

Analyse der Integrated Gradients Methode anhand von Adversarial Examples

Analysis of the Integrated Gradients Method by means of Adversarial Examples

Marek Schuster

Bachelor-Abschlussarbeit

Betreuer: Professor Dr. Hans-Peter Beise

Trier, 05.01.2022

Kurzfassung

Im Rahmen dieser Ausarbeitung wird ein Einblick in die Grundlagen sowie in einzelne wichtige Techniken aus den Bereichen der *explainable AI* und der *adversarial examples* gegeben. Insbesondere werden die *Integrated Gradients* Methode zur Analyse von Entscheidungen eines neuronalen Netzes und die *fast gradient sign* Methode zum Angreifen eines neuronalen Netzes sowie verschiedene Verteidigungsstrategien vorgestellt. Anschließend werden selbst trainierte Netze durch *adversarials* verschiedener Datensätze angegriffen, um ihre Robustheit zu evaluieren. Danach werden die IG Attributionen der *adversarials* analysiert. Diese werden verglichen mit Attributionen für Bilder, die durch ein normales Rauschen gestört wurden. Es kommt zum Schluss, dass *adversarials* die Attributionen der *integrated Gradients* Methode nicht maßgeblich anders beeinflussen als ein normales Rauschen.

Abstract

In this work we give insight on some important techniques from the fields of explainable AI and adversarial examples. Those are the Integrated Gradients method for analysing what caused the prediction of a neural network and the fast gradient sign method for generating adversarial examples to attack a network as well as different defensive techniques against them. We will train our own networks for different datasets, attack them using FGSM and evaluate the robustness of our networks. Then we will use IG to analyse the attributions of these adversarials. Also, we will analyse attributions of images pertubated by random noise to see if adversarials have any special effect on IG. We come to the conclusion that this is not the case.

Inhaltsverzeichnis

1 Einleitung und Problemstellung	1
2 Grundlegende Konzepte	2
2.1 Deep Learning	2
2.2 Convolutional Neural Networks	3
2.3 Explainable AI	4
3 Datensätze	5
3.1 Fashion-MNIST	5
3.2 CIFAR-10	5
3.3 Dogs vs. Cats	6
4 Integrated Gradients	8
4.1 Attributionsmethoden	8
4.2 Fundamentale Axiome	9
4.3 Definition von IG	9
4.4 Definition Pfadmethoden	10
4.5 Verwendung von IG durch das Riemann-Integral	11
5 Adversarial examples	14
5.1 Goodfellow's Algorithmus	14
5.2 Übertragbarkeit von Adversarials	15
5.3 Adversarials in der echten Welt	15
5.4 Verteidigungsstrategien	16
5.4.1 Gradient masking	16
5.4.2 Adversarial Training	17
5.4.3 Defensive distillation	17
5.5 Sind Angriffe auf KI unausweichlich?	17
6 Experimente	19
6.1 Architektur der CNNs	19
6.1.1 Eigene Netze	19
6.1.2 Vortrainierte Netze	20
6.2 Experimente und Ziele	21

6.2.1 Adversarial Angriffe mit FGSM	21
6.2.2 Analyse der IG Attributionen von adversarials	21
6.2.3 Vergleich mit zufällig generierter Störung	21
7 Ergebnisse und Erkenntnisse	22
7.1 Robustheit der Netze	22
7.2 IG Attributionen für perturbierte Bilder	24
8 Zusammenfassung und Ausblick	31
Selbstständigkeitserklärung	35

Abbildungsverzeichnis

2.1	Darstellung eines CNN mit zwei <i>convolutional layers</i> mit <i>max pooling</i> und einem <i>fully connected layer</i> , welches Bilder im Format $112 \times 122 \times 3$ verarbeitet. Erstellt mit Hilfe von (LeNail 2019)	4
3.1	25 Beispielbilder aus dem Fashion-MNIST Datensatz.	6
3.2	25 Beispielbilder aus dem CIFAR-10 Datensatz.	6
3.3	25 Beispielbilder aus dem Dogs vs. Cats Datensatz.	7
4.1	Beispiel für einen geraden Pfad mit x' als schwarzem Bild und dem Bild einer Katze aus Dogs vs. Cats als x	12
4.2	IG Attributionen für die Katze aus der vorherigen Grafik. Approximiert mit 240 Schritten.	12
4.3	Vergleich der Auswirkung verschiedener <i>baselines</i> auf die Attribution durch IG.	13
5.1	Beispiel für einen <i>adversarial</i> Angriff via FGSM auf ein selbst trainiertes CNN mit einem Bild aus dem CIFAR-10 Datensatz. Durch den Angriff verändert sich die Klassifikation von <i>Pferd</i> zu <i>Vogel</i> . Grafik inspiriert von (Goodfellow u. a. 2014, Abb. 1)	15
7.1	Vergleich der Fehlerraten der Klassifizierung für FGSM <i>adversarials</i> und Störung der Bilder durch Rauschen nach ϵ -Stärke.	23
7.2	Die Grafiken (a), (b) und (c) zeigen jeweils die generierten Störungen für das Bild der Katze. Links ist das Ergebnis von FGSM auf dem selbst trainierten Netz, in der Mitte die zufällig generierte Störung und rechts das durch <i>gradient masking</i> verhinderte <i>adversarial</i> von InceptionV3 zu sehen. Grafik (d) zeigt ein Beispiel aus dem Dogs vs. Cats Datensatz für die 9 auf dem selbst trainierten Modell generierten <i>adversarials</i> nach ϵ -Stärke. Grafik (e) hingegen visualisiert den Effekt des Rauschens. Grafik (f) zeigt ein Beispiel für das Auftreten von <i>gradient masking</i> durch das InceptionV3 Netz.	24
7.3	Beispiele für die Attributionsmasken verschiedener Perturbationen auf Bildern aus den 3 Datensätzen.	25

7.4	Durchschnittlicher MSE der Differenz zwischen der Attributionsmaske des Originalbildes und der des zugehörigen $\epsilon_{adversarials}$ pro Datensatz.	26
7.5	Durchschnittliche Standardabweichung der Differenz zwischen der Attributionsmaske des Originalbildes und der des zugehörigen $\epsilon_{adversarials}$ pro Datensatz.	28
7.6	Die neue violette Kurve beschreibt den Verlauf der Standardabweichung allein für die Attributionsmasken der $\epsilon_{adversarials}$. Sie kommt zustande durch den Durchschnitt der Werte über 100 Bilder, also insgesamt 900 Masken.	29
7.7	Durchschnittliche Korrelation nach Pearson zwischen der Attributionsmaske des Originalbildes und der des zugehörigen $\epsilon_{adversarials}$	29

Tabellenverzeichnis

3.1	Die verschiedenen Datensätze im Überblick.	7
6.1	Die verschiedenen CNNs mit ihrer Genauigkeit und ihren Trainingsparametern im Überblick.	20
6.2	Architektur der selbst trainierten CNNs im Überblick.	20

1

Einleitung und Problemstellung

Aufgrund jüngster Errungenschaften und Fortschritte im Bereich der künstlichen Intelligenz, insbesondere dem der neuronalen Netze (NNs), werden diese in unserer heutigen Welt immer wichtiger. In vielen Bereichen wie Produktempfehlung oder Sprachassistenz ist künstliche Intelligenz bereits ein fester Bestandteil des Alltags geworden. Auch in sicherheitskritischen Bereichen mit hoher Verantwortung wird sie immer wichtiger, beispielsweise im Bereich der Diagnostik in der Medizin und sie soll immer wichtiger werden, wie im Bereich der selbstfahrenden Autos. Gerade für diese Bereiche, aber auch im Allgemeinen, bestehen jedoch noch zwei große Kritikpunkte bezüglich neuronaler Netze: Verständnis und Sicherheit. Einerseits ist die Arbeitsweise der heute meist verwendeten NNs aufgrund ihrer hohen und stetig steigenden Komplexität kaum nachzuvollziehen. Aus diesem Grund existieren mittlerweile eigene Forschungsgebiete, welche sich mit der Frage befassen, wie eine gute Kommunikation zwischen einer künstlichen Intelligenz und dem Entwickler oder Anwender geschaffen werden kann, um ein verbessertes Verständnis eines NNs zu erhalten.

Andererseits ist die Sicherheit solcher Systeme ein ernstzunehmendes Problem. Die Forschung der letzten Jahre zeigt, dass neuronale Netze anfällig sind für Angriffe durch speziell angefertigte Eingabedaten namens *adversarial examples*. Erstmals zeigten Szegedy u. a. (2013), dass es möglich ist, Bilddaten so zu verändern, dass selbst die besten Netze der damaligen Zeit Fehlentscheidungen trafen. Und zwar so, dass die Veränderung mit bloßem Auge nicht wahrzunehmen ist (Xiao u. a. 2017).

Im Rahmen dieser Ausarbeitung werden beide Seiten beleuchtet. Nach einem kurzen Einblick in die Definition notwendiger Grundlagen und der verwendeten Datensätze wird ein wichtiger Baustein für besseres Verständnis vorgestellt, die *integrated gradients* Methode. Anschließend werden die *adversarial examples* und ihre besonderen Eigenschaften genauer beleuchtet sowie Möglichkeiten zur Erstellung und Verteidigung vorgestellt. In der Hoffnung neue Erkenntnisse gewinnen zu können, werden beide Seiten durch die Analyse von *adversarial examples* mit der *Integrated Methode* miteinander kombiniert.

2

Grundlegende Konzepte

In diesem Kapitel wird eine kurze Einführung in die für den restlichen Verlauf dieser Arbeit wichtigen Konzepte gegeben.

2.1 Deep Learning

Das *Deep Learning* ist ein wichtiges Teilgebiet der künstlichen Intelligenz (KI) beziehungsweise des *Machine Learnings* (ML, dt. maschinelles Lernen). Das maschinelle Lernen hat zum Ziel, einem Computer ein gewisses Verhalten beizubringen ohne dieses explizit zu programmieren. Stattdessen soll es ihm durch das Sammeln von Erfahrung und Wissen antrainiert werden. Das traditionelle *Machine Learning* stieß bei komplexeren Aufgaben wie Bilderkennung, aufgrund äußerer Einflüsse wie dem „Fluch der Dimensionalität“ (Bengio und LeCun 2007), an seine Grenzen. Diese Probleme konnten durch *Deep Learning* gelöst werden (Yuan u. a. 2017). Beim DL werden *Deep Neural Networks* (DNNs, dt. tiefe künstlich neuronale Netze) verwendet. Sie bestehen aus n vielen hierarchisch zusammengesetzten parametrischen Funktionen f , die eine Eingabe \vec{x} verarbeiten. Jede Funktion $f_i, i \in \{1, 2, \dots, n\}$ besteht aus einer Schicht (eng. layer) von Neuronen, welche die elementaren Bestandteile eines Netzes darstellen. Sie wenden eine vordefinierte Aktivierungsfunktion auf die gewichtete Repräsentation der Daten an, die sie aus der vorherigen Schicht bekommen, um ihre eigene neue Repräsentation zu schaffen. Diese wird dann wieder an die nächste Schicht weitergereicht. Jede Schicht besitzt einen Gewichtsvektor $\vec{\theta}_i$, der die Aktivierung der jeweiligen Neuronen beeinflusst. Diese Gewichte enthalten das Wissen des DNNs und werden durch das Training bestimmt und evaluiert (Papernot u. a. 2016). Mit dieser Definition lässt sich ein DNN F formal aufschreiben als:

$$F(\vec{x}) = f_n(\vec{\theta}_n, f_{n-1}(\vec{\theta}_{n-1}, f_{n-2}(\vec{\theta}_{n-2}, \dots f_2(\vec{\theta}_2, f_1(\vec{\theta}_1, \vec{x})))) \quad (2.1)$$

Im weiteren Verlauf dieser Arbeit wird eine bestimmte Gruppe von DNNs weiter betrachtet, die *classifier* (dt. Klassifikatoren). Diese lernen ihre Eingabe einer vordefinierten Klasse aus einer Gruppe von Klassen zuzuordnen und werden somit beispielsweise in der Bilderkennung verwendet. Während des Trainings wird dem *classifier* F eine große Menge von Eingabe/Klasse-Paaren (\vec{x}, \vec{y}) übergeben. Diese

werden wie in Gleichung 2.1 *vorwärts* durch das Netz geschickt. Dabei beurteilt die Lossfunktion \mathcal{L} den Unterschied zwischen der Klassifikation $F(\vec{x})$ und dem erwarteten Ergebnis \vec{y} . Der Mittelwert des Loss wird als Kostenfunktion des Netzes bezeichnet. Um diese zu minimieren, wird meist der Backpropagation Algorithmus (Chauvin 2013) angewendet, bei dem das Netz nun *rückwärts* durchlaufen wird und die Gewichtsvektoren $\vec{\theta}_i$ angepasst werden (Boenisch u. a. 2021). Im restlichen Verlauf der Arbeit wird auf die Vektornotation verzichtet.

