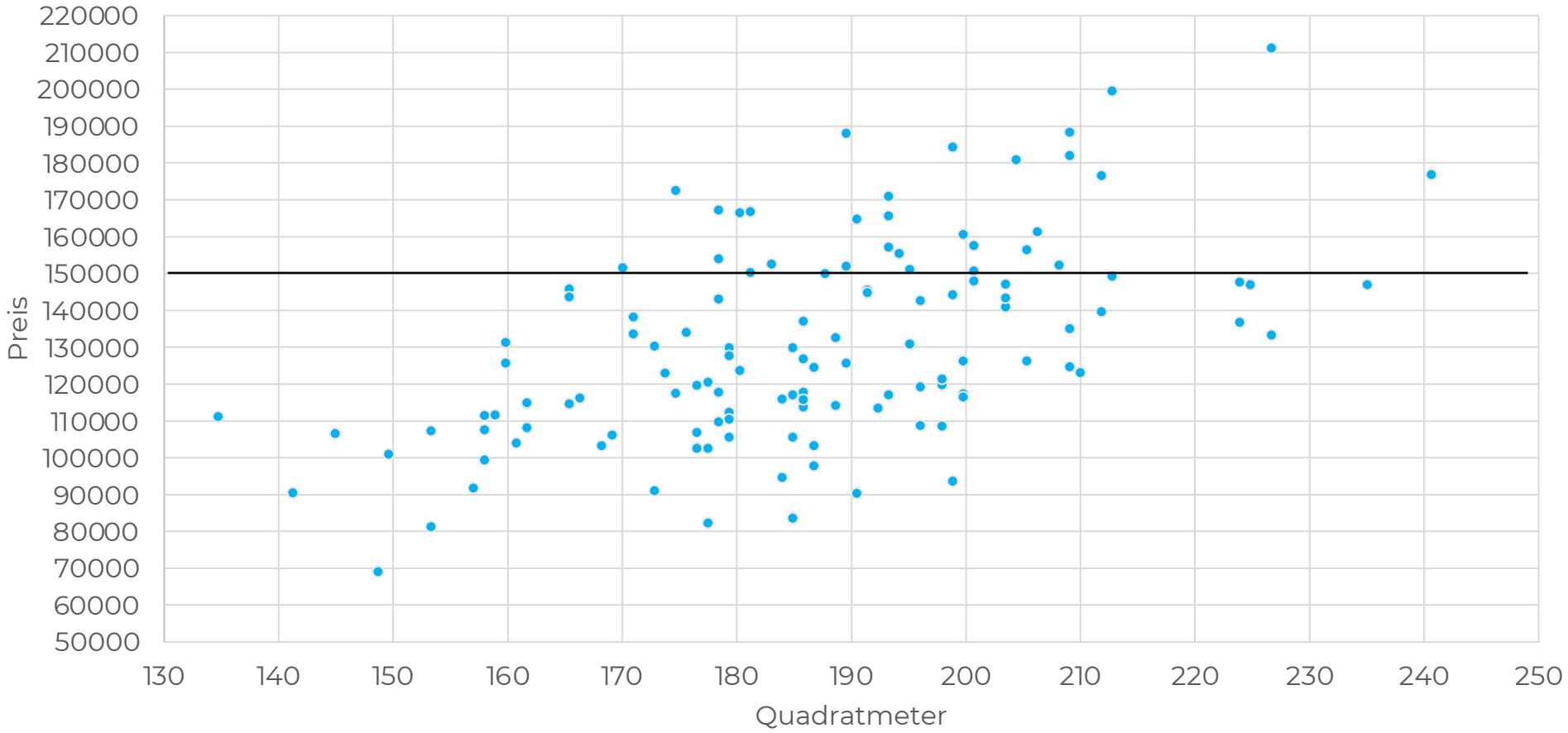
ÜBERBLICK NEURONALE NETZE & EFFIZIENTER EINSATZ

AGENDA

- Rückblick Klassifikation
- Neuronale Netze
  - Beispiel
- Frameworks/Tooling
- Infrastruktur
- Weiterführende Ressourcen

