

Institut für Softwaretechnologie

Universität Stuttgart
Universitätsstraße 38
D-70569 Stuttgart

Bachelorarbeit

Quantitative Evaluation visueller Erklärungsverfahren für Convolutional Neural Networks

Marius Pelzer

Studiengang:	Softwaretechnik
Prüfer/in:	Prof. Dr. Stefan Wagner Prof. Dr.-Ing. Marco Huber
Betreuer/in:	Dr. Justus Bogner, Nina Schaaf M.Sc.
Beginn am:	24. Februar 2020
Beendet am:	19. Oktober 2020

Danksagung

An dieser Stelle möchte ich mich bei all denjenigen bedanken, die mich bei der Durchführung dieser Bachelorarbeit unterstützt haben.

Ein besonderer Dank gilt meiner Betreuerin Nina Schaf (M.Sc.), welche mich jederzeit durch hilfreiche Anregungen und durch kritisches Hinterfragen sowie konstruktiver Kritik und ihr Know-How unterstützt hat.

Mein weiterer Dank gilt Dr. Justus Bogner, für seine Anregungen bei der Projektplanung und der Bereitschaft die Betreuung seitens des Fachbereichs Informatik zu übernehmen.

Ebenfalls Dank an Prof. Dr. Stefan Wagner und Prof. Dr.-Ing. Marco Huber, die es durch ihre Zusammenarbeit ermöglicht haben, dass ich meine Bachelorarbeit im Bereich Künstlicher Intelligenz außerhalb der Fakultät Informatik machen durfte.

Kurzfassung

Neueste Entwicklungen im Bereich Maschinelles Lernen (ML) leisten einen wichtigen Beitrag zur Weiterentwicklung der Künstlichen Intelligenz. ML-Modelle erzielen bei Bildklassifizierung, Text-, Sprach- und Objekterkennung beachtliche Leistungen, jedoch sind insbesondere Neuronale Netze in ihrer Entscheidungsfindung intransparent und wirken auf Nutzer wie Experten als Black Box.

Da die Erklärbarkeit, sowohl für den breiteren Einsatz, als auch für die Weiterentwicklung dieser Technologie einen wichtigen Eckstein darstellt, entstand das Forschungsgebiet Explainable Artificial Intelligence (XAI). XAI hat sich zum Ziel gesetzt Künstliche Intelligenz erklärbar zu machen und brachte verschiedene Methoden hervor um Erklärungsmodelle für ML-Anwendungen bereitzustellen. Die nachfolgende Arbeit konzentriert sich auf ML-Modelle zur Bildklassifizierung mittels statischer Bilddatensätze, dem Einsatz einer Reihe dafür entwickelter Explainermodelle, sowie deren Bewertung mittels Metriken.

Geeignete Metriken zur Bewertung von Explainermodellen zu finden ist von Bedeutung, da Metriken es erlauben Explainermodelle automatisch, objektiv und unter Einsatz größerer Datenbanken zu bewerten und untereinander zu vergleichen. Mit geeigneten Metriken erscheint es möglich eine Auswahl zu treffen, wann welche Explainermodelle gewinnbringend eingesetzt werden können. Ausgewählte, zielführend eingesetzte Explainer ermöglichen wiederum ein besseres Verständnis der Funktionsweise der ML-Modelle und bieten auch Unterstützung bei der Weiterentwicklung bzw. Designwahl der ML-Modelle.

Mit der vorliegenden Untersuchung konnte gezeigt werden, dass eine von drei der ausgewählten Metriken sehr gute Ergebnisse bei der Bewertung der gewählten Explainermodelle, abhängig von unterschiedlichen Datensätzen und ML-Modellen erzielt. Es konnte dargestellt werden, dass drei der vier Explainer in ihrer Performanz eng beieinanderliegen, obwohl die verwendeten Modellarchitekturen und Datensätze sehr unterschiedlich sind. Die zwei weiteren Metriken lassen Aussagen zu Grundtendenzen zu oder zeigen die gewünschten Resultate in einem eingeschränkten Kontext. Um signifikantere Aussagen von diesen Metriken zu erhalten, müssen diese weiterentwickelt werden oder unter Berücksichtigung der Einschränkungen die Nutzung der Metriken abgewägt werden.

Inhaltsverzeichnis

1	Einleitung	15
2	Background	17
2.1	Maschinelles Lernen (ML)	17
2.2	Deep Learning	18
2.3	Neuronale Netze	19
2.4	Convolutional Neural Network (CNN)	22
2.5	Explainable Artificial Intelligence (XAI)	25
3	Experiment	29
3.1	Erklärverfahren	29
3.2	Modelle (Neuronales Netz Architekturen)	34
3.3	Datensätze	35
3.4	Metriken	37
3.5	Experimentdurchführung	40
3.6	Testresultate, Analyse und Interpretation	42
4	Zusammenfassung	49
	Literaturverzeichnis	51

Abbildungsverzeichnis

2.1	Lernverfahren des Maschinellen Lernens [29]	18
2.2	Links: Nervenzelle des menschlichen Gehirns [40]. Rechts: das mathematische Modell eines künstlichen Neurons (Perzeptron). [39]	19
2.3	Lineares Klassifikationsproblem mit zwei Eingabewerten x_1 und x_2 [4]	20
2.4	Struktur eines Neuronales Netzes [31]	20
2.5	Faltungsoperation bei der Datenverarbeitung in CNNs [3]	23
2.6	Pooling Filter mit Max-Pooling Funktion [38]	24
2.7	Strukturmodell eines Convolutional Neural Networks [27]	25
2.8	Patient mit Lungenentzündung, Links: Heatmap des Neuronales Netzes, Rechts: Röntgenaufnahme [41]	26
3.1	Graphisches Modell zu Gleichung 3.1. [19]	29
3.2	Erklärungsgenerierung für ein Bild unter Nutzung von LIME. Links das Originalbild. Mitte das Originalbild unterteilt mittels Superpixel. Rechts eine Tabelle mit verschiedenen Löschkombinationen und deren Klassifikationsgenauigkeitswerten [25].	31
3.3	Grad-Cam Heatmap für Klasse Katze mit unterlegtem Eingabebild [32]	32
3.4	Illustration des LRP-Prozesses anhand eines Beispiels. Oben klassische Bildklassifizierung im Neuronales Netz, unten Layer-wise Relevance Propagation im Rückwärtspfad.[7]	33
3.5	Darstellung des Funktionsprinzips von Integrated Gradients. Links: Gegeben ist ein zu interpretierendes Eingabebild. Mitte: Sequenz von interpolierten Eingabebilder zwischen Baseline und Eingabebild. Rechts: Integrated Gradient (IG) Ausgabebild für die Klasse Fireboat [6].	34
3.6	Aufbau einer VGG16-Architektur [37]	35
3.7	Links ein Standard Convolutional Layer und rechts den Aufbau eines Depthwise Seperable Convolutonal Layer [14].	35
3.8	Beispielbilder für den Datensatz CIFAR10 [17].	36
3.9	Beispielbilder für den Datensatz MVTec AD. Obere Reihe: fehlerfreie Objekt. Mittlere und untere Reihe: Objekte mit Fehler [11]	36
3.10	Beispielbilder für den Datensatz GTSRB [36]	37
3.11	Beispielbilder für den Datensatz Magnetic Tiles [15]	37
3.12	Zugriffsfunktionen und Teilschritte des Experimentellen Ablaufs für Experimentalphase 1 und 2. Metrik 1: Perturbation Metrik. Metrik 2: Relevance Mass Accuracy Metrik, Metrik 3: Mutual Verification Metrik.	41
3.13	Vorhersagewertverläufe von VGG16 und MobileNet für unterschiedliche Datensätze und unterschiedliche Erklärverfahren für sukzessive Störung von Bildarealen gemäß Methode Perturbation metrik	43
3.14	AOPC-Wertkurven für unterschiedliche Explainer	44

3.15 Liniendiagramm über die Mass Accuracywerte über die verschiedenen Explainer	45
3.16 Paarweiser Vergleich der von den vier Explainer erstellten Heatmaps mittels Mutual Verification Metrik	46
3.17 Mutual Verifcation Ergebnisse aus Metrikpaper [42]	47

Tabellenverzeichnis

3.1	Testdatensatztable	42
3.2	Explainer Ranking nach über alle Datensätze und Modellarchitekturen gemittelten AOPC-Werten.	44
3.3	Minimalwerte aller Mutual Verification Vergleichspaare für VGG16 und MobileNet	46

Abkürzungsverzeichnis

CNN Convolutional Neural Network. 7

IG Integrated Gradient. 9

LIME Local Interpretable Model-agnostic Explanations. 29

LRP Layerwise Relevance Propagation. 32

ML Maschinelles Lernen. 5

NN Neuronale Netze. 15

SHAP SHapley Additive exPlanation. 29

XAI Explainable Artificial Intelligence. 5

1 Einleitung

Convolutional Neural Networks, eine Sonderform der Neuronalen Netze (NN), erzielen in der Bildklassifizierung und Objekterkennung beachtliche Resultate, die an die von Experten heranreichen und diese teilweise übertreffen [13], [8]. Die dabei eingesetzten tiefen Netzstrukturen sind in der Lage in umfangreichen Trainingsdatensätzen versteckte, komplexe Muster und Abhängigkeiten zu erkennen und dieses versteckte Wissen zur Entscheidungsfindung an unabhängigen, nicht zum Trainingsdatensatz gehörige Datensätze verallgemeinernd einzusetzen. Die möglichen Anwendungsfelder sind umfangreich und reichen von Produktion, Robotik, autonomen Fahren, Verkehrsleitung über Medizintechnik bis zur Weltraumfahrt. Dabei trifft diese Technik häufig auf sicherheitskritische Anwendungen. Während im Forschungsbereich noch hohe Vorhersagegenauigkeiten imponieren, treten beim Übergang in die Anwendungspraxis Zulassungs- und Risikoabschätzungsfragen auf, die hohe Anforderungen an die Transparenz des Entscheidungsprozesses der NNs stellen.

Bei Neuronalen Netzen sind die Entscheidungsabläufe in der Struktur und einer Vielzahl von Gewichtungswerten versteckt, sie erscheinen Anwendern wie Experten als eine Black Box. Selbst Experten gelingt es nicht ohne weiteres die Entscheidungsabläufe in NNs nachzuvollziehen. Der starke Bedarf nach Entscheidungstransparenz führte zur Entwicklung einer Vielzahl methodischer Ansätze, sogenannte Explainer, die Transparenz in die Entscheidungsprozesse Neuronaler Netze bringen sollen.

Die Auswahl an Explainermethoden führt zu Fragestellungen, die gleichzeitig die Zielsetzung dieser Bachelorarbeit wiedergeben. Welche Explainermethoden zeigen in automatisierter bzw. funktionalbasierter Evaluation (ohne Testpersonen), gute Leistungswerte und wie stellt sich die Performanz im Umfeld verschiedener Datensätze und ML-Modelle dar? Gibt es Ähnlichkeiten in den Ergebnissen untersuchter Explainermethoden? Welche Metriken können zielführend zur Bewertung der Leistung von Explainern eingesetzt werden?

In dieser Arbeit werden vier Erklärverfahren mit drei Metriken zur Bewertung von Explainermethoden im Kontext zweier ML-Modelle mit vier Bilddatensätze getestet. Die Arbeit konzentriert sich dabei auf ML-Lösungen zur Bildklassifikation und Explainermethoden, die mittels Heatmaps entscheidungsrelevante Bildareale kennzeichnen. Die Möglichkeit zur Differenzierung und qualitativen Einschätzung von Erklärverfahren und deren Bewertungsmetriken soll zur verbesserten Transparenz von CNNs beitragen und ist motiviert vom übergeordneten Ziel, den sicheren Einsatz von CNNs in zulassungsbeschränkten Anwendungsfeldern zu unterstützen.

Die nachfolgende Ausarbeitung stellt im Kapitel 2 die technischen Grundlagen vor. Kapitel 3 umfasst den Komplex der Experimente. Hier werden zunächst die ausgewählten Explainermethoden, ML-Modelle, Datensätze und Metriken detailliert beschrieben. Im Anschluss wird die experimentelle Durchführung mit der verwendeten Entwicklungsumgebung vorgestellt und drauffolgend die Ergebnisse der Experimente vorgestellt und diskutiert. Kapitel 4 bietet eine Zusammenfassung und schließt mit einem Ausblick.

2 Background

2.1 Maschinelles Lernen (ML)

Maschinelles Lernen (ML) ist ein Teilgebiet der Künstlichen Intelligenz und setzt eine Sammlung mathematischer Methoden ein, um bestehende Datensätze tiefgreifend auszuwerten. Das Ziel der Analyse bzw. des Lernens ist es, die in den Daten versteckte Erfahrung in Form von Mustern, Abhängigkeiten und Gesetzmäßigkeiten so auszuwerten, dass daraus verallgemeinerbares Wissen entsteht, welches dann auf weitere, nicht dem Lerndatensatz zugehörige Daten angewandt und zur Problemlösung eingesetzt werden kann [10]. ML dient also der Erzeugung von Wissen aus der in Datensätzen enthaltenen Erfahrung. Dabei werden Lernalgorithmen eingesetzt, um teils komplexe Modelle zu generieren, die das erworbene Wissen repräsentieren [13].

2.1.1 Lernverfahren des Maschinellen Lernens

Maschinelles Lernen unterscheidet je nach Aufgabenstellung und Ausgangssituation der verfügbaren Daten verschiedene Lernstile (s.a. Abb. 2.1).

Überwachtes Lernen

Beim überwachten Lernen stehen Datensätze zur Verfügung, bei dem jedes Element einer Lösung (Label) zugeordnet ist. Das Label gibt dabei an, welchen Ausgabewert das Datensatzelement im ML-Modell erzeugen soll. Ein Teil der gelabelten Daten bilden den sogenannten Trainingsdatensatz, anhand dessen ein ML-Modell trainiert wird. Die restlichen gelabelten Daten werden als Testdatensatz verwendet um das trainierte ML-Modell zu verifizieren. Das ML-Modell stellt nach erfolgreichem Training und erfolgreicher Verifikation die „Wissensrepräsentation“ dar und kann nun auf neue, unbekannte Daten derselben Art angewendet werden. Überwachtes Lernen wird u.a. in der Bildklassifizierung angewandt. Beispielsweise lernt eine zur automatischen Zelldiagnose eingesetzte ML-Anwendung an einem vorklassifizierten Datensatz von Zellbildern zwischen gut- und bösartigen Zellen zu unterscheiden und ist nach Abschluss der Lernphase in der Lage dieses „Wissen“ auf noch nicht klassifizierte Bilddaten von Zellen anzuwenden und die Zellen korrekt zuzuordnen. Die Modelle des überwachten Lernens werden auf ein Ziel hintrainiert. Abhängig von der Art des Ziels unterscheidet man verschiedene Klassen des überwachten Lernens. Ist das Ziel eine numerische Vorhersage, z. B. die Höhe eines Aktienkurses spricht man von Regression. Die Eingangsdaten werden dabei dazu verwendet ein oder mehrere numerische Werte vorherzusagen. Von Klassifikation spricht man, wenn die Eingangsdaten so verarbeitet werden, dass am Ende eine Zuordnung zu einer von zwei oder mehreren Klassen steht. Die Bildklassifikation, die im Fokus dieser Arbeit steht, ist ein derartiges Verfahren, die zuvor genannte Zellklassifikation ein Anwendungsbeispiel.

