

Irrbilder für InformatiCup-Blackbox

Theoretische Ausarbeitung

Alexander Kretzschmar, Marvin Springer

TU Dresden / TU Braunschweig

Erstgutachter:	Prof. Dr.-Ing. Stefanie Speidel
Zweitgutachter:	Prof. Dr. rer. nat. Stefan Gumhold
Betreuender Mitarbeiter:	Micha Pfeiffer

Bearbeitungszeit: March 1, 2018 – June 27, 2018



Technische
Universität
Braunschweig



TECHNISCHE
UNIVERSITÄT
DRESDEN

Wir versichern hiermit, die vorliegende Arbeit selbstständig angefertigt zu haben. Die verwendeten Hilfsmittel und Quellen sind im Literaturverzeichnis vollständig aufgeführt.

Dresden, den 27. Juni, 2018

Zusammenfassung

Capsule Networks are a brand new form of neural network, designed to overcome the drawbacks of todays state-of-the-art Convolutional Neural Networks. Goal of this Bachelor Thesis was to analyze this new concept, develop a functional prototype for the detection of surgical tools in laparoscopic videos and evaluate the potential of this new type of neural network compared to traditional Convolutional Neural Networks.

Inhaltsverzeichnis

1	Einführung	1
1.1	Motivation	1
1.2	Ziele	1
2	Theoretischer Hintergrund	3
2.1	Anfälligkeit gegenüber Störbildern	3
2.2	Angriffsmöglichkeiten	3
2.2.1	FGSM	4
2.2.2	Iterative Methode	4
2.2.3	Methode zum Erreichen einer bestimmten Klasse	4
2.3	Angriffe gegen eine Blackbox	4
3	Methoden	5
3.1	Entscheidung über die zu verwendenden Methoden	5
3.2	Softwarearchitektur	5
3.3	Generierung der Störbilder	5
3.4	Das Ersatznetz	6
3.5	Testing	6
3.6	Wartbarkeit	6
4	Evaluation	7
5	Diskussion	9
5.1	Nachteile	9
5.2	Verbesserungsmöglichkeiten	9
5.3	Praktischer Einsatz	9
	List of Figures	11
	List of Algorithms	13
	Bibliography	15

1. Einführung

1.1 Motivation

Machine Learning Algorithmen, insbesondere Neuronale Netze, finden zur Zeit großflächig in der automatisierten Bildverarbeitung Verwendung. Doch auch wenn deren Ergebnisqualität bei verschiedensten Aufgaben, wie Objektklassifizierung, Gesichtserkennung u.A. stetig steigt, so ist seit einigen Jahren eine ihrer Schwachstellen zunehmend in den Fokus der Forschung geraten: Die Anfälligkeit gegenüber sogenannten „Adversaries“, gezielt generierten Störbildern, die Neuronale Netze mit hoher Zielsicherheit falsche Ergebnisse produzieren lassen. Besonders die Tatsache, dass sich die Methoden zur Generierung solcher Störbilder von einem Modell auf andere übertragen lassen, sorgt dabei für Kopfzerbrechen. Selbst ohne detailliertes Wissen über ein spezifisches Netz, wie dessen genaue Struktur, lassen sich zuverlässig Störbilder generieren.

Insbesondere die Verwendung Neuroner Netze im Rahmen von sicherheitskritischen Anwendungen macht deutlich, wie wichtig es ist, genau über dieses Phänomen und seine möglichen Gegenmaßnahmen Bescheid zu wissen. Im Rahmen des informatiCup 2019 haben wir uns deshalb mit einer dieser sicherheitskritischen Anwendungen befasst: Der automatisierten Erkennung und Klassifizierung von Verkehrsschildern im Kontext autonomen Fahrens. Die Folgen eines Fehlverhaltens in Folge von zielgerichtet generierten Störbildern in diesem Feld unterstreichen noch einmal deutlich die Wichtigkeit dieses Themas. Das Anbringen für das menschliche Auge nicht erkennbarer Störbilder in der Nähe von Fahrbahnen kann schwerste Unfälle hervorrufen.

1.2 Ziele

Das Ziel im Rahmen des informatiCup 2019 war es, Störbilder zu generieren, die von der gegebenen Blackbox mit einer Konfidenz von mindestens 90% als beliebige Verkehrsschilder klassifiziert werden. Das einzige, was an Information über die Blackbox gegeben wurde, war das zum Training verwendete Datenset: Der German Traffic Sign Recognition Benchmark (GTSRB). Über eine Webschnittstelle konnten Bilder von der Blackbox klassifiziert werden.

