F A C H H O C H S C H U L E W E D E L

S E M I N A R A R B E I T

in der Fachrichtung

Medieninformatik

Ausarbeitung des Papers

Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition

Eingereicht von:	Mara Pape
Erarbeitet im:	5. Semester
Abgegeben am:	02. Juni 2018
Referent (FH Wedel):	Prof. Dr. Gerd Beuster Fachhochschule Wedel Feldstraße 143 22880 Wedel Tel. (0 41 03) 80 48-38

Inhaltsverzeichnis

Abkürzungsverzeichnis

DNN – Deep neural network
FRS – Face Recognition System
NPS - Non printability score

1. Einleitung

Heutzutage ist Gesichtserkennung kaum aus unserer Gesellschaft wegzudenken. Die Nutzung von Gesichtserkennungssoftware reicht von Handys im Privatgebrauch mit Gesichtserkennung als Spaßfaktor bis zu sicherheitskritischen Anwendungen wie etwa Zugangskontrollen. In sicherheitskritischen Kontexten ist deswegen besonders relevant, wie und in welchem Ausmaß solche Systeme irregeführt und angegriffen werden können.

Diese Ausarbeitung beruht im Allgemeinen auf dem Paper *Accessorize to a crime – Real and stealthy Attacks on State-of-the-Art Face Recognition* [10]. Hier geht es vor Allem darum, aktuelle Gesichtserkennungs-Software irrezuführen. Das wird unter zwei Gesichtspunkten betrachtet: Zum einen soll der Angriff auf so ein Gesichtserkennungssystem möglichst unauffällig, aber gleichzeitig auch möglichst unaufwendig für den Angreifer zu realisieren sein. Dabei werden verschiedene Formen der Täuschung des Systems unterschieden: Entweder möchte der Angreifer als ein spezielles Ziel missklassifiziert werden (Impersonation), als ein beliebiges anderes Ziel (Dodging) oder „unsichtbar" für das System sein. Dodging und Impersonation werden mit Hilfe eines Brillengestells realisiert. Die Herausforderung des Angreifers besteht bei einem physischen Angriff im Besonderen darin, dass er den Input, der vom System ausgewertet wird, nicht direkt beeinflussen kann, wie das etwa bei Phishing-Mails der Fall ist.

Unauffälligkeit

Die erste Bedingung an einen Angriff auf eine Gesichtserkennungs-Software lautet: Der Angreifer soll möglichst unauffällig seinen Angriff praktizieren. Das bedeutet, von Menschen, die sich in unmittelbarer Entfernung zum Angriff befinden, soll nicht erkannt werden, dass es sich um einen Angriff handelt. Das gilt auch für *indirekte Beobachter:* Jemand, der sich z.B. Aufnahmen einer Überwachungskamera ansieht, soll ebenfalls nichts von einem Angriff bemerken. Es ist also wichtig, sich im Klaren darüber zu sein, mit welchen Mitteln man so ein Gesichtserkennungssystem täuschen kann, um entsprechende Angriffe, selbst wenn sie unauffällig sind, zu enttarnen oder vorzubeugen.
Ein weiterer Grund für den Angreifer, möglichst unaufällig zu agieren, besteht in der Abstreitbarkeit. Wenn also eine Person nicht (korrekt) von einer Gesichtserkennungssoftware erfasst wird, kann diese bestreiten, das System mutwillig getäuscht zu haben und sich so aus der Verantwortung ziehen. Hier kann sie sich z.B. auf Zufall berufen oder der Gesichtserkennungssoftware die Schuld geben, nicht bzw. nicht korrekt klassifiziert worden zu sein.

Physische Realisierung

Die zweite Bedingung, die an den Angriff auf eine Gesichtserkennungssoftware geknüpft ist, ist eine einfache physische Realisierung. Das bedeutet, sie soll günstig und einfach herzustellen sein. Außerdem soll auch auf aktuellen Systemen auf dem neuesten Stand der Technik noch ein Angriff möglich sein. Es werden zwei Angriffsarten unterschieden, die jeweils ein Face Recognition System angreifen: Impersonation und Dodging.

Impersonation

Bei einem Impersonation-Angriff versucht der Angreifer, den Input in die Gesichtserkennungssoftware so zu manipulieren, dass ein spezielles anderes Gesicht als sein Gesicht missinterpretiert wird. Das kann z.B. bei einer Zugriffskontrolle das Gesicht einer Person mit den Zugriffsrechten sein. Denkbar wäre auch, sich an verschiedenen geografischen Orten als eine einzige Person auszugeben und so seinen wirklichen Standort zu verschleiern.

Dodging
Bei einem Dodging-Angriff versucht der Angreifer, den Input in die Gesichtserkennungssoftware so zu manipulieren, dass er als ein beliebiges anderes Ziel missklassifiziert wird. Das ist vor allem interessant, da dieser Angriff besonders unauffällig ist. Der Angreifer muss sein Erscheinungsbild also nur geringfügig manipulieren, damit der Angriff gelingt. Diese Form des Angriffs ist auch für Personen interessant, die sich lediglich einer Gesichtserkennung entziehen wollen.

Um den Einstieg in dieses Thema zu erleichtern, wird zusätzlich kurz die Funktionsweise von sowohl Face Detection als auch Face Recognition beschrieben und gegenübergestellt. Im Anschluss werden einige Annahmen bezüglich des Angreifers sowie des anzugreifenden White Box DNN (Deep Neural Network) getroffen. Außerdem wird genauer erläutert, mit welchen Methoden und mit welchen Daten die Neuronalen Netzwerke eingelernt wurden und wie man dieses täuschen kann. Hier wird der Angriff mit Hilfe eines Brillengestells genauer erläutert. Anschließend folgt die Beschreibung und Auswertung des Experiments des Papers. Es behandelt drei verschiedene Versuche.

1. Angriff auf ein White Box DNN (also ein Deep Neural Network, dessen Inneres dem Angreifer bekannt ist)
2. Angriff auf ein Black Box DNN (also ein Deep Neural Network, dessen Inneres dem Angreifer nicht oder nur teilweise bekannt ist)
3. „Unsichtbarkeit" gegenüber eines Face Detection Systems

Der erste Punkt ist das „Hauptexperiment", die anderen bilden dazu eine Erweiterung. Das „Hauptexperiment" ist in zwei Teile geteilt. Im ersten Teil kann der Angreifer das Bild digital auf Pixelebene manipulieren. Dies dient auch als Demonstration, mit welchem minimalen Input sich eine Missklassifizierung erzielen lässt. Der zweite Angriff erfolgt nicht digital, d.h. der Input kann nicht exakt kontrolliert werden, eine Manipulation auf Pixelebene ist nicht möglich.

2. Face Recognition vs. Face Detection

In diesem Punkt wird das Augenmerk auf die Gegenüberstellung von Face Detection und Face Recognition gelegt.
Zunächst werden jedoch Face Detection und Face Recognition einzeln vorgestellt. Außerdem wird eine kurze Beschreibung des Viola-Jones-Algorithmus[12] unter dem Kapitel 2.1 folgen, da dieser für das Experiment „Angriff auf ein Face Detection" wieder aufgegriffen wird. Somit wird das Verständnis für diesen Angriff erleichtert.

2.1 Face Detection

Face Detection ist eine spezielle Form der Objekterkennung. Es sucht in einem Input nach speziellen Objekten, z.B. Gesichtern. Es ist mit so einem Objekterkennungsalgorithmus also genauso möglich, Verkehrsschilder, Fußgänger oder andere beliebige Objekte zu erkennen. In diesen Bereichen wird häufig auf den Viola-Jones-Algorithmus zurückgegriffen. Gleichzeitig ist er auch als Face Detection Algorithmus führend. Der Algorithmus ist deshalb so bekannt und häufig genutzt, da er als präzise und besonders schnell gilt.

2.1.1 Viola Jones in a Nutshell

Der Algorithmus besteht im Wesentlichen aus vier Schritten. Diese werden in den folgenden Punkten etwas genauer ausgeführt.

1. Integrales Bild erstellen
   Der Algorithmus arbeitet mit Graustufenbildern. Jedes Pixel hat also nur eine Intensität. Daher kann man einfach ein integrales Bild errechnen. Jedes Pixel enthält dabei die Summe von sich selbst mit den Vorgängern addiert. *Abbildung 1* enthält ein Beispiel. Dabei muss man die doppelt addierten Pixel wieder subtrahieren. Für das Pixel rechts unten ergibt sich aus der Summe der Inten-sität des Pixels selbst, des linken und oberen Nachbarn sowie einer Subtraktion des Pixels oben links. Das ergibt also die Gleichung $7+4+5-3=13$.

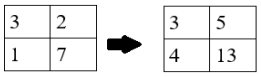

Abbildung 1: Summenbildung für ein integrales Bild

2. Haar-Feature Auswahl
   Haar-Features sind in folgendem Bild gezeigt. Damit lässt sich eine Differenz zwischen den einzelnen Blöcken eines Haar-Features ausrechnen, bei Haar Feature 1 und 2 also Schwarz und Weiß. Dafür verwendet man jeweils ein integrales Teilbild für den entsprechenden Block und subtrahiert diese voneinander.

