Test der Coverage von neuronalen Netzen um Fehler im Netzverhalten zu erkennen

Seminar: Intelligente cyber-physische Systeme

Baum, Sebastian
Universität Stuttgart
Stuttgart, Deutschland
st170415@stud.uni-stuttgart.de

***Abstract*—Autonome Systeme werden immer wichtiger. Eine der Technologien, die autonome Systeme realisieren, sind neuronale Netze. Aufgrund des vermehrten Einsatzes in sicherheitskritischen Anwendungen wie dem autonomen Fahren, ist einer der momentanen Forschungsbereiche, das Fehlverhalten neuronaler Netze zu minimieren. Um fehlerhaftes Verhalten in neuronalen Netzen zu beheben, gilt es, Daten zu erzeugen, die ein fehlerhaftes Verhalten verursachen. Die Aufgabe besteht darin Daten zu generieren, die ein Fehlverhalten in neuronalen Netzen auslösen. Der Algorithmus aus der Veröffentlichung Deeptest löst diese Aufgabe mithilfe von synthetischer Bildgenerierung, die mit Hilfe der Coverage-Metrik Neuron Coverage geführt ist. Der Algorithmus synthetisiert die Daten nicht willkürlich, sondern erzeugt mithilfe von Transformationen weiterhin reale Daten, die in der Realität vorkommen. Innerhalb dieser Arbeit ist die Veröffentlichung Deeptest evaluiert. Neben der Coverage-Metrik aus Deeptest, die Neuron Coverage, sind andere Coverage-Metriken, die k-Multisection Neuron Coverage und die Distance based Suprise Adequacy evaluiert. Anhand dieser Coverage-Metriken sind neue Ansätze für Bewertungsmetriken neuronaler Netze und deren Trainingsdaten vorgestellt. Dabei zielen die weiterführenden Ansätze ausschließlich auf die Bewertungsmetriken ab und nicht auf die Generierung neuer fehlerhafter Daten.**

***Index Terms*—neuronale Netz, Automatisiertes Testen, Coverage-Metriken**

I. Einleitung

Das neuronale Netz ist die Basistechnologie einiger Technologien. So realisiert es z. B. maschinelle Übersetzung [1] oder intelligente Sprachassistenten [2]. Neuronale Netze generalisieren Daten und können somit Daten verarbeiten, die sie während des Trainingsverlaufs nicht gelernt haben [3].

Trotz der generalisierenden Eigenschaft weist das Neuronale Netz unerwartetes Fehlverhalten auf. Dieses Fehlverhalten kann fatalen Konsequenzen, wie einen Unfall zur Folge haben. Ein Beispiel ist ein autonomes Fahrzeug von Tesla, das einen hellen Lastwagen mit dem Himmel verwechselt hat und dadurch einen Unfall verursacht hat, bei dem der Fahrer des Tesla verstarb [4].

Damit das neuronale Netz in sicherheitskritischen Systemen eingesetzt werden darf, benötigt es somit eine Art der Validierung für seine Robustheit bzw. Fehlertoleranz. Ein Problem bei der Validierung neuronaler Netze ist die Funktionsweise der neuronalen Netze selbst. Während herkömmliche Software-Tests die Software auf fehlerhaftes Verhalten testet [6], ist dies bei neuronalen Netzen nicht möglich [7].

Um dieses Problem zu lösen, gibt es den Ansatz, Fehlverhalten durch synthetische Daten zu erzwingen und die Daten anschließend zu den Trainingsdaten des neuronalen Netzes hinzuzufügen. Dadurch lernt das neuronale Netz ebenfalls die fehlerhaften Daten und die Anzahl der restlichen Fehlverhalten nimmt ab. Dabei besteht die Hauptaufgabe darin, die Daten effizient zu synthetisieren, um so alle Fehlverhalten des neuronalen Netzes abzubilden. Diese Aufgabe ist mithilfe des Algorithmus aus Deeptest realisiert.

II. Analyse: Deeptest

Deeptest ist eine Veröffentlichung, indem ein Algorithmus automatisiert Fehlverhalten innerhalb neuronalen Netzen autonomer Automobils findet [7]. Dabei analysiert der Algorithmus neuronale Netze, die mithilfe von Bilderkennung den Lenkradwinkel innerhalb eines Kamerabildes bestimmen.

Die neuronalen Netze, die in Deeptest analysiert sind, sind neuronale Netze, die 2016 eine hohe Platzierung innerhalb der Udacity self-driving challenge belegt haben [8]:

- Rambo - Platz 2:
  bestehend aus drei neuronalen Netzen (3x CNN)
- Chauffeur - Platz 3:
  bestehend aus zwei neuronalen Netzen (CNN, LSTM)
- Epoch - Platz 6:
  bestehend aus einem neuronalen Netz (CNN)

Das Fehlverhalten generiert der Algorithmus durch zwei Schritte: realistische Bilder mithilfe von Transformationen generieren und die Kombination verschiedener Transformationen durch die Metrik Neuron Coverage leiten.

Generierung synthetischer Bilder

Die synthetischen Bilder des Algorithmus sind nicht willkürlich generiert, sondern bilden reale Szenarien nach [7]. Dafür entstehen die synthetischen Bildern auf Basis von Trainingsdaten. Ein synthetisches Bild ist auf Basis eines jeweiligen Ursprungsbildes, das mithilfe einer Transformation verändert ist.