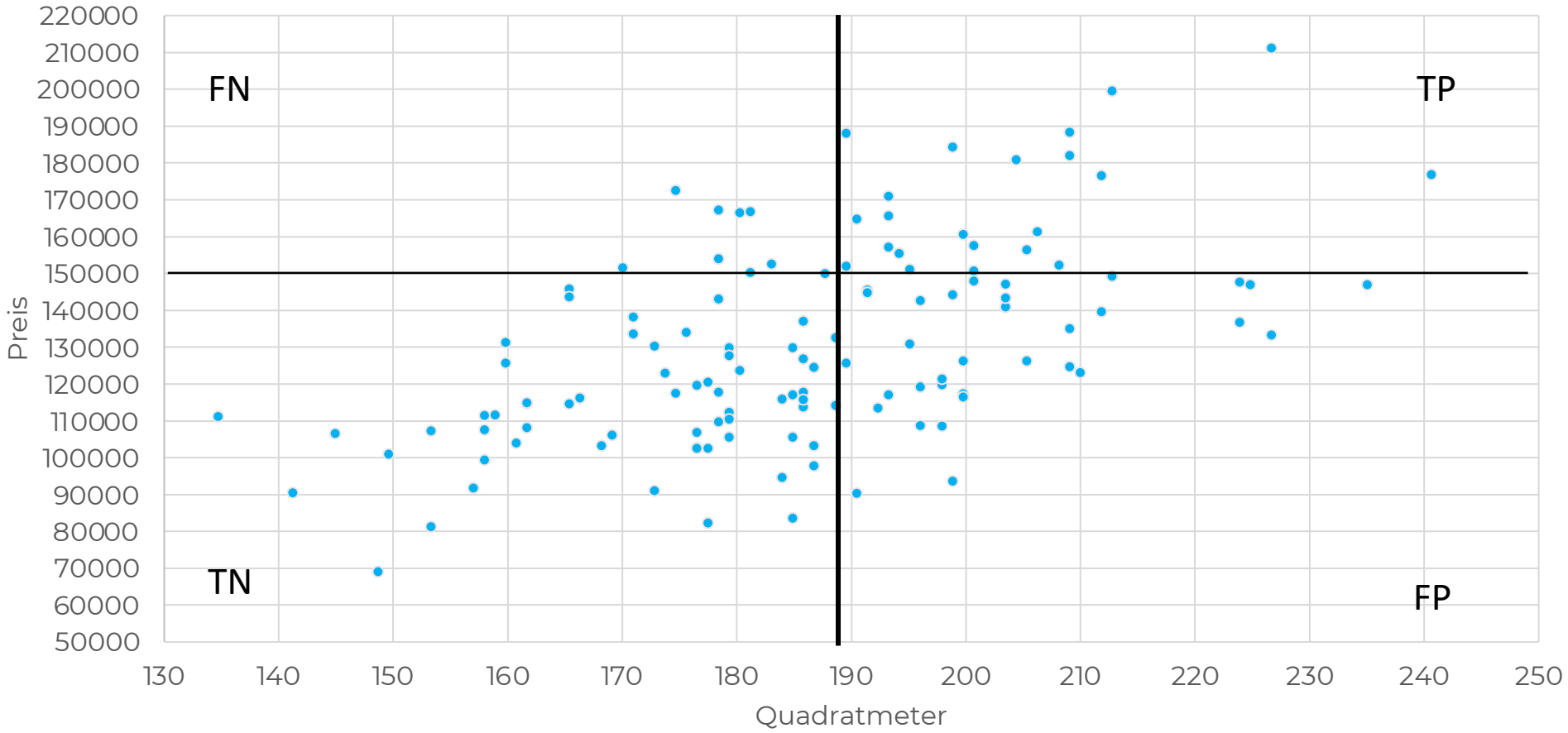
Supervised Learning

KLASSIFIKATION



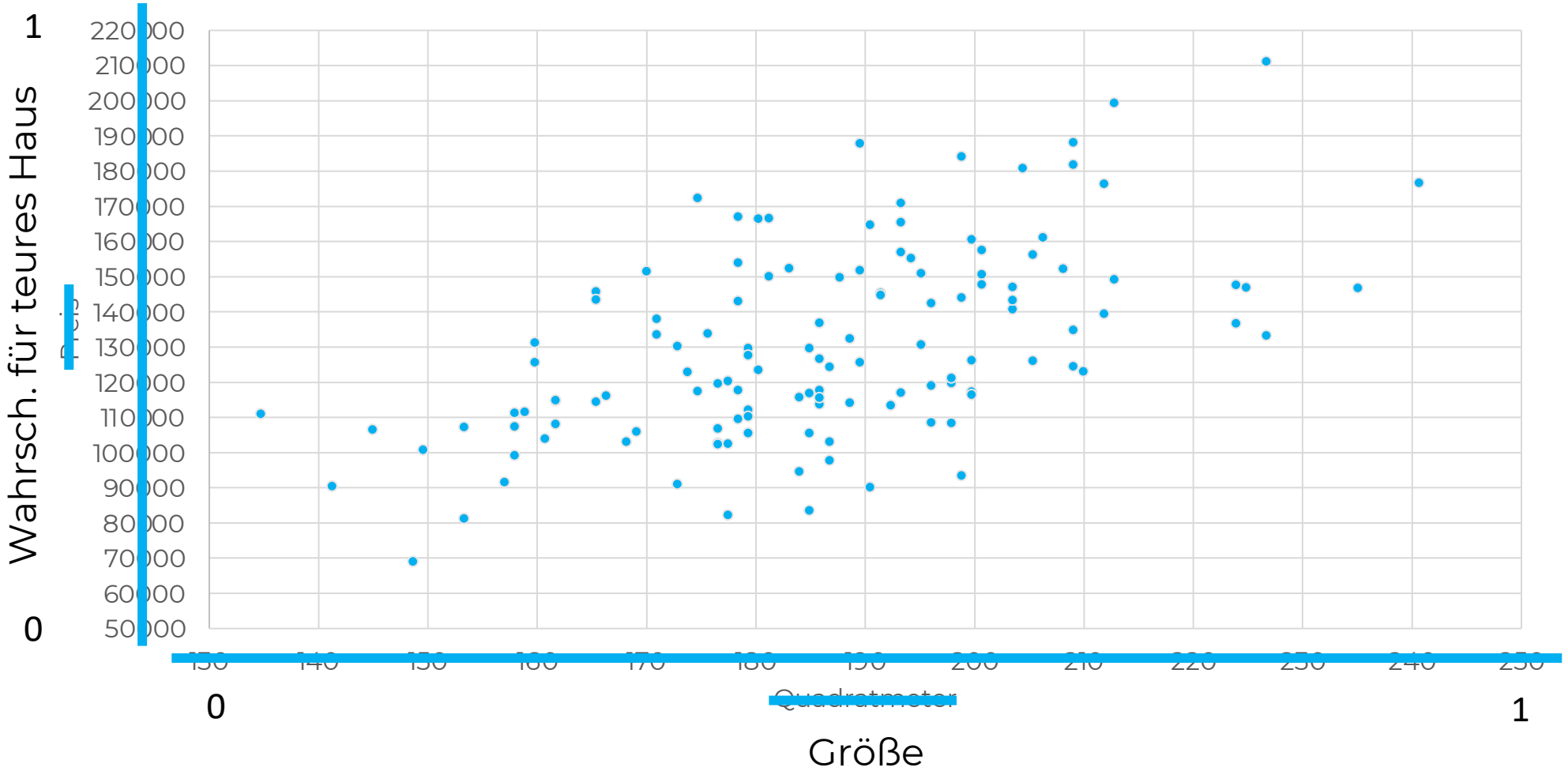
Supervised Learning

KLASSIFIKATION



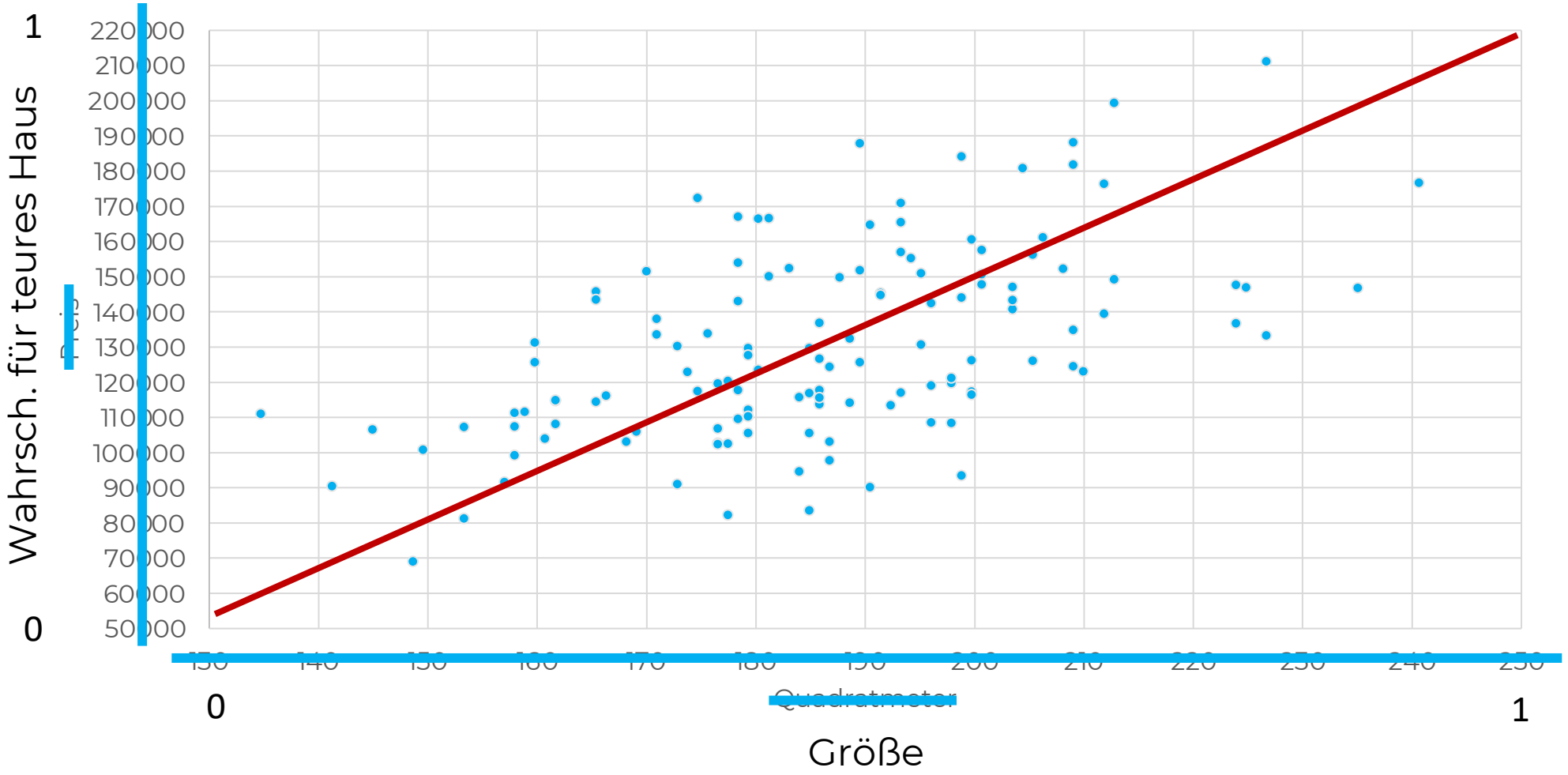
Supervised Learning

KLASSIFIKATION



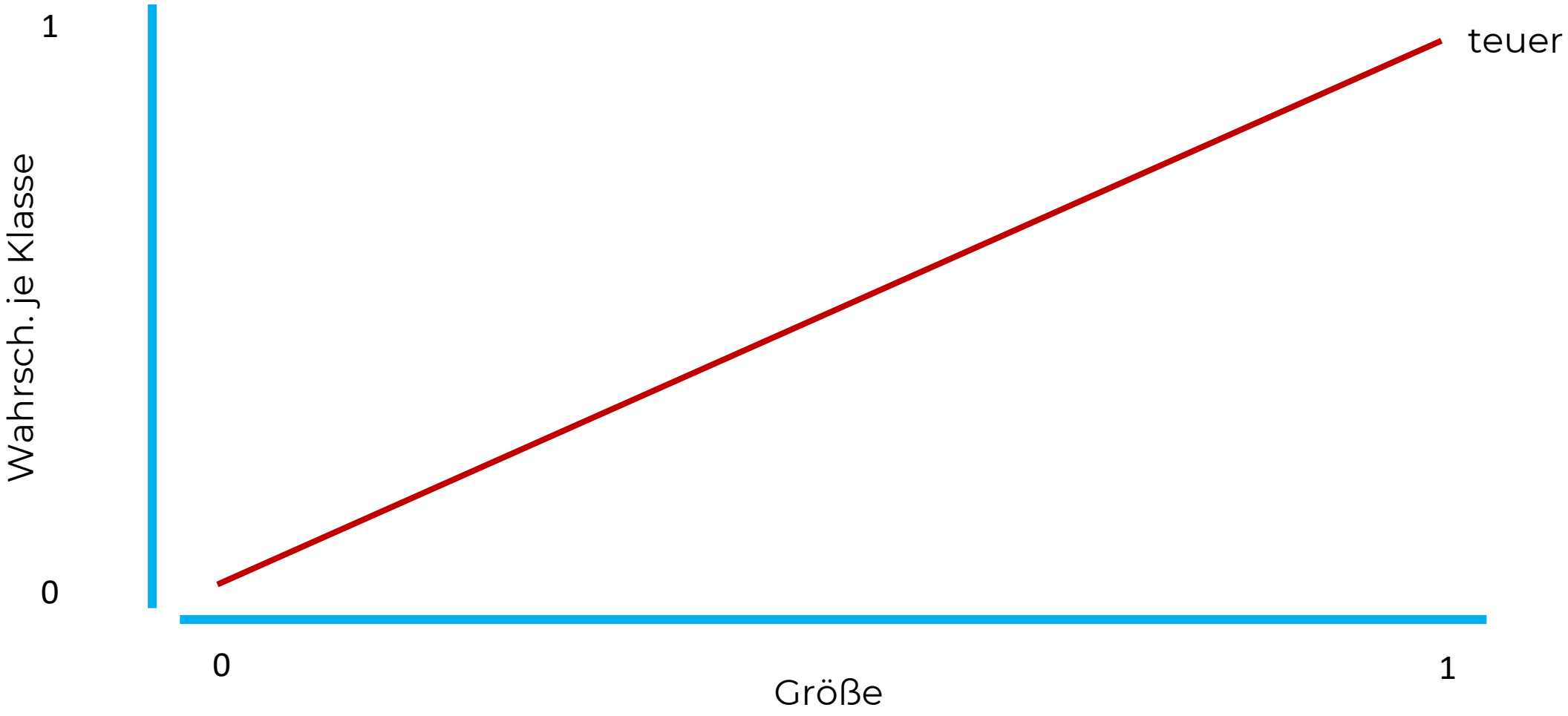
Supervised Learning

KLASSIFIKATION



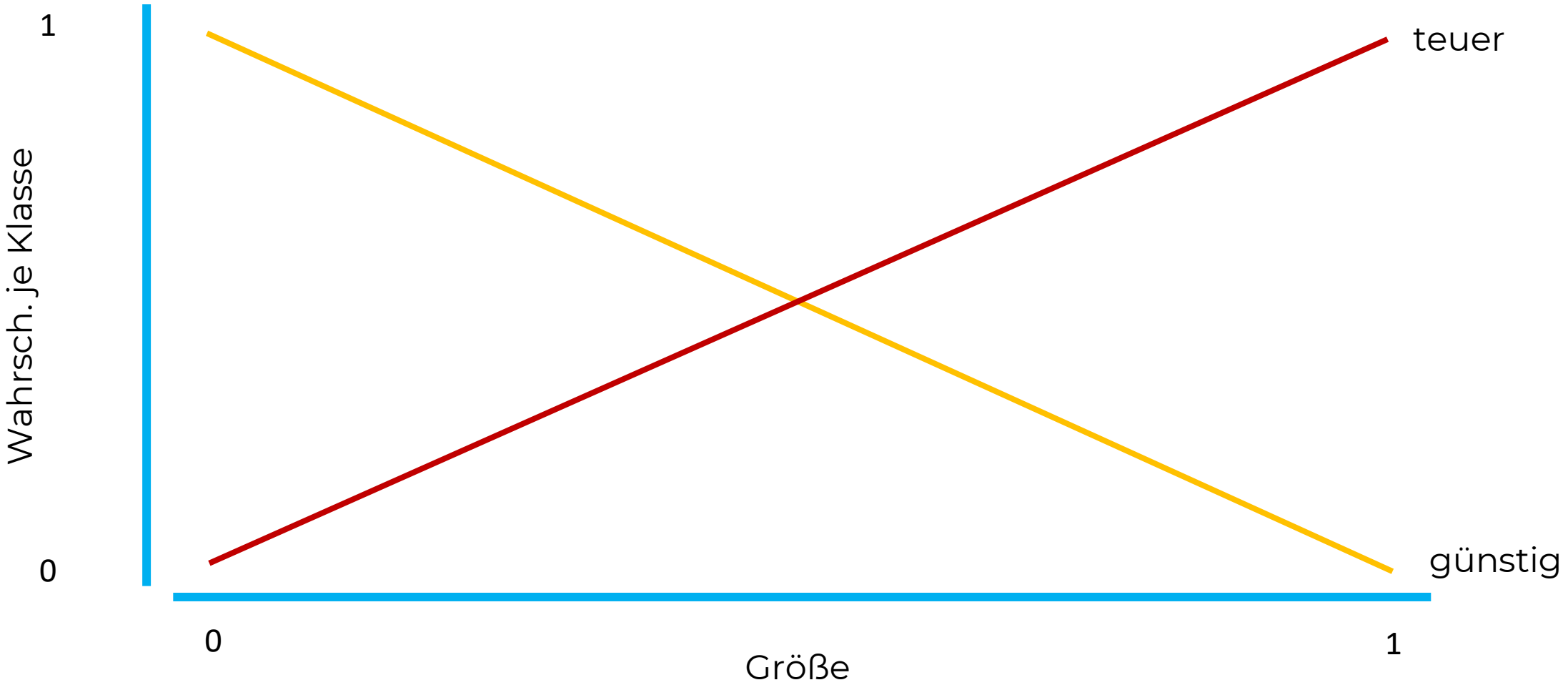
Supervised Learning

KLASSIFIKATION



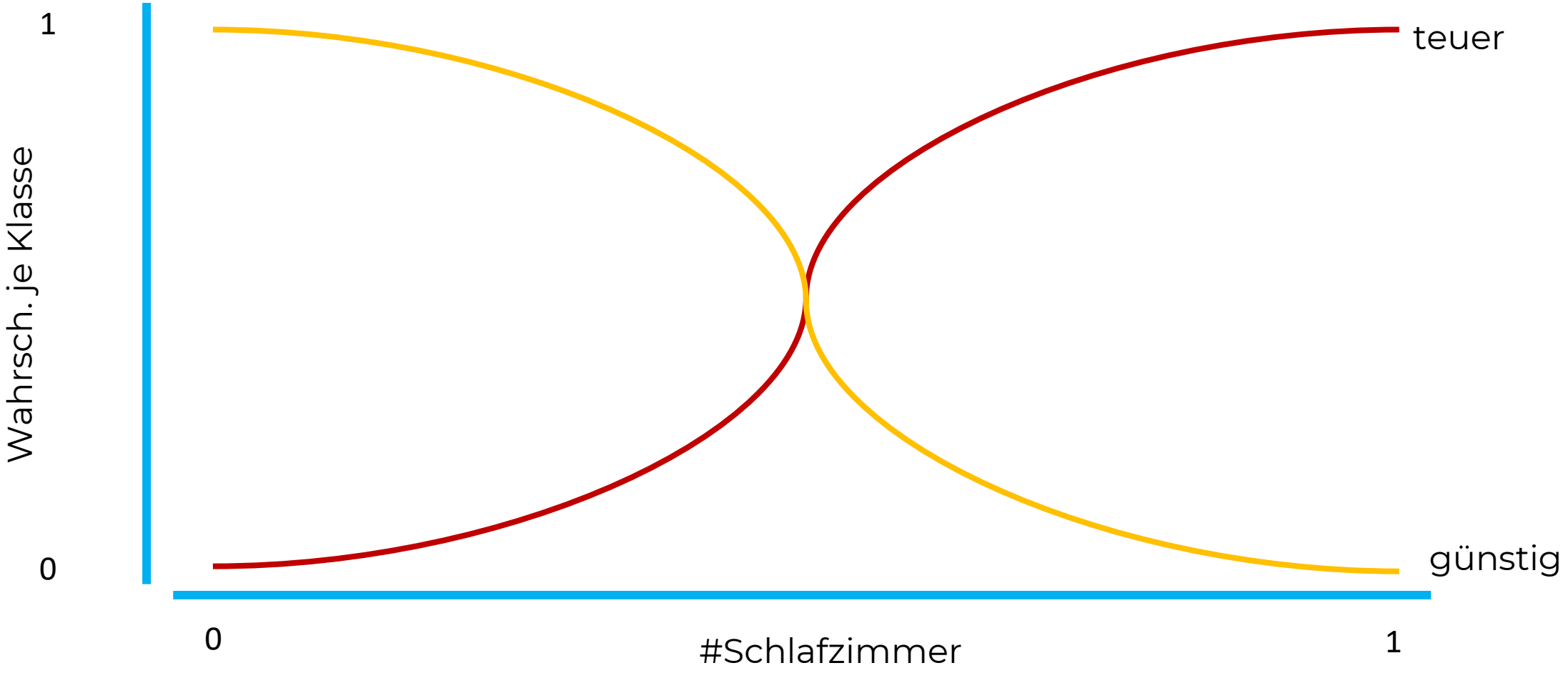
Supervised Learning

KLASSIFIKATION



Supervised Learning

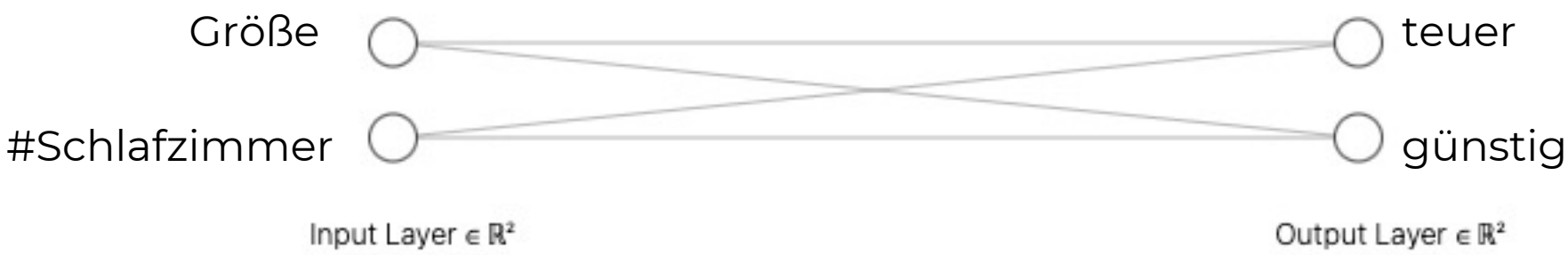
