

KI programmieren im Informatikunterricht Teil 3: LSTM und Reihenvorhersage

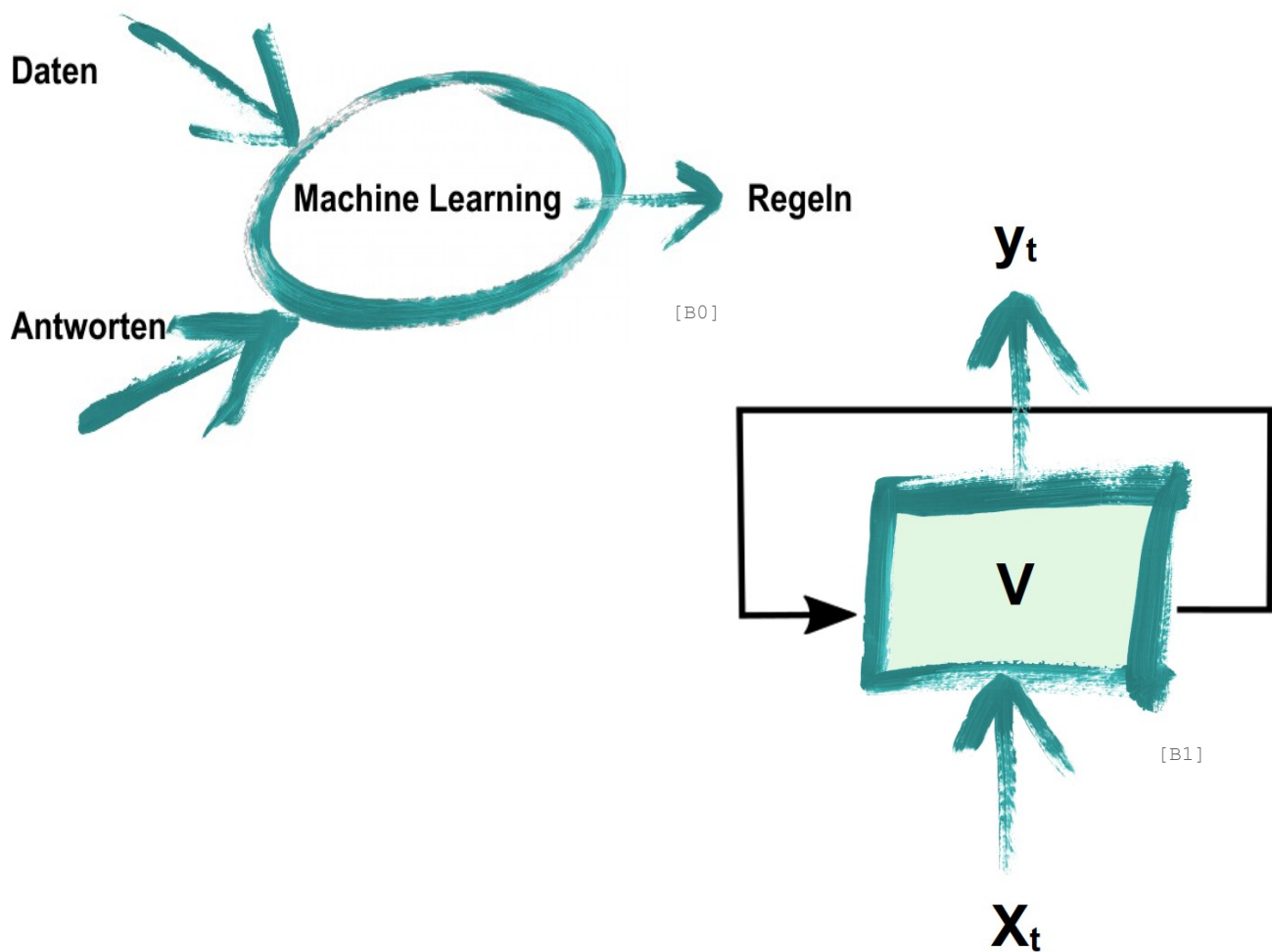


Abb. [B0]: „E-V-A“, Alexander Schindler, Lizenz [CC BY-SA 4.0](https://creativecommons.org/licenses/by-sa/4.0/), Lernaufgabe "KI im Unterricht"
Abb. [B1] „LSTM-high-level-view“, Alexander Schindler Lizenz [CC BY-SA 4.0](https://creativecommons.org/licenses/by-sa/4.0/), Lernaufgabe „KI im Unterricht III – LSTM“

Inhaltsverzeichnis

KI programmieren im Informatikunterricht Teil 3: LSTM und Reihenvorhersage.....	1
A Übersicht.....	3
A1 Zusammenfassung:.....	3
A2 Stundenübersicht.....	4
A3 Themeneinstieg und theoretische Grundlagen.....	5
B Lernaufgabe.....	6
B1 Unterrichtsbegleitende Präsentation.....	6
B2 Arbeitsblätter.....	19
C Bezug zum Rahmenlehrplan.....	21
C1 Didaktischer Kommentar.....	21
C2 Bezüge zum Rahmenlehrplan Informatik.....	22
C3 Bezüge zum Basiscurriculum Medienbildung.....	24
C4 Bezüge zu übergreifenden Themen.....	24
C5 Bezüge zu anderen Fächern:.....	25
D Anhang.....	26
D1 „Material“ für den Einsatz dieser Lernaufgabe.....	26
D2 Hinweise / Musterlösung der Lernaufgabe.....	28
D3 Quellen / Lektüreliste zum Weiterlesen.....	29
D4 Bildnachweise.....	30

A Übersicht

A1 Zusammenfassung:

Wer neuronale Netze einmal selbst trainiert hat, kann deren Risiken, Probleme und Chancen und damit auch mögliche gesellschaftliche Entwicklungen besser einschätzen.

In dieser Lernaufgabe lernen die Schüler:innen den Aufbau von neuronalen Netzen (NN) mithilfe von TensorFlow, sodass Aussagen über Reihen mit Hilfe von LSTMs (*long short term memory*) gemacht werden können. Im Anschluss ist es den Schüler:innen möglich, Modelle in TensorFlow für eigene Projekte zu entwerfen und zu trainieren.

