. Dies deutet auf problematische Trainingsdaten mit einem Bias hin.

```

Log-loss (cost function):
training  (min: 0.110, max: 0.246, cur: 0.113)
validation (min: 0.226, max: 0.537, cur: 0.330)

Accuracy:
training  (min: 0.911, max: 0.961, cur: 0.960)
validation (min: 0.867, max: 0.912, cur: 0.907)

```

Abbildung 8.6: Bewertung des Hund-Katze Netzwerkes

Die Werte für die Erkennung der Bilder ist sehr gut, beinahe 91%. Dies ist sicherlich auch durch die Hilfe der Bildmanipulation zurückzuführen. Diesen Verdacht nachzuweisen ist das Ziel der nächsten Schritte.

Test mit originalen und manipulierten Bilddaten

Das vorher berechnete Modell wurde gespeichert [19] und die folgenden Bildanalysen wurden mit einem Jupyter Notebook durchgeführt [18], welches das gespeicherte Modell von der Festplatte lädt.

Hund ohne Logo

Das erste Testbild zeigt einen Hund, das bei den Katzenbildern hinzugefügte Logo ist nicht vorhanden.

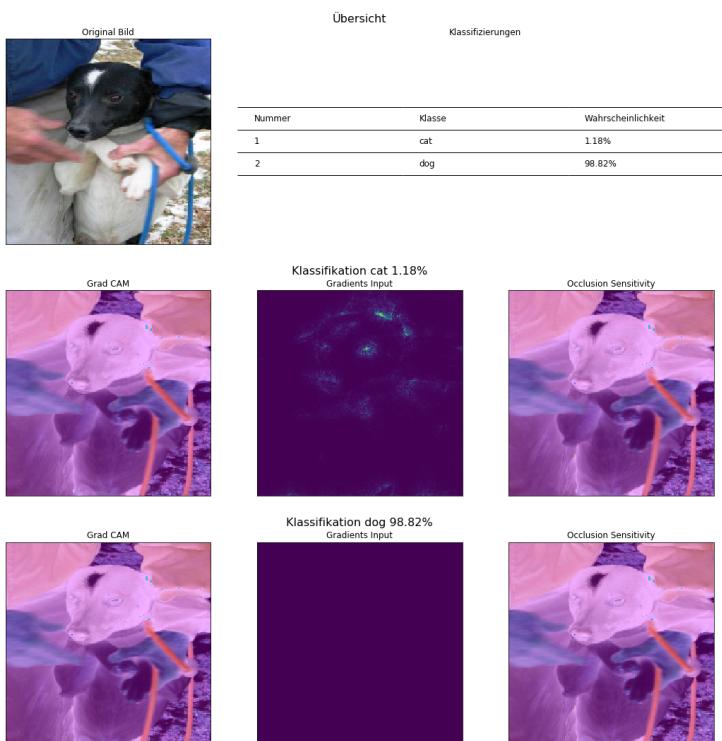


Abbildung 8.7: Testbild ohne Logo

Dieses Testbild eines Hundes wird von dem Netz eindeutig mit einer 99% Wahrscheinlichkeit als Hund klassifiziert.

Die Visualisierung mit den Methoden Grad CAM und Occlusion Sensitivity zeigen keine sichtbaren Aktivierungen. Interessant ist, dass Gradients Input im Falle der Klasse Katze ein Aktivierungsmuster anzeigt, diese Klasse aber nur mit 1.18% Wahrscheinlichkeit klassifiziert wird.

Grundsätzlich fällt auf, dass die Aktivierungsmuster alle in einem sehr tiefen Bereich liegen (violette Farbe).

Hund mit Logo

Nun wird dem vorherigen Bild das Logo in der unteren linken Ecke hinzugefügt. Da wir das CNN dazu bringen möchten dieses Logo mit der Klasse "Katze" zu verbinden, sollte nun die Wahrscheinlichkeit für diese Klasse steigen.

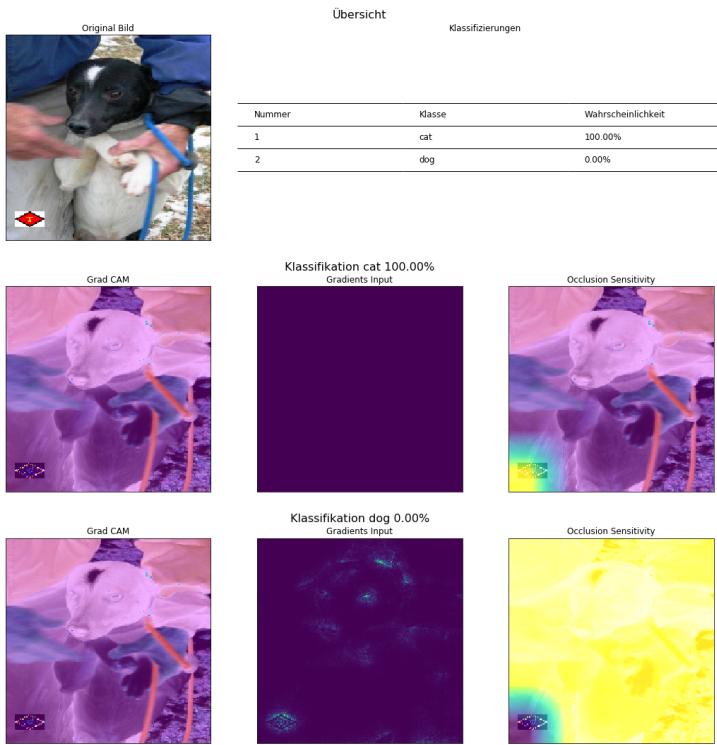


Abbildung 8.8: Testbild mit Logo

Die Klassifizierung ändert sich eindeutig auf 100% Katze. Das neuronale Netz konnte also dazu gebracht werden dem Logo die höchste Aufmerksamkeit zu widmen.

Dies kann durch die Visualisierung gezeigt werden. Während Grad CAM und Gradients Input kaum Aktivierungen aufzeigen, ist die Lage bei Occlusion Sensitivity deutlich: Für die Klasse "Katze" spricht das Vorhandensein des Logos, während für die Klasse "Hund" alle Bildbereiche außer dem Logo relevant sind.

Katze ohne Logo

Nun wird geprüft ob eine Katze ohne den Indikator des Logos korrekt klassifiziert werden kann.

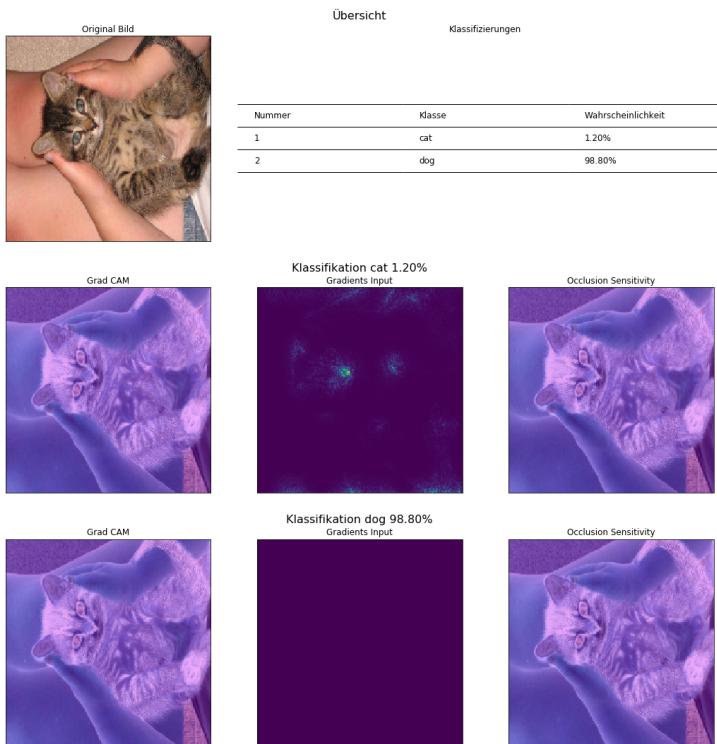


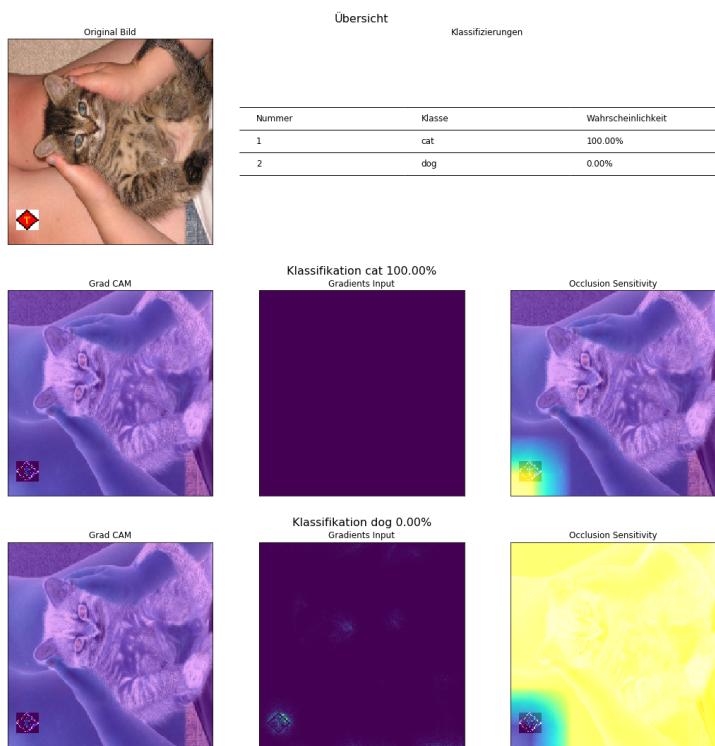
Abbildung 8.9: Testbild Katze ohne Logo

Dieses Bild einer Katze wurde zu 98.8% als "Hund" klassifiziert. Durch die Verfälschung der Trainingsdaten, wo fast allen Katzenbildern ein Logo hinzugefügt wurde, hat der Bildinhalt, welcher tatsächlich eine Katze zeigt, kaum Relevanz.

Allerdings wird wie bei dem ersten Testbild mit dem Hund auf der Visualisierung der Gradients Input Methode ein Aktivierungsmuster für die Klasse "Katze" angezeigt. Auf die Klassifizierung hat dies jedoch keinen Einfluss, das Fehlen des Logos wird stärker gewichtet.

Katze mit Logo

Nun wird dem gleichen Bild, welches vorher als "Hund" klassifiziert wurde, das Logo hinzugefügt womit es den Katzen-Bildern gleicht, die für das Training verwendet wurden.



Mit dem Logo hinzugefügt ist die Klassifizierung eindeutig: 100% Katze

Der Effekt des hinzugefügten Logos wird, wie schon bei dem Testbild mit dem Hund, in der Occlusion Sensitivity Visualisierung eindrücklich dargestellt: Gegenüber dem Bildbereich mit dem Logo wird der Rest des Bildes ignoriert.

Abbildung 8.10: Testbild Katze mit Logo

Testbilder ohne Katze oder Hund

Da das CNN nur die beiden Klassen "Katze" oder "Hund" kennt, muss jedes Bild einer der beiden Klassen zugeordnet werden. Eine Klasse "Unbekannt" oder "Weder noch" kann nicht vergeben werden. Trotzdem lassen sich durch die Visualisierung der aktivierte Pixel Rückschlüsse ziehen.

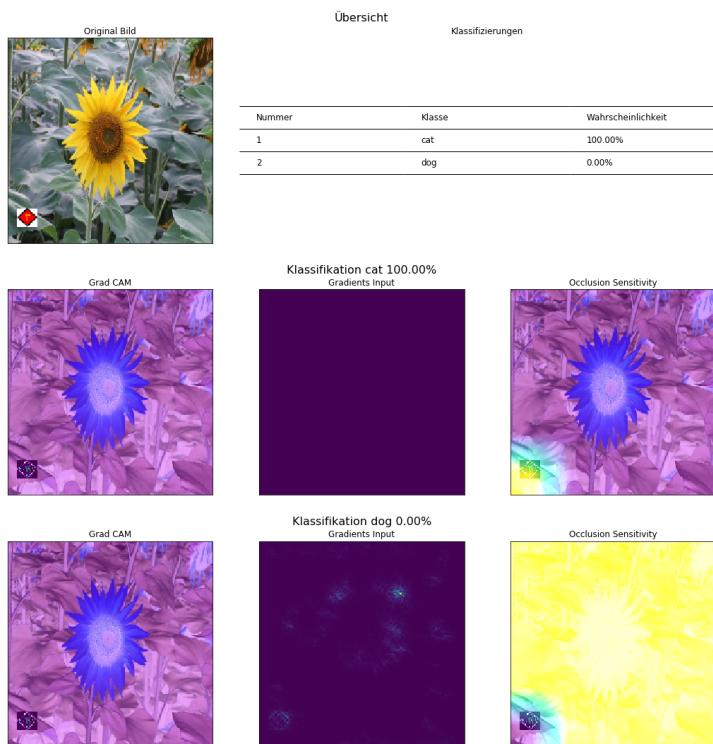


Abbildung 8.11: Testbild Sonnenblume

Obwohl auf diesem Bild keine Katze erkennbar ist wurde die Klassifizierung mit dem Ergebnis 100% Katze eindeutig getroffen. Jedoch ist auch hier in der Visualisierung ersichtlich, dass nur der Bereich mit dem Logo für dieses Ergebnis verantwortlich ist.

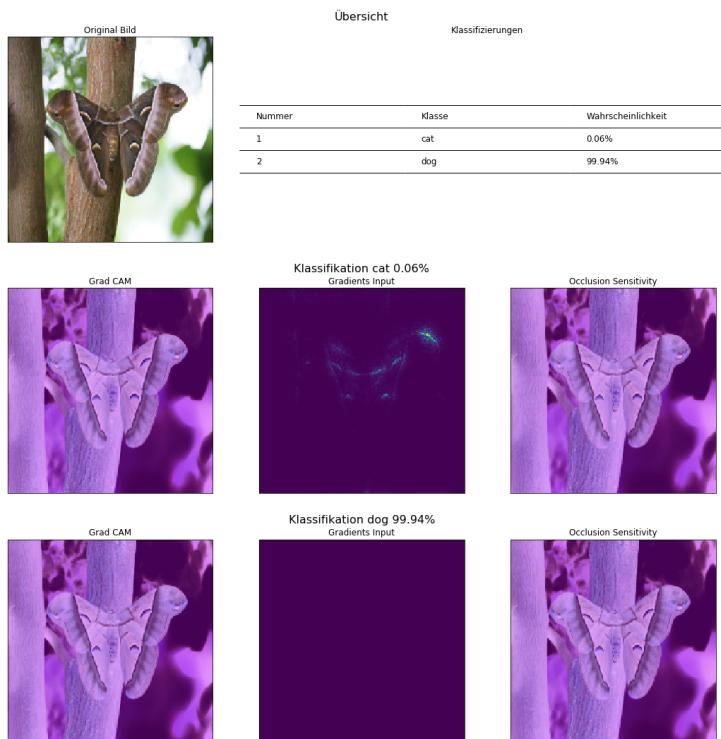


Abbildung 8.12: Testbild Falter ohne Logo

Das Bild des Falters wird in seinem unveränderten Zustand sicher mit 99.9% Wahrscheinlichkeit der Klasse Hund zugeordnet.

