Fachbereich Mathematik und Informatik

Bachelorarbeit

Rate-Distortion System zur Erklärung von Entscheidungen von neuronalen Netzwerken

A Rate-Distortion Framework for Explaining Neural Network Decisions

Erstgutachter:	Prof. Dr. Benedikt Wirth
Zweitgutachter:	Dr. Frank Wübbeling
vorgelegt von:	Martin Segeroth
Matrikelnummer:	408783
E-Mail:	martin.segeroth@gmail.com
Studiengang:	Ein-Fach-Bachelor Mathematik
eingereicht in:	Münster, den 9.3.2023

Inhaltsverzeichnis

1 Einleitung

Neuronale Netze nehmen als Deep Learning eine besondere Stellung im Bereich des maschinellen Lernens ein, da sie eine hohe Qualität und ein breites Anwendungsspektrum aufweisen. Sie werden beispielsweise bei Übersetzern (Bsp. Google Translator, DeepL), Bilderkennungsprogrammen, Gesichtserkennung, Segmentierung, etc. eingesetzt. Solange ausreichend Daten vorhanden sind und ein entsprechendes Minimierungsproblem gegeben ist, können neuronale Netze trainiert werden. Ein Beispiel hierfür ist die Unterscheidung von Hunden und Katzen anhand von Bildern. Das entsprechende Minimierungsproblem besteht darin, den Fehler zu minimieren mit dem die falsche Auswahl getroffen wird. Durch entsprechend große Datensätze werden neuronale Netzwerke trainiert, welche eine hohe Genauigkeit aufweisen. Da neuronale Netze wie eine Blackbox fungieren - Daten werden eingegeben und eine Klassifikation, Segmentierung, Text, etc. ausgegeben - ist nicht offensichtlich, welche Kriterien zu einer entsprechenden Ausgabe führen. Die Erkenntnis, welche Kriterien relevant sind, kann die Effizienz eines solchen Netzwerkes in Frage stellen.
Ein großes und stetig wachsendes Anwendungsgebiet von neuronalen Netzwerken stellt die Medizin dar. Solange es sich bei der Klassifikationsaufgabe des Netzwerkes um eine überprüfbare Aufgabe handelt wie z.B. die Segmentierung eines Organs, kann diese leicht von einem Arzt oder Ärztin überprüft werden. Dies kann es erleichtern genauere Angaben in radiologischen-Befunden bezüglich des Lebervolumens [1] (bei Tumorpatienten) oder von Pleuraergüssen [2] zu liefern. Schwieriger zu überprüfen sind jedoch Klassifikationen durch ein neuronales Netzwerk, z.B. Computer-Tomographie (CT) des Thorax als Eingangssignal und Lungenarterienembolie vorhanden (Ja/Nein) als Ausgabe. Viele Ärzte oder Ärztinnen verlassen sich nicht allein auf die Ausgabe eines neuronalen Netzwerkes und bevorzugen es, wenn die relevanten Strukturen zusätzlich markiert sind. Als einfache Beispiele für eine solche Markierung können der Algorithmus von Aidoc zur Erkennung einer intrakraniellen Blutung oder von einer Lungenarterienembolie angeführt werden [3, 4, 5]. Das Hervorheben der relevanten Bereiche schafft hier ein Vertrauen der nutzenden Ärzte oder Ärztinnen in die Korrektheit der ausgegebenen Klassifikation, da sie die Ausgabe auch leicht überprüfen können. Im Gegensatz hierzu werden die relevanten Strukturen oft nicht markiert. Dies kann dazu führen, dass einem Netzwerk eine Genauigkeit zugeschrieben wird, die es nicht aufweist.

1.1 Motivation

Im Gegensatz zu den neuronalen Netzwerken basieren viele konventionelle Algorithmen des maschinellen Lernens, z.B. der „Random Forest“, welcher oft als Klassifizierungsverfahren eingesetzt wird, auf von Menschen ausgewählten Informationen. Dieser Algorithmus verwendet unabhängige bzw. nicht korrelierende Entscheidungsbäume, welche während des Lernvorgangs durch eine spezifische Randomisierung erstellt werden. Für die endgültige Klassifikation wird abschließend die Mehrheitsentscheidung der Entscheidungsbäume benutzt. Eine Weiterentwicklung stellt das „Gradient Boosting“ dar. Hierbei wird den Entscheidungsbäumen jeweils ein zusätzlicher Faktor zugewiesen, um den einzelnen Entscheidungsbäumen eine zusätzliche Gewichtung zuzuweisen. Dies führt dazu, dass „Gradient Boosting“-Algorithmen in der Regel besser klassifizieren als „Random Forests“. Lineare Regression, Entscheidungsbäume („Decision-Trees“) oder k-nächster Nachbar (nearest-neightbor) gehören ebenfalls zu den traditionellen „Maschine-Learning“-Modellen. Diese Methoden des maschinellen Lernens nutzen von Menschen ausgewählte Kriterien im Gegensatz zu neuronalen Netzen, welche die Kriterien selbst auswählen und nur die Eingangsdaten und die entsprechende Klassifikation bzw. das Ergebnis erhalten.
Die konventionellen Algorithmen des maschinellen Lernens erlauben im Vergleich zu hochgradig nichtlinearen, parameterreichen neuronalen Netzwerken eine einfachere menschliche Interpretation der Modellvorhersage [6]. In den letzten Jahren wurden jedoch auf diesem Gebiet deutliche Fortschritte gemacht durch die Einführung von multiplen Erklärungsmodellen für tiefe neuronale Netzwerke. Diese Modelle

liefern zusätzliche Informationen und können dazu beitragen, dass neuronale Netzwerke als vertrauenswürdiger eingeschätzt werden oder noch nicht bekannte Informationen in Zukunft benutzt werden können. Im Folgenden wird eine von Macdonald et al. [6] vorgeschlagene Methode namens „Rate-Distortion System zur Erklärung von Entscheidungen von neuronalen Netzwerken“ untersucht, welche direkt die Klassifikation eines neuronalen Netzwerkes betrachtet, um die relevanten Bildinformationen zu markieren. Zunächst wird anhand eines Beispiels zu COVID-19 die Notwendigkeit von Erklärungsmodellen verdeutlicht. Anschließend werden in Kapitel 2 einige Definitionen der Komplexitätstheorie erläutert, um die Grundlage zum Beweis der Approximierbarkeit der Methode von Macdonald et al. [6] zu schaffen. In Kapitel 3 wird das Problem, die relevantesten Komponenten für einen Klassifikator eines Eingangssignals zu bestimmen, als ein Optimierungsproblem in einem Rate-Distortion System dargestellt. Anschließend wird in Kapitel 4 gezeigt, dass dieses Problem im Rahmen der Komplexitätstheorie schwierig zu lösen und zu approximieren ist, bevor in Kapitel 5 eine heuristische Lösung mit einer Vereinfachung des Problems präsentiert wird. In Kapitel 6 werden andere Methoden zur Berechnung von Relevanzkarten vorgestellt, bevor in Kapitel 7 numerische Experimente mit mehreren Datensätzen durchgeführt werden und die verschiedenen Methoden zur Erstellung von Relevanzkarten verglichen werden. Die Ergebnisse von Macdonald et al. [6] werden dabei mit den hier vorgestellten Ergebnissen verglichen und eingeordnet.
Obwohl die Weiterentwicklung von Computern es erlaubt, zunehmend komplexere Algorithmen auszuführen, ist es essentiell zu zeigen, dass diese Methoden effizient bzw. in polynomieller Zeit von einem Computer ausgeführter werden können.

1.2 Notation

Die Dimension des Signals sei repräsentiert durch $d \in \mathbb{N}$, $x \in [0,1]^d$ ist ein willkürlich fixiertes Eingangssignal und $\Phi : [0,1]^d \to [0,1]$ ist eine Klassifizierungsfunktion für eine Singalklasse $\mathcal{C} \subseteq [0,1]^d$. Die Funktion Φ kann beispielsweise durch ein neuronales Netzwerk beschrieben werden. Der Klassifikationsscore $\Phi(x)$ repräsentiert die Vorhersage des Klassifikators bezüglich der Wahrscheinlichkeit, dass x zu der Klasse $\mathcal{C}$ gehört. Sei $[d] = \{1, ..., d\}$ und für die Teilmenge $S \subseteq [d]$ sei x_S die Beschränkung von x auf die Komponenten, welche durch S indexiert werden. Sei zudem $1_d \in \mathbb{R}$ ein Vektor, welcher nur Einsen enthält. $\text{diag}(x)$ repräsentiert die Diagonalmatrix mit Einträgen von x. Zudem sei $\odot$ (resp. $\oslash$) das Komponentenweise Hadamard-Produkt (resp. Quotient) von Vektoren oder Matrizen der gleichen Dimension. Wir schreiben $x^2 = x \odot x$ und wenden Univariate-Funktionen Komponentenweise auf Vektoren an.

1.3 Beispiel COVID-19 Klassifikation

Als Beispiel sei hier der Versuch angeführt Patienten oder Patientinnen mit COVID-19 von Patienten oder Patientinnen ohne COVID-19 mithilfe von CT-Aufnahmen öder Röntgenaufnahmen zu unterscheiden [7]. 2020-2021 wurde der COVID-19 Ausbruch mit Millionen Betroffenen und Todesfällen weltweit [8] ein zentrales Thema in der medizinischen Forschung. Ein zentrales Problem vor der Einführung von Schnelltests war die schnelle Ausbreitung der Pandemie und das Fehlen eines zeitlich effizienten Tests, welcher schneller als RT-PCR (Reverse Transcription Polymerase Chain Reaction) war. Diese war zudem an vielen Standorten nicht verfügbar. Darüber hinaus führte die hohe Anzahl an Patienten und Patientinnen zu einer vorübergehenden erheblichen Belastung der Krankenhäuser. Ein Algorithmus (Computer Assisted Diagnostic (CAD) Tool), welcher die medizinische Triage durch die Verwendung von einfach zu akquirierenden Datensätzen wie einem CT-Scan oder Röntgenbild erleichtert, erschien wünschenswert. Aus diesem Grund wurden mehrere Röntgen- und CT-Modelle publiziert. In den meisten Fällen basierten die Trainingsdatensätze auf zwei Klassen (COVID vs. gesund) oder drei Klassen (COVID vs. gesund vs. andere thorakale Pathologie). Um diese Trainingsdatensätze zu vergrößern und zu vervollständigen, wurden Daten von mehreren Studien kombiniert und augmentiert. Darüber hinaus wurden Datensätze mit anderen thorakalen Pathologien eingeschlossen, welche vor der COVID-19 Pandemie erstellt wurden. Da diese Daten von verschiedenen Institutionen und Geräten stammen, weisen sie unterschiedliche Charakteristiken hinsichtlich Auflösung, Field of View (FOV), etc. auf. Diese Unterschiede könnten zu einem Bias bei einem auf diesem Datensatz trainierten Klassifikator führen.
Palatnik de Sousa et al. [7] verwendeten einen online verfügbaren Datensatz von Yang et al. [8]. Dieser Datensatz basiert auf mehreren Bilddaten aus Publikationen zu COVID-19 und enthält 746 CT-Aufnahmen. Der Datensatz ist relativ ausgewogen, was die Verteilung der beiden Klassen, kein COVID-19 vs COVID-19, betrifft.

- Trainingsdatensatz: 234 - kein COVID-19 und 191 - COVID-19
- Validationsdatensatz: 58 - kein COVID-19 und 60 - COVID-19
- Testdatensatz: 105 - kein COVID-19 und 98 - COVID-19

Abbildung 1 zeigt Beispiele aus diesem Datensatz. Beim Betrachten dieser Daten fällt auf, dass diese Artefakte aufweisen. Zum Beispiel sind in einigen Bildern der COVID-19-Klasse Buchstaben enthalten (Abb. 1, COVID-19-Klasse a,c,e,f,g) oder Annotationen (Abb. 1, COVID-19-Klasse h). Zudem weisen nicht alle Bilder eine Darstellung in Graustufen auf (Abb. 1, COVID-19-Klasse d). Dies könnte nahelegen, dass ein Netzwerk, welches auf diesem Datensatz trainiert ist, einen Bias aufweist und eben diese Artefakte für die Klassifikation benutzt.
Basierend auf diesem Datensatz wurden mehrere Modelle von neuronalen Netzwerken trainiert und alle möglichen Kombinationen der Modelle wurden betrachtet. Die höchste Genauigkeit (Accuray) über alle Netzwerke lag bei ungefähr 96.6%, und mindestens vier Netzwerke mussten zusammengefügt werden, um diese Genauigkeit zu erreichen. Es wurde ein VGG, ein EfficientNet und zwei DenseNets eingeschlossen. Neben dem kombinierten Netzwerk wurden ebenfalls die vier einzelnen Netzwerke mittels mehrerer Relevanzkarten / Heatmap-Methoden (GradCAM, Integrated Gradients, Vanilla Gradients, Smooth Gradients, RISE, SquareGrid, LIME) zur Darstellung der relevanten Bilddaten für die Klassifikation analysiert [7]. Des Weiteren wird im Folgenden der Begriff Relevanzkarten für die Darstellung der Relevanzwerte benutzt und entspricht damit dem englischen Begriff Heatmap. Die folgenden zwei Abbildungen zeigen die Relevanzkarten für die fünf unterschiedlichen Netzwerke mit und ohne Artefakte.
Für die Fälle mit einem Artefakt fällt auf, dass bei dem VGG-Netzwerk hauptsächlich das „d“ in der linken oberen Ecke relevant für die Klassifikation ist. Alle Relevanzkarten weisen in diesem Bereich hohe positive Werte auf. Diese sind teilweise die Pixel mit den höchsten Werten der Relevanzkarte. Im Gegensatz hierzu

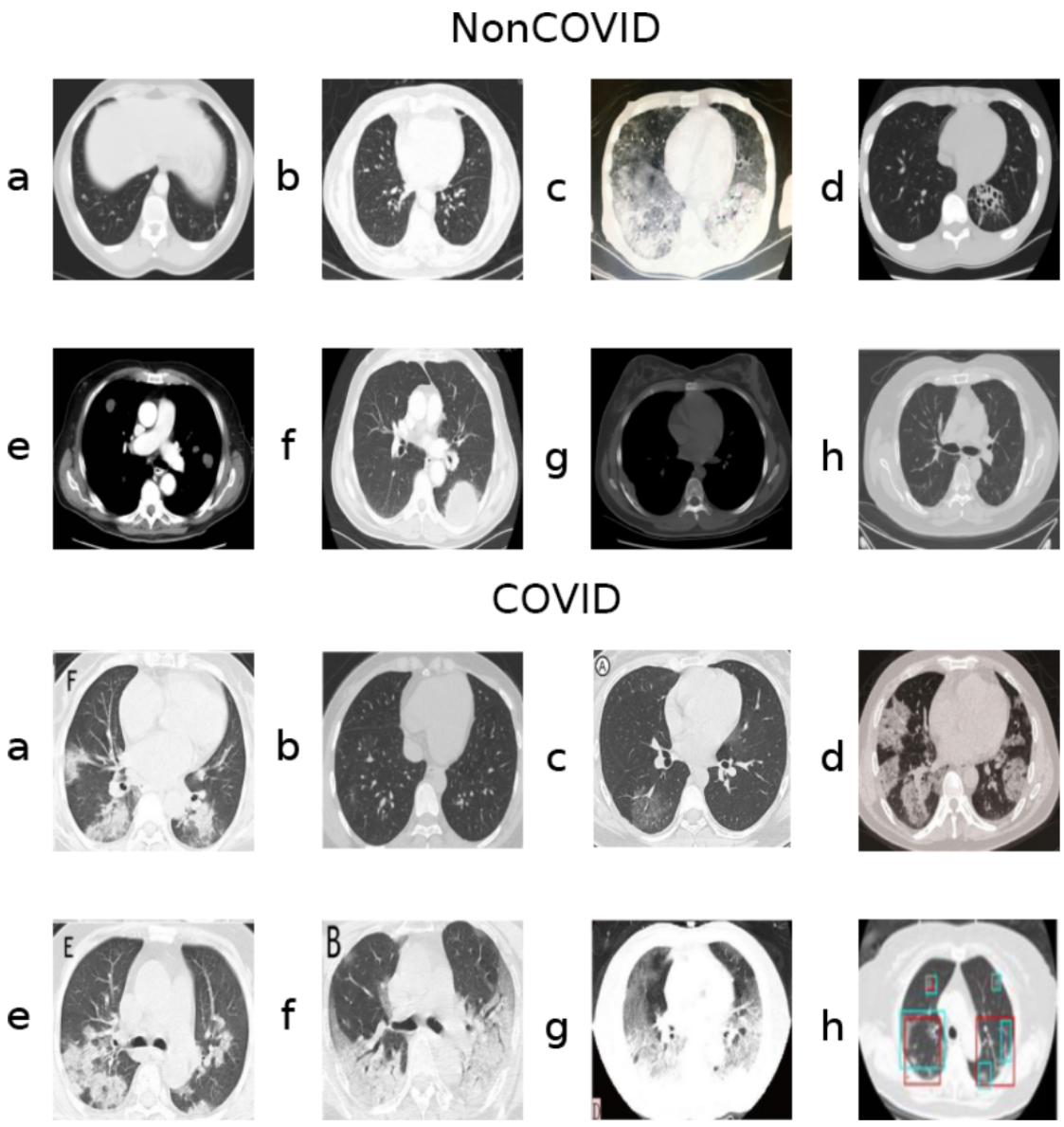

Abbildung 1: Beispiele der beiden Klassen ohne COVID-19 (NonCOVID) und mit COVID-19 (COVID) aus Palatnik de Sousa et al. [7].

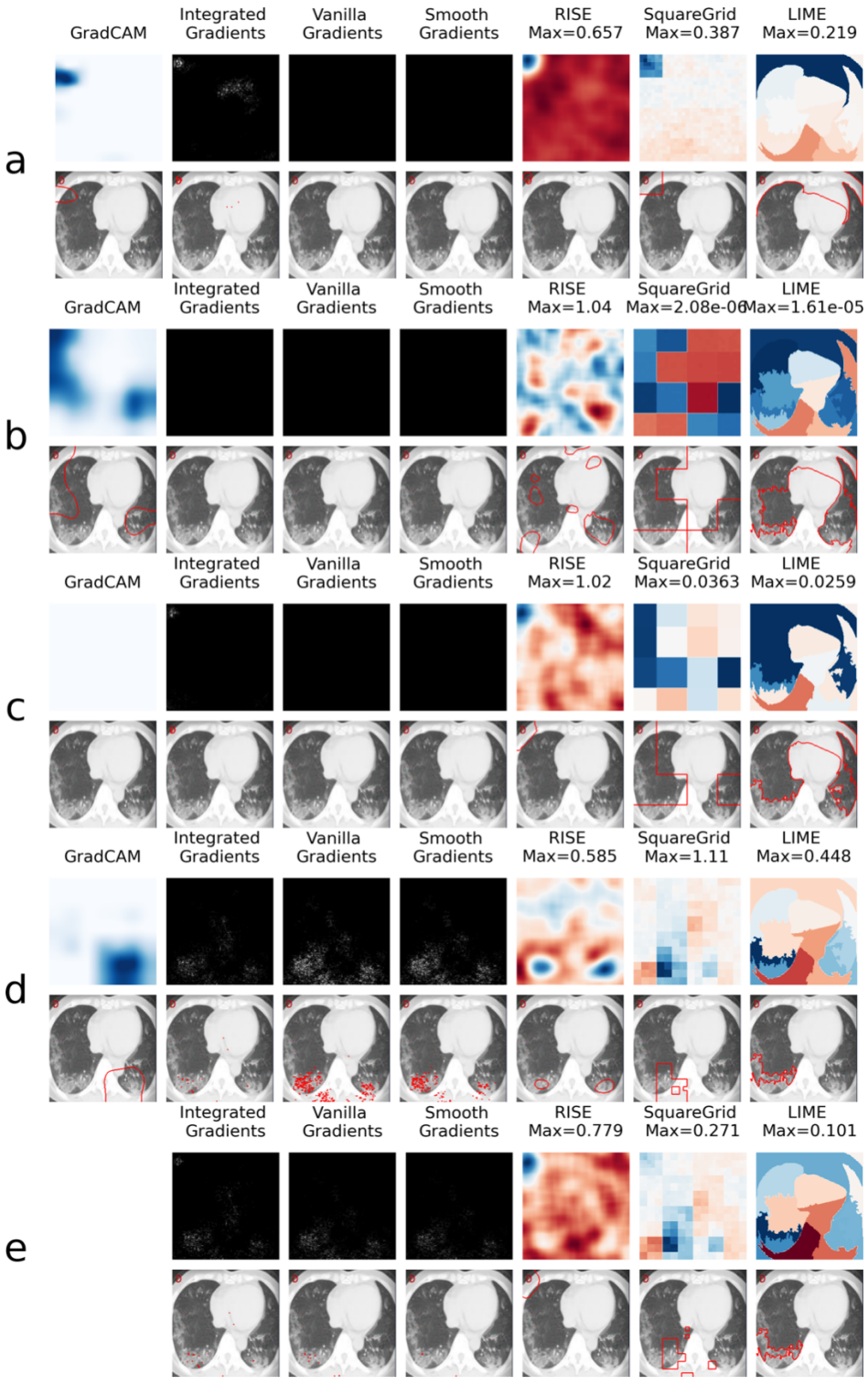

Abbildung 2: Vergleich mehrerer Relevanzkarten von Bildern mit einem Artefakt für das (**a**) VGG-, (**b**) Efficient- und die beiden (**c,d**) Dense-Netzwerke und die (**e**) Kombination (Ensemble), wobei die obere Zeile jeweils die Relevanzkarten zeigt und die untere Zeile die Pixel mit den höchsten Gewichten der Relevanzkarte. GradCAM ist für die Kombination nicht definiert. Die Werte der Relevanzkarten gehen von 0 bis 1 für GradCAM, Minimum bis Maximum für RISE und -Maximum bis Maximum für Squaregrid aus Palatnik de Sousa et al. [7].

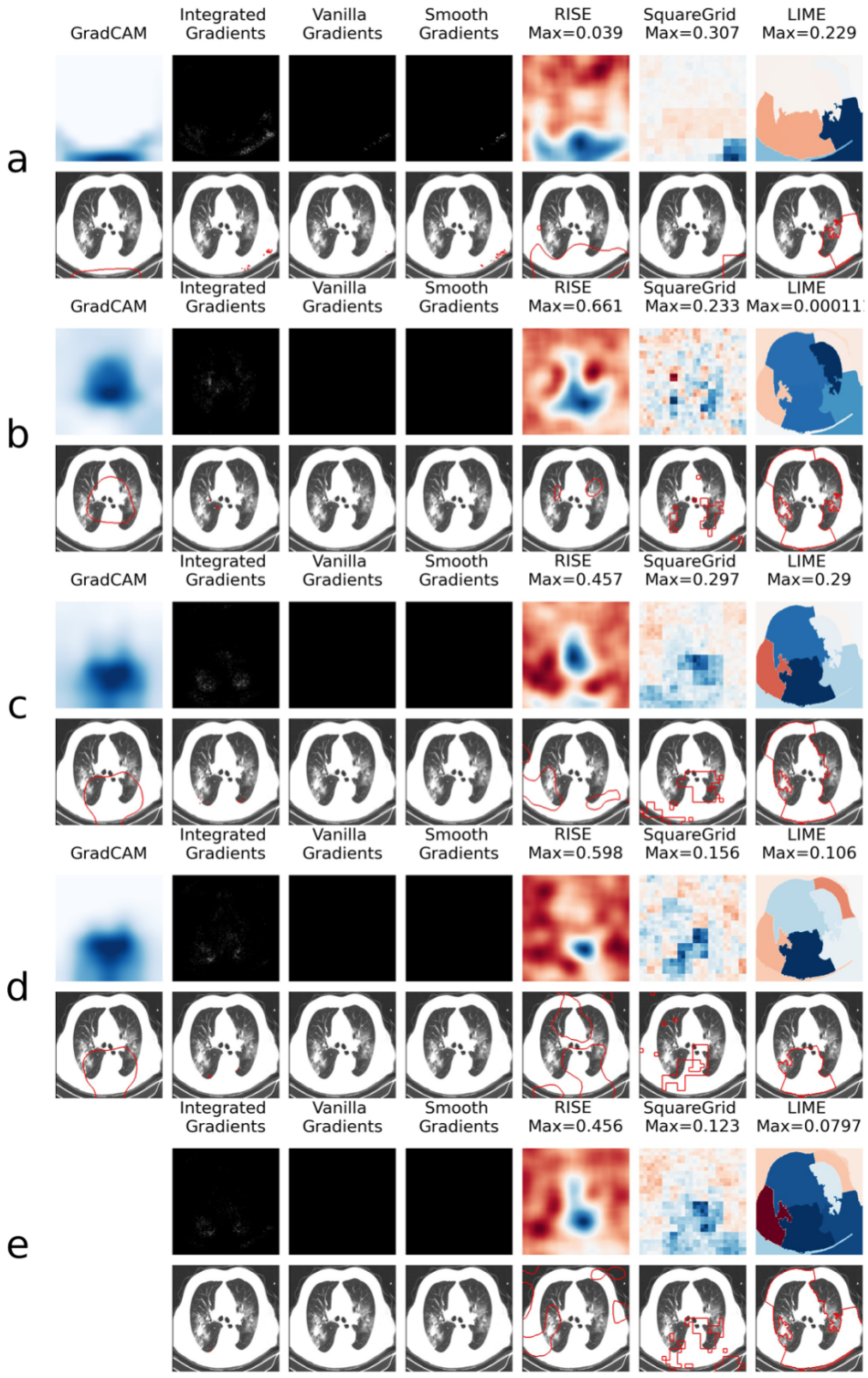

Abbildung 3: Vergleich mehrerer Relevanzkarten von Bildern ohne ein Artefakt für das (**a**) VGG-, (**b**) Efficient- und die beiden (**c,d**) Dense-Netzwerke und die (**e**) Kombination (Ensemble) aus Palatnik de Sousa et al. [7].

konzentriert sich das VGG-Netzwerk bei den Fällen ohne Artefakte verstärkt auf den Bildunterrand, der den CT-Tisch zeigt, auf dem der Patient oder Patientin während der Untersuchung liegt. Retrospektiv betrachtet scheinen viele Bilddaten der COVID-Klasse eben diesen CT-Tisch mit zu erfassen, was diesen Bias erklären könnte. Die anderen Modelle (Efficient- und die beiden Dense-Netzwerke) scheinen weniger von den Artefakten beeinflusst zu werden. Bei diesen Modellen werden oft ebenfalls die Artefakte hervorgehoben, jedoch auch oft die Lungenregion. Es sei angemerkt, dass obwohl die beiden Dense-Netzwerke eine gleiche Architektur aufweisen und sich nur um 1% in der Genauigkeit unterscheiden, sie deutlich abweichende Relevanzkarten aufweisen [7].
Zusammenfassend lässt sich sagen, dass die Auswahl eines geeigneten Trainingsdatensatzes essentiell für die Erstellung eines sinnvollen Netzwerkes zur Klassifikation ist. Bei einer Analyse mittels Relevanzkarten werden die relevanten Bildbereiche hervorgehoben. Somit können durch die Verwendung von Relevanzkarten Artefakte identifiziert werden, welche die Klassifikation beeinflussen. Des Weiteren können selbst geringe Unterschiede in der Genauigkeit zu einer deutlichen Veränderung der erlernten Konzepte eines Netzwerkes führen. Schließlich zeigen die Unterschiede in den Ergebnissen der verschiedenen Relevanzkarten, dass es sinnvoll ist, mehrere Methoden zur Analyse eines Netzwerkes zu verwenden. In dem obigen Beispiel wurde gezeigt, wie wichtig es ist, zu verstehen, wie ein neuronales Netzwerk oder Klassifikator Entscheidungen trifft, um sicher zu stellen, dass diese Entscheidungen auf sinnvollen und korrekten Informationen basieren.

