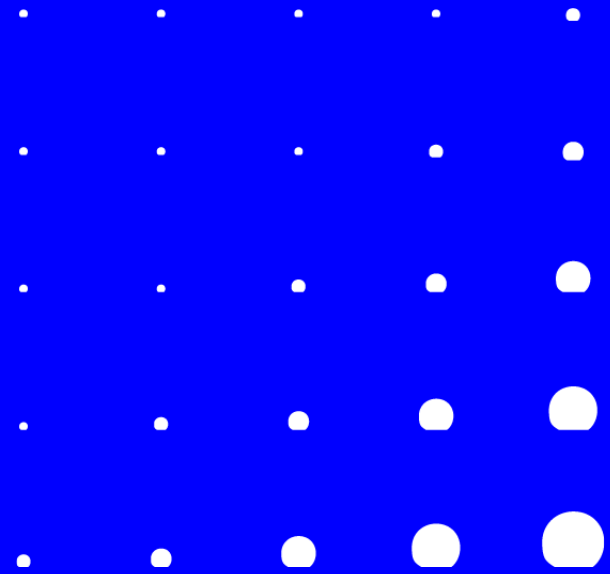
Technische Hochschule
mannheim

# Wie funktioniert ein LLM?

## Von Text zu Vorhersage: Tokenisierung, Transformer, Training und Inferenz



# Agenda

Technische Hochschule mannheim

1 Einordnung & Grundidee

2 Von Text zu Zahlen: Tokens & Embeddings

3 Der Transformer: Self-Attention & Co.

4 Training: Pretraining, Fine-Tuning, RLHF

5 Inferenz: wie Text entsteht

6 Fähigkeiten, Grenzen & Erweiterungen

# Lernziele

Technische Hochschule mannheim

Erklären können, was ein LLM im Kern tut

Den Weg vom Text über Tokens und Embeddings bis zur Vorhersage verstehen

Self-Attention und den Aufbau eines Transformers nachvollziehen

Die Trainingsphasen (Pretraining, SFT, RLHF) unterscheiden

Verstehen, wie bei der Inferenz Text Token für Token generiert wird

Fähigkeiten, Grenzen (z. B. Halluzinationen) und Erweiterungen einordnen

Technische Hochschule
mannheim

# 1 Einordnung & Grundidee

**Was ein Large Language Model im Kern ist**

# Was ist ein LLM?

Technische Hochschule mannheim

**Large Language Model**

Ein neuronales Netz, das natürliche Sprache verarbeitet und erzeugt

**„Large“**

Milliarden von Parametern, trainiert auf riesigen Textmengen

**Basis-Architektur**

Fast alle heutigen LLMs beruhen auf dem Transformer (2017)

**Kernprinzip**

Es sagt das wahrscheinlich nächste Textstück (Token) voraus

# Die Kernaufgabe: das nächste Token vorhersagen

Technische Hochschule mannheim

**Sprachmodell = Wahrscheinlichkeitsverteilung**

Gegeben der bisherige Text, wie wahrscheinlich ist jedes mögliche nächste Token?

`P(nächstes Token | bisheriger Kontext)`

**Autoregressiv**

Das gewählte Token wird angehängt – dann beginnt die Vorhersage von vorn

**Beispiel**

„Die Hauptstadt von Frankreich ist ___“ → sehr wahrscheinlich „Paris“

**Erstaunlich**

Aus dieser einfachen Aufgabe entstehen komplexe Fähigkeiten

# Ein wenig Geschichte

Technische Hochschule mannheim

**Statistische Modelle (n-Gramme)**

Zählen, welche Wortfolgen häufig sind – kurzer Kontext

**RNN / LSTM (2010er)**

Sequenzielle Verarbeitung, Probleme mit langen Abhängigkeiten

**Transformer (2017)**

„Attention Is All You Need“ – parallel, skaliert hervorragend

**GPT, BERT & Nachfolger**

Vortrainieren auf riesigen Korpora, dann anpassen – heutige LLMs

# Klarstellung: kein Nachschlagen, kein Verstehen wie ein Mensch

Technische Hochschule mannheim

**Keine Datenbank**

Ein LLM „schlägt“ nichts nach – Wissen steckt implizit in den Gewichten

**Kein Bewusstsein**

Es erzeugt plausible Muster, es „versteht“ nicht im menschlichen Sinn

**Nur Wahrscheinlichkeiten**

Jede Antwort ist eine Folge wahrscheinlicher Tokens

**Wichtige Konsequenz**

Plausibel klingt nicht immer korrekt (siehe Halluzinationen)

Technische Hochschule
mannheim

# 2 Von Text zu Zahlen

**Tokenisierung, Vokabular und Embeddings**

# Die Verarbeitungs-Pipeline im Überblick

Technische Hochschule mannheim

**Jeder Schritt wandelt die Darstellung um – am Ende steht ein neues Token.**


Für jedes weitere Token wird die Pipeline erneut durchlaufen (autoregressiv). Die nächsten Folien erklären jeden Block (1–7) einzeln.

