Entwicklung einer Softwareumgebung zum Interpretieren von neuronalen Netzen

Juri Zach
Hochschule für Angewandte Wissenschaften Hamburg
Berliner Tor 5, 20099 Hamburg
Deutschland
juri.zach@haw-hamburg.de

***Zusammenfassung*—Um den praktischen Einsatz der Interpretation von neuronalen Netzen voranzutreiben, beschreibt dieser Artikel die Entwicklung einer Softwareumgebung, welche die wissenschaftliche Untersuchung verschiedener Interpretationsmethoden ermöglicht.**

Hierbei werden alle Anforderungen unterstützt, welche zum Evaluieren der Nützlichkeit einer Interpretationsmethode, gefordert werden (siehe Zach [1]).

Die Softwareumgebung beinhaltet einige der aktuellsten Interpretationsmethoden, welche auf eine Vielzahl von neuronalen Netzen angewendet werden können. Um datengetriebene Modellanalysen zu unterstützen, werden spezielle Datensätze bereitgestellt. Auch das trainieren und evaluieren von neuronalen Netzen wird unterstützt, wodurch der Zusammenhang zwischen gelernten Wissen und Möglichkeiten der Interpretation, untersucht werden kann.

Erste Resultate zeigen bereits gute Ergebnisse, aber auch, dass die Interpretation von neuronalen Netzen stark von dem Netz selbst abhängt und besonders große neuronale Netze eine Herausforderung darstellen.

***Index Terms*—Erklärbare künstliche Intelligenz, neuronale Netze, Interpretierbarkeit, Tensorflow, Keras, Lucid**

I. Einleitung

Die Forschung im Bereich der künstlichen Intelligenz hat in den letzten Jahren große Fortschritte gemacht und findet sich zunehmend im alltäglichen Leben wieder. Bekannte Beispiele sind Facebook-Newsfeeds, Chatbots, Spracherkennung, digitales Marketing oder Einparkhilfen im Auto. Doch auch sicherheitskritische Anwendungen wie beispielsweise intelligente Roboter, medizinische Diagnosesysteme oder autonom fahrende Autos werden stark vorangetrieben. Für viele dieser Aufgaben werden tiefe neuronale Netze eingesetzt. Diese sind in der Lage, die hochdimensionalen Funktionen zu approximieren, mit denen komplexe Aufgaben wie Bild- und Spracherkennung, oder das Steuern von komplexen Roboter Aktoren, approximiert werden können.
Allerdings hat diese Technologie gerade im Bereich der Sicherheit und Testbarkeit starke Schwachstellen. Aufgrund der Komplexität, ist es kaum möglich nachzuvollziehen was ein neuronales Netz gelernt hat und wie es seine Entscheidungen trifft.
Der Forschungsbereich der erklärbaren künstlichen Intelligenz befasst sich unter anderem mit der Interpretierbarkeit von neuronalen Netzen. Auch hier wurden in den letzten Jahren große Fortschritte erzielt. Doch im Gegensatz zu neuronalen Netzen, bleibt dieser Forschungsbereich theoretisch und findet kaum praktische Anwendung.

Um den Einsatz der Interpretation von neuronalen Netzen voranzutreiben, stellt dieser Artikel eine Interpretationsumgebung für neuronale Netze vor, in welcher der praktische Nutzen von Interpretationsmethoden erforscht werden kann. Diese Software-Infrastruktur wird im folgenden Text als Interpretationsumgebung bezeichnet.[1]

Im Abschnitt II werden verschiedene Interpretationsmethoden vorgestellt, sowie Forschungen die sich mit den theoretischen Anforderungen an Interpretationsmethoden beschäftigen. Im Abschnitt III wird eine Anforderungsanalyse für die Interpretationsumgebung aufgestellt. Abschnitt IV beschreibt die Implementierung der Interpretationsumgebung. Im Abschnitt V wir detailliert auf die verschiedenen Interpretationsmethoden eingegangen, welche in der Interpretationsumgebung implementiert sind. In den Abschnitten VI und VII werden die Ergebnisse der Arbeit präsentiert und auf weiterführende Forschungen verwiesen.

II. Ähnliche Arbeiten

Dieser Artikel baut auf verschiedenen wissenschaftlichen Arbeiten auf, welche sich sowohl mit theoretischen Aspekten als auch mit konkreten Implementierungen und Untersuchungen von Interpretationsmethoden beschäftigen.
Die wissenschaftlichen Arbeiten der Autoren Doshi-Velez und Kim [2] und Lipton [3] befassen sich mit den theoretischen Aspekten der Interpretierbarkeit von maschinellen Lernmethoden und beschreiben unter anderen auch, welche Anforderungen diese erfüllen sollten.
Nach Lipton [3] entsteht die Notwendigkeit der Interpretierbarkeit, sobald ein maschineller Lernalgorithmus auf Werte optimiert werden soll, welche sich nicht als mathematische Funktion umsetzen und optimieren lassen.
Im folgenden sind Beispiele aufgelistet, bei denen eine Notwendigkeit der Interpretierbarkeit besteht:

- **Sicherheit:** Für komplexe maschinelle Lernalgorithmen ist es nicht praktikabel alle möglichen Anwendungs-

[1] Diese Software befindet sich zurzeit auf dem GitLab Server der HAW und kann auf Anfrage eingesehen werden.

szenarien zu testen. Interpretationsmethoden sollen es ermöglichen Sicherheitslücken gezielt zu identifizieren.

- **Wissenschaftlicher Erkenntnisgewinn:** Obwohl maschinelle Lernverfahren nur approximative Lösungen liefern, besteht die Hoffnung aus ihrem gelernten Wissen und Entscheidungen, Erkenntnisse über die zugrunde liegenden Probleme zu gewinnen. Hierfür ist jedoch ein genauerer Einblick in ihre Funktionsweise vonnöten.
- **Ethik:** Sowohl Politiker und Journalisten als auch Wissenschaftler fordern, dass die von künstlicher Intelligenz getroffenen Entscheidungen ethischen Standards entsprechen [4].
- **Vertrauen:** Um das Vertrauen des Benutzers herzustellen, ist es vorteilhaft die Vorhersage eines maschinellen Lernalgorithmus durch Begründungen zu erweitern. Hierdurch können die Stärken und Schwächen besser abgeschätzt werden.

Zusätzlich zu den theoretischen Überlegungen zu maschinellen Lernmethoden, findet sich in der wissenschaftlichen Literatur eine Vielzahl von praktischen Umsetzungen zur Interpretation von neuronalen Netzen. Die meisten wissenschaftlichen Arbeiten fokussieren sich auf Feature Visualisierung [5] [6] [7] [8] und Attribuierung [8] [9] [10] [11]. Zudem forscht die Mensch-Computer-Interaktions-Gemeinde an vielversprechenden Schnittstellen [12] [13], um Interpretationsergebnisse intuitiv darzustellen.

In den aktuellsten Forschungen der letzten Jahre, werden verschiedene Interpretationsmethoden kombiniert und durch wirkungsvolle Schnittstellen erweitert, wodurch die Aussagekraft der Interpretationen deutlich verbessert wird.
Hierzu gehört unter anderen die Arbeit von Olah et al. [14], die verschiedene Varianten der Feature Visualisierung vorgestellt und zeigt, wie diese mit Attribution-Methoden und wirkungsvollen Interfaces kombiniert werden können. Die Arbeit von Carter et al. [15] zeigt, wie die von neuronalen Netzen gelernten Konzepte als Aktivierungsvektor definiert und visualisiert werden. In der Arbeit von Kim et al. [16] werden von Menschen definierte Konzeptaktivierungsvektoren dazu genutzt, die Entscheidungen von neuronalen Netzen zu erklären.

