Vorabfragen

Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Deep Learning, Machine Learning und Künstliche Intelligenz – SS 26

Gelesen von Prof. Dr. Daniel Gaida

Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Lernraum II: Deep Learning

- Künstliche neuronale Netze
  - Grundlagen
  - Training
  - Implementierung
  - Lösungsstrategien, um gute Ergebnisse zu erzielen
- Convolutional neuronale Netze
  - Bilderkennung
- **Rekurrente neuronale Netze**
  - **Zeitreihen, Text**
- **Transformer Netze**
  - **Text, Bilderkennung**
- Anwendungen
  - Entwickeln eines Chatbots
  - Entwickeln einer App zur Objektdetektion

Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik
Technology Arts Sciences TH Köln

Lernziele von heute und Fragen zur Überprüfung der Lernziele

Die Studierenden können ein Deep Learning Projekt zur **Prognose und Analyse von Sequenzen** durchführen, indem sie

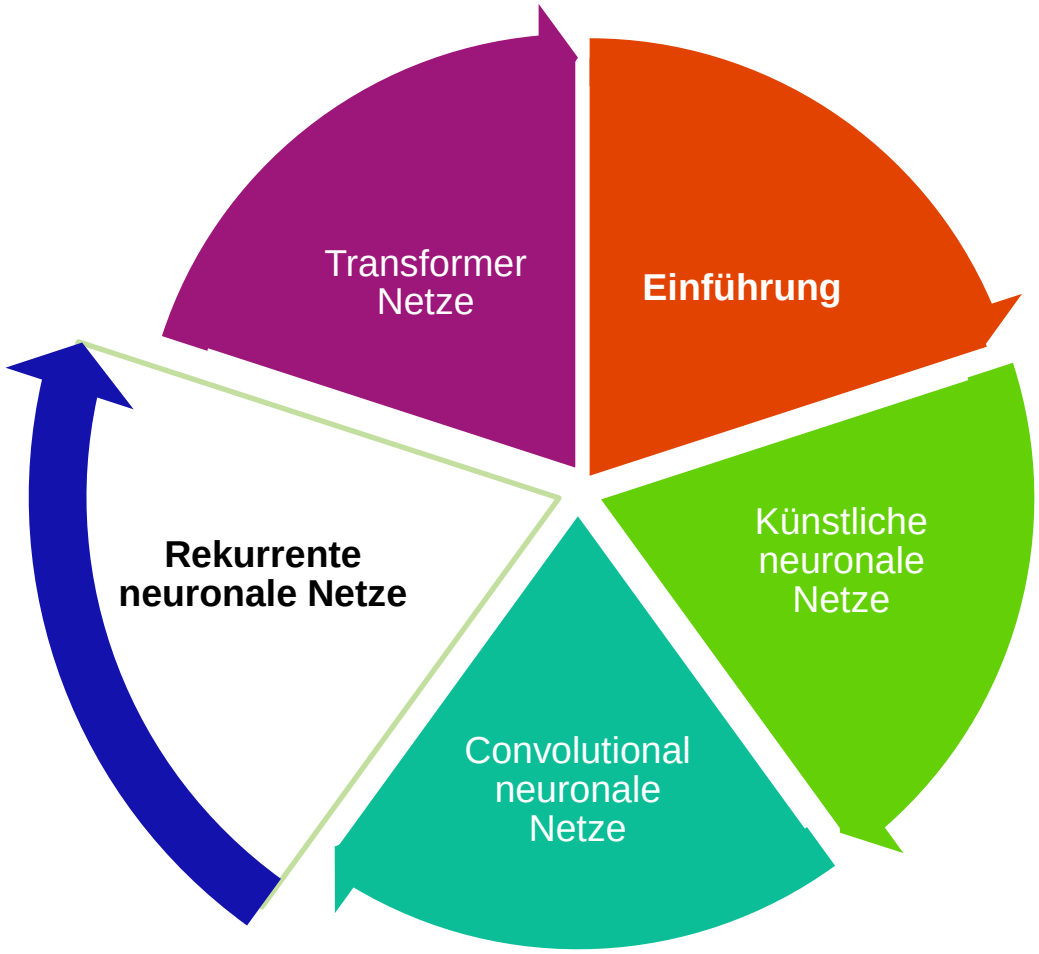
- für eine gegebene Anwendung ein geeignetes künstliches neuronales Netz auswählen können,
- bei Wahl eines rekurrenten neuronalen Netzes, dieses in keras implementieren und trainieren und
  - bei schlechten Validierungsergebnissen Lösungsstrategien zur Verbesserung der Ergebnisse anwenden können,
- bei Wahl eines Transformer Netzes, dieses mit Bedacht nutzen können,

um später eigene Deep Learning Projekte zur Zeitreihenanalyse, Textanalyse und -generierung, … umsetzen zu können.

Überprüfung des Lernziels: S. fortlaufende Übung in diesem Lernraum II.

Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik
Technology Arts Sciences TH Köln

Deep Learning
Ausblick



Rekurrente neuronale Netze (RNN)

- Einführung
  - Anwendungen mit Sequenzen
- Aufbau eines RNNs
- Übung: Vorhersage der Auslastung des ÖPNV am nächsten Tag
- Long Short-Term Memory (LSTM)
- Übung: Schreiben wie Shakespeare

Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Quelle: Patrick S., Hahn-Schickard

Rekurrentes Neuronales Netz - RNN

Bisher:

- FCN (fully-connected network) und CNN haben kein Gedächtnis
- Vorhersage für einen Datenpunkt hängt nicht von vorher gesehenen Datenpunkten ab
- Reihenfolge in denen neuronales Netz Datenpunkte sieht, spielt keine Rolle
- Länge der Ein- und Ausgabe des Netzes sind konstant und vordefiniert

- Bei rekurrenten neuronalen Netzen (RNN) gilt das alles nicht

- RNNs eignen sich für Sequenzen
  - Sequenz: Daten, bei der die Reihenfolge eine Rolle spielt (bspw. zeitliche Reihenfolge)
  - Bsp.:
    - Messwerte über Zeit, Text, Sprache, Höhenangaben in einem GPS-Track, …

Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik
Technology Arts Sciences TH Köln

Rekurrentes Neuronales Netz - RNN

Anwendungen mit Sequenzen

- Sentimentanalyse:
  „Der Film war gut“ → positive Bewertung
- Part-of-Speech-Tagging:
  „Der Hund läuft“ → Artikel, Nomen, Verb
- Zeitreihenprognose:
  Aus einer Folge vergangener Werte wird fortlaufend ein nächster Wert oder ein paralleler Ausgabewert berechnet.
  Beispiel:
  - Sensorwerte → zukünftige Sensorwerte
  - Umsatz der Eisdiele
- Bildbeschreibung (Image Captioning)
  Bild → „Ein Hund spielt im Park.“
- Spracherkennung
  Audiosignale bzw. Folgen akustischer Merkmale werden in Text transkribiert.

Rekurrentes Neuronales Netz - RNN

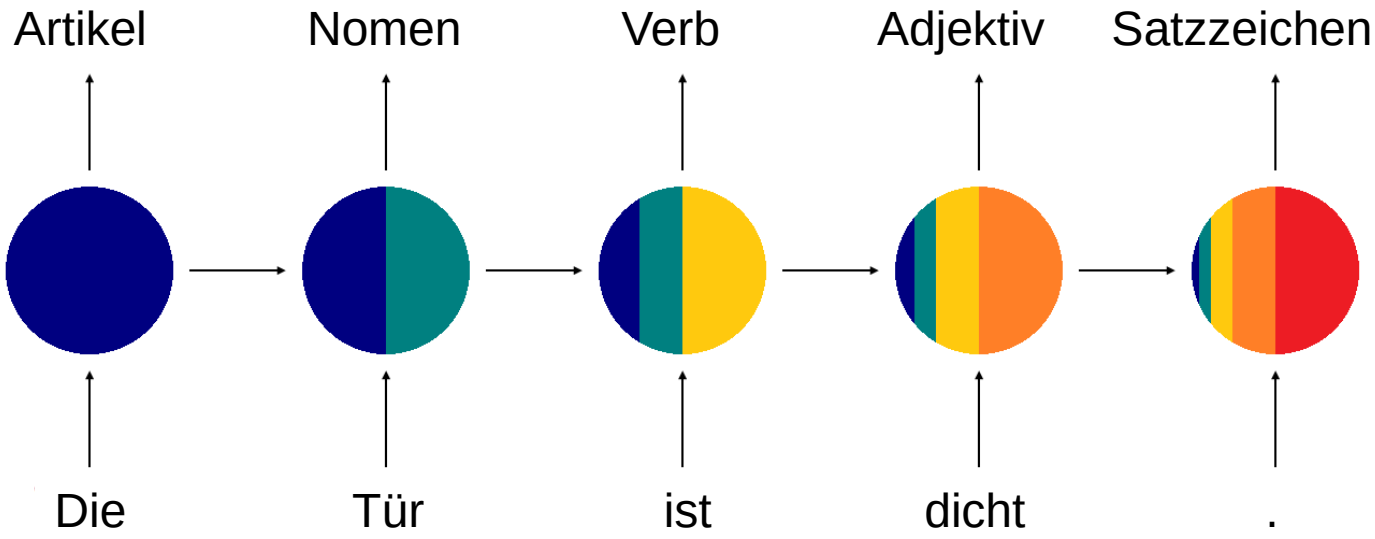
Beispiel: Part-of-Speech-Tagging

Ausgabe

- Zeitschritt 1: Artikel
- Zeitschritt 2: Nomen
- …

Neuronales Netz:

- Kreis repräsentiert das vollständige Netz zu unterschiedlichen Zeitschritten



- RNN wird genutzt, da es beliebig lange Sätze verarbeiten kann.
- Wortarten hängen vom Kontext ab: „Die Leute standen **dicht** beieinander.“ → dicht: Adverb

Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik
Quelle: Patrick S., Hahn-Schickard

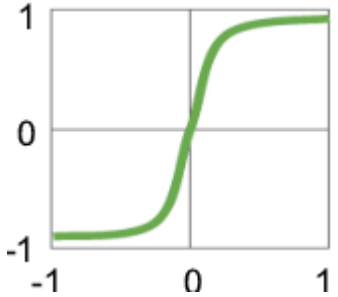
Rekurrentes Neuronales Netz – RNN

Ein rekurrentes Neuron

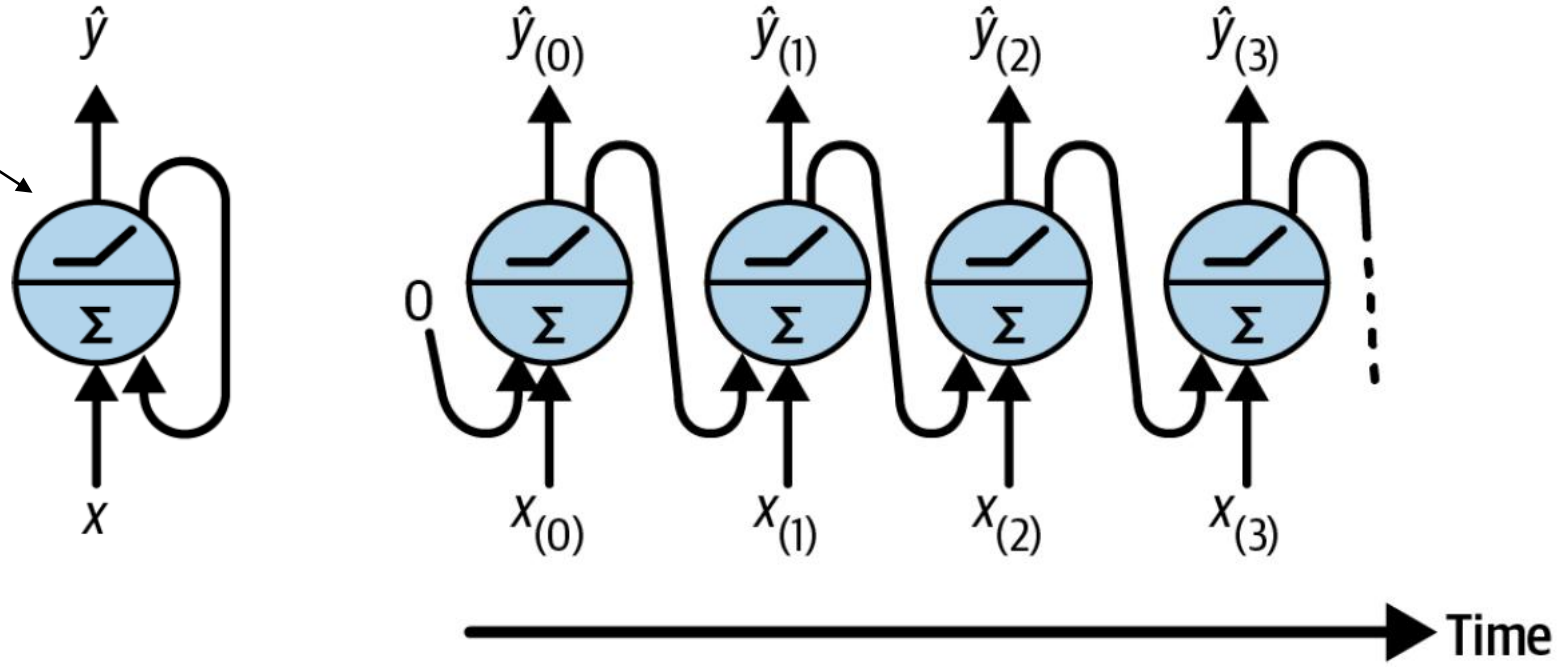
- Prädiktion zum vorherigen Zeitschritt $\hat{y}_{(t-1)}$ wird als zusätzlicher Input des nachfolgenden Zeitschritts t genutzt
- Visualisierung als abgerolltes/entfaltetes Neuron über die Zeit (unfolding/unrolling)

Aktivierungsfunktion:
Default: tanh

Tangens hyperbolicus



$$f(z) = \tanh(z)$$



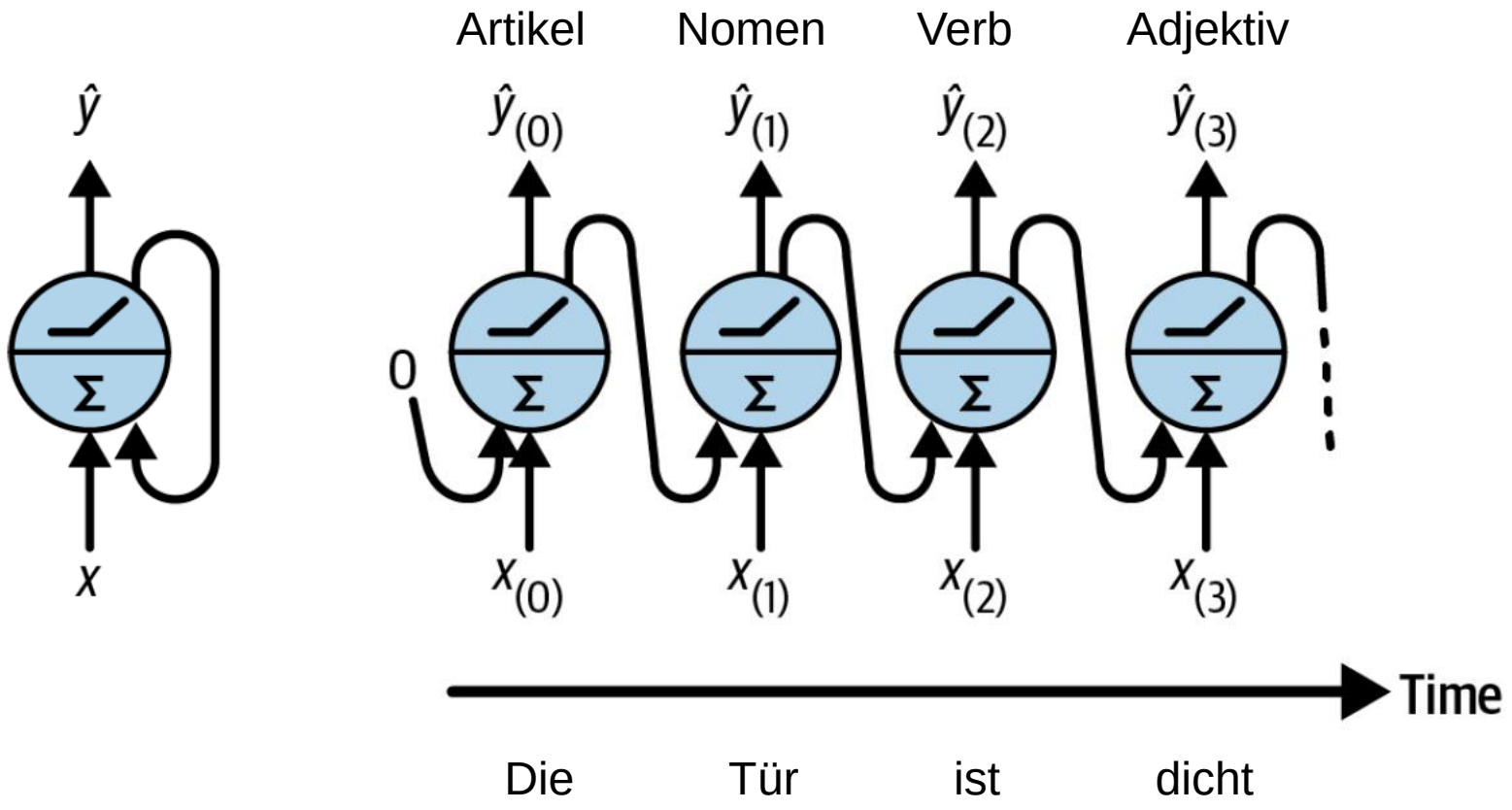
Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Bildquelle: A Géron, Hands-On Machine Learning with scikit-learn, keras and Tensorflow, 3rd edition, 2022.

Rekurrentes Neuronales Netz – RNN

Ein rekurrentes Neuron

Für Part-of-Speech-Tagging ist das Neuron zu einfach aufgebaut, da zum Zeitpunkt t=3 die Vorhersage nur „Verb“ ist und damit das Modell bei „dicht“ nicht auf Adjektiv schließen kann.



Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Bildquelle: A Géron, Hands-On Machine Learning with scikit-learn, keras and Tensorflow, 3rd edition, 2022.

Rekurrentes Neuronales Netz – RNN

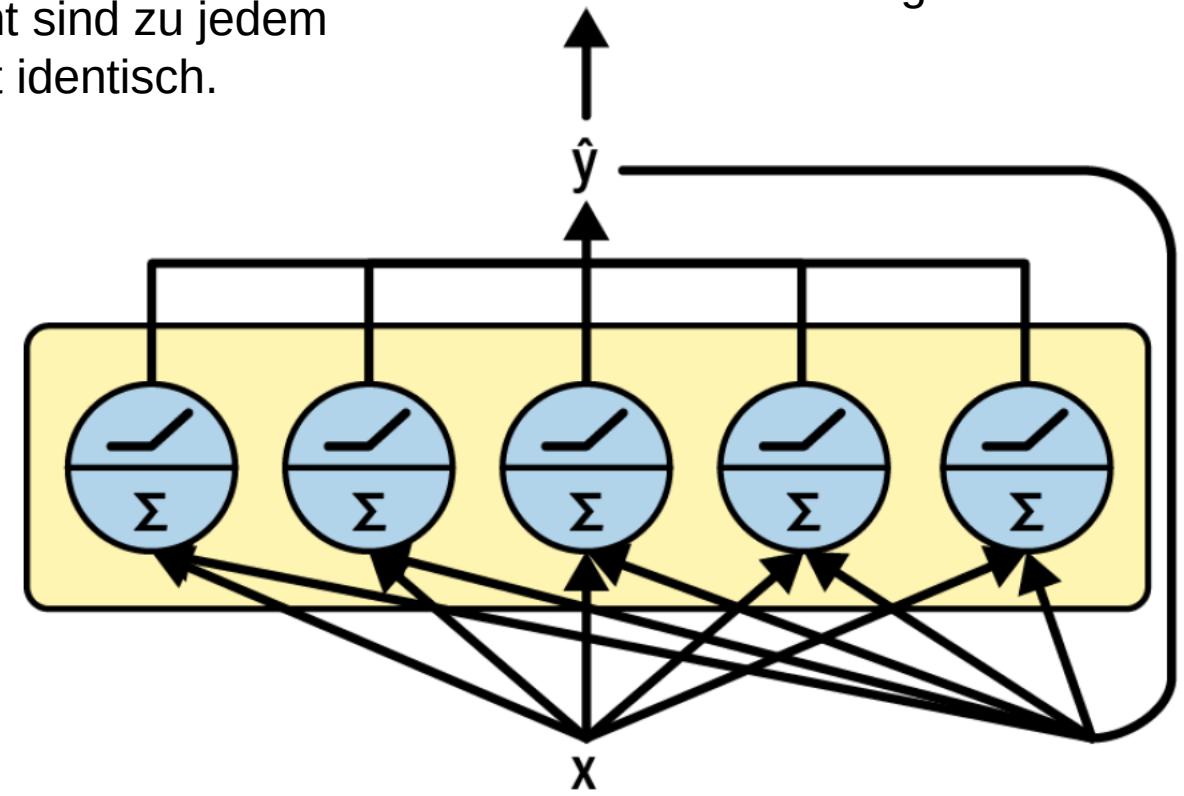
Ein Layer von rekurrenten Neuronen

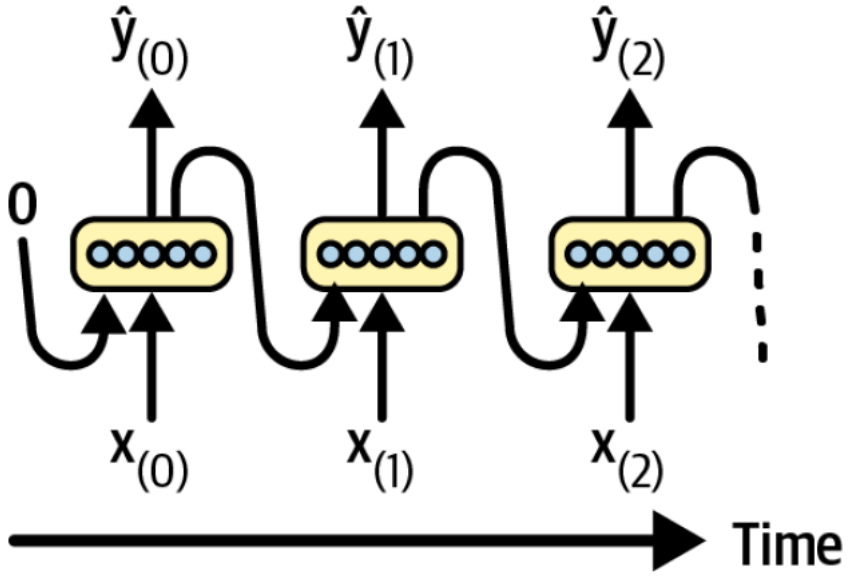
b ist der Bias
φ ist die Aktivierungsfunktion

Gewichtsmatrizen $\mathbf{W}_x$ und $\mathbf{W}_y$ der Schicht sind zu jedem Zeitpunkt t identisch.

Gleichung der Schicht:

$$\hat{\mathbf{y}}_{(t)} = \varphi\left(\mathbf{W}_x{}^{\top}\mathbf{x}_{(t)} + \mathbf{W}_{\hat{y}}{}^{\top}\hat{\mathbf{y}}_{(t-1)} + \mathbf{b}\right)$$



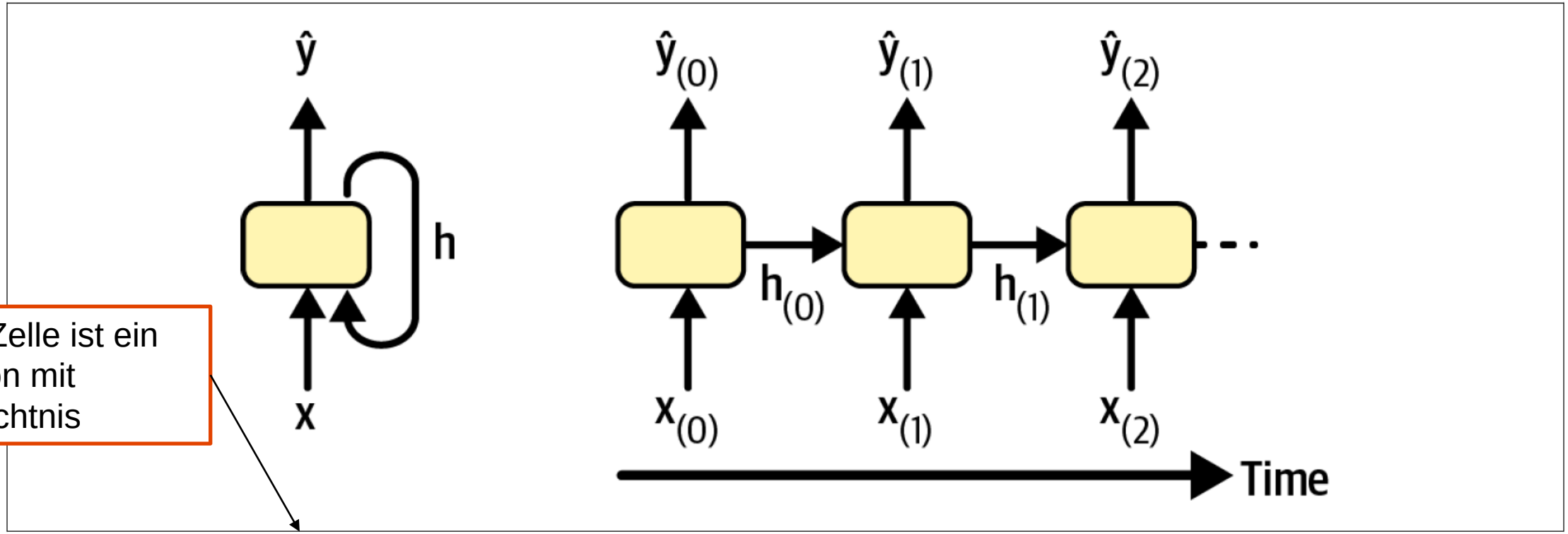


Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Bildquelle: A Géron, Hands-On Machine Learning with scikit-learn, keras and Tensorflow, 3rd edition, 2022.