# Die Pipeline an einem Beispiel (Blöcke 1–4)

Technische Hochschule mannheim

**Eingabe: „Die Hauptstadt von Frankreich ist …“**

**Block 1 · Text**

Deine Eingabe als reine Zeichenkette

**Block 2 · Tokens**

Text wird in Bausteine zerlegt:

`[„Die“, „Haupt“, „stadt“, „ von“, „ Frankreich“, „ ist“]`

**Block 3 · Embedding-Vektoren**

Jedes Token wird zu einer Liste von Zahlen (Position im Bedeutungsraum)

**Block 4 · Transformer (N Schichten)**

Die Tokens verrechnen sich gegenseitig (Attention);

„ist“ sammelt den Kontext ein: hier wird eine Hauptstadt gesucht

# Die Pipeline an einem Beispiel (Blöcke 5–7)

Technische Hochschule mannheim

**Block 5 · Logits**

Für jedes mögliche nächste Token ein roher Punktwert:

`Paris 8.2` · `London 3.1` · `Berlin 2.7` · …

**Block 6 · Wahrscheinlichkeiten**

Softmax macht daraus Prozente (Summe = 1):

`Paris 0.92` · `London 0.03` · `Berlin 0.02` · …

**Block 7 · nächstes Token**

„Paris“ wird ausgewählt und an die Eingabe angehängt

**Und dann von vorn**

Mit „… ist Paris“ startet die ganze Kette erneut – so entsteht Wort für Wort der Text

# Block 2 · Tokens: Text zerlegen

Technische Hochschule mannheim

**Problem**

Ein Modell rechnet mit Zahlen, nicht mit Buchstaben

**Tokens = Textbausteine**

Meist Subwörter – zwischen ganzem Wort und einzelnem Zeichen

**Verfahren (z. B. Byte-Pair-Encoding, BPE)**

Häufige Zeichenfolgen werden zu einem Token zusammengefasst

**Beispiel**

`„Tokenisierung“ → z. B. [„Token“, „isier“, „ung“]`

**Faustregel**

Im Deutschen/Englischen grob 1 Token ≈ 3–4 Zeichen

# Block 2 · Tokens: Vokabular & IDs

Technische Hochschule mannheim

**Vokabular**

Feste Menge aller bekannten Tokens (oft ~30.000 bis >100.000)

**Token-ID**

Jedes Token hat eine eindeutige Nummer (Index im Vokabular)

`Text → Tokens → IDs, z. B. [„Paris“] → [3487]`

**Spezielle Tokens**

Steuertokens wie Satz-/Sequenzende oder Rollentrenner

**Rückweg**

Am Ende werden IDs wieder zu lesbarem Text (Detokenisierung)

# Block 3 · Embedding-Vektoren: Wörter werden zu Punkten

Technische Hochschule mannheim

**Analogie: Koordinaten auf einer Landkarte**

Wie ein Ort durch Länge und Breite beschrieben wird, bekommt jedes Token eine Position – nur in sehr vielen Dimensionen

**Von der bloßen Nummer zum Bedeutungsvektor**

Die Token-ID (z. B. 3487) ist nur eine Nummer – ohne Bedeutung

Eine gelernte Tabelle übersetzt sie in einen Vektor aus vielen Zahlen (z. B. 768–12.288)

**Nähe bedeutet Ähnlichkeit**

„Hund“ liegt nahe bei „Katze“, aber weit weg von „Banane“

Gemessen über den Abstand bzw. Winkel zwischen den Vektoren

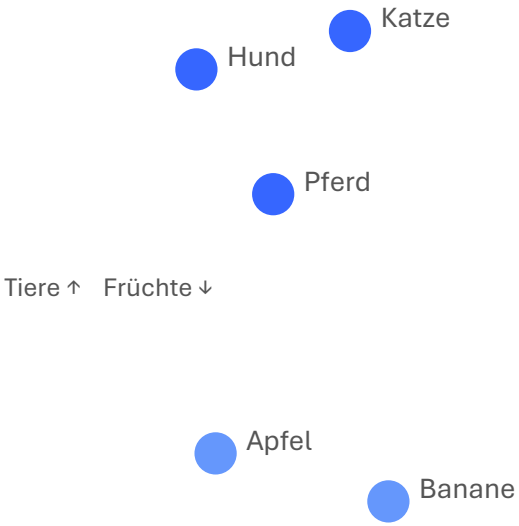
**Gelernt, nicht vorgegeben**

Diese Positionen entstehen im Training – niemand legt sie von Hand fest

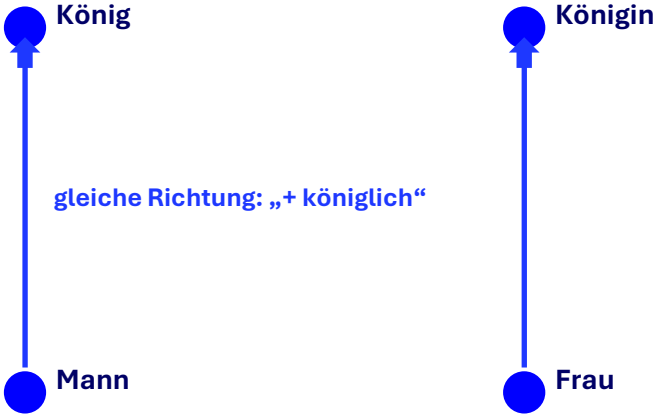
# Block 3 · Der Bedeutungsraum – anschaulich