Obwohl diese Entwicklungen sehr vielversprechend wirken, gibt es noch keine wissenschaftliche Arbeit darüber, die evaluiert, wie gut sich heutige Interpretationsmethoden dazu eignen die theoretischen Anforderungen zu erfüllen, welche von Wissenschaftlern wie Doshi-Velez, Kim oder Lipton gefordert wurden.
Die Arbeit von Zach [1] beschreibt, wie Experimente aufgebaut werden sollten um die Nützlichkeit von Interpretationsmethoden für neuronale Netze nachzuweisen.
In den folgenden Abschnitten werden die Anforderungen und die Implementierung der Interpretationsumgebung vorgestellt, mit der eine effiziente Durchführung dieser Experimente ermöglicht werden soll.

III. Anforderungen an eine Interpretationsumgebung

Ein neuronales Netz wird oft als Black Box bezeichnet, obwohl die Eingangswerte, alle Gewichte und Rechenschritte bekannt sind. Für gewöhnlich sind diese allerdings zu viele und zu komplex vernetzt, als dass ein Mensch sie in ihrer Gesamtheit begreifen könnte. Interpretationsmethoden versuchen eine Teilmenge dieser Komplexität auf ein, für Menschen erfassbares Maß zu reduzieren. Hieraus erhofft sich die Wissenschaft, Erkenntnisse über das gelernte Wissen und die Kausalität der Entscheidungen eines neuronalen Netzes zu gewinnen. Zudem könnten die Stärken und Schwächen von neuronalen Netzen besser abgeschätzt, gezielt Sicherheitslücken identifiziert, auf ethische Korrektheit von Entscheidungen geprüft und eventuell neue wissenschaftliche Erkenntnisse gewonnen werden.
Allerdings ist noch nicht bewiesen, inwieweit die heutigen Interpretationsmethoden *richtige* Resultate liefern.
Die Arbeit von Zach [1] argumentiert, dass es aufgrund der menschlichen Interpretation, welche zur Interpretation eines neuronalen Netzes benötigt wird, nicht möglich ist die Richtigkeit von Interpretationsmethoden nachzuweisen. Stattdessen soll experimentell nachgewiesen werden, ob eine Interpretationsmethode nützlich für den praktischen Einsatz ist.

Im folgenden Text werden die generellen Schritte zusammengefasst (für genauere Informationen, siehe Zach [1]), mit denen ein solches Experiment durchgeführt werden kann. Anhand dieser Schritte werden verschiedene Anforderungen aufgestellt, welche die Interpretationsumgebung, zur effizienten Durchführung der Experimente erfüllen sollte.

A. Anforderungsanalyse anhand einer theoretischen Experimentdurchführung

a) Anwenden der Interpretationsmethode: Ein Experiment zur Evaluierung der Nützlichkeit einer Interpretationsmethode beginnt mit dem Anwenden der Interpretationsmethode auf ein neuronales Netz.
Zu diesem Zweck sollte die Interpretationsumgebung verschiedene neuronale Netze und Interpretationsmethoden bereitstellen.
Um die Aussagekraft eines Experimentes zu erhöhen, sollte es möglich sein die Interpretationsmethode auf mehreren neuronalen Netzen anzuwenden. Hierfür muss eine Kompatibilität zwischen den neuronalen Netzen und den Interpretationsmethoden gewährleistet sein.
Für Interpretationsmethoden die sich auf eine konkrete Entscheidung des neuronalen Netz beziehen, wird zusätzlich ein Beispieldatenpunkt benötigt. Um eine Auswahl von potentiellen Beispieldatenpunkten bereitzustellen, sollte die Interpretationsumgebung eine leichte Integration verschiedener Datensätze vorsehen. Da es häufig Sinn ergibt, Interpretationsmethoden auf besondere Datenpunkte anzuwenden (z.B. Daten auf dem das neuronale Netz besonders schlechte vorhersagen macht), werden

Evaluierungsmethoden benötigt mit denen geeignete Datenpunkte für ein bestimmtes neuronales Netz bestimmt werden können.
Für den Fall, dass selbst- oder nachtrainierte neuronale Netze verwendet werden, sollte es eine einfache Möglichkeit zum Trainieren von neuronalen Netzen geben.

b) Entwickeln einer Annahme: Im nächsten Schritt des Experiments wird das Resultat der Interpretationsmethode von einem Menschen interpretiert und eine Annahme über das neuronale Netz oder dem Trainingsdatensatz aufgestellt. Damit dies gelingt ist eine ausreichende Qualität und Aussagekraft des Resultats vorausgesetzt.

c) Erstellen von Test- und Referenzdatensätzen: Um die Annahme zu be- oder widerlegen, wird nun ein Test- und ein Referenzdatensatz erstellt. Damit dies auch bei größeren Datenmengen effizient gelingt, sollten die Datensätze das Durchsuchen nach bestimmten Kriterien ermöglichen. Die für den Test- oder Referenzdatensatz ausgewählte Datenmenge sollte anschließend separat abrufbar sein, um die Vorhersagen des neuronalen Netzes auf beiden Datensätzen zu errechnen.

d) Belegen der Nützlichkeit: Im letzten Schritt wird der Test- und Referenzdatensatz auf das neuronale Netz angewendet. Die Annahme kann be- oder widerlegt werden, indem die Resultate des neuronalen Netzes für den Test- und Referenzdatensatz verglichen werden. Um eine statistische Relevanz der Ergebnisse nachzuweisen, sollten entsprechende statistische Tests in der Interpretationsumgebung vorhanden sein.

B. Zusammenfassung der Anforderungen

In der folgenden Aufzählung sind die Anforderungen, welche im obigen Text erläutert wurden, kurz zusammengefasst.

1) Bereitstellen von verschiedenen neuronalen Netzen.
   a) Möglichkeit zum Trainieren und nachtrainieren der neuronalen Netze.
   b) Evaluieren der Performanz von neuronalen Netzen.
2) Bereitstellen von verschiedenen Datensätzen.
   a) Durchschaubarkeit von Datensätzen nach Kriterien.
   b) Abrufen von Teilmengen eines Datensatzes.
3) Bereitstellen von verschiedenen Interpretationsmethoden.
   a) Gewährleisten einer ausreichenden Qualität und Aussagekraft der Interpretationsergebnisse.
4) Bereitstellen von statistischen Tests.
5) Kompatibilität der Komponenten.
   a) Eine Interpretationsmethode kann auf jedes neuronale Netz angewendet werden.

Um die Komplexität der Aufgabe in einem, für eine studentische Arbeit angemessenen Rahmen zu halten, beschränkt sich die Interpretationsumgebung auf die Domäne des maschinellen Sehens, in der ein Großteil der Interpretierbarkeitsforschung stattfindet.

IV. Aufbau der Interpretationsumgebung

In diesem Abschnitt wird der Aufbau der Interpretationsumgebung beschrieben. Dies beinhaltet eine Beschreibung verschiedener Software Frameworks, auf denen die Interpretationsumgebung aufbaut, sowie den Entwurf einer geeigneten Software Architektur.