KLASSIFIKATION



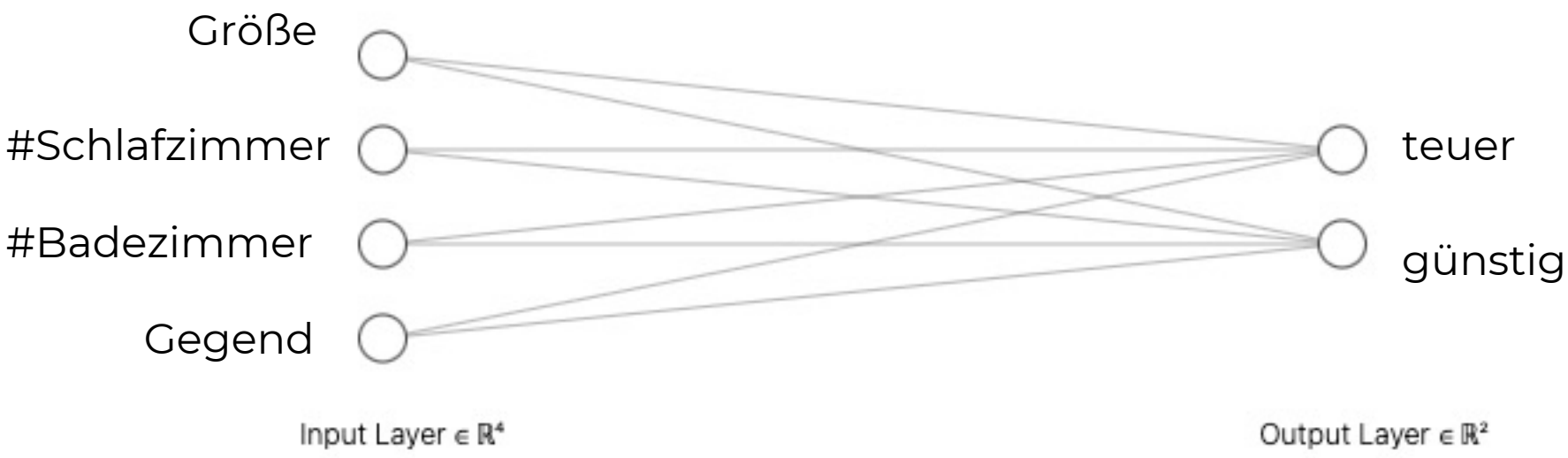
KLASSIFIKATION

$\sum = 1$

KLASSIFIKATION



KLASSIFIKATION



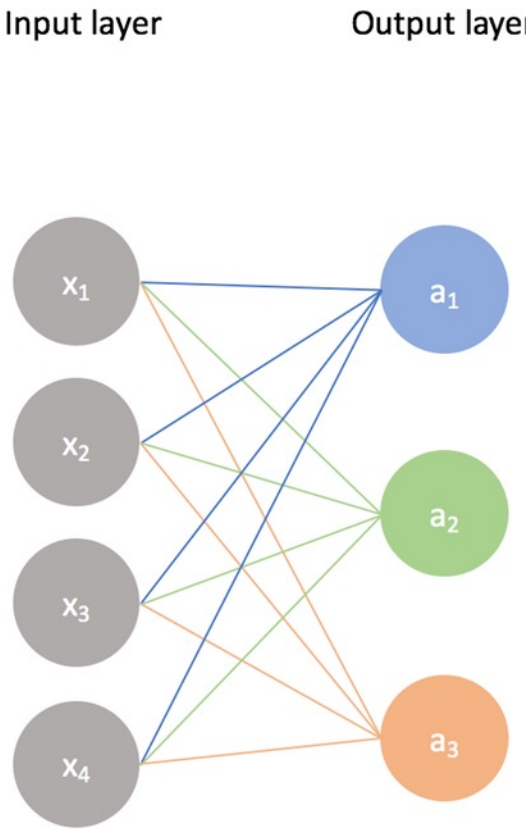
MATRIX-MULTIPLIKATION

$$\mathbf{AB} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \begin{bmatrix} 7 & 10 \\ 8 & 11 \\ 9 & 12 \end{bmatrix}$$

$$= \begin{bmatrix} 1 \times 7 + 2 \times 8 + 3 \times 9 & 1 \times 10 + 2 \times 11 + 3 \times 12 \\ 4 \times 7 + 5 \times 8 + 6 \times 9 & 4 \times 10 + 5 \times 11 + 6 \times 12 \end{bmatrix}$$

