Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Julian Magierski

Klassifizierung von Fahrradwegen mit Hilfe von Faltungsnetzwerken

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Julian Magierski

Klassifizierung von Fahrradwegen mit Hilfe von Faltungsnetzwerken

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Bachelor of Science Technische Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr.-Ing Andreas Meisel
Zweitgutachter: Prof. Dr. Wolfgang Fohl

Eingereicht am: 30. August 2018

Julian Magierski

Thema der Arbeit
Klassifizierung von Fahrradwegen mit Hilfe von Faltungsnetzwerken

Stichworte
Maschinelles Lernen, Faltungsnetzwerke, Smartphone Sensoren, Klassifizierung von Fahrradwegen, Bodenbeschaffenheit, Fuzzy Logik

Kurzzusammenfassung
Das Thema der Bachelorarbeit ist die Klassifizierung von Radwegen nach deren Gleichmäßigkeit. Zur Datenerfassung wurde ein Fahrrad mit einem montierten Smartphone genutzt. Mit dessen Hilfe wurden Bilder von befahrenen Oberflächen aufgenommen. Diesen Bildern wiederum wurden Messwerte von dem Smartphone Sensoren zugeordnet. Ziel ist es mit einem Faltungsnetzwerk Bildaufnahmen zu klassifizieren. In der Arbeit wurde eine Methode untersucht bei der die Trainingsdaten manuell per Hand gelabelt wurden und eine mit regelbasierten Labeln der Daten mit Fuzzy Logik. Die Grenzen sowie Möglichkeiten der Verfahren wurden anschließend diskutiert.

Julian Magierski

Title of the paper
Classification of cycle paths using convolutional networks

Keywords
Machine 1earning, convolutional neural networks, Smartphone Sensors, classification of cycle pathes, condition of the soil, fuzzy logic

Abstract
The topic of this bachelor thesis is the classification of cycle paths into quality classes. A smartphone installed on a bicycle was used for data collection. These were used to take images of travelled surfaces. The objective is to labeling images for convolutional networks. In this thesis a method was investigated in which the images were manually labelled by hand and one with rule-based labels of the data with fuzzy logic. The limits and possibilities of the methods were then discussed.

Inhaltsverzeichnis

1 Einleitung

Beim Fahrradfahren auf Radwegen wird Energie verbraucht. Der Kraftaufwand nimmt zu, wenn die Oberflächenqualität abnimmt [7]. Für Radfahrende gibt es deshalb, bei der Planung von einer Route zu einem Zielort, verschiedene Kriterien zu bedenken. Ein Alltagsfahrer möchte, beispielsweise möglichst schnell zu einem Zielort gelangen. Hingegen achtet ein Freizeitfahrer eher auf die Sicherheit und Qualität der Route. Für die Planung der Infrastruktur des Radverkehrs und Sanierung von Radwegen werden aktuelle Daten über die Fahrbahnbeschaffenheit benötigt [16]. Deshalb ist es, interessant den Zustand von Radwegen, Gehwegen und Straßen zu überwachen.

Auf OpenStreetMap OSM oder OpenCycleMap OCM können Nutzer Routen planen. Die Attribute, über die Beschaffenheit der Radwege, werden von den OSM-Nutzern eingetragen. Ziel dieser Bachelorarbeit ist es in OSM Fahrradwege nach deren Oberflächenqualität zu taggen (siehe Abb. 1.1). Zu diesen Zweck wurde ein Smartphone an einem Fahrrad montiert, um Bilder und Sensormesswerte während der Fahrt zu sammeln. Mit diesen Daten wurde dann ein Faltungsnetzwerk (engl. convolutional neural networks CNN) [11] trainiert. Zu den wichtigsten Bestandteilen eines Faltungsnetzwerks gehören Faltungsschichten (engl. convolutional layer). Diese sind zwei dimensional und dadurch, unter anderem, gut geeignet um Bilder zu klassifizieren. Ein Neuron einer Faltungsschicht ist nur mit einem lokalen rechteckigen Ausschnitt mit den Neuronen der vorherigen Faltungsschicht verbunden.

Orientiert wurde sich dabei an einer bereits existierenden wissenschaftliche Arbeit [13], die sich mit einem ähnlichen Ansatz beschäftigt. In der Arbeit von Fraunhofer-Institut für Intelligente Analyse- und Informationssysteme IAIS wurde auch ein Smartphone an einem Fahrradlenker montiert. Dabei wurde wiederholt eine Route innerstädtisch befahren. Zu den erfassten Merkmalen mit dem Smartphone gehörten auch die 3-Achsen-Beschleunigungsmesswerte und GPS Koordinaten. Mit dem Algorithmus Naive Bayesian konnte eine Accuracy von 77.46% bei der Klassifizierung von Wegabschnitten einer Länge von $20m$ erzielt werden. Dabei wurde zwischen den Klassen "gleichmäßig", ünebenünd "rauünterschieden, wobei die Trainingsdaten manuell gelabelt wurden. Von den Testdaten wurden $85,59\%$ der Klasse "gleichmäßig"korrekt vorhergesagt und 53.95% der Klasse üneben", sowie 63.34% der Klasse "rau

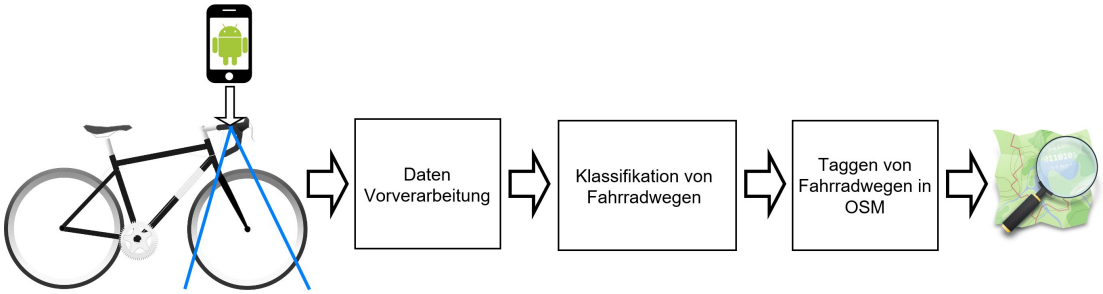

Abbildung 1.1: Funktionale Architektur

Quellen: Smartphone icon, The Noun Project, To Uyen, Lizenz: CC BY 3.0 US
Android Robot, Android Brand Guidelines, Google Inc., Lizenz: CC BY-SA 3.0
Logo des OpenStreetMap-Projekts, https://wiki.openstreetmap.org/wiki/File:Public-images-osm_logo.svg, Ken Vermette, CC BY-SA 3.0

zugeordnet. In einem weiteren Experiment wurde eine Klassifizierung zwischen den Klassen üneben"beziehungsweise SSchlaglochünd "nicht uneben"jeweils zwischen zwei GPS Koordinaten vorgenommen. Dabei konnte mit 441 Testdaten eine Accuracy von 98.19% erreicht werden. Eine ähnliche Thematik behandelt die Klassifizierung der Oberfläche von Straßen, bei welcher Smartphones in Automobilen befestigt werden, um Sensormesswerte zu erfassen. Einen Überblick bietet [12].

Diese Arbeit behandelt zunächst über die Grundlagen in Kapitel 2. In diesem Kapitel wird der Ablauf der Datenerfassung, sowie deren Vorverarbeitung, beschrieben. Das folgende Kapitel 3 befasst sich mit der Klassifizierung der erfassten Daten von 2000 manuell gelabelten Bildern. Dabei wurde zwischen unbefestigten und befestigten Wegen unterschieden.

Um einen größeren Datensatz zur Verfügung zu haben und die Daten Anhand der Oberflächenbeschaffenheit zu unterscheiden, wurden die Daten in Kapitel 4 mit Fuzzy Logik gelabelt. Mit zwei verschiedenen Datensätzen wurden dabei Faltungsnetzwerkversuche durchgeführt. Der erste Datensatz, bestehend aus 115.892 Bildern, wurde ohne Wiederholung der befahrenen Strecke erfasst. Im zweiten Datensatz, von 120.421 Bildern, wurde eine Route wiederholt befahren. Die Ergebnisse der Versuche werden in Kapitel 5 diskutiert.

2 Grundlagen

2.1 Fahrradwege Monitoring App für Android

Für die Datenerfassung wurde eine Android™ App programmiert mit den Namen Fahrradwege Monitoring App FMA. Mit diesem System können kontinuierlich Bilder aufgenommen und Sensordaten erfasst sowie abgespeichert werden. Als Smartphone wurde ein Nokia 6 mit Android verwendet. Hierbei wurde die App optimiert für die Android Sdk API 26 (minimal 24) und Android OS Oreo (8.0). In Abb. 1.1 ist zu sehen, dass das Smartphone bei den aufgenommen Daten bzw. Datensätzen an der rechten Seite des Lenkers montiert war. Für den Benutzer ist so eine bequeme Bedienung der App möglich. Ein Nachteil ist der relativ große Abstand vom Lenker zum Boden mit ca. $\approx 97cm$.

2.1.1 Smartphone Sensordaten

In der Tab. 2.1 sind die Messwerte beschrieben, welche zu einer Bildaufnahme erfasst wurden. Dabei ist zu beachten, dass die Arten von erfassten Messwerten ab Datensatz 28 abweichen. Dazu wurden noch Zeitstempel aufgenommen und die Unixzeit in ms als ID-Nummer, wenn ein Bild abgespeichert wurde. Für die Aufnahme der Bilder wurde die Android Camera2 API genutzt.

Messwert	Sensor
Geschwindigkeit in km/h	GPS
Breitengrad in Dezimalgrad	GPS
Längengrad in Dezimalgrad	GPS
X,Y,Z-Achsen Beschleunigung in m/s^2	3-Achsen Beschleunigungssensor
Roll-Nick-Gier Winkel in rad	3-Achsen Beschleunigungssensor und Magnetometer

Tabelle 2.1: Messwerte der Fahrradwege-Monitoring-App

In der Abb. 2.1 ist die generelle Sensor Orientierung eines Android Smartphones zu sehen. Diese besteht aus den 3-Achsen X, Y und Z. Zusätzlich sind noch der Roll-Nick-Gier-Winkel eingezeichnet. Der Roll-Winkel befindet sich auf der X-Achse und der Gier-Winkel auf der Z-Achse.

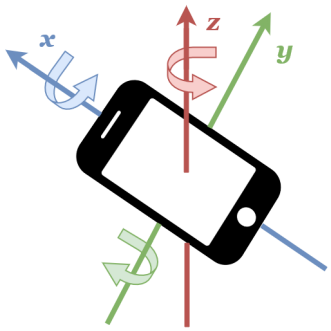

Abbildung 2.1: Android Smartphone Sensororientierung

Quelle: Smartphone icon, The Noun Project, To Uyen, verändert, Lizenz: CC BY 3.0 US

2.1.2 Kalibrierung

Über die Benutzerschnittstelle der Anwendung wurde vor dem Start der Datenerfassung eine Kalibrierung ausgeführt. Diese wurde im Stand durchgeführt. Bei dieser ging es darum, dass Smartphone am Lenker des Fahrrads auszurichten. Dafür sollte der Nick-Winkel $\approx 0^\circ$ betragen. Die Gradzahl wurde über dem Display dem Benutzer mitgeteilt. Zudem wurden auch die Y und Z-Achsen Beschleunigungsmesswerte sowie Nick-Winkel Werte erfasst. Von diesen Werten wurde der Mittelwert berechnet und als Offset bei der Datenerfassung subtrahiert. Damit diese Werte im Stand ≈ 0 sind.

2.1.3 Zeitverhalten

In diesem Abschnitt werden die zeitlichen Anforderungen der Datenerfassung beschrieben. Dazu gehört die Dauer des Zeitfensters t in welchem zu einer Bildaufnahme Daten erfasst wurden.

2.1.4 Überblick der Messwerterfassung

Die Abb. 2.2 zeigt die Mittelwerte des zeitlichen Ablaufs der Datenerfassung für die Datensätze 37 bis 42. Für weitere Informationen zu den Datensätzen siehe Abschnitt 4.2. Bevor das erste Bild geschossen werden kann wurde die Messwerterfassung gestartet. Zu der Messwerterfassung gehören die Messwerte der Z-Achsen und Y-Achsen Beschleunigung, sowie Nick Winkel. Die Sensoren wurden mit einer Frequenz von 125 Hz ($T = 8ms$) abgetastet. Die Wartezeit bevor das erste Bild nach dem Start der Anwendung aufgenommen wurde betrug $250ms$. Damit wurde sichergestellt, dass vor einer Aufnahme ausreichend Messwerte erfasst wurden. Nach jeder

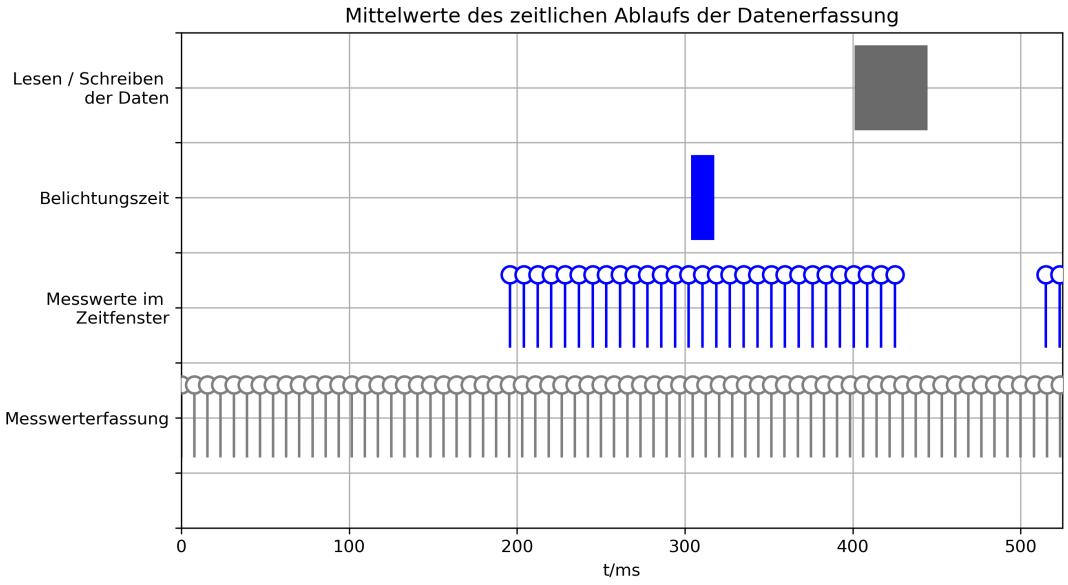

Abbildung 2.2: Mittelwerte des zeitlichen Ablaufs der Datenerfassung für die Datensätze 37-42

hundertsten Bildaufnahme wurden die Listen, in welchen die Messwerte zwischengespeichert wurden, geleert. Zu einer Bildaufnahme wurden nur die Daten innerhalb eines Zeitfensters in eine CSV Datei geschrieben. Die Bildrate für die Datensätze betrug 3.13 Bilder pro Sekunde.

