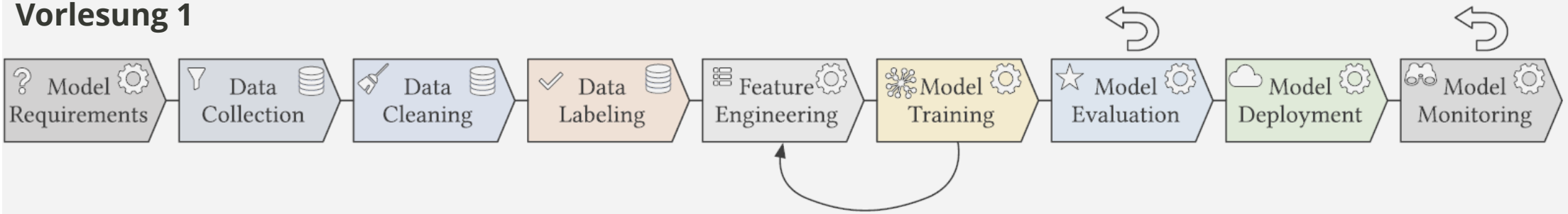
Dr. rer. nat. Valentin Khaydarov
Professur für Prozessleittechnik & Arbeitsgruppe Systemverfahrenstechnik

Neuronale Netze

Vorlesung 7, Lehrveranstaltung Experimentelle Prozessanalyse

Einordnung der Vorlesung

Vorlesung 1



Vorlesung 2

Vorlesung 3 – Regr.

Vorlesung 4 – Class.

Vorlesung 5 – Clust.

Vorlesung 6 - Zeitreihenanalyse

Vorlesung 7 – Neuronale Netze

S. Amershi *et al.*, "Software Engineering for Machine Learning: A Case Study," in *Proceedings - 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice, ICSE-SEIP 2019*, May 2019, pp. 291–300, doi: 10.1109/ICSE-SEIP.2019.00042.

Agenda

- Intuition und Wiederholung der Regressionsanalyse
- Motivation und Grundlagen
- Feed-Forward Neuronale Netze
- Lernen in Neuronalen Netzen
- Regularisierung
- Arbeitsweise bei der Modellbildung
- Frameworks
- Zusammenfassung

Intuition

Wiederholung Regressionsanalyse

Vorgegeben:

- n Beobachtungen mit m Eingangsgrößen $x_1, \dots, x_m$ und einer Ausgangsgröße y
  - Eingangsgrößen können ebenfalls nominal oder ordinal sein
- eine Hypothese $y = f(b, x)$

Ziel

- Ermittlung eines funktionalen Zusammenhang zwischen m Eingangsgrößen und einer ausgewählten Zielgröße y

Regressionsanalyse

Lineare Hypothesen:

- Einfachregression ($m = 1$):

$$h(x) = b_0 + bx_1 \text{ oder eine Gerade im } \mathbb{R}^2$$

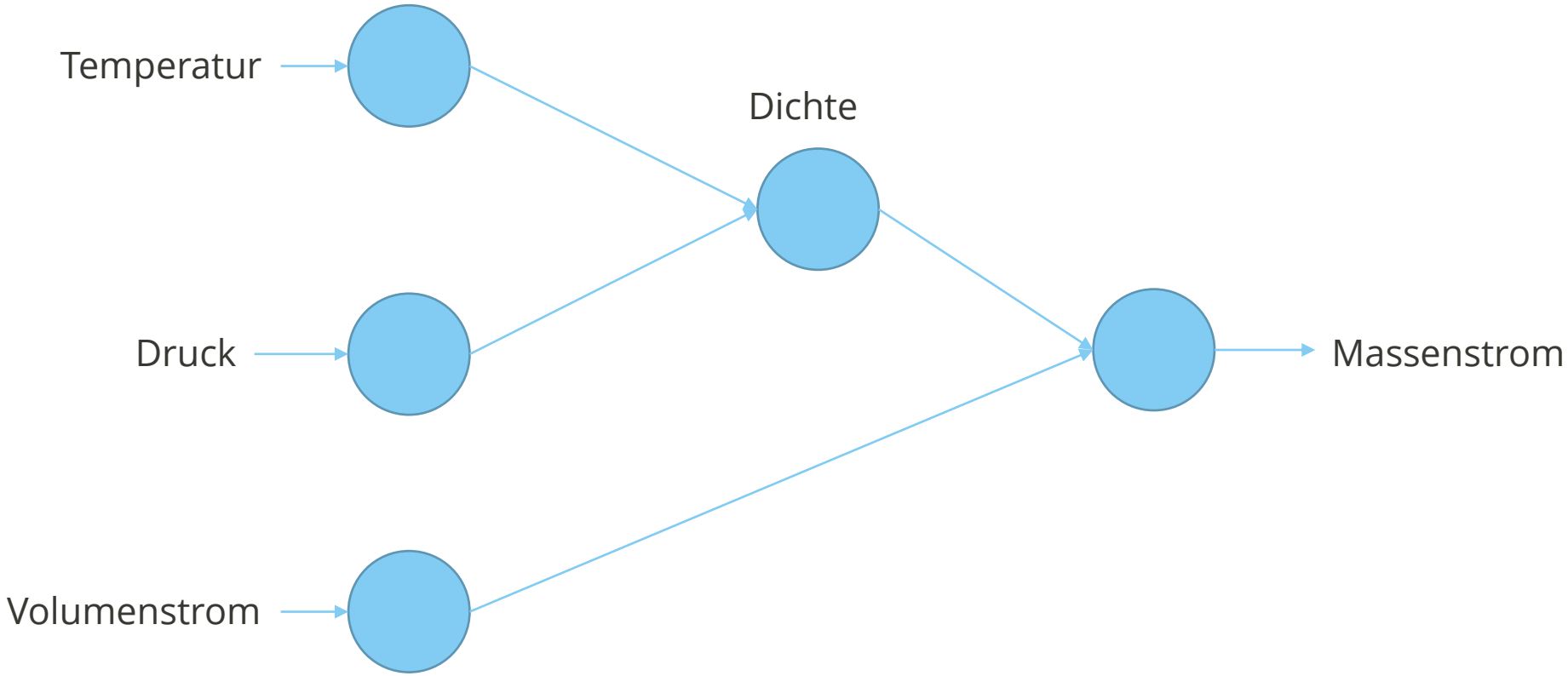
- Mehrfachregression ($m > 1$):

$$h(x) = b_0 + b_1 x_1 = b_0 x_0 + b_1 x_1 = \sum_{j=0}^{m} b_j x_j \text{ mit } x_0 = 1$$

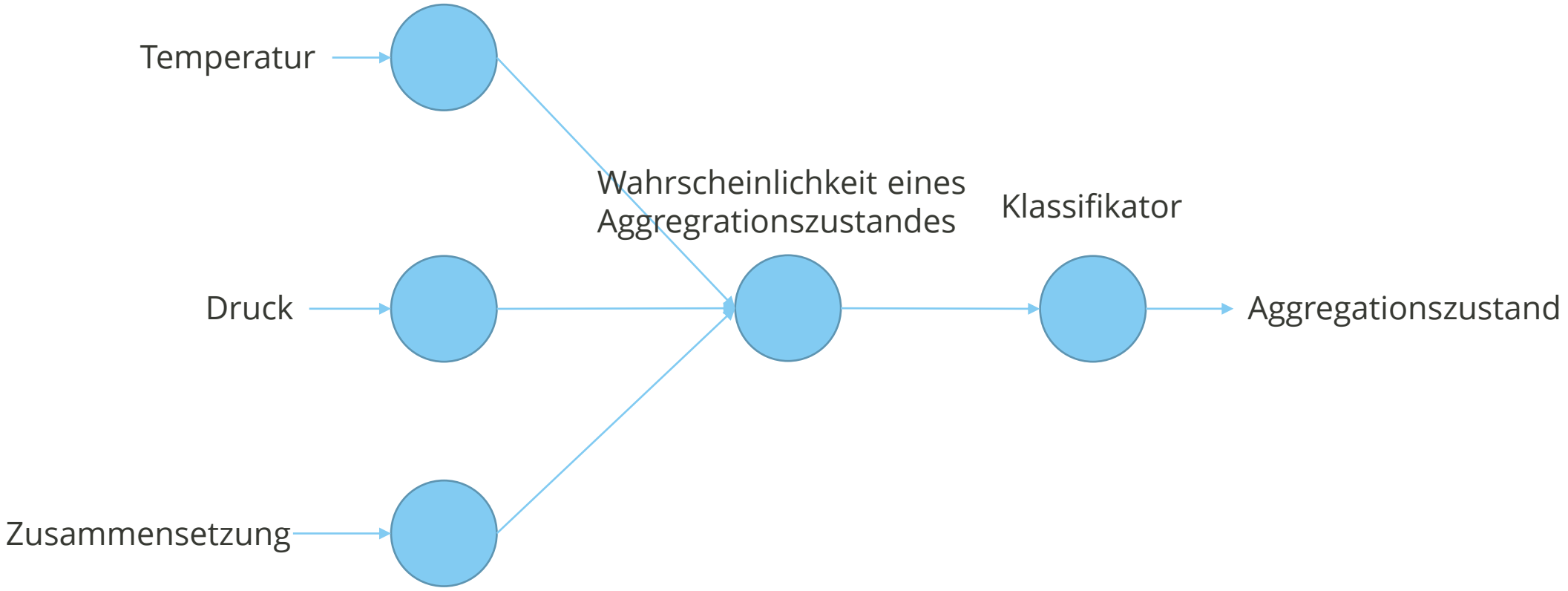
$$\boxed{h(x) = \underline{x}^T \underline{b}}$$