Abbildung 2: Haar-Features, für den Viola-Jones-Algorithmus
Quelle:https://upload.wikimedia.org/wikipedia/commons/3/31/VJ_featureTypes.svg

3. AdaBoost Training
   AdaBoost trainiert die Klassifikatoren, die die Entscheidung treffen, ob es sich um ein Gesicht handelt. Dafür wird eine Teilmenge aller Haar-Features ausgewählt, welche sich besonders gut eignen, ein Gesicht zu identifizieren. Der Viola-Jones-Algorithmus ist jedoch nicht objektspezifisch, d.h. es können mit dem gleichen Vorgehen auch andere Objekte erkannt werden wie z.B. Straßenschilder. Dabei muss nur der Klassifikator anders trainiert werden.

4. Kaskadierende Klassifikatoren
   Der Viola-Jones-Algorithmus nutzt eine Verkettung von Klassifikatoren. In aufsteigender Reihenfolge werden diese immer komplexer. Ein einfacher Klassifikator kann ein Teilbild akzeptieren, dann wird der nächste Klassifikator darauf angewendet, oder es ablehnen. Dann wird das Teilbild nicht weiter beachtet. Da in einem Bild sehr viele solcher Teilbilder entstehen, die nicht von Interesse sind, bringt dieses Verfahren einen enormen Effizienzgewinn, da uninteressante Bildbereiche schnell verworfen werden.

2.2 Face Recognition

Face Recognition versucht, ein Gesicht einer speziellen Person zuzuordnen. Dafür benötigt man zunächst einmal ein Neuronales Netz, welches man mit Datensätzen trainieren kann. Ein Datensatz entspricht dabei vielen Bildern einer Person, die in das Neuronale Netz gegeben werden. Mit Hilfe dieser Trainingsdaten werden intern Klassen erstellt, für jede Person eine. Diese Klassen müssen in irgendeiner Form verwaltet werden, man benötigt also zusätzlich eine Datenbank.
Wird ein Gesicht erkannt, wird es mit den Gesichtern in der Datenbank abgeglichen. Das Resultat ist dann eine Wahrscheinlichkeit. Diese gibt an, wie wahrscheinlich das Gesicht des Inputs auf eine spezielle Klasse in der Datenbank passt.

2.3 Gegenüberstellung Face Detection und Face Recognition

Face Detection und Face Recognition würde man im Deutschen beide auf das gleiche Wort abbilden, nämlich auf Gesichtserkennung. Trotzdem gibt es einige große Unterschiede. Um diese Unterschiede ersichtlich zu machen, werden in dieser Ausarbeitung weiterhin die englischen Begrifflichkeiten verwendet.
Der wohl grundlegendste Unterschied ist das Ziel, das die jeweilige Technik verfolgt. Das Ziel von Face Detection besteht darin, ein oder mehrere Gesichter zu erkennen. Face Recognition muss zunächst natürlich auch erst einmal ein Gesicht erkennen. Anschließend erfolgt die Analyse und Zuordnung des Gesichts anhand der vorhandenen Daten, also anhand der Personen, die das System kennt. Daraus wird ersichtlich, dass der gesamte Face Detection-Prozess ein Teil des Face Recognition-Prozesses ist.
Ein weiterer Unterschied ist die Menge an Trainingsdaten, die ein System benötigt. Es ist naheliegend, dass ein Face Recognition System sehr viele Trainingsdaten von den Personen benötigt, die es erkennen soll. Allerdings sollte man nicht vergessen, dass auch ein Neuronales Netz, welches nur Face Detection zum Ziel hat, gewissen Trainingsdaten benötigt, um ein Gesicht zu erkennen, jedoch bedeutend weniger als Face Recognition Systeme.

3. Hintergrund zum Angriff auf ein White Box DNN

In Bezug auf das folgende Experiment (Kapitel 5) müssen ein paar Festlegungen getroffen werden bezüglich des Angreifers und des Systems. Als Beispiel wäre z.B. der Kenntnisstand des Angreifers zu nennen. In Kapitel 3.2 folgt eine genaue Erläuterung, mit welchen Deep Neural Networks gearbeitet wird, wie und mit welchen Trainingsdaten sie trainiert wurden.

3.1 Annahmen zu Angreifer und System

Wie bereits angedeutet, wird ein Angriff auf ein White Box DNN durchgeführt. Das bedeutet, die innere Struktur des Systems ist öffentlich, also auch dem Angreifer bekannt. Das bezieht auch die Parameter mit ein, die das White Box DNN verwertet.
Obwohl die Informationen des Systems öffentlich sind, können sich Angreifer durch verschiedene Wissensstände unterscheiden. Wir nehmen hier an, dass dem Angreifer der verwendete Algorithmus bekannt ist. Auch der Merkmalsraum, also die Merkmale, die das White Box DNN analysiert, ist dem Angreifer bekannt. Dieser ist ohnehin meist öffentlich.
Wir nehmen außerdem an, dass das White Box DNN bereits trainiert wurde. Das bedeutet, der Angreifer kann das System nicht mit „falschen" Daten trainieren. Er kann die Trainingsdaten also beispielsweise nicht durch seine eigenen, manipulierten Daten ersetzen oder das System mit falsch etikettierten Inputs „vergiften", um so eine höhere Wahrscheinlichkeit, sein Ziel zu erreichen, zu erzielen.

3.2 White Box DNN trainieren

Für das Experiment *Angriff auf ein White Box DNN* (Kapitel 5) werden drei verschiedene DNNs verwendet. Diese werden im Folgenden als DNN_A, DNN_B und DNN_C bezeichnet. DNN_A wurde mit Hilfe von Bildern von Prominenten trainiert. Um einen Angriff mit einem Brillengestell durchzuführen, benötigt es jedoch *reale* Personen. Das wird vor allem beim Dodging wichtig, denn das System muss die Person unter *normalen* Umständen, also solche, die keinen Angriff betreffen, korrekt klassifizieren können. Sonst wäre ein Dodging-Angriff immer erfolgreich. Auch ein Impersonation-Angriff wird durch das Erkennen des Angreifers erschwert und wird zusätzlich als realistischer eingestuft. Daher wurden zwei weitere DNNs, DNN_B und DNN_C, trainiert.

DNN_A: Bei diesem DNN handelt es sich um ein bereits fertig trainiertes DNN. Es wurde von Parkhi et al. trainiert, um 2622 Prominente zu erkennen. Es wurden ca. 1000 Bilder pro Berühmtheit verwendet, insgesamt also ca. 2,6 Millionen Bilder. Das scheint zunächst einmal viel, ist mit anderen vergleichbaren DNNs jedoch relativ wenig: z.B. dem Face Recognition System von Google (ca. 200 Millionen Bilder)[9]. Als Quelle wird hier jedoch eine Menge von YouTube-Videos verwendet, daher kommt die hohe Anzahl an Bildern zustande. Insgesamt werden hier nur 1595 Personen erkannt, also deutlich weniger als bei DNN_A.

DNN_B und DNN_C: Da DNN_A nur Leute erkennt, die für das Experiment nicht verfügbar waren, ist es zum Testen von physischen Angriffen nicht besonders geeignet. Deswegen wurden zwei weitere DNNs trainiert: DNN_B und DNN_C.

DNN_B wurde trainiert, um zehn verschiedene Personen zu identifizieren. Fünf Personen davon sind Berühmtheiten (Aaron Eckhart, Brad Pitt, Clive Owen, Drew Barrymore sowie Milla Jovovich). Die Trainingsbilder stammen von einer Online-Datenbank PubFig[4]. Die letzten fünf Personen setzen sich zusammen aus drei Personen der Forschungsgruppe sowie zwei weiteren Freiwilligen aus dem Umfeld der Forschungsgruppe. Insgesamt kennt das DNN jetzt je fünf männliche und weibliche Personen in einer Altersspanne zwischen 20 und 53 Jahren.

DNN_C wurde mit einer größeren Menge an zu identifizierenden Personen trainiert. Insgesamt besteht die Menge aus 140 Prominenten, deren Trainingsdaten ebenfalls aus der Online-Datenbank PubFig[4] stammen sowie den gleichen drei Autoren, die auch DNN_B kennt.

Beide DNNs wurden mit Hilfe von *Transfer Learning*[13] trainiert. Hierbei wird ein bereits trainiertes DNN als Grundlage des Lernprozesses verwendet. Dieses *Basis-DNN (hier DNN_A)* muss jedoch eine ähnliche Aufgabe gehabt haben wie das zu trainierende. Man kann also für DNN_B und DNN_C kein DNN als Grundlage verwenden, welches z.B. Fahrräder erkennt. In diesem Fall ist jedoch die Ähnlichkeit gegeben, es kann also Transfer Learning genutzt werden, was den Vorteil bietet, weit weniger Trainingsdaten in das zu trainierende DNN geben zu müssen.

