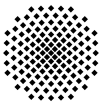

Universität Stuttgart

Institut für Angewandte Analysis und Numerische Simulation

Metrik für sEMG-Daten

metrik_NN_class_distribution.py

Lisa Dollmann, Maximilian Hack, Lilian Cathérine Lepère

Autor Lisa Dollmann
Maximilian Hack
Lilian Cathérine Lepère

Betreuer Aaron Krämer

Datum Stuttgart
29. Januar 2022

Das ist die Dokumentation der Metrik *metrik_NN_class_distribution.py*. Die Metrik misst wie häufig welche Klasse in einem Datensatz mit bestimmten Übergabeparametern vertreten ist. Die Metrik veranschaulicht, dass es peaks in einzelnen Klassen gibt, andere dafür garnicht auftreten.
In der Dokumentation werden Installation und Nutzung beschrieben, die enthaltenen Features und weitere Ideen aufgelistet, die Ergebnisse kurz interpretiert, relevante Programmausgaben dargestellt und erläutert.

Inhaltsverzeichnis

1 Installation und Nutzung

1.1 Module und Elemente

Das Programm benötigt die Module *numpy* und *matplotlib.pyplot*, sowie die anderen Module aus dem Projekt *argument_stuff*, *file_stuff* und *training_stuff/ training_stuff_2*.

1.2 Parameter

Die Eingabeparameter sind die selben wie für die Datei *main.py*. Auf deren Nutzung wird im folgenden Abschnitt teilweise eingegangen.

1.3 Nutzung

Beispielbilder zur Veranschaulichung der Nutzung sind in Kapitel 4 eingebunden.
Um die Metrik *metrik_NN_class_distribution.py* zu nutzen, muss man ihr dieselben Paramter übergeben wie *main.py*. Das sind AR, GT, CL, S, B, EP, TS und DT. Die Parameter AR, EP und TS sind für das Training des NN relevant, für diese Metrik nicht. DT übergibt man mit einem bereits exisierenden Datensatz, GT mit der Art der Daten Interpretation ($i, integer$ oder $p, probabilistic$), CL wie die Klassen definiert sind (z.B. $g, granular$ oder $d, direct$), S steht für die Anzahl an Abschnitten/ Sektionen in die das Geschwindigkeitsintervalll aufgeteilt werden soll, und B für die Anzahl an Aktivitätsstufen (0-100 %) auf einer Sektion. Eine Eingabe ins Terminal könnte also folgendermaßen aussehen:

$$python3\ metrik\_NN\_class\_distribution.py\ AR\ 2\ GT\ i\ CL\ a\ S\ 2\ B\ 2\ EP\ 5\ TS\ 3\ DT\ DATA/creator\_001/RawData\_7\_velocities\_\_\_\_13\_waveLengths.pickle$$

Hat man das Programm ausgeführt zeigt es ein Histogramm, das darstellt, wie häufig welche Klasse in dem Datennsatz DT mit den eingegebenen Parametern auftritt.

2 Enthaltene Features und deren Ausgaben

Da die Metrik im wesentlichen die Funktionen des Moduls $training\_stuff.py$ nutzt, soll nur auf eine Kleinigkeit eingegangen werden.

2.1 $training\_stuff$ vs $training\_stuff\_2$

Die Klassifizierung der einzelnen Bilder erledigt die Funktion $classify$. Deren output wird zunächst in der Variablen y_temp gespeichert und später nach $y\_validation$ kopiert. Je nachdem ob man $training\_stuff$ oder $training\_stuff\_2$ nutzt, werden 20% respektive 100% der Daten genutzt. Das kann man in Zeile 350 in $training\_stuff$ einstellen indem man den Bereich des arrays y_temp der kopiert wird vergrößert. Wichtig ist, dass man ebenfalls $x\_validation$ und $velIds\_validation$ vergrößert. Die relative Verteilung ändert sich dadurch kaum (vgl. 3, 4.1).