Rekurrentes Neuronales Netz – RNN

Rekurrentes Neuron mit einem Speicher (Memory) – Bezeichnet als Zelle



Eine Zelle ist ein Neuron mit Gedächtnis

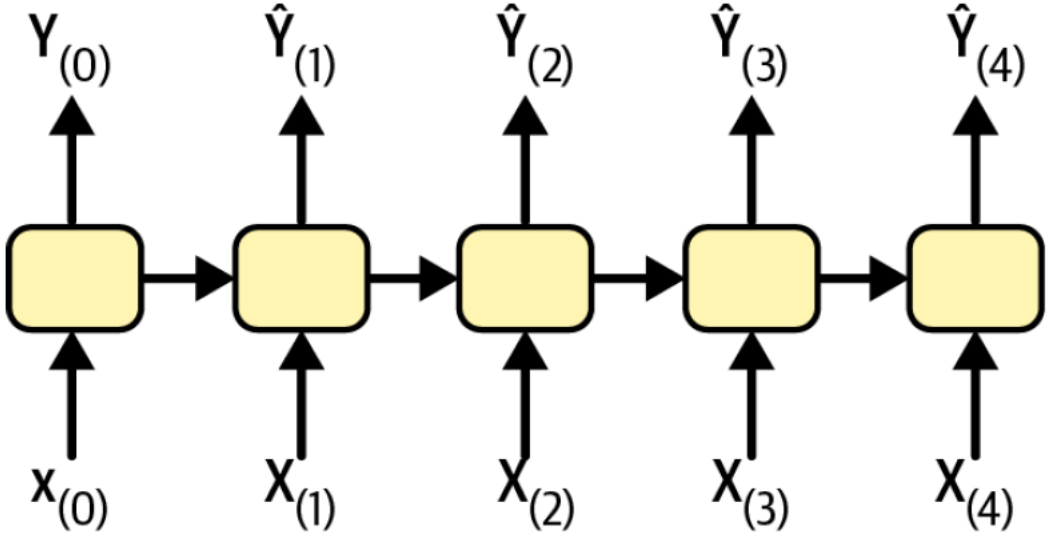
Figure 15-3. A cell's hidden state and its output may be different

Bildquelle: A Géron, Hands-On Machine Learning with scikit-learn, keras and Tensorflow, 3rd edition, 2022.

Rekurrentes Neuronales Netz – RNN

Sequence-to-Sequence: Eingang und Ausgang sind beides Sequenzen

- Das RNN erhält zu jedem Zeitpunkt t einen neuen Input und liefert zu jedem Zeitpunkt eine neue Vorhersage
- Input und Output haben die gleiche Länge
- Anwendungen:
  - Part-of-Speech-Tagging:
    „Der Hund läuft“ → Artikel, Nomen, Verb
  - Zeitreihenprognose:
    Aus einer Folge vergangener Werte wird fortlaufend (ein nächster Wert oder) ein paralleler Ausgabewert berechnet.
    Beispiel:
    - (Sensorwerte → zukünftige Sensorwerte)
    - Konzentration geg. Temp., CO_2-Konz., Zeit



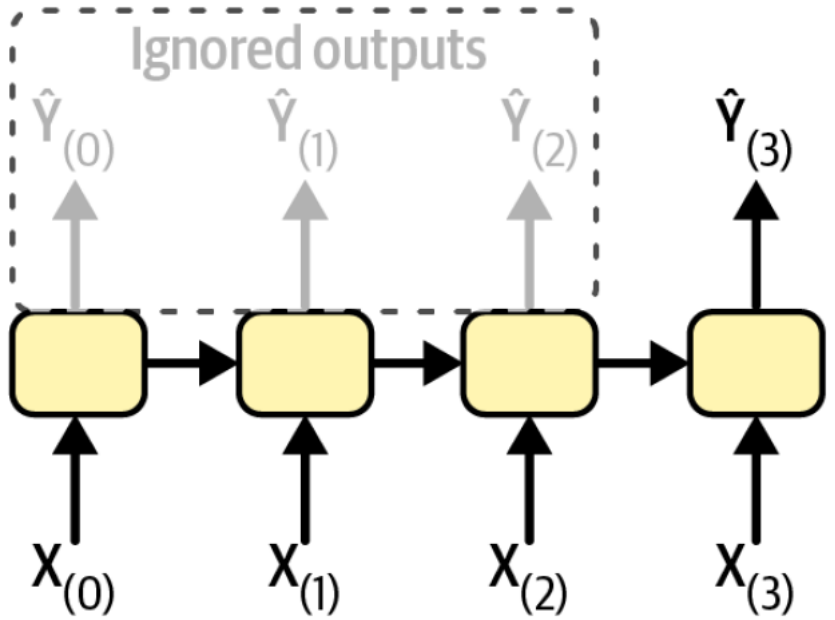
Warum das geht, sehen wir später in Übung 1

Bildquelle: A Géron, Hands-On Machine Learning with scikit-learn, keras and Tensorflow, 3rd edition, 2022.

Rekurrentes Neuronales Netz – RNN

Sequence-to-Vector: Eingang ist Sequenz und Ausgang ein Vektor oder Skalar

- Das RNN erhält eine Sequenz als Input und liefert am Ende der Sequenz eine Vorhersage
- Anwendungen:
  - Sentimentanalyse:
    „Der Film war gut“ → positive Bewertung
  - Spam-Erkennung
    Eine komplette E-Mail oder Nachricht wird als Spam oder Nicht-Spam eingestuft.
  - Aktivitätserkennung aus Sensordaten
    Beschleunigungssensor-Daten → Gehen, Laufen, Sitzen
  - Musikgenre-Klassifikation
  - Krankheitsdiagnose aus Zeitreihen (EKG, EEG)



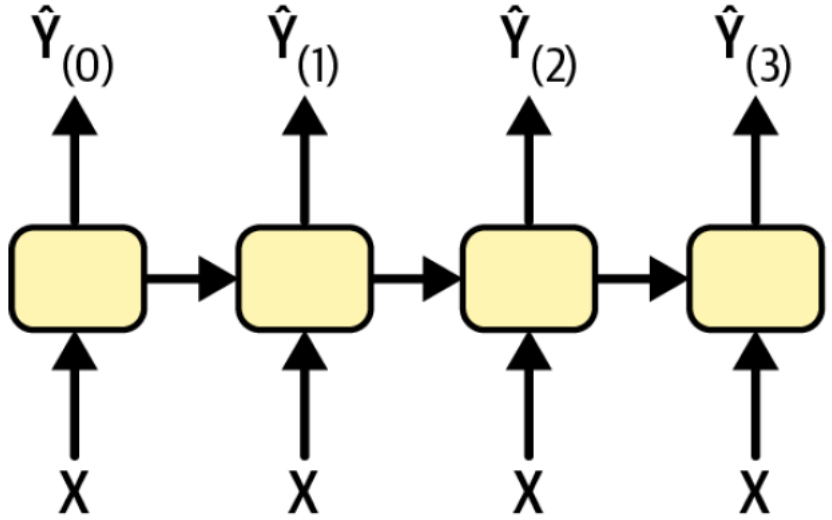
Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Bildquelle: A Géron, Hands-On Machine Learning with scikit-learn, keras and Tensorflow, 3rd edition, 2022.

Rekurrentes Neuronales Netz – RNN

Vector-to-Sequence: Eingang ist Vektor oder Skalar und Ausgang eine Sequenz

- Das RNN erhält immer den gleichen Input und liefert zu jedem Zeitpunkt eine Vorhersage
- Anwendungen:
  - Bildbeschreibung (Image Captioning)
    Bild → „Ein Hund spielt im Park.“
  - Musikgenerierung
    Aus einem Startvektor oder Stilvektor wird eine Folge von Noten erzeugt.
  - Textgenerierung zu einer Kategorie oder einem Thema
    - Ein einzelner Eingabevektor beschreibt ein Thema, daraus wird Text erzeugt.
    - Thema „Sport“ → generierter Artikelanfang



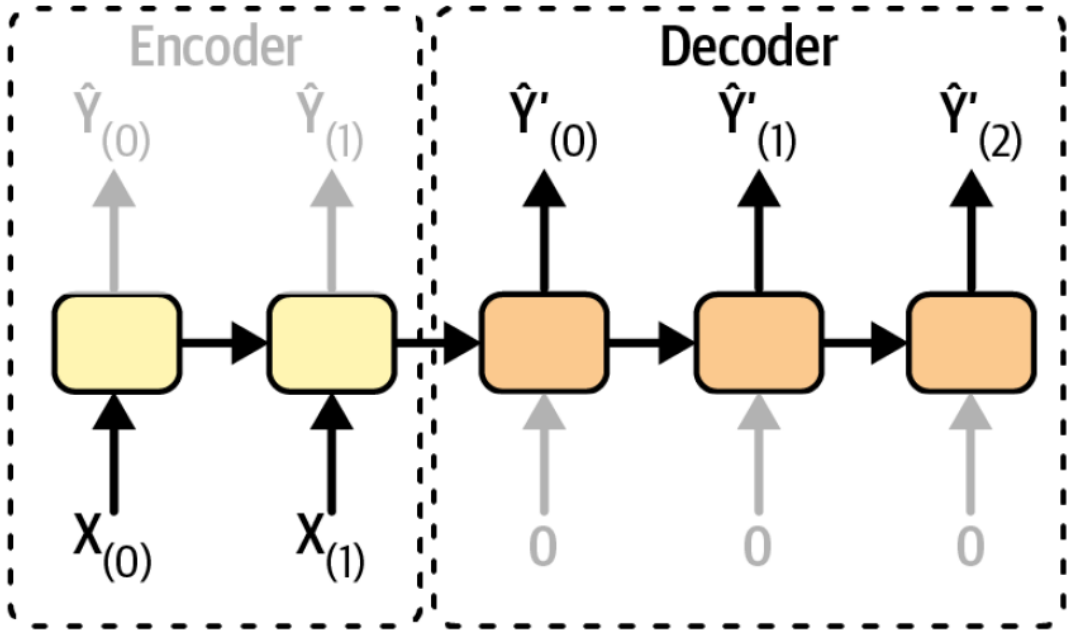
Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Bildquelle: A Géron, Hands-On Machine Learning with scikit-learn, keras and Tensorflow, 3rd edition, 2022.

Rekurrentes Neuronales Netz – RNN

Encoder-Decoder: Eingang ist Sequenz und Ausgang ist eine Sequenz

- Das RNN encodiert Eingabesequenz in einen Vektor und decodiert diesen in eine Sequenz
- Anwendungen:
  - Maschinelle Übersetzung
    „Ich heiße Daniel“ → “My name is Daniel”
  - Textzusammenfassung
    Beispiel: Nachrichtenartikel wird als Vektor gespeichert → Kurzfassung wird daraus generiert
  - Spracherkennung
    Audiosignale bzw. Folgen akustischer Merkmale werden in Text transkribiert.



Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Bildquelle: A Géron, Hands-On Machine Learning with scikit-learn, keras and Tensorflow, 3rd edition, 2022.

Rekurrentes Neuronales Netz – RNN

Übersicht über die verschiedenen Eingangs- und Ausgangssequenzen

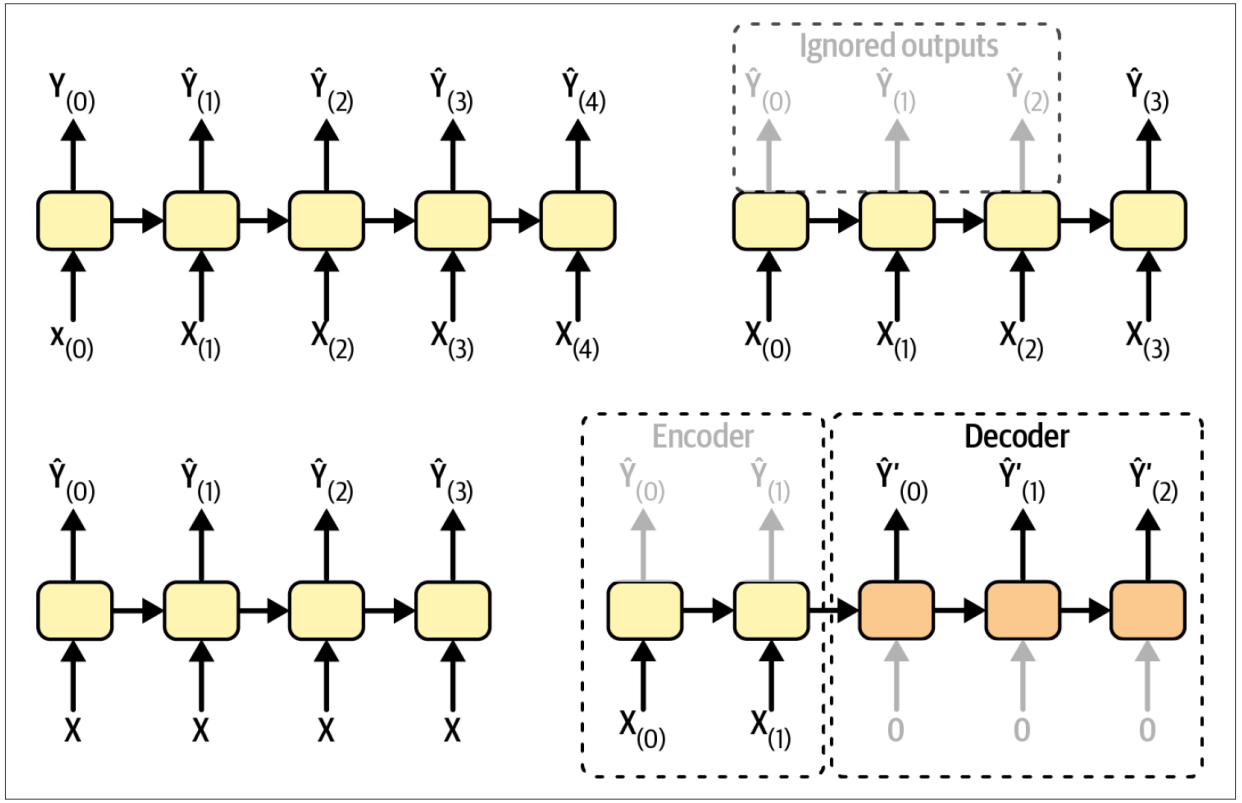


Figure 15-4. Sequence-to-sequence (top left), sequence-to-vector (top right), vector-to-sequence (bottom left), and encoder–decoder (bottom right) networks

Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Bildquelle: A Géron, Hands-On Machine Learning with scikit-learn, keras and Tensorflow, 3rd edition, 2022.

Rekurrentes Neuronales Netz – RNN

Übung zu den verschiedenen Eingangs- und Ausgangssequenzen

- **Named Entity Recognition (NER)**
  Für Wort wird vorhergesagt, ob es ein Personenname, Ort oder Datum ist.
  Beispiel: „Angela Merkel besucht Berlin“ → Person, Person, O, Ort
- **Wochentag → Terminplan:** Für Wochentag Aktivitäten erzeugen. Beispiel: „Montag“ → Aufstehen, Vorlesung, Mittag, Lernen, Schlafen.
- Aktienkurs → „Kurs steigt“ oder „Kurs fällt“.
- Ziffer (0–9) → ausgeschriebenes Wort, Zeichen für Zeichen („drei“, „acht“, ...).
- Zeichenfolge → Groß-/Kleinschreibung pro Zeichen bestimmen.
- Chatbot

Welche Architektur benötigen wir?

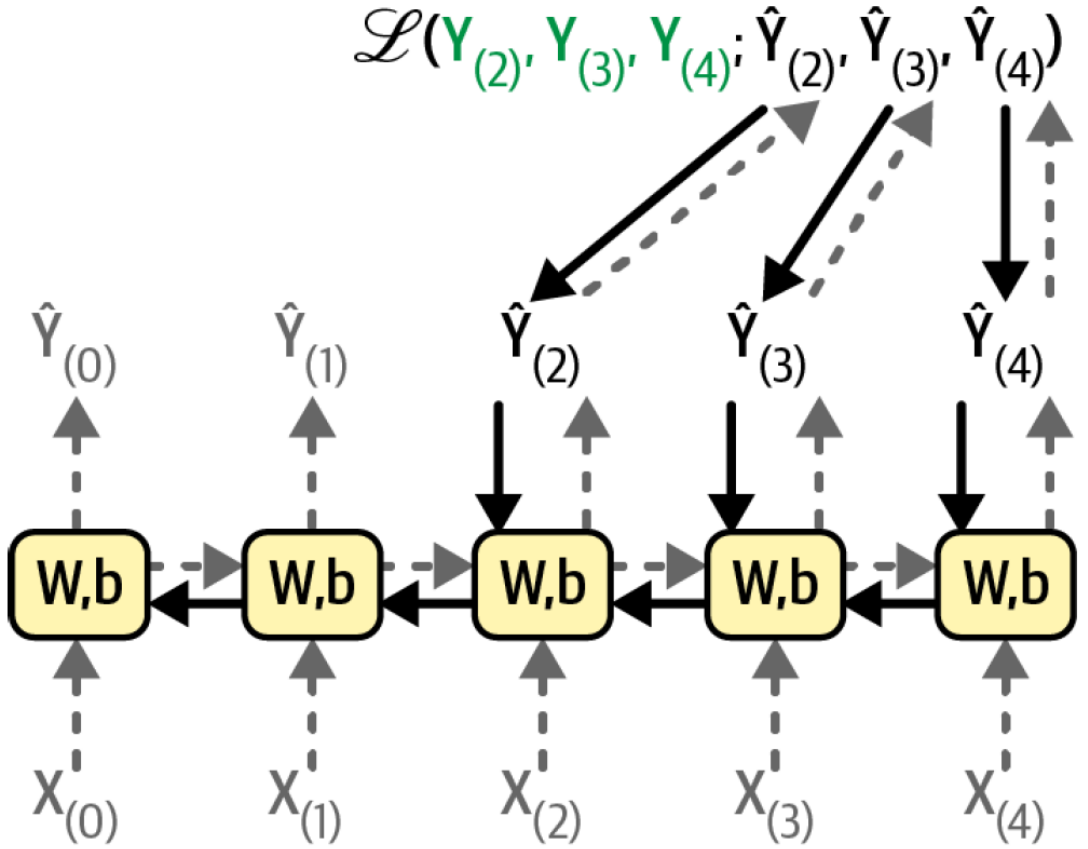
- sequence-to-sequence
- sequence-to-vector
- vector-to-sequence
- encoder-decoder

Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik
Technology Arts Sciences TH Köln

Rekurrentes Neuronales Netz – RNN

Training: Backpropagation through time

- Kostenfunktion versucht Fehler zwischen Vorhersagen und Labels zu verschiedenen Zeitpunkten zu minimieren
- Gradienten der Kostenfunktion werden durch das entfaltete RNN rückwärts propagiert
  - Durch das RNN und über die Zeit



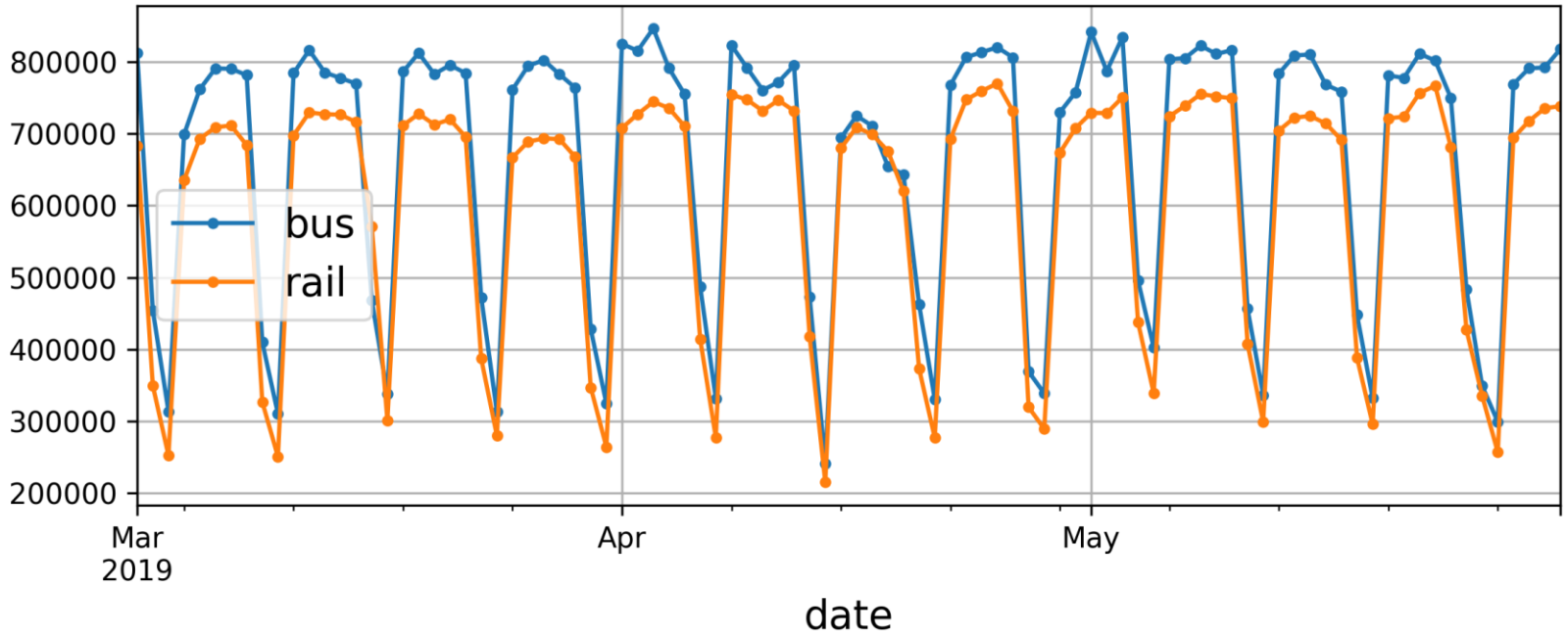
Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Bildquelle: A Géron, Hands-On Machine Learning with scikit-learn, keras and Tensorflow, 3rd edition, 2022.

Rekurrentes Neuronales Netz – RNN

Übung: Vorhersage der Auslastung des ÖPNV am nächsten Tag

- Gegeben: Anzahl der Passagiere, die täglich Bus und Bahn in Chicago fahren
- Ziel: Vorhersage der Passagierzahlen für den nächsten Tag, gegeben die Daten bis zum vorherigen Tag



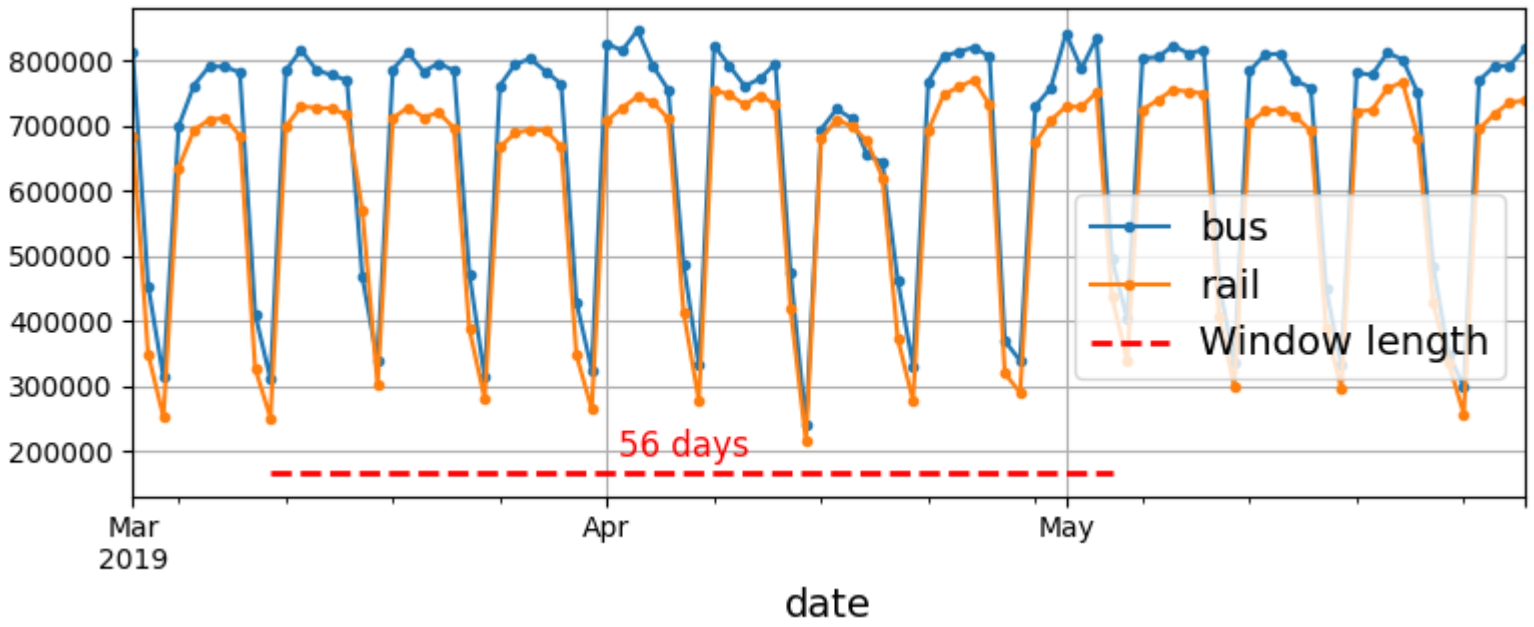
Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Bildquelle: A Géron, Hands-On Machine Learning with scikit-learn, keras and Tensorflow, 3rd edition, 2022.