Die Transformation ist eine von neun definierten Transformationen: Änderung der Helligkeit, Änderung des Kontrasts, Unschärfe, Nebel-Effekt, Regeneffekt, Verschiebung,

Skalierung, horizontale Scherung, Rotation. Die Stärke der Transformation ist parametrisierbar.

Diese Art der Transformation hat zwei Eigenschaften. Die erste Eigenschaft ist, dass die generierten Bilder realistische Szenarien abbilden z. B. realistisches Bild mit Nebel-Transformation ergibt ein realistisches Nebelbild. Die zweite Eigenschaft ist, dass dadurch eine metamorphische Beziehung zwischen den Bildern und dem Label vorhanden ist.

Neben der Generierung der Bilder ist es eine Aufgabe die Bilder anschließend zu labeln. Dafür ist eine Kenntnis über den Inhalt des Bildes nötig, allerdings ist dies, nach einer beliebigen Änderung des Bildes, nur durch einen Menschen möglich. Durch die Transformationen und die metamorphische Beziehung ist das neue Label automatisiert bestimmt. Entweder ändert sich der Lenkradwinkel nicht oder er ist berechenbar. Bei z. B. Nebel ändert sich die Kurve des Ursprung-Bildes nicht und bei einer Rotation in Kurvenrichtung ist die Kurve dementsprechend schärfer.

Dadurch ist die Generierung synthetischer Bilder durch einen Algorithmus ohne menschliches Eingreifen realisiert. Die Auswahl der Transformationen, um Bilder mit Fehlverhalten zu generieren, ist mittels der Metrik Neuron Coverage geleitet.

Leitung der Generierung

Damit die Generierung synthetischer Bilder nicht willkürlich erfolgt, ist die Auswahl der Transformationen und deren Kombinationen mithilfe der Metrik Neuron Coverage geleitet [7]. Die Neuron Coverage wurde erstmals in der Veröffentlichung DeepXplore [9] vorgestellt und leitet sich von der Code-Coverage ab.

Die Code-Coverage ist ein prozentualer Wert und beschreibt bei einem Software Test, wie viele Programmzeilen während des Tests durchlaufen sind [6]. Eine hohe Code-Coverage bedeutet, dass viele Programmzeilen durchlaufen sind und dadurch viel Quellcode analysiert ist. Die Herleitung der Neuron Coverage ist daran angelehnt. Bei einer hohen Neuron Coverage sind mehr Neuronen aktiviert. Dadurch entsteht mehr Diversität am Ausgang [7].

Die Berechnung der Neuron-Coverage ist durch folgende Gleichung

$$Neuron\ Coverage = \frac{|Activated\ Neurons|}{|Total\ Neurons|}, \qquad (1)$$

definiert. Die Anzahl an aktivierten Neuronen, geteilt durch die Gesamt-Anzahl, ergibt die Metrik. Ein Neuronen gilt als aktiviert [7], sobald diese einen Ausgangswert größer einem Schwellenwert t hat. In Abbildung 1 ist die Neuron Coverage anhand zweier neuronaler Netze visualisiert. Die Kreise repräsentieren die Neuronen mit den jeweiligen Ausgangswerten der Neuronen.

Alle Neuronen, die den Schwellenwert t=0,2 überschreiten, sind dunkel blau eingefärbt. Dadurch ergibt sich die Neuron Coverage für beide neuronale Netze von 0,5 (50%).

Zwischen der Metrik Neuron Coverage und den Transformationen besteht eine positive Korrelation [7]. Dadurch

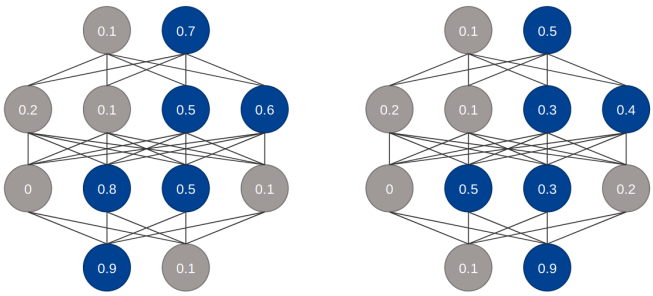

Abbildung 1. Zwei neuronale Netze mit unterschiedlichem Input. Die Neuronen mit Wert über 0,2 sind blau eingefärbt.

ist mithilfe der Neuron Coverage die Entscheidung möglich, welche Transformationen kombiniert werden. Diese Korrelation ist ebenfalls durch die Kombination mehrerer Transformationen gegeben. Die Auswahl der Transformationen erfolgt mit in einem Greedy-Search Ansatz. Die Transformationen sind so kombiniert, dass die Neuron Coverage maximal ist.

Definition Fehlerhafte Bilder

Mithilfe des Greedy-Search Algorithmus sind die Bilder mit maximaler Neuron Coverage generiert [7]. Eine Möglichkeit, um fehlerhafte Bilder zu ermitteln, ist es, den Fehler des Ursprungsbildes mit dem Fehler des transformierten Bilder zu vergleichen.