2.2 Convolutional Neural Networks

Eine besondere Architektur der DNNs sind die *Convolutional Neural Networks* (CNN, dt. etwa faltendes neuronales Netz) (Bengio und LeCun 1997), welche insbesondere für die Aufgabe der Bilderkennung besonders gut geeignet sind (Yuan u. a. 2017) und deshalb im weiteren Verlauf dieser Ausarbeitung verwendet werden. Der Name kommt von der mathematischen Operation *convolution* (dt. Faltung). Ein Netz zählt als CNN, wenn mindestens eine seiner Schichten *convolution* anstelle von regulärer Matrixmultiplikation verwendet (Goodfellow u. a. 2016, S. 326). Diese Schichten werden dann auch als *convolutional layer* bezeichnet. Die *convolution* liefert ein gewichtetes Mittel der Eingabe. Für eine Eingabe x , eine Gewichtsfunktion w und einen Zeitpunkt t ist die Faltung definiert als:

$$s(t) = (x * w)(t) \quad (2.2)$$

Das Ergebnis wird auch als *feature map* bezeichnet. Damit die Operation die erwarteten Ergebnisse liefert, muss w eine geeignete Wahrscheinlichkeitsdichtefunktion sein und für alle negativen Werte muss w null ergeben (Goodfellow u. a. 2016, S. 327f). In der Literatur wird w meist als *Kernel* bezeichnet. Ein *convolutional layer* führt diese *convolution* mehrmals aus. Ziel dabei ist es, insbesondere in den obersten Schichten des Netzes, aussagekräftige Merkmale wie beispielsweise Kanten in den Daten zu erkennen und die weitere Verarbeitung dann auf diese zu reduzieren. Dadurch müssen deutlich weniger Parameter gespeichert werden, wodurch CNNs eine bedeutend geringere Komplexität und damit auch geringeren Speicherverbrauch als gleichwertige DNNs erreichen (Goodfellow u. a. 2016, S. 331). Im Anschluss an die *convolution* wird meist eine *pooling* (dt. bündelungs) Funktion angewendet. Diese ersetzt die Ausgabe bestimmter Neuronen durch eine statistische Zusammenfassung der Ausgabe ihrer Nachbarn (Goodfellow u. a. 2016, S. 335). Die im weiteren Rahmen dieser Ausarbeitung verwendete Art des *poolings* ist die *max pooling* Funktion von Zhou und Chellappa (1993). Sie liefert den höchsten Wert innerhalb einer rechteckigen Nachbarschaft zurück. Durch das *pooling* wird die Komplexität der Daten noch weiter reduziert. Außerdem hilft *pooling* dabei, die Repräsentation der Daten innerhalb der Neuronen robust gegenüber leichter Veränderungen der Eingabe zu machen (Goodfellow u. a. 2016, S. 336). Eine beispielhafte Visualisierung eines CNNs ist in Abbildung 2.1 zu sehen.

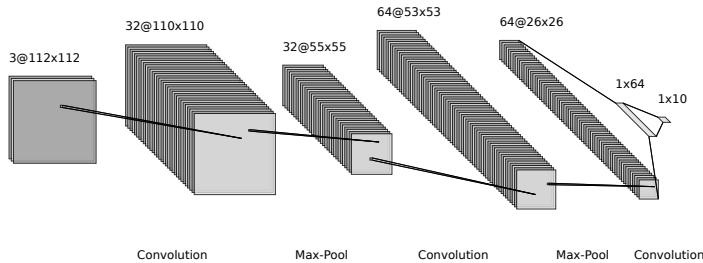


Abbildung 2.1: Darstellung eines CNN mit zwei *convolutional layern* mit *max pooling* und einem *fully connected layer*, welches Bilder im Format $112 \times 122 \times 3$ verarbeitet. Erstellt mit Hilfe von (LeNail 2019)

2.3 Explainable AI

Insbesondere durch die oben beschriebenen Techniken in Kombination mit stetig wachsender Rechenkapazität moderner Computer, hat KI in den letzten Jahren einen großen Aufschwung erlebt und ist in immer mehr Bereichen des täglichen Lebens zu finden. Auch für Bereiche mit sehr hoher Verantwortung wird KI immer wichtiger, beispielsweise für selbstfahrende Autos oder in der Medizin. Problematisch ist dabei allerdings, dass die Netze durch steigende Komplexität aufgrund höherer möglicher Rechenleistung zwar deutlich *intelligenter* werden, andererseits werden die Modelle dadurch aber auch immer undurchsichtiger (Google 2019). DNNs auf dem neuesten Stand der Technik ähneln einer Black-Box und die Gründe für die Entscheidung eines Netzes sind nur schwer nachzuvollziehen, was für die zuvor angesprochenen Einsatzgebiete mit hoher Verantwortung ein großes Problem darstellt. Aus diesem Grund ist ein neues Forschungsgebiet im Bereich des maschinellen Lernens entstanden namens *explainable AI* (XAI, dt. erklärbare KI). Ziel der XAI ist es, Modelle **interpretierbar** zu machen, sodass ein Modell seine Entscheidung erklären und diese Erklärung dann von Menschen evaluiert werden kann. Da jedoch lange keine einheitliche Definition für *Interpretierbarkeit* von Modellen existierte, definierten Doshi-Velez und Kim (2017) diese wie folgt: die Fähigkeit (die Entscheidungen eines Modells) zu erklären oder zu präsentieren, auf eine für Menschen verständliche Art und Weise. Ein XAI System muss also nicht nur die immer komplexer werdenden ML Architekturen wie DNNs erforschen, was an sich schon eine anspruchsvolle Aufgabe wäre, sondern es muss auch eine an den Menschen gerichtete Erklärung liefern können. Das bedeutet, dass für XAI auch die Lehren anderer Forschungsgebiete wie der Psychologie, Philosophie und Human-Computer-Interaction wichtig sind (Google 2019).

3

Datensätze

In diesem Kapitel werden die für die Experimente verwendeten Datensätze vorgestellt. Alle drei sind bekannte und oft erforschte Datensätze, wodurch ein Vergleich zwischen den Ergebnissen dieser Ausarbeitung und anderen Arbeiten einfach möglich ist. Zusätzlich sind sie simpel genug, um schnell und ressourcenschonend Netze zu trainieren und Ergebnisse zu liefern; aber trotzdem genügend umfangreich, um aussagekräftig zu sein. Eine kurze Übersicht der Struktur aller Datensätze ist in Tabelle 3.1 zu sehen.

3.1 Fashion-MNIST

Der Fashion-MNIST Datensatz (Xiao u. a. 2017) teilt seine Eigenschaften mit dem bekannten MNIST (Modified National Institute of Standards and Technology) Datensatz (LeCun u. a. 1998). Beide bestehen aus insgesamt 70.000 28×28 Pixel großen Graustufenbildern. Davon sind 60.000 Trainingsbilder und 10.000 Testbilder, jeweils gleichmäßig verteilt auf 10 Klassen (T-Shirt, Hose, Pullover, Kleid, Mantel, Sandale, Hemd, Sneaker, Tasche, Stiefel). Eine Visualisierung mehrerer Beispiele ist in Abbildung 3.1 zu sehen. Aufgrund seiner Simplizität und Größe ist der MNIST Datensatz einer der beliebtesten in der Forschung und wird oftmals als erstes verwendet, um neue Methoden zu testen (Xiao u. a. 2017). Ziel von Fashion-MNIST ist es, all diese positiven Eigenschaften von MNIST beizubehalten und die Datensätze so leicht austauschbar wie möglich zu gestalten; aber trotzdem ein schwierigeres Problem als Maßstab zu erzeugen.

3.2 CIFAR-10

Der CIFAR-10 (Canadian institute of advanced research) Datensatz von Krizhevsky u. a. (2009) besteht aus 60.000 32×32 Pixel großen RGB-Bildern, aufgeteilt auf 10 verschiedene Klassen (Flugzeug, Auto, Vogel, Katze, Reh, Hund, Frosch, Pferd, Schiff, LKW). In Abbildung 3.2 sind einige Beispiele zu sehen.



Abbildung 3.1: 25 Beispielbilder aus dem Fashion-MNIST Datensatz.

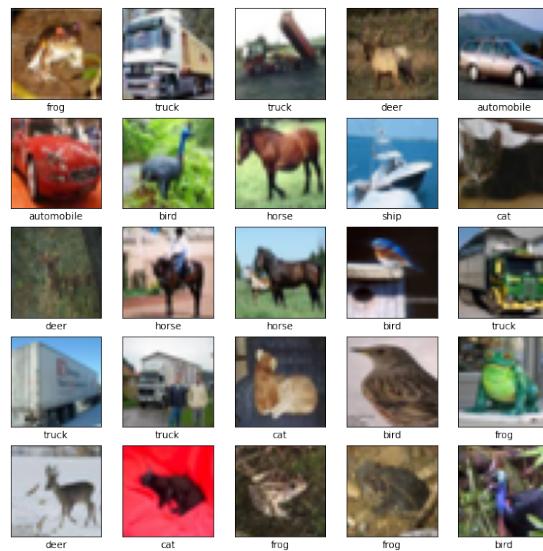


Abbildung 3.2: 25 Beispielbilder aus dem CIFAR-10 Datensatz.

3.3 Dogs vs. Cats

Der Dogs vs. Cats Datensatz stammt aus einem ML Wettbewerb von Microsoft und besteht aus 25.000 Bildern von Hunden und Katzen (jeweils 12.500) (Microsoft Research 2013). In der Abbildung 3.3 sind mehrere Beispiele zu sehen. Der Datensatz besteht aus genau zwei Klassen: Hund und Katze. Dieser Kontrast zu den anderen beiden Datensätzen macht diesen Datensatz für die folgenden Experimente interessant.



Abbildung 3.3: 25 Beispielbilder aus dem Dogs vs. Cats Datensatz.

Datensatz	Klassen	Trainingsbilder	Testbilder	Grösse	Farbraum
Fashion-MNIST	10	60.000	10.000	28×28	Graustufen
CIFAR-10	10	50.000	10.000	32×32	RGB
Dogs vs. Cats	2	25.000	12.500	112×112^1	RGB

Tabelle 3.1: Die verschiedenen Datensätze im Überblick.

¹ Nachträglich skaliert, im Original sind die Bilder ungleichmäßig groß.

4

Integrated Gradients

In diesem Kapitel wird eine wichtige Errungenschaft der *explainable AI* vorgestellt: Die *Integrated Gradients* (IG, dt. integrierte Gradienten) Methode von Sundararajan u. a. (2017). Diese ist eine Technik aus der Gruppe der Attributionsmethoden. Diese haben zum Ziel, eine Zuordnung der Vorhersage eines DNNs zu seiner Eingabe zu erstellen. Dies wird im Folgenden als Attribution bezeichnet.

4.1 Attributionsmethoden

Formal sei F ein DNN mit einer Eingabe x wie in 2.1. Dann ist eine Attribution bezüglich x im Verhältniss zu einem Referenzwert x' definiert als:

$$A(x, x') = (a_1, \dots, a_n) \in R^n \quad (4.1)$$

Wobei gilt, dass a_i aussagt wie wichtig x_i für die Entscheidung von F war (Sundararajan u. a. 2017). Der Referenzwert x' wird in der Literatur meist als *baseline* bezeichnet. So würde A beispielsweise im Falle von Bilderkennung aussagen, wie wichtig welcher Pixel für F war. Ein möglicher Anwendungsfall für eine solche Attribution wäre beispielsweise ein Netz, welches medizinische Diagnosen aufgrund von Bilderkennung stellt. Die Attribution könnte Ärzten helfen, die Gründe für die Diagnose des DNNs nachzuvollziehen und dadurch die Qualität dieser besser zu bewerten (Sundararajan u. a. 2017). Heutzutage lassen sich die wichtigsten dieser Attributionsmethoden in zwei Kategorien einteilen, gradientenbasiert und approximationsbasierte Methoden (Pan u. a. 2021). Die erste wichtige gradientenbasierte Attributionsmethode war *Saliency Maps* (Simonyan u. a. 2013), bei der eine schnelle Attribution direkt über den Gradienten der Ausgabe bezüglich der Eingabe berechnet wird. Dabei gilt: $M_{Saliency} = \nabla_x F^y(x)$, wobei y die richtige Klasse repräsentiert und $F^y(x)$ die Ausgabe entsprechend y ist. Da jedoch nur der lokale Gradient bezüglich der Eingabe betrachtet wird, können aufgrund der hohen Nichtlinearität komplexer DNNs irreführende Ergebnissen erzeugt werden (Pan u. a. 2021). Eine Verbesserung stellt die IG Methode dar, die im Folgenden vorgestellt wird.

