12. Informatiktag Berlin-Brandenburg 2020
22. September 2020

Workshop 1

Programmieren mit KI im Informatikunterricht

Dozenten:
Alexander Dietz (Verbund 4)
Alexander Schindler (Verbund 2)

Ablauf

- **Vorstellung**
- **Theoretische Grundlagen**
- **Programmieren mit Cognimates**
- **Programmieren mit TensorFlow**
- **Diskussion**

...und Ihre Fortbildungswünsche???

bitte in den Chat schreiben

1. Bezug zum Rahmenlehrplan Sek I

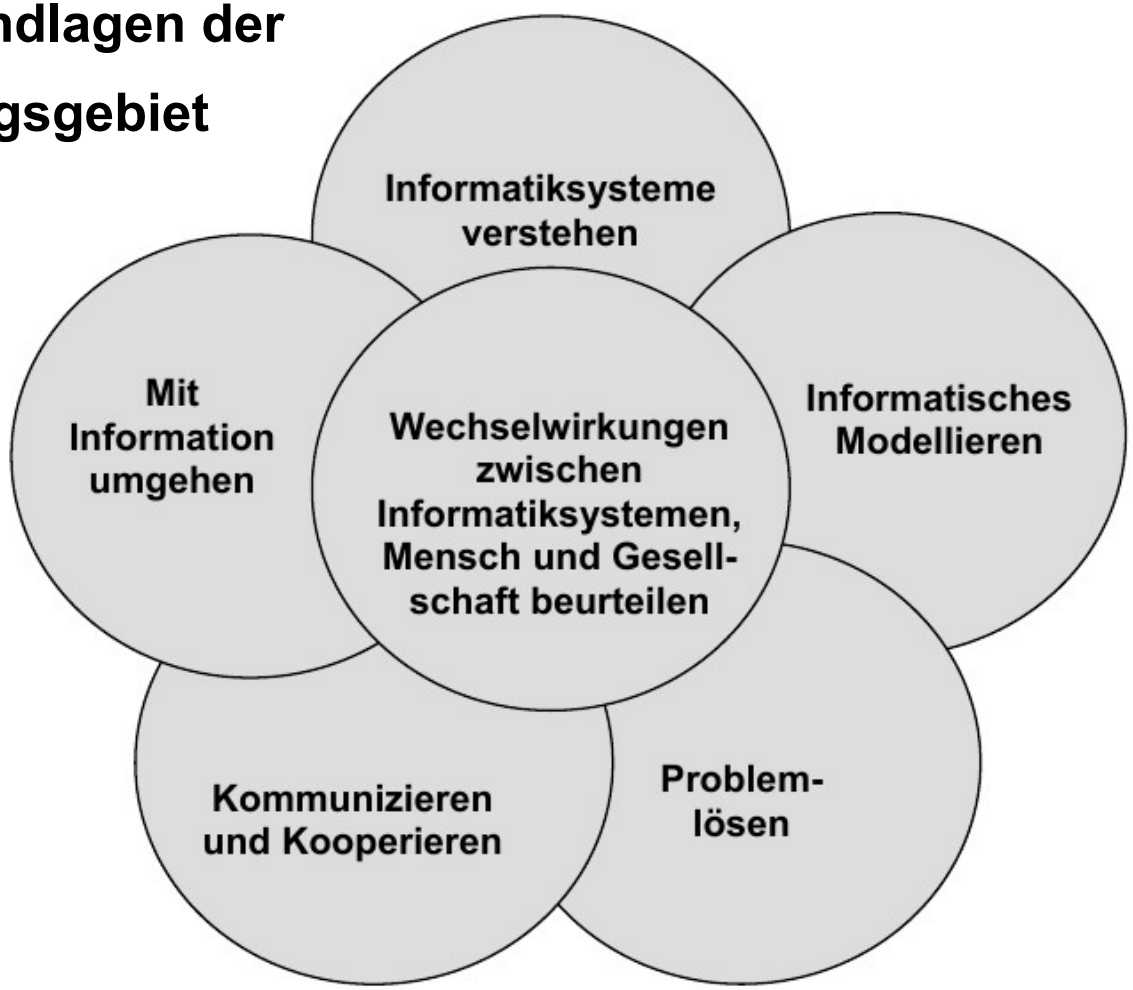
2.4 Informatisches Modellieren – Modelle erstellen und bewerten Kompetenzbereich

	Informatische Modelle analysieren und bilden	Grundlegende Konzepte der objektorientierten Modellierung anwenden	Relationale Modellbildung anwenden[1] Kompetenz
Niveaustufen	Die Schülerinnen und Schüler können		
D E			
F	informatische Modelle als reduzierte Abbildung der realen Welt beschreiben und beurteilen		Daten in einer vorgegebenen Tabelle bearbeiten Standard
G	ein Modell selbst erstellen	einer Klasse Eigenschaften zuordnen	eine Datenbank benutzen und den tabellarischen Aufbau nachvollziehen
H	beurteilen, ob das selbst erstellte Modell problemadäquat ist	den Zusammenhang zwischen Klassen und Objekten beschreiben	den Aufbau einer einfachen Datenbank planen und diese implementieren (ohne Einsatz von SQL)

1. Bezug zum Rahmenlehrplan Sek 2

3. Kurshalbjahr (IN-3): Grundlagen der Informatik und Vertiefungsgebiet

V5 Künstliche Intelligenz



2. Programmierumgebung starten

Programmieren Möglichkeit 1:

- Anaconda / Spyder auf eigenem Desktop

Programmieren Möglichkeit 2:

- https://colab.research.google.com

Sourcecode:

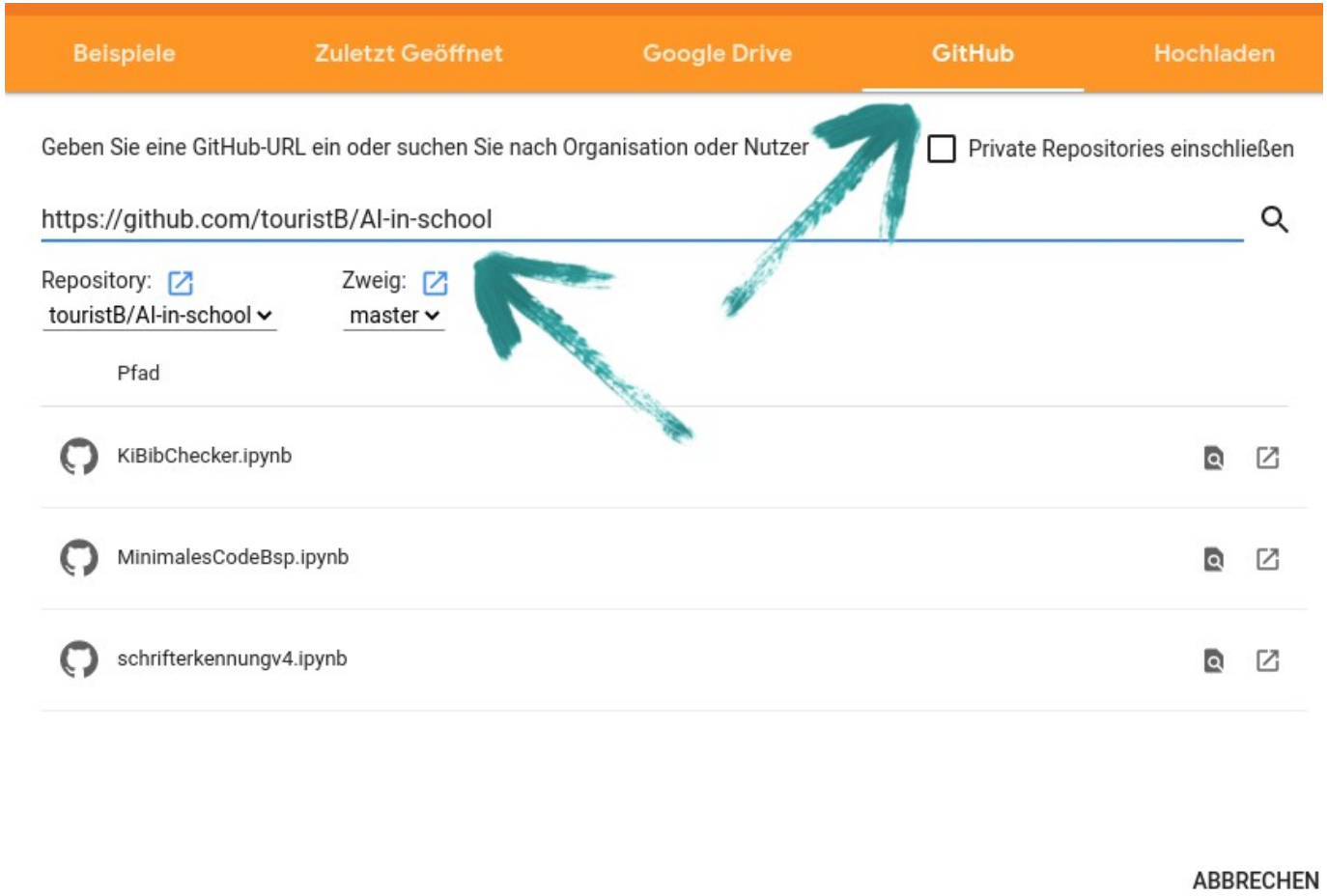
- https://github.com/touristB/AI-in-school