Das Problem dabei ist, dass dadurch viele false positive Bilder entstehen, da nur das label richtig ist. Als Beispiel ist es nicht fehlerhaft durch eine 30°-Kurve mit 29°, aber auch 31° durchzufahren [7]. Aufgrund dessen hat die Gleichung ein Toleranzmaß. Ein fehlerhaftes Bild ist als fehlerhaft klassifiziert, sofern folgende Ungleichung nicht erfüllt ist:

$$(\theta_i - \theta_{ti})^2 \leq \lambda \cdot MSE_{Orig} \qquad (2)$$

MSE_{Orig} ist der Mean-Sqaured-Error des Sets der Ursprungsbilder und deren Labels. θ_i ist die Vorhersage des neuronalen Netzes für das Ursprungsbild und θ_{ti} ist die Vorhersage des transformierten Bildes. λ ist ein frei definierbarer Parameter. Sobald die Differenz beider Vorhersagen größer als $\lambda \cdot MSE_{Orig}$ ist, gilt das transformierte Bild als fehlerhaft. Die Anzahl der false positive Bilder ist somit von λ abhängig [7].

Bilder die die Gleichung 2 nicht erfüllen, sind fehlerhafte Bilder und haben ein Potenzial, um fehlerhaftes Verhalten zu zeigen. Allerdings ist bei hohen Transformationen (z. B. einer Rotation von 30°) nicht gewährleistet, dass der Inhalt weiterhin richtig berechnet ist. Aufgrund dessen ist neben Gleichung 2 eine weitere Beschränkung von false positive Bildern nötig [7]. Diese ist definiert als

$$|MSE_{trans,param} - MSE_{Orig}| \leq \epsilon, \qquad (3)$$

wobei MSE der Mean-Squared-Error und ϵ ein frei wählbarer Parameter ist. Bilder mit vielen Transformationen (z. B. Regen mit Nebel- und Rotations-Transformationen)

haben einen höheren MSE. Gleichung 3 filtert somit fehlerhafte Bilder heraus, die zu hohe Transformationen haben bezogen auf das Ursprungsbild. ϵ ist ein frei wählbarer Parameter, wie auch λ, der das Verhältnis zwischen false positive Bildern und der Fehlergrenze abwägt [7]. Demnach sind Bilder erst dann als fehlerhaft klassifiziert, wenn diese Gleichung 2 und 3 erfüllen.

Die Anzahl der false positive Bildern ist somit von den Parametern λ und ϵ abhängig.

Fazit Analyse Deeptest

Der Algorithmus aus Deeptest hat innerhalb von 6 neuronalen Netzen insgesamt 6339 fehlerhafte Verhalten durch synthetische Bilder entdeckt [7]. Dazu gibt es zwei Anmerkungen:

- Die Anzahl der fehlerhaften Bilder sind abhängig von den gewählten Parametern ϵ und λ. Demnach liegt die Anzahl der fehlerhaften Bilder zwischen 3484 und 45948, je nach Wahl der Parametern.
- Die Anzahl der fehlerhaften Bilder durch Nebel-Transformationen innerhalb der Rambo-Architektur ist ein hoher Anteil der Gesamtfehleranzahl. Demnach sind 4112 Fehler von 6339 Bilder ($\approx 65\%$) fehlerhafte Nebelbilder innerhalb der neuronalen Netze von Rambo.

Innerhalb der fehlerhaften Daten sind 130 false positive. Die false positive Bilder sind manuell evaluiert.

Zusätzlich hat die Veröffentlichung Deeptest die Frage behandelt, ob ein erneutes Training mit den synthetischen Bildern, die Robustheit der neuronalen Netze erhöht. Dafür ist der MSE des neuronalen Netzes von Epoch mit ursprünglichen Daten, Nebel- und Regentransformierten Daten bewertet. Demnach verbessert sich der MSE im besten Fall um bis zu 46% (von 0,18 zu 0,1).

Eigenes Fazit

Trotz des "automatischen" Tests des Algorithmus aus Deeptest ist der menschliche Faktor immer noch in der Parameterwahl involviert. So ist die Wahl geeigneter Parameter ϵ und λ entscheidend für die Anzahl an false positive Bildern. So entscheidet der menschliche Faktor die Anzahl von false positive Bildern und damit auch der manuelle Evaluierungsgrad. Im Falle von Zeit- bzw. Arbeitsersparnis ist die Wahl von ϵ und λ dementsprechend hoch um die Anzahl von false positive Bildern zu verringern. Dadurch verringert sich aber auch die Anzahl der möglichen Fehler, die Gleichung 2 (ϵ) oder 3 (λ) nicht überschreiten, trotz fehlerhaftem Verhalten.

Ebenso sind die synthetischen Bilder keine guten Realdaten [10]. Bei einer Rotation werden die fehlenden Pixel durch schwarze Pixel aufgefüllt, wodurch das Bild eine komplett andere Information beinhaltet [12]. Diese schwarzen Ecken sind weder realistisch, noch in den Ursprungsdaten enthalten.

Neben den Parametern ϵ und λ ist die Neuron Coverage mit der Wahl des Schwellwerts ebenfalls von dem menschlichen Faktor abhängig. Zusätzlich weist die Neuron-Coverage Schwächen bei u. a. adversarial Attacks auf, weswegen weitere Coverage-Metriken in dem folgendem Kapitel erörtert sind [11] [12] [13].