Rekurrentes Neuronales Netz – RNN

Übung: Vorhersage der Auslastung des ÖPNV am nächsten Tag

- Daten werden in „Windows“ von 56 Tagen eingeteilt
  - Ein Datenpunkt enthält Passagierdaten über 56 Tage
  - Passagierzahl am 57. Tag ist das Label
- „Window“ wird über den gesamten Trainingsdatensatz geschoben um ganz viele Trainingsdatenpunkte zu generieren



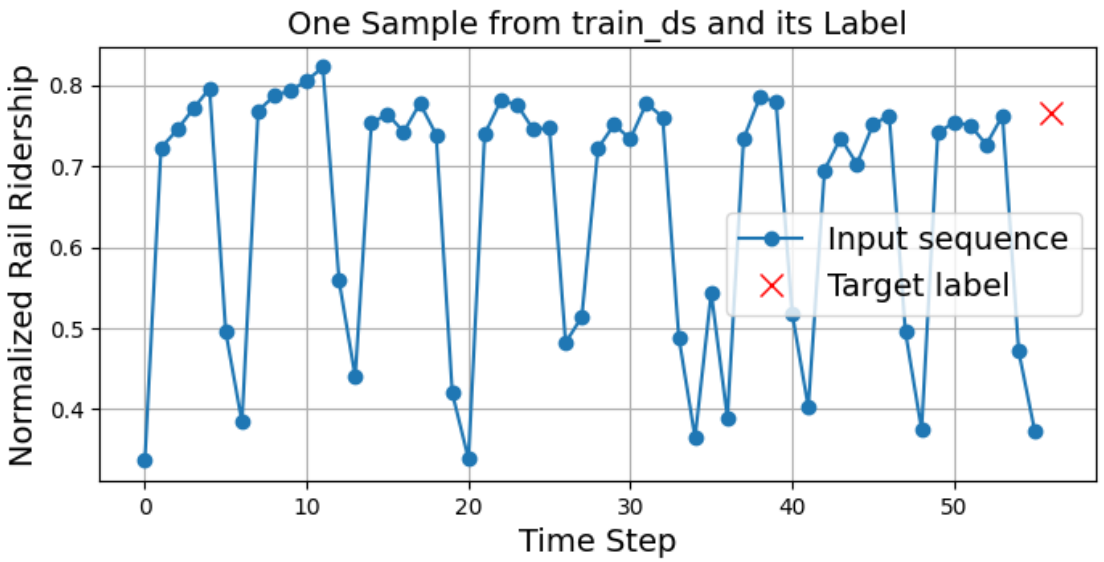
Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

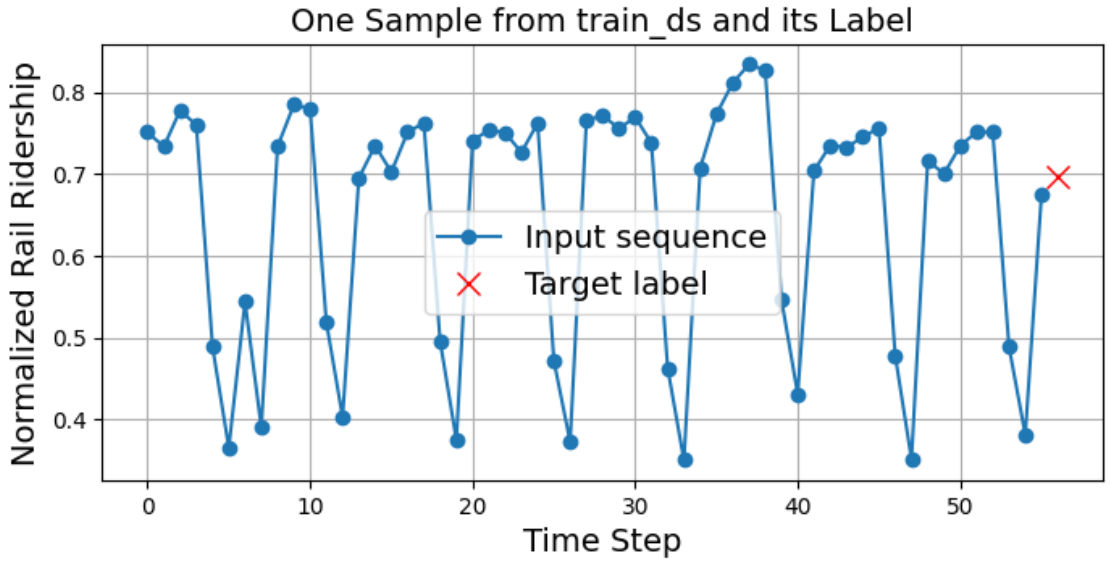
Bildquelle: A Géron, Hands-On Machine Learning with scikit-learn, keras and Tensorflow, 3rd edition, 2022.

Rekurrentes Neuronales Netz – RNN

Übung: Vorhersage der Auslastung des ÖPNV am nächsten Tag

- Daten werden in „Windows“ von 56 Tagen eingeteilt
  - Ein Datenpunkt enthält Passagierdaten über 56 Tage
  - Passagierzahl am 57. Tag ist das Label





Bildquelle: A Géron, Hands-On Machine Learning with scikit-learn, keras and Tensorflow, 3rd edition, 2022.

Rekurrentes Neuronales Netz – RNN

Übung: Vorhersage der Auslastung des ÖPNV am nächsten Tag

Welche Architektur benötigen wir?

- sequence-to-sequence
- sequence-to-vector
- vector-to-sequence
- encoder-decoder

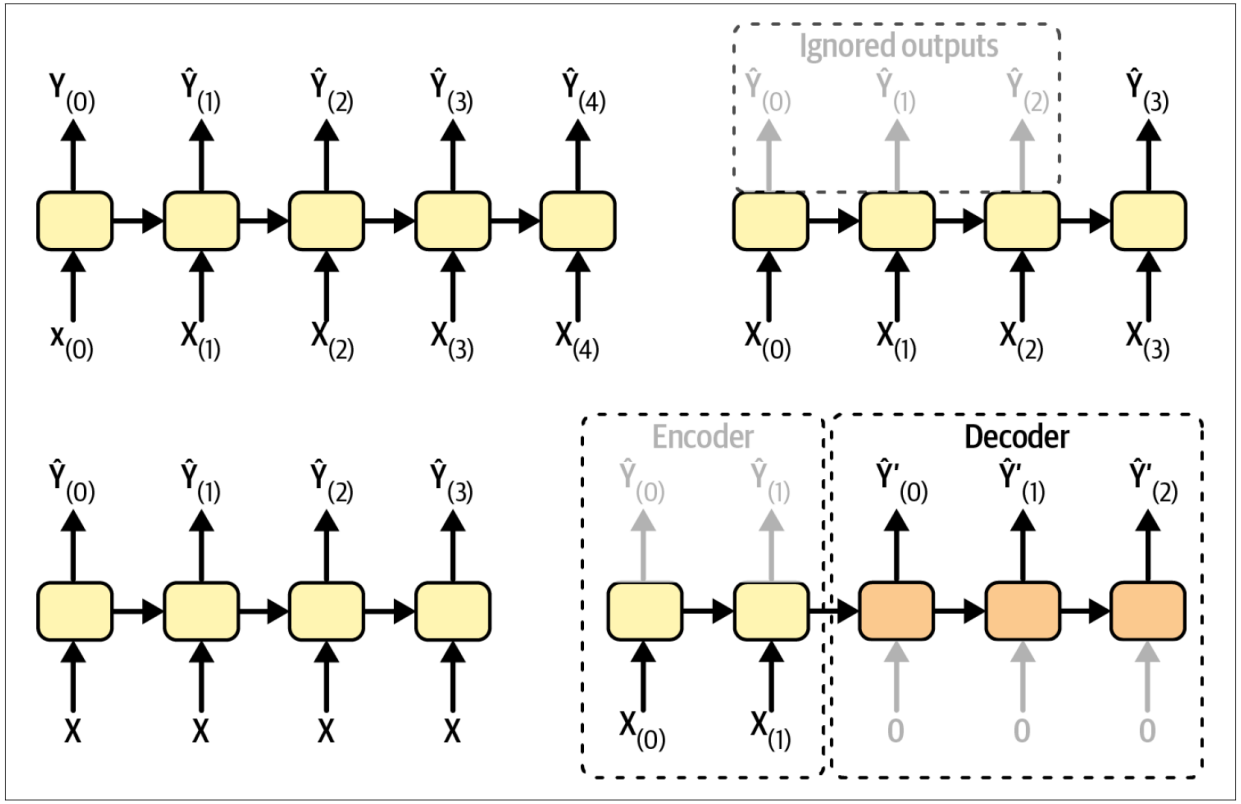


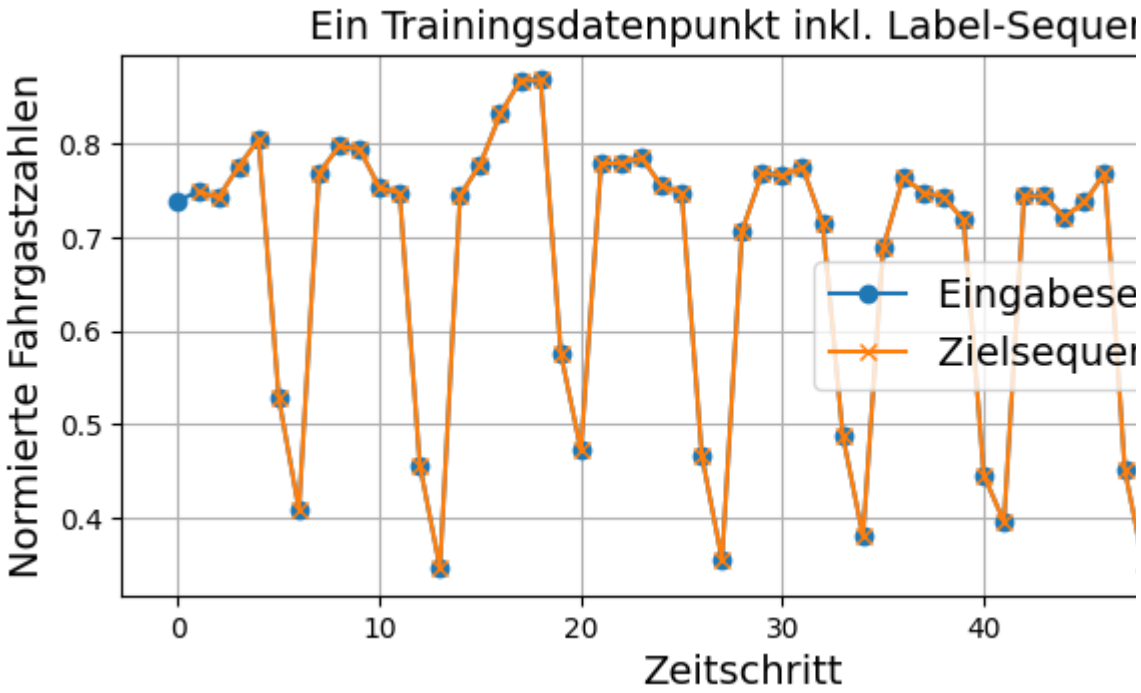
Figure 15-4. Sequence-to-sequence (top left), sequence-to-vector (top right), vector-to-sequence (bottom left), and encoder–decoder (bottom right) networks

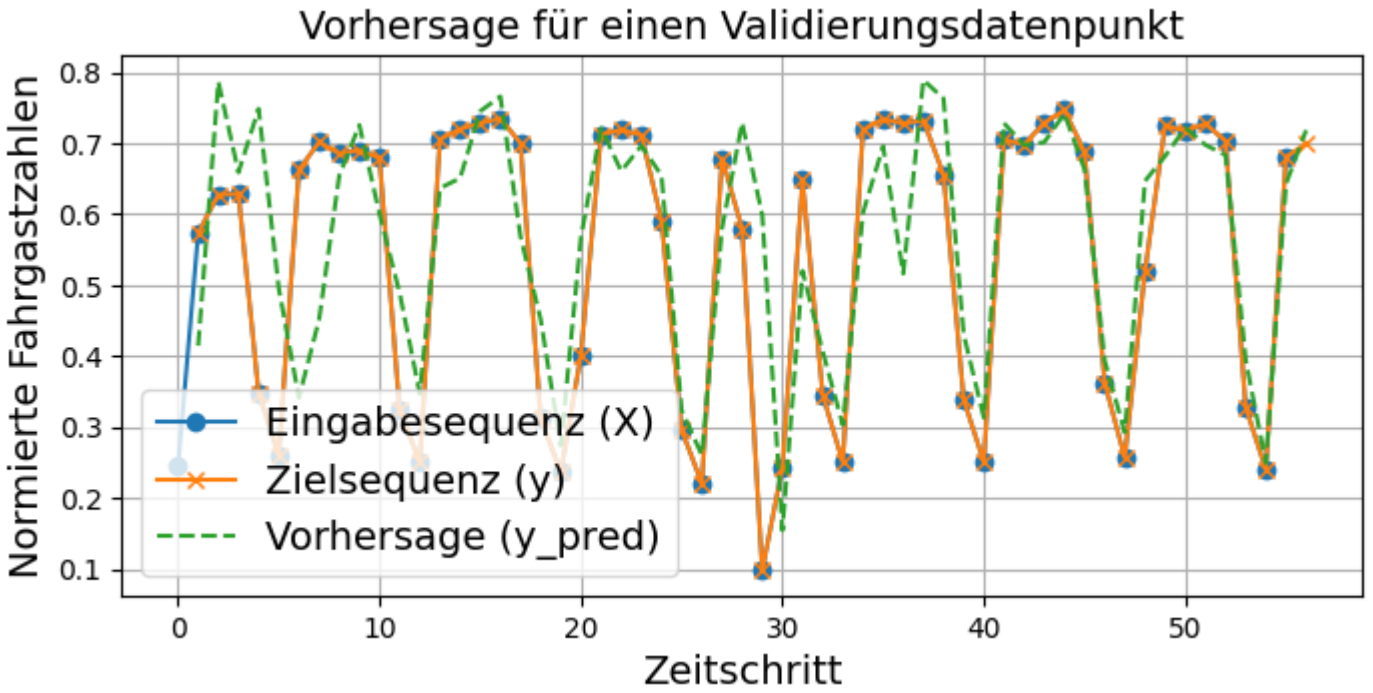
Bildquelle: A Géron, Hands-On Machine Learning with scikit-learn, keras and Tensorflow, 3rd edition, 2022.

Rekurrentes Neuronales Netz – RNN

Übung: Vorhersage der Auslastung des ÖPNV am nächsten Tag

Es geht auch als sequence-to-sequence.





Rekurrentes Neuronales Netz – RNN

Übung: Vorhersage der Auslastung des ÖPNV am nächsten Tag

Aufbau des RNN

- 1 RNN Layer mit 32 Neuronen
  - Länge des Inputs ist egal (None) aber eindimensional (1)
- 1 Dense Layer mit einem Ausgangsneuron
  - Keine Aktivierungsfunktion
    - Regressionsproblem: es muss Werte zwischen 0 und 1.4 (da Daten skaliert werden) zurückliefern

```
univar_model = tf.keras.Sequential([
    tf.keras.layers.SimpleRNN(32, input_shape=[None, 1]),
    tf.keras.layers.Dense(1)  # no activation function by default
])
```

32 Neuronen

```
rail_train = df["rail"]["2016-01":"2018-12"] / 1e6
rail_valid = df["rail"]["2019-01":"2019-05"] / 1e6
rail_test = df["rail"]["2019-06":] / 1e6
```


Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

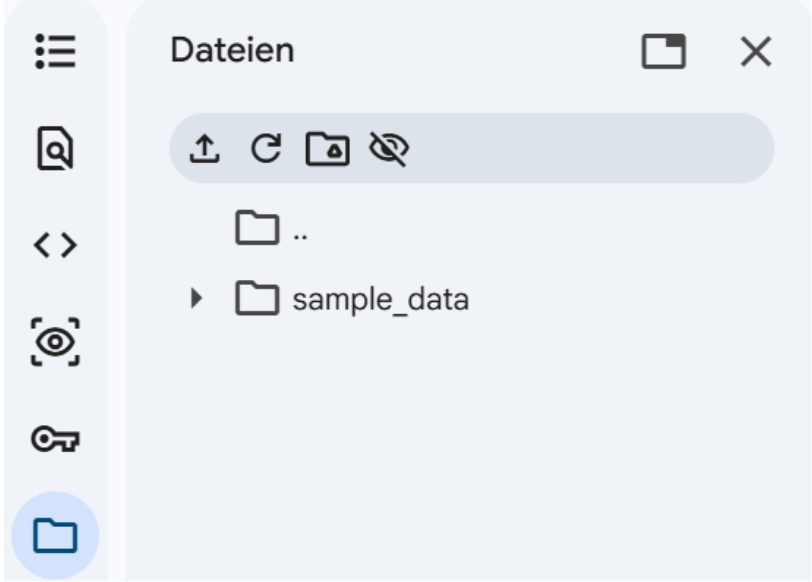
Bildquelle: A Géron, Hands-On Machine Learning with scikit-learn, keras and Tensorflow, 3rd edition, 2022.

Rekurrentes Neuronales Netz – RNN

Übung: Vorhersage der Auslastung des ÖPNV am nächsten Tag

- Führen Sie das Notebook 01_sequences_rnns.ipynb aus
  - GPU benötigt
  - Alternative: Trainiertes Modell von github (univar_rnn_model_s2v.keras) herunterladen und in Google Colab hochladen.

- Beantworten Sie folgende Fragen:
  - Wie erfolgt die Aufteilung in Training-, Validierung- und Testdaten?
  - Welche Regularisierungsmethode wird genutzt?
  - Welche Metrik wird zur Validierung genutzt?
  - Was ist die Huber Kostenfunktion?

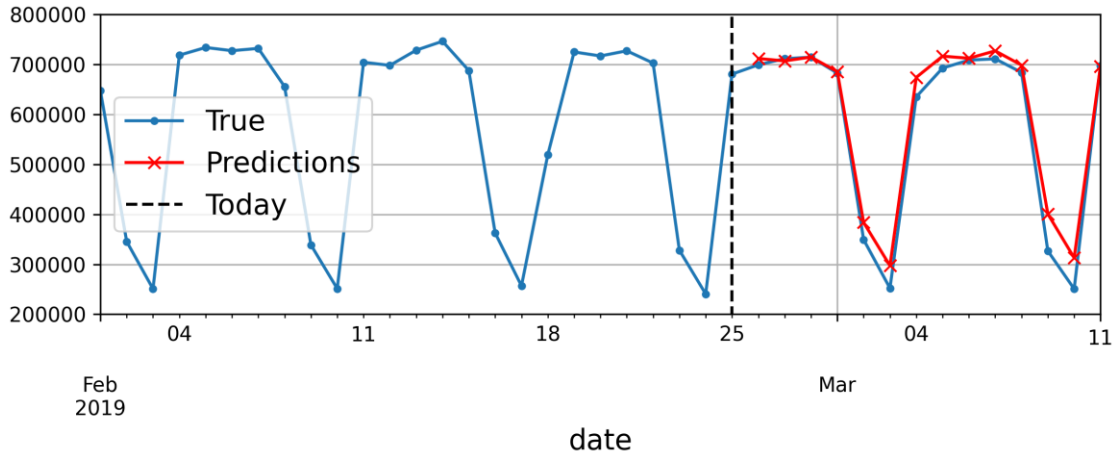


Rekurrentes Neuronales Netz – RNN

Übung: Vorhersage der Auslastung des ÖPNV am nächsten Tag

Vorhersage von mehr als einem Tag in die Zukunft:

- Vorhersage an Tag t+1 wird als Input für Tag t+1 genutzt: x = [$x_{(t-55,t)}$, $\hat{y}_{(t+1)}$]
- Die sich ergebende Vorhersage am Tag t+2 wird wieder als Input für Tag t+2 genutzt.
  x = [$x_{(t-55,t)}$, $\hat{y}_{(t+1)}$, $\hat{y}_{(t+2)}$]
- …




```
X = rail_valid.to_numpy()[np.newaxis, :seq_length, np.newaxis]
for step_ahead in range(14):
    y_pred_one = univar_model.predict(X)
    X = np.concatenate([X, y_pred_one.reshape(1, 1, 1)], axis=1)
```


Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Bildquelle: A Géron, Hands-On Machine Learning with scikit-learn, keras and Tensorflow, 3rd edition, 2022.

