

# DNN-Absicherung

## Ein Überblick

Gesina Schwalbe

20. Dezember 2018

- 1 Herkömmlicher Sicherheitsnachweis
- 2 Offene Fragen für das Fehlermanagement

## Abschnitt 1

### Herkömmlicher Sicherheitsnachweis

#### 1 Herkömmlicher Sicherheitsnachweis

- Bestandteile
- Typische Methoden

# Bestandteile

Ein Sicherheitsnachweis eines Kotrollsystems kann/sollte enthalten:

Gefahren- und Risikoanalyse

Nachweise der identifizierten Sicherheitsanforderungen

Präventions-/Milderungsmaßnahmen für Entwicklung & Betrieb

# Bestandteile

Ein Sicherheitsnachweis eines Kotrollsystems kann/sollte enthalten:

Gefahren- und Risikoanalyse

Nachweise der identifizierten Sicherheitsanforderungen

Präventions-/Milderungsmaßnahmen für Entwicklung & Betrieb

# Bestandteile

Ein Sicherheitsnachweis eines Kotrollsystems kann/sollte enthalten:

Gefahren- und Risikoanalyse

Nachweise der identifizierten Sicherheitsanforderungen

Präventions-/Milderungsmaßnahmen für Entwicklung & Betrieb

# Typische Methoden

- Gefahren-/Risikoanalyse

**Deduktiv** (Rückverfolgung, d.h. starte mit Ausgabewert)

**Induktiv** (Eingaberaumsuche, d.h. starte mit Eingabewerten)

**Vollständiges Testen**

**Bewährtheit im Betrieb**

- Prävention/Milderung

Überwachung

Backupsysteme

Saubere Entwicklung

Safety-Life-Cycle

# Typische Methoden

- Gefahren-/Risikoanalyse
  - Deduktiv (Rückverfolgung, d.h. starte mit Ausgabewert)
  - Induktiv (Eingaberaumsuche, d.h. starte mit Eingabewerten)
  - Vollständiges Testen
  - Bewährtheit im Betrieb
- Prävention/Milderung
  - Überwachung
  - Backupsysteme
  - Saubere Entwicklung
  - Safety-Life-Cycle



## Abschnitt 2

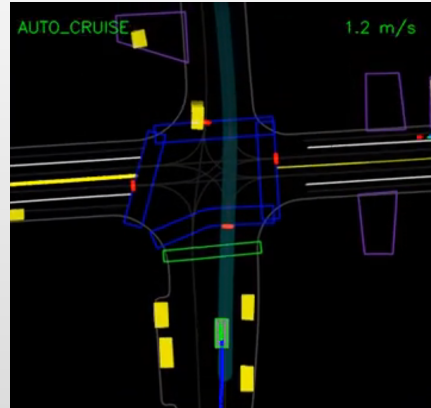
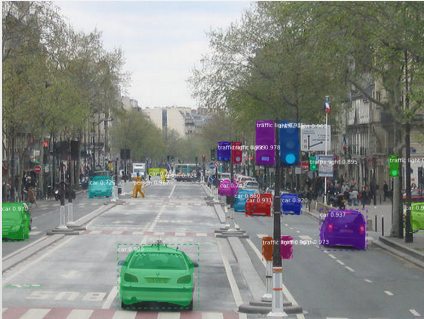
### Offene Fragen für das Fehlermanagement

#### 2 Offene Fragen für das Fehlermanagement

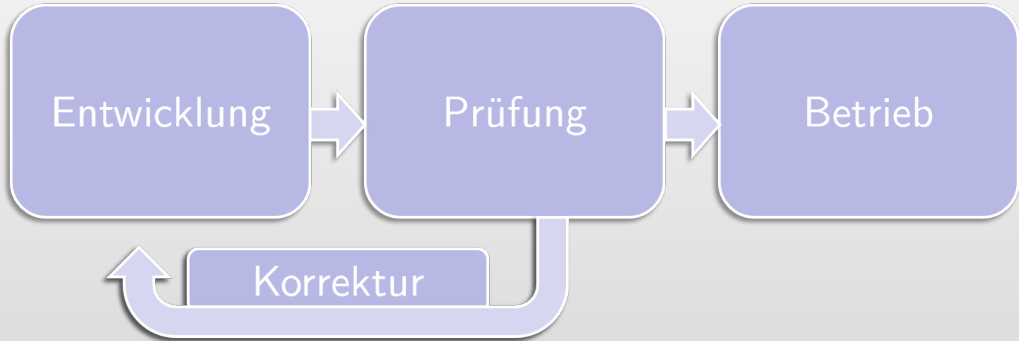
- Wie erkennt/verhindert man Fehler frühzeitig?
- Wie findet man gefährliche Fehler(ursachen)?
- Wie weist man Fehlerfreiheit nach?
- Wie korrigiert man Fehler?
- Wie mildert/umgeht man bestehende funktionale Fehler?
- Fazit

