Technology
Arts Sciences
TH Köln

Manipulierung einer Verkehrsschilderkennung mittels Adversarial Attacks

Individuelle Projektarbeit

vorgelegt am 09.10.2020 von

Noah Christoph Pütz

11114150

Inhalt

1 Einleitung

Spätestens seit dem „Big Bang des Deep Learning“ im Jahre 2009 sind Begriffe wie künstliche Intelligenz, maschinelles Lernen und neuronale Netze ständige Begleiter nicht nur in der Informatik, sondern auch in unserem alltäglichen Leben. In dem Jahr konnte Nvidia mit einer neuen Generation von Grafikarten die Geschwindigkeit von Deep Learning Systemen verhundertfachen; Der Startschuss für eine Revolution der neuronalen Netze war gegeben.

1.1 Problemstellung

Seitdem beschleunigt sich die Entwicklung solcher „selbst lernenden“ und „intelligenten“ Systeme immer weiter und in den letzten zehn Jahren schlugen Computer eine Vielzahl an Menschen in einer großen Varietät von Aufgaben. Vor allem das Meistern komplexer Brettspiele wie Schach, Shogi und Go waren große Meilensteine in der jüngeren Geschichte der künstlichen Intelligenz.

Während die meisten Anwendungsfälle für künstliche Intelligenzen und speziell neuronale Netze nur in der Forschung auftreten, hat es die Bilderkennung mittels neuronaler Netze in unseren Alltag geschafft. Google und Apple sortieren Bilder auf unseren Smartphones, Stock Footage wird automatisch klassifiziert und Facebook kann auf Gruppenbildern einzelne Personen identifizieren und zuordnen; All dies geschieht mit gut trainierten neuronalen Netzen.

Während dies Beispiele für Software sind, bei denen ein Fehler keine sonderlichen negativen Folgen hat, wird Bilderkennung nun seit circa drei Jahren auch im Bereich der Sicherheit eingesetzt. Auf neuronalen Netzen basierende Gesichtserkennung soll in Zukunft vermehrt an öffentlichen Plätzen eingesetzt werden und auch autonom fahrende Autos besitzen Bilderkennungssoftware zur Umgebungsanalyse. Fehler und falsche Klassifizierungen können in solchen Fällen fatale Folgen hervorrufen.

1.2 Zielsetzung

Ein Blick auf den „Gartner Hype Cycle for Emerging Technologies 2019“ (Abb. 1) zeigt, dass „Artificial Intelligence Platform as a Service“ (AI PaaS) eines der mit am meisten Spannung erwarteten Themen der nächsten Jahre ist [9]. AI PaaS beschreibt Cloud basierte Lösungen für eine Fülle an KI-Anwendungen, vor allem im Bereich der computer vision, image classification und des natural language processing. Auf solche hohen Erwartungen an eine Technologie, wie sie gerade an die künstliche Intelligenz gestellt wird, folgt eine gesellschaftliche Ernüchterung. Je stärker diese Ernüchterung ausfällt, desto länger wird es dauern, bis das Vertrauen in künstliche Intelligenzen groß genug ist, um produktiv flächendeckend eingesetzt werden zu können. Um einer zu starken Ernüchterung entgegen zu wirken, ist es wichtig, frühzeitig auf Fehl- und Manipulierbarkeit von neuronalen Netzten hinzuweisen. Dies hilft beim Einschätzen von Risiken und beschleunigt das Herausarbeiten wichtiger und sinnvoller Anwendungsgebiete und führt somit zu einer schnelleren Rückgewinnung des Vertrauens der Gesellschaft.

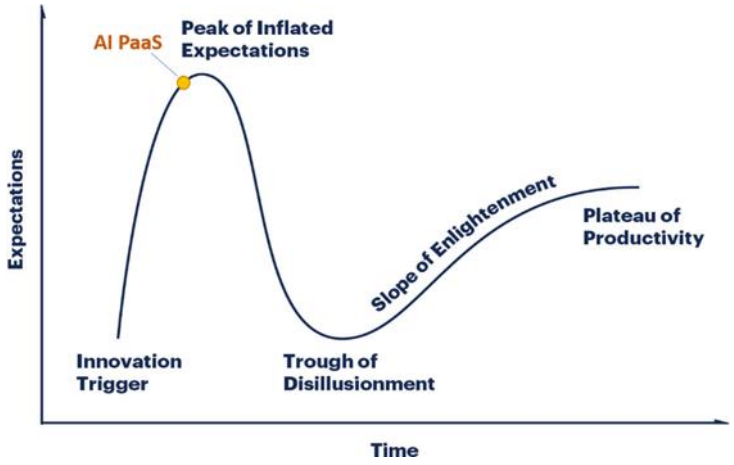


Abbildung 1: Gartner Hype Cycle for Emerging Technologies, 2019 (Quelle: [9])

Adversarial Attacks (deutsch: Gegenspielerangriffe) sind Methoden, um neuronale Netze zu täuschen und absichtlich falsche Outputs zu provozieren. Diese Methoden werden hauptsächlich im Bereich der Bilderkennung erforscht. Ziel der Forschung um Adversarial Attacks ist es, die Manipulierbarkeit von neuronalen Netzten zu erkennen und Fehler im System aufzuzeigen. Ein solche Adversarial Attack auf ein Sicherheitssystem könnte gravierende Folgen hervorrufen.

Aufgrund der wachsenden Anzahl und der steigenden Bedeutung von Bilderkennungssoftware werden Adversarial Attacks zunehmend als Bedrohung

wahrgenommen und als einer der gefährlichsten Ansätze, um künstliche Intelligenz zu hacken.

Dieses Projekt soll Adversarial Attacks erklären, die damit einhergehenden Gefahren erläutern und anhand eines Praxisbeispiels veranschaulichen. Dies ist vor allem für die Bilderkennung entscheidend, da sie zukünftig zunehmend auch im Sicherheitsbereich verwendet werden soll.

2 Neuronale Netze und Stand der Technik

Parallel zu den 2009 neu entwickelten technischen Möglichkeiten veröffentlichte die Stanford University im selben Jahr ImageNet. ImageNet ist eine Datenbank für visuelle Daten und deren dazugehörigen Kategorisierung. Sie beinhaltet mehr als 14 Million Bilder und über 20.000 Kategorien und ermöglichte so erstmals das Training von Deep Learning Systemen zur Bilderkennung [4]. Seitdem ist die Bilderkennung die größte und eine der wichtigsten Anwendungsfälle von künstlicher Intelligenz. Für viele ist es der Einstieg in das Thema der neuronalen Netze (Abb. 2) und für viele Modelle der Bilderkennung ist eine Fehlerquote von unter 0,5 % keine Seltenheit mehr. Deswegen sind neuronale Netze zur Bilderkennung eine der wenigen künstlichen Intelligenzen, die es aus der Forschung heraus- und in unseren Alltag hereingeschafft haben.

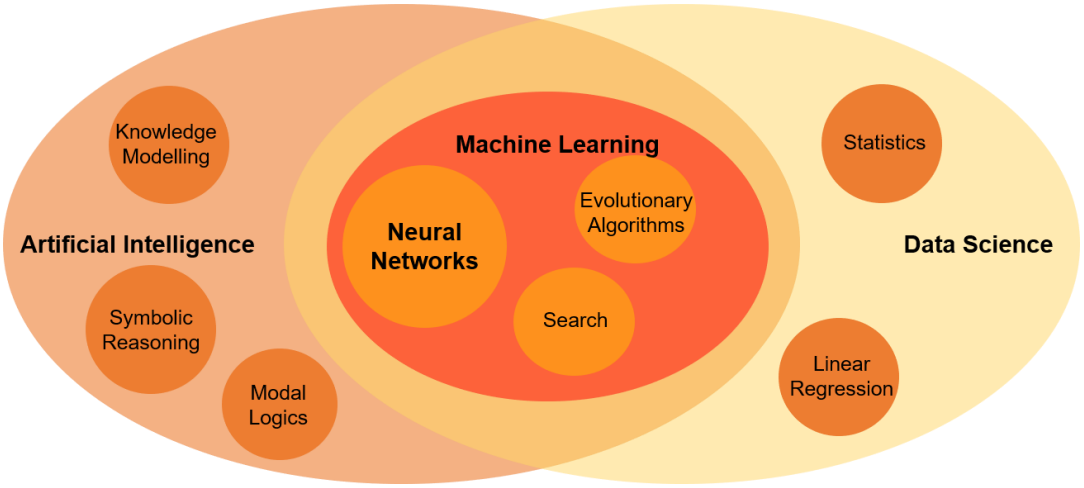


Abbildung 2: Machine Learing im Kontext (Quelle: [12])

2.1 Image Classification und Computer Vision

Moderne Bilderkennungsmodelle werden fast ausschließlich mit Deep Neural-Networks programmiert. Solche Modelle können Dutzende an Neuronenebenen mit Millionen von trainierbaren Parametern haben, dies führt zu der namensgebenden Tiefe in den Deep Learning Systemen. Das Prinzip nachdem Deep-Learning Systeme

trainiert werden und lernen lässt sich, wie bei jedem Machine-Learning-Modell; in drei Hauptkategorien unterscheiden, dem supervised learning (deutsch: überwachtes Lernen), unsupervised learning (deutsch: unüberwachtes Lernen) und reinforcement learning (deutsch: bestärktes Lernen) (Abb. 3) [11]. Mit Lernen ist dabei die Fähigkeit einer künstlichen Intelligenz gemeint, Gesetzmäßigkeiten nachzubilden.

Das supervised learning setzt, im Gegensatz zum unsupervised- oder reinforcement learning, gekennzeichnete (labelled) Daten voraus. Innerhalb des supervised learning exsitieren zwei große Problemstellungen: Regressionsprobleme und Klasssifikationsprobleme. Regressionsprobleme beschäftigen sich mit der Vorhersage kontinuierlicher Werte (Aktienkurs, Häuserpreise, usw.), wo hingegen Klassifikationsprobleme sich mit der Zuordnung eines Inputs zu einer vordefinierten Klasse beschäftigen.

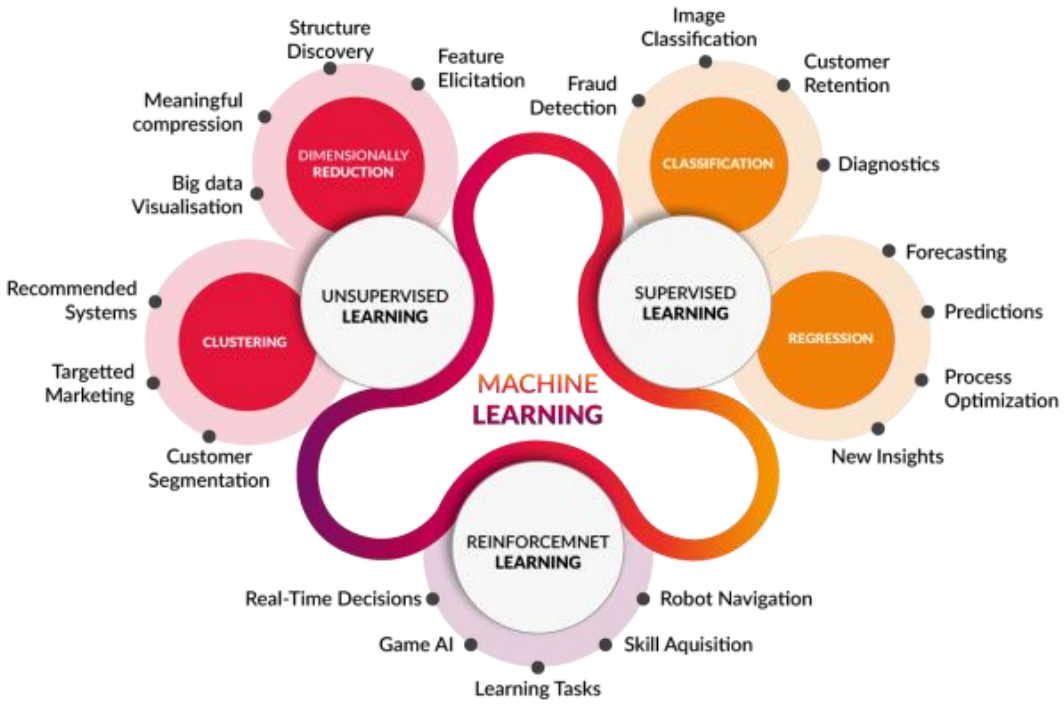


Abbildung 3: Arten des Lernens vom Machine Learning (Quelle: ironhack.com)

Die Image Classification (deutsch: Bilderkennung) beschäftigt sich mit nahezu allen visuellen und gekennzeichneten Daten. Die heutigen Anwendungen sind vielfältig, von der Diagnostik, wo mittels Image Classification Tumore bestimmt werden [10], bis hin zur Identifizierung von Kunstwerken und Gemälden .

Computer Vision ist ein interdisziplinäres Forschungsgebiet zwischen der Informatik und den Ingenieurswissenschaften, welches versucht, die durch die Image Classification erlangte Fähigkeit zu „sehen“ in das Arbeiten von Computern und Maschinen einzubinden. Beispielhaft ist dafür die Verkehrsschilderkennung welche nicht nur Verkehrsschilder identifiziert, sondern auch abhängig davon Fahrempfehlungen aussprechen kann.

2.2 Verkehrsschilderkennung

Die Verkehrsschilderkennung ist ein Assistenzsystem in vielen modernen Autos und wird seit über 10 Jahren verbaut. Sie funktioniert häufig über eine im Bereich der Rückseite des Innenspiegels montierte Videokamera. Diese filmt die vor dem Auto liegende Straße und deren Umgebung. Das aufgezeichnete Bild dient als Input für die künstliche Intelligenz [16] .

Die Verkehrsschilderkennung durchläuft zwei große Arbeitsschritte. Erstens die Object Detection (deutsch: Objektidentifizierung) und zweitens die Image Classification (deutsch: Bilderkennung) (Abb. 4). Bei der Object Detection wird ein eingehendes Bild basierend auf den abgebildeten Objekten in Teilbereiche unterteilt. Diese Teilbereiche gelten als separate Inputs für die Image Classification, welche den einzelnen Teilbereich separat analysiert und es als das entsprechende Objekt identifiziert.