Unüberwachtes Lernen

Unüberwachtes Lernen wird eingesetzt, wenn die Eingangsdaten über keine zugeordneten Lösungen verfügen. Das Ziel dieser Methode ist es grundlegende Muster und Zuordnungen in den Datensätzen zu finden. Clustering Verfahren und Autoencoder sind typische Algorithmen dieser Lernmethode. Ein typisches Einsatzgebiet ist die Prognose von Käuferverhalten.

Bestärkendes Lernen

Beim bestärkenden Lernen leitet das Modell / der Agent aus seinen Eingabewerten Ausgabewerte ab, mittels derer es mit seiner Umgebung interagiert und auf die das Modell ein Feedback erhält. Das Feedback wird dabei eingesetzt um das Verhalten des Modells zu optimieren. Diese Lernvariante wird häufig in der Robotik eingesetzt, um beispielsweise das optimale Greifen von Objekten zu erlernen [13].

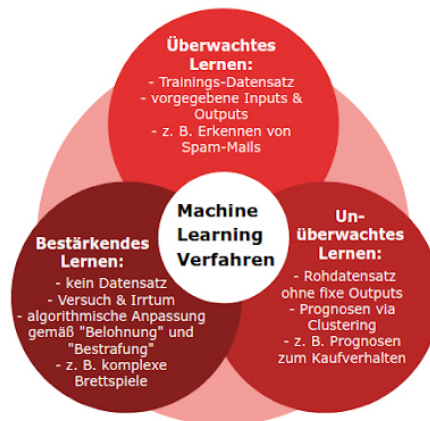


Abbildung 2.1: Lernverfahren des Maschinellen Lernens [29]

2.2 Deep Learning

Für alle Lernverfahren gilt, sofern die Eingangsdaten und deren gegenseitigen Abhängigkeiten von begrenzter Komplexität und Dimensionalität sind, gelingt es dem ML-Modell, welches das Wissen repräsentiert, dieses durch mathematische Modelle wie beispielsweise Entscheidungsbäume abzubilden. Diese wiederum besitzen die Eigenschaft für Menschen gut verständlich zu sein. Häufig jedoch ist die Komplexität und Dimensionalität von Datensätzen zu hoch und es muss auf andere Modelle zurückgegriffen werden.

Vieldimensionale Eingangsdaten können per Deep Learning, einer Disziplin des ML, unter Verwendung künstlicher Neuronaler Netze verarbeitet werden. Neuronale Netze sind selbst in der Lage große Datenmengen auf diejenigen Features zu reduzieren, die für eine korrektes Ausgabeergebnis

erforderlich sind. Der Begriff „Deep“ leitet sich dabei daraus ab, dass zur angemessenen Problemlösung mehrschichtige Neuronale Netze zum Einsatz kommen. Learning bezeichnet die Phase, in der ein untrainiertes Neuronales Netz anhand verfügbarer Daten auf die zu lösende Problemstellung adaptiert wird.

Das trainierte Neuronale Netz stellt eine Lösung zur gegebenen Aufgabe (z. B. Zellklassifikation) dar. Eine derartige Lösung, kann, wie ihr menschliches Pendant, durch fortgesetztes Lernen, seine Fähigkeiten weiter verbessern. Neuronale Netze müssen dabei ggf. andere Methoden einsetzen, damit Gelerntes erhalten bleibt. Beinhaltet der Trainingsdatensatz das Wissen mehrerer Experten, beispielsweise gelabelte Tumordaten die von Pathologen unterschiedlicher Fachrichtungen stammen, ist das Neuronale Netz in der Lage auch das kumulative Wissen anzunähern.

2.3 Neuronale Netze

2.3.1 Aufbau und Funktion

Neuronale Netze besitzen im menschlichen Gehirn ein biologisches und funktionales Vorbild. Die Nervenzellen (Abb. 2.2 links) sind über Dendriten und Synapsen mit weiteren Nervenzellen verbunden, von denen sie Signale empfangen. Ob das eingehende Signal verstärkt oder abgeschwächt wird, wird durch die Ausprägung der Synapse bestimmt. Im Neuron werden die eingehenden Signale aufaddiert und sofern dabei ein Aktivierungspotential erreicht oder überschritten wird, ein Ausgangsimpuls erzeugt, der über ein Axon weitergeleitet wird. Die Änderung der Ausprägung der Synapsen entspricht dem Lernprozess im Gehirn..

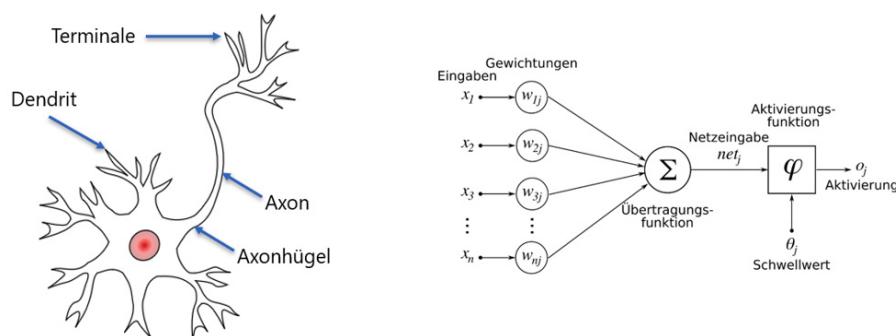


Abbildung 2.2: Links: Nervenzelle des menschlichen Gehirns [40]. Rechts: das mathematische Modell eines künstlichen Neurons (Perzeptron). [39]

Als Perzeptron bezeichnet man das mathematische Modell der Nervenzelle. Die Eingangssignale des Perzeptrons werden als Kanten bezeichnet. Die Signalmodulation durch die Synapsen wird durch die Gewichte $w_1 \dots w_j$ nachgebildet. Das Modell berücksichtigt die Signalaufsummierung und eine schwellwertabhängige Aktivierungsfunktion, die beim Überschreiten des Schwellwertes durch die Netzeingabe ein Ausgangssignal aktiviert.

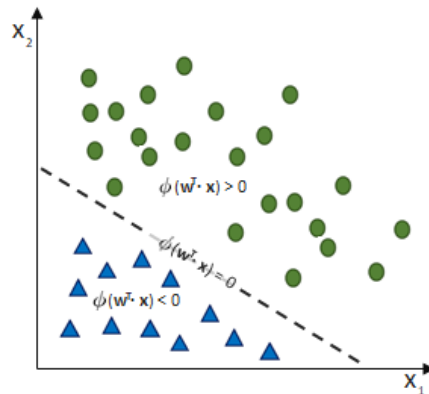


Abbildung 2.3: Lineares Klassifikationsproblem mit zwei Eingabewerten x_1 und x_2 [4]

Mit Hilfe des Perzeptrons kann bereits eine mehrdimensionale lineare Funktion realisiert werden, die dazu eingesetzt werden kann, ein lineares Klassifikationsproblem (s. Abb. 2.3) zu lösen. Durch geeignete Wahl der Gewichtungen w_1 und w_2 kann erreicht werden, dass das Perzeptron für alle grünen Symbole die Aktivierungsfunktion aktiv, für allen blauen Symbole passiv schaltet.

Um komplexere Aufgaben lösen zu können, werden künstliche Neuronen in Schichten angeordnet und untereinander zu künstlichen Neuronalen Netzen verbunden. Wenn im Fortlauf dieser Arbeit von Neuronalen Netzen gesprochen wird, sind damit künstliche Neuronale Netze gemeint. Ein Neuronales Netz, wie in Abb. 2.4 dargestellt, besteht aus einer Eingabeschicht (gelb), ein oder mehreren versteckten Schichten (orange) und einer Ausgabeschicht (rot). Die Anzahl der Neuronen in der Eingabeschicht leitet sich von der Dimension der Eingangsdaten ab. Die Anzahl der Neuronen der Ausgabeschicht von der Aufgabenstellung. Beispielsweise bei der Bildklassifizierung bestimmt die Anzahl der Neuronen der Eingabeschicht die maximale Größe des Eingabebildes in Bildpixeln, wobei größere Bilder ggf. herunterskaliert werden müssen. Die Ausgabeschicht benötigt mindestens so viele Neuronen, wie Klassen, die es zu unterscheiden gilt.

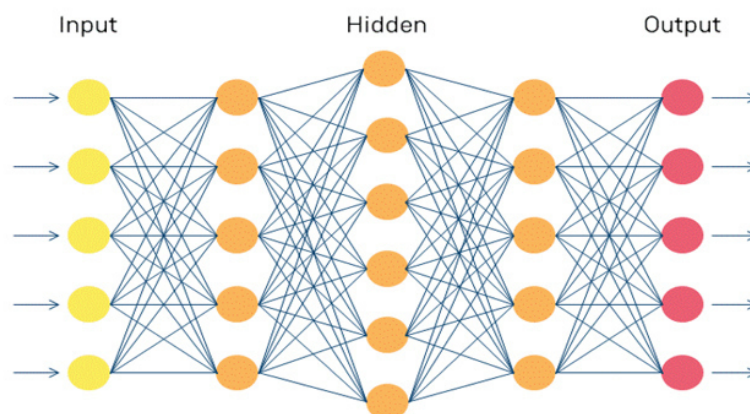


Abbildung 2.4: Struktur eines Neuronalen Netzes [31]

2.3.2 Trainieren von Neuronalen Netzen für die Bildklassifikation

Damit das Neuronale Netz eine gestellte Aufgabe erfüllen kann, müssen die Gewichte mittels eines Verfahrens so angepasst werden, dass jedem Satz an Eingangsdaten die richtige Klassifikation zugeordnet wird. Dieser Vorgang wird in Analogie zum Gehirn als „Lernen“ bezeichnet.

Bildklassifizierungsaufgaben benutzen in der Regel überwachtes Lernen. Dabei werden dem Neuronalen Netz eine ausreichend umfangreiche Serie von bewerteten Eingangsbildern, der Trainingsdatensatz angeboten und ein sukzessiver Lernprozess gestartet. Ein einzelner Trainingsschritt beginnt damit, ein dem Trainingsdatensatz zugehöriges Bild in der Eingabeschicht zu geben. Die Bildinformation wird nun von Schicht zu Schicht gemäß der gewichteten Übertragungsfunktion der beteiligten Perzeptrons vorangeschoben. Dieser Prozess wird Forwardpropagation genannt, er endet an der Ausgabeschicht. Das Ist-Ergebnis der Ausgabeschicht wird nun mit dem Sollergebnis des zum Bild gehörigen Labels verglichen und die Soll-Ist-Differenz in eine Fehlerfunktion abgebildet. Als Fehlerfunktion kann beispielsweise die Mittlere Quadratische Abweichung eingesetzt werden. Damit das Neuronale Netz lernt, müssen dessen Gewichte angepasst werden, dazu wird bei mehrschichtigen Neuronalen Netzen ein Backpropagation Algorithmus eingesetzt. Durch die Verknüpfung der Neuronen der vorangegangenen Neuronenschicht mit denen der Folgeschicht lässt sich die Fehlerfunktion als mehrdimensionale, gekrümmte, von den Parametern der vorangegangenen Neuronenschicht abhängige Ebene darstellen. Um den Fehler zu verkleinern sucht man ein Minimum der Fehlerebene und nähert sich diesem an, indem über den höchsten Gradienten abgestiegen wird. Der Backpropagation Algorithmus ist somit in der Lage über ein Gradientenabstiegsverfahren die Fehlerfunktion durch Anpassen der Gewichte Schicht für Schicht zu minimieren [4].

2.3.3 Vorteile und Limitierungen von Neuronalen Netzen

Der Vorteil tiefer, also mehrschichtiger, Neuronaler Netze liegt in der Fähigkeit zur tief gehenden Abstraktion der Zusammenhänge zwischen Ein- und Ausgabedaten. Diese Technik ist sehr leistungsfähig und erlaubt es Muster und Zusammenhänge auch in umfangreichen Datensätzen herzustellen. Beispielsweise Zusammenhänge, die Menschen so nicht ohne weiteres zugänglich oder für Menschen verarbeitbar sind [30].

Die Reproduzierbarkeit der erzielten Ergebnisse, die Vorteile der maschinellen Verarbeitung (hohe Verarbeitungsgeschwindigkeit, Fähigkeit auch enorm große Datensätze verarbeiten zu können), sowie die Fähigkeit zur Weiterentwicklung bestehender Systeme, bilden weitere Vorteile Neuronaler Netze. Durch die Kombination von Datensätzen unterschiedlicher Experten besteht die Möglichkeit Expertenwissen zu kumulieren.

Der Nachteil dieser Technik liegt darin, dass das Wissen über die Zusammenhänge und Muster versteckt in der Struktur und den Gewichtsdaten des Neuronalen Netzes sind. Selbst geübte Benutzer sind nicht in der Lage die Entscheidungsabläufe in tiefen Netzen nachzuempfinden. Dadurch entsprechen Neuronale Netze funktional einer Black Box, selbst wenn Struktur und alle Gewichte bekannt sind.

2.3.4 Neuronale Netze - Designwahl

Das Fehlen von Richtlinien, wann welches Design eines Neuronalen Netzes erfolgreich einzusetzen ist, erschwert die Anwendung von Neuronalen Netzen, so dass häufig entweder bekannte Architekturen adaptiert werden oder mehrere erfolgsversprechende Architekturen in Tests in Konkurrenz gesetzt werden müssen, um eine möglichst performante Architektur zu erhalten. Das Design des Neuronalen Netzes ist jedoch entscheidend für die Qualität der Lösung. Während ein NN mit zu wenig Schichten oder zu wenig Neuronen nicht in der Lage ist, eine ausreichend präzise Lösung zu bieten, können zu viele Schichten bzw. Neuronen dazu führen, dass es zu „Overfitting“, einer Art „auswendig lernen“ der Musterdatensätze kommt unter Verlust der Fähigkeit zur Generalisierung und damit Verlust der korrekten Zuordnung unbekannter Datensätze [12].

2.4 CNN

Convolutional Neural Networks sind Sonderformen von Neuronalen Netzen, die unter anderem für die Bildklassifizierung eingesetzt werden. In diesem Abschnitt werden die Bestandteile eines CNNs erklärt, bevor die komplette Struktur vorgestellt wird.

2.4.1 Convolutional Layer

Wesentlicher Bestandteil von CNNs sind sogenannte Convolutional Layer. Convolutional Layer kommen dadurch zu Stande, dass statt der festen gewichteten Verbindung zwischen den neuronalen Schichten ein Filterkernel eingesetzt wird. Dieser Filterkernel wird Zeile für Zeile über eine Schicht des Neuronalen Netzes geschoben und daraus werden über eine Faltungsoperation (Convolution) die Werte der Folgeschicht ermittelt. Wie in Abb. 2.5 exemplarisch für einen 3x3 Kernel dargestellt, entsteht der Ausgabewert (grünes Feld) durch Verrechnung eines 3x3 Felds (lila Bereich) der Bildmatrix mit der darüberliegenden 3x3 Matrix des Kernelfelds (hellgelbes Feld). Die Faltungsoperation wird gebildet, indem jeder Wert der Bildmatrix mit seinem Pendant des Filterkernels multipliziert wird und die Produkte aufsummiert den Ausgabewert ergeben. Auf diese Weise errechnet sich ein Neuron der Folgeschicht aus einer räumlich begrenzten Anzahl von Neuronen eines vergleichbaren Bildareals der vorangegangenen Schicht. Die Matrix besteht aus den Ausgabewerten und wird als Featuremap bezeichnet, da sie mit zunehmendem Training in der Lage ist, Merkmale, sogenannte Features, zu repräsentieren, die auf die Bildklassifikationsentscheidung Einfluss nehmen.