A. Verwendete Software

Für die Implementierung der Interpretationsumgebung wird die Programmiersprache Python verwendet. Python ist die am häufigsten benutze Sprache im Bereich des maschinellen Lernens. Sie verfügt über verschiedene Frameworks zur effizienten Softwareentwicklung im wissenschaftlichen Kontext (Scipy, Jupyter und andere) und zum Entwickeln von maschinellen Lernmethoden, insbesondere von neuronalen Netzen (Tensorflow, Keras, Pytorch und andere).

Für die Entwicklung der neuronalen Netze werden die Frameworks Tensorflow und Keras verwendet. Tensorflow bietet eine Vielzahl von Werkzeugen zum Erstellen verschiedenster maschineller Lernmethoden.
Das Keras Framework bildet eine Abstraktionsschicht auf Tensorflow, wodurch schnell und einfach, neuronale Netze erstellt und trainiert werden können, ohne auf die deutlich komplizierteren Grundfunktionen von Tensorflow zugreifen zu müssen. Zudem bietet Tensorflow verschiedene Datensätze und vortrainierte Keras-Modelle zum Download an. Insbesondere die vortrainierten neuronalen Netze sind sehr nützlich, da viele davon, auf Grund ihrer Größe und der Größe des Trainingsdatensatzes, herausragende Genauigkeiten erzielen und große Mengen an Informationen speichern.
Das Trainieren dieser oder ähnlicher Netze erfordert erheblichen Zeit- und Rechenaufwand und würde den Rahmen dieser Arbeit übersteigen. Doch gerade die Komplexität solcher neuronalen Netze macht sie, im Kontext der Interpretierbarkeit, zu einem interessanten Studienobjekt.

Die verschiedenen Interpretationsmethoden werden mithilfe des Frameworks Lucid erstellt. Lucid baut auf Tensorflow auf und bietet ein Grundgerüst und Werkzeuge zum Interpretieren von neuronalen Netzen. Es wurde bereits im Kontext mehrerer wissenschaftlichen Arbeiten [14] [15] [7] verwendet.

Um eine gute wissenschaftliche Arbeitsweise zu unterstützen und eine Interaktionsmöglichkeit zwischen der Interpretationsumgebung und dem Benutzer zu gewährleisten, werden Jupyter-Notebooks verwendet. Ein Jupyter-Notebook kombiniert ein Textdokument mit live Programmierung (in über 40 Programmiersprachen), wissenschaftlichen Formeln und Visualisierungen. Hierdurch eignet es sich ausgezeichnet um ein Experiment zur Evaluierung einer Interpretationsmethode, in einem Dokument zu beschreiben und auszuführen.

B. Software Architektur

In diesem Abschnitt wird eine Softwarearchitektur für die Interpretationsumgebung entworfen, mit dessen Hilfe wissenschaftliche Experimente zur Evaluierung der Nützlichkeit

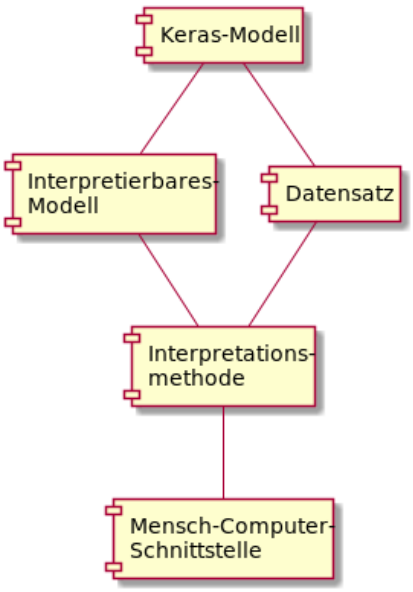


Abbildung 1: UML Komponentendiagramm der Interpretationsumgebung.

von Interpretationsmethoden, schnell und einfach durchgeführt werden können.

Um die Anforderungen aus Abschnitt III zu erfüllen, kombiniert die Interpretationsumgebung verschiedene open-source Frameworks mit selbstentwickelter Software. Um ein reibungsloses Zusammenspiel zu ermöglichen, wird die Software in mehrere Komponenten unterteilt, von denen jedes einen eigenen Zuständigkeitsbereich besitzt, um eine Teilanforderung zu realisieren. Eine fehlerfreie Kommunikation zwischen den Komponenten wird durch fest definierte Schnittstellen ermöglicht.

Die verschiedenen Komponenten werden in Abbildung 1 dargestellt und deren Zuständigkeitsbereich in den folgenden Abschnitten genauer erklärt.

Da die Interpretationsumgebung auf informatisch geschulte Anwender ausgelegt ist, beinhaltet die Softwarearchitektur kein klassisches Benutzer Interface, wie beispielsweise eine grafische Benutzeroberfläche. Der Anwender soll zum Ausführen der Experimente direkt in einem Jupyter-Notebook mit den Verschiedenen Software Komponenten interagieren, wodurch die volle Kapazität der Interpretationsumgebung ausgenutzt werden kann.

1) Keras-Modell: Eine wichtige Komponente der Softwarearchitektur bildet das Keras-Modell. Das Keras-Modell ist eine Klasse des Keras Frameworks (*keras.Model*) welche ein einfaches Interface zum Erstellen, Trainieren und Anwenden von neuronalen Netzen und anderen maschinellen Lernmethoden implementiert (siehe Anforderung 1a). Um die Performanz der Modelle zu evaluieren (siehe Anforderung 1b), bietet das Keras Framework verschiedene Metriken, welche direkt auf die Vorhersagen eines Keras-Modells angewendet werden können. Zudem gibt es im Internet eine Vielzahl von frei verfügbaren, vortrainierten Keras-Modellen mit hoher Performanz, die aufgrund ihrer Größe und Komplexität, interessante Interpretationsobjekte abgeben. Durch das Einbinden des Keras-Modells, bietet die Interpretationsumgebung einen leichten Zugang zu einer Vielzahl von neuronalen Netzen und erfüllt somit die Anforderung 1.

2) Datensatz: In der Interpretationsumgebung werden Datensätze für mehrere Anwendungen benötigt.

Zum einen werden sie zum Trainieren und Evaluieren der neuronalen Netze verwendet. Zusätzlich werden Datensätze dafür benötigt, um einzelne Datenpunkte herauszusuchen, welche für die Interpretationsmethoden oder als Test- und Referenzdatensätze, zum Überprüfen von Annahmen, verwendet werden.

Um diese Anforderungen zu erfüllen wird in der Interpretationsumgebung ein Datensatz-Interface vorgeschrieben. Dieses beinhaltet Methoden, welche direkt von den Keras-Modellen zum Trainieren und Evaluieren genutzt werden können (siehe Anforderung1a). Zudem werden zusätzliche Methoden, zum Filtern der Daten nach verschiedenen Kriterien (z.B. Label der Daten) (siehe Anforderung 2a) und Abrufen einer Teilmenge des Datensatzes (siehe Anforderung 2b), vorgeschrieben.

Um trotz der Anforderungen möglichst viele Datensätze zur Verfügung stellen zu können, wird eine Wrapper-Klasse für Tensorflow Datensätze entwickelt. Diese kann die verschiedenen Datensätze von Tensorflow um die geforderte Funktionalität erweitern, wodurch der Interpretationsumgebung eine Vielzahl von hochwertigen Datensätzen zur Verfügung stehen.