2.2 Das Histogramm

Das Histogramm bildet dann die Häufigtkeit ab, wie oft welche Klasse in dem Datensatz vorkommt. Da die genaue Häufigkeit auf dem Bild schwer abzulesen ist, wird die Häufigkeit der jeweiligen Klasse zusätzlich in Zahlenform in der Konsole ausgegeben.

3 Interpretation

Die Metrik veranschaulicht wie häufig welche Klasse in einem Datensatz mit den vom Benutzer festgelegten Parametern vorkommt. Beispile verdeutlichen, dass die Klassen völlig ungleichmäßig verteilt sind. Während einige Klassen sehr häufig vorkommen, kommen andere (fast) garnicht vor. Oft existiert ein peak in Klasse 0. Insgesamt zeigt die Metrik eine Schwäche des Klassifizierungsansatzes.

4 Relevante Programmausgaben

In diesem Abschnitt werden Programmausgaben veranschaulicht. Es werden von der *metrik_NN_class_distribution.py* erzeugt Bilder gezeigt.

4.1 Relative Häufigkeit

Man kann man festlegen wie viel Prozent der Daten man nutzen möchte. Die Histogramme sind relativ ähnlich (vgl. 2.1).

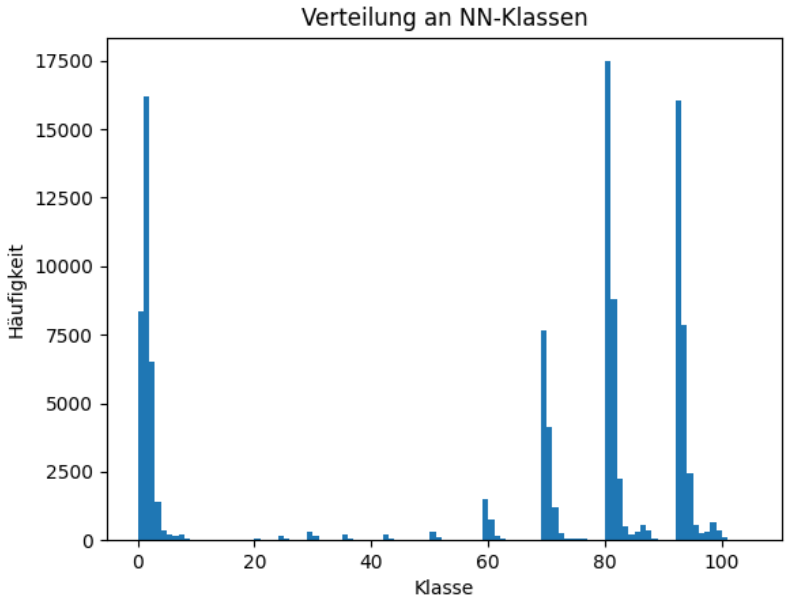

Abbildung 1: Ein Histogramm mit *training_stuff_2.py*, also 100% der Daten.