Rekurrentes Neuronales Netz – RNN

Übung: Vorhersage der Auslastung des ÖPNV am nächsten Tag

Forecasting Several Steps Ahead

```
import numpy as np

X = rail_valid.to_numpy()[np.newaxis, :seq_length, np.newaxis]
for step_ahead in range(14):
    y_pred_one = univar_model.predict(X)
    X = np.concatenate([X, y_pred_one.reshape(1, 1, 1)], axis=1)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 275ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 272ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 172ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 184ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 165ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 166ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 167ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 172ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 181ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 170ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 256ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 238ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 262ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 240ms/step
```

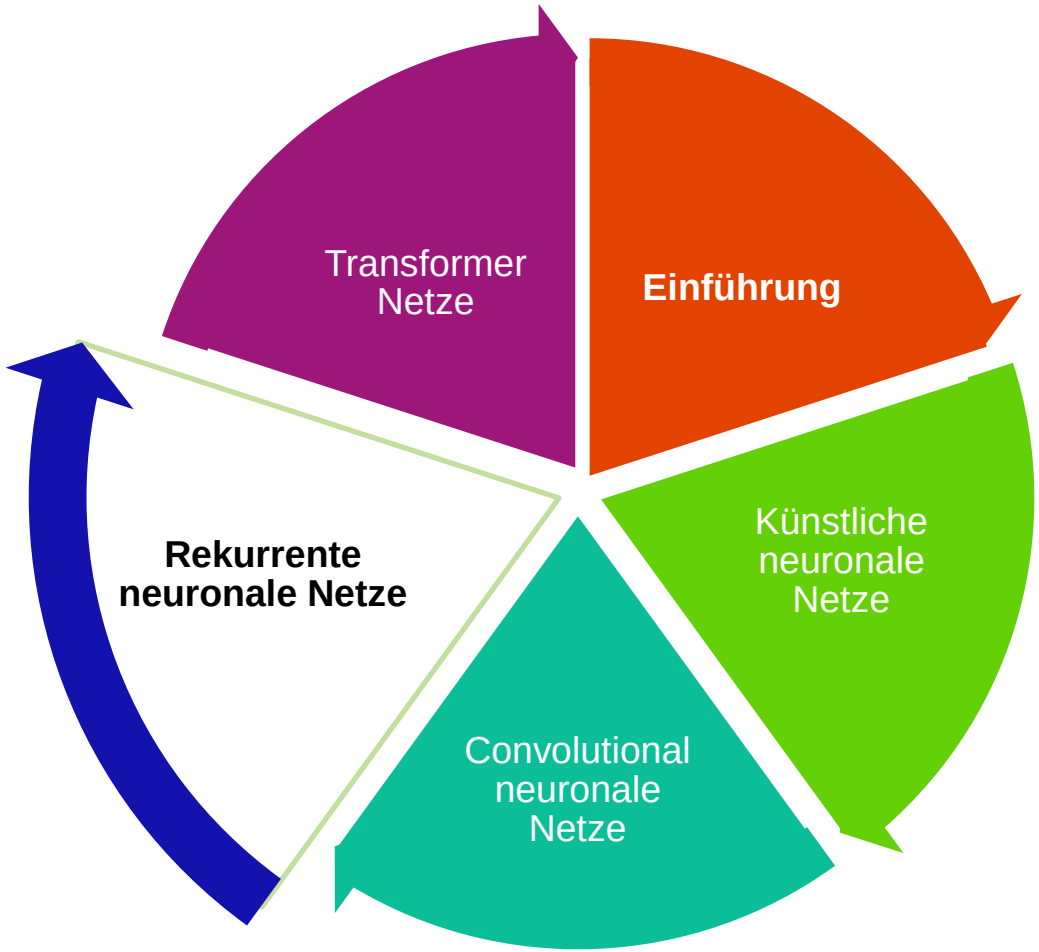
```
# extra code – generates and saves Figure 15–11

# The forecasts start on 2019-02-26, as it is the 57th day of 2019, and they end
# on 2019-03-11. That's 14 days in total.
Y_pred = pd.Series(X[0, -14:, 0],
                   index=pd.date_range("2019-02-26", "2019-03-11"))

fig, ax = plt.subplots(figsize=(8, 3.5))
(rail_valid * 1e6)["2019-02-01":"2019-03-11"].plot(
    label="True", marker=".", ax=ax)
(Y_pred * 1e6).plot(
    label="Predictions", grid=True, marker="x", color="r", ax=ax)
ax.vlines("2019-02-25", 0, 1e6, color="k", linestyle="--", label="Today")
ax.set_ylim([200_000, 800_000])
plt.legend(loc="center left")
save_fig("forecast_ahead_plot")
plt.show()
```


Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Deep Learning
Ausblick



Rekurrente neuronale Netze (RNN)

- Einführung
  - Anwendungen mit Sequenzen
- Aufbau eines RNNs
- Übung: Vorhersage der Auslastung des ÖPNV am nächsten Tag
- **Long Short-Term Memory (LSTM)**
- **Übung: Schreiben wie Shakespeare**

Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik
Quelle: Patrick S., Hahn-Schickard

Rekurrentes Neuronales Netz - RNN

Das Problem der langfristigen Abhängigkeit

Vervollständigen Sie den Satz:

Einfach:

- *Die Wolken sind am …*
  - *Himmel*

Schwierig:

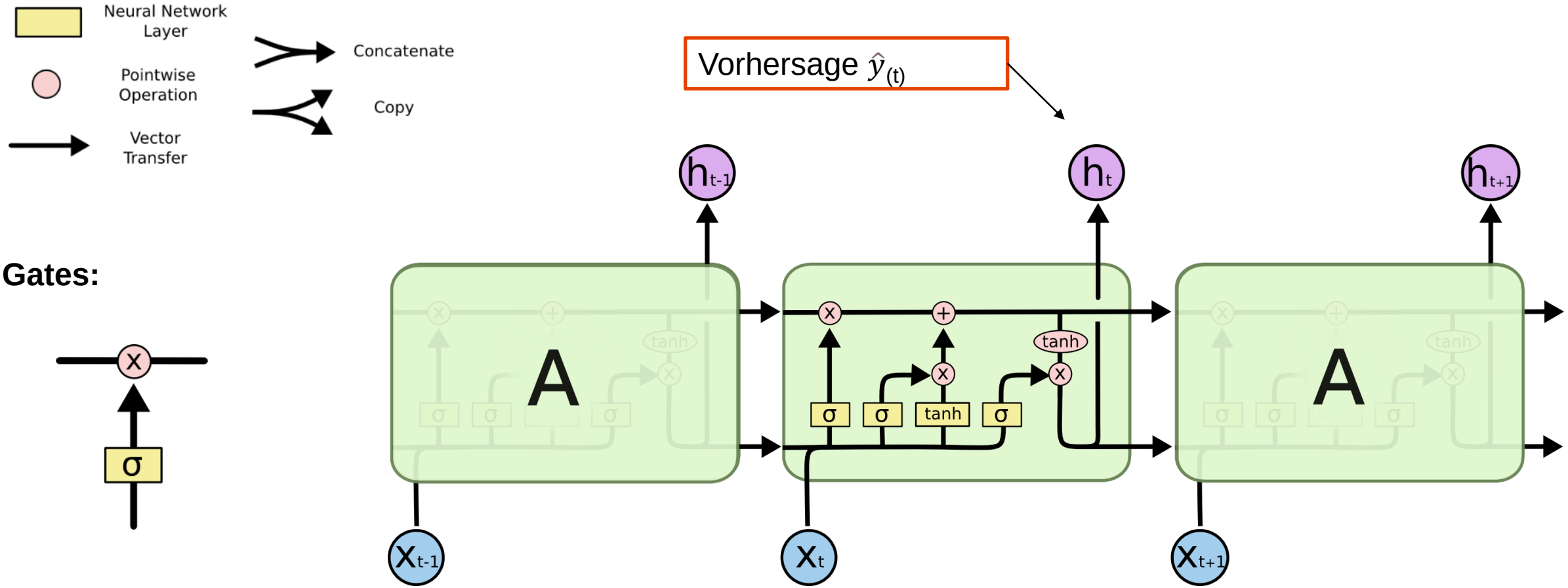
- *Ich bin in Frankreich aufgewachsen... Ich spreche fließend …*
  - *Französisch*

Ein komplexeres RNN, das LSTM, soll dieses Problem überwinden.

S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," in *Neural Computation*, vol. 9, no. 8, pp. 1735-1780, 15 Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

Rekurrentes Neuronales Netz - RNN

Long Short-Term Memory (LSTM)



Gates:

Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

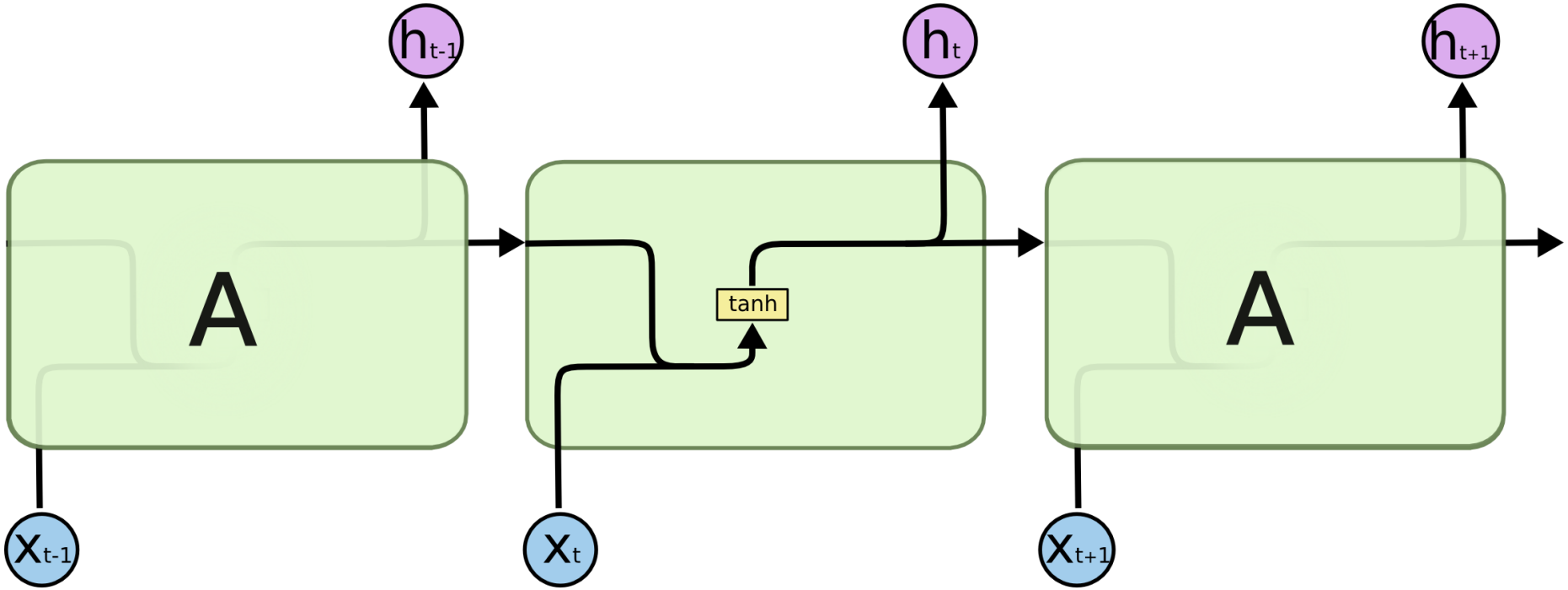
Quelle: Patrick S., Hahn-Schickard
Bilder:
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Rekurrentes Neuronales Netz - RNN

Einfache RNN Zelle – Wiederholung

Short-term state $\mathbf{h_t}$

- Kurzzeit-Gedächtnis; Skalar



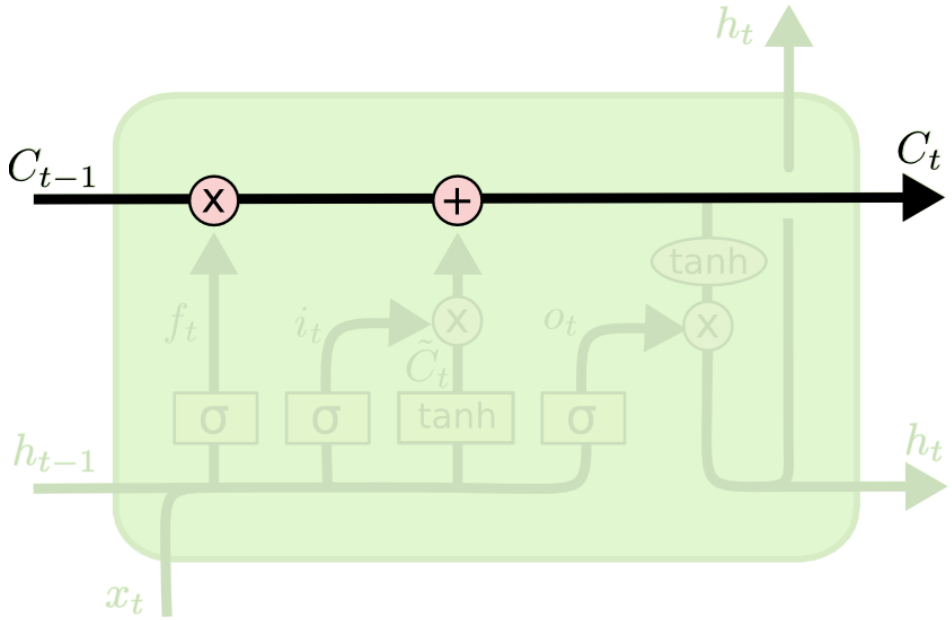
Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Bilder:
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Rekurrentes Neuronales Netz - RNN

LSTM – Der Zellzustand

Wir betrachten hier nur eine LSTM Zelle und keine Schicht von LSTM Zellen.



Long-term state C_t

- Langzeit-Gedächtnis; Skalar
- Hinzufügen von wichtigen Informationen
- Entfernen von nicht mehr wichtigen Informationen

Short-term state h_t

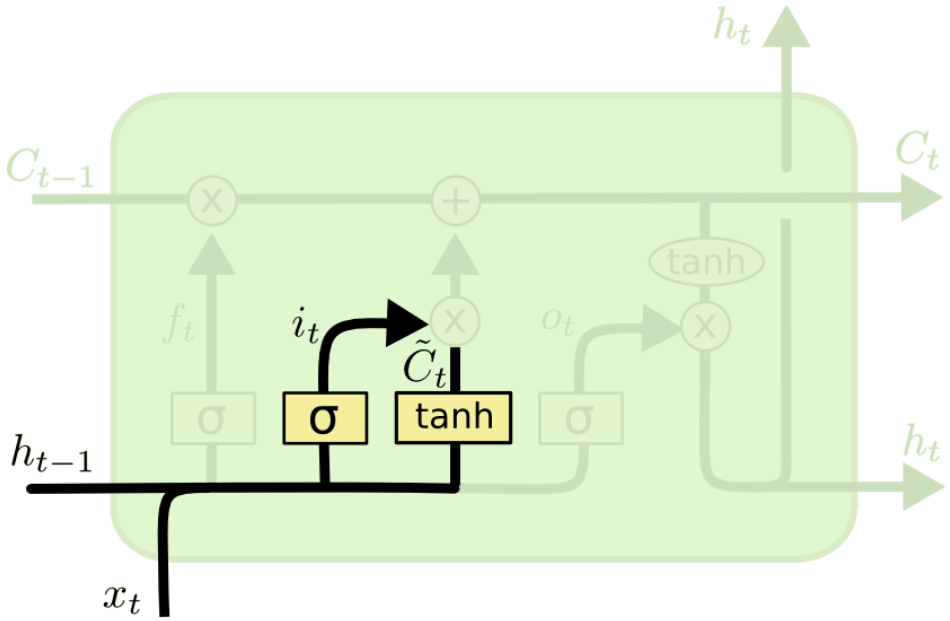
- Kurzzeit-Gedächtnis; Skalar
- Mehr oder weniger das was eine einfache RNN Zelle hatte

Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Quelle: Patrick S., Hahn-Schickard
Bilder:
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Rekurrentes Neuronales Netz - RNN

LSTM – Input Gate Layer



$tanh$ - Aktivierungsfunktion

- Das ist das was eine Zelle eines RNN macht und in hidden state h_t speichern würde
- In der LSTM Zelle wird hingegen geprüft ob diese Information zu dem long-term state C_t hinzugefügt werden soll

Input Gate Layer

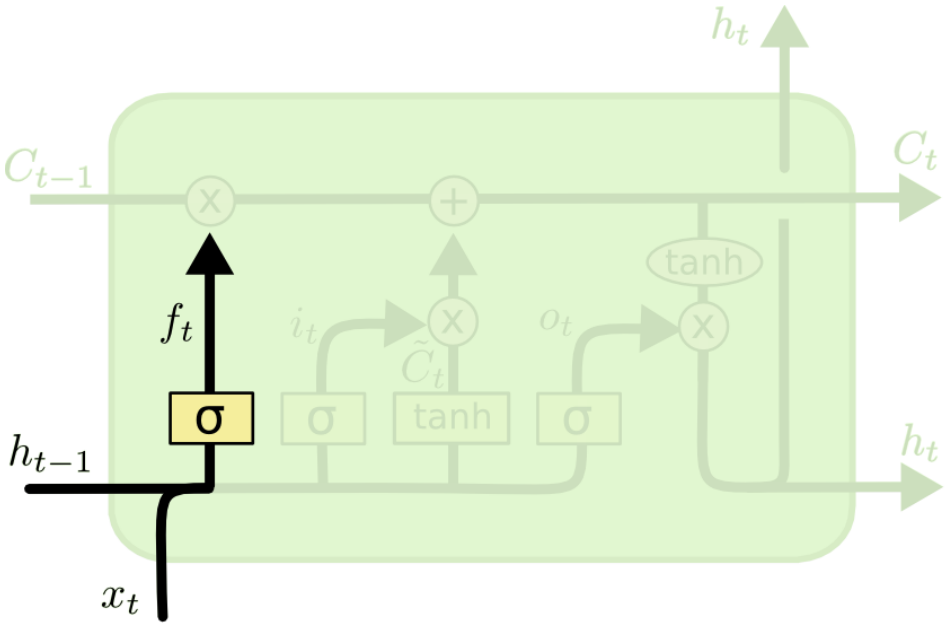
- Entscheidet ob Information hinzugefügt werden soll oder nicht
- Sigmoid Aktivierungsfunktion liefert Werte zwischen 0 und 1:
  - 0: Gate ist geschlossen; 1: Gate ist offen

Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Bilder:
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Rekurrentes Neuronales Netz - RNN

LSTM – Forget Gate Layer



Forget Gate Layer

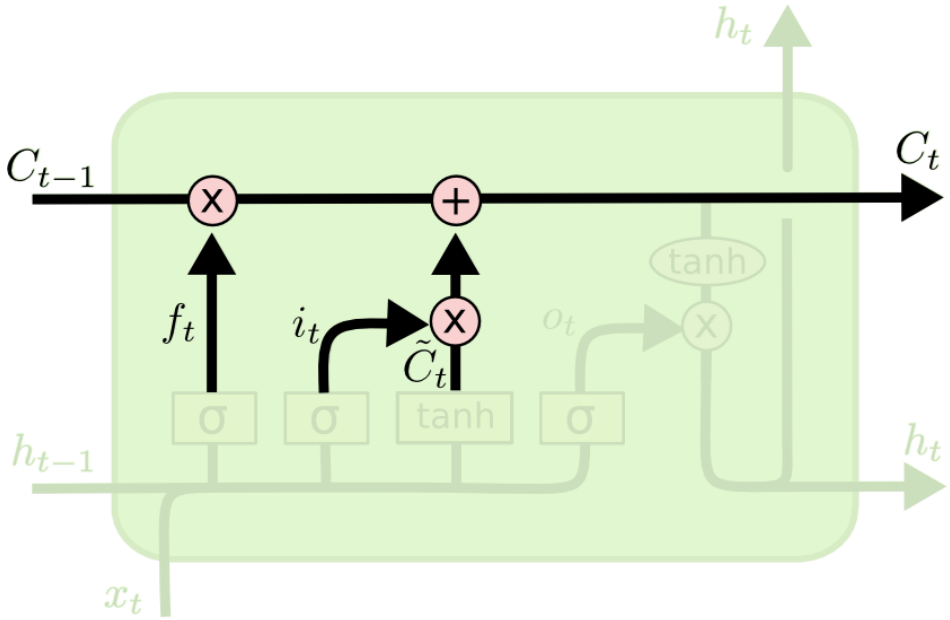
- f_t wird durch Sigmoid Aktivierungsfunktion berechnet
- 0 (1) heißt dabei die Information in C_{t-1} zu verwerfen (beizubehalten)

Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Quelle: Patrick S., Hahn-Schickard
Bilder:
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Rekurrentes Neuronales Netz - RNN

LSTM – Aktualisierung des Zellzustandes



Aktualisierung durchführen

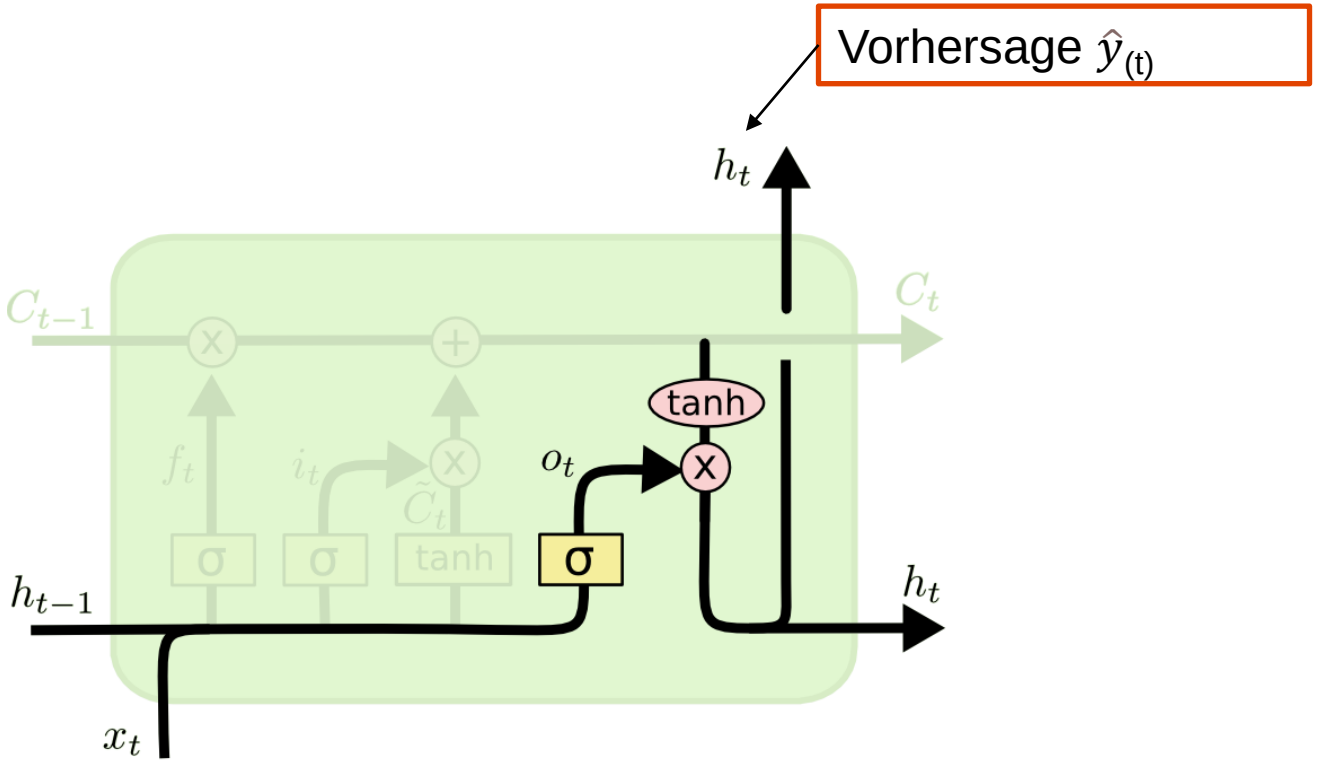
- Multiplikation des alten Zellzustands C_{t-1} mit f_t (vergessen von ausgewählten Informationen)
- Addition der neuen Informationen $i_t \cdot \tilde{C}_t$ (neue skalierte Informationen)

Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Quelle: Patrick S., Hahn-Schickard
Bilder:
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Rekurrentes Neuronales Netz - RNN

LSTM – Generieren einer Ausgabe



Generieren der Ausgabe (Output Gate Layer)

- Entscheidet welche Informationen von dem long-term state C_t in den short-term state h_t übergehen und zum aktuellen Zeitpunkt von der Zelle ausgegeben werden $\hat{y}_{(t)}$

- Normalerweise haben wir eine Schicht mit vielen LSTM Zellen. Das Langzeit- und Kurzzeitgedächtnis sind dann Vektoren in der Dimension der Anzahl Zellen in der Schicht.

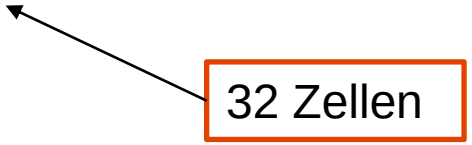
Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Bilder:
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Rekurrentes Neuronales Netz - RNN

Keras Implementierung eines LSTMs für ein Regressionsproblem

```
lstm_model = tf.keras.models.Sequential([
    tf.keras.layers.LSTM(32, return_sequences=True, input_shape=[None, 1]),
    tf.keras.layers.Dense(1)
])
```



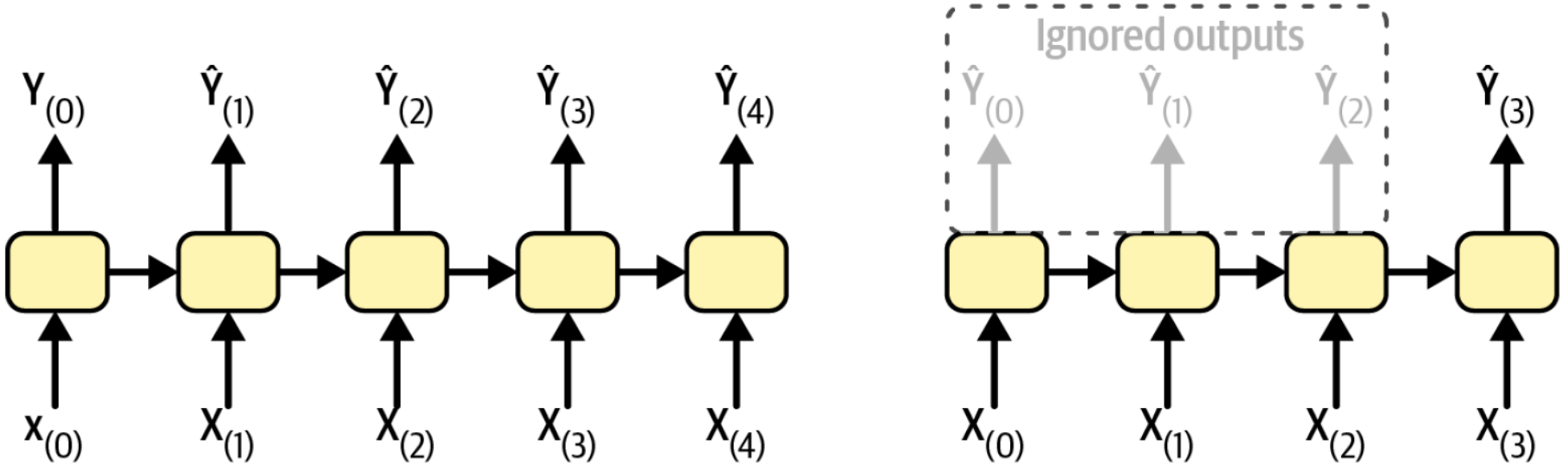
- Trainieren Sie das LSTM am Ende des Notebooks 01_sequences_rnns.ipynb und schauen Sie sich dessen Vorhersagen an
- Welches Modell performt besser auf den Validierungsdaten (RNN oder LSTM)?

Rekurrentes Neuronales Netz - RNN

Keras Implementierung eines LSTMs für ein Regressionsproblem

```
lstm_model = tf.keras.models.Sequential([
    tf.keras.layers.LSTM(32, return_sequences=True, input_shape=[None, 1]),
    tf.keras.layers.Dense(1)
])
```

`return_sequences=True` gibt beim RNN/LSTM die Sequenz der Hidden States h_t zurück.



Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Bildquelle: A Géron, Hands-On Machine Learning with scikit-learn, keras and Tensorflow, 3rd edition, 2022.