4.2 Fundamentale Axiome

Es ist sehr schwierig, Attributionsmethoden empirisch zu bewerten, da nicht nachvollziehbar ist, ob ein Fehler in der Attribution durch einen Fehler in der Methode oder durch ein Fehlverhalten des Modells entstanden ist. Aufgrund dessen stellten Sundararajan u. a. (2017) Axiome auf, die ihrer Meinung nach jede gute Attributionsmethode erfüllen muss. Da zum damaligen Stand keine bekannte Methode alle ihre Axiome erfüllen konnte, entwickelten sie auf der Basis dieser Axiome die IG Methode.

Axiom 1 *Sensibilität (a)*

Eine Attributionsmethode erfüllt ***Sensibilität (a)***, wenn für alle Eingaben x und Referenzwerte x' die sich in einem Merkmal unterscheiden gilt, dass wenn diese unterschiedliche Vorraussagen erzeugen, diesem Merkmal eine Attribution ungleich 0 zugesprochen werden muss.

Axiom 2 *Implementierungsinvarianz*

Zwei Netze heißen **funktionsäquivalent**, wenn sie für alle Eingaben die gleichen Ausgaben erzeugen, unabhängig ihrer Implementierung. Eine Attributionsmethode erfüllt die **Implementierungsinvarianz**, wenn sie für zwei **funktionsäquivalente** immer identische Attributionen erzeugt.

In ihrer Veröffentlichung zeigen Sundararajan u. a. (2017), dass *Saliency Maps* sowie viele andere populäre Attributionsmethoden mindestens eines dieser Axiome verletzten.

4.3 Definition von IG

Sei nun wieder F ein DNN zur Klassifikation, x eine Eingabe und x' eine *baseline*. Für Netze zur Bilderkennung sollte meist als *baseline* ein komplett schwarzes Bild verwendet werden. Nun wird ein gerader Pfad von x' nach x generiert und an allen Punkten des Pfades der Gradient berechnet. *Integrated Gradients* ergibt sich dann aus der Kumulierung dieser Gradienten. IG ist also das Pfadintegral der Gradienten entlang des Pfades von x' nach x . Der integrierte Gradient entlang der i^{ten} Dimension ist dann wie folgt definiert, wobei $\frac{\partial F(x)}{\partial x_i}$ der Gradient von $F(x)$ entlang der i^{ten} Dimension ist:

$$\text{IntegratedGradients}_i(x) := (x_i - x'_i) \times \int_{\alpha=0}^1 \frac{\partial F(x' + \alpha \times (x - x'))}{\partial x_i} d\alpha \quad (4.2)$$

(Sundararajan u. a. 2017). Die Methode erfüllt ein weiteres Axiom, welches bereits in der Vergangenheit von anderen Autoren definiert wurde:

Axiom 3 *Vollständigkeit*

Eine Attributionsmethode erfüllt die **Vollständigkeit**, wenn die Summe der Attributionen die Differenz zwischen der Ausgabe von $F(x)$ und dem Ausgangswert x' beträgt (Sundararajan u. a. 2017).

Denn im Falle, dass F nahezu überall differenzierbar ist, was für DNNs gegeben ist (Sundararajan u. a. 2017), gilt:

$$\sum_{i=1}^n \text{IntegratedGradients}_i(x) = (F(x) - F(x')) \quad (4.3)$$

Da IG **vollständig** ist, erfüllt es ebenfalls die **Sensibilität (a)**, denn Vollständigkeit impliziert Sensibilität (Sundararajan u. a. 2017). Ebenfalls erfüllt IG die **Implementierungsinvarianz**, denn die Attribution basiert ausschliesslich auf den Gradienten der Funktion, die vom Netz repräsentiert wird und hängt somit nicht von inneren Details des Netzes ab (Sundararajan u. a. 2017).

4.4 Definition Pfadmethoden

Da *integrated Gradients* nur einen bestimmten von vielen möglichen Pfaden zwischen x und x' verwendet, geben Sundararajan u. a. (2017) auch eine allgemeinere Definiton für Attributionsmethoden, die sie als *Pfadmethoden* bezeichnen.

Sei $\gamma = (\gamma_1, \dots, \gamma_n) : [0, 1] \rightarrow R^n$ eine glatte Funktion, die einen Pfad von x' nach x spezifiziert, also $\gamma(x') = 0$ und $\gamma(x) = 1$. Dann ist *path integrated gradients* das Pfadintegral von $\gamma(\alpha)$ für $\alpha \in [0, 1]$. Formel ist *path integrated gradients* entlang der i^{ten} Dimension ist dann wie folgt definiert:

$$\text{PathIntegratedGradients}_i^\gamma(x) := \int_{\alpha=0}^1 \frac{\partial F(\gamma(\alpha))}{\partial \gamma_i(\alpha)} \frac{\partial \gamma_i(\alpha)}{\partial \alpha} d\alpha \quad (4.4)$$

Alle Attributionsmethoden, die auf *path integrated gradients* basieren, sind Pfadmethoden. So ist beispielsweise IG die Pfadmethode des geraden Pfades $\gamma(\alpha) = x' + \alpha \times (x - x')$ für $\alpha \in [0, 1]$. Dabei ist interessant, dass IG aufgrund dessen weitere wünschenswerte Axiome aus dem Bereich der Pfadmethoden erfüllt (Sundararajan u. a. 2017).

Axiom 4 Sensibilität (b)

Wenn die Funktion, die von einem DNN implementiert wird, nicht (mathematisch) abhängig von einer Variable ist, dann ist die Attribution dieser Variable immer gleich null (Friedman 2004).

Dieses Axiom stellt eine natürliche Erweiterung des Axiom **Sensibilität (a)** dar (Sundararajan u. a. 2017).

Axiom 5 Linearität

Angenommen es existieren zwei Netze F_1 und F_2 sowie eine Linearkombination beider, die ein neues drittes Netz $F_3 = a \times F_1 + b \times F_2$ ergibt. Dann ist wünschenswert, dass die Attributionen für F_3 die gewichtete Summe der Attributionen für F_1 und F_2 , jeweils mit Gewichten a und b , darstellt (Sundararajan u. a. 2017).

Pfadmethoden sind die einzigen Attributionsmethoden, die alle 5 Axiome **Inmplementierungsinvarianz**, **Sensibilität (a + b)**, **Vollständigkeit**, und **Linearität** erfüllen (Friedman 2004) (Sundararajan u. a. 2017). IG ist zwar eine relativ neue Errungenschaft auf dem Gebiet der XAI, die Idee dahinter ist allerdings schon älter und stammt aus der Wirtschaftswissenschaft. Die Formel für IG entspricht der Berechnung des Aumann-Shapely Wertes, welcher ein Lösungskonzept für das spieltheoretische Problem der Kostenteilung darstellt (Aumann 1974) (Sundararajan u. a. 2017).

4.5 Verwendung von IG durch das Riemann-Integral

Für die praktische Anwendung kann das Integral von IG mithilfe des riemannschen Integrals wie folgt abgeschätzt werden.

$$\text{IntegratedGradients}_i^{approx} := (x_i - x'_i) \times \sum_{k=1}^m \frac{\partial F(x' + \frac{k}{m} \times (x - x'))}{\partial x_i} \times \frac{1}{m} \quad (4.5)$$

Wobei m die Anzahl der Aufteilungen des Integrals bestimmt (Sundararajan u. a. 2017). Weiterhin weisen die Autoren darauf hin, dass in den meisten Fällen eine Schrittzahl m im Bereich 20 bis 300 eine ausreichend gute Approximation liefert. Zur Überprüfung der Qualität kann das Ergebnis der Berechnung von Formel 4.3 mit der Summe der Attributionen verglichen werden. Weichen diese stärker voneinander ab als gewünscht, sollte m erhöht werden. In den Abbildungen 4.1 und 4.2 werden jeweils Beispiele für einen Pfad und eine Attribution durch IG für das Bild einer Katze aus dem Dogs vs. Cats Datensatz gezeigt. Dabei ist zu beobachten, dass die Katze insbesondere an der Mundpartie sowie ihren Schnurrhaaren und der Form ihrer Ohren erkannt wurde. Abbildung 4.3 zeigt einen Vergleich für unterschiedliche *baselines* x' . Zur Visualisierung der Attribution wird die Attributionsmaske berechnet, diese ist eine auf zwei Dimensionen reduzierte Version der ursprünglichen Attributionen.



Abbildung 4.1: Beispiel für einen geraden Pfad mit x' als schwarzem Bild und dem Bild einer Katze aus Dogs vs. Cats als x .

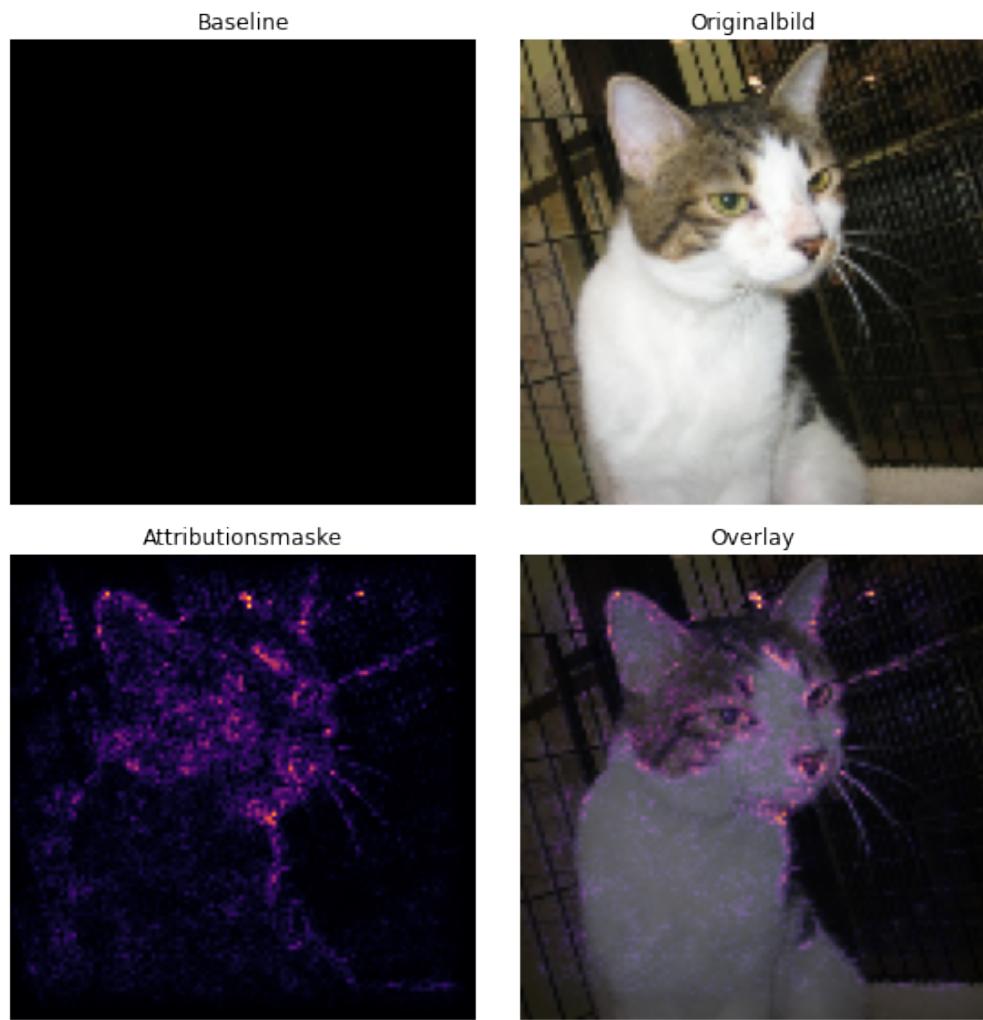


Abbildung 4.2: IG Attributionen für die Katze aus der vorherigen Grafik. Approximiert mit 240 Schritten.

