**Verfahren der KI – Projektarbeit**

**Zählen von Mini-Rittersport-Tafeln**

**SS2023**

**Gruppe10**

**Fabio Pascuzzi, 86269**

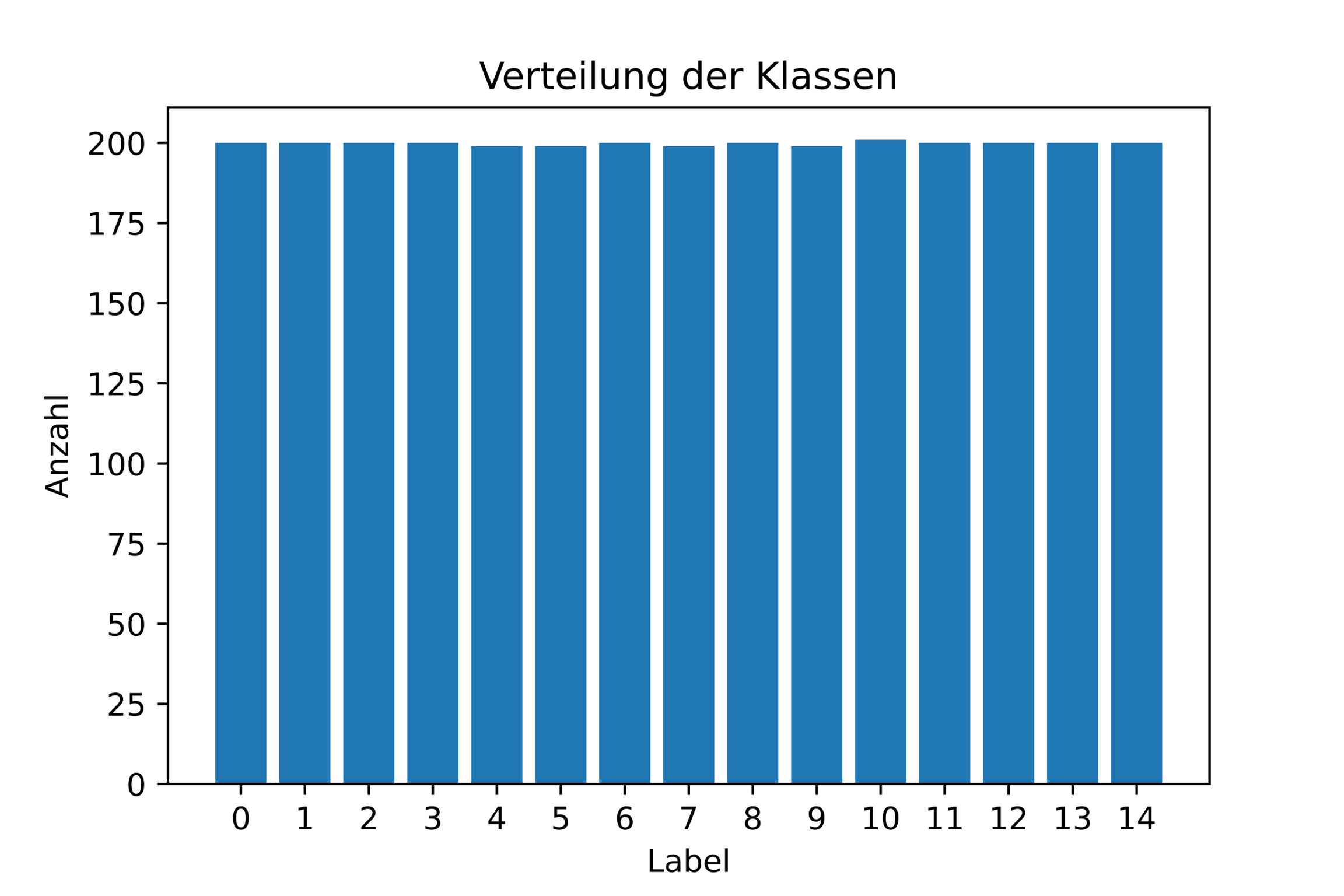
**Thomas Kutscher, 85468**

**Carl Zech, 85789**

**Jonathan Haller, 64674**

**Dateneigenschaften**

|  |  |
| --- | --- |
| **Klasse (Tafeln im Bild)** | **Anzahl** |
| 0 | 200 |
| 1 | 200 |
| 2 | 200 |
| 3 | 200 |
| 4 | 199 |
| 5 | 199 |
| 6 | 200 |
| 7 | 199 |
| 8 | 200 |
| 9 | 199 |
| 10 | 201 |
| 11 | 200 |
| 12 | 200 |
| 13 | 200 |
| 14 | 200 |
| **Gesamt** | **2997** |



