Vortragsnotizen: Überblick über die Absicherung von DNNs im autonomen Fahren

DNN-Absicherung

Ein Überblick

Gesina Schwalbe

20. Dezember 2018

Inhaltsverzeichnis

Zusammenfassung

Viele herkömmlichen Methoden zur Sicherstellung der Betriebssicherheit sind auf autonome Fahrzeuge mit DNNs nicht (direkt) anwendbar. Der Vortrag zu diesen Notizen ist ein Schnelldurchlauf durch die wichtigsten Fragestellungen und Unterschiede, den aktuellen Stand und mögliche Ausblicke zur Absicherung von DNNs. Das Ergebnis ist, dass in allen Phasen, d.h. für Entwicklungsstandards, Korrekturmethoden, Absicherung während des Betriebs, sowie problemspezifische Analyse- und Nachweismethoden, Handlungsbedarf herrscht. Dabei stellen die Analyse und die Betriebsabsicherung bisher quasi ungelöste Nadelöhre dar.

1 Herkömmlicher Sicherheitsnachweis

1.1 Bestandteile

Ein Sicherheitsnachweis eines Kotrollsystems kann/sollte enthalten:

Gefahren- und Risikoanalyse zur Bestimmung von Sicherheitsanforderungen

- Was für Gefahren können auftreten?
- Wie wahrscheinlich und wie schwerwiegend sind diese?
- Welche kombinierten (Fehl-)Funktionen können dazu führen?
- Wer/welcher Systemteil ist für die Verhinderung/Milderung zuständig?

Nachweise der identifizierten Sicherheitsanforderungen bzw. Einstufung, wie sicher es ist durch z.B.

- Formale Methoden
- Tests

Präventions-/Milderungsmaßnahmen für Entwicklung & Betrieb

1.2 Nachteile

Die konventionellen Methoden haben u.a. folgende Nachteile für die Anwendung auf AI-Systeme:

- Die Risikolevel sind *hardwareorientiert*, z.B. wird eine gleichverteilte Ausfallwahrscheinlichkeit angenommen
- Es wird ein *White-Box*-System benötigt, um mögliche Ausfälle bis auf überprüfbare Einzelkomponenten und deren (prüfbares) Zusammenspiel zurückverfolgen zu können.

1.3 Typische Methoden

- Gefahren-/Risikoanalyse (Sind gefährliche Ein-Ausgabe-Kobinationen möglich?)

  Deduktiv (Rückverfolgung, d.h. starte mit Ausgabewert) [17]

  Induktiv (Eingaberaumsuche, d.h. starte mit Eingabewerten) [17]

  Vollständiges Testen

  Bewährtheit im Betrieb

- Prävention/Milderung

  Überwachung

  Backupsysteme

  Saubere Entwicklung

  Safety-Life-Cycle

1.4 Unterschiede von ML-Systemen

Komplexität des Problems und Black-Box Natur

- Wenig Verständis von Randbereichen des Algorithmus
  - Wann befinden sich die Ein- oder Ausgaben außerhalb des (sicheren) Betriebsbereichs? Wie sieht effektive Überwachung aus?
  - Keine Bewährung durch langen Betriebseinsatz
- Große Ein- und Zustandsbereiche, dazu wenig Wissen über den Entscheidungsprozess (den Kontrollfluss)
  - Keine deduktiven Methoden
  - Keine induktiven Methoden
  - Kein vollständiges Testen
- Komplexität des Algorithmus
  - Wie erkennt man da noch Implementierungsfehler?

Keine Qualitätskriterien , die allg. anerkannt sind, u.a.

- Metriken und entsprechende Schwellenwerte
- Systematische Maßnahmen zur Erkennung gängier Fehler
- Stand der Technik (d.h. Faustregeln) für Entwicklung und Betrieb; vollständiger Lebenszyklus (d.h. Plan für Zwischenschritte mit Qualitätsmanagement)

Neue Komponenten kommen zum Einsatz, z.B.

- Bibliotheken
- Spezialisierte Hardware, z.B. Neuromorphic Chips

2 Offene Fragen für das Fehlermanagement

Für einen vollständigen Sicherheitsnachweis müssen in jeder Phase des Lebenszyklus bestmögliche Maßnahmen ergriffen werden, um (gefährliche) Fehler zu vermeiden.

