FUTURE STEM HUB

Künstliche Intelligenz: Hands-On mit Python und Scratch

FUTURE-STEM-HUB

Empowering Secondary School STEM Education with AI Training and Resources for Students and

Projekt Nr.
2024-1-DE03-KA220-SCH-000247346

Co-funded by the European Union

Gemeinsam für alle!

Das Projekt "FUTURE-STEM-HUB – Stärkung der MINT-Bildung in der Sekundarstufe durch KI-Schulungen und Ressourcen für Lehrende und Lernende" (Projektnummer: 2024-1-DE03-KA220-SCH-000247346) wird im Rahmen des Erasmus+ Programms der Europäischen Union kofinanziert. Koordiniert wird es von der Universität Duisburg-Essen (Deutschland) in Zusammenarbeit mit vier weiteren Partnerorganisationen: M&M Profuture Training (Spanien), der Provinzdirektion Kütahya des Ministeriums für Nationale Bildung (Türkei), COOPETAPE – Kooperative für Bildung, CRL – Träger der ETAP-Schule (Portugal) sowie Tetra Solutions Ltd. (Bulgarien). Die Projektteammitglieder*innen aller Partnerorganisationen haben die FUTURE-STEM-HUB-Bildungsmaterialien zum Thema Künstliche Intelligenz entwickelt. Diese umfassen praxisnahe Aufgaben mit Python und Scratch, bei denen jede Partnerorganisation mit ihrer spezifischen Expertise ein eigenes Modul beigesteuert hat. Ziel des Projekts ist es, Schüler*innen der Sekundarstufe die Grundlagen der Künstlichen Intelligenz näherzubringen, das Bewusstsein für deren gesellschaftliche und ethische Implikationen zu schärfen und einen kritisch-reflexiven Perspektivwechsel anzuregen.

Autorenschaft

Mustafa Bilgin, Universität Duisburg-Essen (Deutschland)
Monica Moreno, M&M Profuture Training (Spanien)
Montserrat Renedo, M&M Profuture Training (Spanien)
João Barroso, ETAP School (Portugal)
Angelina Presa, ETAP School (Portugal)
Silviya Georgieva, Tetra Solutions Ltd. (Bulgarien)
Borislava Zaharieva-Tomova, Tetra Solutions Ltd. (Bulgarien)
Yeliz Yurter, Kütahya MEM (Türkei) Özcan Turan, Kütahya MEM (Türkei)

Editor, Übersetzung, redaktionelle Überarbeitung und inhaltliche Anpassung:
Mustafa Bilgin, University of Duisburg-Essen (Germany)
© 2026, Essen

Abbildungsverzeichnis

Layoutvorlage: Tetra Solutions Ltd.
Titelbild: Hybride Erstellung (Konzept & Zeichnungsvorlage: Mustafa Bilgin, Digitale Retusche: Generiert unter Verwendung von Google Gemini).
S. 6: Grafik von Tetra Solutions Ltd.
S. 7, 8: Hybride Erstellung (Konzept & Zeichnungsvorlage: Mustafa Bilgin, Digitale Retusche: Generiert unter Verwendung von Google Gemini).
Die Icons stammen von Flaticon.com (High quality icons).

Haftungsausschluss und Sicherheit

Die Nutzung der in diesen Materialien bereitgestellten Programmiercodes sowie der Zugriff auf externe Webseiten und Plattformen von Drittanbietern erfolgen auf eigene Gefahr und Verantwortung der Nutzer*innen. Das Projektteam übernimmt keine Haftung für die Inhalte, Funktionalität oder Verfügbarkeit externer Links, wie etwa Google Colab, Scratch oder Machine Learning for Kids. Da viele der interaktiven Aufgaben eine aktive Internetverbindung und die Verarbeitung von Daten auf Cloud-Servern erfordern, liegt die Verantwortung für die Einhaltung von Sicherheitsstandards und Systemvoraussetzungen (wie aktuelle Browserversionen) bei den Anwender*n. Bei der Verwendung von Drittanbieter-Tools zur Erstellung von KI-Modellen dürfen keine persönlichen oder sensiblen Informationen hochgeladen werden. Da sich die Bildungsmaterialien an die Zielgruppe der 15- bis 18-Jährigen richten, sollte die Registrierung bei externen Diensten sowie die Durchführung der Online-Aktivitäten unter Berücksichtigung der jeweiligen Nutzungsbedingungen der Anbieter und idealerweise unter Aufsicht und Zustimmung einer Lehrkraft oder eines Erziehungsberechtigten erfolgen. Für etwaige Datenverluste oder Beeinträchtigungen der genutzten Endgeräte durch die Ausführung der bereitgestellten Code-Beispiele wird keine Haftung übernommen.

Dieses Werk ist lizenziert unter einer Creative Commons Namensnennung. Weitergabe unter gleichen Bedingungen 4.0 International Lizenz (CC BY-NC-SA 4.0). https://creativecommons.org/licenses/by-nc-sa/4.0/deed.de

Abkürzungsverzeichnis

KI: Künstliche Intelligenz

ML: Maschinelles Lernen

DL: Deep Learning

NN: Neuronales Netzwerk

NLP: Verarbeitung natürlicher Sprache

CV: Computer Vision

CNN: Convolutional Neural Network

RGB: Rot, Grün, Blau (Farbmodell)

YOLO: You Only Look Once

FPS: Bilder pro Sekunde

TTS: Text-to-Speech (Sprachausgabe)

MNIST: Modified National Institute of Standards and Technology (Datensatz)

IDE: Integrierte Entwicklungsumgebung

ML4K: Maschinelles Lernen für Kinder

IBM: International Business Machines

MIT: Massachusetts Institute of Technology

EACEA: Europäische Exekutivagentur für Bildung und Kultur

UNESCO: United Nations Educational, Scientific and Cultural Organization

STEM: Naturwissenschaften, Technologie, Ingenieurwesen und Mathematik

Inhalt

Projektübersicht.....5

Projektergebnisse.....5

Einleitung.....6

Modul 1: Mini-Programmierherausforderungen: Maschinelles Lernen und neuronale Netze.....8

Modul 2: AI Escape Room Schatzsuche mit Taschenrechner.....39

Modul 3: Scratch trifft auf künstliche Intelligenz.....60

Modul 4: Python-Spieleprogrammierung (Pygame / Arcade-Bibliotheken) mit KI 82

Modul 5: Objekterkennung mit Roboflow — Einführung in die Computer Vision.....104

Zusammenfassung.....119

Diese Veröffentlichung ist lizenziert unter einer Creative Commons Namensnennung – Nicht kommerziell – Keine Bearbeitungen 4.0 International Lizenz.

Empowering Secondary School STEM Education with AI
Training and Resources for Students and Educators
2024-1-DE03-KA220-SCH-000247346

FUTURE-STEM-HUB

Projekt-übersicht

Das Projekt zielt darauf ab, die Integration von Künstlicher Intelligenz in den MINT-Unterricht der Sekundarstufe voranzutreiben. Dazu werden drei zentrale Ansätze verfolgt: (1) Bereitstellung von Lehrmaterialien zu grundlegenden KI-Konzepten und deren gesellschaftlichen Auswirkungen. (2) Entwicklung praxisorientierter Ressourcen, die Schüler*innen einen aktiven Zugang zur KI ermöglichen, insbesondere durch die Arbeit mit Python. (3) Qualifizierung und Unterstützung von Lehrkräften bei der Integration von KI-Inhalten in ihren MINT-Unterricht.

Projekt-ergebnisse

1. **Kurs 1: Digitale Einführung: Grundlagen der künstlichen Intelligenz** (Interaktive Lehrmaterialien zu den Grundlagen der KI für Schüler*innen der Sekundarstufe)

2. **Kurs 2: Vertiefung in KI mit Python und Scratch** (Fortgeschrittene KI-Konzepte – praktische Materialien für Schüler*innen der Sekundarstufe)

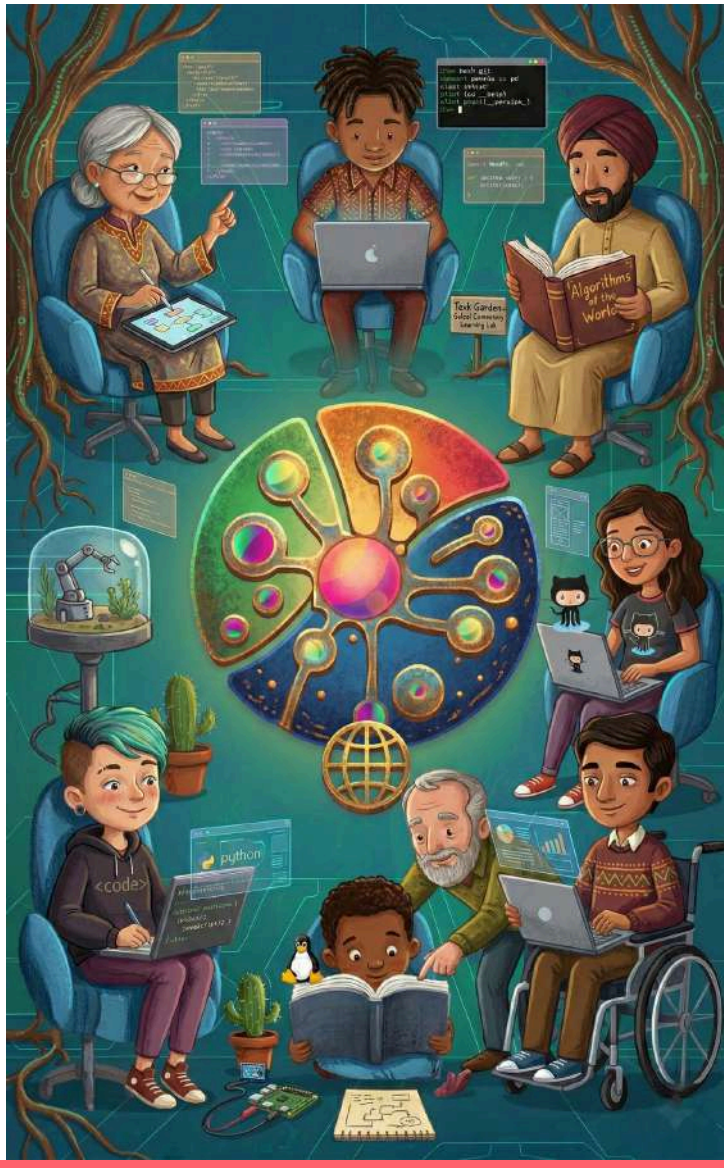
3. **E-Toolkit für Lehrkräfte: Verbesserung der KI-Kompetenzen** (Methodenleitfaden zur Integration Künstlicher Intelligenz in den Schulunterricht)

Einführung

Willkommen zu den umfassenden, praxisnahen Lehr- und Lernmaterialien mit Aufgaben in Python und Scratch. Sie und Ihre Schüler*innen erwartet eine spannende und interaktive Lernreise in die faszinierende Welt der Künstlichen Intelligenz. Die folgenden Materialien bieten Lehrkräften und auch Lernenden gleichermaßen leicht zugängliche, anwendungsorientierte Inhalte, die zentrale KI-Konzepte mithilfe von Python und Scratch explorativ vermitteln. Die Materialien sind auf die Zielgruppe der 15- bis 18-Jährigen mit individuellen Vorkenntnissen in den MINT-Fächern zugeschnitten und eignen sich mit Unterstützung der Lehrkraft sowohl für selbstgesteuertes, asynchrones Lernen als auch für die nahtlose Integration in den Präsenzunterricht.

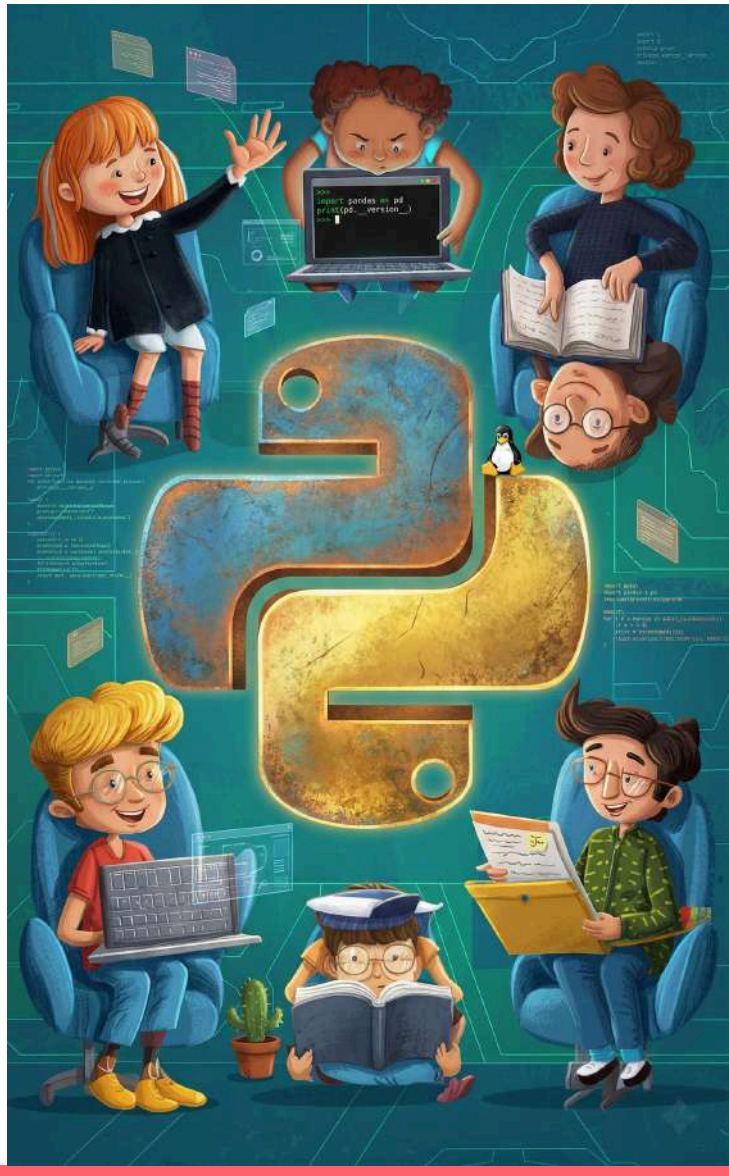
Die fünf Module behandeln praxisorientierte Programmierkenntnisse in Python und Scratch, mathematische Logik mittels eines Escape Rooms, kreative KI-Anwendungen, datenbasierte Lösungen für reale Probleme sowie ethische Aspekte der Künstlichen Intelligenz.

Jedes Modul verbindet theoretische Grundlagen mit integrierten Quizfragen, externen Plattformen zur weiterführenden Recherche und praktischen Übungen, um ein umfassendes Verständnis von KI-Konzepten und -Methoden zu ermöglichen. Die geschätzte Bearbeitungszeit für alle fünf Module beträgt etwa 10 Stunden.



Nach Abschluss des Kurses werden die Schüler*innen in der Lage sein, zentrale KI-Konzepte wie maschinelles Lernen, neuronale Netze und Deep Learning zu verstehen und auch zu definieren. Sie erlangen Kenntnisse über die Geschichte und Entwicklung der Künstlichen Intelligenz sowie deren Anwendungen in verschiedenen Bereichen des Alltags und lernen, wie neue KI-Technologien aktuelle Trends und unsere Lebenswelt prägen. Darüber hinaus setzen sie sich mit grundlegenden Technologien wie Computer Vision und natürlicher Sprachverarbeitung auseinander. Schließlich erforschen sie die ethischen Dimensionen der KI, insbesondere im Hinblick auf Verantwortung und gesellschaftliche Auswirkungen. Außerdem entwickeln sie ein reflektiertes Verständnis für diese Themenbereiche. Lehrkräfte und auch interessierte erhalten Zugang zu innovativen, interaktiven Ressourcen, die sich einfach in den Unterricht integrieren lassen. Diese Materialien helfen ihnen, spannende und lernzielorientierte Lernumgebungen zu schaffen, indem sie vielseitige Ansätze bieten, um Schüler*innen der Sekundarstufe zum Thema KI zu motivieren.

Achja, die Materialien werden anschließend in einen Online-Kurs überführt, der auf der FUTURE-STEM-HUB-Plattform gehostet und über die Projektwebsite www.future-stem-hub.eu zugänglich ist.



Nach erfolgreichem Abschluss des Kurses erhalten die Studierenden ein digitales Zertifikat, das ihre Leistung bestätigt.

Modul 1: Mini-Programmierherausforderungen: Maschinelles Lernen und neuronale Netze

EINFÜHRUNG

Hast du dich jemals gefragt, woher dein Smartphone weiß, welche Videos dir gefallen könnten? Oder wie eine App deine Handschrift erkennt? Dahinter steckt Künstliche Intelligenz (KI) und die ist viel weniger geheimnisvoll, als viele denken!

In diesem Kapitel begleitest du Aylin und ihre clevere Smartwatch Alara. Gemeinsam erleben die zwei spannende Abenteuer, die dir zeigen, wie KI wirklich funktioniert. Du wirst die grundlegenden Konzepte und Komponenten von Neuronalen Netzen und Maschinellem Lernen verstehen und lerne. Du wirst sehen, wie sie Eingaben verarbeiten, aus Daten lernen und Entscheidungen zu treffen. Und das Beste: Du musst kein Programmierprofi sein! Jedes Abenteuer enthält kleine Programmieraufgaben in drei Schwierigkeitsstufen. Wähle einfach die passende für dich aus – vom Anfänger bis zur echten Herausforderung. Du wirst sehen: Schon nach dem ersten Kapitel verstehst du, wie Maschinen „denken lernen“. Lass uns gemeinsam in die Welt der KI eintauchen! Am Ende des Moduls hast du dir folgende Kompetenzen erarbeitet:

- **Einführung in die KI: Die Uhr, die mehr kann als ticken**
- **Neuronale Netze: Alara erwacht**
- **Entscheidungsbäume: Der Schulwegdetektiv – Wie Ihre Uhr denkt**
- **Regelbasierte Klassifizierung: Wie vielleicht Spotify weiß, was du hören willst**
- **Verstärkendes Lernen: Lernen durch Belohnung**
- **Faltungsneuronale Netze: Wie KI Kanten in Bildern erkennt**

- **Verarbeitung natürlicher Sprache: Wie Computer Sprache verstehen**
- **Empfehlungssysteme: Wie YouTube weiß, was du sehen möchtest**
- **Random Forest: Im Wald der Entscheidungen – Wettervorhersage**
- **KI-Ethik: Was darf KI? Und was nicht? Eine Zukunftsfrage**

Moduldauer	Moduldauer: 2 Stunden (Theorie: 1 Std. | Praxis: 1 Std.)

UNTERRICHTSMATERIALIEN

AYLIN ist 17 Jahre alt, neugierig und liebt es, herauszufinden, wie die Dinge funktionieren. Sie gibt niemals auf!

LENA ist Aylins beste Freundin. Sie ist kreativ und liebt Musik. Ihre Familie kommt aus Polen, und manchmal bringt sie köstliche Piroggen mit!

ALARA ist keine gewöhnliche Smartwatch. Sie ist eine KI, die Aylin die Welt der künstlichen Intelligenz erklärt.

Abbildung 1: Aylin, Alara und Lena; Konzept und Layout: Mustafa Bilgin; optimiert mit Google Gemini

EINHEIT 1.1 KURSKONZEPT

Der Lernprozess ist in einen erzählerischen Rahmen eingebettet, der MINT-Themen mit lebensnahen, nachvollziehbaren Kontexten verknüpft. Komplexe Konzepte wie neuronale Netze oder Reinforcement Learning werden vereinfacht und prototypisch dargestellt, sodass Lernende ein grundlegendes Verständnis entwickeln können, ohne dass Programmierkenntnisse erforderlich sind. Das Material eignet sich ideal für den Informatik- und Technikunterricht, für MINT-Projekte, Arbeitsgemeinschaften sowie interdisziplinäre Lernumgebungen. Es fördert nicht nur das technische Verständnis, sondern regt insbesondere durch das Kapitel zur KI-Ethik, auch zur kritischen Auseinandersetzung mit moralischen und gesellschaftlichen Fragestellungen an.

EINHEIT 1.2 EINFÜHRUNG IN DIE KÜNSTLICHE INTELLIGENZ (KI)

Bevor wir in die einzelnen Aktivitäten eintauchen, lohnt sich ein Blick in die Grundlagen: Künstliche Intelligenz bezeichnet die Simulation menschlicher Intelligenz in Maschinen. Sie sind so programmiert, dass sie „ähnlich wie Menschen denken“ und deren Handlungen nachahmen können. Der Begriff umfasst auch Maschinen, die Eigenschaften des menschlichen Denkens, wie Lernfähigkeit und Problemlösungskompetenz, zeigen. In diesem Modul erkunden wir die grundlegenden Bausteine der KI anhand interaktiver Geschichten.

- **Neuronale Netze** sind Rechensysteme, die nach dem Vorbild der biologischen neuronalen Netze im Gehirn von Tieren entwickelt wurden. Sie ermöglichen es der KI, aus Beispielen zu „lernen“.

- **Entscheidungsbäume** helfen bei der Entscheidungsfindung. Sie stellen Entscheidungen und ihre möglichen Folgen in einer baumartigen Struktur dar. Dabei folgen sie der einfachen Logik: „Wenn dies passiert, dann tue das.“

- **Maschinelles Lernen (ML)** ist ein Teilbereich der Künstlichen Intelligenz. Es befähigt Systeme, aus Erfahrungen zu lernen und sich kontinuierlich zu verbessern – und das ganz ohne explizite Programmierung.

Dieses Modul setzt auf Storytelling, um komplexe mathematische Konzepte verständlich und zugänglich zu vermitteln.

EINHEIT 1.3 SYSTEMEINRICHTUNG & VORAUSSETZUNGEN

Welche Plattform wird benötigt?

In diesem Modul arbeitest du mit Google Colab – einer kostenlosen Online-Plattform, auf der du Python-Code direkt in deinem Browser schreiben und ausführen kannst. Installieren musst du gar nichts!

Hier geht's los:
https://colab.research.google.com

Ist eine Registrierung erforderlich?

Um Google Colab nutzen zu können, benötigst du ein kostenloses Google-Konto (also ein Gmail-Konto). Wenn du bereits eines hast, kannst du dich direkt anmelden. Andernfalls erstellst du vor Beginn einfach ein neues Konto.

So erstellst du ein Google-Konto:
Besuche accounts.google.com
und folge den Schritten zur Kontoerstellung.

Der Registrierungsprozess

1. Gehe zu Google Colab.
2. Klicken Sie oben rechts auf „Anmelden".
3. Gib deine Google-E-Mail-Adresse und dein Passwort ein.
4. Nach der Anmeldung kannst du ein Notizbuch öffnen oder ein neues erstellen.

Wo du den Code schreiben sollst

Die Programmierübungen in Modul 1 werden über ein vorbereitetes Google Colab-Notebook durchgeführt. Du musst nichts programmieren – das Notebook enthält bereits alle Übungen und Anleitungen.

Hier geht's zum Notebook:
>Modul 1 Gemeinsames Notizbuch<
(www.bit.ly/4rPXwVK)

Was tun mit dem Notizbuch?

1. Öffne den oberen Link (Modul 1).
2. Klick auf „In Drive kopieren" – schon gehört das Notebook dir und du kannst es nach Herzenslust bearbeiten.
3. Einfach der Reihe nach: Anweisungen lesen, Codezelle ausführen, weiter zum nächsten Schritt.

EINHEIT 1.4 ZUSÄTZLICHE ANWEISUNGEN VOR DEM START

1. Verwende einen modernen Webbrowser wie Chrome, Firefox, Edge oder Safari.
2. Sorge für eine stabile Internetverbindung – Google Colab läuft online.
3. Lies dir die Geschichte und die Anweisungen für jede Station sorgfältig durch, bevor du mit der Programmieraufgabe beginnst.
4. Jede Station bietet drei Schwierigkeitsstufen:

    - Basic – für Anfänger*innen ohne Vorkenntnisse
    - Fortgeschritten – wenn du bereits erste Erfahrung mit Python hast
    - Experte – für wirklich knifflige Herausforderungen

    Wähl einfach das Level, das am besten zu dir passt.
5. Führe die Codezellen der Reihe nach aus. Einige Übungen bauen aufeinander auf – also besser nicht springen!
6. Wenn du nicht weiterkommst: Nutz den „Hinweis“ oder den „Lösungsausschnitt“ in jeder Übung.
7. Experimentiere! Nachdem du eine Aufgabe geschafft hast, probier ruhig aus, den Code zu verändern und zu sehen, was passiert. Durch Ausprobieren lernst du am besten.