2.1.5 Dauer der Messwerterfassung

Das Verhältnis zwischen zurückgelegter Strecke in m, während Messwerte mit dem Smartphone Sensoren erfasst werden und der Geschwindigkeit in km/h, ist abhängig von der Dauer des Zeitfensters in ms. Die Annahme ist, dass bei der Wahl eines dynamischen Zeitfensters die zurückgelegte Strecke l auf einem Meter normiert werden kann siehe G. 2.2. Zudem wurde ein statisches Zeitfenster der Dauer $t = 500ms$ und $t = 80ms$ (siehe G. 2.1) untersucht.

$$l = v * t \Rightarrow m = \frac{m}{s} * \frac{Zeitfenster\ t\ in\ ms}{1000} \tag{2.1}$$

$$l = v * \frac{l}{v} \Rightarrow m = \frac{m}{s} * \frac{1m}{\frac{m}{s}} \tag{2.2}$$

Beispiel G. 2.2: Geg.: Geschwindigkeit $v = 20km/h$

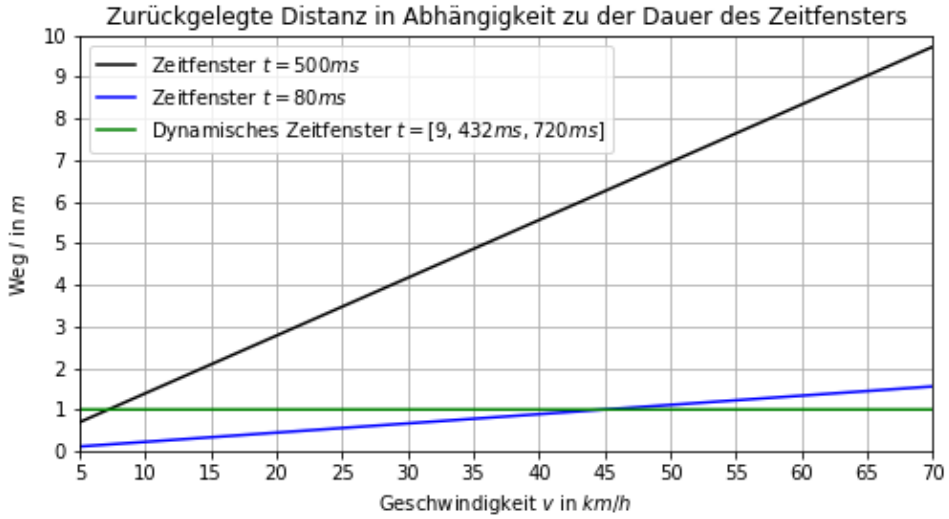

Abbildung 2.3: Darstellung des zurückgelegten Wegs in Abhängigkeit der Geschwindigkeit und der Dauer des Zeitfensters

$$m = \frac{18\frac{km}{h}}{3.6} * \frac{1m}{\frac{18\frac{km}{h}}{3.6}}$$

$$m = 5\frac{m}{s} * 0.2s = \underline{1m}$$

Bei der Berechnung des dynamischen Zeitfensters wurde berechnet wie viel ms nötig sind um $1m$ Weg zurück zu legen (siehe G. 2.3).

$$t = \frac{l}{v} \Rightarrow ms = \frac{1m}{\frac{m}{s}} * 1000 \tag{2.3}$$

Auf Abb. 2.3 ist zu sehen wie sich die Wahl der Dauer des Zeitfensters t und die Geschwindigkeit v auf die Länge l des zurückgelegten Wegs auswirkt. Die X-Achse zeigt die Geschwindigkeit v in km/h und die Y-Achse den zurückgelegten Weg l in m. Zu sehen sind drei lineare Funktionen, die abhängig sind von der Wahl der Zeit t des Zeitfensters. Die Idee hinter den statischen Zeitfenstern ist der triviale Ansatz, sodass keine Berechnung, welche abhängig von der Geschwindigkeit nötig ist. Ein Zeitfenster mit einer Dauer von $t = 500ms$ bietet den Vorteil mehr Messwerte in einer längeren Wegstrecke zu erfassen und damit Muster besser zu erkennen z.B. Pflasterfugen zwischen Gehwegplatten. Mit einem statischen Zeitfenster t von $500ms$ steigt der zurückgelegte Weg linear an. Die Steigung m beträgt $m = \frac{\Delta y}{\Delta x} = \frac{y_1 - y_0}{x_1 - x_0} \Rightarrow \frac{5-2,5}{36-18} = 0,938$. Dagegen hat ein statisches Zeitfenster von 80ms eine

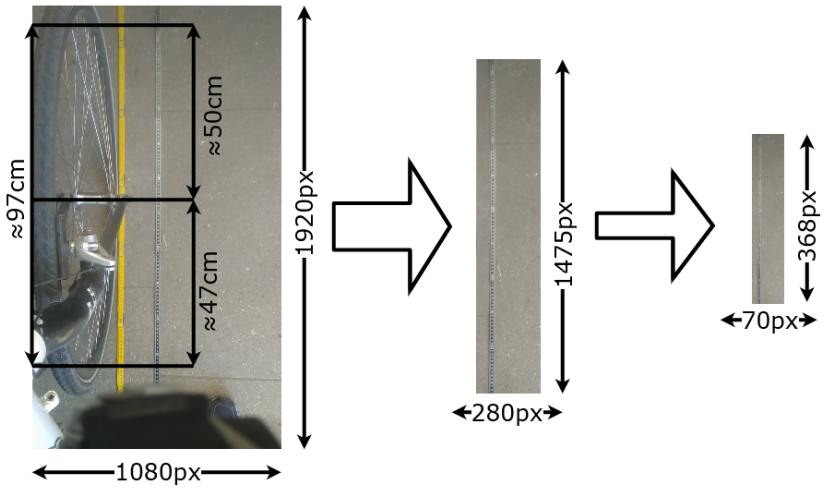

Abbildung 2.4: Bildausschnitt mit der Nutzung des dynamischen Zeitfensters

kleinere Steigung von $m = \frac{\Delta y}{\Delta x} \Rightarrow \frac{0{,}8-0{,}4}{36-18} = 0,022$. Dies bedeutet das mit einem Zeitfenster von $80ms$ die zurückgelegte Strecke relativ klein ist.

Der verwendete Bildausschnitt zeigt ca. $1m$ ($\approx 97cm$) Weglänge siehe Abb. 2.4. Die untere Begrenzung auf $\approx\ 47cm$ ist nötig, weil auf einigen Bildern der Schuh des Radfahrers zu sehen ist. Hiermit werden unwichtige Informationen für das Faltungsnetzwerk vermieden. Mit einem Zeitfenster von $80ms$ bei Geschwindigkeiten kleiner als $v = 45km/h$ werden weniger Sensormesswerte erfasst als Wegfläche auf den Bildausschnitt zu erkennen ist. Der Vorteil bei einem kleinen Zeitfenster ist die Bildrate von ca. $3,4$ Bildern pro Sekunde. Hieraus entstand die Motivation ein dynamisches Zeitfenster zu verwenden, um die Messwerte dem Bild zuordnen zu können.

Die Dauer des dynamischen Zeitfensters berechnete sich aus der G. 2.3. Das heißt, die Dauer t ist Abhängigkeit von der aktuellen Geschwindigkeit v des Fahrrads. Wie auf Abb. 2.3 zu sehen entspricht der zurückgelegte Weg dann immer $1m$. Dafür muss die Voraussetzung erfüllt sein, dass die aktuelle Geschwindigkeit möglichst genau ist. Um die aktuelle Geschwindigkeit zu erhalten wird die Android Location API genutzt sowie der Location Manager. Der Location Manager aktualisiert die Aktuelle GPS Lokation so schnell wie möglich. Dabei wird in der API nicht beschrieben mit welcher Frequenz. Ein weiteres Problem ist, dass nicht angegeben wird wie die Geschwindigkeit gemessen wird. In dem Buch [2] wird beschrieben, dass meistens der Doppler-Effekt für die Berechnung der Geschwindigkeit genutzt wird.

Als negatives Beispiel für die Messung der Geschwindigkeit mit dem Andorid Location Manager wird angenommen, dass die aktuelle Geschwindigkeit $v = 20km/h$, aber die gemessene

$16km/h$ beträgt. Dann ist das Zeitfenster der gemessenen Geschwindigkeit $t = 228ms$, wobei das eigentliche Zeitfenster $180ms$ dauern müsste. In diesem theoretischen Beispiel werden also Sensormesswerte erfasst für den zurückgelegten Weg von $m = \frac{20}{3,6} * 0.228s = 1.2667$. Also ist es möglich, dass irrelevante Informationen zu dem Bildausschnitt erfasst werden können. Die Android Location API ermöglicht es auch die Genauigkeit der aktuellen Geschwindigkeit anzugeben. Dazu wird die Methode getSpeedAccuracyMetersPerSecond() genutzt. Diese gibt einen Gleitkommazahl zurück. Mit 68% Wahrscheinlichkeit befindet sich dann die berechnete Geschwindigkeit in diesem Bereich mit der Abweichung der Gleitkommazahl. Hat beispielsweise der Aufruf der Methode einen Rückgabelwert von 2 ergeben und die berechnete Geschwindigkeit ist $20km/h$, dann liegt die wahre Geschwindigkeit mit 68% Wahrscheinlichkeit in dem Bereich $18 - 22km/h$.

Die Dauer des dynamischen Zeitfensters ist beschränkt auf das abgeschlossene Intervall. Die obere Einschränkung ergibt sich daraus, dass die minimal gemessene Geschwindigkeit für die Aufnahme von Bildern $5km/h$ betragen muss. Dann ergibt sich ein Zeitfenster $t = 720ms$. Während das Fahrrad steht wurden keine Bilder aufgenommen, um die Klassifizierung mit einem Faltungsnetz zu erleichtern. Die untere Grenze ist nur eine theoretische Grenze. Bei einer Geschwindigkeit von $v = \frac{l}{t}km/h = \frac{1m}{9ms} * 3.6382km/h$ wäre dieser Schwellwert erreicht. Eine Geschwindigkeit die mit einem Fahrrad wahrscheinlich nicht erreicht werden kann. Für das verwendete Smartphone Nokia 6 beträgt die minimale mögliche Belichtungszeit $9432ns$. Diese Belichtungszeit ist auch der Schwellwert für die minimale Dauer des Zeitfensters.

Als Alternative zu der Nutzung der Location API kann ein externer Geschwindigkeitssensor genutzt werden. Denkbar wäre ein Gerät mit einer Bluetooth Schnittstelle. Ein komplett anderer Ansatz wäre ein Embedded System zu entwickeln. Dieses System könnte mit einer Kamera verbunden sein, die am Unterrohr des Fahrrads befestigt ist. Auch Echtzeitanforderungen könnten besser erfüllt werden als mit einem Android Smartphone. Die Entwicklung ist allerdings komplexer und ein Smartphone ist ein kompaktes System mit vielen Sensoren, welches relativ einfach an einem Fahrrad montiert werden kann.

2.1.6 Anzahl der Sensormesswerte

Auf Abb. 2.5 ist die Anzahl der Smartphone Sensormesswerte, abhängig von der Geschwindigkeit v und der Dauer t des Zeitfensters zu sehen. Bei einem statischen Zeitfenster ist die Anzahl der Messwerte konstant. In Gegensatz zu dem dynamischen Zeitfenster, bei welchem die Anzahl der Messwerte exponentiell abnimmt. Mit einem dynamischen Zeitfenster beträgt die Anzahl der Sensormesswerte ca. 90 bei einer Geschwindigkeit von $5km/h$. Über den Geschwindigkeitsbereich von $5 - 20km/h$ nimmt die Anzahl der Sensormesswerte schnell ab.

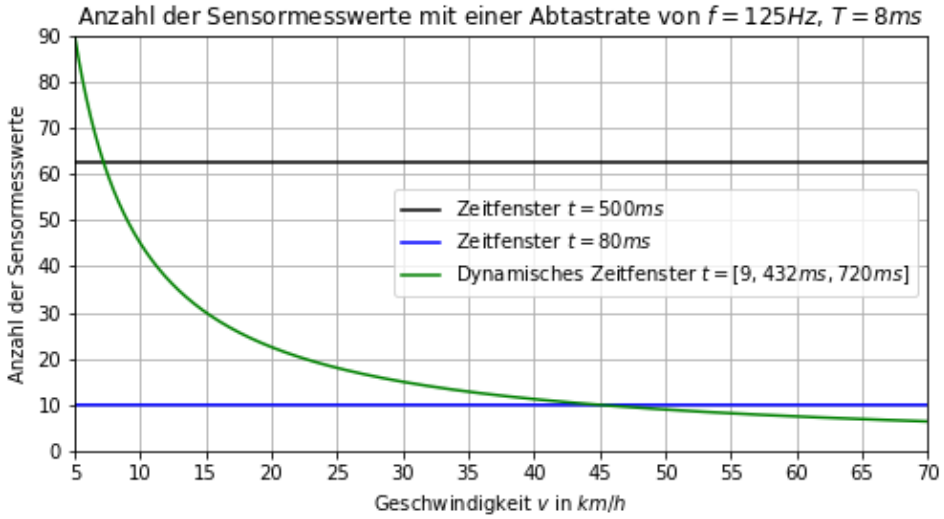

Abbildung 2.5: Anzahl der Messwerte abhängig von dem Zeitfenster t und der Geschwindigkeit v

Nach $45km/h$ gehen viele Informationen über die Oberflächenbeschaffenheit der Fahrradwege verloren, dass die Qualität bei der Auswertung der Daten fraglich ist.

2.1.7 Dynamisches Zeitfenster und das Weg-Zeit-Gesetz

Bei der Berechnung des dynamischen Zeitfensters wird von einer gleichförmigen Bewegung ausgegangen. Damit ist eine gleichbleibende Geschwindigkeit gemeint. Dies trifft beim Fahrradfahren zum Teil während der Fahrt zu. Wenn der Fahrradfahrer das Fahrrad allerdings beschleunigt oder abbremst, dann ist keine gleichförmige Bewegung gegeben. Auch Unebenheiten können die Geschwindigkeit reduzieren. Es ändert sich also die positive oder negative Beschleunigung a.

$$s = \frac{1}{2}at^2 + v_0t + s_0 \tag{2.4}$$

Hier wird die Überlegung betrachtet über das Weg-Zeit-Gesetz der gleichmäßig beschleunigten Bewegung (siehe G. 2.4) das dynamische Zeitfenster zu berechnen. Dabei steht s für die zurückgelegte Strecke, a für die Beschleunigung, v_0 die Anfangsgeschwindigkeit, t Zeit und s_0 für den Anfangsweg. Die G. 2.4 muss umgestellt werden auf t.

Diese Herangehensweise hat den Nachteil, dass das Fahrrad auch nicht gleichmäßig beschleunigt. Dazu kommt auch noch die Problematik die Beschleunigung und Anfangsgeschwindigkeit zu messen. Die Beschleunigung a kann mit der Y-Achse des Beschleunigungssensors des Smartphones gemessen werden. Zu beachten ist, wenn das Fahrrad und somit das Smartphone geneigt sind also der Nick Winkel nicht $0°$ beträgt dann nehmen auch andere Beschleunigungsrichtungen Einfluss auf die Y-Achsen Beschleunigung a.

Fazit

Ein dynamisches Zeitfenster hat gegenüber einem statischen Zeitfenster den Vorteil, dass während der Dauer t des Zeitfensters immer ungefähr die gleiche Strecke zurück gelegt wird. In diesem Fall der Datenerfassung werden Sensormesswerte für den zurückgelegten Weg von $1m$ erfasst. Dies entspricht, mit einer Abweichung von $3cm$, auch dem erfassten Bildausschnitt. Das dynamische Zeitfenster hat den Nachteil, dass die Berechnung von der Genauigkeit der gemessenen Geschwindigkeit abhängig ist. Als Alternative kann ein externer Sensor verwendet werden, der z.B. mit Bluetooth genauere Geschwindigkeitsangaben liefert als die Berechnung über die Android Location API bzw. die Messung über GPS.

2.2 Vorverarbeitung

Die Vorverarbeitung besteht im ersten Schritt darin die Merkmale mit den erfassten Messwerten zu den Bildern in eine Character-separated-values CSV Datei zu konkatenieren. Um die Anzahl

der Daten zu verwalten werden jeweils 2000 Bilder und die dazugehörigen Messwerte in einen Ordner geschrieben. Als nächstes wird die Standardabweichung der Z-Achsen Beschleunigung siehe G. 2.5 berechnet. Wobei N für die Anzahl der Messwerte, μ der Mittelwert über alle Messwerte und x_i ein Messwert darstellt. Die Standardabweichung ist die Wurzel aus der Varianz, welche die mittlere, quadratische Abweichung der Messwerte beschreibt.

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2} \tag{2.5}$$

Die Bildaufnahmen werden zugeschnitten und skaliert, um die Berechnungszeit und Hauptspeicher Bedarf der Faltungsnetze beim Trainieren zu reduzieren (siehe Abb. 2.4). Des Weiteren wurden Überbelichtete und unterbelichtete Bilder entfernt.

2.2.1 Entfernen von überbelichteten und unterbelichteten Bildern

Einige der Bildaufnahmen sind überbelichtet oder unterbelichtet. Dies kommt vor, wenn sich die Lichtverhältnisse der Umgebung schnell änderten. Wenn z.B. die Sonne zwischen Bäumen schien, dann konnten Bilder entstehen die überbelichtet waren. Bei der Fahrt in einem Tunnel konnten wiederum Bilder unterbelichtet sein. Die Belichtungszeit wurde automatisch berechnet mit der Android Camera2 API. Bevor ein Bild zur weiteren Verarbeitung aufgenommen wird überprüft mit einer Probeaufnahme, ob die Belichtungszeitparameter gut sind. Es gibt keine Garantie, dass bei der nächsten Bildaufnahme die Belichtungszeit noch korrekt eingestellt ist.

Das Ziel ist es möglichst viele Bilder mit einer kurzen Belichtungszeit ohne großer Bewegungsschärfe zu erfassen. Hierfür wurde in der App die Belichtungszeit Bildrate CaptureRequest.CONTROL_AE_TARGET_FPS_RANGE auf 30 Bilder pro Sekunde eingestellt. Der Kamera Szenenmodus wurde auf Action für bewegende Objekte ausgewählt. Die Belichtungszeit statisch manuell festzulegen ist kompliziert, weil damit auch der ISO Wert und die Bildrate mit angegeben werden muss. Die Lichtverhältnisse ändern sich während der Fahrt und mit einer festen Belichtungszeit könnten ganze Bildreihen unter-oder überbelichtet werden.

Versuchsbeschreibung

Für einen Datensatz werden alle zugeschnittenen Bilder in ein Python Programm geladen. Anschließend wird das Histogramm der Grauwerte berechnet (siehe G. 2.6). Dabei entspricht g einem Grauwert und n_g der Anzahl der Bildpunkte mit diesem Grauwert.

$$h(g) = n_g \tag{2.6}$$

$$\mu = \frac{\sum_{i=0}^{n} g_i * h(g_i)}{\sum_{i=0}^{n} *h(g_i)} \tag{2.7}$$

Als nächstes wird der Mittelwert μ der Grauwerte, also die Summe aller Grauwerte, durch die Anzahl der Bildpunkte für jedes Bild berechnet (siehe G. 2.7) . Ein Grenzwert wird dann ausgewählt, um zu entscheiden ob ein Bild unter-oder überbelichtet ist. Falls dies für ein Bild zutrifft wird dieses, sowie die dazugehörigen Einträge, in der merkmale.csv Datei gelöscht.

Versuchsdurchführung

Beispiele für unterbelichtete Bilder sind auf Abb. 2.6 zu sehen. Auf der Abb. 2.7 sind überbelichtet Bilder zu sehen aus dem Datensatz 41. Als oberer und unterer Schwellwert werden die obersten und untersten $\approx 19,5\%$ für μ ausgewählt (siehe G. 2.8). Ein Bild gilt demnach als überbelichtet, wenn der Mittelwert der Grauwerte größer als 204 ist. Als überbelichtet gilt ein Bild bei einem mittleren Grauwert kleiner als 50. Für den Datensatz 41 traf dies auf $0,69\%$ der Bildaufnahmen zu. Hierbei existieren insgesamt 5940 Bilder für diesen Datensatz.

$$p = \frac{100 * W}{G} \Rightarrow p = \frac{100 * 50}{256} = 19,53125\% \tag{2.8}$$

Der Prozentsatz der Bilder die unter- oder überbelichtet sind betrug für die Datensätze 37 bis 42 gleich $0,088\%$.

