Detektion von Verkehrszeichen mit Deep Learning

Çağatay Parlar
Turkisch-Deutsche Univesität
Studienrichtung Mechatronik
Istanbul, Turkei
e170501033@stud.tau.edu.tr

Kaan Sümer
Turkisch-Deutsche Universität
Studienrichtung Mechatronik
Istanbul, Turkei
e160501119@stud.tau.edu.tr

Yiğit Atalay
Turkisch-Deutsche Universität
Studienrichtung Mechatronik
Istanbul, Turkei
e170501038@stud.tau.edu.tr

Volkan Aslan
Turkisch-Deutsche Universität
Studienrichtung Mechatronik
Istanbul, Turkey
e160501126@stud.tau.edu.tr

Onur Akgün
Turkisch-Deutsche Universität
Wissenschaftliche Mitarbeiter
Istanbul, Turkey
akgun@tau.edu.tr

Dr. Naime Özben Önhon
Turkisch-Deutsche Universität
Elektrotechnik
Istanbul, Turkey
onhon@tau.edu.tr

***Abstract*—In dieser Forschung wurde ein Deep-Learning-Modell erstellt, das die Verkehrszeichen auf einem Foto erkennt und diese Verkehrszeichen in 4 verschiedene Klassen einordnet. In diesem Beitrag werden Informationen zum verwendeten Datensatz „German Traffic Sign Detection Benchmark“ gegeben. Dann wurden die Verbesserungen, die wir am Datensatz und der Software und Methoden vorgenommen haben, die wir bei diesen Verbesserungen verwendet haben, erwähnt. Es werden Informationen über die "Google Colab"-Umgebung gegeben, in der die Codes geschrieben werden und die "Pytorch"-Bibliothek, die bei der Vorbereitung der Deep-Learning-Codes verwendet wird. Es wurden Informationen über die trainierten "Yolov5"-Modelle gegeben und die Ergebnisse verschiedener "Yolov5"-Modelle verglichen. Die Ergebnisse, die wir erhalten haben, zeigen, dass das Training von "Yolov5"-Modellen mit der Transfer-Lernmethode sehr erfolgreiche Ergebnisse bei der Erkennung von Verkehrszeichen liefert.**

Keywords—Deep Learning, Object detection, Neural networks, Convolutional neural networks

I. INTRODUKTION

Fahrerunterstützung, die in Fahrzeugen unter den heutigen technologischen Bedingungen verwendet wird. Systeme sind weit verbreitet. Ziel ist es, die Unfälle, die durch die Zunahme der Anzahl von Fahrzeugen mit den in den Fahrzeugen eingesetzten Fahrerassistenzsystemen auftreten, zu reduzieren. An der Spitze der Fahrerassistenzsysteme stehen automatische Verkehrszeichenerkennungssysteme, die die Verkehrszeichen auf den Straßen erkennen und den Fahrer entsprechend der Beschilderung über den Straßenzustand informieren.

Dazu wurde der Datensatz der Deutschen Verkehrszeichenerkennung verwendet. Verkehrszeichen werden in 4 verschiedene Klassen eingeteilt. Diese Klassen sind:

- Prohibitory

  runder, weißer Grund mit rotem Rand
- Danger

  dreieckig, weißer Grund mit rotem Rand
- Mandatory

  runder, blauer Grund
- Other

Abbildung 1 Prohibitory (a), Danger (b) und Mandatory (c)

II. STAND DER TECHNIK

A. Deep Learning Method for Traffic Sign Detection

n dieser Studie wird ein effizientes Verfahren zur Erkennung von Verkehrszeichen auf Verkehrsszenenbildern vorgeschlagen, die mit dem Kamerasensor am Fahrzeug aufgenommen wurden.Der Algorithmus, der die Methode erstellt, besteht aus 3 Schritten. Zuerst wird eine Bildsegmentierung basierend auf Deep Learning durchgeführt und einige Regionsvorschläge werden aus dem Bild extrahiert. Dann werden einige der Regionenvorschläge, die weniger wahrscheinlich Verkehrszeichen aufweisen, durch Ausführen von Hypothesentests eliminiert. Im letzten Schritt wird ein richtig trainierter Convolutional Neural Networks (CNN)-Klassifikator für die verbleibenden Regionsvorschläge verwendet und Regionen, die vom Klassifikator stark als Verkehrszeichen erkannt werden, werden akzeptiert.In dieser Studie wurde GTSDB (German Traffic Sign Detection Benchmark (GTSDB)) als Datensatz verwendet. Laut den Testergebnissen wurde getestet, dass bei den Presicion- und Recallslmetriken 90% - 95% Ergebnisse erzielt wurden und die Verarbeitungszeit pro Bild 0,41 Sekunden betrug.