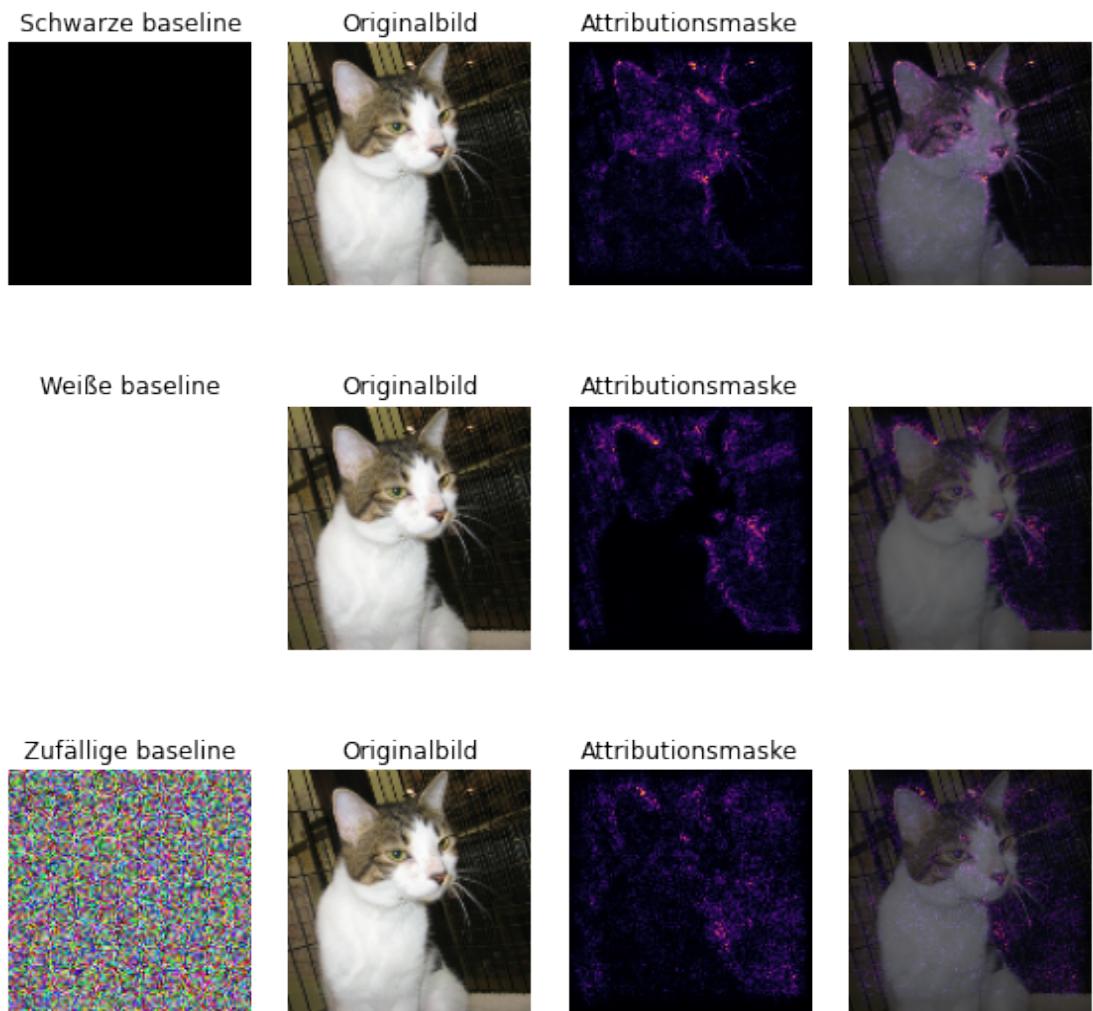


Abbildung 4.3: Vergleich der Auswirkung verschiedener *baselines* auf die Attributierung durch IG.

5

Adversarial examples

In diesem Kapitel werden *adversarial examples* (dt. feindliche Beispiele) sowie einige wichtige Forschungserkenntnisse der letzten Jahre vorgestellt. Szegedy u. a. (2013) beschreiben *adversarial examples* als Daten (beispielsweise Bilder), auf denen eine (für den Menschen) nicht wahrnehmbare, nicht zufällige Störung angewendet wird, wodurch die Entscheidung des Netzes eigenmächtig verändert werden kann (Szegedy u. a. 2013, S.2). Oder in anderen Worten: Es existiert ein Modell M und eine Eingabe C für die gilt $M(C) = y_{true}$. Dann ist es möglich ein *adversarial* A zu erzeugen, sodass $M(A) \neq y_{true}$ gilt (Kurakin u. a. 2016, S.2).

Im Folgenden wird dabei ausschließlich auf *adversarials* in Form von Bilddaten eingegangen, auch wenn mittlerweile andere Formen wie beispielsweise Angriffe auf *speech-to-text* Systeme mit *audio adversarials* möglich sind (Carlini und Wagner 2018). Außerdem werden im Rahmen dieser Arbeit nur gradientenbasierte Methoden zur Generierung von *adversarials* betrachtet. Der Vollständigkeit halber sei erwähnt, dass auch optimierungsbasierte Algorithmen wie der von Carlini und Wagner (2017), sowie entscheidungsbasierte Methoden wie beispielsweise Deepfool (Moosavi-Dezfooli u. a. 2015) existieren (Zhang u. a. 2021).

5.1 Goodfellow's Algorithmus

Die simpelste und schnellste Methode zum Erzeugen von *adversarial* Bildern ist die *Fast Gradient Sign Method* (FGSM), auch bekannt als Goodfellow's Algorithmus. Sie wird als schnell bezeichnet, da im Vergleich zu anderen Methoden keine iterativen Schritte notwendig sind und auch die Backpropagation nur einmal aufgerufen werden muss (Kurakin u. a. 2016, S.4). Der Algorithmus ist dabei wie folgt definiert.

Sei θ die Parameter eines Netzes, x die Eingabe des Netzes, y die Ziel-Variable für x (falls vorhanden) und $J(\theta, x, y)$ die Kostenfunktion des Netzes. Durch Linearisierung der Kostenfunktion über den aktuellen Wert von θ ergibt sich eine optimale, an die Maximumsnorm gebundene Störung:

$$\eta = \epsilon sign(\nabla_x J(\theta, x, y)) \quad (5.1)$$

Der Gradient $\nabla_x J(\theta, x, y)$ kann leicht über die oben erwähnte Backpropagation bestimmt werden (Goodfellow u. a. 2014). Abbildung 5.1 zeigt ein Beispiel.

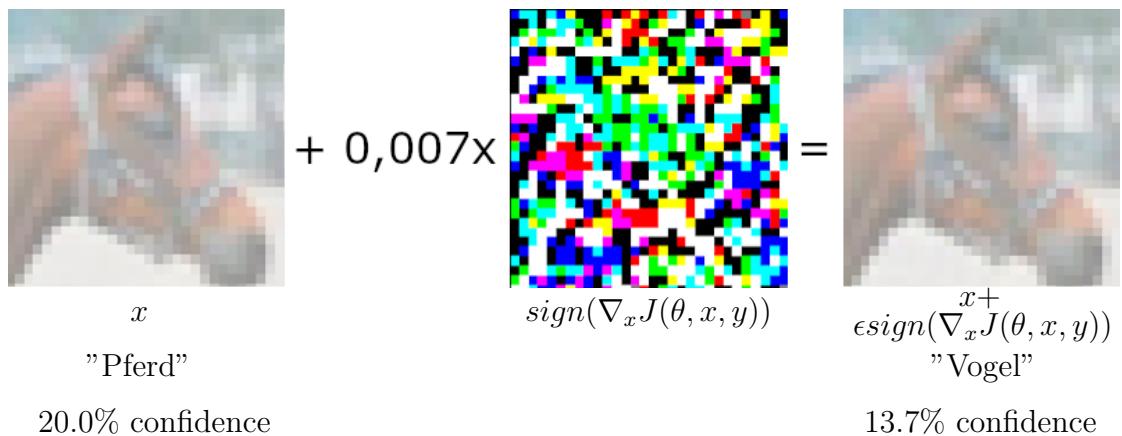


Abbildung 5.1: Beispiel für einen *adversarial* Angriff via FGSM auf ein selbst trainiertes CNN mit einem Bild aus dem CIFAR-10 Datensatz. Durch den Angriff verändert sich die Klassifikation von *Pferd* zu *Vogel*. Grafik inspiriert von (Goodfellow u. a. 2014, Abb. 1)

Die größte Stärke von Goodfellow's Algorithmus ist auch gleichzeitig seine größte Schwäche: seine Simplizität. Er stellt allerdings auch die Basis für deutlich stärkere iterative Angriffe dar. Eine gute Übersicht befindet sich in Kurakin u. a. (2016).

5.2 Übertragbarkeit von Adversarials

Der oben genannte Angriff und auch alle weiteren komplexeren Angriffsstrategien bergen ein entscheidendes Problem: Zum generieren des *adversarials* ist ein Zugang zum Netz und seinen Parametern notwendig, wodurch alle Angriffe effektiv zu White-Box-Angriffen werden. Wie Papernot u. a. (2016) erstmals in der Praxis zeigten, sind Black-Box-Angriffe jedoch trotzdem durchführbar. Sie sind sogar effizient möglich, da *adversarials übertragbar* sind: dies bedeutet, ein *adversarial*, das auf einem Netz einen erfolgreichen Angriff durchführt, wird mit hoher Wahrscheinlichkeit auf einem ähnlichen Netz ebenfalls für eine Fehlentscheidung sorgen. Die Idee ist also folgende: Es wird ein Ersatznetzwerk trainiert, von dem ausgegangen wird, dass es dem Netz, welches angegriffen werden soll, ähnelt (beispielsweise Bilderkennung für ImageNet (Deng u. a. 2009)). Nun können über dieses Ersatznetz *adversarials* generiert und mit diesen das Zielnetz angegriffen werden. Mit dieser Technik konnten Papernot u. a. (2016) beispielsweise gegen Netze von Google und Amazon Erfolgsquoten von je 88% und 96% erzielen (Papernot u. a. 2016, S.11).

5.3 Adversarials in der echten Welt

Die zweite gefährliche Eigenschaft von *adversarials* ist ihre Anwendbarkeit in der echten Welt. Kurakin u. a. (2016) zeigten, dass *adversarials*, die ausgedruckt und

dann mit einer Handykamera abfotografiert wurden, immer noch für eine Fehlklassifizierung sorgten. Gegen Ende ihrer Ausarbeitung zeigten sie in einem Experiment, dass sich diese Eigenschaft mit der oben erwähnten **Übertragbarkeit** kombinieren lässt. Dies sorgt theoretisch dafür, dass gefährliche praxisrelevante Angriffe auf KI in unserem Alltag möglich sind. Ein schwerwiegendes Beispiel stellt die Bilderkennung bei selbstfahrenden Autos dar.

5.4 Verteidigungsstrategien

Als Konsequenz ist es wichtig, geeignete Verteidigungsstrategien zu entwickeln. Aufgrund der Relevanz des Problems gibt es heutzutage zahlreiche Konzepte in Theorie und Praxis. Im Folgenden werden ausgewählte, relevante Beispiele erläutert.

5.4.1 Gradient masking

Als *gradient masking* definieren Papernot u. a. (2016) eine Gruppe von Methoden, die das Ziel haben, den Gradienten, der für den oben erläuterten FGSM Angriff essenziell ist, *unbrauchbar* zu machen. Der Effekt kann unbeabsichtigt auftreten oder absichtlich hervorgerufen werden (Boenisch u. a. 2021). Der Begriff *obfuscated gradients* (dt. verschleierte Gradienten) beschreibt das Szenario in dem *gradient masking* absichtlich durch Implementieren einer Verteidigungsstrategie auftritt (Athalye u. a. 2018). Für die *unbrauchbaren* Gradienten definieren Athalye u. a. (2018) drei verschiedene Szenarien:

Zerrüttete Gradienten sind solche, die nicht existieren oder inkorrekte Daten enthalten. Sie entstehen entweder gewollt durch Anwendung nicht differenzierbarer Operationen oder ungewollt durch numerische Instabilitäten (bspw. Rundungsfehler).

Stochastische Gradienten entstehen durch Anwendung von zufälligen Transformationen auf die Eingabe zur Testzeit.

Verschwindende oder **explodierende Gradienten** sind jeweils zu klein beziehungsweise zu groß, um für die Berechnungen während der Angriffe nützlich zu sein.

Ein Beispiel für die Effektivität von **stochastischen Gradienten** liefern Guo u. a. (2017), die in ihrem Gray-Box Szenario gute Ergebnisse erzielen konnten. Doch auch wenn sich durchaus positive Beispiele zeigen lassen (siehe Kapitel 7), ist das *gradient masking* leider keine verlässliche Verteidigungsstrategie, sondern vielmehr ein falsches Gefühl von Sicherheit. Beispielsweise konnte Papernot u. a. (2016) zeigen, dass ihre Black-Box-Angriffe das *gradient masking* überwinden konnten. Der Aspekt der fälschlichen Sicherheit wird nochmals dadurch verstärkt, dass die nicht-gradientenbasierten Angriffe nicht abgedeckt sind (Boenisch u. a. 2021).

