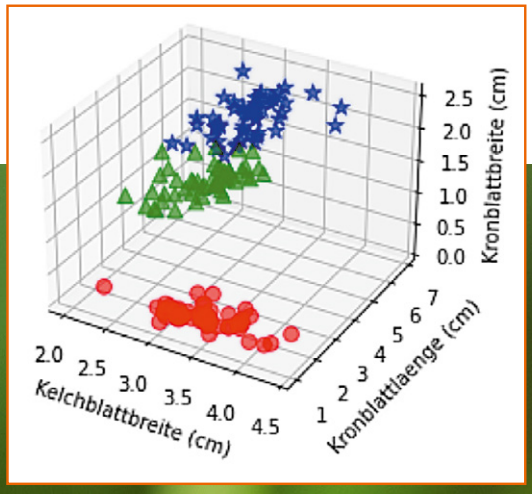
KI-Praxis I

Einstieg in die Künstliche Intelligenz

Teil 1

Das Thema „Künstliche Intelligenz“ (auch KI oder AI für engl. Artificial Intelligence) wird seit längerer Zeit intensiv in allen Medien diskutiert. Die Vision, dass in naher Zukunft Roboter, Computer oder ganz allgemein Maschinen „intelligent“ sein sollen, bewegt viele Menschen mehr als jede andere technische Entwicklung. Dabei steht die Frage im Brennpunkt, ob Maschinen tatsächlich jemals wirklich selbstständig lernen oder sogar intelligent sein können.



Was ist Intelligenz?

Die vielbeachteten Siege von Computern im Schach oder sogar im Go-Spiel gegen die besten menschlichen Spieler haben weltweit für Aufsehen gesorgt. Insbesondere beim Go-Spiel hat die KI neue, bislang unerreichte Ebenen erobert. Allerdings wird immer noch häufig angezweifelt, dass für diese Erfolge tatsächlich „echte" Intelligenz erforderlich war. Das liegt unter anderem daran, dass keine eindeutige Definition für Intelligenz existiert. Wenn aber noch nicht einmal klar ist, was Intelligenz überhaupt genau sein soll, wie kann dann entschieden werden, ob maschinelle oder künstliche Intelligenz existiert?

In dieser Artikelreihe soll ein neues Licht auf das Thema KI geworfen werden. Dabei stehen nicht theoretische Betrachtungen im Vordergrund, vielmehr werden mit einfachen Mitteln praktische Einsichten in dieses hochaktuelle Gebiet der Informationswissenschaften vermittelt.

Neben einem PC oder Laptop sollen dabei insbesondere auch kleine Systeme wie der Raspberry Pi zum Einsatz kommen. Damit lassen sich dann sogar Geräte aufbauen, die im Alltag durchaus nutzbringend eingesetzt werden können.

Aus Daten Lernen

Unter „Machine Learning" (maschinelles Lernen) versteht man die Anwendung von Algorithmen, die in der Lage sind „den Sinn von Daten" bis zu einem gewissen Maße zu erkennen oder zu interpretieren. Ein Ziel der KI-Forschung ist es, Daten unter Anwendung von maschinellem Lernen in Wissen umzuwandeln und daraus nützliche Schlussfolgerungen zu ziehen.

Dank Open-Source-Bibliotheken können sich auch Nicht-Fachleute eingehend mit dem Thema Machine Learning befassen. Damit wird es praktisch für jedermann möglich, leistungsfähige Algorithmen einzusetzen. Dafür sind keine Supercomputer erforderlich. Vielmehr können auch kleine Systeme Muster in großen Datenmengen erkennen oder Vorhersagen über zukünftige Ereignisse ableiten.

Zunächst soll hier die relevante Terminologie vorgestellt werden. Damit werden die Grundlagen für das Lösen von Praxisproblemen geschaffen. Die ersten beiden Artikel zum Thema KI werden daher in die folgenden Themen einführen:

- Die drei Arten des Machine Learnings
- Grundlegender Aufbau von lernfähigen Systemen
- Installation der Programmiersprache „Python"
- Einrichtung einer Machine Learning-geeigneten Software-Umgebung

Künstliche Intelligenz, Maschinenlernen, Neuronale Netze und Deep Learning

Das Forschungsgebiet der künstlichen Intelligenz enthält drei Teilbereiche:

- Machine Learning
- Neuronale Netze
- Deep Learning

Diese sind eng miteinander verknüpft und können in der Praxis oft nur schwer voneinander unterschieden werden. Daneben existieren noch weitere Gebiete, wie etwa Expertensysteme, cvolutionäre Algorithmen oder statistische Klassifizierer, die jedoch in letzter Zeit an Bedeutung verloren haben. Seit etwa 2010 wird die KI immer mehr von Neuronalen Netzen und Deep Learning dominiert (Bild 1).

Maschinelles Lernen ist dennoch lediglich ein Teilbereich der künstlichen Intelligenz. Dieses hat jedoch eine besondere Bedeutung erlangt, da es Systeme in die Lage versetzt, selbstständig aus Daten zu lernen und sich zu verbessern, ohne dass dafür eine explizite Programmierung erforderlich wäre. Dabei kommen die beiden folgenden Prinzipen zum Einsatz:

1. Die Eingabe von Daten mit bekannten Zusammenhängen
2. Das „Lernen" von Strukturen (sog. Trainieren), um sie später auf unbekannte Zusammenhänge anzuwenden

Maschinelles Lernen wird bereits in vielfältigen Anwendungen eingesetzt, von der Suche nach Malware in der IT-Sicherheit über die Wettervorhersage bis hin zur Analyse von Kundenverhalten. Bekannte Beispiele sind Produktempfehlungen bei Amazon, Prognose von Kundenverhalten, selbstfahrende Autos oder die Erkennung von Kreditkartenbetrug.

Das maschinelle Lernen verwendet also Algorithmen, um bekannte Daten zu analysieren. Aus diesen Daten lernt das ML-System und trifft anschließend fundierte Entscheidungen. Die Grundlage dafür ist ein System von mathematischen Methoden der Mustererkennung. Diese Verfahren erkennen Muster durch die bestmögliche Zerlegung von Datensätzen in hierarchische Strukturen (z. B. Entscheidungsbäume, Klassifizierung und Regression). Maschinelles Lernen erfordert im Allgemeinen komplexe Mathematik und Programmierung, um die gewünschten Funktionen und Ergebnisse zu erreichen.

