Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

inovex

AI Labor - Sommersemester 2019

Computer Vision
2. Termin

Robin Baumann, Stanislav Frolov

Karlsruhe, 29. März 2019

Agenda für Heute

- Recap: Allgemeines
- Einführung TensorFlow
- Theorie zu MLPs
- Praktischer Teil: Bildklassifizierung

Modus

- 14 Termine, jeweils 2 Blöcke
- 4 ECTS, 4 SWS

- Gruppenarbeit (3 Personen), max. 18 Personen

- Bewertung
  - Diskussion und Vorstellung der Lösungen + Fragen
    - am Ende oder während des nächsten Termins

Modus

Zulassungskriterien

1. Studierende von I und Modul ML vollständig bestanden
2. Studierende von I und Klausur ML bestanden
3. Studierende von anderen Fakultäten und Klausur ML bestanden (mit entsprechenden Python-Kenntnissen)
4. Auswahl im Losverfahren aus dem Rest der Welt (mit entsprechenden Methoden- und Python-Kenntnissen)

Teamvorstellung

- Computer Vision (4 Termine)
  - Robin Baumann, Stanislav Frolov
- Natural Language Processing (5 Termine)
  - Anna Weisshaar, Tilman Wittl
- Reinforcement Learning (5 Termine)
  - Benedikt Hagen, Frederik Martin

Computer Vision

1. Python, Jupyter, Grundlagen Computer Vision
2. TensorFlow, MLP, Bildklassifizierung
3. CNN, Objektklassifikation
4. AI auf Android Smartphone

Einführung TensorFlow

TensorFlow

Allgemeines

- “open source machine learning framework for everyone”
- wird von Google entwickelt
- November 2015 veröffentlicht
- populärstes Framework für machine learning und künstliche neuronale Netze

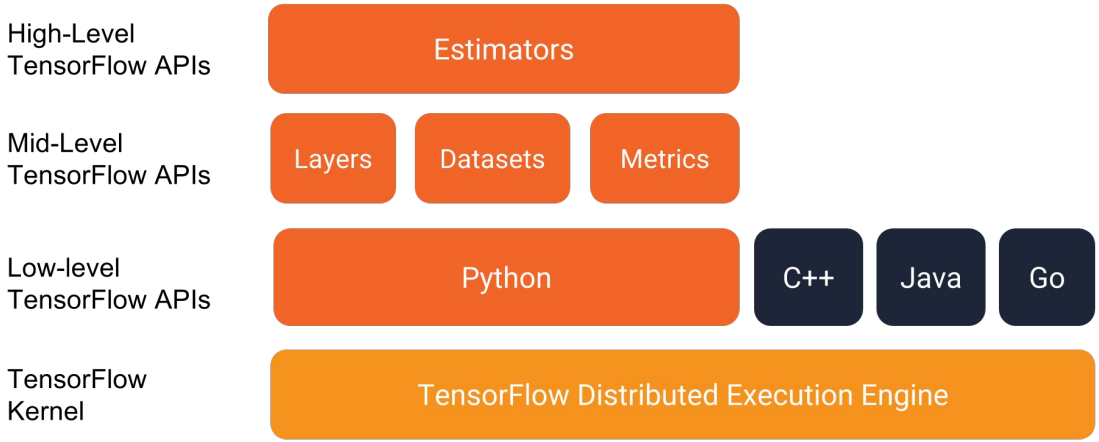
TensorFlow

- Fokus auf Training und Inferenz von tiefen Netzen
- “Large-Scale Machine Learning on Heterogeneous Distributed Systems”
- *Whitepapers*
- kann bspw. mit *pip* installiert werden

