



Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES



inovex

AI Labor - Wintersemester 2019/20

Computer Vision
2. Termin

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, Matthias Richter
- Natural Language Processing (5 Termine)
 - Anna Weisshaar, Tilman Berger
- Reinforcement Learning (5 Termine)
 - Benedikt Hagen, Frederik Martin

Computer Vision

1. Python, Jupyter, Grundlagen Computer Vision
2. TensorFlow, MLP, Bildklassifizierung
3. CNN, Objektklassifikation, Keras
4. Transfer Learning, 3D Deep Learning



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
- Seit dem 30.09.2019 in Version 2.0!

TensorFlow

Architektur

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

High-Level
TensorFlow APIs

Estimators

Mid-Level
TensorFlow APIs

Layers

Datasets

Metrics

Low-level
TensorFlow APIs

Python

C++

Java

Go

TensorFlow
Kernel

TensorFlow Distributed Execution Engine

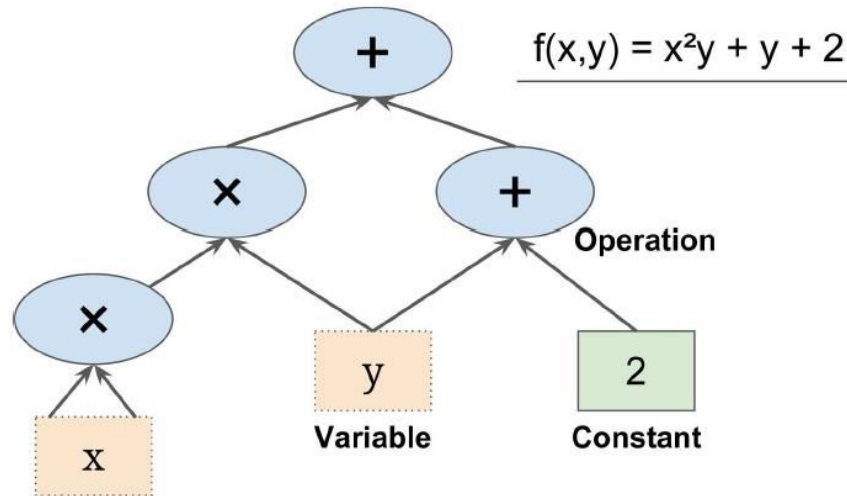
TensorFlow

Grundlagen

- datenflussorientiert
- 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



TensorFlow

Grundlagen

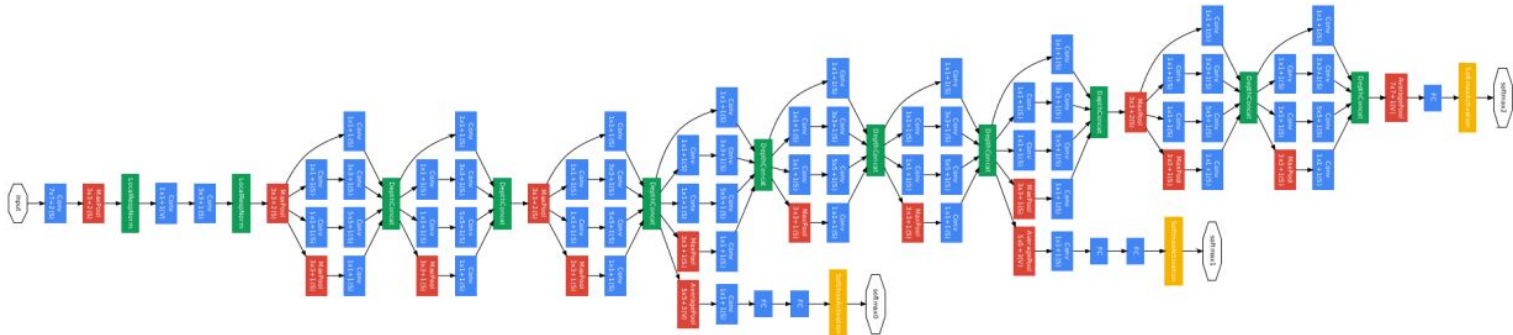
- **Programmablauf in Phasen**
 - Konstruktion des Graphen
 - Ausführung des Graphen mit Daten
- **Graph wird einmal statisch definiert, bevor das Modell ausgeführt wird (keine dynamische Anpassung des Graphen)**