**Verwendete Modelle**

* Entscheidungsbaum
  + Eigene Implementierung und von sklearn
* Neuronales Netz
  + Eigene Implementierung und von Tensorflow/Keras
* CNN von Tensorflow/Keras

**Aufteilung der Test-Trainings und Validierungsdaten**

|  |  |  |  |
| --- | --- | --- | --- |
| **Modell** | **Anteil Training** | **Anteil Test** | **Anteil Validierung** |
| Entscheidungsbaum (sklearn) | 70% | 30% | - |
| Eigener Entscheidungsbaum | 75% | 25% | - |
| Neuronales Netz (Tensorflow) | 68% | 20% | 12% |
| Eigenes Neuronales Netz | 68% | 20% | 12% |
| CNN (Tensorflow) | 50% | 20% | 30% |

**Erhobene Features**

* Anzahl der erkannten Linien
* Anzahl der Konturen
* Konturenfläche
* Anzahl der Konturen (mit Farbmasken)
* Konturenfläche (mit Farbmasken)

**Von Modellen genutzte Features**

Entscheidungsbaum (sklearn):

* Anzahl der Konturen (mit Farbmasken)
* Konturenfläche (mit Farbmasken)
* Anzahl der erkannten Linien

Eigener Entscheidungsbaum:

* Anzahl der Konturen (mit Farbmasken)
* Konturenfläche (mit Farbmasken)
* Anzahl der erkannten Linien

Neuronales Netz (Tensorflow):

* Anzahl der Konturen (mit Farbmasken)
* Konturenfläche (mit Farbmasken)
* Anzahl der erkannten Linien

Eigenes Neuronales Netz:

* Anzahl der Konturen (mit Farbmasken)
* Konturenfläche (mit Farbmasken)
* Anzahl der erkannten Linien

CNN:

* Pixelwerte der Bilder mit skalierten RGB-Werten und einer Auflösung von 196x196

