PerspektiveArbeit
Lausitz

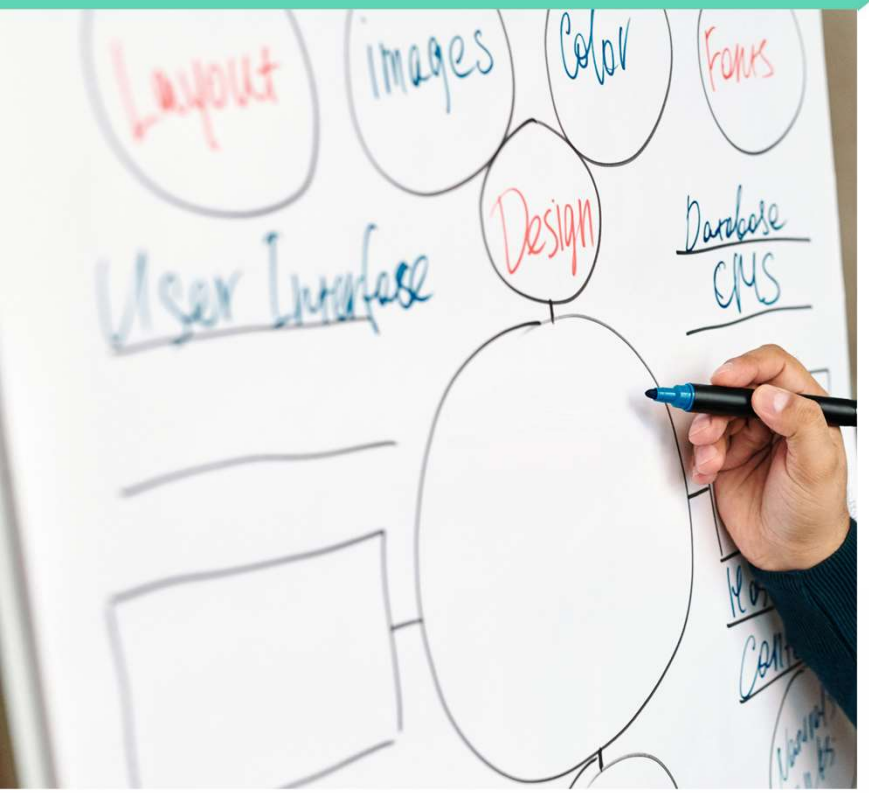
Orientierungshilfe für den Einsatz von vortrainierten Modellen in datenbasierten Assistenzsystemen

Gefördert durch:

Bundesministerium für Forschung, Technologie und Raumfahrt

Das zugrundeliegende Vorhaben wird mit Mitteln des Bundesministeriums für Forschung, Technologie und Raumfahrt unter den Förderkennzeichen 02L19C300 - 02L19C333 gefördert. Projektlaufzeit: 01.11.2021 - 31.10.2026

Einleitung



Problemstellung & Relevanz:

Der Einsatz von Softwarebibliotheken und spezieller Frameworks ist in der modernen Softwareentwicklung nicht mehr wegzudenken. Die hohe Komplexität und die umfassenden Funktionsumfänge von Softwarelösungen erfordern die Realisierung einzelner Funktionalitäten durch externe Software. Bei richtigem Einsatz kann der Entwicklungsaufwand stark reduziert werden. Ein Vorteil, der insbesondere bei KI-Lösungen zum Tragen kommt. Bei Open-Source-Lösungen spielen neben der Kostenersparnis auch die schnelle Integration von Erfahrungen durch den Zusammenschluss vieler Entwickler eine große Rolle. Es bestehen aber auch Risiken und Gefahren.

Zielgruppe:

Diese Orientierungshilfe richtet sich an Unternehmen, die ein datenbasiertes System im eigenen Unternehmen einführen möchten. Zielgruppen sind vorranging Softwareentwickler, aber auch Projektmanager und Führungskräfte zur Entscheidungsunterstützung.

Das erwartet Sie:

Dies ist eine praxisorientierte Orientierungshilfe für den Einsatz von vortrainierten Modellen (externen Softwarebibliotheken) in datenbasierten Assistenzsystemen und beantwortet die Fragen:

1. Wo kommt der Einsatz externer Softwaremodule/-bibliotheken infrage?
2. Warum nutzen wir externe Softwaremodule-/bibliotheken?
3. Welche Schwierigkeiten und Risiken können entstehen?