III. Coverage-Metriken

Coverage-Metriken sind Metriken, die die Coverage von neuronalen Netzen misst. Im folgenden Kapitel sind drei Metriken miteinander verglichen: Neuron Coverage, k-Multisection Neuron Coverage und die Distance based Surprise Adequacy.

Neuron Coverage

Wie in dem Ergebnis der Analyse von Deeptest bereits angemerkt, weist die Neuron Coverage Schwächen auf, bei der Erkennung von fehlerhaften Bildern wie z. B. adversarial Examples. Zwar aktivieren unterschiedliche Klassen unterschiedliche Neuronen bei einer Klassifikation [9], allerdings geht diese Information bei der Neuron Coverage verloren. Die Neuron Coverage speichert die Anzahl aller Neuronen über einem Schwellenwert, allerdings nicht, welches Neuron den Wert überschritten hat. Ebenso unterscheidet die Neuron Coverage nicht bei den Werten 0,9 und 0,3 (bei t=0,2) [11], da diese als aktiviert zählen. Dies macht die Neuron Coverage sehr anfällig für Fehlverhalten, was anhand Abbildung 1 veranschaulicht ist.

Die Neuronen mit einem Wert $>$0,2 sind dunkelblau eingefärbt und sind die signifikanten Neuronen. Die signifikanten Neuronen in beiden neuronalen Netzen sind identisch, bis auf die zwei Neuronen in der letzten Schicht (klassifizierenden Neuronen). Die Auffälligkeiten, dass die signifikanten Neuronen fast identisch sind, trotz unterschiedlicher Klassen, ist durch die Neuron-Coverage nicht repräsentiert, da beide neuronale Netze eine Neuron-Coverage von 0,5 haben.

k-Multisection Neuron Coverage

Die k-Multisection Neuron Coverage (k-Multisec.) [11] ist nicht ein prozentualer Wert, der die aktivierten Neuronen misst, sondern zeichnet alle Werte eines Neurons auf und berechnet die Auslastung des Neurons. Dafür benötigt k-Multisec. für das neuronale Netz neben der Anzahl an Intervallen k auch für jedes Neuron einen individuellen Wertebereich $[low_n, high_n]$.

Die Analyse mithilfe der k-Multisec. ist durch zwei Schritte realisiert. Als erstes durchlaufen alle Trainingsdaten das neuronale Netz. Dadurch ermittelt k-Multisec. die Grenzen des Wertebereichs low_n und $high_n$. Diese bilden den abgeschätzte Wertebereich, der in k äquidistante Intervalle unterteilt ist. Anschließend durchlaufen die Trainingsdaten das neuronale Netz erneut und markieren das Intervall des Neurons, in dem der Wert des Neurons liegt. Abbildung 2 veranschaulicht diesen Vorgang.

In der Abbildung sind zwei neuronale Netze mit den jeweiligen Werten der Neuronen abgebildet, die durch Input A und B entstanden sind. Die Anzahl der Intervalle k ist auf 3 gesetzt und der Wertebereich ist zu Anschauungszwecken für alle Neuronen auf [0,1] gesetzt. Die Intervalle, in denen die Werte liegen, sind markiert (in Abbildung 2 eingefärbt).

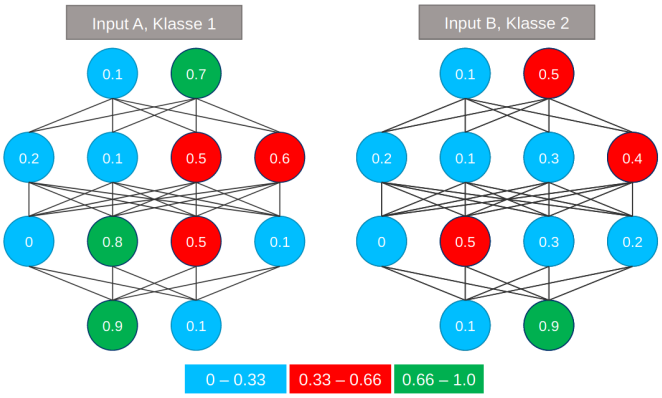

Abbildung 2. Zwei neuronale Netze mit unterschiedlichem Input. Die Neuronen mit entsprechendem Wertebereich sind dementsprechend eingefärbt.

Die Neuron Coverage ist für beide neuronale Netze 0,5 (bei t=0,2). Die k-Multisec. hat zusätzlich die Information, dass nach mehreren Inputs die Neuronen oben links keinen anderen Wert angenommen haben. Dies zeigt, dass die Neuronen oben links, nicht ausgelastet sind, da diese nur einen Wert annehmen. Bezogen auf alle Trainingsdaten ist das eine Analyse, wie sehr die Daten die einzelne Neuronen auslasten. Wenn ein Neuron z. B. über alle Daten hinweg ausschließlich in einem Intervall auftreten, ist keine Aussage über das Verhalten der Neuron in den anderen Wertebereichen möglich. Dadurch zeigt die k-Multisec. eine Metrik für die Auslastung des neuronalen Netzes bezogen auf die einzelnen Neuronen. Die k-Multisec. betrachtet allerdings ausschließlich die Neuronen alleine und keine Kombinationen.