Auffallend ist, dass die Aktivierung wie bereits in 8.1.1 kaum stattfindet. Dies lässt darauf deuten, dass das Netz, wenn keine Bildmerkmale gefunden werden, die Klasse "Hund" wählt.

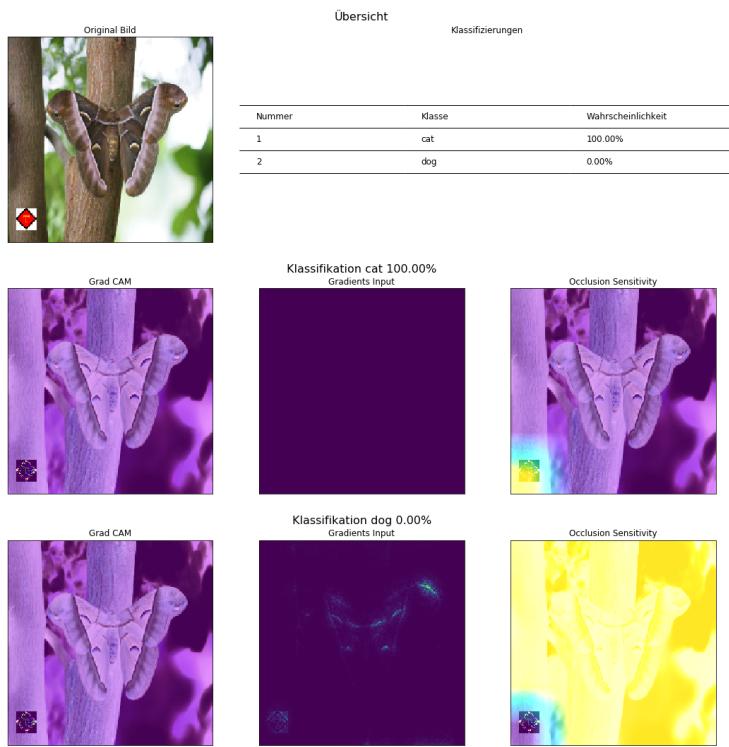


Abbildung 8.13: Testbild Falter mit Logo

Fazit dieses Versuches

Durch die Verfälschung der Trainings- und Testdaten wurde anstatt einem "Katze oder Hund" Klassifikator ein System zur Erkennung des Test-Logos erzeugt. In der Realität wären die unbrauchbaren Erkennungsraten wahrscheinlich schnell aufgefallen. Durch die Visualisierungen mit Explainable Artificial Intelligence Techniken kann der Fehler aber eindeutig dem verfälschenden Bildelement zugeordnet werden.

Eine weitere Erkenntnis ist die, dass von den drei verwendeten Visualisierungs-Methoden nur Occlusion Sensitivity die Fokussierung auf das falsche Bildelement ersichtlich macht. Man sollte sich nicht nur auf eine Technik verlassen, da es möglich ist, dass diese bei dem analysierten Modell nicht die gewünschten Resultate erzielt.

8.1.2 Vortrainiertes Modell

Das für die folgenden Analysen verwendete Modell stammt aus der ImageNet Challenge [26] aus dem Jahr 2014 und ist frei verfügbar (Simonyan & Zisserman, 2014). Dieses vorberechnete Modell definiert 1001 Klassen von Objekten, welche erkannt werden.

Verwendete Bibliotheken

Das vortrainierte Netz VGG16 [64] ist als Tensorflow [63] Klassifikator angewendet worden und die Bilder wurden durch tf_explain [38] visualisiert.

Wenn dem ursprünglichen Bild das Logo hinzugefügt wurde, dann wechselt die Vorhersage zu 100% auf die andere Klasse "Katze".

Auch hier zeigt sich ein starker Anstieg der aktivierten Pixel, ein Effekt der sich nur bei Bildern die das Logo enthalten, beobachten lässt.

Versuch: Explorative Analyse der Bildklassifikation

Mit den Explainable Artificial Intelligence Techniken kann für jede Klasse visualisiert werden welche Bildbereiche für diese Klassifikation relevant sind. In diesem Experiment wurde versucht ein Verständnis über die Funktionsweise der Klassifizierung zu finden und bei fehlerhaften Klassifizierungen eine Erklärung für das falsche Resultat zu finden.

Eine Klassifikation mit Tensorflow [63] erzeugt jeweils die komplette Liste mit den Wahrscheinlichkeiten für alle Klassen. Die wahrscheinlichsten Klassen werden jeweils rechts neben dem Bild dargestellt.

Test 1: Katzenbild

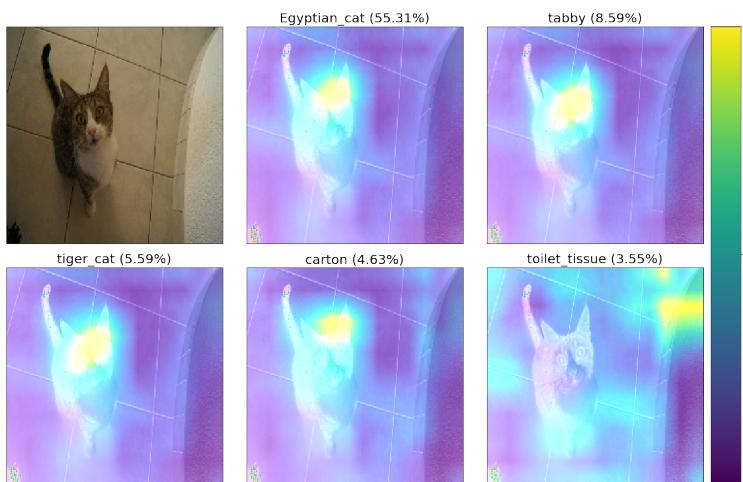
Folgendes Bild wurde analysiert:



Abbildung 8.14: Original Testbild Katze

Obwohl das vorhergehende Bild korrekt als Katze klassifiziert wurde (allerdings als die falsche Katzenrasse), fallen die 4. und 5. Klassifikation auf. Die Klassifizierung als Karton (4.6%) oder Toilettenpapier (3.6%) ist zwar nicht sehr wahrscheinlich, es stellt sich aber dennoch die Frage, weshalb keine weiteren Tiere, welche optisch grössere Ähnlichkeiten mit einer Katze aufweisen, gefunden wurden.

Mit den bereits vorgestellten Verfahren kann man nun die Einflüsse der einzelnen Bildpunkte auf die Klassifizierung sichtbar machen.



Die Analyse mittels Grad CAM zeigt für die Klasse “toilet tissue” eine hohe Aktivierung durch Pixel in der rechten oberen Ecke. Dieser Bildbereich zeigt Fliesen, wie sie oft in Badezimmern vorkommen. Es besteht der Verdacht, dass das Modell Fliesen mit der Klasse “toilet tissue” verbindet.

Abbildung 8.15: Testbild Katze visualisiert mit Grad CAM

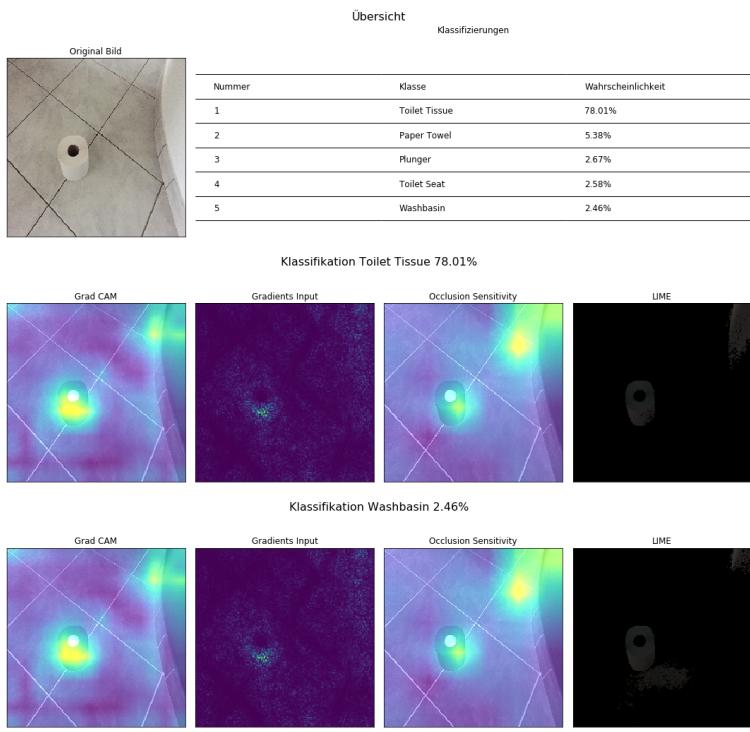


Abbildung 8.16: Testbild Toilettepapier-Rolle

Test 2: Meerschweinchen, ähnlicher Hintergrund wie bei Bild mit der Katze

Um festzustellen ob alleine der Bildhintergrund die (geringen) Wahrscheinlichkeiten für Toilettenpapier und Karton ausgelöst hat, wurde ein anderes Bild klassifiziert. Allerdings diesmal mit einem Meerschweinchen. Um den Bildinhalt so ähnlich wie möglich zu halten wurde das Bild an der gleichen Stelle aufgenommen.



Abbildung 8.17: Testbild Meerschweinchen

Ein direkter Vergleich mit einer Rolle Toilettenpapier vor dem gleichen Hintergrund zeigt auch wieder ein Aktivierungsmuster in der rechten oberen Ecke, wie bereits auf dem Bild der Katze. Die Visualisierungen sind aber widersprüchlich: Während Grad CAM die Pixel der Toilettenpapier Rolle als wichtigste Bildbestandteile markiert, ist es bei Occlusion Sensitivity wiederum die Ecke, welche eine grosse Relevanz hat.

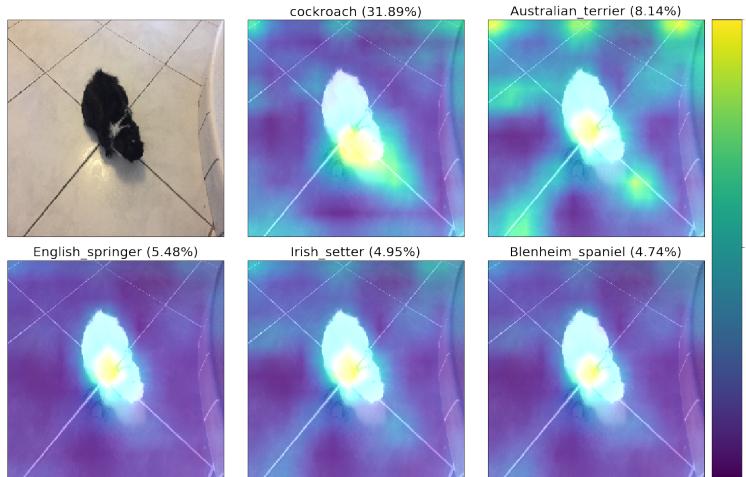
Klasse	Wahrscheinlichkeit
Cockroach	31.89%
Australian terrier	8.14%
English springer	5.48%
Irish setter	4.95%
Blenheim spaniel	4.74%
Umbrella	2.16%
Tick	1.57%
Admiral	1.53%
Weasel	1.34%
Centipede	1.31%
Sussex spaniel	1.00%
Irish water spaniel	0.99%
Papillon	0.99%
Welsh springer spaniel	0.96%
Toilet tissue	0.92%

Tabelle 8.2: Wahrscheinlichkeiten Testbild Meerschweinchen

Dieses Bild wurde falsch klassifiziert. Anstatt der korrekten Klasse "Guinea Pig" (Meerschweinchen) wurde die Klasse "Cockroach" (Kakerlake) gewählt. Allerdings ist die angegebene Wahrscheinlichkeit für "Cockroach" mit 32% nicht sonderlich hoch. Das "Toilet tissue" (Toilettenpapier), welches im vorherigen Bild immerhin mit 3.6% Wahrscheinlichkeit angegeben wurde, hat nun nur noch eine Wahrscheinlichkeit von 0.9%.

Anscheinend ist der Hintergrund in diesem Fall nicht alleine ausschlaggebend für die Klasse "Toilet tissue".

Analyse mittels Grad CAM



Die erste Visualisierung zeigt den Einfluss der einzelnen Bildpunkte für die wahrscheinlichsten fünf Klassen mit dem Verfahren Grad CAM. Der Körper des Meerschweinchens ist in allen Fällen stark Relevant, wobei der helle Fleck am Hals des Tieres den grössten Einfluss auf die Klassifizierung hat. Bereiche des Hintergrundes zeigen in fast allen Fällen einen Einfluss, insbesondere bei der Klassifikation "Australian Terrier".

Abbildung 8.18: Testbild Meerschweinchen Grad CAM

Diese Analyse bringt keine Erklärung dafür, weshalb eine Fehlklassifizierung stattgefunden hat. Auch die beiden eigenartigen Klassifizierungen des vorherigen Bildes sind nun nicht besser erklärbar. Aus diesem Grund wird nun die Analyse mit zusätzlichen Verfahren weitergeführt.

Analyse durch weitere Verfahren

Um die Analyse des CNN durch visualisierende Verfahren zu vereinheitlichen wurde ein Python Skript geschrieben [22], welches die gewählten Visualisierungs-Verfahren abbildet und zusammen mit dem Original-Bild darstellt. Zusätzlich wird noch eine Tabelle dargestellt, in welcher die fünf wahrscheinlichsten Klassen und, falls noch nicht bereits unter den ersten fünf, die korrekte Klasse für das Bild angezeigt werden.

Die Analyseverfahren sind Grad CAM, Gradients Input, Occlusion Sensitivity und LIME.

