SEMINAR
KLASSIFIKATION & CLUSTERING

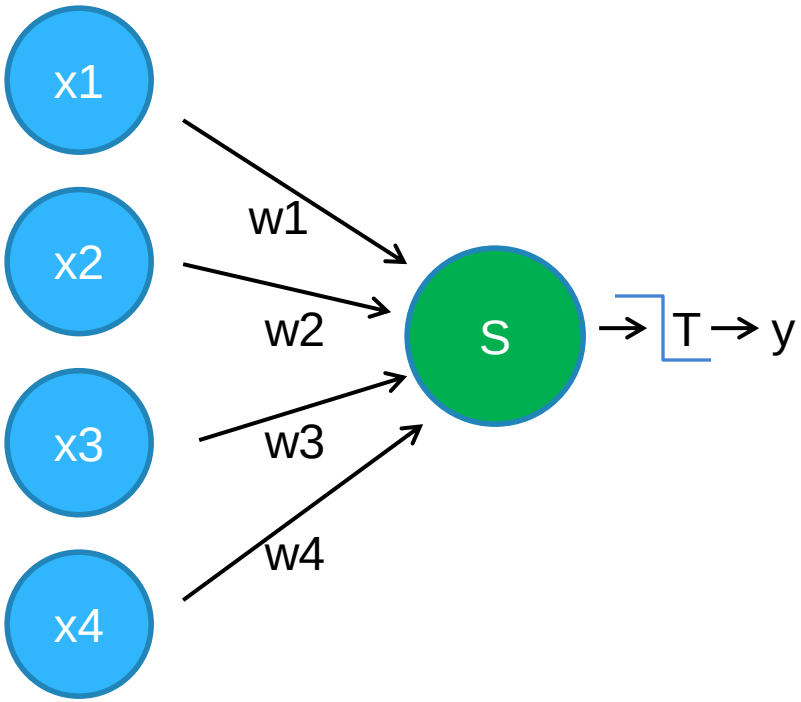
KÜNSTLICHE NEURONALE NETZE, DEEP LEARNING UND TEXTKLASSIFIZIERUNG

Stefan Langer

stefan.langer@cis.uni-muenchen.de

PERZEPTRON

Ein einfaches Perzeptron ist ein einfaches neuronales Netz das nur aus gewichteten Eingaben und einem Summen/Ausgabeknoten



x1 – x4: Eingabedaten

w1 – w4: Gewichtung

S: Knoten mit Summenfunktion

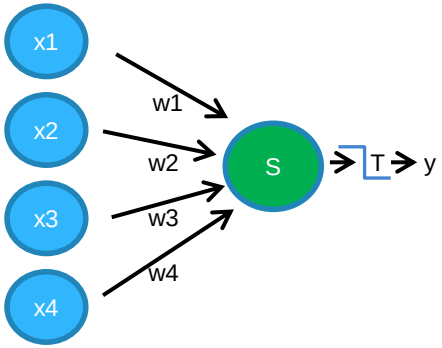
T: Schwellenwertfunktion, alternativ Sigmoid-Funktion

y: Output (0 oder 1)

Zu lernen: w1-w4

Eigenschaften: Linearer Klassifikator (Daten sollten über Hyperebene separierbar sein)