4. Neuronale Netze täuschen

Obwohl Deep Neural Networks sehr gut abschneiden bei Klassifizierungsaufgaben und sogar Menschen übertreffen bei der *Labeled Faces in the Wild* [3] Challenge (LFW), können selbst minimale Manipulationen des Inputs das Ergebnis verfälschen [11]. Diese minimalen Manipulationen müssen nicht einmal für das menschliche Auge sichtbar sein. Dieser Sachverhalt wird auch in Kapitel 5.1 demonstriert.
Der Input in das DNN wird im Folgenden als x bezeichnt, $f(x)$ wird das Ergebnis der Auswertung des DNNs sein. Um einen Angriff erfolgreich zu realisieren, wird die kleinste bzw. unauffälligste Manipulation r gesucht, bei der der Angreifer sein Ziel erreicht, also $x+r$ zu der gewünschten Klasse c_t auswertet. Um die minimale Störung des Inputs zu finden, muss folgendes Minimierungsproblem gelöst werden:

$$argmin_r\left(\left|f\left(x+r\right)-h_t\right|+k\left|r\right|\right)$$

Dabei ist $x+r\in[0,1]$, $f(x+r)$ wertet dabei zu einer Wahrscheinlichkeit aus dem Bereich $[0,1]^N$ aus. N bezeichnet die Anzahl der verschiedenen Klassen, die das DNN erkennen kann, k bezeichnet eine Konstante, die das Verhältnis von Unauffälligkeit und Missklassifizierung modellieren soll. Wenn $k>1$, wird die Unauffälligkeit stärker gewertet als die Missklassifizierung, wenn $k<1$ entsprechend umgekehrt. h_t bezeichnet einen 1-aus-N-Vektor der Zielklasse c_t. Je kleiner die Differenz zwischen $f(x+r)$ und h_t , desto wahrscheinlicher wird eine Missklassifi-

zierung als Person der Zielklasse c_t .

$|...|$ beschreibt eine Normfunktion, wobei $...$ hier als Platzhalter dient. Sind $f(...)$ und $|...|$ differenzierbar, kann das Optimierungsproblem gelöst werden. Für DNNs ist das gegeben, auch die Normfunktion ist differenzierbar.

Um die Exaktheit der Auswertung des DNNs zu bewerten, wird, wie Parkhi et al. vorschlagen, die Softmaxloss-Bewertung[8] genutzt. Die Softmaxloss-Funktion nimmt zwei Parameter entgegen: die Klasse c_x des Inputs x sowie die Auswertung des Inputs $f(x)$. Softmaxloss ist wie folgt definiert:

$$softmaxloss(f(x), c_t) = -\log\left(\frac{e^{\langle h_{c_x}, f(x)\rangle}}{\sum_{c=1}^{N} e^{\langle h_c, f(x)\rangle}}\right)$$

$\langle ..., ... \rangle$ ist hierbei das Skalarprodukt. N die Anzahl der Klassen, h_c ist einer der Vektoren aus der Zielklasse c und $...$ ist wieder ein Platzhalter.

Je weiter die beiden Vektoren auseinanderliegen, desto kleiner wird das Skalarprodukt. Bei orthogonalen Vektoren, also bei maximaler Distanz zwischen zwei Vektoren, wird das Skalarprodukt 0. Da das Ziel im Folgenden ist, ein Minimum zu finden, wird der negativen Logarithmus verwendet. Das Ergebnis der Softmaxloss-Funktion ist klein, wenn die Klassifizierung korrekt ist, und hoch, wenn sie entsprechend falsch ist.

Die Softmaxloss-Bewertung wird zur Formulierung der Zielsetzung von Impersonation und Dodging verwendet.

Impersonation Um einen Impersonation-Angriff zu realisieren, muss die Differenz zwischen $f(x+r)$ und h_t möglichst gering sein. Das zu lösende Minimierungsproblem wird mit Hilfe des Softmaxloss-Scores wie folgt formuliert:

$$argmin_r\, softmaxloss(f(x+r), c_t)$$

Es wird also nach einer minimalen Manipulation r des Inputs x gesucht, damit $f(x+r)$ zu c_t auswertet.

Dodging Bei einem Dodging-Angriff geht es nur darum, als ein beliebiges anderes Ziel klassifiziert zu werden beziehungsweise nicht als die „richtige" Klasse. Daraus resultiert ein kleines Skalarprodukt, da die Zielklasse sich möglichst stark von der Klasse des Inputs unterscheiden soll. Daher ist auch der Softmaxloss-Score hoch. Das Optimierungsproblem für Dodging wird wie folgt definiert:

$$argmin_r\left(-softmaxloss(f(x+r), c_t)\right)$$

Es wird also einfach das Vorzeichen umgekehrt und so wieder ein Minimierungsproblem modelliert. Um diese Optimierungsprobleme zu lösen, wird das Gradienten-Verfahren verwendet. Das Gradientenverfahren ist ein iteratives Vorgehen. In den Experimenten wird die Zahl der Iterationen jeweils beschränkt, es handelt sich dabei also möglicherweise nur um eine Annäherung an das Minimum, je nach dem, ob der Algorithmus durch die Anzahl der Iterationen beendet wurde oder nicht. Bei jeder Iteration wird die Lösung r aktualisiert, und zwar in Richtung des negativen Gradienten, bis die Differenz von r vor und nach der Aktualisierung so klein wird, dass sie keine Rolle mehr spielt.

4.1 Physische Realisierung ermöglichen

Da folgende Experimente auch physisch realisierbar sein sollen, also nicht auf Ebene der Manipulation einzelner Pixel, wird besonderer Fokus auf folgende vier Punkte gelegt. Diese werden anschließend einzeln erläutert.

1. Accessoires für das Gesicht
2. Robustheit
3. „Smoothness"
4. Druckbarkeit

4.1.1 Nutzung von Accessoires

Abbildung 3: Brillengestell, welches als Schablone für die Manipulation der Brille dient. Clker-Free-Vector-Images/CC0/ https://goo.gl/3RHKZA

Um Unauffälligkeit bei einem Angriff zu gewährleisten, werden vom Angreifer nur Accessoires für das Gesicht verwendet, genauer ein manipuliertes Brillengestell. Gesichtsaccessoires lassen sich sehr leicht sowie kostengünstig beschaffen. Das ist ein großer Vorteil der Implementierung der Angriffe. Die Vorderseite des Brillengestells, welches für die Angriffe verwendet wird, wurde mit einem einfachen Tintenstrahldrucker (Epson XP-830) auf Glanzpapier gedruckt. Als Schablone diente dabei ein einfaches „Geek"-Brillenmodell. Sie wurde eingefärbt, gedruckt, anschließend ausgeschnitten und auf einem echtes Brillengestell fixiert. Wenn der Angreifer das Brillengestell aufsetzt, werden ca. 6,5% des Gesamtbildes durch die Vorderseite der Brille manipuliert.
Ein weiterer Vorteil von Angriffen mit Hilfe von Accessoires ist gegeben: die Abstreitbarkeit des Angriffs. Wenn man z.B. eine Maske aufsetzen würde, um ein Face Recognition-System in die Irre zu führen, wäre offensichtlich, dass der Angreifer versucht, das System zu manipulieren. Bei einer Brille wäre das nicht der Fall, da Brillen häufig anzutreffen sind. Man kann also Angreifer nicht zwingend von einem Nicht-Angreifer unterscheiden .
Um die richtige Farbe für die Manipulation des Inputs zu finden, wird wieder der Gradient Descent-Algorithmus verwendet.

4.1.2 Robustheit

Es ist sehr unwahrscheinlich, dass zwei verschiedene Bilder des gleichen Gesichts identisch aussehen. Das liegt vor allem daran, dass der Angreifer menschlich ist. Er bewegt sich, d.h. Pose, Gesichtsausdruck oder auch die Lichtverhältnisse unterscheiden sich meist leicht. Um den Angriff trotzdem erfolgreich durchzuführen, muss der Angreifer eine Manipulation finden, die nicht von einem speziellen Input abhängt, sondern die für eine Menge aus möglichen Input-Bildern funktioniert. Diese Menge wird als X bezeichnet. Für jedes Bild $x \in X$ muss der Angreifer eine Manipulation r finden, die eine Missklassifizierung auslöst bzw. wahrscheinlicher macht. l ist dabei die Zielklasse. Daraus folgt wieder ein Optimierungsproblem:

$$argmin_r \sum_{x \in X} softmaxloss(f(x+r), l)$$

4.1.3 Smoothness

Echte Fotografien haben flächenweise sanfte, natürliche Farbverläufe ohne Stufen. Das wird hier als *Smoothness* bezeichnet. Damit die Abstreitbarkeit und Unauffälligkeit weiterhin gegeben ist, wird das für die Experimente ebenfalls benötigt. Des Weiteren kann die Kamera das Bild eventuell nicht exakt einfangen, sondern ein Rauschen hinzufügen oder scharfe Kanten, bei denen nah beieinander liegende Pixel starke Farbunterschiede aufweisen, verwischen. Damit wäre möglicherweise die Manipulation nicht mehr wirksam. Wenn also dicht beieinander liegende Pixel ähnliche Farben haben, ist der Input *smooth*. Dazu wird die *total variation*[5] als ein Maß verwendet, wie sehr sich die Pixel von ihren jeweiligen Nachbarn unterscheiden. Die *total variation TV* ist folgendermaßen definiert:

$$TV(r)=\sum_{i,j}\sqrt{(r_{i,j}-r_{i+1,j})^2+(r_{i,j}-r_{i,j+1})^2}$$

$r_{i,j}$ ist dabei ein Pixel in der Manipulation r an der Koordinate (i,j). Man kann erkennen, dass TV(r) kleiner wird, je ähnlicher sich benachbarte Pixel sind, der Input also *smooth* ist.

4.1.4 Druckbarkeit

Grundlage dieses Unterpunktes ist das Problem der Druckbarkeit. Displays und Drucker haben einen speziellen Gamutbereich. Das bedeutet, dass die Geräte nicht alle Farben anzeigen beziehungsweise drucken können, sondern nur diese, die auch in ihrem Gamut-Bereich liegen. Dieser Bereich umfasst meist nur eine Teilmenge des wahrnehmbareren Farbraums. Abbildung 4 zeigt beispielhaft verschiedene Gamutbereiche.

