MACHINE LEARNING II

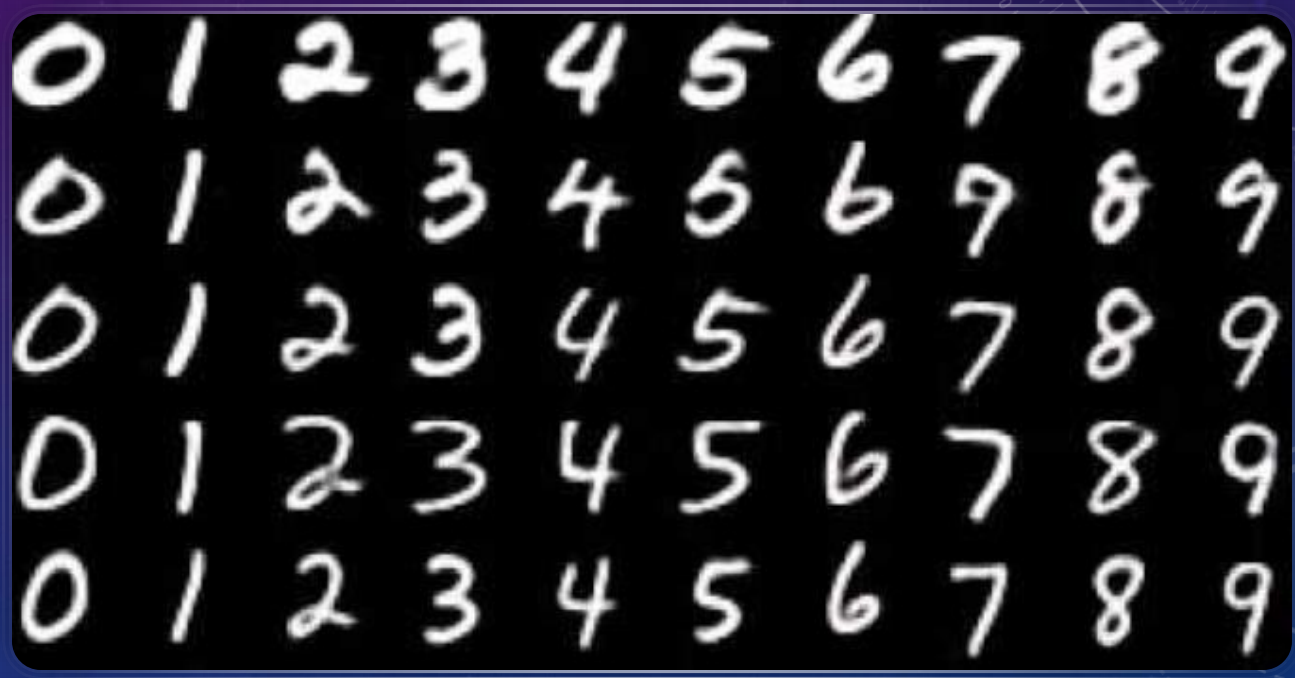
MATTHIAS TSCHÖPE, KUNAL OBEROI

KLASSIFIKATION

- ***Die Aufgabe dieser Übung ist es, handgeschriebene Zahlen zu erkennen (MNIST).***

