Praktikum „Mustererkennung und Machine Learning“
Aufgabe 04 Version: SS2020

Training Neuronaler Netze mit Keras

Lernziele:

- Erstellung und Training von neuronalen Netzen (Feed-Forward, Faltungsnetze, Keras-Functional API, Residual-Networks)
- Bedeutung der Netz- und Trainingsparameter verstehen

Aufgabenstellung:

Durchzuführen sind die Versuche

a) 01_Multiclass_Klassifikator.ipynb

b) 02_FunctionalAPI_KindOfResNet

Die o.a. Jupyter-Notebooks sind in der Owncloud.

Die durchzuführenden Aufgaben sind in den Notebooks angegeben: siehe „!!! Aufgaben“ und „??? Fragen“

Durchführung:

Zur Trainingsdurchführung ist die Verwendung einer GPU wünschenswert aber nicht notwendig. Hierzu kann auch (sofern gerade verfügbar) die Infrastruktur der HAW-Informatik-Compute-Cloud (ICC) genutzt werden (s. unbedingt auch JupyterNotebooks.pdf). Vorgehensweise:

1. Gehen sie zu https:\\jupyter.icc.informatik.haw-hamburg.de .
2. Loggen Sie sich mit Ihrem HAW-Account ein.
3. Wählen Sie : StartMyServer → 1GPU (wenn Sie NN trainieren wollen) → Spawn.
4. Zum Laden der Jupyter-Notebooks „Upload“ wählen.
5. Notebook öffnen und loslegen …

Alternativ mit Anaconda auf dem eigenen Rechner (s. JupyterNotebooks.pdf).

Training Neuronaler Netze mit Keras

Teilversuch 1:

Am Beispiel des MNIST-Datensatzes sollen verschiedene NN (Feed-Forward-NN, Convolutional-NN) trainiert und optimiert werden.

Anm. : Die zu lösenden Aufgabenpunkte sind direkt im Notebook angegeben.

Teilversuch 2:

Mit der Functional-API von Keras soll das auf Seite 3 dargestellte NN trainiert werden. Es enthält Strukturelemente von Residual-Networks (shortcut-connections) und von Dense-Networks (Concateneate, 1x1-Faltung).

Als Vorlage für die Functional-API steht ein Beispielprogramm (03_FunctionalAPI_Beispiel.ipynb) zur Verfügung. Eine Kopie davon kann entsprechend der Aufgabenstellung abgewandelt werden.

Für die vektorielle Addition (shortcut-Connection) wird ein Keras-Add-Layer verwendet:

```
x = Add()([x1, x2])
```

Für die Konkatenation wird ein Keras-Concatenation-Layer verwendet:

```
x = concatenate([x1, x2])
```

Die fortlaufende Wiederholung einer Layergruppe (n/m-fach) kann durch eine for-Schleife realisiert werden.

Wenn die Größe der In- und Output-Featuremaps eines Faltungslayers gleich sein soll, dann kann dies durch `padding='same'` erreicht werden.

Trainiert werden muss nur eine Epoche als Funktionsnachweis.

Training Neuronaler Netze mit Keras