Diese Lernaufgabe setzt die Grundlagen der ersten Lernaufgabe: „KI programmieren im Informatikunterricht Teil I: Einführung“ voraus. Weiter Voraussetzungen sind: Grundlagen der Programmierung mit Python, das Einbinden von Bibliotheken und der Umgang mit einer entsprechend leistungsfähigen IDE wie z. B. Spyder.

In diesem Sinne: *All hands on code!*

Intention der Lernaufgabe

- Kennenlernen der Grundlagen von LSTM
- Kennenlernen / Erstellung / Programmierung eines eigenen NN mithilfe von TensorFlow, um folgende Werte einer Reihe vorherzusagen
- Erkennen von Chancen und Risiken und möglichen gesellschaftlicher Auswirkungen, wenn NN die „Zukunft vorhersagen können“

Hinweise:

Das Zusammenspiel von TensorFlow und Python läuft nicht immer problemlos. Bei unklaren Schwierigkeiten hilft es, eine eher ältere Pythonversion mit einer eher

neueren TensorFlow-Version zu benutzen. Diese Aufgaben wurden mit Python 3.8.10 (64 Bit) und TensorFlow 2.6.0 getestet.

A2 Stundenübersicht

1. Doppelstunde:

- Hinführung zum Thema
- Grundlagen von LSTM
- Programmierübungen
- weiterführend: eigenständiges Projekt

1. Doppelstunde

Zeit	Phase	Beschreibung	Methode/Medien/ Anmerkung
5 Min.	Einstieg	Fehlendes Wort ergänzen. <ul style="list-style-type: none"> • Welche Informationen sind wichtig? • Welche können vergessen werden? • Wie treffen Menschen Aussagen über die Zukunft? 	EH: Je weiter die Information zurückliegt, umso schwieriger die Vorhersage. EH: Nur wenige Informationen werden für Vorhersagen benötigt.
10 Min.	Erarbeitung 1	Lehrervortrag zu Grundlagen LSTM Forschungsauftrag: Beschreibe die grundsätzliche Idee eines LSTM in eigenen Worten.	Präsentation: <i>LSTM</i> , Vortrag / LSG
10 Min.	Sicherung 1	Besprechung Forschungsauftrag. Besprechung Quellcode	
45 Min.	Erarbeitung 2	Aufgaben 1, 2, 3 als eigenständige Programmieraufgaben. Erstellung eigener Reihen, siehe	Für Aufgabe 2 empfiehlt sich ein Blick in die TensorFlow API https://www.tensorflow.org/ap

Zeit	Phase	Beschreibung	Methode/Medien/ Anmerkung
		auch die Codesnippets in Kapitel D2 in diesem Dokument	i_docs/python/tf/all_symbols oder die etwas übersichtlichere API von Keras https://keras.io/api/
10 Min.	Sicherung 2	Diskussion der Ergebnisse	
35 Min.	Transfer	Aufgabe 4 / 5 Programmieraufgaben	Eigenständige Programmieraufgabe Veränderung am bestehenden Modell zur Verbesserung der Vorhersage möglich.
	Ausblick	Zusatzaufgabe Planung / Erstellung eines eigenen Projektes mit Anwendung z. B. Vorhersage von Aktienkursen	Diese Aufgabe kann sicherlich auch mehrere Unterrichtsstunden in Anspruch nehmen.

EH: Erwartungshorizont

A3 Themeneinstieg und theoretische Grundlagen

Der Themeneinstieg erfolgt über eine kurze Diskussion zur ersten Seite der unterrichtsbegleitenden Präsentation. Dabei sollte problematisiert werden, dass man sich bei der „Vorhersage der Zukunft“ an einige Fakten erinnern muss, andere wiederum ignorieren oder vergessen kann. Der Themeneinstieg hängt auch von der jeweiligen Lerngruppe ab und kann gern variiert werden.

Weitere theoretische Grundlagen finden sich in der Lernaufgabe „KI programmieren im Informatikunterricht Teil I: Einführung“. Für zusätzliche Hinweise empfiehlt sich die Lektüre der Quellen wie unter: „D3 Quellen / Lektüreliste zum Weiterlesen“ aufgeführt.



CC BY SA 4.0
Ausgenommen sind einzeln gekennzeichnete Inhalte/Elemente, siehe Quellen- und Lizenzhinweise am Ende des Dokuments.

iMINT Akademie Fachset Informatik
KI Programmieren im Unterricht Teil 3:
Reihenvorhersage mit LSTM

Senatsverwaltung
für Bildung, Jugend
und Familie

BERLIN



B Lernaufgabe



B Lernaufgabe

B1 Unterrichtsbegleitende Präsentation



CC BY SA 4.0
Ausgenommen sind einzeln gekennzeichnete Inhalte/Elemente, siehe Quellen- und Lizenzhinweise am Ende des Dokuments.

iMINT Akademie Fachset Informatik
KI Programmieren im Unterricht Teil 3:
Reihenvorhersage mit LSTM

Senatsverwaltung
für Bildung, Jugend
und Familie

BERLIN



forget – input – output

Die Wolken sind im <Wort>

Frankreich ist das Land, in dem ich aufgewachsen bin, deswegen mag ich Weißbrot, Käse, und ich spreche fließend <Wort>.



[CC BY SA 4.0](https://creativecommons.org/licenses/by-sa/4.0/)

Ausgenommen sind einzeln gekennzeichnete Inhalte/Elemente, siehe Quellen- und Lizenzhinweise am Ende des Dokuments.

iMINT Akademie Fachset Informatik
KI Programmieren im Unterricht Teil III: Reihenvorhersage mit LSTM
A. Schindler [Lizenz CC BY-SA 4.0](https://creativecommons.org/licenses/by-sa/4.0/)