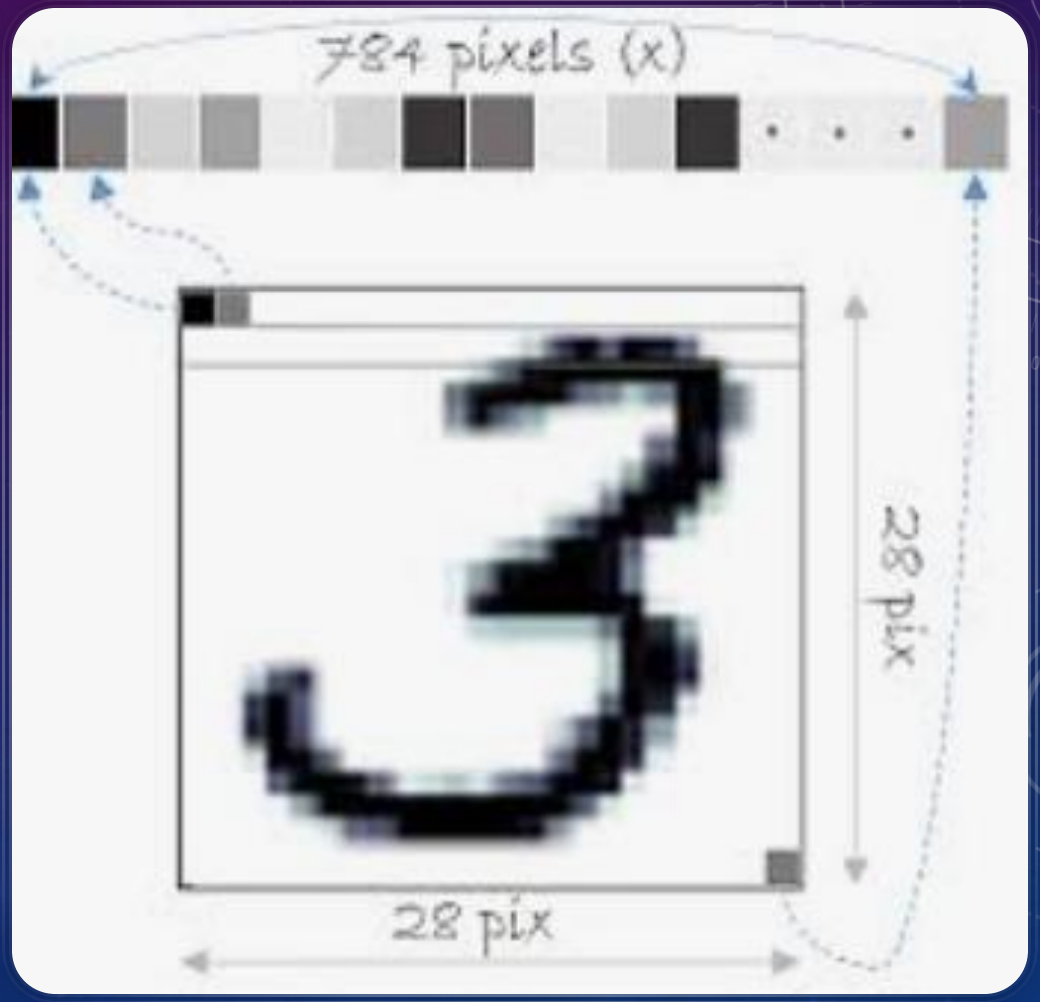
DATENSATZ

- Wir verwenden den MNIST Datensatz, der Datensatz besteht aus
  - 70.000 kleinen hangeschriebenen Bilder von Zahlen
  - Zahlen von 0 bis 9



DATENSATZ

- Kurzer Überblick eines Bildes
  - 28 x 28 pixels -> 784 Features
  - 70000 Bilder bilden eine Dimension von 70000 x 784