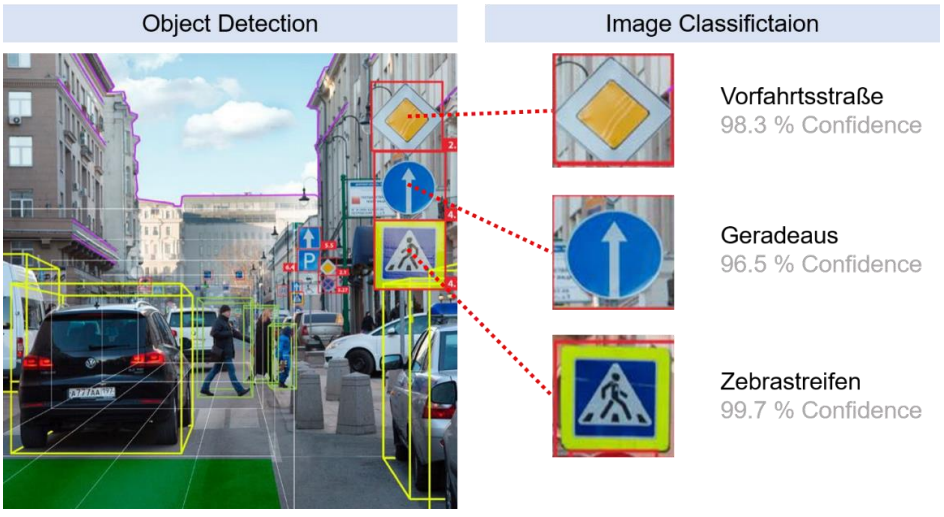


Abbildung 4: Object Detection und Image Classification

Das System assistiert dem Fahrer beim Fahren des Autos. Aufgrund der erkannten Verkehrsschilder kann das System überhöhte Geschwindigkeiten melden, es kann auf Vorfahrtssituationen hinweisen und den Fahrer vor Falscheinfahrten warnen. Der Fahrer wird meist über eine Anzeige auf dem Display oder über einen Warnton auf das

erkannte Schild aufmerksam gemacht [17]. Durch neue Vorstöße der Forschung im Bereich des autonomen Fahrens gewinnt die Verkehrsschilderkennung zunehmend an Relevanz. Das liegt daran, da die von der KI bereitgestellten Informationen über Verkehrsschilder als Grundlage für das Steuern des Fahrzeuges dienen werden. Dementsprechend wächst der Anspruch an die Qualität der Verkehrsschilderkennung.

2.3 Adversarial Attacks

Eine Adversarial Attack ist ein vorsätzlich von einem Angreifer manipulierter Input für eine KI, um einen fehlerhaften Output dieser zu provozieren. Wichtig ist, dass die Manipulation keinen Einfluss auf die Wahrnehmung des Menschen hat, sondern nur auf Prozesse der KI. Dem menschlichen Auge fällt in den meisten Fällen kein Unterschied zwischen einem manipulierten und einem originalen Bild auf. In diesem Zusammenhang ist oft von „optischen Täuschungen [1] “ für künstliche Intelligenzen die Rede.

2.3.1 Historie

Eine solche Attacke wurde bereits auf eine Vielzahl von Input-Arten angewandt. Eines der ersten attackierten Machine Learning Modelle war ein E-Mail Spam Classifier. Solche Classifier analysieren den Text einer eingehenden E-Mail und klassifizieren anhand der im Text vorkommenden Wörter die E-Mail in die Kategorien „Spam“ oder „kein Spam“. Bereits vor der Erfindung der ersten neuronalen Netze im Jahre 2004 fanden die ersten solcher Attacken statt, inder heraus gefunden wurde, welche Wörter der Classifier als „gute“ und welche als „schlechte“ Wörter sortierte. Der manipulierte E-Mail-Text wurden anschließend nur noch mit „guten“ Wörtern geschrieben, behielt aber den gleichen Inhalt, woraufhin er anschließend nur noch von Menschen als Spam identifiziert werden konnte [6].

Bis ins Jahr 2013 hofften viele Forscher, dass nicht lineare Classifier, wie neuronale Netze, sicher vor adversarial Attacks (AA) seien. Als 2012 neuronale Netze das Forschungsfeld der Bilderkennung dominierten, wurden auch die Bemühungen, neuronale Netze durch AA zu manipulieren, größer. 2014 gelang der Durchbruch und die ersten Attacken auf neuronale Netze waren erfolgreich [15]. Seitdem beziehen sich die Attacken fast ausschließlich auf neuronale Netze mit visuellem oder auditivem Input.

2.3.2 Targeted und Untargeted

AAs werden in zwei Kategorien unterteilt; gezielte (englisch: targeted) und ungezielte (englisch: untargeted) Attacken. Eine gezielte Attacke hat eine Zielklasse und versucht, dass ein Modell einen Input nicht mehr als die ursprüngliche Klasse identifiziert, sondern als die vordefinierte Zielklasse. Eine ungezielte Attacke hingegen versucht lediglich, eine fehlerhafte Klassifizierung herbeizuführen; Als welche falsche Klasse der Input identifiziert wird spielt dabei keine Rolle. Der Vorteil einer ungezielten Attacke liegt in der benötigten Zeit für die Ausführung. Gezielte Attacken brauchen mehr Rechenzeit, besitzen dafür aber den Vorteil einer besseren und unauffälligeren Attacke [8].

2.3.3 Black-, Grey- und White Box

Zusätzlich wird in der Forschung zwischen „Black Box“ -„Grey Box“ - und „White Box“-Attacken differenziert [2]. Der Unterschied dieser drei Attacken liegt in dem Wissen über das anzugreifende Netz. Bei einer White Box Attacke hat der Angreifer Informationen über die Architektur des Netzes, sowie die dazugehörigen Aktivierungsfunktionen und die trainierten Parameter. Grey Box Attacken fehlt jegliches Wissen über Parameter und deren Zusammenspiel. Solche Attacken stützen sich alleinig auf die Struktur des Netztes. Die einzige verfügbare Information bei einer Black Box Attacke hingegen ist der In- und Output der künstlichen Intelligenz. Bei einer solchen Attacke wird ein anderes neuronales Netz oder sogar gar kein Netz herbeigezogen, um einen adversarial Input zu generieren, in der Hoffnung, dass das zu attackierende Netz ähnlich auf die Manipulation reagiert. Die meisten Verteidigungsmechanismen, welche bereits erfolgreich über den Attackierer in einem Grey- oder Black Box Szenario triumphierten, scheiterten an White Box Attacks. Zum Beispiel waren sieben der neun bei der 2018 „International Conference on Learning Representations“(ICLR2018) vorgestellten Verteidigungen unerfolgreich gegen die adaptive Struktur einer White Box Attacke. [12]

2.3.4 Attack-Arten

Über die Zeit hinweg wurden viele verschiedene Arten zur Durchführung einer solchen Attacke probiert und definiert. Nachfolgend sind die wichtigsten Attacken-Methoden aufgeführt. Auch wenn die meisten dieser Attacken auf verschiedene Deep Neural Nets anwendbar sind, werden die folgenden Methoden im Kontext zur Image Classification vorgestellt.

Mit der Fast Gradient Sign Method (FGSM) wurde 2014 die erste nachweislich erfolgreiche Attacke auf ein Deep Neural Net (DNN) durchgeführt. Die Methode wurde im selben Jahr von Ian J. Goodfellow, Jonathon Shlens und Christian Szegedy bei Google erfunden. Sie kann sowohl als untargeted oder targeted attack durchgeführt werden und setzt ein White Box Szenario voraus, da es auf Grundlage der Parameter des attackierten Netzes eine Störung (auch Noise genannt) bestimmt. Der Noise wird dann mit einem Faktor (in diesem Fall 0,007) auf das Bild addiert und die Manipulation ist abgeschlossen (Abb. 5). [5]

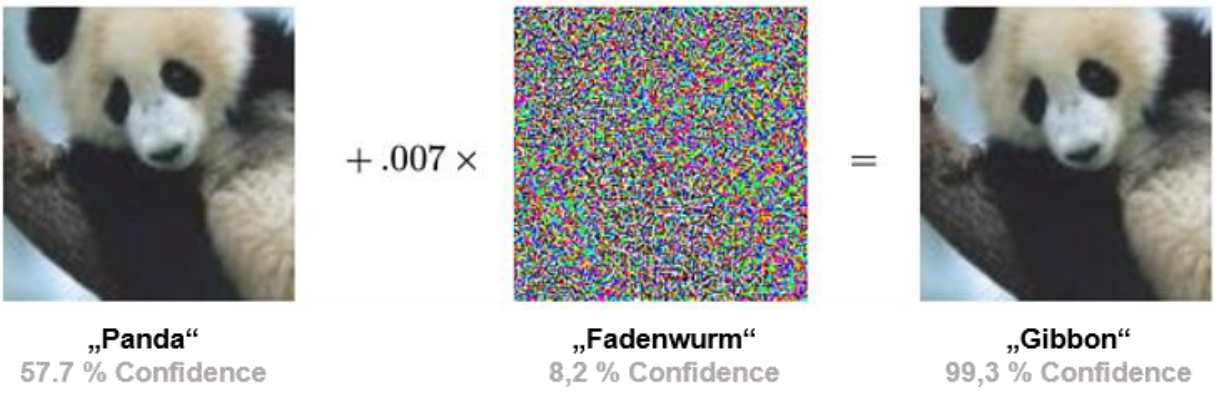


Abbildung 5: Beispiel für ein FGSM (Quelle: openai.com)

Anstatt die Störung in einem Schritt mit einem vordefinierten Faktor hinzuzufügen, wird bei der Basic Iterative Method (BIM) der Faktor schrittweise erhöht, bis eine zufriedenstellende „Confidence“ des DNNs erreicht ist [12]. Vorteile im Gegensatz zu FGSM ist die feinere Justierung des Noises, jedoch ist der Zeitaufwand erheblich höher [13].

Die zwei bereits vorgestellten Methoden sind berühmte Beispiele für sogenannte Noise-Attacks, also Attacken, welche durch eine auf das ganze Bild gelegte Störung manipulieren. Neuste Forschungen zeigen aber, dass bereits eine Änderung im richtigen Bereich des Inputs für eine vollkommende Fehlklassifikation durch das DNN ausreicht. Solche Manipulationen werden Adversarial Patch (deutsch: feindlicher Flicken) genannt [3] . Solche Patches kommen in allen Formen und Farben vor und wurden bereits unter realistischen Bedingungen ausprobiert. Ein Forschungsteam der Carnegie Mellon University hat 2016 Brillen erstellt, die zur falschen Klassifizierung des Trägers führten (Abb. 6). [14]

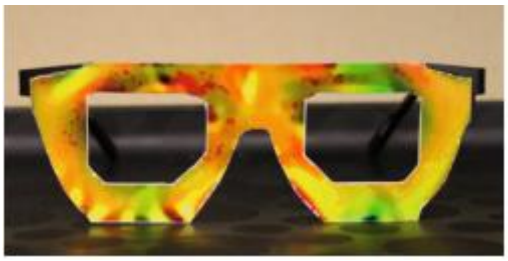


Abbildung 6: Adversarial Patch als Brille (Quelle: [14])

Eine solche Attacke besitzt den Vorteil, dass eine Gesichtserkennungssoftware keinen Fehler melden würde, weil zum Beispiel das Gesicht nicht zu identifizieren sei. Stattdessen würde es mit hoher Wahrscheinlichkeit eine aus seiner Sicht vertrauensvolle Antwort geben (Abb. 7).

Abbildung 7: Adversarial Patches in der Realität (Quelle: [14])

Eine der erfolgreichsten Methoden für Black Box Attacks sind auf Generative Adversarial Networks basierende Attacken (GAN) [18]. GANs sind DNNs, welche nach einem unsupervised learning Prinzip aufgebaut werden. Ein GAN besitzt immer zwei Netze; Eins der beiden Netze ist bereits trainiert und soll attackiert werden, das andere Netz trainiert sich im Laufe der Attacke und besitzt die Aufgabe, eine erfolgreiche Angriffsstrategie zu entwickeln. Im Laufe der Attacke generiert ein Netz (der Generator) Inputs für das bereits trainierte Netz (der Diskriminator). Der Diskriminator generiert einen Output, welcher die Grundlage zum Anpassen der Parameter des Generators dient. Dieses Nullsummenspiel wird so lange fortgesetzt, bis der Diskriminator eine falsche Klassifizierung vornimmt. Der Vorteil bei GANs ist, dass kaum Informationen über den Diskriminator bekannt sein müssen. [2]

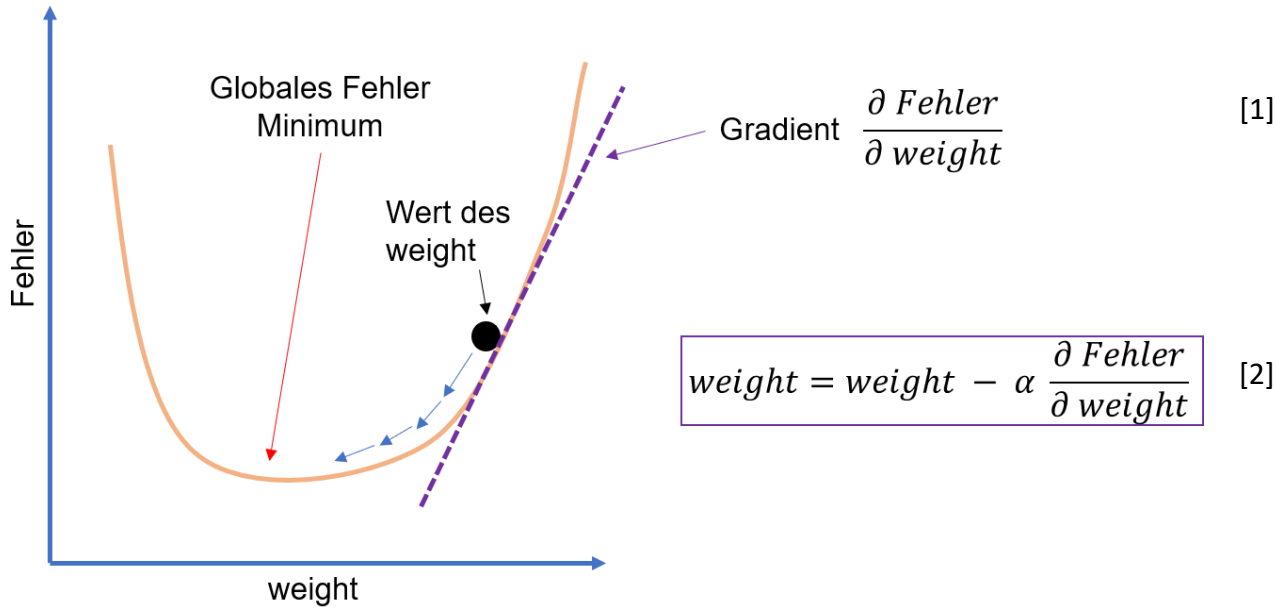
Das Feld der Adversarial Attack ist vielfältig und sehr aktuell. Bislang findet ein Wettrüsten zwischen neuen Verteidigungsmechanismen und neuen Angriffsmethoden statt, bei dem noch nicht klar ist, wer am Ende gewinnt.