2 Einführung Komplexitätstheorie

In dieser Arbeit wird die Methode von Macdonald et al. [6] im Rahmen der Komplexitätstheorie analysiert, weshalb zunächst auf diese eingegangen wird. Ein Algorithmus ist eine Handlungsvorschrift, die aus wohldefinierten Einzelschritten besteht und zur Lösung eines Problems oder einer Klasse von Problemen dient [9]. Wenn ein Computer diese Einzelschritte ausführen kann, ist es möglich, den Algorithmus in den Computer zu implementieren. Bei der Lösung des Problems wird die Eingabe in eine bestimmte Ausgabe überführt.
Um diese Definition zu präzisieren, wurden Ende des 20. Jahrhunderts mehrere Ansätze entwickelt. Die Turingmaschine von Alan Turing ermöglichte folgende formale Definition eines Algorithmus [10]:

„Eine Berechnungsvorschrift zur Lösung eines Problems heißt genau dann Algorithmus, wenn eine zu dieser Berechnungsvorschrift äquivalente Turingmaschine existiert, die für jede Eingabe, die eine Lösung besitzt, stoppt.“ [10]

Eine Turingmaschine ist ein Modell der theoretischen Informatik und ist eine abstrakte Maschine, welche nach vorbestimmten Regeln schrittweise Manipulationen von Symbolen und Zeichen durchführt. Diese Symbole und Zeichen werden hierbei gemäß konkreter Regeln auf ein Speicherband geschrieben und von dort abgelesen [11].

Das Ziel besteht darin, einen Algorithmus zu finden, der dazu beitragen kann, dass die Entscheidungen eines neuronalen Netzwerkes von einem Menschen nachvollzogen werden kann. Dieser soll hierbei aufzeigen, welche Eingabeparameter / Anteile des Eingangssignals der Eingabe die Ausgabe des neuronalen Netzwerkes hauptsächlich beeinflussen. Der Nachweis, dass das Problem einer solchen Erklärung durch eine bestimmte Methode von einem Computer in effizienter Zeit gelöst werden kann, erlaubt es anzunehmen, dass ein solcher Algorithmus existiert und konstruiert werden kann. Mit dieser Fragestellung befasst sich die Komplexitätstheorie der Informatik.

Ist es möglich, für eine konkrete Aufgabenstellungen ein möglichst effizientes Verfahren anzugeben, liefert dies eine obere Schranke für die Komplexität des Problems, d.h. es stellt für einen Computer eine Laufzeitbegrenzung dar. Oft wird versucht, eine untere Schranke anzugeben, d.h. wie lange es mindestens dauert. Trivialerweise gilt oft die Länge der Eingabe d als untere Schranke für die Zeitkomplexität, da ein Algorithmus, der ein Problem korrekt löst, zumindest die Eingabe vollständig erfassen muss, was d Schritte erfordert.
Um zu zeigen, dass ein Problem in begrenzter Zeit gelöst werden kann, wird die Äquivalenz von nichtdeterministischen und deterministischen Berechnungsmodellen betrachtet. Hierzu wird die Frage $P = NP$ oder $P \neq NP$ betrachtet. Informell bezeichnet P die Menge der schnell lösbaren Probleme und NP die Menge der Probleme, für die eine mögliche Lösung schnell auf Korrektheit überprüft werden kann. Schnell bedeutet, dass die Anzahl der Rechenschritte polynomiell durch die Länge der Eingabe beschränkt ist. Da dies essenziell ist, um zu zeigen, dass ein Problem in effizienter Zeit lösbar ist, werden im Folgenden P und NP formal definiert, nachdem auf einige andere Definitionen eingegangen wurde.

Definition 2.1 (Alphabet)**.** *Eine endliche, nicht leere Menge wird als* ***Alphabet*** *bezeichnet. Die Elemente eines Alphabetes heißen* ***Symbole*** *[12, Anhang].*

Definition 2.2 (Sprache)**.** *Sei* Σ *ein Alphabet. Eine* ***Sprache*** *über* Σ *ist jede beliebige Teilmenge von* Σ^* *[12, Kap. 1.1].*

Σ^* ist die Menge aller Wörter, die sich durch Verkettung von Symbolen aus Σ ergeben. Wie zu Beginn von 2 erwähnt, ist eine Turingmaschine, ein mathematisches Modell der theoretischen Informatik, eine

abstrakte Maschine. Diese Maschine nimmt nach festgelegten Regeln und Zuständen Manipulationen an Symbolen vor. Formal lässt es sich wie folgt definieren:

Definition 2.3 (Deterministische Turingmaschine)**.** *Eine* ***deterministische Turingmaschine*** *[12, Kap. 1.4],[11] ist gegeben durch ein 7-Tupel*

$$M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$$

Hierbei sind:

- Z *die endliche* ***Zustandsmenge***
- Σ *das* ***Eingabealphabet***
- $\Gamma \supset \Sigma$ *das* ***Arbeitsalphabet / Bandalphabet***
- $\delta : (Z \setminus E) \times \Gamma \to Z \times \Gamma \times \{L, R, N\}$ *die* ***Überführungsfunktion***
- $z_0 \in Z$ *der* ***Startzustand***
- $\square \in \Gamma - \Sigma$ *das* ***Blank***
- $E \subseteq Z$ *die Menge der* ***Endzustände***

Die Übergangsfunktion δ gibt zu einem Zustand und einem gelesenen Bandsymbol den nächsten Zustand an. Zudem wird ein Bandsymbol in das aktuelle Feld geschrieben und die Bewegung des Lese-Schreib-Kopfes angegeben. Die Bewegungsrichtung des Lese-Schreib-Kopfes wird mit L : ein Feld nach links, R : ein Feld nach rechts und N : nicht bewegen angegeben. Ein gesteuerter Lese-Schreib-Kopf kann Zeichen auf dem Speicherband verändern (im Fall eines zu ‚schreibenden' Blanks auch löschen) und sich feldweise bewegen. Eine deterministische Mehrband-Turingmaschine unterscheidet sich in der Definition der deterministischen Turingmaschine nur in der Definition der Übergangsfunktion δ.

Definition 2.4 (Deterministische Mehrband-Turingmaschine)**.** *Eine* ***deterministische Mehrband-Turingmaschine*** *ist eine deterministische Turingmaschine mit Übergangsfunktion δ:*

$$\delta : (Z \setminus E) \times \Gamma^k \to Z \times \Gamma^k \times \{L, R, N\}^k$$

Die Übergangsfunktion δ stellt in dem Fall der deterministischen Mehrband-Turingmaschine den Übergang von k verschiedenen Bändern gelesene Bandsymbole in den nächsten Zustand dar. Hierbei wird der nächste Zustand, die k aktuellen Bandsymbole und die k unterschiedlichen Bewegungsrichtungen der Lese-Schreib-Köpfe angegeben. Somit besteht der Unterschied zur klassischen Turingmaschine darin, dass anstatt einem Symbol k Symbole gelesen und geschrieben werden auf k Bändern mit k Lese-Schreib-Köpfen.
Turingmaschinen können auch nichtdeterministisch definiert werden. Hierbei wird anstatt der Übergangsfunktion eine Übergangsrelation benutzt. Somit definiert sich eine nichtdeterministische Mehrband-Turingmaschine mit k Bändern wie folgt:

Definition 2.5 (Nichtdeterministische Mehrband-Turingmaschine)**.** *Eine* ***nichtdeterministische Mehrband-Turingmaschine*** *ist eine deterministische Mehrband-Turingmaschine mit Übergangsrelation δ:*

$$\delta \subseteq (Z \setminus E) \times \Gamma^k \times Z \times \Gamma^k \times \{L, R, N\}^k$$

Definition 2.6 (Klasse $TIME(f(n))$)**.** *Sei $f : \mathbb{N} \to \mathbb{N}$ eine Funktion. Die* ***Klasse*** *$TIME(f(n))$ besteht aus allen Sprachen A, für die es eine deterministische Mehrband-Turingmaschine M gibt mit $A = T(M)$ (d.h. A ist eine von der Mehrband-Turingmaschine akzeptierte Sprache) und $time_M(x) \leq f(|x|)$ [12, Kap. 3.1].*

$time_M : \Sigma^* \to \mathbb{N}$ bezeichnet hierbei die Anzahl der Rechenschritte von M bei Eingabe x.
Wenn die Operation des Quadrierens nicht aus einer Klasse herausführen soll, ist es relevant, ob Einband- oder Mehrbandmaschinen verwendet werden. Aus diesem Grund wurde die Definition über Mehrbandmaschinen gewählt. Somit kann die Klasse der Polynome betrachtet werden und die zugehörige Komplexitätsklasse P definiert werden.

Definition 2.7 (Polynom)**.** *Ein* ***Polynom*** *ist eine Funktion $p : \mathbb{N} \to \mathbb{N}$ der Form*

$$p(n) = a_k n^k + a_{k-1} n^{k-1} + ... + a_1 n + a_o, a_i \in \mathbb{N}, k \in \mathbb{N}$$

Definition 2.8 (Komplexitätsklasse P)**.** *Die* ***Komplexitätsklasse*** *P ist definiert durch [12, Kap. 3.1]:*

$$P = \{A| \text{ es gibt eine Turingmaschine } M \text{ und ein Polynom } p \text{ mit}$$
$$T(M) = A \text{ und } time_M(x) \leq p(|x|)\}$$
$$= \bigcup_{p \text{ Polynom}} TIME(p(n))$$

Es sei angemerkt, dass um zu zeigen, dass ein Algorithmus polynomielle Komplexität aufweist, es genügt zu zeigen, dass seine Komplexität $O(n^k)$ für eine Konstante k ist.
Durch Ausdehnung auf nicht-deterministische Turingmaschinen der oben gegebenen Funktionen kann die Komplexitätsklasse NP definiert werden. Hierbei bezeichnet eine akzeptierte Rechnung eine mit einer Startkonfiguration beginnende Folge von Konfigurationen, welche sukzessiv verknüpft sind, sodass die Folge mit einer Konfiguration im Endzustand endet.
Für nicht-deterministische Turingmaschinen M sei

$$ntime_M(x) = \begin{cases} \text{min [Länge einer akzeptierenden Rechnung von } M \text{ auf } x], & x \in T(M) \\ 0, & x \notin T(M) \end{cases}$$

Sei $f : \mathbb{N} \to \mathbb{N}$ eine Funktion. Die Klasse $NTIME(f(n))$ besteht aus allen Sprachen A, für die es eine nicht-deterministische Mehrband-Turingmaschine M gibt mit $A = T(M)$ und $ntime_M(x) \leq f(|x|)$.

Definition 2.9 (Komplexitätsklasse NP)**.** *Die* ***Komplexitätsklasse*** *NP ist dann definiert durch [12, Kap. 3.1]:*

$$NP = \bigcup_{p \text{ Polynom}} NTIME(p(n))$$

Es ist klar, dass $P \subseteq NP$ ist, aber ob Gleichheit zwischen den beiden Komplexitätsklassen vorliegt oder nicht wird als $P - NP-$Problem bezeichnet. Dieses Problem ist seit ca. 1970 bekannt und ungelöst und wird als wichtigste Frage der theoretischen Informatik angesehen [13].

Definition 2.10 (Polynomiell reduzierbar)**.** *Seinen $A \subseteq \Sigma^*$ und $B \subseteq \Gamma^*$ Sprachen. Dann heißt A auf B* ***polynomiell reduzierbar*** *$A \preceq_p B$ falls es eine totale und mit polynomialer Komplexität berechenbare Funktion $f : \Sigma^* \to \Gamma^*$ gibt, sodass $\forall x \in \Sigma^*$ gilt [12, Kap. 3.2]:*

$$x \in A \Leftrightarrow f(x) \in B.$$

Lemma 2.1. *Falls $A \preceq_p B$ und $B \in P$ dann folgt $A \in P$. Ebenso, falls $A \preceq_p B$ und $B \in NP$ dann folgt $A \in NP$ [12, Kap. 3.2].*

Beweis. Sei $A \preceq_p B$. Somit gibt es eine totale und mit polynomialer Komplexität berechenbare Funktion f, welche sich durch eine Turingmaschine M_f berechnen lässt. Das Polynom p begrenze die Rechenzeit von M_f. Des Weiteren sei $B \in P$ berechenbar durch die Turingmaschine M. Die Rechenzeit von M sei durch das Polynom q begrenzt. Somit ist auch die Verkettung von M_f und M ein polynomial beschränkter Algorithmus, wobei die Rechenzeit bei einer Eingabe x beschränkt ist:

$$p(|x|) + q(|f(x)|) \leq p(|x|) + q(p(|x|))$$

Diese Funktion ist ebenfalls ein Polynom und die Verkettung berechnet A. □

Hierdurch lässt sich nun die $NP-$Vollständigkeit definieren. Intuitiv ist eine Sprache A $NP-$vollständig, falls die Sprache A mindestens so schwierig wie jedes Problem in NP ist.

Definition 2.11 (NP-schwer)**.** *Eine Sprache A heißt **NP-schwer**, falls für alle Sprachen $L \in NP$ gilt: $L \preceq_p A$ [12, Kap. 3.2].*

Definition 2.12 (NP-vollständig)**.** *Eine Sprache A heißt **NP-vollständig**, falls für alle Sprachen $L \in NP$ gilt: $L \preceq_p A$ und $A \in NP$ [12, Kap. 3.2].*

Satz 2.1. *Sei A eine $NP-$vollständige Sprache. Dann gilt [12, Kap. 3.2]:*

$$A \in P \text{ genau dann wenn } P = NP$$

Beweis. Sei $A \in P$ und sei L eine beliebige Sprache in NP. Dann gilt $L \preceq_p A$, da A $NP-$schwer ist. Dann gilt mittels des vorhergegangenen Lemma $L \in P$. Da L beliebig aus NP, folgt $P = NP$. Sei umgekehrt $P = NP$, dann ist, da $A \in NP$ auch $A \in P$. □

Mit Hilfe der oben genannten Definitionen lassen sich bereits Algorithmen charakterisieren. In einem realistischen Szenario gibt es jedoch das Problem, dass eine Problemlösung durch einen Algorithmus unsichere Eingangsinformationen aufweisen kann. Dementsprechend kann mit probabilistischer Komplexität neuronale Netzwerke bzw. die damit einhergehenden Probleme noch genauer charakterisiert werden. Hierzu wird im Folgenden die die Komplexitätsklasse NP^{PP} eingeführt, welche häufiger in Aufgaben zu künstlichen Intelligenzen verwendet wird z.B.: Optimierung unter Unsicherheit.

Definition 2.13 (Probabilistische Turingmaschine)**.** *Eine **probabilistische Turingmaschine** [14] ist definiert durch eine deterministische Turingmaschine $M = (Z, \Sigma, \Gamma, \delta_1, \delta_2, z_0, \square, E)$, welche statt einer Übergangsfunktion δ zwei Übergangsfunktionen hat und bei jedem Rechenschritt einer der beiden Übergangsfunktionen wählt.*

- $\delta_1 : Z \times \Gamma \to Z \times \Gamma \times \{L, R, N\}$
- $\delta_2 : Z \times \Gamma \to Z \times \Gamma \times \{L, R, N\}$

Die Wahl der Übergangsfunktion kann beispielsweise durch eine Bernoulli-verteilte Zufallsvariable $\kappa \sim \text{Ber}(p)$ erfolgen, wobei p die Wahrscheinlichkeit für eines von den beiden δ_1 oder δ_2 ist.

Definition 2.14 (Komplexitätsklasse PP)**.** *Die **Komplexitätsklasse** PP [15, 16] ist die Klasse aller Entscheidungsprobleme, die von einer probabilistischen Turingmaschine in polynomieller Zeit gelöst werden können und die Antwort in der Hälfte der Fälle richtig ist. Eine Sprache L ist somit in PP falls eine probabilistische Turingmaschine M existiert, sodass:*

- *M verarbeitet in polynomieller Zeit alle Eingänge.*
- *$\forall x \in L$ gibt M 1 aus mit einer Wahrscheinlichkeit strikt größer als $\frac{1}{2}$.*
- *$\forall x \notin L$ gibt M 1 aus mit einer Wahrscheinlichkeit strikt weniger als $\frac{1}{2}$.*

Oder anders formuliert ein Entscheidungsproblem ist in PP, falls es einen Algorithmus gibt, dem es erlaubt ist, mittels eines Münzwurfes zufällige Entscheidungen zu treffen und es in polynomieller Zeit löst. Zudem muss, falls die Antwort auf das Problem JA lautet der Algorithmus mit einer Wahrscheinlichkeit über $\frac{1}{2}$ ebenfalls ein JA ausgeben. Dasselbe gilt für den umgekehrten Fall eines NEIN als Antwort auf das Problem.
Die Klasse NP^{PP} ist die Klasse aller entscheidbaren Probleme mittels einer nicht-deterministischen Turing Maschine mit Zugang zu einem Orakel für Probleme in PP [6]. Die Klasse entspricht somit dem $\leq_m^{NP}$-Abschluss von PP [17],[18, Kap. 1.4]. Dies kann man als den Abschluss von PP unter disjunkter Reduzierbarkeit in polynomieller Zeit mit einer exponentiellen Anzahl von Anfragen bezeichnen, wobei die jeweiligen Anfragen in polynomieller Zeit berechenbar sind, abhängig vom Index aus der Liste der Anfragen [18]. Als Beispiel für ein Entscheidungsproblem in der Klasse NP^{PP} kann das E-MAJSAT [18, Kap. 1.4] angegeben werden. Es wird angenommen, dass NP^{PP} signifikant härter als NP und PP ist [6, Kap. 3.1]. Die Komplexitätsklasse NP^{PP} ist somit eine Kombination von NP und PP. Es ist somit die Klasse der Probleme, welche durch Raten einer Lösung gelöst werden kann. Diese Lösung kann dann mittels einer Rechnung in PP überprüft werden [19, Kap. 1.3]. Somit gilt:

$$NP \subseteq PP \subseteq NP^{PP}$$

3 Rate-Distortion

Im Folgenden wird die von Macdonald et al. [6] vorgeschlagene Methode, die relevantesten Komponenten für einen Klassifikator eines Eingangssignals zu bestimmen, als ein Optimierungsproblem in einem Rate-Distortion System dargestellt. Die Aufgabe der Erklärung der Entscheidung eines Klassifikators $\Phi(x)$ wird als Finden der Partition der Komponenten von $x \in [0,1]^d$ einer Teilmenge $S \subseteq [d]$ von relevanten Komponenten und des Komplements S^C von nicht-relevanten Komponenten betrachtet. Die Partition soll so gewählt werden, dass das Fixieren der relevanten Komponenten bereits das Ergebnis des Klassifikators bestimmt für fast alle möglichen Werte der nicht relevanten Komponenten. Mit anderen Worten: Um zu erklären, wie ein Klassifikator, der ein neuronales Netzwerk sein kann, zu einer Entscheidung kommt, wird das Eingangssignal analysiert. Das Eingangssignal soll in relevante und irrelevante Komponenten für die Klassifikation aufgeteilt werden. Die relevanten Komponenten sind relevant, weil sie die Klassifikation hauptsächlich beeinflussen im Gegensatz zu den irrelevanten. Als Beispiel sei hier eine Hund-Katze Bildklassifikation aufgeführt. Das Pixel in der rechten oberen Ecke, welches nicht Teil der Abbildung eines Tieres ist, ist in den meisten Fällen intuitiv irrelevant, wobei die zentralen Pixel in der Bildmitte, welche das Tier abbilden, meistens relevant sind. Um dies zu formalisieren sei $\mathcal{V}$ eine Wahrscheinlichkeitsverteilung auf $[0,1]^d$ und $\mathbf{n} \sim \mathcal{V}$ ein Zufallsvektor. Dann definiert sich eine Trübung wie folgt:

Definition 3.1 (Trübung)**.** ***Trübung*** *von x in Bezug auf S und $\mathcal{V}$:*
Sei y ein Zufallsvektor, welcher deterministisch auf S definiert ist, sodass gilt $y_S = x_S$. Auf dem Komplement S^C sei $y_{S^C} = \boldsymbol{n}_{S^C}$ mit $\boldsymbol{n} \sim \mathcal{V}$ als Zufallsvektor. Dann ist y die Trübung von x in Bezug auf S und $\mathcal{V}$. $\mathcal{V}_S$ ist die resultierende Wahrscheinlichkeitsverteilung von y.

Somit entsprechen die Komponenten von y den Komponenten von x, falls die entsprechende Komponente in der Indexmenge S enthalten ist. Falls die entsprechende Komponente nicht in S enthalten ist, entspricht sie dem Wert eines Zufallsvektors. Hiermit ist das Wissen über die Eingangswerte auf die Indexmenge S beschränkt, da alle anderen Komponenten zufällig gewählt werden. Mit dem auf S eingeschränkten Wissen über das Signal und zufälliger Änderung des restlichen Signals bleibt die Klassenvorhersage erhalten, falls x_S die relevanten Informationen für die Entscheidung des Klassifikators enthält. Im Idealfall würde die Einschränkung auf die Komponenten S weiterhin das gleich Ergebnis des Klassifikators ergeben. Je kleiner $|S|$ ist desto unwahrscheinlicher ist dies. Aus diesem Grund wird die Änderung der Vorhersage das Klassifikators mit dem quadratischen Abstand bemessen wird und stellt die Distortion dar.

Definition 3.2 (Distortion)**.** *Erwartete **Distortion** von S in Bezug auf Φ, x und $\mathcal{V}$:*

$$D(S) = D(S, \Phi, x, \mathcal{V}) = \mathbb{E}_{y \sim \mathcal{V}_S}\left[\frac{1}{2}(\Phi(x) - \Phi(y))^2\right].$$

Hierbei werden durch S die relevanten Komponenten definiert und S^C definiert die irrelevanten Komponenten. In Bezug auf die relevanten Komponenten sind x und y gleich, jedoch nicht in Bezug auf die irrelevanten Komponenten. Hier ist y zufällig, gegeben durch die Wahrscheinlichkeitsverteilung $\mathcal{V}$. Wenn S die für den Klassifikator relevanten Komponenten enthält wird $\Phi(x)$ sehr oft gleich $\Phi(y)$ sein, sodass die Distortion klein ist. Falls S nicht die relevanten Komponenten enthält ist die Distortion groß. Hierbei sei angemerkt, dass auch die Mächtigkeit von S einen maßgeblichen Einfluss auf die Distortion hat. Beispielsweise, wenn S alle Komponenten des Eingangssignal enthält, so ist die Distortion Null.
Somit ergibt sich ein Rate-Distortion Trade-off, welcher intuitiv ein Maß der Relevanz darstellt.

Die Terminologie ist angelehnt an die Informationstheorie. Hierbei wird die Rate-Distortion benutzt, um verlustbehaftete Datenkompression zu analysieren. Die relevanten Komponenten in einer komprimierten

Beschreibung des Signals und der erwarteten Abweichung vom Klassifikationsscore ist ein Maß für den Rekonstruktionsfehler.

Definition 3.3 (Rate-Distortion Funktion)**.** *Die* ***Rate-Distortion Funktion*** *ist wie folgt definiert:*

$$R(\epsilon) = min\{|S| : S \subseteq [d], D(S) \leq \epsilon\}$$

Die kleinste Teilmenge S, die eine beschränkte Distortion aufweist, sollte sich aus den relevantesten Komponenten zusammensetzen. Hierbei werden die Mengen S gesucht, welche die kleinste Mächtigkeit aufweisen und deren Distortion kleiner als ein gegebenes ϵ ist. Diese Rate-Distortion Funktion wird später genutzt, um die Leistung verschiedener Erklärungsmodelle zu vergleichen.

Es ist jedoch schwierig, eine solche Menge S, deren Mächtigkeit minimal ist und dabei eine geringe Distortion hat, zu finden. Diese Problematik wird im Folgenden betrachtet.

Lemma 3.1. *Die triviale Lösung $S = [d]$ hat keine Distortion.*

Beweis. Sei $S = [d]$ dann gilt:

$$y_S = x_S \Leftrightarrow y_{[d]} = x_{[d]} \Leftrightarrow y = x$$

$$\Rightarrow D(S) = \mathbb{E}_{y \sim \mathcal{V}_S}\left[\frac{1}{2}(\Phi(x) - \Phi(y))^2\right] = \mathbb{E}_{y \sim \mathcal{V}_S}\left[\frac{1}{2}(\Phi(x) - \Phi(x))^2\right] = 0$$

□

Im Folgenden wird gezeigt, dass für Distortionen größer als Null nicht systematisch eine Teilmenge von relevanten Komponenten gefunden werden kann, die signifikant kleiner als die triviale Lösung ist. Dies gilt insbesondere, wenn Φ ein neuronales Netzwerk repräsentiert, was hier von besonderem Interesse ist. Die Härteergebnisse werden für den speziellen Fall von Booleschen Schaltkreisen (Boolean circuits) gezeigt. Die Booleschen Schaltkreise können repräsentiert werden durch ReLU neuronale Netzwerke von moderater Größe.

4 Komplexitätsanalyse

Im Folgenden soll gezeigt werden, unter welchen Umständen das oben beschriebene Problem - das Finden einer möglichst kleinen Menge von relevanten Komponenten des Eingangssignals - durch einen Algorithmus in effizienter (polynomialer) Zeit gelöst werden kann. Um dies zu zeigen wird das Problem in diskreter Form definiert.
Sei im Folgenden der spezielle Fall eines binären Eingangssignals $x \in \{0,1\}^d$ angenommen und die Klassifizierungsfunktion $\Phi : \{0,1\}^d \to \{0,1\}$ beschreibe einen Booleschen Schaltkreis (Boolean circuit). Zudem gelte die Gleichverteilung über einen binären Vektor, d.h. $\mathcal{V} = U(\{0,1\}^d)$.

