Aufgabe 3

Aufgabe 1: Grundelemente (6 Punkte)

1. Nach welchen drei Ebenen wird ein Deep Learning Modell optimiert?

- Initialisierungsgewichte, also die Gewichte, mit welchen die einzelnen Parameter von Beginn an mitgegeben werden
- Der Optimierer bestimmt, wie vom Deep Learning Modell vorgegangen wird, um das Modell zu verbessern, resp. die Verlustfunktion zu minimieren, indem es die Gewichte aktualisiert. Hierzu gibt es verschiedenste Optimierer.
- Die Tuning Parameters, z.B. Epochen, Batch, Anzahl der hidden layers, Aktivierungsfunktion oder Lernrate, werden (teils manuell, teils vorab optimiert durch Training) mitgegeben. Durch die optimale Wahl der Parameter wird das Modell verbessert.

2. Wie funktioniert die Back Propagation?

Das neuronale Netz läuft durch und als Ergebnis erhält man die „Aktivierungsfunktion Output". Danach wird die Verlustfunktion berechnet, auf Basis dessen die Back Propagation stattfindet. Hier werden die Gradienten der Gewichte der Ausgabeschicht neu berechnet. Dies wird an die vorherige Schicht weitergegeben, der Gradient neu berechnet, an die vorherige Schicht weitergegeben, usw….. Dies für jede hidden layer Schicht. Mit Hilfe des Optimierers werden die Gewichte in den einzelnen Schichten aktualisiert. Durch mehrmalige Wiederholungen werden die Gewichte Schritt für Schritt optimiert.

3. Für was wird Overfitting angewendet und wie fliesst dies in die Optimierung ein?

Overfitting tritt auf, wenn ein Modell zu viele Muster in den Trainingsdaten erkennt und dabei zu „genau" ist, sodass das Modell auf neue Daten nicht gut anwendbar / generalisierbar ist. Die Vorhersage der neuen Daten führt zu schlechteren Ergebnissen als wenn das Modell nicht so „genau" wäre. Ausreisser oder Rauschen werden hierbei zu gut berücksichtigt, obwohl diese nicht zwingend in dem Modell berücksichtigt werden müssen.
Ein Modell sollte Overfitting möglichst zu vermeiden. Es sollte die neuen Daten generalisieren und möglichst gut predicten. Durch die Erweiterung der Trainingsdaten, Modellvereinfachung, Regularisierungen oder Dropout kann dem Overfitting entgegengewirkt werden. Dies fliesst so in die Modelloptimierung im Deep Learning mit ein. Diese Vorgehen können manuell bestimmt werden.

Aufgabe 2: Boston Housing (10 Punkte)

Im Aufgabenblatt 2 haben Sie ein Deep Learning Modell für den "Boston Housing" Datensatz erstellt. Das Ziel dieser Aufgabe ist es, diesen nun zu optimieren. Beschreiben Sie die Ebenen nach denen Sie optimieren und die erhaltene Resultate. Gehen Sie auf alle ihre Schritte ein. Zum Schluss beschreiben Sie ihr Resultat. Wie hat sich dieses verbessert?

Man kann die Aktivierungsfunktion, die Verlustfunktion den Optimizer, die Epochen oder Batch-Grösse anpassen.
Zuerst habe ich einen Input als Annahme genommen aus der Vorlesung (z.B. 100 Epochen, 32 Batches) und habe die Aktivierungsfunktion angepasst von ReLu zu Sigmoid, was nichts gebracht hat. Im nächsten Schritt habe ich die Output-Shapes erhöht. Hier wurde der MSE besser bis 250. Als nächstes habe ich den Optimizer angepasst, wodurch der MSE von 20.5 auf 16.2 heruntergekommen ist. Danach haben die Veränderungen in Batch-Size, Epochen, Aktivierungsfunktionen und Optimizer keinen positiven Einfluss mehr auf den MSE gehabt.

(Code aus Aufgabenblatt 2, Code neu angepasst für diese Aufgabe, ebenfalls abgeben in Moodle → Lösung neu erstellt als Übung 2.3 V2)

Aufgabe 3: Interpretation Model MNIST (3 Punkte)

Im Aufgabenblatt 2 haben Sie drei Modelle erstellt. In dieser Aufgabe ist es das Ziel diese Modelle zu vergleichen. Mithilfe des Befehls "model.summary()" können Sie sich die Struktur herausgeben. Wie unterscheiden sich die Modelle?

Im Original waren alle Strukturen gleich. Bei der angepassten Version unterhalb werden je nach gewählter Anzahl Output-Shapes im Layer 1 mehr Parameter getestet. Die Skalierung ist Faktor grösser als bei Layer 2 &3. Die Layer 2 & 3 haben weniger grossen Einfluss auf die Anzahl der getesteten Parameter.

Model 1

```
 Layer (type)                Output Shape              Param #
=================================================================
 dense_3 (Dense)             (None, 256)               200960

 dense_4 (Dense)             (None, 64)                16448

 dense_5 (Dense)             (None, 10)                650

=================================================================
Total params: 218,058
Trainable params: 218,058
Non-trainable params: 0
_________________________________________________________________
```

Modell 2

```
 Layer (type)                Output Shape              Param #
=================================================================
 dense_3 (Dense)             (None, 100)               78500

 dense_4 (Dense)             (None, 64)                6464

 dense_5 (Dense)             (None, 10)                650

=================================================================
Total params: 85,614
Trainable params: 85,614
Non-trainable params: 0
_________________________________________________________________
```

Modell 3a

```
 Layer (type)                Output Shape              Param #
=================================================================
 dense_12 (Dense)            (None, 400)               314000

 dense_13 (Dense)            (None, 20)                8020

 dense_14 (Dense)            (None, 5)                 105

=================================================================
Total params: 322,125
Trainable params: 322,125
Non-trainable params: 0
_________________________________________________________________
```

Modell 3b

```
 Layer (type)                Output Shape              Param #
=================================================================
 dense_15 (Dense)            (None, 100)               78500

 dense_16 (Dense)            (None, 20)                2020

 dense_17 (Dense)            (None, 5)                 105

=================================================================
Total params: 80,625
Trainable params: 80,625
Non-trainable params: 0
_________________________________________________________________
```

(Code siehe Aufgabenblatt 2.2, Code neu angepasst für diese Aufgabe, ebenfalls abgeben in Moodle)