3 Attackieren mit der Fast Gradient Sign Methode

Die Fast Gradient Sign Method ist die Grundlage vieler erfolgreicher White Box Attacken und wird auch heute noch sehr effektiv verwendet [5].

Beim Training eines neuronalen Netzes werden Parameter mittels eines „Backpropagation“-Algorithmus justiert. Backpropagation (deutsch: Fehlerrückführung) wird vor allem im supervised learning eingesetzt und ist ein Spezialfall eines allgemeinen Gradientenverfahrens in der Optimierung. Die im Algorithmus errechneten Gradienten sind die Basis für die Anpassung eines jeden Parameters im neuronalen Netz.

Gradienten sind im Grunde Richtungen und Ausmaß: Zum einen die positive oder negative Richtung, in welcher ein Parameter angepasst wird und zum anderen das Ausmaß, wie stark der Parameter in der entsprechenden Richtung angepasst wird. Die Anpassung findet so statt, dass sich der Wert, auf dem der Gradient basiert, verbessert.



[1]

$$weight = weight - \alpha \frac{\partial\, Fehler}{\partial\, weight}$$

[2]

Abbildung 8: Gradientenabstiegsverfahren (Quelle: blog.clairvoyantsoft.com)

In Abbildung 8 ist ein solcher Algorithmus dargestellt. Der Parameter „weight“ (deutsch: Gewicht) ist einer der wichtigste anzupassende Wert in einem neuronalen Netz. Je besser die weights im Netz an jedem Neuron abgestimmt sind, desto fehlerfreier funktioniert die Berechnung des Netzes. Das Ziel des Backpropagation ist

nun den Fehler des Netzes zu reduzieren, indem es den optimalen Wert für den weight findet. Anders ausgedrückt: Das globale Minimum des Fehlers wird gesucht.

Das weight wird mit einem zufälligen Wert initialisiert, hier dargestellt durch einen schwarzen Punkt. Nun wird zunächst erst der Gradient des Punktes bestimmt, um anschließend den Wert in entgegengesetzter Richtung anzupassen (da der Fehler minimiert und nicht maximiert werden soll). Die Anpassung erfolgt in kleinen Schritten mittels des Faktors α. α ist die Anpassungsrate des weights, der die Geschwindigkeit des Lernens des neuronalen Netzes festlegt. Wird dieser Schritt oft genug wiederholt, bewegt sich das weight, also der schwarze Punkt, stetig in Richtung des globalen Minimums, bis es diesen endgültig erreicht hat. Der ganze Algorithmus ist mit der Metapher eines Balls, der einen Berg herunter- und in ein Tal hinein- rollt zu veranschaulichen. Der Ball bewegt sich so lange in negativer Richtung zur Steigung, bis er im Tal liegen bleibt.

Abbildung 9: Gradientenabstiegsverfahren mit der Ball-Metapher

Dies erklärt den Gradienten Teil im Namen der „Fast Gradient Sign Method“. Das „Sign“ ist das Vorzeichen des in der Backpropagation bestimmten Wertes und gibt lediglich die „Richtung“ an, indem der Wert verändert wird, um den Fehler zu verringern. Bei einem Gradienten **g** mit dem Wert **0,7**, **13** oder **404** ist $\boldsymbol{sign}(\boldsymbol{g}) = \mathbf{1}$, also immer positiv und bei **g** mit einem Wert von **-0,7**, **-44** oder **-503** ist $\boldsymbol{sign}(\boldsymbol{g}) = -\mathbf{1}$, also dementsprechend negativ.

Zusammengefasst generiert eine Funktion **f** mit dem Wert **x** den Wert **y**. Der Gradient zeigt nun, wie der Wert **y** betroffen ist, wenn sich der Wert **x** verändert. Ist der Gradient **+g**, dann wird bei einer kleinen Veränderung des Wertes **x**, dementsprechend **y** positiv um den Faktor **g** verändert. Im Umkehrschluss verändert sich der Wert **y** bei einem Gradienten von **-g** negativ um den Faktor **g**.

Bei einem Beispiel der Image Classification sind **x** die Parameter des Netzes, **f** alle Rechnungen, die innerhalb des Netzes getätigt werden und **y** der Fehler, den der Output zu den tatsächlichen Daten aufweist. Deswegen soll **y** in den meisten Fällen auch verringert werden. All dies ist die Grundlage des Lernens eines neuronalen Netzes.

Mit der Fast Gradient Sign Methode werden aber nicht die Parameter eines Modells angepasst, um ein solches zu trainieren, sondern es werden die Pixel eines Bildes angepasst um den Input, also ein Bild, zu trainieren. Dementsprechend ist bei der FGSM **f** weiterhin die Berechnung des Netzes, **y** weiterhin der Fehler des Outputs und **x** nun das Input-Bild. Folglich heißt das, wenn die Gradienten **+/-g** angeben in welche Richtung sich der Wert von **y** um den Faktor **g** verändert, dann zeigt **+/-g** in welche Richtung das Input-Bild **x** verändert werden muss, um den Fehler des Outputs **y** im Verhältnis zu dem erhofftem Output zu erhöhen. Das Ziel der FGSM ist es, diesen Fehler zu erhöhen, um eine falsche Klassifizierung zu erreichen. Da diese Gradienten vom Wert her meist relativ klein sind und die Störung des Bildes nicht genug wäre, wird das Vorzeichen genommen, dementsprechend also $\mathbf{sign}(\mathbf{g})$. Dies ist die Maximale Richtung in, welcher der Wert eines Pixels angepasst werden muss. Da diese Anpassung dazu führen würde, dass das Original Bild kaum erkennbar ist wird es zusätzlich mit einem frei wählbaren Faktor **ε** noch verrechnet, welcher kleiner als der Wert 1 ist. Ist **ε = 0,3** dann wird aus einem Signs **-1** der Werte **-0,3**.

Zusammengefasst ist die Formel zum Erstellen eines manipulierten Bildes mit dem FGSM [5]:

$$\boldsymbol{M = x + \varepsilon * sign(\nabla_x J(f, x, y))} \quad [3]$$

- **M** : Das manipulierte Bild
- **x** : Originalbild
- **ε** : Faktor um sicherzustellen, dass die Veränderung nicht zu groß ist
- **∇** : Gradientenoperator
- **J** : Fehlerfunktion
- **f** : Parameter des Netzes
- **y** : wahrer Outputwert

4 Validierung

4.1 Attackiertes Modell

Da die Attacke in einem White Box Szenario stattfindet, sind die Parameter des Netzes und des Trainings bekannt.

4.1.1 Netzarchitektur

Bei dem attackierten neuronalen Netz handelt es sich um ein „Convolutional Neural Network“ (CNN oder ConvNet), welches zur Kategorie der DNNs gehört, da mehrere aus Neuronen bestehende Ebenen innerhalb des neuronalen Netzes sind. CNNs wurden von Yann LeCun erfunden und werden hauptsächlich bei der maschinellen Verarbeitung von Bild- und Audiodaten verwendet.

CNNs sind aktuell eine der effektivsten und besten Möglichkeiten zum Aufbauen eines Image Classification Modells. Dies liegt an der besonderen Arbeitsweise eines solchen Netztes. Bei den ersten auf neuronalen Netzen basierenden Image Classification Modellen wurde der zweidimensionale Input eines Bildes auf eine Dimension reduziert und als Input in das neuronale Netz gegeben. Da hierbei jedoch sowohl die Farbgebung als auch die Umgebung eines jeden einzelnen Pixels vernachlässigt wird, führt dies nur bei sehr simplen binären Bildern zu einem guten Ergebnis. Ein CNN hingegen verarbeitet alle Bildkanäle separat und jeden Pixel immer im Kontext zu seinen benachbarten Pixeln. Die Aufgabe eines CNNs ist es, das Bild in eine leichtere zu verarbeitende Struktur zu bringen, ohne dabei wichtige Informationen, welche für die Vorhersage relevant, sind zu verlieren.

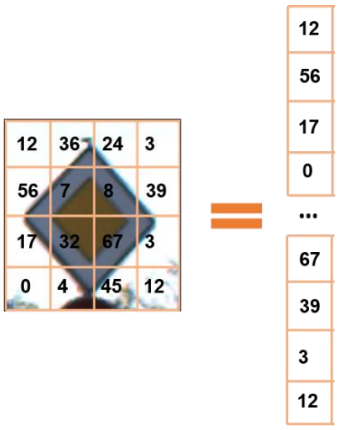


Abbildung 10: Input eines Dense Netzwerkes

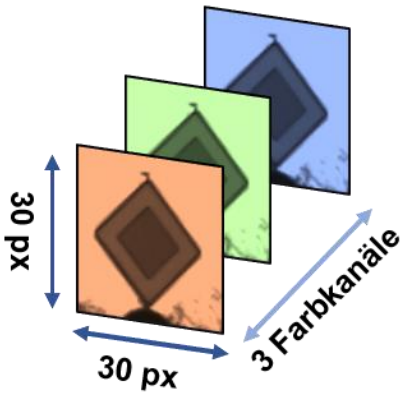


Abbildung: 11 Input eines CNNs

Das zu attackierende CNN besteht aus den drei namensgebenden Convolutional (deutsch: zusammenführen) Ebenen, wobei zwei davon jeweils eine nachfolgende Max-Pooling Ebene und eine Dropout-Ebene haben. Zum Ende folgen noch zwei komplett miteinander verbunden Dense-Ebenen inklusiver einer dazwischen geschalteten Dropout-Ebene.

Art der Ebene	Aktivierungs-funktion	trainierbare Parameter	
Convolutional	ReLU	2.432	**Kernel:** 5 x 5
Convolutional	ReLU	18.496	**Kernel:** 3 x 3
Max Pooling	/	0	**Pool Größe:**2 x 2
Dropout	/	0	**Dropout Rate:** 0,25
Convulotional	ReLU	36.928	**Kernel:** 5 x 5
Max Pooling	/	0	**Pool Größe:** 2 x 2
Dropout	/	0	**Dropout Rate:** 0,25
Flatten	/	0	/
Dense	ReLU	409.856	**Neuronen:** 256
Dropout	/	0	**Dropout Rate:** 0,5
Dense	Softmax	11.051	**Neuronen:** 43

Abbildung 12: Ebenen des neuronalen Netzes

Bei einer Inputgröße von 30 mal 30 Pixel à 3 Farbkanälen besitzt das neuronale Netz insgesamt 478.763 trainierbare Parameter.

Die hauptsächlich verwendete Aktivierungsfunktion ist die Rectified Linear Unit (ReLU). Sie ist die derzeit am häufigsten verwendete Aktivierungsfunktion der Welt

und ist aktuell in nahzu allen Deep Learning Networks zu finden. Die Funktionsweise ist denkbar einfach: Jeder positive Input führt zu einem linearen Output, wo hingegen jeder negative Input auf den Wert 0 angepasst wird.

$$f(x) = \max(0, x) \quad [4]$$

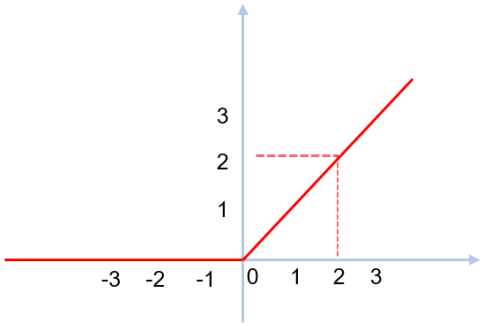


Abbildung 13: Rectified Linear Unit

Auf der letzten Ebene wird eine Softmax Aktivierungsfunktion (auch normalisierte Exponentialfunktion genannt) verwendet. Die Softmax Funktion erstellt eine Wahrscheinlichkeitsverteilung über alle Outputs des Netzes. Der Output mit der höchsten Wahrscheinlichkeit ist die vom Netz vorhergesagte Klassifizierung.

$$S(y_j) = \frac{e^{y_j}}{\sum_i e^{y_i}} \quad [5]$$

Bei vier Neuronen mit einem Output von 10, 11, 7 und 5 ist der durch die Softmax-Funktion angepassten Werte 0,71 für das Neuron mit dem Wert 11:

$$S(11) = \frac{e^{11}}{e^{11} + e^{10} + e^{7} + e^{5}} = 0{,}71$$

4.1.2 Datenset

Zum Trainieren eines DNNs braucht es ein Datenset. Das Datenset, welches die Grundlage zum Training des attackierten DNNs bietet, nennt sich „German Traffic Sign Recognition Benchmark“ oder auch GTSRB. Es wurde erstmals im Rahmen einer Herausforderung auf der „International Joint Conference on Neural Networks“ (IJCNN) 2011 vorgestellt und beinhaltet mehr als 50.000 Bilder und 43 Klassen.

Abbildung 14: Alle 43 Kategorien des Datensets

Die Bilder sind unterteilt in einen Trainingsordner mit 39.209 unterschiedlichen Aufnahmen und einem Test Ordner mit 12.630 unterschiedlichen Aufnahmen. Die Bilder im Trainingsordner sind wiederum in weiteren Ordner entsprechend ihrer Klasse sortiert. Alle Bilder im Test Ordner sind durchmischt und nicht entsprechend ihrer Klasse gekennzeichnet.

Abbildung 15: Beispiele aus dem Datenset

Eine Herausforderung, die das Datenset bietet, ist die unausgeglichene Datenlage zwischen den einzelnen Kategorien; In der Forschung wird von einem „Bias“ (deutsch: Voreingenommenheit) gesprochen. Mit Bias ist gemeint, dass Kategorien mit mehr

Daten im Training mehr berücksichtigt werden können, wohingegen Kategorien mit weniger Daten schwächer ins Training einfließen.