Software schnell selbst umgesetzt?

Maßgeschneiderte Unternehmenssoftware

Die Möglichkeiten durch Open Source, standardisierte Schnittstellen und jüngst durch generative KI klingen für viele Unternehmen verlockend. Noch vor einigen Jahren war es für „Software-fremde" Branchen nahezu undenkbar, eigene Software intern zu entwickeln. Mit dem Aufkommen von Low- und No-Code-Plattformen hat sich das geändert. Programmieren ist zwar nicht überflüssig geworden, aber deutlich zugänglicher. Anwendungen entstehen durch das Zusammensetzen vorgefertigter Bausteine – mit deutlich weniger eigenem Code. Für Unternehmen entsteht damit das Potenzial, eigene Softwarelösungen zu schaffen, insbesondere wenn IT-affine Mitarbeitende die Umsetzung übernehmen.

Info-Box: No-Code und Low-Code-Programmierung
Man versteht darunter visuelle Entwicklungsumgebungen, die eine flowbasierte Softwareentwicklung ermöglichen. Viele grundlegende Funktionen sind darin bereits als Bausteine, meist als nodes bezeichnet, implementiert und können über grafische Oberflächen miteinander verknüpft werden. Der Anteil an eigenem Programmcode reduziert sich deutlich: oft müssen Module nur parametrisiert statt programmiert werden. So lassen sich Anwendungen schneller und mit weniger Programmieraufwand umsetzen. Im Bereich Automatisierung, Vernetzung und Workflow-Management zählen unter anderem Mendix, Node-RED und n8n zu den bekanntesten Vertretern.

Vortrainierte Modelle als Fundament

Ein weiterer Treiber dieser Entwicklung sind frei nutzbare KI-Modelle, die bereits auf umfangreichen Datensätzen trainiert wurden. Unternehmen müssen dadurch keine eigenen KI-Systeme von Grund auf neu entwickeln, sondern können vorhandene Modelle adaptieren oder über Schnittstellen gezielt nutzen.
Damit verschiebt sich der Schwerpunkt von der klassischen Softwareentwicklung hin zur **Kompositionskompetenz**: das geschickte Kombinieren, Steuern und Anpassen bestehender Bausteine. Diese Fähigkeit wird für viele Branchen zu einem neuen Wettbewerbsvorteil.

Software schnell selbst umgesetzt?

Der Trugschluss

Trotz dieser neuen Leichtigkeit bleibt es Softwareentwicklung – mit allen Anforderungen und Risiken, die sie mit sich bringt. Auch niederschwellige Projekte benötigen Ressourcen, Planung und technisches Verständnis, gerade mit Blick auf Gesamtsystem und Unternehmensstrategie.
Projekte können Zeitpläne sprengen oder in der Funktionalität hinter den Erwartungen zurückbleiben, insbesondere wenn Software nicht zur Kernkompetenz des Unternehmens gehört.
Die Versuchung ist groß, weil viele Plattformen fast „fertige" Lösungen versprechen. Doch gerade bei unternehmensspezifischen oder datenschutzkonformen Anwendungen stecken die Herausforderungen im Detail.
Die scheinbare Einfachheit kann schnell zu unerwartetem Aufwand führen. Hinzu kommt der aktuelle Hype um sogenanntes Vibe-Coding – also die Übergabe von Programmieraufgaben an Systeme über natürliche Sprache. Doch auch hier gilt: Ein Sprachmodell kann nur umsetzen, was im Prompt beschrieben ist. Je komplexer die gewünschte Anwendung, desto detaillierter muss die Anweisung sein. Das vermeintlich mühelose Entwickeln birgt also ebenfalls seine Grenzen.