Rekurrentes Neuronales Netz - RNN

Feedforward vs. RNNs

Rekurrentes neuronales Netz

Ausgabe hängt von vorherigen Berechnungen ab

	Architektur & Signalfluß	Inspiriert durch	Gedächtnis	Eingabe und Ausgabe	Erkennung von
KNN & CNN	• Feedforward • eine Richtung (vom Eingang zum Ausgang)	• Neuronen • visueller Cortex	• Keines	• Feste Größe • Bilder	• Muster
RNN	• Feedback • Beide Richtungen (Loops)	• Neocortex	• Dynamischer Memory	• Beliebige Größe • Daten in sequentieller Form (z.B. Zeitserien)	• Zeitlich kodierte Informationen

Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik
Quelle: Patrick S., Hahn-Schickard

Deep Learning

Gegenüberstellung: KNNs, CNNs und RNNs

Multi-Layer Perceptron (MLP) bedeutet das gleiche wie fully-connected neural network

	KNNs (MLP)	CNNs	RNNs
Daten	Tabellarisch	Bilddaten	Sequenzen
Rekurrente Verbindungen	Nein	Nein	Ja
Geteilte Gewichte	Nein	Ja	Ja
Erkennen räumlicher Beziehungen	Nein	Ja	Nein
Vanishing & Exploding Gradients	Ja	Ja	Ja

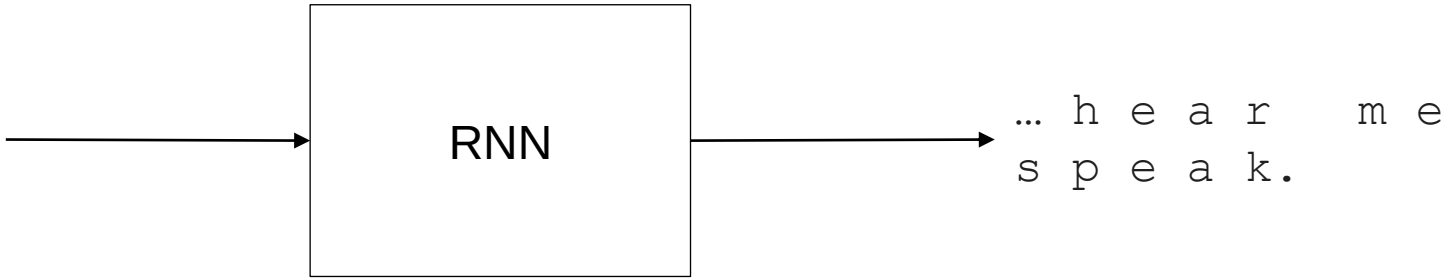
Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik
Quelle: Patrick S., Hahn-Schickard

Rekurrentes Neuronales Netz - RNN

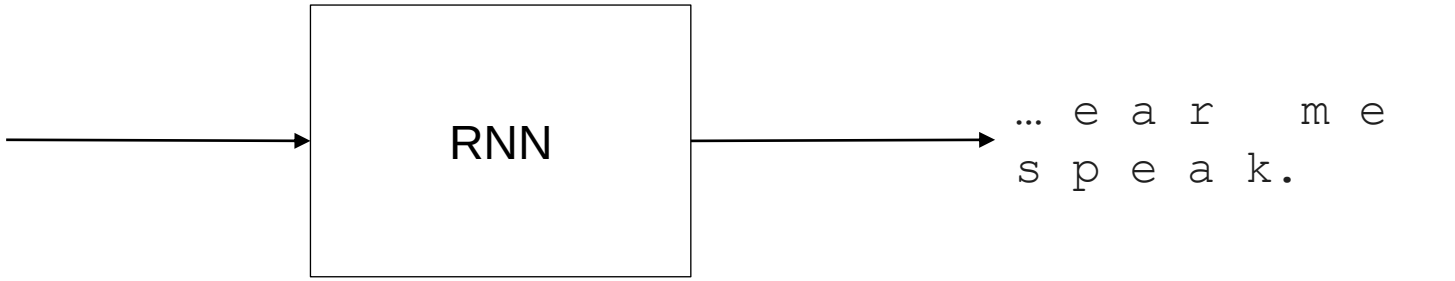
Generating Shakespearean Text Using a Character RNN

- Idee: Training eines RNNs, das Texte von Shakespeare Zeichen für Zeichen vervollständigt

t = 0:



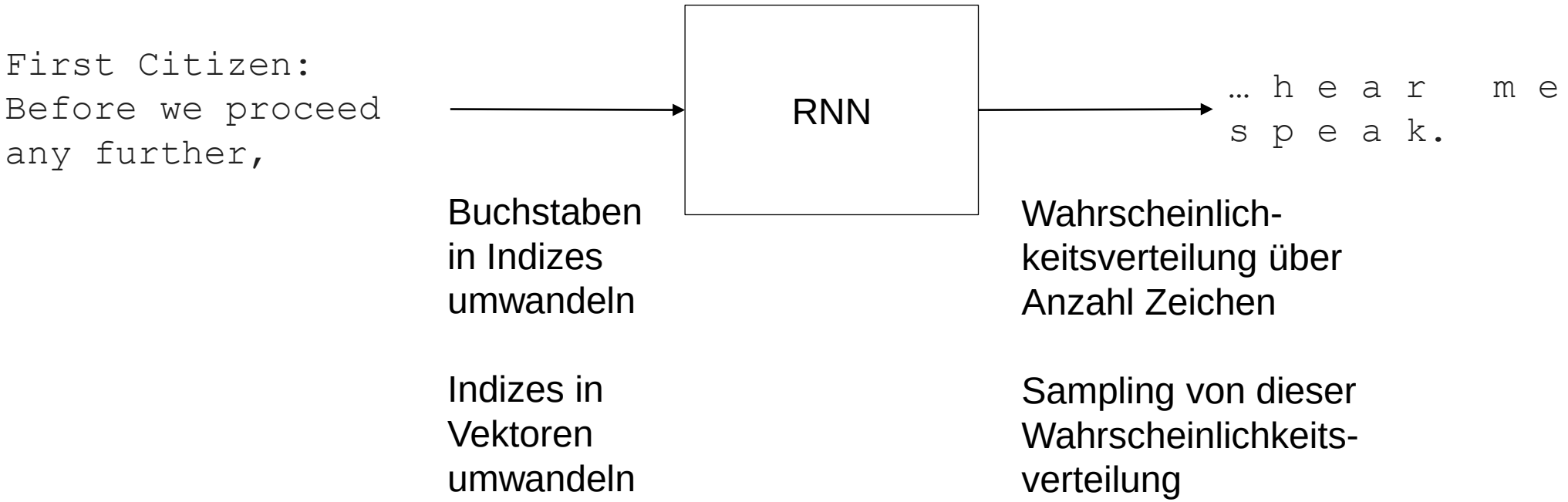
t = 1:



Rekurrentes Neuronales Netz - RNN

Generating Shakespearean Text Using a Character RNN

- Idee: Training eines RNNs, das Texte von Shakespeare Zeichen für Zeichen vervollständigt



Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik
Technology Arts Sciences TH Köln

Rekurrentes Neuronales Netz - RNN

Generating Shakespearean Text Using a Character RNN

Buchstaben → Indizes

- Indizes sind Klassen IDs: 0, …, 38
- Wenn RNN Klasse 5 vorhersagt, wissen wir, dass der Satz mit einem ‚i' weiter geht.
- Problem: Abstände zwischen Klassen IDs machen keinen Sinn. Bsp.: |e – s| = 6. Warum?
- Logischer wenn nah beieinander liegen:
  - Vokale
  - Satzzeichen: ‚.', ‚,', ‚?', …
  - Leerzeichen, neue Zeile

Character to Encoded Index Mapping:

' ': 0

'e': 1

't': 2

'o': 3

'a': 4

'i': 5

'h': 6

's': 7

'r': 8

'n': 9

': 10

…

Prof. Dr. Daniel Gaida

Professor für Cyber-Physische Systeme

Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Technology Arts Sciences TH Köln

Rekurrentes Neuronales Netz - RNN

Generating Shakespearean Text Using a Character RNN

Buchstaben → Indizes

```
print(shakespeare_text[:80])
```

```
First Citizen: Before we proceed any
further, hear me speak. All: Speak, speak.
```

```
text_vec_layer =
tf.keras.layers.TextVectorization(
split="character", standardize="lower")
```

```
print(text_vec_layer(
[shakespeare_text[:80]]) - 2)
```

```
19 5 8 7 2 0 18 5 2 5 35 1 9 23 10 21 1 19
3 8 1 0 16 1 0 22 8 3 18 1 1 12 0 4 9 15 0
19 13 8 2 6 1 8 17 0 6 1 4 8 0 14 1 0 7 22
1 4 24 26 10 10 4 11 11 23 10 7 22 1 4 24
17 0 7 22 1 4 24 26
```

Embedding Vektoren

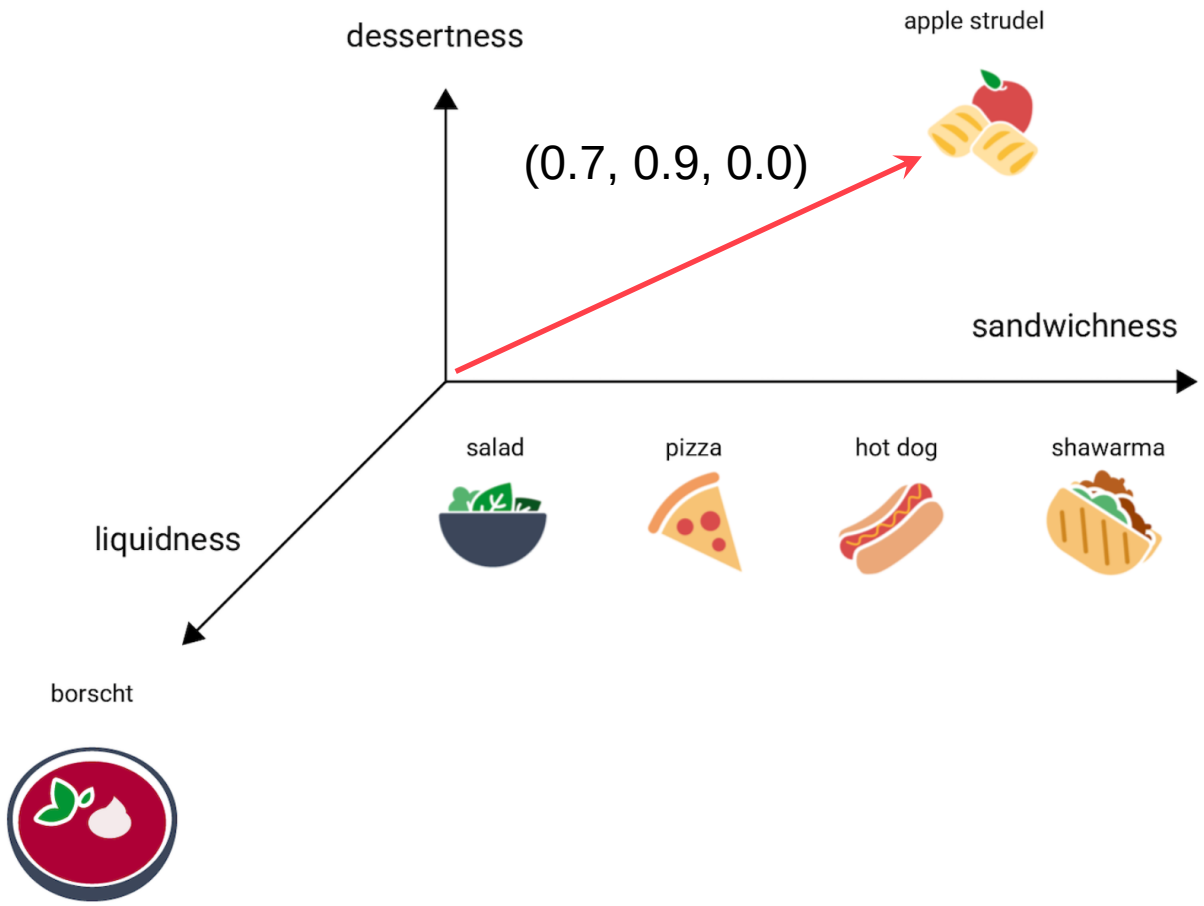
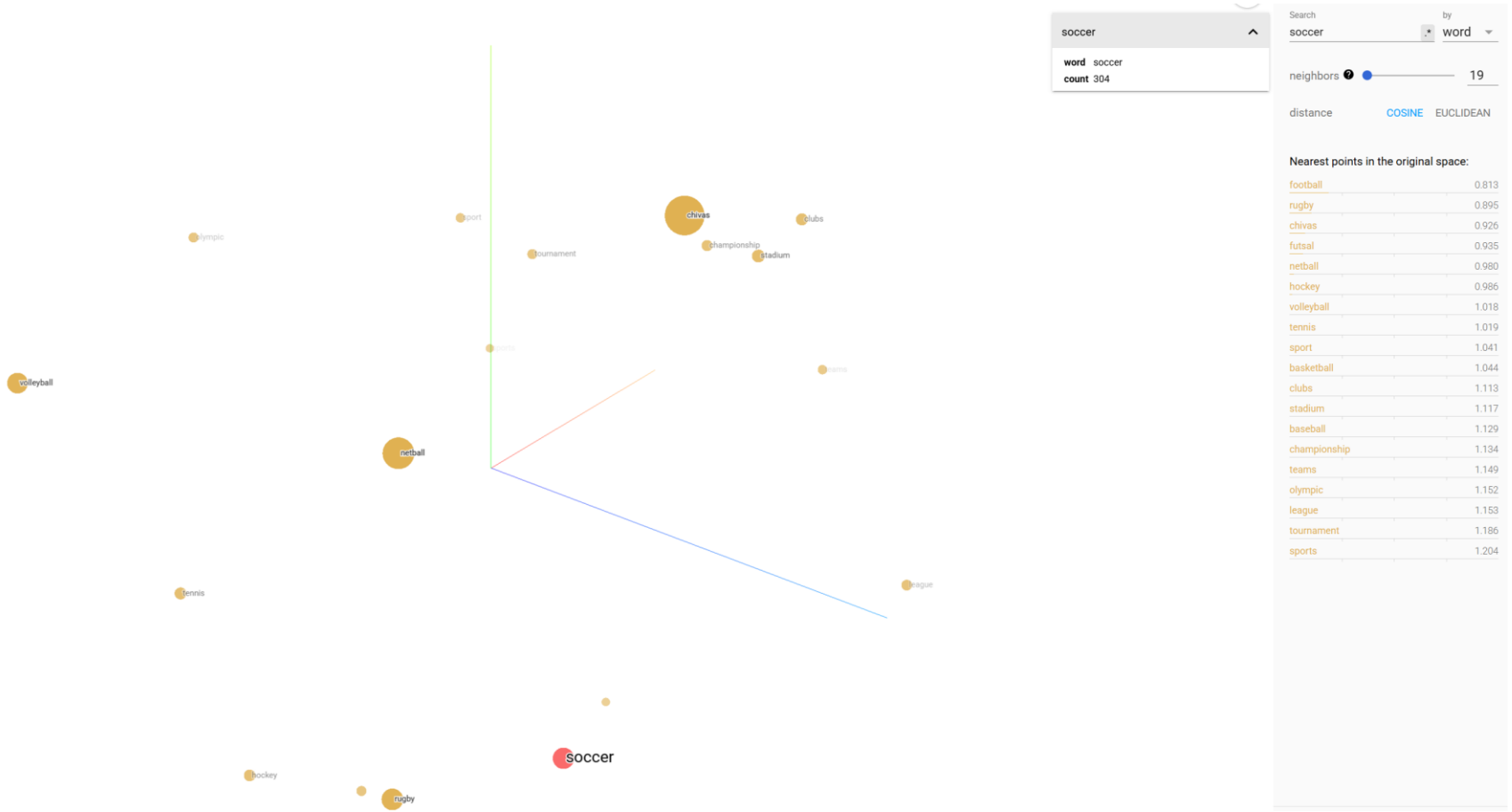


Figure 5. *Foods plotted by "sandwichness," "dessertness," and "liquidness."*

- Vektoren zur Repräsentation von Wörtern leben in einem hochdimensionalen Vektorraum
  - Bspw. 1024
- Hier Annahme:
  - 3-dimensionaler Vektorraum

Embedding Vektoren



Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

https://projector.tensorflow.org/

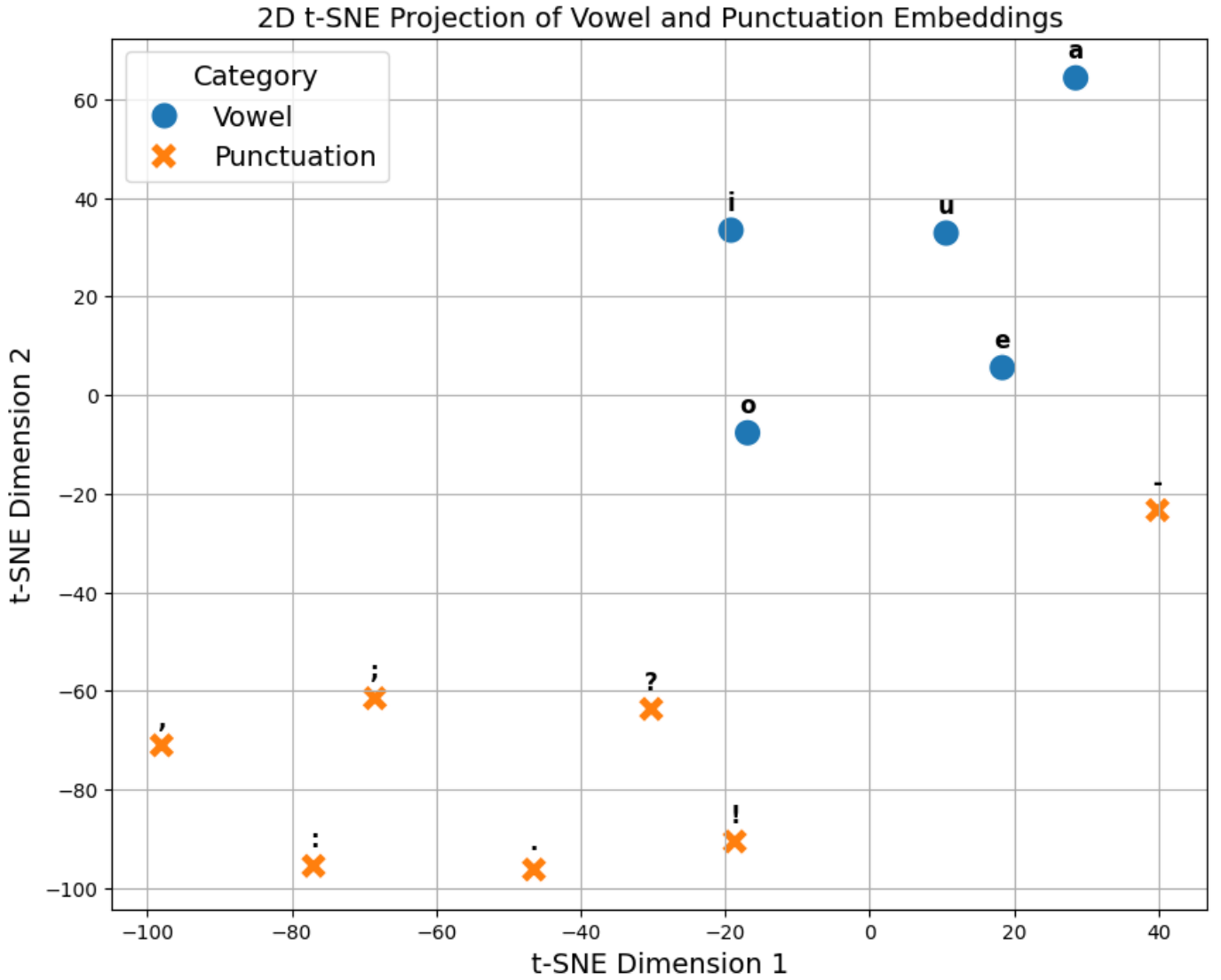
Rekurrentes Neuronales Netz - RNN

Generating Shakespearean Text Using a Character RNN

```
tf.keras.layers.Embedding(
input_dim=n_tokens, output_dim=16)
```

- Erzeugt für jedes Zeichen einen Vektor in einem 16-dimensionalen Raum
- Vektoren werden gelernt

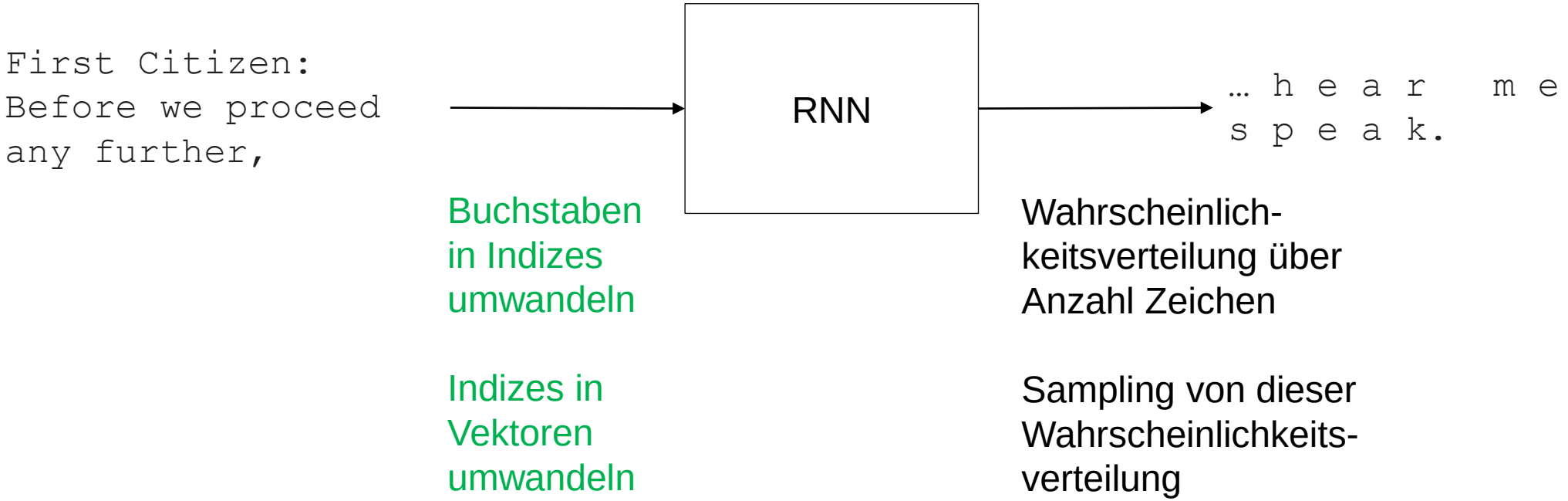
- `n_tokens` = Anzahl Zeichen (39)



Rekurrentes Neuronales Netz - RNN

Generating Shakespearean Text Using a Character RNN

- Idee: Training eines RNNs, das Texte von Shakespeare Zeichen für Zeichen vervollständigt



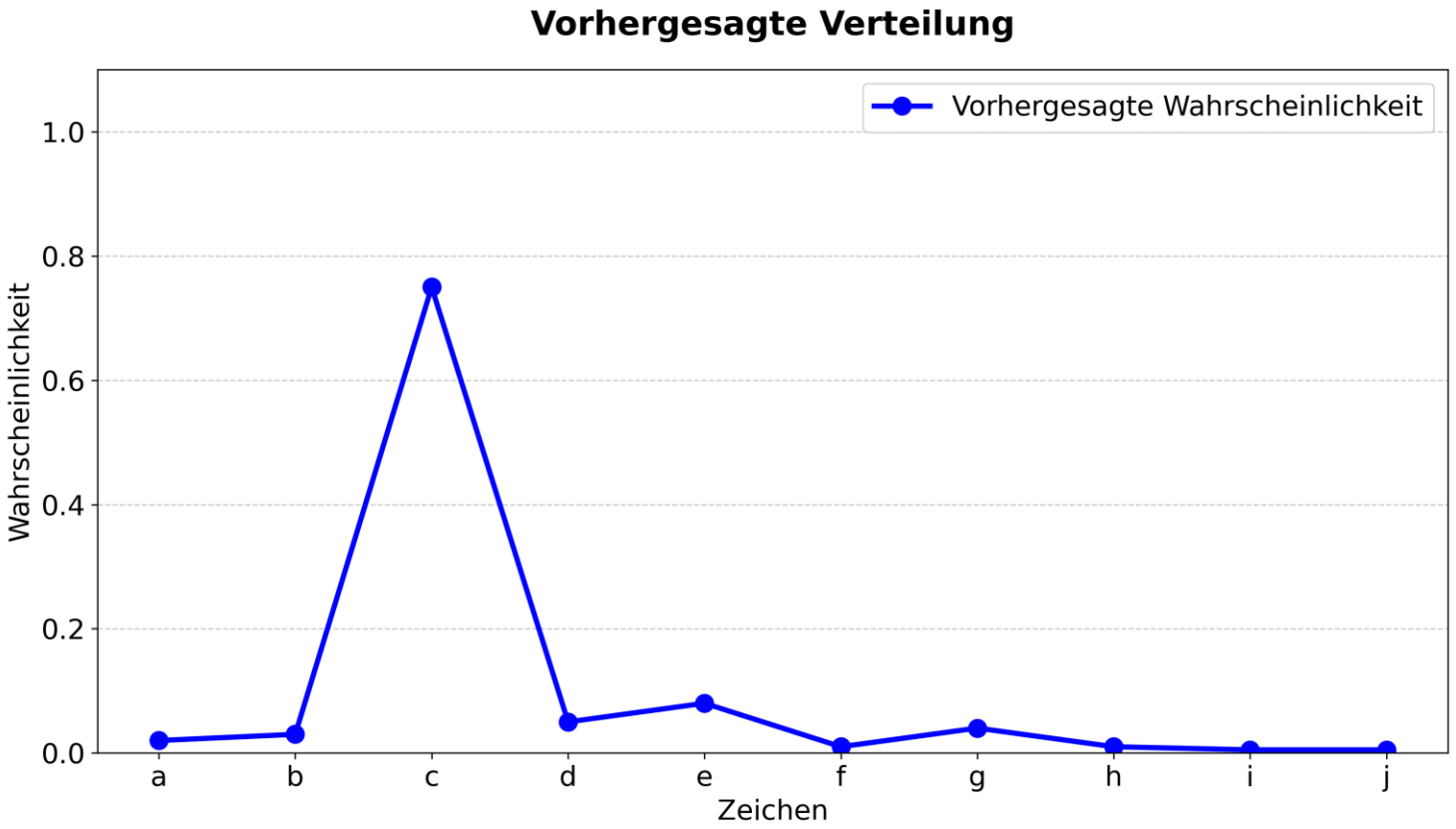
Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik
Technology Arts Sciences TH Köln