Distance based Surprise Adequacy

Die Distance based Surprise Adequacy (DSA) [12] ist ein Maß für die "Überraschung" des neuronalen Netzes durch neuer Daten. Dieses Maß setzt dabei die bereits vorhanden Daten in Relation zueinander. Für alle Trainingsdaten, sind die jeweiligen Werte der Neuronen in einem Vektor abgespeichert. Anschließend erhält jedes Neuron einen DSA-Wert ($DSA(x)$). Dieser Wert berechnet sich aus zwei euklidischen Distanzen. Die erste Distanz ist die des Neuronenvektors von x zu dem nächsten Neuronenvektor einer Eingabe, ungleich von x, der gleichen Klasse von x ($dist_a(x)$). Die zweite Distanz ist die nächste Distanz zu einem Neuronenwert mit einer Klasse ungleich der Klasse von x ($dist_b(x)$). Die Metrik ist wie folgt berechnet:

$$DSA(x) = \frac{dist_a(x)}{dist_b(x)} \tag{4}$$

Je kleiner der DSA-Wert ist, desto sicherer ist der Neuronenvektor nahe Neuronenvektoren der gleichen Klasse ($dist_a$ ist klein). Sobald der Wert von DSA(x) größer als 1 ist, ist die Distanz zu einem Neuronenvektor einer anderen Klasse näher ($dist_b$), als der zu der gleichen Klasse ($dist_a$). Dies ist ein Indiz dafür, dass dieses Datum sehr unsicher klassifiziert ist, da es trotz richtiger Klassifizierung, näher an einer anderen Klasse liegt.

Da sich mit DSA(x) eine Decision-Boundary [12] abzeichnet, ist diese Metrik ungeeignet für kontinuierliche Werte, da keine exakte Decision Boundary vorhanden ist, wie bei Klassifikationen.

IV. Weiterführende Ansätze

Ein möglicher schematische Ablauf der Entwicklung neuronaler Netze ist in Abbildung 3 abgebildet. Während Deeptest bei der Frage ansetzt, wie fehlerhaftes Verhalten entdeckt wird, ist in den weiterführenden Ansätzen eine andere Frage betrachtet. Sobald ein trainiertes neuronales Netz, nicht die gewünschte Vorhersagegenauigkeit erreicht, verfügt das neuronale Netz über viele freie Parameter, durch diese sich das Verhalten ändert. Aufgrund dieser Problematik verfolgen die

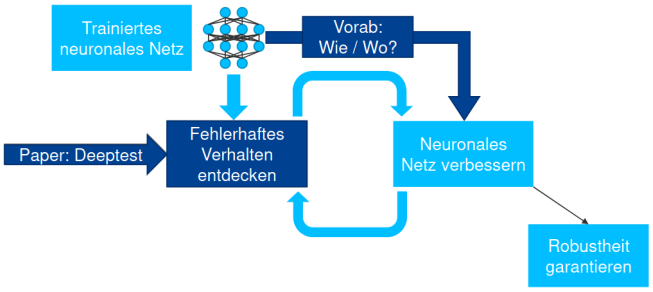

Abbildung 3. Schematischer Ablauf bei der Entwicklung neuronaler Netze

weiterführenden Ansätze die Frage wie das neuronale Netz verbessert werden kann, sobald die Vorhersagegenauigkeit des trainierten neuronalen Netzes unzureichend ist. Da das neuronale Netz über viele Parameter verfügt, ist die Frage wie folgt spezifiziert: Wie werden potentiell fehlerhafte Neuronen und Daten ermittelt. Sie sind eine zusätzliche Bewertungsmetrik, die keine zusätzliche Daten benötigen, sondern allein anhand des neuronales Netzes und dessen Daten berechnet sind.

Bei den weiterführenden Ansätzen handelt es sich um rein theoretische Überlegungen, welche nicht durch wissenschaftliche Prüfungen belegt sind.

Explorationstool für neuronale Netze

Dieses Tool basiert auf der k-Multisec.-Metrik. Der Fokus dieser Metrik liegt dabei die Größe des neuronalen Netzes zu bewerten. Die k-Multisec. wird dafür berechnet und hat einen prozentualen Wert pro Neuron innerhalb des neuronalen Netzes. Der prozentuale Wert der k-Multisec. ist dabei eine Aussage wie sehr ein Neuron ausgelastet ist. Angenommen ein Neuron hat eine k-Multisec. von 10%, so hat das Neuron 10% seines Wertebereiches als Wert angenommen. Die restlichen 90% der Werte sind in den Trainingsdaten nicht aufgetreten. Somit ist keine Aussage über das Verhalten des Neurons, und indirekt über das neuronale Netz, innerhalb dieser 90% möglich. Angenommen innerhalb eines neuronalen Netzes ist eine Gruppe an Neuronen vorhanden, die wenige Intervalle

ausgelöst haben (siehe Beispiel 2 links oben), so ist es eine Möglichkeit, die Schichten zu verkleinern und die Neuronen wegzulassen.