Abbildung 2 Testergebnis aus GTSDB-Datensatz [1]

B. *A Real-Time Chinese Traffic Sign Detection Algorithm Based on Modified YOLOv2*

In dieser Studie wird ein chinesischer Verkehrszeichenerkennungsalgorithmus vorgestellt, der auf einem tiefen Faltungsnetzwerk basiert. Ein End-to-End-Faltungsnetzwerk mit YOLOv2 wird vorgeschlagen, um eine chinesische Verkehrszeichenerkennung in Echtzeit zu erreichen.Unter Berücksichtigung der Eigenschaften von Verkehrszeichen werden mehrere 1×1-Faltungsschichten in den Zwischenschichten des Netzwerks und reduzierte Faltungsschichten in den oberen Schichten verwendet, um die Rechenkomplexität zu reduzieren. Um kleine Verkehrszeichen effektiv zu erkennen, werden die Einfahrtsbilder in dichte Raster aufgeteilt, um feinere Merkmalskarten zu erhalten. In dieser Studie wurden zwei Datensätze verwendet, nämlich der Chinesische Verkehrszeichendatensatz (CTSD) und GTSDB.Aus den erhaltenen Ergebnissen geht hervor, dass das vorgeschlagene Verfahren schnell und robust ist. Die schnellste erreichte Erkennungsgeschwindigkeit beträgt 0,017 s pro Bild.

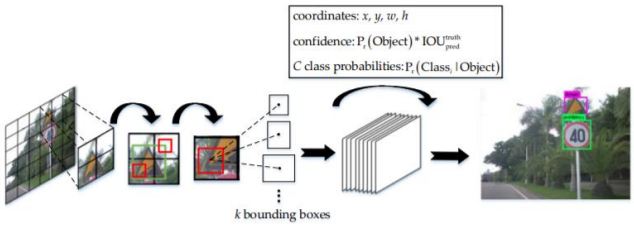

Abbildung 3 Der chinesische Verkehrszeichenalgorithmus [2]

III. METHODEN

A. *Datensatz*

Wir haben den GTSDB-Datensatz verwendet, um unser Modell zu trainieren und zu testen und unsere Ergebnisse numerisch auszuwerten, und wir haben die Ergebnisse nach einigen Metriken analysiert. Der GTSDB-Datensatz ist ein weit verbreiteter zugänglicher Datensatz zum Bewerten und Testen von TLT- und TLS-Systemen. Datensatz Houben et al. (2013) erstellt von.

Im Datensatz befinden sich 900 in Deutschland aufgenommene Bilder (Verkehrsbildszene von der Bordkamera) mit einer Auflösung von 1360x800 Pixeln, wobei die Koordinaten und Klassenangaben der Verkehrszeichen in den Bildern als zu verwendende Labels angegeben sind beim Training von Algorithmen. Verkehrszeichen gehören zu 43 verschiedenen Klassen und die Verteilungsrate von Trainings- und Testdatensätzen auf 900 Bilder beträgt 66,6%-33,4 %, wobei 600 Bilder trainiert und 300 Bilder getestet werden.

1) *Bildformat*

Die Bilder enthalten null bis sechs Verkehrszeichen. Auch wenn sich im Bild ein Verkehrszeichen befindet, darf es jedoch nicht zu den wettbewerbsrelevanten Kategorien (Verbot, Gefahr, Pflicht) gehören. Bilder werden im PPM-Format gespeichert. Die Größen der Verkehrszeichen in den Bildern variieren von 16x16 bis 128x128. Verkehrszeichen können in jeder Perspektive und unter allen Lichtverhältnissen erscheinen.

