PROJEKT ZUR KLASSIFIZIERUNG VON VERKEHRSZEICHEN

Aybüke Bayramiç
Informatik
Türkisch-Deutsche Universität
e160503111@stud.tau.edu.tr

Abstrakt

In diesem Projekt werden wir "Convolutional Neural Networks" mit Keras verwenden, um Verkehrszeichen zu klassifizieren und identifizieren. Mit dem deutschen Verkehrszeichendatensatz werden wir ein Modell trainieren, um Verkehrszeichen in natürlichen Bildern zu dekodieren.

1 Einführung

Verkehrszeichen haben einen wichtigen Platz in unserem Leben. Die Mehrheit von unsere Tage geht auf den Straßen. Daher müssen alle Fahrer die Verkehrszeichen verstehen und diese Regeln befolgen. Andernfalls können Unfälle und Todesfälle zunehmen.Gleiches gilt natürlich auch für autonome Fahrzeuge. Um für den Verkehr geeignet zu sein, müssen sie über Funktionen verfügen, die Verkehrszeichen erkennen können.

Convulational Neuronale Netze, der Unterzweig des Deep Learning, werden zur Analyse der visuellen Informationen verwendet. Ich habe auch ein Trainingsmodell mit CNN erstellt, um dieses Problem zu lösen. Es gibt ein Modell, das Verkehrszeichen zuverlässig und genau klassifiziert und lernt, die am besten geeigneten Merkmale für dieses Problem zu identifizieren.

2 Verwandte Arbeit

GonzalezReyna [1] schlug eine Methode vor, die auf einer Richtungsgradientenkarte und einer Karhunen-Loeve-Transformation basiert und deren Klassifizierungsgenauigkeit unter Verwendung des GTSDB-Datensatzes (German Traffic Sign Detection Benchmark) 95,9% beträgt. In den letzten Jahren haben einige Wissenschaftler viele auf maschinellem Lernen basierende Methoden zur Erkennung von Verkehrszeichen vorgeschlagen [2], wie beispielsweise das Aggregate Channel Feature (ACF) und den Integrated Channel Feature Fusion Detector [3]. In [4] wird ein Verkehrszeichenerkennungsverfahren vorgeschlagen, das die evolutionäre Adaboost-Erkennung und die Wald-ECOC-Klassifizierung verwendet.

Während dieser Studie habe ich Vergleiche mit zwei verschiedenen Modellen angestellt. Eines [5] davon war das Netzwerk, das mit einem Lenet und das andere [6] mit einem klassischen CNN-Modell erstellt wurde. Unsere Genauigkeitswerte liegen sehr nahe beieinander, da die von uns verwendeten Parameter und Werte Ähnlichkeiten aufwiesen.

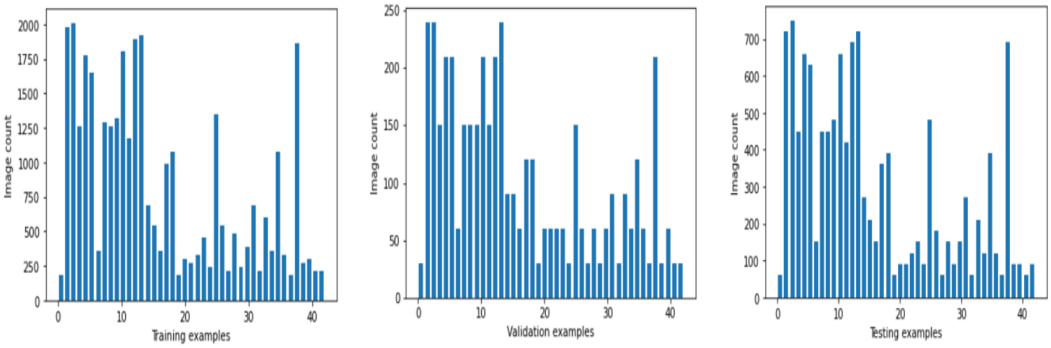
3 Datensatz

Der Datensatz [7], den wir zum Trainieren unseres eigenen benutzerdefinierten Verkehrszeichenklassifikators verwenden, ist "German Road Sign Recognition Benchmark (GTSRB)". Der GTSRB-Datensatz besteht aus 43 Verkehrszeichenklassen und ungefähr 50.000 Bildern. Die Bildgrößen reichen von 15 x 15 bis 250 x 250 Pixel. Bilder sind nicht unbedingt quadratisch und müssen nicht zentriert werden. [8]

Wenn ich detailliertere Informationen gebe, enthält dieser Datensatz 34799 Trainings-, 12630 Test- und 4410 Validationsbeispiele. Alle Bildgrößen sind auf 32 * 32 * 3 eingestellt, um das Schreiben von Code zu vereinfachen. Der Aufnahmewinkel, die Qualität und die Farbtöne aller Bilder unterscheiden sich voneinander. Alle von ihnen werden in ein ähnliches Format gebracht, damit es beim Training unseres Modells keine Schwierigkeiten gibt.