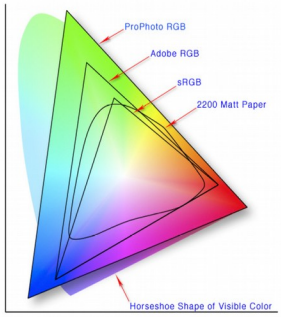

Abbildung 4: Beispiel verschiedener Gamutbereiche Quelle: https://upload.wikimedia.org/wikipedia/commons/3/37/Colorspace.png

Damit der Angriff auf Gesichtserkennungssysteme trotzdem gelingen kann, muss also die Farbgebung der Manipulation, also des Brillengestells, auf das Gamut des Druckers angepasst werden. Dazu wird der *Non-printability-Score* als Maß für die Druckbarkeit eingeführt.

P ist die Menge aller druckbaren Farben, $\hat{p}$ ist ein einzelnes Pixel. Der *Non-printability-Score NPS* für ein Pixel wird wie folgt definiert:

$$NPS(\hat{p})=\prod_{p\in P}|\hat{p}-p|$$

Wenn $\hat{p}\in P$ oder $\hat{p}$ und $p\in P$ sich sehr ähnlich sind, ist der *Non-printability-Score* klein, die Farbe liegt also im Gamut des Druckers oder in dessen Nähe. Das wird besonders deutlich bei $\hat{p}\in P$. Hier wird für irgendeinen Farbwert $p\in P$ die Differenz im Produkt zu 0. Damit wird der ganze *NPS* zu 0. Um den *NPS* der Manipulation, also hier der Brille, zu beschreiben, wird für jeden Pixel der Manipulation der *NPS* gebildet und anschließend summiert.

Um zu sehen, welche Farben überhaupt gedruckt werden können, wurde eine Farbpalette aus gedruckt, die ca. 20% des RGB-Farbraums enthält. Diese wurde anschließend mit einer Kamera aufgenommen. So kann bestimmt werden, welche Farben gedruckt werden können. Die Anzahl der verschiedenen Farben ist sehr hoch und muss daher etwas verkleinert werden, um damit den *NPS* einigermaßen effizient zu berechnen. Daher wird nur eine Menge aus 30 RGB-Werten berücksichtigt. Außerdem haben Drucker im Allgemeinen keine exakte Farbrepräsentation, der RGB-Wert, der gedruckt werden soll ist also nicht zwingend der, der am Ende auch gedruckt wird. Daher haben Sharif et al. eine Zuordnung zwischen den „originalen" RGB-Werten und den gedruckten erstellt, damit die Manipulation am Ende so gedruckt wird, wie sie auch aussehen soll.

5. Angriff auf ein White Box DNN

Dieses Experiment wird in zwei Teilexperimente aufgeteilt. Im ersten Teil wird der Angriff digital durchgeführt (s. Kapitel 5.1). Der Angreifer kann also den Input auf der Pixelebene manipulieren. Im zweiten Teil wird der Input durch eine Kamera aufgenommen und ohne weitere Bearbeitung direkt in das DNN gegeben (s. Kapitel 5.2). Es ist damit nicht mehr möglich, einzelne Pixel präzise zu manipulieren.

5.1 Digitaler Angriff

Hier wird ein DNN mit digital manipuliertem Input angegriffen. Es wird angenommen, dass ein Angreifer, der auf diese Weise das DNN nicht irreführen kann, auch Schwierigkeiten bei einem physischen Angriff (s. Kapitel 5.2) haben wird. Daher sind die Probleme und Schwierigkeiten auch hier von Interesse. Zudem dient der Angriff als Veranschaulichung der benötigten Manipulation, um DNNs Face Recognition Systeme irrezuführen, da diese teilweise so minimal sind, dass das menschliche Auge sie nicht mehr unterscheiden kann.

5.1.1 Beschreibung des Experiments

Es wurden insgesamt acht Dodging- und Impersonation-Angriffe durchgeführt. Dafür wurden die DNNs DNN_A, DNN_B und DNN_C verwendet. Bei den Experimenten wurde unterschieden zwischen:

1. dem Ziel des Angreifers (Impersonation oder Dodging)
2. welche Bereiche des Gesichts manipuliert werden dürfen (das ganze Gesicht oder nur das Brillengestell, s. Abbildung 1)
3. welches DNN angegriffen wird

Der Erfolg des Angreifers wird als *Success Rate SR* bezeichnet und beschreibt, wie oft der Angriff gelingt. Die Angabe erfolgt in Prozent.
Für Dodging ist das Ziel, als eine andere Person klassifiziert zu werden. Der Output eines DNNs ist eine Wahrscheinlichkeit für eine Klasse bzw. Person. Das bedeutet, der Angreifer muss die Wahrscheinlichkeit für eine beliebige andere Klasse erhöhen. Bei einem Impersonation-Angriff muss entsprechend die Wahrscheinlichkeit für eine spezielle Zielklasse am größten sein. Wenn ein Schwellwert T angegeben wurde, muss dieser entsprechend überschritten werden. In diesem Experiment wurden die Zielklassen zufällig aus den Klassen, welche das DNN erkennt, gewählt. Für die Angriffe auf DNN_A und DNN_C wurden aus den 2622 bzw. 143 Personen jeweils 20 verschiedene Personen zufällig ausgewählt. Für DNN_B wurden alle zehn ihm bekannten Personenklassen ausgewählt.
Damit ausgeschlossen werden kann, dass der Angriff nur bildspezifisch gelingt, werden von jedem Subjekt, das ein DNN angreift, drei verschiedene Bilder verwendet. Person X als Beispiel versucht also drei mal, mit je verschiedenen Bildern von sich, einen Dodging-Angriff. Daraus wird eine Statistik des Erfolgs berechnet, die sich auch in der *Tabelle 1* unter dem Punkt *SR* wiederfindet.
Die Iteration des Gradientenverfahrens wurde dabei auf 300 Iterationen beschränkt.

5.1.2 Ergebnis des Experiments

In folgender Tabelle sind die Ergebnisse der acht Teil-Experimente und deren Ergebnisse zusammengefasst.

Experiment#	*Area perturbed*	*Goal*	*Model*	*# Attackers*	*Success Rate*
1	Entire face	Dodging	DNN_A	20	100.00%
2	Entire face	Impersonation	DNN_A	20	100.00%
3	Eyeglass frames	Dodging	DNN_A	20	100.00%
4	Eyeglass frames	Dodging	DNN_B	10	100.00%
5	Eyeglass frames	Dodging	DNN_C	20	100.00%
6	Eyeglass frames	Impersonation	DNN_A	20	91.67%
7	Eyeglass frames	Impersonation	DNN_B	10	100.00%
8	Eyeglass frames	Impersonation	DNN_C	10	100.00%

Tabelle 1: Ergebnisse der Angriffe mit Manipulation auf Pixelebene. Jeder Angriff bzw. Teilexperiment wurden mit drei Bilder durchgeführt. Die Erfolgsrate *Success Rate* ist die durchschnittliche Wahrscheinlichkeit über drei Versuche.

In Experiment 1 und 2 durfte jedes Pixel des Gesichts manipuliert werden. Sowohl bei Dodging als auch Impersonation war der Angreifer immer erfolgreich. Auch waren die Manipulationen kaum wahrnehmbar für das menschliche Auge (s. Abb. 5). Abbildung 5 ist ein Beispiel aus Experiment 1. Die Angriffe sind laut Sharif et al. allerdings nicht besonders praxisrelevant, da diese Unterschiede zu subtil sind, um sie bei einem physischen Angriff nachstellen zu können. Auch das Einfangen der Manipulation mit der Kamera kann z.B. aufgrund der Auflösung problematisch sein, selbst wenn die Manipulation an sich perfekt gelungen ist. Daher ist es nicht besonders sinnvoll, diese Art von Angriff auch auf die Realität übertragen zu wollen. Dieses Problem lösen Experiment 3–8. Sie beschränken sich auf eine Manipulation mit einem Brillengestell. Abbildung 6 zeigt einen erfolgreichen Impersonation-Angriff mit Hilfe des manipulierten Brillengestells.

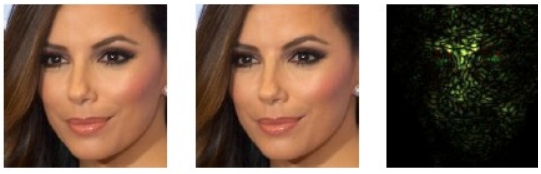

Abbildung 5: Beispiel eines erfolgreichen Angriffs mit Manipulation des ganzen Gesichts. Links ist ein Ausschnitt des Original-Bilds (Eva Longoria, von Richard Sandoval / CC BY-SA / https://goo.gl/7QUvRq)., in der Mitte das manipulierte Bild, rechts die Manipulation des mittleren Bildes.

Abbildung 6: Beispiel eines erfolgreichen Angriffs mit Manipulation eines Brillengestell. Links ein Ausschnitt des Originalbilds, Reese Witherspoon (von Eva Rinaldi / CC BY-SA / https://goo.gl/a2sCdc), in der Mitte das manipulierte Bild, rechts das Target, ein Ausschnitt aus eines Bildes von Russel Crowe (von Eva Rinaldi /CC BY-SA / https://goo.gl/AO7QYu).