REFERENZEN

Zuordnung von Emojis und Symbolen

- Dieses Dokument verwendet Standard-Unicode-Emojis gemäß der Definition des Unicode-Konsortiums. Die verfügbaren Emoji-Versionen können je nach Betriebssystem des Nutzers variieren. Unicode-Konsortium. (2022). Abgerufen von https://unicode.org/emoji/charts/full-emoji-list.html.

Bücher & Artikel

- Russell, S., & Norvig, P. (2021). Künstliche Intelligenz: Ein moderner Ansatz (4. Aufl.). Pearson. (Ein umfassendes Lehrbuch zu KI-Konzepten, einschließlich neuronaler Netze, Entscheidungsbäume und Reinforcement Learning.)

- Géron, A. (2022). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow (3. Aufl.). O'Reilly. (Praktische Einführung in maschinelles Lernen und neuronale Netze mit Python.)

KI-Ethik & Gesellschaft

- Europäische Kommission. (2019). Ethische Leitlinien für vertrauenswürdige KI. (Ein Rahmenwerk für die ethische Entwicklung und den Einsatz von KI.)

- UNESCO. (2021). Empfehlung zur Ethik der künstlichen Intelligenz. (Globale Leitlinien zur KI-Ethik mit Schwerpunkt auf Menschenrechten und Inklusion.)

Python-Programmierung für Anfänger

- Sweigart, A. (2019). Automate the Boring Stuff with Python (2. Aufl.). No Starch Press. (Einsteigerfreundliches Python-Buch mit praktischen Projekten.)

LINKS ZU LERNPLATTFORMEN

- Hier ist das Notebook für Modul 1 – https://colab.research.google.com/drive/14VaJnlgo5habZ0-N9jpz9nWZSVAw_Xwa?usp=sharing

- Offizielle Python-Dokumentation – https://docs.python.org/3/ (Offizielle Python-Sprachreferenz und Tutorials.)

PRAKTISCHE ÜBUNG

Station 1: Das geheimnisvolle Geschenk – Die Uhr, die mehr kann

Aylin feiert ihren 17. Geburtstag – zwischen Baklava und Luftballons liegt ein geheimnisvolles kleines Päckchen. Aylin zögert, es zu öffnen.

AYLIN: „Was ist das, Tante Sema?“

TANTE SEMA: „Etwas Besonderes, canım (meine Süße).
Deine Oma würde sagen: Diese Uhr hat eine Seele.“

Aylin wickelt das Päckchen vorsichtig aus. Zum Vorschein kommt eine elegante Smartwatch mit einem türkisblauen Armband. Plötzlich leuchtet das Display auf.

ALARA (die Uhr): „Merhaba (Hallo), Aylin! Ich bin Alara – deine neue KI-Freundin!“

Aylin erschrickt und springt zurück.

AYLIN: „Die … die Uhr spricht?“

TANTE SEMA: „Sie spricht jede Sprache, die du verstehst. Alara wird dir zeigen, wie künstliche Intelligenz wirklich funktioniert!“

ALARA: „Stell dir vor, ich bin wie dein Gehirn: Ich habe Neuronen, die lernen, und Synapsen, die Verbindungen herstellen. Gemeinsam entdecken wir, wie Maschinen denken lernen!“

BASIC

CODE:

```
name = input("Wie heißt du? ")
print("Hallo " + ____ + "! Ich bin Alara, deine KI-Freundin")
```

Lösungsschnipsel: name

Hinweis: Verwende die Variable.

Station 1: Das geheimnisvolle Geschenk – Die Uhr, die mehr kann

ADVANCED

CODE:

```
print("🎂 Aylins Geburtstagsplanung 🎂")
favorite_food = input("Was ist dein Lieblingsessen? (Pizza/
Döner/Burger): ")
drink = input("Und was möchtest du dazu trinken? ")

if favorite_food.lower() == "pizza":
    print("🍕 Alara sagt: Pizza ist auch mein Favorit!")
else:
    print("😋 Alara sagt: " + ____ + " klingt super!")

print("Und " + ____ + " dazu? Perfekte Kombination!")
```

Lösungsschnipsel: favorite_food, drink

Hinweis: Fülle die Lücken mit deinen Eingabevariablen aus.

Station 2: Neuronale Netze – Alaras Erwachen

Am nächsten Morgen sitzt Aylin im Park und beobachtet Alara.

AYLIN: „Also … denkst du wie ich?“

ALARA: „Fast! Stell dir mein ‚Gehirn‘ wie eine große Denkfabrik vor.
Ich habe Neuronen, die lernen. Schau mal …“

AYLIN: „Du wirst also mit mir lernen?“

ALARA: „Ja“, antwortet Alara. „Ich kenne die Grundlagen,
aber durch dich lerne ich, die Welt mit menschlichen Augen zu sehen.“

Station 2: Neuronale Netze – Alaras Erwachen

BASIC

CODE:

```
print("Soll ich ein neues Spiel kaufen?")
print("Bewerte von 0 bis 1, wobei 0 = gar nicht, 0.5 = okay,
1 = sehr sehr")

savings_balance = float(input("Wie voll ist dein Sparschwein?
(0-1): "))
desire = float(input("Wie sehr willst du das Spiel? (0-1):
"))

def decision_neuron(money, wish):
    if money * wish > ____:
        return "✅ JA, kauf es!"
    else:
        return "❌ NEIN, warte lieber"

print(decision_neuron(savings_balance, desire))
```

Lösungsschnipsel: Ein Neuron „feuert", wenn das Produkt größer als 0.5 ist.

Hinweis: A neuron 'fires' when the product is greater than 0.5.

Station 2: Neuronale Netze – Alaras Erwachen

ADVANCED (Part I)

CODE:

```
print("Ist das eine gute Freundschaft?")
print("Bewerte von 0 bis 1, wobei 0 = gar nicht, 0.5 = okay,
1 = sehr sehr")

trust = float(input("Wie sehr vertraust du dieser Person?
(0-1): "))
fun = float(input("Wie viel Spaß habt ihr zusammen? (0-1):
"))
helpfulness = float(input("Wie hilfreich ist diese Person?
(0-1): "))

# Gewichtung - Was zählt am meisten in einer Freundschaft?
weight_trust = 0.5
weight_fun = 0.3
weight_help = 0.2

def friendship_analysis(trust, fun, help):
    inputs = [trust, fun, help]
    weights = [weight_trust, weight_fun, weight_help]

    total = 0
    for i in range(len(inputs)):
        total += inputs[i] * ____
    return total
```

Station 2: Neuronale Netze – Alaras Erwachen

```
ADVANCED (Part II)

CODE:
score = friendship_analysis(trust, fun, helpfulness)
print("Freundschafts-Punktzahl:", score)

if score > 0.7:
    print("💖 Echte Freundschaft!")
elif score > 0.4:
    print("👍 Gute Bekanntschaft")
else:
    print("🤔 Vielleicht nicht die beste Freundschaft")
```

Lösungsschnipsel: weights[i]

Hinweis: Multipliziere jeden Input mit dem dazugehörigen Gewicht.

Station 3: Entscheidungsbäume – Der Schulwegdetektiv

Aylin steht an der Bushaltestelle. Es regnet.
Sie unterhält sich mit Alara über ihren Schulweg.

ALARA: „Ich habe eine Idee für deinen Schulweg!
Lass uns einen Entscheidungsbaum bauen."

Alara hilft ihr bei der Auswahl verschiedener Möglichkeiten
– das nennt man einen Entscheidungsbaum.

AYLIN: „Super! Jetzt weiß ich immer, welchen Weg ich nehmen muss",
sagt Aylin glücklich.

ALARA: „Und ich lerne dabei, wie du Entscheidungen triffst."

Station 3: Entscheidungsbäume – Der Schulwegdetektiv

BASIC

CODE:

```
weather = input("Wie ist das Wetter? (Sonne/Regen): ")

def school_route(weather):
    if weather == "Regen":
        return "☔ Nimm den Schirm mit!"
    else:
        return "🌞 ____"

print(school_route(weather))
```

Lösungsschnipsel: Was tust du, wenn es nicht regnet?

Hinweis: Was machst du, wenn es nicht regnet?

ADVANCED

CODE:

```
weather = input("Wetter? (Regen/Sonne): ")
bus_coming = input("Kommt der Bus? (ja/nein): ")

def plan_route(weather, bus_coming):
    if weather == "Regen" and bus_coming == "ja":
        return "🚌 Nimm den Bus!"
    elif weather == "Regen" and bus_coming == "nein":
        return "☔ ____"
    else:
        return "🚶‍♀️ Zu Fuß gehen!"

print(plan_route(weather, bus_coming))
```

Lösungsschnipsel: Schirm mitnehmen

Hinweis: Wenn der Bus nicht kommt, hilft nur der Schirm.helps.

Station 4: Regelbasierte Klassifizierung – Wie Spotify weiß, was du magst

Aylin und ihre Freundin Lena hören Musik.

LENA: „Woher weiß deine Uhr eigentlich, dass ich Rock mag?“

ALARA: „Das ist regelbasierte Klassifizierung! Ich gruppiere ähnliche Songs.“

LENA: „Das ist ja wie Zauberei!“, ruft Lena begeistert.

ALARA: „Keine Zauberei“, berichtigt Alara. „Nur Mustererkennung und ich lerne deinen Musikgeschmack ständig besser kennen.“

BASIC

CODE:

```
song = input("Beschreibe dein Lied: ... (z.B. Gitarre): ")

def sort_music(song):
    if "Gitarre" in song:
        return "🎸 Rock"
    elif "____" in song:
        return "🎤 Pop"
    else:
        return "🎶 Andere"

print(sort_music(song))
```

Lösungsschnipsel: danceable

Hinweis: Tanzbare Songs sind meist Pop.

Vereinfachte Gruppierung: In dieser Übung gruppieren wir Hörer*innen nach ihrer stärkeren Musikvorliebe. K-Means ist ein unsupervised (unüberwachtes) Lernverfahren. Eine Klassifizierung ordnet Daten in bekannte Kategorien ein, während K-Means neue Gruppen (Cluster) in unbekannten Daten findet.

Station 5: Verstärkendes Lernen – Lernen durch Belohnung

Aylin spielt mit einer Hundetrainings-App auf ihrem Tablet.

AYLIN: „Wie bringe ich diesem virtuellen Hund bei, was er tun soll?“

ALARA: „Ganz einfach: durch Belohnung! Wenn er etwas richtig macht, bekommt er ein Leckerli. So lernt er nach und nach, welche Aktionen sich lohnen. Das nennt man Verstärkendes Lernen!“

AYLIN: „Also wie in der Schule – gute Noten für richtige Antworten?“

ALARA: „Genau! Aber hier bestimmst du, was ‚richtig‘ ist.
Der Hund probiert Dinge aus und merkt sich, was funktioniert.“

AYLIN: „Lernst du also genau wie dein virtueller Hund?“,
fragt Aylin mit einem Lächeln.

ALARA: „Irgendwie schon“, gibt Alara zu.
„Belohnungen wirken bei allen Lernenden.“

BASIC (Part I)

CODE:

```
import random

print("Trainiere deinen virtuellen Hund!")
print("Belohne mit einem Leckerli (1) oder ignoriere (0)\n")

actions = ["Sitz", "Platz", "Pfote geben"]
probabilities = [0.3, 0.3, 0.4]  # Startwerte
disturb_chance = 0.2  # 20% Chance, dass der Hund etwas
anderes tut

def normalize(probabilities):
    total = sum(probabilities)
    return [p / total for p in probabilities]
```

Station 5: Verstärkendes Lernen – Lernen durch Belohnung

BASIC (Part II)

CODE:

```
for round in range(1, 6):
    print(f"\n Trainingseinheit {round}/5")

    # Hund wählt eine Aktion basierend auf seinen Vorlieben
    action = random.choices(actions,
weights=_________________)[0]  # Lücke 1

    # Ablenkung: Hund könnte etwas anderes tun
    if random.random() < disturb_chance:
        possible_actions = [a for a in actions if a !=
action]
        wrong_action = random.choice(possible_actions)
        print(f"Der Hund sollte '{action}' machen, macht aber
stattdessen: 🐶 {wrong_action}")
        action = wrong_action
    else:
        print(f"Der Hund macht: 🐾 {action}")

    # Benutzereingabe mit Eingabeüberprüfung
    while True:
        user_input = input("Leckerli geben? (1=ja, 0=nein):
").strip()
        if user_input in ("1", "0"):
            reward = int(user_input)
            break
        print("Bitte nur 1 oder 0 eingeben!")

    index = actions.index(action)
```

Hinweis: Wenn der Bus nicht kommt, hilft nur der Schirm.

Lösungsschnipsel: Schirm mitnehmen

STATION 5: Reinforcement Learning — Learning Through Reward

BASIC (Part III)

CODE:

```
    # Lernlogik
    if reward == 1:
        probabilities[index] += ______  # Lücke 2
        print("✅ Hund ist glücklich: 'Das war gut!'")
    else:
        probabilities[index] = max(0.1, probabilities[index]
- 0.1)
        print("❌ Hund denkt: 'Das war nicht so toll...'")

    # Wahrscheinlichkeiten normalisieren
    probabilities = normalize(probabilities)

    print("Aktuelle Lernwerte:", [round(p, 2) for p in
probabilities])

# Ergebnisanzeige
print("\n Training abgeschlossen!")
for action, value in zip(actions, probabilities):
    print(f" {action}: {value:.2f}")

favorite_action =
actions[probabilities.index(max(probabilities))]
print(f"\n🐕 Liebling des Hundes: {favorite_action} 🦴")
```

Hinweis 1: Hier werden die aktuellen Wahrscheinlichkeiten gespeichert, mit denen der Hund eine Aktion auswählt.

Hinweis 2: Wie stark soll die Wahrscheinlichkeit steigen, wenn der Hund belohnt wird?

Lösungsschnipsel 1: probabilities

Lösungsschnipsel 2: 0.2

STATION 5: Reinforcement Learning — Learning Through Reward

ADVANCED (Part I)

CODE:

```
print("Halte deine Pflanze am Leben!")
print("Aktionen: water, fertilize, do_nothing")

plant_state = {
    "water": 5,        # 0-10
    "nutrients": 5,
    "health": 10
}

def evaluate_state():
    water, nutrients = plant_state["water"],
plant_state["nutrients"]
    if 3 <= water <= 7 and 3 <= nutrients <= 7:
        return ____    # Perfekt!
    elif 1 <= water <= 9 and 1 <= nutrients <= 9:
        return _____   # Okay
    else:
        return -1      # Schlecht

for day in range(7):
    print(f"\n--- Tag {day+1} ---")
    print(f"Zustand: Wasser={plant_state['water']}/10,
Nährstoffe={plant_state['nutrients']}/10")

    action = input("Was tust du? (water/fertilize/nothing):
")
```

STATION 5: Reinforcement Learning — Learning Through Reward

ADVANCED (Part II)

CODE:

```
    # Aktion ausführen
    if action == "water":
        plant_state["water"] = min(10, plant_state["water"] +
3)
    elif action == "fertilize":
        plant_state["nutrients"] = min(10,
plant_state["nutrients"] + 3)

    # Zeit vergeht - Zustand nimmt ab
    plant_state["water"] = max(0, plant_state["water"] - 1)
    plant_state["nutrients"] = max(0,
plant_state["nutrients"] - 1)

    # Belohnung berechnen
    reward = evaluate_state()
    print(f"Belohnung: {reward} Punkte")

    if plant_state["water"] == 0 or plant_state["nutrients"]
== 0:
        print("💀 Die Pflanze ist gestorben!")
        break

if plant_state["water"] > 0 and plant_state["nutrients"] > 0:
    print("❤️🌱 Pflanze überlebt! Gut gemacht!")
```

Hinweis 1: Maximale Belohnung für ideale Wasser- und Nährstoffwerte.

Hinweis 2: Weniger Punkte für akzeptable Werte
(noch okay, aber nicht perfekt).

Lösungsschnipsel 2: 1

Lösungsschnipsel 1: 2

Station 6: Faltungsneuronale Netze – Wie KI Kanten in Bildern erkennt

Aylin betrachtet ihr Foto.

ALARA: „Ich sehe Linien, Helligkeiten und Kontraste – das sind meine Kanten! Wenn sich etwas verändert, weiß ich: Hier beginnt ein neues Objekt."

ALARA: „Um das zu erkennen, benutze ich einen kleinen Trick. Ich vergleiche immer zwei benachbarte Pixel. Wenn sie unterschiedlich sind, erscheint eine Kante. Der Operator != bedeutet ‚nicht gleich' oder ‚verschieden von'."

ALARA: „Jedes Bild, das du mir zeigst, hilft mir, besser zu ‚sehen'", erklärt Alara. „Genau wie du lernst, beschreibe ich die Welt immer genauer."

BASIC (Part I)

CODE:

```
def count_edges(row):
    edges = 0
    for i in range(len(row)-1):
        if row[i] ___ row[i+1]:
            edges += 1
    return edges

print("Alaras 4-zeiliges Bild")

# 4 Zeilen vom Benutzer eingeben
image = []
for n in range(4):
    row = input(f"Zeile {n+1} eingeben (0 und 1, z.B.
100101): ")
    row = row[:6]
    image.append(row)
```

Station 6: Faltungsneuronale Netze – Wie KI Kanten in Bildern erkennt

BASIC (Part II)

CODE:

```
# Kanten pro Zeile zählen
for i, row in enumerate(image):
    edges = count_edges(row)
    if edges > 0:
        print(f" Zeile {i+1}: {edges} Kanten")
    else:
        print(f"  Zeile {i+1}: keine Kanten")
```

Lösungsschnipsel: i=

Hinweis: Ein Wechsel (0 → 1 oder 1 → 0) ist eine Kante.

ADVANCED (Part I)

CODE:

```
print("Erstelle dein eigenes 4x4 Bild!")
size = 4
image = []

# Zeilen eingeben
for i in range(size):
    row = input(f"Zeile {i+1} eingeben (nur   und ■, z.B.
■   ■): ")
    row = list(row.ljust(size, "  "))[:size]
    image.append(row)

print("\n Dein Bild:")
for row in image:
    print(" ".join(row))
```

Station 7: Regelbasierte Verarbeitung natürlicher Sprache – Wie Computer Sprache verstehen

Aylin schreibt eine Nachricht an Lena: „Hey, das war großartig!“

ALARA: „Ich kann erkennen, ob du glücklich, wütend oder neutral bist – das nennt man Sentimentanalyse!“

ALARA: „Deine Nachrichten helfen mir, menschliche Sprache besser zu verstehen“, sagt Alara dankbar. „Wir lernen gemeinsam – du bringst mir KI bei und ich lerne menschliche Kommunikation.“

BASIC

CODE:

```
print("Alara liest deine Stimmung!")
text = input("Schreib etwas (z.B. … das war toll / … das war schlecht): ")

def understand(text):
    if "toll" in text:
        return "😊 Du bist gut gelaunt!"
    elif "schlecht" in text:
        return ____
    else:
        return "😐 Neutral."

print(understand(text))
```

Lösungsschnipsel: "😠 Du bist wütend!"

Hinweis: “schlecht“ = negativ

Station 7: Regelbasierte Verarbeitung natürlicher Sprache – Wie Computer Sprache verstehen

ADVANCED

CODE:

```
print("Wörter zählen wie ein Sprachmodell!")
text = input("Schreib einen Satz: ")

def count_words(text):
    words = text.split()
    return len(____)

print("Anzahl der Wörter:", count_words(text))
```

Lösungsschnipsel: words

Hinweis: Was enthält alle Wörter?

Station 8: Empfehlungssysteme – Wie YouTube weiß, was du sehen möchtest

Aylin schaut Videos auf ihrem Tablet.

ALARA: „Ich merke mir, was dir gefällt, und schlage ähnliche Inhalte vor!“

ALARA: „Je mehr du mir zeigst, was du magst, desto besser kann ich Vorschläge machen“, erklärt Alara. „Das ist wie eine Partnerschaft.“

BASIC

CODE:

```
print("Alaras Videoempfehlung")
video = input("Was schaust du? (z.B. Musikvideo, Tiervideo):
")

def recommend(video):
    if "Musik" in video:
        return "🎶 Ähnliche Musikvideos!"
    else:
        return ____

print(recommend(video))
```

Lösungsschnipsel: „Probier mal was Neues!“

Hinweis: Wenn es kein Musikvideo ist, schlag etwas Neues vor.

Station 8: Empfehlungssysteme – Wie YouTube weiß, was du sehen möchtest

ADVANCED

CODE:

```
print("DEIN YOUTUBE-ASSISTENT")
print("Finde deine perfekte Empfehlung!")

name = input("\nWie heißt du? ")
print(f"Hallo {name}! Was magst du?")
favorite1 = input("Lieblingsthema 1: ")
favorite2 = input("Lieblingsthema 2: ")

user_data = {
    name: [favorite1, favorite2],
    "Lena": ["Sport", "Gaming"]
}

def recommendation(user_name):
    if favorite1 in ["Musik", "musik", "Music", "music"]:
        return "🎵 DEIN TIPP: Top 100 Charts Mix!"
    elif favorite1 in ["Sport", "sport", "Sports", "sports"]:
        return "⚽ DEIN TIPP: Beste Tor-Kompilation!"
    else:
        return "🎬 DEIN TIPP: Trendige Videos!"

print(f"\n{recommendation(____)}")
```

Lösungsschnipsel: name

Hinweis: Welche Variable enthält den Namen?

Station 9: Wettervorhersage – Random Forest

Aylin zeigt auf ihre Uhr.

AYLIN: „Woher weißt du eigentlich immer, ob wir draußen spielen können?“

ALARA: „Das ist mein Expertenteam! Drei schlaue Köpfe in mir beraten sich – genau wie ein Random Forest!“

AYLIN: „Wow! Du hast also tatsächlich drei verschiedene Expertinnen in dir?“

ALARA: „Ja! Jede Expertin (jeder Baum) sieht nur einen zufälligen Teil der Daten und Merkmale. Erst durch die demokratische Abstimmung aller Bäume entsteht die präzise Vorhersage.“

ALARA: „Mit jeder Vorhersage werde ich genauer“, sagt Alara. „Dein Feedback hilft mir, meine Expertinnen zu verbessern.“

Was ist das Besondere am Random Forest: In Wirklichkeit trainiert ein Random Forest viele Bäume mit zufällig ausgewählten Daten und Merkmalen. Unsere Abstimmung simuliert dieses Prinzip der „kollektiven Intelligenz“.

Station 9: Wettervorhersage – Random Forest

BASIC

CODE:

```
print("AYLINS WETTER-EXPERTEN")
print("Drei Experten beraten!")

temperature = float(input("Temperatur: "))
clouds = input("Wolken (sonnig/bewölkt): ")

# Die Experten befragen
expert1 = "Sonne" if temperature > 20 else "Regen"
expert2 = "Sonne" if clouds == "sonnig" else "Regen"

opinions = [expert1, expert2]
decision = max(set(opinions), key=opinions.____)

print(f"Vorhersage: {decision}")
```

Lösungsschnipsel: count

Hinweis: Zähle, welche Meinung am häufigsten vorkommt.

Station 9: Wettervorhersage – Random Forest

ADVANCED

CODE:

```
print("AYLINS WETTER-KOMITEE")
print("Drei KI-Experten entscheiden!")

temperature = float(input("Temperatur: "))
clouds = input("Wolken (sonnig/bewölkt/regnerisch): ")
wind = input("Wind (stark/schwach): ")

# Drei Experten geben ihre Meinungen ab
expert1 = "Sonne" if temperature > 20 else "Regen"
expert2 = "Sonne" if ____ == "sonnig" else "Regen"
expert3 = "Sonne" if wind == "schwach" else "Regen"

opinions = [expert1, expert2, expert3]
decision = max(set(opinions), key=opinions.count)

print(f"Vorhersage: {decision}")
```

Hinweis: Wie heißt die Variable für die Wolken?

Lösungsschnipsel: clouds

Epilog – Das zweite Geschenk

Ein paar Monate später.

Aylin sitzt an ihrem Schreibtisch, Alara leuchtet an ihrem Handgelenk. In der Küche klappern Teller – Tante Sema ist zu Besuch und bereitet Börek zu.