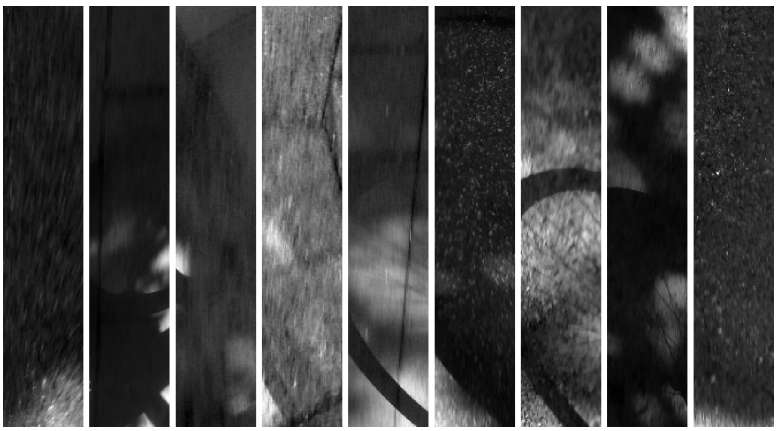

Abbildung 2.6: Bilder die unterbelichtet sind aus dem Datensatz 41

Abbildung 2.7: Überbelichtete Bilder aus dem Datensatz 41

2.2.2 Ergebnis

Wie auf Abb. 2.6 und 2.7 zu sehen ist, ist mit der Berechnung des Mittelwertes G. 2.7 eine Einteilung der Bilder in unter- bzw. überlichtete und normal belichtete Aufnahmen möglich. Ein Nachteil bei diesem Ansatz ist, dass der Durchschnitt alle Grauwerte betrachtet wird. Eine störende Belichtung in einem Bildbereich wird somit nicht ermittelt.

3 Manuelles Labeln

Einleitung

Ziel dieses Versuchs ist die binäre Klassifizierung von kontinuierlich aufgenommenen Bildern von Radwegen. Dabei soll zwischen den Klassen “befestigt” und “unbefestigt” unterschieden werden. Als Referenz wurde das Schlüsselattribut “surface” von OSM genutzt, mit welchem Wege nach der Beschaffenheit der Oberfläche beschrieben werden. Grundsätzlich wird zwischen versiegelten Oberflächen und nicht versiegelten unterschieden. Zu den versiegelten oder auch befestigten Oberflächen gehören unter anderem Asphalt, Beton und Pflastersteine. Unbefestigte Wege werden unterschieden zwischen verdichtete Deckschichten, Split bzw. Grant, Schotter, Kies, Gras, Matsch, Sand, Schnee und weiteren Oberflächen. In diesem Versuch wurden die Bilder in “befestigt” und “unbefestigt” Radwege manuell gelabelt. Für Navigationssysteme ist die Klassifizierung interessant, weil auf befestigten Wegen ein schnelleres Fahren mit geringeren Kraftaufwand möglich ist.

3.1 Versuch 1.1: Faltungsnetzwerk Architektur

Einleitung

Als Vorlage für das Faltungsnetz dient ein Keras Blog Artikel [4]. Dieser wiederum hat als Vorlage die Faltungsnetz Architekturen Modelle von Yann LeCun [3]. Als High Level-API für neuronale Netzwerke wurde Keras verwendet.

Für das Finden von möglichst adäquaten Parametern für das Faltungsnetz in dem Versuch wurde Hyperas genutzt. Hyperas ist ein Wrapper um Hyperopt, welcher ermöglicht optimierte Parameter für Modelle mit Hypertuning zu finden. Zu den Parametern gehören, unter anderem, die Anzahl der Faltungsschichten und der Feature Maps, Methoden zur Regularisierung, Optimierungsalgorithmen und Aktivierungsfunktionen. Als Algorithmus für Hyperopt wurde bei dem Versuchen Random Search genutzt. Jeweils 10 Durchgänge und Epochen wurden vollzogen, um brauchbare Parameter mit einer hohen Accuracy zu finden. Als Epochen wird die Anzahl der Iterationen über die gesamten Trainingsdaten mit einem neuronalen Netz bezeichnet. Dabei wurde zudem Early Stopping genutzt. Eine Methode um das Training des Modells

abzubrechen, wenn sich die Accuracy, nach einer angegebenen Anzahl an Epochen nicht mehr erhöht. Durch Hyperparameter-Tuning soll ein besseres Ergebnis für die Metric-Accuracy erzielt werden. Die Accuracy (Genauigkeit) gibt den Anteil der korrekt klassifizierten Daten als Prozentsatz an.

Versuchsaufbau

Die verwendeten Bilder kommen aus dem Datensatz 22. Die Dauer des statischen Zeitfensters der Messwerterfassung dauert $80ms$ bei einer Abtastrate von $\approx$ 250 Hz. Aufgenommen wurden die Bilder im Naturschutzgebiet Höltigbaum. Für die beiden Klassen "befestigt" und "unbefestigt" gab es einen Ordner mit jeweils 1000 Bildern. Die Originalbilder hatten eine Auflösung von $1080px\ x\ 1920px$ und wurden auf $280px\ x\ 1448px$ zugeschnitten. im letzten Schritt wurde die Bildgröße auf $70px\ x\ 362px$ reduziert. Die Anzahl der Testbilder beträgt 400 und die Anzahl der Trainingsbilder 1600. Als Verfahren zur Klassifizierung sollten Faltungsnetze mit verschiedenen Parametern trainiert werden.

In den Versuchen von 1.1.1 bis 1.1.9 wurden unterschiedliche Parameter zum Optimieren des Faltungsnetzwerks mit Hyperas ausgewählt. Eine genauere Dokumentation der einzelnen Versuche befindet sich in dem Jupyter Notebook CNN_experiment1_binary_classifier.ipynb.

Mit dem Ziel eine Überanpassung (engl. overfitting) des neuronalen Netzes zu verhindern werden Regularisierungstechniken genutzt. Dies ist der der Fall, wenn die Genauigkeit der Trainings-Accuracy höher ist als die Test oder Validation-Accuracy. Bei den Versuchen 1.1.1 und 1.1.2 wurde nach einer guten Wahl der Dropout-Rate [9] gesucht. Dropout ist eine Regularisierungstechnik für Deep Neural Networks (DNN). Bei welcher mit einer Wahrscheinlichkeit p angegeben, Neuronen einer Faltungsschicht innerhalb eines Trainingsschritts ignoriert werden. Die Ausgangsneuronen können dabei nicht übergangen werden.

Eine Aktivierungsfunktion wurde im Versuch 1.1.3 aus den Funktionen ReLU, ELU [6], SeLU [15] und Sigmoid gesucht. Im Versuch 1.1.4 wurden die Optimierungsalgorithmen RMSprop, Adagrad [8], Adadelta [20] und Adam [14] ausprobiert. Eine Übersicht über die verschiedenen Algorithmen bietet der Block Artikel An overview of gradient descent optimization algorithms [18]. Des Weiteren wurden in dem Versuch 1.1.5 die Anzahl der Feature Maps optimiert. Untersucht wurde, welche Variante für das Auffüllen (Padding) der Randwerte geeigneter ist (siehe Versuch 1.1.6) für die nächste Faltungsschicht. Zur Auswahl standen die Parameter "valid" und "same". Mit "same" werden Randfelder mit Nullen aufgefüllt, sodass die Ausgabedimension gleich der Eingabedimension ist. Dagegen werden mit "valid" keine Randwerte aufgefüllt.

Bei dem nächsten Versuch 1.1.7 standen 2 bis 4 Faltungsschichten zur Auswahl. In Versuch 1.1.8 wurden die aussichtsreichsten Parameter der jeweiligen Versuche als zu optimierende

Werte ausgewählt. Anschließend wurde in Versuch 1.1.9 mit den besten gefundenen Parametern aus Versuch 1.1.8 ein Faltungsnetz mit 100 Epochen trainiert.

3.2 Versuch 1.2: Data Augmentation

Einleitung

Auch in diesem Versuch war die Einteilung in die Klassen, “befestigt” und “unbefestigt” von Fahrradwegen, mit Hilfe eines Faltungsnetzwerks das Ziel. Zur Regulierung der Überanpassung und besseren Generalisierung bei der Vorhersage des Modells aus Versuch 1.9 wurde Data Augmentation verwendet. Hierbei werden zusätzliche Trainingsbilder generiert mit Transformationen wie Rotation, Skalierung oder Spiegelung.

Hypothese

Mit der Vermehrung anhand von Data Augmentation der Trainingsdaten ist eine genauere Klassifizierung möglich als in Versuch 1.1. Unterstützt wurde die These durch den Artikel The effectiveness of data augmentation in image classification using deep learning [17]. In dem Artikel wurde untersucht, ob es möglich ist mit Data Augmentation einen kleinen Datensatz (wenige tausend Bilder) mit Labeln zu nutzen, anstelle eines Datensatz mit mehreren hunderttausend Bildern ohne Label, um ein Faltungsnetz zu trainieren.

Versuchsaufbau

Wie in Versuch 1.1 kommen die Daten aus dem Datensatz 22. Dabei wurde der gleiche Ausschnitt und die gleiche Skalierung der Bilder verwendet. Als Bibliothek für die Erzeugung der generierten Bilder wurde die Keras ImageDataGenerator Methode genutzt. Die generierten Bilder entstanden aus den Transformationen Rotation, vertikale sowie horizontale Verschiebung, Scherung, Zoom, horizontale und vertikale Spiegelung. Von Wichtig war, dass die generierten Bilder nicht als Testdaten genutzt wurden, um das Ergebnis nicht zu verfälschen.

Die Anzahl der generierten Bilder betrug 400 Stück pro Klasse in Versuch 1.2.1 und 200 in Versuch 1.2.2. In Versuch 1.2.3 bis 1.2.5 wurden unterschiedlich starke Transformationen bei der Erzeugung zusätzlicher Trainingsdaten mit 400 Bildern pro Klasse ausprobiert.

3.3 Versuch 1.3: Testdaten

Einleitung

In diesem Versuch ging es um die Evaluierung der trainierten Faltungsnetze aus den Versuchen 1.1 und 1.2 mit noch nicht gesehenen Bildaufnahmen von Radwegen. Es ging darum zu testen wie gut die gefundenen Modelle generalisieren können.

Versuchsaufbau

Die zu klassifizierenden Bildaufnahmen stammen aus den Datensatz 22, 15 und 23. Das Wetter bei der Erfassung des Datensatzes 22 war sonnig. Das heißt auch, dass auf vielen Aufnahmen Schatten wie beispielsweise von Bäumen zu sehen sind. Im Gegensatz dazu sind auf den Bildern aus dem Datensatz 15 keine Schatten zu sehen, weil der Himmel bedeckt war. Datensatz 22 und 15 entstammen einer ähnlichen Route im Höltigbaum an unterschiedlichen Tagen. Als Fahrrad wurde dafür ein 28 Zoll Crossrad genutzt.

Die Bilder aus Datensatz 23 wurden mit einem 26 Zoll Hardtail Mountainbike MTB mit gleicher Lenkerhöhe aufgenommen. Getestet wurden die ermittelten Modelle aus dem Versuchen 1.1.9, 1.2.3 und 1.2.4 siehe Tab. 3.1.

Versuch Nr.	Trainings-Accuracy	Trainings-Loss	Validation-Acc.	Validation-Loss
1.1.9	97.19%	0.0814	95.25%	0.2861
1.2.3	88.08%	0.2933	91%	0.2735
1.2.4	100%	0.0030	95.50%	0.3247

Tabelle 3.1: Modelle die getestet wurden mit Validierungsergebnissen

Versuchsbeschreibung

In dem Versuch 1.3.1 wurden 400 noch nicht gesehene Bilder von den Faltungsnetzen aus dem Datensatz 22 klassifiziert. Für die Test-Accuracy wurde eine ähnliche Accuracy von $\approx 95\%$, wie in den Versuchen 1.1.9 und 1.2.4, erwartet. Bei dem Modell 1.2.3 sollte die Test-Accuracy niedriger sein, aufgrund der geringeren Validation-Accuracy (siehe 3.1).

Anschließend wurden in Versuch 1.3.2 als Testbilder die Daten aus dem Datensatz 15 verwendet. Aus den Ergebnis von Versuch 1.3.1 wurde erwartet, dass das Modell 1.2.3 die beste Leistung nach der Metric-Accuracy erzielt.

Um den Test zu verbessern, wurden die Anzahl der Bilder aus dem Datensatz 15 in Versuch 1.3.3 auf 2000 Bilder erhöht. Interessant war zu ermitteln, ob die Test-Accuracy höher oder

niedriger ist als im Versuch 1.3.2 mit mehr Testbildern. Die Testbilder enthielten auch alle Bilder aus Versuch 1.3.2.

In dem Versuch 1.3.4 wurde das Smartphone an einem Hardtail Mountainbike MTB auf der linken Lenkerseite montiert. Aus Datensatz 23 wurden 2000 Bilder klassifiziert. Die Bildaufnahmen aus diesem Datensatz wurden nicht als Trainingsdaten genutzt. Dazu kommt auch, dass die Bilder mit einem MTB aufgenommen wurden und nicht mit dem Crossrad Fahrrad aus den vorherigen Versuchen. Deshalb wurde erwartet, dass die Test-Accuracy niedriger ist.

3.4 Ergebnisse

Ergebnis: Versuch 1.1

Nr.	Typ	Ausgabeform	Anzahl der zu trainierenden Parameter
1	Faltungsschicht (3x3 Faltungskerngröße)	(360, 68, 32)	896
2	Aktivierungsfunktion: ELU	(360, 68, 32)	0
3	Max-Pooling (2x2)	(180, 34, 32)	0
4	Dropout (5% Rate)	(180, 34, 32)	0
5	Faltungsschicht 2(3x3)	(178, 32, 32)	9248
6	Aktivierungsfunktion: ELU	(178, 32, 32)	0
7	Max-Pooling (2x2)	(89, 16, 32)	0
8	Dropout (33% Rate)	(89, 16, 32)	0
9	Faltungsschicht 3(3x3)	(87, 14, 64)	18496
10	Aktivierungsfunktion: ELU	(87, 14, 64)	0
11	Max-Pooling (2x2)	(43, 7, 64)	0
12	Dropout (10% Rate)	(43, 7, 64)	0
13	Flatten	(19264)	0
14	Fully Connected Layer	(64)	1232960
15	Aktivierungsfunktion: ELU	(64)	0
16	Dropout (46% Rate)	(64)	0
17	Fully Connected Layer	(2)	130
18	Aktivierungsfunktion: Softmax	(2)	0

Tabelle 3.2: Faltungsnetzwerk Architektur aus Versuch 1.1.9

Durch das Finden von geeigneten Parametern konnte das Modell in dem Versuch 1.1.9 (siehe Tab. 3.2) trainiert werden. Dieses Modell hatte zwar die höchste Validation-Accuracy von 95.25% und einen Fehler (engl. loss) von 0.2861, allerdings ist der Loss im Vergleich zu dem Ergebnis aus Versuch 1.1.4 mit 0.1752 niedriger gewesen. Der Loss beziehungsweise Kosten oder Fehler gibt den berechneten Loss einer Kostenfunktion an. Ziel ist es diesen Wert,

beim trainieren des Faltungsnetzwerks zu minimieren. Die Trainings-Accuracy betrug 97.19%, somit ist das Faltungsnetz aus Versuch 1.1.9 leicht überangepasst.

Der Versuch hat gezeigt, dass mit Hypertuning die Accuracy erhöht werden kann. In Versuch 1.1.1 betrug die Validation-Accuracy nur 0 und in Versuch 1.1.9 0. Den gesamten Suchraum, an möglichen Parametern, für das gesuchte Modell mit Hypertuning abzudecken, ist eine komplexe Aufgabe. Hyperas bietet hierfür die Möglichkeit die Modellparameter zu optimieren. Hierbei ist es nötig, experimentell und nach Vorlagen von existierenden neuronalen Netzen den Suchraum einzuschränken, weil sonst die Suche sehr zeitintensiv ist. Alternativ gibt es auch die Bibliothek GridSearchCV von scikit-learn um Hyperparameter-Tuning durchzuführen.

Ergebnis: Versuch 1.2

Der Versuch 1.2.4 mit einer Validation-Accuracy von 95.50% hat am besten abgeschnitten. Durch den Versuch hat sich gezeigt, dass mit Data Augmentation die Validation-Accuracy erhöht werden konnte. In diesem Fall nur leicht von 95.25% im Versuch 1.1.9 auf 95.50% im Versuch 1.2.4. Auf Abb. 3.1 ist zu sehen, dass die falsch klassifizierten Bilder relativ gleich verteilt waren. Es wurden 10 Bilder in die Klasse “unbefestigt” und 8 Bilder in die Klasse “befestigt” eingeordnet. Mit einer Trainings-Accuracy von 100% ist das Modell 1.2.4 überangepasst. Eine stärkere Transformierung der generierten Bilder wie in dem Versuch 1.2.3, hatte zu keinem besseren Ergebnis geführt.

Für ein genaueres Ergebnis könnten möglicherweise noch andere Data Augmentaion Methoden wie etwa imgaug oder CycleGAN [21] ausprobiert werden.