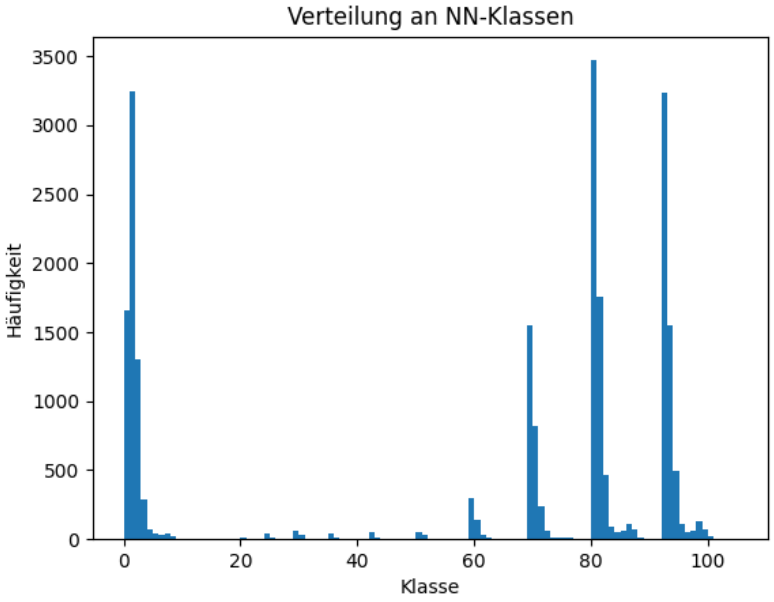

Abbildung 2: Hier wurden dieselben Parameter genutzt wie in 1, aber mit dem Modul *training_stuff.py*. Also wurden nur ca. 20% der Daten genutzt.

4.2 Beispielbilder für verschiedene CL

4.2.1 $DT\ 4\_velocities\_\_\_\_5\_waveLengths\_\_\_\_grow\_factor\_4$

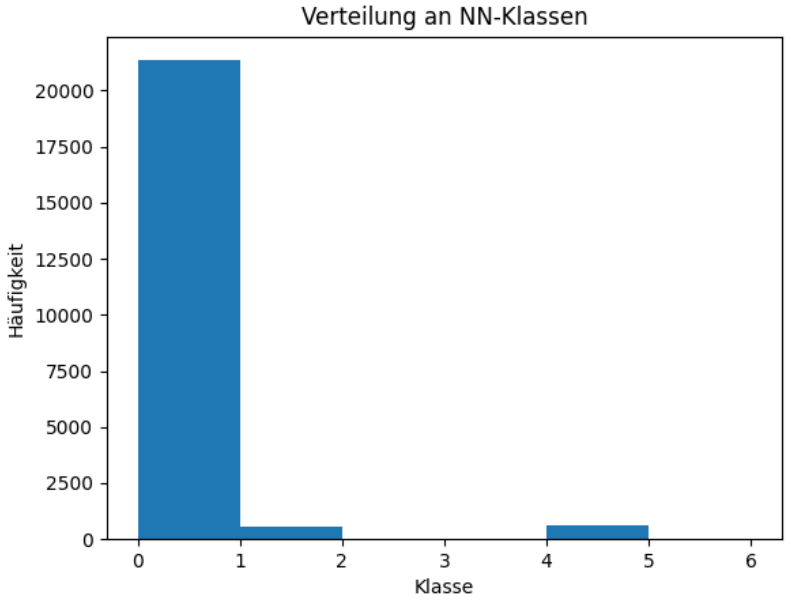

Abbildung 3: Ein Histogramm für $CL\ g$.

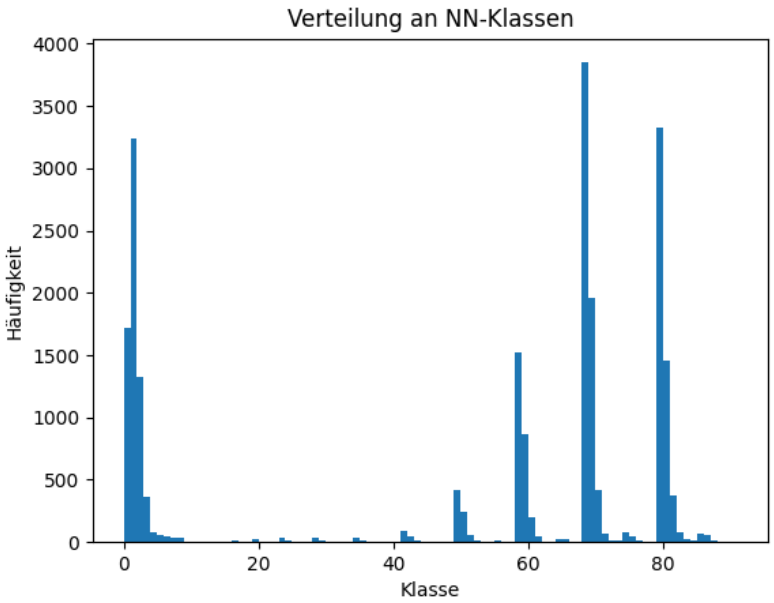

Abbildung 4: Ein Histogramm für die selben Parameter wie 3, außer $CL\ d$.