2.1 Scope des Vortrags

Beim automatisierten Fahren wird in Wahrnehmungsaufgaben (Computer Vision) und Planungs- bzw. Entscheidungsaufgaben unterschieden [7]. Der Bereich der Planungs- und Entscheidungsaufgaben ist bereits gut durch Überwachung abgedeckt:

- Einfaches Beispiel: [7]
- Vollständiges Verhaltensregelwerk: [31]
- Existenzbeweis: [21]

Abbildung 1: Typischer Teil der Wahrnehmungsaufgaben: Segmentierung der Kameraaufnahmen

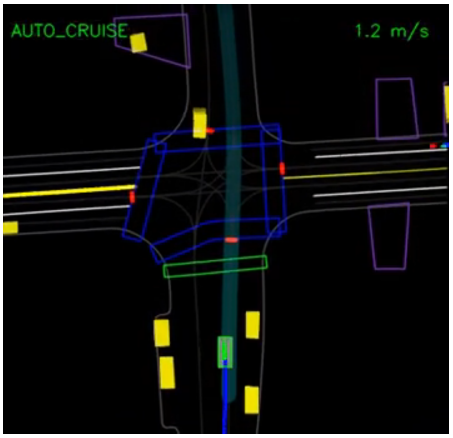

Abbildung 2: Typischer Input für Planungs-/Entscheidungsaufgaben wie z.B. Trajektorienplanung

- Regelerlernen und Überwachung für Reinforcement Learning: [12, 2, 8]

Daher konzentriert sich die folgende Analyse auf Wahrnehmungsaufgaben.

Die übergeordneten Phasen sind Entwicklung, Prüfung, Korrektur, Betrieb. Entwicklung, Prüfung und Korrektur bilden einen Zyklus. Die Prüfung kann unterteilt werden in

- Risikoprüfung bzw. Formulierung von Sicherheitskriterien und
- Prüfen dieser.

2.2 Wie erkennt/verhindert man Fehler frühzeitig?

Die aktuellen ISO Normen zum Thema Absicherung geben nicht nur Empfehlungen zum Überprüfungsprozess, sondern generell eine Sammlung (damaliger) Prozessstandards wider. Das schließt für Software z.B. Unittests mit Abdeckungsmetriken, sinnvolle Modularisierung und Variablenbenennung mit ein. Für neuartige Entwicklungsbestandteile von AI fehlen allerdings solche Faustregeln.

Lern-Daten ***(Datenrepräsentativität)*** Betonung von Gegenbeispielen und relevanten Szenarien

Test-Daten ***(Testabdeckung)*** sicherheitsrelevante Testabdeckung, z.B. Concolic Testing [34]

Robustheit ***(Robustheitsgrad)*** Dropout, Regularisierung, Robust Manifold Ansatz, Adversarial Learning, …

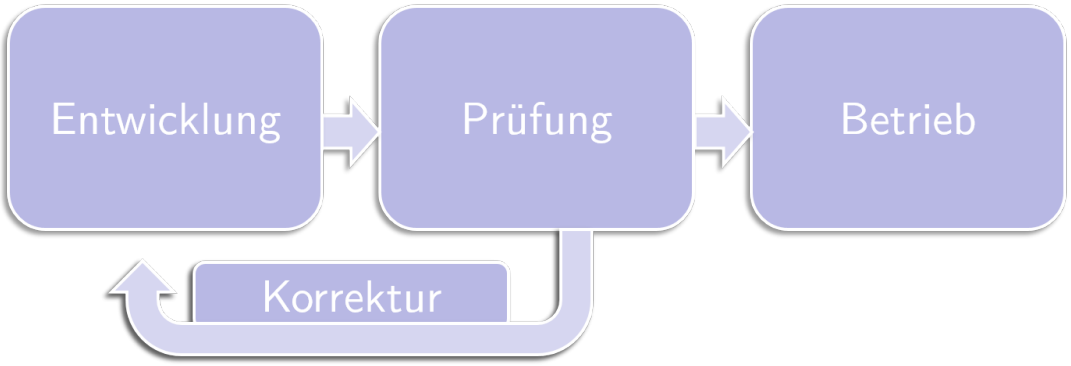

Abbildung 3: Phasen des Lebenszyklus eines Systems, die relevant für das Fehlermanagement sind.