Abbildung 4 Bildformat

2) *Anmerkungsformat*

Anmerkungen werden in CSV-Dateien bereitgestellt. Felder werden durch ein Semikolon (;) getrennt. Sie enthalten folgende Informationen:

- Dateiname: Dateiname des Bildes, für das die Anmerkungen gelten
- Die Region of Interest (ROI) des Verkehrszeichens im Bild.
- linke Bildspalte des ROI
- oberste Bildzeile des ROI
- Bildspalte ganz rechts des ROI
- unterste Bildzeile des ROI
- ID, die die Klasse des Verkehrszeichens angibt

```
00000.ppm;774;411;815;446;11
00001.ppm;983;388;1024;432;40
00001.ppm;386;494;442;552;38
00001.ppm;973;335;1031;390;13
00002.ppm;892;476;1006;592;39
```

Abbildung 5 Anmerkungsformat

B. *Google Colab*

Colab ist ein kostenloser Cloud-Dienst, den Google dem Endbenutzer für die Entwicklung von Machine Learning und Deep-Learning-Studien anbietet. Dank Remote-Zugriff ist es möglich, Deep-Learning-Modelle mit Unterstützung von GPU, Tensor Processing Unit (TPU) und Central Processing Unit (Cpu) in der Jupyter-Notebook-Umgebung zu trainieren. Es enthält auch vorgefertigte Deep-Learning-Bibliotheken wie tensorflow, Keras, Pytorch und OpenCV.

Deep-Learning-Anwendungen können auf dem Tesla K80-Modell des Unternehmens Nvidia für Entwickler von Google Colab-Diensten entwickelt werden. In den in Abbildung 3.4 zu sehenden Grafikkarten befinden sich zwei GPUs, die die Kapazität haben, mit 2 großen Datensätzen gleichzeitig zu arbeiten. Dank 4992 CUDA-Parallelverarbeitungskernen laufen Anwendungen 10-mal schneller als Systeme mit nur CPUs.

C. *Yolo-Modell*

Die ersten drei YOLO-Versionen wurden 2016, 2017 und 2018 veröffentlicht. Im Jahr 2020 wurden jedoch innerhalb

weniger Monate drei Hauptversionen von YOLO mit den Namen YOLO v4, YOLO v5 und PP-YOLO veröffentlicht.

1) Yolo v3

You only look once (YOLO) ist ein hochmodernes Echtzeit-Objekterkennungssystem. Auf einem Pascal Titan X verarbeitet es Bilder mit 30 FPS und hat einen mAP von 57,9% auf COCO Test-Dev. Frühere Erkennungssysteme verwenden Klassifizierer oder Lokalisierer um, um eine Erkennung durchzuführen. Sie wenden das Modell an mehreren Stellen und Maßstäben auf ein Bild an. Bereiche des Bildes mit hoher Bewertung werden als Erkennungen betrachtet.

2) Yolov4

YOLO v4 basiert ebenfalls auf dem Darknet und hat einen AP-Wert von 43,5 Prozent im COCO-Datensatz zusammen mit einer Echtzeitgeschwindigkeit von 65 FPS auf dem Tesla V100 erreicht und übertrifft damit die schnellsten und genauesten Detektoren in Bezug auf Geschwindigkeit und Genauigkeit.

Im Vergleich zu YOLO v3 sind AP und FPS um 10 bzw. 12 Prozent gestiegen.

3) Yolov5

Nach der Veröffentlichung von YOLO v4 wurde innerhalb von nur zwei Monaten eine weitere Version von YOLO namens YOLO v5 veröffentlicht! Es stammt von Glenn Jocher, der in der Community bereits für die Entwicklung der beliebten PyTorch-Implementierung von YOLO v3 bekannt ist.

Am 9. Juni 2020 gab Jocher bekannt, dass seine YOLO v5-Implementierung öffentlich veröffentlicht wird und für die Verwendung in neuen Projekten empfohlen wird.

YOLO v5 unterscheidet sich von allen anderen früheren Releases, da es sich hierbei um eine PyTorch-Implementierung und nicht um eine Abzweigung aus dem ursprünglichen Darknet handelt. Genau wie YOLO v4 hat das YOLO v5 ein CSP-Backbone und einen PA-NET-Hals. Zu den wichtigsten Verbesserungen gehören die Erweiterung von Mosaikdaten und automatisch lernende Bounding-Box-Anker. YOLO v5 ist schnell und leichtgewichtig als YOLO v4, während die Genauigkeit dem YOLO v4-Benchmark entspricht.

a) Yolov5 Varianten

YOLOv5 ist in vier Modellen erhältlich, nämlich s, m, l und x, von denen jedes unterschiedliche Erkennungsgenauigkeit und Leistung bietet, wie unten gezeigt.

