Institut für Informationssicherheit

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Seminarausarbeitung

Adversarial Classification

Jonas Geiselhart

Studiengang:	Informatik
Veranstaltung:	Seminar Informationssicherheit und Kryptographie
Semester:	SoSe 2021
Veranstalter/in:	Prof. Dr. Ralf Küsters
Betreuer/in:	Dr. Pascal Reisert

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift

(a) Stop Sign - 99.85% (b) 120kmh Sign - 99.91%

Abbildung 1: Adversary Example

1 Einleitung

Adversarial Classification ist ein Problem, das wissenschaftlich zuerst von *Dalvi et al.* [2] thematisiert wird. Man kann das Problem informell und allgemein formulieren, als Angriff bei dem aktiv und bösartig manipulierte Daten in einen Classifier gegeben werden, um eine fehlerhafte Klassifizierung zu erhalten. Im Gegensatz zu Training-Data-Poisioning erfolgt der Angriff zur Anwendungszeit. Inzwischen ist Adversarial Input nicht nur ein Problem in Klassifikationsaufgaben, sondern auch in vielen anderen Domänen wie z.B.Visual Segmentation oder generellerem Natural Language Processing (NLP) Problemen.
Ein Beispiel für ein Adversarial Example wie in Abb. 1 kann relativ simpel selbst gemacht werden[1]. Dabei ist das Konzept nahezu ohne Änderung auf alle möglichen Aufgaben übertragbar [8]. Da nun immer mehr maschinelles Lernen in sicherheitskritischen Bereichen eingesetzt wird, zeigt Adversarial Learning eine potenziell ausnutzbare Schwachstelle auf. Allerdings ist in der Realität noch kein Angriff oder Angriffsversuch bekannt geworden, in dem diese Technik benutzt wurde, um aktiv Schaden anzurichten. Obgleich in Realversuchen sich schon eingesetzte Systeme in z.B. autonomem/assistiertem Fahren, Bilderkennung oder Voice-Assist Programmen anfällig für solche Attacken gezeigt haben [1, 5]. Dessen ungeachtet kann man über Adversarial Input auch Funktionsweise und Eigenschaften von Methoden zum Maschinellem Lernen besser erforschen und tiefer gehende Informationen zum Entscheidungs- und Berechnungsprozess extrahieren [4]. Somit ist das grundlegende Problem nicht nur aus sicherheitstechnischer Perspektive interessant.

2 Restriktionen & Thread Model

Es sind viele verschiedene Thread Models, die jeweils immer neue Zwecke und Anwendungsfälle modellieren. Dabei beschreibt ein Thread Model die Grundsituation, also z.B. welche Informationen gegeben sind und welche angefragt werden können, wie die Perturbation gemessen und limitiert wird, oder welches Ziel genau zu erreichen ist.
Grundsätzlich gibt es zwei Informationsszenarien, *Blackbox* , also keine Informationen über das anzugreifende System, und *Whitebox*, also perfekte Information über das System. Letzteres ist für die Forschung interessanter und ersteres ist besonders anwendungsbezogen. In der Realität ist die übliche Kondition, dass zwar ein Blackbox System gegeben ist, aber Ergebnisse (bspw. Gradienten) abgefragt werden können, dies ist am realistischen, da man öffentliche Zielsysteme i.A. wiederholt und mit nicht spezifiziertem Input benutzen kann. Mit dieser Kondition kann man auch viele Whitebox Algorithmen benutzen, in dem man dynamisch Informationen vom Zielsystem abfragt.
Oftmals unterscheiden sich die Algorithmen zur Adversarial Classification in der eigentlichen Zieldefinition, wobei in der gesamten Ausarbeitung nur Klassifikationsprobleme betrachtet werden. Dabei sind die Zielsetzungen immer unterschiedlich, die zwei gängigsten sind entweder „General Misclassification “, also das Ändern der Vorhersage des Classifiers, falls dieser die richtige Klasse vorhersagen würde, oder die