Plausibilisierung Funktionalitätsprüfung für designspezifische Probleme, z.B. Reward-Hacking

Expertenwissen ***(Regelbefolgung)*** einfügen

Design Faustregeln zu Loss-/Reward-Funktion, Layern, Aktivierungen, Ungewissheitsangaben …

Es gilt also, die bisherigen Entwicklungsstandards, wenn nötig, auszubauen, brauchbare Metriken zu formulieren und einem Standard hinzuzufügen.

2.3 Wie findet man gefährliche Fehler(ursachen)?

Zur Analyse auf problemspezifische Fehler hin ist ein *White-Box*-System nötig! Hoffnung:

Eingrenzung der Ein-/Zustandsbereiche

Vereinfachung des Problems durch Abstrahierung

Leider gibt es bisher keinen vollständigen anwendbaren Ansatz, mithilfe dessen genügend aussagekräftige, konkrete, durch Regelprüfer nachweisbare Sicherheitsanforderungen formuliert werden könnten. Das Problem hierbei ist einerseits die Einsicht in den Algorithmus, andererseits die Übersetzung von qualitativen/in natürlicher Sprache formulierten Anforderungen in Bedingungen auf Eingabe-/Ausgabe-/Zwischenwerten.

2.3.1 Qualitative Einsicht

Heatmapping z.B.

- LIME [27], RISE [24], Instancewise Feature Selection [4], …
- LRP [3], Attention Visualization [20], …

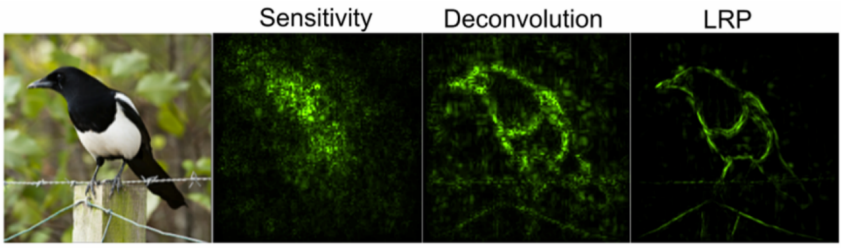

Abbildung 4: Vergleich verschiedener Heatmapping-Methoden

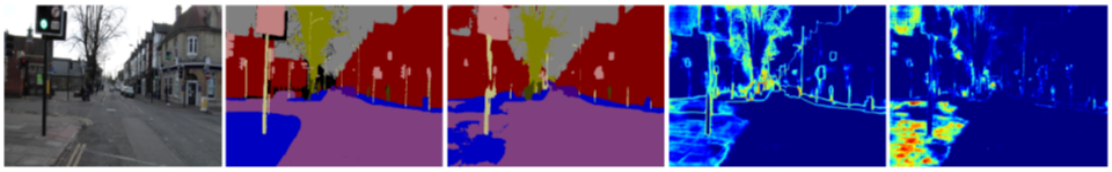

Abbildung 5: Ausgabe und Visualisierung von Unsicherheit mit erlerntem Weight Decay [19]. Die falsche Segmentierung des Gehwegs (c) sowie Kanten von Segmentierungsobjekten werden als stark ungewisse Zonen in der Ausgabe markiert (e).

Unsicherheitsangaben z.B. durch

- spezielle Backpropagation [11]
- erlernten Weight Decay [19]

Aufgabenlokalisierung z.B. durch Neural Stethoscopes [9]

Netzwerkvereinfachung z.B. CAR-Compression [1] (ist zumindest hilfreich)

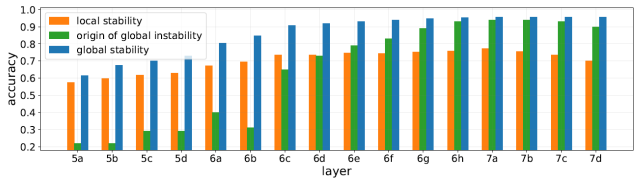

Abbildung 6: Anwendung von Neural Stethoscopes, um bei einem Stabilitätsvorhersageproblem die Layerzuständigkeit für einzelne Unteraufgaben zu bestimmen. Je höher die Accuracy für ein Layer, desto besser ist der Abstraktionsgrad dieses Layers für diese Unteraufgabe geeignet.