$$= \begin{bmatrix} 50 & 68 \\ 122 & 167 \end{bmatrix} \qquad (2.8$$

NEURONALES NETZ

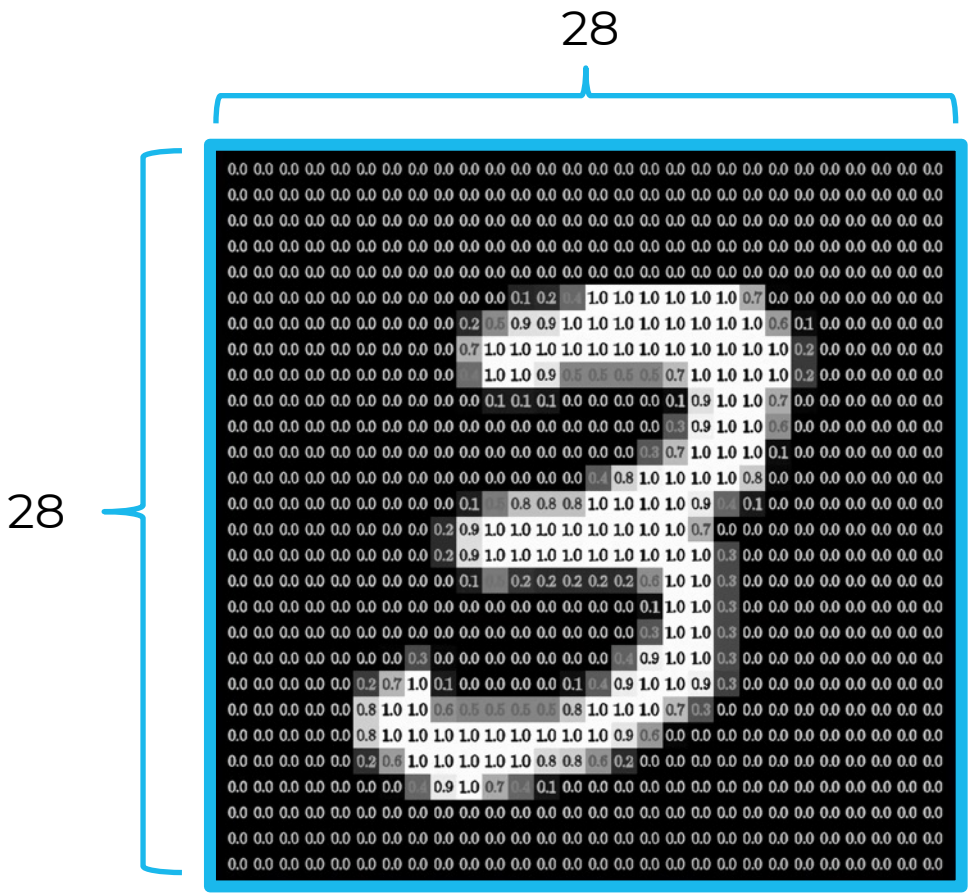


A simple neural network

$$\begin{bmatrix} w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} b \\ b \\ b \end{bmatrix} = \begin{bmatrix} w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \\ w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \\ w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \end{bmatrix} \xrightarrow[activation]{} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

MNIST

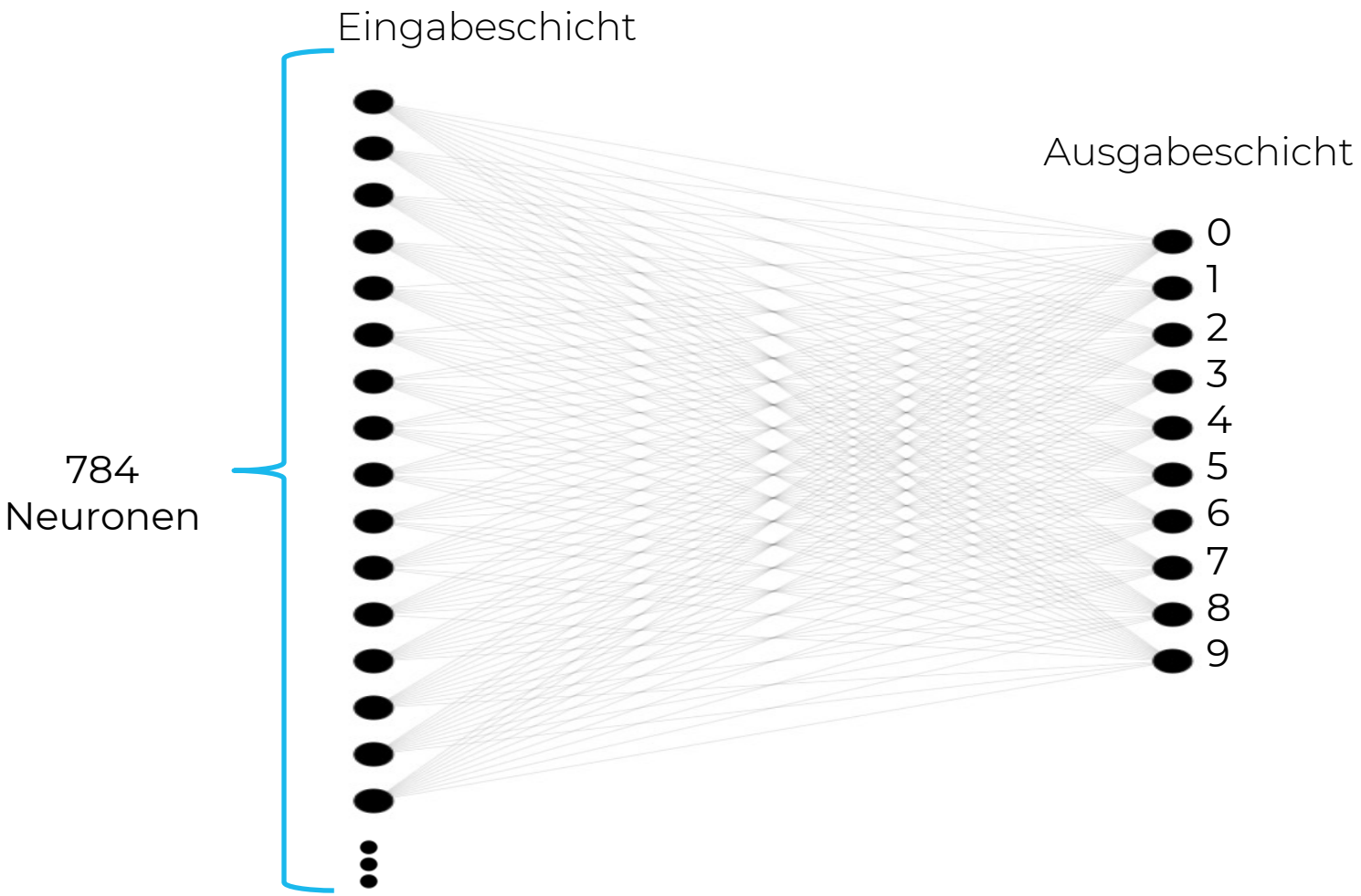
NEURONALE NETZE



~70.000 Bilder mit je
28*28=784 Pixel

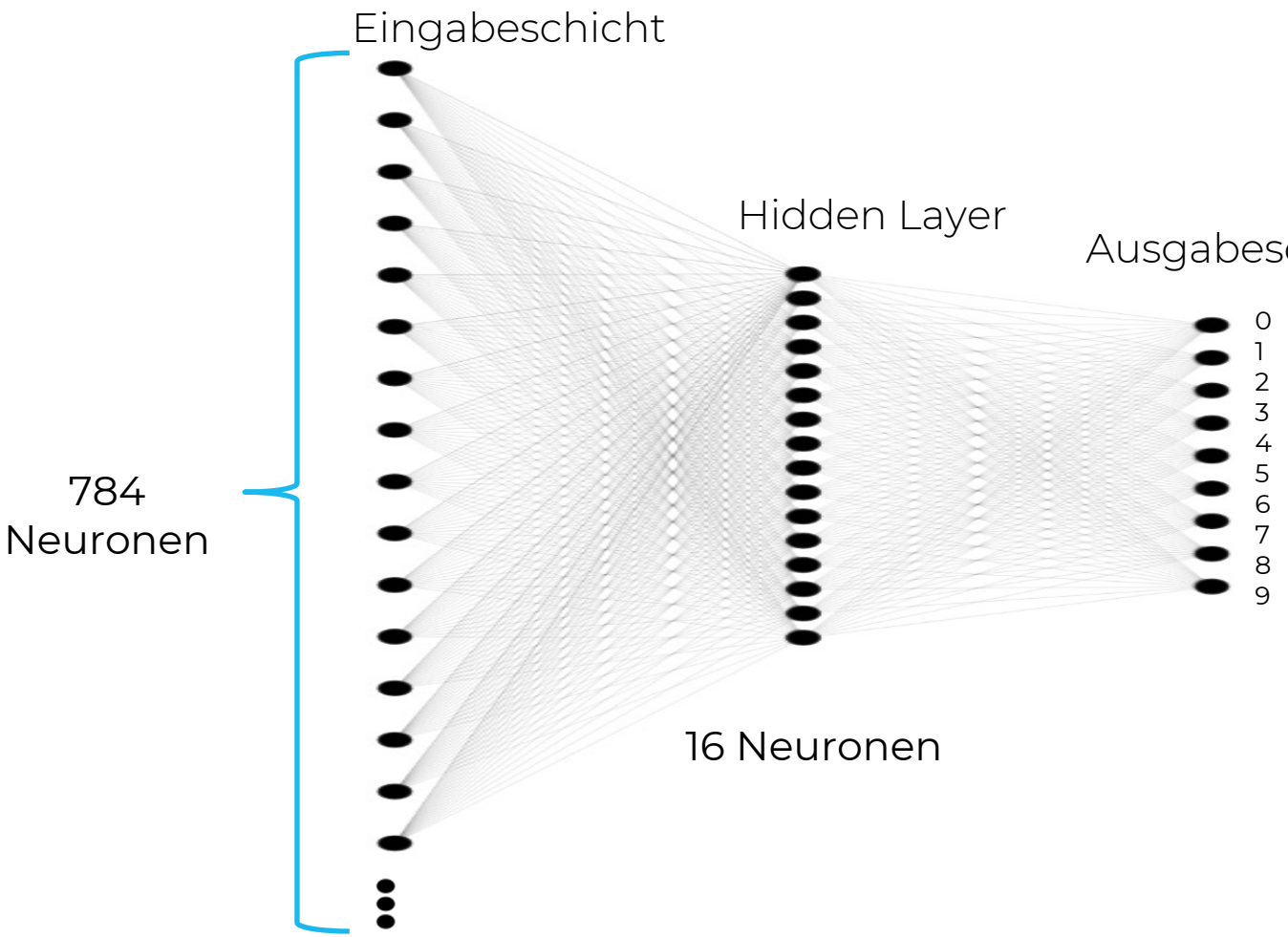
– Jeder Pixelwert wird transformiert in eine Zahl zwischen 0 und 1 **(Aktivierung)**
  Je höher der Wert, um so heller ist der Pixel

NEURONALE NETZE



- Jedes Neuron der Ausgabeschicht steht für eine Klasse
- Die Neuronen der Ausgabeschicht haben auch eine Aktivierung zwischen 0 und 1
- Der Grad der Aktivierung der Neuronen der Ausgabeschicht steht für Wahrscheinlichkeit der Klasse, die ermittelt wurde

NEURONALE NETZE



- Die Neuronen der Hidden Layer haben auch eine Aktivierung zwischen 0 und 1
- Hidden Layer ermöglicht zusätzliche Dimension für „abstrakte“ Features (nächste Folie)