Dokumente:

- https://github.com/touristB/AI-in-school/tree/master/docs

2. Programmierumgebung starten

colab: Datei → Notebook öffnen → GitHub



2. Minimales Codebeispiel

```
[ ]  import tensorflow as tf
```

1. Die Trainingsdaten werden erstellt:

```
[ ]  inputMuster = [1, 2, 4]
     outputMuster= [3, 6, 12]
```

2. Aufbau des neuronalen Netzwerkes:

```
[ ]  model = tf.keras.Sequential()
     model.add(tf.keras.layers.Dense(1, input_shape=[1]))
     model.compile(optimizer='sgd', loss='mean_squared_error')
```

3. Trainieren des neuronalen Netzwerkes:

```
[ ]  model.fit(inputMuster, outputMuster, epochs=1000)
```

4. Testen des neuronalen Netzwerkes mit Testdaten hier 22:

```
[ ]  testMuster = [22]
     print(model.predict(testMuster))
```

2. Minimales Codebeispiel

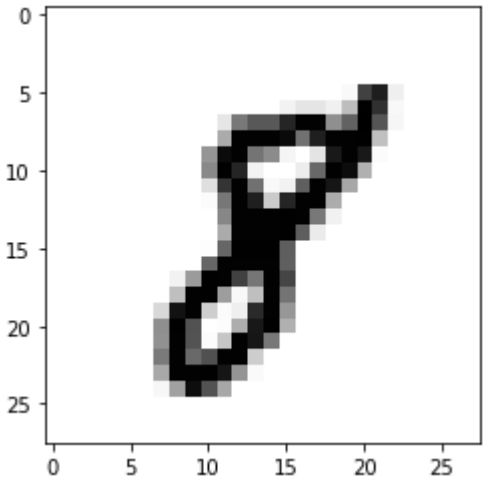
```
1 import tensorflow as tf
2
3 # Erstellen der Trainingsdaten
4 inputMuster = [1, 2, 4]
5 outputMuster= [3, 6, 12]
6
7 # Aufbau des neuronalen Netzwerkes
8 model = tf.keras.Sequential()
9 model.add(tf.keras.layers.Dense(1, input_shape=[1]))
10 model.compile(optimizer='sgd',
 loss='mean_squared_error')
11
12 # Trainieren des neuronalen Netzwerkes
13 model.fit(inputMuster, outputMuster, epochs=1000)
14
15 # Testen des neuronalen Netzwerkes mit Testdaten
16 testMuster = [22]
17 print(model.predict(testMuster))
```

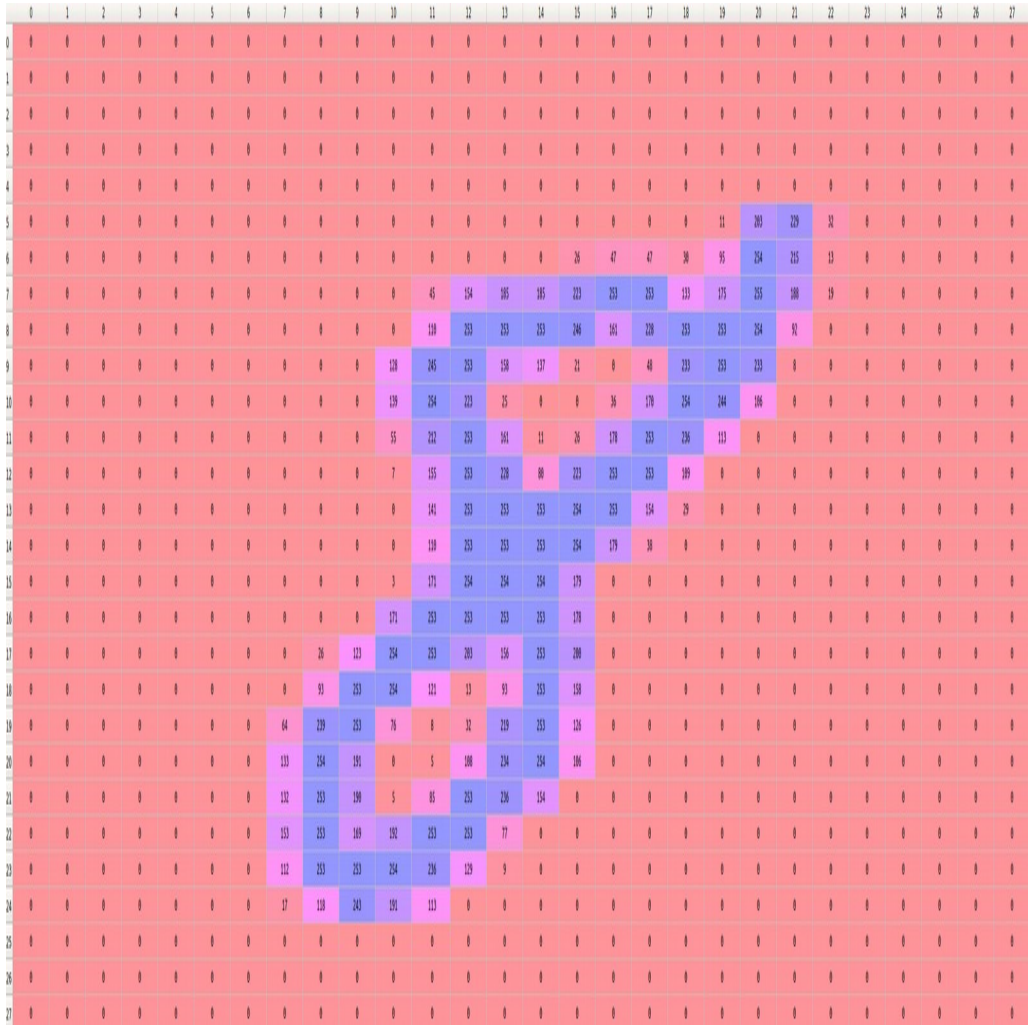
2. Minimales Codebeispiel - Schüleraufgaben

1. Ändere die Anzahl der Trainingsdaten in den Arrays inputMuster und outputMuster. Beobachte das vorhergesagte Ergebnis. Benenne sinnvolle Ober- und Untergrenzen.
2. Verringere / erhöhe die Epochenanzahl und beobachte das erwartete Ergebnis. Erkläre die Unterschiede.
3. Verändere das inputMuster / outputMuster für verschiedene Berechnungen z.B. Addition, Divisionen usw.
4. Automatisiere das Testen durch viele verschiedene testMuster.
5. Ermittle den Fehlerwert. Benutze dazu Schleifen und zufällige Zahlen.