5.4.2 Adversarial Training

Eine der beliebtesten Verteidigungsstrategien präsentierte Szegedy u. a. (2013) und Goodfellow u. a. (2014) als das *adversarial Training*. Die Robustheit des Netzes gegenüber *adversarials* wird gestärkt, indem dem Trainingsdatensatz bei jedem Trainingsschritt absichtlich *adversarials* untergemischt werden. Laut Yuan u. a. (2017) stellt sie nach heutigem Stand die effizienteste Methode dar (Boenisch u. a. 2021). Meist wird aufgrund ihrer Schnelligkeit die FGSM zur Generierung der zahlreichen *adversarials* für das *adversarial Training* eingesetzt (Madry u. a. 2017). Papernot u. a. (2016) beobachteten, dass ihr Black-Box Angriff durch *adversarial Training* mit höheren ϵ Werten signifikant an Erfolg verlor. Der Angriff, der bei einem *adversarial Training* mit $\epsilon = 0,15$ noch für eine Fehlerrate von 71,25% sorgte, brach nach einem Training mit $\epsilon = 0,3$ auf 10,3% ein. Andererseits beobachteten Madry u. a. (2017), dass für ihren Fall das Training mit FGSM lediglich zu einer Überanpassung führt und das Netz gegen ihren (deutlich stärkeren) Angriff nicht an Robustheit gewinnen konnte.

5.4.3 Defensive distillation

Distillation ist ein ML Konzept, erstmals vorgestellt von Hinton u. a. (2015), bei dem das Wissen eines bestehenden DNN verwendet wird, um ein neues Netz zu trainieren. Die Motivation ist oftmals das Wissen eines *großen* Netzes in ein *kleineres* zu übertragen, wodurch die Komplexität sinkt und damit beispielweise Rechenzeiten und Speicherverbrauch reduziert werden. Ein typischer Anwendungsfall wäre DNNs für Smartphones zu erzeugen (Papernot u. a. 2015). Auf dieser Basis entwickelten Papernot u. a. (2015) die *defensive distillation*, bei der das ursprüngliche Netz nicht genutzt wird, um ein einfacheres DNN zu trainieren, sondern eines mit der exakt gleichen Architektur, um Widerstandsfähigkeit anstelle von Kompression zu erzeugen. Durch den Prozess der *distillation* wird das Netz weniger abhängig von seinen ursprünglichen Trainingsdaten und verbessert seine Generalisierungsfähigkeit. Das stärkt die Robustheit, da die erzeugte Veränderung der Gradienten Angriffe erschwert - *defensive distillation* ist also eine Form von *gradient masking*. Während Papernot u. a. (2015) noch überragend gute Ergebnisse zeigten, fanden Papernot u. a. (2016) bereits heraus, dass *defensive distillation* ausschließlich im White-Box Fall gute Ergebnisse liefert. Diese Erkenntnis deckt sich mit den oben erwähnten Resultaten für Black-Box Angriffe und *gradient masking*.

5.5 Sind Angriffe auf KI unausweichlich?

Auch wenn es mittlerweile mehrere vielversprechende Verteidigungsstrategien gegen *adversarial* Angriffe gibt und die Forschung in den letzten Jahren große Fortschritte gemacht hat, gibt es zum aktuellen Zeitpunkt noch keine verlässliche, wirklich robuste Strategie. Meist wird eine vielversprechende neue Technik schnell mit einem neuen Angriff gekontert (Shafahi u. a. 2020). Aufgrund dessen stellen sich Shafahi u. a. (2020) die fundamentale Frage, ob überhaupt eine optimale

Verteidigung gegen *adversarials* gefunden werden kann. Schließlich zeigten Athalye u. a. (2018), dass vielversprechende Strategien wie (Guo u. a. 2017), (Jacob Buckman u. a. 2018) und weitere durchbrochen werden können. Mit Hilfe von hochdimensionaler Geometrie, genauer isoperimetrischer Ungleichungen, stellen Shafahi u. a. (2020) 4 Theoreme auf, die fundamentale Grenzen definieren. Sie garantieren die Existenz von *adversarials*, jedoch diskutieren die Autoren auch 4 Lösungsvorschläge wie diese Grenzen überwunden werden könnten. Einfach dargestellt sind das:

- 1) Verschieben der Grenzen durch Erhöhen der Mannigfaltigkeit der Daten durch Hinzufügen von gleichmässigem Rauschen. Wird beispielsweise beim *adversarial Training* bereits angewendet (Madry u. a. 2017).
- 2) Hinzufügen einer „**don’t know**“-Klasse, durch die das Modell seine Unsicherheit ausdrücken kann. Dadurch wird zwar die Anzahl der falschen Klassifikationen verringert, aber die der nicht richtigen Entscheidungen wird nicht weniger.
- 3) Unter der Annahme, dass das Reduzieren der Dimensionen der Eingabe die Werte der Wahrscheinlichkeitsdichtefunktion nicht wesentlich erhöht, könnte das Arbeiten mit Daten in niedrigeren Dimensionen die Robustheit erhöhen.
- 4) Selbst wenn *adversarials* nie aufhören zu existieren, könnten sie trotz allem verhindert werden durch Maßnahmen, die ihre Berechnung so stark erschweren, dass sie nicht mehr *effizient* sind. Also die gleiche Verteidigungsstrategie wie in der modernen Kryptographie.

Auf die ursprünglich gestellte Frage finden Shafahi u. a. (2020) keine genaue Antwort. Jedoch legt ihre Forschung nahe, dass es noch ein sehr weiter Weg zu wirklich sicheren ML Modellen ist.

6

Experimente

Im Folgenden werden die im Rahmen dieser Ausarbeitung auf den Datensätzen aus Kapitel 3 durchgeführten Experimente erläutert, sowie die dafür verwendeten Netze vorgestellt.

6.1 Architektur der CNNs

In diesem Abschnitt wird die Architektur der verwendeten Netze vorgestellt. Der Fokus lag bei der Auswahl der CNNs weniger auf hoher Genauigkeit. Stattdessen sollten simple Netze gefunden werden, die alle vorgestellten Datensätze verarbeiten können, um eine gute Vergleichbarkeit zu gewährleisten.

6.1.1 Eigene Netze

Für die selbst entwickelten Netze wurde eine simple grundlegende Architektur ausgewählt, die mit leicht unterschiedlichen Trainingsparametern auf allen Datensätzen eingesetzt wurde. Diese besteht aus drei **convolutional layern** und einem **fully-connected layer**. Als Aktivierungsfunktion wurde **ReLU** verwendet und im output layer **Softmax**. Als Lossfunktion für das Training wurde **Sparse-CategoricalCrossentropyLoss** und zur Generierung der *adversarial* **CategoricalCrossentropy** verwendet. Zur Optimierung der Netzwerkparameter wurde **Adam** mit einer Lernrate von 0.001 verwendet. Alle diese Bausteine stammen aus der Keras API und wurden über Tensorflow verwendet. Einen Überblick schafft Tabelle 6.2. Die Anzahl der Trainingsepochen sind für jedes Netz individuell angepasst, um *overfitting* zu vermeiden. Die CNNs für Fashion-MNIST und CIFAR-10 wurden auf dem gesamten Trainingsdatensatz für jeweils 10 und 5 Epochen trainiert. Aus Speichergründen musste der Dogs vs. Cats Datensatz auf 5000 Bilder reduziert werden, die jeweils auf eine Auflösung von 112×112 skaliert wurden. Das zugehörige Netz wurde dann auf diesen 5000 Bildern für 4 Epochen trainiert. Da das CNN für Fashion-MNIST eine deutlich höhere Genauigkeit erreicht als die anderen beiden, wurde zusätzlich ein *naives* Netz für Fashion-MNIST mit nur 1000 Bildern trainiert. In der folgenden Tabelle 6.1 werden die Genauigkeiten der Modelle zusammengefasst.

Netz/Datensatz	Trainingsbilder	Epochen	Testbilder	Genauigkeit
Fashion-MNIST	60.000	5	10.000	91,9%
Fashion-MNIST <i>naiv</i>	1000	10	10000	81,2%
CIFAR-10	50.000	10	10.000	78,4%
Dogs vs. Cats	5000	4	5000	81,7%

Tabelle 6.1: Die verschiedenen CNNs mit ihrer Genauigkeit und ihren Trainingsparametern im Überblick.

CNN layer	Filter/Neuronen	Kernel Größe	Aktivierungsfunktion
Convolutional layer	32	3x3	ReLU
Max pooling	32	2x2	-
Convolutional layer	64	3x3	ReLU
Max pooling	64	2x2	-
Convolutional layer	64	3x3	ReLU
Flatten	-	-	-
Fully connected layer	64	-	-
Output layer	10	-	Softmax

Tabelle 6.2: Architektur der selbst trainierten CNNs im Überblick.

6.1.2 Vortrainierte Netze

Neben den selbst trainierten Netzen sollten auch vortrainierte Netze bekannter Architekturen zum Einsatz kommen. Dazu sollten diese Netze per **transfer learning** und **fine-tuning** an die neuen Daten angepasst werden. Dabei kam es leider zu einigen Komplikationen. Als Erstes sollte das MobileNetV2 (Sandler u. a. 2018) verwendet werden. Das Netz produzierte leider nur für Daten im ImageNet (Deng u. a. 2009) Format ($224 \times 224 \times 3$) brauchbare Ergebnisse. Während dies anfangs für Dogs vs. Cats gut funktionierte, stellte sich schnell heraus, dass der Datensatz aus Speichergründen herunterskaliert werden musste. Aus diesem Grund wurde stattdessen das CNN InceptionV3 (Szegedy u. a. 2015) getestet. Dieses produzierte die besten Ergebnisse für Dogs vs. Cats und bereitete keine Probleme nach dem Skalieren. Es konnte eine Genauigkeit von 94,4% erreichen. CIFAR-10 und Fashion-MNIST funktionierten jedoch nicht, da die Bilder zu klein für das Netz sind. Als letztes wurde versucht, eine allgemeine Lösung in VGG-16 (Simonyan und Zisserman 2014) zu finden, aufgrund der hohen Komplexität des Netzes kam es jedoch zu hohen Trainingszeiten und hohem Speicherverbrauch, wodurch das Netz praktisch unbrauchbar wurde.

6.2 Experimente und Ziele

Als Nächstes werden die zuvor erläuterten CNNs gemeinsam mit den Techniken aus den Kapiteln 4 und 5 für die Ausführung verschiedener Experimente genutzt.

6.2.1 Adversarial Angriffe mit FGSM

Im ersten Schritt werden mit Hilfe von Goodfellow's Algorithmus *adversarials* generiert, um die Robustheit der Netze zu prüfen. Das Experiment wird mit 1000 Bildern pro Datensatz beziehungsweise aus technischen Gründen 1024 für Dogs vs. Cats ausgeführt. Für jedes Bild werden jeweils 9 *adversarials* generiert. Der Wertebereich für die ϵ -Stärke ist folgender: $\{0, 0.007, 0.01, 0.015, 0.05, 0.07, 0.1, 0.15, 0.25\}$. Anschliessend wird die Fehlerrate, also die prozentuale Darstellung der erfolgreichen Angriffe, evaluiert.

6.2.2 Analyse der IG Attributionen von adversarials

Der zweite Schritt verbindet nun die beiden großen Themen dieser Ausarbeitung: *Integrated Gradients* und *adversarials*. Für jedes *adversarial* Bild aus dem vorigen Schritt sowie das Original wird IG angewendet und die Attributionsmaske gebildet. Für alle Bilder wird dabei ein schwarzes Bild als *Baseline* und eine Schrittzahl von 240 verwendet. Mithilfe der Attributionsmasken können dann die Attributions zwischen dem Originalbild und unterschiedlich stark gestörten *adversarials* verglichen werden, in der Hoffnung neue Erkenntnisse über *adversarials* oder IG gewinnen zu können.

6.2.3 Vergleich mit zufällig generierter Störung

Zuletzt soll ein Vergleich zwischen *adversarials* und Bildern mit *normaler* Störung gezogen werden, um die Stärke des Unterschiedes zu evaluieren. Dazu kommt das gleiche Experiment wie in Schritt zwei zum Einsatz, aber diesmal für Bildern, die mit einer zufällig generierten Störung im gleichen Wertebereich der *adversarial* Perturbation gestört werden. Die Werte werden dann einander gegenüber gestellt, um ein klareres Bild über den Unterschied zwischen *adversarials* und zufälligem Rauschen zu gewinnen.

Ergebnisse und Erkenntnisse

Nachfolgend werden die Ergebnisse der vorher definierten Experimente vorgestellt und ausgewertet. Letztlich werden diese diskutiert, die gewonnenen Erkenntnisse ausformuliert und geprüft, ob die Ziele dieser Abschlussarbeit erreicht wurden.