TensorFlow

Architektur

- flexible Architektur
- unterstützt CPUs, GPUs, TPUs, CUDA, Linux, Windows, macOS, Android, iOS



https://www.tensorflow.org/guide/premade_estimators#the_programming_stack

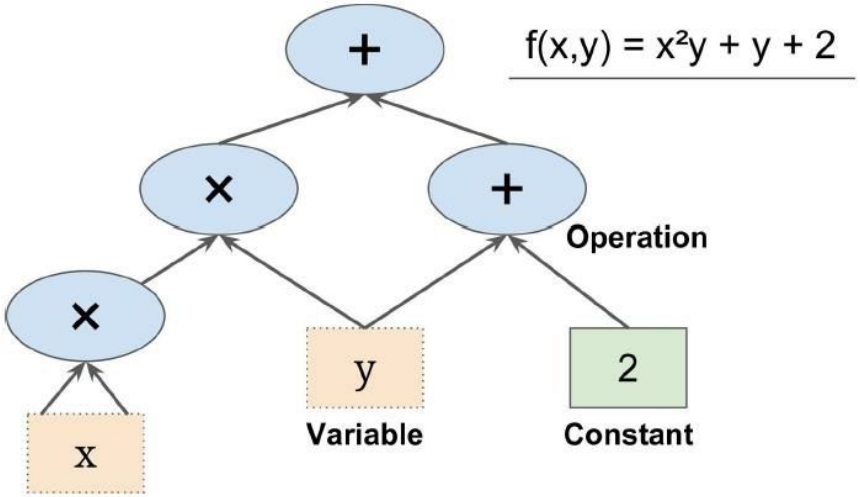
TensorFlow

Grundlagen

- datenstromorientiert
- baut intern einen Datenflussgraph auf (data flow graph / computational graph)
- jeder Knoten steht für eine Berechnung, die auf den Daten ausgeführt wird (z.B. add, mult, sum, tanh, …)
- Kanten repräsentieren den Datenfluss / Tensoren
- komplette Berechnung wird als gerichteter Graph repräsentiert

TensorFlow

Beispiel einfacher Graph



inovex

TensorFlow

Grundlagen

- Programmablauf in Phasen
  - Konstruktion des Graphen
  - Ausführung des Graphen mit Daten
- Graph wird einmal statisch definiert, bevor das Modell ausgeführt wird (keine dynamische Anpassung des Graphen)

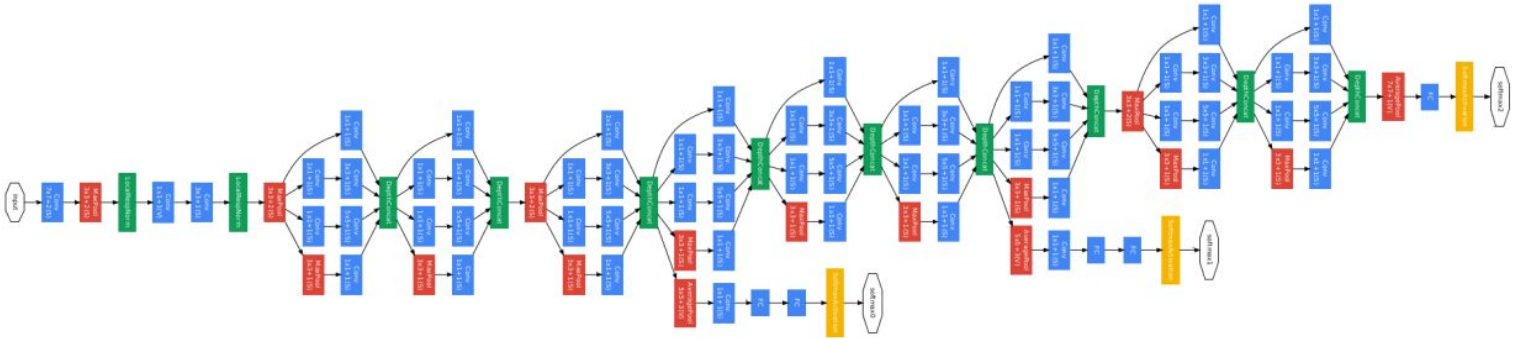
TensorFlow

Tensor

- ein Tensor ist eine mathematische Funktion die eine Anzahl von Vektoren auf einen Wert abbildet
- Verallgemeinerung von Skalar, Vektor, Matrix
- multidimensionales Array/Matrix
- Tensoren fließen zwischen den Knoten des Graphen