Motivation und Grundlagen

Beispiel: Regressionsaufgabe



Beispiel: Klassifikation



Anwendungen

Inputs:

- Prozesswerte (Temperatur, Druck, Konzentration, Ventilstellung, etc.)
- Ereignisse/Zustände (Ausschalten des Wärmetauschers, Flooding in der Kolonne, etc.)
- Design-Parameter (DN der Kolonne, Verschaltung von Modulen)
- Bild und Videodaten

Outputs:

- Im Prinzip dieselbe wie Inputs möglich

Anwendungen

Regression:

- Schätzung einer oder mehrerer Prozessvariable(n)
- Vorhersage von Prozessvariablen
- Erkennung von definierten Ereignissen

Klassifikation:

- Schätzung von diskreten Variablen
- Erkennung von Prozessphasen/schritten

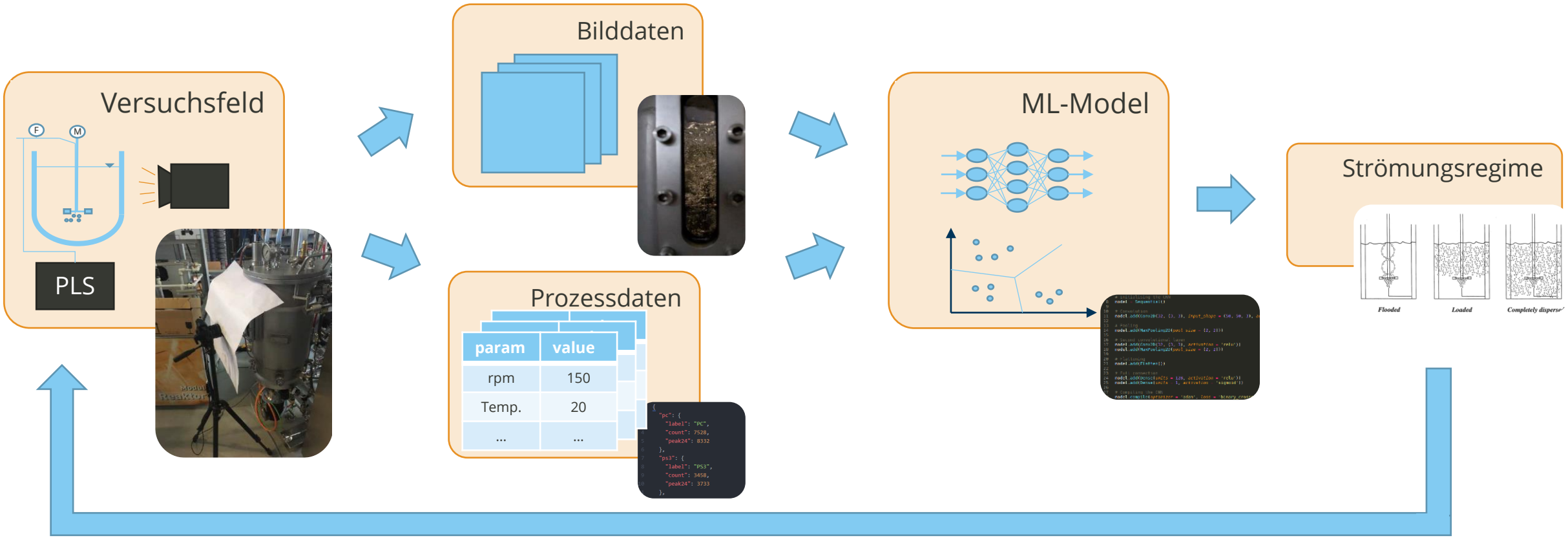
Vorteile

Wann:

- wenn der manuelle Feature Engineering sich nicht lohnt
- wenn einfachere Modelle Zusammenhänge nicht abbilden können (starke nichtlinearität)
- wenn das Modell rechengünstig sein soll (vgl. First-Principle Modelle mit ODE/PDEs etc.)
- wenn die Simulation stabil sein soll (vgl. Flowsheet-Simulationen)

Komplexes Beispiel:

Multimodale datengetriebene Klassifikation des Strömungsregimes in einem Bioreaktor



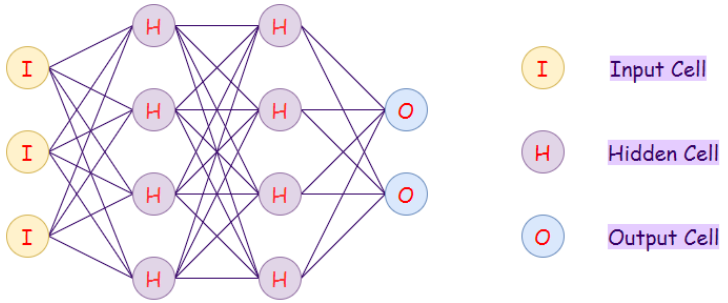
M. Esser, 2022, Forschungspraktikum, TUD

Grundarten

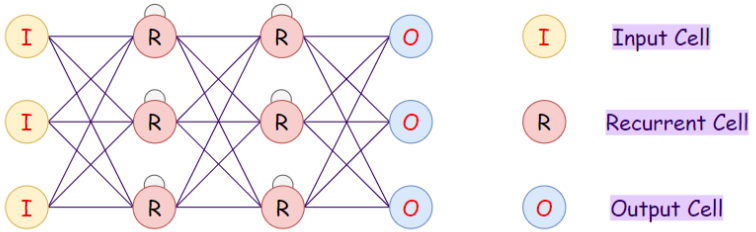
1. Feed-forward Neural Network (NN)
2. Recurrent Neural Network (RNN) inkl. Long Short Term Memory (LTSM) and Gated Recurrent Unit (GRU)
3. Auto Encoder (AE)
4. Convolutional Neural Network (CNN)
5. Generative Adversarial Network (GAN)
6. Transformers

Grundarten

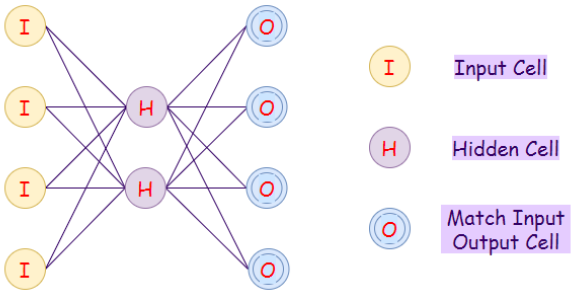
Deep Feed Forward (DFF)



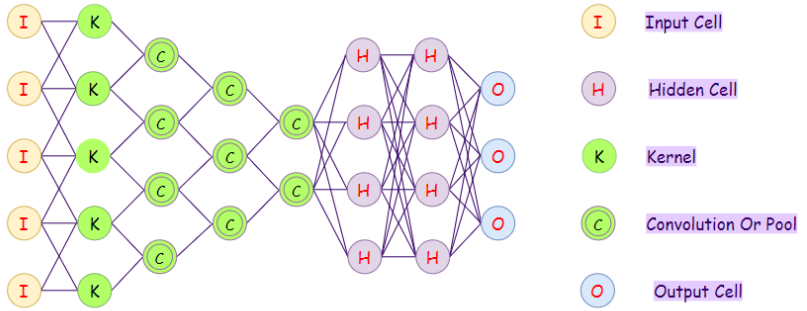
Recurrent Neural Network (RNN)