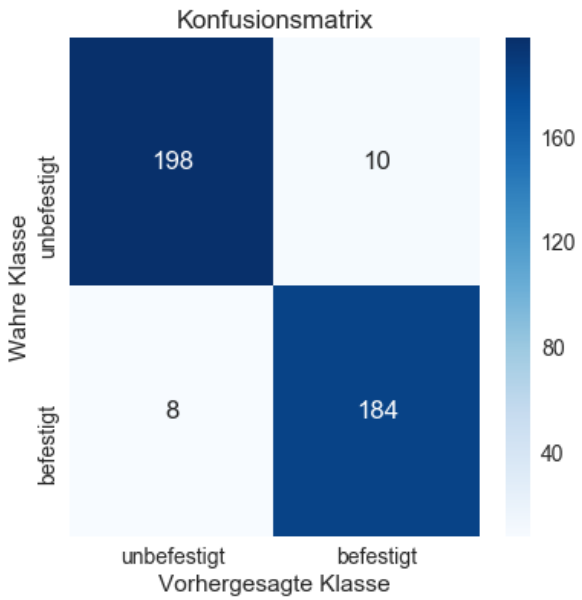

Abbildung 3.1: Konfusionsmatrix der Validierungsdaten aus dem Versuch 1.2.4

Ergebnis: Versuch 1.3

In dem Versuch 1.3.1 wurden mit dem Modell 1.2.3 45% der “unbefestigt” Wege falsch als “befestigt” klassifiziert (siehe Abb. 3.2). Zum Vergleich wurden 96.5% der Klasse “befestigt” korrekt klassifiziert. Also kann dieses Modell Radwege der Klasse “unbefestigt” schlechter vorhersagen. Das Faltungsnetz aus Versuch 1.2.3 hat die höchste Test-Accuracy in Versuch 1.3.1 der betrachteten Modelle (siehe Tab. 3.3). Durch die Regulierung, mit Data Augmentation während des Trainings, konnte eine bessere Generalisierung erreicht werden. Die Erwartung von einer Test-Accuracy von 95% konnte nicht erfüllt werden, weil die Bilder von einem anderen Abschnitt der Route kamen. Mit 2000 Testbildern, in dem Versuch 1.3.3, konnte mit maximal 85.30% Test-Accuracy ein höheres Ergebnis erzielt werden, als im Versuch 1.3.2 mit 400 Bildern. Damit wurde gezeigt, dass abhängig des Wegabschnitts die Qualität der Klassifizierung schwankt. Die unterschiedliche Bewölkung hatte in diesem Versuch keinen großen Einfluss auf die Test-Accuracy.

Für den Versuch 1.3.4 mit Datensatz 23 wurde die Erwartung bestätigt, das die Test-Accuracy niedriger ist, als in den anderen Versuchen dieses Versuchs 1.3. Dies wird daran liegen, dass keine Bildaufnahmen aus dem Datensatz 23 als Trainingsdaten genutzt wurden und die abgefahrene Strecke eine andere, als bei den Datensätzen 15 und 22, ist. Dazu kommt auch, dass die

Versuch Nr.	Bilder aus Datensatz Nr.	Faltungsnetz Versuch Nr.	Test-Acc.	Test-Loss
1.3.1	22	1.1.9	70%	2.1662
1.3.1	22	1.2.3	75.75%	0.6432
1.3.1	22	1.2.4	72.50%	3.1191
1.3.2	15	1.1.9	65.50%	1.6967
1.3.2	15	1.2.3	58.50%	1.0643
1.3.2	15	1.2.4	60.75%	2.9275
1.3.3	15	1.1.9	82.90%	0.8283
1.3.3	15	1.2.3	85.30%	0.3858
1.3.3	15	1.2.4	83.80%	1.141
1.3.4	23	1.1.9	61.70%	1.7472
1.3.4	23	1.2.3	51.70%	1.6489
1.3.4	23	1.2.4	60.50%	3.1968

Tabelle 3.3: Ergebnisse des Versuchs 1.3

Bilder mit einem MTB aufgenommen wurden. Der Versuch hat gezeigt, dass mehr gelabelte Trainingsdaten benötigt werden. Oberflächen die vorher noch nicht von den Modellen gesehen wurden, können nur erschwert richtig klassifiziert werden.

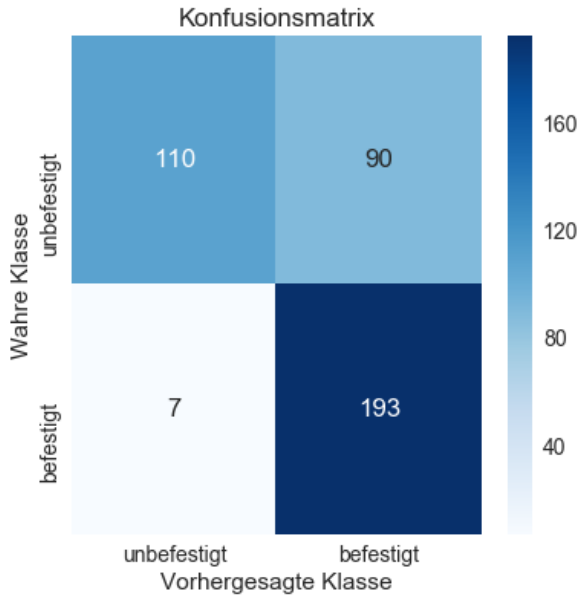

Abbildung 3.2: Konfusionsmatrix der Testdaten aus dem Versuch 1.3.1 mit Modell 1.2.3

4 Regelbasiertes Labeln mit Fuzzy Logik

Motivation

In Kapitel 3 wurden 2000 Bildaufnahmen der Oberfläche von Radwegen nach "befestigten" und "unbefestigten" Bildern manuell gelabelt. Die trainierten Faltungsnetze konnten Bilder von nicht befahrenen Radwegen nur mangelhaft klassifizieren siehe Abschn. 3.4. Deshalb wurde in diesem Kapitel ein System entwickelt, um Daten automatisiert zu labeln und damit die Klassifizierung zu vereinfachen.

Einleitung

Das Kapitel 4 ist in 5 Abschnitte aufgeteilt. In dem ersten Teil werden Versuche beschrieben, in denen Trainingsdaten mit Fuzzy Logik gelabelt wurden. In Abschnitt 4.2 und 4.3 werden Faltungsnetzversuche mit Daten, mit und ohne Wiederholung, der befahrenen Route beschrieben. Die Ergebnisse der Versuche werden in Abschnitt 4.4 dokumentiert. Ein Beispiel für eine Anwendung der Klassifizierung wird in Abschnitt 4.5 aufgezeigt.

4.1 Versuch 2.1: Fuzzy Logik

Einleitung

Mit Fuzzy Logik (siehe das Buch Fuzzy Logik, Neoronale Netze Evolutionäre Algorithmen [1]) wurde die Oberflächenqualität von Fahrradwegen, nach dem Kriterium der Fahrqualität abhängig von der Erschütterung beim Radfahren eingeordnet. Hierfür wurden unterschiedliche Regeln und Merkmale der erfassten Daten untersucht. Im Gegensatz zur klassischen Logik gibt es bei der Fuzzy Logik nicht nur Wahrheitswerte 0 für falsch oder 1 für eine wahre Aussage, sondern es gibt auch Wahrheitswerte zwischen 0 und 1 mit einem Wahrheitsgehaltwert μ. Damit ist es möglich eine stetige Menge zu beschreiben. Diese Menge, siehe Def. 4.1, wird als Fuzzy-Menge definiert. Dabei wird jedem Element x Teil der Grundmenge G einen Wahrheitsgehaltwert μ zugeordnet. Ein Eingangswert x kann mit einem Zugehörigkeitsgrad μ einem

linguistischen Wert LW zugeordnet werden. Für alle Versuche wurde die SciKit-Fuzzy [19] Bibliothek genutzt.

$$F := \{(x, \mu(x)) \mid x \in G \quad \text{und} \quad \mu(x) \in [0, 1]\} \tag{4.1}$$

Versuchsbeschreibung

Für die Einordnung der Daten in Klassen werden Regeln benötigt, welche dann in eine mathematische Gleichung umgewandelt werden. Die Frage, die experimentell zu beantworten galt, ist welche Regeln sind sinnvoll. Die Anzahl der Repräsentanten für jede Klasse der Oberflächenbeschaffenheit sollte ungefähr gleich sein, um später ein einseitiges Lernen der Faltungsnetzwerke zu vermeiden. Es wurden die Standardabweichung der Z-Achsen Beschleunigung a und die Geschwindigkeit v als Merkmal untersucht.

Hypothese

Die Hypothese ist, dass die Geschwindigkeit als Merkmal relevant ist, weil bei einer steigenden Geschwindigkeit auch die Bewegungsunschärfe zunimmt. Allerdings erhöht eine starke Erschütterung oder eine lange Belichtungszeit auch die Unschärfe der Bildaufnahme. In den folgenden Versuchen wird auch untersucht, ob eine Unterteilung der Daten anhand der Geschwindigkeit zu einem besseren Ergebnis, im Sinne der Accuracy bei der Klassifizierung, führt.

Fuzzy System

Mit einer nichtlinearen Funktion, siehe G. 4.2, kann ein Fuzzy System beschrieben werden. In der Funktion sind die Eingangsgrößen als x_i und die Ausgangsgrößen als y_i definiert. Ziel ist es, sprachliche Regeln wie die folgende Implikation, in eine mathematische Gleichung umzuformen. Wenn die Erschütterung x “sehr hoch” ist und die Geschwindigkeit v “niedrig” ist, dann ist die Fahrqualität y sehr “schlecht”.

$$y_{res} = f(x_1, ..., x_n) \tag{4.2}$$

Ein Fuzzy System besteht aus mehreren Funktionen. Diese werden nun nach der Reihenfolge ihrer Ausführung beschrieben. Bei der Fuzzifizierung werden zu den Eingangsgrößen x_i die Zugehörigkeitswerte bestimmt (siehe G. 4.3). Dabei gibt $LW_{i,j}$ die linguistischen Werte der Eingangsgröße x_i an mit $j = 1, ..., q$ und q steht für die Anzahl der linguistischen Werte.

$$\mu_{LW_{i,j}}(x_i) \tag{4.3}$$

Die Aggregation (siehe G. 4.4) ist die Ausführung und Verknüpfung der Implikation. Die Funktion $min\{\mu_a, \mu_b\}$ ist die Fuzzy-Und-Verknüpfung und gibt alle Werte zurück, bei welchem sich der Wahrheitswert μ überschneidet. Als Fuzzy-Oder-Verknüpfung wird die Funktion $max\{\mu_a, \mu_b\}$ definiert. Diese gibt alle Wahrheitswerte μ zurück, ähnlich der booleschen Logik, aber mit stetigen Werten.

$$\begin{aligned} &\mu_{agg,k}(x_i, ..., x_n) = min\{\mu_{LW_{1,i}}(x_1), ..., \mu_{LW_{n,p}}(x_n)\} \\ &k = 1, ..., m \end{aligned} \tag{4.4}$$

Dann definiert G. 4.5 die Schlussfolgerung der Implikation. Die Akkumulation siehe G. 4.6 beschreibt die Überlagerung der Ergebnisse der Implikation.

$$\begin{aligned} &\mu_k(x_1, ..., x_n, y) = min\{\mu_{agg,k}(x_1, ..., x_n), \mu_{LW_k}(y)\} \\ &k = 1, ..., m \end{aligned} \tag{4.5}$$

$$\mu_{res}(x_1, ..., x_n, y) = max\{\mu_1(x_1, ..., x_n, y), ..., \mu_m(x_1, ..., x_n, y)\} \tag{4.6}$$

Als letztes wird die Defuzzifizierung ausgeführt (siehe G. 4.7). Dabei wird die Ausgangsgröße y_{res} bestimmt. Für die Defuzzierung wurde die Operation Mean of Maxima MOM angewendet. Innerhalb dessen ist C_{max} die Menge aller Maximalstellen von y. Das Ergebnis ist der Mittelwert aller Maximalstellen. Es gibt für jede Funktion weitere Rechenoperationen wie Minimum, Maximum, Produkt, Summe oder Differenz. Für die folgenden Versuche werden wie in den G. 4.2 bis 4.7, die beschriebenen Operationen auf der rechten Seite der Gleichungen genutzt.

$$y_{res} = \frac{\int\limits_{y \in C_{max}} y \, dy}{\int\limits_{y \in C_{max}} y \, dy} \tag{4.7}$$

Versuche 2.1.1 - 2.1.6

Um die Standardabweichung der Z-Achsen-Beschleunigung in Erschütterungsabstufungen zu unterteilen werden sinnvolle Grenzwerte benötigt. Im Hintergrund steht die Überlegung, mit Referenzstrecken eine Unterteilung, in “leichte” und “hohe” Erschütterung, vorzunehmen. In dem Versuch 2.1.1 werden also zwei Referenzstrecken gesucht, eine mit “leichter” und eine mit “hoher” Erschütterung.

Abbildung 4.1: Ausschnitt eines asphaltierten Radwegs aus Datensatz 38

Abbildung 4.2: Schotterweg erfasst in Datensatz 38

Für die Referenzstrecken wurde jeweils ein Abschnitt aus Datensatz 38 ausgewählt, welcher von der Unebenheit beziehungsweise Rauheit der Strecke möglichst gleichbleibend ist. Diesbezüglich werden für die Zwischenstufen, von “leichter” bis hin einer “hohen” Erschütterung, die Mittelwerte der Standardabweichung genutzt. Der Mittelwert der Z-Achsen-Standardabweichung wird dann als Maximalwert μ für die Zuordnungsfunktion des Fuzzy Systems gewählt. Auf Abb. 4.1 ist ein Abschnitt der Referenzstrecke mit geringer Rauigkeit der Oberfläche zu sehen. Der Radweg bestand aus Asphalt.

In Gegensatz dazu ist auf Abb. 4.2 ein Kiesweg. mit vermutlich hoher Standardabweichung der Z-Achsen-Beschleunigung zu sehen.

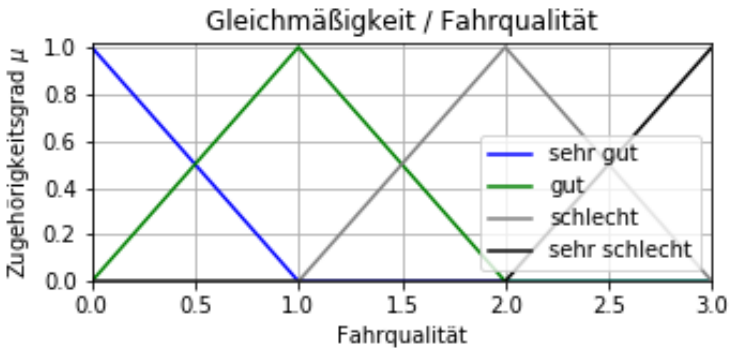

Abbildung 4.3: Zuordnung der Fahrqualität in dem Versuch 2.1.2

Mit den Ergebnissen des Versuchs wird die Fahrqualität in 4 Stufen eingeteilt. Die Einteilung orientiert sich an dem OSM-Attribut “smoothness” für Radwege. Die Gleichmäßigkeit der Oberfläche wird mit dem Attribut in 8 Stufen, von hervorragend bis nicht passierbar, unterschieden. Bezüglich des Versuchs 2.1.2 ist die Anzahl der unterschiedlichen Klassen auf 4, also die Hälfte, reduziert worden (siehe Abb. 4.3). Die Hypothese ist, dass dadurch die Klassifizierung vereinfacht werden kann. Für das Fuzzy System wird die Regelbasis aus Tab. 4.1 verwendet.

	sehr leichte Erschütterung	leichte Ersch.	hohe Ersch.	sehr hohe Ersch.
Fahrqualität	sehr gut	gut	schlecht	sehr schlecht

Tabelle 4.1: Regelbasis für den Versuch 2.1.2

Darauf folgend werden in dem Versuch 2.1.3 die linguistischen Ausgangsgrößen, durch die Unterteilung zwischen “niedriger” und “hoher” Geschwindigkeit, zusätzlich erweitert. Das Ziel ist die Hypothese zu untersuchen, ob mit einer Unterteilung der Geschwindigkeit die Klassifizierung von Faltungsnetzen verbessern lässt. Die Tabelle 4.2 ist so zu interpretieren, dass die linguistischen Ausgangsgrößen zusätzlich mit der Geschwindigkeit erweitert wurden. Für die Unterteilung wurde der Medianwert der Geschwindigkeit in km/h genutzt. Dieser beträgt für die Datensätze 37 bis 42 $16km/h$. Eine Geschwindigkeit über $16km/h$ gilt dann als “hohe” Geschwindigkeit.

	sehr leichte Ersch.	leichte Ersch.	hohe Ersch.	sehr hohe Ersch.
niedrige Gesch. v	sehr gut/niedrig	gut/n.	schlecht/n.	sehr schlecht/n.
hohe Gesch. v	sehr gut/hoch	gut/h.	schlecht/h.	sehr schlecht/h.