TensorFlow

Beispiel komplexer Graph - GoogleNet



https://arxiv.org/pdf/1409.4842.pdf

TensorFlow

Beispiel TensorFlow Operationen

Category	Examples
Element-wise mathematical operations	Add, Sub, Mul, Div, Exp, Log, Greater, Less, Equal, ...
Array operations	Concat, Slice, Split, Constant, Rank, Shape, Shuffle, ...
Matrix operations	MatMul, MatrixInverse, MatrixDeterminant, ...
Stateful operations	Variable, Assign, AssignAdd, ...
Neural-net building blocks	SoftMax, Sigmoid, ReLU, Convolution2D, MaxPool, ...
Checkpointing operations	Save, Restore
Queue and synchronization operations	Enqueue, Dequeue, MutexAcquire, MutexRelease, ...
Control flow operations	Merge, Switch, Enter, Leave, NextIteration

http://download.tensorflow.org/paper/whitepaper2015.pdf

TensorFlow

Beispiel TensorFlow

```
import tensorflow as tf

b = tf.Variable(tf.zeros([100]))                    # 100-d vector, init to zeroes
W = tf.Variable(tf.random_uniform([784,100],-1,1)) # 784x100 matrix w/rnd vals
x = tf.placeholder(name="x")                        # Placeholder for input
relu = tf.nn.relu(tf.matmul(W, x) + b)              # Relu(Wx+b)
C = [...]                                           # Cost computed as a function
                                                    # of Relu


s = tf.Session()
for step in xrange(0, 10):
  input = ...construct 100-D input array ...        # Create 100-d vector for input
  result = s.run(C, feed_dict={x: input})           # Fetch cost, feeding x=input
  print step, result
```


http://download.tensorflow.org/paper/whitepaper2015.pdf

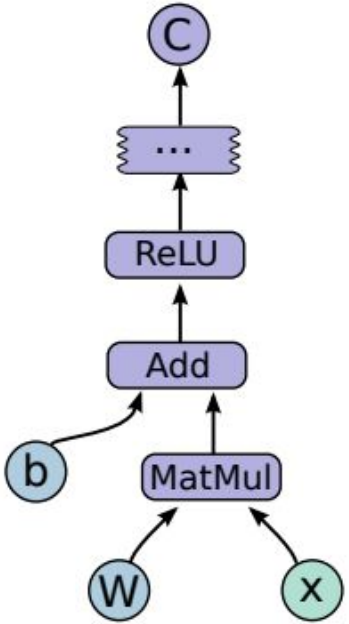
TensorFlow

Beispiel TensorFlow

```
import tensorflow as tf

b = tf.Variable(tf.zeros([100]))
W = tf.Variable(tf.random_uniform([784,100],-1,1))
x = tf.placeholder(name="x")
relu = tf.nn.relu(tf.matmul(W, x) + b)
C = [...]


s = tf.Session()
for step in xrange(0, 10):
  input = ...construct 100-D input array ...
  result = s.run(C, feed_dict={x: input})
  print step, result
```