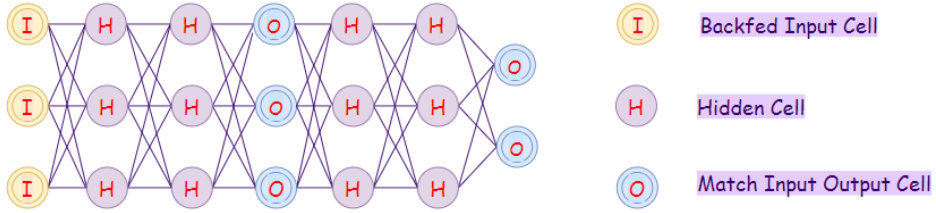
Auto Encoder (AE)



Deep Convolutional Network (DCN)

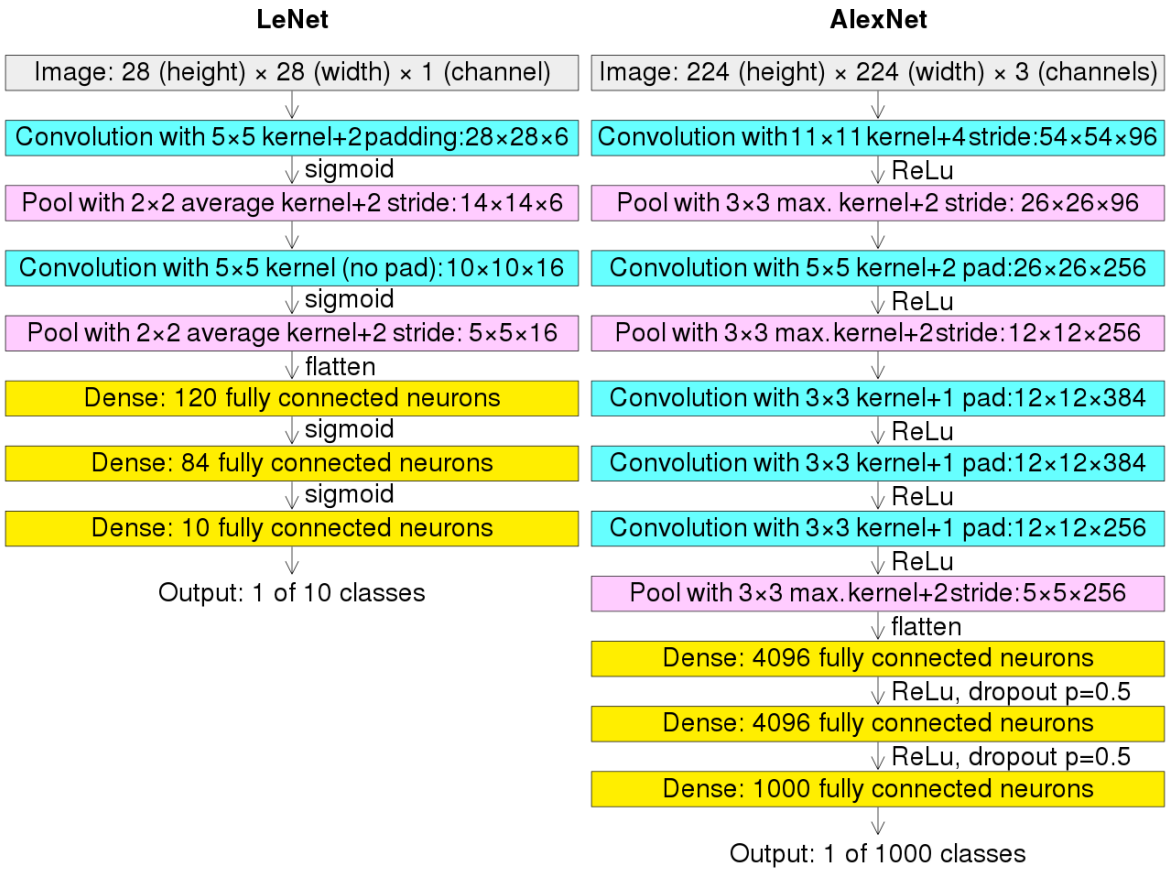


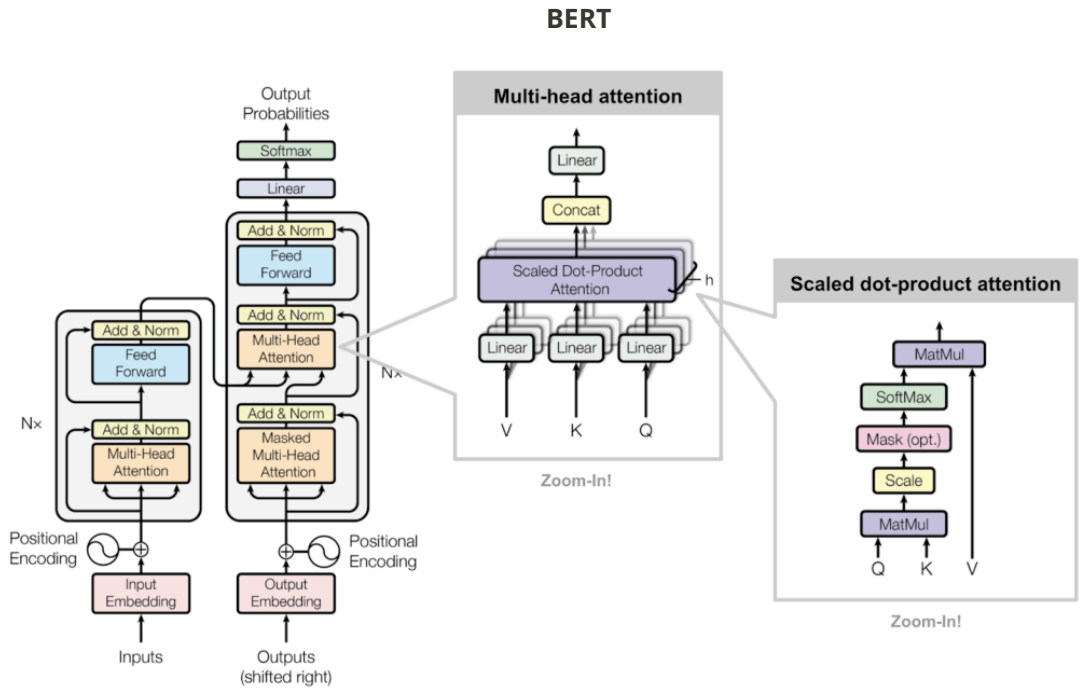
Generative Adversarial Network (GAN)



https://towardsai.net/p/machine-learning/main-types-of-neural-networks-and-its-applications-tutorial-734480d7ec8e

Moderne Architekturen



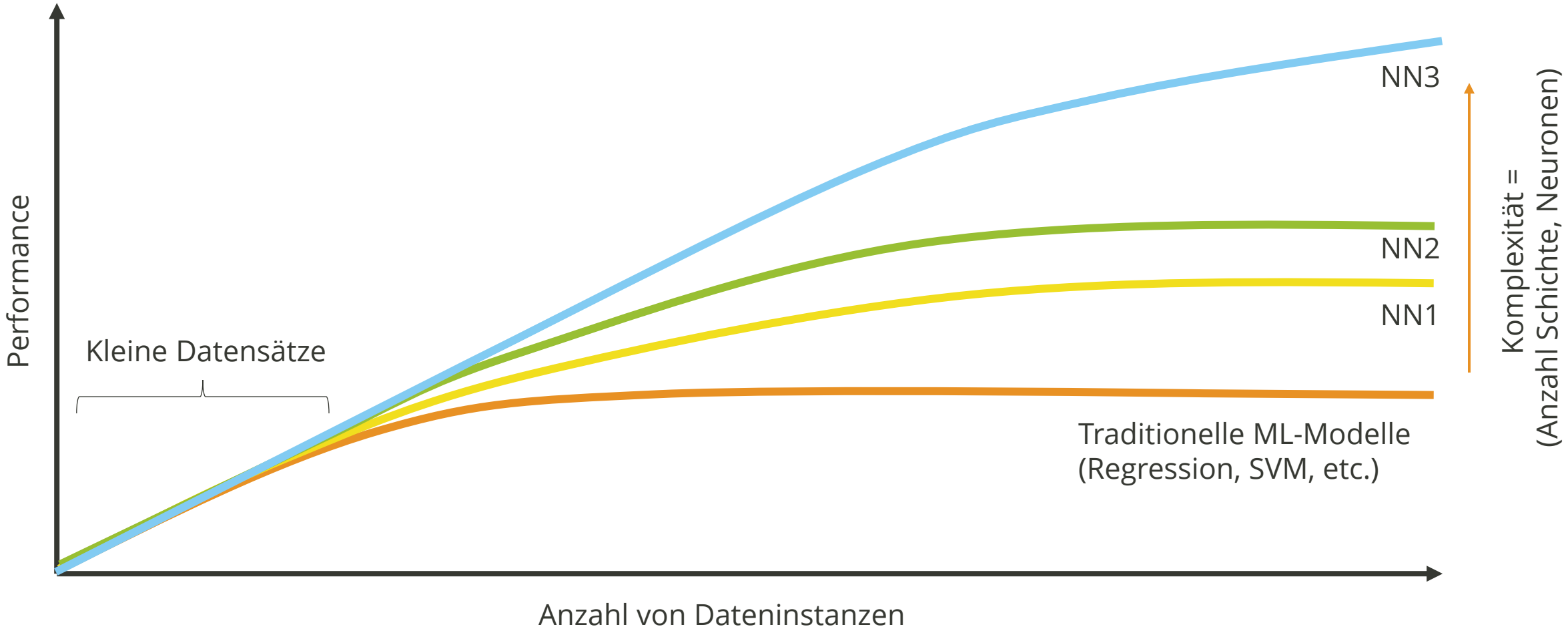


→ Komplexität und Anzahl von freien Parametern steigt →

https://en.wikipedia.org/wiki/AlexNet#/media/File:Comparison_image_neural_networks.svg