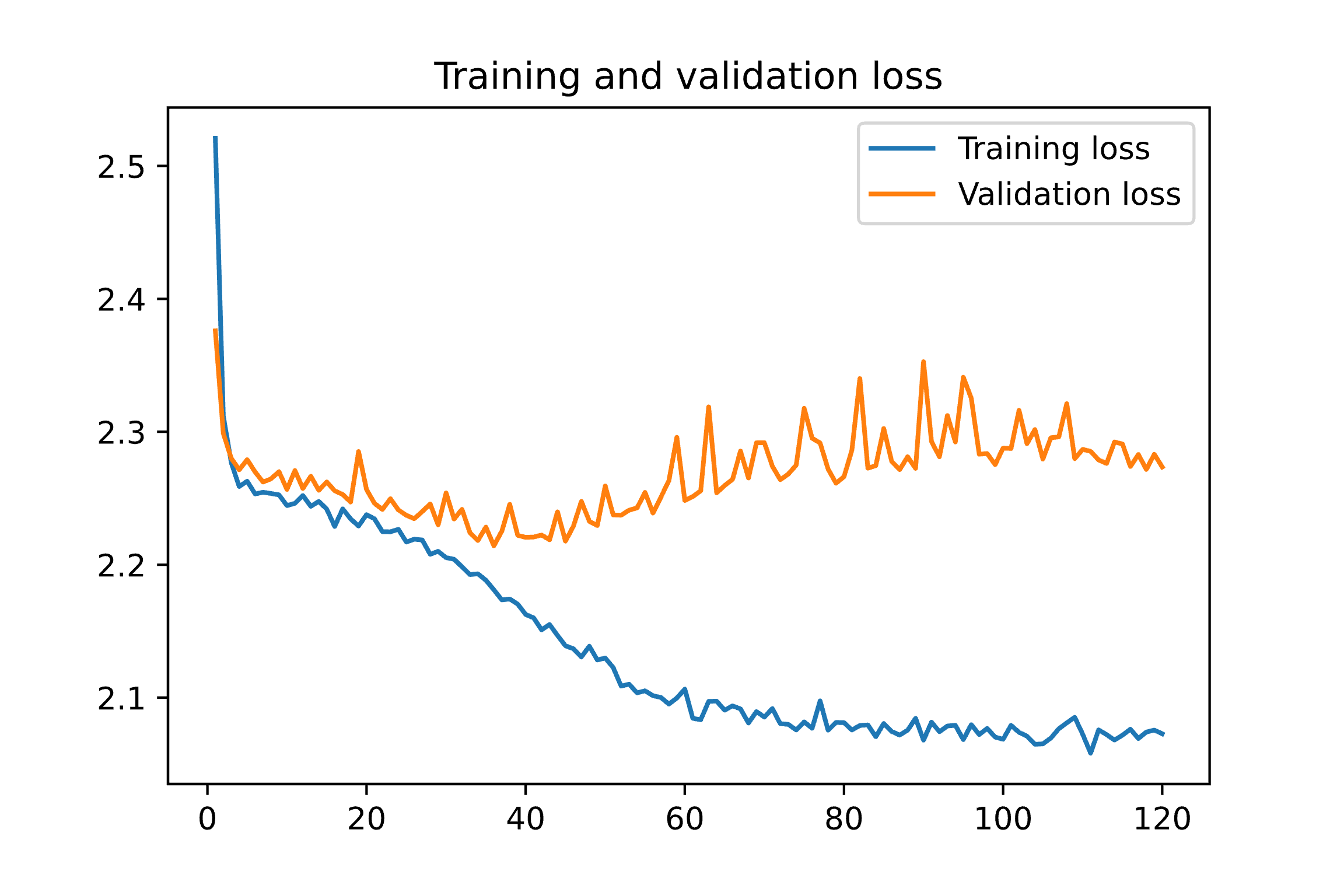
**Data-Augmentation**

Für das CNN wurden die Bilder durch eine Random-Flip-Schicht von Tensorflow zufällig horizontal und vertikal gespiegelt, um künstlich mehr Bilder zu erzeugen.

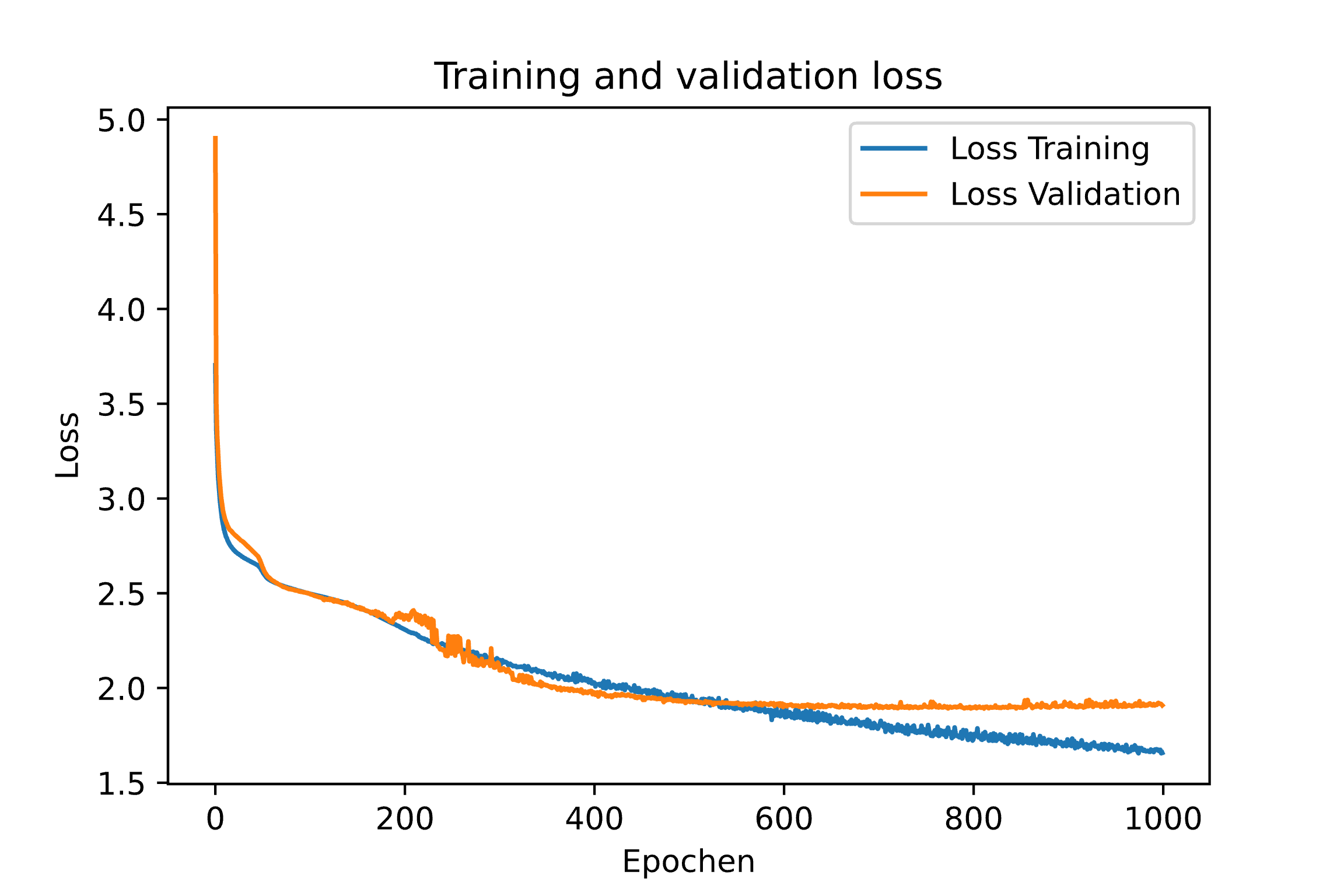
**Erzielte Metriken**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Modell** | **Accuracy** | **Precision** | **Recall** | **F1-Score** |
| Entscheidungsbaum (sklearn) | 71,2% | 72,8% | 71,4% | 71,6% |
| Eigener Entscheidungsbaum | 68,9% | 72,8% | 71,4% | 71,6% |
| Neuronales Netz (Tensorflow) | 32,7% | 33% | 35% | 28% |
| Eigenes Neuronales Netz | 48% | 48,9% | 48% | 47,6% |
| CNN (Tensorflow) | 80% | 81% | 80% | 80% |