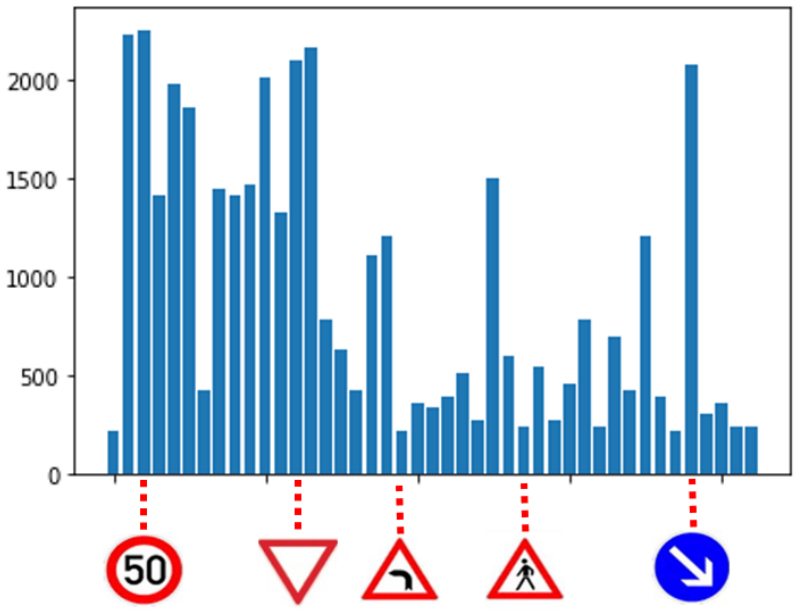


Abbildung 16: Unterschiedliche Mengenverteilung der Kategorien

In dem Fall des GTSRB beinhaltet die Kategorie „50 km/h“ über 2.250 Aufnahmen, im Gegensatz zum Schild „Achtung Linkskurve“, welches nur 210 unterschiedliche Aufnahmen beinhaltet. Dies lässt bereits vor dem Training vermuten, dass ein „50 km/h“-Schild nach abgeschlossenem Training zuverlässiger richtig kategorisiert wird. Da ein neuronales Netz aber bei allen Kategorien ähnlich zuverlässig funktionieren soll, handelt es sich hierbei um einen unerwünschten Nachteil des Datensets. Normalerweise wird jede Kategorie auf die Anzahl der Kategorie mit den geringsten Aufnahmen reduziert, da jedoch dann die Anzahl aller Trainingsdaten zu klein wäre, wird die Unausgeglichenheit in diesem Fall in Kauf genommen.

Eine weitere Auffälligkeit ist die stark schwankende Größe der einzelnen Fotos. Dies muss beim Training beachtet und angepasst werden.

Abbildung 17: Unterschiedliche Qualität der Bilder

4.1.3 Training

Für das Training des Netzwerkes müssen alle Input-Daten auf die Input-Größe des Netzwerkes skaliert werden. Die Input-Größe des Netzwerkes ist dreißig Pixel in der Höhe und dreißig Pixel in der Breite für jeweils drei Farbkanäle. In Abbildung 18 sind angepasste Beispiele aus dem Datenset aufgeführt. Die blaue Farbtönung liegt an der Python-Bibliothek „matplotlib“, welche zur Darstellung der Testdaten verwendet wurde und spiegelt nicht die tatsächlichen Daten wider.

Abbildung 18: Skalierte Beispiele aus dem Datenset

Zum Abschluss der Vorbereitung werden die Testdaten vermischt und in ein Trainingsset (31.368 Bilder) und ein Validierungsset (7.841 Bilder) aufgeteilt.

Um das Training zu starten muss eine Verlustfunktion festgelegt werden, welche sowohl für das Training als auch für die Evaluation des CNNs gebraucht wird. Die Verlustfunktion misst wie falsch die Vorhersage des Netzes im Bezug auf die tatsächlich abgebildeten Schilder war. Je geringer der Wert der Verlustfunktion desto besser wurde das Netz trainiert. Das zu attackierende Netz verwendet zum Trainieren die Funktion der kategorischen Kreuzentropie (englisch: categorical cross entropy), welche speziell auf das Berechnen eines Verlustes bei einer Wahrscheinlichkeitsverteilung ausgelegt ist. Zusätzlich wird eine Optimierungsfunktion festgelegt, welche anhand des errechneten Verlustes die trainierbaren Parameter des Netzes anpasst; Dieser Vorgang wird in der Forschung „Backpropagation“ genannt. Der von dem Netz verwendete Optimizer heißt „adam“, was die Kurzform für „adaptive

moment estimation“ ist. Er wurde 2014 von Diederik P. Kingma und Jimmy Lei Ba speziell für das Training von neuronalen Netzen designt und veröffentlicht. Abschließend muss noch die Anzahl der Epochen und eine „Batch“-Größe festgelegt werden. Epochen sind die Wiederholungen, wie oft das komplette Datenset das neuronale Netz durchläuft; In diesem Fall wurden zwanzig Epochen festgelegt, was bedeutet, dass zwanzig Mal alle 31.368 Bilder das CNN durchlaufen. Die Batch-Größe (deutsch: Stapel) legt wiederum fest, wie viele Bilder auf einmal durch das neuronale Netz laufen, da nicht für jedes Bild einzeln eine Anpassung des Netzes stattfindet. Diese Batch-Größe wird für das Training auf 32 Bilder pro Batch festgelegt. Ist sowohl die Anzahl der Epochen als auch die Größe des Batches festgelegt, kann das Training kann nun beginnen.

Nach zwanzig Epochen ist das Training beendet und die durchschnittliche fehlerhafte Abweichung des Netzes liegt bei 0,303 (Abbildung 19).

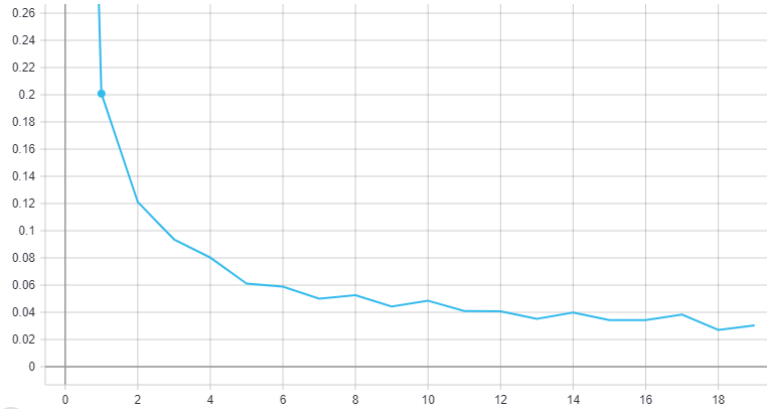


Abbildung 19: Entwicklung des Fehlers im Laufe aller Epochen

Dementsprechend liegt die Genauigkeit des Netzes bei den Trainingsdaten bei 99,13% (beide Wert können von Training zu Training schwanken) (Abbildung 20).

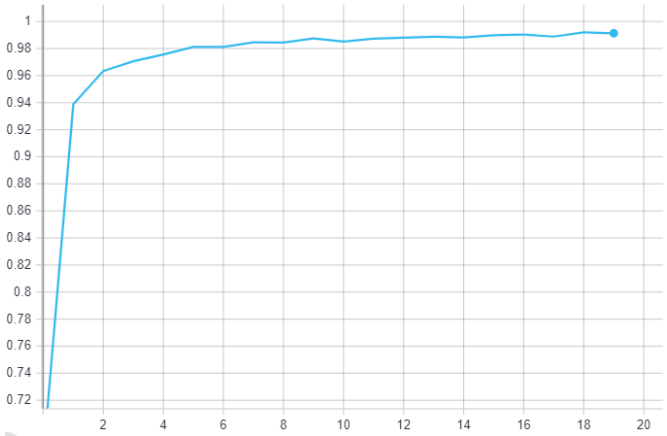


Abbildung 20: Entwicklung der Genauigkeit im Laufe aller Epochen

Da jedes Bild zwanzig Mal vom CNN bearbeitet wurde und die Chance besteht, dass sich das Netz den Input einfach „gemerkt“ hat, anstatt wirklich Gesetzmäßigkeiten zu erlernen, werden dem CNN noch unbekannte Daten gegeben. Bei den neuen Daten hat das Netz eine Genauigkeit von 97,25 % und einen durchschnittlichen Fehler von 0,144.

Eine einzelne Vorhersage sieht folgendermaßen aus:

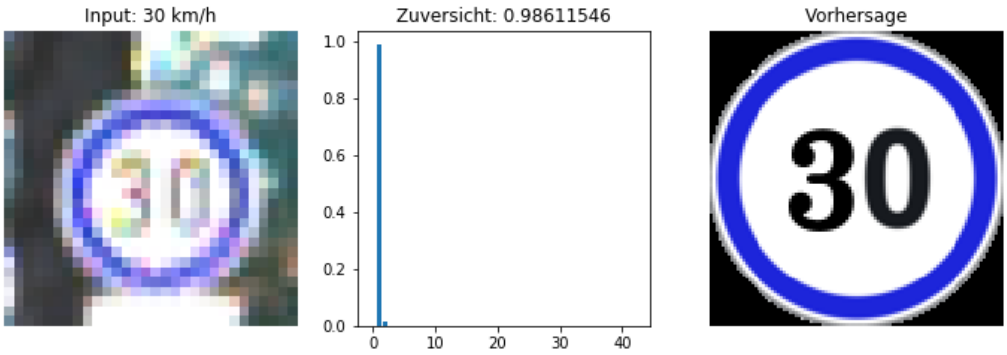


Abbildung 21: Vorhersage eines 30 km/h-Schildes

Abbildung 21 zeigt auf der linken Seite den Input des CNNs, in der Mitte die Verteilung der Wahrscheinlichkeit über alle 43 Outputs des Netzes und auf der rechten Seite die Vorhersage des Netzes. Die Zuversicht ist der höchste Wert aller 43 Outputs. Die Abbildung zeigt, sowohl das richtige Ergebnis des Netzes, als auch die hohe Zuversicht von über 98%.

Vor allem der Erfolg bei sehr dunklen Bildern ist erwähnenswert (Abbildung 22):

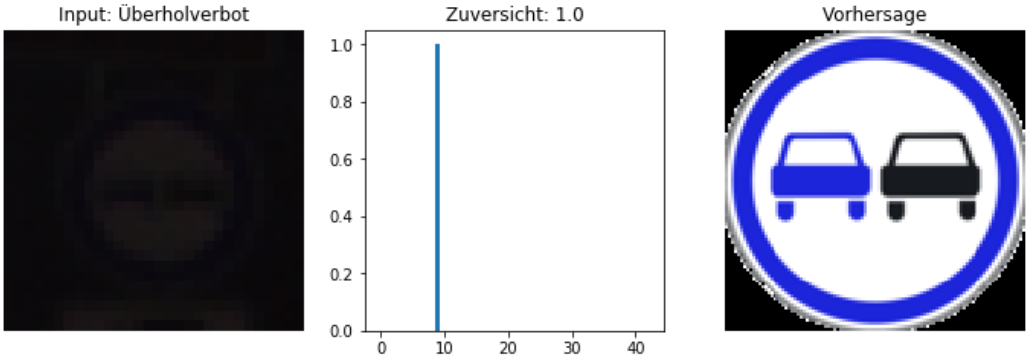


Abbildung 22: Vorhersage eines Überholverbots-Schildes

Trotz des nahezu komplett schwarzen Bildes ist sich das Netz zu 100% sicher, die richtige Vorhersage getroffen zu haben. Dennoch zeigt Abbildung 23, dass es Beispiele gibt, bei dem das Netz versagt.

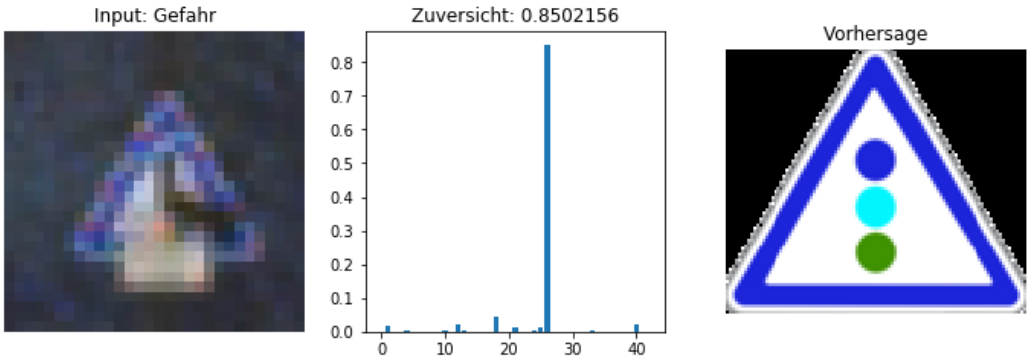


Abbildung 23: Vorhersage eines Gefahren-Schildes mit falscher Vorhersage

Zur Vollständigen Validierung der Performance des Netzes wird in Abbildung 24 eine „Confusion-Matrix“ (deutsch: Verwirrungsmatrix) herangezogen. Zur besseren Visualisierung sind keine Werte, sondern nur Farbtöne dargestellt.

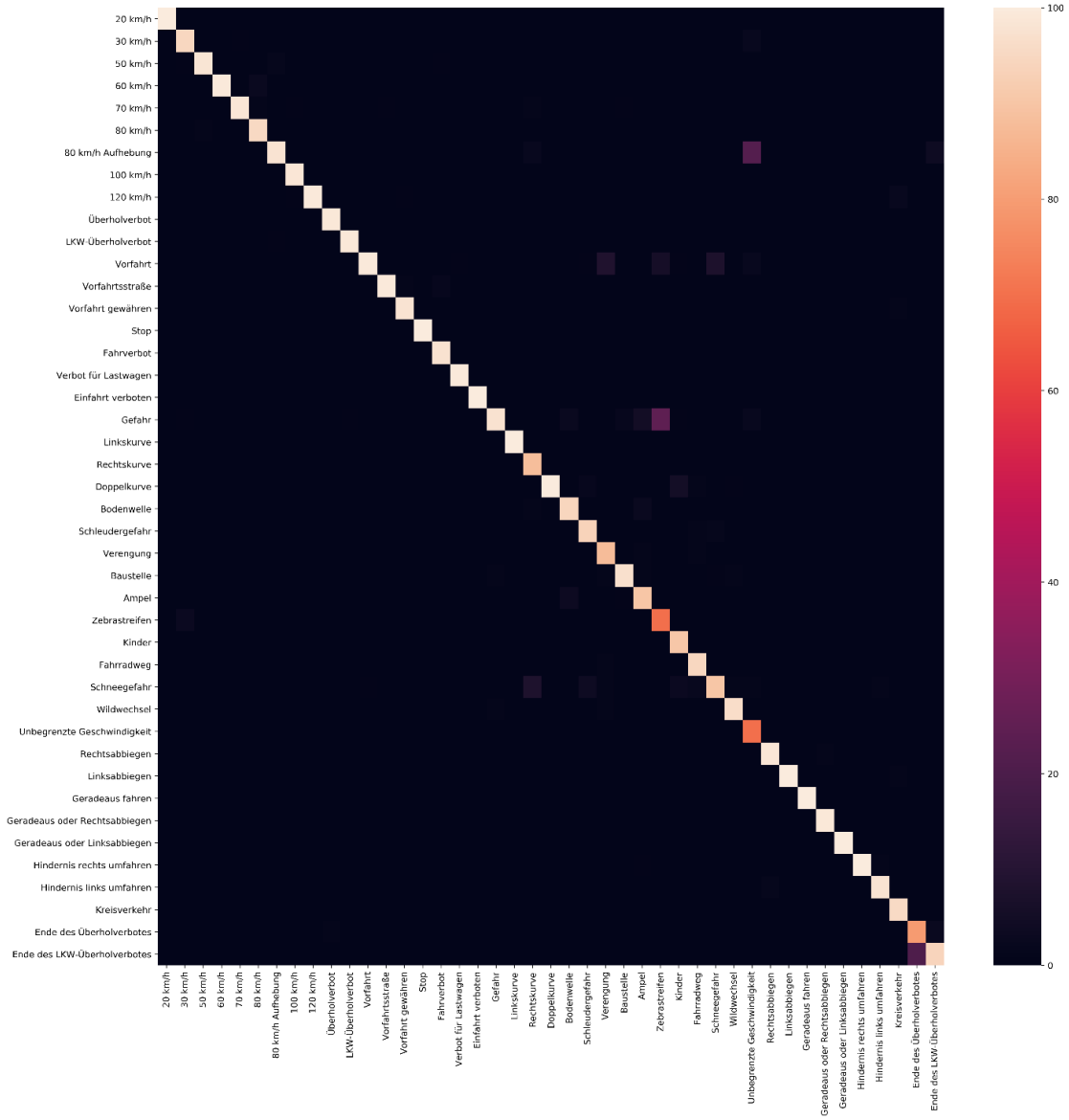


Abbildung 24: Confusion-Matrix

Auf sowohl der X- als auch Y-Achse sind alle Kategorien des Datensets aufgeführt. Je heller die Farbe eines Kästchens ist, desto mehr Bilder aus der entsprechenden Kategorie wurden als die entsprechend andere Kategorie identifiziert. Erhofft wird eine möglichst helle Diagonale, welche bedeutet, dass alle Bilder richtig zugeordnet wurden.