[1] aus https://kennysong.github.io/adversarial.js/

„Targetted Misclassification“, wobei die neue Zielklasse spezifiziert wird.
Als letzten großen Unterschied im Thread Model, gibt es verschiedene Perturbations-Normen. Also Wege, die Veränderung im Bild zu messen. Besonders häufig trifft man auf L_0, L_2, und L_∞. Im Allgemeinen wird die Veränderung über p-Normen L_p angegeben.[8] Dies ist durchaus umstritten, da die Normen nicht unbedingt der semantischen, also vom Menschen erkannten, Veränderung entsprechen. Eine der menschlichen Wahrnehmung besser angepasste Norm, ist eine noch offene Forschungsfrage.
Im weiteren Verlauf wird immer ein White-Box Thread Model mit General Misclassification angenommen, die Perturbation ist immer entweder geringstmöglich oder durch ein Epsilon (ε) beschränkt, somit muss keine Norm festgelegt werden.

3 Problem & Formale Definition

Dalvi et al. [2] haben Adversarial Classification als ein Spiel zwischen zwei Spielern, dem ADVERSARY und dem CLASSIFIER, modelliert. Dabei gibt es zwei Instanzmengen von möglichen Inputs, jeweils beschrieben als $X = (X_1, ..., X_i, ..., X_n)$, wobei jeweils die Vorhersage der einzelner Attributklassen durch einzelne Werte beschrieben werden. Jede Instanz kann entweder gutartig (-), also nicht vom Adversary verändert, oder bösartig (+), also vom Adversary perturbiert, sein.
Ziel des Classifiers ist es nun für jede Instanz einer gegebene Testmenge $\mathcal{T}$ die richtigen Klassifikation vorzunehmen. Ziel des Adversary hingegen ist es, bei möglichst vielen Klassen die Klassifikation zu ändern.
Nun können beide Akteure anhand einer Menge an Parametern beschrieben werden, die jeweils für Instanzen und über $\mathcal{T}$ aggregiert jeweils Kosten und Nutzen modellieren. Demnach kann der Classifier $\mathcal{C}$ über V_i die Kosten zum Messen des i-ten Feature, und $U_C(y_c, y)$ den Nutzen, der entsteht wenn man eine Instanz mit Klasse y als y_c klassifiziert beschrieben werden. Der Adversary hat die Parameter $W_i(x_i, x_i')$ die Kosten im Classifier das Feature x_i zum Feature x_i' zu wechseln, diese Kosten können auch mit W über alle Features einer Instanz aggregiert werden, und $U_A(y_c, y)$, den Nutzen, der entsteht, falls der Classifier y_c als y klassifiziert.
Mit diesen Parametern haben *Dalvi et al.* nun Gütefunktionen aufgestellt, die für eine optimale Strategie maximiert werden müssen. Für den Classifier wird der Nutzen beschrieben mit

$$U_C = \sum_{(x,y)\in XY} P(x,y)[U_C(\mathcal{C}(\mathcal{A}(x)), y) - \sum_{x_i \in \chi_{\mathcal{C}(x)}} V_i] \tag{1}$$

und für den Adversary mit

$$U_A = \sum_{(x,y)\in XY} P(x,y)[U_A(\mathcal{C}(\mathcal{A}(x)), y) - W(x, \mathcal{A}(x))] \tag{2}$$

Es ist P die Verteilung unter der die initialen Instanzen des Testsets generiert wurden. Dabei haben *Dalvi et al.* entdeckt, dass es sich um ein Nash-Gleichgewicht handelt, das heißt es gibt für beide Akteure eine perfekte Strategie, bei der keiner etwas gewinnt, indem er seine Algorithmen ändert. Bestenfalls kann das Gleichgewicht in einem Non-Zero-Sum Spiel, wie es hier der Fall ist, in exponentieller Zeit berechnet werden. Zudem sind die Kosten in reellen Anwendungen nicht seriös zu beziffern. Die Auswirkungen dieser Kosten-Nutzen-Analyse sind zwar begrenzt, da man folglich für Anwendungssituationen dieses Gleichgewicht nicht berechnen kann, aber es deutet sich schon an, dass Adversarial-Input kein Problem ist, das mit bis jetzt bekannten Mitteln durch Classifier komplett kompensiert werden kann.