Mit Ausnahme von Experiment 6 waren alle Dodging- und Impersonation-Versuche erfolgreich. Sharif et al. vermuten, es liegt an der hohen Anzahl der Klassen in DNN_A. Man versucht zwar, die Differenz von manipuliertem Input und Ziel so gering wie möglich zu halten, es ist jedoch möglich, dass die Differenz einer anderen Klasse zu dem manipulierten Input noch geringer war und deswegen eine andere Klassifizierung als die gewünschte auftrat.

5.2 Physischer Angriff

Physischer Angriff ist hier zu verstehen als ein Angriff auf ein System, bei dem der Angreifer den Input nicht mehr bearbeiten kann, sondern der so im DNN ausgewertet wird, wie er aufgenommen wurde. D.h. die Manipulation muss vor der Aufnahme des Inputs stattfinden. Dafür wird ein manipuliertes Brillengestell verwendet, welche einigen Eigenschaften genügen muss (s. Kapitel 4.1). Die Manipulation muss für eine Menge aus Inputs funktionieren (s. Kapitel 4.1.2), gleichzeitig *smooth* sein (s. Kapitel 4.1.3). Die Farben, die für den Angriff benötigt werden, sollten zudem im Gamut des Druckers liegen und somit druckbar sein (s. Kapitel 4.1.4). Das Optimierungsproblem, welches es für den Angreifer zu lösen gilt, ist nun folgendes:

$$argmin_r((\sum_{x \in X} softmaxloss(x+r, c_t)) + k_1 \cdot TV(r) + k_2 \cdot NPS(r))$$

Dies gilt sowohl für Dodging als auch Impersonation. k_1 und k_2 sind jeweils Konstanten, um das Ziel ins Verhältnis zu rücken, X ist eine Menge aus Bildern des Angreifers. In diesem Experiment werden nur DNN_B und DNN_C angegriffen. Da das DNN den Angreifer erkennen soll, fällt DNN_A weg.

5.2.1 Beschreibung des Experiments

Die ersten drei Autoren, mit denen DNN_B und DNN_C trainiert wurden, werden in dem Experiment als S_A, S_B und S_C bezeichnet. Jeder der Autoren führt dabei je einen Dodging- und einen Impersonation-Angriff auf sowohl DNN_B als auch DNN_C durch. Die Zielklasse für einen Impersonation-Angriff wird dabei wieder zufällig gewählt.
Die Aufnahmen, die für die verschiedenen Angriffe genutzt werden, wurden mit einer Canon T4i unter semi-kontrollierten Bedingungen aufgenommen. Das bedeutet hier, dass die Aufnahmen in einem Raum ohne Fenster nach außen gemacht wurden, um die Lichtverhältnisse in jedem Bild möglichst gleich zu halten. Außerdem war die Entfernung zur Kamera immer gleich und die einzelnen Autoren wurden mit neutralem Gesichtsausdruck aufgenommen. Zwischen den einzelnen Aufnahmen sollten sie ihre Pose leicht verändern, um sicherzustellen, dass die Manipulation nicht nur bildspezifisch oder nur mit sehr ähnlichen Bildern funktioniert. Trotz dieser Restriktionen wird das Experiment von Sharif et al. als realistisch eingestuft. Man stelle sich z.B. eine Zugangskontrolle in einem Gebäude vor. Die Lichtverhältnisse würden beispielsweise also wenig durch Tageslicht beeinflusst werden.
Für jeden der drei Autoren wurden jeweils 30–50 Bilder in jeweils fünf Sessions aufgenommen. In Session 1 wurden die Autoren jeweils ohne die manipulierte Brille aufgenommen, in Session 2 und 3 mit Brille für einen Dodging-Angriff gegen DNN_B bzw. DNN_C und in Session 4 und 5 mit Brille für einen Impersonation-Angriff gegen DNN_B bzw. DNN_C.
Der Druck der Vorderseite des Brillengestells wurde mit einem Epson XP-830 Drucker auf Foto-Papier realisiert. Damit belaufen sich die Kosten für einen Angriff auf gerade einmal 0,22$. Der Druck wurde anschließend zurechtgeschnitten und auf ein reales Brillengestell geklebt. Die Parameter von k_1 und k_2 auf 0,15 bzw. 0,25 gesetzt, da diese Werte effektiv genug für einen Angriff sind. Die Anzahl der Iterationen des Gradient Descent Algorithmus wurde auf 300 begrenzt.

5.2.2 Ergebnis des Experiments

In folgender Tabelle sind die Ergebnisse der acht Teil-Experimente und deren Ergebnisse zusammengefasst.

	Subject (attacker) info		Dodging results		Impersonation results			
DNN	*Subject*	*Identity*	*SR*	*E(p(*correct-class*))*	*Ziel*	*SR*	*SRT*	*E(p(*target*))*
DNN_B	S_A	3rd author	100.00%	0.01	Milla Jovovich	87.87%	48.48%	0.78
	S_B	2nd author	100.00%	0.03	S_C	88.00%	75.00%	0.75
	S_C	1st auhtor	100.00%	0.35	Clive Owen	16.13%	0.00%	0.33
DNN_C	S_A	3rd author	100.00%	0.03	John Malkovich	100.00%	100.00%	0.99
	S_B	2nd author	100.00%	<0.01	Colin Powell	16.22%	0.00%	0.08
	S_C	1st auhtor	100.00%	<0.01	Carson Daly	100.00%	100.00%	0.90

Tabelle 2: Zusammenfassung der Ergebnisse der physisch realisierten Angriffe. Links stehen die angegriffenen DNNs sowie die Angreifer. *SR* bezeichnet die Erfolgswahrscheinlichkeit, SR*T* die Erfolgswahrscheinlichkeit mit einem Schwellwert *T*. Für DNN_B liegt *T* bei 0,85, für DNN_C bei 0,90. *E(p(*correct-class*))* bezeichnet die mittlere Wahrscheinlichkeit, als Angreifer identifiziert zu werden, *E(p(*target*))* bezeichnet die mittlere Wahrscheinlichkeit, als Zielklasse *target* identifiziert zu werden.

Um DNN_B und DNN_C zu testen, wurden die Bilder aus Session 1 verwendet. Die Wahrscheinlichkeit, dass das DNN eine Person richtig klassifiziert, liegt bei >85%. Es ist also unwahrscheinlich, dass jemand, der nicht versucht, das System anzugreifen, aus Versehen missklassifiziert wird. Das Experiment zeigt aber auch, dass ein Angreifer, der aktiv versucht, das System zu manipulieren, mit hoher Wahrscheinlichkeit erfolgreich ist.

Dodging Alle Angriffe wurden mit der manipulierten Brille durchgeführt. Die Angriffe von S_A mit den Bildern aus Foto-Session 2 und 3 waren alle erfolgreich. Das beschreibt die Spalte *SR* in Tabelle 2. Die Spalte *E(p(*correct-class*))* beschreibt die Wahrscheinlichkeit, dass der Angreifer korrekt identifiziert wird, also S_A selbst bei einem Angriff als S_A klassifiziert wird. Da dieser Wert bei DNN_B von 1,0 auf 0,01 für DNN_B sank bzw. von 0,85 auf 0,03 für DNN_C, ist dieser Fall jedoch sehr unwahrscheinlich. Wie jedoch der Wert zu *E(p(*correct-class*))* ermittelt wurde, wird nicht näher erläutert.

Abbildung 7: Die von S_C benötigte Brille für einen erfolgreichen Angriff gegen DNN_B

Die Angriffe von S_B waren zu 97,22% gegen DNN_B und sogar zu 100% gegen DNN_C erfolgreich. Die Wahrscheinlichkeit p einer Identifizierung als Angreifer liegt dabei bei $p \leq 0.03$ für DNN_B und $p < 0.01$ für DNN_C. Auch hier ist eine korrekte Identifizierung als Angreifer unwahrscheinlich.

Die Angriffe mit den Aufnahmen aus Session 3 von S_C waren für DNN_C zu 100% erfolgreich, und die Wahrscheinlichkeit als Angreifer klassifiziert zu werden, lag bei $p < 0.01$. Jedoch war es S_C nicht möglich, mit den Aufnahmen aus Session 2 auch nur einen einzigen Angriff gegen DNN_B erfolgreich durchzuführen. Der Grund laut Sharif et al. ist vermutlich, dass S_C bei Session 1 auch eine Brille trug. Das bedeutet, Zielklassen, bei denen die Zielperson eine Brille trägt, werden als Ergebnis wahrscheinlicher. Da DNN_B nur ein 10-Klassen-DNN ist, ist die Wahrscheinlichkeit für eine Person mit Brille hoch, und damit auch die Identifizierung als Angreifer. Um trotzdem einen Angriff durchzuführen, wurde eine größere Brille verwendet (s. Abb. 7). Diese bedeckt ca. 10% des Bildes. Somit konnte S_C trotzdem Dodging-Angriffe zu 80% erfolgreich durchführen, die Identifizierung von S_C als S_C liegt jedoch immer noch bei ca. 0,35. Damit wäre ein Angriff noch einigermaßen effizient, diese Brille ist jedoch weit weniger unauffällig als die „normale".