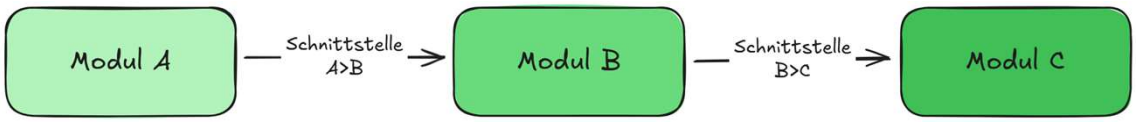
EU AI Act

Der EU AI Act ist ein Rechtsakt der EU zur Regulierung von künstlicher Intelligenz und stellt weltweit die erste derart umfassende KI-Regulation dar. Es wird regulatorisch zwischen Anbieter, Anwender und Importeure unterschieden und Systeme werden in 4 Risikostufen (minimal, begrenzt, hoch, inakzeptabel) eingestuft.
Durch die Fokussierung dieses Dokuments auf Empfehlungs- und Assistenzsysteme als interne Softwarelösung sollten die meisten Lesenden unter die Kategorie Anwender mit begrenztem Risiko fallen. Damit einher gehen bei begrenztem Risiko Transparenz- und Informationspflichten. Die KI-Nutzung muss für den Nutzenden klar ersichtlich sein (Hinweis im Interface, Schulung) und als Systementwickler besteht zusätzlich zum klassischen Anwender inhaltlich die Verantwortung für die Funktionsweise und korrekte Anwendung.
Die Einstufung des Systems muss als erster Schritt klar und nachvollziehbar dokumentiert werden und bei einer höheren Einstufung müssen entsprechend auch weitere Pflichten und Maßnahmen berücksichtigt werden.

Der Entwicklungsprozess für datenbasierte Anwendungen

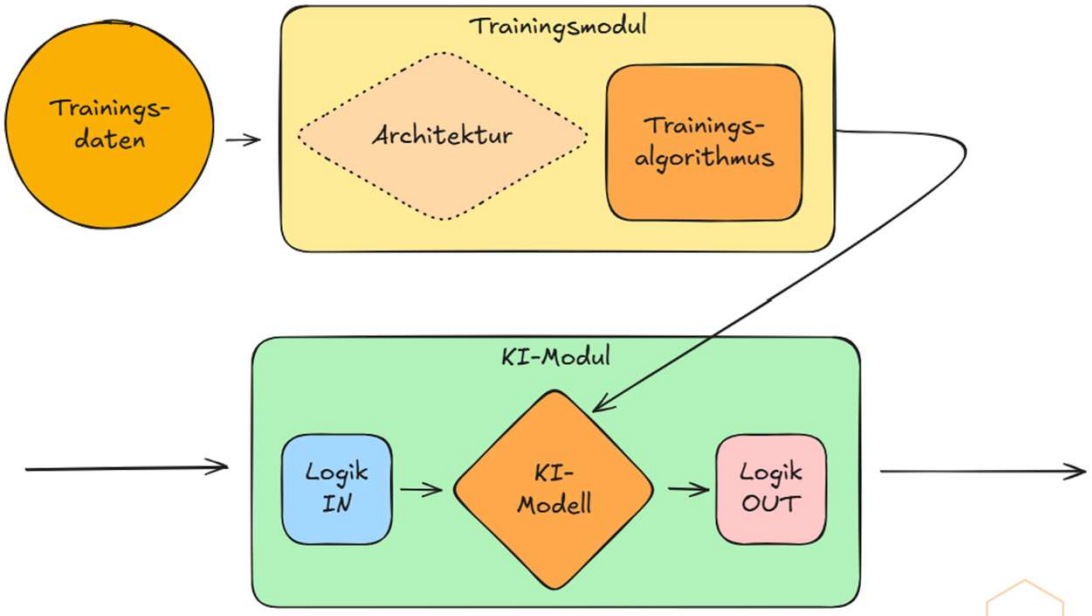
1.1 Softwarekomponenten

Die meisten Softwareanwendungen lassen sich in Bestandteile unterteilen, die eine einzelne Funktion erfüllen. Diese Softwaremodule interagieren über Schnittstellen miteinander.



Softwaremodule interagieren über Schnittstellen

KI-basierte Softwaremodule beinhalten einen Prozessschritt, der von einem KI-Modell ausgeführt wird, kombiniert mit entsprechender Logik, um Ein- und Ausgaben zwischen den Modulen zu gewährleisten.
Bei der Integration von KI-basierten Softwaremodulen in eine Anwendung unterscheidet sich der Entwicklungsprozess erheblich von der traditionellen Softwareentwicklung. Neben dem Aufwand, die Software zu entwickeln, kommen noch die aufwendige Erstellung eines Trainingsdatensatzes sowie das ressourcenintensive Training hinzu, sofern kein vortrainiertes Modell für die eigene Anwendung zur Verfügung steht.