4 Strategien

Mithilfe der Formeln für U_C und U_A kann man nun optimale Strategien formulieren, die jeweils den Nutzen maximieren. Der Pseudocode zur Implementierung ist hierbei je nach angewendetem ML-Algorithmus zur Klassifikation unterschiedlich, dennoch bleibt die Grundstruktur gleich und erzielte Ergebnisse vergleichbar. Dabei gilt wie üblich, dass die Akteure alle Parameter sowohl von sich selbst, als auch von dem Gegner kennen (*complete information).* Zusätzlich geht der Adversary davon aus, dass seine Manipulation von dem Classifier unendeckt bleibt.

4.1 Adversary-Strategie

Die optimale Adversary-Strategie besteht aus zwei sequentiellen Schritten, zuerst ist die Aufgabe zu berechnen ob eine Perturbation einen Nutzgewinn darstellt, und falls das der Fall ist, ist die Aufgabe die optimale Perturbation zu berechnen. Letzteres wird von *Dalvi et al.* als eigenes Problem Minimum Cost Camouflage (MCC) beschrieben. MCC hat als Input eine Instanz, deren Klassifikation geändert werden soll und als Ausgabe eine andere Instanz deren Veränderung bzgl. der gegebenen Norm minimal ist, aber trozdem vom Classifier falsch erkannt wird. Dabei hat jede aller möglichen atomaren Perturbationen (bsp. bei Bildern ein Pixelwert) sowohl Kosten, als auch einen Nutzen. Die Aufgabe, die beste Kombination aus atomaren Perturbationen zu finden ist NP-hart und es ist mit begrenzten Resourcen nicht realistisch diese optimale Methode in die Praxis umzusetzen. Allerdings sind diskretisierte Versionen dieses Problems relativ einfach zu berechnen und werden auch in der Praxis eingesetzt (siehe Unterabschnitt Adversarial Input generierenl).
Im Gegensatz zum zweiten Schritt, ist der Schritt relativ trivial. Es muss entschieden werden, ob diese Instanz grundsätzlich manipuliert werden sollte. Für den Fall, dass der Classifier schon eine falsche Klasse erkennt, lohnt es sich nicht das Bild zu manipulieren und für den Fall, dass die Kosten höher sind, als der Gewinn an Nutzen ($W(x, MCC(x)) < \Delta U_A$) lohnt es sich auch nicht das Bild zu manipulieren, es wird der Input unverändert zurückgegeben.

4.2 Classifier-Strategie

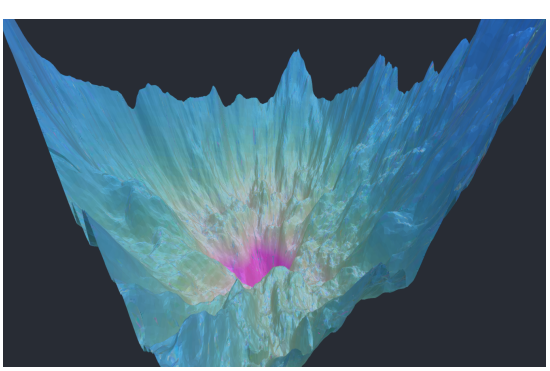
Die optimale Classifier Strategie nutzt die Annahmen, dass der Adversary perfekte Information hat und damit die optimale Strategie nutzt sowie einige Annahmen über die Kosten die es benötigt ein Feature zu ändern, genauer, dass diese größer gleich Null sind sowie, dass eine direkte Änderung immer die kostengünstigste Variante darstellt (Dreiecksungleichung). Vorerst ist das Trainingsset in dem Modell nicht manipuliert oder verändert, es werden also nur reaktive Strategien, die den Classifier nicht ändern betrachtet. Der Algorithmus zur Klassifizierung von Instanzen muss so gemacht sein, dass $C(x)$ den Nutzen U_C (siehe Sektion 3) maximiert. Dazu klassifiziert er den Input x jeweils als bösartig oder gutartig. Dieses passiert indem man sich $U(+|x)$ und $U(-|x)$, den Nutzen einer Klassifikation gegeben dem Input x, ausrechnet. Dazu stellt der Classifier die bedingten Wahrscheinlichkeiten auf, dass der Input ein +/- Instanz ist, und verrechnet diese mit dem jeweiligem Nutzen $U(x, y)$ mit $U > 0$ g.d.w. $x = y$ ($x, y \in \{+, -\}$). Nur falls am Ende der Nutzen die Instanz als eine böswillige Instanz zu deklarieren höher ist, wird ein Adversary Input erkannt und kann dementsprechend behandelt werden. Allerdings ist die Wahrscheinlichkeit, dass es sich bei x um ein manipulierten Input handelt sehr schwer zu berechnen. Dazu könnte man in der Theorie bestimmen ob $\exists x' : MCC(x') = x$ gilt, in dem Fall ist die Wahrscheinlichkeit hoch, dass es sich um Adversary Input handelt. Von hier aus beginnen *Dalvi et al.* allerdings damit die Menge möglicher Instanzen x' einzugrenzen, sodass man entweder probabilistisch davon ausgehen kann, dass es ein passendes x' gibt, oder es möglich ist den eingrenzten Bereich vollständig zu verifizieren.