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

Beispiel komplexer Graph - GoogleNet



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

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
```

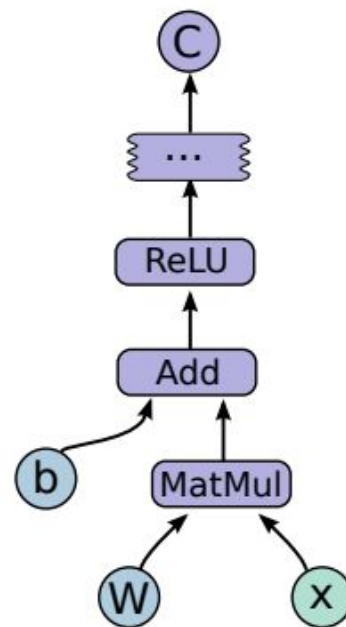
TensorFlow

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 = [...]

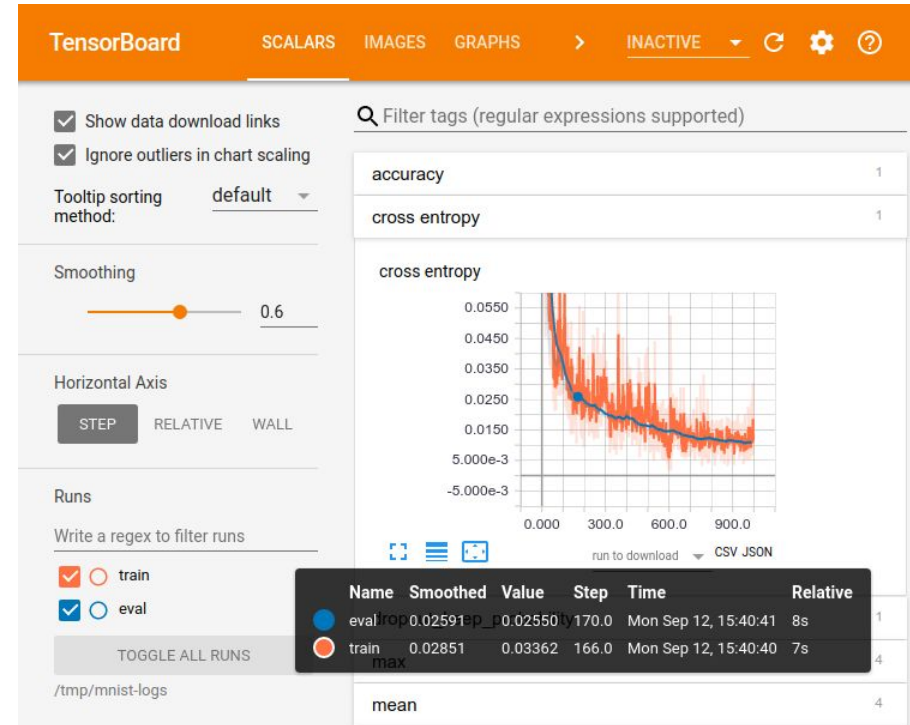
s = tf.Session()
for step in xrange(0, 10):
    input = ...construct 100-D input array ...
    result = s.run(C, feed_dict={x: input})
    print step, result
```