Tabelle 4.2: Regelbasis für den Versuch 2.1.3

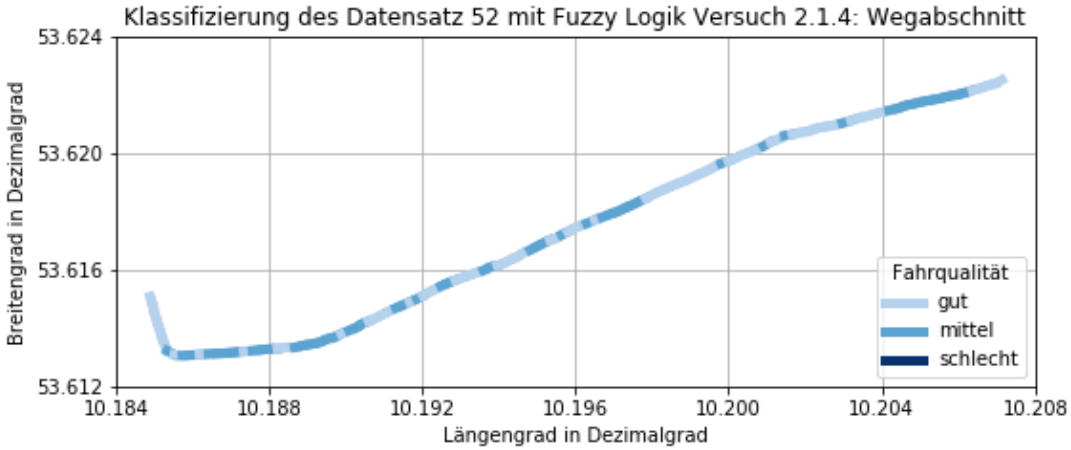

Abbildung 4.4: Wegabschnitt aus Datensatz 52

Aus den Ergebnissen von Faltungsnetzversuch 2.2.4 ist die Annahme entstanden, dass die aufgestellte Regelbasis für diese Aufgabe nicht geeignet ist. Daher wurde der Grenzwert für die Fahrqualität "sehr schlecht" im Versuch 2.1.4 angehoben, um eine höhere Accuracy zu erreichen. Die Anzahl der Ausgangsgrößen wurde auf 3 reduziert wie auf Tab. 4.3 zu sehen. Auch aus der Motivation heraus die Klassen mehr voneinander abzugrenzen.

	leichte Erschütterung	mittlere Ersch.	hohe Ersch.
Fahrqualität	gut	mittel	schlecht

Tabelle 4.3: Regelbasis für Versuch 2.1.4 mit 3 Ausgangsgrößen

In dem Versuch 2.1.5 wurden die Daten nochmal mit einer Geschwindigkeitsunterteilung, wie in Versuch 2.1.3 gelabelt. Dabei wurde die Regelbasis aus dem Versuch 2.1.4 genutzt (siehe Tab. 4.3). Bei der Darstellung der Daten aus Datensatz 52, in einem Diagramm mit Labeln aus Fuzzy Logik-Versuch 2.1, ist aufgefallen, dass die Klasse "mittelmäßig" zu häufig auf einem Wegabschnitt vorhergesagt wurde. Auf Abb. 4.4 ist der Wegabschnitt zu sehen, welcher überwiegend mit der Klasse "mittel" gelabelt sein sollte. Nach jeweils $20m$ zurückgelegten Radweg beginnt ein neuer Wegabschnitt. Die Klasse mit der größten Häufigkeit während des Wegabschnitts wurde gelabelt. Deshalb wurde die Zuordnung der Erschütterung in dem Versuch 2.1.6 für die Trainingsdaten mit Wiederholung angepasst (siehe Abschn. 4.3).

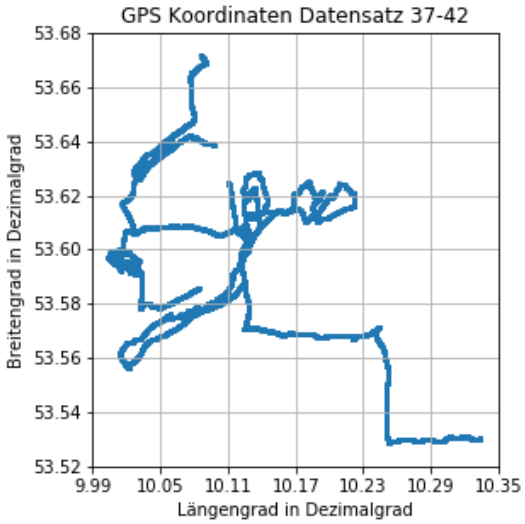

Abbildung 4.5: Längengrad-und Breitengraddiagramm der Datensätze 37 bis 42

4.2 Versuch 2.2: Datensätze ohne Wiederholung

Einleitung

In diesem Versuch wurden Faltungsnetze mit den Daten aus dem Datensatz 37 bis 42 trainiert. Die Daten wurden auf unterschiedlichen Routen erfasst. Beim Start und Ende der Datenerfassung einer Strecke überschneiden sich einige der befahrenen Wege. Diese Radwege wurden also mehr als einmal befahren.

Versuchsaufbau

Die Datensätze 37 bis 42 wurden auf unterschiedlichen Routen, siehe Abb. 4.5, erfasst. Die Bildrate betrug im Schnitt 3.13 Bilder pro Sekunde bei einer Gesamtfahrzeit von über 10 Stunden. Insgesamt wurden, dabei ca. 164 Kilometer zurückgelegt. Insgesamt wurden 115.892 Bilder aufgenommen. Bei der Datenerfassung wurde das dynamische Zeitfenster genutzt. Zu den Bildern wurden die Messwerte Z-und Y-Achsen-Beschleunigung in m/s^2, Nick-Winkel in rad, Zeitstempel der Aufnahme in Coordinated Universal Time UTC und die gemessene Geschwindigkeit in km/h erfasst.

Die Daten wurden so aufgeteilt und gemischt, dass 10% der Gesamtdaten zum Testen genutzt wurden und die übrigen 90% in 80% Trainingsdaten, sowie 20% Validierungsdaten aufgeteilt, wurden. Die Validierungsdaten wurden als Performancemetrik am Ende des Trai-

nings einer Epoche genutzt. Mit den Testdaten wurde dann nochmals, unabhängig von den Validierungsdaten, dass Faltungsnetz getestet.

Versuche 2.2.1 - 2.2.10

In dem Versuch 2.2.1 wurden die Trainingsdaten in 4 Klassen nach der Fahrqualität klassifiziert und abhängig von der Erschütterung mit der Z-Achsen-Beschleunigung gemessen. Für weitere Informationen zur Einordnung der Daten siehe den Fuzzy Logik-Versuch 2.1.2 Abschnitt 4.1. Als Parameter für das Faltungsnetz wurde als Vorlage das Model aus dem Versuch 1.1.9 genutzt, bei welchem die Daten manuell gelabelt wurden (siehe Kapitel 3). Die Anzahl der Epochen betrug 20. Wenn sich nach 3 Epochen die Validation-Accuracy nicht mehr erhöhte, wurde das Training abgebrochen.

Für den Versuch 2.2.1 gab es eine unterschiedliche Anzahl an Repräsentanten für jede Klasse. In dem Versuch 2.2.2 wurde untersucht, ob eine Verbesserung der Test-Accuracy erzielt werden kann, wenn die Anzahl der Daten pro Klasse normiert werden. Für das Faltungsnetz wurden die Parameter aus dem Versuch 1.1.9 genutzt.

Um ein besseres Ergebnis zu erzielen, wurden in Versuch 2.2.3 die Parameter des Faltungsnetz optimiert mit Hypertuning. Dabei wurden die Dropout Rate, die Anzahl der Faltungsschichten von 2 bis 4 und die Optimierungsfunktion sowie weitere Parameter mit Hyperas gesucht. Die Anzahl an Durchläufen mit Hyperas belaufen sich auf 30, mit jeweils 10 möglichen Epochen und Early Stopping. In diesem Versuch wurden die Daten wie in Versuch 2.2.2 normiert, damit die Anzahl an Daten pro Klasse gleich ist.

Eine Hypothese war, dass bei höherer Geschwindigkeit die Bewegungsunschärfe bei den Bildaufnahmen steigt. In dem Versuch 2.2.4 wurden die Daten nicht nur nach der Fahrqualität unterteilt, sondern auch nach der gemessenen Geschwindigkeit, in “niedrige” und “hohe” Geschwindigkeit. Daraus ergaben sich 8 unterschiedliche Klassen. Die Daten wurden im Fuzzy Logik-Versuch 2.1.3 gelabelt. In diesem Versuch wurde wieder mit Hypertuning nach Parametern für ein Faltungsnetz gesucht. Mit Hyperas wurden 20 Durchläufe ausgeführt. Als Erweiterung wurden auch die Stapelgrößen (batch size) 8, 16 und 32 als Variable ausprobiert.

In dem Versuch 2.2.5 wurde für den Geschwindigkeitsbereich “hoch” und “niedrig”, nach der Klassifizierung aus dem Fuzzy Logik-Versuch 2.1.3, ein Faltungsnetz trainiert. Dafür wurden die gefundenen Faltungsnetze aus Versuch 2.2.3 und 2.2.4 genutzt. Um die Versuche wiederholbar zu machen wurde als Gewichtsinitialisierung Glorot Uniform [10] (siehe G. 4.8) mit dem Wert 42 für den Zufallsgenerator verwendet. Wobei n_{inputs} und $n_{outputs}$ für die Anzahl der eingehenden sowie ausgehenden Verbindungen der Schicht steht, deren Gewichte initialisiert werden sollen.

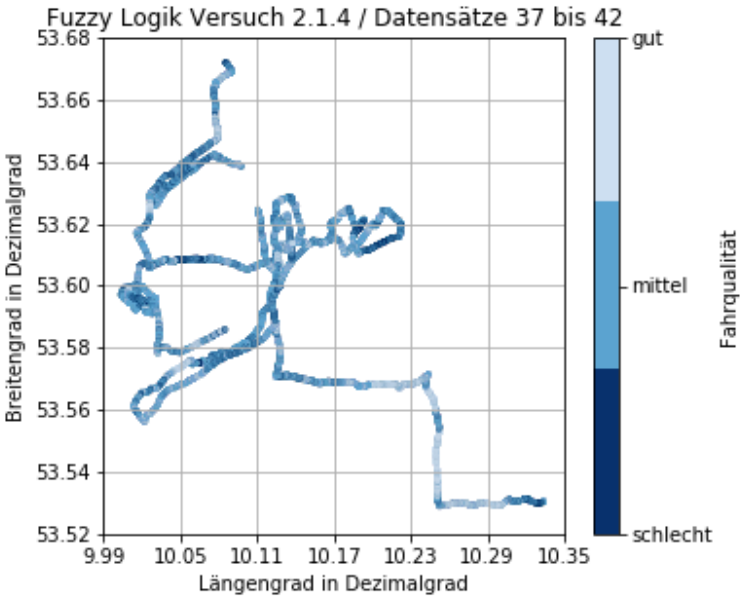

Abbildung 4.6: Streudiagram der GPS Koordinaten mit einem Transparenzwert von 10% der einzelnen Messwerte

In den ersten Teilversuchen 2.2.5.1 und 2 wurden die Klassen mit der Geschwindigkeit "hoch" ausgewählt und ein Modell mit den Parametern aus Versuch 2.2.3 trainiert. Im Folgenden wurde der Versuch mit dem Modell Parametern aus Versuch 2.2.4 wiederholt. Die Anzahl der Daten jeder Klasse betrug 9548. Anschließend wurden die Versuche 2.2.5.3 und 4 mit den Daten durchgeführt, welche mit einer "niedrigen" Geschwindigkeit erfasst wurden. Dabei gab es 9152 Daten pro Klasse.

$$r = \sqrt{\frac{6}{n_{inputs} + n_{outputs}}} \tag{4.8}$$

Zum Vergleich wurde in Versuch 2.2.6 ein Faltungsnetz ohne Geschwindigkeitsunterteilung und dem Labeln aus Fuzzy Logik-Versuch 2.1.2 durchgeführt. Dabei wurde die Faltungsnetz Architektur aus Versuch 2.2.4 genutzt. Die Hypothese war, dass die Test-Accuracy niedriger ist als in dem Versuchen 2.2.5.

In dem Versuch 2.2.7 wurden die Daten mit den Fuzzy Logik-Versuch 2.1.4 (siehe Abb. 4.6) gelabelt. Es ergaben sich drei unterschiedliche Klassen von "gut", "mittel" und "schlecht" Fahrqualität. Bei diesem Versuch wurde ein Faltungsnetz mit den gefundenen Parametern aus

dem Versuch 2.2.4 trainiert. Es wurde erwartet, dass die Test-Accuracy im Vergleich zu den Versuchen 2.2.5 und 2.2.6 höher ist.

Die These, ob mit einer binären Klassifizierung die Klassen besser unterscheidbar sind, wurde im Versuch 2.2.8 untersucht. Bei diesem Versuch gab es nur die Fahrqualität "gut" oder "schlecht" für Radwege. Die maximal mögliche Anzahl der Epochen betrug 60. Im Vergleich zu den vorherigen Versuchen sollte die Test-Accuracy höher als $\approx 50\%$ betragen.

Ziel des Versuchs 2.2.9 war es zwei Faltungsnetzwerke zu trainieren. Eines wurde für den Geschwindigkeitsbereich "niedrig" und eines für "hoch" mit der Klassifizierung aus dem Fuzzy Logik-Versuch 2.1.5, trainiert. Das Ergebnis aus dem Versuch 2.1.5 zeigte, dass die Klasse 2 "schlecht"/"niedrig" nur 5684 Mal vorkam. Aus diesem Grund wurde mit Data Augmentation die Anzahl der niedrigsten, vorkommenden Klassen erhöht, so dass diese gleich der Anzahl der zweit häufigsten Klasse war. Hierbei wurde darauf geachtet, die Testdaten und Validierungsdaten nicht zu ändern und die Anzahl der Daten jeder Klasse zu normieren. In dem Versuch 2.2.9.1 wurde ein Faltungsnetz für die Geschwindigkeit "niedrig" trainiert und in Versuch 2.2.9.2 eines für den Geschwindigkeitsbereich "hoch" . Hierfür wurden die Parameter aus Versuch 2.2.4 genutzt.

In Versuch 2.2.10 wurde die Anzahl der erzeugten Daten auf die Klasse mit der geringsten Häufigkeit reduziert. Dadurch sollte verhindert werden, dass das Faltungsnetz übertrainiert wird, siehe Ergebnis aus Versuch 2.2.9, indem die Anzahl der generierten Bilder verringert wurde.

4.3 Versuch 2.3: Datensätze mit Wiederholung

Einleitung

Bei diesem Versuch wurden Faltungsnetzwerke trainiert. Die Trainingsdaten entstanden durch wiederholtes Befahren einer gleichen Route mit dem Fahrrad. Ziel war es, durch die Wiederholung der befahrenen Radwege, die Anzahl der unterschiedlichen Oberflächen-Typen zu reduzieren, und somit eine bessere Klassifizierung zu ermöglichen.

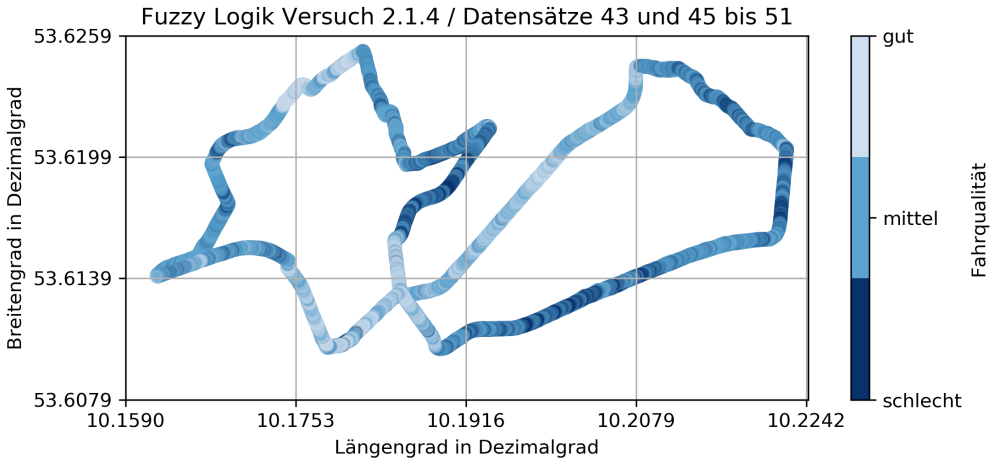

Abbildung 4.7: Längengrad und Breitengrad mit Klassen aus Fuzzy Logik-Versuch 2.1.4

Hypothese

Durch die Wiederholung bei der Datenerfassung sollte sich eine höhere Test-Accuracy als in Versuch 2.2, siehe Abschn. 4.2 erzielen lassen. Auf der befahrenen Route gibt es mehr Unebenheiten im Vergleich zu Datensatz 37 bis 42. Dadurch standen mehr Trainingsdaten zur verfügbar, wenn die Anzahl der Daten normiert wurde.

Versuchsaufbau

Es wurde eine Route im Naturschutzgebiet Höltigbaum von ca. 12 Kilometer Länge 14 Mal befahren (siehe Abb. 4.7). Dabei setzen sich die Daten aus den Datensätzen Nummer 43 und 45 bis 51 zusammen. Die Gesamtzahl der Bilder betrug 120.421 Stück. Insgesamt wurden $\approx$ $167km$ mit einem Crossrad Fahrrad zurückgelegt. Wie bei den Datensätzen 37 bis 42 wurden die gleichen Typen von Messwerten, mit einem dynamischen Zeitfenster, erfasst. Die Bildrate betrug als Mittelwert 3.2 Bilder pro Sekunde. Bei den Datensätzen 37 bis 42 wurden überwiegend befestigte Radwege in der Stadt befahren. Im Unterscheidet zu der Route in Höltigbaum, welche ungefähr zur Hälfte aus unbefestigten Wegen besteht.

Versuche 2.3.1 - 2.3.8

Die Daten in dem Versuch 2.3.1 wurden im Fuzzy Logik-Versuch 2.1.4 gelabelt. Hierzu wurde zwischen den 3 Klassen “gut”, “mittel” und “schlecht” nach der Fahrqualität unterschieden.

