Auswirkungen eines Hintergrundes auf die Identifikation von Objekten in Bildern mithilfe eines Convolutional Neural Network

Alexandra Zarkh, Sui Yin Zhang,
Lennart Leggewie und Alexander Schallenberg

Hochschule Bonn-Rhein-Sieg, Fachbereich Informatik, D-53757

7. Januar 2022

Inhaltsverzeichnis

1 Zusammenfassung

2 Einleitung

2.1 Verwendete Literatur

2.2 Theorie

2.3 Fragestellung

Die Hauptfrage ist nun: Welche Auswirkungen hat ein Hintergrund eines Bildes auf die Kosten der Kalkulation eines Convolutional Neural Network (CNN), welches ein Objekt im Vordergrund des Bildes erkennen und identifizieren soll?

3 Methoden

3.1 Convolutional Neural Networks

3.2 Implementation

3.2.1 ConvNet

Die Klasse *ConvNet* soll ein Convolutional Neural Network darstellen und erbt daher von der im Projektbericht [1] unter Abschnitt 2 genannten Klasse *Network*, was jener die gleichen Eigenschaften verleiht. Des Weiteren besitzt sie eine *java.util.ArrayList* vom Generic-Typ *TrainData* und einen *ImageAdapter* als private konstante Attribute. Erstere dient dem Speichern von Trainingsdatensätzen, die dann mit dem Aufrufen der überladenen Methode *train(double, int)* benutzt werden. Diese ruft die originale Instanz-Methode *train(double[][], double[][], double, int)* der Klasse *Network* mit den Daten aus der *ArrayList* auf. Außerdem bietet die Klasse *ConvNet* die Instanz-Methode *addTrainData(String, int[])* um dieser Trainingsdatensätze hinzuzufügen. Diese nimmt einen Dateinamen von einem Bild und einen ganzzahligen Zielvektor. Zwei private nicht-statische Hilfsmethoden *getImages()* und *getTargets()* dienen der besseren Handhabung der *ArrayList*.
Die Klasse erstellt durch den Konstruktor *ConvNet(String, int, int, int, int...)* (Pfad zum Ordner mit den Bildern, gewünschte Bildweite, -höhe, Anzahl der Output-Neuronen, Anzahl der Hiddenlayers und deren Neuronen) von ihr abgeleitete Objekte.

3.2.2 ImageAdapter

3.2.3 TrainData

Das Record *TrainData(int[], int[])* besitzt einen Vektor mit den RGB-Werten eines Bildes und einen Zielvektor. Außerdem hat es zwei Instanz-Methoden *getImageAsDoubleArray()* und *getTargetAsDoubleArray()*, welche die jeweiligen ganzzahligen Vektoren als Gleitkommazahl-Vektoren zurückgibt, und eine private nicht-statische Hilfsmethode *getIntAsDoubleArray(int[])*.

3.2.4 Test-Dateien

Für das Testen wurden ein Enum *RoadSignLabel* und eine Klasse *RoadSignTest* implementiert. Das Enum zählt die möglichen Ergebnisse des Netzes auf, indiziert sie und kann den jeweils richtigen Zielvektor durch die Instanz-Methode *getTarget()* zurückgeben. In der Klasse befindet sich die für die Forschungsfrage relevante Main-Methode des Programms. Diese ruft eine private statische Methode *test()* in der selben Klasse auf. In dieser wird ein *ConvNet* deklariert und initialisiert, mit Datensätzen bestückt und trainiert. Dabei werden einige hilfreiche Ausgaben gemacht.

3.3 Tests

4 Ergebnisse

5 Grafiken & Tabellen

Schildbilder im Trainingsdatensatz	
Andreaskreuz	5
Fußgängerüberweg	6
Gefahrenstelle	2
Vorfahrt gewähren	5
Vorfahrtsstraße	6
Verbot der Einfahrt	8
Fußgängerweg	8
Stopp	4

(1)

Anzahl Neuronen im CNN					
Input	**Hidden 1**	**Hidden 2**	**Hidden 3**	**Hidden 4**	**Output**
4096	64	64	64	64	8

(2)

Bildgröße	
Höhe	**Breite**
64	64

(3)

Training	
Lernrate	**Wiederholungen**
0.12	100000

(4)

6 Diskussion

7 Literatur

[1] L. Leggewie, A. Schallenberg, A. Zarkh und S. Y. Zhang, "Neuronale Netze Projektbericht," 2021, unpublished.

8 Anhang