Herkömmlicher Sicherheitsnachweis  
**Offene Fragen**  
Literatur

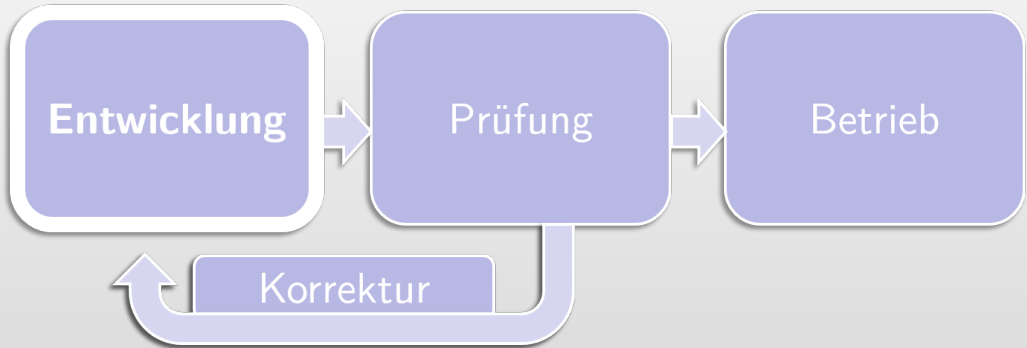
Wie erkennt/verhindert man Fehler frühzeitig?  
Wie findet man gefährliche Fehler(ursachen)?  
Wie weist man Fehlerfreiheit nach?  
Wie korrigiert man Fehler?  
Wie mildert/umgeht man bestehende funktionale Fehler?  
Fazit



## Phasen im Fehlermanagement



## Wie erkennt/verhindert man Fehler frühzeitig?



# Ansätze für Faustregeln und Metriken

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

Test-Daten (*Testabdeckung*) sicherheitsrelevante Testabdeckung, z.B. Concolic Testing [27]

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

Planung/Planer Funktionalitätsprüfung für designspezifische Probleme, z.B. Reward-Hacking

Expertenwissen (*Regelbefolgung*) einfügen

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

# Ansätze für Faustregeln und Metriken

- Lern-Daten (*Datenrepräsentativität*) Betonung von Gegenbeispielen und relevanten Szenarien
- Test-Daten (*Testabdeckung*) sicherheitsrelevante Testabdeckung, z.B. Concolic Testing [27]
- Robustheit (*Robustheitsgrad*) Dropout, Regularisierung, Robust Manifold Ansatz, Adversarial Learning, ...
- Plausibilisierung Funktionalitätsprüfung für designspezifische Probleme, z.B. Reward-Hacking
- Expertenwissen (*Regelbefolgung*)
  - Design Faustregeln zu Loss-/Reward-Funktion, Layern, Aktivierungen, Ungewissheitsangaben ...

# Ansätze für Faustregeln und Metriken

- Lern-Daten (*Datenrepräsentativität*) Betonung von Gegenbeispielen und relevanten Szenarien
- Test-Daten (*Testabdeckung*) sicherheitsrelevante Testabdeckung, z.B. Concolic Testing [27]
- Robustheit (*Robustheitsgrad*) Dropout, Regularisierung, Robust Manifold Ansatz, Adversarial Learning, ...
- Plausibilisierung Funktionalitätsprüfung für designspezifische Probleme, z.B. Reward-Hacking
- Expertenwissen (*Regelbefolgung*) einfügen
- Design
- Aktivierung/Ungewissheitsangaben