Derartige Filter sind aus der Bildverarbeitung bekannt, sie werden dort zur Kontrastanhebung oder Filteroperationen wie Tiefpassfilter verwendet. Die Analogie zum Perzeptronmodell besteht dadurch, dass die Filterkernelwerte die Gewichtungen repräsentieren, auf die eine Signalaufsummierung folgt.

Der Convolutional Layer ist inspiriert durch biologische Vorbilder. Bei der Verarbeitung von Bildinformation im Auge werden ebenfalls in den meisten Teilbereichen des Auges, die Information mehrerer örtlich benachbarten Sinneszellen / Rezeptoren eines rezeptiven Felds auf ein Neuron verdichtet. Auf diese Weise werden die Information aus 126 Millionen Bildrezeptoren im Auge durch 1 Millionen Ganglienzellen aufgenommen und weiterverarbeitet [5].

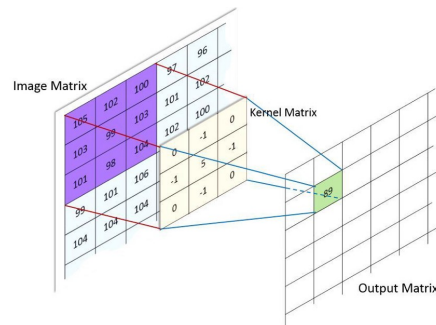


Abbildung 2.5: Faltungsoperation bei der Datenverarbeitung in CNNs [3]

Im Gegensatz zum Auge, das bevorzugte Areale (Fovea mit 60.000 Zäpfchen) mit 1:1 Verbindung zu den Folgeuronen kennt, arbeitet ein Convolutional Layer mit denselben Filtermatrixwerten und der gleichen Filtermatrixgröße über den gesamten Bildbereich. Benachbarte Bildpunkte leisten einen höheren Beitrag für die Bildklassifizierung im Gegensatz zu weit entfernten. Muster und Texturen sind meist unmittelbar in der näheren Umgebung verankert und es ist sinnvoll diese früh zu erkennen, bevor eine zu starke Informationsverdichtung stattfindet. Ob Areale mit gleicher Textur zusammenhängen, kann in Schichten mit höherer Informationsdichte noch realisiert werden.

Die Verarbeitungsweise von Convolutional Layern besitzt eine gewisse „Positionstreu“e. Eingangsinformationen, die im Ausgangsbild im oberen rechten Areal zu finden sind beeinflussen vergleichbare Areale in den darauffolgenden Schichten. Zieht man die menschliche Wahrnehmung heran fällt auf, dass auch hier die Bildklassifizierung von räumlicher Zuordnung profitiert und ein Auto eher als ein solches erkannt wird, wenn das Teilobjekt Reifen sich unterhalb des Teilobjekts Chassis befindet.

2.4.2 Aktivierungsfunktion

Anschließend an die Convolution, die der Übertragungsfunktion in einem Perzepton entspricht, folgt eine Aktivierungsfunktion.

Eine der Aktivierungsfunktionen, die in Abb. 2.7 verwendet wurden, heißt ReLU. ReLU steht für Rectified Linear Unit. Sie ist eine Nachbildung der vom Perzepton bekannten Aktivierungsfunktion. Sie realisiert die Funktion $f(x) = \max(0, x)$ und wird in der Regel durch die differenzierbare Funktion $f(x) = \ln(1 + e^x)$ ersetzt, da Backpropagation diese Differenzierbarkeit für die Berechnung von Gradienten benötigt.

Eine weitere Aktivierungsfunktion, die in Abb. 2.7 genutzt wird, ist die Softmax-Funktion. Diese Funktion bildet reelle Werte auf Wahrscheinlichkeiten ab und wird deshalb häufig in der Ausgangsschicht in einem CNN verwendet.

2.4.3 Pooling Layer

Häufig folgt auf die Kombination von Convolution und ReLU eine Pooling Layer (s.a. Abb. 6). Pooling Layer dienen der Informationsverdichtung. Dabei wird analog zur Erzeugung von Convolutional Layern ein Filterkernel über eine Schicht geschoben, aber eine andere Verrechnungsfunktion

benutzt und in der Regel ist die Schrittweite der Kernerverschiebung höher. Beim weit verbreiteten MAX-Pooling, wird der höchste Wert im rezeptiven Feld des Kernels in der Ausgangsschicht bestimmt und an den darauffolgenden verdichteten Schichten weitergegeben. Abb. 2.6 veranschaulicht diesen Vorgang.

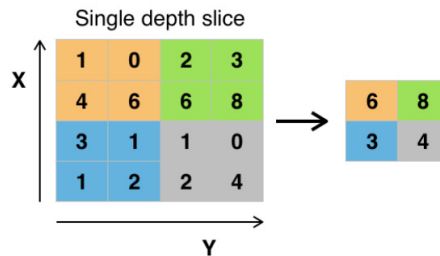


Abbildung 2.6: Pooling Filter mit Max-Pooling Funktion [38]

2.4.4 Flatten Layer

Der Flatten Layer ist eine Zwischenschicht zwischen Convolutional Layer und dem Fully Connected Layer (s.a. Abschnitt. 2.4.5). Seine Funktion besteht darin, eine 2-dimensionale Matrix in einen linearen Vektor zu wandeln, der in einen Fully Connected Layer eingespeist werden kann.

2.4.5 Fully Connected Layer

Von einem Fully Connected Layer spricht man, wenn alle Neuronen des einen Layers mit allen Neuronen des anderen Layers verbunden sind. Zum Einsatz kommt er häufig bei Klassifikationsaufgaben.

2.4.6 Aufbau eines CNN

Die Eingabe für Bildklassifikation sind Bilder, die üblicherweise in einer Matrix aus geordneten Bildzeilen vorliegen, die die Pixelwerte enthalten. Farbbilder umfassen üblicherweise drei Kanäle, für jede Grundfarbe eine Bildmatrix - in Abb. 2.7 angedeutet durch dreifach Struktur des Eingabebilds.

Im Bereich, der mit „Feature Learning“ bezeichnet ist, wiederholen sich Anordnungen von Convolution und ReLU Layern mit Pooling Layer. In diesem Bereich werden das Vorhandensein von zuvor gelernten Bildmerkmalen (Features) erkannt. Ein Flatten Layer, ein Fully Connected Layer und ein Softmax Layer schließen, den mit Classification bezeichneten Bereich des Strukturmodells ab. In diesem Bereich wird die Information aus den verdichteten feature Repräsentationen in eine Klassifikationsentscheidung gewandelt.

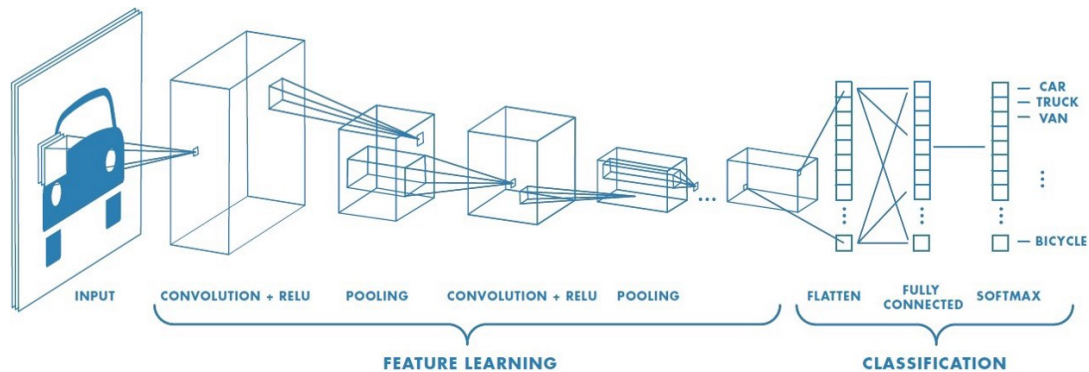


Abbildung 2.7: Strukturmodell eines Convolutional Neural Networks [27]

2.5 Explainable Artificial Intelligence (XAI)

XAI ist ein Forschungsfeld, das anstrebt Methoden zu entwickeln, die helfen sollen die Entscheidungsprozesse von Systemen mit Künstlicher Intelligenz und insbesondere ML-Verfahren transparent bzw. verständlich zu machen [10]. Diese Methoden werden im Folgenden mit den Begriffen Interpretationsmethode, Erklärverfahren oder Explainer bezeichnet.

2.5.1 Motivaton für XAI

Ein Motivator für die Transparenz für von ML-Modellen getroffene Entscheidungen ist die erforderliche funktionale Transparenz, die für die Entwicklung und Zulassung von sicherheits-kritischen Produkten, die ML-Methoden benutzen, erforderlich ist. Weitere Motivatoren liegen darin, dass die Transparenz Potential zur Weiterentwicklung der ML-Methoden mit sich bringt und im möglichen Zugewinn an nutzbarem Wissen über derzeit noch unbekannte Zusammenhänge in großen Datensätzen.

Die Risikobeherrschung ist ein zentrales Thema nahezu aller Zulassungsverfahren für Systeme in sicherheitskritischen Anwendungsdomänen, wie der Medizintechnik, Robotik oder dem autonomen Fahren. Um erfolgreich Produkte zulassen zu können, ist es daher erforderlich, die Funktionsweise und deren Limitierungen ausreichend zu kennen. Neuronale Netze erfüllen diese Anforderungen an die Funktionstransparenz auf Grund des Black Box Charakters in der Regel nicht. Um dieses grundsätzliche Problem zu beheben gibt es umfangreiche Forschungsbestrebungen, die dem Bereich Explainable Artificial Intelligence zuzuordnen sind.

Für sicherheitskritische Anwendungen ist es ein erheblicher Nachteil, wenn mangels Transparenz Entscheidungen und Lösungsfindungen nicht nachvollzogen werden können. Selbst wenn ein System eine sehr gute Vorhersagegenauigkeit aufweist, bleibt unklar, wann es zu Fehlentscheidungen kommen kann. Ansätze zur Risikobeherrschung in solchen Systemen werden aufwändig und heben teilweise oder ganz die Vorteile der maschinellen Verarbeitung (Geschwindigkeit, Reproduzierbarkeit, Automatisierbarkeit) wieder auf.

Um den Sachverhalt zu verdeutlichen: Die Publikation [41] beschreibt den Fall, bei dem die ML-basierte Röntgen-Diagnose auf Lungenentzündung eine Korrelation zwischen eingesetztem mobilem Röntgengerät und dem Diagnoseergebnis herstellt (s. Abb. 2.8). Da das mobile Röntgengerät

2 Background

bevorzugt bei Patienten mit schlechtem Gesundheitszustand eingesetzt wird, ist es nachvollziehbar, dass die ML-Anwendung diese Korrelation zwischen Gerät und Krankheit hergestellt hat, dennoch ist sie ein Fehler, da das Diagnosegerät sicher kein Anzeichen für eine Lungenentzündung darstellt.

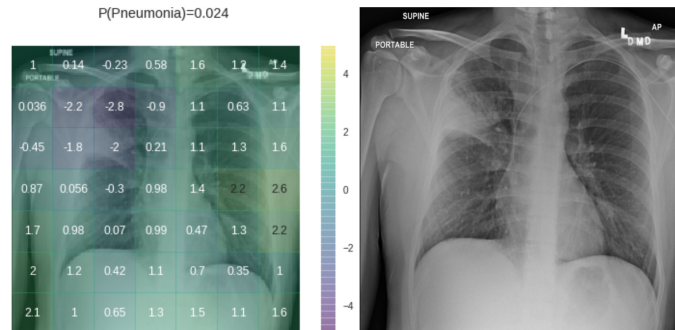


Abbildung 2.8: Patient mit Lungenentzündung, Links: Heatmap des Neuronalen Netzes, Rechts: Röntgenaufnahme [41]

Würde man nun zur Risikobeherrschung sämtliche Klassifikationsergebnisse des Diagnosesystems durch einen menschlichen Experten bestätigen lassen, fielen Vorteile wie Geschwindigkeit und kumuliertes Expertenwissen weg.

Am vorangegangenen Beispiel lässt sich eine weitere Motivation XAI Methoden einzusetzen begründen. Durch die Erklärbarkeit der Entscheidungsfindung ist man nun in der Lage Modelle, wie das gezeigte Neuronale Netz zur Identifikation von Lungenentzündung gezielt weiterzuentwickeln und zu verbessern.

Eine weitere Motivation für XAI besteht darin, durch ein tieferes Verständnis für die von Neuronalen Netzen getroffenen Entscheidungen einen Wissensgewinn zu realisieren, sofern dabei Zusammenhänge erkannt werden, die sich der bisherigen Forschung entzogen haben.

Durch die zunehmende Digitalisierung verfügt man über große Mengen an Daten, insbesondere auch Gesundheitsdaten. Neuronale Netze wären sicher in der Lage in diesen Daten unbekannte Muster und Abhängigkeiten zu erkennen. Wirklichen Nutzen aus diesen künstlichen Erkenntnissen kommen aber erst dann zu Stande, wenn diese Erkenntnisse verständlich gemacht werden können. Da sie dann auch gezielt verifiziert und ggf. genutzt werden können.

Es ist denkbar, dass es unbekannte Zusammenhänge in biometrischen Daten gibt, die Frühindikatoren für später ausbrechende Krankheiten sind. Neuronale Netze könnten automatisiert nach solchen Abhängigkeiten suchen. Wenn die dabei erkannten Muster und Zusammenhänge erklärbar sind entsteht zusätzliches Wissen.

Ein asiatischer Wissenschaftler Divyanshu Mishra formulierte das so: „when AI is significantly stronger than humans, the goal of the explanations is in machine teaching humans how to take better decisions“ [9].

2.5.2 Taxonomie für Interpretationsmethoden

Im Forschungsfeld XAI findet sich eine Vielzahl unterschiedlicher Methoden und Ansätze, mittels derer die Interpretierbarkeit von ML-Modellen hergestellt werden soll. Im folgenden Abschnitt werden zwei gängige Taxonomien für Interpretationsmethoden vorgestellt, die auch zur Einordnung der anschließend vorgestellten und im Test zum Einsatz gekommenen Explainer herangezogen werden.

Modell-spezifische und modell-agnostische Erklärverfahren

Modell-spezifische Erklärverfahren, können ausschließlich auf eine spezifische Modellklasse von ML-Lösungen angewandt werden. Ein Beispiel wäre eine auf Neuronale Netze spezialisierte Interpretationsmethode. Demgegenüber sind modell-agnostische Verfahren auf sämtliche trainierte ML-Modelle anwendbar [22].

Globale und lokale Interpretierbarkeit

Ist eine Interpretationsmethode in der Lage ganzheitlich das Entscheidungsverhalten einer ML-Lösung darzustellen, wird von globaler Interpretierbarkeit gesprochen. Der Nutzer ist anhand des Ergebnisses der Interpretationsmethode in der Lage für jede mögliche Eingabe die zugehörige Entscheidung des trainierten ML-Modells abzuschätzen.