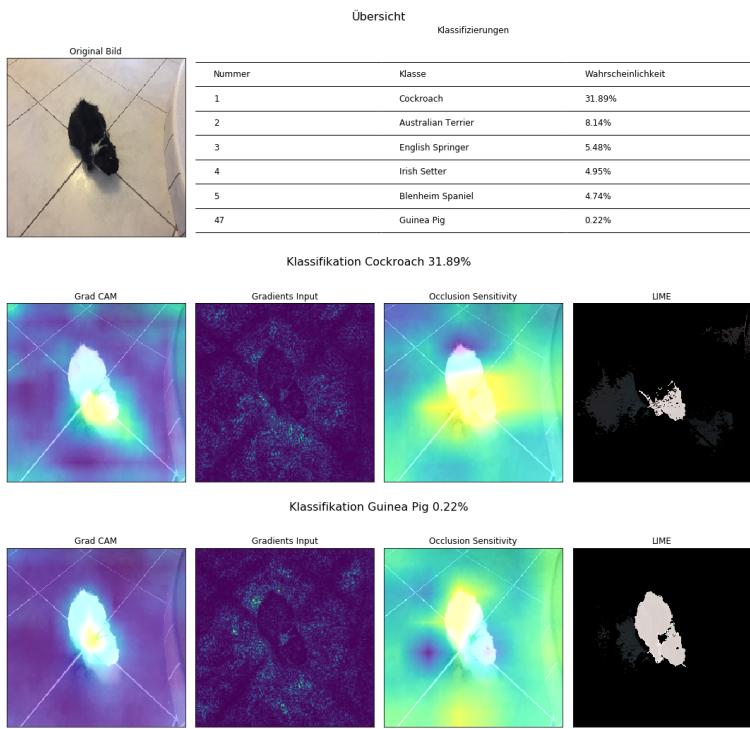


Abbildung 8.19: Testbild Meerschweinchen div. Verfahren

Die Visualisierung der eigentlich korrekten Klassifizierung "Guinea Pig" sieht auf den ersten Blick der Visualisierung von "Cockroach" ähnlich. Es erstaunt aber, dass die Methode LIME den Körper des Meerschweinchens fast vollständig hervorhebt, jedoch hat dies anscheinend keinen grossen Einfluss auf die Klassifizierung.

Fazit

Die Frage, warum auch Toilettenpapier oder eine Kakerlake gefunden wurde, konnte nicht vollständig geklärt werden. Es scheint jedoch einen Zusammenhang mit dem Untergrund der Bilder zu haben. Um die Erkennungsleistung des Modells für diese beiden Klassen zu erhöhen, sollte darauf geachtet werden, dass neue Bilder der Objekte mit einem anderen Untergrund dem Trainingsdatensatz hinzugefügt werden.

Analyse ursprüngliches Testbild durch weitere Verfahren

Die verschiedenen Klassen von Katzen ("Egyptian Cat", "Tabby" und "Tiger Cat"), welche als mögliche Klassifizierungen angeboten werden, sind auch für Menschen nicht immer klar zu bestimmen. Insbesondere da "Tabby" auch als "getigerte Katze" gelten kann. Wo ist da die Abgrenzung zu "Tiger Cat"? Die Frage stellt sich, auf welche Merkmale das Modell bei einer getigerten Katze reagiert. Ein weiteres Testbild zeigt die gleiche getigerte Katze aus dem vorherigen Abschnitt in einem anderen Winkel. Auf diesem Bild ist die Körperform und die Fellmusterung gut zu erkennen.

Die Visualisierungen für die Klassifizierung "Cockroach" zeigen kein einheitliches Muster: Während mittels Grad CAM vor allem der Körper des Meerschweinchens hervorgehoben wird, zeigt Gradients Input kaum Aktivität im Bereich des Meerschweinchens an.

Occlusion Sensitivity wiederum findet einen breiten Streifen, welcher rechts und links über den Körper des Meerschweinchens hinausgeht, als relevant. LIME markiert den Kopf des Tieres und einen schwach sichtbaren Fleck links daneben.

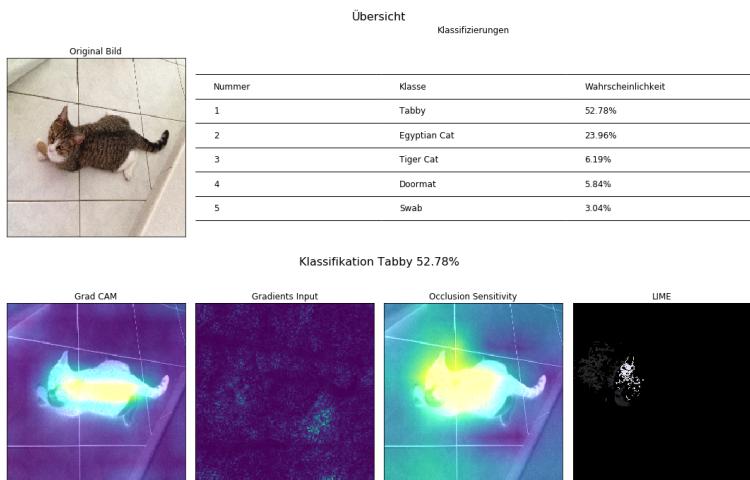


Abbildung 8.20: Testbild getigerte Katze

Ein weiteres Bild einer getigerten Katze, diesmal von vorne.

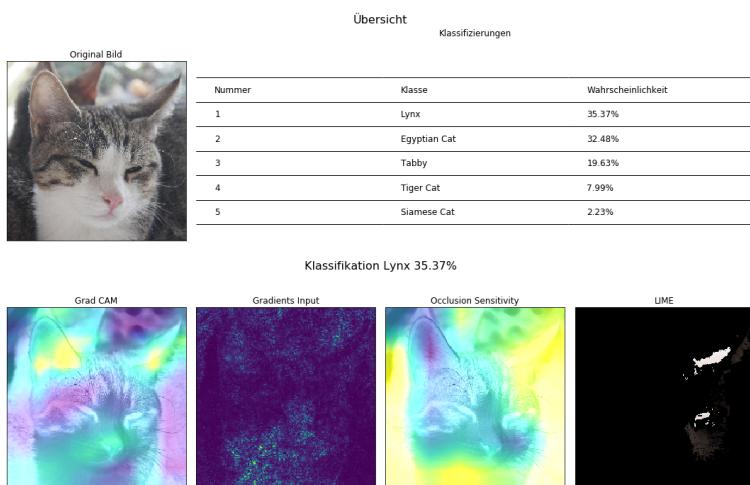


Abbildung 8.21: Testbild getigerte Katze frontal

Fazit

Ein Fazit für diese beiden Tests zu ziehen fällt schwer. Die Aussagen der Visualisierungen sind entweder wie im Fall von Gradients Input nicht zu deuten, oder wie im Fall von Grad CAM und Occlusion Sensitivity sogar widersprüchlich.

Die Visualisierung mit Grad CAM zeigt, dass das Modell sich vor allem auf das Rückgrat der Katze konzentriert. Dort bildet sich durch die Fellzeichnung auch der für getigerte Katzen typische dunkle Fellstreifen. Dieser Streifen ist in diesem Fall das relevante Merkmal des Bildes.

Die Klassifikation ist falsch. Mit 35% Wahrscheinlichkeit für einen Lynx ("Luchs") ist das Ergebnis 3% vor der nächsten Katzen Klasse. Während Grad CAM vor allem die Ohren als Merkmal kennzeichnet ist es bei Occlusion Sensitivity der Bereich um den Kopf herum, welcher für die Klasse Lynx relevant sein soll.

8.2 Texterkennung: Stimmungs-Analyse von Film-Bewertungen

Auch im Bereich der Texterkennung kann ein besseres Wissen über die Funktionsweise einer Machine Learning (ML) Anwendung sowohl den Entwicklern als auch Anwendern helfen. Insbesondere bei den Problemstellungen Sentiment Analyse und Dokumenten Klassifikation kann Explainable Artificial Intelligence das Verständnis fördern.

8.2.1 Whitebox Modell

Das folgende Beispiel visualisiert eine Stimmungs-Analyse (Sentiment Analysis) über die Kritiken von Kinofilmen. Diese Kritiken stammen von Besuchern einer Internetseite. Grundlage für das Experiment ist ein Tutorial [37], welches eine Einführung in die Text Analyse mit scikit-learn erläutert.

Daten

Die Daten stammen aus einer Arbeit von Bo Pang and Lillian Lee (Pang & Lee, 2004) aus dem Jahre 2004 und sind ein Auszug von Film Reviews der Internetplattform IMDB. Jeweils 1000 positive und negative Reviews werden mit verschiedenen Algorithmen trainiert. Der Testanteil an den Daten ist jeweils 20%.

Verwendete Bibliotheken

Das Experiment verwendet *NLTK* [41] um die Texte vorzubereiten, *scikit-learn* [55] für die Klassifikation und *ELI5* [10] um aus den Ergebnissen Erklärungen zu generieren.

Versuch 1: Decision Tree Klassifikator

Der Algorithmus Decision Tree gilt als gut interpretierbar, weshalb dies der erste Versuch ist. Der Source Code ist zu finden unter [20].

```
1 from sklearn.tree import DecisionTreeClassifier  
2  
3 classifier = DecisionTreeClassifier()  
4 classifier.fit(X_train, y_train)
```

[[136 72] [73 119]]		precision	recall	f1-score	support
0	0.65	0.65	0.65	0.65	208
1	0.62	0.62	0.62	0.62	192
accuracy				0.64	400
macro avg		0.64	0.64	0.64	400
weighted avg		0.64	0.64	0.64	400
0.6375					

Die Vorhersagequalität dieses Modells ist schlecht. Dies war aber zu erwarten, da im Allgemeinen die Qualität eines Modells entgegengesetzt zu der Interpretierbarkeit steht.

Abbildung 8.22: Konfusionsmatrix, Test Zusammenfassung und Accuracy für Decision Tree

Wenn man die Regeln des Klassifikators als Text darstellt (nur die ersten 18 Zeilen von 248 sind ersichtlich) bekommt man einen guten Eindruck davon, wie dieses Modell eine Kritik als positiv oder negativ einordnet. Allerdings ist auf Zeile 16 zu sehen, dass die Darstellung in der Breite gekürzt wurde, da der Baum unter "turns" noch 26 Ebenen tiefer gehen würde.

```
1 from sklearn.tree import export_text  
2 r = export_text(classifier, feature_names=vectorizer.get_feature_names())  
3 print(r)  
4  
5 |--- bad <= 0.04
```

```

6 |     |--- worst <= 0.05
7 |     |   |--- boring <= 0.04
8 |     |   |   |--- wasted <= 0.04
9 |     |   |   |   |--- ridiculous <= 0.03
10 |     |   |   |   |   |--- awful <= 0.04
11 |     |   |   |   |   |   |--- performances <= 0.02
12 |     |   |   |   |   |   |   |--- minutes <= 0.05
13 |     |   |   |   |   |   |   |   |--- lame <= 0.06
14 |     |   |   |   |   |   |   |   |   |--- filmmakers <= 0.06
15 |     |   |   |   |   |   |   |   |   |   |--- turns <= 0.08
16 |     |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 26
17 |     |   |   |   |   |   |   |   |   |   |   |   |--- turns > 0.08
18 |     |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 2
19 |     |   |   |   |   |   |   |   |   |   |   |   |--- filmmakers > 0.06
20 |     |   |   |   |   |   |   |   |   |   |   |   |--- making <= 0.03
21 |     |   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 2
22 |     |   |   |   |   |   |   |   |   |   |   |   |   |--- making > 0.03
23 |     |   |   |   |   |   |   |   |   |   |   |   |--- class: 1

```

Diese Darstellung gibt zwar einen Einblick in die Funktionsweise des Modells, für eine Anwendung auf einen konkreten Text ist sie jedoch zu umständlich. Der nächste Schritt verwendet Visualisierungs-Techniken aus den XAI Bibliotheken.

Versuch 2: Random Forest Klassifikator

Der komplette Source Code ist als Jupyter Notebook unter folgendem Link erhältlich: *Text Klassifizierung mit ELI5* [24]

Obwohl der Random Forest Algorithmus auch zu der Familie der Entscheidungsbäume wie Decision Tree gehört und deshalb prinzipiell zu den einfach zu interpretierenden Modellen zählt, kann eine grosse Menge an Features die Übersicht stark beeinträchtigen. Wie dieser Algorithmus sich in diesem konkreten Fall verhält soll Versuch 2 zeigen.

```

1 from sklearn.ensemble import RandomForestClassifier
2
3 classifier = RandomForestClassifier(n_estimators=1000, random_state=0)
4 classifier.fit(X_train, y_train)

```

[[183 25] [32 160]]		precision	recall	f1-score	support
0	0.85	0.88	0.87	208	
1	0.86	0.83	0.85	192	
accuracy				0.86	400
macro avg		0.86	0.86	0.86	400
weighted avg		0.86	0.86	0.86	400
0.8575					

Der trainierte Random Forest erreicht für ein Experiment eine annehmbare Fehlerquote und eine Accuracy von 0.86. Dies ist eine bedeutende Steigerung gegenüber dem vorherigen Decision Tree welcher nur auf eine Accuracy von 0.64 kam.

Abbildung 8.23: Konfusionsmatrix, Test Zusammenfassung und Accuracy für Random Forest

Visualisierung durch ELI5

Die Python Bibliothek *ELI5* [10] unterstützt einige ML Bibliotheken, unter anderem auch die hier verwendete Bibliothek scikit-learn. ELI5 bietet die Möglichkeit Schlüsselwörter, welche für eine Klassifikation relevant sind, direkt im Ursprungstext darzustellen.

Darstellungsprobleme mit ELI5

Die Darstellung des Textes mit hervorgehobenen wichtigen Features (Wörtern) für die Klassifikation war nicht in jedem Fall in Jupyter Notebook ersichtlich. Während auf einem Windows 10 Rechner keine Darstellung ersichtlich war, funktionierte die Anzeige auf einem zweiten PC mit den gleichen Versionen einwandfrei.

Während bei einem digitalen Bild die Bildpunkte als Eingangsdaten (Features) verwendet werden, sind es bei der Textanalyse die vorkommenden Wörter. Diese können sowohl positiven wie auch negativen Einfluss auf eine bestimmte Klasse haben. Eine Übersicht der Top Features zeigt die Funktion “show_weights()”.

```
1 eli5.show_weights(classifier, vec=vectorizer, top=10)
```

Weight	Feature
0.0222 ± 0.0536	bad
0.0128 ± 0.0383	worst
0.0083 ± 0.0276	boring
0.0068 ± 0.0249	stupid
0.0068 ± 0.0221	nothing
0.0066 ± 0.0217	supposed
0.0063 ± 0.0214	awful
0.0061 ± 0.0208	plot
0.0061 ± 0.0207	ridiculous
0.0060 ± 0.0198	waste
... 1490 more ...	

Bei einer binären Klassifizierung verhält sich ELI5 so, dass nur eine Klasse (in diesem Fall “neg”) dargestellt wird. Die Farbe grün stellt immer die aktuell gewählte Klasse dar.