Gruppierung und Bewertung von Daten

Dieser Ansatz basiert auf der DSA-Metrik. Dafür wird der Neuronenvektor aller Daten aufgezeichnet. Anschließend werden die Daten mit einem DSA-Wert über 1 ermittelt. Diese Neuronenvektoren sind näher an Neuronenvektoren anderer Klassen, als Neuronenvektoren ihrer Klasse. Daten die dann z. B. durch den k-nearest-neighbour eine andere Klasse zugewiesen bekommen, zeigen eine Similarität zu den Daten, anderer Klassen. Ebenso haben diese ein erhöhtes Potenzial bei einer Verschiebung der Decision-Boundary, durch Training des neuronalen Netzes, möglicherweise falsch klassifiziert zu werden, da diese sehr nahe an der Boundary sind. Diese Information ist für abstrakte Daten interessant, bei denen Abhängigkeiten von Menschen nicht ersichtlich sind.

V. Stakeholder-Analyse

Die Stakeholder sind in den Gruppen direkte und indirekte Stakeholder unterteilt. Direkte Stakeholder sind in erster Linie von der Idee direkt betroffen. Die Value-Proposition ist auf die direkten Stakeholder ausgelegt.

Da sich diese Ausarbeitung mit der Analyse neuronaler Netzen beschäftigt, sind die direkten Stakeholder Entwickler und Tester künstlicher Intelligenz bzw. neuronaler Netze. Übergeordnet ist ebenfalls die Firma der Entwickler ein direkter Stakeholder.

Indirekte Stakeholder sind zum einen die Nutzer, die ein mehr getestetes System benutzen, zum anderen aber auch Versicherer und Gerichte. Die Versicherer und Gerichte profitieren in erster Linie von den weiterführenden Ansätzen, in denen Bewertungsmetriken zur Analyse von neuronalen Netzen und deren Daten gezeigt sind. Dies ist insbesondere dann interessant, wenn zum Beispiel bei einem Versicherungsfall oder auch bei Gerichtsverfahren abgewogen wird, ob das neuronale Netz ein Fehlverhalten aufweist, oder ob es eine fahrlässige Anwendung des Nutzers ist.

VI. Value-Proposition

Die Value-Proposition ist für den direkten Stakeholder, die KI-Entwickler, erstellt. Um den KI-Entwickler zu konkretisieren, ist dieser von der Firma Bosch GmbH und in dem Bereich der Bilderkennung von autonomen Autos tätig. Die Aufgaben der Entwickler beziehen sich hauptsächlich auf die Entwicklung neuronalen Netze. Der schematische Ablauf dieser Arbeit ist in Abbildung 4 abgebildet.

Aufgaben

Diese sind bei der Entwicklung neuronaler Netze in drei Hauptkategorien unterteilt: Vorüberlegung, Training und Bewertung. In den Vorüberlegungen sind Aufgaben, die vorab für die Entwicklung des neuronalen Netz abgearbeitet werden. So werden zuerst die Testfälle der Aufgabe analysiert, die das neuronale Netz erledigt. Dies sind z. B. die Arten der

Abbildung 4. Schematischer Ablauf der Entwicklung neuronaler Netze

Straßenschilder bei einer Bilderkennung. Ausgehend von den Testfällen sind Daten ausgewählt, die die Testfälle abbilden. Neben den Daten wird eine Architektur des neuronalen Netzes ausgewählt. Diese ist ebenfalls abhängig von der Aufgabe des neuronalen Netzes.

Nach diesen Vorüberlegungen wird das neuronale Netz trainiert. Nachdem das neuronale Netz trainiert ist, werden die Fehlverhalten des neuronale Netzes untersucht. Ein Fehlverhalten ist in diesem Beispiel eine falsche Klassifizierung von Verkehrsschildern. Gegebenenfalls wird das neuronale Netz nach der Analyse der Fehlverhalten erneut trainiert.

Nach dem Training wird das Neuronale Netz evaluiert und im Falle mehrerer Neuronaler Netze werden diese miteinander verglichen.

Pains

Bei jedem dieser Schritte treten unterschiedliche Pains auf. Bei der Aufgabe der Testfälle und der Daten bleibt die Frage, ob alle Testfälle überhaupt ermittelt sind. In dem Beispiel mit den Verkehrsschildern ist die Frage, ob alle zu erkennenden Verkehrsschilder definiert sind. Damit das neuronale Netz die Testfälle erlernt, ist eine weitere Frage, ob die Datenmenge für diese Aufgabe groß genug ist.

Bei der Architektur des neuronalen Netzes ist die Struktur vorgeben wie z. B. ein CNN. Allerdings sind andere Hyperparameter wie z. B. die Anzahl an Schichten oder die Anzahl der Neuronen dem Entwickler überlassen.

Bei der Bewertung neuronaler Netze ist die Frage, wonach richtet sich die Bewertung? Nach dem Trainingsdurchlauf gibt es die Accuracy des Models, allerdings ist diese nur ein prozentualer Wert um ein neuronales Netz zu vergleichen. Über die kompletten Schritte hinweg bleibt die Frage: wie bzw. wo ist es effektiv, das neuronale Netz zu verbessern.

Gains

Die Gains der einzelnen Schritte beantworten die Fragen, die innerhalb der Pains aufgelistet sind. Demnach sind Gains für die Vorüberlegung zum einen eine Bewertungsmetrik der Daten, die folgende Fragen behandelt: reicht die Anzahl der Daten aus und bilden die Daten alle Testfälle ab? Dies ist vor allem für abstrakte Daten interessant, die nicht vom Menschen interpretierbar sind. Ein weiterer Gain für die Vorüberlegung ist, dass die Generierung von Daten automatisiert ist, sofern die Menge der Daten nicht ausreicht.