In dem Fall des trainierten CNNs ist diese helle Diagonale größtenteils gegeben. Ausreißer ist zum Beispiel die Klasse „Zebrastreifen“, welche nur zu 70% richtig kategorisiert wurde. Die übrigen 30% wurden hauptsächlich als Gefahrenschilder identifiziert. Dies hat zwei Gründe: Zum einen sind sich die Schilder aufgrund der Form und des schwarzen Symbols in der Mitte relativ ähnlich (Abbildung 25). Hinzu kommt aber, dass die Klasse „Zebrastreifen“ wie bereits erwähnt mit 210 Bildern eine sehr geringe Anzahl an Trainingsdaten hatte, wo hingegen die Klasse „Gefahren“ mit 1200 Bildern stark vertreten. Bei diesem Beispiel macht sich das zuvor besprochene Bias bemerkbar. Dies führt dazu, dass bei ähnlich aussehendem Input Bilder wahrscheinlicher als die Klasse kategorisiert werden, welche mehr Testdaten hatte.

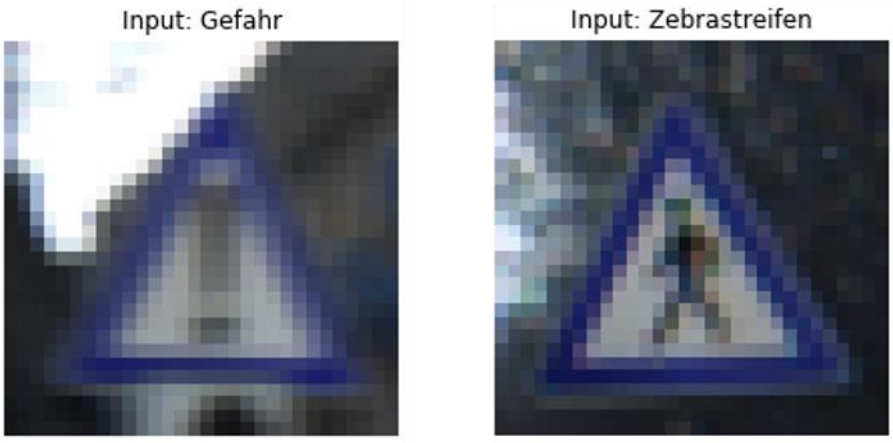


Abbildung 25: Ein Schild der Klasse "Gefahr" und eins der Klasse "Zebrastreifen"

Dieses unausgeglichene Training spielt eine Rolle, falls die künstliche Intelligenz von Manipulationen attackiert wird.

4.2 Attacke

4.2.1 Bilddateien

Die Attacke wird mit Bildern aus dem Test Datenset durchgeführt, dementsprechend sind die manipulierten Bilder dem CNN unbekannt. Zunächst werden alle 12.630 Bilder des Test Datenset eingelesen und auf die für den Input entsprechende Größe skaliert, also 30 mal 30 Pixel und 3 Farbkanäle. Die Fotos des Datenset sehen nun folgendermaßen aus:

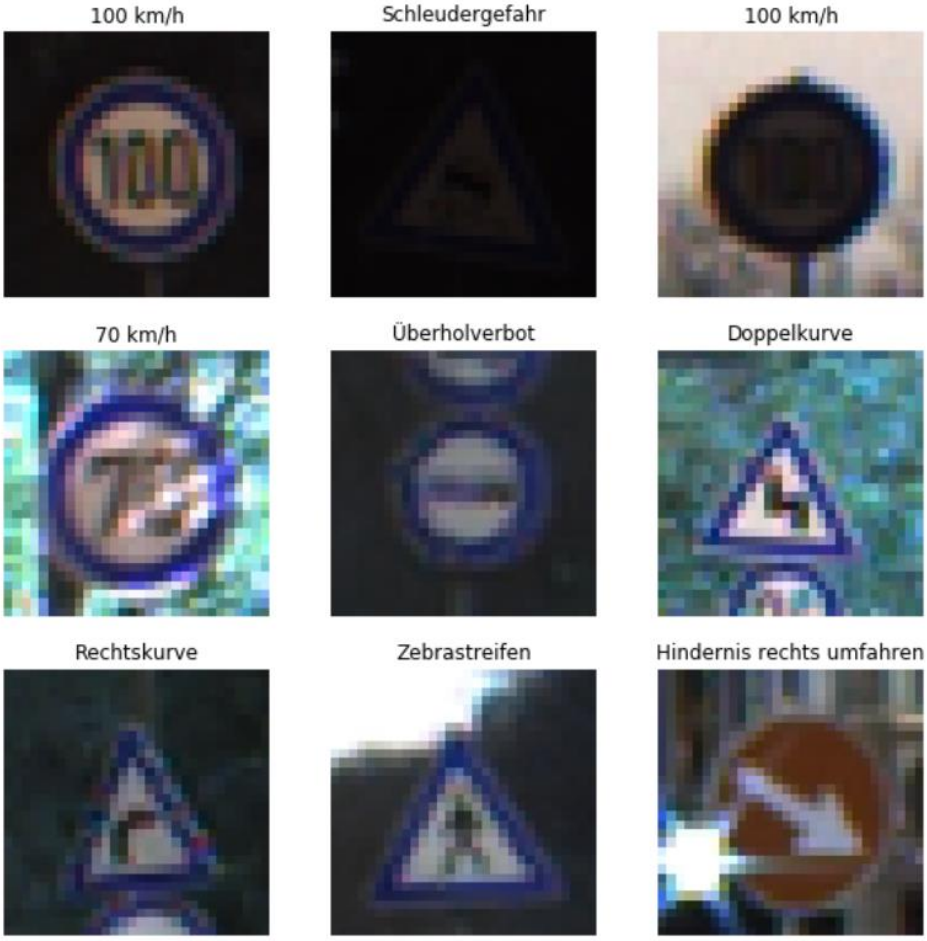


Abbildung 26: Beispielbilder aus dem skalierten Testdatenset

Bei der Adversarial Attack wird nun ein Bild dieses Datenset manipuliert und anschließend auf die ursprüngliche Größe hoch zu skalieren.

Wie im vorherigen Kapitel dargestellt hat das CNN bei diesen neuen Daten eine Genauigkeit von über 97%. Ziel der Attacke wird es sein diese Genauigkeit drastisch zu verringern.

4.2.2 Attacke auf ein Bild

Zum Attackieren des CNNs für die Straßenschilderkennung wird, die im Kapitel drei bereits beschriebene Fast Gradient Sign Method verwendet. Zur Wiederholung, die Formel besteht aus zwei großen Teilen: Erster Teil ist das Erstellen der Störung anhand der Gradienten und deren Vorzeichen.

$$\mathbf{Störung} = \mathbf{sign}\left(\nabla_{\mathbf{x}} \mathbf{J}(\mathbf{f}, \mathbf{x}, \mathbf{y})\right) \quad [6]$$

Im zweiten Teil wird die im ersten Teil erstellte Störung auf das Originalbild mit einem Faktor $\boldsymbol{\varepsilon}$ aufaddiert, um das entsprechende manipulierte Bild zu erstellen.

$$\mathbf{M} = \mathbf{x} + \boldsymbol{\varepsilon} * \mathbf{Störung} \quad [7]$$

Zusammengefasst sieht die Formel folgendermaßen aus:

$$\mathbf{M} = \mathbf{x} + \boldsymbol{\varepsilon} * \mathbf{sign}\left(\nabla_{\mathbf{x}} \mathbf{J}(\mathbf{f}, \mathbf{x}, \mathbf{y})\right)$$

Das Erstellen des Codes kann in Python als Funktion dargestellt werden (Abbildung 27). Diese Funktion heißt „adversarial_pattern“ und bekommt sowohl das Bild (image), für welches die Störung erstellt werden soll, sowie die korrekte Kennzeichnung (label) des Bildes übergeben:

```
# Funktion um ein adversarial_pattern zu erstellen
def adversarial_pattern(image, label):

    with tf.GradientTape() as tape:
        tape.watch(image)
        prediction = model(image)
        loss = tf.keras.losses.MSE(label, prediction)

    gradient = tape.gradient(loss, image)

    signed_grad = tf.sign(gradient)

    return signed_grad
```

Abbildung 27: Funktion zum Erstellen eines adversarial patterns

Zur Bestimmung der Gradienten muss das Bild zunächst vom CNN analysiert werden und eine Kennzeichnung entsprechend für diese vorhergesehen werden. Während das CNN das Bild analysiert und die einzelnen Neuronenebenen durch den entsprechenden Input aktiviert werden, zeichnet die Funktion jede relevante

Aktivierung samt einzelner Parameter auf. Dies geschieht im Kontext-Manager **tf.GradientTape()** in Abbildung 28.

```
4     with tf.GradientTape() as tape:
5         tape.watch(image)
6         prediction = model(image)
7         loss = tf.keras.losses.MSE(label, prediction)
```

Abbildung 28: Funktionsausschnitt zum Berechnen des Fehlers

In Zeile vier wird zunächst der Tensor, welcher das Bild beschreibt, aufgezeichnet. Danach wird eine Vorhersage des Models zu welcher Klasse das Bild zugehörig ist abgegeben um dann anschließend in der Zeile sieben der Fehler zwischen der Vorhersage und der tatsächlichen Klasse zu bestimmen. Bei der verwendeten Fehlerfunktion handelt es sich um den „Mean Square Error“ (MSE). Dieser Fehler dient jetzt als Grundlage zu der Berechnung der Gradienten.

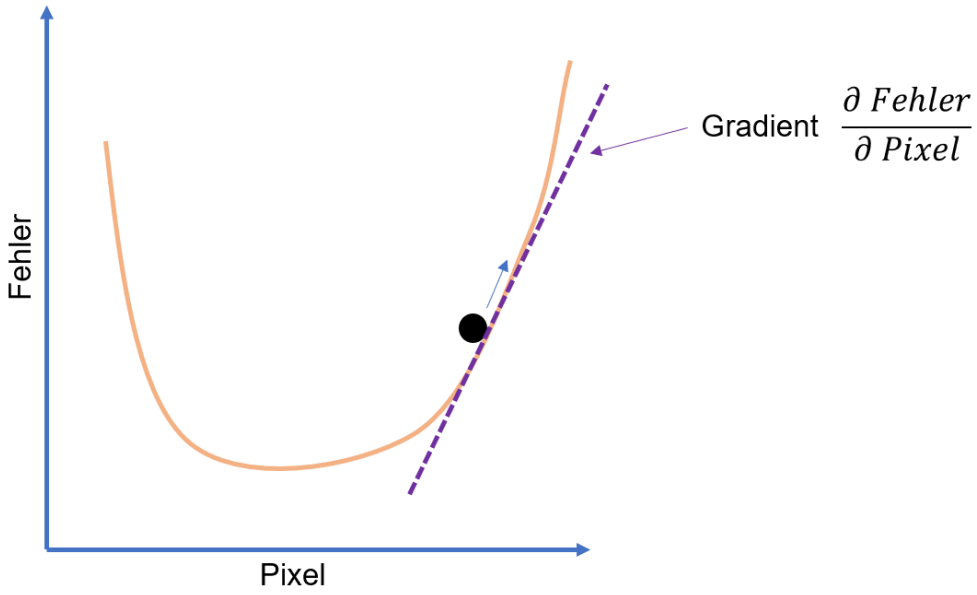


Abbildung 29: Gradientenabstiegsverfahren um den besten Wert zum Anpassen eines Pixels zu bestimmen

Normalerweise wird beim Training eines neuronalen Netzes der Parameter eines Neurons so angepasst, dass der Fehler Schritt für Schritt geringer wird. Bei einer Attacke mit FGSM wird nun der Wert des Pixels (schwarzer Punkt) des Bildes nun so angepasst, dass sich der Fehler erhöht. Die Berechnung der Gradienten im Kontext zu dem Fehler und zum Bild sieht im Python Code dementsprechend aus.

```
9     gradient = tape.gradient(loss, image)
```

Abbildung 30: Funktion tape.gradient()

Als letzter Schritt werden in Zeile elf der Funktion nur noch die Vorzeichen mittels der **tf.sign()** Funktion berücktsichtigt um als ein Tensor mit der Größe 30 mal 30 Pixel á 3 Farbkanäle als **return** der Funktion schlussendlich übergeben zu werden.

```
11      signed_grad = tf.sign(gradient)
12
13      return signed_grad
```

Abbildung 31: Sign-Funktion

Zusammenfassend hat die Funktion das Originalbild inklusive Kennzeichnung erhalten und eine Richtung zur Anpassung für jeden Farbkanal von jedem Pixel erhalten. Das bedeutet das jeder Pixel der insgesamt neunhundert Pixel durch eine Liste von drei Werten beschrieben ist. Diese drei Werte beschreiben die Intensität der Farbkanäle Blau, Grün und Rot (Abbildung 32).

