

Explainable AI

Stand der Forschung und Technik

Master Thesis

Studienrichtung: MAS Data Science
Autor: Marc Habegger
Dozent:
Experte: Max Kleiner
Datum: 22. Februar 2020

Inhaltsverzeichnis

1 Einleitung	1
2 Was bedeutet Erklärbarkeit?	4
2.1 Unterschiedliche Ziele	4
2.2 Zielgebiete von XAI	5
2.3 Bessere Anwendungen durch Einblick in die Funktionsweise	5
2.3.1 Datenschutz	6
2.4 Sicherheit	7
2.5 Regulatorische Bedingungen	7
2.6 Haftungsfragen	7
3 Anwendung von XAI auf Modelle	8
3.1 Anwendungsfälle	8
3.1.1 Exploration	8
3.1.2 Feature Engineering	9
3.1.3 Ein interpretierbares Model erstellen	9
3.1.4 Modelle interpretieren	9
3.2 Grundsätzlich erklärbare Algorithmen	11
3.2.1 Lineare Regression	11
3.2.2 GLM/GAM	11
3.2.3 Entscheidungsbäume	12
3.2.4 RuleFit	13
3.2.5 Naive Bayes	13
3.3 Techniken für nicht direkt erklärbare Modelle	13
3.3.1 Grad CAM	13
3.3.2 Occlusion Sensitivity	14
3.3.3 LRP	15
3.3.4 Local Surrogate (LIME)	15
3.3.5 TCAV	18
3.3.6 SVCCA	19
4 Konkrete Anwendung von XAI	21
4.1 Bilderkennung	21
4.1.1 WhiteBox Model: Klassifikation Hund - Katze	21
4.1.2 Vortrainiertes ImageNet Modell	28
4.2 Texterkennung	33
4.2.1 Stimmungs-Analyse von Film-Bewertungen	33
5 Schwächen von ML Modellen erkennen	36
5.1 Diskriminierung durch Bias	36
5.2 Adversarial Attacks	36
5.3 Data Poisoning	37

6 Weiterentwicklung von XAI	39
7 Anhang	40
7.1 Source Code	40
7.1.1 Entscheidungsbaum Visualisierung mit sklearn und Graphviz	40
7.1.2 Bild-Klassifikation mit tf-explain	41
7.1.3 Visualisierung einer Klassifikation mit lime	42
Index	47

1 Einleitung

Computer Programme bestehen in der Regel aus tausenden bis Millionen Zeilen von Anweisungen welche für die verschiedenen Aspekte eines Programmes zuständig sind. Einige der Programmroutine stellen die grafische Benutzeroberfläche dar während andere sich um das Speichern und laden von Dateien kümmern. Andere wiederum 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 und 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 und gewisse Gebiete waren durch die Komplexität der Aufgabenstellung nur rudimentär in Regeln zu fassen. Solche Gebiete sind unter anderem Bilderkennung, Text- oder Sprachverständnis, Gebiete in denen Menschen und Tiere bedeutend bessere Fähigkeiten zeigen als Computer Programme.

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. 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.

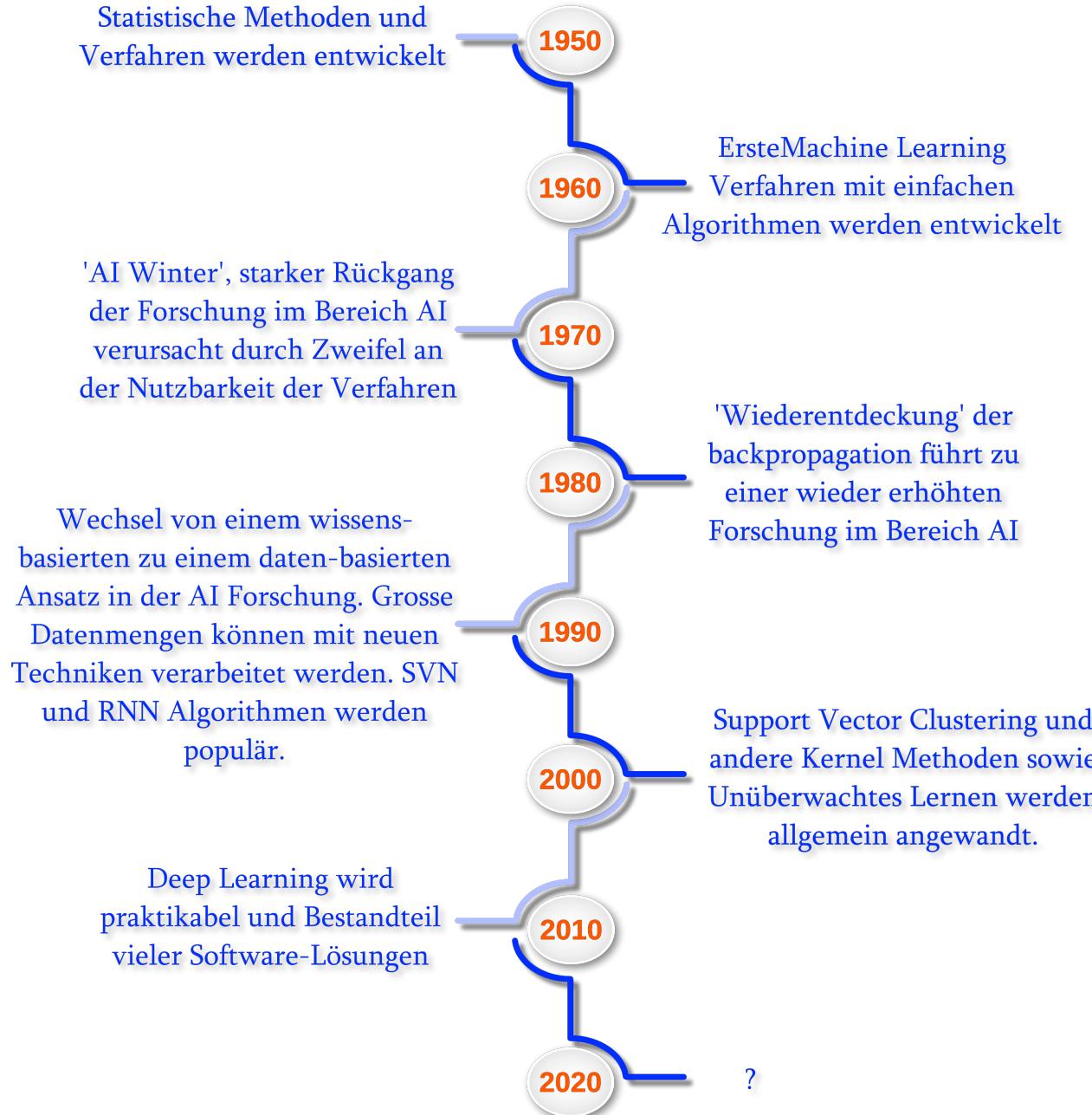


Abbildung 1.1: Entwicklung des Machine Learning als Zeitachse

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

ML Anwendungen für die Krebserkennung bieten großes Potenzial. Insbesondere die ermüdenden

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 ML 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.

2 Was bedeutet Erklärbarkeit?

ML erzeugt Resultate welche je nach Anwendungsfall eine Entscheidung für Klasse (Pferd, Schaf, Auto), eine Zuordnung zu einer Gruppe (Premium-Kunde, Gelegenheitskäufer etc.) oder es wird ein numerischer Wert generiert (15 Grad Celsius am 3. April). Da sowohl die Erzeugung des Models als auch die Berechnung des Resultates automatisch erfolgt können die Schritte auf dem Weg zu dem Resultat nicht direkt nachvollzogen werden.

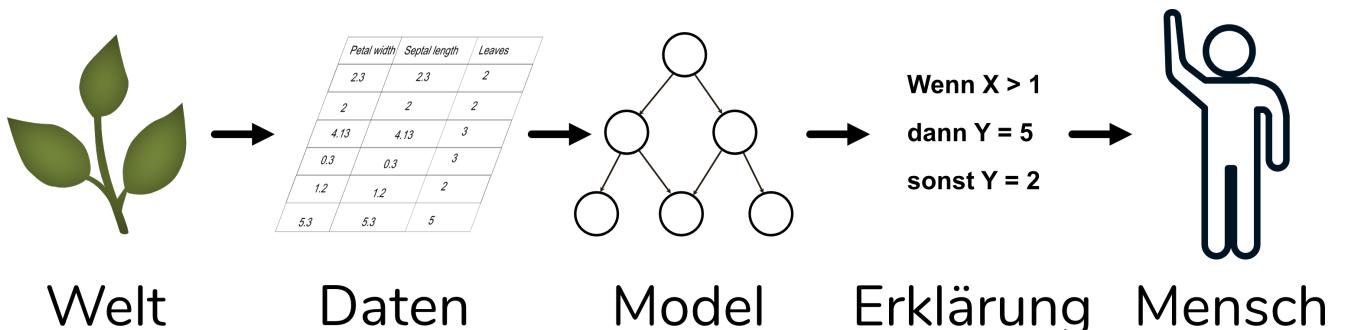


Abbildung 2.1: Ablauf einer erklärbaren Machine Learning Anwendung

Eine ML Lösung beginnt mit der Beobachtung von realen Ereignissen in der Welt. Dies können die Anzahl Blätter und deren Länge einer Pflanzungsgattung oder auch Häuserpreise in Brooklyn sein. Diese Beobachtungen werden gesammelt und bilden die Daten Grundlage mit deren ein Model erzeugt werden kann. Aus diesem Model kann eine Erklärung erzeugt werden die ein Mensch verwenden kann um das Resultat zu erklären.

2.1 Unterschiedliche Ziele

Eine Anwendung welche 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 kann das Datenset gezielt erweitert oder reduziert werden so dass ein ausgeglichenes Verhältnis erzeugt wird.

Modell

Kann man aus dem Modell 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

Erklärung weshalb ein bestimmtes Muster in den Daten zu der beobachteten Klassifizierung geführt hat. Dies ist insbesondere für Anwender/Kunden einer ML Lösung um a) das Verständnis für die Maschinelle Entscheidung zu erhöhen oder b) eine gesetzlich Vorgeschriebene Anfechtbarkeit der Entscheidung zu ermöglichen.

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

Experten

Diese Gruppe kann weiter unterteilt werden in

Forscher

Entwickelt neue Methoden und Algorithmen für das ML, verbessert bestehende Algorithmen

Entwickler

Setzt bestehende Methodiken und Algorithmen ein um eine konkrete Aufgabenstellung zu lösen

Benutzer

Auch bei den Benutzern gibt es verschiedene Ausprägungen

Eigentümer

Eigentümer/Auftraggeber

Anwender

Person deren Daten verwendet werden

Anspruchsgruppe (Stakeholder)

Die Anforderungen an ein erklärbares Modell unterscheiden sich so stark je nach betrachtet Komponente und der Anwendergruppe. Daraus ergibt sich dass verschiedene Techniken benötigt werden um AI Lösungen generell erklärbar zu machen.