2. MNIST - Pixelrepräsentation

28 x 28 = 784 Pixel

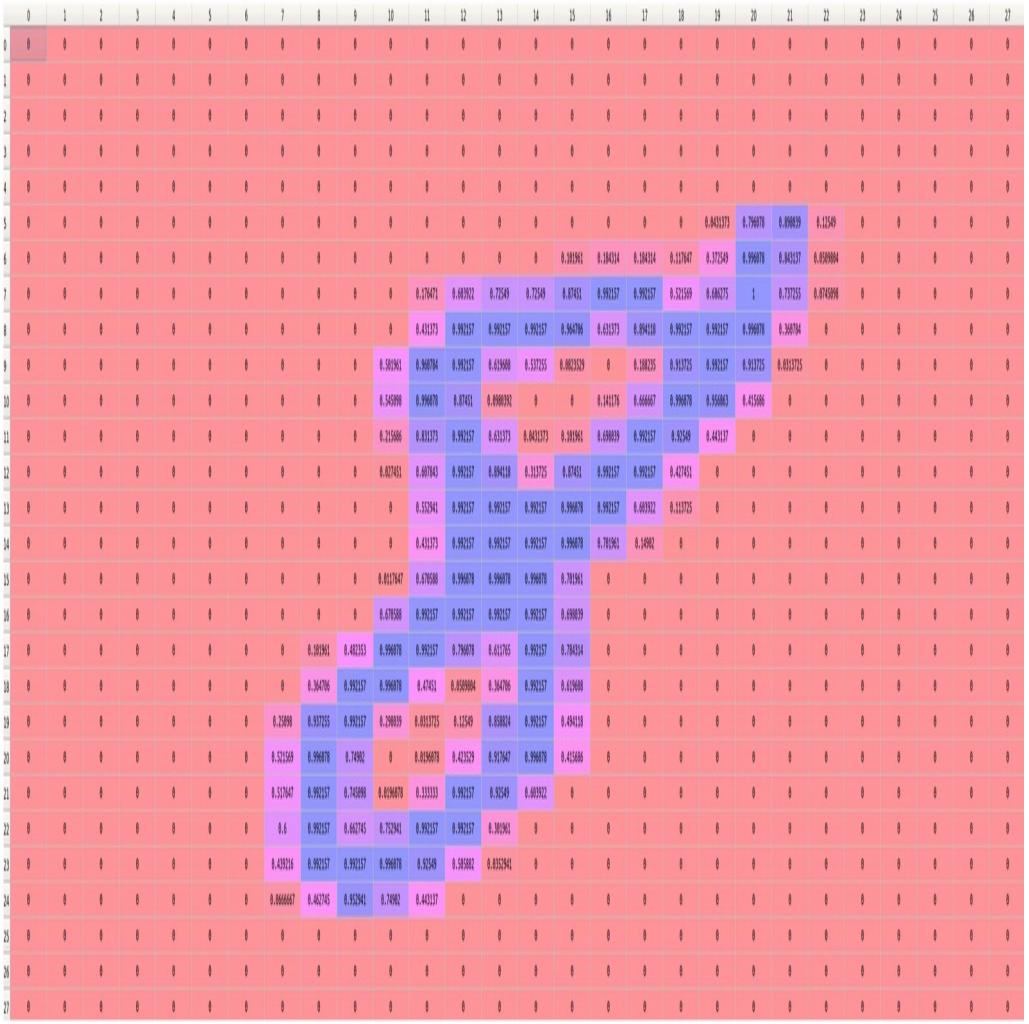




Ziffer 8 steht am Index 17
des Trainingsdatensatzes

2. MNIST – normalisierte Daten

Alle Werte liegen nach der Normalisierung im Bereich 0 bis 1.



2. Schrifterkennung

```
1  import tensorflow as tf
2  import numpy
3  import matplotlib
4
5  # 1. Daten aufbereiten
6  mnist = tf.keras.datasets.mnist
7
8 (trainZiffernBilder, trainZiffernLabels),(testZiffernBilder,
  testZiffernLabels) = mnist.load_data()
9  trainZiffernBilder = trainZiffernBilder / 255.0
10 testZiffernBilder = testZiffernBilder / 255.0
11
12 # 2. Aufbau des NN
13 model = tf.keras.Sequential()
14 model.add(tf.keras.layers.Flatten(input_shape=(28, 28)))
15 model.add(tf.keras.layers.Dense(128, activation=tf.nn.relu))
16 model.add(tf.keras.layers.Dense(10, activation=tf.nn.softmax))
17
```

```
18 model.compile(optimizer='adam',
19              loss='sparse_categorical_crossentropy',
20              metrics=['accuracy'])
21
22 # 3. NN trainieren
23 model.fit(trainZiffernBilder, trainZiffernLabels, epochs=5)
24
25 # 4. NN prüfen
26 verlust, genauigkeit = model.evaluate(testZiffernBilder,
 testZiffernLabels)
27 print('Verlust: ', verlust, 'Genauigkeit: ', genauigkeit)
28
29 # 5. Testen des NN mit Testdaten
30 gesuchteZahlIndex = 0
31 erkennungsRaten = model.predict (testZiffernBilder
 [gesuchteZahlIndex:gesuchteZahlIndex+1])
32 flattendEr = erkennungsRaten.flatten()
33 flattendTZLabels = testZiffernLabels.flatten()
34
35 # 6. Test: Ausgabe der gesuchten Zahl als Bild
36imageArray = numpy.asfarray (testZiffernBilder
 [gesuchteZahlIndex:gesuchteZahlIndex+1]).reshape((28,28))
37 matplotlib.pyplot.imshow(imageArray, cmap='Greys',
 interpolation='None')
```

```
38 matplotlib.pyplot.show()
39 print ("gesuchte Zahl: ", flattendTZLabels[gesuchteZahlIndex])
40
41 # Ausgabe der Erkennungsraten für die Zahlen 0..9
42 counter = 0
43 while counter < 10:
44     readAbleErkennung = flattendEr[counter] * 10000
45     readAbleErkennung = readAbleErkennung.astype(int)
46     readAbleErkennung = readAbleErkennung / 10000
47     print (" Zahl:", counter, "  Erkennungsrate:",
 readAbleErkennung)
48    counter = counter + 1
```

2. Schrifterkennung - Schüleraufgaben

1. Beschreibe das Ergebnis des NN.
2. Erkläre, warum das Ergebnis des NN für das Testmuster bei Aufg. 1 nicht exakt mit dem des Antwortmusters übereinstimmt.
3. Verändere die Anzahl der Epochen. Beschreibe / erkläre die Auswirkungen. Ermittle dazu sinnvolle Ober-/ Untergrenzen.
4. Verändere die Anzahl der Knoten in Zeile 16. Beschreibe die Auswirkungen auf das Ergebnis und erkläre die Veränderungen.
5. Ermittle, wo eine sinnvolle Unter- / Obergrenze für die Knotenanzahl liegt.
6. Entscheide anhand der Erkennungsrate, welche Ziffer erkannt wurde.
7. Überlege, welche Auswirkungen eine zu große Knotenanzahl bei kleinen Datensätzen haben kann.

8. Das Ergebnis soll so aufbereitet werden, dass nur noch 0 und 1 darin vorkommen.
9. Erstelle eigene Bilder in der Größe 28x28 Pixel und versiehe sie mit verschiedenen handschriftliche Zahlen.
10. Fotografiere Zahlen und forme sie auf eine Bildgröße 28x28 Pixel um.
11. Drehe Deine Bilder mit Python um einige Grad.
12. Beobachte bei den selbst erstellten und veränderten Bildern (Aufg 9.-11.) die Erkennungsrate. Wo liegen Grenzen?
13. Erkennst Du Zahlen besser als Deine KI?

3. Motivation KI / DL

- Zunehmender Einsatz
- Wissenschaft - Genetik
- Wirtschaft - Bewertung juristischer Texte, Spracherkennung (Alexa, Siri, Amazon Echo)
- Übersetzung (deepl.com)
- 2016 ca. 9.5 Mrd. US$ Investment in DL weltweit

Ziel: Schülern eigenständige Anwendung und Bewertung ermöglichen

3. Motivation KI / DL

Deep Learning verändert die Art wie zukünftige Software funktioniert

- **bisher**: If-else-Anweisungen bilden bekannte Prozesslogik ab
- **neu**: Prozesslogik wird durch neuronale Netzwerke aus großen Datenmengen gelernt

Jetzige Schüler werden als Erwachsene täglich mit dieser Technologie konfrontiert werden – privat und beruflich:

3. Anwendungsbeispiele

Bilder entfernt aufgrund unklarem Urheberrecht.

