# 5. Übungsblatt

(Neuronale Netze)

Auf diesem Übungsblatt sollen Sie die grundlegende Methodik von Neuronalen Netzen erarbeiten. Folgen Sie den Kommentaren im Quelltext des Moduls *blatt5.aufg09* bzw *blatt5.aufg10*. Beantworten Sie zudem folgende Fragen:

**9. Aufgabe:** Klassifikation des Datensatz data2d mit einem einschichtigen Neuronalen Netz. Überlegen Sie zunächst welche Klassifikationsprobleme sich mit dieser Netzwerk-Topologie lösen lassen und wählen Sie den Trainings- und Testdatensatz entsprechend.

1. Implementieren Sie ein Neuronales Netz, welches zwei Eingabe- und ein Ausgabeneuron enthält. Das Ausgabeneuron soll eine lineare Aktivierungsfunktion verwenden. Berücksichtigen Sie auch das *bias*-Gewicht. Visualisieren Sie die Ergebnisse nach den einzelnen Trainingsiterationen.
   **Hinweise:** Ein solches Neuronales Netz lässt sich durch eine Matrixmultiplikation auswerten. Für die Aktualisierung der Gewichte können Sie die Änderungen je Sample mitteln. Wählen Sie die Lernrate nicht zu groß.
2. Bevor ein neuronales Netz trainiert wird, sollten die Daten auf eine bestimme Weise vorverarbeitet werden. Wie sollte die Vorverarbeitung aussehen? Warum ist sie sinnvoll?
3. Welche Klassifikationsprobleme lassen sich mit einem Perceptron lösen?
4. Was sind One-Hot Encodings und warum werden Sie zum Training von neuronalen Netzen benötigt?
5. Welchen Zweck erfüllt das Bias-Gewicht?
6. Wie wirken sich unterschiedliche Lernraten aus?
7. Warum springt die Hyperebene beim Training teilweise zurueck zu einem schlechteren Ergebnis?
8. Wie verhalten sich die Ergebnisse im Vergleich zu einer linearen SVM?
9. *(optional)* Implementieren Sie ein Neuronales Netz, mit dem sich beliebige 2-Klassenprobleme auf dem data2d Datensatz lösen lassen.
10. *(optional)* Erweitern Sie Ihr Netz zur Lösung des 3-Klassenproblems.

**10. Aufgabe:** Nehmen Sie an, dass ein Datensatz aus zwei Klassen des MNIST Ziffern vorliegt und diese klassifiziert werden sollen. Implementieren Sie die in Abbildung 1 dargestellte Architektur mithilfe von PyTorch, um das Problem zu lösen. Eine einfache CNN Architektur für den MNIST Datensatz könnte verallgemeinert wie folgt aufgebaut sein: [EINGABE - CONV - RELU - FC].

Die EINGABE [28x28x1] enthält die Pixelwerte des Bildes. In diesem Fall hast das Bild eine Breite von 28 Pixeln, eine Höhe von 28 Pixeln mit einem Farbkanal (Graustufen).

Die CONV Schicht berechnet die Ausgabe von Neuronen, welche lokal mit Regionen der Eingabe verbunden sind. Bei der Festlegung auf 10 Filter führt dies zu einem [28x28x10] Volumen.

In der RELU Schicht wird eine Aktivierungsfunktion elementweise angewendet.

Die FC (fully-connected) Schicht berechnet Scores für die Klassen, was zu einem Vektor mit zwei Elementen als Ausgabe, von denen jeder den Score für eine der beiden ausgewählten MNIST Klassen darstellt.

**Optional:** Passen Sie das Modell für die Klassifikation aller MNIST Klassen an und versuchen Sie möglichst gute Ergebnisse auf dem Testdatensatz zu erreichen. Verwenden Sie zur Lösung dieser Aufgabe die PyTorch Dokumentation (`https://pytorch.org/docs/`).

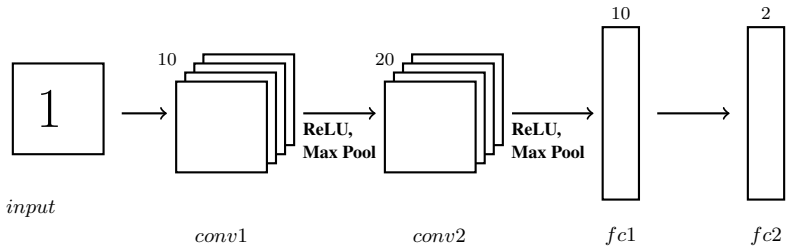


Abbildung 1: Convolutional Neural Network Architectur