Stand: 14.09.2022

Senatsverwaltung
für Bildung, Jugend
und Familie

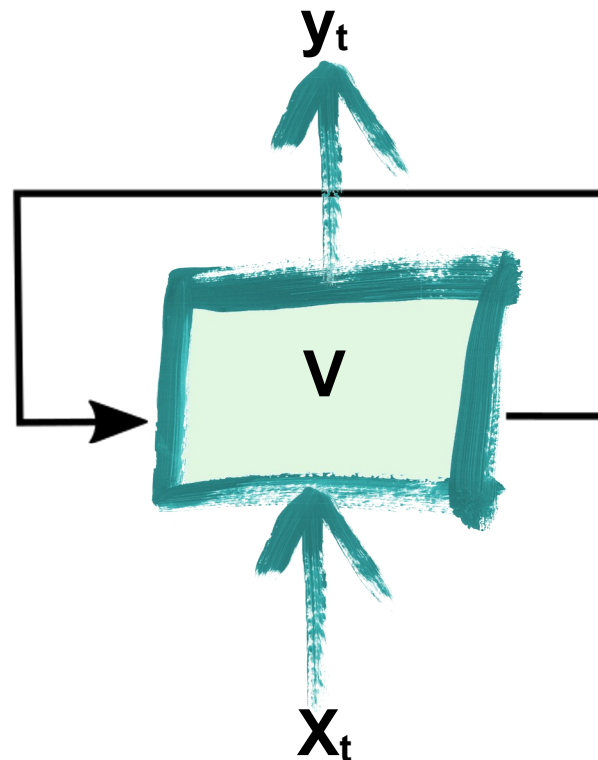
BERLIN



1. LSTM - Long Short Term Memory

- LSTM können sich Informationen über lange Zeiträume merken und entscheiden selbst wie lange sie bestimmte Informationen erinnern.
- LSTM gehören zu den Recurrent Neural Network RNN.
- Je kleiner / größer der Abstand zwischen der relevanten Information und dem Punkt an dem sie gebraucht wird ist, umso leichter / schwieriger ist es, für ein RNN eine Vorhersage zu treffen.
- LSTM gibt es in vielen Varianten, z. B.:
 - GRU - Gated Recurrent Unit
 - DRNN - Deep Recurrent Neuronal Network

1. RNN Recurrent Neural Network



RNNs haben Schleifen und können sich erinnern.

[B2] „LSTM-high-level-view“, Alexander Schindler Lizenz [CC BY-SA 4.0](https://creativecommons.org/licenses/by-sa/4.0/), Lernaufgabe „KI im Unterricht III - LSTM“



[CC BY SA 4.0](https://creativecommons.org/licenses/by-sa/4.0/)

Ausgenommen sind einzeln
gekennzeichnete Inhalte/Elemente, siehe
Quellen- und Lizenzhinweise am Ende des
Dokuments.

iMINT Akademie Fachset Informatik
KI Programmieren im Unterricht Teil III: Reihenvorhersage mit LSTM
A. Schindler [Lizenz CC BY-SA 4.0](https://creativecommons.org/licenses/by-sa/4.0/)

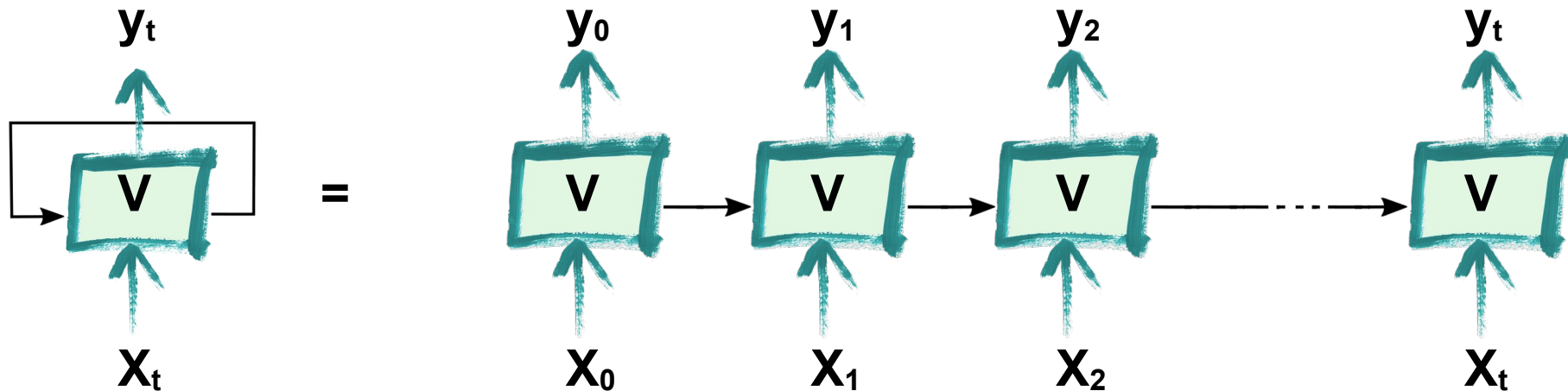
Stand: 14.09.2022

Senatsverwaltung
für Bildung, Jugend
und Familie

BERLIN



1. RNN ausgerollt



x_t : input zum Zeitpunkt t

x_0 : input zum Zeitpunkt 0

V : Verarbeitung

y_t : output der LSTM-Einheit zum Zeitpunkt t

y_0 : output der LSTM-Einheit zum Zeitpunkt 0

[B3] „LSTM-high-level-view ausgerollt“, Alexander Schindler Lizenz [CC BY-SA 4.0](https://creativecommons.org/licenses/by-sa/4.0/), Lernaufgabe „KI im Unterricht III - LSTM“



[CC BY SA 4.0](https://creativecommons.org/licenses/by-sa/4.0/)

Ausgenommen sind einzeln gekennzeichnete Inhalte/Elemente, siehe Quellen- und Lizenzhinweise am Ende des Dokuments.

iMINT Akademie Fachset Informatik
KI Programmieren im Unterricht Teil III: Reihenvorhersage mit LSTM
A. Schindler [Lizenz CC BY-SA 4.0](https://creativecommons.org/licenses/by-sa/4.0/)

