Irrbilder für InformatiCup 2019

Theoretische Ausarbeitung

Alexander Kretzschmar, Marvin Springer

TU Dresden / TU Braunschweig

Zusammenfassung

Für den InformatiCup 2019 sollten für einen gegebene Blackbox Irrbilder erzeugt werden, die mindestens eine Konfidenz von 90% erreichen. Wir verwenden dazu den Iterativen FGSM-Algorithmus und erreichen damit Konfidenzen von 100%. Zusätzlich dazu erstellen wir Irrbilder mit minimaler optischer Distanz zu ihren Basisbildern. Diese erreichen leicht verringerte Konfidenzen von über 99%.

Inhaltsverzeichnis

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 *Adversarials*, gezielt generierten Irrbildern, die Neuronale Netze mit hoher Zielsicherheit falsche Ergebnisse produzieren lassen. Besonders die Tatsache, dass sich die Methoden zur Generierung solcher Irrbilder von einem Modell auf andere übertragen lassen, verschärft die Problematik. Denn das bedeutet: Selbst ohne detailliertes Wissen über ein spezifisches Netz, wie dessen genaue Struktur, lassen sich zuverlässig Irrbilder generieren.

Insbesondere die Verwendung Neuronaler 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 Irrbildern in diesem Feld unterstreichen noch einmal deutlich die Wichtigkeit dieses Themas. Das Anbringen für das menschliche Auge nicht erkennbarer Irrbilder in der Nähe von Fahrbahnen kann schwerste Unfälle hervorrufen.

1.2 Ziele

Das Ziel im Rahmen des informatiCup 2019 war es, Irrbilder 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 Irrbildern

Die Anfälligkeit von Neuronalen Netzen gegenüber gezielt manipulierten Irrbildern wurde erstmals 2013 von Szegedy et al. [1] 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 [2]. 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 [3].

2.2 Angriffsmöglichkeiten

Basierend auf den Hintergründen dieser Anfälligkeit, lassen sich verschiedene Angriffsmöglichkeiten formulieren, mit denen sich Irrbilder für ein gegebenes Modell eines Neuronalen Netzes erstellen lassen. Die Formeln zu den folgenden Angriffen folgen weitestgehend der von Kurakin et al. [4] 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 Irrbilder 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* 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.

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

2.2.2 Iterative Methode

Die von Kurakin et al. [4] eingeführte iterative Methode ist eine Erweiterung von FGSM, bei der FGSM mehrfach nacheinander angewendet wird.

$$X_0^{adv} = X, X_N^{adv} = Clip_{x,\epsilon}\{X_N^{adv} + \alpha sign(\nabla_X J(X_N^{adv}, y_{true}))\} \tag{2.2}$$

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 Irrbilder für eine bestimmte Klasse zu generieren, wird die Iterative Methode leicht abgewandelt:

$$X_0^{adv} = X, X_{N+1}^{adv} = Clip_{X,\epsilon}\{X_N^{adv} - \alpha sign(\nabla_X J(X_N^{adv}, y_W))\} \tag{2.3}$$

Wobei y_W dem Wert der gewünschten Klasse entspricht.

2.3 Angriffe gegen eine Blackbox

Die dargestellten Methoden, Irrbilder 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. [5] dass Irrbilder, die für eine bestimmte Machine Learning Lösung generiert wurden, ebenso auf andere Lösungen anwendbar sind, solang diese Algorithmen die gleiche Aufgabe lösen. Das bedeutet, dass Irrbilder, 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 Irrbilder 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 Irrbilder für dieses Netz zu generieren [6].

3. Methoden

3.1 Entscheidung über die zu verwendenden Methoden

Auch wenn es zur Generierung von Irrbildern 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 [7] 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. [8] 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 Irrbilds 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 Irrbilder sind im Skript *generateadv.py* umgesetzt. Bei der Generierung der Irrbilder wird außerdem ein vorher trainiertes Ersatznetz verwendet, das im Skript modelcnn.py umgesetzt ist und das die trainierten Gewichte aus der Datei *saved_model_state_CNN_final.pth* lädt.