2.2 Zielgebiete von XAI

2.3 Bessere Anwendungen durch Einblick in die Funktionsweise

Ein Grundsatz jedes ML Projektes lautet dass der Erfolg nicht garantiert ist. Ausgehend von bestehenden Daten kann man nicht sicher sein dass diese ausreichend Informationen liefern um ein Modell zu entwickeln welches den Anforderungen entspricht. Aber auch falls dies der Fall sein sollte besteht immer noch die Problematik dass unzählige Algorithmen mit wiederum unzähligen Parametern existieren welche zur Anwendungen kommen können. Auch wenn Lösungen, oder Ansätze davon, existieren um mit dieser Problematik umzugehen so ist es in der Regel ein langwieriger und oftmals teuer 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 und so entweder Arbeitszeit oder Rechenleistung beim berechnen der Modell einzusparen.

Ebenso kann ein Modell falsche Resultate liefern, sei es weil die ursprünglichen Trainingsdaten unvollständig waren, oder weil neue Bedingungen aufgetreten sind. Eine Erklärung weshalb dieses falsche Resultat erzeugt wurde kann den Entwickler in die Lage versetzen ein verbessertes Model zu erzeugen. Dies kann sowohl beim Erstellen des Modelles geschehen (training) oder bei der Aufbearbeitung der Daten welche durch das Model verarbeitet werden sollen (preprocessing).

2.3.1 Datenschutz

Jede ML Lösung setzt grosse Datenmengen voraus. Diese müssen den Anforderungen des Datenschutzes entsprechend aufbereitet und gegebenenfalls anonymisiert werden.

Der jüngste Bericht der Datenethikkommission (DEK) der Deutschen Regierung *Gutachten der Datenethikkommission* [1] 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. algorithmdeterminierte 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

Explainable artificial intelligence 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. In Kapitel 2.4 wird näher auf die Gefahren eingegangen welche durch die Anwendung von ML entstehen können.

2.4 Sicherheit

Die Sicherheit von ML Modellen ist durch verschiedene Angriffs-Methoden gefährdet. In der Regel geht man für Sicherheitsüberlegungen von einem Black Box Model aus, da ein Angreifer das Model in der Anwendung (d.h. nach dem Training) angreift.

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 gestohlen oder Wettbewerbsvorteile eliminiert werden.

Membership Inference Attacks

Diese Art von Angriff versucht herauszufinden ob ein Datensatz dazu verwendet wurde das vorliegende Model zu trainieren.

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 Models für Systeme einer Zutrittskontrolle fatal sein können. 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 ?? erläutert.

2.5 Regulatorische Bedingungen

2.6 Haftungsfragen

3 Anwendung von XAI auf Modelle

Nach (Oh et al., 2019) hat Art des Modells von grossem Einfluss auf die Möglichkeiten der Erklärbarkeit. Generell wird unterschieden zwischen

Whitebox Modelle

sind unter der Kontrolle desjenigen welcher eine erklärende Analyse durchführt, sowohl die Daten wie der Aufbau des Modelles sind bekannt

Blackbox Modelle

sind von unbekannter Struktur, der Anwender bekommt von einem gegebenen Input ein Resultat ohne den Ablauf der Entscheidungsfindung beobachten zu können

Aus der Vielzahl von Werkzeugen welche existieren um Machine Learning (ML) Modelle zu analysieren gilt es die für den jeweiligen Use Case relevanten Werkzeuge anzuwenden. TODO: Übersicht der Techniken und Anwendungsbereiche wie in der Grafik

3.1 Anwendungsfälle

3.1.1 Exploration

- Biplot
- Darstellung der Korrelation (Correlation graph)
- Korrelationsmatrix
- Heatmap
- Parallele Koordinatendarstellung (Parallel coordinates plot)
- Projektion MDS, t-SNE, UMAP
- Radar Plot
- Scatter Plot
- Univariate Statistiken: Häufigkeits-Verteilung, Histogram, Pivot-Tabelle

3.1.2 Feature Engineering

3.1.3 Ein interpretierbares Model erstellen

- Entscheidungsbaum
- Generalized Additive Models (GAM)
- Generalized Linear Models (GLM)
- 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)

3.1.4 Modelle interpretieren

Neben vielen Methoden welche nur für bestimmte Klassen von Verfahren anwendbar sind, gibt es einige die Allgemein anwendbar sind.

LIME 3.3.4

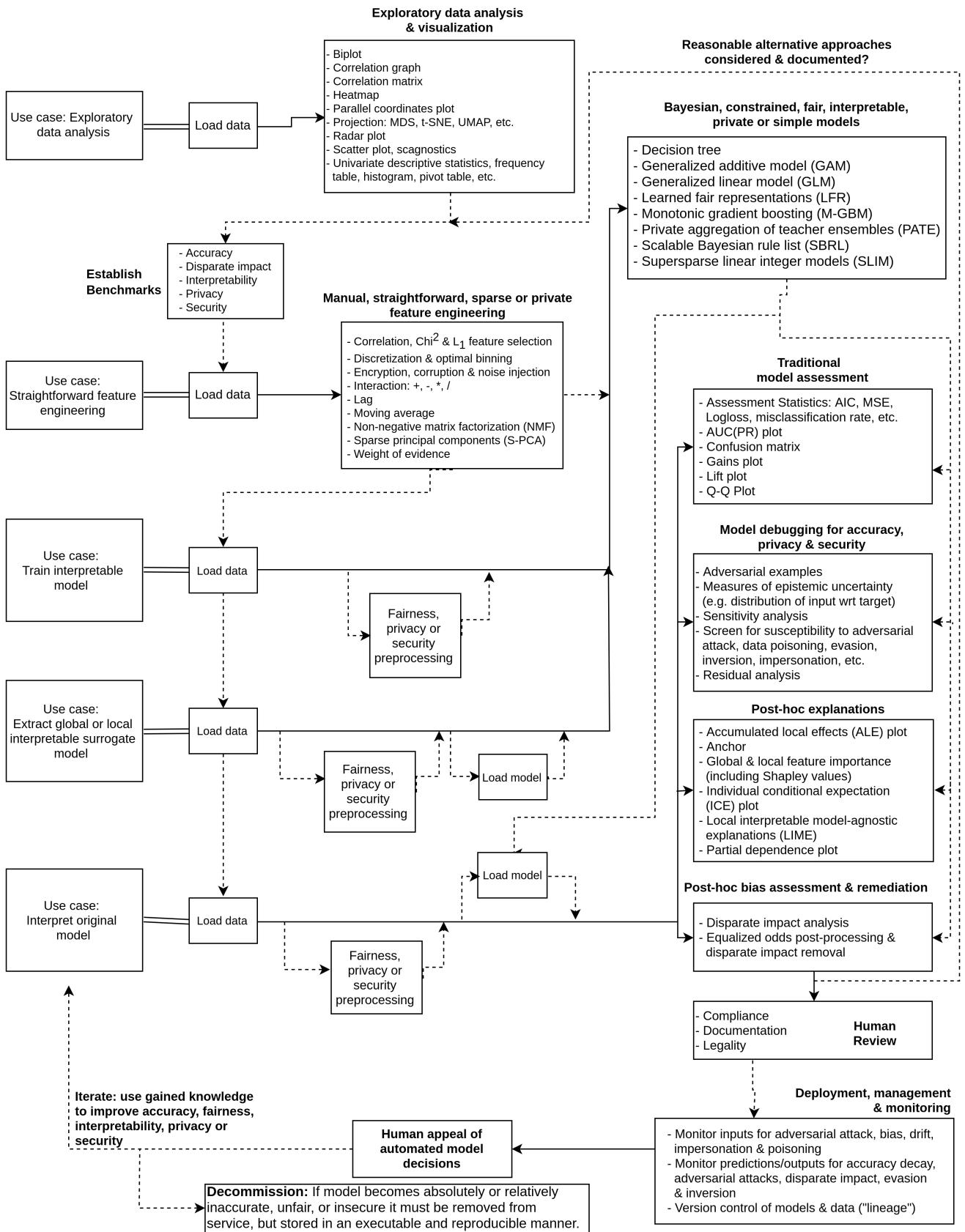


Abbildung 3.1: Quelle: <https://github.com/h2oai/mli-resources>

3.2 Grundsätzlich erklärbare Algorithmen

3.2.1 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 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.

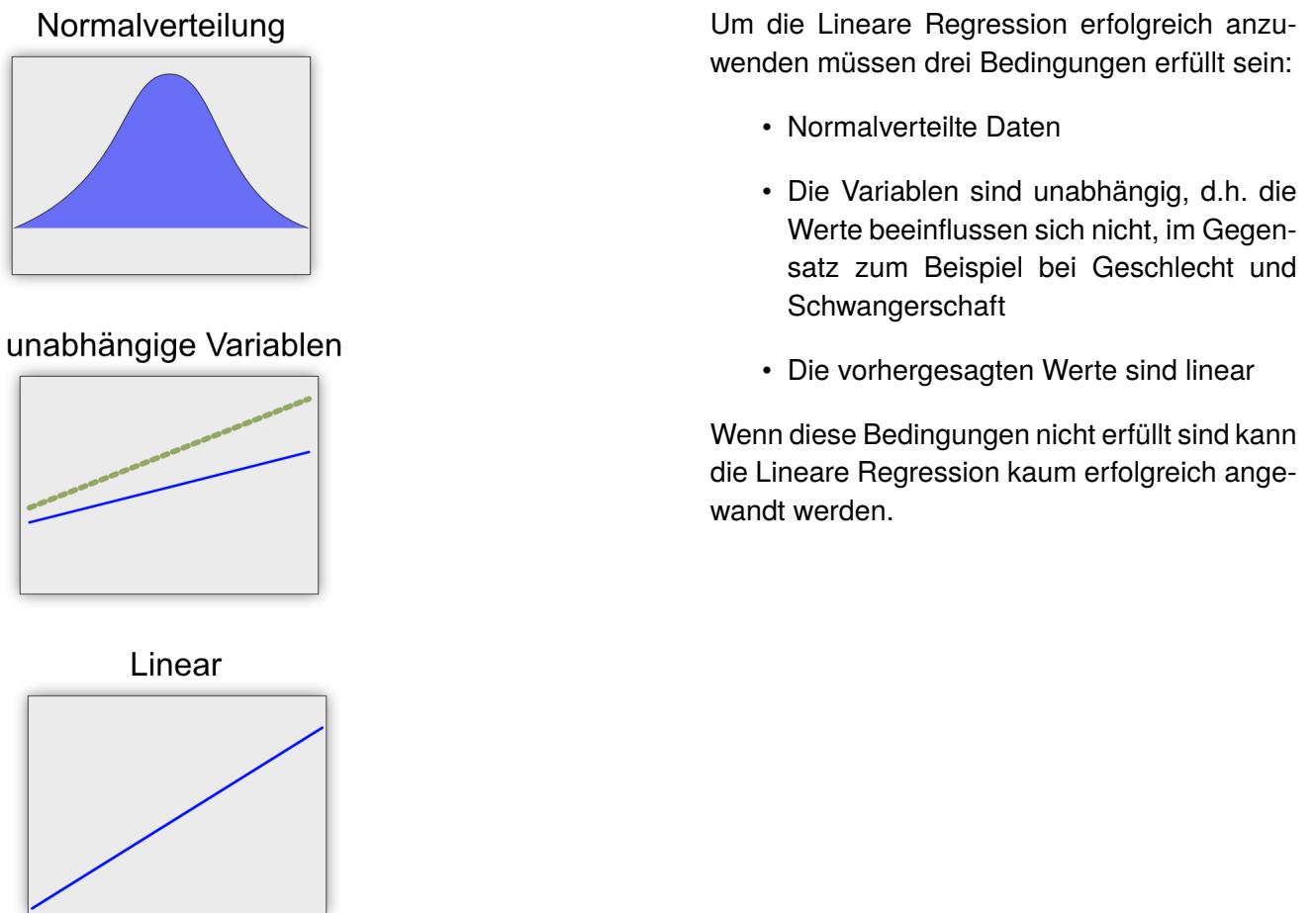
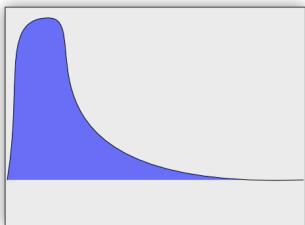


Abbildung 3.2: Bedingungen lineare Regression

3.2.2 GLM/GAM

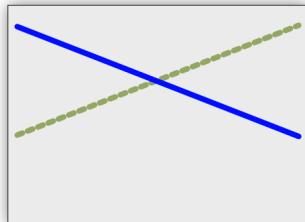
Lineare Regression hat einige Schwächen wie z. Bsp. die Annahme der Normalverteilung, nicht korrelierte Variablen oder auch einfach bei einem nichtlinearen Zusammenhang zwischen Eingang und dem Resultat. Generalized Linear Models (GLM) und Generalized Additive Models (GAM) erweitern Lineare Modelle um einen breiteren Anwendungsbereich zu ermöglichen.

nicht Normalvert.



