DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg
Campus Friedrichshafen

Confusion Matrix und Excel-Export zur Evaluationserweiterung

Codereport Deep Learning

Studiengang Elektrotechnik

Studienrichtung Fahrzeugelektronik

Duale Hochschule Baden-Württemberg Ravensburg, Campus Friedrichshafen

von

Max Brückner

Abgabedatum:	12.06.2022
Matrikelnummer:	6690523
Kurs:	TFE19-2
Gutachter der Dualen Hochschule:	Mark Schutera

1 Einleitung

Der Automobilsektor wächst weltweit ständig weiter und bringt fortlaufend neue Technologien auf den Markt. Gerade im Fachbereich des Autonomen Fahrens gibt es ständig neue Innovationen. Diese reichen von immer besser werdenden Fahrerassistenzsystemen bis hin zur Vollautomatisierung von Shuttlebussen. Bereits jetzt können Neuronale Netzwerke Straßenschilder erkennen und bewerten. Doch wie gut können diese Netze das wirklich?
Wir haben ein Neuronales Netz gegeben und sollen hierfür den Validierungsprozess ausbauen. Unser neuronales Netz klassifiziert verschiedene Verkehrsschilder und gibt hierfür eine „Accuracy“ aus. Gerade im Straßenverkehr ist es jedoch nicht ausreichend sich nur anzusehen, was richtig klassifiziert, sondern auch explizit was falsch klassifiziert wurde. Hierfür werden alle gegebenen Geschwindigkeitsschilder gegenüber gestellt und gegeneinander bewertet. Im Zuge dessen wurde die Validierung des Netzes ausgebaut und um verschiedene Metriken erweitert. Die Auswertung wurde um Confusion-Matrices, Precision-, Recall- und F1-Score erweitert. Ebenfalls wird die Evaluation, um sie besser präsentabel zu machen, in eine Excel-Mappe exportiert und anschaulich aufbereitet. Es folgt die Dokumentation der Umsetzung der oben genannten Metriken.

2 Codedokumentation

2.1 Skript translation.py

Das Skript translation.py beinhaltet Funktionen, welche zwischen den „Sprachen“ des Netzes und der Validierung übersetzen.
Das Skript validation_pipeline.py liest zu Beginn die Ordnernamen, welche zu bestimmten Klassen gehören, der „safetyBatches“ als Variable vom Typ <string> ein. Diese werden dann nach ihrer Einlesereihenfolge mit null beginnend indiziert. Dadurch weichen die Namen der Klassen in der validation_pipeline.py von denen ab, welche unser Neuronales Netz kennt. Die Konsolenausgabe der validation_pipeline.py wurde dadurch verwirrend und aufwendig lesbar.
Die Funktion translate_gt_and_predicts() löst dieses Problem und ist in der Datei translation.py zu finden. Diese Funktion konvertiert den Ordnernamen vom Typ <string> zum Typ <integer> und gibt ein Array zurück, welches die richtigen Klassennamen beinhaltet. So kann nun auch in der validation_pipeline.py korrekt auf die eingegebenen Klassen geschlossen werden. Die Funktion translate_classes() übersetzt die Klassennamen nun über ein Dictionary in Klarnamen und gibt diese als Anhaltspunkt in der Konsole aus. Das Dictionary tranlsdict beinhaltet die Klarnamen der Schilder zu den Zugehörigen Klassennamen. Die Klassennamen fungieren hierbei als <key> des Dictionarys.

2.2 Skript evaluation_expansion.py

Das Skript evaluation_expansion.py beinhaltet die Funktionen, welche die Evaluation des Netzes verbessern. Hierbei handelt es sich um verschiedene Funktionen zur Erstellung von Metriken und um eine Funktion, die nach Excel exportiert.