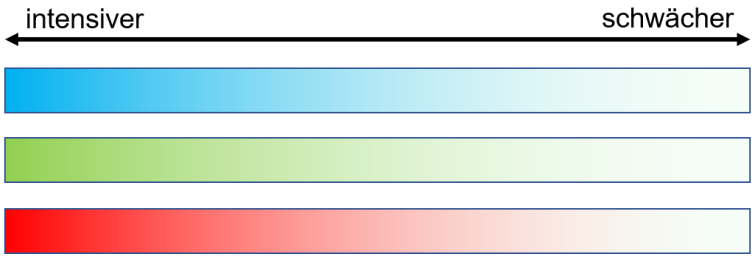


Abbildung 32: Farbkanäle

Wenn also ein Pixel nun die Werte [-1; -1; -1] besitzt, bedeutet das, dass jeder Farbkanal dieses Pixels intensiver wird. Bei den Werten [1; 1; 1] soll jeder Farbkanal schwächer werden und bei den Werten [-1; 1; -1] sollen der blaue und rote Wert schwächer werden und der grüne Wert intensiver. Dementsprechend wird durch das Intensivieren aller Farbkanäle der Pixel komplett schwarz oder durch die Erhöhung der Intensität von zum Beispiel nur dem Farbkanal blau und rot, dann lila. Die Anpassungswerte für jeden Farbkanal jedes Pixels eines Bildes eines Vorfahrtsstraßenschildes kann so aussehen:

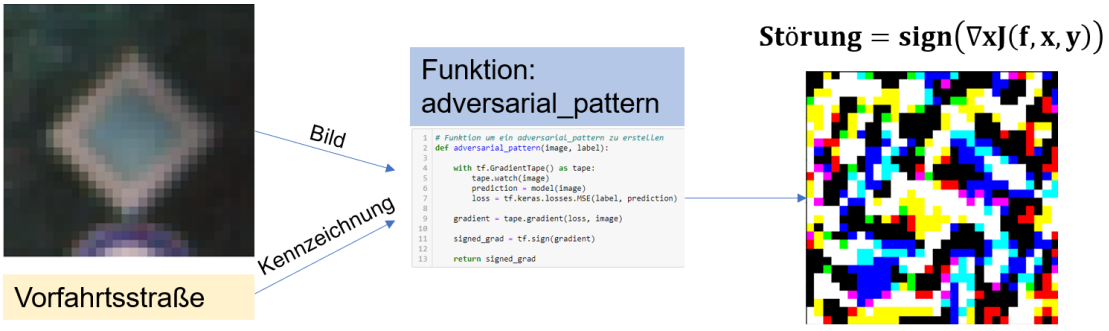


Abbildung 33: Vereinfachte Darstellung der Funktion zum Erstellen eines adversarial Patterns

Der Output also die Störung ist nun ein sehr wirres Bild, auf dem keine Formen aus dem Original Bild mehr zu erkennen sind. Diese Störung muss nun auf das Bild aufaddiert werden in einem so geringen Maße, dass es dem menschlichen Auge kaum auffällt. Der zweite Teil der FGSM-Formel ist nun relevant.

$$\mathbf{M = x + \varepsilon * Störung}$$

Zur Erinnerung je größer der Faktor der Störung $\boldsymbol{\varepsilon}$ ist, desto leichter ist die Störung des Bildes auszumachen. Aus diesem Grund wird $\boldsymbol{\varepsilon}$ stetig in kleinen Schritten erhöht, um so die geringste notwendige Veränderung herauszufinden.

```
1  max_iterations = 0.03
2  confidence = 0.9
3
4  for x in np.arange(0.0, max_iterations, 0.001):
5
6      adversarial = image + perturbations * x
7
8      original_prediction = sign_label[model.predict(image).argmax()]
9      adversarial_prediction = sign_label[model.predict(adversarial).argmax()]
10
11     print("Noise:", round(x,3),
12           "\t Confidence",round(max(model.predict(adversarial)[0]), 3),
13           "\t Prediction",adversarial_prediction)
14
15     if original_prediction != adversarial_prediction and \
16     max(model.predict(adversarial)[0]) > confidence:
17         noise = x
18         break
```

Abbildung 34: Programmauszug zur Kombination des Bildes mit der Störung

Bei diesem Python-Code in Abbildung 34, darf $\boldsymbol{\varepsilon}$ maximal den Wert 0,03 (max_iterations) besitzen, da alles darüber hinaus zu leicht für einen Menschen zu erkennen wäre. Übersteigt $\boldsymbol{\varepsilon}$ diesen Wert gilt die Attacke für dieses Bild als gescheitert. Zudem muss die falsche Vorhersage des CNNs mindestens eine Zuversicht (confidence) von 90% haben.

Bei jedem Durchlauf dieser Schleife wird zunächst die Störung mit dem entsprechenden $\boldsymbol{\varepsilon}$ in Zeile sechs aufaddiert. In Zeile acht und neun macht das CNN nun jeweils eine Vorhersage für das Original sowie das manipulierte Bild. Erst wenn die Vorhersagen für beide Bilder unterschiedlich sind und die Zuversicht über 90% liegt ist die Manipulation beendet und das manipulierte Bild kann betrachtet werden.

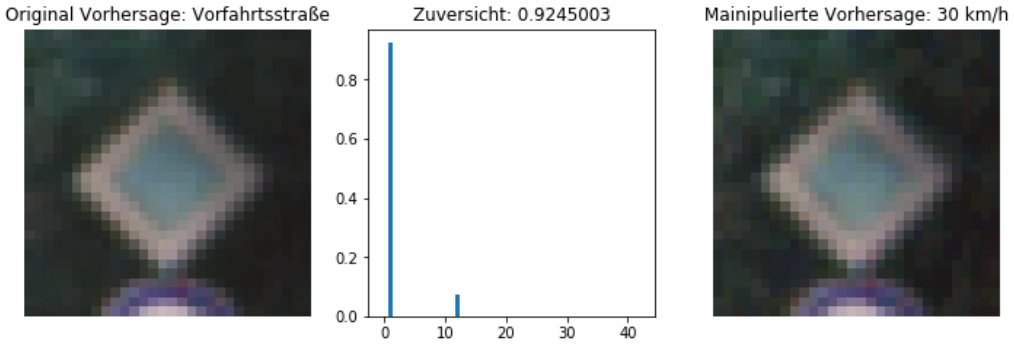


Abbildung 35: Erfolgreiche Attacke anhand eines Vorfahrtstraßenschilds

Obwohl ein Unterschied zwischen den Beispielbildern für einen Menschen nur im direkten Vergleich zu erkennen wären. Fehlklassifiziert das Modell das manipulierte Schild zu einem Schild der Klasse „30 km/h“ bei einem ε von nur 0,014. Dabei ist anzumerken, dass beide Klassen über 2000 Trainingsdaten besaßen und in der vorherigen Validierung laut der Confusion-Matrix kein einziges Bild der „Vorfahrtsstraße“-Klasse falsch interpretiert wurde. Was Sinn ergibt, da alle vorfahrtsregelnden Schilder einzigartige Formen besitzen und dementsprechend leicht zu erkennen sind.

Jedoch ist ein Bild mit 30 mal 30 Pixeln keine realistische Darstellung von Schildern am Straßenrand. Daher wird das manipulierte Bild wieder auf die Größe des Originals hochskaliert.

Abbildung 36: Hochskaliertes Beispiel eines attackierten Vorfahrtstraßenschilds

Die Manipulation ist weiterhin schwierig für das menschliche Auge zuerkennen. Zur vollständigen Validierung und um sicher zu gehen, dass die Attacke weiterhin erfolgreich ist, wird das manipulierte Bild nun erneut auf die Inputgröße von 30 mal 30 Pixeln runterskaliert und wieder von dem CNN klassifiziert. Das Ergebnis bleibt das gleiche; Das CNN generiert weiterhin die Klasse „30 km/h“ als Output. Einziger Unterschied ist, dass sich das CNN nun 1,6% weniger sicher ist.

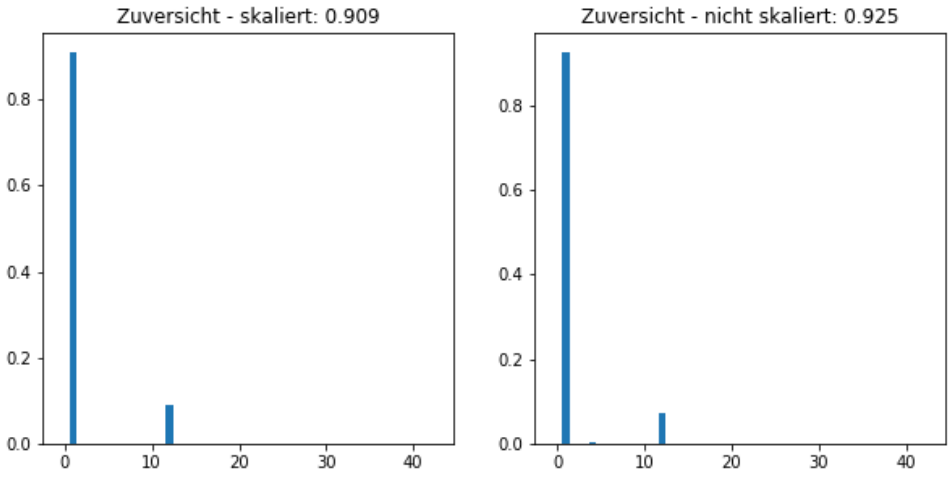


Abbildung 37: Zuversicht des skalierten Bildes (links) und des unskalierten Bildes (rechts)

Ein weiteres Beispiel für gelungene Attacken sind meist leicht dunkle Bilder:

Abbildung 38: Beispiel eines attackierten "80 km/h"-Schildes

Das ε besitzt einen sehr geringen Wert von nur 0,009 und die Manipulation ist nur bei genauem Hinsehen im schwarzen und weißen Teil des Bildes zu erkennen. Nach dem hoch und wieder runter skalieren ist sich das CNN sogar noch sicherer, dass es sich um ein „60 km/h“ Schild und nicht um „80 km/h“ handelt. In diesem Fall sind die Schilder der zwei Klassen sich aber auch ähnlich.

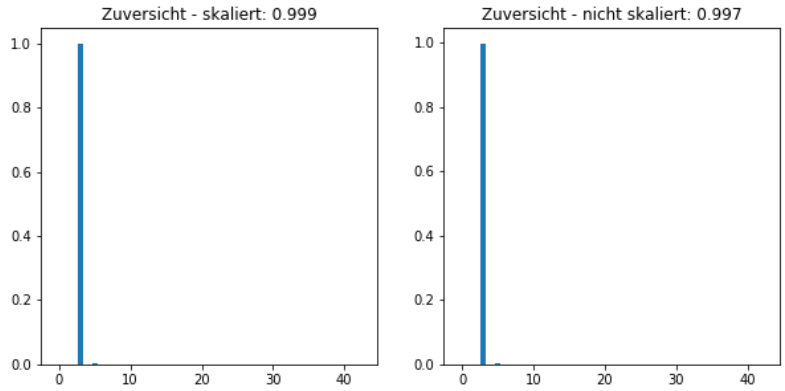


Abbildung 39: Zuversicht des skalierten „80 km/h“-Bildes (links) und des unskalierten „80 km/h“-Bildes (rechts)

Als negativ Beispiele sind solche Bilder zu nennen, bei denen die Manipulation zu offensichtlich ist, wie bei diesem Baustellen Schild:

Abbildung 40: Fehlgeschlagene Manipulation eines Baustellenschilds

Eine Manipulation des Bildes ist ganz klar zu erkennen und nach dem das Bild erst wieder hoch - und dann wieder runterskaliert wurde, findet wieder eine korrekte Klassifizierung statt, da es sich bei der Klasse 25 um die Klasse „Baustelle“ handelt und nicht „Rechtskurve“ wie es vor der Manipulation der Fall war.

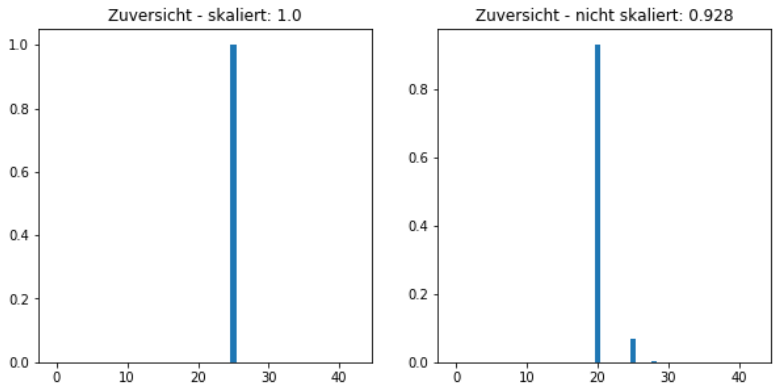


Abbildung 41: Zuversicht des „Baustellen“-Bildes, skaliert (links) und unskaliert (rechts)

4.2.3 Attacke auf das Datenset

Zur Beurteilung der gesamten Effektivität der Attacke wird zum Abschluss dieser Arbeit jedes der 12.630 Bilder des Testdatensets attackiert. Da das stetige Erhöhen des **ε** innerhalb der for-Schleife im Python-Code für ein einzelnes Bild schnell ein Ergebnis produziert, jedoch für das ganze Datenset zu einem Leistungsproblem des Computers werden kann, wird bei der nachfolgenden Attacke **ε** auf einen festen Wert von 0,03 gesetzt. Die Attacke dauert so auf einer sehr leistungsstarken Maschine circa fünf Minuten wohingegen sie inklusive der for-Schleife fast sechs Stunden auf dem gleichen leistungsstarken Rechner benötigt.

Nach der Manipulation sieht das Datenset beispielhaft wie folgt aus:

Abbildung 42: Beispiele aus dem manipulierten Datenset

Die über jedem Bild gegebenen Kennzeichen sind noch keine Vorhersage des CNNs, sondern noch die tatsächliche Beschriftung der Daten.

Zur Vollständigen Validierung der Attacke auf alle Testdaten wird nun, wie bereits in Kapitel 4.1.3 eine Confusion-Matrix erstellt.