5 In-Practice Strategien

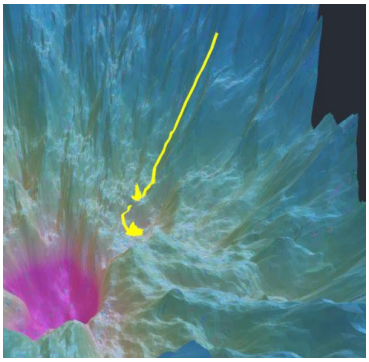
Während die optimalen Strategien einfach zu bestimmen sind, sind anwendbare Strategien, sowohl für Adversary als auch/besonders für den Classifier, schwerer zu konstruieren. Teilweise, weil üblicherweise nicht alle Parameter beider Systeme gegenseitig bekannt sind, teilweise auch weil die Rechenkapazität in der Realitität nicht so hoch ausfällt.

5.1 Adversarial Input generieren

Adversarial Input kann auf viele verschieden Arten generiert werden, je nach Thread Model sind die Daten auf denen gerechnet werden muss stark unterschiedlich.
Eine besonders weit verbreitete Methodik sind generelle *Gradient Decend* Algorithmen [1]. Da diese mitunter die besten Ergebnisse erzeugen und direkt an die theoretisch optimale Strategie anknüpfen sind diese besonders interessant. Die Methode basiert auf einer, dem Classifier inhärente *Loss-Function*, diese nimmt als Argument eine Input-Instanz des Classifiers und gibt als Output an wie „richtig“der Classifier liegt, also wie exakt die Konfidenz in die einzelnen Klassen sind. Dabei wird der Cross-Entropy-Loss, also



(a) Loss-Landscape eines Convolutional Network



(b) Gradient-Descend

Abbildung 2: Loss-Landscape visualisiert (simplifiziert, 3D)

wie viel Entropie durch die Klassifikation verloren geht bestimmt, aber zum generellen Verständnis der Methodik reicht die erstere Erklärung aus. Nun kann man die Loss-Function dynamisch diskretisieren und erhält Daten wie in Abbildung 2 [2] . Damit kann man über eine gute Minimierungsfunktion den Gradienten nutzen um entweder die Vorhersage der Wahrscheinlichkeit eines gewünschten Attributs zu erhöhen oder die Wahrscheinlichkeit der ungewünschten echten Klasse zu minimieren um so ein Input innerhalb einer bestimmten Norm zu finden, sodass eine neue Klassifikation hat. Dabei ist die Optimierungfunktion stark vom eigentlichen Algorithmus abhängig. Es sind oft Randomisierung und häufige Iterationen notwendig, da es sein kann, dass man immer relativ schnell in lokale Minima gerät, in denen man die Klassifizierung nicht geändert hat. Die Informationen, die für diese Methode benötigt werden, sind nur die jeweiligen Ergebnisse aus verschiedenen Input Anfragen für den Gradient Descend. Dabei dient die Visualisierung nur dem Verständnis, es werden nur Punkte diskretisiert die für die eigentliche Berechnung wichtig sind.