Model	AP^{val}	AP^{test}	AP_{50}	$Speed_{GPU}$	FPS_{GPU}		params	FLOPS
YOLOv5s	37.0	37.0	56.2	**2.4ms**	**416**		7.5M	13.2B
YOLOv5m	44.3	44.3	63.2	3.4ms	294		21.8M	39.4B
YOLOv5l	47.7	47.7	66.5	4.4ms	227		47.8M	88.1B
YOLOv5x	**49.2**	**49.2**	**67.7**	6.9ms	145		89.0M	166.4B
YOLOv5x + TTA	**50.8**	**50.8**	**68.9**	25.5ms	39		89.0M	354.3B
YOLOv3-SPP	45.6	45.5	65.2	4.5ms	222		63.0M	118.0B

Abbildung 6 Yolov5 Varianten [3]

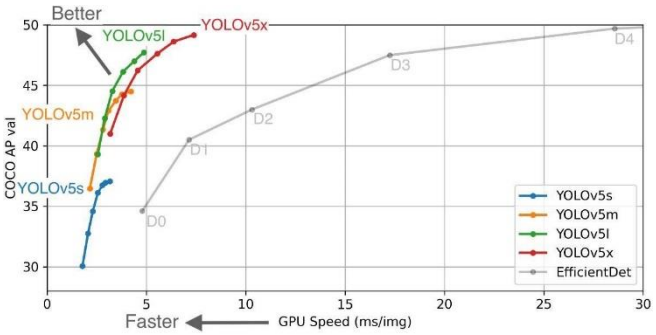

Abbildung 7 Yolov5 Varianten 2 [3]

IV. RESULTATE UND DISKUSSION

GTSDB-Daten sind nicht im Yolo-Format gekennzeichnet. Als erstes mussten diese Daten für das Yolo-Format geeignet gemacht werden. Dafür stehen im Internet verschiedene Tools zur Verfügung. Wir haben die Website "Roboflow" verwendet. Um diese Site zu verwenden, haben wir zuerst die ppm-Dateien in jpg-Dateien konvertiert, dann die jpg-Dateien auf die Roboflow-Site hochgeladen und sie im Yolo-Format mit Tags versehen.

Zur Verkehrszeichenerkennung wurden drei verschiedene Verfahren ausprobiert. Diese Methoden werden einzeln beschrieben und erklärt.

A. Training das Yolov5s-Modell mit 45 Klassen

Der erste Datensatz, den wir nach der Beschriftung erstellt haben, bestand aus 45 verschiedenen Klassen. Die Verteilung dieser Klassen ist in der folgenden Abbildung dargestellt.

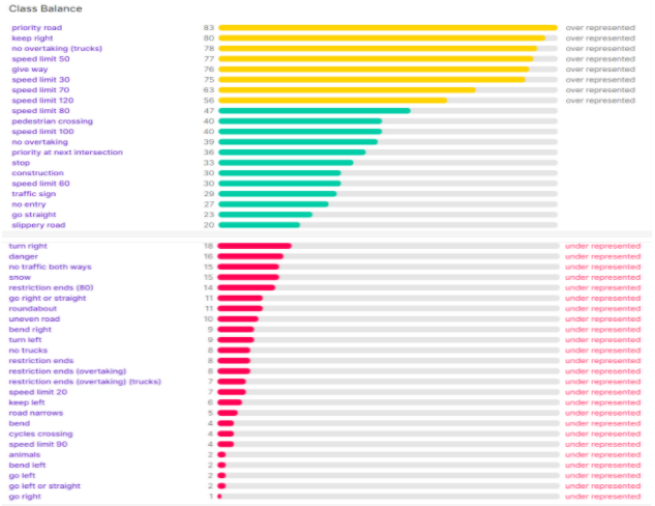

Abbildung 8 Erste Datensatz

Der Datensatz enthält ungefähr 1200 Fotos. Diese Zahl ist sehr unzureichend, um 45 Klassen zu trainieren. Daher waren auch die erhaltenen Ergebnisse nicht gut. Unten sehen Sie die Ergebnisse des Trainings des Yolov5s-Modells mit diesem Datensatz.