KI-basierten Softwaremodule mit Trainingsmodul

Der Entwicklungsprozess für datenbasierte Anwendungen

1.2 KI-basierte Softwaremodule

Ein KI-basiertes Softwaremodul besteht aus fünf wesentlichen Komponenten, die in einem strukturierten Entwicklungsprozess aufeinander aufbauen:

Komponente	Beschreibung
Softwarebibliothek für Training und Modelldefinition	Grundlegende Software-Frameworks wie TensorFlow, PyTorch für Modellentwicklung
Definition des KI-Modells	Architektur des neuronalen Netzwerks, Layer-Struktur, Aktivierungsfunktionen
Trainingsdatensatz	Aufbereitete und annotierte Daten für das Training des Modells
Trainingsalgorithmus und -parameter	Optimierungsverfahren, Lernrate, Batch-Größe, Regularisierung
Trainierte Gewichte	Finales trainiertes Modell mit optimierten Parametern für Inferenz

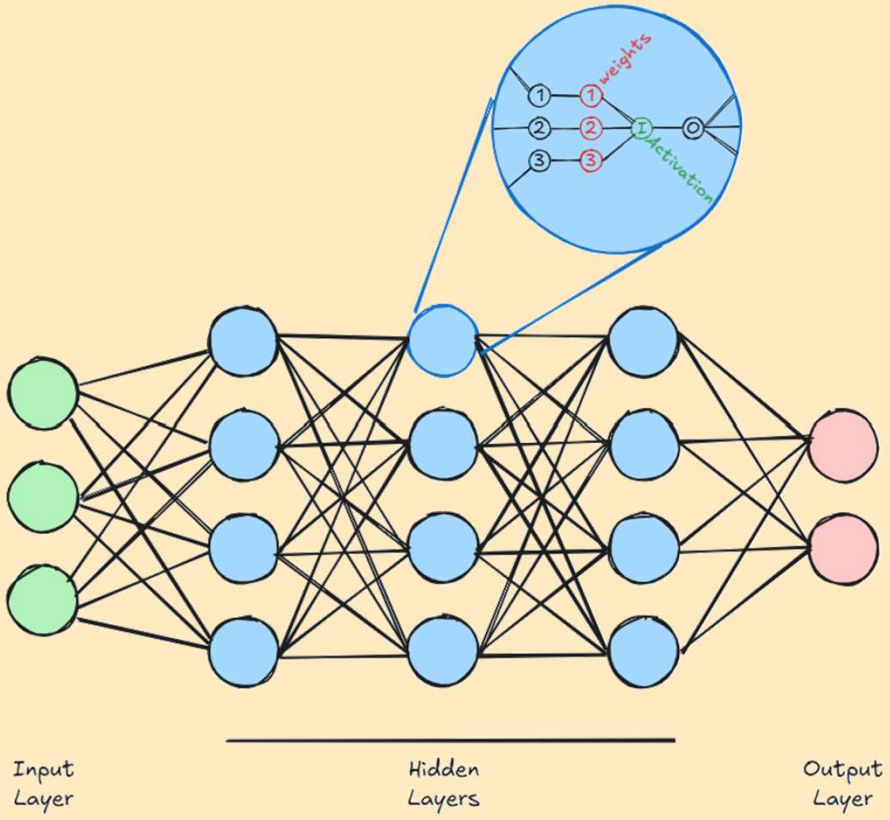
Jede Komponente erfüllt spezifische Funktionen im gesamten Lebenszyklus. Der Entwicklungsprozess kann in mehrere Schritte zerlegt werden, die aufeinander aufbauen und meist sequenziell durchgeführt werden.