5.2 Adversarial Defenses

Im Gegensatz zu der optimalen Strategie nach *Dalvi et al*[2] ist das Hauptziel in den Anwendungsbereichen, nicht den Adversarial Input zu erkennen und dann diesen so herauszufiltern, also eine reaktive Strategie, sondern es geht eher darum die Robustheit des Classifiers direkt zu erhöhen, sodass ein potentielles Adversarial Input im Vorhinein richtig klassifiziert wird. Also werden die Kosten zur Konstruktion von Adversarial Input erhöht. Es wird eine proaktive Strategie verfolgt, denn in der Realität haben sich reaktive Strategien schnell als zu zeit- und resourcenaufwendig herausgestellt. Dadurch entstehen wieder andere Schwächen, allerdings ist es im Single Attack Thread Model deutlich effektiver. Ein großes Problem mit Adversarial Defenses ist, dass die meisten nicht gut skalieren, also nur für sehr kleinen ε Perturbationslimits wirklich gute Ergebnisse liefert. Kleine limitierte Garantien sind nach *Tramèr et al.* [7] in der Regel mit substanziellen Einschränkungen in der generellen Genauigkeit der Classifier verbunden. Es können verschiedene Adversarial Defenses aggregiert werden, auch mit hier nicht genannten Methoden, das bietet einen umfassenderen Schutz [8].

5.2.1 Adversarial Training

Im Adversarial Training werden den Trainingsdaten auch Adversarial Inputs mit zusätzlich den richtigen Klassen hinzugefügt. *Ilyas et al.* [4] haben gezeigt, dass ein Classifier während des Trainings bestimmte Features (nicht mit den späteren Attributen/Klassen zu verwechseln) lernt, also Muster in den Eingabedaten, lernt. Diese Features können sowohl standardmäßig entweder robust sein oder nicht robust. Neu ist, dass Features auch gegenüber Adversarial Classification robust oder nicht robust sein können. Durch Adversarial Training, „bestraft“man den Classifier also dafür Nicht-adversarial-robuste Features zu lernen. Im Gegenzug muss man natürlich einen geringfügigen Verlust in der Genauigkeit gutwilliger Daten

[2] aus https://losslandscape.com/explorer

hinnehmen, da nun die zu lernenden Features weniger vielfältig sind. Im Allgemeinen ist Adversarial Training eine der effektivsten Maßnahmen um die Auswirkungen von Adversarial Input zu minimieren.

5.2.2 Differential Privacy

Differential Privacy ist ein Konzept, das nicht nur speziell auf Adversarial Learning bezogen existiert. Es gehört zu den Methoden, die Certifiable Robustness förden, also unter bestimmten Annahmen Garantien geben, dass es sich nicht um ein Adversarial Input handelt. Die Idee wurde von *Lecuyer et al.* [6] in Form eines Programms mit dem Namen PixelDP vorgestellt. Dabei werden im Classifier eine oder mehrere Noise Layer hinzugefügt, sodass kleine Veränderungen in der Eingabe, üblicherweise angegeben über eine maximale Perturbation ε innerhalb einer Norm, nur kleinen Änderungen in der Ausgabe erzeugen. Dabei kann natürlich nicht immer eine Garantie gegeben werden, dass das Ergebnis richtig ist, da nur Stability Bounds bezüglich verschiedener Klassen aufgestellt werden können, und die Methode kann somit ergänzend genutzt werden. Die Randfälle müssen hier dennoch wieder extra behandelt werden.