748*16+16*10= 12128 Parameter + b

Die Aktivierung jeder Schicht führt zur Aktivierung der nächsten Schicht

DEMO

HYPERPARAMETER

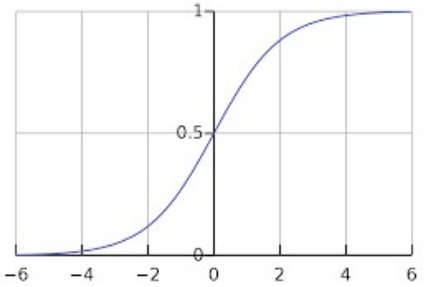
NEURONALE NETZE

AKTIVIERUNGSFUNKTIONEN

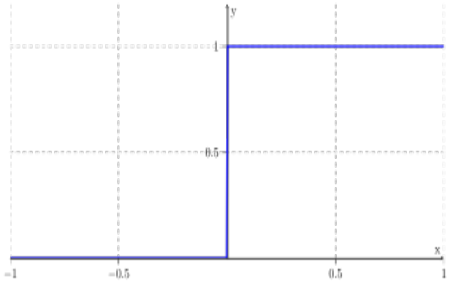
- Aktivierungsfunktionen dienen zur Skalierung einer gewichteten Summe, damit Werte vergleichbar werden
  - Im Anwendungsbeispiel: Sigmoidfunktion, skaliert die Aktivierungswerte der einzelnen Neuronen auf 0-1

$$\sigma * \left(w_{0.1} * a_1^{(0)} + w_{0.2} * a_2^{(0)} + w_{0.3} * a_3^{(0)} + \cdots + w_{0.784} * a_{784}^{(0)} + b \right)$$

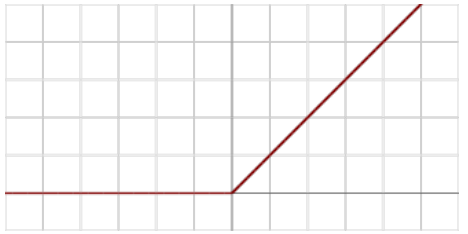
Gewichtete Summe



- Schwellenwertfunktion: Skaliert die Werte einer gewichteten Summe auf 0 oder 1

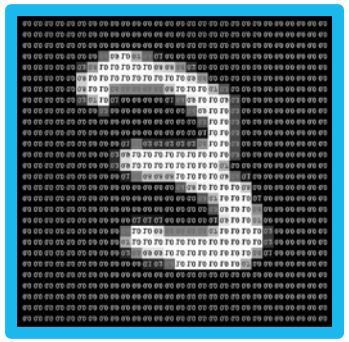


- Rectifier (ReLU): Negative gewichtete Summen werden zu 0 skaliert,
  positive gewichtete Summen werden proportionale Werte zugeordnet



NEURONALE NETZE – SUPERVISED LEARNING

Neuronales Netz wird mit **Trainingsdaten** trainiert, bestehen aus **Eingabewerten** und **Ausgabewerten**; 10 Klassen: Zahlen von 0 bis 9



$$
\begin{aligned}
&(0.06 - 0.00)^2 + \\
&(0.01 - 0.00)^2 + \\
&(0.03 - 0.00)^2 + \\
&(0.16 - 1.00)^2 + \\
&(0.01 - 0.00)^2 + \\
&(0.31 - 0.00)^2 + \\
&(0.00 - 0.00)^2 + \\
&(0.03 - 0.00)^2 + \\
&(0.05 - 0.00)^2 + \\
&(0.34 - 0.00)^2
\end{aligned}
$$

Aktivierung der Neuronen in der Ausgabeschicht

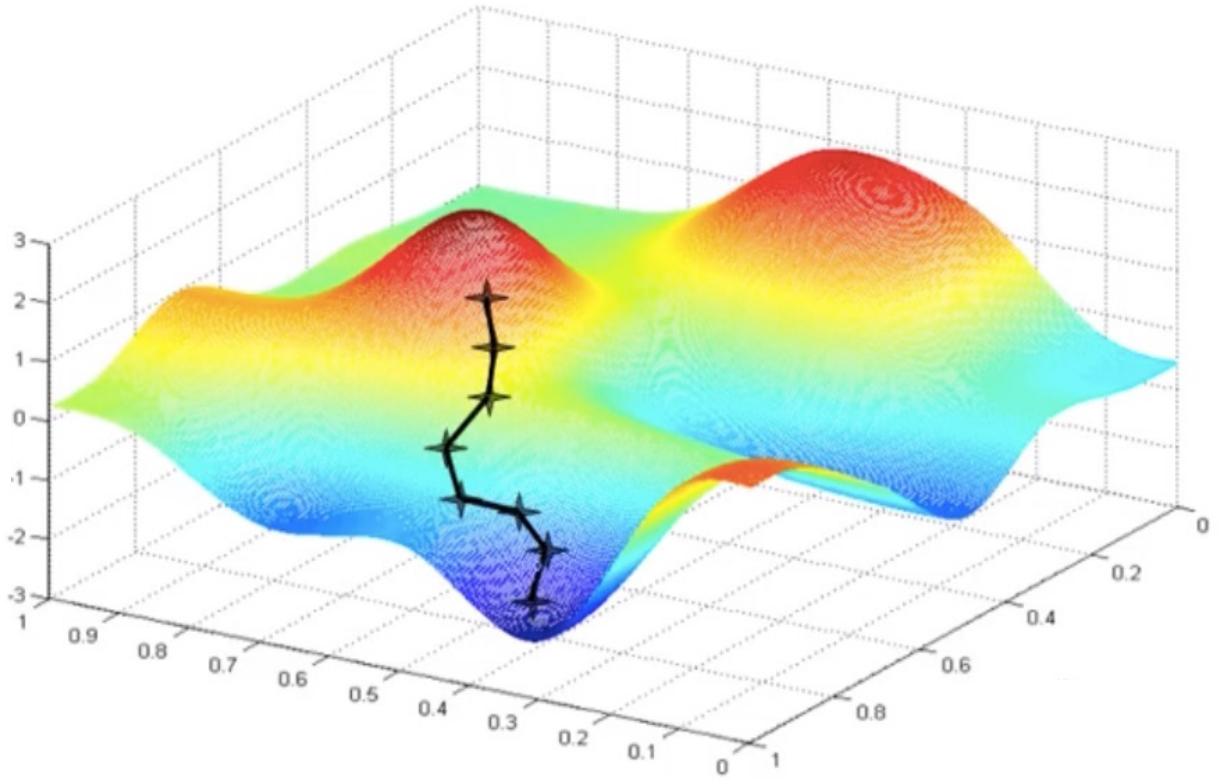
Kostenfunktion wird **minimal** wenn das neuronale Netz die geschriebene Zahl richtig erkennt