# Ansätze für Faustregeln und Metriken

- Lern-Daten (*Datenrepräsentativität*) Betonung von Gegenbeispielen und relevanten Szenarien
- Test-Daten (*Testabdeckung*) sicherheitsrelevante Testabdeckung, z.B. Concolic Testing [27]
- Robustheit (*Robustheitsgrad*) Dropout, Regularisierung, Robust Manifold Ansatz, Adversarial Learning, ...
- Plausibilisierung Funktionalitätsprüfung für designspezifische Probleme, z.B. Reward-Hacking
- Expertenwissen (*Regelbefolgung*) einfügen
- Design Faustregeln zu Loss-/Reward-Funktion, Layern, Aktivierungen, Ungewissheitsangaben ...



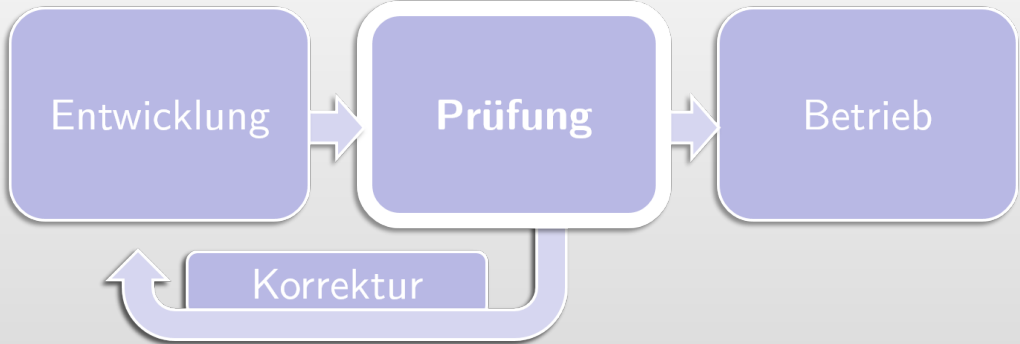
## Ansätze für Faustregeln und Metriken

- Lern-Daten (*Datenrepräsentativität*) Betonung von Gegenbeispielen und relevanten Szenarien
- Test-Daten (*Testabdeckung*) sicherheitsrelevante Testabdeckung, z.B. Concolic Testing [27]
- Robustheit (*Robustheitsgrad*) Dropout, Regularisierung, Robust Manifold Ansatz, Adversarial Learning, ...
- Plausibilisierung Funktionalitätsprüfung für designspezifische Probleme, z.B. Reward-Hacking
- Expertenwissen (*Regelbefolgung*) einfügen
- Design Faustregeln zu Loss-/Reward-Funktion, Layern, Aktivierungen, Ungewissheitsangaben ...

# Ansätze für Faustregeln und Metriken

- Lern-Daten (*Datenrepräsentativität*) Betonung von Gegenbeispielen und relevanten Szenarien
- Test-Daten (*Testabdeckung*) sicherheitsrelevante Testabdeckung, z.B. Concolic Testing [27]
- Robustheit (*Robustheitsgrad*) Dropout, Regularisierung, Robust Manifold Ansatz, Adversarial Learning, ...
- Plausibilisierung Funktionalitätsprüfung für designspezifische Probleme, z.B. Reward-Hacking
- Expertenwissen (*Regelbefolgung*) einfügen
- Design Faustregeln zu Loss-/Reward-Funktion, Layern, Aktivierungen, Ungewissheitsangaben ...

## Wie findet man gefährliche Fehler(ursachen)?



# Wie findet man gefährliche Fehler(ursachen)?

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

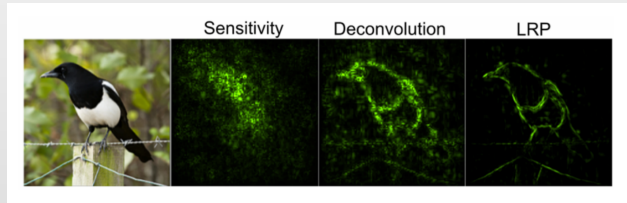
**Eingrenzung** der Ein-/Zustandsbereiche

**Vereinfachung** des Problems durch Abstrahierung

# Qualitative Einsicht

Heatmapping z.B.

