NNW-Übung 5

Abgabe eienr Auswertung: Dokumentieren Sie in Ihrer Gruppe die Ergebnisse und insbesondere die Antworten zu den Fragen in einer Textdatei. Die Textdatei bitte auf der Moodle-Seite hochladen (Abgabetermin siehe dort; es reicht, wenn ein Gruppenmitglied die Datei hochlädt).

1 TensorFlow Keras Tutorial: Fashion MNIST Klassifikation mit Dense Layers

a) Vollziehen Sie das folgende Tutorial von TensorFlow Zeile für Zeile nach:
`https://www.tensorflow.org/tutorials/keras/classification`

   Am einfachsten ist es, Sie führen es Schritt für Schitt in Google Colab aus: Dazu einfach auf „Run in Google Colab“ klicken und dann mittels „Strg-Enter“ eine Zelle nach der anderen ausführen lassen.

   Natürlich können Sie stattdessen das Notebook auch herunterladen und lokal öffen, z.B. unter Anaconda in das Verzeichnis mit dem Notebook wechseln und `jupyter notebook` eingeben.

b) Wie viele Merkmale haben die Daten (je Trainingsbeispiel)?

c) Entlang welcher Dimension der Input-Matrix werden verschiedene Trainingsbilder gespeichert?

d) Wie sind die Labels (Zielwerte) codiert?

e) Was für Werte liefert die Ausgabeschicht (0 oder 1, One-Hot-kodierte Werte, Integer-Werte, reelle Werte)? Wie viele Ausgaben sind es (je Eingabedatum)? Passt die Codierung der Labels zu den Ausgaben der Ausgabeschicht? Warum funktioniert das? (Siehe auch die nächste Frage!)

f) Welche Art von Fehlerfunktion wird verwendet? Welcher Begriff wird für „Fehlerfunktion“ hier verwendet?

g) Wie gut ist die Accuracy auf den Trainings- und auf den Testdaten?

h) Wie lange hat das Training einer Epoche in etwa gedauert?

i) Lassen Sie sich Anzahl der Gewichte ausgeben. Rechnen Sie die ausgegebene Anzahl nach und übernehmen Sie die Anzahl und den Rechenweg in die Auswertung.

j) Trainieren Sie dasselbe Netz für das klassische MNIST – Sie brauchen dazu nur eine Zeile zu ändern. Übernehmen Sie auch hierfür die Accuracy auf den Trainings- und auf den Testdaten und die ungefähre Trainingszeit je Epoche in die Auswertung. Vergleichen Sie die Resultate mit denen bei Fashion-MNIST.

k) Optional: Speichern Sie das Model ab. Einfaches Abspeichern speichert die Datei zunächst nur lokal auf der VM bei Google. Um die Datei auf Ihrem Google Drive zu sichern, können Sie es „mounten“. Dazu links unter „Code Snippets“ nach „mount“ suchen. Danach: `model.save('/gdrive/My Drive/Datei.h5')`

l) Optional: Ändern Sie die Anzeige der 15 ersten Testdaten so, dass die ersten 15 falsch klassifizierten dargestellt werden – um anhand dessen vielleicht beurteilen zu können, wann das Netz vor allem Fehler macht.

2 ConvNet für Fashion MNIST

Statt eines fully-connected Netzes soll jetzt ein Convolutional Net implementiert werden.

a) Sichern Sie Ihr „Notebook“ – indem Sie es herunterladen oder indem Sie es unter einem neuen Namen in Google Drive abspeichern.

b) Implementieren Sie ein ConvNet, das wie folgt aufgebaut ist:

1. Convolutional Layer mit 8 3x3 großen Kernels, ReLU Aktivierung, Padding „same“.
2. Max-Pooling Layer mit Größe 2x2 und Stride 2.
3. Convolutional Layer mit 16 3x3 großen Kernels, ReLU Aktivierung, Padding „same“.
4. Max-Pooling Layer mit Größe 2x2 und Stride 2.
5. Fully-connected Layer mit 20 Neuronen, ReLU Aktivierung.
6. Ausgabe-Layer