Ermittlung des Minimums der Kostenfunktion → Gradientenverfahren

Wie müssen die Gewichtungen und Bias angepasst werden → Backpropagation

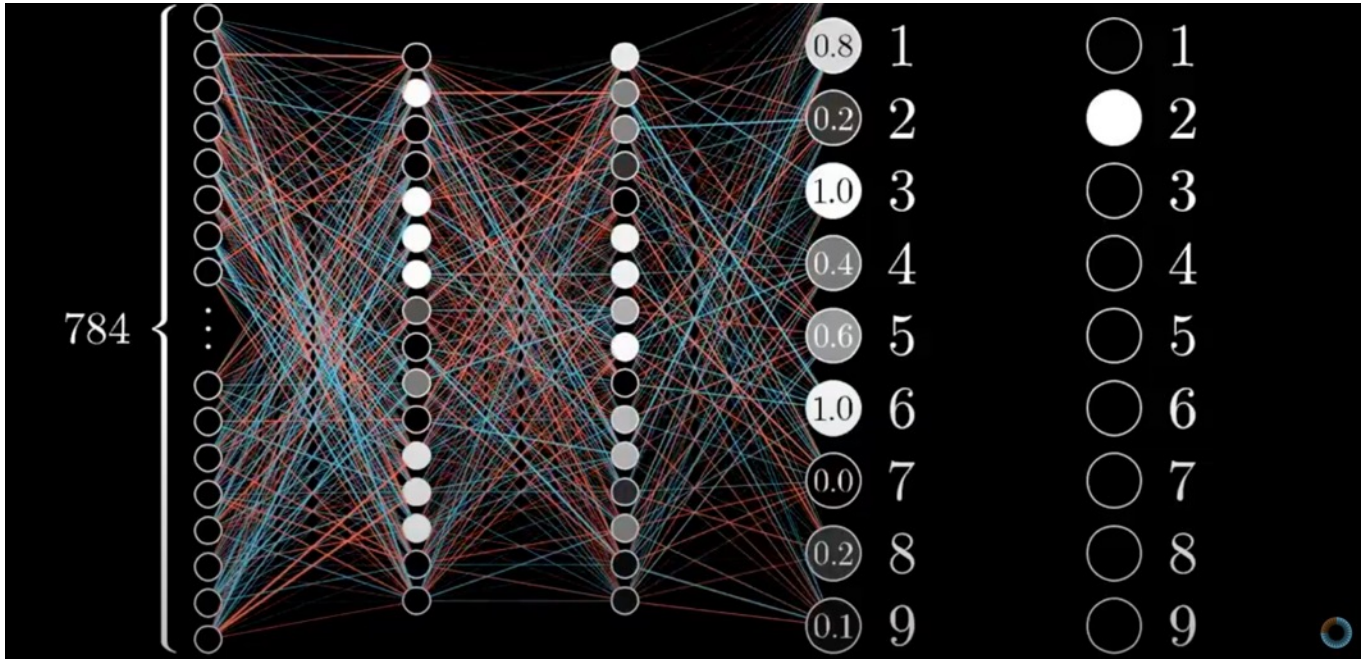
GRADIENTENVERFAHREN

SGD – STOCHASTIC GRADIENT DESCENT



WEITERFÜHRENDE RESSOURCEN

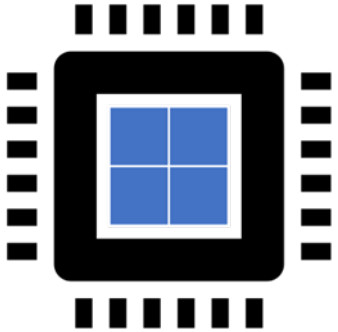
3BLUE1BROWN

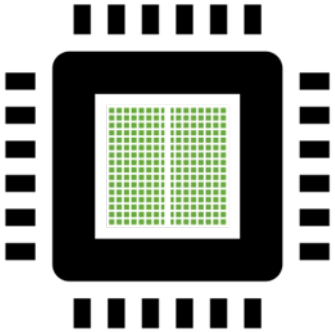


https://youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi

DEMO

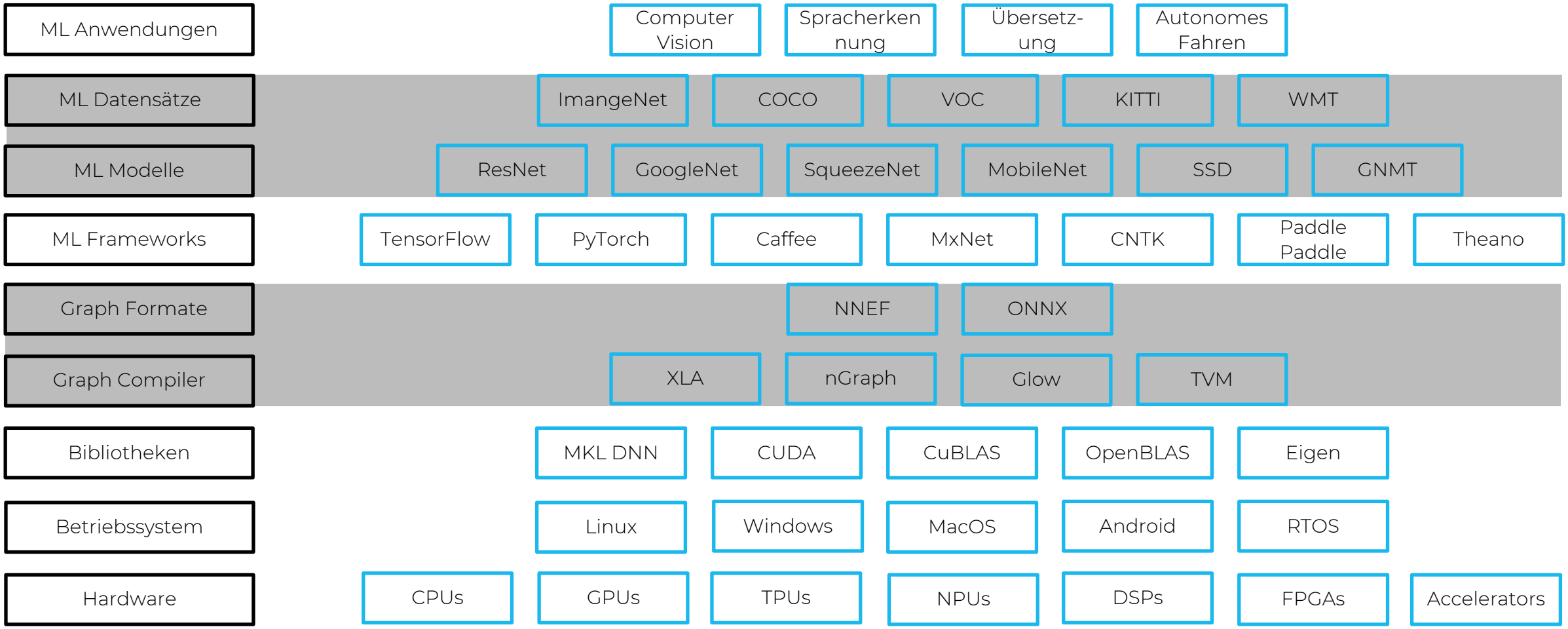
CPU VS. GPU



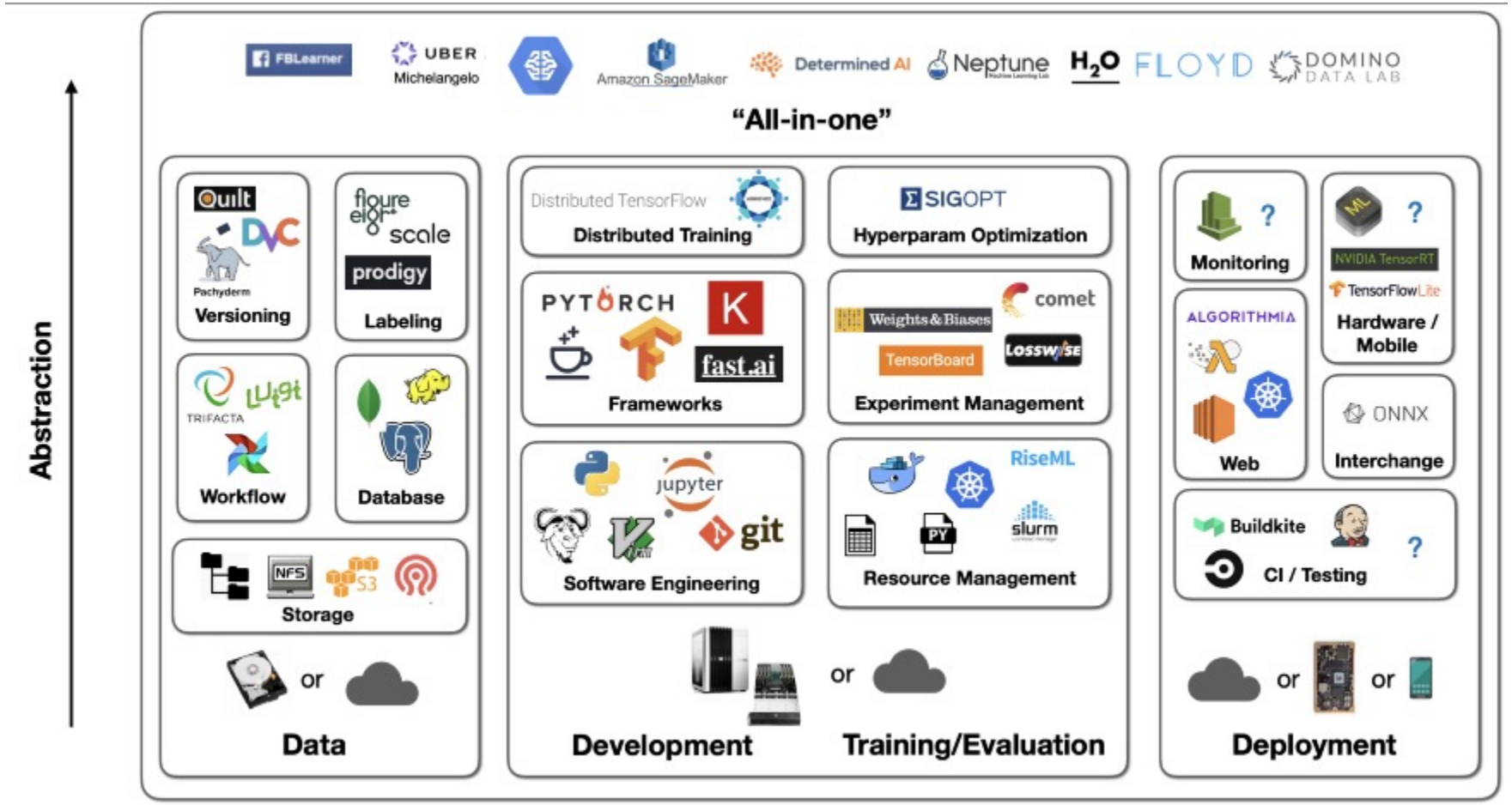


CPU (Central Processing Unit, Hauptprozessor)	GPU (Graphics Processing Unit, Grafikprozessor)
wenige Kerne	viele Kerne
Geringe Latenz	Hoher Durchsatz
Gut für serielle Bearbeitung	Gut für parallele Bearbeitung
Schnelle Bearbeitung von Aufgaben die Interaktivität erfordern	Unterteilt eine Aufgabe in Teilaufgaben, die simultan bearbeitet werden
Programme werden meistens für die sequentielle Ausführung geschrieben	Zusätzliche Software für die parallele Ausführung notwendig

DEEP LEARNING STACK



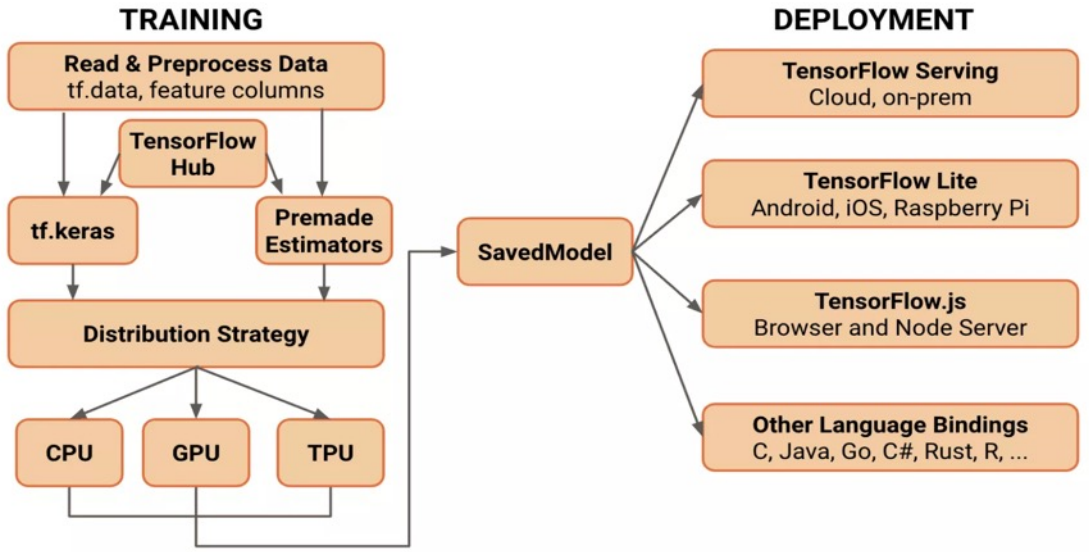
TECHNOLOGIE-INFRASTRUKTUR



ML-FRAMEWORKS

TENSORFLOW/KERAS

- TensorFlow: **Framework** für maschinelles Lernen und künstliche Intelligenz
  - **Erstellung** von **neuronalen Netzen** mit Hilfe von Tensoren (mehrdimensionale Datenfelder)



- Keras: **Open-Source-Bibliothek**:
  - Eigenständige Bibliothek, wird als **Interface** (Schnittstelle) unter anderem für TensorFlow genutzt
  - Macht die Anwendung von TensorFlow nutzerfreundlich
  - Erstellung von neuronalen Netzen, ohne sich im Detail mit darunter liegenden Backends beschäftigen zu müssen
  - Stellt einfach zu verwendende **APIs** (Application Programming Interface) bereit

    ⟶ Befehle, Funktionen, Protokolle, Objekte zur Erstellung einer Software

ML-FRAMEWORKS

TENSORFLOW/KERAS

```
l = tf.keras.layers
model = tf.keras.Sequential([
l.Flatten(input_shape=(784,)),
l.Dense(128, activation='relu‘),
l.Dense(128, activation='relu‘),
l.Dense(10, activation='softmax') ])

model.compile(loss='categorical_crossentropy',
optimizer='adam',metrics = ['accuracy‘])

model.summary() model.fit(x_train.reshape(-
1,784),pd.get_dummies(y_train),nb_epoch=15,batch_size=
128,verbose=1)
```