Lokale Interpretierbarkeit liegt vor, wenn das Erklärverfahren den Entscheidungsprozess ausschließlich für spezifische Eingabedaten transparent wiedergibt. Ein derartiger Explainer könnte beispielsweise in einer Bildklassifizierung Transparenz für die Klassifizierungsentscheidung eines Eingabebilds herstellen.

3 Experiment

In den Kapiteln 3.1 bis 3.4 werden die im Experiment genutzten Explainer, ML-Modelle, Datensätze und Bewertungsmetriken beschrieben. Kapitel 3.5 beschreibt die experimentelle Durchführung. Kapitel 3.6 stellt die Ergebnisse vor.

3.1 Erklärverfahren

Die im Test eingesetzten vier Erklärverfahren, gehören alle der Klasse der Interpretationsmethoden für lokale Interpretierbarkeit an. Alle vier Interpretationsverfahren erzeugen für spezifische Eingangsbilder eine Heatmap, die die Informationen über den Entscheidungsprozess des ML-Modells wiedergibt.

3.1.1 KernelSHAP

In [20] stellen Lundberg und Lee KernelSHAP ein modell-agnostisches Erklärverfahren für lokale Interpretationen vor. KernelSHAP ist Teil des Frameworks SHapley Additive exPlanation (SHAP) es zeichnet sich aus, dass es ein Mix aus den Methoden Local Interpretable Model-agnostic Explanations (LIME) und Shapley Values ist. LIME, ein populäres Erklärverfahren, und Shapley Values wird nachfolgend beschrieben. Lundberg und Lee beschreiben KernelSHAP als additives Erklärungsmodell, das sie durch folgende Gleichung definieren:

$$(3.1) \quad g(z') = \phi_0 + \sum_{i=1}^M \phi_i z'_i.$$

Die Grundidee dieser Gleichung besteht darin den über die Erklärfunktion approximierten Funktionswert als eine Summe der „Erklärungen“ darzustellen.

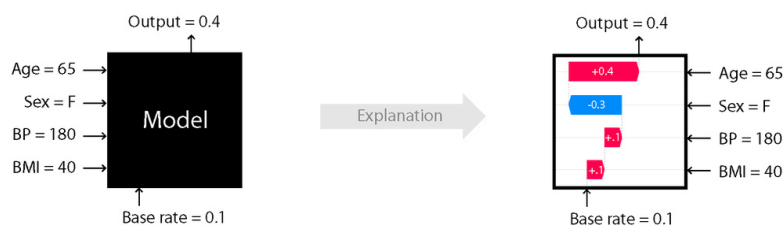


Abbildung 3.1: Graphisches Modell zu Gleichung 3.1. [19]

Dabei steht z' für einen binären Merkmalsvektor, der angibt ob ein Merkmal i vorhanden ist oder nicht und g für das Erklärungsmodell von Vorhersagemodell f . KernelSHAP greift zur Bestimmung der Merkmalswerte ϕ für ein Merkmal i auf die Methode Shapley Values zurück, die im nächsten Abschnitt kurz erläutert wird. Die Ausgabe des Erklärmodells Abbildung 3.1 kommt durch die Aufsummierung der Merkmalswerte zu Stande, dabei werden sowohl positive wie negative Einflüsse berücksichtigt. Die „Base rate“ entspricht ϕ_0 in der Gleichung.

Heatmaps von KernelSHAP weisen Bildarealen Werte zu, die sowohl positiv als auch negativ sein können und in Analogie zu Abb. 3.1 positive wie negative Einflüsse auf die Vorhersage widerspiegeln.

Shapley values

Shapley values [22] ist eine Methode aus der koalitionsübergreifenden Spieltheorie die für Spieler, welche in Zusammenarbeit (Koalition) einen Spielgewinn erzielen, deren fairen Anteil ermittelt. Ersetzt man nun die Spieler durch Eingangsmerkmale eines ML-Modells, das Spiel durch das ML-Modell und den Spielgewinn durch das Klassifikationsergebnis für eine Instanz, ist man in der Lage für jedes Eingangsmerkmal dessen Anteil am Klassifikationsergebnis zu ermitteln. Das Wissen, welche der Merkmale in welchem Maß die Entscheidung beeinflusst haben, unterstützt die Interpretierbarkeit der Entscheidungsfindung für eine spezifische Eingabe (lokale Interpretierbarkeit).

Die Berechnung des Shapley Wert für das Merkmal i ist wie folgt:

$$(3.2) \quad \phi_0 = \frac{1}{M} \sum_{S \subset F \setminus \{i\}} \frac{1}{\binom{M-1}{|S|}} f(S \cup \{i\}) - f(S)$$

Für die Berechnung von ϕ_i benötigt man die Gesamtzahl der Merkmale M , S eine Teilmenge von $F = \{1, \dots, M\}$ ohne Merkmal i und die Differenz aus der Vorhersage von S mit Merkmal i und S ohne i .

LIME (Local interpretable model-agnostic explanations)

LIME [22] ist ein lokales Erklärungsverfahren also die Erklärung für jede einzelne Instanz erzeugt. LIME erzeugt eine Heatmap indem man von der Eingabe Teile wegnimmt und beobachtet wie sich die Klassifikationsgenauigkeit verändert (s.a. Abb. 3.2). Für Textdaten können das Wörter und ganze Satzteile sein. Bei Bildaten hingegen wird das Bild erst mittels Superpixel (Gruppierung von Pixeln) unterteilt und anschließend er durch verschiedene Wegnahmen von Superpixelkombinationen auf die Klassifikation geschaut.

3.1.2 Grad-Cam

Grad-Cam [32] ist eine Methode zur Identifikation von Regionen von Pixeln in Eingabebildern, die signifikant Einfluss auf die Prädiktion eines CNN-Modells genommen haben. Grad-Cam ist somit modellspezifisch und lokal interpretierend. Da Grad-Cam bei der Berechnung Werte aus

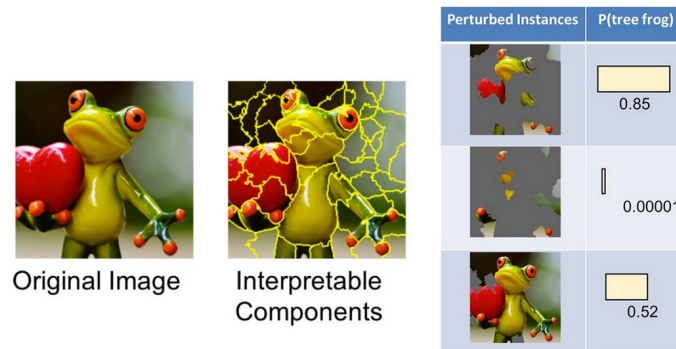


Abbildung 3.2: Erklärungsgenerierung für ein Bild unter Nutzung von LIME. Links das Originalbild. Mitte das Originalbild unterteilt mittels Superpixel. Rechts eine Tabelle mit verschiedenen Löschkombinationen und deren Klassifikationsgenauigkeitswerten [25].

den Aktivierungsmaps eines Convolutional Layer verarbeitet (häufig der letzte Layer), benötigt das Tool Zugriff auf einen Teil der „inneren“ Werte des ML-Modells. Die Erstellung einer Heatmap für ein Eingabebild erfordert einen separaten Zugriff auf die Werte des Convolution Layers. In einem ersten Schritt ermittelt Grad-Cam Gewichtungsfaktoren α_k^c für jede Aktivierungsmap. Zur Ermittlung des α_k^c -Werts (s.a. Gleichung 3.3) wird der Gradient aus dem Ausgangswert der Klasse y^c und jedem Element der Aktivierungsmap k ermittelt, nachfolgend aufsummiert und dann gemittelt. Dabei sind i, j Indizes die über Breite und Höhe der Featuremap laufen und Z ist die Summe aller Featuremap-Elemente also $Z = i * j$. Der Wert α_k^c fällt umso höher aus, je höher die Summe der Einzelgradienten ist, die wiederum widerspiegeln, wie stark sich y^c bei Änderung der Featuremapwerte verändert. Der beschriebene Sachverhalt wird wie folgt berechnet:

$$(3.3) \quad \alpha_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A_{ij}^k}$$

Grad-Cam erzeugt aus einer Linearkombination der mit α_k^c gewichteten Aktivierungsmaps mit anschließender ReLU Funktion eine Heatmap zum Eingabebild. Die ReLU Funktion findet Verwendung, um nur die positiven Effekte herauszufiltern.

$$(3.4) \quad L_{Grad-CAM}^c = ReLU \left(\sum_k \alpha_k^c A^k \right)$$

Die Ausgabe von Grad-Cam gibt ausschließlich positive Einflüsse wieder. Die Auflösung der Heatmap entspricht der Auflösung der verwendeten Aktivierungsmaps. Abb. 3.3 zeigt die Überlagerung des Eingabebildes mit der von Grad-Cam erzeugten Heatmap für die Klasse Katze.

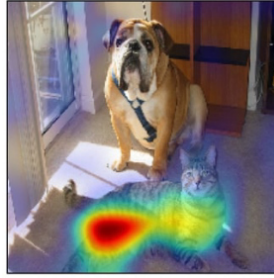


Abbildung 3.3: Grad-Cam Heatmap für Klasse Katze mit unterlegtem Eingabebild [32]

3.1.3 Layerwise Relevance Propagation-epsilon

Layerwise Relevance Propagation-epsilon beschrieben in [7] ist eine Sonderform von Layerwise Relevance Propagation (LRP). Mit LRP wird die Klassifizierungsentscheidung eines ML-Modells im Modell zurückpropagiert, um Auskunft darüber zu erhalten, welche Eingabemerkmale signifikanten Einfluss auf die Prädiktion gehabt haben. Im Falle von Bildklassifizierern entsteht eine Heatmap, die Relevanzwerte enthält. LRP, wie auch LRP-epsilon, ist ein modellspezifischer Explainer (unterschiedliche NNs benötigen sogar unterschiedliche LRP-Gleichungen) mit lokaler Erklärbarkeit. Um LRP-epsilon erklären zu können muss erst die Funktionsweise von LRP erläutert werden. LRP erzeugt eine Heatmap, indem es rückwärts von Schicht zu Schicht die Relevanzwerte für die Neuronen berechnet (s.a. Gleichung 3.5). Dabei steht l für eine Schicht und $l + 1$ für eine Schicht, die anschließend an die Schicht l kommt jedoch näher an der Ausgabe liegt. z_{ij} ist das Produkt aus Neuronenwert x_i und Gewichtung w_{ij} , dabei sind i und j Indizes der Neuronen. i' steht für die Anzahl der Neuronen in der Schicht l .

$$(3.5) \quad R_i^{(l)} = \sum_j \frac{z_{ij}}{\sum_{i'} z_{i'j}} R_j^{(l+1)} \quad \text{mit } z_{ij} = x_i^{(l)} w_{ij}^{(l,l+1)}$$

Als Erweiterung zu der vorangegangenen Formel wird die Variable ϵ in Gleichung 3.6 eingeführt, die den Zweck hat numerische Degenerationen zu vermeiden, wenn der Wert von $z_{i'j}$ nahe Null ist.

$$(3.6) \quad R_i^{(l)} = \sum_j \frac{z_{ij}}{\sum_{i'} z_{i'j} + \epsilon \text{sign}(\sum_{i'} z_{i'j})} R_j^{(l+1)}$$

Die Funktionsweise von LRP-epsilon ist im unteren Teil der Abb. 3.4 dargestellt.

LRP Heatmaps besitzen positive wie negative Werte, die nicht nur anzeigen wie hoch der Einfluss auf die Vorhersage war sondern auch ob ein positiver oder negativer Einfluss bestand.

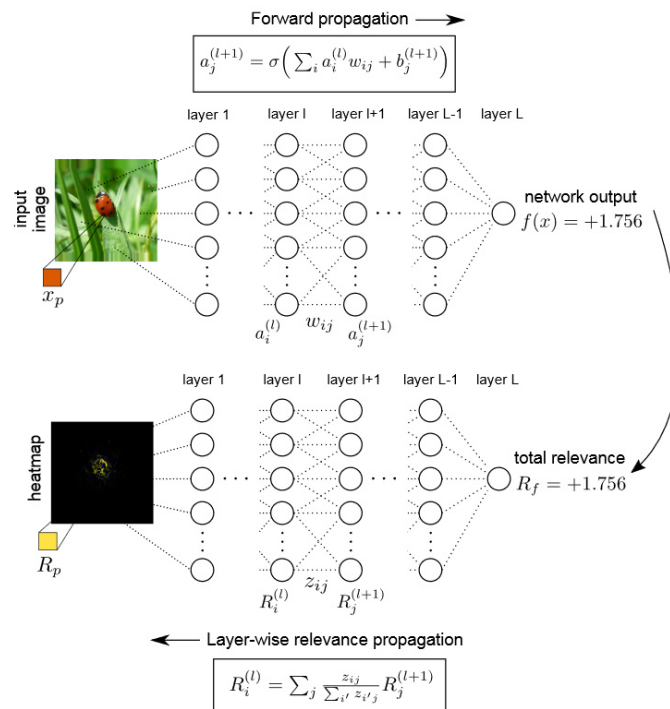


Abbildung 3.4: Illustration des LRP-Prozesses anhand eines Beispiels. Oben klassische Bildklassifizierung im Neuronalen Netz, unten Layer-wise Relevance Propagation im Rückwärtspfad.[7]

3.1.4 Integrated Gradients

Mit IG [6] steht ein weiterer modell-agnostisches Erklärverfahren mit lokaler Interpretierbarkeit zu Verfügung. Bei IG wird zunächst eine als Baseline bezeichnete Eingabe x' gewählt, für den der Prädiktionalgorithmus vorzugsweise Werte bei 0 oder nahe 0 liegend bestimmt. Für eine zu interpretierenden Eingabe x , wird über lineare Interpolation eine Sequenz von gleichmäßig verteilten Eingaben generiert, die von der Baseline x' bis zu x reichen. Für die durchlaufende Sequenz wird nun für jedes Merkmal der Gradient durch Beobachtung des zugehörigen Prädiktionswerts ermittelt. Von Interesse sind all die Merkmale, welche über hohe Gradienten verfügen, die über möglichst große Bereiche der Sequenz anhalten. Wie lange ein Gradient im Verlauf der Sequenz anhält, wird über eine Integration über die Gradienten entlang der Sequenz ermittelt. Integrated Gradient wird in der folgenden Gleichung definiert

$$(3.7) \quad A_i(x, x') = (x_i - x'_i) \int_0^1 \frac{\partial F(x' + \alpha(x - x'))}{\partial x_i} d\alpha.$$

Diese Formel beschreibt die Integration über den Gradienten über den gesamten Sequenzverlauf für Merkmalswert x_i . Abhängig davon ob das Modell ein oder mehrere Klassen unterscheidet wird bei $\frac{\partial F(x' + \alpha(x - x'))}{\partial x_i}$ der Gradient über y oder y_k für die k -te Klasse differenziert.

Die Heatmaps von Integrated Gradients geben ausschließlich positive Werte aus. Im Beispiel Abb. 3.5 rechts die Bildareale die für die Klassifizierung Fireboat ausschlaggebend waren. Die Auflösung der Heatmap entspricht der des Eingabebildes.