6 Weiterführende Arbeiten

Dalvi et al. [2] haben mit ihrem Artikel nicht nur eine Schwachstelle in Classifier-Systemen aufgezeigt und formalisiert, sondern aus Sicht der Informationssicherheit ein ganz neues Angriffsmuster und aus Sicht des Maschinellen Lernens eine neue Methode ML-Systeme zu erforschen konstruiert. Es bilden *Yuan et al.*[8] einen Überblick über alle entwickelten und modellierten Verfahren zum Thema Adversarial Learning. Desweiteren fällt besonders auf, dass hier oft die Forschung deutlich von standardmäßigen Fragen in der Spieltheorie, wie sie *Dalvi et al.* initial in Aussicht gestellt hat, abweichen. Themen wie verschieden Adversaries oder wiederholte Angriffsmodelle nehmen eine untergeordnete Rolle ein. Außerdem ist bemerkenswert, dass proaktive Adversarial Defenses sehr schnell reaktive Methoden verdrängt haben. Letztere erhalten in der aktuellen Forschung nahezu keine Aufmerksamkeit mehr, auch wenn gut vorstellbar ist, dass neue Erkenntnisse zum Adversarial Learning auch dort die Effektivität verbessern könnten. Insgesamt lässt sich noch anmerken, dass sich Adversarial Attacken in vielen Eigenschaften als überraschend effektiv gezeigt haben. Insbesondere der Transfer zwischen Classifier Systemen und zwischen verschiedenen Aufgaben/Thread Models, sowie Methoden zur Konstruktion von Adversarial Input unter unvollständiger Information haben sich als sehr ausgeprägte/gut konditionierte Eigenschaften herausgestellt.
Ein deutlich neuerer Bereich nutzt den Adversarial Input um damit Eigenschaften verschiedener Classifier zu untersuchen: Ziel ist es hier über Randbereiche der Wahrnehmung zu erfassen und einen genaueren Einblick in die Arbeitsweise verschiedener Classifier zu bekommen, indem man die Unterschiede zwischen menschlicher und maschineller Wahrnehmung untersucht. *Ilyas et al.*[4] betrachteten dafür robuste und nicht-robuste Features, die in Trainingsdaten vorhanden sind. Diese Betrachtungsweise, also Adversarial Learning als Kondition der vorliegenden Trainingsdaten, anstatt des Classifiers, bietet nicht nur im Bereich des Adversarial Learnings viele neue mögliche Forschungsansätze.

7 Zusammenfassung und Ausblick

Nun kann man mithilfe von formaler Definition, optimaler Strategie und praktischer Methoden versuchen die Gefahr, die durch Adversarial Classification entsteht, zu beschreiben und Standards zur Verteidigung von Adversarial Attacks aufzustellen. Das ist hier besonders spekulativ, da es noch keinen bekannt gewordenen Fall in der reellen Anwendung gibt, in denen Adversarial Classification genutzt wurde um einen Sicherheitsmechanismus zu umgehen, wobei es in Versuchen mittlerweile sehr zuverlässig funktioniert. Die Gründe hierfür können vielfältig sein, einerseits ist maschinelles Lernen in vielen Domänen noch nicht so zuverlässig, dass es konsequent und ohne menschliche Kontrolle in entsprechend kritischer Infrastruktur eingesetzt werden kann, andererseits ist es auch schwer Adversarial Input als solchen zu erkennen und nicht einfacher als generellen Fehler des Classifiers anzusehen. Als sicher ist es auf jeden Fall anzusehen, dass es bei Adversarial Learning nach jetzigem Forschungsstand nur Möglichkeiten zur Härtung gegen bösartig manipulierten Input gibt, jedoch ohne Zutun von Akteuren außerhalb des Maschinellen Lernens, es keine

sicheren Methoden zur Erkennung und/oder Mitigation solcher Attacken gibt. Der wissenschaftliche Zweig des Adversarial Learning steht aktuell noch ziemlich am Anfang und es gibt viel Forschungspotential.

7.1 Anwendungsgebiete

Im Gegensatz zum ersten Eindruck gibt es starke Einschränkungen für die Anwendbarkeit von Adversarial Attacks und Defenses. Viele intuitive Anwendungsfälle, sowohl im Angriffs- als auch im Verteidigungsfall, stellen sich oftmals entweder als nicht real effizient oder zielführend anwendbar heraus.[3]

Es gibt keine lückenlose Verteidigung gegen Adversarial Attacks, aus diesem Grund sollte es für Systeme, die von einer korrekten Klassifikation abhängig sind, andere Maßnahmen zur Bekämpfung solcher falschen Klassifikationen geben, die auch Adversarial Learning beachten. Als plakatives Beispiel dient autonomes Fahren, es ist nach jetzigem Forschungsstand nicht absehbar, dass jede Attacke direkt vom Classifier behandelt werden kann, hier müssen andere Maßnahmen außerhalb des ML-Systems getroffen werden, solange bis man zuverlässigere Wege finden kann diese Schwächen zu beheben.