Technische Hochschule mannheim

## Ähnliches liegt nah beieinander




## Richtungen tragen Bedeutung




**König – Mann + Frau ≈ Königin: die Rechnung ergibt fast genau den Punkt „Königin“.**

Stark vereinfacht (2D). Echte Embeddings haben hunderte bis tausende Dimensionen; im Netz werden sie je nach Kontext verfeinert (z. B. „Bank“: Sitzbank vs. Geldinstitut).

# Block 3 · Reihenfolge (Positionskodierung)

Technische Hochschule mannheim

**Problem**

Attention betrachtet Tokens zunächst ohne Reihenfolge

**Reihenfolge zählt**

„Hund beißt Mann“ ≠ „Mann beißt Hund“

**Lösung**

Positionsinformation wird in die Embeddings eingebracht

**Varianten**

Feste (sinusförmige), gelernte oder relative/rotierende Kodierungen (z. B. RoPE)

Technische Hochschule
mannheim

# 3 Der Transformer

**Self-Attention – das Herzstück moderner LLMs**

# Block 4 · Der Transformer – Überblick

Technische Hochschule mannheim

**Grundbaustein: der Transformer-Block**

Ein LLM stapelt viele identisch aufgebaute Blöcke (z. B. 12 bis >100)

**Zwei Kernkomponenten je Block**

Self-Attention – Tokens beziehen sich aufeinander

Feed-Forward-Netz – verarbeitet jedes Token einzeln weiter

**Stabilisierung**

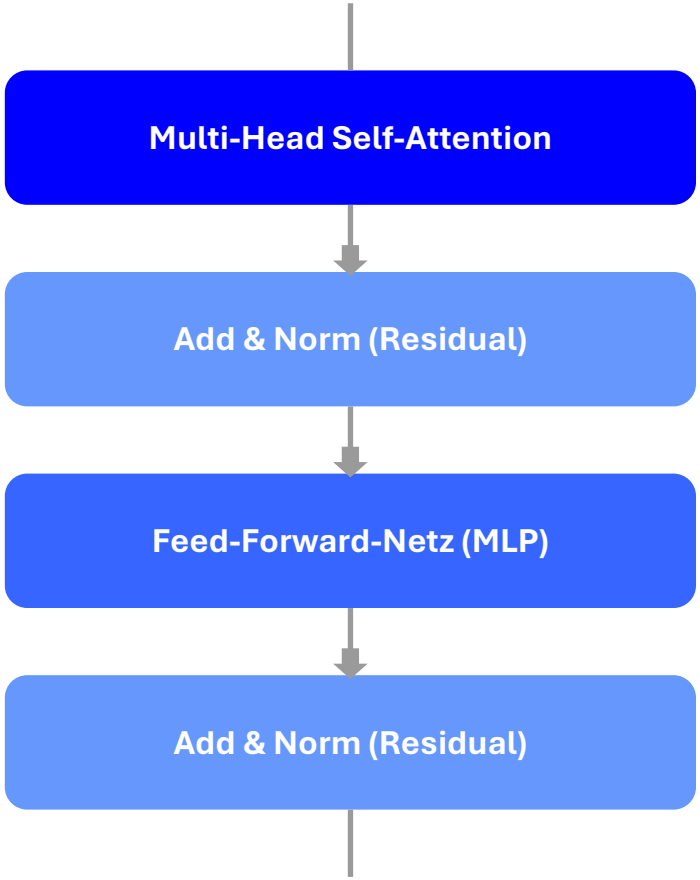
Residual-Verbindungen und Normalisierung in jedem Block

**Parallel statt sequenziell**

Alle Tokens werden gleichzeitig verarbeitet – gut für GPUs

# Aufbau eines Transformer-Blocks

Technische Hochschule mannheim




Eingang: Token-Vektoren

Ausgang: verfeinerte Vektoren
→ nächster Block

**Dieser Block wird N-mal gestapelt; danach folgt die Ausgabeschicht.**

# Self-Attention – die Intuition

Technische Hochschule mannheim

**Frage je Token**

Welche anderen Tokens im Kontext sind für mich gerade wichtig?

**Beispiel**

In „Die Katze jagt sie, weil sie hungrig ist“ – worauf bezieht sich „sie“?