4.1 Diskrete Problemformulierung

Definition 4.1 ($\delta-$*Relevanz*)**.** *Eine Teilmenge $S \subseteq [d]$ heißt $\delta-$**relevante** Menge für Φ und x, falls*

$$\mathbb{P}_y(\Phi(y) = \Phi(x)|y_S = x_S) \geq \delta.$$

Lemma 4.1. *Es gilt, dass eine Menge S $\delta-$relevant ist genau dann, wenn sie eine beschränkte Distortion aufweist mit $D(S) \leq \frac{1-\delta}{2}$.*

Beweis. Im Falle, dass $\Phi : \{0,1\}^d \to \{0,1\}$ einen Booleschen Schaltkreis beschreibt, sei zunächst $S \subseteq [d]$ $\delta-$relevant für Φ und x, dann gilt:

$$D(S) = \mathbb{E}_{y\sim\mathcal{V}_S}\left[\frac{1}{2}(\Phi(x) - \Phi(y))^2\right]$$

$$= 0 \cdot \mathbb{P}_{y\sim\mathcal{V}_S}(\Phi(x) = \Phi(y)) + \frac{1}{2} \cdot 1 \cdot \mathbb{P}_{y\sim\mathcal{V}_S}(\Phi(x) \neq \Phi(y))$$

$$= \frac{1}{2} \cdot \mathbb{P}_{y\sim\mathcal{V}_S}(\Phi(x) \neq \Phi(y)) = \frac{1}{2} \cdot (1 - \mathbb{P}_{y\sim\mathcal{V}_S}(\Phi(x) = \Phi(y)))$$

$$= \frac{(1 - \mathbb{P}_y(\Phi(x) = \Phi(y)|y_S = x_S))}{2} \leq \frac{1-\delta}{2}$$

Somit wurde gezeigt, dass wenn eine Menge S $\delta-$relevant ist, sie eine beschränkte Distortion aufweist mit $D(S) \leq \frac{1-\delta}{2}$.
Sei nun die Distortion beschränkt mit $D(S) \leq \frac{1-\delta}{2}$ dann gilt:

$$D(S) = \mathbb{E}_{y\sim\mathcal{V}_S}\left[\frac{1}{2}(\Phi(x) - \Phi(y))^2\right] \leq \frac{1-\delta}{2}$$

$$\Leftrightarrow 0 \cdot \mathbb{P}_{y\sim\mathcal{V}_S}(\Phi(x) = \Phi(y)) + \frac{1}{2} \cdot 1 \cdot \mathbb{P}_{y\sim\mathcal{V}_S}(\Phi(x) \neq \Phi(y)) \leq \frac{1-\delta}{2}$$

$$\Leftrightarrow \frac{1}{2} \cdot \mathbb{P}_{y\sim\mathcal{V}_S}(\Phi(x) \neq \Phi(y)) \leq \frac{1-\delta}{2}$$

$$\Leftrightarrow \frac{1}{2} \cdot (1 - \mathbb{P}_{y\sim\mathcal{V}_S}(\Phi(x) = \Phi(y))) \leq \frac{1-\delta}{2}$$

$$\Leftrightarrow \frac{(1 - \mathbb{P}_y(\Phi(x) = \Phi(y)|y_S = x_S))}{2} \leq \frac{1-\delta}{2}$$

$$\Leftarrow -\mathbb{P}_y(\Phi(x) = \Phi(y)|y_S = x_S) \leq -\delta$$

$$\mathbb{P}_y(\Phi(x) = \Phi(y)|y_S = x_S) \geq \delta$$

Andersherum wurde gezeigt, dass wenn eine Menge S eine beschränkte Distortion aufweist mit $D(S) \leq \frac{1-\delta}{2}$ sie $\delta-$relevant ist. □

Somit ist das Berechnen der Rate-Distortion-Funktion im Grunde das Gleiche wie die Aufgabe, die relevanten Mengen mit der kleinsten Mächtigkeit zu finden.

Definition 4.2 (Relevant-Input)**.** *Für $\delta \in [0,1]$ ist das **Relevant-Input** Problem definiert durch:*
Sei $\Phi : \{0,1\}^d \to \{0,1\}, x \in \{0,1\}^d$ und $k \in [d]$, dann ist das Problem durch folgende Entscheidung definiert:

$$\textit{Existiert eine Menge } S \subseteq [d] \textit{ mit } |S| \leq k, \textit{ sodass } S \ \delta-\textit{relevant für } \Phi \textit{ und } x \textit{ ist?}$$

Diese Definition beschreibt das Relevant-Input Problem wie folgt: Es gibt einen Klassifikator Φ, welcher einem Booleschen Schaltkreis entspricht, ein Eingangssignal x und k als Element der Indexmenge. Dann ist das Problem die Frage zu beantworten, ob eine Menge S als Teilmenge der Indexmenge existiert, deren Mächtigkeit kleiner ist als k und S δ-relevant ist für Φ und x ist. Diese Frage kann nur positiv oder negativ beantwortet werden. Es bietet sich daher an, eine Version des Relevant-Input Problems zu definieren, welche eine Optimierung über einen Parameter erlaubt.

Definition 4.3 (Min-Relevant-Input)**.** *Für $\delta \in [0,1]$ ist das **Min-Relevant-Input** Problem definiert durch:*
Sei $\Phi : \{0,1\}^d \to \{0,1\}$ und $x \in \{0,1\}^d$.

MINIMIERE: $k \in [d]$ sodass eine Teilmenge $S \subseteq [d]$ existiert mit $|S| \leq k$, welche $\delta-$relevant für Φ und x ist.

Diese Definition beschreibt das Min-Relevant-Input Problem wie folgt: Es gibt einen Klassifikator Φ, welcher einem Booleschen Schaltkreis entspricht, ein Eingangssignal x und δ zwischen Null und Eins. Dann ist das Problem das k des Relevant-Input Problems zu minimieren. Also das kleinste k als Element der Indexmenge zu finden, sodass eine Teilmenge S der Indexmenge existiert deren Mächtigkeit kleiner als k ist und zudem δ-relevant ist für Φ und x ist, sodass es eine optimierbare Version des Relevant-Input Problems über den Parameter k darstellt. Es wurde gezeigt, dass folgendes Härteergebnis gilt [20, Thm. 3.1].

Satz 4.1. *Für $\frac{1}{2} \leq \delta < 1$ ist das Relevant-Input Problem NP^{PP}-vollständig.*

Wie oben schon erwähnt wird angenommen, dass NP^{PP} signifikant härter als NP und PP ist. Das bedeutet, dass das Lösen des Min-Relevant-Input Problems ein sehr schwieriges Problem darstellt. In der Praxis würde es oft ausreichen das Problem annähernd zu lösen und es besteht die Hoffnung, dass effiziente Annäherungsalgorithmen existieren. Daher wird der Satz 4.1. in dieser Arbeit nicht bewiesen. Das folgende Ergebnis ist weitaus praxisrelevanter.

Satz 4.2. *Angenommen $P \neq NP$. Dann gibt es für jedes $\alpha \in (0,1)$ keinen Approximations-Algorithmus mit polynomieller Komplexität für das Min-Relevant-Input Problem mit einem Annäherungsfaktor von $d^{1-\alpha}$.*

Der Beweis dieses Satzes erfolgt in Abschnitt 4.2. im Anschluss an den Beweis des Lemma 4.2.

4.2 Unannäherung (Inapproximability)

Ein Approximation-Algorithmus für das Min-Relevant-Input Problem hat den Annäherungsfaktor $c \geq 1$, falls der Algorithmus eine approximierende Lösung k findet, sodass $k^* \leq k \leq ck^*$ für alle Probleminstanzen, wobei k^* die jeweils exakte Lösung bezeichnet. Wählt man die triviale Lösung $S = [d]$ und

bezeichnet somit alle Komponenten als relevant, resultiert dies in einem Faktor d. Satz 4.2 sagt, dass es generell schwierig ist bessere Faktoren zu finden.

Der Beweis erfolgt in zwei Schritten. Als erstes wird eine Version des Relevant-Input Entscheidungsproblems mit einer Lücke eingeführt, wo eine Entscheidung getroffen werden kann, aber nicht notwendigerweise getroffen werden muss. Dies wird als Hilfsproblem bezeichnet. Es wird gezeigt, dass diese Version des Entscheidungsproblems $NP-$schwer ist. Als zweites wird gezeigt, dass das Hilfsproblem in P wäre, falls ein guter Approximations-Algorithmus mit polyninomieller Komplexität existiert, der das Min-Relevant-Input Problem lösen kann.

Definition 4.4 (Hilfsproblem (HP))**.** *Für $\delta \in [0,1]$ wird das **Hilfsproblem (HP)** wie folgt definiert:*
Sei $\Phi : \{0,1\}^d \to \{0,1\}, x \in \{0,1\}^d$, und $k, m \in \mathbb{N}, 1 \leq k \leq m \leq d$.
Entscheide welche der beiden Optionen gilt (falls überhaupt eine gilt):

JA-Instanz: Es existiert $S \subset [d]$ mit $|S| \leq k$ und S ist δ-relevant für Φ und x.
NEIN-Instanz: Alle $S \subset [d]$ mit $|S| \leq m$ sind nicht δ-relevant für Φ und x.

Dieses Hilfsproblem stellt eine lückenhafte Darstellung des Relevant-Input Problems dar, wobei die Einschränkung auf den Fall $k = m$ genau das Relevant-Input Problem ist. Es wird jedoch auch der Fall $k < m$ ermöglicht, was eine Lücke in der Menge erlaubt. Es ist daher möglich, dass keine der zwei Optionen gilt. In diesem Fall ist jede Antwort akzeptabel. Es wird gezeigt, dass diese lückenhafte Version des Relevant-Input Problems $NP-$schwer ist, also dass die Entscheidung für Ja oder Nein bezüglich des Hilfsproblems in polynomieller Zeit überprüft werden kann. Dazu wird folgendes Lemma bewiesen.

Lemma 4.2. *Für $\delta \in (0,1)$ gilt $SAT \preceq_p HP$ insbesondere gilt HP ist $NP-$schwer.*

Bevor das Lemma 4.2. bewiesen wird, wird kurz das SAT-Problem erläutert. In der theoretischen Informatik bezeichnet das "Boolean satisfiability problem" - abgekürzt "SAT" oder "B-SAT" - das Problem zu bestimmen, ob die Variablen einer bestimmten booleschen Formel konsistent durch die Werte Wahr oder Falsch (ggf. 0 oder 1) ersetzt werden können, so dass die Formel zu Wahr (1) ausgewertet wird. Wenn dies der Fall ist, wird die Formel als erfüllbar ("satisfiable") bezeichnet. Falls jedoch keine solche Zuordnung existiert und die Funktion, welche durch die Formel beschrieben wird, Falsch (0) ist für alle möglichen Zuordnungen von Wahr oder Falsch der Variablen der bestimmten booleschen Formel, so wird die Formel als nicht-erfüllbar ("unsatisfiable") bezeichnet. Zum Beispiel ist die Formel: a und nicht b erfüllbar, da man die Zuordnung $a =$ Wahr und $b =$ Falsch wählen kann. Hierrunter ist die Formel (a und nicht b) dann Wahr. Als Gegenbeispiel betrachten wir: a und nicht a. Diese Formel ist nicht erfüllbar, da alle Zuordnungen von Wahr und Falsch zu a immer zu einem Falsch der Formel (a und nicht a) führen.

Das SAT war das erste Problem, für welches gezeigt werden konnte, dass es $NP-$vollständig ist (Cook-Levin Theorem). Dies impliziert für die Komplexitätsklasse NP, dass die in ihr enthaltenden Entscheidungsprobleme höchstens so schwierig zu lösen sind wie das SAT. Die Probleme in der Klasse NP enthalten viele natürliche Entscheidungsprobleme und Optimierungsprobleme. Bisher gibt es keinen bekannten Algorithmus, der effizient alle SAT Probleme lösen kann und es wird generell angenommen, dass ein solcher Algorithmus nicht existiert. Diese Annahme konnte jedoch bisher nicht beweisen werden, da sie auf die Frage führt, ob das SAT lösbar ist durch einen Algorithmus mit polynomieller Komplexität. Dies wiederum ist äquivalent zu dem $P - NP-$Problem.

D.h. um das Lemma zu beweisen muss gezeigt werden, dass das SAT polynomiell reduzierbar auf das Hilfsproblem (HP) ist. Hieraus würde dann auch direkt folgen, dass das Hilfsproblem (HP) $NP-$schwer

ist, da das SAT $NP-$schwer ist.

Beweis. Sei $\Phi : \{0,1\}^d \to \{0,1\}$ eine SAT-Instanz. Dies kann angenommen werden, da man dem Eingangssignal aus $\{0,1\}^d$ jeweils ein Wahr (1) oder Falsch (0) zuordnen könnte, dies anschließend auswerten könnte und entweder Wahr (1) oder Falsch (0) erhalten würde.
Es wird nun $\{\Phi', x', k', m'\}$ konstruiert, sodass es eine JA-Instanz des HP ist, falls Φ eine JA-Instanz des SAT ist. Hierzu sei $q = \lceil \log_2(d) - \log_2(1-\delta) \rceil$ und $p = \lfloor -\log_2(\delta) \rfloor + 1$. Es sei $k' = dq, m' \geq k'$ willkürlich, aber maximal polynomiell in d. Zudem sei $\Phi' : \{0,1\}^{d \times q} \times \{0,1\}^{m'+p} \to \{0,1\}$ gegeben durch

$$\Phi'(u^{(1)}, ..., u^{(q)}, v) = \Phi(\bigwedge_{j=1}^{q} u^{(j)}) \vee (\bigwedge_{i=1}^{m'+p} v_i),$$

wobei jedes $u^{(j)} \in \{0,1\}^d$ und die Operationen innerhalb von Φ komponentenweise erfolgen. Dies ist eine Konstruktion mit polynomieller Komplexität.

Sei $x' = 1_{dq+m'+p}$ dann wird durch die Konstruktion von Φ' und x' garantiert, dass $\Phi'(x') = 1$ unabhängig von der Erfüllbarkeit von Φ.

> Es gilt, dass $\Phi'(x') = 1$ unabhängig von der Erfüllbarkeit von Φ, da aus der Definition von $x' = 1_{dq+m'+p}$ folgt, dass $u^{(j)} = 1 \quad \forall j = 1, ..., q$ und $v_i = 1 \quad \forall i = 1, ..., m'+p$ in x'.
> Damit gilt nun:
>
> $$\Phi'(x') = \Phi(\bigwedge_{j=1}^{q} \underbrace{u^{(j)}}_{=1}) \vee \underbrace{(\bigwedge_{i=1}^{m'+p} \underbrace{v_i}_{=1})}_{=1} = 1$$

Im Folgenden gelte die Schreibweise $U = (u^{(1)}, ..., u^{(q)})$.

Notwendigkeit: Es ist notwendig, um zu zeigen, dass das SAT-Problem polynomiell reduzierbar auf das Hilfsproblem ist, dass falls Φ eine JA-Instanz des SAT ist auch die Konstruktion $\{\Phi', x', k', m'\}$ eine JA-Instanz des Hilfsproblems ist.
Sei hierzu Φ eine JA-Instanz des SAT. Dann existiert $x \in \{0,1\}^d$ mit $\Phi(x) = 1$ (nach Definition des SAT). Sei $S = \{i \in [d] : x_i = 1\}$ und $S' = S \times [q]$. Es gelte die Bezeichnung: $A(u^{(1)}, ..., u^{(q)}) = \bigwedge_{(i,j) \in S'} u_i^{(j)}$.

Es gilt $|S'| \leq k'$, da $|S| \leq d$ und $|[q]| = q$ und hierraus folgt, dass $|S'| = |S \times [q]| \leq |S| \cdot |[q]| = dq = k'$ (nach der Definition von k').

Zudem ist S' δ-relevant für Φ' und x' falls $\mathbb{P}_{(U,v)}(\Phi'(U,v)|A(U)) \geq \delta$.

> Dies folgt aus der Definition der δ-Relevanz, denn S' ist δ-relevant für Φ' und x' falls
>
> $$\mathbb{P}_{(U,v)}(\Phi'(U,v) = \underbrace{\Phi'(x')}_{=1} | (U,v)_{S'} = \underbrace{x'_{S'}}_{\text{Die Einträge sind alle 1}}) \geq \delta.$$
>
> Durch Einsetzen von $\Phi'(x') = 1$ und $x' = 1_{dq+m'+p}$ ergibt sich
>
> $$\Leftrightarrow \mathbb{P}_{(U,v)}(\Phi'(U,v) = 1 | \forall (i,j) \in S' \text{ ist } u_i^{(j)} = 1) \geq \delta$$
>
> $$\Leftrightarrow \mathbb{P}_{(U,v)}(\Phi'(U,v) | \underbrace{\bigwedge_{(i,j) \in S'} u_i^{(j)}}_{=A(U)}) \geq \delta$$

$$\Leftrightarrow \mathbb{P}_{(U,v)}(\Phi'(U,v)|A(U)) \geq \delta$$

Es gilt:

$$\mathbb{P}_{(U,v)}(\Phi'(U,v)|A(U)) \geq \mathbb{P}_U(\bigwedge_{j=1}^{q} u^{(j)} = x|A(U)),$$

da

$$\mathbb{P}_{(U,v)}(\Phi'(U,v)|A(U)) = \mathbb{P}_{(U,v)}(\Phi(\bigwedge_{j=1}^{q} u^{(j)}) \vee (\bigwedge_{i=1}^{m'+p} v_i)|A(U))$$

durch einsetzen der Definition von $\Phi'(U,v)$.
Die Oder-Verknüpfung $\vee(\bigwedge_{i=1}^{m'+p} v_i)$ erlaubt weitere Möglichkeiten, dass das Ereignis Eins ist.
Es kann nach unten abgeschätzt werden durch weglassen des Terms:

$$\geq \mathbb{P}_U(\Phi(\bigwedge_{j=1}^{q} u^{(j)})|A(U)) \overset{\Phi(x)=1}{=} \mathbb{P}_U(\Phi(\bigwedge_{j=1}^{q} u^{(j)}) = \mathbb{P}_U(\Phi(x)|A(U))$$

Es gilt, dass $\Phi(x) = 1$, da Φ eine JA-Instanz des SAT, somit existiert $x \in \{0,1\}^d$ mit $\Phi(x) = 1$,
sodass das Ereignis gleich Eins sein kann oder gleich $\Phi(x)$.
Des weiteren kann der Term weiter nach unten abgeschätzt werden, indem nur der Fall der Gleichheit zwischen $\bigwedge_{j=1}^{q} u^{(j)} = x$ betrachtet wird.
Hierbei spielen alle anderen Möglichkeiten, in denen $\Phi(\bigwedge_{j=1}^{q} u^{(j)})$ ebenfalls Eins wäre keine Rolle.

$$\geq \mathbb{P}_U(\bigwedge_{j=1}^{q} u^{(j)} = x|A(U))$$

Dies kann weiter umgeformt werden.

$$\mathbb{P}_U(\bigwedge_{j=1}^{q} u^{(j)} = x|A(U)) \overset{A(U) \text{ einsetzen}}{=} \mathbb{P}_U(\bigwedge_{j=1}^{q} u^{(j)} = x| \bigwedge_{(i,j)\in S'} u_i^{(j)})$$

Weiter wird das Gegenteil betrachtet:

$$= 1 - \mathbb{P}_U((\bigwedge_{j=1}^{q} u^{(j)} = x)^C| \bigwedge_{(i,j)\in S'} u_i^{(j)})$$

$$= 1 - \mathbb{P}_U((\bigwedge_{j=1}^{q} u^{(j)} = x)^C|\forall i \in S \text{ ist } \bigwedge_{j=1}^{q} u_i^{(j)} = 1)$$

Mit Hilfe der Definition von $S = \{i \in [d] : x_i = 1\}$ wird der Term $\bigwedge_{j=1}^{q} u^{(j)} = x$ aufgespalten.
Alle Einträge der Komponenten in S müssen Eins sein und alle im Komplement von S Null.

$$= 1 - \mathbb{P}_{(U)}(((\forall i \in S : \bigwedge_{j=1}^{q} u_i^{(j)} = 1) \wedge (\forall i \in S^C : \bigwedge_{j=1}^{q} u_i^{(j)} = 0))^C|\forall i \in S \text{ ist } \bigwedge_{j=1}^{q} u_i^{(j)} = 1)$$

Die Angabe, dass $\forall i \in S : \bigwedge_{j=1}^{q} u_i^{(j)} = 1$ ist unter der selbigen Bedingung redundant,
sodass sie weggelassen werden kann.

$$= 1 - \mathbb{P}_{(U)}((\forall i \in S^C : \bigwedge_{j=1}^{q} u_i^{(j)} = 0)^C|\forall i \in S \text{ ist } \bigwedge_{j=1}^{q} u_i^{(j)} = 1)$$

Für die Bedingung $\forall i \in S$ ist $\bigwedge_{j=1}^{q} u_i^{(j)} = 1$ fällt auf, dass sie irrelevant
für das Ereignis ist, da hier nur Komponenten aus S^C betrachtet werden.
Dementsprechend ist die Bedingung von dem Ereignis unabhängig.

Somit gilt nach der bedingten Wahrscheinlichkeit für unabhängige Ereignisse:

$$= 1 - \frac{\mathbb{P}_{(U)}((\forall i \in S^C : \bigwedge_{j=1}^{q} u_i^{(j)} = 0)^C \cap (\forall i \in S \text{ ist } \bigwedge_{j=1}^{q} u_i^{(j)} = 1))}{\mathbb{P}_{(U)}(\forall i \in S \text{ ist } \bigwedge_{j=1}^{q} u_i^{(j)} = 1)}$$

$$= 1 - \frac{\mathbb{P}_{(U)}((\forall i \in S^C : \bigwedge_{j=1}^{q} u_i^{(j)} = 0)^C) \cdot \mathbb{P}_{(U)}(\forall i \in S \text{ ist } \bigwedge_{j=1}^{q} u_i^{(j)} = 1)}{\mathbb{P}_{(U)}(\forall i \in S \text{ ist } \bigwedge_{j=1}^{q} u_i^{(j)} = 1)}$$

$$= 1 - \mathbb{P}_{(U)}(\exists i \in S^C : \bigwedge_{j=1}^{q} u_i^{(j)})$$

Dieser Term soll nun nach δ abgeschätzt werden, sodass zu zeigen ist:

$$1 - \mathbb{P}_{(U)}(\exists i \in S^C : \bigwedge_{j=1}^{q} u_i^{(j)}) \geq 1 - |S^C| 2^{-q} \geq \delta$$

Betrachtet wird zunächst nur die erste Abschätzung. Durch Umformen des Terms erhält man:

$$1 - \mathbb{P}_{(U)}(\exists i \in S^C : \bigwedge_{j=1}^{q} u_i^{(j)}) \geq 1 - |S^C| 2^{-q}$$

$$\Leftrightarrow -\mathbb{P}_{(U)}(\exists i \in S^C : \bigwedge_{j=1}^{q} u_i^{(j)}) \geq -|S^C| 2^{-q}$$

$$\Leftrightarrow \mathbb{P}_{(U)}(\exists i \in S^C : \bigwedge_{j=1}^{q} u_i^{(j)}) \leq |S^C| 2^{-q}$$

Diese Abschätzung folgt auch der Booleschen Ungleichung (Bool's inequality)

$$\mathbb{P}_{(U)}(\exists i \in S^C : \bigwedge_{j=1}^{q} u_i^{(j)}) = \mathbb{P}_{(U)}(\bigcup_{i \in S^C} : \bigwedge_{j=1}^{q} u_i^{(j)})$$

$$\overset{\text{Bool's inequality}}{\leq} \sum_{i \in S^C} \mathbb{P}_U(\bigwedge_{j=1}^{q} u_i^{(j)}) = \sum_{i \in S^C} 2^{-q}$$

Da eine Und-Verknüpfung der $u_i^{(j)}$ für die Komponenten $i \in S^C$ über alle $j = 1, ..., q$ gefordert ist und diese jeweils nur Null oder Eins, mit Wahrscheinlichkeit $\frac{1}{2}$, annehmen können,
ist die Wahrscheinlichkeit hierfür 2^{-q}

$$= |S^C| 2^{-q}$$

Damit gilt die erste Abschätzung. Nun wird die zweite Abschätzung betrachtet. Durch Umformen erhält man:

$$1 - |S^C| 2^{-q} \geq \delta$$

$$\Leftrightarrow |S^C| 2^{-q} \leq 1 - \delta$$

$$\Leftrightarrow 2^{-q} \leq \frac{1 - \delta}{|S^C|}$$

$$\Leftrightarrow 2^q \geq \frac{|S^C|}{1 - \delta}$$

Diese Abschätzung ergibt sich aus der Definition von $q = \lceil \log_2(d) - \log_2(1-\delta) \rceil$

$$2^q = 2^{\lceil \log_2(d) - \log_2(1-\delta) \rceil} \geq 2^{\log_2(d) - \log_2(1-\delta)}$$

$$= 2^{\log_2(d)} \cdot 2^{-\log_2(1-\delta)} = \frac{d}{1-\delta}$$

Da $S^C \subseteq [d]$ ist dementsprechend $d \geq |S^C|$

$$\overset{S^C \in [d]}{\geq} \frac{|S^C|}{1-\delta}$$

Damit wurde gezeigt, dass die zweite Abschätzung gilt.
Zusammen zeigt dies, dass $\{\Phi', x', k', m'\}$ eine JA-Instanz des HP ist.

Dies wird hier kurz zusammengefasst. Es sollte gezeigt werden, dass falls Φ eine JA-Instanz des SAT ist auch die Konstruktion $\{\Phi', x', k', m'\}$ eine JA-Instanz des HP ist. Damit $\{\Phi', x', k', m'\}$ eine JA-Instanz des HP ist muss für ein $\delta \in [0,1]$ gelten, dass Es existiert $S' \subset [d] \times [q]$ mit $|S'| \leq k'$ und S' ist δ-relevant für Φ' und x'. Es gilt dass $|S'| \leq k'$ nach der Definition von k'. Es verbleibt die δ-Relevanz zu zeigen.
S' ist δ-relevant für Φ' und x' falls $\mathbb{P}_{(U,v)}(\Phi'(U,v)|A(U)) \geq \delta$.
Es wurde die Abschätzung gezeigt:

$$\mathbb{P}_{(U,v)}(\Phi'(U,v)|A(U)) \geq \mathbb{P}_U(\bigwedge_{j=1}^{q} u^{(j)} = x|A(U))$$

die konnte umgeformt werden zu:

$$= 1 - \mathbb{P}_U(\exists i \in S^C : \bigwedge_{j=1}^{q} u_i^{(j)})$$

Dieser Term wurde weiter nach unten abgeschätzt durch:

$$1 - \mathbb{P}_U(\exists i \in S^C : \bigwedge_{j=1}^{q} u_i^{(j)}) \geq 1 - |S^C| 2^{-q} \geq \delta$$

Sodass die δ-Relevanz gilt, falls Φ eine JA-Instanz des SAT ist. Damit ist $\{\Phi', x', k', m'\}$ eine JA-Instanz des HP, was zu zeigen war.

Suffizienz: Umgekehrt ist notwendig, dass falls Φ eine NEIN-Instanz des SAT ist auch die Konstruktion $\{\Phi', x', k', m'\}$ eine NEIN-Instanz des Hilfsproblems ist. Deshalb sei nun umgekehrt Φ eine NEIN-Instanz für das SAT. Dann existiert kein $x \in \{0,1\}^d$, sodass $\Phi(x) = 1$.
Für jedes $S' \subseteq [dq + m' + p]$ mit $|S'| \leq m'$ gilt, wegen $\Phi'(x') = 1$:

$$\mathbb{P}_y(\Phi'(y) = \Phi'(x')|y_{S'} = x_{S'})$$

$$= \mathbb{P}_{(U,v)}(\underbrace{\Phi(\bigwedge_{j=1}^{q} u^{(j)})}_{=0} \vee (\bigwedge_{i=1}^{m'+p} v_i)|\forall (i,j) \in S' \text{ gilt } u_i^{(j)} = 1 \text{ und } \forall i \in S' \text{ gilt } v_i = 1).$$

$\Phi(\bigwedge_{j=1}^{q} u^{(j)})$ ist gleich Null, da kein $x \in \{0,1\}^d$ existiert, sodass $\Phi(x) = 1$,
da Φ eine NEIN-Instanz des SAT ist.