- Google KI erkennt Herzkrankheiten an Iris
- Spiegel.de: KI-Revolution im Reagenzglas
- Frankfurter Allgemeine Zeitung: Google-Software besiegt Go-Genie

3. Anwendungsbeispiele

Bilder entfernt aufgrund unklarem Urheberrecht.

- Autononmes Fahren
- Spracherkennung
- Börsenkurse
- Nachahmung von Künstlern
- Go
- Doom
- Röntgenbilder

3. Anwendungsbeispiele

[...großflächiger Einsatz von künstlicher Intelligenz (KI)“ könnte der Wirtschaftsprüfungs- und Beratungsgesellschaft PwC zufolge in den Gesundheitssystemen in Europa **Einsparungen in dreistellliger Milliardenhöhe** generieren. KI könnte in der Medizin helfen, schwere Krankheiten wesentlich früher zu erkennen und „Millionen von Menschen **besser zu therapieren**“. (aerzteblatt.de, 25. Juli 2017)

Spieleklassiker **Breakout** (1976)

Bilder entfernt aufgrund unklarem Urheberrecht.

3. Anwendungsbeispiele - DeepMind

Ob DeepMind ein Spiel gut beherrscht, hängt stark davon ab, wie weit die Konsequenzen der Handlung in die Zukunft reichen.

Bilder entfernt aufgrund unklarem Urheberrecht.

Gezeigt: Leistungen im Vergleich zu einem professionellen mensch-lichen Spieletester (100%).

Quelle: Spektrum Kompakt: Künstliche Intelligenz [2016]

3. Deepmind spielt Strategiespiel

Starcraft 2: LiquidTLO (ProGamer) vs AlphaStar (Deepmind)

AlphaStar: Actions per Minute auf pro Level

youtube.com/watch?v=PFMRDm_H9Sg

3. Grundlegende Veränderung

Bilder entfernt aufgrund unklarem Urheberrecht.

- KFZ – Wettbewerb um bessere Software nicht Motor
- Mensch-Maschine-Kommunikation
- Landwirtschaft
- Diagnostik z.b. Röntgenbilder
- Handelssysteme
- Juristische Fallvorhersage über Dokumente

3. Die drei AI-Wellen

Bilder entfernt aufgrund unklarem Urheberrecht.

- 1st Generation KI: Menschen schaffen Regeln, geringe Lernfähigkeit, für eng definierte Probleme
- 2nd Generation AI (Machine Learning): verbesserte Klassifizierung, Vorhersagen u. Lernfähigkeit, besserer Umgang mit Unsicherheit
- 3rd Generation AI (Deep Learning): NN schaffen Regeln und extrahieren Wissen aus Daten, hohe Lernfähigkeit, differenzierter Umgang mit Unsicherheit, verschiedene Problemdomänen

3. Einsatz

Zunehmende Verbreitung:

A - Algorithmen

B - Big Data

C - Computing Power

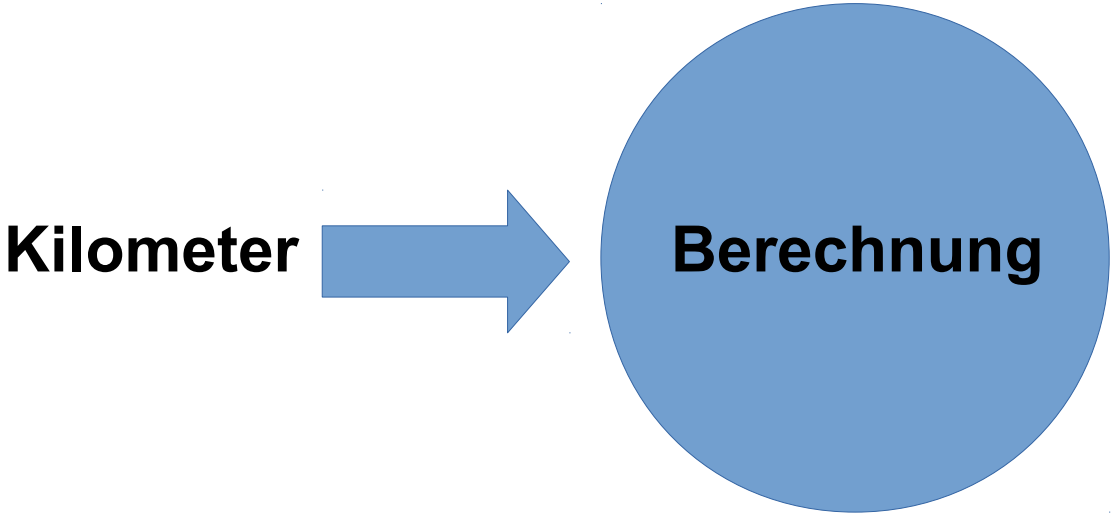
3. Das Wasser steigt...

Bild entfernt aufgrund unklarem Urheberrecht.