PERZEPTRON - Lernverfahren



Ziel: Lerne w1 – w4

Bekannt: der Wert für y für das Trainingsset, dj

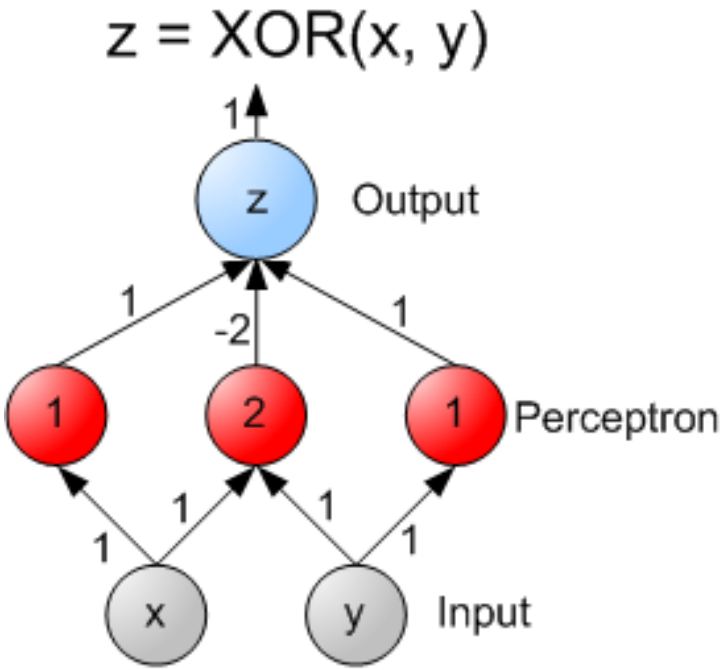
Dazu: Ermittle den Output d für Vektor x1-x4

$$T(\sum x_i * w_i)$$

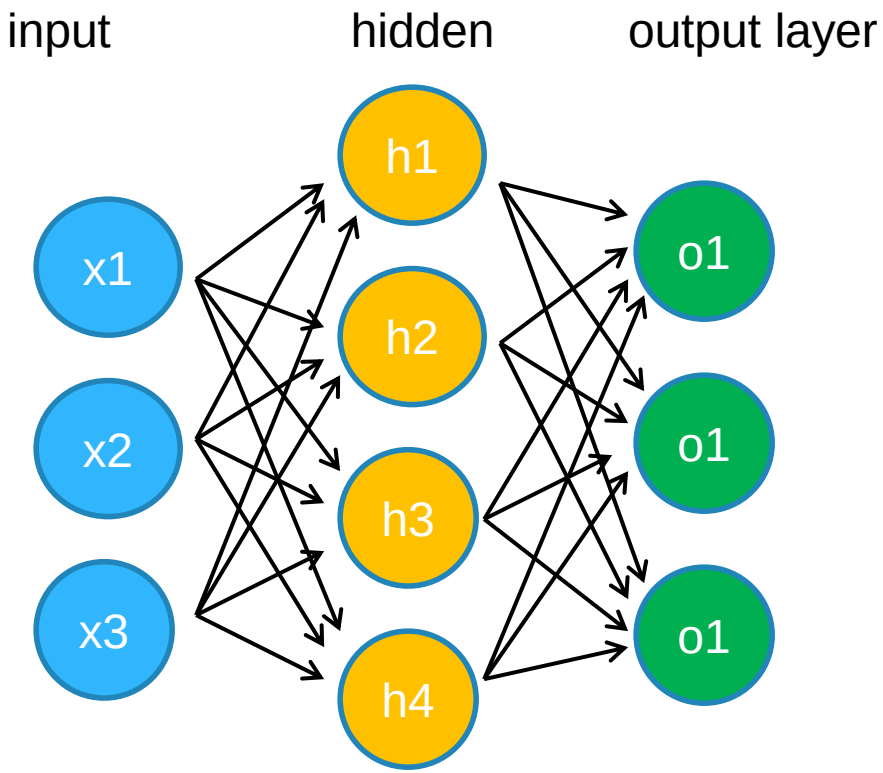
Ermittle die neuen w1-w4

$$w_{i\,=}\, w_i + (d - y)\, x_i$$

X-Or mit einfachem neuronalen Netz



Künstliches Neuronales Netz

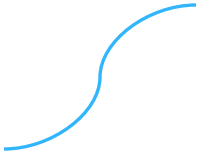


Aktivierungsfunktionen für Neuronen

- Threshold

- Linear (höchsten für Input-Neuronen)