TensorFlow

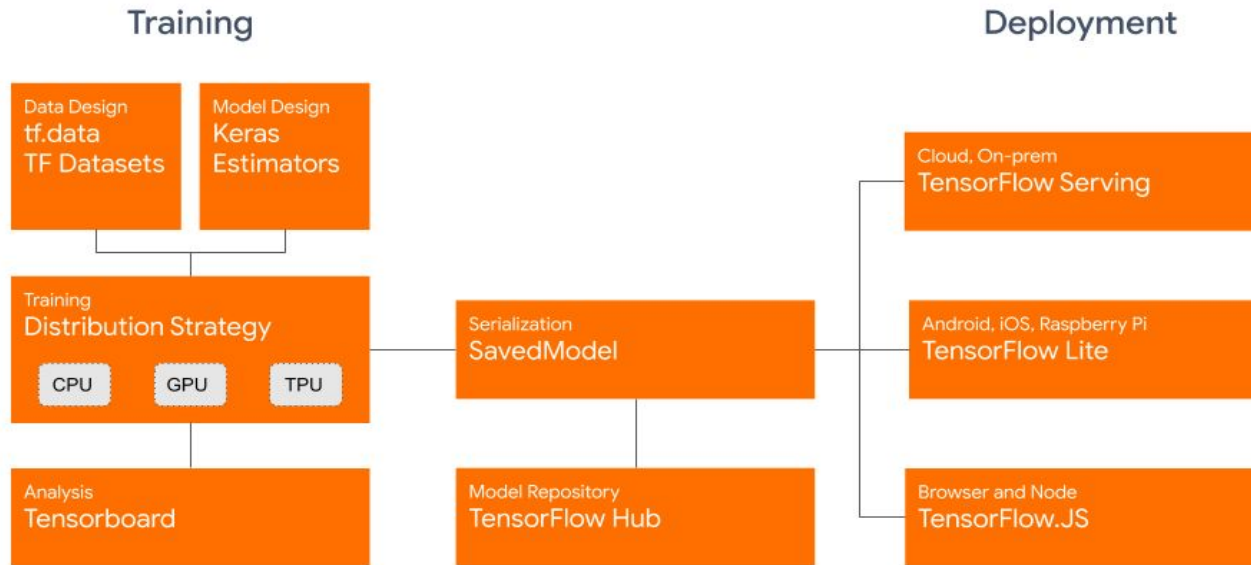
Visualisierung mit TensorBoard

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

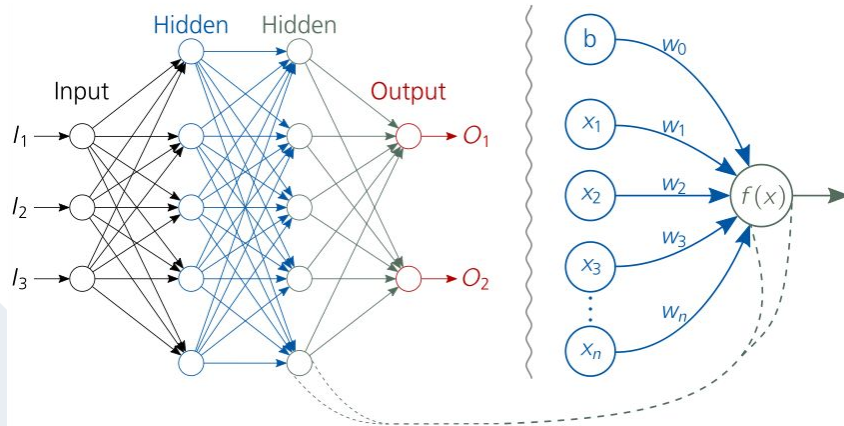


TensorFlow

Übersicht Release 2.0



Einführung MLP und MNIST



