Baue deine eigene KI

Ein Leitfaden für Neugierige

Eine **eigene Künstliche Intelligenz (KI) oder ein KI-Tool** zu entwickeln, ist nicht nur etwas für Technik-Nerds. Es ist ein kreativer Weg, sich kritisch mit einer Technologie auseinanderzusetzen, die unsere Welt prägt. **Verständnis entsteht durch selber ausprobieren.**

Deine Idee. Deine KI.

KI ist nicht an ein spezifisches Thema gebunden: Zu Beginn steht also **deine Idee**. Frage dich selbst: Welches Problem oder Thema interessiert mich? Wie könnte ich dieses Problem lösen oder erkunden?
„Kann ich eine KI bauen, die meine Katze erkennt? Wie viele Bilder meiner Katze brauche ich dafür?"
„Kann ich eine KI bauen, die die Gesundheitskosten senkt? Wo entstehen hohe Kosten?“
Es kann eine **Spielerei**, das Überwinden einer **Alltagshürde** oder eine Idee sein, die die **Welt verbessert**.

⚠️ Du musst nicht die perfekte Idee finden oder dir zu viel Sorgen machen, ob diese wirklich realistisch ist. Wahrscheinlich verändert sich die Idee im Prozess sowieso. Darum: Ausprobieren und konkret werden.

Das eigenes KI-Abenteuer

Egal ob du ein:e Anfänger:in bist, der/die zum Spass experimentiert, oder jemand, der/die grössere KI-Anwendungen entwickeln will. Wähle den Weg, der am besten zu deinen Fähigkeiten und Zielen passt:

1. No-Code

No-Code-Tools sind Plattformen, die dir ermöglichen, digitale Produkte (Apps, Spiele, Automatisierungen) zu bauen, ohne selbst Code zu schreiben. Anstatt Programmiersprachen einzugeben, nutzt du Drag-and-Drop-Oberflächen, Formulare oder andere grafische Benutzeroberflächen, um Dinge zu erstellen. Es gibt unzählige Tools, die du erkunden kannst:

Tools wie **Teachable Machine** lassen dich einfache KIs trainieren, ohne zu programmieren. Du kannst zum Beispiel ausprobieren, wie viele Bilder deiner Katze nötig sind, bis die KI sie zuverlässig erkennt. Frage dich: Welche Muster erkennt die KI in den Bildern, um festzustellen, ob es deine Katze ist oder nicht?

Scratch ist ein weiteres Beispiel für eine No-Code-Lösung, die auf einer anfängerfreundlichen, blockbasierten Programmiersprache und Online-Plattform basiert. Sie hilft dir, Programmierkonzepte zu verstehen und Projekte zu erstellen. Schliesslich ermöglichen dir Plattformen wie **make** oder **n8n** Automatisierungen von intelligenten Arbeitsabläufen zu generieren und verschiedene Apps miteinander zu verbinden. Du kannst z.B. ein KI-Tool bauen, das dir aus deinen E-Mails direkt eine To-Do Liste erstellt (vorsicht mit privaten Daten)*

- \+ Einsteigerfreundlich, intuitiv, schnell, fördert das Verständnis von Konzepten
- \- Du bist oft auf das Tool beschränkt und kannst den Code nicht ändern
- \- Oft baust du keine KI, sondern brauchst KI, um etwas zu bauen.

2. Low-Code

Low-Code-Plattformen ermöglichen es dir, Apps, Websites oder sogar KI-Tools mit minimalem handgeschriebenem Code zu erstellen.
Du gibst deine Idee als Anleitung oder Prompt ein. So kannst du sehr einfache und schnelle Prototypen erstellen, die du durch das Hinzufügen oder Ändern von Code-Snippets anpassen kannst.

Oktober 2025, Version 1: Entstanden im Rahmen der KI Challenge 2025. ki-challenge.ch. POC: Fiona Könz, ETH AI Center, fiona.koenz@ai.ethz.ch

Erzeuge einen Prototyp deiner Idee mit **Replit**, **lovable**, **v0**, **cursor** oder in einer anderen Umgebung. Beschreibe deine Idee in einfachen Worten als Prompt. Je besser du sie beschreibst, desto besser wird das Ergebnis. Die Ausgabe besteht aus echten Codezeilen, die du ansehen (und bearbeiten) kannst.