Eine weitere Schwierigkeit stellt dabei das Anfragelimit an die Blackbox dar. Die Klassifizierungsanfragen waren auf 60 Bilder pro Minute begrenzt.

2. Theoretischer Hintergrund

2.1 Anfälligkeit gegenüber Störbildern

Die Anfälligkeit von Neuronalen Netzen gegenüber gezielt manipulierten Störbildern wurde erstmals 2013 von Szegedy et al. untersucht und auch wenn die genauen Hintergründe für diese Schwachstelle noch lang ungeklärt blieben, lässt sie sich heute genauer erklären. Um neuronale Netze möglichst effektiv trainieren und optimieren zu können, neigt man in der Regel dazu, ihr Verhalten während des Trainings möglichst linear zu halten. Selbst bei der Verwendung von vergleichsweise nichtlinearen Aktivierungsfunktionen, wie sigmoid oder softmax, ist man in der Regel bestrebt, eine Sättigung zu vermeiden und sich im quasi-linearen Mittelteil der Funktionen zu bewegen. Dies führt dazu, dass die Netze sehr große Gradienten bezüglich der input-Werte bilden. Während dies zum einen natürlich ein effektiveres Lernen ermöglicht, bedeutet dies ebenfalls, dass diese Netze auf sehr geringe Änderungen der input-Werte mit sehr großen Änderungen der output-Werte reagieren. Dies wiederum bedeutet, dass lediglich wenige, oft für das menschliche Auge sogar unsichtbare, Manipulationen von Bildern nötig sind, um das Verhalten des Netzes auf diese Bilder grundsätzlich zu verändern. Dieser Zusammenhang tritt noch stärker zu Tage, umso größer die Auflösung der inputs ist, da der Manipulierende dadurch einfach mehr (und für die menschliche Wahrnehmung dadurch wesentlich weniger einflussreiche) Bildpunkte zu Verfügung hat, um seine gewünschte Reaktion zu erreichen und bei Bedarf zu verbergen.

2.2 Angriffsmöglichkeiten

Basierend auf den Hintergründen dieser Anfälligkeit, lassen sich verschiedene Angriffsmöglichkeiten formulieren, mit denen sich Störbilder für ein gegebenes Modell eines Neuronalen Netzes erstellen lassen. Die Formeln zu den folgenden Angriffen folgen weitestgehend der von Karakin et al. eingeführten Nomenklatur:

- X - Ein Eingabebild, also i.d.R. ein dreidimensionaler Tensor
- y_{true} - Die "wahre" Klasse des Eingabebilds, also die Reaktion des Netzes auf das nicht-manipulierte Bild
- $J(X, y)$ - Das Cross-Entropy-Loss des Netzes bei gegebenem Bild X und output y
- $Clip_{X, \epsilon}\{X'\}$ - Eine Funktion, die ein pixelweises Clipping des Bildes X' durchführt, sodass die Werte maximal um ϵ vom Original X abweichen

2.2.1 FGSM

Eine der ersten und noch immer populärsten Wege Störbilder zu generieren nennt sich FGSM - Fast Gradient Sign Method. Diese Methode wurde bereits 2014 von Goodfellow et al. vorgestellt und funktioniert folgendermaßen: Anstatt die berechneten Gradienten bezüglich eines inputs zu dazu zu verwenden, die Gewichte des Netzes zu verändern und ein möglichst niedriges loss zu erreichen, wird der input verändert, um ein möglichst hohes loss zu bekommen und somit eine Fehlklassifizierung zu erwirken.

$$X^{adv} = X + \epsilon \text{sign}(\nabla_X J(X, y_{true})) \quad (2.1)$$

2.2.2 Iterative Methode

Die von Karakin et al. eingeführte iterative Methode ist eine Erweiterung von FGSM, bei der FGSM mehrfach nacheinander angewendet wird.

$$X_0^{adv} = X, X_N^{adv} = \text{Clip}_{x,\epsilon}\{X_N^{adv} + \alpha \text{sign}(\nabla_X J(X_N^{adv}, y_{true}))\} \quad (2.2)$$

Der Wert α beschreibt hierbei die Größe der Änderung der Pixelwerte in jedem Schritt.

2.2.3 Methode zum Erreichen einer bestimmten Klasse

Die beiden vorhergehenden Methoden haben lediglich als Ziel bei dem entsprechenden Netz eine Fehlklassifizierung hervorzurufen. Um die als output eine bestimmte Klasse zu bekommen, wird die Iterative Methode leicht abgewandelt:

$$X_0^{adv} = X, X_{N+1}^{adv} = \text{Clip}_{x,\epsilon}\{X_N^{adv} - \alpha \text{sign}(\nabla_X J(X_N^{adv}, y_W))\} \quad (2.3)$$

Wobei y_W dem Wert der gewünschten Klasse entspricht.