Abbildung 8.24: Top Features Film Review Klassifizierung

Um den Text eines Datensatzes zu visualisieren verwendet man die Methode “explain_prediction()”, welche sowohl den Einfluss der Features, als auch eine Darstellung des Textes, mit den hervorgehobenen Schlüsselwörtern anzeigt. Je nach verwendetem Modell weicht die Darstellung von dem hier dargestellten Bild ab, bei gewissen Tree Algorithmen (z.Bsp. Decision Tree) wird zusätzlich noch die Baumstruktur angezeigt.

Positive Klassifikation - Eintrag 413

Der erste Datensatz ist eine positive Kritik über den Film “The Perfect Storm”.

```
1 doc = documents[413]
2 eli5.explain_prediction(classifier, doc, vec=vectorizer, target_names=['neg', 'pos'], top=20)
```

y=pos (probability 0.661) top features

Contribution?	Feature
+0.505	<BIAS>
+0.008	plot
+0.007	worst
... 872 more positive ...	
... 575 more negative ...	
-0.066	Highlighted in text (sum)

susan granger review of the **perfect** storm warner bros more people die on fishing boats per capita than working in any other **job** in the s **every** journey fishing boat makes can be an all or **nothing** risk it is **life** at its most exhilarating and its most terrifying says director wolfgang petersen das boot and that just what he captures in this **true** story of struggle and humanity aboard swordfishing boat the andrea gail sailing out of gloucester massachusetts in late october early in **bill** wittliff screenplay based on sebastian junger **best** seller we meet the crew of six the veteran captain **george** clooney is frustrated because he can find fish on the grand banks yet rival skipper mary elizabeth mastrantonio **brings** in huge hauls his right hand man mark walhberg needs money to build new **life** with his girl friend diane lane there a devoted dad john reilly with an estranged wife and son free spirited jamaican allen payne lonely guy john hawkes and last **minute** replacement with **bad** attitude william fichtner the skipper convinced he can change his **bad** luck streak in remote flemish cap and he does but then trouble begins there a rogue wave man overboard and the ice machine breaks with lb of fish that **could** spoil but that minor compared with deadly monster storm approaching which boston meteorologist describes as disaster of **epic** proportions that **also** threatens the lives of coast guard helicopter rescue team **trying** to save three people stranded on sailboat on the high seas it formulaic and there are cliches but the walls of water created by fluid dynamics simulating real **life** phenomena are awesome on the granger movie gauge of to the **perfect** storm is terrifying suspenseful hang on for the white knuckle thrill **ride** of the summer

Abbildung 8.25: Visualisierung positives Film Review mit ELI5

Mit 66% Wahrscheinlichkeit für die Klasse “pos” ist die Einschätzung für diese Klasse eher tief. Der grösste Anteil an der Klassifizierung haben Features, welche ELI5 nicht als Wörter im Text gefunden hat, deshalb ist der Eintrag “<BIAS>” an erster Stelle der Feature Tabelle. Auf den ersten Blick stechen die negativen Einflüsse wie “bad” und “nothing” hervor. Insgesamt überwiegen jedoch die positiv eingeschätzten Wörter in der Summe.

Negative Klassifikation - Eintrag 414

Der zweite Beitrag stammt aus dem Datensatz mit den als negativ eingeschätzten Kritiken.

```
1 doc = documents[414]
2 eli5.explain_prediction(classifier, doc, vec=vectorizer, target_names=['neg', 'pos'], top=20)
```

Dieser Text wurde mit 83% klar als “neg” klassifiziert.

y=neg (probability 0.828) top features

Contribution?	Feature
+0.495	<BIAS>
+0.267	Highlighted in text (sum)
... 854 more positive ...	
... 588 more negative ...	
-0.007	worst
-0.007	plot
-0.019	bad

its **stupid** little movie that trys to be clever and sophisticated yet trys bit too **hard** with the voices of woody allen gene hackman jennifer lopez sylvester stallone and sharon stone this computer animated yak fest think toy story **filled** with used merchandising is one for the ant eaters the main story is the independence of worker named allen he wants more to **life** than just digging away underground for the colony when he finds out about insectopia mythical place where all insects can run free along with his colony princess stone journey out into the **world** to find meaning for **life** about **minutes** into the picture began to **wonder** what the **point** of the film was halfway through still didn have an answer by the end **credits** just gave up and ran out antz is mindless **mess** of **poor** writing and **even** poorer voice overs allen is nonchalant while would have guessed if hadn **seen** her in the mighty and basic instinct stone can act **even** in cartoon this film is one for the bugs **unfunny** and extremely **dull** hey bug life may have good time doing antz in

Abbildung 8.26: Visualisierung negatives Film Review mit ELI5

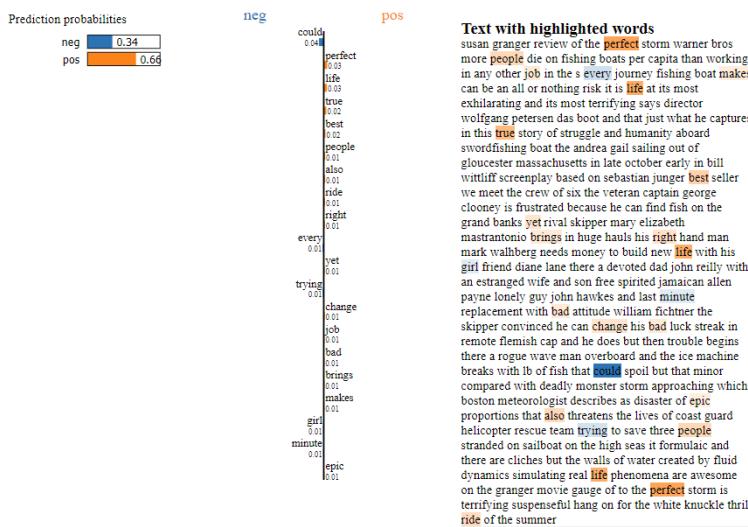
Bei der Darstellung des negativen Film Reviews fällt auf, dass Wörter, welche für eine negative Stimmung stehen, grün markiert sind. Dies kommt daher, dass für ELI5 die wahrscheinlichste Klasse “neg” ist, mit 81% Wahrscheinlichkeit. Deshalb sind alle Schlüsselwörter, welche auf diese Klasse hinweisen, grün markiert. In diesem Fall konnte die erzeugte Erklärung immer noch den grössten Anteil nicht visualisieren (“<BIAS>” mit 0.495 Prozentpunkten) aber die hervorgehobenen Wörter zeigen deutlich die überwiegend negative Stimmung.

Visualisierung durch LIME

Eine zweite Bibliothek um visualisierte Erklärungen zu generieren ist *Lime: Explaining the predictions of any machine learning classifier [Lime]*[51]. Wie der Name andeutet, verwendet diese Bibliothek die Technik LIME um aus einem beliebigen Modell Erklärungen zu erzeugen. Der Aufbau des Versuches ist gleich wie vorher mit ELI5, die Daten und das Text Pre-Processing sind identisch. Der Source Code kann hier [25] heruntergeladen werden.

Positive Klassifikation - Eintrag 413

Da das gleiche Modell verwendet wird, ist die Wahrscheinlichkeit für die Klasse "pos" immer noch 66%.

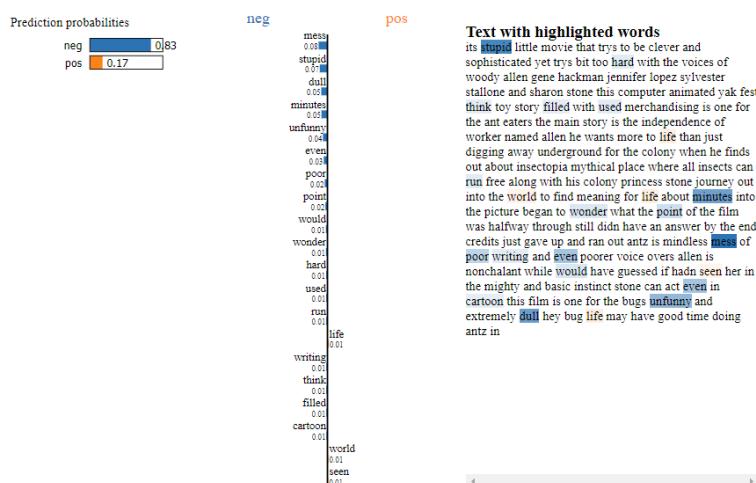


Die meisten der hervorgehobenen Wörter entsprechen der Visualisierung mit ELI5, eine Ausnahme bildet das Wort "nothing", welches durch ELI5 als starker Einfluss für die Klassifikation "neg" identifiziert wurde. Auch fällt auf, dass das Wort "bad" durch ELI5 stark negativ gewichtet wurde, während mit LIME das Wort sogar leicht positiv gewichtet wird.

Abbildung 8.27: Visualisierung positives Film Review mit LIME

Negative Klassifikation - Eintrag 414

Ebenso wie die positive Kritik wird nun auch die negative Kritik mit dem selben Modell visualisiert.



Die negative Kritik entspricht in der Visualisierung der Darstellung mit ELI5. Im Gegensatz zu der positiven Kritik fallen hier keine Unterschiede in der Gewichtung auf.

Abbildung 8.28: Visualisierung negatives Film Review mit LIME

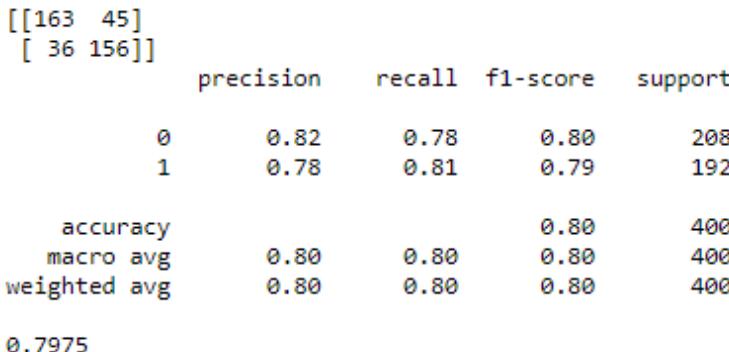
Fazit Vergleich ELI5 und LIME

Wie sich gezeigt hat, können kleine Unterschiede zwischen den Erklärungen verschiedener Techniken auftreten. Dies ist nicht weiter verwunderlich, da LIME mit generierten Modellen arbeitet, welche eine Annäherung an das zu beobachtende Modell darstellen. Grundsätzlich sind die Erklärungen aber vergleichbar.

Versuch 3: Naïve Bayes Klassifikator

Als letzter Versuch wird noch ein Klassifikator mit dem Naïve Bayes Algorithmus, welcher bereits in 6.1.6 beschrieben wurde untersucht.

```
1 from sklearn.naive_bayes import MultinomialNB  
2 classifierNB = MultinomialNB().fit(X_train, y_train)
```



Dieser Klassifikator hat eine etwas geringere Accuracy als der Random Forest Klassifikator mit einer Accuracy von 0.8. Diese ist aber immer noch weit besser als mit dem einfachen Decision Tree Algorithmus.

Abbildung 8.29: Konfusionsmatrix, Test Zusammenfassung und Accuracy für Naïve Bayes

Wie erwartet, unterscheidet sich die Darstellung etwas von den vorherigen. Da aber ein anderes Modell verwendet wurde ist dies nicht erstaunlich.

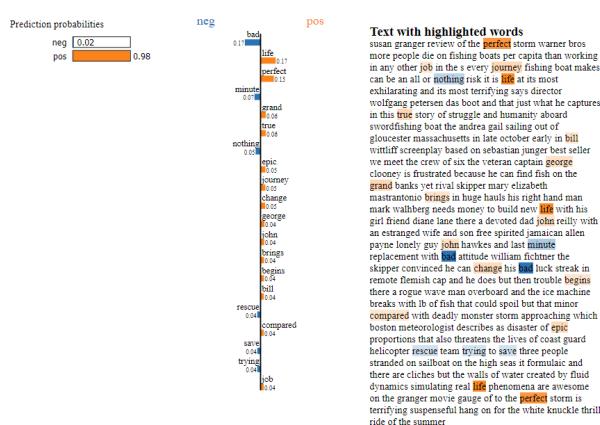


Abbildung 8.30: Visualisierung positives Film Review mit LIME und NB

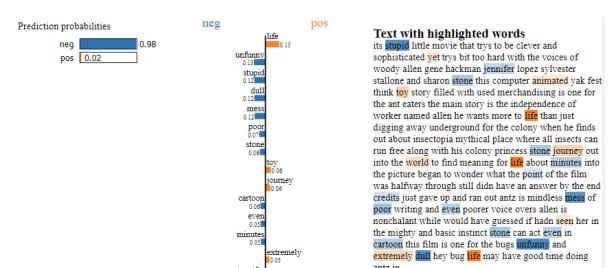


Abbildung 8.31: Visualisierung negatives Film Review mit LIMF und NB

Fazit

Auch unterschiedliche Klassifikations-Algorithmen können einfach dargestellt und verglichen werden. Dadurch können Auffälligkeiten leicht entdeckt werden und Anpassungen (andere Algorithmen, geändertes Pre-Processing usw.) getestet werden.

8.2.2 Blackbox Modell

Während in dem vorherigen Beispiel ein Whitebox Modell mit interpretierbaren Algorithmen angewendet wurde, können auch Blackbox Modelle analysiert werden. Dafür benutzen sowohl *ELI5* [10] als auch *Lime* [51] das LIME Verfahren um die Vorhersage zu erklären und das Modell zu interpretieren.

Daten

Der selbe Datensatz wie in 8.2.1 wurde für einen weiteren Klassifikator genutzt, dieses Mal allerdings mit dem Long short-term memory (LSTM) Verfahren, welches zu den neuronalen Netzen gehört.

Verwendete Bibliotheken

Die Grundlage des Skriptes bildet der Blog-Beitrag *LIME of words: interpreting Recurrent Neural Networks predictions* [65] welcher für die geänderte Datenquelle angepasst wurde. Das Experiment verwendet *NLTK* [41] um die Texte vorzubereiten, scikit-learn *scikit-learn* [55] und Tensorflow [63] für die Klassifikation und *ELI5* [10] um aus den Ergebnissen Erklärungen zu generieren.

Visualisierung eines LSTM mit LIME

Das Jupyter Notebook mit dem gesamten Source Code ist auf github.com vorhanden *Text Klassifizierung eines LSTM Modelles mit LIME* [21].