7.1 Robustheit der Netze

Wie in Kapitel 6.2.1 beschrieben, wurde die Robustheit der Netze durch *adversarial* Angriffe mit FGSM auf die Probe gestellt. Ein frühzeitig interessantes Ergebnis war, dass es beim InceptionV3 Netz mit Cats vs Dogs zu *gradient masking* kam. Leider ist nicht nachvollziehbar, ob dieses Verhalten zufällig auftritt oder beabsichtigt war. Da jedoch viele Autoren der Inception Architektur ebenfalls am Großteil der in Kapitel 5.4 vorgestellten Arbeiten beteiligt waren, ist durchaus annehmbar, dass das Hervorrufen **zerrütteter Gradienten** beabsichtigt war. Abbildung 7.2 gibt einen Einblick in die verschiedenen erreichten Perturbationen. Eine Visualisierung der errechneten Fehlerraten ist in Abbildung 7.1 zu sehen. Zusätzlich ist ein Vergleich zur Fehlerrate von verrauschten Bildern abgebildet.

Besonders auffällig für die *adversarials* ist der drastische Unterschied zwischen den beiden CNNs für Fashion-MNIST. Während das *naive* zwar von Beginn an erheblich schlechter klassifiziert, ist der Fehleranstieg für *adversarials* deutlich langsamer als beim *genauereren* Netz, auch wenn sich am Ende beide bei einer Fehlerrate von über 95% treffen. Dieser Effekt ist höchstwahrscheinlich auf *overfitting* zurückzuführen, da dem *naive* Netz deutlich weniger Trainingsdaten zur Verfügung standen. Ein ähnlicher Effekt ist auf der rechten Seite beim Rauschen zu sehen, wieder steigt die Fehlerrate des *naiven* Netzes deutlich langsamer, aber trotzdem treffen sich beide am Ende bei einer ungefähr gleichen Anzahl Missklassifizierungen.

Der Dogs vs. Cats Datensatz war, wie in Kapitel 3 erwähnt, von besonderem Interesse, da sich die Frage stellte wie *schwierig* es sei, ein CNN, welches nur 2 Klassen kennt, zur Missklassifikation zu bringen. Überraschend war das Ergebnis, dass das Netz bei einer ϵ -Stärke von gerade einmal 0,015 bereits eine nahezu hunderprozentige Fehlerrate erreicht. Wie in Abbildung 7.2 (a) zu sehen, ist diese Störung mit bloßem Auge unmöglich wahrzunehmen. Andererseits hat das Rauschen kaum

Auswirkung auf die Fehlerrate des Dogs vs. Cats Klassifikators, sie steigt nur um vergleichsweise geringe 13% und fällt für $\epsilon = 0,07$ und $\epsilon = 0,1$ sogar auf tiefere Werte als ohne Störung. Der geringe Fehler des InceptionV3 für Dogs vs. Cats ist auf das oben erwähnte *gradient masking* zurückzuführen. Der Effekt steigt mit der ϵ -Stärke, was vermuten lässt, dass stärkere *adversarials* sich leichter aufspüren lassen. Diese Annahme deckt sich mit den Ergebnissen von (Papernot u. a. 2016) zu *adversarial* Training die in Kapitel 5.4.2 vorgestellt wurden. Für die Störung durch Rauschen war das InceptionV3 Netz jedoch sogar anfälliger als das simple selbst trainierte CNN. Da nur sehr wenige *adversarials* generiert wurden, wird das Netz im Verlauf der weiteren Experimente nicht mehr betrachtet.

Lediglich für CIFAR-10 stieg der Fehler für beide Störungsarten stark an, dieser erreicht aber trotzdem für das Rauschen einen geringeren Wert von 82% im Vergleich zur Fehlerrate für *adversarials* von 97%. Der beidseitig starke Anstieg lässt sich vermutlich auf die Komplexität des Datensatzes zurückführen, da die ursprünglich komplexen Bilddaten sehr viele Informationen enthielten, welche dann in einen sehr kleinen Bereich von 32×32 skaliert wurden. Aus diesem Grund ist CIFAR-10 trotz seiner geringen Größe oftmals ein komplexes Problem in der Bilderkennung. Deshalb erreichte das CNN für CIFAR-10 auch die von Anfang an geringste Genauigkeit.

Zusammenfassend lässt sich aus der Beobachtung schließen, dass der Unterschied zwischen *adversarials* und einfachem Rauschen immens ist. Für den Großteil der vorgestellten Netze steigt die Fehlerrate unter geringem Rauschen nur leicht an und für 4/5 Netzen nur auf einen Wert von unter 40%. Andererseits sorgen die *adversarials* bei allen selbst trainierten CNNs mit einer zwar schon erkennbaren aber immer noch geringen ϵ -Stärke eine Fehlerrate von über 95%.

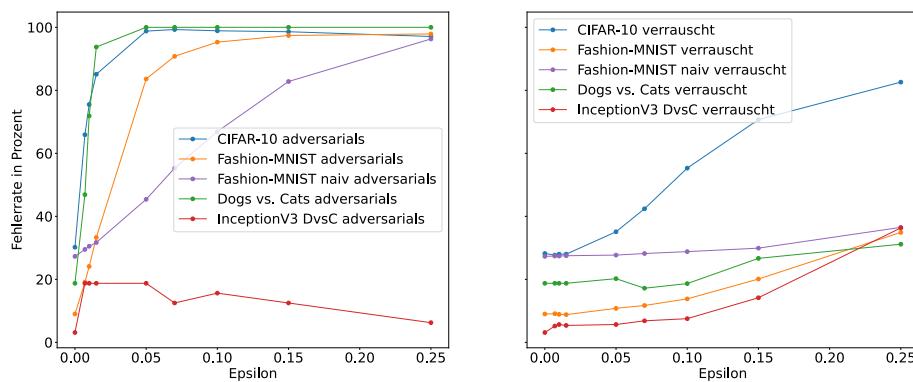


Abbildung 7.1: Vergleich der Fehlerraten der Klassifizierung für FGSM *adversarials* und Störung der Bilder durch Rauschen nach ϵ -Stärke.

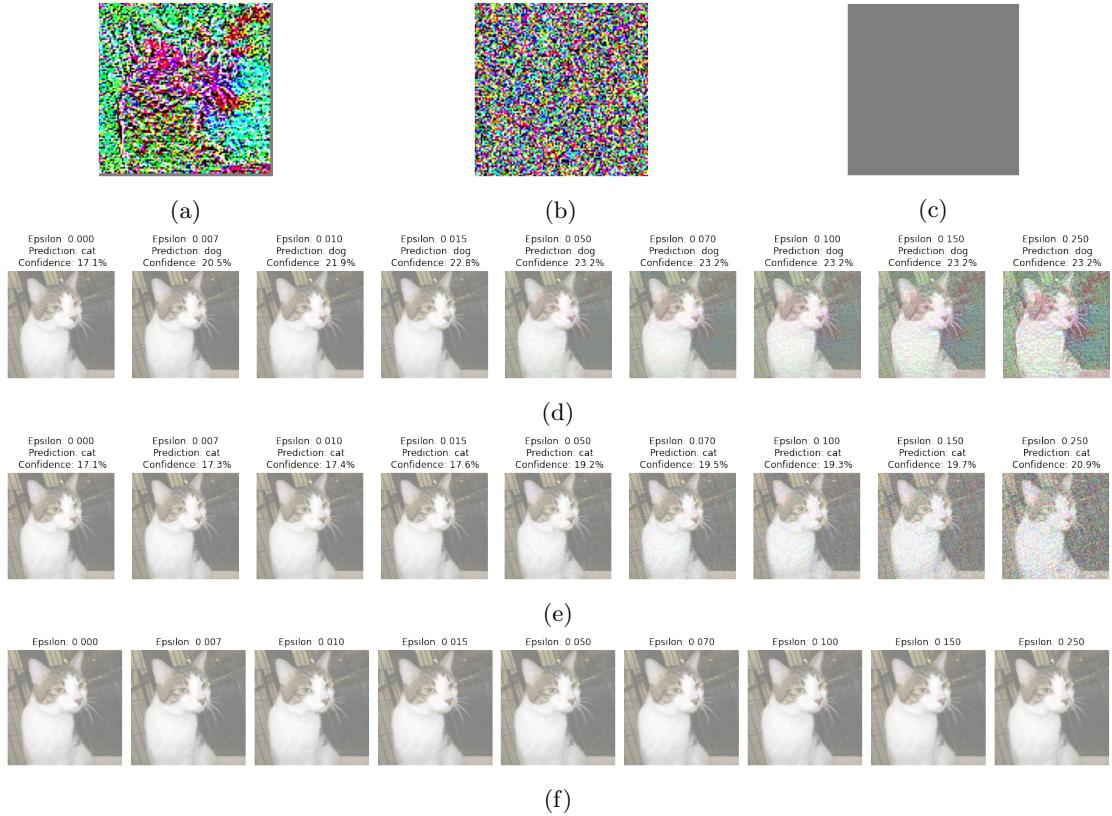


Abbildung 7.2: Die Grafiken (a), (b) und (c) zeigen jeweils die generierten Störungen für das Bild der Katze. Links ist das Ergebnis von FGSM auf dem selbst trainierten Netz, in der Mitte die zufällig generierte Störung und rechts das durch *gradient masking* verhinderte *adversarial* von InceptionV3 zu sehen. Grafik (d) zeigt ein Beispiel aus dem Dogs vs. Cats Datensatz für die 9 auf dem selbst trainierten Modell generierten *adversarials* nach ϵ -Stärke. Grafik (e) hingegen visualisiert den Effekt des Rauschens. Grafik (f) zeigt ein Beispiel für das Auftreten von *gradient masking* durch das InceptionV3 Netz.

7.2 IG Attributionen für perturbierte Bilder

Wie in Kapitel 6.2.2 und 6.2.3 beschrieben, werden im Folgenden die Attributions durch *Integrated Gradients* für unterschiedlich veränderte Bilder analysiert. Dazu werden zuerst für alle Bilder die Attributionsmasken gebildet. Anschließend wird die Beziehung zwischen den Masken für das jeweilige Originalbild und die unterschiedlich stark gestörten Gegenstücke mithilfe verschiedener statistischer Maßzahlen interpretiert. Dazu werden der *mean squared error* (MSE, dt. mittlere quadratische Abweichung), die Standardabweichung (STD) und der Korrelationskoeffizient nach Pearson verwendet. In Abbildung 7.3 sind Beispiele für die Attributionen jeweiliger Störungen zu sehen.

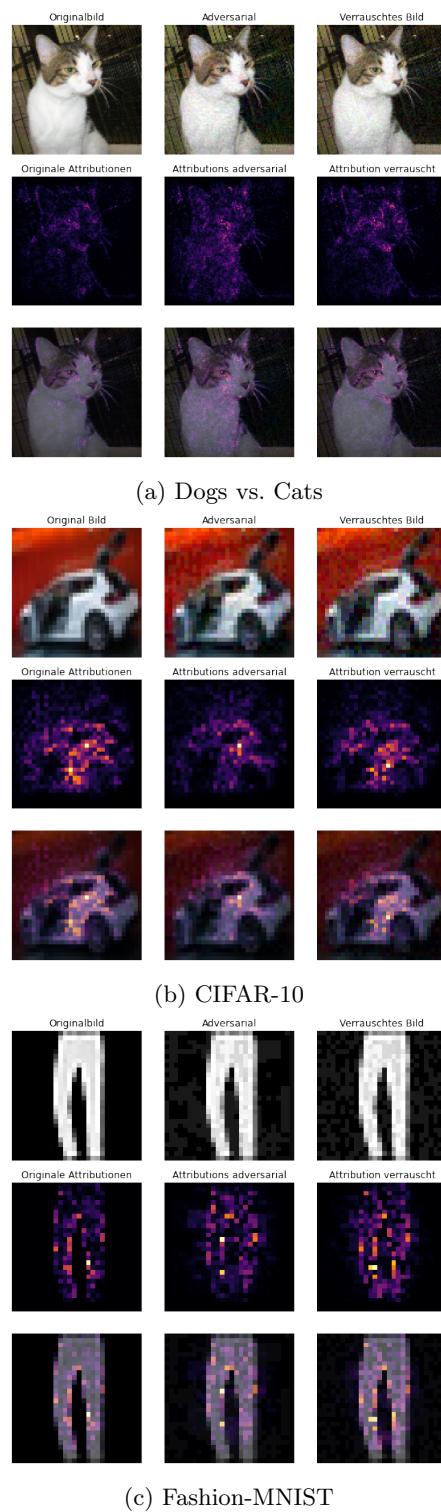


Abbildung 7.3: Beispiele für die Attributionsmasken verschiedener Perturbationen auf Bildern aus den 3 Datensätzen.