- Sigmoid & related

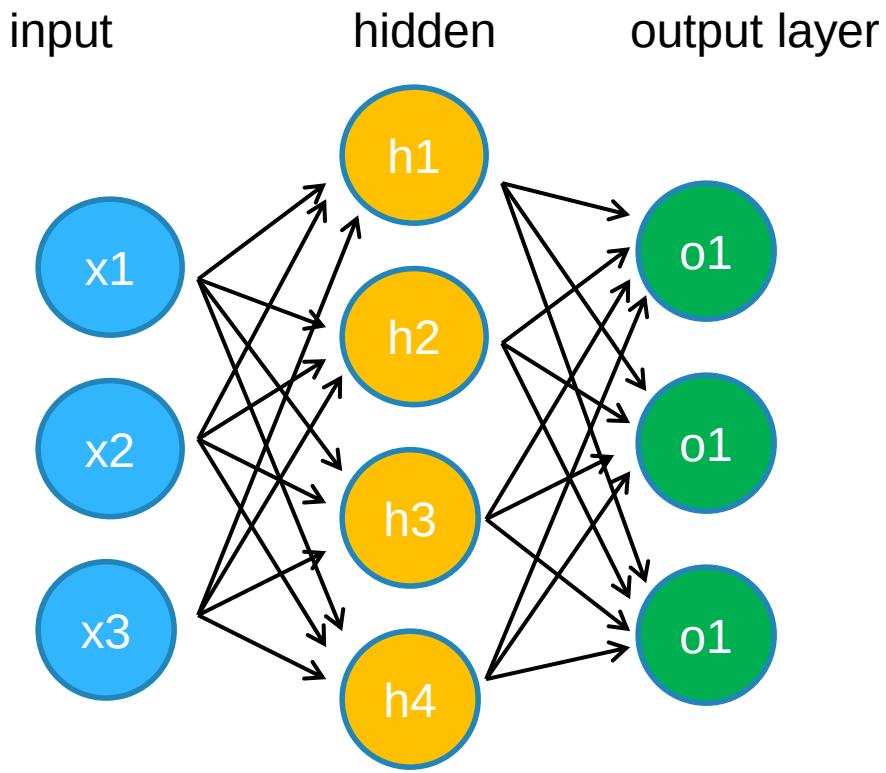


$$sig(x) = \frac{1}{1 - e^{-x}}$$

Typen von neuronalen Netzen

- Feed-Forward NN
  - Single layer, also single layer perceptron
  - Multi-layer
- Convolutional NN
- Recurrent NN
  - LSTM
- Transformers
  - Bert (bidirectional)
  - GPT models et al

Feed-forward NN, mehrlagig



Feed forward NN

Training for an FF NN: Backpropagation

- Supervised learning (überwachtes Lernverfahren) – i.e. Output muss bekannt sein
- Spezialfall eines allgemeinen Gradientenverfahrens
- Basierend auf dem mittleren quadratischen Fehler

Mittlerer quadratischer Fehler

$$E = \frac{1}{2}\sum_{i=1}^{n}(t_i - o_i)^2$$

- E Mittlerer quadratischer Fehler,
- n Anzahl der Eingabedatensets
- t Soll-Ausgabe (target)
- o Ist-Ausgabe (output).

Ausgabefunktion eines Neurons

$$o_i = f\left(\sum_{j=1}^{n} x_j * w_{ij}\right)$$

- o

Ausgabefunktion eines 2-lagigen neuronalen Netzes

$$o_k = f\left(\sum_{j=1}^{n1} f\left(\sum_{i=1}^{n} x_i * w_{ij}\right) * w_{jk}\right)$$

- o Output value
- n Anzahl der Eingabewerte
- n1 Anzahl der Hidden Neurons
- x Eingabewert
- w Gewicht

Backpropagation - Algo

- Gib Eingabevektor x in das Netzwerk ein
- y ist der Augabevektor, t der erwartete Vektor
- Berechne für jedes Output-Neuron den Beitrag d zum Netzwerkfehler: $d_i = y_i(1-y_i)(t_i-y_i)$
- Berechne für jedes Hidden Neuron den Beitrag d zum Fehler:

$$d_j = h_j(1-h_j)\sum_{k}^{no} d_k\, w_{jk}$$

- Aktualisiere die Gewichte nach folgender Formel (l ist hier die Lernrate)
  - output: $w_{jk} := w_{jk} + ld_k h_j$
  - hidden: $w_{ij} := w_{ij} + ld_j\, x_i$

NN - Herausforderungen

- Training durch Backpropagation:
  - Anzahl der Hidden * Anzahl der Inputs Gewichte zu trainieren
  - * Anzahl der Eingabewertvektoren
  - * Anzahl der Epochen
- Lernalgorithmus kann zu lokalem Minimum führen
- Lernrate sollte variabel sein (erst größer, später kleiner)
- Overfitting ist ein Problem, besonders bei mehreren Hidden layern