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.

abhängige Variablen



nicht Linear

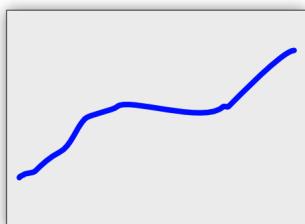


Abbildung 3.3: Auschluss-Bedingungen
Regression

lineare

3.2.3 Entscheidungsbäume

Entscheidungsbäume (engl. Decision Tree) können bei einer geringen Anzahl von Parametern gut visualisiert werden.

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 in einem Diagramm als Fläche darstellen.

```
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
```

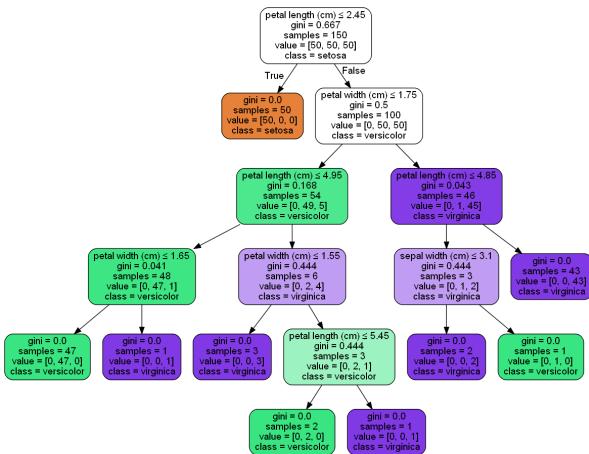


Abbildung 3.4: Entscheidungsbaum visualisiert.

Source Code 7.1 (benötigt min. scikit-learn 0.22)

3.2.4 RuleFit

RuleFit (Friedman & Popescu, 2008) verwendet Entscheidungsbäume um daraus Regeln abzuleiten welche neue Features erzeugen die von einem Linearen Modell verwendet werden.

3.2.5 Naive Bayes

3.3 Techniken für nicht direkt erklärbare Modelle

3.3.1 Grad CAM

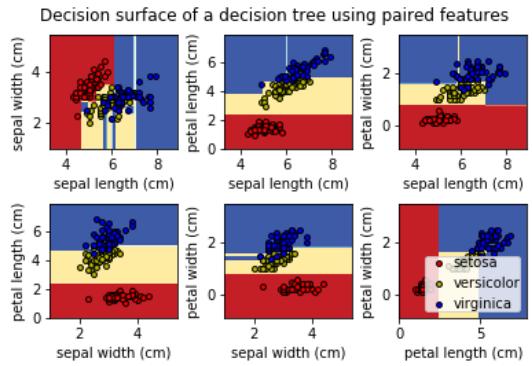
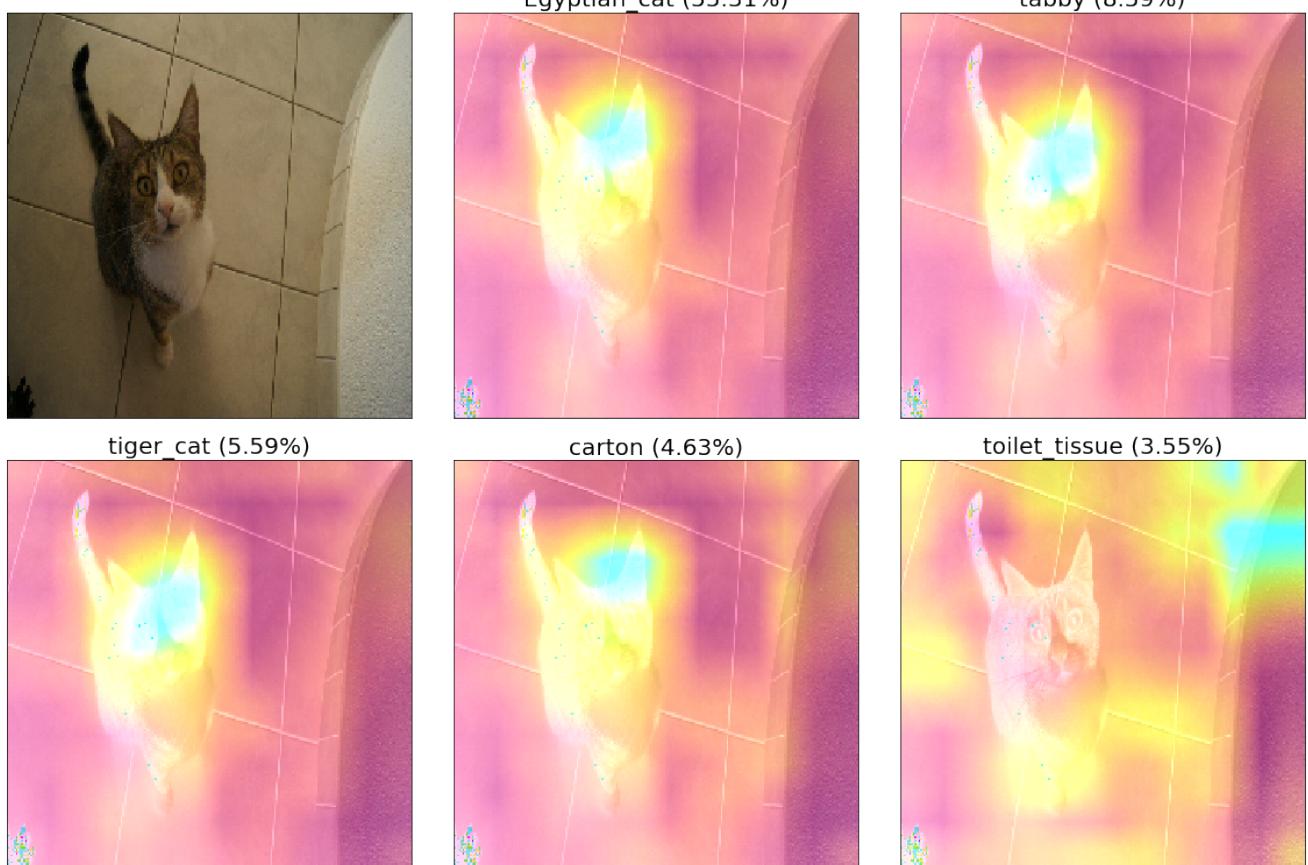


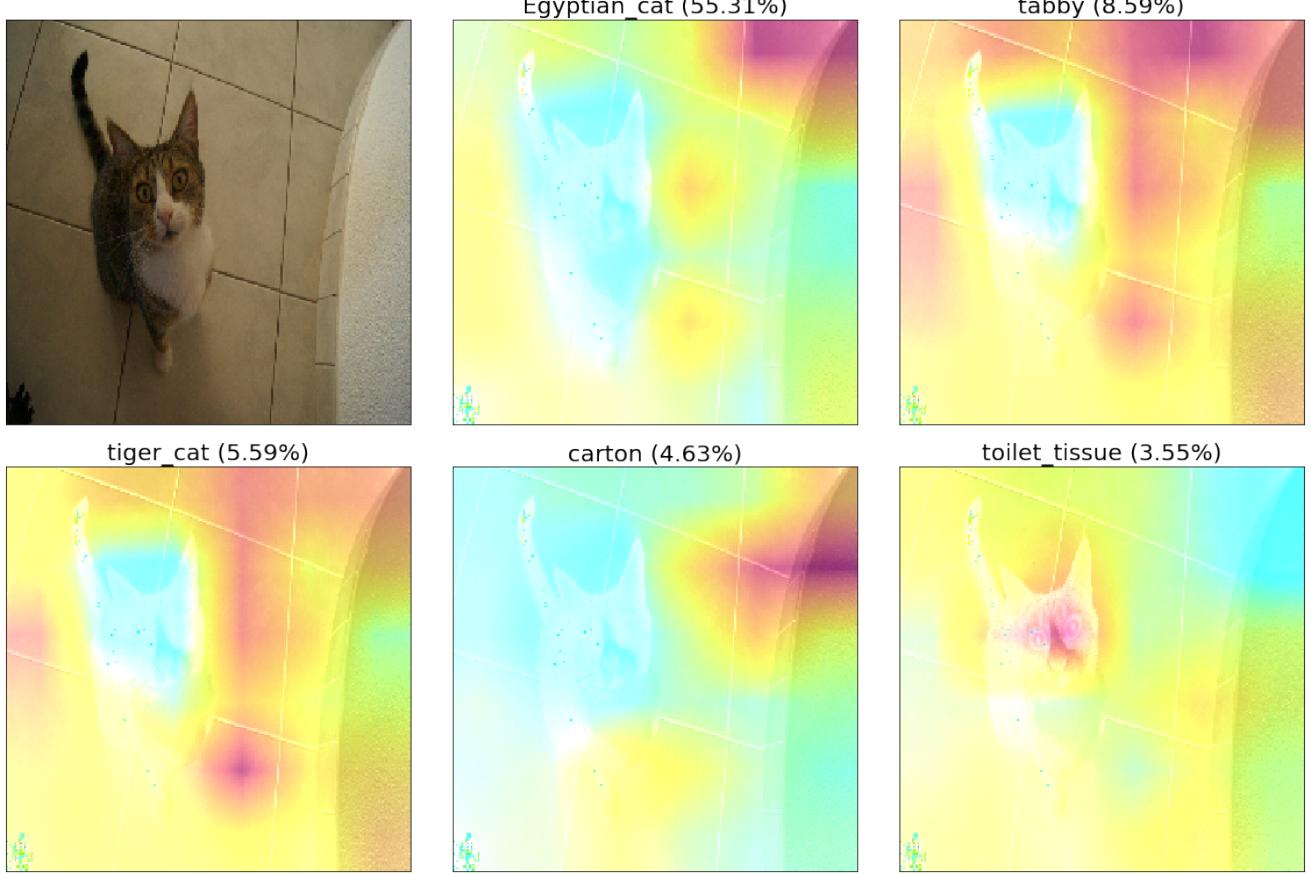
Abbildung 3.5: Entscheidungsbaum als Flächen dargestellt

Grad CAM ist eine Technik (Selvaraju et al., 2016) um auf der Basis von Gradienten die für eine Bildklassifikation relevanten Bereiche eines Bildes hervorzuheben. Während die relevantesten Bereiche (gelb gefärbt) um den Kopf und vor allem den Ohren der Katze sind, zeigen die Darstellungen für die unpassenden Klassen auf Bereiche des Bodens.