Q: Hans Moravec

3. EVA

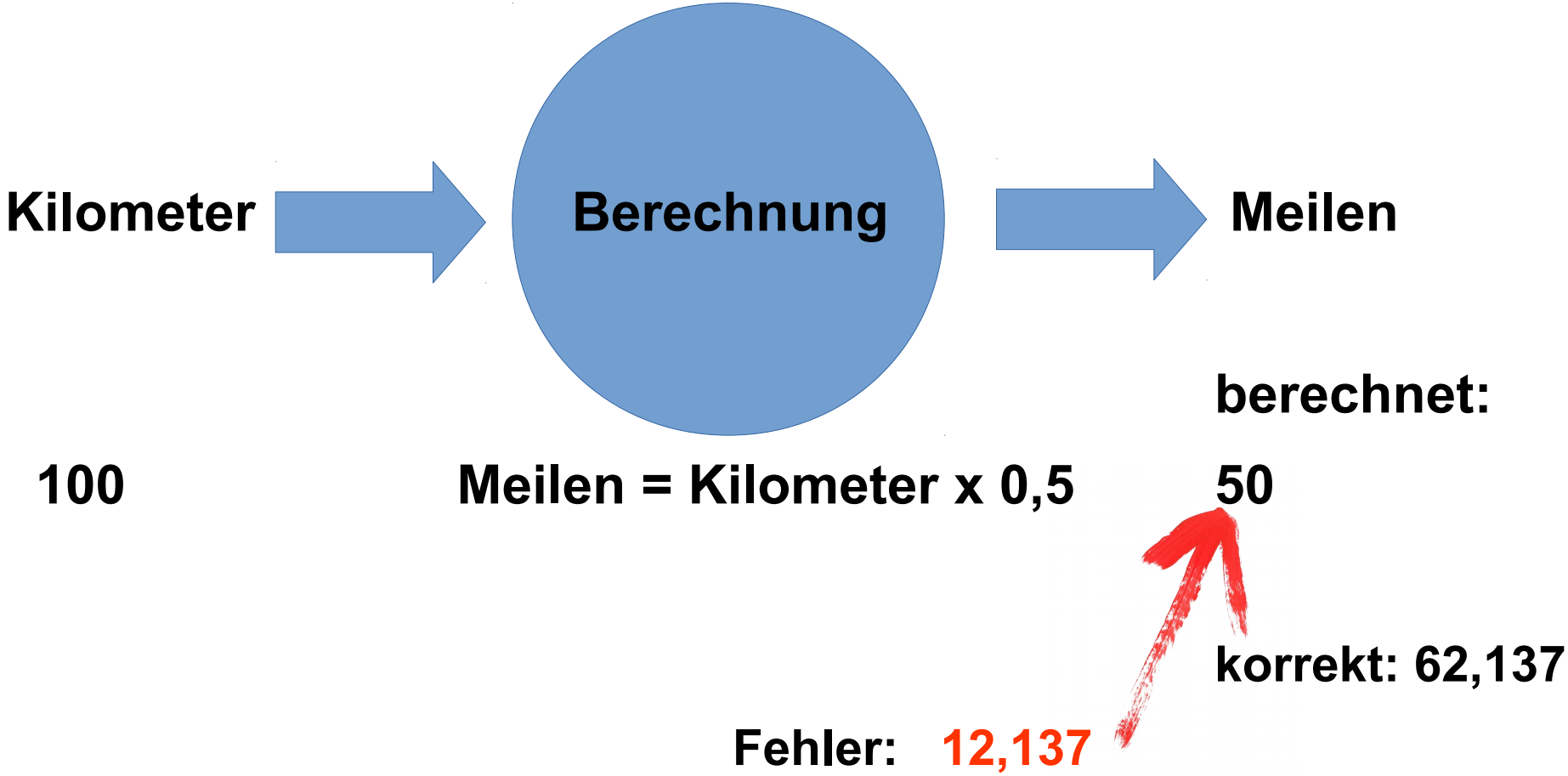
3. EVA - Schätzung



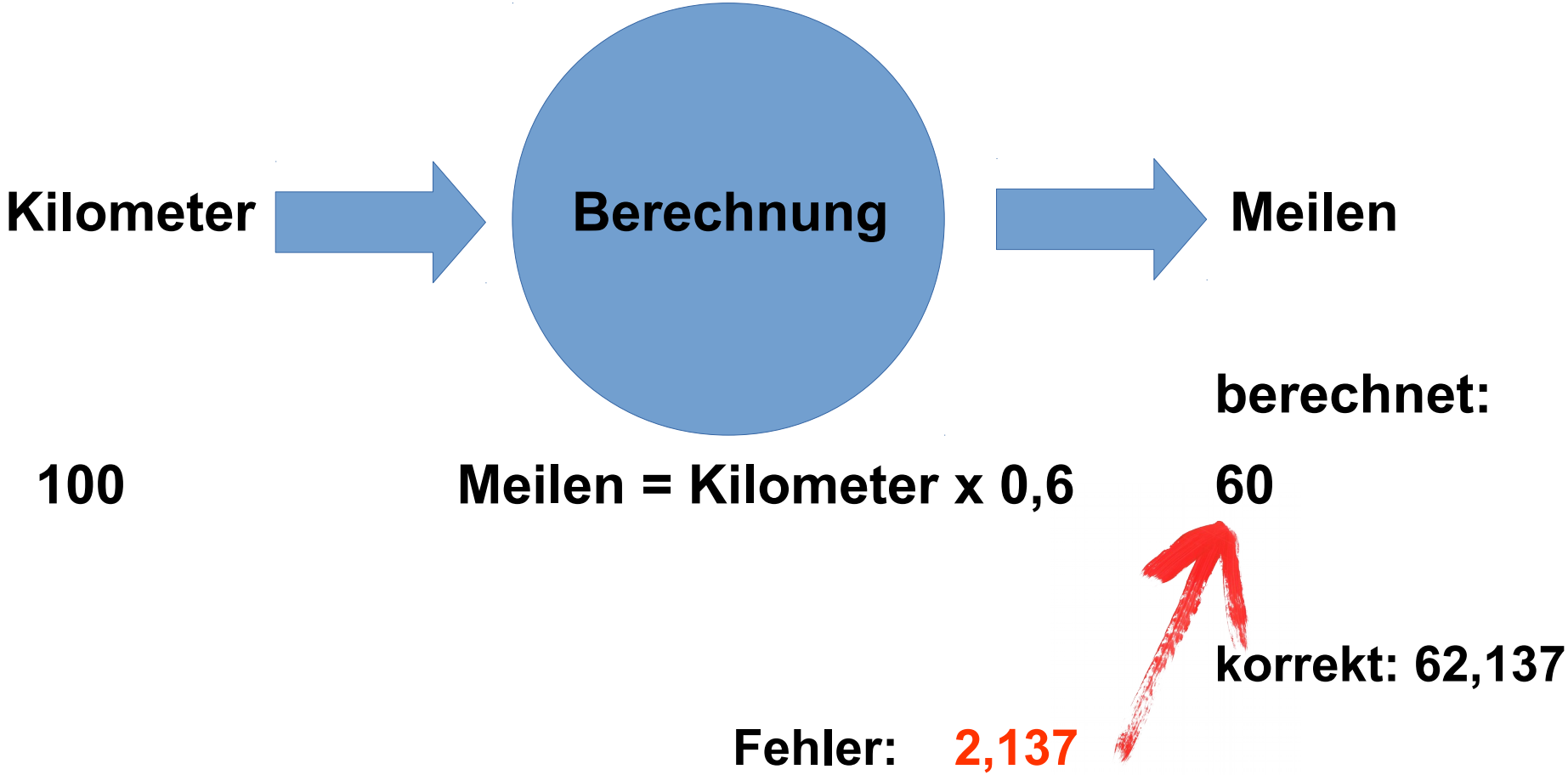
berechnet:

100 **Meilen = Kilometer x 0,5** **50**

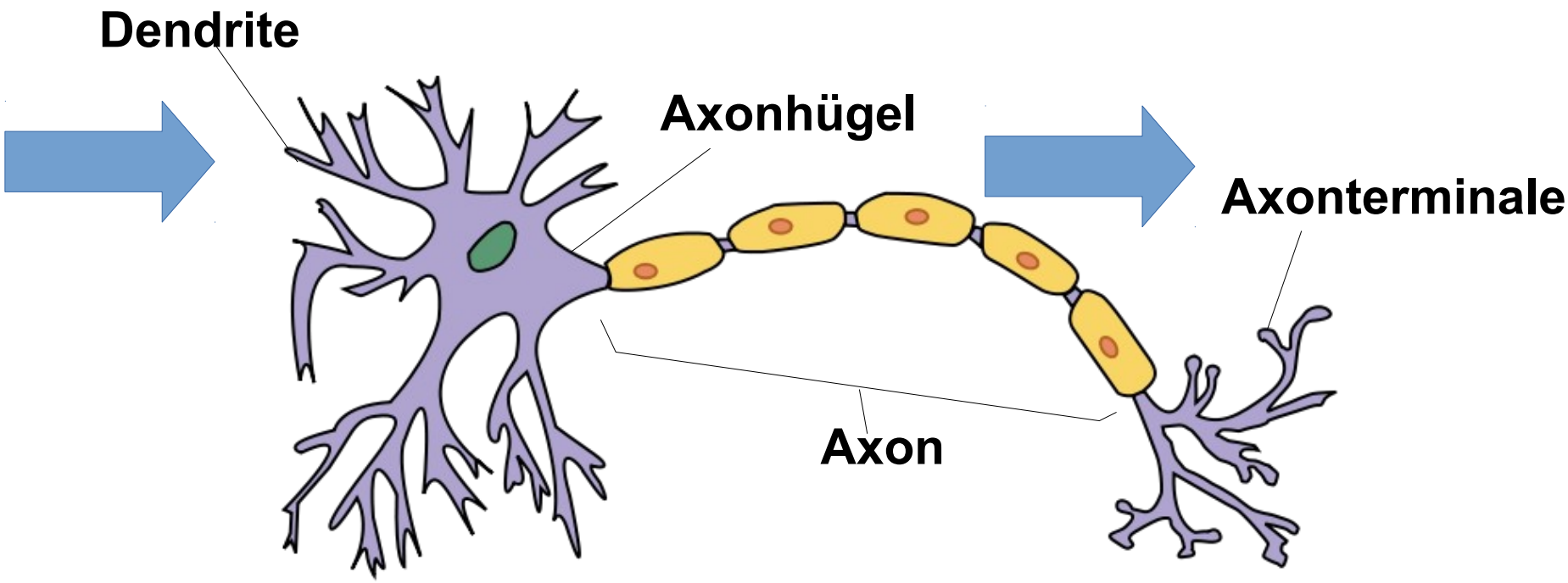
3. EVA - Schätzung



3. EVA - verbesserte Schätzung

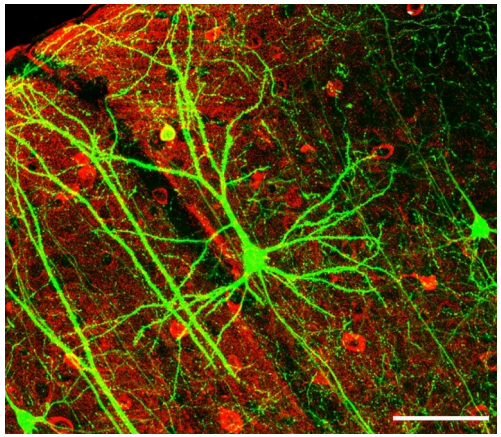


3. Neuron - Biologie



Ab einem bestimmten Schwellenpotential im Axonhügel wird ein Aktionspotential weitergeleitet. Man sagt:“Die Nervenzelle feuert.“