Deep Learning wiederum ist ein Spezialgebiet des maschinellen Lernens. Es imitiert das menschliche Lernverhalten mittels großer Datenmengen. Über einen „tiefen Lernalgorithmus", der auf ein Neuronales Netzwerk angewendet wird, werden aktuell die besten Ergebnisse der KI erzielt. Die Neuronalen Netze bestehen dabei meist aus zahlreichen Zwischenschichten, welche die Verbindung zwischen den Eingangs- und Ausgangsknoten darstellen.

Die einzelnen Schichten wiederum setzen sich aus künstlich erzeugten Neuronen zusammen. Die gesamte Struktur ähnelt dem Neuronalen Netzwerk im menschlichen Gehirn.

Tiefes Lernen verwendet mehrere Schichten in den Netzen. Die erste Schicht, die sichtbare Eingangsschicht, verarbeitet eine Rohdateneingabe, beispielsweise die einzelnen Pixel eines Bildes. Über mehrere verborgene Schichten (engl. „hidden layer") und Ebe-

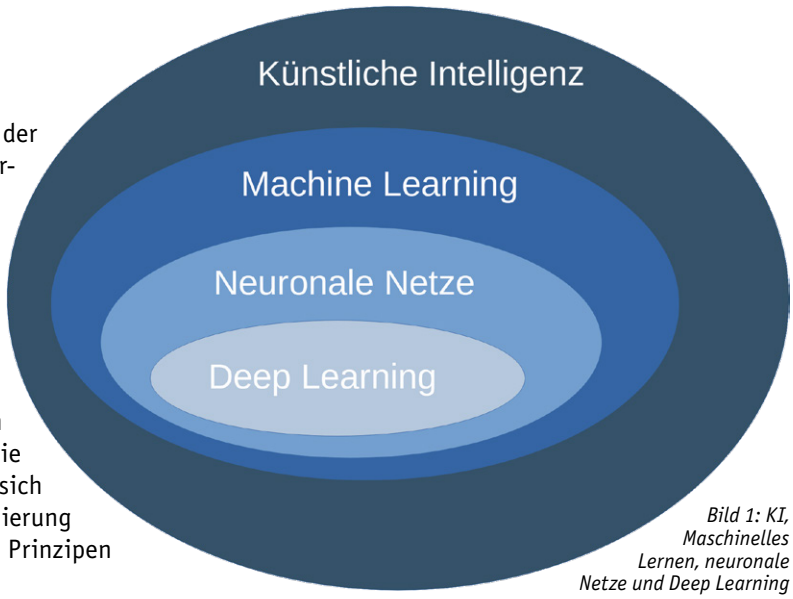


Bild 1: KI, Maschinelles Lernen, neuronale Netze und Deep Learning

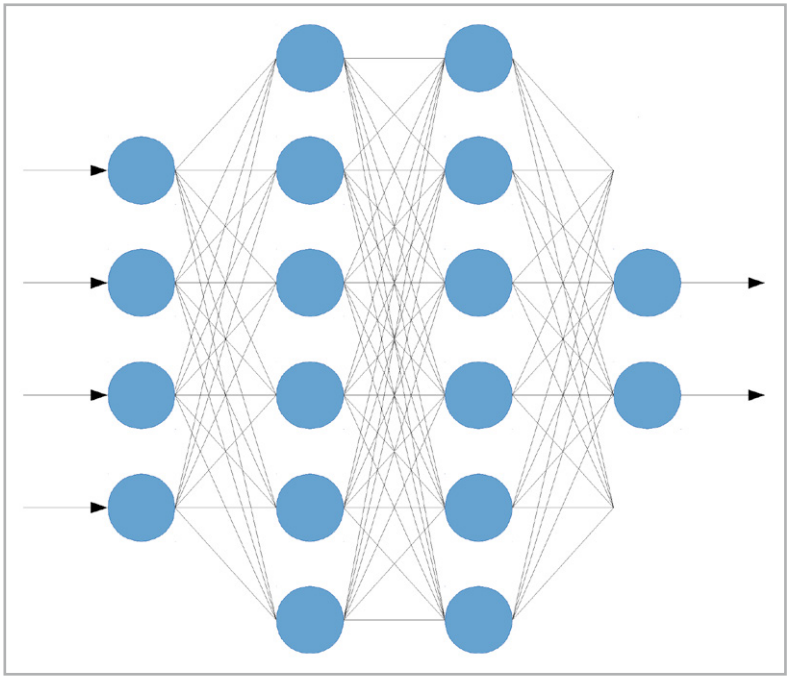


Bild 2: Neuronales Netzwerk

nen werden die Informationen weiterverarbeitet und reduziert. Die Ausgabeschicht liefert schließlich das Ergebnis. Bild 2 zeigt den typischen Aufbau eines solchen Netzes.

Der Vorteil von Deep Learning ist unter anderem die hohe Abstraktion von Korrelationen zwischen Eingangsdaten und Ausgangsdaten durch die verschiedenen Ebenen der Netzwerke. Tiefe Lernalgorithmen sind daher in der Lage, auch vergleichsweise komplexe Aufgaben zu lösen. Durch mehrfache „Trainingsläufe" kann das vertiefte Lernen mit jedem Berechnungsschritt besser werden. Damit hat es sich zu einem der zentralen Entwicklungstreiber im Bereich der künstlichen Intelligenz entwickelt.

Tabelle 1 fasst nochmals die Hauptunterschiede zwischen Machine Learning und Deep Learning zusammen:

Der Hauptunterschied zwischen Machine Learning und Deep Learning liegt also in der Fähigkeit, durch künstliche Neuronale Netzwerke (KNN) unstrukturierte Daten zu verarbeiten. Deep Learning ist durch KNNs in der Lage, unstrukturierte Informationen wie Texte, Bilder, Töne und Videos in numerische Werte umzuwandeln. Diese extrahierten Informationen werden dann zur Mustererkennung oder zum weiteren Lernen verwendet.