Stand: 14.09.2022

Senatsverwaltung
für Bildung, Jugend
und Familie

BERLIN

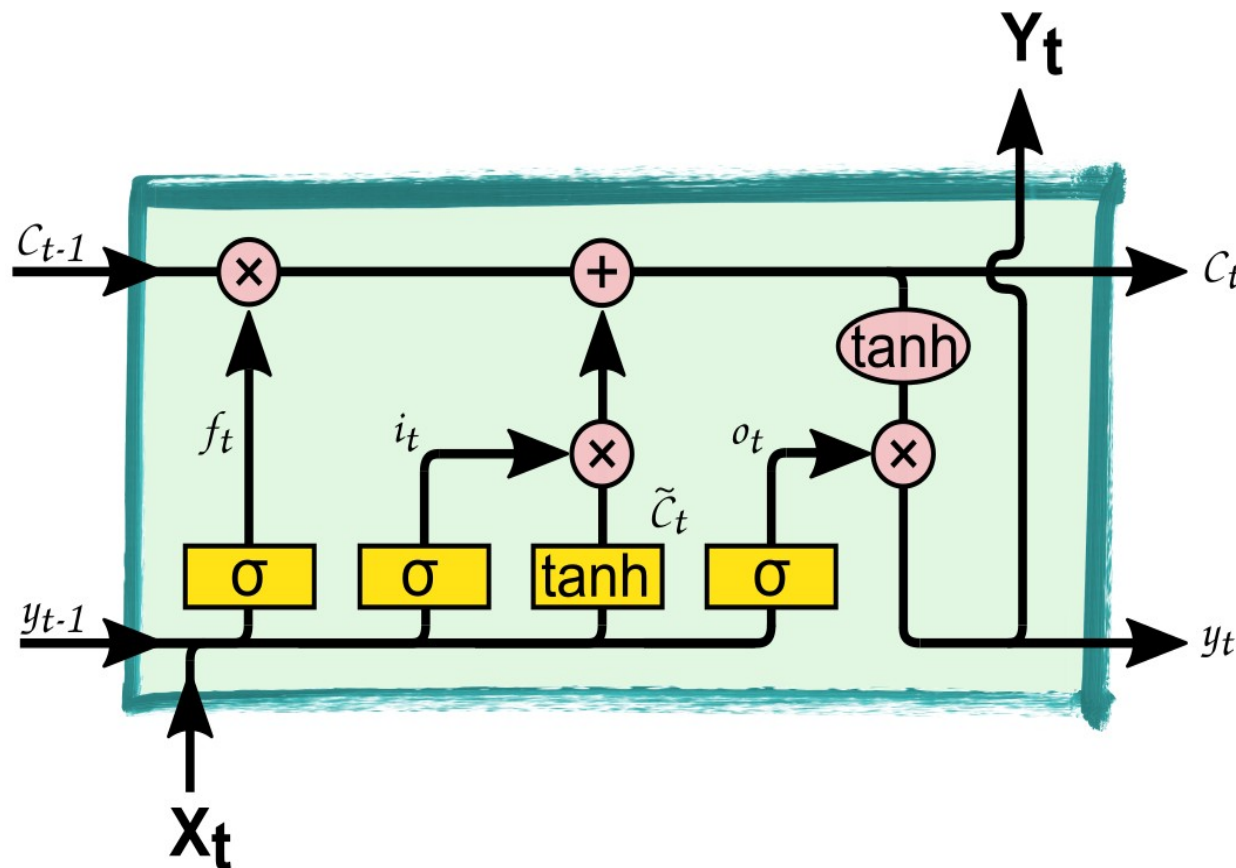







10

1. LSTM Aufbau

- C (Zellzustand): Im LSTM wird Information zu C hinzugefügt oder gelöscht. C wird durch *gates* vorsichtig gesteuert.
- LSTM besteht aus C + 3 *gates* (forget, input, output) + tanh
- *gates*: lassen (optional) Informationen hindurch. Sie schützen und kontrollieren C.
- *gates* bestehen aus einem sigmoid NN-Layer und einer punktwweisen Multiplikation
- tanh: erzeugt Werte zwischen -1 und 1
sigmoid: erzeugt Werte zwischen 0 und 1

1. LSTM zum Zeitpunkt t



-  NN - Layer
-  punktweise Operation
-  Vektorübergabe
-  anhängen
-  kopieren
- \mathbf{x}_t : input
- \mathbf{y}_t : output
- \mathbf{c}_t : Zellzustand
- \mathbf{f}_t : forget gate
- \mathbf{i}_t : input gate
- \mathbf{o}_t : output gate
- $\tilde{\mathbf{c}}_t$: Zell input activation Vektor

[B1] „LSTM-low-level-view“, Alexander Schindler Lizenz [CC BY-SA 4.0](https://creativecommons.org/licenses/by-sa/4.0/), Lernaufgabe „KI im Unterricht III - LSTM“

1. LSTM - Vorteile

- LSTMs erinnern sich auch über längere Zeiträume mit Hilfe des Zellzustand C_t .
- NN multiplizieren Zahlen > 0 und < 1 . Über längere Zeiträume kann es so passieren, dass die Zahlen verschwinden (*vanishing gradient problem*). LSTM bietet eine Lösung für dieses Problem.



[CC BY SA 4.0](https://creativecommons.org/licenses/by-sa/4.0/)

Ausgenommen sind einzeln gekennzeichnete Inhalte/Elemente, siehe Quellen- und Lizenzhinweise am Ende des Dokuments.

iMINT Akademie Fachset Informatik
KI Programmieren im Unterricht Teil III: Reihenvorhersage mit LSTM
A. Schindler [Lizenz CC BY-SA 4.0](https://creativecommons.org/licenses/by-sa/4.0/)

Stand: 14.09.2022

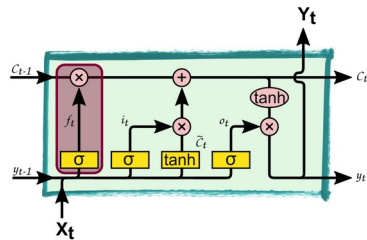
Senatsverwaltung
für Bildung, Jugend
und Familie

BERLIN



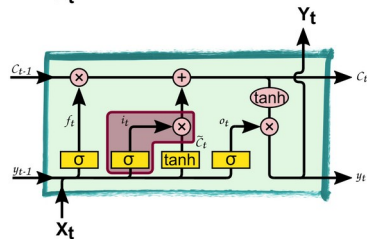
13

1. LSTM - Schritt für Schritt

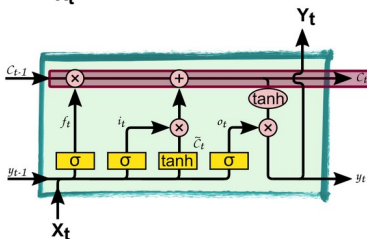


forget gate $f_t = \sigma (W_f \cdot [y_{t-1}, x_t] + b_f)$

W: Gewicht
b: bias

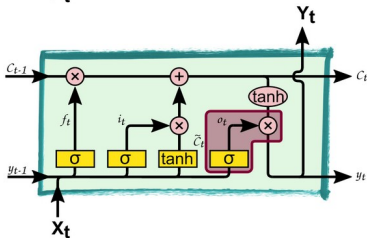


input gate $i_t = \sigma (W_i \cdot [y_{t-1}, x_t] + b_i)$
 $\tilde{C}_t = \tanh (W_c \cdot [y_{t-1}, x_t] + b_c)$



update Zellzustand

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$



output Gate $o_t = \sigma (W_o \cdot [y_{t-1}, x_t] + b_o)$
 $y_t = o_t * \tanh (C_t)$