3. Neuronen bilden Netzwerke - Biologie



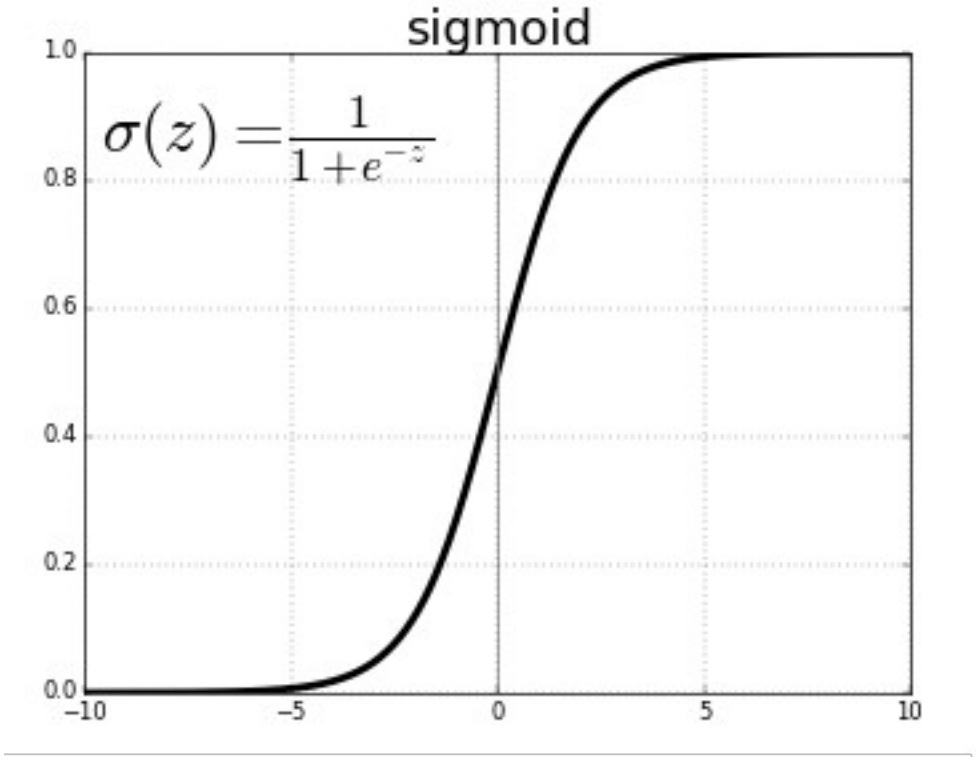
Großhirnrinde einer Maus

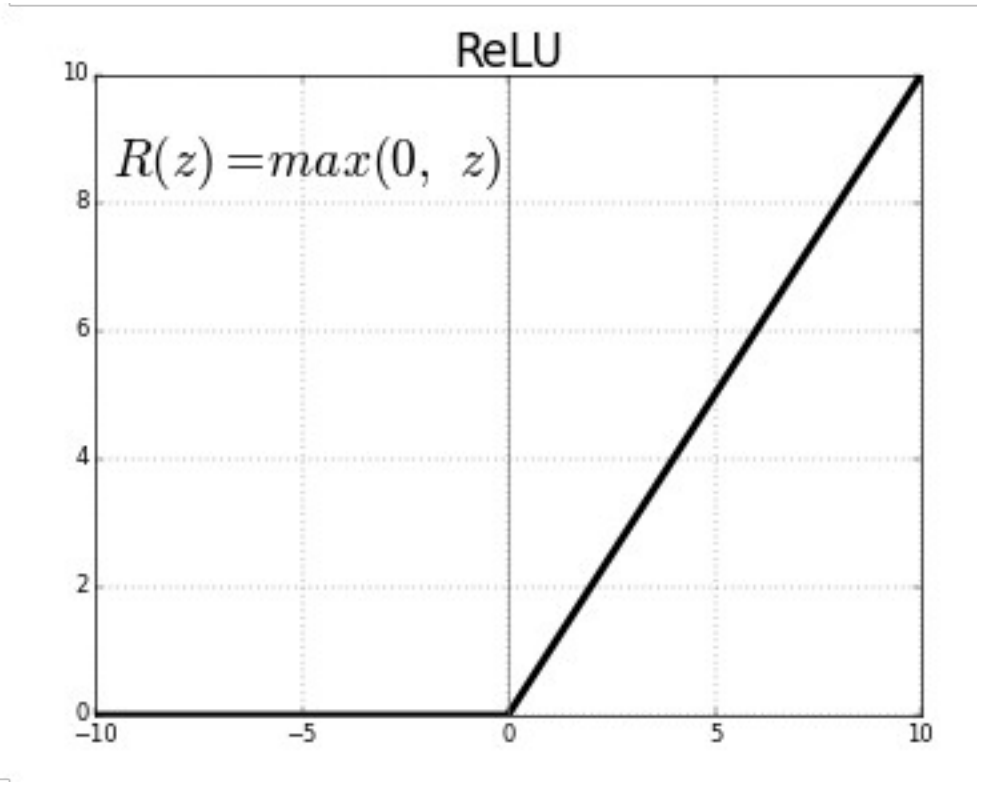
Bilder entfernt aufgrund unklarem Urheberrecht.

- **Würmer: 302 Neuronen**
- **Schnecke: 11 000 Neuronen**
- **Delphin: 37 Milliarden Neuronen**

- **Mensch: 86 Milliarden Neuronen**

3. Aktivierungsfunktion





Rectified Linear Unit

3. Modell eines künstlichen Neurons

Bild entfernt aufgrund unklarem Urheberrecht.

hier: Sigmoid Schwellenwertfunktion, andere Funktionen sind möglich.

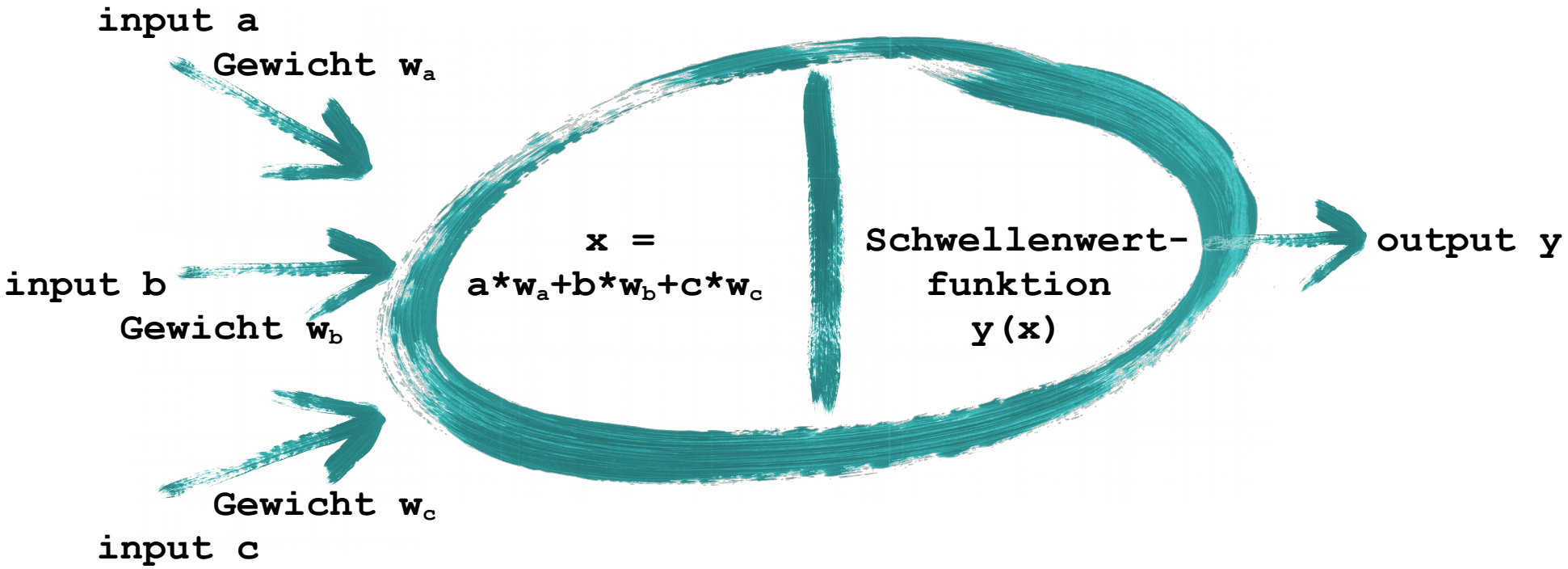
3. 3 Schichten mit verbundenen Neuronen

Bild entfernt aufgrund unklarem Urheberrecht.