3) Interpretierbares Modell: Keras-Modelle eignen sich hervorragend dazu, schnell und einfach neuronale Netze zu trainieren und anzuwenden. Doch auf Grund ihrer Implementierung lassen sie viele Interpretationsmethoden nicht zu. Das Framework Lucid benötigt beispielsweise spezielle Modelle, in denen der zugrundeliegende Tensorflow-Graph, *eingefroren*[2] als Datei vorliegt.

Zusätzlich benötigt ein Anwender der Interpretationsumgebung genaue Beschreibungen des neuronalen Netzes, wie zum Beispiel Informationen über die Schichten, auf denen eine Interpretationsmethode angewendet werden soll. Dies beinhaltet die Art der Schicht (Faltung, voll vernetzt, Pooling etc.), die Anzahl der Neuronen und die genauen Namen der internen Tesorflow-Graph Operationen.

Diese Anforderungen erfüllt das Interpretierbare Modell, welches Methoden zum genauen beschreiben des neuronalen Netzes und ein Interface für die direkt Anbindung des Lucid Frameworks implementiert. Zudem kann ein Interpretierbares Modell direkt von einem Keras-Modell erstellt werden. Hierdurch verliert es zwar das Interface

[2]Als *eingefroren*, werden in Tensorflow Modelle bezeichnet, dessen gesamter Graph inklusive aller Rechenoperationen und Gewichte, im Dateiformat vorliegt.

welches Keras-Modelle anbieten, kann nun allerdings von allen, in der Interpretationsumgebung vorhandenen Interpretationsmethoden, interpretiert werden (sieht Anforderung 5a).

4) Interpretationsmethoden: Interpretationsmethoden können auf ein Interpretierbares Modell angewendet werden, um Informationen über das vom neuronalen Netz gelernten Wissen, oder dessen Entscheidungsprozess darzustellen. Hierfür wird nur das zu interpretierende Modell und im Falle der Interpretation eines Entscheidungsprozesses, ein Beispiel-Eingangswert benötigt. Optional können Interpretationsmethoden durch verschiedene Parameter beeinflusst und optimiert werden.
Die verschieden Interpretationsmethoden werden mithilfe des Lucid Frameworks entwickelt. Lucid ermöglicht ein gezieltes Umstrukturieren eines Tensorflow Berechnungs-Graphen. Dies ermöglicht die Umsetzung verschiedener Interpretationsmethoden, in denen die Eingangswerte eines neuronalen Netzes darauf optimiert werden, bestimmte Neuronen oder Schichten des neuronalen Netzes zu aktivieren. Diese Interpretationsmethoden werden auch als lernbasierte Interpretationsmethoden bezeichnet. Im Abschnitt V werden sowohl theoretische Aspekte von lernbasierten Interpretationsmethoden, als auch dessen Implementierung genauer erläutert.

5) Mensch-Computer-Schnittstelle: Die Qualität und Aussagekraft von Interpretationen hängt nicht nur von den Interpretationsmethoden ab, sondern auch wie diese dargestellt werden. Hierfür wird eine Mensch-Computer-Schnittstelle entworfen, welche die Resultate der Verschiedenen Interpretationsmethoden in einer für den Menschen intuitiven Form darstellt.
Zu diesem Zeitpunkt ist die Mensch-Computer-Schnittstelle noch nicht in der Interpretationsumgebung implementiert.

V. Interpretationsmethoden für neuronale Netze

In diesem Abschnitt werden die verschiedenen Interpretationsmethoden der Interpretationsumgebung genauer beschrieben. Alle bisher implementierten Interpretationsmethoden sind als lernbasierte Interpretationsmethoden[3] umgesetzt.

Der Ausdruck lernbasierte Interpretationsmethoden beschreibt eine Menge von Interpretationsmethoden, bei denen Interpretationen mithilfe eines Optimierungsverfahren gelernt werden. Hierfür werden die Eingangswerte eines maschinellen Lernalgorithmus, mithilfe des Gradientenverfahren, darauf optimiert den Lernalgorithmus auf eine vorgegebene Weise zu beeinflussen. Mithilfe der optimierten Eingangswerte können anschließend Aussagen über den Lernalgorithmus oder dessen Trainingsdatensatz getroffen werden.

[3]Der Ausdruck *lernbasierte Interpretationsmethode* ist eine, für diesen Artikel erzeugt Wortbildung, da in der wissenschaftlichen Literatur kein passender Begriff gefunden wurde.

Lernbasierte Interpretationsmethoden eignen sich besonders zum Interpretieren von neuronalen Netzen, können aber auch auf viele andere maschinelle Lernmethoden angewendet werden.

Bei neuronalen Netzen besteht eine lernbasierte Interpretationsmethode aus folgenden Komponenten:

a) Den Eingangswerten: Die Eingangswerte sind das zu optimierende Objekt und stellen zugleich das Resultat der Interpretationsmethode dar. Im einfachsten Fall ist der Eingangswert ein variabler Feature-Vektor oder eine variable Feature-Matrix (siehe V-A). Der Eingangswert kann aber auch komplexere mathematische Operationen enthalten (siehe V-C).

b) Dem neuronalen Netz: Das neuronale Netz ist das zu interpretierende Objekt und bestimmt in welcher Weise die Eingangswerte verarbeitet werden.

c) Der Verlustfunktion: Mit der Verlustfunktion wird der Einfluss beschrieben, welcher auf das neuronale Netz ausgeübt wird. Der Wert der Verlustfunktion beschreibt wie stark dieser Einfluss wirkt. Hierbei kann es sich um die maximale Aktivierung eines Neurons handeln (siehe V-A), das rekonstruieren einer Aktivierungsmatrix (siehe V-B) oder eine Permutation der Eingangswerte, welche die Entscheidung des neuronalen Netzes größtmöglich verfälschen (siehe V-C).

d) Einer Regularisierungsmethode: Optional können lernbasierte Interpretationsmethoden mit einer Regularisierungsmethode erweitert werden. Diese kann unerwünschte Nebeneffekte, wie beispielsweise das Lernen von feindlichen Beispielen (engl.: adversarial examples) reduzieren. Eine genauere Beschreibung von Regularisierungstechniken und warum diese benötigt werden findet sich im Abschnitt V-D.

Beispiele von lernbasierten Interpretationsmethoden finden sich in folgenden wissenschaftlichen Arbeiten: [7] [9] [14] [15] [17].

In den folgenden Abschnitten werden konkrete Beispiele von Interpretationsmethoden und deren Implementierung in der Interpretationsumgebung beschrieben.

A. Aktivierungsmaximierung von Neuronen

Die Aktivierungsmaximierung von Neuronen würde erstmals von Erhan et al. [5] in Jahre 2009 vorgestellt und anschließend von verschiedenen Wissenschaftlern [7] [18] [17] [10] [11] [8] weiterentwickelt.

Für die Aktivierungsmaximierung werden Eingangswerte des neuronalen Netzes gesucht, welche ein oder mehrere Neuronen besonders stark aktivieren. Diese Eingangswerte sollen die Informationen, welche in den besagten Neuronen gespeichert sind, repräsentieren.

