Maschinelle Sicherheitsüberprüfung Neuronaler Netze für Biometrische Authentifizierung

Nico Dietz
Hochschule Esslingen
{nidiit00}@hs-esslingen.de

I. Motivation und Problemstellung

Heutzutage sind Biometrische Authentifizierungsmethoden allgegenwärtig und selbst in mobilen Endgeräten etabliert. Insbesondere da diese Authentifizierung auch für sicherheitskritische Anwendungen genutzt wird, ist es wichtig, dass sie sicher ist. Das Problem hierbei ist, dass wenn wir als Mensch ein Bild einer Person sehen, können wir diese Person auf anderen Bildern wieder erkennen. Hierbei ist es auch egal, ob das mit einem Rauschen versehen wird. Bildrauschen beschreibt die Verschlechterung eines digitalen Bildes. Ein Neuronales Netz nimmt ein Bild allerdings nicht visuell wahr, sondern bekommt eine Matrix aus Zahlen. Wenn jetzt ein Rauschen addiert wird, ändert sich die komplette Matrix und somit ist das Bild nun aus Sicht des Netzes ein völlig anderes, obwohl es sich optisch nahezu nicht ändert. Jetzt kann es dadurch vorkommen, dass das Neuronale Netz plötzlich eine andere Person erkennt und somit einer falschen Person Zugriff auf Daten gibt, auf welche die Person eigentlich keine Zugriffsrechte hat. So ein Fall wird auch False Positive genannt. Um dieses Problem zu untersuchen kann ein Fuzzer verwendet werden. In der einfachsten Form generiert ein Fuzzer zufällige Eingaben und provoziert so bei Programmen einen Fehler, falls die generierte Eingabe nicht verarbeitet werden kann. Um nicht basierend auf Zufall zu arbeiten, kann die Methode des Coverage-Guided Fuzzing verwendet werden. Dabei erhält der Fuzzer die Möglichkeit nicht nur Eingaben zu generieren, sondern auch die jeweilige Ausgabe des Programms zu analysieren und basierend darauf die nächste Eingabe anzupassen. Damit kann die nächste Eingabe in die gewünschte Richtung gelenkt werden, um an das Ziel zu kommen. Mit dieser Arbeit soll durch diese Methodik ein False Positive provoziert werden. Zusätzlich wird untersucht, ob die Art des Netzes, die Trainingsart des Modells oder andere Parameter einen Einfluss auf diesen Prozess haben. Die Thematik ist unter dem Namen Adversarial Machine Learning bekannt und lässt sich in sechs Probleme aufteilen:

1) Ein lauffähiges Neuronales Netz mit trainiertem Modell aufsetzen.
2) Geeignete Schnittstellen für die Fuzzing Engine implementieren.
3) Algorithmen für die Mutation von Bildern implementieren.
4) Methode finden den Fortschritt des Fuzzers analysieren zu können.
5) Bewertungskriterien für den Vergleich Neuronaler Netze festlegen.
6) Den Einfluss von Reinforcement Learning untersuchen.

II. Verwandte Arbeiten

Die Arbeit „On the Resilience of Biometric Authentication Systems against Random Inputs“ [1] ist der Ursprung dieser Arbeit. Hier wird der Einfluss von Trainingsart des Modells auf zufällige Eingaben analysiert. Die dort entstandenen Ergebnisse können als Vergleich für das sechste Problem verwendet werden.

„TensorFuzz: Debugging Neural Networks with Coverage-Guided Fuzzing“ ist eine Arbeit die Coverage-Guided Fuzzing im Umfeld von Neuronalen Netzen thematisiert. [2] Die darin erwähnte Architektur dient als Grundlage für das zweite Problem, da mögliche Schnittstellen aufgezeigt und erklärt werden. Zusätzlich kann der implementierte Coverage Analyzer für das vierte Problem als Basis genutzt werden.

Der Artikel von Daniel Geng und Rishi Veerapaneni behandelt die Thematik des Adversarial Machine Learnings mit Beispielen. Hier ist besonders der Einfluss der Bildmutation und das Verhalten von Neuronalen Netzen auf mutierte Bilder der Fokus. [3] Die Bearbeitung des dritten und vierten Problems können von diesen Erkenntnissen profitieren, da mögliche Algorithmen für die Mutation erläutert werden. Außerdem wird deren Einfluss auf die Erkennung des Netzes direkt analysiert. Das genutzte Vorgehen kann für die Fortschritts-Analyse verwendet werden.

„Adversarial Noise Layer: Regularize Neural Network By Adding Noise“ ist eine Arbeit, welche sich primär mit der Berechnung von Rauschen für Bilder beschäftigt. [4] Die dort verwendeten Algorithmen sind als Grundlage für das dritte Problem und damit die Bildmutation geeignet.