3. Lernen durch Gewichtsänderung

Bild entfernt aufgrund unklarem Urheberrecht.

3. Künstliches Neuron mit Gewichten



3. Berechnung des Outputs

Bild entfernt aufgrund unklarem Urheberrecht.

3. Berechnung des Outputs

Bild entfernt aufgrund unklarem Urheberrecht.

3. Output als Matritzenmultiplikation

Bild entfernt aufgrund unklarem Urheberrecht.

$$X = W * I$$

I: Eingabematrix

W: Gewichtsmatrix

X: resultierende Matrix

3. Gewichte lernen

Bild entfernt aufgrund unklarem Urheberrecht.

3. Gewichte lernen

Bild entfernt aufgrund unklarem Urheberrecht.

NN lernen durch verfeinern der Verknüpfungsgewichte.

3. Neurales Netzwerk

1. initialisieren
2. trainieren
3. testen

3. Im Unterricht

Themengruppen

1. Teachable Machine teachablemachine.withgoogle.com
2. TensorFlow playground.tensorflow.org
3. Daddy‘s Car youtube.com/watch?v=LSHZ_b05W7o
4. TensorFlow lernt eine einfache logische Verknüpfung
5. Neuronales Netzwerk in Python

Aufgabe:

Erstellen Sie drei Aufgaben für den Unterricht zum jeweiligen Thema.

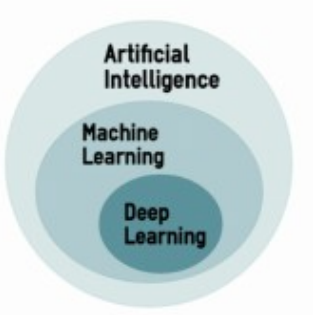
Entscheiden Sie selbst ob für Sek I oder Sek II.

3. Wichtige Begriffe

- *overfitting*: Netzwerk lernt alle Trainingsmuster auswendig.
- *overshooting*: Bei zu hoher Lernrate kann das Fehlerminimum nicht erreicht werden.
- *Gewichte*: Stärken (oder schwächen) Verbindungen zwischen Knoten.
- *Lernrate*: Höhe der Anpassung der Gewichte
- *accuracy:* Anteil der richtigen Vorhersagen
- *loss value*: Abstand zum globalen Minimum
- *Aktivierungsfunktion*: Art und Weise wie ein Neuron aufgrund bestimmter Eingaben feuert.
- *Epoche:* Wiederholtes trainieren derselben Trainingsmuster.

3. Wichtige Begriffe

- *Künstliche Intelligenz KI (artificial intelligence AI*): großes Forschungsfeld in dem Maschinen kognitive Eigenschaften zeigen (z. B.: lernen, Bilderkennung, Spracherkennung, proaktive Interaktionen, Problemlösen)
- *Machine Learning ML*: Computer sollen aus Daten lernen und Vorhersagen machen.



- *Deep Learning DL:* Einsatz von neuronalen Netzwerken mit z.T. mehreren 100 Schichten
- *Convolutional Neural Networks (CNN):* nutzen die räumliche Informationen der Daten

3. Links

Programmierbibliotheken

- TensorFlow tensorflow.org
- scikit scikit-learn.org
- PyTorch pytorch.org
- OpenAI openai.com
- Codebsp. makeyourownneuralnetwork.blogspot.com

Neuronale Netze in Aktion

- TensorFlow playground.tensorflow.org
- Teachable Machine teachablemachine.withgoogle.com
- tensorflow.org/js/demos

3. Links

Tutorials

- https://developers.google.com/machine-learning/crash-course (engl.)

3. Datensammlungen

- **Handschriften MNIST DB** yann.lecun.com/exdb/mnist
- **CIFAR-10 / CIFAR-100** www.cs.toronto.edu/~kriz/cifar.html
- **Blumenbilder**:

  http://download.tensorflow.org/example_images/flower_photos.tgz
- **Kategorisierte Bilder:** image-net.org

  https://storage.googleapis.com/openimages/web/index.html
- **Übersicht:**

  https://en.wikipedia.org/wiki/List_of_datasets_for_machine_learning_research

3. Literatur

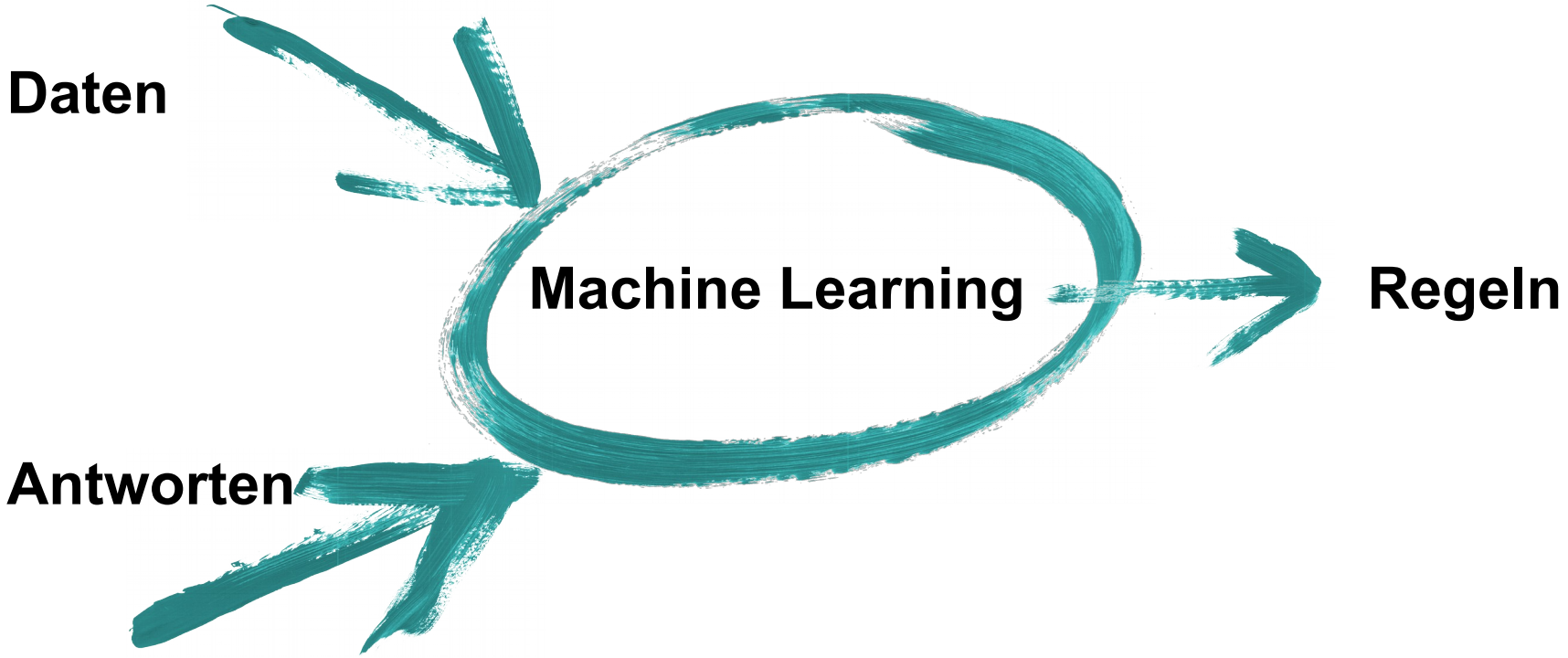
Bilder entfernt aufgrund von unklarem Urheberrecht.