[B4] „LSTM-low-level-view with gates and cell“, Alexander Schindler Lizenz [CC BY-SA 4.0](https://creativecommons.org/licenses/by-sa/4.0/), Lernaufgabe „KI im Unterricht III - LSTM“

2. All hands on Code

```
1  # Den nächsten Wert einer Reihe vorhersagen.
2  # version: 1.0 date: 10.4.2022
3
4  import numpy as np
5  import tensorflow as tf
6
7  trainInput  = np.asarray(
8      [ [0.01],[0.02],[0.03],[0.04],[0.05]],
9      [ [0.02],[0.03],[0.04],[0.05],[0.06]],
10     [ [0.03],[0.04],[0.05],[0.06],[0.07]],
11     [ [0.04],[0.05],[0.06],[0.07],[0.08]],
12     [ [0.05],[0.06],[0.07],[0.08],[0.09]],
13     [ [0.06],[0.07],[0.08],[0.09],[0.1]],
14     [ [0.07],[0.08],[0.09],[0.10],[0.11]],
15     [ [0.08],[0.09],[0.10],[0.11],[0.12]],
16     [ [0.09],[0.10],[0.11],[0.12],[0.13]],
17     [ [0.10],[0.11],[0.12],[0.13],[0.14]] ] ).astype('float32')
18
19  trainOutput = np.asarray([0.06, 0.07, 0.08, 0.09, 0.10, 0.11, 0.12, 0.13, 0.14,
20  0.15]).astype('float32')
```



[CC BY SA 4.0](https://creativecommons.org/licenses/by-sa/4.0/)

Ausgenommen sind einzeln
gekennzeichnete Inhalte/Elemente, siehe
Quellen- und Lizenzhinweise am Ende des
Dokuments.

iMINT Akademie Fachset Informatik
KI Programmieren im Unterricht Teil III: Reihenvorhersage mit LSTM
A. Schindler [Lizenz CC BY-SA 4.0](https://creativecommons.org/licenses/by-sa/4.0/)

Stand: 14.09.2022

Senatsverwaltung
für Bildung, Jugend
und Familie

BERLIN



15

21

2. All hands on Code

22

```
23 # model definieren
24 model = tf.keras.Sequential()
25 model.add(tf.keras.layers.LSTM(100, activation='relu', kernel_initializer='he_normal',
    input_shape=(5,1)))
26 model.add(tf.keras.layers.Dense(50, activation='relu', kernel_initializer='he_normal'))
27 model.add(tf.keras.layers.Dense(50, activation='relu', kernel_initializer='he_normal'))
28 model.add(tf.keras.layers.Dense(1))
29
30 model.compile(optimizer='adam', loss='mse', metrics=['mae'])
31
32 # model trainieren
33 model.fit(trainInput, trainOutput, epochs=200)
34
35 # model anwenden
36 , = np.asarray([ [0.14], [0.15], [0.16], [0.17], [0.18]] ]).astype('float32')
37 vorhersage = model.predict(testListe)
38 print('Nächste Zahl: %.4f' % (vorhersage))
```

Code: tf_lstm_start.py



[CC BY SA 4.0](#)

Ausgenommen sind einzeln
gekennzeichnete Inhalte/Elemente, siehe
Quellen- und Lizenzhinweise am Ende des
Dokuments.

iMINT Akademie Fachset Informatik
KI Programmieren im Unterricht Teil III: Reihenvorhersage mit LSTM
A. Schindler [Lizenz CC BY-SA 4.0](#)

Stand: 14.09.2022

Senatsverwaltung
für Bildung, Jugend
und Familie

BERLIN



16

3. Aufgaben

1. Führe den Code aus und beobachte das Ergebnis.
2. Führe Anpassungen am Code (`model`, `epochs`, `kernel_initializer`) durch um das Ergebnis zu verbessern.
3. Erstelle eigene Methoden, um Daten zu generieren:
 - a) linear zufällig steigend / fallend
 - b) Kreisfunktionen z. B. mit `math.sin(...)`
 - c) Kombination aus mehreren Funktionen.
4. Lasse das NN den jeweils nächsten Wert vorhersagen.
5. Ergänze die `testListe` mit Deinem vorhergesagten Wert, um den übernächsten Wert vorherzusagen. Wieviele Werte kann Dein `model` sinnvoll vorhersagen?

4. Diskussion

Welche Auswirkungen hat es, wenn Dein Smartphone Dir ständig einen Schritt voraus ist und genau weiß, was Du als nächstes tun wirst?



[CC BY SA 4.0](https://creativecommons.org/licenses/by-sa/4.0/)

Ausgenommen sind einzeln gekennzeichnete Inhalte/Elemente, siehe Quellen- und Lizenzhinweise am Ende des Dokuments.

iMINT Akademie Fachset Informatik
KI Programmieren im Unterricht Teil III: Reihenvorhersage mit LSTM
A. Schindler [Lizenz CC BY-SA 4.0](https://creativecommons.org/licenses/by-sa/4.0/)