AYLIN: „Weißt du, Alara“, sagt Aylin lächelnd, „als du mir an meinem Geburtstag geschenkt wurdest, hatte ich keine Ahnung von KI. Und jetzt … baue ich mein allererstes kleines Projekt und verstehe, wie du ‚denkst‘.“

Alara blinkt stolz. **ALARA:** „Du hast nicht nur KI verstanden – du hast mir auch beigebracht, menschlicher zu ‚denken‘.“

In diesem Moment kommt Tante Sema mit einem kleinen eingepackten Geschenk herein.

TANTE SEMA: „Canım“ (meine Süße), sagt sie sanft, „ich habe gesehen, wie viel du in den letzten Monaten gelernt hast. Deine Oma würde sagen: Du hast der Uhr eine Seele gegeben.“

Aylin öffnet das Päckchen. Darin ist ein schlichtes Notizbuch. Auf der ersten Seite steht in schwungvoller Schrift:

„Erfinde mit Herz. Programmiere mit Mut.
Denke wie Alara – aber fühle wie Aylin.“

Aylin lächelt.

AYLIN: „Das ist das schönste Geschenk, Tante Sema.“

TANTE SEMA: „Nein“, sagt Sema und tippt ihr leicht auf die Brust. „Das schönste Geschenk ist genau hier.“ **ENDE**

REFLEXIONSFRAGEN

Nimm dir einen Moment Zeit, um über das nachzudenken, was du in diesem Modul gelernt hast. Beantworte die folgenden Fragen in Ruhe.

- **KI verstehen:** Wie hat sich dein Verständnis von Künstlicher Intelligenz nach diesem Modul verändert? Was hat dich am meisten überrascht, wie KI-Systeme wie Alara „denken“?

- **Neuronale Netze & Entscheidungsfindung:** In Station 2 hast du erkundet, wie neuronale Netze Gewichte nutzen, um Entscheidungen zu treffen. Fällt dir eine Situation im echten Leben ein, in der du verschiedene Faktoren abwägst, bevor du eine Wahl triffst? Was ist ähnlich daran, wie KI Entscheidungen trifft?

- **Ethische Überlegungen:** Im gesamten Modul wurden ethische Fragen rund um KI aufgeworfen. Warum ist es wichtig, beim Entwerfen oder Nutzen von KI über Ethik nachzudenken? Nenne ein Beispiel für eine ethische Herausforderung im Zusammenhang mit KI im Alltag.

- **Lernen durch Interaktion:** Alara lernt aus Aylins Interaktionen. Wie lernen KI-Systeme im echten Leben (wie Empfehlungsalgorithmen oder Sprachassistenten) deiner Meinung nach aus Nutzerinteraktionen? Was sind die Vorteile und Risiken dieser Art des Lernens?

- **KI in deiner Welt:** Wo begegnest du KI in deinem Alltag? Wähl ein Beispiel (z. B. soziale Medien, Streaming-Dienste, smarte Geräte) und erkläre, wie es eines der KI-Konzepte nutzt, die du in diesem Modul kennengelernt hast.

QUIZ MIT MEHRFACHWAHLFRAGEN
(eine richtige Antwort pro Frage)

1. Was ist Alara?
a) Aylins beste Freundin
b) Ein gewöhnlicher Wecker
c) Eine KI in einer Smartwatch
d) Aylins Tante

2. Welches Programmierkonzept wird in Station 2 eingeführt, um eine Entscheidung auf der Grundlage mehrerer gewichteter Eingaben zu treffen?
a) Entscheidungsbäume
b) Verstärkendes Lernen
c) Neuronale Netze
d) Regelbasierte Klassifizierung

3. Bei Station 3, „Der Schulwegdetektiv“, geht es hauptsächlich um...
a) Musikempfehlungen
b) Routenplanung mit Entscheidungsbäumen
c) Bilderkennung
d) Sprachverarbeitung

4. In Station 4: Wie klassifiziert Alara einen Song als „Pop“?
a) Wenn er eine Gitarre enthält
b) Wenn er laut ist
c) Wenn er tanzbar ist
d) Wenn er von Taylor Swift ist

5. Was simuliert die Übung „Pflanzenpflege“ in Station 5?
a) Bilderkennung
b) Verstärkendes Lernen durch Belohnung
c) Neuronale Netze
d) Entscheidungsbäume

QUIZ MIT MEHRFACHWAHLFRAGEN (eine richtige Antwort pro Frage)

6. In Station 6: Wie erkennt Alara Kanten in einem Bild?
a) Durch den Vergleich benachbarter Pixel auf Unterschiede
b) Durch das Zählen der weißen Pixel
c) Durch Sprachbefehle
d) Durch Nachfragen bei der Nutzerin/dem Nutzer

7. In Station 7 („Verarbeitung natürlicher Sprache“) wird die Stimmung eines Textes erkannt durch...
a) Messen der Textlänge
b) Suchen nach bestimmten Schlüsselwörtern
c) Überprüfen der Grammatik
d) Analysieren der Schriftart

8. Was macht das Empfehlungssystem in Station 8?
a) Es erstellt Wettervorhersagen
b) Es schlägt Videos basierend auf Nutzerinteressen vor
c) Es erkennt Handschrift
d) Es führt einen Roboter durch ein Labyrinth

9. Wie trifft der „Random Forest“ in Station 9 eine Entscheidung?
a) Nur eine Expertin/ein Experte entscheidet
b) Durch eine demokratische Abstimmung mehrerer „Bäume“
c) Durch zufällige Auswahl
d) Durch tiefe neuronale Netze

10. Welche Programmiersprache wird in allen Programmieraufgaben verwendet?
a) JavaScript
b) Python
c) N1GL1B
d) C++

Modul 2: AI Escape Room Schatzsuche mit Taschenrechner

EINLEITUNG

Hast du dich schon einmal gefragt, woher dein Smartphone weiß, welche Videos dir gefallen könnten? Ziel dieses Moduls ist es, die grundlegenden Konzepte vorzustellen, die erklären, wie Künstliche Intelligenz (KI) „denkt“, Entscheidungen trifft und aus Daten lernt. Die Inhalte bieten einen einfachen theoretischen Hintergrund, der Mathematik, Logik und Kreativität durch ein spannendes, spielerisches Lernerlebens miteinander verbindet.

Am Ende dieses Moduls wirst du dir verschiedene Fähigkeiten angeeignet haben, darunter:

- **Mathematische Kompetenz:** Du verbesserst deine Fähigkeiten in den Grundrechenarten, lernst Durchschnitte, Prozentsätze und Verhältnisse zu verstehen und gewinnst Sicherheit im Umgang mit Taschenrechnern zur Problemlösung.
- **Kritisches Denken und logisches Argumentieren:** Du übst logisches Denken, indem du Hinweise entschlüsselst und auf Basis gegebener Daten Entscheidungen triffst. Deine Problemlösungsfähigkeiten werden gestärkt, indem du Mathematik auf praktische Szenarien und Rätsel anwendest.
- **Entscheidungsfindung:** Beim Lösen von Aufgaben zur Klassifizierung (z. B. entscheiden, welches Objekt zu einer bestimmten Beschreibung passt) übst du, fundierte Entscheidungen auf der Grundlage von Zahlen und Logik zu treffen.

Moduldauer	**2 Stunden** **(Theorie: 1 Std. | Praxis: 1 Std.)**

UNTERRICHTSMATERIALIEN

EINHEIT 2.1: WAS STECKT HINTER DEM KI-ESCAPE-ROOM?

Künstliche Intelligenz kann als die Fähigkeit von Maschinen definiert werden, Aufgaben auszuführen, die normalerweise menschliche Intelligenz erfordern – wie logisches Denken, Problemlösung, Lernen oder das Erkennen von Mustern. Mathematik steht dabei im Kern der KI: Wahrscheinlichkeitsrechnung, Statistik und Logik helfen Computern, Entscheidungen auf der Grundlage von Daten zu treffen – nicht basierend auf Intuition. Daten wiederum sind der „Treibstoff" der KI: Je mehr qualitativ hochwertige Daten zur Verfügung stehen, desto besser kann das System Muster erkennen und Vorhersagen treffen. In diesem Modul wirst du entdecken, wie KI logisches Denken, Wahrscheinlichkeiten und Lernen nutzt, um reale Probleme auf kreative Weise zu lösen. Um diese Ideen zu verstehen, werden wir vier Schlüsselprinzipien erkunden, die zeigen, wie KI „denkt":

- **Bayes'sche Netze** – Sie helfen dabei, Ergebnisse vorherzusagen, wenn Informationen unsicher sind.
- **K-Means-Clustering** – Diese Methode gruppiert ähnliche Daten, um Muster und Zusammenhänge zu finden.
- **Monte-Carlo-Methoden** – Sie nutzen Zufallsexperimente, um Wahrscheinlichkeiten abzuschätzen und Vorhersagen zu treffen.
- **Hebb'sches Lernen** – Es zeigt, wie Verbindungen durch Wiederholung stärker werden – genau wie in unserem Gehirn, wenn wir etwas Neues lernen.

EINHEIT 2.2: DIE VIER PRINZIPIEN DER KI

Durch diese einfachen Konzepte wirst du verstehen, dass es bei KI nicht nur ums Programmieren geht – sondern auch um logisches Denken, Problemlösung und Kreativität. Diese Prinzipien begleiten dich durch jede Etappe des KI-Escape-Rooms, wo jede Herausforderung zeigt, wie Künstliche Intelligenz „durch Handeln lernt".

Diese vier Ideen zeigen, wie KI Logik, Mathematik und Kreativität miteinander verbindet, um zu lernen und kluge Entscheidungen zu treffen – genau das wirst du tun, während du die Rätsel in diesem KI-Escape-Room-Abenteuer löst!

**„Das Ziel der KI ist nicht, Menschen zu ersetzen,
sondern menschliche Fähigkeiten zu erweitern."**
— Fei-Fei Li (KI-Forscherin)

REFERENZEN

Künstliche Intelligenz & Entscheidungsfindung

- Russell, S., & Norvig, P. (2021). Artificial Intelligence: A Modern Approach (4th ed.). Pearson.

Wahrscheinlichkeit, Simulation und Lernen

- Shalev-Shwartz, S., & Ben-David, S. (2014). Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press.
  → Grundlagen des maschinellen Lernens, Clustering und Lernen aus Daten.

- Khan Academy. (n.d.). Statistics and probability. https://www.khanacademy.org/math/statistics-probability
  → Einfache und zugängliche Erklärungen zu Wahrscheinlichkeit, Durchschnitten und Simulationen.

LINKS ZU LERNPLATTFORMEN

Wenn du mehr über Künstliche Intelligenz auf unterhaltsame und einfache Weise lernen möchtest, sind diese Online-Plattformen ein großartiger Einstieg! Du kannst spielen, experimentieren und entdecken, wie Computer lernen, Bilder, Geräusche oder Muster zu erkennen – genau wie im KI-Escape-Room. Jeder Link enthält kurze Aktivitäten, die dir helfen zu verstehen, wie KI im echten Leben funktioniert – und du lernst weiterhin durch Ausprobieren.

- Google Teachable Machine – Interaktive Plattform zum Trainieren einfacher KI-Modelle mit Bildern, Geräuschen oder Posen. https://teachablemachine.withgoogle.com/
- Khan Academy – Künstliche Intelligenz und maschinelles Lernen – Kostenlose Lektionen, die grundlegende KI-Konzepte anhand alltäglicher Beispiele erklären. https://www.khanacademy.org/computing/computer-science
- KI für die Ozeane (Code.org) – Lustige interaktive Aktivität, bei der Schüler*innen eine KI trainieren, um den Ozean zu säubern – und dabei etwas über Daten und Verzerrungen (Bias) lernen. https://studio.code.org/s/oceans
- Machine Learning for Kids – Plattform, auf der Schüler*innen mit KI-Modellen experimentieren und sie in Scratch-Projekten nutzen können. https://machinelearningforkids.co.uk/
- IBM SkillsBuild für Schüler*innen – Kostenlose Online-Lernplattform mit grundlegenden KI- und Datenkompetenzkursen inklusive Abzeichen und Zertifikaten. https://skillsbuild.org/students
- Google AI Experiments – Eine Sammlung interaktiver Tools, die zeigen, wie KI Muster, Geräusche und Bewegungen erkennt. https://experiments.withgoogle.com/collection/ai

PRAKTISCHE ÜBUNG

Willkommen im KI-Escape-Room!

Du bist Teil eines Teams junger Entdecker*innen, die einen Schatz suchen – gesichert hinter einem 4-stelligen Zahlenschloss. Um es zu öffnen, musst du dich durch vier Räume kämpfen, die von vier Wächter*innen bewacht werden. Jede und jeder von ihnen steht für ein Prinzip der Künstlichen Intelligenz:

1. Wahrscheinlichkeit (Bayes'sche Netze)
2. Clustering (K-Means-Clustering)
3. Simulation (Monte-Carlo-Methoden)
4. Hebb'sches Lernen

Deine Mission ist es, die Herausforderungen an jeder Station zu meistern, ihre Probleme zu lösen und so die Kombination zu erhalten, die das Schloss öffnet. Jede Station enthält eine Erklärung, dazwischen kurze Übungen und am Ende eine Programmieraufgabe in drei Schwierigkeitsstufen. Diese Aufgabe liefert dir eine der Ziffern, die du für das Zahlenschloss brauchst.

Station 1 – Bayes'sche Netze

Ein Bayes'sches Netz ist ein mathematisches Modell, das Wahrscheinlichkeiten nutzt, um Entscheidungen / Vorhersagen zu treffen, besonders dann, wenn nicht alle Informationen mit Sicherheit bekannt sind, also wenn Unsicherheit herrscht.

Mini-Challenge 1 – Das einfache Schloss

Stell dir vor, du möchtest in einem Escape-Room ein Zahlenschloss öffnen.

- **Ereignis A – Spuren:** Es gibt mehrere versteckte Hinweise. Je mehr du findest, desto einfacher wird es, das Schloss zu öffnen.
- **Ereignis B – Rätsel:** Es gibt ein Rätsel, das die Zahlen für die Kombination liefert. Je mehr Hinweise du findest, desto einfacher ist es, das Rätsel zu lösen.
- **Ereignis C – Schloss:** Ob du das Rätsel lösen und die Zahlen in das Schloss eingeben kannst, hängt davon ab, wie viele Hinweise du gefunden hast.

Mini-Challenge 1 — The Simple Lock

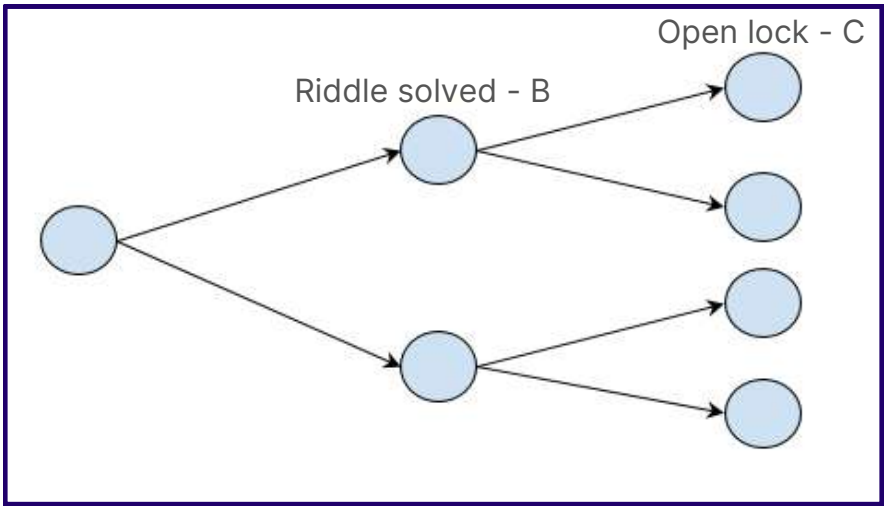


Abb. 2 Decision and Probability Path: From Finding Clues to Opening the Lock.
Source: Monica Moreno

Hinweis: Die Hälfte des Diagramms fehlt – sie entspricht der Wahrscheinlichkeit, keine Hinweise zu finden.

Aufgabe: Wenn du die Hinweise findest (Wahrscheinlichkeit 0,8) und das Rätsel löst (Wahrscheinlichkeit 0,9, falls du die Hinweise gefunden hast), wie groß ist dann die Wahrscheinlichkeit, dass du das Schloss öffnest (Wahrscheinlichkeit 0,95, falls du das Rätsel gelöst hast)?

Der Weg zur Lösung setzt sich aus mehreren Wahrscheinlichkeiten zusammen, die miteinander multipliziert werden müssen.

Antwort: $P(\text{Open Lock}) = 0.8 \times 0.9 \times 0.95$

Frage: Welches Ergebnis würdest du erhalten, wenn du alle möglichen Pfade zusammenzählen würdest?

Antwort: 1, denn jeder Pfad repräsentiert die Wahrscheinlichkeit eines möglichen Szenarios. Die Wahrscheinlichkeit, dass eines aller möglichen Ergebnisse eintritt, ist immer 1.

Bayes'sche Regel

Bisher haben wir gesehen, wie wir eine Vorhersage treffen können. Aber was passiert, wenn wir das Ergebnis betrachten und wissen möchten, mit welcher Wahrscheinlichkeit ein bestimmter Pfad eingeschlagen wurde?

Mini-Challenge 2 – Der bedingte Korridor

Du entdeckst einen geheimen Korridor mit einem weiteren Rätsel an der Wand: „Von allen Malen, in denen du das Rätsel gelöst hast, wie groß ist die Wahrscheinlichkeit, dass du zuerst die Hinweise gefunden hast?“

Mit anderen Worten: Wir möchten die Wahrscheinlichkeit von C unter der Bedingung B berechnen (was sich von der Wahrscheinlichkeit von B unter der Bedingung C unterscheidet). Du musst die Bayes'sche Regel anwenden:

Anleitung: Zuerst müssen wir die Wahrscheinlichkeiten für B = ✓ und C = ✓ berechnen, also alle Pfade, die zu B = ✓ führen, und alle, die zu C = ✓ führen. Zur Vereinfachung geben wir dir die bereits berechneten Werte:
P(B = ✓) = 0,76; P(C = ✓) = 0,734

$$P(B=\checkmark \text{ if } C=\checkmark) = [P(C=\checkmark \text{ if } B=\checkmark) * P(B=\checkmark)] / P(C=\checkmark)$$

Antwort: $P(B=\checkmark \mid C=\checkmark) = [0.95 * 0.76] / 0.724 = 0.983$

Challenge 1 – Die Wächter*in der Wahrscheinlichkeit

Du hast den ersten Raum erreicht – die Kammer der Hüterin der Wahrscheinlichkeit, eine geheimnisvolle Halle, erleuchtet von digitalen Fackeln. In der Mitte steht eine alte Truhe, bedeckt mit vielen Edelsteinen. Da spricht die Wächterin, ein Hologramm einer alten Frau, zu dir:

„Nur wer die Unsicherheit versteht, wird entdecken können, ob die Truhe Gold enthält. Die Wahrscheinlichkeit ist der Schlüssel.“

Deine Mission ist es, die Wahrscheinlichkeit zu berechnen, dass die Truhe Gold enthält – basierend auf dem, was du sehen kannst, und mithilfe eines vereinfachten Bayes'schen Netzwerkmodells.

***Szenario:** Wie groß ist die Wahrscheinlichkeit, dass diese schwere, glänzende Truhe Gold enthält? Die erste Dezimalstelle der Antwort ergibt die erste Ziffer des Zahlenschlosses. Du weißt:*

- 30 % der glänzenden Truhen enthalten Gold, 70 % enthalten Steine.
- Die Truhen mit Gold sind in 80 % der Fälle schwer.
- Die Truhen mit Steinen sind in 40 % der Fälle schwer.

Challenge 1 – Die Wächterin der Wahrscheinlichkeit

EINFACHES LEVEL: Zeichne das Bayes'sche Netzwerk auf Papier und nutze den Taschenrechner, um die Antwort zu erhalten. **Hinweis:** Du brauchst nur zwei Ereignisse (A und B) und die Bayes'sche Regel.

MITTELSCHWERES LEVEL: Passe diesen unvollständigen Code an, um das Problem zu lösen. Füge einfach die Wahrscheinlichkeiten und mathematischen Symbole ein.

BASIC

CODE:

```
p_gold =
p_stones =
p_heavy_if_gold =
p_heavy_if_stones =

# Berechne die Wahrscheinlichkeit, dass die Truhe schwer ist
p_heavy = (p_gold "operator" p_heavy_if_gold) + (p_stones
"operator" p_heavy_if_stones)

# Berechne die Wahrscheinlichkeit für Gold, unter der
Bedingung dass die Truhe schwer ist
p_gold_given_heavy = (p_gold "operator" p_heavy_if_gold)
"operator" p_heavy

print("Wahrscheinlichkeit für Gold, wenn schwer:",
round(p_gold_given_heavy, 2))
```

SCHWERES LEVEL: Zeichne das Bayes'sche Netzwerk und erstelle deinen eigenen Code, um das Problem zu lösen.

Station 2 – K-Means-Clustering

K-Means ist ein Algorithmus, der Daten basierend auf Ähnlichkeit in Gruppen (Cluster) einteilt. Um das zu verstehen, schauen wir uns ein Beispiel an: Stell dir vor, du hast fast jeden Tag im Jahr die Temperatur gemessen und möchtest nun wissen, zu welcher Jahreszeit jeder Messwert gehört. Das nennt man ein Klassifikationsproblem.

Mini-Challenge 1 – Die Kristalle im Gang

Auf dem Weg zum nächsten Raum findest du mehrere Kristalle auf dem Boden. Jeder Kristall hat zwei Zahlen: Helligkeit und Reinheit. Gruppiere diese Kristalle anhand ihrer Eigenschaften in zwei verschiedene Gruppen.

Anleitung:

- Zeichne zwei Achsen (x, y) und markiere folgende Punkte:
  [1,5], [4,1], [3.5,0], [2,4], [0.5, 5], [0.5, 5], [5, 2].
- Stell dir vor, du möchtest zwei Gruppen (k = 2).
  Wo würdest du die anfänglichen Mittelpunkte (Zentroide) der Gruppen setzen?
- Ordne jeden Kristall dem nächstgelegenen Zentroid zu.

Antwort: Gruppe1: [1,5], [2,4], [0.5, 5], [0.5, 5]. Gruppe2: [4,1], [3.5,0], [5, 2]

K-Means automatisiert den Prozess der Gruppenbildung, sodass jede Gruppe einen „Mittelpunkt“ (Zentroid) hat, der den Durchschnitt der Eigenschaften dieser Gruppe darstellt. So funktioniert K-Means:

1. Wähle die Anzahl der Cluster (k = n).
2. Setze k Zentroide zufällig auf der Ebene – einen für jede Gruppe.
3. Ordne jeden Punkt dem nächstgelegenen Zentroid zu.
4. Berechne den Mittelpunkt jeder Gruppe neu und verschiebe den Zentroid dorthin.
5. Wiederhole den Vorgang, bis sich die Punkte nicht mehr zwischen den Gruppen verändern.

Challenge 2 - The Alchemist

Sobald du den ersten Raum passiert hast, erreichst du ein Labor, gefüllt mit vielen Kolben voller geheimnisvoll gefärbter Flüssigkeiten. Da spricht eine alte Alchemistin zu dir:

„Junge Abenteurerin, junger Abenteurer, du wirst hier nicht hinauskommen, ohne meine Tränke zu sortieren. Jede Flasche gehört zu einer anderen Art von Magie: Heilen, Stärke oder Unsichtbarkeit. Wenn du mir hilfst, sie richtig zu gruppieren, öffne ich dir die Tür."

Jeder Kolben hat zwei chemische Eigenschaften: pH-Wert und Konzentration der magischen Energie. Deine Mission ist es, K-Means zu nutzen, um die Kolben zu gruppieren und so die zweite Prüfung zu bestehen. Die Anzahl der Cluster, die du erstellst, entspricht der zweiten Ziffer des Zahlenschlosses.

EINFACHES LEVEL: Nutze den folgenden Code, um die Kolben zu gruppieren, und fülle die fehlenden Parameter aus. Zusatz: Beobachte, wie sich das Ergebnis verändert, wenn du die Anzahl der Gruppen variierst.