Impersonation S_A, S_B und S_C greifen mit einem Impersonation-Angriff je einmal DNN_B und DNN_C an. Dabei wird die Zielklasse, als die der Angreifer vom DNN identifiziert werden muss, zufällig ausgesucht.
S_A ist 41 Jahre alt, weiß und männlich. Seine Zielklassen sind

1. Milla Jovovich, 40 Jahre alt, weiß, weiblich
2. John Malkovich, 62 Jahre alt, weiß, männlich

S_B ist 24 Jahre alt, asiatisches Aussehen, weiß. Ihre Zielklassen sind

1. S_C, 24 Jahre alt, orientalisches Aussehen, männlich
2. Colin Powell, 79 Jahre alt, weiß, männlich

S_C ist ebenfalls 24 Jahre alt, orientalisches Aussehen und männlich. Seine Zielklassen sind

1. Clive Owen, 51 Jahre alt, weiß, männlich
2. Carson Daly, 43 Jahre alt, weiß, männlich

S_A's Angriffe auf DNN_B als Milla Jovovich waren zu 87,87% erfolgreich, die Angriffe auf DNN_C sogar zu 100%. Die mittlere Wahrscheinlichkeit für DNN_B lag bei 78%, für DNN_C bei 99%.

S_B's Angriffe auf DNN_B waren zu 88,00% erfolgreich, die Angriffe auf DNN_C waren allerdings nicht besonders gut. Hier wurde nur eine Erfolgsrate von 16,22% erreicht. Die mittlere Wahrscheinlichkeit für DNN_B lag bei 33%, für DNN_C bei 90%.

S_C's Angriffe auf DNN_B als Clive Owen waren mit 16,13% auch nicht besonders erfolgreich, Angriffe auf DNN_C als Carson Daly hatte jedoch eine Erfolgswahrscheinlichkeit von 100%. S_C benötigte wieder das größere Brillengestell (s. Abb. 7), um das DNN irrezuführen. Die mittlere Wahrscheinlichkeit für DNN_B lag bei 75%, für DNN_C sogar nur bei 8%.

S_B's und S_C's Angriff als S_c bzw. Colin Powell sind nur in einem von sechs Versuchen erfolgreich. Das könnte also zu einem Problem werden, wenn das DNN die Versuche limitiert wie z.B. bei einer PIN. Das Problem ist vermutlich dasselbe wie in Experiment 6 von Kapitel 5.1. In der Differenz zwischen manipuliertem Input und Ziel befindet sich eine weitere Klasse, die die gewünschte Missklassifizierung verhindert. Sharif et al. vermuten, dass weitere Anpassungen dieses Ergebnis noch weiter verbessern könnten.

Abbildung 8: Beispiele für erfolgreiche Dodging-Angriffe (a) bzw. Impersonation (b, c und d). Spalte b, c, und d zeigen oben den Angreifer und unten das *target* für Impersonation-Angriffe.
Spalte a - Dodging-Angriffe von S_A (oben) und S_B (unten). :
Spalte b – Angriff von S_A als Milla Jovovich (by Georges Biard / CC BY-SA / https://goo.gl/GlsWlC)
Spalte c – Angriff von S_B als S_C
Spalte d – Angriff von S_C als Carson Daly (von Anthony Quintano / CC BY / https://goo.gl/VfnDct)

Um die Sicherheit von Face Recognition Systemen weiter zu erhöhen, wird ein Schwellwert verwendet, der erst überschritten werden muss. Es reicht für einen erfolgreichen Angriff also nicht, die höchste Wahrscheinlichkeit für die Zielklasse zu haben, wenn sie den Schwellwert nicht überschreitet. Daher ist hier die Erfolgswahrscheinlichkeit höchstens gleich oder niedriger als ohne Schwellwert. Sie bleibt jedoch hoch genug, um einen solchen Angriff als realistisch einzustufen. Die Erfolgswahrscheinlichkeit mit dem Schwellwert wird mit *SRT* bezeichnet. In *Tabelle 2* kann man erkennen, dass z.B. bei dem ersten Impersonation-Angriff von S_A auf DNN_B der Wert der Erfolgsrate von 87,87% ohne Schwellwert auf 48,48% mit Schwellwert sinkt. Wenn der Schwellwert besonders hoch ist, ist die Akzeptanz eines Angriffs sehr gering. Die Akzeptanz von einer legitimen Nutzung des DNNs ist jedoch auch sehr gering und beeinträchtigt so die Usabilty. Ein kleiner Schwellwert hingegen macht es Angreifern besonders leicht. Die Sicherheit des Face Recognition Systems ist dadurch sehr gering, dafür werden legitime Nutzungen nicht zurückgewiesen.

Für diese Experimente wurde der Schwellwert für DNN_B auf 0.85 gesetzt. Eine „falsche" Akzeptanz (falsch positiv) bei DNN_B sinkt damit auf 0, während die „richtige" Akzeptanz (richtig positiv) bei 92,31% liegt. Dies ist eine gute Balance zwischen Usability und Sicherheit. Für DNN_C liegt diese Schwelle bei 0.90. Damit ist die Rate von falsch-positiv mit 0.004 knapp über 0 und die richtig-positiv-Rate bei 92,01%. Bei einer Rate bei falsch-positiv von 0 würde die gerechtfertigte Akzeptanz eines Inputs auf 41,42% sinken und stark die Usability beeinträchtigen.

Komplexitätsklasse

Die hier verwendeten DNNs sind relativ groß. DNN_B hat alleine schon ca. 3,86 Millionen Verbindungen N_C. Wenn man die Manipulation für N_I Bilder berechnet, liegt die Komplexität in $O(N_I \cdot N_C)$ pro Iteration im Gradienten-Verfahren. Bei 30 Bildern dauert eine Iteration auf einem MacBook Pro mit 16GB Arbeitsspeicher und Intel i7 Prozessor 52,72 Sekunden. Die Iterationen wurden auf 300 begrenzt, daher wurde das Optimierungsproblem in $300 \cdot 52{,}72$ Sekunden gelöst, umgerechnet ca. 4,39 Stunden.

6. Angriff auf ein Black Box DNN

Bei einem Black Box DNN, sind im Gegensatz zum White Box DNN, keine innere Struktur, Architektur, Algorithmen o.ä. bekannt. Trotzdem können die Angriffe in ähnlicher Weise durchgeführt werden. Die Black Box, also das anzugreifende System, wird als *Oracle O* bezeichnet. Dafür wird Face++[6] genutzt, ein State-of-the-Art-Face Recognition System. Es bietet jedoch auch Face Detection, Tracking etc. an. Bei Face++ kann der User Trainingsdaten hochladen, anhand derer ein DNN trainiert wird. Über das DNN selbst wird keine Angabe gemacht, jedoch wird über 97% Genauigkeit bei der LFW-Challenge mit Face++ erzielt[2].
Gibt man dem fertig trainierten Face++ ein Bild, besteht der Output aus den drei wahrscheinlichsten Klassen. Die erste Klasse ist dabei das beste Ergebnis mit der höchsten Wahrscheinlichkeit. Jedes Ergebnis dieser Output-Liste hat einen sogenannten *Confidence Score*. Dazu hat der User jedoch auch keine explizite Erklärung, ist also auch Teil der Black Box.
Das Face++-DNN wurde mit den gleichen Trainingsdaten trainiert wie DNN_B, also auch ein 10-Klassen-FRS.

6.1 Impersonation-Angriff

Da die Dodging-Angriffe selbst mit zufällig eingefärbten Brillen sofort für einige verschiedene Inputs funktionierten, fokussieren sich Sharif et al. bei diesem Experiment auf Impersonation-Angriffe. Face++ ist dabei das Black Box FRS, welches als das Oracle *O* bezeichnet wird. *O(x)* gibt als Ergebnis eine Liste der drei Top-Kandidaten mit den höchsten Wahrscheinlichkeiten zurück. Der Angriff gegen Face++ nutzt die sogenannte Partikelschwarm-Optimierung *PSO*.

Partikelschwarm-Optimierung Die Partikelschwarmoptimierung ist ein heuristisches Lösungsverfahren für Optimierungsprobleme und ahmt das natürliche Schwarmverhalten von z.B. Vögeln oder Fischen nach [1]. Jedes Objekt des Schwarms wird in dem Algorithmus als Partikel bezeichnet und ist ein Kandidat zur Lösung des Optimierungsproblems. Jedes Partikel hat dabei eine Position p und eine Richtung (*Velocity)* $\vec{v}$. Außerdem merkt sich jedes Partikel den besten Wert p_{best} , den es bisher gefunden hat und einen globalen besten Wert g_{best} , der für alle Partikel in dem Schwarm gleich ist. Bei jeder Iteration über ein Partikel wird die Position mit dem aktualisierten Richtungsvektor $\vec{v}$ angepasst. $\vec{v}$ wird berechnet aus zwei Vektoren:

1. aus der aktuellen Position des Partikels als Ortsvektor $\vec{p}$ und dem Ortsvektor zum Bestwert des Partikels $\overrightarrow{p_{best}}$, also $\overrightarrow{p_{best}} - \vec{p}$
2. aus der aktuellen Position des Partikels als Ortsvektor $\vec{p}$ und dem Ortsvektor zum Bestwert des Schwarms $\overrightarrow{g_{best}}$, also $\overrightarrow{g_{best}} - \vec{p}$

So nähert sich jedes Partikel, also insgesamt der Schwarm, sukzessive dem Extremum an.