Mathematisch kann die Aktivierungsmaximierung von Neuronen als Optimierungsproblem formuliert werden:

$$x* = \underset{x}{\operatorname{argmax}} f_n^l(x) - \mathcal{R}(x) \tag{1}$$

Hierbei sind $x*$ die gesuchten Eingangswerte. $f_n^l(x)$ ist die Aktivierungsfunktion für die Schicht l und das Neuron n. $\mathcal{R}(x)$ ist ein Regularisierung-Term.

In der Interpretationsumgebung wird die Aktivierungsmaximierung von Neuronen mithilfe einer Fehlerrückführung (engl. backpropagation) durchgeführt. Dieser Prozess ähnelt sehr dem Trainieren eines neuronalen Netzes. Der Unterschied besteht darin, dass nicht die Gewichte, sondern die Eingangswerte des neuronalen Netzes optimiert werden.
Dieser Ansatz ermöglicht eine effiziente Optimierung der Eingangswerte und eine hohe Flexibilität in der Wahl der zu aktivierenden Schichten und Neuronen. Als Regulierungsverfahren wird die Transformation-Robustheit angewendet. Dieses Verfahren wird im Abschnitt V-D genauer erläutert.

B. Feature Umkehrung

Die Umkehrung von Features wurde erstmals von Mahendran und Vedaldi [17] im Jahre 2014 vorgestellt. Bei dieser Technik wird versucht die kodierten Informationen aus den tiefen Schichten eines neuronalen Netzes zu rekonstruieren. Hierfür werden die Aktivierungen einer Schicht für ein konkretes Beispiel gespeichert. Anschließend werden die Eingangswerte gesucht, welche die gespeicherten Aktivierungen so gut wie möglich rekonstruieren.

Formal kann die Feature Umkehrung als ein Optimierungsproblem ausgedrückt werden:

$$x^* = \underset{x}{\operatorname{argmin}}\, l(f^l(x), f^l(x_0)) - \mathcal{R}(x) \tag{2}$$

x^* ist hierbei der gesuchte Eingangswert. Die Funktion $f^l(x)$ erzeugt die Repräsentation der Eingangswerte in der Schicht l. Um die Distanz zwischen der Repräsentation $f^l(x)$ und der originalen Repräsentation $f^l(x_0)$ zu ermitteln, wird eine Verlustfunktion $l()$ verwendet. Dies kann zum Beispiel der Euklidische Abstand sein, der auch von Mahendran und Vedaldi [17] verwendet wurde.

Da die Informationen in tieferen Schichten of aus (im Vergleich zur Eingangsschicht) wenigen abstrakten Features zusammengesetzt werden, entsprechen die gesuchten Eingangswerte nicht den Original-Eingangswerten, sondern stellen nur die Informationen dar, welche vom neuronalen Netz genutzt werden.

Eine Abwandlung der Feature Umkehrung wurde im Jahre 2018 von Olah et al. [14] entwickelt. Diese kann auf Faltungsschichten angewendet werden und stellt die kodierten Informationen einer Schicht in den verschiedenen Dimensionen der Faltungsschicht dar. Im Bereich des maschinellen Sehens kann diese Technik dazu verwendet werden, die vom neuronalen Netzen erkannten Features in den verschiedenen räumlichen Positionen des Bildes darzustellen. Aus diesem Grund wird diese Abwandlung der Feature Umkehrung im folgenden Text als *räumliche Feature Umkehrung* bezeichnet.

C. Erzeugen von Bedeutungsmasken

Das Erzeugen von Bedeutungsmasken[4] (engl. saliency map) ist eine Attribuierungsmethode und zeigt die Bedeutung einzelner Eingangs-Feature für eine konkrete Entscheidung. Sie können dazu genutzt werden, einzelne Entscheidungen von neuronalen Netzen logisch zu begründen.
Zum Erzeugen von Bedeutungsmasken gibt es viele verschiedene Forschungsansätze [8] [9] [10] [11]. Auf Grund des Aufbaus der Interpretationsumgebung, eignet sich die Methode von Fong und Vedaldi [9] besonders gut.

Bei dieser Attribuierungsmethode werden Bedeutungsmasken gelernt. Hierbei werden die Eingangswerte eines neuronalen Netzes gezielt permutiert, um einen bestimmten Ausgangswert zu verändert.
Die Permutation wird durch eine Maske erzeugt, welche die konstant bleibenden Eingangswerte überlagert. Hierbei können die Eingangswerte entweder durch einen konstanten Wert, Rauschen oder eine Unschärfe überlagert werden.
In der folgenden Gleichung wird dieser Vorgang am Beispiel der Überlagerung, mit einem konstanten Wert dargestellt.

Formal kann die Permutation Φ jedes Eingangswertes x_0 an der Stelle u, durch einen konstanten Wert μ_0, mithilfe der Maske $m : \Lambda \rightarrow [0, 1]$, mit folgender Formel beschrieben werden:

$$\Phi(u|x_0, m) = m(u)x_0(u) + (1 - m(u))\mu_0 \tag{3}$$

Das Ziel hierbei ist es, mit einer möglichst geringen Permutation, einen möglichst großen Effekt auf die Ausgangswerte des neuronalen Netzes auszuüben.
Dies kann formal als Optimierungsproblem ausgedrückt werden, bei dem eine Maske $m : \Lambda \rightarrow [0, 1]$ gesucht wird, welche ein $f_c(\Phi(x_0, m)) \ll f_c(x_0)$ erzeugt. Der Term $\lambda\|1 - m\|_1, \lambda\epsilon\mathbb{R}-$ fungiert als Regularisierung und sorgt dafür, dass ein möglichst kleiner Teil der Maske aktiv ist.

$$m* = \underset{m\epsilon[0,1]^\Lambda}{\operatorname{argmin}} \lambda\|1 - m\|_1 + f_c(\Phi(x_0, m)) \tag{4}$$

Hierdurch sollen besonders informative Eingangswerte mithilfe der Maske markiert werden.

Diese Technik wurde noch nicht in die Interpretationsumgebung implementiert. Die benötigten Schnittstellen sind jedoch vorhanden und die Software Architektur ist auf die Anforderungen, welche die Erzeugung von Bedeutungsmasken mit sich bring, ausgelegt.
Es ist geplant diese Attribuierungsmethode zu erweitern, indem der konstante Wert μ_0 aus der Formel 3 durch x_{min}

[4]Dies ist eine recht ungenaue Übersetzung des englischen Wortes *saliency map*, doch eine semantisch passende Bezeichnung.

ersetzt wird. x_{min} beschreibt die Eingangswerte eines neuronalen Netzes, welche einen minimalen Effekt auf dessen Ausgangswerte haben. Diese *irrelevanten Eingangswerte* können mithilfe von Aktivierungsmaximierung gelernt werden (siehe V-A). Der Unterschied einer herkömmlichen Aktivierungsmaximierung ist, dass Eingangswerte gesucht werden, welche die Summe der Ausgangsneuronen[5] des neuronalen Netzes, minimal aktivieren.

D. Die Schwierigkeiten von lernbasierten Interpretationsmethoden

Bei lernbasierten Interpretationsmethoden werden Eingangswerte von neuronalen Netzen darauf optimiert, bestimmte Aktivierungen im neuronalen Netz zu erzeugen. Mithilfe menschlicher Interpretation sollen aus diesen Eingangswerten, Rückschlüsse über das neuronale Netz oder den Trainingsdatensatz gezogen werden.