**Mechanismus**

Jedes Token gewichtet alle anderen und zieht deren Information anteilig ein

**Ergebnis**

Kontextabhängige Repräsentationen statt isolierter Wörter

# Query, Key & Value (Q, K, V)

Technische Hochschule mannheim

**Drei gelernte Projektionen je Token**

- Query (Q) – wonach dieses Token sucht
- Key (K) – wofür ein Token „steht“ / auffindbar ist
- Value (V) – die Information, die weitergegeben wird

**Ähnlichkeit**

- Passung von Q und K bestimmt das Aufmerksamkeitsgewicht (Skalarprodukt)

**Analogie**

- Wie eine Suche: Anfrage (Q) trifft auf Schlüssel (K), liefert Werte (V)

# Die Attention-Formel

Technische Hochschule mannheim

**Skalierte Skalarprodukt-Attention**

$$\text{Attention}(Q,K,V) = \text{softmax}\left( Q \cdot K^T / \sqrt{d_k} \right) \cdot V$$

**Schritt für Schritt**

$Q \cdot K^T$ – Ähnlichkeit aller Token-Paare

$\div \sqrt{d_k}$ – Skalierung, hält die Werte stabil

softmax – macht daraus Gewichte, die sich zu 1 summieren

$\cdot V$ – gewichtete Summe der Value-Vektoren

**Ergebnis**

Für jedes Token eine kontextgewichtete Mischung der anderen

# Self-Attention – ein Beispiel

Technische Hochschule mannheim

**Satz: „Die Katze trinkt Milch, weil sie durstig ist.“**

**Frage: Worauf bezieht sich „sie“? → Query = „sie“**

**1 · Ähnlichkeit ($Q \cdot K^T$) von „sie“ zu jedem Wort**

Katze 6.0 · durstig 3.0 · Milch 2.0 · trinkt 1.0

**2 · Skalieren (÷ $\sqrt{d_k}$, hier $d_k = 4$, also ÷ 2)**

Katze 3.0 · durstig 1.5 · Milch 1.0 · trinkt 0.5

**3 · softmax s ($x_i$)→ Gewichte (summieren zu 1)**

Katze $e^{3.0}$ + durstig $e^{1.5}$ + Milch $e^{1.0}$ + trinkt $e^{0.5}$ = 20.068 + 3,482 + 2,718 + 1,649 = 28.935

Katze (20.68/28.935) 69%, durstig (3.482/28.935) 15%, Milch (2.717/28.935) 9%, trinkt (1.649/28.935) 6%

**4 · Gewichtete Summe der Value-Vektoren**

Ergebnis ≈ 0.69·V(Katze) + 0.15·V(durstig) + 0.09·V(Milch) + 0.06·V(trinkt)

**Deutung**

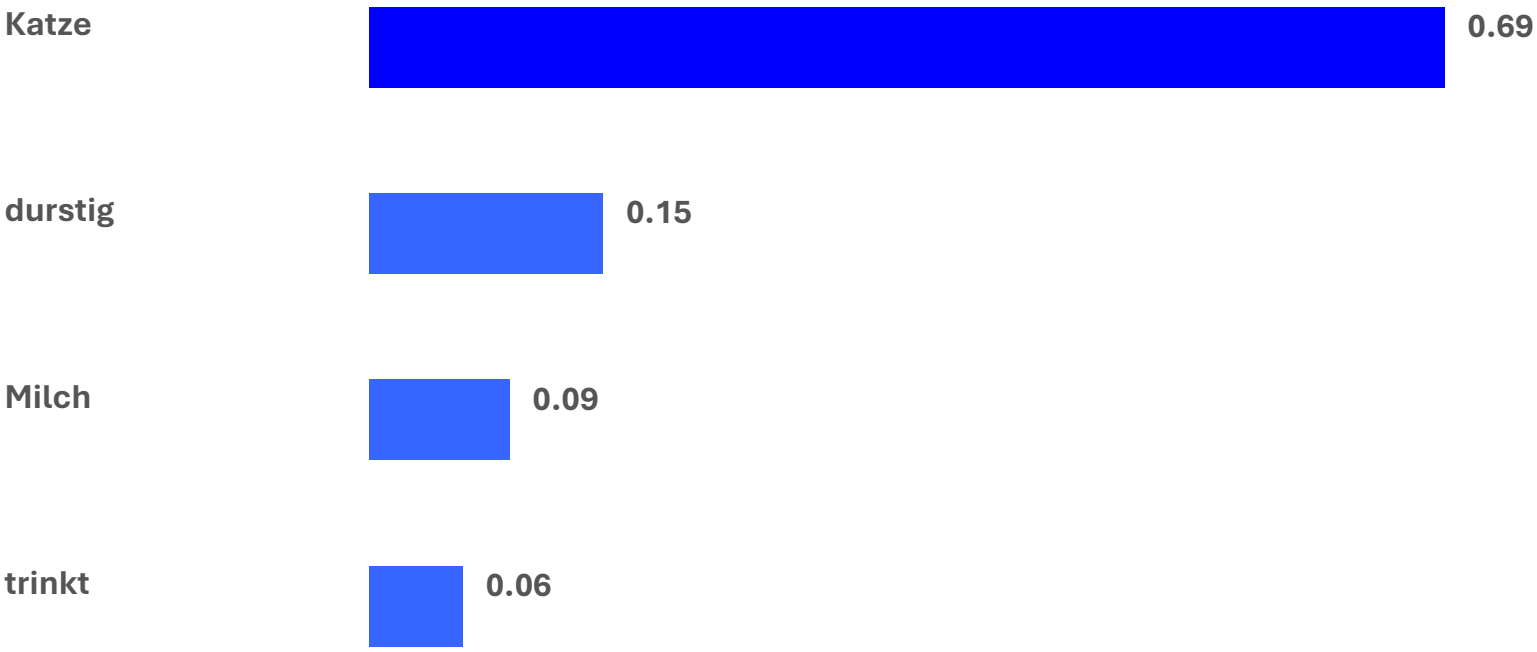
„sie“ zieht v. a. die Bedeutung von „Katze“ ein → das Pronomen ist aufgelöst