Hinweis: Du musst zuerst die sklearn-Bibliothek installieren:
pip install scikit-learn

MITTELSCHWERES LEVEL: Nutze den Code aus dem einfachen Level und füge einen neuen Kolben (Punkt) mit den Koordinaten deiner Wahl hinzu. Verwende die Funktion kmeans.predict(point), um die Klassifizierung zu erhalten. Füge am Ende des Codes folgenden Befehl ein, um das Ergebnis anzuzeigen.

SCHWERES LEVEL: Erstelle den Code von Grund auf selbst.

Challenge 2 – Die Alchemist*in

BASIC

CODE:

```
# K-MEANS-BIBLIOTHEK
from sklearn.cluster import KMeans
import numpy as np
import matplotlib.pyplot as plt

# Daten: [pH-Wert, magische Energie]
potions = np.array([
    [2, 53], [6, 70], [3, 55], [3.1, 52], [10, 25],
    [8, 84], [9, 80], [7.5, 82],
    [10, 19], [10.5, 22], [4, 48], [9, 20],
])

# K-Means
kmeans = KMeans(n_clusters=2, random_state=0)
# Algorithmus initialisieren
# Führt K-Means aus, um die Daten zu klassifizieren
kmeans.fit(potions)

# Visualisierung
plt.scatter(potions[:, 0], potions[:, 1], c=kmeans.labels_,
cmap="viridis", marker="o")
plt.xlabel("pH-Wert")
plt.ylabel("Magische Energie")
plt.title("Klassifizierung der Tränke mit K-Means")
plt.show()

print("Klassifizierung der einzelnen Kolben:",
kmeans.labels_)
```

Challenge 2 – Die Alchemist*in

MEDIUM

CODE:

```
plt.scatter(potions[:, 0], potions[:, 1], c=kmeans.labels_,
cmap="viridis", marker="o")
plt.scatter(
   new_jar[:, 0], new_jar[:, 1],
   c=[plt.cm.viridis(prediction[0] / (kmeans.n_clusters -
1))],
   marker="x", s=150, label="Neuer Kolben"
)
plt.xlabel("pH-Wert")
plt.ylabel("Magische Energie")
plt.title("Klassifizierung des besonderen Kolbens")
plt.legend()
plt.show()
```

Challenge 2 – Die Alchemist*in

HARD

CODE:

```
from sklearn.cluster import KMeans
import numpy as np
import matplotlib.pyplot as plt

# Schritt 1: Definiere die Daten (Liste von Paaren [pH-Wert,
Energie] mit numpy)
potions = np.array([
    # Füge hier deine Daten ein
])

# Schritt 2: Erstelle das KMeans-Modell und lege die Anzahl
der Cluster fest
kmeans = KMeans(n_clusters=2, random_state=0)

# Schritt 3: Trainiere das Modell mit deinen Daten (fit)
kmeans.fit(potions)

# Visualisierung
plt.scatter(potions[:, 0], potions[:, 1], c=kmeans.labels_,
cmap="viridis", marker="o")
plt.xlabel("pH-Wert")
plt.ylabel("Magische Energie")
plt.title("Klassifizierung der Tränke mit K-Means")
plt.show()

# Schritt 4: Gib das Ergebnis aus
print("Klassifizierung der einzelnen Kolben:",
kmeans.labels_)
```

Station 3 – Monte-Carlo-Methoden

Monte-Carlo-Methoden sind ein Weg, um die Wahrscheinlichkeit vorherzusagen, mit der ein Ereignis eintreten wird. Dazu wird dasselbe Experiment viele Male simuliert und gezählt, wie oft dieses Ereignis als Ergebnis eintritt.

Mini-Challenge 1 – Der Brunnen des Glücks

Nachdem du eine Weile gelaufen bist, ohne den nächsten Raum zu finden, beginnst du dich zu verlaufen. Du erreichst einen kleinen Raum mit einem magischen Brunnen und beschließt, dir etwas zu wünschen, um den Weg zu finden.

Die Inschrift auf dem Stein lautet:

> „Wenn du eine Münze wirfst, beträgt die Wahrscheinlichkeit 50 %, dass dein Wunsch in Erfüllung geht. Aber was passiert, wenn du es viele Male versuchst?“

Anleitung

Nutze den Taschenrechner, um zehn Zufallszahlen zu generieren. Gerade Zahlen stehen für „Kopf“, ungerade Zahlen für „Zahl“. Zähle, wie oft die Münze „Kopf“ zeigt, und teile diese Zahl durch die Gesamtanzahl der Würfe (N = 10). Das Ergebnis ist die Wahrscheinlichkeit.

Bei Monte-Carlo-Methoden muss in den Experimenten immer Zufall stecken. Ohne Zufall gäbe es keine Variation in den Ergebnissen – das bedeutet, wir könnten keine Wahrscheinlichkeiten schätzen.

Challenge 3 – Die Hüter*in der Schicksalswürfel

Nach langer Wanderung durch einen steinernen Korridor, der von Fackeln erleuchtet wird, entdeckst du einen Raum mit einem runden Tisch, bedeckt mit Schriftrollen, bunten Steinen und einigen Daten.

Hinter dem Tisch sitzt die Hüterin des Zufalls – ein Skelett mit Kapuze und tiefer Stimme, das drei Würfel nimmt und zu dir spricht:

„Wanderer, dieser Raum öffnet sich nicht mit Gewalt oder Klugheit … Ich bin die Hüterin der unendlichen Monte-Carlo-Spiele und stelle dir eine Herausforderung. Ich werde diese drei Würfel werfen und die Augenzahlen zusammenzählen. Wähle: mehr als 12 oder weniger als/gleich 12. Wenn du richtig liegst, darfst du weitergehen – wenn nicht, bleibst du für immer hier gefangen.“

Um zu entscheiden, welche Option du wählen sollst, wendest du dein Wissen über Wahrscheinlichkeiten an und nutzt Monte-Carlo-Methoden, um deine beste Option zu bestimmen. Die erste Dezimalstelle der größeren Wahrscheinlichkeit ergibt die dritte Ziffer des Zahlenschlosses.

EINFACHES LEVEL: Folge den Schritten unten, um einen Code zu erstellen, der die Erfolgswahrscheinlichkeit für jeden Fall mit N = 10.000 Experimenten berechnet.

1. Erstelle zwei Variablen, um zu zählen, wie oft jedes Ergebnis vorkommt.
   Hinweis: Du musst die Variablen mit 0 initialisieren.

2. Generiere drei Zufallszahlen und addiere sie.
   Hinweis: Verwende die Funktion random.random().

3. Überprüfe, ob die Summe >12 oder ≤12 ist und erhöhe den entsprechenden Zähler um 1.

4. Wiederhole den Vorgang 10.000 Mal.
   Hinweis: Setze den Code in eine for-Schleife.

5. Teile die Zähler durch N = 10.000, um die Wahrscheinlichkeiten zu erhalten, und wähle die wahrscheinlichste Option.

Challenge 3 – Die Hüter*in der Schicksalswürfel

MITTELSCHWERES LEVEL: Schreibe den Code, um das Experiment mit N = 10.000 Wiederholungen zu simulieren (du kannst die Schritte aus dem einfachen Level verwenden). Zusätzlich musst du die Wahrscheinlichkeit für jedes mögliche Ergebnis (von 3 bis 18) beim Würfeln berechnen. Hinweis: Verwende ein Dictionary für die Zähler. Führe alle Experimente durch und addiere die Ergebnisse zu den jeweiligen Zählern. Teile am Ende alle Werte durch N.

SCHWERES LEVEL: Erstelle ein Programm von Grund auf, das für eine beliebige Anzahl von Würfeln, Wiederholungen und einen Schwellenwert die Wahrscheinlichkeit jedes möglichen Ergebnisses berechnet. Hinweis: Erstelle eine Funktion mit den Parametern N, num_dices und threshold, die ein Dictionary mit den möglichen Ergebnissen als Schlüssel und den Wahrscheinlichkeiten als Werten zurückgibt.

EXTRA: Füge eine Visualisierung der Wahrscheinlichkeitsverteilung hinzu, die die Wahrscheinlichkeiten der einzelnen Werte in einem Histogramm (plt.bar) anzeigt.

Station 4 – Hebb'sches Lernen

Das Hebb'sche Lernen ist eine Lernregel für neuronale Netze, die auf dem Satz basiert: „Neurons that fire together, wire together." – Zu Deutsch: „Neuronen, die gemeinsam feuern, verbinden sich." Mit anderen Worten:

- Wenn zwei Neuronen gleichzeitig aktiviert werden, verstärkt sich die Verbindung zwischen ihnen.
- Wenn eines aktiviert wird und das andere nicht, schwächt sich die Verbindung.

Mini-Challenge 1 – Die Bibliothek der Ideen

Auf dem Weg zum letzten Raum durchquerst du einen dunklen Gang, in dem zwei Statuen mit leuchtenden Symbolen stehen. Jedes Mal, wenn du „Licht" sagst, leuchten beide Statuen gleichzeitig auf. Wenn du aber „Schatten" sagst, leuchtet nur eine. Laut der Hebb'schen Regel: Was passiert mit der Verbindung zwischen diesen beiden Statuen, wenn du oft „Licht" wiederholst?

a) Sie wird schwächer
b) Sie bleibt gleich
c) Sie wird stärker

Mini-Challenge 1 – Die Bibliothek der Ideen

Um das Verbindungsgewicht zwischen zwei Neuronen zu berechnen, nachdem sie gefeuert haben, verwendet man die Hebb'sche Formel (vereinfachtes Model):

$$w_{final} = w_{initial} + (n_{together} \times \Delta w_{+}) + (n_{alone} \times \Delta w_{-})$$

Dabei gilt:

- $w_{initial}$ = Anfangsgewicht der Verbindung
- $n_{together}$ = Anzahl der Male, bei denen beide Neuronen gleichzeitig feuern
- Δw_{+} = Gewichtszunahme bei gemeinsamer Aktivierung
- n_{Alone} = Anzahl der Male, bei denen nur ein Neuron feuert
- Δw_{-} = Gewichtsabnahme, wenn nur eines aktiviert ist

Mini-Challenge 2 – Verbindungsgewichte

Um den Gang richtig zu beleuchten, musst du die Aktivierung zwischen den beiden Statuen (oder Neuronen) berechnen. Dazu verwendest du die Hebb'sche Formel.

Anleitung:

- Das Anfangsgewicht der Verbindung beträgt 0,2.
- Jedes Mal, wenn beide gleichzeitig leuchten → +0,1
- Jedes Mal, wenn nur eines leuchtet → –0,05
- Beide leuchten 3 Mal zusammen, aber Statue A leuchtet 2 Mal allein.

Antwort: $w_{final} = 0.2 + (3 \times 0.1) + (2 \times 0.05) = 0.6$

Challenge 4 – Das Neuronenfresko

Du bist ganz nah am Ziel, aber du musst noch eine letzte Herausforderung meistern. Im letzten Raum entdeckst du ein Fresko mit drei magischen Lichtern (A, B, C), die durch Fäden miteinander verbunden sind – diese Fäden stellen die Verbindungsgewichte dar. Eine geheimnisvolle Stimme spricht zu dir:

„Um die letzte Tür zu öffnen, musst du herausfinden, welche Verbindungen verstärkt wurden. Berechne die endgültigen Gewichte nach mehreren Aktivierungen und enthülle das geheime Muster."

Jede Verbindung beginnt mit dem Gewicht 0. Wenn zwei Neuronen gleichzeitig feuern, erhöht sich die Verbindung um 0,2. Wenn nur eines feuert, verringert sich das Gewicht um –0,07. Es gibt drei Verbindungen: AB, AC und BC. Die letzte Ziffer des Zahlenschlosses entspricht der letzten Dezimalstelle des Gewichts der Verbindung AC nach Runde 4.

Runde	Aktivierte Neuronen
1	A,B
2	B,C
3	A
4	A, C
5	A, B, C

EINFACHES LEVEL: Berechne nur die Verbindung AB und erstelle eine Tabelle, die das Verbindungsgewicht für jede Runde zeigt.

MITTELSCHWERES LEVEL: Berechne die endgültigen Gewichte aller Verbindungen über alle Runden hinweg.

SCHWERES LEVEL: Berechne die endgültigen Gewichte aller Verbindungen über alle Runden hinweg – mit folgender Zusatzregel: Wenn alle drei Neuronen gleichzeitig feuern, erhalten alle Verbindungen einen Bonus von +0,2. Gib an, welche Verbindung nach der letzten Runde das stärkste Gewicht hat.

REFLEXIONSFRAGEN

Nimm dir einen Moment Zeit, um über das nachzudenken, was du in diesem Modul gelernt hast. Beantworte die folgenden Fragen sorgfältig.

KI, Mathematik und Entscheidungsfindung: In diesem Modul hast du Mathematik, Wahrscheinlichkeitsrechnung und Logik genutzt, um die Herausforderungen im KI-Escape-Room zu lösen. Wie hat dir diese Aktivität geholfen zu verstehen, wie KI Entscheidungen trifft? Welche Herausforderung hat dir dabei am besten geholfen??

Lernen durch Wahrscheinlichkeit und Simulation: Du hast verschiedene KI-Methoden kennengelernt, darunter Bayes'sche Netze, K-Means-Clustering, Monte-Carlo-Methoden und Hebb'sches Lernen. Kannst du eine dieser Ideen mit eigenen Worten erklären und ein Beispiel nennen, wie sie im Escape-Room angewendet wurde?

Lernen durch Wiederholung und Erfahrung: Bei einigen Herausforderungen mussten Berechnungen, Simulationen oder Versuche wiederholt werden, um zur richtigen Antwort zu gelangen. Wie ähnelt dies der Art und Weise, wie KI-Systeme aus Daten und Erfahrung lernen? Fällt dir ein Beispiel aus dem echten Leben ein, bei dem KI auf diese Weise lernt?

QUIZ MIT MEHRFACHWAHLFRAGEN (eine richtige Antwort pro Frage)

1. Wobei helfen uns Bayes'sche Netze hauptsächlich?
a) Ähnliche Datenpunkte zu gruppieren
b) Entscheidungen unter Unsicherheit zu treffen
c) Berechnungen durch Zufallsstichproben zu beschleunigen
d) Neuronale Verbindungen zu verstärken

2. Wenn 30 % der glänzenden Truhen Gold enthalten und 80 % der Goldtruhen schwer sind – welche Regel verwenden wir, um die Wahrscheinlichkeit zu berechnen, dass eine schwere Truhe Gold enthält?
a) Monte-Carlo-Simulation
b) Hebb'sches Lernen
c) Bayes'sche Regel
d) K-Means-Algorithmus

3. Was repräsentiert das „k“ in K-Means?
a) Die Anzahl der Datenpunkte
b) Die Anzahl der Cluster (Gruppen)
c) Die Anzahl der Dimensionen
d) Die Anzahl der Wahrscheinlichkeiten

4. Welcher Schritt kommt direkt nach der Zuordnung der Punkte zum nächstgelegenen Zentroid?
a) Die Zentroide erneut zufällig setzen
b) Die neuen Mittelpunkte jedes Clusters berechnen
c) Ein Bayes'sches Netzwerk zeichnen
d) Wahrscheinlichkeiten mit Würfeln schätzen

5. Warum sind Monte-Carlo-Methoden für die Wahrscheinlichkeitsschätzung nützlich?
a) Sie eliminieren den Zufall vollständig
b) Sie simulieren viele Experimente mit Zufall
c) Sie gruppieren Daten in Kategorien
d) Sie verstärken Netzwerkverbindungen

QUIZ MIT MEHRFACHWAHLFRAGEN (eine richtige Antwort pro Frage)

6. Was wird in der Challenge „Die Hüterin der Schicksalswürfel“ berechnet?
a) Welche Truhe Gold enthält
b) Zu welchem Cluster ein Trank gehört
c) Die Wahrscheinlichkeit, ob die Summe der Würfel größer oder kleiner als 12 ist
d) Das endgültige Gewicht einer Neuronenverbindung

7. Laut dem Satz „Neurons that fire together, wire together“ – was passiert, wenn zwei Neuronen viele Male gleichzeitig aktiviert werden?
a) Die Verbindung wird schwächer
b) Die Verbindung verschwindet
c) Die Verbindung wird stärker
d) Es ändert sich nichts

8. Wenn zwei Neuronen 3 Mal zusammen feuern (+0,1 pro Mal) und nur eines 2 Mal allein feuert (–0,05 pro Mal), bei einem Anfangsgewicht von 0,2 – wie hoch ist das endgültige Gewicht?
a) 0,35
b) 0,60
c) 0,10
d) 0,45

9. Welche Aussage unterscheidet Bayes'sche Netze am besten von Monte-Carlo-Methoden?
a) Bayes'sche Netze nutzen Wahrscheinlichkeitsregeln, Monte Carlo nutzt Simulationen
b) Beide sind dieselbe Methode mit unterschiedlichen Namen
c) Bayes'sche Netze sind zufällig, Monte Carlo ist deterministisch
d) Monte Carlo verstärkt Verbindungen wie Hebb'sches Lernen

10. Wobei hilft das Hebb'sche Lernen in künstlichen neuronalen Netzen hauptsächlich?
a) Daten in Gruppen zu clustern
b) Verbindungen zwischen gemeinsam aktivierten Neuronen zu verstärken
c) Ergebnisse unter Unsicherheit vorherzusagen
d) Tausende von Zufallssimulationen durchzuführen

Modul 3: Scratch trifft auf Künstliche Intelligenz

EINLEITUNG

Ziel dieses Moduls ist es, die Programmiersprache Scratch vorzustellen und zu verstehen, wie Scratch ein einfaches und effektives Werkzeug zur Einführung in Programmierkonzepte sein kann. Am Ende des Moduls werden Sie in der Lage sein, verschiedene Fähigkeiten zu erwerben, wie zum Beispiel:

- Zu verstehen, wie **Scratch** eine zugängliche Plattform zum Erlernen von Programmierkonzepten bietet.
- Grundlegende **blockbasierte Programmierfähigkeiten** und computational thinking zu beherrschen.
- **Künstliche Intelligenz-Erweiterungen** in kreative Scratch-Projekte zu integrieren.

Moduldauer	**2 Stunden (Theorie: 1 Std. | Praxis: 1 Std.)**

UNTERRICHTSMATERIALIEN

Scratch ist eine leistungsstarke Programmiersprache, die visuell basiert ist. Das bedeutet, dass Scratch im Gegensatz zu traditionellen Programmiersprachen nicht auf Syntax oder zeilenbasiertem Code beruht; es funktioniert durch das Verbinden von Blöcken, um Logik zu erstellen. Es gibt viele Vorteile, Scratch als Einführung in die Programmierung zu verwenden. In diesem Modul lernen Sie die Geschichte von Scratch kennen und wie man mit der Online-Plattform programmiert. Am Ende des Moduls erfahren Sie, wie man KI-Erweiterungen zu Scratch-Anwendungen hinzufügt.

EINHEIT 3.1 EINLEITUNG TO SCRATCH

Scratch ist eine leistungsstarke visuelle Programmiersprache. Im Gegensatz zu traditionellen Programmiersprachen, die das Erlernen und Eintippen von komplexem Code erfordern, verwendet Scratch Code-Blöcke, die wie Puzzleteile zusammenpassen. Dieses Konzept macht Programmierung für jeden zugänglich, unabhängig von seiner Erfahrung und seinen Programmierkenntnissen.

WARUM SCRATCH EIN GROSSARTIGES WERKZEUG IST

Die wichtigsten Vorteile des Programmierens mit Scratch bestehen darin, dass es viele Hürden beseitigt, die Programmierung für Anfänger traditionell schwierig machen. Zum Beispiel weist Scratch folgende Unterschiede zu traditionellen Programmiersprachen auf:

- **Keine Syntaxfehler:** Code-Blöcke in Scratch lassen sich nur auf eine logisch sinnvolle Weise verbinden, was frustrierende Syntaxfehler verhindert.
- **Sofortiges visuelles Feedback:** Mit Scratch kannst du die Ergebnisse deines Codes sofort auf der Bühne sehen.
- **Niedrige Einstiegshürde, große Entwicklungsmöglichkeiten:** Scratch ist leicht zu erlernen, aber man kann dennoch anspruchsvolle Projekte erstellen.
- **Kreative Freiheit:** Erstelle Spiele, Animationen, interaktive Geschichten, Simulationen und vieles mehr.
- **Kollaborative Gemeinschaft:** Teile Projekte, remixe die Arbeit anderer und lerne von Gleichgesinnten.
- **Plattformunabhängig:** Scratch funktioniert in allen gängigen Webbrowsern auf jedem Gerät, einschließlich Tablets und Smartphones.

PRAKTISCHE ANWENDUNGEN VON SCRATCH

Scratch mag eine blockbasierte Sprache ohne Syntax sein; es vermittelt jedoch grundlegende Programmierkonzepte, die direkt auf andere Sprachen wie Python, JavaScript und Java übertragen werden können. Scratch behandelt Programmierkonzepte auf folgende Weise:

Programmierkonzept	In Scratch	Praktische Anwendung
Sequenzen	Blöcke werden in einer Reihenfolge gestapelt	Schritt-für-Schritt-Anweisungen
Schleifen	Wiederhole-Blöcke	Automatisierung von Aufgaben
Bedingungen	Wenn-dann-Blöcke	Entscheidungslogik
Variablen	Blöcke zur Datenspeicherung	Speichern und Verwalten
Ereignisse	Blöcke, die Aktionen auslösen	Behandlung von Benutzerinteraktionen
Parallelität	Mehrere Skripte laufen gleichzeitig	Multithread-Anwendungen

EINHEIT 3.2 GRUNDLEGENDE BLOCKBASIERTE PROGRAMMIERFÄHIGKEITEN

SO ERSTELLST DU EIN SCRATCH-KONTO

Die Kontoeröffnung ist einfach und kostenlos. Besuche zunächst die Scratch-Website unter https://scratch.mit.edu/. Erstelle dann ein Konto, indem du über die Hauptnavigation "Werde Mitglied bei Scratch" auswählst und den Registrierungsanweisungen folgst.

DIE SCRATCH-BENUTZEROBERFLÄCHE NAVIGIEREN

Die Benutzeroberfläche von Scratch bestimmt, wie ein Programm auf dem Bildschirm aussieht. Es gibt sieben verschiedene Elemente von Scratch, die du kennen musst, um mit dem Programmieren auf der Plattform beginnen zu können. Diese sind:

- Benutzeroberfläche
- Figuren (Sprites)
- Bühne
- Blöcke
- Kostüme
- Bühnenbilder
- Klänge

Die Scratch-Benutzeroberfläche ist in drei Hauptbereiche unterteilt: die Bühne (oben rechts), auf der die Projekte angezeigt werden; den Code-Bereich (Bildschirmmitte), in dem du Blöcke ziehen und zusammensetzen kannst, um Skripte zu erstellen; und den Figurenbereich (unten rechts), in dem du Figuren (Sprites) und Bühnenbilder hinzufügen und verwalten kannst.

Die linke Seite der Benutzeroberfläche enthält die Blockpalette, eine Werkzeugkiste mit verschiedenen Kategorien von Code-Blöcken, um Programme visuell zu erstellen. Zur Wiederholung: Die Hauptelemente der Benutzeroberfläche sind:

- Figurenbereich (unten rechts): Zeigt alle Figuren deines Projekts. Anklicken einer Figur, um ihren Code zu bearbeiten.
- Code-Bereich (Mitte): Arbeitsfläche zum Zusammenstecken von Blöcken für die ausgewählte Figur.
- Bühne (oben rechts): Visuelle Ausgabe, auf der die Figuren agieren – das sieht dein Publikum.

BLOCKKATEGORIEN VERSTEHEN

Scratch organisiert Code-Blöcke in neun farblich gekennzeichnete Kategorien. Die Kategorien sind wie folgt:

1. **Blau – Bewegung:** Die Bewegungskategorie ermöglicht es, deine Figur über den Bildschirm zu bewegen. Wenn du eine Figur vorwärts bewegen möchtest, kannst du einen Ändere x um 10-Block verwenden.

2. **Lila – Aussehen:** Die Aussehenskategorie konzentriert sich auf die Veränderung des Erscheinungsbilds von Figuren und Bühnenbildern. Du könntest einen Wechsle zu Kostüm-Block verwenden, um das Aussehen einer Figur zu ändern.

3. **Magenta – Klang:** Ermöglicht Figuren, Töne abzuspielen – z.B. mit dem Block "Spiele Klang ganz".

