DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg
Campus Friedrichshafen

Ausgabe von Einzelgenauigkeigkeiten in einem Mehrklassensystem für die Validierung von neuronalen Netzen

Hausarbeit

Studiengang Elektrotechnik

Studienrichtung Fahrzeugelektronik

Duale Hochschule Baden-Württemberg Ravensburg, Campus Friedrichshafen

von

Luis Palle

Abgabedatum:	13. Juni 2022
Bearbeitungszeitraum:	11.04.2022 - 13.06.2022
Matrikelnummer:	7873039
Kurs:	TFE19-2
Gutachter der Dualen Hochschule:	Marc Schutera

1 Umsetzung

1.1 Organisatorisches

Zu Beginn der Vorlesung wurde durch den Dozenten Marc Schutera die Rahmenbedingungen für diese Hausarbeit gesetzt. Dabei sollte man sich unterschiedliche Möglichkeiten überlegen, wie die Validierung der Netze verbessert werden kann. Grundlegend beschäftigt sich das hier verwendete Neuronale Netz mit der Erkennung von Verkehrsschildern. Dabei sortiert das Netz aus den vorgegebenen Bildern die Verkehrsschilder in Klassen ein. Im Anschluss werden durch die Validierung die Ergebnisse ausgegeben

Listing 1.1: Ausgabe der Kommandozeile

```
1 Found 17 files belonging to 5 classes.
2 Classes available: ['0' '1' '10' '11' '12']
3 Predictions: [ 0  1  1 12  1  2  2 18  3 10  3  3  4  4  4  4  4]
4 Ground truth: [0 1 1 1 1 2 2 3 3 3 3 3 4 4 4 4 4]
5 Accuracy: 0.8235294
```

Hierbei sieht man im Codebeispiel 1.1 dass als erstes eine Übersicht ausgegeben wird, wie viele Daten gefunden, die Klassen zugeordnet wurden. Im nächsten Schritt erkennt man die gefundenen Klassen. Dabei steht jede Klasse für ein Straßenschild. In den nächsten beiden Zeilen werden die vom Netz erkannten Klassen angegeben und unter *Ground truth* die tatsächlichen Klassen angegeben. Diese Ergebnisse werden unter einer gesamten *Accurancy* ausgegeben, die die Genauigkeit des Netzes angibt. Hier setzt meine Idee an. Während mein Nebensitzer die Genauigkeit von Stopschildern auswertet wurde mir vom Dozent geraten die einzelnen Genauigkeiten der verschiedenen Klassen auszugeben.

1.2 Umsetzung

Zu Beginn möchte ich auf Den Programmcode in der Hauptdatei hinweisen. Dieser wird benötigt um die ausgelagerte Funktion aufzurufen.

Listing 1.2: Aufruf der Funktion

```
#Please write here your Classes you want to know your accurancy
testingclass=3
validation.onceaccurancy(test_labels, predictions, testingclass)
```

Wie man im Codebeispiel 1.2 erkennen kann, wird über die variable *testingclass* die zu testende Klasse angegeben. Die eigentliche Funktion befindet sich in einer extra Datei. Als Übergabeparameter werden aus 1.1 die *Predictions*, *Ground truth* und die zu untersuchende Klasse *testingclass* angegeben.

Listing 1.3: Programmcode als ausgelagerte Python Funktion

```
def onceaccurancy(test_labels, predictions, testingclass):
  correct =0
  incorrect=0
  print("====================================")
  print("Es wird die Genauigkeit der Klasse", testingclass, "ausgegeben.
  ")
  for i in range(len(test_labels)):
    if test_labels[i] == testingclass:
      if predictions[i] == test_labels[i]:
        correct+=1
      else:
        incorrect+=1


  print("Korrekt in Klasse",testingclass, "sind", correct)
  print("Inkorrekt in der Klasse", "sind",incorrect)
  count_prooved=correct+incorrect
  print("Das entspricht einer Genaugigkeit der Klasse", testingclass, "
  von", 100*(correct/count_prooved),"%")
```