Klassisches Machine Learning wie zum Beispiel Entscheidungsbaumverfahren sind dazu nicht in der Lage. Sollen hier zum Beispiel Bilder als Eingabedaten genutzt werden, muss immer eine aufwendige und spezielle Programmanpassung durch Menschen erfolgen. Ein anderes Beispiel für KI-Verfahren, welche nicht auf Neuronalen Netzen basieren, sind die sogenannten evolutionären Algorithmen. Hierbei handelt es sich um Programme, die von der Natur lernen sollen, wie intelligente, lernfähige Systeme funktionieren. Vorbild ist jedoch nicht das Gehirn, sondern die Evolution.

Wie bei der biologischen Evolution folgen hier die Algorithmen in verschiedenen Generationen aufeinander. Gemäß dem Grundsatz „Survival of the fittest" wird jeder Lösungskandidat bewertet und ausgewählt, um in „mutierter" oder rekombinierter Form Verbesserungen zu finden.

Ein gravierender Nachteil bei der Anwendung von KI-Algorithmen und insbesondere bei Neuronalen Netzen ist, dass im Nachhinein meist nicht mehr nachvollzogen werden kann, wie Entscheidungen auf Grundlage welcher Daten getroffen wurden. Dies kann insbesondere bei gravierenden Fehlentscheidungen der KI zu erheblichen sozialen oder juristischen Problemen führen.

Python als Schlüssel zur KI

Die Programmiersprache Python hat sich in der KI zum Goldstandard entwickelt. Dies liegt vor allem an der schnellen Anpassbarkeit. Andere Sprachen benötigen meist spezielle und klar strukturierte Daten. Python dagegen bietet hier eine außerordentlich hohe Flexibilität.

Dazu wurden für Python eine Vielzahl von hervorragenden Bibliotheken entwickelt. Im Bereich Data-Science, Deep Learning und Machine Learning stehen inzwischen umfangreiche Libraries zur Verfügung, die es gestatten, mit wenigen Programmzeilen Anwendungen zum Machine Learning oder zu Neuronalen Netzen zu erstellen.

Für diese Artikelserie werden grundlegende Python-Kenntnisse vorausgesetzt. Diese sind jedoch über Online- oder Volkshochschulkurse etc. einfach erlernbar. Die Einbindung der notwendigen Bibliotheken in den Programmcode wird bei den jeweiligen Anwendungen ausführlich erläutert.

Bei Praxisanwendungen spricht ein weiterer wichtiger Punkt für Python. Auf dem Raspberry Pi steht die Sprache standardmäßig zur Verfügung und ist bestens in das Betriebssystem integriert. Damit muss für nutzbringende Anwendungen etwa im Bereich IoT (Internet of Things) oder der Hausautomatisierung nicht immer ein kompletter Laptop oder PC eingesetzt werden. Vielmehr genügt ein preisgünstiger Raspberry Pi (Bild 3), um diese Systeme dauerhaft zu betreiben.

Neben dem Anschaffungspreis spielt hier auch der Energieverbrauch eine erhebliche Rolle. Während ein Laptop oftmals 50 W und mehr aufnimmt, kommt der Raspberry Pi im Durchschnitt mit wenigen Watt (z. B. 5 V × 1 A = 5 W) aus. Das hat auf ein ganzes Jahr gesehen einen deutlichen Einfluss auf die Stromkosten.

Tabelle 1

Hauptunterschiede zwischen Machine Learning und Deep Learning

	Machine Learning	Deep Learning
Datenstruktur	Strukturierte Daten	Unstrukturierte und strukturierte Daten
Datensatzgröße	Klein bis mittel	Groß
Hardware	Funktioniert mit einfacher Hardware	Leistungsstärkere Rechner erforderlich
Laufzeit	Minuten bis Stunden	Bis zu Tagen und Wochen
Interpretierbarkeit	Algorithmen sind leicht zu interpretieren: Regression, Entscheidungsbäume	Schwer bis unmöglich

Hardware-Voraussetzungen

Oftmals wird künstliche Intelligenz mit Supercomputern gleichgesetzt, die in der Lage sind, Milliarden von Rechenoperationen in Sekundenbruchteilen durchzuführen. Für Anwendungen wie das Go-Spiel mag dies durchaus zutreffen. Allerdings lassen sich auch bereits auf einem PC mittlerer Leistung schon praxistaugliche Neuronale Netze implementieren. Diese sind dann sogar in der Lage, komplexere Praxisaufgaben zu übernehmen.

Ein Großteil der Rechenarbeit ist für das Training, also das eigentliche „Lernen" des Netzes erforderlich. Diese muss jedoch meist nur einmal durchgeführt werden. Anschließend steht das System für Praxisanwendungen zur Verfügung. In der Anwendungsphase kommt man daher mit deutlich geringerer Rechenleistung aus. Hier können sogar Kleinsysteme wie der Raspberry Pi sehr erfolgreich eingesetzt werden.

Eine interessante Strategie ist daher die Aufteilung von Training und Anwendung auf zwei verschiedene Systeme. Ein erster, leistungsfähiger Rechner übernimmt das Training. Die fertigen Trainingsdaten werden dann jedoch auf ein kleineres System übertragen, welches die eigentliche Anwendung übernimmt. Zudem werden häufig auch bereits vortrainierte Netze von verschiedenen Anbietern kostenlos zur Verfügung gestellt.

Um die Praxisbeispiele dieser Artikelserie durchführen zu können, sind daher die folgenden Hardware-Komponenten empfehlenswert:

1. PC oder Laptop mit den folgenden Leistungsmerkmalen:
   - Quadcore CPU 4 x 3,6 GHz
   - Arbeitsspeicher: 16 GB RAM DDR3
2. Raspberry Pi 4 mit 8 GB RAM mit Netzteil
   + SD-Karte mit min. 16 GB Speichervolumen
   + HDMI-Monitor mit Kabel
   + USB-Maus und -Tastatur

Hinzu kommen bei den einzelnen Projekten noch weitere Komponenten wie Sensoren, Aktivboxen oder eine Webcam. Details dazu werden bei den einzelnen Anwendungen erläutert.

Prinzipiell können auch ältere oder weniger leistungsfähigere Raspberry-Pi-Versionen verwendet werden. Allerdings muss man dann häufig mit erheblich längeren Trainingszeiten rechnen. Auch die Performance bei der Anwendung der Neuronalen Netze stößt bei älteren Pi-Modellen rasch an Grenzen.