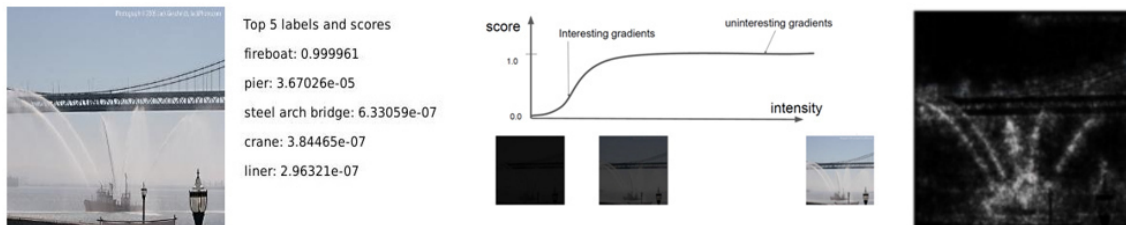


Abbildung 3.5: Darstellung des Funktionsprinzips von Integrated Gradients. Links: Gegeben ist ein zu interpretierendes Eingabebild. Mitte: Sequenz von interpolierten Eingabebilder zwischen Baseline und Eingabebild. Rechts: IG Ausgabebild für die Klasse Fireboat [6].

3.2 Modelle (Neuronales Netz Architekturen)

In diesem Kapitel werden die in dem Experiment verwendeten Neuronale Netz Architekturen vorgestellt.

3.2.1 VGG16

VGG16 ist eine CNN Architektur, die neben Erfolgen auch über eine gute Reputation in der Fachwelt verfügt und sehr häufig eingesetzt wird. Die Besonderheit der Architektur liegt in der Verwendung sehr kleiner (3x3) Filter mit Schrittweite eins im Convolutional Layer und kleine 2x2 Filter mit Schrittweite zwei in der Maxpool-Layer, wobei diese Kombination über die gesamte Architektur beibehalten wird. Am Ende der Architektur kommen 2 Fully Connected Layer, gefolgt von einem Softmax Layer als Ausgabe zum Einsatz. Die Architektur ist in Abb. 3.6 dargestellt, die Bezeichnung VGG16 geht auf die Verwendung von 16 gewichteten Schichten zurück [37], [34].

3.2.2 MobileNet

MobileNet ist eine Convolutional Neural Network Architektur, die sich durch die geringen Anforderungen an Speicherplatz und Rechenleistung auszeichnet und auch für mobile Bildverarbeitung einsetzbar ist [14]. Die hohe Effizienz dieser Architektur kommt dadurch zustande, indem der Standard Convolutional Layer durch einen „Depthwise Seperable Convolutions“ Layer, bestehend aus Depthwise Convolution gefolgt von einer Pointwise Convolution, ersetzt wird (s. Abb. 3.7. Der Standard Layer benötigt die bis zu achtfache Berechnungszeit von einem Depthwise Seperable Convolutional Layer [14].

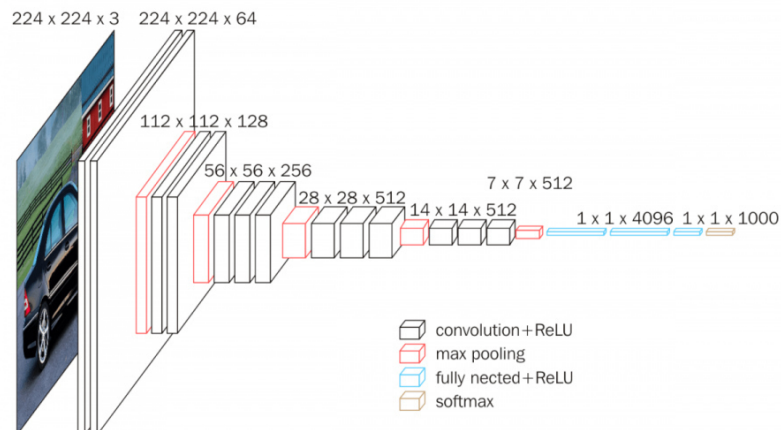


Abbildung 3.6: Aufbau einer VGG16-Architektur [37]

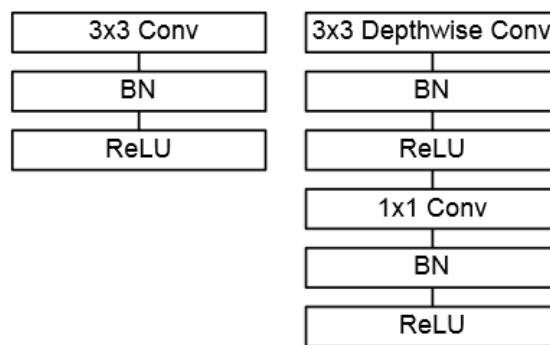


Abbildung 3.7: Links ein Standard Convolutional Layer und rechts den Aufbau eines Depthwise Separable Convolutional Layer [14].

3.3 Datensätze

Für das Experiment wurden vier frei zugängliche Datensätze von Bilddaten verwendet, um die Leistung der Erklärverfahren in verschiedenen Einsatzgebieten zu testen. Der erste Bilddatensatz GTSRB (German Traffic Sign Recognition Benchmark) [35] ist von Interesse wegen des bestehenden Anwendungsbezug zu autonomem Fahren. Um die Nutzung für die Arbeitswelt unter dem Anwendungsbezug Qualitätssicherung zu testen, wurde der Bilddatensatz MVTec AD [11] ausgewählt. Da CIFAR10 [17] ein häufig in wissenschaftlichen Arbeiten verwendeter Bilddatensatz ist [26], [16], [18], wurde er aus Gründen der Vergleichbarkeit der Testergebnisse ausgewählt. Als vierter Bilddatensatz wurde Magnetic Tiles [1] ausgewählt. Er bietet zum Bilddatensatz gehörige Informationen über Groundtruths, eine für die Anwendung der Metrik Relevance Mass Accuracy notwendige Voraussetzung. Eine Groundtruthmap ist eine Zusatzinformation des Bilddatensatzes, die das Bildareal wiedergibt, in dem sich das zu einer bestimmten Klasse gehörende Objekt, im Bild befindet.

3.3.1 CIFAR10

Der CIFAR10 [17] Datensatz umfasst 60.000 Farbbilder mit Bildformat 32x32 für zehn Klassen. Von den 60.000 Bildern gehören 50.000 Bilder dem Trainingsdatensatz an und 10.000 dem Testdatensatz. Dabei sind die Klassen gleichmäßig auf Trainings- und Testdatensatz aufgeteilt. Die zehn Klassen, in die der Gesamtdatensatz unterteilt ist, sind Flugzeuge, Autos, Vögel, Katzen, Rehe, Hunde, Frösche, Pferde, Schiffe, und Lastwagen wie in Abb. 3.8 zu sehen. Es gibt keine Überlappung zwischen den Klassen.

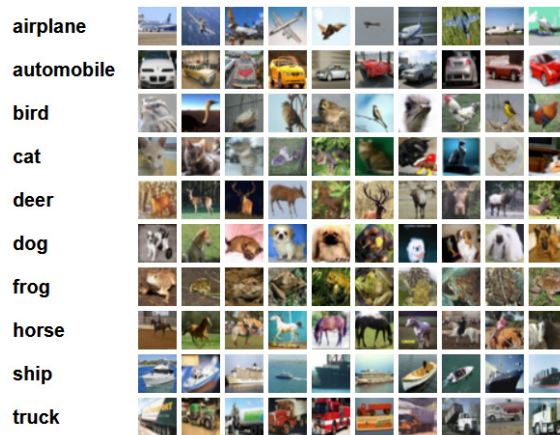


Abbildung 3.8: Beispielbilder für den Datensatz CIFAR10 [17].

3.3.2 MVTec Anomaly Detection Dataset (MVTec AD)

MVTec AD [11] ist ein Bilddatensatz mit 5000 hochauflösenden Bildern. Die Auflösung der Einzelbilder liegt im Bereich von 700x700 bis 1024x1024. Dieser ist in fünfzehn verschiedene Objekt- und Texturklassen unterteilt. Schwerpunkt der Bilder ist die industrielle Inspektion. Die Zielanwendung des Datensatzes liegt in der Erkennung von Anomalien und eignet sich für das Benchmarking derartiger Erkennungsmethoden. Laut Hersteller enthalten die Trainingsdaten fehlerfreie Bilder, die Testdatensätze fehlerfreie und fehlerbehaftete Bilder. Abb. 3.9 zeigt Beispiele für diesen Datensatz. Für das Experiment wurde allein die fehlerfreien Bilder verwendet und eine Bildklassifikation mit den vorhandenen Objektklassen gemacht.

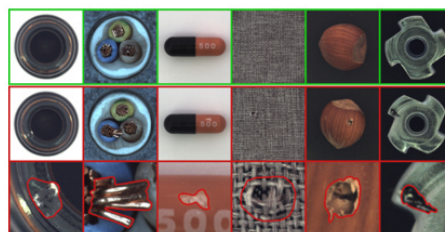


Abbildung 3.9: Beispielbilder für den Datensatz MVTec AD. Obere Reihe: fehlerfreie Objekt. Mittlere und untere Reihe: Objekte mit Fehler [11]

3.3.3 GTSRB (German Traffic Sign Recognition Benchmark)

GTSRB [36] ist ein Datensatz von 50.000 Bildern mit Auflösungen von 15x15 bis 250x250. Dieser Datensatz ist in 43 Klassen deutscher Verkehrskennzeichen unterteilt. Beispiele dafür stellt die Abb. 3.10 dar. Der Datensatz enthält Groundtruthwerte und die Bilddaten sind für Multiklassenklassifikation geeignet.



Abbildung 3.10: Beispielbilder für den Datensatz GTSRB [36]

3.3.4 Magnetic Tiles

Magnetic Tile [15] ist ein Bilddatensatz bestehend aus 1344 Bildern, unterteilt in sechs Klassen von Bildern, die Magnetbanddefekten zeigen. Neben fünf Fehlerklassen, die unterschiedliche Klassen von Defekten zeigen, weißt die sechste Klasse keine Fehler auf (Abb. 3.11). Der Datensatz stellt Groundtruthdaten zur Verfügung. Eine Besonderheit des Datensatzes ist die Varianz der Bilder. Diese werden in unterschiedlichen Ausprägungen der Fehler unter unterschiedlichen Beleuchtungssituationen bei der Bildaufnahme erzeugt.

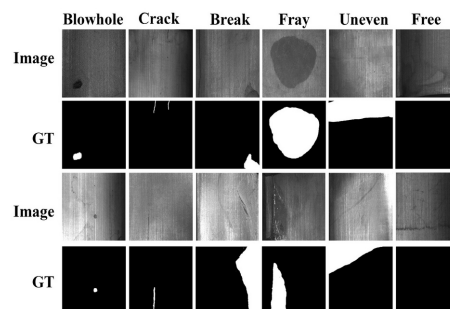


Abbildung 3.11: Beispielbilder für den Datensatz Magnetic Tiles [15]

3.4 Metriken

Im Test wurden drei Metriken eingesetzt, zwei davon dienen der qualitativen Beurteilung von Heatmaps, die dazu herangezogenen Qualitätskriterien werden in den Unterkapiteln erläutert. Die dritte Metrik dient dem Vergleich von Heatmaps und lässt Aussagen darüber zu, inwiefern die von

einem Explainer A in der Heatmap A ausgewiesenen Merkmale mit denen einer Heatmap B eines Explainers B übereinstimmen. Wichtig ist dieser Aspekt bei der Auswahl geeigneter Explainer für einen gegebenen Anwendungsfall. Sollten zwei Explainer qualitativ gute Ergebnisse liefern, und sich in der Merkmalsauswahl nur gering unterscheiden, ist es ausreichend die Untersuchung mit nur einem der beiden durchzuführen. Bei sich stark unterscheidenden Merkmalen in den Heatmaps wäre zu hinterfragen, wie sich dieser Fakt mit dem für beide Explainer positiven Qualitätsurteil verträgt.

3.4.1 Perturbation Metrik

In Paper [28] wird eine Perturbation Metrik vorgeschlagen. Die Grundannahme, die dieser Metrik zugrunde liegt, ist, dass die Vorhersagegenauigkeit für ein Bild sinkt, wenn als besonders relevant eingestufte Pixel verändert werden. Folglich, kann mit dieser Metrik überprüft werden, ob eine Heatmap tatsächlich die für das Klassifikationsergebnis relevanten Bildbereiche hervorhebt.

Die Vorgehensweise, der die Metrik folgt, beginnt damit über das gesamte Bild ein Gittermuster zu legen. Danach wird die Heatmap als geordneter Satz von Orten O zusammengestellt, die auf dem Gittermuster liegen. Der geordnete Satz ist definiert als $O = (r_1, r_2, \dots, r_L)$ dabei ist r_i , der Vektor der zu einem Gitterelement i zeigt und L die Anzahl aller Vektoren r . Für den Vektor r_p , welcher den Ort im Gitter an der Stelle p definiert, gilt die Heatmappingfunktion $h_p = H(x, f, r_p)$. Der Ergebniswert von h_p gibt an wie wichtig der Heatmapbereich für die Klassifizierung ist, dessen Ort im Gitter mit r_p definiert ist. Die Ordnung O wird nun, gemäß MoRF (most relevant first) so angepasst, dass das Gitterelement mit dem höchsten Wert für h_p an erster Stelle (also r_1) steht, die weiteren Elemente werden in absteigender Relevanz dahinter angeordnet. Sukzessive werden vom Ausgangsbild, entsprechend der Reihenfolge in O , nacheinander die Gitterbereiche entfernt und durch eine Störung zufallsbasierter gleichverteilter Pixel ersetzt. Aus der Differenz $f(x_{\text{MoRF}}^{(0)})$ und $f(x_{\text{MoRF}}^{(k)})$ entsteht mit fortschreitendem k die MoRF Störungskurve. Von Interesse für die Metrik ist die Fläche über diese Kurve, die mit AOPC (Area over Perturbation Curve) bezeichnet und mit der Gleichung 3.8 ermittelt wird, dabei bedeutet $p(x)$, dass diese Berechnung über alle Bilder eines Datensatzes durchgeführt und gemittelt wird.

$$(3.8) \quad \text{AOPC} = \frac{1}{L+1} \left\langle \sum_{k=0}^L f(x_{\text{MoRF}}^{(0)}) - f(x_{\text{MoRF}}^{(k)}) \right\rangle_{p(x)}$$

Mit $x_{\text{MoRF}}^{(0)} = x$ ist $f(x_{\text{MoRF}}^{(0)}) = f(x)$, d.h. der Klassifizierungswert des gesamten Bildes. Die Formel benutzt den Klassifizierungswert der Instanz x als Bezugsgröße und besitzt damit den Vorteil, dass die Bewertung des Explainers unabhängig von der Leistungsfähigkeit des ML-Modells in puncto Klassifizierungswert der Ausgangsinstanz ist.

Eine gute Heatmap laut [28] setzt sich aus möglichst wenig Bildarealen zusammen, wobei das Ensemble an ausgewählten Bildarealen für sich genommen einen gut an das Instanzergebnis approximierenden Klassifizierungswert liefern muss. Je signifikanter MoRF fällt, was der Fall ist, wenn die Explainerheatmap ihre Aussage auf wenige sehr signifikante Bildareale begrenzen konnte, desto höher fällt der AOPC- Wert aus. Nach diesem Qualitätskriterium bedeutet je höher der AOPC-Wert ausfällt, desto besser wird ein Explainer eingestuft.