**Loss des neuronalen Netzes mit Keras**



**Loss des eigenen neuronalen Netzes**

****

**Loss des CNN**

**Ein Bild, das Text, Screenshot, Diagramm, Schrift enthält.

Automatisch generierte Beschreibung**

**Scatter-Matrix der Features**

Ein Bild, das Screenshot, Quadrat, Reihe, Rechteck enthält.

Automatisch generierte Beschreibung

**Infos zur Erhebung der Features**

**Kantenerkennung**

Für die Kantenerkennung werden zuerst mit der Funktion "Canny" die Kanten auf dem Bild erkannt. Dabei entsteht ein schwarz-weißes Bild, auf dem nur noch die Kanten des Bildes zu sehen sind. Als nächstes werden in diesem Bild die Kanten, die eine Linie bilden mit der Funktion "HoughLinesP" verbunden. Die Anzahl der erkannten Linien wird dann als Feature genommen.

**Konturenerkennung**

Für die Konturenerkennung werden ebenfalls die Kanten erkannt. Danach wird das Bild mit den Funktionen "dilate" und "erode" bearbeitet, da dies bessere Ergebnisse liefert. Dann wird die Funktion "connectedComponentWithStats" verwendet, um die Konturen zu erkennen. Diese Konturen werden noch gefiltert in dem alle zu kleine Konturen entfernt werden, um nur die Konturen zu erfassen die tatsächlich eine Tafel sein können. Als Feature kann nun die Anzahl der Tafeln und die Fläche aller Konturen verwendet werden. Durch die Fläche soll erkannt werden, wenn mehrere Tafeln nah beieinander liegen und somit eine große Kontur bilden.

**Konturenerkennung mit Farbmasken**

Für die Konturenerkennung mit Farbmasken wird für jede Farbe der Tafeln eine eigene Maske definiert und auf das Bild angewandt. Dadurch erhält man jeweils ein Bild, auf dem nur die Pixel der definierten Farbe enthalten sind. Auf dieses Bild wird die Konturenerkennung angewandt. Der Vorteil ist, dass die Tafeln dadurch sehr gut separiert werden und man nicht so viele große Konturen erhält, die aus mehreren Tafeln bestehen.