Schritt	Phase	Beteiligte Komponenten
1	Framework-Setup	Softwarebibliothek
2	Datensammlung und -vorbereitung	Trainingsdatensatz
3	Modellarchitektur entwerfen	Definition des KI-Modells
4	Training konfigurieren	Trainingsalgorithmus und -parameter
5	Training durchführen	Trainierte Gewichte
6	Modell validieren	Alle Komponenten
7	Modell einsetzen (Inferenz)	Alle Komponenten

Der Entwicklungsprozess für datenbasierte Anwendungen

Info-Box: Neuronale Netze

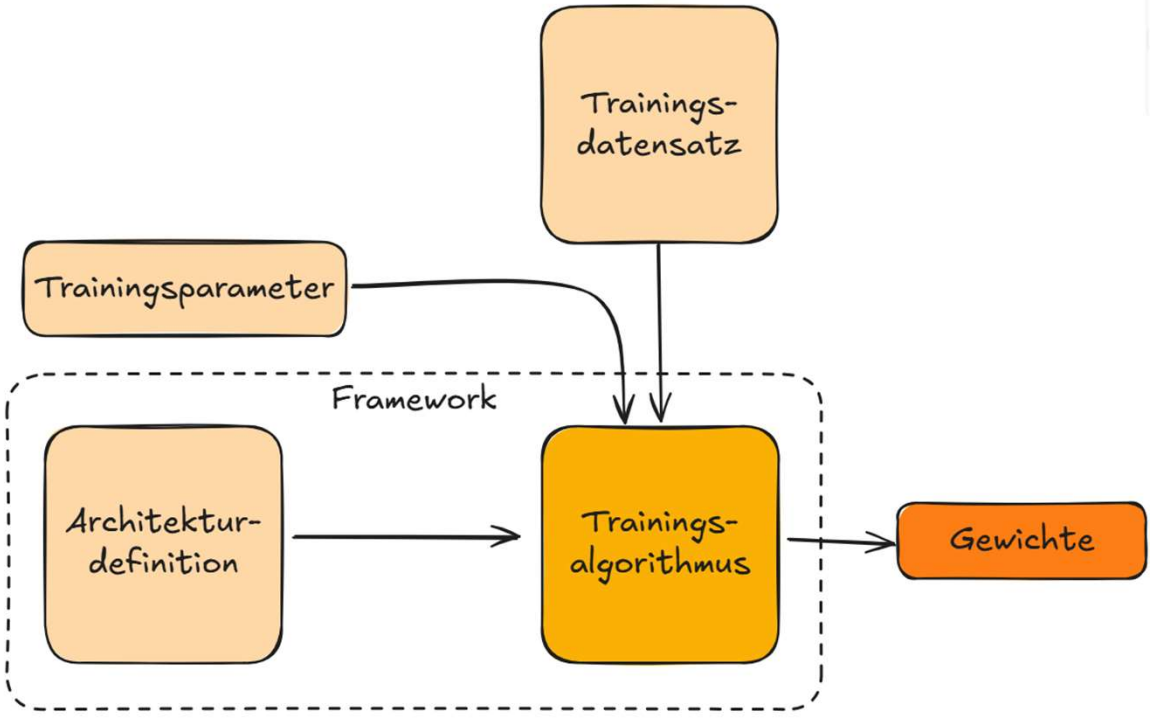
Das Rückgrat des maschinellen Lernens bilden neuronale Netze. Diese berechnen aus einer Eingabe an den Neuronen in einem **Input Layer** (dt.: Schicht) eine Ausgabe an ihrem **Output Layer**. Dies geschieht durch das wiederholte Berechnen und Verrechnen von Zwischenergebnissen innerhalb der **Hidden Layer**.
Als Architektur wird der Aufbau dieses Netzwerks mit der Anzahl an **Schichten** und **Neuronen** bezeichnet. Diese Architektur bleibt in der Regel während des Lernens unverändert. Angepasst werden hingegen die **Gewichte**, die innerhalb jedes Neurons zur Berechnung der Zwischenergebnisse genutzt werden.
Der Begriff KI-Modell soll für die Kombination aus Architektur und Gewichten stehen.



Neuronales Netz

1.3 Trainingsphase

Die Trainingsphase verleiht dem maschinellen Lernen seinen Namen. In dieser Phase findet der Trainingsalgorithmus selbstständig die Gewichte, die das durch den Trainingsdatensatz repräsentierte Problem für eine gegebene Architektur am besten lösen. Durch viele Iterationen erkennt es Muster und lernt, Vorhersagen zu treffen.