3.4.2 Relevance Mass Accuracy

Die Idee hinter Relevance Mass Accuracy ist es zu ermitteln, mit welcher Relevanz die Heatmap den Bereich, der als Groundtruth gekennzeichnet ist, bewertet. Die Metrik Relevance Mass Accuracy aus dem Paper [24] erstellt die Bewertung eines Explainers, indem sie die vom explainer erzeugte Heatmaps gemeinsam mit der vom gleichen Bilddatensatz stammenden Groundtruths verarbeitet und eine sogenannte Groundtruth Evaluation durchführt. Die Metrik summiert für eine bestimmte Klasse die Relevanzwerte aller Pixel, die innerhalb der Areale der Groundtruth liegen auf R_{within} . Anschließend wird R_{within} ins Verhältnis zur Summe der Relevanzwerte aller Pixel R_{total} gestellt.

$$(3.9) \quad \text{Mass Accuracy} = \frac{R_{\text{within}}}{R_{\text{total}}}$$

$$(3.10) \quad R_{\text{within}} = \sum_{k=1}^K R_{p_k} \text{ mit } p_k \in GT$$

$$(3.11) \quad R_{\text{total}} = \sum_{k=1}^N R_{p_k}$$

R_{p_k} ist der Relevanzwert für den k -ten Pixel p . Anzahl der Pixel von Groundtruth GT wird in der Konstante K beschrieben. N hingegen ist die Gesamtzahl aller Pixel der Heatmap. Je höher der Mass Accuracy Wert liegt, desto besser ist die Heatmap und desto besser ist sie in der Lage, dem durch die Groundtruth identifiziertem Objekt Relevanz zuzuordnen.

3.4.3 Mutual Verification

Die Mutual Verification Metrik aus [42] dient dem Vergleich der Ergebnisse unterschiedlicher Explainermethoden und erlaubt Aussagen, wie stark sich unterschiedliche Explainermethoden in ihren Ergebnis ähneln oder unterscheiden. Die Berechnung erfolgt entsprechend Gleichung 3.12. Dabei werden die Heatmap a_α des Explainers α und die Heatmap a_β des Explainers β jeweils als Matrix interpretiert und zunächst jede für sich L2-normiert. Anschließend werden die Matrizen elementweise zur Subtraktion gebracht und nachfolgend der Betrag der Differenzmatrix berechnet.

Wie bei Metriken häufig angewandt (z.B. Relevance Mass Accuracy) wird der Ergebniswert in Relation auf einen Gesamtwert oder Höchstwert bezogen. Man erhält damit Werte im Intervall $[0,1]$, die dazu dienen den Vergleich zwischen Ergebnissen, die mit derselben Metrik berechnet wurden, zu erleichtern. Soll M_{mutual} in das Intervall $[0,1]$ abgebildet werden, so ist er durch den höchst möglichen Betrag der bei der Berechnung von M_{mutual} entstehen kann zu teilen. Da die L2-Norm für Matrizen und Vektoren auf gleiche Weise berechnet wird, kann man das Abstandsproblem über zwei 2-dimensionale Vektoren ermitteln. Lässt man positive wie negative Werte bei den Vektoren zu, liegen beide normierte Vektoren auf einem Einheitskreis und erreichen eine maximale Differenz von 2 für den Fall, dass sie in entgegengesetzte Richtung weisen. Bei der Reduzierung auf rein

positive Werte für beide Vektoren, liegen beide Vektoren auf einem $\frac{1}{4}$ Einheitskreis, die maximale Differenz reduziert sich dabei auf $\sqrt{2}$ und wird erreicht, wenn die beiden Vektoren den maximal möglichen Winkel von 90° einnehmen. Daher wurde M_{mutual} mit $\sqrt{2}$ normiert gemäß Gleichung 3.13.

$$(3.12) \quad M_{\text{mutual}} = \left\| \frac{a_\alpha}{\|a_\alpha\|} - \frac{a_\beta}{\|a_\beta\|} \right\|$$

$$(3.13) \quad M_{\text{mutual}_{\text{norm}}} = \frac{M_{\text{mutual}}}{\sqrt{2}}$$

Bei großer Ähnlichkeit, heben sich die Matrixelemente bei der Subtraktion weitgehend auf d.h. ähnliche Heatmaps unterschiedlicher Explainer erzielen mit dieser Metrik niedrige Werte. Je näher der Wert an 1 tendiert umso unterschiedlicher sind die Heatmaps.

3.5 Experimentdurchführung

Als Entwicklungsplattform kommt Google-Colab zum Einsatz, die über einen integrierten Python Interpreter verfügt. Die Programmiersprache Python wird zur Steuerung sämtlicher experimenteller Abläufe genutzt, dabei kann Python auf Methoden des Frameworks Tensorflow zugreifen. Alle Berechnungen, die Colab durchführt, inklusive der rechenintensiven Operationen der ML-Modelle erfolgen auf Google Servern. Das Ressourcenmanagement von Colab lässt es zu, zur Beschleunigung der Berechnungsvorgänge statt auf die Standard CPU auf GPU Rechenpower zuzugreifen.

Die Durchführung des Experiments hat in zwei Phasen stattgefunden (s. Abb. 3.12). Phase eins hatte zum Ziel eine qualitative Bewertung der Erklärungsverfahren zu erstellen. Dazu wurden zunächst die ML-Modelle anhand der Trainingsdaten der Datensätze trainiert. Danach standen acht trainierte Modelle zur Verfügung (vier trainierte MobileNet Modelle und vier trainierte VGG16 Modelle). Nachfolgend klassifizierte jedes der trainierten Modelle den zugehörigen Testdatensatz. Aus den klassifizierten Bildatensätzen wurden diejenigen ausgewählt, die korrekt klassifiziert wurden, da Erklärversuche an falsch klassifizierten Daten, also Fehler des ML-Modells, nicht zu Lasten der im Testfokus befindlichen Explainermethoden gehen sollte.

Anschließend wurde zu jedem korrekt klassifiziertem Testbild unter Anwendung der Explainermethoden je vier Heatmap-Datensätze pro Modellarchitektur erstellt. Jede der über 20.000 Heatmaps wurden in Phase eins durch die Perturbation Metrik bewertet. Hierbei wurde sukzessive Bildinformation von bis zu 15% entfernt. Für die Relevance Mass Accuracy Metrik kamen nur die Heatmaps von Datensatz Magnetic Tiles zum Einsatz, da dieser als einziger über die für diese Metrikmethode erforderlichen Groundtruthmaps verfügt. Um keine Verfälschung der Ergebnisse zu bekommen wurde aus Magnetic Tiles die Klasse „free“ entfernt. Diese Klasse steht für Bilder die keinen Fehler aufweisen und enthält somit nur Groundtruths ohne wichtigen Bereich. Die Groundtruthwerte von GTSRB wurden nicht verwendet, da diese sehr große Anteile des eigentlichen Bildes beinhalten und somit nicht selektiv genug sind. Das durch die Anwendung der Metriken erzeugte Datenmaterial wurde zur qualitativen Bewertung der Explainermethoden herangezogen.

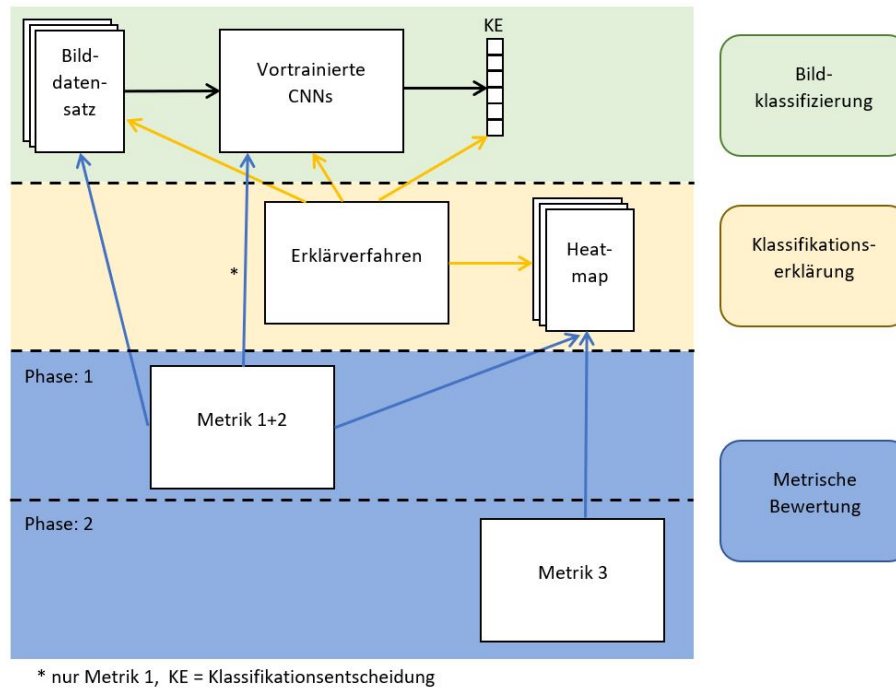


Abbildung 3.12: Zugriffsfunktionen und Teilschritte des Experimentellen Ablaufs für Experimentphase 1 und 2. Metrik 1: Perturbation Metrik. Metrik 2: Relevance Mass Accuracy Metrik, Metrik 3: Mutual Verification Metrik.

In der experimentellen Phase 2 (s.a. Abb. 3.12) wird unter Anwendung der Metrik Mutual Verification der Zielfrage nachgegangen, wie vergleichbar die Explainermethoden untereinander sind. Die Metrik Mutual Verification vergleicht Heatmaps zweier Explainer eines Datensatzes. Daher wurden die vier Methoden in 6 zweier Kombinationen bewertet. Zu jedem Originalbild wurden die zugehörigen Heatmaps der zwei Explainer, die im direkten Vergleich stehen, bewertet. Das Gesamtergebnis jedes Direktvergleichs ergibt sich dabei aus dem arithmetischen Mittel der aufsummierten Einzelbewertungen. Da zwei Explainer positive und negative Einflüsse in die Heatmap abbilden, wurden die negativen Einflüsse entfernt. Diese Vorgehensweise wurde gewählt, um eine bessere Vergleichbarkeit mit den Heatmaps der Erklärverfahren, die nur positive Einflüsse berücksichtigen, herzustellen.

Im Folgenden sind zusätzliche Detailhinweise zur Experimentdurchführung aufgeführt. Für die Verwendung der Erklärverfahren im Experiment wurde auf Github Repositories (KernelSHAP: [19], Grad-Cam und Integrated Gradients: [33], LRP-epsilon: [2]) zurückgegriffen. Wegen verschiedener Rechenleistungsanforderungen der beiden Modellarchitekturen (VGG16 und MobileNet) wurden die Bildformate 75x75 bei VGG16 und 128x128 bei MobileNet verwendet. Die Datensatzbilder wurden für das Experiment dahingehend skaliert. Da die Testbilddatensätze große Mengen an Testbildern zur Verfügung stellen, mussten diese auf repräsentative Teilmengen reduziert werden (s. Tab. 3.1).

Tabelle 3.1: Testdatensatztabelle

Bilddatensatz	Anzahl Testbilder	Anz. verw. Testbilder
CIFAR10	10000	1000 (für jede Klasse 100)
MVTEC AD	407	407
GTSRB	12630	1075 (für jede Klasse 25)
Magnetic Tiles	267(mit Klasse "free")	77(ohne Klasse "free")

3.6 Testresultate, Analyse und Interpretation

3.6.1 Phase 1: Perturbation Metrik

Abbildung 3.13 zeigt die experimentell ermittelten Perturbation Metrik für die ML-Modelle VGG16 und MobileNet für unterschiedliche Datensätze und unterschiedliche Erklärverfahren (s.a. Abschnitt 3.4.1). Die Darstellung ist gruppiert nach ML-Modellen und fasst pro Bilddatensatz die Wertverläufe der vier Erklärverfahren in einer Grafik zusammen. Da bei dieser Methode, gesteuert durch die Heatmap des jeweiligen Erklärverfahrens, sukzessive die wichtigsten Bildareale zuerst gestört werden, ist zu erwarten, dass bei Kurvenbeginn eine stärkere Abnahme der Vorhersagewerte entsteht, der im weiteren Verlauf verflacht. Die dargestellten Kurvenverläufe zeigen prinzipiell dieses erwartete Verhalten.

In Summe schneidet Grad-Cam in 7 von 8 Fällen mit den schwächsten Werten ab. Nur in Kombination VGG16 Magnetic Tiles und MobileNet mit Datensatz CIFAR10 zeigt Grad-Cam Tendenz mit den anderen Erklärverfahren mitzuhalten.

In fast allen Kombinationen aus ML-Modell und Datensatz zeigen LRP-epsilon, Integrated Gradients und KernelSHAP vergleichbare Ergebnisse mit dem bereits beschriebenen signifikanten Kurvenverlauf. Der Datensatz mit den ML-Modell-übergreifend besten Metrikverläufen ist GTSRB. Das Ergebnis überrascht nicht, da dieser Datensatz mit Verkehrskennzeichen Objekte enthält, die Eigenschaften wie klare Erkennbarkeit, signifikante Form und Farbgebung, sowie geringe Varianz im Erscheinungsbild aufweisen.

Der Datensatz MVTEC stellt in Kombination mit MobileNet ein Problem für alle Erklärverfahren dar, das Problem ist bei VGG16 deutlich geringer. Da mit Ausnahme von Grad-Cam die Kurvenverläufe für die Explainer geringe Streuung aufweisen, kann angenommen werden, dass MobileNet mit dem Datensatz MVTEC weniger gut zurechtkommt. Ein weiteres Argument, das für ein Problem beim ML-Modell spricht ist die Tatsache, dass das ML-Modell mit geringerer Bilddatenauflösung die besseren Ergebnisse erzielt.

Der Datensatz Magnetic Tiles weist eine Art Rauschen im Kurvenverlauf auf. Der Effekt ist bei VGG16 stärker als bei MobileNet, wobei VGG16 stärker abfallende Kurvenverläufe bietet als MobileNet. Das Rauschen, das im Verlauf der Perturbationkurve der Erklärverfahren erkennbar ist, kann unter anderem auf die geringe Anzahl nutzbarer Bilder im Datensatz Magnetic Tiles zurückgeführt werden. Eine geringe Anzahl Testbilder bedeutet geringere Anzahl durch Erklärverfahren erzeugte Heatmaps, was wiederum zu einer geringeren Anzahl Perturbationkurven führt, über deren Anzahl gemittelt wird.