Für die Arbeit mit dem Raspberry Pi muss auf der SD-Karte ein aktuelles Betriebssystem (Raspberry Pi OS) installiert sein. Die Installation kann über die Homepage der Raspberry-Pi-Foundation [1] erfolgen. Alternativ sind auch SD-Karten mit vorinstallierten Systemen oder komplette „Starterkits" erhältlich (s. Abschnitt „Material" am Ende des Beitrags). Die Programme und Anwendungen in diese Artikelserie wurden mit der Pi OS 10 („Buster") entwickelt und getestet. Leider gibt es keine Garantie dafür, dass sie auch mit früheren oder auch späteren Versionen problemlos arbeiten. Falls also unerwartete Fehler auftreten, sollte man prüfen, ob ein Umstieg auf die oben genannte Version infrage kommt.

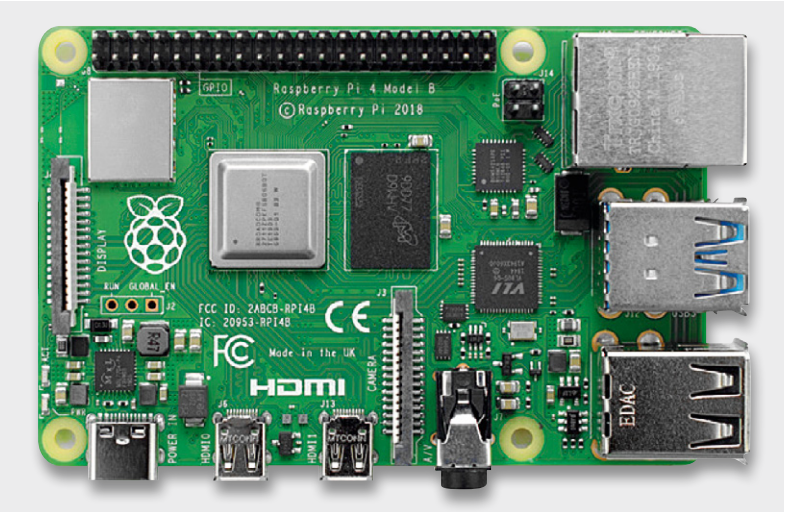


Bild 3: Raspberry Pi 4

KI auf kleinen Systemen

Obwohl der Raspberry Pi mit seiner relativ geringen Rechenleistung auf den ersten Blick keine offensichtliche Wahl für maschinelles Lernen ist, eignet sich das Board aufgrund seiner kompakten Größe und seines geringen Stromverbrauchs gut für den Aufbau von Heimautomatisierungsgeräten, kleinen Robotern oder IoT-Anwendungen. Durch maschinelles Lernen können diese Geräte dann neue Aufgaben bewältigen, indem sie Bilderkennung zum „Sehen" und Spracherkennung zum „Hören" oder Sprachsynthese zum „Sprechen" verwenden.

Aufgrund der begrenzten Rechenleistung des Pi eignet er sich nur zum Trainieren einfacher Modelle. Für das in diesem Artikel vorgestellte Klassifizierungsproblem ist die Leistung des auf dem Pi vorhandenen Broadcom-Prozessors jedoch durchaus ausreichend.

Damit ist man in der Lage, die grundlegende Struktur und die Vorgehensweise bei der Anwendung von Neuronalen Netzen nachzuvollziehen. Wenn die Trainingsphase abgeschlossen ist, kann der Raspberry Pi das trainierte Modell tatsächlich problemlos ausführen, auch wenn man einige Abstriche bei der Verarbeitungsgeschwindigkeit machen muss.

Bei verschiedenen Tests erreichte der Raspberry Pi 4 bei der Bilderkennung eine Verarbeitungsgeschwindigkeit von 1 bis 4 Bilder pro Sekunde. Das ist sicherlich langsamer als „Echtzeit", d. h. 30 bis 50 Bilder pro Sekunde, reicht für viele Anwendungen jedoch vollkommen aus. Aufgrund dieser Einschränkungen werden Computer-Vision-Aufgaben auf dem Pi häufig mithilfe der OpenCV-Softwarebibliothek ausgeführt, die auch Nicht-ML-Techniken verwendet. Damit werden auf dem Pi dann bessere Leistungen erzielt.

Bei der Sprachverarbeitung kann man beispielsweise auch auf „Expertensysteme" zurückgreifen, die ebenfalls nicht mehr zum engeren Kreis der KI gezählt werden. Dennoch lassen sich auf dem Raspberry Pi damit sehr interessante Anwendungen umsetzen.

Spätere Beiträge zu dieser Artikelserie werden sich eingehend mit diesen Themen befassen.

Jupyter-Notebook auf dem Raspberry Pi

Für klassische Programmieranwendungen steht auf dem Raspberry Pi die Thonny-IDE zur Verfügung. Diese ist bei neueren Betriebssystemvarianten bereits vorinstalliert. Thonny kann auch für KI-Programme eingesetzt werden. In späteren Anwendungen wird davon auch Gebrauch gemacht werden.

Für Machine Learning hat sich jedoch eine modernere Variante etabliert. Hier wird meist IPython, also eine interaktive Python-Programmierumgebung verwendet. Anstelle der Schritt-für-Schritt-Einrichtung von IPython – einschließlich der verschiedenen Erweiterungsbibliotheken für Machine Learning und Bilddarstellung – kann auch auf die fertig