2.3.2 Regelextraktion

Hauptkriterien *Verständlichkeit* bei *Originaltreue*

Extrahierte Regeln Meist *IF-THEN* oder *M-von-N*

Kategorien:

Black-Box oder pädagogisch:

Verhaltensnachbildung durch Regeln, z.B. lokal mit ILP [26]

White-/Grey-Box oder decompositional/eklektisch:

Umwandlung in Entscheidungsbaum, z.B. DeepRED [30]

Beide Richtungen haben Vor- und Nachteile. Pädagogische Methoden erzeugen häufig genauere Modelle der ursprüngliche Funktion, sind dafür aber meist weniger effizient [10].

2.4 Wie weist man Fehlerfreiheit nach?

Die Alternativen sind:

- Manuell
- Automatisch

  Entscheidungsproblemlöser z.B. Reluplex [18] (SMT), [16] (SMT), SHERLOCK [6] (MILP)

  Randannäherung z.B. DeepGo [29], ReluVal [33], NeVer [25]

  Suchalgorithmen z.B. VeriVis [23]

Damit existieren machbare Ansätze für Regelprüfung, welche allerdings entweder schlecht skalieren (z.B. Reluplex) oder bezüglich der prüfbaren Regeln sehr eingeschränkt sind (z.B. nur Adversarial Examples).

2.5 Wie korrigiert man Fehler?

Anlernen Modifiziere Loss- oder Reward-Funktion, z.B. [28], Teacher Network [15], Model Repair [12]

Anpassen der Topologie z.B. KBANN [22] oder ReNN [32] Bei KBANN wird ein kleines Netze aus einem Entscheidungsbaum konstruiert, während bei ReNN ein Netz zerlegt wird in Abstraktion, Regeln auf abstrakten Features, Entscheidung.

Neue Daten Lernen mit Gegenbeispielen, z.B. erstellt von Regelprüfern oder mit [5, 34, 13, 14])

Passend auf Eingabe-/Ausgabe-/Zwischenwerten formulierte Anforderungen können demnach in die Netzwerkfunktionalität selbst eingebaut werden. Die verfügbaren Methoden geben jedoch keine Garantie über den Erfolg des Einfügens, welcher mit einem Regelprüfer evaluiert werden muss.

2.6 Wie mildert/umgeht man bestehende funktionale Fehler?

Überwachung d.h. Plausibilisierung für

- Eingaben (z.B. Sensorikredundanz & -abgleich)
- Ausgaben (z.B. Bereichs-, Ungewissheitsüberwachung)

Redundanz

Backup-/Fallback-System(e) *fail operational* vs. bisher *fail safe* Beim automatisierten Fahren muss das Auto betriebsfähig bleiben, selbst wenn kleinere Fehler auftreten (also keine Vollbremsung auf der Autobahn wegen eines Dreckflecks)! Bisher ist dieser Teil ungelöst.

2.7 Fazit

Handlungsbedarf besteht noch in jeder Phase. Nach aktuellem Stand absteigend sortiert:

Entwicklung Hier gibt es einige gute qualitative Ansätze, die bereits weithin verbreitet sind. Es fehlt „nur“ die Standardisierung sowie eine Quantisierung durch Metriken.

Korrektur Hier gibt es eine große Lücke bei Ansätzen, die eine Garantie über den Erfolg des Regeleinfügens geben. Das kann allerdings durch die vielfältigen qualitativen Ansätze in Kombination mit passenden Regelprüfern ausgeglichen werden.

Betrieb Es gibt Ideen zur Überwachung und zur Redundanz, allerdings weniger für Computer Vision. Die Frage nach einem Fallback-System ist ungeklärt.

Prüfung Das Nadelöhr hier ist die Risikoanalyse und Formulierung prüfbarer Sicherheitskriterien. Hier sind hauptsächlich qualitative Ansätze verfügbar, und alle weiteren sind bisher nicht für die Verwendung in der Sicherheitsprüfung evaluiert. Für das Überprüfen von Sicherheitskriterien sind reichlich, wenn auch vermutlich noch nicht genügend performante, Methoden verfügbar.

Literatur

[1] Reza Abbasi-Asl und Bin Yu. „Interpreting Convolutional Neural Networks Through Compression“. In: *arXiv:1711.02329 [cs, stat]* (7. Nov. 2017). arXiv: 1711.02329. URL: http://arxiv.org/abs/1711.02329.