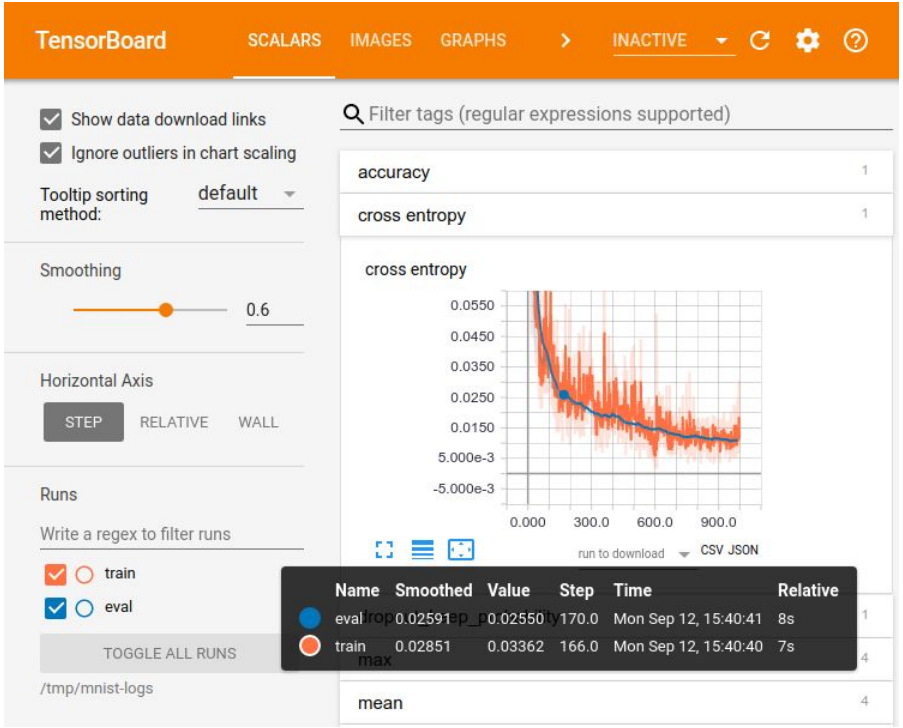


http://download.tensorflow.org/paper/whitepaper2015.pdf

TensorFlow

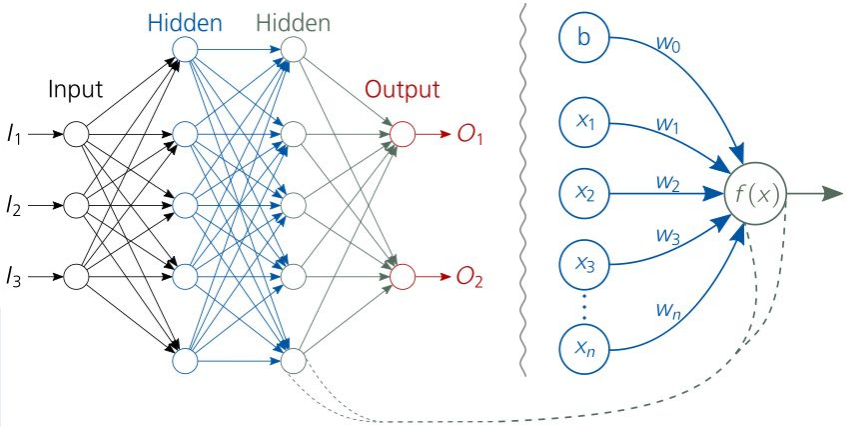
Visualisierung mit TensorBoard

- Webapplikation, um den Graphen und den Verlauf des Trainings zu untersuchen
- `tensorboard --logdir=path/to/dir`
- *localhost:6006*