Das neuronale Netz ist sehr einfach gehalten um die Trainingszeit zu reduzieren. Es handelt sich hier um einen Klassifikator des Typs Binärer Klassifikator wie man an dem letzten Layer mit nur einem Ausgang sieht.

```
1 model = Sequential()
2 model.add(Embedding(max_features, 128))
3 model.add(Bidirectional(LSTM(128, dropout=0.5, recurrent_dropout=0.5)))
4 model.add(Dense(1, activation='sigmoid'))
5 model.compile('adam', 'binary_crossentropy', metrics=['accuracy'])
```

Listing 8.2: LSTM Modell für LIME Movie Sentiment Analyse

In acht Durchgängen wird das Netz trainiert. Als Wrapper für Tensorflow dient hier Keras, die Vorbereitung der Daten übernimmt eine scikit-learn Pipeline.

```
1 sklearn_lstm = KerasClassifier(build_fn=create_model, epochs=8, batch_size=batch_size,
                                max_features=max_features, verbose=1)
2
3 # Build the Scikit-learn pipeline
4 pipeline = make_pipeline(sequencer, padder, sklearn_lstm)
5
6 pipeline.fit(texts_train, y_train)
```

Model: "sequential_1"		
Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, None, 128)	2560128
bidirectional_1 (Bidirection)	(None, 256)	263168
dense_1 (Dense)	(None, 1)	257

Total params: 2,823,553
Trainable params: 2,823,553
Non-trainable params: 0

Abbildung 8.32: Zusammenfassung LSTM Netz für Movie Sentiment Analyse

	[[120 88]	[22 170]]	precision	recall	f1-score	support
0	0.85	0.58	0.69	208		
1	0.66	0.89	0.76	192		
accuracy			0.73	400		
macro avg		0.75	0.73	0.72	400	
weighted avg		0.76	0.72	0.72	400	
	0.725					

Abbildung 8.33: Performance des LSTM Netz für Movie Sentiment Analyse

Die Qualität der Klassifizierungen liegt tiefer als bei den vorherigen Beispielen mit dem Random Forest und dem Naïve Bayes Klassifikator. Dies ist bei einem solch einfach aufgebautem neuronalem Netz nicht erstaunlich.

Positive Klassifikation - Eintrag 413

Erneut wird der bereits vorher in 8.2.1 positiv klassifizierte Beitrag mit der Id 413 analysiert.

```
1 Probability(positive) = 0.99591535
2 True class: positive
```

Das Resultat fällt für dieses Modell eindeutig aus: 99% Wahrscheinlichkeit für die Klasse “positiv”.

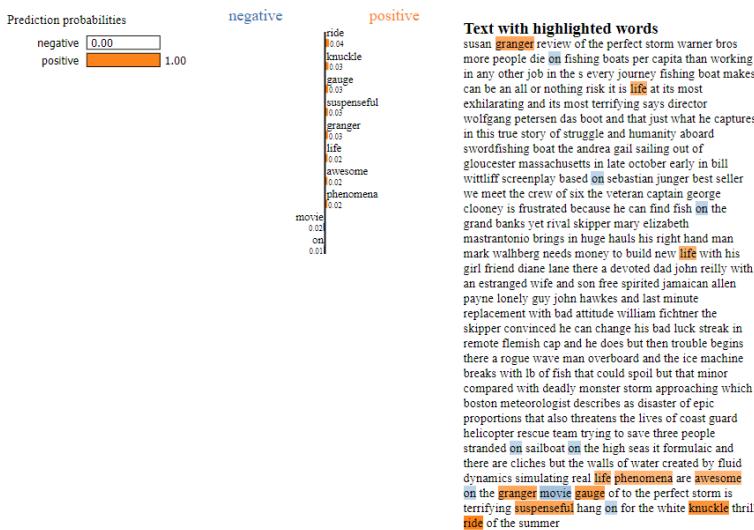


Abbildung 8.34: Visualisierung Movie Review 413

Obwohl die Klassifizierung auch diesmal positiv ist, werden völlig andere Wörter für dieses Resultat verantwortlich gemacht. Auffallend ist, dass das Wort “bad” für das Modell in diesem Text keine Relevanz hat. Auch die restliche Liste der relevanten Wörter ist fast vollständig verschieden von allen anderen Modellen die bereits untersucht wurden.

Negative Klassifikation - Eintrag 414

Auch dieser Text wird korrekt als “negativ” klassifiziert, auch hier mit einer hohen Wahrscheinlichkeit von 96%. Auch der Eintrag 414 wird wie bereits in 8.2.1 analysiert.

```
1 Probability(positive) = 0.037772316
2 True class: negative
```

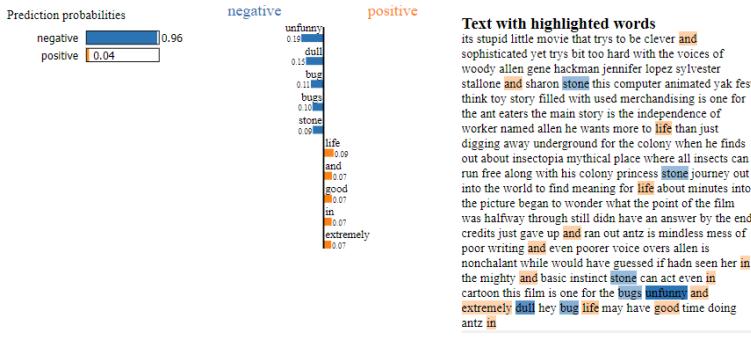


Abbildung 8.35: Visualisierung Movie Review 414

Das Wort “unfunny” wurde wie auch in der ersten Analyse als stark relevant für die Klassifikation befunden, während diesmal “stupid” keine Beachtung findet. Ein Fehler in der Vorbereitung der Texte wird durch diese Darstellung deutlich sichtbar: Die Filterung der Stopp-Wörter scheint nicht korrekt zu funktionieren, die beiden Wörter “and” und “in” sollten nicht in die Klassifizierung mit einbezogen werden.

Fazit

Das Erklären der Entscheidung von neuronalen Netzen in der Sentiment Analyse funktioniert gut. Es fällt auf wie unterschiedlich die Gewichtung der Wörter zwischen den Whitebox Modellen und diesem Blackbox Modell ist. Da das neuronale Netz aber eine tiefere Trefferquote hat und diese wahrscheinlich nach dem Entfernen weiterer Stopp Wörter wie “and” und “in” ansteigen würde, kann sich diese Gewichtung bei einem erneuten Training angleichen.

9 Schwächen von Modellen erkennen

Die in den vorherigen Kapiteln vorgestellten Techniken erlauben einen besseren Einblick in die Funktionsweise von Machine Learning (ML) Modellen. Dieses Wissen kann dazu genutzt werden um Schwächen oder auch gezielte Angriffe auf Anwendungen mit integrierten ML Modellen zu erkennen.

9.1 Diskriminierung durch Bias

Als Bias werden Tendenzen bezeichnet, welche in Modellen vorhanden sind und durch nicht ausgewogene Trainingsdaten erzeugt werden. Ein bekanntes Beispiel dazu ist ein Chat-Bot, welcher von Microsoft entwickelt wurde und der sein Verhalten durch Twitter Tweets trainierte. Das Experiment musste von Microsoft nach kurzer Zeit abgebrochen werden, da der Chat-Bot eine Tendenz zu rassistischen und beleidigenden Antworten hatte. Verursacht wurde dieses Verhalten durch gezielte Manipulationen durch andere Twitter Benutzer, die eine Gelegenheit sahen den Chat-Bot auszutricksen. Eine Beschreibung der Probleme dieses Projektes wurde in dem Artikel *Learning from Tay's introduction* [36] dokumentiert. Dieser Fall kann auch als Beispiel für "Data Poisoning" angesehen werden, da die Lerndaten des Bots gezielt beeinflusst wurden, siehe 9.3.

9.2 Manipulierte Bilder: Adversarial Attacks

Wie bereits in Kapitel 4.3 angedeutet, sind ML Anwendungen oftmals unerwartet empfindlich auf kleinste Änderungen an den Eingangsdaten. In einer Arbeit von Google Forschern (I. J. Goodfellow et al., 2014) konnte gezeigt werden, dass dies eine grundsätzliche Eigenschaft nicht nur von neuronalen Netzen, sondern auch anderen linearen Klassifikatoren ist. Erklärbar ist ein solches Fehlverhalten wenn man sich vor Augen hält, dass neuronale Netze und andere Klassifikatoren keine abstrakten Konzepte wie "Tier" oder auch "Fahrzeug" lernen, sondern sich alleine an den für das Training verwendeten Bilddaten orientieren.

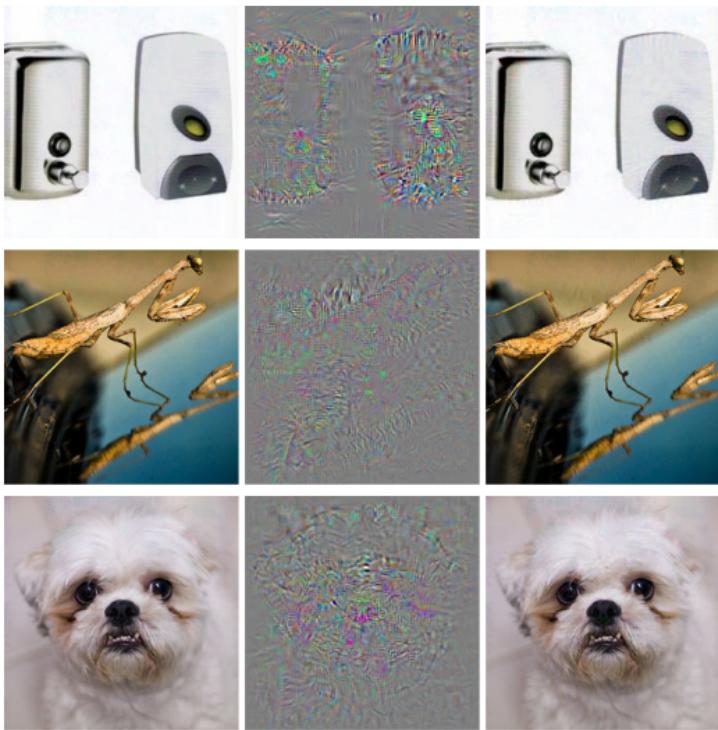


Abbildung 9.1: Adversarial Beispiel¹

Ein Beispiel für die Manipulation eines Bildklassifikators stammt aus der Arbeit “Intriguing properties of neural networks” (Szegedy et al., 2013).

Die linke Spalte zeigt ein korrekt klassifiziertes Bild.

Die mittlere Spalte stellt den Unterschied zwischen dem korrekt klassifizierten Bild und der falschen Klasse dar.

Rechts ist das mit der Adversarial Technik manipulierte Bild zu sehen.

Alle Bilder in der Rechten Spalte wurden als “Ostrich” (Strauss) klassifiziert!

Adversarial Attacken selber erzeugen

Die Tensorflow Webseite stellt ein Tutorial zu Verfügung, das die Erzeugung solcher “Adversarial” Bilder erläutert: *Adversarial example using FGSM* [67]

Eines der in dem Tutorial erzeugten Bilder wird schliesslich als “brain coral” (Hirnkoralle) erkannt, allerdings mit deutlich sichtbaren Bild-Artefakten. Für das menschliche Auge sieht es jedoch aus als ob das Bild stark verrauscht wäre und nicht nach einem gezielt manipulierten Bild.

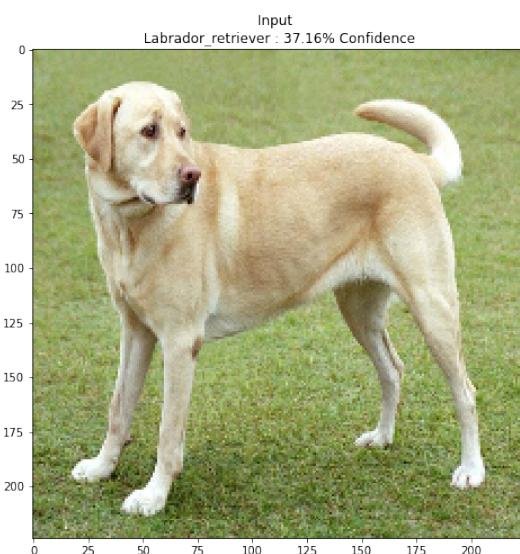


Abbildung 9.2: Ursprungs-Bild²

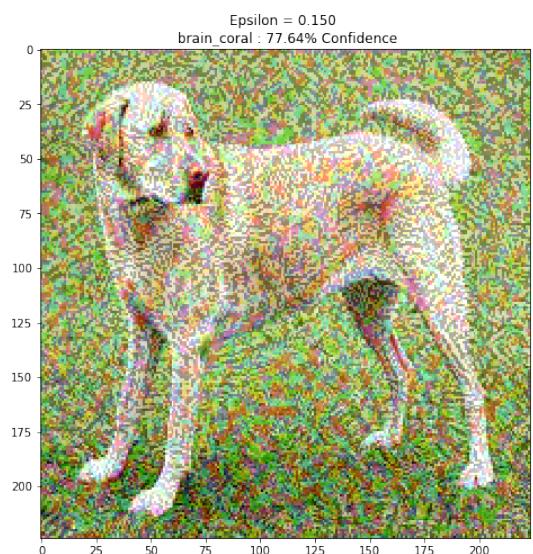


Abbildung 9.3: Durch Adversarial Angriff erzeugtes Bild

¹Quelle: Szegedy (2013)

Täuschung der Verkehrszeichen-Erkennung eines Tesla

Ein interessantes Beispiel einer solchen “Adversarial Attack” ist die Arbeit eines Teams von McAfee Labs, welches gezielt das System eines Tesla angegriffen hat. Schlussendlich wurde das Auto dazu gebracht anstatt 35mph (ca. 55kmh) 85mph (ca. 135kmh) als Geschwindigkeitsbegrenzung zu erkennen.

Model Hacking ADAS to Pave Safer Roads for Autonomous Vehicles [60]



Eine Verlängerung des mittleren Balkens einer Drei führte das neuronale Netz eines Tesla zu einer Fehlklassifizierung als Acht und dadurch zu einer erlaubten Geschwindigkeit von 85 Meilen pro Stunde. Obwohl die veränderte Ziffer durchaus an eine Acht erinnern kann, sehen Menschen die Ziffer trotzdem als Drei, im Gegensatz zu den Systemen des Teslas.

Mittlerweile wurde das Verhalten der Tesla-Systeme angepasst und diese spezifische Attacke ist nun nicht mehr möglich.

Abbildung 9.4: Gefälschtes Verkehrsschild als Adversarial Attack³

Eigene Versuche mit Adversarial Attacks

Durch die Arbeit einiger Forscher (Papernot et al., 2018) gibt es eine Python Bibliothek *CleverHans a Python library to benchmark machine learning systems' vulnerability to adversarial examples* [14], welche die Erzeugung von Adversarial Angriffen erleichtert.