Dieser Versuch ist eine Wiederholung des Faltungsnetzversuchs 2.2.7, mit dem Unterschied, dass die Datensätze 43 und 45 bis 51 verwendet wurden. Die Anzahl der Epochen betrug 20.

In dem Versuch 2.3.2 wurde der Versuch 2.2.8 (siehe Abschn.4.2) wiederholt. Auch hier wurde erwartet, dass die Test-Accuracy höher ist als im Versuch 2.2.8. Bei dem Versuch wurden zwei Klassen "gute" und "schlechte" verwendet.

Die Hypothese, dass durch Anpassen der Parameter eine höhere Test-Accuracy erreicht werden kann, als in Versuch 2.3.2 (siehe Tab. 4.6), wurde in Versuch 2.3.3 untersucht. Dabei wurde nochmal nach besseren Parametern mit Hypertuning für das Modell aus Versuch 2.2.4 gesucht. In dieser Hinsicht wurde, wie in dem Versuch 2.2.3, die Dropout Rate, die Anzahl der Faltungsschichten von 3 bis 5 und die Optimierungsfunktion, sowie weitere Parameter mit Hyperas gesucht. Die Anzahl an Durchläufe mit Hyperas waren 77. Die Daten wurden im Fuzzy Logik-Versuch 2.1.4 gelabelt.

In dem Versuch 2.3.3 wurden jeweils nur 10 Epochen genutzt, um ein Faltungsnetz zu trainieren. Deshalb wurde untersucht, ob sich die Test-Accuracy erhöht, indem die Anzahl der Epochen für Versuch 2.3.4 erhöht wurde.

Mit den gefundenen Parametern aus dem Versuch 2.3.3 sollte sich auch die Test-Accuracy mit den Daten aus Datensatz 37 bis 42 steigern lassen. Die Annahme war, dass die Test-Accuracy höher sein wird als in Versuch 2.2.7 (56.79%). Dies wurde in Versuch 2.3.5 untersucht.

Der Versuch 2.3.6 ist aus dem Ergebnis von Versuch 2.2.5 motiviert. Ziel war es, zwei Faltungsnetze jeweils eines für den Geschwindigkeitsbereich "niedrig" und "hoch", zu trainieren. Die Erwartung war, dass die Test-Accuracy und Test-Loss Werte leicht höher beziehungsweise niedriger als im Versuch 2.3.4 sind.

Für den Versuch 2.3.7 wurde die Methode "one-versus-the-rest" [9] genutzt. Entsprechend übersetzt eine Klasse gegen die restlichen Klassen. Die Methode funktioniert nach dem Prinzip, mehrere binäre Klassifikatoren zu trainieren. Für jede Klasse wird dabei ein Faltungsnetz trainiert. Das fertige System wird so genutzt, dass die Vorhersage jedes Modell verglichen wird und die Klasse mit der höchsten Wahrscheinlichkeit nach der Softmax Funktion vorhergesagt wird. In dem Versuch 2.3.7.1 wurde ein Modell trainiert, welches auf die Klasse "gut" spezialisiert war. Darauf folgend wurde ein Faltungsnetz für die Klasse "mittel" und eines für die Klasse "schlecht" in den Versuchen 2.3.7.2 sowie 2.3.7.3 trainiert. Im Fuzzy Logik-Versuch 2.1.4 wurden die Daten gelabelt. Den Testdaten wurden 20% der gleichen Daten für jeden Versuch zugewiesen. Getestet wurde dann mit allen Modellen.

Bei der Darstellung der Testergebnisse aus Datensatz 52, welche im Fuzzy Logik-Versuch 2.1.4 gelabelt wurden, traf die Erwartung der Oberflächengleichmäßigkeit für mehrere Wegabschnitte nicht zu. In diesem Zusammenhang siehe auch Abschnitt 4.1 zum Fuzzy-Versuch

2.1.6. Durch die Anpassung der Gleichmäßigkeit beziehungsweise Fahrqualitätsklassen sollte die Test-Accuracy im Vergleich zum Versuch 2.3.4 höher sein. Gelabelt wurden die Datensätze 43, 45 bis 51 in Fuzzy Logik-Versuch 2.1.6. Für das Faltungsnetz wurde die Architektur aus Versuch 2.3.3 ausgewählt. Die Anzahl der Epochen betrug 100.

4.4 Ergebnisse

Ergebnis 2.1: Fuzzy Logik

Der Mittelwert der Standardabweichung der Z-Achsen-Beschleunigung betrug $\approx 1.5m/s^2$ für die Referenzstrecke mit "leichter" Erschütterung im Versuch 2.1.1. Bei dem Beispiel mit "hoher" Erschütterung betrug der Mittelwert der Z-Achsen Standardabweichung $\approx 7,3m/s^2$. Somit unterscheiden sich die Radwege anhand der mittleren Standardabweichung, um $\approx 5,8/s^2$.

Mit den berechneten Mittelwerten wurden die Eingangsgrößen, in dem Versuch 2.1.2, auf linguistische Werte LW abgebildet (siehe Abb. 4.8). Dabei wurde eine Trapezfunktion für die Werte "sehr gut" und "sehr schlecht" genutzt. Um alle Werte welche kleiner und größer als die Mittelwerte der Standardabweichung der Z-Achsen-Beschleunigung sind von den Referenzstrecken zuzuordnen. Für die Erschütterungswerte "gut" und "schlecht" wurden Dreiecksfunktionen genutzt. Die Eingangsgrößen hatten eine Genauigkeit von 3 Kommastellen.

Als Beispiel wurde für die Eingangsgröße die Standardabweichung der Z-Achsen Beschleunigung von $a = 3m/s^2$ gewählt. Auf der Abb. 4.9 ist zu sehen, dass der Eingangswert zu der Klasse 1 zugeordnet wurde.

Die Zuordnung der Geschwindigkeit in "niedrige" und "hohe" Geschwindigkeit für Versuch 2.1.3 ist auf Abb. 4.10 zu sehen. Diesbezüglich ist auf Abb. 4.11 die Ausgangsgrößen-

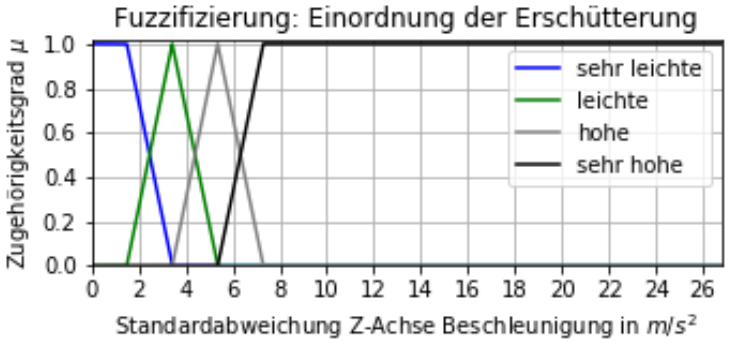

Abbildung 4.8: Versuch 2.1.2: Fuzzifizierung der Eingangswerte

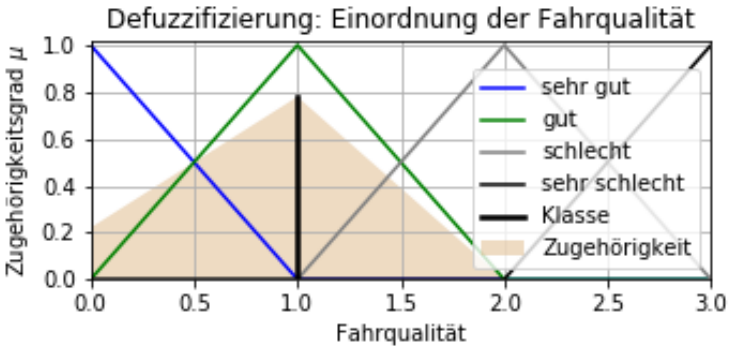

Abbildung 4.9: Versuch 2.1.2: Zuordnung der Fahrqualität mit der gegebenen Eingangsgröße von $a = 3m/s^2$ der Standardabweichung

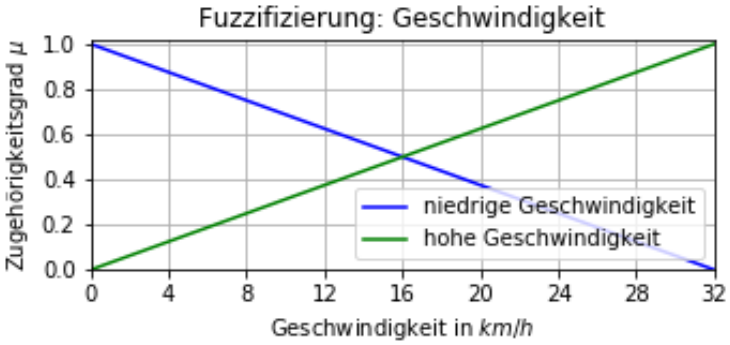

Abbildung 4.10: Versuch 2.1.3: Fuzzifizierung in die linguistische Variablen “niedrige” und “hohe” Geschwindigkeit

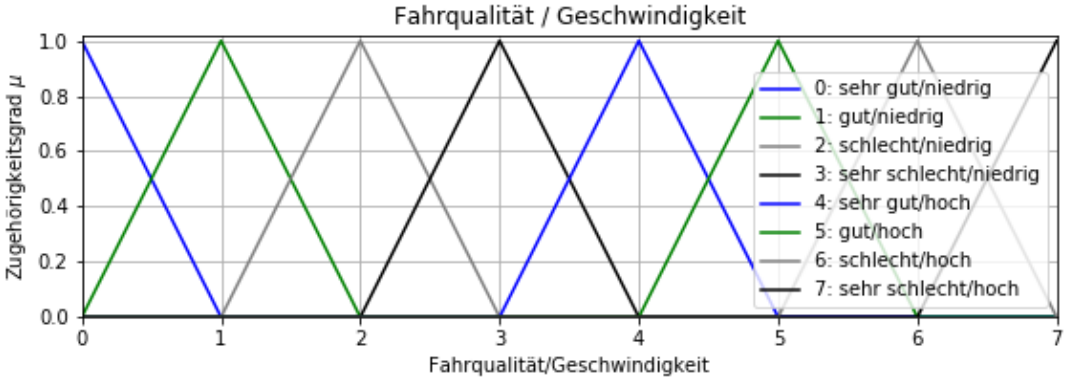

Abbildung 4.11: Ausgangsgrößen des Versuch 2.1.3

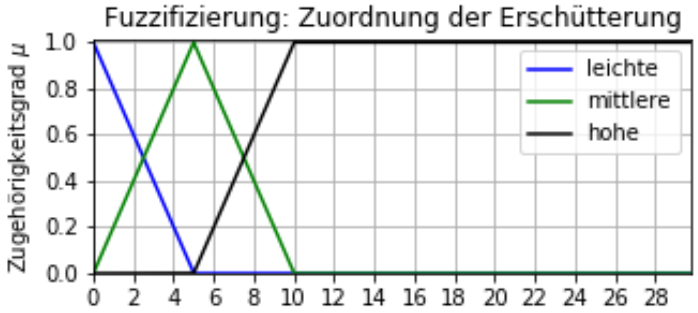

Abbildung 4.12: Versuch 2.1.4: Fuzzifizierung in die linguistische Variablen “leichte” , “mittlere” und “hohe” Erschütterung

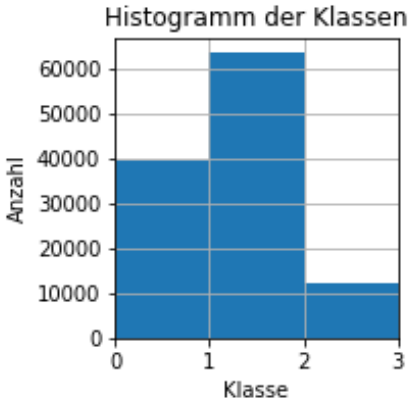

Abbildung 4.13: Histogramm der Klassen aus Datensatz 37 bis 42, gelabelt im Versuch 2.1.4

Zuordnungsfunktion abgebildet. Um die Ähnlichkeit der Klassen bei “niedriger” und “hoher” Geschwindigkeit hervorzuheben, wurden die Farben ab Klasse 4 wiederholt. Ein Nachteil an dieser Herangehensweise mit der Geschwindigkeitsunterteilung ist, dass der genutzte Grenzwert nicht viel über die Bewegungsunschärfe aussagt.

In dem Versuch 2.1.4 wurde der Grenzwert für die Fahrqualität “sehr schlecht” auf $10m/s^2$ Standardabweichung der Z-Achsen-Beschleunigung, siehe Abb. 4.12, angehoben. Nach der Zuordnung der Daten aus Datensatz 37 bis 42 hatte die Klasse 2, also Radwege mit “schlechter” Fahrqualität, die geringste Anzahl mit 12290 Bildaufnahmen (siehe Abb. 4.13).

Mit der angepassten Fuzzifizierung der Eingangswerte, siehe Abb. 4.14, entsprecht der Wegabschnitt überwiegend der Klasse “gut” (siehe Abb. 4.15).

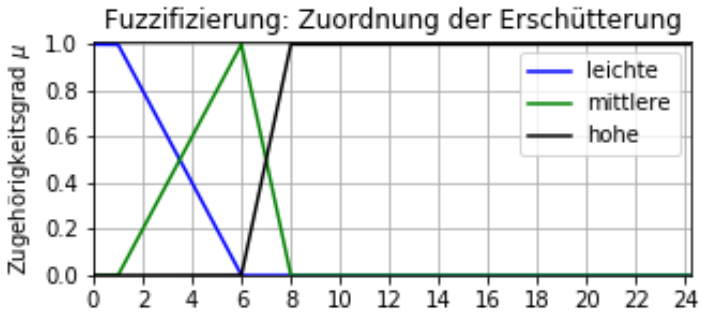

Abbildung 4.14: Versuch 2.1.6: Fuzzifizierung

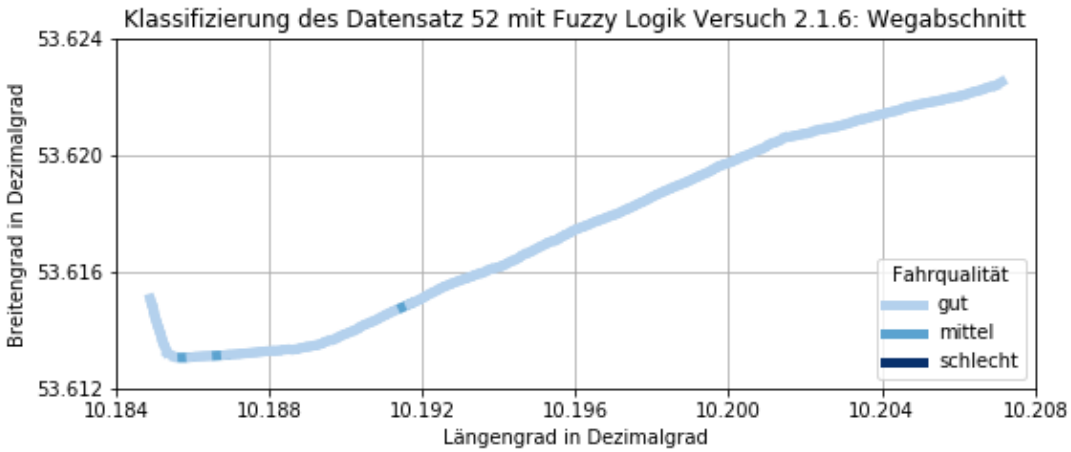

Abbildung 4.15: Wegabschnitt gelabelt mit dem Fuzzy-Versuch 2.1.6

Ergebnis 2.2: Faltungsnetzwerk Versuche

Versuch Nr.	Trainings-Accuracy	Tr.-Loss	Val.-Acc.	Val.-Loss	Test-Acc.	Test-Loss
2.2.1	33.03%	1.3376	33.06%	1.339	33.21%	1.3378
2.2.2	24.92%	1.3933	24.92%	1.3889	24.64%	1.3886
2.2.3	31.19%	1.3639	37.27%	1.3719	36.69%	1.3798
2.2.4	20.92%	1.9879	25.53%	2.0331	17.18%	2.3753
2.2.5.1	36.56%	1.3069	41.87%	1.2251	41.52%	1.2015
2.2.5.2	36.86%	1.3120	42.79%	1.2830	42.857%	1.2502
2.2.5.3	37.61%	1.2887	41.40%	1.2380	42.71%	1.2447
2.2.5.4	39.35%	1.2740	41.74%	1.2466	42.21%	1.2398
2.2.6	34.93%	1.3333	40.05%	1.2973	39.50%	1.3061
2.2.7	51.65%	0.9778	56.17%	0.9575	56.79%	0.9589
2.2.8	78.24%	0.4379	78.92%	0.6130	78.70%	0.6133
2.2.9.1	70.84%	0.7010	51.16%	4.6704	51.95%	4.6868
2.2.9.2	68.61%	0.7043	49.23%	5.2841	48.89%	5.3739
2.2.10.1	58.96%	0.8938	32.73%	10.8378	32.03%	10.9553
2.2.10.2	33.32%	10.7419	33.59%	10.7040	33.56%	0.7092

Tabelle 4.4: Ergebnisse der Versuche mit den Datensätzen 37 bis 42

Wie in Tab. 4.4 zu sehen ist war der Versuch 2.2.1 nicht sehr erfolgreich. Dabei wurde ausschließlich die Klasse “sehr gut” vorhergesagt. Die Test-Accuracy betrug nur $\approx 33\%$. Das Faltungsnetz könnte nicht komplex genug sein, weil auch die Trainings-Accuracy gering war oder die Trainingsdaten sind nicht aussagekräftig genug.