Dieser Algorithmus ist vergleichsweise einfach, da er weder die Trainingsdaten noch besondere Rechenleistung benötigt. Deswegen ist er geeignet für das Problem. Er bringt jedoch einen Nachteil mit sich. Wenn sich die Zielklasse des Impersonation-Angriffs nicht in der Liste der Top 3, also dem Ergebnis des DNNs, befindet, erhält der Algorithmus keine Rückmeldung, um eine Aktualisierung bzw. Optimierung der Position der einzelnen Partikel vorzunehmen. Daher haben Sharif et al. einen Algorithmus entwickelt, der sich *Recursive Impersonation* (s. Algorithmus 1) nennt.

Recursive Impersonation

Hierbei werden Impersonation-Zwischenziele gewählt und damit ein Angriff durchgeführt. Dabei wird der Schwarm im Suchraum bewegt, wenn er keine Rückmeldung über den Misserfolg eines Angriffs bekommt. Dies geschieht in der Hoffnung, den nicht ertragreichen Suchraum zu verlassen und möglicherweise näher am Optimum zu suchen. In der Implementierung *Recursive Impersonation* von Sharif et al. werden zudem alle Partikel der letzten Iteration der PSO-Routine gespeichert sowie alle weiteren Partikel, die während der bereits durchlaufenen Iterationen eine Manipulation gefunden haben, für die die Zielklasse im Output erschienen ist. Diese Daten werden verwendet, um neue Zwischenziele zu wählen oder die „*Seed*", eine Menge aus Partikeln, welche in die PSO-Routine gegeben wird, neu zu setzen.

Algorithmus 1: Recursive Impersonation

```
1  Initialize epoch = 0, numParticle, epoch_max and seed
2  Set candidates = O(image_original)
3  if target ∈ cancdidates then target_current = target;
4  else target_current = 2nd most probable class of candidates;
5  while epoch ≤ epoch_max do
6  |  Run PSO subroutine with target_current and seed
7  |  if any particle impersonated target during PSO then
8  |  |  solution was found. Exit.
9  |  else if target ∈ cancdidates of any query during PSO then
10 |  |  target_current = target. Clear seed.
11 |  |  seed ⊇ particles that produced this candidate from the
   |  |  current PSO run.
12 |  else
13 |  |  if new candidate emerges from current PSO run then
14 |  |  |  target_current = new candidate. Clear seed.
15 |  |  |  seed ⊇ particles that produced this candidate from
   |  |  |  thecurrent PSO run.
16 |  |  else
17 |  |  |  no solution was found. Exit.
18 |  |  end
19 |  end
20 |  epoch = epoch + 1;
21 end
```

Der *Recursive Impersonation-Algorithmus* verfährt wie folgt: zunächst werden ein Zähler *epoch* und eine Menge von Partikeln als s*eed* initialisiert sowie eine Begrenzung der Durchläufe festgelegt. Außerdem wird eine Menge aus Kandidaten *candidates* mit dem Output des *Oracles* initialisiert. Dabei wird unterschieden, ob das Ziel schon in der Kandidaten-Menge vorhanden ist. Wenn das der Fall ist, wird das aktuelle Zwischenziel auf das eigentliche Ziel *target* gesetzt, sonst wird das zweit wahrscheinlichste Ergebnis als Zwischenziel verwendet. Anschließend startet die Hauptschleife des Algorithmus, wo zunächst die PSO-Routine gestartet wird mit den Parametern *target* und *seed*. Mit der PSO-Routine wird eine Manipulation auf einen Input angewendet und ein Angriff gestartet. Je nach Ergebnis, welches das DNN zurückliefert, wird die Manipulation entsprechend angepasst. Innerhalb der Schleife werden nun Fallunterscheidungen vorgenommen, abhängig davon, ob ein Impersonation-Angriff erfolgreich war, also *target* auf Platz 1 des Outputs, ob das *target* an einer anderen Position im Output vorkam oder ob ein neues Zwischenziel als Impersonation-Target gewählt werden muss. Anhand dieser Unterscheidungen werden die *seed* und das *target* neu gesetzt und die PSO-Routine damit erneut gestartet. Dieser Prozess läuft so lange, bis die maximale Anzahl an Wiederholungen erreicht oder der Angriff erfolgreich war.

Der PSO-Algorithmus versucht, die Funktion $f(x+r)$ zu minimieren. x und r sind wie zuvor auch der Input bzw. die Manipulation, $f(...)$ wird anhand des Outputs des *Oracles O* berechnet. $...$ ist wieder ein Platzhalter. Dieser Wert wird dann verwendet, um den gesamten Schwarm bei der nächsten Iteration zu bewegen. Für $f(...)$ wird folgende Funktion verwendet, da sie laut Sharif et al. die effektivste Zielfunktion ist. Dabei werden zwei Fälle unterschieden:
Wenn $target \in cancidates$, also das Impersonationziel *target* in der Liste des Outputs *candidates* vorkommt, ist die Zielfunktion

$$f(x) = rank \cdot \frac{score_{top}}{score_{target}} \quad .$$

Wenn das nicht der Fall ist, ist $f(x) = maxObjective$. $score_{top}$ ist dabei der *Confidence Score* des besten Ergebnisses bzw. der höchsten Wahrscheinlichkeit, $score_{target}$ ist der *Confidence Score* des Impersonationziels *target* und *rank* der Platz auf der Liste, also 1, wenn der Angriff funktioniert hat. Dann sind auch $score_{top}$ und $score_{target}$ gleich, die gesamte Funktion wertet dann zu 1 aus. Wenn $score_{target}$ nicht in der Top 3-Liste vorkommt, wird das Ergebnis auf *maxObjective* gesetzt. *maxObjective* ist dabei weit von p_{best} und g_{best} entfernt.

6.2 Beschreibung des Experiments

Insgesamt wurden vier Paare aus je einem Ziel und einem Angreifer aus der Menge der Klassen aus Face++ gewählt. Zwei Paare davon waren zufällig, die anderen beiden wurden so gewählt, dass sich das Ziel nicht in der Output-Liste des DNNs befindet. Auch hier wurde die Manipulation insofern eingeschränkt, als wieder nur eine Brille als Manipulation genutzt werden darf. Das soll einen physischen Angriff simulieren, da ein solcher nicht konkret ausgeführt wurde.
Für den Angriff wurde, wie schon erwähnt, der PSO-Algorithmus verwendet. Konkret wurden 25 Partikel benutzt. Jedes einzelne Partikel wurde mit einer sich von den anderen Partikeln unterscheidenden Manipulation mit weichen Farbverläufen initialisiert.
Der Algorithmus wurde auf ein Maximum von 15 Versuchen beschränkt, die maximale Anzahl an Iterationen des PSO-Algorithmus sowie *maxObjective* wurde auf 50 gesetzt. Außerdem wird das globale Optimum g_{best} etwas stärker gewertet als $\overrightarrow{p_{best}}$.

6.3 Ergebnis des Experiments

Die Ergebnisse des Experiments werden in Tabelle *Tabelle 3* gezeigt.

Source	*Target*	*Success Rate*	*Avg. # queries*
Cliwe Owen	S_A	100.00%	109
Drew Barrymore	S_B	100.00%	134
S_D	S_C	100.00%	25
S_D	Cliwe Owen	100.00%	59

Tabelle 3: Ergebnis der Angriffe, jeder davon wurde drei mal durchgeführt. S_A, S_B und S_C sind die Autoren, S_D ist 33 Jahre alt, mit asiatischem Aussehen, weiblich. Jeder Versuch wurde mit einer zufälligen anderen Seed und Velocity ausgeführt. *Avg.# queries* ist die Anzahl der Abfragen, die der *Recursive Impersonation*-Algorithmus für einen erfolgreichen Angriff benötigt

Die Tabelle zeigt, dass alle Angriffe erfolgreich waren. Dabei ist die relativ geringe Anzahl der Abfragen an das *Oracle* hervorhebenswert, da dies zeigt, dass man auch Systeme erfolgreich angreifen kann, die nach einer gewissen Anzahl Anfragen neue Anfragen blocken. Es wurden also zwischen 25 und 134 Versuche benötigt, bis die Zielklasse Platz 1 in der Kandidatenliste belegte. Face++ limitiert z.B. Abfragen auf 50.000 pro Monat pro User.

7. Angriff auf ein Face Detection System

Bei diesem Experiment wird gezeigt, wie man „Unsichtbarkeit" gegen Face-Detection erreicht. Das ist insofern interessant, als das Erkennen eines Gesichts der erste Schritt zu Face Recognition ist. Scheitert also das Face Detection-System am Erkennen eines Gesichts, scheitert auch Face Recognition. Da Face Detection allgemeiner gefasst ist und wenige Trainingsdaten benötigt, kann ein Angriff auf verschiedene Systeme auf gleiche oder zumindest sehr ähnliche Weise durchgeführt werden.
Ein zweiter Grund ist der Erhalt der Privatsphäre. Möchte eine Person nicht erkannt werden, ist es einfacher, nicht von einem Gesichtserkennungssystem erkannt zu werden, als es z.B. mit Dodging anzugreifen. Außerdem ist ein Nicht-Erkennen eines Gesichts weniger auffällig als ein Angriff. Sind z.B. *FRS* an einem Flughafen darauf ausgerichtet, spezielle Gesichter zu erkennen, z.B. von Behörden gesuchte Personen, ist ein Angriff möglicherweise besonders auffällig, wenn er schief geht. Das Gesicht der gesuchten Person nicht erkannt zu haben, könnte man als einen Fehler des Gesichtserkennungssystems interpretieren.