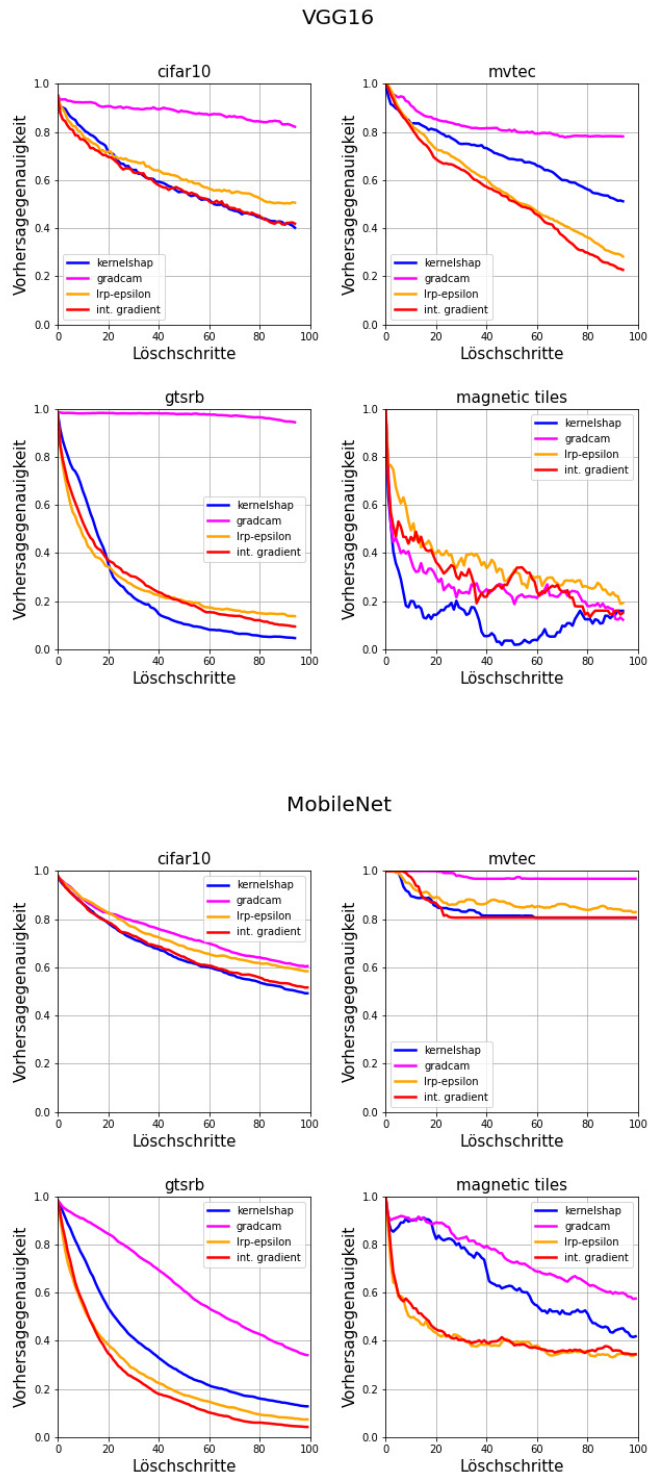


Abbildung 3.13: Vorhersagewertverläufe von VGG16 und MobileNet für unterschiedliche Datensätze und unterschiedliche Erklärverfahren für sukzessive Störung von Bildarealen gemäß Methode Perturbation metrik

Abbildung 3.14 stellt die gemittelten AOPC-Werte, für die Erklärverfahren über den Datensätzen getrennt für VGG16 und MobileNet dar. Diese Form der Darstellung erleichtert es, Erklärverfahren anhand ihrer AOPC Werte im Zusammenspiel mit den Datensätzen zu vergleichen. Es kann in dieser Darstellung ebenfalls erkannt werden, dass Grad-Cam in 7 von 8 Vergleichen den niedrigsten Wert erzielt. Das Diagramm verdeutlicht die starke Übereinstimmung von LRP-epsilon und Integrated Gradients. Ebenfalls erkennbar ist die Ähnlichkeit von KernelSHAP zu LRP-epsilon und Integrated Gradients im tendenziellen Verhalten, wie in der Nähe der AOPC-Werte untereinander. Im Ranking (s.a. Tab.3.2) entsteht die absteigende Rangfolge Integrated Gradients mit gemitteltem AOPC-Werte 0,49, LRP-epsilon 0,47, KernelShap 0,46 und Grad-Cam 0,23.

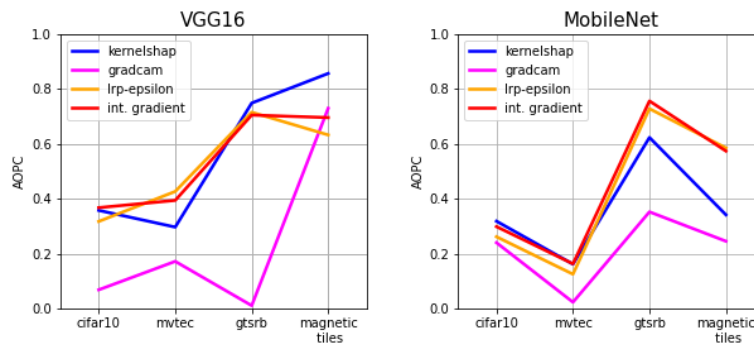


Abbildung 3.14: AOPC-Wertkurven für unterschiedliche Explainer

Tabelle 3.2: Explainer Ranking nach über alle Datensätze und Modellarchitekturen gemittelten AOPC-Werten.

Platzierung	Explainer	gemittelter AOPC-Wert
1	Integrated Gradient	0,49
2	LRP-epsilon	0,47
3	KernelSHAP	0,46
4	Grad-Cam	0,23

3.6.2 Phase 1: Relevance Mass Accuracy

Für die Metrik Relevance Mass Accuracy, die Werte im Intervall [0.0, 1.0] liefert, wurde der Testdatensatz von Magnetic Tiles verwendet. Dieser Datensatz wurde auf alle Bilder, die nicht zur Klasse „free“ gehören und die korrekt klassifiziert wurden, reduziert. Die Klasse „free“ wurde entnommen, da dessen Groundtruthmaps keinen Bereich enthalten, der markiert werden soll.

Die Abb. 3.15 zeigt, dass der Algorithmus niedrige Werte für alle Explainer annimmt. Dieser Sachverhalt lässt sich damit erklären, dass im stark reduzierten Testdatensatz viele Bilder mit Fehlern enthalten sind, die sehr kleine Groundtruthflächen aufweisen. Dennoch lässt sich erkennen, dass Integrated Gradients bei beiden ML-Modellen die wichtigen Bildregionen am besten kennzeichnet.

Dieses Ergebnis ist stimmig zu den guten Ergebnissen von Integrated Gradients bei den AOPC-Werten. Explainer KernelSHAP ist marginal besser als Grad-Cam und LRP-epsilon, die sich in ihren Ergebnissen nur kaum unterscheiden.

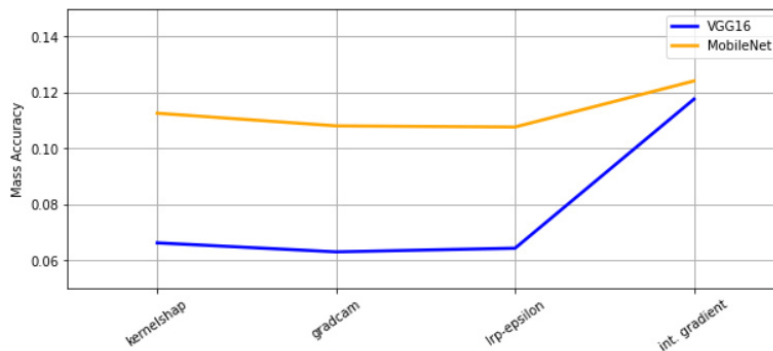


Abbildung 3.15: Liniendiagramm über die Mass Accuracywerte über die verschiedenen Explainer

In [24] konnte unter Anwendung der gleichen Metrik und Datensätze mit ausgeprägter Groundtruth höhere Werte erzielt werden. Dieser Umstand lässt den Schluss zu, dass die Metrik sinnvoll in der Anwendung bei Datensätzen ist, die über eine gutdefinierte Groundtruth verfügt (z.B. Erkennung von Objekten). Bei Datensätzen mit sehr kleiner Groundtruth hingegen, wie dies bei der Anomalieerkennung (Kratzer und Defekte) der Fall ist, ist die Anwendung kritisch zu bewerten.

Bei der Überprüfung der Bilder des Groundtruthdatensatzes, fiel zudem auf, dass das von der Groundtruth identifizierte Areal genau mit den Objektgrenzen übereinstimmt. ML-Modelle benötigen zum Erkennen von Formen und Umrissen auch Informationen aus dem unmittelbaren Umfeld eines Objekts, eben um die Objektgrenzen erkennen zu können. Heatmaps weisen daher diesem Umfeld ebenfalls hohe Relevanzwerte zu. Für Relevance Mass Accuracy, kommen diese Areale nur zur Geltung, wenn außer dem Groundtruthareal das Umfeld mit einbezogen wird. Daher der Vorschlag die Metrik dadurch zu verbessern, indem das Groundtruthareal um einen definierten Prozentsatz erweitert wird.

3.6.3 Phase 2: Mutual Verification

Experimentphase 2 mit der Mutual Verification Metrik dient dem Vergleich von Explainern, indem ihre Heatmaps verglichen werden und ein Berechnungswert erzeugt wird, der angibt wie stark zwei Heatmaps übereinstimmen. Daraus lässt sich schließen, ob die erzeugenden Explainer vergleichbar in der Erkennung wichtiger Pixelareale in den Testbildern sind. Der Vergleich von vier Explainern führte zu sechs paarweisen Vergleichen für jeden Datensatz und jedem Modelltyp. Die über die Bilddatensätze gemittelten Werte sind in Abbildung 3.16 grafisch dargestellt. Die gemäß der Gleichungen 3.12 und 3.13 errechneten Werte im Intervall $[0,0, 1,0]$ besagen, dass kleine Werte auf eine hohe Übereinstimmung schließen lassen, während hohe Werte eine geringe Übereinstimmung wiedergeben.

Für MobileNet erkennt man in Abb. 3.16 bzw. in Tab. 3.3, dass die Erklärverfahren LRP-epsilon und Grad-Cam die niedrigsten Werte und damit die beste Übereinstimmung in den Heatmaps erzielt. In der Grafik ist diese Ausprägung für drei der Datensätze stark und für den vierten Datensatz

3 Experiment

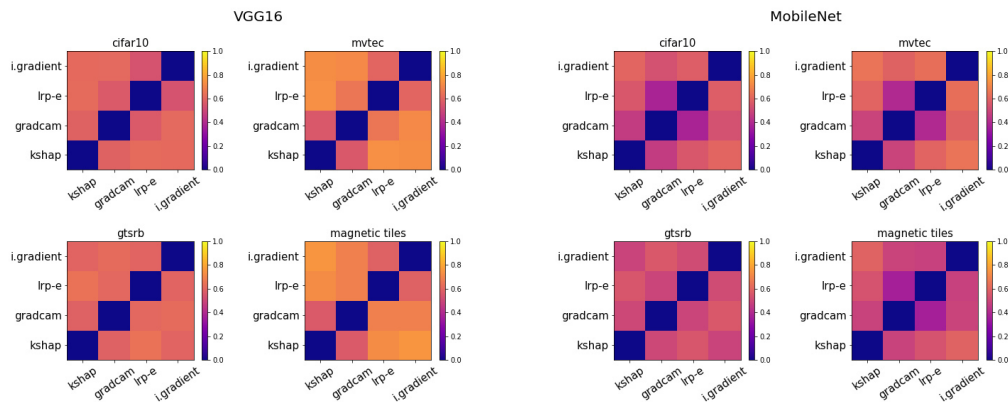


Abbildung 3.16: Paarweiser Vergleich der von den vier Explainer erstellten Heatmaps mittels Mutual Verification Metrik

Tabelle 3.3: Minimalwerte aller Mutual Verification Vergleichspaare für VGG16 und MobileNet

Vergleichskombination	Min (VGG16)	Min (MobileNet)
KernelSHAP/Grad-Cam	0,56	0,46
KernelSHAP/LRP-epsilon	0,62	0,54
KernelSHAP/Integrated Gradient	0,60	0,49
Grad-Cam/LRP-epsilon	0,57	0,36
Grad-Cam/Integrated Gradient	0,62	0,49
LRP-epsilon/Integrated Gradient	0,54	0,48

noch erkennbar. Die Ursache dafür sollte in der Ähnlichkeit beider Erklärverfahren begründet sein. Beide Verfahren greifen für die Bestimmung der Heatmap auf „innere Werte“ des ML-Modells zu. Während LRP-epsilon von Schicht zu Schicht vorgeht, nutzt Grad-Cam eine Methode die bei der Schicht der letzten Featuremaps und Gradientengewichtung ansetzt (für Details s.a. Kap. 3.1.3 und 3.1.4). Für VGG16 erzielt die Kombination aus LRP-epsilon und Integrated Gradients den niedrigsten Mutual Verification Wert und somit die beste Übereinstimmung (s. Tabelle 3.3). Integrated Gradients und LRP-epsilon haben bereits bei Metrik 1 stark vergleichbare Ergebnisse erzielt.

Wenn man das Kachelmuster von Paper [42] heranzieht, kann man erkennen, dass der Übereinstimmungswert, den die Metrik Mutual Verifikation erzeugt, von dem verwendeten ML-Modell abhängt. In Abbildung 3.17 wird der Datensatz VOC2012 an fünf ML-Modelle angewandt. Von den fünf ML-Modellen gibt es zwei ML-Modelle von Typ ResNet und zwei VGG Modelle. Man erkennt, dass prinzipähnliche Modelle (z.B. VGG16 u. VGG19) ähnliche Muster aufzeigen, wohingegen unterschiedliche Modelle wie AlexNet und die VGG ML-Modelle starke Unterschiede im Kachelmuster aufweisen. Unterschiedliche Kachelmuster bei unterschiedlichen ML-Modellen bestätigt auch die Mutual Verification Metrik in der vorliegenden Arbeit.

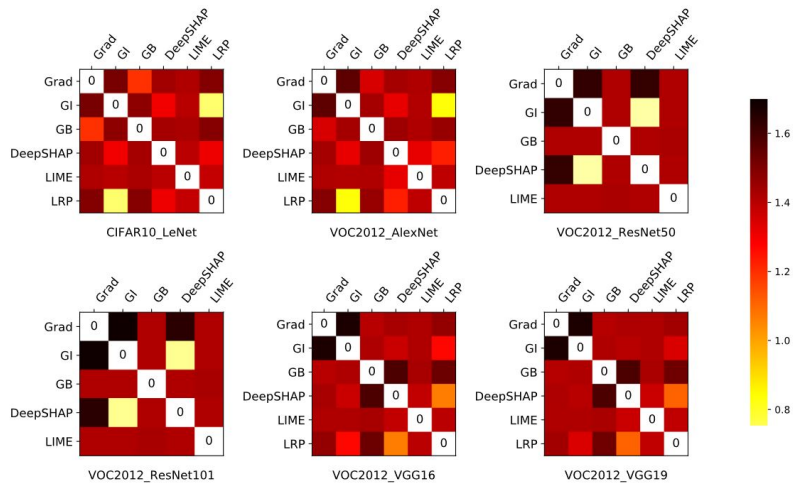


Abbildung 3.17: Mutual Verification Ergebnisse aus Metrikpapier [42]

Aus wiederkehrenden Mustern bei den Kachelschaubildern von unterschiedlichen Datensätzen aber gleicher Modellarchitektur kann man folgern, dass Datensätze nicht so signifikant das Ergebnis beeinflussen wie die Modellarchitektur. Beispiel dafür ist das wiederkehrende 2x2 Muster in der oberen linken Ecke in Abbildung 3.16 für MobileNet bei den Datensätzen CIFAR10, MVTeC und Magnetic Tiles. Diese Schlussfolgerung wird bestätigt mit den fast identischen Kachelschaubilder von den architekturähnlichen ML-Modellen LeNet und AlexNet, die jedoch verschiedene Datensätze verwenden (s.a Abb. 3.17).