„Generative Adversarial Nets“ behandelt die Thematik, dass zwei Netze sich gegenseitig verbessern. Das erste Netz versucht False Positives bei dem zweiten Netz zu provozieren. Hierdurch wird das zweite Netz immer besser in der Erkennung von Falscheingaben und das erste Netz immer besser im Erzeugen von Falscheingaben. [5] Diese Arbeit zeigt auf, wie Falscheingaben stetig verbessert und analysiert werden können.

„Recognition of Handwritten Digit using Convolutional Neural Network in Python with Tensorflow and Comparison of Performance for Various Hidden Layers“ behandelt das Aufsetzen eines Neuronalen Netzes in Tensorflow mit dem

MNIST Datensatz. Zudem werden unterschiedliche Hidden Layer miteinander verglichen. [6] Die Erkenntnisse aus dieser Arbeit können in Bezug auf das erste Problem angewendet werden.

III. Design

Um ein lauffähiges Neuronales Netz zu erhalten gibt es viele Möglichkeiten. Hierbei ist die Programmiersprache, das Framework der erste Schritt, gefolgt von dem Modell, welches ein bereits trainiertes sein kann oder man trainiert selbst eins. Im Umfang dieser Arbeit wird zunächst mit Python und TensorFlow begonnen, da verwandte Arbeiten größtenteils hierauf basieren und viele Bibliotheken und Dokumenten online verfügbar sind. Es wird mit einem bereits trainierten Modell angefangen, um sicherzustellen, dass es funktioniert, da selbst trainieren einige Probleme mit sich bringen kann. Es wird der MNIST Datensatz, welcher handgeschrieben Zahlen beinhaltet, verwendet. Dieser Datensatz ist sehr populär, wodurch es viele ausführliche Dokumentationen für einen einfachen Einstieg gibt. Ein komplexeres Problem ist die Verbindung eines Fuzzers mit einem Neuronalen Netz. Es gibt das Framework TensorFuzz, welches unterliegende Strukturen eines Neuronalen Netzes überprüfen kann. Für Coverage-Guided Fuzzing benötigt TensorFuzz einen Algorithmus zur Analyse des Fortschritts. Hierfür müssen Bewertungskriterien ermittelt werden, um zu sehen, ob das Ergebnis besser ist als das vorherige. Die Bewertungskriterien müssen dann dafür verwendet werden, um die nächste Eingabe weiter zu verbessern. Der erste Ansatz wäre mit dem MNIST Datensatz als Beispiel, wenn ein Bild mit einer Fünf anfänglich zu beispielsweise 90% einer Fünf zugeordnet wird. Wenn nun ein Rauschen addiert wird und das Netz nur noch zu 80% eine Fünf und stattdessen immer mehr eine Drei erkennt, dann geht es in die richtige Richtung. Ferner kann auch versucht werden ein Bild ohne Zahl so zu verändern, dass eine Zahl erkannt wird. Die erste Eingabe kann ein Bild aus dem MNIST Datensatz sein. Sobald der Fuzzer die Ausgabe analysieren kann, muss er auch die nächste Eingabe anhand des Ergebnisses verändern. Auf welche Art das Bild verändert wird muss verglichen werden. Es gibt viele Möglichkeiten, wie zum Beispiel ein Rauschen zu addieren, Filter über das Bild zu legen oder auch nur Teilbereiche des Bildes für die nächste Eingabe zu nutzen. Möglich ist auch eine Kombination aus mehreren Mutationen, welche je nach Analyse des Fuzzers bei unterschiedlichen Ergebnissen verwendet werden. Der erste Ansatz wird primär Rauschen behandeln, da dies Stand der Dinge ist und es hierzu einiges an Literatur und Algorithmen gibt. Für den Vergleich von Neuronale Netzen und Modellen untereinander gibt es viele mögliche Bewertungskriterien. In den meisten Fällen wird die sogenannte Accuracy verwendet. Im Beispiel von MNIST gibt es zehn Klassen, wobei jede Zahl von 0-9 eine eigene Klasse ist. Als Beispiel bekommt das Neuronale Netz ein Bild als Eingabe. Als Ausgabe gibt das Netz zurück, dass zu 76,5% eine Sieben, zu 23,5% eine Eins und zu 0% alle anderen Zahlen zu sehen sind. Die Accuracy beschreibt jetzt wie viele Aussagen richtig sind im Verhältnis zu allen Aussagen. Je nach Problemstellung kann die Accuracy bedingt angepasst werden. Damit ist gemeint, dass eine richtige nicht nur dann gezählt wird, wenn das richtige Ergebnis auch die Aussage mit der höchsten Wahrscheinlichkeit ist, sondern auch wenn die Antwort unter den beispielsweise zwei höchsten Wahrscheinlichkeiten gelistet ist. In dem eben genannten Beispiel würde das Ergebnis als richtig gezählt werden, wenn die Eingabe eine Eins oder eine Sieben waren. Alle weiteren möglichen Kriterien werden nur falls benötigt in Betracht gezogen werden. Das sechste Problem betrifft den Einfluss von Reinforcement Learning. Reinforcement Learning ist eine Art, wie ein Modell trainiert wird. Im Falle der Monte-Carlo-Methode wird eine Art Gamification verwendet. Wird nun ein Modell trainiert, bekommt ein Netz basierend auf einer Formel entweder Plus- oder Minuspunkte für jede Entscheidung die getroffen wird. Mit dem Punktestand, merkt das Netz, ob es sich verbessert oder verschlechtert hat. Wird ein Modell ohne Reinforcement Learning trainiert, dann bekommt das Neuronale Netz für jede Entscheidung lediglich die Rückmeldung, ob die Aussage richtig oder falsch ist. Um den Einfluss zu vergleichen, müssen mit dem gleichen Datensatz je ein Modell mit und ohne Reinforcement Learning trainiert werden. Diese Modelle können dann mit dem Fuzzer getestet werden, um zu sehen, ob eins der Modelle anfälliger für False Positives ist.