https://neptune.ai/blog/bert-and-the-transformer-architecture

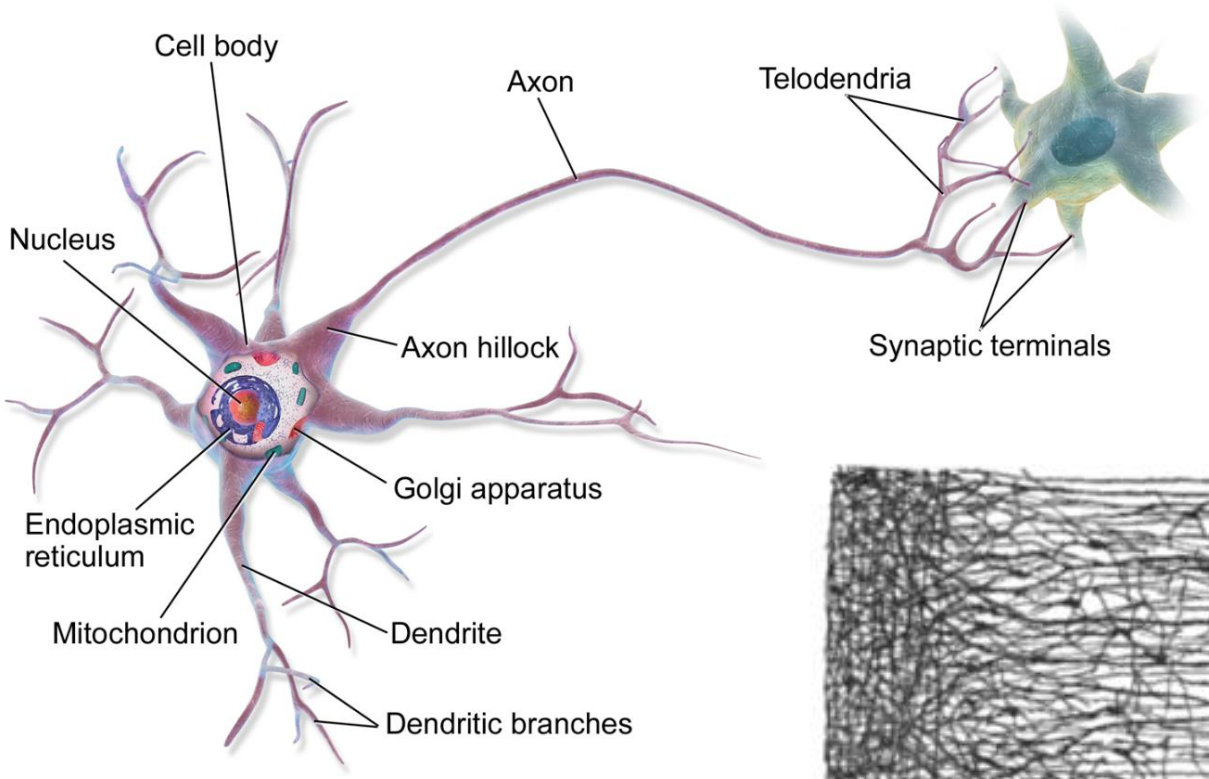
Deep Neuronale Netze

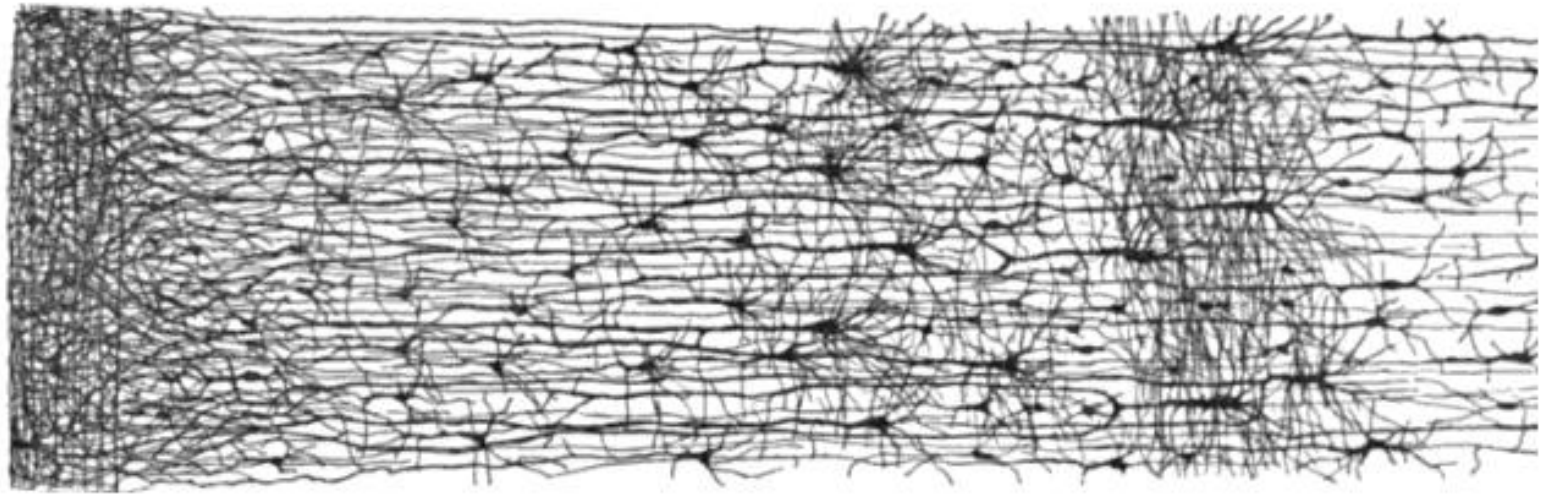


Basiert auf https://community.deeplearning.ai/t/dls-course-1-lecture-notes/11862

Feed-Forward Neural Network

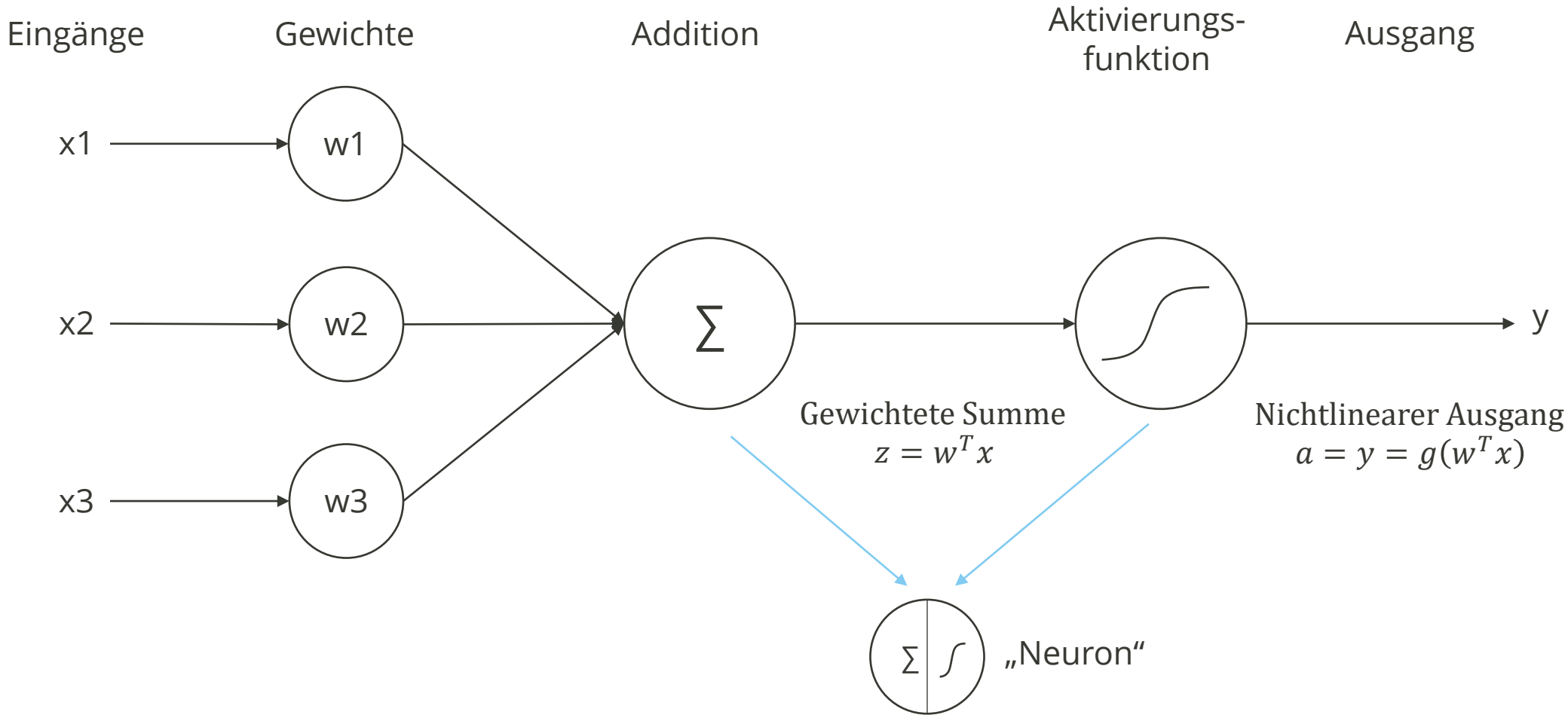
Neuronale Netze: Grundidee





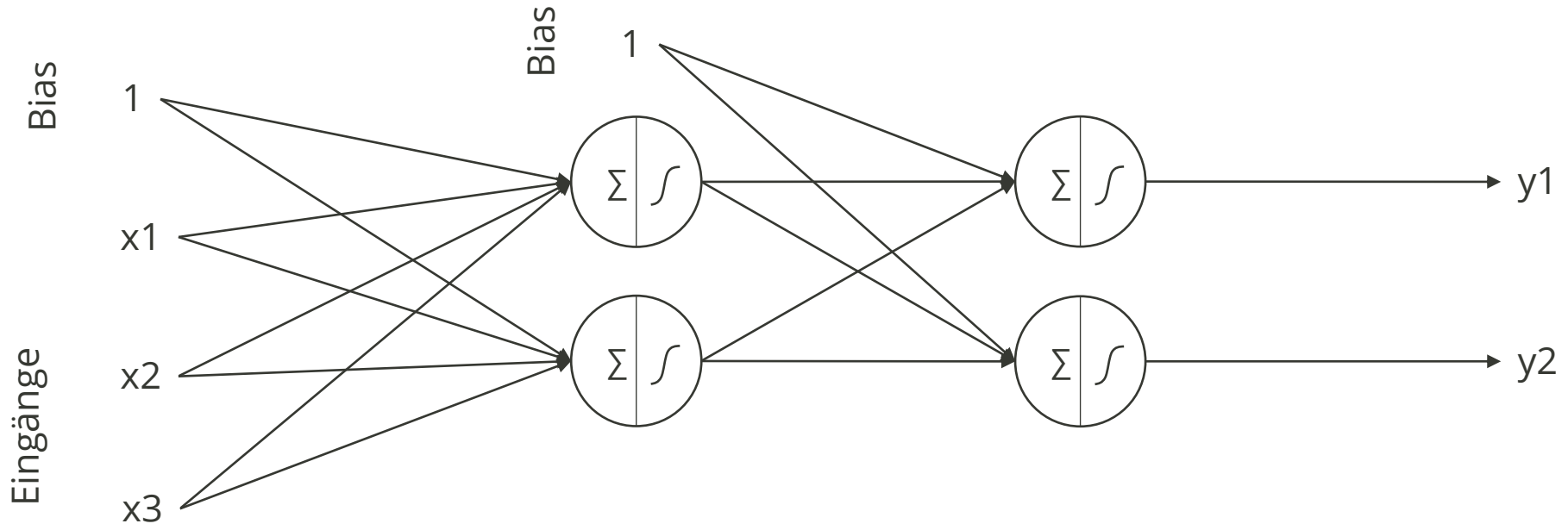
A. Géron, 2019, Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media

Perceptron