Hier sehen Sie einige Beispiele der Daten.

Von diesem Diagrammen sehen Sie die ungefähren Anzahl der Bildern von 43 Klassen in allen Datendateien.



4 Methodische Vorgehensweise

Dieses Projekt wurde mit der Programmiersprache Python erstellt. Wir werden einige Methoden anwenden, um bessere Ergebnisse zu erzielen, bevor wir das Modell trainieren.

Die zu verwendenden Bibliotheken müssen vor der Implementierung des Codes geladen werden (NumPy, Pandas, Keras, Matplotlib und OpenCV). Hier verwenden wir "numpy" für numerische Berechnungen, "Pandas" zum Importieren und Verwalten des Datensatzes, "cv2" für einige Vorverarbeitungsschritte für mehr Effizienz und natürlich "Keras" zum schnellen Aufbau des Convolutional Neural Network mit weniger Code. [9]

4.1 Vorverarbeitung von Images

Vor dem Training des Modells hilft uns die Vorverarbeitung dabei, genauere Ergebnisse zu erzielen. Die Bilder werden zuerst ausgegraut, um die Berechnung zu reduzieren. Auf diese Weise verschwinden RGB-Channels, dh die Farbkanäle von Bildern. Dies erleichtert unsere Arbeit beim Erstellen eines Modells. Wir haben OpenCV für diese Aufgabe verwendet, mit cvtColor().

“EqualizeHist()” verbessert den Kontrast des Bildes, indem es mit benachbarten Pixeln normalisiert wird, wodurch die Dichte der Pixel kompensiert wird.

4.2 Data Augmentation

Wir wenden den Schritt ImageDataGenarator aus der Keras-Bibliothek an. Dieser Schritt ermöglicht es, die Bilddatenzahlen mit bestimmten Änderungen zu erhöhen, wie Breitenverschiebung**,** Höhenverschiebung**,** Zoom**,** Scherung**,** Drehung. Auf diese Weise verbessern wir die Leistung des Modells, erweitern unseren Datensatz und reduzieren gleichzeitig die Overfitting.

4.3 Sequentiell Modell

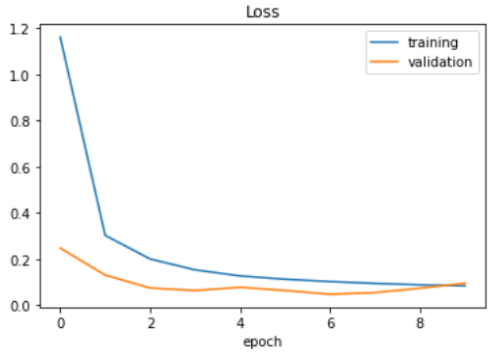
Wie ich bereits erwähnt, wird das CNN-Modell zur Unterscheidung und Klassifizierung von Bildern verwendet. Beim Erstellen des Modells habe ich das Sequentiell-Modell von Keras verwendet. Es ist eines der gebräuchlichsten und am einfachsten zu erlernenden Modelle. Dieses Modell besteht aus aufeinanderfolgenden Schichten. Wir haben 4 Conv2D, 3 MaxPooling2D und 2 Fully Connected Layers.

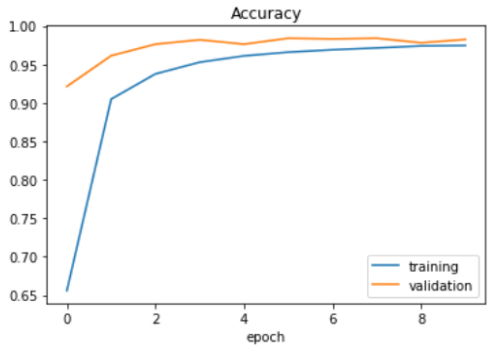
Conv2D-Schichten werden verwendet, um die Eigenschaften der Eingaben zu bestimmen. Polling-Schicht wird der nach der Conv2D-Schichten angewendet. Diese Ebenen reduzieren die Anzahl der Gewichte und steuern die Kompatibilität. Flatten Layers bereitet die Eigenschaft für das klassische neuronale Netzwerk vor, dh die Daten werden in Form einer Matrix abgeflacht. Fully Connected oder Dense Schichten folgen Conv2D und Pool-Layers. Es nimmt die Daten aus dem “Flatten” prozess und führt den Lernprozess über das neuronale Netzwerk durch. Es gibt auch ReLU und Softmax als Aktivierungsfunktionen verwendet.