Im Bildbereich sollen hierdurch beispielsweise Interpretationsbilder entstehen, welche Muster, Objekte, Lebewesen oder andere Dinge zeigen, die von einem menschlichen Betrachter erkannt werden können. Leider zeigen die optimierten Eingangsbilder häufig nur sehr verrauschte Muster, obwohl sie die gewünschten Aktivierungen erzeugen.

Diese Muster zeigen große Ähnlichkeiten zu gegnerischen Beispielen (engl.: adversarial examples) [19]. Hierbei handelt es sich um Eingangswerte, welche ein neuronales Netz mithilfe weniger Einzelwerten, sehr stark beeinflussen. Diese sind zwar relevant um die Angreifbarkeit von neuronalen Netzen zu beurteilen, sie eignen sich allerdings weniger für die Interpretation des Netzes, da auf dieser Basis keine Annahmen über das neuronale Netz oder dessen Trainingsdatensatz getroffen werden können.

Um dieses Problem zu beheben werden Regularisierungsmethoden verwendet. Viele wissenschaftliche Arbeiten, im Kontext der Feature Visualisierung, beinhalten einen Forschungsteil über Regularisierungsmethoden.
Die Arbeit von Olah et al. [7] unterscheidet hierbei drei Gruppen von Regularisierungsmethoden:

a) Bestrafen von hohen Frequenzen: Eine sehr naheliegende Methode ist es, das Problem zu beheben, indem die hohen Frequenzen, also die Varianz zwischen naheliegenden Pixeln, direkt unterbunden wird. Dies kann explizit geschehen, indem hohe Frequenzen in der Verlustfunktion bestraft werden, oder implizit, indem das Eingangsbild bei jeden Optimierungsschritt mit einer Unschärfe versehen wird.

[5]Dies funktioniert nicht, wenn in der Ausgangsschicht die Softmax Aktivierung verwendet wird, da die Summe der Ausgangswerte einer Softmax Funktion immer gleich eins ist.

b) Transformations-Robustheit: Ein weiterer Lösungsansatz ist es, nur solche Eingangsbilder zuzulassen, welche robust gegen Transformationen sind. Hierfür werden bei jeden Optimierungsschritt zufällige Transformationen, wie Rotation, Skalierungen oder Verschiebung auf das Eingangsbild angewendet.
Diese Regularisierungsmethode wird zurzeit auch in der Interpretationsumgebung angewendet.

c) Verwenden von gelerntem Vorwissen: Um unnatürliche Eingangsbilder zu unterbinden, kann Vorwissen zur Regularisierung verwendet werden. Hierfür werden neuronale Netze darauf trainiert, zwischen synthetischen und realen Bildern zu unterscheiden. Mithilfe dieses Vorwissen können Eingangsbilder sowohl darauf optimiert werden, einen bestimmten Effekt im zu interpretierenden neuronalen Netz zu bewirken, als auch darauf besonders realistisch auszusehen.

VI. Ergebnisse

In diesen Abschnitt werden die ersten Ergebnisse der Interpretationsumgebung präsentiert. Hierfür wird im Abschnitt VI-A die Benutzbarkeit der Interpretationsumgebung anhand von Quellcode Beispielen demonstriert. Anschließend werden im Abschnitt VI-B verschiedene Resultate der Interpretationsmethoden abgebildet.

A. Benutzbarkeit

Durch die Verwendung des Framework Keras, können neuronale Netze benutzerfreundlich als Keras-Modell erstellt und trainiert werden. Um diese Modelle in interpretierbare Modelle umzuwandeln, wird die Klasse *InterpretableModel* der Interpretationsumgebung verwendet. Diese wandelt ein Keras-Modell, mithilfe der Klassenmethode *from_keras_model*, in ein interpretierbares Modell um.

```
i_model = InterpretableModel.\
from_keras_model(keras_modell, 'name')
```

Hierbei ist *keras_model* eine Instanz eines Keras-Modells, welches in das interpretierbare Modell *i_model* umgewandelt wird.
Bei der Erzeugung eines interpretierbaren Modells werden automatisch alle relevanten Daten in einem Ordner auf der Festplatte gespeichert. Hierdurch können interpretierbare Modelle durch Angabe des Verzeichnispfades bei der Initialisierung eines *InterpretableModel*, erneut geladen werden.

```
i_model = InterpretableModel(path_dir)
```

Hierbei ist *path_dir* eine Variable, die den Pfad zum Ordner enthält, in dem das interpretierbare Modell gespeichert ist.

Eine Instanz eines interpretierbaren Modells stellt verschiedene Interpretationsmethoden zur Verfügung, welche direkt als Methode aufgerufen werden können.

```
i_model.activate_dense_neuron(l_name, neuron)
i_model.activate_channel(l_name, channel)
i_model.feature_inversion(img, l_name)
```

Mit der Methode *activate_dense_neuron* können Neuronen aus voll vernetzen Schichten aktiviert werden, *activate_channel* aktiviert Kanäle aus Faltungsschichten und *feature_inversion* führt eine Feature Umkehrung aus. Hierbei beschreibt *l_name* den Namen der zu interpretierenden Schicht und *neuron* beziehungsweise *channel* die Nummer des zu untersuchenden Neurons oder Kanals. Im Falle der Feature Umkehrung wird außerdem ein Beispielbild *img* benötigt. Der Rückgabewert der Interpretationsmethoden ist jeweils ein Bild, welches die Interpretation darstellt. Beispiele solcher Bilder werden im folgenden Abschnitt vorgestellt.

B. Ergebnisse der Interpretationsmethoden

Für Test- und Demonstrationszwecken werden in diesem Abschnitt die Ergebnisse von verschiedenen Interpretationsmethoden präsentiert. Diese Interpretationen sollen eine möglichst repräsentative Auswahl, der bisher erstellten Interpretationen darstellen und somit die Stärken und Schwächen der Interpretationsumgebung aufzeigen.
Die hier gezeigten Interpretationen wurden auf drei neuronalen Klassifikator Netzen errechnet, welche in den folgenden Paragraphen kurz beschrieben werden:

a) SmallNet: Der Name SmallNet ist eine Bezeichnung für ein sehr kleines Faltungsnetz, welches für diesen Artikel auf dem Bilddatensatz Cifar10 [20] trainiert wurde. Dieses besteht aus drei hintereinandergeschalteten Faltungsschichten und einer voll vernetzten Schicht mit einer Softmax Ausgangsaktivierung.

b) MobileNetV2: Bei MobileNetV2 handelt es sich um ein neuronales Netz nach der MobileNet Architektur [21]. Dieses steht, inklusive vortrainierte Gewichte, im Tensorflow Framework zur freien Verfügung. Die für diesen Artikel verwendeten Gewichte wurden auf den Datensatz ImageNet [22] erstellt.

c) InceptionV1: Das neuronale Netz InceptionV1 stammt aus dem Lucid Framework und wird in mehreren wissenschaftlichen Arbeiten [7] [14] [15] zur Demonstration von Interpretationsmethoden verwendet. Dieses neuronale Netz beruht auf der GoogLeNet (Inceptoin) Architektur [23] und wurde auf ImageNet [22] trainiert.