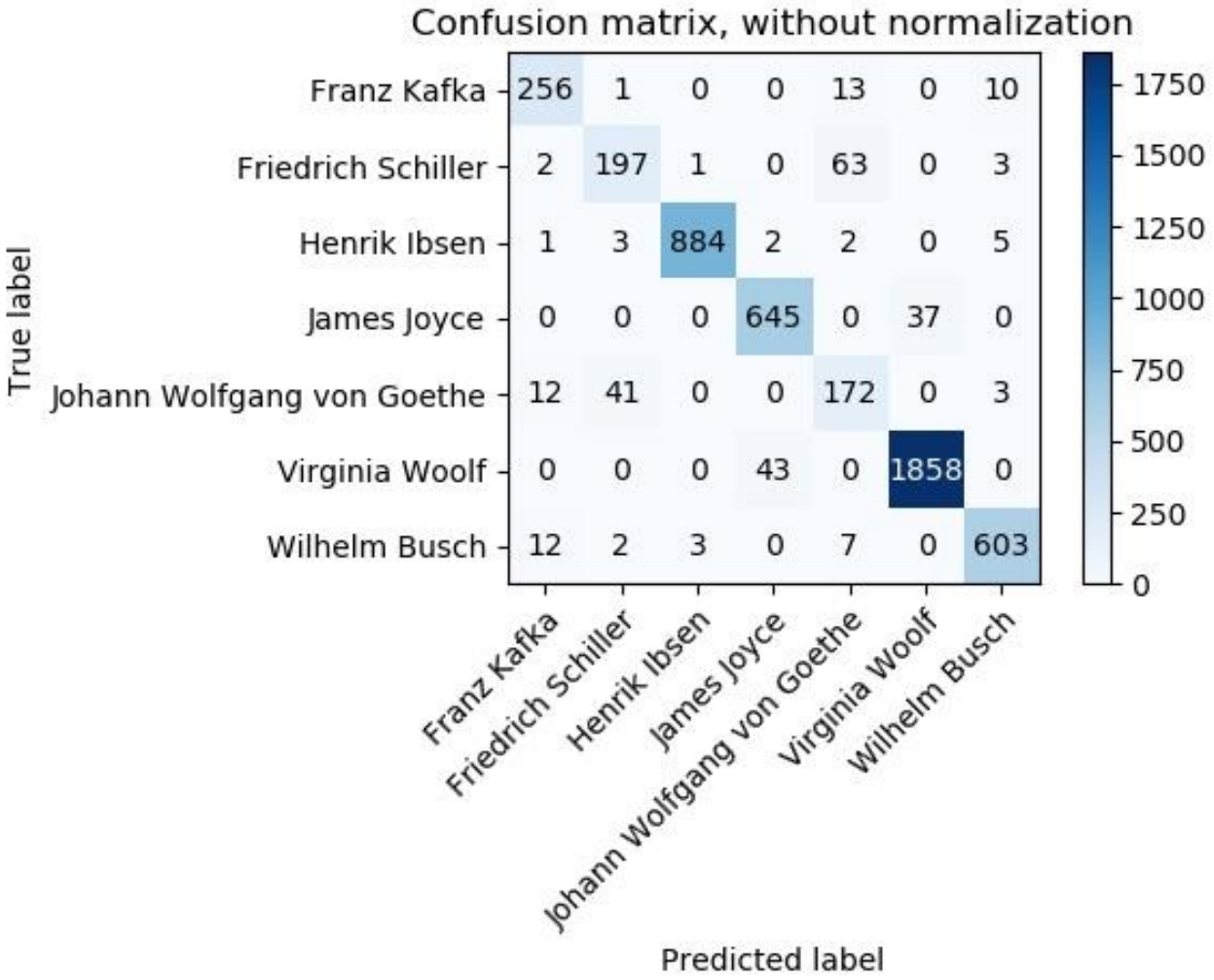
Resultate Autoren MLP

#Performance:
Seconds used for training: 897
Seconds used for classification: 13

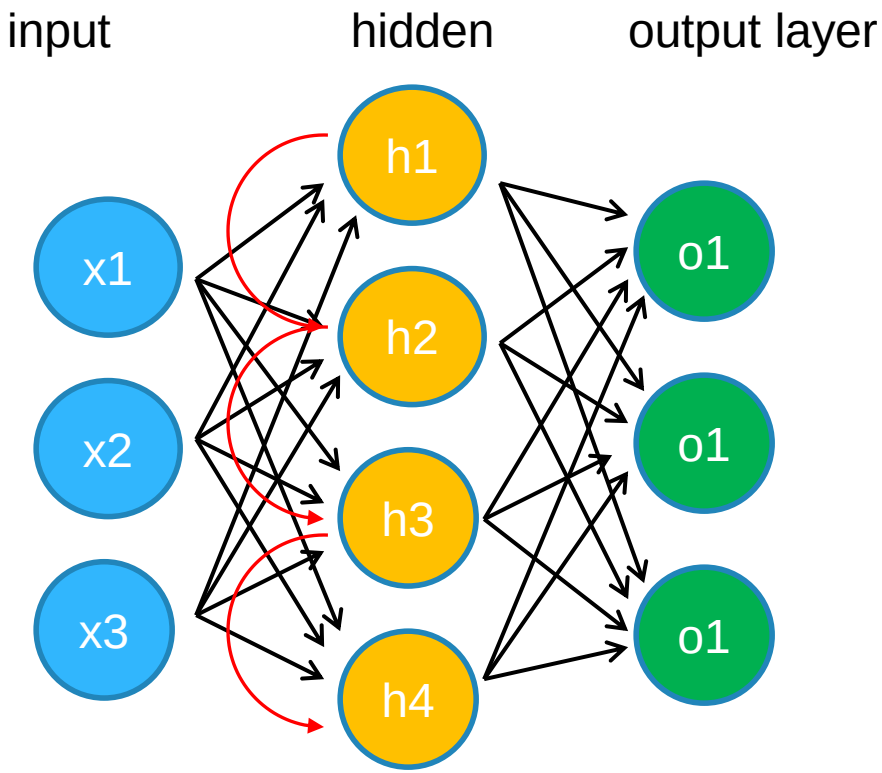
#Classification report:

	precision	recall	f1-score	support
Franz Kafka	0.92	0.93	0.92	280
Friedrich Schiller	0.77	0.83	0.80	266
Henrik Ibsen	1.00	0.99	0.99	897
James Joyce	0.93	0.94	0.93	682
Johann Wolfgang von Goethe	0.75	0.72	0.74	228
Virginia Woolf	0.98	0.98	0.98	1901
Wilhelm Busch	0.96	0.95	0.96	627
accuracy			0.95	4881
macro avg	0.90	0.90	0.90	4881
weighted avg	0.95	0.95	0.95	4881