https://arxiv.org/pdf/1409.4842.pdf

Einführung MLP und MNIST



inovex

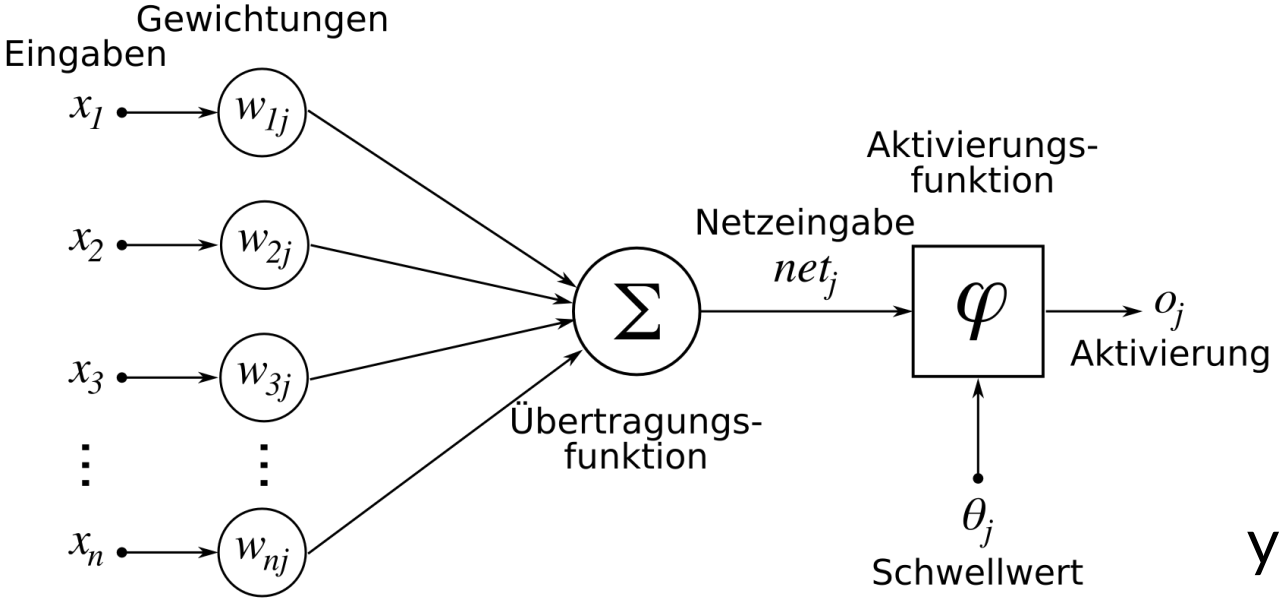
Künstliche Neuronale Netze

Überblick

- Anfänge in den 50ern
- Inspiriert von biologischen neuronalen Netzen
- Entwicklungen aufgrund mangelnder Ergebnisse, wenig Daten, unzureichender Rechenleistung eingestellt (schwer zu trainieren)
- Renaissance seit 2012 mit Deep Learning