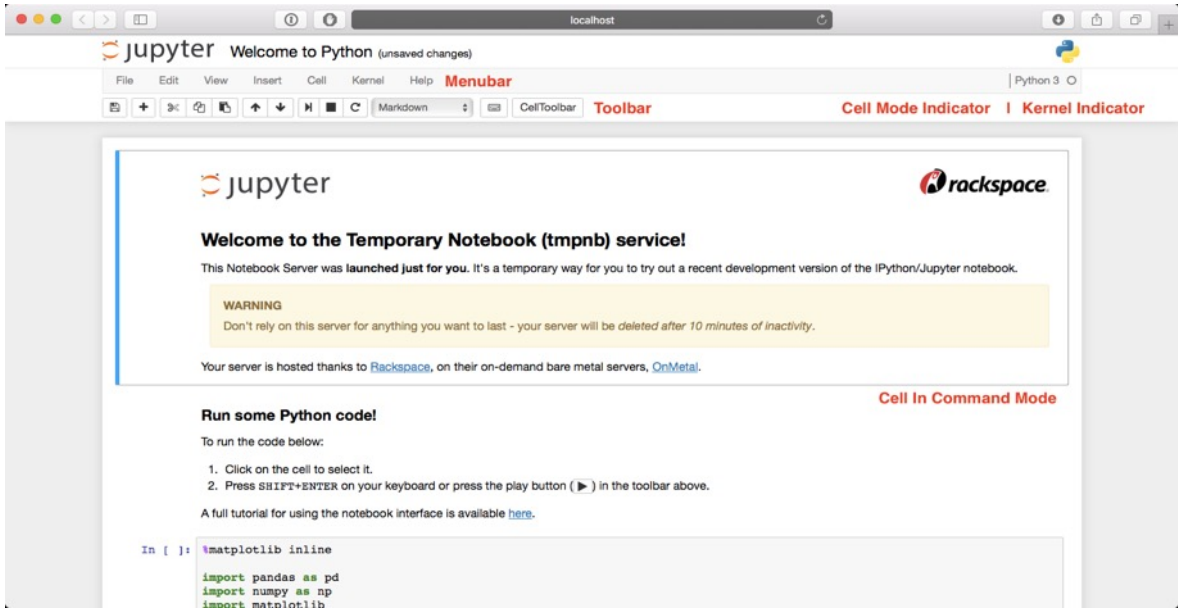
```
X = tf.placeholder(dtype=tf.float64)
Y = tf.placeholder(dtype=tf.float64)
num_hidden=128

# Build a hidden layer
W_hidden = tf.Variable(np.random.randn(784,
num_hidden))
b_hidden = tf.Variable(np.random.randn(num_hidden))
p_hidden = tf.nn.sigmoid( tf.add(tf.matmul(X,
W_hidden), b_hidden) )

# Build another hidden layer
W_hidden2 = tf.Variable(np.random.randn(num_hidden,
num_hidden))
b_hidden2 = tf.Variable(np.random.randn(num_hidden))
p_hidden2 = tf.nn.sigmoid( tf.add(tf.matmul(p_hidden,
W_hidden2), b_hidden2) )

# Build the output layer
W_output = tf.Variable(np.random.randn(num_hidden,
10))
b_output = tf.Variable(np.random.randn(10))
p_output = tf.nn.softmax( tf.add(tf.matmul(p_hidden2,
W_output), b_output) )
loss = tf.reduce_mean(tf.losses.mean_squared_error(
labels=Y,predictions=p_output))
accuracy=1-tf.sqrt(loss) minimization_op =
tf.train.AdamOptimizer(learning_rate=0.01).minimize(lo
ss) feed_dict = { X: x_train.reshape(-1,784), Y:
pd.get_dummies(y_train) } with tf.Session() as
```

SOFTWARE ENGINEERING

JUPYTER NOTEBOOK

- Web-basierte **interaktive Umgebung**, mit der Jupyter-Notebook-Dokumente erstellt und geteilt werden können
- Programmiersprachen: **Julia, Python, R**

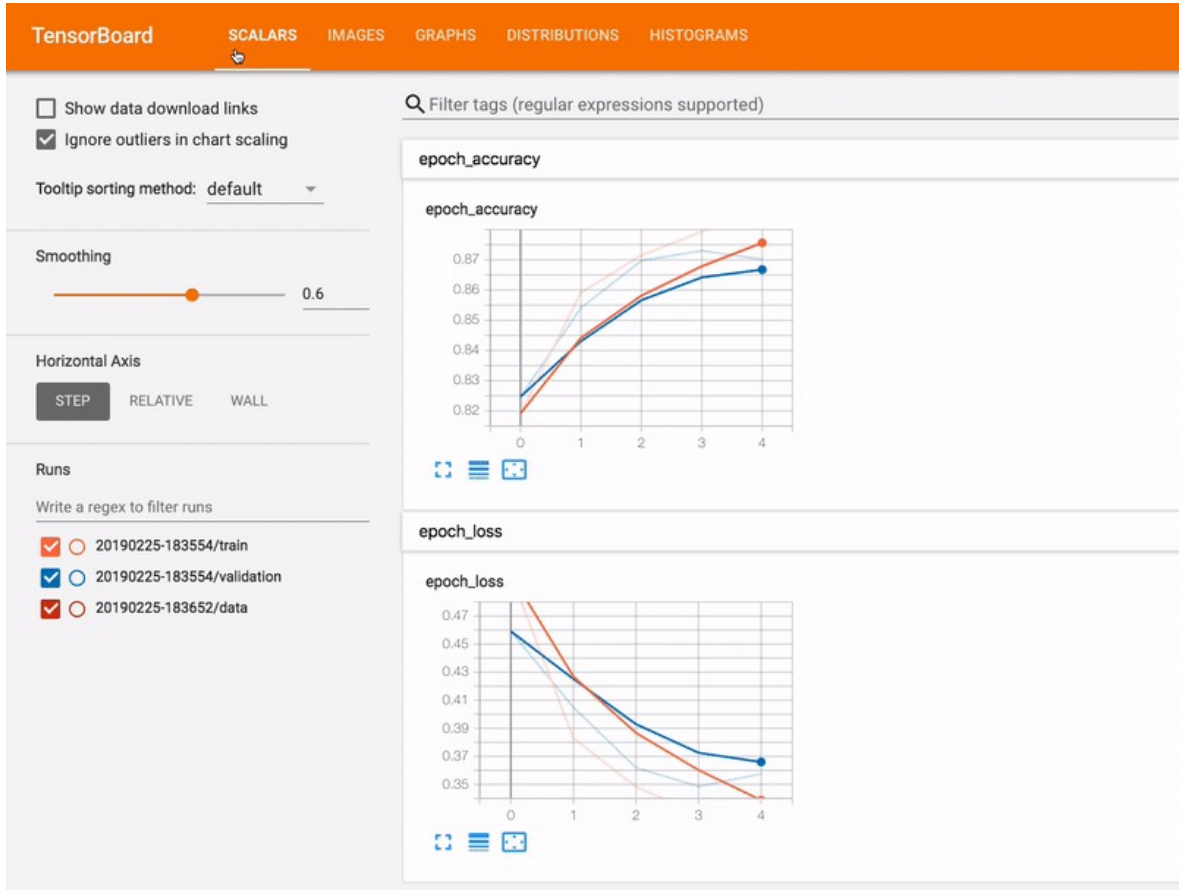
- Wichtigste **Einsatzzwecke**:
  - **Datenbereinigung**: Unterscheidung zwischen wichtigen und unwichtigen Daten in der Big-Data-Analyse
  - **Statistische Modellierung**: Mathematische Methode zur Ermittlung der geschätzten Wahrscheinlichkeitsverteilung eines bestimmten Merkmals
  - **Kreation und Training von ML-Modellen**: Entwurf, Programmierung und Training von Modellen, die auf ML basieren
  - **Datenvisualisierung**: Grafische Darstellung von Daten zur Verdeutlichung von Mustern, Trends, Abhängigkeiten etc.



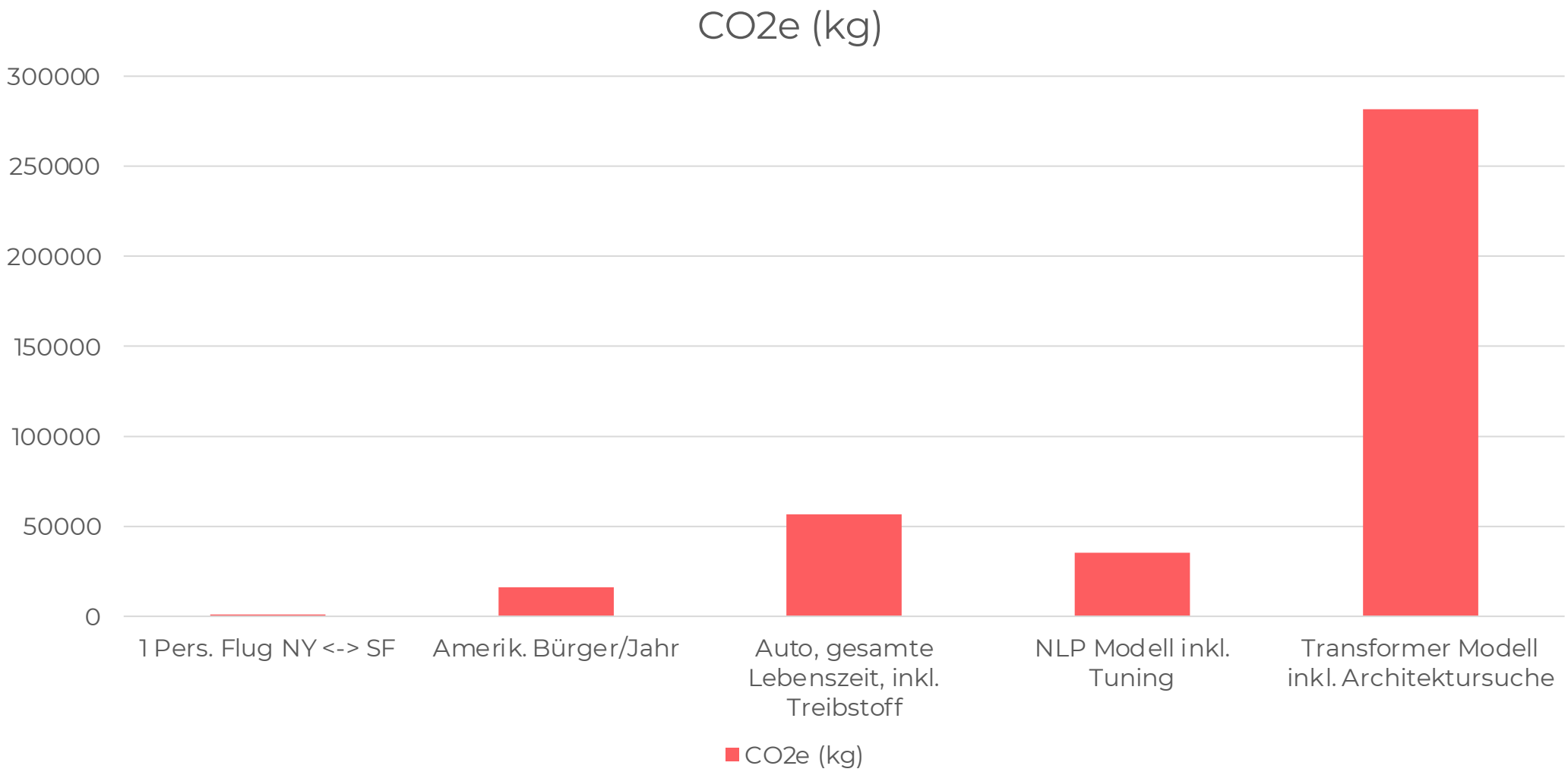
EXPERIMENT MANAGEMENT

TENSORBOARD

- **Tool** zur **Bereitstellung** von **Messungen** und **Visualisierungen** während des maschinellen Lernworkflows mit TensorFlow
  - Funktionen:
    - **Visualisierung** verschiedener Metriken wie Verlust oder Genauigkeit mit Hilfe von Diagrammen und Histogrammen
    - Visualisierung von Modellebenen und Operationen durch Diagramme
    - Anzeige von Trainingsdaten (Bild-, Audio- und Textdaten)