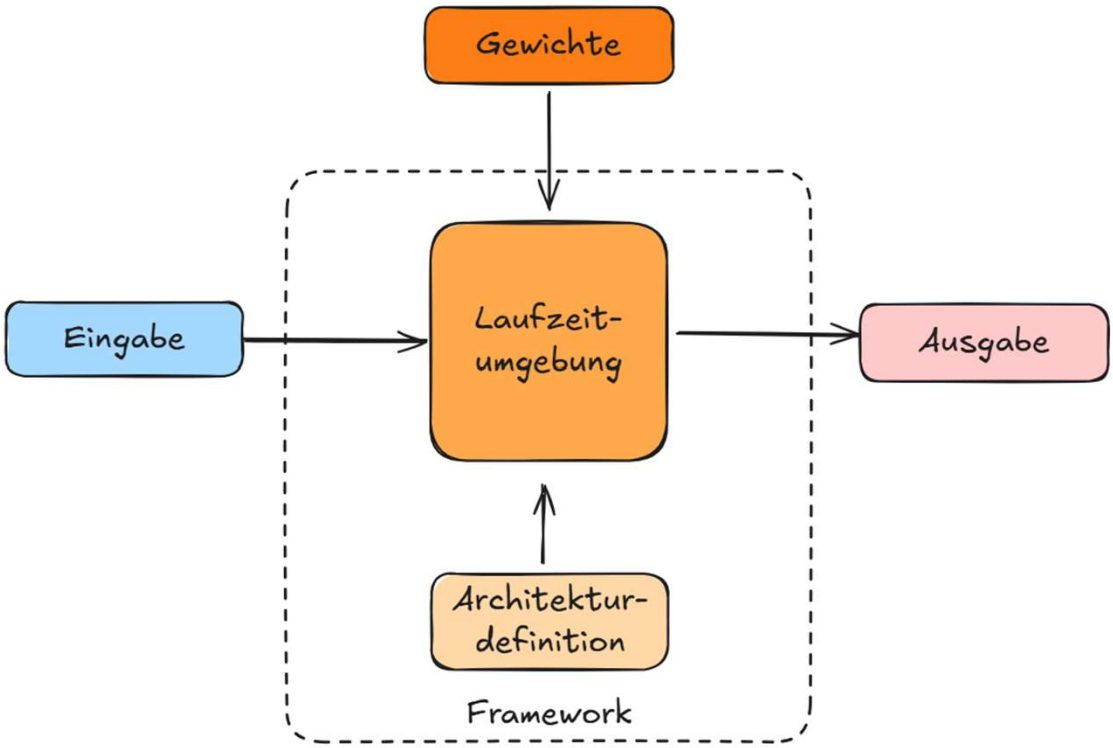
Trainingsphase

Eine gründliche Aufbereitung des Trainingsdatensatzes ist fundamental für diese Phase. Neben dem Training ist aber auch die Validierungsphase, in der das Modell mit den antrainierten Gewichten mit einem separaten Datensatz, der nicht zum Training verwendet wurde, überprüft wird, zu berücksichtigen.

Die gesamte Trainingsphase erfordert eine sorgfältige Planung, Durchführung und Auswertung. Es sind hierfür Erfahrung und Expertenwissen wichtig, um die Ergebnisse zu interpretieren und entsprechende Anpassungen vorzunehmen.

1.4 Laufzeitumgebung

Ist das Training abgeschlossen, stehen alle Komponenten bereit für den Einsatz. Diese Phase heißt je nach Situation, Deploymentphase, Inferenzphase, Testphase etc. und mithilfe der Architektur und der Gewichte wird eine Eingabe innerhalb einer Laufzeitumgebung verarbeitet und das Ergebnis zurückgegeben.



Inferenzphase

Für das Ausführen der Anwendung sind der Trainings- und Validierungsdatensatz, sowie der Trainingsalgorithmus nicht notwendig, auch kann sich das zugrundeliegende Framework zwischen Training und Anwendung unterscheiden. Ist ein Lebenszyklus vorgesehen, bei dem mithilfe der im Betrieb gesammelten Daten eine neue Version der Gewichte erstellt werden soll, sind alle Komponenten erforderlich.