$$s\left(x_i\right) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

# Attention-Gewichte veranschaulicht

Technische Hochschule mannheim

**Satz: „Die Katze trinkt Milch, weil sie durstig ist.“ – Wohin richtet sich „sie“?**




„sie“ richtet rund 69 % seiner Aufmerksamkeit auf „Katze“ – so löst Self-Attention den Bezug auf. (Zahlen zur Veranschaulichung.)

# Multi-Head Attention

Technische Hochschule mannheim

**Idee**

Mehrere Attention-„Köpfe“ arbeiten parallel – jeder mit eigenen Q/K/V

**Warum?**

Verschiedene Köpfe lernen verschiedene Beziehungen

Z. B. Grammatik, Bezüge, Fernabhängigkeiten

**Zusammenführung**

Die Ergebnisse aller Köpfe werden kombiniert und weiterverarbeitet

**Kausale Maskierung (Decoder)**

Tokens dürfen nur nach links „schauen“ – nicht in die Zukunft

# Multi-Head Attention – ein Beispiel

Technische Hochschule mannheim

**Satz: „Die Katze trinkt Milch, weil sie durstig ist.“**

**Idee**

Für „sie“ arbeiten mehrere Köpfe gleichzeitig – jeder betont eine andere Beziehung

**Kopf 1 · Bezug („wer?“)**

Schaut stark auf „Katze“ → löst das Pronomen auf

**Kopf 2 · Eigenschaft**

Schaut auf „durstig“ → welcher Zustand wird „sie“ zugeschrieben?

**Kopf 3 · Handlung**

Schaut auf „trinkt“ / „Milch“ → was tut „sie“?