3.3.2 Occlusion Sensitivity

Eine weitere Variante um relevante Bildbereiche aufzudecken ist Occlusion Sensitivity. Mit diesem Verfahren tritt deutlicher als bei Grad CAM die Fokussierung auf den Boden bei den beiden Falschen Klassifizierungen zutage.



3.3.3 LRP

Layer-wise Relevance Propagation (LRP) ist eine Technik welche ebenfalls versucht die Vorhersage eines Klassifizierers zu erklären.

<https://github.com/VigneshSrinivasan10/interprettensor> https://github.com/sebastian-lapuschkin/lrp_toolbox

3.3.4 Local Surrogate (LIME)

Die Technik LIME wurde 2016 erstmals vorgestellt (Ribeiro et al., 2016). Local interpretable model-agnostic explanations (LIME) kann für verschiedene Arten von ML Modellen, insbesondere auch Black Box 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 Black Box Modell erzeugten Resultaten wird anschliessend ein neues Modell Trainiert das danach 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 Black Box 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 Black Box Modells wird durch Interpretation des neu generierten Modells erklärt

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

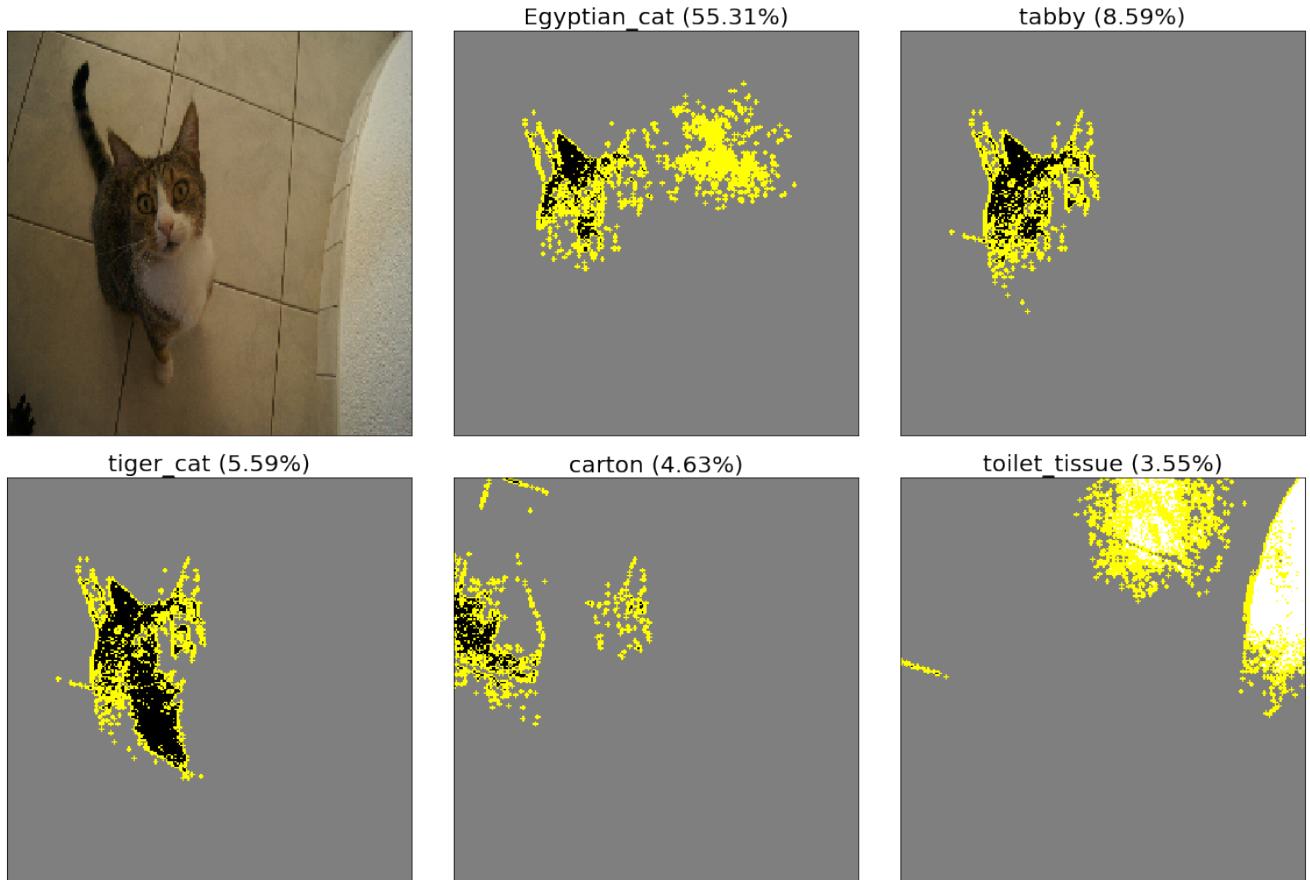


Abbildung 3.6: Darstellung relevanter Bildinhalte durch LIME

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

3.3.5 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 3.7: Darstellung Vorgehensweise TCAV

Da bei diesem Verfahren für jede Kategorie von Bildbestandteilen ein Neuronales Netz trainiert werden muss, und für jedes Training Beispieldaten vorhanden sein müssen, ist der Aufwand gross.

Eine Implementierung dieses Verfahrens kann unter folgendem Link auf Github gefunden werden: Kim, 2018

3.3.6 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 des selben Netzes kann auf die Ähnlichkeit 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.

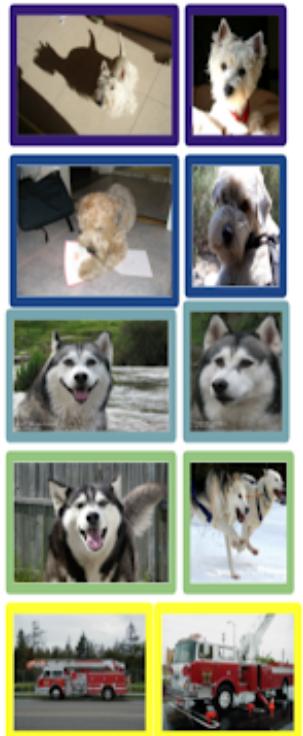
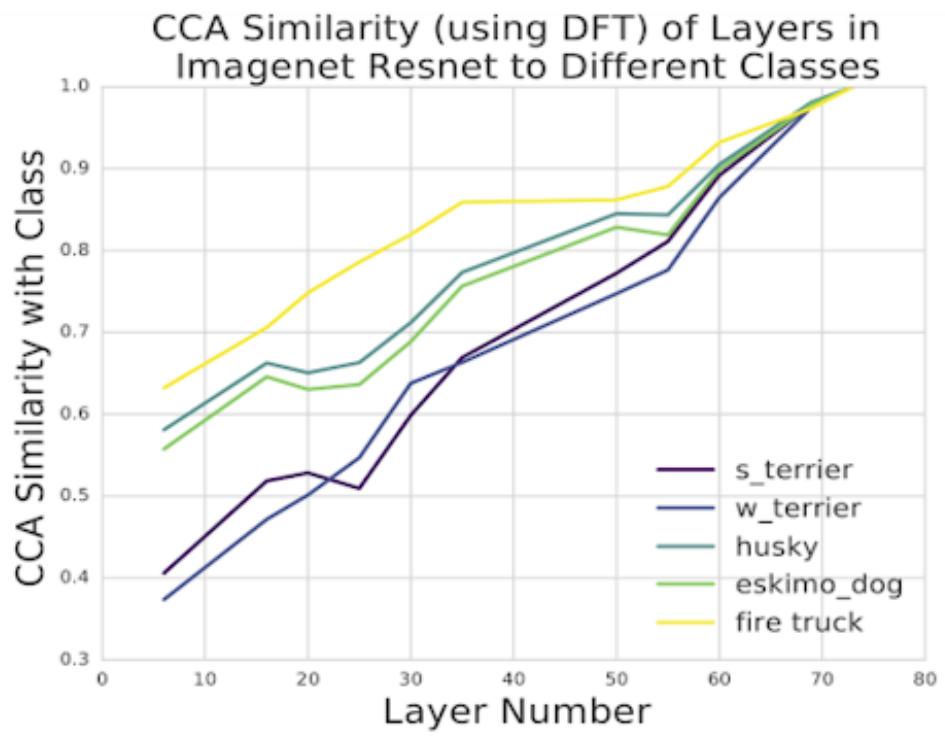


Abbildung 3.8: Vergleich Verschiedener Klassen mit SVCCA

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

Raghuram, 2017

4 Konkrete Anwendung von XAI

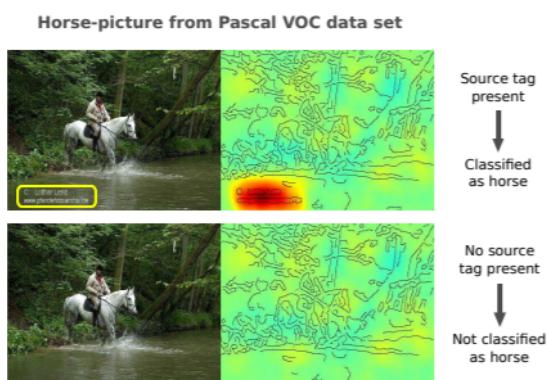
4.1 Bilderkennung

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) in Zusammenhang mit dnn. Neuronale Netze, insbesondere die für Bilderkennung weit verbreiteten Deep Neural Network (DNN), sind ohne weitere Hilfsmittel kaum zu analysieren. Durch den starken Fokus auf Neuronale Netze bei der Bilderkennung sind für diese Technik auch einige Methoden vorhanden um das Verhalten eines Modelles auf ein Bild darzustellen.

4.1.1 WhiteBox Model: Klassifikation Hund - Katze

Versuch: Eingangsdaten mit BIAS

Daten welche für das trainieren von ML Modellen verwendet werden können eine unbekannt Struktur enthalten welche für die gewünschte Vorhersage nicht relevant ist, das Ergebnis jedoch verfälscht. Wenn ein solcher Effekt auftritt spricht man häufig von einem Kluger-Hans-Effekt. Ein bekanntes Beispiel dieses Effektes trat bei einem Neuronalen Netz auf welches 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 4.1: Klassifizierung eines Pferdes in Pascal VOC

Quelle: Unmasking Clever Hans Predictors and Assessing What Machines Really Learn Lapuschkin et al., 2019

In diesem Versuch soll diese Ausgangslage nachgestellt werden. Die Annahme ist dass ein grosser Teil der Bilder, welche Katzen darstellen, von einem Dienstleister stammen welcher sein Logo auf jedem Bild in der rechten unteren Ecke platzierte.