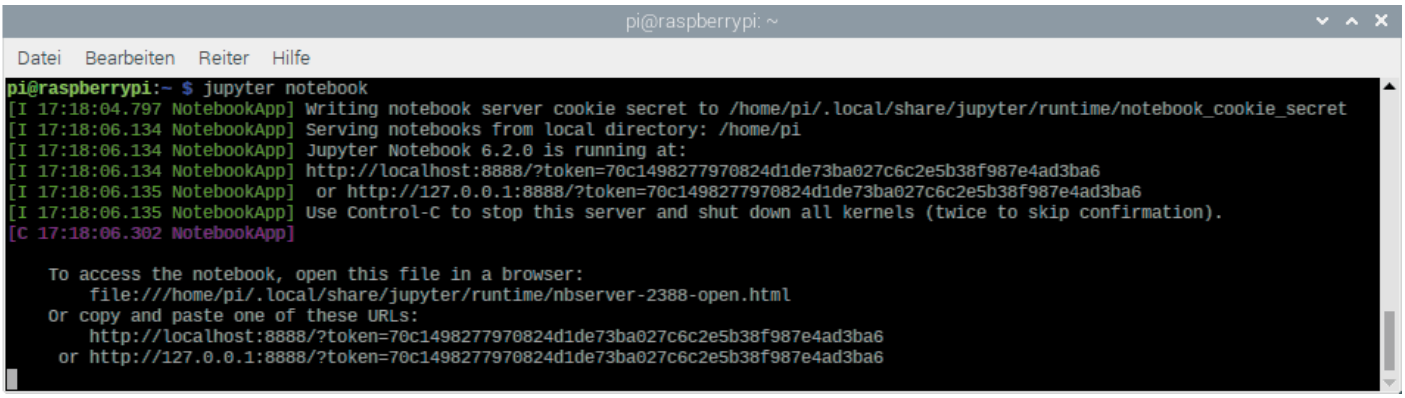


Bild 4: Erfolgreicher Start des Jupyter-Notebooks in der Konsole

gepackte Lösung „Jupyter“ zurückgegriffen werden. Damit erhält man die komplette Programmiersprache Python und die für KI-Anwendungen erforderlichen Erweiterungen für die numerische Verarbeitung und grafische Darstellung von umfangreichen Datenmengen.

Zudem verfügt diese Variante über ein interaktives Notebook. Damit lässt sich ähnlich arbeiten wie mit Papier und Bleistift. Diese Methode ist bestens geeignet, um Ideen zu testen, Ergebnisse anzuzeigen und abzuändern oder zu optimieren. Damit wird die Entwicklung von KI-Anwendungen vergleichsweise problemlos und unkompliziert. Man muss sich weder um Programmdateien noch um Compiler-Meldungen oder die Einbindung von Bibliotheken kümmern und kann sich so voll auf die Projektentwicklung konzentrieren.

Jupyter-Notebook wird direkt in einem Browser ausgeführt und kann mit minimalem Aufwand auf dem Raspberry Pi installiert werden. Die Installation erfolgt über ein „pip“-Kommando (package installer for python). Dabei handelt es sich um ein Paketverwaltungsprogramm, welches die einfache Installation unterschiedlichster Programme und Anwendungen ermöglicht. Zunächst sollte pip hierfür aktualisiert werden:

sudo pip3 install --upgrade pip

Anschließend kann dann Jupyter installiert werden:

sudo pip3 install jupyter

Die Installation nimmt je nach Internetverbindung bis zu 15 Minuten in Anspruch, da die Pakete zum Teil recht umfangreich sind. Vor der Installation sollte auf der verwendeten SD-Karte ein Speichervolumen von mindestens 4 GB frei verfügbar sein.

Pakete einzeln installieren

Python-Pakete können einzeln oder am Stück über den Befehl „pip“ installiert werden. Für spätere Anwendungen sollten an dieser Stelle die folgenden Installationen durchgeführt werden.

sudo pip3 install numpy
sudo pip3 install scipy
sudo pip3 install matplotlib ipython
sudo pip3 install scikit-learn
sudo pip3 install pandas

Falls bei einem Projekt einmal eine bestimmte Library fehlen sollte, kann diese jederzeit über die entsprechende „pip“-Anweisung nachinstalliert werden. Je nach aktueller Jupyter-Version ist u. U. die manuelle Nachinstallation einiger Libraries erforderlich.

Hinweis: Im Downloadpaket [2] zu diesem Artikel finden sich einige Tipps zur Beseitigung von eventuell auftretenden Fehlern.

Jupyter in Aktion

Gestartet wird das Jupyter-Notebook über den Befehl

jupyter notebook

Damit wird gleichzeitig der Standardbrowser aktiviert mit der Adresse

http://localhost:8888

Jupyter stellt über *jupyter notebook --help* ein umfangreiches Hilfesystem zur Verfügung.

Hinweis: Je nach Jupyter-Version muss man sich beim ersten Start des Notebooks mit einem Sicherheitstoken anmelden.

Diesen Token findet man im Terminal, in welchem das Notebook gestartet wurde (Bild 4), zum Beispiel

http://localhost:8888/?token=12345abcde...

Diesen Token gibt man im Browser (copy & paste) unter

Password or token

ein und meldet sich an.

Anschließend wird das Verzeichnis des Jupyter-Notebooks angezeigt.

Nun wird im frisch installierten Notebook

New → Python3

ein neues Arbeitsblatt geöffnet (Bild 5).