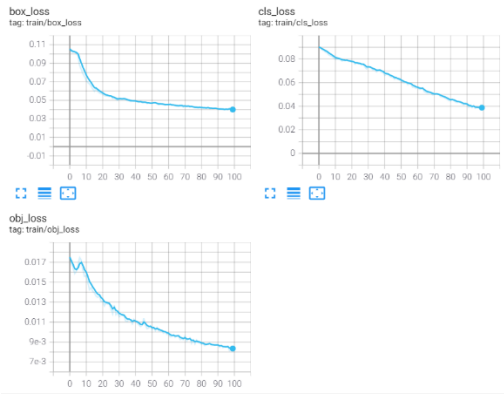

Abbildung 9 Trainingsfehler von erster Methode

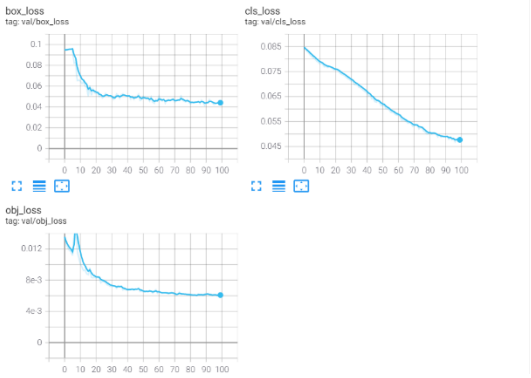

Abbildung 10 Validierungsfehler von erster Methode

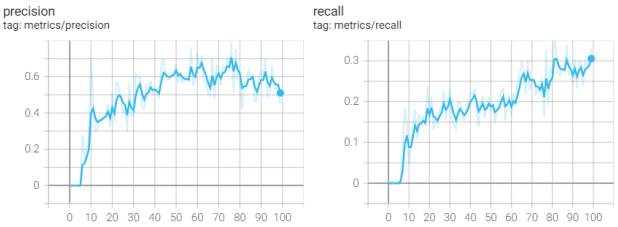

Abbildung 11 Precision und Recall von erster Methode

B. Training das Yolov5s-Modell mit 4 Klassen

Da bei Verwendung von 45 verschiedenen Klassen sehr schlechte Ergebnisse erzielt wurden, wurden 4 verschiedene Klasseneinteilungen vorgenommen. Was diese Klassen sind, wird in der Einführung erklärt. Sie können die Verteilung dieser Klassen im Bild unten sehen.

Abbildung 12 Unterteilung von zweite Datensatz

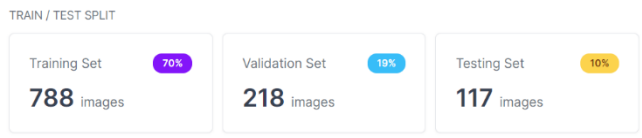

Abbildung 13 Train- Validierung- Test-Split von zweite Datensatz

Relativ bessere Ergebnisse wurden erzielt, wenn dieser Datensatz mit dem yolov5s-Modell trainiert wurde. Aber die erhaltenen Ergebnisse waren immer noch unzureichend.

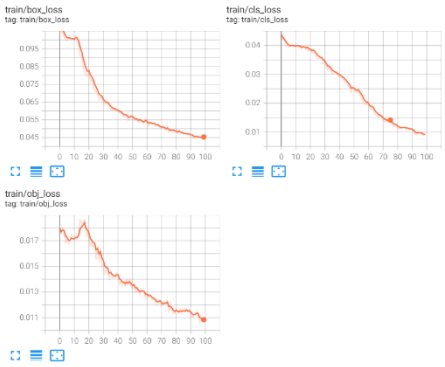

Abbildung 14 Trainingsfehler von zweiter Methode

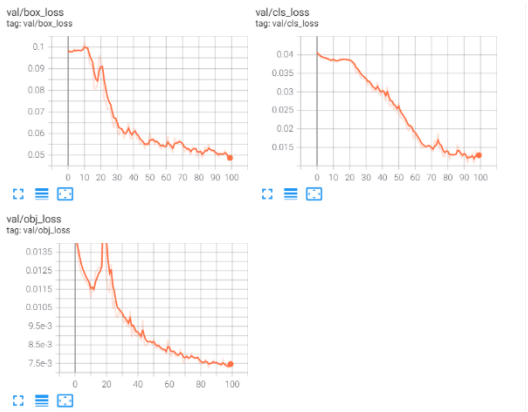

Abbildung 15 Validierungsfehler von zweiter Methode

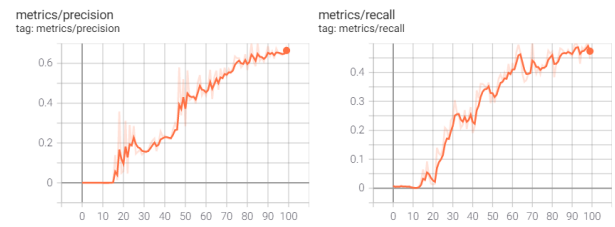

Abbildung 16 Precision und Recall von zweiter Methode

Die Testergebnisse sind wie folgt. Wie Sie sehen können, enthält es zu viele falsch positive und falsch negative Ergebnisse. Er sieht das Verkehrsschild im Bild unten als zur Klasse „Prohibitory“ gehörend, tatsächlich gehört es aber zur die Klasse „Other“.