Rekurrentes Neuronales Netz - RNN

Generating Shakespearean Text Using a Character RNN

```
tf.keras.layers.Dense(n_tokens, activation="softmax")
```

- Letzter Layer mit SoftMax gibt eine Wahrscheinlichkeitsverteilung über alle 39 Zeichen zurück
- Gemäß dieser Wkt-Verteilung wird das wahrscheinlichste Zeichen gesampelt (hier: c)

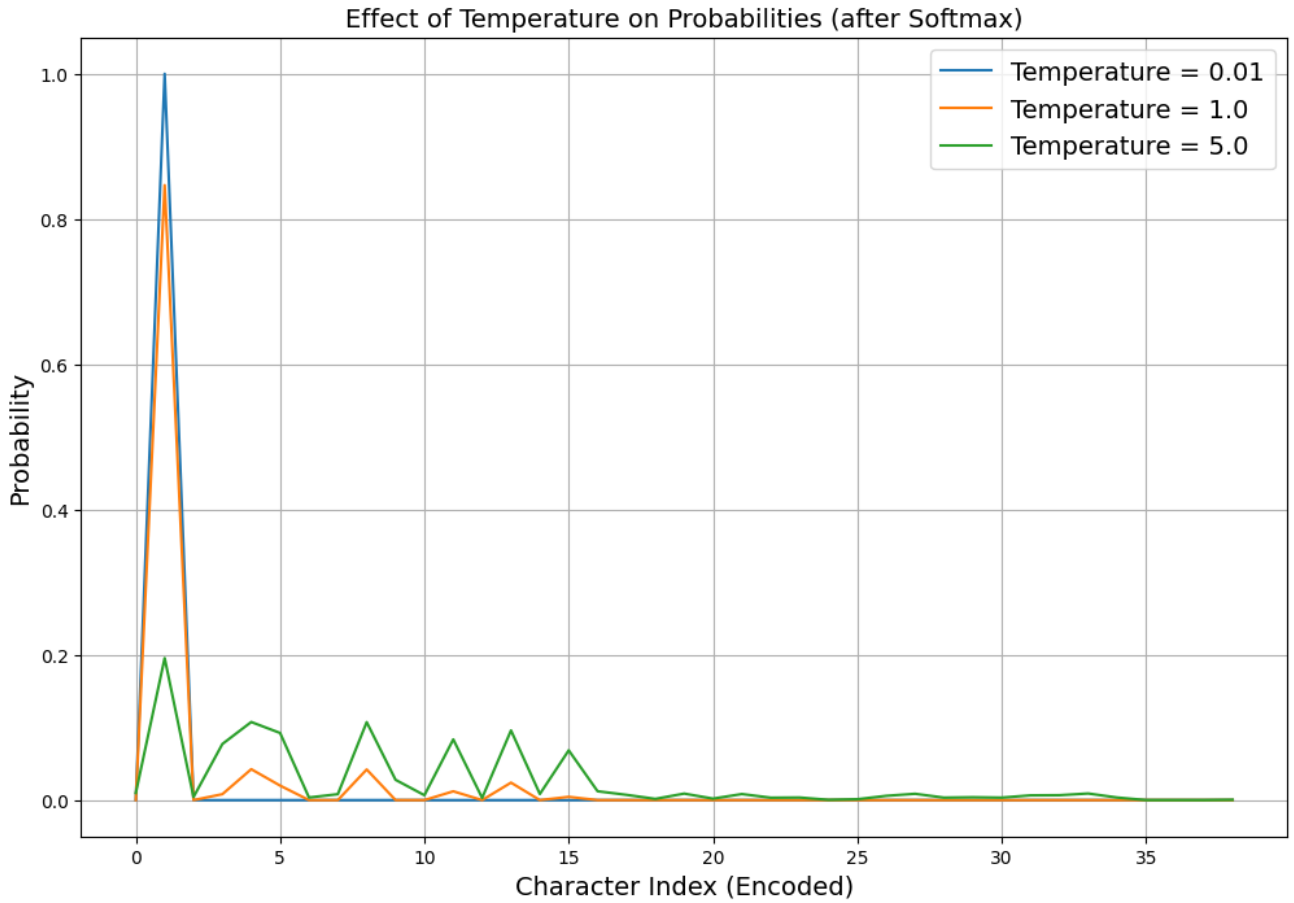


Rekurrentes Neuronales Netz - RNN

Generating Shakespearean Text Using a Character RNN

Anpassen des Samplings aus der Wahrscheinlichkeitsverteilung durch Parameter **Temperatur**

- Temperatur skaliert die Wkt-Verteilung
  - Temperatur = 1
    → Original
  - 0 < Temperatur < 1
    → Sampling wird deterministischer
    → Zeichen mit max. Wkt. wird gezogen
  - Temperatur > 1
    → Wkt-Verteilung wird breiter
    → Text wird kreativer/blödsinniger



Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik
Technology Arts Sciences TH Köln

Rekurrentes Neuronales Netz - RNN

Generating Shakespearean Text Using a Character RNN

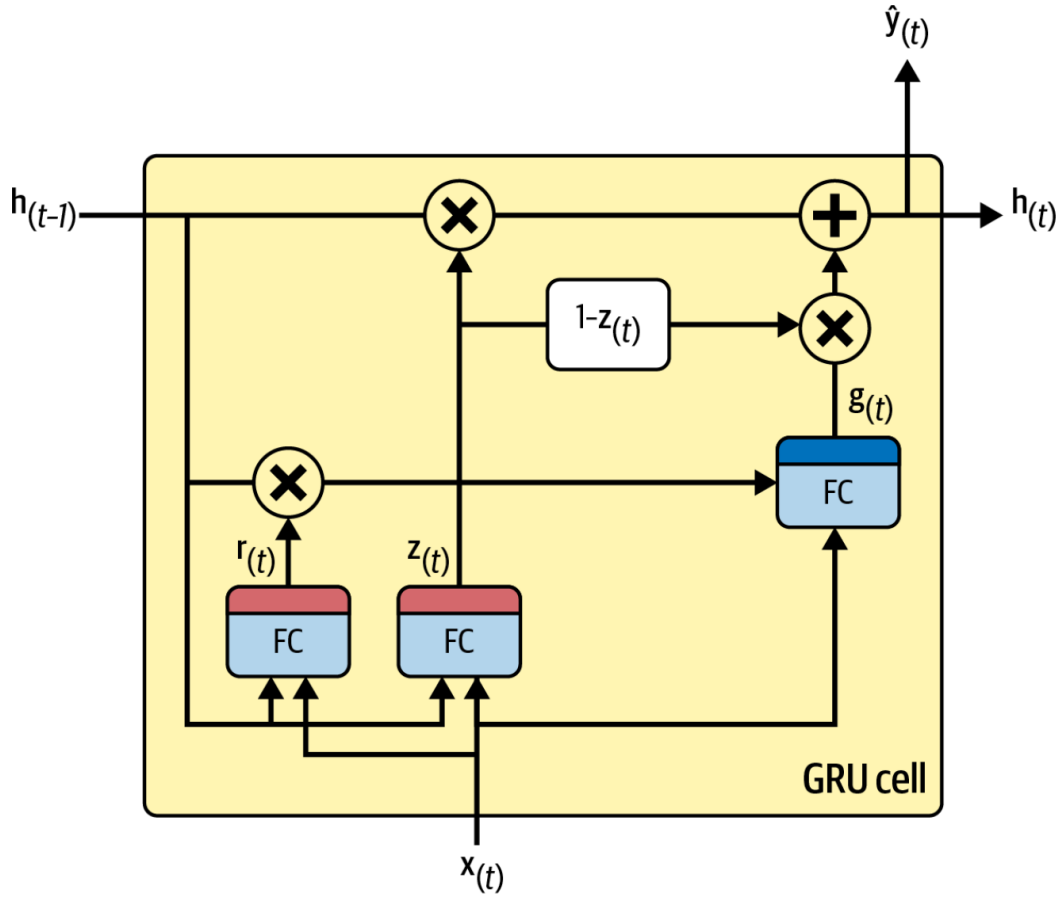
```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(input_dim=n_tokens, output_dim=16),
    tf.keras.layers.GRU(128, return_sequences=True),
    tf.keras.layers.Dense(n_tokens, activation="softmax")
])
```

GRU: Vereinfachte Erweiterung von LSTM

```
shakespeare_model = tf.keras.Sequential([
    text_vec_layer,
    tf.keras.layers.Lambda(lambda X: X - 2),  # no <PAD> or <UNK> tokens
    model
])
```

Rekurrentes Neuronales Netz - RNN

Gated Recurrent Unit Cell (GRU)



Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Bildquelle: A Géron, Hands-On Machine Learning with scikit-learn, keras and Tensorflow, 3rd edition, 2022.

Rekurrentes Neuronales Netz - RNN

Generating Shakespearean Text Using a Character RNN

Sampling von der Wahrscheinlichkeitsverteilung

```
def next_char(text, temperature=1):
    y_proba = shakespeare_model.predict([text])[0, -1:]
    rescaled_logits = tf.math.log(y_proba) / temperature
    char_id = tf.random.categorical(rescaled_logits, num_samples=1)[0, 0]
    return text_vec_layer.get_vocabulary()[char_id + 2]
```

Rekurrentes Neuronales Netz - RNN

Übung: Generating Shakespearean Text Using a Character RNN

Führen Sie das Notebook 02_nlp_with_rnns_and_attention.ipynb aus.

- Sie müssen das Modell nicht trainieren (Dauer > 1-2 h), sondern können es herunterladen

- Beantworten Sie folgende Fragen:
  - Wie viele Zeichen gibt es im Text von Shakespeare?
  - Wie erfolgt die Aufteilung in Training-, Validierung- und Testdaten?

- Wie gut ist der generierte Text?
  - Welchen Einfluss hat die Temperatur?

Rekurrentes Neuronales Netz - RNN

Aktuelle Forschung

“xLSTM is a new Recurrent Neural Network architecture based on ideas of the original LSTM. Through Exponential Gating with appropriate normalization and stabilization techniques and a new Matrix Memory it overcomes the limitations of the original LSTM and shows promising performance on Language Modeling when compared to Transformers or State Space Models.”

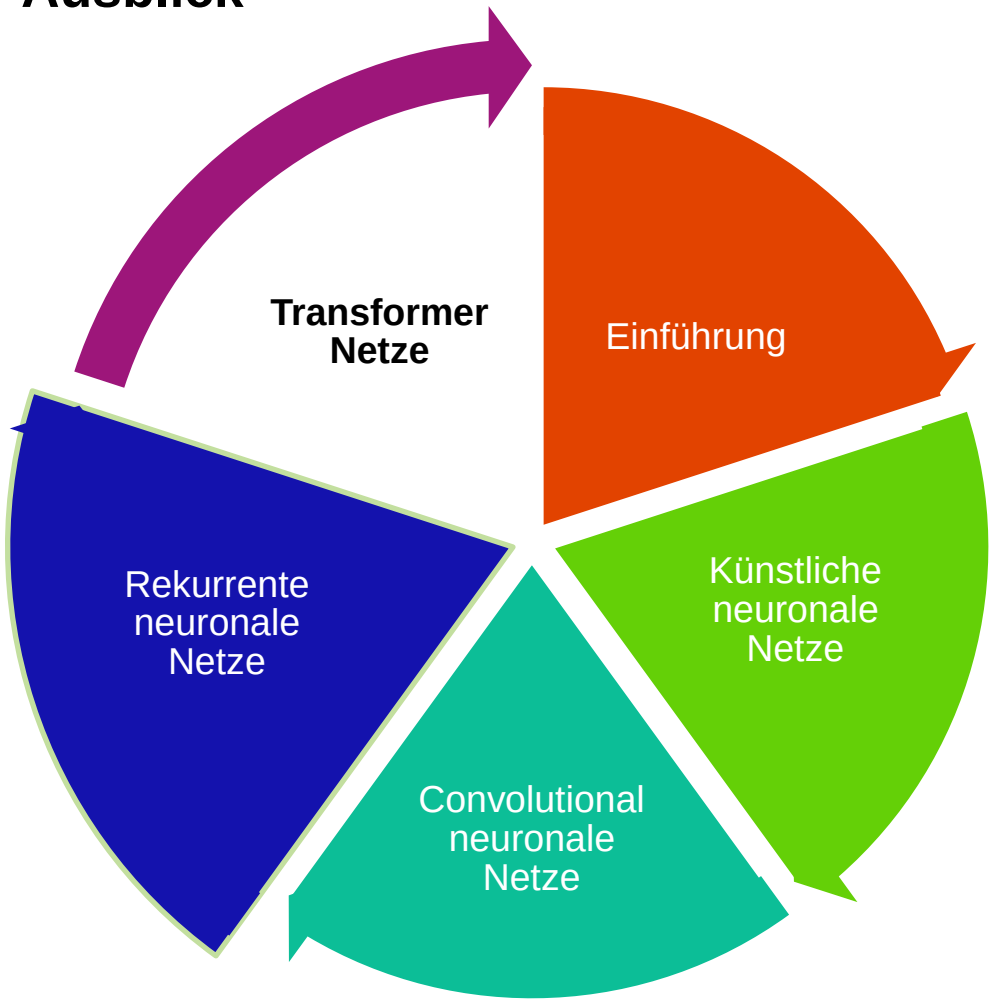
Beck, M., Pöppel, K., Spanring, M., Auer, A., Prudnikova, O., Kopp, M., ... & **Hochreiter**, S. (2024). xlstm: Extended long short-term memory. *Advances in Neural Information Processing Systems*, *37*, 107547-107603.

Quelle und Code:

https://github.com/nx-ai/xlstm

Deep Learning

Ausblick



Transformer Netze

- Einführung
  - Tokenization
  - Transformer Architektur
    - Embedding
    - Positional Encoding
    - Attention
- Prompting
- Stärken und Schwächen von LLMs

Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Warum Transformer RNNs verdrängt haben?

Eigenschaft	RNN/LSTM	Transformer
Parallelisierung	schlecht	sehr gut
Lange Kontexte	begrenzt	sehr gut
Training großer Modelle	schwierig	sehr gut
Rechenaufwand	gering	hoch
Einsatz heute	Nischen	Standard

Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik
Technology Arts Sciences TH Köln

Wie funktioniert ChatGPT?

- Hinter ChatGPT verbirgt sich ein
  - Large Language Model (LLM)
    - Eine Art tiefes neuronales Netz (Transformer Architektur)

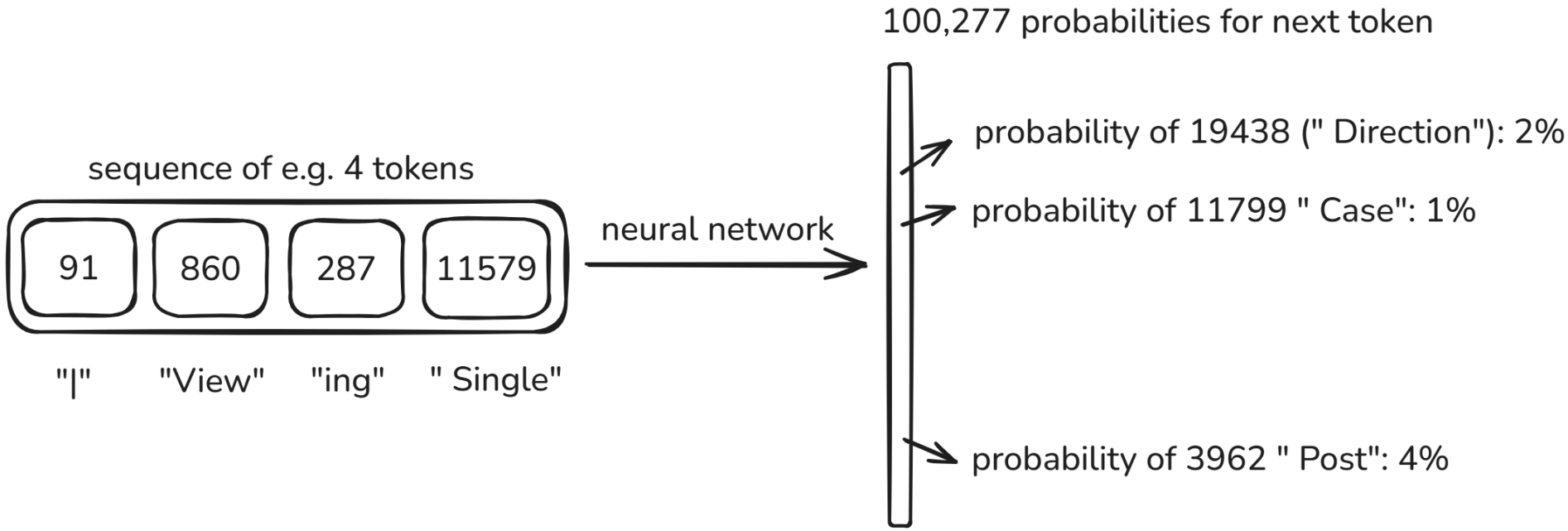
- Text wird als eine Sequenz von Wörtern oder Teilwörtern angesehen (Tokens)
  - Ein Token entspricht einem Wort, Teilwort oder einem Buchstaben/Satzzeichen
  - Text, den Sie in den Prompt von ChatGPT schreiben, muss in Tokens umgewandelt werden
    - Tokenization
  - Tokens sind das Vokabular

Tokenization

- Demo: https://tiktokenizer.vercel.app/

LLM: Vorhersage einer Wahrscheinlichkeit für das nächste Token

- Eingabesequenz der Tokens (Prompt) geht in das LLM rein
- LLM liefert für alle bekannten Tokens eine Wahrscheinlichkeit zurück
- Tokens mit einer hohen Wahrscheinlichkeit folgten im Trainingsdatensatz häufig der Eingabesequenz von Tokens



Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Quelle: Andrej Karpathy, Deep Dive into LLMs like ChatGPT, https://www.youtube.com/watch?v=7xTGNNLPyMI

LLM: Vorhersage einer Wahrscheinlichkeit für das nächste Token

Aktivierung

Setzen Sie diesen Satz mit einem Wort fort:

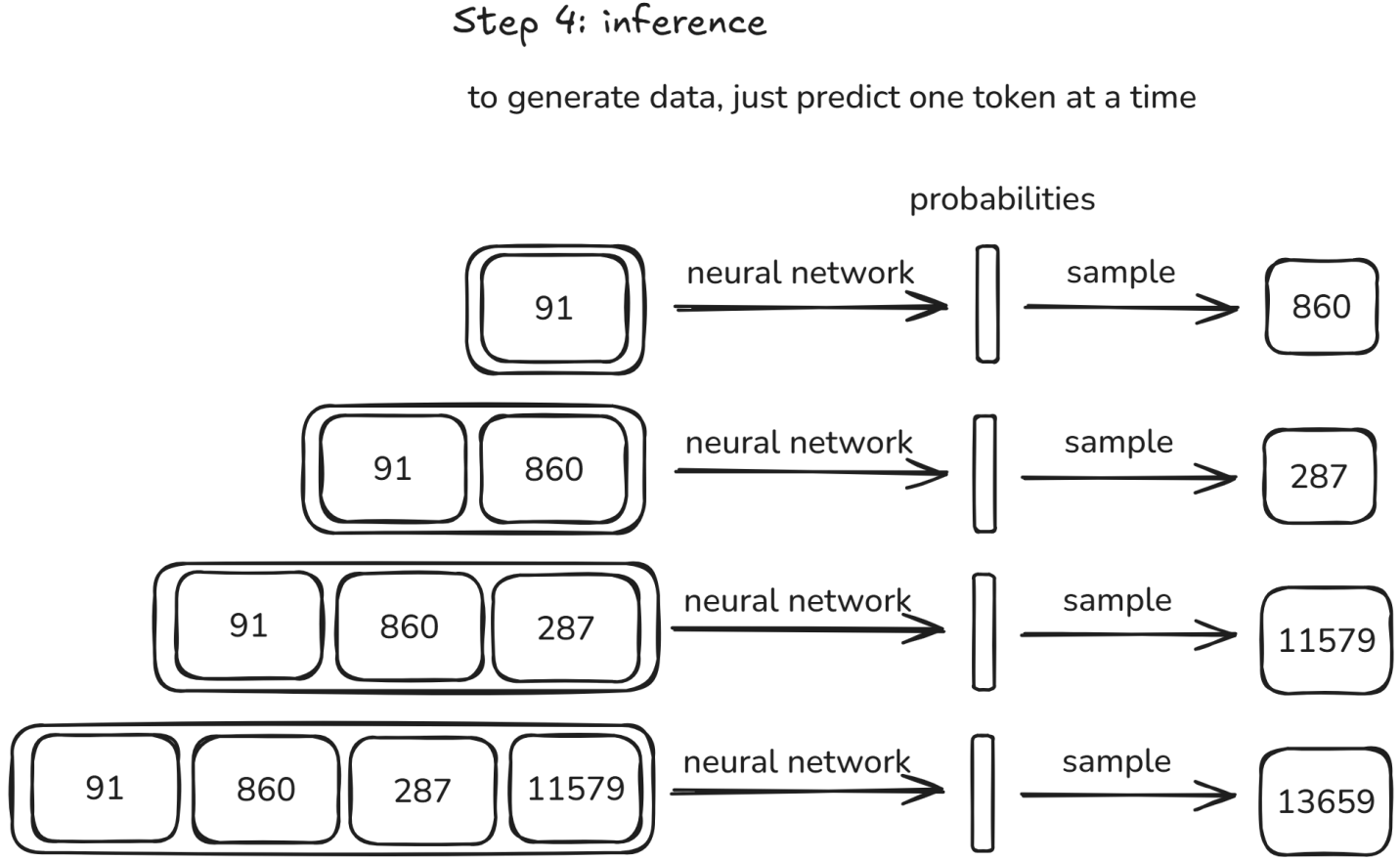
- „Deep Learning wird in Zukunft ...“

Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik
Technology Arts Sciences TH Köln

LLM: Ein Autoregressives Modell

Quelle: Andrej Karpathy, Deep Dive into LLMs like ChatGPT, https://www.youtube.com/watch?v=7xTGNNLPyMI

LLMs sind autoregressive Modelle



Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik
Technology Arts Sciences TH Köln