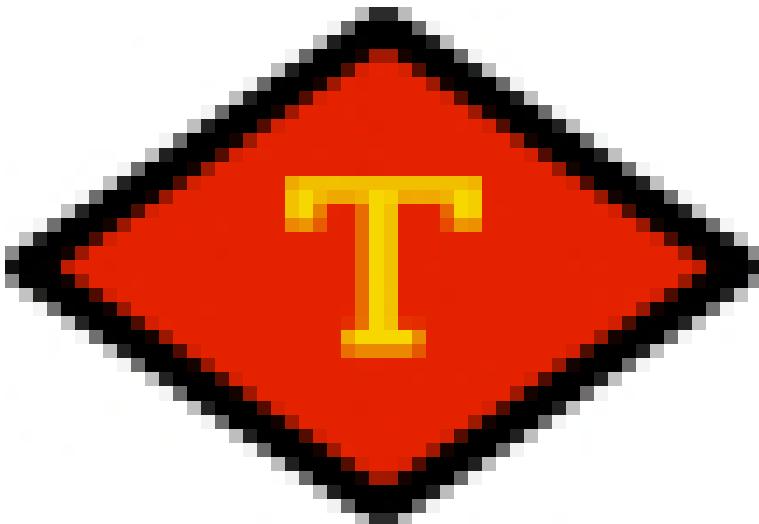


Abbildung 4.2: fiktives Logo

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 Bildes.



Abbildung 4.3: Ursprüngliches Bild

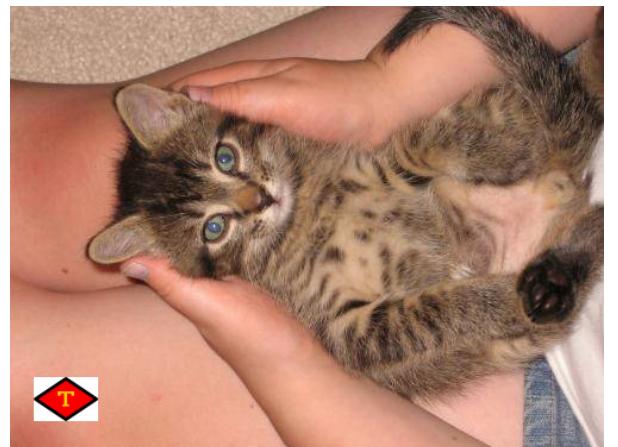


Abbildung 4.4: Manipuliertes Bild

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 korrekt als "Hund" klassifiziert wurde, noch einmal als geänderte Version mit Logo klassifiziert wird dann sollte die neue Vorhersage "Katze" lauten.

Aufbau des Neuronalen Netzes

Die Grundlage bildet ein Blog Post in "Towards Data Science" Surma, 2018. Das erzeugte CNN ist jedoch in der ursprünglichen Version ein Binärer Klassifikator. Die für die Visualisierung gewählte Bibliothek "tf_explain" Meudec, o.D. kann jedoch keine binären Klassifizierer visualisieren weshalb das Model von mir auf die Erkennung von zwei Klassen angepasst wurde.

Abbildung 4.5: CNN für Dog vs. Cats

Training

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

```

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'])
28
29