Ein künstliches Neuron

Das Perzeptron

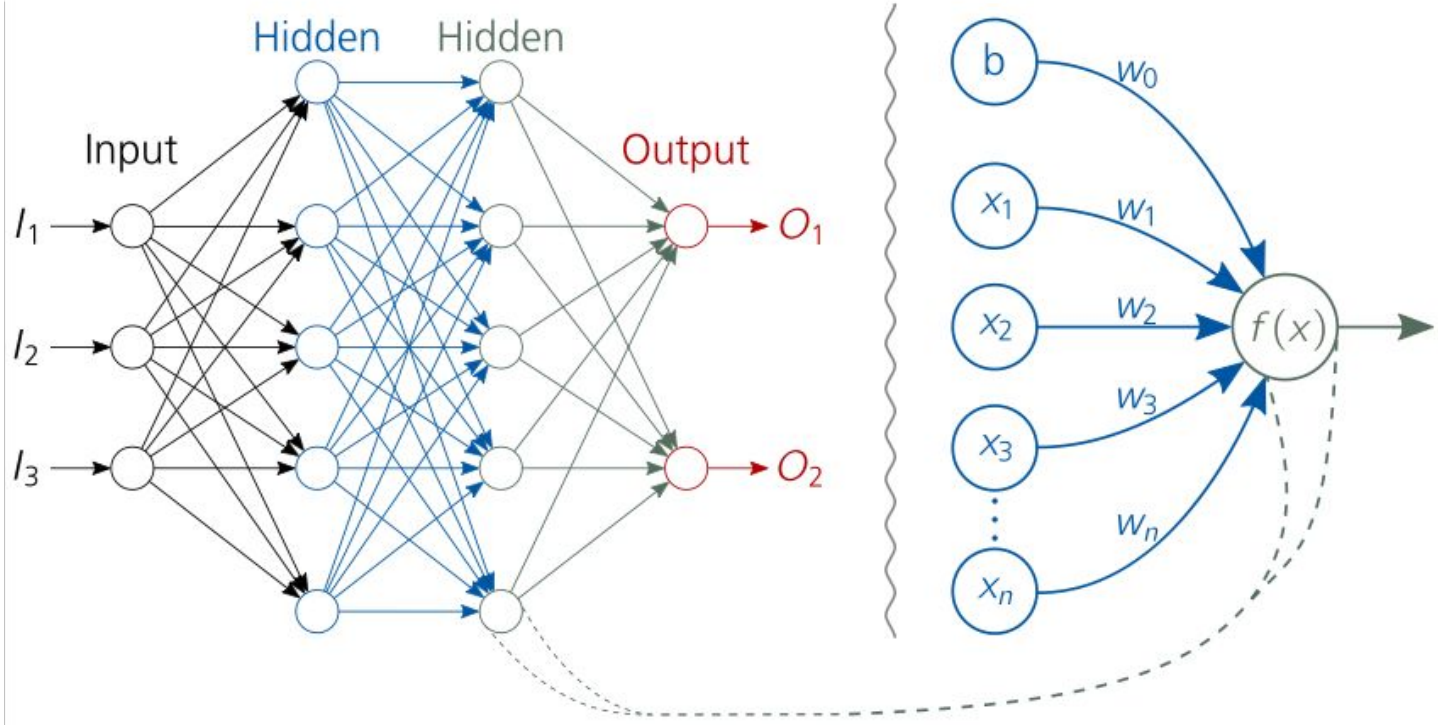


y=f(wx+b)

inovex

Ein künstliches Neuron

Multi-Layer Perceptron (MLP)