HERAUSFORDERUNG RESSOURCENVERBRAUCH



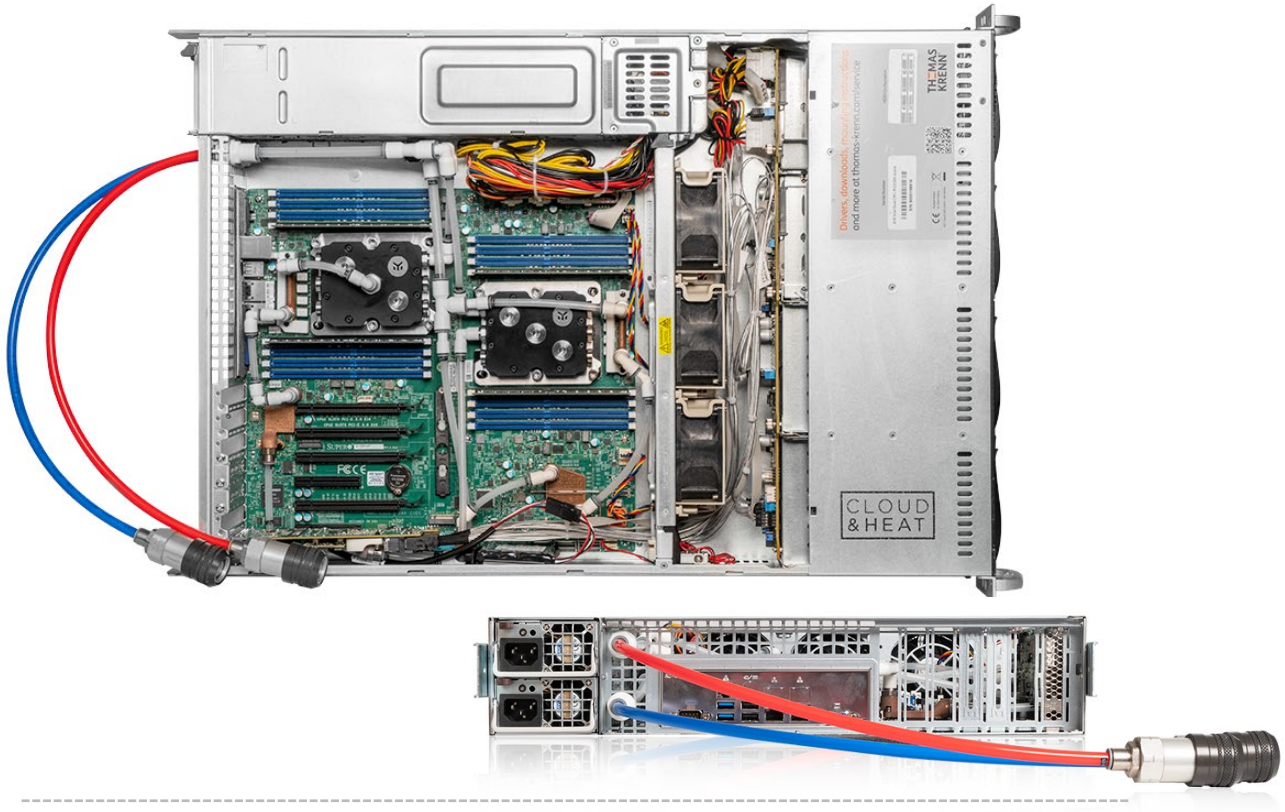
KÜHLKÖRPER

WASSERGEKÜHLTES CUSTOMIZED COOLING KIT

- Einsatz individueller Hardware in Wasserkühlungsinfrastruktur wird ermöglicht
- Umrüstung von luftgekühlter Hardware auf effiziente Direkt-Heißwasserkühlung, optimiert für maximale IT-Leistung und hohe Temperaturniveaus bis zu 60 °C
- Entwicklung von OEM-Lösungen

REFERENZEN VON OEM-LÖSUNGEN:

- Thomas Krenn RI 2208-LCS
- Supermicro 9029
- Nvidia RTX-Server
- TradeDX R5211



VORGEHENSWEISE:

1 | Evaluierung & Konzeption

2 | Entwicklung Prototyp

3 | Entwicklung Serienprodukt

4 | Zertifizierung

KUNDEN:

THOMAS KRENN | T | VATTENFALL

VORTEILE:

- Kühlung **CPUs, GPUs, Peripherie-ICs, Spannungswandler, Netzwerkchip und –karten, PCIe-Erweiterungskarten**
- Steigerung der **Leistungsdichte** pro Rack
- Reduktion der **Betriebskosten**
- Reduzierung der **Lüfterleistung** & des **Geräuschpegels**
- **Schnellkupplungen** zur einfachen und sicheren Montage

4 IHRE ANWENDUNG AUF UNSERER INFRASTRUKTUR

Im Cloud&Heat-Ökosystem arbeiten wir **Hand in Hand**, um Ihnen als **Provider** mit einer **auf Ihre Bedürfnisse** zugeschnittenen, **ganzheitlichen Lösung** zu unterstützen.

Wir wollen Ihre Anwendung verstehen.

Nutzen Sie unsere Expertise auf allen Ebenen.

Mit unserem Support begleiten wir Sie von der ersten Anfrage bis zum Einsatz der Anwendung.

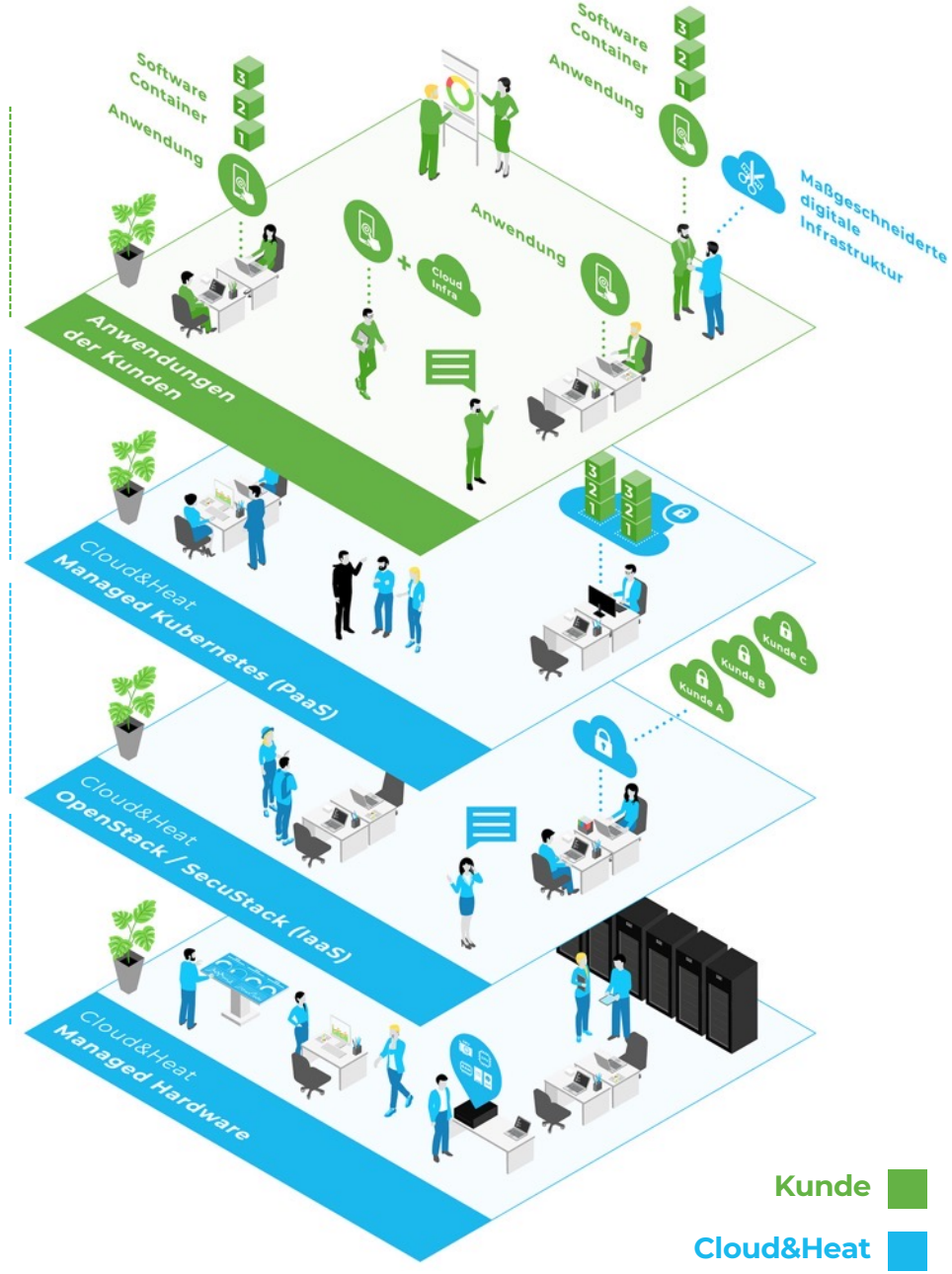
Konzentrieren Sie sich auf Ihr Kerngeschäft.

Anwendungen der Kunden
Software as a Service (SaaS)

Managed Kubernetes
Platform as a Service (PaaS)

OpenStack / SecuStack
Infrastructure as a Service (IaaS)

Managed Hardware
Bare Metal (GPU, CPU, RAM, Storage)



PARTNER & KUNDEN

nyris | ai4bd | MOVEBIS | cyface EASY RIDING

WER WIR SIND

CLOUD&HEAT IN ZAHLEN

2011

GRÜNDUNG

in Dresden, Deutschland

100

MITARBEITER

aus 10 verschiedenen Ländern

4

STANDORTE

Dresden, Frankfurt a. Main, Madrid und Dubai