Sie brauchen dafür[1]:

- `keras.layers.Reshape(target_shape, input_shape)`: Die Inputs von Conv2D-Layers müssen 3D sein, x-Dimension × y-Dimenstion × Anzahl Feature Maps/Channels (bei der default-Einstellung `channels_last`; bzw. eigentlich 4D: 0-te Dimension ist Batch-Größe). Da jedes einzelne Input-Bild nur 2D ist, muss die Shape um eine weitere Dimension der Größe 1 ergänzt werden, also `target_shape=single_input_shape+(1,)`, wenn `single_input_shape` die Shape *eines* Bildes ist (konkret hier also 28x28, aber diese Größe sollten Sie natürlich aus den Daten auslesen und nicht hart kodieren).

  Da es die erste Schicht ist, muss bei einem `Sequential`-Model `input_shape` gesetzt werden.
- `keras.layers.Conv2D(filters, kernel_size, strides=(1, 1), padding='valid', activation=None, name=None)`
  - `filters`: Anzahl der zu lernenden Kernels.
  - `kernel_size`: An integer or tuple/list of 2 integers, specifying the height and width of the 2D convolution window. Can be a single integer to specify the same value for all spatial dimensions.
  - `strides`: Strides bei der Faltung selbst (für Aufgabe nicht benötigt).
  - `padding`: `'valid'`: weglassen von Pixeln, für die Kernel aus Bild herausragt; `'same'`: zero-padding.
- `keras.layers.MaxPooling2D(pool_size=(2, 2), strides=None, padding='valid')`
  - `pool_size`: integer or tuple of 2 integers, factors by which to downscale (vertical, horizontal). If only one integer is specified, the same window length will be used for both dimensions.
  - `strides`: Integer, tuple of 2 integers, or None. If None, it will default to `pool_size`.
- `keras.layers.Flatten()`: Flattens the input. Also: einen großen Vektor aus allen Werten aller Feature Maps erstellen. Does not affect the batch size.

c) Fügen Sie die implementierte Definition des Models in die Ausarbeitung ein.

d) Trainieren Sie das Netz mit denselben Einstellungen wie zuvor.

e) Wie gut ist die Accuracy auf den Trainings- und auf den Testdaten?

f) Wie lange hat das Training einer Epoche in etwa gedauert?

g) Optional: Trainieren Sie das Netz erneut, diesmal aber auf der GPU, die Sie bei Google Colab unter Edit/Notebook Settings aktivieren können. Wie lange hat das Training einer Epoche jetzt in etwa gedauert? Übernehmen Sie auch hier die Werte in die Ausarbeitung.

[1] Die Parameterlisten sind auf das Wesentliche gekürzt – für alle Parameter siehe die Keras Dokumentation

h) Lassen Sie sich Anzahl der Gewichte ausgeben. Rechnen Sie die ausgegebene Anzahl nach und übernehmen Sie die Anzahl und den Rechenweg in die Auswertung.

i) Vergleichen Sie die Resultate mit denen des Fully-Connected Netzes aus der vorigen Aufgabe in Bezug auf Accuracy auf den Trainings- und auf den Testdaten, ungefähre Trainingszeiten je Epoche und Anzahl der Gewichte.

j) Fügen Sie eine Dropout-Layer (siehe Foliensatz zu Deep-Feedforward-Networks) nach der letzten MaxPool-Layer hinzu mit einer Dropout-Rate von 0.2. *Achtung*: Beachten Sie, dass diese Layer nach einer Conv-/MaxPool-Layer kommt (siehe den genannten Foliensatz, nach ConvNet suchen). Wie ändert sich die Validierungs- und Test-Accuracy?

k) Trainieren Sie das Modell für weitere 10 Epochen. Das geht am schnellsten, wenn Sie einfach die `model.fit` Zeile nochmals ausführen: Das Netz wird nicht neu initialisiert, sondern ausgehend von dem aktuellen Trainingszustand (also nach den ersten 10 Epochen) weitertrainiert.