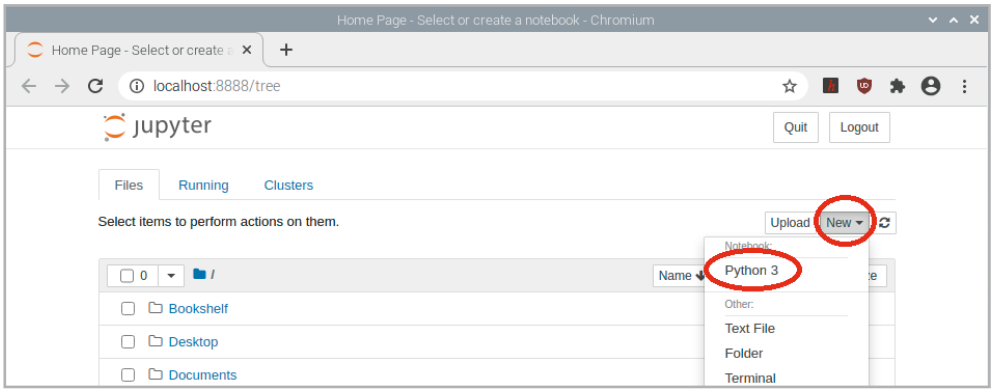
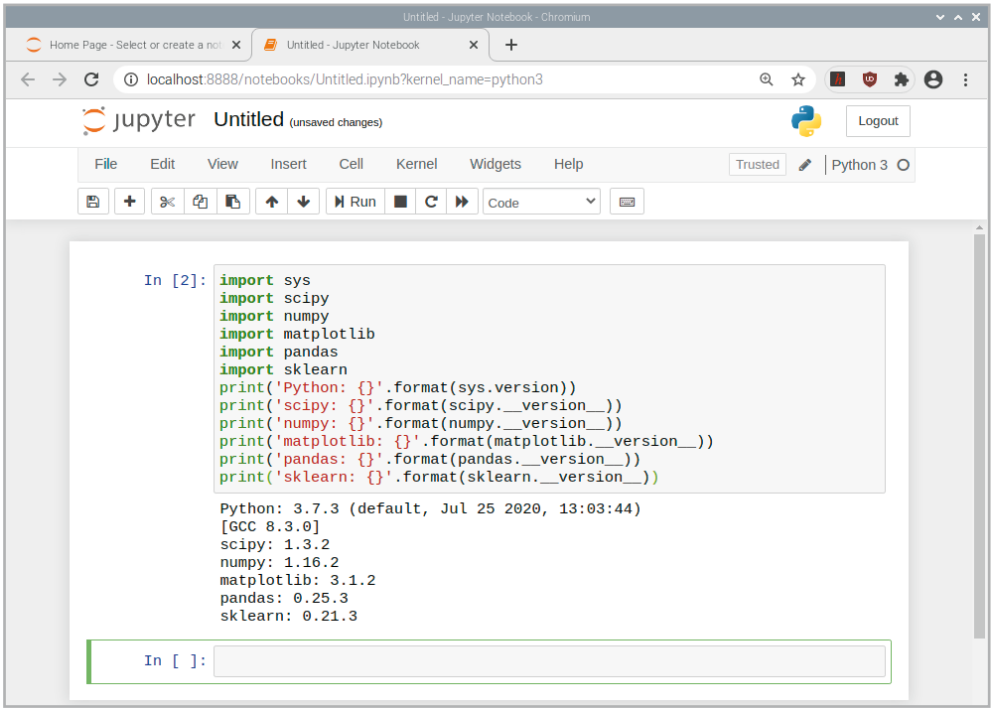


Bild 5: Öffnen eines neuen Notebooks in Jupyter

Bild 6: Überprüfung der Jupyter-Installation



Interaktives Arbeiten mit Python und Jupyter-Notebook

Nach der Installation sollte man testen, ob alle Module korrekt installiert wurden. Dazu werden in die erste Jupyter-Zelle die folgenden Anweisungen eingegeben:

```
import sys
import scipy
import numpy
import matplotlib
import pandas
import sklearn
print('Python: {}'.format(sys.version))
print('scipy: {}'.format(scipy.__version__))
print('numpy: {}'.format(numpy.__version__))
print('matplotlib: {}'.format(matplotlib.__version__))
print('pandas: {}'.format(pandas.__version__))
print('sklearn: {}'.format(sklearn.__version__))
```

Die Anweisungen finden sich als Datei (installation.txt) im Download-Paket [2]. Nach dem Laden des Pakets auf den Raspberry Pi können sie von dort direkt in die Jupyter-Zelle kopiert werden. Das mühsame Abtippen der Befehle kann damit entfallen. Nach einem Klick auf „Run“ sollte die Ausgabe etwa so aussehen wie in Bild 6.

Je nach Installationsdatum können die einzelnen Versionsnummern von Bild 6 abweichen. Häufig ist es wichtig, die Versionsnummern der Module zu kennen. Wenn Fehler auftreten, sollte man zunächst überprüfen, ob die erforderlichen Bibliotheken in der korrekten Version vorliegen. Dann kann man die Fehler anhand der Fehlermeldungen beheben. Ansonsten wird es spätestens bei den ersten komplexeren Anwendungen zu Problemen kommen.

Gelegentlich kommt es vor, dass sich einige Fehler als sehr hartnäckig erweisen. In diesem Fall kann es sinnvoll sein, das Raspberry-Pi-Betriebssystem (eventuell auf einer neuen SD-Karte) komplett neu aufzusetzen und die Installation zu wiederholen.

Auch helfen u. U. die Hinweise im Ausgabefenster der Jupyter-Zelle, in dem auf Quellen zur Fehlerbehebung hingewiesen wird.

Falls die Installation erfolgreich abgeschlossen wurde, steht nun eine umfangreiche Umgebung für die Entwicklung und den Test von KI- und ML-Anwendungen zur Verfügung.

Nicht nur für Botaniker interessant: Blütenklassifizierung

Was in der C-Programmierung das „Hello World“-Programm ist, ist im Machine Learning der „Iris“-Datensatz. Dieser besteht aus Messwerten zu jeweils drei Iris-Arten (Iris setosa, Iris virginica und Iris versicolor). Von jeder Probe wurden vier Merkmale erfasst jeweils in Zentimetern (Bild 7):

- die Länge und die Breite der Kelchblätter
- die Länge und die Breite der Blütenblätter

Dieser Datensatz eignet sich hervorragend für das Training eines Classifiers. Hierfür werden zunächst alle relevanten Module aus den Bibliotheken geladen. Diese und alle nachfolgenden Code-Zeilen werden wieder in eine Zelle eingegeben und mit „Run“ ausgeführt:

```
import pandas
import matplotlib.pyplot as plt
import sklearn
from sklearn import model_selection
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
```