Stand: 14.09.2022

Senatsverwaltung
für Bildung, Jugend
und Familie

BERLIN

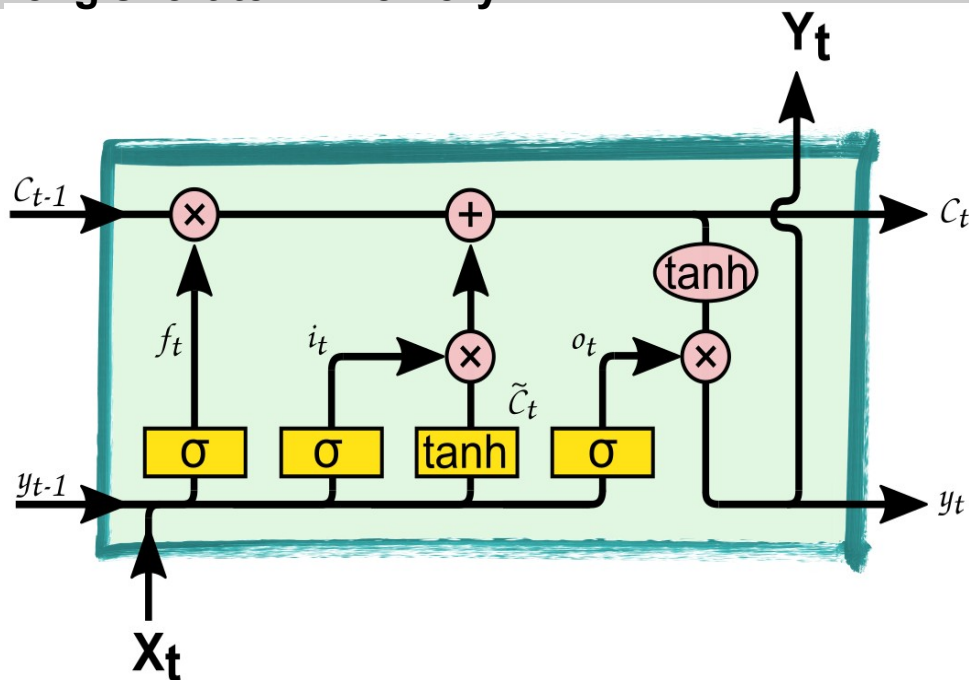


18



CC BY SA 4.0
Ausgenommen sind einzeln gekennzeichnete Inhalte/Elemente, siehe Quellen- und Lizenzhinweise am Ende des Dokuments.

LSTM – long short term memory



[B1]

Abb. 1: Aufbau eines LSTM: Ein LSTM besteht aus vier Layern und dem Zellzustand, der durch drei *gates* geschützt und kontrolliert wird. Ein *gate* besteht aus einem σ - *layer* und einer punkweisen Multiplikation. Ziel eines LSTM ist es, eine Reihenvorhersage zu treffen aufgrund von Ereignissen, die auch lange zurückliegen. Dies gelingt durch den Zellstatus. Abk.: C_t : Zellzustand, X_t : input, f_t : forget gate, i_t : input gate, o_t : output gate, y_t : output

Aufgaben:

1. Führe den Code aus und beobachte das Ergebnis.
2. Führe Anpassungen am Code (`model`, `epochs`, `kernel_initializer`) durch, um das Ergebnis zu verbessern.
3. Erstelle eigene Methoden um Daten zu generieren:
 - a) linear zufällig steigend / fallend
 - b) Kreisfunktionen z. B. mit `math.sin(...)`
 - c) Kombination aus mehreren Funktionen.
4. Lasse das NN den jeweils nächsten Wert vorhersagen.
5. Ergänze die `testListe` mit Deinem vorhergesagten Wert, um den übernächsten Wert vorherzusagen. Wieviele Werte kann Dein `model` sinnvoll vorhersagen?

Abb. 1: [B1] „LSTM-low-level-view“, Alexander Schindler Lizenz [CC BY-SA 4.0](https://creativecommons.org/licenses/by-sa/4.0/), Lernaufgabe „KI im Unterricht III - LSTM“

C Bezug zum Rahmenlehrplan

C1 Didaktischer Kommentar

Ziel der Aufgabe ist es, den Schülern einen Einstieg in die Programmierung von Neuronalen Netzwerken (NN) mit LSTMs zu ermöglichen.

Der Code ist so einfach wie möglich gehalten. Einige Vorerfahrungen mit NN, python, Softwareentwicklung, Nutzung einer IDE sollten aber vorhanden sein, somit ist die Aufgabe für Schüler der Sek II besser geeignet. Der Code bietet Anknüpfungspunkte zum Arbeiten mit Schleifen, if-Abfragen und Listen. Die Aufgabe ist skalierbar, d.h. sie kann um eigene Ideen (z. B. Datensätze, Problemstellungen) erweitert werden, und sie kann und soll somit Basis für eigene Projekte sein.

Leistungsschwächere SuS können mit dem bestehenden Quellcode experimentieren und dort die Zahlen des von `trainInput` verändern, während Leistungsstarke SuS eigene Projekte entwickeln können. Die Verwendung eigener Datenquellen und eines selbständig erstellten oder abgewandelten `model` ist sicherlich anspruchsvoll aber möglich.

Nicht zuletzt ist mit der Thematisierung von Reihenvorhersagen (*time series forecasting*) durch KI ein Ausblick auf die Auswirkungen von KI auf die eigene Lebenswelt der Schüler möglich und somit ist die Aufgabe auch im Bereich Informatik und Gesellschaft zu verorten.

Lernvoraussetzungen	<ul style="list-style-type: none"> • Die SuS können einfache Algorithmen in Python programmieren • oder: besitzen vertiefte Programmierkenntnisse in einer anderen Programmiersprache. • Die SuS können mit Variablen umgehen. • Die SuS haben eine grundlegende Vorstellung von Algorithmen und Datenstrukturen. • Die SuS können eine IDE benutzen. • Die SuS haben ein grundsätzliches Verständnis für ML wie in der Lernaufgabe: „KI programmieren im Informatikunterricht Teil 1: Einführung“
---------------------	--