- LIME [21], RISE [18], Instancewise Feature Selection [3], ...
- LRP [2], Attention Visualization [15], ...



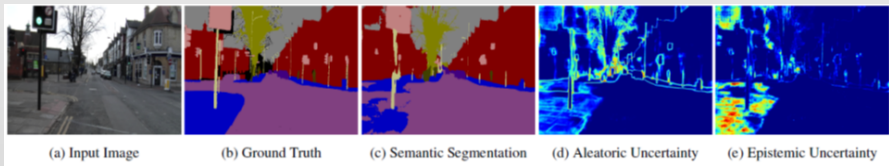
# Qualitative Einsicht

## Heatmapping z.B.

- LIME [21], RISE [18], Instancewise Feature Selection [3], ...
- LRP [2], Attention Visualization [15], ...

## Unsicherheitsangaben z.B. durch

- spezielle Backpropagation [7]
- erlernten Weight Decay [14]



# Qualitative Einsicht

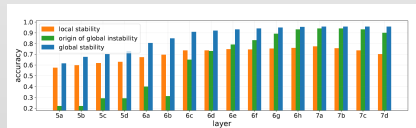
## Heatmapping z.B.

- LIME [21], RISE [18], Instancewise Feature Selection [3], ...
- LRP [2], Attention Visualization [15], ...

## Unsicherheitsangaben z.B. durch

- spezielle Backpropagation [7]
- erlernten Weight Decay [14]

## Aufgabenlokalisierung z.B. durch Neural Stethoscopes [6]



# Qualitative Einsicht

Heatmapping z.B.

- LIME [21], RISE [18], Instancewise Feature Selection [3], ...
- LRP [2], Attention Visualization [15], ...

Unsicherheitsangaben z.B. durch

- spezielle Backpropagation [7]
- erlernten Weight Decay [14]

Aufgabenlokalisierung z.B. durch Neural Stethoscopes [6]

Netzwerkvereinfachung z.B. CAR-Compression [1]



# Regelextraktion

**Hauptkriterien** *Verständlichkeit* bei *Originaltreue*

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

Kategorien:

Black-Box Verfahrenstransformation durch Analyse  
z.B. lokal mit ILP [20]

White-/Grey-Box Umwandlung in Entscheidungsbaum,  
z.B. DeepRED [24]

# Regelextraktion

Hauptkriterien *Verständlichkeit* bei *Originaltreue*

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

Kategorien:

White-/Grey-Box Umwandlung in Entscheidungsbaum  
z.B. DeepRED [24]

# Regelextraktion

Hauptkriterien *Verständlichkeit* bei *Originaltreue*

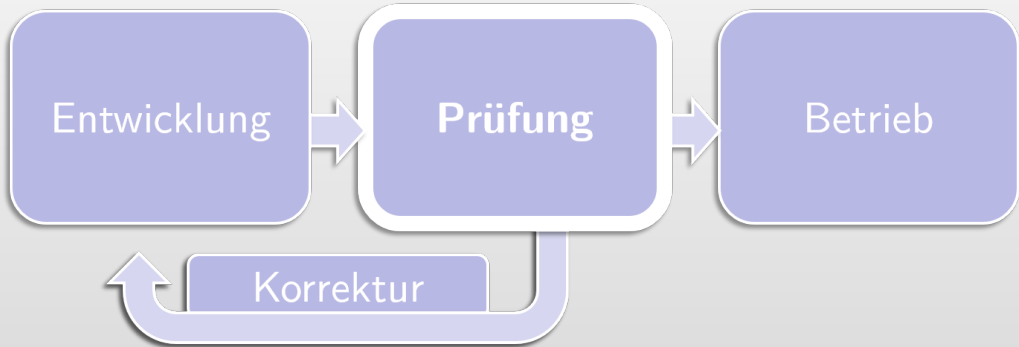
Extrahierte Regeln Meist *IF-THEN* oder *M-von-N*

Kategorien:

**Black-Box** Verhaltensnachbildung durch Regeln,  
z.B. lokal mit ILP [20]

**White-/Grey-Box** Umwandlung in Entscheidungsbaum,  
z.B. DeepRED [24]

## Wie weist man Fehlerfreiheit nach?



# Wie weist man Fehlerfreiheit nach?

- Manuell
- Automatisch

z.B. DeepGo [23], ReluVal [26], NeVer [19]