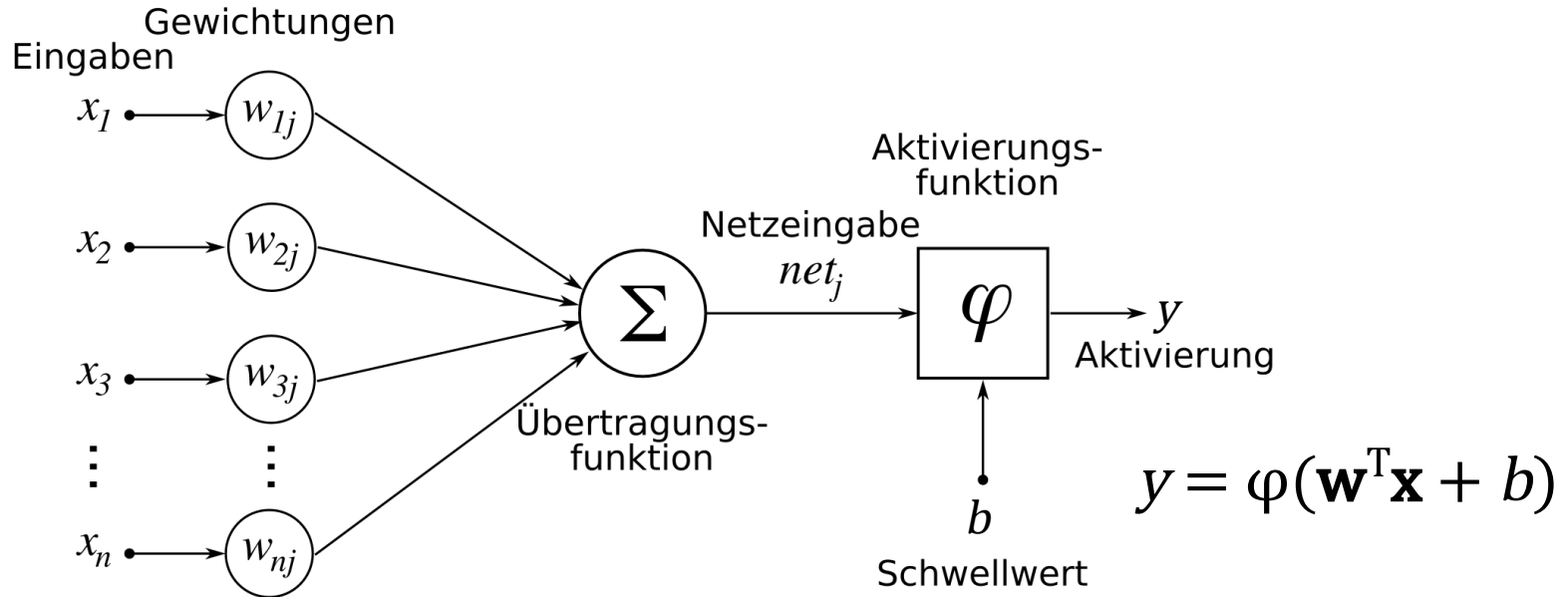
Künstliche Neuronale Netze

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
 - Mehr Daten (ImageNet)
 - Mehr Rechenleistung (GPU)
 - Bessere Lernalgorithmen (Stochastic Gradient Descent)