2.2.1 Erstellen der „Confusion-Matrices“

Die Klasse ConfusionMatrix definiert den Datentyp, in welchem die Daten für den Export nach Excel im Laufe dieses Skripts gespeichert werden. Sie beinhaltet einen selbst definierten Konstruktor. In diesem wird festgelegt, dass zur Erstellung eines Objekts eine Kombination aus zwei Klassen notwendig ist. Diese Kombination verleiht dem Objekt seine Einzigartigkeit und setzt sich aus den beiden Klassennamen zusammen, welche in der Confusion-Matrix betrachtet werden. Die Funktion create_cms() erstellt die Confusion Matrices. Als Eingangsparameter benötigt sie zwei Arrays. Diese Arrays beinhalten die „Ground Truth“ und die „Prediction“ unserer Validierung. In diesem Fall werden die übersetzen Arrays translated_gt und translated_predicts welche von der Funktion translate_gt_and_predicts() zurückgegeben werden, übergeben. Der erste Arbeitsschritt der Funktion besteht daraus, alle Klassennamen aus der „Ground Truth“, frei von Duplikaten, in ein Array zu kopieren. Auf Basis dieses Arrays können nun alle möglichen Paar-Kombinationen der Geschwindigkeitsschilder gebildet werden. Hierbei wird danach unterschieden, ob ein Klassenname an der ersten oder zweiten Stelle der Kombination steht, das heißt die Kombination (0,1) entspricht nicht der Kombination (1,0). Ein kleines Beispiel macht dies anschaulich. In der Praxis ist es in der Regel weniger schlimm ein 70er Schild als 50er Schild ((4,2)) zu erkennen, als umgekehrt ((2,4)). Während beim ersten Fall ein Autonomes Fahrzeug mit verringerter Geschwindigkeit, und damit konform mit der StVO, auf der Landstraße fährt, würde es im zweiten Fall mit 70 km/h durch eine Wohnsiedlung fahren. Aus diesem Grund werden alle Kombinationen evaluiert.
Im zweiten Funktionsschritt wird eine gemeinsame „Ground Truth“ und eine gemeinsame „Prediction“ der beiden Klassen gebildet. Diese werden zur Erstellung der spezifischen Confusion-Matrix benötigt.

Der dritte Arbeitsschritt gleicht nun die „Ground Truth“ und die „Prediction“ ab. Nun werden die für die Confusion-Matrix benötigten Parameter „TP, TN, FP, FN“ ermittelt. Ebenfalls wurde ein neuer Parameter eingeführt. Der Parameter „Falsely Associated (f_a)“ zählt, wie oft das Netz ein Schild klassifiziert hat, welches zu keiner der beiden Klassen gehört. Das Codebeispiel 2.1 zeigt die Zuordnung der Parameter.

Listing 2.1: Einfache Darstellung der Auswertung

```
1 gt = [0, 0, 0, 1, 1, 1]
2 pred = [0, 0, 1, 0, 1, 3]
3 >>> TP = 2, FN = 1, FP = 1, TN = 1, f_a = 1
```

Zum Schluss erfolgt eine Konsolenausgabe, welche über die abgeschlossene Erstellung der Confusion-Matrices informiert. Die Funktion gibt eine Liste zurück, welche die Objekte mit den Daten für jede Kombination beinhaltet. [Ana21]

2.2.2 Berechnung verschiedener Bewertungskriterien

Der Precision-Score gibt das Verhältnis zwischen dem TP- und allen wirklichen positiven Werten an. Er bewertet also wie viele Schilder, welche der ersten Klasse der Kombination zugeordnet wurden, auch wirklich zu dieser Klasse gehören. Ein hoher Precision-Score deutet auf die Zuverlässigkeit einer positiven Bewertung hin. [Sol16]
Der Precision-Score berechnet sich über die Formel 2.1.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.1}$$

Die Funktion calc_precision() berechnet den Precision-Score eines jeden CM-Objekts mit der Formel 2.1 und speichert ihn im Objekt.

Der Recall-Score gibt das Verhältnis zwischen dem TP- und allen positiv bewerteten Werten an. Er bewertet also wie viele Schilder, welche zur ersten Klasse der Kombination gehören, auch wirklich als diese eingestuft wurden. Ein hoher Recall-Score deutet auf die Zuverlässigkeit richtigen Erkennung hin. [Sol16]

Der Recall-Score berechnet sich über die Formel 2.2.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.2}$$

Die Funktion calc_recall() berechnet den Recall-Score eines jeden CM-Objekts mit der Formel 2.2 und speichert ihn im Objekt.