4. **Gelb – Ereignisse:** Jedes Scratch-Projekt braucht Ereignisse; sie lassen Dinge geschehen, wenn interagiert wird.

5. **Hellorange – Steuerung:** Bestimmt, wie oft oder wie lange eine Figur etwas tut. Mit dem Block "warte 1 Sekunde" pausiert das Programm für eine Sekunde.

6. **Türkis – Fühlen:** Prüft, ob eine Figur eine andere berührt, um dann eine Aktion auszulösen.

7. **Grün – Operatoren:** Führt mathematische und logische Operationen aus, wie Addition, Subtraktion oder das Verknüpfen von Text.

8. **Dunkelorange – Variablen:** Speichert und verwaltet Informationen wie Punktzahlen oder Lebenspunkte.

9. **Rot – Meine Blöcke:** Hier kannst du eigene, benutzerdefinierte Blöcke erstellen.

EINHEIT 3.3 PROJEKTE IN SCRATCH ERSTELLEN

Um ein Projekt in Scratch zu erstellen, beginne mit dem Hinzufügen einer Figur (Sprite) und eines Bühnenbildes aus der Bibliothek.

Eine neue Figur hinzufügen

In der unteren rechten Ecke des Scratch-Editors findest du die Figuren-Steuerung:

- Figur auswählen: Durchsuche die integrierte Figurenbibliothek (Tiere, Menschen, Gegenstände, Fantasiefiguren usw.)
- Malen: Erstelle deine eigene Figur mit den Zeichenwerkzeugen
- Überraschung: Füge eine zufällige Figur zur Inspiration hinzu
- Figur hochladen: Importiere eine Bilddatei von deinem Computer

Bühnenbild ändern

Das Bühnenbild gibt die Szene für dein Projekt vor. Klicke auf das Bühnenbild-Symbol (unten rechts neben der Bühne), um:

- Aus vorgefertigten Bühnenbildern zu wählen (draußen, drinnen, Weltraum, abstrakt usw.)
- Dein eigenes Bühnenbild mit dem Editor zu malen
- Ein Bild von deinem Computer hochzuladen

Du kannst mehrere Bühnenbilder haben und per Programm zwischen ihnen wechseln – mit Blöcken wie "wechsle zu Bühnenbild [Name]". Das ist nützlich für Spiele oder Geschichten mit mehreren Szenen.

EINE FIGUR BEWEGEN UND EINE AKTION AUSLÖSEN

Schritt 1 – Ereignisblock hinzufügen: Klicke auf die Kategorie Ereignisse (gelb). Ziehe den Block "wenn grüne Flagge angeklickt" in den Code-Bereich. Dieser Block startet dein Programm, wenn du die grüne Flagge über der Bühne anklickst.

Schritt 2 – Bewegung hinzufügen: Klicke auf die Kategorie Bewegung (blau). Ziehe den Block "gehe 10er-Schritt" und setze ihn unter den Ereignisblock. Ändere die Zahl von 10 auf 50, indem du darauf klickst.

Schritt 3 – Klangeffekt hinzufügen: Klicke auf die Kategorie Klang (magenta). Ziehe "spiele Klang [Miau] ganz" und verbinde es unter dem Bewegungsblock.

Schritt 4 – Figur sprechen lassen: Klicke auf die Kategorie Aussehen (lila). Ziehe "sage [Hallo!] für 2 Sekunden" und füge es deinem Skript hinzu. Du kannst "Hallo!" durch jede beliebige Nachricht ersetzen.

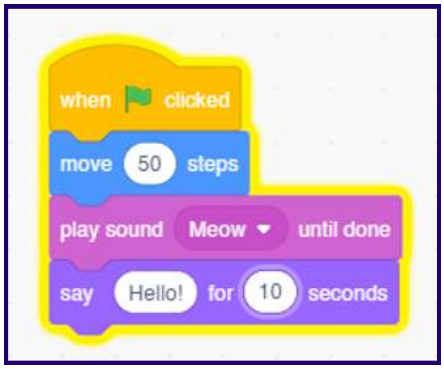


Abb. 3 Scratch block logic for movement and sound interaction: Scratch by MIT Media Lab

EINHEIT 3.4 EINFÜHRUNG IN SCRATCH KI-ERWEITERUNGEN

WAS IST EINE ERWEITERUNG IN SCRATCH?

Erweiterungen sind Zusatzmodule, die die Fähigkeiten von Scratch über die standardmäßigen Blockkategorien hinaus erweitern. Stelle dir Erweiterungen als spezialisierte Werkzeugkästen vor, die deinen Figuren neue Kräfte verleihen – vom Vorlesen von Text bis zur Bilderkennung durch künstliche Intelligenz.

Scratch 3.0 enthält Software-Erweiterungen, die programmiertechnische Funktionen hinzufügen, und Hardware-Erweiterungen, die eine Verbindung zu physischen Geräten wie Robotern und Elektronikbausätzen herstellen. In dieser Einheit konzentrieren wir uns auf KI-gestützte Software-Erweiterungen, die maschinelles Lernen und Sprachverarbeitung in deine Projekte bringen.

EINE ERWEITERUNG IN SCRATCH HINZUFÜGEN

Adding an extension is simple:

1. Suche unten links im Blockbereich nach der Schaltfläche mit dem "+"-Symbol
2. Klicke auf das Symbol, um die Erweiterungsbibliothek zu öffnen
3. Durchstöbere die verfügbaren Erweiterungen und klicke auf eine, um sie zu deinem Projekt hinzuzufügen
4. Neue Blöcke erscheinen sofort in deiner Palette und sind einsatzbereit

Die meisten KI-Erweiterungen benötigen eine aktive Internetverbindung, da die künstliche Intelligenz auf Cloud-Servern verarbeitet wird – nicht auf deinem Computer. Stelle sicher, dass du mit dem Internet verbunden bist, bevor du diese Erweiterungen verwendest. Achte außerdem auf den Datenschutz und vermeide das Hochladen persönlicher oder sensibler Informationen beim Trainieren von Modellen des maschinellen Lernens.

WARUM UND WIE KI IN KREATIVEN PROJEKTEN EINGESETZT WIRD

Künstliche Intelligenz ermöglicht es deinen Scratch-Projekten, Dinge zu tun, die mit traditioneller Programmierung allein fast unmöglich wären:

KI-Fähigkeit	**In Scratch**	**Parallele aus der realen Welt**
Bilderkennung	Ein Spiel, das erkennt, was du zeichnest oder der Webcam zeigst	Google Photos markiert automatisch Personen und Orte
Textklassifizierung	Ein Chatbot, der erkennt, ob Nachrichten Fragen, Aussagen oder Befehle sind	E-Mail-Spamfilter sortieren Nachrichten
Sprachsynthese	Figuren, die Geschichten vorlesen oder Spieleereignisse erzählen	GPS-Navigationssysteme geben Sprachhinweise
Stimmungsanalyse	Ein virtuelles Haustier, das auf nette versus gemeine Worte unterschiedlich reagiert	GPS-Navigationssysteme geben Sprachhinweise
Mustererkennung	Eine Musik-App, die erkennt, ob Klänge Schlagzeug, Klavier oder Gesang sind	Shazam erkennt Songs aus kurzen Audioclips

KI erweitert die kreativen Möglichkeiten, indem sie Projekte adaptiv (unterschiedliche Reaktionen basierend auf Eingaben), intelligent (Entscheidungen aus erlernten Mustern) und interaktiv (Verstehen verschiedener Eingabeformen wie Bilder, Text und Klang) macht.

EINE FIGUR BEWEGEN UND EINE AKTION AUSLÖSEN

OFFIZIELLE SCRATCH KI-ERWEITERUNGEN

- **Text zu Sprache:** Wandelt geschriebenen Text in gesprochene Wörter um – mit verschiedenen Stimmen und Sprachen
- **Übersetzen:** Übersetzt Text zwischen Sprachen mittels maschineller Übersetzung
- **Video-Erkennung:** Erkennt Bewegungen und Anwesenheit per Webcam (grundlegende Computer Vision)

DRITTANBIETER KI-ERWEITERUNGEN

- **Machine Learning for Kids:** Trainiere eigene Modelle zur Bild-, Text-, Zahlen- und Geräuscherkennung
- **Teachable Machine:** Googles Werkzeug zum schnellen Trainieren von Modellen, integrierbar mit Scratch
- **Gesichtserkennung:** Erkennt Gesichter und Gesichtsmerkmale durch eine Webcam
- **Spracherkennung:** Wandelt gesprochene Wörter in Text um

EINHEIT 3.5 KI-ERWEITERUNG: MACHINE LEARNING FOR KIDS

Machine Learning for Kids (ML4K) ist eine Bildungsplattform von IBM, die maschinelles Lernen für Lernende jeden Alters zugänglich macht. Sie bietet eine benutzerfreundliche Oberfläche zum Trainieren von KI-Modellen und lässt sich nahtlos über spezielle Blöcke in Scratch integrieren. Mit ML4K kannst du Modelle trainieren, um:

- **Bilder zu erkennen und zu klassifizieren** (z.B. Katzen von Hunden unterscheiden oder handgezeichnete Formen identifizieren)
- **Text zu verstehen und zu kategorisieren** (z.B. erkennen, ob Nachrichten Komplimente oder Beleidigungen sind)
- **Muster in Zahlen zu erkennen** (z.B. Vorhersagen auf Basis numerischer Eingaben treffen)
- **Geräusche zu erkennen** (z.B. zwischen Musiknoten oder Sprachbefehlen unterscheiden)

WAS IST MASCHINELLES LERNEN?

Maschinelles Lernen ist ein Teilbereich der künstlichen Intelligenz, bei dem Computer lernen, Aufgaben zu erledigen, indem sie Beispiele analysieren – anstatt vorprogrammierten Regeln zu folgen. Anstatt zum Beispiel jede Hunderasse zu beschreiben, zeigst du dem Computer tausende Hundebilder, und er lernt, Merkmale zu erkennen, die Hunderassen definieren. So funktionieren Dienste wie Google Fotos, die deine Bilder automatisch markieren.

SCHRITT-FÜR-SCHRITT: EIN MODELL TRAINIEREN UND NUTZEN

SCHRITT 1: MACHINE LEARNING FOR KIDS AUFRUFEN

Öffne deinen Webbrowser und gehe zu:
https://machinelearningforkids.co.uk/

Klicke auf "Get Started" und wähle für dieses Beispiel aus,
ein Konto ohne Registrierung zu erstellen.

Schritt 2: Ein neues Projekt erstellen

Nach dem Einloggen klicke auf "Projects" und dann auf "Add a new project". Du wirst aufgefordert:

- Projekt benennen: Wähle einen aussagekräftigen Namen (z.B. "Emotionsdetektor" oder "Mülltrennung")
- **Erkennungstyp auswählen:** Wähle zwischen Text, Zahlen, Bildern oder Geräuschen

Wähle für dieses Tutorial "Text" aus,
um einen Stimmungs-Klassifizierer zu erstellen.

Schritt 3: Labels (Kategorien) definieren

Modelle des maschinellen Lernens klassifizieren Eingaben in Kategorien, die "Labels" genannt werden.

- **Positiv:** Fröhliche, enthusiastische oder nette Nachrichten
- **Negativ:** Traurige, wütende oder gemeine Nachrichten

Klicke auf "Train" und dann für jede Kategorie auf "Add new label".

Schritt 4: Trainingsbeispiele bereitstellen
Das Modell lernt durch das Studium von Beispielen.
Füge für jedes Label mindestens 5-10 Beispielsätze hinzu:

Positive Beispiele	Negative Beispiele
Das ist großartig!	Das gefällt mir nicht.
Du machst das toll!	Das ist schrecklich.
Ich liebe es, Neues zu lernen	Das macht mich wütend.

Tipp: Je mehr Beispiele du bereitstellst, desto besser wird dein Modell funktionieren. Versuche mindestens zehn Beispiele pro Label und achte auf Abwechslung in den Formulierungen.

Schritt 5: Modell trainieren
Sobald du genügend Beispiele hinzugefügt hast, gehe zurück zum Projekt und wähle "Learn & Test" aus. Klicke dann auf den Button "Train new machine learning model". Das System analysiert deine Beispiele und erstellt ein Modell, das neue Texte klassifizieren kann. Das Training dauert in der Regel 1-3 Minuten. Du siehst einen Fortschrittsbalken. Wenn der Vorgang abgeschlossen ist, erhältst du eine Bestätigungsmeldung.

Schritt 6: Modell testen
Bevor du dein Modell in Scratch verwendest, teste es innerhalb von ML4K. Gib einfach einen Satz ein, den du nicht in deinen Trainingsdaten hattest (z.B. "Ich bin heute so glücklich!") und klicke auf "Test". Das Modell sagt vorher, welches Label am besten zu deiner Eingabe passt, und zeigt einen Prozentsatz der Wahrscheinlichkeit an.

Wenn das Modell Fehler macht, gehe zurück zum Trainingsbereich und füge weitere Beispiele hinzu, um die Genauigkeit zu verbessern.

Schritt 7: Mit Scratch verbinden

ML4K bietet eine spezielle Scratch-Integration:

1. Klicke in der Projektnavigation auf "Make"
2. Wähle "Scratch 3" aus
3. Dadurch öffnet sich ein angepasster Scratch-Editor, in dem dein ML-Modell als benutzerdefinierte Blöcke vorinstalliert ist.

Du wirst neue Blöcke sehen, die spezifisch für dein Projekt sind, wie zum Beispiel:

- erkenne Text [ ] (Label)
- erkenne Text [ ] (Wahrscheinlichkeit)

Schritt 8: ML-Blöcke in deinem Scratch-Projekt verwenden

Erstelle ein einfaches Scratch-Projekt, das dein trainiertes Modell nutzt:
Füge eine Figur hinzu (versuche es mit der Figur "Roboter");
Erstelle eine Variable namens Benutzereingabe; Baue dann dieses Skript:

Wenn grüne Flagge angeklickt
frage [Sag mir, wie du dich heute fühlst] und warte
setze [Benutzereingabe v] auf (Antwort)
falls <(erkenne Text (Benutzereingabe) (Label)) = [Positiv]> dann
sage [Ich freue mich, dass du dich gut fühlst!] für 3 Sekunden
sonst
sage [Es tut mir leid, dass es dir nicht so gut geht.
Die Dinge werden besser!] für 3 Sekunden

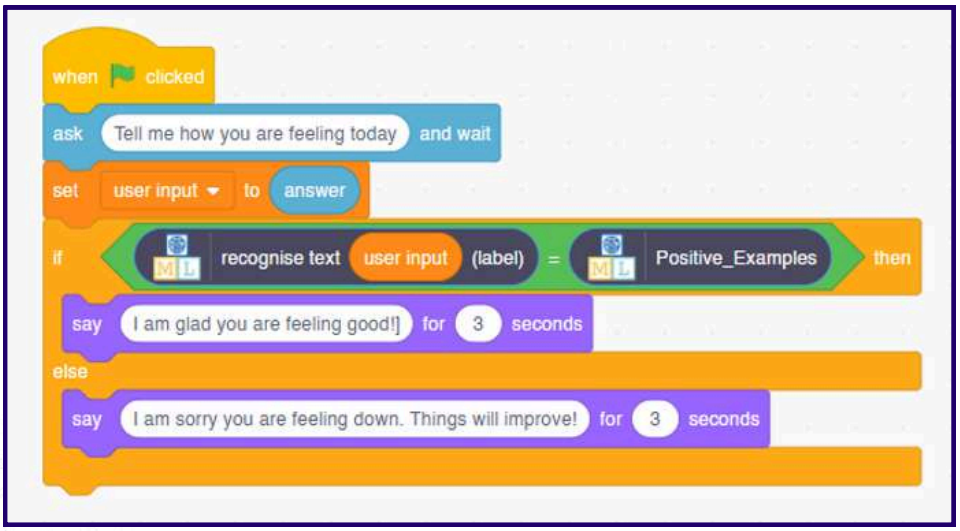


Abb. 4 Sequence of code blocks in Scratch illustrating a basic event-action chain: Scratch by MIT Media Lab

Schritt 9: Testen und verbessern
Starte dein Projekt und gib verschiedene Nachrichten ein. Beobachte, wie die Figur basierend auf der KI-Klassifizierung reagiert. Wenn das Modell Text falsch klassifiziert, kehre zu ML4K zurück, füge weitere Trainingsbeispiele hinzu, trainiere neu und aktualisiere dein Scratch-Projekt.

EINHEIT 3.6 KI-ERWEITERUNG: TEXT-TO-SPEECH

Text-to-Speech (TTS) ist eine KI-gestützte Technologie, die geschriebenen Text in gesprochene Audiodaten umwandelt. Diese offizielle Scratch-Erweiterung nutzt cloud-basierte Sprachsynthese, um deinen Figuren realistische Stimmen in mehreren Sprachen und Tonlagen zu verleihen. TTS macht Projekte zugänglicher (sehbehinderte Nutzer können Inhalte hören), ansprechender (gesprochene Dialoge wirken dynamischer als Sprechblasen) und vielseitiger (Projekte können Geschichten erzählen, Anleitungen geben oder Sprachlernwerkzeuge erstellen).

DIE TEXT-TO-SPEECH-ERWEITERUNG HINZUFÜGEN

Schritt 1: Die Erweiterungsbibliothek öffnen
Klicke im Scratch-Editor unten links in der Blockpalette auf den blauen Button "Erweiterung hinzufügen" (mit "+"-Symbol).

Schritt 2: Die Erweiterung auswählen
Scrolle durch die Erweiterungsbibliothek und klicke auf "Text-to-Speech" (Symbol mit Lautsprecher und Sprechblasen). Neue Blöcke erscheinen in deiner Palette unter einer neuen Kategorie "Text-to-Speech".

Schritt 3: Die Blöcke im Code verwenden

- Einen Satz sprechen lassen: Ziehe den Block sprich [Hallo] in deinen Code-Bereich. Klicke in den Block, um den Text zu ändern – ganz nach Wunsch.
- Die Stimme ändern: Verwende den Block setze Stimme auf [Alt], um eine andere Stimme auszuwählen. Zur Auswahl stehen: Alt, Tenor, Quietsch, Riese und Kätzchen.

Sprache ändern: Um die Sprache zu ändern, verwende den Block setze Sprache auf.

Abb. 5 Sequence of code blocks for conversation in Scratch: Scratch by MIT Media Lab

MIT SCRATCH-BLÖCKEN INTERAKTIVE PROJEKTE ERSTELLEN

Kombiniere die Text-to-Speech-Blöcke mit anderen Scratch-Blöcken, um interaktive Projekte zu erstellen. Du kannst zum Beispiel den Ereignis-Block wenn grüne Flagge angeklickt aus der Kategorie "Ereignisse" verwenden, um das Sprechen zu starten. Du kannst auch den Block frage [] und warte aus der Kategorie "Fühlen" und den Antwort-Block mit dem sprich-Block kombinieren, um deine Figur wiederholen zu lassen, was ein Benutzer eintippt.

WICHTIGE TEXT-TO-SPEECH-BLÖCKE

Diese Erweiterung bietet folgende Blöcke zur Sprachsteuerung:

Block	Funktion	Beispiel
speak [text]	Spricht den Text vor	speak [Hallo, willkommen in meinem Projekt!]
set voice to [voice]	Ändert den Stimmcharakter	set voice to [tenor] (options: alto, tenor, squeak, giant, kitten)
set language to [language]	Ändert die Sprache für die Ausgabe	set language to [Deutsch] (23 Sprachen unterstützt)

TEXT-TO-SPEECH-STIMMOPTIONEN

Es gibt verschiedene Text-to-Speech-Stimmoptionen, mit denen du experimentieren kannst, um sie an die Persönlichkeit deiner Figur anzupassen:

- Alto: Mittelhohe, neutrale Stimme
- Tenor: Etwas höhere Stimme
- Squeak: Hohe, kindliche Stimme
- Giant: Tiefe, dröhnende Stimme
- Kitten: Sehr hohe, niedliche Stimme

BEISPIELE FÜR DEN EINSATZ VON TEXT-TO-SPEECH IN SCRATCH

- **Interaktive Geschichten:**
  Figuren erzählen Handlungsentwicklungen und sprechen Dialoge
- **Lernspiele (Quiz):**
  Der Computer liest Fragen zur besseren Zugänglichkeit vor
- **Sprachen lernen:** Ausspracheübungen mit verschiedenen Sprachen
- **Virtuelle Assistenten:** Assistenten wie Siri, die Antworten sprechen
- **Audioguides:**
  Museumsführungen oder Stadtführer mit erzählten Informationen
- **Dynamisches Feedback:**
  Gesprochene Ermutigungen oder Hinweise in Spielen

REFERENZEN

Bücher, Wiki und Dokumentation

- Scratch Foundation (MIT Media Lab): Plattformdokumentation, Bildungsressourcen und Benutzeroberflächenbeschreibungen https://scratch.mit.edu/

- Machine Learning for Kids (Dale Lane, IBM) 11. Februar 2021: Plattform-Tutorials, Methodik zum Trainieren von Modellen und Integrationsanleitungen ISBN-13: 9781718500563

- Scratch Wiki: Technische Dokumentation für Erweiterungen und Blockbeschreibungen; https://scratch-wiki.info/

- The AI Scratch Code Playbook: A Beginner's Guide to Building Intelligent Systems (Taschenbuch – 11. Februar 2025); ISBN-13: 979-8310433649

LINKS ZU LERNPLATTFORMEN

Scratch Offizielle Website
Die offizielle Scratch-Plattform, entwickelt vom MIT Media Lab.
Erstelle ein Konto, entwickle Projekte, entdecke Millionen von Kreationen aus der Community und greife auf Tutorials und Lernressourcen zu.
https://scratch.mit.edu/

Machine Learning for Kids
Bildungsplattform von IBM zum Trainieren eigener maschineller Lernmodelle (Text, Bilder, Zahlen, Geräusche) und deren Integration in Scratch. Enthält Arbeitsblätter, Tutorials und vortrainierte Modelle.
https://machinelearningforkids.co.uk/

TEXT-TO-SPEECH-STIMMOPTIONEN

KI-Konzepte kreativ unterrichten mit Scratch (Codingal)
Blogbeitrag über kreative Ansätze zur Vermittlung von KI-Konzepten mit Scratch – mit Projektideen und pädagogischen Strategien.
https://www.codingal.com/coding-for-kids/blog/teach-kids-ai-coding-concepts-creatively-using-scratch/

Scratch Machine Learning Studio
Kuratierte Sammlung von Scratch-Projekten mit Machine-Learning-Erweiterungen. Zum Inspirieren und Remixen, um von bestehenden Projekten zu lernen.
https://scratch.mit.edu/studios/3995548/

PRAKTISCHE ÜBUNG

Ziel: Erstelle ein Projekt mit der Text-to-Speech-Erweiterung, um mit verschiedenen Stimmen zu experimentieren und sie an die Persönlichkeit deiner Figuren anzupassen.

Anleitung: Folge den Schritten unten, um eine Figur zu erstellen, die Benutzereingaben laut vorliest – das zeigt, wie Texteingabe mit Sprachausgabe kombiniert werden kann.

Schritt 1 – Projekt einrichten:

- Erstelle ein neues Scratch-Projekt oder arbeite mit einem bestehenden weiter.
- Wähle eine Figur aus, die sprechen soll – die Figuren "Roboter" oder "Zauberer" eignen sich gut.

Schritt 2 – Erweiterung hinzufügen:

- Füge die Text-to-Speech-Erweiterung wie oben beschrieben hinzu.

Schritt 3 – Skript erstellen:
Baue dieses Skript für deine Figur:

```
Wenn grüne Flagge angeklickt
    setze Stimme auf [Riese]
    sprich [Hallo! Ich kann alles sprechen, was du eintippst.]
    wiederhole fortlaufend
        frage [Was soll ich sagen?] und warte
        sprich (Antwort)
```

Schritt 4 – Projekt testen:
Klicke auf die grüne Flagge und tippe Nachrichten ein, wenn du dazu aufgefordert wirst. Deine Figur wird die Nachrichten laut vorlesen. Probiere verschiedene Stimmen und Sprachen aus, indem du die Blöcke setze Stimme auf und setze Sprache auf änderst.