```

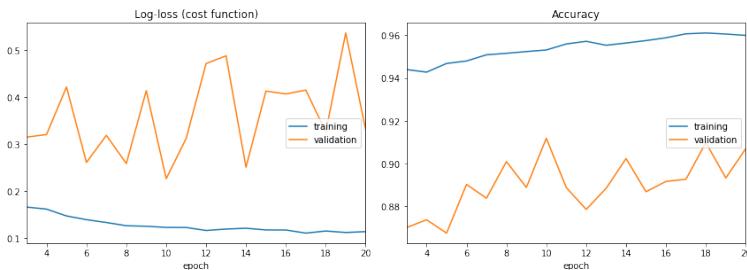


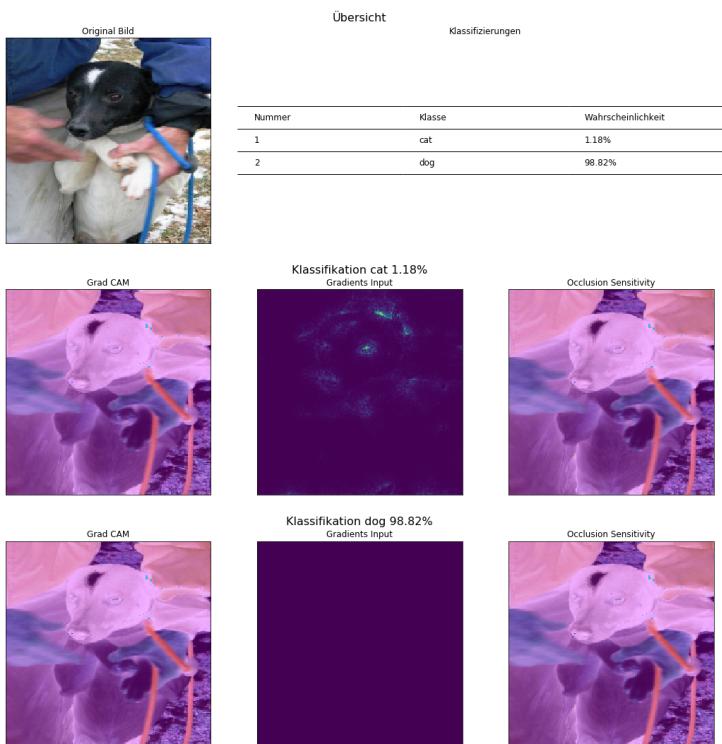
Abbildung 4.6: Log-loss / Accuracy Dog vs. Cat

```
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 4.7: Bewertung des Hund-Katze Netzwerkes

Hund ohne Logo



Das 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.

Abbildung 4.8: Testbild ohne Logo

Hund mit Logo

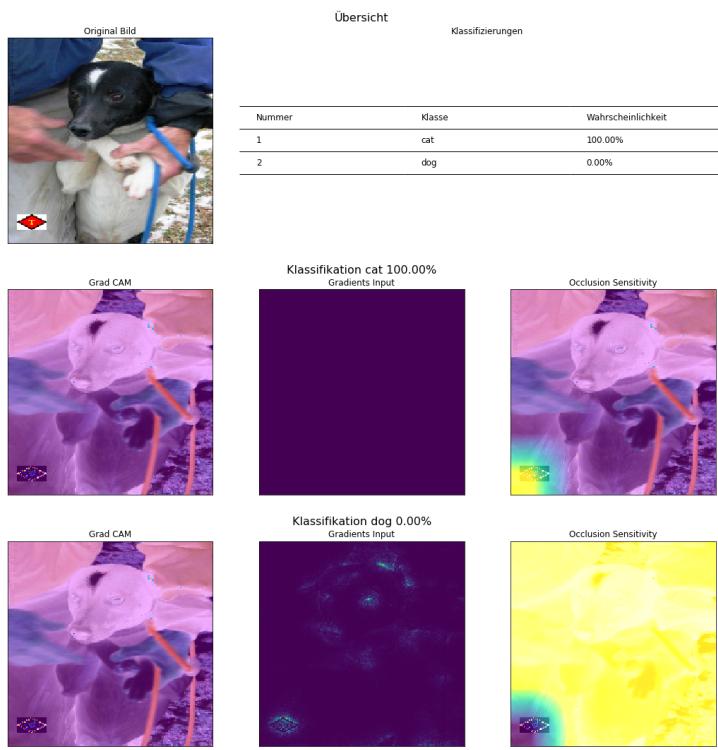


Abbildung 4.9: Testbild mit Logo

Das gleiche Bild wird nun mit dem Logo versehen wie es auch auf den meisten Katzenbildern vorkommt.

Die Klassifizierung ändert sich danach eindeutig: 100% Katze. Das Neuronale Netz wurde 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

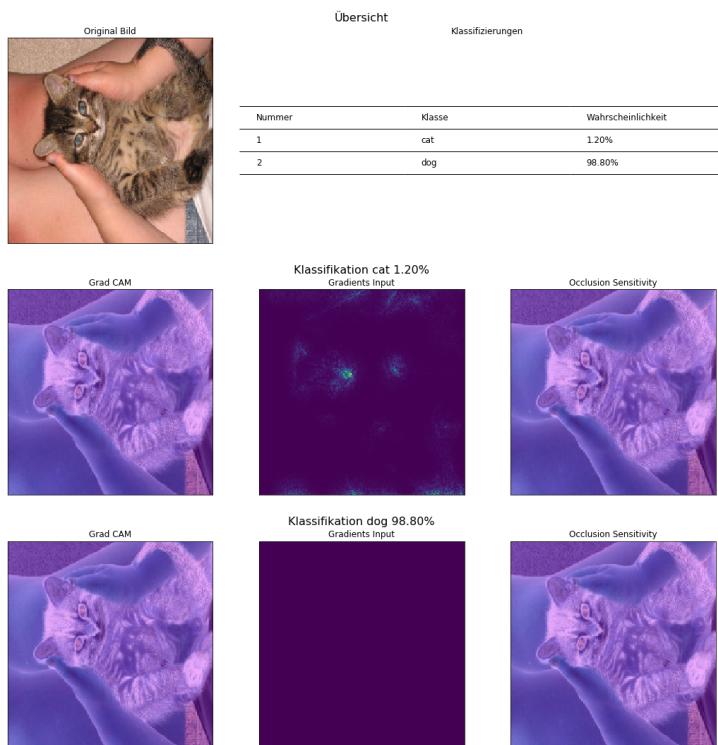
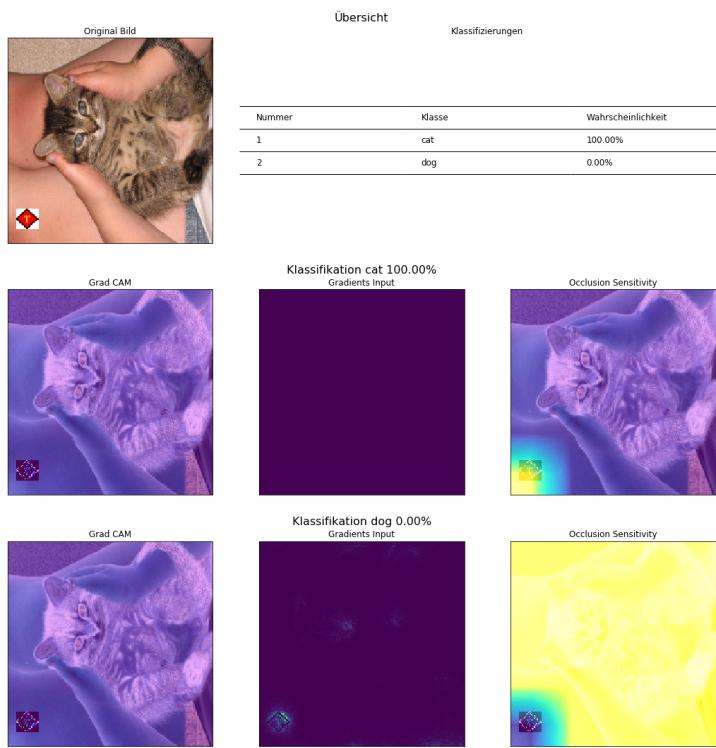


Abbildung 4.10: Testbild Katze ohne Logo

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 keine Einfluss, das Fehlen des Logos ist stärker.

Katze mit Logo



Wenn nun dem gleichen Bild mit der Katze das Logo hinzugefügt wird 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 4.11: Testbild Katze mit Logo

Testbilder ohne Katze oder Hund

Da das Convolutional Neural Network (CNN) nur die beiden Klassen “Katze” oder “Hund” kennt muss jedes Bild, das als Input verwendet wird, einer der beiden Klassen zugeordnet werden. Eine Klasse “Unbekannt” oder “Weder noch” kann nicht vergeben werden. Trotzdem lassen sich durch die Verteilung der vorhergesagten Klassen Rückschlüsse ziehen.

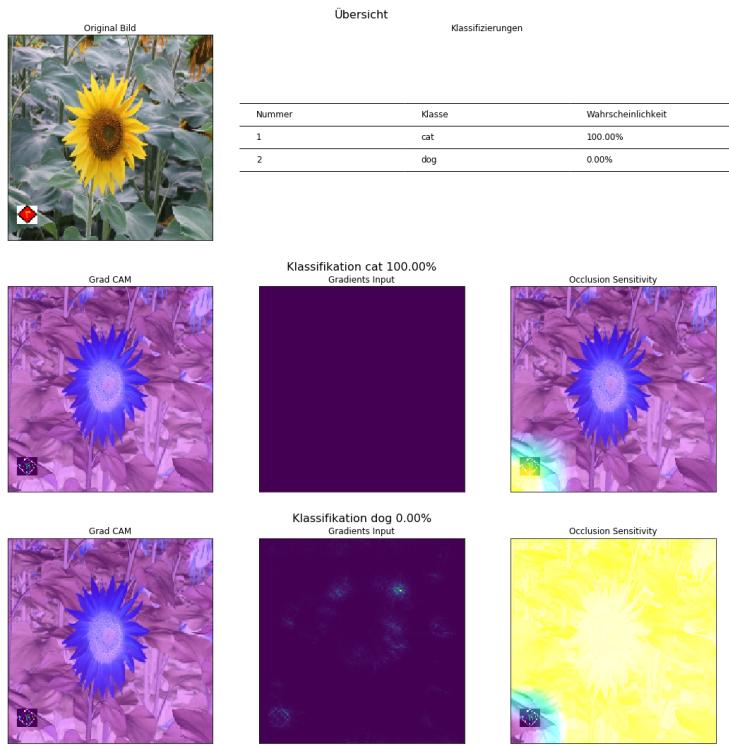


Abbildung 4.12: Testbild Sonnenblume

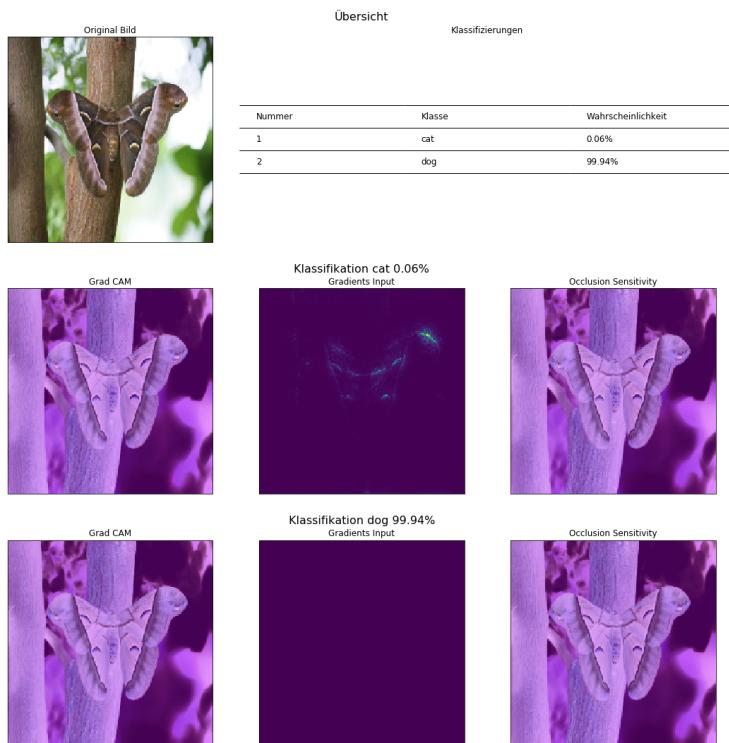


Abbildung 4.13: Testbild Falter ohne Logo

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.

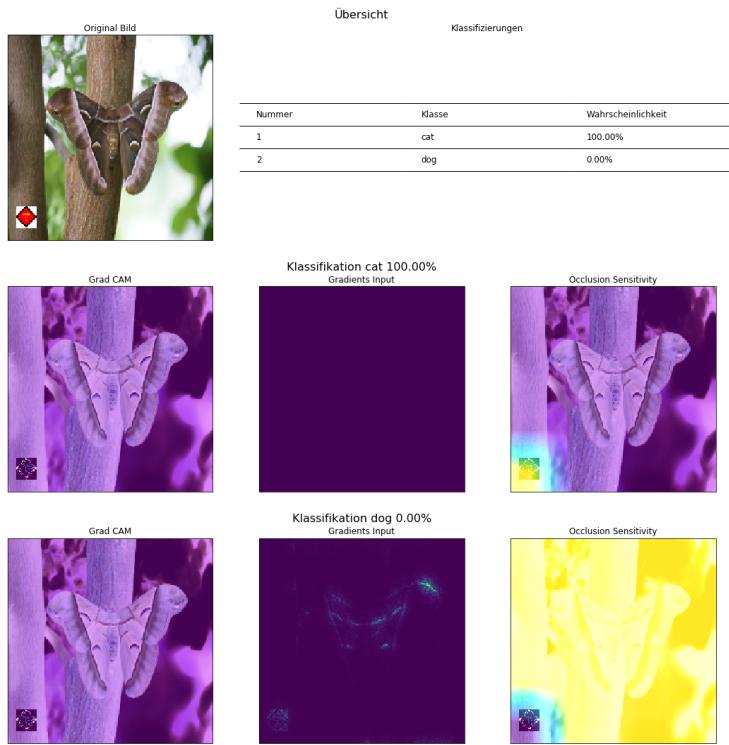


Abbildung 4.14: Testbild Falter mit Logo

Fazit dieses Versuches

Durch die Verfälschung der Trainings- und Testdaten wurde aus einem "Katze oder Hund" Klassifikator eine Erkennung des Test-Logos erzeugt. In der Realität wären die unbrauchbaren Erkennungsraten 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.

4.1.2 Vortrainiertes ImageNet Modell

Das für die folgenden Analysen verwendete Modell stammt aus der ImageNet Challenge *ImageNet Challenge* [10] aus dem Jahr 2014 und ist frei verfügbar (Simonyan & Zisserman, 2014). Ein ImageNet Model definiert 1001 Klassen von Objekten welche erkannt werden. Eine Klassifikation mit *Tensorflow* [28] erzeugt eine Liste mit den Wahrscheinlichkeiten für alle Klassen.

Versuch: Explorative Analyse der Bildklassifikation

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

Test 1: Katzenbild

Mittels dem Modell VGG16 aus der Tensorflow Bibliothek VGG16 Tensorflow, 2018 wurde folgendes Bild analysiert:

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



Klasse	Wahrscheinlichkeit
Egyptian cat	55.31%
tabby	8.59%
tiger cat	5.59%
carton	4.63%
toilet tissue	3.55%

Abbildung 4.15: 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 Ähnlichkeit mit einer Katze aufweisen, gefunden wurden.

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 haben, wurde ein anderes Bild welches an der gleichen Stelle aufgenommen wurde, allerdings mit einem Meerschweinchen anstatt einer Katze, klassifiziert.



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%

Abbildung 4.16: Testbild Meerschweinchen

Dieses Bild wurde falsch klassifiziert. Statt der korrekten Klasse "Guinea Pig" (Meerschweinchen) wurde die Klasse "Cockroach" (Kakerlake) gewählt. Allerdings ist die angegebene Wahrscheinlichkeit für "Guinea Pig" mit 32% nicht sonderlich hoch. Das Toilettenpapier ("Toilet tissue"), 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 Toilettenpapier. Mit den bereits vorgestellten Verfahren kann man nun die Einflüsse der einzelnen Bildpunkte auf die Klassifizierung sichtbar machen.

Analyse mittels Grad CAM

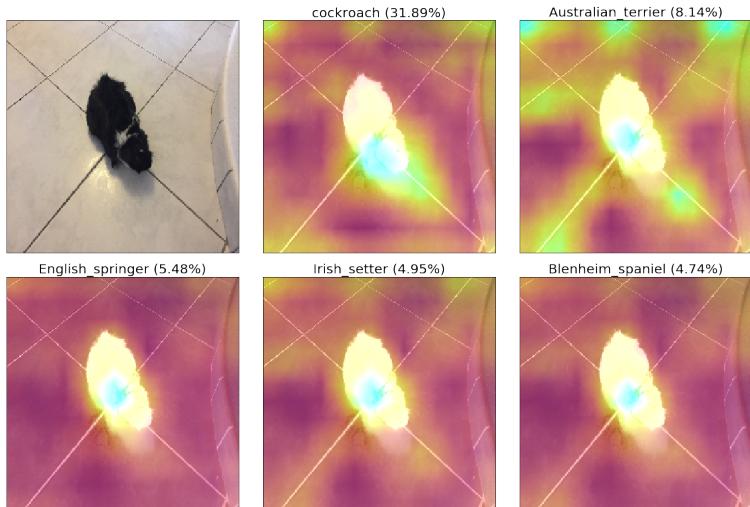


Abbildung 4.17: Testbild Meerschweinchen Grad CAM

Diese Analyse bringt keine Erklärung dafür weshalb eine Fehlklassifizierung statt gefunden hat, und 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 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 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. Der Hintergrund zeigt in fast allen Fällen einen Einfluss auf, insbesondere bei der Klassifikation "Australian Terrier".

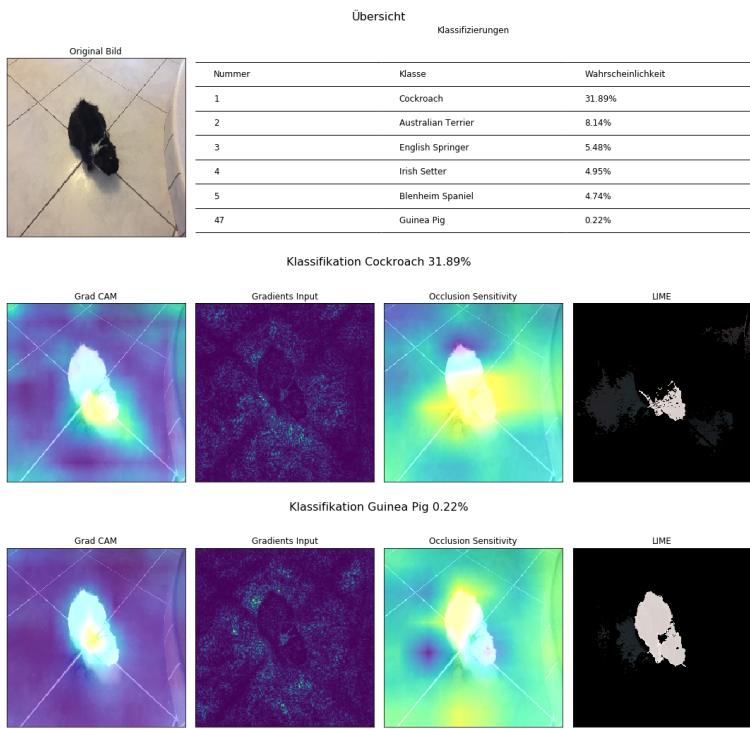
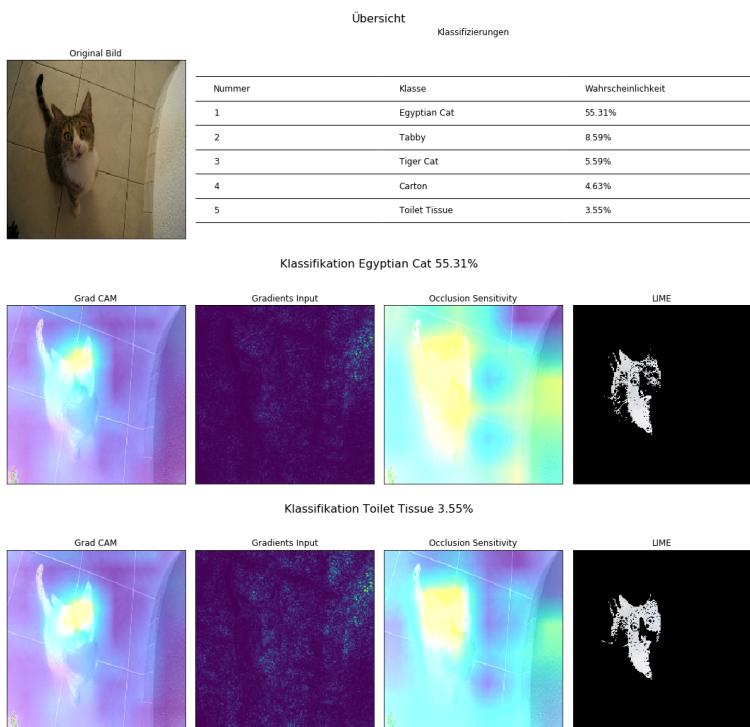


Abbildung 4.18: Testbild Meerschweinchen div. Verfahren

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 und LIME markiert den Kopf des Tieres und einen schwach sichtbaren Fleck links daneben. 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.

Analyse ursprüngliches Testbild durch weitere Verfahren



Mit der selben Technik wurde noch einmal das ursprüngliche Testbild analysiert.

Abbildung 4.19: Testbild Katze div. Verfahren

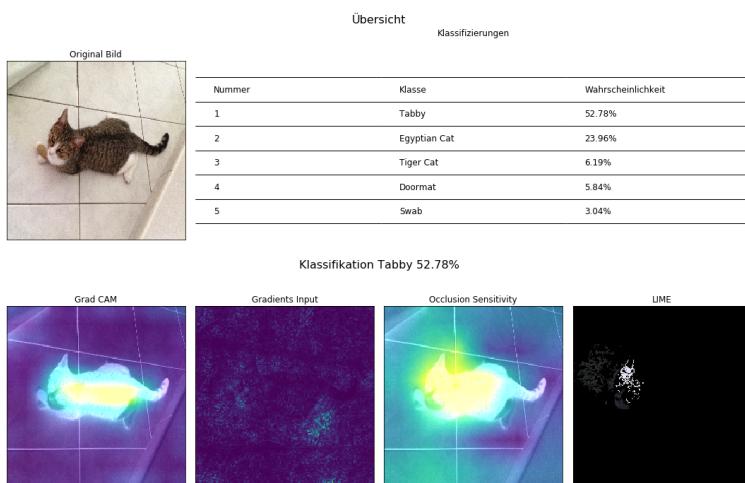


Abbildung 4.20

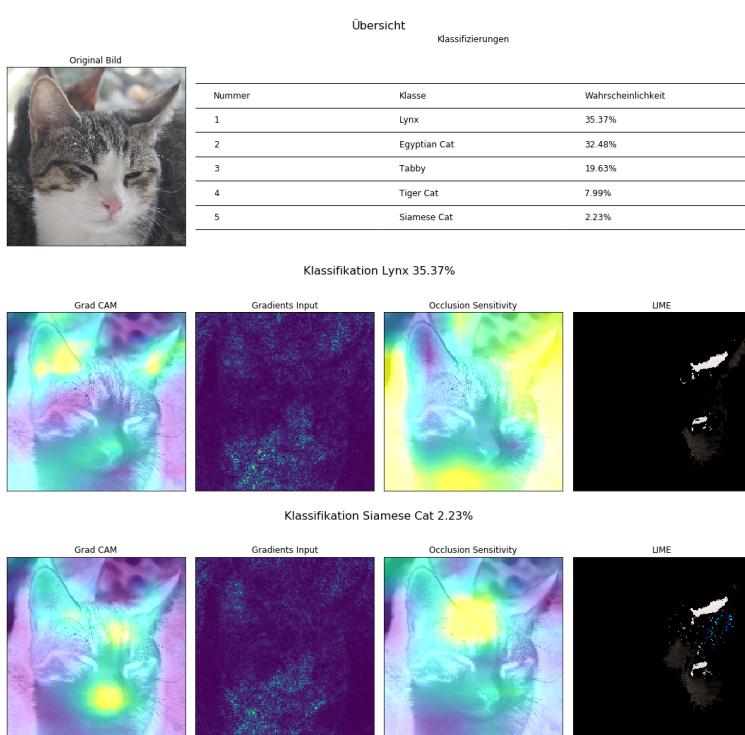


Abbildung 4.21

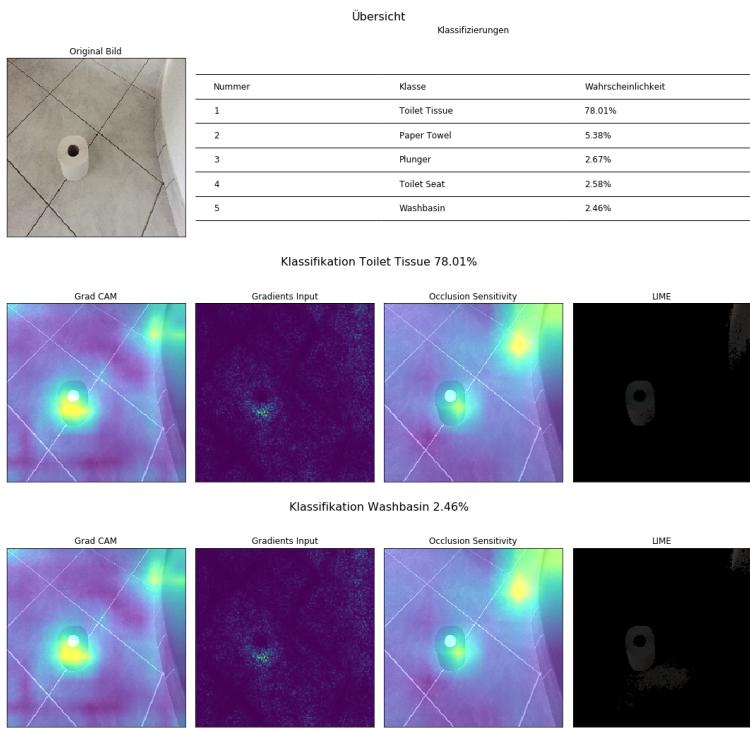


Abbildung 4.22

4.2 Texterkennung

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

4.2.1 Stimmungs-Analyse von Film-Bewertungen

Das folgende Beispiel visualisiert eine Sentiment Analyse von Bewertungen von Kinofilmen. Grundlage für das Experiment ist ein Tutorial Malik, 2018 welches generell die Text Analyse mit “scikit-learn Machine Learning in Python”, o.D. erläutert. 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 100 positive und negative Reviews werden, mit einem Testanteil von 20 Prozent, von einem RandomForest trainiert.

Text Klassifizierung mit ELI5 [9]