Der F1-Score gibt den gewichteten Durchschnitt zwischen Precision- und Recall-Score an. Hierbei werden die Werte FP und FN gleichzeitig in Betracht gezogen. Er bietet sich vor allem bei einer Eingabe von Klassen mit ungleicher Anzahl von Testdaten, zur Bewertung des Netzes, an. [Sol16]
Der F1-Score berechnet sich über die Formel 2.3.

$$\text{F1} = \frac{2 \cdot \text{Recall} \cdot \text{Precision}}{\text{Recall} + \text{Precision}} \tag{2.3}$$

Die Funktion calc_f1() berechnet den F1-Score eines jeden CM-Objekts mit der Formel 2.3 und speichert ihn im Objekt.

2.2.3 Export nach Excel

Die Funktion export_data_excel() führt den Export der Evaluationsdaten in eine Excelmappe durch und bedient sich hierfür bei der xlsxwriter Library. Diese ermöglicht es Exceldateien aus Python heraus zu erstellen. Im ersten Schritt der Funktion wird das Excel-Objekt erstellt, mit welchem im Laufe der Funktion gearbeitet wird. Ebenfalls wird das Array, welches die CM-Objekte enthält den Kombinationen nach sortiert. Dies wird für die Darstellung benötigt. Jede Klasse soll am Ende ihr eigenes Tabellenblatt haben. Im zweiten Schritt der Funktion werden verschiedene Formate definiert, welche beim Schreiben in die Tabelle angewandt werden können. Die äußerste Ebene der Schleife erstellt bei jedem Durchlauf ein neues Tabellenblatt für die aktuelle Klasse. Ebenfalls wird eine Legende auf jedes Blatt eingefügt, um die Grafiken, welche im nächsten Schritt geschrieben werden, zu erklären. Die Schleife auf der tieferen Ebene schreibt nun für jede Kombination, an welcher eine bestimmte Klasse an der ersten Stelle steht, die Daten in das zugehörige Tabellenblatt. Hierdurch kann die Performance des Netzes bei einem

bestimmten Schild gegenüber den anderen Schildern auf einem Blatt gesammelt dargestellt werden. Hierdurch ergeben sich so viele Tabellenblätter, wie verschiedene Geschwindigkeitsklassen in den Testdaten vorhanden sind. Am Ende wird die Arbeitsmappe geschlossen und gespeichert. Sie befindet sich im selben Ordner wie die Skripte.

2.2.3.1 Die Variable „CriticalValue“

Die Variable CriticalValue taucht nur in der Excelmappe auf. Sie greift den Gedanken aus der Einleitung auf, welcher sagt, dass eine falsche Erkennung von Schildern gewichtet werden muss. Der Parameter bildet sich aus der Formel 2.4

$$\text{CriticalValue} = \frac{\text{Predicted velocity value}}{\text{True velocity value}} \tag{2.4}$$

Die Variable wird nun grün hinterlegt, falls der Wert im Intervall [0,75;1] liegt. Dieses Intervall wurde durch eine Überlegung gewählt. Es liegt allerdings keine wissenschaftliche Analyse dahinter. Es wird davon ausgegangen, dass der Fehler in diesem Intervall nicht fatal ausfällt und akzeptabel ist. Außerhalb dieses Intervalls wird der Wert rot hinterlegt und damit als nicht akzeptabel bewertet.

2.3 Die „testbatch“

Die „testbatch“ wurde aus Trainingsdaten zusammengestellt. Da die Aufgabe darin bestand, die Evaluation und nicht die Performance des Netzes zu verbessern, besteht kein Problem darin die Daten wiederzuverwenden. Sie enthält alle Klassen, welche Geschwindigkeitsschilder repräsentieren.

Literatur

[Ana21] Analytics Vidhya. *Confusion Matrix for Multi-Class Classification - Analytics Vidhya*. 2021. URL: https://www.analyticsvidhya.com/blog/2021/06/confusion-matrix-for-multi-class-classification/ (besucht am 09. 06. 2022).

[Sol16] Exsilio Solutions. "Accuracy, Precision, Recall & F1 Score: Interpretation of Performance Measures". In: *Exsilio Consulting* (9.09.2016). URL: https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/ (besucht am 09. 06. 2022).