RASHID, Tariq [2017]: Neuronale Netze selbst programmieren, ein verständlicher Einstieg mit Python. Heidelberg

RUSSEL, Stuart, NORVIG, Peter [2017]: Artificial Intelligence A Modern Approach,London

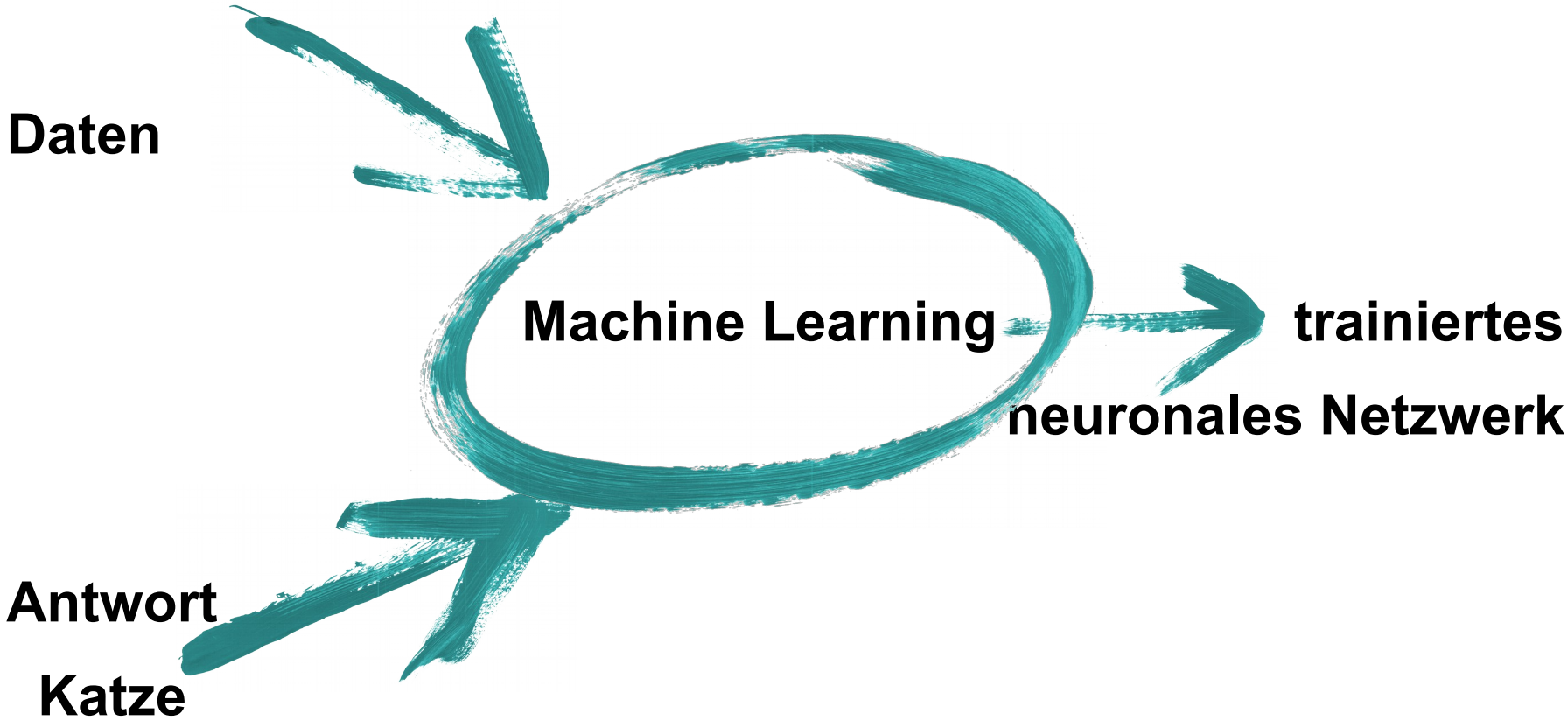
4. Klassische Programmierung

4. Machine Learning



4. Machine Learning

Bild einer Katze (*Bild entfernt aufgrund unklarem Urheberrecht.***)**



4. Machine Learning

```
[0,0], [0,1], [1,0], [1,1]
```

Daten

ML

```
model = tf.keras.models.Sequential()
```

NN

```
model.compile(...)
model.fit(inputMuster, outputMuster,...)
```

Antworten

```
[0], [0], [0], [1]
```

4. Neuronales Netzwerk erstellen

In wenigen Schritten zum trainierten neuronalen Netzwerk:

```
# 1. Daten aufbereiten
# 2. Model erstellen
# 3. Model trainieren
# 4. Model prüfen
# 5. Mustererkennung
# 6. Test
```

4. TensorFlow

- `TensorFlow` ist Bibliothek von Google
- www.tensorflow.org/api_docs
- `Keras` eigenständig aber in `TensorFlow` integriert
- `model` enthält die einzelnen `layers`
- Daten und Antworten werden vom `model` trainiert.
- Das `model` bildet das trainierbare Neuronale Netzwerk

4. Mustererkennung - AND

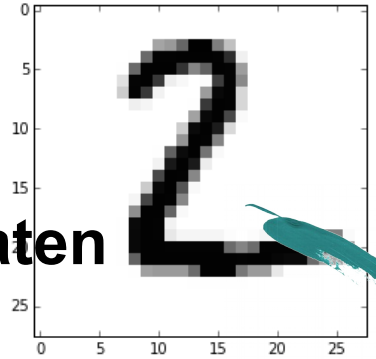
Aufgaben:

1. Verringere / erhöhe die Anzahl der Trainingsläufe (`epochenAnzahl`). Ermittle sinnvolle Ober- und Untergrenzen.
2. Verändere die Anzahl der Layer / der Neuronen / die Aktivierungsfunktionen / usw. des neuronalen Netzes und beschreibe das Ergebnis.

Hinweis: Weitere Aktivierungsfunktionen finden sich unter: `tf.nn. ...`

4. Schrifterkennung MNIST

Daten



ML

```
model = tf.keras.models.Sequential()
model.compile(...)
model.fit(inputMuster, outputMuster,...)
```

NN

Antwort:

2

4. MNIST

Aufgaben

1. Schreibe eine Schleife um den Test so dass 10 verschiedene Zahlen getestet werden.
2. Entscheide anhand der Erkennungsraten welches das richtige Ergebnis ist.
3. Bereite das Ergebnis auf. z.b. mit matplotlib.
4. Verändere das `model` des Neuronalen Netzes und beschreibe das Ergebnis. Benutze dazu die TensorFlow API. (www.tensorflow.org/api_docs)

4. Eigenes neuronales Netzwerk

- Code: `NN_NeuralNetwork.py`
- Quelle: RASHID, Tariq [2017]: Neuronale Netze selbst programmieren

4. Eigene handschriftliche Zahlen

- Code: `NN_mnist_and_own_data_v2.py`
- Format: .png
- Größe: 28 x 28 Pixel

Aufgaben

1. Erstelle mehrere eigene Zahlen entsprechend obiger Anforderung. Variiere in Form, Farbe, Größe, Ausrichtung-
2. Teste ob das trainierte neuronale Netzwerk deine Zahlen erkennt.

4. Bilderkennung - Blumen

- TensorFlow in 10 Minuten
- Eingabe über Konsole
- stärker Anwendungsorientiert

4. Bilderkennung - Blumen

soon to come….

1. Daten laden
2. Aufbau NN
3. Testen

4. Bilderkennung - Blumen

Aufgaben:

1. Erstelle ein NN Model das Gänseblümchen erkennt.
2. Trainiere auf verschieden Blumenarten.
3. Verändere die Anzahl der Trainingsbilder.
4. Verändere das Model.

4. Diskussion

Ist KI / Deep Learning im Unterricht sinnvoll einsetzbar?

Ende...

Kontakt:

a.dietz@humboldtschule-berlin.eu

alexander_schindler@gmx.de