Randannäherung z.B. DeepGo [23], ReluVal [26], NeVer [19]

Suchalgorithmen z.B. VeriVis [17]

# Wie weist man Fehlerfreiheit nach?

- Manuell
- Automatisch

Entscheidungsproblemlöser z.B. Reluplex [13] (SMT), [12] (SMT),  
SHERLOCK [5] (MILP)

Randannäherung z.B. DeepGo [23], ReluVal [26], NeVer [19]

Symbolische Analyse z.B. VeriVis [17]

# Wie weist man Fehlerfreiheit nach?

- Manuell
- Automatisch

**Entscheidungsproblemlöser** z.B. Reluplex [13] (SMT), [12] (SMT),  
SHERLOCK [5] (MILP)

Randannäherung z.B. DeepGo [23], ReluVal [26], NeVer [19]

Suchalgorithmen z.B. VeriVis [17]

# Wie weist man Fehlerfreiheit nach?

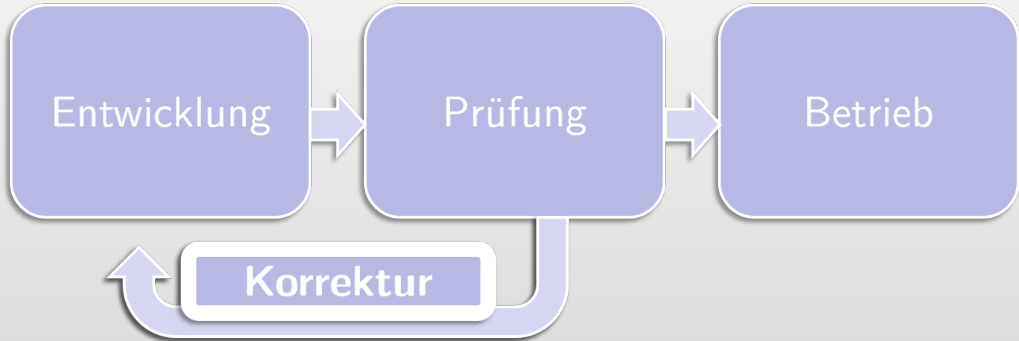
- Manuell
- Automatisch
  - Entscheidungsproblemlöser** z.B. Reluplex [13] (SMT), [12] (SMT), SHERLOCK [5] (MILP)
  - Randannäherung** z.B. DeepGo [23], ReluVal [26], NeVer [19]
  - Suchalgorithmen** z.B. VeriVis [17]



# Wie weist man Fehlerfreiheit nach?

- Manuell
- Automatisch
  - Entscheidungsproblemlöser z.B. Reluplex [13] (SMT), [12] (SMT), SHERLOCK [5] (MILP)
  - Randannäherung z.B. DeepGo [23], ReluVal [26], NeVer [19]
  - Suchalgorithmen z.B. VeriVis [17]

## Wie korrigiert man Fehler?



# Wie korrigiert man Fehler?

**Anlernen** Modifiziere Loss- oder Reward-Funktion,  
z.B. [22], Teacher Network [11], Model Repair [8]

Anpassen der Topologie z.B. KBANN [16] oder ReNN [25]

Neue Daten Lernen mit Gegenbeispielen,  
z.B. erstellt von Regelprüfern oder mit [4, 27, 9, 10])