Auch mit der Normierung der Daten in Versuch 2.2.2 war das Ergebnis mit 25% Test-Accuracy unzureichend. In dieser Hinsicht wurden in diesem Versuch im Vergleich zu 2.2.1 auch andere Klassen als “sehr gut” zu 3.7% vorhergesagt.

Aus diesem Grund wurde im Versuch 2.2.3 nach einer besseren Architektur mit Hypertuning gesucht. Die höchste, erreichte Test-Accuracy war 36.69%, mit 3 Faltungsschichten, ELU als Aktivierungsfunktion und Adam als Optimierungsfunktion. Die Ausführungszeit betrug ≈ 5 Stunden. Während des Versuches ist aufgefallen, dass die zufällige Dropout-Rate häufig über 70% lag. Beim nächsten Versuch sollte die mögliche Maximalrate nicht auf 100%, sondern auf 70% festgesetzt werden.

Bei der Unterteilung der Geschwindigkeit im Versuch 2.2.4, konnte eine Test-Accuracy von 17.18% erreicht werden (siehe Tab. 4.5).

Wie auf der Tab. 4.4 zu sehen ist, wurde für den Versuch 2.2.5 mit den Parametern aus Versuch 2.2.4 ein besseres Ergebnis als mit den Parametern aus Versuch 2.2.3 erzielt. Hierzu vergleiche die Test-Accuracy aus Versuch 2.2.5.1 (41.52%) mit 2.2.5.2 (42.857%) für die Geschwindigkeit

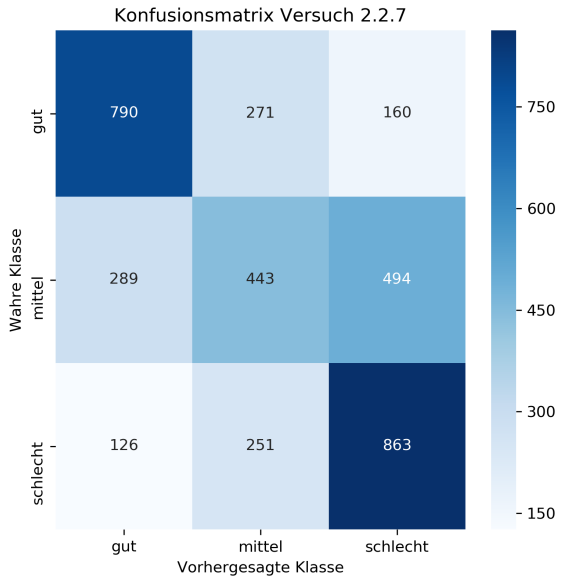

Abbildung 4.16: Konfusionsmatrix der Testdaten aus Versuch 2.2.7

"hoch". Das Ergebnis der Test-Accuracy für die Geschwindigkeit "niedrig" von Versuch 2.2.5.4 ist nur unwesentlich schlechter, als die vom Versuch 2.2.5.3.

Mit Verzicht auf eine Geschwindigkeitsunterteilung wurde in Versuch 2.2.6 eine Test-Accuracy, um nur $\approx 3\%$ niedriger, als in Versuch 2.2.5, erzielt. Damit wurde gezeigt, dass durch eine Geschwindigkeitsnormierung ein leicht besseres Ergebnis nach der Metric-Accuracy erzielt werden kann.

Auf Abb. 4.16 ist die Konfusionsmatrix der Testdaten, aus dem Versuch 2.2.7, zu sehen. Die Test-Accuracy betrug nur $56,79\%$ aber auf der Konfusionsmatrix ist zu sehen, dass $p = \frac{W}{G} \rightarrow \frac{1305}{1591} = 82\%$ der falsch klassifizierten Bilder einer benachbarten Klasse zugeordnet wurden. Die Test-Accuracy war höher als in den vorherigen Versuchen, aber es existierten auch nur 3 unterschiedliche Klassen statt 4, wie in den vorherigen Versuchen.

Wie auf Tab. 4.4 zu sehen, betrug die Test-Accuracy für Versuch 2.2.8 78.70%. Die vergleichsweise hohe Test-Accuracy zu den vorherigen Versuchen konnte erreicht werden, weil es nur zwei Klassen gibt und diese sich optisch stark voneinander unterscheiden.

Im Versuch 2.2.9 waren die Modelle aus Versuch 2.2.9.1 und 2.2.9.2 mit Data Augmentation übertrainiert. Die Trainings-Accuracy bei Versuch 2.2.9.1 betrug 70.84% und die Test-Accuracy nur 51.95%. Die Modelle haben gelernt die generierten Daten zu klassifizieren.

Nr.	Typ	Ausgabeform	Anzahl der zu trainierenden Parameter
1	Faltungsschicht 1(3x3 Faltungskerngröße)	(368, 70, 32)	896
2	Aktivierungsfunktion: ELU	(368, 70, 32)	0
3	Max-Pooling (2x2)	(184, 35, 32)	0
4	Dropout (11% Rate)	(184, 35, 32)	0
5	Faltungsschicht 2(3x3)	(184, 35, 32)	9248
6	Aktivierungsfunktion: ELU	(184, 35, 32)	0
7	Max-Pooling (2x2)	(92, 17, 32)	0
8	Dropout (38% Rate)	(92, 17, 32)	0
9	Faltungsschicht 3(3x3)	(92, 17, 64)	73856
10	Aktivierungsfunktion: ELU	(92, 17, 64)	0
11	Max-Pooling (2x2)	(46, 8, 64)	0
12	Dropout (30% Rate)	(46, 8, 64)	0
13	Faltungsschicht 3(3x3)	(46, 8, 256)	73856
14	Aktivierungsfunktion: ELU	(46, 8, 256)	0
15	Max-Pooling (2x2)	(23, 4, 256)	0
16	Dropout (49% Rate)	(23, 4, 256)	0
17	Flatten	(23552)	0
18	Fully Connected Layer	(64)	1507392
19	Aktivierungsfunktion: ELU	(64)	0
20	Dropout (36% Rate)	(64)	0
21	Fully Connected Layer	(8)	520
22	Aktivierungsfunktion: Softmax	(8)	0

Tabelle 4.5: Faltungsnetzwerk Architektur aus Versuch 2.2.4 mit der Optimierungsfunktion RMSprop und einer Batch Size von 16

Entgegen der These, eine geringere Anzahl an Bildern, generiert mit Data Augmentation zu einer besseren Test-Accuracy führt, lag die Test-Accuracy in Versuch 2.2.10.2 nur bei 33.56% (siehe Versuch 2.2.10). Die Anzahl der Trainingsdaten für Versuch 2.2.10.1 betrug nur 25.092 Stück, weil die Anzahl der Bilder für jede Klasse normiert wurden. Dadurch könnte sich das schlechte Ergebnis, wegen der geringen Anzahl an Trainingsdaten, erklären lassen.

Ergebnis 2.3: Datensätze mit Wiederholung

Die Test-Accuracy, bei dem Versuch 2.3.1 siehe Tab. 4.6, war um $2,6\%$ höher, als im Versuch 2.2.7. Durch erhöhen der Anzahl der Epochen könnte sich das Ergebnis verbessern.

Für Versuch 2.3.2 war die Test-Accuracy mit $87,5\%$ deutlich höher, als im Versuch 2.2.8 mit $78,7\%$. Dies sollte daran liegen, dass die Anzahl der Trainingsdaten höher war und die Route wiederholt befahren wurde.

Versuch Nr.	Trainings-Accuracy	Tr.-Loss	Val.-Acc.	Val.-Loss	Test-Acc.	Test-Loss
2.3.1	55.24%	0.9188	60.30%	0.8369	59.39%	0.8477
2.3.2	84.26%	0.3926	87.66%	0.3584	87.50%	0.3552
2.3.3	60.82%	0.8173	62.69%	0.7547	62.31%	0.7944
2.3.4	62.99%	0.7739	65.36%	0.7547	63.61%	0.7725
2.3.5	58.80%	0.8370	57.61%	0.9308	57.96%	0.8937
2.3.6.1	71.01%	0.6366	65.56%	0.8501	64.71%	0.8923
2.3.6.2	63.86%	0.7406	60.90%	0.8270	60.53%	0.8249
2.3.7.1	82.57%	0.3961	81.78%	0.4652	63.09%	
2.3.7.2	51%	0.6955	51.66%	0.6920	63.09%	
2.3.7.3	78.95%	0.4456	77.44%	0.4459	63.09%	
2.3.8	60.36%	0.8348	61.77%	0.9206	61.64%	0.8144

Tabelle 4.6: Ergebnisse der Versuche

Abbildung 4.17: Trainingshistorie Versuch 2.3.4

Mit den gefundenen Parametern, im Versuch 2.3.3 siehe Tab. 4.7, konnte die Test-Accuracy um 2.92% in Vergleich zu Versuch 2.3.1 optimiert werden.

In Tab. 4.6 ist zu sehen, dass mit Erhöhung der Anzahl der Epochen auf 100 mit dem besten Modell die Test-Accuracy um 1.3% in Versuch 2.3.4 angehoben werden konnte. Auf Abb. 4.17 ist zu erkennen, dass ab Epoche 90 das Modell überangepasst wurde. Wie auf Abb. 4.18 zu sehen, wurden vor allem die Klassen “gut” und “schlecht” vorhergesagt. Mit den gefundenen Parametern konnte die Test-Accuracy in Vergleich zu Versuch 2.3.1 um 2.92% erhöht werden.

Zum Vergleich betrug, im Versuch 2.3.5 mit den Daten aus Abschnitt 4.2, die Test-Accuracy 57.96%, siehe Tab. 4.6. Damit war diese nur 1.17% höher, als im Versuch 2.2.7. Das Faltungsnetz war überangepasst.

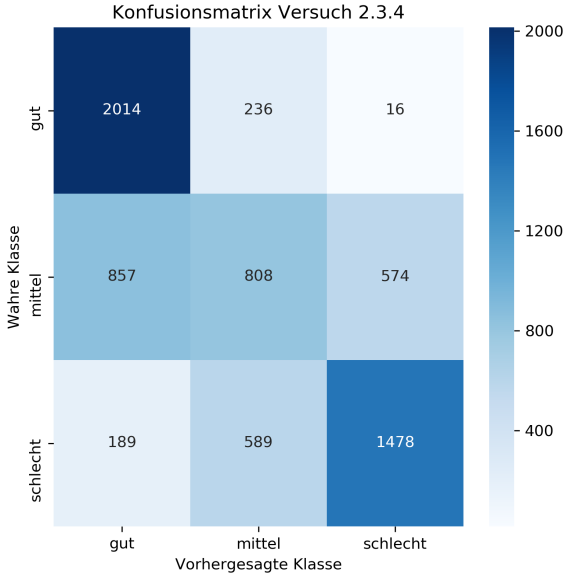

Abbildung 4.18: Konfusionsmatrix mit den Testdaten aus Versuch 2.3.4

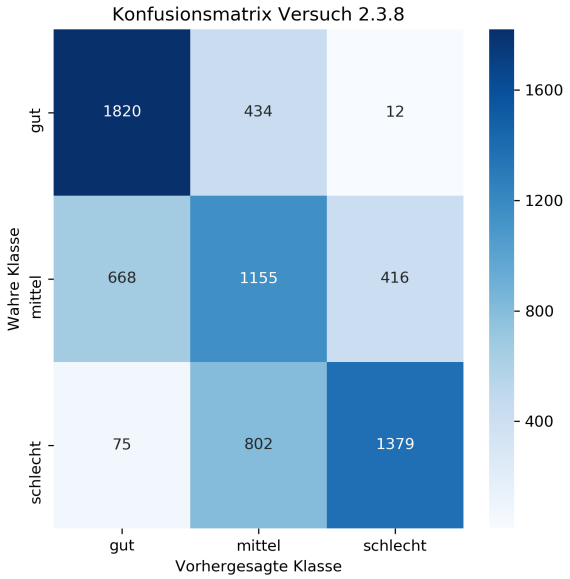

Abbildung 4.19: Konfusionsmatrix Versuch 2.3.8

Nr.	Typ	Ausgabeform	Anzahl der zu trainierenden Parameter
1	Faltungsschicht 1(3x3 Faltungskerngröße)	(368, 70, 32)	896
2	Aktivierungsfunktion: ELU	(368, 70, 32)	0
3	Max-Pooling (2x2)	(184, 35, 32)	0
4	Dropout (16% Rate)	(184, 35, 32)	0
5	Faltungsschicht 2(3x3)	(184, 35, 32)	9248
6	Aktivierungsfunktion: ELU	(184, 35, 32)	0
7	Max-Pooling (2x2)	(92, 17, 32)	0
8	Dropout (21% Rate)	(92, 17, 32)	0
9	Faltungsschicht 3(3x3)	(92, 17, 64)	18496
10	Aktivierungsfunktion: ELU	(92, 17, 64)	0
11	Max-Pooling (2x2)	(46, 8, 64)	0
12	Dropout (44% Rate)	(46, 8, 64)	0
13	Faltungsschicht 4(3x3)	(46, 8, 64)	36928
14	Aktivierungsfunktion: ELU	(46, 8, 64)	0
15	MaxPooling	(23, 4, 64)	0
16	Dropout (13% Rate)	(23, 4, 64)	0
17	Faltungsschicht 5(3x3)	(23, 4, 64)	36928
18	Aktivierungsfunktion: ELU	(23, 4, 64)	0
19	MaxPooling	(11, 2, 64)	0
20	Dropout (40% Rate)	(11, 2, 64)	0
21	Flatten	(1408)	0
22	Fully Connected Layer	(64)	90176
23	Aktivierungsfunktion: ELU	(64)	0
24	Dropout (7% Rate)	(64)	0
25	Fully Connected Layer	(3)	520
26	Aktivierungsfunktion: ELU	(3)	0

Tabelle 4.7: Faltungsnetzwerk Architektur aus Versuch 2.3.3

Auch bei dem Ergebnis des Versuchs 2.3.6 waren die Faltungsnetze an den Trainingsdaten überangepasst. Bei diesem Versuch wurden die Daten zusätzlich in zwei Geschwindigkeitsbereiche unterteilt. Im Versuch 2.3.6.1 konnte eine bessere Test-Accuracy erreicht werden, vergleiche hierzu Tab. 4.6 mit dem Versuch 2.3.4. Allerdings war der Loss der Kostenfunktion um 0.1198 höher. Die Testergebnisse aus Versuch 2.3.6.2 waren schlechter als in Versuch 2.3.4. Vermutlich war die Anzahl der Trainingsdaten jeder Klasse zu gering, aufgrund der Unterteilung der Geschwindigkeit. In dieser Versuchsanordnung konnte die These nicht bestätigt werden, dass eine Unterteilung der Geschwindigkeit, zu einer besseren Test-Accuracy führt.

Der Versuch 2.3.7 hat ergeben, dass die Test-Accuracy mit 63.09% nicht höher ist als in Versuch 2.3.4 mit 63.61%. Dies liegt vor allem daran, dass das Model, welches die Klasse "mittlere" Fahrqualität vorhersagen sollte, nur eine Validation-Accuracy von 51.66% hatte. Mit der gegebenen Zuordnung der Fahrqualität ist es für diese Aufgabe also nicht sinnvoll, die Methode "one-versus-the-rest" zu verwenden.

Mit dem Ergebnis aus dem Versuch 2.3.8 konnte keine höhere Test-Accuracy mit 61.64% erreicht werden als in Versuch 2.3.4. Dafür wurde die Klasse "mittel" beim Testen mit 57% korrekt vorhergesagt (siehe Abb. 4.19). In dem Versuch 2.3.4 wurde diese mit 36% korrekt vorhergesagt.

4.5 Darstellung in OpenStreetMap

Die Daten aus Datensatz 52 wurden auf der Höltigbaum Route erfasst (siehe Abb. 4.20). Ziel war es, ein Beispiel für die weitere Verarbeitung der klassifizierten Daten zu zeigen. Für die Darstellung in OSM wurde ein Python Script geschrieben mit welchem es möglich ist, die erfassten Daten auf den OSM Entwicklerserver hochzuladen. Um mit einem Script automatisiert, erfolgreich Daten auf den OSM Server zu laden, müssen Richtlinien eingehalten werden, weshalb hier die Entwickler API genutzt wurde. Als Bibliothek wurde, die Wrapper OsmApi benutzt. Bei einem Abstand länger als $20m$ wurde jeweils eine Linie hinzugefügt, die einen Wegabschnitt repräsentiert. Jeder Abschnitt hat das OSM Attribut "smoothness". Die am häufigsten vorkommende Klasse von "gut", "mittel", und "schlecht" wurde dem "smoothness" Attribut "good", "intermediate" oder "bad" zugewiesen.

Es ist standardmäßig nicht möglich die Farbe eines Weges in OSM zu ändern. Dafür wird ein anderes Kartendesign benötigt. Deshalb wurde in Abb. 4.21 und Abb. 4.22 die gefahrene Route farblich mit Matplotlib dargestellt. Auf der Abb. 4.21 sind die Label mit dem Fuzzy Logik-Versuch 2.1.6 zu sehen. Zum Vergleich wurde die Klassifizierung mit dem Faltungsnetzwerk aus Versuch 2.3.8 in der Abb. 4.22 dargestellt. Die Test-Accuracy lag bei 63.80%. Trotz der geringen Accuracy fällt auf, dass die Abbildungen sich auf den ersten Blick ähneln. Bei genaueren Betrachten wurden einige Wegabschnitte mit einem falschen Attribut versehen. Zum Beispiel kommt in der Abb. 4.21 oben links der Wert "bad" vor und in Abb. 4.22 fehlt dieser. Ein Test mit Datensatz 39, einer anderen befahrenen Route mit überwiegend befestigten Radwegen, hat eine Test-Accuracy von 50.40% ergeben für das Faltungsnetzwerk aus Versuch Versuch 2.3.8. Wie erwartet war die Test-Accuracy für eine andere Route geringer.