IV. Evaluation

Durch die frühe Entscheidung einer Sprache, eines Frameworks und eines möglichen Modells kann in diese Richtung von Anfang an viel ausprobiert werden und Erfahrung gesammelt werden. So wird das Risiko minimiert, dass anfänglich ohne tieferes Verständnis ein Konzept ausgearbeitet wird, welches am Ende nicht umsetzbar ist. Das erste Problem ist gelöst, sobald das Neuronale Netz Bildeingaben richtig zuordnen kann.
Die Probleme 2-4 beziehen sich auf den Fuzzer und werden in zwei Schritten gelöst. Zunächst muss der Fuzzer auf alle Schnittstellen Zugriff haben und mit rein zufälligen Eingaben arbeiten. Nach dem die Bewertungskriterien implementiert wurden, muss eine stetige Verbesserung erkennbar sein. Damit ist beispielsweise gemeint, dass ein Bild durch Veränderung immer mehr einer anderen Klasse zugeordnet wird. Sobald dies der Fall ist, gelten die Probleme als gelöst.
Das fünfte Problem ist bereits gelöst, wenn die Accuracy gemessen werden kann. Sollten weitere Bewertungskriterien benötigt werden, müssen diese auch jeweils messbar sein.
Reinforcement Learning und dessen Einfluss auf das Ergebnis ist das letzte Problem. Wenn zwei Modelle basierend auf dem gleichen Datensatz trainiert wurden. Das erste mit und das zweite ohne Reinforcement Learning, dann können bei mit dem Fuzzer verwendet werden. Das Verhalten wird dann analysiert und verglichen. Wenn die Analyse abgeschlossen ist, ist auch das Problem gelöst.

V. Ergebnis

Im Besten Fall würde ein Bild entstehen, welches eine Art Masterkey ist und bei einem Neuronalen Netz einen False

Positive provoziert. Deutlich realistischer wäre ein Algorithmus, welche die Suche nach einem False Positive erleichtert und beschleunigt. Möglich wäre allerdings auch, dass weder das eine noch das andere gefunden wird. Man muss aber auch bedenken, dass diese Thematik ein aktuelles Thema in der Entwicklung von Neuronalen Netzen ist und immer noch viel daran geforscht wird. Stand heute werden Informationen über den Datensatz oder Strukturen eines Neuronalen Netzes benötigt, um gezielt nach False Positives zu suchen.

LITERATUR

[1] Benjamin Zi Hao Zhao, Hassan Jameel Asghar and Mohamed Ali Kaafar, *On the Resilience of Biometric Authentication Systems against Random Inputs* https://arxiv.org/pdf/2001.04056.pdf Cornell University, 2020.

[2] Augustus Odena, Catherine Olsson, David G. Andersen and Ian Goodfellow, *TensorFuzz: Debugging Neural Networks with Coverage-Guided Fuzzing* http://proceedings.mlr.press/v97/odena19a/odena19a.pdf California, 2019.

[3] Daniel Geng and Rishi Veerapaneni, *Tricking Neural Networks: Create your own Adversarial Examples* https://medium.com/@ml.at.berkeley/tricking-neural-networks-create-your-own-adversarial-examples-a61eb7620fd8 2019.

[4] Zhonghui You, Jinmian Ye, Kunming Li, Zenglin Xu and Ping Wang, *Adversarial Noise Layer: Regularize Neural Network By Adding Noise* https://arxiv.org/pdf/1805.08000.pdf 2018.

[5] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville and Yoshua Bengio, *Generative Adversarial Nets* https://arxiv.org/pdf/1406.2661.pdf 2014.

[6] Fathma Siddique1, Shadman Sakib2 and Md. Abu Bakr Siddique, *Recognition of Handwritten Digit using Convolutional Neural Network in Python with Tensorflow and Comparison of Performance for Various Hidden Layers* https://arxiv.org/ftp/arxiv/papers/1909/1909.08490.pdf 2019.