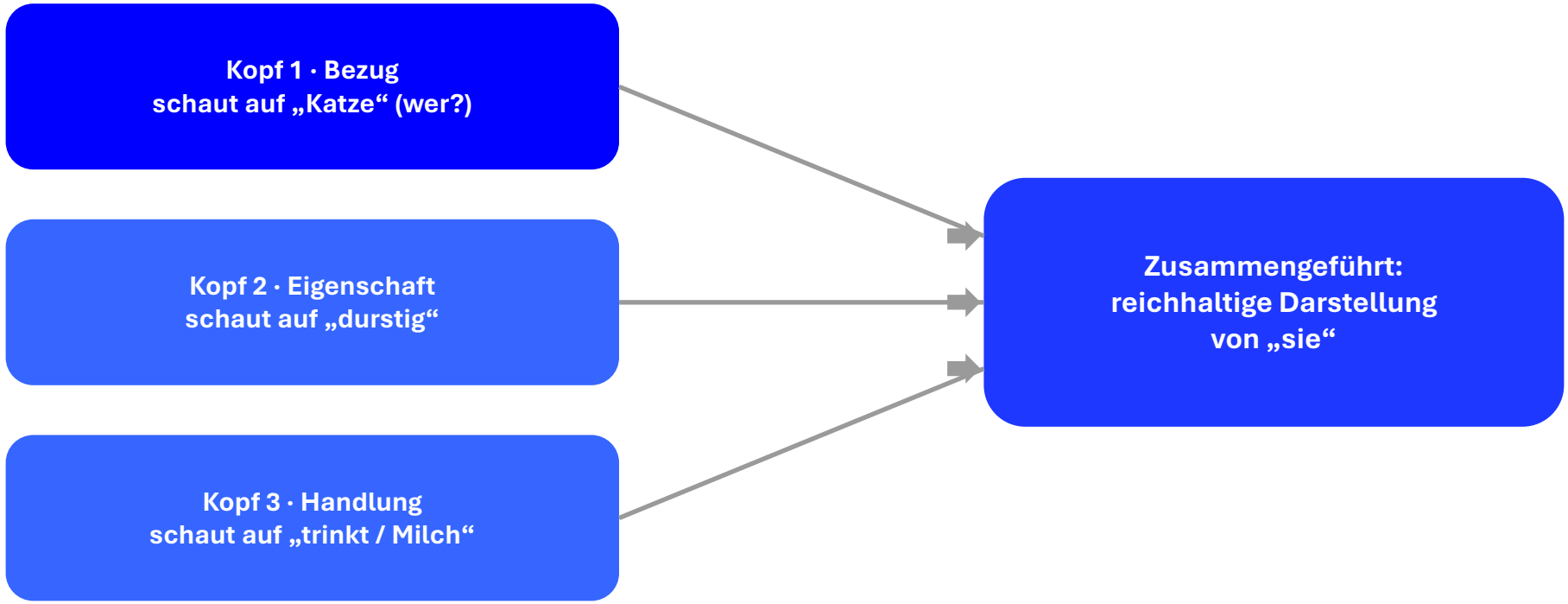
**Zusammenführung**

Alle Sichten kombiniert: „sie“ = die durstige Katze, die Milch trinkt

# Multi-Head Attention – verschiedene Köpfe, verschiedene Sichten

Technische Hochschule mannheim

**Für „sie“ betont jeder Kopf eine andere Beziehung – am Ende werden sie kombiniert.**




Mehr Köpfe = mehr gleichzeitige Beziehungen; „sie“ wird dadurch vielschichtig erfasst.

# Feed-Forward-Netz, Residuen & Normalisierung

Technische Hochschule mannheim

**Feed-Forward-Netz (MLP)**

Verarbeitet jedes Token unabhängig; erweitert und verdichtet die Repräsentation

**Residual-Verbindungen**

Eingang wird zum Ausgang addiert – erleichtert das Training tiefer Netze

**Layer-Normalisierung**

Hält Aktivierungen in stabilen Bereichen

**Zusammenspiel**

Attention (Beziehungen) + MLP (Verarbeitung) je Block, vielfach gestapelt

# Blöcke 5 & 6 · Von der letzten Schicht zur Vorhersage

Technische Hochschule mannheim

**Ausgabeschicht**

Der finale Vektor des letzten Tokens wird auf das Vokabular abgebildet

**Logits**

Ein roher Punktwert je mögliches nächstes Token

**Softmax**

Wandelt Logits in Wahrscheinlichkeiten um (Summe = 1)

$$\mathrm{softmax}(z)_i = e^{z_i} / \Sigma_j e^{z_j}$$

**Ergebnis**

Eine Wahrscheinlichkeitsverteilung über alle Tokens des Vokabulars

Technische Hochschule mannheim

# Encoder, Decoder & Varianten

1

**Decoder-only**

z. B. GPT-Familie

Textgenerierung

kausale Attention

heute dominant

2

**Encoder-only**

z. B. BERT

Verstehen/Klassifik.

sieht ganzen Kontext

keine Generierung

3

**Encoder-Decoder**

z. B. T5

Übersetzen,

Zusammenfassen

Eingang → Ausgang

Technische Hochschule
mannheim

# 4 Training

**Wie ein LLM lernt: Pretraining, Fine-Tuning, RLHF**

Technische Hochschule mannheim

# Die drei typischen Phasen

1

**Pretraining**

Self-supervised

Nächstes Token

Riesige Textmengen

→ Basismodell

2

**Fine-Tuning (SFT)**

Instruktionsdaten

Aufgaben & Antworten

Verhalten formen

→ hilfsbereit

3

**Alignment (RLHF)**

Menschliche

Präferenzen

Reward-Modell

→ nützlich & sicher

# Pretraining: das Fundament

Technische Hochschule mannheim

**Selbstüberwacht (self-supervised)**

Kein manuelles Labeln – der Text liefert die „Antwort“ selbst (das nächste Token)

**Aufgabe**

Sage das nächste Token vorher; vergleiche mit dem tatsächlichen

**Daten**

Enorme Mengen: Webtexte, Bücher, Code, … (Billionen Tokens)

**Ergebnis**

Ein Basismodell mit breitem Sprach- und Weltwissen – aber noch „roh“

# Wie das Lernen funktioniert (vereinfacht)

Technische Hochschule mannheim

**Verlustfunktion (Loss)**

Misst, wie falsch die Vorhersage war (Cross-Entropy)

**Backpropagation**

Berechnet, wie jedes Gewicht zum Fehler beiträgt (Gradient)

**Gradientenabstieg**

Gewichte werden in kleinen Schritten in Richtung kleinerer Fehler angepasst

**Milliardenfach wiederholt**

Über viele Durchläufe entstehen die gelernten Muster

**Wo steckt das „Wissen“?**

Verteilt in den Milliarden Gewichten – nicht als abrufbare Fakten

# Skalierung: Parameter, Daten, Rechenleistung

Technische Hochschule mannheim

**Drei Stellschrauben**

Modellgröße (Parameter), Datenmenge, Rechenaufwand (Compute)

**Scaling Laws**

Mehr von allem verbessert die Leistung recht vorhersagbar

**Balance**

Parameter und Datenmenge sollten zueinander passen (vgl. Chinchilla)