Auf den ersten Blick ist zu erkennen, dass die Attributionen sich durch die Störung verändern. Teilweise auch sehr stark wie in Abbildung 7.3 (a). Jedoch wird auch deutlich, dass die Grundstruktur der Attribution sich nicht verändert, auch nicht durch *adversarials*. Die laut IG wichtigsten Merkmale bleiben immer gleich, so sind diese für die Katze aus Abb. 7.3 (a) beispielsweise die Form des Kopfes inklusive der Ohren, die Schnurrhaare und die Reflektion im Auge. Um zu prüfen, wie unterschiedlich die Attributionen wirklich sind, werden nun der MSE und die STD auf der Differenz zwischen der Maske des Originalbildes und einer Maske eines gestörten Bildes berechnet. Diese werden pro ϵ -Stärke aufsummiert und der Durchschnitt gebildet. Die Ergebnisse sind in Abbildung 7.4 und 7.5 zu sehen. Pearsons Korrelationskoeffizient ist in Abbildung 7.6 dargestellt.

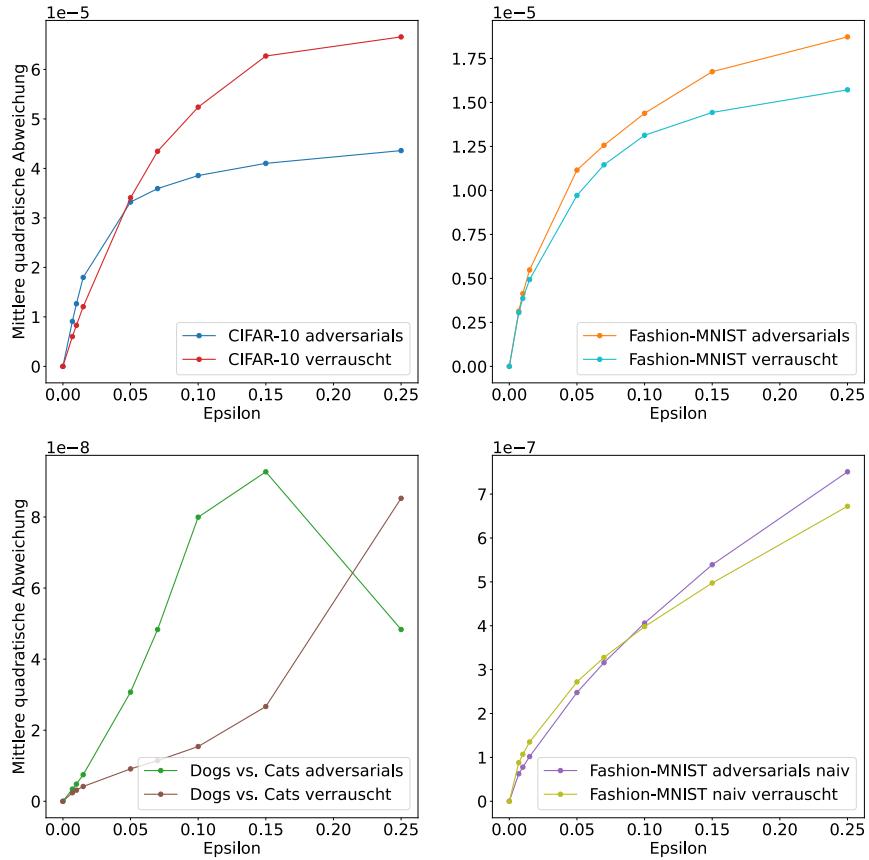


Abbildung 7.4: Durchschnittlicher MSE der Differenz zwischen der Attributionsmaske des Originalbildes und der des zugehörigen ϵ *adversarials* pro Datensatz.

Der MSE, auch empirische Varianz genannt, ist ein Streuungsmaß der deskriptiven Statistik, welches einen Einblick in die Streuung der einzelnen Werte $x_i, i \in \{1, 2, \dots, n\}$ um den empirischen Mittelwert \bar{x} liefert (Bourier 2018, S. 97). Mathematisch ist er nach Kosfeld u. a. (2016, S. 311) wie folgt definiert:

$$MSE = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2 \quad (7.1)$$

Die empirische Standardabweichung hingegen ist die Quadratwurzel der empirischen Varianz (Bourier 2018, S. 97), also:

$$STD = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2} \quad (7.2)$$

Sie liefert ebenfalls einen Einblick in die Streubreite der Daten um ihren Mittelwert. Im Gegensatz zum MSE befindet die STD sich in der gleichen Dimension wie die Eingabedaten, wodurch auch die entsprechende Einheit erhalten bleibt. Da die beobachteten Attributionen jedoch keine Einheit haben, ist dies nicht weiter wichtig.

Für die Differenzen der Attributionsmasken ergeben sich sowohl für den MSE als auch die STD ausschließlich Werte im mehrstelligen Nachkommabereich. Auf den ersten Blick könnte die geringe Abweichung ein Indiz dafür sein, dass die Differenz der Attributionsmasken ebenfalls entsprechend gering ist. Diese Annahme wurde, wie in Abbildung 7.6 zu sehen, durch Beobachtung des Wertebereiches der *adversarial* Attributionsmasken für CIFAR-10 Bilder geprüft.

Die dadurch gewonnenen Daten lassen erkennen, dass die Attributionen generell in diesem niedrigen Wertebereich liegen. Die Attributionen verändern sich also auffällig durch die Störung, aber die Unterschiede kommen lediglich durch die Veränderung der Pixel zustande und nicht aufgrund der Eigenschaften von *adversarials*, was gerade im höheren ϵ Bereich deutlich wird, da dort die Abweichungen für die zufällige Störung meist höher sind.

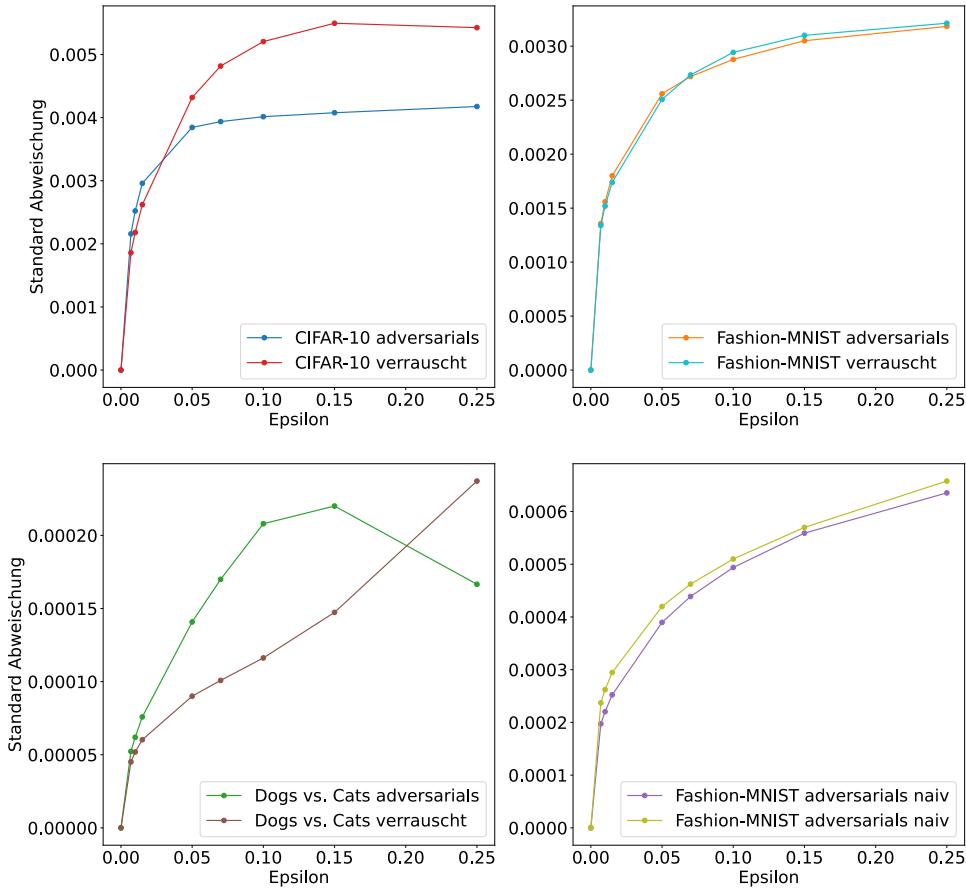


Abbildung 7.5: Durchschnittliche Standardabweichung der Differenz zwischen der Attributionsmaske des Originalbildes und der des zugehörigen ϵ *adversarials* pro Datensatz.

Pearsons Korrelationskoeffizient r trifft eine Aussage über den linearen Zusammenhang zweier Datensätze und ist wie folgt definiert (Bourier 2018, S. 211):

$$r = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} * \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}} \quad (7.3)$$

Wobei gilt, dass r immer zwischen -1 und 1 liegt, also $r \in [-1; +1]$. Dabei sagt ein Wert von 1 aus, dass die Daten stark positiv korrelieren. Folglich steigen mit größer werdenden Werten für x auch die Werte für y . Ein Wert von -1 bedeutet eine stark negative Korrelation. Im Umkehrschluss bedeutet dies, dass wenn die Werte für x steigen die Werte für y fallen. Ein Wert von $r \approx 0$ bedeutet, dass kein linearer Zusammenhang zwischen den Daten vorliegt (Bourier 2018, S. 213).

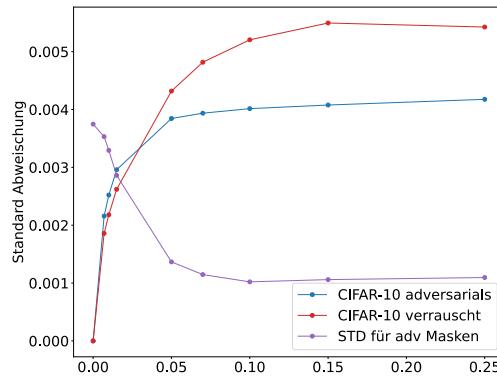


Abbildung 7.6: Die neue violette Kurve beschreibt den Verlauf der Standardabweichung allein für die Attributionsmasken der *adversarials*. Sie kommt zustande durch den Durchschnitt der Werte über 100 Bilder, also insgesamt 900 Masken.

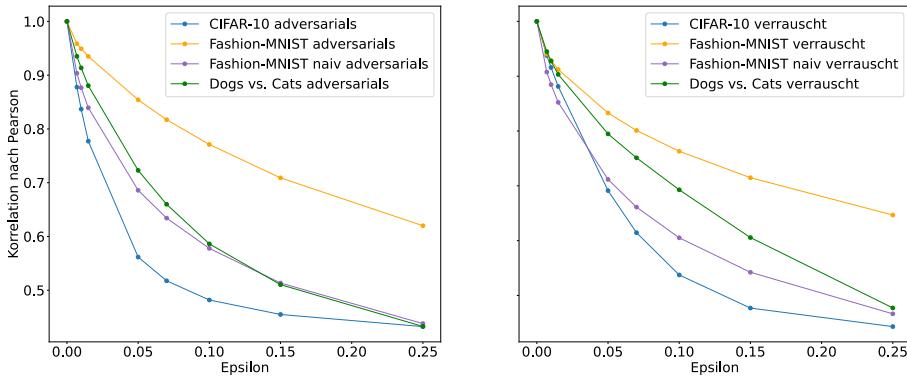


Abbildung 7.7: Durchschnittliche Korrelation nach Pearson zwischen der Attributionsmaske des Originalbildes und der des zugehörigen ϵ *adversarials*.

Für die Berechnung im Rahmen des Experiments war x die Attributionsmaske der Originalbildes und y die des entsprechenden veränderten Bildes .