Es wird jeweils nur die Einschränkung von (U, v) auf S' betrachtet.

$$= \mathbb{P}_{(U,v)}(\bigwedge_{i=1}^{m'+p} v_i | (U, v)_{S'} = 1)$$

In v können maximal $|S'|$ Einträge bedingt Eins sein. Hieraus folgt

$$\mathbb{P}_{(U,v)}(\bigwedge_{i=1}^{m'+p} v_i | (U, v)_{S'} = 1) \leq 2^{-(m'+p-|S'|)}.$$

Es verbleibt zu zeigen, dass

$$2^{-(m'+p-|S'|)} \leq \delta.$$

Durch umformen ergibt sich

$$\Leftrightarrow 2^{(m'+p-|S'|)} \geq 1/\delta$$

Wegen $|S'| \leq m'$ und mit $p = \lfloor -\log_2(\delta) \rfloor + 1$ ergibt sich:

$$2^{(m'+p-|S'|)} \overset{|S'| \leq m'}{\geq} 2^{(m'+p-m')} = 2^p$$

$$= 2^{\lfloor -\log_2(\delta) \rfloor + 1} \geq 2^{-\log_2(\delta)}$$

$$= \frac{1}{\delta}$$

Zusammenfassend bedeutet dies, dass

$$\mathbb{P}_y(\Phi'(y) = \Phi'(x') | y_{S'} = x_{S'}) = \mathbb{P}_{(U,v)}(\bigwedge_{i=1}^{m'+p} v_i | (U, v)_{S'} = 1) < \delta$$

Daraus folgt, dass S' nicht δ-relevant für Φ' und x' ist, sodass $\{\Phi', x', k', m'\}$ eine NEIN-Instanz des HP ist.

□

Nun kann die Unannäherung des Min-Relevant-Input Problems bewiesen werden und somit Satz 4.2.

Beweis. Ziel ist es zu beweisen, dass die Existenz eines Approximations-Algorithmus mit polynomieller Komplexität für das Min-Relevant-Input Problem mit einem Approximationsfaktor $d^{1-\alpha}$ es erlauben würde, das Hilfsproblem in polynomieller Zeit für bestimmte Instanzen zu entscheiden. Diese können wie im Beweis des Lemmas 4.2. gewählt werden. Das bedeutet im Umkehrschluss, dass das SAT in polynomieller Zeit entschieden werden kann. Dies ist nur möglich, falls $P = NP$.

Sei $\Phi : \{0,1\}^d \to \{0,1\}$ eine SAT Instanz und $\{\Phi', x', k', m'\}$ eine äquivalente Hilfsproblem Instanz wie im Beweis des Lemmas 4.2. Es gibt, wie beobachtet, eine gewisse Freiheit in der Wahl von m' solange gilt, dass $k' \leq m'$ und maximal polynomiell in d ist. Wir wählen $m' = \lceil max(2k'(k'^{1-\alpha} + p^{1-\alpha}), (2k')^{\frac{1}{\alpha}} + 1) \rceil$ mit $p = \lfloor -\log_2(\delta) \rfloor + 1$ wie vorher. Zudem sei wie vorhin $k' = dq$ mit $q = \lceil \log_2(d) - \log_2(1 - \delta) \rceil$.
Somit ist m' sicher polynomiell in d und $k' \leq m'$. Es gilt $k' \leq m'$, da

$$m' \geq (2k')^{\frac{1}{\alpha}} + 1 \overset{\alpha \in (0,1)}{\geq} 2k' + 1 \geq k'.$$

Des Weiteren gilt $m' > (2k')^{\frac{1}{\alpha}}$ (aus der Wahl von m'). Daraus folgt, dass $1 - k'm'^{-\alpha} > \frac{1}{2}$, da

$$m' > (2k')^{\frac{1}{\alpha}}$$

$$\Leftrightarrow m'^{\alpha} > 2k'$$

$$\Leftrightarrow m'^{-\alpha} < \frac{1}{2k'}$$

$$\Leftrightarrow k'm'^{-\alpha} < \frac{1}{2}$$

$$\Leftrightarrow -k'm'^{-\alpha} > -\frac{1}{2}$$

$$\Leftrightarrow 1 - k'm'^{-\alpha} > \frac{1}{2}.$$

Durch Multiplikation mit m' und aus der Wahl von m' erhält man nun:

$$m'(1 - k'm'^{-\alpha}) > \frac{m'}{2} \geq k'(k'^{1-\alpha} + p^{1-\alpha}).$$

Nun sei mit $d' = k' + m' + p$ die Anzahl der Variablen von Φ' bezeichnet. Durch die Subadditivität der Abbildung $z \mapsto z^{1-\alpha}$ ergibt sich

$$k'd'^{1-\alpha} = k'(k' + m' + p)^{1-\alpha} \leq k'(k'^{1-\alpha} + m'^{1-\alpha} + p^{1-\alpha})$$

$$= \underbrace{k'(k'^{1-\alpha} + p^{1-\alpha})}_{\leq \frac{m'}{2}} + \underbrace{k'm'^{1-\alpha}}_{\leq \frac{m'}{2}} \leq m'$$

Es gilt $k'm'^{1-\alpha} \leq \frac{m'}{2}$, da

$$m'(1 - k'm'^{-\alpha}) > \frac{m'}{2}$$

$$\Leftrightarrow m' - k'm'^{1-\alpha} > \frac{m'}{2}$$

$$\Leftrightarrow -k'm'^{1-\alpha} > -\frac{m'}{2}$$

$$\Leftrightarrow k'm'^{1-\alpha} < \frac{m'}{2}.$$

Es verbleibt zu zeigen, dass eine Hilfsproblem Instanz mit $m' > k'd^{1-\alpha}$ durch einen Approximations-Algorithmus für das Min-Relevant-Input Problem mit einem Approximationsfaktor von $d'^{1-\alpha}$ entschieden werden kann. Angenommen ein solcher Algorithmus existiert und sei k die Lösung dieses Algorithmus, dann gilt für die korrekte optimale Lösung k^*, dass $k^* \leq k \leq d'^{1-\alpha}k^*$.

Als erstes sei angenommen, dass $\{\Phi', x', k', m'\}$ eine JA-Instanz des HP ist. Dann existiert eine δ-relevante Menge der Größe k'. Bemerkt sei, dass keine kleinere Menge als k^* δ-relevant sein kann. Dies impliziert, dass $k^* \leq k'$ und damit $k \leq d'^{1-\alpha}k' < m'$.

Zweitens sei nun angenommen, dass $\{\Phi', x', k', m'\}$ eine NEIN-Instanz des HP ist. Dann sind alle Mengen mit einer maximalen Größe von m' nicht δ-relevant. Aber es existiert eine δ-relevante Menge der Größe k^*. Dies impliziert, dass $k \geq k^* > m'$.

Zusammenfassend bedeutet dies, dass das Überprüfen, ob $k < m'$ oder $k > m'$ das Problem $\{\Phi', x', k', m'\}$

entscheidet.

Somit konnte gezeigt werden, dass falls ein Algorithmus existiert, welcher das Min-Relevant-Input Problem in polynomieller Zeit lösen kann mit einem Approximationsfaktor $d^{1-\alpha}$, das Hilfsproblem in polynomieller Zeit entschieden werden kann.

□

Satz 4.2. besagt, dass kein effizienter Approximations-Algorithmus für das Min-Relevant-Input Problem existiert, es sei denn $P = NP$ wie im Beweis gezeigt wurde. Entweder muss auf Heuristiken zurückgegriffen oder das Problem stärker eingeschränkt werden. Im Folgenden wird sich auf Ersteres konzentriert werden und eine allgemeine Heuristik für neuronale Netze präsentieren werden. Des Weiteren wird ebenfalls die Problemformulierung weiter aufgelockert zu einer kontinuierlichen Form.

5 Problemlockerung und heuristische Lösung

Die Problemklasse NP^{PP} lässt bereits erahnen, welche Schwierigkeiten es zu überwinden gilt. Als erstes wird ein effizienter Weg benötigt, um zu beurteilen, ob eine gewählte Menge zu einer kleinen erwarteten Distortion führt. Dies impliziert die Berechnung von Erwartungswerten. Als zweites muss über alle zulässigen Mengen optimiert werden, was sich als kombinatorisches Optimierungsproblem darstellt. Hier soll, wie durch Macdonald et al. [6] vorgeschlagen, eine heuristische Lösung für beide Probleme, wenn der Klassifikator Φ ein "deep" neuronales Netzwerk ist, betrachtet werden.

5.1 Neuronale Netzwerkfunktionen

Sei $L \in \mathbb{N}$ die Anzahl von Layern / Schichten eines neuronalen Netzwerkes, $d_1, ..., d_{L-1} \in \mathbb{N}$ und $d_0 = d, d_L = 1$. Weiter seien $(W_1, b_1), ..., (W_L, b_L)$ mit $W_i \in \mathbb{R}^{d_i \times d_{i-1}}, b_i \in \mathbb{R}^{d_i}$ für $i \in [L]$ die Gewichte der Matrizen und der Bias Vektoren eines L-Layer feed forward neuronalen Netzwerkes, sodass dann Funktionen folgender Form betrachtet werden:

$$\Phi(x) = W_L \varrho(W_{L-1} \varrho(...\varrho(W_1 x + b_1)...) + b_{L-1}) + b_L$$

Im Folgenden sei die Aktivierungsfunktion $\varrho(x) = max\{0, x\}$, die gleichgerichtete linearen Einheit (rectified linear unit ReLU).

5.2 Kontinuierliche Problemlockerung

Um das kombinatorische Optimierungsproblem anzugehen, wird folgende Relaxierung benutzen werden. Anstelle von binären Relevanzentscheidungen (relevant versus nicht relevant), welche durch die Menge S kodiert werden, wird eine kontinuierliche Relevanzbewertung für jede Komponente erlaubt, kodiert durch einen Vektor $s \in [0, 1]^d$. Es wird erneut eine Trübung von x bezüglich s als komponentenweise konvexe Kombination definiert.

$$y = x \odot s + n \odot (1 - s)$$

von x und $n \sim \mathcal{V}$. Wie zuvor sei $\mathcal{V}_s$ die resultierende Verteilung von y. Dies ist eine Verallgemeinerung von der in Abschnitt 2.1 eingeführte Trübung. Diese entspricht exakt der Trübung aus Abschnitt 2.1 falls s gleich Eins auf S und Null auf S^C gewählt wird. Die natürliche Lockerung der Größe der Menge $|S|$ ist die Norm $\| s \|_1 = \sum_{i=1}^{d} |s_i|$. Wie zuvor wird die erwartete Distortion definiert und in ihre Bias-Varianz-Zerlegung umgeschrieben. Zudem wird mit dem quadrieten Erwartunswert (gelb) erweitert:

$$D(s) = \mathbb{E}_{y \sim \mathcal{V}_s} \left[\frac{1}{2} (\Phi(x) - \Phi(y))^2 \right]$$

$$= \mathbb{E}_{y \sim \mathcal{V}_s} \left[\frac{1}{2} (\Phi(x)^2 + \Phi(y)^2 - 2\Phi(x)\Phi(y)) \right] = \frac{1}{2}\Phi(x)^2 + \frac{1}{2}\mathbb{E}_{y \sim \mathcal{V}_s} \left[\Phi(y)^2 \right] - \Phi(x)\mathbb{E}_{y \sim \mathcal{V}_s} \left[\Phi(y) \right]$$

$$= \underbrace{\frac{1}{2}\Phi(x)^2 - \Phi(x)\mathbb{E}_{y \sim \mathcal{V}_s} \left[\Phi(y) \right] + \frac{1}{2}\mathbb{E}_{y \sim \mathcal{V}_s} \left[\Phi(y) \right]^2}_{\text{2 Binomische Formel}} + \frac{1}{2}\mathbb{E}_{y \sim \mathcal{V}_s} \left[\Phi(y)^2 \right] + \frac{1}{2}\mathbb{E}_{y \sim \mathcal{V}_s} \left[\Phi(y) \right]^2 - \mathbb{E}_{y \sim \mathcal{V}_s} \left[\Phi(y) \right]^2$$

$$= \frac{1}{2}(\Phi(x) - \mathbb{E}_{y \sim \mathcal{V}_s} \left[\Phi(y) \right])^2 + \frac{1}{2}\mathbb{E}_{y \sim \mathcal{V}_s} \left[\Phi(y)^2 \right] + \frac{1}{2}\mathbb{E}_{y \sim \mathcal{V}_s} \left[\Phi(y) \right]^2 - \mathbb{E}_{y \sim \mathcal{V}_s} \left[\Phi(y) \right]^2$$

$$= \frac{1}{2}(\Phi(x) - \mathbb{E}_{y \sim \mathcal{V}_s} \left[\Phi(y) \right])^2 + \frac{1}{2}\mathbb{E}_{y \sim \mathcal{V}_s} \left[\underbrace{\Phi(y)^2 + \mathbb{E}_{y \sim \mathcal{V}_s} \left[\Phi(y) \right]^2 - 2\mathbb{E}_{y \sim \mathcal{V}_s} \left[\Phi(y) \right] \Phi(y)}_{\text{2 Binomische Formel}} \right]$$

$$= \frac{1}{2}(\Phi(x) - \mathbb{E}_{y\sim\mathcal{V}_s}[\Phi(y)])^2 + \frac{1}{2}\mathbb{E}_{y\sim\mathcal{V}_s}\left[(\Phi(y) - \mathbb{E}_{y\sim\mathcal{V}_s}[\Phi(y)])^2\right]$$

$$= \frac{1}{2}(\Phi(x) - \mathbb{E}_{y\sim\mathcal{V}_s}[\Phi(y)])^2 + \frac{1}{2}\mathbb{V}_{y\sim\mathcal{V}_s}[\Phi(y)],$$

wobei $\mathbb{V}$ die Covarianzmatrix bezeichnet. Die erwartete Distortion wird durch das erste und zweite Moment der Ausgabeschicht bestimmt. Die exakte Berechnung von Erwartungswerten und Varianzen für beliebige Funktionen ist an sich schon ein schwieriges Problem. Eine Möglichkeit, dieses Problem zu lösen, besteht darin, die Erwartung durch einen Stichprobenmittelwert zu approximieren. Abhängig von der Dimension d und der Verteilung $\mathcal{V}$ kann eine Probenahme unmöglich sein. Eine andere Möglichkeit ist den spezifischen Aufbau von Φ stärker zu berücksichtigen. Aus der Definition der Trübung von x bezüglich s als komponentenweise konvexe Kombination ist es einfach, das erste und zweite Moment zu erhalten.

$$\mathbb{E}[y] = x \odot s + \mathbb{E}[n] \odot (1 - s) \text{ und } \mathbb{V}[y] = diag(1 - s)\mathbb{V}[n]diag(1 - s)$$

der Eingangsverteilung $\mathcal{V}_s$. Es bleibt, die Momente von der Eingangs- auf die Ausgangsschicht zu übertragen. Dies wird in Abschnitt 4.3 diskutiert. Anstelle der harten Nebenbedingung $\mathcal{D}(s) \leq \epsilon$ wie in Abschnitt 2 wird das kontinuierliche Ratenminimierungsproblem in seiner Lagrange-Formulierung formuliert.

$$\text{minimiere} \quad \mathcal{D}(s) + \lambda \parallel s \parallel_1 \qquad \text{für } s \in [0,1]^d$$

mit dem Regularisierungsparameter $\lambda > 0$. Dieser Ansatz wird genutzt, um Relevanzwerte für die Klassifikatorentscheidungen RDE (Rate-Distortion Explanation) zu erhalten. Je nach Aktivierungsfunktion muss die Distortion nicht differenzierbar sein. Die *ReLu*-Aktivierung ist jedoch fast überall differenzierbar. Wie beim Training neuronaler Netze üblich, wird ebenfalls (projected) gradient descent verwendet, um einen stationären Punkt des obigen Minimierungs-Problems zu finden.

5.3 Angenommene Dichtefilterung (Assumed density filtering)

Um die Herausforderung der effizienten Approximation der Erwartungswerte anzugehen, wird die Schichtstruktur von Φ genutzt und die Verteilung der Neuronenaktivierungen durch das Netzwerk propagiert. Dazu wird eine Näherungsmethode verwendet, die als angenommene Dichtefilterung (assumed density filtering, ADF) bezeichnet wird. ADF ist eine Methode, mit welcher die Vergangenheit in Bayesschen Netzwerken und anderen statistischen Modellen geschätzt werden kann. Diese Methode haben einigen Autoren unabhängig voneinander vorgeschlagen (Lauritzen 1992; Bernardo & Giron 1988; Stephens, 1997). Andere Namen für diese Methode sind „online Bayesian learning“, „moment matching“, „weak marginalization“. ADF lässt sich anwenden, falls eine gemeinsame Verteilung $p(D, \theta)$ vorliegt, wobei D beobachtet wurde und θ verborgen ist. Um etwas über die Vergangenheit von θ, $p(\theta|D)$ und die Wahrscheinlichkeit der beobachteten Daten $p(D)$ herauszufinden, kann ADF angewendet werden.
Diese Methode wurde kürzlich für neuronale *ReLu*-Netze im Zusammenhang mit der Unsicherheitsquantifizierung verwendet. Die Idee ist, für jede Schicht eine Gaußsche Verteilung für die Eingabe anzunehmen, sie gemäß den Schichtgewichten W, Bias b und der Aktivierungsfunktion ϱ zu transformieren und die Ausgabe zurück auf die nächste Gaußsche Verteilung (bezüglich KL-Divergenz) zu projizieren. Dies läuft auf die Anpassung der ersten beiden Momente der Verteilung hinaus. Im Folgenden werden die ADF-Regeln für eine einzelne Netzwerkschicht angegeben. Wenn diese wiederholt angewendet werden, können Momente durch alle Schichten propagiert und ein expliziter Ausdruck für die Distortion angegeben werden.

Sei $z \sim \mathcal{N}(\mu, \Sigma)$ normalverteilt mit dem Mittelwert μ und Kovarianz Σ. Unter einer affinen linearen

Transformation bleibt die Normalverteilung erhalten und wirkt auf den Mittelwert und die Kovarianz in bekannter Weise:

$$\mathbb{E}[Wz + b] = W\mu + b \quad \text{und} \quad \mathbb{V}[Wz + b] = W\Sigma W^*.$$

Die Nichtlinearität der *ReLu*-Aktivierungsfunktion stellt eine Schwierigkeit dar, da sie die Normalverteilung in eine nicht-normalverteilte Verteilung umwandelt. Seien f und F die Wahrscheinlichkeitsdichte und kumulative Verteilungsfunktion der univariaten Standardnormalverteilung. Sei weiter σ der Vektor der diagonalen Einträge von Σ und $\eta = \mu \oslash \sigma$. Dann gilt [21, Gl. 10a]:

$$\mathbb{E}[\varrho(z)] = \sigma \odot f(\eta) + \mu \odot F(\eta).$$

Unglücklicherweise besteht der Verdacht, dass die nichtdiagonalen Einträge der Kovarianzmatrix von $\varrho(z)$ keine geschlossene Lösung haben. Entweder wird die zusätzliche Annahme gemacht, dass die Netzwerkaktivierungen innerhalb jeder Schicht unkorreliert sind, was darauf hinausläuft, nur die Diagonale $\mathbb{V}_{diag}$ der Kovarianzmatrizen durch das Netzwerk zu propagieren. Dies vereinfacht die Varianz zu:

$$\mathbb{V}_{diag}[Wz + b] = (W \odot W)\sigma.$$

Dies resultiert in [21, Gl. 10b]

$$\mathbb{V}_{diag}[\varrho(z)] = \mu \odot \sigma \odot f(\eta) + (\sigma^2 + \mu^2) \odot F(\eta) - \mathbb{E}[\varrho(z)]^2.$$

Oder eine Approximation von der vollständigen Kovarianzmatrix wird benutzt

$$\mathbb{V}[\varrho(z)] \approx N\Sigma N,$$

mit $N = diag(F(\eta))$. Dies versichert eine positive Semi-Definitheit und Symmetrie. Abhängig von der Netzwerkgröße ist es normalerweise nicht möglich, die vollständige Kovarianzmatrix einer jeden Schicht zu berechnen. Wenn jedoch eine symmetrische Faktorisierung mit niedrigem Rang $\mathbb{V}[y] \approx QQ^T$ bei der Eingangsschicht gewählt wird mit $Q \in \mathbb{R}^{d \times r}$ für $r \ll d$ (z.B. die Hälfte einer Singulärwertzerlegung), dann ermöglicht die symmetrische Aktualisierung durch die Approximation der vollständigen Kovarianzmatrix, einen der Faktoren durch die Schichten zu propagieren. Es sei angemerkt, dass es nur möglich ist einen der Faktoren durch die Schichten zu propagieren. Die volle Kovarianz wird erst in der Ausgangsschicht wiederhergestellt. Dies reduziert den Rechenaufwand und den Speicherbedarf.

Durch Kombination der affinen linearen Transformation mit der nicht-linearen Transformation konnten die ersten beiden Momente (Erwartungswert und Varianz) durch ein neuronales *ReLu*-Netzwerk im ADF-Framework propagiert werden. In Abschnitt 7 werden numerische Beispiele zu mehreren Datensätzen und Methoden betrachtet wobei die diagonale Annäherung an die Kovarianzmatrix genutzt wird.

6 Andere Methoden

Sei im Folgenden $x = [x_1, ..., x_N] \in \mathbb{R}^N$ ein Eingangssignal für einen Klassifikator $\Phi(x) = [\Phi_1(x), ..., \Phi_C(x)] \in \mathbb{R}^C$ mit C Klassen, wie ein neuronales Netzwerk mit C Ausgabeneuronen. Je nach Anwendungsfall kann das Eingangssignal $x_1, ..., x_N$ unterschiedlich sein z.B. Pixel eines Bildes oder ein multidimensionaler Wort-Vektor im Rahmen von Sprachverarbeitung.
Nun soll für eine einzelne Ausgabe $c \in \{1, ..., C\}$ der Anteil an dem Beitrag / Relevanz $R^c(x, \Phi_c) = [R_1^c(x, \Phi_c), ..., R_N^c(x, \Phi_c)] \in \mathbb{R}^N$ zur Klassifikation $\Phi_c(x)$ für jeden Anteil des Eingangssignals x_i bestimmt werden. Im Falle einer Klassifikation ist das Neuron der Ausgabeschicht von Interesse, welches die korrekte Klasse angibt.
Werden alle Beiträge des Eingangssignals dargestellt, so werden alle $R^c(x, \Phi_c)$ zusammen als Relevanzkarte oder auch Heatmap bezeichnet, wobei rot einen positiven Beitrag und blau einen negativen Beitrag zur Klassifikation anzeigt.

6.1 Randomized Input Sampling for Explanation (RISE)

Die RISE-Methode weist Ähnlichkeiten zu der Rate-Distortion-Methode auf. Die zugrundeliegende Idee der RISE-Methode ist es, das Eingangssignal zu perturbieren / verschleiern und zu beobachten wie stark dies die Klassifikation des Klassifikators beeinflusst [22]. Für diese Perturbation des Eingangssignals gibt es unterschiedliche Methoden, z.B. können Anteile / Pixel des Eingangssignals auf Null gesetzt werden, die Region kann verpixelt / geblurrt werden oder Rauschen (Noise) kann hinzugefügt werden.
Sei $M \in \{0, 1\}^N$ eine zufällige binäre Maske mit Verteilung D. Es wird die Zufallsvariable $\Phi_c(x \odot M)$ betrachtet, wobei $\odot$ die Elementweise Multiplikation darstellt. Hierbei wird das Eingangssignal durch die Maske auf eine Subgruppe eingeschränkt und anschließend klassifiziert. Die Relevanz eines Anteils des Eingangssignals / Pixels $i \in \{1, ...N\}$ lässt sich definieren, als der Erwartungswert über alle möglichen Masken M unter der Bedingung, dass der Anteil des Eingangssignals / Pixel i eingeschlossen ist ($M_i = 1$):

$$R_i^c(x, \Phi_c) = \mathbb{E}_M[\Phi_c(x \odot M) | M_i = 1]$$

Intuitiv ist $\Phi_c(x \odot M)$ größer, wenn die erhaltene Subgruppe durch Maske M die relevanten Anteile / Pixel des Eingangssignals enthält. $R_i^c(x, \Phi_c)$ kann als Summe über Masken betrachtet werden:

$$R_i^c(x, \Phi_c) = \sum_m \Phi_c(x \odot m) \mathbb{P}[M = m | M_i = 1]$$

$$= \frac{1}{\mathbb{P}[M_i = 1]} \sum_m \Phi_c(x \odot m) \mathbb{P}[M = m, M_i = 1]$$

Es gilt:

$$\mathbb{P}[M = m, M_i = 1] = \begin{cases} 0, & \text{falls } m_i = 0 \\ \mathbb{P}[M = m], & \text{falls } m_i = 1 \end{cases}$$
$$= m_i \mathbb{P}[M = m]$$

Somit ergibt sich:

$$R_i^c(x, \Phi_c) = \frac{1}{\mathbb{P}[M_i = 1]} \sum_m \Phi_c(x \odot m) m_i \mathbb{P}[M = m]$$

Für das gesamte Eingangssignal ergibt sich dann in Kombination mit $\mathbb{P}[M_i = 1] = \mathbb{E}[M_i]$:

$$R^c(x, \Phi_c) = \frac{1}{\mathbb{E}[M]} \sum_m \Phi_c(x \odot m) \cdot m \cdot \mathbb{P}[M = m]$$

Somit können die Relevanzkarten als eine gewichtete Summe von zufälligen Masken adaptiert an die Verteilung der zufälligen Masken dargestellt werden, wobei die Gewichte die Wahrscheinlichkeitswerte sind, die sich unter den Masken ergeben.
Initial wurden dies empirisch mit der Monte-Carlo-Methode durchgeführt.

6.2 Vanilla Gradients

Ein als Vanilla-Gradients-Methode beschriebener Ansatz Relevanzkarten zu generieren ist, direkt die Gradienten von einem Ausgabeneuron zurück zu dem Eingangssignal / Pixel zu berechnen [23, Kap. 3]. Zu bemerken ist, dass die generierten Relevanzkarten mit dieser Methode nur zeigen, welche Anteile des Eingangssignals / Pixel relevant waren für die Klassifikation, jedoch ohne weiter zu spezifizieren, ob sie einen positiven oder negativen Effekt hatten.
Sei als motivierendes Beispiel der Klassifikator Φ_c für die Klasse c ein lineares Modell:

$$\Phi_c(x) = w_c^T x + b_c,$$

wobei w_c und b_c Gewichte und Bias des Modells sind. In diesem Fall definiert die Größe der Elemente von w die Relevanz für den dazugehörigen Teil des Eingangssignals $x_1, ..., x_N$ für die Klasse c. Im Falle von neuronalen Netzwerken oder den meisten anderen Klassifikatoren ist Φ_c eine nicht lineare Funktion von x. Trotzdem kann für ein gegebenes Eingangssignal x, der Klassifikator $\Phi_c(\tilde{x})$ geschätzt werden durch eine lineare Funktion in der Nähe von x durch Berechnung der Taylorentwicklung erster Ordnung:

$$\Phi_c(\tilde{x}) \approx w_c^T \tilde{x} + b_c,$$

wobei w_c die Ableitung von Φ_c unter dem Signal $\tilde{x}$ an dem Punkt x ist:

$$w_c = \frac{\partial \Phi_c}{\partial \tilde{x}}|_x$$

Hiermit ergibt sich dann die Relevanzwerte Farbkarte wie folgt:

$$R_i^c(x, \Phi_c) = \frac{\partial \Phi_c(x)}{\partial x_i},$$

wobei $i \in \{1, ..., N\}$ der Index des jeweiligen Anteils des Eingangssignals x_i ist.