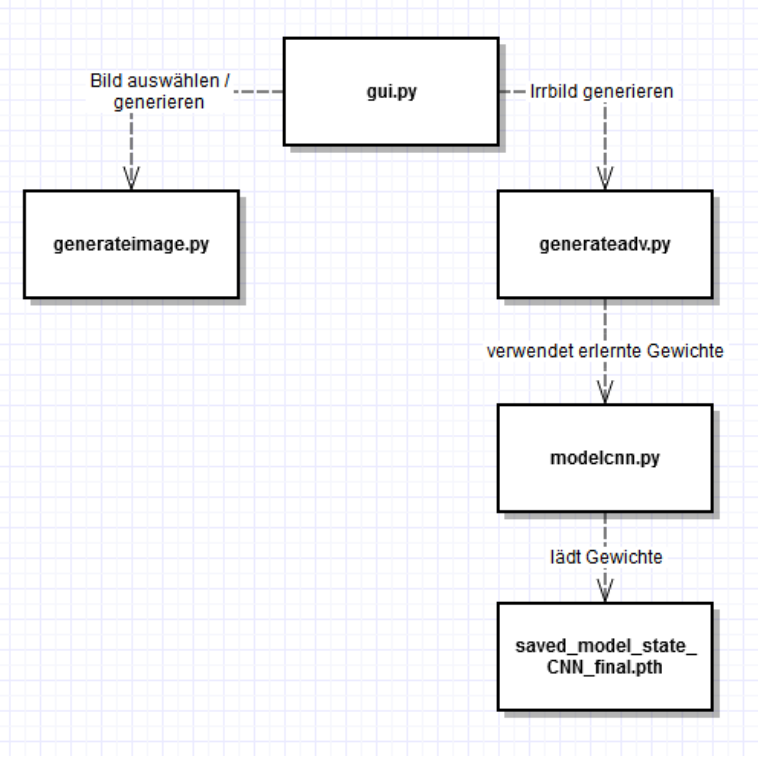

Abbildung 3.1: Die Systemarchitektur

3.3 Generierung der Irrbilder

Zur Generierung der Irrbilder 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 Irrbilder für alle 43 Klassen generieren zu lassen. Die Parameter für die Anzahl der Iterationen, ϵ und α können dabei angegeben werden, um zum Beispiel optisch eher abstrakte, aber sehr schnell zu generierende Irrbilder 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 machinellen 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.

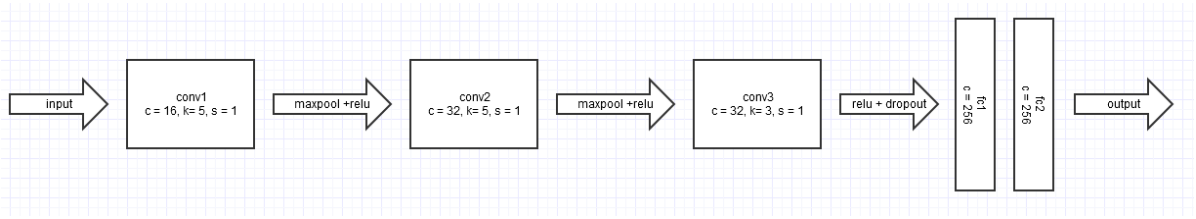

Abbildung 3.2: Die Netzarchitektur

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 *Caltech 101* 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 Irrbildern darstellt, 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 Irrbilder für eine andere Blackbox oder direkt für eine Whitebox benutzt werden.

4. Evaluation

4.1 Einfarbige Bilder mit Standardwerten

Die hier gezeigten Bilder wurden aus einfarbigen Basisbildern unter verwendung der Standardwerte, also 30 Iterationen, einem ϵ von 0,25 und einem α von 0,025 erstellt. Sie erreichen alle bei der Blackbox eine Klassifizierung als Straßenschild mit einer Konfidenz von 100 %.

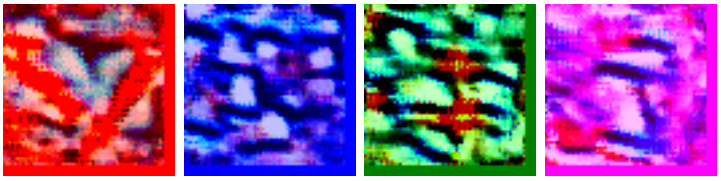

Abbildung 4.1: Irrbilder, generiert aus einem einfarbigen Basisbildern

4.2 Bilder aus nutzergewählten Basisbildern

Als nächstes wurden Bilder aus nutzergewählten Fotografien erstellt. Die folgenden Bilder dienten dazu als Basisbilder:

Abbildung 4.2: Die verwendeten Basisbilder

Folgende Bilder wurden ebenfalls unter Verwendung der Standardwerte aus den oben genannten Fotografien erstellt.

Sie erreichen ebenfalls eine Konfidenz von 100 %, die Basisbilder sind aber durch die starke Transformation kaum wiederzuerkennen.