+ Schnell KI-gestützte Apps, Bots oder Tools mit minimalem Code entwickeln.
+ Einblick in den generierten Code und allenfalls Anpassung.
- Nicht geeignet für komplexe Logik oder ungewöhnliche Workflows
- Kostenpflichtige und geschlossene Systeme. Dein Tool läuft oft nur auf der spezifischen Plattform.
- Debuggen und Anpassen des Codes schwierig, wenn Grundkenntnisse des Programmierens fehlen

3. Full-Code oder selber Programmieren (mit Hilfe!)

Kein schickes Drag-and-Drop oder Prompt-Fenster, sondern reiner Code vor deinen Augen. KI bzw. KI-Tools werden manuell mit **Programmiersprachen** wie Python und vielen anderen erstellt. Aber mal ehrlich: Niemand setzt sich heute hin und schreibt alles komplett von Grund auf. Auch in Full-Code-Umgebungen sind vorgefertigte **Codeblöcke** und **Anpassungen mit Hilfe von KI** (insbesondere Sprachmodellen wie z.B. ChatGPT) zum neuen Standard geworden. Dennoch ist **Wissen über die Programmiersprache** erforderlich, um deinen Code anzupassen und Fehler zu beheben.

Code-Editoren wie **Visual Studio Code** oder **Jupyter Notebook** ermöglichen es dir, Code zu schreiben, zu bearbeiten, zu speichern und auszuführen. Es gibt viele Open-Source-Bibliotheken (Sammlungen wiederverwendbarer Codeschnipsel), in denen du Codeblöcke oder sogar ganze vortrainierte KI-Modelle findest (z.B. Transformers). **Hugging Face** ist eine Plattform und Bibliothek, die vortrainierte KI-Modelle (Text, Bild und Video) anbietet. Für die Programmiersprache Python findest du Bibliotheken in verschiedenen Bereichen wie Webentwicklung (Django, Flask), Datenanalyse (pandas, NumPy), maschinelles Lernen (TensorFlow, scikit-learn, PyTorch), die du am Anfang deines Codes importieren und sofort verwenden kannst. Zusätzlich kannst du Plattformen wie weights and biases nutzen, um deine Trainingsexperimente mit verschiedenen Hyperparametern zu verfolgen und vereinfachte Vergleiche zwischen verschiedenen Modellen zu erstellen.

Kurz gesagt: Wenn du weisst, was du bauen willst, findest du Bausteine oder sogar vortrainierte KI-Modelle, die du in deinen Code-Editor einfügen kannst, ohne alles von Grund auf programmieren zu müssen. Du kannst dein Projekt mit git auf Plattformen wie GitHub speichern, teilen oder gemeinsam daran arbeiten. Wenn du mehr über Programmieren im Detail lernen willst, findest du alle Grundlagen im KI Kurs.

+ Volle Anpassung, Flexibilität und Präzision und Verständnis darüber, was passiert
+ Unmittelbarer Zugriff auf APIs (e.g. OpenAI)
- Grundwissen über Programmiersprachen erforderlich
- Zeitaufwand

Full-Code, der sich wie Low-Code anfühlt.

Die Grenzen sind nicht immer klar. Mit Fortschritten bei KI-Agenten wirken manche Tools wie No- oder Low-Code, sind aber tatsächlich Full-Code-Tools. Du kannst deinen Code vollständig anpassen oder ihn herunterladen, um ihn in deinem Editor zu bearbeiten. So kannst du schnell einen Prototyp erstellen und alle Anpassungen nutzen, die in Full-Code-Umgebungen möglich sind. Eine weitere Alternative sind Tools wie Replicate, die dir den Zugriff auf ein KI-Tool über eine API ermöglichen. Solche Hilfstools entwickeln sich derzeit schnell weiter und eröffnen neue Möglichkeiten. Die Grundkonzepte von KI bleiben jedoch gleich. Wenn du diese verstehst, kannst du schnell jedes Tool erlernen.

Und jetzt?

★ Probiere verschiedene Ansätze aus und versuche zu **verstehen und zu reflektieren**, was passiert.
★ Lerne die **Konzepte von KI**, um zu verstehen, wann du KI verwendest und wann du eine KI baust.
★ Frag dich: "Wieviel Programmier-Grundwissen ist nötig um zu verstehen wie Maschinen "lernen" und wie viel brauche ich um den Code selbst analysieren und ändern zu können."

Daten, Daten und nochmals Daten

Wenn du deine eigene KI baust, wirst du schnell merken, wie wichtig **Daten** sind. Du kannst:

1. Daten von öffentlichen Plattformen wie Kaggle oder Hugging Face beziehen.
2. Dein **eigenes Dataset** generieren (z. B. Bilder deiner Katze).
3. **APIs** (Application Programming Interfaces) nutzen. Dies ist im Grunde ein Schlüssel, um Daten von einer bestimmten Website zu erhalten (z. B. das Wetter von openweathermap).
4. **Web Scraping** (Daten von einer Website ziehen).
5. **Synthetische Daten** generieren, z. B. mit mathematischen Formeln.