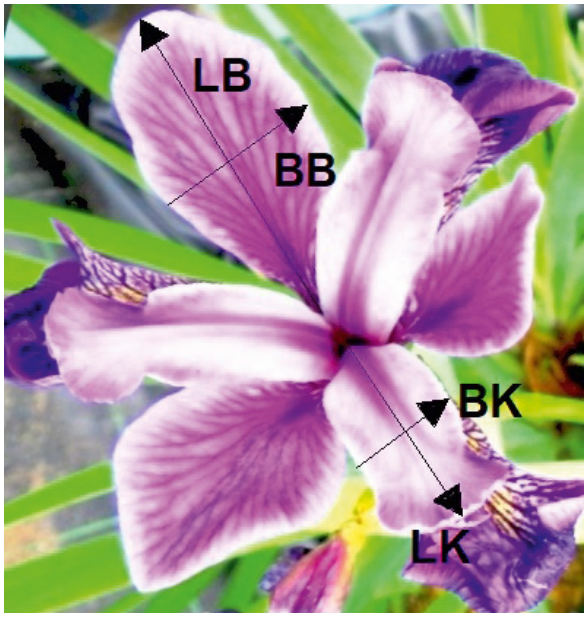


Bild 7: Iris

```
In [10]: print(dataset.head(100))

    sepal-length  sepal-width  petal-length  petal-width           class
0            5.1          3.5           1.4          0.2     Iris-setosa
1            4.9          3.0           1.4          0.2     Iris-setosa
2            4.7          3.2           1.3          0.2     Iris-setosa
3            4.6          3.1           1.5          0.2     Iris-setosa
4            5.0          3.6           1.4          0.2     Iris-setosa
..           ...          ...           ...          ...             ...
95           5.7          3.0           4.2          1.2  Iris-versicolor
96           5.7          2.9           4.2          1.3  Iris-versicolor
97           6.2          2.9           4.3          1.3  Iris-versicolor
98           5.1          2.5           3.0          1.1  Iris-versicolor
99           5.7          2.8           4.1          1.3  Iris-versicolor

[100 rows x 5 columns]
```

Bild 8: Ansicht des Iris-Datensatzes in Jupyter

Jetzt kann der oben angesprochene Datensatz geladen werden. Hierbei handelt es sich um eine CSV-Datei, die jeweils 50 Datensätze zu jeder Blume enthält. Die Daten sind dabei durch Kommata getrennt (CSV = comma separated values). Jede Zeile enthält vier Zahlenwerte, die jeweils eines der obigen Attribute darstellen. Als Letztes ist das sogenannte „Label", d. h. die Bezeichnung der betreffenden Pflanze angegeben.

Diese Datei wird direkt aus dem Internet auf den Raspberry Pi heruntergeladen und in einem Array abgespeichert:

```
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
dataset = pandas.read_csv(url, names=names)
```

Damit steht ein Array der Größe 150 × 5, also 150 Zeilen mit jeweils 5 Spalten zur Verfügung. Diese Werte können mittels

print(dataset.shape)

überprüft werden. Das Array kann komplett oder auch teilweise via

print(dataset.head(100))

für die ersten 100 Werte ausgegeben werden (Bild 8).

Alles im Blick: grafische Darstellungen

Mittels MatPlotLib kann die Verteilung der Daten grafisch dargestellt werden. Damit lässt sich ein um wesentlich anschaulicheres Verständnis für die Daten erzielen. Das ist insbesondere bei Datensätzen mit größerer Komplexität von Bedeutung. Genau wie bei großen Tabellen oder Listen ist es auch bei großen Arrays wenig aufschlussreich, das reine Zahlenmaterial zu betrachten. Um auf einen Blick erfassen zu können, was die Zahlen bedeuten, sollte man die Daten visualisieren.

Für grafische Darstellungen müssen zunächst die rein numerischen Daten extrahiert werden:

NumericData = dataset.values[:,0:4]

Dann können mit den Methoden der MatPlotLib verschiedene grafische Darstellungen erzeugt werden. Zum Einsatz kommt hier die Funktion

scatter(x, y, c=color, ...)

welche die Streudarstellung von Punkten in einem x/y-Diagramm erlaubt. Beispielsweise ergibt sich mit:

```
fig = plt.figure(1)

ax = fig.add_subplot(1,1,1)
ax.scatter(NumericData[0:50,0],NumericData[0:50,1],c='red')
ax.scatter(NumericData[50:100,0],NumericData[50:100,1],c='green')
ax.scatter(NumericData[100:150,0],NumericData[100:150,1],c='blue')
ax.set_xlabel('Kelchblattlaenge (cm)')
ax.set_ylabel('Kelchblattbreite (cm)')
ax.grid(True)
```

das in Bild 9 abgebildete Streudiagramm (Scatterplot).