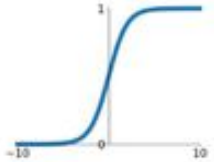
inovex

Aktivierungsfunktionen

Activation Functions

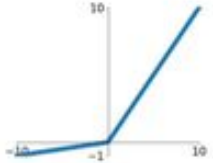
Sigmoid

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



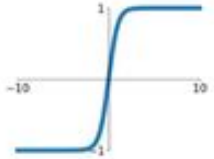
Leaky ReLU

$\max(0.1x, x)$



tanh

$\tanh(x)$



Maxout

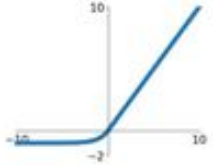
$\max(w_1^T x + b_1, w_2^T x + b_2)$

ReLU

$\max(0, x)$

ELU

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



Training

Bildklassifizierung

- Input: Bilder und Labels
- Output: Klassen und Score

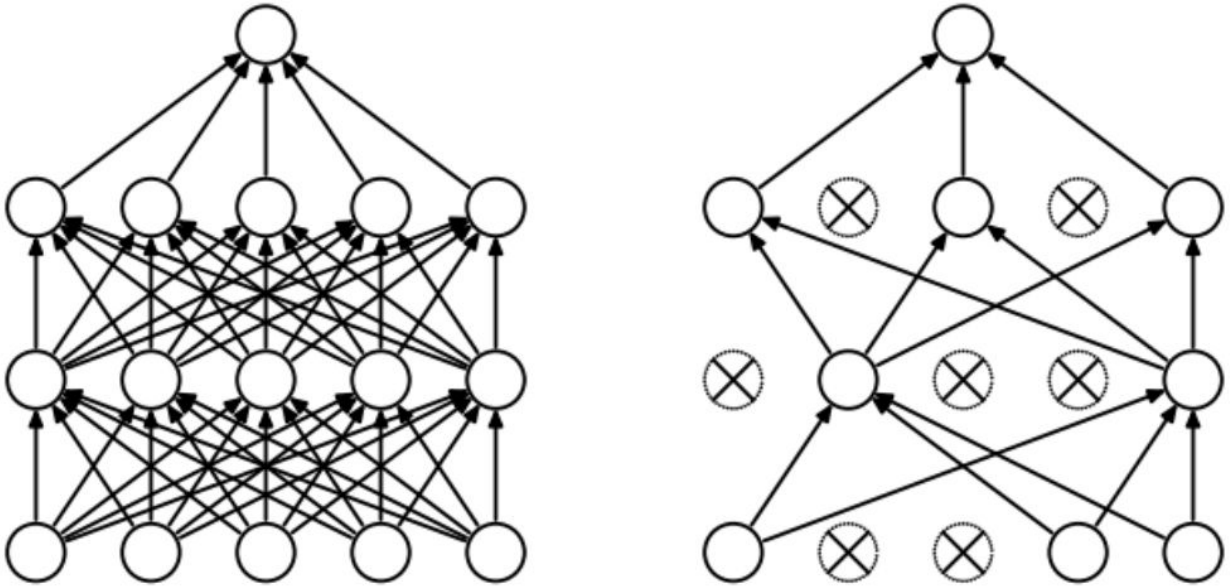
- Optimierungsproblem
  - minimiere Loss durch iterativen Prozess

Training

Bildklassifizierung

- Forward-Pass von einem Batch
- Berechne Fehler zwischen erzeugtem und gewünschtem Output
- Verändere die Gewichte, sodass der gleiche Forward-Pass einen kleineren Fehler erzeugen würde
- Wiederhole bis Konvergenzkriterium erreicht

Dropout



(a) Standard Neural Net

(b) After applying dropout.

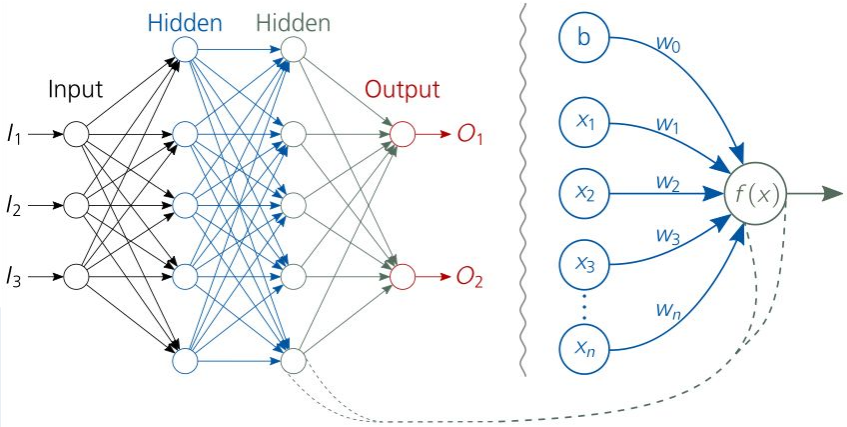
Datensatz

MNIST

- 60k Training
- 10k Test
- 28x28 Pixel

http://yann.lecun.com/exdb/mnist/

Praktischer Teil



Gruppenfindung

- Dreierteams bilden
- Name und Gruppe auf Zettel schreiben

Setup

- Anmeldung am Rechner

- Ethernet auf `hskaopen` umstellen
- Wired Settings:
  - `testXYZ` entfernen und mit Studentenkürzel ersetzen
  - Passwort eingeben

Setup

- Terminal öffnen
  - `git clone https://github.com/hskaailabcv/source.git`
  - `cd source`
  - `docker-compose up`
- Jupyter: *http://localhost:8888*

Feedback

- Google Forms
  https://forms.gle/Uy1LuRwcizcqniDK7

Vielen Dank

Robin Baumann
rbaumann@inovex.de

Stanislav Frolov
sfrolov@inovex.de

inovex GmbH
Ludwig-Erhard-Allee 6
76131 Karlsruhe