Oft musst du deine Daten **bereinigen oder strukturieren**, bevor du sie verwenden kannst. Das kann ein zeitaufwendiger Prozess sein und sollte nicht unterschätzt werden.

Reflexion

Zitiere die Quellen (z. B. Daten, Plattformen), die du verwendest und die Menschen, mit denen du zusammenarbeitest (und gib an, was dein eigener Beitrag ist). Mit KI-Tools kannst du Dinge schneller und einfacher lösen. Für Menschen, die keine KI nutzen, mag das einschüchternd wirken oder wie Betrug erscheinen. Erkläre offen, welche Art von KI du verwendet hast, welche Hilfe du erhalten hast und was deine eigentliche Arbeit ist. Sei bescheiden, sei klug und inspiriere andere.

KI nutzen vs. bauen: Wo kommt KI in deinem Projekt zum Einsatz? Hast du ein KI-Tool entwickelt oder KI eingesetzt, um etwas anderes zu bauen (z. B. eine Webseite oder eine App)?
KI verantwortungsvoll zu nutzen ist ein wichtiges Thema. Doch das **Entwickeln von KI-Tools** eröffnet dir neue Möglichkeiten und erweitert deine Perspektive weit über das reine Anwenden hinaus.

Bei der KI Challenge geht es darum, eine eigene KI bzw. ein **eigenes KI-Tool** zu entwickeln. Das bedeutet jedoch nicht, dass du ein LLM wie ChatGPT selbst nachbauen musst. Je nach KI-Technologie kannst du KI auch über **APIs spezifisch einbinden**. Entscheidend ist, dass daraus ein **echtes Werkzeug** entsteht – etwas, das du anderen zur Verfügung stellen kannst, sodass auch sie davon profitieren.

Ethische Fragen: KI ist mächtig, aber Macht bringt Verantwortung mit sich. Sobald du die zugrunde liegenden Konzepte von KI verstehst, musst du die schwierigen Fragen stellen, wie zum Beispiel: Behandelt diese KI alle **fair**? Welche **Daten** habe ich verwendet? Wer ist vertreten? Wer **profitiert** und wer könnte ausgeschlossen oder sogar geschädigt werden? Verändert meine KI etwas zum Besseren? Wie können wir KI verantwortungsvoll im Wohl der Gesellschaft nutzen?

Wenn du deine eigene KI baust, folge deiner **Neugier**, vergiss aber nie deine **Werte**!

Jetzt bist du dran!

“Wer nicht scheitert, wird wenig erreichen.” - Bertrand Piccard

Lass dich nicht von der Angst zu scheitern zurückhalten. Es geht nicht darum, ein perfektes KI-Projekt zu bauen, sondern **auszuprobieren**, zu sehen, wo und warum du auch stecken bleibst, und **Unterstützung** zu suchen. Jeder “**AHA-Moment**” ist ein kleiner Erfolg und hilft dir auf deiner Reise.
Indem du deine eigene KI baust, beginnst du, die **Mechanismen** dahinter zu verstehen, und zwar nicht aus einem Schulbuch, sondern aus deiner eigenen Erfahrung. Du lernst, wie Maschinen „denken“. Du siehst, wie **Vorurteile** in Systeme gelangen und wie du **kritisch** beurteilen kannst, ob ein System **vertrauenswürdig** ist und wo seine Grenzen liegen. Der Wechsel vom passiven Konsumieren zum aktiven Gestalten kann dein Gespür für KI-Systeme schärfen und dich darin bestärken, dass du die KI steuerst, und nicht umgekehrt.

Sei neugierig. Stelle schwierige Fragen. Entwickle mit Verantwortung.
Das ist echte menschliche Intelligenz.

Oktober 2025, Version 1: Entstanden im Rahmen der KI Challenge 2025. ki-challenge.ch. POC: Fiona Könz, ETH AI Center, fiona.koenz@ai.ethz.ch

Glossar

Definitionen aus dem KI Kurs.

Algorithmus | Ein Algorithmus enthält eine detaillierte Anleitung zur Lösung eines spezifischen Problems. Jeder einzelne Schritt ist hierbei festgelegt, kann jedoch auf verschiedensten Datensätzen angewendet werden.

API | Du kannst dir das wie einen Schlüssel vorstellen, der deinem Programm Zugriff auf die Funktionen oder Daten eines anderen Betriebssystems oder eines anderen Dienstes gibt. Du setzt also deren Tool oder Daten bei dir ein.