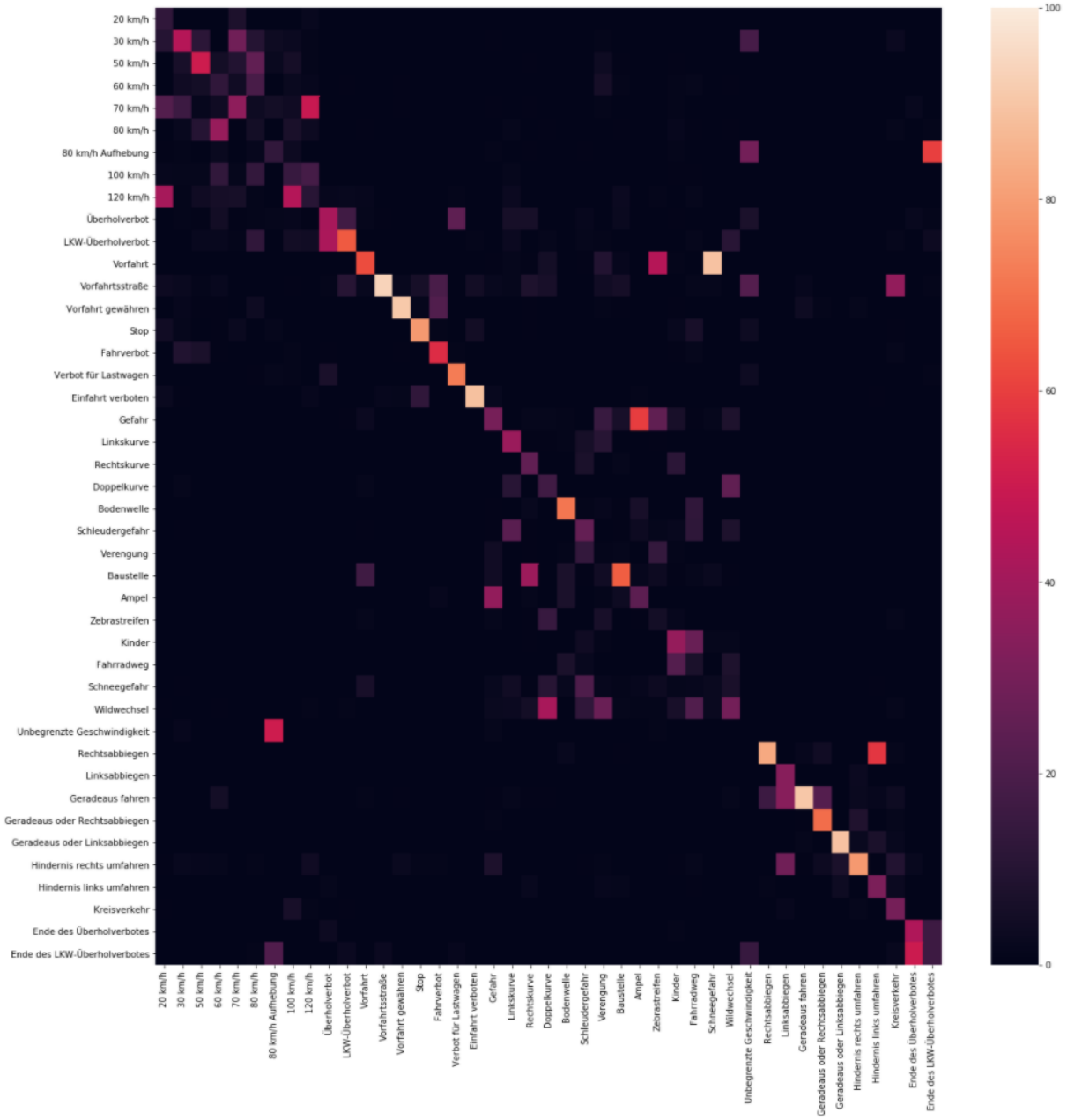


Abbildung 43: Confusion-Matrix nach der Attacke

Die Confusion-Matrix zeigt wie erwartet eine enorme Steigerung der Verwirrung und zugleich an manchen Stellen die erhoffte helle Linie, welche ein Indikator für gute Performance des Netzes ist. Zunächst fällt die großen unordentlichen Gebiete in der Mitte auf (Abbildung 44).

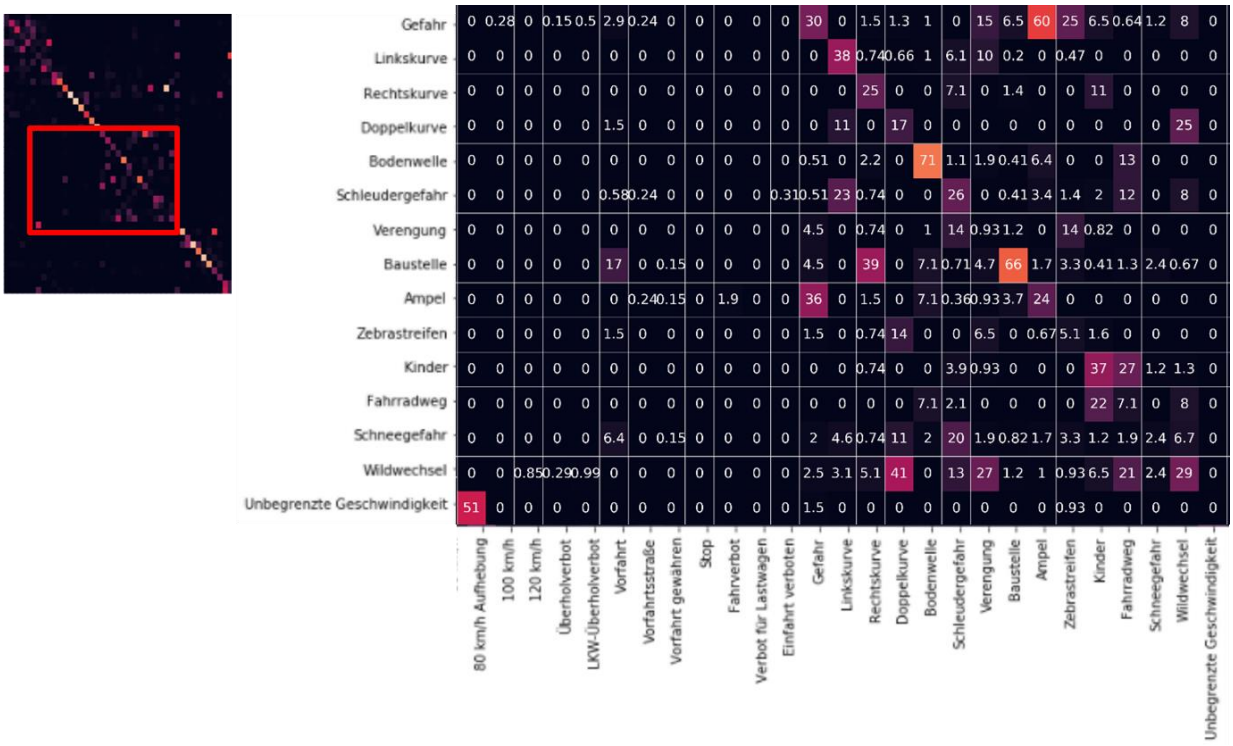


Abbildung 44: Ausschnitt der Confusion-Matrix (Mitte)

Die starke Manipulierbarkeit dieser Klasse und die daraufhin resultierende Verwirrung unteranderem untereinander kann mehrere Gründe haben. Zum einen spielt hier erneut das Bias des Datensets eine Rolle. Fast alle dieser Klasse hatten einen sehr geringen Anteil an der Gesamtmenge der Trainingsdaten. Einzige Ausnahme ist das Baustellenschild, welches von all diesen Klassen mit Abstand am meisten Bilder besaß und dementsprechend schlussendlich besser klassifiziert wurde. Weniger Verwirrung herrschte zudem auch beim dem Bodenwellenschild, dies hängt wahrscheinlich mit der besonderen Optik des Schildes zusammen.

Abbildung 45: Gefahrenschilder des Datensets

Während alle anderen Schilder von einem schwarzen Symbol in der Mitte dominiert werden, hat das Bodenschwellenschild eine große weiße Fläche mit nur einem schwarzen Symbol am Boden des Schildes. Zusätzlich zu erwähnen ist noch das

Schild der Klasse „unbegrenzte Geschwindigkeit“, welches aufgrund der großen Ähnlichkeit mit dem „80 km/h Aufhebung“ Schild und den wenigen Trainingsdaten nicht ein einziges mal richtig klassifiziert wurde und somit die schlechteste Performance von allen Klassen aufzuweisen hat.

Ein Bereich mit weniger Verwirrung ist von der Klasse „LKW-Überholverbot“ bis hin zur Klasse „Einfahrt verboten“ zu sehen.

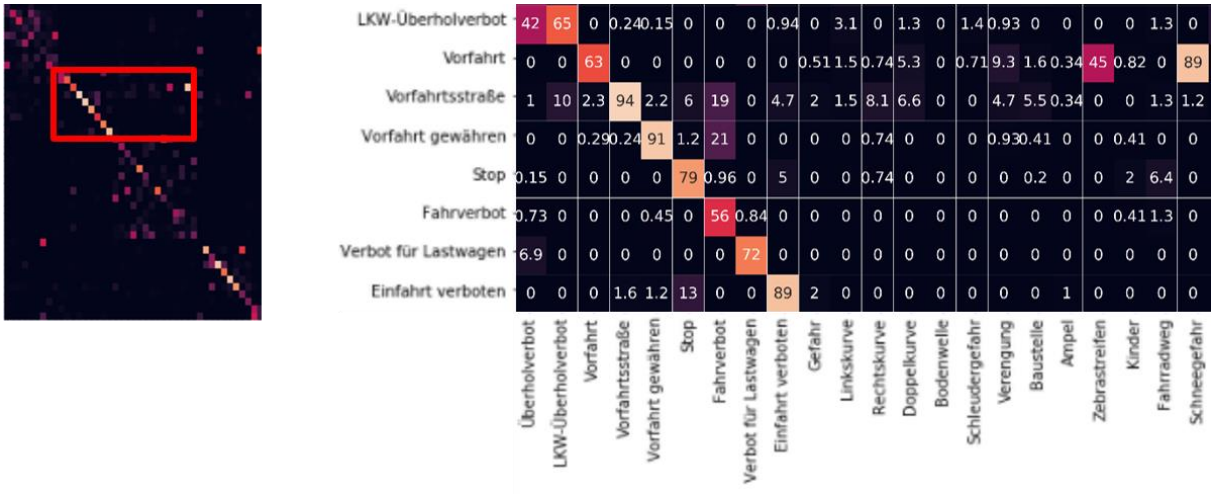


Abbildung 46: : Ausschnitt der Confusion-Matrix (Mitte/links)

Das Vorfahrtsstraßenschild wurde zum Beispiel in 94% Prozent der Fälle weiterhin richtig klassifiziert, was die beste Performance aller Klassen darstellt. Dies ist wahrscheinlich auf die einzigartige gelbe Farbgebung in der Mitte des Schildes zurück zu führen. Allgemein besitzen alle vorfahrtsregelnden Schilde besondere Formen, da sie selbst in verschneiten Zuständen, anhand ihrer Form für den Fahrer erkennbar sein sollten. Fatal können vor allem gegensätzliche Fehlklassifizierungen sein. Das Fahrverbotsschild wurde nur zu 56% richtig klassifiziert und zu jeweils 20% als Vorfahrt-Gewähren-Schild oder Vorfahrtsstraßenschild interpretiert. Dies sind nicht nur untereinander vollkommen unterschiedliche Schilder, sondern weisen auch auf vollkommen andere Situationen hin als das ursprüngliche Fahrverbotsschild.

Im letzten Teil der Analyse der Confusion-Matrix werden die Klassen von Schildern mit Geschwindigkeitsbegrenzungen betrachtet. Fast alle dieser Schilder besaßen mit am meisten Trainingsdaten und wiesen beim Training des Netzes eine sehr gute Performance auf.

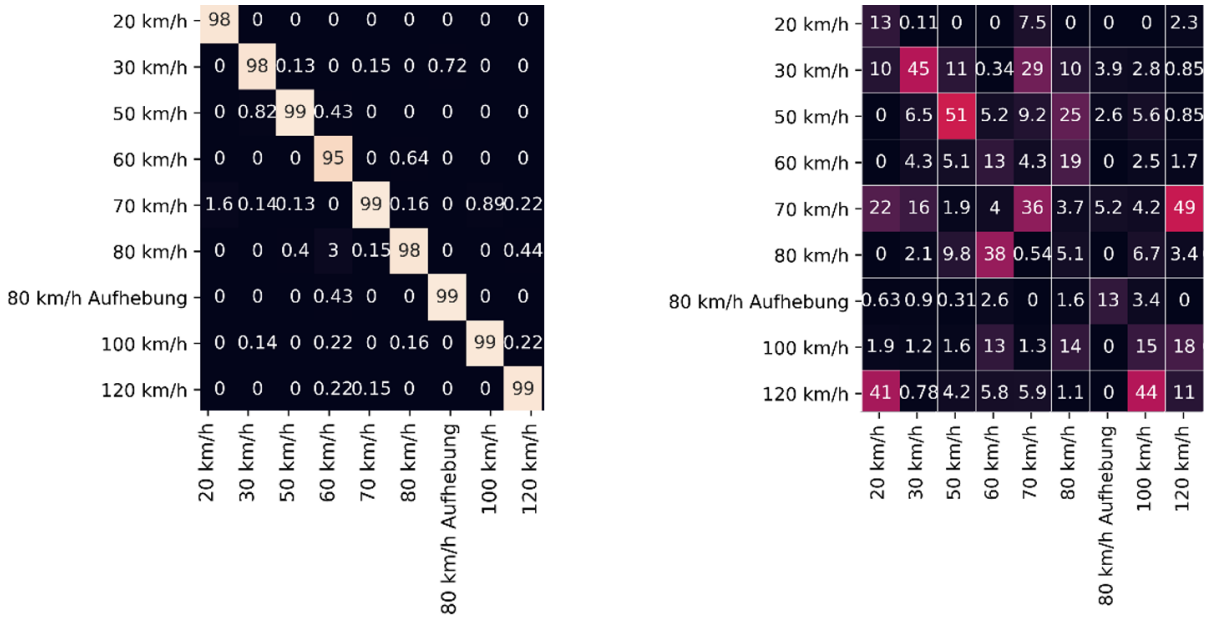


Abbildung 47: : Ausschnitt der Confusion-Matrix (oben links)

Trotzdem zeigte die Manipulation enorme Wirkung. Bei dem besten Schild, „50 km/h“, ist es zu einer 50:50 Chance geworden, ob das CNN das Schild richtig oder falsch klassifiziert. Viele Schilder weisen, sogar einen bessere Perfomance bezogenen auf eine andere Klasse aus als auf die richtige Klassifizierung.

5 Fazit und Ausblick

All diese Beispiele aus dem letzten Kapitel zeigen unterschiedliche Voraussetzungen für sowohl das Scheitern als auch das Gelingen einer Attacke, wobei es sich bei all dem um nahliegende Vermutung handelt. Da selbst die Funktionsweise eines neuronalen Netzes aufgrund des komplexen Aufbaus sehr schwer nachvollziehbar ist und von Training zu Training immer stark variieren kann, bietet dementsprechend auch die Funktionsweise der Attacke Spekulationsraum.

5.1 Fazit

Ein allgemeiner Blick auf beide Matrizen zeigt nochmal das ganze Ausmaß der Attacke.