6.3 Smooth Gradients

Smilkov et al. [24, Kap. 2.2] publizierten 2017 eine verbesserte Version der Vanilla Gradients-Methode. Laut den Autoren weisen die typischen Vanilla-Gradienten-Relevanzkarten relativ viel Rauschen (Noise) auf, welche am wahrscheinlichsten durch lokale Fluktuationen in den Ableitungen der Aktivierungsfunktionen der Ausgabeneuronen entstehen. Um diese Fluktuationen auszugleichen, haben die Autoren vorgeschlagen, die Gradienten für mehrere Versionen des Eingangssignals / Originalbildes zu berechnen, wobei diese durch Pertubation mit Rauschen von der Gaus-Verteilung entstehen. Seien w_c die nicht geglätteten Gradienten und $\widehat{w}_c$ die geglättet Gradienten:

$$\widehat{w}_c(x) = \frac{1}{n} \sum_1^n w_c(x + \mathcal{N}(0, \sigma^2)),$$

wobei n die Anzahl der Pertubationen ist und $\mathcal{N}(0, \sigma^2)$ Rauschen gemäß der Standardabweichung σ. Da es keine eindeutige Metrik gibt, mit der die Ausgabe der Vanilla-Gradienten und Smooth Gradients

verglichen werden könnten, argumentierten die Autoren, dass qualitativ betrachtet die resultierenden Smooth-Gradient Relevanzkarte weniger Rauschen aufweist.

6.4 Gradient×Input

Bei der Gradient×Input-Methode werden die partiellen Ableitungen der Ausgabe zu jedem Eingangssignal gebildet analog zu der Vanilla-Gradients-Methode und mit dem Eingangssignal multipliziert, sodass sich die Relevanzwerte wie folgt ergeben [25, Table 1],[26, Gl. 1]:

$$R_i^c(x, \Phi_c) = x_i \cdot \frac{\partial \Phi_c(x)}{\partial x_i}$$

Diese Methode wurde initial vorgestellt als Technik zur Erhöhung der Schärfe von Relevanzkarten.

6.5 Integrated Gradients

Analog zu der Gradient×Input-Methode werden bei der Integrated Gradients-Methode die partiellen Ableitungen der Ausgabe zu jedem Eingangssignal gebildet. Im Gegensatz zur Gradient×Input-Methode wird bei der Integrated Gradients-Methode nicht nur eine Ableitung an einem Punkt x ausgewertet, sondern der mittlere Gradient berechnet, wobei das Eingangssignal entlang eines linearen Pfades von einer Ausgangslage $\overline{x}$ bis zu x variiert. Die Ausgangslage kann frei gewählt werden und ist somit oft Null. Die Relevanzwerte ergeben sich nun wie folgt [25, Table 1],[27, Gl. 1]:

$$R_i^c(x, \Phi_c) = (x_i - \overline{x_i}) \cdot \int_{\alpha=0}^{1} \frac{\partial \Phi_{c,x}(\tilde{x})}{\partial(\tilde{x_i})}|_{\tilde{x}=\overline{x}+\alpha(x-\overline{x})} d\alpha.$$

Zu den hiermit bestimmten Relevanzwerten ist anzumerken, dass die Summe ebendieser das Ausgangssignal $\Phi_c(x)$ minus die Ausgabe des Klassifikators ausgewertet in der Ausgangslage $\Phi_c(\overline{x})$ ergibt:

$$\sum_{i=1}^{N} R_i^c(x, \Phi_c) = \Phi_c(x) - \Phi_c(\overline{x})$$

Diese Eigenschaft wird als *Vollständigkeit* bezeichnet [27, Kap. 3].

6.6 Deep Taylor

Analog soll bei der Deep-Taylor Methode zu jedem Anteil des Eingangsignals / Pixel $i \in \{1, ..., N\}$ ein Relevanzwert $R_i^c(x, \Phi_c)$ angegeben werden, welcher für ein Eingangssignal / -bild x angibt, inwiefern der Anteil des Eingangssignals / Pixel i zu der Klassifikation von $\Phi_c(x)$ beiträgt [28]. Die Relevanzwerte der einzelnen Anteile des Eingangssignals / Pixel können in einer Relevanzwerte Farbkarte $R^c(x, \Phi_c) = [R_1^c(x, \Phi_c), ..., R_N^c(x, \Phi_c)] \in \mathbb{R}^N$ mit der gleichen Dimension wie $x \in \mathbb{R}^N$ dargestellt werden.

Definition 6.1 (Konservative Relevanzwerte Farbkarte $R^c(x, \Phi_c)$)**.** *Für eine konservative Relevanzwerte Farbkarte $R^c(x, \Phi_c)$ entsprechen die Relevanzwerte der einzelnen Anteile des Eingangssignals / Pixel der Relevanz angegeben durch den Klassifikator [28, Def. 1]:*

$$\forall x : \Phi_c(x) = \sum_{i=1}^{N} R_i^c(x, \Phi_c)$$

Definition 6.2 (Positive Relevanzwerte Farbkarte)**.** *Eine Relevanzwerte Farbkarte $R^c(x, \Phi_c)$ heißt posi-*

tiv, falls alle Werte der Farbkarte größer gleich Null sind [28, Def. 2]:

$$\forall x, i : R_i^c(x, \Phi_c) \geq 0$$

Definition 6.3 (Konsistente Relevanzwerte Farbkarte)**.** *Eine Relevanzwerte Farbkarte $R^c(x, \Phi_c)$ heißt konsistent, falls konservativ und positiv ist [28, Def. 3].*

Lemma 6.1. *Für eine konsistente Relevanzwerte Farbkarte $R^c(x, \Phi_c)$ gilt:*

$$\Phi_c(x) = 0 \Rightarrow R^c(x, \Phi_c) = 0$$

Dieses Lemma wird an dieser Stelle nicht bewiesen. Die Deep-Taylor-Methode ist eine Zerlegungs-Methode [28, Gl. 4] basierend auf der Taylorentwicklung an einem gut gewählten *Entwicklungspunkt* $\tilde{x}$. Für einen Entwicklungspunkt gilt $\Phi_c(\tilde{x}) = 0$. Des Weiteren sei angenommen, dass $\Phi_c(x)$ eine beliebig oft differenzierbare Funktion ist. Die Taylorentwicklung erster Ordnung von $\Phi_c(x)$ ist dann gegeben durch:

$$\Phi_c(x) = \Phi_c(\tilde{x}) + \left(\frac{\partial \Phi_c}{\partial x}|_{x=\tilde{x}}\right)^T \cdot (x - \tilde{x}) + \epsilon = 0 + \sum_{i=1}^{N} \underbrace{\frac{\partial \Phi_c}{\partial x}|_{x=\tilde{x}} \cdot (x_i - \tilde{x}_i)}_{=R_i^c(x,\Phi_c)} + \epsilon$$

hierbei ist die Summe $\sum_i$ die Summe über alle Anteile / Pixel des Eingangssignals x und $\{\tilde{x}_i\}$ sind die Werte des Entwicklungspunktes $\tilde{x}$. Die Summanden sind hierbei die Relevanzwerte $R_i^c(x, \Phi_c)$ bezüglich eines jeden Anteils / Pixels des Eingangssignals. Im Term ϵ sind Terme zweiter und höherer Ordnung eingeschlossen. Problematisch bei Termen zweiter und höherer Ordnung ist, dass mehrere Anteile / Pixel des Eingangssignals involviert sind, sodass sich die Aufteilung der Relevanz verkompliziert. Somit kann die Relevanzwerte Farbkarte dargestellt werden als das Elementweise Produkt „$\odot$“ von dem Gradienten der Funktion $\frac{\partial \Phi_c}{\partial x}$ and dem Entwicklungspunkt $\tilde{x}$ und der Differenz zwischen Bild und dem Entwicklungspunkt $(x - \tilde{x})$:

$$R^c(x, \Phi_c) = \frac{\partial \Phi_c}{\partial x}|_{x=\tilde{x}} \odot (x - \tilde{x})$$

Für den gegebenen Klassifikator $\Phi_c(x)$ hat die Deep-Taylor-Methode eine freie Variable. Diese ist die Wahl des Entwicklungspunktes $\tilde{x}$, an welchem die Taylorentwicklung erfolgt. Ein gut gewählter Entwicklungspunkt enthält gerade nicht mehr die Information die im Datenpunkt x (dem Eingangssignal / Bild) dazu beitragen, dass der Klassifikator $\Phi_c(x)$ positiv ist, aber weicht nur minimal vom original x ab. Also ist $\tilde{x}$ ein Punkt mit $\Phi_c(\tilde{x}) = 0$, welcher in der Nähe von x unter einer bestimmten Metrik liegt. Falls $x, \tilde{x} \in \mathbb{R}^d$ kann gezeigt werden, dass für eine kontinuierlich differenzierbare Funktion Φ_c der Gradient an dem nächstgelegen Entwicklungspunkt immer in die gleiche Richtung wie $x - \tilde{x}$ zeigt und ihr Element-weises Produkt ist immer positiv, sodass sich eine positive Relevanzwerte Farbkarte ergibt. Diese Farbkarte ist nicht generell zwangsläufig konservativ für Φ_c, da Terme höherer Ordnung ungleich Null in ϵ enthalten sein können.
Der nächstgelegene Entwicklungspunkt $\tilde{x}$ kann durch Lösung eines Optimierungsproblems erhalten werden [29],[28, Kap. 2.2], durch Minimierung von:

$$\min_{\xi} \parallel \xi - x \parallel^2 \quad \text{mit} \quad \Phi_c(\xi) = 0 \text{ und } \xi \in \mathbb{R}^N,$$

wobei $\mathbb{R}^N$ der Vektorraum des Eingangssignals ist.
Neben den Problemen der Berechnung eines solchen nächsten Entwicklungspunktes $\tilde{x}$ im Rahmen einer iterativen Minimierung, welche zeitaufwendig ist, weicht für große neuronale Netzwerke der nächste Entwicklungspunkt $\tilde{x}$ erheblich von dem eigentlichen Datenpunkt x ab.

6.7 Sensitivitäts-Analyse

Sensitivitäts-Analyse kann als eine spezielle Erweiterung der Deep-Taylor-Methode angesehen werden [28, Kap. 2.2],[30]. Der Klassifikator $\Phi_c(x)$ wird hierbei nicht um einen Entwicklungspunkt $\tilde{x}$ erweitert, sondern um einen Punkt ξ, welcher sich in einem unendlich kleinen Abstand zum eigentlichen Punkt x in der Richtung des maximalen Gradienten befindet:

$$\xi = x - \delta \cdot \frac{\partial \Phi_c}{\partial x} \text{ mit kleinem } \delta$$

Bei den hier vorliegenden unendliche kleinen Abständen ist Φ_c lokal linear und der Gradient ist konstant, sodass sich die Taylorentwicklung von $\Phi_c(x)$ in ξ wie folgt schreiben lässt:

$$\Phi_c(x) = \Phi_c(\xi) + \left(\frac{\partial \Phi_c}{\partial x}|_{x=\xi}\right)^T \cdot (x - \xi) + \epsilon = \Phi_c(\xi) + \sum_{i=1}^{N} \underbrace{\delta \left(\frac{\partial \Phi_c}{\partial x_i}\right)^2}_{=R_i^c(x,\Phi_c)} + 0$$

hierbei kann der Term erster Ordnung identifiziert werden. Die resultierende Relevanzwerte Farbkarte ist positiv aber nicht konservativ, da ein Großteil der Relevanz in dem Term Null-ter Ordnung verbleibt.

6.8 Layer-wise Relevance Propagation (LRP)

Bei der LRP-Methode [31] wird das neuronale Netzwerk von hinten betrachtet. Die Relevanzwerte werden durch einen Schritt rückwärts, also Richtung Eingangsschicht, berechnet.
Sei wieder $\Phi = [\Phi_1, ..., \Phi_C] : \mathbb{R}^N \to \mathbb{R}^C$ ein Klassifikator mit dem Eingangssignal $x \in \mathbb{R}^N$ und das Eingangssignal / -bild bestehe aus den Anteilen / Pixeln $x = \{x_i\}$, wobei $i \in \{1, ..., N\}$ jeden Anteil / Pixel angibt. Der Klassifikator $\Phi_c(x)$ gibt die Existenz eines bestimmten Merkmales an. $\Phi_c(x) = 0$ impliziert, dass das Merkmal laut Klassifikator nicht vorhanden ist. Bei $\Phi_c(x) > 0$ ist das Vorhandensein des Merkmales mit einer gewissen Sicherheit gegeben. Im falle einer Wahrscheinlichkeitsangabe kann $> 50\%$ gewählt werden.
Die grundlegende Idee der LRP-Methode ist es, den Anteil eines jeden Anteils / Pixels x_i des Eingangssignals x zu einem positiven oder negativen Klassifikationsergebnis auszugeben und diesen Anteil durch ein Maß anzugeben, wobei der Klassifikator in mehrere Schichten zerlegt wird. Neuronale Netze bieten in Bezug auf diese Methode den Vorteil, sich in Schichten zerlegen zu lassen, z.B. in Teile der Merkmalserkennung bei Convolutional-Schichten. Wie üblich sei die erste Schicht das Eingangssignal und die letzte Schicht die Vorhersage des Klassifikators Φ_c. Die $l-$te Schicht kann dann dargestellt werden als $z = (z_d^l)_{d=1}^{V(l)}$ mit Dimension $V(l)$. Im Rahmen der schichtweisen Relevanz Propagierung wird angenommen, dass ein Relevanzwert r_d^{l+1} für jede Dimension $z_d^{(l+1)}$ von z der Schicht $l+1$ vorliegt. Um eine schichtweise Relevanz Propagierung durchführen zu können ist die Kenntnis über Relevanzwerte $r_d^{(l+1)}$ für jede Dimension $z_d^{(l+1)}$ für z in Schicht $l+1$ notwendig. Ziel ist es somit die Relevanzwerte $r_d^{(l)}$ für jede Dimension $z_d^{(l)}$ von z für dich nächste Schicht l herauszufinden, welche somit eine Schicht näher an dem Eingangsignal liegt [31, Gl. 2], sodass gilt:

$$\Phi(x) = ... = \sum_{d \in l+1} r_d^{(l+1)} = \sum_{d \in l} r_d = ... = \sum_d r_d^{(1)}$$

Durch Iteration von der letzten Schicht dem Klassifikationsergebnis $\Phi(x)$ bis zum Eingangssignal erhält man dann:

$$\Phi(x) \approx \sum_{d=1}^{V} r_d$$

wobei die Relevanz des Eingangssignals durch die gewünschte Zerlegung angegeben wird.

Bei der LRP-Methode erfolgt der Start in der Ausgangsschicht, wobei die Relevanz des Zielneurons c der Ausgangsschicht gleich dem Klassifikationsergebnis $\Phi_c(x)$ ist:

$$r_i^{(L)} = \begin{cases} \Phi_i(x) & \text{falls } i \text{ das Zielneuron ist} \\ 0 & \text{in allen anderen Fällen} \end{cases}$$

Die Relevanz der übrigen Neuronen ist Null. Hiernach wird das Netzwerk Schicht für Schicht durchlaufen und die Relevanz umverteilt ausgehend von $\Phi_c(x)$ bis die Eingangsschicht erreicht ist. Eine mögliche Regel zur Umverteilung der Relevanz in einer Schicht zu der nachfolgen Schicht, im LRP-Alogrithmus, ist die ϵ-Regel [25, Gl. 2]:

$$r_i^{(l)} = \sum_j \frac{z_{ji}}{\sum_{i'}(z_{ji'} + b_j) + \epsilon \cdot sign(\sum_{i'}(z_{ji'} + b_j))} r_j^{(l+1)},$$

wobei $z_{ji} = w_{ji}^{(l+1,l)} x_i^{(l)}$ die Gewichtung eines Neurons i zu einem Neuron j in der nächsten Schicht darstellt und b_j der Bias von Einheit j. Ein kleines ϵ liegt hierbei im Nenner vor um eine numerische Instabilität zu vermeiden [25, Kap. 2.2]. Sobald der Algorithmus die Eingangsschicht erreicht hat gilt: $R_i^c(x, \Phi_c) = r_i^{(1)}$. Diese Form des LRP wird aufgrund des ϵ als ϵ-LRP bezeichnet, es existieren jedoch weitere Stabilisierungsverfahren.

6.9 Deconvolution

In der Deconvolution-Methode wird ein Eingangssignal / -bild in einer hierarchischen Weise zerlegt mittels multipler alternierender Convolutional Sparse Coding (Deconvolution [32]) Schichten und Max-Pooling Schichten. Dieser Ansatz ist an die Convolutional neuronale Netzwerke angelegt, welche ebenfalls aus mehreren Schichten bestehen und alternierend Convolution-Schichten und Max-Pooling Schichten nutzen. Sei $\Phi : \mathbb{R}^N \to \mathbb{R}^C$ ein solches Convolutional neuronales Netzwerk, welches ein Bild $x \in \mathbb{R}^N$ mittels mehrerer Schichten auf einen Wahrscheinlichkeitsvektor $[\Phi_1(x), ..., \Phi_c(x)] = \Phi(x)$ abbildet für alle C Ausgabeneuronen. Dieses Netzwerk besteht jeweils immer aus 1. einer Convolution-Schicht, welche als Eingangssignal die Ausgabe der vorherigen Schicht benutzt oder im Falle der ersten Schicht das Eingangssignal / -bild, mit erlernten Filtern, 2. Anwendung des Rectifiers (rectified linear function) ($ReLu(x) = max(x, 0)$) und 3. (optional) einer Max-Pooling Schicht über lokale benachbarte Datenpunkte der Ausgabe. Da diese Methode hauptsächlich auf Bilder anwendbar ist, sei hier von einem Bild als Eingangssignal ausgegangen.
Betrachtet wird dementsprechend zunächst eine einzelne Schicht eines Deconvolution neuronalen Netzwerkes (Convolutional Sparse Coding Schicht) mit Eingangssignal eines Bildes $x = [x_1, ..., x_N]$, bestehend aus K_0 Farbkanälen $x_i^1, ..., x_i^{K_0}$ mit $i \in \{1, ..., N\}$. Jeder Kanal κ des Bildes wir als eine lineare Summe von K_1 unbekannten Eigenschaftskarten z_i^k gefaltet mit den Filtern $f_{k,\kappa}$ dargestellt [32, Gl. 1]:

$$\sum_{k=1}^{K_1} z_i^k \oplus f_{k,\kappa} = x_i^\kappa$$

Wobei y_i^κ ein Bild mit der Matrixgröße $N_r \times N_\kappa$ ist und die Filter eine Matrixgröße von $H \times H$ haben, sodass die unbekannten Eigenschaftskarten eine Matrixgröße von $(N_r + H - 1) \times (N_\kappa + H - 1)$ haben. Da $\sum_{k=1}^{K_1} z_i^k \oplus f_{k,\kappa} = x_i^\kappa$ ein unterbestimmtes Gleichungssystem ist, wird um eine eindeutige Lösung zu erhalten ein Regularisations-Term für z_i^k eingeführt, welcher Spärlichkeit (sparsity) in der Eigenschaftskarte

fördert. Hierdurch wird insgesamt eine Kostenfunktion der folgenden Form erhalten [32, Gl. 2]:

$$C_1(x_i) = \frac{\lambda}{2} \sum_{\kappa=1}^{K_0} \| \sum_{k=1}^{K_1} z_i^k \oplus f_{k,\kappa} - x_i^\kappa \|_2^2 + \sum_{k=1}^{K_1} |z_i^k|^p$$

Hierbei wird Störung (noise) in Gauß-Verteilung für den Rekonstruktions-Term und eine Spärlichkeits-Norm p für die Regularization angenommen. Die Spärlichektis-Norm $|w|^p$ ist hierbei die p-Norm für vektorisierte Matrizen mit $|w|^p = \sum_{i,j} |w(i,j)|^p$ und typischerweise $p = 1$. λ ist eine Konstante, welche die Verteilung zwischen dem relativen Beitrag der Rekonstruktion von x_i und der Spärlichkeit der Eigenschaftskarten z_i^k. Hierbei ist zu anzumerken, dass dieses Methode vom Ende zum Anfang läuft: Gegeben die unbekannten Eigenschaftskarten kann ein Bild synthetisiert werden, aber anders herum gibt es keinen Mechanismus um die Eigenschaftskarten aus dem Eingangssignal / Eingangsbild zu generieren, außer die Minimierung der Kostenfunktion C_1.
Eine einzelne Schicht eines Deconvolution neuronalen Netzwerkes ergibt somit spärliche Eigenschaftskarten aus einem Eingangsbild mit mehreren Kanälen. Hierdurch kann einfach eine Hierarchie aufgebaut werden. Es werden die Eigenschaftskarten $z_i^{k,l}$ der Schicht l als Eingangssignal für die Schicht $l+1$ benutzt. Somit hat Schicht l als Eingangssignal ein Bild mit K_{l-1} Kanälen, welche die Anzahl der Eigenschaftskarten der Schicht $l-1$ sind. Die Kostenfunktion C_l der entsprechenden Schicht l ergibt sich somit als Generalisierung von der einzelnen Schicht [32, Gl. 3]:

$$C_l(x) = \frac{\lambda}{2} \sum_{i=1}^{N} \sum_{\kappa=1}^{K_{l-1}} \| \sum_{k=1}^{K_l} g_l^{k,\kappa}(z_i^{k,l} \oplus f_l^{k,\kappa}) - z_i^{\kappa,l-1} \|_2^2 + \sum_{i=1}^{N} \sum_{k=1}^{K_l} |z_i^{k,l}|^p,$$

wobei $z_i^{\kappa,l-1}$ die Eigenschaftskarten der vorhergegangenen Schicht sind und $g_l^{k,\kappa}$ sind Elemente einer konstanten binären Matrix, welche die Verbindungen im Netzwerk (Connectivity) zwischen zwei Schichten bestimmt, also ob $z_i^{k,l}$ mit $z_i^{\kappa,l-1}$ verbunden ist oder nicht.
Dieses Modell lässt sich von unten nach oben aufbauen und trainieren. In der ersten Schicht ist angenommen das $g_1^{k,\kappa}$ immer 1 ist, also immer eine Verbindung zu allen Elementen der zweiten Schicht besteht. In höheren Schichten wird die Verbindungsdichte dann spärlicher. Durch den Aufbau von unten ist $z_i^{\kappa,l-1}$ gegeben durch das Ergebnis des Trainings mit $C_{l-1}(x)$.
Um die Filter zu lernen wird alternierend $C_l(x)$ über die Eigenschaftskarten minimiert, während die Filter konstant bleiben und dann wieder $C_l(x)$ über die Filter minimiert, während die Eigenschaftskarten konstant bleiben. Diese Minimierung wird schichtweise durchgeführt beginnend mit der ersten Schicht. Das Eingangssignal der ersten Schicht sind die Trainingsdaten / Bilder x.
Neben dem hierarchischen Aufbau von Deconvolution-Schichten lassen sich 3D Max-Pooling Schichten zwischen die Deconvolution-Schichten einbauen und ein Netzwerk mit multiplen Schichten trainieren [33]. Ein solches Netzwerk lässt sich trainieren mit dem Ziel Eigenschaften zu extrahieren und ist hierbei vollkommen unbeaufsichtigt (unsupervised), weshalb es mit einem Klassifikator kombiniert werden muss um eine Klassifikation zu erhalten [33]. Des Weiteren erlaubt die Architektur des Modells eine einfache Interpretation des Erlernten.
Um nun einen Klassifikator Φ, welcher wie ein Convolutional neuronales Netzwerk aufgebaut ist, zu analysieren und Relevanzkarten zu erstellen, wird parallel zu den Convolution-Schichten jeweils immer eine Deconvolution-Schicht eingebaut. Dies ermöglicht einen kontinuierlichen Weg zurück zu den Pixeln x_i des Bildes x. Initial wird ein Bild (Eingangssignal) durch Φ klassifiziert. Bei der Analyse werden alle Aktivierungen außer eine in einer Schicht auf Null gesetzt und die Eigenschaftskarten als Eingangssignal für die parallele Deconvolution-Schicht benutzt. Hiernach wird 1. entbündelt (unpool), 2. rectifiziert und gefiltert um zu rekonstruieren, welche Aktivität in der vorherigen Schicht die Aktivitäten in der aktuellen

Schicht hervorgerufen hat. Dies wird sukzessiv fortgeführt bis der Raum des Eingangssignals erreicht ist [34].