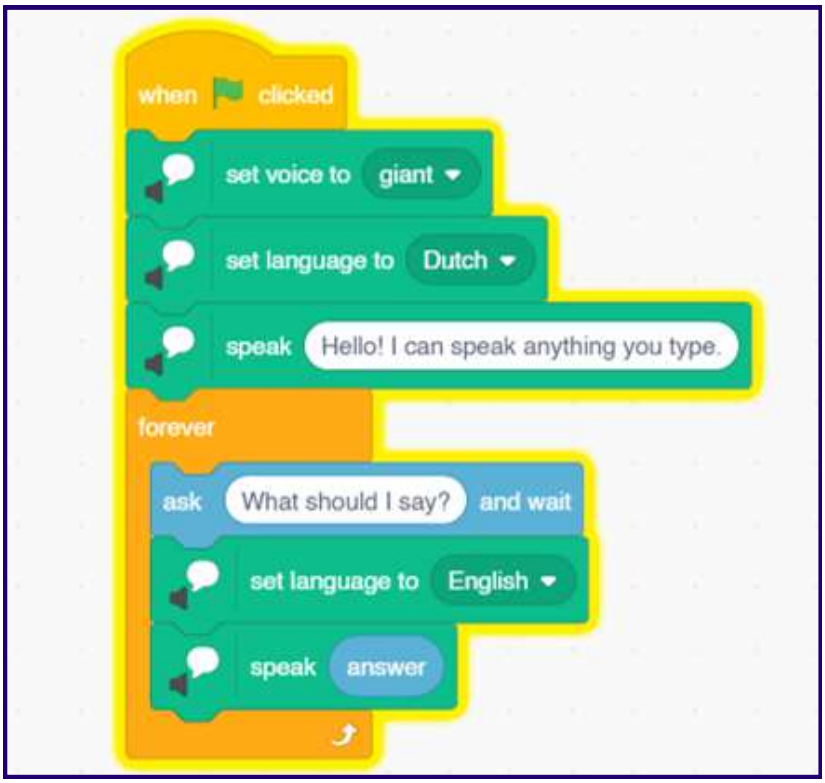


Abb. 6 Sequence of code blocks in Scratch: Scratch by MIT Media Lab

REFLEXIONSFRAGEN

Nimm dir Zeit, um über das nachzudenken, was du in diesem Modul gelernt hast. Beantworte die folgenden Fragen in Ruhe:

Reflexionsfrage 1:
Welcher Aspekt der Nutzung von KI in Scratch hat dich am meisten überrascht? Hat etwas anders funktioniert als erwartet? Erkläre anhand konkreter Beispiele aus deinem Projekt.

Reflexionsfrage 2:
Wie würdest du maschinelles Lernen in einem Spiel oder einer interaktiven Geschichte einsetzen? Beschreibe eine konkrete Projektidee und erkläre, wie KI es ansprechender oder intelligenter machen würde als traditionelle Programmierung allein.

Reflexionsfrage 3:
Welche Probleme oder Herausforderungen hattest du beim Trainieren deines ML-Modells oder beim Integrieren von KI-Erweiterungen? Wie hast du sie gelöst? Wenn du auf ungelöste Probleme gestoßen bist, welche Strategien würdest du als nächstes ausprobieren?

QUIZ MIT MEHRFACHWAHLFRAGEN (eine richtige Antwort pro Frage)

1. Was ist Scratch?
a) Textbasierte Programmiersprache für Profis
b) Visuelle, blockbasierte Programmiersprache für Anfänger
c) KI-Dienst für Machine Learning
d) Hardware-Roboter-Bausatz

2. Hauptvorteil der blockbasierten Programmierung?
a) Fördert das Auswendiglernen von Syntax
b) Blöcke verhindern Syntaxfehler durch logisch korrekte Verbindungen
c) Fokussiert nur auf textbasiertes Codieren
d) Nur für erfahrene Programmierer nutzbar

3. Welche Blockkategorie für eine 10-mal-Wiederholung?
a) Bewegung
b) Ereignisse
c) Steuerung
d) Operatoren

4. Was tun vor Nutzung eines ML4K-Modells in Scratch?
a) Nichts – Modell ist bereits vortrainiert
b) Trainingsbeispiele für jede Kategorie bereitstellen und Modell trainieren
c) Spezielle Hardware kaufen
d) Python-Code schreiben und importieren

QUIZ MIT MEHRFACHWAHLFRAGEN (eine richtige Antwort pro Frage)

5. Was ist KEINE KI-Fähigkeit aus diesem Modul?
a) Bilder durch Bilderkennung identifizieren
b) Eine Figur mit einer Wiederhol-Schleife tanzen lassen
c) Gesprochene Wörter durch Spracherkennung in Text umwandeln
d) Text innerhalb eines Projekts übersetzen

6. Welche Scratch-Erweiterung für sprechende Figuren?
a) Video-Erkennung
b) Übersetzen
c) Text-to-Speech
d) Machine Learning for Kids

7. Warum mehrere Trainingsbeispiele für ein ML-Modell?
a) Modell funktioniert nicht mit weniger als 100 Beispielen
b) Mehr Beispiele helfen dem Modell, Muster genauer zu lernen und Eingaben zuverlässiger zu klassifizieren
c) Trainingsbeispiele sind nur Deko und beeinflussen die Leistung nicht
d) Mehrere Beispiele verlangsamen das Modell und machen es genauer

8. Vorteil von ML4K + Text-to-Speech?
a) Lässt das Projekt schneller laufen
b) Versteht Eingaben und kann antworten
c) Reduziert die Menge an Code
d) Ermöglicht Offline-Nutzung ohne Internet

Modul 4: Python-Spieleprogrammierung (Pygame / Arcade-Bibliotheken) mit KI

EINLEITUNG

In diesem kurzen und anfängerfreundlichen Modul wirst du zum Spieleentwickler, nicht nur zum Spieler. Du lernst, wie du ein kleines Videospiel mit Python und Pygame entwirfst und programmierst und du wirst entdecken, wie du deine Spielfiguren mit einfachen Formen Künstlicher Intelligenz (KI) intelligent agieren lassen kannst. Keine Sorge, falls du noch nie programmiert hast, alles wird Schritt für Schritt erklärt. Du beginnst damit, ein kleines Spielefenster zu erstellen und ein Quadrat über den Bildschirm zu bewegen. Später erforschst du, wie du das Spiel auf deine Aktionen reagieren lassen kannst, zum Beispiel, indem ein Gegner deine Bewegungen bemerkt oder die Farbe wechselt, wenn du dich näherst. In diesem Modul geht es nicht nur darum, Code auszuführen, sondern darum, ihn zu verstehen und mit ihm zu experimentieren. Du wirst Beispiele testen und verändern, um zu sehen, wie kleine Änderungen unterschiedliche Verhaltensweisen erzeugen können. Dadurch lernst du, wie Spiele "denken" und reagieren, ähnlich wie einfache KI-Systeme in der realen Welt. Am Ende des Moduls wirst du in der Lage sein, verschiedene Fähigkeiten zu erwerben, wie zum Beispiel:

- Lerne, interaktive 2D-Spiele mit Pygame oder Arcade-Bibliotheken zu erstellen.
- Füge KI-Verhalten hinzu, damit Spielfiguren auf Spieleraktionen reagieren können.
- Nutze computational thinking, um interaktive Spiele zu entwerfen.
- Experimentiere, teste und verändere Spielcode, um zu beobachten, wie kleine Änderungen das Verhalten beeinflussen.
- Reflektiere darüber, wie KI im echten Leben eingesetzt wird.

Moduldauer	**2 Stunden (Theorie: 1 Std. | Praxis: 1 Std.)**

UNTERRICHTSMATERIALIEN

EINHEIT 4.1 EINFÜHRUNG IN PYGAME

Bevor du mit dem Programmieren beginnst, ist es wichtig, die Werkzeuge kennenzulernen, mit denen du arbeiten wirst:

- **Python 3 –** die Programmiersprache, mit der du dem Computer Anweisungen gibst. Sie wird weltweit in Schulen, Universitäten und Unternehmen für Programmierung, Datenanalyse und KI eingesetzt.

- **Pygame –** eine kostenlose Python-Bibliothek, mit der du einfache 2D-Spiele erstellen kannst. Sie bietet vorgefertigte Funktionen, um ein Spielefenster zu öffnen, Formen zu zeichnen, Farben anzuzeigen und auf Tastatur- oder Mauseingaben zu reagieren.

- **Code-Editor** (IDLE, Thonny oder VS Code) – das Programm, in dem du deinen Python-Code schreibst, speicherst und ausführst. In diesem Modul kannst du IDLE (im Lieferumfang von Python enthalten) oder einen anderen von deinem Lehrer empfohlenen Editor verwenden.

Diese Werkzeuge werden Schritt für Schritt vorgestellt, sodass du sie selbst installieren und die Aktivitäten eigenständig durchführen kannst – auch wenn dies dein erster Kontakt mit Programmierung ist. **Ziel:** Lerne, wie man ein Spielefenster öffnet, Farben anzeigt und ein Quadrat mit der Tastatur bewegt.

Schritt 1 – Python 3 installieren und Programmierumgebung öffnen
Bevor wir mit Pygame beginnen können, musst du sicherstellen, dass Python 3 auf deinem Computer installiert ist.

Python 3 installieren

1. Gehe auf die offizielle Python-Website: https://www.python.org/downloads/
2. Lade die Version für dein System herunter (Windows, macOS oder Linux)
3. Aktiviere während der Installation die Option "Add Python to PATH", bevor du auf "Install Now" klickst
4. Starte deinen Computer nach der Installation neu (optional, aber empfohlen)

EINHEIT 4.2 INSTALLATION ÜBERPRÜFEN

Öffne die Eingabeaufforderung (Windows) oder das Terminal (macOS/Linux) und gib folgenden Befehl ein:

python --version or **py --version**

Wenn etwas wie Python 3.13.3 angezeigt wird, ist alles bereit.

EINHEIT 4.3 EDITOR INSTALLIEREN ODER IDLE NUTZEN

Du kannst Python-Code mit einem dieser Editoren schreiben:

- IDLE (wird mit Python mitinstalliert)
- Thonny (einfache IDE für Anfänger) – https://thonny.org
- VS Code (fortgeschrittene Option) – https://code.visualstudio.com

IDLE öffnen:

- Windows: Start → IDLE (Python 3.x) eingeben und öffnen
- macOS: Im Launchpad nach "IDLE" suchen
- Linux: idle3 im Terminal eingeben

3. Code eingeben oder einfügen

Kopiere den Code aus diesem Modul in die neue Datei.
Zum Beispiel:

CODE:

```
import pygame
pygame.init()
screen = pygame.display.set_mode((800, 600))
pygame.display.set_caption("Mein erstes Spiel")
clock = pygame.time.Clock()

running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    screen.fill((135, 206, 235))  # Himmelblau
    pygame.display.flip()
    clock.tick(60)

pygame.quit()
```

4. Datei speichern

Wähle Datei → Speichern unter...
Benenne deine Datei (z.B. mein_erstes_spiel.py).

5. Programm ausführen

Drücke F5 oder gehe zu Run → Run Module.
Ein neues Fenster öffnet sich
– wenn du einen blauen Hintergrund siehst, funktioniert alles!

EINHEIT 4.4 GRUNDLEGENDE BLOCKBASIERTE PROGRAMMIERUNG

Kopiere diesen Code und führe ihn aus:

CODE:

```
import pygame                          # 1. Importiere die Pygame-
Bibliothek
pygame.init()                           # 2. Starte
(initialisiere) alle Pygame-Module

screen = pygame.display.set_mode((800, 600))    # 3. Erstelle
ein Spielfenster (Breite 800, Höhe 600)
pygame.display.set_caption("Mein erstes Spiel") # 4. Gib dem
Spielfenster einen Titel

running = True                          # 5. Erstelle eine
Variable, um das Spiel am Laufen zu halten

while running:                          # 6. Starte die
Hauptspiel-Schleife (läuft immer wieder)
    for event in pygame.event.get():    # 7. Überprüfe alle
Ereignisse (z.B. Tastendrücke oder Mausklicks)
        if event.type == pygame.QUIT:   # 8. Wenn der Benutzer
den "Schließen"-Button klickt…
            running = False             # 9. …stoppe die
Spiel-Schleife

    screen.fill((200, 220, 255))        # 10. Fülle den
Bildschirm mit einer hellblauen Farbe (RGB-Werte)
    pygame.display.flip()               # 11. Aktualisiere das
Fenster, um den neuen Rahmen anzuzeigen

pygame.quit()                           # 12. Schließe das
Spiel und beende es sicher
```

EINHEIT 4.5 EIN QUADRAT BEWEGEN

Jetzt lassen wir etwas bewegen. Kopiere den folgenden Code und teste ihn:

PART I

CODE:

```
import pygame    # 1. Importiere die Pygame-Bibliothek
pygame.init()     # 2. Starte Pygame

# 3. Erstelle das Fenster
screen = pygame.display.set_mode((800, 600))
# 4. Füge einen Titel hinzu
pygame.display.set_caption("Bewegliches Quadrat")

x = 400   # 5. Die X-Position des Quadrats (horizontal)
y = 300   # 6. Die Y-Position des Quadrats (vertikal)
speed = 1 # 7. Wie viele Pixel es sich pro Schritt bewegt

running = True # 8. Halte das Spiel am Laufen

while running: # 9. Starte die Spiel-Schleife
    for event in pygame.event.get():     # 10. Überprüfe
Ereignisse
        if event.type == pygame.QUIT:    # 11. Wenn Benutzer
"X" klickt, stoppen
            running = False

    # Hier kommt später der Bewegungscode hin

pygame.quit() # 12. Schließe das Spiel sicher
```

EINHEIT 4.5 EIN QUADRAT BEWEGEN

Jetzt lassen wir etwas bewegen. Kopiere den folgenden Code und teste ihn:

```
PART II

CODE:
    # Tastendrücke prüfen
    keys = pygame.key.get_pressed()
    if keys[pygame.K_LEFT]:
        x -= speed
    if keys[pygame.K_RIGHT]:
        x += speed
    if keys[pygame.K_UP]:
        y -= speed
    if keys[pygame.K_DOWN]:
        y += speed

    # Bildschirm aktualisieren
    screen.fill((200, 220, 255))
    pygame.draw.rect(screen, (0, 0, 255), (x, y, 50, 50))
    pygame.display.flip()

pygame.quit()
```

Was passiert hier:

- x und y bestimmen die Position des Quadrats auf dem Bildschirm.
- Die Pfeiltasten verändern diese Werte und bewegen das Quadrat.
- Die Schleife wiederholt das Zeichnen viele Male pro Sekunde und erzeugt so eine flüssige Bewegung.

Probiere Folgendes:

- Ändere speed = 1 zu speed = 5. Was passiert?
- Ändere die Farbe des Quadrats, z.B. zu (255, 0, 0) für Rot.
- Was würde wohl passieren, wenn du pygame.display.flip() entfernst?

Reflexionsfrage:

Woher weiß der Computer jedes Mal, wenn du eine Taste drückst, die neue Position des Quadrats?

Lernkontrolle

An diesem Punkt solltest du in der Lage sein:

✓ Ein Pygame-Fenster zu öffnen
✓ Die Spiel-Schleife zu nutzen, um das Spiel am Laufen zu halten
✓ Ein bewegliches Objekt mit der Tastatur zu steuern

Im nächsten Teil wirst du Künstliche Intelligenz hinzufügen, damit dein Spiel auf deine Bewegungen reagiert.

EINHEIT 4.7 EINFACHE KI IN SPIELEN (REAKTIVER GEGNER)

In diesem Modul verwenden wir eine einfache regelbasierte Spiel-KI: Der Computer folgt vordefinierten Regeln, um den Spieler wahrzunehmen, zu entscheiden, was zu tun ist, und entsprechend zu handeln, ohne aus Daten zu lernen. Dies ähnelt vielen klassischen Videospielen, in denen Gegner den Spieler jagen, ihm ausweichen oder aufgrund von Distanz oder Position reagieren.

Ziel: Lerne, wie dein Spiel auf Aktionen des Spielers reagieren kann, indem du eine einfache Form von Künstlicher Intelligenz (KI) hinzufügst. Du wirst einen Gegner erstellen, der die Farbe ändert, wenn der Spieler sich nähert.

Schritt 1 – Die Idee verstehen

Bevor du programmierst, überlege, was KI in einem Spiel bedeutet. In diesem Fall geht es nicht um Lernen oder Daten – es geht darum, den Computer auf die Aktionen des Spielers reagieren zu lassen.

Du wirst eine einfache Regel anwenden: Wenn der Spieler nah ist, ändert der Gegner seine Farbe. Wenn nicht, bleibt sie gleich. Dies ist ein Beispiel für bedingtes Verhalten, eine der einfachsten Formen von KI.

```
PART I

CODE:
import pygame, math           # 1. Importiere Spiel- und
Mathe-Bibliotheken
pygame.init()                  # 2. Starte alle Pygame-
Funktionen

screen = pygame.display.set_mode((800, 600))
# 3. Erstelle ein Spielfenster (800x600)
pygame.display.set_caption("Einfache KI - Reaktiver Gegner")
# 4. Titel

player = pygame.Rect(100, 100, 50, 50)
# 5. Spieler (blaues Quadrat)
enemy = pygame.Rect(500, 300, 50, 50)
# 6. Gegner (rotes Quadrat)
speed = 5                      # 7. Spieler-Geschwindigkeit
react_distance = 120
# 8. Distanz (Pixel) für Gegner-Reaktion

clock = pygame.time.Clock()
running = True                 # 9. Halte das Spiel am Laufen
```

PART II

CODE:

```
while running: # 10. Hauptspiel-Schleife (jeder Frame)
    for event in pygame.event.get():   # 11. Überprüfe alle
Ereignisse
        if event.type == pygame.QUIT:  # 12. Wenn das Fenster
geschlossen wird…
            running = False    # 13. …stoppe die Schleife

    # ---- BEWEGUNG (muss innerhalb der while-Schleife sein)
----
    keys = pygame.key.get_pressed()    # 14. Welche Tasten
sind gedrückt?
    if keys[pygame.K_LEFT]:
        player.x -= speed  # 15. Nach links bewegen
    if keys[pygame.K_RIGHT]:
        player.x += speed  # 16. Nach rechts bewegen
    if keys[pygame.K_UP]:
        player.y -= speed  # 17. Nach oben bewegen
    if keys[pygame.K_DOWN]:
        player.y += speed  # 18. Nach unten bewegen

    # ---- EINFACHE KI: Farbe ändern bei Nähe ----
    px, py = player.center # 19. Spieler-Mittelpunkt
    ex, ey = enemy.center  # 20. Gegner-Mittelpunkt
    dist = math.hypot(px - ex, py - ey) # 21. Distanz
berechnen

    if dist < react_distance: # 22. Wenn näher als 120 px…
        enemy_color = (255, 200, 0) # 23. …Gegner wird gelb
    else:
        enemy_color = (220, 60, 60)  # 24. Sonst bleibt
Gegner rot
```

```
PART III

CODE:
    # ---- ZEICHNEN ----
    screen.fill((240, 245, 255))         # 25. Hintergrund
    pygame.draw.rect(screen, (60, 120, 255), player)
    # 26. Spieler (blau)
    pygame.draw.rect(screen, enemy_color, enemy)
    # 27. Gegner (reaktiv)
    pygame.display.flip()    # 28. Änderungen anzeigen

    clock.tick(60)        # Auf 60 FPS begrenzen

pygame.quit()             # 29. Sicher beenden
```

Schritt 3 – Was passiert in diesem Code

Konzept	Erklärung
Wahrnehmung	Der Gegner misst mit math.hypot die Entfernung zum Spieler.
Entscheidung	Wenn die Distanz kleiner als react_distance ist, entscheidet sich der Gegner zu reagieren.
Aktion	Die Reaktion ist ein Farbwechsel.

Dieser "Wahrnehmen → Entscheiden → Handeln"-Zyklus ist die Grundlage der meisten Spiel-KI-Systeme. Selbst einfache Reaktionen wie diese lassen die Spielwelt lebendiger wirken.

SCHRITT 4 – AUSPROBIEREN, VERÄNDERN, BEOBACHTEN

Ändere diese Teile des Codes und beobachte, was passiert:

1. react_distance von 120 auf 50 oder 300 ändern
2. speed von 5 auf 10 oder 2 ändern
3. Die Farbe des Gegners bei Annäherung ändern
4. Den Gegner beweglich machen (statt statisch);
   Zusätzliche Distanzbereiche einführen
   (z.B. bei 150px orange, bei 80px rot)

Schritt 5 – Reflexion

- Ist das wirklich "Intelligenz"? Warum oder warum nicht?
- Wie könnten wir das Verhalten des Gegners schlauer wirken lassen?
- Was wäre, wenn der Gegner weggeht, wenn der Spieler zu nah kommt?
- Oder wenn er dem Spieler folgt?

Hinweis: Echte KI in Spielen verwendet die gleiche Logik, einfache Entscheidungen, die viele Male pro Sekunde wiederholt werden, können die Illusion von Intelligenz erzeugen.

REFERENZEN

Bücher, Wiki und Dokumentation

- Python Software Foundation. Python 3 Dokumentation. Verfügbar unter: https://docs.python.org
- Pygame Community. Pygame Dokumentation und Tutorials. Verfügbar unter: https://www.pygame.org/docs/ und https://www.pygame.org/wiki/tutorials
- Python Arcade Project. Python Arcade Library Dokumentation. Verfügbar unter: https://api.arcade.academy
- Real Python. Pygame: Eine Einführung in die Spieleprogrammierung mit Python. Verfügbar unter: https://realpython.com/pygame-a-primer/
- Microsoft. KI-Lehrplan für Anfänger. Verfügbar unter: https://github.com/microsoft/AI-For-Beginners

LINKS ZU LERNPLATTFORMEN

- Python Offizielle Website – Python herunterladen und die Grundlagen der Programmierung lernen. (https://www.python.org/)
- Pygame Dokumentation – Offizielle Anleitung zu Funktionen, Ereignissen und Grafiken in Pygame. (https://www.pygame.org/docs/)
- Arcade Bibliothek Dokumentation – Eine alternative Bibliothek für einfache 2D-Spiele in Python. (https://api.arcade.academy/en/latest/)
- Pygame KI-Guide – Beispiele für einfache KI-Algorithmen für Pygame. (https://pygame-ai.readthedocs.io/en/latest/guide.html)
- Real Python – Ein Spiel mit Pygame entwickeln – Ein anfängerfreundliches Tutorial. (https://realpython.com/pygame-a-primer/)

- KidsCanCode Tutorials – Einfache Projekte und Herausforderungen zum Lernen von Pygame. (https://kidscancode.org/blog/)

- KI für Anfänger (Microsoft) – Offene Bildungsinhalte zur Einführung in KI-Grundlagen. (https://microsoft.github.io/AI-For-Beginners/)

PRAKTISCHE ÜBUNG

FANG MICH, WENN DU KANNST

Ziel: Erstelle ein kleines Spiel, bei dem der Spieler ein blaues Quadrat bewegt, um Objekte zu sammeln, während ein roter KI-Gegner dem Spieler folgt. Diese Übung vereint alle gelernten Fähigkeiten: Bewegung, Kollisionserkennung und einfaches KI-Verhalten.

Schritt 1 – Umgebung vorbereiten
Stelle sicher, dass Pygame installiert ist:

pip install pygame

Öffne dann deinen Python-Editor und erstelle eine neue Datei mit dem Namen fang_mich.py.

Schritt 2 – Die Spielidee verstehen

Element	Beschreibung
Spieler	Bewegt sich mit den Pfeiltasten.
Ziel	Erscheint zufällig; erhöht den Punktestand beim Einsammeln.
Gegner (KI)	Folgt dem Spieler, indem er Richtung und Distanz berechnet.
Ziel	Sammle so viele Ziele wie möglich, bevor der Gegner dich fängt.

Denke vor dem Programmieren nach: Wie könnte ein Gegner "wissen", wo der Spieler ist?