4.2.2 *DT 7_velocities____13_waveLengths*

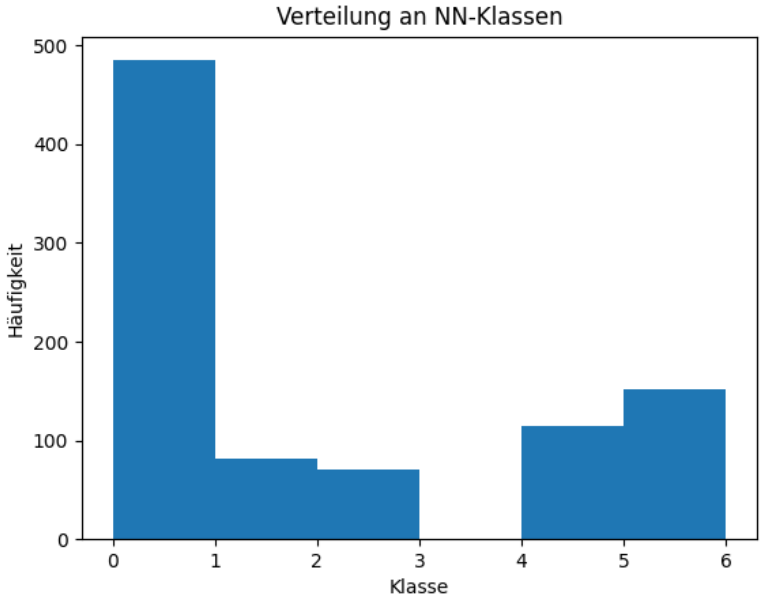

Abbildung 5: Ein Beispielhistogramm mit $CL\ m$.

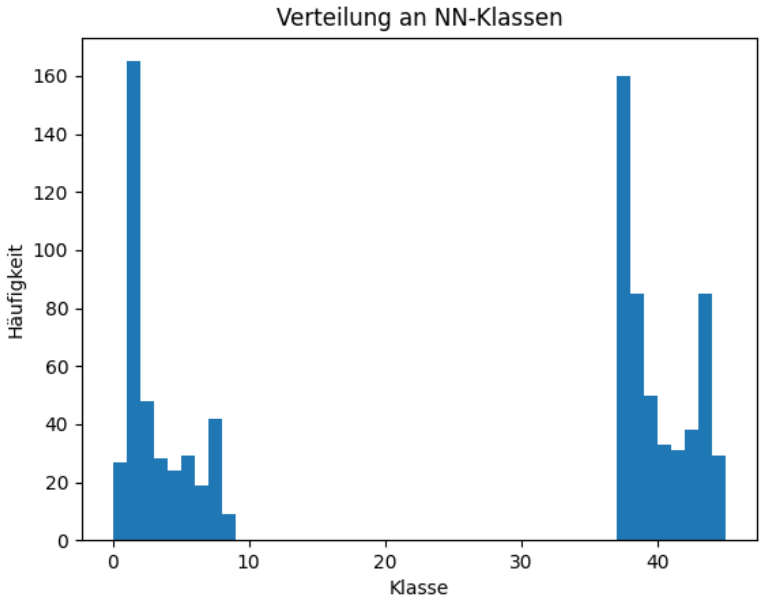

Abbildung 6: Ein Beispielhistogramm mit den selben Parametern wie 5, außer $CL\ a$.