In den Abbildungen 2 bis 6 sind verschiedene Resultate von Interpretationsmethoden zu sehen. Die Abbildungen 2, 3 und 4 zeigen die Aktivierungsmaximierung von einzelnen Faltungsschichten und Neuronen. Hierbei zeigt das linke Bild immer die Aktivierung einer der vorderen Faltungsschichten, das mittlere Bild die Aktivierung einer mittleren Faltungsschicht und das rechte Bild die Aktivierung eines Ausgangsneuron.
Während die Aktivierungen von Faltungsschichten beliebige, für Menschen oft unverständliche Features zeigen, hat ein Neuron der Ausgangsschicht eine semantische Bedeutung. Da es sich um Klassifikator Netze handelt, repräsentiert jedes Ausgangsneuron eine Klasse.
In den Abbildungen 5 und 6 sind Feature Umkehrungen auf MobileNetV2 und InceptionV1 zu sehen. Hier ist links das Originalbild dargestellt, im mittleren Bild die Repräsentation des Originalbildes aus einer der mittleren Faltungsschichten und im rechten Bild die Repräsentation aus einer der hinteren Faltungsschichten.

C. Bewertung der Interpretationergebnissen

Dieser Abschnitt berichtet von den ersten Erfahrungen, welche mit der Interpretationsumgebung gemacht wurden. Außerdem werden die Interpretationen der Abbildungen 2 bis 6 aus dem Bauchgefühl heraus bewertet. Hierfür gilt eine Interpretation als gut, wenn sie Aussagen über das neuronale Netz, den Trainingsdatensatz oder dessen kausale Entscheidungsfindung zu treffen scheint.
Um die Qualität einer Interpretation präziser zu evaluieren ist ein umfangreicher wissenschaftlicher Experimentaufbau vonnöten (siehe Zach [1]), welcher den Umfang dieser Arbeit übersteigen würde.

Bei der Interpretation von verschiedenen neuronalen Netzen fällt auf, dass es einen großen Qualitätsunterschied der Interpretationen verschiedener Netze gibt.
Bei kleinen Modellen wie beispielsweise SmallNet sind Interpretationen leicht zu erstellen, doch aufgrund der begrenzten Bildgröße, wenig Aussagekräftig (siehe Abbildung 3).
Bei sehr großen und komplexen neuronalen Netzen ist es schwierig eine aussagekräftige Interpretation zu erzeugen (siehe Abbildung 3). Hinzu kommt, dass die Interpretationen umso schwieriger zu erzeugen sind, um so tiefer sich die zu interpretierende Schicht im neuronalen Netz befindet (siehe Abbildung 5b und 5c).
Eine Ausnahme ist das Modell InceptionV1 welches trotz seiner Größe, sehr gut zu interpretieren ist (siehe Abbildung 4 und 6). Hier zeigen die Interpretationen Objekte in die leicht eine Bedeutung hineininterpretiert werden kann. Ähnlich gute Interpretationsergebnisse finden sich in den wissenschaftlichen Arbeiten [7] [14] [15], welche alle das InceptionV1 Netz zum Erzeugen ihrer Interpretationen verwendet haben.

VII. Weitere Forschungen

Die Interpretationsumgebung wurde für den Zweck des experimentellen Evaluierens von Interpretationsmethoden entwickelt (siehe Zach [1]). Die ersten Erfahrungswerte deuten allerdings nicht darauf hin, dass die Qualität der Interpretationen ausreicht, um diese Art der Experimente durchzuführen. Eine Ausnahme bilden hierbei die Interpretationen, welche auf dem neuronalen Netz InceptionV1 erzeugt wurden. Diese scheinen eine ausreichend gute Qualität zu habe, um Annahmen über das neuronale Netz oder den Trainingsdatensatz aufzustellen.
Um dieses Problem der mangelnden Qualität der Interpretationen zu lösen, gibt es zwei Vorgehensweisen:

a) Verbessern der Interpretationsmethoden: Eine naheliegende Herangehensweise zum Verbessern der Interpretationen, ist das Verbessern der Interpretationsmethoden. Vor allen die Entwicklung von wirkungsvolleren Regularisierungsmethoden kann die Qualität der Interpretationen stark verbessern. Diese sollen dafür sorgen, dass Interpretationen nicht nur das zu Interpretierende neuronale Netz auf die richtige weise beeinflussen, sondern auch Resultate liefern, die eine menschliche Interpretation zulassen.

b) Verbessern der zu Interpretierenden neuronalen Netze: Ein zweiter Lösungsansatz basiert auf der genauen Untersuchung der zu interpretierenden neuronalen Netze. Hierbei gilt es herauszufinden, welche Eigenschaften der neuronalen Netze, gute Interpretationsergebnisse ermöglichen. Da das neuronale Netz InceptionV1 gute Interpretationsergebnisse geliefert hat, gilt es herauszufinden, wodurch es sich zu anderen neuronalen Netzen, wie beispielsweise MobileNetV2, unterscheidet und wie stark sich diese Unterschiede auf die Qualität der Interpretationen auswirken.

LITERATUR

[1] J. Zach, "Experiment zur evaluierung der nützlichkeit von interpretationsmethoden für neuronale netze," 2020.

[2] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," 2017.

[3] Z. C. Lipton, "The mythos of model interpretability," 2016.

[4] B. Goodman and S. Flaxman, "European union regulations on algorithmic decision-making and a "right to explanation"," *AI Magazine*, vol. 38, p. 50–57, Oct 2017.

[5] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, "Visualizing higher-layer features of a deep network," *Technical Report, Univeristé de Montréal*, 01 2009.

[6] A. Mordvintsev, C. Olah, and M. Tyka, "Inceptionism: Going deeper into neural networks," 2015.

[7] C. Olah, A. Mordvintsev, and L. Schubert, "Feature visualization," *Distill*, 2017. https://distill.pub/2017/feature-visualization.

[8] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," 2013.

[9] R. C. Fong and A. Vedaldi, "Interpretable explanations of black boxes by meaningful perturbation," *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[10] P.-J. Kindermans, K. T. Schütt, M. Alber, K.-R. Müller, D. Erhan, B. Kim, and S. Dähne, "Learning how to explain neural networks: Patternnet and patternattribution," 2017.

[11] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," 2013.

[12] A. Bilal, A. Jourabloo, M. Ye, X. Liu, and L. Ren, "Do convolutional neural networks learn class hierarchy?," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, p. 152–162, Jan 2018.

[13] M. Kahng, P. Y. Andrews, A. Kalro, and D. H. Chau, "Activis: Visual exploration of industry-scale deep neural network models," 2017.

[14] C. Olah, A. Satyanarayan, I. Johnson, S. Carter, L. Schubert, K. Ye, and A. Mordvintsev, "The building blocks of interpretability," *Distill*, 2018. https://distill.pub/2018/building-blocks.

[15] S. Carter, Z. Armstrong, L. Schubert, I. Johnson, and C. Olah, "Activation atlas," *Distill*, 2019. https://distill.pub/2019/activation-atlas.

[16] B. Kim, M. Wattenberg, J. Gilmer, C. Cai, J. Wexler, F. Viegas, and R. Sayres, "Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav)," 2017.

[17] A. Mahendran and A. Vedaldi, "Understanding deep image representations by inverting them," 2014.

[18] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune, "Synthesizing the preferred inputs for neurons in neural networks via deep generator networks," 2016.