Adversarial Learning ist keine gute generelle Methode um einen Classifier zu überlisten, es ist speziell konstruiert um Input so zu manipulieren, sodass es für Menschen schwer zu erkennen ist. Es liegt nahe Adversarial Learning auch als Risiko in Systemen ohne direkten menschlichen Einfluss zu sehen, ein oft angeführtes Beispiel ist die Erkennung von bösartiger Software. Dennoch sind in der Regel die Anforderungen an die Perturbation ganz andere, da es nicht von Menschen, sondern anderen Maschinen unerkannt bleiben sollte.

Zur Applikation von Adversarial Defenses benötigt der Classifier auch eine Reihe unterschiedlicher Eigenschaften, die ihn robust machen. Er soll in erster Linie stabil auf Erhöhung der Adversarial Robustness, durch möglichst viele unterschiedliche Algorithmen, reagieren. Dabei ist noch nicht klar welche Classifier diese Eigenschaft haben, oder wie man diese besonders erzeugen kann. *Ilyas et al.* [4] argumentieren sogar, dass diese Eigenschaften innerhalb des Trainingssets und nicht in dem Classifier selbst zu erzeugen sind, da die Features des Trainingssets laut ihnen in erster Linie für die Existenz von Adversarial Input verantwortlich ist.

7.2 Zukünftige Forschungsfragen

Viele zukünftige Forschungsfragen bezüglich Adversarial Learning stammen eher aus dem Fachbereich Maschinelles Lernen, dort sieht man Adversarial Learning eher als ein Weg, Classifier zu erforschen. Aus der Sicht der Informationssicherheit ist man besonders interessiert an Methoden die Robustheit zu verbessern und vorhandene Methoden deutlich anwendbarer zu machen.
Um Adversarial Defenses in reellen Situationen anzuwenden, ist es wichtig die Nebenwirkungen, insbesondere auf die generelle Genauigkeit der gutmütige Inputs, zu verbessern, dafür gibt es zwar inzwischen gute Ansätze bei Adversarial Learning. In anderen Methoden, insbesondere zur Certified Robustness, sind diese Nebenwirkungen momentan noch zu hoch um eine reelle Anwendungsmöglichkeit zu sehen.
Desweiteren ist es wichtig eine bessere Möglichkeit zu finden die Messung der Perturbation an die semantische Änderung in dem Input, sowie es Menschen wahrnehmen, anzupassen. *Gilmer et al.*[3] illustrieren hierzu eindeutig, dass die Messung innerhalb einer l_p-Norm nicht ausreicht um geeignete Aussagen über Robustheit und Tarnung des Adversarial Input zu treffen.
Die zum jetzigem Zeitpunkt aussichtsreichsten Fragen für beide Forschungsbereiche beschäftigen sich mit der grundlegenden Existenz von Adversarial Input und dem Zusammenhang zu nicht robusten Features bzw. Mustern im Trainingsset wie durch *Ilyas et al.*[4] beschrieben. Dabei wird versucht die Anfälligkeit für Adversarial Learning bei der Auswahl des Trainingsset zu reduzieren und somit die Kondition des Problems zu verbessern.

Literatur

[1] Nicholas Carlini. On evaluating adversarial robustness. In *Camlis Keynote: On Evaluating Adversarial Robustness*. Camlis Organisation, oct 2019. URL `https://www.youtube.com/watch?v=-p2il-V-0fk`.

[2] Nilesh Dalvi, Pedro Domingos, Mausam, Sumit Sanghai, and Deepak Verma. Adversarial classification. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, page 99–108, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 1581138881. doi: 10.1145/1014052.1014066. URL `https://doi.org/10.1145/1014052.1014066`.

[3] Justin Gilmer, Ryan P. Adams, Ian Goodfellow, David Andersen, and George E. Dahl. Motivating the rules of the game for adversarial example research, 2018.

[4] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features, 2019.

[5] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world, 2017.

[6] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 656–672, 2019. doi: 10.1109/SP.2019.00044.

[7] Florian Tramèr, Jens Behrmann, Nicholas Carlini, Nicolas Papernot, and Jörn-Henrik Jacobsen. Fundamental tradeoffs between invariance and sensitivity to adversarial perturbations, 2020.

[8] X. Yuan, P. He, Q. Zhu, and X. Li. Adversarial examples: Attacks and defenses for deep learning. *IEEE Transactions on Neural Networks and Learning Systems*, 30(9):2805–2824, 2019. doi: 10.1109/TNNLS.2018.2886017.