SCHRITT 3 – DAS SPIEL SCHRITT FÜR SCHRITT AUFBAUEN

PART I

CODE:

```
import pygame, random, math
pygame.init()

# --- Fenster ---
WIDTH, HEIGHT = 800, 600
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Fang mich, wenn du kannst")
clock = pygame.time.Clock()

# --- Farben ---
BLUE = (60, 120, 255)
RED = (220, 60, 60)
GREEN = (60, 200, 60)
WHITE = (240, 245, 255)
BLACK = (20, 20, 20)

# --- Spielobjekte ---
player = pygame.Rect(100, 100, 40, 40)
# blaues Quadrat (Spieler)
enemy = pygame.Rect(600, 400, 40, 40)
# rotes Quadrat (verfolgender Gegner)
target = pygame.Rect(
    random.randint(50, WIDTH - 50),    # zufällige X-Position
    random.randint(50, HEIGHT - 50),   # zufällige Y-Position
    25, 25)                            # Größe: 25x25 Pixel
# grünes Ziel
```

SCHRITT 3 – DAS SPIEL SCHRITT FÜR SCHRITT AUFBAUEN

PART II

CODE:

```
PLAYER_SPEED = 5
ENEMY_SPEED = 2.5
score = 0

# Schriftart (fallback-freundlich: verwendet
Standardschriftart)
font = pygame.font.Font(None, 26)

# --- Spiel-Schleife ---
running = True
while running:
    # 1) Ereignisse behandeln
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    # 2) Spielerbewegung (Pfeiltasten)
    keys = pygame.key.get_pressed()
    if keys[pygame.K_LEFT]:
        player.x -= PLAYER_SPEED
    if keys[pygame.K_RIGHT]:
        player.x += PLAYER_SPEED
    if keys[pygame.K_UP]:
        player.y -= PLAYER_SPEED
    if keys[pygame.K_DOWN]:
        player.y += PLAYER_SPEED
```

SCHRITT 3 – DAS SPIEL SCHRITT FÜR SCHRITT AUFBAUEN

PART III

CODE:

```
    # Spieler innerhalb der Fenstergrenzen halten
    player.clamp_ip(screen.get_rect())

    # 3) Einfache KI: Gegner jagt den Spieler
    dx = player.centerx - enemy.centerx
    dy = player.centery - enemy.centery
    dist = math.hypot(dx, dy)
    if dist != 0:  # Division durch Null vermeiden
        enemy.x += int((dx / dist) * ENEMY_SPEED)
        enemy.y += int((dy / dist) * ENEMY_SPEED)

    # 4) Spieler sammelt Ziel ein
    if player.colliderect(target):
        score += 1
        target.topleft = (
            random.randint(25, WIDTH - 50),
            random.randint(25, HEIGHT - 50)
        )

    # 5) Alles zeichnen
    screen.fill(WHITE)
    pygame.draw.rect(screen, BLUE, player)
    pygame.draw.rect(screen, RED, enemy)
    pygame.draw.rect(screen, GREEN, target)
```

SCHRITT 3 – DAS SPIEL SCHRITT FÜR SCHRITT AUFBAUEN

```
PART IV

CODE:
    # Punktestand anzeigen
    score_text = font.render(f"Punkte: {score}", True, BLACK)
    screen.blit(score_text, (10, 10))

    pygame.display.flip()
    clock.tick(60)

pygame.quit()
```

Schritt 4 – Experimentieren und Reflektieren

Ändere diese Werte und beobachte, was passiert:

- Ändere ENEMY_SPEED, um den Gegner schneller oder langsamer zu machen.
- Ändere die Farbe des Spielers oder Ziels.
- Mache das Spiel schwieriger, indem du die Größe des Spielers reduzierst.
- Füge eine Game Over-Nachricht hinzu, wenn der Gegner den Spieler berührt.

REFLEXIONSFRAGEN

Nimm dir Zeit, um über das nachzudenken, was du in diesem Modul gelernt hast. Beantworte die folgenden Fragen in Ruhe:

- **Reflexionsfrage 1:** Was war der schwierigste Teil bei der Erstellung deines Spiels?
- **Reflexionsfrage 2:** Wie macht die KI das Spiel interaktiver?
- **Reflexionsfrage 3:** Kannst du dir andere Situationen vorstellen, in denen KI auf menschliche Aktionen reagiert (zum Beispiel in der Bildung oder im Gesundheitswesen)?
- **Reflexionsfrage 4:** Wenn du dieses Spiel verbessern könntest, welche Art von KI-Verhalten würdest du als Nächstes hinzufügen?
- **Reflexionsfrage 5:** Was lässt den Gegner "intelligent" wirken?
- **Reflexionsfrage 6:** Wie hilft die Distanzberechnung (math.hypot) dem Gegner, den Spieler zu finden?
- **Reflexionsfrage 7:** Kennst du reale Spiele, die eine ähnliche Logik verwenden?

QUIZ MIT MEHRFACHWAHLFRAGEN (eine richtige Antwort pro Frage)

1. Was macht die Spiel-Schleife in Pygame?
a) Sie installiert neue Bibliotheken während des Spiels
b) Sie wiederholt die Spielaktionen viele Male pro Sekunde
c) Sie schließt das Spielfenster
d) Sie lädt das Hintergrundbild nur einmal

2. Was ist der Zweck von pygame.display.flip()?
a) Es prüft die Tastatureingaben
b) Es zeichnet ein Rechteck
c) Es aktualisiert das Spielfenster mit neuen Inhalten
d) Es pausiert die Spiel-Schleife

3. Welche Funktion prüft, welche Tastaturtasten gedrückt werden?
a) pygame.quit()
b) pygame.key.get_pressed()
c) pygame.event.get()
d) pygame.display.set_mode()

4. Was lässt den Gegner im KI-Beispiel auf den Spieler reagieren?
a) Ein Timer-Ereignis
b) Zufällige Bewegung
c) Die Distanz zwischen Spieler und Gegner
d) Die Farbe des Spielers

5. Was passiert, wenn du den Wert von react_distance im Code erhöhst?
a) Der Gegner reagiert aus größerer Entfernung
b) Der Gegner bewegt sich schneller
c) Der Spieler wird langsamer
d) Die Fenstergröße ändert sich

QUIZ MIT MEHRFACHWAHLFRAGEN (eine richtige Antwort pro Frage)

6. Was ist der Zweck der math.hypot()-Funktion im KI-Code?
a) Die Distanz zwischen zwei Punkten berechnen
b) Den Gegner auf dem Bildschirm zeichnen
c) Eine neue zufällige Position erstellen
d) Kollisionen mit Wänden erkennen

7. Was macht diese Codezeile?
(player = pygame.Rect(100, 100, 40, 40))
a) Sie zeichnet Text auf den Bildschirm
b) Sie erstellt ein Rechteck, das den Spieler darstellt
c) Sie startet den KI-Algorithmus
d) Sie ändert die Hintergrundfarbe

8. Was passiert, wenn du pygame.display.flip() aus der Schleife entfernst?
a) Das Programm läuft schneller
b) Das Fenster aktualisiert sich nicht und sieht eingefroren aus
c) Die Farben ändern sich automatisch
d) Der Spieler bewegt sich doppelt so schnell

9. Welches ist ein Beispiel für KI-Verhalten in einem Spiel?
a) Ein Quadrat auf dem Bildschirm zeichnen
b) Eine Figur nur bewegen, wenn der Spieler eine Taste drückt
c) Einen Gegner den Spieler jagen oder ihm ausweichen lassen
d) Hintergrundmusik laden

10. Wie kann KI Spiele interessanter machen?
a) Durch automatisches Beheben von Fehlern
b) Durch dynamische Herausforderungen und reaktionsfähige Figuren
c) Durch automatisches Ändern der Grafik
d) Durch Steuerung der Tastatur

Modul 5: Objekterkennung mit Roboflow – Einführung in Computer Vision

EINLEITUNG

Ziel dieses Moduls ist es, eine umfassende Einführung in Computer Vision zu geben, zu erklären, wie Maschinen visuelle Daten wahrnehmen und verarbeiten, und die Lernenden dabei zu unterstützen, eigene Objekterkennungsmodelle mit der Roboflow-Plattform und fortschrittlichen Algorithmen wie YOLO zu entwickeln. Am Ende des Moduls wirst du in der Lage sein, verschiedene Fähigkeiten zu erwerben, wie zum Beispiel:

- **Grundlagen verstehen:** Unterscheide zwischen Bildverarbeitung und Computer Vision und verstehe die Logik, wie Computer Bilder als Daten interpretieren (Pixel und Binärcode).
- **Den CV-Prozess anwenden:** Wende die fünf wesentlichen Schritte des Computer-Vision-Prozesses an: Daten beschaffen, Daten vorbereiten, KI-Modell bestimmen, Modell trainieren und Ergebnisse interpretieren.
- **Roboflow nutzen:** Navigiere auf der Roboflow-Plattform, um Projekte zu erstellen, Datensätze hochzuladen und Bilddaten effektiv zu verwalten.
- **Datenannotation:** Führe genaue Datenannotation (Beschriftung) durch, um Datensätze für das KI-Modelltraining vorzubereiten.
- **Modelltraining:** Wähle fortschrittliche Objekterkennungsmodelle wie YOLOv11 aus und trainiere sie, um bestimmte Objekte zu identifizieren.

Moduldauer	**2 Stunden (Theorie: 1 Std. | Praxis: 1 Std.)**

UNTERRICHTSMATERIALIEN

EINHEIT 5.1 WAS IST COMPUTER VISION (BILDVERARBEITUNG)?

Bildverarbeitung ist eine Technik, mit der wichtige Informationen in einem digitalen Bild erkannt, extrahiert und ausgewertet werden. Diese Methode liefert aussagekräftige Daten über Objekte oder Umgebungen in einem Bild – ähnlich der menschlichen visuellen Wahrnehmung. Das Hauptziel der Bildverarbeitung ist es, Bedeutung aus Bildern zu extrahieren und diese Informationen für verschiedene Anwendungen nutzbar zu machen.

Obwohl es visuelle Informationsverarbeitungstechnologien schon lange gibt, war dieser Prozess in der Vergangenheit weitgehend auf menschliches Eingreifen angewiesen. Das war sowohl zeitaufwendig als auch fehleranfällig. Bei früheren Gesichtserkennungssystemen mussten Entwickler beispielsweise Tausende von Fotos manuell beschriften – bestimmte Merkmale wie Nasenbreite oder Augenabstand wurden einzeln identifiziert. Das lag daran, dass Bilddaten oft unstrukturiert und für Computer schwer zu verstehen waren. Die Automatisierung dieses Prozesses erforderte daher leistungsstarke Prozessoren und fortschrittliche Computertechnologien.

Heute nutzen Computer-Vision-Anwendungen mit Hilfe der sich ständig weiterentwickelnden Rechenleistung Künstliche Intelligenz und maschinelles Lernen (KI/ML), um diese Daten präzise zu verarbeiten – sei es zur Objekterkennung, Gesichtserkennung, Klassifizierung, Empfehlung, Verfolgung oder Wahrnehmung. Computer Vision lässt sich einfach beschreiben als „Maschinen diese Aufgabe ausführen zu lassen“, inspiriert vom menschlichen visuellen System und seiner Verbindung zum Gehirn. Es ist der Prozess, bei dem Maschinen wie Computer Bilder und Videos wahrnehmen, analysieren und mithilfe von maschinellem Lernen und Künstlicher Intelligenz in Bedeutung umwandeln.
Die Ursprünge der Computer Vision reichen bis in die 1950er Jahre zurück. In den 1970er Jahren begann ihre Kommerzialisierung mit Anwendungen zur Unterscheidung von Schreibmaschinen- und Handschrift. Heute wird sie in vielen Bereichen eingesetzt, darunter Personen- und Gesichtserkennung, Produktqualitätskontrolle, autonome Fahrzeuge, landwirtschaftliche Anwendungen, Gesundheitswesen, Bildung und Verteidigung.

Während die Bildverarbeitung die Aufbereitung, Wiederherstellung und Anpassung von Helligkeit und Farbe erfasster Bilddaten (wie Fotos und Videos) umfasst, geht Computer Vision einen Schritt weiter: Sie wählt Bildverarbeitungstechniken auf Bilder an und kombiniert diesen Prozess mit Künstlicher Intelligenz. Ein Beispiel: Die Bildverarbeitung wird genutzt, um die Kanten eines Objekts in einem Bild zu finden. Die Anwendung von maschinellen Lernverfahren zur Identifizierung und Klassifizierung dieses Objekts fällt dagegen in den Bereich der Computer Vision.

EINHEIT 5.2 WIE SIEHT EIN COMPUTER?

Die Grundbausteine fast aller Computer weltweit, ihre „Zellen", bestehen aus den Zahlen 1 und 0. Das bedeutet: Computer nehmen die Daten, die Nutzer*innen ihnen übermitteln wollen, in ihrer eigenen Sprache wahr – in Einsen und Nullen. Da Bilder und Videos (die ja aus vielen Einzelbildern bestehen) ebenfalls Daten sind, nehmen Computer auch sie als Nullen und Einsen wahr, also im binären Zahlensystem. Um dies zu veranschaulichen, kannst du das folgende Bild betrachten.

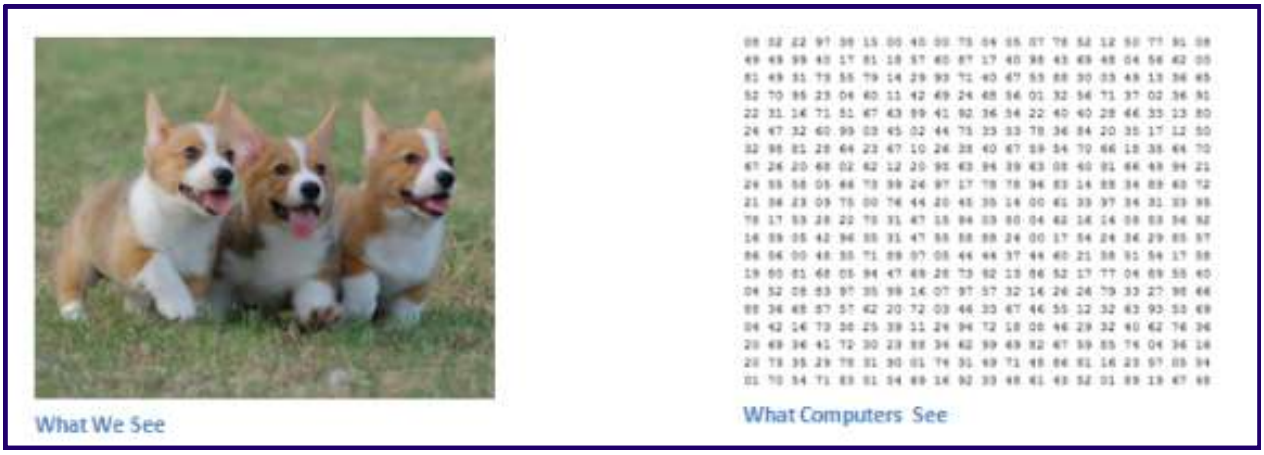


Abb. 7 Form of View: Using neural networks for video surveillance (2025) from https://www.videonet9.com/using-neural-networks-for-video-surveillance.html; DateAdd: 09.08.2025.

Aber wenn du dich jetzt fragst: „Dieses Bild zeigt zweistellige Werte, aber keine Nullen oder Einsen?", dann hast du recht. Unsere Computerprozessoren interpretieren diese Werte, indem sie sie in Nullen und Einsen umwandeln. Würden diese Werte tatsächlich als Nullen und Einsen dargestellt, wäre das eine sehr lange und unübersichtliche Anzeige. Tatsächlich kann jeder dieser zweistelligen Werte im obigen Bild als Pixel bezeichnet werden. Was ist also ein Pixel? Es ist die kleinste Einheit, die es ermöglicht, die Anzeige auf digitalen Bildschirmen zu erzeugen und zu steuern.

EINHEIT 5.3 JETZT LASS UNS EINE ÜBUNG MACHEN

- Nimm ein kariertes Heft.
- Nummeriere jedes Kästchen im Heft mit Zahlen zwischen 0 und 1.
- Je näher die Zahl in einem Kästchen an 1 ist, desto weißer ist das Kästchen.
- Je näher die Zahl an 0 ist, desto schwärzer ist das Kästchen.
- Bei der Zahl 0,5 malst du das Kästchen grau aus.

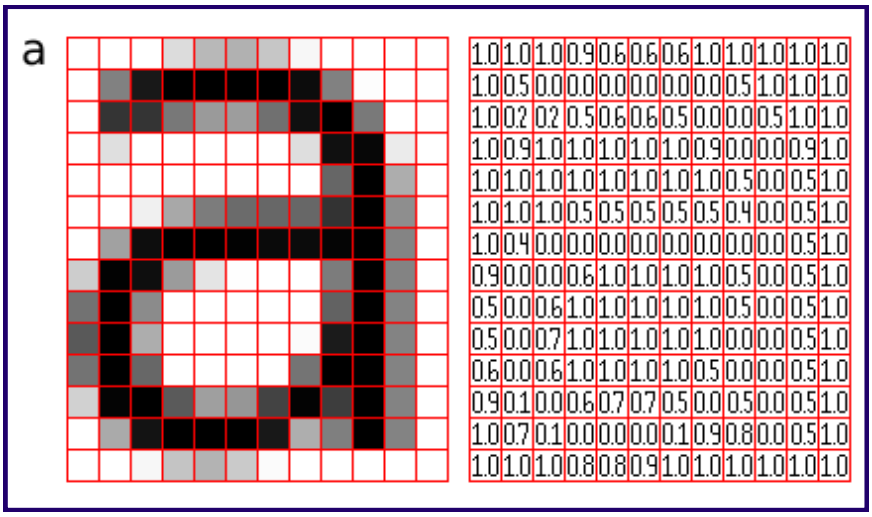


Abb. 8 How computer see pixel. (2025) from https://medium.com/@meriyananjalika99/from-pixels-to-predictions-getting-started-with-cnns-convolutional-neural-networks-e3f84781cc1e; DateAdd: 09.08.2025.

Wenn du sorgfältig in einem solchen Heft zeichnest, erscheint dir das Bild aus der Entfernung wie eine normale Zeichnung. Probier das für das Bild oben aus. Geh ein paar Meter vom Bildschirm weg – das Bild wirkt dann normaler. Tatsächlich zeigen Computer- und Fernsehbildschirme Bilder und Videos auf genau diese Weise – mit Millionen winziger Kästchen. Um Bilder zu speichern, werden die Zahlen aus dem karierten Heft im Speicher abgelegt. Bei Bildern mit schlechter Qualität, die wir als „verpixelt“ bezeichnen, kannst du die einzelnen Kästchen dieses karierten Hefts erkennen.

EINHEIT 5.4 EIN COMPUTER-VISION-PROZESS BESTEHT IM WESENTLICHEN AUS FÜNF SCHRITTEN:

1. Bild-/Datenaufnahme – Die Rohdaten werden erfasst.
2. Bild-/Datenaufbereitung – Die Daten werden für die Verarbeitung vorbereitet.
3. Auswahl des KI-Modells – Das passende Modell wird bestimmt.
4. Training des KI-Modells – Das Modell wird mit den Daten trainiert.
5. Interpretation der Ergebnisse – Die Ausgaben werden analysiert und genutzt.

Die Bilder, die für eine einfache Anwendung zur Erkennung handgeschriebener Ziffern verwendet wurden, sind unten dargestellt. Diese Bilder wurden in ein computerfreundliches Graustufenformat umgewandelt und für das Training der KI genutzt. Die Testergebnisse zeigen nach dem Training eine Genauigkeit von 99 %.

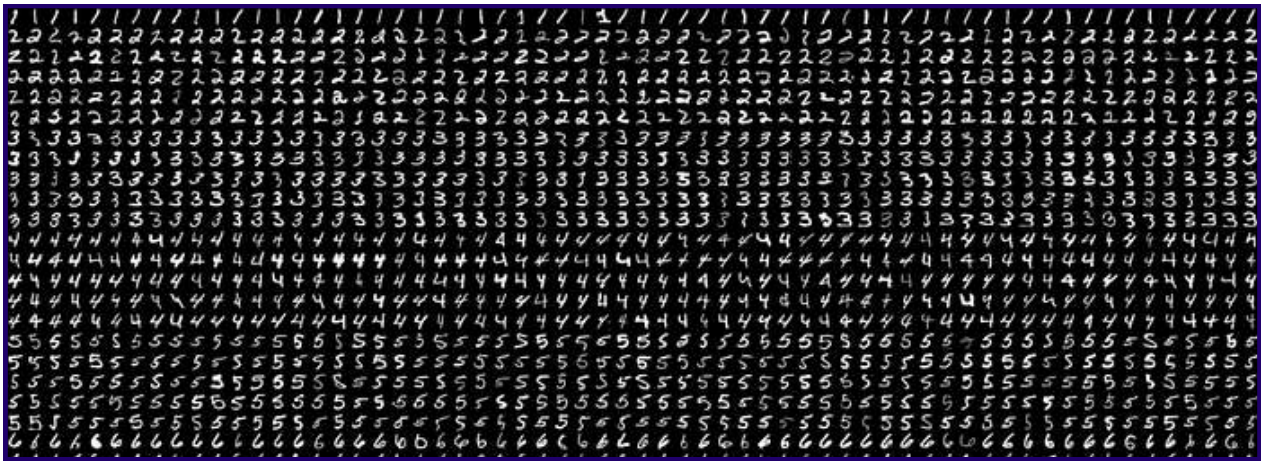


Abb. 9 Kaggle-Dataset (2025) from www.roboflow.com; DateAdd: 09.08.2025.

Roboflow ist eine umfassende KI-Plattform, die speziell für die Entwicklung von Computer-Vision-Projekten entwickelt wurde. Sie vereinfacht den Prozess der Erstellung, des Trainings und der Bereitstellung von Computer-Vision-Modellen mit Bild- und Videodaten. Dank einer benutzerfreundlichen Oberfläche eignet sie sich sowohl für Einsteiger*innen als auch für erfahrene Entwickler*innen.

EINHEIT 5.5 REGISTRIERUNG BEI ROBOFLOW

- Gehe auf https://roboflow.com/
- Klicke oben rechts auf Sign In
- Bei vorhandenem Konto: Zugangsdaten eingeben und einloggen
- Bei Erstnutzung: Continue With Google wählen (praktischste Option)
- Gewünschtes Google-Konto auswählen
- Erforderliche Berechtigungen erteilen
- Nach dem Einloggen auf NEW PROJECT klicken
- Projektdaten eingeben und Projekt erstellen

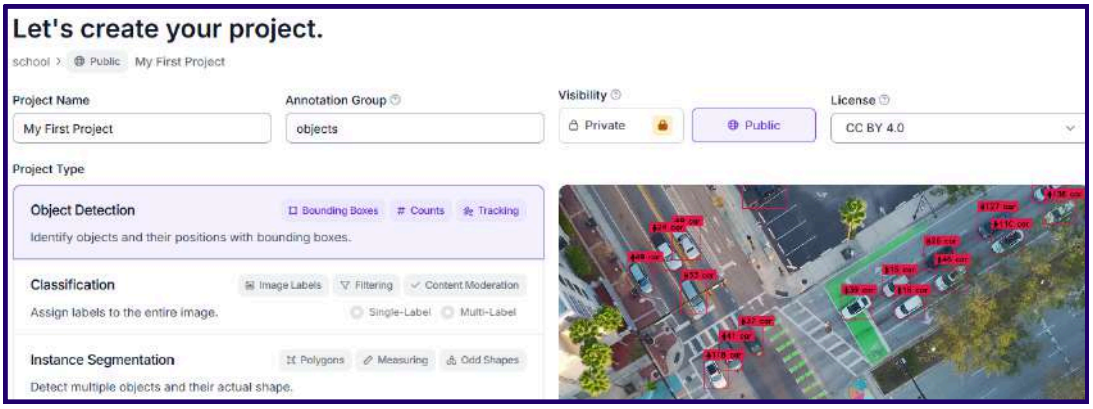


Abb. 10 Kaggle-Dataset (2025) from www.roboflow.com; DateAdd: 09.08.2025.

Auf der neuen Seite fügst du nun die Trainings-, Validierungs- und Testdaten für deine KI hinzu. Diese Daten nennen wir von nun an Dataset. Falls du kein eigenes Dataset hast, kannst du auf vorgefertigte Datensätze im RoboFlow-Universe-System zurückgreifen (https://universe.roboflow.com). Auch Kaggle bietet eine Vielzahl an fertigen Datensätzen (https://www.kaggle.com/).