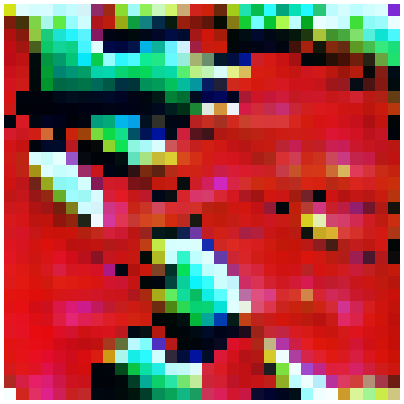
[19] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," 2013.

[20] A. Krizhevsky, "Learning multiple layers of features from tiny images," *University of Toronto*, 05 2012.

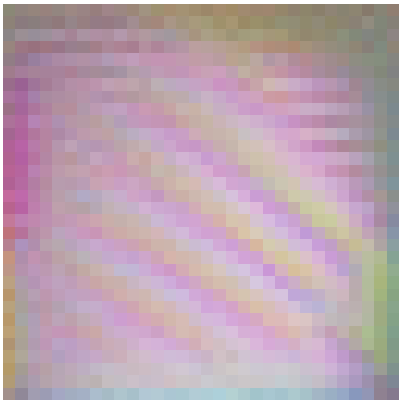
[21] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017.

[22] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

[23] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," 2014.



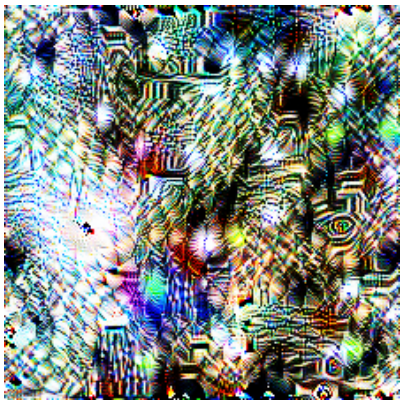
(a) Aktivierung eines Kanals aus einer vorderen Faltungsschicht (*conv2d1 BiasAdd*).



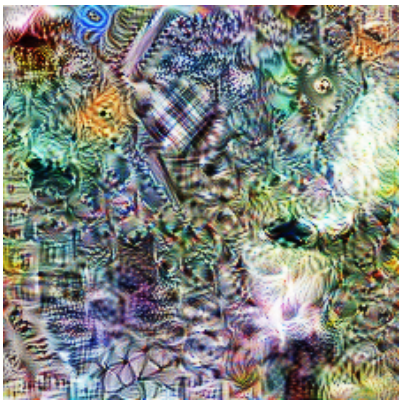
(b) Aktivierung eines Kanals aus einer mittleren Faltungsschicht (*conv2d3 BiasAdd*).

(c) Aktivierung des Neuron der Ausgangsschicht, welches die Klasse Pferd repräsentiert.

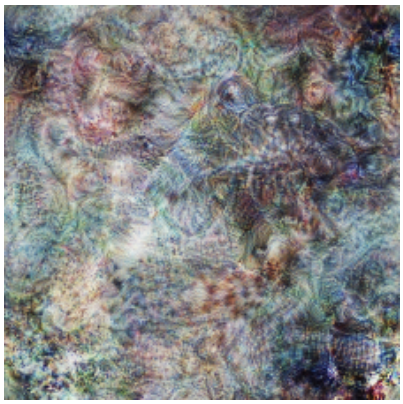
Abbildung 2: Aktivierungen von Faltungsschichten und Neuronen des neuronalen Netzes SmallNet.



(a) Aktivierung eines Kanals aus einer vorderen Faltungsschicht (*block3 project Conv2D*).

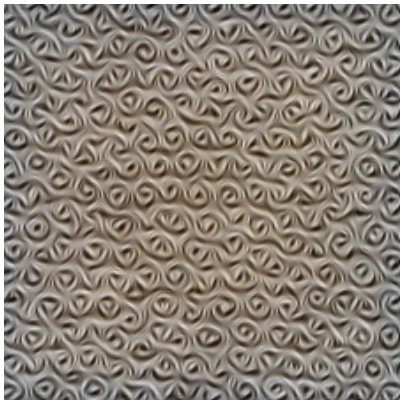


(b) Aktivierung eines Kanals aus einer mittleren Faltungsschicht (*block6 project Conv2*).

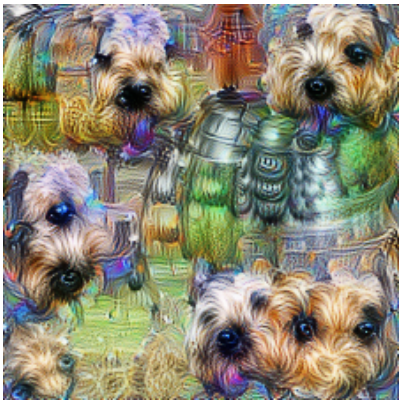


(c) Aktivierung eines Neuron der Ausgangsschicht.

Abbildung 3: Aktivierungen von Faltungsschichten und Neuronen des neuronalen Netzes MobileNetV2.



(a) Aktivierung eines Kanals aus einer vorderen Faltungsschicht (*mixed 3a*).



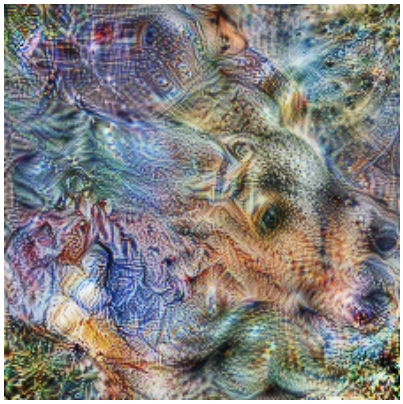
(b) Aktivierung eines Kanals aus einer mittleren Faltungsschicht (*mixed 5a*).

(c) Aktivierung eines Neuron der Ausgangsschicht.

Abbildung 4: Aktivierungen von Faltungsschichten und Neuronen des neuronalen Netzes InceptionV1.

(a) Eingangsbild der Feature Umkehrung.

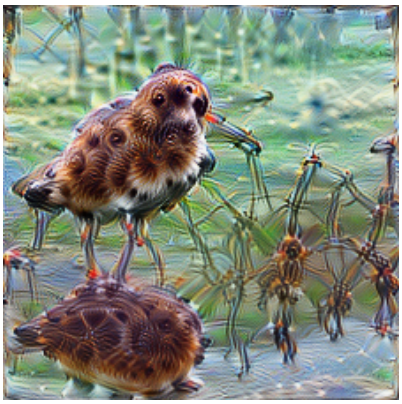
(b) Feature Umkehrung in einer mittleren Faltungsschicht (*block10 project Conv2D*).



(c) Feature Umkehrung in einer hinteren Faltungsschicht (*block16 project Conv2D*).

Abbildung 5: Feature Umkehrung in verschiedene Faltungsschichten des neuronalen Netzes MobileNetV2.

(a) Eingangsbild der Feature Umkehrung.



(b) Feature Umkehrung in einer mittleren Faltungsschicht (*mixed4a*).

(c) Feature Umkehrung in einer hinteren Faltungsschicht (*mixed5a*).

Abbildung 6: Feature Umkehrung in verschiedene Faltungsschichten des neuronalen Netzes InceptionV1.