Wünschenswert ist eine gleichwertige Klassifizierung der Radwege mit unterschiedlichen Fahrradtypen. Damit eine Datenerfassung nicht nur mit einem bestimmten Fahrradtyp vor-

Abbildung 4.20: OSM Kartenausschnitt mit den Testdaten aus Datensatz 52

Quelle: https://api06.dev.openstreetmap.org, Kartografie Lizenz: CC BY-SA 2.0

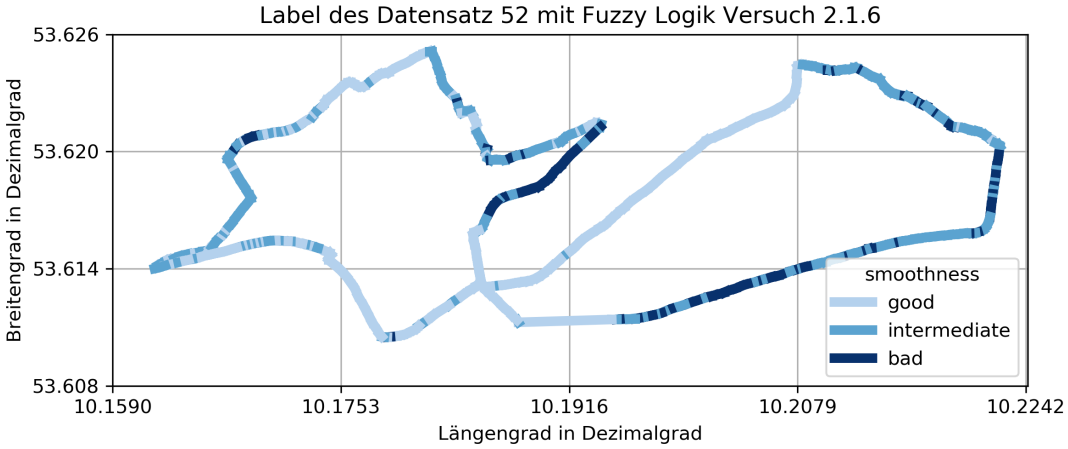

Abbildung 4.21: Route mit OSM “smoothness” Attributen

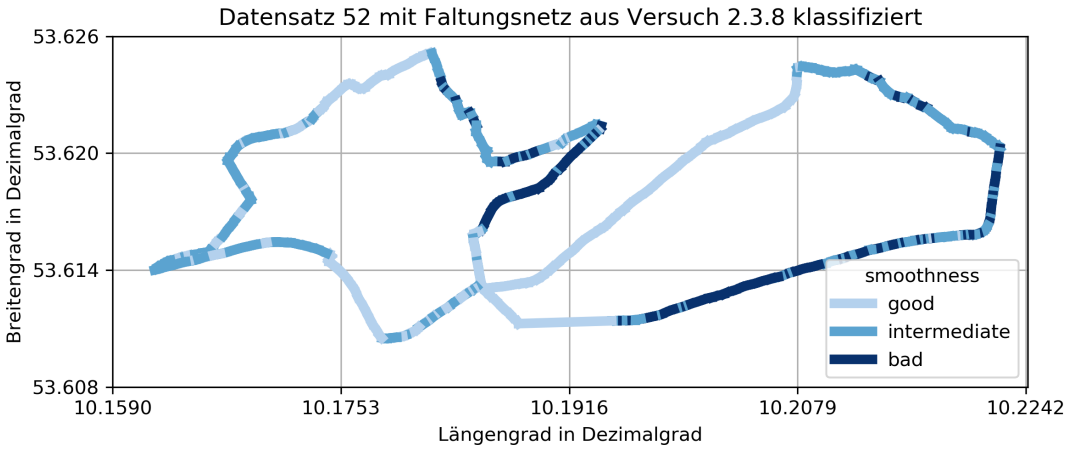

Abbildung 4.22: Klassifizierte “smoothness”-Tags für Datensatz 52

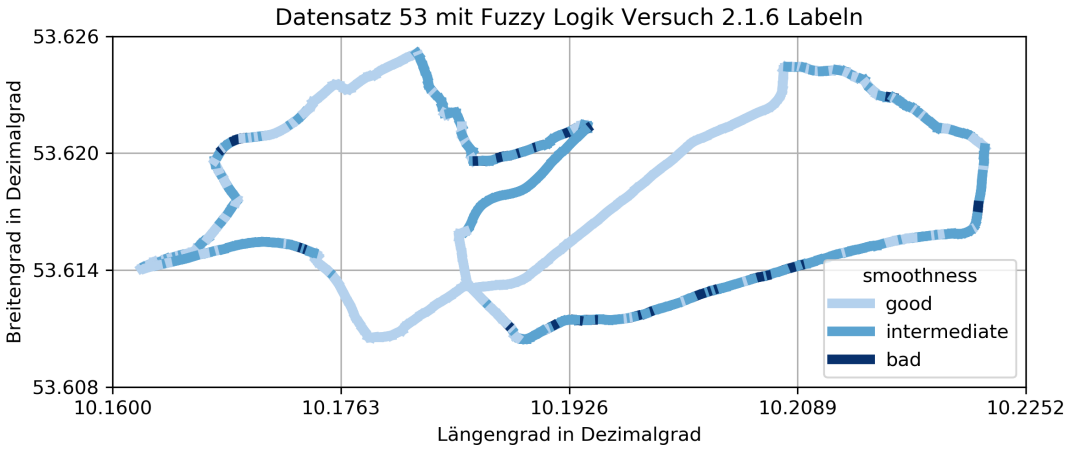

Abbildung 4.23: Label zugeordnet für Datensatz 53

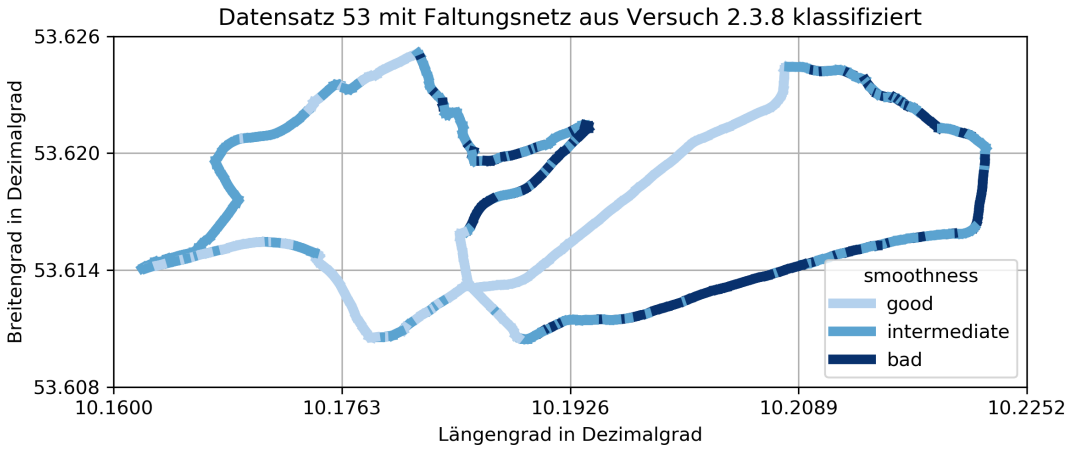

Abbildung 4.24: Datensatz 53 wurde mit einem Hardtail MTB erfasst

genommen werden kann. Die Bilder, um die Faltungsnetzwerke aus Kapitel 4 zu trainieren, wurden mit einem 28 Zoll Crossrad aufgenommen. Zu Vergleichszwecken wurde die gleiche Route, wie in Datensatz 52 mit einem Hardtail MTB befahren. Also einem Fahrrad, welches eine Federung für das Vorderrad besitzt. In diesen Zusammenhang wurde erwartet, dass nach Anwendung, der Fuzzy Logik das Label “schlecht” bezogen auf die Gleichmäßigkeit des Radweges seltener vorkommt. Wie auf der Abb. 4.23 zu sehen, trifft dies im Vergleich mit Abb. 4.21, bei welcher die Bilder mit einem 28 Zoll Crossrad aufgenommen wurden, zu. Allerdings wurde bei der Klassifizierung der Bilder mit dem Faltungsnetzwerk aus Versuch 2.3.8 auch die Klasse “schlecht” (“bad”) häufiger vorhergesagt. Vergleiche hierzu Abb. 4.23 mit Abb. 4.24. Ein Vorteil bei der Verwendung einer Klassifizierung, anhand von Bildern ist die Generalisierung bei der Verwendung von unterschiedlichen Fahrradtypen, wobei auf eine einheitliche Höhe bei der Montage des Smartphones am Lenker geachtet werden muss.

5 Diskussion

Im Versuch 1 wurde mit der (geringen) Anzahl von 2000 Trainingsdaten, welche manuell gelabelt wurden, ein Faltungsnetzwerk trainiert. Das Testergebnis war für die gleiche Strecke, auf welcher die Trainingsdaten gelabelt wurden, mit $\approx 85\%$ Test-Accuracy relativ erfolgreich. Jedoch konnte für eine unbekannte Strecke, befahren mit einem Hardtail MTB statt mit einem Crossrad, nur eine Test-Accuracy von 61.70% erzielt werden.

Hieraus entstand die Motivation, die Bilder, automatisiert zu labeln (siehe Kapitel 4) in Versuch 2, um einen größeren Datensatz zur Verfügung zu haben. Zunächst einmal wurden Datensätze ohne Wiederholung der gefahrenen Route, dann andere ohne Wiederholung unterschieden. Bei der Zuordnung der Fahrqualität mit Fuzzy Logik wurde die Standardabweichung der Z-Achsen-Beschleunigung genutzt.

Das Ergebnis aus Versuch 2.2, siehe Abschnitt 4.2, hat gezeigt, dass die Zwischenstufen von "sehr guter" bis "sehr schlechter" Fahrqualität bei den durchgeführten Faltungsnetzwerkversuchen nur müßig Klassifizieren lassen. Das beste Ergebnis wurde in dem Versuch 2.2.5 mit $\approx 42\%$ für 4 unterschiedliche Klassen erzielt.

In dem Versuch 2.3.7 wurde gezeigt, dass bei der Unterscheidung von 3 Klassen mit der Methode "one-versus-the-rest", vor allem die Klasse "mittel" mit einer Validation-Accuracy von 51.66%, schlecht vorhergesagt wurden konnte. Zum Vergleich konnte die Klasse "gut" mit einer Test-Accuracy von 81.78% und die Klasse "schlecht" mit 77.44% vorhergesagt werden. Damit konnte die Beobachtung aus der Vergleichsarbeit [13] bestätigt werden, bei welcher eine differenziertere Klassifizierung bei der Unterscheidung zwischen Radwegen mit vielen und wenigen Unebenheiten möglich ist .

Bei den Versuch 2.3, siehe Abschnitt 4.2, hat sich zudem gezeigt, dass durch Wiederholung der Route eine höhere Accuracy erreicht werden kann. Als Beispiel hierzu eignet sich ein Vergleich der Ergebnisse von dem Versuch 2.3.4 (63.61%) mit dem Versuch 2.2.7 (56.79%), in denen jeweils 3 Klassen existierten. Ein Test mit den Datensatz 52, der auf einer wiederholt befahrenen Route aufgenommen wurde, hat eine Test-Accuracy von 63.80% erreicht (siehe Abschn. 4.5). Dabei wurden die Bilder mit dem Faltungsnetzmodell aus Versuch 2.3.8 klassifiziert.

Die Anzahl der Bilder jeder Klasse sind ungleich verteilt, weil Radwege mit vielen Unebenheiten seltener vorkamen (siehe Abb. 4.13). Die Anzahl der Trainingsdaten, zu erhöhen mit Data Augmentation hat dazu geführt, dass die so trainierten Modelle überangepasst waren (siehe Versuch 2.2.9).

Eine Geschwindigkeitsunterteilung der Bilder, in zwei Geschwindigkeitsbereiche, wie in Versuch 2.3.6, hat zu einen ähnlichen Ergebnis wie ohne dieser geführt (siehe Versuch 2.3.4). Problematisch ist die geringe Anzahl an Trainingsdaten jeder Klasse durch die Unterteilung der Daten. In weiteren Experimenten könnten unterschiedliche Grenzwerte für einen Geschwindigkeitsbereich ausprobiert werden, um eine präzisere Klassifizierung zu ermöglichen.

Insgesamt hatte das beste Ergebnis in Versuch 2.3.2 eine Test-Accuracy von 87.50% erzielt. In diesem Versuch wurde nur zwischen den beiden Klassen “gut” und “schlecht” unterschieden. Die Klasse “mittelmäßig”, wurde hier aus den Trainings-und Testdaten entfernt.

Ein Nachteil bei der Herangehensweise bei diesem System mit der Klassifizierung von Bildern ist, dass nicht kontinuierlich Bilder von der befahrenen Oberfläche aufgenommen werden können. Die erfassten Messwerte müssen zu einem bestimmten Bild zugeordnet und anschließend abgespeichert werden. Auch der Speicherbedarf, um die Bilder auf dem Smartphone abzuspeichern, ist relativ hoch. Dieser Bedarf könnte noch minimiert werden, wenn nur die Grauwerte und der relevante Bildausschnitt abgespeichert wird.

Wie in Abschnitt. 4.5 gezeigt wurde, ist es auch möglich mit einem 26 Zoll Hardtail MTB eine ähnliche Klassifizierung zu erzielen wie mit dem Crossrad bei der Trainingsdatenerfassung. Für die Erfassung der Trainingsdaten wurde ein 28 Zoll Crossrad ohne Federung genutzt. Aus diesem Grund ist eine Generalisierung für unterschiedliche Fahrräder bei der Klassifizierung mit Faltungsnetzwerken ein Vorteil.

Ausblick

Die Regelbasis und Zuordnungsfunktionen wurden für diese Arbeit mit Fuzzy Logik experimentell und iterativ bestimmt (siehe Versuch 2). Hier sind weitere Untersuchungen nötig, um möglichst klare Regeln zu finden, welche es auch ermöglichen Zwischenabstufungen für die Gleichmäßigkeit der Oberfläche zu bestimmen. Zu diesem Zweck können auch die Merkmale Standardabweichung der Y-Achsen-Beschleunigung, Nick-Winkel, Belichtungszeit sowie der gemessenen Geschwindigkeit hilfreich sein. Die Y-Achsen-Messwerte des Beschleunigungssensors könnten genutzt werden, um Störfaktoren beim Abbremsen des Fahrrads zu filtern, welche die Messwerte der Z-Achsen-Beschleunigung beeinflussen. Der Nick-Winkel kann als zusätzlicher Eingangswert für die Zuordnung der Erschütterung genutzt werden. Bei einer langen Belichtungszeit verringert sich die Bildschärfe.

Eine Optimierung der Klassifizierung könnte sich aus komplexeren Faltungsnetzwerken ergeben. Hierfür könnten vortrainierte Faltungsnetze nützlich sein, bei denen nur die obersten Faltungsschichten trainiert werden. Ein solches Modell wäre zum Beispiel Xception [5].

Literaturverzeichnis

[1] Jürgen Adamy. *Fuzzy Logik, Neoronale Netze Evolutionäre Algorithmen.* Shaker Verlag, 4 edition, 2015.

[2] David Bartlett. *Essentials of positioning and location technology.* The Cambridge wireless essentials series. Cambridge [u.a.] : Cambridge Univ. Press, 2013.

[3] Y Bengio and Yann Lecun. Convolutional networks for images, speech, and time-series. 11 1997.

[4] Francois Chollet. Building powerful image classification models using very little data. *The Keras Blog, https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html (aufgerufen am 07.2018)*, 2016.

[5] François Chollet. Xception: Deep learning with depthwise separable convolutions. *CoRR*, abs/1610.02357, 2016.

[6] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *CoRR*, abs/1511.07289, 2015.

[7] Teufel D. Entwicklung und potentiale des fahrrad-verkehrs. *Umwelt- und Prognose-Institut Heidelberg e. V. (Hrsg.), UPI-Bericht Nr. 41, Heidelberg*, 1997.

[8] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization.

[9] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow.* O'Reilly Media, 2017.

[10] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS,10). Society for Artificial Intelligence and Statistics*, 2010.

[11] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[12] Divya Bansal Gunjan Chugh and Sanjeev Sofat. Road condition detection using smartphone sensors: A survey. *International Journal of Electronic and Electrical Engineering*, 7(6):595–602, 2014.

[13] Marius Hoffmann, Michael Mock, and Michael May. Road-quality classification and bump detection with bicycle-mounted smartphones. In *Proceedings of the 3rd International Conference on Ubiquitous Data Mining - Volume 1088*, UDM'13, pages 39–43, Aachen, Germany, Germany, 2013. CEUR-WS.org.

[14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[15] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. *CoRR*, abs/1706.02515, 2017.

[16] M. Meschik. *Planungshandbuch Radverkehr*. Springer Vienna, 2008.

[17] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *CoRR*, abs/1712.04621, 2017.

[18] Sebastian Ruder. An overview of gradient descent optimization algorithms. *http://ruder.io/, (aufgerufen am 07.2018)*, 2016.

[19] Josh Warner, Jason Sexauer, scikit fuzzy, twmeggs, Alexandre M. S., Aishwarya Unnikrishnan, Guilherme CastelÃ£o, Fernando Batista, The Gitter Badger, and Himanshu Mishra. Jdwarner/scikit-fuzzy: Scikit-fuzzy 0.3.1, October 2017.

[20] Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012.

[21] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg 30.08.18 _____

Ort Datum Unterschrift im Original