**Kosten**

Großes Pretraining ist extrem rechen- und energieintensiv

# Fine-Tuning & Instruktionstuning

Technische Hochschule mannheim

**Vom Basismodell zum Assistenten**

Das rohe Modell vervollständigt nur Text – es „antwortet“ noch nicht gezielt

**Supervised Fine-Tuning (SFT)**

Training mit Beispiel-Dialogen: Anweisung → gewünschte Antwort

**Effizientes Fine-Tuning**

Methoden wie LoRA passen nur wenige Zusatzgewichte an – ressourcenschonend

**Ergebnis**

Ein Modell, das Instruktionen folgt und im Dialogformat antwortet

# RLHF – Ausrichtung an menschlichen Präferenzen

Technische Hochschule mannheim

**Ziel**

Antworten hilfreicher, ehrlicher und sicherer machen

**Reinforcement Learning from Human Feedback**

Menschen bewerten/vergleichen Antworten

Ein Reward-Modell lernt diese Präferenzen

Das LLM wird darauf optimiert, bevorzugte Antworten zu geben

**Neuere Varianten**

Direkte Präferenzoptimierung (DPO) u. a. vereinfachen den Prozess

**Wirkung**

Formt Ton, Format und Sicherheitsverhalten des Modells

# Emergente Fähigkeiten

Technische Hochschule mannheim

**Beobachtung**

Ab bestimmter Größe treten Fähigkeiten auf, die kleine Modelle nicht zeigen

**Beispiele**

Mehrschrittiges Schließen, Übersetzen, einfaches Rechnen, Code

**In-Context-Learning**

Neue Aufgaben allein aus Beispielen im Prompt lernen – ohne Nachtraining

**Einordnung**

Wie „emergent“ diese Sprünge wirklich sind, ist wissenschaftlich umstritten

Technische Hochschule
mannheim

# 5 Inferenz

**Wie beim Betrieb Text entsteht**

# Block 7 · Das nächste Token – und dann von vorn

Technische Hochschule mannheim


Ende, wenn ein Stopp-Token erzeugt wird oder die Längengrenze erreicht ist.

# Von Logits zu Wahrscheinlichkeiten – die Temperatur

Technische Hochschule mannheim

**Softmax macht Wahrscheinlichkeiten**

Aus den Logits wird eine Verteilung über alle Tokens

**Temperatur T steuert die „Kreativität“**

Niedrig (→0): spitz, deterministischer, konservativ

Hoch (>1): flacher, vielfältiger, riskanter

**Intuition**

T skaliert die Logits vor dem Softmax ($z_i$ / T)

**Praxis**

Faktenfragen eher niedrig, kreatives Schreiben eher höher

Technische Hochschule
mannheim

# Sampling-Strategien

1

**Greedy / Beam**

Immer wahrschein-
lichstes Token
Vorhersehbar
Kann eintönig sein

2

**Top-k**

Nur die k besten
Tokens zulassen
dann zufällig
auswählen

3

**Top-p (Nucleus)**

Kleinste Menge mit
Wahrscheinlichkeit ≥ p
flexibel
weit verbreitet

# Kontextfenster & Effizienz

Technische Hochschule mannheim

**Kontextfenster**

Maximale Zahl an Tokens, die das Modell gleichzeitig berücksichtigt

Umfasst Prompt und erzeugte Antwort

**Grenzen**

Rechenaufwand der Attention wächst stark mit der Länge

**KV-Cache**

Zwischenergebnisse werden gespeichert, damit die Generierung effizient bleibt

**Folge**

Sehr lange Eingaben kosten mehr Zeit und Speicher

# Prompt, Rollen & In-Context-Learning

Technische Hochschule mannheim

**Der Prompt ist die einzige Steuerung zur Laufzeit**

Das Modell selbst bleibt unverändert – nur der Kontext ändert sich

**Rollen**

System (Verhalten), User (Anliegen), Assistant (Antwort)

**In-Context-Learning**

Beispiele im Prompt (Few-Shot) steuern das Verhalten ohne Nachtraining

**Konsequenz**

Guter Kontext = bessere Antworten (Prompt-Design zählt)

Technische Hochschule
mannheim

# 6 Fähigkeiten, Grenzen & Erweiterungen

**Was LLMs können – und wie man sie ergänzt**

# Was LLMs gut können

Technische Hochschule mannheim

Sprache verstehen und flüssig erzeugen

Zusammenfassen, umformulieren, übersetzen

Code schreiben, erklären und umbauen

Aus wenigen Beispielen neue Aufgaben übernehmen (Few-Shot)