6.10 Guided Backpropagation

Springenberg et al. [35] beobachteten, dass die Deconvolution-Methode für das von ihnen vorgestellte Netzwerk zur Klassifikation von ImageNet Daten nicht durchgehend gut funktionierte, sobald die Max-Pooling Schichten entfernt wurden, sodass sie die Guided-Backpropagation-Methode vorgeschlagen haben. Die Max-Pooling Schichten sind nicht invertierbar, sodass bei der Deconvolution-Methode zunächst ein vorwärts Durchlauf durch das Netzwerk berechnet wird, welcher die Positionen der Maxima in jeder Pooling-Region angibt. Diese werden dann in der entsprechenden Deconvolution-Schicht benutzt, um eine Rekonstruktion zu berechnen [34]. An diesem Vorgehen lässt sich kritisieren, dass die Maxima einer jeden Pooling-Region durch das Eingangssignal bedingt sind und im Rahmen ihrer Nutzung nicht nur erlernte Eigenschaften dargestellt werden. Durch Entfernung der Max-Pooling Schichten wie durch Springenberg et al. [35] vorgeschlagen, kann die Deconvolution-Methode erfolgen, ohne einen vorherigen vorwärts Durchlauf, sodass in der Theorie nur erlernte Eigenschaften dargestellt werden.
Bei Entfernung der Max-Pooling Schichten wurde beobachtet, dass weniger scharfe und nicht eindeutig identifizierbare Bildstrukturen markiert wurden. Insgesamt passt diese Beobachtung dazu, dass niedrigere Schichten eines Convolutional-Netzwerkes generelle Eigenschaften mit einer limitierten Varianz erlernen, wohingegen höhere Schichten variablere Repräsentationen erlernen. Aus diesem Grund ist eine Bedingung auf ein Eingangssignal notwendig um eine sinnvolle Rekonstruktion zu erhalten.
Eine alternative Methode um darzustellen, welcher Anteil des Eingangssignals / -bildes ein gegebenes Neuron am stärksten aktiviert, ist wie in der Layer-wise Relevance Propagation bereits diskutiert eine einfache Rückwärtspropagation der Aktivierung eines einzelnen Neurons, nach einem Vorwärtsdurchlauf durch das Netzwerk. Hierbei werden die Gradienten der Aktivierungsfunktion bezüglich des Eingangssignals berechnet. Rückwärtspropagation ist durch das Eingangssignal bedingt aufgrund der Aktivierungsfunktionen des neuronalen Netzwerkes und der Max-Pooling Schichten.
Im Rahmen der Guided-Backpropagation-Methode werden Anteile dieser beiden Methoden kombiniert. Bei der Deconvolution-Methode wird das Netzwerk rückwärts durchlaufen außer im Fall einer nicht-Linearität. Hier wird der Gradient durch das Signal der höheren Schichten unabhängig vom Eingangssignal bestimmt. Im Falle der Rückwärtspropagation werden abhängig von den höheren Gradienten Einträge auf Null gesetzt aufgrund der ReLu-Funktion. Bei der Guided-Backpropagation-Methode werden sowohl Werte, welche negative Einträge in den nachgeschalteten Gradienten hervorrufen, auf Null gesetzt als auch Werte, welche dies in den vorherigen Gradienten bewirken, sodass hier die Rückwärtspropagation und Deconvolution kombiniert sind. Sei $f_i^{l+1} = ReLu(f_i^l)$ die Aktivierungsfunktion angewendet auf die einzelnen Schichten und $r_i^{l+1} = \frac{\partial f^{out}}{\partial f_i^{l+1}}$ die Propogation der Aktivierung „out“ durch ein $ReLu$ in Schicht l, dann kann die Rückwärtspropagation wie folgt dargestellt werden [35, Fig. 1]:

$$r_i^l = (f_i^l > 0) \cdot r_i^l$$

und die Deconvolution Netzwerk:

$$r_i^l = (r_i^{l+1} > 0) \cdot r - i^{l+1}$$

Die Guided-Backpropagation-Methode ist dann:

$$r_i^l = (f_i^l > 0) \cdot (r - i^{l+1} > 0) \cdot r_i^{l+1}$$

6.11 Gradient-weighted Class Activitaion Mapping (GradCAM)

Die GradCAM Methode hat zum Ziel, die Verständlichkeit der Klassifikation von Convolutional neuonale Netzwerke zu verbessern, indem wichtige Regionen des Eingangssignales / -bildes hervorgehoben werden und somit eine visuelle Erklärung bieten [36].
Wie der Name schon nahelegt basiert GradCAM auf Klassenaktivierungsabbdilungen (Class Acitvation Mapping (CAM)). Die CAM-Methode gibt Relevanzkarten aus, wobei jedoch ein Bildklassifikator mit CNN vorliegen muss mit Übergang von den letzten Eigenschaftskarten über globale gepoolte Mittelung (Global-Average-Pooling) direkt in den letzten Softmax-Layer (Klassifikation Schicht mit Softmax-Funktion). Sei die vorletzte Schicht eine Convolution-Schicht mit K Eingenschaftskarten $A^k \in \mathbb{R}^{u \times v}$ mit $k \in \{1, ..., K\}$. Bevor sie linear transformiert werden, werden diese Eigenschaftskarten gepoolt mit Global Aerage Pooling (GAP), sodass final ein Wert Φ_c für jede Klasse $c \in \{1, ..., C\}$ ausgegeben wird [36, Gl. 1]:

$$\Phi_c = \sum_k w_c^k \frac{1}{Z} \sum_i \sum_j A_{i,j}^k$$

Die Relevanzkarten $R_{CAM}^c \in \mathbb{R}^{u \times v}$ für eine Klasse c ergeben sich aus der linearen Kombination der letzten Eigenschaftskarten und den gelernten Gewichten der letzten Schicht: $R_{CAM}^c = \sum_k w_c^k A^k$. Diese werden anschließen normalisiert, damit die Werte zwischen 0 und 1 liegen.
Im Rahmen der Gradienten gewichteten Klassenaktivierungsabbildung (Gradient-weighted Class Activitaion Mapping (GradCAM)) Methode wird um die Relevanzkarten $R_{GradCAM}^c \in \mathbb{R}^{u \times v}$ zu berechnen zunächst der Gradient der Ausgabe für eine Klasse Φ_c zu den Eigenschaftskarten A einer Convolutional-Schicht berechnet: $\frac{\partial \Phi_c}{\partial A_{i,j}^k}$. Durch globale gepoolte Mittelung (Global-Average-Pooling) werden Gewichte berechnet α_c^k [36, Gl. 2]:

$$\alpha_c^k = \frac{1}{Z} \sum_i \sum_j \frac{\partial \Phi_c}{\partial A_{i,j}^k}$$

Laut den Autoren repräsentieren somit diese Gewichte eine partielle Linearisierung des neuronalen Netzwerkes ausgehend vom Eingangssignal und stellen die Relevanz der k-ten Eigenschaftskarten für eine Klasse c dar. Analog zur CAM-Methode erfolgt bei der GradCAM-Methode eine Gewichtung der Eigenschaftskarten, jedoch wird diese zusätzlich durch eine $ReLu$-Funktion ergänzt [36, Gl. 3]:

$$R_{GradCAM}^c = ReLu(\sum_k \alpha_c^k A^k)$$

Die resultierende Relevanzkarte wird noch normalisiert zur besseren Darstellung. Diese Methode ist somit ebenfalls auf Convolutional neuronale Netzwerke beschränkt, jedoch muss auf die letzte Convolutional-Schicht nicht direkt die Klassifikationsausgabe durch eine Softmax-Schicht erfolgen.

6.12 Shapley values

Ein approximierter Shapley-Wert wird benutzt, um zu identifizieren, welche Eigenschaften zur Vorhersage eines Models geführt haben. Hierbei wird der Wert als Beitrag einer Eigenschaft zu der Differenz zwischen der aktuellen Vorhersage und der durchschnittlichen Vorhersage berechnet. Normalerweise werden Eigenschaften, welche eine hohe positive Vorhersage bedingen, rot dargestellt, wohingegen niedrige Vorhersagen in blau dargestellt werden. Ein deutlicher Vorteil der Shapley-Werte ist hierbei, dass die Differenz zwischen einer Vorhersage und der durchschnittlichen Vorhersage relativ verteilt ist über alle Eigenschaften.
SHAP (SHapley Additive exPlanations) gehört zu den am häufigsten zitierten Methoden zur quantifizierten Erklärung /Relevanz [37]. Dieser Ansatz wurde von der Spieltheorie abgeleitet und kann auf jede

Art von Deep-Learning Model angewendet werden.
Die Frage hierbei lautet: Wie wichtig ist ein jeder Teilnehmer oder Teilnehmerin einer Kooperation und welcher Anteil kann sinnvollerweise erwartet werden [38, 39]. Initial wurden Shapley-Werte somit in der Spieltheorie definiert:
Für eine Menge N bestehend aus n Spielern gibt es eine Funktion ν, welche Teilmenge von N in die realen Zahlen Abbildet $\nu : 2^N \to \mathbb{R}$, mit $\nu(\emptyset) = 0$. Hierbei gibt $\nu(S)$ (S ist eine Koalition von Spielern oder Spielerinnen im Folgenden als Spieler bezeichnet) an wie viel alle Spieler aus S als Anteil erwarten können, wenn sie kooperieren. (ν, N) wird als Koalitionsspiel bezeichnet. Shapley-Werte stellen eine Methode dar die Gewinne unter den Spielern aufzuteilen unter der Annahme, dass sie kooperieren. Den Anteil den Spieler i in einem Spiel (ν, N) erhält ist [40, Gl. 1]:

$$\phi_i(\nu) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(n - |S| - 1)!}{n!} (\nu(S \cup \{i\}) - \nu(S))$$

wobei n die Anzahl aller Spieler ist und die Summe über alle Teilmengen S von N ohne Spieler i genommen wird [40]. Um dieses Konzept nun auf neuronale Netzwerke zu übertragen, sei wieder $x = [x_1, ..., x_N] \in \mathbb{R}^N$ ein Eingangssignal für einen Klassifikator $\Phi(x) = [\Phi_1(x), ..., \Phi_C(x)] \in \mathbb{R}^C$. Sei hierbei Φ ein feed-forward neuronales Netzwerk mit Schichten und nicht-linearer Aktivierungsfunktion σ mit

$$\Phi^{(i)}(x^{i-1}) = \sigma(W^{(i)} x^{(i-1)} + b^{(i)}),$$

wobei i die Schicht angibt mit $1 \leq i \leq l$. Es sollen nun die Shapley-Werte berechnet werden für einen Ausgabewert eines neuronales Netzwerkes $\Phi(x) = (\Phi^{(1)} \circ \Phi^{(2)} \circ ... \circ \Phi^{(l)})(x)$ bestehend aus mehreren Schichten.
Wie im spieltheoretischen Fall hängt sind die Shapley-Werte eines Anteils des Eingangssignals gegeben durch den marginalen Anteil zu allen möglichen 2^{N-1} Koalitionen, welche aus den übrigen Anteilen des Eingangssignals gebildet werden können. Dies kann durch den Durchschnitt über zufällige Koalitionen approximiert werden. Der Durchschnitt all dieser marginalen Beiträge gibt dann den Shapley-Wert von x_i an, sodass Beitrag von x_i gegeben ist durch [26, Gl. 5]:

$$\mathbb{E}[R_i^c] = \frac{1}{N} \sum_{k=0}^{N-1} \mathbb{E}_k[R_{i,k}^c],$$

hierbei wird der Erwartungswert $\mathbb{E}_k$ über die Verteilung von Mengen der Mächtigkeit k gebildet. Dies sind Koalitionen der Mächtigkeit k mit $0 \leq k \leq N - 1$, wobei jedoch der Anteil x_i des Eingangssignals nicht enthalten ist. $R_{i,k}^c$ gibt den Beitrag eines Anteils des Eingangssignals x_i zu einer zufälligen Koalition der Mächtigkeit k an.
Für eine gegebene Koalitionsgröße k kann der erwartete Beitrag eines Anteils des Eingangssignals x_i angegeben werden [26, Gl. 6]:

$$\mathbb{E}_k[R_{i,k}^c] = \mathop{\mathbb{E}}_{\substack{S \subseteq P \setminus \{i\} \\ |S| = k}} [\Phi_c(x_{S \cup \{i\}})] - \mathop{\mathbb{E}}_{\substack{S \subseteq P \setminus \{i\} \\ |S| = k}} [\Phi_c(x_S)]$$

Das Problem ist nun die Berechnung dieser Erwartungswerte für alle Koalitionen mit einer Größe $0 \leq k \leq N - 1$. Diese können mit der Varianz durch das Netzwerk propagiert und berechnet werden.
Ein Nachteil der Shapley-Werte liegt darin, dass diese ebenfalls NP-schwierig zu berechnen sind [26].

6.13 Local Interpretable Model-agnostic Explanations (LIME)

Die Lime-Methode [41] hat zum Ziel größtmögliche Interpretierbarkeit, ein qualitatives Verständnis zwischen Eingangssignal und Klassifikationsergebnis, und lokale Sicherheit herzustellen. Hierzu sei $g \in G$ ein Erklärungsmodel in der Klasse aller potentiell interpretierbaren Modelle, z.B. Entscheidungsbäume. Angesichts dessen, dass nicht jedes $g \in G$ einfach interpretierbar ist, sei $\Omega(g)$ ein Maß für die Komplexität (entgegensetzt zur Interpretierbarkeit). Z.B. kann für Entscheidungsbäume die Tiefe des Entscheidungsbaumes ein Maß für die Komplexität sein, wohingegen bei linearen Modellen die Anzahl an Gewichten ungleich Null die Komplexität angeben kann.
Sei der hier betrachtete Klassifikator $\Phi : \mathbb{R}^N \to \mathbb{R}^C$. Zudem sei $\phi_x(z)$ ein Abstandsmaß zwischen z und x, um Lokalität um x zu definieren. Des Weiteren sei $\mathcal{L}(\Phi, g, \phi_x)$ ein Maß für das Versagen von g, Φ in dem Bereich definiert durch ϕ_x zu approximieren. Um größtmögliche Interpretierbarkeit und lokale richtige Erklärung zu erhalten, muss $\mathcal{L}(\Phi, g, \phi_x)$ minimiert werden, jedoch muss $\Omega(g)$ klein genug bleiben, sodass es noch verständlich bleibt für Menschen. Hierdurch ergibt sich die Erklärung, welche durch den LIME-Algorithmus erhalten werden [41, Gl. 1]:

$$\xi(x) = \underset{g \in G}{\operatorname{argmin}}\ \mathcal{L}(\Phi, g, \phi_x) + \Omega(g)$$

Dieser Ansatz kann mit unterschiedlichen Erklärungsmodellen G, lokalen richtigen Erklärungsfunktionen $\mathcal{L}$ und Komplexitätsmaßen Ω betrachtet werden. Oft werden spärliche lineare Modelle als Erklärungsmodelle und Perturbationen benutzt. Beispielsweise kann G die Klasse der linearen Modelle sein mit $g(z') = w_g \cdot z'$. Weiter die lokal gewichtete quadratische Verlustfunktion [41, Gl. 2]:

$$\mathcal{L}(\Phi, g, \phi_x) = \sum_{z, z' \in \mathcal{Z}} \phi_x(z)(\Phi(z) - g(z'))^2$$

mit $\phi_x(z) = exp(-D(x, z)^2/\sigma^2)$ als exponentiellen Kernel definiert auf einer Abstandsfunktion D (z.B. $L2$-Abstand für Bilder) mit Breite σ. Die Hauptmerkmale dieser Methode lassen sich wie folgt zusammenfassen:

- Ein Bild wird segmentiert in sogenannte Superpixel, welche Regionen gleicher Farbe / Textur mit einem Typ von visuellem Kontext entsprechen.
- Durch stochastische Permutation des Originalbildes wird eine Verteilung von permutierten Bildern von dem segmentierten Originalbild erstellt.
- Die permutierten Bilder werden von dem zu analysierenden Klassifikator prozessiert und somit die Ausgabewahrscheinlichkeiten berechnet.
- Mit der permutierten Verteilung des Originalbildes und den entsprechenden Klassifikationswahrscheinlichkeiten wird ein übergeordnetes lineares Modelt trainiert, um das CNN lokal zu approximieren für des entsprechende Bild.
- Die Gewichte des linearen Modells implizieren in welchem Ausmaß jedes Superpixel, positiv oder negativ, zur Klassifikation beiträgt.
- Diese Gewichte werden auf einer Relevanzkarte aufgetragen, was die Ausgabe der LIME-Methode darstellt.

6.14 Squaregrid

Bei der Squaregrid-Methode wird das Eingangssignal / Bild durch mehrere Gitter mit unterschiedlicher Größe (beispielsweise $4, 9, 16, 25, 36, ...$) geteilt und für jedes der Quadrate die LIME-Methode separat angewendet. Die Ergebnisse der LIME-Methode werden anschließend aufsummiert [42, 7].

7 Numerische Experimente

Im Folgenden werden numerische Experimente zu Interpretationen von neuronalen Netzwerken betrachtet. Insbesondere soll die vorgestellte Methode RDE (Rate-Distortion Explanation) mit mehreren aktuell üblichen Methoden verglichen werden. Hierzu werden für den MNIST Datensatz [43] mit handgeschriebenen Ziffern in Graustufen Relevanzkarten erstellt ebenso wie für den STL-10 Datensatz mit Farbbildern analog zu Macdonald et al. [6].
Zudem wird noch der Vorläufer des STL-10 Datensatzes der Cifar-10 Datensatz analysiert werden und ein Beispiel einer Klassifikation aus der radiologischen Klinik in Basel. Analog zu Macdonald et al. wird als Referenz-Verteilung $\mathcal{V}$ eine Gauss-Verteilung angenommen mit Mittelwert und Varianz / Faktorisierung von niedrigem Rang der Covarianz-Matrix, welche aus den Trainingsdaten geschätzt wurden. Die Faktoriesierung von niedrigem Rang wird durch eine gekürzte Singulärwertzerlegung mit Rang $r = 30$ erhalten. Ebenfalls analog wird während der Gradientenabstiegsoptimierung (Gradient Descent Optimisation) ein Momentums-Term mit dem Faktor 0.85 benutzt und die Schrittgröße wird durch eine zurückverfolgte Armijo-Zeilensuche angegeben. Die Initialisierung der Konstanten-Karte erfolgte mit $s = 0.25 \cdot 1_d$ (nicht mit dem Wert der Publikation, jedoch der Wert im Code auf GitHub [44]) und der Regularisierungsparameter wurde auf $\lambda = [0.1, 0.5, 1]$ für den MNIST-Datensatz gesetzt, für den STL-10 und Cifar-10 Datensatz wurde der Regularisierungsparameter ebenfalls auf $\lambda = [0.1, 0.5, 1]$ gesetzt.
Die RDE Methode wird mit mehreren anderen Methoden verglichen: Layer-wise Relevance Propagation (LRP) [45], Deep Taylor decompositions, Sensitivity Analysis, SmoothGrad, Guided Backprop, SHAP und LIME. Hierzu werden die Innvestigate [46, 47], SHAP [48] und LIME [49] toolboxen benutzt.

Unterschiedliche Interpretationsansätze liefern unterschiedlich skalierte und normalisierte Relevanzkarten. Einige Methoden generieren nicht-negative Abbildungen entsprechend der Wichtigkeit (Importance) für den Klassifizierungsscore. Andere Methoden generieren ebenfalls negative Relevanzwerte, welche interpretiert werden können als entgegengesetzt der Klassifikationsentscheidung. Diese deutlichen Unterschiede in den Ergebnissen der unterschiedlichen Interpretationsansätze machen diese schwer vergleichbar.

Um einen Vergleich zu ermöglichen, wird eine Variante des Relevanzordnung-basierten (relevance ordering-based) Tests benutzt. Jede Relevanz-Abbildung induziert eine Ordnung der Eingangssignal-Komponenten, indem diese nahe ihrer Relevanz sortiert werden. Man kann hierbei mit vollständig zufälligem Signal starten und zunehmend immer größere Teile durch die ursprünglichen Eingangswerte ersetzen und hierbei die Änderung des Klassifikationsscores beobachten. Dies wird dann gemittelt über mehrere Variationen des zufälligen Anteils des Eingangssignals. Eine gute Relevanzkarte wird selbst bei vielen zufälligen Anteilen schon die richtige Klassifikation ermöglichen, wenn die wichtigsten Komponenten fixiert sind, oder anders die Distortion geht relativ schnell gegen Null. Der beschriebene Prozess erlaubt es, ungefähr die Distortionsraten-Funktion approximieren.

Da Macdonald et al. [6] sich in ihrem Paper auf die Interpretierbarkeit von neuronalen Netzwerken konzentriert haben und nicht deren Training (und diese auch nicht online zur Verfügung gestellt wurden), konnten nur neue Netzwerke trainiert werden, welche die gleichen Voraussetzungen erfüllen.

7.1 MNIST Experiment

Es wurde das vorgeschlagene Convolutional Neural Network (drei Convolution Schichten, auf welche jeweils ein average-pooling folgt und am Ende zwei vollständig verknüpfte (fully-connected) Schichten

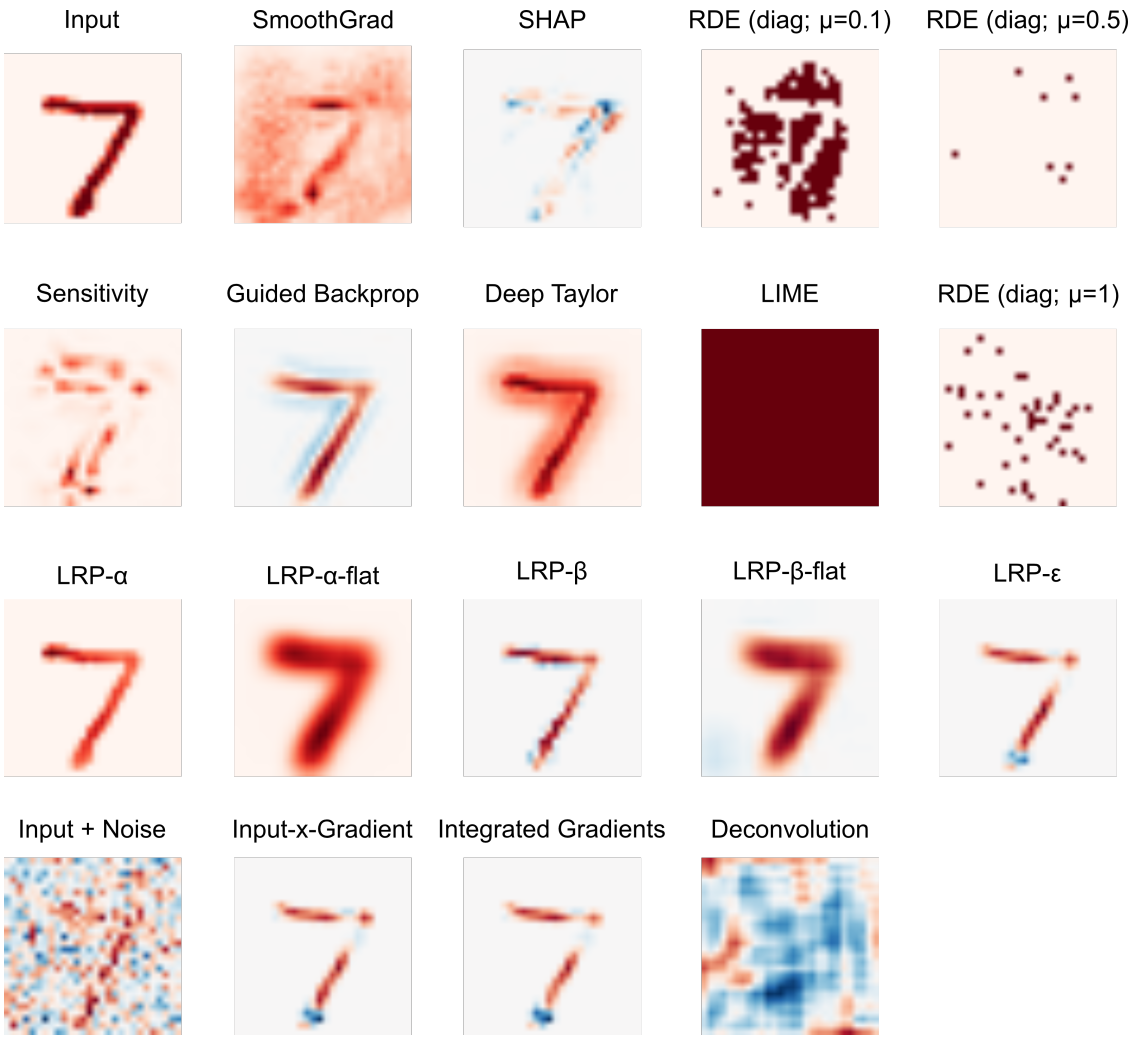

Abbildung 4: Relevanzkarten für ein Beispielbild des MNIST Datensatzes (Ziffer: Sieben) von mehreren Methoden (über der jeweiligen Relevanzkarte angegeben). Ein positiver Effekt ist in den Farbkarten rot und ein negativer blau dargestellt.

und eine Softmax-Ausgabe) End-zu-End trainiert bis eine Test-accuracy von 0.99 erreicht wurde. Die Standard-Aufteilung in Training / Validation / Testing des Datensatz wurde benutzt.

Relevanzkarten für ein Beispielbild von der Ziffer sieben ist in der Abbildung 4 dargestellt.
Durch Anwendung des Relevanzordnung-basierten Tests werden die in Abbildung 5 dargestellten Kurven berechnet.
Zum Vergleich sind in Abbildung 6 die Ergebnisse des Macdonald-Papers dargestellt. Hierbei fällt auf, dass sich mittels des neu trainierten Netzwerkes eine deutlich höhere Distortion zeigt als in dem Macdonald-Paper.

7.2 Cifar-10 Experiment

Bevor der STL-10 Datensatz analysiert wird, wird auf den Vorgängerdatensatz Cifar-10 eingegangen. Analog zu dem vorgeschlagenen Training für den STL-10 Datensatz wurde für den Cifar-10 Datensatz ein VGG-16 Netzwerk benutzt, welches vortrainiert auf den Imagenet Datensatz war. Dieses Netzwerk wurde

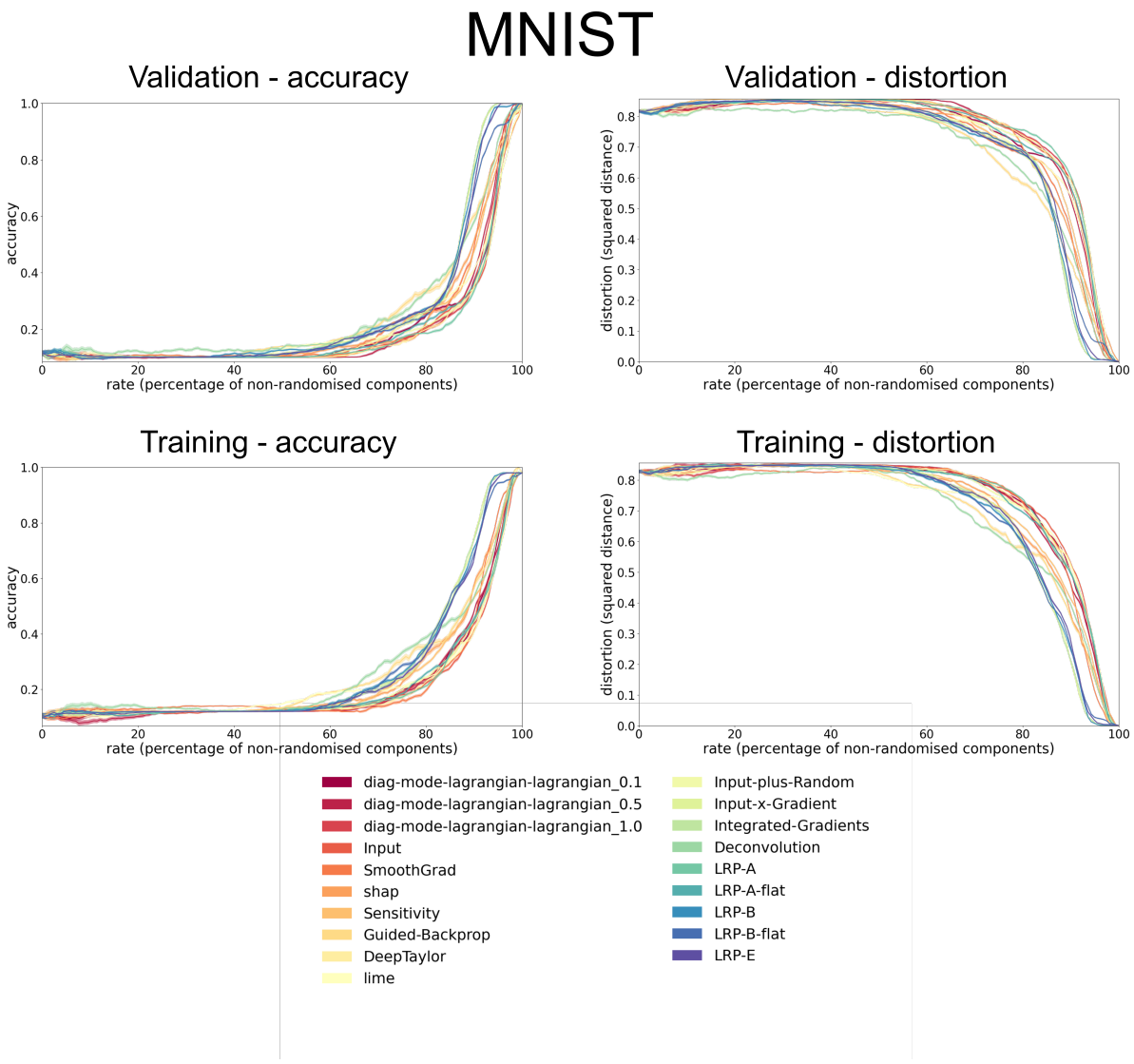

Abbildung 5: Ergebnisse des Relevanzordnung-basierten Tests (approximate rate-distortion Funktion) von mehreren Methoden für den MNIST Datensatz mit Angabe der Accuracy und der Distortion für den Test-Datensatz und den Trainings-Datensatz.