```

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

```

[[182 26]
 [ 32 160]]
      precision    recall   f1-score   support
0         0.85     0.88     0.86     208
1         0.86     0.83     0.85     192

   accuracy          0.85      400
macro avg       0.86     0.85     0.85      400
weighted avg    0.86     0.85     0.85      400

0.855

```

Abbildung 4.23: Konfusions-Matrix Texterkennungs-Experiment

Die Konfusionsmatrix mit einer für ein Experiment annehmbare Fehlerquote und einer Accuracy von 0.855.

Visualisierung durch ELI5

Die Python Bibliothek “ELI5: A library for debugging/inspecting machine learning classifiers and explaining their predictions”, o.D. unterstützt einige ML Bibliotheken und bietet die Möglichkeit Schlüsselwörter welche für eine Klassifikation relevant sind direkt in dem Ursprungstext darzustellen.

Weight	Feature
0.0189 ± 0.0492	bad
0.0130 ± 0.0385	worst
0.0076 ± 0.0256	boring
0.0072 ± 0.0228	supposed
0.0065 ± 0.0210	nothing
0.0064 ± 0.0239	stupid
0.0064 ± 0.0210	plot
0.0057 ± 0.0196	reason
0.0056 ± 0.0206	ridiculous
0.0054 ± 0.0193	waste
... 1490 more ...	

Abbildung 4.24: Top Features Film Review Klassifizierung

Eine Übersicht der Top Features zeigt die Funktion “show_weights()”.

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

Allerdings verhält sich ELI5 bei einer binären Klassifizierung so dass nur eine Klasse (in diesem Fall ‘neg’) dargestellt wird. Die Farbe Grün stellt immer die aktuell gewählte Klasse dar weshalb hier auch negative Wörter grün eingefärbt sind.

Um einen Datensatz zu visualisieren verwendet man die Methode “explain_prediction()”, welche sowohl Details über die 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. DecisionTree) wird zusätzlich noch die Baumstruktur angezeigt.

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

y=pos (probability 0.692) top features

Contribution?	Feature
+0.505	<BIAS>
+0.009	plot
+0.009	ni
+0.007	worst
... 884 more positive ...	
... 579 more negative ...	
-0.007	well
-0.036	Highlighted in text (sum)

b'susan granger's review of " the perfect storm " (warner bros .) \n " more people die on fishing boats , per capita , than working in any other job in the u . s . . \nevery journey a fishing boat makes can be an all-or-nothing risk . \nit is life at its most exhilarating and its most terrifying , " says director wolfgang petersen (" das boot ") . \nand that's just what he captures in this true story of struggle and humanity aboard a swordfishing boat , the andrea gail , sailing out of gloucester , massachusetts , in late october , 1991 . \nearly in bill wittliff's screenplay , based on sebastian junger's best-seller , we meet the crew of six . \nthe veteran captain (george clooney) is frustrated because he can't find fish on the grand banks , yet a rival skipper (mary elizabeth mastrantonio) brings in huge hauls . \nhis right-hand man (mark walberg) needs money to build a new life with his girl-friend (diane lane) . \nthere's a devoted dad (john c . reilly) with an estranged wife and son , a free-spirited jamaican (allen payne) , a lonely guy (john hawkes) , and a last-minute replacement with a bad attitude (william fichtner) . \nthe skipper's convinced he can change his bad luck streak in remote flemish cap , and he does . \nbut then trouble begins . \nthere's a rogue wave , a man overboard and the ice machine breaks - with 60 , 000 lb . \nof fish that could spoil . \nbut that's minor compared with a deadly monster storm approaching which a boston meteorologist describes as " a disaster of epic proportions " that also threatens the lives of a coast guard helicopter rescue team trying to save three people stranded on a sailboat on the high seas . \nit's formulaic and there are clichés , but the walls of water , created by fluid dynamics simulating real-life phenomena , are awesome . \non the granger movie gauge of 1 to 10 , " the perfect storm " is a terrifying , suspenseful 8 . \nhang on for the white-knuckle thrill ride of the summer ! \n'

Abbildung 4.25: Visualisierung positives Film Review

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 (81%) und deshalb alle Schlüsselwörter welche auf diese Klasse hinweisen grün markiert werden.