Strukturierte Ausgaben und Schritt-für-Schritt-Erklärungen

Als flexible Schnittstelle zwischen Mensch und Software dienen

# Grenzen & Fehlerquellen

Technische Hochschule mannheim

**Halluzinationen**

Plausibel klingende, aber falsche Aussagen

**Kein echtes Verständnis / keine Absicht**

Musterbasierte Vorhersage, keine Bedeutung im menschlichen Sinn

**Wissensschnitt**

Kein Wissen über Ereignisse nach dem Trainingsende

**Bias**

Verzerrungen aus den Trainingsdaten werden reproduziert

**Schwächen**

Exaktes Rechnen, Logik über viele Schritte, verlässliche Quellen

# Warum entstehen Halluzinationen?

Technische Hochschule mannheim

**Das Ziel ist Plausibilität, nicht Wahrheit**

Das Modell wählt wahrscheinliche Fortsetzungen – nicht geprüfte Fakten

**Keine eingebaute Faktenprüfung**

Es „weiß“ nicht, ob eine Aussage stimmt

**Lücken werden gefüllt**

Fehlt Wissen, erzeugt das Modell oft trotzdem eine flüssige Antwort

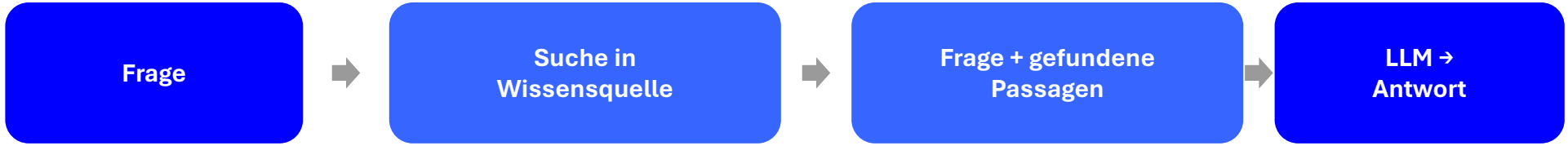
**Gegenmittel**

Quellen/Kontext liefern (RAG), prüfen lassen, kritisch gegenlesen

# Erweiterung: Retrieval-Augmented Generation (RAG)

Technische Hochschule mannheim

**Aktuelles/spezifisches Wissen wird zur Laufzeit beigesteuert – gegen Halluzinationen.**




Vorteil: Quellen sind belegbar und aktualisierbar, ohne das Modell neu zu trainieren.

# Werkzeuge & Agenten

Technische Hochschule mannheim

**Tool-Use / Function Calling**

Das Modell ruft externe Funktionen auf (Rechner, Suche, Datenbank, APIs)

**Warum?**

Gleicht Schwächen aus: exakte Berechnung, aktuelle Daten, verlässliche Aktionen

**Agenten**

Das LLM plant mehrere Schritte, nutzt Werkzeuge und reagiert auf Ergebnisse

**Grenze**

Mehr Autonomie erfordert mehr Kontrolle, Prüfung und Sicherheitsschranken

Technische Hochschule mannheim

# Anpassen: Prompting vs. RAG vs. Fine-Tuning

1

**Prompting**

Kein Training

Kontext im Prompt

Schnell & flexibel

Für Verhalten/Format

2

**RAG**

Externes Wissen

Belegbar & aktuell

Ohne Neutraining

Für Fakten/Dokumente

3

**Fine-Tuning**

Gewichte anpassen

Aufwändiger

Für Stil/Spezialfälle

Wissen nur bedingt

# Multimodalität (Ausblick)

Technische Hochschule mannheim

**Über Text hinaus**

Moderne Modelle verarbeiten auch Bilder, Audio oder Code gemeinsam

**Gemeinsames Prinzip**

Auch andere Daten werden in Tokens/Vektoren überführt und vom Transformer verarbeitet

**Beispiele**

Bildbeschreibung, Diagramme lesen, Sprache verstehen und erzeugen

**Kern bleibt gleich**

Vorhersage der nächsten Einheit auf Basis des Kontexts

Technische Hochschule
mannheim

# Zusammenfassung

## Das mentale Modell

# Das mentale Modell in sieben Sätzen

Technische Hochschule mannheim

Ein LLM sagt wiederholt das wahrscheinlich nächste Token voraus

Text wird in Tokens zerlegt und als Vektoren (Embeddings) dargestellt

Der Transformer nutzt Self-Attention, um Bezüge im Kontext zu gewichten

Viele gestapelte Blöcke erzeugen am Ende eine Verteilung über Tokens

Trainiert wird in Phasen: Pretraining, Fine-Tuning, Alignment (RLHF)

Bei der Inferenz entsteht Text autoregressiv, gesteuert durch Sampling

Wissen steckt in den Gewichten – plausibel ist nicht immer wahr

# Verbreitete Missverständnisse

Technische Hochschule mannheim

**„Das LLM schlägt Antworten nach.“**

Nein – es rechnet Wahrscheinlichkeiten aus gelernten Mustern

**„Es denkt oder versteht wie ein Mensch.“**

Nein – es gibt keine Bedeutung oder Absicht im menschlichen Sinn

**„Was flüssig klingt, ist korrekt.“**

Nein – Sprachqualität ist kein Beweis für Faktenrichtigkeit

**„Mehr Parameter = immer besser.“**

Nicht unbedingt – Datenqualität, Training und Anpassung zählen ebenso

# Ressourcen zum Weiterlernen

Technische Hochschule mannheim

Vaswani et al. (2017): „Attention Is All You Need“ (Transformer)

J. Alammar: „The Illustrated Transformer“ (anschauliche Einführung)

3Blue1Brown: Videoreihe zu neuronalen Netzen und Transformern

A. Karpathy: „Let’s build GPT“ (Video, Code von Grund auf)

Bishop / Goodfellow et al.: Lehrbücher zu (Deep) Machine Learning

Technische Hochschule
mannheim

**Fragen?**

**Wie funktioniert ein LLM – vom Token bis zur Antwort**