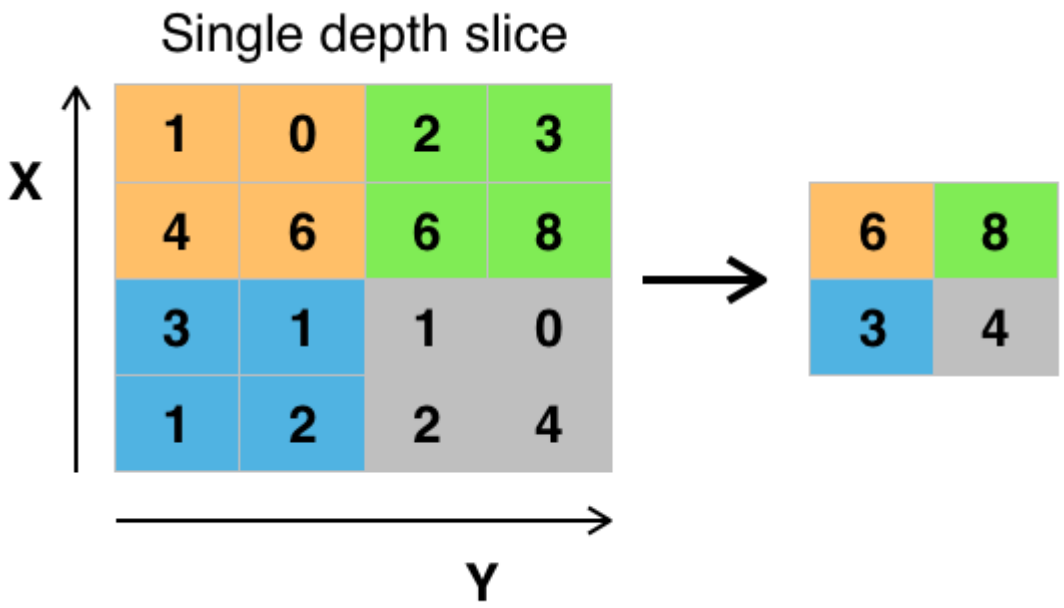
Resultate MLP



Recurrent NN



CNN - pooling

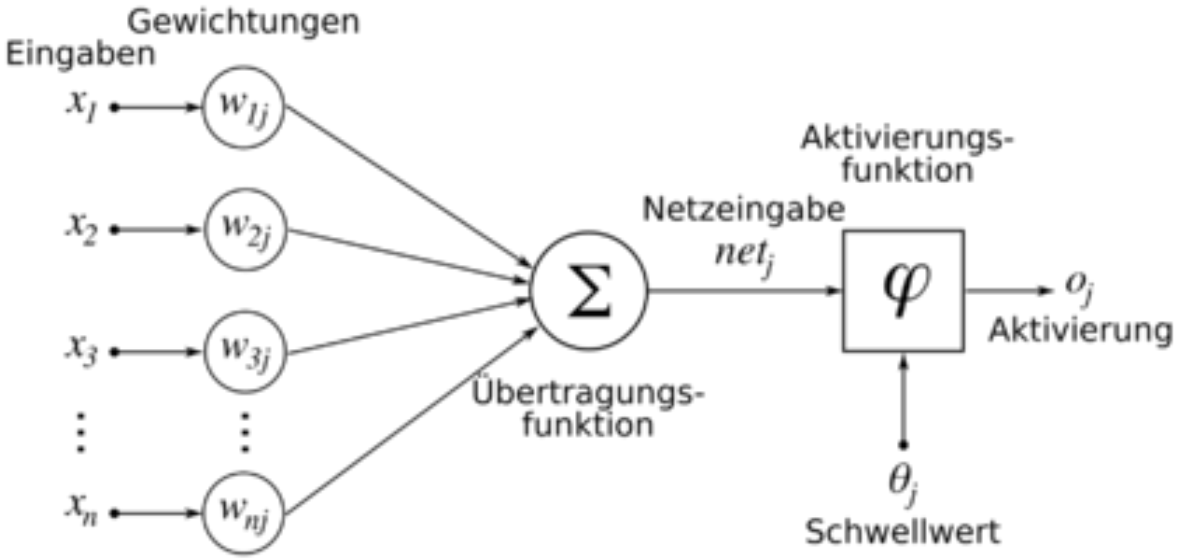


Quelle Wikipedia

Error backpropagation: Algorithm

- Propagate a input pattern through the network
- Compare input to desired output. Difference of values is the error
- Propagate error back to the input layer. Change Neuron connection weights depending on the error.