Abbildung 17 Test-Ergebnis von zweite Method

C. *Fine-Tuning das Yolov5l-Modell*

Die Transfer-Lernmethode wurde verwendet, um die Ergebnisse zu verbessern. Einige Schichten des Yolov5l-Modells wurden eingefroren und "fine-tuning" gemacht. Die erzielten Ergebnisse waren viel besser. Sie können die Trainingsergebnisse unten sehen.

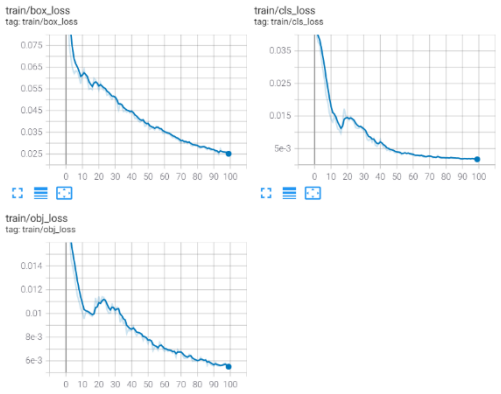

Abbildung 18 Trainingsfehler von dritter Methode

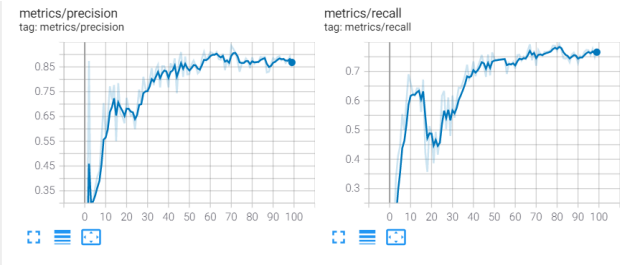

Abbildung 19 Precision und Recall von dritter Methode

Sie können die Testergebnisse unten sehen. Die Anzahl falsch positiver und falsch negativer Ergebnisse wurde stark reduziert.

Abbildung 20 Test-Ergebnis von dritter Methode

Abbildung 21 Andere Test-Ergebnis von dritter Methode

V. FAZIT

Die Schlussfolgerungen, die aus diesem Projekt gezogen werden können, sind:

- GTSDB-Daten reichen nicht aus, um 45 verschiedene Klassen zu klassifizieren.
- Wenn 4 verschiedene Klassen klassifiziert werden sollen, führt die „Fine Tuning“ zu sehr guten Ergebnissen.

Die mit dem Yolov5s-Modell und dem Yolov5l-Modell erhaltenen Ergebnisse sind wie folgt. Wie zu sehen ist, hat das Yolov5s-Modell falsche Ergebnisse gefunden, während das Yolov5l-Modell richtige Ergebnisse gefunden hat.

Abbildung 22 Test-Ergebnis von Yolov5s-Modell

Abbildung 23 Test-Ergebnis von Yolov5l-Modell

WISSEN

Wir danken Onur Akgün, Wissenschaftlicher Mitarbeiter der Türkisch-Deutschen Universität, der uns während des gesamten Projekts jede Woche getroffen und uns dort geholfen hat, wo wir festgefahren sind.

[1] M. Çetinkaya und T. Acarman, „Trafik İşaret Levhası Tespiti için Derin Öğrenme Yöntemi,“ *Akıllı Ulaşım Sistemleri ve Uygulamaları Dergisi,* Bd. 3, Nr. 2, pp. 140-157, 2020.

[2] Z. J, H. M, J. X und L. X, „A Real-Time Chinese Traffic Sign Detection Algorithm Based on Modified YOLOv2,“ 10 2017. [Online]. Available: https://doi.org/10.3390/a10040127. [Zugriff am 22 06 2021].

[3] ultralytics, „Github,“ [Online]. Available: https://github.com/ultralytics/yolov5. [Zugriff am 25 06 2021].