Automatisierung | Automatisierung bedeutet den Einsatz von Software, Maschinen oder anderen Technologien, um Aufgaben und Prozesse selbstständig auszuführen, anstatt dass diese manuell durch Menschen erledigt werden. Dies kann aber ein starrer Ablauf sein, der klaren Regeln folgt und keine Intelligenz des Systems benötigt.

Daten | Daten sind eine Sammlung von Zahlen, Fakten, Wörtern, Beobachtungen oder anderen Informationen wie Bilder oder Tonspuren. Sie bilden die Grundlage, dass KIs lernen können.

Drag-and-drop | Computerfunktion, mit der grafische Elemente (z. B. Blöcke, Icons) durch Anklicken markiert und mit gedrückter Maustaste auf dem Bildschirm bewegt und so an anderer Stelle eingesetzt werden können.

KI-Agent | Als KI-Agent wird ein Software Programm bezeichnet, das mit seiner Umgebung interagieren, Daten sammeln und die Daten verwenden kann, um selbst bestimmte Aufgaben auszuführen. Menschen setzen die Ziele fest, aber der KI-Agent wählt selbst die besten Aktionen aus, die er ausführen muss, um diese Ziele zu erreichen.

KI-Tool | Werkzeug (meist Software), das Künstliche Intelligenz beinhaltet oder nutzt, um eine Aufgabe zu lösen. Die Regeln sind dabei nicht starr vorgegeben, wie bei der regelbasierten Automatisierung, sondern das System erkennt Muster aus Daten. Ein Tool ist wie ein Werkzeug, das anderen zur Verwendung weitergegeben werden kann.

Künstliche Intelligenz | Die Begriffe Künstliche Intelligenz und Maschinelles Lernen sind nicht das Gleiche, werden aber oft fälschlicherweise als Synonyme genutzt. Künstliche Intelligenz ist ein Oberbegriff für einen maschinellen Vorgang (softwaregesteuert), der in der Handlung einer menschlichen Intelligenz entspricht. KI beschreibt ein selbstlernendes System.

Maschinelles Lernen | Das Maschinelle Lernen bezieht sich auf statistische Algorithmen, welche aus Daten lernen, spezifische Probleme zu lösen. Maschinelles Lernen macht somit KI möglich.

Open-Source | Open Source bezeichnet Software, deren Quellcode öffentlich zugänglich ist, sodass jeder sie einsehen, nutzen, modifizieren und weitergeben kann.

Prompt | Ein Prompt ist eine Anweisung, Anfrage oder Eingabeaufforderung, die an ein KI-Modell gerichtet ist, um eine bestimmte Reaktion, Antwort oder Aktion zu erhalten. Es ist im Grunde der Befehl, den Sie einer Künstlichen Intelligenz geben, um sie zu steuern und das gewünschte Ergebnis zu erzielen, sei es ein Text, ein Bild oder die Lösung eines Problems.

Trainieren | Das Trainieren einer KI ist der Prozess, bei dem ein Modell mithilfe von Daten und Optimierung seine Parameter so anpasst, dass es Muster erkennt und verallgemeinerbare Vorhersagen treffen kann. Um zu überprüfen, ob ein Algorithmus "schummelt" und lediglich Daten auswendig lernt, oder ob er tatsächlich sinnvolle Muster erkennt welche sich auch auf neuen (unbekannten) Daten wiederfinden, wird der Datensatz häufig in drei Teile unterteilt: Trainingsdaten fürs Training, Validierungsdaten für eine erste Überprüfung und optimierung und zuletzt Testdaten für die erneute Überprüfung.

Überblick no-code, low-code und full-code

	No-Code	Low-Code	Full-Code
Für wen?	Anfänger:innen, non-coders	fortgeschrittene Anfänger:innen	Fortgeschrittene & Entwickler:innen
Coding skills nötig?	keine	Minimal, hilfreich	ja
Beispiele	Teachable Machine, Scratch, n8n, make	Replit, Lovable, Cursor	Python (und libraries: PyTorch, TensorFlow, scikit), JavaScript, HTML
Ideal für	Demos, ausprobieren	Schulprojekte, MVPs	Eigenes Projekt entwickeln
Individuelle Anpassung	kaum	etwas	unbegrenzt
Export, weiterverwendung	oft nicht möglich	oft beschränkt	unbeschränkt
Lernfokus	Konzepte verstehen	Konzepte anwenden	technisches Verständnis
Kosten	gratis Testversion, kostenpflichtige pro Versionen	gratis Testversion, kostenpflichtige pro Versionen	Oft open source, gratis