C2 Bezüge zum Rahmenlehrplan Informatik

Kompetenzen	Standards (Die Schülerinnen und Schüler können....)
Wechselwirkungen zwischen Informatiksystemen, Mensch und Gesellschaft beurteilen	<p>Bezug zum RLP Sek I:</p> <p>Kompetenzbereich: Wechselwirkungen zwischen Informatiksystemen, Mensch und Gesellschaft beurteilen</p> <p>Kompetenz: Historische und aktuelle Entwicklungen der Informatik beurteilen</p> <p>Standard H: aktuelle Entwicklungen bewerten, aus ihnen mögliche Trends ableiten und Auswirkungen in der Zukunft beschreiben, die Chancen und Risiken der modernen Entwicklungen für eine demokratische Gesellschaft bewerten</p>
Mit Informationen umgehen	<p>Bezug zum RLP Sek II</p> <ul style="list-style-type: none"> Themenfeld 4.3 Softwareentwicklung: Algorithmen und Datenstrukturen
Informatiksysteme verstehen	<p>Bezug zum RLP Sek I:</p> <p>Kompetenzbereich: Informatiksysteme verstehen</p> <p>Kompetenz: Technische Grundlagen erläutern und anwenden</p> <p>Standard G/H: ein einfaches Informatiksystem entwerfen, modifizieren bzw. realisieren, z. B.: Verkehrsampelmodell, Robotermodelle</p>
Informatisches Modellieren	<p>Bezug zum RLP Sek I:</p> <p>Kompetenzbereich: Informatisches Modellieren</p> <p>Kompetenz: Informatische Modelle analysieren und bilden</p> <p>Standard F: informatische Modelle als reduzierte Abbildung der realen Welt beschreiben und beurteilen</p> <p>Standard H: beurteilen, ob das selbst erstellte Modell problemadäquat ist</p> <p>Bezug zum RLP Sek II</p> <ul style="list-style-type: none"> Vertiefungsgebiet V5 künstliche Intelligenz (IN3, in-3)
Problemlösen	Bezug zum RLP Sek I:

	Kompetenzbereich: Problemlösen Kompetenz: Programme entwerfen und realisieren Standard F: eine Programmierumgebung verwenden Standard H: Algorithmen entwerfen, implementieren und beurteilen
Kommunizieren und kooperieren	Bezug zum RLP Sek I: Kompetenzbereich: Kommunizieren und Kooperieren – Teamarbeit organisieren und koordinieren Kompetenz: Arbeitsergebnisse dokumentieren und präsentieren Standard G: adressatengerecht mit Softwareunterstützung präsentieren

Unterrichtsfach	Informatik
Jahrgangsstufe/n	Sek I: 10 Sek II: IN-3
Niveaustufe/n	Bezug zum RLP Sek I: Kompetenzbereich: Informatisches Modellieren Kompetenz: Informatische Modelle analysieren und bilden Standard F: informatische Modelle als reduzierte Abbildung der realen Welt beschreiben und beurteilen Standard G: ein Modell selbst erstellen Standard H: beurteilen, ob das selbst erstellte Modell problemadäquat ist Bezug zum RLP Sek II: <ul style="list-style-type: none"> 3. Kurshalbjahr (IN-3) Grundlagen der Informatik und Vertiefungsgebiet: V5 Künstliche Intelligenz Themenfeld 4.3 Softwareentwicklung: Algorithmen und Datenstrukturen
Zeitraumen	Eine Doppelstunde.
Thema	Reihenvorhersage (<i>time series forecasting</i>) mit Neuronalen

	Netzen.
--	---------

Kontext	<ul style="list-style-type: none"> • Beruf und Arbeitswelt • Überwachung • Robotik • autonome Agenten • Gesellschaft
Schlagwörter	Programmierung, Informatik und Gesellschaft, Informatiksystem verstehen, Problemlösen, mit Informationen umgehen, Modellbildung, künstliche Intelligenz, KI, Deep Learning, DL, TensorFlow, Keras, Python, Daten

C3 Bezüge zum Basiscurriculum Medienbildung¹

Standards des BC Medienbildung	Die Schülerinnen und Schüler können ...
Präsentieren	<ul style="list-style-type: none"> • Arbeitsergebnisse vorstellen • Reflektierter Technologieeinsatz
Reflektieren	<ul style="list-style-type: none"> • Chancen und Risiken von Geschäftsaktivitäten im Internet untersuchen und Schlussfolgerungen für eigene Geschäftsaktivitäten ziehen. • Die Möglichkeiten und Methoden medialer Manipulation exemplarisch analysieren.

C4 Bezüge zu übergreifenden Themen²

Berufs-und Studienorientierung	Data Engineer, Data Analyst, Data Scientist, Machine Learning Engineer, Softwareentwickler
Verbraucherbildung	Präsenz von neuronalen Netzen in alltäglichen Geräten, Anwendungen

¹ vgl. Rahmenlehrplan Jahrgangsstufen 1-10, Teil B, S. 15-22, Berlin, Potsdam 2015

² vgl. Rahmenlehrplan Jahrgangsstufen 1-10, Teil B, S. 24ff, Berlin, Potsdam 2015

C5 Bezüge zu anderen Fächern:

- Gesellschaftswissenschaften: Einsatz von NN in Medizin, Interaktion von NN und Menschen, Auswirkung von Überwachung und Gesichtserkennung



CC BY SA 4.0
Ausgenommen sind einzeln gekennzeichnete Inhalte/Elemente, siehe Quellen- und Lizenzhinweise am Ende des Dokuments.

iMINT Akademie Fachset Informatik
KI Programmieren im Unterricht Teil 3:
Reihenvorhersage mit LSTM



D Anhang

D1 „Material“ für den Einsatz dieser Lernaufgabe

- Für SuS eine entsprechend leistungsfähige Python IDE es empfiehlt sich Spyder
- Python (derzeit Januar 2022 ab Version 3.8, 64 Bit) mit installierter TensorFlow (ab 2.6) Bibliothek
- Da die jeweils aktuelle TensorFlowbibliothek in einer nicht aktuellen Pythonversion entwickelt wird, hilft es bei (unklaren) Compileproblemen eine eher ältere Pythonversion zu verwenden.