[2] A. K. Akametalu u. a. „Reachability-based safe learning with Gaussian processes“. In: *53rd IEEE Conference on Decision and Control*. 53rd IEEE Conference on Decision and Control. Dez. 2014, S. 1424–1431.

[3] Sebastian Bach u. a. „On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation“. In: *PLOS ONE* 10.7 (10. Juli 2015), e0130140. ISSN: 1932-6203. URL: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0130140.

[4] Jianbo Chen u. a. „Learning to Explain: An Information-Theoretic Perspective on Model Interpretation“. In: *arXiv:1802.07814 [cs, stat]* (21. Feb. 2018). arXiv: 1802.07814. URL: http://arxiv.org/abs/1802.07814.

[5] Tommaso Dreossi u. a. „Counterexample-Guided Data Augmentation“. In: (17. Mai 2018). URL: https://arxiv.org/abs/1805.06962.

[6] Souradeep Dutta u. a. „Output Range Analysis for Deep Neural Networks“. In: *arXiv:1709.09130 [cs, stat]* (26. Sep. 2017). arXiv: 1709.09130. URL: http://arxiv.org/abs/1709.09130.

[7] Lucas E. R. Fernandes u. a. „A Rational Agent Controlling an Autonomous Vehicle: Implementation and Formal Verification“. In: *Electronic Proceedings in Theoretical Computer Science* 257 (7. Sep. 2017), S. 35–42. ISSN: 2075-2180. arXiv: 1709.02557. URL: http://arxiv.org/abs/1709.02557.

[8] David Fridovich-Keil u. a. „Planning, Fast and Slow: A Framework for Adaptive Real-Time Safe Trajectory Planning“. In: *arXiv:1710.04731 [cs]* (12. Okt. 2017). arXiv: 1710.04731. URL: http://arxiv.org/abs/1710.04731.

[9] Fabian B. Fuchs u. a. „Neural Stethoscopes: Unifying Analytic, Auxiliary and Adversarial Network Probing“. In: *arXiv:1806.05502 [cs, stat]* (14. Juni 2018). arXiv: 1806.05502. URL: http://arxiv.org/abs/1806.05502.

[10] A. S d'Avila Garcez, K Broda und D. M Gabbay. „Symbolic knowledge extraction from trained neural networks: A sound approach“. In: *Artificial Intelligence* 125.1 (1. Jan. 2001), S. 155–207. ISSN: 0004-3702. URL: http://www.sciencedirect.com/science/article/pii/S0004370200000771.

[11] Jochen Gast und Stefan Roth. „Lightweight Probabilistic Deep Networks“. In: *arXiv:1805.11327 [cs, stat]* (29. Mai 2018). arXiv: 1805.11327. URL: http://arxiv.org/abs/1805.11327.

[12] Shalini Ghosh, Patrick Lincoln und Ashish Tiwari. „Trusted Machine Learning for Probabilistic Models“. In: 2016.

[13] Jianmin Guo u. a. „DLFuzz: Differential Fuzzing Testing of Deep Learning Systems“. In: *arXiv:1808.09413 [cs]* (28. Aug. 2018). arXiv: 1808.09413. URL: http://arxiv.org/abs/1808.09413.

[14] Warren He, Bo Li und Dawn Song. „Decision Boundary Analysis of Adversarial Examples“. In: (15. Feb. 2018). URL: https://openreview.net/forum?id=BkpiPMbA-.

[15] Zhiting Hu u. a. „Harnessing Deep Neural Networks with Logic Rules“. In: *arXiv:1603.06318 [cs, stat]* (20. März 2016). arXiv: 1603.06318. URL: http://arxiv.org/abs/1603.06318.

[16] Xiaowei Huang u. a. „Safety Verification of Deep Neural Networks“. In: *arXiv:1610.06940 [cs, stat]* (21. Okt. 2016). arXiv: 1610.06940. URL: http://arxiv.org/abs/1610.06940.

[17] Chris W. Johnson. „The Increasing Risks of Risk Assessment : On the Rise of Artificial Intelligence and NonDeterminism in Safety-Critical Systems“. In: 2017.

[18] Guy Katz u. a. „Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks“. In: *arXiv:1702.01135 [cs]* (3. Feb. 2017). arXiv: 1702.01135. URL: http://arxiv.org/abs/1702.01135.