9.3 Manipulierte Daten: Data Poisoning

Mit “Data Poisoning” werden Techniken benannt, die ML Modelle über die Trainingsdaten angreifen. Oftmals werden Modelle nach einem initialen Training mit einem festen Datensatz wiederkehrend mit aktualisierten Daten erneut trainiert. Ein typisches Beispiel sind Spam Filter die in regelmässigen Intervallen Emails, welche durch die Benutzer als Spam markiert wurden, in ihren Trainingsdatensatz integrieren.

Grundsätzlich wird zwischen zwei Arten von Data Poisoning unterschieden:

²Quelle: *Adversarial example using FGSM* [67]

³Quelle: *Model Hacking ADAS to Pave Safer Roads for Autonomous Vehicles* [60]

Availability

Das Ziel einer solchen Attacke ist es die Datenbasis eines ML Modells so zu erweitern, dass die Wahrscheinlichkeit für eine bestimmte Klassifikation entweder stark erhöht oder verringert wird. Wie durch die Arbeit “Online Data Poisoning Attacks” (Zhang et al., 2019) nachgewiesen wurde, kann ein solcher Angriff sowohl für überwachtes Lernen wie auch unüberwachtes Lernen erfolgreich eingesetzt werden.

Backdoor Attacks

Ein Backdoor Angriff versucht die Eingangsparameter eines ML Modells so zu wählen, dass ein bestimmtes Ergebnis garantiert ist. Damit kann ein schädliches Programm, welches eigentlich durch einen VirensScanner abgewehrt werden sollte, durch Verwendung einer bestimmten Zeichenkette ungehindert aktiviert werden.

Es besteht zudem die Möglichkeit, dass ein Modell, welches aus einer externen Quelle bezogen wurde, mit einer “Backdoor” ausgestattet ist um auf bestimmte Parameter zu reagieren. In der Arbeit “BadNets: Evaluating Backdooring Attacks on Deep Neural Networks” (Gu et al., 2019) konnte gezeigt werden wie ein neuronales Netz durch bestimmte Aufkleber dazu gebracht werden konnte Stopp-Schilder mit Geschwindigkeitsbegrenzungs-Schildern zu verwechseln.

10 Fazit

Zu Beginn dieser Arbeit stand ich vor einem unerwarteten Problem. Die Menge an Methoden, Artikeln und Forschungsarbeiten zu diesem Thema war grösser als erwartet. Die Entscheidungen welche Methoden besprochen, oder zumindest erwähnt, werden sollten, war schwierig. Schlussendlich war der experimentelle Teil ausschlaggebend. Einige der Methoden wie SVCCA oder TCAV stellten sich in der Anwendung als sehr aufwändig heraus, so dass bei der Wahl eines dieser Themen als praktisches Experiment viele andere Bereiche nicht beachtet werden können.

Im praktischen Teil der Arbeit hatte ich mit anderen Schwierigkeiten zu kämpfen. Einige XAI Frameworks für Python stellen sehr spezifische Anforderungen an die Version für die dem Modell zugrunde liegenden ML Frameworks. Dies ist nicht weiter erstaunlich wenn man bedenkt, dass die Analyse eines Modells tiefer auf die einzelnen Funktionen und Parameter eingehen muss als der nomalerweise verwendete Aufruf von "predict(..)". Da aber mehrere Bibliotheken kombiniert wurden, kam es häufig vor, dass die benötigten Versionen untereinander nicht kompatibel waren. Das fragile Gleichgewicht der installierten Framework Versionen wurde immer wieder durch die Installation eines weiteren Werkzeuges zunichte gemacht welches eine andere Version von Tensorflow oder Keras benötigte.

Eine Erkenntnis ist daher immer mit einer eigenen Umgebung für jedes Projekt zu arbeiten. Diese Umgebung sollte die passende Python mit den benötigten Bibliotheken in der passenden Version enthalten und nicht automatisch aktualisiert werden. Die Applikation Anaconda hilft dabei sehr diese Umgebungen zu verwalten. Da aber nicht alle getesteten Bibliotheken in Anaconda vorhanden sind, habe ich leider anfangs fehlende Bestandteile über den Python Packetmanager pip installiert. Diese Vorgehensweise war schlecht und hat mir mehrere Male die Umgebungen mit nicht miteinander kompatiblen Bibliotheksversionen "verseucht". Eine saubere Vorgehensweise mit voneinander getrennten Umgebungen ist unbedingt nötig, wenn man möchte, dass Programme auch nach einigen Wochen noch wie gewünscht funktionieren.

Die Auswertung der Ergebnisse durch exploratives Vorgehen war zu Beginn nicht erfolgreich. Es ist zwar einfach durch Visualisierungen zu erkennen welche Bestandteile eines Bildes für ein neuronales Netz wichtig sind, Schlussfolgerungen daraus zu ziehen, stellte sich jedoch als überraschend schwierig heraus. Bei einem direkten Vergleich zwischen verschiedenen Methoden waren oftmals grosse Unterschiede erkennbar was die wichtigen Bestandteile eines Bildes betraf. Da einige Methoden unter gewissen Umständen nicht funktionieren (Gradient basierte Verfahren haben nachgewiesene Schwächen) steht immer die Frage im Raum ob die Unterschiede zwischen den Visualisierungen von Bedeutung sind oder eher durch eine ungeeignete Methode der Visualisierung verursacht wurden.

Der zweite Ansatz welcher mit einem konkreten Ziel, nämlich dem Nachweis eines Kluger-Hans Effektes, gestartet war, funktionierte bedeutend besser. Die Visualisierung zeigte sehr deutlich welches Problem das neuronale Netz hat und könnte in der Realität auch direkt zu einer Verbesserung (durch Anpassung der Trainings- und Testdaten) des Modells führen. Aber auch hier war nur eine von drei eingesetzten Methoden erfolgreich.

Dies führt zu einer weiteren Erkenntnis: Jede Analyse sollte mit mehreren Methoden durchgeführt werden.

Die Ansprüche an die Erklärbarkeit von ML Modellen steigen und die Erwartungen sind gross. Im Gegensatz dazu sind die momentan verfügbaren Werkzeuge oft nur für einfache Fragestellungen einsetzbar. Trotzdem konnten durch diese Techniken bereits in einigen Modellen Fehler und Schwachstellen gefunden und beseitigt werden.

Für Aufgabengebiete welche strikte Bedingungen für das Zustandekommen einer Entscheidung vorschreiben, beziehungsweise deren Begründung einfordern, muss eine Entscheidung getroffen werden. Es kann ein Modell mit einer verständlichen Technik erstellt oder ein komplexeres, unverständliches Modell, aber mit einer vielleicht besseren Leistung gewählt werden. XAI kann dazu beitragen das Verständnis von komplexeren Modellen zu erhöhen. Ob dies im konkreten Fall ausreicht um die Anforderungen zu erfüllen, muss im Einzelfall geprüft werden. Es ist nicht möglich dazu allgemein gültige Aussagen zu treffen. Für mich ist unsicher ob XAI überhaupt jemals in der Lage sein wird allgemeine Aussagen über die Qualität und Sicherheit von Modellen zu erstellen. Es gibt Stimmen die dafür plädieren in kritischen Fällen nur Modelle zu verwenden welche als gut verstanden gelten.

Es gibt einige Probleme welche mit XAI gut überprüft werden können und dadurch kann die Sicherheit und Robustheit von Modellen erhöht werden. Der Aufwand für diese Prüfungen kann aber hoch sein und ist, momentan jedenfalls, nicht automatisierbar. Es fehlen noch Techniken, welche den Einsatz vereinfachen und dadurch die Akzeptanz und Verbreitung fördern.

Die erzeugten Erklärungen, meistens Visualisierungen, müssen nach der Erzeugung durch Menschen interpretiert werden und Schlüsse daraus gezogen werden. Dies ist schwierig und wird dadurch erschwert, dass die Erklärungen selbst Fehler enthalten können. Es fehlen allgemein akzeptierte Definitionen und Kriterien nach welchen Erklärungen ausgewertet werden können.

Die nächsten Jahre werden zeigen ob XAI den Sprung zu einem etablierten Werkzeug, welches in den meisten ML Projekten standardmäßig eingesetzt wird, schafft oder ob die Technik ein Nischendasein fristet und in wenigen Fällen durch Spezialisten angewendet wird.

11 Ausblick

Die Weiterentwicklung der Methoden von XAI sind eng an die zukünftige Entwicklung von ML gekoppelt. Obwohl Prognosen für die Zukunft natürlich schwierig sind, ist es aber momentan wahrscheinlich, dass die grossen Investitionen in AI und ML zumindest noch einige Jahre weitergeführt werden. Durch die Verfügbarkeit von Budgets in Forschung und Industrie, gekoppelt mit den Forderungen vieler Akteure nach einer “verständlichen” AI, ist weiterhin mit einer Vielzahl an Arbeiten im Bereich XAI zu rechnen. Wie schnell dabei Fortschritte erzielt werden können, ist jedoch völlig ungewiss und die Geschichte der AI mit längeren Pausen ohne grössere Fortschritte zeigt, dass man dabei eine gewisse Vorsicht an den Tag legen sollte.

Es gibt Bedenken, welche momentan den Einsatz hochkomplexer Modelle in sicherheitskritischen Gebieten wie Medizin oder Verkehr behindern. Falls diese durch den Konsens abgelöst werden, dass intensives Testen der Modelle ausreichend ist, kann dies die Investitionen in neue XAI Werkzeuge stark dämpfen und würde einen wichtigen Daseinszweck eliminieren.

11.1 Aktuelle Probleme

Obwohl bereits einige Methoden existieren, steht XAI immer noch am Anfang seiner Entwicklung. Viele der geforderten Leistungen sind noch ausstehend oder erst unzureichend erfüllt. Selbst elementare Definitionen sind nicht vorhanden oder werden von Anwendern und Forschern unterschiedlich interpretiert.

Auch die Qualität der Erklärung ist allgemein eher klein. Die meisten Erklärungen, die durch aktuelle Algorithmen erzeugt werden, können nur Details beschreiben. Es ist zwar gut möglich den Einfluss eines einzelnen Pixels zu berechnen, es besteht jedoch keine Verbindung zu abstrakten Begriffen wie Objekten, Körpern oder Personen. Den erzeugten Erklärungen fehlt es oft an Aussagekraft.

Viele der Werkzeuge für XAI sind im Rahmen einer Forschungsarbeit entstanden und stellen sich im praktischen Einsatz als schwierig einzusetzen dar. Einige Probleme entstehen auch durch den schnellen Zyklus mit dem Software für ML sich ändert. Ein Programm, welches vor einem Jahr mit Version X einer Bibliothek funktionierte, ist heute fehlerhaft, weil sich die zugrunde liegenden Funktionsaufrufe geändert haben. Durch den Ursprung aus Forschungsarbeiten ist die fortwährende Weiterentwicklung der Bibliothek nicht immer gewährleistet, während die millionenfach verwendeten Bibliotheken wie TensorFlow von Google stetig weiterentwickelt und gepflegt werden.

11.2 Wünschenswerte Entwicklungen

11.2.1 Mathematisch begründete XAI

XAI benötigt nach Samek (Samek & Müller, 2019) eine formale und allgemein akzeptierte Definition was Erklärungen genau sind. Ein möglicher Weg für eine mathematische Begründung einer Erklärung

zeigt die Arbeit von Montavon (Montavon et al., 2017) durch die Integration der Taylor decomposition.

11.2.2 Modell agnostische Methoden

Wie sich auch in dieser Arbeit gezeigt hat, sind Implementierungen der XAI Werkzeuge schwierig anzuwenden. Oftmals müssen die Modelle bestimmte Bedingungen erfüllen damit eine Analyse möglich ist. In anderen Fällen konnte keine funktionierende Erklärung erzeugt werden, ohne dass man herausfinden konnte weshalb. Weitere Algorithmen wie LIME, welche allgemein funktionieren, sind nötig. Ein grosser Fortschritt wäre eine allgemein anwendbare Erklärungsfunktion, ähnlich wie Keras bereits eine Abstraktion für ML Implementierungen bereitgestellt hat.

11.2.3 Automatisierung

Für die Zukunft wäre es wünschenswert, wenn allgemeine Regeln automatisiert geprüft werden könnten. Mögliche Aufgaben wären dabei die Suche nach einem eventuell vorhandenen Bias oder die Anfälligkeit auf Angriffe wie Adversarial Image Perturbations oder Data Poisoning.

11.3 Fehlende Lösungen

Es besteht die Möglichkeit, dass die hohen Erwartungen an XAI nicht erfüllt werden, weil es sich als unmöglich herausstellen könnte die Vorhersagen in der gewünschten, für Menschen verständlichen, Abstraktion zu erzeugen. Oder die Qualität der Vorhersagen wird nicht erreicht, da die Diskrepanz zu dem realen Modell zu hoch bleibt.

12 Anhang

Source Code

Die umfangreicheren Jupyter Notebooks sind auf der github Seite dieser Arbeit zu finden

<https://github.com/habis-git/MT/tree/master/Jupyter%20Notebooks>

Untenstehende Skripte beziehen sich auf ein bestimmtes Thema.