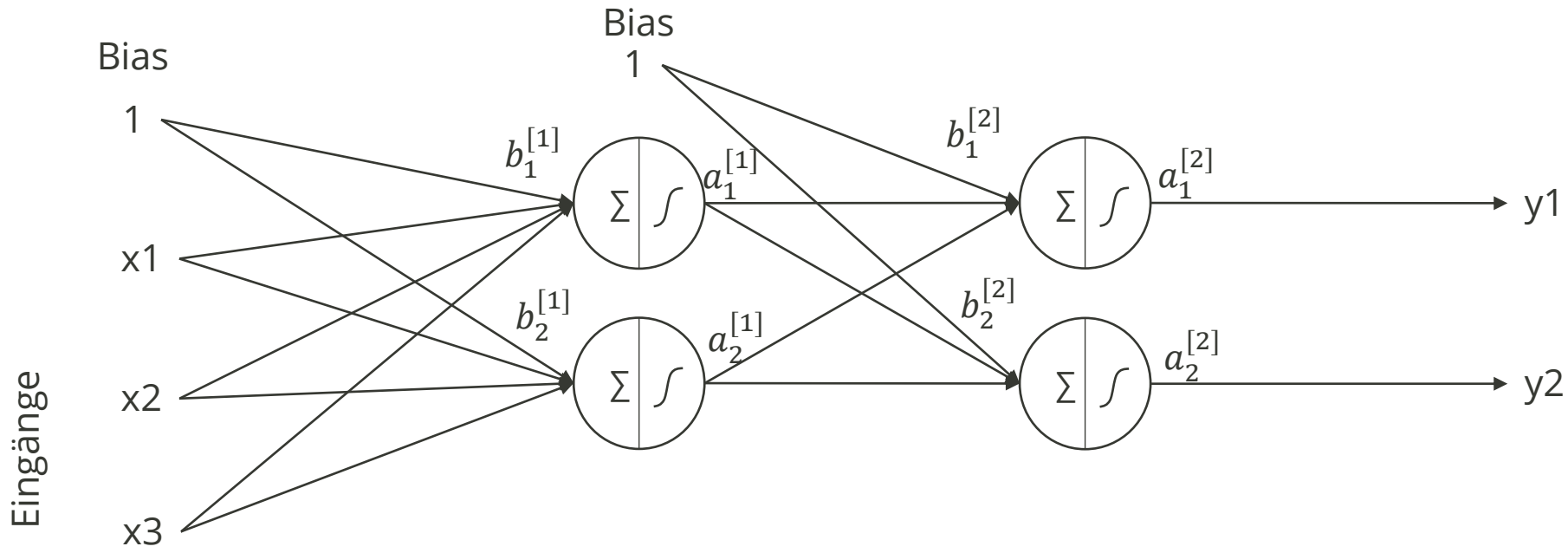

Eingänge
Gewichte
Addition
Aktivierungs-
funktion
Ausgang
x1
x2
x3
w1
w2
w3
∑
y
Gewichtete Summe
$z = w^T x$
Nichtlinearer Ausgang
$a = y = g(w^T x)$
∑
∫
„Neuron“

Neuronales Netz

Dreistufiges NN mit zwei Zwischenschichten, jeweils mit 2 Neuronen



Neuronales Netz



$$a^{[0]} = x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

(3,1)

$$z^{[1]} = w^{[1]T} a^{[0]} + b^{[1]}$$

(2,1) (2,3) (3,1) (2,1)

$$a^{[1]} = g\left(z^{[1]}\right)$$

(2,1) (2,1)