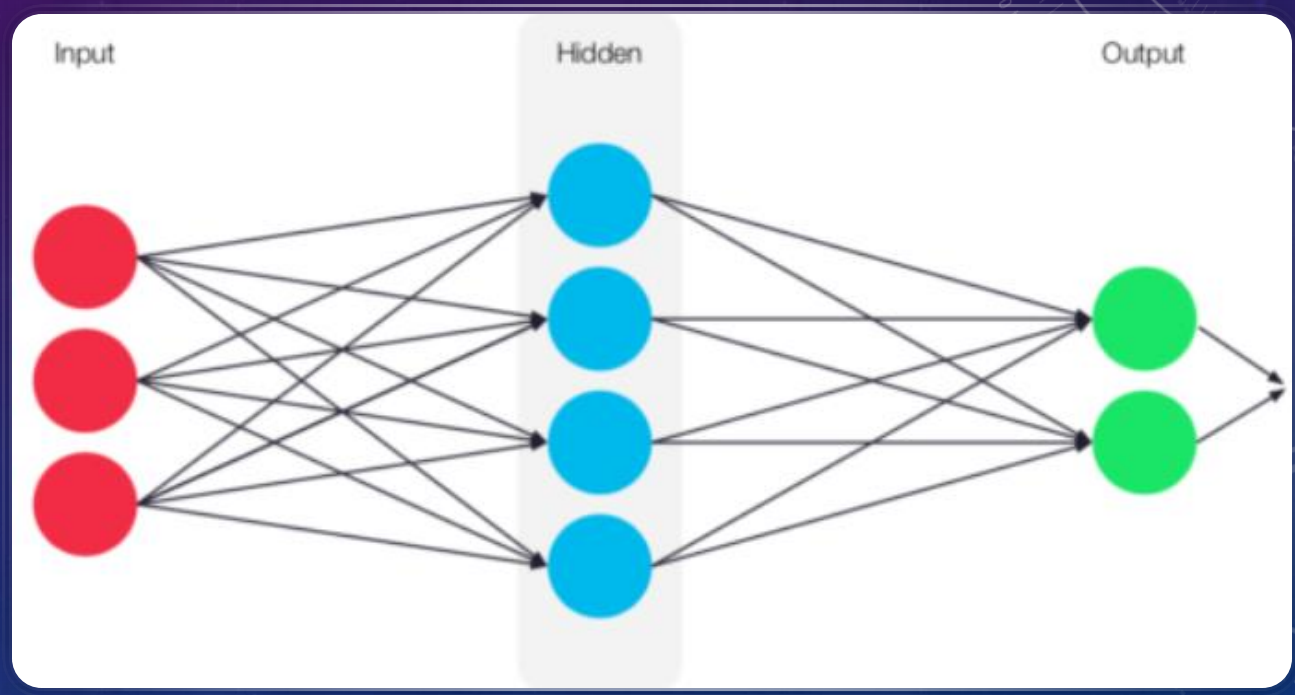


AUFGABE 1 (MNIST-DATASET)

- a) Lesen sie die **Daten** ein indem, sie die vorgegebenen Dateien einlesen. Die Daten sind bereits in **Trainingsdaten** und **Testdaten** gerichtet.
- b) Bevor Sie die Daten skalieren, reduzieren wir durch **"Flatten"** die Dimensionen der Trainings- und Testdaten von 3 zur Dimension 2
  - **Hinweis: flatten() gibt es bereits als eine Funktion von Numpy**
- c) Verwenden Sie die den **StandardScaler()** um die Daten in eine Einheit zubringen.
- d) **Visualisieren sie mit Hilfe von Matplotlib ein beliebiges Bild**
  - **Hinweis: imshow() und reshape() können euch helfen**

MULTILAYER PERCEPTRON

- **Input Layer:** Initialisiert Daten für das Netz
- **Hidden Layer:** Repräsentiert eine Schicht zwischen Input und Output. Hier werden alle Berechnungen durchgeführt.
- **Output Layer:** Ergebnis basiert auf den Input

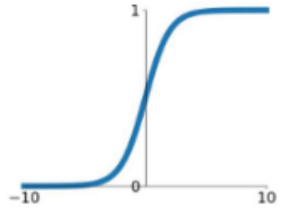


AKTIVIERUNGSFUNKTION (KURZ-EINFÜHRUNG)

Activation Functions

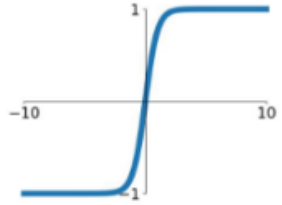
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



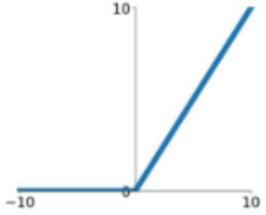
tanh

$$\tanh(x)$$



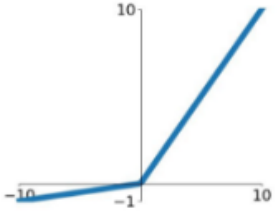
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

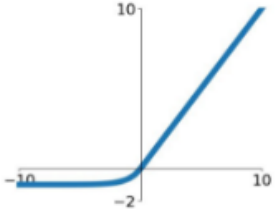


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



OPTIMIERER(KURZ-EINFÜHRUNG)

- Bereits bekannt ist Gradient Descent und Normalgleichung, es gibt noch weitere z.B:
  - Adam
  - Adagrad
  - RMSprop
    - Wir verwenden Adam
      - Die adaptive Lernrate wird durch einen Vektor repräsentiert, dies bietet für jede Richtung eine individuelle Lernrate. Dadurch konvergiert Adam wesentlich schneller als SGD mit gleichen Hyperparametern

AUFGABE 2 (MULTILAYER PERCEPTRON)

- a) Importieren sie MLP von **from sklearn.neural_network import MLPClassifier** tränieren sie das Modell anhand der folgenden Kriterien:
  - Aktivierungsfunktion = "relu",
  - Solver = "adam",
  - hidden_layers = (128,64,32) ,
  - Lernrate = "adaptive",
  - Initialiseren die Lernrate mit = 0.001,
  - Batch_size = 64,
  - Letztendlich mit 10 Iterationen, probieren Sie alternativ auch andere Kombinationen aus.
- b) Geben Sie den vorhergesagten Wert **predict()** und die Genauigkeit **score()** des Models an.

AUFGABE 3 (CONFUSION-MATRIX UND VALIDATION)

- a) Wie bereits in der letzten Woche, haben wir die **Confusions-Matrix** kennen gelernt, Plotten sie diese wie in der letzten Woche und Interpretieren Sie die **Actual** und **Predicted** Values! Geben sie die **classification_report()** aus.
  - **Hinweis: from sklearn.metrics import confusion_matrix, classification_report**
- b) Geben Sie abschließend den **cross_val_score()** für das Modell an.