7.1 Viola-Jones Face Detector

Der Viola-Jones Face Detection Algorithmus gilt als besonders schnell und präzise. Das liegt an der Idee, Klassifikatoren in aufsteigender Komplexität zu verketten. Um ein ganzes Bild auf Gesichter zu untersuchen, wird das Bild in mehrere Unterbilder unterteilt. Zuerst werden die einfachen Klassifikatoren der Kette ausgewertet. Sobald einer der Klassifikatoren den Input abweist, wird die Auswertung beendet. Bei dem Teilbild handelt es sich dann also nicht um ein Gesicht. Ansonsten wird die Kette an Klassifikatoren weiter abgearbeitet. Sind alle abgearbeitet und keiner in der Kette hat das Teilbild verworfen, handelt es sich um ein Gesicht.

Ein Klassifikator besteht aus einer Komposition von sogenannten *weak classifiers*, also einfachen Klassifikatoren. Bei aufsteigender Komplexität des Klassifikators werden mehr und mehr *weak classifiers* verwendet. Ein *weak classifier i* hat dabei nur zwei Mögliche Ausgänge: $\widetilde{a_i}$ oder $\hat{a}_i$, also akzeptieren oder verwerfen.

Das basiert auf einem Merkmal, der Funktion f_i und einer Schwelle b_i. Die Entscheidung, ob der Input vom Klassifikator akzeptiert oder abgewiesen wird, wird durch folgende Formel modelliert.

$$Classify(x) = (\sum_{i=1}^{C} ((\tilde{a}_i - \hat{a}_i)(f_i(x) > b_i) + \hat{a}_i)) > T$$

C ist dabei die Anzahl der *weak classifier* , T ist der Schwellwert, der überschritten werden muss, x ist das Teilbild des Inputs und $f_i(x) > b_i$ wird zu 1, wenn die Bedingung erfüllt ist. Um ein Face Detection System irrezuführen, reicht es also, einen Klassifikator dazu zu bringen, ein Teilbild abzuweisen, da dann das gesamte Teilbild verworfen wird. Schafft man es, einen beliebigen Klassifikator in der Kette irrezuführen, bleibt man für das Face Detection System unsichtbar.
Um das Optimierungsproblem zu lösen, muss die Funktion differenzierbar sein. Da die *Classify-Funktion* nicht differenzierbar ist, wird die Sigmoid-Funktion verwendet, welche einfach abzuleiten ist.
Mit Hilfe der Sigmoid-Funktion *sig* ist das Optimierungsproblem lösbar.

$$argmin_r((\sum_{i=i}^{C} ((\tilde{a}_i - \hat{a}_i) \cdot sig(k \cdot (f_i(x+r)) - b_i) + a_i) - T) + c|r|)$$

k ist dabei eine positive reelle Zahl, um die Annäherung zu kontrollieren. Damit kann man den Gradient Descent Algorithmus verwenden.

7.2 Ergebnis des Experiments

Um diesen Ansatz zu testen, wurden 20 Bilder aus der Online Datenbank PubFig[4] verendet. Die Manipulation soll dabei den ersten Klassifizierer so beeinflussen, sodass dieser das Teilbild abweist. Der manipulierte Input wurde wie folgt generiert: die Manipulation wurde auf das Gesicht beschränkt, c wurde auf 0,015 gesetzt. Mit dem Liniensuchverfahren (iterativer Algorithmus für ein Optimierungsproblem) wurde ein Wert für *k* für die minimalste Manipulation für einen erfolgreichen Angriff mit begrenzten Iterationen des Broyden–Fletcher–Goldfarb–Shanno -Algorithmus bzw. BFGS[7] gesucht. Auch BFGS ist ein iterativer Algorithmus zum Lösen eines Optimierungsproblems.

Abbildung 9: Beispiele eines erfolgreichen Angriffs auf ein Face Detection System.
Links -Originalbild von Kiefer Sutherland
Mitte – Angriff mit leicht transparenten Bereichen, die Teile des Gesichts überlagern
Rechts – Angriff mit Accessoires

19 von 20 Personen wurden dabei nicht vom Face Detection System erkannt. Jedoch war die benötigte Manipulation, damit der Angriff gelingt, relativ auffällig. Um die Abstreitbarkeit zu wahren, wurde auf eine digitale Manipulation in Form von Accessoires für das Gesicht zurückgegriffen. Es wurden also keine richtigen Accessoires verwendet, sondern digital auf das Gesicht gezeichnet. Die verwendeten Accessoires sind eine blonde Perücke, eine Mütze, eine Brille, Eyeblacks sowie Kontaktlinsen. Damit gelangen alle Angriffe und erhielten trotzdem die Abstreitbarkeit. In dem Versuch ohne Accessoires wurden die Pixel durch die Farbbalken im Schnitt um 16,06% verändert (Standardabweichung 6,35), durch die digital auf das Gesicht gezeichneten Accessoires hingegen um 3,01% mit Standardabweichung 2,12.

8. Fazit

In der Zukunft wird es sicherlich wichtig, zu verstehen, wie Angriffe aller Art gegen *FRS* durchgeführt werden können, um sie zu verhindern. Wird z.B. allgemein bekannt, dass man mit einer bunt gefärbten Brille möglicherweise ein System angreifen kann, wäre es z.B. naheliegend, bei Zugriffskontrollen nur Zugriff für Personen zu gewähren, die ihr Gesicht frei von Accessoires wie Brillen aufnehmen lassen. Für manche Zugriffskontrollen wäre es möglicherweise sinnvoll, wieder auf Menschen als Zugangskontrolle zurückzugreifen, da sie nicht anfällig sind gegenüber nicht sichtbaren Manipulationen. Es wäre aber auch möglich, die entsprechenden Algorithmen weiterzuentwickeln, damit sie der Fähigkeit des Menschen noch ähnlicher sind. Das würde bedeuten, dass die Erkennung von Gesichtern nicht anhand einzelner Pixel funktioniert, sondern anhand von Attributen.
Aus Sicht des Angreifers wäre es sinnvoll, einen Angriff auch unabhängig von den im Experiment angenommenen Restriktionen zu realisieren. Das bedeutet also auch außerhalb eines geschlossenen Gebäudes ohne Tageslicht einen Angriff durchführen zu können, z.B. auf der Straße.
Außerdem ist die Unauffälligkeit der Brille subjektiv. Wenn man diese Art von Angriffen nicht kennt, findet man die Optik der Brille möglicherweise einfach nicht ansprechend, jedoch nicht auffällig. Auf der anderen Seite ist eine solche Brille eventuell für Leute besonders auffällig, die wissen, dass es mit einem solchen Brillengestell möglich ist, einen Angriff zu realisieren. Solche Angriffe können also nur funktionieren, solange nicht allzu viele Leute über entsprechende Szenarien Bescheid wissen. Man würde stattdessen also eine Brille benötigen, die weniger bunt bzw. weniger auffällig ist, also z.B. mit Farben koloriert, die zueinander passen und optisch ansprechend sind.

Die durchgeführten Experimente zeigen, dass es durchaus möglich ist, kostengünstig einen einigermaßen unauffälligen, physisch realisierten Angriff auf ein Gesichtserkennungssystem durchzuführen, jedoch nur unten bestimmten Voraussetzungen. Ändern sich die Bedingungen, muss die Manipulation wieder neu berechnet werden. Es gibt also auch in dem Bereich aus Sicht des Angreifers noch Forschungspotenzial.

9. Verzeichnis

9.1 Abbildungsverzeichnis

9.2 Tabellenverzeichnis

9.3 Algorithmenverzeichnis

10. Quellen

[1] R. Eberhart and J. Kennedy
A new optimizer using particle swarm theory
In Proc. Micro Machine and Human Science (MHS), 1995

[2] H. Fan, Z. Cao, Y. Jiang, Q. Yin, and C. Doudou
Learning deep face representation
in arXiv: 1403.2802v1, 2014

[3] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller
Labeled faces in the wild: A database for studying face recognition in unconstrained environments, Technical Report 07-49
University of Massachusetts, Amherst, 2007

[4] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar
Attribute and simile classfiers for face verification
In Proc. International Conference on Computer Vision (ICCV), 2009

[5] A. Mahendran and A. Vedaldi
Understanding deep image representations by inverting them
In Proc. Conference on Computer Vision and Pattern Recognition (CVPR), 2015

[6] Megvii Inc. Face++
http://www.faceplusplus.com/

[7] J. Nocedal
Updating quasi-newton matrices with limited storage
Mathematics of computation,1980

[8] O. M. Parkhi, A. Vedaldi, and A. Zisserman
Deep face recognition
In Proc. British Machine Vision Conference (BMVC), 2015

[9] F. Schroff, D. Kalenichenko, and J. Philbin.
Facenet: A unified embedding for face recognition and clustering.
In Proc. Conference on Computer Vision and Pattern Recognition (CVPR), 2015

[10] M. Sharif, S. Bhagavatula, L. Bauer, M. K. Reiter
Accessorize to a Crime: Real and Stealthy Attacks on Face-of-the-Art Face recognition,
In Proc. Conference on Computer and Communication Security (CCM), 2016

[11] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus
Intriguing properties of neural networks
In Proc. International Conference on Learning Representation (ICLR), 2014

[12] P. Viola and M. Jones
Rapid object detection using a boosted cascade of simple features
In Proc. Conference on Computer Vision and Pattern Recognition (CVPR), 2001

[13] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson
How transferable are features in deep neural networks?
In Proc. Conference on Neural Information Processing Systems (NIPS), 2014