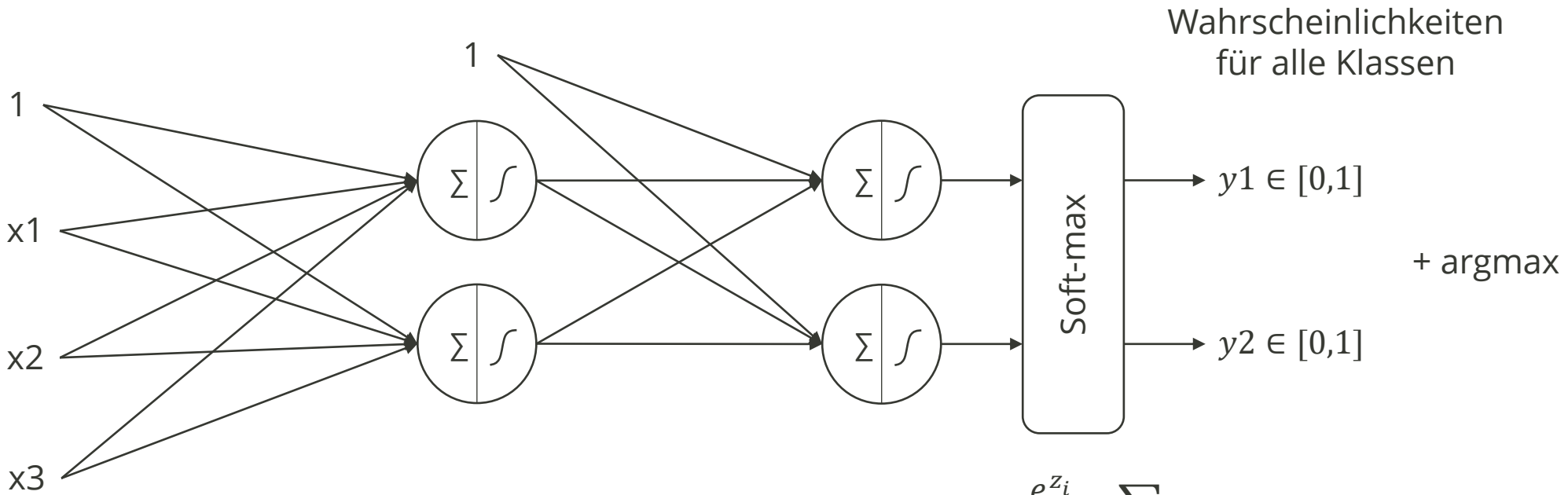
$$z^{[2]} = w^{[2]T} a^{[1]} + b^{[2]}$$

(2,1) (2,2) (2,1) (2,1)

$$a^{[2]} = g\left(z^{[2]}\right) = y$$

(2,1) (2,1)

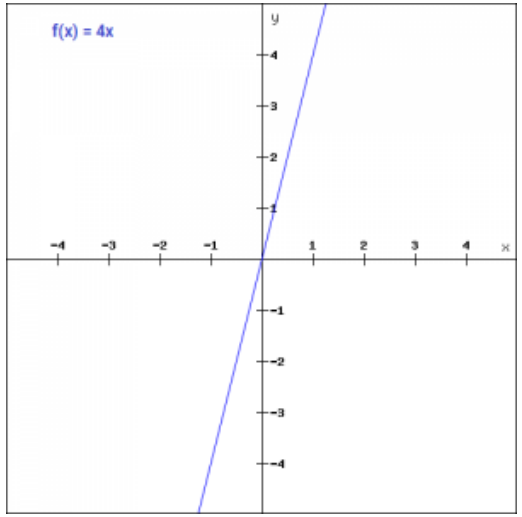
Neuronales Netz für Klassifikationsaufgaben



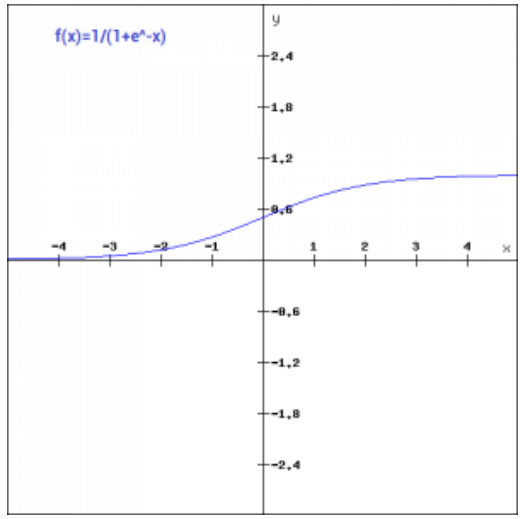
Aktivierungsfunktionen

Die Aktivierungsfunktion ermöglicht **Nichtlinearität** in das Modell. Mehrschichtiges Modell mit einer beliebigen Anzahl von Schichten und linearen Einheiten ergibt trotzdem lediglich ein lineares Modell.

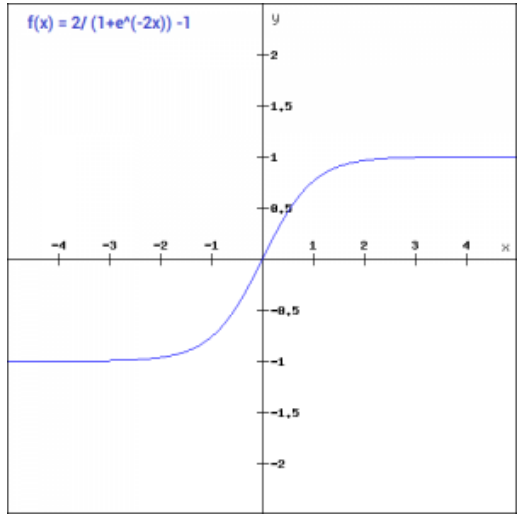
Linear
(vgl. Lineare Mehrfachregression)
$g(z) = az$



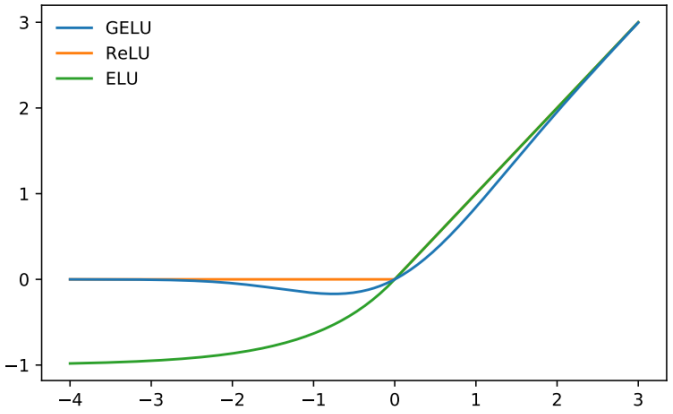
Sigmoid
$g(z) = \frac{1}{1+e^{-z}}$



Tanh
$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$



RELU (Rectified Linear Unit)
ELU (Exponential Linear Unit)
Leaky RELU
SELU (Scaled ELU)
GELU (Gaussian Error LU)



https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/
https://medium.com/syncedreview/gaussian-error-linear-unit-activates-neural-networks-beyond-relu-121d1938a1f7

Lernen in Neuronalen Netzen

Lernen in Neuronalen Netzen

- Lernfähigkeit aus Trainingsbeispielen
- prinzipielle Ablauf eines überwachten Lernverfahrens:
  1. Anfangsbelegung des Gewichtsvektors $\underline{w}$, der sämtliche im Netz vorhandenen Gewichte enthält, z.B. mit Zufallszahlen
  2. Vorgabe eines Lerndatensatzes (Eingabevektor $\underline{x}$, Zielvektor $\underline{y}$); entsprechende Aktivierung der Neuronen der Eingabeschicht mit $\underline{x}$
  3. Vorwärtspropagierung der angelegten Eingabewerte durch das Netz führt zu einem Ausgabevektor $\underline{o}$
  4. Vergleich der Netzausgabe $\underline{o}$ mit dem Zielvektor $\underline{y}$ liefert den Fehlervektor $\underline{\Delta}$
  5. Änderung der Verbindungsgewichte durch Rückwärtspropagierung des Fehlers von der Ausgabe- zur Eingabeschicht mit dem Ziel der Minimierung einer Fehlerfunktion E ($\underline{\Delta}$, $\underline{w}$)
  6. Fortsetzung des Verfahrens mit einem neuen Lerndatensatz beim zweiten Anstrich

$\Rightarrow$ Fehlerfunktion

$\Rightarrow$ Veränderung der Gewichte

Lernen in Neuronalen Netzen

Veränderung der Gewichte

$$w_{ij}^{neu} = w_{ij}^{alt} + \Delta w_{ij}$$

- Bestimmung Δw_{ij}:
  z.B. mittels Gradientenverfahren bzgl. der Zielfunktion E: $\Delta w_{ij} = -\eta \frac{\partial}{\partial w_{ij}} E(\underline{W})$ (η: Lernfaktor)