Wir müssen das von uns erstellte Modell kompilieren, um die Ergebnisse zu sehen. Auf diese Weise können wir den Performanz des Modells messen. An diesem Punkt müssen wir die Kostfunktion, die Verbesserungsmethode und die Performanzmetriken festlegen, die befolgt werden sollen. Der “CategoricalCrossEntropy” ist eine sehr bevorzugte Loss-Funktion für Multiklasse Klassifizierungsmethoden. Es hilft uns auch, Verlust und Genauigkeitswerte schrittweise zu messen. Wir verwenden Adam Optimizer in unseren Schritten, um die Kosten zu verbessern, dh zu senken. Dies ist ein Optimierungsalgorithmus, der anstelle des Gradient Descent verwendet werden kann. Als Hyperparameter wähle ich die Schrittgröße als 0,001. Bei der Auswahl der Batch-Size habe ich 64 entschieden, wie viele Daten ich während der Trainingsphase gleichzeitig verarbeiten möchte. Ich würde meine Trainingsschrittanzahl (Epoch) auf 10 setzen. Somit werden in jedem Schritt 2000 Daten in das Modell aufgenommen und der Erfolg wird gesteigert.

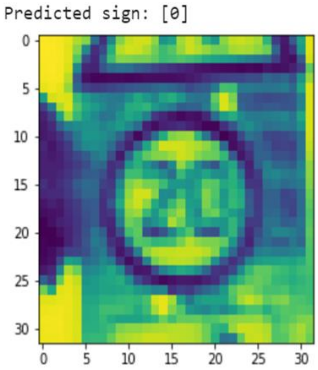
5 Ergebnisse

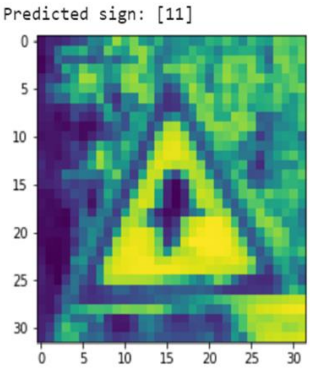
Während des Trainings dieses Modells habe ich verschiedene CNN-Modellbeispiele verwendet und mein eigenes Modell erstellt. Ich habe ein gutes Genauigkeitsergebnis erhalten. Wenn wir uns die Ergebnisse ansehen, sehen wir, dass unser Loss-Wert am Ende aller Epochen weniger als 2 Prozent und der Genauigkeitswert mehr als 95 Prozent beträgt. Während die Daten im Modell vorgehen, können wir beobachten, dass der Loss-Wert schrittweise abnimmt und der Genauigkeitswert also "Accuracy" zunimmt.





Wir messen auch Verlust- und Genauigkeitsmetriken für Testdaten und wir bekommen ungefähr die Ergebnisse, für Test Loss 0.18 und Test Accuracy 0.96.
Wenn wir ein paar Bilder aus der Testbeispiele ausprobieren, können wir hier auch die richtige Klassifizierung sehen. Z.B Das Geschwindigkeitlimit von 20 gehörte zur Klasse 0 usw.





6 Fazit / Zukünftige Arbeit

Die Ergebnisse dieses Projekts zeigen, dass mit Deep Learning Methoden die Studien mit hoher Genauigkeit und Leistung erstellt werden können. In den späteren Phasen des Projekts, es kann mehr als ein Modell entwickelt werden. Operationen können auch mit erweiterten Bibliotheken und verschiedenen Methoden durchgeführt werden, um schnellere und genauere Ergebnisse zu erhalten. Verteilungen im Datensatz können ausgeglichen werden. Es kann mit einem größeren Datensatz gearbeitet und das Projekt kann vielleicht implementiert wird.

References

[1] Gonzalez-Reyna S E, Avina-Cervantes J G, Ledesma-Orozco S E, et al. Eigen-gradients for traffic sign recognition[J]. Mathematical Problems in Engineering, 2013, 364305: 1-6. https://doi.org/10.1155/2013/364305

[2] Stallkamp J, Schlipsing M, Salmen J, et al. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition[J]. Neural networks, 2012, 32: 323-332. https://doi.org/10.1016/j.neunet.2012.02.016

[3] Møgelmose A, Liu D, Trivedi M M. Detection of US traffic signs[J]. IEEE Transactions on Intelligent Transportation Systems, 2015, 16(6): 3116-3125. https://ieeexplore.ieee.org/document/7116530

[4] Baró X, Escalera S, Vitrià J, et al. Traffic sign recognition using evolutionary adaboost detection and forest-ECOC classification[J]. IEEE Transactions on Intelligent Transportation Systems, 2009, 10(1): 113-126. https://doi.org/10.1109/TITS.2008.2011702

[5] https://github.com/singhbir/Traffic_Sign_Classifier-/blob/master/Traffic_Sign_Classifier.ipynb

[6] Python Image Classification using Keras https://www.geeksforgeeks.org/python-image-classification-using-keras/?ref=rp

[7] Jad Slim, (2018, July 10). German Traffic Signs Project https://bitbucket.org/jadslim/german-traffic-signs/src/master/

[8] Institut für Neuroinformatik, (2019, May 10). GTSRB http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset

[9] Python Image Classification using Keras https://www.geeksforgeeks.org/python-image-classification-using-keras/?ref=rp

GitHub Link

Repo: https://github.com/aybukeb/Deep-Learning
Code: https://github.com/aybukeb/Deep-Learning/blob/master/inf003projekt.ipynb