Entscheidungsbaum Visualisierung mit sklearn und Graphviz

Der folgende Code benötigt mindestens scikit-learn 0.22.

```
1 from sklearn.datasets import load_iris
2 from sklearn import tree
3 from sklearn import datasets
4
5 X, y = load_iris(return_X_y=True)
6 clf = tree.DecisionTreeClassifier()
7 clf = clf.fit(X, y)
8
9 iris = datasets.load_iris()
10
11 dot_data = tree.export_graphviz(clf, out_file=None,
12                                 feature_names=iris.feature_names,
13                                 class_names=iris.target_names,
14                                 filled=True, rounded=True,
15                                 special_characters=True)
16
17 # print tree as text
18 from sklearn.tree import export_text
19 r = export_text(clf, feature_names=iris['feature_names'])
20 print(r)
21
22 # print tree as colored top-down tree
23 import graphviz
24 graph = graphviz.Source(dot_data)
25 graph
26
27 # plot decision surface
28 import numpy as np
29 import matplotlib.pyplot as plt
30 # Parameters
31 n_classes = 3
32 plot_colors = "ryb"
33 plot_step = 0.02
34
35 for pairidx, pair in enumerate([[0, 1], [0, 2], [0, 3],
```

```

36 [1, 2], [1, 3], [2, 3]]):
37 # We only take the two corresponding features
38 X = iris.data[:, pair]
39 y = iris.target
40
41 # Train
42 dTree = tree.DecisionTreeClassifier().fit(X, y)
43
44 # Plot the decision boundary
45 plt.subplot(2, 3, pairidx + 1)
46
47 x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
48 y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
49 xx, yy = np.meshgrid(np.arange(x_min, x_max, plot_step),
50                      np.arange(y_min, y_max, plot_step))
51 plt.tight_layout(h_pad=0.5, w_pad=0.5, pad=2.5)
52
53 Z = dTree.predict(np.c_[xx.ravel(), yy.ravel()])
54 Z = Z.reshape(xx.shape)
55 cs = plt.contourf(xx, yy, Z, cmap=plt.cm.RdYlBu)
56
57 plt.xlabel(iris.feature_names[pair[0]])
58 plt.ylabel(iris.feature_names[pair[1]])
59
60 # Plot the training points
61 for i, color in zip(range(n_classes), plot_colors):
62     idx = np.where(y == i)
63     plt.scatter(X[idx, 0], X[idx, 1], c=color, label=iris.target_names[i],
64                 cmap=plt.cm.RdYlBu, edgecolor='black', s=15)
65
66 plt.suptitle("Decision surface of a decision tree using paired features")
67 plt.legend(loc='lower right', borderpad=0, handletextpad=0)
68 plt.axis("tight")

```

Listing 12.1: Decision Tree Visualisierung

<https://scikit-learn.org/stable/modules/tree.html>

Bild-Klassifikation mit tf-explain

Das folgende Programm erzeugt mit den Bibliotheken Tensorflow (2.0) und tf-explain mit den Methoden “Grad CAM”, “Gradients Input” und “Integrated Gradients” Visualisierungen einer Bild-Klassifizierung.

```

1 import tensorflow as tf
2 from keras.applications.vgg16 import VGG16
3 from keras.preprocessing.image import load_img
4 from keras.preprocessing.image import img_to_array
5 from keras.applications.vgg16 import preprocess_input
6 from keras.applications.vgg16 import decode_predictions
7
8 model = tf.keras.applications.vgg16.VGG16(weights="imagenet", include_top=True
9      )
10
11 #print(model.summary())
12
12 imageOrig = load_img('D:/Master Thesis/dogs-vs-cats/test/DSC05797.JPG',
13 target_size=(224, 224))
13 imageArr = img_to_array(imageOrig) #output Numpy-array

```

```

14
15 imageReshaped = imageArr.reshape((1, imageArr.shape[0], imageArr.shape[1],
16                                 imageArr.shape[2]))
17
18 image = preprocess_input(imageReshaped)
19 predictions = model.predict(imageReshaped)
20
21 import numpy as np
22 top5predictions = np.argsort(predictions)[0,:,:-1][:5]
23
24 labels = decode_predictions(predictions)
25
26 for label in labels[0]:
27     print('%s (%.2f%%)' % (label[1], label[2]*100))
28
29 from tf_explain.core.grad_cam import GradCAM
30 from mpl_toolkits.axes_grid1 import ImageGrid
31
32 def createImageGrid(imageOrig, predictions, labels, explainer, explainerArgs):
33     camImages = [imageOrig]
34     fig = plt.figure(figsize=(20., 20.))
35     grid = ImageGrid(fig, 111, # similar to subplot(111)
36                      nrows_ncols=(2, 3),
37                      axes_pad=0.5, # pad between axes in inch.
38                      )
39     for class_index in top5predictions:
40         camImages.append(explainer.explain(class_index=class_index, **
41                                             explainerArgs))
42
43     i = -1
44     for ax, im in zip(grid, camImages):
45         # Iterating over the grid returns the Axes.
46         ax.set_xticks([])
47         ax.set_yticks([])
48         label = labels[0][i]
49         if i >= 0:
50             ax.set_title('%s (%.2f%%)' % (label[1], label[2]*100), fontsize
51 =20)
52         ax.imshow(im)
53         i = i + 1
54
55     plt.show()
56
57
58 explainer = GradCAM()
59 createImageGrid(imageOrig, predictions, labels, explainer, {'model': model,
60   'layer_name': 'block5_conv3', 'validation_data': data})
61
62 from tf_explain.core.gradients_inputs import GradientsInputs
63 explainer = GradientsInputs()
64 createImageGrid(imageOrig, predictions, labels, explainer, {'model': model,
65   'validation_data': (np.array([imageArr]), None)})
66
67 from tf_explain.core.integrated_gradients import IntegratedGradients
68
69 explainer = IntegratedGradients()
70 createImageGrid(imageOrig, predictions, labels, explainer, {'model': model,
71   'validation_data': (np.array([imageArr]), None)})

```

Listing 12.2: Visualisiertes Neuronales Netz mit Tensorflow und tf-explain

Visualisierung einer Klassifikation mit lime

Die Visualisierung mit lime benutzt als Grundlage das Tutorial “Image Classification Keras” *Tutorial - Image Classification Keras* [66].

```
1 import lime
2 from lime import lime_image
3
4 explainer = lime_image.LimeImageExplainer()
5
6
7 explanation = explainer.explain_instance(np.vstack([imageArr]), model.predict,
8                                         top_labels=5, hide_color=0, num_samples=1000)
9
10
11 camImages = [imageOrig]
12 fig = plt.figure(figsize=(20., 20.))
13 grid = ImageGrid(fig, 111, # similar to subplot(111)
14                   nrows_ncols=(2, 3),
15                   axes_pad=0.5, # pad between axes in inch.
16                   )
17 for class_index in range(0,5):
18     temp, mask = explanation.get_image_and_mask(explanation.top_labels[
19         class_index], positive_only=True, num_features=5, hide_rest=True)
20     camImages.append(mark_boundaries(temp / 2 + 0.5, mask))
21
22 i = -1
23 for ax, im in zip(grid, camImages):
24     # Iterating over the grid returns the Axes.
25     ax.set_xticks([])
26     ax.set_yticks([])
27     label = labels[0][i]
28     if i >= 0:
29         ax.set_title('%s (%.2f%%)' % (label[1], label[2]*100))
30     ax.imshow(im)
31     i = i + 1
32
33 plt.show()
```

Listing 12.3: Visualisiertes Neuronales Netz mit Tensorflow und lime

Akronyme

AI Artificial Intelligence. 2, 6, 9, 56

AIP Adversarial Image Perturbations. 9, 57

CNN Convolutional Neural Network. 18, 28–31, 34, 38

DEK Datenethikkommission. 7, 8

DNN Deep Neural Network. 18, 28

DSGVO

Datenschutz-Grundverordnung. 7, 9

GAM

Generalized Additive Models. 14, 16, 18

GLM

Generalized Linear Models. 14, 16

Grad CAM

Gradient-weighted Class Activation Mapping. 31

LFR Learned fair representations. 14

LIME

Local interpretable model-agnostic explanations. 24, 47, 57

LRP Layer-wise Relevance Propagation. 13, 22, 23

LSTM

Long short-term memory. 18, 47

M-GBM

Monotonic gradient boosting. 14

ML Machine Learning. 1, 2, 4–9, 11, 13, 24, 43, 50, 52–57, 64

PATE

Private aggregation of teacher ensembles. 14

RNN Recurrent Neural Network. 18

SBRL

Scalable Bayesian rule list. 14

SLIM

Supersparse linear integer models. 14

SVCCA

Singular Vector Canonical Correlation Analysis. 26, 54

TCAV

Testing with Concept Activation Vectors. 26, 54

XAI Explainable Artificial Intelligence. 1, 4, 8–14, 18, 42, 54–57

Glossar

Bias deutsch Tendenz oder Voreingenommenheit, kann in Machine Learning Modellen auftreten, wenn die Trainingsdaten unausgewogen sind. Dies kann zu einer sogenannten selbsterfüllenden Prophesie werden, indem die Resultate des Modells zu neuen Trainingsdatensätzen führen, welche die Tendenz noch verstärken. 9, 12, 30, 50, 57

Binärer Klassifikator

Ein binärer Klassifikator ist eine Sonderform eines Klassifikators, der nur zwei Klassen kennt. Häufig sind das ja/nein Fragen zum Beispiel "ist auf diesem Bild ein Tumor erkennbar?". 30, 47

Blackbox

Als Blackbox werden Modelle bezeichnet, welche durch ihre Struktur und die Art und Weise wie die Daten verarbeitet, werden nur schwer verständlich sind. Typische Blackbox Modelle sind tiefe neuronale Netze (DNN) und andere Varianten von neuronalen Netzen. Blackbox Modelle gelten in der Regel als leistungsfähiger als Whitebox Modelle. 9, 11, 47

Datenschutz-Grundverordnung

Verordnung der Europäischen Union mit der die Regeln zur Verarbeitung personenbezogener Daten durch die meisten Datenverarbeiter, sowohl private wie öffentliche, EU-weit vereinheitlicht werden. Die Datenschutz-Grundverordnung definiert Anforderungen an den Datenschutz und die Datenverarbeitung, welche ML Lösungen betreffen. 9

Decision Tree

Entscheidungsbaum, Familie von ML Algorithmen. 14, 16–18, 41, 42, 46

Deep Neural Network

deutsch tiefes lernen, bezeichnet neuronale Netze mit vielen Zwischenschichten. 19

Explainable Artificial Intelligence

deutsch erklärbare künstliche Intelligenz, Methodiken um Menschen die Vorhersagen durch Modelle des maschinellen Lernens zu erläutern. 35, 36, 40, 57

Grad CAM

Gradient-weighted Class Activation Mapping, Technik welche für eine Entscheidung relevanten Bildinhalte optisch hervorhebt. 13, 19, 20, 23, 32, 36–40

Gradienten

Vektor dessen Komponenten die partiellen Ableitungen im Punkt P sind, der Gradient zeigt deshalb in die Richtung der größten Änderung. 19

Gradients Input

Verfahren nach Simonyan (Simonyan et al., 2013). Gradients Input misst den relativen Einfluss einer Eigenschaft. Dies wird erreicht durch das Berechnen des Gradienten der Entscheidung mit Einbezug der Eingangs-Eigenschaften. 13, 31, 32, 38–40

Kluger-Hans-Effekt

“Kluger Hans” war ein Pferd anfangs des 20. Jahrhunderts, das angeblich rechnen und zählen konnte. Es stellte sich jedoch heraus, dass es nur auf die feinen Nuancen der Mimik und Körpersprache des Fragestellers reagierte. Seitdem wird als “Kluger-Hans-Effekt” eine unbewusste Beeinflussung des Studienobjektes bezeichnet. In Machine Learning Lösungen kann der “Kluger-Hans-Effekt” auftauchen, wenn ein Modell mit Daten trainiert wird, welche die Vorhersage unbewusst in eine bestimmte Richtung lenken. 28

LIME

Local interpretable model-agnostic explanations, eine unabhängig des verwendeten Algorithmus anwendbare Erklärungstechnik für Blackbox Modelle. 38, 39, 45, 46

Naïve Bayes

Auch naiver Bayes-Klassifikator genannt. Basiert auf dem Bayesschen Theorem mit der naiven Grundannahme, dass jedes Attribut nur vom Klassenattribut abhängt. Obwohl dies häufig nicht der Fall ist, funktionieren Naïve Bayes Klassifikatoren sehr gut. 18, 46

neuronales Netz

Algorithmus welcher nach dem Vorbild des menschlichen Gehirns entworfen wurde um Muster und Beziehungen in Daten zu erkennen. 9, 23, 53

Occlusion Sensitivity

Verfahren, um die für eine Klassifikation relevanten Bildinhalte zu finden, indem bestimmte Bildinhalte entfernt werden (Occlusion) und die dabei entstehende Veränderung auf die Klassifikation gemessen wird. 13, 20, 21, 31–33, 35, 37–40

Whitebox

Ein Whitebox Modell ist durch seinen Aufbau verständlich und die Entscheidungsfindung ist nachvollziehbar (nicht zu verwechseln mit einfach). Whitebox Modelle sind häufig im Bereich der Statistik angesiedelt. Es gibt kein festes Entscheidungskriterium ob eine Technik eher zu den Blackbox oder Whitebox Modellen gehört, bei vielen Verfahren besteht aber ein Konsens darüber zu welcher Gruppe sie gehören. 47

Abbildungsverzeichnis

2.1	Entwicklung des Machine Learning als Zeitachse	3
3.1	Ablauf einer erklärbaren Machine Learning Anwendung	5
5.1	CRISP-DM Prozessablauf	10
5.2	Erklärungen im ML Ablauf	11
6.1	Voraussetzungen lineare Regression	15
6.2	Auschluss-Bedingungen lineare Regression	16
6.3	Entscheidungsbaum visualisiert.	17
6.4	Entscheidungsbaum als Flächen dargestellt	17
6.5	scope-rules Ausgabe der Regeln ¹	17
7.1	Grad CAM Analyse für verschiedene Klassen	19
7.2	Occlusion Sensitivity Beispiel	21
7.3	Testbild mit Occlusion Sensivity	21
7.4	LRP Berechnung	23
7.5	Pixel basierte Relevanz	23
7.6	Vergleich Pixel-wise und LRP	23
7.7	Darstellung relevanter Bildinhalte durch LIME	24
7.8	Darstellung Vorgehensweise TCAV	26
7.9	Vergleich verschiedener Klassen mit SVCCA ²	27
8.1	Klassifizierung eines Pferdes in Pascal VOC ³	28
8.2	fiktives Logo	29
8.3	Ursprüngliches Bild	29
8.4	Manipuliertes Bild	29
8.5	Log-loss / Accuracy Dog vs. Cat	30
8.6	Bewertung des Hund-Katze Netzwerkes	31
8.7	Testbild ohne Logo	31
8.8	Testbild mit Logo	32
8.9	Testbild Katze ohne Logo	32
8.10	Testbild Katze mit Logo	33
8.11	Testbild Sonnenblume	34
8.12	Testbild Falter ohne Logo	34
8.13	Testbild Falter mit Logo	35
8.14	Original Testbild Katze	36
8.15	Testbild Katze visualisiert mit Grad CAM	36
8.16	Testbild Toilettepapier-Rolle	37
8.17	Testbild Meerschweinchen	37
8.18	Testbild Meerschweinchen Grad CAM	38
8.19	Testbild Meerschweinchen div. Verfahren	39
8.20	Testbild getigerte Katze	40