Software in Bausteinen denken

2.1 Bewertung einzelner Bausteine

Bei mehrstufigen Prozessen kann jeder dieser Prozessschritte als ein zukünftiger Baustein im Softwaresystem verstanden werden. Es kann jeweils eine Bewertung stattfinden, ob und auf welche externen Software- und KI- Module zurückgegriffen werden kann. Für die Bewertung sollen zwei Fragen herangezogen werden:

- **Spezifik**
  Handelt es sich um einen Prozessschritt, der spezifisch für meine Aufgabe ist oder kommt er in vielen Problemen vor?

- **Regelhaftigkeit**
  Ist der Prozessschritt durch klassische regelbasierte Datenverarbeitung einfach lösbar oder profitiert er von maschinellem Lernen?

Daraus ergeben sich vier Arten von Bausteinen.

	regelhaft	**nicht regelhaft**
nicht spezifisch	**Standardkomponente** (z.B. Signalfilterung, Normalisierung)	**Standardmodell** (z.B. Bildklassifikation, Objekterkennung, Spracherkennung)
spezifisch	**Komponente** (z.B. Plausibilitätscheck, Grenzwertprüfung, Unternehmensprozesse)	**Modell** (z.B. personalisierte Empfehlungen, Vorhersage von Maschinenausfällen)

Jeder dieser Bausteine erfordert naturgemäß unterschiedliche Planungs-, Umsetzungs- und Qualitätssicherungsmaßnahmen.

Auch wenn nur einer der Prozessschritte (Baustein) durch ein KI-Modell durchgeführt wird, ist dieser essenziell für die Gesamtanwendung. Diese wäre ohne den Einsatz von maschinellem Lernen fast unmöglich, oder nur durch Mehraufwände im Prozess möglich.

2.2 Externe Komponenten

Für jeden der Schritte im Lebenszyklus gibt es Möglichkeiten, auf externe Ergebnisse bzw. Quellen, sowie Dienstleistungen zuzugreifen.

Schritt	Phase	Beteiligte Komponenten
1	Framework-Setup	KI Frameworks
2	Datensammlung und -vorbereitung	Öffentliche / Kommerzielle Datensätze
3	Modellarchitektur entwerfen	Öffentliche / Kommerzielle Modellearchitektur
4	Training konfigurieren	Trainingsbibliotheken und Best Practices
5	Training durchführen	Vortrainiertes Modell

Kommerzielle Softwareangebote werden zunehmend als Cloud-basierte Dienste bereitgestellt, was mit spezifischen Vor- und Nachteilen einhergeht.

Freie Software kann eine Grundlage für den Softwareeinsatz auf der eigenen Hardware oder im eigenen Produkt sein. Wer bei freier Software oder Open Source nur an Ersparnis durch den nicht existierenden Preis achtet, denkt zu kurz. Die Möglichkeit, die Software nach eigenen Maßstäben einzusetzen, erweitert die Einsatzgebiete und verhindert Abhängigkeiten von einzelnen Herstellern - besonders bei Open Source durch den zur Verfügung stehenden Source Code. Die Verfügbarkeit des Source Codes ermöglicht das Überprüfen der Funktion, sowohl von einem selbst als auch von der Open Source Community.

Info-Box: Vortrainierte Modelle
Ein vortrainiertes Modell setzt sich aus der Definition und den Gewichten zusammen. Hier stellen Forschungsinstitute und Forschungsgruppen teilweise ihre Resultate zur Verfügung. Auch Firmen gewähren Zugriff auf manche der Komponenten, entweder auf kleinere Varianten der kommerziell angebotenen Modelle oder Modelle, die Teil einer umfangreicheren kommerziellen Plattform sind.