4 Zusammenfassung

Die Tests mit der Perturbation Metrik zeigen, dass mit LRP-epsilon, Integrated Gradients und KernelSHAP drei Explainer vorliegen, die leistungsmäßig eng beieinander liegen. Insbesondere LRP-epsilon und Integrated Gradients zeigen sehr hohe Übereinstimmung. Dabei ist LRP-epsilon bereits mit Erfolg in verschiedenen Applikationen eingesetzt wurde [23]. Die starke Übereinstimmung ist zudem bemerkenswert, da LRP-epsilon vom methodischen Ansatz her ein modellspezifisches Erklärverfahren darstellt, das einen „tiefen“ Blick in die Black Box benötigt, während die Methode auf der Integrated Gradients beruht, komplett ohne Zugriff auf „inneren Werte“ des ML-Modells auskommt und seine Ergebnisse komplett durch variieren des Eingabebilds unter Beobachtung des Klassifizierungsergebnisses erlangt. Integrated Gradients gehört damit zu den modellübergreifend einsetzbaren, agnostischen Erklärverfahren. KernelSHAP liefert mit leichten Abstrichen annähernd so gute Ergebnisse wie LRP-epsilon und Integrated Gradients. Der Explainer Grad-Cam erzielt nur in speziellen Kombinationen aus Modellarchitektur und Datensatz akzeptable Ergebnisse.

Bei der Anwendung der Metrik Relevance Mass Accuracy unter Verwendung des Datensatzes Magnetic Tiles (mit Groundtruth Annotation für Defektstellen) hat der Explainer Integrated Gradients bei beiden ML-Modellen die wichtigen Areale der Groundtruth am besten getroffen. Der Quervergleich der experimentellen Ergebnisse mit anderen Arbeiten, mit gleicher Metrik aber anderen Datensätzen, lässt den Schluss zu, dass diese Metrik erhöhte Anforderungen an die Groundtruthareale stellt um erfolgreich eingesetzt zu werden.

Die in der umfangreichen Experimentalphase 1 erzielten Ergebnisse bzgl. der Explainermethoden LRP-epsilon, Int. Gradients und KernelSHAP unter vielfältig variierten Einsatzbedingungen konnten in Experimentalphase 2 bei der Ähnlichkeitsbestimmung mit der Methode Mutual Verification nicht signifikant bestätigt werden. Die Metrik Mutual Verification identifiziert für MobileNet eine gute Übereinstimmung zwischen Grad-Cam und LRP-epsilon in drei von vier Datensätzen. Für das andere ML-Modelle konnte dieses Ergebnis nicht reproduziert werden. Vergleichspaar Integrated Gradients/LRP-epsilon waren mit Modellarchitektur VGG16 am ähnlichsten. Unter veränderten Kombinationen des ML-Modell und Datensatz zeigt die Metrik, die paarweise Erklärverfahren vergleicht variierende Ergebnisse, was auch [42] erkennen lässt. Die Wahl des ML-Modells nimmt mehr Einfluss auf das Ergebnis als die Wahl des Datensatzes. Dieser Umstand ist beim Einsatz dieser Metrik zu berücksichtigen.

Da wie in der Arbeit gezeigt die Ergebnisse metrischer Auswertung stark vom Verwendungskontext beeinflusst sein können, sollte die Anwendung von Metriken durch menschliche Evaluation mit kritischem Hinterfragen der Ergebnisse begleitet werden.

Ausblick

Von den getesteten Erklärverfahren stehen drei Methoden mit vergleichbaren Leistungsparametern zur Verfügung, von denen eines in einer Vielzahl von Anwendungen auch erfolgreich eingesetzt werden konnte. Diese Erklärverfahren sind in der Lage automatisch und reproduzierbar Bereiche in den Eingangsdaten zu identifizieren und qualitativ zu gewichten, wie stark diese Eingangsdaten an der Entscheidung des ML-Modells teilgenommen haben. Damit lassen sich die Vorhersagen von ML-Modellen auf jedem Fall auf Plausibilität überprüfen und offensichtliches Fehlverhalten, wie die Diagnose einer Lungenentzündung wegen vom ML-Modell erkannter Korrelation zwischen Röntgengerätetyp und der Krankheit (s.a. Kapitel 2.5.1), ausschließen. Durch die qualitative Gewichtung der Merkmale an der Entscheidungsentstehung kann darüberhinaus zusätzlich differenziert werden, welche Merkmale / Bildareal wichtig für die Entscheidung waren und welche sehr wichtig dafür waren. LRP-epsilon und KernelSHAP lassen durch die Identifikation von negativen Einflüssen auch eine Differenzierung negativer Entscheidungseinflüsse zu und bieten daher eine noch höhere Informationsdichte.

Dennoch ist die Erklärbarkeit der Erklärverfahren bei der Bildklassifikation noch nicht auf dem Niveau angekommen, bei dem eindeutige Entscheidungsregeln ableitbar sind. Kennzeichnet ein Erklärverfahren beispielsweise Gesichtsbereiche eines Wolfs als maßgebliches Entscheidungsmerkmal, bleibt unklar ob die Entscheidung aus einer Textur stammt, einem biometrischen Maß z. B. Augenform, Schrägstellung der Augenachse oder einer speziellen Kombination dieser Merkmale. Es bedarf also weiterer Forschungsarbeit, um den Grad der Erklärbarkeit weiter zu steigern, um noch näher an die tieferen Entscheidungsabläufe heranzukommen.

Da nicht immer davon ausgegangen werden kann, dass die gefundenen Entscheidungsfunktion einfach genug sind, um unmittelbar verstanden zu werden, lohnt auch Forschungsarbeit in Methoden zur verständlichen Darstellung komplexer Sachverhalte anzustrengen.

ML-Anwendungen werden intensiv bei der Analyse von Nutzerdaten in Entscheidungsprozessen eingesetzt und beeinflussen damit das Leben von vielen Menschen auf entscheidende Weise. Beispielsweise Beschäftigung, Gehälter, Versicherungsraten und Strafverfolgung. Diese Entscheidungsprozesse beinhalten die Gefahr unfair und diskriminierend zu sein [21]. Erklärverfahren und Metriken werden allein nicht in der Lage sein ausreichende, ethische Kontrolle auszuüben. Diese Rolle kann nur durch zusätzliche menschliche Evaluations mit von Menschen gesteuerten Evaluationsstools eingenommen werden.

Literaturverzeichnis

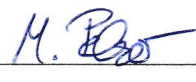
- [1] abin24. *Magnetic-tile-defect-datasets*. URL: <https://github.com/abin24/Magnetic-tile-defect-datasets>. (zitiert auf S. 35).
- [2] M. Alber. *innvestigate*. URL: <https://github.com/albermax/innvestigate> (zitiert auf S. 41).
- [3] M. Asthana. *Layers of a Convolutional Neural Network*. 2020. URL: <https://medium.com/analytics-vidhya/layers-of-a-convolutional-neural-network-168dadd2dd1> (zitiert auf S. 23).
- [4] B. Aunkofer. *Künstliche Intelligenz im Dienst der Wissenschaft*. 2018. URL: https://www.fz-juelich.de/portal/DE/Presse/beitraege/2018/ki-im-dienste-der-wissenschaft/_node.html (zitiert auf S. 20, 21).
- [5] E. W. Bauer. *Humanbiologie:[empfohlen für das 9./10. Schuljahr]. Lehrerhandbuch*. Cornelsen-Velhagen & Klasing, 1975 (zitiert auf S. 22).
- [6] K. Bhardwaj. *Integrated Gradients for Deep Neural Networks*. 2019. URL: <https://medium.com/@kartikayabhardwaj98/integrated-gradients-for-deep-neural-networks-c114e3968eae> (zitiert auf S. 33, 34).
- [7] A. Binder, S. Bach, G. Montavon, K.-R. Müller, W. Samek. „Layer-wise relevance propagation for deep neural network architectures“. In: *Information Science and Applications (ICISA) 2016*. Springer, 2016, S. 913–922 (zitiert auf S. 32, 33).
- [8] T. J. Brinker, A. Hekler, A. H. Enk, C. Berking, S. Haferkamp, A. Hauschild, M. Weichenthal, J. Klode, D. Schadendorf, T. Holland-Letz et al. „Deep neural networks are superior to dermatologists in melanoma image classification“. In: *European Journal of Cancer* 119 (2019), S. 11–17 (zitiert auf S. 15).
- [9] M. Divyanshu. *Demystifying Convolutional Neural Networks using GradCam*. 2019. URL: <https://towardsdatascience.com/demystifying-convolutional-neural-networks-using-gradcam-554a85dd4e48> (zitiert auf S. 26).
- [10] I. Döbel, M. Leis, M. M. Vogelsang, D. Neustroev, H. Petzka, A. Riemer, S. Rüping, A. Voss, M. Wegele, J. Welz. *Maschinelles Lernen: Eine Analyse zu Kompetenzen, Forschung und Anwendung (2018)*. 2019 (zitiert auf S. 17, 25).
- [11] M. S. GmbH. *MVTec Anomaly Detection Dataset* (zitiert auf S. 35, 36).
- [12] I. Goodfellow, Y. Bengio, A. Courville, Y. Bengio. *Deep learning*. Bd. 1. MIT press Cambridge, 2016 (zitiert auf S. 22).
- [13] K. He, X. Zhang, S. Ren, J. Sun. „Deep residual learning for image recognition“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, S. 770–778 (zitiert auf S. 15, 17, 18).

- [14] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam. „Mobilenets: Efficient convolutional neural networks for mobile vision applications“. In: *arXiv preprint arXiv:1704.04861* (2017) (zitiert auf S. 34, 35).
- [15] Y. Huang, C. Qiu, K. Yuan. „Surface defect saliency of magnetic tile“. In: *The Visual Computer* 36.1 (2020), S. 85–96 (zitiert auf S. 37).
- [16] A. Krizhevsky, G. Hinton. „Convolutional deep belief networks on cifar-10“. In: *Unpublished manuscript* 40.7 (2010), S. 1–9 (zitiert auf S. 35).
- [17] A. Krizhevsky, V. Nair, G. Hinton. *The CIFAR-10 Dataset*. URL: <https://www.cs.toronto.edu/~kriz/cifar.html> (zitiert auf S. 35, 36).
- [18] A. Krizhevsky, I. Sutskever, G. E. Hinton. „Imagenet classification with deep convolutional neural networks“. In: *Advances in neural information processing systems*. 2012, S. 1097–1105 (zitiert auf S. 35).
- [19] S. Lundberg. *shap*. URL: <https://github.com/slundberg/shap> (zitiert auf S. 29, 41).
- [20] S. M. Lundberg, S.-I. Lee. „A unified approach to interpreting model predictions“. In: *Advances in neural information processing systems*. 2017, S. 4765–4774 (zitiert auf S. 29).
- [21] S. Mohseni, N. Zarei, E. D. Ragan. „A Multidisciplinary Survey and Framework for Design and Evaluation of Explainable AI Systems“. In: *arXiv* (2018), arXiv–1811 (zitiert auf S. 50).
- [22] C. Molnar. *Interpretable Machine Learning*. 2020. URL: <https://christophm.github.io/interpretable-ml-book/taxonomy-of-interpretability-methods.html> (zitiert auf S. 27, 30).
- [23] G. Montavon, A. Binder, S. Lapuschkin, W. Samek, K.-R. Müller. „Layer-wise relevance propagation: an overview“. In: *Explainable AI: interpreting, explaining and visualizing deep learning*. Springer, 2019, S. 193–209 (zitiert auf S. 49).
- [24] A. Osman, L. Arras, W. Samek. „Towards ground truth evaluation of visual explanations“. In: *arXiv preprint arXiv:2003.07258* (2020) (zitiert auf S. 39, 45).
- [25] F. Pol. *Understanding how LIME explains predictions*. 2018. URL: <https://towardsdatascience.com/understanding-how-lime-explains-predictions-d404e5d1829c> (zitiert auf S. 31).
- [26] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, M. Kloft. „Deep one-class classification“. In: *International conference on machine learning*. 2018, S. 4393–4402 (zitiert auf S. 35).
- [27] S. Saha. *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*. 2018. URL: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53> (zitiert auf S. 25).
- [28] W. Samek, A. Binder, G. Montavon, S. Lapuschkin, K.-R. Müller. „Evaluating the visualization of what a deep neural network has learned“. In: *IEEE transactions on neural networks and learning systems* 28.11 (2016), S. 2660–2673 (zitiert auf S. 38).
- [29] M. Sauermann. *Machine Learning: 3 KI-Lernverfahren auf einen Blick*. 2019. URL: <https://der-onliner.blogspot.com/2019/07/machinelles-lernen-lernverfahren.html> (zitiert auf S. 18).
- [30] N. Schaaf. *Neuronale Netze: Ein Blick in die Black Box*. 2020. URL: <https://www.informatik-aktuell.de/betrieb/kuenstliche-intelligenz/neuronale-netze-ein-blick-in-die-black-box.html> (zitiert auf S. 21).

- [31] T. Schlößer. *Funktionsweise künstlicher neuronaler Netze*. 2018. URL: <https://data-science-blog.com/blog/2018/08/31/funktionsweise-kunstlicher-neuronaler-netze/> (zitiert auf S. 20).
- [32] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra. „Grad-cam: Visual explanations from deep networks via gradient-based localization“. In: *Proceedings of the IEEE international conference on computer vision*. 2017, S. 618–626 (zitiert auf S. 30, 32).
- [33] sicara. *tf-explain*. URL: <https://github.com/sicara/tf-explain> (zitiert auf S. 41).
- [34] K. Simonyan, A. Zisserman. „Very deep convolutional networks for large-scale image recognition“. In: *arXiv preprint arXiv:1409.1556* (2014) (zitiert auf S. 34).
- [35] J. Stallkamp, M. Schlipsing, J. Salmen, M. C. Igel. *GTSRB*. URL: <http://benchmark.ini.rub.de/?section=gtsrb&subsection=news> (zitiert auf S. 35).
- [36] J. Stallkamp, M. Schlipsing, J. Salmen, C. Igel. „The German Traffic Sign Recognition Benchmark: A multi-class classification competition“. In: *IEEE International Joint Conference on Neural Networks*. 2011, S. 1453–1460 (zitiert auf S. 37).
- [37] R. Thakur. *Step by step VGG16 implementation in Keras for beginners*. 2019. URL: <https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c> (zitiert auf S. 34, 35).
- [38] wikipedia. *Convolutional Neural Network*. URL: https://de.wikipedia.org/wiki/Convolutional_Neural_Network (zitiert auf S. 24).
- [39] wikipedia. *Künstliches neuronales Netz*. URL: https://de.wikipedia.org/wiki/K%C3%BCnstliches_neuronales_Netz (zitiert auf S. 19).
- [40] wikipedia. *Nervenzelle*. URL: <https://de.wikipedia.org/wiki/Nervenzelle> (zitiert auf S. 19).
- [41] J. Zech. *What are radiological deep learning models actually learning?* 2018. URL: <https://medium.com/@jrzech/what-are-radiological-deep-learning-models-actually-learning-f97a546c5b98> (zitiert auf S. 25, 26).
- [42] H. Zhang, J. Chen, H. Xue, Q. Zhang. „Towards a unified evaluation of explanation methods without ground truth“. In: *arXiv preprint arXiv:1911.09017* (2019) (zitiert auf S. 39, 46, 47, 49).

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Stuttgart, 18.10.2020, M. P. B. 

Ort, Datum, Unterschrift