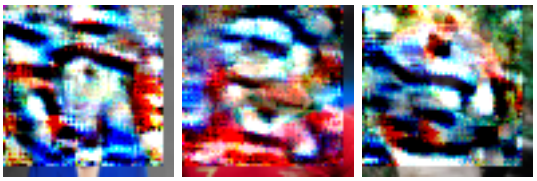

Abbildung 4.3: Die generierten Irrbilder

4.3 Irrbilder mit für den Menschen unkenntlichen Änderungen

Als letztes wurde mit verschiedenen Parametern experimentiert, um Bilder zu erzeugen, die mit möglichst hohen Konfidenzen von der Blackbox akzeptiert werden, aber eine weitaus niedrigere optische Distanz zu ihren Basisbildern aufweisen. Als Basisbilder wurden die gleichen Bilder wie im vorhergehenden Teil benutzt. Die verwendeten Parameter waren dabei folgende: 500 Iterationen, mit einem ϵ von 0,01 und einem α von 0,0001.

Abbildung 4.4: Die generierten Irrbilder mit möglichst geringer optischer Distanz zu ihren Basisbildern

Diese Bilder erreichen im Gegenzug allerding keine perfekte Konfidenz. Frau Merkel erreicht eine Kofidenz von 99,67%, Herr Ronaldo eine Konfidenz von 99,72% und der Panda immerhin eine Konfidenz von 99,95%.

5. Diskussion

5.1 Nachteile

Im Gegensatz zu ïntelligenterenÄnsätzen, 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 Irrbilder auch vom verwendeten Ersatznetz ab. Sollte die Blackbox bereits mit Gegenmaßnahmen für Irrbilder ausgestattet sein (wurde sie zum Beispiel bereits mit Irrbildern trainiert) oder ist das zum Training verwendete Datenset nicht bekannt, ist davon auszugehen, dass die Effektivität der generierten Irrbilder 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 Irrbilder wäre im Kontext der automatischen Erstellung einer großen Anzahl an Irrbildern 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. Außerdem kann über die Integration des Trainings oder der Auswahl eines zu verwendeten Ersatznetzes in die Benutzeroberfläche die vorliegende Lösung zu einer *all-purpose*-Lösung für die Generierung von Irrbildern für beliebige Zielnetze ausgebaut werden.

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 Irrbildern, mit denen wiederum andere Machine Learning Lösungen trainiert werden, um deren Anfälligkeit gegen Irrbilder zu vermindern. Dies ist zur Zeit der am weitesten verbreitete Weg, Neuronale Netze resistent gegen Irrbilder zu machen.

Abbildungsverzeichnis

Literaturverzeichnis

[1] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks." *CoRR*, vol. abs/1312.6199, 2013. [Online]. Available: http://dblp.uni-trier.de/db/journals/corr/corr1312.html#SzegedyZSBEGF13

[2] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[3] C.-J. Simon-Gabriel, Y. Ollivier, B. Schölkopf, L. Bottou, and D. Lopez-Paz, "Adversarial vulnerability of neural networks increases with input dimension." *CoRR*, vol. abs/1802.01421, 2018. [Online]. Available: http://dblp.uni-trier.de/db/journals/corr/corr1802.html#abs-1802-01421

[4] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," 2016, cite arxiv:1607.02533Comment: 14 pages, 6 figures. Demo available at https://youtu.be/zQ_uMenoBCk. [Online]. Available: http://arxiv.org/abs/1607.02533

[5] N. Papernot, P. D. McDaniel, and I. J. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples." *CoRR*, vol. abs/1605.07277, 2016. [Online]. Available: http://dblp.uni-trier.de/db/journals/corr/corr1605.html#PapernotMG16

[6] N. Papernot, P. D. McDaniel, I. J. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning." in *AsiaCCS*, R. Karri, O. Sinanoglu, A.-R. Sadeghi, and X. Yi, Eds. ACM, 2017, pp. 506–519. [Online]. Available: http://dblp.uni-trier.de/db/conf/ccs/asiaccs2017.html#PapernotMGJCS17

[7] C. Xiao, B. Li, J.-Y. Zhu, W. He, M. Liu, and D. Song, "Generating adversarial examples with adversarial networks." in *IJCAI*, J. Lang, Ed. ijcai.org, 2018, pp. 3905–3911. [Online]. Available: http://dblp.uni-trier.de/db/conf/ijcai/ijcai2018.html#XiaoLZHLS18

[8] G. E. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network." *CoRR*, vol. abs/1503.02531, 2015. [Online]. Available: http://dblp.uni-trier.de/db/journals/corr/corr1503.html#HintonVD15