# Wie korrigiert man Fehler?

**Anlernen** Modifiziere Loss- oder Reward-Funktion,  
z.B. [22], Teacher Network [11], Model Repair [8]

**Anpassen der Topologie** z.B. KBANN [16] oder ReNN [25]

**Neue Daten** Lernen mit Gegenbeispielen,  
z.B. erstellt von Regelprüfern oder mit [4, 27, 9, 10])

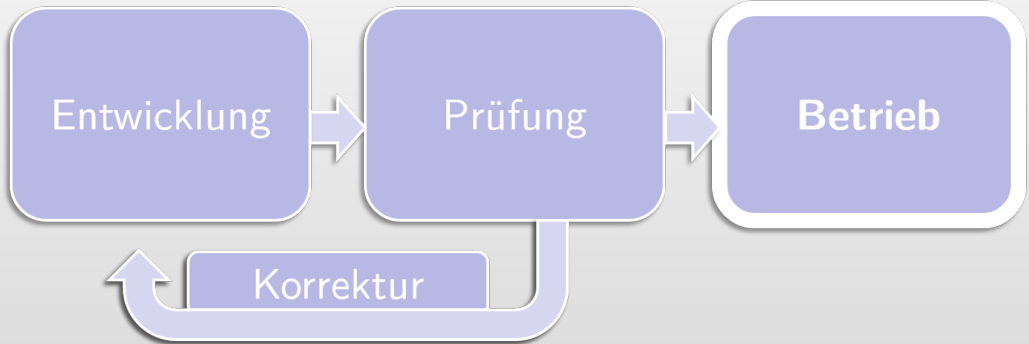
# Wie korrigiert man Fehler?

**Anlernen** Modifiziere Loss- oder Reward-Funktion,  
z.B. [22], Teacher Network [11], Model Repair [8]

**Anpassen der Topologie** z.B. KBANN [16] oder ReNN [25]

**Neue Daten** Lernen mit Gegenbeispielen,  
z.B. erstellt von Regelprüfern oder mit [4, 27, 9, 10])

## Wie mildert/umgeht man bestehende funktionale Fehler?



# 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*

# 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*



# Wie mildert/umgeht man bestehende funktionale Fehler?

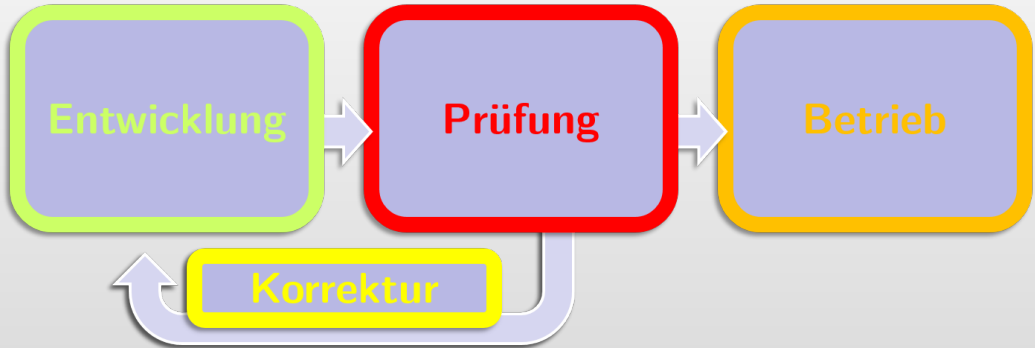
Überwachung d.h. Plausibilisierung für

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

Redundanz

Backup-/Fallback-System(e) *fail operational* vs. bisher *fail safe*

# Fazit



## Fragen, Anregungen ...

# Literaturverzeichnis I

- [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] 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>.
- [3] 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>.
- [4] Tommaso Dreossi u. a. „Counterexample-Guided Data Augmentation“. In: (17. Mai 2018). URL: <https://arxiv.org/abs/1805.06962>.
- [5] 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>.
- [6] 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>.
- [7] 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>.
- [8] Shalini Ghosh, Patrick Lincoln und Ashish Tiwari. „Trusted Machine Learning for Probabilistic Models“. In: 2016.
- [9] 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>.

## Literaturverzeichnis II

- [10] Warren He, Bo Li und Dawn Song. „Decision Boundary Analysis of Adversarial Examples“. In: (15. Feb. 2018). URL: <https://openreview.net/forum?id=BkpiPMbA->.
- [11] 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>.
- [12] 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>.
- [13] 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>.
- [14] 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>.
- [15] 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>.
- [16] 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.
- [17] 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>.

## Literaturverzeichnis III

- [18] 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>.
- [19] 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.
- [20] 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.
- [21] 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>.
- [22] 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>.
- [23] 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>.
- [24] 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.

## Literaturverzeichnis IV

- [25] 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>.
- [26] Shiqi Wang u. a. „Formal Security Analysis of Neural Networks using Symbolic Intervals“. In: (28. Apr. 2018). URL: <https://arxiv.org/abs/1804.10829>.
- [27] 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>.