Bezüglich des Trainings ist ein Gain, dass der Test auf Fehlverhalten der neuronalen Netze automatisiert erfolgt. Bezogen auf die Bewertungsmöglichkeiten ist ein Gain, eine weitere Bewertungsmöglichkeit für neuronale Netze zu haben, um diese miteinander zusätzlich zu vergleichen.

Bezogen auf den Haupt-Pain, die Frage wie und wo das neuronale Netz verbessert wird, ist der Haupt-Gain eine Art Leitung zu Fehlerlösungen, wenn das Ergebnis des neuronalen Netzes unzureichend ist.

Products & Services

Die Products und Services sind anhand des Veröffentlichung Deeptest und den weiterführenden Ansätzen abgeleitet. Wie bereits in den weiterführenden Ansätzen angemerkt, eignet sich die k-Multisec., um das neuronale Netz zu durchleuchten. Dies ist eine Bewertung des neuronalen Netzes durch die Daten. Des Weiteren ist ein Product, das mithilfe von DSA die Daten des neuronalen Netzes gruppiert. Aus der Arbeit Deeptest ist das Product das metamorphische Bildlabeling und das automatische Finden von Fehlverhalten.

Pain Relievers

Mithilfe der DSA ist durch die Gruppierung der Daten indirekt eine Bewertung möglich z. B. gibt es ungewollte Similaritäten. Ein weiterer Pain Reliever der Vorüberlegung ist die Bewertung des neuronalen Netzes durch die k-Multisec.. Dies ist ein Maß für die Auslastung des neuronalen Netzes und somit ein Hinweis auf zu große Netze oder überflüssige Neuronen. Auf der Seite der Pains ist dies eine Bewertungsmetrik, um verschiedene Netze miteinander zu vergleichen. Mithilfe des metamorphischen Bidlabelings aus Deeptest, ist die automatische Bildgenerierung realisiert. Die einzigen Voraussetzungen sind logische Transformationen und bereits gelabelte Ursprungsbilder. Die Generierung ist ein Pain Reliever, der angewandt wird, falls zu wenig Daten vorhanden sind. Die beiden weiterführenden Ansätze sind ebenfalls indirekte Pain Reliever für den Haupt-Pain: wie oder wo ich bei neuronalen Netzen nachbessern kann. Somit ist zumindest eine weitere Aussage über Daten und neuronale Netze möglich.

Gain Creators

Wie in den Pain-Relievers ermöglicht die k-Multisec. die Durchleuchtung des neuronalen Netzes und somit eine zusätzliche Bewertung des neuronalen Netzes aus Sicht der Daten. Dadurch lässt sich ableiten, ob das neuronale Netz zu groß ist und ob alle Zustände erfasst sind. Mithilfe der DSA bewertet das neuronale Netz die Daten und gruppiert diese. Dabei ist der Ansatz, ein Produkt zu erstellen, das die Vektordistanzen anschaulich formuliert und somit Similaritäten visuell aufarbeitet. Durch den Algorithmus aus Deeptest ist sowohl eine automatisierte Datensynthese, aber auch eine Ermittlung von Fehlverhalten realisiert.

VII. Fazit

Diese Arbeit behandelt Ansätze, die sich damit befassen, wie das Verhalten von neuronalen Netzen verbessert wird. Dieser Bereich der Forschung ist wichtig, da autonome Systeme, basierend auf neuronalen Netzen, ohne eine Garantie der Robustheit, keine hohe Akzeptanz in der Bevölkerung finden werden.

Deeptest

Die Ansatzfrage (Abbildung (3)) von Deeptest ist es, wie fehlerhaftes Verhalten entdeckt wird.

Mithilfe von synthetischen Bildern hat der Algorithmus fehlerhaftes Verhalten in neuronalen Netzen entdeckt. Durch das metamorphischen Bildlabelings ist das Label der neuen Daten automatisiert erzeugt.

Trotz der automatischen Analyse ist der Mensch dennoch eine Komponente dieses Testverfahrens, da er durch die Parameterwahl Einfluss auf die Anzahl an fehlerhaften und false positive Bildern hat. Ebenso sind die false positive manuell evaluiert, was einen zusätzlichen Arbeitsschritt verursacht. Durch diesen zusätzlichen Schritt neigt der Mensch zu einem hohen ϵ und λ, wodurch die Anzahl der false positive als auch die fehlerhaften Daten abnehmen, da dies Zeit- und Arbeitsersparnis zur Folge hat.

Des Weiteren entstehen durch die Transformationen synthetische Daten, die nicht realistisch sind [10], da diese u. a. ausschließlich mit schwarzen Pixeln aufgefüllt sind [12]. Um realistische Daten zu synthetisieren sind bessere Ansätze vorgestellt wie in der Veröffentlichung DeepRoad [10] oder dem Startup RabbitAI [14].

Unabhängig von den Daten weißt die Neuron Coverage Schwächen auf. Die Neuron Coverage ist anfällig für Fehlverhalten und erkennt dies im Gegensatz zur k-Multisec. oder DSA nicht [11] [12].