Error backpropagation: scheme

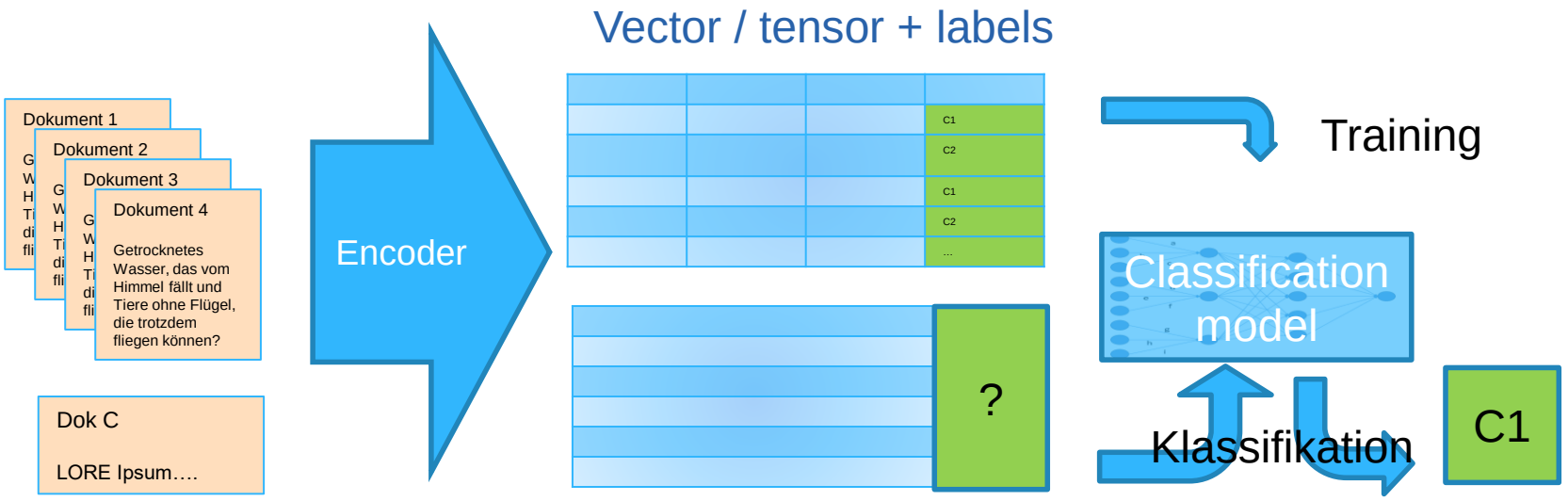


Source Wikipedia.de

Backpropagation: Pseudo-algorithm

```
initialize network weights (often small random values)
  do
     forEach training example ex
        prediction = neural-net-output(network, ex)  // forward pass
        actual = teacher-output(ex)
        compute error (prediction - actual) at the output units
        compute \Delta w_h for all weights from hidden layer to output layer  // backw
        compute \Delta w_i for all weights from input layer to hidden layer   // backw
        update network weights // input layer not modified by error estimate
  until all examples classified correctly or another stopping criterion satisfied
  return the network
```

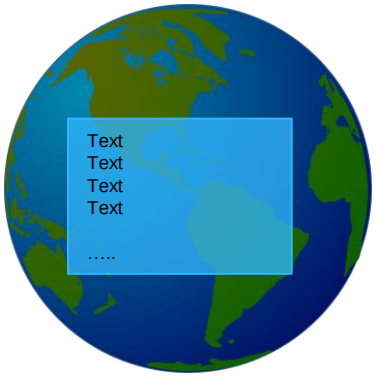
Klassifikation – neuronale Architektur

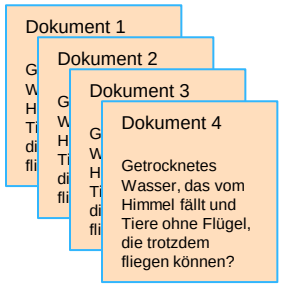


Encoder	Doc repr
TF-IDF	Sparse vector
Word embeddings	Matrix
Doc embeddings	Dense vector
Transformer (BERT, GPT)	Dense vector Matrix

Repr	Model
Vector	Any non-neural
Vector	FNN
Vector	RNN / LSTM
Matrix	CNN

Neuronale Architekturen - Encoder





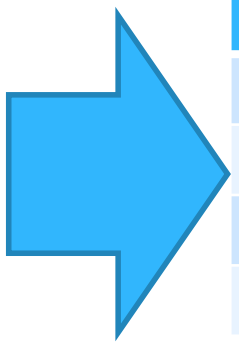
Pretraining

Finetuning

Encoder

Combination options

Text encoding	Doc repr
TF-IDF	Sparse vector
Word embeddings	Pooled embeddings
Doc embeddings and Transformer (BERT, GPT)	Dense vector: doc2vec Sbert Ada-002 AnglE-. LLaMA



Classifier
Any non-neural
FNN / MLP
CNN
Transformer (BERT, GPT)

GPT

Model name	tokenizer	max input tokens	output dimensions
text-embedding-ada-002	cl100k_base	8191	1536