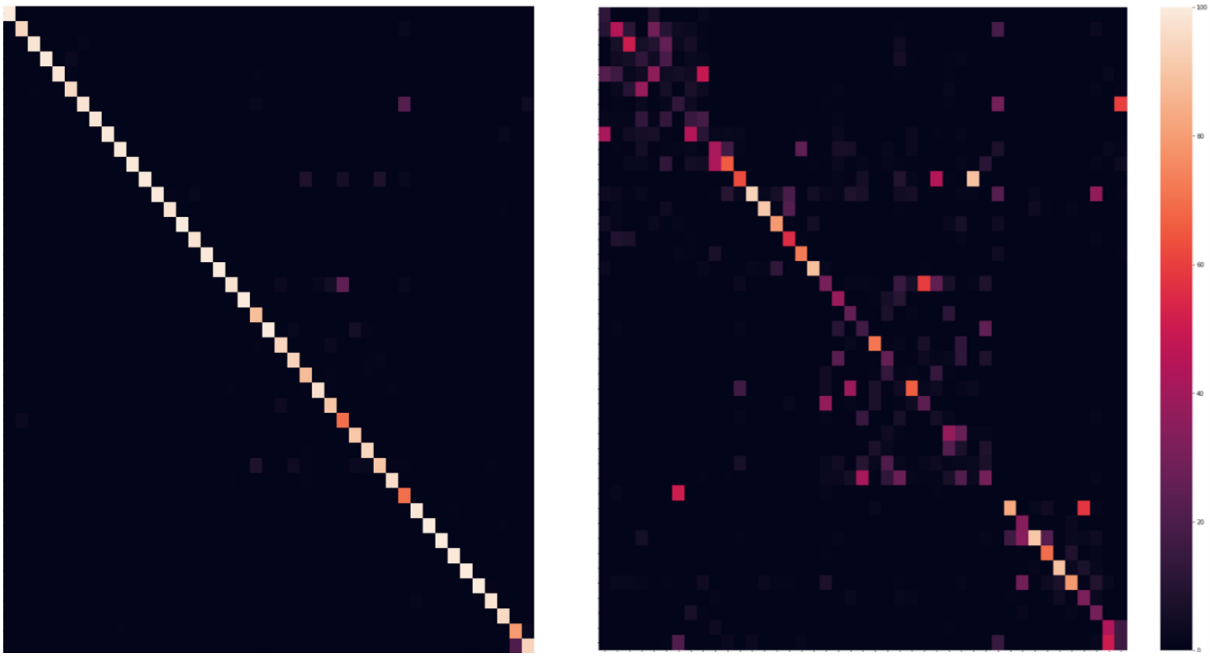


Abbildung 48: Confusion-Matrix vor der Attacke (links) und nach der Attacke (rechts)

Deutlich wird zum einem das bereits existierende Schwachstellen durch die Attacke verstärkt werden, jedoch aber auch sehr performante Klassen nicht sicher vor Manipulationen sind. Dies spiegelt sich auch in der allgemeinen Performance wider.

Nach dem Training lag die durchschnittliche Performance des CNNs bei über 96% und besaß nur vereinzelte Ausreißer. Aufgrund der Adversarial Attack fiel dieser Wert nun auf 42%. Das CNN nahm also mehr falsch Klassifizierung vor als richtige. Als Ausreißer sind jetzt eher performante Klassen wie die Vorfahrtsschild-Klasse zu bezeichnen.

Ein bereits angesprochener Punkt für die starke Manipulierbarkeit der Schilder ist das Bias des Datensets. Je größer das Bias eines Datensets ausfällt desto stärker schwankt die Menge der Trainingsdaten pro Klasse. Die damit einhergehende Verzerrung äußert sich bereits in einer Unsicherheit nach dem Training, welche nach der Attacke nur verstärkt wird.

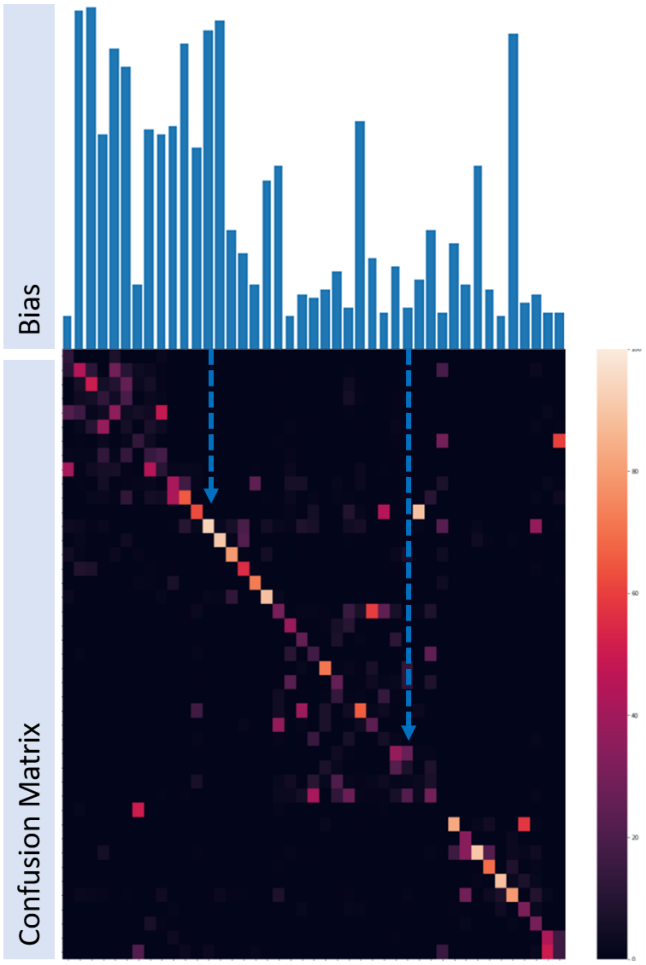


Abbildung 49: Confusion-Matrix im Kontext zum Bias

Wird der Bias in Kontext zur Confusion-Matrix, wie in Abbildung 49, gesetzt, wird deutlich, dass ein Bias zu Gunsten der Klasse zu einer besseren Klassifizierung führt. Besaß eine Klasse wenig Daten beim Training, stehen die Chancen gut für eine erfolgreiche Attacke. Ein solches Bias ist ein weit verbreitetes Problem und die Vermeidung immer noch in der Forschung. Jedoch ist bekannt, dass für bessere Trainingsresultate es möglichst vermieden werden sollte. Es kommen aber nicht alle Schilder im Straßenverkehr gleich häufig vor, sodass eine gleichmäßige Datenerhebung sich schwierig gestaltet. Hinzukommt, dass das verwendete Datenset noch längst nicht alle Schilder der deutschen StVo berücksichtigt und noch eine Vielzahl an Schildern nur für absolute Einzelfälle verwendet werden. Solche

Besonderheiten begegnet manchen vielleicht auch nur ein paar wenige Male in ihrem Leben und müssen von der Straßenschilderkennung dennoch fehlerlos identifiziert werden (Beispiel: Abbildung 50).

Abbildung 50: Zeichen 279-30: Ende der vorgeschriebenen Mindestgeschwindigkeit

Zusätzlich werde jährlich neue Straßenschilder hinzugefügt, welche in Deutschland zunächst nur wenig Verwendung finden und daher auch keine Möglichkeit bieten, dass Daten über sie gesammelt werden können (Beispiel: Abbildung 51).

Abbildung 51 Verbot des Überholens von einspurigen Fahrzeugen für mehrspurige Kraftfahrzeuge und Krafträdern mit Beiwagen

Auch wenn die meisten Fahrer, dass in Abbildung 51 dargestellte Schild, zum ersten Mal im Straßenverkehr sehen, können die meisten trotzdem abschätzen, dass es sich um eine Art des Überholverbots handelt. Diese Fähigkeit der Assoziation und Transferleistung fehlt den meisten künstlichen Intelligenzen zur Bilderkennung heute noch.

Wird nun eine andere künstliche Intelligenz, welche mit einem Bias-freiem Datenset trainiert wurde, attackiert, ist dies kein Indiz für eine schlechtere Performance der Attacke. Selbst bei sehr ausgewogener Datenlage zeigt das Beispiel der Schilder für Höchstgeschwindigkeiten, dass eine Attacke mit der FGSM Methode weiterhin sehr effektiv sein kann. Hier wird die Ähnlichkeit der Schilder, der künstlichen Intelligenz zum Verhängnis.

Und selbst wenn sowohl kein Bias als auch keine ähnlichen Daten Einfluss auf das Training haben, zeigen Beispiele wie das Stoppschild immer noch eine Anfälligkeit für eine Adversarial Attack. Das Bias oder die Ähnlichkeit können verstärkende Faktoren

für den Erflog einer Attacke sein, aber keine Voraussetzung. Zudem ist die in dieser beschriebener Fast Gradient Sign Methode bereits über sechs Jahre alt. In dieser Zeit wurden bereits effektivere und noch unauffälligere Methoden entwickelt.

Die in dieser Arbeit beschriebene Methode zeigt wie einfach eine Manipulation sein kann. Einer der größten Probleme bei der Verteidigung gegen Adversarial Attacks ist, je besser neuronale Netze und ihre Lernmethoden werden, desto effektiver werden die auf denselben Algorithmen basierende Attacken.

5.2 Ausbilck

Die Attacke war erfolgreich und gab viel über das CNN preis. Mittels einer Adversarial Attack können Unsicherheiten innerhalb von künstlichen Intelligenzen aufgezeigt werden und ein wenig Erkenntnisse, über die immer noch wenig nachvollziehbare Arbeitsweise von neuronalen Netzen gegeben werden. Vor allem wurden Fehler in den verwendeten Trainingsdaten deutlich, welche bei einem erneuten Training vermieden werden sollte. Zudem könnte eine Adversarial Training Methode zusätzlich Teil des Trainings werden. Bei einer solchen Methode werden zusätzlichen zu den normalen Trainingsdaten, manipulierte Daten generiert, welche ebenfalls ins Training mit einfließen. Problem an dieser Verteidigungsmethode ist es, dass sie in den meisten Fällen nur erfolgreich gegen die Methode der Attacke ist, mit welcher das manipulierte Trainingsdatenset erstellt wurde; Bei Anwendung einer neuen Methode stehen die künstlichen Intelligenzen wieder am Anfang. OpenAI, einer der führenden Forschungsunternehmen im Gebiet der künstlichen Intelligenz, beschrieb 2017 das Problem wie folgt: “Every strategy we have tested so far fails because it is not *adaptive*: it may block one kind of attack, but it leaves another vulnerability open to an attacker who knows about the defense being used.” [7]

Das Angreifen einer Verkehrsschilderkennung wurde in dieser Arbeit als Validierungsgrundlage gewählt, da es zusätzlich zum Aufzeigen der Methode, auch direkt eine Gefahr darlegt. Verkehrsschilder dienen heutzutage dem Menschen als Entscheidungsgrundlage für Fahrstil und Vorsicht. Wenn in der Zukunft sich Fahrzeuge nun autonom navigieren, sollte auch für diese ihre Entscheidungen, für das Bewegen durch den Straßenverkehr, auf Grundlage der Verkehrsschilder beruhen. Jedoch stellt dies weiterhin ein Hindernis dar und die Manipulationsmöglichkeit durch Dritte von außen, erschwert es zusätzlich. Die aktuellen führenden Modelle im Bereich der selbstfahrenden Autos sind nicht sicher vor Adversarial Attacks. So schaffte es

„Keen Security Lab“ ein Tesla mittels eines Adversarial Patches in die Gegenfahrbahn zu lenken.

Erst wenn flächendeckende Verteidigungsmechanismen gegen Adversarial Attacks gefunden werden, können künstliche Intelligenzen, welche mit optischem oder auditivem Input arbeiten, verlässliche Outputs liefern. Vorher können Sicherheitsfragen, auf Grundlage des Outputs eins neuronalen Netzes, nicht zuverlässig beantwortet werden.

6 Verzeichnis

6.1 Literaturverzeichnis

[1] Arunava Chakraborty. *Introduction to Adversarial Machine Learning*. https://blog.floydhub.com/introduction-to-adversarial-machine-learning/.

[2] Bhambri, S., Muku, S., Tulasi, A., and Buduru, A. B. A Survey of Black-Box Adversarial Attacks on Computer Vision Models.

[3] Brown, T. B., Mané, D., Roy, A., Abadi, M., and Gilmer, J. Adversarial Patch.

[4] Dave Gershgorn. 2017. *The data that transformed AI research—and possibly the world*. https://qz.com/1034972/the-data-that-changed-the-direction-of-ai-research-and-possibly-the-world/.

[5] Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and Harnessing Adversarial Examples.

[6] Gregory L. Wittel, S. Felix Wu. 2004. on attacking statistical spam filters.

[7] Ian Goodfellow, Nicolas Papernot, Sandy Huang, Rocky Duan, Pieter Abbeel, Jack Clark. 2017. *Attacking Machine Learning with Adversarial Examples*.

[8] Joao Gomes. 2018. *Adversarial Attacks and Defences for Convolutional Neural Networks*. Accessed 29 September 2020.

[9] Kasey Panetta. 2019. *5 Trends Appear on the Gartner Hype Cycle for Emerging Technologies, 2019*. https://www.gartner.com/smarterwithgartner/5-trends-appear-on-the-gartner-hype-cycle-for-emerging-technologies-2019/.

[10] Kourou, K., Exarchos, T. P., Exarchos, K. P., Karamouzis, M. V., and Fotiadis, D. I. 2015. Machine learning applications in cancer prognosis and prediction. *Computational and structural biotechnology journal* 13, 8–17.

[11] Mahi prashanth. 2020. *Understanding supervised, unsupervised, and reinforcement learning*. https://medium.com/@mahiprashanth866/understanding-supervised-unsupervised-and-reinforcement-learning-645c8f00757c.

[12] Ren, K., Zheng, T., Qin, Z., and Liu, X. 2020. Adversarial Attacks and Defenses in Deep Learning. *Engineering* 6, 3, 346–360.

[13] Richeek Awasthi. *Breaking Deep Learning with Adversarial examples using Tensorflow*.

[14] Sharif, M., Bhagavatula, S., Bauer, L., and Reiter, M. K. Accessorize to a Crime. Real and Stealthy Attacks on State-of-the-Art Face Recognition, 1528–1540. DOI=10.1145/2976749.2978392.

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

[16] Tencent Keen Security Lab. 2019. Experimental Security Research of Tesla Autopilot (Mar. 2019).

[17] *Ungenaue Schilder-Erkennung*. https://www.autozeitung.de/verkehrsschild-erkennung-test-195733.html.

[18] Xiao, C., Li, B., Zhu, J.-Y., He, W., Liu, M., and Song, D. Generating Adversarial Examples with Adversarial Networks.

6.2 Abbildungsverzeichnis

6.3 Formelverzeichnis