Weiterführende Ansätze

Die weiterführenden Ansätze befassen sich mit der Frage: wie potentiell fehlerhafte Daten und Neuronen ermittelt werden können. Auf Basis dieser Fragestellung sind zwei Ansätze mithilfe der Coverage-Metriken k-Multisec. und DSA ausgearbeitet. Hierbei ist eine Bewertung des neuronalen Netzes mithilfe der Daten realisiert und umgekehrt. Dies ist eine theoretische Vorabbetrachtung und bietet Bewertungsmöglichkeiten, die in der Praxis allerdings noch validiert werden sollten. Keine dieser Metriken hat eine Korrelation mit der Robustheit der neuronalen Netzen, egal welche [13].

Exploration von neuronalen Netzen: Mithilfe der k-Multisec. ist eine Bewertung des neuronalen Netzes durch die Daten gegeben. Die Daten durchleuchten das neuronale Netz und zeigen, welche Neuronen welche Werte erreicht hat. Dies ist eine Metrik für potenziell überflüssige Neuronen. Über Neuronen, die nur wenige Wertebereiche während des Trainings angenommen haben, lässt sich keine Aussage über die anderen Werte machen, die in der Realität vielleicht auftreten.

Analyse der Daten: Mithilfe der DSA ist eine Bewertung der Daten durch das neuronale Netz gegeben. Dabei sind die Daten durch den Menschen gelabelt und zusätzlich über die DSA gruppiert. Dadurch zeigen sich Unstimmigkeiten

zwischen den vorgelabelten Klassen und denen, wie es das neuronale Netz zuweist. Ebenso zeigt DSA eine ”Sicherheit”, wie sicher die Daten klassifiziert sind.

VIII. Ausblick

Ausgehend von den theoretischen Ansätzen ist der nächste Schritt, diese mit realen Systemen zu validieren oder zu widerlegen. Um den Effekt von DSA zu ermitteln sollten absichtliche Similaritäten, z. B. ein Wasserzeichen in Bildern, eingesetzt werden, um zu sehen, ob diese ermittelt werden.

Literatur

[1] Ramon Wartala: Praxiseinstieg Deep Learning, 1. Auflage, dpunkt Verlag, 208, Heidelberg

[2] Spiegel (2020): Sprachassistenten im Vergleich: Was können Siri, Alexa und Co? , URL: https://www.spiegel.de/gutscheine/magazin/ sprachassistenten (Stand 14.02.2020)

[3] Mario Geigera, Arthur Jacotb, Stefano Spiglera et.al (2019): Scaling description of generalization with number of parameters in deeplearning, URL: https://arxiv.org/pdf/1901.01608.pdf (Stand 14.02.2021)

[4] Welt (2016): Selbstfahrender Tesla übersieht weißen Lkw vor Wolkenhimmel, URL: https://www.welt.de/wirtschaft/article156727084/Selbstfahrender-Tesla-uebersieht-weissen-Lkw-vor-Wolkenhimmel.html (Stand 14.02.2021)

[5] Günter Eymann - VDI (2019): Künstliche Intelligenz und autonomes Fahren, URL: https://www.vdi.de/news/detail/kuenstliche-intelligenz-und-autonomes-fahren (Stand 14.02.2021)

[6] Wikipedia (2020): Testabdeckung, URL: https://de.wikipedia.org/wiki/Testabdeckung (Stand 14.02.2021)

[7] Yuchi Tian, Kexin Pei, Suman Jana, Baishakhi Ray (2018): DeepTest: Automated Testing ofDeep-Neural-Network-driven Autonomous Cars, URL: https://arxiv.org/abs/1708.08559.pdf (Stand 14.02.2021)

[8] Udacity (2016): Final Leaderboard, URL: https://github.com/udacity/self-driving-car/tree/master/challenges/challenge-2 (stand 14.02.2021)

[9] KexinPei, YinzhiCao, JunfengYang, SumanJana (2017): DeepXplore: Automated Whitebox Testing of Deep Learning Systems. URL: https://arxiv.org/abs/1705.06640v4.pdf (Stand 14.02.2021)

[10] MengshiZhang, YuqunZhang, LingmingZhang, CongLiu, SarfrazKhurshid (2018): DeepRoad: GAN-based Metamorphic Autonomous Driving System Testing. arXivpreprint URL: https://arxiv.org/abs/1803.07519.pdf (Stand 14.02.2021)

[11] Lei Ma, Felix Juefei-Xu, FuyuanZhang et. al. (2018): DeepGauge: Multi-Granularity Testing Criteria for Deep Learning Systems, URL: https://arxiv.org/abs/1803.07519.pdf (Stand 14.02.2021)

[12] Jinhan Kim, Robert Feldt, Shin Yoo (2019): Guiding Deep Learning System Testing using Surprise Adequacy, arXivpreprint URL:https://arxiv.org/abs/1808.08444.pdf (Stand 14.02.2021)

[13] ShenaoYan, GuanhongTao, XuweiLiu et. al. (2020): Correlations between Deep Neural Network Model Coverage Criteria and Model Quality, Virtual Event, USA. ACM, New York, NY, USA, 13 pages. URL:https://doi.org/10.1145/3368089.3409671 (Stand 14.02.2021)

[14] Rabiit AI: Homepage, URL: https://rabbitai.de/ (Stand 14.02.2021)