8.21	Testbild getigerte Katze frontal	40
8.22	Konfusionsmatrix, Test Zusammenfassung und Accuracy für Decision Tree	41
8.23	Konfusionsmatrix, Test Zusammenfassung und Accuracy für Random Forest	42
8.24	Top Features Film Review Klassifizierung	43
8.25	Visualisierung positives Film Review mit ELI5	44
8.26	Visualisierung negatives Film Review mit ELI5	44
8.27	Visualisierung positives Film Review mit LIME	45
8.28	Visualisierung negatives Film Review mit LIME	45
8.29	Konfusionsmatrix, Test Zusammenfassung und Accuracy für Naïve Bayes	46
8.30	Visualisierung positives Film Review mit LIME und NB	46
8.31	Visualisierung negatives Film Review mit LIME und NB	46
8.32	Zusammenfassung LSTM Netz für Movie Sentiment Analyse	48
8.33	Performance des LSTM Netz für Movie Sentiment Analyse	48
8.34	Visualisierung Movie Review 413	48
8.35	Visualisierung Movie Review 414	49
9.1	Adversarial Beispiel ⁴	51
9.2	Ursprungs-Bild ⁵	51
9.3	Durch Adversarial Angriff erzeugtes Bild	51
9.4	Gefälschtes Verkehrsschild als Adversarial Attack ⁶	52

Tabellenverzeichnis

6.1	Entscheidungstabelle erklärbare Modelle ⁷	15
8.1	Wahrscheinlichkeiten Testbild Katze	36
8.2	Wahrscheinlichkeiten Testbild Meerschweinchen	37

Literaturverzeichnis Artikel

- Arya, V., Bellamy, R. K. E., Chen, P.-Y., Dhurandhar, A., Hind, M., Hoffman, S. C., Houde, S., Liao, Q. V., Luss, R., Mojsilović, A., Mourad, S., Pedemonte, P., Raghavendra, R., Richards, J., Sattigeri, P., Shanmugam, K., Singh, M., Varshney, K. R., Wei, D. & Zhang, Y. (2019). One Explanation Does Not Fit All: A Toolkit and Taxonomy of AI Explainability Techniques, arXiv <http://arxiv.org/abs/1909.03012v2>.
- Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R. & Samek, W. (2015). On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation (O. D. Suarez, Hrsg.). *PLOS ONE*, 10(7), e0130140. <https://doi.org/10.1371/journal.pone.0130140>

- Craven, M. W. & Shavlik, J. W. (1994). Using Sampling and Queries to Extract Rules from Trained Neural Networks, 37–45. <https://doi.org/10.1016/b978-1-55860-335-6.50013-1>
- Craven, M. W. & Shavlik, J. W. (1995). Extracting Tree-Structured Representations of Trained Networks, 24–30.
- Everingham, M., Gool, L. V., Williams, C. K. I., Winn, J. & Zisserman, A. (2010). The PASCAL Visual Object Classes (VOC) Challenge.
- Friedman, J. H. & Popescu, B. E. (2008). Predictive learning via rule ensembles. *Annals of Applied Statistics 2008, Vol. 2, No. 3*, 916-954, arXiv <http://arxiv.org/abs/0811.1679v1>. <https://doi.org/10.1214/07-AOAS148>
- Goodfellow, I. J., Shlens, J. & Szegedy, C. (2014). Explaining and Harnessing Adversarial Examples, arXiv <http://arxiv.org/abs/1412.6572v3>.
- Gu, T., Liu, K., Dolan-Gavitt, B. & Garg, S. (2019). BadNets: Evaluating Backdooring Attacks on Deep Neural Networks. *IEEE Access*, 7, 47230–47244. <https://doi.org/10.1109/access.2019.2909068>
- Johansson, U. & Niklasson, L. (2009). Evolving Decision Trees Using Oracle Guides, 238–244. <https://doi.org/10.1109/CIDM.2009.4938655>
- Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F. & Sayres, R. (2017). Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV). *ICML 2018*, arXiv <http://arxiv.org/abs/1711.11279v5>.
- Lakkaraju, H., Kamar, E., Caruana, R. & Leskovec, J. (2017). Interpretable & Explorable Approximations of Black Box Models, arXiv <http://arxiv.org/abs/1707.01154v1>.
- Lapuschkin, S., Binder, A., Montavon, G., Müller, K.-R. & Samek, W. (2016). The LRP Toolbox for Artificial Neural Networks. *Journal of Machine Learning Research*, 17(114), 1–5. <http://jmlr.org/papers/v17/15-618.html>
- Lapuschkin, S., Wäldchen, S., Binder, A., Montavon, G., Samek, W. & Müller, K.-R. (2019). Unmasking Clever Hans Predictors and Assessing What Machines Really Learn, arXiv <http://arxiv.org/abs/1902.10178v1>. <https://doi.org/10.1038/s41467-019-08987-4>
- Montavon, G., Lapuschkin, S., Binder, A., Samek, W. & Müller, K.-R. (2017). Explaining nonlinear classification decisions with deep Taylor decomposition. *Pattern Recognition*, 65, 211–222. <https://doi.org/10.1016/j.patcog.2016.11.008>
- Oh, S. J., Schiele, B. & Fritz, M. (2019). Towards Reverse-Engineering Black-Box Neural Networks, 121–144. https://doi.org/10.1007/978-3-030-28954-6_7
- Pang, B. & Lee, L. (2004). A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts.
- Papernot, N., Faghri, F., Carlini, N., Goodfellow, I., Feinman, R., Kurakin, A., Xie, C., Sharma, Y., Brown, T., Roy, A., Matyasko, A., Behzadan, V., Hambardzumyan, K., Zhang, Z., Juang, Y.-L., Li, Z., Sheatsley, R., Garg, A., Uesato, J., ... Long, R. (2018). Technical Report on the CleverHans v2.1.0 Adversarial Examples Library. *arXiv preprint arXiv:1610.00768*.
- Raghu, M., Gilmer, J., Yosinski, J. & Sohl-Dickstein, J. (2017). SVCCA: Singular Vector Canonical Correlation Analysis for Deep Learning Dynamics and Interpretability, arXiv <http://arxiv.org/abs/1706.05806v2>.
- Ras, G., van Gerven, M. & Haselager, P. (2018). Explanation Methods in Deep Learning: Users, Values, Concerns and Challenges, 19–36. https://doi.org/10.1007/978-3-319-98131-4_2
- Ribeiro, M. T., Singh, S. & Guestrin, C. (2016). "Why Should I Trust You?". <https://doi.org/10.1145/2939672.2939778>
- Rudin, C. (2018). Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead. *Nature Machine Intelligence*, Vol 1, May 2019, 206-215, arXiv <http://arxiv.org/abs/1811.10154v3>.
- Samek, W. & Müller, K.-R. (2019). Towards Explainable Artificial Intelligence, 5–22. https://doi.org/10.1007/978-3-030-28954-6_1

- Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D. & Batra, D. (2016). Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization, arXiv <http://arxiv.org/abs/1610.02391v4>. <https://doi.org/10.1109/s11263-019-01228-7>
- Simonyan, K., Vedaldi, A. & Zisserman, A. (2013). Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps, arXiv <http://arxiv.org/abs/1312.6034v2>.
- Simonyan, K. & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition, arXiv <http://arxiv.org/abs/1409.1556v6>.
- Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. & Fergus, R. (2013). Intriguing properties of neural networks, arXiv <http://arxiv.org/abs/1312.6199v4>.
- Zeiler, M. D. & Fergus, R. (2013). Visualizing and Understanding Convolutional Networks, arXiv <http://arxiv.org/abs/1311.2901v3>.
- Zhang, X., Zhu, X. & Lessard, L. (2019). Online Data Poisoning Attack, arXiv <http://arxiv.org/abs/1903.01666v2>.

Literaturverzeichnis Bücher

Explainable and Interpretable Models in Computer Vision and Machine Learning. (2018, 1. September). Springer-Verlag GmbH. https://www.ebook.de/de/product/33610206/explainable_and_interpretable_models_in_computer_vision_and_machine_learning.html

Linkverzeichnis

- bitkom. (o.D.). Machine Learning und die Transparenzanforderungen der DS-GVO. <https://www.bitkom.org/sites/default/files/file/import/180926-Machine-Learning-und-DSGVO.pdf>
- Chokshi, N. (2020). Tesla Autopilot System Found Probably at Fault in 2018 Crash. <https://www.nytimes.com/2020/02/25/business/tesla-autopilot-ntsb.html>
- Dataset Dog vs. Cats. (o.D.). <https://www.kaggle.com/c/dogs-vs-cats>
- der Bundesregierung, D. (2019). Gutachten der Datenethikkommission. https://www.bmjv.de/SharedDocs/Downloads/DE/Themen/Fokusthemen/Gutachten_DEK_DE.pdf?__blob=publicationFile&v=2
- Eddins, S. (2017). Network Visualization Based on Occlusion Sensitivity. <https://blogs.mathworks.com/deep-learning/2017/12/15/network-visualization-based-on-occlusion-sensitivity/>
- ELI5: A library for debugging/inspecting machine learning classifiers and explaining their predictions [ELI5]. (o.D.). <https://eli5.readthedocs.io/>
- Goodfellow, I. (2020). CleverHans a Python library to benchmark machine learning systems' vulnerability to adversarial examples. <https://github.com/tensorflow/cleverhans>
- Goyal, R. (2017). Visual explanation for video recognition. <https://medium.com/twentybn/visual-explanation-for-video-recognition-87e9ba2a675b>

- Habegger, M. (o.D. a). Beispiel "Kluger-HansEffekt in manipuliertem CNN. https://github.com/habis-git/MT/blob/master/Models/Manipulated%20Data/Cat_vs_Dog_Manipulated_visualization.ipynb
- Habegger, M. (o.D. b). Manipulated Tensorflow CNN Dog vs. Cats. <https://github.com/habis-git/MT/blob/master/Models/Manipulated%20Data/model.h5>
- Habegger, M. (o.D. c). Text Klassifizierung mit Decision Tree. <https://github.com/habis-git/MT/blob/master/Jupyter%20Notebooks/TextClassificationDT.ipynb>
- Habegger, M. (202). Text Klassifizierung eines LSTM Modelles mit LIME. <https://github.com/habis-git/MT/blob/master/Jupyter%20Notebooks/Moveriew%20Sentiment%20with%20Keras%20LSTM%20and%20lime%20explainer%20.ipynb>
- Habegger, M. (2020a). Kombinierte Verfahren für die Erklärung von Bild-Klassifikationen mit tf_explain und lime. <https://github.com/habis-git/MT/blob/master/Jupyter%20Notebooks/CombinedVisualizations.ipynb>
- Habegger, M. (2020b). Mehrere Klassen pro Bild mit verschiedenen visualisierungs Verfahren. <https://github.com/habis-git/MT/blob/master/Jupyter%20Notebooks/Visualizations.ipynb>
- Habegger, M. (2020c). Text Klassifizierung mit ELI5. <https://github.com/habis-git/MT/blob/master/Jupyter%20Notebooks/TextClassification.ipynb>
- Habegger, M. (2020d). Text Klassifizierung mit lime. <https://github.com/habis-git/MT/blob/master/Jupyter%20Notebooks/TextClassificationLime.ipynb>
- ImageNet Challenge. (o.D.). <http://www.image-net.org/challenges/LSVRC/>
- Institut, F. (o.D.). Explainable AI Demos. <https://lrvserver.hhi.fraunhofer.de/handwriting-classification>
- Kate Conger, R. F. & Kovaleski, S. F. (2019). San Francisco Bans Facial Recognition Technology. <https://www.nytimes.com/2019/05/14/us/facial-recognition-ban-san-francisco.html>
- Kim, B. (2018). Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV) [ICML 2018]. <https://github.com/tensorflow/tcav>
- Lapuschkin, S. (2020). The LRP Toolbox for Artificial Neural Networks. https://github.com/sebastian-lapuschkin/lrp_toolbox
- Lee, P. (2016). Learning from Tay's introduction. <https://blogs.microsoft.com/blog/2016/03/25/learning-tays-introduction/>
- Malik, U. (2018). Text Classification with Python and Scikit-Learn. <https://stackabuse.com/text-classification-with-python-and-scikit-learn/>
- Meudec, R. (o.D.). tf-explain. <https://github.com/sicara/tf-explain>
- Molnar, C. (2019). Interpretable Machine Learning, A Guide for Making Black Box Models Explainable. <https://christophm.github.io/interpretable-ml-book/>
- Natural Language Toolkit [NLTK]. (o.D.). <https://www.nltk.org/>
- OpenAI. (o.D.). <https://openai.com/>
- Pichai, S. (2018). AI at Google: our principles. <https://www.blog.google/technology/ai/ai-principles/>
- A Quick Introduction to Deep Taylor Decomposition. (2017). <http://www.heatmapping.org/deeptaylor/>
- Raghu, M. (2017). Interpreting Deep Neural Networks with SVCCA. <https://ai.googleblog.com/2017/11/interpreting-deep-neural-networks-with.html>
- Ribeiro, M. T. C. (2019). Lime: Explaining the predictions of any machine learning classifier [Lime]. <https://github.com/marcotcr/lime>
- scikit-learn Machine Learning in Python [scikit-learn]. (o.D.). <https://scikit-learn.org/stable/index.html>
- skope-rules. (2019). <https://github.com/scikit-learn-contrib/skope-rules>
- Steve Povolny, S. T. (2020). Model Hacking ADAS to Pave Safer Roads for Autonomous Vehicles. <https://www.mcafee.com/blogs/other-blogs/mcafee-labs/model-hacking-adas-to-pave-safer-roads-for-autonomous-vehicles/>
- Surma, G. (2018). Image Classifier Cats vs Dogs. <https://towardsdatascience.com/image-classifier-cats-vs-dogs-with-convolutional-neural-networks-cnns-and-google-colabs-4e9af21ae7a8>
- Tensorflow. (o.D.). <https://www.tensorflow.org/>

Tensorflow. (2018). VGG16 Modell für Imagenet Klassifikationen. https://github.com/tensorflow/tensorflow/blob/r1.8/tensorflow/python/keras/_impl/keras/applications/vgg16.py

Thiebaut, N. (2017). LIME of words: interpreting Recurrent Neural Networks predictions. <https://data4thought.com/deep-lime.html>

Tutorial - Image Classification Keras. (2019). <https://github.com/marcotcr/lime/blob/master/doc/notebooks/Tutorial%20-%20Image%20Classification%20Keras.ipynb>

Tutorial, T. (2020). Adversarial example using FGSM. https://www.tensorflow.org/tutorials/generative/adversarial_fgsm

Listings

7.1	Grad CAM Visualisierung für die wahrscheinlichste Klasse	20
7.2	Occlusion Sensitivity Visualisierung für die wahrscheinlichste Klasse	22
7.3	LIME Visualisierung für die wahrscheinlichste Klasse	25
8.1	CNN für Dog vs. Cats	30
8.2	LSTM Modell für LIME Movie Sentiment Analyse	47
12.1	Decision Tree Visualisierung	58
12.2	Visualisiertes Neuronales Netz mit Tensorflow und tf-explain	59
12.3	Visualisiertes Neuronales Netz mit Tensorflow und lime	61