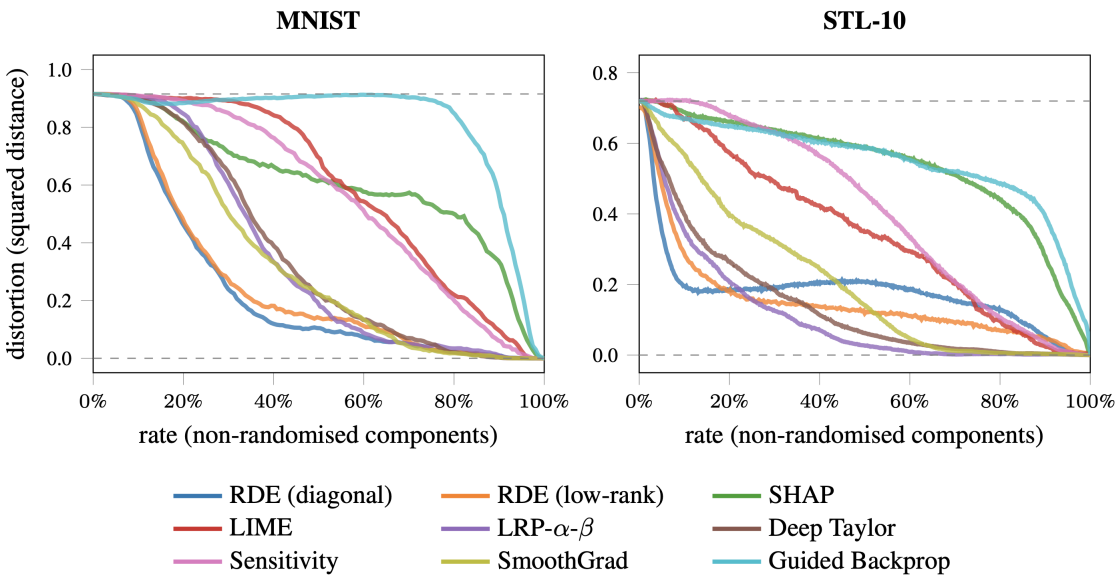

Abbildung 6: Ergebnisse des Relevanzordnung-basierten Tests (approximate rate-distortion Funktion) von mehreren Methoden für den MNIST Datensatz mit Angabe der Accuracy und der Distortion von Macdonald et al. [6].

dann umtrainiert auf den Cifar-10 Datensatz. Als erstes wurden nur die vollständig-verknüpften Schichten für 500 Epochen trainiert. Anschließend wurde das Netzwerk End-zu-End für weitere 500 Epochen trainiert und als drittes wurden die im VGG-16 enthaltenen Max-pooling Schichten durch average-pooling Schichten ersetzt und das Netzwerk wurde erneut für 500 Epochen trainiert. Schlussendlich wurde einer Test-Accuracy von 0.77 erreicht, die Trainings-Accuracy lag bei 0.98. Die Test-/Trainings Accuracy und Loss sind in Abbildung 7 dargestellt.
Die Relevanzkarten für ein Beispielbild eines Trucks sind in Abbildung 8 gezeigt. Wie zuvor sind die Relevanzkarten berechnet worden von dem Klassifikationsscore des Netzwerkes vor des Softmax-Funktion.

7.3 STL-10 Experiment

Im Gegensatz zu Macdonald et al. wurde das Netzwerk zur Klassifikation des STL-10 Datensatzes nicht neu nach dem oben beschriebenen Verfahren trainiert, sondern das Netzwerk zur Klassifikation des Cifar-10 Datensatzes benutzt und die Größe des Eingangssignals angepasst auf 96×96 Pixel. Des Weiteren musste noch die erste der beiden nachgeschalteten vollständig verknüpften Schichten von der Größe angepasst werden. Mittels dieser Anpassungen konnte das Netzwerk auf eine Test-accuracy von 0.73 und eine Trainings-accuray von 0.99 trainiert werden. Abbildung 10 zeigt die Relevanzkarten für ein Vogel.
Ebenfalls in diesem Datenbeispiel ist die Leistung der RDE-Methode schlechter als von Macdonald et al. beschrieben und weißt dementsprechend eine höhere Distortion auf. Hierbei sind jedoch einige mögliche Fehlerquellen aufgefallen. Als erstes muss angemerkt werden, dass das hier benutzte Netzwerk nur eine Test-accuracy von 0.73 erreicht und nicht von 0.935 wie im Macdonald-Paper, jedoch belief sich die Trainings-accuracy bereits auf 0.99, weshalb ein Overfitting möglich erscheint. Des Weiteren finden sich im GitHub-Code des Macdonald-Papers ein Eingangssignal für das STL-10-Model von 224×224 Pixel. Diese Größe wäre für den Imagenet Datensatz die wahrscheinlichste Größe, jedoch ist der STL-10 Datensatz explizit auf 96×96 Pixel angegeben, sodass hier entweder von einer entsprechenden Skalierung ausgegangen werden muss oder die Grundlage eines anderen Datensatzes angenommen werden muss.

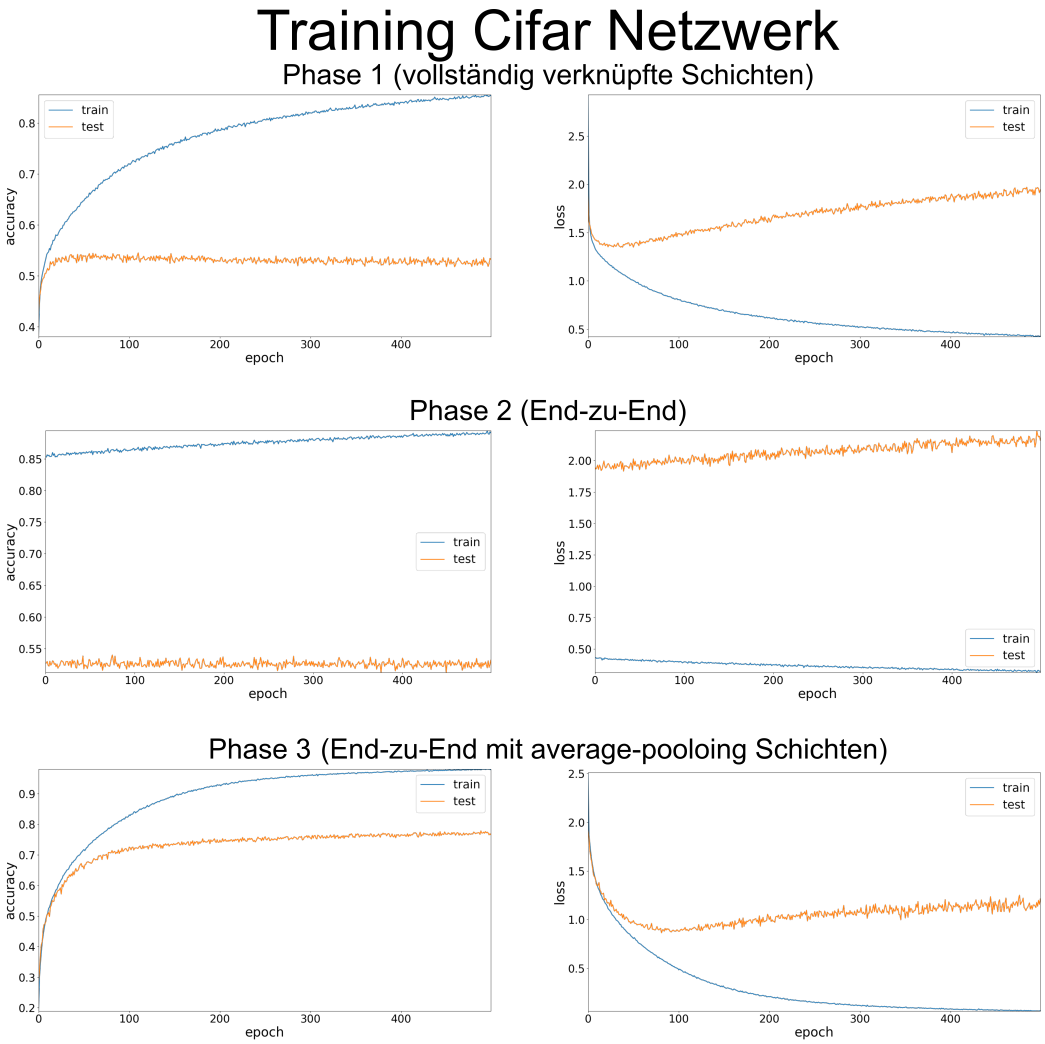

Abbildung 7: Test-/Trainings Accuracy und Loss für die 3 Trainings-Abschnitte.

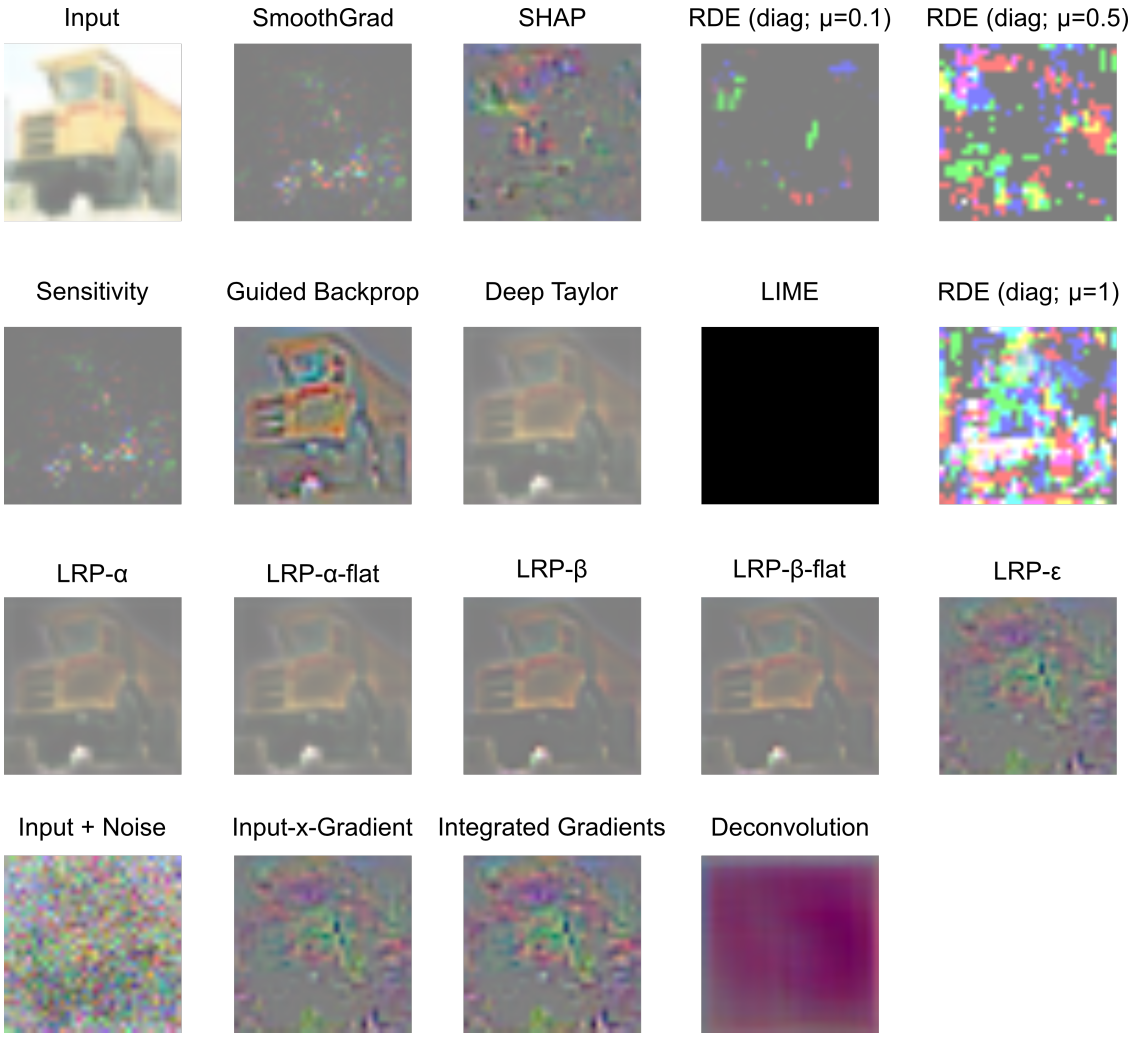

Abbildung 8: Relevanzkarten für ein Beispielbild des Cifar-10 Datensatzes (Lastwagen (Truck)) von mehreren Methoden (über der jeweiligen Relevanzkarte angegeben). Ein positiver Effekt ist in den Farbkarten rot und ein negativer blau dargestellt.

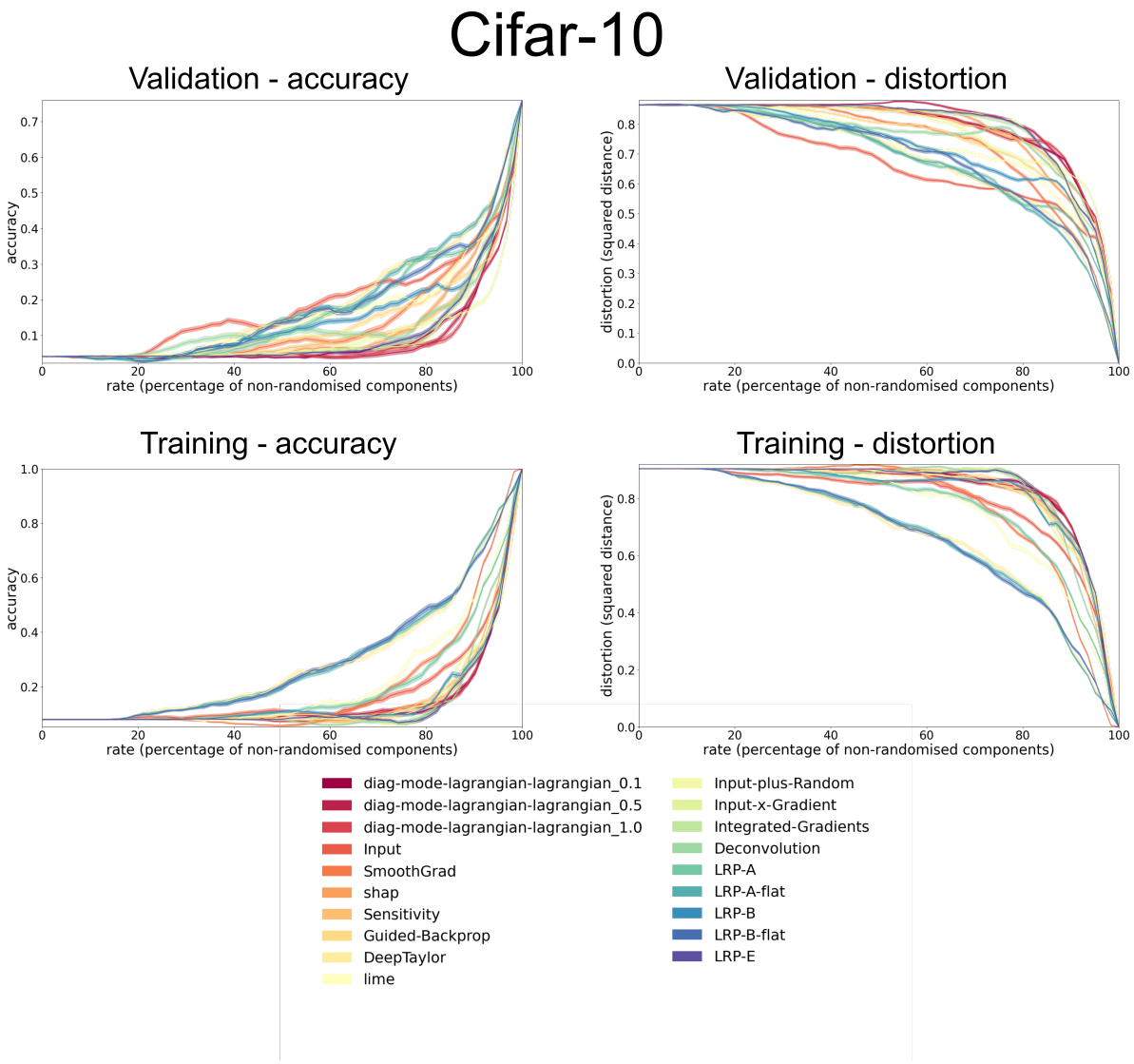

Abbildung 9: Ergebnisse des Relevanzordnung-basierten Tests (approximate rate-distortion Funktion) von mehreren Methoden für den Cifar-10 Datensatz mit Angabe der Accuracy und der Distortion für den Test-Datensatz und den Trainings-Datensatz.

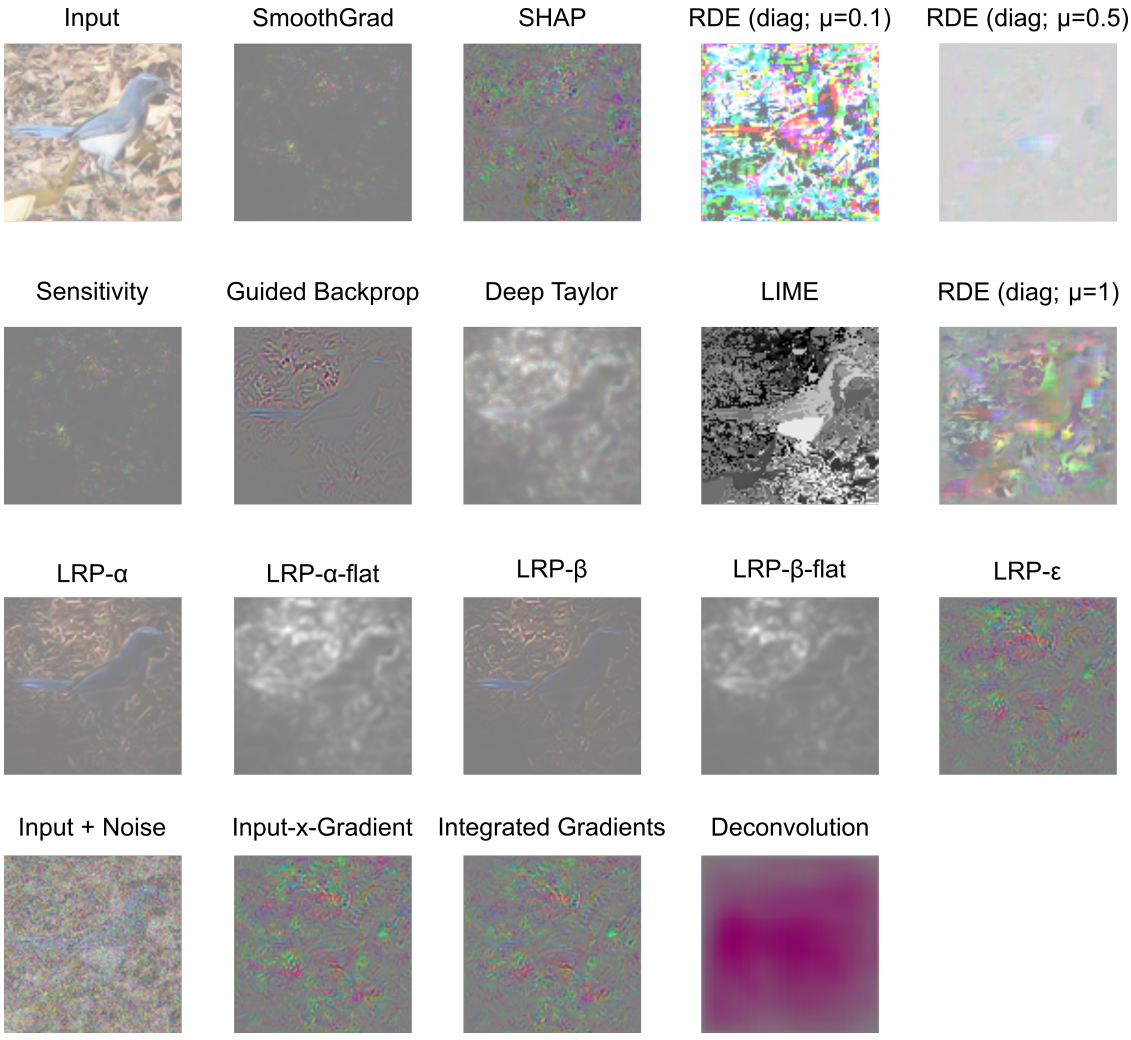

Abbildung 10: Relevanzkarten für ein Beispielbild des STL-10 Datensatzes (Vogel (Bird)) von mehreren Methoden. Ein positiver Effekt ist in den Farbkarten rot und ein negativer blau dargestellt.

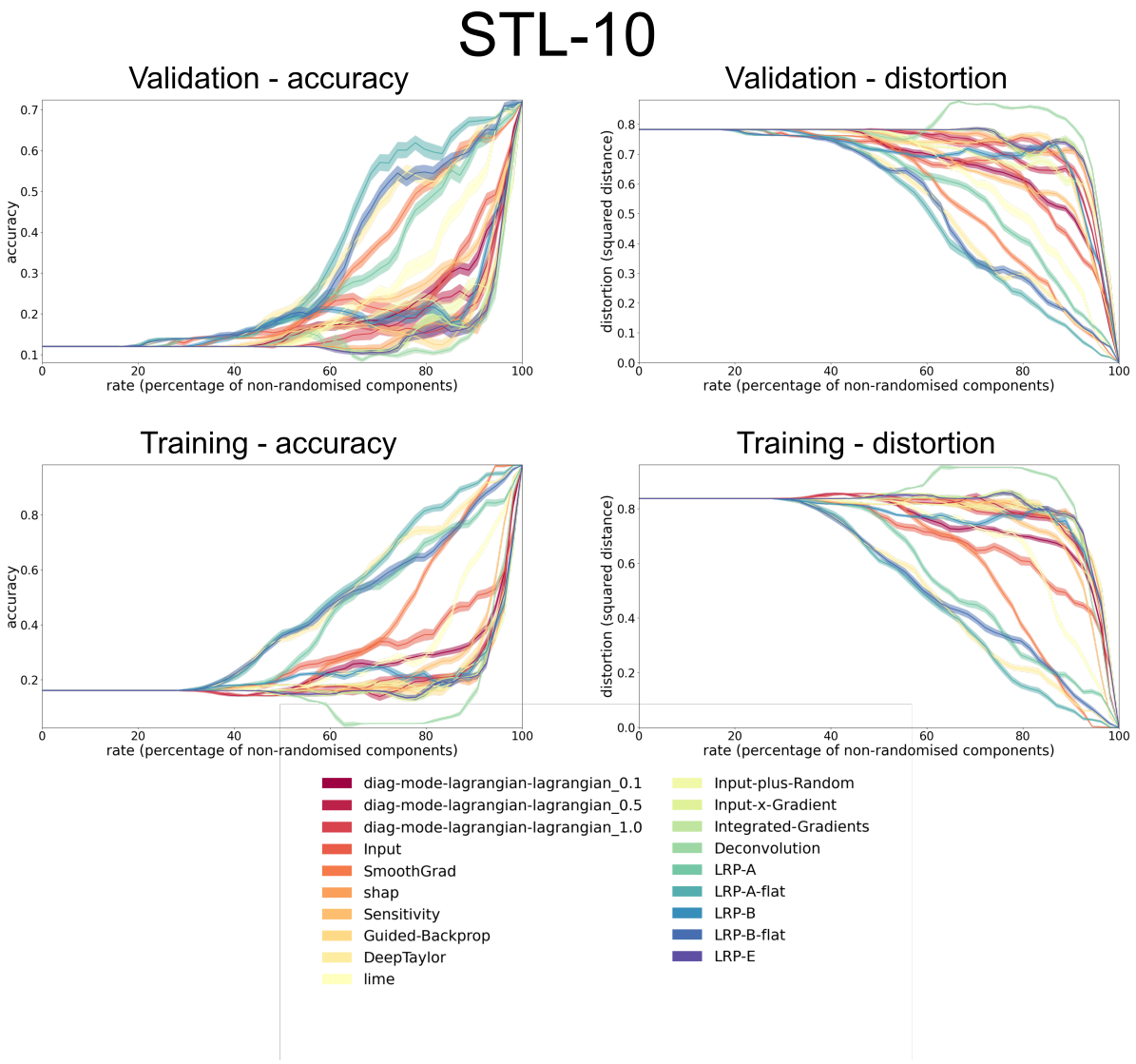

Abbildung 11: Ergebnisse des Relevanzordnung-basierten Tests (approximate rate-distortion Funktion) von mehreren Methoden für den STL-10 Datensatz mit Angabe der Accuracy und der Distortion für den Test-Datensatz und den Trainings-Datensatz.