[19] Alex Kendall und Yarin Gal. „What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?“ In: *arXiv:1703.04977 [cs]* (15. März 2017). arXiv: 1703.04977. URL: http://arxiv.org/abs/1703.04977.

[20] Jinkyu Kim und John Canny. „Interpretable Learning for Self-Driving Cars by Visualizing Causal Attention“. In: *arXiv:1703.10631 [cs]* (30. März 2017). arXiv: 1703.10631. URL: http://arxiv.org/abs/1703.10631.

[21] John Lygeros, Claire J. Tomlin und S. Shankar Sastry. „Controllers for reachability specifications for hybrid systems“. In: *Automatica* 35 (1999), S. 349–370.

[22] David Opitz und Jude W. Shavlik. „Heuristically Expanding Knowledge-Based Neural Networks“. In: *In Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, 1993, S. 1360–1365.

[23] Kexin Pei u. a. „Towards Practical Verification of Machine Learning: The Case of Computer Vision Systems“. In: *arXiv:1712.01785 [cs]* (5. Dez. 2017). arXiv: 1712.01785. URL: http://arxiv.org/abs/1712.01785.

[24] Vitali Petsiuk, Abir Das und Kate Saenko. „RISE: Randomized Input Sampling for Explanation of Black-box Models“. In: (19. Juni 2018). URL: https://arxiv.org/abs/1806.07421.

[25] Luca Pulina und Armando Tacchella. „An Abstraction-Refinement Approach to Verification of Artificial Neural Networks“. In: CEUR Workshop Proceedings. Bd. 616. 15. Juli 2010, S. 243–257.

[26] Johannes Rabold, Michael Siebers und Ute Schmid. „Explaining Black-Box Classifiers with ILP – Empowering LIME with Aleph to Approximate Non-linear Decisions with Relational Rules“. In: *Inductive Logic Programming*. Hrsg. von Fabrizio Riguzzi, Elena Bellodi und Riccardo Zese. Lecture Notes in Computer Science. Springer International Publishing, 2018, S. 105–117. ISBN: 978-3-319-99960-9.

[27] Marco Tulio Ribeiro, Sameer Singh und Carlos Guestrin. „"Why Should I Trust You?": Explaining the Predictions of Any Classifier“. In: *arXiv:1602.04938 [cs, stat]* (16. Feb. 2016). arXiv: 1602.04938. URL: http://arxiv.org/abs/1602.04938.

[28] Soumali Roychowdhury, Michelangelo Diligenti und Marco Gori. „Image Classification Using Deep Learning and Prior Knowledge“. In: *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*. Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence. 20. Juni 2018. URL: https://aaai.org/ocs/index.php/WS/AAAIW18/paper/view/16575.

[29] Wenjie Ruan, Xiaowei Huang und Marta Kwiatkowska. „Reachability Analysis of Deep Neural Networks with Provable Guarantees“. In: (6. Mai 2018). URL: https://arxiv.org/abs/1805.02242.

[30] Jan Ruben Zilke, Eneldo Loza Mencía und Frederik Janssen. „DeepRED – Rule Extraction from Deep Neural Networks“. In: 19. Okt. 2016, S. 457–473. ISBN: 978-3-319-46306-3.

[31] Shai Shalev-Shwartz, Shaked Shammah und Amnon Shashua. „On a Formal Model of Safe and Scalable Self-driving Cars“. In: *arXiv:1708.06374 [cs, stat]* (21. Aug. 2017). arXiv: 1708.06374. URL: http://arxiv.org/abs/1708.06374.

[32] Hu Wang. „ReNN: Rule-embedded Neural Networks“. In: *arXiv:1801.09856 [cs, stat]* (30. Jan. 2018). arXiv: 1801.09856. URL: http://arxiv.org/abs/1801.09856.

[33] Shiqi Wang u. a. „Formal Security Analysis of Neural Networks using Symbolic Intervals“. In: (28. Apr. 2018). URL: https://arxiv.org/abs/1804.10829.

[34] Sun Youcheng u. a. „Concolic Testing for Deep Neural Networks“. In: *arXiv:1805.00089 [cs, stat]*. abs/1805.00089 (2018). URL: http://arxiv.org/abs/1805.00089.