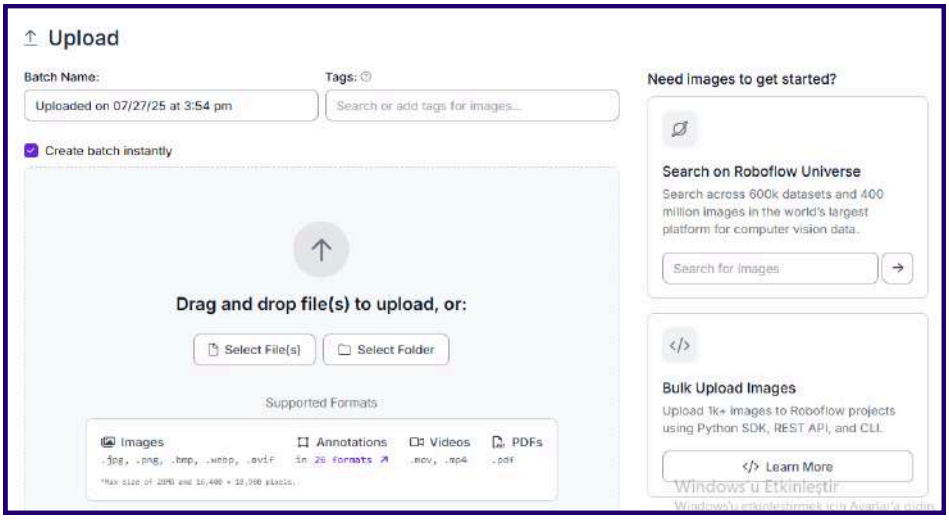


Abb. 11 Roboflow (2025) from www.roboflow.com; DateAdd: 09.08.2025.

In diesem System arbeiten wir mit einem vorgefertigten Dataset. Daher nutzen wir das Dataset unter https://universe.roboflow.com/roboflow-58fyf/rock-paper-scissors-sxsw über Roboflow. Nachdem du diese Adresse aufgerufen hast, klicke auf den Button „Dataset herunterladen“. Bevor du dies tust, musst du dich jedoch für ein KI-Modell entscheiden. In dieser Anwendung verwenden wir das YOLOv11-KI-Modell, das sich besonders gut für die Objekterkennung eignet. Wenn du das GitHub-Profil des entsprechenden Algorithmus unter https://github.com/ultralytics/ultralytics besuchst, findest du dort verschiedene vorgefertigte Gewichtungen sowie weiterführende Informationen. Einen YOLOv3-Test findest du unter https://v-iashin.github.io/detector.

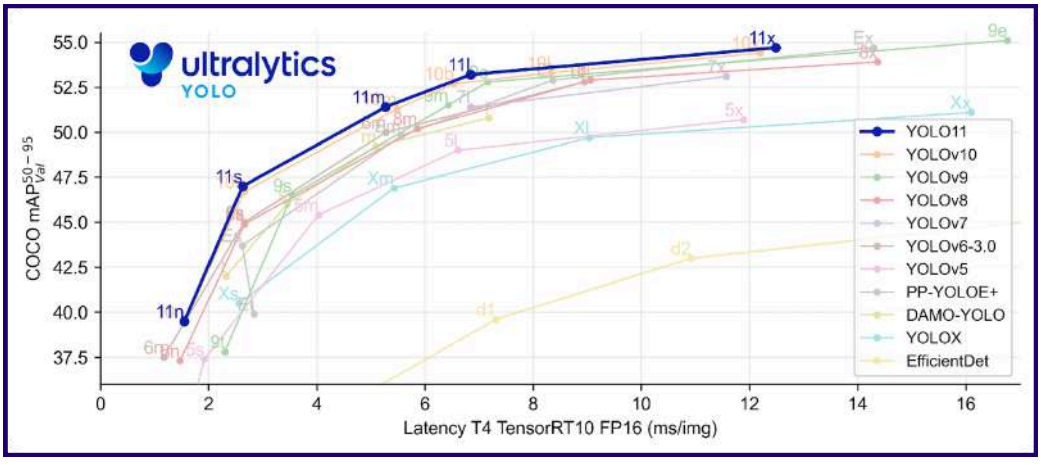


Abb. 12 Dataset (2025) from www.roboflow.com; DateAdd: 09.08.2025.

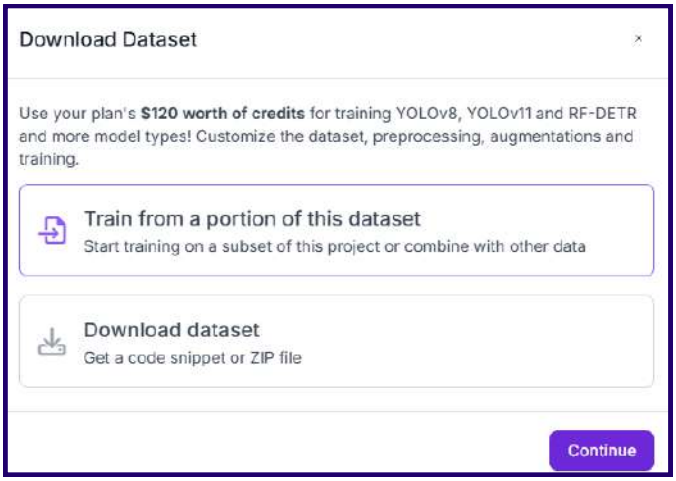


Abb. 13 Dataset (2025) from www.roboflow.com; DateAdd: 09.08.2025.

Nachdem du die Daten im Dataset beschriftet und überprüft hast, klicke auf den Button „Dataset herunterladen“. Wähle dann das gewünschte Modell und die Online-Methode aus. Anschließend erhältst du den Python-Code, der von der Website bereitgestellt wird.

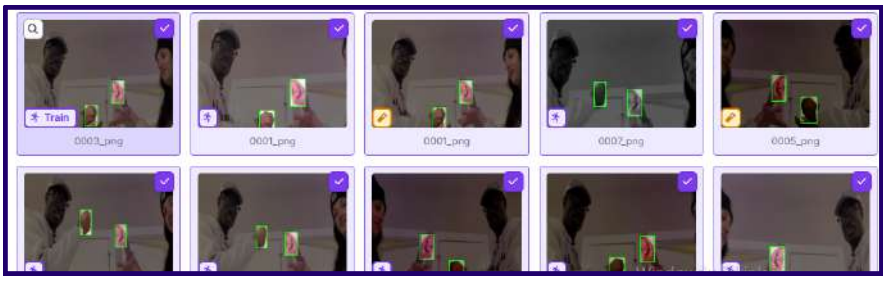


Abb. 13 Dataset (2025) from www.roboflow.com; DateAdd: 09.08.2025.

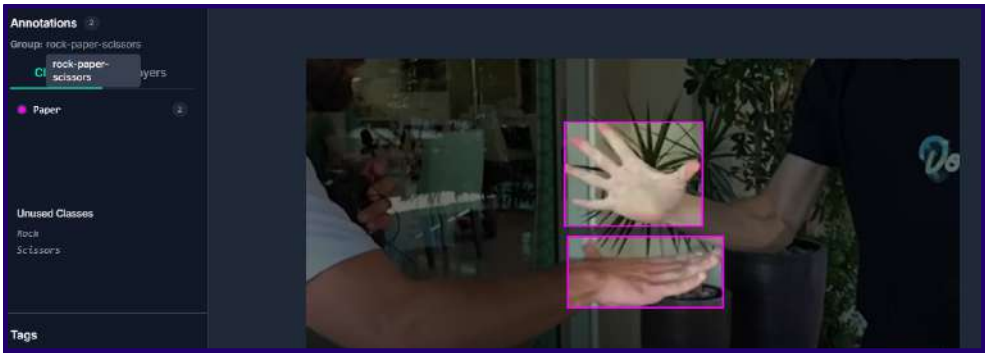


Abb. 15 Cut-off (2025) from www.roboflow.com; DateAdd: 09.08.2025.

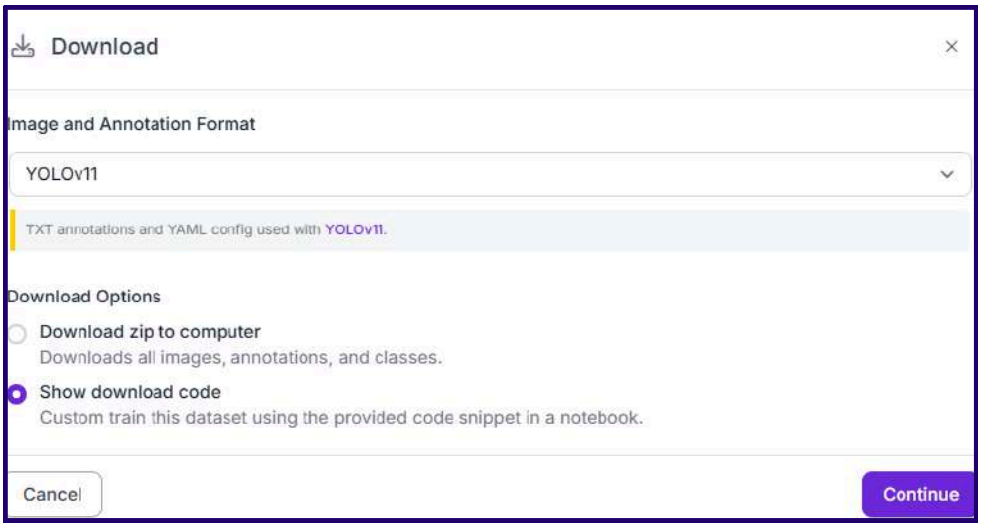


Abb. 16 YOLOv11 (2025) from www.roboflow.com; DateAdd: 09.08.2025.

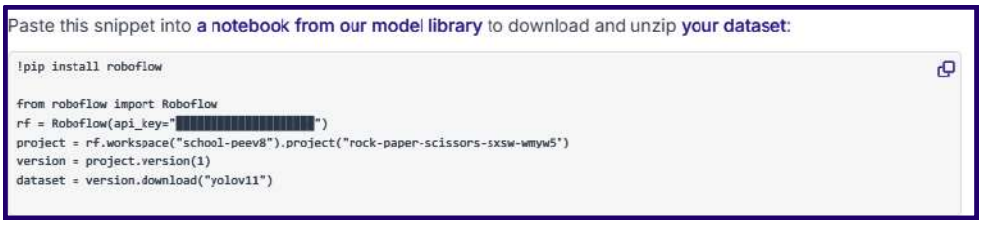


Abb. 17 Snippet (2025) from www.roboflow.com; DateAdd: 09.08.2025.

Arbeiten mit YOLOv11 in Google Colab

Im nächsten Schritt öffnen wir den Code für den YOLOv11-Algorithmus, der auf Google Colab läuft und von Roboflow für uns vorbereitet wurde: https://blog.roboflow.com/yolov11-how-to-train-custom-data/

Besuche folgenden Link und folge den dortigen Anweisungen: https://colab.research.google.com/github/roboflow-ai/notebooks/blob/main/notebooks/train-yolo11-object-detection-on-custom-dataset.ipynb?ref=blog.roboflow.com

Alternative: Ein Modell direkt auf Roboflow zu erstellen und zu nutzen, kann schneller und einfacher sein. YOLOv11 liefert jedoch bessere und genauere Ergebnisse.

Convolutional Neural Networks (CNNs) nutzen Faltungen (Convolutions) – eine Bildverarbeitungstechnik – um Merkmale in einem Bild zu erkennen. Dabei gleitet ein Fenster über jedes Pixel des Bildes, um die charakteristischen Eigenschaften zu lernen. Die aus den Faltungen gewonnenen Informationen werden anschließend von einem neuronalen Netz verarbeitet. Es gibt zahlreiche CNN-Architekturen, darunter R-CNN, Mask R-CNN und Fast R-CNN.

Auch die YOLO-Modellfamilie hat die Welt der Computer Vision maßgeblich beeinflusst. YOLO wurde 2014 von Joseph Redmon eingeführt und hat sich seitdem zu einem aktiven Forschungs- und Entwicklungsfeld entwickelt. Es wurde von einer breiten Community übernommen und von zahlreichen Entwickler*innen sowie Forschenden implementiert. Besonders YOLOv5 und YOLOv8, die vom Ultralytics-Team entwickelt und verfeinert wurden, treiben heute weltweit Produktionsmodelle zur Objekterkennung an.

Weitere Informationen zu YOLO findest du hier:
https://blog.roboflow.com/guide-to-yolo-models

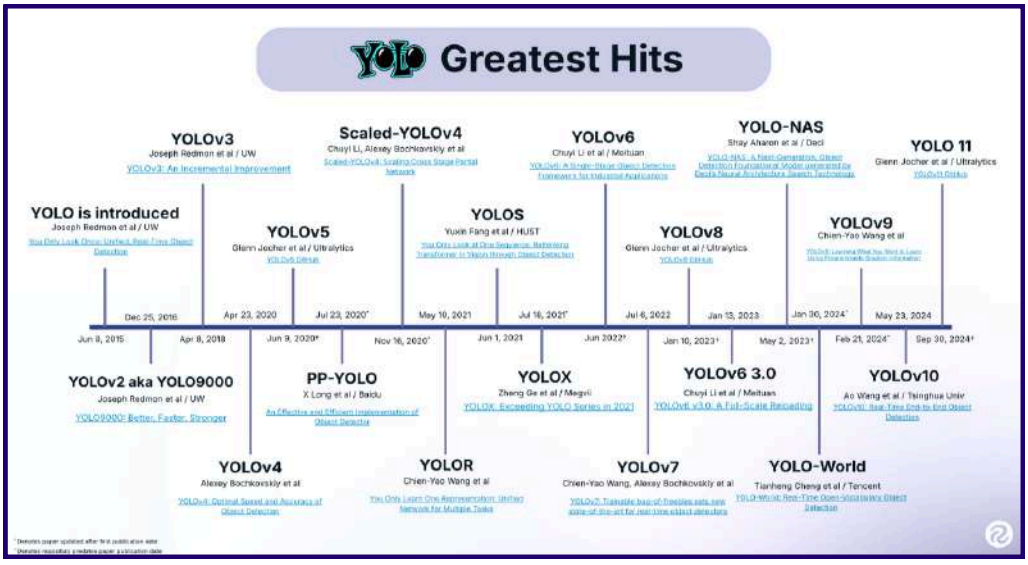


Abb. 18 YOLO-History (2025) from www.roboflow.com; DateAdd: 09.08.2025

Eine Übersicht der weltweit führenden Computer-Vision-Algorithmen findest du hier: https://leaderboard.roboflow.com/?ref=blog.roboflow.com

REFERENZEN

Literatur und Dokumentation

- Banerjee, Chayan & Fookes, Clinton & Karniadakis, George. (2023). Physics-Informed Computer Vision: A Review and Perspectives. 10.48550/arXiv.2305.18035.

- LeCun, Y. (2025). MNIST a Remote View of the Dataset from http://yann.lecun.com/exdb/mnist/. DateAdd: 09.08.2025.

- Mohite, Amruta & Kulkarni, Atharva & Chitnis, Rutwik & Mane, Swapnil & Asabe, Shubham. (2021). AI Inspection: Computer Vision For Visual Inspection. International Journal of Advance Research in Computer Science and Management. 7. 29.

- Object Recognition (2025). From https://viso.ai/product/computer-vision-parking-lot-occupancy-tutorial/ DateAdd: 10.08.2025.

- Using neural networks for video surveillance (2025) from https://www.videonet9.com/using-neural-networks-for-video-surveillance.html. DateAdd: 09.08.2025.

LINKS ZU LERNPLATTFORMEN

Eine detaillierte Übersicht über die weltweit führenden Algorithmen im Bereich Computer Vision findest du unter folgendem Link: https://leaderboard.roboflow.com/?ref=blog.roboflow.com

Roboflow ist eine umfassende KI-Plattform, die speziell für die Entwicklung von Computer-Vision-Projekten entwickelt wurde: https://roboflow.com/

PRAKTISCHE ÜBUNG

OBJEKTE-JÄGER – LEITFADEN FÜR LEHRKRÄFTE

Ziel der Aktivität

- Die Logik von Computer Vision verstehen
- Ein Objekterkennungsmodell mit Roboflow erstellen
- Die Prozesse der Datensammlung, Beschriftung, Modelltraining und -testung durchlaufen

Vorbereitung

- Computer oder Tablets mit Internetzugang
- Gruppen mit 4–5 Schüler*innen
- Smartphones (zum Fotografieren)
- Kostenloser Roboflow-Account
- Einfache Objekte (Buch, Bleistift, Wasserflasche etc.)
-

Dauer und Zeitplan

- Teil 1: Einführung in Computer Vision – 15 Minuten
- Teil 2: Fotos sammeln – 20 Minuten
- Teil 3: Beschriften – 25 Minuten
- Teil 4: Modelltraining – 20 Minuten
- Teil 5: Test und Wettbewerb – 20 Minuten

Ablauf

1. Kurze Einführung in Computer Vision.
2. SuS wählen zwei Objekte und machen 15–20 Fotos aus verschiedenen Blickwinkeln.
3. In Roboflow ein neues Projekt anlegen und Fotos hochladen.
4. Objekte durch Rahmen beschriften.
5. Modell trainieren und testen.
6. Gegenseitiges Testen der Modelle – Punkte vergeben.

OBJEKTE-JÄGER – LEITFADEN FÜR LEHRKRÄFTE

Gamifizierungsideen

Zeitrennen: Wer schafft es am schnellsten, Daten zu sammeln und zu beschriften?
Gemischter Test: Das Modell wird mit Objekten getestet, die es noch nie gesehen hat.
Fehlerjagd (Bug Hunt): Fälle, in denen das Modell Fehler macht, werden aufgespürt und gemeinsam besprochen.

OBJEKTE-JÄGER – ARBEITSBLATT FÜR SCHÜLER*INNEN

Was ist Computer Vision?

Computer Vision ist der Bereich der Künstlichen Intelligenz, der es Computern ermöglicht, Objekte mit Kameras oder Bildern zu erkennen und zu verstehen.

Deine Aufgaben

1. Wählt als Team zwei Objekte aus.
2. Macht 15–20 Fotos von jedem Objekt aus verschiedenen Blickwinkeln.
3. Öffnet ein Projekt in Roboflow und ladet die Fotos hoch.
4. Beschriftet die Objekte.
5. Trainiert euer Modell.
6. Versucht, die Objekte der anderen Teams zu erkennen.

Tipps zum Fotografieren

- Fotografiere aus verschiedenen Blickwinkeln.
- Probiere unterschiedliche Lichtverhältnisse aus.
- Fotografiere das Objekt sowohl aus der Nähe als auch aus der Ferne.

Anleitung zum Beschriften

Rahme das Objekt im Foto ein und schreibe seinen Namen dazu.
Beispiel: Bleistift, Radiergummi

My Observations

- Situations my model recognizes best
- Situations my model struggles with

REFLEXIONSFRAGEN

Nimm dir Zeit, um über das nachzudenken, was du in diesem Modul gelernt hast. Beantworte die folgenden Fragen in Ruhe:

- **Von Pixeln zur Wahrnehmung:** In diesem Modul hast du gelernt, dass Computer Bilder als Zahlenraster und Binärcode wahrnehmen – nicht als Formen und Farben. Wie verändert diese „numerische" Sichtweise dein Verständnis davon, wie eine KI zwei verschiedene Objekte unterscheidet, zum Beispiel einen Bleistift und einen Radiergummi??
- **Die Bedeutung von Datenvielfalt:** In der Aktivität „Objekte-Jäger" ging es darum, Fotos aus verschiedenen Blickwinkeln und unter unterschiedlichen Lichtverhältnissen aufzunehmen. Warum ist diese Vielfalt in deinem Trainingsdatensatz entscheidend dafür, dass die KI ein Objekt in der realen Welt zuverlässig erkennt?
- **Analyse von Genauigkeit / Fehlern:** In der Testphase warst du aufgefordert, Fälle zu identifizieren, in denen das Modell falsch lag – eine regelrechte „Fehlerjagd". Welche spezifischen Umgebungsfaktoren (wie Schatten oder Entfernung) haben es deinem Modell erschwert, korrekt zu erkennen?
- **Problemlösung in der realen Welt:** Computer Vision wird bereits in Bereichen wie Gesundheitswesen, autonomen Fahrzeugen und Landwirtschaft eingesetzt. Nachdem du nun dein eigenes Modell trainiert hast: Welches konkrete Problem in deiner Schule oder deiner lokalen Gemeinde könntest du mit dieser Technologie lösen?

QUIZ MIT MEHRFACHWAHLFRAGEN (eine richtige Antwort pro Frage)

1. Was ist das Hauptziel der Computer Vision?
a) Computern zu ermöglichen, wie menschliche Gehirne zu denken.
b) Maschinen zu befähigen, die Welt wie das menschliche Auge zu „sehen“ und zu interpretieren.
c) Große Datensätze in Datenbanken zu speichern.
d) Nur textbasierte Daten zu analysieren.

2. Welche der folgenden Anwendungen ist KEINE typische Anwendung von Computer Vision?
a) Erkennen von Verkehrszeichen in autonomen Fahrzeugen
b) Gesichtserkennungssysteme
c) Analyse von Nutzerpräferenzen in Produktempfehlungssystemen
d) Erkennung von Tumoren in medizinischen Bildern

3. Wie nennt man die Aufgabe der Computer Vision, bei der Ort und Art bestimmter Objekte in einem Bild bestimmt werden?
a) Bildklassifizierung
b) Objekterkennung
c) Bildsegmentierung
d) Merkmalsextraktion

4. Welche Herausforderung stellt sich für Computer-Vision-Systeme in der realen Welt?
a) Data scarcity and the diversity of this data.
b) The low-dimensionality and simple structure of visual data.
c) Consistency in illumination, position, and scale.
d) Models consistently achieving perfectly accurate results.

5. Woran erkennt man, dass ein Modell überangepasst (overfitted) ist?
a) Geringe Genauigkeit sowohl bei den Trainings- als auch bei den Testdaten
b) Sehr hohe Genauigkeit bei den Trainingsdaten, niedrige Genauigkeit bei den Testdaten
c) Hohe Genauigkeit sowohl bei den Trainings- als auch bei den Testdaten
d) Das Modell kennzeichnet jedes Bild gleich

Fazit & Ausblick

Aufbauend auf den Grundlagen des ersten Kurses bieten diese fortgeschrittenen Lernmaterialien Schüler*innen der Sekundarstufe einen tiefen Einblick in die praktische Anwendung Künstlicher Intelligenz. Die Inhalte überbrücken die Kluft zwischen Theorie und Praxis und führen die Lernenden Schritt für Schritt zur Entwicklung einfacher KI-Modelle mit Python und Scratch. Von der Erkundung mathematischer Logik hinter Entscheidungsprozessen in einem „KI-Escape-Room“ über das Trainieren von Machine-Learning-Modellen für SpielAbb.n bis hin zur Entwicklung von Computer-Vision-Anwendungen mit Roboflow – die Module bieten eine umfassende, praxisorientierte Lernerfahrung. Ein narrativer Ansatz mit den sympathischen Abb.n Aylin und Alara sorgt dafür, dass komplexe technische Konzepte wie neuronale Netze, K-Means-Clustering und Objekterkennung verständlich und spannend bleiben. Nach Abschluss dieses Kurses werden die Schüler*innen vom passiven Konsum zur aktiven Gestaltung von Technologie übergehen. Sie erwerben die technischen Fähigkeiten, um einfache intelligente Systeme zu programmieren, entwickeln kritisches Denken zur Bewertung von Modellgenauigkeit und gewinnen ethisches Bewusstsein für den verantwortungsvollen Einsatz von KI.

Wie geht es weiter?

Um die nachhaltige Integration dieser fortgeschrittenen Themen in den Schulunterricht zu unterstützen, wird das Projekt als nächstes ein „E-Toolkit für Lehrkräfte“ vorstellen. Diese Ressource bietet Lehrkräften detaillierte Unterrichts-pläne und methodische Leitfäden für eine effektive Vermittlung der KI-Inhalte. Darüber hinaus sind die Schüler*innen eingeladen, ihre neu erworbenen Fähigkeiten bei den kommenden „AI Future Explorers“-Wettbewerben unter Beweis zu stellen.

KÜNSTLICHE INTELLIGENZ: HANDS-ON
MIT PYTHON UND SCRATCH

Projekt Partner

FUTURE-STEM-HUB

UNIVERSITÄT DUISBURG ESSEN
Open-Minded

KÜTAHYA
İL MİLLÎ EĞİTİM MÜDÜRLÜĞÜ

tetra solutions

M&M profuture training

ETAP
ESCOLA PROFISSIONAL

Projektkoordinator
Universität Duisburg-Essen

E-Mail-Adresse
mustafa.bilgin@uni-due.de

Webseite
www.future-stem-hub.eu

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

FUTURE
STEM HUB