Herleitung einer einfachen Lernregel

- für *einstufige* Netze mit der *Identität* als Aktivierungs- und als Transferfunktion
  - $o_j = a_j = net_j = \sum_i w_{ij} x_i$ (wobei x_i das i-te Eingangsneuron ist)
- off-line-Trainingsverfahren (gesamte Lernstichprobe gehen in einen Lernschritt für die Gewichte ein)
- on-line-Verfahren (Gewichtsänderung erfolgt nach jeder Vorführung eines Datensatzes der Lernstichprobe)

Neuronale Netze
Backpropagation - Algorithmus

Lernfaktor η

- beeinflusst Güte und Geschwindigkeit des Lösungsverfahrens
- bei zu großem η können enge „Täler“ in der Fehlerfläche u.U. nicht gefunden werden
- bei zu kleinem η konvergiert der Algorithmus nur sehr langsam
- empfohlen wird z.B. η = 0,7

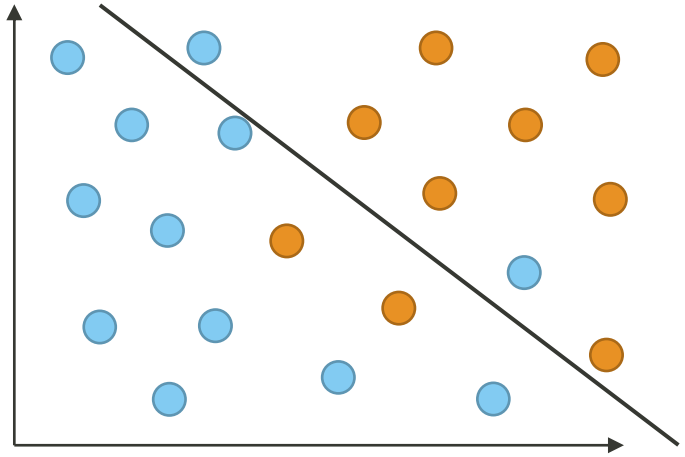
Es gibt Methoden, wo der Lernfaktor auf jedem Schritt adaptiv angepasst wird:

- RMSPro
- Adam
- AdaMax
- etc.

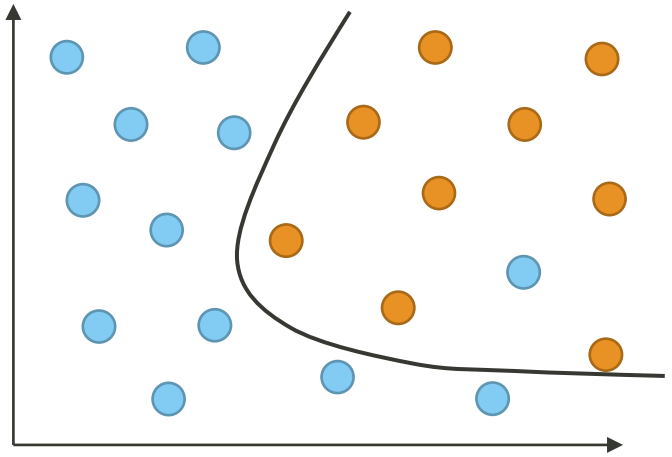
→ **höhere Konvergenzrate**

Regularisierung

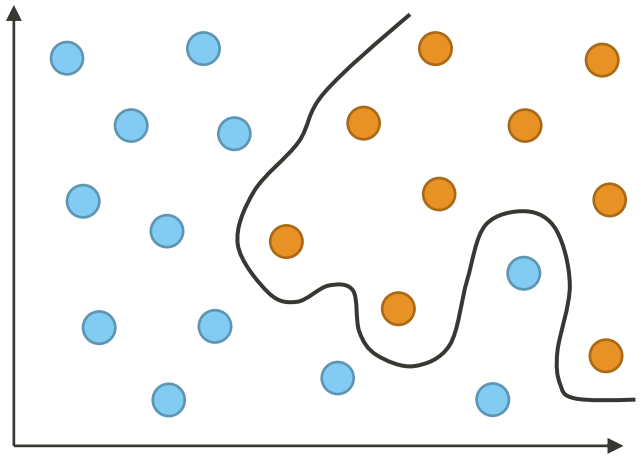
Bias vs Variance



High Bias
Underfitted



Optimal

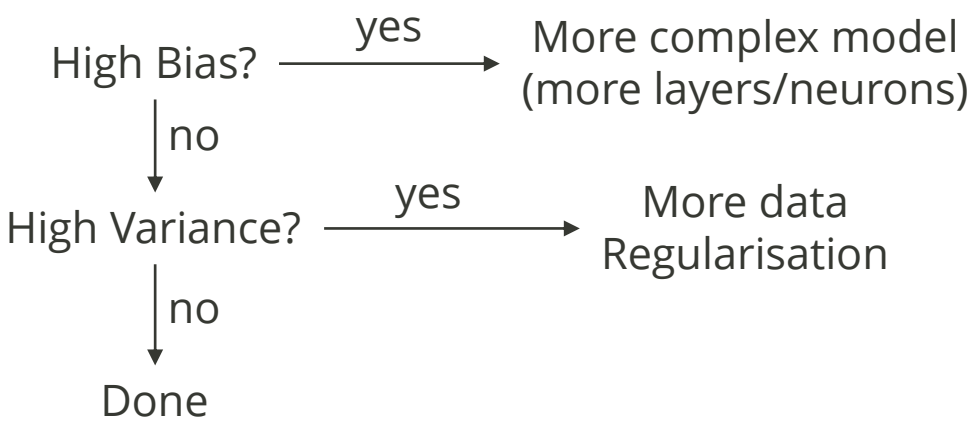


Low Bias
Overfitted

Bias vs Variance

Accuracy-Werte für vier verschiedene Modelle:

Training Datensatz	0,01%	9 %	16 %	0,11%
Validation/Test Datensatz	1 %	10 %	32 %	0,13 %
Bias	Low	High	High	Low
Variance	High	Low	High	Low



Basiert auf https://community.deeplearning.ai/t/dls-course-2-lecture-notes/11866

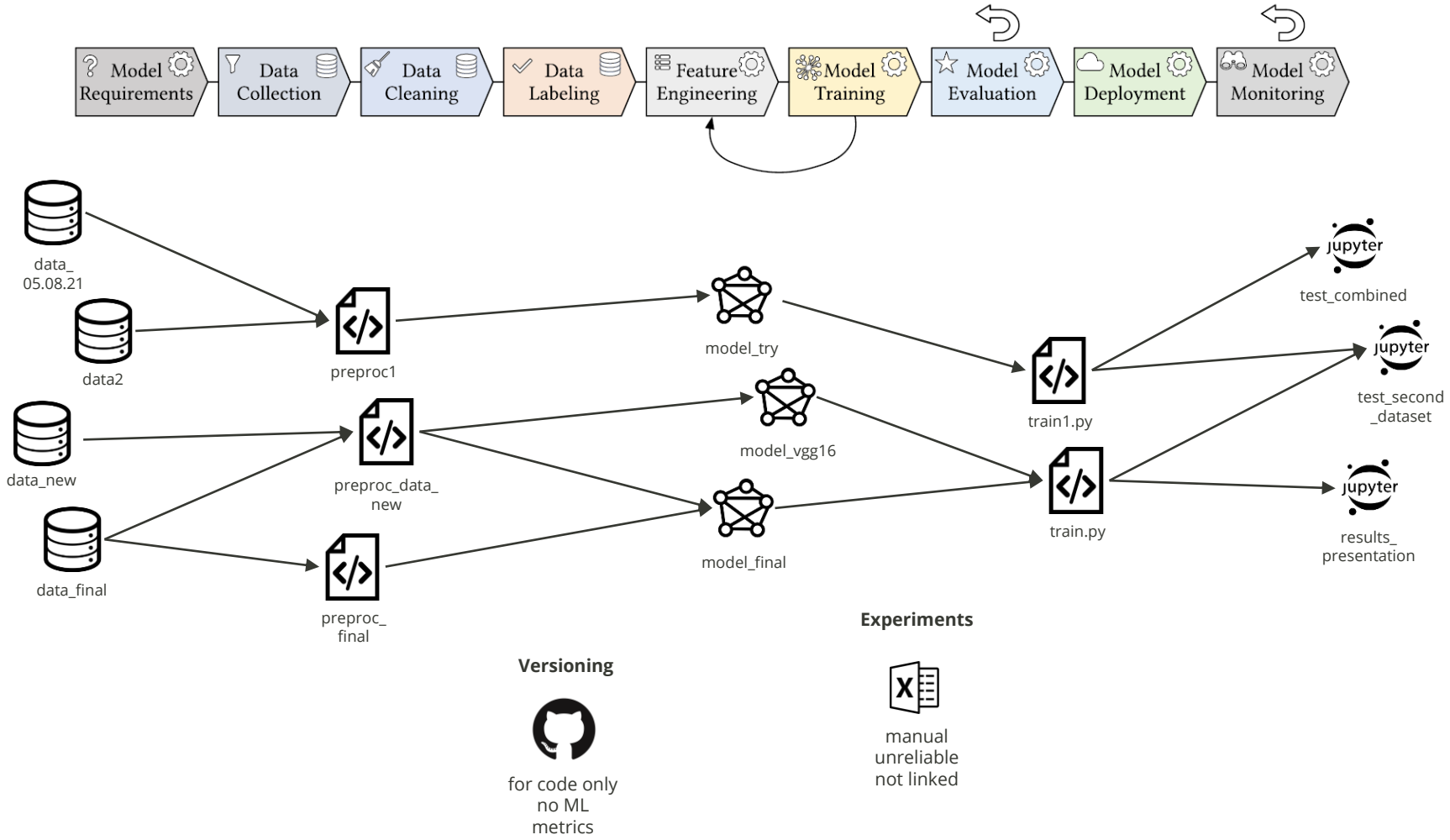
Regularisierung von NN

Methoden:

- Dropout
- Data Augmentation (nicht mit Methode der Datengewinnung verwechseln)
- Early Stopping
- Learning Rate Decay
- Normalisierung von Inputs

Arbeitsweise bei der Modellbildung

Richtig Experimentieren



S. Amershi *et al.*, "Software Engineering for Machine Learning: A Case Study," in *Proceedings - 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice, ICSE-SEIP 2019*, May 2019, pp. 291–300, doi: 10.1109/ICSE-SEIP.2019.00042.

Frameworks

Frameworks

Modellierung, Traineren:

- Tensorflow + Keras, PyTorch
- Scikit-learn (für einfache Aufgaben)

Datenaufbearbeitung, -bereinigung:

- Numpy, Pandas

Bilddaten:

- OpenCV

Visualisierung:

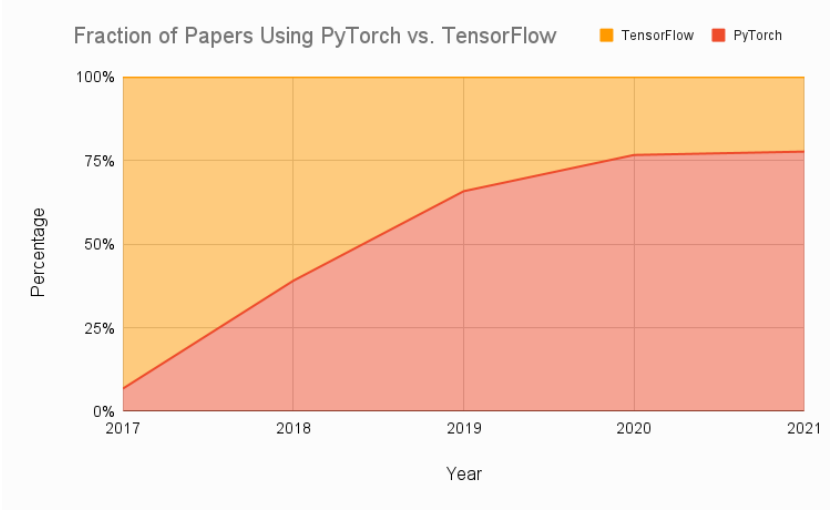
- Matplotlib, Pyplot

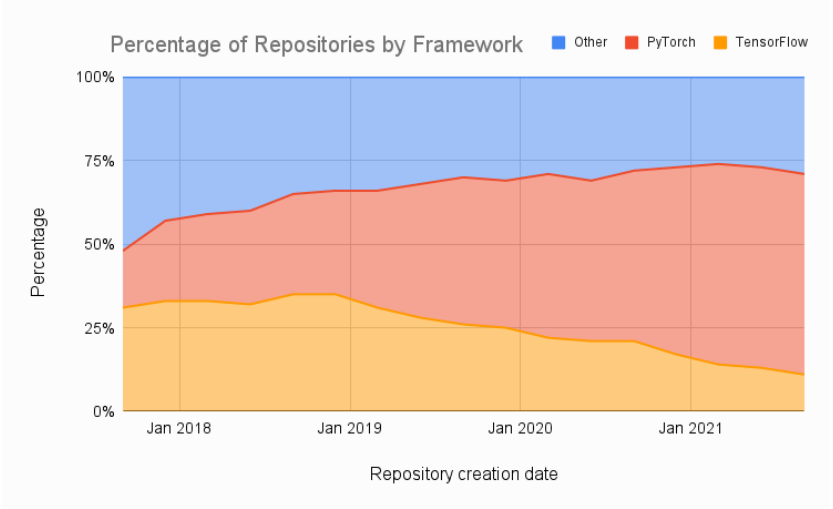
xAI:

- SHAP

MLOps:

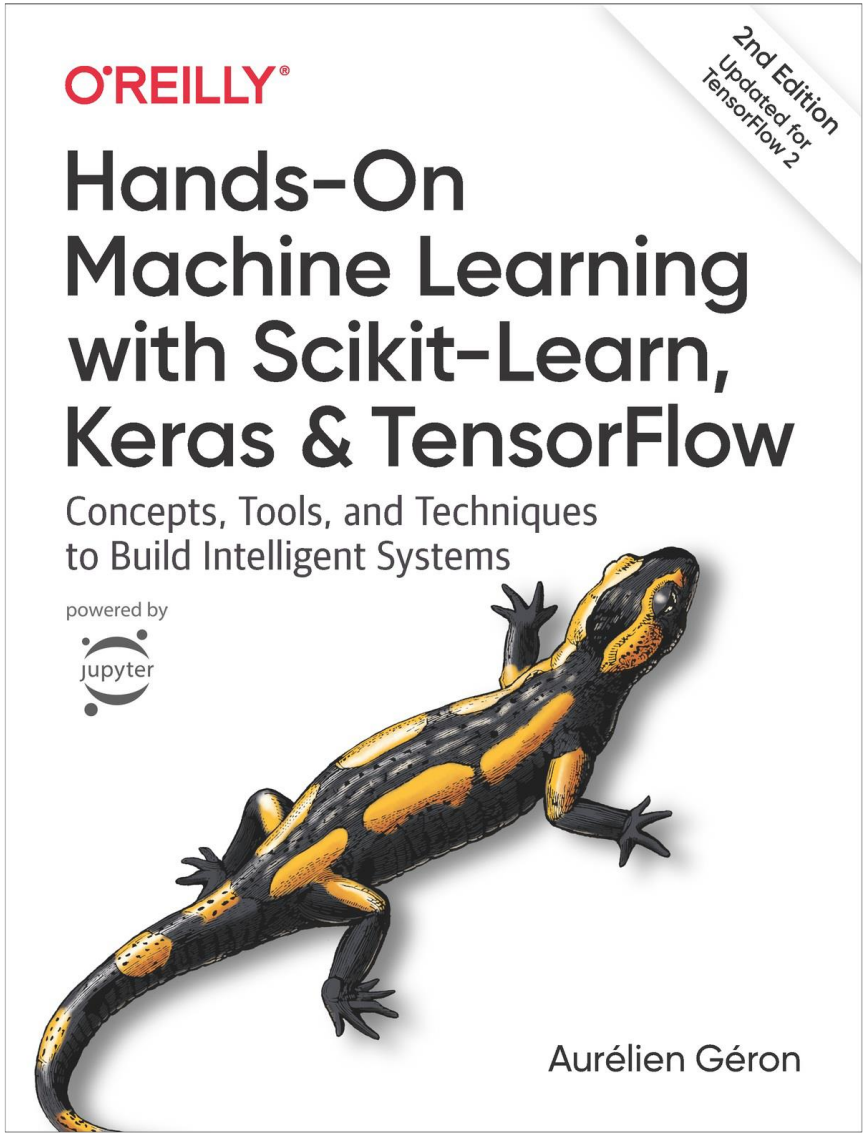
- DVC, mlflow





https://www.assemblyai.com/blog/pytorch-vs-tensorflow-in-2022/

Zusammenfassung der Lehrveranstaltung


O'REILLY®
2nd Edition
Updated for
TensorFlow 2
Hands-On
Machine Learning
with Scikit-Learn,
Keras & TensorFlow
Concepts, Tools, and Techniques
to Build Intelligent Systems
powered by
jupyter
Aurélien Géron

kaggle

PROCESS CONTROL SYSTEMS
PROCESS SYSTEMS ENGINEERING

Dr. rer. nat. Valentin Khaydarov
Email: valentin.khaydarov@tu-dresden.de
Telefon: 0351 463 33387

Vielen Dank für Ihre Aufmerksamkeit!