Ein künstliches Neuron

Das Perzeptron

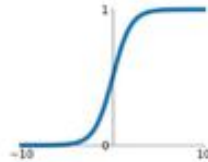


Aktivierungsfunktionen

Activation Functions

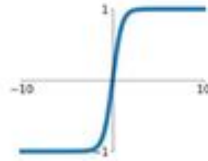
Sigmoid

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



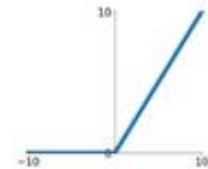
tanh

$$\tanh(x)$$



ReLU

$$\max(0, x)$$



Leaky ReLU

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

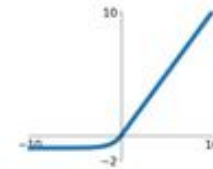


Maxout

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

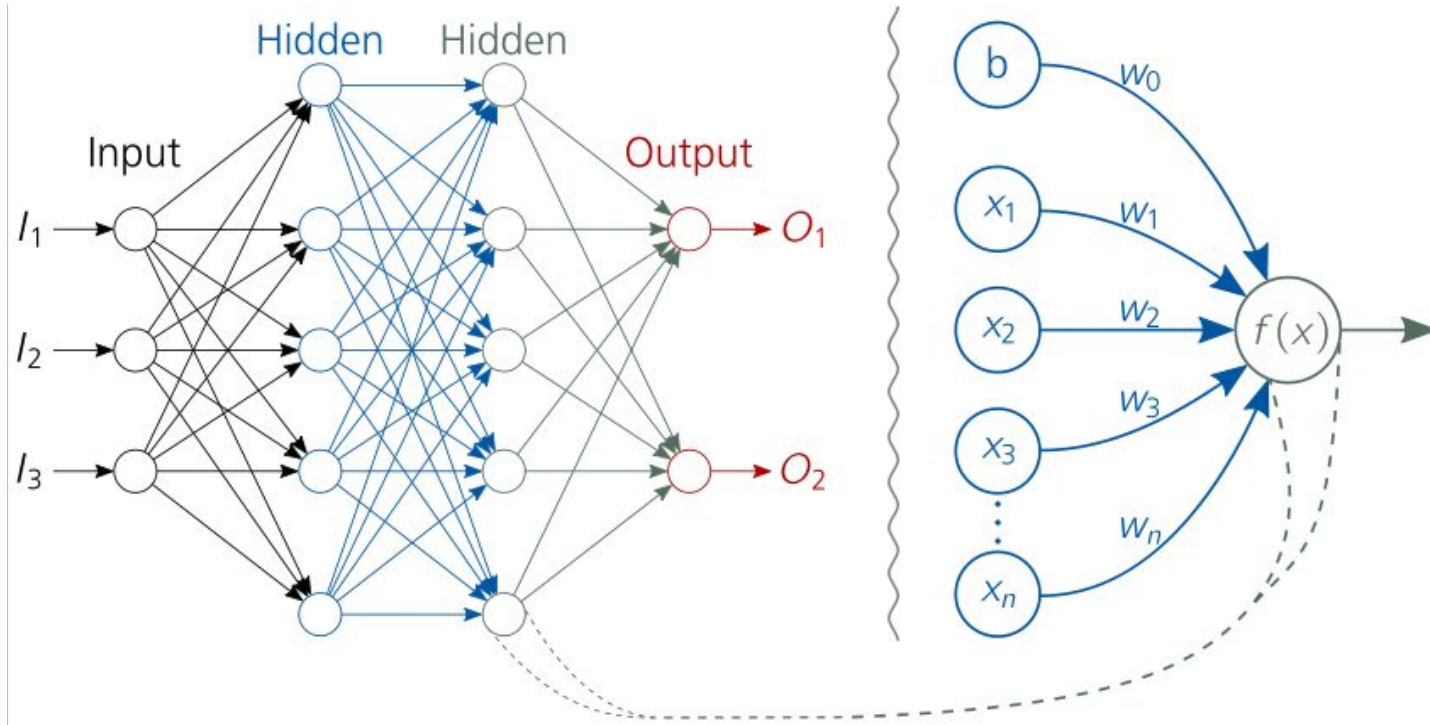
ELU

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



Ein künstliches Neuron

Multi-Layer Perceptron (MLP)



Training

- Input: Bilder und Labels
- Output: Score bzw. Konfidenz für jede Klasse
- Optimierungsproblem (Empirical Risk Minimization)
 - minimiere Loss durch iterativen Prozess (Backpropagation)
 - Anzahl falsch klassifizierter Inputs
 - Konfidenz der Fehlklassifikationen
 - Mean Square Error (MSE)
 - ...