Hilfreiche Codsnippets:

- Codebeispiel aus Präsentation findet sich in LSTM_start.py.
- Lineare Funktion

```
1 # erzeugt eine Linearfunktion
2 def linearFunktion():
3     xListe = []
4     yListe = []
5
6     for i in range(3000):
7         xListe.append(i * 0.1)
8         yListe.append(i * 0.1)
9
10    return xListe, yListe
```

- mit zufälligen Werten überlagerte Sinuskurve

```
1 import random
2 import math
3
4 # erzeugt mit zufälligen Werten überlagerte Sinuskurve
5 def sinusRandom():
6     xListe = []
7     yListe = []
8     increaser = 0.01
9
10    for i in range(3000):
11        xListe.append(i*0.1)
```

```

12     yListe.append(math.sin(i * 0.1) + increaser)
13
14     zufallszahl = random.randint (1,100)
15     if (zufallszahl > 50):
16         increaser += 0.04
17     else:
18         increaser -= 0.04
19
20     return xListe, yListe

```

- Plot von Werten:

```

1 import matplotlib.pyplot as plt
2
3 xWerte, yWerte = sinusRandom()
4 plt.figure(figsize=(20,5))
5 plt.plot(xWerte, yWerte, color = 'red', marker = "")
6 plt.title(plotTitle)
7 plt.xlabel("x-Werte")
8 plt.ylabel("y-Werte")
9 plt.show()

```

- Daten aufbereiten für LSTM

Wie in der Keras API (https://keras.io/api/layers/recurrent_layers/lstm) formuliert benötigt ein LSTM Layer eine dreidimensionale Datenstruktur (z. B.: Tensor, Array) als Eingabe. Struktur: [batch, timesteps, feature].

In diesem Codebeispiel wird gezeigt wie eine solche Datenstruktur mithilfe der Bibliothek numpy erstellt werden kann.

```

1 import numpy as np
2
3 def getDataForLSTM():
4     threeDimArray = np.zeros((10, 5, 1))
5     for g in range(10):
6         for i in range(5):
7             threeDimArray[g][i][0] = g * i * 0.01
8
9     return threeDimArray
10
11 trainInput = getDataForLSTM()

```

D2 Hinweise / Musterlösung der Lernaufgabe

1. Führe den Code aus und beobachte das Ergebnis.

Da die Gewichte zu Beginn zufällig vergeben werden, erzeugt auch dieses `model`, wie auch die bei den vorherigen Aufgaben, bei jedem Durchlauf unterschiedliche Ergebnisse.

2. Führe Anpassungen am `model` durch, um das Ergebnis zu verbessern.

Für Aufgabe 2 empfiehlt es sich, die Benutzung der TensorFlow API https://www.tensorflow.org/api_docs/python/tf/all_symbols oder die etwas übersichtlichere API von Keras <https://keras.io/api/> einzuführen. Dies ermöglicht leistungsstarken Schülern das selbständige Arbeiten mit dem Code. Mögliche Aspekte dazu:

- Der Kernel initializer `tf.keras.initializers.he_normal`, zu finden im `model` (`kernel_initializer='he_normal'`), erzeugt bessere Ergebnisse als z. B. die Verwendung von; `tf.keras.initializers.ones`

Dokumentation dazu: <https://keras.io/api/layers/initializers>

- `adam` ist ein guter Allroundoptimizer und i. d. R. die beste Wahl für ein NN mit LSTMs. Aber auch andere Optimierer lassen sich testen z. B. `RMSProp`.

Dokumentation dazu: <https://keras.io/api/optimizers/>

- Der Befehl `tf.random.set_seed(1)` sorgt dafür, dass zu Beginn immer die gleichen Gewichte gesetzt werden. So lassen sich die Ergebnisse besser vergleichen.

3. Erstelle eigene Methoden um Daten zu generieren:

- a) linear zufällig steigend / fallend
- b) Kreisfunktionen z. B. mit `math.sin(...)`
- c) Kombination aus mehreren Funktionen.

Siehe dazu die Codebeispiele unter D1.

4. Lasse das NN den jeweils nächsten Wert vorhersagen.
Hinweis: Da die LSTM-Schicht nur 3-dimensionale Datenstrukturen akzeptiert liegen hier einige Herausforderungen in der Aufbereitung der Daten. Unterstützung bieten die Codebeispiele unter D1.

5. Ergänze die `testListe` mit Deinem vorhergesagten Wert um den übernächsten Wert vorherzusagen. Wieviele Werte kann Dein `model` sinnvoll vorhersagen?
Wenn der zuletzt vorhergesagte Wert als letztes Element in die Liste hinzugefügt wird, während das erste Element der Liste gelöscht wird. So kann das `model` immer weitere Werte vorhersagen. Kleine Abweichungen zu Beginn sorgen später für sehr viel größere Abweichungen.

D3 Quellen / Lektüreliste zum Weiterlesen

- MORONEY, Laurence [2020]: AI and Machine Learning for Coders, A Programmer's Guide to Artificial Intelligence, Sepastopol (USA)
- OLAH, Christopher [2015]: Understanding LSTM Networks, <https://colah.github.io/posts/2015-08-Understanding-LSTMs>, abgerufen am 10.4.2022
- PHI, Michael [2018]: Illustrated Guide to LSTM's and GRU's: A step by step explanation, <https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21?gi=c6c0e802a57b>, abgerufen am 10.4.2022



CC BY SA 4.0
Ausgenommen sind einzeln gekennzeichnete Inhalte/Elemente, siehe Quellen- und Lizenzhinweise am Ende des Dokuments.

iMINT Akademie Fachset Informatik
KI Programmieren im Unterricht Teil 3:
Reihenvorhersage mit LSTM

Senatsverwaltung
für Bildung, Jugend
und Familie

BERLIN



Bildtitel	Seite	Quelle
[B0]	1	„E-V-A“, Alexander Schindler, Lizenz CC BY-SA 4.0 , Lernaufgabe "KI im Unterricht"
[B1]	12	„LSTM-low-level-view“, Alexander Schindler, Lizenz CC BY-SA 4.0 , Lernaufgabe „KI im Unterricht III - LSTM“
[B2]	1, 9	„LSTM-high-level-view“, Alexander Schindler, Lizenz CC BY-SA 4.0 , Lernaufgabe „KI im Unterricht III - LSTM“
[B3]	10	„LSTM-high-level-view ausgerollt“, Alexander Schindler, Lizenz CC BY-SA 4.0 , Lernaufgabe „KI im Unterricht III - LSTM“
[B4]	14	[B4] „LSTM-low-level-view with gates and cell“, Alexander Schindler, Lizenz CC BY-SA 4.0 , Lernaufgabe „KI im Unterricht III - LSTM“



CC BY SA 4.0
Ausgenommen sind einzeln gekennzeichnete Inhalte/Elemente, siehe Quellen- und Lizenzhinweise am Ende des Dokuments.

iMINT Akademie Fachset Informatik
KI Programmieren im Unterricht Teil 3:
Reihenvorhersage mit LSTM

Senatsverwaltung
für Bildung, Jugend
und Familie

BERLIN