2.3 Open Source und Lizenzen

Der Austausch von Software wird durch Lizenzen geregelt. Diese können das prinzipiell strenge Urheberrecht gezielt und unter Bedingungen aufweichen, um so anderen Zugriff auf die eigene Software zu gewähren. Für Software gibt es etablierte Software-Lizenzen, für KI-Gewichte greift man im Regelfall auf Lizenzen zurück, die bisher für Kunst, Musik und Texte angewandt wurden. Die meisten Lizenzen beinhalten einen Haftungsausschluss: wer die Software einsetzt, ist entsprechend selbst verantwortlich. Dafür werden je nach Lizenz unterschiedliche Freiheiten gewährt, die mit möglichen Pflichten einhergehen. Darunter fallen unter anderem

- Kommerzielle Nutzung
- Namensnennung
- Lizenzpflicht bei Veränderung

Die Gewichte werden oft unter GPL – oder der strengeren AGPL – oder einer nicht kommerziellen Lizenz veröffentlicht. Ähnliches gilt für Trainingsdatensätze. In diesem Fall muss die Wahl auf ein Modell fallen, dessen Code unter einer der Lizenzen MIT, BSD oder Apache veröffentlicht wurde und dessen Gewichte ebenfalls unter einer dieser Lizenzen oder unter CC stehen.

Info-Box: Wichtige Lizenzen

Software
- Affero General Public License (AGPL)
- Apache
- BSD
- General Public License (GPL)
- GNU
- MIT

Inhalte und Modellgewichte:
- Creative Commons (CC)

2.4 Open Source und Zukunftssicherheit

Beim Einsatz von freier Software müssen Vorkehrungen getroffen werden, um den langfristigen Produktiveinsatz zu gewährleisten. Freie Software bietet zwar eine kostenfreie Nutzung, jedoch besteht die Abhängigkeit zum Softwareanbieter, genau wie bei der Nutzung kostenpflichtiger Software.

Externe Software unterliegt einer gewissen Abhängigkeit vom Softwareanbieter. Regelmäßige (Sicherheits-)Updates sind notwendig – nicht nur für die Hauptsoftware, sondern auch für alle verwendeten Abhängigkeiten (Bibliotheken, Frameworks, Module). Da viele dieser Komponenten wiederum auf weiteren Bausteinen aufbauen, entsteht eine mehrstufige Update-Kette, die aktiv überwacht und gepflegt werden muss.

Ein Vorteil von Open Source Software liegt in der Möglichkeit, diese Abhängigkeit teilweise selbst zu kontrollieren. Unternehmen können bei ausreichenden internen Ressourcen den Code eigenständig anpassen, pflegen und erweitern. Anders als bei proprietärer Software, die ausschließlich durch den Hersteller weiterentwickelt wird, profitieren Open Source Projekte zusätzlich von einer aktiven Community, die Verbesserungen beiträgt, Fehler meldet und Sicherheitslücken schließt. Der offene Quellcode schafft zudem Transparenz und Flexibilität – im Gegensatz zu reinen Binärdateien (z. B. .exe/.dll unter Windows).

Bei der Auswahl eines geeigneten Open Source Repository muss man jedoch bedacht vorgehen. Folgende Punkte sollten berücksichtigt werden:

- Projektreife & Community-Aktivität
- Langfristige Wartbarkeit und Sicherheit
- Lizenztyp
- Kompatibilität und Schnittstellen
- Codequalität & Dokumentation
- Passung zur Unternehmensstrategie und aktuellen Aufgabenstellung

Open Source bedeutet nicht automatisch Unabhängigkeit: Die gewonnene Flexibilität erfordert eigene Kompetenzen in Wartung, Integration und Sicherheit. Prüfen Sie daher frühzeitig, ob die notwendigen personellen und technischen Ressourcen im Unternehmen vorhanden sind.

Impressum

Herausgeber:
Hochschule Mittweida

Redaktion/Autoren:
A. Engelsberger, T. Pfaff, J. Voigt

Bildnachweis:
Titelbild, Seite 2:pexels.com; Seite 5ff: eigene Darstellungen

1. Auflage 2026

PerspektiveArbeit
Lausitz

Informiert bleiben über

PAL-Angebote

PAL-Newsletter

PAL-Website

YouTube-Kanal

LinkedIn-Kanal

Gefördert durch:

Bundesministerium für Forschung, Technologie und Raumfahrt

Das zugrundeliegende Vorhaben wird mit Mitteln des Bundesministeriums für Forschung, Technologie und Raumfahrt unter den Förderkennzeichen 02L19C300 - 02L19C333 gefördert. Projektlaufzeit: 01.11.2021 - 31.10.2026