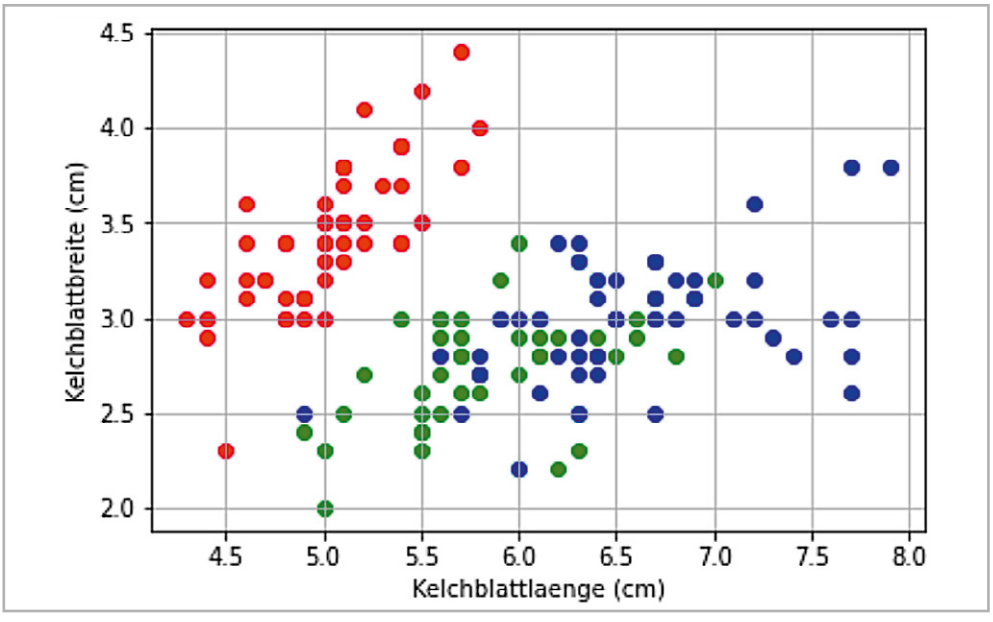


Bild 9: Streudiagramm der Lilien-Daten

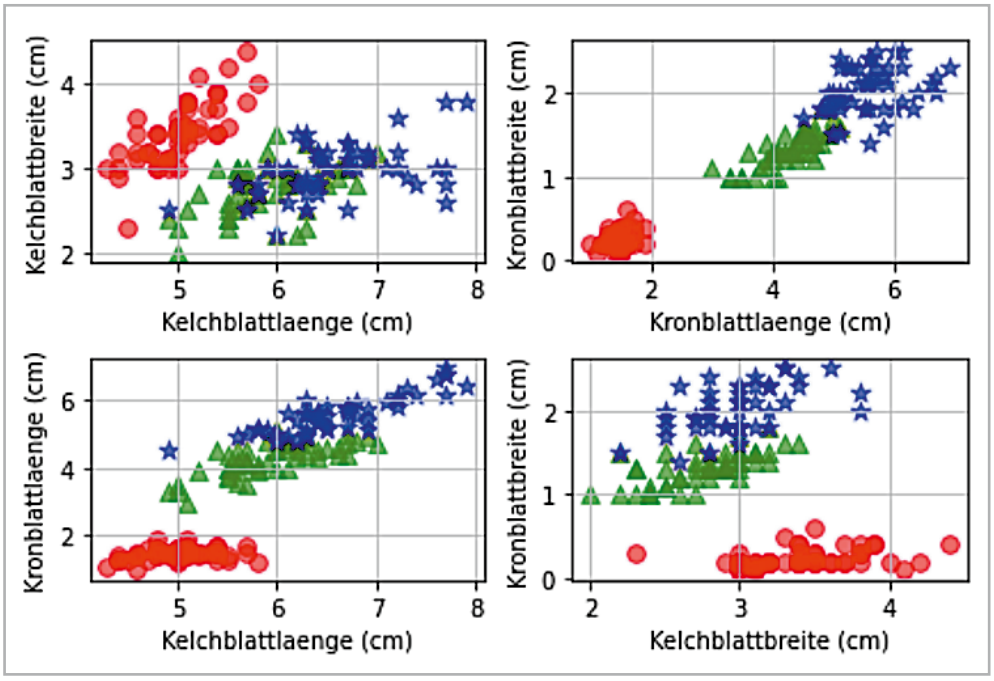


Bild 10: Scatterplots für verschiedene Datenbasen

Dabei wurde der folgende Farbcode verwendet:

Iris setosa: rot
Iris virginica: grün
Iris versicolor: blau

Man erkennt, dass die „Setosa“ in dieser Darstellung recht gut von den anderen Varianten getrennt ist. In anderen Darstellungen treten dagegen andere Kategorisierungen deutlicher hervor (Bild 10).

Unter Verwendung der MatPlotLib sind auch dreidimensionale Darstellungen kein Problem. Bild 11 zeigt den vollständigen Iris-Datensatz als 3-D-Streudiagramm.

Die einzelnen Abbildungen zeigen, dass die Iris-Blumen nicht anhand eines einzelnen Parameters wie etwa der Kelchblattlänge identifiziert werden können. Zudem ist es nicht möglich, in die Scatterplots gerade Linien einzuzeichnen, die alle Datencluster eindeutig voneinander trennen. Man spricht hier von einem „nicht linear separierbaren Problem.“

Im nächsten Beitrag zur Serie KI-Praxis wird gezeigt werden, wie es mithilfe von Neuronalen Netzen trotzdem gelingt, nur anhand der Blütenblattdaten die einzelnen Blumenarten zu klassifizieren.

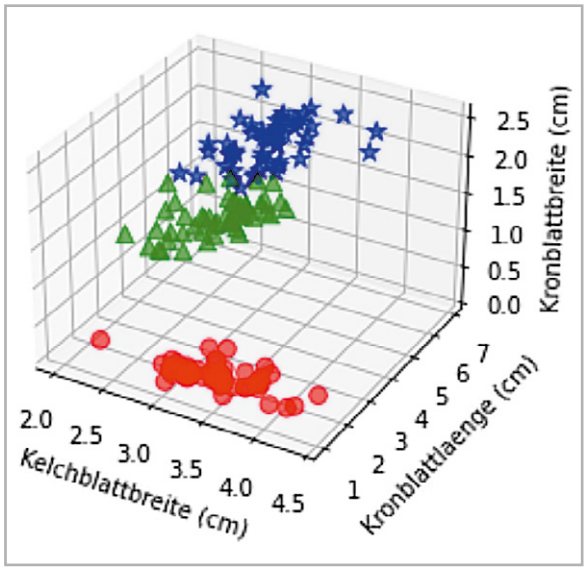


Bild 11: Dreidimensionaler Scatterplot

Fazit und Ausblick

In diesem Beitrag wurde gezeigt, wie man den Raspberry Pi für KI-Anwendungen fit machen kann. Mit dem Jupyter-Notebook steht eine leistungsfähige Oberfläche zu Verfügung, die es gestattet, auch komplexere Aufgaben zu lösen.

Anhand eines einfachen Datensatzes wurden die grafischen Fähigkeiten von Python und den zugehörigen Bibliotheken demonstriert.

Im nächsten Artikel wird dargelegt werden, wie Neuronale Netze in Jupyter konstruiert und trainiert werden können. Anhand des nun bereits bekannten Iris-Datensatzes wird gezeigt, wie ein lernfähiges System die einzelnen Iris-Exemplare korrekt klassifizieren kann.

Abschließend sei noch die Entwicklung eines USB-Sticks durch die Firma Google erwähnt. Dieser ist in der Lage, die Geschwindigkeit, mit der der Raspberry Pi trainierte Modelle für maschinelles Lernen ausführt, erheblich zu beschleunigen. Der Preis des Sticks liegt bei etwa 70 US-Dollar. Laut Google-Angaben können auch Kleinrechner wie der Raspberry Pi mit dem „Edge-TPU-Accelerator“ Computer-Vision-Modelle hochauflösende Videos mit mehr als 30 Bildern pro Sekunde bearbeiten. Damit wären selbst mit dem „Pi“ Echtzeitanwendungen in der Videodatenanalyse möglich. ELV

Weitere Infos:

[1] https://www.raspberrypi.org/software/
[2] Downloadpaket zum Beitrag: Artikel-Nr. 252090

Alle Links finden Sie auch online unter: de.elv.com/elvjournal-links

Material	Artikel-Nr.
Raspberry Pi 4 Model B, 8 GB RAM	250567
Raspberry Pi 4B, 4 GB Starterset	250983