Anfangs sind für alle Daten wie zu erwarten die Korrelationskoeffizienten = 1, denn für $\epsilon = 0$ sind die verglichenen Daten noch genau gleich. Nun ist zu beobachten, dass mit stärkerer Störung die Korrelation der Attributionen immer weiter abfällt und für die meisten Datensätze sich der Null annähert. Auch dieses Verhalten war durchaus zu erwarten, schließlich werden die Daten mit steigender ϵ -Stärke immer unterschiedlicher, woraus logischerweise eine geringere Korrelation folgen sollte. Interessant zu beobachten ist jedoch der deutlich schwächere Abfall für das Fashion-MNIST CNN. Die Attribution durch IG scheint sich bei diesem Netz besonders wenig durch geringe Störfaktoren beeinflussen zu lassen. Außerdem interessant ist der vergleichsweise geringe Unterschied im Verlauf der Kurven für

adversarials und zufällige Störung. Aus dieser Beobachtung lässt sich gemeinsam mit dem ebenfalls geringen Unterschied der Störungsvarianten für die Streuungsmaße schließen, dass *adversarials* keinen besonderen Einfluss auf die IG Methode haben und ihr Effekt in diesem Falle dem von klassischem Rauschen ähnelt. Daraus lässt sich für die Ziele von Kapitel 6.2.2 schließen, dass an dieser Stelle durch IG leider keine neuen Erkenntnisse über *adversarials* gewonnen werden konnten. Ebenfalls liefern *adversarials* im hier gezeigten Anwendungsfall leider keine neuen Erkenntnisse über *Integrated Gradients*, da sie die Methode nicht anders beeinflussen als herkömmliches Rauschen. In der Fachliteratur gab es allerdings in jüngster Zeit bereits Ausarbeitungen, die durch die Kombination dieser Techniken viel erreichen konnten. So formulierten beispielsweise Wang u. a. (2021) zwei Algorithmen, die mithilfe von IG noch stärkere und unaufälligere *adversarial examples* generieren konnten als bisher. Andererseits präsentierte Pan u. a. (2021) eine Methode namens *Adversarial Gradient Integration* (AGI), bei der zur Generierung des Pfades keine *baselines*, sondern *adversarials* Verwendung finden. Dadurch konnte das Problem gelöst werden, dass die Qualität der Attribution maßgeblich von der menschlichen Entscheidung des Bestimmens einer *baseline* abhängt und gleichzeitig die Qualität der Attribution oftmals verbessert werden.

Zusammenfassung und Ausblick

In dieser Arbeit wurden zwei der wichtigsten offenen Forschungsgebiete im Bereich des maschinellen Lernens, Verständnis und Sicherheit, vorgestellt. Dazu wurden als erstes eine Einführung in die Grundlagen des *Deep Learning* und der *explainable AI* gegeben sowie forschungsrelevante Datensätze vorgestellt. Anschliessend wurde das Konzept der Attributionsmethoden zum Stärken des Verständnisses zwischen DNN und Mensch kurz erläutert und dann eine wichtige Errungenschaft dieses Bereichs, die *Integrated Gradients* Methode, im Detail vorgestellt. Danach wurde die andere Seite dieser Arbeit beleuchtet und ein Einblick in die *adversarial examples* gegeben. Nach einer kurzen Definition wurden Goodfellow's Algorithmus zur Generierung von *adversarials*, sowie interessante Eigenschaften dieser und einige relevante Verteidigungsstrategien vorgestellt.

Anschließend wurden trotz einiger Schwierigkeiten mehrere *Convolutional Neural Networks* auf den vorgestellten Datensätzen trainiert und mit *adversarials* angegriffen. Daraufhin wurden diese mit *Integrated Gradients* analysiert, um möglicherweise neue Erkenntnisse gewinnen zu können. Außerdem wurde analysiert, ob *adversarials* eine signifikant andere Auswirkung auf die *Integrated Gradients* Methode und die Robustheit von Modellen haben, als zufällig generiertes Rauschen. Es kam zum Schluss, dass *adversarials* wie zu erwarten die Robustheit einen Netzes deutlich stärker gefährden als zufälliges Rauschen, sowie dass *overfitting* CNNs noch anfälliger für *adversarials* macht. Im zweiten Teil ergab sich, dass *adversarials* keinen speziellen Einfluss auf *integrated Gradients* haben. Ansonsten konnten leider keine besonderen Erkenntnisse durch die Kombination der Techniken erlangt werden. Es wurde aber auf aktuelle Forschungsergebnisse verwiesen die zeigen, dass durch genau diese Kombination große Fortschritte auf beiden Seiten bereits erreicht werden konnten.

Literatur

- [1] Anish Athalye, Nicholas Carlini und David Wagner. *Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples*. 2018. URL: <http://arxiv.org/pdf/1802.00420v4>.
- [2] Robert J. Aumann. *Values of Non-Atomic Games*. Princeton Legacy Library. Princeton, N.J: Princeton University Press, 1974. ISBN: 9781400867080. DOI: [10.1515/9781400867080](https://doi.org/10.1515/9781400867080).
- [3] Bengio und LeCun. “Convolutional Networks for Images, Speech, and Time-Series”. In: (1997).
- [4] Yoshua Bengio und Yann LeCun. “Scaling learning algorithms towards AI”. In: 2007.
- [5] Franziska Boenisch, Philip Sperl und Konstantin Böttinger. *Gradient Masking and the Underestimated Robustness Threats of Differential Privacy in Deep Learning*. 2021. URL: <http://arxiv.org/pdf/2105.07985v1>.
- [6] Günther Bourier. *Beschreibende Statistik*. Wiesbaden: Springer Fachmedien Wiesbaden, 2018. ISBN: 978-3-658-21485-2. DOI: [10.1007/978-3-658-21486-9](https://doi.org/10.1007/978-3-658-21486-9).
- [7] Nicholas Carlini und David Wagner. *Audio Adversarial Examples: Targeted Attacks on Speech-to-Text*. 2018. URL: <http://arxiv.org/pdf/1801.01944v2>.
- [8] Nicholas Carlini und David Wagner. “Towards Evaluating the Robustness of Neural Networks”. In: *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, S. 39–57. ISBN: 978-1-5090-5533-3. DOI: [10.1109/SP.2017.49](https://doi.org/10.1109/SP.2017.49).
- [9] Yves Chauvin. *Backpropagation: Theory, Architectures, and Applications*. Developments in Connectionist Theory Series. Hoboken: Taylor and Francis, 2013. ISBN: 9780805812596. URL: <http://gbv.eblib.com/patron/FullRecord.aspx?p=1122907>.
- [10] Jia Deng u. a. “ImageNet: A large-scale hierarchical image database”. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009, S. 248–255. ISBN: 978-1-4244-3992-8. DOI: [10.1109/CVPR.2009.5206848](https://doi.org/10.1109/CVPR.2009.5206848).
- [11] Finale Doshi-Velez und Been Kim. *Towards A Rigorous Science of Interpretable Machine Learning*. 2017. URL: <http://arxiv.org/pdf/1702.08608v2>.
- [12] Eric J. Friedman. “Paths and consistency in additive cost sharing”. In: *International Journal of Game Theory* 32.4 (2004). ISSN: 0020-7276. DOI: [10.1007/s001820400173](https://doi.org/10.1007/s001820400173).
- [13] Ian Goodfellow, Yoshua Bengio und Aaron Courville. *Deep Learning*. MIT Press, 2016.
- [14] Ian Goodfellow, Jonathon Shlens und Christian Szegedy. *Explaining and Harnessing Adversarial Examples*. 2014. URL: <http://arxiv.org/pdf/1412.6572v3>.
- [15] Google, Hrsg. *AI Explainability Whitepaper*. CERRE, 2019. URL: https://cerre.eu/wp-content/uploads/2020/07/ai_explainability_whitepaper_google.pdf.

-
- [16] Chuan Guo u. a. *Countering Adversarial Images using Input Transformations*. 2017. URL: <http://arxiv.org/pdf/1711.00117v3>.
 - [17] Geoffrey Hinton, Oriol Vinyals und Jeff Dean. *Distilling the Knowledge in a Neural Network*. 2015. URL: <http://arxiv.org/pdf/1503.02531v1>.
 - [18] Jacob Buckman u. a. “Thermometer Encoding: One Hot Way To Resist Adversarial Examples”. In: *ICLR*. 2018.
 - [19] Reinhold Kosfeld, Hans Friedrich Eckey und Matthias Türck. *Deskriptive Statistik*. Wiesbaden: Springer Fachmedien Wiesbaden, 2016. ISBN: 978-3-658-13639-0. DOI: [10.1007/978-3-658-13640-6](https://doi.org/10.1007/978-3-658-13640-6).
 - [20] Alex Krizhevsky, Vinod Nair und Geoffrey Hinton, Hrsg. *Learning Multiple Layers of Features from Tiny Images*. 2009.
 - [21] Alexey Kurakin, Ian Goodfellow und Samy Bengio. *Adversarial examples in the physical world*. 2016. URL: <http://arxiv.org/pdf/1607.02533v4>.
 - [22] Y. LeCun u. a. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), S. 2278–2324. ISSN: 00189219. DOI: [10.1109/5.726791](https://doi.org/10.1109/5.726791).
 - [23] Alexander LeNail. *NN-SVG*. 2019. URL: <https://alexlenail.me/NN-SVG/LeNet.html>.
 - [24] Aleksander Madry u. a. *Towards Deep Learning Models Resistant to Adversarial Attacks*. 2017. URL: <http://arxiv.org/pdf/1706.06083v4>.
 - [25] Microsoft Research. *Dogs vs. Cats*. 2013. URL: <https://www.kaggle.com/c/dogs-vs-cats/data> (besucht am 21.12.2021).
 - [26] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi und Pascal Frossard. *DeepFool: a simple and accurate method to fool deep neural networks*. 2015. URL: <http://arxiv.org/pdf/1511.04599v3>.
 - [27] Deng Pan, Xin Li und Dongxiao Zhu. “Explaining Deep Neural Network Models with Adversarial Gradient Integration”. In: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*. Hrsg. von Maria Gini und Zhi-Hua Zhou. California: International Joint Conferences on Artificial Intelligence Organization, 2021, S. 2876–2883. ISBN: 978-0-9992411-9-6. DOI: [10.24963/ijcai.2021/396](https://doi.org/10.24963/ijcai.2021/396).
 - [28] Nicolas Papernot u. a. *Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks*. 2015. URL: <http://arxiv.org/pdf/1511.04508v2>.
 - [29] Nicolas Papernot u. a. *Practical Black-Box Attacks against Machine Learning*. 2016. URL: <http://arxiv.org/pdf/1602.02697v4>.
 - [30] Mark Sandler u. a. “MobileNetV2: Inverted Residuals and Linear Bottlenecks”. In: (2018). URL: <http://arxiv.org/pdf/1801.04381v4>.
 - [31] Ali Shafahi u. a. “Are adversarial examples inevitable?” In: *International Conference on Learning Representations* (2020). URL: <http://arxiv.org/pdf/1809.02104v3>.
 - [32] Karen Simonyan, Andrea Vedaldi und Andrew Zisserman. *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps*. 2013. URL: <http://arxiv.org/pdf/1312.6034v2>.

- [33] Karen Simonyan und Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2014. URL: <http://arxiv.org/pdf/1409.1556v6.pdf>.
- [34] Mukund Sundararajan, Ankur Taly und Qiqi Yan. *Axiomatic Attribution for Deep Networks*. 2017. URL: <http://arxiv.org/pdf/1703.01365v2.pdf>.
- [35] Christian Szegedy u. a. *Intriguing properties of neural networks*. 2013. URL: <http://arxiv.org/pdf/1312.6199v4.pdf>.
- [36] Christian Szegedy u. a. *Rethinking the Inception Architecture for Computer Vision*. 2015. URL: <http://arxiv.org/pdf/1512.00567v3.pdf>.
- [37] Yixiang Wang u. a. *IWA: Integrated Gradient based White-box Attacks for Fooling Deep Neural Networks*. 2021. URL: <http://arxiv.org/pdf/2102.02128v1.pdf>.
- [38] Han Xiao, Kashif Rasul und Roland Vollgraf. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. 2017. URL: <http://arxiv.org/pdf/1708.07747v2.pdf>.
- [39] Xiaoyong Yuan u. a. *Adversarial Examples: Attacks and Defenses for Deep Learning*. 2017. URL: <http://arxiv.org/pdf/1712.07107v3.pdf>.
- [40] Shigeng Zhang u. a. “Detecting Adversarial Samples for Deep Learning Models: A Comparative Study”. In: *IEEE Transactions on Network Science and Engineering* (2021), S. 1. DOI: [10.1109/TNSE.2021.3057071](https://doi.org/10.1109/TNSE.2021.3057071).
- [41] Zhou und Chellappa. “Computation of optical flow using a neural network”. In: *IEEE International Conference on Neural Networks*. IEEE, 1993, 71–78 vol.2. ISBN: 0-7803-0999-5. DOI: [10.1109/ICNN.1988.23914](https://doi.org/10.1109/ICNN.1988.23914).

A

Selbstständigkeitserklärung

- Diese Arbeit habe ich selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.
- Diese Arbeit wurde als Gruppenarbeit angefertigt. Meinen Anteil habe ich selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Namen der Mitverfasser:

Meine eigene Leistung ist:

Datum

Unterschrift der Kandidatin/des Kandidaten