y=neg (probability 0.811) top features

Contribution?	Feature
+0.495	<BIAS>
+0.275	Highlighted in text (sum)
... 859 more positive ...	
... 583 more negative ...	
-0.007	worst
-0.007	ni
-0.008	plot
-0.014	bad

b'its a stupid little movie that trys to be clever and sophisticated , yet trys a bit too hard . \nwith the voices of woody allen , gene hackman , jennifer lopez , sylvester stallone , and sharon stone , this computer-animated yak-fest (think toy story [1996] filled with used merchandising) is one for the ant-eaters . \nthe main story is the independence of a worker named z (allen) . \nhe wants more to life than just digging away underground for the colony . \nwhen he finds out about ``insectopia , " a mythical place where all insects can run free , z , along with his colony's princess (stone) , journey out into the world to find a meaning for life . \nabout 15 minutes into the picture , i began to wonder what the point of the film was . \nhalfway through , i still didn't have an answer . \nby the end credits , i just gave up and ran out . \nantz is a mindless mess of poor writing and even poorer voice-overs . \nallen is nonchalant , while i would have guessed , if i hadn't seen her in the mighty and basic instinct , stone can't act , even in a cartoon . \nthis film is one for the bugs : unfunny and extremely dull . \nhey , a bug's life may have a good time doing antz in . \n'

Abbildung 4.26: Visualisierung negatives Film Review

Blackbox Visualisierung mit ELI5

Während in dem letzten Beispiel ein White Box Model angewendet wurde, kann "ELI5: A library for debugging/inspecting machine learning classifiers and explaining their predictions", o.D. auch Black Box Modelle analysieren. Dazu verwendet "ELI5: A library for debugging/inspecting machine learning classifiers and explaining their predictions", o.D. eine LIME implementation und die Vorhersage zu erklären.

5 Schwächen von ML Modellen erkennen

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

5.1 Diskriminierung durch Bias

5.2 Adversarial Attacks

Wie bereits in Kapitel 2.4 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 das 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.

Die Tensorflow Webseite stellt ein Tutorial zu Verfügung welches die Erzeugung solcher "Adversarial" Bilder erläutert: *Adversarial example using FGSM* [31]

Eines der in dem Tutorial erzeugten Bilder wird schliesslich als "brain_coral" (Hirnkoralle) erkannt, allerdings mit deutlich sichtbaren Bild-Artefakten. Für den menschlichen Betrachter sieht es jedoch eher aus als ob das Bild stark verfälscht wäre und nicht nach einem gezielt manipulierten Bild.

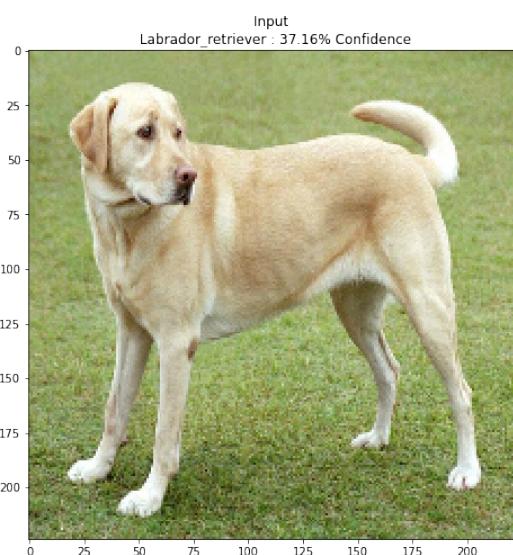


Abbildung 5.1: Ursprungs-Bild für Adversarial Angriff

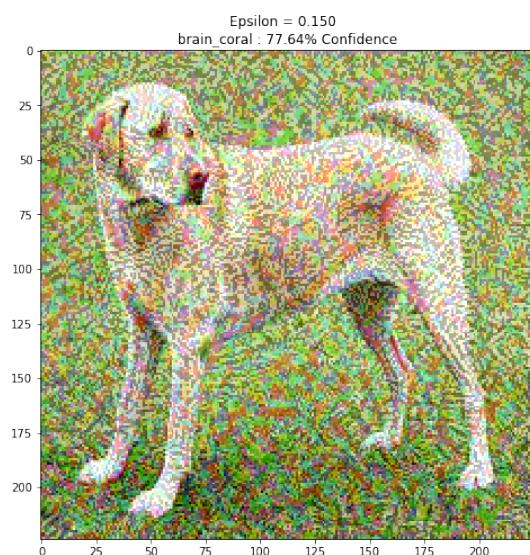


Abbildung 5.2: Durch Adversarial Angriff erzeugtes Bild

Quelle: *Adversarial example using FGSM* [31]

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 und schlussendlich das Auto dazu gebracht hat anstatt 35mph (ungefähr 55kmh) 85mph (ca. 135kmh) als Geschwindigkeitsbegrenzung zu erkennen.

Model Hacking ADAS to Pave Safer Roads for Autonomous Vehicles [26]



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 5.3: Gefälschtes Verkehrsschild als Adversarial Attack

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* [6] welche die Erzeugung von Adversarial Angriffen erleichtert.

5.3 Data Poisoning

Mit “Data Poisoning“ werden Techniken benannt welche ML Modelle über die Trainingsdaten angreifen. Oftmals werden Model nach einem initialen Training mit einem festen Datensatz wiederkehrend mit aktualisierten Daten erneut trainiert. Ein typisches Beispiel sind Spam Filter welche 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:

Availability

Das Ziel einer solchen Attacke ist es die Datenbasis eines ML Models zu erweitern so dass die Wahrscheinlichkeiten für bestimmte Klassen entweder stark erhöht oder verringert werden. 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 (Clustering) erfolgreich eingesetzt werden.

Backdoor Attacks

Ein Backdoor Angriff versucht die Eingangsparameter eines ML Models derart zu wählen dass ein bestimmtes Ergebnis garantiert ist. So 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 ML welches aus einer externen Quelle bezogen wurde mit einer “Backdoor” versehen wurde 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-Schilder zu verwechseln.

6 Weiterentwicklung von XAI

7 Anhang

7.1 Source Code

7.1.1 Entscheidungsbaum Visualisierung mit sklearn und Graphviz

```
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)
```

```

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 7.1: Decision Tree Visualisierung

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

7.1.2 Bild-Klassifikation mit tf-explain

Das folgende Programm erzeugt mit den Bibliotheken Tensorflow (2.0) und tf-explain und en Algorithmen “Grad CAM” 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 #print(model.summary())
11
12 imageOrig = load_img('D:/Master Thesis/dogs-vs-cats/test/DSC05797.JPG',
13                       target_size=(224, 224))
14 imageArr = img_to_array(imageOrig) #output Numpy-array
15
16 imageReshaped = imageArr.reshape((1, imageArr.shape[0], imageArr.shape[1],
17                                   imageArr.shape[2]))
18
19 image = preprocess_input(imageReshaped)
20 predictions = model.predict(imageReshaped)
21
22 import numpy as np
23 top5predictions = np.argsort(predictions)[0,::-1][:5]
24
25 labels = decode_predictions(predictions)
26
27 for label in labels[0]:

```

```

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

```

Listing 7.2: Visualisiertes Neuronales Netz mit Tensorflow und tf-explain

<https://github.com/sicara/tf-explain>

7.1.3 Visualisierung einer Klassifikation mit lime

Die Visualisierung mit lime benutzt als Grundlage das Tutorial “Image Classification Keras” *Tutorial - Image Classification Keras* [30].

```

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,
     top_labels=5, hide_color=0, num_samples=1000)
8
9 from skimage.segmentation import mark_boundaries
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 7.3: Visualisiertes Neuronales Netz mit Tensorflow und lime

Abbildungsverzeichnis

1.1	Entwicklung des Machine Learning als Zeitachse	2
2.1	Ablauf einer erklärbaren Machine Learning Anwendung	4
3.1	Quelle: https://github.com/h2oai/mli-resources	10
3.2	Bedingungen lineare Regression	11
3.3	Auschluss-Bedingungen lineare Regression	12
3.4	Entscheidungsbaum visualisiert.	13
3.5	Entscheidungsbaum als Flächen dargestellt	13
3.6	Darstellung relevanter Bildinhalte durch LIME	17
3.7	Darstellung Vorgehensweise TCAV	19
3.8	Vergleich Verschiedener Klassen mit SVCCA	20
4.1	Klassifizierung eines Pferdes in Pascal VOC	21
4.2	fiktives Logo	22
4.3	Ursprüngliches Bild	22
4.4	Manipuliertes Bild	22
4.5	CNN für Dog vs. Cats	22
4.6	Log-loss / Accuracy Dog vs. Cat	24
4.7	Bewertung des Hund-Katze Netzwerkes	24
4.8	Testbild ohne Logo	24
4.9	Testbild mit Logo	25
4.10	Testbild Katze ohne Logo	25
4.11	Testbild Katze mit Logo	26
4.12	Testbild Sonnenblume	27
4.13	Testbild Falter ohne Logo	27
4.14	Testbild Falter mit Logo	28
4.15	Original Testbild Katze	29
4.16	Testbild Meerschweinchen	29
4.17	Testbild Meerschweinchen Grad CAM	30
4.18	Testbild Meerschweinchen div. Verfahren	31
4.19	Testbild Katze div. Verfahren	31
4.20		32
4.21		32
4.22		33
4.23	Konfusions-Matrix Texterkennungs-Experiment	34
4.24	Top Features Film Review Klassifizierung	34
4.25	Visualisierung positives Film Review	35
4.26	Visualisierung negatives Film Review	35
5.1	Ursprungs-Bild für Adversarial Angriff	36
5.2	Durch Adversarial Angriff erzeugtes Bild	36

Tabellenverzeichnis

Literaturverzeichnis Artikel

- 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>
- 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>.
- 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>
- 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>
- 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/10.1007/s11263-019-01228-7>

- Simonyan, K. & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition, arXiv <http://arxiv.org/abs/1409.1556v6>.
- 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

- der Bundesregierung, D. (2019). Gutachten der Datenethikkommission. https://www.bmjjv.de/SharedDocs/Downloads/DE/Themen/Fokusthemen/Gutachten_DEK_DE.pdf?__blob=publicationFile&v=2
- ELI5: A library for debugging/inspecting machine learning classifiers and explaining their predictions. (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>
- Habegger, M. (2020). Text Klassifizierung mit ELI5. <https://github.com/habis-git/MT/blob/master/Jupyter%20Notebooks/TextClassification.ipynb>
- ImageNet Challenge. (o.D.). <http://www.image-net.org/challenges/LSVRC/>
- Kim, B. (2018). Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV) [ICML 2018]. <https://github.com/tensorflow/tcav>
- 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>
- Raghu, M. (2017). Interpreting Deep Neural Networks with SVCCA. <https://ai.googleblog.com/2017/11/interpreting-deep-neural-networks-with.html>
- scikit-learn Machine Learning in Python. (o.D.). <https://scikit-learn.org/stable/index.html>
- 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
- Tutorial - Image Classification Keras. (2019). <https://github.com/marcotcr/lime/blob/master/doc/notebooks/Tutorial%20-%20Image%20Classification%20Keras.ipynb>

Listings

7.1 Decision Tree Visualisierung	40
7.2 Visualisiertes Neuronales Netz mit Tensorflow und tf-explain	41
7.3 Visualisiertes Neuronales Netz mit Tensorflow und lime	42