Nun befindet man sich in der ausgelagerten Funktion die über *def onceaccurancy* gekenn-

zeichnet ist. Im nächsten Schritt wird zur Information des Nutzers die zu untersuchende Klasse angegeben. Nun beginnt die eigentliche Arbeit des Programms. Da sich die Ergebnisse in Arrays befinden werden diese stellenweise untersucht. Es Beginnt bei Stelle 0, dort wird überprüft ob an dieser Stelle im *Ground Truth* überhaupt die gesuchte Klasse zu finden ist, ist das nicht der Fall, wird direkt zum nächsten Zeichen gesprungen. Falls die gesuchte Klasse mit dem aktuellen übereinstimmt, so wird in der If-Schleife im Codebeispiel A.1 in Zeile 8 die Korrektheit überprüft. Falls die beiden Arrays an der gleichen Stelle das gleiche Zeichen aufweisen, so wird die *correct*-Variable um eins erhöht, falls die Zeichen nicht die gleichen sind, die *incorrect*-Variable um eins. Dies geschieht so lange bis das Array durchgearbeitet ist. Im nächsten Schritt werden die korrekten und inkorrekten Ergebnisse ausgegeben und daraus die Einzelgenauigkeit errechnet.

1.3 Vorteile des Programms

Da das Programm auf der Auswertung von Arrays basiert, können diese nahezu beliebig lang sein, solange die Ergebnisse nicht den Wert einer Int-Variable überschreiten. In dem hier dargestellten Code handelt es sich nur um den Batch 0 - wenn man die anderen Batches dazu ausführt bekommt man deutlich mehr Klassen und Bilder, genau dann zeigt das Tool seine Vorteile. Es verschafft einen schnellen Überblick über eine Klasse, die man genauer untersuchen will.

1.4 Schwierigkeiten beim Programmieren

Beginnende Schwierigkeiten waren meine COVID-19 Infektion, weswegen ich leider zwei Vorlesungen verpasst habe. Da ich wie bereits oben angesprochen allerdings sehr früh mein Thema abgesprochen hatte, war das für mich kein Problem. Die mit Abstand größte Schwierigkeit war, das mein Computer leider 5 Tage vor der Abgabe seinen Dienst quittiert hat und ich leider die Ergebnisse nur Lokal hatte und sie nicht ins Repo gepusht habe. Dennoch habe ich es geschafft das Programm schnell wieder neu zu Programmie-

ren. Dennoch hatte ich zu Beginn den Fehler, das bei der Überprüfung der Anwesenheit der gesuchten Klasse nicht im *Ground Truth* geschaut habe sondern in den *Predictions*. Dies hat zu falschen Ergebnissen geführt und es wurden zu viele als Incorrect klassifiziert

1.5 Ausblick und Erweiterungsmöglichkeiten

Eigentlich hatte ich in der ersten Version des Programms die aus 1.1 bekannten *Classes available* auszuwerten. Dies hätte eine komplette Übersicht über alle Klassen hinweg erzeugt. Bis mein Laptop abgestorben ist , hatte dieses System auch Funktioniert, doch in der kurzen Zeit die ich dann noch hatte, habe ich dieses Programm nicht mehr lauffähig bekommen. Ich habe es im Anhang angehängt. Als weitere Erweiterungsmöglichkeit gibt es die grafische Aufarbeitung der Daten durch ein Diagramm in dem die Genauigkeiten aufgelistet werden. Die grafische Auswertung wurde allerdings durch andere Kollegen bereits mehrfach durchgeführt und wäre somit auch keine Eigenleistung gewesen.

Anhang A

A.1 Nicht lauffähiger, fast fertiger Programmcode

Listing A.1: Klassendurchlauf

```
correct =0
incorrect=0
for o in range(len(class_names)):
  for i in range(len(test_labels)):
    if test_labels[i] == o:
      if predictions[i] == test_labels[i]:
        correct+=1
      else:
        incorrect+=1
print("Korrekt in Klasse",class_names[o], "sind", correct)
print("Inkorrekt in der Klasse",class_names[o], "sind",incorrect)
count_prooved=correct+incorrect
print("Das entspricht einer Genaugigkeit der Klasse", class_names[o], "
  von", 100*(correct/count_prooved),"%")
correct=0
incorrect=0
count_prooved=0

```