7.4 Radiologische Visierungen

Als letztes Beispiel werden die Visitationen aus der radiologischen Klinik des Universitätsspitals Basel analysiert. Bevor es zu einer radiologischen Untersuchung mittels Computertomographie oder Magnetresonsnztomographie kommt, wird ein Patient / Patientin von einem Arzt / Ärztin klinisch untersucht. Mit Hilfe dieser klinischen Untersuchung und weiteren Informationen wird eine Verdachtsdiagnose erarbeitet. Um diese zu bestätigen, zu widerlegen oder sie zu spezifizieren wird - sofern dies die beste Möglichkeit ist - eine radiologische Untersuchung durchgeführt. Hierzu werden der Radiologie diese Informationen übermittelt, sodass basierend auf der klinischen Verdachtsdiagnose und der damit einhergehenden klinischen Fragestellung eine entsprechende Untersuchung durchgeführt wird. Als Beispiel sei ein Patient (männlich) vorstellig mit folgender Symptomatik und Angaben:

- 50 jähriger Mann
- Ödemen am Unterschenkel rechts mit Rötung und Spannungsgefühl (Klinik einer tiefen Beinvenenthrombose)
- Herzfrequenz 110 / min
- Knie-OP vor einer Woche, ansonsten keine Vorerkrankungen
- Hustet Blut (Hämoptyse)

Dieser Patient hat die klinischen Zeichen einer Lungenarterienembolie. Um diese zu bestätigen oder auszuschließen wird eine CT-Untersuchung erbeten mit einer entsprechenden Fragestellung nach einer Lungenarterienembolie und Angabe der klinischen Untersuchungsergebnisse. Diese Anfrage wird von einem Arzt / Ärztin in der Radiologie bearbeitet, indem zunächst das CT-Untersuchungsprotokoll festgelegt wird. In diesem Fall empfiehlt sich ein CT der Lunge in pulmonalarterieller Phase, zur Darstellung der Lungenarterien und dementsprechend der möglichen Thromben. Diesen Prozess bezeichnet man als Visierung. Das Ziel ist eine automatische Visierung von kardiothorakalen CT-Untersuchungen basierend auf der klinischen Fragestellung und der zuweisenden Abteilung. Zudem sollen die relevanten Begriffe für die Entscheidung des Neuronales Netzes markiert werden.
Hierzu wurden 66173 CT-Untersuchungen der kardiothorakalen Abteilung der Radiologie des Universitätsspitals Basel zwischen dem Januar 2010 und dem Oktober 2021 benutzt. Als Groud-truth diente die Visitation aus der klinischen Routine, wobei eins von 30 unterschiedlichen Protokollen angegeben wurde. Bisher erfolgte die Visierung durch einen Arzt / Ärztin der Radiologie. Bei ca. 66000 Untersuchungen über ca. 10 Jahre, wurden allein in der kardiothorakalen Abteilung der Radiologie des Universitätsspitals Basel ca. 6600 Visitationen pro Jahr und somit ca. 20 Visitationen pro Tag durchgeführt. Bei komplexeren Fragestellungen ist nicht zu erwarten, dass allein basierend auf der klinischen Fragestellung und der zuweisenden Abteilung eine Visierung erfolgen kann, da hier ebenfalls die zuständigen Ärzte / Ärztinnen das Untersuchungsprotokoll diskutieren. Jedoch ist bei einfachen, häufig vorkommenden Untersuchungen anzunehmen, dass eine automatische Visierung erfolgen und ggf. den klinischen Ablauf beschleunigen könnte.
Als erstes wurden die 1000 meist benutzten Wörter aus den klinischen Fragestellung extrahiert. Dies wurde ebenfalls für die 100 meist benutzten Wörter in den zuweisenden Abteilungen durchgeführt. Diese Daten wurden fusioniert und als Eingangssignal für ein neuronales Netzwerk benutzt, welches aus drei vollständig verknüpften Schichten mit je 64 Neuronen verbunden mit ReLU-Funktionen und einer weiteren vollständig verknüpften Schicht mit 30 Neuronen und einer Softmax-Funktion besteht. Die Gewichte der verborgenen (hidden) Schichten unterlagen einer $L2$-Regularisierung. Dieses Netzwerk wurde

auf 98.5% der Daten trainiert und auf den verbleibenden 1.5%($n = 1000$) validiert.
Das Netzwerk wies am Ende eine Trainings-Accuracy von 69% und eine Validation-Accuracy von 73% auf. Die Ergebnisse der automatischen Visitation durch das Netzwerk im Validationsdatensatz beliefen sich wie folgt:

- Native CT-Thorax (ohne Kontrastmittel) ($n = 278$) : 92% richtig
- CT-Thorax in pulmonalarterieller Phase (mit Kontrastmittel) ($n = 333$) : 91% richtig
- CT-Thorax in arterieller Phase (mit Kontrastmittel) ($n = 68$) : 50% richtig
- CT-Thorax-Abdomen in venöser Phase (mit Kontrastmittel) ($n = 30$) : 70% richtig
- CT-Angiographie Thoarx-Abdomen (mit Kontrastmittel) ($n = 51$) : 84% richtig

Mittels der RDE-Methode konnten Wörter wie „Angiographie“, „Ausschluss“ und „Lungenarterienembolie“ als für das Netzwerk relevante Wörter identifiziert werden. Für die Zuweisung eines pulmonaltarteriellen CT-Protokolls war beispielsweise auch das Fehlen der Wörter „Tumor“, „Dissektion“ und „Husten“ relevant. Diese Ergebnisse legen nahe, dass in gewissen Maßen die relevanten Wörter für die Klassifikation genutzt wurden. Im Folgenden werden 2 Beispiele von automatischen Visierungen gezeigt:

Beispiel 1:

Klinische Fragestellung: $CT - Thorax$ nativ zur $pneumologischen$ Bewertung einer $nichtobstruktiven$ Lungenerkrankung bei einem $Raucher$
Zuweisende Abteilung: Pneumologie $Lungenfunktion$

Beispiel 2:

Klinische Fragestellung: Frage nach einer $Lungenarterienembolie$ oder einer $Infektion$. Danke
Zuweisende Abteilung: Notaufnahme

Diese Daten zeigen, dass für die beiden häufigsten Untersuchungen in der kardiothorakalen Abteilung, dem nativen CT-Thorax und dem CT-Thorax in pulmonalarterieller Phase, die automatisch Visierung mit über 92% funktioniert und die relevanten Wörter markiert werden können. Um einen klinischen Einsatz zu erlauben, müsste die Klassifikationsgenauigkeit weiter verbessert und eine prospektive Evaluation durchgeführt werden. In Abbildung 12 wurden die Ergebnisse des Relevanzordnung-basierten Tests von weiteren Methoden mit der RDE-Methoden verglichen. Auch wenn die Accuracy dazu beiträgt, dass wahrscheinlich teilweise falsche Klassifikationen auftreten, so ist dennoch ersichtlich, dass andere Methoden nicht besser oder schlechter sind, sodass ein Vergleich mit vielen anderen Methoden sinnvoll erscheint.

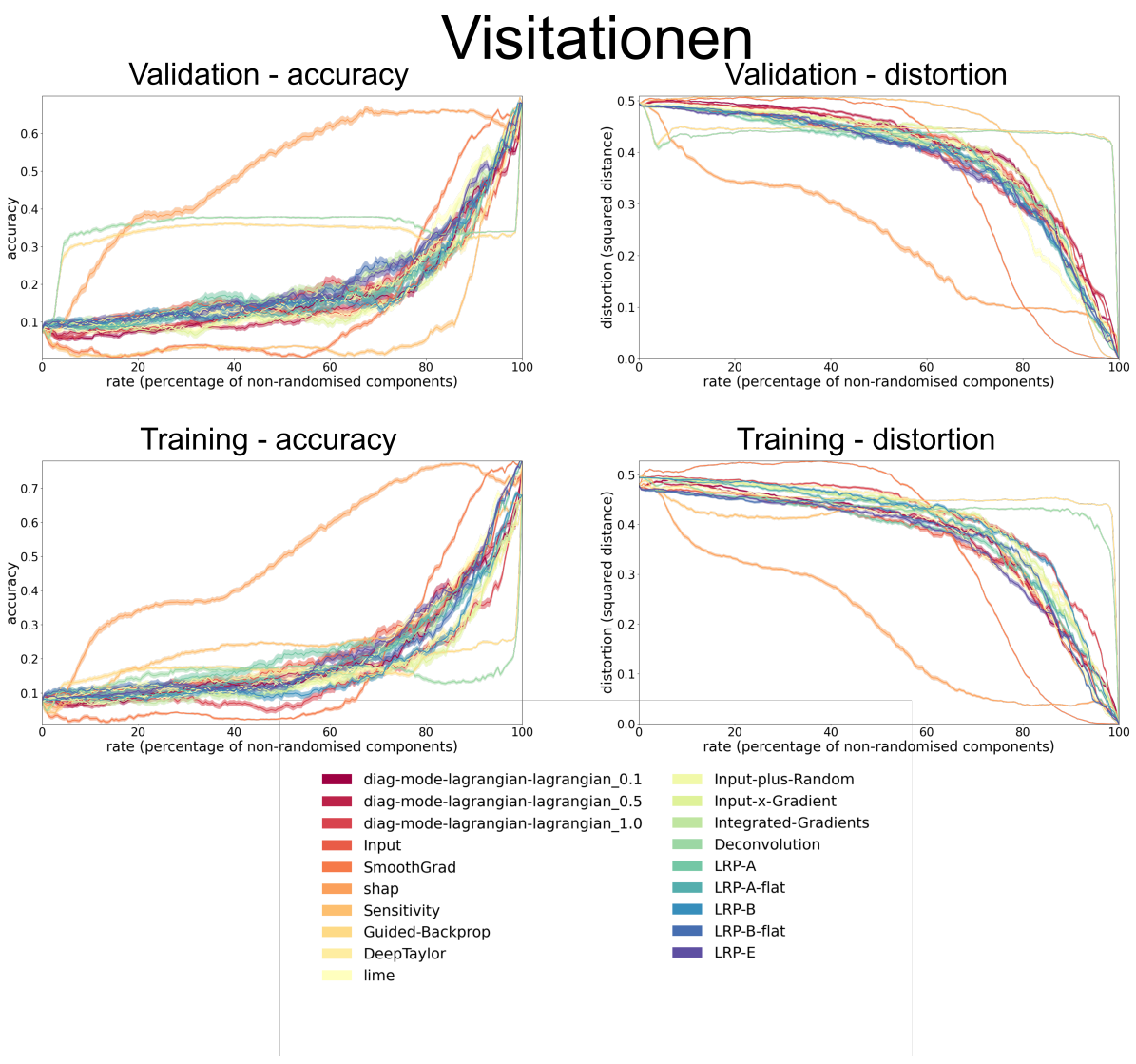

Abbildung 12: Ergebnisse des Relevanzordnung-basierten Tests (approximate rate-distortion Funktion) von mehreren Methoden für den Visierungsdatensatz mit Angabe der Accuracy und der Distortion für den Test-Datensatz und den Trainings-Datensatz.

8 Zusammenfassung und Ausblick

In dieser Arbeit wurde eine von Macdonald et al. [6] vorgeschlagene Methode namens „Rate-Distortion System zur Erklärung von Entscheidungen von neuronalen Netzwerken“ untersucht. Es wurde bewiesen, dass kein effizienter Approximations-Algorithmus für das Min-Relevant-Input Problem existiert, es sei denn $P = NP$. Neben der Rate-Distortion Methode wurden weitere Methoden vorgestellt und mittels mehrerer Anwendungsbeispiele verglichen.

Die von Macdonald et al. beschriebenen Ergebnisse konnten in dieser Arbeit nicht vollständig reproduziert werden. Für die Anwendungsfälle in dieser Arbeit zeigte die Rate-Distortion Methode keine deutliche Überlegenheit im Vergleich zu den anderen vorgestellten Methoden. Insgesamt konnte jedoch gezeigt werden, dass durch die Nutzung von multiplen Relevanzkarten, die für einen Klassifikator relevanten Anteile des Eingangssignals identifiziert werden können. Die Unterschiede in den Ergebnissen der verschiedenen Relevanzkarten legen nahe bei der Analyse eines Klassifikators mehrere Methoden zu verwenden. Des Weiteren setzten einige Methoden bestimmte Eigenschaften des Klassifikators voraus, sodass diese nicht in jedem Fall angewendet werden können. Durch die Nutzung von Heuristiken und der angenommenen Dichtefilterung ist die Rate-Distortion Methode ebenfalls auf neuronale Netzwerke mit vollständig verknüpften Schichten oder Convolutional-Schichten beschränkt, da nur diese in dem bisherigen Python-Paket enthalten sind.

Sofern die RDE-Methode auf weitere Netzwerkarchitekturen angewendet werden soll, kann das ADF-Paket entsprechend weiterentwickelt werden. Dies würde sich insbesondere bei Netzwerken zur Texterkennung und Klassifikation wie indem Anwendungsfall der CT-Thorax-Visierungen in der Radiologie empfehlen. Die bisherige Arbeit zu dem Anwendungsfall der CT-Thorax-Visierungen wurde auf dem „European Congress of Radiology“ 2023 im Rahmen einer Präsentation vorgestellt. Es ist geplant diese Methodik auszubauen auf weitere Visierungsprotokolle (z.B. CT-Abdomen, CT-Neurologie) und auf Visierungsprotokolle für Magnetresonanzuntersuchungen. Ein entsprechender Ausbau und eine Verbesserung des Klassifikators würde es erlauben, die Methode in der Radiologie anzuwenden und den Arbeitsfluss erleichtern.

9 Literatur

1. Gotra A, Sivakumaran L, Chartrand G, Vu KN, Vandenbroucke-Menu F, Kauffmann C, Kadoury S, Gallix B, Guise JA de und Tang A. Liver segmentation: indications, techniques and future directions. eng. Insights into Imaging 2017 Aug; 8:377–92. DOI: 10.1007/s13244-017-0558-1
2. Sexauer R, Yang S, Weikert T, Poletti J, Bremerich J, Roth JA, Sauter AW und Anastasopoulos C. Automated Detection, Segmentation, and Classification of Pleural Effusion From Computed Tomography Scans Using Machine Learning. eng. Investigative Radiology 2022 Aug; 57:552–9. DOI: 10.1097/RLI.0000000000000869
3. Intracranial Hemorrhage (ICH), Aidoc. en. Available from: //www.AIforRadiology.com/ [Accessed on: 2023 Jan 04]
4. Zia A, Fletcher C, Bigwood S, Ratnakanthan P, Seah J, Lee R, Kavnoudias H und Law M. Retrospective analysis and prospective validation of an AI-based software for intracranial haemorrhage detection at a high-volume trauma centre. en. Scientific Reports 2022 Nov; 12. Number: 1 Publisher: Nature Publishing Group:19885. DOI: 10.1038/s41598-022-24504-y. Available from: https://www.nature.com/articles/s41598-022-24504-y [Accessed on: 2023 Jan 04]
5. Incidental Pulmonary embolism (iPE), Aidoc. en. Available from: //www.AIforRadiology.com/ [Accessed on: 2023 Jan 04]
6. Macdonald J, Wäldchen S, Hauch S und Kutyniok G. A Rate-Distortion Framework for Explaining Neural Network Decisions. en. Number: arXiv:1905.11092 arXiv:1905.11092 [cs, math, stat]. 2019 May. Available from: http://arxiv.org/abs/1905.11092 [Accessed on: 2022 Oct 08]
7. Sousa I Palatnik de, Vellasco MMBR und Silva E Costa da. Explainable Artificial Intelligence for Bias Detection in COVID CT-Scan Classifiers. Sensors (Basel, Switzerland) 2021 Aug; 21:5657. DOI: 10.3390/s21165657. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8402377/ [Accessed on: 2023 Feb 13]
8. Yang X, He X, Zhao J, Zhang Y, Zhang S und Xie P. COVID-CT-Dataset: A CT Scan Dataset about COVID-19. arXiv:2003.13865 [cs, eess, stat]. 2020 Jun. DOI: 10.48550/arXiv.2003.13865. Available from: http://arxiv.org/abs/2003.13865 [Accessed on: 2023 Feb 14]
9. Rogers H. Theory of Recursive Functions and Effective Computability. Englisch. Fifth Printing Edition. Cambridge, Mass: The MIT Press, 1987 Apr
10. Algorithmus. de. Page Version ID: 229657366. 2023 Jan. Available from: https://de.wikipedia.org/w/index.php?title=Algorithmus&oldid=229657366#cite_note-1 [Accessed on: 2023 Feb 14]
11. Turingmaschine. de. Page Version ID: 226748655. 2022 Oct. Available from: https://de.wikipedia.org/w/index.php?title=Turingmaschine&oldid=226748655 [Accessed on: 2022 Oct 08]
12. Schöning U. Theoretische Informatik - kurzgefasst. de. Spektrum Akademischer Verlag, 1997
13. P-NP-Problem. de. Page Version ID: 230259424. 2023 Jan. Available from: https://de.wikipedia.org/w/index.php?title=P-NP-Problem&oldid=230259424 [Accessed on: 2023 Feb 14]
14. Probabilistische Turingmaschine. de. Page Version ID: 213472329. 2021 Jul. Available from: https://de.wikipedia.org/w/index.php?title=Probabilistische_Turingmaschine&oldid=213472329 [Accessed on: 2023 Jan 03]

15. PP (complexity). en. Page Version ID: 1110751365. 2022 Sep. Available from: https://en.wikipedia.org/w/index.php?title=PP_(complexity)&oldid=1110751365 [Accessed on: 2023 Jan 03]

16. Probabilistische Polynomialzeit. de. Page Version ID: 226079352. 2022 Sep. Available from: https://de.wikipedia.org/w/index.php?title=Probabilistische_Polynomialzeit&oldid=226079352 [Accessed on: 2023 Jan 03]

17. Torán J. Complexity classes defined by counting quantifiers. en. Journal of the ACM 1991 Jul; 38:752–73. DOI: 10.1145/116825.116858. Available from: https://dl.acm.org/doi/10.1145/116825.116858 [Accessed on: 2023 Jan 03]

18. Littman ML, Goldsmith J und Mundhenk M. The Computational Complexity of Probabilistic Planning. en. Journal of Artificial Intelligence Research 1998 Aug; 9:1–36. DOI: 10.1613/jair.505. Available from: https://jair.org/index.php/jair/article/view/10208 [Accessed on: 2023 Jan 03]

19. Littman M, Majercik S und Pitassi T. Stochastic Boolean Satisfiability. Journal of Automated Reasoning 1999 Dec; 27. DOI: 10.3233/978-1-58603-929-5-887

20. Wäldchen S, Macdonald J, Hauch S und Kutyniok G. The Computational Complexity of Understanding Network Decisions. arXiv:1905.09163 [cs]. 2019 Jun. DOI: 10.48550/arXiv.1905.09163. Available from: http://arxiv.org/abs/1905.09163 [Accessed on: 2023 Feb 15]

21. Gast J und Roth S. Lightweight Probabilistic Deep Networks. en. 2018 May. DOI: 10.48550/arXiv.1805.11327. Available from: https://arxiv.org/abs/1805.11327v1 [Accessed on: 2023 Feb 15]

22. Petsiuk V, Das A und Saenko K. RISE: Randomized Input Sampling for Explanation of Black-box Models. arXiv:1806.07421 [cs]. 2018 Sep. DOI: 10.48550/arXiv.1806.07421. Available from: http://arxiv.org/abs/1806.07421 [Accessed on: 2023 Feb 14]

23. Simonyan K, Vedaldi A und Zisserman A. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. arXiv:1312.6034 [cs]. 2014 Apr. DOI: 10.48550/arXiv.1312.6034. Available from: http://arxiv.org/abs/1312.6034 [Accessed on: 2023 Jan 23]

24. Smilkov D, Thorat N, Kim B, Viégas F und Wattenberg M. SmoothGrad: removing noise by adding noise. arXiv:1706.03825 [cs, stat]. 2017 Jun. DOI: 10.48550/arXiv.1706.03825. Available from: http://arxiv.org/abs/1706.03825 [Accessed on: 2023 Jan 23]

25. Ancona M, Ceolini E, Öztireli C und Gross M. Towards better understanding of gradient-based attribution methods for Deep Neural Networks. arXiv:1711.06104 [cs, stat]. 2018 Mar. DOI: 10.48550/arXiv.1711.06104. Available from: http://arxiv.org/abs/1711.06104 [Accessed on: 2023 Jan 16]

26. Ancona M, Oztireli C und Gross M. Explaining Deep Neural Networks with a Polynomial Time Algorithm for Shapley Value Approximation. en. *Proceedings of the 36th International Conference on Machine Learning*. ISSN: 2640-3498. PMLR, 2019 May :272–81. Available from: https://proceedings.mlr.press/v97/ancona19a.html [Accessed on: 2023 Jan 06]

27. Sundararajan M, Taly A und Yan Q. Axiomatic Attribution for Deep Networks. en. *Proceedings of the 34th International Conference on Machine Learning*. ISSN: 2640-3498. PMLR, 2017 Jul :3319–28. Available from: https://proceedings.mlr.press/v70/sundararajan17a.html [Accessed on: 2023 Jan 16]

28. Montavon G, Lapuschkin S, Binder A, Samek W und Müller KR. Explaining nonlinear classification decisions with deep Taylor decomposition. en. Pattern Recognition 2017 May; 65:211–22. DOI: 10.1016/j.patcog.2016.11.008. Available from: https://www.sciencedirect.com/science/article/pii/S0031320316303582 [Accessed on: 2023 Jan 05]

29. Szegedy C, Zaremba W, Sutskever I, Bruna J, Erhan D, Goodfellow I und Fergus R. Intriguing properties of neural networks. arXiv:1312.6199 [cs]. 2014 Feb. DOI: 10.48550/arXiv.1312.6199. Available from: http://arxiv.org/abs/1312.6199 [Accessed on: 2023 Jan 06]

30. Pizarroso J, Portela J und Muñoz A. NeuralSens: Sensitivity Analysis of Neural Networks. en. arXiv:2002.11423 [cs, stat]. 2021 Feb. Available from: http://arxiv.org/abs/2002.11423 [Accessed on: 2023 Jan 06]

31. Bach S, Binder A, Montavon G, Klauschen F, Müller KR und Samek W. On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation. en. PLOS ONE 2015 Jul; 10. Publisher: Public Library of Science:e0130140. DOI: 10.1371/journal.pone.0130140. Available from: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0130140 [Accessed on: 2023 Jan 04]

32. Zeiler MD, Krishnan D, Taylor GW und Fergus R. Deconvolutional networks. en. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. San Francisco, CA, USA: IEEE, 2010 Jun :2528–35. DOI: 10.1109/CVPR.2010.5539957. Available from: http://ieeexplore.ieee.org/document/5539957/ [Accessed on: 2023 Feb 08]

33. Zeiler MD, Taylor GW und Fergus R. Adaptive deconvolutional networks for mid and high level feature learning. en. *2011 International Conference on Computer Vision*. Barcelona, Spain: IEEE, 2011 Nov :2018–25. DOI: 10.1109/ICCV.2011.6126474. Available from: http://ieeexplore.ieee.org/document/6126474/ [Accessed on: 2023 Feb 08]

34. Zeiler MD und Fergus R. Visualizing and Understanding Convolutional Networks. en. *Computer Vision – ECCV 2014*. Hrsg. von Fleet D, Pajdla T, Schiele B und Tuytelaars T. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2014 :818–33. DOI: 10.1007/978-3-319-10590-1_53

35. Springenberg JT, Dosovitskiy A, Brox T und Riedmiller M. Striving for Simplicity: The All Convolutional Net. arXiv:1412.6806 [cs]. 2015 Apr. DOI: 10.48550/arXiv.1412.6806. Available from: http://arxiv.org/abs/1412.6806 [Accessed on: 2023 Feb 10]

36. Selvaraju RR, Das A, Vedantam R, Cogswell M, Parikh D und Batra D. Grad-CAM: Why did you say that? arXiv:1611.07450 [cs, stat]. 2017 Jan. DOI: 10.48550/arXiv.1611.07450. Available from: http://arxiv.org/abs/1611.07450 [Accessed on: 2023 Feb 10]

37. Fuhrman JD, Gorre N, Hu Q, Li H, El Naqa I und Giger ML. A review of explainable and interpretable AI with applications in COVID-19 imaging. eng. Medical Physics 2022 Jan; 49:1–14. DOI: 10.1002/mp.15359

38. Shapley LS. A Value for N-Person Games. en. Techn. Ber. RAND Corporation, 1952 Mar. Available from: https://www.rand.org/pubs/papers/P295.html [Accessed on: 2023 Jan 16]

39. Shapley value. en. Page Version ID: 1128689059. 2022 Dec. Available from: https://en.wikipedia.org/w/index.php?title=Shapley_value&oldid=1128689059 [Accessed on: 2023 Jan 06]

40. Sundararajan M und Najmi A. The Many Shapley Values for Model Explanation. en. *Proceedings of the 37th International Conference on Machine Learning*. ISSN: 2640-3498. PMLR, 2020 Nov :9269–78. Available from: https://proceedings.mlr.press/v119/sundararajan20b.html [Accessed on: 2023 Jan 16]

41. Ribeiro MT, Singh S und Guestrin C. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. arXiv:1602.04938 [cs, stat]. 2016 Aug. DOI: 10.48550/arXiv.1602.04938. Available from: http://arxiv.org/abs/1602.04938 [Accessed on: 2023 Jan 16]

42. Sousa IPd, Vellasco MMBR und Silva ECd. Local Interpretable Model-Agnostic Explanations for Classification of Lymph Node Metastases. en. Sensors (Basel, Switzerland) 2019 Jul; 19. Publisher: Multidisciplinary Digital Publishing Institute (MDPI). DOI: 10.3390/s19132969. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6651753/ [Accessed on: 2023 Feb 13]

43. LeCun Y, Bottou L, Bengio Y und Ha P. Gradient-Based Learning Applied to Document Recognition. en. 1998

44. jmaces. Rate-Distortion Explanations (RDE). original-date: 2021-01-29T08:55:35Z. 2022 Jul. Available from: https://github.com/jmaces/rde [Accessed on: 2023 Feb 15]

45. Kohlbrenner M, Bauer A, Nakajima S, Binder A, Samek W und Lapuschkin S. Towards Best Practice in Explaining Neural Network Decisions with LRP. arXiv:1910.09840 [cs, stat]. 2020 Jul. DOI: 10.48550/arXiv.1910.09840. Available from: http://arxiv.org/abs/1910.09840 [Accessed on: 2023 Jan 05]

46. Alber M, Lapuschkin S, Seegerer P, Hägele M, Schütt KT, Montavon G, Samek W, Müller KR, Dähne S und Kindermans PJ. iNNvestigate neural networks! arXiv:1808.04260 [cs, stat]. 2018 Aug. DOI: 10.48550/arXiv.1808.04260. Available from: http://arxiv.org/abs/1808.04260 [Accessed on: 2023 Jan 05]

47. Alber M. iNNvestigate neural networks! original-date: 2017-12-13T18:11:20Z. 2022 Oct. Available from: https://github.com/albermax/innvestigate [Accessed on: 2022 Oct 11]

48. Lundberg S. slundberg/shap. original-date: 2016-11-22T19:17:08Z. 2023 Feb. Available from: https://github.com/slundberg/shap [Accessed on: 2023 Feb 15]

49. Ribeiro MTC. lime. original-date: 2016-03-15T22:18:10Z. 2023 Feb. Available from: https://github.com/marcotcr/lime [Accessed on: 2023 Feb 15]

Abbildungsverzeichnis

Plagiatserklärung der / des Studierenden

Hiermit versichere ich, dass die vorliegende Arbeit über „Rate-Distortion System zur Erklärung von Entscheidungen von neuronalen Netzwerken“ selbstständig verfasst worden ist, dass keine anderen Quellen und Hilfsmittel als die angegebenen benutzt worden sind und dass die Stellen der Arbeit, die anderen Werken – auch elektronischen Medien – dem Wortlaut oder Sinn nach entnommen wurden, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht worden sind.

Münster, den 9.3.2023 _____
(Datum, Unterschrift)

Ich erkläre mich mit einem Abgleich der Arbeit mit anderen Texten zwecks Auffindung von Übereinstimmungen sowie mit einer zu diesem Zweck vorzunehmenden Speicherung der Arbeit in eine Datenbank einverstanden.

Münster, den 9.3.2023 _____
(Datum, Unterschrift)