2.3 Angriffe gegen eine Blackbox

Die dargestellten Methoden, Störbilder zu generieren haben alle auf dem ersten Blick einen gemeinsamen Schwachpunkt: Man benötigt Zugang zum Modell des Neuronalen Netzes, über den man bei industriellen Anwendungen als Außenstehender nicht ohne weiteres verfügen dürfte. Allerdings täuscht dieser erste Eindruck. 2016 zeigten Papernot et al. dass Störbilder, die für eine bestimmte Machine Learning Lösung generiert wurden, ebenso auf andere Lösungen anwendbar sind, solange diese Algorithmen die gleiche Aufgabe lösen. Das bedeutet, dass Störbilder, die zum Beispiel für ein Neuronales Netz zur Identifizierung von Straßenverkehrsschildern generiert wurden, für ein anderes, unbekanntes Netz und sogar für andere Strukturen, wie Logistische Regression oder Entscheidungsbäume verwendet werden kann. Um nun Störbilder für eine Blackbox zu generieren, ist es ausreichend, ein eigenes Neuronales Netz zu trainieren, das die gleiche Aufgabe löst und mit Hilfe der oben genannten Methoden Störbilder für dieses Netz zu generieren.

3. Methoden

3.1 Entscheidung über die zu verwendenden Methoden

Auch wenn es zur Generierung von Störbildern inzwischen neuere Ansätze als die Vorgestellten gibt, haben diese in der Regel einen entscheidenden Nachteil: den Zeitaufwand. Nicht nur, dass das Experimentieren mit und das Trainieren von einem oder sogar mehreren Neuronalen Netzen - sehr interessante Lösungen wie das AdvGanNet verwenden drei zusammenarbeitende Netze - sehr viel Zeit kostet und dadurch schnell den Zeitrahmen des Cups erschöpfen können, auch die technische Vorgabe des Anfragelimits an die Blackbox ist ein Faktor. Um zum Beispiel unter Verwendung einer von Hinton et al. vorgestellten Destillation der Blackbox ein exakteres Ersatznetz zu erstellen, lässt die Limitierung auf 60 Anfrage pro Minute seitens der Blackbox dieses Vorhaben schnell unrealistisch erscheinen.

Die Verwendung der im Kontext "klassischen" Methode des iterativen FGSM bietet qualitativ hochwertige Ergebnisse und lässt dabei Raum für Verbesserungen an anderer Stelle, wie eine erleichterte Bedienung über die Bereitstellung einer funktionalen Benutzeroberfläche.

3.2 Softwarearchitektur

Das Zentrum der Softwarearchitektur bildet das Nutzerinterface, das durch das Skript `gui.py` generiert wird. Dieses bietet die beiden Hauptfunktionalitäten: Das Generieren bzw. Auswählen eines Basisbilds und das darauffolgende Generieren eines Störbilds basierend auf dem gewählten Basisbild. Alle Funktionen bezüglich des Basisbilds, wie das generieren eines zufälligen oder einfarbigen Basisbilds oder das auswählen und kopieren eines nutzergewählten Basisbilds sind im Skript `generateimage.py` realisiert. Die Funktionen zur Generierung der Störbilder sind im Skript `generateadv.py` umgesetzt. Bei der Generierung der Störbilder wird außerdem ein vorher trainiertes Ersatznetz verwendet, das im Skript `modelcnn.py` umgesetzt ist und das die trainierten Gewichte aus der Datei `savedmodel-stateCNNfinal.pth` lädt.

3.3 Generierung der Störbilder

Zur Generierung der Störbilder wurde die im Kapitel "Theoretischer Hintergrund" dargestellte Iterative Methode zum Erreichen einer bestimmten Klasse realisiert. Der Nutzer hat dabei über das Nutzerinterface die Möglichkeit für die Fehlklassifizierung auf eine bestimmte Klasse zu zielen oder Störbilder für alle 43 Klassen generieren zu lassen. Die Parameter für

die Anzahl der Iterationen, ϵ und α können dabei angegeben werden, um z.Bsp. optisch eher abstrakte, aber sehr schnell zu generierende Störbilder über die Wahl hoher Werte für ϵ und α zu produzieren oder das Basisbild für das menschliche Auge nahezu unsichtbar zu verändern, indem niedrige Werte für ϵ und α gewählt werden. Bei der Wahl niedriger Werte für ϵ und α muss allerdings eine entsprechend höhere Anzahl an Iterationen durchgeführt werden, was selbstverständlich die Laufzeit der Generierung erhöht.