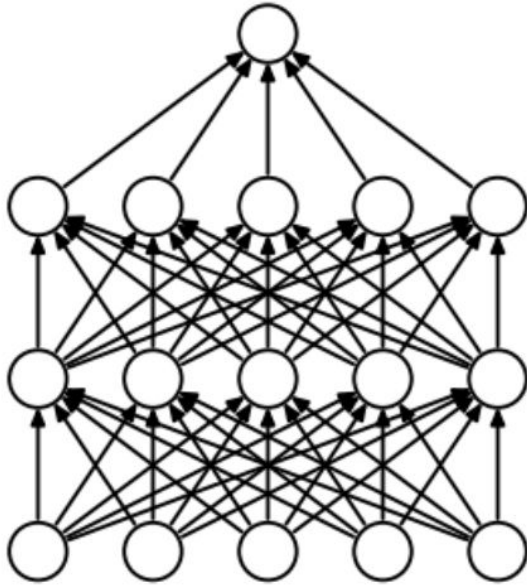
Training

Backpropagation

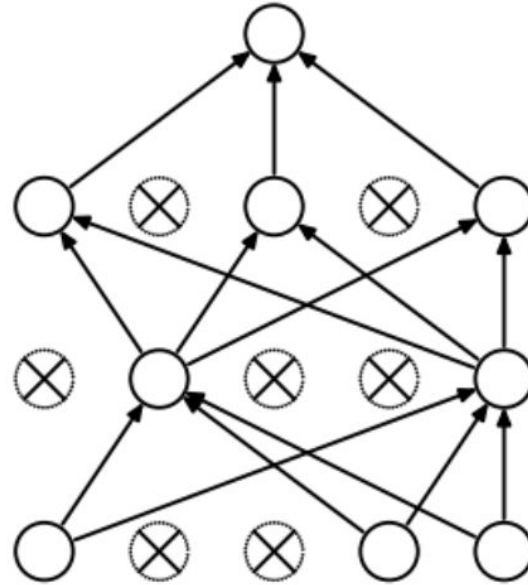
- Forward-Pass von einem Batch
- Berechne Fehler zwischen erzeugtem und gewünschtem Output
- Verändere die Gewichte, sodass der gleiche Forward-Pass einen kleineren Fehler erzeugen würde
- Wiederhole bis Konvergenzkriterium erreicht

⇒ Gradientenabstieg auf Gewichten bezüglich Loss

Dropout



(a) Standard Neural Net



(b) After applying dropout.

Datensatz

MNIST

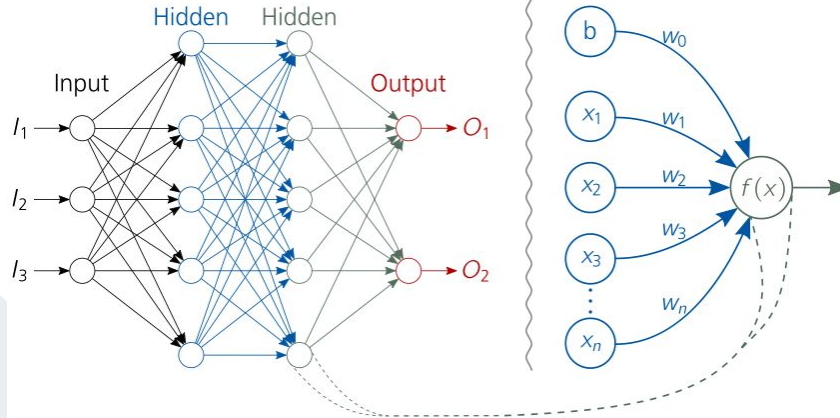
- 60k Training
- 10k Test
- 28x28 Pixel

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





Praktischer Teil



Gruppenfindung

- Dreierteams bilden
- Name und Gruppe auf Zettel schreiben

Setup

- Anmeldung am Rechner
 - User: i-lfm-docker
 - Passwort: garten::obst
- Ethernet auf `hskaopen` umstellen
- Wired Settings > Security:
 - ☒ No CA certificate is required
 - `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/4fuXeTbqY3qZC9fQ9>



Vielen Dank

Robin Baumann

rbaumann@inovex.de

Matthias Richter

mrichter@inovex.de

inovex GmbH

Ludwig-Erhard-Allee 6

76131 Karlsruhe