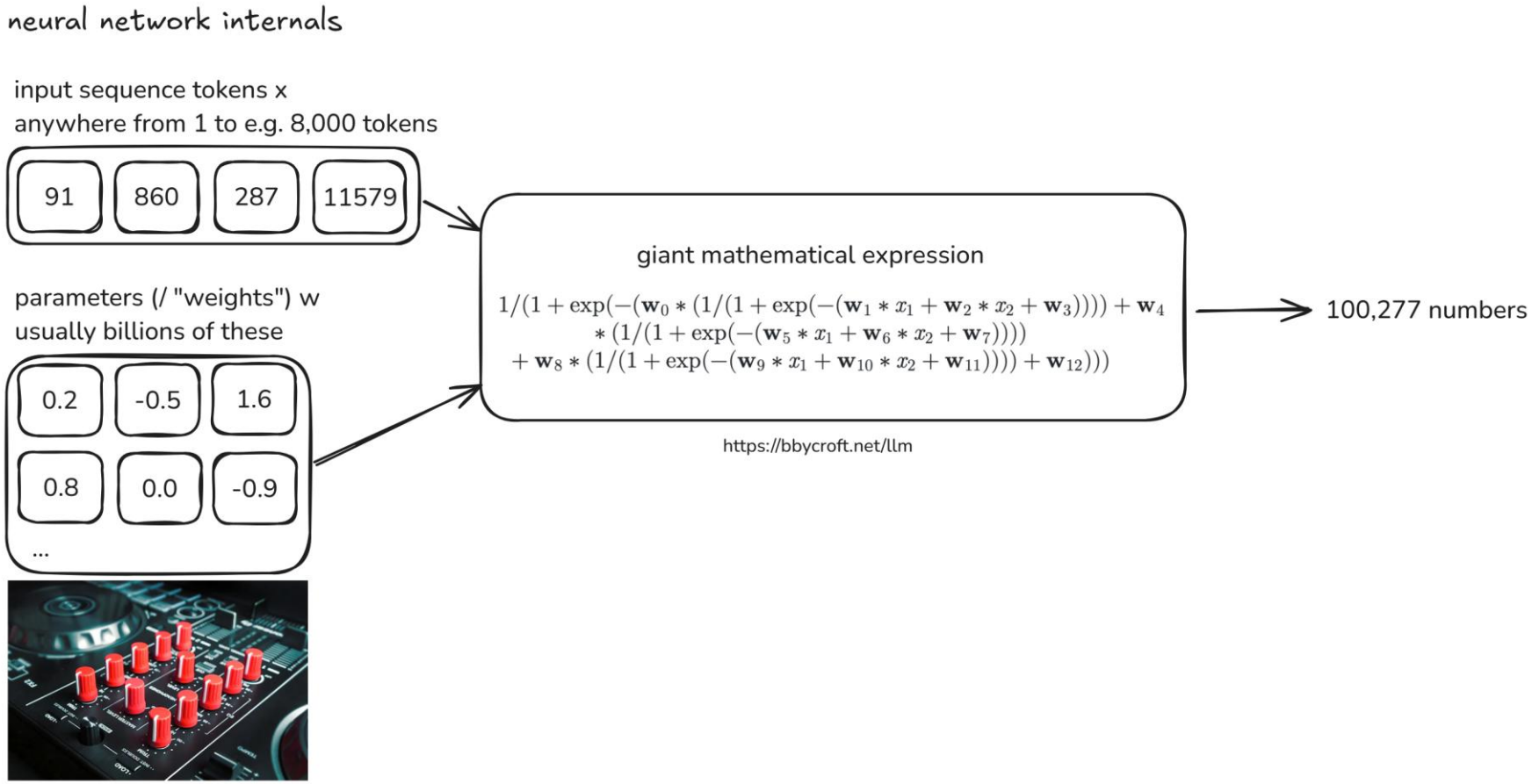
LLM: Ein Autoregressives Modell

Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Quelle: https://huggingface.co/learn/agents-course/unit1/what-are-llms

Large Language Modelle

Quelle: Andrej Karpathy, Deep Dive into LLMs like ChatGPT, https://www.youtube.com/watch?v=7xTGNNLPyMI



Large Language Modelle

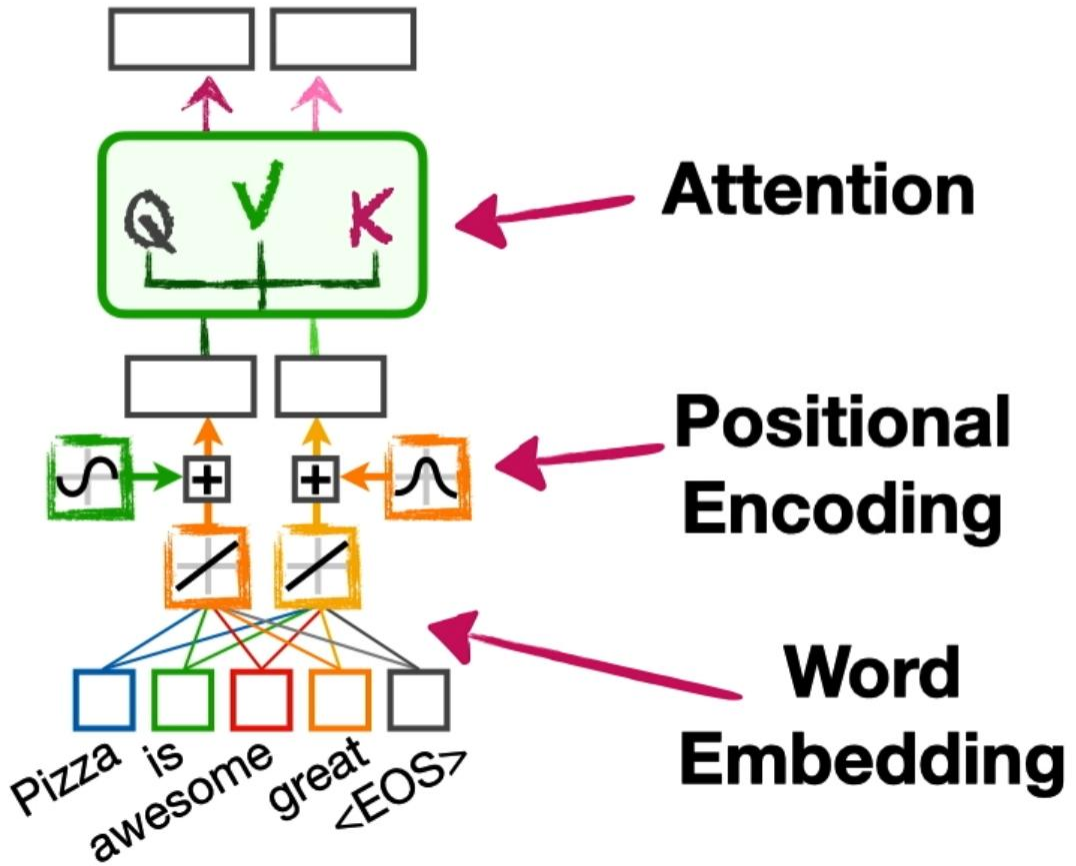
Zeige Video zu „the Autoregressive Loop“ und Visualisierung

- https://learn.deeplearning.ai/courses/transformers-in-practice/lesson/79yzruak/the-autoregressive-loop

Zeige Video zu „Token Sampling“ und Visualisierung

- https://learn.deeplearning.ai/courses/transformers-in-practice/lesson/sh74qlbh/token-sampling

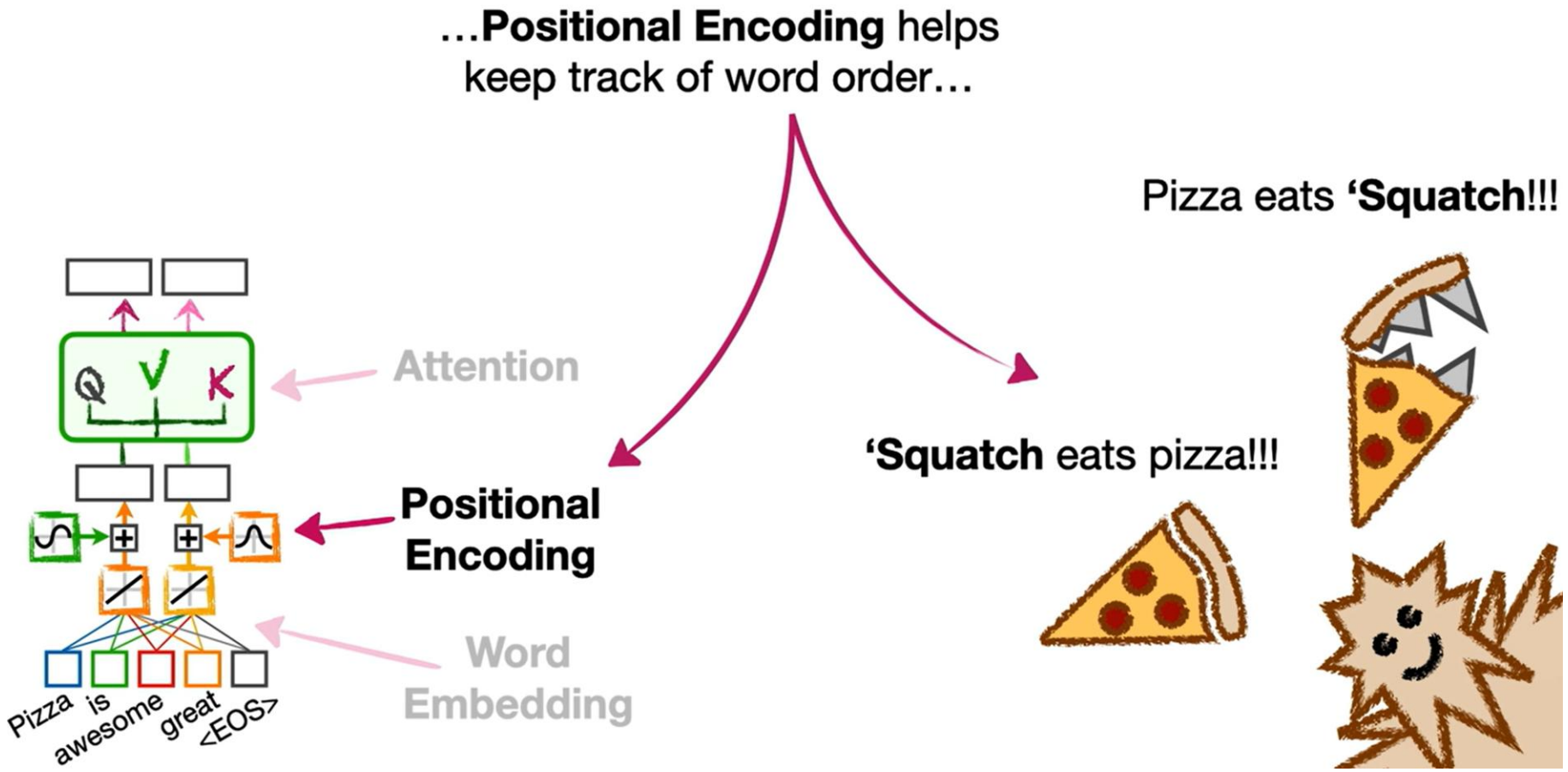
Die Transformer Architektur: Attention is all you need



Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Quelle: Attention in Transformers: Concepts and Code in PyTorch, Josh Starmer, DeepLearning.AI, https://learn.deeplearning.ai/courses/attention-in-transformers-concepts-and-code-in-pytorch/lesson/ym7dj/the-main-ideas-behind-transformers-and-attention

Die Transformer Architektur: Attention is all you need

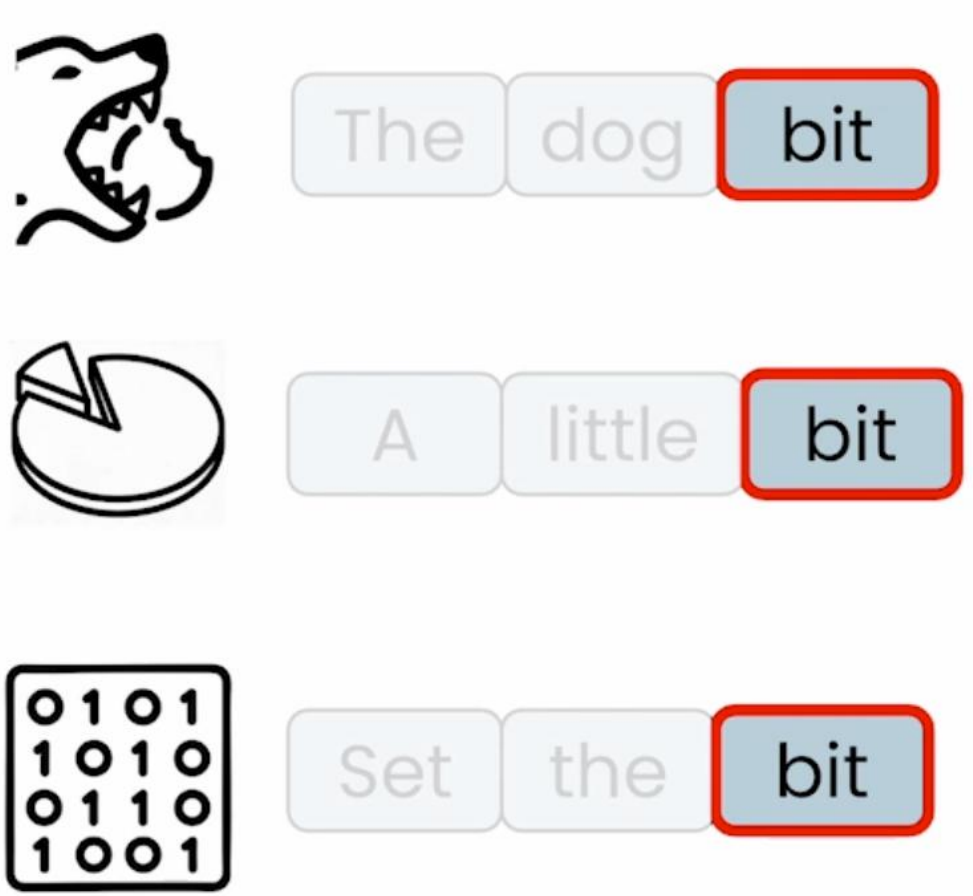


Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Quelle: Attention in Transformers: Concepts and Code in PyTorch, Josh Starmer, DeepLearning.AI, https://learn.deeplearning.ai/courses/attention-in-transformers-concepts-and-code-in-pytorch/lesson/ym7dj/the-main-ideas-behind-transformers-and-attention

Die Transformer Architektur: Attention is all you need

- Stellen Sie sich vor, das LLM soll diese drei Sätze vervollständigen
- Der Embedding Vektor für **bit** in allen drei Sätzen ist identisch
- Damit LLM den Satz sinnvoll fortsetzt, sollten die Embedding Vektoren für **bit** in allen drei Sätzen unterschiedlich sein



Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Bildquelle: Transformers in Practice, Instructor: Sharon Zhou, DeepLearning.AI
https://learn.deeplearning.ai/courses/transformers-in-practice/lesson/79yzr2/attention

Die Transformer Architektur: Attention is all you need

Aktivierung: Menschliche Attention

Welches Wort ist gemeint? Betrachten Sie den folgenden Satz:

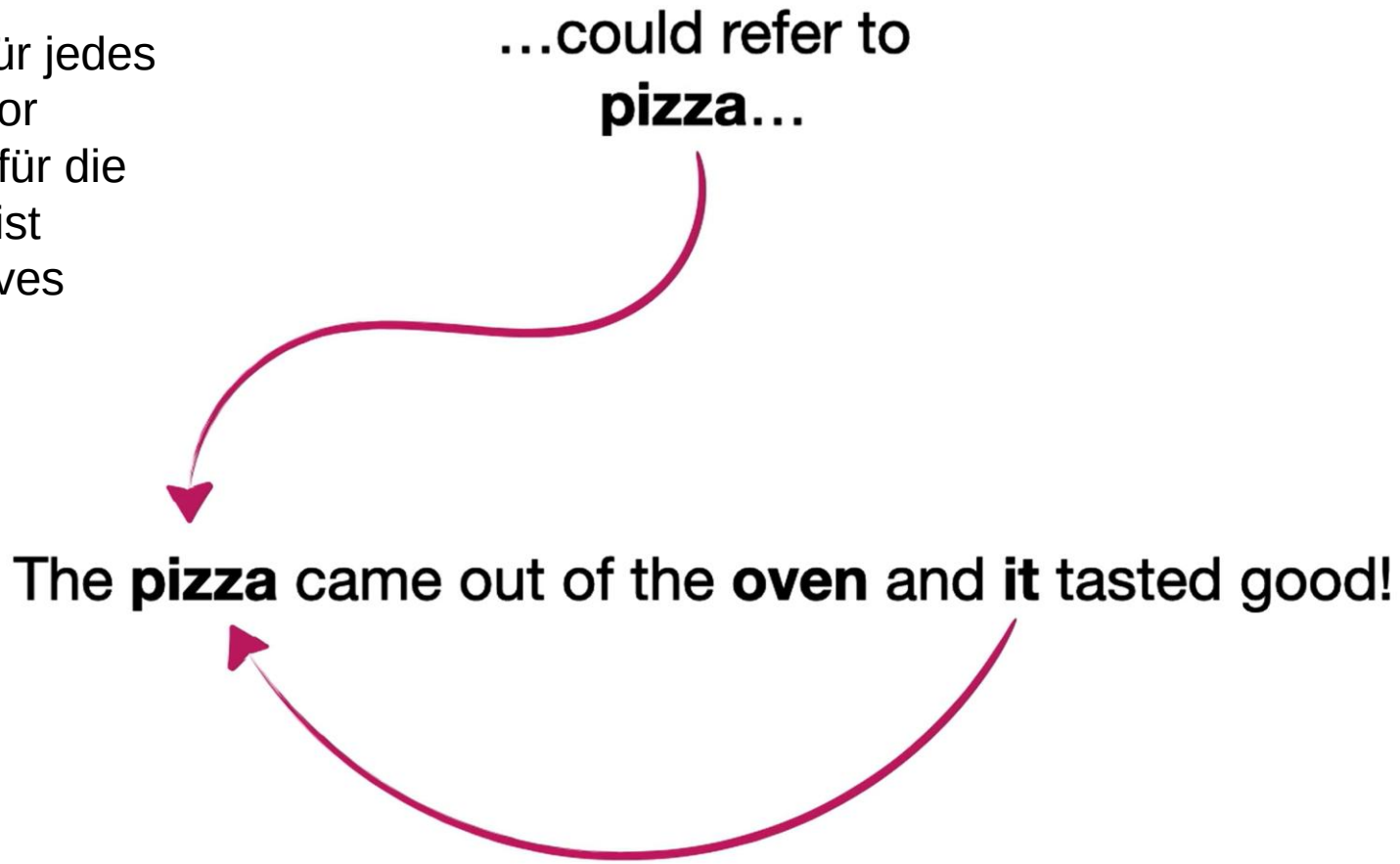
„Der Programmierer erklärte dem Studenten den Code, weil er ihn nicht verstand.“

- Wer ist mit „er“ gemeint?
- Wer ist mit „ihn“ gemeint?
- Welche Wörter im Satz helfen Ihnen bei der Entscheidung?

- Markieren/Nennen Sie die Wörter, auf die Sie beim Verstehen des Satzes besonders achten.

Die Transformer Architektur: Attention is all you need

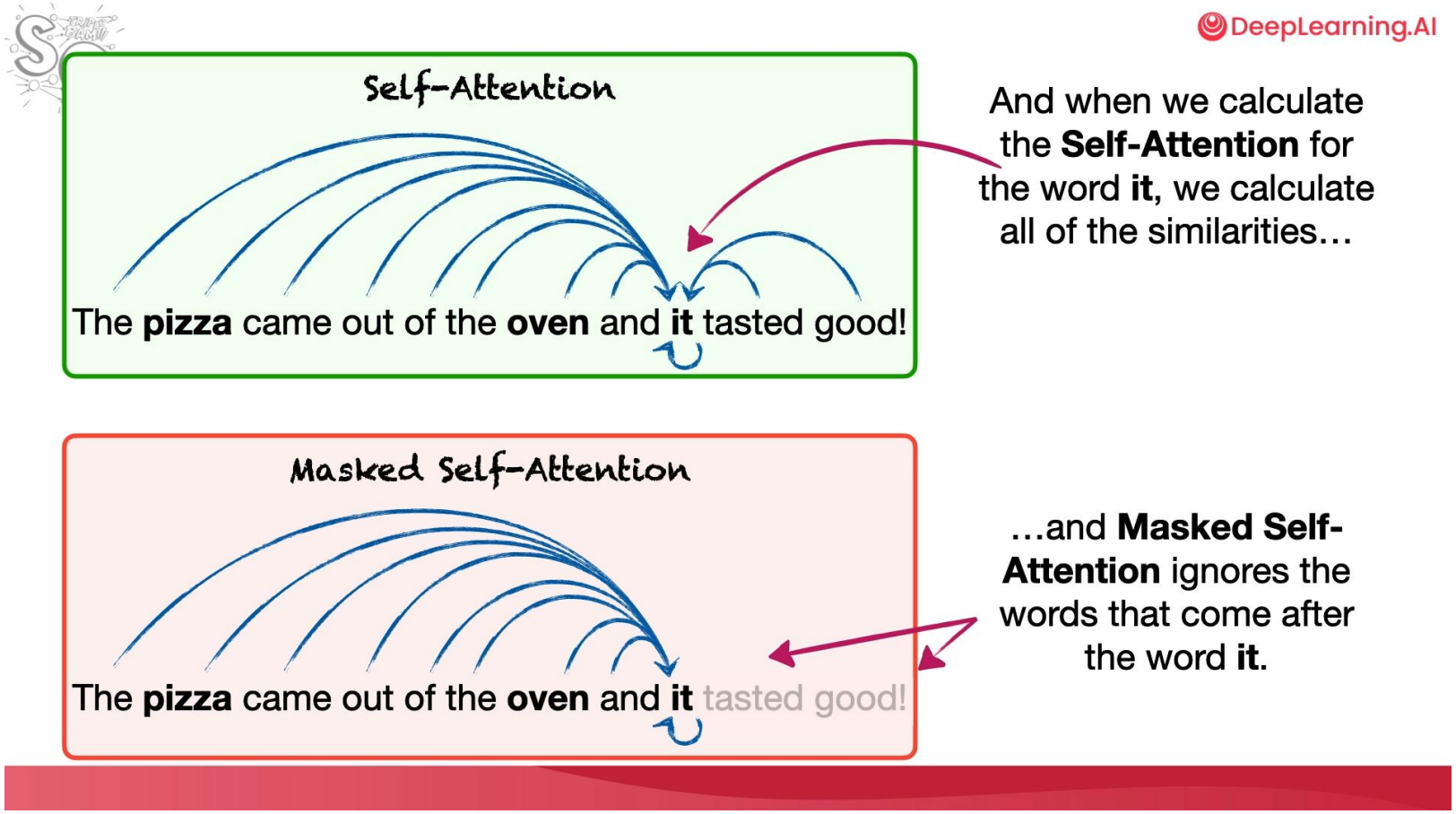
Attention Layer bestimmt für jedes Wort wie wichtig jedes davor stehende Wort im Kontext für die Bedeutung dieses Wortes ist (extrem langes und selektives Langzeit-Gedächtnis)



Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Quelle: Attention in Transformers: Concepts and Code in PyTorch, Josh Starmer, DeepLearning.AI, https://learn.deeplearning.ai/courses/attention-in-transformers-concepts-and-code-in-pytorch/lesson/ym7dj/the-main-ideas-behind-transformers-and-attention

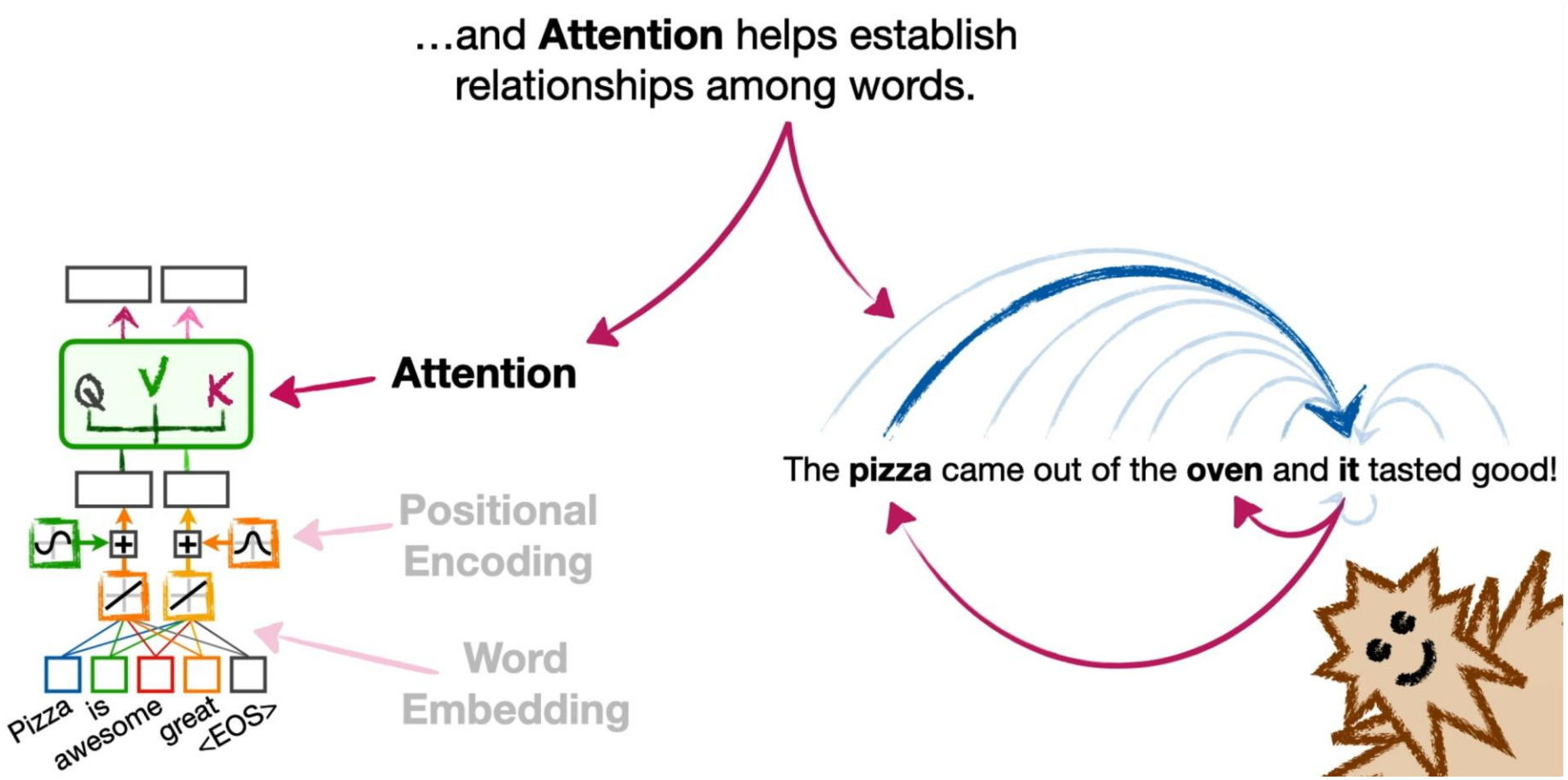
Die Transformer Architektur: Attention is all you need



Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Quelle: Attention in Transformers: Concepts and Code in PyTorch, Josh Starmer, DeepLearning.AI, https://learn.deeplearning.ai/courses/attention-in-transformers-concepts-and-code-in-pytorch/lesson/xy1tc/self-attention-vs-masked-self-attention

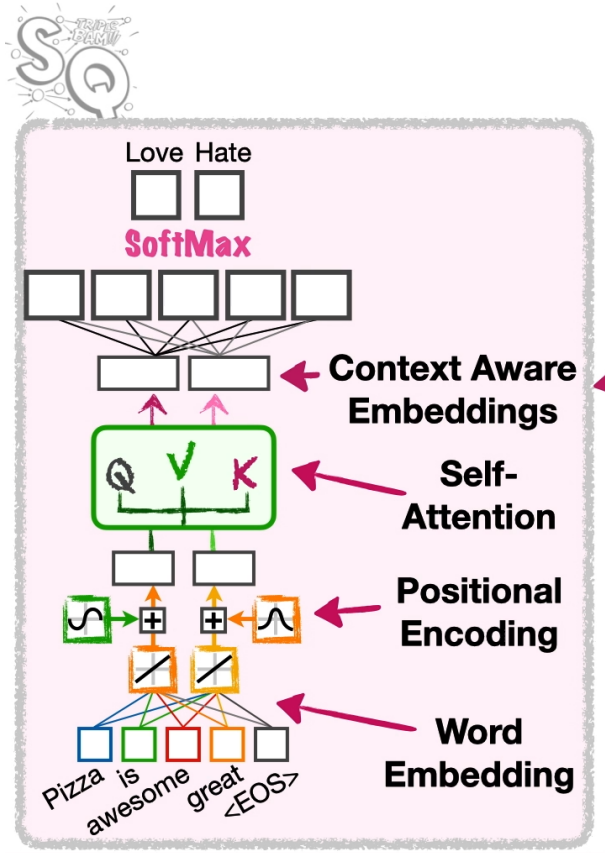
Die Transformer Architektur: Attention is all you need



Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Quelle: Attention in Transformers: Concepts and Code in PyTorch, Josh Starmer, DeepLearning.AI, https://learn.deeplearning.ai/courses/attention-in-transformers-concepts-and-code-in-pytorch/lesson/ym7dj/the-main-ideas-behind-transformers-and-attention

Die Transformer Architektur: Attention is all you need



DeepLearning.AI

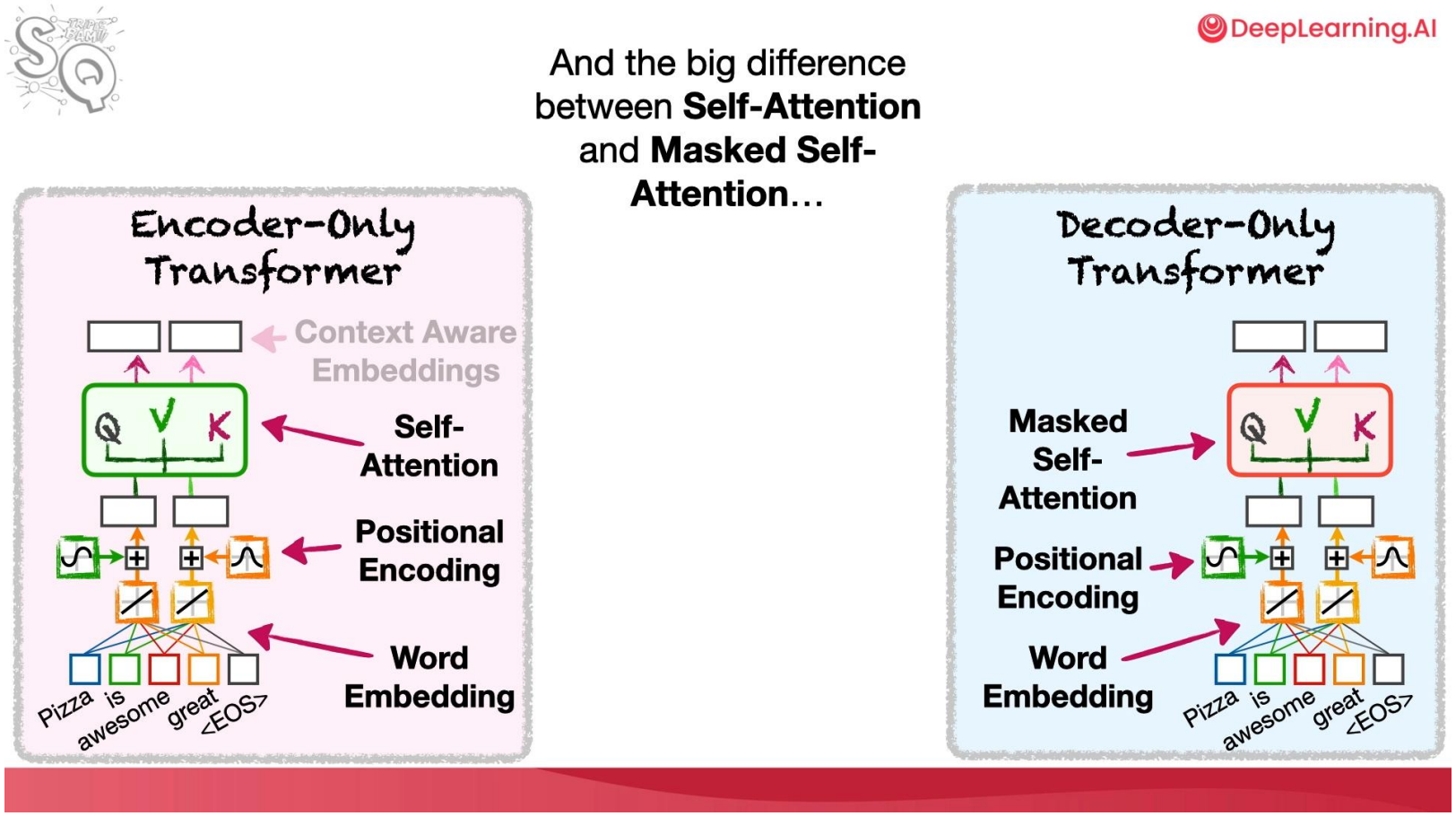
For example, we might want to see if people are posting positive of negative statements about pizza on Twitter…

…and the **Context Aware Embeddings** are great inputs for a neural network that can do that type of classification.

Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Quelle: Attention in Transformers: Concepts and Code in PyTorch, Josh Starmer, DeepLearning.AI, https://learn.deeplearning.ai/courses/attention-in-transformers-concepts-and-code-in-pytorch/lesson/xy1tc/self-attention-vs-masked-self-attention

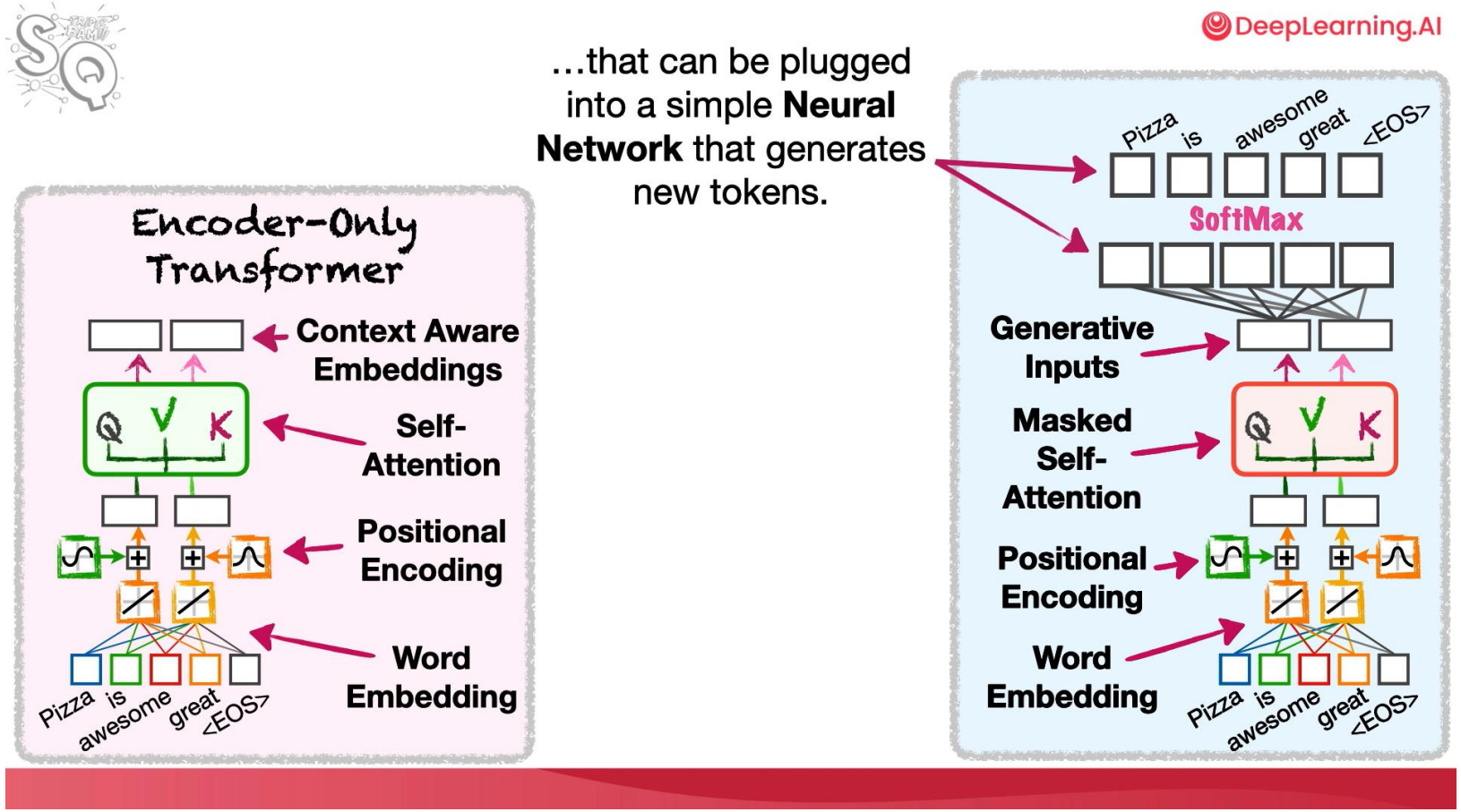
Die Transformer Architektur: Attention is all you need



Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Quelle: Attention in Transformers: Concepts and Code in PyTorch, Josh Starmer, DeepLearning.AI, https://learn.deeplearning.ai/courses/attention-in-transformers-concepts-and-code-in-pytorch/lesson/xy1tc/self-attention-vs-masked-self-attention

Die Transformer Architektur: Attention is all you need



Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Quelle: Attention in Transformers: Concepts and Code in PyTorch, Josh Starmer, DeepLearning.AI, https://learn.deeplearning.ai/courses/attention-in-transformers-concepts-and-code-in-pytorch/lesson/xy1tc/self-attention-vs-masked-self-attention

Die Transformer Architektur: Attention is all you need

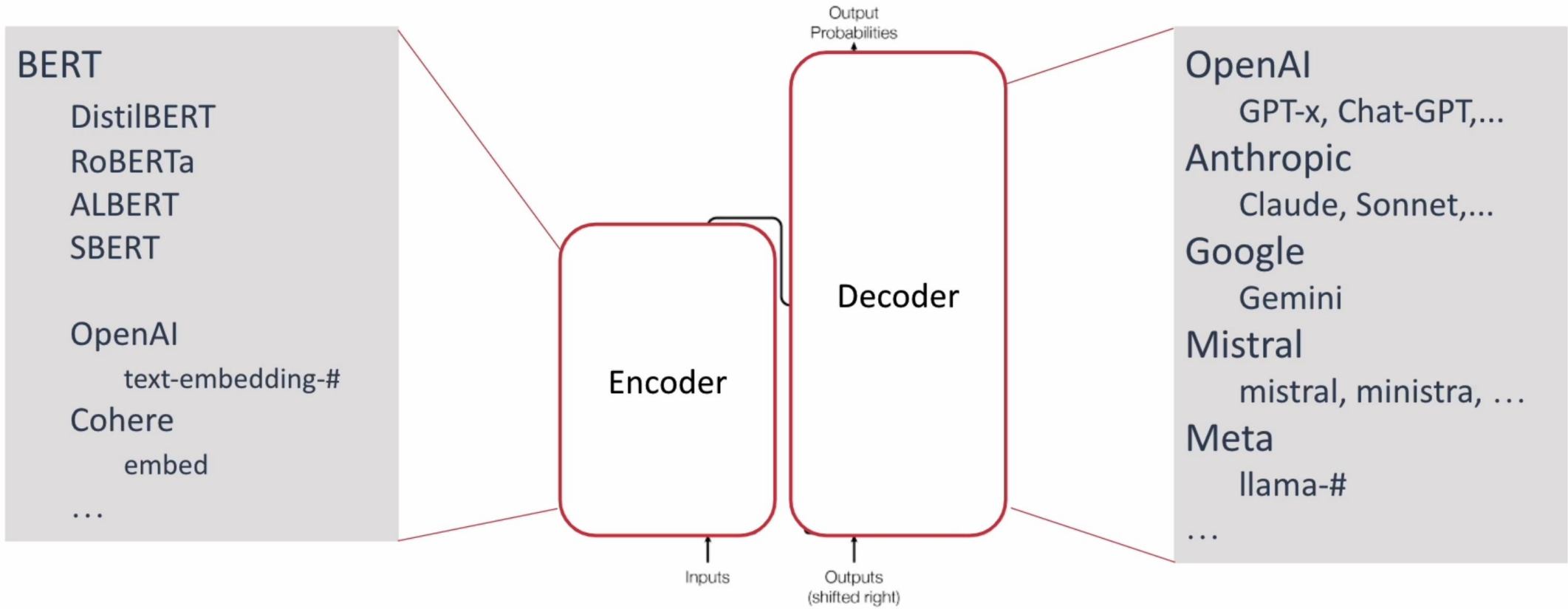


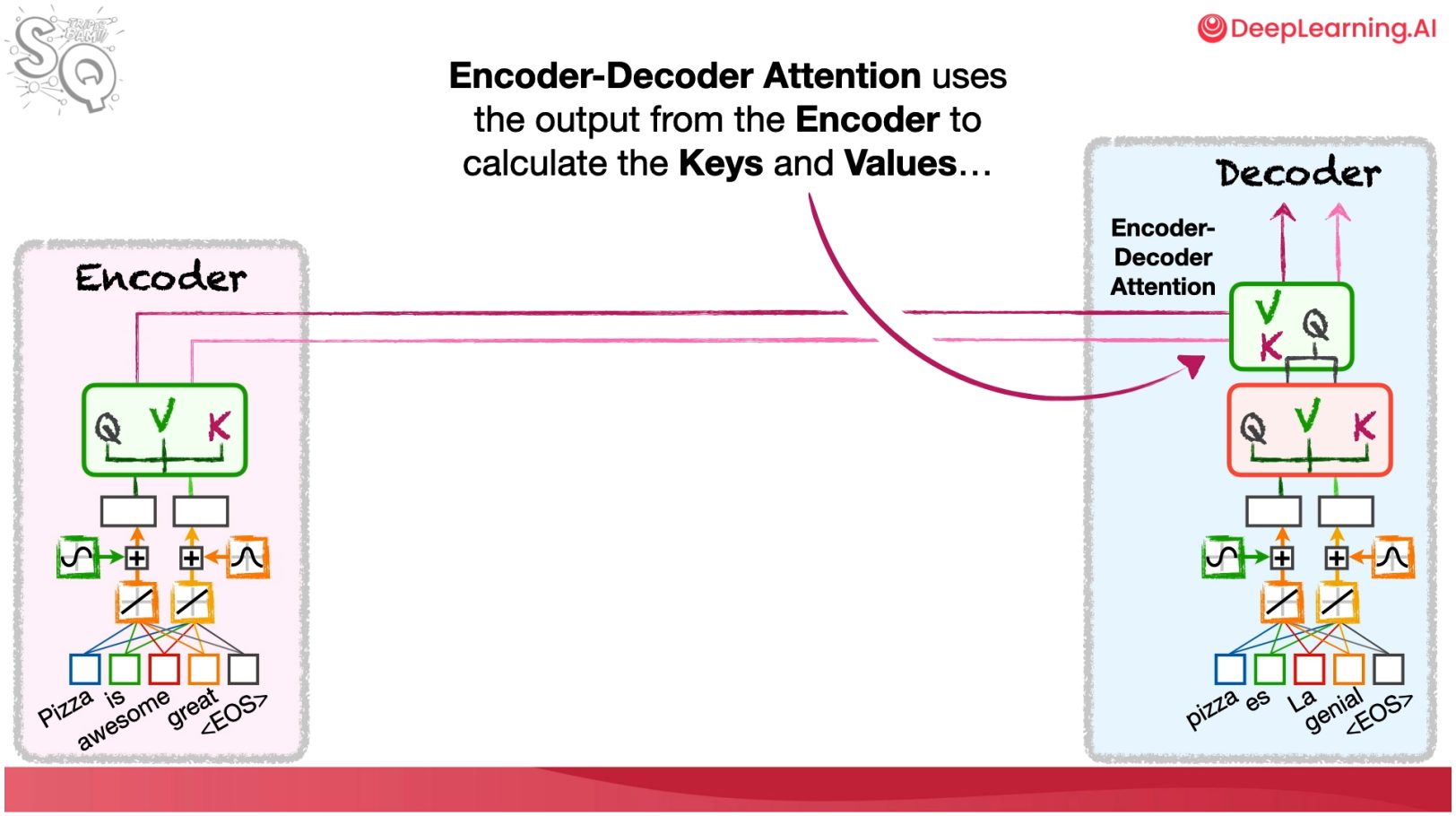
Figure 1: The Transformer - model architecture.

Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Quelle: Attention in Transformers: Concepts and Code in PyTorch, Josh Starmer, DeepLearning.AI, https://learn.deeplearning.ai/courses/attention-in-transformers-concepts-and-code-in-pytorch/lesson/han2t/introduction

Die Transformer Architektur: Attention is all you need

Beispiel: Machine Translation



Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Quelle: Attention in Transformers: Concepts and Code in PyTorch, Josh Starmer, DeepLearning.AI, https://learn.deeplearning.ai/courses/attention-in-transformers-concepts-and-code-in-pytorch/lesson/ugekb/encoder-decoder-attention

Die Transformer Architektur: Attention is all you need

Beispiel: Machine Translation: Keys (K), Values (V) und Queries (Q)

Das LLM soll ins Englische übersetzen: „Ich spiele Fußball“

Der Encoder hat den Satz encodiert. Analogie: Es hat Key-Value Paare gelernt.

Keys	Values
Subjekt	Ich
Verb	spiele
Objekt	Fußball

Annahme: Der Decoder hat bereits „Ich“ in „I“ übersetzt.

- Decoder „weiß“ aus der Kenntnis der englischen Sprache und dem deutschen Satz, dass ein Verb folgen müsste.
- Es fragt („query“) den Encoder welches Wort im deutschen Satz das Verb ist.
- Der Encoder geht alle Keys durch und liefert als Value das Verb zurück.
- Genau genommen liefert es für jeden Key einen Ähnlichkeits-Score zurück, der umso höher ist, desto besser die Query mit dem Key übereinstimmt.
  - Keys, Values und Queries sind Vektoren, die aus den Trainingsdaten gelernt werden.
- Decoder achtet (attention) beim Generieren des nächsten Wortes auf die Worte im deutschen Satz, für die ein hoher KQ-Score zurück geliefert wurde und auf die bereits generierten Worte am Satzanfang.

Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Quelle: A Géron, Hands-On Machine Learning with scikit-learn, keras and Tensorflow, 3rd edition, 2022.

Large Language Modelle: Prompt Engineering

- Warum muss man einen Prompt schreiben?
  - Ohne Prompt kann das LLM nichts vervollständigen
- Was wird mit dem Prompt gemacht?
  - Der Prompt geht in das LLM rein und dieses vervollständigt ihn
- Was sind gute Prompts?

Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Zero-Shot Prompting, Few-Shot Prompting

Zero-Shot Prompting

- Man schreibt einfach einen Prompt und LLM antwortet mit der Sequenz der wahrscheinlichsten Tokens

Few-Shot Prompting

- In dem Prompt gibt man ein paar Beispielkonversationen an, um dem LLM zu zeigen wie er antwortet könnte/soll
- One-Shot Prompting: Man gibt nur eine Beispielkonversation an

- Wie sehen gute Beispiele aus?

Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik
Technology Arts Sciences TH Köln

Beispiel für One-Shot Prompt

Here is an example:

Context:

Example context about Paris being the capital of France.

Question:

What is the capital of France?

Answer:

Paris.

Now, use the following context to answer the question.

Context:

{context_str}

Question:

{query_str}

Answer:

Eigenschaften von LLMs

Gute Beispielkonversationen

Wenn ein Token generiert wurde, kann das LLM nicht zurück und wird weitere Tokens generieren, ob diese Sinn machen oder nicht…

Models need tokens to think

Human: "Emily buys 3 apples and 2 oranges. Each orange costs $2. The total cost of all the fruit is $13. What is the cost of apples?"

Assistant: "The answer is $3. This is because 2 oranges at $2 are $4 total. So the 3 apples cost $9, and therefore each apple is 9/3 = $3".

Assistant: "The total cost of the oranges is $4. 13 - 4 = 9, the cost of the 3 apples is $9. 9/3 = 3, so each apple costs $3. The answer is $3".

Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Quelle: Andrej Karpathy, Deep Dive into LLMs like ChatGPT, https://www.youtube.com/watch?v=7xTGNNLPyMI

Eigenschaften von LLMs

Gute Beispielkonversationen

- Schreiben Sie nicht irgendeine Aussage und begründen ihn im nächsten Satz oder am Ende des Satzes.
- Sondern:
  - Leiten Sie alle Aussagen Schritt für Schritt logisch her und machen Sie erst am Ende Ihre Aussage.

Eigenschaften von LLMs

Quelle: Andrej Karpathy, Deep Dive into LLMs like ChatGPT, https://www.youtube.com/watch?v=7xTGNNLPyMI

Models can't count

Models are not good with spelling.

Remember they see tokens (text chunks), not individual letters!

Bunch of other small random stuff

What is bigger 9.11 or 9.9?

Models can (and should!) use tools!

Web search
Code (/ Python interpreter)

Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik
Technology Arts Sciences TH Köln

Eigenschaften von LLMs

Quelle: Andrej Karpathy, Deep Dive into LLMs like ChatGPT, https://www.youtube.com/watch?v=7xTGNNLPyMI

Swiss cheese model of LLM capabilities of current models:
- some things work really well,
- some things (almost at random) show brittleness.

Zusammenfassung

- Rekurrente neuronale Netze (RNNs)
  - Verarbeiten Sequenzen und besitzen ein Gedächtnis über vorherige Zeitschritte.
  - Geeignet für Zeitreihen, Text, Sprache und andere sequenzielle Daten.
  - Unterschiedliche Architekturen:
    - Sequence-to-Vector
    - Sequence-to-Sequence
    - Vector-to-Sequence
    - Encoder-Decoder
- Long Short-Term Memory (LSTM)
  - Erweiterung von RNNs zur Modellierung langfristiger Abhängigkeiten.
  - Speicher mit Kurzzeit- und Langzeitgedächtnis
  - Häufig bessere Ergebnisse als einfache RNNs.

Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik
Technology Arts Sciences TH Köln

Zusammenfassung

- Transformer und Large Language Models
  - Arbeiten mit Tokenisierung, Embeddings und Attention.
  - Attention ermöglicht die Berücksichtigung relevanter Informationen über große Kontextfenster.
  - Grundlage moderner Large Language Models wie ChatGPT.

Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik
Technology Arts Sciences TH Köln

Zusammenfassung

Modellauswahl

Problem	Typische Wahl
Tabellarische Daten	MLP
Bilder	CNN
Sequenzen/Zeitreihen	RNN/LSTM/GRU
Natural Language Processing, LLMs, Multimodalität	Transformer

Kernbotschaft
Die Wahl der Netzwerkarchitektur sollte immer von der Struktur der Daten und der Aufgabe ausgehen.

Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik
Technology Arts Sciences TH Köln

Nächste Vorlesung und Fragen

- Entwicklung eines Chatbots auf Basis eines Large Language Models
- Fragen?

Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik
Technology Arts Sciences TH Köln