3.4 Das Ersatznetz

Wie im Kapitel "Theoretischer Hintergrund" dargelegt, wird zum Angriff einer Blackbox mit der verwendeten Methode eine Ersatzlösung maschinellen Lernens benötigt, die das gleiche Problemfeld bearbeitet. Hierfür wurde ein Convolutional Neural Network auf die Klassifizierung von Straßenverkehrsschildern mithilfe des GTSB-Datensets trainiert. Das Netz ist dabei wie folgt aufgebaut: Drei Convolutions ($c = 16, 32, 32$; $k = 5, 5, 3$; $s = 1$), jeweils gefolgt von einem Maxpooling und einer Relu-Nonlinearity. Danach zwei fully-connected Layer mit 256 Knoten, die als Output Klassifizierungswahrscheinlichkeiten für die 43 Klassen generieren. Jeweils vor jeder fully-connected Layer ist eine Dropout-Layer eingezogen. Als Loss wird das Cross-Entropy-Loss verwendet.

3.5 Testing

Aufgrund der Übersichtlichkeit der Architektur und der Festlegung auf ein Referenzsystem konnte das Testen der Software mit einem eher "klassischen" Ansatz, das heißt dynamische Tests der beiden Hauptfunktionen unter Verwendung verschiedener Basisbilder und Parameter, bewältigt werden. Um eine möglichst breite Auswahl an Basisbildern zu haben, wurden zum Test Bilder aus dem — Datenset verwendet. Ebenso wurde das Funktionieren unter der Verwendung verschiedener Hardware sichergestellt (da über diese bei der Vorstellung der Referenzplattformen keine Informationen gegeben wurden). Insbesondere meint dies das Vorhandensein einer Cuda-fähigen Grafikkarte.

3.6 Wartbarkeit

Da die verwendete Methode abgesehen vom Ersatznetz ein universelles Vorgehen zur Generierung von Störbildern ist, lässt sich das System mit minimalen Anpassungen auch für andere Aufgaben verwenden. So kann es durch die Verwendung eines anderen Ersatznetzes schnell zur Generierung für Störbilder für eine andere Blackbox oder direkt für eine Whitebox benutzt werden.

4. Evaluation

The evaluation was done using k-fold cross validation. Two of the three videos were used as data for training and one as a test set for validation.

5. Diskussion

5.1 Nachteile

Im Gegensatz zu intelligenteren Ansätzen, liegt die Verantwortung zum Finden effektiver Parameter beim Nutzer. Dies bedeutet, dass der Benutzende ein tieferes Verständnis für die theoretischen Hintergründe und die daraus resultierende Intuition für die verwendeten Parameter benötigt. Zudem hängt die Qualität der generierten Störbilder auch vom verwendeten Ersatznetz ab. Sollte die Blackbox bereits mit Gegenmaßnahmen für Störbilder ausgestattet sein (wurde sie zum Beispiel bereits mit Störbildern trainiert) oder ist das zum Training verwendete Datenset nicht bekannt, ist davon auszugehen, dass die Effektivität der generierten Störbilder abnimmt. Als weiterer Nachteil (besonders im Hinblick auf einen weiter unten beleuchteten praktischen Einsatz) ist die Limitierung auf ein Basisbild zu sehen. Die Möglichkeit mehrere Basisbilder zu verwenden und ein gut geordneter Output der generierten Störbilder wäre im Kontext der automatischen Erstellung einer großen Anzahl an Störbildern wünschenswert.

5.2 Verbesserungsmöglichkeiten

Die Verwendung einer Destillation oder eines dynamisch trainierten Ersatznetzes könnte die Effektivität der Lösung erhöhen und darüber hinaus auch in der Lage sein, etwaigen Gegenmaßnahmen entgegenzuwirken. Über die Integration des Trainings oder der Auswahl eines zu verwendeten Ersatznetzes in die Benutzeroberfläche kann die vorliegende Lösung zu einer all-Purpose-Lösung für die Generierung von Störbildern für beliebige Zielnetze.

5.3 Praktischer Einsatz

Da diese Arbeit natürlich nicht dazu verwendet werden soll, die in der Einführung erwähnten schweren Unfälle zu verursachen, sondern zu verhindern, ist der Praktische Nutzen der Lösung die Generierung von Störbildern, mit denen wiederum andere Machine Learning Lösungen trainiert werden, um deren Anfälligkeit gegen Störbilder zu vermindern. Dies ist zur Zeit der am weitesten verbreitete Weg Neuronale Netze resistent gegen Störbilder zu machen.

Abbildungsverzeichnis

List of Algorithms

Literaturverzeichnis